Journal of Machine Learning Research 1 (2001) 245–279 Submitted 02/01; Published 8/01

Bayes Point Machines

Ralf Herbrich RHERB@MICROSOFT.COM
Microsoft Research, St George House, 1 Guildhall Street, CB2 3NH Cambridge, United Kingdom

Thore Graepel GURU@CS.TU-BERLIN.DE
Technical University of Berlin, Franklinstr. 28/29, 10587 Berlin, Germany

Colin Campbell C.CAMPBELL@BRISTOL.AC.UK
Department of Engineering Mathematics, Bristol University, BS8 1TR Bristol, United Kingdom

Editor: Christopher K. I. Williams

Abstract

Kernel-classifiers comprise a powerful class of non-linear decision functions for binary classification. The support vector machine is an example of a learning algorithm for kernel classifiers that singles out the consistent classifier with the largest margin, i.e. minimal real-valued output on the training sample, within the set of consistent hypotheses, the so-called *version space*. We suggest the *Bayes point machine* as a well-founded improvement which approximates the Bayes-optimal decision by the centre of mass of version space. We present two algorithms to stochastically approximate the centre of mass of version space: a billiard sampling algorithm and a sampling algorithm based on the well known perceptron algorithm. It is shown how both algorithms can be extended to allow for soft-boundaries in order to admit training errors. Experimentally, we find that — for the zero training error case — Bayes point machines consistently outperform support vector machines on both surrogate data and real-world benchmark data sets. In the soft-boundary/soft-margin case, the improvement over support vector machines is shown to be reduced. Finally, we demonstrate that the real-valued output of single Bayes points on novel test points is a valid *confidence* measure and leads to a steady decrease in generalisation error when used as a rejection criterion.

1. Introduction

Kernel machines have recently gained a lot of attention due to the popularisation of the support vector machine (Vapnik, 1995) with a focus on classification and the revival of Gaussian processes for regression (Williams, 1999). Subsequently, support vector machines have been modified to handle regression (Smola, 1998) and Gaussian processes have been adapted to the problem of classification (Williams and Barber, 1998; Opper and Winther, 2000). Both schemes essentially work in the same function space that is characterised by kernels and covariance functions, respectively. Whilst the formal similarity of the two methods is striking, the underlying paradigms of inference are very different. The support vector machine was inspired by results from statistical/PAC learning theory while Gaussian processes are usually considered in a Bayesian framework. This ideological clash can be viewed as a continuation in machine learning of the by now classical disagreement between Bayesian and frequentistic statistics (Aitchison, 1964). With regard to algorithmics the two schools of thought appear to favour two different methods of learning and predicting: the support vector community — as a consequence of the formulation of the support vector machine as a quadratic programming problem — focuses on learning as optimisation while the Bayesian community favours sampling schemes based on the Bayesian posterior. Of course there exists a strong relationship between the two ideas, in particular with the Bayesian maximum a posteriori (MAP) estimator being the solution of an optimisation problem.

In practice, optimisation based algorithms have the advantage of a unique, deterministic solution and the availability of the cost function as an indicator of the quality of the solution. In contrast, Bayesian algorithms based on sampling and voting are more flexible and enjoy the so-called “anytime” property, providing a relatively good solution at any point in time. Often, however, they suffer from the computational costs of sampling the Bayesian posterior.

In this paper we present the Bayes point machine as an approximation to Bayesian inference for linear classifiers in kernel space. In contrast to the Gaussian process viewpoint we do not define a Gaussian prior on the length $\|\mathbf{w}\|$ of the weight vector. Instead, we only consider weight vectors of length $\|\mathbf{w}\| = 1$ because it is only the spatial direction of the weight vector that matters for classification. It is then natural to define a uniform prior on the resulting ball-shaped hypothesis space. Hence, we determine the centre of mass of the resulting posterior that is uniform in version space, i.e. in the zero training error region. It should be kept in mind that the centre of mass is merely an approximation to the real Bayes point from which the name of the algorithm was derived. In order to estimate the centre of mass we suggest both a dynamic system called a *kernel billiard* and an approximative method that uses the perceptron algorithm trained on permutations of the training sample. The latter method proves to be efficient enough to make the Bayes point machine applicable to large data sets.

An additional insight into the usefulness of the centre of mass comes from the statistical mechanics approach to neural computing where the generalisation error for Bayesian learning algorithms has been calculated for the case of randomly constructed and unbiased patterns $\mathbf{x}$ (Opper and Haussler, 1991). Thus if ζ is the number of training examples per weight and ζ is large, the generalisation error of the centre of mass scales as $0.44/\zeta$ whereas scaling with ζ is poorer for the solutions found by the linear support vector machine (scales as $0.50/\zeta$; see Opper and Kinzel, 1995), Adaline (scales as $0.24/\sqrt{\zeta}$; see Opper et al., 1990) and other approaches.

Of course many of the viewpoints and algorithms presented in this paper are based on extensive previous work carried out by numerous authors in the past. In particular it seems worthwhile to mention that linear classifiers have been studied intensively in two rather distinct communities: The machine learning community and the statistical physics community. While it is beyond the scope of this paper to review the entire history of the field we would like to emphasise that our geometrical viewpoint as expressed later in the paper has been inspired by the very original paper "Playing billiard in version space" by P. Ruján (Ruján, 1997). Also, in that paper the term "Bayes point" was coined and the idea of using a billiard-like dynamical system for uniform sampling was introduced. Both we (Herbrich et al., 1999a,b, 2000a) and Ruján and Marchand (2000) independently generalised the algorithm to be applicable in kernel space. Finally, following a theoretical suggestion of Watkin (1993) we were able to scale up the Bayes point algorithm to large data sets by using different perceptron solutions from permutations of the training sample.

The paper is structured as follows: In the following section we review the basic ideas of Bayesian inference with a particular focus on classification learning. Along with a discussion about the optimality of the Bayes classification strategy we show that for the special case of linear classifiers in feature space the centre of mass of all consistent classifiers is arbitrarily close to the Bayes point (with increasing training sample size) and can be efficiently estimated in the linear span of the training data. Moreover, we give a geometrical picture of support vector learning in feature space which reveals that the support vector machine can be viewed as an approximation to the Bayes point machine. In Section 3 we present two algorithms for the estimation of the centre of mass of version space — one exact method and an approximate method tailored for large training samples. An extensive list of experimental results is presented in Section 4, both on small machine learning benchmark datasets as well as on large scale datasets from the field of handwritten digit recognition. In Section 5 we summarise the results and discuss some theoretical extensions of the method presented. In order to unburden the main text, the lengthy proofs as well as the pseudocode have been relegated to the appendix.

We denote n–tuples by italic bold letters (e.g. $\boldsymbol{x} = (x_1, \ldots, x_n)$), vectors by roman bold letters (e.g. $\mathbf{x}$), random variables by sans serif font (e.g. X) and vector spaces by calligraphic capitalised letters (e.g. $\mathcal{X}$). The symbols $\mathbf{P}, \mathbf{E}$ and $\mathbf{I}$ denote a probability measure, the expectation of a random variable and the indicator function, respectively.

2. A Bayesian Consideration of Learning

In this section we would like to revisit the Bayesian approach to learning (see Buntine, 1992; MacKay, 1991; Neal, 1996; Bishop, 1995, for a more detailed treatment). Suppose we are given a training sample $z = (x,y) = ((x_1,y_1),\ldots,(x_m,y_m)) \in (\mathcal{X}\times\mathcal{Y})^m$ of size m drawn iid from an unknown distribution $\mathbf{P}_\mathsf{Z} = \mathbf{P}_\mathsf{XY}$. Furthermore, assume we are given a *fixed* set $\mathcal{H} \subseteq \mathcal{Y}^\mathcal{X}$ of functions $h : \mathcal{X} \to \mathcal{Y}$ referred to as *hypothesis space*. The task of learning is then to find the function h^* which performs best on new yet unseen patterns $z = (x,y)$ drawn according to $\mathbf{P}_\mathsf{XY}$.

Definition 1 (Learning Algorithm) *A* (deterministic) learning algorithm $\mathcal{A} : \bigcup_{m=1}^{\infty} \mathcal{Z}^m \to \mathcal{Y}^\mathcal{X}$ *is a mapping from training samples z of arbitrary size $m \in \mathbb{N}$ to functions from $\mathcal{X}$ to $\mathcal{Y}$. The image of $\mathcal{A}$, i.e. $\{\mathcal{A}(z) \mid z \in \mathcal{Z}^m\} \subseteq \mathcal{Y}^\mathcal{X}$, is called the* effective hypothesis space $\mathcal{H}_{\mathcal{A},m}$ *of the learning algorithm $\mathcal{A}$ for the training sample size $m \in \mathbb{N}$. If there exists a hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^\mathcal{X}$ such that for every training sample size $m \in \mathbb{N}$ we have $\mathcal{H}_{\mathcal{A},m} \subseteq \mathcal{H}$ we shall omit the indices on $\mathcal{H}$.*

In order to assess to quality of a function $h \in \mathcal{H}$ we assume the existence of a *loss function* $l : \mathcal{Y}\times\mathcal{Y} \to \mathbb{R}^+$. The loss $l(y,y\prime) \in \mathbb{R}^+$ is understood to measure the incurred cost when predicting y while the true output was y'. Hence we always assume that for all $y \in \mathcal{Y}$, $l(y,y) = 0$. A typical loss function for classification is the so called *zero-one* loss l_{0-1} defined as follows.

Definition 2 (Zero-One Loss) *Given a fixed output space $\mathcal{Y}$, the* zero-one *loss is defined by*

$$l_{0-1}(y,y') := \mathbf{I}_{y \neq y'} .$$

Based on the concept of a loss l, let us introduce several quality measures for hypotheses $h \in \mathcal{H}$.

Definition 3 (Generalisation and Training Error) *Given a probability measure $\mathbf{P}_\mathsf{XY}$ and a loss $l : \mathcal{Y}\times\mathcal{Y} \to \mathbb{R}^+$ the* generalisation error $R[h]$ *of a function $h : \mathcal{X} \to \mathcal{Y}$ is defined by*

$$R[h] := \mathbf{E}_\mathsf{XY}[l(h(\mathsf{X}),\mathsf{Y})] .$$

Given a training sample $z = (x,y) \in (\mathcal{X}\times\mathcal{Y})^m$ of size m and a loss $l : \mathcal{Y}\times\mathcal{Y} \to \mathbb{R}^+$ the training error $R_{\text{emp}}[h,z]$ *of a function $h : \mathcal{X} \to \mathcal{Y}$ is given by*

$$R_{\text{emp}}[h,z] := \frac{1}{m}\sum_{i=1}^{m} l(h(x_i),y_i) .$$

Clearly, only the generalisation error $R[h]$ is appropriate to capture the performance of a *fixed* classifier $h \in \mathcal{H}$ on new patterns $z = (x,y)$. Nonetheless, we shall see that the training error plays a crucial role as it provides an estimate of the generalisation error based on the training sample.

Definition 4 (Generalisation Error of Algorithms) *Suppose we are given a fixed learning algorithm $\mathcal{A} : \bigcup_{m=1}^{\infty} \mathcal{Z}^m \to \mathcal{Y}^\mathcal{X}$. Then for any fixed training sample size $m \in \mathbb{N}$ the* generalisation error $R_m[\mathcal{A}]$ *of $\mathcal{A}$ is defined by*

$$R_m[\mathcal{A}] := \mathbf{E}_{\mathsf{Z}^m}[R[\mathcal{A}(\mathbf{Z})]] ,$$

that is, the expected generalisation error of the hypotheses found by the algorithm.

Note that for any loss function $l : \mathcal{Y}\times\mathcal{Y} \to \mathbb{R}^+$ a small generalisation error $R_m[\mathcal{A}]$ of the algorithm $\mathcal{A}$ guarantees a small generalisation error for most randomly drawn training samples z because by Markov's inequality we have for $\varepsilon > 0$,

$$\mathbf{P}_{\mathsf{Z}^m}(R[\mathcal{A}(\mathbf{Z})] > \varepsilon \cdot \mathbf{E}_{\mathsf{Z}^m}[R[\mathcal{A}(\mathbf{Z})]]) \leq \frac{1}{\varepsilon} .$$

Hence we can view $R_m[\mathcal{A}]$ also as a performance measure of $\mathcal{A}$'s hypotheses for randomly drawn training samples z. Finally, let us consider a probability measure $\mathbf{P}_\mathsf{H}$ over the space of all possible mappings from $\mathcal{X}$ to $\mathcal{Y}$. Then, the *average generalisation error of a learning algorithm* $\mathcal{A}$ is defined as follows.

Definition 5 (Average Generalisation Error of Algorithms) *Suppose we are given a fixed learning algorithm* $\mathcal{A}:\bigcup_{m=1}^{\infty}\mathcal{Z}^m\to\mathcal{Y}^{\mathcal{X}}$. *Then for each fixed training sample size* $m\in\mathbb{N}$ *the* average generalisation error $\overline{R}_m[\mathcal{A}]$ of $\mathcal{A}$ *is defined by*

$$\overline{R}_m[\mathcal{A}]:=\mathbf{E}_{\mathsf{H}}\left[\mathbf{E}_{\mathsf{Z}^m|\mathsf{H}=h}\left[\mathbf{E}_{\mathsf{X}}\left[\mathbf{E}_{\mathsf{Y}|\mathsf{X}=x,\mathsf{H}=h}\left[l\left(\left(\mathcal{A}\left(\mathbf{Z}\right)\right)(x),\mathsf{Y}\right)\right]\right]\right]\right], \tag{1}$$

that is, the average performance of the algorithm's $\mathcal{A}$ *solution learned over the random draw of training samples* and *target hypotheses.*

The average generalisation error is the standard measure of performance of an algorithm $\mathcal{A}$ if we have little knowledge about the potential function h^* that labels all our data expressed via $\mathbf{P}_{\mathsf{H}}$. Then, the measure (1) averages out our ignorance about the unknown h^* thus considering performance of $\mathcal{A}$ *on average*.

There is a noticeable relation between $R_m[\mathcal{A}]$ and $\overline{R}_m[\mathcal{A}]$ if we assume that given a measure $\mathbf{P}_{\mathsf{H}}$, the conditional distribution of outputs y given x is governed by[1]

$$\mathbf{P}_{\mathsf{Y}|\mathsf{X}=x}(y)=\mathbf{P}_{\mathsf{H}}\left(\mathsf{H}(x)=y\right). \tag{2}$$

Under this condition we have that

$$\overline{R}_m[\mathcal{A}]=R_m[\mathcal{A}].$$

This result, however, is not too surprising taking into account that under the assumption (2) the measure $\mathbf{P}_{\mathsf{H}}$ fully encodes the unknown relationship between inputs x and outputs y.

2.1 The Bayesian Solution

In the Bayesian framework we are not simply interested in $h^*:=\operatorname{argmin}_{h\in\mathcal{H}}R[h]$ itself but in our *knowledge* or *belief* in h^*. To this end, Bayesians use the concept of *prior* and *posterior belief*, i.e. the knowledge of h^* *before* having seen any data and *after* having seen the data — which in the current case is our training sample z. It is well known that under consistency rules known as Cox's axioms (Cox, 1946) beliefs can be mapped onto probability measures $\mathbf{P}_{\mathsf{H}}$. Under these rather plausible conditions the only consistent way to transfer prior belief $\mathbf{P}_{\mathsf{H}}$ into posterior belief $\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=z}$ is therefore given by Bayes' theorem:

$$\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=z}(h)=\frac{\mathbf{P}_{\mathsf{Z}^m|\mathsf{H}=h}(z)}{\mathbf{E}_{\mathsf{H}}\left[\mathbf{P}_{\mathsf{Z}^m|\mathsf{H}=h}(z)\right]}\cdot\mathbf{P}_{\mathsf{H}}(h)=\frac{\mathbf{P}_{\mathsf{Y}^m|\mathsf{X}^m=x,\mathsf{H}=h}(y)}{\mathbf{E}_{\mathsf{H}}\left[\mathbf{P}_{\mathsf{Y}^m|\mathsf{X}^m=x,\mathsf{H}=h}(y)\right]}\cdot\mathbf{P}_{\mathsf{H}}(h). \tag{3}$$

The second expression is obtained by noticing that

$$\mathbf{P}_{\mathsf{Z}^m|\mathsf{H}=h}(z)=\mathbf{P}_{\mathsf{Y}^m|\mathsf{X}^m=x,\mathsf{H}=h}(y)\,\mathbf{P}_{\mathsf{X}^m|\mathsf{H}=h}(x)=\mathbf{P}_{\mathsf{Y}^m|\mathsf{X}^m=x,\mathsf{H}=h}(y)\,\mathbf{P}_{\mathsf{X}^m}(x)$$

because hypotheses do not have an influence on the generation of patterns. Based on a given loss function l we can further decompose the first term of the numerator of (3) — known as the likelihood of h. Let us assume that the probability of a class y given an instance x and an hypothesis h is inverse proportional to the exponential of the loss incurred by h on x. Thus we obtain

$$\begin{aligned}\mathbf{P}_{\mathsf{Y}|\mathsf{X}=x,\mathsf{H}=h}(y) &= \frac{\exp\left(-\beta\cdot l\left(h(x),y\right)\right)}{\sum\limits_{y'\in\mathcal{Y}}\exp\left(-\beta\cdot l\left(h(x),y'\right)\right)}=\frac{1}{C(x)}\exp\left(-\beta\cdot l\left(h(x),y\right)\right)\\ &= \begin{cases}\frac{1}{1+\exp(-\beta)} & \text{if } l\left(h(x),y\right):=l_{0-1}\left(h(x),y\right)=0\\ \frac{\exp(-\beta)}{1+\exp(-\beta)} & \text{if } l\left(h(x),y\right):=l_{0-1}\left(h(x),y\right)=1\end{cases},\end{aligned} \tag{4}$$

where $C(x)$ is a normalisation constant which in the case of the zero-one loss l_{0-1} is independent[2] of x and $\beta\geq 0$ controls the assumed level of noise. Note that the loss used in the exponentiated loss likelihood function

1. In fact, it already suffices to assume that $\mathbf{E}_{\mathsf{Y}|\mathsf{X}=x}\left[l(y,\mathsf{Y})\right]=\mathbf{E}_{\mathsf{H}}\left[l(y,\mathsf{H}(x))\right]$, i.e. the prior correctly models the conditional distribution of the classes as far as the fixed loss is concerned.
2. Note that for loss functions with real-valued arguments this need not be the case which makes a normalisation independent of x quite intricate (see Sollich, 2000, for a detailed treatment).

is not to be confused with the decision-theoretic loss used in the Bayesian framework, which is introduced only after a posterior has been obtained in order to reach a risk optimal decision.

Definition 6 (PAC Likelihood) *Suppose we are given an arbitrary loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$. Then, we call the function*

$$\mathbf{P}_{\mathsf{Y}|\mathsf{X}=x,\mathsf{H}=h}(y) := \mathbf{I}_{y=h(x)}\,, \tag{5}$$

of h the PAC likelihood for *h. Note that* (5) *is the limiting case of* (4) *for $\beta \to \infty$.*

Assuming the PAC likelihood it immediately follows that for any prior belief $\mathbf{P}_{\mathsf{H}}$ the posterior belief $\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=z}$ simplifies to

$$\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=z}(h) = \begin{cases} \frac{\mathbf{P}_{\mathsf{H}}(h)}{\mathbf{P}_{\mathsf{H}}(V(z))} & \text{if } h \in V(z) \\ 0 & \text{if } h \notin V(z) \end{cases}\,, \tag{6}$$

where the version space $V(z)$ is defined as follows (see Mitchell, 1977, 1982).

Definition 7 (Version Space) *Given an hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ and a training sample $z = (x, y) \in (\mathcal{X} \times \mathcal{Y})^m$ of size $m \in \mathbb{N}$ the* version space *$V(z) \subseteq \mathcal{H}$ is defined by*

$$V(z) := \left\{ h \in \mathcal{H} \mid \forall i \in \{1, \ldots, m\} : h(x_i) = y_i \right\}.$$

Since all information contained in the training sample z is used to update the prior $\mathbf{P}_{\mathsf{H}}$ by equation (3) all that will be used to classify a novel test point x is the *posterior belief* $\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=z}$.

2.2 The Bayes Classification Strategy

In order to classify a new test point x, for each class y the *Bayes classification strategy*[3] determines the loss incurred by each hypothesis $h \in \mathcal{H}$ applied to x and weights it according to its posterior probability $\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=z}(h)$. The final decision is made for the class $y \in \mathcal{Y}$ that achieves the minimum expected loss, i.e.

$$Bayes_z(x) := \operatorname*{argmin}_{y \in \mathcal{Y}} \mathbf{E}_{\mathsf{H}|\mathsf{Z}^m=z}\left[l(\mathsf{H}(x), y)\right]. \tag{7}$$

This strategy has the following appealing property.

Theorem 8 (Optimality of the Bayes Classification Strategy) *Suppose we are given a fixed hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$. Then, for any training sample size $m \in \mathbb{N}$, for any symmetric loss $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$, for any two measures $\mathbf{P}_{\mathsf{H}}$ and $\mathbf{P}_{\mathsf{X}}$, among all learning algorithms the Bayes classification strategy $Bayes_z$ given by* (7) *minimises the average generalisation error $\overline{R}_m[Bayes_z]$ under the assumption that for each h with $\mathbf{P}_{\mathsf{H}}(h) > 0$*

$$\forall y \in \mathcal{Y} : \forall x \in \mathcal{X} : \qquad \mathbf{E}_{\mathsf{Y}|\mathsf{X}=x,\mathsf{H}=h}\left[l(y, \mathsf{Y})\right] = l(y, h(x))\,. \tag{8}$$

Proof Let us consider a fixed learning algorithm $\mathcal{A}$. Then it holds true that

$$\begin{aligned} \overline{R}_m[\mathcal{A}] &= \mathbf{E}_{\mathsf{H}}\left[\mathbf{E}_{\mathsf{Z}^m|\mathsf{H}=h}\left[\mathbf{E}_{\mathsf{X}}\left[\mathbf{E}_{\mathsf{Y}|\mathsf{X}=x,\mathsf{H}=h}\left[l((\mathcal{A}(\mathbf{Z}))(x), \mathsf{Y})\right]\right]\right]\right] \\ &= \mathbf{E}_{\mathsf{X}}\left[\mathbf{E}_{\mathsf{H}}\left[\mathbf{E}_{\mathsf{Z}^m|\mathsf{H}=h}\left[\mathbf{E}_{\mathsf{Y}|\mathsf{X}=x,\mathsf{H}=h}\left[l((\mathcal{A}(\mathbf{Z}))(x), \mathsf{Y})\right]\right]\right]\right] \\ &= \mathbf{E}_{\mathsf{X}}\left[\mathbf{E}_{\mathsf{Z}^m}\left[\mathbf{E}_{\mathsf{H}|\mathsf{Z}^m=z}\left[\mathbf{E}_{\mathsf{Y}|\mathsf{X}=x,\mathsf{H}=h}\left[l((\mathcal{A}(\mathbf{Z}))(x), \mathsf{Y})\right]\right]\right]\right] \\ &= \mathbf{E}_{\mathsf{X}}\left[\mathbf{E}_{\mathsf{Z}^m}\left[\mathbf{E}_{\mathsf{H}|\mathsf{Z}^m=z}\left[l((\mathcal{A}(\mathbf{Z}))(\mathsf{X}), \mathsf{H}(\mathsf{X}))\right]\right]\right], \end{aligned} \tag{9}$$

where we exchanged the order of expectations over $\mathcal{X}$ in the second line, applied the theorem of repeated integrals (see, e.g. Feller, 1966) in the third line and finally used (8) in the last line. Using the symmetry of the loss function, the inner-most expression of (9) is minimised by the Bayes classification strategy (7)

3. The reason we do not call this mapping from $\mathcal{X}$ to $\mathcal{Y}$ a classifier is that the resulting mapping is (in general) not within the hypothesis space considered beforehand.

for any possible training sample z and any possible test point x. Hence, (7) minimises the whole expression which proves the theorem. ■

In order to enhance the understanding of this result let us consider the simple case of $l = l_{0-1}$ and $\mathcal{Y} = \{-1,+1\}$. Then, given a particular classifier $h \in \mathcal{H}$ having non-zero prior probability $\mathbf{P}_{\mathsf{H}}(h) > 0$, by assumption (8) we require that the conditional distribution of classes y given x is delta peaked at $h(x)$ because

$$\begin{aligned}\mathbf{E}_{\mathsf{Y}|\mathsf{X}=x,\mathsf{H}=h}\left(l_{0-1}(y,\mathsf{Y})\right) &= l_{0-1}(y,h(x)) \,, \\ \mathbf{P}_{\mathsf{Y}|\mathsf{X}=x,\mathsf{H}=h}(-y) &= \mathbf{I}_{y\neq h(x)} \,, \\ \mathbf{P}_{\mathsf{Y}|\mathsf{X}=x,\mathsf{H}=h}(y) &= \mathbf{I}_{h(x)=y} \,.\end{aligned}$$

Although for a fixed $h \in \mathcal{H}$ drawn according to $\mathbf{P}_{\mathsf{H}}$ we *do not know* that $Bayes_z$ achieves the smallest generalisation error $R[Bayes_z]$ we can guarantee that on average over the random draw of h's the Bayes classification strategy is superior. In fact, the optimal classifier for a fixed $h \in \mathcal{H}$ is simply h itself[4] and in general $Bayes_z(x) \neq h(x)$ for at least a few $x \in \mathcal{X}$.

2.3 The Bayes Point Algorithm

Although the Bayes classification strategy is *on average* the optimal strategy to perform when given limited amount of training data z, it is computationally very demanding as it requires the evaluation of $\mathbf{P}_{\mathsf{H}|\mathsf{Z}^m=z}(l(\mathsf{H}(x),y))$ for each possible y at each new test point x (Graepel et al., 2000). The problem arises because the Bayes classification strategy does not correspond to any one single classifier $h \in \mathcal{H}$. One way to tackle this problem is to require the classifier $\mathcal{A}(z)$ learned from any training sample z to lie within a *fixed* hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ containing functions $h \in \mathcal{H}$ whose evaluation at a particular test point x can be carried out efficiently. Thus if it is additionally required to limit the possible solution of a learning algorithm to a given hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$, we can in general only hope to approximate $Bayes_z$.

Definition 9 (Bayes Point Algorithm) *Suppose we are given a fixed hypothesis space $\mathcal{H} \subseteq \mathcal{X}^{\mathcal{Y}}$ and a fixed loss $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$. Then, for any two measures $\mathbf{P}_{\mathsf{X}}$ and $\mathbf{P}_{\mathsf{H}}$, the* Bayes point algorithm *$\mathcal{A}_{\mathrm{bp}}$ is given by*

$$\mathcal{A}_{\mathrm{bp}}(z) := \operatorname*{argmin}_{h\in\mathcal{H}} \mathbf{E}_{\mathsf{X}}\left[\mathbf{E}_{\mathsf{H}|\mathsf{Z}^m=z}\left[l(h(\mathsf{X}),\mathsf{H}(\mathsf{X}))\right]\right] ,$$

that is, for each training sample $z \in \mathcal{Z}^m$ the Bayes point algorithm chooses the classifier $h_{\mathrm{bp}} := \mathcal{A}_{\mathrm{bp}}(z) \in \mathcal{H}$ that mimics best the Bayes classification strategy (7) *on average over randomly drawn test points. The classifier $\mathcal{A}_{\mathrm{bp}}(z)$ is called the* Bayes point.

Assuming the correctness of the model given by (8) we furthermore remark that the Bayes point algorithm $\mathcal{A}_{\mathrm{bp}}$ is the best approximation to the Bayes classification strategy (7) in terms of the average generalisation error, i.e. measuring the distance of the learning algorithm $\mathcal{A}$ for $\mathcal{H}$ using the distance $\|\mathcal{A} - Bayes\| = \overline{R}_m[\mathcal{A}] - \overline{R}_m[Bayes]$. In this sense, for a fixed training sample z we can view the *Bayes point* h_{bp} as a projection of $Bayes_z$ into the hypothesis space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$.

The difficulty with the Bayes point algorithm, however, is the need to know the input distribution $\mathbf{P}_{\mathsf{X}}$ for the determination of the hypothesis learned from z. This somehow limits the applicability of the algorithm as opposed to the Bayes classification strategy which requires only broad prior knowledge about the underlying relationship expressed via some prior belief $\mathbf{P}_{\mathsf{H}}$.

4. It is worthwhile mentioning that the only information to be used in any classification strategy is the training sample z and the prior $\mathbf{P}_{\mathsf{H}}$. Hence it is impossible to *detect* which classifier $h \in \mathcal{H}$ labels a fixed m–tuple x only on the basis of the m labels y observed on the training sample. Thus, although we might be lucky in guessing *h for a fixed* $h \in \mathcal{H}$ and $z \in \mathcal{Z}^m$ we cannot do better than the Bayes classification strategy $Bayes_z$ when considering the average performance — the average being taken over the random choice of the classifiers and the training samples z.

2.3.1 THE BAYES POINT FOR LINEAR CLASSIFIERS

We now turn our attention to the special case of linear classifiers where we assume that N measurements of the objects x are taken by features $\phi_i : \mathcal{X} \to \mathbb{R}$ thus forming a (vectorial) feature map $\phi : \mathcal{X} \to \mathcal{K} \subseteq \ell_2^N = (\phi_1(x), \ldots, \phi_N(x))'$. Note that by this formulation the special case of vectorial objects x is automatically taken care of by the identity map $\phi(x) = x$. For notational convenience we use the shorthand notation[5] $\mathbf{x}$ for $\phi(x)$ such that $\langle \mathbf{x}, \mathbf{w} \rangle := \sum_{i=1}^{N} \phi_i(x) w_i$. Hence, for a fixed mapping ϕ the hypothesis space is given by

$$\mathcal{H} := \left\{ x \mapsto \operatorname{sign}(\langle \mathbf{x}, \mathbf{w} \rangle) \;\middle|\; \mathbf{w} \in \mathcal{W} \right\}, \qquad \mathcal{W} := \{ \mathbf{w} \in \mathcal{K} \mid \|\mathbf{w}\| = 1 \}. \tag{10}$$

As each hypothesis $h_{\mathbf{w}}$ is uniquely defined by its *weight vector* $\mathbf{w}$ we shall in the following consider prior beliefs P_{W} over $\mathcal{W}$, i.e. possible weight vectors (of unit length), in place of priors P_{H}. By construction, the output space is $\mathcal{Y} = \{-1, +1\}$ and we furthermore consider the special case of $l = l_{0-1}$ as defined by Definition 2. If we assume that the input distribution is spherically Gaussian in the feature space $\mathcal{K}$ of dimensionality $d = \dim(\mathcal{K})$, i.e.

$$\mathbf{f}_{\mathsf{X}}(\mathbf{x}) = \frac{1}{\pi^{\frac{d}{2}}} \exp\left(- \|\mathbf{x}\|^2 \right), \tag{11}$$

then we find that the centre of mass

$$\mathbf{w}_{\mathrm{cm}} = \frac{\mathbf{E}_{\mathsf{W}|\mathsf{Z}^m = z}[\mathsf{W}]}{\left\| \mathbf{E}_{\mathsf{W}|\mathsf{Z}^m = z}[\mathsf{W}] \right\|} \tag{12}$$

is a very good approximation to the Bayes point $\mathbf{w}_{\mathrm{bp}}$ and converges towards $\mathbf{w}_{\mathrm{bp}}$ if the posterior belief $\mathsf{P}_{\mathsf{W}|\mathsf{Z}^m = z}$ becomes sharply peaked (for a similar result see Watkin, 1993).

Theorem 10 (Optimality of the Centre of Mass) *Suppose we are given a fixed mapping* $\phi : \mathcal{X} \to \mathcal{K} \subseteq \ell_2^N$. *Then, for all* $m \in \mathbb{N}$, *if* P_{X} *possesses the density* (11) *and the prior belief is correct, i.e.* (8) *is valid, the average generalisation error of the centre of mass as given by* (12) *always fulfils*

$$\left| \overline{R}_m[\mathcal{A}_{\mathrm{cm}}] - \overline{R}_m[\mathcal{A}_{\mathrm{bp}}] \right| \leq \mathbf{E}_{\mathsf{Z}^m}[\kappa(\varepsilon(\mathsf{Z}))],$$

where

$$\kappa(\varepsilon) := \begin{cases} \frac{\arccos(\varepsilon)}{\pi} - \frac{1-\varepsilon}{2} & \text{if } \varepsilon < 0.23 \\ 0.11 & \text{otherwise} \end{cases},$$

and

$$\varepsilon(z) := \min_{\mathbf{w} : \mathsf{P}_{\mathsf{W}|\mathsf{Z}^m = z}(\mathbf{w}) > 0} |\langle \mathbf{w}_{\mathrm{cm}}, \mathbf{w} \rangle|.$$

The lengthy proof of this theorem is given in Appendix A.1. The interesting fact to note about this result is that $\lim_{\varepsilon \to 1} \kappa(\varepsilon) = 0$ and thus whenever the prior belief P_{W} is not vanishing for some $\mathbf{w}$,

$$\lim_{m \to \infty} \mathbf{E}_{\mathsf{Z}^m}[\kappa(\varepsilon(\mathsf{Z}))] = 0,$$

because for increasing training sample size the posterior is sharply peaked at the weight vector labelling the data[6]. This shows that for increasing training sample size the centre of mass (under the posterior $\mathsf{P}_{\mathsf{W}|\mathsf{Z}^m = z}$) is a good approximation to the optimal projection of the Bayes classification strategy — the *Bayes point*. Henceforth, any algorithm which aims at returning the centre of mass under the posterior $\mathsf{P}_{\mathsf{W}|\mathsf{Z}^m = z}$ is called a *Bayes point machine*. Note that in the case of the PAC likelihood as defined in Definition 6 the centre of mass under the posterior $\mathsf{P}_{\mathsf{W}|\mathsf{Z}^m = z}$ coincides with the centre of mass of version space (see Definition 7).

5. This should not be confused with x which denotes the sample $(x_1, \ldots, x_m)$ of training objects.
6. This result is a slight generalisation of the result in Watkin (1993) which only proved this to be true for the uniform prior P_{W}.

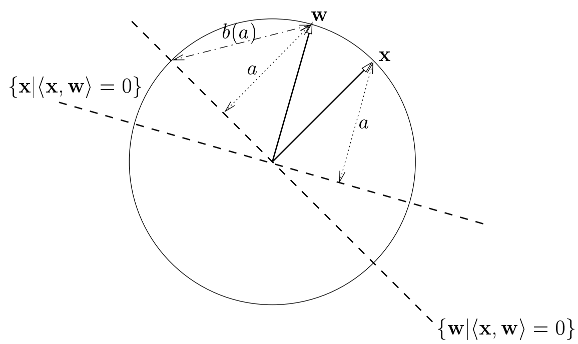


Figure 1: Shown is the margin $a = \gamma_x(\mathbf{w}) = \langle \mathbf{x}, \mathbf{w} \rangle$ under the assumption that $\|\mathbf{w}\| = \|\mathbf{x}\| = 1$. At the same time, a (length of the dotted line) equals the distance of $\mathbf{x}$ from the hyperplane $\{\mathbf{x} \mid \langle \mathbf{x}, \mathbf{w} \rangle = 0\}$ (dashed line) as well as the distance of the weight vector $\mathbf{w}$ from the hyperplane $\{\mathbf{w} \mid \langle \mathbf{x}, \mathbf{w} \rangle = 0\}$ (dashed line). Note, however, that the Euclidean distance of $\mathbf{w}$ from the separating boundary $\left\{\mathbf{w} \in \mathcal{W} \mid \langle \mathbf{x}, \mathbf{w} \rangle = 0\right\}$ equals $b(a)$ where b is a strictly monotonic function of its argument.

2.4 A (Pseudo) Bayesian Derivation of the Support Vector Machine

In this section we would like to show that the well known support vector machine (Boser et al., 1992; Cortes, 1995; Vapnik, 1995) can also be viewed as an approximation to the centre of mass of version space $V(z)$ in the noise free scenario, i.e. considering the PAC likelihood given in Definition 6, and additionally assuming that

$$\forall x_i \in x: \ \|\mathbf{x}_i\| = \|\phi(x_i)\| = \text{const.}$$

In order to see this let us recall that the support vector machine aims at maximising the *margin* $\gamma_z(\mathbf{w})$ *of the weight vector* $\mathbf{w}$ *on the training sample* z given by

$$\gamma_z(\mathbf{w}) := \min_{i \in \{1,\ldots,m\}} \underbrace{\frac{y_i \langle \mathbf{x}_i, \mathbf{w} \rangle}{\|\mathbf{w}\|}}_{\gamma_{x_i}(\mathbf{w})} = \frac{1}{\|\mathbf{w}\|} \min_{i \in \{1,\ldots,m\}} y_i \langle \mathbf{x}_i, \mathbf{w} \rangle , \tag{13}$$

which for all $\mathbf{w}$ of unit length is merely the minimal real-valued output (flipped to the correct sign) over the whole training sample. In order to solve this problem algorithmically one takes advantage of the fact that fixing the real-valued output to one (rather than the norm $\|\mathbf{w}\|$ of the weight vector $\mathbf{w}$) renders the problem of finding the margin maximiser $\mathbf{w}_{\text{SVM}}$ as a problem with a quadratic objective function ($\|\mathbf{w}\|^2 = \mathbf{w}'\mathbf{w}$) under linear constraints ($y_i \langle \mathbf{x}_i, \mathbf{w} \rangle \geq 1$), i.e.

$$\mathbf{w}_{\text{SVM}} := \operatorname*{argmax}_{\mathbf{w} \in \mathcal{W}} \left(\min_{i \in \{1,\ldots,m\}} y_i \langle \mathbf{x}_i, \mathbf{w} \rangle \right) \tag{14}$$

$$\propto \operatorname{argmin}_{\mathbf{w} \in \left\{ \mathbf{v} \,\middle|\, \min_{i \in \{1,\ldots,m\}} y_i \langle \mathbf{x}_i, \mathbf{v} \rangle = 1 \right\}} \left(\|\mathbf{w}\|^2 \right) . \tag{15}$$

Note that the set of weight vectors in (15) are called the weight vectors of the *canonical hyperplanes* (see Vapnik, 1998, p. 412) and that this set is *highly* dependent on the given training sample. Nonetheless, the solution to (15) is (up to scaling) equivalent to the solution of (14) — a formulation much more amenable for theoretical studies.

Interestingly, however, the quantity $\gamma_{x_i}(\mathbf{w})$ as implicitly defined in (13) is not only the distance of the point $y_i\mathbf{x}_i$ from the hyperplane having the normal $\mathbf{w}$ but also $\|\mathbf{x}_i\|$ times the Euclidean distance of the point $\mathbf{w}$ from the hyperplane having the normal $y_i\mathbf{x}_i$ (see Figure 1). Thus $\gamma_z(\mathbf{w})$ can be viewed as the radius of the ball $\left\{\mathbf{v} \in \mathcal{W} \mid \|\mathbf{w}-\mathbf{v}\| \leq b\left(\gamma_z(\mathbf{w})\right)\right\}$ that only contains weight vectors in version space $V(z)$. Here, $b : \mathbb{R}^+ \to \mathbb{R}^+$ is a strictly monotonic function of its argument and its effect is graphically depicted in Figure 1. As a consequence thereof, maximising the margin $\gamma_z(\mathbf{w})$ over the choice of $\mathbf{w}$ returns the classifier $\mathbf{w}_{\mathrm{SVM}}$ that is the centre of the largest ball still inscribable in version space. Note that the whole reasoning relied on the assumption that all training points x_i have a constant norm in feature space $\mathcal{K}$. If this assumption is violated, each distance of a classifier $\mathbf{w}$ to the hyperplane having the normal $y_i\mathbf{x}_i$ is measured on a different scale and thus the points with the largest norm $\|\mathbf{x}_i\|$ in feature space $\mathcal{K}$ have the highest influence on the resulting solution. To circumvent this problem is has been suggested elsewhere that input vectors should be normalised in feature space before applying any kernel method — in particular the support vector machine algorithm (see Herbrich and Graepel, 2001; Schölkopf et al., 1999; Joachims, 1998; Haussler, 1999). Furthermore, all indices $I_{\mathrm{SV}} \subseteq \{1,\ldots,m\}$ at which the minimum $y_i\langle\mathbf{x}_i,\mathbf{w}_{\mathrm{SVM}}\rangle$ in (14) is attained are the ones for which $y_i\langle\mathbf{x}_i,\mathbf{w}\rangle = 1$ in the formulation (15). As the latter are called *support vectors* we see that the support vectors are the training points at which the largest inscribable ball touches the corresponding hyperplane $\left\{\mathbf{w} \in \mathcal{W} \mid (y_i\langle\mathbf{x}_i,\mathbf{w}\rangle = 0)\right\}$.

2.5 Applying the Kernel Trick

When solving (15) over the possible choices of $\mathbf{w} \in \mathcal{W}$ it is well known that the solution $\mathbf{w}_{\mathrm{SVM}}$ admits the following representation

$$\mathbf{w}_{\mathrm{SVM}} = \sum_{i=1}^{m} \alpha_i \mathbf{x}_i\,,$$

that is the solution to (15) must live in the linear span of the training points. This follows naturally from the following theorem (see also Schölkopf et al., 2001).

Theorem 11 (Representer Theorem) *Suppose we are given a fixed mapping $\phi : \mathcal{X} \to \mathcal{K} \subseteq \ell_2^N$, a training sample $z = (x,y) \in \mathcal{Z}^m$, a cost function $c : \mathcal{X}^m \times \mathcal{Y}^m \times \mathbb{R}^m \to \mathbb{R} \cup \{\infty\}$ strictly monotonically decreasing in the third argument and the class of linear functions in $\mathcal{K}$ as given by* (10)*. Then any $\mathbf{w}_z \in \mathcal{W}$ defined by*

$$\mathbf{w}_z := \operatorname*{argmin}_{\mathbf{w}\in\mathcal{W}} c\left(x,y,\left(\langle\mathbf{x}_1,\mathbf{w}\rangle,\ldots,\langle\mathbf{x}_m,\mathbf{w}\rangle\right)\right) \tag{16}$$

admits a representation of the form

$$\exists \alpha \in \mathbb{R}^m : \qquad \mathbf{w}_z = \sum_{i=1}^{m} \alpha_i \mathbf{x}_i\,. \tag{17}$$

The proof is given in Appendix A.2. In order to see that this theorem applies to support vector machines note that (14) is equivalent to the minimiser of (16) when using

$$c\left(x,y,\left(\langle\mathbf{x}_1,\mathbf{w}\rangle,\ldots,\langle\mathbf{x}_m,\mathbf{w}\rangle\right)\right) = \min_{y_i \in y} -y_i\langle\mathbf{x}_i,\mathbf{w}\rangle\,,$$

which is strictly monotonically decreasing in its third argument. A slightly more difficult argument is necessary to see that the centre of mass (12) can also be written as a minimiser of (16) using a specific cost function c. At first we recall that the centre of mass has the property of minimising $\mathbf{E}_{\mathsf{W}|\mathsf{Z}^m=z}\left[\|\mathbf{w}-\mathsf{W}\|^2\right]$ over the choice of $\mathbf{w} \in \mathcal{W}$ (see also (30)).

Theorem 12 (Sufficiency of the linear span) *Suppose we are given a fixed mapping $\phi : \mathcal{X} \to \mathcal{K} \subseteq \ell_2^N$. Let us assume that $\mathbf{P}_{\mathsf{W}}$ is uniform and $\mathbf{P}_{\mathsf{Y}|\mathsf{X}=x,\mathsf{W}=\mathbf{w}}(y) = f(\operatorname{sign}(y\langle\mathbf{x},\mathbf{w}\rangle))$, i.e. the likelihood depends on the sign of the real-valued output $y\langle\mathbf{x},\mathbf{w}\rangle$ of $\mathbf{w}$. Let $\mathcal{L}_x := \left\{\sum_{i=1}^m \alpha_i\mathbf{x}_i \mid \alpha \in \mathbb{R}^m\right\}$ be the linear span of mapped data points $\{\mathbf{x}_1,\ldots,\mathbf{x}_m\}$ and $\mathcal{W}_x := \mathcal{W} \cap \mathcal{L}_x$. Then for any training sample $z \in \mathcal{Z}^m$ and any $\mathbf{w} \in \mathcal{W}$*

$$\int_{\mathcal{W}} \|\mathbf{w}-\mathbf{v}\|^2 \, d\mathbf{P}_{\mathsf{W}|\mathsf{Z}^m=z}(\mathbf{v}) = C \cdot \int_{\mathcal{W}_x} \|\mathbf{w}-\mathbf{v}\|^2 \, d\mathbf{P}_{\mathsf{W}|\mathsf{Z}^m=z}(\mathbf{v})\,, \tag{18}$$

that is, up to a constant $C \in \mathbb{R}^+$ that is independent of $\mathbf{w}$ *it suffices to consider vectors of unit length in the linear span of the mapped training points* $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$.

The proof is given in Appendix A.3. An immediate consequence of this theorem is the fact that we only need to consider the m–dimensional sphere $\mathcal{W}_x$ in order to find the centre of mass under the assumption of a uniform prior P_W. Hence a loss function c such that (16) finds the centre of mass is given by

$$c\left(x, y, \left(\langle \mathbf{x}_1, \mathbf{w} \rangle, \ldots, \langle \mathbf{x}_m, \mathbf{w} \rangle\right)\right) = 2\left(1 - \int_{\mathbb{R}^m} \sum \alpha_i \langle \mathbf{x}_i, \mathbf{w} \rangle \, d\mathsf{P}_{\mathsf{A}|\mathsf{Z}^m=(x,y)}\right)$$

where $\mathsf{P}_{\mathsf{A}|\mathsf{Z}^m=z}$ is only non-zero for vectors α such that $\|\sum_{i=1}^m \alpha_i \mathbf{x}_i\| = 1$ and is independent of $\mathbf{w}$.

The tremendous advantage of a representation of the solution $\mathbf{w}_z$ by (17) becomes apparent when considering the real-valued output of a classifier at any given data point (either training or test point)

$$\langle \mathbf{w}_z, \mathbf{x} \rangle = \left\langle \sum_{i=1}^m \alpha_i \mathbf{x}_i, \mathbf{x} \right\rangle = \sum_{i=1}^m \alpha_i \langle \mathbf{x}_i, \mathbf{x} \rangle = \sum_{i=1}^m \alpha_i k\left(x_i, x\right) .$$

Clearly, all that is needed in the feature space $\mathcal{K}$ is the *inner product function*

$$k\left(x, \tilde{x}\right) := \langle \phi\left(x\right), \phi\left(\tilde{x}\right) \rangle . \tag{19}$$

Reversing the chain of arguments indicates how the kernel trick may be used to find an efficient implementation. We fix a symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ called *kernel* and show that there exists a feature mapping $\phi_k : \mathcal{X} \to \mathcal{K} \subseteq \ell_2^N$ such that (19) is valid for all $x, \tilde{x} \in \mathcal{X}$. A sufficient condition for k being a valid inner product function is given by Mercer's theorem (see Mercer, 1909). In a nutshell, whenever the evaluation of k at any given sample $(x_1, \ldots, x_m)$ results in a positive semidefinite matrix $\mathbf{G}_{ij} := k\left(x_i, x_j\right)$ then k is a so called *Mercer kernel*. The matrix $\mathbf{G}$ is called the Gram matrix and is the only quantity needed in support vector and Bayes point machine learning. For further details on the kernel trick the reader is referred to Schölkopf et al. (1999); Cristianini and Shawe-Taylor (2000); Wahba (1990); Vapnik (1998).

3. Estimating the Bayes Point in Feature Space

In order to estimate the Bayes point in feature space $\mathcal{K}$ we consider a Monte Carlo method, i.e. instead of exactly computing the expectation (12) we approximate it by an average over weight vectors $\mathbf{w}$ drawn according to $\mathsf{P}_{\mathsf{W}|\mathsf{Z}^m=z}$ and restricted to $\mathcal{W}_x$ (see Theorem 12) . In the following we will restrict ourselves to the PAC likelihood given in (5) and P_W being uniform on the unit sphere $\mathcal{W} \subset \mathcal{K}$. By this assumption we know that the posterior is uniform over version space (see (6)). In Figure 2 we plotted an example for the special case of $N = 3$–dimensional feature space $\mathcal{K}$.

It is, however, already very difficult to sample uniformly from version space $V(z)$ as this set of points lives on a convex polyhedron on the unit sphere in[7] $\mathcal{W}_x$. In the following two subsections we present two methods to achieve this sampling. The first method develops on an idea of Ruján (1997) (later followed up by a kernel version of the algorithm in Ruján and Marchand, 2000) that is based on the idea of playing billiards in version space $V(z)$, i.e. after entering the version space with a very simple learning algorithm such as the kernel perceptron (see Algorithm 1) the classifier $\mathbf{w}$ is considered as a billiard ball and is bounced for a while within the convex polyhedron $V(z)$. If this billiard is ergodic with respect to the uniform distribution over $V(z)$, i.e. the travel time of the billiard ball spent in a subset $W \subseteq V(z)$ is proportional to $\frac{W}{V(z)}$, then averaging over the trajectory of the billiard ball leads in the limit of an infinite number of bounces to the centre of mass of version space.

The second method presented tries to overcome the large computational demands of the billiard method by only approximately achieving a uniform sampling of version space. The idea is to use the perceptron

7. Note that by Theorem 12 it suffices to sample from the projection of the version space onto $\mathcal{W}_x$.

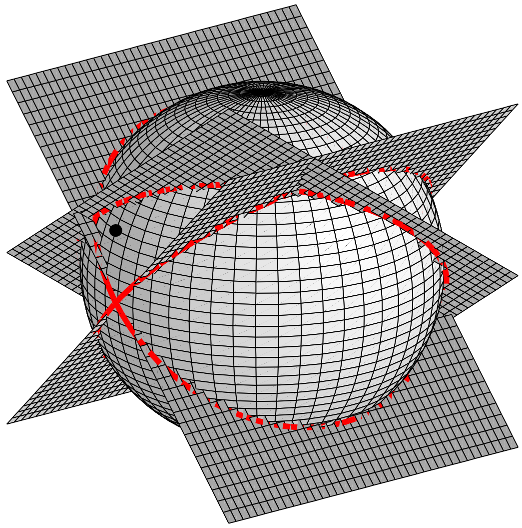


Figure 2: Plot of a version space (convex polyhedron containing the black dot) $V(z)$ in a 3–dimensional feature space $\mathcal{K}$. Each hyperplane is defined by a training example via its normal vector $y_i\mathbf{x}_i$.

learning algorithm in dual variables with different permutations $\Pi : \{1,\ldots,m\} \to \{1,\ldots,m\}$ so as to obtain different consistent classifiers $\mathbf{w}_i \in V(z)$ (see Watkin, 1993, for a similar idea). Obviously, the number of different samples obtained is finite and thus it is impossible to achieve exactness of the method in the limit of considering all permutations. Nevertheless, we shall demonstrate that in particular for the task of handwritten digit recognition the achieved performances are comparable to state-of-the-art learning algorithms.

Finally, we would like to remark that recently there have been presented other efficient methods to estimate the Bayes point directly (Rychetsky et al., 2000; Minka, 2001). The main idea in Rychetsky et al. (2000) is to work out all corners $\mathbf{w}_i$ of version space and average over them in order to approximate the centre of mass of version space. Note that there are exactly m corners because the i–th corner $\mathbf{w}_i$ satisfies $\langle \mathbf{x}_j, \mathbf{w}_i \rangle = 0$ for all $j \neq i$ and $y_i \langle \mathbf{x}_i, \mathbf{w}_i \rangle > 0$. If $\mathbf{X} = (\mathbf{x}_1,\ldots,\mathbf{x}_m)$ is the $N \times m$ matrix of mapped training points $x = (x_1,\ldots,x_m)$ flipped to their correct side and we use the approach (17) for $\mathbf{w}$ this simplifies to

$$\mathbf{X}'\mathbf{w}_i = \mathbf{X}'\mathbf{X}\alpha_i = \mathbf{G}\alpha_i = (0,\ldots,0,y_i,0\ldots,0)' =: y_i\mathbf{e}_i$$

where the r.h.s. is the i–th unit vector multiplied by y_i. As a consequence, the expansion coefficients α_i of the i–th corner $\mathbf{w}_i$ can easily be computed as $\alpha_i = y_i\mathbf{G}^{-1}\mathbf{e}_i$ and then need to be normalised such that $\|\mathbf{w}_i\| = 1$. The difficulty with this approach, however, is the fact that the inversion of the $m \times m$ Gram matrix $\mathbf{G}$ is $O(m^3)$ and is thus as computationally complex as support vector learning while not enjoying the anytime property of a sampling scheme.

The algorithm presented in Minka (2001, Chapter 5) (also see Opper and Winther, 2000, for an equivalent method) uses the idea of approximating the posterior measure $\mathbf{P}_{\mathsf{W}|\mathsf{Z}^m=z}$ by a product of Gaussian densities so that the centre of mass can be computed analytically. Although the approximation of the cut-off posterior over $\mathbf{P}_{\mathsf{W}|\mathsf{Z}^m=z}$ resulting from the delta-peaked likelihood given in Definition 6 by Gaussian measures seems very crude at first glance, Minka could show that his method compares favourably to the results presented in this paper.

3.1 Playing Billiards in Version Space

In this subsection we present the billiard method to estimate the Bayes point, i.e. the centre of mass of version space when assuming a PAC likelihood and a uniform prior $\mathbf{P}_{\mathsf{W}}$ over weight vectors of unit length (the pseudo

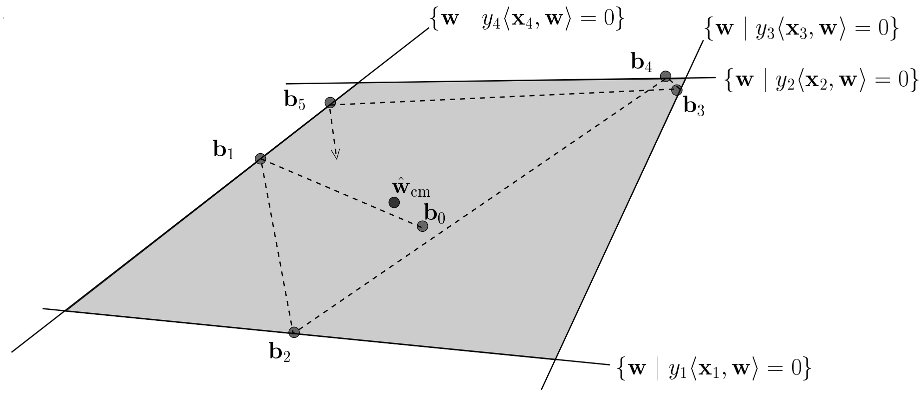


Figure 3: Schematic view of the kernel billiard algorithm. Starting at $\mathbf{b}_0 \in V(z)$ a trajectory of billiard bounces $\mathbf{b}_1, \ldots, \mathbf{b}_5, \ldots$ is calculated and then averaged over so as to obtain an estimate $\hat{\mathbf{w}}_{\mathrm{cm}}$ of the centre of mass of version space.

code is given on page 275). By Theorem 12 each position $\mathbf{b}$ of the billiard ball and each estimate $\mathbf{w}_i$ of the centre of mass of $V(z)$ can be expressed as linear combinations of the mapped input points, i.e.

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i \mathbf{x}_i, \quad \mathbf{b} = \sum_{i=1}^{m} \gamma_i \mathbf{x}_i, \qquad \alpha, \gamma \in \mathbb{R}^m .$$

Without loss of generality we can make the following ansatz for the direction vector $\mathbf{v}$ of the billiard ball

$$\mathbf{v} = \sum_{i=1}^{m} \beta_i \mathbf{x}_i, \qquad \beta \in \mathbb{R}^m .$$

Using this notation inner products and norms in feature space $\mathcal{K}$ become

$$\langle \mathbf{b}, \mathbf{v} \rangle = \sum_{i=1}^{m} \sum_{j=1}^{m} \gamma_i \beta_j k(x_i, x_j), \qquad \|\mathbf{b}\|^2 = \sum_{i,j=1}^{m} \gamma_i \gamma_j k(x_i, x_j), \tag{20}$$

where $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a Mercer kernel and has to be chosen beforehand. At the beginning we assume that $\mathbf{w}_0 = 0 \Leftrightarrow \alpha = 0$. Before generating a billiard trajectory in version space $V(z)$ we first run any learning algorithm to find an initial starting point $\mathbf{b}_0$ inside the version space (e.g. support vector learning or the kernel perceptron (see Algorithm 1)). Then the kernel billiard algorithm consists of three steps (see also Figure 3):

1. Determine the closest boundary in direction $\mathbf{v}_i$ starting from current position $\mathbf{b}_i$.

   Since it is computationally very demanding to calculate the flight time of the billiard ball *on* geodesics of the hyper-sphere $\mathcal{W}_x$ (see also Neal, 1997) we make use of the fact that the shortest distance in Euclidean space (if it exists) is also the shortest distance on the hyper-sphere $\mathcal{W}_x$. Thus, we have for the flight time τ_j of the billiard ball at position $\mathbf{b}_i$ in direction $\mathbf{v}_i$ to the hyperplane with normal vector $y_j \mathbf{x}_j$

   $$\tau_j = -\frac{\langle \mathbf{b}_i, \mathbf{x}_j \rangle}{\langle \mathbf{v}_i, \mathbf{x}_j \rangle} . \tag{21}$$

   After calculating all m flight times, we look for the smallest positive, i.e.

   $$c = \operatorname*{argmin}_{j \in \{i \mid \tau_i > 0\}} \tau_j .$$

Determining the closest bounding hyperplane in Euclidean space rather than on geodesics causes problems if the surface of the hyper-sphere $\mathcal{W}_x$ is almost orthogonal to the direction vector $\mathbf{v}_i$, in which case $\tau_c \rightarrow \infty$. If this happens we randomly generate a direction vector $\mathbf{v}_i$ pointing *towards* the version space $V(z)$. Assuming that the last bounce took place at the hyperplane having normal $y_{c'}\mathbf{x}_{c'}$ this condition can easily be checked by

$$y_{c'} \langle \mathbf{v}_i, \mathbf{x}_{c'} \rangle > 0 . \tag{22}$$

Note that since the samples are taking from the bouncing points the above procedure of dealing with the curvature of the hyper-sphere does not constitute an approximation but is exact. An alternative method of dealing with the problem of the curvature of the hyper-sphere $\mathcal{W}$ can be found in Minka (2001, Section 5.8)

2. Update the billiard ball's position to $\mathbf{b}_{i+1}$ and the new direction vector to $\mathbf{v}_{i+1}$.

   The new point $\mathbf{b}_{i+1}$ and the new direction $\mathbf{v}_{i+1}$ are calculated from

$$\begin{aligned} \mathbf{b}_{i+1} &= \mathbf{b}_i + \tau_c \mathbf{v}_i , & (23)\\ \mathbf{v}_{i+1} &= \mathbf{v}_i - 2\frac{\langle \mathbf{v}_i, \mathbf{x}_c \rangle}{\|\mathbf{x}_c\|^2}\mathbf{x}_c . & (24) \end{aligned}$$

   Afterwards the position $\mathbf{b}_{i+1}$ and the direction vector $\mathbf{v}_{i+1}$ need to be normalised. This is easily achieved by equation (20).

3. Update the centre of mass $\mathbf{w}_i$ of the whole trajectory by the new line segment from $\mathbf{b}_i$ to $\mathbf{b}_{i+1}$ calculated on the hyper-sphere $\mathcal{W}_x$.

   Since the solution $\mathbf{w}_\infty$ lies on the hyper-sphere $\mathcal{W}_x$ (see Theorem 11) we cannot simply update the centre of mass using a weighted vector addition. Let us introduce the operation $\oplus_\mu$ acting on vectors of unit length. This function has to have the following properties

$$\begin{aligned} \left\|\mathbf{s} \oplus_\mu \mathbf{t}\right\|^2 &= 1 , \\ \left\|\mathbf{t} - \mathbf{s} \oplus_\mu \mathbf{t}\right\| &= \mu \left\|\mathbf{t} - \mathbf{s}\right\| , \\ \mathbf{s} \oplus_\mu \mathbf{t} &= \rho_1 \left(\langle \mathbf{s}, \mathbf{t} \rangle , \mu\right) \mathbf{s} + \rho_2 \left(\langle \mathbf{s}, \mathbf{t} \rangle , \mu\right) \mathbf{t} , \\ \rho_1 \left(\langle \mathbf{s}, \mathbf{t} \rangle , \mu\right) \geq 0 \quad &, \quad \rho_2 \left(\langle \mathbf{s}, \mathbf{t} \rangle , \mu\right) \geq 0 . \end{aligned}$$

   This rather arcane definition implements a weighted addition of $\mathbf{s}$ and $\mathbf{t}$ such that μ is the fraction between the resulting chord length $\left\|\mathbf{t} - \mathbf{s} \oplus_\mu \mathbf{t}\right\|$ and the total chord length $\|\mathbf{t} - \mathbf{s}\|$. In Appendix A.4 it is shown that the following formulae for $\rho_1 \left(\langle \mathbf{s}, \mathbf{t} \rangle , \mu\right)$ and $\rho_2 \left(\langle \mathbf{s}, \mathbf{t} \rangle , \mu\right)$ implement such a weighted addition

$$\begin{aligned} \rho_1 \left(\langle \mathbf{s}, \mathbf{t} \rangle , \mu\right) &= \mu \sqrt{-\frac{\mu^2 - \mu^2 \langle \mathbf{s}, \mathbf{t} \rangle - 2}{\langle \mathbf{s}, \mathbf{t} \rangle + 1}} , \\ \rho_2 \left(\langle \mathbf{s}, \mathbf{t} \rangle , \mu\right) &= -\rho_1 \left(\langle \mathbf{s}, \mathbf{t} \rangle , \mu\right) \langle \mathbf{s}, \mathbf{t} \rangle \pm \left(\mu^2 \left(1 - \langle \mathbf{s}, \mathbf{t} \rangle\right) - 1\right) . \end{aligned}$$

   By assuming a constant line density on the manifold $V(z)$ the whole line between $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$ can be represented by the midpoint $\mathbf{m}$ on the manifold $V(z)$ given by

$$\mathbf{m} = \frac{\mathbf{b}_i + \mathbf{b}_{i+1}}{\|\mathbf{b}_i + \mathbf{b}_{i+1}\|} .$$

   Thus, one updates the centre of mass of the trajectory by

$$\mathbf{w}_{i+1} = \rho_1 \left(\langle \mathbf{w}_i, \mathbf{m} \rangle , \frac{\Xi_i}{\Xi_i + \xi_i} \right) \mathbf{w}_i + \rho_2 \left(\langle \mathbf{w}_i, \mathbf{m} \rangle , \frac{\Xi_i}{\Xi_i + \xi_i} \right) \mathbf{m} ,$$

where $\xi_i = \|\mathbf{b}_i - \mathbf{b}_{i+1}\|$ is the length of the trajectory in the i–th step and $\Xi_i = \sum_{j=1}^{i} \xi_j$ for the accumulated length up to the i–th step. Note that the operation $\oplus_\mu$ is only an approximation to addition operation we sought because an exact weighting would require the arc lengths rather than chord lengths.

As a stopping criterion we suggest computing an upper bound on ρ_2, the weighting factor of the new part of the trajectory. If this value falls below a pre-specified threshold (TOL) we stop the algorithm. Note that the increase in Ξ_i will always lead to termination.

3.2 Large Scale Bayes Point Machines

Clearly, all we need for estimating the centre of mass of version space (12) is a set of unit length weight vectors $\mathbf{w}_i$ drawn uniformly from $V(z)$. In order to save computational resources it might be advantageous to achieve a uniform sample only approximately. The classical perceptron learning algorithm offers the possibility to obtain up to $m!$ different classifiers in version space simply by learning on different permutations of the training sample. Of course due to the sparsity of the solution the number of different classifiers obtained is usually considerably less.

A classical theorem to be found in Novikoff (1962) guarantees the convergence of this procedure and furthermore provides an upper bound on the number t of mistakes needed until convergence. More precisely, if there exists a classifier $\mathbf{w}_{\mathrm{SVM}}$ with margin $\gamma_z(\mathbf{w}_{\mathrm{SVM}}) > 0$ (see (13)) then the number of mistakes until convergence — which is an upper bound on the sparsity of the solution — is not more than $\varsigma^2\gamma_z^{-2}(\mathbf{w}_{\mathrm{SVM}})$, where ς is the smallest real number such that $\|\mathbf{x}_i\|_{\mathcal{K}} \leq \varsigma$. The quantity $\gamma_z(\mathbf{w}_{\mathrm{SVM}})$ is maximised for the solution $\mathbf{w}_{\mathrm{SVM}}$ found by the support vector machine, and whenever the support vector machine is theoretically justified by results from learning theory (see Shawe-Taylor et al., 1998; Vapnik, 1998) the ratio $\varsigma^2\gamma_z^{-2}(\mathbf{w}_{\mathrm{SVM}})$ is considerably less than m, say $d \ll m$. Algorithmically, we can benefit from this sparsity by the following "trick": since

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i \mathbf{x}_i$$

all we need to store is the m–dimensional vector α. Furthermore, we keep track of the m–dimensional vector $\mathbf{o}$ of real-valued outputs

$$o_i = \langle \mathbf{x}_i, \mathbf{w}_t \rangle = \sum_{j=1}^{m} \alpha_j k(x_i, x_j)$$

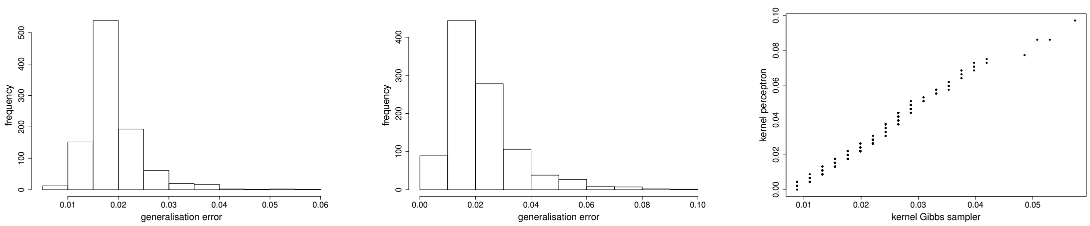
of the current solution at the i–th training point. By definition, in the beginning $\alpha = \mathbf{o} = 0$. Now, if $o_i y_i < 0$ we update α_i by $\alpha_i + y_i$ and update $\mathbf{o}$ by $o_j \leftarrow o_j + y_i k(x_i, x_j)$ which requires only m kernel calculations (the evaluation of the i–th row of the Gram matrix $\mathbf{G}$). In summary, the memory requirement of this algorithm is $2m$ and the number of kernel calculations is not more than $d \cdot m$. As a consequence, the computational requirement of this algorithm is no more than the computational requirement for the evaluation of the margin $\gamma_z(\mathbf{w}_{\mathrm{SVM}})$! We suggest to use this efficient perceptron learning algorithm in order to obtain samples $\mathbf{w}_i$ for the computation of the centre of mass (12).

In order to investigate the usefulness of this approach experimentally, we compared the distribution of generalisation errors of samples obtained by perceptron learning on permuted training samples with samples obtained by a full Gibbs sampling (see Graepel and Herbrich, 2001, for details on the kernel Gibbs sampler). For computational reasons, we used only 188 training patterns and 453 test patterns of the classes "1" and "2" from the MNIST data set[8]. In Figure 4 (a) and (b) we plotted the distribution over 1000 random samples using the kernel[9]

$$k(x, x') = \left(\langle x, x' \rangle + 1\right)^5 . \tag{25}$$

Using a quantile-quantile (QQ) plot technique we can compare both distributions in one graph (see Figure 4 (c)). These plots suggest that by simple permutation of the training sample we are able to obtain a sample of classifiers exhibiting a similar distribution of generalisation error to the one obtained by time-consuming Gibbs sampling.

8. This data set is publicly available at `http://www.research.att.com/~yann/ocr/mnist/`.
9. We decided to use this kernel because it showed excellent generalisation performance when using the support vector machine.



(a) (b) (c)

Figure 4: **(a)** Histogram of generalisation errors (estimated on a test set) using a kernel Gibbs sampler. **(b)** Histogram of generalisation errors (estimated on a test set) using a kernel perceptron. **(c)** QQ plot of distributions (a) and (b). The straight line indicates that the two distributions only differ by an additive and multiplicative constant, i.e. they exhibit the same rate of decay.

A very advantageous feature of this approach as compared to support vector machines are its adjustable time and memory requirements and the "anytime" availability of a solution due to sampling. If the training sample grows further and we are not able to spend more time learning, we can adjust the number of samples $\mathbf{w}$ used at the cost of slightly worse generalisation error (see also Section 4).

3.3 Extension to Training Error

To allow for training errors we recall that the version space conditions are given by

$$\forall (x_i, y_i) \in z: \qquad y_i \langle \mathbf{x}_i, \mathbf{w} \rangle = y_i \sum_{j=1}^{m} \alpha_j k(x_i, x_j) > 0 \,. \tag{26}$$

Now we introduce the following version space conditions in place of (26):

$$\forall (x_i, y_i) \in z: \qquad y_i \sum_{j=1}^{m} \alpha_j k(x_i, x_j) > -\lambda y_i \alpha_i k(x_i, x_i) \,, \tag{27}$$

where $\lambda \geq 0$ is an adjustable parameter related to the "softness" of version space boundaries.

Clearly, considering this from the billiard viewpoint, equation (27) can be interpreted as allowing penetration of the walls, an idea already hinted at in Ruján (1997). Since the linear decision function is invariant under any positive rescaling of expansion coefficients α, a factor α_i on the right hand side makes λ scale invariant as well. Although other ways of incorporating training errors are conceivable our formulation allows for a simple modification of the algorithms described in the previous two subsections. To see this we note that equation (27) can be rewritten as

$$\forall (x_i, y_i) \in z: \qquad y_i \left(\sum_{j=1}^{m} \alpha_j \left(1 + \lambda \mathbf{I}_{i=j}\right) k(x_i, x_j) \right) > 0 \,.$$

Hence we can use the above algorithms but with an additive correction to the diagonal terms of the Gram matrix . This additive correction to the kernel diagonals is similar to the quadratic margin loss used to introduce a soft margin during training of support vector machines (see Cortes, 1995; Shawe-Taylor and Cristianini, 2000). Another insight into the introduction of soft boundaries comes from noting that the distance between two points $\mathbf{x}_i$ and $\mathbf{x}_j$ in feature space $\mathcal{K}$ can be written

$$\left\| \mathbf{x}_i - \mathbf{x}_j \right\|^2 \quad = \quad \left\| \mathbf{x}_i \right\|^2 + \left\| \mathbf{x}_j \right\|^2 - 2 \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle \,,$$

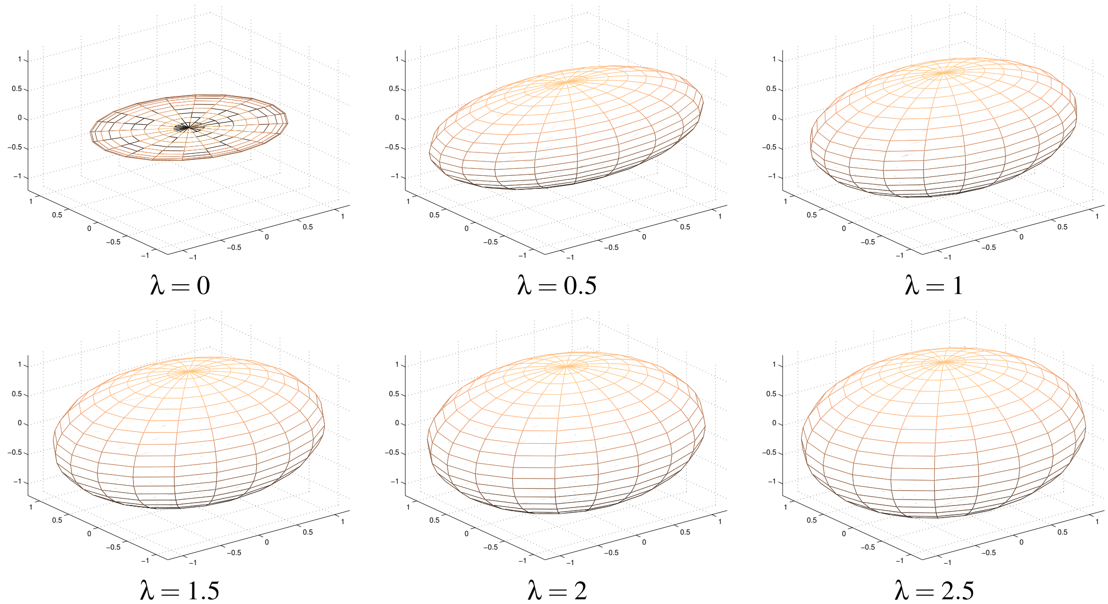


Figure 5: Parameter spaces for a two dimensional toy problem obtained by introducing training error via an additive correction to the diagonal term of the kernel matrix. In order to visualise the resulting parameter space we fixed $m = 3$ and normalised all axes by the product of eigenvalues $\sqrt{\lambda_1\lambda_2\lambda_3}$. See text for further explanation.

which in the case of points of unit length in feature space becomes $2(1+\lambda-k(x_i,x_j))$. Thus, if we add λ to the diagonal elements of the Gram matrix, the points become equidistant for $\lambda \to \infty$. This would give the resulting version space a more regular shape. As a consequence, the centre of the largest inscribable ball (support vector machine solution) would tend towards the centre of mass of the whole of version space.

We would like to recall that the effective parameter space of weight vectors considered is given by

$$\mathcal{W}_x := \left\{ \mathbf{w} = \sum_{i=1}^{m} \alpha_i \mathbf{x}_i \;\middle|\; \|\mathbf{w}\|^2 = \sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle = 1 \right\}.$$

In terms of α this can be rewritten as

$$\left\{ \alpha \in \mathbb{R}^m \;\middle|\; \alpha' \mathbf{G} \alpha = 1 \right\} \qquad \mathbf{G}_{ij} = \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle = k(x_i, x_j).$$

Let us represent the Gram matrix by its spectral decomposition, i.e. $\mathbf{G} = \mathbf{U}\Lambda\mathbf{U}'$ where $\mathbf{U}'\mathbf{U} = \mathbf{I}$ and $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_m)$ being the diagonal matrix of eigenvalues λ_i. Thus we know that the parameter space is the set of all coefficients $\widetilde{\alpha} = \mathbf{U}'\alpha$ which fulfil

$$\left\{ \widetilde{\alpha} \in \mathbb{R}^m : \widetilde{\alpha}' \Lambda \widetilde{\alpha} = 1 \right\}.$$

This is the defining equation of an m–dimensional axis parallel ellipsoid. Now adding the term λ to the diagonal of $\mathbf{G}$ makes $\mathbf{G}$ a full rank matrix (see Micchelli, 1986). In Figure 5 we plotted the parameter space for a 2D toy problem using only $m = 3$ training points. Although the parameter space is 3–dimensional for all $\lambda > 0$ we obtain a pancake like parameter space for small values of λ. For $\lambda \to \infty$ the set $\widetilde{\alpha}$ of admissible coefficients becomes the m–dimensional ball, i.e. the training examples become more and more orthogonal with increasing λ. The way we incorporated training errors corresponds to the choice of a new kernel given by

$$k_\lambda(x, \widetilde{x}) := k(x, \widetilde{x}) + \lambda \cdot \mathbf{I}_{x=\widetilde{x}}.$$

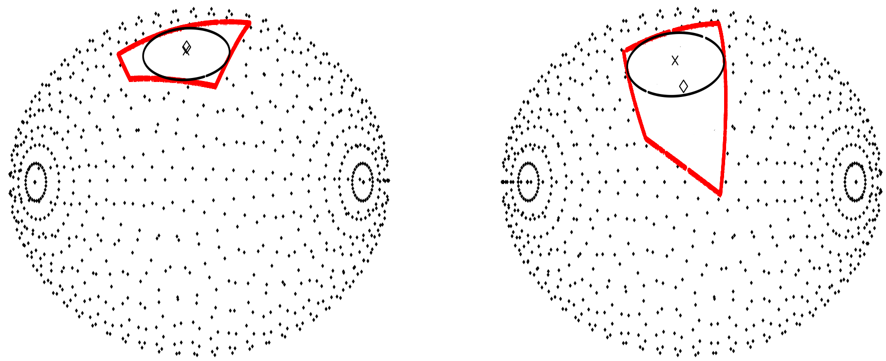


Figure 6: Version spaces $V(z)$ for two 3–dimensional toy problems. **(Left)** One can see that the approximation of the Bayes point (diamond) by the centre of the largest inscribable ball (cross) is reasonable if the version space is regularly shaped. **(Right)** The situation changes in the case of an elongated and asymmetric version space $V(z)$.

Finally, note that this modification of the kernel has *no effect* on new test points $x \notin x$ that are not elements of the training sample x. For an explanation of the effect of λ in the context of Gaussian processes see Opper and Winther (2000).

4. Experimental Results

In this section we present experimental results both on *University of California, Irvine* (UCI) benchmark datasets[10] and on two bigger task of handwritten digit recognition, namely *US postal service* (USPS) and *modified National Institute of Standards* (MNIST) digit recognition tasks. We compared our results to the performance of a support vector machine using reported test set performance from Rätsch et al. (2001) (UCI) Schölkopf (1997, p. 57) (USPS) and Cortes (1995) (MNIST). All the experiments were done using Algorithm 2 in Appendix B.

4.1 Artificial Data

For illustration purposes we setup a toy dataset of 10 training and 10000 test points in $\mathbb{R}^3$. The data points were uniformly generated in $[-1,1]^3$ and labelled by a randomly generated linear decision rule using the kernel $k(x,\tilde{x}) = \langle x,\tilde{x}\rangle$. In Figures 6 we illustrate the potential benefits of a Bayes point machine over a support vector machine for elongated version spaces. By using the billiard algorithm to estimate the Bayes point (see Subsection 3.1), we were able to track all positions $\mathbf{b}_i$ where the billiard ball hits a version space boundary. This allows us to easily visualise the version spaces $V(z)$. For the example illustrated in Figure 6 (right) the support vector machine and Bayes point solutions with hard margins/boundaries are far apart resulting in a noticeable reduction in generalisation error of the Bayes point machines (8.0%) compared to the support vector machine (15.1%) solution whereas for regularly shaped version spaces (Figure 6 (left)) the difference is negligible (6.1% to 6.0%).

10. publicly available at `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

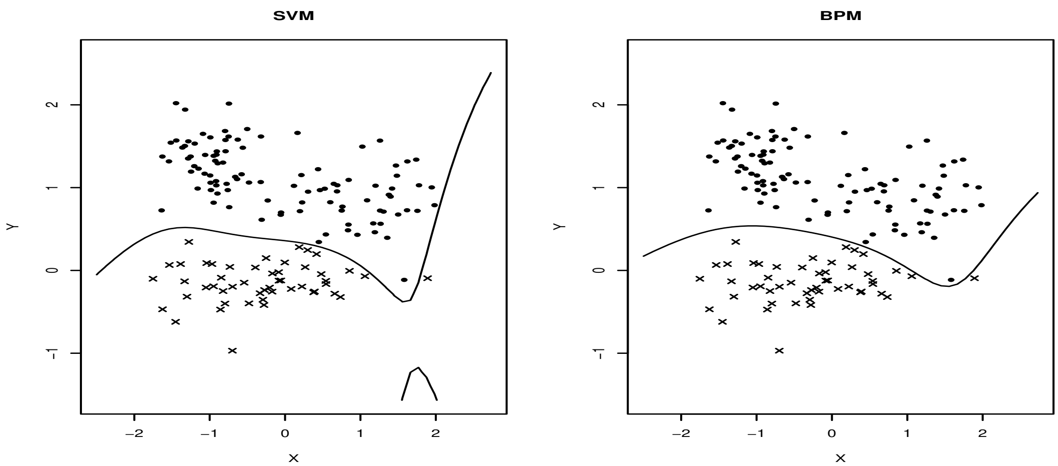


Figure 7: Decision functions for a 2D toy problem of a support vector machine (SVM) **(left)** and Bayes point machine (BPM) **(right)** using hard margins ($\lambda = 0$) and RBF kernels with $\sigma = 1$. Note that the Bayes point machine result in a much "flatter" function sacrificing margin ($\gamma_z(\mathbf{w}_{\mathrm{SVM}}) = 0.036 \rightarrow \gamma_z(\mathbf{w}_{\mathrm{cm}}) = 0.020$) for smoothness.

In a second illustrative example we compared the "smoothness" of the resulting decision function when using kernels both with support vector machines and Bayes point machines. In order to model a non-linear decision surface we used the radial basis function (RBF) kernel

$$k(x, \widetilde{x}) = \exp\left(-\frac{\|x - \widetilde{x}\|^2}{2\sigma^2}\right). \tag{28}$$

Figure 7 shows the resulting decision functions in the hard margin/boundary case. Clearly, the Bayes point machine solution appears much smoother than the support vector machine solution although its geometrical margin of 0.020 is significantly smaller.

The above examples should only be considered as aids to enhance the understanding of the Bayes point machines algorithm's properties rather than strict arguments about general superiority.

4.2 UCI Benchmark Datasets

To investigate the performance on real world datasets we compared hard margin support vector machines to Bayes point machines with hard boundaries ($\lambda = 0$) when using the kernel billiard algorithm described in Subsection 3.1. We studied the performance on 5 standard benchmarking datasets from the UCI Repository, and `banana` and `waveform`, two toy datasets (see Rätsch et al., 2001). In each case the data was randomly partitioned into 100 training and test sets in the ratio 60%:40%. The means and standard deviations of the average generalisation errors on the test sets are presented as percentages in the columns headed `SVM` (hard margin) and `BPM` ($\lambda = 0$) in Table 1. As can be seen from the results, the Bayes point machine outperforms support vector machines on almost all datasets at a statistically significant level. Note, however, that the result of the t-test is strictly valid only under the assumption that training and test data were independent — an assumption which may be violated by the procedure of splitting the one data set into 100 different pairs of training and test sets (Dietterich, 1998). Thus, the resulting p–values should serve only as an indication for the significance of the result.

In order to demonstrate the effect of positive λ (soft boundaries) we trained a Bayes point machine with soft boundaries and compared it to training a support vector machine with soft margin using the same Gram

	SVM (hard margin)	BPM (hard boundary)	σ	p-value
Heart	25.4±0.40	**22.8±0.34**	10.0	1.00
Thyroid	5.3±0.24	**4.4±0.21**	3.00	1.00
Diabetes	33.1±0.24	**32.0±0.25**	5.0	1.00
Waveform	13.0±0.10	**12.1±0.09**	20.0	1.00
Banana	16.2±0.15	**15.1±0.14**	0.5	1.00
Sonar	**15.4±0.37**	15.9±0.38	1.0	0.01
Ionosphere	11.9±0.25	**11.5±0.25**	1.5	0.99

Table 1: Experimental results on seven benchmark datasets. We used the RBF kernel given in (28) with values of σ found optimal for SVMs. Shown is the estimated generalisation error in percent. The standard deviation was obtained on 100 different runs. The final column gives the p–values of a paired t–test for the hypothesis "*BPM is better than SVM*" indicating that the improvement is statistically significant.

matrix (see equation (27)). It can be shown that such a support vector machine corresponds to a soft margin support vector machine where the margin slacks are penalised quadratically (see Cortes, 1995; Shawe-Taylor and Cristianini, 2000; Herbrich, 2001). In Figure 8 we have plotted the generalisation error as a function of λ for the toy problem from Figure 6 and the dataset `heart` using the same setup as in the previous experiment. We observe that the support vector machine with an ℓ_2 soft margin achieves a minimum of the generalisation error which is close to, or just above, the minimum error which can be achieved using a Bayes point machine with positive λ. This may not be too surprising taking the change of geometry into account (see Section 3.3). Thus, also the soft margin support vector machine approximates Bayes point machine with soft boundaries.

Finally we would like to remark that the running time of the kernel billiard was not much different from the running time of our support vector machine implementation. We did not use any chunking or decomposition algorithms (see, e.g. Osuna et al., 1997; Joachims, 1999; Platt, 1999) — which in case of support vector machines would have decreased the running time by orders of magnitudes. The most noticeable difference in running time was with the `waveform` and `banana` dataset where we are given $m = 400$ observations. This can be explained by the fact that the computational effort of the kernel billiard method is $O\left(B \cdot m^2\right)$ where B is the number of bounces. As we set our tolerance criterion TOL for stopping very low ($\approx 10^{-4}$), the approximate number B of bounces for these datasets was $B \approx 1\,000$. Hence, in contrast to the computational effort of using the support vector machines of $O\left(m^3\right)$ the number B of bounces lead to a much higher computational demand when using the kernel billiard.

4.3 Handwritten Digit Recognition

For the two tasks we now consider our inputs are $n \times n$ grey value images which were transformed into n^2–dimensional vectors by concatenation of the rows. The grey values were taken from the set $\{0,\ldots,255\}$. All images were labelled by one of the ten classes "0" to "9". For each of the ten classes $y = \{0,\ldots,9\}$ we ran the perceptron algorithm $L = 10$ times each time labelling all training points of class y by $+1$ and the remaining training points by -1. On a Pentium III 500 MHz with 128 MB memory each learning trial took $10-20$ minutes (MNIST) or $1-2$ minutes (USPS), respectively[11]. For the classification of a test image x

11. Note, however, that we made use of the fact that $\approx 40\%$ of the grey values of each image are 0 since they encode background. Therefore, we encoded each image as an index-value list which allows much faster computation of the inner products $\langle x, \widetilde{x} \rangle$ and speeds up the algorithm by a factor of 2–3.

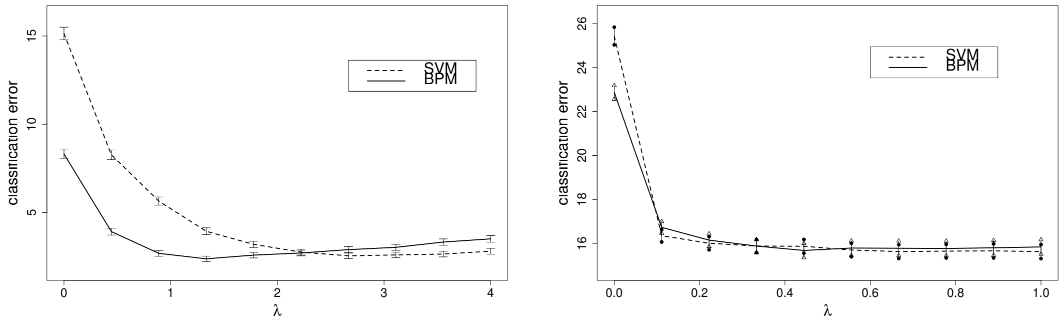


Figure 8: Comparison of soft boundary Bayes point machine with soft margin support vector machine. Plotted is the generalisation error versus λ for a toy problem using linear kernels **(left)** and the `heart` dataset using RBF kernels with $\sigma = 3.0$ **(right)**. The error bars indicate one standard deviation of the estimated mean.

we calculated the real-valued output of all 100 different classifiers[12] by

$$f_i(x) = \frac{\langle \mathbf{x}, \mathbf{w}_i \rangle}{\|\mathbf{w}_i\| \|\mathbf{x}\|} = \frac{\sum_{j=1}^{m} (\alpha_i)_j k(x_j, x)}{\sqrt{\sum_{r=1}^{m}\sum_{s=1}^{m} (\alpha_i)_r (\alpha_i)_s k(x_r, x_s)} \sqrt{k(x,x)}} ,$$

where we used the kernel k given by (25). Here, $(\alpha_i)_j$ refers to the expansion coefficient corresponding to the i–th classifier and the j–th data point. Now, for each of the ten classes we calculated the real-valued decision of the Bayes point estimate $\widehat{\mathbf{w}}_{\mathrm{cm},y}$ by[13]

$$f_{\mathrm{bp},y}(x) = \langle \mathbf{x}, \widehat{\mathbf{w}}_{\mathrm{cm},y} \rangle = \frac{1}{L}\sum_{i=1}^{L} \langle \mathbf{x}, \mathbf{w}_{i+yL} \rangle .$$

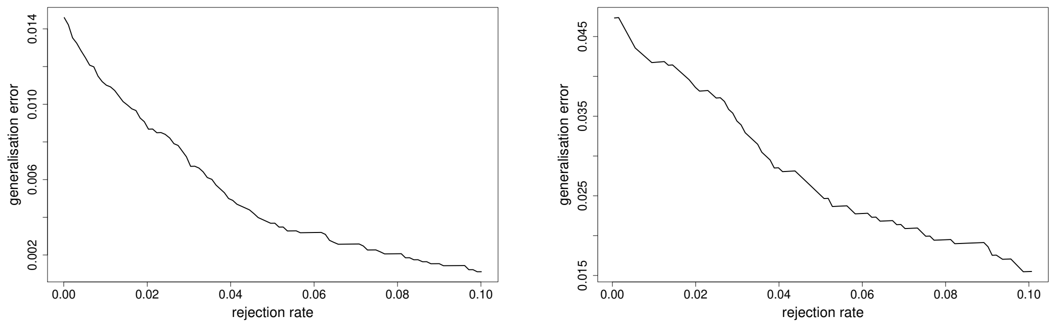
In a Bayesian spirit, the final decision was carried out by

$$h_{\mathrm{bp}}(x) := \operatorname*{argmax}_{y \in \{0,\ldots,9\}} f_{\mathrm{bp},y}(x) .$$

Note that $f_{\mathrm{bp},y}(x)$ can be interpreted as an (unnormalised) approximation of the posterior probability that x is of class y when restricted to the function class (10) (see Platt, 2000). In order to test the dependence of the generalisation error on the magnitude $\max_y f_{\mathrm{bp},y}(x)$ we fixed a certain rejection rate $r \in [0,1]$ and rejected the set of $r \cdot 10000$ test points with the smallest value of $\max_y f_{\mathrm{bp},y}(x)$.

MNIST Handwritten Digits In the first of our large scale experiment we used the full MNIST dataset with 60000 training examples and 10000 test examples of 28×28 grey value images of handwritten digits. The plot resulting from learning only 10 consistent classifiers per class and rejection based on the real-valued output of the single Bayes points is depicted in Figure 9 (left). As can be seen from this plot, even without rejection the Bayes point has excellent generalisation performance when compared to support vector machines which achieve a generalisation error of[14] 1.4%. Furthermore, rejection based on the real-valued

12. For notational simplicity we assume that the first L classifiers are classifiers for the class “0”, the next L for class “1” and so on.
13. Note that in this subsection y ranges from $\{0,\ldots,9\}$.
14. The result of 1.1% with the kernel (25) and a polynomial degree of four could not be reproduced and is thus considered invalid (personal communication with P. Haffner). Note also that the best results with support vector machines were obtained when using a soft margin.



MNIST USPS

Figure 9: Generalisation error as a function of the rejection rate for the MNIST and USPS data set. **(Left)** For MNIST, the support vector machine achieved 1.4% without rejection as compared to 1.46% for the Bayes point machine. Note that by rejection based on the real-valued output the generalisation error could be reduced to 0.1% indicating that this measure is related to the probability of misclassification of single test points. **(Right)** On USPS, the support vector machine achieved 4.6% without rejection as compared to 4.73% for the Bayes point machine.

output $f_{\text{bp}}(x)$ turns out to be excellent thus reducing the generalisation error to 0.1%. One should also bear in mind that the learning time for this simple algorithm was comparable to that of support vector machines which need ≈ 8 hours per digit[15] (see Platt, 1999, p. 201, Table 12.2).

USPS Handwritten Digits In the second of our large scale experiments we used the USPS dataset with 7291 training examples an 2007 test examples of 16×16 grey value images of handwritten digits. The resulting plot of the generalisation error when rejecting test examples based on the real-valued outputs of the single Bayes points is shown in Figure 9 (right). Again, the resulting classifier has a generalisation error performance comparable to support vector machines whose best results are 4.5% when using a soft margin and 4.6% in the hard margin scenario. In Figure 10 we plotted the 25 most commonly used images $x_i \in x$ with non-zero coefficients $(\alpha_j)_i$ across the 100 different classifiers learned. Though no margin maximisation was performed it turns out that in accordance with the "support vector philosophy" these are the hard patterns in the datasets with respect to classification. Moreover, as can be seen from the 1–st, 6–th and 8–th example there is clearly noise in the dataset which could potentially be taken into account using the techniques outlined in Subsection 3.3 at no extra computational cost.

5. Discussion and Conclusion

In this paper we presented two estimation methods for the Bayes point for linear classifiers in feature spaces. We showed how the support vector machine can be viewed as an (spherical) approximation method to the Bayes point hyperplane. By randomly generating consistent hyperplanes playing billiards in version space we showed how to stochastically approximate this point. In the field of Markov Chain Monte Carlo methods such approaches are known as *reflective slice sampling* (Neal, 1997). Current investigations in this field include the question of ergodicity of such methods. The second method of estimating the Bayes point consists of running the perceptron algorithm with several permutation of the training sample in order to average over the sample thereby obtained. By its inherent simplicity it is much more amenable to large scale problems and in particular compares favourably to state-of-the-art methods such as support vector learning.

15. Recently, DeCoste and Schölkopf (2002) demonstrated that an efficient implementation of the support vector machine reduces the amount of learning time to ≈ 1 hour per digit.

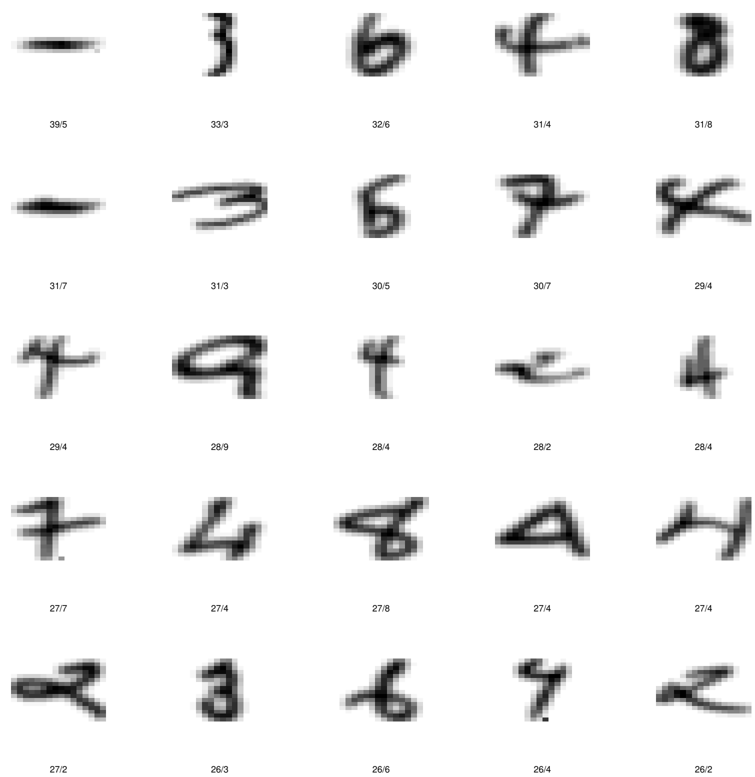


Figure 10: Shown are the 25 most commonly used examples $x_i \in x$ (non-zero coefficients $(\alpha_j)_i$ for many $j \in \{1,\ldots,100\}$) from the USPS dataset across the 100 different classifiers learned using the perceptron learning algorithm. The two numbers below each digit give the number of classifiers they appeared in and the true class $y \in \{0,\ldots,9\}$ in the training sample. Interestingly, in accordance with the philosophy behind support vectors these are the "hardest" patterns with respect to the classification task although no explicit margin maximisation was performed.

The centre of mass approach may also be viewed as a multidimensional extension of the *Pitman estimator* (Pitman, 1939) if the weight vector $\mathbf{w}$ is thought of as a location parameter to be estimated from the data. Unfortunately, neither the centre of mass of version space nor the support vector solution are invariant under general linear transformations of the data but only under the class of orthogonal transformations (see, e.g. Schölkopf, 1997). For the centre of mass this is due to the normalisation of the weight vector. Note that it is this normalisation that makes it conceptually hard to incorporate a bias dimension into our framework.

We presented a derivation of the Bayes point as the optimal projection of the Bayes classification strategy. This strategy is known to be the optimal strategy, i.e. the classification strategy which results in classifications with the smallest generalisation error, when considering the generalisation error *on average* over the random draw of target hypothesis according to the prior $\mathbf{P}_{\mathsf{H}}$. It is worthwhile to mention, however, that recent results in the PAC community allow one to obtain performance guarantees for the Bayesian classification strategy even *for single target hypotheses* $h \sim \mathbf{P}_{\mathsf{H}}$ which hold for most random draws of the training sample used (see McAllester, 1998, 1999). The results indicate that the fraction of the volume of parameter space to the volume of version space plays a crucial role in the generalisation error of Bayesian classifiers. It could be shown elsewhere (Herbrich et al., 1999b) that these bounds can be extended to *single classifiers* and then involve the volume of the largest point symmetric body around the classifier fully contained in version space (see Figure 2). These results may additionally motivate the centre of mass as a classifier with good volume ratio and thus good generalisation. The results also indicate that under circumstances where the shape of the version space is almost spherical the classical support vector machine gives the best result (see, e.g. Herbrich and Graepel, 2001).

In a series of experiments it has been shown that the Bayes point, i.e. the centre of mass of version space, has excellent generalisation performance — even when only broadly approximated by the average classifier found with simple perceptrons. Furthermore, it was be demonstrated that the real-valued output of the Bayes point on new test points serves as a reliable confidence measure on its prediction. An interesting feature of the Bayes point seems to be that the "hardest" patterns in the training sample tend to have the largest contribution in the final expansion too. This is in accordance with the support vector philosophy although the Bayes point machine algorithm does not perform any kind of margin maximisation explicitly.

Bayes points in feature space constitute an interesting bridge between the Bayesian approach to machine learning and statistical learning theory. In this paper we have shown that they outperform hard margin support vector machines. It is well known that the introduction of a soft margin improves the generalisation performance of support vector machines on most datasets by allowing for training errors. Consequently, we introduced a mechanism for Bayesian learning with training errors admitted. A comparison of the generalisation performance of the two types of systems shows that they exhibit a much closer generalisation performance than in the hard boundary/margin case.

Although it is generally believed that sparsity in terms of the expansion coefficients α is an indicator for good generalisation (see, e.g. Littlestone and Warmuth, 1986; Herbrich et al., 2000b) the algorithms presented show that also dense classifiers exhibit a good generalisation performance. An interesting question arising from our observation is therefore, which properties of single classifiers in version space are responsible for good generalisation?

Acknowledgements

This work was partially done during a research stay of Ralf Herbrich at University of Bristol and Royal Holloway University London in 1998. He would like to thank Colin Campbell and John Shawe-Taylor for the excellent research environment and also for the warm hospitality during that stay. We are also greatly indebted to Matthias Burger, Søren Fiig Jarner, Ulrich Kockelkorn, Klaus Obermayer, Manfred Opper, Craig Saunders, Peter Bollmann-Sdorra, Matthias Seeger, John Shawe-Taylor, Alex Smola, Jason Weston, Bob Williamson and Hugo Zaragoza for fruitful and inspiring discussions. Special thanks go to Patrick Haffner for pointing out the speed improvement by exploiting sparsity of the MNIST and USPS images and to Jun Liao for pointing out a mistake in Algorithm 2. Finally we would like to thank Chris Williams and the three anonymous reviewers for their comments and suggestions that greatly improved the manuscript.

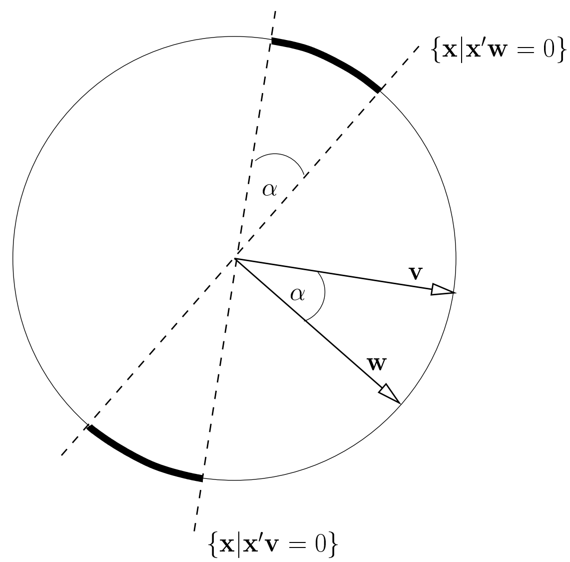


Figure 11: The fraction of points on the circle which are differently classified by $\mathbf{w}$ and $\mathbf{v}$ is depicted by the solid black arc. Note that this fraction is in general given by $\frac{2\alpha}{2\pi}=\frac{\alpha}{\pi}=\frac{\arccos(\langle\mathbf{w},\mathbf{v}\rangle)}{\pi}$.

Appendix A. Proofs

A.1 Convergence of Centre of Mass to the Bayes Point

In this section we present the proof of Theorem 10. We start with a simple lemma.

Lemma 13 (Generalisation Error for Spherical Distributions in Feature Space) *Suppose we are given a fixed mapping $\phi:\mathcal{X}\to\mathcal{K}\subseteq\ell_2^N$ resulting in $\{\mathbf{x}:=\phi(x)\mid x\in\mathcal{X}\}$. Furthermore let us assume that P_{X} is governed by* (11)*. Then, for all* $\mathbf{w}$*,* $\|\mathbf{w}\|=1$*, and* $\mathbf{v}$*,* $\|\mathbf{v}\|=1$*, it holds true that*

$$\mathsf{E}_{\mathsf{X}}\left[l_{0-1}\left(\operatorname{sign}\left(\langle\mathsf{X},\mathbf{w}\rangle\right),\operatorname{sign}\left(\langle\mathsf{X},\mathbf{v}\rangle\right)\right)\right]=\frac{\arccos\left(\langle\mathbf{w},\mathbf{v}\rangle\right)}{\pi}\,.$$

Proof For a fixed value $r\in\mathbb{R}^+$ let us consider all $\mathbf{x}$ such that $\|\mathbf{x}\|^2=r^2$. Given $\mathbf{w}\in\mathcal{K}$ and $\mathbf{v}\in\mathcal{K}$ we consider the projection $\mathbf{P}_{\mathbf{w},\mathbf{v}}:\mathcal{K}\to\mathcal{K}$ into the linear space spanned by $\mathbf{w}$ and $\mathbf{v}$ and its complement $\mathbf{P}^{\perp}_{\mathbf{w},\mathbf{v}}$, i.e.

$$\forall\mathbf{x}\in\mathcal{K}:\qquad \mathbf{x}=\mathbf{P}_{\mathbf{w},\mathbf{v}}\left(\mathbf{x}\right)+\mathbf{P}^{\perp}_{\mathbf{w},\mathbf{v}}\left(\mathbf{x}\right)\,.$$

Then for $\mathbf{w}$ (and $\mathbf{v}$) it holds true that

$$\begin{aligned}\operatorname{sign}\left(\langle\mathbf{x},\mathbf{w}\rangle\right) &= \operatorname{sign}\left(\left\langle\mathbf{P}_{\mathbf{w},\mathbf{v}}\left(\mathbf{x}\right)+\mathbf{P}^{\perp}_{\mathbf{w},\mathbf{v}}\left(\mathbf{x}\right),\mathbf{w}\right\rangle\right)\\ &= \operatorname{sign}\left(\left\langle\mathbf{P}_{\mathbf{w},\mathbf{v}}\left(\mathbf{x}\right),\mathbf{w}\right\rangle+\left\langle\mathbf{P}^{\perp}_{\mathbf{w},\mathbf{v}}\left(\mathbf{x}\right),\mathbf{w}\right\rangle\right)\\ &= \operatorname{sign}\left(\left\langle\mathbf{P}_{\mathbf{w},\mathbf{v}}\left(\mathbf{x}\right),\mathbf{w}\right\rangle\right)\,.\end{aligned}$$

Hence for any value of $r\in\mathbb{R}^+$ the notion of Figure 11 applies and gives

$$\forall r\in\mathbb{R}^+:\qquad \mathsf{E}_{\mathsf{X}|\,\|\mathsf{X}\|=r}\left[l_{0-1}\left(\operatorname{sign}\left(\langle\mathsf{X},\mathbf{w}\rangle\right),\operatorname{sign}\left(\langle\mathsf{X},\mathbf{v}\rangle\right)\right)\right]=\frac{\arccos\left(\langle\mathbf{w},\mathbf{v}\rangle\right)}{\pi}\,.$$

Thus integrating over r results in

$$\begin{aligned}\mathbf{E}_{\mathsf{X}}\left[l_{0-1}\left(\operatorname{sign}\left(\langle \mathbf{X},\mathbf{w}\rangle\right),\operatorname{sign}\left(\langle \mathbf{X},\mathbf{v}\rangle\right)\right)\right] &= \int_0^{+\infty}\frac{2}{\sqrt{\pi}}\exp\left(-r^2\right)\cdot\frac{\arccos\left(\langle \mathbf{w},\mathbf{v}\rangle\right)}{\pi}\,dr \\ &= \frac{\arccos\left(\langle \mathbf{w},\mathbf{v}\rangle\right)}{\pi}\,.\end{aligned}$$

■

According to Definition 9 and the previous lemma, in order to find the Bayes point $\mathbf{w}_{\text{bp}}$ for a given training sample z we need to find the vector $\mathbf{v}$ which minimises the following function

$$\begin{aligned}&\mathbf{E}_{\mathsf{X}}\left[\mathbf{E}_{\mathsf{W}|\mathsf{Z}^m=z}\left[l_{0-1}\left(\operatorname{sign}\left(\langle \mathbf{X},\mathbf{v}\rangle\right),\operatorname{sign}\left(\langle \mathbf{X},\mathbf{W}\rangle\right)\right)\right]\right] \\ &\quad= \mathbf{E}_{\mathsf{W}|\mathsf{Z}^m=z}\left[\mathbf{E}_{\mathsf{X}}\left[l_{0-1}\left(\operatorname{sign}\left(\langle \mathbf{X},\mathbf{v}\rangle\right),\operatorname{sign}\left(\langle \mathbf{X},\mathbf{W}\rangle\right)\right)\right]\right] \\ &\quad= \mathbf{E}_{\mathsf{W}|\mathsf{Z}^m=z}\left[\frac{\arccos\left(\langle \mathbf{v},\mathbf{W}\rangle\right)}{\pi}\right]\end{aligned}$$

subject to the constraint $\|\mathbf{v}\|=1$. Hence we have to determine the saddle point of the following Lagrangian

$$L_{\text{exact}}\left(\mathbf{v},\alpha\right)=\mathbf{E}_{\mathsf{W}|\mathsf{Z}^m=z}\left[\frac{\arccos\left(\langle \mathbf{v},\mathbf{W}\rangle\right)}{\pi}\right]+\alpha\left(\langle \mathbf{v},\mathbf{v}\rangle-1\right)\,,$$

w.r.t. $\mathbf{v}$ and α. The difficulty with this expression, however, is that by

$$\begin{aligned}\nabla_{\mathbf{v}}L_{\text{exact}}\left(\mathbf{v},\alpha\right)\big|_{\mathbf{v}_{\text{bp}}} &= \mathbf{E}_{\mathsf{W}|\mathsf{Z}^m=z}\left[-\frac{\mathbf{W}}{\sqrt{1-\left(\left\langle \mathbf{v}_{\text{bp}},\mathbf{W}\right\rangle\right)^2}}\right]+2\alpha\mathbf{v}_{\text{bp}}=0\,, \\ 2\alpha\mathbf{v}_{\text{bp}} &= \mathbf{E}_{\mathsf{W}|\mathsf{Z}^m=z}\left[\frac{\mathbf{W}}{\sqrt{1-\left(\left\langle \mathbf{v}_{\text{bp}},\mathbf{W}\right\rangle\right)^2}}\right]\,,\end{aligned}$$

the resulting fix-point equations for all components v_i are coupled because of the $\left(1-\left(\langle \mathbf{v},\mathbf{w}\rangle\right)^2\right)^{-\frac{1}{2}}$ term within the expectation thus involving all components. Nevertheless, we can find a good proxy for $\arccos\left(\langle \mathbf{v},\mathbf{w}\rangle\right)/\pi$ by $\left(1-\langle \mathbf{v},\mathbf{w}\rangle\right)/2$ (see Figure 12 on page 270). This is made more precise in the following lemma.

Lemma 14 (Quality of Euclidean Distance Proxy) *Suppose we are given a fixed mapping* $\phi:\mathcal{X}\to\mathcal{K}\subseteq\ell_2^N$ *resulting in* $\{\mathbf{x}:=\phi(x)\mid x\in\mathcal{X}\}$. *Furthermore let us assume that* $\mathbf{P}_{\mathsf{X}}$ *is governed by* (11). *Given a fixed vector* $\mathbf{v}\in\mathcal{K}$ *of unit length, i.e.* $\|\mathbf{v}\|=1$, *let us finally assume that*

$$\min_{\mathbf{w}:\mathbf{P}_{\mathsf{W}|\mathsf{Z}^m=z}(\mathbf{w})>0}\left|\langle \mathbf{v},\mathbf{w}\rangle\right|>\varepsilon\,. \tag{29}$$

Then we know that

$$\begin{aligned}\mathbf{E}_{\mathsf{W}|\mathsf{Z}^m=z}\left[\mathbf{E}_{\mathsf{X}}\left[l_{0-1}\left(\operatorname{sign}\left(\langle \mathbf{X},\mathbf{W}\rangle\right),\operatorname{sign}\left(\langle \mathbf{X},\mathbf{v}\rangle\right)\right)\right]\right] &\leq \mathbf{E}_{\mathsf{W}|\mathsf{Z}^m=z}\left[\frac{1}{4}\|\mathbf{W}-\mathbf{v}\|^2\right]+\kappa(\varepsilon)\,, \\ \mathbf{E}_{\mathsf{W}|\mathsf{Z}^m=z}\left[\mathbf{E}_{\mathsf{X}}\left[l_{0-1}\left(\operatorname{sign}\left(\langle \mathbf{X},\mathbf{W}\rangle\right),\operatorname{sign}\left(\langle \mathbf{X},\mathbf{v}\rangle\right)\right)\right]\right] &\geq \mathbf{E}_{\mathsf{W}|\mathsf{Z}^m=z}\left[\frac{1}{4}\|\mathbf{W}-\mathbf{v}\|^2\right]-\kappa(\varepsilon)\,,\end{aligned}$$

where

$$\kappa(\varepsilon):=\begin{cases}\frac{\arccos(\varepsilon)}{\pi}-\frac{1-\varepsilon}{2} & \textit{if } \varepsilon<0.23\\ 0.11 & \textit{otherwise}\end{cases}\,.$$

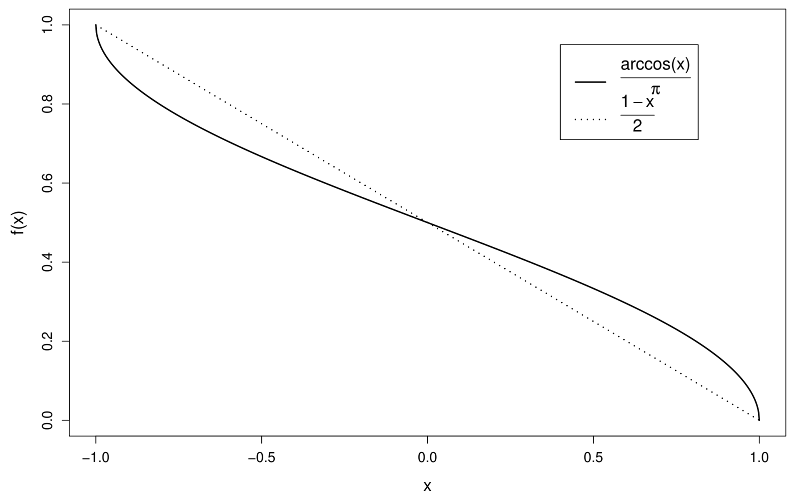


Figure 12: Plot of the functions $\arccos(x)/\pi$ and $(1-x)/2$ vs. x. As we can see, translating the latter function by not more than ≈ 0.11 shows that it is a both an upper and lower bound for $\arccos(x)/\pi$ and thus a reasonable proxy.

Proof Using Lemma 13 we only need to show that under the assumption (29) it holds true that

$$\frac{1}{4}\|\mathbf{v}-\mathbf{w}\|^2-\kappa(\varepsilon)\leq\frac{\arccos(\langle\mathbf{v},\mathbf{w}\rangle)}{\pi}\leq\frac{1}{4}\|\mathbf{v}-\mathbf{w}\|^2+\kappa(\varepsilon)\,.$$

At first we notice that

$$\frac{1}{4}\|\mathbf{v}-\mathbf{w}\|^2=\frac{1}{4}\left(\|\mathbf{v}\|^2-2\langle\mathbf{v},\mathbf{w}\rangle+\|\mathbf{w}\|^2\right)=\frac{1-\langle\mathbf{v},\mathbf{w}\rangle}{2}\,.$$

Thus let us determine the maximal difference in

$$f(x)=\frac{\arccos(x)}{\pi}-\frac{1-x}{2}$$

in the interval $(-1,1)$. A straightforward calculation reveals that the maximum occurs at $x^*=\sqrt{1-4\pi^{-2}}$ and is $0.10<f(x^*)<0.11$. Hence whenever $\varepsilon<1-\sqrt{1-4\pi^{-2}}<0.23$ we can directly use $f(x)$ which itself is in the worst case upper bounded by 0.11. Noticing that $\forall x\in(0,1):f(x)=-f(-x)$ proves the lemma. ■

If we replace $\arccos(\langle\mathbf{v},\mathbf{w}\rangle)/\pi$ by $\|\mathbf{w}-\mathbf{v}\|^2/4$ on the basis of the previous lemma we obtain the simpler problem of determining the saddle point of the Lagrangian

$$L_{\text{approx}}(\mathbf{v},\alpha)=\mathbf{E}_{\mathsf{W}|\mathsf{Z}^m=z}\left[\frac{1}{4}\|\mathsf{W}-\mathbf{v}\|^2\right]+\alpha\left(\mathbf{v}'\mathbf{v}-1\right)\,.$$

Taking the derivative w.r.t. $\mathbf{v}$ thus yields

$$\begin{aligned}\nabla_{\mathbf{v}}L_{\text{approx}}(\mathbf{v},\alpha)\big|_{\mathbf{v}_{\text{cm}}}&=\mathbf{E}_{\mathsf{W}|\mathsf{Z}^m=z}\left[-\frac{1}{2}\mathsf{W}\right]+2\alpha\mathbf{v}_{\text{cm}}=0\,,\\ 2\alpha\mathbf{v}_{\text{cm}}&=\mathbf{E}_{\mathsf{W}|\mathsf{Z}^m=z}\left[\frac{1}{2}\mathsf{W}\right]\,.\end{aligned}\qquad(30)$$

The value of α is determined by multiplying the whole expression by $\mathbf{v}_{\rm cm}$ and utilising the constraint $\mathbf{v}_{\rm cm}'\mathbf{v}_{\rm cm} = 1$, i.e.

$$\begin{aligned} 2\alpha\mathbf{v}_{\rm cm}'\mathbf{v}_{\rm cm} &= 2\alpha = \mathsf{E}_{\mathsf{W}|\mathsf{Z}^m=z}\left[\frac{1}{2}\langle \mathsf{W}, \mathbf{v}_{\rm cm}\rangle\right] \\ \alpha &= \frac{1}{4}\mathsf{E}_{\mathsf{W}|\mathsf{Z}^m=z}\left[\langle \mathsf{W}, \mathbf{v}_{\rm cm}\rangle\right] . \end{aligned}$$

Resubstituting this expression into (30) finally yields

$$\mathbf{v}_{\rm cm} = \frac{1}{4\alpha}\mathsf{E}_{\mathsf{W}|\mathsf{Z}^m=z}[\mathsf{W}] = \frac{\mathsf{E}_{\mathsf{W}|\mathsf{Z}^m=z}[\mathsf{W}]}{\left\langle \mathsf{E}_{\mathsf{W}|\mathsf{Z}^m=z}[\mathsf{W}], \mathbf{v}_{\rm cm}\right\rangle},$$

whose only solution is given by

$$\mathbf{v}_{\rm cm} = \frac{\mathsf{E}_{\mathsf{W}|\mathsf{Z}^m=z}[\mathsf{W}]}{\left\|\mathsf{E}_{\mathsf{W}|\mathsf{Z}^m=z}[\mathsf{W}]\right\|} .$$

A.2 Proof of Theorem 11

Proof Given $z = (x,y) \in \mathcal{Z}^m$ we consider the projection $\mathbf{P}_x : \mathcal{W} \to \mathcal{K}$ that maps all vectors of unit length in the linear span of the points $\{\phi(x_1), \ldots, \phi(x_m)\} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ and the projection $\mathbf{P}_x^\perp : \mathcal{W} \to \mathcal{K}$ that maps into the complement of the linear span of $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$. Thus, for any vector $\mathbf{w} \in \mathcal{K}$ we have $\mathbf{w} = \mathbf{P}_x(\mathbf{w}) + \mathbf{P}_x^\perp(\mathbf{w})$ which immediately implies that for all $(x_i, y_i) \in z$

$$\langle \mathbf{x}_i, \mathbf{w}\rangle = \left\langle \mathbf{x}_i, \mathbf{P}_x(\mathbf{w}) + \mathbf{P}_x^\perp(\mathbf{w})\right\rangle = \langle \mathbf{x}_i, \mathbf{P}_x(\mathbf{w})\rangle + \left\langle \mathbf{x}_i, \mathbf{P}_x^\perp(\mathbf{w})\right\rangle = \langle \mathbf{x}_i, \mathbf{P}_x(\mathbf{w})\rangle .$$

Suppose $\mathbf{w}_z$ is a minimiser of (16) but $\mathbf{P}_x(\mathbf{w}_z) \neq \mathbf{w}_z$, i.e. $\|\mathbf{P}_x(\mathbf{w}_z)\| < \|\mathbf{w}_z\| = 1$. Then

$$\begin{aligned} c(x,y,(\langle \mathbf{x}_1, \mathbf{w}_z\rangle), \ldots, \langle \mathbf{x}_m, \mathbf{w}_z\rangle) &= c(x,y,(\langle \mathbf{x}_1, \mathbf{P}_x(\mathbf{w}_z)\rangle), \ldots, \langle \mathbf{x}_m, \mathbf{P}_x(\mathbf{w}_z)\rangle) \\ &> c\left(x,y,\|\mathbf{P}_x(\mathbf{w}_z)\|^{-1}(\langle \mathbf{x}_1, \mathbf{P}_x(\mathbf{w}_z)\rangle), \ldots, \langle \mathbf{x}_m, \mathbf{P}_x(\mathbf{w}_z)\rangle\right) \\ &= c\left(x,y,\left(\left\langle \mathbf{x}_1, \frac{\mathbf{P}_x(\mathbf{w}_z)}{\|\mathbf{P}_{x(\mathbf{w}_z)}\|}\right\rangle\right), \ldots, \left\langle \mathbf{x}_m, \frac{\mathbf{P}_x(\mathbf{w}_z)}{\|\mathbf{P}_{x(\mathbf{w}_z)}\|}\right\rangle\right), \end{aligned}$$

where the second line follows from the assumption that c is strictly monotonically decreasing in the third argument. We see that $\mathbf{w}_z$ cannot be the minimiser of (16) and by contradiction it follows that the minimiser must admit the representation (17). ■

A.3 Sufficiency of the Linear Span — Proof of Theorem 12

Proof Let us rewrite the l.h.s. of (18)

$$\begin{aligned} \int_{\mathcal{W}} \|\mathbf{w}-\mathbf{v}\|^2 \, d\mathsf{P}_{\mathsf{W}|\mathsf{Z}^m=z}(\mathbf{v}) &= \int_{\mathcal{W}} 2(1 - \langle \mathbf{w}, \mathbf{v}\rangle) \, d\mathsf{P}_{\mathsf{W}|\mathsf{Z}^m=z}(\mathbf{v}) \\ &= 2 - 2\int_{\mathcal{W}} \sum_{\mathbf{b}\in\{-1,+1\}^m} \prod_{i=1}^m \mathbf{I}_{\mathrm{sign}(y_i\langle \mathbf{x}_i, \mathbf{v}\rangle)=b_i} \langle \mathbf{w}, \mathbf{v}\rangle \, d\mathsf{P}_{\mathsf{W}|\mathsf{Z}^m=z}(\mathbf{v}) \\ &= 2 - 2C \sum_{\mathbf{b}\in\{-1,+1\}^m} \underbrace{\int_{\mathcal{W}} \prod_{i=1}^m \mathbf{I}_{\mathrm{sign}(y_i\langle \mathbf{x}_i, \mathbf{v}\rangle)=b_i} \langle \mathbf{w}, \mathbf{v}\rangle f(b_i) \, d\mathbf{v}}_{A(\mathbf{w},z,\mathbf{b})}, \end{aligned} \tag{31}$$

where the first line follows by the assumption that $\mathbf{w},\mathbf{v}\in\mathcal{W}$, the second line is true because only one summation $\mathbf{b}$ leads to a non-zero product and the third line follows from

$$\begin{aligned} d\mathsf{P}_{\mathsf{W}|\mathsf{Z}^m=z}(\mathbf{v}) &= \frac{\prod_{i=1}^m f\left(\operatorname{sign}\left(y_i\left\langle \mathbf{x}_i,\mathbf{v}\right\rangle\right)\right)\cdot\frac{1}{\int_{\mathcal{W}} d\tilde{\mathbf{w}}}}{\int_{\mathcal{W}}\prod_{i=1}^m f\left(\operatorname{sign}\left(y_i\left\langle \mathbf{x}_i,\mathbf{u}\right\rangle\right)\right)\frac{1}{\int_{\mathcal{W}} d\tilde{\mathbf{w}}}\,d\mathbf{u}}\,d\mathbf{v} \\ &= \frac{\prod_{i=1}^m f\left(\operatorname{sign}\left(y_i\left\langle \mathbf{x}_i,\mathbf{v}\right\rangle\right)\right)}{\int_{\mathcal{W}}\prod_{i=1}^m f\left(\operatorname{sign}\left(y_i\left\langle \mathbf{x}_i,\mathbf{u}\right\rangle\right)\right)d\mathbf{u}}\,d\mathbf{v} = C\cdot\prod_{i=1}^m f\left(b_i\right)d\mathbf{v}\,. \end{aligned}$$

Let us consider the expression $A(\mathbf{w},z,\mathbf{b})$. For a fixed set $z=(x,y)\in\mathcal{Z}^m$ let $\mathbf{P}_x:\mathcal{W}\to\mathcal{K}$ and $\mathbf{P}_x^{\perp}:\mathcal{W}\to\mathcal{K}$ be the projection of unit length vectors into $\mathcal{L}_x$ and its orthogonal complement, respectively. Then, by construction we know that

$$y_i\left\langle \mathbf{x}_i,\mathbf{v}\right\rangle = y_i\left\langle \mathbf{x}_i,\mathbf{P}_x(\mathbf{v})+\mathbf{P}_x^{\perp}(\mathbf{v})\right\rangle = y_i\left\langle \mathbf{x}_i,\mathbf{P}_x(\mathbf{v})\right\rangle + y_i\left\langle \mathbf{x}_i,\mathbf{P}_x^{\perp}(\mathbf{v})\right\rangle = y_i\left\langle \mathbf{x}_i,\mathbf{P}_x(\mathbf{v})\right\rangle\,.$$

As a consequence,

$$\operatorname{sign}\left(y_i\left\langle \mathbf{x}_i,\mathbf{P}_x(\mathbf{v})+\mathbf{P}_x^{\perp}(\mathbf{v})\right\rangle\right)=b_i \;\Leftrightarrow\; \operatorname{sign}\left(y_i\left\langle \mathbf{x}_i,\mathbf{P}_x(\mathbf{v})-\mathbf{P}_x^{\perp}(\mathbf{v})\right\rangle\right)=b_i\,, \tag{32}$$

which implies that

$$A(\mathbf{w},z,\mathbf{b}) = \int_{\mathcal{W}}\prod_{i=1}^m \mathbf{I}_{\operatorname{sign}(y_i\langle \mathbf{x}_i,\mathbf{P}_x(\mathbf{v})\rangle)=b_i}\left\langle \mathbf{w},\mathbf{P}_x(\mathbf{v})\right\rangle f(b_i)\,d\mathbf{v}\,, \tag{33}$$

because by (32) all the inner products with orthogonal components are vanishing. Noticing that $\forall\mathbf{v}\in\mathcal{W}:\|\mathbf{P}_x(\mathbf{v})\|\le 1$ we can rewrite (33) as

$$\begin{aligned} A(\mathbf{w},z,\mathbf{b}) &= \int_{\mathcal{K}\setminus\mathcal{L}_x}\int_{\mathcal{L}_x}\mathbf{I}_{\|\mathbf{v}+\mathbf{u}\|=1}\prod_{i=1}^m \mathbf{I}_{\operatorname{sign}(y_i\langle \mathbf{x}_i,\mathbf{u}\rangle)=b_i}\left\langle \mathbf{w},\mathbf{u}\right\rangle f(b_i)\,d\mathbf{u}\,d\mathbf{v} \\ &= \int_0^1\left(\int_{\mathcal{K}\setminus\mathcal{L}_x}\mathbf{I}_{\|\mathbf{v}\|=1-r}\,d\mathbf{v}\right)\int_{\mathcal{L}_x}\mathbf{I}_{\|\mathbf{u}\|=r}\prod_{i=1}^m \mathbf{I}_{\operatorname{sign}(y_i\langle \mathbf{x}_i,\mathbf{u}\rangle)=b_i}\left\langle \mathbf{w},\mathbf{u}\right\rangle f(b_i)\,d\mathbf{u}\,dr \end{aligned}$$

Lastly, for all $\mathbf{u}$ with $\|\mathbf{u}\|=r$ we use the fact that

$$\begin{aligned} \prod_{i=1}^m \mathbf{I}_{\operatorname{sign}(y_i\langle \mathbf{x}_i,\mathbf{u}\rangle)=b_i}\left\langle \mathbf{w},\mathbf{u}\right\rangle &= r\cdot\prod_{i=1}^m \mathbf{I}_{\operatorname{sign}\left(r\cdot y_i\left\langle \mathbf{x}_i,\frac{\mathbf{u}}{r}\right\rangle\right)=b_i}\left\langle \mathbf{w},\frac{\mathbf{u}}{r}\right\rangle \\ &= r\cdot\prod_{i=1}^m \mathbf{I}_{\operatorname{sign}\left(y_i\left\langle \mathbf{x}_i,\frac{\mathbf{u}}{r}\right\rangle\right)=b_i}\left\langle \mathbf{w},\frac{\mathbf{u}}{r}\right\rangle\,, \end{aligned} \tag{34}$$

that is the feasibility of a point $\mathbf{u}$ does not change under rescaling of $\mathbf{u}$. Combining (34), (33) and (31) we have shown that there exists a constant $C\in\mathbb{R}^+$ such that

$$\int_{\mathcal{W}}\|\mathbf{w}-\mathbf{v}\|^2\,d\mathsf{P}_{\mathsf{W}|\mathsf{Z}^m=z}(\mathbf{v}) = C\cdot\int_{\mathcal{W}_x}\|\mathbf{w}-\mathbf{v}\|^2\,d\mathsf{P}_{\mathsf{W}|\mathsf{Z}^m=z}(\mathbf{v})\,.$$

■

A.4 A derivation of the operation $\oplus_\mu$

Let us derive operation $\oplus_\mu$ acting on vectors of unit length. This function has to have the following properties (see Section 3.1)

$$\left\|\mathbf{s}\oplus_\mu\mathbf{t}\right\|^2 = 1\,, \tag{35}$$

$$\left\|\mathbf{t}-\mathbf{s}\oplus_\mu\mathbf{t}\right\| = \mu\left\|\mathbf{t}-\mathbf{s}\right\|\,, \tag{36}$$

$$\mathbf{s}\oplus_\mu\mathbf{t} = \rho_1\mathbf{s}+\rho_2\mathbf{t}\,, \tag{37}$$

$$\rho_1\ge 0 \quad,\quad \rho_2\ge 0\,. \tag{38}$$

Here we assume that $\|\mathbf{s}\|^2 = \|\mathbf{t}\|^2 = 1$. Inserting equation (37) into (35) results in

$$\|\rho_1\mathbf{s}+\rho_2\mathbf{t}\|^2 = \langle\rho_1\mathbf{s}+\rho_2\mathbf{t},\rho_1\mathbf{s}+\rho_2\mathbf{t}\rangle = \rho_1^2+\rho_2^2+2\rho_1\rho_2\langle\mathbf{s},\mathbf{t}\rangle = 1\,. \tag{39}$$

In a similar fashion combining equation (37) and (36) gives

$$\begin{aligned}
\left\|\mathbf{t}-\mathbf{s}\oplus_\mu\mathbf{t}\right\|^2 &= \mu^2\,\|\mathbf{t}-\mathbf{s}\|^2\\
\|(1-\rho_2)\,\mathbf{t}-\rho_1\mathbf{s}\|^2 &= \mu^2\,\|\mathbf{t}-\mathbf{s}\|^2\\
(1-\rho_2)^2-2\,(1-\rho_2)\,\rho_1\,\langle\mathbf{s},\mathbf{t}\rangle+\rho_1^2 &= 2\mu^2\,(1-\langle\mathbf{s},\mathbf{t}\rangle)\,.
\end{aligned} \tag{40}$$

Note that equation (39) is quadratic in ρ_2 and has the following solution

$$\rho_2 = -\rho_1\,\langle\mathbf{s},\mathbf{t}\rangle\underbrace{\pm\sqrt{\rho_1^2\,(\langle\mathbf{s},\mathbf{t}\rangle)^2-\rho_1^2+1}}_{A}\,. \tag{41}$$

Let us insert equation (41) into the l.h.s. of equation (40). This gives the following quadratic equation in ρ_1

$$\begin{aligned}
(1-\rho_2)^2-2\,(1-\rho_2)\,\rho_1\,\langle\mathbf{s},\mathbf{t}\rangle+\rho_1^2 &=\\
(1+\rho_1\,\langle\mathbf{s},\mathbf{t}\rangle-A)\,(1-A-\rho_1\,\langle\mathbf{s},\mathbf{t}\rangle)+\rho_1^2 &=\\
(1-A)^2-(\rho_1\,\langle\mathbf{s},\mathbf{t}\rangle)^2+\rho_1^2 &=\\
2-2A &= 2\mu^2\,(1-\langle\mathbf{s},\mathbf{t}\rangle)\,.
\end{aligned}$$

Solving this equation for ρ_1 results in

$$\rho_1 = \mu\sqrt{-\frac{\mu^2-\mu^2\,\langle\mathbf{s},\mathbf{t}\rangle-2}{\langle\mathbf{s},\mathbf{t}\rangle+1}}\,.$$

Inserting this formula back into equation (41) we obtain

$$\rho_2 = -\rho_1\,\langle\mathbf{s},\mathbf{t}\rangle\pm\left(\mu^2\,(1-\langle\mathbf{s},\mathbf{t}\rangle)-1\right)\,.$$

Appendix B. Algorithms

Algorithm 1 Dual perceptron algorithm with permutation

Require: A permutation $\Pi : \{1,\ldots,m\} \to \{1,\ldots,m\}$
Ensure: Existence of a version space $V(z)$, a linearly separable training sample in feature space
  $\alpha = \mathbf{o} = 0$
  repeat
    for $i = 1,\ldots,m$ **do**
      if $y_{\Pi(i)} o_{\Pi(i)} \leq 0$ **then**
        $\alpha_i \leftarrow \alpha_i + y_{\Pi(i)}$
        for $j = 1,\ldots,m$ **do**
          $o_{\Pi(j)} \leftarrow o_{\Pi(j)} + y_{\Pi(i)} k\left(x_{\Pi(i)}, x_{\Pi(j)}\right)$
        end for
      end if
    end for
  until the if branch was never entered within the **for** loop
  return the expansion coefficients α

Algorithm 2 Kernel billiard algorithm (in dual variables)

Require: A tolerance TOL $\in [0,1]$ and $\tau_{\max} \in \mathbb{R}^+$
Require: Existence of a version space $V(z)$, a linearly separable training sample in feature space
Ensure: for all $i = 1,\ldots,m$, $y_i \sum_{j=1}^{m} \gamma_j k(x_i, x_j) > 0$

```
α = 0, β = random, normalise β using equation (20)
Ξ = ξ_max = 0, p_min = 1
while ρ_2(p_min, Ξ/(Ξ + ξ_max)) > TOL do
  repeat
    for i = 1,...,m do
      d_i = Σ_{j=1}^m γ_j k(x_j, x_i), ν_i = Σ_{j=1}^m β_j k(x_j, x_i)
      τ_i = −d_i/ν_i
    end for
    c' = argmin_{i:τ_i>0} τ_i
    if τ_{c'} ≥ τ_max then
      β = random, but fulfils equation (22), normalise β using equation (20)
    else
      c = c'
    end if
  until τ_{c'} < τ_max
  γ' = γ + τ_c β, normalise γ' using equation (20)
  β_c = β_c − 2ν_c/k(x_c, x_c)
  ζ = γ + γ', normalise ζ using equation (20)
  ξ = sqrt( Σ_{i=1}^m Σ_{j=1}^m (γ_i − γ'_i)(γ_j − γ'_j) k(x_i, x_j) )
  p = Σ_{i=1}^m Σ_{j=1}^m ζ_i α_j k(x_i, x_j)
  α = ρ_1(p, Ξ/(Ξ+ξ)) α + ρ_2(p, Ξ/(Ξ+ξ)) ζ
  p_min = min(p, p_min), ξ_max = max(ξ, ξ_max), Ξ = Ξ + ξ, γ = γ'
end while
return the expansion coefficients α
```

In mathematical notation:

$\alpha = 0$, $\beta =$ random, normalise β using equation (20)
$\Xi = \xi_{\max} = 0$, $p_{\min} = 1$
while $\rho_2\left(p_{\min}, \Xi/\left(\Xi + \xi_{\max}\right)\right) > \text{TOL}$ **do**
  repeat
    for $i = 1,\ldots,m$ **do**
      $d_i = \sum_{j=1}^{m} \gamma_j k(x_j, x_i)$, $\nu_i = \sum_{j=1}^{m} \beta_j k(x_j, x_i)$
      $\tau_i = -d_i/\nu_i$
    end for
    $c' = \operatorname{argmin}_{i:\tau_i>0} \tau_i$
    if $\tau_{c'} \geq \tau_{\max}$ **then**
      $\beta =$ random, but fulfils equation (22), normalise β using equation (20)
    else
      $c = c'$
    end if
  until $\tau_{c'} < \tau_{\max}$
  $\gamma' = \gamma + \tau_c \beta$, normalise γ' using equation (20)
  $\beta_c = \beta_c - 2\nu_c / k(x_c, x_c)$
  $\zeta = \gamma + \gamma'$, normalise ζ using equation (20)
  $\xi = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{m} (\gamma_i - \gamma'_i)\left(\gamma_j - \gamma'_j\right) k(x_i, x_j)}$
  $p = \sum_{i=1}^{m} \sum_{j=1}^{m} \zeta_i \alpha_j k(x_i, x_j)$
  $\alpha = \rho_1\left(p, \frac{\Xi}{\Xi+\xi}\right) \alpha + \rho_2\left(p, \frac{\Xi}{\Xi+\xi}\right) \zeta$
  $p_{\min} = \min(p, p_{\min})$, $\xi_{\max} = \max(\xi, \xi_{\max})$, $\Xi = \Xi + \xi$, $\gamma = \gamma'$
end while
return the expansion coefficients α

References

J. Aitchison. Bayesian tolerance regions (with discussion). *Journal of the Royal Statistical Society – Series B*, 26:161–175, 1964.

C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the Annual Conference on Computational Learning Theory*, pages 144–152, Pittsburgh, PA, July 1992. ACM Press.

W. Buntine. *A Theory of Learning Classification Rules*. PhD thesis, University of Technology, Sydney, Autralia, 1992.

C. Cortes. *Prediction of Generalization Ability in Learning Machines*. PhD thesis, Department of Computer Science, University of Rochester, 1995.

R. Cox. Probability, frequency, and reasonable expectations. *American Journal of Physics*, 14:1–13, 1946.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, UK, 2000.

D. DeCoste and B. Schölkopf. Training invariant support vector machines. *Machine Learning*, 2002. Accepted for publication. Also: Technical Report JPL-MLTR-00-1, Jet Propulsion Laboratory, Pasadena, CA, 2000.

T. G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1998.

W. Feller. *An Introduction To Probability Theory and Its Application*, volume 2. John Wiley and Sons, New York, 1966.

T. Graepel and R. Herbrich. The kernel Gibbs sampler. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 514–520, Cambridge, MA, 2001. MIT Press.

T. Graepel, R. Herbrich, and K. Obermayer. Bayesian Transduction. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 456–462, Cambridge, MA, 2000. MIT Press.

D. Haussler. Convolutional kernels on discrete structures. Technical Report UCSC-CRL-99-10, Computer Science Department, University of California at Santa Cruz, 1999.

R. Herbrich. *Learning Kernel Classifiers: Theory and Algorithms*. MIT Press, 2001. in press.

R. Herbrich and T. Graepel. A PAC-Bayesian margin bound for linear classifiers: Why SVMs work. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 224–230, Cambridge, MA, 2001. MIT Press.

R. Herbrich, T. Graepel, and C. Campbell. Bayes point machines: Estimating the Bayes point in kernel space. In *Proceedings of IJCAI Workshop Support Vector Machines*, pages 23–27, 1999a.

R. Herbrich, T. Graepel, and C. Campbell. Bayesian learning in reproducing kernel Hilbert spaces. Technical report, Technical University of Berlin, 1999b. TR 99-11.

R. Herbrich, T. Graepel, and C. Campbell. Robust Bayes point machines. In *Proceedings of ESANN 2000*, pages 49–54, 2000a.

R. Herbrich, T. Graepel, and J. Shawe-Taylor. Sparsity vs. large margins for linear classifiers. In *Proceedings of the Annual Conference on Computational Learning Theory*, pages 304–308, 2000b.

T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning*, pages 137–142, Berlin, 1998. Springer.

T. Joachims. Making large–scale SVM learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 169–184, Cambridge, MA, 1999. MIT Press.

N. Littlestone and M. Warmuth. Relating data compression and learnability. Technical report, University of California Santa Cruz, 1986.

D. J. C. MacKay. *Bayesian Methods for Adaptive Models*. PhD thesis, Computation and Neural Systems, California Institute of Technology, Pasadena, CA, 1991.

D. A. McAllester. Some PAC Bayesian theorems. In *Proceedings of the Annual Conference on Computational Learning Theory*, pages 230–234, Madison, Wisconsin, 1998. ACM Press.

D. A. McAllester. PAC-Bayesian model averaging. In *Proceedings of the Annual Conference on Computational Learning Theory*, pages 164–170, Santa Cruz, USA, 1999.

J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, London*, A 209:415–446, 1909.

C. A. Micchelli. Algebraic aspects of interpolation. *Proceedings of Symposia in Applied Mathematics*, 36: 81–102, 1986.

T. Minka. *Expectation Propagation for approximative Bayesian inference*. PhD thesis, MIT Media Labs, Cambridge, USA, 2001.

T. M. Mitchell. Version spaces: a candidate elimination approach to rule learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 305–310, Cambridge, Massachusetts, 1977. IJCAI.

T. M. Mitchell. Generalization as search. *Artificial Intelligence*, 18(2):202–226, 1982.

R. Neal. *Bayesian Learning in Neural Networks*. Springer, 1996.

R. M. Neal. Markov chain Monte Carlo method based on 'slicing' the density function. Technical report, Department of Statistics, University of Toronto, 1997. TR–9722.

A. B. J. Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 12, pages 615–622. Polytechnic Institute of Brooklyn, 1962.

M. Opper and D. Haussler. Generalization performance of Bayes optimal classification algorithms for learning a perceptron. *Physical Review Letters*, 66:2677, 1991.

M. Opper and W. Kinzel. *Statistical Mechanics of Generalisation*, page 151. Springer, 1995.

M. Opper, W. Kinzel, J. Kleinz, and R. Nehl. On the ability of the optimal perceptron to generalize. *Journal of Physics A*, 23:581–586, 1990.

M. Opper and O. Winther. Gaussian processes for classification: Mean field algorithms. *Neural Computation*, 12(11):2655–2684, 2000.

E. E. Osuna, R. Freund, and F. Girosi. Support vector machines: Training and applications. Technical report, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1997. AI Memo No. 1602.

E. J. G. Pitman. The estimation of the location and scale parameters of a continuous population of any given form. *Biometrika*, 30:391–421, 1939.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208, Cambridge, MA, 1999. MIT Press.

J. Platt. Probabilities for SV machines. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–73, Cambridge, MA, 2000. MIT Press.

G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for adaboost. *Machine Learning*, 42(3):287–320, 2001.

P. Ruján. Playing billiards in version space. *Neural Computation*, 9:99–122, 1997.

P. Ruján and M. Marchand. Computing the Bayes kernel classifier. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 329–347, Cambridge, MA, 2000. MIT Press.

M. Rychetsky, J. Shawe-Taylor, and M. Glesner. Direct Bayes point machines. In *Proceedings of the International Conference on Machine Learning*, 2000.

B. Schölkopf. *Support Vector Learning*. R. Oldenbourg Verlag, München, 1997. Doktorarbeit, TU Berlin. Download: http://www.kernel-machines.org.

B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In *Proceedings of the Annual Conference on Computational Learning Theory*, 2001.

B. Schölkopf, S. Mika, C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. Smola. Input space vs. feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5):1000–1017, 1999.

J. Shawe-Taylor, P. L. Bartlett, R. C. Williamson, and M. Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory*, 44(5):1926–1940, 1998.

J. Shawe-Taylor and N. Cristianini. Margin distribution and soft margin. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 349–358, Cambridge, MA, 2000. MIT Press.

A. J. Smola. *Learning with Kernels*. PhD thesis, Technische Universität Berlin, 1998. GMD Research Series No. 25.

P. Sollich. Probabilistic methods for support vector machines. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 349–355, Cambridge, MA, 2000. MIT Press.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995. ISBN 0-387-94559-8.

V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, New York, 1998.

G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, Philadelphia, 1990.

T. Watkin. Optimal learning with a neural network. *Europhysics Letters*, 21:871, 1993.

C. K. I. Williams. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In M. Jordan, editor, *Learning and Inference in Graphical Models*, pages 599–621. MIT Press, 1999.

C. K. I. Williams and D. Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI*, 20(12):1342–1351, 1998.