A priori Knowledge from Non-Examples

Diplomarbeit im Fach Bioinformatik
vorgelegt am **29. März 2007** von **Fabian Sinz**

Fakultät für Kognitions- und Informationswissenschaften
Wilhelm-Schickard-Institut für Informatik der Universität Tübingen
in Zusammenarbeit mit
Max-Planck-Institut für biologische Kybernetik Tübingen

Betreuer:
Prof. Dr. Bernhard Schölkopf (MPI für biologische Kybernetik)
Prof. Dr. Andreas Schilling (WSI für Informatik)
Prof. Dr. Hanspeter Mallot (Biologische Fakultät)

Erklärung

Die folgende Diplomarbeit wurde ausschließlich von mir (Fabian Sinz) erstellt. Es wurden keine weiteren Hilfmittel oder Quellen außer den Angegebenen benutzt.

Tübingen, den 29. März 2007, Fabian Sinz

Contents

1 Introduction **4**
- 1.1 Foreword . . . 4
- 1.2 Notation . . . 5
- 1.3 Machine Learning . . . 7
  - 1.3.1 Machine Learning . . . 7
  - 1.3.2 Philosophy of Science in a Nutshell - About the Impossibility to learn without Assumptions . . . 11
  - 1.3.3 General Assumptions in Bayesian and Frequentist Learning . . . 14
  - 1.3.4 Implicit prior Knowledge via additional Data . . . 17
- 1.4 Mathematical Tools . . . 18
  - 1.4.1 Reproducing Kernel Hilbert Spaces . . . 18
- 1.5 Regularisation . . . 22
  - 1.5.1 General regularisation . . . 22
  - 1.5.2 Data-dependent Regularisation . . . 27

2 The Universum Algorithm **32**
- 2.1 The Universum Idea . . . 32
  - 2.1.1 VC Theory and Structural Risk Minimisation (SRM) in a Nutshell . . . 32
  - 2.1.2 The original Universum Idea by Vapnik . . . 35
- 2.2 Implementing the Universum into SVMs . . . 37
  - 2.2.1 Support Vector Machine Implementations of the Universum . . . 38
    - 2.2.1.1 Hinge Loss Universum . . . 38
    - 2.2.1.2 Least Squares Universum . . . 46
    - 2.2.1.3 Multiclass Universum . . . 48
  - 2.2.2 Related work . . . 51
- 2.3 Contradictions and Margin . . . 52
- 2.4 Analysis of the Hinge and Least Squares Universum Implementations . . . 58
  - 2.4.1 Hard Margin Universum and Subspace Projection . . . 58

2.4.2 Least Squares Universum, Kernel Oriented Principal Component Analysis and Kernel Fisher Discriminant Analysis . . . 65
2.5 On the Relation between the Universum and Maximum Entropy Distributions . . . 74

3 Applications and Experiments 88
3.1 Optical Character Recognition . . . 88
3.2 Text classification . . . 93
3.3 Universum for Neural Applications . . . 99
3.3.1 Brain Computer Interfaces (BCI) . . . 99
3.3.2 Stimulus Reconstruction from Spikes . . . 113

4 Discussion, Conclusion and Future Work 121
4.1 Discussion and Conclusion . . . 121
4.2 Future Work . . . 124
4.3 Definitions . . . 127

Chapter 1

Introduction

1.1 Foreword

The subject of this diploma thesis is the utilisation of ”non-examples” in a supervised machine learning task. Choosen carefully, these examples can carry valuable prior knowledge about the underlying learning problem and can increase the performance of the learning algorithm. The reason why the additional data points are called ”non-examples” is that they do not share the same distribution with the other data examples. While the usage of additional unlabelled examples from the same distribution has been heavily investigated in the last decade, the utilisation of data from a different distribution has almost been ignored. Although the idea had been mentioned by Vladimir Vapnik in his book *The Nature of Statistical Learning Theory* [99] over ten years ago, only recently the first algorithm based on that idea was published [103]. However, since the underlying idea by Vapnik is rather conceptual, the authors of [103] used several relaxations in order to arrive at a formulation of a learning algorithm that could be efficiently implemented. This strategy bears the risk that the augmented performance of the learning algorithm is due to effects introduced with the relaxation and can therefore not be imputed on the initial idea. This is the reason why this thesis pursues two approaches: the analysis of the algorithmic implementation by [103] and some variations of it, and a theoretical comparison of the initial idea by Vapnik with the maximum entropy principle in empirical inference.

This thesis is composed of four parts. The first part gives a rather general introduction on machine learning. Since this field is very wide and polymorphic, this introduction cannot and does not intend to be complete or rigorous. Its goal is to serve as a very general overview and to specificly stress a fundamental problem in machine learning: the impossibility to learn without prior assumptions. In fact, as will be pointed out in a brief section about philosophy of science, this fundamental impossibility is not a genuine property of machine learning but rather a problem of all empirical sciences. Since there is no way to escape this fact, the more modest strategy must be to smartly incorporate prior knowledge about a problem into a learning algorithm. The motivation for the first section is to show that the utilisation of additional unlabelled examples, from the same distribution or not, is a generic and elegant way to make prior assumptions.

Chapters 2 and 3 represent the analytical and experimental work done by the author. Chapter 2 comprises the theoretical analysis. At its beginning, the initial idea by Vapnik and the algorithmic implementation of [103] are introduced in sections 2.1 and 2.2.1.1. In sections 2.2.1.2 and 2.2.1.3, the algorithm of [103] is extended. The remaining sections of chapter 2 analyse the original algorithm and the introduced variations as well as the original idea by Vapnik.

The third part applies the algorithms to real world data. Two neural applications, namely *Brain Computer Interfaces (BCI)* and *stimulus prediction from spike trains*, are investigated in greater detail. For the section about BCIs, actual experiments with a BCI were carried out in order to explore the possibility of recording such an additional dataset of non-examples.

The last chapter summarises the thesis, draws conclusions from the theoretical analysis and the empirical experiments and discusses possible future work.

Throughout this thesis, little marginal notes are placed next to the main text. Their intended purpose is to give a rough picture of the line of though underlying the text.

The recentness of this branch of machine learning offers the great chance of exploring new possibilities and getting new insights, but restricts the work of half a year to be of interim character at the same time. However, the author hopes that his thesis can at least quicken the insterests of the reader a little bit for this new topic.

1.2 Notation

In order to increase the readability of formulae, I stick with some notation rules. In many cases these rules agree with the notational conventions in the machine learning community.

- Scalar values, functions, general variables and constants are written as Latin or Greek letters. Usually variables like x, y, z are taken from the last letters in the Roman alphabet, constants like a, b, c from the first. There is no such rule for variables or constants denoted by Greek letters. In almost all cases, data points are denoted by x or z and their labels by y.

- Functions use parentheses, functionals or operators use brackets (if they use any). E.g. $f(x)$ is the value of the function f at x and $R[f]$ is the value of the functional R at f.
- Matrices are denoted by upper case bold Latin letters (e.g. $\mathbf{A}$, $\mathbf{B}$).
- To stress that something is an element of a vector space, it is written in bold font (e.g. $\mathbf{x}$ is a vector).
- Sometimes a function f is also written as $f(\cdot)$. The "$\cdot$" indicates that the argument is left open. This is especially useful in situations in which a function takes several arguments. For example, if k is a function taking two arguments, then $k(x, \cdot)$ is the function k with the first argument fixed to x and the second argument left open. So, k is reduced to a function $k(x, \cdot)$ of one argument.
- Sets are denoted with calligraphic font (e.g.$\mathcal{X}$, $\mathcal{Y}$) or in Gothic type (e.g. $\mathfrak{U}$, $\mathfrak{L}$). Some special, frequently occuring sets are marked by reserved symbols. The domain of a learning algorithm is always denoted by $\mathcal{D}$, its range by $\mathcal{R}$. The letter $\mathfrak{L}$ is used for a specific instance of a labelled set, $\mathfrak{Y}$ denotes only the labels of $\mathfrak{L}$. The symbol $\mathfrak{X}$ is either denotes a general training set, labelled or unlabelled, or merely the input points x of a labelled set $\mathfrak{L}$. The exact meaning should always be clear from the context. Finally, the symbol $\mathfrak{U}$ is used for a Universum set.
- Fields, rings and vector spaces are written in the usual notation $\mathbb{R}$, $\mathbb{Z}$ or $\mathbb{R}^n$.
- As in [32], random variables are denoted by the same letter as the values of the random variable, but in sans-serif font. So X is the random variable and x is one of its values.
- The expectation operator is denoted in bold sans-serif font. The index indicates the distribution the expectation is taken over. Here, the notation convention for random variables applies, too, e.g. the expectation of $f(\mathsf{X})$ with respect to $\mathbf{\mathsf{P}}(\mathsf{X}|\mathsf{Y},\mathsf{Z})$ is written as $\mathbf{\mathsf{E}}_{\mathsf{X}|\mathsf{Y},\mathsf{Z}}[f(\mathsf{X})]$. Whenever the subscript of $\mathbf{\mathsf{E}}$ is omitted, the distribution the expectation is taken over should be clear from the context.
- When a random variable is fixed to some value, it is again written in Latin font, e.g. $\mathbf{\mathsf{P}}(\mathsf{X}|\mathsf{Y}=y,\mathsf{Z})$ is written as $\mathbf{\mathsf{P}}(\mathsf{X}|y,\mathsf{Z})$. The same rules apply to the expectation operator.

1.3 Machine Learning

1.3.1 Machine Learning

The subject of machine learning research is to automate the process of inference. It investigates whether and by which means algorithms can efficiently extract rules or patterns from a finite amount of data which extrapolate to unseen parts of the data domain with low error. The process of inferring such rules is called *learning*. The set of data points the algorithm is learning on is usually called *training data* or *training examples*. In many cases, the elements of this set are additionally equipped with values from another set $\mathcal{R}$, called *labels* or *target values*. These labels indicate the *category* or *class* of a single data point. The purpose of most learning algorithms is to predict the labels on data points for which the label is not known. Those data points are referred to as *unseen* or *unlabeled*.

Machine learning as mapping from data to functions

Thus, a machine learning algorithm A may be seen as a mapping from the power set $\wp$ of a certain data domain $\mathcal{D}$, possibly equipped with labels from $\mathcal{R}$, into the set of functions $\mathcal{F} := \{f | \, f : \mathcal{D} \to \mathcal{R}\}$ from that domain to $\mathcal{R}$:

$$A : \wp(\, (\mathcal{D} \times \mathcal{R}) \cup \mathcal{D}) \quad \to \quad \mathcal{F}. \tag{1.1}$$

Among other features, the training set $\mathfrak{X} \in \wp(\, (\mathcal{D} \times \mathcal{R}) \cup \mathcal{D})$ that is supplied to the algorithm for learning and the range $\mathcal{R}$ of the learnt function, can be used to categorize learning algorithms. Figure 1.1 shows a simple taxonomy of learning algorithms.

Machine learning as an inverse problem

As (1.1) already suggests, learning can be seen as an inverse problem. Given a set of input-output pairs sampled from an unknown function in a process that might be afflicted with noise, the goal of a learning algorithm is to reconstruct the underlying function that generated these data points. There are two major paradigms in machine learning to address this problem.

Two major machine learning paradigms

Bayesian learning models the distribution of the data via a *prior*, a distribution encoding prior knowledge about the learning problem, and a *likelihood function* that establishes a link between the data points and the noisy function values via the application of *Bayes Rule*. The result of Bayesian learning is a distribution conditioned on the training data. This distribution is called *posterior*. The actual function is then constructed by using the estimated posterior. In most cases the likelihood function depends on parameters θ that are adjusted during the learning procedure. This adjustement can either be done by imposing another prior on the parameters and averaging over all possible values of θ with respect to the prior or by choosing a fixed value for θ according to some criterion like *maximum likelihood* (ML), *maximum a posteriori* (MAP) or other criteria.

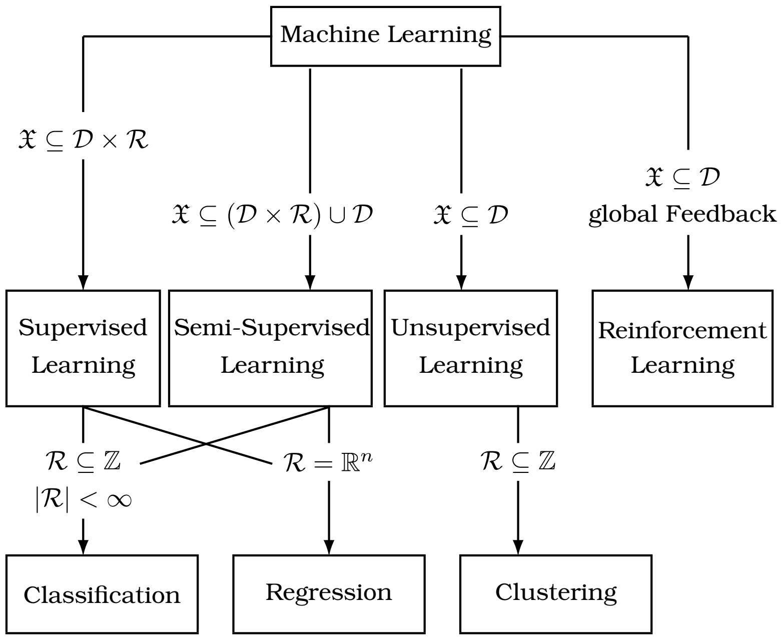

Machine Learning
$\mathfrak{X} \subseteq \mathcal{D} \times \mathcal{R}$
$\mathfrak{X} \subseteq (\mathcal{D} \times \mathcal{R}) \cup \mathcal{D}$
$\mathfrak{X} \subseteq \mathcal{D}$
$\mathfrak{X} \subseteq \mathcal{D}$ global Feedback
Supervised Learning
Semi-Supervised Learning
Unsupervised Learning
Reinforcement Learning
$\mathcal{R} \subseteq \mathbb{Z}$ $|\mathcal{R}| < \infty$
$\mathcal{R} = \mathbb{R}^n$
$\mathcal{R} \subseteq \mathbb{Z}$
Classification
Regression
Clustering

Figure 1.1: Very coarse taxonomy of learning algorithms with respect to the type of data points contained in the set $\mathfrak{X}$ that is fed to the algorithm for learning and the range $\mathcal{R}$ of the learnt function.

Statistical or *frequentist* learning algorithms aim at directly constructing the function without estimating a distribution in between. Figure 1.2 shows a simple example for a learning problem. The task is to discriminate between two Gaussian distributions. The left figure shows the underlying distribution while the right figure depicts the associated decision function.

The inverse problem is ill-posed

There are two major reasons why this inverse problem is ill-posed. On the one hand, most learning problems are noisy. Therefore it might happen that the same input occurs twice in the same data set with different target values for each instance, making it impossible to fit the target values exactly. If the target values are noisy, fitting an exact function is not even desirable since the goal is to model the underlying function without noise. Intuitively the algorithm should not trust the data too much. This problem can be dealt with in the case of non-Bayesian learning by not requiring the algorithm to fit the data exactly

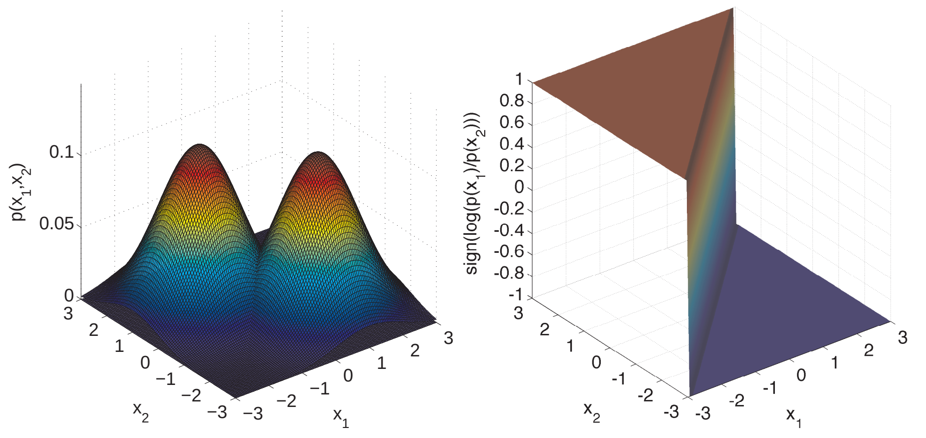


Figure 1.2: Simple example of a learning problem: The task is to discriminate between two Gaussian distributions (left). Once the true class conditioned distributions $\mathsf{P}_{\mathsf{X}|\mathsf{Y}=1}(x)$ and $\mathsf{P}_{\mathsf{X}|\mathsf{Y}=-1}(x)$ are known, a decision function is given by $\operatorname{sign}\left(\log\left(\frac{\mathsf{P}_{\mathsf{X}|\mathsf{Y}=1}(x)}{\mathsf{P}_{\mathsf{X}|\mathsf{Y}=-1}(x)}\right)\right)$ (right). Learning this decision function can either be addressed by identifying this function directly or modeling the class conditional distributions first.

but only as well as possible. The quality of the fit is usually measured by a loss function. There is a direct correspondence between loss functions and noise distributions, i.e. the choice of a certain loss function makes implicit assumptions about the noise (see 1.3.3). Bayesian learning usually deals with noisy target values by additionally estimating a noise distribution or encoding the noise in the likelihood function. In order to avoid the situation in which the learning algorithm just declares everything as noise, assumptions have to be made about the noise and the data distribution. Usually it is assumed that the noise is uncorrelated and independent, meaning that the noise covariance matrix is diagonal and that the noise on one data point does not depend on the noise on other data points. The assumptions about the data are encoded in the prior.

Machine learning problems are afflicted with noise

On the other hand, especially in the case where the set of sampled data points is small, there might be many possible functions or likelihood parameters that fit the data. In that case the inverse problem is under-determined and therefore there is no unique solution. To make the solution unique, an additional criterion for the resulting function is introduced, i.e. the problem is *regularised*. The regulariser encodes assumptions about the underlying function. Its type is usually determined by the choice of the *induction principle*, that is a rule which tells when to prefer one function over another if both are valid solutions to the inverse problem. For example, an assumption implicitly made by

Machine learning optimises a regulariser almost all regularisers is that the function of interest has to be smooth. Many people interpret this as an application of Occam's[1] razor, which basically states that explanations should not be unnecessarily complex. It can also be seen as a continuity-like criterion, stating that similar input values should have similar output values.

Machine learning minimises a loss function

As mentioned above, when dealing with noisy data, an exact interpolation of the data used for learning is often hardly possible and not even desirable. Hence, the learning algorithm only aims at fitting the target values as well as possible. The quality of the learnt function is measured by a loss function that assigns a loss value to each pair of predicted and expected target value. One simple example of a loss function is the so called *zero-one loss* which is just the fraction of misclassified examples. A very common loss function for regression is the quadratic loss used in least squares regression. Since the loss function encodes an important aspect of the learning problem by specifying the "cost" for deviations from the exact fit of the data, it should be incorporated into the learning or prediction process. In Bayesian learning it is used in the prediction step by choosing that element of $\mathcal{R}$ that minimises the expected loss with respect to the posterior. If the assumptions made by the prior about the underlying distribution are correct, this is the optimal strategy for optimising the expected loss on unseen data points. However, choosing the prior is not easy and in most cases computational considerations outweigh modeling issues. Frequentist learning mostly incorporates the loss function into the learning process itself by directly minimising the considered loss. Taking the regulariser into account, it is possible to prove bounds for the deviation of the expected loss on all data points from the empirical error on the data points used for training the algorithm.

Machine learning as an optimisation of regulariser and loss function

In general, the process of learning involves the optimisation of an optimality criterion. In unsupervised learning this criterion is a function on the completely unlabelled data points which encodes desired properties of the solution. One such function could be a loss measure between the original data and a lower dimensional representation of it. The goal in supervised learning is to find a function that has low expected loss with respect to the distribution of the data. The optimality criterion in frequentist supervised learning algorithms is therefore to jointly minimise the error on the available set of data points and the regulariser that promises better generalisation via error bounds.

If all involved objects have a specified distribution, there is no optimisation in Bayesian learning during the training stage. In this case the desired posterior distribution can simply be calculated by Bayes rule. The expected loss is then minimised in the prediction step. If not all parameters have a specified distribution, a set of parameters is chosen that is optimal according to a criterion like maximum likelihood,

[1] *William of Occam, approx. 1285 - 1347*

maximum a posteriori or maximum entropy (see 1.3.3). This procedure involves a optimisation step.

1.3.2 Philosophy of Science in a Nutshell - About the Impossibility to learn without Assumptions

The question whether and how general laws can be inferred from a finite amount of observations is almost as old as science itself. The following paragraphs are a brief overview over the parts of philosophy of science that are relevant to this thesis and do not claim completeness. The main statement of this section is that there is no empirical law or rule that is provably true and that even the procedure of inferring this rule from data needs prior assumptions about the underlying process in order to have predictive power. Since machine learning is the automated version of inferring rules from finite data, this implies that machine learning algorithms must make prior assumptions. This necessity motivates the use of non-examples as one possible way of incorporating prior knowledge.

Induction in the philosophy of Aristotle

The Greek philosopher *Aristotle*[2], who is regarded as the founder of logic, already knew that when proving a statement basic true propositions are necessary for its deduction. These basic propositions, which he called *archai*, are gained by a process called *epagoge.* Cicero[3] translated the Greek word "epagoge" into the Latin word "inductio", the etymological root of the English word "induction". The process of epagoge might seem a little odd from the viewpoint of current philosophy of science: Aristotle believed that general concepts are actually contained in everything that can be observed. Therefore it is possible to obtain generally true sentences by observation and commemoration, meaning that the archai are directly evident from observations (see [48]). It is clear that this neither explains the emergence of generally true propositions nor justifies their truth.

The scepticism of Sextus Empiricus

A philosophical position much closer to the current view is that of *Sextus Empiricus*[4] . He argued that it is not possible to conclude a general statement from a finite amount of observations since there can always be exceptions to this statement that are not included in the observations. Unless it is possible to check all instances, one cannot rule out the possibility of an exception and therefore the truth of the inferred general statement cannot be taken for granted. The considerations of Sextus Empiricus do not hold, of course, for a general statement being true due to its logical structure. Philosophy of science is concerned with empirical sentences since these are the only ones that carry information. If someone utters that it is raining outside or it is not raining

[2]Aristotle: 384-322 B.C.
[3]Cicero: 106-43 B.C.
[4]Sextus Empiricus: 200-250 AD

outside at some place in the world, then the truth of his statement is immediately evident. However, he does not convey anything informative about the real world, in particular he did not say anything about the current weather. But as soon as a general statement contains any information, the arguments of Sextus Empiricus apply.

Inductive inference in David Hume's philosophy

Another famous philosopher, concerning himself with the question whether a finite amount of observations can justify the truth of a general statement, was *David Hume*[5] [43, 44]. He split the problem of inductive inference into two: a logical and an empirical one [48]. The latter asks why humans expect that the experiences made in the past will match experiences in the future, or, in other words, how humans justify the use of past experience as a basis for predictions about the future. His answer is that once humans have observed two events that are "constantly conjoined", meaning they always occur together, they infer a causal relation between the two. So, in some sense, we are simply used to both events appearing together and thus assume that one causes the other. This does not justify the truth of a general statement inferred by induction. The question of justification is addressed in Hume's logical problem, which is the question whether it is logically justified to deduce future events from past experience. Keuth [48] further splits this problem into the question about the possibility of an *analytic induction principle* and the possibility of a *synthetic induction principle*. The terms "analytic" and "synthetic" have been used by *Immanuel Kant* (April 22, 1724 - February 12, 1804) in his book *Kritik der reinen Vernunft* [46] to discriminate between statements that can be recognised as true from their mere form[6], and statements that are not true merely because of formal reasons. Since the latter are not merely true due to their logical structure they are the ones that carry actual informational content. According to Kant, there are synthetic statements that can be recognised as true without referring to any experience. This view is critisised by many contemporary philosophers of science.

Analytic vs. synthetic induction principle

An analytic induction principle would be the possibility to *prove* general statements from finite observations, i.e. logically justify their truth from a finite amount of empirical data. A synthetic induction principle would be the possibility to justify the truth of general statements on factual grounds. Unfortunately, both principles cannot exist for logical reasons.

About the impossibility of an analytic induction principle

Assume that Fn is the future prediction that shall be proven from a number of observations $Fa, \ldots, Fm$, where $a, ..., m$ are certain objects having the property F and $n \notin \{a, ..., m\}$ is a certain object for which the property F shall be predicted. It is possible to prove that n has F, i.e. $\{Fa, ..., Fm\} \Rightarrow Fn$ if and only if $Fa \wedge ... \wedge Fn \rightarrow Fn$ is a logically true sentence. If $Fa \wedge ... \wedge Fn \rightarrow Fn$ were logically true, it must not hap-

[5]David Hume: April 26, 1711 - August 25, 1776

[6]Those statements are called *tautologies* in logic.

pen that n has not F, i.e. Fn is false. But, in general, there is always the logical possibility for empirical objects like n that Fn is false. Fn could be false without violating the consistency of the empirical world. Therefore, $Fa \wedge ... \wedge Fn \rightarrow Fn$ cannot be true for logical reasons. Thus, an analytic induction principle is impossible.

About the impossibility of a synthetic induction principle

The question whether we are able to know that a synthetic induction principle exists can again be split into two parts, namely if we can know *a priori* that it exists or whether we can know *a posteriori* that it exists. If there was a statement that designated a synthetic induction principle to be true and which would turn out to be false a posteriori, the induction principle would loose its justification. But since there is no logical reason why a general empirical statement cannot turn out to be false, every statement generated by an a priori synthetic induction principle must also be true a priori. However, empirical statements can only be true a posteriori. Therefore, an a priori synthetic induction principle for a posteriori true statements cannot exist. This answers the first part.

The only possibility left to justify general empirical statements is an a posteriori synthetic induction principle. In this case it would be a general empirical statement and the only way to show its truth was to check all instances or to use another induction principle. The former is impossible due to the infinite range of an induction principle. The latter leads to an infinite regress. Therefore there can be no a posteriori synthetic induction principle. This means that any general empirical law cannot ultimately be proven, which condemns it to be a mere assumption, even though it might be well reviewed.

About the impossibility of a general algorithm yielding almost true statements

Philosophy of science concerned itself almost exclusively with the question whether the truth of general laws or statements, which are inferred from a finite number of observations, can be proven. This corresponds to the question if a learning algorithm can provably learn a function from a finite amount of data which has zero expected loss on the entire domain of the data. The preceding paragraphs showed that logical reasons render such an inference principle impossible. But what if the problem is relaxed to the question whether it is possible to design a general learning algorithm that learns the true function up to some small, bounded error? Unfortunately, this is not possible either, as is established by the so called *No-Free-Lunch theorem* [23, 105, 106, 10]. Its four parts basically state that there is no superior learning algorithm regarding the off-training set error, which is the error on all examples not contained in the training set. In detail the statements are [23]:

The No-Free-Lunch theorem

1. *Uniformly averaged over all target functions, for any two learning algorithms trained on sets of equal size, the expected off-training set error is the same.* So, if all target functions are equally likely, no matter how clever a learning algorithm is chosen, it will not

outperform any other. Furthermore, this means that for each learning algorithm there is at least one target function for which random guessing is the better algorithm.

2. *For any fixed dataset, uniformly averaged over all possible target functions, the expected off-training set error of two arbitrary learning algorithms is the same.* This means that even if the data set used for learning is known, without any knowledge about the target function, it is impossible to choose a learning algorithm that outperforms the others.

3. *Uniformly averaged over all distributions over target functions, for any two learning algorithms trained on sets of equal size, the expected off-training set error is the same.* This part makes basically the same statement as (1) only that it ascends one step in the hierarchy and considers the situation in which not all functions but all distributions over possible target functions are equally likely. Therefore, if one does not know anything about the distribution of target functions, on average, any learning algorithm is as good as any other.

4. *For any fixed training set, uniformly averaged over all distributions over target functions, for any two learning algorithms trained on sets of equal size, the expected off-training set error is the same.* This relates to statement (2) in the same way statement (3) relates to (1).

About the necessity to make assumptions for learning

In summary, the No-Free-Lunch theorem says that the only way to do better than random guessing is to narrow down the class of possible target functions by making specific assumptions about it. If the assumptions are correct, one learning algorithm can possibly outperform others. From the preceding paragraphs it is clear that, for an unknown target function, it is impossible to prove the correctness of ones assumptions. So all that remains is to hope that the assumptions are correct indeed. As will be seen later, one possible way of introducing a bias for the function class admissible to the learning algorithm and therefore incorporating prior knowledge in the learning problem is to use additional data that implicitly specify certain desired properties of the resulting function. These data points can have the same distribution as the training data, like in semi-supervised learning, or they can have a different distribution. The latter case is investigated in this thesis.

1.3.3 General Assumptions in Bayesian and Frequentist Learning

Assumptions via a prior

The No-Free-Lunch-Theorem states that there is no other way to perform better than random guessing but to make assumptions about the class of target functions. The various assumptions that can be made mostly differ in their level of generality, varying from very specific to very general. The most specific assumptions are made in those learning algorithms that are specifically tailored to a certain application. Here, a priori knowledge about, for instance, the linearity of the target function, its positive range, its limited domain or its invariance with respect to a group of transformations [68, 69] can directly be incorporated into the algorithm by restricting the accessible functions to a class that obeys the desired properties. Probably the simplest example of this kind is linear least squares regression [29, 93] which is an everyday tool in all branches of empirical science. Of course, if the assumptions are not correct, the generalisation performance may be arbitrarily poor. This phenomenon can be observed in the case of linear regression, too, when fitting samples from a non-linear function. In Bayesian learning these assumptions are reflected in the choice of the prior for predicting functions or in the parameters that indirectly influence the posterior distribution of target values.

Assumptions via a noise model

Another rather general assumption is the choice of the *noise model*, that is the assumed distribution of noise on the target values. In Bayesian learning the noise model is reflected in the likelihood function $\mathbf{P}_{\mathsf{L}|\mathsf{F}}(\mathfrak{L}|f)$, i.e. the probability of the observed data $\mathfrak{L} = \{(x_1, y_1), ..., (x_m, y_m)\}$ given a certain target function f. A very general assumption being made in this respect in almost all learning algorithms is that the noise at one point in the domain of the observations is independent of the noise at any other location. In this case the probability for the noise can be written as a product of independent noise probabilities

$$\mathbf{P}_{\mathsf{L}|\mathsf{F}}(\mathfrak{L}|f) \quad = \quad \prod_{i=1}^{m} \mathbf{P}_{noise}(f(x_i) - y_i).$$

Independent noise model and cumulative loss penalisation

When looking at the log likelihood $\log \mathbf{P}_{\mathsf{L}|\mathsf{F}}(\mathfrak{L}|\mathsf{F})$ this term becomes a sum of noise terms over all observations

$$-\log \mathbf{P}_{\mathsf{L}|\mathsf{F}}(\mathfrak{L}|f) \quad = \quad -\sum_{i=1}^{m} \log \mathbf{P}_{noise}(f(x_i) - y_i).$$

The term $\sum_{i=1}^{m} \log \mathbf{P}_{noise}(f(x_i) - y_i)$ corresponds to the additive penalisation of errors with respect to some loss function ℓ in frequentist learning

$$\text{maximise}_f \sum_{i=1}^{m} \log \mathbf{P}_{noise}(f(x_i) - y_i) \quad \triangleq \text{minimise}_f \quad \sum_{i=1}^{m} \ell(f(x_i) - y_i).$$

Linear least squares assumes independent Gaussian noise

Therefore, the choice of the noise model determines the optimal loss function and the assumption of independence allows the independent minimisation of the empirical error made on the training set. Looking again at the example of linear least squares, it is easy to see that the underlying noise model is independent Gaussian noise:

$$\begin{aligned}\log \mathbf{P}_{\mathsf{L}|\mathsf{F}}(\mathfrak{L}|f) &\overset{independence}{=} \log \prod_{i=1}^{m} \mathbf{P}_{noise}(f(x_i) - y_i) \\ &\overset{Gaussian}{=} \log \prod_{i=1}^{m} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(f(x_i)-y_i)^2}{2\sigma^2}\right) \\ &= -c_1 \cdot \sum_{i=1}^{m} (f(x_i)-y_i)^2 + c_2\end{aligned}$$

Here, c_1 and c_2 are constants that do not depend on $\mathfrak{L}$. Therefore they can be ignored when maximising the log likelihood or minimising the loss, respectively. Thus, assuming independent Gaussian noise leads to the minimisation of the quadratic loss $\ell(f,x,y) = (f(x_i) - y_i)^2$. Of course, the tight link between loss functions and noise distributions is also valid in the other direction. Choosing a certain loss function is equivalent to the assumption of a certain noise distribution. In practice, however, the loss function is not always chosen according to a specific noise distribution, but also according to geometric considerations or certain properties the solution of the learning problem should have, resulting in more sophisticated noise models [89].

Occam's Razor and simple functions

The most general assumption made in most learning algorithms is that the resulting function should be simple. This approach is often motivated by a principle called *Occam's Razor.* Occam stated that "entia non sunt multiplicanda praeter necessitatem" which translates to "entities (explanations) should not be multiplied beyond necessity". In machine learning Occam's Razor is understood as the preference for learning algorithms that do not yield functions being more complex than necessary. Surely, "necessary" is a very vague formulation describing the trade-off between the quality of the fit to the training data and the simplicity of the resulting function and depends on the specific learning problem at hand. Simplicity is mostly understood as smoothness of the learnt function, which is controlled by a regulariser in frequentist learning (see 1.5.1) or by an appropriate prior over target functions[7] in Bayesian learning. Nevertheless, the No-Free-Lunch Theorem states that, in general, simpler or smoother functions are not any better than complex ones. The reason for the frequent empirical successes of Occam's razor might be that we expect nature to behave in a smooth way, i.e. that small variations in the inputs result in only small variations in the outputs. This is the case in many real world functions such as physical laws, so smoothness is one of the weakest assumptions that can be made about a putative target function [68, 69]. But as

Possible motivations for Occam's Razor

[7]In general, the use of a prior does not necessarily lead to smooth prediction functions.

[23] points out, it might as well be that only the smooth relationships are the ones that quicken our interest and there is no good reason to believe that all real world relationships have to be smooth.

Another motivation for the implementation of Occam's razor in learning algorithms would be the assumption that adding more data points to the training set does not, on average, degrade the generalisation performance of the learning algorithm. Then, according to [23], a version of Occam's razor can be derived. But again, as emphasised by [23], this amounts to a non-uniform prior over target functions. Since we have no secure knowledge about possible target functions, the assumption that the generalisation error does not increase with more training data is a premise and cannot be proven.

1.3.4 Implicit prior Knowledge via additional Data

Semi-supervised learning: Using prior knowledge implicit in data from the same distribution

The goal of supervised learning is to establish a correspondence between data points $\mathfrak{X}$ from the input domain $\mathcal{D}$ to their labels $\mathfrak{Y}$ by examining a finite labelled data set $\mathfrak{L} = \{x_i, y_i\}_{i=1,\dots,m}$, $x_i \in \mathfrak{X}$, $y_i \in \mathfrak{Y}$. This allows for a special way of incorporating prior knowledge: In many situations additional unlabelled data from the domain $\mathcal{D}$ and the same distribution as the labelled data is available. One example from bioinformatics would be the classification of proteins into families: Sequencing a protein or the corresponding DNA/RNA is cheap and highly automated, but determining its protein family requires more manual work and is therefore much more expensive. In practice, usually only a few examples are labelled while the bigger part remains unlabelled. The problem specific information which is implicitly present in those data points, even though they have no label, can be used to bias the function class of the learning algorithm in a certain way. This setting is called *semi-supervised learning* which is part of the more general framework of *data-dependent regularisation*. Those additional data points can add valuable information to the learning problem. Different methods of semi-supervised learning and data-dependent regularisation are briefly reviewed in section 1.5.2.

Universum learning: Using prior knowledge implicit in data from another distribution

The setting in which the unlabelled data is from a different distribution than the training points has only been investigated very little so far [99, 98, 103]. The situation is more subtle here. If the unlabelled data is just from another distribution and not related to the learning problem in any respect, it is clearly useless to the learning algorithm. However, there are situations in which additional unlabelled data is not from the same distribution but related to the learning problem at hand. One example is hand written digit recognition. When classifying patches of handwritten fives against patches of handwritten eights, examples of all other handwritten digits are clearly not from the same distribution but from the more general distribution of handwritten digits.

Since those unlabelled examples, which will be called *Universum examples* later, are from a more general framework of learning problems, they can still be used as a source for problem specific prior knowledge. In the case of digit recognition, the other digits could contain information about certain transformations (like rotation, translation, line thickening etc.) handwritten digits - and therefore also fives and eights - are usually afflicted with.

The use of this kind of data points in classification and the implicit assumptions made about this Universum set in the original idea by [99, 98] as well as the implementation of [103] are the main focus of this thesis.

1.4 Mathematical Tools

1.4.1 Reproducing Kernel Hilbert Spaces

As mentioned in the introduction, one underlying assumption found in almost all learning algorithms is the simplicity of the learning process' solution, motivated by the principle of Occam's Razor. Probably the simplest functions beneath constant ones are linear functions. In fact, the associated function class of many early learning algorithms like the perceptron [76] was the class of all linear functions on the input domain. This choice of function class is very restrictive in the sense that only linear target functions on the domain can be approximated. As soon as the true target function is non-linear, even the best linear approximation to it might give bad predictions. In principle, there are several ways to escape the realm of linearity. One very straightforward way is to extend the function class to a non-linear one for which each element is described by a set of parameters. For example, the learning algorithm could be allowed to use the set of all polynomials up to some order n. Each polynomial $f(x) = \sum_{k=0}^{n} a_k x^k$ of order n can be described by a set of $n+1$ coefficients. The learning task would then be to estimate those parameters such that the target values at the given data points are appropriately approximated. The problem with this so called *parametric* approach is that the desired function class might not be easily parametrised or lead to a vast amount of parameters. If the number of parameters to be estimated is large and the amount of given training examples is small, the problem is under-determined. Then the solution highly depends on the method used for ensuring a unique solution.

Linear functions are simple to estimate but not very expressive

Another way to deal with non-linear target functions is to use non-linear transformations ϕ_k as *features* to encode the data points x_i in another representation $(\phi_1(x_i), ..., \phi_n(x_i))$ in which the assumed target function is linear $f(x) = \sum_{i=1}^{n} a_i \phi_n(x_i)$. The approach of changing the data representation is called *non-parametric.* One drawback of chang-

Non-linear functions are more expressive but hard to estimate ing the representation explicitly is that the amount of functions ϕ_k might be very large resulting in high computational costs and therefore slowing down the learning and the prediction process. This is clearly undesirable.

Non-linearity by changing the geometry of the space: The kernel trick

A very popular approach in machine learning that uses a non-linear transformation of the inputs but avoids the explicit computation of the features ϕ_k is the so called *kernel trick.* It exploits the fact that many learning algorithms can be formulated such that the data points enter the algorithm only in terms of dot products. The canonical dot product $\langle \mathbf{x}, \mathbf{x}' \rangle = \sum_{i=1}^{d} x_i x_i'$ can then be replaced by an appropriate function $k(\mathbf{x}, \mathbf{x}')$ which corresponds to a dot product between the non-linearly transformed data points. The value of this dot product called *kernel* can in most cases be obtained without explicitly computing the non-linear transformation of the data points. In fact, most of the time the situation is just the other way round: People plug a kernel into their algorithms, for which they know that it corresponds to a dot product between non-linearly transformed data points and do not care so much about the explicit representation of the transformed data.

In the remainder of this section, kernels and their associated spaces will be introduced more formally. In particular, it will be shown that using kernels corresponds to mapping the data points to elements of a so called *reproducing kernel Hilbert space* of functions, and carrying out computations there. The material of this section is taken from [81, 85].

DEFINITION KERNEL: Let $\mathcal{D}$ be a nonempty set. A symmetric function $k : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ on $\mathcal{D}$ is called *positive semi-definite kernel* if for any subset $\{x_1, ..., x_n\} \subseteq \mathcal{D}$ the *Gram matrix* $(\mathbf{K})_{ij} := k(x_i, x_j)$ is *positive semi-definite,* that is

$$\sum_{i,j=1}^{n} a_i a_j \mathbf{K}_{ij} \geq 0 \tag{1.2}$$

for all $a_i, a_j \in \mathbb{R}$. An equivalent requirement is that all eigenvalues of $\mathbf{K}$ are non-negative.

▷

Kernel as a similarity function between arbitrary objects

A kernel can be thought of as a similarity function between two elements of $\mathcal{D}$. However, not every similarity function on $\mathcal{D}$ is a valid kernel since the positive semi-definiteness requirement might be violated. There is no requirement for $\mathcal{D}$ other than being a nonempty set. Therefore, a kernel allows for computing similarities between arbitrary objects such as strings, graphs or probability distributions. Furthermore, as will be shown next, a kernel gives rise to a non-linear mapping into a Hilbert space of functions in which computations can be carried out that potentially would not have been possible in $\mathcal{D}$, because $\mathcal{D}$ might not even be a vector space. Beyond that, if the kernel is chosen carefully, the transformation of the elements of $\mathcal{D}$ might bring forward certain features that make it easier for the learning algorithm to learn the desired target function, i.e. the elements of the training set are transformed in a way such that a linear function on the transformed examples is a good approximation of the target function.

The following paragraphs show how to construct a Hilbert space $\mathcal{H}$ of functions for a given kernel. For this purpose, a mapping Φ is defined that assigns a function to each data point. In a next step it is demonstated that the image of the input domain $\mathcal{D}$ under the transformation Φ lies in a space of functions. Finally, this space will be turned into a Hilbert space by constructing a dot product on it elements.

Kernels map elements of $\mathcal{D}$ to a Hilbert space of functions $\mathcal{H}$

For a given kernel k the transformation that maps an element $x \in \mathcal{D}$ to a function is

$$\begin{array}{rcl} \Phi : \mathcal{D} & \rightarrow & \mathcal{H} \\ x & \mapsto & k_x := k(x, \cdot), \end{array}$$

where $k(x, \cdot)$ is just the kernel k with one element fixed to x and $\mathcal{H} := \text{span}\{k_x | x \in \mathcal{D}\}$ (cf. 4.3) the space of functions that map $\mathcal{D}$ into $\mathbb{R}$. Clearly, k_x is a function from $\mathcal{D}$ into $\mathbb{R}$. The next step is to turn $\mathcal{H}$ into a Hilbert space of functions by constructing a dot product that has the so called *reproducing property.* This lets $\mathcal{H}$ become a *reproducing kernel Hilbert space (RKHS).*

DEFINITION HILBERT SPACE

Let $\mathcal{H}$ be a vector space, i.e. a set closed under addition and scalar multiplication that has the following properties for all $h_1, h_2, h_3 \in \mathcal{H}$ and $a_1, a_2 \in \mathbb{R}$:

1. Commutativity: $h_1 + h_2 = h_2 + h_1$
2. Associativity of addition and scalar multiplication: $h_1 + (h_2 + h_3) = (h_1 + h_2) + h_3$ and $a_1(a_2 h_1) = (a_1 a_2) h_1$
3. Additive and multiplicative identity: $h_1 + \mathbf{0} = h_1$ and $\mathbf{1} h_1 = h_1$
4. Existence of an additive inverse: $h_1 + (-h_1) = 0$
5. Distributivity of scalar and vector sums: $a_1(h_1 + h_2) = a_1 h_1 + a_1 h_2$ and $(a_1 + a_2) h_1 = a_1 h_1 + a_2 h_2$

$\mathcal{H}$ is called a *Hilbert space* if it is endowed with a dot product $\langle \cdot, \cdot \rangle : \mathcal{H} \times \mathcal{H} \rightarrow \mathbb{R}$ that induces a norm $||h||_{\mathcal{H}} := \sqrt{\langle h, h \rangle}$ such that the limit of every Cauchy sequence of elements in $\mathcal{H}$ is also in $\mathcal{H}$ (for the definition of Cauchy sequence see 4.3).

▷

$\mathcal{H}$ is a valid Hilbert space of functions

By definition, $\mathcal{H} := \text{span}\{k_x | x \in \mathcal{D}\}$ fulfills all the properties required to be a vector space. The first step to turn it into a Hilbert space is to define a dot product (for the definition of dot product see 4.3).

Let $f = \sum_{i=1}^{n} \alpha_i k_{x_i}$ and $g = \sum_{j=1}^{m} \beta_j k_{x_j}$ be two functions for $x \in \mathcal{D}$, $m, n \in \mathbb{N}$ and $\alpha_i, \beta_j \in \mathbb{R}$, then the following bilinear function is a valid dot product in $\mathcal{H}$:

$$\begin{aligned} \langle \cdot, \cdot \rangle : \mathcal{H} \times \mathcal{H} &\rightarrow \mathbb{R} \\ \langle f, g \rangle &= \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \beta_j k(x_i, x_j) \end{aligned}$$

$\langle \cdot, \cdot \rangle$ is obviously symmetric and bilinear. The positive semi-definiteness follows from the positive semi-definiteness of k since

$$\begin{aligned} \langle \sum_{i=1}^{n} \alpha_i k_{x_i}, \sum_{i=1}^{n} \alpha_i k_{x_i} \rangle &= \sum_{i,j=1}^{n} \alpha_j \alpha_i k(x_i, x_j) \\ &\overset{(1.2)}{\geq} 0 \end{aligned}$$

One very important property directly following from the definition of $\langle \cdot, \cdot \rangle$ is that $\langle f, k_x \rangle = f(x)$ for all functions $f \in \mathcal{H}$. In particular, this includes the special case $\langle k_x, k_{x'} \rangle = k(x, x')$, which is called the *reproducing property*.

As already stated in the definition of a Hilbert space, a dot product induces a norm via $|| \cdot ||_{\mathcal{H}} := \sqrt{\langle \cdot, \cdot \rangle}$. All that remains to do in order to turn $\mathcal{H}$ into a proper Hilbert space is to make it complete. This can be done by a simple mathematical trick as noted in [81]: All limit points of Cauchy sequences are just included by definition. Therefore, the mathematically correct definition is $\mathcal{H} := \overline{\text{span}\{k_x | x \in \mathcal{D}\}}$, where the overline denotes the *completion*, i.e. the inclusion of all limit points. From now on $\mathcal{H}$ is used in this sense.

Hilbert spaces have particularly nice properties

Hilbert spaces have nice properties such as the possibility of defining a projection (see [81] or 4.3). This will be very useful later, especially in sections (2.4.1) and (2.4.2). Another very useful property when carrying out computations with kernels is the *Cauchy-Schwarz inequality*

$$\begin{aligned} |\langle k_x, k_{x'} \rangle|^2 &= |k(x, x')|^2 \\ &\leq k(x, x) \cdot k(x', x'). \end{aligned}$$

The proof can be found in [81].

The kernel trick: Replace dot products with kernels

By virtue of the properties of $\mathcal{H}$ and the transformation Φ, all learning algorithms that can be stated such that the data points solely enter the algorithm in terms of dot products, can implicitly work in a Hilbert space of functions by simply replacing all dot products by the kernel function k. The resulting function is linear in $\mathcal{H}$, but due to the non-linear transformation of the data points via Φ, the resulting function is actually non-linear in $\mathcal{D}$. This combines two advantages: On the one hand non-linear functions can be learnt, while on the other hand all mathematics valid for linear functions and spaces can be used for backing the algorithms with sound theory.

Learning in infinite dimensional kernel spaces: The representer theorem

An important theorem by Kimeldorf and Wahba and its generalised version by Schölkopf and Smola [50, 81], called *representer theorem*, ensures that all learning algorithms that can be formulated in a certain way[8] have a solution that can be expressed as a linear combination of the training examples transformed by Φ. Since the number of training examples is finite, $\mathcal{H}$ might even be infinite dimensional without problems. Only a finite dimensional subspace spanned by the data points has to be searched for a solution of the learning problem and the learning process is reduced to finding the coefficiensts for the linear combination of kernel functions. When evaluating the learnt function at a given point, only this linear combination of kernel functions between the training points and the new point has to be computed.

1.5 Regularisation

1.5.1 General regularisation

Two needs for regularisation: Uniqueness and prior information

As stated in 1.3.2, there are basically two reasons why a learning algorithm must make assumptions about the domain of target functions it is supposed to approximate. The first is the ill-posedness of the inverse problem: Datasets of finite size are mapped to a function that is supposed to describe a general relation between the labels and the distribution of the data. When dealing with a finite amount of data several solutions are possible and therefore the inverse problem might not be uniquely solvable. The second reason is given by the No-Free-Lunch Theorem saying that the only chance to perform better than random guessing is to make (correct) assumptions about the class of possible target functions. The same problem applies to Bayesian learning: Many functions might have the same likelihood $\mathsf{P}_{\mathsf{F}|\mathsf{L}}(f|\mathfrak{L})$ making the solution non-unique as well. As noted in 1.3.3, learning algorithms incorporate prior knowledge in various ways in order to avoid the above two problems. This incorporation of general assumptions about the target function is called *regularisation*. This section presents a more formal overview about regularisation in Bayesian and Frequentist learning and points out some links between them. Starting with the Bayesian way of regularising problems, the link to regularisation in Frequentist formulations of learning problems is established. In addition, the representer theorem for both cases is stated. At the end of this section, the special case of choosing an RKHS of functions and its relation to regularisation is discussed briefly.

[8]The class of learning algorithms which the representer theorem covers is actually very large.

Regularising with a prior

Probably the most straightforward way of implementing prior knowledge into a learning problem is to use a prior over functions or - equivalently - a prior over parameters of functions. This is the essence of Bayesian regularisation. Given a data set $\mathfrak{L}$, a likelihood function $\mathsf{P}_{\mathsf{L}|\mathsf{F}}$ in terms of a noise or observational model and a prior P_{F}, the posterior probability of a function f describing the data after seeing $\mathfrak{L}$ can be calculated by Bayes rule

$$\begin{aligned} \mathsf{P}_{\mathsf{F}|\mathfrak{L}}(f) &= \frac{\mathsf{P}_{\mathsf{L}|f}(\mathfrak{L})\mathsf{P}_{\mathsf{F}}(f)}{\int_{\mathcal{F}} \mathsf{P}_{\mathfrak{L}|\mathsf{F}}(g)dg} \\ &= \frac{\mathsf{P}_{\mathsf{L}|f}(\mathfrak{L})\mathsf{P}_{\mathsf{F}}(f)}{\mathsf{E}_{\mathsf{F}}[\mathsf{P}_{\mathfrak{L}|\mathsf{F}}]}, \end{aligned}$$

where $\mathcal{F}$ denotes the set of functions which is accessible to the learning algorithm. Since the denominator does not depend on f, the posterior only depends on the nominator up to some multiplicative normalisation constant

Posterior $\propto$ Likelihood $\times$ Prior: The Posterior is the likelihood reweighted by prior beliefs

$$\mathsf{P}_{\mathsf{F}|\mathfrak{L}}(f) \propto \mathsf{P}_{\mathsf{L}|f}(\mathfrak{L})\mathsf{P}_{\mathsf{F}}(f).$$

The posterior can therefore be seen as the likelihood reweighted by the prior. If the likelihood has several maxima for a given $\mathfrak{L}$, the maximally likely function is the one with maximum product of prior and likelihood. This method of estimating a function is called *maximum a posteriori (MAP).* If the prior is flat, i.e. does no prefer any value, it does not influence the solution and the resulting function will be the one with maximum likelihood. This is exactly what is done in *maximum likelihood estimation (ML).* In general, a prior does not necessarily make the problem uniquely solvable. Nevertheless, when not being flat, it implements an assumption about possible target functions. Figure 1.3 shows a simple example for the effect of a prior on the MAP estimation of a function. In this case, the prior has been chosen badly since it strongly prefers target functions that are different from the true one generating the data.

Regularisation by choice of function class

Another a priori assumption implicitly made in Bayesian and Frequentist learning is the choice of the function class that is admissible to the learning algorithm. In parametric function estimation the family of functions is fixed to functions that can be described on a fixed form in terms of the parameters. The distributions are then specified in terms of the parameters.

The most famous example for non-parametric Bayesian function estimation are *Gaussian Processes [73].* A Gaussian Process is a distribution over functions that is fully specified by a mean and a covariance function. Here, the corresponding function class is chosen via the covariance function. In general, Bayesian estimation does not estimate a single function but a distribution over functions instead, which is then used to predict function values at new points.

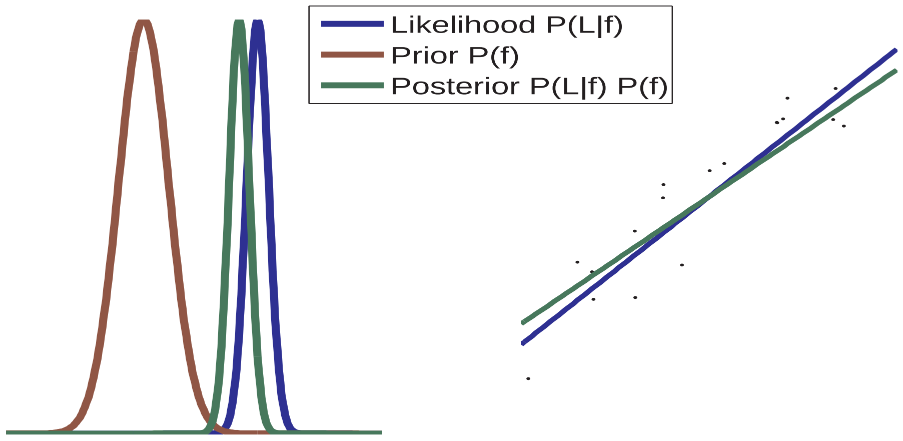


Figure 1.3: Toy example for regularisation by a (bad) prior. A few points $x_i \in \mathbb{R}$ have been sampled from the real line and the corresponding outputs have been generated via $y_i = x_i + \mathsf{E}$ with $\mathsf{E} \sim \mathcal{N}(0, 0.1)$. The considered functions are all possible linear functions without offset $f(x) = cx$ with a prior $\mathbf{P}_{\mathsf{c}}(c) = \frac{1}{0.2\sqrt{2\pi}} \exp\left(-\frac{|c-0.2|^2}{2 \cdot 0.2^2}\right)$. The likelihood was chosen to be $\mathbf{P}_{\mathsf{Y}|x,c}(y) = \frac{1}{0.1\sqrt{2\pi}} \exp\left(-\frac{|c \cdot x - y|^2}{2 \cdot 0.1^2}\right)$ for a single datum and $\mathbf{P}_{L|c}(\mathfrak{L}) = \prod_{i=1}^n \mathbf{P}_{\mathsf{Y}|x_i,c}(y_i)$ for all observations. All density functions have been normalised to the same height for illustrative reasons. The right figure shows the resulting regression functions. The blue line corresponds to the value c_{ML} of C with maximum likelihood, while the green line corresponds to the maximum a posteriori value c_{MAP}. While the data points have a great influence on the choice of regression function, the prior twists it a little bit towards its most probable value. Therefore, the solution is a trade-off between observed data and prior belief.

Link between Bayesian and Frequentist regularisation

As already noted in 1.3.3, there is a close link between a likelihood and a loss function. There is also a close link between Bayesian regularisation and standard Frequentist regularisation. In Frequentist learning the problem is mostly expressed as an optimisation problem over functions that involves two terms, a regularisation term $\Omega[f]$ and a loss term, which measures the performance of the current function on the data:

$$\text{minimise} \quad F[f] \quad = \lambda\Omega[f] + Loss[f, \mathfrak{D}], \tag{1.3}$$

where λ is a parameter that controls the trade-off between the regularisation and the loss term. Assuming that the examples have been drawn *identically* and *independently* *(i.i.d)*, the loss term decomposes into a sum over the loss function ℓ on the single data items:

$$\text{minimise} \quad F[f] \quad = \lambda\Omega[f] + \sum_{i=1}^{m} \ell[f, d_i],$$

with $d_i = (x_i, y_i)$ in supervised learning and $d_i = x_i$ in unsupervised learning. The case of semi-supervised learning is somehow special and will be treated in section 1.5.2. In almost all cases the regulariser Ω is expressed as a norm on the result of an operator D applied to the function f:

$$\text{minimise} \quad F[f] \quad = \lambda||Df|| + \sum_{i=1}^{m} \ell[f, d_i].$$

Very often the squared norm of the function is taken instead, which does not make any principal difference. This special form of the problem, though covering a large portion of existing learning algorithms, has a intimate relation to the Bayesian MAP estimation of a function (see e.g. [70, 92, 81]). If the prior over functions is given as a Gaussian in the range of the operator D, i.e. $\mathsf{P}_{\mathsf{F}}(f) \propto \exp(-||Df||^2)$, and assuming a certain noise model $\mathsf{P}_{\mathsf{D}|f}(\mathfrak{D}) = \prod_{i=1}^{m} \mathsf{P}(\ell(d_i)|f) \propto \prod_{i=1}^{m} \exp(-\ell[f, d_i])$, the negative logarithm of the posterior has the following form

$$\begin{aligned} -\log \mathsf{P}_{\mathsf{F}|\mathfrak{D}}(f) &\propto \log\left(\exp(-||Df||^2) \cdot \prod_{i=1}^{m} \exp(-\ell[f, d_i])\right) \qquad (1.4) \\ &= ||Df||^2 + \sum_{i=1}^{m} \ell[f, d_i], \end{aligned}$$

recovering the form of a Frequentist estimation problem. In general, however, $\mathsf{P}_{\mathsf{D}|f}(\mathfrak{D}) = \prod_{i=1}^{m} \exp(-\ell[f, d_i])$ is not necessarily a proper density since it might not integrate to one. In practice, however, this does not pose a problem in most cases.

The assumptions made about functions in the Frequentist setting are encoded in the type of operator and norm considered. Since $||Df||^2$ is minimised, functions that have a small norm under D are favored by the optimiser. In this sense, D and $||.||$ are used to express one's uneasiness about certain functions by choosing D and $||.||$ such that those functions will cause $||Df||^2$ to be large. Nevertheless, as long as $||Df||^2 < \infty$, the function can still be chosen when strongly suggested by the loss term, just like in the Bayesian case when dealing with a likelihood and a prior. The case in which the prior is zero corresponds to $||Df||^2 = \infty$, i.e. the function f cannot be the solution of the learning problem.

Regularisation in an RKHS

A special case, though very widely used, is when the functions live in an RKHS $\mathcal{H}$ of functions and the corresponding norm of that RKHS $||\cdot||_{\mathcal{H}}$ is used to regularise the problem. The most prominent example in Bayesian learning are Gaussian processes [73] using the kernel function as covariance function. Together with an appropriate noise model this leads to a representation of the learnt function as $f^* = \sum_{i=1}^{m} \alpha_i k(x_i, \cdot)$, where $x_1, ..., x_m$ are the training inputs. The coefficients $\alpha_1, ..., \alpha_m$ are obtained when computing the posterior for the function value of a new input given the training data. In the most widely used form, the noise model for regression is Gaussian. The noise models for classification are more involved.

In Frequentist learning the optimisation problem has the form of equation (1.3) with $\Omega[f] = \Theta(||f||_{\mathcal{H}})$, where $\Theta : [0, \infty) \to \mathbb{R}$ is a strictly monotonic increasing function and $f \in \mathcal{H}$ with the representation $f = \sum_{x \in \mathcal{X}} \alpha_x k(x, \cdot)$. While it is clear from the calculation of the posterior of Gaussian processes that the resulting function can be written as a linear combination of kernels, it is a priori not obvious that the minimising function f^* of

$$\text{minimise}_{f \in \mathcal{H}} F[f] \quad = \quad \Theta(||f||_{\mathcal{H}}) + Loss[f, \mathfrak{D}] \tag{1.5}$$

Solutions to regularised problems in RKHS: The representer theorem

also has this representation. But this is in fact the case. The famous representer theorem and later generalisations [50, 81] state that the minimiser $f^* \in \mathcal{H}$ admits the representation $f^* = \sum_{i=1}^m \alpha_i k_{x_i}$ just as for Gaussian Processes. The theorem is very general since the proof does not even require $Loss[f, \mathfrak{D}]$ to be a sum over single loss terms. Therefore, it allows for coupling the single items $d_i = (x_i, y_i)$ or $d_i = x_i$. The proof decomposes a possible solution f into a part $f_{||}$ that lies in the span of functions k_{x_i} and a part $f_\perp$ that is orthogonal to it, meaning $\langle f_\perp, k_{x_i} \rangle_{\mathcal{H}} = 0$ for all $k_{x_1}, ..., k_{x_m}$. It then shows that the values $f(x_j)$ at the training points can be expressed as a linear a combination of $k_{x_i}(x_j) = k(x_i, x_j)$ and that the regulariser $\Theta(||f||_{\mathcal{H}})$ cannot increase by setting $f_\perp = 0$. Therefore the best choice for the regulariser is $f^* = \sum_{i=1}^m \alpha_i k_{x_i}$ which has $f^*_\perp = 0$. This has the important implication that, even though the RKHS of functions might have infinitely many dimensions, the solution lies in the span of the training points and is therefore determined by a finite number for coefficients $\alpha_1, ..., \alpha_m$.

The choice of the kernel determines the regularisation properties

As already noted, the choice of the norm $|| \cdot ||$ for the regulariser determines the properties of the functions that are favored. Therefore, the choice of the kernel and its RKHS determine the properties of the regulariser. This might be hard to judge in the somewhat artificial construction of an RKHS of functions as in section 1.4.1. In fact, for certain types of kernels it is possible to define an operator Υ that maps the elements of the RKHS to another space $\mathcal{A}$ with a corresponding dot product that has the same value as the dot product associated with $\mathcal{H}$, i.e. $\langle f, g \rangle_{\mathcal{H}} = \langle \Upsilon f, \Upsilon g \rangle_{\mathcal{A}}$ for all $f, g \in \mathcal{H}$. The advantage of mapping functions from $\mathcal{H}$ to another space via Υ is that it allows to state the regulariser in terms of Υ and the norm in $\mathcal{A}$, which is often more amenable to an intuitive interpretation. For example, the dot product of two functions f, g in the space spanned by a translation invariant kernel, i.e. a kernel that can be expressed as a function of $x - x'$ for $x, x' \in \mathbb{R}^N$, can be expressed as

$$\langle \Upsilon f, \Upsilon g \rangle_{\mathcal{A}} \quad = \quad (2\pi)^{\frac{N}{2}} \int_{\Omega[v]} \frac{\overline{F[f](\omega)} \cdot F[g](\omega)}{v(\omega)} d\omega.$$

Here, $F[f](\omega)$ denotes the Fourier transformation of f, $\Omega[v]$ the support of the function v and $\bar{\cdot}$ denotes complex conjugation. The function $v(\omega)$, a nonnegative function, which is symmetric around zero and approaches zero as $\omega \rightarrow \infty$, is given by the Fourier transform of the kernel. In this representation it is easier to see that the norm in $\mathcal{H}$ favors functions that have a high value for $v(\omega)$ and that functions outside the support $\Omega[v]$ cannot be a solution to the regularised learning problem. Therefore, the regulariser favors functions with large low frequency and a small high frequency components. Intuitively, such a function will not dither much and is therefore smooth.[81] treats this topic in more detail.

$||Df||$ regularisers are agnostic to the specific data distribution

A possible drawback of the form of regularisers as shown in equations (1.3) and (1.5) is the lack of reference to the data distribution that might possibly add valuable information about the learning problem. This means that the form and therefore the implicit assumptions made by the regulariser is the same for all problems. According to the No-Free-Lunch Theorem this cannot work in all cases. *Data-dependent regularisers*, which try to overcome this problem by incorporating additional data points in the regulariser, are treated in the following section.

1.5.2 Data-dependent Regularisation

This section reviews existing approaches for using additional data to regularise a learning problem. Rather than claiming to be complete, this review intends to show a few examples from a broad spectrum of methods for data-dependent regularisation.

In data-dependent regularisation, the regulariser becomes a function of additional data points

In general, the influence of the additional data on the selection of the function is structurally built into the algorithm, while the choice of a certain function depends on the specific instance of additional data used in the respective machine learning problem. Thus, instead of specifying a general criterion such as smoothness, the regulariser is also a function of the additional data. Varying the data therefore changes the regulariser. The regulariser can be adapted to a specific learning problem by passing a dataset, which meets the assumptions made in the design of the data-dependent regulariser, to the learning algorithm.

In most cases, additional knowledge is knowledge about $\mathbf{P}_X$

In almost all cases, the additional data is assumed to be drawn from the same distribution $\mathbf{P}_X$ as the inputs of the labelled data. This means that the additional knowledge the algorithm gets is knowledge about $\mathbf{P}_X$. When enough unlabelled data is available, one could even attempt to estimate the input distribution. Since the data is supposed to reflect the input distribution, $\mathbf{P}_X$ itself is often used in the derivation of the algorithm or in a theoretical analysis of the benefits from $\mathbf{P}_X$.

In order to profit from the unlabelled data, the structure of the algorithm must be built in a way such that information about its distribution is used by the algorithm. This is not natural for all algorithms. For instance, in Bayesian classification there are two ways of estimating the label y for a given input point x. In a *generative model* $\mathsf{P}_{\mathsf{X}|\mathsf{Y}}$ the label y is thought of as a hidden state determining the generation of x. Inference of the label corresponds to inferring this state by taking the one that makes a given x maximally probable. However, these models are often difficult to estimate [83].

Conventional Bayesian discriminative models ignore additional information about P_X

Discriminative models $\mathsf{P}_{\mathsf{Y}|\mathsf{X}}$, which model the dependency between the inputs and the labels in the opposite way, are often more robust and easier to estimate. Unfortunately, in their standard form they have the inherent drawback of not using additional information about the input distribution P_X at all, because the distribution and the latent function which shall be inferred are independent. This means that information from additional unlabelled data is not used at all by a standard discriminative model. [83] proposes a solution by changing the structure of the model such that the latent target function becomes a priori dependent on the input distribution. This example illustrates that the relation between the choice of the function and additional data has to be structurally fixed in the algorithm and defines the aspects of the data the algorithm is supposed to use.

The cluster assumption

Another example for a structurally fixed relation between the choice of a function and the additional data points is the so called *cluster assumption*. Many semi-supervised classification algorithms assume that the data points of each class cluster. Consequently, the decision boundary should cut through regions of low data density, i.e. not cut through a cluster. The data density is measured with the help of unlabelled data points from the joint distribution of both classes.

Information Regularisation

In [16] the authors estimate a conditional distribution $\mathsf{Q}_{\mathsf{Y}|\mathsf{X}}$ of labels given the data points by covering the space of x with a set $\mathcal{T}$ of overlapping regions T and penalising the deviation of $\mathsf{Q}_{\mathsf{Y}|\mathsf{X}}$ on unlabelled examples from the values of labelled examples in that region by the Kullback-Leibler divergence [18] between the conditional $\mathsf{Q}_{\mathsf{Y}|x}$ and the best common choice $\mathsf{Q}_{\mathsf{Y}|T}$ across all examples in that region:

$$\text{minimise}_{\mathsf{Q}_{\mathsf{Y}|\mathsf{X}}} \qquad \sum_{x\in T} \mathsf{P}_{\mathsf{X}|T}(x) \sum_{y\in\mathcal{R}} \mathsf{Q}_{\mathsf{Y}|\mathsf{X}}(y|x) \log \frac{\mathsf{Q}_{\mathsf{Y}|\mathsf{X}}(y|x)}{\mathsf{Q}_{\mathsf{Y}|T}(y)}.$$

Assuming that the joint distribution is given by $\mathsf{Q}_{\mathsf{X},\mathsf{Y}}(x,y) = \mathsf{P}_{\mathsf{X}|T}(x)\mathsf{Q}_{\mathsf{Y}|\mathsf{X}}(x,y)$, this can be seen as minimising the mutual information between X and Y. Multiple regions are incorporated by minimising a weighted sum over all regions. The weights are chosen beforehand or can be calculated from a prior over the examples by using the probability mass of a region with respect to that prior as its weight. Here, the unlabelled examples are assumed to be from the same unknown distribution P_X as the labelled training examples. In this regularisation scheme, called *information regularisation*, all examples lying inside one region are a priori assumed to be equivalent, i.e. to have to same labels. Therefore, the regions play a role similar to clusters in the cluster assumption. However, the approach seems more general since it allows to specify a topology via $\mathcal{T}$ and minimise the information induced between the data points and their labels relative to that topology. Due to the overlap of the regions label information can be passed between regions by examples lying in the intersections, i.e. the $\mathbf{Q}_{\mathsf{Y}|\mathsf{X}}$ should have a bias towards the same label in both regions since the way of regularisation favors distributions with constant labels over one region. Thus, the additional unlabelled examples influence the choice of $\mathbf{Q}_{\mathsf{Y}|\mathsf{X}}$.

Metric-based approaches

Another way of using unlabelled data from the input distribution $\mathbf{P}_{\mathsf{X}}$ is proposed in [82]. Given a set of m labelled and one of u unlabelled data points, the authors use the additional data to induce two (pseudo) metrics on the space of hypothesis:

$$d(f,g) := \varphi\left(\int \ell\left(f(x),g(x)\right) d\mathbf{P}_{\mathsf{X}}\right) \quad \approx \quad \varphi\left(\frac{1}{u}\sum_{j=1}^{u} \ell\left(f(z_j),g(z_j)\right)\right) =: \tilde{d}(f,g)$$

$$d(\mathbf{P}_{\mathsf{Y}|\mathsf{X}},h) := \varphi\left(\iint \ell\left(h(x),y\right) d\mathbf{P}_{\mathsf{Y}|\mathsf{X}} d\mathbf{P}_{\mathsf{X}}\right) \quad \approx \quad \varphi\left(\frac{1}{m}\sum_{i=1}^{m} \ell(h(x_i),y_i)\right) =: \hat{d}(\mathbf{P}_{\mathsf{Y}|\mathsf{X}},h),$$

where φ is an associated normalisation function that recovers the standard metric axioms. The two metrics become real metrics, if $\mathbf{P}_{\mathsf{X}}$ is everywhere strictly greater than zero. If $\mathbf{P}_{\mathsf{X}}$ has no mass at some points on the space, two functions might have zero distance although they are not identical. Using the two metrics and an arbitrary origin function ϕ, the authors define two general objectives for learning algorithms:

$$\text{minimise}_h \qquad \hat{d}(h,\mathbf{P}_{\mathsf{Y}|\mathsf{X}}) + |\tilde{d}(\phi,h) - \hat{d}(\phi,h)|$$

$$\text{minimise}_h \qquad \hat{d}(h,\mathbf{P}_{\mathsf{Y}|\mathsf{X}}) + \max\left\{\frac{\tilde{d}(\phi,h)}{\hat{d}(\phi,h)}, \frac{\hat{d}(\phi,h)}{\tilde{d}(\phi,h)}\right\}$$

The intuition behind those two objectives is that the behavior of a function on the unlabelled examples should be similar to the behavior on the training dataset. This is measured by the distance to the origin function ϕ. In contrast to information regularisation, this method does not assume any specific kind of structure on the data, like the assumption that the function values should be equal in some predefined neighborhood. It is only assumed that the unlabelled data points are sampled from the same distribution $\mathbf{P}_{\mathsf{X}}$ as the training examples. Since the distance between the functions off the training set is measured on the unlabelled examples, the choice of a specific hypothesis h by the learning algorithm is influenced by the instances of the unlabelled examples, because they determine the important points where the distance should be small.

[6, 7] investigate how the knowledge of $\mathbf{P}_{\mathsf{X}}$ can in theory be used to augment the performance of binary classifiers under the cluster assumption. The authors propose three methods to use this assumption

together with P_X: a density based approach, where P_X is incorporated into the classifier; a geometry based approach, where the distances are modified locally in order to take P_X into account; and a method based on spectral clustering, where the first eigenvectors of a weighted graph adjacency matrix give rise to a representation of the data points that takes into account their manifold and cluster structure. Only the first two approaches are described in the following.

Implementing the cluster assumption by modifying the regulariser with the density

The first approach joins a general smoothness requirement with the cluster assumption by penalising the norm of the gradient weighted by the density $||\mathsf{P}_\mathsf{X}\nabla f||$ during the optimisation of the learning algorithm. Intuitively, a small gradient leads to a smooth function since the function cannot vary much locally. By incorporating the input density, the minimisation of the gradient becomes more important in regions of high density and is completely ignored in regions with zero probability mass. This preference for a small gradient in high density regions makes it difficult for the function to change the label in those regions and therefore implements the cluster assumption.

Implementing the cluster assumption by changing the geometry of the space

The geometry based approach locally changes the metric of the Euclidean space such that points that can be joined by a path which does not cut low density regions are put closer together. This implements the cluster assumption since the points that lie inside a common cluster are contracted towards each other. Here, the Euclidean space is seen as a Riemannian manifold with a metric tensor representing the normal Euclidean distance. A Riemannian manifold also forms a metric space when considering the geodetic distance, i.e. the length of the shortest path joining two points. By locally reweighting the metric tensor with the inverse density $\frac{1}{\mathsf{P}_\mathsf{X}}$, paths through low density regions become longer, while paths inside high density regions become shorter. As the authors show, the density based and the geometric approach are in fact equivalent, i.e. changing the measure and keeping the geometry or changing the geometry and keeping the Lebesgue measure leads to the same regulariser.

Weighting with the density and changing the geometry is equivalent

Fading between trusting the data and relying on prior knowledge by using P_X

A usage of the input density P_X which is completely opposite to the density based approach of [6, 7] is investigated in [11]. Here, the authors use a special kind of regulariser involving the Radon Nikodym derivative [24] which can be interpreted as penalising the norm of the first derivative of the function inversely scaled by the input density. The intuition behind scaling the first derivative with one over the density instead of the density itself, as in the implementation of the cluster assumption, is that the learning algorithm should trust the data in regions where the input density is high and give more weight to the smoothness requirement of the regulariser in regions with low density. The smoothness requirement is realised by the first derivative. It seems quite interesting that the implementation of this assumption compared to the cluster assumption interchanges the areas of heavy regularisation. Since the regulariser corresponds to prior assumptions about the problem, scaling inversely with the input density can be seen as fading between strong trust in the data, whenever it is available, and reliance on the prior assumption if the data density is low. Interestingly, the solution of a problem regularised in such a way is a linear combination of radial and sigmoid functions. Those kinds of expansions have been successfully used in neural networks. The results of [11] could be seen as a mathematical reason for their good performance in practice.

Biased Regularisation: Using a Gaussian prior with nonzero mean.

In an interesting practical approach of using a priori knowledge from labelled data of the same distribution, i.e. $(x, y) \sim \mathsf{P}_{\mathsf{XY}}$ is shown in [49] for the problem of personalised handwriting recognition. Here, the problem is to adapt a classifier to a specific style of handwriting with a very low number of labelled examples from that specific handwriting. The authors solve the problem by condensing a huge digit dataset with all sorts of handwriting styles into a generic linear classifier $\mathbf{w}_0$. When adapting this generic classifier to a specific style of handwriting, the prior knowledge about digits, represented by $\mathbf{w}_0$, is used to bias the new solution $\mathbf{w}$ towards $\mathbf{w}_0$ by using the regulariser $||\mathbf{w} - \mathbf{w}_0||^2$. As noted in section 1.5.1, minimising a squared norm regulariser and some noise model corresponds to the MAP estimate with a Gaussian prior $\mathbf{w} \sim \mathsf{P}_{\mathsf{W}} \propto \exp(||\mathbf{w}||^2)$. Using $||\mathbf{w} - \mathbf{w}_0||^2$ instead of $||\mathbf{w}||^2$ means using a Gaussian prior with mean $\mathbf{w}_0$. This makes perfect sense since the best guess for a classifier before having seen a single digit from the new handwriting style is $\mathbf{w}_0$.

In the following chapter, one specific algorithm that makes use of additional data which is *not* from P_{X} is introduced and analysed. In the experimental chapter it is applied to real world problems, while the focus lies on the approrpiateness for neural applications.

Chapter 2

The Universum Algorithm

2.1 The Universum Idea

2.1.1 VC Theory and Structural Risk Minimisation (SRM) in a Nutshell

This sections presents a brief overview about *VC theory (Vapnik Chervonenkis theory)* of statistical learning and the *structural risk minimisation (SRM)* principle for inductive and transductive learning building upon SRM. These basics will be needed in section 2.1.2 to understand the original idea for inference with a Universum by Vapnik [14, 99, 103].

The central goal of each supervised learning problem is to find a function f or a distribution over function values $\mathsf{P}_{\mathsf{F}|\mathsf{X}}$ that minimises the risk R. The functional R is the expected error of a function with respect to the underlying distribution P_{XY} on the domain $\mathcal{D} \times \mathcal{R}$ and a loss function ℓ:

Risk minimisation: Minimise the expected loss with respect to P_{XY}

$$R[f] \quad = \quad \int_{\mathcal{D}\times\mathcal{R}} \ell(f(x), y) d\mathsf{P}_{\mathsf{XY}}(x, y). \tag{2.1}$$

However, since the underlying distribution P_{XY} is unknown, directly minimising the true risk (2.1) is not possible. The Bayesian learning framework tries to address this problem by making prior assumptions about the underlying distribution and estimating a posterior distribution $\mathsf{P}_{\mathsf{F}|\mathsf{X},\mathfrak{L}}$ over functions given training data $\mathfrak{L} = \{(x_i, y_i)\}_{i=1,\dots,m}$. When predicting a new label $\hat{y}$, the minimising value of the expected risk with respect to the posterior

Bayesian approach: Minimise the risk with respect to the posterior

$$\hat{y} \quad := \quad \operatorname{argmin}_{y\in\mathcal{Y}} \mathsf{E}_{\mathsf{F}|x,\mathfrak{L}}[\ell(\mathsf{F}(x), y)]$$

is chosen. If the assumptions about the underlying distribution and the noise made by the choice of the prior and the likelihood are correct, this is the optimal strategy [23, 106].

Frequentist learning addresses the problem of minimising (2.1) without employing distributions, by directly estimating a single function f. A naïve approach for minimising the risk with a single function f would be to replace the integral in (2.1) by a sum over all examples in the training set and to minimise this quantity, called *empirical risk* R_{emp}, instead:

Frequentist approach: Find a single function that minimises the risk

$$R_{emp}[f] \quad = \quad \frac{1}{m}\sum_{i=1}^{m} \ell(f(x_i), y_i). \tag{2.2}$$

The hope behind this strategy, called *empirical risk minimisation*, is that the empirical risk R_{emp} will converge to the true risk (2.1) by the law of large numbers as the size m of the training set tends to infinity. This property is called *consistency* in statistics [81]. However, without any assumption about the class of target functions, the empirical risk does not carry any information about the value of the true function on locations $x \notin \mathfrak{L}$ that are not in the training set, and it turns out, in fact, that without any additional assumptions, empirical risk minimisation is not consistent [81]. When looking at the No-Free-Lunch theorem (see 1.3.2) this does not seem suprising.

Empirical risk minimisation is not consistent

The difficulties with empirical risk minimisation can be illustrated with a little example taken from [81]: If a learning algorithm is allowed to use the class of all functions, it can always achieve $R_{emp}[f^*] = 0$ with the simple function

$$f^*(x) \quad = \begin{cases} y_i & \text{if } x = x_i \text{ for some } x_i \in \{x_i\}_{i=1,\ldots,m} \\ 1 & \text{else} \end{cases} \quad .$$

Since f^* has constant output on all examples that are not in the training set, it will perform terribly as a predictor.

Worst case over all functions in the class determines the consistency of empirical risk minimisation

The main insight of VC theory is that the worst case over all functions admissible to a learning algorithm determines the consistency of empirical risk minimisation [81]. This means that the one-sided uniform convergence of the empirical risk to the true risk over all functions in the considered function class $\mathcal{F}$,

$$\lim_{m\to\infty} \mathbf{P}(\sup_{f\in\mathcal{F}}(R[f] - R_{emp}[f]) > \varepsilon) \quad = \quad 0,$$

is a necessary and sufficient condition for the consistency of empirical risk minimisation. The key idea of VC-theory is to bound the quantity $\mathbf{P}(\sup_{f\in\mathcal{F}}(R[f] - R_{emp}[f]) > \varepsilon)$ in terms of a measure for the *complexity* or *capacity* of the considered function class $\mathcal{F}$. The intuition behind this is that if the complexity of $\mathcal{F}$ is low, i.e. the ability of $\mathcal{F}$ to fit arbitrary values is weak, the value of $R_{emp}[f]$ can tell something about $R[f]$ since it cannot happen that the elements of $\mathcal{F}$ vary too much.

VC-theory: Bound largest difference between true and empirical risk in terms of the complexity of $\mathcal{F}$

One such bound in terms of the number of training examples m, the admissible function class $\mathcal{F}$ and $\varepsilon > 0$ is:

$$\mathbf{P}(\sup_{f\in\mathcal{F}}(R[f] - R_{emp}[f]) > \varepsilon) \quad \leq \quad \phi(m, \mathcal{F}, \varepsilon),$$

Given this bound, it is possible to derive confidence intervals for $R[f] - R_{emp}[f]$ by setting the left side to some value $0 \leq \delta \leq 1$ and solving for ε [81]. This results in a statement like

Bounds yield confidence intervals for the true risk

$$R[f] \quad \leq \quad R_{emp}[f] + \phi'(\delta, m, \mathcal{F}) \tag{2.3}$$

which holds for all functions in $\mathcal{F}$ with probability $1 - \delta$. In both cases, $\mathcal{F}$ enters in ϕ and ϕ' via the capacity measure, where ϕ' is increasing in the capacity of $\mathcal{F}$ and decreasing in m and δ. Thus, the higher the capacity of $\mathcal{F}$, the looser is the bound on the difference $R[f] - R_{emp}[f]$.

In order to guarantee a low upper bound on the risk $R[f]$, a learning algorithm has to minimise the right hand side of 2.3. However, minimising $\phi'(\delta, m, \mathcal{F})$ requires the limitation of the capacity of $\mathcal{F}$. Since the capacity measures are functions of the function class $\mathcal{F}$ and not a single element $f \in \mathcal{F}$, the limitation cannot be achieved via the optimisation over single functions in $\mathcal{F}$. Instead, a structure on $\mathcal{F}$ has to be built. A structure on $\mathcal{F}$ is a system $\mathcal{S}$ of nested subsets $... \subset S_{d-1} \subset S_d \subset S_{d+1} ... \subset \mathcal{F}$ with increasing size and capacity, where each subset $S_i \subseteq \mathcal{F}$ has a fixed maximal capacity. The goal of *structural risk minimisation (SRM)* is then to select an element S^* of the structure that has low empirical training error (in order to bound the first term of (2.3)) and low capacity (in order to bound the second term of (2.3)) where $\phi'(\varepsilon, m, \mathcal{F})$ is replaced by $\phi'(\varepsilon, m, S^*)$. Building such a structure is clearly a rather conceptual setting. In practice, bounds like (2.3) are used to design regularisers that are a trade-off between statistical optimality and practicability from an engineering point of view. A popular example are *Support Vector Machines (SVM)* that are motivated by an upper bound on a capacity measure called *VC dimension.* This upper bound is established in terms of the maximal distance of the closest training point to the hyperplane that separates the two classes. This, in turn, boils down to searching for the normal vector of a separating hyperplane that has minimal Euclidean length, as achieved by the regularisation of the normal vector with $||\cdot||_2^2$.

Structual Risk Minimisation: Optimise (2.3) over nested subsets of functions

The structural risk minimisation described above refers to the setting of *inductive learning,* in which the objective is to infer a function on the entire domain $\mathcal{D}$ of the data points. In the transductive setting the goal is merely to predict the labels for a given fixed set of test points. Predictions on other points than the specified test examples are not of interest for transductive learning. In this case structural risk minimisation boils down to building a similar structure on a finite set of equivalence classes $F_1, ..., F_N$ on $\mathcal{F}$. The equivalence classes F_i arise

Transduction: Structural Risk Minimisation on a finite set of equivalence classes since there is only a finite number of possible labellings of the training and test set. All functions that label the training and the test set in a certain way cannot be distinguished from their function values on the given data and are therefore considered equivalent. If the training set contains m and the test set n examples, there is a finite number of different labellings N of the training and the test set which is at most $N \leq |\mathcal{Y}|^{m+n}$. Labelling the test set then corresponds to choosing any function from an appropriate equivalence class and use that for predicting the unknown labels. Similar to the case of structural risk minimisation for inductive learning, the equivalence classes F_k are organised into a structure according to a chosen capacity measure, i.e. each element S_k of the structure contains all equivalence classes that have a certain maximal capacity. The goal of structural risk minimisation is then similar to the one of inductive inference, which is selecting an element S^* of the structure that has low empirical error on the labelled training set and low capacity. Here, the capacity increases with the index k of the element S_k of the structure. Probabilistic bounds similar to those in inductive inference can be derived [98], which use the number of equivalence classes N_k of an element S_k instead of the capacity measure:

Confidence bounds for transduction use the number of equivalence classes instead of the capacity measure

$$R_{emp}^{(train \cup test)}[F_r] = R_{emp}^{(train)}[F_r] + \tilde{\phi}(N_k, \eta, m), \quad (2.4)$$

where the bound holds with probability $1 - \eta$ and $F_r \in S_k$ (see [14, 99, 100, 98] for the actual bounds). The function $\tilde{\phi}$ is increasing in N_k and decreasing in η and m. The important fact is that $R_{emp}^{(train \cup test)}[F_r]$ depends solely on the number N_k of equivalence classes in the chosen element S_k, no matter which capacity measure was used to build the structure.

The type of capacity measure to build up the structure does not matter

The key idea for inference with the use of an additional dataset called *Universum* is to employ this set to build up the structure $\mathcal{S}$ on $F_1, ..., F_N$, i.e. to use a capacity measure based on the Universum. Since the specific instance of this dataset influences the way the structure $\mathcal{S}$ is built, it also has an impact on which equivalence class will be considered optimal. Therefore, structural risk minimisation with a Universum becomes an instance of data-dependent regularisation. The following section 2.1.2 describes the idea of inference with a Universum in more detail.

Structural Risk Minimisation with a Universum is data-dependent regularisation

2.1.2 The original Universum Idea by Vapnik

This section presents the original idea by Vapnik [14, 103, 100] of using an additional dataset $\mathfrak{U}$ for regularisation which is <u>not</u> necessarily from the same distribution as the training data as it is the case in semi-supervised or transductive learning. Vapnik calls this set $\mathfrak{U}$ *the Universum*, stressing the intuition that it is supposed to embed the learning

The original idea of a Universum is a set of data points from the same problem framework, but not from the same distribution as the labelled data.

problem into a larger framework of problems specified by $\mathfrak{U}$. One example of such a set would be a collection of various digits, letters and symbols, used as additional data for the classification of two other digits. So far, the Universum idea is restricted to supervised classification. Therefore, only this case is discussed in this thesis.

As already mentioned in the preceding section, the idea of transductive inference is to build a structure on a finite number of equivalence classes $F_1, ..., F_N$ given a certain training and test set, instead of building such a structure on an infinite object like most function classes. One very useful fact is that the second term in the probabilistic error bound (2.4) does not depend on how the structure $\mathcal{S}$ was built on $F_1, ..., F_N$, but only on on the number N_k of equivalence classes contained in the chosen element S_k. This means that a bound in the flavor of (2.4) is still valid if the structure $\mathcal{S}$ was build using another quantity to group the equivalence classes instead of the capacity measured by e.g. the margin on the training and test data [98]. For example, given a prior distribution $\mathbf{P}_\mathsf{F}$ over functions in $\mathcal{F}$, one possibility would be to arrange the $F_1, ..., F_N$ according to the probability mass $\mu(F_k) = \int_{F_k} d\mathbf{P}_\mathsf{F}(f)$ contained in F_k. Following that strategy, an element $S_k \in \mathcal{S}$ would get assigned all equivalence classes F_i that have $\mu(F_i) \geq a_k$ with $a_{k-1} > a_k > a_{k+1}$. Instead of selecting the element with good training error and low capacity, one would select the element with low training error and large probability mass. However, in most cases defining a prior might not be obvious and handling the integrals for computing the probability mass will probably be intractable in many cases. Now, the main point of Vapnik's idea is to use the Universum set $\mathfrak{U}$ for building the structure.

Possible ways to build a structure $\mathcal{S}$ ond $\mathcal{F}$

As already mentioned, the Universum can be thought of as a set from the same domain $\mathcal{D}$ and "problem category" as the training and test points, but not from the same distribution. To pick up the example from above: The learning task could be to classify pictures of handwritten fives and eights. In this case the Universum could be everything that could be perceived as a digit but is neither a five nor an eight (This is the reason for the name *Universum*: It was thought of as a superset of objects the examples at hand are drawn from). Intuitively, those data items carry some prior knowledge about the considered learning problem, like certain transformation in the data one wishes to be invariant against or simply that the classification function should not be supported in regions of high Universum density. How could it be used to build a structure on the set of equivalence classes? Vapnik proposes two related approaches. The first is to group the single F_i according to their *VC entropy* [14, 103, 100] on the Universum, that is the logarithm of the expected number of different labellings Δ_{F_i} that F_i can realise on q points sampled from the distribution $\mathbf{P}_\mathsf{U}$ of the Universum:

Two possible ways to use a Universum: Maximise the VC-entropy on it, or maximise the number of contradictions on it

$$H^{F_i}_{\mathbf{P}_\mathsf{U}}(q) \quad = \quad \log \mathbf{E}_{(\mathsf{U})^q}[\Delta_{F_i}(\mathsf{U}_1, ..., \mathsf{U}_q)].$$

The VC entropy is a quantity that depends on the considered set of functions F_i, the distribution of the data points P_U and the number of sampled points q. It measures the diversity of functions from the considered class F_i. Vapnik gives the following intuition for using this approach [98]: When being concerned with a specific learning problem, the resulting classifier should avoid making a statement about other learning problems from the same category. In some sense, it should be as unspecific as possible about any other problem than the one it is being trained on. Therefore, specifying a Universum takes the role of using prior knowledge from the prior distribution of functions P_F, as in the Bayesian approach, by preferring those equivalence classes that are most unspecific about any other problem that could be defined on the Universum points.

Intuition: Be unspecific about any other problem than the one that is being solved.

Unfortunately, the employment of the VC entropy suffers from the fact that the distribution P_U of the Universum is unknown. Vapnik [98] proposes to relax the measure by replacing the VC-entropy by the number of contradictions an equivalence class produces on the given Universum set $\mathfrak{U}$. A point $z \in \mathfrak{U}$ counts as *contradiction* if there are two functions in F_i that classify z in two different ways. An immediate consequence is that the maximal number of contradictions is bound from above by $|\mathfrak{U}|$. In that respect, it agrees with $H^{F_i}_{\mathsf{P}_\mathsf{U}}(q)$ since the maximal number of different labellings F_i can realise on q points is 2^q, therefore $\mathbf{E}_{(\mathsf{U})^q}[\Delta_{F_i}(\mathsf{U}_1, ..., \mathsf{U}_\mathsf{q})] \leq 2^q$, and thus $\log \mathbf{E}_{(\mathsf{U})^q}[\Delta_{F_i}(\mathsf{U}_1, ..., \mathsf{U}_\mathsf{q})] \leq q$. Now, the goal of structural risk minimisation with a Universum is to choose the element S^* of the structure that has low training error and *maximal number of contradictions*[1] on $\mathfrak{U}$ since the goal is to have maximal diversity on $\mathfrak{U}$. The maximal diversity can be seen as making no statement about the labels of the points in $\mathfrak{U}$. But even this simplification might still be hard to implement into a learning algorithm. The next section presents a further approximation as proposed by [103].

Maximum number of contradictions is used to approximate the VC-entropy

2.2 Implementing the Universum into SVMs

This section describes the implementation of the Universum idea. Although, in principle, the implementation into many algorithms is con-

[1]One might be tempted to relate the number of contradictions as proposed by Vapnik with the philosophy of Karl Popper [72, 71], who classifies a theory according to the possible number of sentences that can be derived from it and that might falsify it. Popper argues that empirical sciences should aim for theories that are potentially easy to falsify due to the large number of predictions that can be deduced from this theory which might *contradict* empirical observations, since those are the theories with maximal empirical content. However, this comparison is not justified, since Popper refers to possible contradicting observations from the *same* distribution of data the theory is making predictions about and not another distribution as the one the Universum set is sampled from as in the case of Vapnik. Furthermore Popper is referring to the potential possibility of the theory to contradict actual observations and not the actual contradiction on observations as given by a Universum set.

ceivable, following [103], the Support Vector Machine [5] is chosen to serve as a model of a learning algorithm. In section 2.4 certain properties of the described algorithms and relations to already existing algorithms are discussed in order to improve the understanding of the algorithm and its effects on data shown in chapter 3.

2.2.1 Support Vector Machine Implementations of the Universum

Support Vector Machines are optimisation problems over a squared norm regulariser and a sum of independent loss terms

The general Support Vector Machine is a supervised learning algorithm for classification [5] and regression [22]. In the following, only the classification version is considered. Given a set of labelled data $\mathfrak{L} = \{(x_i, y_i)\}_{i=1,\dots,m}$ the SVM can be stated as a minimisation problem consisting of a regularisation and an empirical loss term in the style of the Frequentist learning problems shown in 1.5.1:

$$\text{minimise}_{h\in\mathcal{H}} \qquad ||h||^2_{\mathcal{H}} + C\sum_{i=1}^{m}\ell(h(x_i), y_i), \tag{2.5}$$

where $\mathcal{H}$ is a Hilbert space of functions and ℓ is a loss function. There are different versions of SVMs, depending on the loss function ℓ. Two versions will be derived for the binary case when $y_i \in \mathcal{R} = \{-1, +1\}$. For binary classification the hypothesis h^* minimising (2.5) is thresholded to get the actual prediction function $\hat{h}(x) = \text{sgn}(h^*(x))$.

Using the kernel trick: Derive all SVMs for $\mathbb{R}^n$ in terms of dot products and change the Hilbert space afterwards by replacing all dot products by kernels

If $\mathcal{H}$ is the space of all *linear* or *affine functions* on $\mathbb{R}^n$, $\mathcal{H}$ is just $\mathbb{R}^n$ itself (see e.g. [59] for self duality of Hilbert spaces) since any element $\mathbf{w} \in \mathbb{R}^n$ corresponds to a linear function on $\mathbb{R}^n$ by fixing one argument of the canonical dot product of $\mathbb{R}^n$ to $\mathbf{w}$, i.e. $\mathcal{H}_{linear} = \{\langle \mathbf{w}, \cdot\rangle | \mathbf{w} \in \mathbb{R}^n\}$. Adding a constant $b \in \mathbb{R}$ to each function yields the space of all affine functions on $\mathbb{R}^n$: $\mathcal{H}_{affine} = \mathcal{H}_{linear} \oplus \mathbb{R} = \{\langle \mathbf{w}, \cdot\rangle + b | \mathbf{w} \in \mathbb{R}^n, \, b \in \mathbb{R}\}$. This is valid for all Hilbert spaces. The space $\mathcal{H}^*$ of linear functions on a Hilbert space $\mathcal{H}$, called *dual space*, is just the Hilbert space itself, independent of the specific choice of $\mathcal{H}$. Hence, the strategy will be to derive all Support Vector Machines for $\mathbb{R}^n$ and replace all dot products by an arbitrary kernel in the end. This strategy yields an SVM formulation for any RKHS. Another advantage of formulating the SVM in its linear form is the geometrical interpretation of maximising the distance or *margin* of a separating hyperplane $h = \langle \mathbf{w}, \cdot\rangle + b$ to the closest example of the training set. Although this interpretation generalises to arbitrary Hilbert spaces, it is less intuitive in the latter case.

2.2.1.1 Hinge Loss Universum

Naïve hard margin SVM

Given a set of labelled examples $\mathfrak{L} = \{(x_i, y_i)\}_{i=1,\dots,m}$, the objective of an SVM is to find a hyperplane $h(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x}\rangle + b$ that classifies all training examples correctly and has *maximal margin*. Since the prediction function will be the sign of the function output $h(\mathbf{x})$, the goal is to put all positive example on the side of the hyperplane the normal vector $\mathbf{w}$ points to, and all negative example on the other side, while keeping the closest examples in $\mathfrak{L}$ as far apart from the hyperplane as possible. Since the distance of a point $\mathbf{x}$ to a hyperplane $E_{\mathbf{w}} := \{\mathbf{x}'|\langle\mathbf{w},\mathbf{x}'\rangle + b = 0\}$ is given by $d(\mathbf{x}, E_{\mathbf{w}}) = \frac{1}{||\mathbf{w}||}(\langle\mathbf{w},\mathbf{x}\rangle + b)$, a simple SVM can be stated as the following constrained optimisation problem:

$$\begin{aligned} \text{maximise}_{\mathbf{w},b} \quad & \gamma \\ \text{s.t.} \quad & \frac{y_i}{||\mathbf{w}||}(\langle\mathbf{w},\mathbf{x}_i\rangle + b) \geq \gamma \text{ for all } 1 \leq i \leq m \\ & \gamma \geq 0. \end{aligned}$$

All training examples are correctly classified if and only if $\frac{y_i}{||\mathbf{w}||}(\langle\mathbf{w},\mathbf{x}_i\rangle + b) \geq 0$ for all $\mathbf{x}_i$. Thus, the above formulation does not allow any misclassification of the training examples. This kind of formulation is also called *hard margin SVM* since the distance of the examples to the hyperplane cannot be smaller than the margin. A drawback of this formulation is the norm in the denominator of the constraints which makes the constraint difficult to handle. Fortunately, a simple observation allows to rephrase the problem into a simpler version: If the goal is to maximise $\frac{y_i}{||\mathbf{w}||}(\langle\mathbf{w},\mathbf{x}_i\rangle + b)$, then γ can equivalently be set to some fixed value, e.g. $\gamma = 1$, and $||\mathbf{w}||$ can be minimised instead. By replacing $||\mathbf{w}|| = \sqrt{\langle\mathbf{w},\mathbf{w}\rangle}$ by $\frac{1}{2}||\mathbf{w}||^2$ it is possible to get rid of the square root while not fundamentally changing the optimisation problem. This removes all problematic parts and yields the following formulation: *Hard margin SVM*

$$\begin{aligned} \text{minimise}_{\mathbf{w},b} \quad & \frac{1}{2}||\mathbf{w}||^2 \\ \text{s.t.} \quad & y_i(\langle\mathbf{w},\mathbf{x}_i\rangle + b) \geq 1 \text{ for all } 1 \leq i \leq m. \end{aligned}$$

This version also uncovers the regulariser $||\mathbf{w}||^2$. There is only one step left to arrive at the final SVM formulation. In the current form, all training examples are forced to be classified correctly. This is like penalising misclassifications with infinite loss. Due to noise, an exact separation is clearly not possible in most practical problems. Therefore, all constraints are relaxed by introducing *slack variables* $\xi_i \geq 0$, one for each training example, that allow for violations of the margin but are minimised at the same time, leading to the following optimisation problem: *Soft margin SVM*

$$\begin{aligned} \text{minimise}_{\mathbf{w},b} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m}\xi_i \\ \text{s.t.} \quad & y_i(\langle\mathbf{w},\mathbf{x}_i\rangle + b) \geq 1 - \xi_i \text{ for all } 1 \leq i \leq m \\ & \xi_i \geq 0. \end{aligned}$$

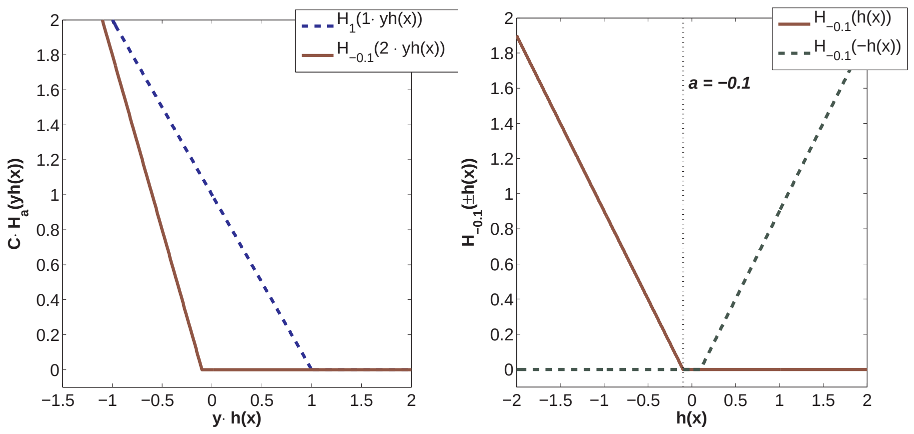


Figure 2.1: *Left:* Hinge loss with regularisation constant $C \cdot H_a(yh(x)) = C \cdot \max(0, a - yh(x))$ for different offsets a and values of C. *Right:* Universum loss $U_a(h(x)) = H_a(h(x)) + H_a(-h(x))$ composed of two hinge losses. The value of the offset $a = -\varepsilon = -0.1$ is depicted by the vertical dotted line.

SVMs minimise the hinge loss on margin violations

Here, C is a constant chosen a priori to specify the cost of each margin violation. The term $C\sum_{i=1}^{m}\xi_i$ and the constraints implicitly define a loss function called *hinge loss:* $H_a(t) = \max(0, a - t)$ with $a = 1$. The hinge loss is depicted in figure *2.1.* It penalises the amount ξ_i by which a training example x_i violates the margin, i.e. the amount $y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b)$ is smaller than one. The violation is exactly the value of ξ_i. By plugging the margin violation ξ directly into H_1, the SVM problem can be stated without constraints. This notation will be used further for the sake of lucidity. Using $h_{\mathbf{w},b}(\mathbf{x}) := \langle \mathbf{w}, \mathbf{x}\rangle + b$ the SVM problem can be compactly formulated as:

$$\text{minimise}_{\mathbf{w},b} \qquad \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m} H_1(y_i h_{\mathbf{w},b}(\mathbf{x}_i)) \tag{2.6}$$

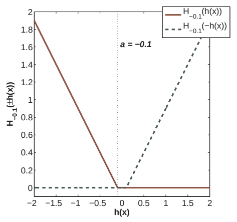
A convenient property of the SVM formulation (2.6) is its convexity. Since a (squared) norm is always convex, the hinge loss is a convex function, and convex functions are closed under addition (see [8]). This implies that the optimisation problem (2.6) always has a unique global minimum.

Maximising the number of contradictions is relaxed into bringing the separating hyperplane close to the Universum points.

Now, how does the set of Universum points $\mathfrak{U}$ enter the algorithm? As already noted in 2.1.2 a quantity like the number of contradictions is hard to optimise. Therefore [103] propose to approximate the original idea as formulated by Vapnik [100, 98] by forcing the hyperplane to be close to the Universum points $\mathbf{z}_1, ..., \mathbf{z}_q \in \mathfrak{U}$, i.e. to enforce h to have small absolute values on the Universum points.

This relaxation is motivated by two intuitions about the original Universum idea. Firstly, the number of contradictions should be maximised. If the Universum points $\mathbf{z}_j$ can be put close to the separating hyperplane, small variations in the orientation of $\mathbf{w}$ will probably let some of the $\mathbf{z}_j$ change the side of the hyperplane and therefore induce a change of sign of $h(\mathbf{z}_j)$. However, the classification of the training points is likely to remain unchanged, since their distance to the hyperplane was maximised. Therefore, there are functions that realise the same labelling on the training points but different labels on the Universum points. Since all $\mathbf{z}_j$ that change the side of the hyperplane count as contradiction, choosing a hyperplane which is close to the Universum points can be seen as a heuristic for maximising the number of contradictions. Secondly, the idea behind using a Universum is that the learning algorithm should not make a statement about the points in $\mathfrak{U}$. But if the strength of the statement about a data point is measured by the distance to the hyperplane, the area around the hyperplane is exactly the place where the resulting hyperplane does not make a strong statement since points that are far away from the hyperplane are also deep inside one of the half-spaces associated with one of the two classes, therefore making a strong statement about the label.

It is important to note that this approximation does not use a set of unlabelled test points as in the original idea. The approximation simply consists of finding a separating hyperplane which is close to the points in the Universum set $\mathfrak{U}$ at the same time. Therefore, the Universum algorithm as proposed by [103] is an inductive rather than a transductive algorithm.



The ε-insensitive loss is used for the Universum points

Bringing the hyperplane close to the Universum points can easily be implemented into an SVM by adding another loss term $C_{\mathfrak{U}} \sum_{j=1}^{q} U_a(h(\mathbf{z}_j))$ to (2.6). $U_a(h(\mathbf{z}))$ is a loss function that is zero within a tube around the hyperplane and increases linearly outside of that tube. Analogously to that of the hinge loss, the slope of the loss is controlled by a regularisation constant $C_{\mathfrak{U}}$. In support vector regression, the loss U_a is known as *ε-insensitive loss [22, 89],* since the loss is insensitive to small variations inside a tube of width 2ε around the hyperplane. The right graphic in figure 2.1 shows the ε-insensitive loss. It also reveals that the ε-insensitive loss can be expressed as the sum of two hinge losses with one of them being reflected on the y-axis:

$$U_a(h(\mathbf{z})) \quad = \quad H_a(h(\mathbf{z})) + H_a(-h(\mathbf{z})). \tag{2.7}$$

After adding the loss term for the Universum points, the SVM formulation (2.6) turns into

$$\text{minimise}_{\mathbf{w},b} \qquad \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m} H_1(y_i h_{\mathbf{w},b}(\mathbf{x}_i)) + C\sum_{j=1}^{q} U_a(h_{\mathbf{w},b}(\mathbf{z}_j)) \tag{2.8}$$

Before deriving the final optimisation problem that is amenable for the

The ε-insensitive loss can be simulated by adding the Universum points twice with opposite labels use of kernels, it is useful to point out another observation that will facilitate the derivation of the problem. When looking at equation (2.7), one can make the simple transformation

$$\begin{aligned} U_a(h(\mathbf{z})) &= H_a(h(\mathbf{z})) + H_a(-h(\mathbf{z}) \\ &= \sum_{y\in\{+1,-1\}} H_a(y \cdot h(\mathbf{z})). \end{aligned}$$

This means that penalising $h(\mathbf{z})$ with the ε-insensitive loss is equivalent to adding z twice to the training set, each time with opposite labels. The only difference to the labelled points $\{(\mathbf{x}_i, y_i)\}_{i=1,\dots,m}$ is that the hinge loss is shifted by $a = -\varepsilon$ for a tube of width 2ε instead of $a = 1$ for the normal labelled examples (see figure 2.1). In the following, $y_j^{(u)}$ denote the fake labels of a Universum point and $y_i^{(l)}$ the real labels from the training set.

Rephrasing equation (2.8) with constraints in terms of slack variables ξ_i, ϑ_j yields the following optimisation problem:

$$\begin{aligned} \text{minimise}_{\mathbf{w},b} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m}\xi_i + C_{\mathfrak{U}}\sum_{j=1}^{q}\vartheta_j \\ \text{s.t.} \quad & y_i^{(l)}(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) \geq 1 - \xi_i \text{ for all } 1 \leq i \leq m \\ & |(\langle \mathbf{w}, \mathbf{z}_j\rangle + b)| \leq \varepsilon + \vartheta_j \text{ for all } 1 \leq j \leq q \\ & \xi_i \geq 0 \text{ for all } 1 \leq i \leq m \\ & \vartheta_j \geq 0 \text{ for all } 1 \leq j \leq q. \end{aligned}$$

Replacing each $\mathbf{z}_j$ by two samples $(\mathbf{z}_j, +1)$ and $(\mathbf{z}_j, -1)$ and expanding the constraints $|(\langle \mathbf{w}, \mathbf{z}_j\rangle + b)| \leq \varepsilon + \vartheta_j$ into two constraints for each $\mathbf{z}_j$, the optimisation problem can equivalently be stated as:

$$\begin{aligned} \text{minimise}_{\mathbf{w},b} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m}\xi_i + C_{\mathfrak{U}}\sum_{j=1}^{2q}\vartheta_j \qquad (2.9)\\ \text{s.t.} \quad & y_i^{(l)}(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) \geq 1 - \xi_i \text{ for all } 1 \leq i \leq m \\ & (\langle \mathbf{w}, \mathbf{z}_j\rangle + b) \geq -\varepsilon - \vartheta_j \text{ for all } 1 \leq j \leq q \\ & -(\langle \mathbf{w}, \mathbf{z}_j\rangle + b) \geq -\varepsilon - \vartheta_j \text{ for all } q+1 \leq j \leq 2q \\ & \xi_i \geq 0 \text{ for all } 1 \leq i \leq m \\ & \vartheta_j \geq 0 \text{ for all } 1 \leq j \leq 2q. \end{aligned}$$

This optimisation problem can be solved using *Lagrange multipliers* (see [8] for an introduction on optimisation). The corresponding

Lagrangian is

$$\begin{aligned} g(\boldsymbol{\lambda}, \boldsymbol{\upsilon}, \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\upsilon}}, \mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\vartheta}) &= \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m}\xi_i + C_{\mathfrak{U}}\sum_{j=1}^{u}\vartheta_j \\ &\quad - \sum_{i=1}^{m}\lambda_i[y_i^{(l)}(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) - 1 + \xi_i] \\ &\quad - \sum_{j=1}^{2q}\upsilon_j[y_j^{(u)}(\langle \mathbf{w}, \mathbf{z}_j\rangle + b) + \varepsilon + \vartheta_j] \\ &\quad - \sum_{i=1}^{m}\tilde{\lambda}_i\xi_i - \sum_{j=1}^{2q}\tilde{\upsilon}_j\vartheta_j \end{aligned} \tag{2.10}$$

The constrained optimisation problem is solved using Lagrange multipliers

with $y_j^{(u)} = 1$ for all $1 \leq j \leq q$ and $y_j^{(u)} = -1$ for all $q+1 \leq j \leq 2q$. Equation (2.10) has to be minimised with respect to the so called *primal variables* $\mathbf{w}$, b and maximised with respect to the Lagrange multipliers $\boldsymbol{\lambda}, \tilde{\boldsymbol{\lambda}}, \boldsymbol{\upsilon}, \tilde{\boldsymbol{\upsilon}}$, also termed *dual variables.* Alternatively, just as in the derivation of standard SVMs [9, 81], it is possible to derive conditions for the optimal primal variables and plug them back into the Lagrangian. Since the problem is convex, the optimal values for $\mathbf{w}$ and b can be specified uniquely. After substituting the optimal values for $\mathbf{w}$ and b the only variables left are the dual variables. This step is important for replacing the standard dot products by kernels later. The partial derivatives with respect to the primal variables yield their optimality conditions [66]:

$$\frac{\partial g}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{m} y_i^{(l)}\lambda_i\mathbf{x}_i - \sum_{j=1}^{2u} y_j^{(u)}\upsilon_j\mathbf{z}_j \quad\Rightarrow\quad \mathbf{w} = \sum_{i=1}^{m} y_i^{(l)}\lambda_i\mathbf{x}_i + \sum_{j=1}^{2u} y_j^{(u)}\upsilon_j\mathbf{z}_j \tag{2.11}$$

$$\frac{\partial g}{\partial b} = -\sum_{i=1}^{m} y_i^{(l)}\lambda_i - \sum_{j=1}^{2u} y_j^{(u)}\upsilon_j \quad\Rightarrow\quad \sum_{i=1}^{m} y_i^{(l)}\lambda_i + \sum_{j=1}^{2u} y_j^{(u)}\upsilon_j = 0 \tag{2.12}$$

$$\frac{\partial g}{\partial \xi_i} = C - \lambda_i - \tilde{\lambda}_i \quad\Rightarrow\quad C - \lambda_i - \tilde{\lambda}_i = 0 \tag{2.13}$$

$$\frac{\partial g}{\partial \vartheta_j} = C_{\mathfrak{U}} - \upsilon_j - \tilde{\upsilon}_j \quad\Rightarrow\quad C_{\mathfrak{U}} - \upsilon_j - \tilde{\upsilon}_j = 0. \tag{2.14}$$

The dual formulation uses only dot products on the training and Universum examples

Condition (2.11) shows that $\mathbf{w}$ is a linear combination of training and Universum points. This will be useful when recovering the learnt function from the dual formulation derived next. Resubstituting the value for $\mathbf{w}$ back into the Langrangian (2.10), setting $\tilde{\mathbf{x}}_i := \mathbf{x}_i$ for $1 \leq i \leq m$, $\tilde{\mathbf{x}}_{j+m} := \mathbf{z}_j$ for $1 \leq j \leq 2q$, $\boldsymbol{\alpha} = \begin{pmatrix}\boldsymbol{\lambda}\\ \boldsymbol{\upsilon}\end{pmatrix}$, $\boldsymbol{\varrho} = \begin{pmatrix}\mathbf{1}\\ -\boldsymbol{\varepsilon}\end{pmatrix}$, $\boldsymbol{y} = \begin{pmatrix}\boldsymbol{y}^{(l)}\\ \boldsymbol{y}^{(u)}\end{pmatrix}$, and using the linearity of the dot product yields the *dual formulation*

$$\begin{aligned} g(\boldsymbol{\alpha}) &= \sum_{i=1}^{m+2q} \varrho_i \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m+2q} y_i y_j \alpha_i \alpha_j \langle \tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j \rangle \qquad (2.15) \\ \text{s.t.} & \sum_{i=1}^{m+2q} y_i \alpha_i = 0 \\ & 0 \leq \alpha_i \leq C \text{ for all } 1 \leq i \leq m \\ & 0 \leq \alpha_i \leq C_{\mathfrak{U}} \text{ for all } m+1 \leq i \leq m+2q. \end{aligned}$$

In the dual formulation (2.15), the optimality conditions (2.13) and (2.14) are enforced by bounding each λ_i and each υ_j by zero from below and by C or $C_{\mathfrak{U}}$ from above, respectively. The value of the Lagrange multipliers $\tilde{\lambda}_i$ and $\tilde{\upsilon}_j$ are implied by the values of C, $C_{\mathfrak{U}}$ and $\boldsymbol{\alpha}$. The condition (2.12) was not resubstituted into (2.10) and is therefore not enforced by default. This is why it is added as a constraint to (2.15). Since (2.15) is merely a function of the Langrange multipliers $\boldsymbol{\alpha}$, only the maximisation part is left. Once the maximising $\boldsymbol{\alpha}^*$ are found, the resulting function is simply

The soft margin is expressed as box constraints on the Lagrange multipliers

$$\begin{aligned} h(\mathbf{x}) &= \langle \mathbf{w}, \mathbf{x} \rangle + b \qquad (2.16) \\ &= \langle \sum_{i=1}^{m+2q} \alpha_i^* y_i \tilde{\mathbf{x}}_i, \mathbf{x} \rangle + b \\ &= \sum_{i=1}^{m+2q} \alpha_i^* y_i \langle \tilde{\mathbf{x}}_i, \mathbf{x} \rangle + b, \end{aligned}$$

where the offset b can be obtained from the fact that all Langrange multipliers $\boldsymbol{\alpha}^*$ that are strictly between zero and C or $C_{\mathfrak{U}}$, respectively, belong to so called *active* constraints, i.e. $(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) = 1$ or $(\langle \mathbf{w}, \mathbf{z}_j \rangle + b) = -\varepsilon$ for all $1 \leq i \leq m$ with $0 < \alpha_i^* < C$ or all $m+1 \leq j \leq m+2q$ with $0 < \alpha_j^* < C_{\mathfrak{U}}$. This hinge loss formulation of an SVM using the Universum will be called $\mathfrak{U}$-SVM from now on.

The learnt function is a sum over kernel functions

The important property of the dual formulation (2.15) and the resulting function (2.16) is that the data points enter it exclusively in dot products $\langle \cdot, \cdot \rangle$. As already mentioned in the beginning, this dot product can now be replaced by an arbitrary kernel function $k : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ representing the dot product in the RKHS associated with it. In this case, the elements of the training set $\mathfrak{L}$ and the Universum set $\mathfrak{U}$ do not even need to be elements in a space since the space structure is imposed by the employed kernel k. When denoting the *Gram matrix* of k with $(\mathbf{K})_{ij} = k(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j)$ for $1 \leq i, j \leq m+2q$ and merging the labels y_i and the Lagrange multipliers α_i into one variable β_i, the $\mathfrak{U}$-SVM formulation

When merging the labels into the Lagrange multipliers, the $\mathfrak{U}$-SVM can be compactly written as a quadratic optimisation problem can be written compactly in matrix notation as

$$\begin{aligned} \text{maximise} \quad & \boldsymbol{\varrho}^\top \boldsymbol{\beta} - \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{K}\boldsymbol{\beta} \\ \text{s.t.} \quad & \mathbf{y}^\top \boldsymbol{\beta} = 0 \\ & 0 \leq y_i \beta_i \leq C \text{ for all } 1 \leq i \leq m \\ & 0 \leq y_i \beta_i \leq C_{\mathfrak{U}} \text{ for all } m < i \leq m + 2q. \end{aligned}$$

Examples with nonzero α determine the solution and are called Support Vectors

An important property of the SVM and $\mathfrak{U}$-SVM as described above is the sparseness of its solution. Equation (2.16) reveals that only those elements $\tilde{\mathbf{x}}_i$ contribute to h that have nonzero α_i. These points are called *support vectors.* Usually only a part of the training and Universum set eventually become support vectors. This sparseness is favorable since it speeds up the computation for the resulting function, because only the kernels between the support vectors and the test points have to be computed.

The sparseness property can be seen by looking at the hinge loss functions in figure 2.1. During the optimisation of the ($\mathfrak{U}$)-SVM the gradient of the loss is zero for all vectors $\tilde{\mathbf{x}}_i$ for which the function value lies in a flat region of the loss function, i.e. which have $h(\mathbf{x}) > 1$. That means that the only gradient information for the optimiser comes from the regulariser $||\mathbf{w}||^2 = \boldsymbol{\beta}^\top \mathbf{K}\boldsymbol{\beta}$. But since the optimiser is minimising $||\mathbf{w}||^2$, the best thing it can do is to shrink the α_i corresponding to $\tilde{\mathbf{x}}_i$ to zero. Thus, all elements $\tilde{\mathbf{x}}_i$ that lie in a region with constant loss will not contribute to the solution, i.e. not become support vectors.

The $\mathfrak{U}$-SVM can be efficiently solved with almost all SVM optimizers.

To conclude the section about the hinge loss formulation[2], a few comments about the actual implementation are necessary. A nice feature of the $\mathfrak{U}$-SVM algorithm is its similarity to the formulation of a standard SVM. The only differences are the linear part $\boldsymbol{\varrho}^\top \boldsymbol{\beta}$ of the objective function and different regularisation constants C and $C_{\mathfrak{U}}$. Each SVM optimiser that allows to change these parts of the SVM formulation is also capable of solving the $\mathfrak{U}$-SVM. Doubling the amount of Universum points by introducing them twice with different labels might seem inefficient, but the kernel matrix only needs to be computed for the original Universum set $\mathfrak{U}$ and the training set $\mathfrak{L}$. It is then easy to write a wrapper function that mimics the kernel matrix for the double part of the Universum points by referring to the already computed kernel values. Nevertheless, the number of variables to optimise remains doubled. This might slow down the learning process for large Universum sets $\mathfrak{U}$.

[2]Implementations in C (the UniverSVM) and Matlab can be downloaded from my website *http://www.tuebingen.mpg.de/~fabee.*

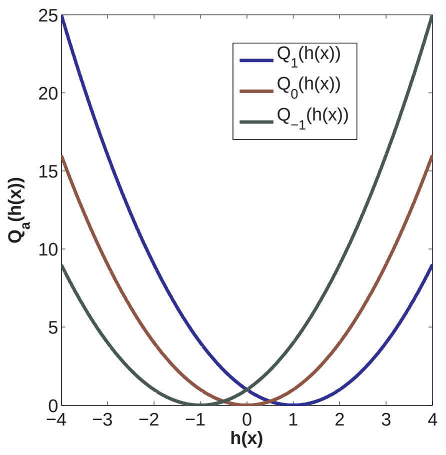
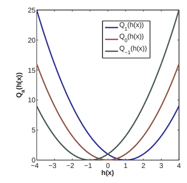


Figure 2.2: Quadratic loss functions $Q_a(t) = (t - a)^2$: The two displayed functions are Q_0 and Q_1 .

2.2.1.2 Least Squares Universum



$\mathfrak{U}_{\mathfrak{ls}}$-SVM is a $\mathfrak{U}$-SVM with the hinge loss replaced by the quadratic loss

A variation of the $\mathfrak{U}$-SVM can be obtained by replacing the hinge loss by the quadratic loss. This version will be needed in section 2.4.2 to show the connection between this learning machine and certain types of feature extraction algorithms. The quadratic loss SVM without Universum is usually called *least squares SVM (LS-SVM)* and was introduced by [97, 96]. For the sake of short notation the least squares $\mathfrak{U}$-SVM will be referred to as $\mathfrak{U}_{\mathfrak{ls}}$-SVM in the following. The main difference between the least squares SVM and a normal SVM is that the least squares SVM requires $\langle \mathbf{w}, \mathbf{x}\rangle + b = 1$ instead of $\langle \mathbf{w}, \mathbf{x}\rangle + b \geq 1$. This means that the least squares SVM penalises every deviation from the margin, no matter in which direction. In contrast, the hinge loss only penalises deviations into the margin. As before, the points in $\mathfrak{U}$ will be required to lie on the hyperplane. The three loss functions are depicted in figure 2.2. Since the quadratic loss is convex, too, the $\mathfrak{U}_{\mathfrak{ls}}$-SVM is again a convex optimisation problem.

The $\mathfrak{U}_{\mathfrak{ls}}$-SVM can be derived analogously to the $\mathfrak{U}$-SVM: Constrained optimisation problem, Lagrange multipliers, optimality conditions, dual formulation

In the following, analogous steps as in section 2.2.1.1 are taken to derive the finale version of the $\mathfrak{U}_{\mathfrak{ls}}$-SVM. Starting from a general formulation in terms of a loss function and a regulariser, slack variables are introduced and a Lagrangian is formulated. Optimality conditions for the primal variables of the Lagrangian are derived. Their substitution into the Lagrangian gives the final dual formulation.

Using quadratic loss functions $Q_a(t) = (t - a)^2$, the general optimisation problem for the $\mathfrak{U}_{\mathfrak{ls}}$-SVM can be stated as

$$\text{minimise}_{\mathbf{w},b}\ \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{2}\sum_{i=1}^{m} Q_{y_i^{(l)}}(h_{\mathbf{w},b}(\mathbf{x}_i)) + \frac{C_{\mathfrak{U}}}{2}\sum_{j=1}^{q} Q_0(h_{\mathbf{w},b}(\mathbf{z}_j)). \quad (2.17)$$

Here, quadratic loss is put on $h_{\mathbf{w},b}(\mathbf{x}_i)$ instead of $y_i h_{\mathbf{w},b}(\mathbf{x}_i)$ as in the hinge loss formulation (2.8). So, instead of requiring $y_i h_{\mathbf{w},b}(\mathbf{x}_i) = 1$, it is required that $h_{\mathbf{w},b}(\mathbf{x}_i) = y_i$. This is just a minor technical difference, though. One could also derive the $\mathfrak{U}_{\mathfrak{ls}}$-SVM with the constraint $y_i h_{\mathbf{w},b}(\mathbf{x}_i) = 1$. Unlike before, the Universum points are not added twice to the problem. Since the quadratic loss does not have an area of zero loss around zero, it is more appropriate to enforce the target value $y^{(u)} = 0$ for all Universum points $\{\mathbf{z}_j\}_{j=1,\ldots,q}$. As before, the loss functions are expressed in terms of slack variables

$$\begin{aligned} \text{minimise}_{\mathbf{w},b} \quad & ||\mathbf{w}||^2 + \frac{C}{2}\sum_{i=1}^{m}\xi_i^2 + \frac{C_{\mathfrak{U}}}{2}\sum_{j=1}^{q}\vartheta_j^2 \qquad (2.18) \\ \text{s.t.} \quad & \langle \mathbf{w}, \mathbf{x}_i\rangle + b = y_i^{(l)} - \xi_i \\ & \langle \mathbf{w}, \mathbf{z}_j\rangle + b = 0 - \vartheta_j. \end{aligned}$$

This yields the Lagrangian

$$\begin{aligned} g(\mathbf{w}, b, \boldsymbol{\lambda}, \boldsymbol{\upsilon}) &= \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{2}\sum_{i=1}^{m}\xi_i^2 + \frac{C_{\mathfrak{U}}}{2}\sum_{j=1}^{q}\vartheta_j^2 \qquad (2.19) \\ & \quad - \sum_{i=1}^{m}\lambda_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b - y_i^{(l)} + \xi_i) - \sum_{j=1}^{q}\upsilon_j(\langle \mathbf{w}, \mathbf{z}_j\rangle + b + \vartheta_j). \end{aligned}$$

Calculating the derivatives of g with respect to the primal variables and setting them to zero yields the following optimality conditions

$$\begin{aligned} \frac{\partial}{\partial \mathbf{w}} g = \mathbf{w} - \sum_{i=1}^{m}\lambda_i \mathbf{x}_i - \sum_{j=1}^{q}\upsilon_j \mathbf{z}_j \quad &\Rightarrow \quad \mathbf{w} = \sum_{i=1}^{m}\lambda_i \mathbf{x}_i + \sum_{j=1}^{q}\upsilon_j \mathbf{z}_j \\ \frac{\partial}{\partial b} g = -\sum_{i=1}^{m}\lambda_i - \sum_{j=1}^{q}\upsilon_j \quad &\Rightarrow \quad \sum_{i=1}^{m}\lambda_i + \sum_{j=1}^{q}\upsilon_j = 0 \\ \frac{\partial}{\partial \xi_i} g = C\xi_i - \lambda_i \quad &\Rightarrow \quad \lambda_i = C\xi_i \\ \frac{\partial}{\partial \vartheta_j} g = C_{\mathfrak{U}}\vartheta_j - \upsilon_j \quad &\Rightarrow \quad \upsilon_j = C_{\mathfrak{U}}\vartheta_j. \end{aligned}$$

Substituting these values into (2.19), setting $\boldsymbol{\alpha} = \binom{\boldsymbol{\lambda}}{\boldsymbol{\upsilon}}$ as before, replacing the dot products by kernel functions and defining $\mathbf{C} = \begin{pmatrix} \frac{1}{C}\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \frac{1}{C_{\mathfrak{U}}}\mathbf{I} \end{pmatrix}$, where $\mathbf{I}$ denotes the identity matrix of appropriate dimension ($m \times m$ or $q \times q$, respectively) and $\mathbf{0}$ a matrix with only zeros, yields the dual formulation in matrix notation

$$\begin{aligned} \text{maximise}_{\boldsymbol{\alpha}} \quad & \begin{pmatrix} \mathbf{y}^{(l)} \\ \mathbf{0} \end{pmatrix}^{\top} \boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^{\top}\mathbf{K}\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}\mathbf{C}\boldsymbol{\alpha} \\ \text{s.t.} \quad & \mathbf{1}^{\top}\boldsymbol{\alpha} = 0. \end{aligned}$$

This is still a convex problem in $\boldsymbol{\alpha}$. To get the optimality conditions for $\boldsymbol{\alpha}$, the derivative with respect to $\boldsymbol{\alpha}$ is set to zero. This yields the condition

$$\begin{aligned} \mathbf{K}\boldsymbol{\alpha} + \mathbf{C}\boldsymbol{\alpha} &= \begin{pmatrix} \mathbf{y}^{(l)} \\ \mathbf{0} \end{pmatrix} \\ \text{s.t.} \qquad & \mathbf{1}^\top \boldsymbol{\alpha} = 0. \end{aligned}$$

The $\mathfrak{U}_{\mathfrak{ls}}$-SVM can be written as a set of linear equations

Apart from the additional constraint, the $\mathfrak{U}_{\mathfrak{ls}}$-SVM is quite similar to the well known *Ridge Regression* learning algorithm (see e.g. [85, 65]). The linear equality constraint can easily be included as an additional linear equation, which leads to the final problem

$$\begin{pmatrix} 0 & \mathbf{1}^\top \\ \mathbf{1} & \mathbf{K}+\mathbf{C} \end{pmatrix} \begin{pmatrix} b \\ \boldsymbol{\alpha} \end{pmatrix} = \begin{pmatrix} 0 \\ \mathbf{y}^{(l)} \\ \mathbf{0} \end{pmatrix}.$$

The $\mathfrak{U}_{\mathfrak{ls}}$-SVM can be solved by a simple matrix inversion

The optimal values $\boldsymbol{\alpha}^*$ and b^* are therefore given by

$$\begin{pmatrix} b^* \\ \boldsymbol{\alpha}^* \end{pmatrix} = \begin{pmatrix} 0 & \mathbf{1}^\top \\ \mathbf{1} & \mathbf{K}+\mathbf{C} \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ \mathbf{y}^{(l)} \\ \mathbf{0} \end{pmatrix}. \tag{2.20}$$

As easily seen from (2.20), solving a $\mathfrak{U}_{\mathfrak{ls}}$-SVM involves just a simple matrix inversion instead of optimising a quadratic function as for the $\mathfrak{U}$-SVM. This makes the programming part easy, but is not quite applicable when having a lot of labelled and Universum examples since a matrix inversion is in $\mathcal{O}(d^3)$ if d is the dimension of the matrix. When dealing with large kernel matrices, other optimisation methods such as gradient descent [8] or conjugate gradient [39, 86] must be used for minimising the quadratic error

$$\left\| \begin{pmatrix} 0 & \mathbf{1}^\top \\ \mathbf{1} & \mathbf{K}+\mathbf{C} \end{pmatrix} \begin{pmatrix} b \\ \boldsymbol{\alpha} \end{pmatrix} - \begin{pmatrix} 0 \\ \mathbf{y}^{(l)} \\ \mathbf{0} \end{pmatrix} \right\|^2.$$

$\mathfrak{U}_{\mathfrak{ls}}$-SVM solutions are not sparse

Another drawback of the least squares SVM and the $\mathfrak{U}_{\mathfrak{ls}}$-SVM is that it does not yield a sparse solution, which slows down the evaluation of the resulting function. As mentioned in 2.2.1.1, sparseness is due to constant parts in the loss function. Since the quadratic loss is constant nowhere, there is no a priori reason why the α should become zero.

2.2.1.3 Multiclass Universum

A multiclass problem can be cast into a set of binary problems in several ways

The most widely used method to deal with more than two classes, i.e. $\infty > |\mathcal{R}| > 2$, is to employ several binary classifiers and use their outputs to predict the label. In general, there are many possibilities to use more than one binary classifier for multiclass classification. The most common scheme is the so called *One vs. Rest* multiclass classification scheme, where a single binary classifier is trained for each possible value $y \in \mathcal{R}$ by assigning all data points with $y_i = y$ a positive label and all others a negative one. When predicting the label of a new data point, the label corresponding to the binary classifier with the largest output value $h(x)$ is returned.

Multiclass as a decoding problem: Outputs of binary classifiers serve as bits of codewords for the different classes

Clearly, dozens of schemes can be thought of how to combine binary classifiers to a multiclass classifier. In this thesis, the decoding approach of [1] is used. The authors view the multiclass problem as a decoding problem over a ternary alphabet $\mathcal{A} = \{-1, 0, 1\}$, where each binary classifier constitutes one "bit" of the code word. The value 0 serves as a *don't care term*, meaning that the output of this classifier is not important for predicting the current class and is therefore ignored. For each multiclass scheme there is a corresponding coding matrix M. Each row of M corresponds to one element in $\mathcal{R}$, and each column corresponds to a single binary classifier. M_{ij} contains the expected output in the jth binary classification problem for the ith element of $\mathcal{R}$. For the sake of simplicity $\mathcal{R}$ is assumed to be $\mathcal{R} = \{1, ..., l\}$ from now on, where l is the cardinality of $\mathcal{R}$. To give an example: The coding matrix M for $l = 4$ and One vs. Rest scheme would be

$$\mathbf{M}_{1vsR} = \begin{pmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 \\ -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{pmatrix}.$$

Another frequently used multiclass classification scheme is the so called *one-versus-one* strategy. In One vs. One, a single classifier is trained for each pair of different values in $\mathcal{R}$, i.e. $\mathrm{M} \in \{-1, 0, 1\}^{l \times \frac{1}{2}(l-1)l}$. For the example above M would be

$$\mathbf{M}_{1vs1} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 1 & 0 \\ 0 & -1 & 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{pmatrix}.$$

Decoding: Taking the closest codeword according to some measure

Once the single binary classifiers are trained, the predictions of the single classifiers $h_1, ..., h_n$ on a new point x yield a codeword

$$\mathbf{h} = (h_1(x), ..., h_n(x))$$

when taking the real valued outputs or

$$\hat{\mathbf{h}} = (\mathrm{sgn}(h_1(x)), ..., \mathrm{sgn}(h_n(x)))$$

when thresholding them. The label of x is obtained by finding the row $\mathrm{M}_{i:}$ of M that is "closest" to $\mathbf{h}$ or $\hat{\mathbf{h}}$ according to some error measure.

Closeness is usually measured by a distance function d. The predicted label is then $\hat{y} = \text{argmin}_i d(\mathbf{M}_{i:}, \hat{\mathbf{h}})$ or $y = \text{argmin}_i d(\mathbf{M}_{i:}, \mathbf{h})$, respectively. The authors of [1] propose several distance functions. Two of them will be used in later chapters. The first and easiest one is the *Hamming distance*

Hamming Decoding

$$d_H(\mathbf{M}_{i:}, \hat{\mathbf{h}}) = \sum_{j=1}^{n} \left(\frac{1 - \hat{h}_j \mathbf{M}_{ij}}{2} \right). \tag{2.21}$$

If either $\mathbf{M}_{ij}$ or $\hat{\mathbf{h}}_j$ equals zero, the respective component contributes the constant value $\frac{1}{2}$ to the sum. The second distance function proposed by [1] is based on the loss function ℓ_j that is used by the single classifiers:

Loss based decoding

$$d'_L(\mathbf{M}_{i:}, \mathbf{h}) = \sum_{j=1}^{n} \ell_j(\mathbf{M}_{ij} \cdot \mathbf{h}_j)$$

While in principle it is possible to use a completely independent Universum set for each of the binary classifiers, it is more interesting to use all examples that are assigned labels $\mathbf{M}_{ij} = 0$ in a single binary classification task for the multiclass strategy. In this case the One vs. Rest strategy is useless, since it always uses all labelled examples for training a single classifier. The One vs. One strategy, however, is applicable since only examples of two classes get assigned a nonzero label in a single subtask. All other examples that are normally ignored for this subtask can now be used as Universum points. In this case, it is advisable to adapt the loss based distance function to a more general version that also takes into account the predictions on Universum points

One-Rest strategy is not useful for multiclass Universum

$$d_L(\mathbf{M}_{i:}, \mathbf{h}) = \sum_{j=1}^{n} \ell_j(\mathbf{M}_{ij}, \mathbf{h}_j), \tag{2.22}$$

i.e. the output $\mathbf{h}_j$ increases the distance if it deviates from its target value zero. In the distance function d'_L used in [1], the output $\mathbf{h}_j$ was not taken into account if $\mathbf{M}_{ij} = 0$ since they used the loss function on the product of both values which was then automatically set to zero.
Another distance function that will be used later and that falls into the loss based framework, is the L_1-norm

L_1 decoding

$$\begin{aligned} d_{L_1}(\mathbf{M}_{i:}, \mathbf{h}) &= ||\mathbf{M}_{i:} - \mathbf{h}||_1 \qquad (2.23) \\ &= \sum_{j=1}^{n} |\mathbf{M}_{ij} - \mathbf{h}_j|. \end{aligned}$$

This approach is theoretically less justified but works well in practice, as experimental evidence suggests in chapter 3.

2.2.2 Related work

Although [103] are the to give an algorithm that explicitly refers to the Universum idea by Vapnik [100, 98], there are some related approaches that shall be mentioned here briefly.

Forcing unused examples to zero in multiclass

[107] invented an algorithm for the One vs. One strategy in multiclass learning that exactly does what has been described at the end of the previous section. Although their algorithm is basically equivalent to the $\mathfrak{U}$-SVM formulation above, their motivation is different. The motivation for employing the examples with $\mathrm{M}_{ij} = 0$ is to get better predictions in the multiclass setting by sharpening the contrast between the different binary classifiers. In particular, they do not consider using a Universum for binary classification.

Non-examples in Bayesian interpretations for SVMs

Additionally, there are two Bayesian algorithms that refer to *non-examples* or *neither class* in the binary classification setting. [90, 91] gives a probabilistic interpretation for a standard hinge loss SVM by establishing the connection between the MAP estimate of a Gaussian process with a Gaussian prior using a covariance function k and a hinge loss based noise model similar to that described in equation (1.4) in section 1.5.1. The problem the author is faced with is that the likelihood that uses a heuristic normalisation term does not integrate to one. Therefore, a third - *the neither* - class is introduced that holds the remaining probability mass by definition. However, since he never assigns any example to this class, the relation of this approach to the Universum is rather an artifact of the need to store left over probability mass.

Zero-class model to attack the independence of the input distribution P_{X} from the latent function in Bayesian semi-supervised learning

A very similar idea can be found in [55]. They introduce a third class to deal with the problem that unlabelled examples used in semi-supervised learning do only contribute to generative models $\mathsf{P}_{\mathsf{X}|\mathsf{Y}}(x_i|y_i)$ but not to discriminative models $\mathsf{P}_{\mathsf{Y}|\mathsf{X}}(y_i|x_i)$ which are closer to Frequentist approaches for classification. As already mentioned in 1.5.2, this effect is due to the fact that the parameters of a distribution for the labels are independent of the value of the corresponding input point if the label is not observed and its value has to be marginalised out. To circumvent this problem, the authors of [55] introduce an additional - *neither* - class to introduce a stochastic dependence between the parameter and the unobserved label in the discriminative model. However, as in [90, 91], examples of this class are never used. Therefore the relation of this approach to the Universum is born from the need to introduce a stochastic dependence between labels and parameters in semi-supervised learning, rather than implementing the Universum idea as a set of actually observed non-examples.

2.3 Contradictions and Margin

One intuition that led to the $\mathfrak{U}$-SVM as proposed by [103] was that Universum points that lie close to the hyperplane give rise to contradictions when the normal w of the hyperplane is perturbed. Clearly, a hyperplane that separates the training examples with margin γ can be tilted to all directions by a certain angle without changing the labelling of the training examples. This section gives an analysis of the number of contradictions in terms of the margin γ and the minimal radius R of the ball that contains all data points. This leads to the formulation of another Universum algorithm that puts less emphasis on Universum examples that are far away from the labelled training points.

If not explicitly mentioned, the offset b is assumed to be zero throughout the whole section. Furthermore, it is assumed that the data can be separated with a nonzero margin γ and that all normals w of hyperplanes have length one. As already mentioned in section 2.2.1.1, minimising the norm while keeping the margin fixed to one is equivalent to maximising the margin γ while keeping the norm of w fixed to one.

Intuition: When separating the data with margin γ, the hyperplane w has an area on the sphere in which it can move without changing the labelling of the training examples.

Let R be the minimal radius of a ball around the origin that contains all training points $\mathfrak{L} = \{(x_i, y_i)\}_{i=1,\dots,m}$. Since $||\mathbf{w}|| = 1$, all normals w live on the sphere $\mathbb{S} \subset \mathcal{H}$. In particular, the normal of the hyperplane that separates the training data with maximal margin is represented by a point on $\mathbb{S}$. This situation leads to the following idea (depicted in figure 2.3), which is subsumed on the lemma below: If w separates the training data with margin γ, then w can be moved on the sphere $\mathbb{S}$ within a radius r without changing the labelling of the training data. Since there is no training point inside the margin, w can be moved along $\mathbb{S}$ at least until the plane $E_{\mathbf{w}} = \{\mathbf{x} \in \mathcal{H} | \langle \mathbf{w}, \mathbf{x} \rangle = 0\}$ hits the intersection of the original margin with the ball of radius R. For all movements that are so small that the hyperplane does not cross this intersection, one can be sure that the labelling of the training data is unchanged. Therefore, the margin γ determines the minimal radius r of a ball on $\mathbb{S}$ within which w can be moved. On the other hand, the number q' of Universum points inside the 2ε-area around decision boundary gives a hint how many points the hyperplance can traverse when moving its normal inside the allowed area on the sphere. As soon as a Universum point is traversed, it counts as a contradiction. Therefore, the number of contradictions can be bounded from below by a function of the margin γ, the radius R and the width 2ε of the tube containing the Universum points.

Lemma: Let $\mathbb{S}$ be the sphere in $\mathcal{H}$, i.e. $\mathbb{S} := \{h \in \mathcal{H} | \, ||h|| = \langle h, h \rangle = 1\}$ and let $\mathfrak{L} = \{(x_i, y_i)\}_{i=1,\dots,m}$ and $\mathfrak{U} = \{z_j\}_{j=1,\dots,q}$ be the set of labelled and the set of Universum examples. Furthermore, let $\{k_{x_i}\}_{i=1,\dots,m}$ and $\{k_{z_j}\}_{j=1,\dots,q}$ denote the training and Universum examples in the RKHS. Let $\mathcal{V}_{\gamma,\varepsilon} \subset \mathbb{S}$ be the subset of $\mathbb{S}$ that separates $\mathfrak{L}$ with a margin of at least

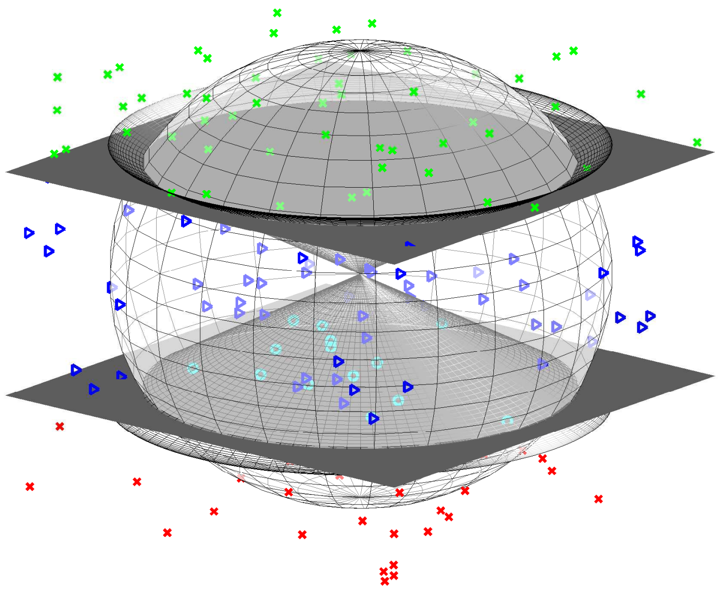


Figure 2.3: Relation between margin and number of contradictions: The green and red crosses represent training data points separated by a maximum margin hyperplane. The ball depicts the sphere $\mathbb{S}$ of all possible hyperplane normals $\mathbf{w}$ with length one. The maximum enclosing ball with radius R and the hyperplane itself are not shown. The margin is depicted by the gray planes. All points marked by circles and triangles are Universum points. The Universum points marked by blue triangles can become contradictions by tilting the hyperplane. The cone depicts the maximum tilt angle that is possible without changing the labelling of the training points.

γ and that contains all Universum examples within a tube of at most 2ε:

$$\mathcal{V}_{\gamma,\varepsilon} \quad := \quad \{\mathbf{w} \in \mathbb{S} | \max_{z \in \mathfrak{U}} |\langle \mathbf{w}, k_z \rangle| \leq \varepsilon \text{ and } \min_{1 \leq i \leq m} |\langle \mathbf{w}, k_{x_i} \rangle| \geq \gamma\}$$

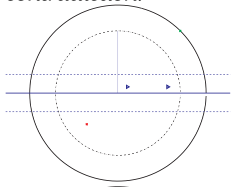
Let $R \in \mathbb{R}$ be the radius of the smallest ball around the origin that contains all elements of $\mathfrak{L}$. If $\mathcal{V}_{\gamma,\varepsilon} \neq \emptyset$, then there exists a subset $\mathcal{V}_{\gamma',\varepsilon} \subset \mathbb{S}$ with $\gamma' \leq \gamma$ such that $\mathcal{V}_{\gamma',\varepsilon}$ has at least

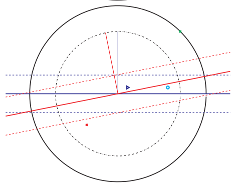
$$\begin{aligned} c \quad &\geq \quad \left|\left\{ z \in \mathfrak{U} | \, ||k_z|| \geq \frac{\varepsilon R^2}{\gamma\sqrt{R^2 - \gamma'^2} - \gamma'\sqrt{R^2 - \gamma^2}} \right\}\right| \\ &\geq \quad \left|\left\{ z \in \mathfrak{U} | \, ||k_z|| \geq \frac{\varepsilon R^2}{\sqrt{R^2 - \gamma'^2}(\gamma - \gamma')} \right\}\right| \end{aligned}$$

contradictions.

PROOF: Figure 2.4 schematically depicts the configuration of the proof's elements. Since all elements of $\mathfrak{U}$ are inside a 2ε tube around each element of $\mathcal{V}_{\gamma,\varepsilon}$, $\mathbf{w} \in \mathcal{V}_{\gamma,\varepsilon}$ has to be tilted around the origin by an angle of at least $|\omega| = |\arcsin \frac{\langle \mathbf{w}, k_z \rangle}{||k_z||}| \leq |\arcsin \frac{\varepsilon}{||k_z||}|$ in order to produce a contradiction on $z \in \mathfrak{U}$ is , with equality if and only if k_z lies directly on the tube, i.e. $|\langle \mathbf{w}, k_z \rangle| = \varepsilon$. When tilting $\mathbf{w}$ around an axis $\mathbf{v}$ by ω with[3] $\mathbf{v} \perp \mathbf{w}$ and $\mathbf{v} \perp k_z$, k_z lies directly on the hyperplane, i.e. $\langle \mathbf{w}, k_z \rangle = 0$.

Moving $\mathbf{w}$ on the sphere means tilting the hyperplane $E_{\mathbf{w}}$. If the hyperplane moves over the Universum element k_z without changing the classification of $\mathfrak{L}$ by $\mathbf{w}$, then k_z counts as a contradiction.





To make sure that the elements of $\mathfrak{L}$ are still separated by some margin $\gamma' \geq 0$, certainly not every angle ω is feasible. But since it is known that all elements of $\mathfrak{L}$ lie inside a ball with radius R and are separated by a margin of γ, one can derive an upper bound for ω such that tilting $\mathbf{w}$ by ω results in a hyperplane that still separates the elements of $\mathfrak{L}$ by a margin of at least γ'. It is now shown that the class $\mathcal{V}_{\gamma',\varepsilon}$ contains hyperplanes for generating a number of contradictions on $\mathfrak{U}$ that depends on the number of examples z having a norm larger than a certain constant.

The maximal tilt angle of the original hyperplane is upper bounded by $|\omega| \leq \arcsin \frac{\gamma}{R}$ [81, 101]. When tilting by $\arcsin \frac{\gamma}{R}$, the new tilted hyperplane $\mathbf{w}'$ intersects the ball of radius R at the intersection of the margin of $\mathbf{w}$ and the subspace span$\{\mathbf{w}, \mathbf{w}'\}$. The margin of $\mathbf{w}'$ becomes $\gamma' = 0$ in this case. Therefore, the following upper bound can be stated:

$$0 \leq \quad |\omega| \quad \leq \arcsin \frac{\gamma}{R} \leq \frac{\pi}{2}.$$

The last inequality follows from $\gamma \leq R$. In the following, it is important to keep in mind that for the interval $0 \leq |\omega| \leq \frac{\pi}{2}$, $\sin \omega$ is a strictly monotonic increasing function of ω. If a nonzero smaller margin γ' should be maintained, $|\omega|$ is upper bounded by

$$|\omega| \quad \leq \quad \arcsin \frac{\gamma}{R} - \arcsin \frac{\gamma'}{R}.$$

[3]The sign of ω is determined by the orientation of (w, k_z, v).

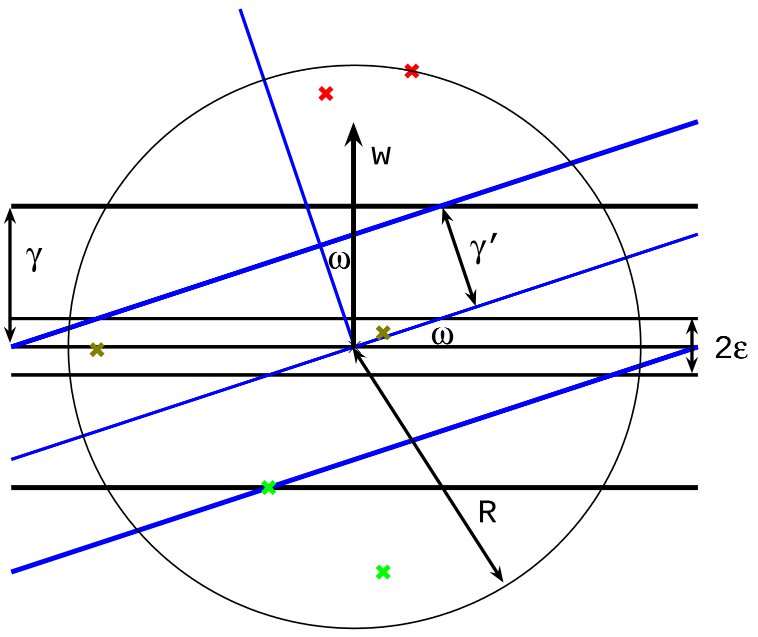


Figure 2.4: Geometrical elements of the proof: The labelled dataset, depicted by the green and red crosses, is separated by $\boldsymbol{w}$ with margin γ. All labelled examples are inside a ball of radius R around the origin. At the same time all Universum examples, depicted by the gray crosses, are located inside a tube of width 2ε around the hyperplane. When tilting $\boldsymbol{w}$ about an angle ω the resulting hyperplane, depicted by the blue lines, only separates the labelled examples with margin γ'. The further $\boldsymbol{w}$ is tilted, i.e. the larger the angle ω, the smaller is the new margin γ'.

All hyperplanes $\mathbf{w}'$ that can be produced by tilting $\mathbf{w}$ by an angle $|\omega| \leq \arcsin\frac{\gamma}{R} - \arcsin\frac{\gamma'}{R}$ are therefore an element of $\mathcal{V}_{\gamma',\varepsilon}$, and ,in particular, $\mathbf{w} \in \mathcal{V}_{\gamma',\varepsilon}$.

By strict monotonicity of $\sin\omega$ in $-\frac{\pi}{2} \leq \omega \leq \frac{\pi}{2}$,

$$\begin{aligned}
\sin\omega &\leq \sin\left(\arcsin\frac{\gamma}{R} - \arcsin\frac{\gamma'}{R}\right)\\
&= \sin\left(-\arcsin\frac{\gamma'}{R}\right)\cos\left(\arcsin\frac{\gamma}{R}\right) + \sin\left(\arcsin\frac{\gamma}{R}\right)\cos\left(-\arcsin\frac{\gamma'}{R}\right)\\
&= -\frac{\gamma'}{R}\sqrt{1-\frac{\gamma^2}{R^2}} + \frac{\gamma}{R}\sqrt{1-\frac{\gamma'^2}{R^2}}\\
&= \frac{1}{R^2}\left(\gamma\sqrt{R^2-\gamma'^2} - \gamma'\sqrt{R^2-\gamma^2}\right)
\end{aligned}$$

holds. Due to the considerations at the beginning of the proof, tilting by an angle $|\omega| > \arcsin\frac{\varepsilon}{||k_z||}$ in the appropriate direction and for the appropriate sign of ω causes a contradiction on a Universum example z. The upper bound on $|\omega|$ is chosen such that, no matter in which direction $\mathbf{w}$ is tilted, $\mathbf{w}'$ still separates the training examples with margin γ'. Therefore,the direction in which $\mathbf{w}$ is tilted does not matter. For each fixed $0 \leq |\omega| \leq \arcsin\frac{\gamma}{R} - \arcsin\frac{\gamma'}{R}$ and for each element $z \in \mathfrak{U}$ fulfilling $||k_z|| > \frac{\varepsilon}{\sin\omega}$, there exists a hyperplane $\mathbf{w}'$ (the tilted version of $\mathbf{w}$) that contradicts $\mathbf{w}$ on z. Thus, the number of contradictions c is lower bounded by

Since all Universum points lie in an ε-tube around the hyperplane and the maximal tilt angle is known, it depends on the distance of z to the origin in $\mathcal{H}$ if z can become a contradiction when tilting the plane with the maximal angle.

$$\begin{aligned}
c &\geq \left|\left\{z \in \mathfrak{U} \middle| \; ||k_z|| > \frac{\varepsilon}{\sin\omega}\right\}\right|\\
&\geq \left|\left\{z \in \mathfrak{U} \middle| \; ||k_z|| > \frac{\varepsilon R^2}{\gamma\sqrt{R^2-\gamma'^2} - \gamma'\sqrt{R^2-\gamma^2}}\right\}\right|\\
&\geq \left|\left\{z \in \mathfrak{U} \middle| \; ||k_z|| \geq \frac{\varepsilon R^2}{\sqrt{R^2-\gamma'^2}(\gamma-\gamma')}\right\}\right|
\end{aligned}$$

□

Clearly, c decreases with ε since less Universum examples fulfill the requirement $||k_z|| > \frac{\varepsilon}{\sin\omega}$. Therefore, making ε as small as possible will generate a small lower bound on $||k_z||$ and will allow for a larger number of contradictions c.

One key insight of the proof of the above lemma is that if the absolute value of the angle ω between a vector to a Universum point and the hyperplane is smaller than $|\omega| \leq \arcsin\frac{\varepsilon}{R}$, the equivalence class contains at least two functions that form a contradiction on that Universum point. This implies that the larger the distance of the Universum point to the origin, the larger is the allowed distance to the hyperplane.

This motivates another Universum algorithm implementing the maximum number of contradictions idea by Vapnik. The new Universum algorithm adapts ε of the ε-insensitive loss for each Universum point depending on its "distance to the origin".

Algorithmic implementation: Increase the width of the ε-tube with the distance to the origin.

If the radius of the smallest ball around the origin containing the training data (not necessarily the Universum) and the maximal margin γ separating the two classes is known, then it is possible to tilt the maximal margin hyperplane by a certain angle $\omega = \arcsin\frac{\gamma}{R}$ without changing the labelling of the training examples. Furthermore, every Universum example z that strictly lies inside the complement of a cone around $\mathbf{w}$ with its apex at $\mathbf{0}$ and an opening angle of ω can become a contradiction by varying $\mathbf{w}$ by at most $|\omega|$. In that sense, the loss function of the original Universum formulation is not quite correct since the Universum examples should rather lie inside the complement of a cone around $\mathbf{w}$ instead of a tube. Therefore, the ε of the ε-insensitive loss function corresponding to maximising the number of contradictions while correctly separating the data points should increase with distance to the origin along the hyperplane. This leads to the following constraint for the Universum examples:

$$\begin{aligned} \frac{|\langle \mathbf{w}, k_z\rangle|\frac{1}{||\mathbf{w}||}}{||k_z||} &\leq \sin\omega = \frac{\gamma}{R} = \frac{\frac{1}{||\mathbf{w}||}}{R} \\ |\langle \mathbf{w}, k_z\rangle| &\leq \frac{1}{R}||k_z|| := \lambda||k_z|| \end{aligned}$$

This is exactly the ε-insensitive loss with ε varying proportional to $||k_z||$.

Algorithmic implementation with offset b leads to a tricky constraint, but the constraint for $b = 0$ can serve as a lower bound.

When considering the set of affine hyperplanes, the apex of the cone is no longer at $\mathbf{0}$ but at the intersection of $\mathbf{w}$ and the hyperplane, i.e. at the point $\delta\mathbf{w}$ with $\langle \mathbf{w}, \delta\mathbf{w}\rangle + b = 0$ which is equivalent to $\delta = -\frac{b}{||\mathbf{w}||^2}$. The angle $\omega = \arcsin\frac{\gamma}{R'}$ must now be computed with a different, a smaller radius R' that depends on the position of the cone's apex. This leads to a more difficult constraint

$$\begin{aligned} \left|\frac{(\langle \mathbf{w}, k_z\rangle + b)\frac{1}{||\mathbf{w}||}}{||k_z - \delta\mathbf{w}||}\right| &\leq \frac{\frac{1}{||\mathbf{w}||}}{R'} := \rho\frac{1}{||\mathbf{w}||} \\ |\langle \mathbf{w}, k_z\rangle + b| &\leq \rho||k_z - \delta\mathbf{w}|| \\ |\langle \mathbf{w}, k_z\rangle + b| &\leq \rho||k_z + \frac{b}{||\mathbf{w}||^2}\mathbf{w}||, \end{aligned}$$

where $\frac{1}{R'} = \rho \in \mathbb{R}$ and $\rho > 0$. This constraint is not easy to handle when deriving the optimisation problem. Therefore, it has to be approximated by lower bounding $||k_z + \frac{b}{||\mathbf{w}||^2}\mathbf{w}||$ with a simpler term, and use this term for the formulation of the optimisation problem's constraint. A lower bound can be obtained by the following calculation:

$$\begin{aligned}\rho\|k_z + \frac{b}{\|\mathbf{w}\|^2}\mathbf{w}\| &= \rho\left(\|k_z\|^2 + \frac{b^2}{\|\mathbf{w}\|^2}\right)^{\frac{1}{2}} \\ &\geq \rho\left(\|k_z\|^2\right)^{\frac{1}{2}} = \|k_z\|.\end{aligned}$$

Since $\|k_z\| = \sqrt{k(z,z)}$ is a constant, one can use the same constraint as if $b = 0$:

$$|\langle \mathbf{w}, k_z\rangle + b| \leq \rho\|k_z\|. \tag{2.24}$$

The Cone-Universum $\mathfrak{U}_\mathfrak{c}$-SVM

ρ is a constant controlling the opening angle of the cone, i.e. ρ is inversely proportional to the cone's opening angle and is therefore proportional to the opening angle of the cone's complement. Instead of calculating the value of $\rho = \frac{1}{R'}$, it can as well just be set to a fixed value at the beginning of the optimisation.

The new Universum algorithm can now be obtained by replacing the constraint $|\langle \mathbf{w}, k_{z_j}\rangle + b| \leq \varepsilon - \vartheta_j$ (2.9) by $|\langle \mathbf{w}, k_{z_j}\rangle + b| \leq \rho\|k_z\| - \vartheta_j$. The resulting SVM will be referred to as $\mathfrak{U}_\mathfrak{c}$-SVM or *Cone-Universum*.

2.4 Analysis of the Hinge and Least Squares Universum Implementations

The next sections present an analysis of a few aspects of the Universum algorithm for SVMs. It turns out that the algorithm as proposed by [103] is closely related to the restriction of the space of possible solutions (see 2.4.1) and feature extraction algorithms like *kernel principal component analysis* (kPCA) [80] and *kernel fisher discriminant analysis* (kFDA*) [61, 60]* (see 2.4.2). The results of this section can help to explain the performance of the $\mathfrak{U}$-SVM as well as the $\mathfrak{U}_{\mathfrak{ls}}$-SVM and give a guideline how to choose a Universum for certain tasks at hand.

2.4.1 Hard Margin Universum and Subspace Projection

$\mathfrak{U}$-SVM with $C_\mathfrak{U} = \infty$ and $b = 0$ is equivalent to projecting out the subspace spanned by the Universum points k_z in $\mathcal{H}$ from the span of training examples, in which the solution of the SVM lives.

In this section it will be shown that a $\mathfrak{U}$-SVM with hard margin on $\mathfrak{U}$, $\varepsilon = 0$ and no offset b, i.e. a $\mathfrak{U}$-SVM with $C_\mathfrak{U} = \infty$ and $b = 0$, is equivalent to training a normal SVM with a special kernel where the subspace spanned by the k_{z_i} with $z_i \in \mathfrak{U}$ is removed from the subspace spanned by the k_{x_i} with $x_i \in \{x_j\}_{j=1,\dots,m}$. Throughout this section, "hard margin $\mathfrak{U}$-SVM" refers to a $\mathfrak{U}$-SVM with hard margin on $\mathfrak{U}$. The margin on the labelled examples can be soft, i.e. $C < \infty$. This means that the resulting hyperplane in the RKHS is orthogonal to the space spanned by the Universum points in the RKHS. In order to get an intuitive understanding, the double role of elements $x, x' \in \mathcal{D}$ as either a function on $\mathcal{D}$ or an argument to a function on $\mathcal{D}$ via the kernel k, has to remembered, i.e.

$$\begin{aligned} x: \; \mathcal{D} \to & \; \mathbb{R} \\ x' \mapsto & \; k_x(x') = k(x,x') = \langle k_x, k_{x'} \rangle. \end{aligned}$$

The same is true for linear combinations of elements in the RKHS $\mathcal{H}$.

Now, if all Universum points $z_1, ..., z_q \in \mathfrak{U}$ are enforced to lie directly on the hyperplane given by $\mathbf{w} = \sum_{i=1}^{m+2q} \beta_i k_{\tilde{x}_i}$ ($\tilde{x}$ can denote an element of the training set and an element of the Universum), then $\langle \mathbf{w}, k_{z_j} \rangle = 0$ for all $z_j \in \mathfrak{U}$. This also holds for all linear combinations of the k_{z_j}, because $\langle \mathbf{w}, \sum_{j=1}^{q} \upsilon_j k_{z_j} \rangle = \sum_{j=1}^{q} \upsilon_j \langle \mathbf{w}, k_{z_j} \rangle = 0$. Therefore, $\mathbf{w}$ is orthogonal to any element of span$\{\mathfrak{U}\}$ and, equivalently, to any function represented by linear combinations of elements of $\mathfrak{U}$ in $\mathcal{H}$. If the kernel is seen as a kind of filter, where the filter response $k(x, x')$ indicates how much the feature represented by an element x is contained in another element x', using a hard margin $\mathfrak{U}$-SVM with $\varepsilon = 0$ corresponds to requiring that the resulting function $\mathbf{w}$ should not use the features represented by the elements of $\mathfrak{U}$.

Enforcing the solution $\mathbf{w}$ of an SVM to be orthogonal to span$\{\mathfrak{U}\}$ means making the resulting function invariant against features represented by the Universum.

The roadmap of this section is as follows: First, it is shown that a hard margin $\mathfrak{U}$-SVM with $\varepsilon = 0$ and no offset b minimises the norm of $\mathbf{w}$ in the orthogonal complement span$\{\mathfrak{U}\}^\perp$ of span$\{\mathfrak{U}\}$ in $\mathcal{H}$ by showing that the optimisation problem has a certain form. Then it is shown that a hard margin SVM without offset b and a kernel that is restricted to the complement of span$\{\mathfrak{U}\}$ has exactly this form therefore demonstrating the equivalence of both approaches. Afterwards, the soft margin case with α_i bounded by C or $C_\mathfrak{U}$ and the case where $b \neq 0$ is allowed, are discussed.

The optimisation problem (2.8) for the $\mathfrak{U}$-SVM is given by

$$\begin{aligned} \text{maximise} \quad & \boldsymbol{\varrho}^\top \boldsymbol{\beta} - \frac{1}{2} \boldsymbol{\beta}^\top \mathbf{K} \boldsymbol{\beta} \\ \text{s.t.} \quad & \mathbf{y}^\top \boldsymbol{\beta} = 0 \\ & 0 \leq y_i \beta_i \leq C \text{ for all } 1 \leq i \leq m \\ & 0 \leq y_i \beta_i \leq C_\mathfrak{U} \text{ for all } m < i \leq m + 2q, \end{aligned} \tag{2.25}$$

where all Universum points have been added twice with opposite labels $\tilde{y}_i$ to the optimisation problem, $\beta_i = \alpha_i y_i$, $\varrho_i = \tilde{y}_i$ for $1 \leq i \leq m$ and $\varrho_i = -\varepsilon \tilde{y}_i$ for $m < i \leq m+2q$. The next lemma establishes what has been described intuitively above.

Lemma 2.1: Setting $\varepsilon = 0$, $C_\mathfrak{U} = \infty$ and $b = 0$ in the optimisation problem (2.25) gives rise to an equivalent optimisation problem which has an optimal solution in the orthogonal complement span$\{\mathfrak{U}\}^\perp$ of span$\{\mathfrak{U}\}$, i.e. the optimal solution of this problem has the property $\langle \mathbf{w}^*, k_z \rangle = 0$ for all $k_z \in$ span$\{\mathfrak{U}\}$.

Proof: The formulation (2.25) will first be adapted to the situation $\varepsilon = 0$, $C_\mathfrak{U} = \infty$ and $b = 0$ in order to make it better suited for analysis. Let $\boldsymbol{\beta}^{(l)}$ and $\boldsymbol{\beta}^{(u)}$ denote the $\boldsymbol{\beta}$ part for the training and the Universum points, respectively. Since $\varepsilon = 0$, the linear part $\boldsymbol{\varrho}^\top \boldsymbol{\beta}$ of the objective function becomes $\mathbf{y}^{(l)\top}\boldsymbol{\beta}^{(l)}$, where $\mathbf{y}^{(l)}$ denotes the vector of labels for the training examples. The linear constraint $\mathbf{y}^\top \boldsymbol{\beta}$ only entered the optimisation problem (2.25) to enforce the optimality condition for the offset b (see 2.2.1.1). Hence, it can be removed for the current situation. Since $C_{\mathfrak{U}} = \infty$ the $\boldsymbol{\beta}^{(u)}$ are unbounded and the constraint $0 \le y_i\beta_i \le C_{\mathfrak{U}}$ for all $m \le i \le m + 2q$ can be removed. Noting that $||\mathbf{w}||^2 = \langle \sum_{i=1}^{m+2q} \beta_i k_{\tilde{x}_i}, \sum_{j=1}^{m+2q} \beta_j k_{\tilde{x}_j} \rangle = \boldsymbol{\beta}^\top \mathbf{K}\boldsymbol{\beta}$, the adapted optimisation problem can be written as

Since $\varepsilon = 0$, the linear part of the SVM objective has only Lagrange multipliers from the labelled training examples $\mathfrak{L}$. The linear equality constaint is removed from the optimisation problem because $b = 0$.

$$\begin{aligned} \text{maximise} \quad & \mathbf{y}^{(l)\top}\boldsymbol{\beta}^{(l)} - \frac{1}{2}||\mathbf{w}||^2 \qquad (2.26)\\ \text{s.t.} \quad & 0 \le y_i\beta_i \le C \text{ for all } 1 \le i \le m. \end{aligned}$$

Let now $\mathbf{w}^{(l)} = \sum_{i=1}^{m} \beta_i k_{\tilde{x}_i}$ denote the part of $\mathbf{w}$ that consists of linear combinations of training points k_{x_i} and $\mathbf{w}^{(u)} = \sum_{i=m+1}^{m+2q} \beta_i k_{\tilde{x}_i}$ the part consisting of Universum points. Furthermore, let $\mathbf{w}_{||}^{(l)} \in \operatorname{span}\{k_z | z \in \mathfrak{U}\}$ be the part of $\mathbf{w}^{(l)}$ that can be expressed by linear combinations of elements in $\mathfrak{U}$ and let $\mathbf{w}_{\perp}^{(l)} = \mathbf{w}^{(l)} - \mathbf{w}_{||}^{(l)} \in \operatorname{span}\{k_z | z \in \mathfrak{U}\}^\perp$ its orthogonal complement. Then $\mathbf{w}$ can be decomposed the following way:

$$\begin{aligned} \mathbf{w} &= \mathbf{w}^{(l)} + \mathbf{w}^{(u)}\\ &= \mathbf{w}_{\perp}^{(l)} + \mathbf{w}_{||}^{(l)} + \mathbf{w}^{(u)}. \end{aligned}$$

Substituting that into equation (2.26) yields

Proof in a nutshell: Decompose $\mathbf{w}$ into a parallel and an orthogonal part. The optimal choice for the optimiser is to set the parallel part to zero.

$$\begin{aligned} \mathbf{y}^{(l)\top}\boldsymbol{\beta}^{(l)} - \frac{1}{2}||\mathbf{w}||^2 &= \mathbf{y}^\top\boldsymbol{\beta}^{(l)} - \frac{1}{2}||\mathbf{w}_{\perp}^{(l)} + \mathbf{w}_{||}^{(l)} + \mathbf{w}^{(u)}||^2\\ &\overset{(i)}{=} \mathbf{y}^\top\boldsymbol{\beta}^{(l)} - \frac{1}{2}||\mathbf{w}_{\perp}^{(l)}||^2 - \frac{1}{2}||\mathbf{w}_{||}^{(l)} + \mathbf{w}^{(u)}||^2, \end{aligned}$$

where (i) follows from the fact that $\mathbf{w}_{||}^{(l)}$ is orthogonal to all elements of $\operatorname{span}\{k_z | z \in \mathfrak{U}\}$ and therefore

$$\begin{aligned} ||\mathbf{w}_{\perp}^{(l)} + \mathbf{w}_{||}^{(l)} + \mathbf{w}^{(u)}||^2 &= \langle \mathbf{w}_{\perp}^{(l)} + \mathbf{w}_{||}^{(l)} + \mathbf{w}^{(u)}, \mathbf{w}_{\perp}^{(l)} + \mathbf{w}_{||}^{(l)} + \mathbf{w}^{(u)} \rangle\\ &= \langle \mathbf{w}_{\perp}^{(l)}, \mathbf{w}_{\perp}^{(l)} + \mathbf{w}_{||}^{(l)} + \mathbf{w}^{(u)} \rangle + \langle \mathbf{w}_{||}^{(l)} + \mathbf{w}^{(u)}, \mathbf{w}_{\perp}^{(l)} + \mathbf{w}_{||}^{(l)} + \mathbf{w}^{(u)} \rangle\\ &= \langle \mathbf{w}_{\perp}^{(l)}, \mathbf{w}_{\perp}^{(l)} \rangle + 2\overbrace{\langle \underbrace{\mathbf{w}_{\perp}^{(l)}}_{\in \operatorname{span}\{k_z|z\in\mathfrak{U}\}^\perp}, \underbrace{\mathbf{w}_{||}^{(l)} + \mathbf{w}^{(u)}}_{\in \operatorname{span}\{k_z|z\in\mathfrak{U}\}} \rangle}^{=0}\\ &\quad + \langle \mathbf{w}_{||}^{(l)} + \mathbf{w}^{(u)}, \mathbf{w}_{||}^{(l)} + \mathbf{w}^{(u)} \rangle\\ &= ||\mathbf{w}_{\perp}^{(l)}||^2 + ||\mathbf{w}_{||}^{(l)} + \mathbf{w}^{(u)}||^2. \end{aligned}$$

Since (2.26) needs to be maximised, the optimal choice of $\boldsymbol{\beta}^{(u)}$ and therefore $\mathbf{w}^{(u)}$ is such that $\mathbf{w}^{(u)} = -\mathbf{w}_{||}^{(l)}$ and therefore $||\mathbf{w}_{||}^{(l)} + \mathbf{w}^{(u)}||^2 =$

$||\mathbf{w}_{||}^{(l)} - \mathbf{w}_{||}^{(l)}||^2 = 0$. Since the elements of $\boldsymbol{\beta}^{(u)}$ are unbounded due to $C_{\mathfrak{U}} = \infty$ and since $\mathbf{w}_{||}^{(l)} \in \text{span}\{\mathfrak{U}\}$, this is always possible and therefore the optimisation problem (2.26) can also be written as

$$\begin{aligned} \text{maximise} \quad & \mathbf{y}^{(l)\top}\boldsymbol{\beta}^{(l)} - \frac{1}{2}||\mathbf{w}_{\perp}^{(l)}||^2 \\ \text{s.t.} \quad & 0 \leq y_i\beta_i \leq C \text{ for all } 1 \leq i \leq m. \end{aligned} \tag{2.27}$$

□

The lemma concludes the first part of this section. The next step is to show that the optimisation problem of a standard SVM with an arbitrary kernel restricted to span$\{\mathfrak{U}\}^{\perp}$ has exactly the same form as in lemma 1. For a given kernel k, its restriction to the subspace spanned by the set $\{k_z | z \in \mathfrak{U}\}$ is given by [42]:

$$k^{||}(x, x') = \sum_{r,s=1}^{q} (\mathbf{K}_{(\mathfrak{U},\mathfrak{U})}^{-1})_{rs} k(x, z_r) k(x', z_s).$$

The kernel $k^{\perp}$ living on the complement of the subspace span$\{k_z | z \in \mathfrak{U}\}$ is then simply obtained by subtracting $k^{||}$ from k:

$$\begin{aligned} k^{\perp}(x, x') &= k(x, x') - k^{||}(x, x') \\ &= k(x, x') - \sum_{r,s=1}^{q} (\mathbf{K}_{(\mathfrak{U},\mathfrak{U})}^{-1})_{rs} k(x, z_r) k(x', z_s). \end{aligned}$$

Here, $\mathbf{K}_{(\mathfrak{U},\mathfrak{U})}$ denotes the kernel matrix computed only on the elements of $\mathfrak{U}$. With these tools at disposal, it is possible to prove the second lemma.

LEMMA 2.2: For $b = 0$ and a given kernel k, the optimisation problem of a standard SVM with the kernel $k^{\perp}$ induced by a set $\mathfrak{U}$ has the form of (2.27).

PROOF: Using the notation from above, the optimisation problem for a standard SVM with kernel $k^{\perp}$ is given by

Proof in a nutshell: Use a kernel that has the Universum projected out and show that it has the same form as the problem in lemma 1.

$$\begin{aligned} \text{maximise}_{\boldsymbol{\beta}^{(l)}} \quad & \mathbf{y}^{(l)\top}\boldsymbol{\beta}^{(l)} - \frac{1}{2}\boldsymbol{\beta}^{(l)\top}\mathbf{K}_{(l,l)}^{\perp}\boldsymbol{\beta}^{(l)} \\ \text{s.t.} \quad & \mathbf{y}^{(l)\top}\boldsymbol{\beta} = 0 \\ & 0 \leq y_i\beta_i \leq C \text{ for all } 1 \leq i \leq m, \end{aligned} \tag{2.28}$$

where $\mathbf{K}_{(l,l)}^{\perp}$ denotes the kernel matrix on the training examples for kernel $k^{\perp}$. It is important to note that in this setting the Universum points enter the SVM optimisation problem solely through the kernel $k^{\perp}$. Apart from using a special kernel, (2.28) is just the formulation of a normal SVM.

As before, the equality constraint $\mathbf{y}^{(l)\top}\boldsymbol{\beta} = 0$ can be ignored. Expanding the kernel matrix in the objective function of (2.28) yields

$$\begin{aligned}
& \mathbf{y}^{(l)\top}\boldsymbol{\beta}^{(l)} - \frac{1}{2}\boldsymbol{\beta}^{(l)\top}\mathbf{K}^{\perp}_{(l,l)}\boldsymbol{\beta}^{(l)} \\
= \; & \sum_{i=1}^{m} y_i\beta_i - \frac{1}{2}\left(\sum_{i,j=1}^{m}\beta_i\beta_j\left(k(x_i,x_j) - \sum_{r,s=1}^{q}(\mathbf{K}^{-1}_{(\mathfrak{U},\mathfrak{U})})_{rs}k(x_i,z_r)k(x_j,z_s)\right)\right) \\
= \; & \sum_{i=1}^{m} y_i\beta_i - \frac{1}{2}\sum_{i,j=1}^{m}\beta_i\beta_j k(x_i,x_j) + \frac{1}{2}\sum_{i,j=1}^{m}\beta_i\beta_j\sum_{r,s=1}^{q}(\mathbf{K}^{-1}_{(\mathfrak{U},\mathfrak{U})})_{rs}k(x_i,z_r)k(x_j,z_s) \\
= \; & \sum_{i=1}^{m} y_i\beta_i - \frac{1}{2}||\mathbf{w}^{(l)}||^2 + \sum_{r,s=1}^{q}\left(\sum_{i=1}^{m}\beta_i k(x_i,z_r)\right)\left(\sum_{j=1}^{m}\beta_j k(x_j,z_s)\right)(\mathbf{K}^{-1}_{(\mathfrak{U},\mathfrak{U})})_{rs} \\
= \; & \mathbf{y}^{(l)\top}\boldsymbol{\beta}^{(l)} - \left(\frac{1}{2}||\mathbf{w}^{(l)}||^2 - \frac{1}{2}||P_{\mathfrak{U}}\mathbf{w}^{(l)}||^2\right),
\end{aligned}$$

where $P_{\mathfrak{U}}$ is the projection operator that projects onto the subspace $\operatorname{span}\{k_z|z \in \mathfrak{U}\}$. Using the same notation as in the proof of lemma 1 $\mathbf{w}^{(l)}$ can be decomposed into $\mathbf{w}^{(l)} = \mathbf{w}^{(l)}_{\perp} + \mathbf{w}^{(l)}_{||}$:

$$\begin{aligned}
& \sum_{i=1}^{m} y_i\beta_i - \frac{1}{2}\left(||\mathbf{w}^{(l)}||^2 - ||P_{\mathfrak{U}}w||_2^2\right) \\
= \; & \mathbf{y}^{(l)\top}\boldsymbol{\beta}^{(l)} - \frac{1}{2}\left(||\mathbf{w}^{(l)}_{||} + \mathbf{w}^{(l)}_{\perp}||^2 - ||P_{\mathfrak{U}}\mathbf{w}^{(l)}||^2\right) \\
\overset{(i)}{=} \; & \mathbf{y}^{(l)\top}\boldsymbol{\beta}^{(l)} - \frac{1}{2}\left(||\mathbf{w}^{(l)}_{\perp}||^2 + \underbrace{||\mathbf{w}^{(l)}_{||}||^2 - ||P_{\mathfrak{U}}\mathbf{w}^{(l)}||^2}_{=0,\text{ since } P_{\mathfrak{U}}\mathbf{w}=\mathbf{w}^{(l)}_{||}}\right) \\
= \; & \mathbf{y}^{(l)\top}\boldsymbol{\beta}^{(l)} - \frac{1}{2}||\mathbf{w}^{(l)}_{\perp}||^2,
\end{aligned}$$

where (i) follows from the orthogonality of $\mathbf{w}^{(l)}_{\perp}$ and $\mathbf{w}^{(l)}_{||}$, i.e. $\langle\mathbf{w}^{(l)}_{\perp}, \mathbf{w}^{(l)}_{||}\rangle = 0$. Since the projection of $\mathbf{w}^{(l)}$ onto $\operatorname{span}\{k_z|z \in \mathfrak{U}\}$ is exactly $P_{\mathfrak{U}}\mathbf{w}^{(l)} = \mathbf{w}^{(l)}_{||}$, both norms $||\mathbf{w}^{(l)}_{||}||^2$ and $||P_{\mathfrak{U}}\mathbf{w}^{(l)}||^2$ cancel, and the optimisation problem has the form of (2.27), as claimed in the lemma.

□

Lemmata 2.1 and 2.2 establish that a hard margin $\mathfrak{U}$-SVM without offset b and the width of the ε-tube around the hyperplane for the ε-insensitive loss set to zero only optimises w on the intersection of the spaces $\operatorname{span}\{k_z|z \in \mathfrak{U}\}^{\perp}$ and $\operatorname{span}\{k_x|x \in \mathfrak{L}\}$. When relaxing the hard margin constraint such that the Universum points are not forced to lie exactly on the hyperplane, the above calculations remain valid apart from one main difference which will be discussed next.

The only change to the above setting is $C_{\mathfrak{U}} < \infty$. This means that the slack variables ϑ_j in equation (2.9) are allowed to have positive values, i.e. deviations from the hyperplane are possible for Universum points. The penalisation of deviations from the margin ϑ_j by a $C_{\mathfrak{U}} < \infty$ transforms into upper bounds on the corresponding Lagrangian multipliers $0 \leq y_j\beta_j = \alpha_j \leq C_{\mathfrak{U}}$ in the dual problem. A comparison of the objective functions of lemma 2.1 and lemma 2.2 shows the effect of a bounded Lagrange multiplier on the projections:

Soft margin on $\mathfrak{U}$ corresponds to "softly projecting out the Universum elements"

$$\begin{array}{ll}\text{Lemma 2.1 :} & \mathbf{y}^{(l)\top}\boldsymbol{\beta}^{(l)} - \frac{1}{2}||\mathbf{w}_{\perp}^{(l)}||^2 - \frac{1}{2}||\mathbf{w}_{||}^{(l)} + \mathbf{w}^{(u)}||^2 \\ \text{Lemma 2.2 :} & \mathbf{y}^{(l)\top}\boldsymbol{\beta}^{(l)} - \frac{1}{2}\left(||\mathbf{w}_{||}^{(l)}||^2 + ||\mathbf{w}_{\perp}^{(l)}||^2 - ||P_{\mathfrak{U}}\mathbf{w}^{(l)}||^2\right)\end{array}$$

As already mentioned in the proof of lemma 2.1, for unbounded $\boldsymbol{\beta}^{(u)}$ the optimal choice for $\boldsymbol{\beta}^{(u)}$ and therefore for $\mathbf{w}^{(u)}$ is $-\mathbf{w}^{(u)} = \mathbf{w}_{||}^{(l)} = P_{\mathfrak{U}}\mathbf{w}^{(l)}$ in order to eliminate the second squared norm. A comparison of the coefficients of $P_{\mathfrak{U}}\mathbf{w}^{(l)}$ with the coefficients in the expansion of $-\mathbf{w}^{(u)}$ reveals the effect of a bounded Lagrange multiplier in terms of the projection:

$$\begin{aligned} -\mathbf{w}^{(u)} &= P_{\mathfrak{U}}\mathbf{w}^{(l)} \\ \sum_{j=1}^{q}(\beta_j^{(u)} + \beta_{j+q}^{(u)})k_{z_j} &= \sum_{j=1}^{q}\underbrace{-1 \cdot \sum_{r=1}^{q}(\mathbf{K}_{(\mathfrak{U},\mathfrak{U})}^{-1})_{rj}\langle \mathbf{w}^{(l)}, k_{z_r}\rangle}_{(\beta_j^{(u)}+\beta_{j+q}^{(u)})}\, k_{z_j}. \end{aligned}$$

The equations above need a little explanation: Since every k_{z_j} appears twice in the sum for $\mathbf{w}^{(u)}$ in (2.9), the two coefficients corresponding to the same element k_{z_j} can merged into a single coefficient $\beta_j^{(u)} + \beta_{j+q}^{(u)}$ yielding $\mathbf{w}^{(u)} = \sum_{j=1}^{q}(\beta_j^{(u)} + \beta_{j+q}^{(u)})k_{z_j}$. When summing only once over $\mathfrak{U}$, two coefficients β_j and β_{j+q} can be combined per element k_{z_j}. The coefficients $(\beta_j^{(u)} + \beta_{j+q}^{(u)})$ for k_{z_j} are then obtained by expanding the expression for $P_{\mathfrak{U}}\mathbf{w}^{(l)}$ and comparing the coefficients. It follows that $\beta_j^{(u)} + \beta_{j+q}^{(u)} = -\sum_{r=1}^{q}(\mathbf{K}_{(\mathfrak{U},\mathfrak{U})}^{-1})_{rj}\langle \mathbf{w}^{(l)}, k_{z_r}\rangle$.

Since the absolute values $|\boldsymbol{\beta}^{(u)}|$ are bounded by $C_{\mathfrak{U}}$, it might not be possible to express the value of $-\sum_{r=1}^{q}(\mathbf{K}_{(\mathfrak{U},\mathfrak{U})}^{-1})_{rj}\langle \mathbf{w}^{(l)}, k_{z_r}\rangle$ by $\beta_j^{(u)} + \beta_{j+q}^{(u)}$, because it might simply be too large. In this casem the optimal value $P_{\mathfrak{U}}\mathbf{w}^{(l)}$ for $\mathbf{w}^{(u)}$ cannot be reached. Thus, the result could be called *soft projection:* To capture as much as possible from $P_{\mathfrak{U}}\mathbf{w}^{(l)}$ by $-\mathbf{w}^{(u)}$ in order to minimise the norm $||\mathbf{w}_{||}^{(l)} + \mathbf{w}^{(u)}||^2 = ||P_{\mathfrak{U}}\mathbf{w}^{(l)} + \mathbf{w}^{(u)}||^2$ as small as possible. Shrinking it to zero is only possible if $P_{\mathfrak{U}}\mathbf{w}^{(l)}$ falls into the set that is expressible as linear combinations of the k_{z_j} with bounded coefficients. This situation is sketchily depicted in figure 2.5.

The case $b \neq 0$ makes a geometrical interpretation difficult.

So far b was assumed to be zero, i.e. no offset b was used. A non-zero b complicates the situation, because the linear equality constraint

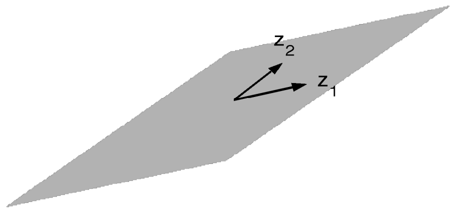


Figure 2.5: The set of all possible linear combinations of $\mathbf{z}_1$ and $\mathbf{z}_2$ for bounded coefficients $0 \leq |\beta_1| \leq 1.2$ and $0 \leq |\beta_2| \leq 1.9$.

$\mathbf{y}^\top \boldsymbol{\beta} = 0$ enters the optimisation problem (2.26). Since the Universum points have associated labels in (2.26), too, the equality constraint poses a balancing condition on all coefficients $\boldsymbol{\beta}$. If the coefficients for the Universum points have large positive values, the coefficients for the training points must have a fair amount of negative values and vice versa in order to make all coefficients sum to zero. Unfortunately, this makes a geometric interpretation difficult. One intuition, is that optimising (2.9) with an offset b will adjust b in a way such that there is an optimal trade-off between centring the hyperplane on the Universum points and placing it between the two labelled classes.

A Universum specifies noise or invariance directions respectively.

How can the relation between the $\mathfrak{U}$-SVM and the kernel on the orthogonal complement of span$\{k_z|z \in \mathfrak{U}\}$ help to characterise helpful Universum sets? On the one hand, when seeing $k_z = k(z, \cdot)$ as a filter that responds to a specific pattern defined by z, enforcing the hyperplane normal $\mathbf{w}$ to be orthogonal to these filters means making it invariant against features inside span$\{k_z|z \in \mathfrak{U}\}$. When $\mathbf{w}$ is used to predict the label of a test example by $\hat{y} = \text{sgn}(\langle \mathbf{w}, k_x \rangle)$, the part of k_x that lies inside span$\{k_z|z \in \mathfrak{U}\}$ is not used for prediction since it does not contribute to $\langle \mathbf{w}, k_x \rangle$. A good choice of the Universum could therefore be a set of data points that represent the desired directions of invariance. Those could be original examples changed by certain transformations the solution $\mathbf{w}$ should be invariant against, like rotations or translations. Another possible Universum set would be samples from the noise distribution, if the loss function does not reflect the real noise distribution for some reasons. There might be cases where such noise samples can be obtained empirically without knowing the actual form of the noise distribution. Another link between the Universum algorithm and the incorporation of invariance against undesired noise directions is explored in the following section.

2.4.2 Least Squares Universum, Kernel Oriented Principal Component Analysis and Kernel Fisher Discriminant Analysis

$\mathfrak{U}_{\mathfrak{ls}}$-SVM is equivalent to a hybrid of kFDA and a special form of kPCA.

The following considerations relate the least squares $\mathfrak{U}_{\mathfrak{ls}}$-SVM (see 2.2.1.2) to the kernelised versions of two well known learning algorithms: *kernel Principal Component Analysis (kPCA)* and *kernel Fisher Discriminant Analysis (kFDA)* [85]. After a brief introduction of both algorithms in the linear domain along with the corresponding optimisation problemsthey are extended to the kernelised versions. Finally, the equivalence to the $\mathfrak{U}_{\mathfrak{ls}}$-SVM will be shown.

Principle component analysis finds a lower-dimensional basis that minimises the reconstruction error.

Principal Component Analysis (PCA) [23] is an unsupervised feature extraction algorithm that has already been used in statistics before many other learning algorithms were invented. The underlying idea is very simple: Given a set $\mathfrak{X} = \{\mathbf{x}_i\}_{i=1,\ldots,m} \in \mathbb{R}^{d_1}$ of vectorial data, the objective is to find an orthonormal basis $\mathcal{B} = \{\mathbf{b}_k\}_{k=1,\ldots,d_2}$ with $d_2 < d_1$ of a subspace of $\mathbb{R}^{d_1}$ such that the *reconstruction error*

$$\ell_{re}(\mathbf{x}, P_{\mathcal{B}}\mathbf{x}) \quad = \quad ||\mathbf{x} - \sum_{i=1}^{d_2} \mathbf{b}_i (P_{\mathcal{B}}\mathbf{x})_i||^2$$

between x and the projection $P_{\mathcal{B}}\mathbf{x}$ of x onto the Basis $\mathcal{B}$ is minimised [21]. The elements of $\mathcal{B}$ are called *principal components*. The motivation for finding a lower dimensional representation $P_{\mathcal{B}}\mathbf{x}$ of x can be denoising the data, by assuming isotropic Gaussian noise in the part of the space that gets projected out. Another motivation is reducing the dimensionality of the data for reasons of algorithmic efficiency.

Principle component analysis is a factor analyser with orthogonal factors and an isotropic Gaussian noise model.

As can be seen from the equation above, the loss function of PCA implements a Gaussian noise model with independent and equal noise in each direction. In fact, PCA is a special case of a more general class of algorithms, called *factor analysis*, [77] that allows for different amounts of noise in each direction of the space.

The principle components point into the directions of maximal variance.

Setting the first derivative of the loss function to zero shows that the desired basis $\mathcal{B}$ is given by the eigenvectors $\mathbf{b}_1, ..., \mathbf{b}_{d_2}$ corresponding to the d_2 largest eigenvalues of the covariance matrix $\mathrm{S} = \frac{1}{|\mathfrak{X}|}\sum_{\mathbf{x}\in\mathfrak{X}} \mathbf{x}\mathbf{x}^\top$. Here, w.l.o.g. $\mathbf{x} \in \mathbb{R}^{d_1}$ is assumed to have zero mean. Since S is symmetric, the eigenvectors are already orthogonal. Geometrically, the first eigenvector $\mathbf{b}_1$ points in the direction of maximal variance of $\mathfrak{X}$, the second eigenvector in the direction of maximal variance that is orthogonal to $\mathbf{b}_1$, and so on. The lower dimensional representation $P_{\mathcal{B}}\mathbf{x}$ of a vector $\mathbf{x}$ is then simply its projection onto the basis $\mathcal{B}$, i.e. $P_{\mathcal{B}}\mathbf{x} = (\langle \mathbf{b}_1, \mathbf{x}\rangle, ..., \langle \mathbf{b}_{d_2}, \mathbf{x}\rangle)^\top$.

One way to obtain the eigenvectors of a symmetric matrix $\mathbf{S}$ is to maximise the *Rayleigh Quotient* [85]

$$\text{maximise}_{\mathbf{w}} \quad \frac{\mathbf{w}^\top \mathbf{S}\mathbf{w}}{\langle \mathbf{w}, \mathbf{w}\rangle}. \tag{2.29}$$

The solution of this optimisation problem yields the eigenvector corresponding to the largest eigenvalue of $\mathbf{S}$. If the resulting vector $\mathbf{w}$ is not of length one, it can simply be normalised since the normalisation factor will appear in the numerator and the denominator of the Rayleigh quotient and hence cancel. For computing another eigenvector, $\mathbf{S}$ is *deflated*, that is $\mathbf{w}$ is projected out by $\mathbf{S}_{/\mathbf{w}} := \mathbf{S} - \mathbf{w}\mathbf{w}^\top$, and the optimisation problem is solved again on $\mathbf{S}_{/\mathbf{w}}$.

Stating the optimisation problem for PCA in terms of the Rayleigh Quotient offers the possibility for an interesting variation of PCA, called *oriented Principal Component Analysis (oPCA)*. Since any symmetric, positive definite matrix $\mathbf{G}$ with full rank gives rise to a dot product $\langle \mathbf{w}, \mathbf{w}'\rangle_{\mathbf{G}} = \mathbf{w}^\top \mathbf{G}\mathbf{w}'$ [27], the dot product in the denominator of (2.29) can be replaced by any other bilinear symmetric form $\mathbf{w}^\top \mathbf{G}\mathbf{w}$. This yields the optimisation problem for oPCA:

Oriented Principle Component Analysis: PCA with an extra noise covariance $\mathbf{G}$.

$$\text{maximise}_{\mathbf{w}} \quad \frac{\mathbf{w}^\top \mathbf{S}\mathbf{w}}{\mathbf{w}^\top \mathbf{G}\mathbf{w}}. \tag{2.30}$$

The solution of oPCA can be obtained by solving a generalised eigenvalue problem $\mathbf{S}\mathbf{b} = \lambda\mathbf{G}\mathbf{b}$, which can in turn be transformed into a normal eigenvalue problem. The effect of $\mathbf{G}$ on the solution $\mathbf{b}_1, ..., \mathbf{b}_{d_2}$ of (2.30) is that the solution must implement a trade-off between maximising $\mathbf{w}^\top \mathbf{S}\mathbf{w}$ and minimising $\mathbf{w}^\top \mathbf{G}\mathbf{w}$. If $\mathbf{G}$ is interpreted as a covariance matrix, the minimising directions are the ones with minimal variance of the data that gave rise to $\mathbf{G}$. Therefore, $\mathbf{G}$ can be seen as the covariance matrix of the noise on the data $\mathcal{X}$, encoding directions the desired solution of (2.30) should be invariant against. Since the resulting principal components of (2.30) are as orthogonal as possible to the eigenvectors of $\mathbf{G}$, the resulting basis $\mathbf{b}_1, ..., \mathbf{b}_{d_2}$ is more robust against the directions of maximal variance specified by $\mathbf{G}$. This is the motivation for oPCA.

Fisher Discriminant Analysis (FDA) [28] is an early supervised classification algorithm that has an objective function which is very similar to the one of oPCA. The underlying idea of FDA is again very simple: Find the *most discriminating* direction $\mathbf{w}$ by maximising the variance between the projections of both classes onto $\mathbf{w}$ (*inter-class variance*) and minimising the variance of the projections within each single class (*intra-class variance*) at the same time. In contrast to the unsupervised

Fisher discriminant analysis finds a linear subspace that maximises the inter-class variance and minimises the inner-class variance.

algorithms PCA and oPCA, FDA requires labelled data. The algorithm is now stated for the binary case, i.e. $\mathcal{R} = \{-1, +1\}$.

Given a set of labelled data points $\mathfrak{L} = \{(y_i, \mathbf{x}_i)\}_{i=1,...,m}$ let $\mathbf{c}^+ := \frac{1}{|\mathcal{I}^+|}\sum_{i\in\mathcal{I}^+}\mathbf{x}_i$ be the mean of all positively labelled examples, where $\mathcal{I}^+$ and $\mathcal{I}^-$ are the index sets for all positive and all negative examples, respectively. Let $\mathbf{c}^-$ be defined analogously. In order to write the optimisation problem in the style of equation (2.30), two matrices $\mathbf{S}_B := (\mathbf{c}^+ - \mathbf{c}^-)(\mathbf{c}^+ - \mathbf{c}^-)^\top$ and $\mathbf{S}_W := \sum_{j\in\mathcal{I}^+}(\mathbf{x}_j - \mathbf{c}^+)(\mathbf{x}_j - \mathbf{c}^+)^\top + \sum_{j\in\mathcal{I}^-}(\mathbf{x}_j - \mathbf{c}^-)(\mathbf{x}_j - \mathbf{c}^-)^\top$ are defined, playing the role of the inter-class variance and the intra-class variance, respectively. The corresponding optimisation problem can then be stated as

$$\text{maximise}_{\mathbf{w}\in\mathbb{R}^{d_1}} \qquad \frac{\mathbf{w}^\top\mathbf{S}_B\mathbf{w}}{\mathbf{w}^\top\mathbf{S}_W\mathbf{w}}. \tag{2.31}$$

As for oPCA, the solution can be obtained by solving the generalised eigenvalue problem $\mathbf{S}_B\mathbf{w} = \lambda\mathbf{S}_W\mathbf{w}$.

PCA, oPCA and FDA can be computed in an RKHS via the kernel trick.

In the following, the kernelised versions of PCA, oPCA and FDA are stated. PCA was generalised to reproducing kernel Hilbert spaces by [80]. The principal components are again linear combinations of the k_{x_i}. Therefore, each principal component $\mathbf{b}_k \in \mathcal{H}$ in the RKHS is represented by a set of coefficients $\boldsymbol{\alpha}^j$ for the different elements k_{x_i}. These sets of coefficients $\{\boldsymbol{\alpha}^j\}_{j=1,...,d_2}$ are given by the eigenvectors of the kernel matrix $\mathbf{K}$ of $\mathfrak{X}$. In order to normalise $\mathbf{b}_k \in \mathcal{H}$ to length one, the coefficients $\boldsymbol{\alpha}$ have to be divided by the square root of the eigenvalue corresponding to $\mathbf{b}_k$. The computation of the projection $Pk_x =: \hat{\mathbf{x}} = (\hat{x}_1, ..., \hat{x}_{d_2})^\top$ of a new point x is the dot product between k_x and the principal components $\mathbf{b}_k$ in $\mathcal{H}$. By virtue of the construction of the dot product in $\mathcal{H}$, the computation boils down to a sum of kernel functions at x and x_i:

$$\begin{aligned}
\hat{x}_i &= \langle \mathbf{b}_i, k_x\rangle \\
&= \left\langle \sum_{j=1}^m \alpha_j^i k_{x_j}, k_x \right\rangle \\
&= \sum_{j=1}^m \alpha_j^i \langle k_{x_j}, k_x\rangle \\
&= \sum_{j=1}^m \alpha_j^i k(x_j, x).
\end{aligned}$$

As pointed out by [95], the kPCA problem can be solved by writing it in a least squares SVM like fashion. The trick is to introduce slack variables ξ_i that hold the value of the projection $\langle \mathbf{w}, k_{x_i}\rangle$ of x_i onto $\mathbf{w}$, where the weight vector $\mathbf{w}$ of the SVM takes over the role of the principal component. Since only the direction of $\mathbf{w}$ and not its length changes the value of the Rayleigh quotient, the length of $\mathbf{w}$ is fixed to one. This is enforced by a constraint $\langle \mathbf{w}, \mathbf{w}\rangle = 1$. When formulating this as an optimisation problem with Lagrange multipliers, it turns out that the resulting optimisation problem is the same as solving a least squares SVM with target values $y_i = 0$ for all $x_i \in \mathfrak{X}$. A similar trick will be used later when relating oriented kernel PCA and kernel FDA to the $\mathfrak{U}_{\mathfrak{ls}}$-SVM.

The variation of kernel oPCA considered in this thesis uses the Universum points to build the noise covariance matrix $\mathbf{G}$ of equation (2.30). This type of *kernel oPCA (koPCA)* is obtained by writing $\mathbf{w}$ as a linear combination of training and Universum points $\mathbf{w} = \sum_{i=1}^{m} \lambda_i k_{x_i} + \sum_{j=1}^{q} \upsilon_j k_{z_j}$ as in the case of SVMs, and using the simple matrix algebra fact that $\mathbf{w}^\top \left(\sum_{i=1}^{m+q} \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^\top\right) \mathbf{w} = \sum_{i=1}^{m+q} \langle \mathbf{w}, \tilde{\mathbf{x}}_i\rangle \cdot \langle \mathbf{w}, \tilde{\mathbf{x}}_i\rangle$:

Kernel oPCA with a noise covariance matrix obtained from the Universum points $\mathfrak{U}$

$$\begin{aligned} &\text{maximise}_{\mathbf{w}} && \frac{\sum_{i=1}^{m} \langle \mathbf{w}, k_{x_i}\rangle \cdot \langle \mathbf{w}, k_{x_i}\rangle}{\sum_{j=1}^{q} \langle \mathbf{w}, k_{z_j}\rangle \cdot \langle \mathbf{w}, k_{z_j}\rangle} \\ \Leftrightarrow\ &\text{maximise}_{\boldsymbol{\lambda},\boldsymbol{\upsilon}} && \frac{\sum_{i=1}^{m} \langle \sum_{k=1}^{m} \lambda_k k_{x_k} + \sum_{l=1}^{q} \upsilon_l k_{z_l}, k_{x_i}\rangle^2}{\sum_{j=1}^{q} \langle \sum_{k=1}^{m} \lambda_k k_{x_k} + \sum_{l=1}^{q} \upsilon_l k_{z_l}, k_{z_j}\rangle^2}. \end{aligned}$$

Using the linearity of the dot product and writing $\boldsymbol{\alpha} = \binom{\boldsymbol{\lambda}}{\boldsymbol{\upsilon}}$ the optimisation problem boils down to

Kernel oPCA can also be written as a Rayeigh Quotient and solved by a generalised eigenvalue problem.

$$\text{maximise}_{\boldsymbol{\alpha}} \quad \frac{\boldsymbol{\alpha}^\top \mathbf{K}_{(\mathfrak{V},\mathfrak{X})} \cdot \mathbf{K}_{(\mathfrak{X},\mathfrak{V})}\boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top \mathbf{K}_{(\mathfrak{V},\mathfrak{U})} \cdot \mathbf{K}_{(\mathfrak{U},\mathfrak{V})}\boldsymbol{\alpha}}, \tag{2.32}$$

where $\mathfrak{V} := \mathfrak{X} \cup \mathfrak{U}$ is the union of the training and Universum examples and $\mathbf{K}_{(\mathfrak{X},\mathfrak{V})}$ and $\mathbf{K}_{(\mathfrak{U},\mathfrak{V})}$ denote the kernel matrices between the training examples and $\mathfrak{V}$ and the Universum examples and $\mathfrak{V}$, respectively. As usual, the solution of (2.32) can be obtained by solving a generalised eigenvalue problem.

Kernel Fisher Discriminant Analysis (kFDA) [60] can be obtained by the same strategy that has been used for koPCA. To avoid drowning in huge formulae, a little more notation is introduced before stating the optimisation problem. Let $\mathbf{1}_+$ be a vector with entries

$$\mathbf{1}_i^+ \;=\; \begin{cases} 1 & \text{for } i \in \mathcal{I}^+ \\ 0 & \text{else} \end{cases}$$

and let $\mathbf{1}^-$ be defined analogously. For $\mathbf{K} := \mathbf{K}_{(\mathfrak{X},\mathfrak{X})}$ let $\boldsymbol{\mu}^+ := \frac{1}{|\mathcal{I}^+|}\mathbf{K}\mathbf{1}^+$ and $\boldsymbol{\mu}^- := \frac{1}{|\mathcal{I}^-|}\mathbf{K}\mathbf{1}^-$ denote projections of the training examples onto the class means in $\mathcal{H}$, i.e. μ_i^+ equals the projection of x_i onto the mean of the positive examples in $\mathcal{H}$ (see [81, 85] for a more detailed treatment of projections in $\mathcal{H}$). Furthermore, let $\boldsymbol{\mu} := \boldsymbol{\mu}^+ - \boldsymbol{\mu}^-$, $\mathbf{M} := \boldsymbol{\mu}\boldsymbol{\mu}^\top$ and $\mathbf{N} := \boldsymbol{\mu}\boldsymbol{\mu}^\top - |\mathcal{I}^+|\boldsymbol{\mu}^+ - |\mathcal{I}^-|\boldsymbol{\mu}^-$. Then the kFDA optimisation problem can be stated as [60]:

Kernel FDA can also be written as a Rayeigh Quotient and solved by a generalised eigenvalue problem.

$$\begin{aligned} &\text{maximise}_{\boldsymbol{\alpha}} && \frac{(\boldsymbol{\alpha}^\top \boldsymbol{\mu})^2}{\boldsymbol{\alpha}^\top \mathbf{N}\boldsymbol{\alpha}} \\ \Leftrightarrow\ &\text{maximise}_{\boldsymbol{\alpha}} && \frac{\boldsymbol{\alpha}^\top \mathbf{M}\boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top \mathbf{N}\boldsymbol{\alpha}}. \end{aligned} \tag{2.33}$$

As usual, this corresponds to a generalised eigenvalue problem. If $\mathbf{N}$ happens to not have full rank, the problem ill-posed. This can be tackled by regularising $\mathbf{N}$, i.e. by replacing it by $\tilde{\mathbf{N}} = \mathbf{N} + \gamma\mathbf{I}$ with $\gamma\mathbf{I}$ being a multiple of the identity matrix.

Interestingly, the kFDA problem can also be casted in an SVM like optimisation problem [62], making it suitable for a bigger amount of training examples. To transform (2.33) into a quadratic program, i.e. the optimisation of a quadratic objective function with linear constraints, two key observations are crucial. Firstly, the matrix $\mathrm{M} = \boldsymbol{\mu}^\top\boldsymbol{\mu}$ has only rank one since it is an outer product of a vector with itself. Secondly, as in the Rayleigh-Quotient formulation of PCA, scaling $\boldsymbol{\alpha}$ does not change the values of the objective function (2.33). Therefore, $\boldsymbol{\alpha}^\top\boldsymbol{\mu}$ can be fixed to an arbitrary non-zero value. The authors of [62] use $\boldsymbol{\alpha}^\top\boldsymbol{\mu} = 2$. When fixing $\boldsymbol{\alpha}^\top\boldsymbol{\mu} = 2$, maximising (2.33) amounts to minimising $\boldsymbol{\alpha}^\top\mathbf{N}\boldsymbol{\alpha}$. Introducing an additional regulariser $\Theta(\boldsymbol{\alpha})$ to cover cases in which $\mathbf{N}$ does not have full rank, the quadratic program for kFDA is given by

Just as PCA and kPCA, kFDA can be written as an SVM style problem

Transforming kPCA into a quadratic problem

$$\begin{aligned} \text{minimise}_{\boldsymbol{\alpha}} \quad & \boldsymbol{\alpha}\mathbf{N}^\top\boldsymbol{\alpha} + C\Theta(\boldsymbol{\alpha}) \\ \text{s.t.} \quad & \boldsymbol{\alpha}^\top\boldsymbol{\mu} = 2. \end{aligned}$$

The authors of [61] go even further and rephrase the optimisation problem in order to get rid of the matrix $\mathbf{N}$. This can be achieved via another observation about the Fisher Discriminant Analysis that has already been stated above. FDA tries to minimise the variance of the projections onto w within each class while maximising the distance between the average outputs for each class. This leads to the final optimisation problem stated in [61]:

$$\begin{aligned} \text{minimise}_{\boldsymbol{\alpha},b,\boldsymbol{\xi}} \quad & ||\boldsymbol{\xi}||^2 + C\Theta(\boldsymbol{\alpha}) \\ \text{s.t.} \quad & \mathbf{K}\boldsymbol{\alpha} + \mathbf{1}\cdot b = \mathbf{y} + \boldsymbol{\xi} \\ & \boldsymbol{\xi}^\top\mathbf{1}^s = 0 \text{ for } s = \pm. \end{aligned} \tag{2.34}$$

Both optimisation problems are equivalent. This can be seen by noting that $\mathbf{K}\boldsymbol{\alpha} + \mathbf{1}\cdot b$ are the projections of the training examples on the most discriminating direction represented by the coefficients $\boldsymbol{\alpha}$. Since $\boldsymbol{\xi}$ holds the deviations of the projections from the fixed means $\boldsymbol{y}$ for each class, the minisation of $||\boldsymbol{\xi}||^2$ minimises the inner-class variance. The regulariser $\Theta(\boldsymbol{\alpha})$ ensures that the class means are pushed as far apart as possible in a maximal margin like manner. This maximises the inter-class variance. The constraints $\boldsymbol{\xi}^\top\mathbf{1}^s = 0$ for $s = \pm$ ensure that the class means are exactly 1 and -1. The proof for this equivalence is omitted in [61].

After introducing all the ingredients, the relation to the $\mathfrak{U}_{ls}$-SVM may finally be discussed. As mentioned above, the Universum examples will take over the role of samples from the noise distribution and their covariance matrix will serve as the matrix $\mathbf{G}$ in the oPCA algorithm (2.30).

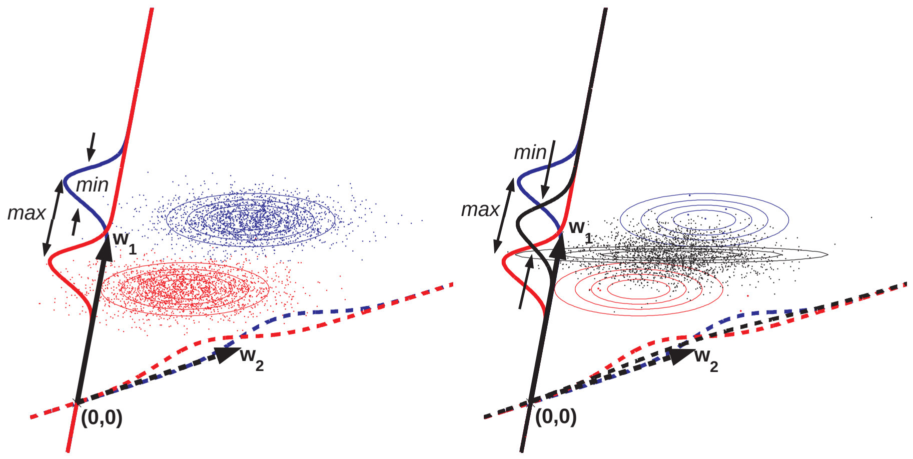


Figure 2.6: Visualisation of the objectives of Fisher Discriminant Analysis (*left*) and the hybrid of Fisher Discriminant Analysis and Oriented Principal Component Analysis (*right*). FDA tries to minimise the variance of the projections onto $\mathbf{w}$ (blue and red Gaussians) while pushing them as far apart as possible. The hybrid does exactly the same with the only difference that it additionally tries to minimise the projections of another dataset (black Gaussian).

The relation between oPCA and $\mathfrak{U}_{ls}$-SVM becomes obvious through the observation that in oPCA the resulting vector $\mathbf{w}$ should be as orthogonal as possible to the eigenvectors of $\mathbf{G}$, i.e. $\mathbf{w}^\top \mathbf{G}\mathbf{w}$ should be as small as possible, while in $\mathfrak{U}_{ls}$-SVM $\mathbf{w}$ should be as orthogonal as possible to the Universum examples in $\mathfrak{U}$. This cannot be the whole story since oPCA is an unsupervised algorithm, while $\mathfrak{U}_{ls}$-SVM is supervised like FDA and kFDA. In fact, $\mathfrak{U}_{ls}$-SVM is equivalent to a hybrid between both algorithms, where the main algorithm consists of kFDA with an additional koPCA term in the denominator. This is shown in the following paragraphs.

Rayleigh Quotient hybrid of kFDA and koPCA.

Defining $\tilde{\mathbf{c}} := \frac{1}{2}(\mathbf{c}^+ + \mathbf{c}^-)$, the primal Rayleigh Quotient optimisation problem is given by:

$$\text{maximise}_{\mathbf{w}} \quad \frac{\mathbf{w}^\top \overbrace{(\mathbf{c}^+ - \mathbf{c}^-)(\mathbf{c}^+ - \mathbf{c}^-)^\top}^{\text{from FDA}} \mathbf{w}}{\mathbf{w}^\top (\underbrace{C \sum_{k=\pm} \sum_{i \in \mathcal{I}^k} (\mathbf{x_i} - \mathbf{c}^k)(\mathbf{x_i} - \mathbf{c}^k)^\top}_{\text{from FDA}} + \underbrace{C_{\mathfrak{U}} \sum_{j=1}^{q} (\mathbf{z}_j - \tilde{\mathbf{c}})(\mathbf{z}_j - \tilde{\mathbf{c}})^\top}_{\text{from oPCA}})\mathbf{w}} \tag{2.35}$$

The zero mean assumption in PCA and oPCA has been replaced by centring the Universum points to $\tilde{\mathbf{c}}$, i.e. placing them exactly in the middle between the two class means $\mathbf{c}^+$ and $\mathbf{c}^-$.

By the same arguments as for kFDA, (2.35) can also be transformed into a quadratic program

$$\begin{array}{ll} \text{minimise}_{\mathbf{w}} & \Theta(\mathbf{w}) \\ & +\mathbf{w}^\top(C\sum_{k=\pm}\sum_{i\in\mathcal{I}^k}(\mathbf{x_i}-\mathbf{c}^k)(\mathbf{x_i}-\mathbf{c}^k)^\top + C_{\mathfrak{U}}\sum_{j=1}^q(\mathbf{z}_j-\tilde{\mathbf{c}})(\mathbf{z}_j-\tilde{\mathbf{c}})^\top)\mathbf{w} \\ \text{s.t.} & \mathbf{w}^\top(\mathbf{c}^+-\mathbf{c}^-)=2 \end{array} \tag{2.36}$$

and rephrased into:

Quadratic program for kFDA-oPCA hybrid

$$\begin{array}{ll} \text{minimise}_{\mathbf{w},b} & \Theta(\mathbf{w}) + C||\boldsymbol{\xi}||_2^2 + C_{\mathfrak{U}}||\boldsymbol{\vartheta}||_2^2 \\ \text{s.t.} & \langle\mathbf{w},\mathbf{x}_i\rangle + b = y_i + \xi_i \text{ for all } 1\le i\le m \\ & \langle\mathbf{w},\mathbf{z}_j\rangle + b = \vartheta_j \text{ for all } 1\le j\le q \\ & \boldsymbol{\xi}^\top\mathbf{1}^k = 0 \text{ for } k=\pm. \end{array} \tag{2.37}$$

Choosing $\Theta(\mathbf{w}) = ||\mathbf{w}||^2$ and ignoring the constraint $\boldsymbol{\xi}^\top\mathbf{1}^k = 0$, equation (2.37) has exactly the same form as the primal optimisation problem for the $\mathfrak{U}_{ls}$-SVM as stated in equation (2.18). Therefore, a $\mathfrak{U}_{ls}$-SVM is equivalent to a hybrid of kFDA and koPCA up to a linear equality constraint.

For transforming (2.36) and (2.37) into algorithms in an RKHS $\mathcal{H}$, a little more notation will be helpful. Let $\mathbf{1}^{\mathfrak{U}}$ be a vector in the style of $\mathbf{1}^+$ indicating if a point is a Universum point and let $\boldsymbol{\mu}^{\mathfrak{U}} := \frac{1}{q}\mathbf{K}_{(\mathfrak{V},\mathfrak{V})}\mathbf{1}^{\mathfrak{U}}$. Furthermore, let $\mathbf{U} := \mathbf{K}_{(\mathfrak{V},\mathfrak{U})}\mathbf{K}_{(\mathfrak{V},\mathfrak{U})}^\top - q\boldsymbol{\mu}_{\mathfrak{U}}(\boldsymbol{\mu}^+ + \boldsymbol{\mu}^-)^\top + \frac{q}{4}(\boldsymbol{\mu}^+ + \boldsymbol{\mu}^-)(\boldsymbol{\mu}^+ + \boldsymbol{\mu}^-)^\top$, $\mathbf{L} = \mathbf{K}_{(\mathfrak{V},\mathfrak{X})}\mathbf{K}_{(\mathfrak{V},\mathfrak{X})}^\top - |\mathcal{I}^+|\cdot\boldsymbol{\mu}^+\boldsymbol{\mu}^{+\top} - |\mathcal{I}^-|\cdot\boldsymbol{\mu}^-\boldsymbol{\mu}^{-\top}$ and $\mathbf{b} = \mathbf{1}\cdot b$. The dual versions of (2.36) and (2.37) then have the following form:

$$\begin{array}{ll} \text{minimise}_{\alpha} & \Theta(\boldsymbol{\alpha}) + C\boldsymbol{\alpha}^\top\mathbf{L}\boldsymbol{\alpha} + C_{\mathfrak{U}}\boldsymbol{\alpha}^\top\mathbf{U}\boldsymbol{\alpha} \\ \text{s.t.} & \boldsymbol{\alpha}^\top(\boldsymbol{\mu}^+ - \boldsymbol{\mu}^-) = 2 \end{array} \tag{2.38}$$

and

Dual quadratic programs for the kFDA-oPCA hybrid

$$\begin{array}{ll} \text{minimise}_{\boldsymbol{\alpha},b} & \Theta(\boldsymbol{\alpha}) + C||\boldsymbol{\xi}||^2 + C_{\mathfrak{U}}||\boldsymbol{\vartheta}||^2 \\ \text{s.t.} & \mathbf{K}_{(\mathfrak{X},\mathfrak{V})}\boldsymbol{\alpha} + \mathbf{b} = \mathbf{y}^{(l)} + \boldsymbol{\xi} \\ & \mathbf{K}_{(\mathfrak{U},\mathfrak{V})}\boldsymbol{\alpha} + \mathbf{b} = \boldsymbol{\vartheta} \\ & \boldsymbol{\xi}^\top\mathbf{1}^k = 0 \text{ for } k=\pm. \end{array} \tag{2.39}$$

Again, equation (2.39) is equivalent to the dual version of (2.18) up to the constraints $\boldsymbol{\xi}^\top\mathbf{1}^k = 0$ for $k = \pm$.

After tediously deriving the optimisation problems (2.38) and (2.39), the following proposition finally states their equivalence and therefore establishes the link between (2.35) and the $\mathfrak{U}_{ls}$-SVM via (2.38) and (2.39).

PROPOSITION: For given C and $C_{\mathfrak{U}}$ the optimisation problems (2.38) and (2.39) are equivalent.

PROOF: The proof consists of two steps: In the first step, it is shown that the feasible regions of (2.38) and (2.39) are identical with respect to $\boldsymbol{\alpha}$. The second step consists of showing that the objective functions coincide, thus establishing the equivalence of both optimisation problems.

The feasible regions of (2.38) *and* (2.39) *coincide with respect to* $\boldsymbol{\alpha}$*.*

Let $\boldsymbol{\alpha}$ be feasible for (2.38), i.e. $\boldsymbol{\alpha}^\top(\boldsymbol{\mu}^+ - \boldsymbol{\mu}^-) = 2$. Then

$$\begin{aligned}\boldsymbol{\alpha}^\top\boldsymbol{\mu}^+ - \boldsymbol{\alpha}^\top\boldsymbol{\mu}^- &= \frac{1}{|\mathcal{I}^+|}\boldsymbol{\alpha}^\top\mathbf{K}_{(\mathfrak{X},\mathfrak{Y})}1^+ - \frac{1}{|\mathcal{I}^-|}\boldsymbol{\alpha}^\top\mathbf{K}_{(\mathfrak{X},\mathfrak{Y})}\mathbf{1}^- \\ &= 2\end{aligned}$$

Now b can be chosen such that

$$\frac{1}{|\mathcal{I}^+|}(\mathbf{K}_{(\mathfrak{X},\mathfrak{Y})}\boldsymbol{\alpha} + \mathbf{b})^\top\mathbf{1}^+ = 1 \quad\text{and}\quad \frac{1}{|\mathcal{I}^-|}(\mathbf{K}_{(\mathfrak{X},\mathfrak{Y})}\boldsymbol{\alpha} + \mathbf{b})^\top\mathbf{1}^- = -1.$$

In particular, it holds for $k = \pm$ that

$$\begin{aligned}\frac{1}{|\mathcal{I}^k|}(\mathbf{K}_{(\mathfrak{X},\mathfrak{Y})}\boldsymbol{\alpha} + b)^\top\mathbf{1}^k - \frac{1}{|\mathcal{I}^k|}\mathbf{y}^\top\mathbf{1}^k &= \frac{1}{|\mathcal{I}^k|}(\mathbf{K}_{(\mathfrak{X},\mathfrak{Y})}\boldsymbol{\alpha} + \mathbf{b} - \mathbf{y})^\top\mathbf{1}^k \\ &= \frac{1}{|\mathcal{I}^k|}\boldsymbol{\xi}^\top\mathbf{1}^k \\ &= 0\end{aligned}$$

and therefore $\boldsymbol{\xi}^\top\mathbf{1}^k = 0$. Thus $\boldsymbol{\alpha}$ is feasible for (2.39).

On the other hand, if $\boldsymbol{\alpha}$ is feasible for (2.39) then $\boldsymbol{\xi}^\top\mathbf{1}^k = 0$ for $k = \pm$. Therefore,

$$\begin{aligned}\boldsymbol{\xi}^\top\mathbf{1}^+ &= \frac{1}{|\mathcal{I}^+|}\boldsymbol{\xi}^\top\mathbf{1}^+ \\ &= \frac{1}{|\mathcal{I}^+|}(\mathbf{K}_{(\mathfrak{X},\mathfrak{Y})}\boldsymbol{\alpha} + \mathbf{b} - \mathbf{y})^\top\mathbf{1}^+ \\ &= \boldsymbol{\mu}^+\boldsymbol{\alpha} + b - 1\end{aligned}$$

and

$$\begin{aligned}\boldsymbol{\xi}^\top\mathbf{1}^- &= \frac{1}{|\mathcal{I}^-|}\boldsymbol{\xi}^\top\mathbf{1}^- \\ &= \frac{1}{|\mathcal{I}^-|}(\mathbf{K}_{(\mathfrak{X},\mathfrak{Y})}\boldsymbol{\alpha} + \mathbf{b} - \mathbf{y})^\top\mathbf{1}^- \\ &= \boldsymbol{\mu}^-\boldsymbol{\alpha} + b + 1.\end{aligned}$$

Since

$$\begin{aligned}\boldsymbol{\alpha}^\top(\boldsymbol{\mu}^+ - \boldsymbol{\mu}^-) &= (\boldsymbol{\xi}^\top\mathbf{1}^+ - b + 1) - (\boldsymbol{\xi}^\top\mathbf{1}^- - b - 1) \\ &= 2,\end{aligned}$$

it follows that $\boldsymbol{\alpha}$ is also feasible for (2.38).

The objectives of (2.38) *and* (2.39) *coincide.*

The second step of the proof, again, consists of two parts. In the first part, it is shown that $C\boldsymbol{\alpha}^\top\mathbf{L}\boldsymbol{\alpha} = C||\boldsymbol{\xi}||_2^2$, in the second $C_{\mathfrak{U}}\boldsymbol{\alpha}^\top\mathbf{U}\boldsymbol{\alpha} =$

$C_{\mathfrak{U}}||\boldsymbol{\vartheta}||_2^2$ is demonstrated. Since the difference between (2.38) and (2.39) is just the sum of the two equalities, the identity of (2.38) and (2.39) follows.
First, $C\boldsymbol{\alpha}^\top \mathbf{L}\boldsymbol{\alpha} = C||\boldsymbol{\xi}||_2^2$ is shown:

$$\begin{aligned}
\boldsymbol{\alpha}^\top \mathbf{L}\boldsymbol{\alpha} - ||\boldsymbol{\xi}||_2^2 &= \boldsymbol{\alpha}^\top \mathbf{L}\boldsymbol{\alpha} - \langle \boldsymbol{\xi}, \mathbf{K}_{(\mathfrak{X},\mathfrak{V})}\boldsymbol{\alpha} + \mathbf{b} - \mathbf{y}\rangle \\
&= \boldsymbol{\alpha}^\top (\mathbf{K}_{(\mathfrak{X},\mathfrak{V})}\mathbf{K}_{(\mathfrak{X},\mathfrak{V})} - |\mathcal{I}^+|\boldsymbol{\mu}^+\boldsymbol{\mu}^{+\top} - |\mathcal{I}^-|\boldsymbol{\mu}^-\boldsymbol{\mu}^{-\top})\boldsymbol{\alpha} \\
&\quad -\langle \boldsymbol{\xi}, K_{(\mathfrak{X},\mathfrak{V})}\boldsymbol{\alpha} + \mathbf{b} - \mathbf{y}\rangle \\
&\overset{(i)}{=} \boldsymbol{\alpha}^\top (\mathbf{K}_{(\mathfrak{X},\mathfrak{V})}\mathbf{K}_{(\mathfrak{X},\mathfrak{V})} - |\mathcal{I}^+|\boldsymbol{\mu}^+\boldsymbol{\mu}^{+\top} - |\mathcal{I}^-|\boldsymbol{\mu}^-\boldsymbol{\mu}^{-\top})\boldsymbol{\alpha} \\
&\quad -\langle K_{xv}\boldsymbol{\alpha} + \mathbf{b} - \mathbf{y}, \mathbf{K}_{(\mathfrak{X},\mathfrak{V})}\boldsymbol{\alpha}\rangle \\
&= -\boldsymbol{\alpha}^\top (|\mathcal{I}^+|\boldsymbol{\mu}^+\boldsymbol{\mu}^{+\top} + |\mathcal{I}^-|\boldsymbol{\mu}^-\boldsymbol{\mu}^{-\top})\boldsymbol{\alpha} - \langle \mathbf{b} - \mathbf{y}, \mathbf{y} + \boldsymbol{\xi} - b\rangle \\
&\overset{(ii)}{=} -\boldsymbol{\alpha}^\top (|\mathcal{I}^+|\boldsymbol{\mu}^+\boldsymbol{\mu}^{+\top} + |\mathcal{I}^-|\boldsymbol{\mu}^-\boldsymbol{\mu}^{-\top})\boldsymbol{\alpha} \\
&\quad +\langle \mathbf{b} - \mathbf{y}, \mathbf{b} - \mathbf{y}\rangle \\
&= -\boldsymbol{\alpha}^\top \mathbf{K}_{(\mathfrak{X},\mathfrak{V})} \left(\frac{1}{|\mathcal{I}^+|}\mathbf{1}^+\mathbf{1}^{+\top} + \frac{1}{|\mathcal{I}^-|}\mathbf{1}^-\mathbf{1}^{-\top}\right) \mathbf{K}_{(\mathfrak{X},\mathfrak{V})}\boldsymbol{\alpha} \\
&\quad +\langle \mathbf{b} - \mathbf{y}, \mathbf{b} - \mathbf{y}\rangle \\
&= -(\mathbf{y} + \boldsymbol{\xi} - \mathbf{b})^\top \left(\frac{1}{|\mathcal{I}^+|}\mathbf{1}^+\mathbf{1}^{+\top} + \frac{1}{|\mathcal{I}^-|}\mathbf{1}^-\mathbf{1}^{-\top}\right)(\mathbf{y} + \boldsymbol{\xi} - \mathbf{b}) \\
&\quad +\langle \mathbf{b} - \mathbf{y}, \mathbf{b} - \mathbf{y}\rangle \\
&= -(\mathbf{b} - \mathbf{y})^\top \left(\frac{1}{|\mathcal{I}^+|}\mathbf{1}^+\mathbf{1}^{+\top} + \frac{1}{|\mathcal{I}^-|}\mathbf{1}^-\mathbf{1}^{-\top}\right)(\mathbf{b} - \mathbf{y}) \\
&\quad +\langle \mathbf{b} - y, \mathbf{b} - \mathbf{y}\rangle \\
&= -|\mathcal{I}^+|(b-1)(b-1) - |\mathcal{I}^-|(b+1)(b+1) + \langle \mathbf{b} - \mathbf{y}, \mathbf{b} - \mathbf{y}\rangle \\
&= 0.
\end{aligned}$$

Here, (i) follows from $\langle \boldsymbol{\xi}, \mathbf{b}\rangle = 0$ as well as $\langle \boldsymbol{\xi}, \mathbf{y}\rangle = 0$, while (ii) follows from $\langle \boldsymbol{\xi}, \mathbf{b} - \mathbf{y}\rangle = 0$. Analogously $C_{\mathfrak{U}}\boldsymbol{\alpha}^\top \mathbf{U}\boldsymbol{\alpha} = C_{\mathfrak{U}}||\boldsymbol{\vartheta}||^2$:

$$\begin{aligned}
\boldsymbol{\alpha}^\top \mathbf{U}\boldsymbol{\alpha} - ||\boldsymbol{\vartheta}||^2 &= \boldsymbol{\alpha}^\top \mathbf{U}\boldsymbol{\alpha} - ||\mathbf{K}_{(\mathfrak{U},\mathfrak{V})}\boldsymbol{\alpha} + \mathbf{b}||^2 \\
&= \boldsymbol{\alpha}^\top (\mathbf{K}_{(\mathfrak{V},\mathfrak{U})}\mathbf{K}_{(\mathfrak{U},\mathfrak{V})} - |\mathcal{I}^{\mathfrak{U}}|\boldsymbol{\mu}^{\mathfrak{U}}(\boldsymbol{\mu}^+ + \boldsymbol{\mu}^-)^\top + \\
&\quad \frac{|\mathcal{I}^{\mathfrak{U}}|}{4}(\boldsymbol{\mu}^+ + \boldsymbol{\mu}^-)(\boldsymbol{\mu}^+ + \boldsymbol{\mu}^-)^\top)\boldsymbol{\alpha} - \langle \mathbf{K}_{(\mathfrak{U},\mathfrak{V})}\boldsymbol{\alpha} + \mathbf{b}, \mathbf{K}_{(\mathfrak{U},\mathfrak{V})}\boldsymbol{\alpha} + \mathbf{b}\rangle \\
&= -\boldsymbol{\alpha}^\top (\boldsymbol{\mu}^{\mathfrak{U}}(\boldsymbol{\mu}^+ + \boldsymbol{\mu}^-)^\top - \frac{|\mathcal{I}^{\mathfrak{U}}|}{4}(\boldsymbol{\mu}^+ + \boldsymbol{\mu}^-)(\boldsymbol{\mu}^+ + \boldsymbol{\mu}^-)^\top)\boldsymbol{\alpha} \\
&\quad -2b|\mathcal{I}^{\mathfrak{U}}|\boldsymbol{\mu}^{\mathfrak{U}\top}\boldsymbol{\alpha} - \mathbf{b}^\top\mathbf{b} \\
&= \frac{|\mathcal{I}^{\mathfrak{U}}|}{4}(\boldsymbol{\alpha}^\top(\boldsymbol{\mu}^+ + \boldsymbol{\mu}^- - 4\boldsymbol{\mu}^{\mathfrak{U}}) - 2b)(\boldsymbol{\alpha}^\top(\boldsymbol{\mu}^+ + \boldsymbol{\mu}^-) + 2b) \\
&= 0,
\end{aligned}$$

where the last equality holds because of

$$\begin{aligned}\boldsymbol{\alpha}^\top(\boldsymbol{\mu}^+ + \boldsymbol{\mu}^-) + 2b &= \underbrace{\boldsymbol{\alpha}^\top\boldsymbol{\mu}^+ + b}_{=1} + \underbrace{\boldsymbol{\alpha}^\top\mu^- + b}_{=-1} \\ &= 0.\end{aligned}$$

This completes the proof.

□

Universum examples could be constructed from noise.

The proposition establishes the link between the $\mathfrak{U}_{\mathfrak{ls}}$-SVM and gives valuable insight into the role of the Universum in the $\mathfrak{U}_{\mathfrak{ls}}$-SVM algorithm. Since the Universum examples play the role of samples from the noise distribution in the oPCA part in (2.35), real samples from the noise, if available, can be centered between both classes of training examples and serve as Universum points. This does also agree with the interpretation of the $\mathfrak{U}$-SVM in 2.4.1. However, the $\mathfrak{U}_{\mathfrak{ls}}$-SVM does not have the constraints $\boldsymbol{\xi}^\top\mathbf{1}^k = 0$ for $k = \pm$. These constraints state that the ξ_i for each class must sum to zero, i.e. that the mean projection onto w equals one or minus one, respectively. This can be seen as a balancing constraint for both classes, i.e. it cannot happen that one class shifts the mean of the other class beyond ± 1. If the constraints are removed, the class with more examples gets a higher priority in the optimisation. Thus, there is no principled difference between the $\mathfrak{U}_{\mathfrak{ls}}$-SVM and the hybrid of kFDA and koPCA.

While the above analysis was concerned with the approximation to the Universum inference principle as proposed by [103], the following section establishes a more conceptual relation between the Universum idea of Vapnik [100, 98] and a *maximum entropy* distribution on labels.

2.5 On the Relation between the Universum and Maximum Entropy Distributions

This section relates the *maximum number of contradictions* idea for binary classification to the *maximum entropy* distribution [45] on labels of the Universum points. In order to get to distributions over labels, so called *version spaces* are introduced. Given a certain class of functions $\mathcal{H}$ and a labelled training set $\mathfrak{L} = \{(x_i, y_i)\}_{i=1,\dots,m}$, the version space $\mathcal{V}(\mathfrak{L}) \subseteq \mathcal{H}$ is the set of all functions that classify $\mathfrak{L}$ correctly [64]. As in section 2.3, the considered set of functions is an RKHS $\mathcal{H}$ spanned by a kernel k. The offset b is again assumed to be zero. After introducing version spaces, a general probabilistic framework as in [32, 33, 78] will be imposed on this setting, allowing the assignment of probabilities to certain subsets of $\mathcal{H}$. By choosing a certain likelihood function and a certain loss function, searching for the maximum entropy distributions on labels of Universum points can be seen as a soft version of choosing the equivalence class with the maximum number of contradictions on the Universum $\mathfrak{U}$. The entropy of the distribution of labels of $\mathfrak{U}$ will turn out to be a lower bound for the number of contradictions. Finally, the relation to the Universum algorithm as proposed by [103] will be investigated.

Before turning to the probabilistic setting, an exact definition of version spaces [64] is given.

The version space is a subset of functions that classify the training set correctly.

DEFINITION: Given a set of functions $\mathcal{F}$ and a set of labelled data points $\mathfrak{L} = \{(x_i, y_i)\}_{i=1,\dots,m}$, the *version space* $\mathcal{V}(\mathfrak{L}) \subseteq \mathcal{F}$ is the subset of $\mathcal{F}$ that is consistent with the set $\mathfrak{L}$, that is

$$\forall f \in \mathcal{V}(\mathfrak{L}) : \qquad f(x_i) = y_i.$$

▷

In the primal space, examples $x \in \mathcal{D}$ are points and $h \in \mathbb{S}$ are hyperplanes. In the dual space, examples are hyperplanes and linear functions are points.

Since the focus of this section is on binary classification, indicator functions on the domain $\mathcal{D}$ of the input examples should be considered. However, those are easily obtained by taking the sign $\mathrm{sgn}(h(x))$ of the output by $h \in \mathcal{H}$ on a certain data point x. Keeping that in mind, the sign function will be omitted and everything will be stated in terms of elements of $\mathcal{H}$. Apart from that, this section heavily uses the self duality of the space $\mathcal{H}$ by switching between the primal view, in which all data examples are points and linear functions are hyperplanes, and the dual view, in which all data points are hyperplanes and all linear functions are points. So, when thinking about an element of $\mathcal{H}$, the best is to picture $h \in \mathcal{H}$ as a point in $\mathbb{R}^n$. This intuition is valid in almost all cases since both, $\mathcal{H}$ and $\mathbb{R}^n$, are Hilbert spaces.

The version space for a training set $\mathfrak{L}$ is an intersection of halfspaces.

It is also important to realise how the data examples (x_i, y_i) create a version space on $\mathcal{H}$. Sticking with the dual view, every element k_{x_i} is seen as a hyperplane in $\mathcal{H}$. Before seeing any training example, the version space is just the function space $\mathcal{H}$ itself. The first training example (x_1, y_1) divides it into two parts: the functions that have positive output on k_{x_1} and the functions that have negative output on k_{x_1}. Whichever half agrees with the label y_i is the new version space for (x_1, y_1). When seeing more training examples, this process carries on, cutting the version space in smaller and smaller pieces. Eventually, the version space might become empty. This is the case if $\mathfrak{L}$ is not linearly separable in $\mathcal{H}$. Since this section tries to give a rather conceptual view about the Universum setting in version spaces, the case $\mathcal{V}(\mathfrak{L}) = \emptyset$ is excluded in the following considerations.

Probabilistic model on the version space

For a given version space $\mathcal{V}$, a probabilistic model can be constructed from a distribution over its elements. Let $\mathfrak{L} = \{(x_i, y_i)\}_{i=1,\dots,m}$ be a set of m independently sampled points from the distribution of training points $\mathbf{P}_{\mathsf{XY}}$, and let $\mathfrak{U} = \{z_j\}_{j=1,\dots,q}$ be a set of q independently sampled points from the distribution of Universum points $\mathbf{P}_{\mathsf{U}}$. In the following, $\mathfrak{X} = \{x_i\}_{i=1,\dots,m}$ and $\mathfrak{Y} = \{y_i\}_{i=1,\dots,m}$ are used to denote the input elements and the labels of $\mathfrak{L}$, respectively. Furthermore, a random variable with an exponent m denotes m independent random variables from the same distribution. Two bracketed random variables (XY) represent a sample from the joint distribution $\mathbf{P}_{\mathsf{XY}}$.

Posterior over functions

Let $\mathbf{P}_{\mathsf{H}}$ be a prior over functions in $\mathcal{H}$. Given a likelihood function $\mathbf{P}_{\mathsf{Y}|\mathsf{X},\mathsf{H}}$, the posterior over hypotheses given the data set $\mathfrak{L}$ is given by [38]

$$\begin{aligned}\mathbf{P}_{\mathsf{H}|\mathfrak{L}}(h) &= \frac{\mathbf{P}_{(\mathsf{XY})^m|h}(\mathfrak{L})}{\mathbf{E}_{\mathsf{H}}[\mathbf{P}_{(\mathsf{XY})^m|\mathsf{H}}(\mathfrak{L})]}\mathbf{P}_{\mathsf{H}}(h)\\ &\overset{(i)}{=} \frac{\mathbf{P}_{\mathsf{Y}^m|\mathfrak{X},h}(\mathfrak{Y})}{\mathbf{E}_{\mathsf{H}}[\mathbf{P}_{\mathsf{Y}^m|\mathfrak{X},\mathsf{H}}(\mathfrak{Y})]}\mathbf{P}_{\mathsf{H}}(h),\end{aligned}$$

where (i) follows from

$$\mathbf{P}_{(\mathsf{XY})^m|h}(\mathfrak{L}) = \mathbf{P}_{\mathsf{Y}^m|\mathfrak{X},h}(\mathfrak{Y})\mathbf{P}_{\mathsf{X}^m|h}(\mathfrak{X}) = \mathbf{P}_{\mathsf{Y}^m|\mathfrak{X},h}(\mathfrak{Y})\mathbf{P}_{\mathsf{X}^m}(\mathfrak{X}).$$

PAC likelihood

Choosing the likelihood function to be the so called *PAC-likelihood*

$$\begin{aligned}\mathbf{P}_{\mathsf{Y}|x,h}(y) &= \delta_{y=h(x)}\\ \text{with} \qquad & \delta_{y=h(x)} = \begin{cases} 1 & \text{if } y = h(x) \\ 0 & \text{else}\end{cases},\end{aligned}$$

the posterior simplifies to

$$\mathbf{P}_{\mathsf{H}|\mathfrak{L}}(h) = \begin{cases} \frac{\mathbf{P}_{\mathsf{H}}(h)}{\mathbf{P}_{\mathsf{H}}(\mathcal{V}(\mathfrak{L}))} & \text{if } h \in \mathcal{V}(\mathfrak{L}) \\ 0 & \text{else}\end{cases}. \qquad (2.40)$$

When using the PAC likelihood, the posterior is proportional to the mass of the version space with respect to the prior.

It is important to note that the PAC-likelihood only assigns nonzero probability to the elements of the version space since the delta function is zero outside of $\mathcal{V}$. Therefore, calculating the posterior amounts to simply restricting the prior belief to the version space $\mathcal{V}$ and normalising it to probability mass one.

Once the posterior is computed, the Bayesian strategy mentioned in 1.3.2 can be used to predict a label on a new point x by choosing the label $y \in \mathcal{R}$ that has minimal expected error with respect to the posterior $\mathbf{P}_{\mathsf{H}|\mathfrak{L}}(h)$ and a given loss function ℓ:

$$B_{\mathfrak{L}}(x) = \text{argmin}_{y\in\mathcal{R}}\mathbf{E}_{\mathsf{H}|\mathfrak{L}}[\ell(\mathsf{H}(x), y)].$$

If the loss function is chosen to be the *zero-one loss* $\ell_{0\text{–}1}(y, y') = 1-\delta_{y=y'}$, the predicted label is just the label corresponding to the larger area of the version space with respect to the probability measure $\mathbf{P}_{\mathsf{H}}$ [38] since

$$\begin{aligned}
B_{\mathfrak{L}}(x) &= \mathbf{argmin}_{y\in\mathcal{R}} \mathsf{E}_{\mathsf{H}|\mathfrak{L}}[\ell_{0-1}(\mathsf{H}(x), y)] \\
&= \mathbf{argmin}_{y\in\mathcal{R}} \int_{\mathcal{H}} \ell_{0-1}(h(x), y)\mathsf{P}_{\mathsf{H}|\mathfrak{L}}(h)dh \\
&= \mathbf{argmin}_{y\in\mathcal{R}} \int_{\mathcal{H}} \ell_{0-1}(h(x), y)\frac{\mathsf{P}_{\mathsf{H}}(h)}{\mathsf{P}_{\mathsf{H}}(\mathcal{V}(\mathfrak{L}))}dh \\
&= \mathbf{argmin}_{y\in\mathcal{R}} \int_{\mathcal{H}} (1-\delta_{h(x),y})\mathsf{P}_{\mathsf{H}}(h)dh \\
&= \mathbf{argmin}_{y\in\mathcal{R}} \int_{\mathcal{H}} \mathsf{P}_{\mathsf{H}}(h)dh - \int_{\mathcal{H}} \delta_{h(x)=y}\mathsf{P}_{\mathsf{H}}(h)dh \\
&= \mathbf{argmin}_{y\in\mathcal{R}} 1 - \int_{\mathcal{H}} \delta_{h(x)=y}\mathsf{P}_{\mathsf{H}}(h)dh \\
&= \mathbf{argmax}_{y\in\mathcal{R}} \int_{\mathcal{H}} \delta_{h(x)=y}\mathsf{P}_{\mathsf{H}}(h)dh.
\end{aligned}$$

Therefore, speaking figuratively, the Bayesian prediction strategy proceeds as follows. When predicting the label of a new point x, the version space $\mathcal{V}$ is divided into two parts: the part that classifies x positively and the part that classifies it negatively. The prediction then consists of choosing the half (one of them might be empty) that has a larger probability mass with respect to the prior P_{H}. As in [32, 33, 38], $B_{\mathfrak{L}}(x)$ is called *prediction strategy* instead of *prediction function* since, in most cases, there might not be a single function $h \in \mathcal{H}$ that predicts exactly the same labels on all inputs.

Bayesian prediction strategy for ℓ_{0-1} is equivalent to choosing the larger half of $\mathcal{V}$ induced by a test example

Nevertheless, to get down to a single classifier h_{bp}, one can search for an element in $\mathcal{H}$ that mimics the Bayesian prediction strategy $B_{\mathfrak{L}}$ as well as possible. This element h_{bp} is called *Bayes Point* [38]:

The Bayes Point $h_{bp} \in \mathcal{H}$ is a single function that best mimics the Bayesian prediction strategy $B_{\mathfrak{L}}$

$$h_{bp}(x) = \mathbf{argmin}_{h\in\mathcal{H}} \mathsf{E}_{\mathsf{X}}[\mathsf{E}_{\mathsf{H}|\mathfrak{L}}[\ell(h(\mathsf{X}), \mathsf{H}(\mathsf{X}))]].$$

The authors of [38] show that under certain conditions the Bayes point can be approximated by the center of mass of the version space $\mathcal{V}$.

In the following treatment the admissible class of functions is assumed to be the sphere $\mathbb{S} \subseteq \mathcal{H}$, i.e. $\mathbb{S} := \{h \in \mathcal{H}|\, ||h||^2 = \langle h, h\rangle = 1\}$. This is no restriction since normalising the prediction function h does not change the sign of its output. As introduced at the beginning of this section, it is possible to picture an $h \in \mathbb{S}$ as a point on a sphere in $\mathbb{R}^n$ and the image $k_x \in \mathcal{H}$ of an example $x \in \mathcal{D}$ as a hyperplane in $\mathbb{R}^n$ cutting through the sphere. Figure 2.7 gives a schematic illustration for three labelled examples. The goal in the following paragraphs is to construct a new prediction strategy that uses the Universum points in $\mathfrak{U}$ and to relate this prediction strategy to a maximum entropy distribution of labels on the elements of $\mathfrak{U}$ induced by a new test point x.

Let x be a new test point which divides the version space $\mathcal{V}(\mathfrak{L}) \subset \mathbb{S}$ into two nonempty halves, meaning there are elements in $\mathcal{V}(\mathfrak{L})$ that

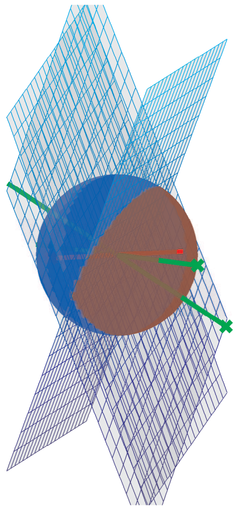


Figure 2.7: Illustration of a version space on $\mathbb{S} \subseteq \mathcal{H} = \mathbb{R}^3$ for two positive (green normal) and one negative (red normal) examples. In this figure $\mathbb{R}^3$ serves as primal and dual space at the same time. All possible linear functions of norm one are represented by the sphere. The examples are vectors in the primal space, and normals of hyperplanes in the dual space.

classify x positively as well as elements that classify it negatively. Instead of choosing the half with the larger probability mass as in the case of the Bayesian prediction strategy with PAC-likelihood and zero-one loss, the new prediction strategy proceeds as follows: All hyperplanes corresponding to points of $\mathfrak{U}$ that cut through $\mathcal{V}(\mathfrak{L})$ will intersect with at least one half of $\mathcal{V}(\mathfrak{L})$ induced by the new data point x. Every half induced by x corresponds to a probability distribution on the labels of a point $z \in \mathfrak{U}$ by simply considering the new version space $\mathcal{V}(\mathfrak{L} \cup (x, y))$, where y is the label of x corresponding to that half, and using the distribution on Y as stated in (2.40) for the elements of $\mathfrak{U}$. To illustrated this idea, let the considered half of $\mathcal{V}(\mathfrak{L})$ be the one that assigns positive labels to x. This half is the version space of $\mathfrak{L} \cup (x, +)$ (where "+" is used as a shortcut for "+1"). Just as above, the posterior as in (2.40) can be computed by using the PAC-likelihood

New prediction strategy: Predict the label y for x which gives rise to a higher entropy on the Universum points.

$$\mathsf{P}_{\mathsf{H}|\mathfrak{L}\cup(x,+)}(h) \quad = \quad \begin{cases} \frac{\mathsf{P}_{\mathsf{H}}(h)}{\mathsf{P}_{\mathsf{H}}(\mathcal{V}(\mathfrak{L}\cup(x,+)))} & \text{if } h \in \mathcal{V}(\mathfrak{L} \cup (x,+)) \\ 0 & \text{otherwise} \end{cases} .$$

This probability distribution can be used to define a distribution on labels of Universum points $z \in \mathfrak{U}$ by taking the probability mass of the part of $\mathcal{V}(\mathfrak{L} \cup (x, +))$ that assigns a certain label to z and renormalising it to one:

$$\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,+),z}(y) = \frac{\mathbf{P}_{\mathsf{H}|\mathfrak{L}\cup(x,+)}\left(\mathcal{V}(\mathfrak{L}\cup(x,+)\cup(z,y))\right)}{\mathbf{P}_{\mathsf{H}|\mathfrak{L}\cup(x,+)}\left(\mathcal{V}(\mathfrak{L}\cup(x,+))\right)}.$$

This corresponds to a two layered approach. First, the new test point x cuts $\mathcal{V}(\mathfrak{L})$ into two halves. Then each Universum point cuts each of the halves again into two. Thus, given a new test point x, one can choose between two probability distributions on labels of a single Universum point z. Each of these label distributions has a Shannon entropy [84]:

Each test point x divides the version space into two parts. Each part can be seen as a probability distribution on labels for $z \in \mathfrak{U}$. This distribution has an entropy.

$$\mathbf{H}[\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z}] = -\sum_{y'\in\{\pm\}} \mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z}(y') \log \mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z}(y').$$

The new prediction strategy will be to choose that label y for x which maximises the entropy on all Universum points in $\mathfrak{U}$

$$U_{\mathfrak{L},\mathfrak{U}}(x) = \operatorname{argmax}_{y\in\mathcal{Y}} \sum_{z\in\mathfrak{U}} \mathbf{H}[\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z}] \qquad (2.41)$$

or, even more general, choose the label y for x that maximises the expected entropy according to the distribution $\mathbf{P}_{\mathsf{U}}$ on the Universum points

$$U_{\mathfrak{L},\mathbf{P}_{\mathsf{U}}}(x) = \operatorname{argmax}_{y\in\mathcal{Y}} \mathbf{E}_{\mathsf{U}}[\mathbf{H}[\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),\mathsf{U}}]]. \qquad (2.42)$$

As described above, $U_{\mathfrak{L},\mathfrak{U}}(x)$ is a choice over distributions that correspond to a certain label y on x. By choosing the distribution with maximum entropy, a label for x is chosen. It is also important to note that taking the expectation over $\mathbf{P}_{\mathsf{U}}$ with respect to a single sample $z \sim \mathbf{P}_{\mathsf{U}}$ at each time is equivalent to taking the expectation over q samples $\mathsf{U}^q = \mathfrak{U}$ with respect to $\mathbf{P}_{\mathsf{U}}$. This is due to the fact that the elements

Choosing a distribution $\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),\mathsf{U}}$ for labels on $z \in \mathfrak{U}$ is equivalent to choosing a label y for x.

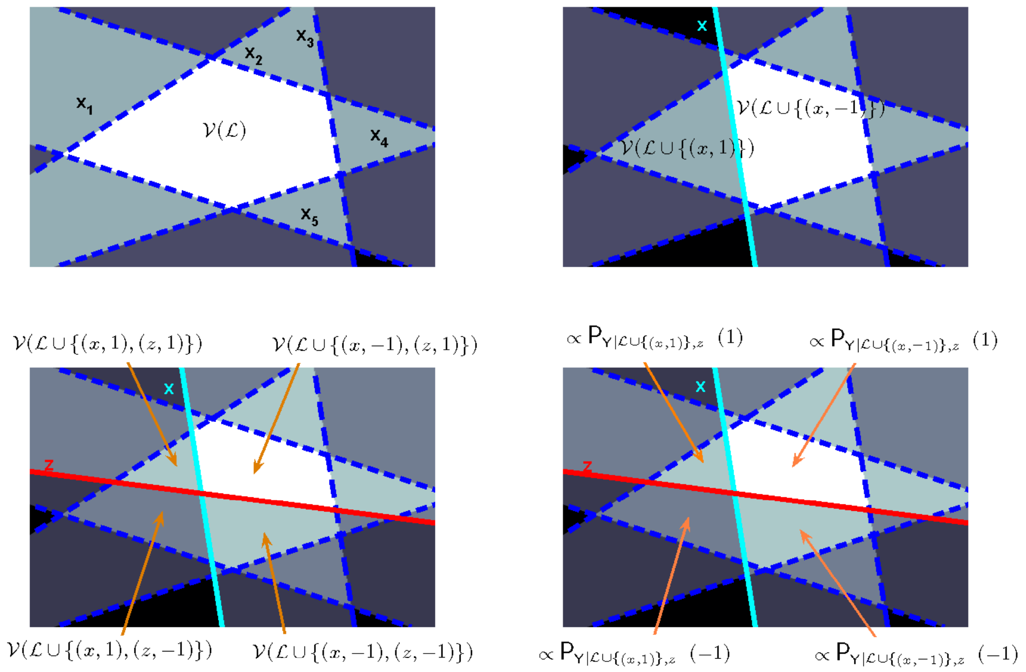

x_1
x_2
x_3
x_4
x_5
$\mathcal{V}(\mathcal{L})$
x
$\mathcal{V}(\mathcal{L} \cup \{(x,-1)\})$
$\mathcal{V}(\mathcal{L} \cup \{(x,1)\})$
$\mathcal{V}(\mathcal{L} \cup \{(x,1),(z,1)\})$
$\mathcal{V}(\mathcal{L} \cup \{(x,-1),(z,1)\})$
x
z
$\mathcal{V}(\mathcal{L} \cup \{(x,1),(z,-1)\})$
$\mathcal{V}(\mathcal{L} \cup \{(x,-1),(z,-1)\})$
$\propto \mathsf{P}_{\mathsf{Y}|\mathcal{L}\cup\{(x,1)\},z}\ (1)$
$\propto \mathsf{P}_{\mathsf{Y}|\mathcal{L}\cup\{(x,-1)\},z}\ (1)$
x
z
$\propto \mathsf{P}_{\mathsf{Y}|\mathcal{L}\cup\{(x,1)\},z}\ (-1)$
$\propto \mathsf{P}_{\mathsf{Y}|\mathcal{L}\cup\{(x,-1)\},z}\ (-1)$

Figure 2.8: Illustration of the decision strategy $U_{\mathfrak{L},\mathfrak{U}}(x)$. The examples of $\mathfrak{L}$ give rise to the version space $\mathcal{V}(\mathfrak{L})$ (*top left*). A new example from $\mathbf{P}_{\mathsf{X}}$ divides $\mathcal{V}(\mathfrak{L})$ into the two parts $\mathcal{V}(\mathfrak{L} \cup (x,+))$ and $\mathcal{V}(\mathfrak{L} \cup (x,-))$ (*top right*). A Universum point $z \sim \mathbf{P}_{\mathsf{U}}$ further cuts those two halves into four in total: $\mathcal{V}(\mathfrak{L} \cup (x,y),(x,y'))$ for $y, y' \in \{\pm\}$ (*bottom left*). With a prior $\mathbf{P}_{\mathsf{H}}$ on elements of $\mathcal{H}$, each part has a certain probability mass with respect to $\mathbf{P}_{\mathsf{H}}$ (*bottom right*). The new prediction strategy is choosing that label y for x for which the conditional distribution $\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z}$ on labels y' of Universum points has a higher Shannon entropy.

of $\mathfrak{U}$ are assumed to be sampled independently from $\mathbf{P}_{\mathsf{U}}$ and therefore

$$
\begin{aligned}
&\mathbf{E}_{(\mathsf{U})^q}\left[\sum_{i=1}^{q}\mathbf{H}[\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z}]\right]\\
=\;& \underbrace{\int\cdots\int}_{q\text{ times}}\sum_{i=1}^{q}\mathbf{H}[\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z_i}]dp(z_q)\dots dp(z_1)\\
=\;& \underbrace{\int\cdots\int}_{q-1\text{ times}}\left(\int\sum_{i=1}^{q-1}\mathbf{H}[\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z_i}]dp(z_q)+\int\mathbf{H}[\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z_q}]dp(z_q)\right)dp(z_{q-1})\dots dp(z_1)\\
=\;& \underbrace{\int\cdots\int}_{q-1\text{ times}}\sum_{i=1}^{q-1}\mathbf{H}[\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z_i}]+\mathbf{E}_{\mathsf{U}}\left[H[\mathsf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z}]\right]dp(z_{q-1})\dots dp(z_1)\\
=\;& \underbrace{\int\cdots\int}_{q-1\text{ times}}\sum_{i=1}^{q-1}\mathbf{H}[\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z_i}]dp(z_{q-1})\dots dp(z_1)+\mathbf{E}_{\mathsf{U}}\left[\mathbf{E}[\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z}]\right]\\
=\;& \sum_{i=1}^{q}\mathbf{E}_{\mathsf{U}}\left[\mathbf{H}[\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z}]\right]\\
=\;& q\mathbf{E}_{\mathsf{U}}\left[\mathbf{H}[\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z}]\right].
\end{aligned}
$$

This means that the entropy over samples of size q and samples of size one only differ by a multiplicative factor of q. Therefore, considering the expectation over $\mathbf{P}_{\mathsf{U}}$ is equivalent to taking the expectation over $\mathbf{P}_{(\mathsf{U})^q}$ when maximising the expected entropy on the Universum points.

Relation between maximum entropy on label distributions and maximum number of contradictions.

How does the prediction strategy $U_{\mathfrak{L},\mathfrak{U}}(x)$ relate to the idea of maximising the number of contradictions on $\mathfrak{U}$? To investigate this question, the idea of Vapnik has to be embedded into the described setting first. Given a sample of training points $\mathfrak{L}$, a Universum $\mathfrak{U}$ and a new test point x, each half $\mathcal{V}(\mathfrak{L}\cup(x,\pm))$ corresponds to an equivalence class on $\mathcal{V}(\mathfrak{L})$ as described in 2.1.2. Now, the idea of maximising the number of contradictions embedded into version spaces is in fact very simple: Assuming that x cuts $\mathcal{V}(\mathfrak{L})$ into two non-empty parts, the goal is to choose the half or equivalence class or label for x (all three notions are equivalent) that has a larger number of contradictions on $\mathfrak{U}$. A half of $\mathcal{V}$ induces a contradiction on a Universum point $z\in\mathfrak{U}$ if and only if the corresponding element $k_z\in\mathcal{H}$ intersects this half, because only then $\mathcal{V}(\mathfrak{L}\cup(x,y))$ contains functions for all possible labels on z. It might also happen that k_z cuts through both halves, therefore inducing a contradiction on both equivalence classes. An important observation is that the entropy of a distribution of labels on a single Universum point z is nonzero if and only if k_z cuts through the half that gives rise to this distribution. This is due to the fact that for a probability distribution on two elements the entropy $\mathbf{H}[p,1-p]=-p\log p-(1-p)\log(1-p)$ is zero if and only if $p=0$ or $p=1$. If $p=\frac{1}{2}$ the entropy is maximal with $\mathbf{H}[\frac{1}{2},\frac{1}{2}]=1$. Taking these notions together, it is possible to lower bound the maximum number of contradictions in terms of the entropy

$z\in\mathfrak{U}$ is a contradiction if k_z cuts the version space.

The entropy of $\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z}$ is nonzero if and only if $z\in\mathfrak{U}$ is a contradiction on $\mathcal{V}(\mathfrak{L}\cup(x,y))$.

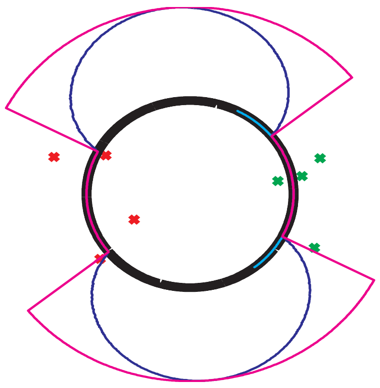


Figure 2.9: Illustration of the link between the entropy on $\mathbf{P}_{\mathsf{Y}|\mathfrak{L},z}$ and the number of contradictions. The space of all considered functions $\mathbb{S} \subseteq \mathcal{H}$ and the set of all Universum points is depicted by the black circle (to make it simple all Universum points are also assumed to have norm one). The red and green points are training examples. The version space is indicated by the cyan part in the circle. The magenta and blue line display the number of contradictions and the entropy for the respective Universum point on the circle: For every possible point (ϕ, r) (in polar coordinates) on the circle the radial component r is enlongated by an amount proportional to the number of contradictions or the entropy, respectively.

as defined in equation (2.41).

Lemma: The number of contradictions $\Gamma_{\mathfrak{L},x}(y,\mathfrak{U})$ on a Universum $\mathfrak{U}$ of an equivalence class induced by (x,y) is lower bounded by the sum of the entropies as defined in (2.41).

The number of contradictions are lower bounded by the entropy.

PROOF: Both, the number of contradictions $\Gamma_{\mathfrak{L},x}(y,\mathfrak{U})$ and $\sum_{z\in\mathfrak{U}} \mathbf{H}[\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z}]$, are greater than zero if k_z cuts the version space $\mathcal{V}(\mathfrak{L}\cup(x,y))$ and zero otherwise. Since each section by an element of $\mathfrak{U}$ increases $\Gamma_{\mathfrak{L},x}(y,\mathfrak{U})$ by one and since the maximal value of $\mathbf{H}[\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z}]$ is one, i.e. when k_z divides $\mathcal{V}(\mathfrak{L}\cup(x,y))$ into two halves of equal size, the following relation holds:

$$0 \leq \sum_{z\in\mathfrak{U}} \mathbf{H}[\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z}] \leq \Gamma_{\mathfrak{L},x}(y,\mathfrak{U}) \leq q.$$

This proves the stated lemma.

□

Figure 2.9 illustrates the situation for a single Universum point. The lemma establishes a link between the prediction strategy $U_{\mathfrak{L},\mathfrak{U}}(x)$ and the strategy proposed by Vapnik [100, 98]. One implication is that maximising $\sum_{z\in\mathfrak{U}} \mathbf{H}[\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z}]$ leads to a large number of contradictions. However, since $\Gamma_{\mathfrak{L},x}(y,\mathfrak{U})$ is a much cruder measure, the entropy can be very small while the number of contradictions is very large. This is the case if the elements of $\mathfrak{U}$ only divide $\mathcal{V}(\mathfrak{L}\cup(x,y))$ into very unequal parts, i.e. one half has very little probability mass. In this situation, the entropy will be very low, but since $\Gamma_{\mathfrak{L},x}(y,\mathfrak{U})$ acts like a step function that immediately jumps to one once k_z cuts through $\mathcal{V}(\mathfrak{L}\cup(x,y))$, z counts as a full contradiction. Nevertheless, the entropy captures an intuitive notion of the usage of the Universum as already described in 2.1.2. The Universum is thought to be a set which the resulting classifier should be most unspecific about. Since a large entropy implies almost equal probabilities on the two different labels for a Universum point z, both notions agree.

The Entropy of $\mathsf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z}$ is only a very loose lower bound on $\Gamma_{\mathfrak{L},x}(y,\mathfrak{U})$.

Maximising the entropy of $\mathsf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z}$ captures the intuition of being unspecific about the elements of $\mathfrak{U}$

To conclude this section, the relationship between the maximum entropy approach as stated in equation (2.42) and the loss function on the Universum points as used by the algorithm stated in 2.2.1.1 is investigated.

Since $U_{\mathfrak{L},\mathfrak{U}}(x)$ is a prediction strategy and not a single function, $U_{\mathfrak{L},\mathfrak{U}}(x)$ must be approximated by a single element $h\in\mathcal{H}$ in order to relate it to the algorithmic relaxation. Here, the same idea as for the Bayesian prediction strategy $B_{\mathfrak{L}}(x)$ and the Bayes point h_{bp} can be employed: $U_{\mathfrak{L},\mathfrak{U}}(x)$ is approximated by the element $h_{ep}\in\mathcal{V}(\mathfrak{L})$ that best mimics the behaviour of $U_{\mathfrak{L},\mathfrak{U}}(x)$ for all possible choices of $\mathfrak{L}$ and $\mathfrak{U}$. Since the h_{ep} is defined analogously to the Bayes Point above, it shall be called *Entropy Point:*

$U_{\mathfrak{L},\mathfrak{U}}(x)$ is approximated by the entropy point $h_{ep}\in\mathcal{V}(\mathfrak{L})$.

$$h_{ep} \quad := \quad \operatorname{argmax}_{h\in\mathcal{H}}\mathsf{E}_{\mathsf{X}}[\mathsf{E}_{\mathsf{U}}[\mathsf{H}[\mathsf{P}_{\mathsf{Y}|\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X})),\mathsf{U}}]]].$$

The entropy point h_{ep} is now a single function and the relation to the loss function on the Universum points of a $\mathfrak{U}$-SVM can be investigated. The loss function for the labelled points is enforced by working inside a version space, which has loss zero on the labelled examples by definition. The following considerations do not take the SVM regularizer into account, since there is no margin maximisation in the maximum entropy approach. One key in the relation between the maximum entropy on labels and the ε-insensitive loss on the Universum points is the center of mass of $\mathcal{V}(\mathfrak{L})$ with respect to the prior P_{H}:

When considering a version space $\mathcal{V}(\mathfrak{L})$ that has been divided into two halves $\mathcal{V}(\mathfrak{L}\cup(x,+))$ and $\mathcal{V}(\mathfrak{L}\cup(x,-))$ by a test example x, one can make the following observations about the dependency between the entropy and the direction k_z points to: As already mentioned above, the entropy $\mathsf{H}[\mathsf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z}]$ is maximal when dividing $\mathcal{V}(\mathfrak{L}\cup(x,y))$ into two parts of equal size, i.e. of probability mass

All elements of $\mathfrak{U}$ that are orthogonal to the center of mass h_{cm} of $\mathcal{V}(\mathfrak{L}\cup(x,y))$ with respect to to P_{H}, have maximal entropy

$$\mathsf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z}(-) \quad = \quad \mathsf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z}(+) = \frac{1}{2}.$$

Now, for each $\mathcal{V}(\mathfrak{L}\cup(x,y))$ there is a point h_{cm} with the property that a hyperplane $E_{k_z}=\{h\in\mathcal{H}|\langle h,k_z\rangle=0\}$ containing h_{cm} has maximum

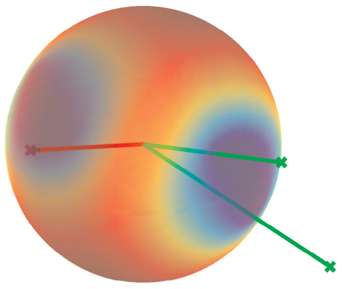


Figure 2.10: Relation between the direction of a Universum point and the entropy on its label-distribution. The sphere represents the set of all function h and all Universum points that are assumed to have norm one for simplicity. The red and green lines depict the normals of hyperplanes induced by training points. Each position on the sphere was colored according to the entropy with respect to a uniform prior P_H (red indicates high values).

entropy. This point h_{cm} is the center of mass of $\mathcal{V}(\mathfrak{L} \cup (x, y))$ since the center of mass has the property that no matter how $\mathcal{V}(\mathfrak{L} \cup (x, y))$ is divided into two pieces, each piece must have the same mass as the other one. This means, on the other hand, that all k_z that are contained in the hyperplane $E_{h_{cm}}$, i.e. $\langle h_{cm}, k_z \rangle = 0$, have entropy $\mathsf{H}[\mathsf{P}_{\mathsf{Y}|\mathcal{S}\cup(x,y),z}] = 1$ on $\mathcal{V}(\mathfrak{L} \cup (x, y))$. When moving an element k_z away from $E_{h_{cm}}$, the hyperplane with k_z as normal divides the version space into two more and more unequal parts and therefore the entropy $\mathsf{H}[\mathsf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z}]$ decreases. Once k_z has been moved so far that the hyperplane with normal k_z does not cut $\mathcal{V}(\mathfrak{L} \cup (x, y))$ anymore, then $\mathsf{H}[\mathsf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z}] = 0$. So, when looking at the sphere, the support of $\mathsf{H}[\mathsf{P}_{\mathsf{Y}|\mathfrak{L}\cup(x,y),z}]$ forms a belt with varying width that has high entropy in the middle and low entropy at the border. Figure 2.10 illustrates this idea.

Minimal Kullback-Leibler-divergence to the uniform label distribution P^0_Y implies maximal entropy

In order to relate the center of mass h_{cm} to the solution of a $\mathfrak{U}$-SVM, it is helpful to define the Entropy Point in terms of the minimal *Kullback-Leibler divergence* [18] between $\mathsf{P}_{\mathsf{Y}|\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X})),\mathsf{U}}$ and the uniform distribution on labels P^0_Y:

$$h_{ep} \quad = \quad \text{argmin}_{h\in\mathcal{H}} \mathsf{E}_\mathsf{X}[\mathsf{E}_\mathsf{U}[\mathsf{D}[\mathsf{P}_{\mathsf{Y}|\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X})),\mathsf{U}} || \mathsf{P}^0_\mathsf{Y}]]].$$

Both definitions are equivalent since zero KL-divergence to P^0_Y implies maximal entropy. By employing two relaxations, the solution of the algorithm in 2.2.1.1 can be seen as an approximation to the Entropy point h_{ep}:

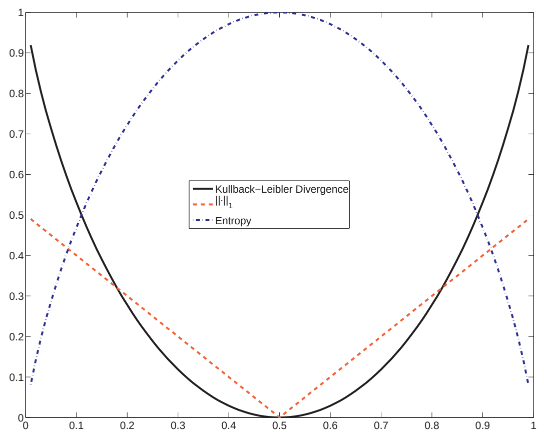


Figure 2.11: Graphs of the Kullback-Leibler-divergence, the L_1 loss and the entropy.

$$\begin{aligned}
h_{ep}(\mathfrak{L}) &= \operatorname{argmin}_{h\in\mathcal{V}(\mathfrak{L})}\mathbf{E}_{\mathsf{X}}[\mathbf{E}_{\mathsf{U}}[\mathbf{D}[\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X})),\mathsf{U}}||\mathbf{P}^0_{\mathsf{Y}}]]] \\
&= \operatorname{argmin}_{h\in\mathcal{V}(\mathfrak{L})}\mathbf{E}_{\mathsf{X}}\left[\mathbf{E}_{\mathsf{U}}\left[\sum_{y\in\{\pm\}}\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X})),\mathsf{U}}(y)\log\frac{\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X})),\mathsf{U}}(y)}{\mathbf{P}^0_{\mathsf{Y}}(y)}\right]\right] \\
&= \operatorname{argmin}_{h\in\mathcal{V}(\mathfrak{L})}\mathbf{E}_{\mathsf{X}}\left[\mathbf{E}_{\mathsf{U}}\left[\sum_{y\in\{\pm\}}\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X})),\mathsf{U}}(y)\cdot(\log\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X})),\mathsf{U}}(y)-\log\mathbf{P}^0_{\mathsf{Y}}(y))\right]\right] \\
&= \operatorname{argmin}_{h\in\mathcal{V}(\mathfrak{L})}\mathbf{E}_{\mathsf{X}}\left[\mathbf{E}_{\mathsf{U}}\left[\sum_{y\in\{\pm\}}\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X})),\mathsf{U}}(y)\log\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X})),\mathsf{U}}(y)\right]\right]-\log\frac{1}{2} \\
&= \operatorname{argmin}_{h\in\mathcal{V}(\mathfrak{L})}\mathbf{E}_{\mathsf{X}}\left[\mathbf{E}_{\mathsf{U}}\left[\sum_{y\in\{\pm\}}\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X})),\mathsf{U}}(y)\log\mathbf{P}_{\mathsf{Y}|\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X})),\mathsf{U}}(y)\right]\right] \\
&= \operatorname{argmin}_{h\in\mathcal{V}(\mathfrak{L})}\mathbf{E}_{\mathsf{X}}\left[\mathbf{E}_{\mathsf{U}}\left[\sum_{y\in\{\pm\}}\frac{\mathbf{P}_{\mathsf{H}}(\mathcal{V}(\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X})),(\mathsf{U},y)))}{\mathbf{P}_{\mathsf{H}}(\mathcal{V}(\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X}))))}\log\frac{\mathbf{P}_{\mathsf{H}}(\mathcal{V}(\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X})),(\mathsf{U},y)))}{\mathbf{P}_{\mathsf{H}}(\mathcal{V}(\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X}))))}\right]\right] \\
&\approx \operatorname{argmin}_{h\in\mathcal{V}(\mathfrak{L})}\mathbf{E}_{\mathsf{X}}\left[\mathbf{E}_{\mathsf{U}}\left[\sum_{y\in\{\pm\}}\left|\mathbf{P}_{\mathsf{H}}(\mathcal{V}(\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X})),(\mathsf{U},y)))-\frac{1}{2}\mathbf{P}_{\mathsf{H}}(\mathcal{V}(\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X}))))\right|\right]\right] \quad (2.43) \\
&\approx \operatorname{argmin}_{h\in\mathcal{V}(\mathfrak{L})}\mathbf{E}_{\mathsf{X}}\left[\mathbf{E}_{\mathsf{U}}\left[\left|\langle h_{cm|\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X}))},k_{\mathsf{U}}\rangle\right|\frac{1}{||k_{\mathsf{U}}||}\right]\right], \quad (2.44)
\end{aligned}$$

In (2.43) the Kullback-Leibler divergence is approximated by the L_1 loss (see also figure 2.11), where the L_1 loss is applied to the deviation of the mass of $\mathcal{V}(\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X})),(\mathsf{U},y))$ from half the mass of $\mathcal{V}(\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X})))$, because $\frac{\mathbf{P}_{\mathsf{H}}(\mathcal{V}(\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X})),(\mathsf{U},y)))}{\mathbf{P}_{\mathsf{H}}(\mathcal{V}(\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X}))))}=\frac{1}{2}$ maximises the entropy.

By employing two relaxations, a relation of the approach in the version space to the algorithmic implementation can be established.

The second relaxation (2.44) assumes that the deviation of the mass of $\mathcal{V}(\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X})),(\mathsf{U},y))$ of the version space from half of the mass of $\mathcal{V}(\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X})),(\mathsf{U},y))$ can be approximated by $\left|\langle h_{cm|\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X}))},k_{\mathsf{U}}\rangle\right|\frac{1}{||k_{\mathsf{U}}||}$.

This relaxation can be motivated by the following considerations:

Every hyperplane induced by a Universum point z that contains $h_{cm|\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X}))}$ divides $\mathcal{V}(\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X})))$ into two halves of equal mass. Therefore, if $\langle h_{cm|\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X}))}, k_z\rangle = 0$, the entropy of z on $\mathcal{V}(\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X})))$ is maximal. As soon as $|\langle h_{cm|\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X}))}, k_z\rangle|$ grows by moving k_z inside the half-space $\{\lambda_1 k_z + \lambda_2 h | \lambda_1 \geq 0\}$, the mass corresponding to $y = 1$ (or $y = -1$ depending on the direction of movement) becomes smaller proportional to the angle between k_z and the hyperplane $E_{h_{cm}}$, until k_z does not cut $\mathcal{V}(\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X})))$ anymore. From that point on, the entropy is zero. Therefore, $\mathbf{P}_{\mathsf{H}}(\mathcal{V}(\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X})),(\mathsf{U},y))) - \frac{1}{2}\mathbf{P}_{\mathsf{H}}(\mathcal{V}(\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X}))))$ can be assumed to be proportional to $\frac{1}{||k_z||}|\langle h_{cm|\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X}))}, k_z\rangle|$ in a limited interval, thus motivating the second relaxation.

By replacing the expectation $\mathbf{E}_{\mathsf{U}}$ by the sum over all elements in $\mathfrak{U}$ and considering only one test point x instead of taking the expectation $\mathbf{E}_{\mathsf{X}}$, (2.44) can already be used for an algorithmic implementation of transductive inference with a Universum, as initially intended by Vapnik [100, 98].

(2.44) *could already be used for an algorithmic implementation of transductive inference with a Universum.*

However, the unlabelled examples have been ignored in the implementation of [103]. The next step is therefore to investigate the relation of h_{ep} to the minimiser h^* of (2.44).

Embedding the objective of the algorithm of [103] into the framework above, the desired function of the algorithm of 2.2.1.1 can be approximately described as:

$$\tilde{h} : = \operatorname{argmin}_{h\in\mathcal{V}(\mathfrak{L})} \mathbf{E}_{\mathsf{U}}[|\langle h, k_{\mathsf{U}}\rangle| \frac{1}{||k_{\mathsf{U}}||}].$$

In fact, in the algorithm of 2.2.1.1 k_{U} is not normalised. The algorithm which pursues this strategy is the Cone-Universum $\mathfrak{U}_{\mathfrak{c}}$-SVM described in 2.3.

The Cone-Universum $\mathfrak{U}_{\mathfrak{c}}$-SVM resembles (2.44) *is more closely.*

The condition for $\tilde{h}$ being equal to the minimiser h^* of (2.44) is described by

$$\begin{aligned} \forall h \in \mathbb{S} : \ |\langle h, k_z\rangle| \frac{1}{||k_z||} &= \mathbf{E}_{\mathsf{X}}[|\langle h_{cm|\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X}))}\rangle| \frac{1}{||k_z||}] \\ \Leftrightarrow \forall h \in \mathbb{S} : \ |\langle h, k_z\rangle| &= \mathbf{E}_{\mathsf{X}}\left[\ \left|\langle \mathbf{E}_{\mathsf{H}|\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X}))}[\mathsf{H}], k_z\rangle\right|\ \right], \end{aligned}$$

since

$$\begin{aligned} h^* &= \operatorname{argmin}_{h\in\mathcal{V}(\mathfrak{L})} \mathbf{E}_{\mathsf{X}}\left[\mathbf{E}_{\mathsf{U}}\left[|\langle h_{cm|\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X}))}, k_{\mathsf{U}}\rangle| \frac{1}{||k_{\mathsf{U}}||}\right]\right] \\ &= \operatorname{argmin}_{h\in\mathcal{V}(\mathfrak{L})} \mathbf{E}_{\mathsf{U}}\left[\mathbf{E}_{\mathsf{X}}\left[|\langle h_{cm|\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X}))}, k_{\mathsf{U}}\rangle| \frac{1}{||k_{\mathsf{U}}||}\right]\right] \\ &= \operatorname{argmin}_{h\in\mathcal{V}(\mathfrak{L})} \mathbf{E}_{\mathsf{U}}\left[|\langle h, k_{\mathsf{U}}\rangle| \frac{1}{||k_{\mathsf{U}}||}\right] \\ &= \tilde{h} \end{aligned}$$

This condition is trivially fulfilled if support$\mathbf{P}_{\mathsf{H}} \subseteq (\text{support}\mathbf{P}_{\mathsf{U}})^{\perp}$ since

$$|\langle h, k_z\rangle| \quad = \quad \mathbf{E}_{\mathsf{X}}\left[\,\left|\langle \mathbf{E}_{\mathsf{H}|\mathfrak{L}\cup(\mathsf{X},h(\mathsf{X}))}[\mathsf{H}], k_z\rangle\right|\,\right]$$

Using a Universum can be seen as defining a prior $\mathbf{P}_{\mathsf{H}}$ of functions that do not support the domain of the Universum.

is trivially true for any $h \in \mathcal{H}$ and $z \in \mathfrak{U}$. Therefore, ignoring the maximum margin regulariser of a $\mathfrak{U}$-SVM, the solution of a hard margin Universum corresponds to the approximated entropy point for the maximum entropy approach in the version space with a prior that is zero on the support of the distribution of the Universum.

Chapter 3

Applications and Experiments

This chapter is concerned with applications and experimental evaluations of the Universum algorithms. The first two applications, optical character recognition and text classification, are typical machine learning problems and have already been studied in [103, 98]. However, the main part of this chapter is the application of the Universum idea to applications in neuroscience, i.e. *brain computer interfaces* (BCI) and *neural decoding* from the recorded spiking activity of a population of neurons.

3.1 Optical Character Recognition

The MNIST dataset: Classifying patches of digits

Dataset and experimental setting A suitable dataset for the comparison of the Universum implementations $\mathfrak{U}$-SVM and $\mathfrak{U}_{\mathfrak{ls}}$-SVM[1] against their SVM counterparts on an optical digit recognition task is the MNIST dataset [56]. This dataset comprises $60,000$ patches of handwritten digits in a resolution of 28×28 for training and another $10,000$ digits for testing. In the following experiments, only the training part was used. The subproblem of classifying digit five against digit eight was chosen since it belongs to the more difficult learning problems on the MNIST dataset. Therefore, all examples of fives and eights were selected from the training part and were further split into a new training part of $10,000$ examples and a new testing part containing the remaining 1272 examples. From the new training set, ten disjoint sets were randomly sampled, each of size 20, 50, 100, 200, 500 and 1000. On each of the ten datasets of a specific size, a 10-fold cross validation determinded the selection

Cross validation: A technique for model selection

[1]The $\mathfrak{U}_{\mathfrak{c}}$-SVM was implicitly tested on the MNIST dataset for reasons that will become clear later.

of the best parameter values among $C \in \{1, 10, 100\}$, $C_{\mathfrak{U}} \in \{0, 1, 10, 100\}$ and $\varepsilon \in \{0.01, 0.05, 0.1, 0.5\}$ for the regularisation trade-offs C, $C_{\mathfrak{U}}$ and the gap width of the ε-insensitive loss.

Cross validation is a technique for model selection where the dataset is split into n parts. For a specific set of hyperparameters, the learning algorithm is trained on $n-1$ parts and tested on the nth part. Repetition for all n parts yields n test scores. This procedure is carried out for all possible sets of hyperparameters. The one which achieves the best average cross validation error is used for training the algorithm on the complete training data.

Switching off the Universum by setting $C_{\mathfrak{U}} = 0$ was a possible choice during model selection.

All algorithms were offered $C_{\mathfrak{U}} = 0$ as a possible choice during the model selection in order to be able to decide against using a Universum by switching off its influence. If the model selection chooses $C_{\mathfrak{U}} = 0$, the impact of the loss on the Universum points on the optimisation problem is zero. Therefore a $\mathfrak{U}$-SVM with $C_{\mathfrak{U}}$ is equivalent to a normal SVM. This does not hold for the $\mathfrak{U}_{\mathfrak{ls}}$-SVM. However, whenever $C_{\mathfrak{U}} = 0$ was selected for the $\mathfrak{U}_{\mathfrak{ls}}$-SVM, just a simple least squares SVM was trained.

All the experiments were done exclusively with the RBF kernel $k(x, y) = \exp\left\{-\frac{||x-y||^2}{2\sigma^2}\right\}$. Previous experiments showed that the kernel parameter $\sigma = 1167$ yields good results. Therefore, this value was used in all experiments on MNIST. Since $k(z, z) = ||k_z||^2 = 1$, the $\mathfrak{U}_{\mathfrak{c}}$-SVM constraint $|\langle w, z\rangle + b| \leq \rho||z||$ coincides with the constraint $|\langle w, z\rangle + b| \leq \rho$ of the $\mathfrak{U}$-SVM. Therefore, the $\mathfrak{U}$-SVM and the $\mathfrak{U}_{\mathfrak{c}}$-SVM are equivalent in the RBF case and only the results for the $\mathfrak{U}$-SVM are reported.

After performing a cross validation for each parameter tuple on each of the ten splits for a specific size, the best tuple of each split was used to train the algorithm on that split and compute the zero-one loss on the $1, 272$ test examples. By this procedure, ten error scores were obtained for each split size, which were used to compute the standard deviation and perform a T-test on a significance level of 5% in order to check whether the mean error of an SVM (LS-SVM) was significantly different from the mean error of a $\mathfrak{U}$-SVM ($\mathfrak{U}_{\mathfrak{ls}}$-SVM).

Universa Three kinds of Universa were used in the experiment:

Universum $\mathfrak{U}_3$: Example patches of digit three

- The Universa $\mathfrak{U}_3^{(n)}$ consist of examples for the letter "3" from the MNIST training set. Here, n denotes the size of a single Universum. Similar to the assembly of the training set splits, ten disjoint sets of Universa were sampled for each of the Universum sizes of 100, 200 and 500. The motivation for using character "3" as Universum examples is the similarity of digit three to both digits eight and five.In view of this resemblance, it seems reasonable to put the three in between both classes, i.e. the area around the hyperplane. Furthermore, a Universum can confer information about the support of the classes. Specifying a Universum means specifying where the classes should not be supported, because the optimiser will try to push the function down to zero in that area. The classes five and eight clearly should have no support in regions of high density of the other digits. But as a result of using the RBF kernel, the resulting function decays with increasing Euclidean distance to the training examples anyway. Therefore, it is reasonable to merely use the digits that are close in Euclidean distance to five and eight, which is again the class three.

- The Universa $\mathfrak{U}_{gen}^{(n)}$ are motivated by the implementation and literally place the Universum examples between both classes. Since the $\mathfrak{U}$-SVM tries to place the Universum points in the middle, the $\mathfrak{U}_{gen}^{(n)}$ Universa are generated such that its data points are expected to be close to a separating hyperplane. Again, n indicates the number of elements in the respective Universum. Unlike the Universa $\mathfrak{U}_{3}^{(n)}$, the $\mathfrak{U}_{gen}^{(n)}$ is artificially generated by the following procedure: Each single pixel of each single Universum point is assigned the value of a randomly drawn example from the split the Universum point is generated for. Figure 3.1 (left) shows an example of such a Universum point. In this way the pixel distribution of the resulting Universum point reflects the empirical pixel distribution of its training split. This procedure is repeated n times for each training split. It is important to use only the data points in the respective training split to generate the Universum point. Otherwise additional information from test points might be used, which obstructs the statistical validity of the error scores. This means that a Universum $\mathfrak{U}_{gen}^{(n)}$ has to be generated anew for each different training set, even for the training splits of the cross validation.

Universum $\mathfrak{U}_{gen}$: Artificially generated examples that reflect the empirical pixel distribution of the training dataset.

- The Universa $\mathfrak{U}_{mean}^{(n)}$ take the motivation by the implementation to the extreme: To generate an element of $\mathfrak{U}_{mean}^{(n)}$, one element from each class of the respective training set is sampled, and the two digit patches are simply averaged. Figure 3.1 (right) shows an example of such a Universum point.

Universum $\mathfrak{U}_{mean}$: Averaged examples of opposite classes.

Results and Discussion Tables 3.1, 3.2 and 3.3 show the mean zero-one error scores for the Universa $\mathfrak{U}_{3}^{(n)}$, $\mathfrak{U}_{gen}^{(n)}$ and $\mathfrak{U}_{mean}^{(n)}$ averaged over the ten splits. In all experiments the loss decreases with the size of the training set, and the SVM seems to perform equally well as the LS-SVM. Figure 3.2 shows the error scores as a function of training set size for an SVM and the $\mathfrak{U}_{3}^{(n)}$-SVMs. In contrast to the $\mathfrak{U}_{3}^{(n)}$-SVMs and the $\mathfrak{U}_{3}^{(n)}$-LS-SVMs, the average error for the SVMs and LS-SVMs using the artificially generated Universa $\mathfrak{U}_{gen}^{(n)}$ and $\mathfrak{U}_{mean}^{(n)}$ is not better than their SVM counterparts'. In fact, the error scores even seem to get a little worse with increasing Universum size. This means that the Universum seems helpful to the learning algorithm during the model selection

Universum $\mathfrak{U}_3$ augments the classification accuracy significantly.

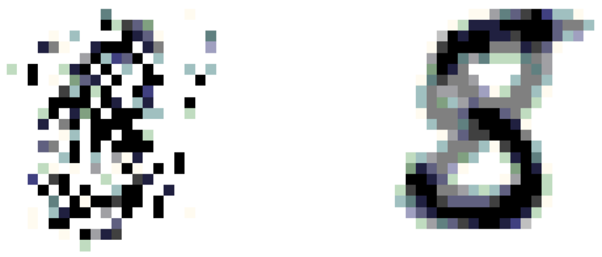


Figure 3.1: Examples of Universum points for the Universa $\mathfrak{U}_{gen}$ (left) and $\mathfrak{U}_{mean}$ (right). The $\mathfrak{U}_{gen}$ example is generated by using pixel values from randomly drawn training examples for each single pixel. The $\mathfrak{U}_{mean}$ example is the mean of two randomly drawn training examples of opposite class.

	SVM	$\mathfrak{U}$-SVM		
$\lvert\mathfrak{L}\rvert\backslash\lvert\mathfrak{U}\rvert$	100	100	200	500
20	29.69 ± 13.61	29.06 ± 14.33	28.58 ± 14.86	28.51 ± 14.93
50	9.49 ± 2.84	8.82 ± 2.09	9.04 ± 2.19	8.88 ± 1.69
100	5.68 ± 0.83	5.42 ± 0.83	5.44 ± 1.02	5.31 ± 0.91
200	3.92 ± 0.53	3.54 ± 0.37	3.49 ± 0.39	$\mathbf{3.28 \pm 0.49}$
500	2.55 ± 0.26	2.32 ± 0.28	2.34 ± 0.20	$\mathbf{2.24 \pm 0.32}$
1000	1.93 ± 0.25	1.82 ± 0.25	1.78 ± 0.26	$\mathbf{1.69 \pm 0.21}$

	LS-SVM	$\mathfrak{U}_{\mathfrak{l}_5}$-SVM		
$\lvert\mathfrak{L}\rvert\backslash\lvert\mathfrak{U}\rvert$	100	100	200	500
20	32.11 ± 14.26	31.53 ± 15.00	30.44 ± 16.22	30.72 ± 15.89
50	9.19 ± 2.57	8.62 ± 2.01	8.13 ± 2.05	8.59 ± 2.03
100	5.61 ± 0.90	5.65 ± 0.93	5.55 ± 0.85	5.13 ± 0.86
200	3.72 ± 0.58	3.57 ± 0.64	3.41 ± 0.52	3.47 ± 0.53
500	2.44 ± 0.26	2.33 ± 0.30	2.30 ± 0.28	0.26 ± 0.33
1000	1.78 ± 0.17	1.68 ± 0.20	$\mathbf{1.56 \pm 0.13}$	1.61 ± 0.27

Table 3.1: Zero-one error scores in percent for five vs. eight classification on the MNIST dataset for SVM, $\mathfrak{U}$-SVM, LS-SVM, $\mathfrak{U}_{\mathfrak{l}_5}$-SVM with Universa $\mathfrak{U}_3^{(n)}$. The error scores are averaged over the ten training splits. Each column has constant Universum size while each row has constant training set size. Results that are significantly (5% significance level) better than the performance of the same algorithm without using a Universum are printed in bold font.

stage, therefore making it choose $C_{\mathfrak{U}} > 0$, but turns out to be harmful for the test error. For SVMs and LS-SVMs using the $\mathfrak{U}_3^{(n)}$ Universa, the performance is not worse in all cases. In some cases it is even significantly better. The error scores for those cases, shown in table 3.1, are printed in bold font. The uselessness of the artificially generated Universa $\mathfrak{U}_{gen}^{(n)}$ and $\mathfrak{U}_{mean}^{(n)}$ and the usefulness of the "natural" Universa $\mathfrak{U}_3^{(n)}$ indicate that it is actually the additional information contained in $\mathfrak{U}_3^{(n)}$ which increases the algorithms' performance. The Universa $\mathfrak{U}_{gen}^{(n)}$ and $\mathfrak{U}_{mean}^{(n)}$ cannot add any additional information to the learning problem since they are generated from the respective training sets. The only way that artificially generating a Universum could prove to be useful for the algorithm's performance is by uncovering non-discriminative features through the process that generates the Universum points. By forcing the normal of the hyperplane learnt by the algorithm to be as orthogonal as possible to those features, the learnt model would become more invariant against them. But this does not seem to be the case here.

Success of $\mathfrak{U}_3$: Invariance against natural transformations?

However, the invariance argument can be used to explain the positive influence of $\mathfrak{U}_3^{(n)}$ on the classification performance. In high dimensional feature spaces, there are many ways to separate two classes of digits. One major problem the learning algorithm faces, is to distinguish discriminative from non-discriminative features. Features induced by frequently occurring transformations on the data, like small rotations, translations, different widths of ink strokes etc., are clearly non-discriminative. With increasing training set size, the learnt function gets more and more invariant against those transformations since they are sufficiently represented in the examples. However, when the training set is small, there might not be enough information to infer the directions of invariance from the data.

Relation of the Universum to the Virtual Support Vector Method

There are two common methods that have been proposed to overcome this problem and have been tested for digit recognition. These methods are related to using a Universum consisting of other digits. In order to make the classifier more robust against transformations from a Lie group $\{\mathcal{L}_t\}$, the data can be whitened with respect to the covariance matrix of the tangent vectors $\mathbf{C} = Cov\left(\{\frac{\partial}{\partial t}|_{t=0}\mathcal{L}_t\mathbf{x}_i\}_{i=1,\ldots,m}\right)$ to the transformation group at the data points [19, 12, 13]. This amounts to rescaling the data points by the inverse square root of the eigenvalues of $\mathbf{C}$ in directions of their eigenvectors, thereby decreasing the directions with large eigenvalues. Since the large eigenvalues of $\mathbf{C}$ are associated with large variance of the transformation in that direction, the influence of the transformation on the dot product is effectively decreased.

Another method which applies to more general groups of transformations, is the use of additional examples that have been artificially transformed by the transformations in question [88]. Although this injects the relevant information about non-discriminative transformations into the dataset, its possibly large size can slow down the learning process. A solution introduced by [79, 19], which is especially suited to SVMs, is the so called *virtual support vector* method. Here, an SVM is first trained on the original dataset, yielding a set of support vectors. Afterwards, the transformations [88] one wishes to be invariant against are applied to the support vectors only, thereby producing new data points. Then, the SVM is retrained on this new dataset. The virtual support vector method uses the fact that an SVM trained on the support vectors yields the same solution as when trained on the whole dataset. Therefore it is sufficient to only transform the support vectors in order to introduce the necessary information about the transformations.

Specifying implicit invariance directions with a Universum

One aspect both methods have in common, is that the transformation has to be known to make the classifier more robust against it. This might be possible for obvious transformations which are easy to parameterise, but impossible for more complex ones. In this case, a Universum might help to specify the directions of invariance.

A linear function is most invariant against a transformation if the transformation only changes the data point along the null space of the function, because such a transformation will not change the value of the function. This means that the normal vector that represents the linear function has to be orthogonal to the range of the transformation. This is also the idea behind the two methods described above. If, now, the Universum points are afflicted with the same transformations as the training points and those transformations constitute a major part of the variance, then putting the Universum points close to the hyperplane will make the normal vector of the hyperplane more orthogonal to the directions of major variance and therefore more invariant against the transformations. In the case of digit classification, it is not unreasonable to assume that the transformations on all digits are the same. An additional set of digits can therefore provide information about the directions of desired invariance. This could explain the better performance of the $\mathfrak{U}$-SVMs and $\mathfrak{U}_{\mathfrak{ls}}$-SVMs using $\mathfrak{U}_3^{(n)}$.

3.2 Text classification

The Reuters dataset: New articles of different categories

Dataset and experimental setting Another common domain for machine learning algorithms is text classification. In order to test the different Universum implementations on that domain, the Reuters dataset was chosen which was already used in [103]. The Reuters dataset consists of more than 800,000 English news articles from the Reuters news agency written between August 20, 1996 and August 19, 1997. In the following experiments the version rcv1-v2 of [58] was used. The Reuters dataset is hierarchically organised. Starting from four top level categories, ECAT (Economics), GCAT (Government/ Social), MCAT

	SVM	$\mathfrak{U}$-SVM		
$\lvert\mathfrak{L}\rvert \backslash \lvert\mathfrak{U}\rvert$	0	100	200	500
20	29.69 ± 13.61	29.64 ± 13.67	29.55 ± 13.78	29.25 ± 14.17
50	9.49 ± 2.84	8.56 ± 1.91	8.81 ± 1.98	8.88 ± 2.07
100	5.68 ± 0.83	5.77 ± 0.91	5.77 ± 0.91	5.72 ± 0.81
200	3.92 ± 0.53	3.96 ± 0.58	4.00 ± 0.59	4.14 ± 0.51
500	2.55 ± 0.26	2.80 ± 0.29	2.74 ± 0.33	2.84 ± 0.41
1000	1.93 ± 0.25	1.99 ± 0.24	2.18 ± 0.23	2.03 ± 0.35
	LS-SVM	$\mathfrak{U}_{\mathfrak{l}_5}$-SVM		
$\lvert\mathfrak{L}\rvert \backslash \lvert\mathfrak{U}\rvert$	0	100	200	500
20	32.11 ± 14.26	32.09 ± 14.29	32.13 ± 14.24	32.16 ± 14.19
50	9.19 ± 2.57	8.90 ± 2.38	9.04 ± 2.42	9.01 ± 2.28
100	5.61 ± 0.90	5.79 ± 0.95	5.80 ± 0.99	5.87 ± 0.96
200	3.72 ± 0.58	3.76 ± 0.60	3.79 ± 0.65	3.79 ± 0.68
500	2.44 ± 0.26	2.54 ± 0.36	2.61 ± 0.35	2.61 ± 0.34
1000	1.78 ± 0.17	1.74 ± 0.15	1.81 ± 0.19	1.81 ± 0.22

Table 3.2: Zero-one error scores in percent for five vs. eight classification on the MNIST dataset for SVM, $\mathfrak{U}$-SVM, LS-SVM, $\mathfrak{U}_{\mathfrak{l}_5}$-SVM with Universa $\mathfrak{U}_{gen}^{(n)}$. The error scores are averaged over the ten training splits. Each column has a constant Universum size while each row has constant training set size.

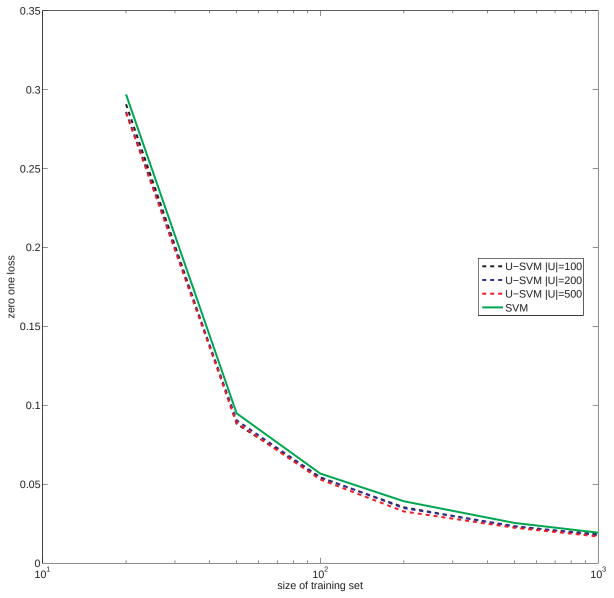


Figure 3.2: Zero-one loss as a function of training set size for five vs. eight classification on the MNIST dataset using Universa $\mathfrak{U}_3^{(n)}$.

	SVM	$\mathfrak{U}$-SVM		
$\lvert\mathfrak{L}\rvert\backslash\lvert\mathfrak{U}\rvert$	0	100	200	500
20	29.69 ± 13.61	29.71 ± 13.59	30.53 ± 12.68	29.52 ± 13.83
50	9.49 ± 2.84	10.39 ± 2.98	11.72 ± 4.94	10.54 ± 3.05
100	5.68 ± 0.83	6.77 ± 2.17	6.55 ± 1.49	$7.43. \pm 3.41$
200	3.92 ± 0.53	4.03 ± 0.64	4.57 ± 1.57	4.13 ± 0.77
500	2.55 ± 0.26	2.72 ± 0.71	2.68 ± 0.44	2.55 ± 0.26
1000	1.93 ± 0.25	1.93 ± 0.25	1.93 ± 0.25	1.93 ± 0.25
	LS-SVM	$\mathfrak{U}_{\mathfrak{ls}}$-SVM		
$\lvert\mathfrak{L}\rvert\backslash\lvert\mathfrak{U}\rvert$	0	100	200	500
20	32.11 ± 14.26	32.72 ± 13.69	32.09 ± 14.29	32.07 ± 14.36
50	9.19 ± 2.57	10.21 ± 3.51	9.80 ± 2.83	9.80 ± 2.74
100	5.61 ± 0.90	5.93 ± 1.86	5.98 ± 1.60	5.97 ± 1.52
200	3.72 ± 0.58	3.77 ± 0.62	4.72 ± 2.27	3.72 ± 0.58
500	2.44 ± 0.26	2.51 ± 0.27	2.44 ± 0.26	2.44 ± 0.26
1000	1.78 ± 0.17	1.78 ± 0.17	1.78 ± 0.17	1.78 ± 0.17

Table 3.3: Zero-one error scores in percent for five vs. eight classification on the MNIST dataset for SVM, $\mathfrak{U}$-SVM, LS-SVM, $\mathfrak{U}_{\mathfrak{ls}}$-SVM with Universa $\mathfrak{U}^{(n)}_{mean}$. The error scores are averaged over the ten training splits. Each column has a constant Universum size while each row has constant training set size.

(Markets) and CCAT (Corporate/ Industrial), the articles are sorted into more specific categories which themselves are split into sub-categories. This generates a tree of different thematic categories. If an article could not be assigned to one of the finer categories, it was assigned to the finest category possible. Therefore, not only the leaves of the tree contain articles, but also intermediate nodes.

In the experiments the task was to separate the category C15 (Performance), which is a sub-category of CCAT, against all other sub-categories of CCAT. This led to a dataset of 13,310 examples (4,178 positive examples and 9,130 negative examples). Each example is represented as a so called *Bag of Words*, that is a histogram of word occurrences in the respective document for a dictionary of 47,237 words. All histogram entries have been transformed by the so called TF.IDF weighting scheme: In this scheme, the entry for the term t in document d is given by $x_d(t) = (1 + \log_e n(t,d)) \cdot \log_e \left(\frac{|\mathcal{D}|}{n(t)}\right)$, where $n(t,d)$ is the number of occurrences of t in d, $n(t)$ is number of documents in the whole corpus that contain t, and $|\mathcal{D}|$ is the total number of documents in the corpus. The values of $n(t)$ have been computed on the entire Reuters dataset, in particular also on the test data and on other data points used for Universa (see below). This use of the data provides ad-

Bag of Words TF.IDF weighting: Rescaled histograms of word occurrences ditional information, but does not obstruct the statistical validity of the error scores since only the documents and not the labels of the documents were used. After the transformation, the Bags of Words were scaled to unit length.

As in the MNIST experiments, $10,000$ examples were used for training and model selection, while the remaining 3310 were used for testing. Again, ten disjoint splits for each of the training set sizes 20, 50, 100, 200, 500 and 1000 were sampled from the training set. The model selection for choosing the hyperparameters C, $C_{\mathfrak{U}}$, ε and the RBF kernel parameter σ was done analogously to that for the MNIST dataset. Possible values for σ were the $2^{-3}, 2^{-2}, ..., 2^{3}$ multiples of the $\frac{1}{2}$-quantile of the pairwise input space distances between the documents of the first split of 1000 examples. This heuristic for choosing the correct range of σ was suggested in [15]. Again, since this procedure does not use the label information of the data points, these values can also be used for splits of smaller size and are a statistically valid use of unlabelled data.

The testing procedure was the same as in the MNIST case.

Universa Two different Universa were used for the Reuters dataset: a natural and an artificial one.

Universum $\mathfrak{U}_{M14}$: Examples from the M14 category

- The natural Universum $\mathfrak{U}_{M14}$ consists of all 2540 examples from the category M14 (Commodity markets), which is a sub-category of MCAT. In contrast to the experiments with the MNIST dataset, only one Universum size was used. This Universum is again motivated by the invariance argument. Since the subject of the M14 examples should be quite different to the subject of the CCAT class, variations in the principle directions of M14 should not change the outcome of the classifier on the CCAT examples.

Universum $\mathfrak{U}_{MoC}$: Mean of the closest of ten randomly drawn examples from opposite classes

- The artificial Universum $\mathfrak{U}_{MoC}$ was generated similar to the $\mathfrak{U}_{mean}$ Universum for digit recognition. In order to ensure that the generated examples lie in between the two labelled classes, ten examples were randomly drawn from each class, and the mean of the two closest points in input space was used as a Universum example. Since this procedure is likely to produce a lot of double Universum examples, especially for small labelled sets, the Universum size was set to the size of the respective training set. This has the additional advantage of keeping the relative influence of the Universum in the optimisation's objective constant.

$\mathfrak{U}_{M14}$ and $\mathfrak{U}_{MoC}$ improve the classification performance; $\mathfrak{U}_{M14}$ significantly, so.

Results and Discussion Tables 3.4 and 3.5 show the averaged zero-one error scores. All $\mathfrak{U}$-SVMs and $\mathfrak{U}_{ls}$-SVMs using Universum $\mathfrak{U}_{M14}$ perform better for the training set sizes 20 and 50, some of them significantly (T-test with significance level 5%). For smaller and larger dataset sizes, the effect vanished. None of the Universum algorithms that use the $\mathfrak{U}_{MoC}$ Universum performs significantly better. However,

	SVM	$\mathfrak{U}$-SVM	LS-SVM	$\mathfrak{U}_{\mathfrak{ls}}$-SVM
20	36.09 ± 11.58	35.54 ± 11.54	34.37 ± 12.31	35.72 ± 11.52
50	23.15 ± 6.40	$\mathbf{15.97 \pm 1.83}$	23.97 ± 6.44	$\mathbf{16.08 \pm 1.79}$
100	14.28 ± 2.26	13.03 ± 0.85	15.65 ± 3.14	$\mathbf{13.05 \pm 1.06}$
200	10.80 ± 1.22	11.33 ± 1.49	11.31 ± 1.39	11.53 ± 0.94
500	9.09 ± 0.63	9.18 ± 0.61	9.17 ± 0.53	9.32 ± 0.44
1000	7.97 ± 0.48	8.01 ± 0.46	8.16 ± 0.52	8.09 ± 0.56

Table 3.4: Zero-one prediction error for the $\mathfrak{U}_{M14}$ Universum averaged over ten splits. Significantly better results are printed in bold font.

there seems to be a slightly better performance again for the dataset sizes 20 and 50. Figure 3.3 shows the dependency between the amount of labelled data and the error scores for the SVM and the $\mathfrak{U}$-SVM with $\mathfrak{U}_{M14}$ and $\mathfrak{U}_{MoC}$.

Reasons for the improvement with $\mathfrak{U}_{M14}$

The reason why the Universum only improves the performance for intermediate dataset sizes might be that the Universum data is misleading for smaller dataset sizes and does not add any further necessary information for large dataset sizes. In the latter case, the algorithm apparently has enough information in the training set to classify the labelled data correctly. For the intermediate sizes of 20 and 50 the $\mathfrak{U}_{M14}$, but also the $\mathfrak{U}_{MoC}$ Universum, improve the performance. The explanation for the success of the $\mathfrak{U}_{M14}$ Universum could be that its Bags of Words contain a lot of words that are not discriminative for the classification task on the labelled data. By forcing those examples to zero, the $\mathfrak{U}$-SVM is forced to ignore those non-discriminative features.

Reasons for the improvement with $\mathfrak{U}_{MoC}$

The slightly better classification accuracy for the $\mathfrak{U}_{MoC}$ Universum compared to SVMs yields some evidence against the hypothesis that the additional information in the Universum set increases the algorithm's performance. In this case, however, it might be that the operation on the labelled examples in the input space uncovered some non-discriminative features. Under certain circumstances, averaging two examples of opposite classes could carve out the non-discriminative features. Using the averaged examples as Universum points could then diminish the relative importance of those features.

The results encourage to use Universum algorithms in text classification, which is well suited for the Universum setup at the same time. While labelling texts can be a tedious procedure, unlabelled texts from totally different subjects are plentiful.

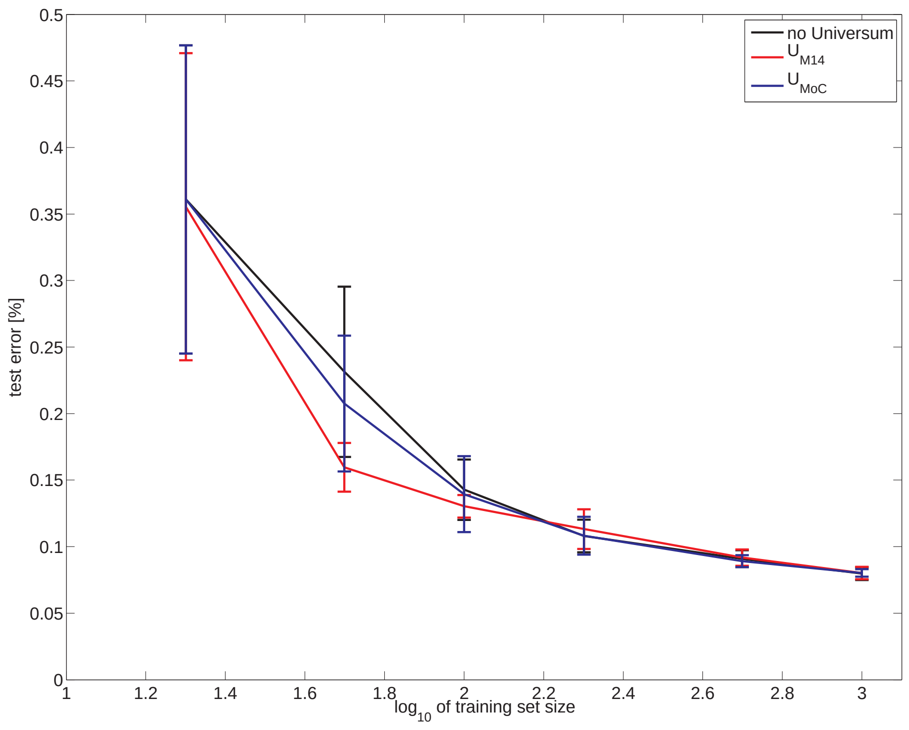


Figure 3.3: Dependency between the size of the training set and the zero-one error scores for an SVM and $\mathfrak{U}$-SVMs using Universum $\mathfrak{U}_{M14}$ and $\mathfrak{U}_{MoC}$. The training set size is plotted on a log scale.

	SVM	$\mathfrak{U}$-SVM	LS-SVM	$\mathfrak{U}_{\mathfrak{ls}}$-SVM
20	36.09 ± 11.58	36.09 ± 11.58	34.37 ± 12.31	34.21 ± 12.50
50	23.15 ± 6.40	20.76 ± 5.10	23.97 ± 6.44	23.60 ± 6.55
100	14.28 ± 2.26	13.95 ± 2.85	15.65 ± 3.14	15.22 ± 2.74
200	10.80 ± 1.22	10.82 ± 1.42	11.31 ± 1.39	10.86 ± 1.33
500	9.09 ± 0.63	8.91 ± 0.45	9.17 ± 0.53	9.51 ± 0.72
1000	7.97 ± 0.48	8.02 ± 0.28	8.16 ± 0.52	8.09 ± 0.64

Table 3.5: Zero-one prediction error for the $\mathfrak{U}_{MoC}$ Universum averaged over ten splits. No significantly better results were found.

3.3 Universum for Neural Applications

3.3.1 Brain Computer Interfaces (BCI)

A brain computer interface is a mapping from brain activity, recorded with EEG, MEG, ECoG, SCR, NIR or fMRI, into $\mathbb{R}$.

Brain computer interfaces (BCI) are devices that allow a user to control a computer by merely using his brain activity [52]. The user's brain activity is captured by electroencephalogaphic (EEG), magnetoencephalographic (MEG), eletrocorticographic (ECoG), or single cell recordings (SCR) [94], or near-infrared (NIR) or functional magnetic resonance imaging (fMRI) and interpreted by a computer system. The interpretation of the computer can aim at different states, e.g. "yes" or "no", or at a continuous signal, e.g. for controlling a cursor. Therefore, a BCI implements a mapping from brain activity into either a discrete set of states or the real numbers. This mapping can either be fixed a priori or involve a learning algorithm. When using a fixed mapping, the user has to learn how to control the BCI via his brain activity. This can involve tedious training since intentionally controlling ones own brain activity is clearly not a natural task for a human. In fact, not every person can learn how to control his brain activity. This is the main motivation for using machine learning algorithms for BCIs. In a training phase, the user is asked to accomplish a certain task, while his brain activity is recorded. This yields a set of labelled signals which is then used to train a supervised learning algorithm. However, the mapping from brain signals to control signals seems to change over time. Therefore, a learnt mapping has only a certain lifetime. It is common practice to use a learning algorithm in a starting phase in order to achieve a good initial performance of the user and then fix the learnt mapping after some time and put the need for adaption on the user's side.

BCIs allow locked-in patients to communicate.

The main motivation for BCI research is to develop devices that allow completely paralysed patients to communicate. Examples for diseases that can lead to complete paralysis are a stroke or amyotrophic lateral sclerosis, which involves the degeneration of motor neurons in the cortex and the spine. If there are no other physical means of communication for a patient, i.e. the patient is even unable to move the eyes or eyelids, then his situation is referred to as *locked-in syndrome*. Designing affordable, comfortably controllable BCIs with a high information rate for locked-in patients is the ultimate goal of BCI research.

Design decisions in a BCI: neurological events, recording technique, output space and type of mapping

There are several components of a BCI that can be varied:

- the type of neurological events to be detected to control the computer system (*motor related potentials*, *slow cortical potentials*, ...)
- the recording technique (EEG, MEG, NIR, ...)
- the outputs (discrete, continuous)
- the mapping from the signal to the output space to be used (a priori fixed or learned mapping).

In some cases the decision for one component implies the decision for another, e.g. the decision for a neurological event discloses all the imaging techniques, that cannot detect that event. The experiments below involve data from *EEG* recordings of *slow motor related potentials* and *event-related desynchronisation (ERD)* as well as data from tactile stimulation using *MEG.*

Events used in this thesis: Slow motor related potentials, event-related desynchronisation and tactile evoked potentials

Extracranial encephalography is the oldest technique to non-invasively measure cerebral activity. It was invented by Hans Berger at the University of Jena in Germany in 1929 and was first published in 1938 [3]. In an EEG recording, up to a few hundred electrodes are placed on the subject's head, to record changes of voltage potential through the skull and the meninges. The recorded changes of voltage potential are mainly due to the physiological activity of the cortex. In contrast to other imaging techniques such as fMRI, EEG has a very good temporal resolution in the order of tens of milliseconds. However, due to the isolation properties of the skull and the tissue between the brain and the electrodes, the spatial resolution is limited.

Extracranial encephalography: EEG

Magnetoencephalography is a non-invasive recording technique that measures the magnetic field induced by the electrical activity of cerebral neurons. Although it is possible to record data from sub-cortical structures, MEG mainly captures data from the cortex. MEG can measure magnetic field fluctuations down to the order of 10^{-14} Tesla. Due to this sensitivity, MEG scanners need to be installed in a magentically shielded room to avoid disturbances from the earth's magnetic field or other objects in the near environment. Furthermore, the sensors must be cooled, usually by a large liquid helium cooling unit. This eliminates the MEG technology from the list of recording techniques for a BCI that can be used in a patients daily life. However, it is argued [40] that MEG is applicable for preliminary studies to search for stable signals that can be used to build a BCI for a certain patient.

Magnetoencephalography: MEG

There are two fundamental types of stimuli used in BCIs: In *endogenous BCIs,* certain mental tasks like mentally rotating an object or imagining movements, are performed by the subject to indicate a certain state. In *exogenous BCIs* external stimuli are presented to the patient who indicates the intended state by focusing his attention on one or the other stimulus. The experiments presented below involve both types. The datasets MpiMov and BciComp consist of movement-related potentials and event-related desynchronisation from imagined movements recorded with an EEG and therefore belong to the domain of endogenous BCIs. The FINGERS dataset contains MEG data recorded from a exogenous BCI task using tactile stimulation of the fingers.

Endogenous versus exogenous BCIs

Data The machine learning experiments below involve data from three experiments:

- The MpiMov dataset consists of EEG recordings from an imagined movement BCI paradigm. The experiment was conducted at the Max-Planck-Institute in Tübingen to examine the possibility of recording Universum data from a baseline condition. The three tasks for the subjects were: imagining a movement of the left hand, imagining a movement of the right hand, not imagining a movement. The exact experimental setup is described below.

MPIMOV*: Motor-related potentials and event-related desynchronisation recorded with EEG at the Max-Planck-Institute in Tübingen*

Imagined movement tasks use a well defined signal for controlling the BCI: When a person is neither moving nor about to move, the motor cortex produces an electrical activity which is dominated by rhythms in the 8-12Hz (α-band) and 18-22Hz (β-band) frequency bands. Desynchronisation in the α-band is also known to correlate with visual tasks [51]. This will become important later when designing another Universum for this dataset. This activity is called *μ-activity* or *μ-rhythm*. At the beginning of the planning phase of a movement, about 1-1.5 seconds before the movement is executed, the μ-rhythm gradually disappears. This event is referred to as *event-related desynchronisation (ERD)* and can be detected by EEG. *Movement-related potentials (MRP)* are slow shifts in the electrical activity which also start 1-1.5 seconds before the execution of a movement and become dominating on the contralateral site a few hundred milliseconds before the movement. Naturally, the best location to record this signal is the area over the motor cortices. It is known that the MRP and ERD can also be recorded when the movement is only imagined [67] or attempted [47]. Unfortunately, not all persons show a motor-related ERD. However, if an ERD is present, it can be detected more or less reliably and is therefore used in many BCI studies (e.g. [34, 52, 54, 53, 57, 4, 104, 30, 37]). Another advantage of using ERD and MRP is the possibility to train subjects to control their μ-activity and thereby improving the classification accuracy [36]. One potential drawback of this BCI paradigm is the decreasing quality of the signal due the long immobility of a real patient leading to the degeneration of the pyramidal cells in the motor cortex.

BCICOMP: *EEG recordings of motor-related potentials and event-related desynchronisation from IDIAP (Switzerland)*

- The experiments for the BCICOMP dataset employed the same basic experimental paradigm as the experiments for the MPIMOV dataset. The data was initially recorded by Jose del R. Millan at IDIAP in Switzerland [63] and provided online as "Data set V <mental imagery, multi-class>" for the third BCI competition hosted by the Fraunhofer Institute FIRST in Berlin. The experiments involved three tasks: imagination of repetitive self-paced left hand movements, imagination of repetitive self-paced right hand movements and generation of words beginning with the same random letter. The experiments below used only the first two conditions as labelled data, while the third class served as a Universum in one experiment. The description of the experimental setup can be found online [2].

- The FINGERS dataset was recorded by Jeremy Hill in a study at the MEG scanner in Tübingen for an attention-based exogenous BCI using tactile stimuli [74]:
  In another attention related BCI paradigm, [41] employ a new approach based on *auditory-evoked potentials (AEP).* In the experiments they presemt two constantly lasting auditory stimuli to the subject, who was asked to indicate the desired state by focusing his attention on one or the other. The data from those experiments proved to be well classifiable after only two hours of recording. Stimuli that are neither motor related nor visual, circumvent the problems of a decaying signal due to complete immobility of the patient and the resulting degeneration of motor neurons or immobility of the eyes.
  [74] investigates a variation of that paradigm, again based on the modulation of attention to constant streams of stimulation by replacing the auditory by a tactile stimulus stream. One challenge in BCI is to increase the possible bit rate of a BCI. This can either be done by reducing the length of the time window that is used to classify the signal from the patient, or by increasing the number of possible states. The number of possible stimuli streams of the auditory approach is clearly limited since concentrating on the different streams becomes harder as the number of different stimuli increases. Another sensory modality that is not likely to suffer from degeneration and allows for several streams of stimuli is tactile stimulation. For the FINGERS dataset, Hill used four different streams of tactile stimuli at the thumb and little finger on each of the subject's hands. The subject indicates the desired class by focusing his attention to the stimulus at a specific finger. Not focusing the attention on any finger was also used as a condition. The FINGERS dataset therefore represents a five class problem.
  The exact experimental setup is described in [74].

FINGERS: Tactile evoked potentials recorded at the MEG center in Tübingen

Universa Depending on the dataset different kinds of Universa were used:

- As in the experiments for the MNIST and the Reuters dataset, Universa $\mathfrak{U}_{mean}$ of averaged examples are used for the experiments on the MPIMOV and FINGERS datasets. 200 Universum points were generated in exactly the same manner as for the MNIST dataset in section 3.1. Hence, the number of Universum points was roughly equal to the number of training points.

$\mathfrak{U}_{mean}$ Universum: Averaged examples

[2] http://ida.first.fraunhofer.de/projects/bci/competition_iii/desc_V.html

- The Universa $\mathfrak{U}_{C3}$ correspond to data from a third condition in the EEG experiments. This third condition was different for the MPIMOV and the BCICOMP dataset.

  $\mathfrak{U}_{C3}$ Universum: Examples from a third condition

  The MPIMOV dataset was exclusively recorded in order to check whether Universum points recorded in a baseline condition can augment the classification accuracy of the learning algorithm. In this third condition the subject was asked to neither imagine a movement nor to move. The motivation for this kind of Universum is that data recorded in this condition should not be discriminative for the imagined movement task. By using a $\mathfrak{U}_{C3}$ Universum, the classifier should become more robust against non-discriminative patterns in the data.

  $\mathfrak{U}_{C3}$ for MPIMOV

  The $\mathfrak{U}_{C3}$ Universum for the BCICOMP dataset consists of the data points of the third condition in the experiments by [63]. In this condition, the subject had to generate words starting with the same random letter. Again, the idea behind using data from this condition is that data from a task that is not related to the movement of the hands should yield non-discriminative features the classification algorithm should be invariant against.

  $\mathfrak{U}_{C3}$ for BCICOMP

  The Universa $\overline{\mathfrak{U}_{C3}}$ denote the $\mathfrak{U}_{C3}$ Universa that have been centered onto the point between the means of the two other classes. As argued in 2.4.2, there is theoretical evidence that the covariance structure of the Universum data contains useful information for the Universum classifier. There, the covariance matrix of the Universum serves as noise covariance. The noise mean was exactly the mean of the class means. By centring the Universum to that point, the noise as specified by the Universum has mean zero.

  $\overline{\mathfrak{U}_{C3}}$ Universa are centered on the average of the class means

  There is also a more intuitive way to motivate this centring procedure: If only the covariance structure is important, a small difference between the mean of the Universum points and the point between the two classes is desired. If the majority of Universum points lie far apart from the labelled data, they will pull the hyperplane towards their center. This might impair the classification performance of the learning algorithm. Especially linear SVMs suffer from such kind of Universum. In the RBF case, Universa that lie far apart do not hurt the classification performance since the RBF function drops down to zero with increasing Euclidean distance of the data points anyway. In this case, the distance in feature space converges to $\sqrt{2}$ and thus has only constant influence. Since RBF kernels do not offer any advantage over linear SVMs for BCI data [40], linear SVMs are used throughout the experiments below and centring the Universum on the training data might be helpful.

- The Universa $\mathfrak{U}_{nm}$ used for the experiment with MPIMOV and BCI-

COMP are motivated by physiological considerations. As described above, event-related desynchronisation (ERD) represents a decline in the frequency α-band. However, desynchronisation in the α-band is also related to vision [51]. Since the ERD is due to imagined movement, the main source of information should be approximately over the motor cortices which are located in the region around the central sulcus. Information coming from the vision areas which are located at the caudal area of the brain should not be useful and therefore not be considered by the classification algorithm.

$\overline{\mathfrak{U}_{nm}}$ Universa: Independent components with a strong activity over the visual cortex

In almost all BCI applications, heavy preprocessing proves to be crucial for classification performance. One preprocessing step is to run an *independent component analysis (ICA)* on the EEG data. ICA estimates a set of independent components $\{\mathbf{s}_1, ..., \mathbf{s}_n\}$ from a set of n mixed sources $\{\mathbf{x}_1, ..., \mathbf{x}_n\}$, i.e. EEG channels for BCI. When training the learning algorithm, the projections onto the independent components are used. [40] use a method that displays the relative influence of each EEG channel on the single independent components, i.e. the training features: If $\mathbf{X}$ denotes a matrix with the recordings form the different channels as row vectors and $\mathbf{S}$ a matrix with independent components as row vectors, then the relation between the data points and the independent components is given by $\mathbf{X} = \mathbf{WS}$, where $\mathbf{W}$ is a $s \times s$ matrix called *the mixing matrix*. If $\mathbf{W}$ has full rank, the relation can also be written as $\mathbf{W}^{-1}\mathbf{X} = \mathbf{S}$. This matrix product can also be written as a sum of outer products between the data points and the columns of the matrix $\mathbf{W}^{-1}$. Therefore, the single entries of each row give a measure for the relative influence of this EEG-channel on the respective independent component. By plotting the columns of $\mathbf{W}^{-1}$, the influence of the single channels can be visualised. Figure 3.4 shows this visualisation for the first subject of the BCI-COMP dataset.

The elements of the $\mathfrak{U}_{nm}$ Universa can now be generated by the following procedure. Independent components that have a strong influence from channels over the motor cortices are selected and the corresponding features in the projected training data are set to zero. By this procedure, the Universum points live in the subspace spanned by the independent components that are likely not to be relevant for the discrimination task. Since training an SVM with a Universum roughly corresponds to removing the Universum data from the space of possible solutions (see section 2.4), using the $\mathfrak{U}_{nm}$ Universum tells the SVM to avoid solutions from the subspace of non-discriminative independent components. Compared to just ignoring the presumably non-discriminative independent components, the learning algorithm can actually decide to use those dimensions. However, it is discouraged to do so.

Since the independent components are also only estimated from data, using this Universum could account for estimation errors made by the ICA.

- As mentioned above, the FINGERS dataset has 5 different classes of data points. The different approaches for multiclass classification with binary classifiers have already been described in 2.2.1.3. Depending on the coding scheme for multiclass classification, there are classes that are not used as labelled data in a specific subproblem. In this case, it seems natural to use this data as Universum instead of simply ignoring it since this should increase the contrast in the outputs of the single classifiers and therefore make the output code words easier to decode. This was done for the experiments on the FINGERS dataset.

Universa for FINGERS: All classes which are currently not used in a subproblem for multiclass classification

Experimental setup for MPIMOV In the experiments for the MPI EEG data, data from three male healthy subjects (referred to as JH, JL and FS in the sequel) was recorded using 40 silver chloride electrodes (see figure 3.5). The overall setup of the experiment followed [52]. The reference electrodes were placed at the positions TP9 and TP10 as shown in figure 3.5. The electrodes FP2 and EO2 were used to record possible artifacts from eye movements or eye blinks. The electrode H2 was attached to the subject's hand in order to monitor whether the subject was unintentionally moving his hand. Before sampling the data at 256Hz, an analog bandpass filter with cutoff frequencies 0.1Hz and 40Hz was applied.

The subjects were seated in an armchair at 1m distance in front of a computer screen which showed cues that indicated the three states "left hand movement", "right hand movement" and "no movement". One experiment session was divided into several runs, each of which covered 60 trials. After each run there was a pause of about five to fifteen minutes. The total length of each trial was 9 seconds. Between the single trials inter-trial intervals with a random length between 2-4 seconds were added in order to give the subject time to relax.

Top-down dataset organisation: session, run, trial

The 9 seconds of a single trial encompassed different phases: In a starting phase of 2 seconds, only the blank screen was shown to the subject. At second 2, a fixation cross was displayed in the center of the screen and an auditory stimulus indicated the start of a new trial. The subject was instructed to fixate the cross for the complete time interval for which it was visible. At second 3, a cue was displayed for 0.5 seconds. The cue could be either an arrow pointing to the side, asking the subject to imagine the hand movement, or a box indicating that the subject should not imagine a movement. The subject performed the task for that trial until the fixation cross disappeared at second 9. Figure 3.6 schematically displays the time course of a trial.

Organisation of a single trial: blank screen, fixation cross onset, auditory signal, cue onset, cue offset and random pause

No outlier detection was performed during the experiments and also no data point was removed after it.

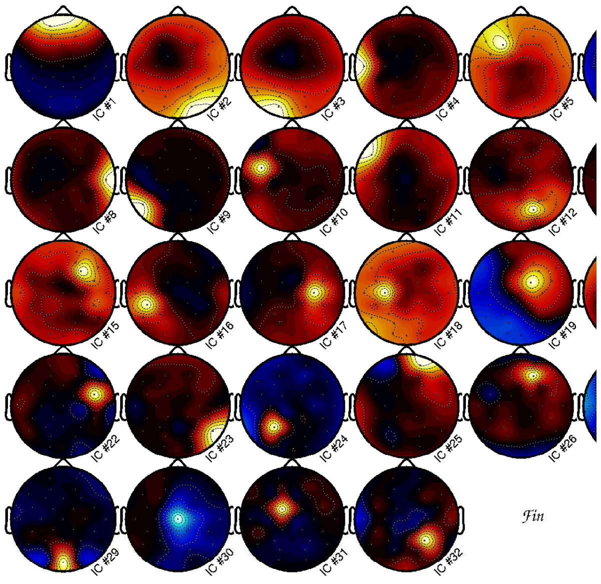


Figure 3.4: Visualsation of the relative influence of the different BCI channels on the different independent components. Bright colors indicate a strong positive (yellow) or a strong negative (blue) influence. The plots display the values of the different columns of the inverse mixing matrix obtained by ICA.

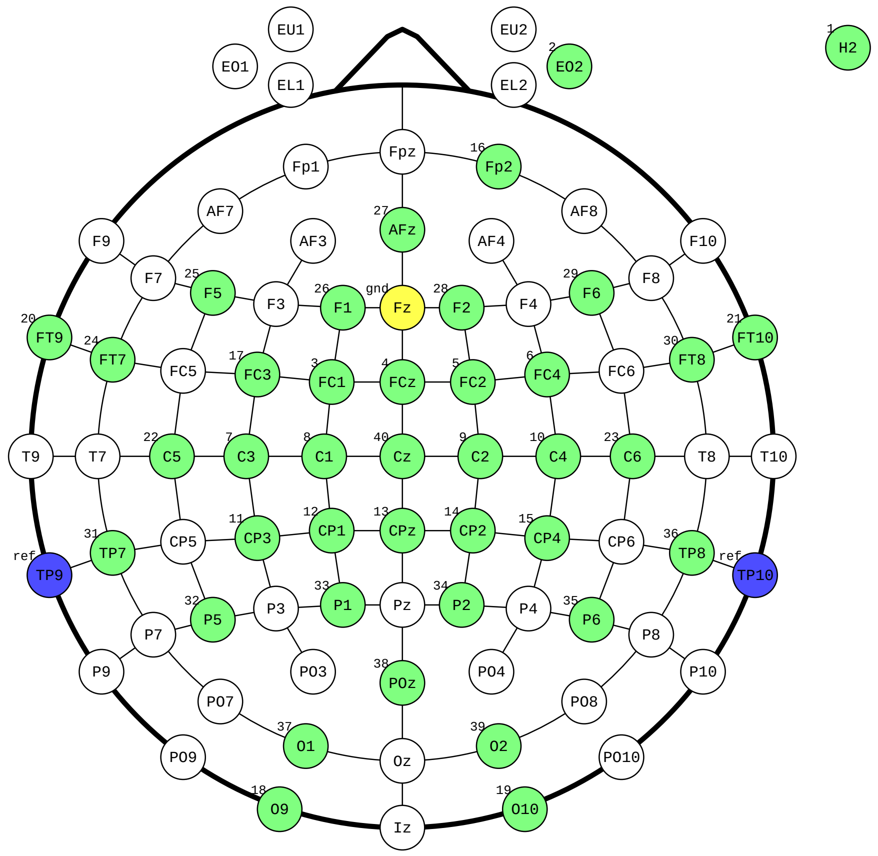


Figure 3.5: Positions of the electrodes for the MPIMOV experiment. Recording electrodes are painted green, reference electrodes blue. The yellow circle marks the position of the ground electrode.

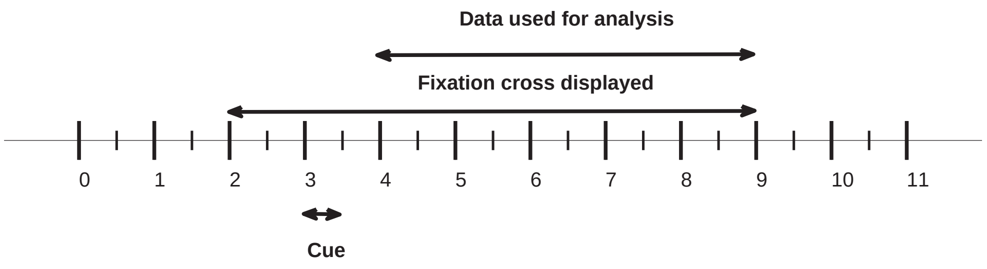


Figure 3.6: Time course of a single trial for the MpiMov experiment. Each trial lasted 9 seconds. Starting from second 0, a fixation cross was displayed between second 2 to 9. At second 3 the cue which indicated the subject's task, was shown for 0.5 seconds. Between second 9 and the start of a new trial, a period of random length between 2 and 4 seconds was added for relaxation.

Data preprocessing

Preprocessing for EEG data: extraction of a time window, low pass filtering, linear detrending, spatial filtering with ICA and power spectral density computation

For the MpiMov recordings, a time window of 500ms after cue offset was extracted for each trial in order to avoid cue related signals in further processing stages. The time series of each sensor and each trial was low-pass-filtered by a zero-phase-distortion method with a smooth falloff between 45 and 50 Hz and downsampled to 100Hz. Afterwards, the data was linearly detrended, i.e. a linear model was fitted to the data and subtracted from it. Since the signals from the cortex get blurred by the layers of bone and tissue between the brain and the electrodes, a spatial filter was applied to the data.

A spatial filter is a matrix $\mathbf{W} \in \mathbb{R}^{r \times s}$ that maps s time series of length t to r time series of length t. If $\mathbf{X} \in \mathbb{R}^{s \times t}$ denotes the input patterns, spatially filtering corresponds to a premultiplication of $\mathbf{W}$. In order to estimate $\mathbf{W}$, an independent component analysis (ICA) was performed on the concatenated time series of m trials for s sensors. ICA is an algorithm that estimates $\mathbf{W}$ by maximising the statistical independence of the resulting r outputs measured to some criterion. For the experiments in this thesis, an infomax ICA [2], as implemented in [20], was used. Infomax ICA maximises the mutual information [18] between the inputs $\mathbf{X}$ and the outputs $\mathbf{WX}$ with respect to $\mathbf{W}$. Due to the high computational cost of Infomax ICA, $\mathbf{W}$ was estimated on the entire dataset and not separately on each of the respective training splits. However, as in the cases above, ICA does not use label information and therefore, this procedure can be seen as using additional information from unlabelled data which does not obstruct the test results. The final data representation was obtained by estimating the *power spectral density (PSD)* of $\mathbf{WX}$ with Welch's method [102] using 5 windows.

Preprocessing for MEG data: extraction of a time window, band pass filtering and spatial filtering with ICA

The BciComp dataset was preprocessed in exactly the same way.

The time series of the Fingers dataset was band-pass-filtered with a zero-phase-distortion method with a smooth falloff between 0.1Hz and

1Hz, and 5Hz and 6Hz, and downsampled to 12.4378 Hz. For the MEG data, only a spatial filter computed by an Infomax ICA was additionally applied.

Machine learning experiments For the MPIMOV and the BCICOMP dataset, the whole dataset was split into ten splits. For each of these splits, a model selection was performed on the remaining nine splits by a ten fold cross validation in order to choose the best hyperparameters among $C \in \{1, 10, 100\}$, $C_{\mathfrak{U}} \in \{0, 1, 10, 100\}$ and $\varepsilon \in \{0.01, 0.05, 0.1, 0.5\}$. Since non-linear kernels do not lead to better results than linear kernels $k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$ on BCI data [40], the latter was applied thoughout all experiments. After model selection, the algorithm was trained on the nine splits with the best set of hyperparameters from cross validation and tested on the single split. This procedure was carried out for all ten splits, yielding ten test error scores for each algorithm, each Universum and each dataset.

Two nested cross-validations: An inner loop for model selection and an outer loop for testing.

Following the decoding approach for multiclass learning (see section 2.2.1.3), a decoding matrix M was designed for the five classes of the FINGERS dataset. Equation (3.1) shows M. The rows correspond to the classes "no finger", "left little finger", "left thumb", "right thumb" and "right little finger". The columns correspond to the different classification problems. M_{ij} therefore denotes the label for the class i in subproblem j. A zero indicates that the corresponding class was used as Universum in this subproblem. For assigning the class memberships to the test points, L_1 and Hamming decoding as well as loss based decoding for Hinge loss, shortly called "Hinge decoding", was used (see section 2.2.1.3 for the decoding strategies).

Multiclass subproblems for Fingers: (i) is there a finger?, (ii) left or right hand?, (iii) thumb or little finger on the left hand?, thumb or little finger on the right hand?

$$\mathrm{M} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 \\ -1 & 1 & -1 & 0 \\ -1 & -1 & 0 & 1 \\ -1 & -1 & 0 & -1 \end{pmatrix} \tag{3.1}$$

The model selection and training were the same as for the other datasets, except that the whole dataset was divided into fifty instead of ten splits. For the model selection $10^{-1}, 10^{0}, ..., 10^{2}$ multiples of the inverse empirical variance of the data in kernel space $\frac{1}{\frac{1}{m}\sum_{i=1}^{m} k(x_i,x_j) - \frac{1}{m^2}\sum_{i,j=1}^{m} k(x_i,x_j)}$ were used as possible values of C and $C_{\mathfrak{U}}$. This heuristic has been proposed in [15]. Additionally, $C_{\mathfrak{U}}$ could be chosen to be zero during the model selection in order to allow the algorithm to switch off the Universum.

Results and Discussion Tables 3.6, 3.7 and 3.8 show the mean zero-one loss for the datasets MPIMOV, BCICOMP and FINGERS for different Universa. Each error score is the mean over ten (fifty for FINGERS) single error values.

No significant improvements, $\mathfrak{U}_{nm}$ and $\mathfrak{U}_{C3}$ seem to help for subjects JH and S2

On the MPIMOV dataset, there is no improvement in the error scores

		MPIMOV		
		FS	JH	JL
SVM		40.00 ± 7.70	40.00 ± 11.32	30.00 ± 15.54
$\mathfrak{U}$-SVM	$\mathfrak{U}_{C3}$	41.33 ± 7.06	34.58 ± 9.22	30.56 ± 17.22
	$\overline{\mathfrak{U}_{C3}}$	39.33 ± 7.67	35.42 ± 9.67	30.00 ± 16.40
	$\mathfrak{U}_{mean}$	40.00 ± 7.70	42.92 ± 11.29	28.33 ± 15.37
	$\mathfrak{U}_{nm}$	39.67 ± 8.23	37.08 ± 11.69	30.00 ± 16.40
	$\overline{\mathfrak{U}_{nm}}$	40.33 ± 6.37	37.50 ± 12.88	29.44 ± 16.15
$\mathfrak{U}_{\mathfrak{ls}}$-SVM		41.00 ± 7.04	40.42 ± 11.96	30.56 ± 15.77
	$\mathfrak{U}_{C3}$	40.67 ± 7.04	37.08 ± 7.20	31.11 ± 17.01
	$\overline{\mathfrak{U}_{C3}}$	40.67 ± 6.81	37.08 ± 7.47	30.56 ± 15.55
	$\mathfrak{U}_{mean}$	41.67 ± 7.90	40.00 ± 9.66	30.56 ± 15.77
	$\mathfrak{U}_{nm}$	40.67 ± 6.81	37.92 ± 12.65	30.00 ± 15.54
	$\overline{\mathfrak{U}_{nm}}$	40.00 ± 6.85	36.67 ± 11.59	30.00 ± 15.54
$\mathfrak{U}_{\mathfrak{c}}$-SVM	$\mathfrak{U}_{C3}$	40.00 ± 7.70	40.00 ± 11.32	30.00 ± 15.54
	$\overline{\mathfrak{U}_{C3}}$	40.00 ± 7.70	40.00 ± 11.32	30.00 ± 15.54
	$\mathfrak{U}_{mean}$	40.00 ± 7.70	40.00 ± 11.32	30.00 ± 15.54
	$\mathfrak{U}_{nm}$	40.00 ± 7.70	40.00 ± 11.32	30.00 ± 15.54
	$\overline{\mathfrak{U}_{nm}}$	40.00 ± 7.70	40.00 ± 11.32	30.00 ± 15.54

Table 3.6: Mean zero-one test error scores for the MPIMOV dataset for different Universa. The mean was taken over ten single error scores. No significant difference in the error scores could be found (T-test significance level 5%).

for the subjects FS and JL compared to an SVM without Universum. For subject JH the $\mathfrak{U}_{C3}$ and $\mathfrak{U}_{nm}$ and their centered versions yield an improvement of approximately 5% for the $\mathfrak{U}$-SVM and 3% for the $\mathfrak{U}_{\mathfrak{ls}}$-SVM. However, the variance is very high and the differences are not significant. The $\mathfrak{U}_{\mathfrak{ls}}$-SVM performs worse than the $\mathfrak{U}$-SVM in almost all cases. The $\mathfrak{U}_{\mathfrak{c}}$-SVM does not give any improvement compared to an SVM. In fact, the mean error scores and the variance are exactly the same, indicating that $C_{\mathfrak{U}} = 0$ was chosen by the model selection.

On the BCICOMP dataset, only improvements for subject S2 could be obtained. The best improvement of approx. 8% was for Universum $\mathfrak{U}_{nm}$ using a $\mathfrak{U}$-SVM. A minor decrease in the errors of about 3% could be obtained with the Universum $\mathfrak{U}_{C3}$, again using a $\mathfrak{U}$-SVM. However, the difference is not significant. As already observed for the MPIMOV dataset, the $\mathfrak{U}_{\mathfrak{ls}}$-SVM performs constantly worse than its hinge loss counterpart. The $\mathfrak{U}_{\mathfrak{c}}$-SVM yields exactly the same results as a normal SVM.

$\mathfrak{U}_{mean}$ does not increase the accuracy on almost any dataset

On both datasets, the $\mathfrak{U}_{mean}$-Universum does not yield any improvement in almost all cases. In fact, some error scores for the BCICOMP

		BCICOMP		
		S1	S2	S3
SVM		12.35 ± 6.82	35.29 ± 13.30	35.26 ± 14.05
$\mathfrak{U}$-SVM	$\mathfrak{U}_{C3}$	13.53 ± 6.83	32.94 ± 11.83	35.26 ± 14.05
	$\overline{\mathfrak{U}_{C3}}$	14.12 ± 6.90	32.35 ± 10.83	35.79 ± 13.77
	$\mathfrak{U}_{mean}$	14.12 ± 7.94	30.59 ± 14.88	35.79 ± 14.42
	$\mathfrak{U}_{nm}$	12.35 ± 7.04	27.65 ± 14.15	36.84 ± 13.81
	$\overline{\mathfrak{U}_{nm}}$	12.94 ± 6.68	27.65 ± 15.70	35.79 ± 13.77
$\mathfrak{U}_{\mathfrak{ls}}$-SVM		13.53 ± 8.34	33.53 ± 13.60	34.21 ± 12.47
	$\mathfrak{U}_{C3}$	12.94 ± 6.68	32.35 ± 10.83	35.79 ± 15.25
	$\overline{\mathfrak{U}_{C3}}$	14.71 ± 7.96	33.53 ± 11.11	35.79 ± 15.25
	$\mathfrak{U}_{mean}$	17.65 ± 10.74	33.53 ± 13.60	34.21 ± 12.47
	$\mathfrak{U}_{nm}$	16.47 ± 7.74	31.18 ± 13.02	35.79 ± 15.25
	$\overline{\mathfrak{U}_{nm}}$	13.53 ± 8.34	33.53 ± 13.88	35.79 ± 15.25
$\mathfrak{U}_{\mathfrak{c}}$-SVM	$\mathfrak{U}_{C3}$	12.35 ± 7.04	35.29 ± 13.30	35.26 ± 14.05
	$\overline{\mathfrak{U}_{C3}}$	12.35 ± 7.04	35.29 ± 13.30	35.26 ± 14.05
	$\mathfrak{U}_{mean}$	12.35 ± 7.04	35.29 ± 13.30	35.26 ± 14.05
	$\mathfrak{U}_{nm}$	12.35 ± 7.04	35.29 ± 13.30	35.26 ± 14.05
	$\overline{\mathfrak{U}_{nm}}$	12.35 ± 7.04	35.29 ± 13.30	35.26 ± 14.05

Table 3.7: Mean zero-one test error scores for the BCICOMP dataset for different Universa. The mean was taken over ten single error scores. No significant difference in the error scores could be found (T-test significance level 5%).

data are even worse. As for the MNIST dataset, this might be taken as an indication for the hypothesis that the additional information and not a changed feature representation helps to improve the accuracy.

On the FINGERS dataset, there is no significant improvement of $\mathfrak{U}$-SVM for multiclass over standard SVMs. Although the error score for $\mathfrak{U}$-SVMs are constantly about 1% better for H_1 decoding compared to other decoding schemes, the best error scores are obtained for L_1-decoding, for which the effect of using a $\mathfrak{U}$-SVM vanishes. Also, centring does not lead to any significant improvement.

The high variance in the test results in all experiments is due to the small amount of training data. In all cases, the model selection for each split of the data had to be performed on a very small dataset compared to the complexity of the signal. Therefore, it can happen that the algorithm picks up misleading spurious patterns during model selection which results in a high variance in the selected parameters and in a high variance of the error scores. This is also the reason why an SVM with a Universum can yield worse error scores than a standard SVM, even if it is offered to switch off the Universum by setting $C_{\mathfrak{U}} = 0$.

Subject	**SVM**			**$\mathfrak{U}$-SVM**		
	H_1-Decoding	Hamming	L_1-Distance	H_1-Decoding	Hamming	L_1-Distance
FB	42.73± 5.49	39.73 ± 5.79	36.77 ± 6.46	41.67 ± 4.93	39.10± 5.16	36.27 ± 6.44
SH	41.40± 8.11	38.80± 6.43	37.56 ±8.52	40.60± 7.68	38.60± 6.16	37.92 ±7.95
JF	52.32± 6.21	50.04±6.96	45.08± 6.51	51.16±6.57	49.48± 7.10	45.08 ±6.29
	SVM			**$\bar{\mathfrak{U}}$-SVM**		
FB	42.07 ± 5.77	39.73 ± 5.79	36.67 ± 6.06	40.83 ± 5.67	38.30 ± 5.73	35.57 ± 6.79
SH	41.88 ± 8.40	38.80 ± 6.43	37.08 ± 7.65	39.72 ± 7.54	37.32 ± 5.89	37.08 ± 7.65
JF	52.20 ± 6.67	50.04 ± 6.96	45.12 ± 6.71	49.80 ± 6.57	50.04 ± 6.96	44.12 ± 6.73

Table 3.8: Mean zero-one loss for the FINGERS dataset using different Universa. Only the $\mathfrak{U}$-SVM algorithm was used. The mean was taken over 50 single error scores.

During the model selection $C_{\mathfrak{U}} > 0$ might seem to be a good choice for the algorithm but turns out to be a bad decision in the testing stage.

Possible explanations for the improved accuracy using $\mathfrak{U}_{nm}$ and $\mathfrak{U}_{C3}$

The better performance of the $\mathfrak{U}_{nm}$ Universum on the subjects JH and S2 indicates that additional information about the usefulness of features might in fact help to increase the accuracy of the classifier. The regularisation constant $C_{\mathfrak{U}}$ for the Universum points was chosen $C = C_{\mathfrak{U}} = 0.1$ in both cases. This means that the non-orthogonality of w on the Universum points was only weakly penalised, but had equal priority to classifying the labelled examples correctly. In fact, the relative influence of the Universum points on the objective function was even greater than the one of the labelled examples because there were more Universum than labelled points. This could indicate that the spatial filtering is not perfect and discriminative information might be spread over several independent components, even over those that are mainly non-discriminative. This is not surprising since for an unsupervised method like ICA there is no possibility of knowing which feature might be important for classification. Using the $\mathfrak{U}_{nm}$ Universum and therefore gently penalising the use of these non-discriminative features can help to improve the classification accuracy, although the factual usefulness varies with the actual subject.

Interestingly, the subjects where the $\mathfrak{U}_{nm}$ Universum yields an improvement are exactly the ones where the $\mathfrak{U}_{C3}$ Universum helps as well. At the same time, those subjects are not the best performing subjects. It might be that the bad accuracy of an SVM on those subjects is due to other neurological signals that mask the discriminative ones during preprocessing and training of a normal SVM. By specifying desired directions of invariance with a set of Universum points, it could happen that the real signal is enhanced possibly leading to an increased classification accuracy.

The slight improvement for the FINGERS dataset could also be explained by the specification of invariant directions. By using the remaining classes of a multiclass subproblem as Universum points, the single subproblems become more independent, because in the case of perfect orthogonality varying the output of one feature or "bit" would not change the value of any other. Thus, all other classifiers are unaffected if a single classifier, i.e. bit, picks up a wrong pattern. This might be beneficial for decoding a code word in the testing stage by making it more robust against classification errors in single bits.

Possible explanations for the slight improvement in the multiclass classification task

In summary, the most promising results were obtained by using the Universa $\mathfrak{U}_{C3}$ and $\mathfrak{U}_{nm}$ for a imagined movement paradigm. Although none of the error scores was significantly better, the largest improvements in accuracy could be obtained by SVMs using those Universa. For larger training datasets this trends might even become significant. Since those Universa are easy to obtain, their employment in an imagined movement task it seems worthwhile trying.

3.3.2 Stimulus Reconstruction from Spikes

Any sensory information captured by the various sensors of the human body is converted into neural activity and transmitted to the central nervous system via sequences of *action potentials* or *spikes*. One major question in neuroscience is to understand how the nervous system encodes information. [75] describe the task of understanding the neural code as taking the perspective of a homunculus sitting in the brain, whose only information about the outside world are sequences of spikes transmitted to the brain from the body's sensory system. Understanding the neural signals and what they mean to the organism, effectively means decoding the neural code.

Decoding the neural code: The homunculus in the brain

Usually, the relation between real world sensory information and neural activity is explored via neurophysiological experiments, in which stimuli are presented to an animal while the activity of selected neurons is recorded. Given the neural activity in response to a certain stimulus, various methods can be used to analyse the neural responses in order to understand the underlying code. One such method is trying to reconstruct the stimulus from the neural signals. Two kinds of insights can be derived from this method [25]: Firstly, it is possible to examine hypotheses about the neural encoding of the signals used in the experiment. The accuracy of a reconstruction method is a measure of the validity of the assumptions this method is based upon. However, it is impossible to show the sufficiency or the necessity of a certain assumption. All that can be gained is evidence for one or another hypothesis. Secondly, the achieved accuracy of the reconstruction method can indirectly serve as a lower bound on the amount of information that is contained in the spike trains. If the method can reconstruct the stimulus perfectly, then all the necessary information about the stimulus

Why stimulus reconstruction from spikes?

must necessarily be conveyed by the recorded neural activity.

Why use a Universum to do stimulus reconstruction from spikes?

The experiments below are based on [25, 26], who apply various machine learning methods to reconstruct the angle of a sine grating shown to a behaving macaque monkey from the spiking activity of 20 complex cells of the visual cortex. Seven different stimuli were used, leading to a seven class problem. If a 𝔘-SVM can achieve a better accuracy with the multiclass approaches described above, the implicit lower bound on the information contained in the data is decreased and gives evidence for the assumption that the signals for the different stimuli are independent in the employed feature representation, but not picked up by a normal SVM.

Data: Spiking activity from 20 complex cells stimulated with sine gratings of eight different orientations

Data The dataset used in the machine learning experiments below has been recorded at the Neurophysiology department (Dept. Logothetis) of the Max-Planck-Institute for Biological Cybernetics, Tübingen, by Andreas Tolias and coworkers. An awake, behaving monkey (Macaca Mulatta), trained to fixate a small spot on the screen, was shown gratings of eight different orientations ($0°$, $22.5°$, $45°$, $67.5°$, $90°$, $112.5°$, $135°$, $157.5°$) with a contrast of 30%. Simultaneously, the neural activities of a population of 20 complex cells in the primary visual cortex (V1) were recorded. The location of the stimuli on the screen was chosen to cover the receptive field of the neurons. Figure 3.7 shows sine wave gratings as used in the experiments. Each single stimulus was presented for 500ms. Each stimulus was shown exactly 30 times, but in randomised order. Therefore, the whole dataset contained 240 examples. The spikes were recorded extracellularly with tetrodes and sampled at 32kHz, high-pass-filtered and thresholded. The signals from different neurons recorded by the same tetrode were separated by spike sorting [31]. The final data representation was a discretised time series, where the single entries indicated the number of spikes in the respective bin.

Data representation: Bin spikes into 10 non-overlapping time windows of length 80ms to account for rate coding and temporal correlation

Figure 3.8 shows 30 trials from one of the neurons. The considered time window had a total length of 800ms. For the plot, a bin size of 1ms was used in the data representation. Due to the refractory period of an action potential, there can be at most one spike in each bin. For the experiments below a much coarser binning was used, dividing each trial into 10 bins. The reason for such a coarse discretisation can be seen from figure 3.8: When looking at the spiking activity, an increase in the firing rate of the neuron approximately 75ms after stimulus onset can be seen. Furthermore, the spike rate differs depending on the orientation of the presented stimulus. This suggests that the stimulus' orientation is encoded by the spike rate and is therefore an example of *rate coding.* If the bin size was chosen too small, the spike rate would not be captured by the learning algorithm. Apart from the spike rate, a typical temporal pattern can be observed between 250ms and 400ms, which also seems to depend on the stimulus presented. This suggests that the temporal correlation of spike times carries information about

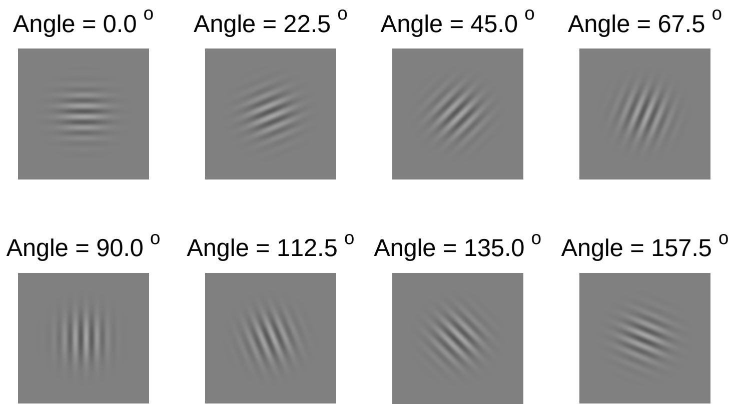


Figure 3.7: Examples of sine wave gratings as used in the recordings from complex cells by Andreas Tolias and coworkers (courtesy of Jan Eichhorn [25]).

the stimulus. This strategy of encoding information is referred to as *temporal coding.* If only the spike count over the whole 800ms was used, the information about the temporal correlation would be lost. According to [25], a bin size of 80ms seems to be a good choice.

Use remaining classes in an one-versus-one decoding approach for multiclass classification as Universum

Universum Since eight different orientations of gratings were used, the reconstruction problem is an eight class problem. As for the FINGERS dataset, the decoding approach to multiclass is used. For the neural data, a one-versus-one encoding matrix was employed, i.e. each possible pair of binary classifiers for eight classes was trained, while the remaining examples were used as Universum. Therefore, the encoding matrix is a $8 \times \frac{8 \cdot 7}{2}$ matrix. An example of an encoding matrix for the one-versus-one approach for four classes has already been given in 2.2.1.3.

Three kernels: RBF kernel, homogeneous spikernel and alignment kernel

Kernels Three different kernels were employed in the experiments below: the RBF kernel $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x}-\mathbf{x}'||^2}{2\sigma^2}\right)$, a common kernel used for dozens of applications, and two kernels that are especially tailored to acting on spike trains [25, 26].

Homogeneous spikernel: Dot production in the space of all possible subsequences of a given length

The first one is the so called *homogeneous spikernel* [26], which is an improvement of the original spikernel [87]. This kernel is designed to even work with sequences of different length and its computation is not as easy as for the RBF kernel. The kernel can conceptually be computed in three steps: In the first step, a sequence s of spike rates is mapped to a function $\Psi_s(\cdot)$ on $\mathbb{R}^n$. In the next step, the L_2 dot product $k_n(s,t) := \int_{\mathbb{R}^n} \Psi_s(\mathbf{u})\Psi_t(\mathbf{u})d\mathbf{u}$ between two such functions is computed. The elements $\mathbf{u} \in \mathbb{R}^n$ represent subsequences of a fixed length n. The final kernel is computed by a weighted sum over all different lengths of $\mathbf{u}$

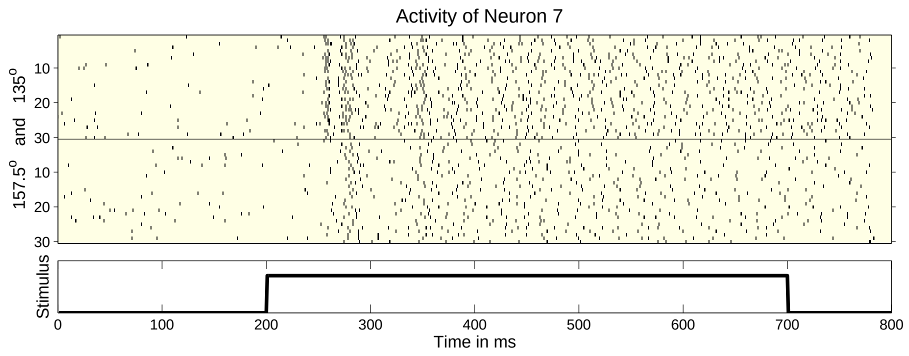


Figure 3.8: Neural activity of a complex cell in the primary visual cortex of a macaque monkey for two different orientations of sine gratings. The figure shows the recorded spiking activity in an 800ms window. The graph in the bottom indicates stimulus on- and offset (courtesy of Jan Eichhorn).

up to a maximum length N: $k(s,t) = \sum_{n=1}^{N} p^n k_n(s,t)$, where the weighting factor p becomes a kernel parameter. In practice, it is not necessary to compute the integrals explicitly. Instead, a dynamic program can be used to obtain the value of $k(s,t)$. Since the dynamic program does not provide any further insight, only the theoretical construction is shown here. The dynamic program can be found in [87, 25, 26].

The function a sequence s is mapped to (the so called *feature map)* is given by

$$\Psi_s : s \mapsto \Psi_s(\mathbf{u}) \quad = \quad C^{\frac{n}{2}} \sum_{\mathbf{i} \in \mathcal{I}_{n,|s|}} \mu^{d(\mathbf{s_i},\mathbf{u})} \lambda^{i_n - i_1},$$

where $\mathcal{I}_{n,|s|}$ is the set of ordered n-tuples $\mathbf{i} = (i_1, ..., i_n)$ of indices $i_1 < i_2 < ... < i_n \in \{1, ..., |s|\}$. C denotes a normalisation constant and $\mu, \lambda \in [0,1]$ are parameters of the kernel. d denotes a distance measure on $\mathbb{R}^n$. For a certain $\mathbf{u}$, $\Psi_s(\mathbf{u})$ yields a mean similarity value between $\mathbf{u}$ and all possible, not necessarily contiguous, subsequences $\mathbf{s_i}$ of s. The similarity is measured by $\mu^{d(\mathbf{s_i},\mathbf{u})}$. Since $0 \leq \mu \leq 1$ the influence of a $\mathbf{u}$ that is not similar to any subsequence of s becomes very low. The additional factor $\lambda^{i_n - i_1}$ decreases the influence of subsequences that are spread out over the whole sequence since $i_n - i_1$ can be seen as a measure of how many elements between i_1 and i_n are not covered by other indices in $\mathbf{i}$. Therefore, Ψ_s favors contiguous subsequences since the elements of a pattern are expected to be close in time. When calculating the integral $k_n(s,t) := \int_{\mathbb{R}^n} \Psi_s(\mathbf{u})\Psi_t(\mathbf{u})d\mathbf{u}$ for two functions Ψ_s and Ψ_t of two sequences s and t, the values of $\Psi_s(\mathbf{u})$ and $\Psi_t(\mathbf{u})$ are compared for each possible $\mathbf{u}$. If $\Psi_s(\mathbf{u})$ and $\Psi_t(\mathbf{u})$ have large values for the same $\mathbf{u}$, $k_n(s,t)$ will have a large value as well and therefore indicate a strong similarity. However, if $\Psi_s(\mathbf{u})$ has large values where

$\Psi_t(\mathbf{u})$ has small ones, $k_n(s,t)$ will be small as well since $\Psi_s(\mathbf{u})$ and $\Psi_t(\mathbf{u})$ dampen each other in the multiplication. Therefore, $k_n(s,t)$ will indicate a low similarity. The same holds for the sum $k(s,t) = \sum_{n=1}^{N} p^n k_n(s,t)$ for the final kernel. If many $k_n(s,t)$ contribute a large value, the final kernel value will be large as well. Since there many more possible subsequences for larger n, the contribution of the single k_n has to be weighted appropriately. This is done by p^n. Choosing $0 < p < 1$, the influence of large subsequence length will be down-weighted. $k(s,t)$ is a valid kernel since the single $k_n(s,t)$ are valid dot products (and therefore positive definite) and kernels are closed under addition.

The homogeneous spikernel can catch rate coded stimuli as well as temporal correlation of the binned spikes.

When using the homogeneous spikernel on a population of neurons, the computation of the kernel is a little more involved. However, there is no principal difference. For a population of M neurons, a specific subsequence $\mathbf{s_i}$ becomes an $M \times n$ matrix $\mathbf{S_i}$ of firing rates. Each single position of $\mathbf{S_i}$ contains the firing rates of the population for a bin corresponding to a certain time window in all simultaneously recorded sequences. $\mathbf{u}$ becomes an $M \times n$ matrix as well, and thus the integration is carried out over $\mathbb{R}^{M\times n}$ instead of $\mathbb{R}^n$.

The homogeneous spikernel can be used on several spike trains from a population of neurons.

The other spike kernel that was employed in the experiments makes use of *alignments*. Alignments are a popular measure in bioinformatics for comparing gene or protein sequences. Given two sequences s and t, aligning them means searching for the cheapest sequence of operations that transforms t into s or vice versa. Each use of an operation is associated with a certain cost. The total cost of transforming s into t is then the sum of the single costs. Common operations are deletion, insertion and substitution. Searching for this cheapest set of operations can be cast into a *shortest path* problem on a graph with $(|s|+1)\cdot(|t|+1)$ nodes as shown in figure 3.9. Each edge in that graph corresponds to a single operation. When using the cost of the operation as edge weight, the shortest path, i.e. the path with minimum total weight, yields the best set of operations to transform s into t and vice versa. The shortest path problem can be solved efficiently, e.g. with the Dijkstra Algorithm [17]. A dynamic program, especially for computing the alignment score, is given in [23]. The global alignment kernel for spikes is then computed as follows: First the minimal cost for transforming one sequence of spike rates into another is computed. Unfortunately, the pairwise alignment scores do not form a valid positive definite kernel. In order to transform the scores into a kernel, the alignment scores of one example with all other training and Universum examples are computed and serve as feature vector. This feature vector is then simply combined with a linear kernel $k(\mathbf{x},\mathbf{x}') = \langle \mathbf{x},\mathbf{x}'\rangle$. In order to use the global alignment kernel for a population of neurons, the linear kernels on the alignment scores of the single neurons are summed up, i.e. $k(\{\mathbf{x}_1,...,\mathbf{x}_M\},\{\mathbf{x'}_1,...,\mathbf{x'}_M\}) = \sum_{i=1}^{M}\langle \mathbf{x}_i,\mathbf{x}'_i\rangle$.

The global alignment kernel: Linear kernel on global alignment scores

A global alignment can be efficiently computed by solving a shortest path problem.

Machine Learning Experiments As in the BCI experiments, the

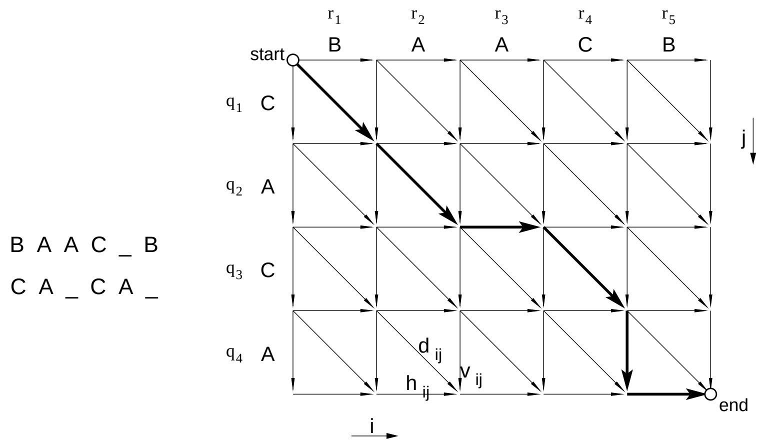


Figure 3.9: Global alignment of two example sequences $\mathbf{r}$ = '$BAACB$' and $\mathbf{q}$ = '$CACA$'. Each path from the start-node to the end-node corresponds to an alignment. Horizontal and vertical edges in the graph denote insertions of a gap into $\mathbf{q}$ or $\mathbf{r}$, respectively. Diagonal edges correspond to substitutions. For the best global alignment, the edges are provided with costs and the path of minimal total cost is computed (courtesy of Jan Eichhorn).

whole training set of 240 examples was split into ten parts. For each part, a model selection was performed on the other nine parts in order to choose the best set of hyperparameters for the algorithms and kernels. For the SVM, C was chosen among $C \in \{1, 10, 100\}$. For the $\mathfrak{U}$-SVM, $\mathfrak{U}_{ls}$-SVM and $\mathfrak{U}_c$-SVM $C_{\mathfrak{U}}$ was selected from $C_{\mathfrak{U}} \in \{0, 1, 10, 100\}$. For $\mathfrak{U}$-SVM and $\mathfrak{U}_{ls}$-SVM, $\varepsilon \in \{0.01, 0.05, 0.1, 0.5\}$ was used. The kernel parameters μ, λ, N and p for spikernel as well as the costs for the alignment kernel are selected according to the values suggested in [25]. After selecting the best set of hyperparameters for the nine splits, the algorithm was trained with those hyperparameters on the nine splits and tested on the remaining one. This was done for all ten splits. The ten error scores resulting from that scheme were averaged and a T-test was performed to test for statistically significant differences in the error scores.

Results and Discussion Tables 3.9, 3.10 and 3.11 show the prediction errors for the homogeneous spikernel, the RBF kernel and the alignment kernel. Apart from the zero-one loss for classification, an angular loss measure is shown which takes into account the angular structure of the different stimuli. As mentioned above, the different stimuli were sine gratings presented in the orientations $0°$, $22.5°$, $45°$, $67.5°$, $90°$, $112.5°$, $135°$ and $157.5°$. Therefore, the stimulus space has a structure that allows to compare the different classes. When using the zero-one loss, a prediction can be either correct or completely wrong

	$\ell_{0\text{–}1}$ for d_{L_1}	$\ell_\circ$ for d_{L_1}	$\ell_{0\text{–}1}$ for d_{ham}	$\ell_\circ$ for d_{ham}	$\ell_{0\text{–}1}$ for d_{hin}	$\ell_\circ$ for d_{hin}
SVM	4.17 ± 3.40	$0.94° \pm 0.77°$	5.42 ± 3.95	$1.22° \pm 0.89°$	6.25 ± 4.05	$1.41° \pm 0.91°$
LS-SVM	4.58 ± 4.14	$1.22° \pm 1.09°$	4.17 ± 4.39	$1.03° \pm 1.21°$	6.25 ± 4.05	$1.22° \pm 1.09°$
$\mathfrak{U}$-SVM	4.17 ± 3.40	$0.94° \pm 0.77°$	5.42 ± 3.95	$1.22° \pm 0.89°$	5.42 ± 4.83	$1.41° \pm 0.91°$
$\mathfrak{U}_{\mathfrak{ls}}$-SVM	6.25 ± 5.29	$1.41° \pm 1.19°$	4.58 ± 4.14	$1.12° \pm 1.15°$	5.00 ± 5.12	$1.12° \pm 1.15°$
$\mathfrak{U}_{\mathfrak{c}}$-SVM	4.58 ± 3.07	$1.12° \pm 0.86°$	6.25 ± 5.29	$1.41° \pm 1.19°$	5.83 ± 4.48	$1.31° \pm 1.01°$

Table 3.9: Zero-one loss $\ell_{0\text{–}1}$ and angular loss $\ell_\circ$ for the homogeneous spikernel averaged over ten splits of the dataset into training and test parts. Three different decoding schemes have been employed: L_1 decoding d_{L_1}, Hamming decoding d_{ham} and hinge loss based decoding d_{hin}.

	$\ell_{0\text{–}1}$ for d_{L_1}	$\ell_\circ$ for d_{L_1}	$\ell_{0\text{–}1}$ for d_{ham}	$\ell_\circ$ for d_{ham}	$\ell_{0\text{–}1}$ for d_{hin}	$\ell_\circ$ for d_{hin}
SVM	7.50 ± 4.73	$1.88° \pm 1.40°$	6.25 ± 4.91	$1.88° \pm 1.88°$	6.67 ± 5.27	$1.69° \pm 1.45°$
LS-SVM	7.08 ± 5.22	$1.78° \pm 1.43°$	5.83 ± 4.48	$1.50° \pm 1.34°$	6.67 ± 4.89	$1.69° \pm 1.38°$
$\mathfrak{U}$-SVM	6.25 ± 4.50	$1.59° \pm 1.33°$	5.83 ± 4.48	$1.50° \pm 1.34°$	7.92 ± 5.71	$2.16° \pm 1.82°$
$\mathfrak{U}_{\mathfrak{ls}}$-SVM	7.50 ± 5.49	$1.88° \pm 1.47°$	5.83 ± 4.48	$1.50° \pm 1.34°$	7.08 ± 4.41	$1.78° \pm 1.28°$

Table 3.10: Zero-one loss $\ell_{0\text{–}1}$ and angular loss $\ell_\circ$ for the RBF kernel averaged over ten splits of the dataset into training and test parts. Three different decoding schemes have been employed: L_1 decoding d_{L_1}, Hamming decoding d_{ham} and hinge loss based decoding d_{hin}. Due to the equivalence of the $\mathfrak{U}_{\mathfrak{c}}$-SVM and the $\mathfrak{U}$-SVM in the RBF case, only the results for the $\mathfrak{U}$-SVM are shown here.

and therefore it does not take into account any metric structure on the outputs. Since there is clearly some structure on the outputs in the current case, confusing two orientations with a small angular difference should be penalised less than confusing two orthogonal gratings. In order to respect the output structure, [25] uses an adapted loss function:

$$\ell_\circ(\alpha, \beta) \quad = \quad \min\{|\alpha - \beta|, 180° - |\alpha - \beta|\}.$$

Here, α denotes the true orientation angle of the stimulus and β denotes the orientation angle associated with the predicted class.

The error scores shown in tables 3.9, 3.10 and 3.11 are comparable to the results of [25, 26]. Unfortunately, the error scores show that no Universum algorithm performs significantly better than its SVM counterpart. In fact, the best performance is achieved by a plain SVM with L_1 decoding. On average, the least squares SVMs seem to perform a little worse than the SVMs using hinge loss. The only case in which the $\mathfrak{U}_{\mathfrak{ls}}$-SVM performs significantly better than an LS-SVM is for the global alignment kernel. However, this cannot be imputed on the Universum

	$\ell_{0\text{–}1}$ for d_{L_1}	$\ell_\circ$ for d_{L_1}	$\ell_{0\text{–}1}$ for d_{ham}	$\ell_\circ$ for d_{ham}	$\ell_{0\text{–}1}$ for d_{hin}	$\ell_\circ$ for d_{hin}
SVM	5.83 ± 5.27	$1.29^\circ \pm 1.60^\circ$	4.58 ± 4.14	$1.03^\circ \pm 0.93^\circ$	5.00 ± 4.30	$1.50^\circ \pm 1.61^\circ$
LS-SVM	53.33 ± 4.30	$16.41^\circ \pm 5.27^\circ$	54.48 ± 6.65	$15.28^\circ \pm 4.33^\circ$	52.08 ± 6.65	$13.03^\circ \pm 1.85^\circ$
$\mathfrak{U}$-SVM	5.83 ± 5.27	$1.59^\circ \pm 1.60^\circ$	4.58 ± 4.14	$1.03^\circ \pm 0.93^\circ$	5.00 ± 4.30	$1.50^\circ \pm 1.61^\circ$
$\mathfrak{U}_{\mathfrak{ls}}$-SVM	5.00 ± 3.83	$1.31^\circ \pm 1.10^\circ$	7.08 ± 5.22	$1.78^\circ \pm 1.50^\circ$	7.92 ± 6.65	$1.97^\circ \pm 1.85^\circ$
$\mathfrak{U}_{\mathfrak{c}}$-SVM	5.83 ± 5.27	$1.59^\circ \pm 1.60^\circ$	5.42 ± 3.43	$1.22^\circ \pm 0.77^\circ$	5.00 ± 4.30	$1.50^\circ \pm 1.61^\circ$

Table 3.11: Zero-one loss $\ell_{0\text{–}1}$ and angular loss $\ell_\circ$ for the global alignment kernel averaged over ten splits of the dataset into training and test parts. Three different decoding schemes have been employed: L_1 decoding d_{L_1}, Hamming decoding d_{ham} and hinge loss based decoding d_{hin}.

set, but rather on the global alignment kernel, which leads to extremely badly conditioned kernel matrices for the employed data. The matrix inversion involved in solving an LS-SVM or a $\mathfrak{U}_{\mathfrak{ls}}$-SVM may therefore lead to numerical problems. Even though the LS-SVM was offered extremely small values for C during the model selection, which amounts to heavy regularisation of the kernel matrix' diagonal, the predictions of a plain LS-SVM are still very bad. Another reason for the very low quality of the LS-SVM solution might be the very small number of training points in a one-versus-one scheme for eight classes on 240 training points in total. Using the remaining points as Universum points seems to have a numerically stabilising effect in this case.

The low performance of the Universum algorithms on this particular neural decoding problems indicates that the features in the spike trains for discriminating the different stimuli are not independent since the classification accuracy cannot be leveraged by a classifier, which implements this independence assumption. Theoretically, enhancing the contrast between the classes by dampening typical features for other stimuli by means of the Universum algorithms could help to augment the algorithm's performance. However, it seems that the employed data representations are not suited for such a benefit from the Universum. It seems that forcing the data from other classes to zero in order to enhance the difference between the two labelled classes also had a harmful impact on the ability of the classifier to detect discriminative features.

Chapter 4

Discussion, Conclusion and Future Work

4.1 Discussion and Conclusion

This thesis investigated the applicability of non-examples $\mathsf{U} \not\sim \mathbf{P}_{\mathsf{X}}$ for data-dependent regularisation on the basis of the so-called Universum algorithm [103] and the underlying idea of inference with a Universum by Vladimir Vapnik [99, 98].

$\mathfrak{U}_c$-SVM: Relation between margin and contradictions on $\mathfrak{U}$

Chapter 2 analysed the relationship between Vapnik's original idea of maximising the number of contradictions, the algorithmic implementation by [103] and the theoretical consequences for possible Universum sets. After stating the original Universum algorithm in an SVM with hinge loss in 2.2.1.1, a least squares version was introduced in 2.2.1.2. A lower bound on the number of contradictions in terms of the margin and the number of Universum examples with bounded norm inside an ε-tube was established in section 2.3. Based on this bound, a change of the constraints of the optimisation problem for SVMs was proposed, leading to the formulation of the $\mathfrak{U}_c$-SVM.

$\mathfrak{U}$-SVM: Specifying unwanted basis functions

The role of the Universum points in the algorithmic implementations was theoretically explored for the hinge loss $\mathfrak{U}$-SVM in 2.4.1 and for the quadratic loss $\mathfrak{U}_{\mathfrak{ls}}$-SVM in 2.4.2. Both sections revealed that the Universum points specify directions the resulting learning algorithm should be invariant against. In particular, it was shown that a hard margin $\mathfrak{U}$-SVM without offset b corresponds to using a kernel, where the span of the Universum points in the Hilbert space $\mathcal{H}$ is removed from the span of the training examples. Therefore, the learning algorithm's solution is restricted to the orthogonal complement of $\mathfrak{U}$ in $\mathcal{H}$ and variations of the data in the span of $\mathfrak{U}$ do not affect the value of the function. Thus, the solution is invariant against features represented by $\mathfrak{U}$. Another view is to interpret each Universum example together with the kernel function as a filter k_z that responds to certain features of the data. Enforcing $\langle \mathrm{w}, k_z \rangle = 0$ can then be seen as keeping the learning algorithm from using certain feature filters, i.e. the ones that are contained in the Universum set $\mathfrak{U}$.

$\mathfrak{U}_{\mathfrak{ls}}$-SVM: Specifying a noise covariance

A similar role of the Universum points in the $\mathfrak{U}_{\mathfrak{ls}}$-SVM was demonstrated in 2.4.2 by establishing a link between the $\mathfrak{U}_{\mathfrak{ls}}$-SVM and a hybrid of the two well-known learning algorithms kernel Fisher Discriminant Analysis and kernel oriented Principal Component Analysis. It was shown that covariance of the Universum points, centered on the mid-point between the two class means, plays the part of the noise covariance in the hybrid. The hybrid of kFDA and koPCA and therefore the $\mathfrak{U}_{\mathfrak{ls}}$-SVM aims at finding a solution that is maximally orthogonal to the principal directions of the noise while still retaining a good classification of the training points. This can again be interpreted as choosing a solution that is maximally invariant against patterns contained in $\mathfrak{U}$.

Universum algorithms: Data-dependent regulariser for implicit invariances

Due to the direct dependence of the learning algorithm's solution on the Universum set, the loss term on the Universum can be seen as a data-dependent regulariser in the style of the Virtual Support Vector method [19, 12, 13] or noise injection [35]. However, while the Virtual Support Vector method needs explicitly known transformations in order to train a more robust classifier, the Universum allows to specify much more general directions of invariance which are implicitly handed over to the algorithm in the set $\mathfrak{U}$. In many applications, specifying a suitable Universum set might be easier than adapting a general purpose prior or regulariser to a given problem. In fact, the Universum loss terms determine a semi-norm[1] $||h||_{\mathfrak{U}}$ on $\mathcal{H}$ which corresponds to a Gaussian prior $\mathsf{P}_{\mathsf{H}} \propto \exp(-||h||_{\mathfrak{U}})$. It is important to note that, unless $\mathfrak{U}$ spans the entire space, P_{H} is not normalisable. However, this is not a big problem in practice, where P_{H} is simply assumed to have a constant value on regions where $||\cdot||_{\mathfrak{U}} = 0$. Using this prior means that the functions $\{h| \; ||h||_{\mathfrak{U}} = 0\}$ are not regularised, i.e. with only the regulariser $||\cdot||_{\mathfrak{U}}$ the algorithm has no preference between two functions that classify the training data equally well. In the case of the SVM implementations in 2.2.1.1 and 2.2.1.2, the general regulariser $||h||_2^2 = \langle h, h \rangle$ is still used and acts on all functions, while the data-dependent regulariser acts only on the complement of $\{h| \; ||h||_{\mathfrak{U}} = 0\}$.

Maximum number of contradictions on $\mathfrak{U}$ vs. maximum entropy on $\mathsf{P}_{\mathsf{Y}|\mathfrak{L},x,z}$

After analysing the algorithmic implementation in sections 2.3 and 2.4, the original idea of maximising the number of contradictions and its relation to a maximum entropy label distribution on the Universum points was explored in 2.5. After introducing the Bayesian model in version space as stated by [32, 38], it was shown that the entropy of the distribution of labels on independently drawn Universum points serves as a lower bound on the number of contradictions. Since every test example x induces two label distributions on $\mathfrak{U}$, the choice of the label

[1]For a semi-norm $||\mathbf{x}|| = 0$ does not necessarily imply $\mathbf{x} = 0$.

for x can be seen as a choice among those distributions. The notion of the maximum entropy distribution also captures the initial idea of the Universum by Vapnik, namely that the learning algorithm should be most unspecific about the examples in $\mathfrak{U}$. Interestingly, relating the maximum entropy model in version space to the employed loss function on the Universum points in the $\mathfrak{U}$-SVM implementation leads to a set of constraints on the Universum points which were introduced for the $\mathfrak{U}_c$-SVM.

Significantly better performance of the Universum on digit recognition and text classification

In the experimental chapter 3, the derived algorithms were tested on real world data. In the first two experimental setups, the algorithms of chapter 2 were tested on two standard well behaved machine learning problems: digit recognition and text classification. In both cases, employing a real world Universum improved the classification accuracy significantly. Artificially generated Universa which could not add any additional information via data points did not augment the performance of the classifier. This indicates that the essence of the Universum implementation is adding additional knowledge to the problem and not transforming the data in a suitable way for the algorithm.

Non-significant, but promising trends on Brain Computer Interface problems

The second and larger part of chapter 3 consisted of applying the Universum to two learning problems of psychology and neuroscience: Brain Computer Interfaces and neural decoding. The experiments with Brain Computer Interfaces involved an EEG experiment to explore the possibility of recording Universum data from an additional experimental condition that would serve as a baseline or background noise in the Universum learning algorithm. Other datasets and possible Universa were also tested. One of them was constructed from physiological evidence in order to account for the non-discriminative desynchronisation in the α-band related to vision. Although the accuracies did not increase significantly, there was an apparent trend of lower error scores for the learning algorithm employing real world Universa, especially the $\mathfrak{U}_{C3}$ from the third condition and the constructed $\mathfrak{U}_{nm}$. A possible reason for the non-significance of the results might be the small amount of data due to the tedious recording procedure in BCI. Nevertheless, low quality of the signals makes BCI a very difficult learning problem and even a slight improvement is already a success. It also turned out that the success of employing a Universum depends on the specific subject. This is not a big surprise since every subject might have his own strategy to produce a detectable signal in a BCI. Therefore, it seems appropriate to design or record Universum sets for each subject individually. Generating data by looking at the information sources for the different independent spatial filters as done for the $\mathfrak{U}_{nm}$ set seems to be a cheap and effortless strategy to try.

No accuracy improvement for neural decoding

On the neural decoding problem, the Universum neither yielded any significant improvement nor did it show any apparent trend. Although the offered Universa consisting of the remaining classes in the one-versus-one scheme were not switched off by setting $C_{\mathfrak{U}} = 0$ in the model selection stage, this choice turned out to be irrelevant or even harmful for the prediction error on the test set. It is hard to say whether more training data would lead to an improvement of the error scores via the Universum, or whether it would make the learning algorithm switch off the influence of the Universum by setting $C_{\mathfrak{U}} = 0$ during the model selection. Neural decoding seems to be one of the problems where it is not very clear what good directions of invariance in terms of spike trains should look like.

Evidence against maximising the number of contradictions?

An interesting observation that can be made in all experiments with the Universum algorithms is that the $\mathfrak{U}_{\mathfrak{c}}$-SVM is not superior to the original $\mathfrak{U}$-SVM in any case, although from a theoretical point of view it is closer to the original idea of maximising the number of contradictions on the Universum set, as explored in sections 2.3 and 2.5. Although the considerations of 2.3 and 2.5 show a relation between maximising the number of contradictions and putting the Universum points close to the decision surface, it is still not clear if the good classification results are an artifact of the algorithmic approximation of Vapnik's idea or if it is actually the original Universum idea which augments the algorithm's performance. This has to be investigated in more detail in the future.

4.2 Future Work

There are several directions of future research that seem interesting to pursue. The first is to catechise the case $b \neq 0$ for the orthogonal subspace setting described in 2.4.1. It might be possible to prove a similar statement for this case as for the case $b = 0$, for which the Universum data is simply projected out from the space of admissible functions.

General Universum kernel for regression and Bayesian inference

Furthermore, one could use the results of section 2.4.1 for designing a kernel that depends on the Universum points and that can be used in any kernelised learning algorithm, especially in Bayesian learning algorithms like Gaussian processes. Using this kernel, the Universum concept could even be used in regression. A first naïve try could be to smoothly control the part of the kernel that projects out the Universum points with an additional kernel parameter $\theta \in (-\infty, \infty)$, e.g.

$$\begin{aligned} k^{\perp}(x,x') &= k(x,x') - \frac{\tanh(\theta)+1}{2} k^{||}(x,x') \\ &= k(x,x') - \frac{\tanh(\theta)+1}{2} \left(\sum_{r,s=1}^{q} (\mathbf{K}^{-1}_{(\mathfrak{U},\mathfrak{U})})_{rs} k(x,z_r) k(x',z_s) \right). \end{aligned}$$

By virtue of this definition, $k^{\perp}$ would be a meta-kernel that could be wrapped around any existing one. The factor $\frac{\tanh(\theta)+1}{2} \in (0,1)$ allows to control the impact of the set $\mathfrak{U}$ in a smooth manner. Since the kernel is then a smooth function of θ, it can be differentiated with respect to θ and therefore be used for inference with Bayesian learning algorithms like Gaussian processes, which often require the kernel to be a differentiable function of its parameters for model selection.

Maximising Contradictions or Invariance Directions: Playing Billiard in version space

The last direction which shall be mentioned here is the exploration of the connection between Vapnik's idea of maximising the number of contradictions on the Universum and the actual algorithmic implementations in greater detail. As indicated by the results of sections 2.3 and 2.5, there seems to be a relation between both, but it is still not clear whether the algorithmic implementation of the underlying idea of contradiction maximisation accounts for the effect on real world data. One possible strategy to pursue this would be to directly implement Vapnik's idea by means of an algorithm by Rujan [78], called *Billiard in Version Space*. The basic idea of the algorithm is actually very simple. A fictitious ball is placed in the version space and moved into a random direction. Once it hits the boundary of the version space, i.e. the hyperplance corresponding to a labelled example, it is reflected. For implementing the original idea, it would be necessary to keep track of two things during the ball's flight in version space: transitions through hyperplanes corresponding to unlabelled test examples and transitions through hyperplanes associated with Universum points. By counting the number of transitions through hyperplanes of Universum points inside a certain sub-compartment of the version space induced by a certain labelling of the test points, one could count the number of contradictions for this specific labelling since each crossed Universum plane inside a sub-compartment gives rise to a contradiction. Transitions through hyperplanes of test examples simply change the sub-compartment of the version space, i.e. swap the label of the example corresponding to the crossed hyperplane. It would be interesting to test the Billiard on artificial as well as on real world data and to explore the necessary relation between the distribution $\mathbf{P}_{\mathrm{U}}$ of the Universum points and the distribution $\mathbf{P}_{\mathrm{XY}}$ of the training data in order for the maximum contradiction idea to be useful. After deriving the relations, one could check if these are also implemented by the $\mathfrak{U}$-SVMs.

Maximising entropy by playing Billiard in version space.

Since, under certain assumptions, the average flight time of the Billiard ball in version space is proportional to the volume of the area it is traversing through, it would even be possible to implement the maximum entropy approach of section 2.5 with the Billiard in version space and compare its performance to the performance of maximising the number of contradictions.

A new inference principle?

Only if choosing the contradiction-maximising labelling of the test examples offers advantages over existing learning methods under well defined conditions, the Universum idea can be seen as a valid new inference principle.

Acknowledgements

First of all, I would like to thank Olivier Chapelle and Bernhard Schölkopf for their supervision, patient discussions and their general support and advice. Their questions, hints, critics and ideas inspired me a lot. I also would like to thank Jeremy Hill, Bernd Battes and Navin Lal for their great help and advice with the Brain-Computer-Interface and the data analysis. Furthermore, I want to thank Jakob Macke, Frank Jäkel, Sebastian Gerwinn and Stefanie Jegelka for spending time discussing with me about my problems with the Universum, as well as Lisa Hartmann and Stefanie Jegelka for proofreading and comments on the manuscript.

Finally I would like to thank Jason Weston, Ronan Collobert, Leon Bottou and Vladimir Vapnik for inviting me to NEC Research, most friendly welcoming me and giving me the chance to participate at the initial algorithmic implementation of the Universum.

Appendix

4.3 Definitions

DEFINITION: CAUCHY SEQUENCE

A sequence $(a_n)_{n\in\mathbb{N}}$ is called *Cauchy sequence* if for each $\varepsilon > 0$ there exists an $n_0 \in \mathbb{N}$ such that

$$\forall n, m > n_0 : |a_m - a_n| \quad < \quad \varepsilon$$

▷

DEFINITION: SPAN

The set of all linear combinations of elements of a subset V of a vector space

$$\text{span}\{V\} \quad := \quad \{\sum_{i\in I} a_i v_i | a_i \in \mathbb{R} \text{ and } v_i \in V\}$$

is called *span* of V.

▷

DEFINITION: DOT PRODUCT

A function $\langle\cdot,\cdot\rangle : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called *dot product* if it fulfills the following properties:

1. Symmetry: $\langle x, x'\rangle = \langle x', x\rangle$ for all $x, x' \in \mathcal{X}$
2. Bilinearity: $\langle a_1x_1 + a_2x_2, a_3x_3 + a_4x_4\rangle = a_1a_3\langle x_1, x_3\rangle + a_1a_4\langle x_1, x_4\rangle + a_2a_3\langle x_2, x_3\rangle + a_2a_4\langle x_2, x_4\rangle$ for all $a_1, ..., a_4 \in \mathbb{R}$ and $x_1, ..., x_4 \in \mathcal{X}$
3. Strict positive definiteness: $\langle x, x\rangle \geq 0$ for all $x \in \mathcal{X}$, with equality only if $x = \mathbf{0}$.

▷

DEFINITION: PROJECTION IN HILBERT SPACES

Let $\mathcal{H}$ be a Hilbert space and $V \subseteq \mathcal{H}$ a closed subspace. Then each element $h \in \mathcal{H}$ can be uniquely written as $h = v + v^{\perp}$, where $v \in V$ and $\langle w, v^{\perp} \rangle = 0$ for all $w \in V$. The element v is the unique element of V minimising the distance to h induced by the norm of $\mathcal{H}$ and is called the projection $Ph := v$ of h onto V. The projection operator is a linear map.

$\triangleright$

Bibliography

[1] Erin L. Allwein, Robert E. Schapire, and Yoram Singer, *Reducing multiclass to binary: A unifying approach for margin classifiers*, Proc. 17th International Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA, 2000, pp. 9–16.

[2] Anthony J. Bell and Terrence J. Sejnowski, *An information-maximization approach to blind separation and blind deconvolution*, Neural Computation **7** (1995), no. 6, 1129–1159.

[3] Hans Berger, *Das Elektrenkephalogramm des Menschen*, Acta Leopoldina **6** (1938), no. 38, 173–309.

[4] G.E. Birch, S.G. Mason, and J.F. Borisoff, *Current trends in brain-computer interface research at the neil squire foundation*, IEEE Transactions on Rehabilitation Engineering **11** (2003), no. 2, 1534–4320.

[5] Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik, *A training algorithm for optimal margin classifiers*, Computational Learing Theory, 1992, pp. 144–152.

[6] O. Bousquet, O. Chapelle, and M. Hein, *Measure based regularization*, Advances in Neural Information Processing Systems **16** (2004).

[7] ______, *Measure based regularization*, Advances in Neural Information Processing Systems (Cambridge, MA USA) (L. Saul Thrun, S. and B. Schölkopf, eds.), vol. 16, MIT Press, December 2004.

[8] Stephen P. Boyd and Lieven Vandenberghe, *Convex optimization*, 1st ed., Cambridge University Press, 2004.

[9] C. J. C. Burges, *A tutorial on support vector machines for pattern recognition*, Data Mining and Knowledge Discovery **2** (1998), no. 2, 121–167.

[10] HO Y. C. and PEPYNE D. L., *Simple explanation of the no-free-lunch theorem and its implications*, Journal of optimization theory

and applications (J. optim. theory appl.) ISSN 0022-3239 **115** (2002), 549–570.

[11] S. Canu and A. Elisseeff, *Regularization, kernels and sigmoid nets*, INSA, Rouen, 1999.

[12] O. Chapelle and B. Schölkopf, *Incorporating invariances in nonlinear support vector machines*, Tech. report, Max Planck Institute for biological Cybernetics, 2001.

[13] ______, *Incorporating invariances in nonlinear svms*, Advances in Neural Information Processing Systems (Cambridge, MA, USA) (T. G. Dietterich, S. Becker, and Z. Ghahramani, eds.), vol. 14, MIT Press, 2002, pp. 609–616.

[14] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-supervised learning*, MIT Press, Cambridge, MA, 2006.

[15] O. Chapelle and A. Zien, *Semi-supervised classification by low density separation*, Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (New York), 2005, pp. 57–64.

[16] Adrian Cordunaunu and Tommi Jaakkola, *Data-dependent regularization*, ch. 10, pp. 169 – 190, MIT Press, 2005.

[17] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to algorithms*, 2 ed., MIT Press, 2001.

[18] Thomas M. Cover and Joy A. Thomas, *Elements of Information Theory*, John Wiley & Sons Inc, 19 September 2006.

[19] D. DeCoste and B. Schölkopf, *Training invariant support vector machines.*, Machine Learning **46** (2002), 161–190.

[20] A. Delorme and S. Makeig, *EEGLab: An open source toolbox for analysis of single-trial dynamics including indenpendent component analysis*, 2003.

[21] K.I. Diamantaras and S.Y. Kung, *Principal component neural networks, theory and applications*, Wiley, 1996.

[22] Harris Drucker, Chris J. C. Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik, *Support vector regression machines*, Advances in Neural Information Processing Systems (Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, eds.), vol. 9, The MIT Press, 1997, p. 155.

[23] Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern classification*, 2 ed., John Wiley and Sons, 2001.

[24] R. Dudley, *Real analysis and probabiliy*, Wadsworth & Brooks/Cole Advanced books & Software, 1989.

[25] J. Eichhorn, *Application of kernel methods to structured data*, Ph.D. thesis, Technical University Berlin, 2006.

[26] J. Eichhorn, A.S. Tolias, A. Zien, M. Kuss, C. E. Rasmussen, J. Weston, N.K. Logothetis, and B. Schölkopf, *Prediction on spike data using kernel algorithms*, Advances in Neural Information Processing Systems (Cambridge, MA, USA) (L. Saul Thrun, S. and B. Schölkopf, eds.), vol. 16, MIT Press, 2004, pp. 1367–1374.

[27] Gerd Fischer, *Lineare Algebra. Eine Einführung für Studienanfänger*, Vieweg Verlag, September 2005.

[28] R.A. Fisher, *The Use of Multiple Measurements in Taxonomic Problems*, Annals of Eugenics **7** (1936), 179–188.

[29] Karl Friedrich Gauß, *Theoria motus corporum coelestium in sectionibus conicis solem ambientium*, F. Perthes and I.H. Besser, Hamburg, 1809.

[30] G.J. McLachlan G.J. McLachlan, D.Peel, *On computational aspects of clustering via mixtures of normal and t-components*, Tech. report, University of Queensland, 1998.

[31] Dilan Görür, Carl Edward Rasmussen, Andreas S. Tolias, Fabian Sinz, and Nikos K. Logothetis, *Modelling spikes with mixtures of factor analysers*, Proceedings of DAGM 2004, 2004, pp. 391–398.

[32] Thore Graepel, Ralf Herbrich, and Klaus Obermayer, *Bayesian Transduction*, Advances in Neural Information System Processing, vol. 12, 2000, pp. 456–462.

[33] ______, *Bayesian Transduction*, Advances in Neural Information System Processing, vol. 12, 2000, pp. 456–462.

[34] J.E. Graimann, B. abd Huggins, S.P. Levine, and G. Pfurtscheller, *Towards a direct brain interface based on human subdural recordings and wavelet packet analysis*, IEEE Trans BME **51** (2004), no. 6, 954–962.

[35] Yves Grandvalet, Stephane Canu, and Stephane Boucheron, *Noise injection: Theoretical prospects*, Neural Computation **9** (1997), no. 5, 1093–1108.

[36] C. Guger, G. Edlinger, W. Harkam, I. Niedermayer, and G. Pfurtscheller, *How many people are able to operate an eeg-based brain-computer interface (bci)?*, IEEE Trans. On Rehabilitation Engineering **11** (2003), no. 2, 145–147.

[37] C. Guger, W. Harkam, C. Hertnaes, and G. Pfurtscheller, *Prosthetic control by an eeg-based brain-computer interface (bci)*, Nov. 1999.

[38] Ralf Herbrich, Thore Graepel, and Colin Campbell, *Bayes point machines*, Journal of Machine Learning Research **1** (2001), 245–279.

[39] Magnus R. Hestenes and Eduard Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards **49** (1952), 409–436.

[40] N. J. Hill, T. N. Lal, M. Schröder, T. Hinterberger, B. Wilhelm, F. Nijboer, U. Mochty, G. Widman, C. E. Elger, B. Schölkopf, A. Kübler, and N. Birbaumer, *Classifying EEG and ECoG signals without subject training for fast bci implementation: Comparison of non-paralysed and completely paralysed subjects*, IEEE Transactions on Neural Systems and Rehabilitation Engineering **14** (2006), no. 2, 183–186.

[41] N.J. Hill, T.N. Lal, K. Bierig, N. Birbaumer, and B. Schölkopf, *An auditory paradigm for brain–computer interfaces*, Advances in Neural Information Processing Systems (Cambridge, MA, USA) (Y. Weiss Saul, L.K. and L. Bottou, eds.), vol. 17, MIT Press, 2005, pp. 569–576.

[42] Thomas Hofmann, Bernhard Schölkopf, and Alexander J. Smola, *Kernel methods for machine learning*, Annals of Statistics (2007), (accepted).

[43] David Hume, *A treatise of human nature*, https://www.gutenberg.org, 2003.

[44] David Hume, *An enquiry concerning human understanding*, http://www.gutenberg.org, 2006.

[45] E. T. Jaynes and G. L. Bretthorst, *Probability Theory. The Logic of Science*, Cambridge University Press, 2003.

[46] Immanuel Kant, *Kritik der reinen Vernunft*, Meiner, 1998.

[47] L. Kauhanen, P. Rantanen, Lehtonen, J. A., I. Tarnanen, H. Alaranta, and M. Sams, *Sensorimotor cortical activity of tetraplegics during attempted finger movements*, Biomedizinische Technik **49** (2004), no. 1, 59–60.

[48] Herbert Keuth, *Die Philosophie Karl Poppers*, UTB für Wissenschaft, 2000.

[49] W. Kienzle and K. Chellapilla, *Personalized handwriting recognition via biased regularization*, International Conference on Machine Learning, 03 2006.

[50] G. S. Kimeldorf and G. Wahba, *Some results on Tchebycheffian spline functions*, Journal of Mathematical Analysis and Applications **33** (1971), 82–95.

[51] K. Kirschfeld, *The physical basis of alpha waves in the electroencephalogram and the origin of the "berger effect"*, Biological Cybernetics **92** (2005), no. 3, 177–85.

[52] T. N. Lal, *Machine learning methods for brain-computer interdaces*, Ph.D. thesis, University Darmstadt, 09 2005, Logos Verlag Berlin MPI Series in Biological Cybernetics, Bd. 12 ISBN 3-8325-1048-6.

[53] T. N. Lal, M. Schröder, J. Hill, H. Preissl, T. Hinterberger, J. Mellinger, M. Bogdan, W. Rosenstiel, T. Hofmann, N. Birbaumer, and B. Schölkopf, *A brain computer interface with online feedback based on magnetoencephalography*, Proceedings of the 22nd International Conference on Machine Learning (S. Wrobel De Raedt, L., ed.), 2005, pp. 465 – 472.

[54] T.N. Lal, M. Schröder, T. Hinterberger, J. Weston, M. Bogdan, N. Birbaumer, and B. Schölkopf, *Support vector channel selection in bci*, IEEE Transactions on Biomedical Engineering **51** (2004), no. 6, 1003–1010.

[55] Neil D. Lawrence and Michael I. Jordan, *Semi-Supervised Learning*, ch. 8: Gaussian Processes and the Null-Category Noise Model, pp. 137–150, MIT University Press, 2006.

[56] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE **86** (1998), 2278–2324.

[57] E.C. Leuthardt, G. Schalk, Wolpaw J.R., J.G. Ojemann, and Moran D.W., *A brain-computer interface using electrocorticographic signals in humans*, Journal of Neural Engineering **1** (2004), no. 2, 63–71.

[58] D. D. Lewis, Y. Yang, T. Rose, and F. Li, *Rcv1: A new benchmark collection for text categorization research*, Journal of Machine Learning Research **5** (2004), 361–397.

[59] David G. Luenberger, *Optimization by Vector Space Methods*, John Wiley & Soms, Inc., 1997.

[60] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. Müller, *Fisher discriminant analysis with kernels*, Proceedings of IEEE Neural Networks for Signal Processing Workshop, 1999.

[61] Sebastian Mika, Gunnar Rätsch, and Klaus-Robert Müller, *A mathematical programming approach to the kernel fisher algorithm*, Advances in Neural Information Processing Systems (NIPS), 2000, pp. 591–597.

[62] Sebastian Mika, Gunnar Rätsch, and Klaus-Robert Müller, *A mathematical programming approach to the kernel fisher algorithm*, Advances in Neural Information Processing Systems, NIPS, 2000.

[63] J. del R. Millán, *On the need for on-line learning in brain-computer interfaces*, IDIAP-RR 30, IDIAP, Martigny, Switzerland, 2003, Published in "Proc. of the Int. Joint Conf. on Neural Networks", 2004.

[64] Tom M. Mitchell, *Machine Learning*, McGraw-Hill Education, 1997.

[65] John Shawe-Taylor Nello Cristianini, *Support vector machines and other kernel-based learning methods*, Cambridge University Press, 2000.

[66] K. B. Petersen and M. S. Pedersen, *The matrix cookbook*, 2005, Version 20051003.

[67] G. Pfurtscheller, C. Neuper, A. Schlogl, and K. Lugger, *Separability of eeg signals recorded during right and left motor imagery using adaptive autoregressive parameters*, IEEE Trans. Rehab. Eng. **6** (1998), no. 3, 316–324.

[68] Tomaso Poggio and Federico Girosi, *A theory of networks for approximation and learning*, Tech. Report AIM-1140, 1989.

[69] Tomaso Poggio and Federico Girosi, *Networks for Approximation and Learning*, Proceedings of the IEEE **78** (1990), no. 9, 1481–1497.

[70] Tomaso Poggio and Steve Smale, *The Mathematics of Learning: Dealing with Data*, Notices of the American Mathematical Society **50** (2003), no. 5, 537–554.

[71] Karl R. Popper, *The Logic of Scientific Discovery*, Routledge, 1959.

[72] Karl R. Popper, *Logik der Forschung*, 10 ed., Siebeck Verlag, Tübingen, 1994.

[73] C. E. Rasmussen and C. K.I. Williams, *Gaussian processes for machine learning*, Adaptive Computation and Machine Learning, The MIT Press, Cambridge, Massachusetts, 01 2006.

[74] Cornelius Raths, *Development of a brain-computer-interface approach based on covert attention to tactile stimuli*, 2006.

[75] Fred Rieke, Rob De Ruyter von Steveninck, David Warland, and William Bialek, *Spikes: Exploring the neural code*, MIT Press, 1997.

[76] F. Rosenblatt, *The perceptron: a probabilistic model for information storage and organization in the brain*, (1988), 89–114.

[77] S. Roweis and Z. Ghahramani, *A unifying review of linear Gaussian models*, Tech. report, California Institute of Technology, 1997.

[78] P. Ruján, *Playing billiard in version space*, Neural Computation **9** (1997), 99–122.

[79] B. Schölkopf, C. Burges, and V. Vapnik, *Incorporating invariances in support vector learning machines.*, Artificial Neural Networks — ICANN'96 (Berlin) (J.C. Vorbrüggen C. von der Malsburg, W. von Seelen and B. Sendhoff, eds.), vol. 1112, Springer Lecture Notes in Computer Science, 1996, pp. 47–52.

[80] B. Schölkopf, A.J. Smola, and K.R. Müller, *Kernel principal component analysis.*, 7th International Conference on Artificial Neural Networks, ICANN 97, Lausanne, Switzerland (Berlin) (M. Hasler W. Gerstner, A. Germond and J.-D. Nicoud, eds.), vol. 1327, Springer Lecture Notes in Computer Science, 1997, pp. 583–588.

[81] Bernhard Schölkopf and Alexander J. Smola, *Learning with kernels - support vector machines, regularization, optimization, and beyond*, MIT Press, Cambridge, MA, 2002.

[82] Dale Schuurmans, Finnegan Southey, Dana Wilkinson, and Yuhong Guo, *Metric-based approaches for semi-supervised regression and classification*, ch. 23, pp. 421–452, MIT Press, 2005.

[83] M. Seeger, *Input-dependent regularization of conditional density models*, Tech. report, Institute for Adaptive and Neural Computation, Edinburgh, 2000.

[84] C.E. Shannon, *A mathematical theory of communication*, Bell System Technical Journal **27** (1948), 379–423 and 623–656.

[85] John Shawe-Taylor and Nello Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, 2004.

[86] Jonathan Richard Shewchuk, *An introduction to the conjugate gradient method without the agonizing pain*, Tech. report, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1994.

[87] Lavi Shpigelman, Yoram Singer, Rony Paz, and Eilon Vaadia, *Embedding spiking neurons in inner-product spaces*, Adcances in Neural Information Processing Systems (NIPS), vol. 15, 2003.

[88] P. Simard, Y. LeCun, J. Denker, and B. Victorri, *An efficient algorithm for learning invariances in adaptive classifiers*, Proc. of International Conference on Pattern Recognition, 1992.

[89] A. J. Smola and B. Schölkopf, *A tutorial on support vector regression*, NeuroCOLT Technical Report NC-TR-98-030, Royal Holloway College, University of London, UK, 1998.

[90] P. Sollich, *Probabilistic interpretation and bayesian methods for support vector machines*, Proceedings of ICANN 1999, 1999.

[91] P. Sollich, *Probabilistic methods for support vector machines*, Advances in Neural Information Processing Systems, 1999.

[92] Florian Steinke, *Tutorial on Regularization and the Relation to Dynamical Systems using Finite Domains*.

[93] Stephen M. Stigler, *Gauss and the Invention of Least Squares*, Annals of Statistics **9** (1981), no. 3, 465–474.

[94] Erich E. Sutter, *The brain response interface: communication through visually-induced electrical brain responses*, Journal of Microcomputer Applications **15** (1992), no. 1, 31–45.

[95] J. A. K. Suykens, T. Van Gestel, J. Vandewalle, and B. De Moor, *A support vector macine formulation to pca analysis and its kernel version*, IEEE Transactions on Neural Networks **14** (2003), no. 2, 447–450.

[96] J. A. K. Suykens, T. Van Gestel, De Brabanter J., B. De Moor, and J. Vandewalle, *Least Squares Support Vector Machines*, World Scientific Pub. Co., Singapore, 2002.

[97] J. A. K. Suykens and J. Vandewalle, *Least squares support vector machine classifiers*, Neural Processing Letters **9** (1999), no. 3, 293–300.

[98] Vladimir Vapnik, *Transductive Inference and Semi-Supervised Learning*, ch. 24, pp. 454–472, MIT press, 2006.

[99] Vladimir N. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag Berlin, 1995.

[100] Vladimir N. Vapnik, *Statistical Learning Theory*, John Wiley & Sons Inc, 1998.

[101] U. von Luxburg, O. Bousquet, and B. Schölkopf, *A compression approach to support vector model selection*, The Journal of Machine Learning Research **5** (2004), 293–323.

[102] P.D. Welch, *The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms.*, IEEE Transactions on Audio Electroacoustics **15** (1967), 70–73.

[103] J. Weston, R. Collobert, F. Sinz, L. Bottou, and V. Vapnik, *Inference with the universum*, 06/25/ 2006, p. 127.

[104] J.R. Wolpaw, D. Flotzinger, G. Pfurtscheller, and D.J. McFarland, *Timing of eeg-based cursor control*, Journal of Clinical Neurophysiology **14** (1997), no. 6, 529–38.

[105] D. H. Wolpert, *On the connection between in-sample testing and generalization error*, Complex Systems **6** (1992), 47–94.

[106] D.H. Wolpert, *The Mathematics of Generalization*, ch. The relationship between PAC, the Statistical Physics framework, the Bayesian framework and the VC framework, Addison Wesley Longman, 1995.

[107] P. Zhong and M. Fukushima, *A new support vector algorithm*, Optimization Methods and Software **21** (2006), 359–372.

Index

μ -activity, 101
μ -rythm, 101
ε -insensitive loss, 41

action potentials, 113
active, 44
affine function, 38
alignment, 117
analytical induction principle, 12
Archai, 11
Aristotle, 11
auditory evoked potentials (AEP), 102

Bag of Words, 95
Bayes Point, 77
Bayes Rule, 7
Bayesian learning, 7
Billiard in Version Space, 125
brain computer interface, 99
brain computer interfaces, 88

capacity, 33
Cauchy sequence, 127
Cauchy-Schwarz inequality, 21
cluster assumption, 28
completion, 21
complexity, 33
Cone-Universum, 58
consistency, 33
contradiction, 37

data-dependent regularisation, 17
data-dependent regulariser, 27
David Hume, 12
deflated, 66
discriminative models, 28
don't care term, 49
dot product, 127
dual formulation, 44
dual space, 38
dual variables, 43

electroencephalogaphic (EEG), 100
empirical risk, 33
empirical risk minimisation, 33
endogenous BCI, 100
Entropy Point, 83
Epagoge, 11
event-related desynchronisation (ERD), 100, 101
exogenous BCI, 100

factor analysis, 65
feature map, 116
features, 18
frequentist learning, 8

Gaussian Processes, 23
generative model, 28
Gram matrix, 19, 44

Hamming distance, 50
hard margin SVM, 39
Hilbert space, 20
hinge loss, 40
homogeneous spikernel, 115

i.i.d, 24
independent component analysis (ICA), 104
induction principles, 9
inductive learning, 34
inference, 7
information regularisation, 28

inter-class variance, 66
intra-class variance, 66
inverse problem, 7

kernel, 19
kernel fisher discriminant analysis, 65
kernel fisher disriminant analysis, 58
kernel principal component analysis, 58, 65
kernel trick, 19
Kullback-Leibler divergence, 84

label, 7
Lagrange multipliers, 42
Lagrangian, 43
learning, 7
least squares SVM, 46
likelihood function, 7
linear function, 38
locked-in syndrome, 99

machine learning, 7
machine learning algorithm, 7
margin, 38
maximal margin, 38
maximum a posteriori, 7, 23
maximum entropy, 74
maximum likelihood, 7
maximum likelihood estimation, 23
maximum number of contradictions, 37, 74
MEG, 100
mixing matrix, 104
motor related potentials, 99

neural decoding, 88
No-Free-Lunch theorem, 13
noise model, 15
non parametric, 18

Occam's Razor, 16
One vs. Rest, 49
one-versus-one, 49
oriented Principal Component Analysis, 66

PAC-likelihood, 76
parameterized, 18
positive definite, 19
positive definite kernel, 19
posterior, 7
power spectral density (PSD), 108
prediction function, 77
prediction strategy, 77
primal variables, 43
Principal Component Analysis (PCA), 65
principal components, 65
prior, 7, 23

rate coding, 114
Rayleigh Quotient, 66
reconstruction error, 65
regularisation, 9, 22
repoducing property, 20
representer theorem, 22
reproducing kernel Hilbert space (RKHS), 20
reproducing property, 21
Ridge Regression, 48

semi-supervised learning, 17
Sextus Empiricus, 11
shortest path, 117
slack variables, 39
slow cortical potentials, 99
slow motor related potential, 100
soft projection, 63
span, 127
spikes, 113
statistical learning, 8
structural risk minimisation, 32, 34
Support Vector Machine, 34
support vectors, 45
synthetic induction principle, 12

target values, 7
temporal coding, 115
the Universum, 35
training data, 7
training examples, 7

Universum, 35
universum examples, 18

VC dimension, 34
VC entropy, 36
VC theory (Vapnik Chervonenkis), 32
version space, 74
virtual support vector, 93

William of Occam, 10

zero-one loss, 10, 76