Université Pierre et Marie Curie – Paris VI

Support Vector Machines: Principes d'induction, Réglage automatique et Connaissances a priori

---

Support Vector Machines: Induction Principles, Adaptive Tuning and Prior Knowledge

THÈSE

remise le 30 juin 2002,
présentée et soutenue publiquement le 2 avril 2004

pour l'obtention du

Doctorat de l'Université Pierre et Marie Curie – Paris VI

(Spécialité Informatique)

par

Olivier Chapelle

Composition du jury

Rapporteurs :	S. Canu B. Schölkopf
Examinateurs :	R. Gilleron J.-P. Nadal V. Vapnik
Directeur de thèse :	P. Gallinari

Laboratoire d'Informatique de Paris 6 — UMR 7606

Mis en page avec la classe thloria.

Acknowledgments

First and foremost, many thanks to my PhD supervisors: Patrick Gallinari for his unconditional support and Vladimir Vapnik for inspiring me, for his valuable insights, patience and guidance. Vladimir made me discover the wonderful world of learning theory and I am very honoured to have had the opportunity to work with him.

I would also like to thank Stéphane Canu and Bernhard Schölkopf for having accepted to review this thesis and giving me valuable comments. Thank you also to Rémi Gilleron and Jean-Pierre Nadal for having made me the honour to take part in the jury.

Most of the research presented in this thesis was conducted during my several internships at AT&T in the group headed by Yann LeCun. I am really indebted to Yann for providing me the opportunity to participate in one of the leading research group for machine learning in the world. This was a very exciting moment in my research life and I received valuable insights from all members of the group, especially Yoshua Bengio, Léon Bottou, Patrick Haffner and Yann LeCun. I would like to express my gratitude to all of them for their invaluable advice, collaboration, feedback and support which founded and upheld my knowledge in machine learning. Also, thank you to the other interns who were with me at AT&T during that time. I had very fruitful discussions with them, especially with Jason Weston, Pascal Vincent and Sayan Mukherjee.

I have also spent the last year of my PhD working for Biowulf technologies. It was a genuine pleasure and learning experience to have been part of this excellent research environment. I would like to thank in particular the "New York team": Olivier Bousquet, André Elisseeff, Bernhard Schölkopf and Jason Weston: not only have I had excellent and stimulating collaborations with them, but I have also discovered true friends.

I would like to express my gratitude to Pascal Massart who had patiently taken his time to explain me the tools of concentration inequalities during several visits at Orsay.

Last but not least, I would like to thank my family and my wife for their love and unconditional support.

Contents

Résumé

Le but principal de l'apprentissage automatique est d'estimer une fonction à partir de données d'entraînement. Parmi les charactéristiques recherchées dans un bon algorithme d'apprentissage, on peut noter: une solide justification théorique basée sur un principe d'induction, une méthode efficace pour ajuster les hyper-paramètres ou encore la possibilité d'inclure des connaissances a priori.

Dans cette thèse, nous étudierons ces trois aspects pour différents algorithmes d'apprentissage, mais nous insisterons plus particulièrement sur les Support Vector Machines (SVM), qui ont été développées dans les années 90 [Boser et al., 1992] et sont depuis largement utilisées. Cette thèse est ainsi divisée en trois parties:

1. La première tente d'apporter une réponse à une question centrale dans la théorie statistique de l'apprentissage: " Étant donné une classe de fonctions, comment choisir la meilleur fonction dans cet ensemble ?" Un principe d'induction est un moyen de répondre à cette question. Dans le chapitre 1, nous introduisons un nouveau principe d'induction et montrons dans le chapitre 2 qu'il est étroitement relié aux SVMs. Inspiré de ce principe, nous suggérons aussi quelques modifications à la fonction de coût utilisée par les SVMs.

2. La première partie n'a pas abordé une question importante: "Quelle classe de fonctions faut-il choisir ?". C'est le problème de la sélection de modèle. Dans le cadre des SVMs, cela revient à choisir un noyau approprié. Pour cela, nous proposons tout d'abord d'estimer ou de borner l'erreur de généralisation d'une SVM (chapitre 3), puis de la minimiser par une descente en gradient sur les paramètres du noyau (chapitre 4). Cette technique peut alors être appliquée à des problèmes tels que la sélection de composantes ou la détection de points anormaux (chapitre 5).

3. En général, la plupart des algorithmes d'apprentissage ne reçoivent qu'un sous-ensemble de données étiquetées pour mener à bien leur inférence. Ce cadre a été celui des deux premières parties. Cependant, quand des informations supplémentaires sont disponibles, les performances peuvent être améliorées. Dans cette troisième partie, nous montrerons comme tirer parti de la connaissance de transformations invariantes (chapitre 6) et de points non étiquetés (chapitre 7). Lorsque certains de ces points non étiquetés peuvent être sélectionés pour étiquettage, on parle d'apprentissage actif (chapitre 8). Enfin, le chapitre 9 porte sur l'étude de la sélection de modèles en régression. Nous y proposons une méthode qui elle aussi utilise des points non étiquetés.

Summary

The main goal of machine learning is to estimate a function based on some training data. Desirable features of a successful learning algorithm include: a solid theoretical motivation based on an induction principle, an efficient way of tuning the hyperparameters and also the possibility to include prior knowledge.

In this thesis, we will study these three aspects of different learning algorithms, but we will mostly focus on the Support Vector Machine (SVM) one, which has been introduced in the mid-90s [Boser et al., 1992] and has become popular since then. This thesis is thus divided in 3 parts:

1. The first one tries to bring some answer to one of the central questions in statistical learning theory: "given a class of functions, how to choose the optimal function in this set ?" An induction principle is supposed to give an answer to this question. In chapter 1, we introduce a new induction principle and show in chapter 2 that it has strong links with SVMs. Based on this principle, we also suggest some modifications of the SVM algorithm.

2. The first part left an important question unsolved: "What class of functions should be chosen ?". That is the model selection problem. In the SVM framework, this translates to the problem of selecting an appropriate kernel function. We propose to do so by first estimating or bounding the generalization error of an SVM (chapter 3) and then performing a gradient descent on the kernel parameters (chapter 4). This technique leads to some direct applications such as features selection or outlier detection (chapter 5).

3. In the classical framework of learning theory, only a set of labeled examples is given to the learning algorithm. This was the setting of the first two parts. However, when additional information is available, performances can be enhanced. In this third part, we will show how to benefit from known invariant transformations (chapter 6) and unlabeled data (chapter 7). If some of the unlabeled data can be queried to be labeled, this is the active learning strategy, which we will study in chapter 8. Finally, chapter 9 investigates the problem of model selection in regression. We propose a method which makes also use of unlabeled points.

Part I

Induction principles and algorithms

Chapter 1

Learning theory

1.1 Introduction to statistical learning theory

1.1.1 Introduction and definitions

According to Vapnik [1998], there are three main problems in machine learning: classification, regression and density estimation. In all cases, the goal is to infer a function from a sample data.

In the *supervised learning* paradigm, training data is comprised of pairs of input/output points $(\mathbf{x}_i, y_i)$, $1 \leq i \leq n$, with $\mathbf{x}_i$ belonging to some space $\mathcal{X}$ and $y_i \in \mathbb{R}$ (regression) or $y_i \in \{-1, 1\}$ (binary classification).

In the *unsupervised learning* paradigm, the two main tasks are density estimation and clustering while the training data is only made of input points $\mathbf{x}_i$. The goal in this case is to infer the underlying structure of the data.

Supervised learning

In the framework of *Statistical Learning Theory* pioneered by Vapnik and Chervonenkis [1971], also refereed to as the PAC model [Valiant, 1984], the training data is supposed to be generated i.i.d. from an unknown distribution $P(\mathbf{x}, y)$.

The dependency between the input $\mathbf{x}$ and the output y is thus encoded in this distribution. For example, if the input points follow a probability distribution $P(\mathbf{x})$ and if the output associated with a point $\mathbf{x}$ is $f(\mathbf{x})$, altered by a gaussian noise of variance σ^2, then

$$P(\mathbf{x}, y) = P(\mathbf{x})\mathcal{N}_\sigma(f(\mathbf{x}) - y),$$

where $\mathcal{N}_\sigma$ is a Gaussian distribution with density $\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{t^2}{2\sigma^2}\right)$.

The quality of a function attempting to model this relationship is measured through the expectation of a loss function with respect to $P(\mathbf{x}, y)$. The traditional loss functions are:

- for classification, $\ell(f(\mathbf{x}), y) = \mathrm{I}_{f(\mathbf{x}) \neq y}$, where I is the indicator function: $\mathrm{I}_A = 1 \Leftrightarrow A$ true.
- for regression, $\ell(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2$.

Since one of the assumption in this framework is that future data whose outputs need to be predicted are also generated from the same distribution, the goal of statistical learning is to find the function minimizing the expected loss,

$$R(f) = \int \ell(\mathbf{x}, f(\mathbf{x}))dP(\mathbf{x}, y). \tag{1.1}$$

The expected loss of a function f is also called the *risk*.

Unsupervised learning

In the density estimation problem, one may consider the Kullback-Liebler distance and its associated loss function

$$\ell(x, f) = -\log f(x).$$

The expected loss

$$R(f) = \int \ell(\mathbf{x}, f) dP(\mathbf{x}) \tag{1.2}$$

is minimized for $f = P$.

1.1.2 Empirical Risk Minimization

From here on, we will only consider the classification problem, but substantially similar results can usually be derived for regression and density estimation.

The learning problem boils down to the search of the function $f \in \mathcal{F}$ that minimizes the expectation (1.1) of the indicator loss function $\ell(f(\mathbf{x}), y) = \mathrm{I}_{f(\mathbf{x}) \neq y}$. This expectation cannot be computed since the distribution $P(\mathbf{x}, y)$ is unknown. However, given a training set $\{(\mathbf{x}_i, y_i)\}_{1 \leq i \leq n}$, it is common to minimize the *empirical risk* instead,

$$R_{emp}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(\mathbf{x}_i), y_i) \tag{1.3}$$

Note that according to the law of large numbers,

$$\forall f, R_{emp}(f) \xrightarrow[n \to \infty]{} R(f).$$

This convergence is the main motivation for the *Empirical Risk Minimization* (ERM) induction principle: one hopes that the function minimizing the empirical risk will also have a small risk.

However, there is an obvious weakness in this approach if the set of functions, on which this minimization is performed, is not constrained. Indeed, for any training set, there is an infinite number of functions that can learn perfectly the training data, i.e. $f(\mathbf{x}_i) = y_i, \ \forall i$. Among those functions, there are as many functions predicting the label 1 on an unseen example than functions predicting -1. This observation is the core of the so-called *No Free Lunch* theorem [Wolpert, 1996].

For this reason, the set of functions $\mathcal{F}$ on which the empirical risk minimization is performed should be restrained. Usually, a set of functions with smooth decision boundaries is chosen since in real-world problems, the true underlying decision boundary is indeed smooth.

More formally, if the set of functions $\mathcal{F}$ is small enough, then there is uniform convergence of the empirical risks to the the true risks and learning occurs (see figure 1.1),

$$P\left\{ \sup_{f \in \mathcal{F}} |R(f) - R_{emp}(f)| > \varepsilon \right\} \xrightarrow[n \to \infty]{} 0. \tag{1.4}$$

It is easy to see that if (1.4) holds true, then the ERM principle is consistent, which means that the risk of the empirical risk minimizer f_n tends in probability to the risk of the best function $f^* \in \mathcal{F}$ as n goes to infinity.

Indeed,

$$0 \leq R(f_n) - R(f^*) = \underbrace{R(f_n) - R_{emp}(f_n)}_{\to 0} + \underbrace{R_{emp}(f_n) - R_{emp}(f^*)}_{\leq 0} + \underbrace{R_{emp}(f^*) - R(f^*)}_{\to 0}.$$

It turns out that one-sided uniform convergence is a necessary and sufficient condition for the ERM to be consistent [Vapnik and Chervonenkis, 1991].

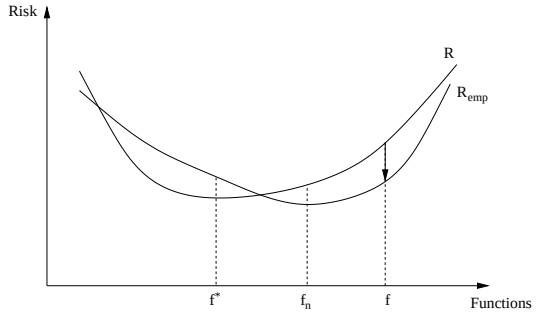


Figure 1.1: Representation of the empirical risk convergence: the x-axis represents the function class. The law of large numbers implies that there is a point-wise convergence of the empirical risk to the true risk (as the sample size goes to infinity). However, we would like that the risk of the empirical risk minimizer converges to the minimal risk in the function class. For this, a uniform convergence statement is needed.

1.1.3 VC theory

Vapnik and Chervonenkis studied in the 70s the ERM principle through uniform convergence bounds [Vapnik and Chervonenkis, 1971]. Unlike most related works in the same period, they were able to derive *non asymptotic* bounds, which can be useful in practice.

As stated below, the bounds depend on two terms: the minimum of the empirical risk and the *complexity* of the class of functions $\mathcal{F}$. The optimal trade-off between these two terms is one of the core problems in machine learning, the so-called bias/variance dilemma [Geman et al., 1995]: if $\mathcal{F}$ is too large, ERM *overfits* which means that the training data has been perfectly learned, but the value of the function at unseen points is almost random and there is no generalization. On the other hand, if $\mathcal{F}$ is too small, it is likely that none of functions in $\mathcal{F}$ could be an appropriate approximation of the true underlying function.

Vapnik and Chervonenkis introduced the following measure of complexity of a class of function $\mathcal{F}$

Definition 1.1 (VC dimension, Vapnik and Chervonenkis 1971) *The VC dimension h of a class of functions $\mathcal{F}$ is defined as the maximum number of points that can be learned exactly (shattered) by a function of $\mathcal{F}$,*

$$h = \max\left\{|X|, X \subset \mathcal{X},\ \textit{such that } \forall b \in \{-1,1\}^{|X|}, \exists f \in \mathcal{F} \ /\ \forall \mathbf{x}_i \in X,\ f(\mathbf{x}_i) = b_i\right\}$$

VC dimension of hyperplanes How many points can be shattered (i.e. separated perfectly whatever the labels are) in $\mathbb{R}^2$ by lines ? 3 points not aligned can be separated in all ways, but 4 points cannot: for example, an XOR cannot be realized by a linear function. Thus the VC dimension of lines in $\mathbb{R}^2$ is 3.

More generally, the following theorem holds

Theorem 1.2 (VC dimension of hyperplanes, Cover 1965; Burges 1998; Vapnik 1998) *Let $\mathcal{F}$ be the set of hyperplanes in $\mathbb{R}^d$,*

$$\mathcal{F} = \{\mathbf{x} \mapsto sgn(\mathbf{w} \cdot \mathbf{x} + b),\ \mathbf{w} \in \mathbb{R}^d,\ b \in \mathbb{R}\}.$$

The VC dimension of $\mathcal{F}$ is $d+1$.

Having defined the VC dimension, we can now introduce one of the key theorem proved by Vapnik and Chervonenkis.

Theorem 1.3 (Vapnik and Chervonenkis, Vapnik 1998) *Let $\mathcal{F}$ be a class of function of VC dimension h, then for any distribution P and for any sample $\{(\mathbf{x}_i, y_i)\}_{1 \leq i \leq n}$ drawn from this distribution, the following inequality holds true*

$$P\left\{\sup_{f\in\mathcal{F}}|R(f)-R_{emp}(f)|>\varepsilon\right\}<4\exp\left\{h\left(1+\log\left(\frac{2n}{h}\right)\right)-\left(\varepsilon-\frac{1}{n}\right)^2 n\right\}$$

This theorem leads directly to a bound on the risk of any function $f\in\mathcal{F}$ and especially the one minimizing the empirical risk:

Corollary 1.4 *Let $\mathcal{F}$ be a class of function of VC dimension h, then for any distribution P and for any sample $\{(\mathbf{x}_i,y_i)\}_{1\leq i\leq n}$ drawn from this distribution, the following inequality holds true with probability $1-\eta$*

$$\forall f\in\mathcal{F},\quad R(f)\leq R_{emp}(f)+\sqrt{\frac{h\left(\log\frac{2n}{h}+1\right)-\log\left(\frac{\eta}{4}\right)}{n}}+\frac{1}{n}$$

Note from theorem 1.3 that as soon as the VC dimension h is finite, uniform convergence holds and the ERM principle is consistent. Vapnik and Chervonenkis showed that it is also a necessary condition [Vapnik and Chervonenkis, 1991].

To understand how to derive a VC bound like the one presented in theorem 1.3, we will sketch a proof of a similar bound. A detailed derivation can be found in [Devroye and Lugosi, 2001]

First note that when the class of functions is finite, combining the Hoeffding's inequality [Hoeffding, 1963] with the union bound gives

$$P\left\{\sup_{f\in\mathcal{F}}|R(f)-R_{emp}(f)|>\varepsilon\right\}\leq 2|\mathcal{F}|\exp(-2n\varepsilon^2)$$

When $\mathcal{F}$ is infinite, the following steps lead to the desired result.

1. From McDiarmid's inequality [McDiarmid, 1989], note that the supremum of the deviations concentrate around its mean,

$$P\left\{\left|\sup_{f\in\mathcal{F}}|R(f)-R_{emp}(f)|-E\sup_{f\in\mathcal{F}}|R(f)-R_{emp}(f)|\right|>\varepsilon\right\}\leq 2\exp(-2n\varepsilon^2)$$

2. First symmetrization: introduce an independent copy of the training set $((\mathbf{x}'_1,y_1),\ldots,(\mathbf{x}'_n,y_n))$ and denote the corresponding empirical risk $R'_{emp}(f)$. Then

$$E\left\{\sup_{f\in\mathcal{F}}|R(f)-R_{emp}(f)|\right\}\leq E\left\{\sup_{f\in\mathcal{F}}|R_{emp}(f)-R'_{emp}(f)|\right\}$$

3. Introduce $\sigma_i\in\{-1,1\}$ i.i.d. and independent from $(\mathbf{x}_i,y_i)$ and $(\mathbf{x}'_i,y'_i)$ with $P(\sigma_i=1)=1/2$. Then

$$E\left\{\sup_{f\in\mathcal{F}}|R_{emp}(f)-R'_{emp}(f)|\right\}=\frac{1}{n}E\left\{\sup_{f\in\mathcal{F}}\left|\sum_{i=1}^{n}\sigma_i(\ell(f(\mathbf{x}_i),y_i)-\ell(f(\mathbf{x}'_i),y'_i))\right|\right\}$$

4. Condition on $(\mathbf{x}_i,y_i)$ and $(\mathbf{x}'_i,y'_i)$ and get the following bound

$$\frac{1}{n}E\left\{\sup_{f\in\mathcal{F}}\left|\sum_{i=1}^{n}\sigma_i(\ell(f(\mathbf{x}_i),y_i)-\ell(f(\mathbf{x}'_i),y'_i))\right|\right\}\leq\sqrt{\frac{2\log 2\mathcal{N}(\mathcal{F},Z_{2n})}{n}},$$

where $\mathcal{N}(\mathcal{F},Z_{2n})$ is the number of different functions in $\mathcal{F}$ on the set $Z_{2n}=(\mathbf{x}_1,\ldots,\mathbf{x}_n,\mathbf{x}'_1,\ldots,\mathbf{x}'_n)$.

5. Bound $\mathcal{N}(\mathcal{F}, Z_{2n})$ using the VC dimension h of $\mathcal{F}$. If $n \leq h$, there exists by definition a set of points Z_n such that $\mathcal{N}(\mathcal{F}, Z_n) = 2^n$, but when $n > h$, one can prove [Vapnik and Chervonenkis, 1971] that

$$\log \mathcal{N}(\mathcal{F}, Z_n) \leq h \left(\log \frac{n}{h} + 1 \right).$$

□

Note that from corollary 1.4, the convergence rate is typically in $1/\sqrt{n}$, whereas for linear regression it is in $1/n$ (see section 9). In classification, better convergence rates can be obtained when the empirical error is small, see for example [Blumer et al., 1989]:

$$\forall f \in \mathcal{F} \text{ such that } R_{emp}(f) = 0,\ R(f) \leq 4 \frac{h \left(\log \frac{2n}{h} + 1 \right) - \log \left(\frac{\eta}{4} \right)}{n}, \quad \text{with probability } 1 - \eta.$$

1.1.4 Regularization theory

The Empirical Risk Minimization is an *ill-posed* problem: a slight change in the training set can entail a large change in the estimated function. This is the reason why one has to restrict the class of function on which the minimization is performed.

In a similar way, it is possible to restrict the class of function by imposing a constraint of the regularity of the function. The regularization method has been pioneered in the 60s by Tikhonov and Arsenin [1977] and consists of introducing a measure of regularity $\Omega(f)$. In regression, $\Omega(f)$ can be for example the L_2 norm of a derivative of f or more generally a norm in a Reproducing Kernel Hilbert Space (RKHS) [Wahba, 1990].

Then instead of minimizing the empirical risk, one minimizes the *regularized* risk,

$$R_{reg}(f) = R_{emp}(f) + \lambda \Omega(f). \tag{1.5}$$

For a given λ, let us note f_λ the function minimizing (1.5). Then f_λ is also a minimizer of R_{emp} on the following set of functions:

$$\Omega_\lambda = \{f,\ \Omega(f) \leq \Omega(f_\lambda)\}.$$

Conversely, the minimizer of R_{emp} on the set of functions satisfying $\Omega(f) \leq c$ can be found by introducing a Lagrange multiplier and if λ is its optimal value, then this minimizer turns out to be the same than the one of equation (1.5) with this λ.

Consequently minimizing a regularized risk is equivalent to ERM on a restricted set of functions. More recent work on regularization theory can be found in [Girosi et al., 1995; Evgeniou et al., 2000]

1.2 Model selection

1.2.1 SRM induction principle

Using the ERM induction principle, the choice of the set of functions $\mathcal{F}$ is crucial: if it is too large, the problem of overfitting can occur, which means that the empirical risk can be very small and the corresponding true risk large. On the other hand, if $\mathcal{F}$ is too small, there is an underfitting problem: $\min_{f \in \mathcal{F}} R(f)$ can be far from $\min R(f)$.

In classification, $\mathcal{F}$ should be restricted in order to have an appropriate complexity (VC dimension). The choice of the class of function $\mathcal{F}$ is called *model selection*.

This is a difficult problem in general, but in the VC theory framework, one can try to find a class of functions which minimizes the bound of corollary (1.4). This is the idea of the *Structural Risk Minimization* (SRM) induction principle, which can be understood in figure 1.2

Consider a family of class of functions $\mathcal{F}_i$, each one being of VC dimension h_i. In the traditional SRM formulation [Vapnik and Chervonenkis, 1974; Vapnik, 1982], the class of functions are nested, $\mathcal{F}_i \subset \mathcal{F}_{i+1}$ and hence of increasing complexity, $h_i \leq h_{i+1}$. However this assumption is not needed.

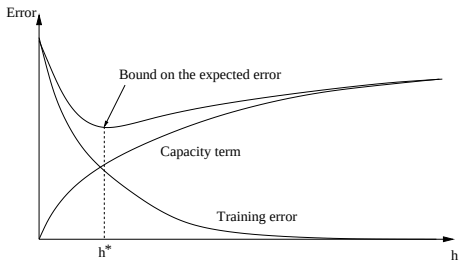


Figure 1.2: Description of the SRM principle: a sequence of class of functions of increasing complexity is constructed and the one which realizes the best trade-off between small empirical error and small complexity is selected

For a given i, let f_n^i be the function minimizing the empirical risk over $\mathcal{F}_i$. Corollary 1.4 implies that with probability $1-\eta_i$,

$$R(f_n^i) \leq R_{emp}(f_n^i) + \sqrt{\frac{\varphi(h_i) - \log(\eta_i/4)}{n}} + \frac{1}{n} \tag{1.6}$$

with $\varphi(h)$ being the complexity term,

$$\varphi(h) = h\left(\log\frac{2n}{h} + 1\right).$$

We have fixed one η_i per class of function. Let us make equation (1.6) uniform over i: with probability $1 - \sum \eta_i$,

$$\forall i, \quad R(f_n^i) \leq R_{emp}(f_n^i) + \sqrt{\frac{\varphi(h_i) - \log(\eta_i/4)}{n}} + \frac{1}{n} \tag{1.7}$$

Suppose there are p class of functions $\mathcal{F}_1, \ldots, \mathcal{F}_p$ and let us choose $\eta_i = \eta/p$. If $\hat{\imath}$ is the model chosen by the SRM principle, i.e. $\hat{\imath}$ minimizes the right hand side of (1.6), then we get from (1.7) that with probability $1-\eta$,

$$R(f_n^{\hat{\imath}}) \leq \min_{1\leq i\leq p} R_{emp}(f_n^i) + \sqrt{\frac{\varphi(h_i) + \log(p) - \log(\eta/4)}{n}} + \frac{1}{n}. \tag{1.8}$$

The additional factor coming from the fact that p models have been tested is only $\log(p)$. If p is small, this factor does not really matter. However when the number of models is "exponentially large" or even infinite, one has to put a prior weight on the model through the choice of the constants η_i. For an infinite number of models, two possible choices are $\eta_i = \eta 2^{-i}$ or $\eta_i = \eta i^{-2}/(\pi^2/6)$. The first choice leads to more bias toward the first models since the $\log(\eta_i)$ term in (1.6) is linear in the first case and logarithmic in the second.

When there is a large number of models per dimension, SRM can be implemented in two ways. To study this, let us take the example of feature selection whose implementation is discussed later in section 5.1.

Suppose that the dimensionality d of the input space is large and that a lot of features are irrelevant. Consider the set of hyperplanes: in this case, it makes sense to try to classify the training data correctly while imposing a sparsity constraint on the normal vector of the hyperplane. Only features with non-zero entries in the normal vector will be relevant in the decision function.

Let $\mathcal{F}_p$ be the set of hyperplanes passing through the origin with at most p non-zero coefficients,

$$\mathcal{F}_p = \{\mathbf{x} \mapsto \mathbf{w}\cdot\mathbf{x},\ \mathbf{w}\in\mathbb{R}^d,\ \|\mathbf{w}\|_0 \leq p\}.$$

The p non-zero coefficients of a function in $\mathcal{F}_p$ can be chosen in $N_p = \binom{d}{p}$ different ways and $\mathcal{F}_p$ can be subdivided in $\mathcal{F}_p = \bigcup_{q=1}^{N_p} \mathcal{F}_{pq}$ each different q representing a choice of the indexes of the possible non-zero coefficients of a function in $\mathcal{F}_p$.

The VC dimension of $\mathcal{F}_{pq}$ is p (hyperplanes passing through the origin have a VC dimension of p and not $p+1$).

At this point, there are two points of view:

- Consider each of the $\mathcal{F}_{pq}$ as a different model and apply directly SRM to it. Of course, for a given dimension p, there is an exponentially large number of models N_p of models of this dimension. One can choose $\eta_{pq} = \eta 2^{-p}/N_p$. Then one has to find the function which minimizes the following quantity,

$$R_{emp}(f_n^p) + \sqrt{\frac{\varphi(p) + p(1+\log(d)) - \log(\eta/4)}{n}}. \tag{1.9}$$

  Remember that $N_p = \binom{d}{p}$ and we bounded $\log(N_p) \leq p\log(d)$.

- Or one can consider $\mathcal{F}_p$ as one model. To compute the complexity term associated with the $\mathcal{F}_p$, we have to upper-bound the log of the number of dichotomies in $\mathcal{F}_p$. This latter is bounded by N_p times the number of dichotomies in one of the $\mathcal{F}_{pq}$ and finally the complexity term is $\varphi(p)+\log(N_p)$.

  Note surprisingly, this approach leads to the same minimization that the one given in equation (1.9).

The number of initial features d appears in (1.9) as a logarithmic term. As pointed out in [Ng, 1998], this is the reason why it is still possible to learn efficiently even when the number of initial features is exponentially large.

1.2.2 Overfitting in model selection

As shown in the previous section, when a lot of models are available, risk bounds become larger and not taking this fact into account can result in an overfitting phenomenon in the model selection stage. To study this problem, let us first extend the model selection setting described in the previous section to a more general one.

In general, a learning algorithm has some parameters $\boldsymbol{\alpha}$ and some hyperparameters $\boldsymbol{\theta}$ which need to be evaluated. In the case of ERM, the hyperparameters $\boldsymbol{\theta}$ represent the choice of the class of function $\mathcal{F}$, whereas the parameters $\boldsymbol{\alpha}$ is a description of the function itself in the class

Learning is a two stages procedure:

1. For a fixed value of $\boldsymbol{\theta}$, find the best parameters $\boldsymbol{\alpha}^0$

$$\boldsymbol{\alpha}^0(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\alpha}} T(\boldsymbol{\alpha}, \boldsymbol{\theta}).$$

2. Find the best value of $\boldsymbol{\theta}$,

$$\boldsymbol{\theta}^0 = \arg\min_{\boldsymbol{\theta}} V(\boldsymbol{\alpha}^0(\boldsymbol{\theta}), \boldsymbol{\theta}).$$

The first stage is the classical risk minimization: the model (described by $\boldsymbol{\theta}$) is fixed and ERM is the case where T is the empirical error. The model selection is the second level (choice of $\boldsymbol{\theta}$). For SRM, V is the upper bound given in corollary 1.4.

Note that naive model selection might be time consuming since for each value of $\boldsymbol{\theta}$ being tested, a minimization over $\boldsymbol{\alpha}$ is required.

Usually, the criterion V is an estimate or an upper-bound of the generalization error. For this reason, it is often algorithm dependent. However, a very popular criterion is the validation error. This criterion does not depend on the learning algorithm and is very easy to compute. In the rest of this section, we will consider this criterion for simplicity.

Suppose that several models $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_p$ are tested. For each model $\boldsymbol{\theta}_i$, the learning algorithm is applied and outputs a function f_i. Let $\mathcal{F}^*$ be the set of functions $\{f_1, \ldots, f_p\}$. The model selection step consists then in choosing the best function in $\mathcal{F}^*$ through to the model selection criterion V.

Suppose that V is a validation error:

$$R_{val}(f) = \frac{1}{n'} \sum_{i=1}^{n'} \ell(f(\mathbf{x}'_i), y'_i),$$

where $\{(\mathbf{x}'_i, y_i)\}_{1 \leq i \leq n'}$ is an independent sample drawn from the same distribution as the training set.

Since for all $f \in \mathcal{F}^*$, $R_{val}(f)$ is an unbiased estimator of the true risk $R(f)$, similarly to ERM, the standard way of performing model selection is to choose the function f_i which minimizes R_{val}.

The empirical risk R_{emp} is not an unbiased estimator of the true risk since the functions in $\mathcal{F}^*$ have been chosen using the training examples. That is the reason why "unseen" examples are needed in order to have an unbiased estimator of the risk.

Let f^* be the minimizer of the validation error,

$$f^* = \arg \min_{f \in \mathcal{F}^*} R_{val}(f).$$

As for ERM, $R_{val}(f^*)$ is not an unbiased estimate of $R(f^*)$. To upper-bound $R(f^*)$, we need a uniform convergence argument. For this purpose, let us derive an upper-bound on

$$P\left\{ \sup_{f \in \mathcal{F}^*} |R(f) - R_{val}(f)| > \varepsilon \right\}.$$

From Hoeffding's inequality [Hoeffding, 1963], we know that

$$\forall f \in \mathcal{F}^*, P\left\{|R(f) - R_{val}(f)| > \varepsilon\right\} < 2\exp(-2n'\varepsilon^2)$$

Since the cardinal of $\mathcal{F}^*$ is p, the union bound leads to

$$P\left\{ \sup_{f \in \mathcal{F}^*} |R(f) - R_{val}(f)| > \varepsilon \right\} \leq 2p\exp(-2n'\varepsilon^2).$$

From this, we can deduce that with confidence $1 - \eta$,

$$R(f^*) \leq R_{val}(f) + \sqrt{\frac{\log(p) - \log(\eta/2)}{2n'}}. \tag{1.10}$$

As long as p is not too large, $R_{val}(f)$ is a good estimate of $R(f)$ and minimizing the validation error is meaningful. However, if p is large (i.e. $\log(p)$ is of the order of n'), then an overfitting problem might occur. This is directly comparable to a class of function which is too large when performing empirical risk minimization.

The two stages are actually equivalent: the model selection stage consists in performing an empirical minimization using a validation set on the set $\mathcal{F}^*$. Thus if $\mathcal{F}^*$ is too large (too many models), overfitting occurs during this stage [Ng, 1997].

Optimizing the hyperparameters

The problem of overfitting in model selection is not mentioned often in the literature since in practice the number of models tested is usually small. Indeed, optimizing the parameters of a given model is usually time consuming and it can be computationally prohibitive to try a lot of different models. However, as discussed in [Bengio, 2000], it is possible to optimize the hyperparameters by gradient descent. In this case, as for ERM, the value of p in equation (1.10) should not be the number of model tested (e.g. the number of gradient steps), but the number of different possible values of the hyperparameters and in this case p can be very large.

Discrete hyperparameters

If one tries to optimize on the validation set m hyperparameters, $(\beta_1, \ldots, \beta_m)$, and each of them can take q different values then the number of function in $\mathcal{F}^*$ is $p = q^m$ and equation (1.10) indicates that the additional estimation error from the model selection stage is of the order $\sqrt{m/n'}$

$$R(f^*) \leq R_{val}(f) + \sqrt{\frac{m \ln(q) - \ln(\eta/2)}{2n'}}. \tag{1.11}$$

Continuous hyperparameters

If the hyperparameters can take continuous values, it is more difficult to have an upper bound of $R(f^*)$, but it is possible to do so using covering numbers and traditional VC bounds.

Since model selection is just performing ERM on $\mathcal{F}^*$ using a validation set, the bound given in corollary 1.4 is still valid, provided that $R_{emp}(f)$ is replaced by $R_{val}(f)$ and h refers to VC dimension of $\mathcal{F}^*$.

However the VC dimension of $\mathcal{F}^*$ is in general impossible to compute since the functions in this set are themselves solutions of an optimization problem. Intuitively, we would like to replace the VC dimension of $\mathcal{F}^*$ by the number of hyperparameters m, as in the discrete case. This is possible to do it under the assumption that the functions in $\mathcal{F}^*$ depend "smoothly" on the model parameters $\boldsymbol{\theta}$. We just give here a sketch of the proof.

Let Θ be the set of possible value of the model parameters. For each of them, let us note $f_{\boldsymbol{\theta}} = f_{\boldsymbol{\alpha}^0(\boldsymbol{\theta})}$ the optimal function in the model $\boldsymbol{\theta}$, i.e. minimizing the training criterion T. We have

$$\mathcal{F}^* = \{f_{\boldsymbol{\theta}},\ \boldsymbol{\theta} \in \Theta\}.$$

Let us assume that $f_{\boldsymbol{\theta}}$ does not change too much with $\boldsymbol{\theta}$, more precisely that the following Lipschitz condition holds:

$$\forall (\boldsymbol{\theta}, \boldsymbol{\theta}') \in \Theta^2, \quad \|f_{\boldsymbol{\theta}} - f_{\boldsymbol{\theta}'}\|_\infty \leq C \left\|\boldsymbol{\theta} - \boldsymbol{\theta}'\right\|_\infty .$$

For such an assumption to make sense, we have to consider a class of real-valued functions; the associated binary functions are thresholded versions of the real ones. The covering number of $\mathcal{F}^*$ can be bounded by a constant (dependent on C) times the covering number of Θ [van der Vaart and Wellner, 1996]. Under the additional assumption that Θ is compact subset of $\mathbb{R}^m$ (which means that the hyperparameters $\boldsymbol{\theta}$ take their values in a finite range), then the logarithm covering number of Θ is bounded in $O(m)$. Finally, standard results [Shawe-Taylor et al., 1998] about classification using real-valued functions give generalization error bounds in terms of covering numbers and a margin error.

Summary

We have pointed how the overfitting danger when there are too many hyperparameters (models). The additional estimation error grows as the square root of the number of hyperparameters. One could consider to do a model selection on the hyperparameters: first try a small number of hyperparameters and then increase it.

Note that in this section, we discussed the problem of overfitting in model selection using a validation error criterion, but the same occurs with any criterion. The example of feature selection with the SRM method in section 1.2.1 showed that when the number of feature selected is high, the bound on the expected risk gets larger (equation 1.9). This can also be seen from equation (1.11) if m is the number of features.

1.3 Vicinal Risk Minimization

1.3.1 A new induction principle

Empirical Risk Minimization (ERM) is equivalent to minimizing the expectation of the loss function with respect to an empirical distribution $P_{emp}(\mathbf{x}, y)$ formed by assembling delta functions located on each

example:

$$dP_{emp}(\mathbf{x}, y) = \frac{1}{n} \sum_{i=1}^{n} \delta_{\mathbf{x}_i}(\mathbf{x}) \delta_{y_i}(y) \tag{1.12}$$

It is quite natural to consider improved density estimates by replacing the delta functions $\delta_{\mathbf{x}_i}(\mathbf{x})$ by some estimate of the density in the *vicinity* of the point $\mathbf{x}_i$, $P_{\mathbf{x}_i}(\mathbf{x})$.

$$dP_{est}(\mathbf{x}, y) = \frac{1}{n} \sum_{i=1}^{n} dP_{\mathbf{x}_i}(\mathbf{x}) \delta_{y_i}(y) \tag{1.13}$$

We can define in this way the *vicinal risk* of a function as:

$$R_{vic}(f) = \int \ell(f(\mathbf{x}), y) \; dP_{est}(\mathbf{x}, y) = \frac{1}{n} \sum_{i=1}^{n} \int \ell(f(\mathbf{x}), y_i) dP_{\mathbf{x}_i}(\mathbf{x}) \tag{1.14}$$

The Vicinal Risk Minimization principle consists of estimating $\arg\min_{f \in \mathcal{F}} R(f)$ by the function which minimizes the vicinal risk (1.14). In general, one can construct the VRM functional using any estimate $dP_{est}(\mathbf{x}, y)$ of the density $dP(\mathbf{x}, y)$, instead of restricting our choices to pointwise kernel estimates.

Spherical gaussian kernel functions $\mathcal{N}_\sigma(\mathbf{x} - \mathbf{x}_i)$ are otherwise an obvious choice for the local density estimate $dP_{\mathbf{x}_i}(x)$. The corresponding density estimate dP_{est} is a Parzen windows estimate. The parameter σ controls the scale of the density estimate. The extreme case $\sigma = 0$ leads to the estimation of the density by delta functions and therefore leads to ERM. This must be distinguished from the case $\sigma \to 0$ because the limit is taken *after* the minimization of the integral, leading to different results as shown in the next section.

The theoretical analysis of ERM [Vapnik, 1998] shows that the crucial factor is the capacity of the class $\mathcal{F}$ of functions. Large classes entail the risk of overfitting, whereas small classes entail the risk of underfitting. Two factors however are responsible for generalization of VRM, namely the quality of the estimate dP_{est} and the size of the class $\mathcal{F}$ of functions. If dP_{est} is a poor approximation to P then VRM can still perform well if $\mathcal{F}$ has suitably small capacity. ERM indeed uses a very naive estimate of dP and yet can provide good results. On the other hand, if $\mathcal{F}$ is not chosen with suitably small capacity then VRM can still perform well if the estimate dP_{est} is a good approximation to dP. One can even take the set of all possible functions (whose capacity is obviously infinite) and still find a good solution if the estimate dP_{est} is close enough to dP with an adequate metric. For example, if dP_{est} is a Parzen window density estimate, then the Vicinal Risk minimizer is the Parzen window classifier. This latter property contrasts nicely with the ERM principle whose results strongly depend on the choice of the class of functions.

1.3.2 Relation to other algorithms

The formulation of VRM presented here has first been introduced in [Chapelle et al., 2000]. It is based on an idea of Vapnik [2000] that he also calls Vicinal Risk Minimization, but the two formulations lead to rather different algorithms. He used this principle to compute new kernels for SVMs which take into account the *vicinity* of a point,

$$\tilde{K}(\mathbf{x}, \mathbf{y}) = \int K(\mathbf{u}, \mathbf{v}) dP_{\mathbf{x}}(\mathbf{u}) dP_{\mathbf{y}}(\mathbf{v}).$$

It turns out that even if the names are identical, both methods are quite different and should not be confused.

We now discuss the relationship of VRM to existing methods.

1.3.3 Non parametric algorithms

When the set of functions $\mathcal{F}$ is unconstrained, the function f minimizing the vicinal risk (1.14) is

in regression, $f(\mathbf{x}) = \frac{\sum_{i=1}^n y_i P_{\mathbf{x}_i}(\mathbf{x})}{\sum_{i=1}^n P_{\mathbf{x}_i}(\mathbf{x})}$, which is the Nadaraya [1964] - Watson [1964] estimator

in classification, $f(\mathbf{x}) = \text{sgn}(\sum_{i=1}^n y_i P_{\mathbf{x}_i}(\mathbf{x}))$, often called Parzen [1962] classifier [Devroye et al., 1996, chapter 10]. When the bandwidth σ goes to 0, this turn out to be the Nearest Neighbor rule.

Noise injection

First note that VRM is obviously related to noise injection which consists of adding perturbed examples, the perturbation being usually a Gaussian noise,

$$\forall i \leq n,\ \forall r \leq n_r,\ \mathbf{x}_i^r = \mathbf{x}_i + \mathcal{N}_\sigma,\ \ y_i^r = y_i.$$

When the number of replicates for each training point n_r goes to infinity, VRM and noise injection are equivalent. Noise injection has first been applied successfully to neural network training [Sietsma and Dow, 1991]. Additional references and theoretical analysis can be found in [Grandvalet et al., 1997].

VRM Regression and Ridge Regression

Consider the case of VRM for regression with spherical Parzen windows (using gaussian kernel) with standard deviation σ and with a family $\mathcal{F}$ of linear functions $f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{w}\cdot\mathbf{x}+b$. We can write the vicinal risk as:

$$\begin{aligned}
R_{vic}(f) &= \frac{1}{n}\sum_{i=1}^n \int (f(\mathbf{x}) - y_i)^2 dP_{\mathbf{x}_i}(\mathbf{x}) \\
&= \frac{1}{n}\sum_{i=1}^n \int (f(\mathbf{x}_i + \boldsymbol{\varepsilon}) - y_i)^2 d\mathcal{N}_\sigma(\boldsymbol{\varepsilon}) \\
&= \frac{1}{n}\sum_{i=1}^n \int (f(\mathbf{x}_i) - y_i + \mathbf{w}\cdot\boldsymbol{\varepsilon})^2\ d\mathcal{N}_\sigma(\boldsymbol{\varepsilon}) \\
&= \frac{1}{n}\sum_{i=1}^n\ (f(\mathbf{x}_i) - y_i)^2\ \ +\ \ \sigma^2||\mathbf{w}||^2
\end{aligned}$$

The resulting expression is the empirical risk augmented by a regularization term. The particular cost function above is known as the Ridge Regression cost function [Hoerl and Kennard, 1970].

This result can be extended to the case of non linear functions f by performing a Taylor expansion of $f(\mathbf{x}_i + \boldsymbol{\varepsilon})$. The corresponding regularization term then combines successive derivatives of function f. Details on the link between regularization and noise injection can be found in [Leen, 1995; Bishop, 1995; An, 1996; Reed et al., 1995].

VRM Classifier and Constrained Logistic Classifier

Consider the case of VRM for classification with spherical Parzen windows with standard deviation σ and with a family $\mathcal{F}$ of linear functions $f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{w}\cdot\mathbf{x} + b$. We can assume without loss of generality that $||\mathbf{w}|| = 1$. We can write the vicinal risk as:

$$\begin{aligned}
R_{vic}(\mathbf{w}, b) &= \frac{1}{n}\sum_{i=1}^n \int \mathrm{I}_{\text{sgn}(\mathbf{w}\cdot\mathbf{x}+b) \neq y_i}\ dP_{\mathbf{x}_i}(\mathbf{x}) \\
&= \frac{1}{n}\sum_{i=1}^n \int \mathrm{I}_{\text{sgn}(\mathbf{w}\cdot\mathbf{x}_i + \mathbf{w}\cdot\boldsymbol{\varepsilon}+b) \neq y_i}\ d\mathcal{N}_\sigma(\boldsymbol{\varepsilon})
\end{aligned}$$

We can decompose $\boldsymbol{\varepsilon} = \mathbf{w}\varepsilon_w + \boldsymbol{\varepsilon}_\perp$ where $\mathbf{w}\varepsilon_w$ represents its component parallel to $\mathbf{w}$ and $\boldsymbol{\varepsilon}_\perp$ represents its orthogonal component. Since $||\mathbf{w}|| = 1$, we have $\mathbf{w}\cdot\boldsymbol{\varepsilon} = \varepsilon_w$. After integrating over $\boldsymbol{\varepsilon}_\perp$ we are left with

the following expression:

$$\begin{aligned} R_{vic}(\mathbf{w}, b) &= \frac{1}{n}\sum_{i=1}^{n} \int \mathrm{I}_{\mathrm{sgn}(\mathbf{w}\cdot\mathbf{x}_i + b + \varepsilon_w) \neq y_i} \; d\mathcal{N}_\sigma(\varepsilon_w) \\ &= \frac{1}{n}\sum_{i=1}^{n} \varphi\left(-\frac{y_i(\mathbf{w}\cdot\mathbf{x}_i + b)}{\sigma}\right), \end{aligned} \tag{1.15}$$

where φ is the integral of a gaussian, $\varphi(x) = \frac{1+\mathrm{erf}(x/\sqrt{2})}{2}$.

By rescaling $\mathbf{w}$ and b by a factor $1/\sigma$, we can write the following equivalent formulation of the VRM:

$$\begin{cases} \underset{\mathbf{w},b}{\arg\min} \;\; \frac{1}{n}\sum_{i=1}^{n} \; \varphi(-y_i(\mathbf{w}\cdot\mathbf{x}_i + b)) \\ \quad \text{with constraint} \;\; ||\mathbf{w}|| = 1/\sigma \end{cases} \tag{1.16}$$

Except for the minor shape difference between sigmoid functions, the above formulation describes a Logistic Classifier with a constraint on the weights. This formulation is also very close to using a single artificial neuron with a sigmoid transfer function and a weight decay.

The above proof illustrates a general identity. Transforming the empirical probability estimate (1.12) by convolving it with a kernel function is equivalent to transforming the loss function $\ell(f(x), y)$ by convolving it with the same kernel function. This is summarized in the following equality, where $\star$ represents the convolution operator.

$$\int \ell(f(x), y) \;\; [\mathcal{N}_\sigma(\cdot) \star dP_{emp}(\cdot, y)] \, (\mathbf{x}) \;\; = \;\; \int [\ell(f(\cdot), y) \star \mathcal{N}_\sigma(\cdot)] \, (\mathbf{x}) \; dP_{emp}(\mathbf{x}, y)$$

1.3.4 Theoretical considerations

The VRM induction principle should not be considered as an algorithm, but as a way to derive new algorithms based on this principle as we will see in the next chapter.

Because there are a lot of different ways to estimate the density in the vicinity of a training point, it is not possible to give general bounds for this induction principle. For this reason, we will only consider the special case of spherical Gaussian estimate.

First note that in the asymptotic case, the uniform convergence of the empirical risk to the the true risk can be studied in the *empirical process* theory [van der Vaart and Wellner, 1996]. Recently, there has been several studies about the convergence of *smoothed* empirical processes [Radulovic and Wegkamp, 2000; Gaenssler and Rost, 2000], where "smoothed" means that the empirical measure is replaced by its convolution with a function such as a Gaussian. Thus, it seems it might be possible to analyze the consistency of the VRM induction possible with these tools. For instance, in [Radulovic and Wegkamp, 2000], it was shown that under smoothness conditions of the underlying probability distribution, there exists certain class of functions for which the smoothed empirical process converges whereas the standard one does not.

This kind of result is not surprising from a statistical learning theory point of view: for instance, when the class of function is not restrained, ERM is not consistent, whereas VRM turns out to be the Parzen window classifier which has been shown to be consistent [Devroye et al., 1996] under the condition that the width σ satisfies $\sigma(n) \to 0$ and $\sigma(n)n^d \to \infty$ as n goes to infinity.

It might thus possible to adapt the proof techniques used to show the convergence of smoothed empirical process in order to get non asymptotic bounds for the VRM principle. Note that the convergence of the smoothed empirical process is a sufficient condition for VRM to be consistent, but it is not necessary.

Also, unlike for ERM, the uniform convergence framework is not suitable to analyze VRM. Consider the following simple 1D example: $P(x)$ uniform on $[-1, 1]$ and $P(y|x) = 1$ iff $xy > 0$. Let us choose the width of the Parzen window estimate used in VRM as the following function of the number of training examples, $\sigma(n) = 1/n^3$. Consider the case where the class of function is not restrained. Then, it is easy

to see that the VRM converges to the 1-Nearest Neighbor solution, which itself converges to the optimal one. On the other hand, for any training set $(x_1, \ldots, x_n)$, let

$$A^+ = \{x \in [0,1], \quad \forall i, \ |x - x_i| > 1/n^2\},$$

$$A^- = \{x \in [-1,0], \quad \forall i, \ |x - x_i| > 1/n^2\}.$$

Now let us define the function f_n as

$$f_n(x) = \begin{cases} 1 & \text{if} \quad x \in [0,1] \backslash A^+ \quad \text{or} \quad x \in A^- \\ -1 & \text{if} \quad x \in [-1,0] \backslash A^- \quad \text{or} \quad x \in A^+ \end{cases}.$$

We have $R(f_n) = (A^+ \cup A^-) \geq 1 - 2/n \rightarrow 1$ and

$$R_{vic}(f_n) \leq \sqrt{2/\pi} \int_{1/n^2}^{\infty} \exp(-t^2/2\sigma(n)^2) dt \rightarrow 0.$$

As a consequence, on this example, we have almost surely,

$$\lim_{n \rightarrow \infty} \sup_f |R(f) - R_{vic} f| = 1.$$

Based on this example, the intuitive difference between VRM and ERM is the following: when the class of functions is very rich, there is usually an infinite number of hypothesis with zero training error and choosing one of them at random is likely to give bad results. On the other hand, if your consider VRM with a bandwidth going to 0 rapidly, a lot of hypothesis will have a vicinal risk almost equal to 0, but the "maximum margin" one minimizes the vicinal risk and there is no tie.

From the previous analysis, it seems that the standard tools of statistical learning theory can not be applied to analyze VRM. Nevertheless, using a standard derivation (see for example page 10 or Bartlett and Mendelson 2001), we can give a bound for the linear case.

From equation (1.16), let us upper-bound

$$\sup_{\|\mathbf{w}\|=1/\sigma, \ b} \left| \underbrace{\frac{1}{n} \sum_{i=1}^{n} \varphi(-y_i(\mathbf{w} \cdot \mathbf{x}_i + b))}_{R_{vic}(f)} - \underbrace{\int \varphi(-y(\mathbf{w} \cdot \mathbf{x} + b)) dP(\mathbf{x}, y)}_{\geq R(f)/2} \right|.$$

Note that the right part of this quantity is larger than $R(f)/2$ because $\varphi(t) \leq \frac{1}{2} \mathrm{I}_{t \geq 0}$.

To do this, we apply the first three steps of page 10. Then we have to bound

$$\frac{2}{n} \ E \sup_{\|\mathbf{w}\|=1/\sigma, \ b} \left| \sum_{i=1}^{n} \varepsilon_i \varphi(-y_i(\mathbf{w} \cdot \mathbf{x}_i + b)) \right|, \tag{1.17}$$

where ε_i are iid variables with $P(\varepsilon_i = 1) = P(\varepsilon_i = -1) = 1/2$.

For this purpose, we need the following lemma [Ledoux and Talagrand, 1991, Theorem 4.12]

Lemma 1.5 (Contraction principle) *Let $F : \mathbb{R} \mapsto \mathbb{R}^+$ a convex and increasing function. Let φ_i be 1-Lipschitz functions with $\varphi_i(0) = 0$. Then for any bounded subset T of $\mathbb{R}^n$,*

$$E \ F \left(\frac{1}{2} \sup_T \left| \sum_{i=1}^{n} \varepsilon_i \varphi_i(t_i) \right| \right) \leq E \ F \left(\sup_T \left| \sum_{i=1}^{n} \varphi_i(t_i) \right| \right).$$

Using this lemma with $\varphi_i = \sqrt{2\pi}(\varphi - 1/2)$ and $F(t) = t$, equation (1.17) is bounded by

$$\frac{4}{\sqrt{2\pi n}} E \sup_{\|\mathbf{w}\|=1/\sigma,\ b} \left| \sum_{i=1}^{n} \varepsilon_i y_i (\mathbf{w} \cdot \mathbf{x}_i + b) \right| + 2 \left| \sum_{i=1}^{n} \varepsilon_i / 2 \right|.$$

Moreover,

$$E \sup_{\|\mathbf{w}\|=1/\sigma} \left| \mathbf{w} \cdot \sum_{i=1}^{n} \varepsilon_i y_i \mathbf{x}_i \right| = \frac{1}{\sigma} E \left\| \sum_{i=1}^{n} \varepsilon_i y_i \mathbf{x}_i \right\| \leq \frac{1}{\sigma} \sqrt{E \left(\sum_{i=1}^{n} \varepsilon_i y_i \mathbf{x}_i \right)^2} \leq \frac{1}{\sigma} \sqrt{n E \left\| \mathbf{x}_i^2 \right\|}.$$

Combining these equations and making the assumption that $|b| \leq B$, we get the following bound on (1.17),

$$\frac{4}{\sqrt{2\pi n}} \left(\frac{\sqrt{E \left\| \mathbf{x}_i^2 \right\|}}{\sigma} + B + 2 \right). \tag{1.18}$$

Now let assume that the points $\mathbf{x}_i$ satisfy $\|\mathbf{x}_i\| \leq R$. In order for the $\mathbf{w} \cdot \mathbf{x}_i + b$ to take different signs, it is necessary to have $|b| \leq R/\sigma$ and (1.18) is bounded by

$$\frac{8}{\sqrt{2\pi n}} \left(\frac{R}{\sigma} + 1 \right).$$

Finally, with probability $1 - \eta$, the linear function minimizing the vicinal risk satisfies,

$$R(f) \leq 2 R_{vic}(f) + \frac{1}{\sqrt{n}} \left(7 \frac{R}{\sigma} + 7 + \log(2/\eta) \right).$$

Chapter 2

Algorithms derived from VRM

2.1 Support Vector Machines

2.1.1 Motivation

Consider the case of VRM for classification with spherical Parzen windows with standard deviation σ and with a family $\mathcal{F}$ of linear functions $f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$.

Suppose the training data is linearly separable, which means that there exists $(\mathbf{w}, b)$ such $\forall i,\ y_i(\mathbf{w} \cdot \mathbf{x}_i + b) > 0$

When $\sigma \to 0$, the vicinal risk (1.15) is dominated by the terms corresponding to the examples whose distance to the decision boundary is minimal. Indeed,

$$1 - |\mathrm{erf}(x)| \underset{x \to \pm\infty}{\sim} \frac{\exp(-x^2)}{\sqrt{\pi}x} \quad \text{and} \quad \varphi(-x) = \frac{1 + \mathrm{erf}(-x/\sqrt{2})}{2} \underset{x \to +\infty}{\sim} \frac{\exp(-x^2/2)}{\sqrt{2\pi}x}.$$

Let $v_i = y_i(\mathbf{w} \cdot \mathbf{x}_i + b) > 0$ (with an hyperplane $(\mathbf{w}, b)$ separating the training examples) and $v_{min} = \min v_i$, the distance of the closest point to the hyperplane.

It turns out that

$$R_{vic}(\mathbf{w}, b) = \frac{1}{n} \sum_{i=1}^{n} \varphi(-v_i/\sigma) \underset{\sigma \to 0}{\sim} \#\{i,\ v_i = v_{min}\} \frac{\exp(-v_{min}^2/2\sigma^2)}{\sqrt{2\pi} v_{min}/\sigma}.$$

From the previous equation, we see that when σ goes to zero, the vicinal is minimized when v_{min} is maximized.

That is the motivation of the *Optimal Separating Hyperplane* (see figure 2.1) solution:

> *Find the separating hyperplane such that the distance of the closest point to it (margin) is maximum.*

2.1.2 Optimal separating hyperplane

As explained above, we would like to find the Optimal Separating Hyperplane (OSH) defined as:

$$(\mathbf{w}_0, b_0) = \arg\max_{(\mathbf{w},b)} \left\{ \min_i y_i(\mathbf{w} \cdot \mathbf{x}_i + b),\ ||\mathbf{w}|| = 1 \right\}.$$

By rescaling $(\mathbf{w}, b)$ such that $\min_i y_i(\mathbf{w} \cdot \mathbf{x}_i + b) = 1$, the OSH is also the solution of the following optimization problem

$$\min\ \mathbf{w}^2 \tag{2.1}$$

under constraints

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \tag{2.2}$$

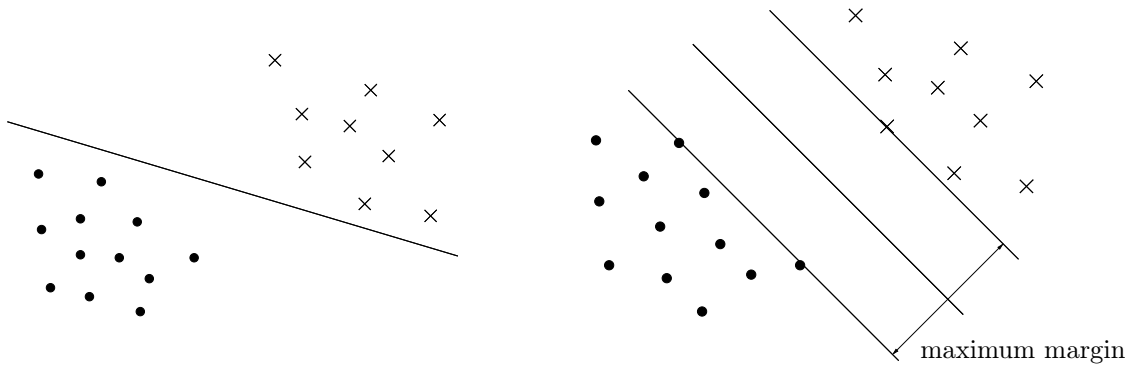


Figure 2.1: *Both hyperplanes separate correctly the training examples. But the Optimal Separating Hyperplane on the right hand side has a larger margin and hence a lower vicinal risk. Intuitively, this latter is less sensitive to the noise in the training set.*

Since $\mathbf{w}^2$ is convex, minimizing equation (2.1) under linear constraints (2.2) can be achieved by the use of Lagrange multipliers. Let us denote the n non negative Lagrange multipliers associated with constraints (2.2) by $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$. Minimizing (2.1) amounts to find the saddle point of the Lagrange function:

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^2 - \sum_{i=1}^{n} \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1]. \tag{2.3}$$

To find this point, one has to minimize this function over $\mathbf{w}$ and b and to maximize it over the Lagrange multipliers $\alpha_i \geq 0$. The saddle point must satisfy the following conditions :

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial b} = \sum_{i=1}^{n} y_i \alpha_i = 0 \tag{2.4}$$

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = 0 \tag{2.5}$$

Substituting equations (2.4) and (2.5) into (2.3), our optimization problem amounts to maximize

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{2.6}$$

with $\alpha_i \geq 0$ and under constraint (2.4). This can be achieved by the use of standard quadratic programming methods [Bazaraa and Shetty, 1979].

Once the vector $\boldsymbol{\alpha}^0 = (\alpha_1^0, \ldots, \alpha_n^0)$ solution of the maximization problem (2.6) has been found, taking (2.5) into account, the OSH $(\mathbf{w}_0, b_0)$ has the following expansion

$$\mathbf{w}_0 = \sum_{i=1}^{n} \alpha_i^0 y_i \mathbf{x}_i \tag{2.7}$$

while b_0 can be determined from the Kuhn-Tucker conditions

$$\alpha_i^0 [y_i(\mathbf{w}_0 \cdot \mathbf{x}_i + b_0) - 1] = 0 \tag{2.8}$$

Note that from equation (2.8), the points for which $\alpha_i^0 > 0$ satisfy (2.2) with equality. Geometrically, it means that they are the closest points to the optimal hyperplane (see figure 2.1). These points play a crucial role, since they are the only points needed in the expression of the OSH (see equation (2.7)). They are called *support vectors* to point out the fact that they "support" the expansion of $\mathbf{w}_0$.

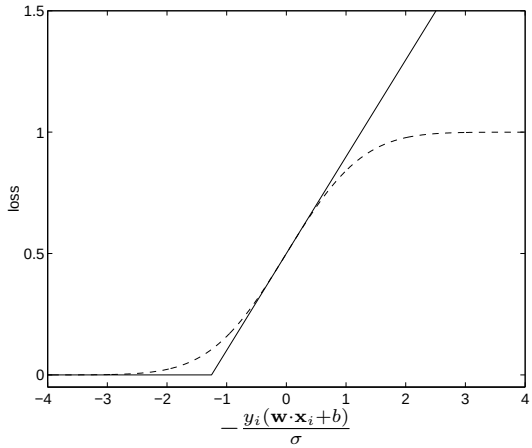


Figure 2.2: The VRM loss function and its convex approximation

Given a support vector $\mathbf{x}_i$, the parameter b_0 can be obtained from the corresponding Kuhn-Tucker condition as

$$b_0 = y_i - \mathbf{w}_0 \cdot \mathbf{x}_i$$

The problem of classifying a new point $\mathbf{x}$ is solved by by looking at the sign of

$$\mathbf{w}_0 \cdot \mathbf{x} + b_0$$

Considering the expansion (2.7) of $\mathbf{w}_0$, the hyperplane decision function can thus be written as

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{n} \alpha_i^0 y_i \mathbf{x}_i \cdot \mathbf{x} + b\right) \tag{2.9}$$

We motivated the maximization of the margin through the VRM induction principle, but note that the standard way of justifying the Optimal Separating Hyperplane is through a VC dimension argument. Indeed, the VC dimension of hyperplanes which separate the training points which a margin M is bounded by R^2/M^2, where R is the radius of the smallest sphere enclosing the training points [Vapnik, 1995]. Generalization bounds depending on the margin can be found in [Shawe-Taylor et al., 1998; Bartlett and Shawe-Taylor, 1999]

2.1.3 Linearly non-separable case

If the data is not linearly separable, the problem of finding the OSH becomes meaningless. Remember that the OSH has been motivated by the minimization of the vicinal risk (1.15) when the bandwidth σ goes to zero. When σ does not go to 0, the corresponding loss function is a sigmoid which unfortunately is not convex. The piecewise linear loss function of figure 2.2 is a convex approximation of the vicinal loss: at the origin it has the same slope as the loss φ and the junction point is $-\sqrt{\pi/2}$.

The new optimization problem corresponding to this piecewise linear loss function is

$$\min \sum_{i=1}^{n} \xi_i \tag{2.10}$$

under constraints

$$\xi_i \geq 0$$

$$\frac{y_i(\mathbf{w} \cdot \mathbf{x}_i + b)}{\sigma} \geq \sqrt{\frac{\pi}{2}} - \xi_i$$

$$\mathbf{w}^2 = 1.$$

Setting $A^2 = \frac{2}{\pi\sigma^2}$, this equivalent to minimizing (2.10) under constraints

$$y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i \tag{2.11}$$
$$\mathbf{w}^2 \leq A^2.$$

This problem can be solved by introducing Lagrange multipliers [Vapnik, 1998],

$$L(\mathbf{w}, b, \boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma) = \sum_{i=1}^{n}\xi_i - \frac{1}{2}\gamma(A^2 - \mathbf{w}^2) - \sum_{i=1}^{n}\alpha_i[y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i] - \sum_{i=1}^{n}\beta_i\xi_i$$

and the dual is

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i - A\sqrt{\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i\cdot\mathbf{x}_j},$$

which has to be maximized under constraints

$$\sum_{i=1}^{n} y_i\alpha_i = 0,$$
$$0 \leq \alpha_i \leq 1.$$

However this last optimization problem is not quadratic and more difficult to solve. For this reason, Cortes and Vapnik [Cortes and Vapnik, 1995] introduced the following formulation of the generalized OSH:

Minimize

$$\frac{1}{2}\mathbf{w}^2 + C\sum_{i=1}^{n}\xi_i \tag{2.12}$$

subject to constraints (2.11) and $\xi_i \geq 0$. The first term is minimized to control the margin as in the separable case; the purpose of the second term is to keep under control the number of misclassified points. The parameter C is chosen by the user, a larger C corresponding to assigning a higher penalty to errors.

In analogy with what was done for the separable case, the use of the Lagrange multipliers leads do the following optimization problem :

Maximize

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i\cdot\mathbf{x}_j$$

subject to :

$$\sum_{i=1}^{n}\alpha_i y_i = 0 \text{ and } 0 \leq \alpha_i \leq C.$$

The only difference from the separable case is that now the α_i have an upper bound of C.

Note that the two approaches are equivalent. Indeed the constraint (2.11) is identical in both cases and in the first one, one minimizes $\sum\xi_i$ under constraints $\mathbf{w}^2 \leq A^2$, and in the second one $\sum\xi_i + \lambda\mathbf{w}^2$ (setting $\lambda = 1/2C$). The first approach is a minimization of an empirical loss on a restrained set of function and the second one is a minimization of a regularized loss and both are equivalent (see section 1.1.4): for each value of C, there exists a value of A (and thus σ) such that both problems have the same solution, and vice-versa.

We have shown that the soft margin SVM algorithm can be seen as a convex approximation of VRM with spherical gaussian estimate. The associated width σ is directly related to the soft margin parameter C. For example, in the hard margin case, $\sigma \to 0$ and $C \to \infty$.

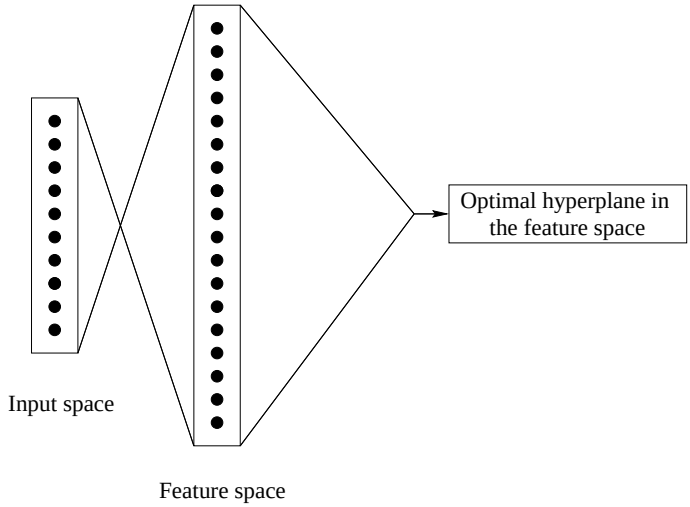


Figure 2.3: *The SV machine maps the input space into a high-dimensional feature space and then constructs an optimal hyperplane in the feature space*

2.1.4 Nonlinear Support Vector Machines

The idea of Support Vector Machines is to map the input data into a high dimensional *feature space* through some non linear mapping chosen a priori [Boser et al., 1992]. In that space the OSH constructed (see figure 2.3).

Example : To construct a decision surface corresponding to a polynomial of degree 2, the following feature space of $\frac{d(d+3)}{2}$ dimensions can be created :

$$z_i = x_i,\ 1 \leq i \leq d$$

$$z_{d+i} = (x_i)^2,\ 1 \leq i \leq d$$

$$z_{2d+1} = x_1 x_2,\ \ \ldots,\ z_N = x_d x_{d-1}$$

where $\mathbf{x} = (x_1, \ldots, x_d)$ is the input vector and $\mathbf{z} = (z_1, \ldots, z_N) = \Phi(\mathbf{x})$ is the image of $\mathbf{x}$ through the mapping Φ. The separating hyperplane constructed in the feature space is a second-degree polynomial in the input space.

One computational problem arises : the dimension of the feature space can be very high and how to construct a separating hyperplane in this high dimensional space ?

The answer to this problem comes from the fact that to construct the optimal separating hyperplane in the feature space, the mapping $\mathbf{z} = \Phi(\mathbf{x})$ does not need to be made explicit. Indeed, if we replace $\mathbf{x}$ by $\Phi(\mathbf{x})$, equation (2.6) becomes :

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

Then the training algorithm would only depend on the data through dot products in the feature space, i.e. on functions of the form $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. Now suppose we have a symmetric function K such that $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. Then only K is needed in the training algorithm and the mapping Φ is never explicitly used.

Given a mapping Φ, the kernel K is obviously $K(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$, but conversely, given a kernel K, which are the conditions for an implicit mapping to exist ?

The answer is given by Mercer's conditions [Vapnik, 1995] :

Theorem 2.1 (Mercer 1909) *Let $K(\mathbf{x}, \mathbf{y})$ be a continuous symmetric function in $L_2(\mathcal{X}^2)$. Then, there exists a mapping Φ and an expansion*

$$K(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{\infty} \Phi(\mathbf{x})_i \cdot \Phi(\mathbf{y})_i$$

if and only if, for any compact subset C and $g \in L_2(C)$,

$$\int_{C \times C} K(\mathbf{x}, \mathbf{y}) g(\mathbf{x}) g(\mathbf{y}) d\mathbf{x} d\mathbf{y} \geq 0 \tag{2.13}$$

Note that for specific cases, it may not be easy to check whether Mercer's condition is satisfied, since equation (2.13) must hold for any $g \in L_2(C)$. However, it easy to prove that the condition is satisfied for the polynomial kernel $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + c)^p,\ c \geq 0$ [Burges, 1998].

Let us take an example. Suppose our input data lie in $\mathbb{R}^2$ and we choose $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^2$, the following mapping is valid :

$$\Phi(\mathbf{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2} x_1 x_2 \\ x_2^2 \end{pmatrix}$$

In this case, the feature space is $\mathbb{R}^3$.

Once a kernel K satisfying the Mercer's condition has been chosen, the training algorithm consists of maximizing

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{2.14}$$

and the decision function is

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \right) \tag{2.15}$$

The idea of replacing a dot product by a positive definite kernel K is called the "kernel trick" in the literature. It has first been used by Aizerman et al. [1964]. In the 90s, it has first been applied to the Optimal Separating Hyperplane [Boser et al., 1992] and then to other linear methods such as Principal Component Analysis [Schölkopf et al., 1998b], Linear Discriminant Analysis [Mika et al., 1999; Baudat and Anouar, 2000] or Independent Component Analysis [Bach and Jordan, 2001].

Kernels are also useful to apply learning algorithms to datasets whose inputs are non-Euclidean, such as strings [Haussler, 1999].

The first kernels investigated for the pattern recognition problem were the following [Vapnik, 1995]

Polynomial $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^p$

Radial Basis Function $K(\mathbf{x}, \mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2 / 2\sigma^2)$

Neural Network $K(\mathbf{x}, \mathbf{y}) = \tanh(a\mathbf{x} \cdot \mathbf{y} - b)$

The first one results in a classifier which has a polynomial decision function. The second one gives a Gaussian radial basis function classifier (RBF) and the last one gives a particular kind of two-layer sigmoidal network. For the RBF case, the number of centers (number of support vectors), the center themselves (the support vectors), the weights (α_i) and the threshold (b) are all produced automatically by the SVM training and give excellent results compared to classical RBF [Schölkopf et al., 1997]. In the same way, for the neural network case, the architecture (number of hidden units) is determined by SVM training.

Note, however, that the hyperbolic tangent kernel only satisfies Mercer's condition for certain values of the parameters a and b [Burges, 1999].

For a review of more recent kernels, see [Schölkopf and Smola, 2002].

2.1.5 Experiments

In this section, we present some experimental results using SVMs on some benchmark databases.

Handwritten digit recognition

Handwritten digit recognition has frequently been used as a benchmark for the comparison of classifiers [LeCun et al., 1995]. For this historical reason, SVMs have first been tested on United States Postal Service database (USPS) [LeCun et al., 1989] and the MNIST database [LeCun et al., 1998].

The advantage of this latter is that it contains 60000 training examples and 10000 test examples, yielding accurate comparison between classifiers. On the other hand, the USPS database contains 9298 handwritten digits (7291 for training and 2007 for testing) ans is used for a quick comparison between algorithm. We used this database several times in the experiments reported in this thesis.

Table 2.1 summarizes the test errors of different learning algorithms on the MNIST database. They are taken from [LeCun et al., 1995].

Classifier	Test error
Linear classifier	8.4%
RBF network	3.6%
PCA + quadratic classifier	3.3%
3 nearest neighbors	2.4%
Neural net	1.6%
SVM	1.1%
Tangent distance	1.1%
LeNet4	1.1%
LeNet5	0.9%
Boosted LeNet4	0.7%

Table 2.1: Summary of the test error on MNIST handwritten digit set. Some algorithms include prior knowledge (bottom) and some do not (top and SVM).

The SVM result has been obtained in [Cortes and Vapnik, 1995]. The authors used a soft margin SVM with a polynomial of degree 4 kernel. Incorporating prior knowledge improve strongly the performances, but a standard SVM yields the best results among the classifiers which do not take prior knowledge into account.

From this comparison, the authors of [LeCun et al., 1995] concluded that "*the Optimal margin classifier has excellent accuracy, which is most remarkable, because unlike the other high performance classifiers, it does not include* a priori *knowledge about the problem. In fact, this classifier would do just as well if the image pixels were permuted with a fixed mapping*"

The state-of-the-art result has been obtained by an SVM trained to be invariant to small deformations of the digits [DeCoste and Schölkopf, 2002].

Text categorization

Support Vector Machines turned out to be robust algorithms when the number of input dimension is high. Text classification is a typical example of a large dimensional problem where SVMs have been applied successfully.

A popular benchmark is the Reuters-21578 collection [1] divided in 9603 training examples and 3299 test ones. There are 135 categories and the learning consists of separating one of them from the others. The features used to classify those documents are typically word frequencies within the document resulting in a feature vector which has about 10000 dimensions.

[1] Available at `http://www.research.att.com/~lewis/reuters21578.html`

Joachims is the first one to have tested SVMs on this database [Joachims, 1998]. A summary of his results are shown in table 2.2. A survey and comparison of classifiers used in text categorization can be found in [Aas and Eikvil, 1999].

Classifier	Score
Bayes	72.0
Rocchio	79.9
C4.5	79.4
KNN	82.3
SVM	86.4

Table 2.2: Comparison of classifier on the Reuters-21578. The score is an average of the precision / recall break-even point [Joachims, 1998].

Other applications

A detailed list of applications of SVMs can be found at `http://www.clopinet.com/isabelle/Projects/\\SVM/applist.html`. It includes for instance face recognition [Papageorgiou et al., 1998], image classification [Chapelle et al., 1999a] or bioinformatics applications [Brown et al., 2000; Zien et al., 2000].

2.2 The kernel PCA map

Some algorithms can not be "kernelized" easily: the extension to the non-linear case is not straightforward when the algorithm can not be expressed in terms of dot products only. In this case, one would like to carry out the algorithm in the feature space $\mathcal{H}$ on the points $\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_n)$, but $\mathcal{H}$ can be of very high dimension and in most cases, it is computationally impossible to work directly in this space.

However, even in the case of a high dimensional feature space $\mathcal{H}$, a training set $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ of size n when mapped to this feature space spans a vectorial subspace $E \subset \mathcal{H}$ whose dimension is at most n. The *kernel PCA map*, introduced in [Schölkopf et al., 1998b] and extended in [Tsuda, 1999b; Schölkopf and Smola, 2002] makes use of this idea.

Let $(\mathbf{v}_1, \ldots, \mathbf{v}_n) \in E^n$ be an orthonormal basis of E with each $\mathbf{v}_i$ being a principal axis of the training set in feature space $\{\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_n)\}$. The kernel PCA map $\psi : \mathcal{X} \to \mathbb{R}^n$ is defined coordinatewise as

$$\psi_p(\mathbf{x}) = \Phi(\mathbf{x}) \cdot \mathbf{v}_p, \quad 1 \leq p \leq n.$$

Note that by definition, for all i and j, $\Phi(\mathbf{x}_i)$ and $\Phi(\mathbf{x}_j)$ lie in E and thus

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) = \psi(\mathbf{x}_i) \cdot \psi(\mathbf{x}_j). \tag{2.16}$$

This reflects the fact that if we retain all principal components, kernel PCA is just a basis transform in E, leaving the dot product of training points invariant.

Let us next give the decomposition of the principal directions $(\mathbf{v}_1, \cdots, \mathbf{v}_n)$. For $1 \leq p \leq n$, $\mathbf{v}_p$ can be written as

$$\mathbf{v}_p = \sum_{i=1}^{n} a_{ip} \Phi(\mathbf{x}_i). \tag{2.17}$$

The fact that $(\mathbf{v}_1, \cdots, \mathbf{v}_n)$ are the principal directions in feature space of the training set means that they are eigenvectors of the covariance matrix,

$$C\mathbf{v}_p = \lambda_p \mathbf{v}_p, \quad 1 \leq p \leq n \tag{2.18}$$

where[2]

$$C = \sum_{i=1}^{n} \Phi(\mathbf{x}_i) \Phi(\mathbf{x}_i)^\top. \tag{2.19}$$

[2]We make here the assumption that the data are centered in feature space. See the end of the section on how to center them if they are not.

Combining equations (2.17), (2.18) and (2.19), and multiplying on the left side by $\Phi(\mathbf{x}_k)^\top$, we get

$$\Phi(\mathbf{x}_k)^\top \left(\sum_{i=1}^{n} \Phi(\mathbf{x}_i)\Phi(\mathbf{x}_i)^\top \right) \sum_{j=1}^{n} a_{jp}\Phi(\mathbf{x}_j) = \lambda_p \Phi(\mathbf{x}_k)^\top \sum_{j=1}^{n} a_{jp}\Phi(\mathbf{x}_j),$$

which can be written as

$$\sum_{i,j=1}^{n} K(\mathbf{x}_i,\mathbf{x}_k)K(\mathbf{x}_i,\mathbf{x}_j)a_{jp} = \lambda_p \sum_{j=1}^{n} a_{jp}K(\mathbf{x}_k,\mathbf{x}_j), \quad 1 \leq p,k \leq n$$

In matrix notation, this last equality reads

$$K^2 a_p = \lambda_p K a_p,$$

which is satisfied whenever

$$K a_p = \lambda_p a_p. \tag{2.20}$$

Thus if a_p is an eigenvector of the kernel matrix K, its components can be used in the expansion (2.17). Note that the eigenvalues of K are the same as the ones of the covariance matrix (2.19).

We have now to enforce the constraint that $\mathbf{v}_p$ has unit length. Equations (2.17) and (2.20) yield

$$\mathbf{v}_p^2 = a_p^\top K a_p = \lambda_p a_p^2.$$

Writing the eigendecomposition of K as

$$K = U\Lambda U^\top,$$

with U being an orthonormal matrix and Λ a diagonal one, we have $a_{ip} = U_{ip}/\sqrt{\lambda_p}$, and the kernel PCA map reads

$$\psi(\mathbf{x}) = \Lambda^{-1/2}U^\top \mathbf{k}(\mathbf{x}) = K^{-1/2}\mathbf{k}(\mathbf{x}), \tag{2.21}$$

where

$$\mathbf{k}(\mathbf{x}) = (K(\mathbf{x},\mathbf{x}_1),\ldots,K(\mathbf{x},\mathbf{x}_n))^\top.$$

As a consequence, training a nonlinear SVM on $\{\mathbf{x}_1,\ldots,\mathbf{x}_n\}$ is equivalent to training a linear SVM on $\{\psi(\mathbf{x}_1),\ldots,\psi(\mathbf{x}_n)\}$ and thus, thanks to the nonlinear mapping ψ, we can work directly in the linear space E.

The data points have been assumed to be centered in the feature space. If they are not, one should replace $\Phi(\mathbf{x})$ by $\Phi(\mathbf{x}) - \frac{1}{n}\sum_i \Phi(\mathbf{x}_i)$ and the new dot product to consider is

$$\begin{aligned}
\tilde{K}(\mathbf{x},\mathbf{y}) &= \left(\Phi(\mathbf{x}) - \frac{1}{n}\sum_{i=1}^{n} \Phi(\mathbf{x}_i) \right) \cdot \left(\Phi(\mathbf{y}) - \frac{1}{n}\sum_{i=1}^{n} \Phi(\mathbf{x}_i) \right) \\
&= K(\mathbf{x},\mathbf{y}) - \frac{1}{n}\sum_{i=1}^{n} K(\mathbf{x},\mathbf{x}_i) - \frac{1}{n}\sum_{i=1}^{n} K(\mathbf{y},\mathbf{x}_i) + \frac{1}{n^2}\sum_{i,j=1}^{n} K(\mathbf{x}_i,\mathbf{x}_j)
\end{aligned}$$

2.2.1 Computational complexity

For each point $\mathbf{x}$, the complexity of computing the kernel PCA map $\psi(\mathbf{x})$ (equation 2.21) is $O(n^2)$: E is an n-dimensional space and each basis vector is expressed as a combination of n training points (cf equation 6.9). Also the eigendecomposition of the K is $O(n^3)$ which makes the use of the empirical kernel PCA map prohibitive for large n.

However approximations and computational speed-ups are possible if the "effective dimension" of the feature space is less than the number of training points n. By effective dimension, we mean the dimension of a subspace on which the mapped points $\Phi(\mathbf{x})$ "almost" lie, i.e. the average distance between a point $\Phi(\mathbf{x})$ in feature space and this subspace is negligible. Note that this effective dimension is directly related

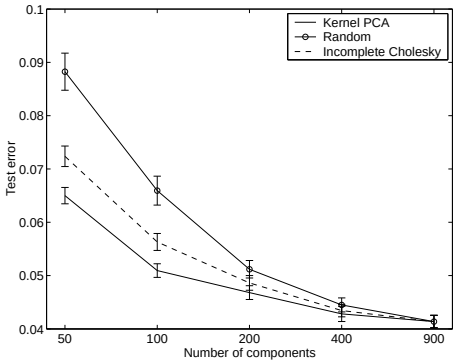


Figure 2.4: Test errors as the function of the number of components used in the low rank methods. The training set size is 900.

to the spectrum decay of the kernel function and for most of them this decay is fast, yielding a low effective dimension [Schölkopf and Smola, 2002; Williamson et al., 2001]. Of course, one could apply the kernel PCA algorithm and keep only the features corresponding to the largest eigenvalues, but this would require already $O(n^3)$ operations to perform the eigendecomposition of K.

A first idea is to select a random subset of the training examples and to compute the empirical kernel PCA map using this subset. This idea has been implemented for Gaussian Processes in [Williams and Seeger, 2001].

The choice of this subset is not as critical as it might seem, but at some computational expense, it is possible to choose the subset in a greedy way in order to get a better approximation [Smola and Schölkopf, 2000].

Finally, it has been proposed in [Fine and Scheinberg, 2001] to use an incomplete Cholesky decomposition to find $K \approx VV^\top$, where V is a $n \times k$ matrix. Performing an SVD decomposition of V gives then an incomplete "eigendecomposition" of K which can be used for the empirical kernel PCA map.

Note that if k is the number of components of this approximate kernel PCA map, all those methods have a complexity of $O(k^2 n)$ to find it. In [Baudat and Anouar, 2001], it has been proposed a greedy approach to find a subset of k training points which approximate well the other points, but the complexity is in $O(k^2 n^2)$.

We performed an experiment to compare the performances obtained with these low rank methods. We considered 3 of them:

Kernel PCA A linear classifier is trained on the first k components given by the kernel PCA algorithm. This is likely to be the best performance one can achieve with k components, but the computational complexity for computing the first k eigenvectors of the Gram matrix is $O(kn^2)$, so it should only be considered as a reference method.

Random A random subset of k points is selected and the complete kernel PCA map is computed using those points.

Incomplete Cholesky Using an incomplete Cholesky decomposition and the SVD algorithm, one can get an approximate empirical kernel PCA map (see above).

Experiments have been carried out on the USPS database, divided into 8 subsets of 900 training examples. An RBF kernel with standard deviation equal to 5 was used. Test errors are plotted in figure 2.4.

2.3 Adapting the vicinity

2.3.1 Adaptive Kernel Widths

It is known in density estimation theory that the quality of the density estimate can be improved using variable kernel widths [Breiman et al., 1977]. In regions of the space where there is little data, it is safer to have a smooth estimate of the density, whereas in the regions of the space there is more data one wants to be as accurate as possible via sharper kernel estimates. The VRM principle can take advantage of these improved density estimates for other problem domains. We consider here the following density estimate:

$$dP_{est}(\mathbf{x}, y) = \frac{1}{n} \sum_i \delta_{y_i}(y) \, \mathcal{N}_{\sigma_i}(\mathbf{x} - \mathbf{x}_i) \, d\mathbf{x} \tag{2.22}$$

where the specific kernel width σ_i for each training example $\mathbf{x}_i$ is computed from the training set. It can be for example a constant times the mean distance of the k nearest neighbors of $\mathbf{x}_i$. It can also be adjusted using unlabeled sample (see chapter 7).

Optimization

In the experiments reported below, the minimization of the vicinal risk has been done through a gradient descent. However, similarly to section 2.1.3, a modification of the optimization problem can lead to a convex and SVM-like one.

We established in section 2.1.3 that the value of the width σ is directly related to the constant C penalizing the training error: σ going to zero is for example equivalent to the hard margin SVM (C goes to infinity). In this section, we are thus more interested by the relative values of the σ_i. For this reason, we will define σ as the mean value of the σ_i and set up the new optimization problem in terms of the *relative* quantities $\gamma_i = \sigma_i / \sigma$.

The convex approximation of minimizing the vicinal risk is to minimize $\sum \xi_i$ under constraints $\xi_i \geq 0$ and

$$\frac{y_i(\mathbf{w} \cdot \mathbf{x}_i + b)}{\|\mathbf{w}\| \, \sigma_i} \geq \sqrt{\frac{\pi}{2}} - \xi_i.$$

This is equivalent to

$$\min \sum_{i=1}^{n} \frac{\xi_i}{\gamma_i}$$

under constraints

$$\xi_i \geq 0, \quad \mathbf{w}^2 = \frac{2}{\pi \sigma^2}$$
$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq \gamma_i - \xi_i. \tag{2.23}$$

Equation (2.23) makes sense intuitively: in regions of the space where the location of a point is not well estimated (large γ), it is safer to enforce a large margin on this point (see figure 2.5).

Taking into account constraints (2.23) the dual writes

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \gamma_i \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j,$$

that one has to maximize subject to constraints: $\sum_{i=1}^{n} \alpha_i y_i = 0$ and $0 \leq \alpha_i \leq C / \gamma_i$.

Experiments

a) Wisconsin Breast Cancer — We made a first test of the method on the Wisconsin breast cancer dataset which contains 569 examples on 30 dimensions. We compared VRM using the set of linear classifiers with various underlying density estimates. The minimization was achieved using gradient descent on the vicinal risk. All hyperparameters were determined using cross-validation. The following table reports results averaged on 100 runs.

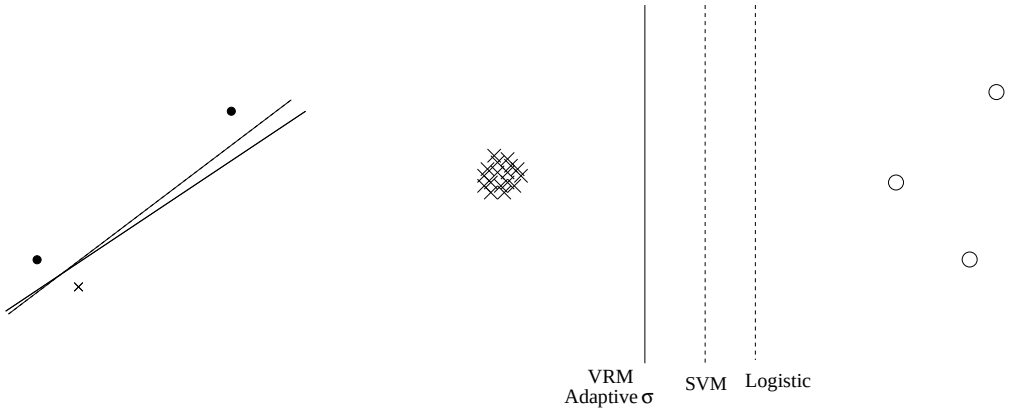


Figure 2.5: Two toy examples illustrating how a linear decision function is influenced by the adaptive width of VRM. In both cases, it tends to push away the hyperplane from points whose location is uncertain. In the example on the right, a logistic classifier (section 1.3.3.0), would put the hyperplane more on the right because there are more crosses than circles.

Training Set	Hard SVM	Soft SVM Best C	VRM Best fixed σ	VRM Adaptive σ_i
10	11.3%	11.1%	10.8%	**9.6%**
20	8.3%	7.5%	6.9%	**6.6%**
40	6.3%	5.5%	5.2%	**4.8%**
80	5.4%	4.0%	3.9%	**3.7%**

The adaptive kernel width σ_i were computed by multiplying a global factor by the average distance of the five closest training examples. The best global factor is determined by cross-validation. These results suggest that VRM with adaptive kernel widths can outperform state of the art classifiers on small training sets.

b) MNIST "1" versus other digits — A second test was performed using the MNIST handwritten digits[3]. We considered the sub-problem of recognizing the ones versus all other digits. The testing set contains 10000 digits (5000 ones and 5000 non-ones). Two training set sizes were considered with 250 or 500 ones and an equal number of non-ones. We simply trained a single linear unit with a sigmoid transfer function using stochastic gradient updates. The slope of the sigmoid is adapted for each example in function of σ_i, which is computed from the training set as the 5/1000th quantile of the distances of all other examples to example $\mathbf{x}_i$. Early stopping was achieved with cross-validation.

Training Set	Hard SVM	VRM Fixed width	VRM Adaptive width
250+250	3.34%	2.79%	**2.54%**
500+500	3.11%	2.47%	**2.27%**
1000+1000	2.94%	2.08%	**1.96%**

The statistical significance of these results can be asserted with very high probability by comparing the list of errors performed by each system [Bottou and Vapnik, 1992]. Again these results suggest that VRM with adaptive kernel widths can be very useful with small training sets.

2.3.2 Adapting the shape of the vicinity

The *Optimal Separating Hyperplane* has been motivated by the VRM principle: it is the one minimizing the vicinal risk when the local density is estimated by spherical Gaussians.

[3] `http://yann.lecun.com/exdb/mnist/index.html`

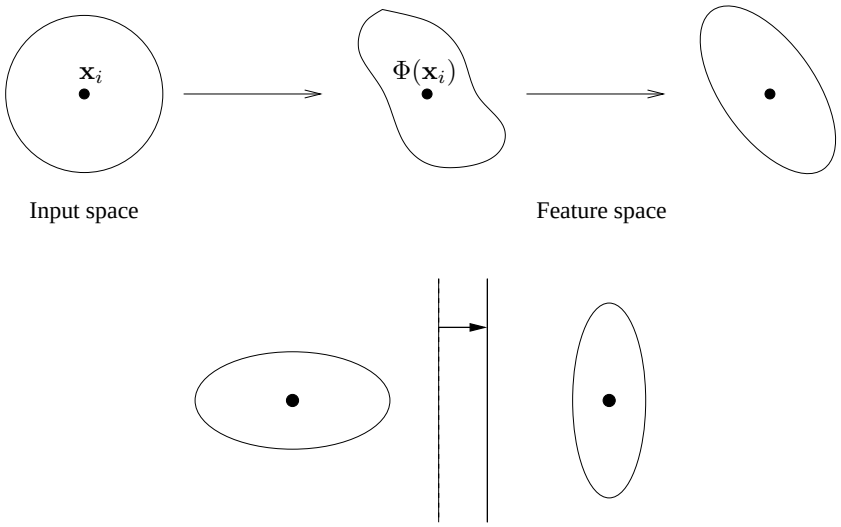


Figure 2.6: VRM in feature space. Top: a sphere in input space (left) is mapped into a "blob" in feature space (middle). This blob should be the new local density estimate $dP_{\Phi(\mathbf{x}_i)}(\mathbf{z})$, but for computational reason, we approximate it by an ellipsoid (right). Bottom: minimizing the vicinal risk can yield a decision function quite different from the maximum margin one (dashed line)

However, if one has additional knowledge, more sophisticated estimation can be used. For example, if one knows that there is a lot of noise in a given direction, then an ellipsoidal gaussian elongated in this direction should be used. One can think about the local estimate $dP_{\mathbf{x}_i}(\mathbf{x})$ of the density as a measure of uncertainty of the location of the point $\mathbf{x}_i$. If no additional knowledge is available, spherical gaussian noise is the most natural assumption.

Non-linear SVMs are hyperplane in a high-dimensional feature space. What about the VRM principle in the feature space ? A sphere in input space in transformed into a "blob" in the feature space, which can be approximated by an ellipsoid (see top of figure 2.6). Depending on the shapes of those ellipsoids, the decision function minimizing the vicinal risk can be significantly different from the maximum margin solution (bottom of figure 2.6).

Note that if the input dimension is d the sphere in the input space is mapped into a d-dimensional manifold in feature space. For more details about the geometry of the feature space, see [Burges, 1999].

Vicinal Risk in feature space

We would like to compute the local covariance matrix in feature space corresponding to a sharp spherical gaussian distribution in input space. However, since the feature space is a high-dimensional one, it is impossible to work directly in this space. Instead, we will work on the subspace E spanned by the training examples and compute the covariance matrix of the manifold projected onto this subspace. For this purpose, we will use the *empirical kernel PCA map* defined in section 2.2.

The covariance matrix around the point $\mathbf{x}_p$ is

$$C_p(\sigma) = \int (\psi(\mathbf{x}) - \psi(\mathbf{x}_p))(\psi(\mathbf{x}) - \psi(\mathbf{x}_p))^\top \mathcal{N}_\sigma(\mathbf{x}_p - \mathbf{x}) d\mathbf{x},$$

where $\psi(\mathbf{x})$ is defined by (2.21).

A first question of interest is how is the shape of the vicinity of the point $\mathbf{x}_p$ in the empirical kernel PCA space when σ goes to zero. In other words, we would like to compute $\lim_{\sigma \to 0} C_p(\sigma)/\sigma^2$. A first order expansion of $\mathbf{k}(\mathbf{x}) \equiv (k(\mathbf{x}, \mathbf{x}_1) \dots k(\mathbf{x}, \mathbf{x}_n))^\top$ around $\mathbf{x}_p$ gives

$$\mathbf{k}(\mathbf{x}_p + \boldsymbol{\varepsilon}) = \mathbf{k}(\mathbf{x}_p) + \nabla \mathbf{k}(\mathbf{x}_p)\ \boldsymbol{\varepsilon} + o(\|\boldsymbol{\varepsilon}\|),$$

where

$$(\nabla \mathbf{k}(\mathbf{x}_p))_{ij} = \left.\frac{\partial K(\mathbf{x}, \mathbf{x}_i)}{\partial x_j}\right|_{\mathbf{x}=\mathbf{x}_p}.$$

Let M_p be the $n \times d$ matrix

$$M_p = \Lambda^{-1/2} U^\top \nabla \mathbf{k}(\mathbf{x}_p). \tag{2.24}$$

Combining (2.21) and (2.24), we have

$$\psi(\mathbf{x}_p + \boldsymbol{\varepsilon}) - \psi(\mathbf{x}_p) = M_p \boldsymbol{\varepsilon} + o(\|\boldsymbol{\varepsilon}\|)$$

and when σ goes to 0,

$$\begin{aligned} C_p(\sigma) &\approx M_p \left(\int \boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^\top \; N(\boldsymbol{\varepsilon}) d\boldsymbol{\varepsilon} \right) M_p^\top \\ &= M_p M_p^\top \frac{\sigma^2}{d} \end{aligned} \tag{2.25}$$

Note that since M_p is $n \times d$ matrix, the rank of C_p is at most d, which can be explained by the fact that the data lie on a d-dimensional manifold in feature space.

Having computed the shape of the vicinity around a point $\Phi(\mathbf{x}_p)$ a feature space, the vicinal risk for this point is

$$\int \mathrm{I}_{\mathrm{sgn}(\mathbf{w} \cdot \mathbf{x} + b) \neq y_p} \; dP_{\mathbf{x}_p}(\mathbf{x}), \tag{2.26}$$

where

$$dP_{\mathbf{x}_p}(\mathbf{x}) = \frac{1}{\sqrt{2\pi \det C_p}} \exp\left(-\frac{1}{2} (\mathbf{x} - \psi(\mathbf{x}_p))^\top C_p^{-1} (\mathbf{x} - \psi(\mathbf{x}_p)) \right) d\mathbf{x}. \tag{2.27}$$

Here $\mathbf{x} \in \mathbb{R}^n$ corresponds to a point in the linear subspace spanned by the training examples in feature space. Note that equation (2.27) is not well defined since in general the matrix C_p has rank d. One can fix this problem by adding an infinitesimal small ridge to this matrix.

Making the change of variable $\mathbf{x} \leftarrow C_p^{-1/2}(\mathbf{x} - \psi(\mathbf{x}_p))$, equation (2.26) reads

$$\int \mathrm{I}_{\mathrm{sgn}(\mathbf{w} \cdot (C_p^{1/2}\mathbf{x} + \psi(\mathbf{x}_p)) + b) \neq y_p} d\mathcal{N}_1(\mathbf{x})$$

and finally the vicinal risk turns out to be equal to

$$R_{vic}(\mathbf{w}, b) = \frac{1}{n} \sum_{i=1}^{n} \varphi\left(-\frac{y_i(\mathbf{w} \cdot \psi(\mathbf{x}_i) + b)}{\sqrt{\mathbf{w}^\top C_i \mathbf{w}}} \right). \tag{2.28}$$

Note that in the particular case where $C_p = \sigma^2 I$, equations (1.15) and (2.28) coincide.

Interpretation in input space

In the SVM framework, the hyperplane $\mathbf{w}$ is expressed as a linear combination of the training point in feature space,

$$\mathbf{w} = \sum_{i=1}^{n} \mu_i \psi(\mathbf{x}_i).$$

Usually, $\mu_i = y_i \alpha_i$.

Taking into account the expression of the empirical kernel PCA map ψ (2.21), the hyperplane can be expressed in matrix notations

$$\mathbf{w} = \Lambda^{-1/2} U^\top K \boldsymbol{\mu} = \Lambda^{1/2} U^\top \boldsymbol{\mu}, \tag{2.29}$$

since by definition the eigendecomposition of the gram matrix K is $K = U \Lambda U^\top$.

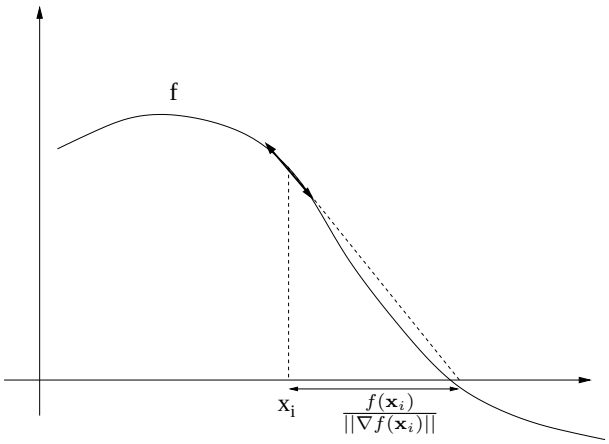


Figure 2.7: In the non-linear case, an approximation of the distance of a point to the decision function can be computed with the gradient information

Combining equations (2.25), (2.24) and (2.29) , the denominator of equation (2.28) is

$$\begin{aligned}\sqrt{\mathbf{w}^\top C_p \mathbf{w}} &= \frac{\sigma}{\sqrt{d}}||M_p^\top \mathbf{w}|| \\ &= \frac{\sigma}{\sqrt{d}}||(\nabla \mathbf{k}(\mathbf{x}_p))^\top U \Lambda^{-1/2}\Lambda^{1/2}U^\top \boldsymbol{\mu}|| \\ &= \frac{\sigma}{\sqrt{d}}||(\nabla \mathbf{k}(\mathbf{x}_p))^\top \boldsymbol{\mu}||\end{aligned}$$

Since the decision function is $f(\mathbf{x}) = \mathbf{w}\cdot\psi(\mathbf{x}) + b = \mathbf{k}(\mathbf{x})\cdot\mu + b$, the vicinal risk (2.28) becomes

$$R_{vic}(\mathbf{w}, b) = \frac{1}{n}\sum_{i=1}^{n}\varphi\left(-\frac{\sqrt{d}}{\sigma}\frac{y_i f(\mathbf{x}_i)}{||\nabla f(\mathbf{x}_i)||}\right).$$

Let us interpret the quantity inside the function φ. In the linear case,

$$\frac{y_i f(\mathbf{x}_i)}{||\nabla f(\mathbf{x}_i)||} = \frac{y_i(\mathbf{w}\cdot\mathbf{x}_i + b)}{||\mathbf{w}||},$$

which is the margin of the point $\mathbf{x}_i$.

However, in the non-linear case, this quantity can be interpreted as an approximation of margin in *input space*. For an intuitive explanation, see figure 2.7.

More formally, the margin in input space of a point $\mathbf{x}_i$ to the decision function is:

$$m_i = \min\{||\mathbf{h}||/\ f(\mathbf{x}_i + \mathbf{h}) = 0\}.$$

A first order approximation of f gives

$$f(\mathbf{x}_i + \mathbf{h}) \approx f(\mathbf{x}_i) + \mathbf{h}\cdot\nabla f(\mathbf{x}_i).$$

Solving in $\mathbf{h}$ gives

$$m_i = \frac{|f(\mathbf{x}_i)|}{||\nabla f(\mathbf{x}_i)||}.$$

Of course this approximation is good only for points near the decision function, but we are actually only interested in the margin of the points near the decision function.

Minimization of the vicinal risk

Using the empirical kernel PCA map, the vicinal risk in feature space is given by equation (2.28) for any $\mathbf{w}$ in the empirical kernel PCA space (the linear subspace spanned by the training examples in feature space).

As before, it would be nice to approximate the minimization of (2.28) as an SVM-like optimization problem, but we did not manage to reach this goal, mainly because the index i appears both in the numerator and denominator of (2.28). However, we will show that this problem is quasiconvex and can hence by solved efficiently by any kind of Newton's algorithm [Dennis and Schnabel, 1983].

Definition 2.2 (Quasiconvexity, Dacorogna 1989) *A function is quasiconvex if its level sets are convex.*

As a direct consequence of this definition, quasiconvex functions do not have any *local minima*. They have the following properties

Proposition 2.3 (Properties of quasiconvex functions, Boyd and Vandenberghe 2004)

- *A function f is quasiconvex if and only if*

$$\forall x, y, \forall \theta \in [0,1],\ \ f(\theta x + (1-\theta)y) \leq \max\{f(x), f(y)\}$$

- *The* max *of quasiconvex functions is quasiconvex.*
- *Twice differentiable quasiconvex functions satisfy*

$$\forall x, \lambda \quad \lambda^\top \nabla f(x) = 0 \Longrightarrow \lambda^\top \nabla^2 f(x) \lambda \geq 0. \tag{2.30}$$

This implies that the Hessian $\nabla^2 f$ can have at most one negative eigenvalue.

Let us prove the following lemma

Lemma 2.4 *Let $\mathbf{x} \in \mathbb{R}^n$ and C an $n \times n$ positive definite symmetric matrix. Then the functional*

$$F:\ \mathbf{w} \mapsto \frac{\sqrt{\mathbf{w}^\top C \mathbf{w}}}{\mathbf{w} \cdot \mathbf{x}}$$

is quasiconvex on the convex set $V = \{\mathbf{w},\ \mathbf{w} \cdot \mathbf{x} > 0\}$.

Proof: Let $\mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^n$ and θ_1, θ_2 such that $\theta_1 + \theta_2 = 1$. Using proposition 2.3, one should try to show that $F(\theta_1 \mathbf{w}_1 + \theta_2 \mathbf{w}_2) \leq \max(F(\mathbf{w}_1), F(\mathbf{w}_2))$.

For $i = 1, 2$, let $\tilde{\mathbf{w}}_i = \mathbf{w}_i / \sqrt{\mathbf{w}_i^\top C \mathbf{w}_i}$ and $\tilde{\theta}_i = \theta_i \sqrt{\mathbf{w}_i^\top C \mathbf{w}_i} / \sum \theta_i \sqrt{\mathbf{w}_i^\top C \mathbf{w}_i}$. Since F is invariant by scaling,

$$F(\theta_1 \mathbf{w}_1 + \theta_2 \mathbf{w}_2) = F(\tilde{\theta}_1 \tilde{\mathbf{w}}_1 + \tilde{\theta}_2 \tilde{\mathbf{w}}_2).$$

Using first the convexity of $\mathbf{w} \mapsto \sqrt{\mathbf{w}^\top C \mathbf{w}}$ and then the convexity of $\mathbf{w} \mapsto 1/\mathbf{w} \cdot \mathbf{x}$ on $\{\mathbf{w},\ \mathbf{w} \cdot \mathbf{x} \geq 0\}$, we have

$$F(\tilde{\theta}_1 \tilde{\mathbf{w}}_1 + \tilde{\theta}_2 \tilde{\mathbf{w}}_2) \leq \frac{1}{(\tilde{\theta}_1 \tilde{\mathbf{w}}_1 + \tilde{\theta}_2 \tilde{\mathbf{w}}_2) \cdot \mathbf{x}} \leq \tilde{\theta}_1 F(\tilde{\mathbf{w}}_1) + \tilde{\theta}_2 F(\tilde{\mathbf{w}}_2) \leq \max(F(\mathbf{w}_1), F(\mathbf{w}_2)) \quad \square$$

Note that the Hessian of F has at most one negative eigenvalue,

$$\nabla^2 F = \frac{\frac{(\mathbf{w}\cdot\mathbf{x})^2}{\sqrt{\mathbf{w}^\top C \mathbf{w}}} \left(C - \frac{C \mathbf{w} \mathbf{w}^\top C}{\mathbf{w}^\top C \mathbf{w}} \right) - \frac{\mathbf{w}\cdot\mathbf{x}}{\sqrt{\mathbf{w}^\top C \mathbf{w}}} \left(C \mathbf{w} \mathbf{x}^\top + \mathbf{x} \mathbf{w}^\top C \right) + 2 \mathbf{x} \mathbf{x}^\top \sqrt{\mathbf{w}^\top C \mathbf{w}}}{(\mathbf{w} \cdot \mathbf{x})^3}$$

The first term is positive by Cauchy-Schwartz inequality and the sum of the two other terms is a matrix of rank 2 with one positive eigenvalue and a negative one.

It is also straightforward to check that F satisfies condition (2.30).

Corollary 2.5 *When $\sigma \to 0$, the vicinal risk (2.28) is a function of*

$$\max_i \frac{\sqrt{\mathbf{w}^\top C_i \mathbf{w}}}{y_i(\mathbf{w} \cdot \psi(\mathbf{x}_i) + b)}. \tag{2.31}$$

This quantity is quasiconvex in $(\mathbf{w}, b)$ on the version space $V = \{(\mathbf{w}, b),\ y_i(\mathbf{w} \cdot \psi(x_i) + b) \geq 0,\ 1 \leq i \leq n\}$.

This corollary is a direct consequence of lemma 2.4 and proposition (2.3).

We will now present two different ways of minimizing the vicinal risk. The quasiconvexity established in the previous corollary ensures that there is no local minimum.

- Gradient descent on (2.28). At each step, a Newton step is computed, but since the Hessian has one negative eigenvalue, a ridge is added to make it positive. More precisely, we implemented a similar algorithm as the Levenberg-Marquardt [Press et al., 1992]. The quasiconvexity of the function made the optimization easy: in our experiments, after just a couple of Newton steps, the minimum was reached.

- To have an optimization problem similar to the SVM one, we considered the following fix point algorithm

  1. Fix $\gamma_i = 1, \quad 1 \leq i \leq n$.
  2. Minimize $\mathbf{w}^2$ under constraints $y_i(\mathbf{w} \cdot \psi(\mathbf{x}_i) + b) \geq \gamma_i$.
  3. Set $\gamma_i = \sqrt{\mathbf{w}^\top C_i \mathbf{w}}/\|\mathbf{w}\|$
  4. Go back to step 2 or stop when $\boldsymbol{\gamma}$ has reached a fixed point.

  The motivation for this procedure is that the value $\boldsymbol{\gamma}$ computed using the optimal values of $(\mathbf{w}, b)$ minimizing equation (2.31) is a fixed point of the procedure. However, we do not have any guarantee of convergence for this algorithm.

Choice of the kernel

Remember that minimizing the vicinal risk in the feature space is approximately equivalent to maximizing the margin in input space. But if the kernel has a large capacity (the RBF kernel for example has an infinite VC dimension [Burges, 1998] and can thus approximate any function), then the function minimizing the vicinal risk is likely to be very similar the one nearest neighbor classification rule.

Let us interpret this phenomenon in feature space. Suppose that N is the dimension of the feature space and that we do not require the normal hyperplane $\mathbf{w}$ to belong to the subspace E of dimension n spanned by the training examples. Let C_i be the $N \times N$ local covariance matrix associated with the point $\mathbf{x}_i$ (if projected on E, that would be (2.25)). As before, the data lie in feature space on a d-dimensional manifold and C_i has rank d. Then it is likely to find an hyperplane $\mathbf{w}$ separating the data such that $\mathbf{w}^\top C_i \mathbf{w} = 0,\ \forall i$, yielding a vicinal risk of 0.

In summary, if the capacity of the kernel is too large, then maximizing the margin in input space gives a decision function similar to the one-nearest-neighbor classification rule. This latter is not so bad since it is asymptotically almost optimal [Devroye et al., 1996], but better results are likely to be reached if the set of functions is restrained. Remember that the VRM induction principle is in between ERM and density estimation. We improved the density estimation in the feature space, but it the kernel is "too rich", we do not take advantages of ERM. To reduce the complexity, there are two options:

Dimension reduction The fact that the optimization of $\mathbf{w}$ is done using the kernel PCA map restrains it already to a subspace of dimension n. Thus the dimension has been reduced from the dimension of feature space N to n. It can be further reduced by keeping only the components of the empirical kernel PCA map corresponding to the largest eigenvalues.

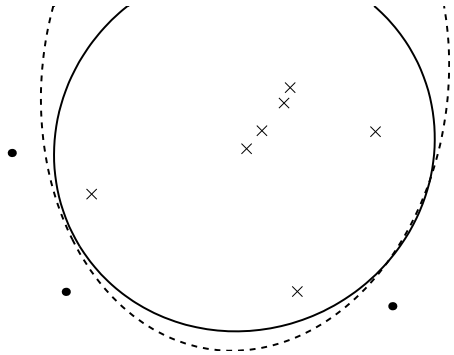


Figure 2.8: The SVM solution (dashed line) does not maximize the margin in input space, whereas the VRM solution does it approximatively. This can be soon with the point on left side.

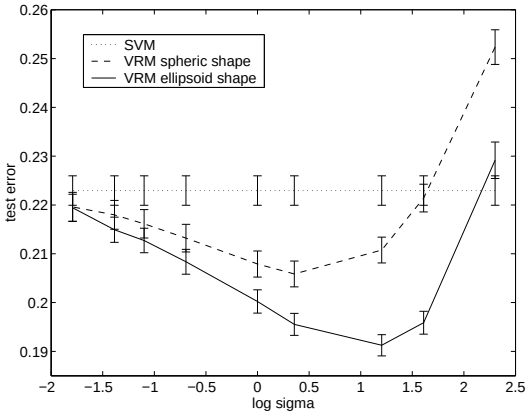


Figure 2.9: Comparison of the test errors on the USPS dataset for different values of σ, the width of the ellipsoid in the density estimation used by VRM.

Regularization The fact that SVMs do not overfit in a high-dimensional feature space can be explained by a regularization argument [Evgeniou et al., 2000]: minimizing $\mathbf{w}^2$ is equivalent to put an upper-bound on $\mathbf{w}^2$ and thus reduce the complexity of the class of functions. In the same way, we could replace C_i (2.25) by

$$\lambda C_i + (1 - \lambda)I,$$

where λ is a trade-off value: $\lambda = 1$ corresponds to the technique described in this section, where $\lambda = 0$ results in the minimization of $\mathbf{w}^2$ as for SVMs.

Experiments

The difference between maximizing the margin in input space and in feature space can be seen on the toy example of figure 2.8: there are 10 training examples, the true decision boundary is a circle and the kernel used is a polynomial of degree 2. The fact that the SVM solution does not maximize the margin in input space was already noticed in a similar work that the one presented here [Tong and Koller, 2000].

Real world experiments have been carried out on the USPS dataset consisting of 7291 training examples and 2007 test ones. For computational reasons, the training set has been divided in 145 subsets of 50 examples and the original 16×16 images have been subsampled to 4×4 images. The goal was to separate digits 0 to 4 against 5 to 9. For this purpose, a polynomial kernel of degree 4 was used (see also Schölkopf et al. [1995]). Results can be found in figure 2.9.

The VRM with spherical shape computes the loss of a training point $\mathbf{x}_i$ as a function of $y_i(\mathbf{w} \cdot \mathbf{x}_i + b)/\|\mathbf{w}\|$ and is equivalent ta a hard margin SVM when the width σ goes to 0, but when σ is not small, it is equivalent to a soft margin SVM. The VRM with a ellipsoidal shape described in this section yields an improvement, but not when σ goes to 0, whereas we expected an improvement also in that case.

Part II

Choosing the paramaters of an SVM

Introduction

Motivation

The SVM algorithm usually depends on several parameters. One of them, denoted C, controls the trade-off between margin maximization and error minimization. Other parameters appear in the non-linear mapping into feature space. They are called *kernel parameters*. For simplicity, we will use a classical trick that allows us to consider C as a kernel parameter, so that all parameters can be treated in a unified framework.

It is widely acknowledged that a key factor in an SVM performance is the choice of the kernel. However, in practice, very few different types of kernels have been used due to the difficulty of appropriately tuning the parameters. We present here a technique that allows to deal with a large number of parameters and thus allows to use more complex kernels.

Another potential advantage of being able to tune a large number of parameters is the possibility of rescaling the attributes. Indeed, when no a priori knowledge is available about the meaning of each of the attributes, the only choice is to use spherical kernels (i.e. give the same weight to each attribute). But one may expect that there is a better choice for the shape of the kernel since many real-world database contain attributes of very different natures. There may thus exist more appropriate *scaling factors* that give the right weight to the right feature. For example, we will see how to use radial basis function kernels (RBF) with as many different scaling factors as input dimensions:

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(-\sum_i \frac{(x_i - y_i)^2}{2\sigma_i^2}\right).$$

The usual approach is to consider $\sigma = \sigma_1 = \cdots = \sigma_n$ and to try to pick the best value for σ. However, using the proposed method, we can choose automatically good values for the scaling factors σ_i. Indeed, these factors are precisely parameters of the kernel.

Moreover, we will demonstrate that the problem of *feature selection* can be addressed with the same framework since it corresponds to finding those attributes which can be rescaled with a zero factor without harming the generalization.

We thus see that tuning kernel parameters is something extremely useful and a procedure that allows to do this would be a versatile tool for various tasks such as finding the right shape of the kernel, feature selection, finding the right trade-off between error and margin, outlier detection. All this gives a rationale for developing such techniques.

Approach

Usually the parameters are tuned using a minimax approach: maximize the margin over the hyperplane coefficients and minimize an estimate of the generalization error over the set of kernel parameters. But when there are multiple parameters, the naive strategy which is an exhaustive search in the parameter space becomes intractable since it would correspond to running the algorithm on every possible value of the parameter vector (up to some discretization). Instead, we propose to do this search using a gradient descent approach.

Outline

The section is organized as follows. The next chapter introduces several ways of assessing the generalization error, most of them being based on the leave-out-error. Then in chapter 4, analytic formulations of the gradients of those estimates are given and the gradient descent approach is presented (this chapter is a direct adaptation from Chapelle et al. 2002b). Finally, chapter 5 presents some applications of this methodology to feature selection and outlier detection.

Chapter 3

Estimating the performance of an SVM

Ideally we would like to choose the value of the kernel parameters that minimize the true risk of the SVM classifier. Unfortunately, since this quantity is not accessible, one has to build estimates or bounds for it. In this section, we present several measures of the expected error rate of an SVM.

3.1 Literature review

3.1.1 Single validation estimate

If one has enough data available, it is possible to estimate the true error on a validation set. This estimate is unbiased and its variance gets smaller as the size of the validation set increases. If the validation set is $\{(\mathbf{x}'_i, y'_i)\}_{1 \leq i \leq p}$, the estimate is

$$T = \frac{1}{p} \sum_{i=1}^{p} \Psi(-y'_i f(\mathbf{x}'_i)), \tag{3.1}$$

where Ψ is the step function: $\Psi(x) = 1$ when $x > 0$ and $\Psi(x) = 0$ otherwise.

3.1.2 VC bounds

The first estimate of the generalization involves the VC dimension of margin classifiers. It has been shown [Vapnik, 1995; Bartlett and Shawe-Taylor, 1999] that the capacity of such algorithms is bounded by

$$\frac{R^2}{M^2},$$

where R is the radius of the smallest sphere enclosing the training points and M is the margin obtained on the training points.

More precisely, the following theorem [Bartlett and Shawe-Taylor, 1999] holds:

Theorem 3.1 *There is a constant c such that with probability at least $1 - \delta$ over n i.i.d. training examples, if the classifier built by an SVM has margin at least M on all training examples except for a fraction a/n of these, then the generalization error of this classifier is no more than*

$$\frac{a}{n} + \sqrt{\frac{c}{n}\left(\frac{R^2}{M^2}\log^2 n + \log(1/\delta)\right)},$$

with

$$\frac{y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i) + b)}{\|\mathbf{w}\|} \geq M \quad \textit{except for at most } a \textit{ points}$$

$$R = \min_{\mathbf{a}} \max_{i} \|\Phi(\mathbf{x}_i) - \mathbf{a}\|$$

If there is no training error, the test error is in general directly related to the capacity of the learning algorithm. Thus, in the separable case, $T = R^2/M^2$ is an estimate of the generalization performance.

This estimate can also be derived when bounding the leave-one-out error next section will show.

3.1.3 Leave-one-out bounds

The *leave-one-out* procedure consists of removing from the training data one element, constructing the decision rule on the basis of the remaining training data and then testing on the removed element. In this fashion one tests all n elements of the training data (using n different decision rules). Let us denote the number of errors in the leave-one-out procedure by $\mathcal{L}(\mathbf{x}_1, y_1, ..., \mathbf{x}_n, y_n)$. It is known that the the leave-one-out procedure gives an almost unbiased estimate of the expected generalization error:

Lemma 3.2 (Leave-one-out error, Luntz and Brailovsky 1969)

$$Ep_{err}^{n-1} = \frac{1}{n} E(\mathcal{L}(\mathbf{x}_1, y_1, ..., \mathbf{x}_n, y_n)),$$

where p_{err}^{n-1} is the probability of test error for the machine trained on a sample of size $n-1$ and the expectations are taken over the random choice of the sample.

Although this lemma makes the leave-one-out estimator a good choice when estimating the generalization error, it is nevertheless very costly to actually compute since it requires running the training algorithm n times. The strategy is thus to upper bound or approximate this estimator by an easy to compute quantity T having, if possible, an analytical expression.

If we denote by f^0 the classifier obtained when all training examples are present and f^i the one obtained when example i has been removed, we can write:

$$\mathcal{L}(\mathbf{x}_1, y_1, ..., \mathbf{x}_n, y_n) = \sum_{p=1}^{n} \Psi(-y_p f^p(\mathbf{x}_p)), \tag{3.2}$$

which can also be written as

$$\mathcal{L}(\mathbf{x}_1, y_1, ..., \mathbf{x}_n, y_n) = \sum_{p=1}^{n} \Psi(-y_p f^0(\mathbf{x}_p) + y_p(f^0(\mathbf{x}_p) - f^p(\mathbf{x}_p))).$$

Thus, if U_p is an upper bound for $y_p(f^0(\mathbf{x}_p) - f^p(\mathbf{x}_p))$, we will get the following upper bound on the leave-one-out error:

$$\mathcal{L}(\mathbf{x}_1, y_1, ..., \mathbf{x}_n, y_n) \leq \sum_{p=1}^{n} \Psi(U_p - 1),$$

since for hard margin SVMs, $y_p f^0(\mathbf{x}_p) \geq 1$ and Ψ is monotonically increasing.

Support vector count

Since removing a non-support vector from the training set does not change the solution computed by the machine (i.e. $U_p = f^0(\mathbf{x}_p) - f^p(\mathbf{x}_p) = 0$ for $\mathbf{x}_p$ non-support vector), we can restrict the preceding sum to support vectors and upper bound each term in the sum by 1 which gives the following bound on the number of errors made by the leave-one-out procedure [Vapnik, 1995]:

$$T = \frac{N_{SV}}{n},$$

where N_{SV} denotes the number of support vectors.

Jaakkola-Haussler bound

For SVMs without threshold, analyzing the optimization performed by the SVM algorithm when computing the leave-one-out error, Jaakkola and Haussler [1999] proved the inequality:

$$y_p(f^0(\mathbf{x}_p) - f^p(\mathbf{x}_p)) \leq \alpha_p^0 K(\mathbf{x}_p, \mathbf{x}_p) = U_p \tag{3.3}$$

which leads to the following upper bound:

$$T = \frac{1}{n}\sum_{p=1}^{n} \Psi(\alpha_p^0 K(\mathbf{x}_p, \mathbf{x}_p) - 1).$$

Opper-Winther bound

For hard margin SVMs without threshold, Opper and Winter [1999] used a method inspired from linear response theory to prove the following: under the assumption that the set of support vectors does not change when removing the example p, we have

$$y_p(f^0(\mathbf{x}_p) - f^p(\mathbf{x}_p)) = \frac{\alpha_p^0}{(\mathbf{K}_{SV}^{-1})_{pp}},$$

where $\mathbf{K}_{SV}$ is the matrix of dot products between support vectors; leading to the following estimate:

$$T = \frac{1}{n}\sum_{p=1}^{n} \Psi\left(\frac{\alpha_p^0}{(\mathbf{K}_{SV}^{-1})_{pp}} - 1\right).$$

Radius-margin bound

For SVMs without threshold and with no training errors, Vapnik [1998] proposed the following upper bound on the number of errors of the leave-one-out procedure:

$$T = \frac{R^2}{M^2},$$

where M is the margin as defined in theorem 3.1 and $R = \max \|\Phi(\mathbf{x}_i)\|$, the maximum norm of the training points in feature space.

Wahba's bound

Denoting by g the cost function used to penalize the errors made by the leave-one-out procedure, Wahba et al. [2000] introduce the *prediction* function defined as:

$$\mu(f) = \sum_{y \in \{-1,1\}} \frac{\partial g(yf)}{\partial f}$$

Lemma 3.3 *The solution computed by the SVM when example p is removed is optimal for the original variational problem if y_p is replaced by $\mu(f^p(\mathbf{x}_p))$.*

Let $F(y)$ the function which gives the output on $\mathbf{x}_p$ of the function learned with y_p replaced by $y \in \mathbb{R}$. A first order expansion of F as well as the previous lemma allows to write the following:

$$f^0(\mathbf{x}_p) - f^p(\mathbf{x}_p) \approx \frac{\partial f^0(\mathbf{x}_p)}{\partial y_p}(y_p - \mu(f^0(\mathbf{x}_p)))$$

In the case of the SVM loss, $g(x) = x + 1$ if $x > -1$, 0 otherwise. Thus we have $\mu(f) = 0$ for $f \in [-1, 1]$, $\mu(f) = -1$ for $f < -1$ and $\mu(f) = 1$ for $f > 1$. Finally, using that g is 1-Lipschitz, we obtain the following approximate bound on the number of error made by the leave-one-out procedure:

$$\sum_{p=1}^{n} g(-y_p f^0(\mathbf{x}_p)) + \sum_{y_p f^0(\mathbf{x}_p) < -1} 2\alpha_p^0 K(\mathbf{x}_p, \mathbf{x}_p) + \sum_{y_p f^0(\mathbf{x}_p) \in [-1,1]} \alpha_p^0 K(\mathbf{x}_p, \mathbf{x}_p)$$

Note that in the hard margin SVM case, this bound turns out to be

$$T = \sum \alpha_p^0 K(\mathbf{x}_p, \mathbf{x}_p),$$

which can be seen as an upper bound of the Jaakkola-Haussler one since $\Psi(x-1) \leq x$ for $x \geq 0$.

3.2 Span estimate

In this section, bounds and tight estimates on the error of the leave-one-out procedure are derived. Using lemma 3.2, this latter is unbiased estimate of the generalization error. Further work could give bounds with high probability using the *stability* concept introduced in [Bousquet and Elissee, 2002]. For this purpose, it would be necessary to bound $\left\|f^0 - f^p\right\|_\infty$. This could be done using the results of section 4.4.6.

We first need to introduce a geometrical concept called the *span* of a set of support vectors Vapnik and Chapelle [2000].

3.2.1 Span of the set of Support Vectors

Let us first consider the separable case. Suppose that

$$(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)$$

is a set of support vectors and

$$\boldsymbol{\alpha}^0 = (\alpha_1^0, ..., \alpha_n^0)$$

is the vector of Lagrange multipliers for the optimal hyperplane.

For any fixed support vector $\mathbf{x}_p$ we define the set Λ_p as a constrained linear combination of the points $\{\mathbf{x}_i\}_{i \neq p}$:

$$\Lambda_p = \left\{ \sum_{i=1,\ i\neq p}^{n} \lambda_i \mathbf{x}_i \ : \ \sum_{i=1,\ i\neq p}^{n} \lambda_i = 1, \text{ and } \forall i \neq p, \ \ \alpha_i^0 + y_i y_p \alpha_p^0 \lambda_i \geq 0 \right\}$$

Note that λ_i can be less than 0.

We also define the quantity S_p, which we call the *span* of the support vector $\mathbf{x}_p$ as the distance between $\mathbf{x}_p$ and this set (see figure 3.1)

$$S_p^2 = d^2(\mathbf{x}_p, \Lambda_p) = \min_{\mathbf{x} \in \Lambda_p} (\mathbf{x}_p - \mathbf{x})^2, \tag{3.4}$$

As shown in figure 3.2, it can happen that $\mathbf{x}_p \in \Lambda_p$, which implies

$$S_p = d(\mathbf{x}_p, \Lambda_p) = 0.$$

In this case, the point $\mathbf{x}_p$ is a linear combination of the other support vectors and the expansion of the separating hyperplane could not include this point. As an aside, the decision function will be unchanged after removing this point.

Intuitively, for smaller $S_p = d(\mathbf{x}_p, \Lambda_p)$, the leave-one-out procedure is less likely to make an error on the vector $\mathbf{x}_p$. Indeed, we will prove (see lemma 3.5)) that if $S_p < 1/(D\alpha_p^0)$ (D is the diameter of the smallest sphere containing the training points), then the leave-one-out procedure classifies $\mathbf{x}_p$ correctly.

By setting $\lambda_p = -1$, we can rewrite S_p as :

$$S_p^2 = \min \left\{ \left(\sum_{i=1}^{n} \lambda_i \mathbf{x}_i \right)^2 \ : \ \lambda_p = -1, \ \sum_{i=1}^{n} \lambda_i = 0, \ \alpha_i^0 + y_i y_p \alpha_p^0 \lambda_i \geq 0 \right\} \tag{3.5}$$

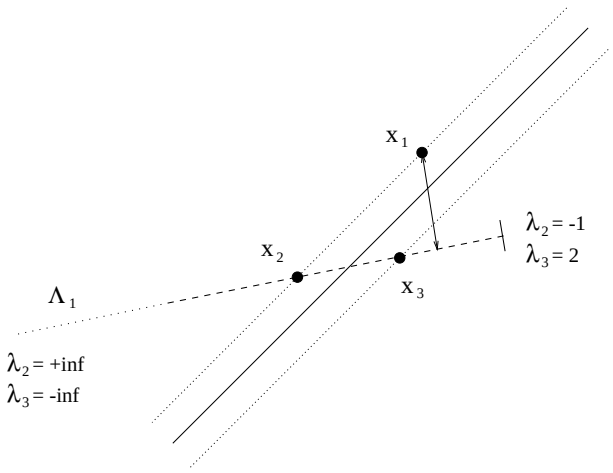


Figure 3.1: *Consider the 2D example above : 3 support vectors with* $\alpha_1 = \alpha_2 = \alpha_3/2$. *The set* Λ_1 *is the semi-opened dashed line :* $\Lambda_1 = \{\lambda_2 \mathbf{x}_2 + \lambda_3 \mathbf{x}_3,\ \lambda_2 + \lambda_3 = 1,\ \lambda_2 \geq -1,\ \lambda_3 \leq 2\}$.

The maximal value of S_p is called the S-span

$$S = \max\{d(\mathbf{x}_1, \Lambda_1), ..., d(\mathbf{x}_n, \Lambda_n)\} = \max_p S_p.$$

We will prove (cf lemma (3.4) below) that $S_p \leq D_{SV}$, where D_{SV} is the diameter of the smallest sphere enclosing all the support vectors. Therefore,

$$S \leq D_{SV}. \tag{3.6}$$

But depending on relative position of the support vectors and on $\boldsymbol{\alpha}^0 = (\alpha_1^0, ..., \alpha_n^0)$ the value of the span S can be much less than diameter D_{SV} of the support vectors. Indeed, in the example of figure 2, $d(\mathbf{x}_1, \Lambda_1) = 0$ and by symmetry, $d(\mathbf{x}_i, \Lambda_i) = 0$, for all i. Therefore in this example $S = 0$.

Non-separable case

Now we generalize the span concept for the non-separable case. In the non-separable case we distinguish between two categories of support vectors: the support vectors for which

$$0 < \alpha_i < C \qquad i = 1, ..., n^*$$

and the support vectors for which

$$\alpha_j = C \quad j = n^* + 1, ..., n_{sv}.$$

We define the span of support vectors using support vectors of the first category.

That means we consider the value $S_p = d(\mathbf{x}_p, \Lambda_p)$ where

$$\Lambda_p = \left\{ \sum_{i=1,\ i\neq p}^{n^*} \lambda_i \mathbf{x}_i \ :\ \sum_{i=1,\ i\neq p}^{n^*} \lambda_i = 1,\ \forall i \neq p\ \ 0 \leq \alpha_i^0 + y_i y_p \alpha_p^0 \lambda_i \leq C \right\} \tag{3.7}$$

The differences in the definition of the span for the separable and the non-separable case are that in the non-separable case we ignore the support vectors of the second category and add an upper bound C in the constraints on λ_i.

Therefore in the non-separable case the value of the span of support vectors depends on the value of C.

It is not obvious that the set Λ_p is not empty. It is proved in the following lemma.

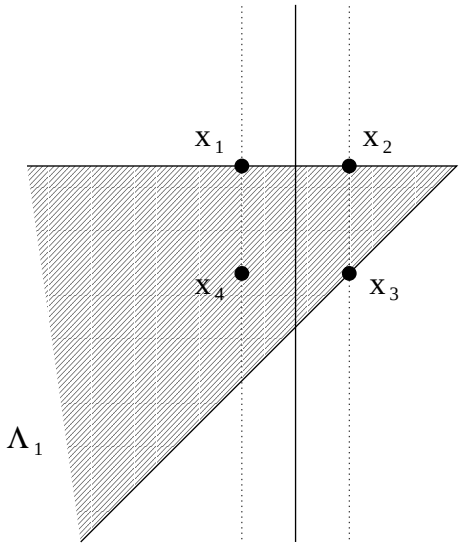


Figure 3.2: *In this example, we have* $\mathbf{x}_1 \in \Lambda_1$ *and therefore* $d(\mathbf{x}_1, \Lambda_1) = 0$. *The set* Λ_1 *has been computed using* $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4$

Lemma 3.4 *Both in the separable and non-separable case, the set* Λ_p *is not empty. Moreover* $S_p = d(\mathbf{x}_p, \Lambda_p) \leq D_{SV}$, *the diameter of the smallest sphere containing the support vectors of the first category.*

Proof in appendix.

3.2.2 The Bounds

Theorem 3.1 shows that generalization ability of SVMs is controlled by R^2/M^2, where M is the margin and R is the radius of the smallest sphere enclosing the training points. This is the theoretical idea motivating the maximization of the margin [Vapnik, 1995]. This section presents new bounds on the generalization ability of SVMs. The major improvement lies in the fact that the bounds will depend on the span of the support vectors, which gives tighter bounds that ones depending on the diameter of the training points.

Let us first introduce our fundamental result :

Lemma 3.5 *If in the leave-one-out procedure a support vector* $\mathbf{x}_p$ *corresponding to* $0 < \alpha_p < C$ *is recognized incorrectly, then the inequality*

$$\alpha_p^0 S_p \max(D, 1/\sqrt{C}) \geq 1$$

holds true.

Proof in appendix

The previous lemma leads us to the following theorem for the separable case :

Theorem 3.6 *Suppose that a SVM separates training data of size n without error. Then the number of errors made by the leave-one-out procedure is bounded by*

$$\frac{SD}{M^2}$$

Proof : Consider a support vector $\mathbf{x}_p$ incorrectly classified by the leave-one-out procedure. Then lemma 3.5 gives $\alpha_p^0 S_p D \geq 1$ (we consider here the separable case and $C = \infty$) and

$$\alpha_p^0 SD \geq 1$$

holds true. Now let us sum the left-hand and the right-hand sides of this inequality over all support vectors where the leave-one-out procedure commits an error

$$\mathcal{L}(\mathbf{x}_1, y_1, ..., \mathbf{x}_n, y_n) \leq SD \sum_{*} \alpha_i^0 \leq SD \sum_{i=1}^{n} \alpha_i^0.$$

Here $\sum_*$ indicates that the sum is taken only over support vectors where the leave-one-out procedure makes an error.

Indeed, from Kuhn-Tucker conditions (2.8) and equations (2.5) and (2.4),

$$0 = \sum_{i=1}^{n} \alpha_i^0 [y_i(\mathbf{w}_0 \cdot \mathbf{x}_i + b_0) - 1] = \mathbf{w}_0^2 - \sum_{i=1}^{n} \alpha_i^0. \tag{3.8}$$

Since the margin is $M = 1/\left\|\mathbf{w}_0\right\|$, we have

$$\mathcal{L}(\mathbf{x}_1, y_1, ..., \mathbf{x}_n, y_n) \leq \frac{SD}{M^2} \quad \square.$$

For the non-separable case the following theorem is true.

Theorem 3.7 *The number of errors made by the leave-one-out procedure is bounded by*

$$S \max(D, 1/\sqrt{C}) \sum_{i=1}^{n^*} \alpha_i^0 + m,$$

where the sum is taken only over α_i corresponding to support vectors of the first category (for which $0 < \alpha_i < C$) and m is the number of support vectors of the second category (for which $\alpha_i = C$).

Proof : The proof of this theorem is similar to the proof of theorem 3.6. We consider all support vectors of the second category (corresponding to $\alpha_j = C$) as an error. For the first category of support vectors we estimate the number $\mathcal{L}^*(\mathbf{x}_1, y_1, ..., \mathbf{x}_n, y_n)$ of errors in the leave-one-out procedure using the lemma 3.5 as in the proof of Theorem 3.6. We obtain

$$\begin{aligned} \mathcal{L}(\mathbf{x}_1, y_1, ..., \mathbf{x}_n, y_n) &\leq \mathcal{L}^*(\mathbf{x}_1, y_1, ..., \mathbf{x}_n, y_n) + m \\ &\leq S \max(D, 1/\sqrt{C}) \sum^{*} \alpha_i + m \end{aligned}$$

Note that in the case when $m = 0$ (separable case), the equality (3.8) holds true. In this case (provided that C is large enough) the bounds obtained in these two theorems coincide.

The bounds shown in the theorems are better than the ones given in the Remark. The value of span S can be much less than the diameter. Note that in Theorems 3.6 and 3.7, it is possible using inequality (3.6) to bound the value of the span S by the diameter of the smallest sphere containing the support vectors D_{SV}. But, as pointed out by the experiments (see section 3.3), this would lead to looser bounds as the span can be much less than the diameter.

The main motivation for the bounds presented in this section is conceptual. Theorem 3.6 pointed out that the number of errors made by the leave-one-out procedure is bounded by SD/M^2, which is related to the R^2/M^2 quantity, but where a new geometrical concept, the span, has been introduced.

However, from a *practical point of view, we strongly recommend not to use those bounds* which can be quite loose. Instead, the estimations presented in the following section should be used when evaluation the generalization performance of a SVM.

3.2.3 Accurate estimation of the leave-one-out error

In the proof of lemma 3.5, it appears that the diameter of the training points D can be replaced by the span of the support vectors *after* the leave-one-out procedure. But since the set of support vectors after the leave-one-out procedure is unknown, we bounded this unknown span by D. Nevertheless this remark motivated us to analyze the case where the set of support vectors remains the same during the leave-one-out procedure.

In this situation, we are allowed to replace D by S in lemma 3.5 and more precisely, the following theorem is true.

Theorem 3.8 *If the sets of support vectors of first and second categories remain the same during the leave-one-out procedure, then for any support vector* $\mathbf{x}_p$*, the following equality holds :*

$$y_p(f^0(\mathbf{x}_p) - f^p(\mathbf{x}_p)) = \alpha_p^0 S_p^2 \tag{3.9}$$

where f^0 *and* f^p *are the decision function given by the SVM trained respectively on the whole training set and after the point* $\mathbf{x}_p$ *has been removed.*

Proof in appendix

Note that under this assumption, the expression of the set Λ_p (3.7), which defines the value of $S_p = d(\mathbf{x}_p, \Lambda_p)$ can be simplified to

$$\Lambda_p = \left\{ \sum_{i \neq p,\ 0<\alpha_i^0<C} \lambda_i \Phi(\mathbf{x}_i), \sum_{i \neq p} \lambda_i = 1 \right\}. \tag{3.10}$$

Theorem 3.8 leads to several remarks.

1. The assumption that the set of support vectors does not change during the leave-one-out procedure can be wrong. Nevertheless, the proportion of points which violate this assumption is usually small compared to the number of support vectors. In this case Theorem 3.8 provides a good approximation of the result of the leave-one-out procedure, as pointed out by the experiments (see Section 3.3, figure 3.4).

2. Theorem 3.8 is stronger than lemma 3.5 for three reasons : the term $S_p \max(D, 1/\sqrt{C})$ becomes S_p^2, the inequality turns out to be an equality and the result is valid for any support vector.

3. It appears in the proof of theorem 3.8 that if one considers the expansion $\mathbf{x}^* = \sum \lambda_i \mathbf{x}_i \in \Lambda_p$ of the point minimizing the distance between $\mathbf{x}_p$ and Λ_p, then the optimal solution after having removed $\mathbf{x}_p$ becomes $\alpha_i^p = \alpha_i^0 + y_i y_p \alpha_p^0 \lambda_i$. This gives a new intuitive interpretation of the coefficients λ_i defining the span: they measure the change in the α when the point $\mathbf{x}_p$ has been removed from the training set.

Finally, the previous theorem enables us to compute the number of errors made by the leave-one-out procedure :

Corollary 3.9 *Under the assumption of Theorem 3.8, the test error prediction given by the leave-one-out procedure is*

$$t_n = \frac{1}{n}\mathcal{L}(\mathbf{x}_1, y_1, ..., \mathbf{x}_n, y_n) = \frac{1}{n} Card\{p :\ \alpha_p^0 S_p^2 \geq y_p f^0(\mathbf{x}_p)\} \tag{3.11}$$

Link with other estimates

The span estimate

$$T = \frac{1}{n} \sum_{p=1}^{n} \Psi(\alpha_p^0 S_p^2 - 1). \tag{3.12}$$

can be related to other approximations. For simplicity, we consider the hard margin case.

Link with Jaakkola-Haussler bound If we consider SVMs without threshold, the constraint $\sum \lambda_i = 1$ can be removed in the definition of the span (3.10). Indeed, this constraint comes from $\sum \alpha_i y_i = 0$, which is a consequence of the optimality condition (2.4) for the value of the threshold.

Then we can easily upper bound the value of the span: $0 \in \Lambda_p$ and $S_p^2 \leq K(\mathbf{x}_p, \mathbf{x}_p)$, and thus recover the Jaakkola-Haussler bound (3.3).

Link with R^2/M^2

For each support vector, we have $y_p f^0(\mathbf{x}_p) = 1$. Since for $x \geq 0$, $\Psi(x-1) \leq x$, the number of errors made by the leave-one-out procedure is bounded by:

$$\sum_p \alpha_p^0 S_p^2.$$

It has been shown in lemma 3.4 that the span S_p is bounded by the diameter of the smallest sphere enclosing the support vectors and since $\sum \alpha_p^0 = 1/M^2$, we finally get

$$T \leq 4 \frac{R^2}{M^2}.$$

A similar derivation as the one used in the *span bound* has been proposed in [Joachims, 2000], where the leave-one-out error is bounded by $|\{p, 2\alpha_p^0 R^2 > y_p f^0(\mathbf{x}_p)\}|$, with $0 \leq K(\mathbf{x}_i, \mathbf{x}_i) \leq R^2, \ \forall i$.

Link with Opper-Winther

When the support vectors do not change, the hard margin case without threshold gives the same value as the Opper-Winther bound, namely:

$$S_p^2 = \frac{1}{(\mathbf{K}_{SV}^{-1})_{pp}}.$$

3.2.4 Practical computation of the span estimate

As it will be shown by the experiments the span estimate (3.11) gives a very accurate estimation of the leave-one-out error. However, if performed naively, the computation of the span S_p involves a minimization and takes $O(n^3)$ operations and the overall complexity is $O(n^4)$, which is the same as the complexity of performing the leave-one-out procedure explicitly.

However, another advantage to make the assumption that the set of support vectors does not change is that the inequality constraints appearing in the definition (3.7) of Λ_p are not needed anymore yielding the simplified expression (3.10). As a consequence, it is possible to compute the span estimate (3.11) in $O(n^3)$ operations as shown below.

As before, we suppose that the training points are ordered such that $0 < alpha_i^0 < C$ for $i = 1, ..., n^*$, $\alpha_i^0 = C$, for $i = n^* + 1, ..., n_{sv}$ and $\alpha_i^0 = 0$, for $i = n_{sv} + 1, ..., n$.

The span of the support vector $\mathbf{x}_p$ is defined as the the distance between the point $\Phi(\mathbf{x}_p)$ and the set Λ_p defined by (3.10). Then the value of the span can be written as:

$$S_p^2 = \min_{\boldsymbol{\lambda}} \max_{\mu} \left(\Phi(\mathbf{x}_p) - \sum_{i \neq p}^{n^*} \lambda_i \Phi(\mathbf{x}_i) \right)^2 + 2\mu \left(\sum_{i \neq p}^{n^*} \lambda_i - 1 \right).$$

We introduced a Lagrange multiplier μ to enforce the constraint $\sum \lambda_i = 1$.

Let $\mathbf{K}^*$ the $n^* \times n^*$ matrix of dot products between support vectors of first category and $\tilde{\mathbf{K}}_{SV}$ the extended matrix

$$\tilde{\mathbf{K}}_{SV} = \begin{pmatrix} \mathbf{K}^* & \mathbf{1} \\ \mathbf{1}^\top & 0 \end{pmatrix}.$$

Finally, let $\tilde{\boldsymbol{\lambda}}$ be the extended vector $(\boldsymbol{\lambda}^\top \mu)^\top$.

Let us make the distinction on the nature of $\mathbf{x}_p$:

Support vector of first category $(0 < \alpha_p^0 < C,\ p \leq n^*)$

The value of the span can be written as:

$$S_p^2 = \min_{\boldsymbol{\lambda}} \max_{\mu} (K(\mathbf{x}_p, \mathbf{x}_p) - 2\mathbf{v}_p^\top \tilde{\boldsymbol{\lambda}} + \tilde{\boldsymbol{\lambda}}^\top \mathbf{H}_p \tilde{\boldsymbol{\lambda}}),$$

where $\mathbf{H}_p$ is the submatrix of $\tilde{\mathbf{K}}_{SV}$ with row and column p removed, and $\mathbf{v}_p$ is the p-th column of $\tilde{\mathbf{K}}_{SV}$.

From the fact that the optimal value of $\tilde{\boldsymbol{\lambda}}$ is $\mathbf{H}_p^{-1}\mathbf{v}_p$, it follows:

$$S_p^2 = K(\mathbf{x}_p, \mathbf{x}_p) - \mathbf{v}_p^\top \mathbf{H}^{-1} \mathbf{v}_p \tag{3.13}$$

$$= 1/(\tilde{\mathbf{K}}_{SV}^{-1})_{pp}. \tag{3.14}$$

The last equality comes from the following block matrix identity, known as the "Woodbury" formula [Lütkepohl, 1996]

$$\begin{pmatrix} \mathbf{A}_1 & \mathbf{A}^\top \\ \mathbf{A} & \mathbf{A}_2 \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{B}_1 & \mathbf{B}^\top \\ \mathbf{B} & \mathbf{B}_2 \end{pmatrix},$$

where $\mathbf{B}_1 = (\mathbf{A}_1 - \mathbf{A}\mathbf{A}_2^{-1}\mathbf{A}^\top)^{-1}$.

Support vector of second category $(\alpha_p^0 = C,\ n^* < p \leq n_{sv})$

A similar derivation leads to a modified version of equation (3.14),

$$S_p^2 = K(\mathbf{x}_p, \mathbf{x}_p) - \mathbf{v}_p^\top \tilde{\mathbf{K}}_{SV}^{-1} \mathbf{v}_p,$$

where $\mathbf{v}_p$ is a $n^* + 1$-dimensional vector whose i-th element is $K(\mathbf{x}_p, \mathbf{x}_i)$ and last element is 1.

Non support vector $(\alpha_p^0 = 0,\ p > n_{sv})$

In this case, it is well known that $f^p(\mathbf{x}_p) = f^0(\mathbf{x}_p)$. This also a direct consequence of theorem 3.8 since $\alpha_p^0 = 0$. It is not necessary to compute the span of such a point since the leave-one-out procedure will not make an error on this point.

The closed form we obtain is particularly attractive since we can compute the value of the span for each support vector just by inverting the matrix $\tilde{\mathbf{K}}_{SV}$.

Some Matlab code to compute the span estimate (3.11) is given in appendix.

3.3 Experiments

The previous bounds on the generalization ability of Support Vector Machines involved the diameter of the smallest sphere enclosing the training points [Vapnik, 1995]. We have shown (cf inequality (3.6)) that the span S is always smaller than this diameter, but to appreciate the gain, we conducted some experiments.

First we compare the diameter of the smallest sphere enclosing the training points, the one enclosing the support vectors and the span of the support vectors using the postal database. This dataset consists of 7291 handwritten digits of size 16x16 with a test set of 2007 examples. Following [Schölkopf et al.,

1999], we split the training set in 23 subsets of 317 training examples. Our task is to separate digits 0 to 4 from 5 to 9. Error bars in figure 3.3 are standard deviations over the 23 trials. The diameters and the span in figure 3.3 are plotted for different values of σ, the width of the RBF kernel we used :

$$K(\mathbf{x}, \mathbf{y}) = e^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}}.$$

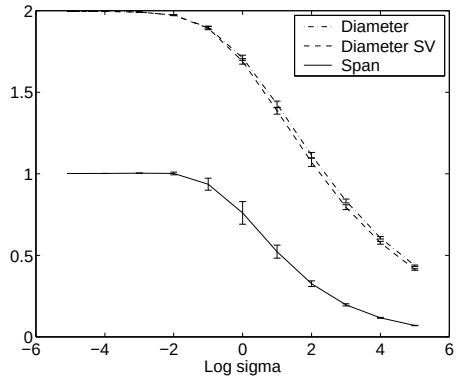


Figure 3.3: *Comparison of D, D_{SV} and S*

In this example, the span is up to 6 times smaller than the diameter.

Now we would like to use the span for predicting accurately the test error. This would enable us to perform efficient model selection, i.e. choosing the optimal values of parameters in SVMs (the width of the RBF kernel σ or the constant C, for instance).

Note that the span S is defined as a maximum $S = \max_p S_p$ and therefore taking into account the different values S_p should provide a more accurate estimation of the generalization error than the span S only. Therefore, we used the *span-rule* (3.11) to predict the test error.

Our experiments have been carried out on two databases : a separable one, the postal database, described above and a noisy one, the breast-cancer database [4]. The latter has been split randomly 100 times into a training set containing 200 examples and a test set containing 77 examples.

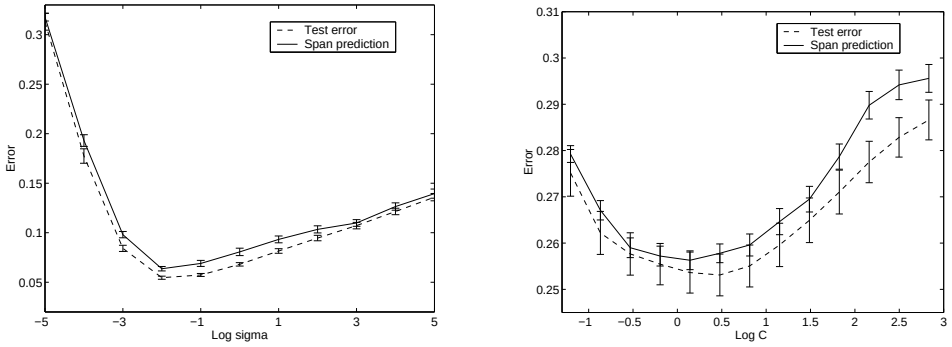
Figure 3.4a compares the test error and its prediction given by the span-rule (3.11) for different values of the width σ of the RBF kernel on the postal database. Figure 3.4b plots the same functions for different values of C on the breast-cancer database. For this latter, a RBF kernel with $\sigma^2 = d = 9$, the number of input dimensions, was used.

The prediction is very accurate and the curves are almost identical.

The span-rule has been computed efficiently as explained in section 3.2.4. The corresponding Matlab code can be found in appendix. Note that the complexity to compute the span estimate is $O(N_{SV}^3)$, where N_{SV} is the number of support vectors, which is the same complexity as a standard SVM training.

3.3.1 Other estimates

On the same database (USPS), we checked the behavior of two other bounds: the Jaakkola-Haussler one (3.3) and R^2/M^2. Results are shown in figure 3.5. Those bounds are much less accurate than the span estimate (see figure 3.4), especially for large values of σ. Nevertheless, they exhibit a satisfying value of the minimum.



(a) choice of σ in the postal database (b) choice of C in the breast-cancer database

Figure 3.4: *Test error and its prediction using the span-rule (3.11).*

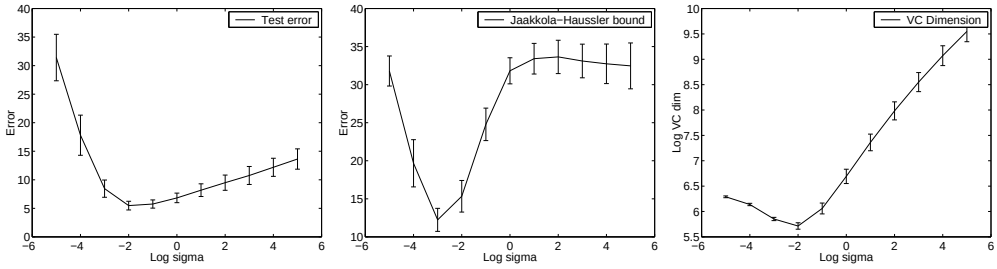


Figure 3.5: Postal database, RBF kernel with different values of σ. Left: test error, middle: Jaakkola-Haussler bound, right: R^2/M^2.

3.4 Appendix

3.4.1 Proof of lemma 3.4

We will prove this result for the non-separable case. The result is also valid for the separable case since it can be seen as a particular case of the non-separable one with C large enough.

Let us define Λ_p^+ as the subset of Λ_p with additional constraints $\lambda_i \geq 0$:

$$\Lambda_p^+ = \left\{ \sum_{i=1,\ i\neq p}^{n} \lambda_i \mathbf{x}_i \in \Lambda_p \ : \ \lambda_i \geq 0,\ i \neq p \right\}. \tag{3.15}$$

We shall prove that $\Lambda_p^+ \neq \emptyset$ by proving that a vector λ of the following form exists :

$$\lambda_j = 0, \quad j = n^* + 1, ..., n \tag{3.16}$$

$$\lambda_i = \mu \frac{C - \alpha_i^0}{\alpha_p^0}, \quad y_i = y_p, \quad i \neq p, \quad i = 1, ..., n^* \tag{3.17}$$

$$\lambda_i = \mu \frac{\alpha_i^0}{\alpha_p^0}, \quad y_i \neq y_p, \quad i = 1, ..., n^* \tag{3.18}$$

$$0 \leq \mu \leq 1 \tag{3.19}$$

[4] Available from `http://horn.first.gmd.de/~raetsch/data/breast-cancer`

$$\sum_{i=1}^{n} \lambda_i = 1 \tag{3.20}$$

It is straightforward to check that if such a vector λ exists, then $\sum \lambda_i \mathbf{x}_i \in \Lambda_p^+$ and therefore $\Lambda_p^+ \neq \emptyset$. Since $\Lambda_p^+ \subset \Lambda_p$, we will have $\Lambda \neq \emptyset$.

Taking into account equations (3.17) and (3.18), we can rewrite constraint (3.20) as follows :

$$1 = \frac{\mu}{\alpha_p^0} \left(\sum_{\substack{i=1,\ i\neq p \\ y_i = y_p}}^{n^*} (C - \alpha_i^0) + \sum_{\substack{i=1 \\ y_i \neq y_p}}^{n^*} \alpha_i^0 \right) \tag{3.21}$$

We need to show that the value of μ given by equation (3.21) satisfies constraint (3.19).

For this purpose, let us define Δ as :

$$\Delta = \sum_{i/\ y_i = y_p}^{n^*} (C - \alpha_i^0) + \sum_{i/\ y_i \neq y_p}^{n^*} \alpha_i^0 \tag{3.22}$$

$$= -y_p \sum_{i=1}^{n^*} y_i \alpha_i^0 + \sum_{i/\ y_i = y_p}^{n^*} C \tag{3.23}$$

Now, note that

$$\sum_{i=1}^{n} y_i \alpha_i^0 = \sum_{i=1}^{n^*} y_i \alpha_i^0 + C \sum_{i=n^*+1}^{n} y_i = 0. \tag{3.24}$$

Combining equations (3.23) and (3.24) we get

$$\begin{aligned} \Delta &= C y_p \sum_{i=n^*+1}^{n} y_i + \sum_{i/\ y_i = y_p}^{n^*} C \\ &= Ck, \end{aligned}$$

where k is an integer.

Since equation (3.22) gives $\Delta > 0$, we have finally

$$\Delta \geq C. \tag{3.25}$$

Let us rewrite equation (3.21) as :

$$1 = \frac{\mu}{\alpha_p^0}(\Delta - (C - \alpha_p^0)).$$

We obtain

$$\mu = \frac{\alpha_p^0}{\Delta - (C - \alpha_p^0)}. \tag{3.26}$$

Taking into account inequality (3.25), we finally get $0 \leq \mu \leq 1$. Thus, constraint (3.19) is fulfilled and Λ_p^+ is not empty.

Now note that the set Λ_p^+ is included in the convex hull of $\{\mathbf{x}_i\}_{i\neq p}$ and since $\Lambda_p^+ \neq \emptyset$, we obtain

$$d(\mathbf{x}_p, \Lambda_p^+) \leq D_{SV},$$

where D_{SV} is the diameter of the smallest ball containing the support vectors of the first category.

Since $\Lambda_p^+ \subset \Lambda_p$ we finally get

$$S_p = d(\mathbf{x}_p, \Lambda_p) \leq d(\mathbf{x}_p, \Lambda_p^+) \leq D_{SV}.$$

3.4.2 Proof of lemma 3.5

Let us first consider the separable case.

Suppose that our training set $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ is ordered such that the support vectors are the first n_{sv} training points. The non-zero Lagrange multipliers associated with these support vectors are

$$\alpha_1^0, ..., \alpha_{n_sv}^0 \tag{3.27}$$

In other words, the vector $\boldsymbol{\alpha}^0 = (\alpha_1^0, \ldots, \alpha_{n_sv}^0, 0, \ldots, 0)$ maximizes the functional

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{3.28}$$

subject to the constraints

$$\boldsymbol{\alpha} \geq 0, \tag{3.29}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0. \tag{3.30}$$

Let us consider the result of the leave-one-out procedure on the support vector $\mathbf{x}_p$. This means that we maximized functional (3.28) subject to the constraints (3.29), (3.30) and the additional constraint

$$\alpha_p = 0, \tag{3.31}$$

and obtained the solution

$$\boldsymbol{\alpha}^p = (\alpha_1^p, ..., \alpha_n^p).$$

Using this solution we construct the separating hyperplane

$$\mathbf{w}_p \cdot \mathbf{x} + b_p = 0,$$

where

$$\mathbf{w}_p = \sum_{i=1}^{n} \alpha_i^p y_i \mathbf{x}_i.$$

We would like to prove that if this hyperplane classifies the vector $\mathbf{x}_p$ incorrectly:

$$y_p(\mathbf{w}_p \cdot \mathbf{x}_p + b_p) < 0 \tag{3.32}$$

then

$$\alpha_p^0 \geq \frac{1}{S_p D}.$$

Since $\boldsymbol{\alpha}^p$ maximizes (3.28) under constraints (3.29), (3.30) and (3.31), the following inequality holds true

$$W(\boldsymbol{\alpha}^p) \geq W(\boldsymbol{\alpha}^0 - \boldsymbol{\delta}), \tag{3.33}$$

where the vector $\boldsymbol{\delta} = (\delta_1, ..., \delta_n)$ satisfies the following conditions

$$\delta_p = \alpha_p^0,$$

$$\boldsymbol{\alpha}_0 - \boldsymbol{\delta} \geq 0,$$

$$\sum_{i=1}^{n} \delta_i y_i = 0.$$

$$\delta_i = 0, \quad i > n_{sv} \tag{3.34}$$

From inequality (3.33) we obtain

$$W(\boldsymbol{\alpha}^0) - W(\boldsymbol{\alpha}^p) \leq W(\boldsymbol{\alpha}^0) - W(\boldsymbol{\alpha}^0 - \boldsymbol{\delta}). \tag{3.35}$$

Since $\boldsymbol{\alpha}^0$ maximizes (3.28) under the constraints (3.29) and (3.30), the following inequality holds true

$$W(\boldsymbol{\alpha}^0) \geq W(\boldsymbol{\alpha}^p + \boldsymbol{\gamma}), \tag{3.36}$$

where $\boldsymbol{\gamma} = (\gamma_1, ..., \gamma_n)$ is a vector satisfying the constraints

$$\boldsymbol{\alpha}_p + \boldsymbol{\gamma} \geq 0,$$

$$\sum_{i=1}^{n} \gamma_i y_i = 0.$$

$$\alpha_i^p = 0 \Longrightarrow \gamma_i = 0, \quad i \neq p \tag{3.37}$$

From (3.35) and (3.36), we have

$$W(\boldsymbol{\alpha}^p + \boldsymbol{\gamma}) - W(\boldsymbol{\alpha}^p) \leq W(\boldsymbol{\alpha}^0) - W(\boldsymbol{\alpha}^0 - \boldsymbol{\delta}) \tag{3.38}$$

Let us calculate both the left-hand side, I_1, and the right-hand side, I_2 of inequality (3.38).

$$\begin{aligned}
I_1 &= W(\boldsymbol{\alpha}^p + \boldsymbol{\gamma}) - W(\boldsymbol{\alpha}^p) \\
&= \sum_{i=1}^{n}(\alpha_i^p + \gamma_i) - \frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i^p + \gamma_i)(\alpha_j^p + \gamma_j) y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\
&\quad - \sum_{i=1}^{n}\alpha_i^p + \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i^p \alpha_j^p y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\
&= \sum_{i=1}^{n}\gamma_i - \sum_{i,j}^{n}\gamma_i \alpha_j^p y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j - \frac{1}{2}\sum_{i,j}^{n} y_i y_j \gamma_i \gamma_j \mathbf{x}_i \cdot \mathbf{x}_j \\
&= \sum_{i=1}^{n}\gamma_i(1 - y_i \mathbf{w}_p \cdot \mathbf{x}_i) - \frac{1}{2}\sum_{i,j}^{n} y_i y_j \gamma_i \gamma_j (\mathbf{x}_i, \mathbf{x}_j)
\end{aligned}$$

Taking into account that

$$\sum_{i=1}^{n}\gamma_i y_i = 0$$

we can rewrite expression

$$I_1 = \sum_{i \neq p}^{n}\gamma_i[1 - y_i(\mathbf{w}_p \cdot \mathbf{x}_i + b_p)] + \gamma_p[1 - y_p(\mathbf{w}_p \cdot \mathbf{x}_p + b_p)] - \frac{1}{2}\sum_{i,j}^{n} y_i y_j \gamma_i \gamma_j \mathbf{x}_i \cdot \mathbf{x}_j.$$

Since for $i \neq p$ the condition (3.37) means that either $\gamma_i = 0$ or $\mathbf{x}_i$ is a support vector of the hyperplane $\mathbf{w}_p$, the following equality holds

$$\gamma_i[y_i(\mathbf{w}_p \cdot \mathbf{x}_i + b_p) - 1] = 0.$$

We obtain

$$I_1 = \gamma_p[1 - y_p(\mathbf{w}_p \cdot \mathbf{x}_p + b_p)] - \frac{1}{2}\sum_{i,j}^{n} y_i y_j \gamma_i \gamma_j \mathbf{x}_i \cdot \mathbf{x}_j. \tag{3.39}$$

Now let us define vector γ as follows:

$$\gamma_p = \gamma_k = a,$$

$$\gamma_i = 0 \quad i \notin \{k, p\},$$

where a is some constant and k is such that $y_p \neq y_k$ and $\alpha_k^p > 0$. For this vector we obtain

$$\begin{aligned} I_1 &= a[1 - y_p(\mathbf{w}_p \cdot \mathbf{x}_p + b_p)] - \frac{a^2}{2} \|\mathbf{x}_p - \mathbf{x}_k\|^2 \\ &\geq a[1 - y_p(\mathbf{w}_p \cdot \mathbf{x}_p + b_p)] - \frac{a^2}{2} D^2. \end{aligned} \tag{3.40}$$

Let us choose the value a to maximize this expression

$$a = \frac{1 - y_p(\mathbf{w}_p \cdot \mathbf{x}_p + b_p)}{D^2}.$$

Putting this expression back into (3.40) we obtain

$$I_1 \geq \frac{1}{2} \frac{(1 - y_p[(\mathbf{x}_p, \mathbf{w}_p) + b_p])^2}{D^2}.$$

Since, according to our assumption, the leave-one-out procedure commits an error at the point $\mathbf{x}_p$ (that is, the inequality (3.32) is valid), we obtain

$$I_1 \geq \frac{1}{2D^2}. \tag{3.41}$$

Now we estimate the right-hand side of inequality (3.38)

$$I_2 = W(\boldsymbol{\alpha}^0) - W(\boldsymbol{\alpha}^0 - \boldsymbol{\delta}).$$

We choose $\delta_i = -y_i y_p \alpha_p^0 \lambda_i$, where λ is the vector that defines the value of $d(\mathbf{x}_p, \Lambda_p)$ in equation (3.5).

We have

$$\begin{aligned} I_2 &= W(\boldsymbol{\alpha}^0) - W(\boldsymbol{\alpha}^0 - \boldsymbol{\delta}) \\ &= \sum_{i=1}^n \alpha_i^0 - \frac{1}{2} \sum_{i,j=1}^n \alpha_i^0 \alpha_j^0 y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j - \sum_{i=1}^n (\alpha_i^0 + y_i y_p \alpha_p^0 \lambda_i) \\ &\quad + \frac{1}{2} \sum_{i,j=1}^n (\alpha_i^0 + y_i y_p \alpha_p^0 \lambda_i)(\alpha_j^0 + y_j y_p \alpha_p^0 \lambda_j) y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\ &= -y_p \alpha_p^0 \sum_{i=1}^n y_i \lambda_i + y_p \alpha_p^0 \sum_{i,j=1}^n \alpha_i^0 \lambda_j y_i \mathbf{x}_i \cdot \mathbf{x}_j + \frac{1}{2} (\alpha_p^0)^2 (\sum_{i=1}^n \lambda_i \mathbf{x}_i)^2. \end{aligned} \tag{3.42}$$

Since

$$\sum_{i=1}^n \lambda_i = 0,$$

and $\mathbf{x}_i$ is a support vector, we have

$$I_2 = y_p \alpha_p^0 \sum_{i=1}^n \lambda_i y_i [y_i(\mathbf{w}_0 \cdot \mathbf{x}_i + b_0) - 1] + \frac{(\alpha_p^0)^2}{2} (\sum_{i=1}^n \lambda_i \mathbf{x}_i)^2 = \frac{(\alpha_p^0)^2}{2} S_p^2. \tag{3.43}$$

Combining (3.38), (3.41) and (3.43) we obtain

$$\alpha_p^0 S_p D \geq 1.$$

Non-separable case

The sketch of the proof is the same. The main difference lies in the choice of the vector $\boldsymbol{\gamma}$. For this purpose, we need a lemma very similar to lemma 3.4.

Lemma 3.10 *For any value a between 0 and C, there exists a vector* $\boldsymbol{\gamma}$ *satisfying*

$$0 \leq \boldsymbol{\alpha}^p + \boldsymbol{\gamma} \leq C \tag{3.44}$$

$$\gamma_p = a \tag{3.45}$$

$$\sum \gamma_i y_i = 0 \tag{3.46}$$

$$y_p y_i \gamma_i \leq 0 \quad \forall i \neq p. \tag{3.47}$$

$$\alpha_i^p \in \{0, C\} \Longrightarrow \gamma_i = 0, \quad i \neq p \tag{3.48}$$

Proof: Without loss of generality, we can reorder the training examples such that the first n^* ones are margin support vectors ($0 < \alpha_i^p < C$), $\alpha_i^p = C$ for $n^* \leq i \leq n_{sv}$ and $\alpha_i^p = 0$ for $i > n_{sv}$. Let us prove that there exists $\mu \in [0, 1]$ such that the vector $\boldsymbol{\gamma}$ defined as below is valid :

$$\gamma_i = -\mu \alpha_i^p, \quad \text{if } y_i = y_p, \quad 1 \leq i \leq n^* \tag{3.49}$$

$$\gamma_i = \mu(C - \alpha_i^p), \quad \text{if } y_i \neq y_p, \quad 1 \leq i \leq n^* \tag{3.50}$$

If $\mu \in [0, 1]$ then the vector $\boldsymbol{\gamma}$ defined by equations (3.49), (3.50) and (3.45) satisfies constraints (3.44) and (3.47). Taking into account equation (3.49) and (3.50), let us rewrite constraint (3.46) as follows:

$$a y_p - \mu \sum_{i/\ y_i = y_p}^{n^*} y_i \alpha_i^p + \mu \sum_{i/\ y_i \neq y_p}^{n^*} y_i (C - \alpha_i^p) = 0$$

$$a - \mu \Delta = 0 \tag{3.51}$$

where

$$\Delta = \sum_{i/\ y_i = y_p}^{n^*} \alpha_i^p + \sum_{i/\ y_i \neq y_p}^{n^*} (C - \alpha_i^p) \tag{3.52}$$

$$= y_p \sum_{i=1}^{n^*} y_i \alpha_i^p + \sum_{i/\ y_i \neq y_p}^{n^*} C \tag{3.53}$$

Now, note that

$$\sum_{i=1}^{n_{sv}} y_i \alpha_i^p = \sum_{i=1}^{n^*} y_i \alpha_i^p + C \sum_{i=n^*+1}^{n_{sv}} y_i = 0 \tag{3.54}$$

Combining equations (3.53) and (3.54), we get

$$\Delta = -C y_p \sum_{i=n^*+1}^{n_{sv}} y_i + \sum_{i/\ y_i \neq y_p}^{n^*} C$$

$$= Ck,$$

where k is an integer. Since equation (3.53) gives $\Delta > 0$, we finally have

$$\Delta \geq C.$$

Combining this last result with (3.51), we finally get

$$\mu = \frac{a}{\Delta} \leq 1. \qquad \square$$

The previous lemma tells us that for any value of a there exists a vector $\boldsymbol{\gamma}$ satisfying certain conditions. The value of a is not chosen yet, but note that $\boldsymbol{\gamma}$ depends on a. Now let us rewrite (3.39)

$$\forall a \in [0, C], \quad I_1 = a[1 - y_p(\mathbf{w}_p \cdot \mathbf{x}_p + b_p)] - \frac{a^2}{2} \left\| y_p \mathbf{x}_p + \sum_{i \neq p}^{n^*} \frac{\gamma_i}{a} y_i \mathbf{x}_i \right\|^2$$

$$\forall a \in [0, C], \quad I_1 = a[1 - y_p(\mathbf{w}_p \cdot \mathbf{x}_p + b_p)] - \frac{a^2}{2} \left\| \mathbf{x}_p - \sum_{i \neq p}^{n^*} \lambda_i \mathbf{x}_i \right\|^2$$

with $\lambda_i = -y_i y_p \gamma_i / a$. From equation (3.47), $\lambda_i \geq 0$ and from (3.46), $\sum_{i \neq p} \lambda_i = 1$. We can conclude that $\sum_{i \neq p}^{n^*} \lambda_i \mathbf{x}_i$ is in the convex hull of the training points $\mathbf{x}_i$ and (3.40) follows.

Another difference with the separable case lies in the choice of a in equation (3.40). The value of a which maximizes equation (3.40) is

$$a^* = \frac{1 - y_p(\mathbf{w}_p \cdot \mathbf{x}_p + b_p)}{D^2}.$$

But we need to fulfill the condition $a \leq C$. Thus, if $a^* > C$, we replace a by C in equation (3.40) and we get :

$$\begin{aligned} I_1 &\geq C[1 - y_p(\mathbf{w}_p \cdot \mathbf{x}_p + b_p)] - \frac{C^2}{2} D^2 \\ &= CD^2(a^* - \frac{C}{2}) \\ &\geq CD^2 \frac{a^*}{2} \\ &= \frac{C}{2}[1 - y_p(\mathbf{w}_p \cdot \mathbf{x}_p + b_p)] \\ &\geq \frac{C}{2} \end{aligned}$$

The last inequality comes from (3.32).
Finally, we have

$$I_1 \geq \frac{1}{2} \min\left(C, \frac{1}{D^2}\right).$$

By combining this last inequality with (3.38) and (3.43) we prove the lemma.

3.4.3 Proof of theorem 3.8

The proof follows the proof of lemma 3.5. Under the assumption that the set of support vectors remains the same during the leave-one-out procedure, we can take

$$\boldsymbol{\delta} = \boldsymbol{\gamma} = \boldsymbol{\alpha}^0 - \boldsymbol{\alpha}^p$$

as $\boldsymbol{\alpha}^0 - \boldsymbol{\alpha}^P$ is a vector satisfying simultaneously the set of constraints (3.34) and (3.37).
Then inequality (3.38) becomes an equality :

$$I_1 = W(\boldsymbol{\alpha}^0) - W(\boldsymbol{\alpha}^p) = I_2 \tag{3.55}$$

From inequality (3.35), it follows that

$$I_2 \leq I_2^* = W(\boldsymbol{\alpha}^0) - W(\boldsymbol{\alpha}^0 - \boldsymbol{\delta}^*), \tag{3.56}$$

where $\delta_i^* = -y_i y_p \alpha_p^0 \lambda_i$ and λ is given by the definition of the span S_p (cf equation (3.5)).

The computation of I_2 and I_2^* is similar to the one involved in the proof of lemma 3.5 (cf equation (3.43))

$$I_2^* = \frac{(\alpha_p^0)^2}{2} S_p^2 - \alpha_p^0 [y_p(\mathbf{w}_0 \cdot \mathbf{x}_p + b_0) - 1]$$

$$I_2 = \frac{(\alpha_p^0)^2}{2} (\sum_i \lambda_i^* \mathbf{x}_i)^2 - \alpha_p^0 [y_p(\mathbf{w}_0 \cdot \mathbf{x}_p + b_0) - 1],$$

where

$$\lambda_i^* = y_i \frac{\alpha_i^p - \alpha_i^0}{\alpha_p^0}$$

From (3.56), we get $(\sum_i \lambda_i^* \mathbf{x}_i)^2 \leq S_p^2$.

Now note that $\sum_{i \neq p} \lambda_i^* \mathbf{x}_i \in \Lambda_p$ and by definition of S_p, $(\sum_i \lambda_i^* \mathbf{x}_i)^2 \geq S_p^2$. Finally, we have

$$(\sum_i \lambda_i^* \mathbf{x}_i)^2 = S_p^2. \tag{3.57}$$

The computation of I_1 gives (cf equation (3.40))

$$I_1 = \alpha_p^0 [1 - y_p(\mathbf{w}_p \cdot \mathbf{x}_p + b_p)] - \frac{(\alpha_p^0)^2}{2} (\sum_i \lambda_i^* \mathbf{x}_i)^2$$

Putting the values of I_1 and I_2 back in equation (3.55), we get

$$(\alpha_p^0)^2 (\sum_i \lambda_i^* \mathbf{x}_i)^2 = \alpha_p^0 y_p [f^0(\mathbf{x}_p) - f^p(\mathbf{x}_p)]$$

and the theorem is proved by dividing by α_p^0 and taking into account equation (3.57) :

$$\alpha_p^0 S_p^2 = y_p [f^0(\mathbf{x}_p) - f^p(\mathbf{x}_p)].$$

3.4.4 Matlab code

We give some Matlab code to compute the span estimate (3.11). It follows the derivation given in section 3.2.4.

```
function loo = fast_span_estimate(K,Y,alpha,b,C)
%
% Compute an estimate of the number of errors made by leave-one-out
% procedure for an SVM.
% It only requires a matrix inversion and hence the complexity is not more
% than the SVM training itself.
%
%   K : kernel matrix
%   Y : labels (-1,1)
%   alpha,b : lagrange multipliers and threshold
%   C : soft margin paramater, aka upper bound on alpha.
%
%   loo = estimate of the fraction of leave-one-out errors.

% Compute the outputs on the training points
```

```
output = Y.*(K*(alpha.*Y)+b);

% Find the indices of the support vectors of first and second category
eps = 1e-5;
sv1 = find(alpha > max(alpha)*eps & alpha < C*(1-eps));
sv2 = find(alpha > C*(1-eps));


% Degenerate case: if sv1 is empty, then we assume nothing changes
% (loo = training error)
if isempty(sv1)
  loo = mean(output < 0);
  return;
end;

% Compute the invert of KSV
l = length(sv1);
KSV = [[K(sv1,sv1) ones(l,1)]; [ones(1,l) 0]];
% a small ridge is added to be sure that the matrix is invertible
invKSV=inv(KSV+diag(1e-12*[ones(1,l) 0]));

% Compute the span for all support vectors.
n = length(K);     % Number of training points
span = zeros(n,1); % Initialize the vector
tmp = diag(invKSV);
span(sv1) = 1./tmp(1:l); % Span of sv of first category
% If there exists sv of second category, compute their span
if ~isempty(sv2)
  V = [K(sv1,sv2); ones(1,length(sv2))];
  span(sv2) = diag(K(sv2,sv2)) - diag(V'*invKSV*V);
end;

% Estimate the fraction of loo error
loo = mean(output - alpha.*span < 0);
```

Chapter 4

Choosing the kernel parameters

4.1 Quadratic loss

In this chapter, we will show how to choose automatically the parameters of an SVM kernel. We will consider soft margin SVMs with quadratic penalization of errors which can be considered as a special case of the hard margin version with the modified kernel [Cortes and Vapnik, 1995; Friess and Harrison, 1998]

$$\mathbf{K} \leftarrow \mathbf{K} + \frac{1}{C}\mathbf{I}, \tag{4.1}$$

where $\mathbf{I}$ is the identity matrix and C a constant penalizing the training errors. In the rest of this chapter, we will focus on the hard margin SVM and use (4.1) whenever we have to deal with non-separable data. Thus C will be considered just as another parameter of the kernel function.

This ridge trick can be understood intuitively as follows [Freund and Schapire, 1998]: n additional virtual components are added in the feature space and the modified mapping is defined as:

$$\begin{array}{lccccccc} \tilde{\Phi}(\mathbf{x}_i)^\top = (\Phi(\mathbf{x}_i) & 0 & \dots & y_i/\sqrt{C} & \dots & 0). & & \\ & 1 & & i & & n & \text{position} & \end{array}$$

Introducing

$$\tilde{\mathbf{w}}^\top = (\mathbf{w} \;\; \xi_1\sqrt{C} \dots \xi_n\sqrt{C}),$$

we have

$$\tilde{\mathbf{w}}^2 = \mathbf{w}^2 + C\sum_{i=1}^{n}\xi_i^2,$$

$$y_i(\tilde{\mathbf{w}} \cdot \tilde{\Phi}(\mathbf{x}_i) + b) \geq 1 \iff y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i) + b) \geq 1 - \xi_i.$$

As a conclusion training a hard margin SVM in the modified feature space is equivalent to training a soft margin SVM with quadratic penalization of the slacks. The new kernel is

$$\tilde{\mathbf{K}}_{ij} = \tilde{\Phi}(\mathbf{x}_i) \cdot \tilde{\Phi}(\mathbf{x}_j) = \mathbf{K}_{ij} + \delta_{ij}/C.$$

Note also that $\tilde{\mathbf{w}} = \sum \alpha_i^0 y_i \tilde{\Phi}(\mathbf{x}_i)$ implies

$$\xi_i C = \alpha_i^0. \tag{4.2}$$

4.2 Smoothing the test error estimates

The estimate of the performance of an SVM through a validation error (3.1) or the leave-one-out error (3.2) requires the use of the step function Ψ. However, we would like to use a gradient descent approach to minimize those estimates of the test error. Unfortunately the step function is not differentiable. As

already mentioned in section 3.2.3.0, it is possible to bound $\Psi(x-1)$ by x for $x \geq 0$. This is how the bound R^2/M^2 is derived from the leave-one-out error. Nevertheless by doing so, large errors count more than one, therefore it might be advantageous instead to use a contracting function of the form $\Psi(x) = (1+\exp(-Ax+B))^{-1}$ (see figure 4.1).

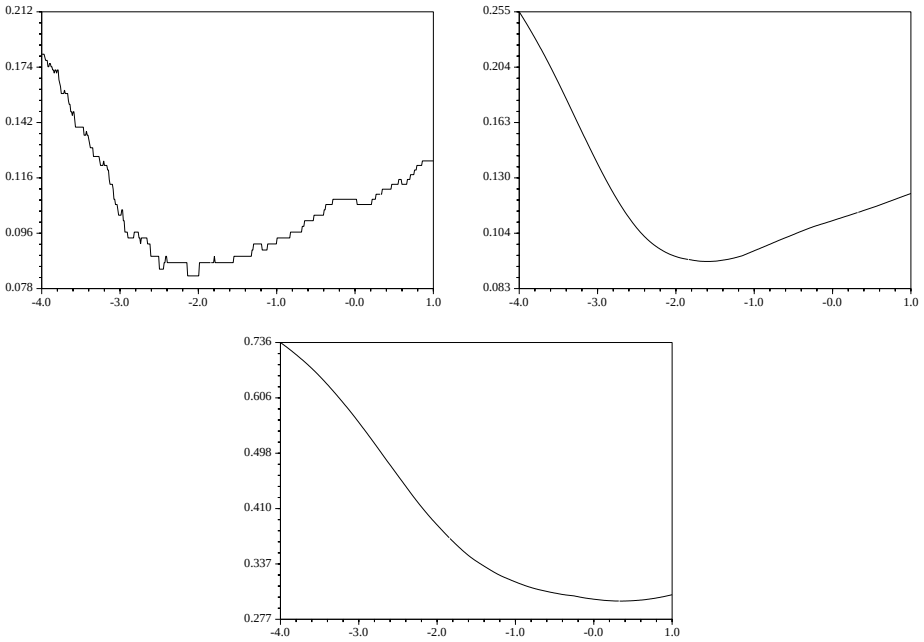


Figure 4.1: Validation error for different values of the width (in log scale) of an RBF kernel. Top left: with a step function, $\Psi(x) = 1_{x>0}$. Top right: sigmoid function, $\Psi(x) = (1+\exp(-5x))^{-1}$. Bottom: linear function, $\Psi(x) = 1+x$ for $x > -1$, 0 otherwise. Note that on the bottom picture, the minimum is not at the right place

However, the choice of the constants A and B is difficult. If A is too small, the estimate is not accurate and A is too large, the resulting estimate is not smooth.

Instead of trying to pick good constants A and B, one can try to get directly a smooth approximation of the test error by estimating posterior probabilities. Recently, Platt proposed the following estimate of the posterior distribution $P(Y=1|X=\mathbf{x})$ of an SVM output $f(\mathbf{x})$ [Platt, 2000]:

$$\tilde{P}_{A,B}(\mathbf{x}) = \tilde{P}(Y=1|X=\mathbf{x}) = \frac{1}{1+\exp(Af(\mathbf{x})+B)},$$

where $f(\mathbf{x})$ is the output of the SVM. The constants A and B are found by minimizing the Kullback-Liebler divergence between $\tilde{P}$ and an empirical estimation of P built from a validation set $(\mathbf{x}'_i, y'_i)_{1\leq i\leq n_v}$:

$$(A^*, B^*) = \arg\max_{A,B} \sum_{i=1}^{n_v} \left(\frac{1+y'_i}{2} \log(\tilde{P}_{A,B}(\mathbf{x}'_i)) + \frac{1-y'_i}{2} \log(1-\tilde{P}_{A,B}(\mathbf{x}'_i)) \right). \tag{4.3}$$

This optimization is carried out using a second order gradient descent algorithm [Platt, 2000].

According to this estimate the best threshold for our SVM classifier f is such that

$$f(\mathbf{x}) = \operatorname{sgn}(\tilde{P}_{A^*,B^*}(x) - 0.5).$$

Note that if $B^* \neq 0$, we obtained a correction compared to the usual SVM threshold.

By definition the generalization error of our classifier is

$$P(Y \neq f(X)) = \int_{\mathbf{x}, f(\mathbf{x})=-1} P(Y=1|\mathbf{x}) d\mu(\mathbf{x}) + \int_{\mathbf{x}, f(\mathbf{x})=1} (Y=-1|\mathbf{x}) d\mu(\mathbf{x}).$$

This error can be empirically estimated as [5]:

$$P(Y \neq f(X)) \approx \sum_{i,\tilde{P}(\mathbf{x}'_i)<0.5} \tilde{P}(\mathbf{x}'_i) + \sum_{i,\tilde{P}(\mathbf{x}'_i)>0.5} 1 - \tilde{P}(\mathbf{x}'_i) \tag{4.4}$$

$$= \sum_{i=1}^{n_v} \min\left(\tilde{P}(\mathbf{x}'_i), 1 - \tilde{P}(\mathbf{x}'_i)\right). \tag{4.5}$$

Note that the labels of the validation set are not used directly in this last step but indirectly through the estimation of the constants A and B appearing in the parametric form of $\tilde{P}_{A^*,B^*}$. To have a better understanding of this estimate, let us consider the extreme case where there is no error on the validation set. Then the maximum likelihood algorithm will yield $A = -\infty$ and $\tilde{P}_{A^*,B^*}(\mathbf{x})$ will only take binary values. As a consequence, the estimate of the error probability will be zero.

If no validation set is available, one could use the training set to find the values A^* and B^* minimizing the functional (4.3), but this would amount to estimating a smoothed training error. The satisfying way to estimate a smoothed generalization error is to use the leave-one-out procedure: find A^* and B^* which minimize (4.3) with $\mathbf{x}'_p = \mathbf{x}_p,\ y'_p = y_p$ and the modified version of the posterior probability

$$\tilde{P}_{A,B}(\mathbf{x}_p) = \frac{1}{1 + \exp(Af^p(\mathbf{x}_p) + B)},$$

where f^p is the function learned when the point $\mathbf{x}_p$ has been removed from the training set. $f^p(\mathbf{x}_p)$ can be estimated accurately using equation (3.11) as

$$f^p(\mathbf{x}_p) \approx f^0(\mathbf{x}_p) - y_p \alpha_p^0 S_p^2.$$

A quite different approach to estimate those posterior probabilities would be to use the Parzen window density estimator (1.13) introduced to compute the vicinal risk. This would lead to the minimization of the following integral

$$\int \frac{1+y}{2} \log(\tilde{P}_{A,B}(\mathbf{x})) + \frac{1-y}{2} \log(1 - \tilde{P}_{A,B}(\mathbf{x}))\ dP_{est}(\mathbf{x}, y).$$

The drawback of this approach is that the choice of the width of the Parzen window estimator is crucial and one would have to choose it by some validation or leave-one-out method.

4.3 Optimizing the kernel parameters

Let's go back to the SVM algorithm. We assume that the kernel K depends on one or several parameters, encoded into a vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)$. We thus consider a class of decision functions parametrized by $\boldsymbol{\alpha}$, b and $\boldsymbol{\theta}$:

$$f_{\boldsymbol{\alpha},b,\boldsymbol{\theta}}(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{x}_i) + b\right).$$

We want to choose the values of the parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$ such that W (see equation (2.6)) is maximized (maximum margin algorithm) and T, the model selection criterion, is minimized (best kernel parameters). More precisely, for $\boldsymbol{\theta}$ fixed, we want to have $\boldsymbol{\alpha}^0 = \arg\max W(\boldsymbol{\alpha})$ and choose $\boldsymbol{\theta}^0$ such that

$$\boldsymbol{\theta}^0 = \arg\min_{\boldsymbol{\theta}} T(\boldsymbol{\alpha}^0, \boldsymbol{\theta}).$$

When $\boldsymbol{\theta}$ is a one dimensional parameter, one typically tries a finite number of values and picks the one which gives the lowest value of the criterion T. When both T and the SVM solution are continuous with respect to $\boldsymbol{\theta}$, a better approach has been proposed by Cristianini et al. [1999]: using an incremental

[5] We note $\tilde{P}(\mathbf{x})$ as an abbreviation for $\tilde{P}_{A^*,B^*}(\mathbf{x})$

optimization algorithm, one can train an SVM with little effort when $\boldsymbol{\theta}$ is changed by a small amount. However, as soon as $\boldsymbol{\theta}$ has more than one component computing $T(\boldsymbol{\alpha}, \boldsymbol{\theta})$ for every possible value of $\boldsymbol{\theta}$ becomes intractable, and one rather looks for a way to optimize T along a trajectory in the kernel parameter space.

Using the gradient of a model selection criterion to optimize the model parameters has been proposed in [Bengio, 2000] and demonstrated in the case of linear regression and time-series prediction. It has also been proposed by [Larsen et al., 1998] to optimize the regularization parameters of a neural network.

Here we propose an algorithm that alternates the SVM optimization with a gradient step is the direction of the gradient of T in the parameter space. This can be achieved by the following iterative procedure:

1. `Initialize` $\boldsymbol{\theta}$ `to some value.`
2. `Using a standard SVM algorithm, find the maximum of the quadratic form` W`:`
$$\boldsymbol{\alpha}^0(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\alpha}} W(\boldsymbol{\alpha}, \boldsymbol{\theta}).$$
3. `Update the parameters` $\boldsymbol{\theta}$ `such that` T `is minimized. This is typically achieved by a gradient step (see below).`
4. `Go to step 2 or stop when the minimum of` T `is reached.`

Solving step 3 requires estimating how T varies with $\boldsymbol{\theta}$. We will thus restrict ourselves to the case where $K_{\boldsymbol{\theta}}$ can be differentiated with respect to $\boldsymbol{\theta}$. Moreover, we will only consider cases where the gradient of T with respect to $\boldsymbol{\theta}$ can be computed (or approximated).

Note that $\boldsymbol{\alpha}^0$ depends implicitly on $\boldsymbol{\theta}$ since $\boldsymbol{\alpha}^0$ is defined as the maximum of W. Then, if we have k kernel parameters $(\theta_1, \ldots, \theta_k)$, the total derivative of $T^0(\cdot) \equiv T(\boldsymbol{\alpha}^0(\cdot), \cdot)$ with respect to θ_p is:

$$\frac{\partial T^0}{\partial \theta_p} = \left.\frac{\partial T^0}{\partial \theta_p}\right|_{\boldsymbol{\alpha}^0 \text{ fixed}} + \frac{\partial T^0}{\partial \boldsymbol{\alpha}^0} \cdot \frac{\partial \boldsymbol{\alpha}^0}{\partial \theta_p}.$$

Having computed the gradient $\nabla_{\boldsymbol{\theta}} T^0$, a way of performing step 3 is to make a *gradient step*:

$$\delta\boldsymbol{\theta} = -\varepsilon \, \nabla_{\boldsymbol{\theta}} T^0,$$

for some small and eventually decreasing ε. The convergence can be improved with the use of second order derivatives (Newton's method):

$$\delta\boldsymbol{\theta} = -(\Delta_{\boldsymbol{\theta}} T^0)^{-1} \nabla_{\boldsymbol{\theta}} T^0,$$

where the Laplacian operator Δ is defined by

$$(\Delta_{\boldsymbol{\theta}} T^0)_{i,j} = \frac{\partial^2 T^0}{\partial \theta_i \partial \theta_j}.$$

In this formulation, additional constraints can be imposed through projection of the gradient.

4.4 Computing the gradient

In this section, we describe the computation of the gradient (with respect to the kernel parameters) of the different estimates of the generalization error. First, for the bound R^2/M^2 (see Theorem 3.1), we obtain a formulation of the derivative of the margin (section 4.4.1) and of the radius (section 4.4.2). For the validation error (see equation (3.1)), we show how to calculate the derivative of the hyperplane parameters $\boldsymbol{\alpha}^0$ and b (see section 4.4.3). Finally, the computation of the derivative of the span bound (3.12) is presented in section 4.4.4.

We first begin with a useful lemma.

Lemma 4.1 *Suppose we are given a $(n \times 1)$ vector $\mathbf{v}_\theta$ and an $(n \times n)$ matrix $\mathbf{P}_\theta$ smoothly depending on a parameter θ. Consider the function:*

$$L(\theta) = \max_{\mathbf{x} \in F} \ \mathbf{x}^\top \mathbf{v}_\theta - \frac{1}{2}\mathbf{x}^\top \mathbf{P}_\theta \mathbf{x}$$

where

$$F = \{\mathbf{x} : \mathbf{b}^\top \mathbf{x} = c, \mathbf{x} \geq 0\}.$$

Let $\bar{\mathbf{x}}$ be the the vector $\mathbf{x}$ where the maximum in $L(\theta)$ is attained. If this minimum is unique then

$$\frac{\partial L(\theta)}{\partial \theta} = \bar{\mathbf{x}}^\top \frac{\partial \mathbf{v}_\theta}{\partial \theta} - \frac{1}{2}\bar{\mathbf{x}}^\top \frac{\partial \mathbf{P}_\theta}{\partial \theta}\bar{\mathbf{x}}.$$

In other words, it is possible to differentiate L with respect to θ as if $\bar{\mathbf{x}}$ did not depend on θ. Note that this is also true if one (or both) of the constraints in the definition of F are removed.

Proof: We first need to express the equality constraint with a Lagrange multiplier λ and the inequality constraints with Lagrange multipliers γ_i:

$$L(\theta) = \max_{\mathbf{x}, \lambda, \boldsymbol{\gamma}} \ \mathbf{x}^\top \mathbf{v}_\theta - \frac{1}{2}\mathbf{x}^\top \mathbf{P}_\theta \mathbf{x} - \lambda(\mathbf{b}^\top \mathbf{x} - c) + \boldsymbol{\gamma}^\top \mathbf{x}. \tag{4.6}$$

At the maximum, the following conditions are verified:

$$\begin{array}{l} \mathbf{v}_\theta - \mathbf{P}_\theta \bar{\mathbf{x}} = \bar{\lambda}\mathbf{b} - \bar{\boldsymbol{\gamma}}, \\ \mathbf{b}^\top \bar{\mathbf{x}} = c, \\ \bar{\gamma}_i \ \bar{x}_i = 0, \quad \forall i. \end{array}$$

We will not consider here differentiability problems. The interested reader can find details in [Bonnans and Shapiro, 2000]. The main result is that whenever $\bar{\mathbf{x}}$ is unique, L is differentiable.

We have

$$\frac{\partial L(\theta)}{\partial \theta} = \bar{\mathbf{x}}^\top \frac{\partial \mathbf{v}_\theta}{\partial \theta} - \frac{1}{2}\bar{\mathbf{x}}^\top \frac{\partial \mathbf{P}_\theta}{\partial \theta}\bar{\mathbf{x}} + \frac{\partial \bar{\mathbf{x}}}{\partial \theta}^\top (\mathbf{v}_\theta - \mathbf{P}_\theta \bar{\mathbf{x}}),$$

where the last term can be written as follows,

$$\frac{\partial \bar{\mathbf{x}}}{\partial \theta}^\top (\mathbf{v}_\theta - \mathbf{P}_\theta \bar{\mathbf{x}}) = \bar{\lambda} \frac{\partial \bar{\mathbf{x}}}{\partial \theta}^\top \mathbf{b} - \frac{\partial \bar{\mathbf{x}}}{\partial \theta}^\top \bar{\boldsymbol{\gamma}}.$$

Using the derivatives of the optimality conditions, namely

$$\frac{\partial \bar{\mathbf{x}}}{\partial \theta}^\top \mathbf{b} = 0,$$

$$\frac{\partial \bar{\gamma}_i}{\partial \theta} \ \bar{x}_i + \bar{\gamma}_i \frac{\partial \bar{x}_i}{\partial \theta} = 0,$$

and the fact that either $\bar{\gamma}_i = 0$ or $\bar{x}_i = 0$ we get:

$$\frac{\partial \bar{\gamma}_i}{\partial \theta} \ \bar{x}_i = \bar{\gamma}_i \frac{\partial \bar{x}_i}{\partial \theta} = 0,$$

hence

$$\frac{\partial \bar{\mathbf{x}}}{\partial \theta}^\top (\mathbf{v}_\theta - \mathbf{P}_\theta \bar{\mathbf{x}}) = 0$$

and the result follows. □

4.4.1 Gradient of the margin

Note that in feature space, the separating hyperplane $\{\mathbf{x} : \mathbf{w} \cdot \Phi(\mathbf{x}) + b = 0\}$ has the following expansion

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i^0 y_i \Phi(\mathbf{x}_i)$$

and is normalized such that

$$\min_{1 \leq i \leq n} y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i) + b) = 1.$$

It follows from the definition of the margin in Theorem 3.1 that this latter is $\gamma = 1/\|\mathbf{w}\|$. Thus we write the bound R^2/M^2 as $R^2\|\mathbf{w}\|^2$.

The previous lemma enables us to compute the derivative of $\|\mathbf{w}\|^2$. Indeed at the optimal the value of the primal and the dual are equal,

$$\frac{1}{2}\|\mathbf{w}\|^2 = W(\boldsymbol{\alpha}^0),$$

and the lemma can be applied to the standard SVM optimization problem (2.6), giving

$$\frac{\partial \|\mathbf{w}\|^2}{\partial \theta_p} = -\sum_{i,j=1}^{n} \alpha_i^0 \alpha_j^0 y_i y_j \frac{\partial K(\mathbf{x}_i, \mathbf{x}_j)}{\partial \theta_p}$$

4.4.2 Gradient of the radius

Computing the radius of the smallest sphere enclosing the training points can be achieved by solving the following quadratic problem [Vapnik, 1998]:

$$R^2 = \max_{\boldsymbol{\beta}} \sum_{i=1}^{n} \beta_i K(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j=1}^{n} \beta_i \beta_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{4.7}$$

under constraints

$$\sum_{i=1}^{n} \beta_i = 1$$
$$\forall i \quad \beta_i \geq 0$$

We can again use the previous lemma to compute the derivative of the radius:

$$\frac{\partial R^2}{\partial \theta_p} = \sum_{i=1}^{n} \beta_i^0 \frac{\partial K(\mathbf{x}_i, \mathbf{x}_i)}{\partial \theta_p} - \sum_{i,j=1}^{n} \beta_i^0 \beta_j^0 \frac{\partial K(\mathbf{x}_i, \mathbf{x}_j)}{\partial \theta_p},$$

where $\boldsymbol{\beta}^0$ maximizes the previous quadratic form.

In practice, couple of experiments we conducted showed that it is better to use the variance, i.e. fix $\beta_i = 1/n$, than the radius.

4.4.3 Gradient of the hyperplane parameters

Let us compute the derivative of $\boldsymbol{\alpha}^0$ with respect to a parameter $\boldsymbol{\theta}$ of the kernel. For this purpose, we need an analytical formulation for $\boldsymbol{\alpha}^0$. First, we suppose that the points which are not support vectors are removed from the training set. This assumption can be done without any loss of generality since removing a point which is not support vector does not affect the solution. Then, the fact that all the points lie on the margin can be written

$$\underbrace{\begin{pmatrix} \mathbf{K}^Y & \mathbf{Y} \\ \mathbf{Y}^\top & 0 \end{pmatrix}}_{\mathbf{H}} \begin{pmatrix} \boldsymbol{\alpha}^0 \\ b \end{pmatrix} = \begin{pmatrix} \mathbf{1} \\ 0 \end{pmatrix}, \tag{4.8}$$

where $\mathbf{K}^Y_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$. If there are n support vectors, $\mathbf{H}$ is a $(n+1) \times (n+1)$ matrix. The parameters of the SVMs can be written as:

$$(\boldsymbol{\alpha}^0, b)^\top = \mathbf{H}^{-1}(1 \cdots 1\ 0)^\top.$$

We are now able to compute the derivatives of those parameters with respect to a kernel parameter θ_p. Indeed, since the derivative of the inverse of a matrix M depending on a parameter θ_p can be written [6]

$$\frac{\partial \mathbf{M}^{-1}}{\partial \theta_p} = -\mathbf{M}^{-1} \frac{\partial \mathbf{M}}{\partial \theta_p} \mathbf{M}^{-1}, \tag{4.9}$$

it follows that

$$\frac{\partial (\boldsymbol{\alpha}^0, b)}{\partial \theta_p} = -\mathbf{H}^{-1} \frac{\partial \mathbf{H}}{\partial \theta_p} \mathbf{H}^{-1} (1 \cdots 1\ 0)^\top,$$

and finally

$$\boxed{\frac{\partial (\boldsymbol{\alpha}^0, b)}{\partial \theta_p} = -\mathbf{H}^{-1} \frac{\partial \mathbf{H}}{\partial \theta_p} (\boldsymbol{\alpha}^0, b)^\top.}$$

We can easily use the result of this calculation to recover the computation $\frac{\partial \|\mathbf{w}\|^2}{\partial \theta_p}$. Indeed, if we denote $\tilde{\boldsymbol{\alpha}} = (\boldsymbol{\alpha}^0, b)$, we have $\|\mathbf{w}\|^2 = (\boldsymbol{\alpha}^0)^\top \mathbf{K}^Y \boldsymbol{\alpha}^0 = \tilde{\boldsymbol{\alpha}}^\top \mathbf{H} \tilde{\boldsymbol{\alpha}}$ and it turns out that:

$$\begin{aligned}
\frac{\partial \|\mathbf{w}\|^2}{\partial \theta_p} &= \tilde{\boldsymbol{\alpha}}^\top \frac{\partial \mathbf{H}}{\partial \theta_p} \tilde{\boldsymbol{\alpha}} + 2 \tilde{\boldsymbol{\alpha}} \mathbf{H} \frac{\partial \tilde{\boldsymbol{\alpha}}}{\partial \theta_p} \\
&= \tilde{\boldsymbol{\alpha}}^\top \frac{\partial \mathbf{H}}{\partial \theta_p} \tilde{\boldsymbol{\alpha}} - 2 \tilde{\boldsymbol{\alpha}} \mathbf{H} \mathbf{H}^{-1} \frac{\partial \mathbf{H}}{\partial \theta_p} \tilde{\boldsymbol{\alpha}} \\
&= -\tilde{\boldsymbol{\alpha}}^\top \frac{\partial \mathbf{H}}{\partial \theta_p} \tilde{\boldsymbol{\alpha}} \\
&= -(\boldsymbol{\alpha}^0)^\top \frac{\partial \mathbf{K}^Y}{\partial \theta_p} \boldsymbol{\alpha}^0.
\end{aligned}$$

4.4.4 Computing the derivative of the span-rule

Combining equation (3.14) and (4.9), we get the derivative of the span

$$\frac{\partial S_p^2}{\partial \theta_p} = S_p^4 \left(\tilde{\mathbf{K}}_{SV}^{-1} \frac{\partial \tilde{\mathbf{K}}_{SV}}{\theta_p} \tilde{\mathbf{K}}_{SV}^{-1} \right)_{pp}$$

Thus, the complexity of computing the derivative of the span-rule with respect to a parameter θ_p of the kernel requires only the computation of $\frac{\partial K(\mathbf{x}_i, \mathbf{x}_j)}{\partial \theta_p}$ and the inversion of the matrix $\tilde{\mathbf{K}}_{SV}$. The complexity of these operations is not larger than that of the quadratic optimization problem itself.

There is however a problem in this approach: the value given by the span-rule is not continuous. By changing smoothly the value of the parameters $\boldsymbol{\theta}$, the coefficients α_p change continuously, but the span S_p^2 does not. There is actually a discontinuity for most support vectors when the set of support vectors changes. This can be easily understood from equation (3.10): suppose that upon changing the value of the parameter from $\boldsymbol{\theta}$ to $\boldsymbol{\theta} + \boldsymbol{\varepsilon}$, a point $\mathbf{x}_m$ is not a support vector anymore, then for all other support vectors $(\mathbf{x}_p)_{p \neq m}$, the set Λ_p is going to be smaller and a discontinuity is likely to appear for the value of $S_p = d(\Phi(\mathbf{x}_p), \Lambda_p)$.

The situation is explained in figure 4.2: we plotted the value of the span of a support vector $\mathbf{x}_p$ versus the width of an RBF kernel σ. Almost everywhere the span is decreasing, hence a negative derivative, but some jumps appear, corresponding to a change in the set of support vectors. Moreover the span is

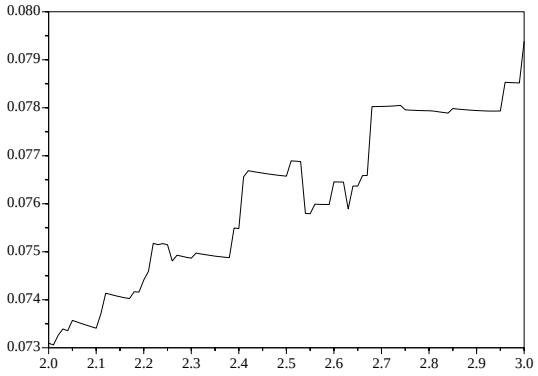


Figure 4.2: Value of $\sum S_p^2$, the sum of the span of the training points for different values of the width of an RBF kernel varying in the small vicinity

globally increasing: the value of the derivate does not give us a good indication of the global evolution of the span.

One way to solve is this problem is to try to smooth the behavior of the span. This can be done by imposing the following additional constraint in the definition of Λ_p in equation (3.10): $|\lambda_i| \leq c\ \alpha_i^0$, where c is a constant. Given this constraint, if a point $\mathbf{x}_m$ is about to leave or has just entered the set of support vectors, it will not have a large influence on the span of the other support vectors, since α_m^0 will be small. The effect of this constraint is to make the set Λ_p become "continuous" when the set of support vectors changes.

However this new constraint prevents us from computing the span as efficiently as in equation (3.14). A possible solution is to replace the constraint by a regularization term in the computation of the span:

$$S_p^2 = \min_{\boldsymbol{\lambda},\ \sum \lambda_i = 1} \left\| \Phi(\mathbf{x}_p) - \sum_{i \neq p}^{n} \lambda_i \Phi(\mathbf{x}_i) \right\|^2 + \eta \sum_{i \neq p}^{n} \frac{1}{\alpha_i^0} \lambda_i^2$$

With this new definition of the span, equation (3.14) becomes:

$$S_p^2 = 1/(\tilde{\mathbf{K}}_{SV} + \mathbf{D})_{pp}^{-1} - \mathbf{D}_{pp},$$

where $\mathbf{D}$ is a diagonal matrix with elements $\mathbf{D}_{ii} = \eta/\alpha_i^0$ and $\mathbf{D}_{n+1,n+1} = 0$. As shown on figure 4.3 , the span is now much smoother and its minimum is still at the right place. In our experiments, we took $\eta = 0.1$.

Note that computing the derivative of this new expression is no more difficult than the previous span expression.

It is interesting to look at the leave-one-out error for SVMs without threshold. In this case, the value of the span with regularization writes:

$$S_p^2 = \min_{\boldsymbol{\lambda}} \left\| \Phi(\mathbf{x}_p) - \sum_{i \neq p} \lambda_i \Phi(\mathbf{x}_i) \right\|^2 + \eta \sum_{i \neq p}^{n} \frac{1}{\alpha_i^0} \lambda_i^2$$

As already pointed out in section 3.2.3.0, if $\eta = 0$, the value of span is:

$$S_p^2 = \frac{1}{(\mathbf{K}_{SV}^{-1})_{pp}}.$$

and we recover the Opper-Winther bound.

[6]This inequality can be easily proved by differentiating $\mathbf{M}\mathbf{M}^{-1} = \mathbf{I}$

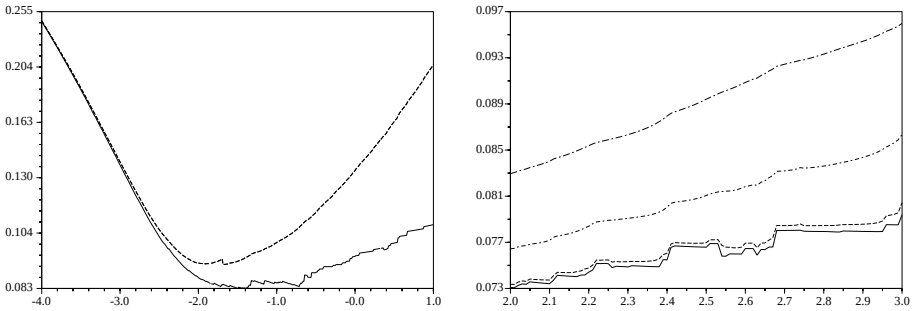


Figure 4.3: Left: the minima of the span prediction with regularization (dashed line) and without regularization (solid line) are close. Right: detailed behavior of the span for different values of the regularizer, $\eta = 0,\ 0.001,\ 0.01,\ 0.1$

On the other hand, if $\eta = +\infty$, then $\boldsymbol{\lambda} = 0$ and $S_p^2 = K(\mathbf{x}_p, \mathbf{x}_p)$. In this case, the span bound is identical to the Jaakkola-Haussler one.

In a way, the span bound with regularization is in between the bounds of Opper-Winther and Jaakkola-Haussler.

4.4.5 Gradient of the posterior probabilities

The posterior probabilities as computed in section 4.2 depend on 2 variables A^0 and B^0 which are determined by the solution of the optimization problem (4.3). This latter depend on quantities such as $f(\mathbf{x}_i)$ which change when an hyperparamter θ_p is modified. The problem is to compute $\partial A^0/\partial\theta_p$ and $\partial B^0/\partial\theta_p$. This is easily solved by the following lemma whose derivation can be found in [Bengio, 2000].

Lemma 4.2 *Let $F(\mathbf{x}, \theta)$ a twice derivable functional. Let*

$$\mathbf{x}^0(\theta) = \arg\min_{\mathbf{x}} F(\mathbf{x}, \theta).$$

Then

$$\frac{\partial \mathbf{x}^0}{\partial \theta} = -\left(\frac{\partial^2 F}{\partial \mathbf{x}^2}\right)^{-1} \frac{\partial^2 F}{\partial \mathbf{x} \partial \theta}$$

Proof: By definition, $\mathbf{x}_0(\theta)$ is the minimum of F and

$$\left.\frac{\partial F(\mathbf{x}, \theta)}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}^0(\theta)} = 0.$$

Taking the derivative of this equality with respect to θ yields the desired result.

4.4.6 Stability of the SVM solution

In this section, we will try to see how robust is the SVM solution to a small perturbation. More precisely, let us consider hard margin linear SVMs without threshold and investigate the behavior of $\mathbf{w}$ when a training point $\mathbf{x}_i$ gets a small perturbation $d\mathbf{x}_i$.

For this analysis, we will suppose without any loss of generality that all points are support vectors (removing the training points which are not support vectors do not affect the solution) and that the point being perturbed is $\mathbf{x}_1$. The goal is thus to compute (or upper-bound) $\|d\mathbf{w}\|$ as a function of $d\mathbf{x}_1$. To simplify the notations, we will introduce the weight vector $\boldsymbol{\lambda}$ defined as

$$\lambda_i = y_i \alpha_i.$$

Since $\mathbf{w} = \sum \lambda_i \mathbf{x}_i$,

$$d\mathbf{w} = \sum_{i=1}^{n} d\lambda_i \mathbf{x}_i + \lambda_1 d\mathbf{x}_1$$

and

$$\|d\mathbf{w}\|^2 = d\boldsymbol{\lambda}^\top K d\boldsymbol{\lambda} + \lambda_1^2 \|d\mathbf{x}_1\|^2 + 2\lambda_1 d\boldsymbol{\lambda}^\top \mathbf{v}, \tag{4.10}$$

with $v_i = d\mathbf{x}_1 \cdot \mathbf{x}_i$.

Similarly to section 4.4.3, an analytic expression for $d\boldsymbol{\lambda}$ can be obtained. The fact that the support vectors lie on the margin can be written as $K\boldsymbol{\lambda} = Y$, where Y is the vector of outputs of training points. As a consequence, $\boldsymbol{\lambda} = K^{-1}Y$ and

$$d\boldsymbol{\lambda} = d(K^{-1})Y = -K^{-1}dK\ K^{-1}Y = -K^{-1}dK\ \boldsymbol{\lambda}. \tag{4.11}$$

Now $dK_{ij} = (\mathbf{x}_i + d\mathbf{x}_i) \cdot (\mathbf{x}_j + d\mathbf{x}_j) - \mathbf{x}_i \cdot \mathbf{x}_j = \mathbf{x}_i \cdot d\mathbf{x}_j + \mathbf{x}_j \cdot d\mathbf{x}_i$. Introducing $\mathbf{u} = (1\ 0 \dots 0)^\top$,

$$dK = \mathbf{u}\mathbf{v}^\top + \mathbf{v}\mathbf{u}^\top$$

and (4.11) gives

$$d\boldsymbol{\lambda} = -K^{-1}(\lambda_1 \mathbf{v} + \boldsymbol{\lambda} \cdot \mathbf{v}\ \mathbf{u}). \tag{4.12}$$

Combining equations (4.10) and (4.12),

$$\begin{aligned}
\|d\mathbf{w}\|^2 &= \lambda_1^2 \|d\mathbf{x}_1\|^2 - d\boldsymbol{\lambda}^\top(\lambda_1 \mathbf{v} + \boldsymbol{\lambda} \cdot \mathbf{v}\ \mathbf{u}) + 2\lambda_1 d\boldsymbol{\lambda}^\top \mathbf{v} \\
&= \lambda_1^2 \|d\mathbf{x}_1\|^2 + d\boldsymbol{\lambda}^\top(\lambda_1 \mathbf{v} - \boldsymbol{\lambda} \cdot \mathbf{v}\ \mathbf{u}) \\
&= \lambda_1^2 \|d\mathbf{x}_1\|^2 - (\lambda_1 \mathbf{v}^\top + \boldsymbol{\lambda} \cdot \mathbf{v}\ \mathbf{u}^\top) K^{-1}(\lambda_1 \mathbf{v} - \boldsymbol{\lambda} \cdot \mathbf{v}\ \mathbf{u}) \\
&= \lambda_1^2(\|d\mathbf{x}_1\|^2 - \mathbf{v}^\top K^{-1}\mathbf{v}) + (\boldsymbol{\lambda} \cdot \mathbf{v})^2 K_{11}^{-1}
\end{aligned} \tag{4.13}$$

Since $\lambda \cdot \mathbf{v} = \mathbf{w} \cdot d\mathbf{x}_1$ and $K_{11}^{-1} = S_1^{-2}$, the invert square of the span of point $\mathbf{x}_1$ (see eq 3.14), the following bound holds

$$\|d\mathbf{w}\| \leq |\lambda_1| \|d\mathbf{x}_1\| + \frac{|\mathbf{w} \cdot d\mathbf{x}_1|}{d(\mathbf{x}_1, E_{\mathbf{x}_2,\dots,\mathbf{x}_n})},$$

since the span S_1 is also the distance of the point $\mathbf{x}_1$ to the subspace spanned by the other support vectors, $E_{\mathbf{x}_2,\dots,\mathbf{x}_n}$.

It is possible to get a tighter upper bound: in equation (4.13), one can see that $\|d\mathbf{x}_1\|^2 - \mathbf{v}^\top K^{-1}\mathbf{v}$ is the distance between $d\mathbf{x}_1$ and the subspace E spanned by the points $0, \mathbf{x}_1, \cdots, \mathbf{x}_n$ and finally,

$$\|d\mathbf{w}\| \leq |\lambda_1| d(d\mathbf{x}_1, E_{\mathbf{x}_1,\dots,\mathbf{x}_n}) + \frac{|\mathbf{w} \cdot d\mathbf{x}_1|}{d(\mathbf{x}_1, E_{\mathbf{x}_2,\dots,\mathbf{x}_n})}. \tag{4.14}$$

Figure 4.4 gives some insight on the qualitative behavior of the bound given in equation (4.14).

Ideally, one would like to show that $d\mathbf{w}$ is bounded by a constant times $d\mathbf{x}_1$. That would prove that the SVM solution is stable. In the case of a soft margin SVM, λ_1 and $\mathbf{w}$ are bounded, but the problem in equation (4.14) comes from $d(\mathbf{x}_1, E_{\mathbf{x}_2,\dots,\mathbf{x}_n})$. If this quantity is small, then $\|d\mathbf{w}\| / \|d\mathbf{x}_1\|$ is large, as shown in figure 4.5.

This instability problem is not solved using the L_1 soft margin formulation, but can be solved with L_2 one. Indeed, as explained in section 4.1, the L_2 penalization of the slacks is equivalent to a normal SVM training with the modified kernel $\tilde{K} = K + 1/C\ I$. Then $\tilde{K}_{11}^{-1} = \mathbf{u}^\top \tilde{K}^{-1}\mathbf{u}$ is bounded by the largest eigenvalue of $\tilde{K}^{-1}$. But since K is positive definite, the smallest eigenvalue of $\tilde{K}$ is at least $1/C$ and

$$\frac{1}{d(\mathbf{x}_1, E_{\mathbf{x}_2,\dots,\mathbf{x}_n})} \leq \sqrt{C}.$$

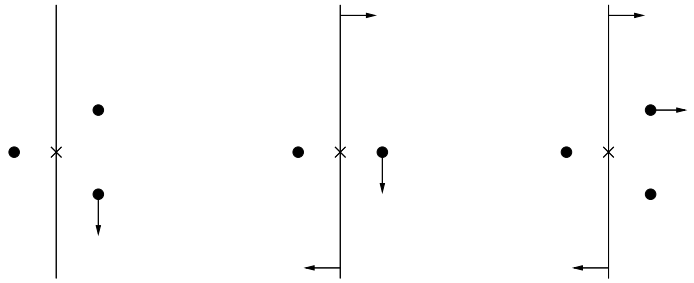


Figure 4.4: Illustration of equation (4.14). When $d\mathbf{x}_1$ is in the subspace spanned by the support vectors *and* $\mathbf{w} \cdot d\mathbf{x}_1 = 0$, then $d\mathbf{w} = 0$ (left). But when one of those conditions is not satisfied, there is a change in the hyperplane position (middle and right).

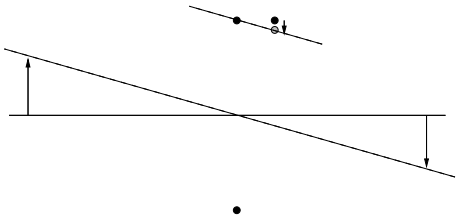


Figure 4.5: When $d(\mathbf{x}_1, E_{\mathbf{x}_2,\ldots,\mathbf{x}_n})$ is small, a small change $d\mathbf{x}_1$ can entail a large change in $\mathbf{w}$. Indeed, in this example, the hyperplane follows the alignment of the points at the top.

4.5 Experiments

Experiments have been carried out to assess the performance and feasibility of our method.

The first set of experiments consists in finding automatically the optimal value of two parameters: the width of an RBF kernel and the constant C in equation (4.1). The second set of experiments corresponds to the optimization of a large number of scaling factors in the case of handwritten digit recognition. We then show that optimizing scaling factors leads naturally to feature selection and demonstrate the application of the method to the selection of relevant features in several databases.

4.5.1 Optimization details

The core of the technique we present here is a gradient descent algorithm. We used the optimization toolbox of Matlab to perform it. It includes second order updates to improve the convergence speed. Since we are not interested in the exact value of the parameters minimizing the functional, we used a loose stopping criterion.

4.5.2 Benchmark databases

In a first set of experiments, we tried to select automatically the width σ of a RBF kernel,

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(-\sum_{i=1}^{n} \frac{(x_i - y_i)^2}{2n\sigma^2}\right)$$

along the constant C penalizing the training error appearing in equation (4.1).

In order to avoid adding positivity constraints in the optimization problem (for the constant C and the width σ of the RBF kernel), we use the parameterization $\boldsymbol{\theta} = (\log C, \log \sigma)$. Moreover, this turns out to give a more stable optimization. The initial values are $C = 1$ and $\log \sigma = -2$. Each component being normalized by its standard deviation, this corresponds to a rather small value for σ.

	Cross-validation	R^2/M^2	Span-bound
Breast Cancer	26.04 ± 4.74	26.84 ± 4.71	25.59 ± 4.18
Diabetis	23.53 ± 1.73	23.25 ± 1.7	23.19 ± 1.67
Heart	15.95 ± 3.26	15.92 ± 3.18	16.13 ± 3.11
Thyroid	4.80 ± 2.19	4.62 ± 2.03	4.56 ± 1.97
Titanic	22.42 ± 1.02	22.88 ± 1.23	22.5 ± 0.88

Table 4.1: Test error found by different algorithms for selecting the SVM parameters C and σ. The first column reports the results from [Rätsch et al., 2001]. In the second and last column, the parameters are found by minimizing R^2/M^2 and the span-bound using a gradient descent algorithm.

	Cross-validation	R^2/M^2	Span-bound
Breast Cancer	500	14.2	7
Diabetis	500	12.2	9.8
Heart	500	9	6.2
Thyroid	500	3	11.6
Titanic	500	6.8	3.4

Table 4.2: Average number of SVM trainings on one training set needed to select the parameters C and σ using standard cross-validation or by minimizing R^2/M^2 or the span-bound.

We used benchmark databases described in [Rätsch et al., 2001]. Those databases, as long as the 100 different training and test splits are available at `http://ida.first.gmd.de/~raetsch/data/benchmarks.htm`.

We followed the same experimental setup as in [Rätsch et al., 2001]. On each of the first 5 training sets, the kernel parameters are estimated using either 5-fold cross-validation, minimization of R^2/M^2, or the span-bound. Finally, the kernel parameters are computed as the median of the 5 estimations.

The results are shown in table 4.1.

It turns out that minimizing R^2/M^2 or the span estimates yields approximately the same performances as picking-up the parameters which minimize the cross-validation error. This is not very surprising since cross-validation is known to be an accurate method for choosing the hyper-parameters of any learning algorithm.

A more interesting comparison is the computational cost of these methods. Table 4.2 shows how many SVM trainings in average are needed to select the kernel parameters on each split. The results for cross-validation are the ones reported in [Rätsch et al., 2001]. They tried 10 different values for C and σ and performed 5-fold cross-validation. The number of SVM trainings on each of the 5 training set needed by this method is $10 \times 10 \times 5 = 500$.

The gain in complexity is impressive: on average *100* times fewer SVM training iterations are required to find the kernel parameters. The main reason for this gain is that there were two parameters to optimize. Because of computational reasons, exhaustive search by cross-validation can not handle the selection of more than 2 parameters, whereas our method can, as highlighted in the next section.

Discussion As explained in section 3.1.3, R^2/M^2 can seem to be a rough upper bound of the span-bound, which is in an accurate estimate of the test error [Chapelle and Vapnik, 1999]. However in the process of choosing the kernel parameters, what matters is to have a bound whose minimum is close to the optimal kernel parameters. Even if R^2/M^2 cannot be used to estimate the test error, the previous experiments show that its minimization yields quite good results. The generalization error obtained by minimizing the span-bound (cf table 4.1) are just slightly better. Since the minimization of the latter is more difficult to implement and to control (more local minima), we recommend in practice to minimize R^2/M^2. In the experiments of the following section, we will only relate experiments with this bound, but similar results have been obtained with the span-bound.

4.5.3 Automatic selection of scaling factors

Finding automatically good scaling factors is also refereed in the machine learning community as adaptive metric. It has been well studied for a base algorithm such as K-Nearest Neighbor [Lowe, 1995; Domeniconi et al., 2000] and it has been more recently applied SVM in [Tsuda, 1999a], where the author tries to minimize a bound on the generalization error based on entropy number.

In this experiment, we try to choose the scaling factors for an RBF and polynomial kernel of degree 2. More precisely, we consider kernels of the following form:

$$K(\mathbf{x},\mathbf{y}) = \exp\left(-\sum_i \frac{(x_i - y_i)^2}{2\sigma_i^2}\right) \tag{4.15}$$

and

$$K(\mathbf{x},\mathbf{y}) = \left(1 + \sum_i \frac{x_i y_i}{\sigma_i^2}\right)^2 \tag{4.16}$$

Most of the experiments have been carried out on the USPS handwritten digit recognition database. This database consists of 7291 training examples and 2007 test examples of digit images of size 16x16 pixels. We try to classify digits 0 to 4 against 5 to 9. The training set has been split into 23 subsets of 317 examples and each of this subset has been used successively during the training.

To assess the feasibility of our gradient descent approach for finding kernel parameters, we first used only 16 parameters, each one corresponding to a scaling factor for a squared tile of 16 pixels as shown on figure 4.6.

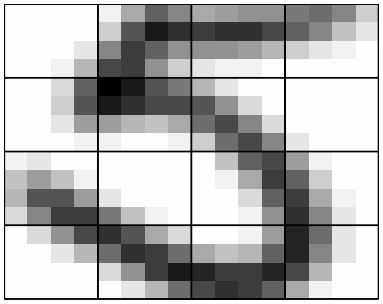


Figure 4.6: On each of the 16 tiles, the scaling factors of the 16 pixels are identical.

The scaling parameters were initialized to 1. The evolution of the test error and of the bound R^2/M^2 is plotted versus the number of iterations in the gradient descent procedure in figures 4.7 (polynomial kernel) and 4.8 (RBF kernel).

Note that for the polynomial kernel, the test error went down to 9% whereas the best test error with only one scaling parameter is 9.9%. Thus, by taking several scaling parameters, we managed to make the test error decrease.

It might be interesting to have a look at the value of the scaling coefficients we have have found. For this purpose, we took 256 scaling parameters (one per pixel) and minimized R^2/M^2 with a polynomial kernel. The map of the scaling coefficient is shown in figure 4.9.

The result is quite consistent with what one could expect in such a situation: the coefficients near the border of the picture are smaller than those in the middle of the picture, so that these coefficients can be directly interpreted as measures of the relevance of the corresponding features.

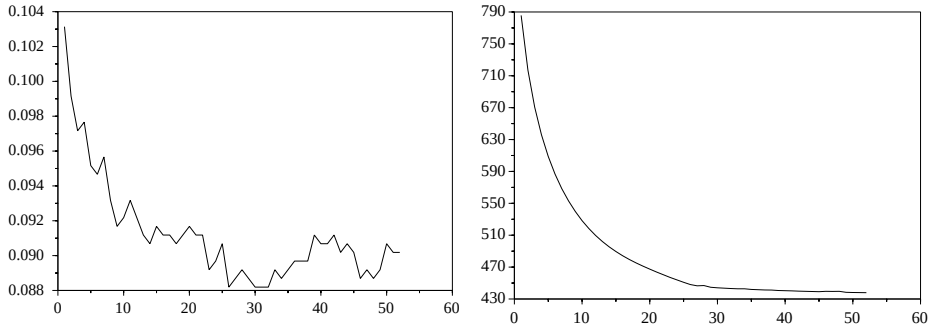


Figure 4.7: Evolution of the test error (left) and of the bound R^2/M^2 (right) during the gradient descent optimization with a polynomial kernel

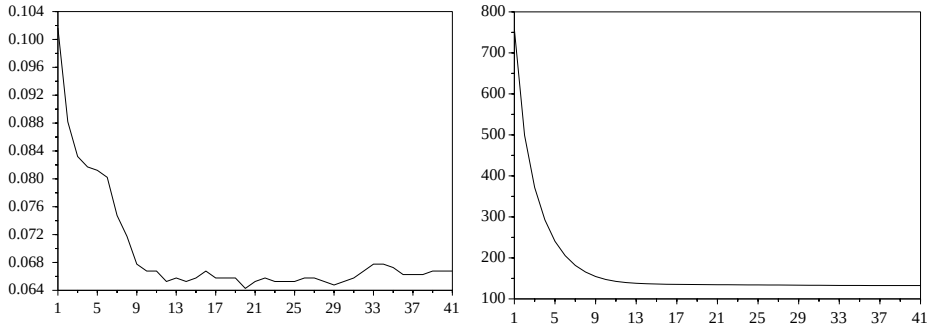


Figure 4.8: Evolution of the test error (left) and of the bound R^2/M^2 (right) during the gradient descent optimization with an RBF kernel

4.5.4 Minimizing other estimates

Span-bound

The span-bound (3.12) has been minimized over the scaling parameters of an RBF kernel. The results of the minimization are reported on figure 4.10. The test error realized at the end of the minimization is better than the one obtained by minimizing R^2/M^2.

Validation error

We also tried to choose the kernel parameters which minimize a validation error. For this purpose, we divided our 317 training examples into a smaller training set of 217 training examples and a validation set of 100 examples. Results are reported in figure 4.11

Discussion This experiment can be considered as a sanity check experiment. Indeed, it proves it is feasible to choose multiple kernel parameters of an SVM and that it does not lead to overfitting. However, the gain in test error was not our main motivation since we did not expect any significant improvement on such a problem where most features play a similar role (taking all scaling factors equal on this database seems a reasonable choice). However as highlighted by figure 4.9, this method can be a powerful tool to perform feature selection.

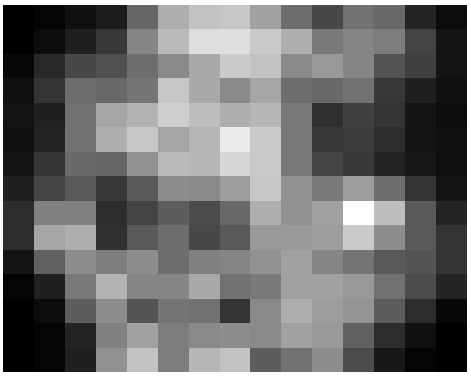


Figure 4.9: Scaling factors found by the optimization procedure: darker means smaller scaling factor

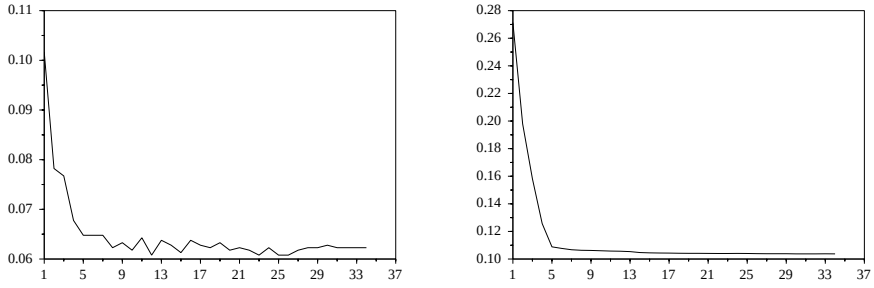


Figure 4.10: Evolution of the test error and of the span-bound during the gradient descent optimization with an RBF kernel

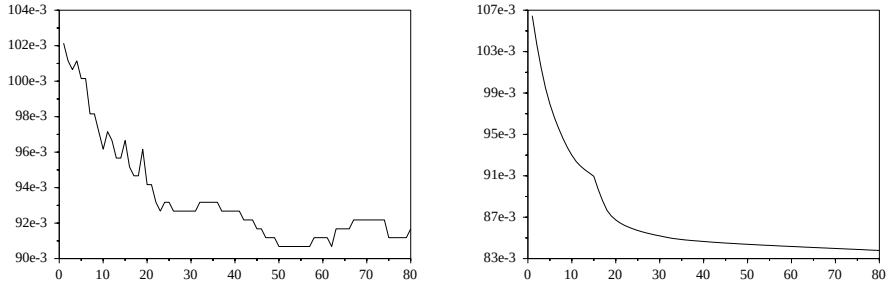


Figure 4.11: Evolution of the validation and test errors during the gradient descent optimization with a polynomial kernel

Chapter 5

Applications

5.1 Feature selection

5.1.1 The Feature Selection problem

The motivation for feature selection is three-fold:

1. Improve the generalization error.
2. Determine the relevant features (for explanatory purposes).
3. Reduce the dimensionality of the input space (for real-time applications).

The feature selection problem can be addressed in the following two ways: (1) given a fixed $m \ll d$, find the m features that give the smallest expected generalization error; or (2) given a maximum allowable generalization error γ, find the smallest m. In both of these problems the expected generalization error is of course unknown, and thus must be estimated. In this section we will consider problem (1). Note that choices of m in problem (1) can usually can be reparameterized as choices of γ in problem (2).

Problem (1) is formulated as follows. Given a fixed set of functions $y = f(\mathbf{x}, \boldsymbol{\alpha})$ we wish to find a preprocessing of the data $\mathbf{x} \mapsto \mathbf{x} * \boldsymbol{\sigma}$, $\boldsymbol{\sigma} \in \{0,1\}^d$, and the parameters $\boldsymbol{\alpha}$ of the function f that give the minimum value of

$$R(\sigma, \alpha) = \int \mathrm{I}_{f((\mathbf{x} * \boldsymbol{\sigma}), \boldsymbol{\alpha}) \neq y} dP(\mathbf{x}, y) \tag{5.1}$$

subject to $\|\sigma\|_0 = m$, where $P(\mathbf{x}, y)$ is unknown underlying probability generating the data, $\mathbf{x} * \boldsymbol{\sigma} = (x_1\sigma_1, \ldots, x_d\sigma_d)$ denotes an elementwise product and $\|\cdot\|_0$ is the "0-norm". [7]

In the literature one distinguishes between two types of method to solve this problem: the so-called filter and wrapper methods [Blum and Langley, 1997]. Filter methods are defined as a preprocessing step to induction that can remove irrelevant attributes before induction occurs, and thus wish to be valid for any set of functions $f(\mathbf{x}, \boldsymbol{\alpha})$. For example, one popular filter method is to use Pearson correlation coefficients.

The wrapper method, on the other hand, is a more direct attempt to minimize (5.1). It is defined as a search through the space of feature subsets using the estimated accuracy from an induction algorithm as a measure of goodness of a particular feature subset. More precisely, for a given $\boldsymbol{\sigma}$, the induction algorithm outputs a function $f((\mathbf{x} * \boldsymbol{\sigma}), \boldsymbol{\alpha}^0(\boldsymbol{\sigma}))$. Then, given an approximation T (such as performance on a validation set) of R, one minimizes $T^0(\boldsymbol{\sigma}) = T(\boldsymbol{\sigma}, \boldsymbol{\alpha}^0(\boldsymbol{\sigma}))$. Wrapper methods can provide more accurate solutions than filter methods [Kohavi and John, 1997]. Filter methods require better estimates of the underlying density as they must define subsets of features which work for any set of functions, rather than just a set with small capacity. In addition, they do not take into account a particular algorithm bias toward retaining certain types of features. Wrapper methods take this bias into account; however, they

[7] Please note that $\|\cdot\|_0$ is not a norm, since $\|c\boldsymbol{\sigma}\|_0 \neq |c| \ \|\boldsymbol{\sigma}\|_0$ for $\boldsymbol{\sigma} \neq 0, |c| \notin \{0,1\}$.

are in general more computationally expensive since the estimated generalization error T of the induction algorithm must be evaluated over each feature set (vector $\boldsymbol{\sigma}$) considered. Since there are 2^d different vectors $\boldsymbol{\sigma}$, wrapper methods usually use greedy approaches such as forward or backward elimination. For a review of filter and wrapper methods see [Blum and Langley, 1997; Leray and Gallinari, 1999].

In this section, we introduce a feature selection algorithm for SVMs that takes advantage of the performance increase of wrapper methods whilst avoiding their computational complexity. Note, some previous work on feature selection for SVMs does exist, however results have been limited to linear kernels [Bradley and Mangasarian, 1998; Guyon et al., 2002] or linear probabilistic models [Jebara and Jaakkola, 2000]. Our approach can be applied to non-linear problems.

Whilst these algorithms are tractable, we show empirically that they provide solutions which can either improve generalization ability or significantly reduce the number of features used in high dimensional problems. This illustrated on two toy problems and on a DNA micro-array database where it is important to find which genes are relevant in performing the classification.

5.1.2 Feature Selection for SVMs

RFE algorithm

We first present an intuitive method of feature selection for *linear* SVMs called *Recursive Feature Elimination* (RFE) [Guyon et al., 2002]. It is based on the idea that the importance of a feature i should be related to the norm of its weight $|w_i|$. Combining this idea with a backward elimination gives the following algorithm:

1. Train an SVM on the active features.
2. Remove some of the features corresponding to small $|w_i|$.
3. Go back to step 1 or stop if there are only m features left.

This algorithm is very simple to implement, but it is not straightforward to extend it to non-linear SVMs.

Scaling factors

The main idea of our feature selection approach for SVMs is based on the automatic tuning of scaling coefficients presented in section 4.5.3. Indeed, if one of the input components is useless for the classification problem, its scaling factor is likely to become small. But if a scaling factor becomes small enough, it means that it is possible to remove it without affecting the classification algorithm. This leads to the following idea for feature selection: keep the features whose scaling factors are the largest.

We consider the following parametrization of the kernel.

$$K_{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{y}) = K(\boldsymbol{\sigma}^\top \mathbf{x}, \boldsymbol{\sigma}^\top \mathbf{y}), \tag{5.2}$$

where $\boldsymbol{\sigma} \in \mathbb{R}^d$.

We compute $\boldsymbol{\sigma}$ and $\boldsymbol{\alpha}$ using the following iterative procedure:

1. Initialize $\boldsymbol{\sigma} = (1, \ldots, 1)$.
2. Solve the SVM optimization problem with kernel (5.2).
3. Minimize the estimate of the error T with respect to $\boldsymbol{\theta}$ with a gradient step.
4. If a local minimum of T is not reached go to step 2.
5. Discard dimensions corresponding to small elements in $\boldsymbol{\sigma}$ and return to step 2 (usually discard half of the components). Stop when $\|\boldsymbol{\sigma}\|_0 \leq m$.

When $T = R^2/M^2 = R^2\mathbf{w}^2$, the RFE algorithm can be seen as an approximation of this method. Indeed, in the linear case, if R^2 is approximated by the variance ($\beta_i = 1/n$ in equation (4.7)), then we get thanks to the results of sections 4.4.1 and 4.4.2,

$$\begin{aligned}\left.\frac{\partial R^2/M^2}{\partial \sigma_p}\right|_{\boldsymbol{\sigma}=1} &= -R^2 \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^p x_j^p + \mathbf{w}^2 \left(\frac{1}{n} \sum_{i=1}^{n} (x_i^p)^2 - \frac{1}{n^2} \sum_{i,j=1}^{n} x_i^p x_j^p \right) \\ &= -R^2 w_p^2 + \mathbf{w}^2 Var(x_p),\end{aligned}$$

with $Var(x_p)$ the variance of the input component p. When applying RFE, a standard preprocessing is to divide each component by its standard deviation and in this case $Var(x_p) = 1$. After one gradient step of our method, the smallest scaling factors would correspond to components with the smallest weights w_p^2 and RFE amounts to make one gradient step and remove components with the smallest σ.

Sparsity term

If one wants to keep only m features, it is possible to approximate the integer programming by minimizing

$$R^2/M^2(\boldsymbol{\sigma}) + \lambda \sum_{i=1}^{d} \sigma_i^p \tag{5.3}$$

subject to $\sum_i \sigma_i = m, \;\; \sigma_i \geq 0, \; i = 1, \ldots, d$.

For large enough λ as $p \to 0$ only m elements of σ will be nonzero. In our experiments, we did not use such a sparsity term.

Refined feature removal

The algorithm presented above has several iterations. At each one, a gradient descent on R^2/M^2 is performed and some features discarded (usually half of them). When the gradient descent algorithm ends up with some very small scaling factors, there is no problem in discarding the corresponding components. That was the motivation for retaining the components with largest scaling factors. However, when all the components have a non negligible scaling factor, it is not obvious that the components with the smallest ones should be removed. Instead, we propose to remove the features such that the resulting R^2/M^2 will be as small as possible (see figure 5.1). It is possible to do so

Explicitly After the gradient descent, for each component, put its scaling factor to 0, retrain an SVM and assign the resulting R^2/M^2 as a score for this feature. Then discard the features with the smallest scores.

Approximatively Similarly to the *Optimal Brain Damage* technique [LeCun et al., 1990], the idea is to use second order information to estimate the effecting of putting a scaling factor to 0. More precisely, let $T = R^2/M^2$ and $\boldsymbol{\sigma}^0$ the scaling coefficients which minimize T (found by gradient descent). Then

$$T(\sigma_1^0, \ldots, \sigma_{p-1}^0, 0, \sigma_{p+1}^0, \ldots, \sigma_d^0) \approx T(\boldsymbol{\sigma}^0) - \sigma_p^0 \underbrace{\left.\frac{\partial T}{\partial \sigma_p}\right|_{\boldsymbol{\sigma}=\boldsymbol{\sigma}^0}}_{=0} + \frac{(\sigma_p^0)^2}{2} \left.\frac{\partial^2 T}{\partial \sigma_p^2}\right|_{\boldsymbol{\sigma}=\boldsymbol{\sigma}^0}.$$

Thus, instead of discarding the features with small scaling factors σ_p, we discard the features with the smallest values of

$$(\sigma_p^0)^2 \left.\frac{\partial^2 T}{\partial \sigma_p^2}\right|_{\boldsymbol{\sigma}=\boldsymbol{\sigma}^0}. \tag{5.4}$$

The explicit approach is more accurate but more time consuming since it requires one SVM training per component. It should be used only when the number of features left is small. Also, in this approach, it would be possible to use the span-estimate (3.11) instead of R^2/M^2 to compute the score associated with each feature.

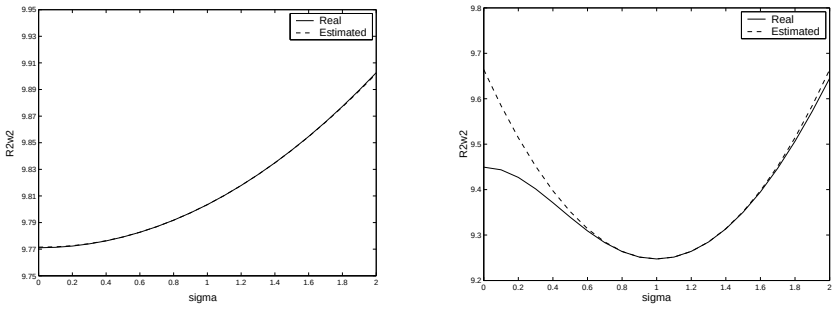


Figure 5.1: Two different examples of the behavior of R^2/M^2 as a function of a scaling factor σ_i. In the example on the left, it is easy to see that this feature should be discarded since R^2/M^2 decreases when σ_i goes to 0. On the right side, the situation is more complex: the gradient descent algorithm found $\sigma_i = 1$ as the optimal scaling factor and removing this feature will result in an increase of R^2/M^2 which can be estimated through the curvature at the optimum, as in equation (5.4).

Non-linear RFE

Although the RFE algorithm presented at the beginning of this section was designed for linear SVMs, the authors of [Guyon et al., 2002] proposed the following generalization of RFE to the non-linear case:

> *For each feature p, compute the kernel $K^{(p)}$ with feature p being removed, keep the same $\boldsymbol{\alpha}^0$ and assign to this feature the score $\sum_{ij} \alpha_i^0 \alpha_j^0 y_i y_j K^{(p)}(\mathbf{x}_i, \mathbf{x}_j)$*

We will include this algorithm in our experiments.

5.1.3 Experiments

Toy data

In the two following artificial datasets our objective was to assess the ability of the algorithm to select a small number of target features in the presence of irrelevant and redundant features.

Linear problem Six dimensions of 202 were relevant. The probability of $y = 1$ or -1 was equal. The first three features $\{x_1, x_2, x_3\}$ were drawn as $x_i = y\mathcal{N}(i, 1)$ and the second three features $\{x_4, x_5, x_6\}$ were drawn as $x_i = \mathcal{N}(0, 1)$ with a probability of 0.7, otherwise the first three were drawn as $x_i = \mathcal{N}(0, 1)$ and the second three as $x_i = y\mathcal{N}(i - 3, 1)$. The remaining features are noise $x_i = \mathcal{N}(0, 20)$, $i = 7, \ldots, 202$.

Nonlinear problem Two dimensions of 52 were relevant. The probability of $y = 1$ or -1 was equal. The data are drawn from the following: if $y = -1$ then $\{x_1, x_2\}$ are drawn from $\mathcal{N}(\mu_1, \Sigma)$ or $\mathcal{N}(\mu_2, \Sigma)$ with equal probability, $\mu_1 = \{-\frac{3}{4}, -3\}$ and $\mu_2 = \{\frac{3}{4}, 3\}$ and $\Sigma = I$, if $y = 1$ then $\{x_1, x_2\}$ are drawn again from two normal distributions with equal probability, with $\mu_1 = \{3, -3\}$ and $\mu_2 = \{-3, 3\}$ and the same Σ as before. The rest of the features are noise $x_i = N(0, 20)$, $i = 3, \ldots, 52$.

In the linear problem the first six features have redundancy and the rest of the features are irrelevant. In the nonlinear problem all but the first two features are irrelevant.

We used a linear SVM for the linear problem and a second order polynomial kernel of degree 2 for the nonlinear problem. For the filter methods and the SVM with feature selection we selected the 2 best features.

Two sets of comparisons have been carried out on the artificial datasets. The first one is taken from [Weston et al., 2000] and compares

- Standard SVMs

- Filter methods followed by SVM training. The three filter methods chose the m largest features according to
  - Pearson correlation coefficient
  - Fisher criterion score, $F(p) = \left| \frac{\mu_p^+ - \mu_p^-}{\sqrt{{\sigma_p^+}^2 + {\sigma_p^-}^2}} \right|$, where $\mu_p^\pm$ is the mean value for the p-th feature in the positive and negative classes and $\sigma_p^{\pm 2}$ is the standard deviation.
  - Kolmogorov-Smirnov test $\mathrm{KS}_{tst}(p) = \sqrt{n} \sup\left(\hat{P}\{X \le f_p\} - \hat{P}\{X \le f_p, y = 1\}\right)$, where f_p denotes the p-th feature from each training example, and $\hat{P}$ is the corresponding empirical distribution.

  The Pearson coefficients and Fisher criterion cannot model nonlinear dependencies.
- Gradient descent on R/M^2 and backward selection, as explained in the previous section.
- Forward selection on the span-estimate.

The results are shown in Figure (5.2) for various training set sizes, taking the average test error on 500 samples over 30 runs of each training set size. The Fisher score (not shown in graphs due to space constraints) performed almost identically to correlation coefficients.

In both problems standard SVMs perform poorly: in the linear example using $n = 500$ points one obtains a test error of 13% for SVMs, which should be compared to a test error of 3% with $n = 50$ using our methods. Our SVM feature selection methods also outperformed the filter methods, with forward selection being marginally better than gradient descent. In the nonlinear problem, among the filter methods only the Kolmogorov-Smirnov test improved performance over standard SVMs.

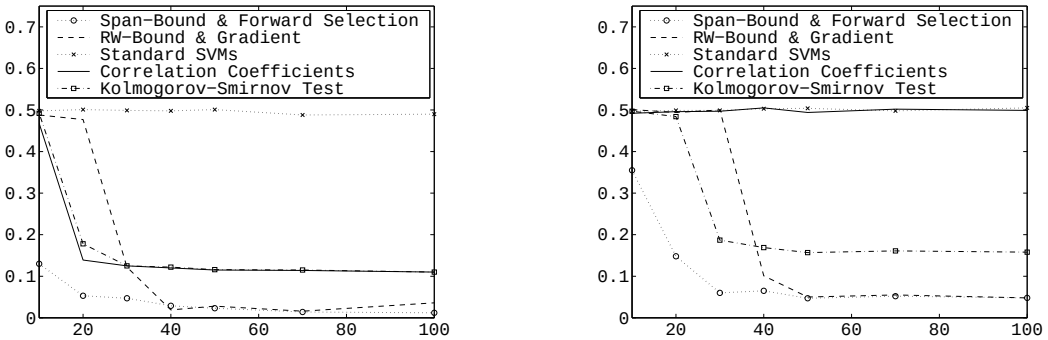


Figure 5.2: A comparison of feature selection methods on (a) a linear problem and (b) a nonlinear problem both with many irrelevant features. The x-axis is the number of training points, and the y-axis the test error as a fraction of test points.

In a second set of experiments, we compared a refined version of the gradient descent on R^2/M^2 and RFE. The algorithm we propose is as before a backward elimination combined with the search for optimal scaling factors, but it has two regimes depending on number of features left:

Coarse (more than 10 features): at each iteration half of the features having the smallest score (5.4) are discarded,

Accurate (less than 10 features): a feature is discarded at a time and it is chosen such that the resulting span-estimate is minimized.

The results for the linear problem are shown in figure 5.3 and for the non-linear one in figure 5.4.

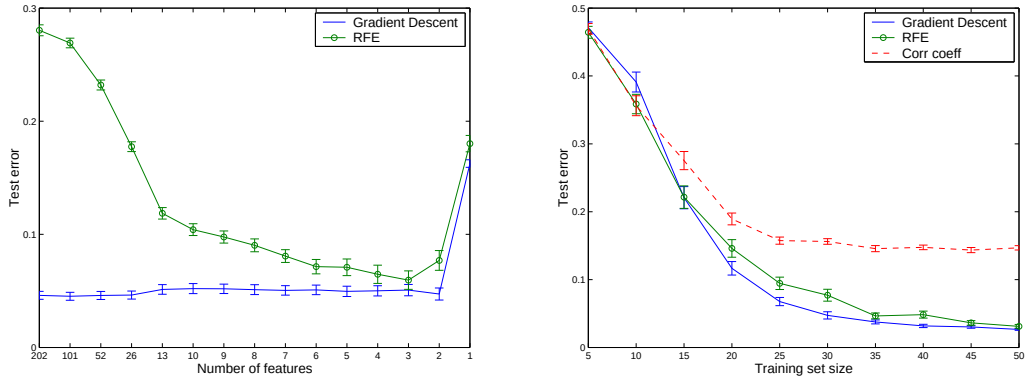


Figure 5.3: Linear toy problem. Left: evolution of the test error as a function of the number of iterations (aka features) in the backwards elimination (30 training samples). Right: test error as a function of training samples with 2 features selected. Results averaged on 100 trials.

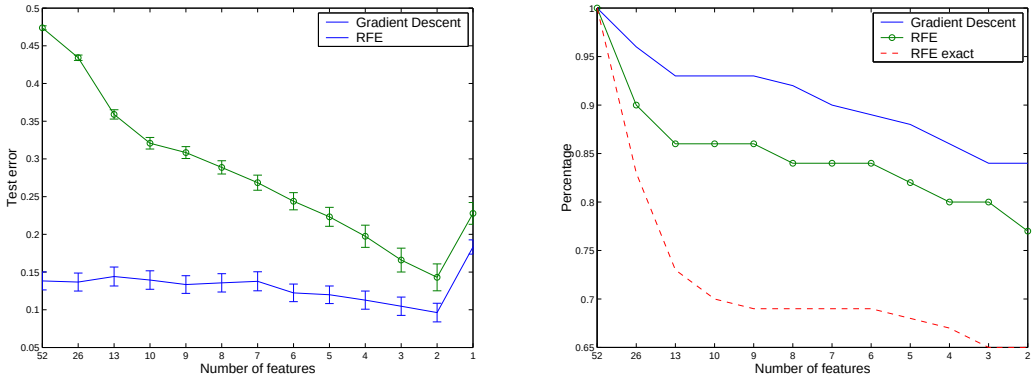


Figure 5.4: Non-linear toy problem. The x-axis is the number of features left at each iteration of the backwards feature elimination. The y-axis is either the test error (left) or the percentage of time the two relevant features are selected (right).

Our approach and the RFE algorithm give the best results compared to the other methods tested in the first set of experiments, RFE being slightly worse.

On those experiments, it seems that the backwards elimination is not even necessary: in the linear problem (figure 5.3, left), an SVM trained with the 202 features results in an average test error of 28%, but when the scaling factors are optimized the test error drops to 4.6%. It urns out that after optimization, most of the scaling factors are near zero and up to a certain point, the backwards elimination is a trivial task since it just removes those features.

As an aside, we would like to point out a strange phenomenom in the RFE algorithm. The non-linear version of RFE had the following motivation: suppose that when a feature p is removed and the new kernel $K^{(p)}$ recomputed, the optimal value of the Lagrange multipliers $\boldsymbol{\alpha}^0$ does not change. Then, under this assumption, the norm of the new weight vector would be

$$\left\|\tilde{\mathbf{w}^p}\right\|^2 = \sum_{i,j=1}^{n} \alpha_i^0 \alpha_j^0 y_i y_j K^{(p)}(\mathbf{x}_i, \mathbf{x}_j).$$

The idea is that an irrelevant feature should behave such that $\|\mathbf{w}\|^2 - \left\|\tilde{\mathbf{w}^p}\right\|^2$ is small. This algorithm can be viewed has an extension of the linear case since for linear SVMs, we have $\|\mathbf{w}\|^2 - \left\|\tilde{\mathbf{w}^p}\right\|^2 = w_p^2$. The quantity $\left\|\tilde{\mathbf{w}^p}\right\|^2$ is easy to compute since the original $\boldsymbol{\alpha}^0$ is used. However, we wanted to see what would be the change in performance of the algorithm if the vector $\boldsymbol{\alpha}^0$ is recomputed after a feature has been removed and thus compute the exact value $\|\mathbf{w}^p\|^2$. This is the line *RFE exact* in the plot on the right hand side of figure 5.3. Surprisingly, there is a decrease in performance, which means that there is something missing in the original justification of the RFE algorithm. The reasons for the good performances of this algorithm are thus still unclear.

Real-life data

Cancer morphology classification For DNA microarray data analysis, one needs to determine the relevant genes in discrimination as well as discriminate accurately. We look at two leukemia discrimination problems [Golub et al., 1999; Mukherjee et al., 1999] and a colon cancer problem [Alon et al., 1999] (see also Guyon et al. 2002 for a treatment of both of these problems).

The first problem was classifying myeloid and lymphoblastic leukemias based on the expression of 7129 genes. The training set consists of 38 examples and the test set of 34 examples. Using all genes a linear SVM makes 1 error on the test set. Using 20 genes 0 errors are made for the gradient descent method on R^2/M^2 and 3 errors are made using the Fisher score. Using 5 genes 1 error is made for R^2/M^2 and 5 errors are made for the Fisher score. The method of [Golub et al., 1999] performs comparably to the Fisher score.

The second problem was discriminating B versus T cells for lymphoblastic cells [Golub et al., 1999]. Standard linear SVMs make 1 error for this problem. Using 5 genes 0 errors are made using the gradient descent on R^2/M^2 and 3 errors are made using the Fisher score.

In the colon cancer problem [Alon et al., 1999] 62 tissue samples probed by oligonucleotide arrays contain 22 normal and 40 colon cancer tissues that must be discriminated based upon the expression of 2000 genes. Splitting the data into a training set of 50 and a test set of 12 in 50 separate trials we obtained a test error of 13% for standard linear SVMs. Taking 15 genes for each feature selection method we obtained 12.8% for R^2/M^2, 17.0% for Pearson correlation coefficients, 19.3% for the Fisher score and 19.2% for the Kolmogorov-Smirnov test. Our method is only worse than the best filter method in 8 of the 50 trials.

5.1.4 Discussion

We have introduced tools to perform feature selection for SVMs. The two main ingredients are the search of optimal scaling factors and the backward elimination scheme. The tools presented here are quite flexible and can lead to powerful feature selection algorithms. They can handle high dimensional datasets, unlike other existing wrapper methods which are computationally unfeasible for such datasets.

However, we would like to point out that we introduced tools and ideas on how to perform feature selection. The way to combine optimally those tools in order to have a general and robust feature selection method is still unclear. On the other hand, the RFE algorithm is straightforward to implement and even if it has not a solid justification, it provides very good results. For this reason, we recommend this latter to the practitioner.

Finally, even if SVMs have been designed to handle high-dimensional data, the experiments showed that they can indeed suffer the curse of dimensionality when many features are irrelevant. Feature selection becomes a necessary step in this case.

5.2 Data cleaning

5.2.1 Introduction

In this section we describe some investigations into the effectiveness of data cleaning techniques which have been developed in [Weston et al., 2001]. The problem of data cleaning has been addressed in the machine learning community by Guyon et al. [1994]; Matic et al. [1992] in the context of character recognition.

Data cleaning is the problem of identifying mislabeled or meaningless data points. During data collection, several kinds of errors can be introduced: e.g hardware failures can cause the insertion of meaningless patterns or human failures can cause the insertion of mislabeled patterns. Removal of these patterns (or correction of *mislabeling*) might improve the performance of the classifier, and in the case of feature selection algorithms leads to more meaningful features. We will focus on data cleaning applications in biclass pattern recognition problems, although the reasoning should also apply to other domains.

It is also important to stress that not all outliers are mislabeled or meaningless patterns. They can be well labeled ambiguous patterns or atypical very informative patterns. This is what makes the tasks challenging and sheds doubt on the possibility of doing "automatic" cleaning. As we will show in experiments, manual cleaning can improve performance substantially (by removing patterns known by an oracle to be bad). However even automatic cleaning yields improvements (not large in our experiments, but in some experiments on character recognition [Matic et al., 1992], automatic cleaning was almost as good as manual cleaning, which is quite surprising). Moreover, even removing well labeled examples can improve performance. We believe these issues contribute to the non-triviality of the problem.

In pattern recognition we are usually given training data $\mathbf{x}_i \in \mathbb{R}^d, i = 1, \ldots, n$ which can be partially corrupted with noise, and thus still be at least partially useful. In this case the learning algorithm should take this into account. Many learning algorithms are designed to deal with this problem. Often algorithms assume a model of the noise, e.g the data and the noise are drawn from two probability distributions $p(\mathbf{x}, y)$ and $\xi(\mathbf{x})$ to give training data of the form $\mathbf{x}_i + \xi_i$, $i = 1, \ldots, n$. This is actually the main idea behind the VRM induction principle.

In the problem of data cleaning one is interested in identifying outliers: the situation where some of the data is completely meaningless or mislabeled, which does not easily fit into the usual model described above. The meaningless data may not be from a single distribution, but it can just be a single accident by the specialist who collected the data. In this case we would like the training point to be removed from the database.

5.2.2 The Problem of Data Cleaning

The data cleaning problem can be understood in at least two different ways: the problem of identifying outliers and the problem of minimizing prediction error.

These two descriptions also correspond to two possible courses of action. In the first one might wish to refer the potentially mislabeled or meaningless examples to the labeler (for example, the specialist who created the data, or someone further along the line of data collection and assembly) and query if this point is in fact mislabeled. The ultimate aim of this is then either to correct mislabelings or to increase generalization ability of the learning algorithm.

The second course of action is *automatic data cleaning* where the learning algorithm chooses which data points are causing generalization ability to decrease and removes them without intervention from the user. This can be seen as a "dual" to the *feature selection problem* where one wishes to remove "bad features" not "bad examples". We thus will also refer to the problem of data cleaning as *example selection*. Note that generalization ability can be improved by removing examples which are not necessarily mislabeled or meaningless.

In the spirit of usual solutions to the feature selection problem one can reduce both problems to two separate subproblems:

1. ranking the data points according to the likelihood that they should be removed; and

2. selecting how many of the patterns should be removed

The second problem can be seen as analogous to model selection. In this section we will only consider the problem of identifying incorrect patterns and the subproblem of ranking the data points.

5.2.3 Experimental setup

We would like to test the ability of algorithms to detect mislabeled patterns. To this end our experimental setting is to deliberately mislabel training points and test if the algorithms can detect it. This mislabeling process could include many mislabeled points but in our experiments we only mislabel a single point at a time.

Algorithms All of the algorithms that we test are of the following form: they are given as input a training set S_n and they output a score s_i for the n data points. The score indicates the likelihood of being a mislabeled pattern, with greater likelihood being assigned a higher score. Then the points are ranked in descending order according to this score.

Scoring algorithms Measuring the success of a data cleaning algorithm even when there is only a single mislabeled point is not obvious. Taking the mean rank of the (single) mislabeled points is an obvious measure, however it can pay too much attention to a few large scores. For example, if one algorithm predicts the mislabeled points on 9 out of 10 runs with rank 1 but on the other run it assigns a rank of 50 it obtains a mean of 5.9. If another algorithm always gives a rank of 5 it has a mean of 5 however we think the first algorithm is more useful for detecting outliers. If one is referring the first one or two points to the data collector for verification then the first algorithm will find outliers whereas the second will not. In [Weston et al., 2001] several measurements of error have been recorded, but for the sake of simplicity we report here only two measurements. Over k runs, we record

- number of times the mislabeled point has been ranked 1,

- "trimmed" mean: if r_i is the rank of the mislabeled point in the run i, we compute $\frac{1}{k}\sum_i \min(c, r_i)$. That is, we take the mean rank of the mislabeled points, but for ranks greater than c we decrease their contribution to the mean. By doing this, we attempt to concentrate the score on the first few rankings. We use $c = 10$ and in tables of results we refer to this score as "mean ≤ 10".

Data generation Our setup is therefore as follows. For the toy data we randomly draw 100 training sets of a fixed size and then flip the label of a single training example in each set. We then score the data cleaning algorithms on each dataset according to the ranking of the mislabeled example in each set, and take the mean score. In real datasets that are of a small size (e.g microarray data) we cannot randomly draw many independent training sets. We therefore use the following procedure: we flip the label of each example in turn so that one has n copies of the original training set but each with a single mislabeled point. Then the data cleaning algorithms are scored according to the ranking of the mislabeled example in each set, and takes the mean score.

5.2.4 Algorithms for Data Cleaning

Several algorithms have been compared in [Weston et al., 2001]. We report here the results of 4 of the most significant ones.

SVM-ξ This algorithm records the distance from the "correct" side of the margin (ξ_i) of each training point. The ranking is thus given by the distance of each point from the "correct" side of the margin, largest ranked first.

- Choose a value of the soft margin parameter C.
- Train the classifier $[\mathbf{w}, b] = \mathrm{SVM}(S_n, C)$.
- Calculate $\xi_i = 1 - y_i(w \cdot \mathbf{x}_i + b)$ for all i and assign $s_i = \xi_i$.

SUB-ξ This algorithm sub-samples the data many times, each time training an SVM and recording the average distance from the "correct" side of the margin (ξ_i) of each test point. The ranking is thus given by the average distance of each point from the "correct" side of the margin, largest ranked first.

- Choose a value of the soft margin parameter C, the number of sub-sample runs p and the sub-sampling size q.
- Perform p SVM trainings, each one on a random subset of q examples.
- Assign to each point its average ξ_i (on the splits where this point was in the test set).

LOO-ξ This algorithm performs leave-one-out, each time training an SVM and recording the distance from the "correct" side of the margin (ξ_i) of the left out training point. The ranking is thus given by the distance of each left out point from the "correct" side of the margin, largest ranked first.

- Choose a value of the soft margin parameter C.
- For all i, train an SVM with the point $\mathbf{x}_i$ left out, $[\mathbf{w}_i, b_i] = \mathrm{SVM}(S_n \setminus \{(\mathbf{x}_i, y_i)\}, C)$.
- Assign $s_i = 1 - y_i(\mathbf{w}_i \cdot \mathbf{x}_i + b_i)$.

GRAD-C This algorithm tries to learn which data point is an outlier by assigning a variable C_i for each training point and minimizes R^2/M^2 using the ridge trick described in section 4.1. Thus the variables C_i are just kernel parameters which are optimized by gradient descent as proposed in chapter 4.

More precisely the goal is to find

$$\mathbf{C}^0 = \arg\min_{\mathbf{C}\geq\mathbf{0}} \max_{\boldsymbol{\alpha}} \{R^2(\mathbf{C})W^2(\boldsymbol{\alpha}, \mathbf{C}), \quad \alpha_i \geq 0, \quad \sum_i \alpha_i y_i = 0\}$$

where

$$W^2(\boldsymbol{\alpha}, \mathbf{C}) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \left(K(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{C_i}\delta_{ij} \right) \tag{5.5}$$

and

$$R^2(\mathbf{C}) = \frac{1}{n}\sum_i \left(K(\mathbf{x}_i, \mathbf{x}_i) + \frac{1 - 1/n}{C_i} \right) - \frac{1}{n^2}\sum_{i,j} K(\mathbf{x}_i, \mathbf{x}_j). \tag{5.6}$$

One then assign $s_i = 1/C_i$.

SVM-GOD Finally, we also compared the results to an oracle algorithm which incorporates knowledge about the true labels. It returns the largest ξ_i from a soft margin SVM trained on a second training set with the same size as the original one but containing only true labels (this is a different set, not just the same set with corrected labels, only the size of the sets is equal).

	rank 1	mean ≤ 10
SVM-ξ C	36	3.68
SUB-ξ	42	3.80
SUB-ξ C	40	3.68
LOO-ξ	35	4.18
LOO-ξ C	37	3.73
GRAD-C	38	3.86
SVM-GOD	41	3.39

Figure 5.5: Comparison of algorithms on toy data. The letter C means that the algorithm uses a soft margin SVM whose value C is chosen on a validation set. Algorithms have been run 100 times and that is the maximum value of the rank score.

Comments on the algorithms

The SVM-ξ method is the most intuitive method: if a point is mislabeled, it will be in all likelihood wrongly classified. However this method might be sensitive to the choice of the soft margin parameter C and one might need sophisticated techniques to choose a good value for C.

The subsampling algorithm SUB-ξ is also sensitive to the choice of C, but works also in the hard margin case. However, it is computationally expensive since it requires several SVM trainings.

The LOO-ξ algorithm can be seen as a special case of the sub-sampling one with $p = n$ and $q = n-1$, but the span-estimate (theorem 3.8) is a fast way to estimate the slack of a point after it has been removed from the training set.

As we did for feature selection, let us analysis what are the values C_i after one gradient step in the method GRAD-C. From equation (5.6), we have

$$\frac{\partial R^2}{\partial C_p} = -\frac{1 - 1/n}{C_p^2}$$

and from equation (5.5) and lemma 4.1,

$$\frac{\partial W^2}{\partial C_p} = \frac{(\alpha_p^0)^2}{2C_p^2}.$$

If all variables C_p are initialized to the same value, after one gradient step, the ranking of the points given by the vector $\mathbf{C}$ will be the same as the one given by α_p^0/C_p which turns out to be equal to ξ_p through equation (4.2).

5.2.5 Experiments

Toy data

We generated a toy problem to first test the methods. The problem consists of two Gaussian distributions in $d = 100$ dimensions with variance $1.5/\sqrt{d}$, one for each class label, drawing 30 training points randomly for each class. We generated 100 of such datasets and flipped the label of the 30^{th} data point (to help prevent bias in the sorting algorithm if there are equal ranks). The problem is then to detect that this data point is mislabeled. We compared all the algorithms described in the previous section in both hard margin and soft margin settings. The results are shown in figure 5.5.

From these experiments we conclude that these algorithms perform similarly and that finding a good value for C improves the results. They are not so far from SVM-GOD which suggests that if you stay within the SVM learning framework to perform data-cleaning you cannot do a lot better than these algorithms.

We also performed an experiment measuring the test error obtained when using a soft margin SVM with the mislabeled example in the training set which gave 14.4% error ($C = 0.53$, chosen on the

	rank 1	mean ≤ 10
SVM-ξ C	13	4.82
SUB-ξ	12	4.91
LOO-ξ	10	5.43
GRAD-C	9	4.98

Figure 5.6: Comparison of algorithms on the colon cancer dataset. The maximum rank score is 62.

same validation set as testing) over 100 runs. Without the mislabeled point an SVM gave 12.9% error ($C = 0.34$). Thus there is a significant improvement when one removes the outlier. This was not obvious to us before as we believed a soft margin SVM would produce close to optimal results. We then tried to remove one mislabeled point using the SVM-ξ algorithm giving an error rate of 14.1% ($C = 0.19$): performance improved, but unfortunately the gain was not as high as possible (12.9%). However, we suggest an automatic cleaning method where we assume that only one example is mislabeled and we systematically remove the top ranked example and retrain. Since we have a substantial number of cases (roughly two thirds) in which the top example is not mislabeled, maybe systematically removing it is not the best solution. However, finding a good automatic cleaning prescription is important, particularly because in reality we don't know how many if at all are mislabeled. We do not deal with these issues here and concentrate instead on the identification problem rather than the automatic cleaning one.

Real data

We performed experiments on a colon cancer problem [Alon et al., 1999]. For this problem, 62 tissue samples probed by oligonucleotide arrays contain 22 normal and 40 colon cancer tissues that must be discriminated based upon the expression of 2000 features. Note that for real data, it is possible that in addition to the points that we deliberately mislabeled, some of them are already mislabeled in the first place.

The results are shown in figure 5.6. The C value of the SVM-ξ method has been chosen has the median of the α values of a hard margin SVM. Again the different algorithms have a similar behavior. More details about those experiments can be found in [Weston et al., 2001].

5.3 Fisher kernel

We propose here an application of the previous chapter to the Fisher kernel [Jaakkola and Haussler, 1998; Jaakkola et al., 1999a], but we did not test the proposed method.

The Fisher kernel has been proposed in [Jaakkola and Haussler, 1998] to incorporate a generative model in a discriminative classifier. Suppose that a generative model $P(\mathbf{x}|\boldsymbol{\theta})$ depending on some parameters $\boldsymbol{\theta}$ is given. Those parameters are estimated by maximizing the likelihood

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log P(\mathbf{x}_i|\boldsymbol{\theta}).$$

Then for each training point, the *Fisher score* is computed

$$U_{\mathbf{x}_i} = \nabla_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} \log P(\mathbf{x}_i|\boldsymbol{\theta}),$$

as well as the empirical *information matrix*,

$$I = \frac{1}{n} \sum_{i=1}^{n} U_{\mathbf{x}_i} U_{\mathbf{x}_i}^{\top}.$$

The new representation of point $\mathbf{x}_i$ is given by

$$\tilde{\mathbf{x}_i} = I^{-1/2} U_{\mathbf{x}_i},$$

on which any standard kernel can be applied.

The effectiveness of this method has been demonstrated in [Jaakkola et al., 1999a] on a bioinformatic application where the generative model was a Hidden Markov Model.

A natural application of the previous chapter is to consider the vector $\boldsymbol{\theta}$ as an hyperparameter and to optimize it by minimizing R^2/M^2. This differs from the original approach where $\boldsymbol{\theta}$ is found by the maximization of the likelihood.

To perform this gradient descent on R^2/M^2, one has to explicit the derivative of the kernel function with respect to the vector $\boldsymbol{\theta}$ which involves the calculation of the second derivatives

$$\frac{\partial^2 \log P(\mathbf{x}_i|\boldsymbol{\theta})}{\partial\theta_p \partial\theta_q}.$$

This approach has not been tested in practice and we do not know if it would give an improvement in performance.

5.4 Scaling factors and ridge regression

In general, it is useful to find automatically scaling factors since the input components can be measurements of different nature and one has to weight the input features accordingly. We saw in section 4.5.3 how to implement the automatic selection of scaling factors for SVMs in the context of classification.

We present in this section a similar approach, but in the context of regression. More precisely, we will see how to find by gradient descent the scaling factors which minimize the leave-one-out error of the ridge regression algorithm. A similar approach has been proposed in [Goutte and Larsen, 1998], where the authors find the scaling factors by minimizing the cross-validation error of the Nadaraya-Watson estimator. More recently, it has been shown that one can estimate the shape of an RBF kernel by minimizing an entropy-based measure [Ormoneit and Hastie, 1999]

5.4.1 Ridge regression

Consider the set of functions

$$f(\mathbf{x}, \boldsymbol{\alpha}) = \sum_{i=1}^{p} \alpha_i \varphi_i(\mathbf{x}),$$

where φ_i are basis functions. In kernel methods, one has usually $p = n$ and $\varphi_i = K(\mathbf{x}_i, \cdot)$.

The Ridge Regression algorithm [Hoerl and Kennard, 1970] proposes to minimize the following functional

$$R_{emp}(\boldsymbol{\alpha}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i, \boldsymbol{\alpha}))^2 + \gamma \left\|\boldsymbol{\alpha}\right\|^2 \tag{5.7}$$

where γ is a fixed positive constant, called the regularization parameter. The minimum is given by the vector of coefficients

$$\boldsymbol{\alpha}^0 = (K^\top K + \gamma I)^{-1} K^\top Y$$

where

$$Y = (y_1, \ldots, y_n)^\top,$$

and K is a matrix with elements

$$K_{ij} = \varphi_j(\mathbf{x}_i), \quad i = 1, \ldots, n, \quad j = 1, \ldots, p.$$

5.4.2 Leave-one-out error

A nice property of the Ridge Regression algorithm is that its leave-one-out error has a closed form [Vapnik, 1982]. Indeed, denoting

$$A_\gamma = K^\top K + \gamma I$$

the error incurred by the leave-one-out procedure is

$$T_{loo} = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{y_i - k_i^\top A_\gamma^{-1} K^\top Y}{1 - k_i^\top A_\gamma^{-1} k_i} \right)^2 \tag{5.8}$$

where

$$k_i = (\varphi_1(\mathbf{x}_i) \ldots, \varphi_p(\mathbf{x}_i))^\top.$$

The numerator measures the deviation between the Ridge Regression estimates of the values at the training points from the true values, i.e. the training error. The denominator measures the correction to these estimates for the leave-one-out procedure.

5.4.3 Optimizing the scaling factors

Now suppose that $\varphi_i = K(\mathbf{x}_i, \cdot)$, where K is an RBF kernel with diagonal covariance matrix given by equation (4.15),

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(- \sum_{i=1}^{d} \frac{(x_i - y_i)^2}{2\sigma_i^2} \right).$$

The hyperparameters of the ridge regression algorithms are thus $\gamma, \sigma_1, \ldots, \sigma_d$, which can be optimized by a gradient descent on T_{loo} (5.8). It is indeed possible to compute analytically

$$\frac{\partial T_{loo}}{\partial \sigma_p} \quad \text{and} \quad \frac{\partial T_{loo}}{\partial \gamma}.$$

To minimize the leave-one-out error T_{loo}, we used the optimization toolbox of Matlab and to avoid to add the constraints $\sigma_p \geq 0$ and $\gamma \geq 0$, we performed the optimization on $\log \sigma_p$ and $\log \gamma$.

5.4.4 Experiment

We conducted some experiments on a database provided by a pharmaceutical company. It consists of 3000 samples that we divided in 2000 for training, 500 for validation and 500 for test. There are 362 input components of three different types: continuous, categorical or binary. For this reason, it should be helpful to have different scaling factors for different features.

For computational reasons, we did not take one basis function per training point, but we chose the first $p = 1000$ training points as support for the basis functions.

The starting point of the gradient descent was set using hand-tuned values of $\sigma_1 = \ldots \sigma_d = \sigma$ and γ: the values for σ and γ were chosen on a grid in order to minimize the validation error.

The evolution of the leave-one-out error and the validation error as a function of the number of gradient steps is reported in figure 5.7. The leave-one-out error and the validation error are very well correlated. Note that toward the end of the optimization, there is a slight increase in the validation error which might be due to an overfitting phenomenon.

The optimization of the scaling coefficients yielded a significant reduction in the mean squared error from 5.5 to 4.

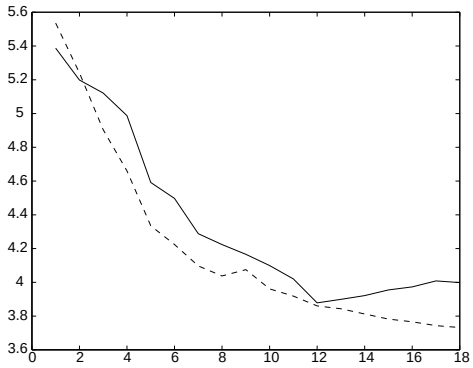


Figure 5.7: Evolution of the mean squared error on a validation set (solid line) and of the leave-one-out error (5.8) (dashed line) as a function of the number of gradient steps.

Part III

Prior knowledge and unlabeled data

Chapter 6

Invariant Support Vector Machines

The choice of an SVM kernel corresponds to the choice of a representation of the data in a feature space and, to improve performance, it should therefore incorporate prior knowledge such as known transformation invariances. We propose a technique which extends earlier work and aims at incorporating invariances in nonlinear kernels [Chapelle and Schölkopf, 2001]

6.1 Introduction

In some classification tasks, an a priori knowledge is known about the invariances related to the task. For instance, in image classification, we know that the label of a given image should not change after a small translation or rotation.

More generally, we assume we know a local transformation $\mathcal{L}_t$ depending on a parameter t (for instance, a vertical translation of t pixels) such that any point $\mathbf{x}$ should be considered equivalent to $\mathcal{L}_t\mathbf{x}$, the transformed point. Ideally, the output of the learned function should be constant when its inputs are transformed by the desired invariance.

We consider here local invariances, which means that we try to find a decision function such that for small t and for all $\mathbf{x}_i$

$$f(\mathcal{L}_t\mathbf{x}_i) \approx f(\mathbf{x}_i).$$

It has been shown [Burges, 1999] that one can not find a non-trivial kernel which is globally invariant. For this reason, we consider here local invariances and for this purpose we associate at each training point $\mathbf{x}_i$ a *tangent vector* $d\mathbf{x}_i$,

$$\begin{aligned} d\mathbf{x}_i &= \lim_{t\to 0} \frac{1}{t}\left(\mathcal{L}_t\mathbf{x}_i - \mathbf{x}_i\right) \\ &= \left.\frac{\partial}{\partial t}\right|_{t=0} \mathcal{L}_t\mathbf{x}_i \end{aligned}$$

In practice $d\mathbf{x}_i$ can be either computed by finite difference or by differentiation. Note that generally one can consider more than one invariance transformation. The multi-invariance case is usually a straight forward extension of the single one.

A common way of introducing invariances in a learning system is to add the perturbed examples $\mathcal{L}_t\mathbf{x}_i$ in the training set [Niyogi et al., 1998]. Those points are often called *virtual examples*. In the SVM framework, when applied only to the SVs, it leads to the *Virtual Support Vector* (VSV) method [Schölkopf et al., 1996].

In the limit, this is equivalent to replacing each initial example by a distribution whose shape represents the desired invariances. This formulation naturally leads to a special case of the VRM induction principle described in section 1.3 in which the local density estimates $P_{\mathbf{x}_i}(\mathbf{x})$ are elongated in the direction of invariance.

An alternative to this is to modify directly the cost function by adding an adequate regularization term which takes into account the tangent vectors. This has been successfully applied to neural networks [Simard et al., 1998], giving the Tangent-Prop algorithm, and linear Support Vector Machines [Schölkopf et al., 1998a]. Those approaches are actually similar since we show in section 1.3 that VRM has strong connections with noise injection and regularization. Tangent-Prop has been formally proved to be equivalent to generating synthetic examples with infinitesimal deformations [Leen, 1995].

The aim of this chapter is to extend these methods to the case of nonlinear SVMs which will be achieved mainly by using the kernel PCA trick [Schölkopf et al., 1998b]. It is organized as follows. We first present the method proposed in [Schölkopf et al., 1998a] to train invariant linear SVMs in section 6.2. In section 6.3, we show how to extend it to the nonlinear case and finally experimental results are provided in section 6.6.

6.2 Invariances for linear SVMs

For linear SVMs, one wants to find a hyperplane whose normal vector $\mathbf{w}$ is as orthogonal as possible to the tangent vectors. This can be easily understood from the equality

$$f(\mathbf{x}_i + d\mathbf{x}_i) - f(\mathbf{x}_i) = \mathbf{w} \cdot d\mathbf{x}_i.$$

For this purpose, it has been suggested [Schölkopf et al., 1998a] to minimize the functional

$$(1-\gamma)\mathbf{w}^2 + \gamma \sum_{i=1}^{n} (\mathbf{w} \cdot d\mathbf{x}_i)^2 \tag{6.1}$$

subject to the constraints

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1.$$

The parameter γ trades off between normal SVM training ($\gamma = 0$) and full enforcement of the orthogonality between the hyperplane and the invariance directions ($\gamma \to 1$).

Let us introduce

$$C_\gamma = \left((1-\gamma)I + \gamma \sum_i d\mathbf{x}_i d\mathbf{x}_i^\top \right)^{1/2}, \tag{6.2}$$

the square root of the regularized covariance matrix of the tangent vectors.

The functional (6.1) then reads

$$\mathbf{w}^\top C_\gamma^2 \mathbf{w}.$$

Before showing how to minimize this functional, we will see that it can be derived within the VRM framework. Indeed, in this framework, it is possible to take invariances into account by choosing the local density estimate $P_{\mathbf{x}_i}$ to be a Gaussian kernel with a covariance matrix whose eigenvectors describe the tangent direction to the invariant manifold. The eigenvalues then represent the respective strengths of the selected invariances. If there is only one invariance, the covariance matrix around the point $\mathbf{x}_i$ is of the form:

$$C_i = \sigma^2 I + d\mathbf{x}_i d\mathbf{x}_i^\top.$$

We saw in section 2.3.2.0, that it is not obvious to turn the VRM optimization problem in a SVM-like one when the the covariance matrix are different for each training point. Instead, one can use the mean covariance matrix,

$$\sigma^2 I + \frac{1}{n} \sum_{i=1}^{n} d\mathbf{x}_i d\mathbf{x}_i^\top,$$

which is the same as (6.2).

In order to minimize (6.1), let us introduce $\tilde{\mathbf{w}} = C_\gamma \mathbf{w}$ and $\tilde{\mathbf{x}}_i = C_\gamma^{-1} \mathbf{x}_i$. We obtain the equivalent optimization problem:

$$\min_{\tilde{\mathbf{w}}} \tilde{\mathbf{w}}^2 \tag{6.3}$$

under constraints

$$y_i(\tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}_i} + b) \geq 1.$$

Here, we made use of the equality $C_\gamma^{-1}\mathbf{w} \cdot \mathbf{x}_i = \mathbf{w} \cdot C_\gamma^{-1}\mathbf{x}_i$, which is valid because C_γ is symmetric. Note also that C_γ is strictly positive definite (and thus invertible) if $\gamma < 1$. For this reason, in the rest of the chapter, we will assume $\gamma < 1$.

The optimization problem (6.3) is the standard SVM one where the training points $\mathbf{x}_i$ have been linearly preprocessed using the matrix C_γ^{-1}.

The output value of the decision function on a test point is :

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{w} \cdot \mathbf{x}_i + b \\ &= \tilde{\mathbf{w}} \cdot C_\gamma^{-1}\mathbf{x} + b \\ &= \sum_{i=1}^{n} \alpha_i y_i \; C_\gamma^{-1}\mathbf{x}_i \cdot C_\gamma^{-1}\mathbf{x} + b \end{aligned}$$

We also obtain the standard formulation of an SVM output, but where the test point is first multiplied by C_γ^{-1}.

To conclude, one could say that a linear invariant SVM is equivalent to a standard SVM where the input space has been transformed through the linear mapping

$$\mathbf{x} \rightarrow C_\gamma^{-1}\mathbf{x}.$$

In [Schölkopf et al., 1998a], it was shown that this method leads to significant improvements in linear SVMs, and to small improvements when used as a linear preprocessing step in nonlinear SVMs. The latter, however, was a hybrid system with unclear theoretical foundations. In the next section we show how to deal with the nonlinear case in a principled way.

6.3 Extension to the nonlinear case

In the nonlinear case, the data are first mapped into a high-dimensional feature space where a linear decision boundary is computed. To extend directly the previous analysis to the nonlinear case, one would need to compute the matrix C_γ in feature space,

$$C_\gamma = \left((1-\gamma)I + \gamma \sum_i d\Phi(\mathbf{x}_i) d\Phi(\mathbf{x}_i)^\top \right)^{1/2} \tag{6.4}$$

and the new kernel function

$$\tilde{K}(\mathbf{x}, \mathbf{y}) = C_\gamma^{-1}\Phi(\mathbf{x}) \cdot C_\gamma^{-1}\Phi(\mathbf{y}) = \Phi(\mathbf{x})^\top C_\gamma^{-2}\Phi(\mathbf{y}) \tag{6.5}$$

However, due to the high dimension of the feature space, it is impossible to do it directly. We propose two different ways for overcoming this difficulty.

6.3.1 Decomposition of the Gram matrix of the tangent vectors

In order to be able to compute the new kernel (6.5), we propose to diagonalize the matrix C_γ (eq 6.4) using the same approach as the one described in section 2.2. Remember that in that section, we showed how it was possible to diagonalize the feature space covariance matrix in the PCA subspace (spanned by a training sample) by computing the eigendecomposition of the Gram matrix of those points. Presently, instead of having a set of training points $\{\Phi(\mathbf{x}_i)\}$, we have a set of tangent vectors $\{d\Phi(\mathbf{x}_i)\}$ and a tangent covariance matrix

$$C^t = \sum_{i=1}^{n} d\Phi(\mathbf{x}_i) d\Phi(\mathbf{x}_i)^\top. \tag{6.6}$$

Let us consider the Gram matrix K^t of the tangent vectors:

$$\begin{aligned} K^t_{ij} &= d\Phi(\mathbf{x}_i) \cdot d\Phi(\mathbf{x}_j) \\ &= K(\mathbf{x}_i+d\mathbf{x}_i, \mathbf{x}_j+d\mathbf{x}_j) - K(\mathbf{x}_i+d\mathbf{x}_i, \mathbf{x}_j) \\ &\quad -K(\mathbf{x}_i, \mathbf{x}_j+d\mathbf{x}_j) + K(\mathbf{x}_i, \mathbf{x}_j) \qquad (6.7) \\ &= d\mathbf{x}_i^\top \frac{\partial^2 K(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_i \partial \mathbf{x}_j} d\mathbf{x}_j \qquad (6.8) \end{aligned}$$

This matrix K^t can be computed either by finite differences (equation 6.7) or with the analytical derivative expression given by equation (6.8). Note that for a linear kernel, $K(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y}$, and (6.8) reads $K^t_{ij} = d\mathbf{x}_i^\top d\mathbf{x}_j$, which is a standard dot product between the tangent vectors.

As in section 2.2, we write the eigendecomposition of K^t as $K^t = U\Lambda U^\top$, which enables us to find the expansion of the eigenvectors $(\mathbf{v}_1, \ldots, \mathbf{v}_n)$ of C^t corresponding to non-zero eigenvalues.

$$\mathbf{v}_p = \frac{1}{\sqrt{\lambda_p}} \sum_{i=1}^{n} U_{ip} d\Phi(\mathbf{x}_i). \qquad (6.9)$$

It is always possible to complete the orthonormal family $(\mathbf{v}_1, \ldots, \mathbf{v}_n)$ to an orthonormal basis of $\mathcal{H}$, $(\mathbf{v}_1, \ldots, \mathbf{v}_n, \mathbf{v}_{n+1}, \ldots, \mathbf{v}_{d_\mathcal{H}})$ with $d_\mathcal{H} = \dim \mathcal{H} \leq \infty$.

In this basis, C^t is diagonal

$$C^t = V \mathrm{D}(\lambda_1, \ldots, \lambda_n, 0, \ldots, 0) V^\top,$$

V being an orthonormal matrix whose columns are the $\mathbf{v}_i$. Remember from equation (6.5) that we are actually interested in C_γ^{-2}

$$C_\gamma^{-2} = V \mathrm{D}\left(\frac{1}{\gamma\lambda_1 + 1 - \gamma}, \ldots, \frac{1}{\gamma\lambda_n + 1 - \gamma}, \frac{1}{1-\gamma}, \ldots, \frac{1}{1-\gamma} \right) V^\top. \qquad (6.10)$$

To be able to compute the new kernel function (6.5), we need to determine the projection of a point in feature space $\Phi(\mathbf{x}_i)$ onto one of the vectors $\mathbf{v}_p$. From equation (6.9), we have, for $1 \leq p \leq n$,

$$\begin{aligned} \Phi(\mathbf{x}) \cdot \mathbf{v}_p &= \frac{1}{\sqrt{\lambda_p}} \sum_{i=1}^{n} U_{ip} (K(\mathbf{x}_i + d\mathbf{x}_i, \mathbf{x}) - K(\mathbf{x}_i, \mathbf{x})) \\ &= \frac{1}{\sqrt{\lambda_p}} \sum_{i=1}^{n} U_{ip} d\mathbf{x}_i^\top \frac{\partial K(\mathbf{x}_i, \mathbf{x})}{\partial \mathbf{x}_i}. \qquad (6.11) \end{aligned}$$

For $p > n$, however, the dot product $\Phi(\mathbf{x}) \cdot \mathbf{v}_p$ is unknown. The trick is to rewrite equation (6.5) as

$$\tilde{K}(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x})^\top \left(C_\gamma^{-2} - \frac{1}{1-\gamma} I \right) \Phi(\mathbf{y}) + \frac{1}{1-\gamma} K(\mathbf{x}, \mathbf{y}) \qquad (6.12)$$

From equation (6.10), one can easily see that the eigenvalues of $C_\gamma^{-2} - \frac{1}{1-\gamma} I$ associated with the eigenvectors $\mathbf{v}_p$ are zero for $p > n$.

Combining equations (6.10), (6.11) and (6.12) we finally get

$$\begin{aligned} \tilde{K}(\mathbf{x}, \mathbf{y}) &= \frac{1}{1-\gamma} K(\mathbf{x}, \mathbf{y}) + \\ &\quad \sum_{p=1}^{n} \Phi(\mathbf{x}) \cdot \mathbf{v}_p \left(\frac{1}{\gamma\lambda_p + 1 - \gamma} - \frac{1}{1-\gamma} \right) \Phi(\mathbf{y}) \cdot \mathbf{v}_p \\ &= \frac{1}{1-\gamma} K(\mathbf{x}, \mathbf{y}) + \sum_{p=1}^{n} \frac{1}{\lambda_p} \left(\frac{1}{\gamma\lambda_p + 1 - \gamma} - \frac{1}{1-\gamma} \right) \\ &\quad \left(\sum_{i=1}^{n} U_{ip}\ d\mathbf{x}_i^\top \frac{\partial K(\mathbf{x}_i, \mathbf{x})}{\partial \mathbf{x}_i} \right) \left(\sum_{i=1}^{n} U_{ip}\ d\mathbf{x}_i^\top \frac{\partial K(\mathbf{x}_i, \mathbf{y})}{\partial \mathbf{x}_i} \right) \end{aligned}$$

6.3.2 Decomposition of the Gram matrix of the input vectors

A drawback of the previous approach appears when one wants to deal with multiples invariances (i.e. more than one tangent vector per training point). Indeed, it requires to diagonalize the matrix K^t (cf equation 6.7) whose size is equal to the number of invariances.

If one wants to introduce multiple invariances, we propose an alternative method. The idea is to use directly the empirical kernel map described in section 2.2. Remember that a training sample of size n span a subspace E whose dimension is at most n. The empirical kernel map ψ (cf eq 2.21) gives in an orthonormal basis the components of a point $\Phi(\mathbf{x})$ projected on E.

From equation (2.16), it is easy to see that training a nonlinear SVM on $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ is equivalent to training a linear SVM on $\{\psi(\mathbf{x}_1), \ldots, \psi(\mathbf{x}_n)\}$ and thus, thanks to the nonlinear mapping ψ, we can work directly in the linear space E and use exactly the technique described for invariant *linear* SVMs (section 6.2). However the invariance directions $d\Phi(\mathbf{x}_i)$ do not necessarily belong to E. By projecting them onto E, some information might be lost. The hope is that this approximation will give a similar decision function to the exact one obtained in section 6.3.1.

Finally, the proposed algorithm consists in training an invariant linear SVM as described in section 6.2 with training set $\{\psi(\mathbf{x}_1), \ldots, \psi(\mathbf{x}_n)\}$ and with invariance directions $\{d\psi(\mathbf{x}_1), \ldots, d\psi(\mathbf{x}_n)\}$, where

$$d\psi(\mathbf{x}_i) = \psi(\mathbf{x}_i + d\mathbf{x}_i) - \psi(\mathbf{x}_i),$$

which can be expressed from equation (2.21) as

$$d\psi(\mathbf{x}_i)_p = \frac{d\mathbf{x}_i^\top}{\sqrt{\lambda_p}} \sum_{j=1}^{n} U_{jp} \frac{\partial K(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_i}.$$

Note that we can try to have an idea of how much information has been lost by projecting $d\Phi(\mathbf{x}_i)$ on E with the ratio

$$\frac{||d\psi(\mathbf{x}_i)||}{||d\Phi(\mathbf{x}_i)||} \leq 1 \tag{6.13}$$

6.4 Comparisons with the VSV method

One might wonder what is the difference between enforcing an invariance and just adding the virtual examples $\mathcal{L}_t\mathbf{x}_i$ in the training set. Indeed the two approaches are related and some equivalence can be shown [Leen, 1995].

So why not just add virtual examples ? This is the idea of the Virtual Support Vector (VSV) method [Schölkopf et al., 1996]. The reason is the following: if a training point $\mathbf{x}_i$ is far from the margin, adding the virtual example $\mathcal{L}_t\mathbf{x}_i$ will not change the decision boundary since neither of the points can become a support vector. Hence adding virtual examples in the SVM framework enforces invariance *only around the decision boundary*, which, as an aside, is the main reason why the virtual SV method only adds virtual examples generated from points that were support vectors in the earlier iteration.

One might argue that the points which are far from the decision boundary do not provide any information anyway. On the other hand, there is some merit in not only keeping the output label invariant under the transformation $\mathcal{L}_t$, but also the *real-valued output*. This can be justified by seeing the distance of a given point to the margin as an indication of its class-conditional probability [Platt, 2000]. It appears reasonable that an invariance transformation should not affect this probability too much.

The situation can be understood from the toy picture of figure 6.1. The point at the bottom right corner might not be a support vector, but it is still worth enforcing that the level curves around that point follow the direction given by its tangent vector.

6.5 Speeding-up

A straightforward way to implement this idea for the method described in section 6.3.2 is to select a random subset of the training points of size $m << n$ and to work in the subspace E' spanned by those

m points. In practice, this means that the kernel PCA map (2.21) can be computed only from those m training examples. Of course, there is a loss of information by projecting on E' instead of E, but if the eigenvalues of the kernel function decay quickly enough and m is large enough, then one can hope that with high probability $||P_E(\mathbf{x}) - P_{E'}(\mathbf{x})||$ is small (P being the projection operator).

More generally, any method used to speed-up kernel methods (see section 2.2.1) should be useful in our approach.

The method described in section 6.3.1 amounts to finding the linear invariant hyperplane in the subspace spanned by the tangent vectors (in feature space) whereas the one presented in section 6.3.2 amounts to doing the same thing but in the subspace spanned by the training points. More generally, one could consider an hybrid method where the linear invariant hyperplane is constructed in the linear space spanned by a subset of the tangent vectors and a subset of the training points. This will be the topic of further research.

6.6 Experiments

In our experiments, we compared a standard SVM with several methods taking into account invariances:

- Standard SVM with virtual examples (cf. the VSV method [Schölkopf et al., 1996]) [VSV]
- Invariant SVM as described in section 6.3.1 [ISVM]
- Invariant hyperplane in kernel PCA coordinates as described in section 6.3.2 [IH_{KPCA}]
- SVM trained with points preprocessed through the linear invariant hyperplane method [LIH]

The last method refers to [Schölkopf et al., 1998a] where the authors describe the invariant linear SVM (see also section 6.2). In their experiments, they linearly preprocessed the training and test points using C_γ^{-1} (equation 6.2). After, instead of training a linear SVM, they trained a nonlinear SVM. Even though there is no guarantee for such an approach to work, experimental results on the NIST set showed a slight improvement over standard nonlinear SVM.

Experiments have been carried on a toy problem and a digit recognition dataset.

6.6.1 Choice of γ

In all the methods described for training an invariant SVM, there is a parameter γ to choose, which is a trade-off between maximization of the margin and enforcement of the invariances (cf equation 6.1). One can set this parameter using a validation set or by gradient descent (see part 2).

In order to have a functional which is scale invariant with respect to the size of $d\mathbf{x}_i$ (in other words, the parametrization of $\mathcal{L}_t$, instead of minimizing the functional (6.1), we actually minimize

$$(1-\gamma)\mathbf{w}^2 + \gamma \sum_{i=1}^{n} \frac{(\mathbf{w} \cdot d\mathbf{x}_i)^2}{S},$$

with

$$S = \sum_{i=1}^{n} d\mathbf{x}_i^2.$$

This equivalent to minimizing functional (6.1) where γ is replaced by

$$\gamma \longleftarrow \frac{\gamma}{S + \gamma(1-S)}.$$

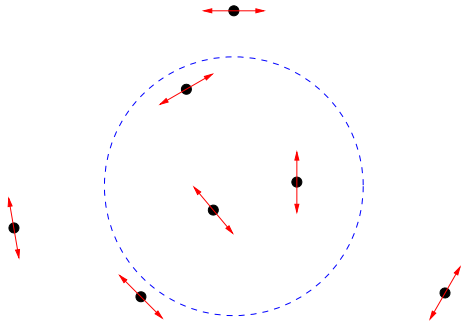


Figure 6.1: Toy problem: the true decision function is the circle. For each point we plot a tangent vector corresponding to a small rotation.

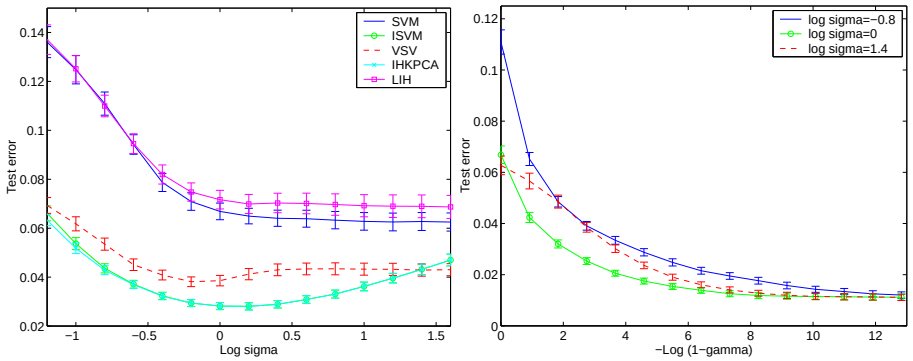


Figure 6.2: Left: test error for different learning algorithms plotted against the width of a RBF kernel and γ fixed to 0.9. Right: test error of $\mathrm{IH}_{\mathrm{KPCA}}$ across γ and for different values of σ. The test errors are averaged over the 100 splits and the error bars correspond to the standard deviation of the means.

6.6.2 Toy problem

The toy problem we considered is the following: the training data has been generated uniformly from $[-1, 1]^2$. The true decision boundary is a circle centered at the origin:

$$f(\mathbf{x}) = \mathrm{sign}(\mathbf{x}^2 - 0.7).$$

The a priori knowledge we want to encode in this toy problem is *local invariance under rotations*. Therefore, the output of the decision function on a given training point $\mathbf{x}_i$ and on its image $R(\mathbf{x}_i, \varepsilon)$ obtained by a small rotation should be as similar as possible. To each training point, we associate a tangent vector $d\mathbf{x}_i$ which is actually orthogonal to $\mathbf{x}_i$(see figure 6.1).

A training set of 30 points was generated and the experiments were repeated 100 times. A Gaussian kernel

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{||\mathbf{x} - \mathbf{y}||^2}{2\sigma^2}\right)$$

was chosen.

The results are summarized in figure 6.2 and the following observations can be made:

- For each σ, adding virtual training examples reduces the test error.
- The ISVM and $\mathrm{IH}_{\mathrm{KPCA}}$ yield almost identical performances, especially for large values of σ. This can be explained by figure 6.3. The ratio (6.13) has been computed for different values of σ. For large values, this ratio is almost 1, which means that there is almost no loss of information by

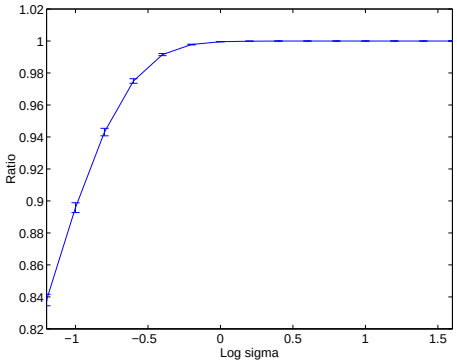


Figure 6.3: Approximation ratio (6.13) for different values of σ. When the ratio is near 1, there is almost no loss of information using IH_{KPCA} and thus in this case ISVM and IH_{KPCA} are identical.

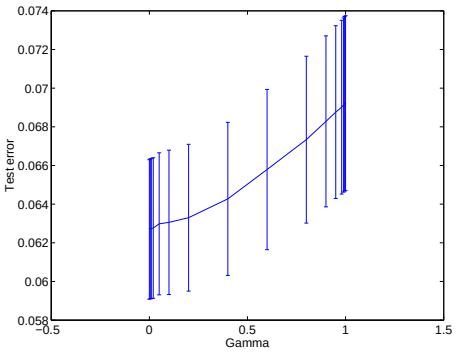


Figure 6.4: Test error versus γ for LIH. Trying to incorporate invariance using this method fails on this toy example.

projecting $d\Phi(\mathbf{x}_i)$ on E (see section 6.3.2) and thus ISVM and IH_{KPCA} produce the same decision boundary.

- The method LIH actually slightly impaired the performance. As pointed out above, this method has no guarantee to improve the performances. Actually, in this situation, we expected to get almost the same results as a standard SVM. Indeed, the covariance matrix of tangent vectors in input space is in this toy example is expected to be roughly a constant times the identity matrix:

$$\sum_{i=1}^{n} d\mathbf{x}_i d\mathbf{x}_i^\top \approx C\ I_2.$$

  Different values of γ have been tried for this method, but none of them led to an improvement (figure 6.4).

Some of the results presented in figure 6.2 are somehow surprising, but might be just particular to this problem:

- The test error of a normal SVM does not increase when σ has large values. We do not have here the traditional "bowl" shape.

- While the test error for SVM and VSV do not increase for large σ, the one of ISVM and IH_{KPCA} do. Maybe a larger value of γ (more invariance) should be applied in this case. This can be seen from

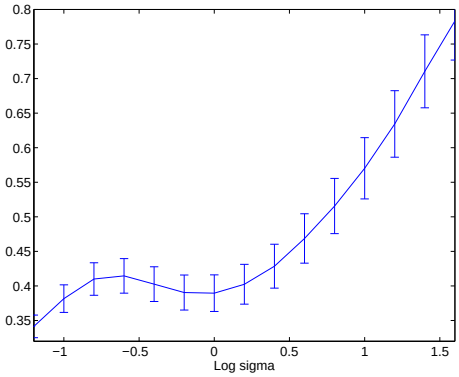


Figure 6.5: Value of the invariance enforcement part (eq 6.14) of the objective function across σ. A small value amounts to a better enforcement of the invariances.

SVM	VSV	LIH	ISVM	IH_{KPCA}
6.25	3.81	6.87	1.11	1.11

Table 6.1: Summary of the test error of the different learning algorithms

figure 6.5, where the following quantity has been plotted

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}\cdot d\Phi(\mathbf{x}_i))^2} \tag{6.14}$$

However, the reason for a larger value of γ remains unclear. Note that equation (6.14) gives a rule of thumb on how to choose a good value for γ. Indeed, after training a reasonable value for (6.14) should be in the range [0.2 - 0.4]. It represents the average difference between the output of a training point and its transformed $\mathcal{L}_t\mathbf{x}_i$.

Some comments about the right hand-side of figure 6.2.

- The more invariances, the better (and it converges to the same value for different σ). However, it might be due to the nature of this toy problem.
- When comparing $\log\sigma = 1.4$ and $\log\sigma = 0$, one notices that the decrease in the test error does not have the same speed. This is actually the dual of the phenomenon observed on the left side of figure 6.2 : for a same value of gamma, the test error tends to increase, when σ is larger.

Finally table 6.1 summarizes the test error of the different algorithms (using optimal parameter settings).

6.6.3 Handwritten digit recognition

As a real world experiment, we tried to incorporate invariances for a handwritten digit recognition task. The USPS dataset have been used extensively in the past for this purpose, especially in the SVM community. It consists of 7291 training and 2007 test examples.

According to [Schölkopf et al., 1995], the best performance has been obtained for a polynomial kernel of degree 3,

$$K(\mathbf{x},\mathbf{y}) = \left(\frac{\mathbf{x}\cdot\mathbf{y}}{256}\right)^3.$$

The factor 256, equaling the dimension of the data, has been included to avoid numerical instabilities. Since the choice of the base kernel is not our main concern here, all the results described in this section

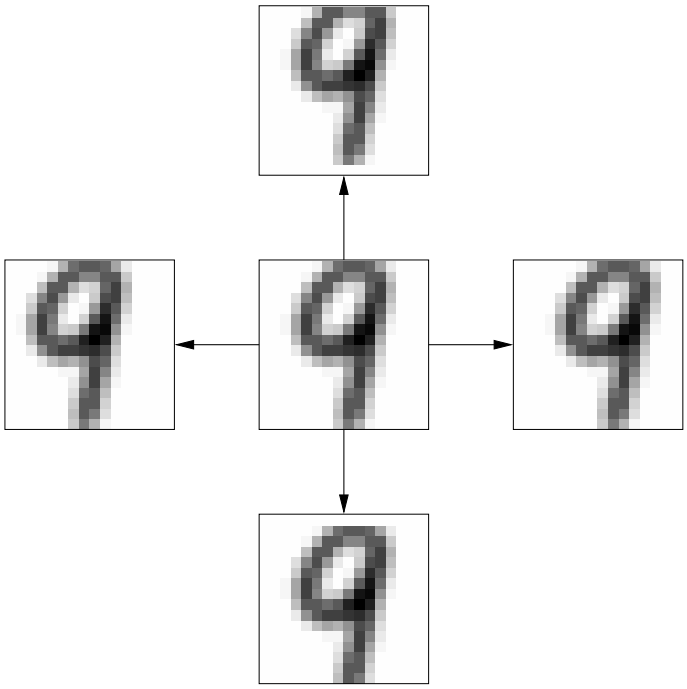


Figure 6.6: The original digit in the center and its translated from 1 pixel on the left, right, up and down

were performed using this kernel. The local transformations we considered are translations (horizontal and vertical). All the tangent vectors have been computed by a finite difference between the original digit and its 1-pixel translated (see figure 6.6).

Following the experimental protocol described in [Schölkopf et al., 1999], w e split the training set into 23 subsets of 317 training examples after a random permutation of the training and test set. Also we concentrated on a binary classification problem, namely separating digits 0 to 4 against 5 to 9. The gain in performance should also be valid for the multiclass case.

Figure 6.7 compares ISVM, IH_{KPCA} and VSV for different values of γ. From those figures, it can be seen that the difference between ISVM (the original method) and IH_{KPCA} (the approximation) is much larger than in the toy example.

The quality of the approximation can be estimated through the approximation ratio (6.13). It is 0.2 for the vertical translation and 0.33 for the horizontal one. Those number are far from being 1, which explains the difference in performance between the 2 methods (especially when $\gamma \to 1$). The difference to the toy example is probably due to the input dimensionality. In 2 dimensions, with an RBF kernel, the 30 examples of the toy problem "almost span" the whole feature space, whereas with 256 dimensions, this is no longer the case.

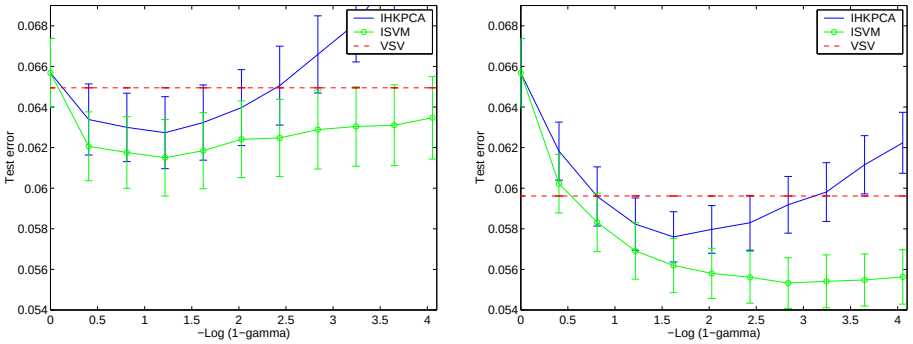
What is noteworthy in these experiments is that our proposed method is much better than the standard VSV. As explained in section 6.4, the reason for this might be that invariance is enforced around *all* training points and not only around support vectors. Note that what we call VSV here is a standard SVM with a *double size* training set containing the original data points and their translates.

The horizontal invariance yields larger improvements than the vertical one. One of the reason might be that the digits in the USPS database are already centered vertically (see figure 6.6).

After having studied single invariances, we wanted to see what kind of gain one can expect with multiple invariances. For the digit recognition task, we thus considered the 4 invariances (one pixel in each direction).

We compared IH_{KPCA} (which scales better than ISVM for multiple invariances, as mentioned at the beginning of section 6.3.2) with LIH in figure 6.8

There is again a significant improvement in performance while LIH fails.



Vertical translation (to the top) Horizontal translation (to the right)

Figure 6.7: Comparison of ISVM, IH_{KPCA} and VSV on the USPS dataset. The left of the plot ($\gamma = 0$) corresponds to standard SVM whereas the right part of the plot ($\gamma \to 1$) means that a lot of emphasis is put on the enforcement of the constraints. The test errors are averaged over the 23 splits and the error bars correspond to the standard deviation of the means.

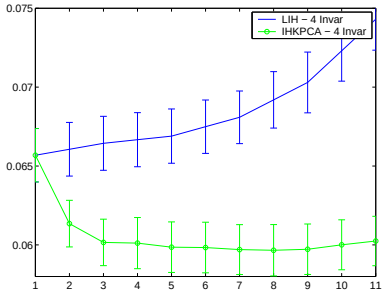


Figure 6.8: Results for 4 invariances and different values of γ. Left is standard SVM ($\gamma = 0$)

6.7 Summing-up

We have extended a method for constructing invariant hyperplanes to the nonlinear case. We have shown results that are superior to the virtual SV method. The latter has recently broken the record on the NIST database which is the "gold standard" of handwritten digit benchmarks [DeCoste and Schölkopf, 2002], therefore it appears promising to also try the new system on that task. For this propose, a large scale version of this method needs to be derived. The first idea we tried is to compute the kernel PCA map using only a subset of the training points. Encouraging results have been obtained on the 10-class USPS database (with the whole training set), but other methods are also currently under study.

Chapter 7

Semi-supervised learning

7.1 Introduction

In this chapter, we will consider the problem of semi-supervised learning, which usually involves few labeled examples and plenty of unlabeled ones. The first semi-supervised algorithm [Blum and Mitchell, 1998] was applied to web page classification. This is a typical scenario where the number of unlabeled examples can be made as large as possible (there are billions of web pages), but it takes time to label them since it requires human intervention.

Since then there has been a lot of interest for this paradigm. An extensive and excellent review of existing techniques for semi-supervised learning can be found in [Seeger, 2001b].

The intuition for semi-supervised learning is that the labeled examples provide information about the decision function itself while the unlabeled examples help to reveal the structure of the data. It has been shown experimentally that under certain conditions, the decision function can be estimated more accurately, yielding lower generalization error [Blum and Mitchell, 1998; Goldman and Zhou, 2000; Joachims, 1999].

7.1.1 Perfect knowledge of $P(\mathbf{x})$

In this chapter, we will try to understand how unlabeled data can help in supervised learning. Before trying to design algorithms, it might be worth assuming that an infinite amount of unlabeled data is available, in other words that the marginal probability distribution $P(\mathbf{x})$ is known perfectly. Even in this case, it is not straightforward to incorporate this information in a classifier. Without any assumption, it has even been proven that the information about the marginal distribution is useless for a purely discriminative classifier [Seeger, 2001b; Zhang and Oles, 2000]. For example, if $P(\mathbf{x})$ is known to be uniform on a hypercube, it seems unlikely that a discriminative classifier can gain in accuracy using unlabeled samples.

The reason for keeping in mind the special case of $P(\mathbf{x})$ perfectly known is that it can be useful in analyzing semi-supervised algorithms and for understanding on which assumptions they rely. Also, a desirable feature of a semi-supervised algorithm is that the accuracy increases as the number of unlabeled data goes to infinity. Unfortunately, some algorithms do no have this property [Cozman and Cohen, 2001] or would not know how to take advantage of the information about $P(\mathbf{x})$.

7.1.2 Literature review

At a first sight, unlabeled data are better suited for classifiers based on generative models. Giving a class of models Θ for the data, a classifier based on a generative model finds $\hat{\boldsymbol{\theta}} \in \Theta$ which maximizes the likelihood of the data,

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log P(\mathbf{x}_i | y_i, \boldsymbol{\theta}),$$

and the resulting classification function is based on the sign of $P(y = 1|\mathbf{x}, \boldsymbol{\theta}) - 1/2$, which can be computed with the Bayes rule. Typically, a mixture model with 2 components (one per class) is used. When unlabeled data are available, the maximizer $\hat{\boldsymbol{\theta}}$ of the likelihood

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log P(\mathbf{x}_i|y_i, \boldsymbol{\theta}) + \sum_{i=1}^{n_u} \log P(\mathbf{x}'_i|\boldsymbol{\theta}) \tag{7.1}$$

can be found with the EM algorithm [Dempster et al., 1977] by treating the class labels of unlabeled points as hidden variables. Note that the standard case is to assume a one-to-one correspondence between the mixture components and the class labels, but it is possible to have several mixture components per class [Nigam et al., 1998] or several classes per component [Miller and Uyar, 1997].

The EM algorithm is an iterative procedure yielding a sequence of model parameters $\hat{\boldsymbol{\theta}}_p$. Given $\hat{\boldsymbol{\theta}}_p$, $\hat{\boldsymbol{\theta}}_{p+1}$ is found by maximizing

$$\sum_{i=1}^{n} \log P(\mathbf{x}_i|y_i, \boldsymbol{\theta}) + \sum_{i=1}^{n_u} \sum_{y\in\{-1,1\}} P(y'_i = y|\mathbf{x}'_i, \hat{\boldsymbol{\theta}}_p) \log \frac{P(\mathbf{x}'_i|\boldsymbol{\theta})}{P(y'_i = y|\mathbf{x}'_i, \hat{\boldsymbol{\theta}}_p)},$$

which is, by Jensen inequality, a lower bound on the likelihood (7.1) [Neal and Hinton, 1998]. Thus step $p+1$ can be viewed as a standard supervised learning on both labeled and unlabeled samples, but where the labels of the unlabeled points are actually "soft" labels given by the posterior probabilities $P(y|\mathbf{x}'_i, \hat{\boldsymbol{\theta}}_p)$.

This approach can be easily extended to any discriminative classifier if one replaces this soft label by a hard one. This gives the popular *decision directed* method [Amini, 2001]: at each iteration, a small set of unlabeled data is labeled according to the current decision boundary and added to the labeled examples set. If this method is used with logistic regression, this will tend to push the decision function away from the labeled examples [Amini and Gallinari, 2001].

In a discriminative setting, the idea of enforcing a large margin on the unlabeled examples has been applied for Support Vector Machines by Vapnik in his transduction framework [Vapnik, 1998]. He suggests to find the hyperplane that maximize the margin on both the labeled and the unlabeled data. Encouraging results were obtained using this method [Bennett and Demiriz, 1998; Joachims, 1999]. The algorithm proposed in the *Maximum Entropy Discrimination* framework [Jaakkola et al., 1999b] tends also to maximize the margin on the unlabeled points, but unlike the transductive SVM algorithm, it gives a soft assignment for the labels of the unlabeled points.

It is worth pointing out that maximizing the margin on the labeled points came from a regularization argument, but maximizing the margin on the unlabeled points should be interpreted as an attempt to put the decision function in regions of low density. Vapnik did not motivate transductive SVMs from this point of view and several authors questioned the validity of this approach [Zhang and Oles, 2000; Seeger, 2001b]. But implicitly, what a transductive SVM does is just trying to put the hyperplane in low density regions as explained in section 7.3.

7.1.3 Ideas presented in this chapter

At a first sight, semi-supervised learning seems to fit quite well in the VRM framework since this induction principle depends on density estimation and unlabeled data can help by generating more accurate density estimates. We will investigate different possibilities of combining VRM and unlabeled data in section 7.2.

The second idea implies to consider the "cluster assumption" [Seeger, 2001b], which says that two points in the same cluster (i.e. which are connected by a high density path) are likely to have the same label. Consequently, the decision function should lie in regions of low density. We will discuss this idea in section 7.3 and present kernels which implement this assumption in section 7.4.

Another idea, related to the previous one, comes from the observation that high-dimensional data often lie on a low dimensional manifold. Thus, using unlabeled data, we show in section 7.5 what techniques can be used to learn this manifold and to find a decision function which satisfies the "low density" region assumption (cf above), i.e. which does does not "cut" this manifold.

Finally, in section 7.6, we propose a transductive algorithm for ridge regression: the labels of the test points are chosen to minimize the leave-one-out error of this algorithms.

7.2 VRM with unlabeled data

There are several ways to incorporate unlabeled data in the VRM framework. The first attempt was suggested in [Chapelle et al., 2000] and is not quite satisfying since it was not properly motivated.

7.2.1 Vicinal risk for unlabeled points

One use of unlabeled data in the framework of VRM is to add the vicinal loss for unlabeled points to the objective function. Given m unlabeled points $\mathbf{x}_1^*, \ldots, \mathbf{x}_m^*$, one obtains the following formulation:

$$R_{vic}(f) = \frac{1}{n}\sum_{i=1}^{n}\int \ell(f(\mathbf{x}), y_i) dP_{\mathbf{x}_i}(\mathbf{x}) + \frac{1}{m}\sum_{i=1}^{m}\int \ell(f(\mathbf{x}), f(\mathbf{x}_i^*)) dP_{\mathbf{x}_i^*}(\mathbf{x}).$$

This approach is similar to transductive SVMs [Bennett and Demiriz, 1998; Vapnik, 1998; Joachims, 1999] since it tends to maximize the margin on both unlabeled data and labeled data. The intuition is that it should work well when both classes are well separated. In section 7.3 a detailed justification will be provided.

To illustrate this approach consider the following example. Two normal distributions on the real line $\mathcal{N}(-1.6, 1)$ and $\mathcal{N}(1.6, 1)$ model the patterns of two classes with equal probability; 20 labeled points and 100 unlabeled points are drawn. The following table compares the true generalization error of VRM with gaussian kernels and linear functions. Results are averaged over 100 runs. Two different kernel widths σ_L and σ_U were used for kernels associated with labeled or unlabeled examples. Best kernel widths were obtained by cross-validation. We also studied the case $\sigma_L \to 0$ in order to provide a result equivalent to a plain SVM.

σ_L	σ_U	Labeled	Labeled+Unlabeled
$\sigma_L \to 0$	Best σ_U	6.5%	5.6%
Best σ_L	Best σ_U	5.0%	**4.3%**

Note that when both σ_L and σ_U tend to zero, this algorithm reverts to the transduction algorithm due to Vapnik which was previously solved by the more difficult optimization procedure of integer programming [Bennett and Demiriz, 1998].

7.2.2 Reweighting

In Vicinal Risk Minimization, the underlying probability distribution is estimated by (1.13). If we know the true $P(\mathbf{x})$, we should find an estimate $P_{est}(\mathbf{x}, y)$ which satisfies

$$\forall \mathbf{x}, \underbrace{\sum_{y\in\{-1,1\}} P_{est}(\mathbf{x}, y)}_{P_{est}(\mathbf{x})} = P(\mathbf{x}). \tag{7.2}$$

For this purpose, we propose the following parametric estimation of $P_{est}(\mathbf{x}, y)$,

$$P_{est}^{\boldsymbol{\lambda}}(\mathbf{x}, y) = \frac{1}{n}\sum_{i=1}^{n} \lambda_i P_{\mathbf{x}_i}(\mathbf{x}) \delta_{y_i}(y),$$

where $(\lambda_1, \ldots, \lambda_n)$ are some parameters which need to be adjusted.

Note that $P_{est}^{\boldsymbol{\lambda}}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \lambda_i P_{\mathbf{x}_i}(\mathbf{x})$, and since $P(\mathbf{x})$ can be very complex it is unlikely to find a vector $\boldsymbol{\lambda}$ such that equation (7.2) is valid $\forall \mathbf{x}$. Instead, it is possible to find a vector $\boldsymbol{\lambda}$ which minimizes a distance between $P_{est}^{\boldsymbol{\lambda}}(\mathbf{x})$ and $P(\mathbf{x})$.

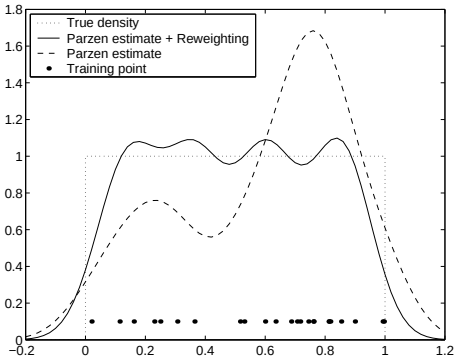


Figure 7.1: The true distribution $P(\mathbf{x})$ is uniform on $[0, 1]$, but because of the finite sampling, it turned out that more points have been drawn in $[0.5, 1]$ than in $[0, 0.5]$. As a consequence, the Parzen estimate of the density (dashed line) if far from the true one (dotted line). By giving more weight to points on the left and less to points on the right, the new Parzen window estimate (solid line) is closer to the true density.

Adjusting the density

Two of the common distances to compare densities are:

- Kullback-Liebler,
$$d_{KL}(P, Q) = \int \log \frac{P(\mathbf{x})}{Q(\mathbf{x})} dP(\mathbf{x})$$
- L_2 distance,
$$d_{L_2}(P, Q) = \int (P(\mathbf{x}) - Q(\mathbf{x}))^2 dP(\mathbf{x})$$

Once the vector $\boldsymbol{\lambda}$ minimizing either the L_2 or the Kullback-Liebler distances has been found, we minimize the vicinal risk using $P^{\boldsymbol{\lambda}}_{est}(\mathbf{x}, y)$.

The coefficients $(\lambda_1, \ldots, \lambda_n)$ can be interpreted as reweighting coefficients: points which are over-represented (they lie in a region where there are more points than there should be according to $P(\mathbf{x})$) will have a small λ, whereas points which are under-represented will have a large λ (see figure 7.1).

Since $(\lambda_1, \ldots, \lambda_n)$ correspond to weights, we impose the following constraints:

$$\lambda_i \geq 0, \quad 1 \leq i \leq n,$$

$$\sum \lambda_i = n$$

Minimizing the KL divergence. We want to minimize

$$\int \log \frac{P(\mathbf{x})}{\frac{1}{n} \sum_{i=1}^{n} \lambda_i P_{\mathbf{x}_i}(\mathbf{x})} dP(\mathbf{x}),$$

which is equivalent to maximize

$$\int \log \sum_{i=1}^{n} \lambda_i P_{\mathbf{x}_i}(\mathbf{x}) dP(\mathbf{x}). \tag{7.3}$$

Note that the Hessian,

$$H_{pq} = - \int \frac{P_{\mathbf{x}_p}(\mathbf{x}) P_{\mathbf{x}_q}(\mathbf{x})}{(\sum_{i=1}^{n} \lambda_i P_{\mathbf{x}_i}(\mathbf{x}))^2} dP(\mathbf{x})$$

is negative definite since for any vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$,

$$\boldsymbol{\alpha}^\top H \boldsymbol{\alpha} = -\int \left(\sum_{p=1}^{n} \frac{\alpha_p P_{\mathbf{x}_p}(\mathbf{x})}{\sum_{i=1}^{n} \lambda_i P_{\mathbf{x}_i}(\mathbf{x})} \right)^2 dP(\mathbf{x}) \leq 0.$$

This maximization problem is thus concave and can be performed easily.

In practice, $P(\mathbf{x})$ is unknown, but we have unlabeled points $(\mathbf{x}'_1, \ldots, \mathbf{x}'_{n_u})$ drawn from this distribution. The empirical counterpart of (7.3) is

$$\sum_{j=1}^{n_u} \log \sum_{i=1}^{n} \lambda_i P_{\mathbf{x}_i}(\mathbf{x}_j). \tag{7.4}$$

Minimizing the L_2 distance

$$\int \left(\frac{1}{n} \sum_{i=1}^{n} \lambda_i P_{\mathbf{x}_i}(\mathbf{x}) - P(\mathbf{x}) \right)^2 dP(\mathbf{x}).$$

This is quadratic in $(\lambda_1, \ldots, \lambda_n)$, so it can also be minimized easily.

If a finite number of unlabeled data $(\mathbf{x}'_1, \ldots, \mathbf{x}'_{n_u})$ is available, $P(\mathbf{x})$ can be estimated by a Parzen window,

$$\tilde{P}(\mathbf{x}) = \frac{1}{n_u} \sum_{i=1}^{n_u} P'_{\mathbf{x}_i}(\mathbf{x}), \text{ and } P'_{\mathbf{x}_i}(\mathbf{x}) = \frac{1}{(2\pi\sigma_u^2)^{d/2}} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}\|^2}{2\sigma_u^2} \right).$$

We thus have to minimize

$$\sum_{j=1}^{n_u} \left(\frac{1}{n} \sum_{i=1}^{n} \lambda_i P_{\mathbf{x}_i}(\mathbf{x}_j) - \frac{1}{n_u} \sum_{i=1}^{n_u} P'_{\mathbf{x}_i}(\mathbf{x}_j) \right)^2.$$

Whether one minimizes the KL divergence or the L_2 distance, it is possible to add some regularization constraints on $\boldsymbol{\lambda}$. For instance, adding to the objective function a term such as $\sum(\lambda_i - 1)^2$ or $-\sum \log \lambda_i$ will prevent the λ_i to take values very different from 1. Also one can add the constraint that the sum of the λ_i over the positive class is equal to the number of positive points.

Weighted SVM

Once the weights $\boldsymbol{\lambda}$ have been found, it is straightforward to include them in any algorithm or induction principle which minimizes a sum of losses on all points such as ERM or VRM: the sum becomes a weighted sum,

$$R^{\boldsymbol{\lambda}}_{vic}(f) = \frac{1}{n} \sum_{i=1}^{n} \lambda_i \int \ell(f(\mathbf{x}), y_i) dP_{\mathbf{x}_i}(\mathbf{x}).$$

It has been shown in section 2.1.3 that Support Vector Machines are a special case of VRM with a modified loss function. Following this equivalence, the natural way to extend the SVM algorithm to the case of weighted training examples is to replace the SVM objective function (2.12) by

$$\frac{1}{2}\mathbf{w}^2 + C \sum_{i=1}^{n} \lambda_i \xi_i,$$

and in the dual, the Lagrange multipliers are bounded by $0 \leq \alpha_i \leq C\lambda_i$. It might seem surprising that for the hard margin case, the weighted SVM is identical to the original SVM. This is another reason to prefer soft-margin SVMs which have been shown to be superior even when the training points are separable. From our point view, hard-margin SVMs should rarely be used.

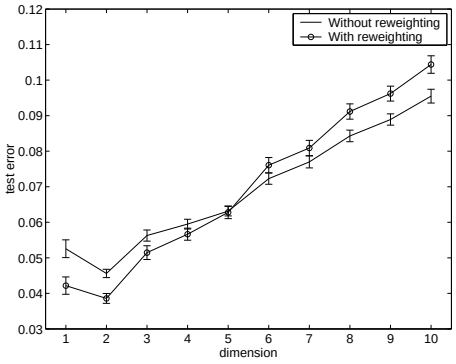


Figure 7.2: Test error of the Parzen window classifier as a function of the dimensionality of the toy problem

Experiments

The reweighting algorithm has been tested on a d dimensional toy problem. The data have been generated from $\mathcal{N}(\mathbf{1} \cdot 2y/\sqrt{d}, I_d)$ and $P(y = 1) = P(y = -1) = 1/2$. The Bayes error for this problem is 2.28%.

We minimize the KL divergence (7.4) in two steps. First, fix all λ_i equal to 1 and optimize on σ, the width the Parzen window estimator; then optimize on $\boldsymbol{\lambda}$. In this way, there is no free parameter to tune.

For simplicity, we considered the standard Parzen window classifier, which is the one minimizing the vicinal risk when the set of functions is not constrained,

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{n} \lambda_i y_i P_{\mathbf{x}_i}(\mathbf{x})\right). \tag{7.5}$$

Here we chose a RBF kernel for the density estimate,

$$P_{\mathbf{x}_i}(\mathbf{x}) = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}\|^2}{2\sigma^2}\right),$$

and where σ has been chosen to minimize (7.4). We compared the performances of the Parzen window classifier (7.5) versus its unweighted version ($\lambda_1 = \cdots = \lambda_n = 1$). There were 10 training points and 1000 test points. The unlabeled set was chosen to be the test set. Results, in figure 7.2, have been averaged over 1000 trials.

It seems that this method is effective only for low-dimensional problem, maybe because some regularization is needed for a higher number of dimensions. Also, when the separation between the two clusters is smaller (1.5 instead of 2), the test error increased with the reweighting from 10.6% to 10.9% (dimension 1). From these experiments, it seems that the reweighting approach as it has been proposed here is not very effective.

Constrained Parzen estimator

The Parzen window estimate [Parzen, 1962] can be seen as the solution of a regularized optimization problem:

Theorem 7.1 (Parzen estimator Vapnik 1998) *Consider the problem of 1D density estimation. For a given density P, let $\mathcal{P}$ be its distribution function,*

$$\mathcal{P}(x) = \int_{-\infty}^{x} P(t)dt$$

and $\mathcal{P}_n$ the empirical distribution function,

$$\mathcal{P}_n(x) = \frac{1}{n} card(i,\ x_i \leq x).$$

Then the density minimizing

$$\|\mathcal{P} - \mathcal{P}_n\|_2^2 + \gamma \|P\|_2^2$$

is given by

$$P(x) = \frac{1}{2n\sqrt{\gamma}} \exp\left(-\frac{|x|}{\sqrt{\gamma}}\right).$$

By changing the regularization term $\|P\|_2^2$, one can get other kernels than the Laplacian kernel.

Rather than finding weighting coefficients, let us rederive the Parzen window estimator under the constraint (7.2). Remember that we are trying to estimate the density of the positive and negative classes, P^+ and P^-, under the constraint that we can rewrite

$$\frac{n^+}{n} P^+(x) + \frac{n^-}{n} P^-(x) = P(x),$$

where $P(x)$ is supposed to be known and n^+, n^- are the number of positive and negative points such that $n = n^+ + n^-$.

We propose to find P^+ and P^- which minimize

$$\begin{aligned} R(P^+, P^-) &= \left\|\mathcal{P}^+ - \mathcal{P}_n^+\right\|_2^2 + \left\|\mathcal{P}^- - \mathcal{P}_n^-\right\|_2^2 + \\ &\quad \gamma\left(\left\|P^+\right\|_2^2 + \left\|P^-\right\|_2^2\right) + \lambda \left\| \frac{n^+}{n} P^+(x) + \frac{n^-}{n} P^-(x) - P(x) \right\|_2^2. \end{aligned} \tag{7.6}$$

Let us follow the same lines as in the proof of theorem 7.1.

Let $\bar{P}$ the Fourier transform of P,

$$\bar{P}(\omega) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} P(x) e^{-i\omega x} dx.$$

The Fourier transform of the distribution function is then

$$\bar{\mathcal{P}}(\omega) = \frac{\bar{P}(\omega)}{i\omega},$$

and for the empirical distribution function,

$$\bar{\mathcal{P}}_n(\omega) = \frac{1}{n} \sum_{p=1}^{n} \frac{e^{-i\omega x_p}}{i\omega}.$$

Recall that according to Parseval's equality the L_2 norm of any function is equal to the L_2 norm of its Fourier transform (up to the 2π constant). We can thus rewrite equation (7.6) as

$$\begin{aligned} R(P^+, P^-) &= \left\| \frac{\bar{P^+}(\omega) - \frac{1}{n^+} \sum_{p=1}^{n^+} e^{-i\omega x_p^+}}{i\omega} \right\|_2^2 + \left\| \frac{\bar{P^-}(\omega) - \frac{1}{n^-} \sum_{p=1}^{n^-} e^{-i\omega x_p^-}}{i\omega} \right\|_2^2 + \\ &\quad \gamma\left(\left\|\bar{P^+}\right\|_2^2 + \left\|\bar{P^-}\right\|_2^2\right) + \frac{\lambda}{n^2} \left\| n^+ \bar{P^+} + n^- \bar{P^-} - n\bar{P} \right\|_2^2. \end{aligned}$$

Minimizing with respect to $\bar{P^+}(\omega)$ gives

$$\frac{1}{2} \frac{\partial R(P^+, P^-)}{\partial \bar{P^+}(-\omega)} = \frac{\bar{P^+}(\omega)}{\omega^2} - \frac{1}{\omega^2 n^+} \sum_{p=1}^{n^+} e^{-i\omega x_p^+} + \gamma \bar{P^+}(\omega) + \frac{\lambda n^+}{n^2} \left(n^+ \bar{P^+}(\omega) + n^- \bar{P^-}(\omega) - n\bar{P}(\omega) \right) = 0.$$

A similar equation can be obtained with $\bar{P^-}(\omega)$ and we obtained a linear system that we have to solve in $\bar{P^+}(\omega)$ and $\bar{P^-}(\omega)$. Solving this system yields

$$\bar{P^+}(\omega)\left[(1+\gamma\omega^2)\left(1+\gamma\omega^2+2\lambda\omega^2\frac{{n^+}^2+{n^-}^2}{n^2}\right)\right] = \left[1+\gamma\omega^2+\lambda\omega^2\frac{{n^-}^2}{n^2}\right]\frac{1}{n^+}\sum_{p=1}^{n^+}e^{-i\omega x_p^+}$$

$$-\lambda\omega^2\frac{n^+n^-}{n^2}\frac{1}{n^-}\sum_{p=1}^{n^-}e^{-i\omega x_p^-}+(1+\gamma\omega^2)\lambda\omega^2\frac{n^+}{n}\bar{P}(\omega). \tag{7.7}$$

Now let us rewrite the vicinal risk as,

$$\begin{aligned}
R_{vic}(f) &= \int \mathrm{I}_{f(x)\neq y}dP(x,y)\\
&= \int_{f(x)=-1}dP(x|y=1)P(y=1)+\int_{f(x)=1}dP(x|y=-1)P(y=-1)\\
&= \int_{f(x)=-1}dP(x|y=1)P(y=1)+P(y=-1)-\int_{f(x)=-1}dP(x|y=-1)P(y=-1)\\
&= \frac{n^-}{n}+\int_{f(x)=-1}\frac{n^+}{n}dP^+(x)-\frac{n^-}{n}dP^-(x).
\end{aligned}$$

So, in order to compute the vicinal risk, we do not need to know P^+ and P^-, but only the difference

$$\frac{n^+}{n}P^+-\frac{n^-}{n}P^-.$$

Let us see what happens to the term in $\bar{P}(\omega)$ in equation (7.7)

$$\frac{n^+}{n}P^+(\omega)-\frac{n^-}{n}P^-(\omega)= \ldots\left(1+\gamma\omega^2+2\lambda\omega^2\frac{{n^+}^2+{n^-}^2}{n^2}\right)^{-1}\lambda\omega^2\frac{n^+-n^-}{n}\bar{P}(\omega)$$

and we can draw the following conclusion: if $n^+=n^-$, *the information about $P(x)$ is not taken into account* in the vicinal risk with this constrained Parzen window estimate. This can be viewed as a negative theoretical result for the usefulness of $P(\mathbf{x})$ for classifiers based on kernel estimates of the density.

7.2.3 Adapting the widths

In the previous section, the weights of the local density estimate have been adjusted in order to fit the global density estimate $P(\mathbf{x})$. In the experiments, we even chose automatically the width of the Parzen window estimate. A natural extension is to come back to the method suggested in section 2.3.1, but instead of choosing in a heuristic way the widths of each training point, let us take advantage of $P(\mathbf{x})$ to adjust them automatically.

We consider the following density estimate,

$$P_{est}^{\boldsymbol{\sigma}}(\mathbf{x})=\sum_{i=1}^{n}\frac{1}{(2\pi\sigma_i^2)^{d/2}}\exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{2\sigma_i^2}\right).$$

The vector $\boldsymbol{\sigma}$ is found by minimizing the KL divergence with the unlabeled estimate of the density,

$$-\sum_{j=1}^{n_u}\log P_{est}^{\boldsymbol{\sigma}}(\mathbf{x}_j).$$

Then the vicinal risk is minimized as in section 2.3.1.

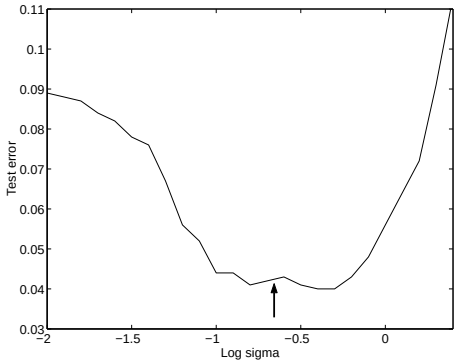


Figure 7.3: Test error of the Parzen window classifier on one run of the toy problem described in section 7.2.2, plotted against the width σ of the kernel estimate. The arrow indicates the value which minimizes the KL divergence (i.e. maximizes the likelihood) of the unlabeled points.

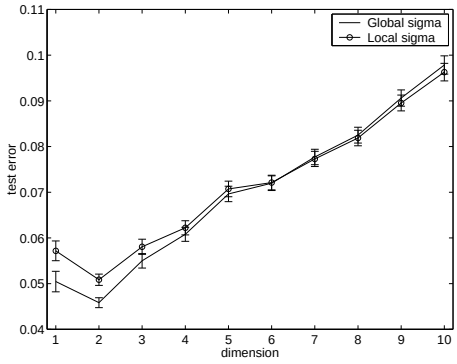


Figure 7.4: Comparison of the test error of the Parzen window classifier, where the width of the kernel is either global or adapted for each training point.

To test this approach, we used the same toy problem as the one of the previous section. We first chose by gradient descent a global value for $\sigma_1 = \ldots \sigma_n = \sigma$ and then the different σ_i are optimized.

We first checked how close the global σ is to the optimal (see figure 7.3). In most of the runs, it is chosen relatively close of the optimal.

The next experiment involves the full optimization on $(\sigma_1, \ldots, \sigma_n)$. Results are presented in figure 7.4. They are quite disappointing: there is a slight deterioration in the performances compared to the case $\sigma_1 = \ldots \sigma_n = \sigma$.

7.2.4 Other perspectives

From the experiments in this section, it seems that there is nothing gained by adapting the density estimate to make it fit the unlabeled data. Actually, one should differentiate global and local adaptation. Optimizing the width and the weight of the Gaussian associated to each training point is a local adaptation, since those parameters are "local" for a given training point. On the other hand, one could optimize the covariance matrix of the kernel estimate. If this covariance matrix is identical for each point, this would be a "global" adaptation.

The problem with local adaptation is that it tries to fit the kernel estimate of the density with the true $P(\mathbf{x})$, but this does not guarantee at all that the class conditional estimates of the density will be nearer to the actual $P(\mathbf{x}, y)$.

In the global adjustment of the parameters, one tries to learn the structure of the data. A simple

example would be to the same covariance matrix Σ associated with each training point,

$$\tilde{P}(\mathbf{x}) = \sum_{i=1}^{n} \frac{1}{(2\pi)^{d/2}\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{x}_i)^\top \Sigma^{-1}(\mathbf{x}-\mathbf{x}_i)\right),$$

and Σ can be learned easily from the unlabeled using the EM algorithm. A similar suggestion can be found in [Tong and Koller, 2000]. Details of this approach will be provided in section 7.5.2.

7.3 Hyperplane in low density regions

To improve the usefulness of unlabeled data, an assumption about the nature of the data seems to be required. One which is made, explicitly or implicitly, by most of the semi-supervised learning algorithm is the so-called "cluster assumption" saying that two points should have the same label if there is a path connecting them passing only through regions of high density.

Another way of stating this assumption is to say that the decision boundary should lie in regions of low density. In real world problems, this makes sense: let us consider handwritten digit recognition and suppose one tries to classify digits 0 from 1. The probability of having a digit inbetween a "0" and a "1" is very low.

A first way to implement the cluster assumption is to make sure that the decision function lies in regions of low density. For this purpose, one can add the following criterion to the vicinal risk:

$$R_{unl}(f) = \int_{\mathbf{x}/f(\mathbf{x})=0} P(\mathbf{x}) d\mathcal{L}_{d-1}(\mathbf{x}),$$

where $\mathcal{L}_{d-1}$ is the $d-1$ dimensional Lebesgue measure.

As for computing the vicinal risk, let us consider the special case of linear classifier with a Parzen estimate of the marginal distribution $P(\mathbf{x})$

$$\begin{aligned}
R_{unl}(f) &= \int_{\mathbf{x}/\ \mathbf{w}\cdot\mathbf{x}+b=0} \frac{1}{n_u}\sum_{i=1}^{n_u} \frac{1}{(2\pi\sigma_u^2)^{d/2}} \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{2\sigma_u^2}\right) d\mathcal{L}_{d-1}(\mathbf{x}) \\
&= \frac{1}{n_u}\sum_{i=1}^{n_u} \int_{\mathbf{x}/\ \mathbf{w}\cdot\mathbf{x}+b=0} \frac{1}{(2\pi\sigma_u^2)^{d/2}} \exp\left(-\frac{\left\|\mathbf{x}-\mathbf{x}_i^\perp\right\|^2 + \left\|\mathbf{x}_i^\perp-\mathbf{x}_i\right\|^2}{2\sigma_u^2}\right) d\mathcal{L}_{d-1}(\mathbf{x}) \\
&= \frac{1}{n_u\sqrt{2\pi\sigma_u^2}} \sum_{i=1}^{n_u} \exp\left(-\frac{\left\|\mathbf{x}_i^\perp-\mathbf{x}_i\right\|^2}{2\sigma_u^2}\right),
\end{aligned}$$

where $\mathbf{x}_i^\perp$ is the orthonormal projection of $\mathbf{x}_i$ on the hyperplane $\mathbf{w}\cdot\mathbf{x}+b=0$. If d_i is the distance between the unlabeled point $\mathbf{x}_i$ and the hyperplane, the term $R_{unl}(f)$ to add to the vicinal risk is (up to a multiplicative factor),

$$\sum_{i=1}^{n_u} \exp\left(-\frac{d_i^2}{2\sigma_u^2}\right).$$

The effect of this term is to push the hyperplane away from unlabeled points and to maximize the margin on those points. For this reason, minimizing this risk is somehow similar to transductive SVMs [Vapnik, 1998; Joachims, 1999] which maximize the margin on both unlabeled and unlabeled points. However, one needs to keep in mind that those margin are from different nature and for example, there is no reason to enforce the same margin distance on the labeled and the unlabeled points.

A drawback of the previous approach is that the function to maximize is non-convex and it is not at all possible to find a convex approximation of it.

For this reason, we will investigate different approaches in the following section.

7.4 Kernels based on the cluster assumption

In this section, we explore different ideas on how to build kernels which implement the cluster assumption. In section 7.4.4, we will propose a framework which unifies the methods proposed in [Szummer and Jaakkola, 2001] and [Ng et al., 2001].

7.4.1 Random walk kernel

A way to implement the "cluster assumption" is to change the metric in the input space such that points which are in the same cluster have a smaller distance and points which are in a different cluster have a larger distance.

The idea of [Szummer and Jaakkola, 2001] is to compute the RBF kernel matrix (with the labeled and unlabeled points)

$$K_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$

and to interpret it as a transition matrix of a random walk on a graph:

$$P(\mathbf{x}_i \to \mathbf{x}_j) = \frac{K_{ij}}{\sum_p K_{ip}}. \tag{7.8}$$

After t steps (where t is a parameter to be determined), the probability of going from a point $\mathbf{x}_i$ to a point $\mathbf{x}_j$ should be quite high if both points belong to the same cluster and should stay low if they are in two different clusters. This idea of a random walk has also been used for clustering [Tishby and Slonim, 2000].

Let D be the diagonal matrix whose elements are

$$D_{ii} = \sum_j K_{ij}. \tag{7.9}$$

The one step transition matrix is $D^{-1}K$ and after t steps it is $P^t = (D^{-1}K)^t$. This matrix could be the basis of a kernel matrix in an SVM classifier, but in [Szummer and Jaakkola, 2001], once it has been computed, the authors keep the interpretation of the transition probabilities and derive a classifier built on those probabilities.

Our approach is a little bit different in the sense that we would like to think in terms of distances: equation (7.8) translates distances (appearing in the RBF kernel) in probabilities and now we would like to use the transition probabilities after t steps as a similarity measure between points.

However, to use P^t as a kernel matrix, several problems arise:

- The transition matrix is not even symmetric and can not be used right away as the kernel matrix of an SVM.
- The distance of a point to itself should be 0, but the probability of going back to the same point after t steps is not even the same for all points.
- In the SVM framework, the choice of a kernel is equivalent to the choice of a regularization function, but in this case, it is not obvious to see what the new kernel corresponds to.

There are several ways to construct a kernel matrix (symmetric and positive) from P^t. A direct way to make it symmetric is to multiply P^t on the left by D or to consider the matrix $(P^t)^\top + P^t$. An other possibility is to consider the i-th row of the matrix P^t as a new feature feature vector (representation) of the point $\mathbf{x}_i$. The kernel corresponding to a linear classifier in this new space is $P^t(P^t)^\top$. We will present in section 7.4.4 a properly justified kernel based on the random walk transition matrix P^t.

7.4.2 Shortest path kernel

In a random walk, the probability of a given path is the product of the probabilities of the transition:

$$P(\mathbf{x}_{i_1} \to \cdots \to \mathbf{x}_{i_t}) = \prod_{p=1}^{t-1} P(\mathbf{x}_{i_p} \to \mathbf{x}_{i_{p+1}}) = \frac{\exp\left(-\left(\sum_{p=1}^{t-1} \left\|\mathbf{x}_{i_p} - \mathbf{x}_{i_{p+1}}\right\|^2\right)/2\sigma^2\right)}{\prod_{p=1}^{t-1} \sum_j K_{i_p j}}. \tag{7.10}$$

Let us see what happens when $\sigma \to 0$. The matrix K becomes almost the identity matrix and the denominator of (7.10) tends to 1. The probability of going from a point $\mathbf{x}_i$ to a point $\mathbf{x}_j$ in t steps is the sum of probabilities of type (7.10) over all the paths going from $\mathbf{x}_i$ to $\mathbf{x}_j$. When $\sigma \to 0$, the sum will be dominated by the shortest path, the one minimizing

$$\sum_{p=1}^{t-1} \left\|\mathbf{x}_{i_p} - \mathbf{x}_{i_{p+1}}\right\|^2.$$

This gives us an insight for a new similarity measure, that we can call the shortest path "kernel":

$$K_{SP}(\mathbf{x}_i, \mathbf{x}_j) = \exp(-d^2_{SP}(\mathbf{x}_i, \mathbf{x}_j)/2\sigma^2) \quad \text{and} \quad d^2_{SP}(\mathbf{x}_i, \mathbf{x}_j) = \min_{p/\text{ path from } \mathbf{x}_i \text{ to } \mathbf{x}_j} \sum \left\|\mathbf{x}_{i_p} - \mathbf{x}_{i_{p+1}}\right\|^2$$

An advantage of considering this extreme case is that $d_{SP}(\mathbf{x}_i, \mathbf{x}_i) = 0$. However note that K_{SP} is in general not positive definite.

7.4.3 Kernel induced by a clustered representation

Another idea to implement the cluster assumption is to change the representation of the input points such that points in the same cluster are grouped together in the new representation. For this purpose, one can use tools of spectral clustering (see [Weiss, 1999] for a review and more recently [Cristianini et al., 2001]). Recently, it has been shown [Ng et al., 2001] that using the first eigenvectors of a similarity matrix, the training points are very well clustered in the new representation that uses those eigenvectors. A simple K-means algorithm in the new space yield a good clustering solution. We will try to use a discriminative learning algorithm in their representation of the data. This algorithm of [Ng et al., 2001], which resembles kernel PCA, is the following:

1. Compute the *affinity* matrix, which is an RBF kernel matrix but with diagonal elements being 0 instead of 1.
2. Let D be a diagonal matrix with diagonal elements equal to the sum of the rows (or the columns) of K and construct the matrix $L = D^{-1/2}KD^{-1/2}$.
3. Find the eigenvectors $(\mathbf{v}_1, \ldots, \mathbf{v}_k)$ of L corresponding the first k eigenvalues.
4. The new representation of the point $\mathbf{x}_i$ is $(v_{i1}, \ldots, v_{ik})$ and is normalized to have length one:

$$\varphi(\mathbf{x}_i)_p = v_{ip} \;/\; (\sum_{j=1}^{k} v_{ij}^2)^{1/2}.$$

The reason to consider the first eigenvectors of the affinity matrix is the following. Suppose there are k clusters in the dataset infinitely far apart from each other. One can show that in this case, the first k eigenvalues of the affinity matrix will be 1 and the eigenvalue $k+1$ will be strictly less than 1 [Ng et al., 2001]. The value of this gap depends on how well connected each cluster is: the better connected, the larger the gap is (the smaller the $k+1$st eigenvalue). Also, in the new representation in $\mathbb{R}^k$ there will be k vectors $\mathbf{z}_1, \ldots, \mathbf{z}_k$ orthonormal to each other such that each training point is mapped to one of those k points depending on the cluster it belongs to.

This simple example show that in this new representation points are naturally clustered and we suggest to train a linear classifier on the mapped points.

7.4.4 Extension of the cluster kernel

Based on the ideas of the previous section, we propose the following algorithm:

1. As before, compute the RBF matrix K from both labeled and unlabeled points (this time with 1 on the diagonal and not 0) and D, the diagonal matrix whose elements are the sum of the rows of K.
2. Compute $L = D^{-1/2}KD^{-1/2}$ and its eigendecomposition $L = U\Lambda U^\top$.
3. Given a transfer function φ, let $\tilde{\lambda}_i = \varphi(\lambda_i)$, where the λ_i are the eigenvalues of L, and construct $\tilde{L} = U\tilde{\Lambda}U^\top$.
4. Let $\tilde{D}$ be a diagonal matrix with $\tilde{D}_{ii} = 1/\tilde{L}_{ii}$ and compute $\tilde{K} = \tilde{D}^{1/2}\tilde{L}\tilde{D}^{1/2}$.

The new kernel matrix is $\tilde{K}$. Different transfer function lead to different kernels:

Linear $\varphi(\lambda) = \lambda$. In this case $\tilde{L} = L$ and $\tilde{D} = D$ (since the diagonal elements of K are 1). It turns out that $\tilde{K} = K$ and no transformation is performed.

Step $\varphi(\lambda) = 1$ if $\lambda \geq \lambda_{cut}$ and 0 otherwise. If λ_{cut} is chosen to be equal to the k-th largest eigenvalue of L, then the new kernel matrix $\tilde{K}$ is the dot product matrix in the representation of [Ng et al., 2001] described in the previous section.

Linear-step Same as the step function, but with $\varphi(\lambda) = \lambda$ for $\lambda \geq \lambda_{cut}$. This is closely related to the approach consisting in building a linear classifier in the space given by the first Kernel PCA components [Schölkopf et al., 1998b]: if the normalization matrix D and $\tilde{D}$ were equal to the identity, both approaches would be identical. Indeed, if the eigendecomposition of K is $K = U\Lambda U^\top$, the coordinates of the training points in the kernel PCA representation are given by the matrix $U\Lambda^{1/2}$.

Polynomial $\varphi(\lambda) = \lambda^t$. In this case, $\tilde{L} = L^t$ and $\tilde{K} = \tilde{D}^{1/2}D^{1/2}(D^{-1}K)^t D^{-1/2}\tilde{D}^{1/2}$. The matrix $D^{-1}K$ is the transition matrix in the random walk described in section 7.4.1 and $\tilde{K}$ can be interpreted as a normalized and symmetrized version of the transition matrix corresponding to a t steps random walk.

This makes the connection between the idea of the random walk kernel of section 7.4.1 and a linear classifier trained in a space induced by either the spectral clustering algorithm of [Ng et al., 2001] or the Kernel PCA algorithm.

How to handle test points If test points are available during training and if they are also drawn from the same distribution as the training points (an assumption which is commonly made), then they should be considered as unlabeled points and the matrix $\tilde{K}$ described above should be built using training, unlabeled and test points.

However, it might happen that test points are not available during training. This is a problem, since our method produces a new kernel matrix, but not an analytic form of the effective new kernel that could readily be evaluated on novel test points. In this case, we propose the following solution: approximate a test point $\mathbf{x}$ as a linear combination of the training and unlabeled points, and use this approximation to express the required dot product between the test point and other points in the feature space. More precisely, let

$$\boldsymbol{\alpha}^0 = \arg\min_{\boldsymbol{\alpha}} \left\| \Phi(\mathbf{x}) - \sum_{i=1}^{n+n_u} \alpha_i \Phi(\mathbf{x}_i) \right\| = K^{-1}\mathbf{v}$$

with $v_i = K(\mathbf{x}, \mathbf{x}_i)$[8]. Here, Φ is the feature map corresponding to K, i.e., $K(\mathbf{x}, \mathbf{x}') = (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}'))$. The new dot product between the test point $\mathbf{x}$ and the other points is expressed as a linear combination of the dot products of $\tilde{K}$,

$$\tilde{K}(\mathbf{x}, \mathbf{x}_i) \equiv (\tilde{K}\boldsymbol{\alpha}^0)_i = (\tilde{K}K^{-1}\mathbf{v})_i.$$

Note that for a linear transfer function, $\tilde{K} = K$, and the new dot product is the standard one.

[8] We consider here an RBF kernel and for this reason the matrix K is always invertible.

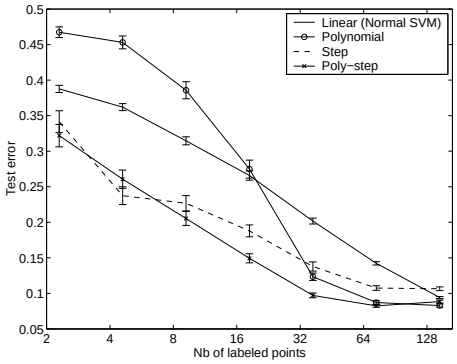


Figure 7.5: Test error on a text classification problem for training set size varying from 2 to 128 examples. The different kernels correspond to different kind of transfer functions.

7.4.5 Experiments with the cluster kernel

Influence of the transfer function

We applied the different kernel clusters of section 7.4.4 to the text classification task of [Szummer and Jaakkola, 2001], following the same experimental protocol. There are two categories `mac` and `windows` with respectively 958 and 961 examples of dimension 7511. The width of the RBF kernel was chosen as in [Szummer and Jaakkola, 2001] giving $\sigma = 0.55$. Out of all examples, 987 were taken away to form the test set. Out of the remaining points, 2 to 128 were randomly selected to be labeled and the other points remained unlabeled. Results are presented in figure 7.5 and averaged over 100 random selections of the labeled examples. The following transfer functions were compared: linear (i.e. standard SVM), polynomial $\varphi(\lambda) = \lambda^5$, step keeping only the $n+10$ where n is the number of labeled points, and poly-step defined in the following way (with $1 \geq \lambda_1 \geq \lambda_2 \geq \dots$),

$$\varphi(\lambda_i) = \begin{cases} \sqrt{\lambda_i} & i \leq n+10 \\ \lambda_i^2 & i > n+10 \end{cases}$$

For large sizes of the (labeled) training set, all approaches yield similar results. The interesting case are small training sets. Here, the step and poly-step functions work very well. The polynomial transfer function does not give good results for very small training sets (but nevertheless outperforms the standard SVM for medium sizes). This might be due to the fact that in this example, the second largest eigenvalue is 0.073 (the largest is by construction 1). Since the polynomial transfer function tends to push to 0 the small eigenvalues, it turns out that the new kernel has "rank almost one" and it is more difficult to learn with such a kernel. To avoid this problem, the authors of [Szummer and Jaakkola, 2001] consider a sparse affinity matrix with non-zeros entries only for neighbor examples. In this way the data are by construction more clustered and the eigenvalues are larger. We verified experimentally that the polynomial transfer function gave better results when applied to a sparse affinity matrix.

Concerning the step transfer function, the value of the cut-off index corresponds to the number of dimensions in the feature space induced by the kernel, since the latter is linear in the representation given by the eigendecomposition of the affinity matrix. Intuitively, it makes sense to have the number of dimensions increase with the number of training examples, that is the reason why we chose a cutoff index equal to $n + 10$.

The poly-step transfer function is somewhat similar to the step function, but is not as rough: the square root tends to put more importance on dimensions corresponding to large eigenvalues (recall that they are smaller than 1) and the square function tends to discard components with small eigenvalues. This method achieves the best results.

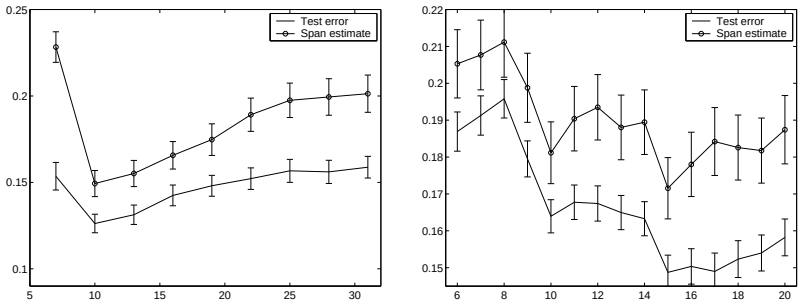


Figure 7.6: The span estimate predicts accurately the minimum of the test error for different values of the cutoff index r in the poly-step kernel (7.11). Left: text classification task, right: handwritten digit classification

Automatic selection of the transfer function

The choice of the poly-step transfer function in the previous section corresponds to the intuition that more emphasis should be put on the dimensions corresponding to the largest eigenvalues (they are useful for cluster discrimination) and less on the dimensions with small eigenvalues (corresponding to intra-cluster directions). The general form of this transfer function is

$$\varphi(\lambda_i) = \begin{cases} \lambda_i^p & i \leq r \\ \lambda_i^q & i > r \end{cases} , \tag{7.11}$$

where $p, q \in \mathbb{R}$ and $r \in \mathbb{N}$ are 3 hyperparameters. As before, it is possible to choose qualitatively some values for these parameters, but ideally, one would like a method which automatically chooses good values. It is possible to do so by gradient descent on an estimate of the generalization error (chapter 4). To assess the possibility of estimating accurately the test error associated with the poly-step kernel, we computed the span estimate in the same setting as in the previous section. We fixed $p = q = 2$ and the number of training points to 16 (8 per class). The span estimate and the test error are plotted on the left side of figure 7.6.

Another possibility would be to explore methods that take into account the spectrum of the kernel matrix in order to predict the test error [Schölkopf et al., 1999].

Comparison with other algorithms

We summarized the test errors (averaged over 100 trials) of different algorithms trained on 16 labeled examples in the following table.

Normal SVM	Transductive SVM	Random walk	Cluster kernel
27.5% ($\pm$ 7)	15.6% ($\pm$ 2.5)	15.5%	12.6% ($\pm$ 5.3)

The transductive SVM algorithm consists in maximizing the margin on both labeled and unlabeled. To some extent it implements also the cluster assumption since it tends to put the decision function in low density regions. This algorithm has been successfully applied to text categorization [Joachims, 1999] and is a state-of-the-art algorithm for performing semi-supervised learning. The result of the Random walk kernel is taken directly from [Szummer and Jaakkola, 2001]. Finally, the cluster kernel performance has been obtained with $p = q = 2$ and $r = 10$ in the transfer function 7.11. The value of r was the one minimizing the span estimate (see left side of figure 7.6).

Digit recognition

In a second set of experiments, we considered the task of classifying the handwritten digits 0 to 4 against 5 to 9 of the USPS database. The cluster assumption should apply fairly well on this database since the

different digits are likely to be clustered.

2000 training examples have been selected and divided into 50 subsets on 40 examples. For a given run, one of the subsets was used as the labeled training set, whereas the other points remained unlabeled. The width of the RBF kernel was set to 5 (it was the value minimizing the test error in the supervised case).

The mean test error for the standard SVM is 17.8% (standard deviation 3.5%), whereas the transductive SVM algorithm of [Joachims, 1999] did not yield a significant improvement ($17.6\% \pm 3.2\%$). As for the cluster kernel (7.11), the cutoff index r was again selected by minimizing the span estimate (see right side of figure 7.6). It gave a test error of 14.9% (standard deviation 3.3%). It is interesting to note in figure 7.6 the local minimum at $r = 10$, which can be interpreted easily since it corresponds to the number of different digits in the database.

It is somehow surprising that the transductive SVM algorithm did not improve the test error on this classification problem, whereas it did for text classification. We conjecture the following explanation: the transductive SVM is more sensitive to outliers in the unlabeled set than the cluster kernel methods since it directly tries to maximize the margin on the unlabeled points. For instance, in the top middle part of figure 7.7, there is an unlabeled point which would have probably perturbed this algorithm. However, in high dimensional problems such as text classification, the influence of outlier points is smaller. Another explanation is that this method can get stuck in local minima, but that again, in higher dimensional space, it is easier to get out of local minima.

7.4.6 Changing distances in input space

We finish this section by pointing out another direction of research to take into account the cluster assumption. One could try to change directly the distance in input space such that paths in regions of low density are longer. By doing so, points in different clusters will have a larger distance.

One can define a Riemanniean metric which a constant times the Euclidean metric, this constant being inversely proportional to the density,

$$ds^2 = \frac{d\mathbf{x}^2}{P(\mathbf{x})}$$

and the distance between two points is the shortest path,

$$D(\mathbf{x}(t_0), \mathbf{x}_(t_1)) = \int_{t_0}^{t_1} \frac{\|\partial_t \mathbf{x}(t)\|}{P(\mathbf{x})} dt,$$

where $t \in [t_0, t_1]$ parametrizes the path between $\mathbf{x}(t_0)$ and $\mathbf{x}(t_1)$.

Rattray [2000] proposed a pseudo-distance based on the idea that points in homogeneous regions should have a small distance:

$$ds^2 = (\log P(\mathbf{x} + d\mathbf{x}) - \log P(\mathbf{x}))^2.$$

Tipping [1999] also proposed a distance well suited for clustering, but which would not implement the cluster assumption.

7.5 Learning the structure of the data

In high dimensional space, data often lie on a manifold of much smaller dimension and ideally one should either compute the distances along the manifold (instead of using the traditional Euclidean distance) or change the data representation such that in the new space, the manifold is "unfolded" and the Euclidean distance makes more sense. The more unlabeled data, the better one can estimate this manifold. Handwritten digits have been an important topic of research in the machine learning community for the last 15 years because there are typical of high-dimensional data lying on a lower dimensional manifold. The Tangent distance [Simard et al., 1993] incorporates knowledge about invariances and attempts to follow this manifold. Using a K Nearest Neighbor algorithm with this distance gives one of the best classification

error on the MNIST dataset [LeCun et al., 1995]. The random walk kernel [Szummer and Jaakkola, 2001] described in section 7.4.1 was also designed as an attempt to follow the manifold on which the data lie.

An example of changing the data representation is the Locally Linear Embedding (LLE) algorithm [Roweis and Saul, 2000]. The authors applied it to handwritten digits and they mapped the digits to a 2-dimensional space where one can clearly see that digits 2 and 3 are well clustered and separated [Roweis et al., 2002].

More generally, many unsupervised algorithms such as Mixture of Factor Analysis [Ghahramani and Hinton, 1997; Ghahramani and Beal, 2000] or the Generative Topographic Mapping [Bishop et al., 1998] could be used as preprocessing step: they can extract from the unlabeled data a relevant representation of the data which can then be used in any supervised algorithm (see also [Seeger, 2001b] for a review).

7.5.1 Mutual information kernel

It is possible to design directly a kernel taking into account the generative model learned from the unlabeled data. We describe in this section the work of Seeger [2001a] who derives such a kernel in a Bayesian setting.

Suppose that the marginal distribution $P(\mathbf{x})$ is parametrized by some parameters $\boldsymbol{\theta}$. We will write $P(\mathbf{x}|\boldsymbol{\theta})$ the probability the point $\mathbf{x}$ has been generated with the parameters $\boldsymbol{\theta}$. There is a flat prior on $\boldsymbol{\theta}$. The training and unlabeled point are first used to compute the posterior, $P(\boldsymbol{\theta}|\{\mathbf{x}_i\})$. The following kernel has been proposed in [Seeger, 2001a]

$$K(\mathbf{x},\mathbf{y}) = \frac{Q(\mathbf{x},\mathbf{y})}{\sqrt{Q(\mathbf{x},\mathbf{x})Q(\mathbf{y},\mathbf{y})}}$$

and

$$Q(\mathbf{x},\mathbf{y}) = \int P(\mathbf{x}|\boldsymbol{\theta})P(\mathbf{y}|\boldsymbol{\theta})P(\boldsymbol{\theta}|\{\mathbf{x}_i\})^{\lambda} d\boldsymbol{\theta},$$

where λ is parameter between 0 and 1 describing how much the unlabeled data should be trusted.

A MAP approximation of the above integral leads to the *Fisher kernel* [Jaakkola and Haussler, 1998; Jaakkola et al., 1999a]. Indeed let $\hat{\boldsymbol{\theta}}$ be the MAP estimate,

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{\theta}|\{\mathbf{x}_i\}).$$

The posterior $P(\boldsymbol{\theta}|\{\mathbf{x}_i\})$ is replaced by $\mathcal{N}(\boldsymbol{\theta}|\hat{\boldsymbol{\theta}}, \mathcal{H}^{-1})$, a gaussian centered at $\hat{\boldsymbol{\theta}}$ with covariance matrix

$$\mathcal{H}_{pq} = -\frac{\partial \log P(\boldsymbol{\theta}|\{\mathbf{x}_i\})}{\partial\theta_p \partial\theta_q}\bigg|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} \propto -\sum_i \frac{\partial \log P(\mathbf{x}_i|\boldsymbol{\theta})}{\partial\theta_p \partial\theta_q}\bigg|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} \tag{7.12}$$

Also the following linear approximation is made

$$\log P(\mathbf{x}|\boldsymbol{\theta}) \approx \log P(\mathbf{x}|\hat{\boldsymbol{\theta}}) + (\boldsymbol{\theta}-\hat{\boldsymbol{\theta}})^{\top} \underbrace{\nabla_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} \log P(\mathbf{x}|\boldsymbol{\theta})}_{U_{\mathbf{x}}}.$$

After integration, we have

$$Q(\mathbf{x},\mathbf{y}) = P(\mathbf{x}|\hat{\boldsymbol{\theta}})P(\mathbf{y}|\hat{\boldsymbol{\theta}}) \exp\left(\frac{1}{2}(U_{\mathbf{x}}+U_{\mathbf{y}})^{\top}(\lambda\mathcal{H})^{-1}(U_{\mathbf{x}}+U_{\mathbf{y}})\right)$$

and

$$K(\mathbf{x},\mathbf{y}) = \exp\left(-\frac{1}{2\lambda}(U_{\mathbf{x}}-U_{\mathbf{y}})^{\top}\mathcal{H}^{-1}(U_{\mathbf{x}}-U_{\mathbf{y}})\right),$$

which is the Fisher kernel proposed in [Jaakkola et al., 1999a], since $\mathcal{H}$ is (up to a constant) equal to empirical Fisher information matrix (see equation 7.12).

Note that when the number of unlabeled points goes to infinity, the posterior $P(\boldsymbol{\theta}|\{\mathbf{x}_i\})$ is sharper and $\mathcal{H}^{-1} \to 0$. The kernel becomes equivalent to a linear kernel (as the original one proposed in [Jaakkola and Haussler, 1998]).

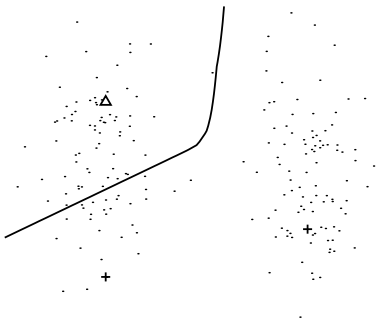


Figure 7.7: Decision function obtained by an SVM with the kernel (7.13). On this toy problem, this kernel implements perfectly the cluster assumption: the decision function cuts a cluster only when necessary.

Example Let consider that the generative model of the data is a gaussian with a fixed variance σ_0^2 and a mean $\boldsymbol{\mu}$. Then $\boldsymbol{\theta} = \boldsymbol{\mu}$ and $U_{\mathbf{x}} = (\mathbf{x} - \boldsymbol{\mu})/\sigma_0^2$. Also $\mathcal{H} = nI/\sigma_0^2$ and the Fisher kernel is just the standard RBF kernel.

A natural idea is to extend the Fisher kernel to a the case of a mixture of Gaussians, $P(\mathbf{x}|\boldsymbol{\theta}, \boldsymbol{\pi}) = \sum_k \pi_k P(\mathbf{x}|\boldsymbol{\theta}_k)$, but as noted in [Tipping, 1999; Seeger, 2001a] this idea does not work. To overcome this problem, it has been proposed in [Seeger, 2001a] to generalize $Q(\mathbf{x}, \mathbf{y})$ as

$$Q(\mathbf{x}, \mathbf{y}) = \sum_{k=1}^{n_{mixture}} \pi_k \int P(\mathbf{x}|\boldsymbol{\theta}_k) P(\mathbf{y}|\boldsymbol{\theta}_k) P(\boldsymbol{\theta}|\mathbf{x}_i)^{\lambda} d\boldsymbol{\theta}.$$

Independently, another extension of the Fisher kernel has been proposed in [Tsuda et al., 2002] which leads, in the case of a mixture of Gaussians (μ_k, Σ_k) to the *Marginalized kernel* whose behavior is similar to the mutual information kernel,

$$K(\mathbf{x}, \mathbf{y}) = \sum_{k=1}^{q} P(k|\mathbf{x}) P(k|\mathbf{y}) \mathbf{x}^{\top} \Sigma_k^{-1} \mathbf{y}. \tag{7.13}$$

To understand the behavior of the Marginalized kernel, we designed a 2D-toy problem (figure 7.7): unlabeled points are drawn from a mixture of two Gaussians whose centers are $(\pm 0.5, 0)$ and standard deviation 0.15 for this first component and 0.4 for the second one. 200 unlabeled points have been sampled from this distribution and the parameters of the mixture distribution have been learn using EM. Then 3 labeled points have been chosen and the we plotted the decision function obtained by an SVM and the Marginalized kernel (7.13).

The behavior of this decision function is intuitively very satisfying: on the one hand, when not enough label data is available, it takes into account the cluster assumption and does not cut clusters (right cluster), but on the other hand, the kernel is flexible enough to cope with different labels in the same cluster (left side).

7.5.2 Unlabeled data invariant learning

As a continuation of the ideas presented in section 7.2.4, let us write the estimated margin probability as

$$\tilde{P}(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{(2\pi)^{d/2} \sqrt{|\Sigma_i|}} \exp\left(-\frac{1}{2} (\mathbf{x} - \mathbf{x}_i)^{\top} \Sigma_i^{-1} (\mathbf{x} - \mathbf{x}_i) \right). \tag{7.14}$$

If the data lie on a manifold, Σ_i will be flat and equation (7.14) is a special case of a Mixture of Factor Analyzers [Ghahramani and Hinton, 1997] with centers $\mathbf{x}_i$ being the training points and with equal mixing parameters $1/n$. The matrices Σ_i can be optimized using the unlabeled samples and the EM algorithm.

Now, let us add the additional constraint that $\Sigma_1 = \cdots = \Sigma_n = \Sigma$. This constraint might seem very restrictive, but doing so in feature space is not so restrictive.

We propose the following algorithm,

1. Fix a kernel function K
2. Compute the empirical kernel PCA map ψ as described in section 2.2 using p randomly selected labeled or unlabeled points. For computational reasons, p should not be much more than 1000.
3. For each labeled, unlabeled and test point, compute $\tilde{\mathbf{x}} = \psi(\mathbf{x}) \in \mathbb{R}^p$. Now, linear techniques can be used on the transformed points $\tilde{\mathbf{x}}$ in $\mathbb{R}^p$.
4. Use the EM algorithm to find the matrix $\Sigma \in \mathbb{R}^{p \times p}$ which maximizes the likelihood of the unlabeled data under the model (7.14).
5. Find the hyperplane of $\mathbb{R}^p$ which minimizes the vicinal risk (1.14) with

$$P_{\mathbf{x}_i}(\mathbf{x}) = \exp\left(-\frac{1}{2}(\tilde{\mathbf{x}} - \tilde{\mathbf{x}_i})^\top \Sigma^{-1} (\tilde{\mathbf{x}} - \tilde{\mathbf{x}_i})\right).$$

As explained in chapter 6, there is a direct link between minimizing the vicinal risk with non spherical shape and invariant learning. The last step is thus just performed by training an invariant SVM, where the matrix C_γ (equation 6.2) is replaced by

$$C_\gamma = ((1 - \gamma I) + \gamma \Sigma)^{1/2}.$$

As before γ is regularization parameter, $\gamma = 0$ means normal SVM training.

Actually in practice we did not estimate Σ (step 4) using the EM algorithm because of computational reasons, but through a more direct approach involving the nearest neighbors. For each labeled point $\mathbf{x}_i$, let $\mathbf{x}_{unl(i,1)}, \ldots \mathbf{x}_{unl(i,k)}$ be the k nearest unlabeled points from $\mathbf{x}_i$. Then we defined

$$\Sigma \equiv \sum_{i=1}^{n} \sum_{j=1}^{k} (\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_{unl(i,j)})(\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_{unl(i,j)})^\top.$$

Experiments have been carried on the USPS database using a polynomial kernel of degree 3 and the task being to discriminate digits 0 to 4 from 5 to 9. The empirical kernel PCA map has been computed using a random set of $p = 500$ examples. The training set of 7291 has been divided in either 23 subset of 317 examples or 243 of 30. For each subset of $n = 30$ or 317 labeled examples, the $7291 - n$ remaining examples form the unlabeled set. Results are presented in figure 7.8 and seem very promising.

7.6 Transduction for ridge regression

A transduction algorithm [Vapnik, 1982, 1998] takes as input training examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ and a test set $\mathbf{x}_1^*, \ldots, \mathbf{x}_m^*$ and outputs the predicted targets $y_1^*, \ldots, y_m^*$. There are two reasons why such a scheme should give better test error than the usual induction (estimate the function) - deduction (evaluating the function) scheme.

1. The generalization error can be seen as the test error with an infinite number of test samples or also the expected test error. Here, the goal is not to minimize this quantity, but the *actual* test error and a transductive learning algorithm should focus on these test points.
2. The test set can be seen as an unlabeled set and all semi-supervised algorithms can take advantage of this extra information.

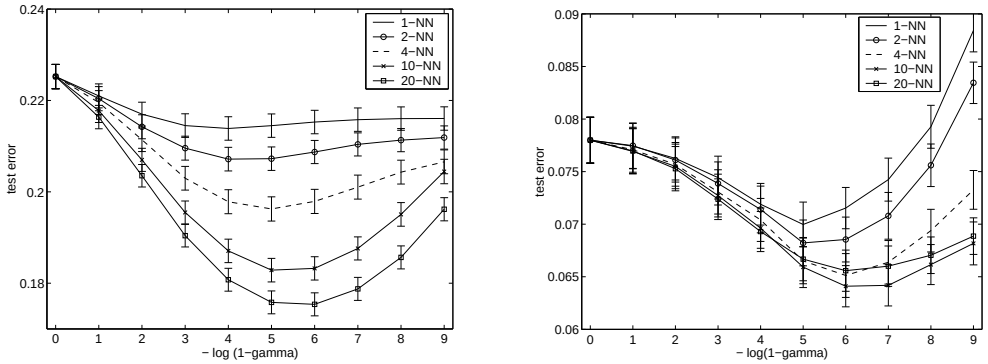


Figure 7.8: Test error on the USPS database using invariant SVMs and where invariances are computed from the k nearest neighbors ($k = 1, 2, 4, 10$ or 20. Results averaged over different training sets of size 30 (left) or 317 (right).

We would like to see transduction from the second point of view. The first reason is more a philosophical argument developed by Vapnik [1998] saying that one should not try to solve a more complicated problem than needed. This is however somehow unclear to us. Indeed, for any transductive algorithm $\mathcal{A}_{trans} : (\mathcal{X} \times \mathcal{Y})^n \times \mathcal{X}^m \mapsto \mathcal{Y}^m$, there exists an associated inductive algorithm defined as follows:

$$\begin{array}{rccl} \mathcal{A}_{ind}: & (\mathcal{X} \times \mathcal{Y})^n & \mapsto & \mathcal{Y}^{\mathcal{X}} \\ & \{(\mathbf{x}_i, \mathbf{y}_i)\} & \rightarrow & f(\mathbf{x}) = \mathcal{A}_{trans}(\{(\mathbf{x}_i, \mathbf{y}_i)\}, \mathbf{x}). \end{array}$$

Thus for $m = 1$, there is not any better transductive algorithm than the best inductive one. For $m > 1$, there can be improvement, but in the case of an i.i.d. setting this improvement will most likely comes from the additional information the points $\mathbf{x}_1^*, \ldots, \mathbf{x}_m^*$ carry about the marginal distribution $P(\mathbf{x})$.

7.6.1 Ridge Regression and Leave-One-Out error

The Ridge Regression algorithm has been described in section 5.4 and there exists a closed form for its leave-one-out error (5.8). To perform transductive inference with this algorithm, the following method has been suggested [Chapelle et al., 1999b]: predict $(y_1^*, \ldots, y_m^*)$ by finding those values which minimize the leave-one-out error of Ridge Regression training on the joint set

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n), (\mathbf{x}_1^*, y_1^*), \ldots, (\mathbf{x}_m^*, y_m^*). \tag{7.15}$$

This is achieved in the following way. Suppose we treat the unknown values $(y_1^*, \ldots, y_m^*)$ as variables and for some fixed value of these variables we minimize the following empirical functional

$$R_{emp}(\boldsymbol{\alpha}|y_1^*, \ldots, y_m^*) = \frac{1}{n+m}\left(\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i, \boldsymbol{\alpha}))^2 + \sum_{i=1}^{m}(y_i^* - f(\mathbf{x}_i^*, \boldsymbol{\alpha}))^2\right) + \gamma \left\|\boldsymbol{\alpha}\right\|^2. \tag{7.16}$$

This functional differs only in the second term from the functional (5.7) and corresponds to performing Ridge Regression with the extra pairs $(\mathbf{x}_1^*, y_1^*), \ldots, (\mathbf{x}_m^*, y_m^*)$. Suppose that vector $Y^* = (y_1^*, \ldots, y_m^*)$ is taken from some set $Y^* \in \mathcal{Y}$ such that the pairs

$$(\mathbf{x}_1^*, y_1^*), \ldots, (\mathbf{x}_m^*, y_m^*)$$

can be considered as a sample drawn from the true distribution $P(\mathbf{x}, y)$. In this case the leave-one-out error of minimizing (7.16) over the set (7.15) approximates the generalization error. We can measure this leave-one-out error using the same technique as in the standard Ridge Regression. Using the closed form (5.8) one obtains

$$T_{loo}(\gamma, y_1^*, \ldots, y_m^*) = \frac{1}{n+m}\sum_{i=1}^{n+m}\left(\frac{\hat{Y}_i - \hat{k}_i^\top \hat{A}_\gamma^{-1}\hat{K}^\top \hat{Y}}{1 - \hat{k}_i^\top \hat{A}_\gamma^{-1}\hat{k}_i}\right)^2. \tag{7.17}$$

where we denote the extended "training set" $\hat{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{x}_1^*, \ldots, \mathbf{x}_m^*)$, $\hat{Y} = (y_1, \ldots, y_\ell, y_1^*, \ldots, y_m^*)^\top$, and

$$\hat{A}_\gamma^{-1} = (\hat{K}^\top \hat{K} + \gamma I)^{-1}$$

$$\hat{K}_{ij} = \varphi_j(\hat{\mathbf{x}}_i), \quad i = 1, \ldots, n+m, \quad j = 1, \ldots, p.$$

$$\hat{k}_i = (\varphi_1(\hat{\mathbf{x}}_i) \ldots, \varphi_p(\hat{\mathbf{x}}_i))^\top.$$

Now let us rewrite the expression (7.17) in an equivalent form to separate the terms with $\hat{Y}$ from the terms with x.

Introducing

$$C = I - \hat{K} A_\gamma^{-1} \hat{K}^\top,$$

and the matrix M with elements

$$M_{ij} = \sum_{k=1}^{n+m} \frac{C_{ik} C_{kj}}{{C_{kk}}^2}$$

we obtain the equivalent expression of (7.17)

$$T_{loo}(\gamma, y_1^*, \ldots, y_m^*) = \frac{1}{n+m} \sum_{i=1}^{n+m} \left[\frac{(C\hat{Y})_i}{C_{ii}} \right]^2 = \frac{\hat{Y}^\top M \hat{Y}}{n+m}. \tag{7.18}$$

In order for the Y^* which minimize the leave–one–out procedure to be valid it is required that the pairs $(\mathbf{x}^*, y^*)$ are drawn according to the distribution $P(\mathbf{x}, y)$. To satisfy this constraint we choose vectors Y^* from the set

$$\mathcal{Y} = \{Y^* : ||Y^* - Y^0|| \leq R\} \tag{7.19}$$

where the vector Y^0 is the solution obtained from classical Ridge Regression.

To minimize (7.18) under constraint (7.19) we use the functional

$$T_{loo}^{\gamma^*}(\gamma) = \hat{Y}^\top M \hat{Y} + \gamma^* ||Y^* - Y^0||^2 \tag{7.20}$$

where γ^* is a constant depending on R.

Now, to find the values at the given points of interest $\mathbf{x}_i^*$ all that remains is to find the minimum of (7.20) in Y^*. Note that the matrix M is obtained using only the vectors $\mathbf{x}$ and $\mathbf{x}^*$. Therefore, to find the minimum of this functional we rewrite Equation (7.20) as

$$T_{loo}^{\gamma^*}(\gamma) = Y^\top M_0 Y + 2{Y^*}^\top M_1 Y + {Y^*}^\top M_2 Y^* + \gamma^* ||Y^* - Y^0||^2 \tag{7.21}$$

where

$$M = \begin{pmatrix} M_0 & M_1 \\ M_1^\top & M_2 \end{pmatrix}$$

and M_0 is a $n \times n$ matrix, M_1 is a $n \times m$ matrix and M_2 is a $m \times m$ matrix. Taking the derivative of (7.21) in Y^* we obtain the condition for the solution

$$2M_1 Y + 2M_2 Y^* - 2\gamma^* Y^0 + 2\gamma^* Y^* = 0$$

which gives the predictions

$$Y^* = (\gamma^* I + M_2)^{-1}(-M_1 Y + \gamma^* Y^0).$$

In this algorithm (which we will call Transductive Regression) we have two parameters to control: γ and γ^*. The choice of γ can be found using the leave-one-out estimator (5.8) for Ridge Regression. This leaves γ^* as the only free parameter, the choice of which can be made according to the variance of the predictions Y^0.

7.6.2 Experiments

To compare our transductive approach with the classical approach we performed a series of experiments on regression problems. We also describe experiments applying our technique to the problem of pattern recognition.

Regression

We conducted computer simulations for the regression problem using two datasets from the DELVE repository: **boston** and **kin-32fh**.

The **boston** dataset is a well-known problem where one is required to estimate house prices according to various statistics based on 13 locational, economic and structural features from data collected by the U.S Census Service in the Boston Massachusetts area.

The **kin-32fh** dataset is a realistic simulation of the forward dynamics of an 8 link all-revolute robot arm. The task is to predict the distance of the end-effector from a target, given 32 inputs which contain information on the joint positions, twist angles and so forth.

Both problems are nonlinear and contain noisy data. Our objective is to compare our transductive inference method directly with the inductive method of Ridge Regression. To do this we chose the set of basis functions

$$\varphi_i(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{2\sigma^2}\right), \quad i = 1, \ldots, n,$$

and found the values of γ and σ for Ridge Regression which minimized the leave-one-out bound (5.8). We then used the same values of these parameters in our transductive approach and added m basis functions corresponding to the points $\mathbf{x}_1^*, \ldots, \mathbf{x}_m^*$. We then chose a fixed value of γ^*.

For the **boston** dataset we followed the same experimental setup as in [Saunders et al., 1998], that is, we partitioned the training set of 506 observations randomly 100 times into a training set of 481 observations and a testing set of 25 observations. We chose the values of γ and σ by taking the minimum average leave–one–out error over five more random splits of the data stepping over the parameter space. The minimum was found at $\gamma = 0.005$ and $\log\ \sigma = 0.7$. For our transductive method, we also chose the parameter $\gamma^* = 10$. In Figure 7.9(a), we plot mean squared error (MSE) on the test set averaged over the 100 runs against $\log\ \sigma$ for Ridge Regression and Transductive Regression. Transductive Regression outperforms Ridge Regression, especially at the minimum.

To observe the influence of the number of test points m on the generalization ability of our transductive method, we ran further experiments, setting $\gamma^* = n/2m$ for different values of m. In Figure 7.9(b) we plot m against MSE on the testing set, at $\log \sigma = 0.7$. The results clearly indicate that increasing the test set size gives improved performance in Transductive Regression. For Ridge Regression, of course, the size of the testing set has no influence on the generalization ability.

We then performed similar experiments on the **kin-32fh** dataset. This time, as we were interested in large testing sets giving improved performance for Transductive Regression we chose 100 splits where we took a subset of only 64 observations for training and 256 for testing. Again the leave–one–out estimator was used to find the values $\gamma = 0.1$ and $\log\ \sigma = 2$ for Ridge Regression, and for Transductive Regression we also chose the parameter $\gamma^* = 0.1$. We plotted MSE on the testing set against $\log\ \sigma$ (Figure 7.9c) and the size of the test set m for $\log\ \sigma = 2.75$ (also, $\gamma^* = 50/m$) (Figure 7.9 d) for the two algorithms. For large test set sizes our method clearly outperforms Ridge Regression.

Pattern Recognition

This technique can also be applied for pattern recognition problems by solving them based on minimizing functional (5.7) with $y = \pm 1$. Such a technique is known as a Linear Discriminant (LD) technique.

Table 7.6.2.0 describes results of experiments on classification in the following problems: 2 class digit recognition on the USPS database ($0-4$ versus $5-9$) splitting the training set into 23 runs of 317 observations and considering a testing set of 2000 observations, and six problems from the UCI database. We followed the same experimental setup as in [Rätsch et al., 2001]: the performance of a classifier is measured by its average error over one hundred partitions of the datasets into training and testing

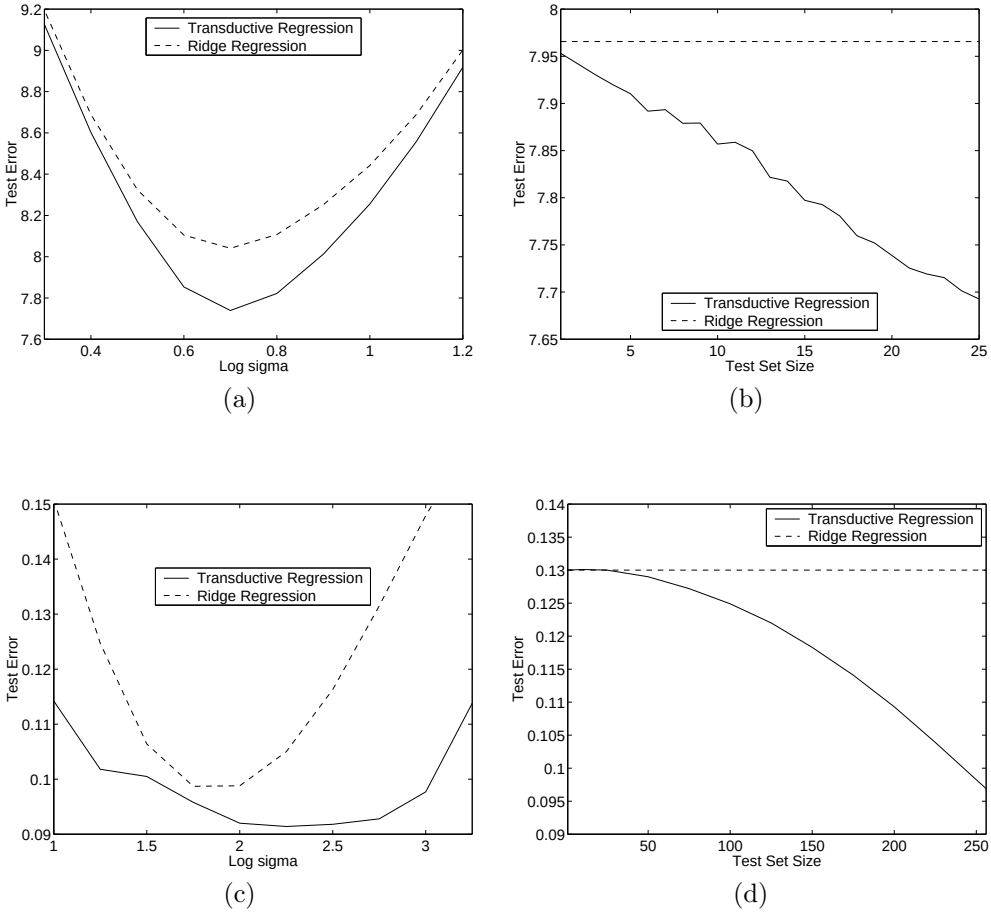


Figure 7.9: *A comparison of Transductive Regression to Ridge Regression on the* **boston** *dataset: (a) error rates for varying σ, (b) varying the test set size, m, and on the* **kin-32fh** *dataset: (c) error rates for varying σ, (d) varying the test set size.*

	AB	AB_R	SVM	TLD
Postal	–	–	5.5	4.7
Banana	12.3	10.9	11.5	11.4
Diabetes	26.5	23.8	23.5	23.3
Titanic	22.6	22.6	22.4	22.4
Breast Cancer	30.4	26.5	26.0	25.7
Heart	20.3	16.6	16.0	15.7
Thyroid	4.4	4.6	4.8	4.0

Table 7.1: *Comparison of percentage test error of AdaBoost (AB), Regularized AdaBoost (AB_R), Support Vector Machines (SVM) and Transductive Linear Discrimination (TLD) on seven datasets.*

sets. Free parameter(s) are chosen via validation on the first five training datasets. The performance of the transductive LD technique was compared to Support Vector Machines, AdaBoost and Regularized AdaBoost [Rätsch et al., 2001].

It is interesting to note that in spite of the fact that LD technique is one of the simplest pattern recognition techniques, transductive inference based upon this method performs well compared to state of the art methods of pattern recognition.

7.6.3 Discussion

We have presented a method which predicts the targets of the test points by finding those which give afterward the lowest leave-one-out error. This method clearly takes advantage of the extra information carried by the unlabeled (test) points, since in the experiments, it was shown that the test error decreases when the testing set size increases. However, the theoretical reasons for this method to work are not completely understood yet.

Chapter 8

Active Learning

8.1 Introduction

The main goal in machine learning is to estimate a function based on a given training set. *Active learning* is an extension of this framework where the learning machine does not only receive the training points passively, but also selects the points to be included in the training set. An active learner may start with a small training set and at each iteration carefully selects one or several points for which it requires labels.

Optimal Experimental Design [Fedorov, 1972] is closely related to active learning as it attempts to find a set of points such that the variance of the estimate is minimized. In contrast to this "batch" formulation, the term *active learning* often refers to an incremental strategy [Roy and McCallum, 2001; Sugiyama and Ogawa, 2000; Cohn et al., 1995; Sung and Niyogi, 1995; MacKay, 1992]:

Definition 8.1 (Aim of pool-based active learning) *Consider a learning algorithm $\mathcal{A}$ which takes as input a training set $(\mathbf{x}_i, y_i)_{1 \leq i \leq n}$ and returns a function $\hat{f}_n$. Given the same training points and a set of unlabeled points $(\mathbf{x}'_i)$, a greedy active learning strategy for this algorithm consists of selecting a point $\mathbf{x}'_i$ such that after this point is labeled, the expected generalization error of the function $\hat{f}_{n+1}$ learned with this additional point, is as small as possible.*

Note that in this standard definition of pool-based active learning, the search is greedy: at each iteration, the goal is to find *one* point which will result in the smallest expected generalization error when added to the training set.

It has been proven in [Eisenberg and Rivest, 1990] that under some smoothness conditions of the underlying probability distribution, active learning is more efficient than the passive version (where the algorithm gets random examples drawn from $P(\mathbf{x}, y)$): the number of learning examples to reach a certain accuracy is reduced.

8.2 The optimal active learning strategy

Let $D = (\mathbf{x}_i, y_i)_{1 \leq i \leq n}$ be training sample. Suppose that we can estimate the generalization error of function learned by our algorithm on this training set. Let us denote by $T(D)$ such an estimate.

The optimal active learning strategy would be the following [Schohn and Cohn, 2000]:

1. Train the classifier using the current training set
2. Fix a point $\mathbf{x}$ in the unlabeled set
   (a) Fix a label y and add the point $(\mathbf{x}, y)$ in the training set
   (b) Retrain the classifier with the additional point.
   (c) Estimate the generalization error $T(D \cup (\mathbf{x}, y))$.

(d) Estimate the probability $\tilde{P}(y|\mathbf{x})$

(e) Compute the expected generalization error $\bar{T}_{\mathbf{x}} = \sum_y \tilde{P}(y|\mathbf{x}) T(D \cup (\mathbf{x}, y))$.

3. Choose for labeling the point $\mathbf{x}$ which has the lowest expected generalization error $\bar{T}_{\mathbf{x}}$ and add it to training set.

There are several problems with this optimal strategy. Beside computational difficulties (at a first sight, a lot of retrainings are necessary), the fundamental problem is how to compute T and $\tilde{P}(y|\mathbf{x})$.

The answer of this problem depends of course on the learning algorithm. The posterior probability is usually estimated using the classifier itself. For SVMs, Platt [2000] proposed a posterior probability which is function of the distance to the hyperplane.

The estimation of the generalization error T is a more subtle problem. Indeed, most estimates make the assumption that the test samples and the training samples are drawn from the same distribution. But in active learning, points are selected and this assumption does not hold anymore. To illustrate this fact, consider the leave-one-out procedure. This is an unbiased estimator of the generalization error under the assumption that the training set is i.i.d. from the true distribution. However points which give the smallest expected leave-one-out error are the points whose labels are already almost certain. This choice of T would tend to select points which are far away from the decision boundary, which is counter-intuitive.

An other example is the PAC-Bayesian bound presented in [McAllester, 1998] which depends on the log of the measure of the version space: for a given training set, the expected generalization error is smaller if the version space is larger. However, for active learning, one wants intuitively to have a version space as small as possible as it reduces the uncertainty of the function to estimate [Tong and Koller, 2001].

While several heuristics have been proposed for active learning corresponding to some intuitive reasons, two papers, at least, tried to minimize the generalization error directly and propose some estimate of it.

- In regression and in a Bayesian framework, the *expected integrated squared difference* has been proposed [Sung and Niyogi, 1995],

$$T = \int_{f \in \mathcal{F}} P(f|D) \left(\int (f(\mathbf{x}) - \hat{f}_n(\mathbf{x}))^2 dP(\mathbf{x}) \right) df, \tag{8.1}$$

  where $P(f|D)$ is the posterior probability of f given the training data.

- For classification, the posterior probability can be used directly [Roy and McCallum, 2001],

$$T = \frac{1}{n_u} \sum_{i=1}^{n_u} \left(1 - \max_{y \in \{-1,1\}} P(y|\mathbf{x}'_i, \hat{f}_n) \right), \tag{8.2}$$

  where $\mathbf{x}'_1, \ldots, \mathbf{x}'_{n_u}$ is the set of unlabeled data and $P(y|x, \hat{f})$ is an estimate of the posterior probability for the point $\mathbf{x}$ given the function $\hat{f}$ learned on the training set.

We are going to see how to adapt those two criterion for linear SVMs, but before let us consider the simple Parzen window classifier.

8.3 Active learning with Parzen window classifier

The optimal strategy using equation (8.2) can be directly implemented for the Parzen window classifier. Indeed the posterior probability is

$$\tilde{P}(y = 1|\mathbf{x}) = \frac{\sum_{i=1}^n K(\mathbf{x}, \mathbf{x}_i) \mathrm{I}_{y_i = 1}}{\sum_{i=1}^n K(\mathbf{x}, \mathbf{x}_i)},$$

with $K(\mathbf{x}, \mathbf{x}_i)$ a gaussian kernel such as $\exp(-\|\mathbf{x} - \mathbf{x}_i\|^2 / 2\sigma^2)$. It is thus possible to compute the estimated generalization error using (8.2) and perform the optimal active learning strategy described above.

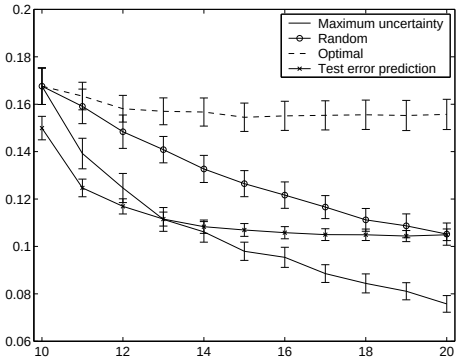


Figure 8.1: Evolution of the test error as a function of the training set size for 3 strategies: random, maximum uncertainty and optimal. The expected generalization error as computed in step (e) of section 8.2 is also plotted.

However, the accuracy of the estimated posterior probability depends on the width σ whose choice is critical. If σ is too large, $\tilde{P}(y = 1|\mathbf{x})$ tends toward the fraction of positive points in the training set, whereas if it is too small, it tends toward 0 or 1 according to the label of the nearest training point from $\mathbf{x}$. One might hope that even if this problem causes the generalization error not to be estimated accurately, the unlabeled point chosen by this active learning strategy is still a good candidate.

A toy experiment has been conducted: points are generated uniformly from $[-1, 1]^2$ and the true decision boundary is a line, $y = \text{sgn}(x_1 + x_2)$. The original training set consists of 10 points and 20 points to be labeled are selected one by one from a pool of 200 unlabeled points. At each iteration, the width of Parzen window estimate is chosen by maximizing the likelihood on the unlabeled points, as explained in the experimental section of 7.2.2. The results averaged 100 times are presented in figure 8.1. Three strategies are compared: random corresponds to passive learning since a random point is drawn from the unlabeled pool at each iteration; maximum uncertainty corresponds to the intuitive heuristic consisting in choosing the unlabeled point whose posterior probability is the nearest from $1/2$; finally optimal corresponds to the strategy presented in the previous section.

The performance of the optimal strategy is terrible since it is even worse than random. The reason is that it depends heavily on the estimation of the posterior probabilities through equation (8.2) and the step (d) of the algorithm (see section 8.2). They might not be accurate because of a combination of the three following reasons:

1. The width of the Parzen window classifier chosen to maximizing the likelihood of the unlabeled data is actually not optimal.

2. As for many classifier, estimation of posterior probabilities tend to be biased because of the learning step itself.

3. As mentioned in the previous section, after several iterations of the active learning process, the training set can not be considered anymore as drawn from the true distribution $P(\mathbf{x}, y)$ since the algorithm introduced a bias by selecting the example.

The first reason is just specific to the Parzen window classifier and our method to choose its width. The second one is more general and, for example, the authors of [Roy and McCallum, 2001] also encountered this difficulty when applying active learning to a Naive Bayes classifier. They proposed a *bagging* approach [Breiman, 1996] in oder to have smoother posterior distributions. Finally the third difficulty is a deep and fundamental one. It would require a theoretical analysis of non i.i.d data, which is not standard in the learning theory framework and we are not aware of any attempt in this direction. In the literature of active learning, most people ignore this problem. For instance, in a Bayesian setting [Sung and Niyogi,

1995], the posterior probability is computed as

$$P(y|\mathbf{x}, D_n) \propto \int_{f \in \mathcal{F}} P(D_n, (\mathbf{x}, y)|f) P(f) df,$$

but in general $P(D_n, (\mathbf{x}, y)|f)$ can not be written as a product because of the dependencies in the training set $D_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ caused by the active learning strategy.

8.4 Estimating posterior probabilities for SVMs

In this section, we will see how to estimate the posterior probability $P(y|\mathbf{x})$ from the decision function computed by an SVM. Intuitively, the further a point is from the decision boundary, the more confident we are about its label. So it makes sense to try to estimate this posterior probability in a parametric way.

8.4.1 Platt's method

As discussed in section 4.2, Platt proposed to estimate the posterior distribution $P(y = 1|\mathbf{x})$ of an SVM output $f(\mathbf{x})$ [Platt, 2000]:

$$\tilde{P}_{A,B}(\mathbf{x}) = \tilde{P}(y = 1|\mathbf{x}) = \frac{1}{1 + \exp(Af(\mathbf{x}) + B)},$$

Given a validation set $(\mathbf{x}'_i, y'_i)_{1 \leq i \leq n_v}$, one can find the constants A and B are found by minimizing the Kullback-Liebler divergence between $\tilde{P}$ and an empirical estimation of P built from this validation set:

$$(A^*, B^*) = \arg\max_{A,B} \sum_{i=1}^{n_v} \left(\frac{1 + y'_i}{2} \log(\tilde{P}_{A,B}(\mathbf{x}'_i)) + \frac{1 - y'_i}{2} \log(1 - \tilde{P}_{A,B}(\mathbf{x}'_i)) \right). \tag{8.3}$$

This optimization is carried out using a second order gradient descent algorithm [Platt, 2000].

8.4.2 How to fit the sigmoid

In section 4.2, three methods to fit the sigmoid have been suggested

Validation set That is the original method proposed in [Platt, 2000]: one uses a validation set $(\mathbf{x}'_i, y'_i)_{1 \leq i \leq n_v}$ to minimize functional (8.3).

Leave-one-out If no validation set is available, one can use the training set, but replace $f(\mathbf{x}_i)$ by

$$f(\mathbf{x}_i) - y_i \alpha_i S_i^2,$$

which estimates the output on $\mathbf{x}_i$ of the function learned if the point would not have been in the training set.

Vicinal Risk Find A and B which minimize

$$\int \frac{1 + y}{2} \log(\tilde{P}_{A,B}(\mathbf{x})) + \frac{1 - y}{2} \log(1 - \tilde{P}_{A,B}(\mathbf{x})) \, dP_{est}(\mathbf{x}, y). \tag{8.4}$$

Ideally, one would like to integrate over the true distribution $P(\mathbf{x}, y)$, but we replaced it by the estimated one $P_{est}(\mathbf{x}, y)$ used in the vicinal risk (1.13).

The problem with the vicinal risk method is that the integral (8.4) is difficult to compute. Indeed if P_{est} is the classical Parzen window estimator,

$$P_{est}(\mathbf{x}, y) = \frac{1}{n} \sum_{i=1, \quad y_i = y}^{n} \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma^2} \right),$$

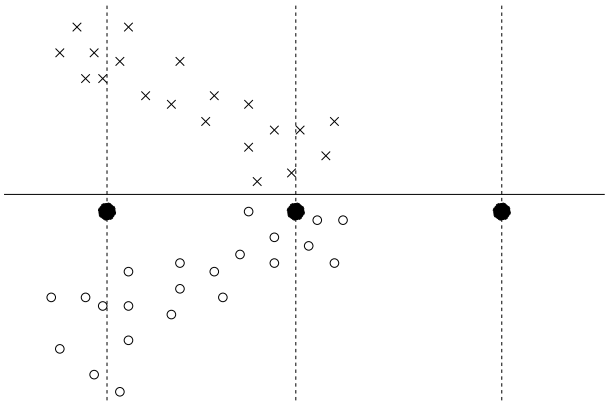


Figure 8.2: Consider three points where we want to evaluate the posterior distribution (3 black circles). They are at equal distance to the hyperplane, but the one on left is very likely to be a circle, whereas the one on the right should have a posterior probability near 0.5

one ends up with integrals of the form

$$\int \log(1+\exp(A(t+t_0)+B))\exp(-t^2/2\sigma^2)dt,$$

which can not be computed analytically. However, similarly to noise injection, it is possible to draw random samples from $P_{est}(\mathbf{x}, y)$ and approximate the integral by the sum on those samples. The width σ can be adjusted by maximizing the likelihood on the unlabeled data (see the experimental section of 7.2.2).

8.4.3 Local fit

One of the drawback of the parametric form presented above is that the posterior probability depends only on the distance to the hyperplane. It can happen that the coefficients A and B of the sigmoid need to be adapted depending on the localization of the point where the posterior probability has to be evaluated (see figure 8.2).

Let $\mathbf{x}$ bet the point of interest. We want to estimate the posterior distribution along the line $\mathbf{x}+t\mathbf{w}$ [Vapnik, 1998]. To do so, at each training point is associated a weighting coefficient which takes into account how far is the point from this line. The squared distance of a point $\mathbf{x}_i$ to the line passing by $\mathbf{x}$ and orthogonal to $\mathbf{w}$ is

$$d^2(\mathbf{x}, \mathbf{w}, \mathbf{x}_i) = \|\mathbf{x}-\mathbf{x}_i\|^2 - \left(\frac{\mathbf{w}\cdot(\mathbf{x}-\mathbf{x}_i)}{\|\mathbf{w}\|}\right)^2.$$

Functional (8.3) becomes

$$\sum_{i=1}^{n_u} \exp\left(-\frac{d^2(\mathbf{x}, \mathbf{w}, \mathbf{x}_i)}{2\sigma^2}\right)\left(\frac{1+y'_i}{2}\log(\tilde{P}_{A,B}(\mathbf{x}'_i)) + \frac{1-y'_i}{2}\log(1-\tilde{P}_{A,B}(\mathbf{x}'_i))\right),$$

where σ needs to be adjusted. The values of A and B maximizing this likelihood depend of course on the point $\mathbf{x}$ where the posterior probability needs to be evaluated. This method requires thus one optimization for each posterior probability that need to be evaluated and is quite computationally expensive.

In the rest of this chapter, we will compute the posterior probabilities by fitting the sigmoid (8.3) with the leave-one-out outputs.

8.5 Reducing the version space

In this section, we will assume that there is no noise in the data, which means that there exists a decision function which separates perfectly the data. Also since we consider only linear SVMs, let us suppose that this target function is linear.

A simple probabilistic interpretation of SVMs could be the following (for more details, see [Sollich, 2002]):

$$P(f|D) \propto \begin{cases} 1 & \text{if } f \text{ separates } D \\ 0 & \text{otherwise} \end{cases}$$

Using this simple posterior probability and replacing $\left\| f - \hat{f}_n \right\|$ by 1, equation 8.1 just turns out to give the size of the version space (defined as the set of functions which classify correctly the training data). Intuitively this measure makes sense: if we know that there is a function classifying correctly the data, the generalization error should be smaller as the size of the version space decreases. At the limit, if there is only one function left in the version space, we are sure to have selected the good one.

Note that we replaced $\left\| f - \hat{f}_n \right\|$ by 1 because the calculation of the integral might be difficult. However, using Stokes's theorem [Spivak, 1999], it might be possible to carry it out.

8.5.1 Version space and SVMs

One intuitive way to reduce the version space is to query points which are near the current decision boundary. For SVMs, this approach has been tested by [Schohn and Cohn, 2000; Campbell et al., 2000; Tong and Koller, 2001]

The size of the version space is lower bounded by a function of the margin [Herbrich et al., 2001]. As a consequence, one can try to reduce the size of the version space by reducing the size of the margin. Of course, the label of an unlabeled point is unknown. However in the worst case, adding a point which is near the current decision boundary will reduce certainly the margin, whatever label it is associated with.

This is the motivation for the heuristic [Schohn and Cohn, 2000; Campbell et al., 2000] which consists in selecting the point nearest from the hyperplane. But this heuristic comes from several approximations. Tong and Koller [2001] took one step back and tried to halve the version space in the following way:

1. Consider an unlabeled point $\mathbf{x}$
   (a) Add $(\mathbf{x}, 1)$ to the training set, retrain the SVM and compute the margin m^+.
   (b) Likewise, compute the margin m^- after adding $(\mathbf{x}, -1)$ to the training set.
   (c) Compute $score(\mathbf{x}) = 1/\max(m^+, m^-)$ or $\min(m^+/m^-, m^-/m^+)$.
2. Selecting for labeling the point $\mathbf{x}$ with the highest score.

The problem with this approach is that it is computationally expensive. Indeed, for each unlabeled point, one has to perform two SVM trainings (one with a positive label and one with a negative one) and compute the resulting margins.

In this section, we propose a computational improvement: approximate the new margin when a point is added without perform a full SVM training. The idea is to use the techniques of section 3.2. To compute efficiently the leave-one-out error, we developed methods which made it possible to approximate it without explicitly retraining an SVM after a point has been removed from the training set. Likewise, we will see in this section that it is possible to approximate the solution with an additional point without performing an explicit training. A similar derivation can also be found in [Cauwenberghs and Poggio, 2000].

8.5.2 Incremental training

As in the derivation of the span-bound (section 3.2), we make the assumption that the set of support vectors will not change after adding a new training example. Let $(\mathbf{x}, y)$ be the new point. If $y(\mathbf{w}\cdot\mathbf{x}+b) \geq 1$,

the new point does not violate Kuhn-Tucker conditions and the the solution will not change. However, if $y(\mathbf{w} \cdot \mathbf{x} + b) < 1$ it will be support vector.

Let us suppose that the points which are not support vectors are removed from the training set. We use the following notations: $(\boldsymbol{\alpha}^0, b^0)$ are the initial SVM parameters; $(\boldsymbol{\alpha}^{n+1}, b^{n+1})$ are the parameters after the point $(\mathbf{x}, y)$ has been added. Let us suppose that the index of the this point is 0.

The assumption that the set of support vectors does not change leads to the following equality expressing that those support vectors lie on the margin after retraining:

$$\underbrace{\begin{pmatrix} K(\mathbf{x}_0, \mathbf{x}_0) & \mathbf{v}^\top \\ \mathbf{v} & \mathbf{H} \end{pmatrix}}_{\tilde{\mathbf{H}}} \begin{pmatrix} \boldsymbol{\alpha}^{n+1} \\ b^{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{1} \\ 0 \end{pmatrix}, \tag{8.5}$$

where H is a modified matrix of dot products between support vectors which has been defined by equation (4.8) and $\mathbf{v}_i = y_i y_0 K(\mathbf{x}_i, \mathbf{x}_0), \mathbf{v}_{n+1} = y_0$.

Remember that the Kuhn-Tucker conditions for $\boldsymbol{\alpha}^0$ imply

$$(\boldsymbol{\alpha}^0, b^0)^\top = \mathbf{H}^{-1}(1 \cdots 1\ 0)^\top. \tag{8.6}$$

In the hard margin case, the invert margin square $\mathbf{w}^2$ is equal to the sum of the Lagrange multipliers. After adding the point $(\mathbf{x}_0, y_0)$ it becomes $\sum_{i=0}^{n} \alpha_i^{n+1}$.

To compute $\boldsymbol{\alpha}^{n+1}$, one can use equation (8.5) and invert the matrix $\tilde{\mathbf{H}}$. However the inversion of such a matrix is almost as expensive as an SVM training. To factorize the computations for different unlabeled data, we can compute the invert of $\tilde{\mathbf{H}}$ using block matrix manipulations:

$$\tilde{\mathbf{H}}^{-1} = \begin{pmatrix} 0 & \mathbf{0}^\top \\ \mathbf{0} & \mathbf{H}^{-1} \end{pmatrix} + S_0^{-2} \begin{pmatrix} 1 & -\mathbf{v}^\top \mathbf{H}^{-1} \\ -\mathbf{H}^{-1}\mathbf{v} & \mathbf{H}^{-1}\mathbf{v}\mathbf{v}^\top \mathbf{H}^{-1} \end{pmatrix}, \tag{8.7}$$

where S_0 is the span (see section 3.2.4) of the point $\mathbf{x}_0$,

$$S_0^2 = K(\mathbf{x}_0, \mathbf{x}_0) - \mathbf{v}^\top \mathbf{H}^{-1} \mathbf{v}. \tag{8.8}$$

From equation (8.5), the new $\mathbf{w}^2$ is

$$\sum_{i=0}^{n} \alpha_i^{n+1} = (\mathbf{1}^\top 0) \tilde{\mathbf{H}}^{-1} (\mathbf{1}^\top 0)^\top \tag{8.9}$$

Besides, from equation (8.6)

$$\mathbf{v}^\top \mathbf{H}^{-1} (\mathbf{1}^\top 0) = y_0 f(\mathbf{x}_0) \tag{8.10}$$

Finally, combining equations (8.9), (8.7), (8.10) and (8.6) gives

$$\sum_{i=0}^{n} \alpha_i^{n+1} = \sum_{i=1}^{n} \alpha_i^0 + \frac{(1 - y_0 f(\mathbf{x}_0))^2}{S_0^2}. \tag{8.11}$$

Based on equation (8.11), the "margin reduction" after adding the point $(\mathbf{x}_0, y_0)$ is

$$\frac{(1 - y_0 f(\mathbf{x}_0))_+^2}{S_0^2},$$

where $x_+ = x$ if $x > 0$ and 0 otherwise (to take into account that if $y_0 f(\mathbf{x}_0) > 1$ nothing happens).

Several comments on this quantity:

- Our main motivation was computational efficiency. We achieved this goal since for any point $(\mathbf{x}_0, y_0)$, we just have to compute the *span* of this point through equation (8.8) and the matrix $\mathbf{H}$ can be inverted beforehand. If there are m unlabeled points, the complexity is reduced from $O(n^3 m)$ to $O(n^2(m+n))$.

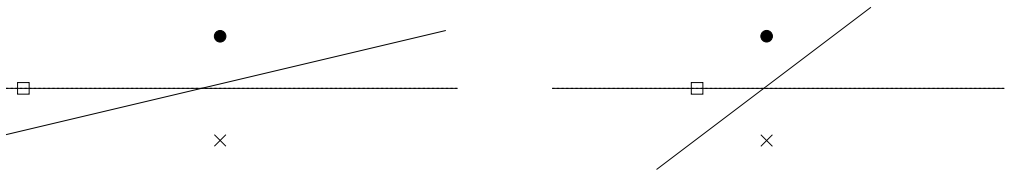


Figure 8.3: In both cases, the point $\mathbf{x}_0$ lie on the decision surface, but it seems preferable to consider the point on the right since it will reduce more the margin (and hence reduce the version space more). This fact is taken into account by the *span* S_0, which is defined as the distance between the point $\mathbf{x}_0$ and the convex hull of the other support vectors. On the right, the span of $\mathbf{x}_0$ is smaller than on the left, yielding a larger decrease in the margin (equation 8.11).

- The term $(1 - y_0 f(\mathbf{x}_0))_+^2$ has an easy interpretation: the more $(\mathbf{x}_0, y_0)$ violates the inequality $y_0 f(\mathbf{x}_0) \geq 1$, the smaller the margin will be after adding this point. This is directly related to the heuristic which consists in choosing the point which is nearest from the hyperplane [Schohn and Cohn, 2000; Campbell et al., 2000].
- The span is a correcting term to this interpretation which is is similar to what is been proposed in [Tong and Koller, 2001] and this correcting term has also an easy geometrical interpretation (see figure 8.3).

Schohn and Cohn [2000] also proposed to choose the point which will reduce the margin the most, but they did not implement this idea because of its computational complexity.

Let us denote the quantity (8.11) as $T_{\mathbf{x},y}$. Of course the label y associated with an unlabeled point $\mathbf{x}$ is unknown and the easiest way to choose an unlabeled point is to assume a worst-case scenario: choose the one which maximizes

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \ \min(T_{\mathbf{x},1}, T_{\mathbf{x},-1}).$$

Note that the margin is proportional to the radius of the largest sphere inscribable in version space [Herbrich et al., 2001; Tong and Koller, 2001] and as a consequence the volume of the version space is bounded by $T_{\mathbf{x},y}^{-d/2}$, where d is the dimension of the feature space. By computing posterior probabilities, one could choose the point which minimizes

$$\sum_y \tilde{P}(y|\mathbf{x}) T_{\mathbf{x},y}^{-d/2}.$$

8.6 Using posterior probabilities

We will now derive a method inspired of equation (8.2), which tries to estimate directly the posterior probabilities and to average them over unlabeled points. This idea is also described in [Schohn and Cohn, 2000], but the authors argued it is computationally infeasible.

This approach is the optimal approach described in section 8.2, where the estimation of the posterior probability is used twice: one directly to compute $P(y|\mathbf{x})$ on the unlabeled point we are interested in and also indirectly to estimate the generalization error when this point $\mathbf{x}$ is added to the training set through equation (8.2).

Thus the expected generalization error after adding the point $\mathbf{x}$ is

$$\bar{T}_{\mathbf{x}} = \sum_y \tilde{P}(Y = y|X = \mathbf{x}, D) \frac{1}{n_u} \sum_{i=1}^{n_u} \left(1 - \max_{y' \in \{-1,1\}} \tilde{P}(Y = y'|X = \mathbf{x}'_i, D \cup (\mathbf{x}, y))\right), \tag{8.12}$$

where D is the training set and $P(Y = y'|X = \mathbf{x}'_i, D \cup (\mathbf{x}, y))$ is the posterior distribution at $\mathbf{x}'_i$ after adding the point $(\mathbf{x}, y)$ in the training set and retrained the SVM.

The estimation of $\tilde{P}(Y = y|X = \mathbf{x}, D)$ is not a problem and has been discussed in section 8.4. The estimation of $\tilde{P}(Y = y'|X = \mathbf{x}'_i, D \cup (\mathbf{x}, y))$ is more difficult since it implies to retrain an SVM.

To compute efficiently this last quantity, we propose the following steps

1. Estimate the SVM parameters $\tilde{\boldsymbol{\alpha}}, \tilde{b}$ after $(\mathbf{x}, y)$ has been added to the training set using equations (8.5) and (8.7) and get the new decision function $\tilde{f}$.
2. For $\mathbf{x}_i$ in the training set, compute $\tilde{f}(\mathbf{x}_i)$
3. Estimate the output of those points after the leave-one-out with $\tilde{f}(\mathbf{x}_i) - \tilde{\alpha}_i \tilde{S}_i^2$, where $\tilde{S_i^2} = \tilde{\mathbf{H}}_{ii}^{-1}$ and $\tilde{\mathbf{H}}^{-1}$ is given by equation (8.7).
4. Use Platt's algorithm to fit the sigmoid to those outputs

If m is the number of unlabeled examples, the complexity is in $O(n^2(n+m))$ instead of $O(n^3 m)$.

8.7 Experiments

The different ideas presented in this section for active learning have been tested on the same toy and real world databases than the ones used in [Schohn and Cohn, 2000; Campbell et al., 2000; Tong and Koller, 2001]. Unfortunately, the results have been disappointing: none of the methods we tested could perform significantly better than the simple and intuitive margin heuristic described in [Schohn and Cohn, 2000; Campbell et al., 2000; Tong and Koller, 2001], which consists in labeling the point which is the nearest from the current decision boundary.

We will just give two examples. The first one is a text classification experiment taken from [Tong and Koller, 2001] on the newsgroup `comp.sys.ibm.pc.hardware`. We followed the same experimental protocol: one random document of each class is selected to be in the initial training set. The queries are selected from a random pool of 500 unlabeled samples. The evolution of test error, averaged over 100 trials, is plotted in figure 8.4 as a function of the training set size. The methods we compared are

Random As in passive learning, a random point is selected from the pool of unlabeled data.

Margin The heuristic consisting in choosing the nearest point from the hyperplane (smallest margin).

Ratio The method of [Tong and Koller, 2001] described in section 8.5.1: with the score $\min(m^+/m^-, m^-/m^+)$. This method tries to halve the version space at each query but it requires at each iteration a lot of SVM trainings.

Fast-Ratio The same method as above, but using the incremental learning techniques of section 8.5.2. At each iteration, only one SVM training is needed and the order of complexity is thus similar to the simple margin method.

In [Tong and Koller, 2001], the authors reported an improvement using the ratio method over the margin one on this database, but we were not able reproduce this result: this method as well as its fast approximation fast-ratio give almost identical results as the baseline one, margin.

An other example is a toy problem similar to the majority one used in [Campbell et al., 2000]. Points are in $\mathbb{R}^5$ and the true decision function is $y = \operatorname{sgn}(x_1 + \cdots + x_5)$ and for $1 \leq i \leq 5$, $x_i \sim \tanh(U_{[-1,1]})$. In this way, points of different classes tend to be separated with a larger margin.

Initially, 3 points of each class are in the training set. Then 10 points to be labeled are selected one by one from the an unlabeled pool of 300 points. The evolution of the test error averaged over 100 trials is reported in figure 8.5. An RBF kernel with width $\sigma^2 = 1.5$ was used. In addition to the random and margin methods, we plotted the GEM one (Generalization Estimate Minimization) which is the optimal method described in section 8.2 with the generalization error estimated by (8.2) and the posterior probabilities computed by Platt's method (section 8.4).

When done naively, this method requires a lot of SVM trainings per iteration, but as explained in section 8.6, it is possible to have a fast version of it.

In this experiment, GEM and its fast approximation behave similarly, but they are much worse than the margin method.

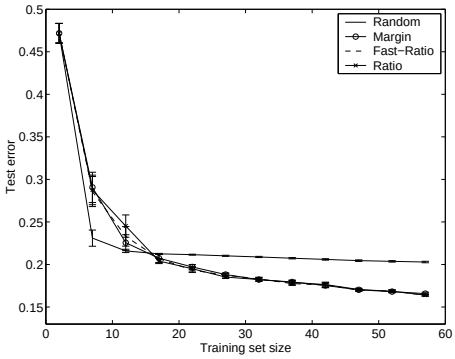


Figure 8.4: Evolution of the test error for different methods on the `comp.sys.ibm.pc.hardware` newsgroup.

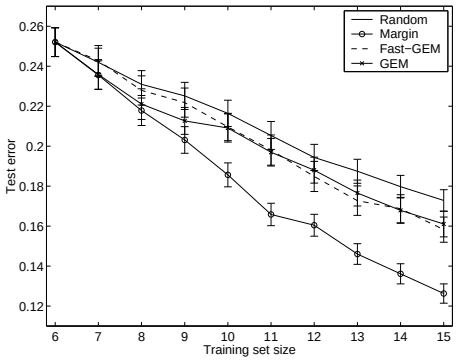


Figure 8.5: Evolution of the test error on a toy problem. The fast GEM method is described in section 8.6.

Discussion

In summary, we were unable to design an active learning strategy which beats the margin heuristic proposed simultaneously in [Schohn and Cohn, 2000; Tong and Koller, 2001; Campbell et al., 2000]. We have tried to select the point which would minimize the resulting generalization error when added to the training set, but it seems that for this purpose, very accurate estimates of the posterior probabilities are needed.

This study pointed out that the main difficulty was not of a computational nature. Indeed the two algorithms proposed in section 2.1 of [Schohn and Cohn, 2000] are very similar to GEM and ratio described in this chapter, but the authors have argued that they are computationally intractable and instead proposed the margin heuristic. However, based on the techniques of fast calculation of the leave-one-out error, we have been able to derive fast approximations of these algorithms and we showed experimentally that these approximations did not worsen the generalization error. The main problem is that even the original algorithms did not improve over the margin method.

Chapter 9

Model selection in regression using unlabeled data

Classical results of model selection for linear regression are based on an asymptotic analysis. We present a new penalization method for performing model selection for regression that is appropriate even for small samples [Chapelle et al., 2002a]. Our penalization is based on an accurate estimator of the ratio of the expected training error and the expected generalization error, in terms of the expected eigenvalues of the input covariance matrix. This ratio can be estimated using unlabeled data.

9.1 Introduction

Consider the problem of estimating a regression function in the set of functions

$$f(\mathbf{x}, \boldsymbol{\alpha}) = \sum_{k=1}^{\infty} \alpha_k \varphi_k(\mathbf{x}) \tag{9.1}$$

where $\{\varphi_k\}$ form a basis of $L_2(\mathbb{R}^p)$, e.g. a Fourier or wavelet basis.

Given a collection of data $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)$, where $y_i = f(\mathbf{x}_i, \boldsymbol{\alpha}_0) + \xi_i$ and $\mathbf{x}_i$, ξ_i are independently generated by unknown distributions $P(x)$ and $P(\xi)$, one wants to find the function $f(\mathbf{x}, \alpha_*)$ that provides the smallest value of the expected loss

$$R(\alpha) = \int L(y, f(\mathbf{x}, \boldsymbol{\alpha})) dP(\mathbf{x}) dP(\xi) \tag{9.2}$$

where $L(y, f(\mathbf{x}, \boldsymbol{\alpha}))$ is a given loss function, usually the quadratic loss $L(y, f(\mathbf{x}, \boldsymbol{\alpha})) = (y - f(\mathbf{x}, \boldsymbol{\alpha}))^2$. To minimize the expected risk (9.2), one minimizes the empirical risk functional

$$R_{emp}(\alpha) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i, \alpha))$$

However since the set (9.1) has an infinite expansion, this idea does not work: for any finite number of (different) examples there are functions which have zero empirical risk and a large value of the expected loss.

To guarantee a small expected risk, one can minimize the empirical functional over only the first $d = d(n)$ functions $\varphi_k(\mathbf{x})$. This is reasonable if the φ_k are ordered in such way that puts the "smoother" components first, introducing a preference for smooth functions. The problem of choosing an appropriate value $d = d(n)$ is called *model selection*.

For the case of quadratic loss and a large number of observations, several penalty-based methods were proposed in the mid-70's, and these are asymptotically optimal. All of these solutions, described in more detail below, minimize functionals of the form

$$R^*_{emp}(\hat{f}_d) = R_{emp}(\hat{f}_d) T(d, n) \tag{9.3}$$

where n is the sample size, $R_{emp}(\hat{f}_d)$ is the minimum of the empirical risk when training with a model of size d (achieved by the function $\hat{f}_d$), and $T(d,n)$ is a correction factor for performing model selection.

In particular Akaike [1970] defined in the context of autoregressive models the "Future Prediction Error" (FPE) correction factor

$$T(d,n) = (1 + d/n)(1 - d/n)^{-1}, \tag{9.4}$$

For small ratios d/n this multiplicative factor has a linear approximation $(1 + 2d/n)$. Generalized Cross-Validation [Wahba et al., 1979] and Shibata's model selector [Shibata, 1981] have the same linear approximation. Some other criteria which provide a different asymptotic behavior have been proposed including RIC [Foster and George, 1994], BIC [Schwartz, 1978] as well as criteria derived from the Minimum Description Length (MDL) principle [Rissanen, 1986; Barron et al., 1998].

During the same years, a general theory of minimizing the empirical risk (for any set of functions, any loss functions, and any number of samples) has been constructed [Vapnik, 1982]. In the framework of this theory, the method of *Structural Risk Minimization* for model selection was proposed. In the case studied here, this yields the following multiplicative factor [Cherkassky et al., 1997], derived from *Uniform Convergence Bounds* (UCB):

$$T(d,n) = \left(1 - c\sqrt{\frac{d(\log n/d + 1) - \log \eta}{n}}\right)_+^{-1} \tag{9.5}$$

where $u_+ = \max(0, u)$ and c, η are some constants. In spite of the fact that in the asymptotic case, this factor is less accurate than classical ones, simulation experiments showed that this correction factor outperforms other classical ones [Cherkassky et al., 1997].

This chapter is the development of an idea described in [Vapnik, 1998]. We first show that the expectation of the loss of the function minimizing the empirical risk depends both on the ratio d/n and the eigenvalues of a covariance matrix. It appears that by taking into account those eigenvalues we obtain a correction factor $T(d,n)$ which for small d/n coincides with Akaike's factor, but which is significantly different for larger d/n. These eigenvalues can be computed from a unlabeled sample.

This analysis aims at characterizing the relation between empirical risk and bias (residual of the approximation and noise) on one hand, and between bias and and generalization error on the other hand. For this purpose we made an independence assumption which might not be satisfied in practice. However, in our experiments the obtained estimator has a very good accuracy which suggests that this assumption is reasonable.

In the last section of the chapter, we compare the estimation accuracy of our method with classical ones and show that one can use it to perform state-of-the-art model selection.

9.2 Risk of the Mean Square Error Estimator

We consider a linear model of dimension d,

$$\mathcal{F}_d = \left\{ \mathbf{x} \to \sum_{i=1}^{d} \alpha_i \varphi_i(\mathbf{x}) \right\} \tag{9.6}$$

with $\alpha_i \in \mathbb{R}$ and the family $\{\varphi_i(\mathbf{x})\}_{i \in \mathbb{N}}$ is orthonormal with respect to the probability measure $P(\mathbf{x})$, which means $E\ \varphi_p(\mathbf{x})\varphi_q(\mathbf{x}) = \delta_{pq}$. [9] We assume without any loss of generality that this family is also a basis of $L_2(\mathbb{R}^p)$ (if it is not, it is always possible to extend it). Let $\hat{f}_d$ be the function minimizing the empirical mean square error over the set of functions $\mathcal{F}_d$, i.e.

$$\hat{f}_d = \arg\min_{f \in \mathcal{F}_d} R_{emp}(f),$$

The following section gives an estimator of the risk of $\hat{f}_d$. This risk estimator will lead directly to the choice of the correcting term in the model selection problem.

[9] Note that the choice of such a family requires knowledge about $P(x)$. See remark 3 of section 9.2.2 for more details.

We suppose without loss of generality that the first function φ_1 is the constant function 1 and then by orthonormality we have for all $p > 1$,

$$E\varphi_p(x) = 0 \tag{9.7}$$

9.2.1 Derivation of the Risk Estimator

In the orthonormal basis $\{\varphi_i(\mathbf{x})\}_{i\in\mathbb{N}}$, the desired regression function can be written as

$$f(\mathbf{x}) = \sum_{i=1}^{\infty} \alpha_i \varphi_i(\mathbf{x})$$

and the regression function minimizing the empirical risk is

$$\hat{f}_d(\mathbf{x}) = \sum_{i=1}^{d} \hat{\alpha}_i \varphi_i(\mathbf{x})$$

Let the i.i.d. noise ξ have variance σ^2 and mean zero, then the risk of this function is

$$\begin{aligned} R(\hat{f}_d) &= \int (f(\mathbf{x}) + \xi - \hat{f}_d(\mathbf{x}))^2 dP(\mathbf{x}) dP(\xi) \\ &= \sigma^2 + \int (f(\mathbf{x}) - \hat{f}_d(\mathbf{x}))^2 dP(\mathbf{x}) \\ &= \sigma^2 + \sum_{i=1}^{d} (\alpha_i - \hat{\alpha}_i)^2 + \sum_{i=d+1}^{\infty} \alpha_i^2 \end{aligned} \tag{9.8}$$

The last equality comes from the orthonormality of the family $\{\varphi_i\}_{i\in\mathbb{N}}$. The first term σ^2 corresponds to the risk of the true regression function, $R(f)$. The second term is the estimation error and the third term is the approximation error that we call r_d,

$$r_d = \sum_{i=d+1}^{\infty} \alpha_i^2 \tag{9.9}$$

To analyze equation (9.8), let us introduce the vector $\beta_i = \hat{\alpha}_i - \alpha_i$ of estimation errors and express the empirical risk in function of β,

$$\begin{aligned} R_{emp} &= \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \sum_{p=1}^{d} (\alpha_p + \beta_p) \varphi_p(\mathbf{x}_i) \right)^2 \\ &= \frac{1}{n} \sum_{i=1}^{n} \tilde{y_i}^2 - \frac{2}{n} \sum_{p=1}^{d} \beta_p \sum_{i=1}^{n} \tilde{y_i} \varphi_p(\mathbf{x}_i) \\ &\quad + \sum_{p,q=1}^{d} \beta_p \beta_q \frac{1}{n} \sum_{i=1}^{n} \varphi_p(\mathbf{x}_i) \varphi_q(\mathbf{x}_i), \end{aligned} \tag{9.10}$$

where

$$\tilde{y_i} = \xi_i + \sum_{p=d+1}^{\infty} \alpha_p \varphi_p(\mathbf{x}_i).$$

If we introduce the $n \times d$ matrix Φ, with $\Phi_{i,p} = \varphi_p(\mathbf{x}_i)$, then the empirical risk is minimized for

$$\beta = (\Phi^T \Phi)^{-1} \Phi^T \tilde{Y}, \tag{9.11}$$

where $\tilde{Y} = (\tilde{y_1}, \ldots, \tilde{y_n})^T$ and the minimum value of the empirical risk is

$$R_{emp}(\hat{f}_d) = \frac{1}{n} \tilde{Y}^T (I - \Phi(\Phi^T \Phi)^{-1} \Phi^T) \tilde{Y}. \tag{9.12}$$

The SVD decomposition of the matrix Φ writes $\Phi = USV^T$, where U and V are orthogonal matrices of size $n \times n$ and $d \times d$ respectively. S is a $n \times d$ diagonal matrix. Then

$$\Phi(\Phi^T\Phi)^{-1}\Phi^T = UI_dU^T,$$

with $S(S^TS)^{-1}S^T = I_d$ being a diagonal $n \times n$ matrix with its first d diagonal elements equal to 1 and the others zero. Thus equation (9.12) writes

$$\begin{aligned} R_{emp}(\hat{f}_d) &= \frac{1}{n}\tilde{Y}^TU(I_n - I_d)U^T\tilde{Y} \\ &= \frac{1}{n}\sum_{p=d+1}^{n}(\sum_{i=1}^{n}\tilde{y}_iU_{ip})^2 \end{aligned} \tag{9.13}$$

Let us now make the assumption that $\tilde{Y}$ and Φ are statistically independent. This assumption will be discussed at the end of the section. Then $\tilde{Y}$ and U are independent and $E\tilde{y}_iU_{ip} = E\tilde{y}_iEU_{ip} = 0$ from (9.7). From equation (9.13), we conclude

$$\begin{aligned} ER_{emp}(\hat{f}_d) &= \frac{1}{n}\sum_{p=d+1}^{n}\sum_{i=1}^{n}E\tilde{y}_i{}^2EU_{ip}^2 \\ &= \left(1 - \frac{d}{n}\right)(r_d + \sigma^2) \end{aligned} \tag{9.14}$$

The second equality is derived using the independence of ξ_i and $\mathbf{x}_i$, the orthonormality of the basis (yielding $E\tilde{y}_i{}^2 = (r_d + \sigma^2)$), and orthogonality of the matrix U (yielding $\sum_{i=1}^{n} EU_{ip}^2 = 1$).

In equation (9.8) we have to estimate $\sum_{p=1}^{d}(\alpha_p - \hat{\alpha_p})^2 = \sum_{p=1}^{d}(\beta_p)^2$. To do this, let us write

$$||\beta||^2 = \tilde{Y}^T\Phi(\Phi^T\Phi)^{-2}\Phi^T\tilde{Y}.$$

and denote by $(\lambda_1, \ldots, \lambda_d)$ the eigenvalues of the covariance matrix $C = \frac{1}{n}\Phi^T\Phi$,

$$C_{pq} = \frac{1}{n}\sum_{i=1}^{n}\varphi_p(x_i)\varphi_q(x_i). \tag{9.15}$$

Then one can show using the same technique as above that

$$E\sum_{p=1}^{d}\beta_p^2 = \frac{\sum_{i=1}^{d}E(1/\lambda_i)}{n}(r_d + \sigma^2)$$

Finally combining this last equality with equation (9.8) and (9.14), we obtain

$$\boxed{ER(\hat{f}_d) = ER_{emp}(\hat{f}_d)\left(1 - \frac{d}{n}\right)^{-1}\left(1 + \frac{E\sum_{i=1}^{d}(1/\lambda_i)}{n}\right)} \tag{9.16}$$

9.2.2 Remarks

1. We have made the assumption that $\tilde{Y}$ and Φ are independent. Actually, the matrix Φ depends only on the first d functions in the basis and $\tilde{Y}$ depends only on the functions beyond d and on the noise. Thus Φ and $\tilde{Y}$ are orthogonal but might not be statistically independent. However in practice this assumption seems to be reasonable (see figure 9.1). Also when the residual is small compared to the noise, $\tilde{y}_i \approx \xi_i$, and the independence of ξ_i and Φ motivates this assumption. Note that the assumption that there is no residual was also made in the derivation of the Akaike Information Criterion [Akaike, 1973]. Finally, the assumption would also be valid if $\varphi_i(x)$ is independent of $\varphi_j(x)$ (e.g. representing independent components of the vector x).

2. We computed the ratio of the expected generalization error and the expected empirical error. However, in practice, one would like to estimate the actual generalization error in function of the actual empirical error.

   To do this, in the previous derivation, one should replace equalities of the type

   $$E\frac{1}{k}\sum_{i=1}^{k}\tilde{y_i}^2 = r_d + \sigma^2$$

   by statements of the following type: With high probability,

   $$\left|\frac{1}{k}\sum_{i=1}^{k}\tilde{y_i}^2 - (r_d + \sigma^2)\right| \leq \frac{c}{\sqrt{k}}$$

   This kind of statement can be done if we have assumptions on the probability distribution of $\tilde{y_i}$ and would lead to risk bounds for the model selection strategy, as shown in [Bartlett et al., 2000].

3. This derivation is based on the assumption that the set of basis functions is orthonormal with respect to the probability measure $P(\mathbf{x})$. However in the learning problem this probability distribution is usually unknown and therefore it is impossible to get an explicit orthonormal basis. Nevertheless, for any given independent set of basis functions $\{\Psi_i(\mathbf{x})\}$ and any probability distribution, using Gram-Schmidt orthonormalization, one can theoretically get a unique orthonormal family $\{\Phi_i(\mathbf{x})\}$ that describes the same set of functions $\mathcal{F}_d$.

   From the previous argument, one can still use (9.16) for a non orthonormal family, keeping in mind however that the eigenvalues appearing in this estimator are the ones corresponding to the covariance matrix constructed from the Gram-Schmidt orthonormalized basis.

   In practice this orthogonalization can be made using unlabeled data (more details are provided in the next section).

9.3 Application to Model Selection

As the goal in model selection is to choose the model with the smallest expected risk, the previous analysis (see eq (9.16)) suggests to take the correcting term $T(d, n)$ as

$$T(d,n) = \left(1 - \frac{d}{n}\right)^{-1}\left(1 + \frac{E\sum_{i=1}^{d}(1/\lambda_i)}{n}\right) \tag{9.17}$$

where λ_i is the i-th eigenvalue of the covariance matrix (9.15).

Note that in the asymptotic case, since the covariance matrix is almost the identity matrix (from the orthonormality assumption), $E(1/\lambda_i) \approx 1$ and we obtain Akaike's term (9.4). However, in the non-asymptotic case the covariance matrix is not well-conditioned and it can happen that $E(1/\lambda_i) \gg 1$ (see figure 9.1).

Direct Eigenvalue Estimator method (DEE)

In the case when along with training data, "unlabeled" data are available, one can compute two covariance matrices: one from unlabeled data $\tilde{C}$ and another from the training data C_{emp}.

There is a unique matrix P [Horn and Johnson, 1985, corollary 7.6.5] such that

$$P^T\tilde{C}P = I \quad \text{and} \quad P^T C_{emp} P = \Lambda,$$

where Λ is a diagonal matrix with diagonal elements $\lambda_1, \ldots, \lambda_n$. To perform model selection, we used the correcting term (9.17) where we replace $E\sum_{i=1}^{d} 1/\lambda_i$ with its empirical value,

$$\begin{aligned}\sum_{i=1}^{d} 1/\lambda_i &= \text{trace}\left(P^{-1}C_{emp}^{-1}(P^T)^{-1}P^T\tilde{C}P\right)\\ &= \text{trace}\,(C_{emp}^{-1}\tilde{C}).\end{aligned}$$

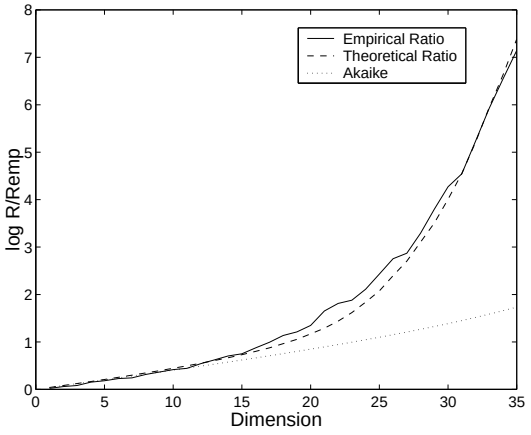


Figure 9.1: Comparison of the ratio (in log scale) of the median of the generalization error and training error (over 1000 trials) with the penalty term (9.17) and with Akaike's penalty. The latter is only accurate when d/n is small. The number of training examples n is 50, the target function is the step function, the noise level is 0.05, the training points points are uniformly generated in $[-\pi, \pi]$ and the empirical risk minimization has been carried out in the Fourier basis.

This enables us to deal with a non orthonormal family. As before the quantity trace(C_{emp}^{-1}) is an indicator of the discrepancy between the empirical covariance matrix C_{emp} and its "expected" value $\tilde{C}$

Smallest Eigenvalue Bound (SEB)

To estimate $E\sum 1/\lambda_i$ appearing in eq (9.16), one can use a lower bound on the smallest eigenvalue of the covariance matrix.

Lemma 9.1 *With probability at least* $1-\eta$

$$\lambda_{min} > 1 - \sqrt{V_d \Lambda_d(n)}, \tag{9.18}$$

where

$$V_d = \sup_{\mathbf{x}} \sup_{||\alpha||_2=1} \left(\sum_{i=1}^{d} \alpha_i \varphi_i(\mathbf{x}) \right)^2 \quad \text{and} \quad \Lambda_d(n) = \frac{d\left(\log \frac{2n}{d} + 1\right) - \log(\eta/4)}{n} \tag{9.19}$$

The proof is in appendix.

In practice, we take $\eta = 0.1$ and $V_d = 1$ [10] and we get the following bound,

$$ER(\hat{f}_d) \leq ER_{emp}(\hat{f}_d) \left(1 - \frac{d}{n}\right)^{-1} \left(1 + \frac{d}{nk}\right) \tag{9.20}$$

where

$$k = \left(1 - \sqrt{\frac{d\left(\log \frac{2n}{d} + 1\right) + 4}{n}}\right)_+ \tag{9.21}$$

Remark: expected risk minimization and model selection

In section 9.2, we derived an unbiased estimator of the risk of the function minimizing the mean square error on a linear model of dimension d. The model selection procedure we proposed is to choose the model minimizing this unbiased estimator.

[10] V_d might be much larger than 1 for some basis φ, but in our experiments $V_d = 1$ seems to be a good choice.

However a more detailed analysis should be carried out. Indeed, if the variance of our estimator is large and the number of models tested is also large, then some overfitting problems might occur (see section 1.2.2. To avoid this, one needs to increase the penalty in order to capture the variance of the risk estimator and the number of models. A related explanation can also be found in remark 2, page 151.

We do not consider here the case where of lot of models are available, but just the standard case of nested regression (in which the number of models is less than the number of training points) and choosing the model which minimizes an unbiased estimator of the test error should give good results.

As explained before, the case of non-nested regression (choice of wavelet coefficients for example) needs some additional analysis and is left for future work.

9.4 Experimental Results

We performed toy experiments in order to compare model selection algorithms. The input distribution is the uniform distribution on $[-\pi, \pi]$ and the set of basis functions is the Fourier basis,

$$\varphi_1(x) = 1$$

$$\varphi_{2p} = \sqrt{2}\ \cos(px)$$

$$\varphi_{2p+1} = \sqrt{2}\ \sin(px)$$

We compared our model selection methods, SEB (Smallest Empirical Bound, cf page 152) and DEE (Direct Eigenvalue Estimator, cf page 151) to eight other methods. Six of them are penalty-based: FPE (Akaike, eq 9.4), Uniform Convergence Bound (UCB) (9.5), GCV [Wahba et al., 1979], RIC [Foster and George, 1994], BIC [Schwartz, 1978], Mallow's C_p (CPM) [Mallows, 1973]. For the UCB method, we took $c = 1$ and $\log \eta = -3$ in equation (9.5).

The two other model selection algorithms we considered are ADJ (a state-of-the-art heuristic method [Schuurmans, 1997]), and CV_5 (5-fold cross-validation).

Note that both ADJ and DEE need some information about the distribution of input data $P(x)$ which can be provided by unlabeled data. In the experiments we drew 1000 unlabeled points.

We first compared the accuracy of some of these methods in the prediction of the generalization error. For this purpose, we considered the regression function

$$f(x) = \frac{1}{10}\left(x + \frac{3}{2}\right)^2,$$

a gaussian noise with standard deviation $\sigma = 0.05$ and a training set of 40 examples. For each $d \leq 23$, we computed the empirical risk minimizer $\hat{f}_d$ and tried to predict the generalization error $R(\hat{f}_d)$. The results are shown in figure 9.2 and are averaged over 1000 trials.

Both DEE and ADJ predict accurately the test error, ADJ being a little bit over pessimistic. When the number of dimension becomes large, FPE underestimates the generalization error while CV5 overestimates (this is explained by the fact that during cross-validation a smaller training set is used).

For the model selection itself we are interested in the generalization error of the function chosen by the model selection procedure. Indeed, as explained at the end of section 9.3, an unbiased estimator of the generalization error with a large variance might give a poor criterion for model selection.

Different experiments have been carried out by changing the variance of the gaussian noise, the number of training points or the target function. For each model selection procedure, if the model $\hat{d}$ is chosen, we compute the log of the approximation ratio,

$$\log \frac{R(\hat{f}_{\hat{d}})}{\min_d R(\hat{f}_d)}. \tag{9.22}$$

The results are shown in boxplot style in figures 9.3 and 9.4, each one corresponding to a different target function: sinc $(\sin(4x)/4x)$ and step $(\mathrm{I}_{x>0})$ functions. All the experiments have been repeated 1000 times.

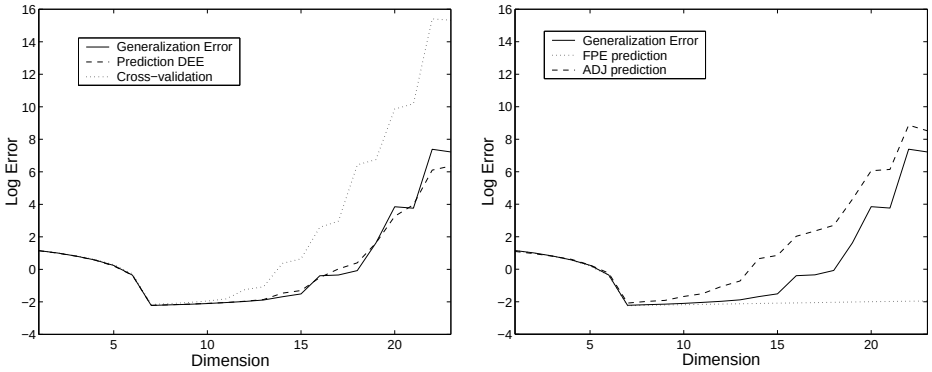


Figure 9.2: Prediction of the generalization error for the following methods: DEE, CV5 (left), FPE, ADJ (right)

	ADJ	SEB	CV5	DEE	UCB	GCV	RIC	BIC	FPE	CPM
a	1.11	1.01	1.12	1.09	1.02	1.16	1.01	1.04	1.48	1.13
b	1.05	1.05	1.05	1.05	1.05	1.05	1.04	1.05	1.10	1.05
c	1.42	1.91	1.75	1.48	1.91	1.94	1.70	1.70	2.32	1.97
d	1.11	1.14	1.18	1.13	1.15	1.21	1.22	1.15	1.74	1.16
e	1.17	1.18	1.21	1.15	1.90	1.21	1.25	1.15	1.50	1.17
f	1.08	1.00	1.02	1.01	1.07	1.01	1.15	1.00	1.13	1.01
g	1.37	1.45	1.49	1.55	1.64	2.60	4.00	4.60	4.21	5.77
h	1.28	1.33	1.36	1.36	1.45	2.34	2.03	5.45	12.33	13.14
i	1.44	1.50	1.52	1.63	1.67	3.19	6.94	7.06	4.52	7.14
j	1.41	1.49	1.49	1.72	2.12	18.11	36.01	36.05	30.22	42.93
k	1.18	1.20	1.21	1.24	1.15	1.37	1.22	1.31	1.98	1.61
l	1.10	1.12	1.11	1.11	1.23	1.16	1.18	1.11	1.84	1.20
	1.23	**1.28**	**1.29**	**1.29**	**1.45**	**3.03**	**4.90**	**5.22**	**5.36**	**6.61**

Table 9.1: Median of the ratio of the test error to the best model for the 12 experiments reported in figures 9.3 and 9.4. The last row is an average over the 12 experiments.

The plots for the sinc function (figure 9.3) show that the model selection procedures have a similar performance when the function is easy to estimate (the Fourier coefficients of this function decrease very rapidly). Only FPE is far from the optimal solution for 50 training points.

The second example is the step function (figure 9.4), which is difficult to approximate in the Fourier basis. In this case, traditional penalty based method (RIC, BIC, CPM, GCV, FPE) fail whereas DEE, SEB, UCB, ADJ and CV5 are able to select a good model.

For each experiment, tables 9.1 and 9.2 indicate the median and the mean (over 1000 trials) of the approximation ratio (9.22).

Judging from these experiments, both proposed methods DEE and SEB perform as well as the state-of-the-art methods, such as the ADJ heuristic or cross-validation and UCB, while classical penalty-based methods fail. It is worth noting that the ADJ heuristic seems to be the best model selection procedure among all the ones we tested.

The comparison between table 9.1 (median of the approximation ratio) and table 9.2 (mean of the approximation) gives a better insight of the behavior of some model selection algorithms. For example, UCB has a median of 1.45, but a mean of 15.4. This is due to the fact that sometimes it selects a very large model (i.e. it overfits) incurring a catastrophic generalization error. The same explanation applies obviously to other penalty-based methods (which have terrible approximation ratios in mean) and to a certain extent to CV5 (see row c of table 9.2) and SEB (see row j). Intuitively, cross-validation gives an almost unbiased estimator of the generalization error, but because of its variance, it might select

	ADJ	DEE	SEB	CV5	UCB	RIC	GCV	FPE	BIC	CPM
a	2.63	2.48	1.77	2.85	2.27	1.64	3.6e3	4.3e3	7	327
b	1.1	1.16	1.09	1.1	1.09	1.05	1.31	26	1.09	1.2
c	3.51	3.12	2.71	7.45	2.67	19.5	158	1.3e4	289	1.3e4
d	1.26	1.29	1.27	1.31	1.32	1.35	2.4e3	6.5e3	1.33	11.3
e	1.42	1.44	1.47	1.58	1.93	1.59	139	3.5e4	1.51	1.2e4
f	1.12	1.07	1.06	1.08	1.08	1.14	7.15	1e4	1.05	1.06
g	1.69	2.44	2.44	2.53	1.88	3.1e4	3.4e4	3.4e4	3.5e4	3.5e4
h	1.44	1.69	1.43	1.48	12.2	637	464	3.4e3	5.9e8	5.9e8
i	2.39	3.67	2.21	2.36	2.18	4e3	3.8e3	4e3	4e3	4e3
j	1.8	2.69	8.29	2.28	156	1.8e4	5.4e4	1e5	1.8e4	5.6e4
k	1.32	1.49	1.38	1.94	1.22	15.3	615	9.7e5	159	700
l	1.14	1.19	1.15	1.16	1.24	1.21	17.9	1.5e3	1.48	63.5
	1.73	**1.98**	**2.19**	**2.26**	**15.4**	**4.5e3**	**8.3e3**	**9.9e4**	**4.9e7**	**4.9e7**

Table 9.2: Mean of the ratio of the test error to the best model for the 12 experiments reported in figures 9.3 and 9.4. The last row is an average over the 12 experiments.

sometimes a model which is far from the optimal one. This is also true for SEB and DEE, even though we expect these methods to have a smaller variance. A discussion on this topic can be found at the end of section 9.3.

9.5 Conclusion

In this chapter we showed that to select models using small sample size the formulas obtained for asymptotic classical models are insufficient. In our analysis, we pointed out that the discrepancy between the empirical covariance matrix and its expectation is critical for small sample size regression. Taking this discrepancy into account, we obtain a model selection algorithm which behaves similarly to the state-of-the-art.

Further research includes improvement of the SEB method thanks to a deeper analysis of the distribution of the eigenvalues of a covariance matrix. The DEE method is very attractive since it provides an unbiased estimator of the generalization error of a given model. It requires unlabeled data, but if such data is not available, we believe this method will still be efficient by generating unlabeled data from a Parzen window estimator of the input density. New experiments will be carried out to assess this supposition.

From a theoretical point of view, we will focus on the remark at the end of section 9.3 and try to extend this method for non-nested regression. A typical application of this in machine learning would be to determine the number of centers in a RBF network.

Appendix

Proof of lemma 1 Consider the quantity

$$Q(\mathbf{x}, \alpha) = \left(\sum_{p=1}^{d} \alpha_p \varphi_p(\mathbf{x}) \right)^2$$

For all α such that $||\alpha|| = 1$, we have $EQ(\mathbf{x}, \alpha) = 1$. On the other hand,

$$\frac{1}{n} \sum_{i=1}^{n} Q(\mathbf{x}_i, \alpha) = \alpha^T C \alpha$$

where $C = \Phi^T\Phi/n$ is the covariance matrix and then

$$\min_{||\alpha||=1} \frac{1}{n}\sum_{i=1}^{n} Q(\mathbf{x}_i, \alpha) = \lambda_{min},$$

where λ_{min} is the smallest eigenvalue of C. In [Vapnik, 1982], it is shown that for any family of functions Q satisfying $0 \leq Q(\mathbf{x}, \alpha) \leq B$ and of VC dimension d, the following inequality holds

$$P\left(\sup_{\alpha} EQ(\mathbf{x}, \alpha) - \frac{1}{n}\sum_{i=1}^{n} Q(\mathbf{x}_i, \alpha) > \varepsilon\right) \leq \exp\left(\frac{d}{n}(\log(2n/d) + 1) - \frac{\varepsilon^2}{B^2}\right) n$$

Using this last inequality, we get that with probability $1 - \eta$,

$$1 - \lambda_{min} < \sqrt{V_d \Lambda_d(n)} \quad \square$$

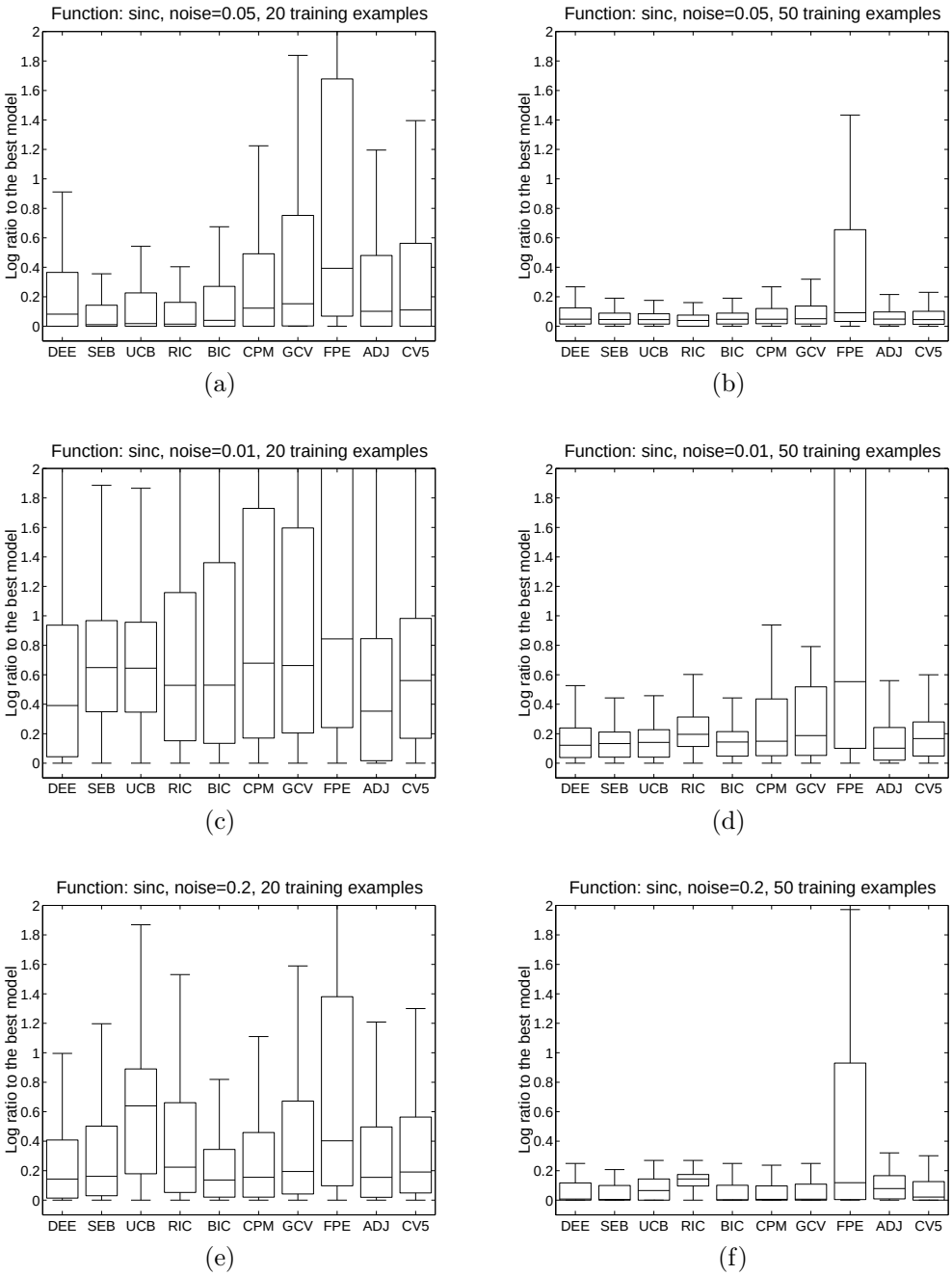


Figure 9.3: Approximation ratios for the sinc function. Numerical results can be found in tables 9.1 and 9.2, each letter corresponding to the same experiment.

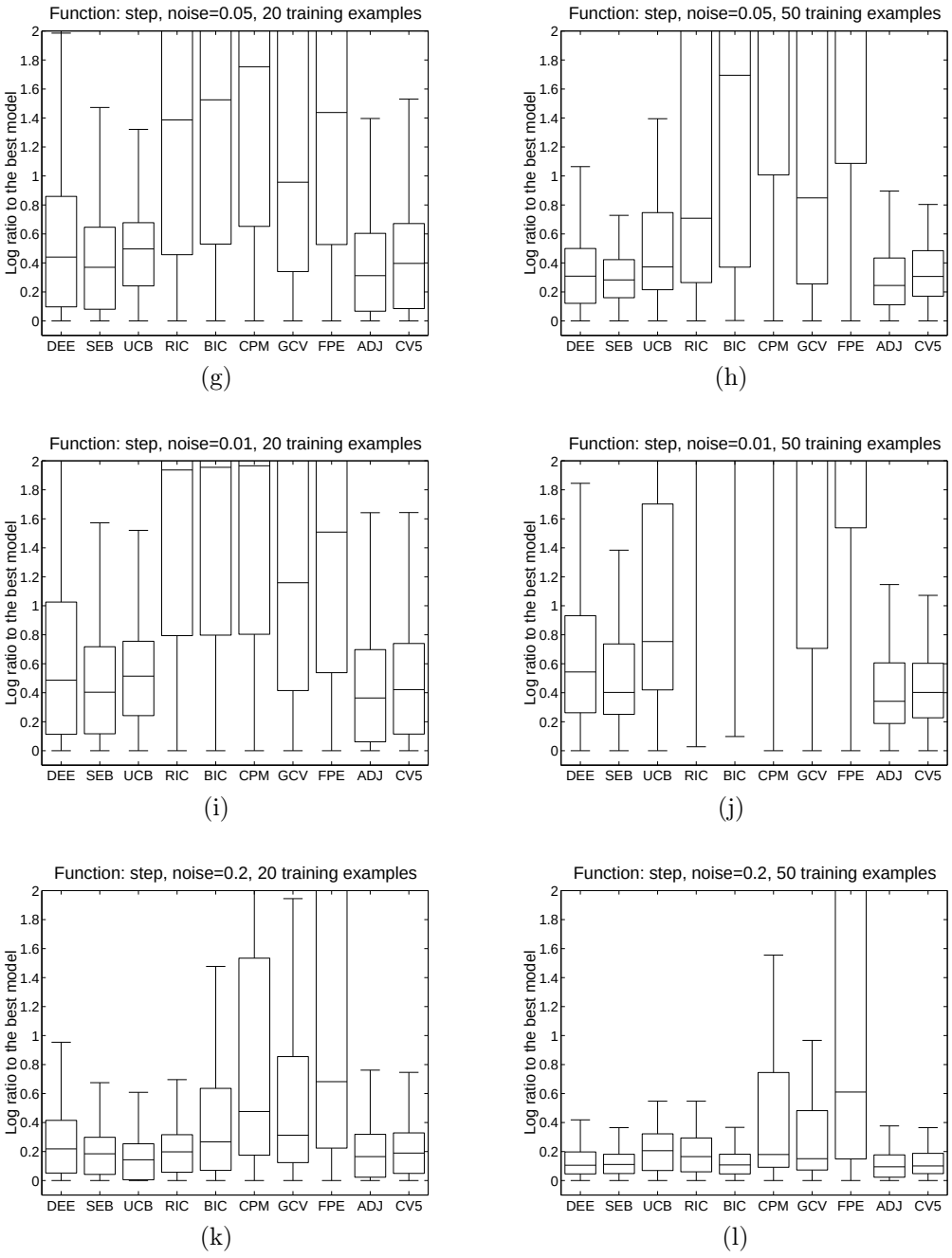


Figure 9.4: Approximation ratios for the step function. Numerical results can be found in tables 9.1 and 9.2, each letter corresponding to the same experiment.

Bibliography

K. Aas and L. Eikvil. Text categorisation: A survey. Technical Report 941, Norwegian Computing Center, June, 1999.

M. Aizerman, E. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

H. Akaike. Statistical predictor identification. *Ann. Inst. Stat. Math.*, 22:202–217, 1970.

H. Akaike. Information theory and an extension of the maximum likelihood principle. In B.N. Petrov and F. Csaki, editors, *2nd International Symposium on Information Theory*, volume 22, pages 267–281, Budapest, 1973.

U. Alon, N. Barkai, D. Notterman, K. Gish, S. Ybarra, D. Mack, and A. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon cancer tissues probed by oligonucleotide arrays. *Cell Biology*, 96:6745–6750, 1999.

M.-R. Amini. *Apprentissage Automatique et Recherche de l'Information : application à l'Extraction d'Information de Surface et au Résumé de texte*. PhD thesis, Université de Paris 6, 2001.

M.-R. Amini and P. Gallinari. Learning for text summarization using labeled and unlabeled sentences. In *11th International Conference of Artificial Neural Networks*, 2001.

G. An. The effects of adding noise during backpropagation training on a generalization performance. *Neural Computation*, 8:643–674, 1996.

F. Bach and M. I. Jordan. Kernel independent component analysis. Technical Report UCB/CSD-01-1166, University of California, Berkeley, november 2001.

A. Barron, J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44:2743–2760, 1998.

P. Bartlett, S. Boucheron, and G. Lugosi. Model selection and error estimation. In *Proceedings of the 13th Annual Conference on Computational Learning Theory, COLT*, pages 286–297, San Francisco, 2000. Morgan Kaufmann.

P. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. In *14th Annual Conference on Computational Learning Theory*, volume 2111, pages 224–240. Springer, Berlin, 2001.

P. Bartlett and J. Shawe-Taylor. Generalization performance of support vector machines and other pattern classifiers. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*. MIT Press, Cambridge, MA, 1999.

G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Computation*, 12(10):2385–2404, 2000.

G. Baudat and F. Anouar. Kernel-based methods and function approximation. In *International Joint Conference on Neural Networks*, pages 1244 – 1249, Washington DC, 2001.

M. Bazaraa and C.M. Shetty. *Nonlinear programming*. John Wiley, New York, 1979.

Y. Bengio. Gradient-based optimization of hyper-parameters. *Neural Computation*, 12(8), 2000.

K. Bennett and A. Demiriz. Semi-supervised support vector machines. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1998.

C. Bishop, M. Svensen, and C. Williams. GTM: The generative topographic mapping. *Neural Computation*, 10(1):215–234, 1998.

C. M. Bishop. Training with noise is equivalent to Tikhonov regularization. *Neural Computation*, 7(1): 108–116, 1995.

A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory*. Morgan Kaufmann Publishers, 1998.

A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, 1989.

J.F. Bonnans and A. Shapiro. *Perturbation Analysis of Optimization Problems*. Springer-Verlag, 2000.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proc. 5th Annu. Workshop on Comput. Learning Theory*, pages 144–152. ACM Press, New York, NY, 1992.

L. Bottou and V. Vapnik. Local learning algorithms. *Neural Computation*, 4(6):888–900, 1992. appendix on confidence intervals.

O. Bousquet and A. Elissee. Stability and generalization. *Journal of Machine Learning Research*, 2: 499–526, 2002.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. Availabe at `http://www.stanford.edu/~boyd/cvxbook.html`.

P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In *Proc. 13th International Conference on Machine Learning*, pages 82–90, San Francisco, CA, 1998.

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

L. Breiman, W. Meisel, and E. Purcell. Variable kernel estimates of multivariate densities. *Technometrics*, 19:135–144, 1977.

M. Brown, W. Grundy, D. Lin, N. Cristianini, C. Sugnet, T. Furey, M. Jr, and D. Haussler. Knowledge-based analysis of microarray gene expression data by using suport vector machines. In *Proceedings of National Academy of Sciences*, volume 97, pages 262–267, 2000.

C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

C. Burges. Geometry and invariance in kernel based methods. In *Advances in Kernel Methods - Support Vector Learning*, pages 89–116. MIT Press, 1999.

C. Campbell, N. Cristianini, and A. Smola. Query learning with large margin classifiers. In *Proceedings of 17th International Conference on Machine Learning*, pages 111–118, San Francisco, CA, 2000. Morgan Kaufmann.

G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems*, pages 409–415, 2000.

O. Chapelle, P. Haffner, and V. N. Vapnik. Support vector machines for histogram-based image classification. *IEEE Transactions on Neural Networks*, 10(5):1055–1064, Sept 1999a.

O. Chapelle and B. Schölkopf. Incorporating invariances in nonlinear Support Vector Machines. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

O. Chapelle and V. Vapnik. Model selection for support vector machines. In *Advances in Neural Information Processing Systems*, 1999.

O. Chapelle, V. Vapnik, and Y. Bengio. Model selection for small sample regression. *Machine Learning*, 48(1-3):9–23, 2002a.

O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1-3):131–159, 2002b.

O. Chapelle, V. Vapnik, and J. Weston. Transductive inference for estimating values of functions. In *Advances in Neural Information Processing Systems*, volume 12, 1999b.

O. Chapelle, J. Weston, L. Bottou, and V. Vapnik. Vicinal risk minimization. In *Advances in Neural Information Processing Systems*, volume 13, 2000.

V. Cherkassky, F. Mulier, and V. Vapnik. Comparison of VC method with classical methods for model selection. In *Proceedings of the World Congress on Neural Networks*, pages 957–962, 1997.

D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 705–712. The MIT Press, 1995.

C. Cortes and V. Vapnik. Support vector network. *Machine learning*, 20:1–25, 1995.

T. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, 14:326–334, 1965.

F. G. Cozman and I. Cohen. Unlabeled data can degrade classification performance of generative classifiers. Technical Report HPL-2001-234, HP-Labs, 2001.

N. Cristianini, C. Campbell, and J. Shawe-Taylor. Dynamically adapting kernels in support vector machines. In *Advances in Neural Information Processing Systems*, 1999.

N. Cristianini, J. Shawe-Taylor, and J Kandola. Spectral kernel methods for clustering. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

B. Dacorogna. *Direct methods in the calculus of variations*. Springer-Verlag, 1989.

D. DeCoste and B. Schölkopf. Training invariant support vector machines. *Machine Learning*, 46(1-3): 161–190, 2002.

N. M. Dempster, A.P. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:185–197, 1977.

J. E. Dennis and R. E. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, NJ., 1983.

L. Devroye, L. Gyorfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag, New-York, 1996.

L. Devroye and G. Lugosi. *Combinatorial Methods in Density Estimation*. Springer-Verlag, New York, 2001.

C. Domeniconi, J. Peng, and D. Gunopulos. An adaptive metric machine for pattern classification. In *Advances in Neural Information Processing Systems*, volume 13, pages 458–464, 2000.

B. Eisenberg and R. L. Rivest. On the sample complexity of pac-learning using random and chosen examples. In *Proceedings of the Third Annual Conference on Computational Learning Theory*, pages 154–162. Morgan Kaufmann, 1990.

T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13(1):1–50, 2000.

V. Fedorov. *Theory of Optimal Experiments*. Academic Press, New York, 1972.

S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, 2001.

D. Foster and E. George. The risk inflation criterion for multiple regression. *Annals of Statistics*, 22(4): 1947–1975, 1994.

Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. In *Computational Learing Theory*, pages 209–217, 1998.

T.T. Friess and R.F. Harrison. Support vector neural networks: The kernel adatron with bias and soft margin. Technical Report 752, University of Sheffield, Department of Automatic Control and Systems Engineering, 1998.

P. Gaenssler and D. Rost. On uniform laws of large numbers for smoothed empirical measures. In E. Giné, D.M. Mason, and J.A. Wellner, editors, *High Dimensional Probability II*, volume 47 of *Progress in Probability*, pages 107–113. Springer, 2000.

S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1995.

Z. Ghahramani and M. J. Beal. Variational inference for bayesian mixtures of factor analysers. In *Advances in Neural Information Processing Systems*, volume 12, Cambridge, MA, 2000. MIT Press.

Z. Ghahramani and G. Hinton. The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, Departement of Computer Science, University of Toronto, 1997.

F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219–269, 1995.

S. Goldman and Y. Zhou. Enhancing supervised learning with unlabeled data. In *Proc. 17th International Conf. on Machine Learning*, pages 327–334. Morgan Kaufmann, San Francisco, CA, 2000.

T. Golub, D. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

C. Goutte and J. Larsen. Adaptive metric kernel regression. In *Neural Networks for Signal Processing VIII*, pages 184–193, Piscataway, New Jersey, 1998.

Y. Grandvalet, S. Canu, and S. Boucheron. Noise injection: Theoretical prospects. *Neural Computation*, 9(5):1093–1108, 1997.

I. Guyon, N. Matić, and V. N. Vapnik. Discovering informative patterns and data cleaning. In *AAAI workshop on Knowledge Discovery in Databases, KDD'94*, pages 145–156, Seattle, WA, July 1994. MIT Press.

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1/3):389, 2002.

D. Haussler. Convolution kernels on discrete structures. Technical Report UCS-CRL-99-10, UC Santa Cruz, 1999.

R. Herbrich, T. Graepel, and C. Campbell. Bayes point machines. *Journal of Machine Learning Research*, 1:245–279, 2001.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

A. E. Hoerl and R. W. Kennard. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, 12:55–67, 1970.

R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

T. Jaakkola, M. Diekhans, and D. Haussler. Using the fisher kernel method to detect remote protein homologies. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, 1999a.

T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing*, volume 11, pages 487–493. The MIT Press, 1998.

T. Jaakkola, M. Meila, and T. Jebara. Maximum entropy discrimination. Technical Report AITR-1668, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1999b.

T. S. Jaakkola and D. Haussler. Probabilistic kernel regression models. In *Proceedings of the 1999 Conference on AI and Statistics*, 1999.

T. Jebara and T. Jaakkola. Feature selection and dualities in maximum entropy discrimination. In *Uncertainity In Artificial Intellegence*, 2000.

T. Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398 in Lectures Notes in Artificial Intelligence, pages 137–142, Heidelberg, 1998. Springer Verlag.

T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the 16th International Conference on Machine Learning*, pages 200–209. Morgan Kaufmann, San Francisco, CA, 1999.

T. Joachims. Estimating the generalization performance of a SVM efficiently. In *Proceedings of the International Conference on Machine Learning*. Morgan Kaufman, 2000.

R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

J. Larsen, C. Svarer, L.N. Andersen, and L.K. Hansen. Adaptive regularization in neural network modeling. In G.B. Orr and K.R. Müller, editors, *Neural Networks : Trick of the Trade*. Springer, 1998.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. J. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324, 1998.

Y. LeCun, J. Denker, S. Solla, R. E. Howard, and L. D. Jackel. Optimal brain damage. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems II*, San Mateo, CA, 1990. Morgan Kauffman.

Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, P. Simard, and V. Vapnik. Comparison of learning algorithms for handwritten digit recognition. In *International Conference on Artificial Neural Networks*, pages 53–50, 1995.

M. Ledoux and M. Talagrand. *Probability in Banach Spaces: Isoperimetry and Processes*. Springer Verlag, 1991.

T. K. Leen. From data distributions to regularization in invariant learning. In *NIPS*, volume 7. The MIT Press, 1995.

P. Leray and P. Gallinari. Feature selection with neural networks. *Behaviormetrika*, 26(1), January 1999. Special issue on Analysis of Knowledge Representation in Neural Network Models.

D. G. Lowe. Similarity metric learning for a variable-kernel classifier. *Neural Computation*, 7(1):72–85, 1995.

A. Luntz and V. Brailovsky. On the estimation of characters obtained in statistical procedure of recognition. *Technicheskaya Kibernetica*, 1969. In Russian.

H. Lütkepohl. *Handbook of Matrices*. Wiley & Sons, 1996.

D. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4): 590–604, 1992.

C. L. Mallows. Some comments on C_p. *Technometrics*, 15(4):661–675, 1973.

N. Matic, I. Guyon, L. Bottou, J. S. Denker, and V. N. Vapnik. Computer aided cleaning of large databases for character recognition. In *Proceedings of the 11th International Conference on Pattern Recognition*, volume B, La Hague, 1992.

D. McAllester. Some PAC-bayesian theorems. In *COLT: Proceedings of the Workshop on Computational Learning Theory*. Morgan Kaufmann, 1998.

C. McDiarmid. On the method of bounded differences. In *Surveys in Combinatorics*, pages 148–188. Cambridge University Press, 1989.

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

S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, editors, *Neural Networks for Signal Processing IX*, pages 41–48. IEEE, 1999.

D. Miller and H. Uyar. A mixture of experts classier with learning based on both labelled and unlabelled data. In *Advances in Neural Information Processing Systems*, volume 9, pages 571–577, 1997.

S. Mukherjee, P. Tamayo, D. Slonim, A. Verri, T. Golub, J.P. Mesirov, and T. Poggio. Support vector machine classification of microarray data. AI Memo 1677, Massachusetts Institute of Technology, 1999.

E. A. Nadaraya. On estimating regression. *Theory of Probability and its Applications*, 9:141–142, 1964.

R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental, sparse and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, 1998.

A. Y. Ng. Preventing overfitting of cross-validation data. In *Proceedings of the 14th International Conference on Machine Learning*, pages 245–253. Morgan Kaufmann, 1997.

A. Y. Ng. On feature selection: learning with exponentially many irrelevant features as training examples. In *Proceedings of the 15th International Conference on Machine Learning*, pages 404–412, San Francisco, CA, 1998. Morgan Kaufmann.

A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

K. Nigam, A. K. McCallum, S. Thrun, and T. M. Mitchell. Learning to classify text from labeled and unlabeled documents. In *Proceedings of AAAI-98, 15th Conference of the American Association for Artificial Intelligence*, pages 792–799, Madison, US, 1998. AAAI Press, Menlo Park, US.

P. Niyogi, T. Poggio, and F. Girosi. Incorporating prior information in machine learning by creating virtual examples. *IEEE Proceedings on Intelligent Signal Processing*, 86(11):2196–2209, November 1998.

M. Opper and O. Winter. Gaussian process classification and SVM: Mean field results and leave-one-out estimator. In *Advances in Large Margin Classifiers*. MIT Press, 1999.

D. Ormoneit and T. Hastie. Optimal kernel shapes for local linear regression. In S. A. Solla, T. K. Leen, and K-R. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *International Conference on Computer Vision*, 1998.

E. Parzen. On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33:1065–1076, 1962.

J. Platt. Probabilities for support vector machines. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press, Cambridge, MA, 2000.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numercial Recipies in C: The Art of Scientific Computing*. Cambridge University Press, 2nd edition, 1992.

D. Radulovic and M. Wegkamp. Weak convergence of smoothed empirical processes: Beyond donsker classes. In E. Giné, D.M. Mason, and J.A. Wellner, editors, *High Dimensional Probability II*, volume 47 of *Progress in Probability*, pages 89–106. Springer, 2000.

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

M. Rattray. A model-based distance for clustering. In *Proceedings of the International Joint Conference on Neural Networks*, 2000.

R. Reed, R. J. Marks II, and S. Oh. Similarities of error regularization, sigmoid gain scaling target smoothing and training with jitter. *IEEE Transactions on Neural Networks*, 6(3):529–538, May 1995.

J. Rissanen. Stochastic complexity and modeling. *Annals of Statistics*, 14:1080–1100, 1986.

S. Roweis, L. Saul, and G. Hinton. Global coordination of local linear models. In *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002.

S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the International Conference on Machine Learning*, 2001.

C. Saunders, A. Gammermann, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proccedings of the 15th International Conference on Machine Learning*, pages 515–521. Morgan Kaufmann, 1998.

G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *Proceedings of 17th International Conference on Machine Learning*, pages 839–846, San Francisco, CA, 2000. Morgan Kaufmann.

B. Schölkopf, C. Burges, and V. Vapnik. Extracting support data for a given task. In U. M. Fayyad and R. Uthurusamy, editors, *First International Conference on Knowledge Discovery & Data Mining*. AAAI Press, 1995.

B. Schölkopf, C. Burges, and V. Vapnik. Incorporating invariances in support vector learning machines. In *Artificial Neural Networks — ICANN'96*, volume 1112, pages 47–52, Berlin, 1996. Springer Lecture Notes in Computer Science.

B. Schölkopf, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Generalization bounds via eigenvalues of the Gram matrix. Technical Report 99-035, NeuroColt, 1999.

B. Schölkopf, P. Y. Simard, A. J. Smola, and V. N. Vapnik. Prior knowledge in support vector kernels. In MIT Press, editor, *NIPS*, volume 10, 1998a.

B. Schölkopf and A. Smola. *Learning with kernels*. MIT Press, Cambridge, MA, 2002.

B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998b.

B. Schölkopf, K. Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik. Comparing support vector machines with gaussian kernels to radial basis function classifiers. *IEEE Trans. on Signal Processing*, 45(11):2758–2765, 1997.

D. Schuurmans. A new metric-based approach to model selection. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, 1997.

G. Schwartz. Estimating the dimension of a model. *Ann. Stat.*, 6:461–464, 1978.

M. Seeger. Covariance kernels from Bayesian generative models. In *Advances in Neural Information Processing Systems*, volume 14, 2001a.

M. Seeger. Learning with labeled and unlabeled data. Technical report, Edinburgh University, 2001b.

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.

R. Shibata. An optimal selection of regression variables. *Biometrica*, 68:461–464, 1981.

J. Sietsma and R. Dow. Creating articial neural networks that generalize. *Neural Networks*, 4(1):67–79, 1991.

P. Simard, Y. Le Cun, and J. Denker. Efficient pattern recognition using a new transformation distance. In *Advances in Neural Information Processing Systems*, volume 5, pages 50–58. Morgan Kaufmann, 1993.

P. Simard, Y. LeCun, J. Denker, and B. Victorri. Transformation invariance in pattern recognition, tangent distance and tangent propagation. In G. Orr and K. Muller, editors, *Neural Networks: Tricks of the trade*. Springer, 1998.

A. J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In P. Langley, editor, *Proceedings of the 17th International Conference on Machine Learning*, pages 911–918, San Francisco, 2000. Morgan Kaufman.

P. Sollich. Bayesian methods for support vector machines: Evidence and predictive class probabilities. *Machine Learning*, 46(1/3):21, 2002.

M. Spivak. *A Comprehensive Introduction to Differential Geometry*, volume 1, pages 343–383. Publish or perish, Houston, 2nd edition, 1999.

M. Sugiyama and H. Ogawa. Incremental active learning for optimal generalization. *Neural Computation*, 12(12):2909–2940, 2000.

K. K. Sung and P. Niyogi. Active learning for function approximation. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 593–600. The MIT Press, 1995.

M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-Posed Problems*. V.H. Winston & Sons, John Wiley & Sons, Washington D.C., 1977.

M. Tipping. Deriving cluster analytic distance functions from gaussian mixture models. In IEE, editor, *Proceedings of the 9th International Conference on Artificial Neural Networks*, London, 1999.

N. Tishby and N. Slonim. Data clustering by markovian relaxation and the information bottleneck method. In *Advances in Neural Information Processing Systems*, volume 13, pages 640–646, 2000.

S. Tong and D. Koller. Restricted bayes optimal classifiers. In *Seventeenth National Conference on Artificial Intelligence*, pages 658–664, 2000.

S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In *Journal of Machine Learning Research*, volume 2, pages 45–66, 2001.

K. Tsuda. Optimal hyperplane classifier with adaptive norm. Technical Report 99-9, ETL, 1999a.

K. Tsuda. Support vector classifier with asymmetric kernel function. In M. Verleysen, editor, *Proceedings of ESANN'99*, pages 183–188, 1999b.

K. Tsuda, T. Kin, and K. Asai. Marginalized kernels for biological sequences. *Bioinformatics*, 2002. To appear. Also presented at ICMB 2002.

L. G. Valiant. A theory of the learnable. In *ACM Symposium on Theory of Computing*, pages 436–445, 1984.

A. van der Vaart and J. Wellner. *Weak convergence and Empirical Processes*. Springer-Verlag, New York, 1996.

V. Vapnik. *Estimation of dependencies based on empirical data*. Springer, 1982.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 2nd edition, 2000.

V. Vapnik and O. Chapelle. Bounds on error expectation for support vector machines. *Neural Computation*, 12(9), 2000.

V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

V. Vapnik and A. Chervonenkis. The necessary and sufficient conditions for consistency in the empirical risk minimization method. *Pattern recognition and Image Analysis*, 1(3):283–305, 1991.

V. Vapnik and A. Ja Chervonenkis. *Theory of Pattern Recognition*. Nauka, 1974. In russian.

G. Wahba. *Splines Models for Observational Data*, volume 59 of *Applied Mathematics*. SIAM, Philadelphia, 1990.

G. Wahba, G. Golub, and M. Heath. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21:215–223, 1979.

G. Wahba, Y. Lin, and H. Zhang. Generalized approximate cross-validation for support vector machines : another way to look at margin-like quantities. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 297–309. MIT Press, 2000.

G.S. Watson. Smooth regression analysis. *Sankhaya: The Indian Journal of Statistics*, 26:359–372, 1964.

Y. Weiss. Segmentation using eigenvectors: A unifying view. In *International Conference on Computer Vision*, pages 975–982, 1999.

J. Weston, O. Chapelle, and I. Guyon. Data cleaning with support vector machines. Technical report, Biowulf Technologies, 2001.

J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for support vector machines. In *Advances in Neural Information Processing Systems*, 2000.

C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001.

R. C. Williamson, A. Smola, and B. Schölkpof. Generalization performance of regularization networks and support vector machines via entropy numbers of compact operators. *IEEE Transactions on Information Theory*, 47(6):2516–2532, 2001.

D. H. Wolpert. The lack of A priori distinctions between learning algorithms. *Neural Computation*, 8(7): 1341–1390, 1996.

T. Zhang and F. J. Oles. A probability analysis on the value of unlabeled data for classification problems. In *Proceedings of the 17th International Conference on Machine Learning*, pages 1191–1198, San Francisco, CA, 2000. Morgan Kaufmann.

A. Zien, G. Rätsch, S. Mika, B. Schölkpof, T. Lengauer, and K. Müller. Engineering support vector machine kernels that recognize translation initiation sites. *BioInformatics*, 16(9):799–807, 2000.

Résumé

Cette thèse présente une étude théorique et pratique des Support Vector Machines (SVM) et d'autres algorithmes d'apprentissage similaires. Dans une première partie, nous introduisons un nouveau principe d'induction dont les SVMs sont un cas particulier, mais d'autres et de nouveaux algorithmes en sont aussi dérivés. Dans une deuxième partie, après avoir étudié comment estimer l'erreur de généralisation d'une SVM, nous proposons de sélectionner les paramètres du noyau d'une SVM en minimisant cet estimateur. Plusieurs appliquations de ce principe, telle que la sélection de composantes sont présentées. Enfin, dans une troisième partie, nous nous intéressons à l'incorporation de connaissances à priori dans un algorithme d'apprentissage et plus particulièrement le cas de transformations invariantes connues et celui de points non étiquettés.

Abstract

This thesis presents a theoretical and practical study of Support Vector Machines (SVM) and related learning algorithms. In a first part, we introduce a new induction principle from which SVMs can be derived, but some new algorithms are also presented in this framework. In a second part, after studying how to estimate the generalization error of an SVM, we suggest to choose the kernel parameters of an SVM by minimizing this estimate. Several applications such as feature selection are presented. Finally the third part deals with the incoporation of prior knowledge in a learning algorithm and more specifically, we studied the case of known invariant transormations and the use of unlabeled data.