Max–Planck–Institut für biologische Kybernetik
Max Planck Institute for Biological Cybernetics

Technical Report No. TR-101

A compression approach to support vector model selection

Ulrike von Luxburg,[1] Olivier Bousquet,[1]
Bernhard Schölkopf[1]

September 2002

[1] Department Schölkopf, email: ulrike.luxburg@tuebingen.mpg.de

This report is available in PDF–format via anonymous ftp at ftp://ftp.kyb.tuebingen.mpg.de/pub/mpi-memos/pdf/TRcompression02.pdf. The complete series of Technical Reports is documented at: http://www.kyb.tuebingen.mpg.de/techreports.html

A compression approach to support vector model selection

Ulrike von Luxburg, Olivier Bousquet, Bernhard Schölkopf

Abstract. In this paper we investigate connections between statistical learning theory and data compression on the basis of support vector machine (SVM) model selection. Inspired by several generalization bounds we construct "compression coefficients" for SVMs, which measure the amount by which the training labels can be compressed by some classification hypothesis. The main idea is to relate the coding precision of this hypothesis to the width of the margin of the SVM. The compression coefficients connect well known quantities such as R^2/ρ^2, the eigenvalues of the kernel matrix and the number of support vectors. To test whether they are useful in practice we ran model selection experiments on several real world datasets. As a result we found that compression coefficients can fairly accurately predict the parameters for which the test error is minimized.

---

1 Introduction

In classification one tries to learn the dependency of labels y on patterns x from a given training set $(x_i, y_i)_{i=1...m}$. We are interested in the connections of two different methods to analyse this problem: a data compression approach and statistical learning theory.

The minimum description length (MDL) principle (cf. Barron et al., 1998, and references therein) states that, among a given set of hypotheses, one should choose the hypothesis that gives the shortest description of the training data. Intuitively, this seems to be a reasonable choice: by Shannon's source coding theorem (cf. Cover and Thomas, 1991) we know that a short code is closely related to the data generating distribution. Moreover, an easy-to-describe hypothesis is less likely to overfit than a more complicated one. The description of the data is given in form of a code which tries to compress the data as well as possible.

There are several connections of the MDL principle to other learning methods. For instance, it is easy to see that selecting the hypothesis with the highest posterior probability in a Bayesian setting is equivalent to choosing the hypothesis with the shortest code (cf. Hansen and Yu, 2001).

In statistical learning theory, one derives generalization bounds to analyse learning algorithms. The compression scheme approach of Littlestone and Warmuth (1986), which describes the generalization ability of a learning algorithm by its ability to reduce the training set to a few important points, led to a bound for SVMs in terms of the number of support vectors (cf. Bartlett and Shawe-Taylor, 1999). Combining this result with large margin bounds leads to the sparse margin bound of Herbrich et al. (2000). In McAllester (1998), a PAC-Bayesian bound for learning algorithms was derived. For a given prior distribution P on the hypotheses space, it bounds the generalization error essentially by $-\ln P(U)/m$, where U is the subset of hypotheses consistent with the training examples. By Shannon's theorem we see that $-\ln P(U)$ is the length of the shortest code for this subset, which is an indication that coding arguments can also be important in statistical learning theory.

This view is supported by the following theorem of Vapnik (1998, sec. 6.2). For any classification hypothesis h out of a finite hypotheses space he defines its compression coefficient $C(h)$ by the number of bits needed to code the training labels by using h, divided by the number m of labels. Then he connects the compression coefficient to the generalization error of a learning algorithm:

Theorem 1 (Vapnik) *Given a finite set of classification hypotheses, then with probability at least $1-\eta$ over the random draw of m training points, the risk is bounded by $R(h) \leq 2\ln(2)C(h) - \ln(\eta)/m$ for all hypotheses h, where C is the compression coefficient of h and the risk is defined as the expected loss of the hypothesis with respect to a fixed loss function.*

Even though this bound is only valid in a very restricted setting where we have a hypotheses space which is independent of the training data it indicates that compression coefficients can be a helpful tool to analyse learning algorithms.

Inspired by these bounds we want to explore the connection between statistical learning theory and data compression in SVMs. To this end we construct compression coefficients for SVMs (sec.2). It turns out that with this approach, we recover several quantities already known to be meaningful in statistical learning theory such as R^2/ρ^2, the eigenvalues of the kernel matrix, and the number of support vectors. These quantities, which often only appear separately in different generalization bounds, can even be integrated into one single compression coefficient. Our results support the view that there is a profound relation between data compression and statistical learning theory. To test the model selection performance of our compression coefficients we ran experiments on several real world data sets (sec.3). We find that the compression coefficients outperform several standard methods and can compete with the state of the art span bound.

2 Compression coefficients for SVMs

The main idea for constructing compression coefficients for SVMs is an interpretation of the margin in terms of coding precision: Suppose the data are (correctly) classified by a hyperplane with normal vector ω and margin ρ. In case of a large margin it is commonly held (e.g. Schölkopf and Smola, 2002) that small perturbations of the direction of the hyperplane will not change the classification result on the training points (see figure 1a). In

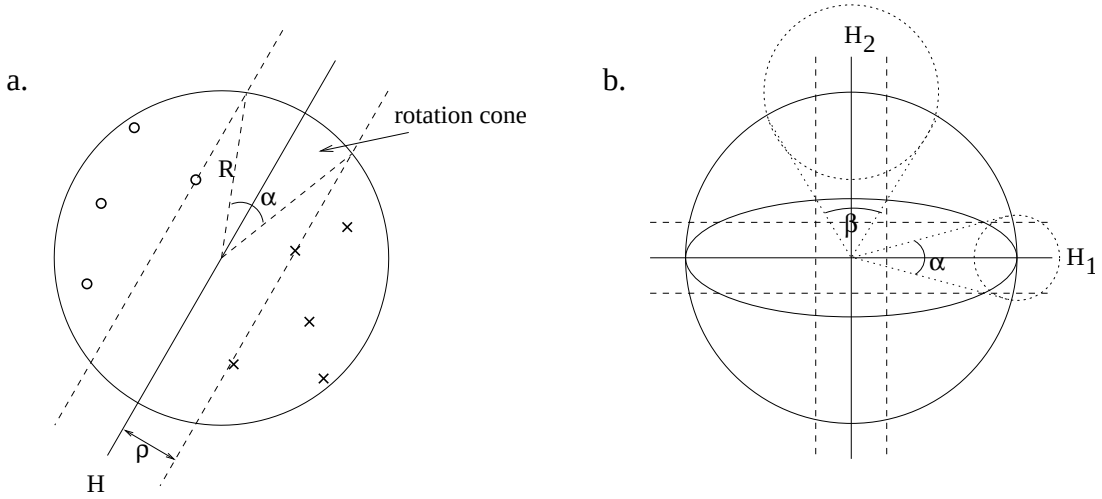

Figure 1: (a.) All hyperplanes within the rotation cone induced by the margin ρ produce the same classification result on the training patterns as H. (b.) In case of an ellipsoid data domain, ρ induces different rotation cone angles depending on the direction of the hyperplane. Their associated covering balls on the spherical hypotheses space also have different sizes. In the example shown, $\alpha \leq \beta$, hence H_1 must be coded with higher accuracy than H_2.

compression language this means that we do not have to code the direction of the hyperplane with high accuracy – the larger the margin, the less accurate we have to code. This is one way to justify the method of maximizing the margin with the help of coding/compression arguments.

More concretely, we are given m pairs $(x_i, y_i)_{i=1,\ldots,m}$ of training patterns with labels. In classification we want to predict the labels y for given patterns x, i.e. we want to learn $P(y|x)$, or at least some aspects of $P(y|x)$, such as the set $\{x|P(y|x) \geq 0.5\}$. So for the coding approach we assume that an imaginary sender and receiver both know the training *patterns*. It will be the task of the sender to transmit the *labels* of the training patterns to the receiver (cf. Vapnik, 1998, sec. 6.2.). Sender and receiver are allowed to agree on the details of the code before transmission starts. First the sender trains an SVM and forms a hypothesis how the data should be separated. In the easiest case this hypothesis is the direction of the separating hyperplane in the feature space. Now he transmits the labels in two parts. First he codes the hypothesis and sends it. Then he transmits which of the training patterns are misclassified by this hypothesis. To decode, the receiver applies the hypothesis to the training patterns and constructs the labels for all patterns. Then he flips the labels of all patterns that are indicated to be misclassified by this hypothesis. After this, the receiver will have reconstructed all labels correctly.

The second part, i.e. coding the misclassifications, is easy. We only have to code the number l of misclassifications ($\log(m)$ bits) and the indices of the misclassified training vectors ($\log \binom{m}{l}$ bits). Here and in the following, $\log$ denotes the base-2 logarithm.

The code of the first part depends on the kind of hypothesis we want to form after having trained the SVM. One way of coding the labels with the help of an SVM was described above: we code the direction of the hyperplane and adapt the precision by which we code this direction to the width of the margin. The space of all hypotheses then consists of the unit sphere in the feature space. Now suppose that all training patterns lie in a ball of radius R around the origin and that they are separated by a hyperplane H through the origin with normal vector ω and margin ρ. Then every "slightly rotated" hyperplane that inside the ball still lies within the margin produces the same classification result on all training points as the original hyperplane. Thus, instead of using ω itself to separate the points we could use any vector in the rotation cone of ω (cf. figure 1a).
To code ω we now choose a set of "equidistant" representation vectors on the sphere of the feature space. Here we have to ensure that in each cone "of width ρ" of the sphere there is at least one representation vector. The number of vectors needed to fulfill this condition can be described as the number of balls of radius ρ needed to cover the sphere of radius R, such that the centers of the covering balls lie on the sphere. An upper bound of this number is the following:

Proposition 2 *To cover the sphere S_R^{d-1} of radius R in the d-dimensional Euclidean space we need less than $2\left\lceil \frac{R\pi}{\rho} \right\rceil^{d-1}$ balls of radius ρ. The exponent in this upper bound is tight.*

Proof We use an inductive procedure to prove the upper bound on the covering number. Let c_d be an upper bound on the number of balls needed to cover the sphere $S_R^{d-1} \subset \mathbb{R}^d$. To construct a ρ-covering on $S_R^d \subset \mathbb{R}^{d+1}$ we cover the cylinder $S_R^{d-1} \times [-R\pi/2, R\pi/2]$ with a grid of $c_{d+1} := c_d \cdot \lceil R\pi/\rho \rceil$ points. The grid is then mapped on the sphere. As the distances between the grid points do not increase by this mapping the projected points form an ρ-cover of the sphere S_R^d. For $d = 2$ the sphere is a circle which can be covered with $\lceil 2R\pi/\rho \rceil \leq 2 \lceil R\pi/\rho \rceil$ balls. By the iteration described above we then get $c_d = 2 \lceil R\pi/\rho \rceil^{d-1}$ for all $d \in \mathbb{N}$.
To show that the exponent is tight we construct a lower bound on the covering number by dividing the area of the surface of the whole sphere by the part of the surface covered by one single ball. As the area of this part is smaller than the whole surface of the small ball we get $(R/\rho)^{d-1}$ as lower bound for the covering number. ☺

The code now looks as follows:

Code 1 *Sender and receiver agree on the training patterns, a fixed kernel and on a procedure for choosing the positions of t balls to cover the unit sphere in a d-dimensional Euclidean space. Now the sender transmits the number n of representation vectors, the index i of the representation vector representing the normal vector of the hyperplane, the number of misclassified training examples, and the indices of the misclassified training examples. To decode this information, the receiver constructs a covering of the sphere in the feature space with $t = n$ balls according to the common procedure, determines the representation vector given by its index among all representation vectors, and constructs a hyperplane using this vector as normal vector. Then he classifies all training patterns according to this hyperplane and flips the labels of the misclassified training examples.*

As the number n of representation vectors is not bounded beforehand we do not have an upper bound on the number of bits needed to code it. Thus, to transmit n we use a code described in Cover and Thomas (1991, p. 149) which uses $2 \log n + 2$ bits. After this, the receiver already knows n and we can code i with $\log n$ bits. Finally, to transmit the number and indices of the misclassified training examples we need $\log(m) + \log \binom{m}{l}$ bits. Using $\binom{m}{l} \leq m^l$ and $n \leq 2 \lceil R\pi/\rho \rceil^{d-1}$ we thus obtain the compression coefficient

$$C_1 \quad \leq \quad \frac{1}{m} \left(3(d-1) \log \left\lceil \frac{R\pi}{\rho} \right\rceil + 3 \log 2 + (l+1) \log m + 2 \right). \tag{1}$$

To refine this analysis we now want to take into account the shape of the data in the feature space. To this end we assume that the training patterns are contained in an ellipsoid with axes $c_1, ..., c_d$ rather than a ball of radius R. When using the rotation argument from above, we observe that in the ellipsoid situation the maximal rotation angle of the hyperplane depends on the actual direction of the hyperplane (cf. figure 1b). The induced covering balls on the spherical hypotheses space now have different sizes, too. Note, that instead of covering the sphere with balls of different sizes we can directly cover the surface of the ellipse with balls of equal sizes. Moreover, to keep the calculations easy, we will cover the whole ellipse instead of the surface only. This means that we use one extra dimension (volume instead of surface), but in high dimensional spaces this does not make much difference.

Proposition 3 *To cover a d-dimensional ellipse with principal axes $c_1, ..., c_d$ we need at most $\prod_{i=1}^{d} \left\lceil \frac{2c_i}{\rho} \right\rceil$ balls of radius ρ. The dependency on $\prod_{i=1}^{d} \frac{c_i}{\rho}$ is tight.*

Proof The smallest parallelepiped containing the ellipse has side lengths $2c_1, ..., 2c_d$ and can be covered with a grid of $\prod_{i=1}^{d} \lceil 2c_i/\rho \rceil$ balls of radius ρ. This gives an upper bound on the covering number.
To obtain a lower bound we divide the volume of the ellipse by the volume of one single ball. Let v_d be the volume of a d-dimensional unit ball. Then the volume of a ball of radius ρ is $\rho^d v_d$ and the volume of an ellipse with axes $c_1, ..., c_d$ is given by $v_d \prod_{i=1}^{d} c_d$. So we need at least $\frac{\prod_{i=1}^{d} c_d}{\rho^d}$ balls. ☺

Proposition 4 *For given training patterns $(x_i)_{i=1,...,m}$ and kernel k, let $\lambda_1, ..., \lambda_m$ be the eigenvalues of the kernel matrix K. Then all training patterns are contained in an ellipse with principal axes $\sqrt{\lambda_1}, ..., \sqrt{\lambda_m}$.*

Proof Let $\mathrm{span}\{\delta_{x_i} | i = 1, ..., m\}$ be the subspace of the feature space spanned by the training examples. It is endowed with the scalar product $\langle \delta_{x_i}, \delta_{x_j} \rangle_K = k(x_i, x_j)$. We map this space to $(l_2, \langle ., . \rangle_{l_2})$ by $\delta_{x_i} \mapsto e_i$, where e_i denotes the i-th unit vector in l_2 and $\langle ., . \rangle_{l_2}$ is the canonical scalar product of l_2. It turns out that for $u, v \in l_2$ we have $\langle u, v \rangle_K = u' K v$. Let $v_1, ..., v_m$ be the l_2-normalized eigenvectors corresponding to the eigenvalues $\lambda_1, ..., \lambda_m$. They satisfy $\langle v_i, v_j \rangle_K = \lambda_i \delta_{ij}$. Now it is easy to check that for all training examples x_i the ellipse inequality

$$\sum_j \left(\frac{\langle \delta_{x_i}, v_j \rangle_K}{||v_j||_K \sqrt{\lambda_j}} \right)^2 \leq 1$$

holds. ☺

Combining these two propositions it turns out that the number of balls of radius ρ we need to cover the data ellipse is given by $n \leq \frac{\prod_{i=1}^{m} 2\sqrt{\lambda_i}}{\rho^d}$. Now we can state the next code:

Code 2 *This code works exactly as code 1, the only difference is that an ellipse is used instead of a ball.*

Counting the number of bits analogously to code 1 we get a compression coefficient of

$$C_2 \quad \leq \quad \frac{1}{m} \left(3 \sum_{i=1}^{d} \log \left\lceil \frac{2\sqrt{\lambda_i}}{\rho} \right\rceil + (l+1) \log m + 2 \right). \tag{2}$$

Both compression coefficients we derived so far depend on the dimension d of the feature space in which we code the hyperplane. It is easy to see that the solution of most kernel algorithms lives in the subspace spanned by the training examples (representer theorem), so that we can always work with $d = m$. For SVMs the solution even lies in the subspace spanned by the support vectors which form a subset of the training examples. Therefore, an easy dimension reduction can be achieved by working in this subspace. To do this we additionally have to code the number s and the indices of the support vectors among the training patterns with $\log m + \log \binom{m}{s}$ bits. This leads to

Code 3 *Sender and receiver agree on the training patterns, the kernel, and the procedure of covering an ellipsoid. After training an SVM the sender transmits the number s of support vectors ($\log m$ bits), the indices of the support vectors ($\log \binom{m}{s}$ bits), the number n of representation vectors ($2 \log n + 2$ bits), the index of the representation vector used ($\log n$ bits), the number l of misclassified patterns ($\log m$ bits), and the indices of the misclassified patterns ($\log \binom{m}{l}$ bits). To decode, the receiver constructs the hypotheses space consisting of the data ellipse projected on the subspace spanned by the support vectors. He covers this ellipse with n balls and chooses the vector representing the normal vector of the hyperplane. Then he labels the training patterns by first projecting them into the subspace and then classifying them according to the hyperplane. Finally, the labels of the misclassified patterns are flipped.*

Using propositions 3 and 4 with the eigenvalues $\gamma_1 \geq ... \geq \gamma_s$ of the kernel matrix restricted to the support vectors we obtain $n \leq \prod_{i=1}^{s} 2\sqrt{\gamma_i}/\rho^s$. So code 3 has a compression coefficient of

$$C_3 \quad \leq \quad \frac{1}{m} \left((l+s+2) \log m + 3 \sum_{i=1}^{s} \log \left\lceil \frac{2\sqrt{\gamma_i}}{\rho} \right\rceil + 2 \right). \tag{3}$$

A further dimension reduction can be obtained with the following idea: The axes of the data ellipse are decreasing fast for large dimensions. Once the axis in one direction is very small we can discard this dimension by projecting the data in a lower dimensional subspace using kernel PCA. Essentially, the idea is to "use up" part of the margin

for the projection error. If, say, the maximal projection error is $\rho/2$, then even after projection the training patterns will lie on the same side of the hyperplane as before. Now we can use the remaining part of the margin to code the hyperplane as before.
More detailed this approach works as follows: First we train the SVM and get the normal vector ω and margin ρ. As hypotheses space we will use the sphere with radius R, i.e. we renorm ω to length R by $\omega_0 := \frac{\omega}{||\omega||}R$. It is important here not to confuse the hypotheses space (sphere) and the data space (ellipse). We place the rotation cone of width ρ around ω_0. Above we argued that in the ellipse setting the rotation cone has different sizes depending on the position of ω in the data ellipse. By using a sphere of radius R and the rotation cone of width ρ we actually consider the smallest possible rotation cone.
Now we perform a principal component decomposition of the training data in feature space. The principal components are the directions of the axes of the ellipse. We sort the principal components such that their eigenvalues $\lambda_1, ..., \lambda_m$ decrease. For $d_P \in \{1, ..., m\}$ let P be the projection on the the subspace spanned by the first d_P eigenvectors.
Now we have to check if the projected hypotheses $P(\omega_0)$ still lies within the rotation cone. If no, the projection error made by P is too big and we cannot project on the d_P-dimensional subspace. If yes, we are still within the allowed bound of precision after projecting ω_0, i.e. we can discard the last $m - d_P$ dimensions. In this case we can encode the projected normal vector $P(\omega_0)$ in an d_P-dimensional space and we call P a valid projection. As we already made some error by projecting ω_0 into this low-dimensional space we have to code more precisely now. For a valid projection we can compute the part of the margin it uses by

$$c_P := \frac{||\omega_0 - P(\omega_0)||}{\rho} = \frac{||\omega - P(\omega)|| \cdot R}{||\omega||\rho} = R \cdot ||\omega - P(\omega)|| \in [0, 1].$$

The remaining part of the margin can be used for the error made by coding $P(\omega_0)$ with a representation vector $r(\omega_0)$ of a covering of the d_P-dimensional data ellipse, i.e. we have to cover the ellipse with balls of radius $\sqrt{1 - c_P^2}\rho$. Then the total error made by this code can be calculated as

$$||\omega_0 - r(\omega_0)||^2 = ||\omega_0 - P(\omega_0)||^2 + ||P(\omega_0) - r(\omega_0)||^2 \leq c_P^2\rho^2 + (1 - c_P^2)\rho^2 = \rho^2$$

i.e the representation vector $r(\omega_0)$ still lies within the original rotation cone.
As we already used in the construction of code 3 we can always work in the subspace spanned by the support vectors. Starting from this subspace we try to project ω on smaller subspaces. For this we use a PCA only of the support vectors. So the code looks as follows:

Code 4 *Sender and receiver agree upon the training patterns, the kernel, the procedure of covering ellipsoids, and on how to perform a PCA. The sender trains the SVM and chooses a valid projection on some subspace. He transmits the number s of support vectors ($\log m$ bits), the position of the support vectors ($\log \binom{m}{s}$ bits), the number $m - d_P$ of dimensions to discard ($\log m$ bits), the number n of representation vectors ($2 \log n + 2$ bits), the index of the representation vector used ($\log n$ bits), the number l of misclassified patterns ($\log m$ bits), and the position of the misclassified patterns ($\log \binom{m}{l}$ bits). To decode, the receiver constructs the hypotheses space consisting of the ellipse projected on the subspace spanned by the support vectors. Then he performs a PCA on the support vectors and projects the hypotheses space on the subspace spanned by the first d_P principal components. In this subspace he covers the remaining ellipse with n balls, chooses the representation vector as indicated and uses it to construct a hyperplane in the subspace. Now he projects all training data in this subspace, classifies it according to the hyperplane and flips the labels of the misclassified patterns.*

For fixed number d_P the compression coefficient for this code is given by

$$\begin{aligned} C_4 &= \frac{1}{m}\left(3\log m + \log\binom{m}{l} + \log\binom{m}{s} + 3\log n + 2\right) \\ &\leq \frac{1}{m}\left(3\log m + \log\binom{m}{l} + \log\binom{m}{s} + 3\sum_{i=1}^{d_P}\log\left\lceil\frac{\sqrt{2\gamma_i}}{\sqrt{1-c_P^2}\,\rho}\right\rceil + 2\right) \quad (4) \\ &\leq \frac{1}{m}\left((l+s+3)\log m + 3\sum_{i=1}^{d_P}\log\left\lceil\frac{\sqrt{2\gamma_i}}{\sqrt{1-c_P^2}\,\rho}\right\rceil + 2\right) \quad (5) \end{aligned}$$

where c_P is determined as described above and γ_i are the eigenvalues of the kernel matrix restricted to the support vectors only.

So far we always considered codes that used the direction of the hyperplane as hypothesis. A totally different approach is to reduce the data by transmitting support vectors only. This quite simple code works as follows:

Code 5 *Sender and receiver agree on training patterns and a kernel. The sender sends the number s of support vectors ($\log(m)$ bits), the indices of the support vectors ($\log \binom{m}{s}$ bits), the labels of the support vectors (s bits), the number l of misclassified training patterns ($\log(m)$ bits), and the indices of the misclassified training patterns ($\log \binom{m}{l}$ bits). To decode this information, the receiver trains an SVM with the agreed kernel and with the support vectors as training set. Then he computes the classification result of this SVM for the original training patterns and flips the labels of the misclassified patterns.*

This code leads to a compression coefficient of

$$C_5 \leq \frac{1}{m}\left((l+s+2)\log m + s\right). \tag{6}$$

3 Experiments

To test the utility of the derived compression coefficients for model selection we ran experiments on different real world datasets: the Pima Indians diabetes database (768 data points), the Wisconsin breast cancer database (683 data points) and the abalone dataset (4177 data points) from the UCI machine learning repository, and the US postal service handwritten digit recognition benchmark (9298 data points). In all experiments we first permuted the whole dataset and divided it into subsets of a certain sample size (100, 200 or 500). Then, to be consistent with the rotation argument, we centered each subset in feature space. We trained hard and soft margin SVMs ($C = 0.1, 10, 1000, 100000, \infty$) with Gaussian kernels ($\sigma = 10^{-3}, 10^{-2}, ..., 10^5$). The test error was computed on the training subset's complements. We plotted the mean values of the test errors and compression coefficients over all training subsets. Error bars depict standard deviations. We compared our results to several other model selection criteria: the trace criterion $\sqrt{\sum \lambda_i / m^2 \rho^2}$ (Bartlett and Mendelson, 2001), the sparse margin criterion $\sqrt{R^2/\rho^2(m-s)}$ (Herbrich et al., 2000), the span bound (cf. Vapnik and Chapelle (2000) and Opper and Winther (2000)) and the well known radius-margin bound $\sqrt{R^2/m\rho^2}$ (Vapnik, 1998). Note that there exists an improved version of the latter bound, which uses rescaling to adapt to the shape of the data in the feature space (Chapelle and Vapnik, 2000). We plan to use this version instead of the standard version in future experiments. In figure 2 we plotted all results for the abalone dataset, in figure 3 we show some results on different datasets.

It turns out that some of the compression coefficients predict the shape of the test error curve quite well, even if their absolute values are > 1 because of too small sample sizes. The best results seem to be obtained by the compression coefficients C_2 and C_3 , but even the very simple coefficient C_5 looks better than some of the standard bounds. The projection bound C_4 does not produce better results than C_3. The standard deviations of the compression coefficients are small compared to the span bound, which indicates that the compression results are quite stable.

In another experiment we went one step further than only plotting the curves and actually used the bounds for model selection. We focused on selecting the value of σ, fixing C to a constant value. Comparing the test errors of the selected model to the optimal ones, we found that in most cases C_2 performs well (better than C_3, although the latter looks better in figure 3), in general only slightly beaten by the span bound, but nearly always better than the radius-margin bound or the sparse margin bound.

4 Discussion

We derived compression coefficients for SVMs which combine well-known quantities such as the margin, the eigenvalues of the kernel matrix and the number of support vectors. The results of our experiments suggest that these compression coefficients can be a good tool to describe the generalization error of SVMs. As they are easy to compute they can be readily used in practice. We think, performance can even be improved by using tighter bounds in the derivation (e.g., when bounding the covering numbers or the binomial coefficients).

What is still missing is a mathematically rigorous proof for a generalization bound in terms of general compression coefficients. In this context it is important to notice that the proof of the compression bound in theorem 1 is only valid in the situation where the hypotheses space is fixed independent of the training data. In our case, however, the

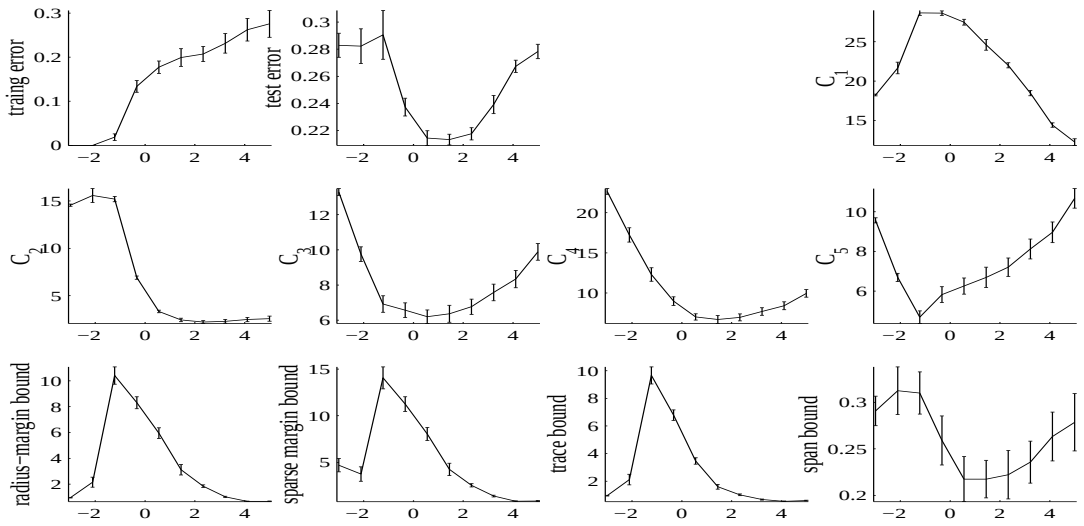

Figure 2: Plots for abalone dataset with $C = 1000$ and sample size $m = 500$. The x–scale shows the logarithm of the kernel width σ. Shown are training and test error, the compression coefficients C_1 to C_5 as well as several other model selection criteria (see text).

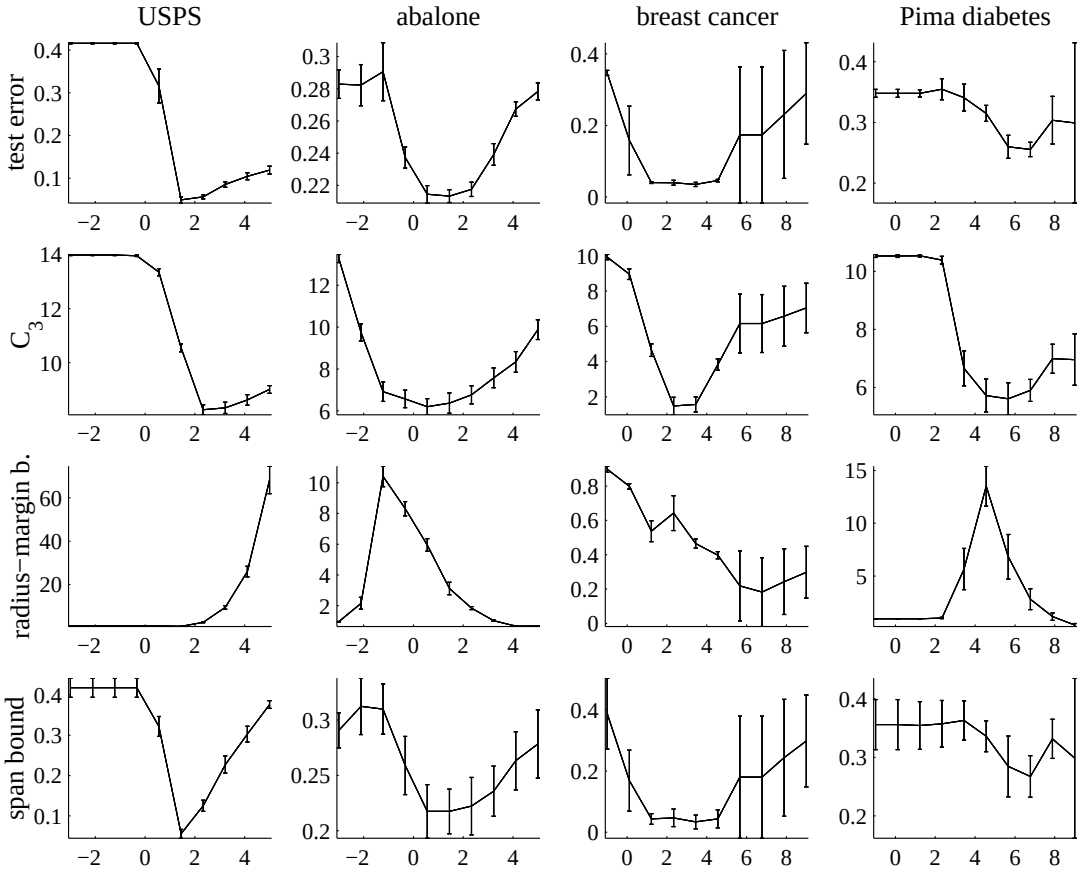

Figure 3: Plots of selected criteria for all datasets. The x–scale shows $\log(\sigma)$. Parameters were $C = 1000, m = 500$ for abalone, $C = 1000, m = 100$ for diabetes, $C = 1000, m = 100$ for breast cancer and $C = \infty, m = 500$ for USPS.

hypotheses space is data dependent and not known beforehand. Only after having trained the SVM we know the hypotheses and how many bits we need to code them. As it is not obvious how to extend the proof of theorem 1 to this more complicated case, it is necessary to develop new methods to derive data dependent compression bounds.

References

A. Barron, J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling. *IEEE Trans. Inform. Theory*, 44(6):2743 – 2760, 1998.

P. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. In *Proceedings of the 14th annual conference on Computational Learning Theory*, pages 273–288, 2001.

P. Bartlett and J. Shawe-Taylor. Generalization performance of support vector machines and other pattern classifiers. In C. Burges B. Schölkopf and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*. MIT press, 1999.

O. Chapelle and V. Vapnik. Model selection for support vector machines. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*. MIT Press, 2000.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 1991.

Mark H. Hansen and Bin Yu. Model selection and the principle of minimum description length. *Journal of the American Statistical Association*, 96(454):746–774, 2001.

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

N. Littlestone and M. Warmuth. Relating data compression and learnability. Unpublished manuscript, 1986.

D. McAllester. Some PAC-bayesian theorems. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, 1998.

M. Opper and O. Winther. Gaussian processes and SVM: mean field and leave-one-out. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 311–326, Cambridge, MA, 2000. MIT Press.

B. Schölkopf and A. Smola. *Learning with kernels. Support Vector Machines, Regularization, Optimization and Beyond*. MIT press, 2002.

V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

V. Vapnik and O. Chapelle. Bounds on error expectation for support vector machines. *Neural Computation*, 12(9): 2013–2036, 2000.