AIC and BIC based approaches for SVM parameter value estimation with RBF kernels

Sergey Demyanov S.DEMYANOV@STUDENT.UNIMELB.EDU.AU
James Bailey BAILEYJ@UNIMELB.EDU.AU
Kotagiri Ramamohanarao KOTAGIRI@UNIMELB.EDU.AU
Christopher Leckie CALECKIE@UNIMELB.EDU.AU
Department of Computing and Information Systems
The University of Melbourne
Parkville, VIC 3010, Australia

Editor: Steven C.H. Hoi and Wray Buntine

Abstract

We study the problem of selecting the best parameter values to use for a support vector machine (SVM) with RBF kernel. Our methods extend the well-known formulas for AIC and BIC, and we present two alternative approaches for calculating the necessary likelihood functions for these formulas. Our first approach is based on using the distances of support vectors from the separating hyperplane. Our second approach estimates the probability that the SVM hyperplane coincides with the Bayes classifier, by analysing the disposition of points in the kernel feature space. We experimentally compare our two approaches with several existing methods and show they are able to achieve good accuracy, whilst also having low running time.

Keywords: AIC, BIC, SVM, RBF kernel, model selection

1. Introduction

The support vector machine (SVM) is well known and one of the most popular methods for classification problems. In its basic form it is a linear classification method, but by using different types of kernels one can model nonlinear separating surfaces, extending the utility and power of the method. One of the most popular SVM kernels is the RBF (Radial Basis Function) kernel.

When using this kernel, it is necessary to choose values for the kernel parameter γ and soft margin parameter C. Together, these define the “dimensionality” of the kernel space (we will later make this concept of dimensionality more precise) and thus a wrong choice for them may cause underfitting or overfitting. Thus, choosing appropriate values for these parameters is crucially important for achieving high classification quality.

The task of searching for the best values of the parameters of a SVM with the RBF kernel has two broad types of approach. The first and most popular is as follows. First, select values for the γ and C parameters. Next, using a test set, estimate the performance of the SVM using these parameter values. One then chooses the parameter values that deliver the best performance compared to any other. More specifically, the original training set is separated into two parts (e.g., 80% and 20%), where the larger part is used for training and the smaller one for testing. This works well when the dataset is large, but can be unreliable when the dataset is small, since the difference between a classifier trained on 100% of the data versus one trained on 80% becomes significant. Moreover, the small size of the test sample may mean the estimated performance is subject to high variance, meaning that a suboptimal choice of parameter values becomes more likely. To address this, "k-fold cross-validation" is often used. The original dataset is split into k parts, where for example k is equal to 5 or 10. Each part is then used as a test set and the other $k-1$ parts are used as a training set. Results over all test sets are then averaged. When k is large this method gives an almost unbiased estimation of true performance. However, it requires the classifier to be trained k times. If we try to choose the best values for C and γ among $10 \times 10 = 100$ parameter value combinations and $k = 10$, it would require 1000 classifiers to be trained. Even for small dataset sizes, this may be prohibitively expensive in terms of running time.

The second approach to selecting of values for γ and C in the SVM RBF kernel uses different approximations of the test error based on the training error and/or different characteristics of the SVM classifier. An example of such an estimation is the radius-margin bound introduced by Vapnik in (Vapnik, 1998). Other examples include the trace bound of Bartlett and Mendelson (2001), compression scheme of Floyd and Warmuth (2005) and sparse margin bound of Herbrich et al. (2000). Also noteworthy are the compression approach described in (Luxburg et al., 2004) and the span estimation introduced in (Vapnik and Chapelle, 2000). Work in (Luxburg et al., 2004) makes a comparison between methods that follow this second type of approach and argues that the compression approach and span estimation are the best. The idea of span estimation was further developed in (Chapelle et al., 2002), where authors provided a gradient descent algorithm for two similar approximations of Leave-One-Out error. It was shown that the proposed method converges to almost optimal parameters for only few iterations of the algorithm. This method has not been considered in our paper, but it will be included in the future work.

In this paper we propose two new techniques for SVM parameter value estimation for the RBF kernel, based on an analysis of the locations of support vectors. It is well known that increasing the flexibility of the classifier can achieve a reduction in the misclassification rate for the training samples. However, the plot for such a reduction usually has an "elbow": after a certain degree of flexibility, the misclassification rate continues to decrease very slowly or even remains constant. This kind of idea is the motivation for our first approach. For the second approach, if, after classification, we observe large connected regions of misclassified objects, this indicates that these regions might be easily modelled by using a more flexible classifier. Once an appropriate classifier has been found, all errors have approximately the same distribution along the hyperplane and it is not possible to significantly decrease the number of errors by small changes in the classifier's shape. This property may be represented as a monotonically decreasing function also having an "elbow". The well known AIC and BIC formulas provide a way to construct a convex function attaining its minimum at this "elbow" point.

We now present an outline of the rest of the paper. In the first section we describe the background for SVMs and kernel functions. Section 2 describes the background for the AIC/BIC formulas and aspects of their derivation. In Section 3 we present general considerations about how AIC and BIC may be applied as part of the search for optimal SVM parameter values for RBF kernel. This is then followed by derivations of our two likelihood functions. After this, we conduct an experimental study of the performance of our methods and compare them to existing techniques. Finally, we discuss the implications of our results and outline future work.

2. Background on Support Vector Machines

The idea behind SVMs is to construct a separating hyperplane that has the largest distance from the closest points of different classes. It has several important properties. First, it uses quadratic programming and can be computed relatively efficiently even for large numbers of points. Second, it is a sparse method, meaning that the location of the hyperplane depends only on a small number of data points, called the support vectors. Third, the ability to use kernels provides an elegant way to generalize for when the separating surface is non linear. However, the problem of finding the most appropriate kernel and parameter values for the kernel remains an open problem.

Let us consider the classification problem with two classes $Y = \{-1, 1\}$ in a space $X = \mathbb{R}^d$. We construct the classifier

$$a(x) = sign(\sum_{i=1}^{n} \omega_i x^i + b) = sign(\langle \omega, x \rangle + b),$$

where $x = (x^1, \ldots, x^d)$ is a feature vector, and ω, b are computed by the SVM. The equation $\langle \omega, x \rangle + b = 0$ describes the separating hyperplane. The original SVM minimization problem is the following:

$$\begin{cases} \frac{1}{2C}\langle \omega, \omega \rangle + \sum_{i=1}^{N} \xi_i \to \min\limits_{\omega, b, \xi} \\ y_i(\langle \omega, x_i \rangle + b) \geq 1 - \xi_i \; \forall i = 1, \ldots, N \\ \xi_i \geq 0 \; \forall i = 1, \ldots, N \end{cases} \tag{1}$$

for some constant $C > 0$. The value of this constant is a parameter of the algorithm.

Instead of this, it is much easier to solve a dual problem, which is a well studied quadratic programming problem:

$$\begin{cases} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \langle x_i, x_j \rangle - \sum_{i=1}^{N} \lambda_i \to \min\limits_{\lambda}; \\ 0 \leq \lambda_i \leq C \; \forall i = 1, \ldots, N; \\ \sum_{i=1}^{N} \lambda_i y_i = 0. \end{cases} \tag{2}$$

When this problem is solved, ω and b are calculated as

$$\begin{cases} \omega = \sum_{i=1}^{N} \lambda_i y_i x_i; \\ b = y_i - \langle \omega, x_i \rangle \; \forall i : 0 < \lambda_i < C. \end{cases}$$

Here $\lambda_i \geq 0$ only for points that have a distance from the hyperplane $y_i(\langle \omega, x_i \rangle + b) \leq 1$. These points are called *support vectors*. Moreover, $\lambda_i < C$ only for *margin support vectors*, i.e., for points having the distance from the hyperplane $d_i = y_i(\langle \omega, x_i \rangle + b) = 1$. For all other support vectors $\lambda_i = C$ and $d_i < 1$. Notice that points having $\lambda_i = 0$ do not influence the solution (which is why the algorithm has a "sparsity" property).

Another property that makes SVMs popular is the possibility to use kernels. If the dataset is poorly separated in the original feature space, it may be the case that it is better separated in some space H with more dimensions. Consider some function $\phi(x) : X \to H$, which transforms original vectors into such a space. The optimization problem does not depend directly on the features, only on dot products, so the only difference compared to the original SVM formulation is that now we need to use the dot product in the new space $\langle \phi(x), \phi(x') \rangle$.

A function $K(x, x')$ is called a kernel if it can be represented in the form $\langle \phi(x), \phi(x') \rangle$ for some function $\phi : X \to H$. Kernels can be very different, which leads to a significant variety of possible algorithms. In all cases we simply substitute the original dot product by its analogue given by the kernel function:

$$\begin{cases} a(x) = sign(\sum_{i=1}^{N} \lambda_i y_i K(x, x_i) + b) \\ b = y_i - \sum_{j=1}^{N} \lambda_i y_i K(x_i, x_j) \ \forall i : 0 < \lambda_i < C. \end{cases} \tag{3}$$

The RBF and polynomial kernels are the most popular classical SVM kernels. The first one is defined for two vectors and $\gamma > 0$ according to the formula

$$K(x, y) = \langle \phi(x), \phi(y) \rangle = \exp(-\gamma ||x - y||^2)$$

Although many kernels have been proposed, no single kernel is the best for all problems. When using very small values of γ in the RBF kernel, the dot product for any two points is close to 1 and so according to the third equation in (2) and (3) $a(x) = sign(b)$. This is the situation of underfitting. At the same time, very large values of γ lead us to an identity dot product matrix, implying each point from the training set is classified to its own class. This situation is called overfitting. Thus, there is the challenge to select the best values of γ (and C) for the RBF kernel.

3. Background on AIC and BIC

Next we provide an overview of these two well known criteria. As mentioned earlier, the problem of model selection is to find a compromise between the quality of classification and generalization. In 1974, Hirotsugu Akaike presented a formula which is expected to obtain its minimum when applied to the best model. It is simply the sum of the negative log likelihood of the observed data under the current model plus the number of its fitted parameters. This method is known as Akaike Information Criterion (AIC). The first term of this function reflects the quality of fitting, whilst the second penalizes overly complex models. Its derivation is based on the Kullback-Leibler divergence - a (non-symmetric) measure of difference between two probability distributions.

Let Z be the observed data, M_i the candidate model, θ_i the vector of parameters in the model M_i, $|\theta_i|$ its length, and $\tilde{\theta}_i$ and $\hat{\theta}_i$ the best and the maximum likelihood vectors of parameters for this model. Assuming that $p(x)$ is a true p.d.f. and $q(x|\theta_i)$ is a parametrized p.d.f. in the model M_i, $\tilde{\theta}_i$ minimizes $D_{KL}(p(x)||q(x|\theta_i))$. This is equivalent to

$$E_p\left[\frac{\partial}{\partial\theta_i}\log(q(x|\theta_i))|_{\theta_i=\tilde{\theta}_i}\right]=0,$$

where E_p is the expectation with respect to the distribution $p(x)$. Applying a Taylor expansion to the function $\log L(\hat{\theta}_i|Z)=\log q(Z|\hat{\theta}_i)$ around the point $\tilde{\theta}_i$ and taking the expectation, we get the first order term equal to zero. After using the approximation for the second order term, we obtain the final result:

$$-\log L(M_i|Z)\approx-\log L(\hat{\theta}_i|Z)+|\theta_i|$$

For our experiments we use the corrected version AIC_c

$$-\log L(M_i|Z)\approx-\log L(\hat{\theta}_i|Z)+|\theta_i|+\frac{|\theta_i|(|\theta_i|+1)}{N-|\theta_i|-1}, \tag{4}$$

which is necessary when the relation between dimensionality and total number of observations $|\theta_i|\ll N$ does not hold. An in-depth discussion about AIC may be found in (Burnham and Anderson, 2002).

In 1978 Schwarz presented another approach, arising from a probabilistic description of the data given a current model. This method came to be known as the Bayesian Information Criterion (BIC). It has almost the same form as the AIC. However, it penalizes complex models more heavily. Instead of expansion around the unknown vector $\tilde{\theta}_i$, the expansion around the known $\hat{\theta}_i$ is used. A comprehensive description of the BIC derivation may be found in (Bhat and Kumar, 2008). Using the same notation as for AIC, we reproduce the final result:

$$\log L(M_i|Z)\approx\log L(\hat{\theta}_i|Z)+\log g_i(\hat{\theta}_i)+\frac{|\theta_i|}{2}\log 2\pi-\frac{\log|H_{\theta_i}|}{2}$$

Here $g(\theta_i)$ is the prior probability of parameters θ_i, H_{θ_i} is the Hessian matrix, consisting of

$$H^{mn}_{\theta_i}=-\frac{\partial^2[\log L(\hat{\theta}_i|Z)+\log g_i(\hat{\theta}_i)]}{\partial\theta^m_i\partial\theta^n_i}|_{\theta_i=\hat{\theta}_i}.$$

When $g(\theta_i)$ is flat, it can be omitted here, as well as in the main formula for $\log L(M_i|Z)$, since it has zero derivative and does not influence the best model $\arg\min_i L(M_i|Z)$. Looking ahead, we will see later that a flat prior is preferable.

Since for independent observations

$$\log L(\hat{\theta}_i|Z)=\sum_{j=1}^{N}\log L(\hat{\theta}_i|z_j)\xrightarrow{N\to\infty}NE_p[\log L(\hat{\theta}_i|z)], \tag{5}$$

the expression for $H^{mn}_{\theta_i}$ is usually transformed into

$$H^{mn}_{\theta_i}=-N\frac{\partial^2E[\log L(\hat{\theta}_i|z)]}{\partial\theta^m_i\partial\theta^n_i}|_{\theta_i=\hat{\theta}_i}=NI^{mn}_{\theta_i}.$$

Of course, the determinant of the Fisher information matrix I_{θ_i} must be non-zero. When $N \to \infty$, it remains constant and therefore is also usually omitted. Thus, the final formula for BIC is the following:

$$- \log L(M_i|Z) \approx - \log L(\hat{\theta}_i|Z) + \frac{|\theta_i|}{2} \log \frac{N}{2\pi} \tag{6}$$

In a classification problem the observed data Z consists of observations X and their labels Y. In the SVM classifier the vector of free parameters θ_i contains the coordinates of the normal vector for the separating hyperplane in the kernel space ω and the value of the bias b. This bias term might also be considered as a coordinate of an extended vector ω, so instead of $L(M_i|Z)$ we use the notation $L(\omega|X, Y)$.

4. Searching for optimal values of γ and C parameters in RBF Kernel

Next, we offer some perspectives about how the AIC and BIC criteria might be used for the problem of choosing optimal parameter values for the SVM. In the case of searching for the best values of γ and C with RBF kernels, each pair of values for γ, C identifies an individual model M_i.

Let some p.d.f. be a true distribution for a dataset. Each kernel function transforms it to a new kernel p.d.f. in the transformed feature space. This new p.d.f implies a corresponding (optimal) Bayes classifier in the same space. This classifier has a form $a(x) = \arg\max_y P_y p_y(x)$. Here P_y are the prior probabilities of the classes, $p_y(x)$ is the p.d.f. for each class, $y \in \{-1, 1\}$. Since it is the best classifier for the current dataset, the size of the region where the class predicted by the SVM differs from the Bayes classifier is desired to be as small as possible. The perfect situation is when all points of the SVM separating hyperplane are the points of the Bayes separating surface and vice versa. So one way to achieve this is to consider only the class p.d.f.'s with the following property: after Maximum Likelihood estimation of their parameters we obtain the (optimal) Bayes classifier coinciding with the SVM classifier for the same sample. However, a true dataset's p.d.f. might differ significantly even from the best p.d.f of this class. Another approach is to directly estimate whether the SVM hyperplane coincides with the Bayes separating surface. We will present two methods, each based on one of these approaches.

In order to employ AIC and BIC functions, we also need to identify the dimensionality (number of free parameters for the separating hyperplane in the kernel space) $|\theta_i|$ for each pair of parameters. It is easy to do this when using a polynomial kernel, but less obvious for the RBF kernel. The number of free parameters for the hyperplane is not higher than the dimensionality of the span of points it separates. Since non-support vectors have no influence on the hyperplane's position, we can consider that $|\theta_i| \leq N_{sv}$. From another side, we have N_{sv}^m margin support vectors (i.e., $0 < \lambda_i < C$), and each of them has the distance from the hyperplane $d_i = y_i(\langle \omega, x_i \rangle + b) = 1$. This system of linear equations for ω and b has at least one solution only when the number of free parameters $|\theta_i| = |\omega| + 1 \geq N_{sv}^m$. Since this lower bound showed better result than N_{sv}, we used it in our experiments.

For large values of γ (corresponding to overfitting), the dimensionality $|\theta_i|$ is often equal to N and therefore the denominator in the AIC formula (4) is equal to -1. In this situation we considered the value of AIC to be ∞, so it was higher than all other non-infinite AIC values. Since we were searching for the parameter value pair that had achieved the minimum for the AIC, such γ and C could not be the solution.

5. Approach 1: Margin-based approach

In this section we derive the likelihood function of our first approach.

As we wrote before, in this first approach we consider the class of p.d.f.'s with the following property: after Maximum Likelihood estimation of their parameters, the plug-in Bayes classifier coincides with the SVM classifier. There exist methods to construct such functions. For instance, Sollich (2002) introduced the third "don't know" class to avoid issues with normalization. In (Grandvalet et al., 2005), probability intervals were used to interpret SVM outputs and thus the SVM solution is approximated with negative log-likelihood. Work in (Platt, 2000) used a simple sigmoid function for the transformation of outputs to posterior probabilities.

Franc, Zien and Schölkopf in (Franc et al., 2011) showed that the joint p.d.f. of an observation x and its class y

$$p(x, y|\omega) = Z(||\omega||) \cdot \exp(-l(d)) \cdot h(\phi(x))$$

with a value $||\omega||$ obtained from the SVM solution that satisfies this condition. Here ω is a normal vector to the SVM separating hyperplane in the kernel feature space, $l(d) = max(1 - d, 0)$ is the hinge loss function, $d = y\langle\omega, x\rangle$ is the distance from the hyperplane, $Z(||\omega||)$ is a normalization constant, and $h(x)$ is some positive integrable function, such that $h(x_1) = h(x_2)\ \forall x_1, x_2 : ||x_1|| = ||x_2||$. They also showed that the plug-in Bayes classifier is linear and coincides with the SVM solution for equal misclassification costs. Thus, assuming that observations have this type of p.d.f., we can use the likelihood function of the SVM solution in the AIC/BIC formulas.

The negative logarithm of this likelihood function given data X, Y can be written as

$$-\log L(\omega|X, Y) = -\sum_{i=1}^{N} \log p(x, y|\omega) = \sum_{i=1}^{N} l(d_i) - \sum_{i=1}^{N} \log(h(\phi(x_i))) - N \log Z(||\omega||). \quad (7)$$

Points with $d_i \geq 1$ do not influence the SVM solution, so they have no influence on the maximum likelihood solution, which gives the same result. This means that fitting $p(x, y|\omega)$ and the conditional p.d.f. $p(x, y|d < 1, \omega)$ must be equivalent. Indeed, in the first term we can use only points having $d_i < 1$, because all others have zero loss. Since $||\omega||$ is given, the solution also does not depend on $h(x)$ and $Z(\omega)$. Therefore, for a fixed $||\omega||$

$$\arg\min_{\omega} [-\log L(\omega|X, Y)] = \arg\min_{\omega} \sum_{i=1}^{N} l(d_i) \quad (8)$$

This result is not surprising if we look at the original function (1) that the SVM algorithm aims to minimize and notice that $\xi_i = l(d_i)$. The first term in the formula (7) is linear as a function of ω, so its second derivative is zero. The second term depends only on $||x||$ and therefore has zero second derivative as well. However the third term $Z(||\omega||)$ gives non-zero diagonal elements for the matrix I_{θ_i} and thus it has a non-zero determinant.

Extending this connection, we can assign a prior distribution for $||\omega||$ to be $g_i(\omega) = \exp(-||\omega||^2/(2C))$ in order to ensure its negative logarithm is equal to the first term in the function (1). However, as our experimental analysis will show, using a simple flat prior for $||\omega||$ gives better results. This method requires only the calculation of

$$\sum_{i=1}^{N} l(d_i) = \sum_{i=1}^{N} l(y_i \sum_{j=1}^{N_{sv}} y_j \lambda_j K(x_i, x_j)),$$

which takes $O(N^2)$ operations. In practice it is a negligible amount of time in comparison to the SVM training procedure.

6. Approach 2: Density-based approach

Next we derive the likelihood function for our second approach based on AIC/BIC.

If for some γ the SVM hyperplane coincides with the (optimal) Bayes classifier for a true data distribution $p(x, y)$, the performance of the SVM cannot be improved by increasing the value of γ (which implies increasing the kernel dimensionality). The Bayes classifier for equal misclassification costs has the form $a(x) = \arg\max_y P_y p_y(x)$. Here P_y are prior probabilities of classes, and $p_y(x)$ is the p.d.f. for each class, $y \in \{-1, 1\}$. This means that for each point x on the separating hyperplane, the equation $P_1 p_1(x) = P_{-1} p_{-1}(x)$ holds.

In order to make the Bayes classifier's surface coincide with SVM separating hyperplane, we require all points that have been classified to the class y satisfy the inequality $P_y p_y(x) > P_{-y} p_{-y}(x)$. For continuous p.d.f. functions it implies that only for points on the hyperplane it is true that $P_1 p_1(x) = P_{-1} p_{-1}(x)$, i.e., this is equivalent to the original requirement. To use the AIC and BIC formulas, we need to calculate the probability that these inequalities hold for every point in the dataset. Assuming that at each point it holds independently from others, total likelihood of the SVM solution can be expressed as

$$L(\omega|X, Y) = \prod_i \Pr\left[P_{y(i)} p_{y(i)}(x_i) > P_{-y(i)} p_{-y(i)}(x_i)\right]$$

Let us fix some point x_i of the dataset and a sphere of radius r around this point. If we consider that density functions $p_{y_i}(x)$ and $p_{-y_i}(x)$ are constant inside this sphere, the probability of a point in it being from class y_i might be calculated using the following formula:

$$q_i = \frac{P_{y_i} p_{y_i}(x_i)}{P_{y_i} p_{y_i}(x_i) + P_{-y_i} p_{-y_i}(x_i)} \tag{9}$$

Using q_i, the previous expression may be rewritten as

$$-\log L(\omega|X, Y) = -\sum_i \log \Pr\left[q_i > \frac{1}{2}\right]$$

Let $n_1^r(x_i)$ and $n_{-1}^r(x_i)$ denote the number of positive and negative points in this sphere. Since we suppose that all points in the dataset are independently drawn from the same distribution, the probability of obtaining values $n_1^r(x_i)$ and $n_{-1}^r(x_i)$ for a certain q_i is simply the value of the binomial distribution $Bi(q_i; n^r_{y(i)}(x_i), n^r_{y(i)}(x_i) + n^r_{-y(i)}(x_i))$ with parameters q_i as a probability of success and $n^r_{y(i)}(x_i)$ successful outcomes among $n^r_{y(i)}(x_i) + n^r_{-y(i)}(x_i)$ experiments. Given the certain values of $n^r_{y(i)}(x_i)$ and $n^r_{-y(i)}(x_i)$, we are interested in the probability that $q_i > 1/2$.

For extreme values $q \leq 0$ and $q \geq 1$ $Bi(q; n^r_y, n^r_1 + n^r_{-y}) = 0$, so this function is integrable and therefore after normalization it may be considered as a probability density function of the parameter q. Hence,

$$\begin{aligned} \Pr\left[q > \frac{1}{2}\right] &= \frac{\int_{1/2}^{1} Bi(q; n^r_y, n^r_y + n^r_{-y})}{\int_0^1 Bi(q; n^r_y, n^r_y + n^r_{-y})} = \\ &= \frac{\int_0^{1/2} q^{n^r_{-y}}(1-q)^{n^r_y}}{\int_0^1 q^{n^r_{-y}}(1-q)^{n^r_y}} = \frac{B(\frac{1}{2}; n^r_{-y}+1, n^r_y+1)}{B(n^r_{-y}+1, n^r_y+1)} \end{aligned} \tag{10}$$

Here $B(a, b)$ is the Beta function and $B(x; a, b)$ is an incomplete Beta function. Both of these functions can be calculated efficiently.

As for our previous approach, we require that non-support vectors have no influence on the likelihood function. This means that for all non-support vectors, $\Pr[q_i > 1/2] = 1$. In our experiments we have assumed that this is also true for all correctly classified points. Thus the only contributing points are the support vectors that have been misclassified.

There is still one free parameter r. For all these points we choose the fixed sphere radius yielding the lowest probability for $L(\omega|X, Y)$. This approach allows us to capture large connected regions of misclassified objects, which is a characteristic of underfitting. Of course, points from the other side of the hyperplane are not included in the sphere, as they are expected to have another value of q_i.

This kind of discrete likelihood function cannot be differentiated, but its expectation with respect to $\{X, Y\}$ for continuous $p(x, y)$ is also continuous. Therefore the determinant of the Fisher information matrix is non-zero and usage of the AIC/BIC formulas is legitimate. This method requires calculation of the distance between all pairs of points in the dataset. Using the kernel function, it may be computed with the following formula:

$$d(x_i, x_j) = \sqrt{K(x_1 - x_2, x_1 - x_2)} = \sqrt{K(x_1, x_1) - 2K(x_1, x_2) + K(x_2, x_2)}$$

requiring $O(N^2)$ operations.

When choosing the best radius r, it is enough to consider only the values of distance between points, because only at these values the likelihood function may change. In order to do this, we need to sort all distances between misclassified points and all other points ($\leq N_{SV}N$ values), what requires $O(N_{SV}N \log N)$ operations in total. Next, we iteratively take each radius and recalculate the probability $\Pr[q_i > 1/2]$ for the point x_i having this radius as a distance from x_i to one of the other points. It takes $O(N_{SV}N)$ operations, one for each radius. For large γ $N_{SV} \approx N$, so the total number of operations for this method is $O(N^2 \log N)$. In practice, it takes much less time, since it is enough to consider a fixed amount of different values of radius (e.g., 50) and choose the minimum value of likelihood function. For a relatively small dataset (≤ 1500 points) this algorithm runs faster than the SVM training procedure.

7. Experiments

This section provides the details of the experimental evaluation of two proposed approaches.

Work by Luxburg et al. (2004) described a model selection approach based on the compression quality for different values of γ and C. They showed how to compress the SVM classification results for all points. The main idea was to code the position of the hyperplane with the required accuracy and then add the information about misclassified points. The values of parameters where the compression rate is highest are then chosen. They made a comparison of their compression schemes with different error generalization bounds and showed that the best quality among all tested methods was achieved for their $C2$ and $C3$ compression codes, and span estimation of Leave-One-Out (LOO) cross-validation error, introduced in (Vapnik and Chapelle, 2000).

Our experiments followed the same design as described in (Luxburg et al., 2004). Training and test subsets were normalized according to the maximum and minimum values of the training set. Instead of fixed sizes $m = 100$ and $m = 500$ we performed computations for all values from 100 to 1500 (when the dataset was large enough), with step size 100. For each dataset, each training set size and each experiment we calculated the test error for the classifier learned with the parameters predicted by a given algorithm. An algorithm curve represents an average test error for 30 different experiments (splits on the training and test parts). Two plots for each dataset are presented: one for each likelihood function. We used the following values of parameters: $C \in [10^0, \dots, 10^5]$, $\gamma \in [10^{-3}, \dots, 10^3]$ and tested using twelve well-known datasets. Since the dataset "usdigits-0" had ten output labels, we considered the first digit as one class and all others as another class. All experiments were conducted using the LibSVM (2011) software.

Table 1: List of datasets

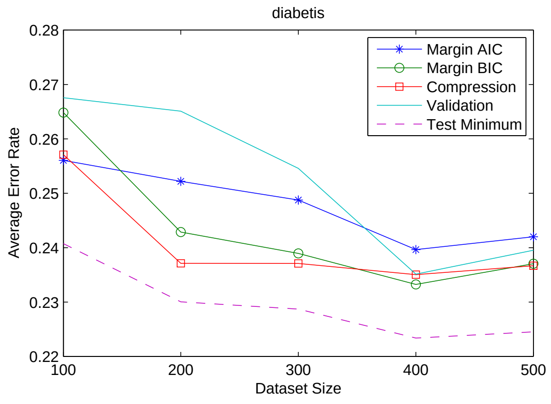
dataset	features	size	positive	negative
diabetis	8	768	268	500
german	20	1000	300	700
image	18	2086	1188	898
banana	2	5300	2376	2924
ringnorm	20	7400	3664	3736
twonorm	20	7400	3703	3697
splice	60	2991	1344	1647
waveform	21	5000	1647	3353
usdigits-0	256	11000	1100	9900
abalone	8	4177	2081	2096
titanic	6	2201	711	1490
thyroid	21	7200	534	6666

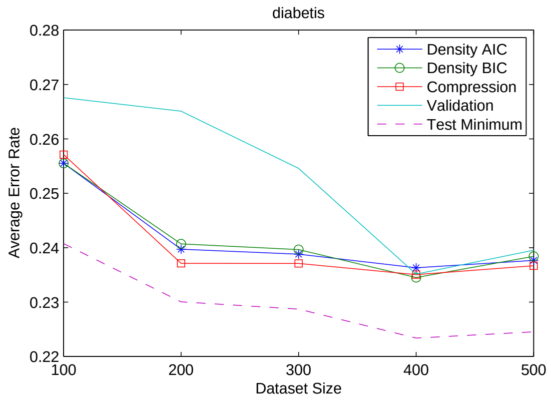
We compared our two approaches against the $C3$ compression code as a baseline, because it involves eigenvalues only for the restricted kernel matrix $K(x_i, x_j)$, where x_i, x_j are the support vectors, and therefore it runs relatively efficiently. However, it requires $O(N_{SV}^3) = O(N^3)$ for large γ, which can be prohibitive even for medium size datasets. Another computationally expensive step is the calculation of the radius of the smallest sphere around support vectors. This is a quadratic programming problem, but it may be approximated by the maximum distance to their centre. This C3 baseline technique is denoted as “Compression”.

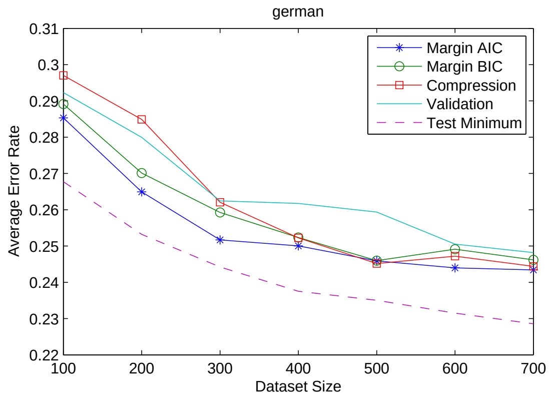
We did not compare against the span estimation method in our experiments, since its computational time was prohibitive. Instead, we compared against another baseline of simple “Hold out” validation: each training set was split into two subsets (75% and 25%), which were used as learning and validation sets, and the best pair of parameters γ and C was the pair that gave the minimum error on this small validation set. This simple method requires only one training procedure and therefore has similar running time to our approaches. This second baseline technique is denoted as “Validation”.

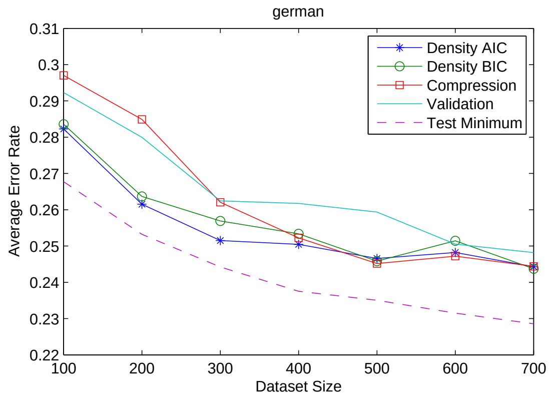
In the plots, the “Test Minimum” curve is an average of the 30 test error minimums - the best (and lowest) possible result. We present the results for variations of our methods corresponding to combinations of i) our two different likelihood functions (Margin and Density) and ii) the two different complexity penalty terms (AIC and BIC). Our four methods are thus termed Density-AIC, Density-BIC, Margin-AIC and Margin-BIC.

We also assessed the statistical significance of the differences in accuracy between each pair of algorithms. In particular, for each dataset and each algorithm, 90 different error values were selected: 30 for each of dataset sizes 1300, 1400 and 1500. The 90 values for the first algorithm were then compared against the 90 values for the second algorithm using a left-tail t-statistic and a p-value was obtained. For the smaller datasets, different dataset sizes were used: 300-500 for “diabetis” and 500-700 for “german”. The p-value was considered to be significant using a threshold of 0.05. Table 2 shows the results.









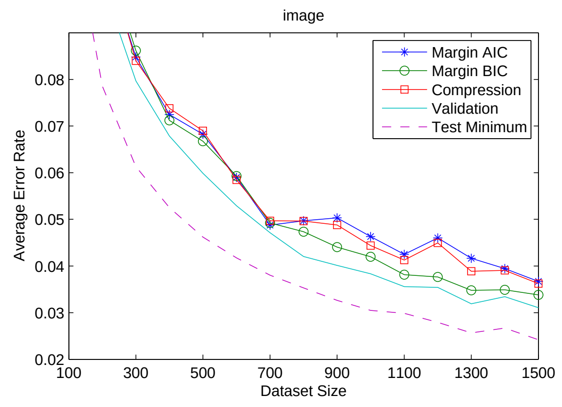


image
Margin AIC
Margin BIC
Compression
Validation
Test Minimum
Average Error Rate
Dataset Size
0.02
0.03
0.04
0.05
0.06
0.07
0.08
100
300
500
700
900
1100
1300
1500

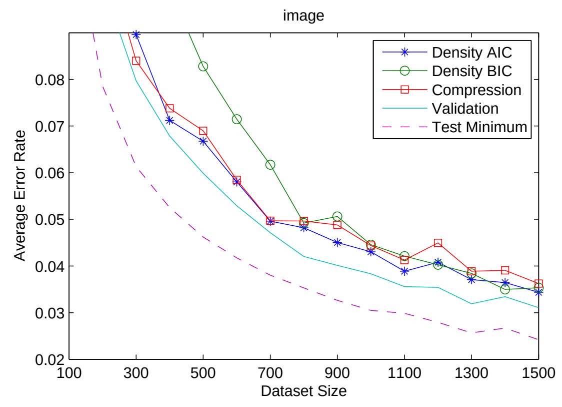
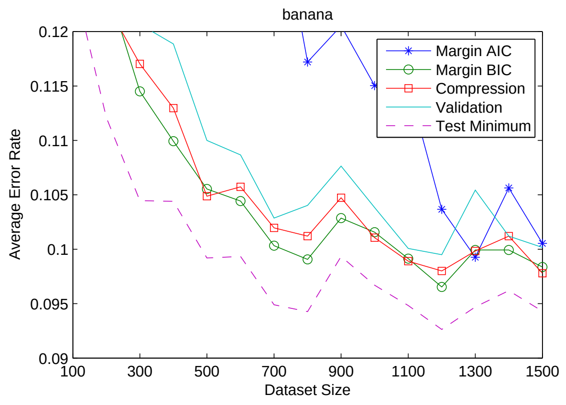
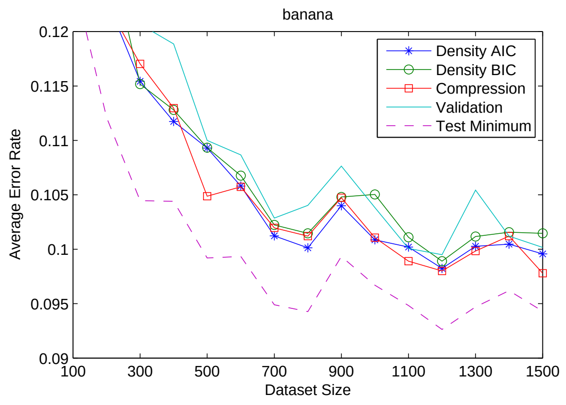
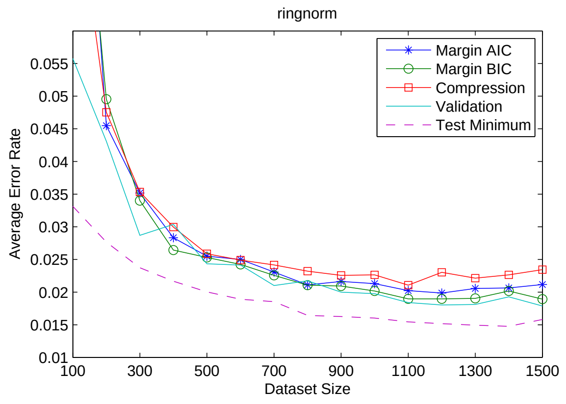
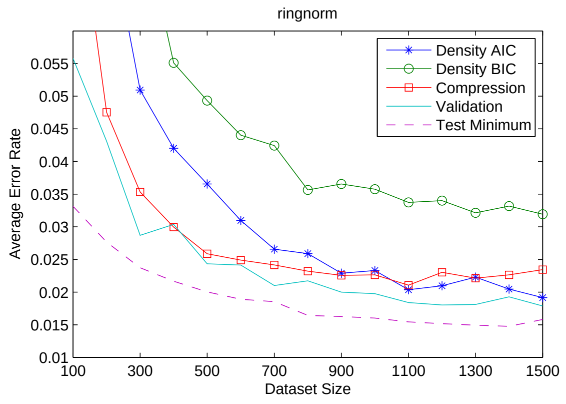


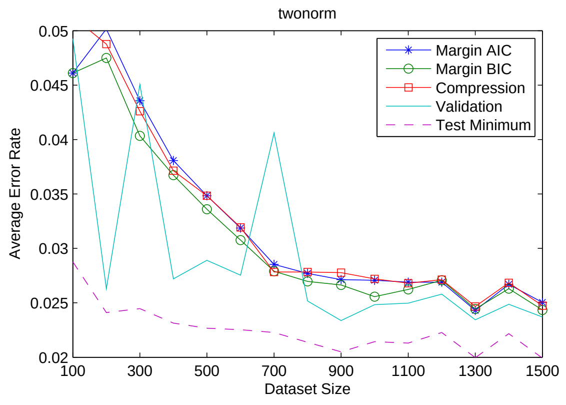
image
Density AIC
Density BIC
Compression
Validation
Test Minimum
Average Error Rate
Dataset Size
0.02
0.03
0.04
0.05
0.06
0.07
0.08
100
300
500
700
900
1100
1300
1500

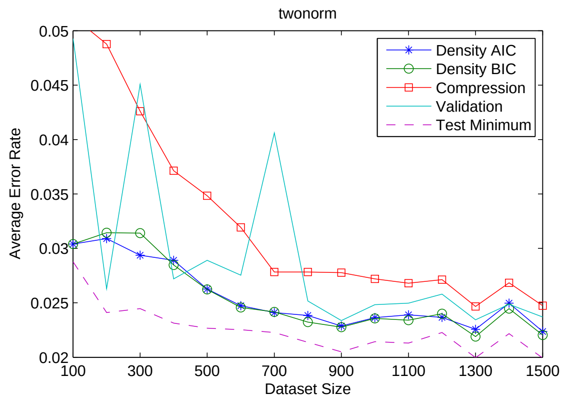

banana
Margin AIC
Margin BIC
Compression
Validation
Test Minimum
Average Error Rate
Dataset Size
0.09
0.095
0.1
0.105
0.11
0.115
0.12
100
300
500
700
900
1100
1300
1500

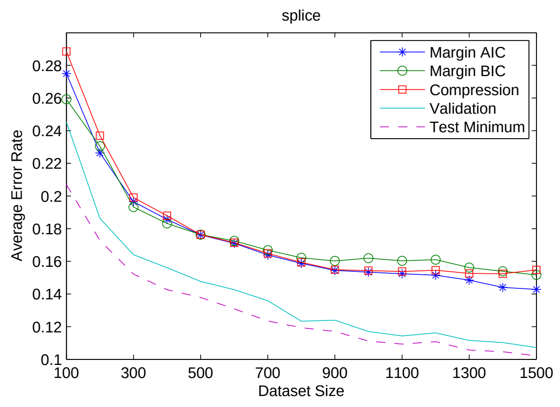

banana
Density AIC
Density BIC
Compression
Validation
Test Minimum
Average Error Rate
Dataset Size
0.09
0.095
0.1
0.105
0.11
0.115
0.12
100
300
500
700
900
1100
1300
1500

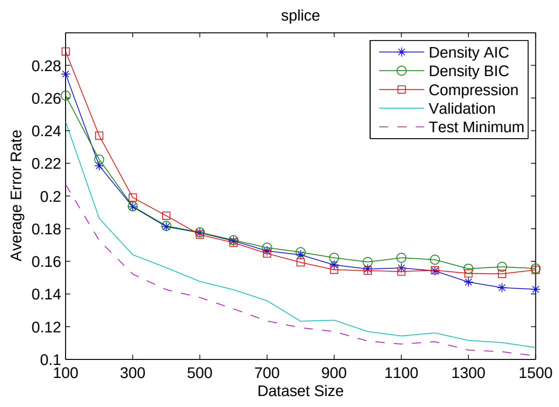

ringnorm
Margin AIC
Margin BIC
Compression
Validation
Test Minimum
Average Error Rate
Dataset Size
0.01
0.015
0.02
0.025
0.03
0.035
0.04
0.045
0.05
0.055
100
300
500
700
900
1100
1300
1500

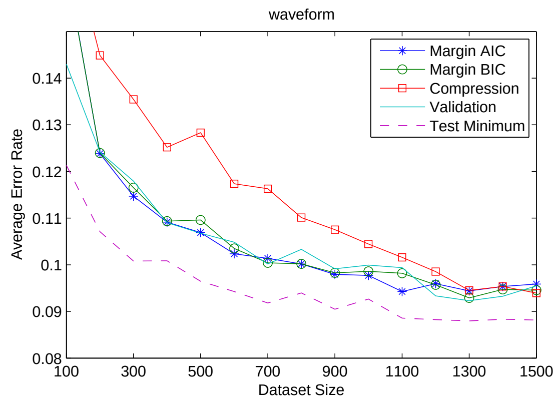

ringnorm
Density AIC
Density BIC
Compression
Validation
Test Minimum
Average Error Rate
Dataset Size
0.01
0.015
0.02
0.025
0.03
0.035
0.04
0.045
0.05
0.055
100
300
500
700
900
1100
1300
1500

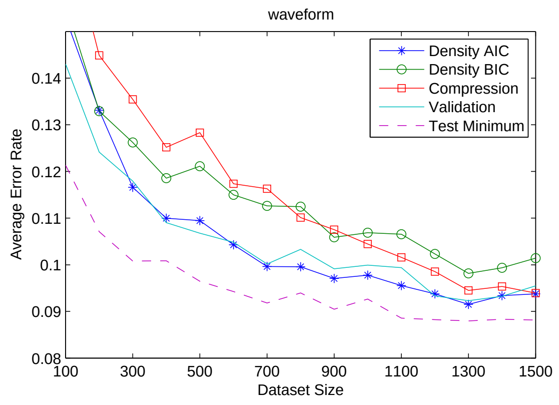

twonorm
Margin AIC
Margin BIC
Compression
Validation
Test Minimum
Average Error Rate
Dataset Size
0.02
0.025
0.03
0.035
0.04
0.045
0.05
100
300
500
700
900
1100
1300
1500

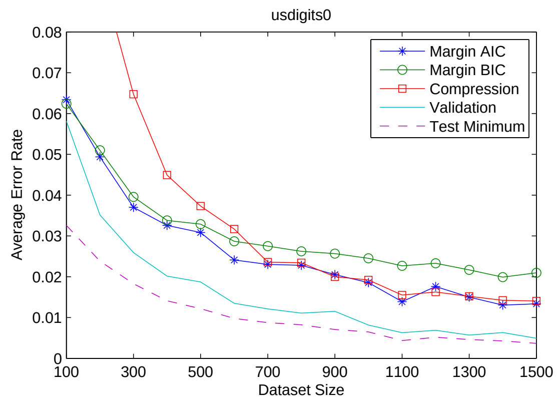

twonorm
Density AIC
Density BIC
Compression
Validation
Test Minimum
Average Error Rate
Dataset Size
0.02
0.025
0.03
0.035
0.04
0.045
0.05
100
300
500
700
900
1100
1300
1500

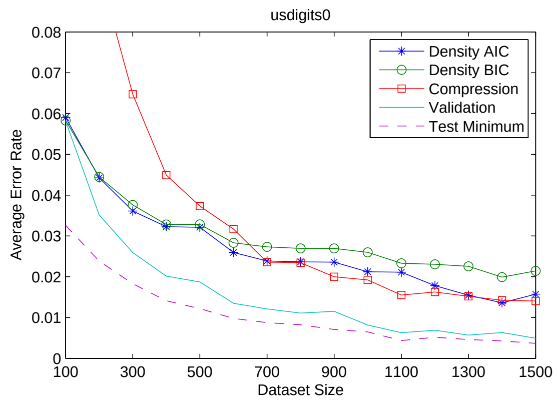

splice
Margin AIC
Margin BIC
Compression
Validation
Test Minimum
Average Error Rate
Dataset Size

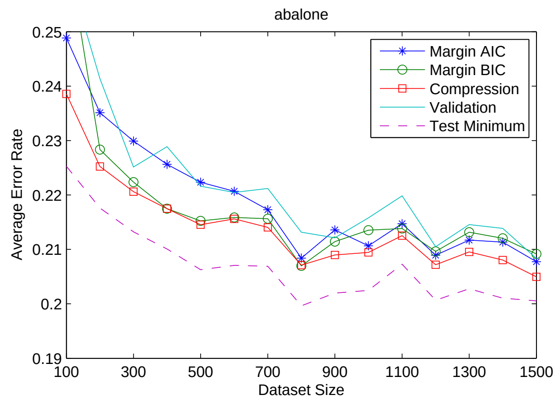

splice
Density AIC
Density BIC
Compression
Validation
Test Minimum
Average Error Rate
Dataset Size

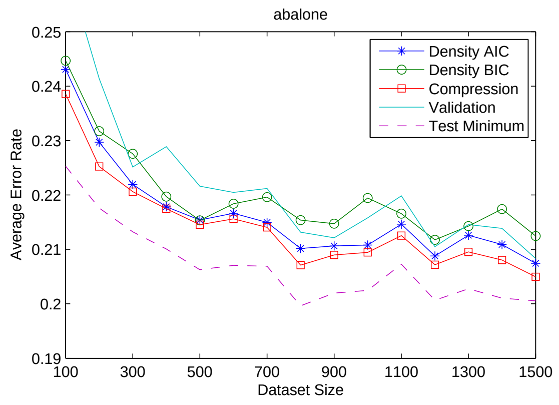

waveform
Margin AIC
Margin BIC
Compression
Validation
Test Minimum
Average Error Rate
Dataset Size

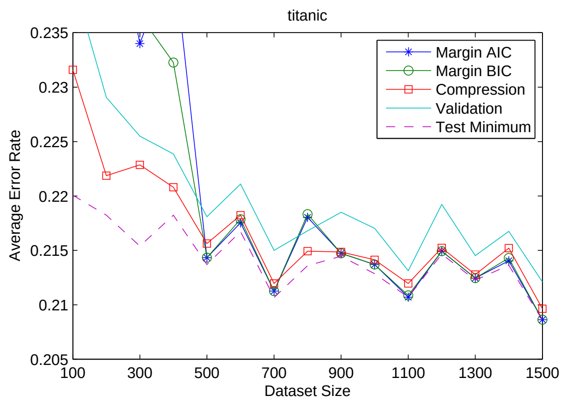

waveform
Density AIC
Density BIC
Compression
Validation
Test Minimum
Average Error Rate
Dataset Size

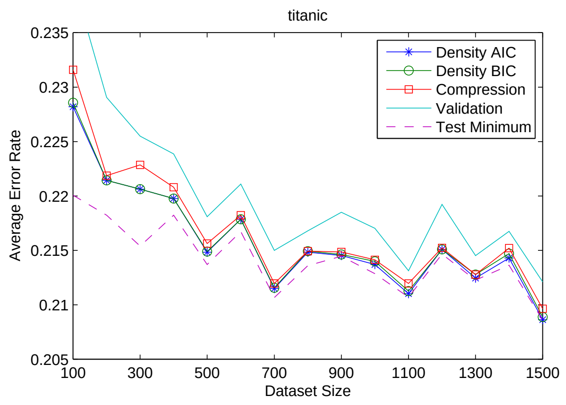

usdigits0
Margin AIC
Margin BIC
Compression
Validation
Test Minimum
Average Error Rate
Dataset Size

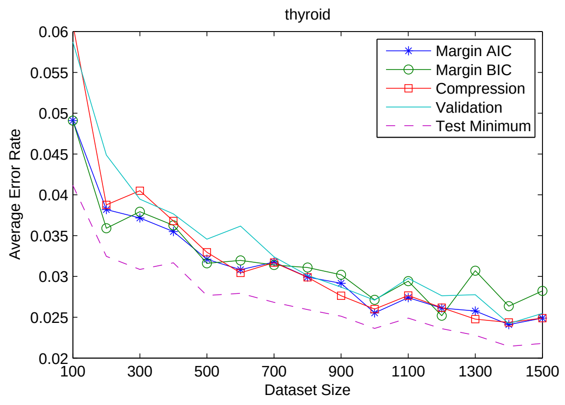

usdigits0
Density AIC
Density BIC
Compression
Validation
Test Minimum
Average Error Rate
Dataset Size


abalone
Margin AIC
Margin BIC
Compression
Validation
Test Minimum
Average Error Rate
Dataset Size


abalone
Density AIC
Density BIC
Compression
Validation
Test Minimum
Average Error Rate
Dataset Size







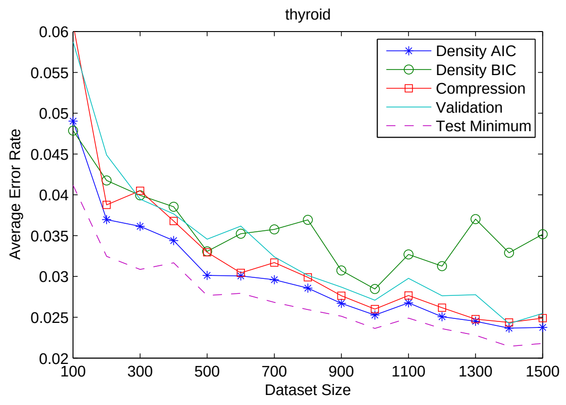


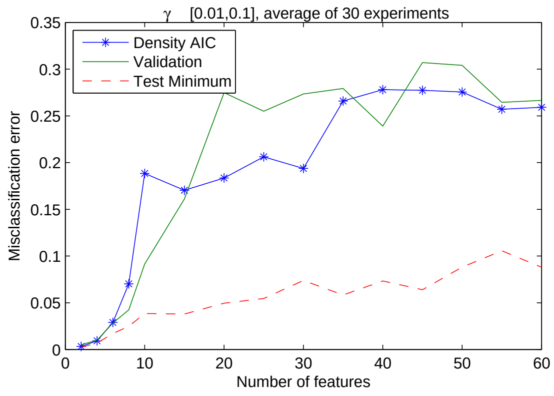
Table 2: Statistical significance results for pairs of algorithms. The value of cell (i, j) corresponds to the number of datasets where algorithm i is statistically significantly better than algorithm j.

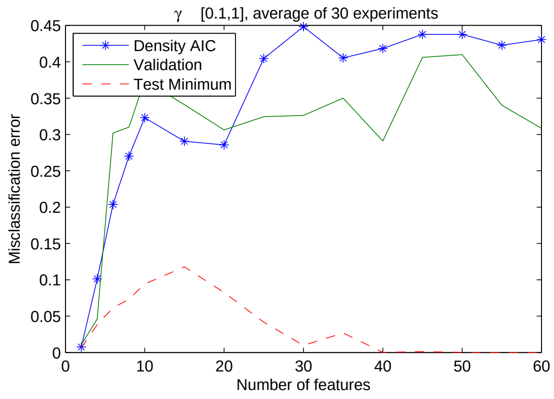
	MAIC	MBIC	DAIC	DBIC	Compression	Validation
MAIC	0	4	1	8	3	2
MBIC	4	0	3	8	4	4
DAIC	5	6	0	8	7	7
DBIC	3	1	1	0	2	4
Compression	4	3	1	7	0	6
Validation	6	6	4	7	5	0

8. Discussion of Results

We now discuss the plots from the previous section. One can see that all curves follow the "Test Minimum" shape almost everywhere, and therefore they appear to be reliable predictors for the best parameters. The single exception is the "Margin AIC" curve for the "banana" dataset, which shows adequate results only for dataset sizes ≥ 1200. In Table 2, we see that the "Density AIC" method achieved the best performance: it was consistently better than each of the other algorithms. Moreover, it appears to be the only one that is better than “Validation”. It is difficult to draw any clear conclusion from the table either about what type of penalty term is better (AIC or BIC) or about likelihood functions (Margin or Density).

While “Density AIC” clearly outperforms the other algorithms, there were two datasets where it did not converge to the “Validation” baseline (“splice” and “usdigits-0”). These datasets differ from others in two things: they have the largest number of features among all considered datasets and a very low rate of unique values for features. The influence of number of features is presented in the plots below.





At first we were required to construct a set of points with the known best value of the kernel parameter γ. In order to do this we employed the following procedure. In each experiment (30 for each number of features) we randomly generated 800 points and their labels in the cube $[-2, 2]^d$ and the kernel parameter γ from a particular range ([0.01,0.1] or [0.1,1]). After that we iteratively classified these points by the SVM classifier with this parameter γ and $C = 10$ and reassigned the labels until convergence. Finally, we generated the test set with 10000 points that was perfectly separated by this classifier. Then we started the algorithms for searching the best parameters as they were described earlier. In the end the error rate of each algorithm was averaged for each number of features.

We can see that on the left plot (true $\gamma \in [0.01, 0.1]$) the Density AIC algorithm performs well almost everywhere, but on the right plot (true $\gamma \in [0.1, 1]$) is starts to perform worse for $d \geq 25$. It is the dimensionality when the algorithm stops to recognize the right value of the parameter and predicts the same value as for the datasets from the left plot. Thus, we can claim that the higher dimensionality of the dataset with the fixed number of points, the lower the maximum value of γ that algorithm is still able to identify. Other experiment showed that the rate of unique values of features have no significant influence on the performance.

9. Summary

We described two new approaches for using AIC and BIC to estimate the best values of parameters γ and C to use in an SVM with RBF kernel. Our approaches are based on two likelihood functions, derived from different perspectives. The first operates using distances of points from the hyperplane, the second analyses the disposition of support vectors. Our two approaches were embodied in four different algorithms: Margin-AIC, Margin-BIC, Density-AIC and Density-BIC. Among these four algorithms, the best performing was

Density-AIC, which generally outperformed all others, including the more computationally expensive baseline methods "C3" compression code described in (Luxburg et al., 2004) and the simple "Validation" method.

References

P. Bartlett, 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.

H.S. Bhat, N. Kumar. On the derivation of the Bayesian Information Criterion, *http://nscs00.ucmerced.edu/~nkumar4/BhatKumarBIC.pdf*, 2008

K. P. Burnham, D. R. Anderson. Model Selection and Multimodel Inference. A Practical Information-Theoretic Approach. Second Edition, 2002.

O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee Choosing multiple parameters for support vector machines. *In Machine learning*, 46(1), pp. 131159, 2002.

S. Floyd and M. K. Warmuth. Sample compression, learnability, and the Vapnik-Chervonenkis dimension. *In Machine Learning*, 21(3), pages 269-304, 1995.

V. Franc, A. Zien, B. Schölkopf. Support Vector Machines as Probabilistic Models, *In Proceedings of the 28th International Conference of Machine Learning*, 2011

Y. Grandvalet, J. Mariethos S. and Bengio. Interpretation of SVMs with an application to unbalanced classification. *In Advances in Neural Information Processing Systems*, NIPS 18, 2005.

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.

U. von Luxburg, O. Bousquet, B. Schölkopf. A Compression Approach to Support Vector Model Selection. *In Journal of Machine Learning Research* 5, pages 293-323, 2004

J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *In Advances in Large Margin Classifiers.* MIT Press, 2000.

P. Sollich. Bayesian methods for support vector machines: Evidence and predictive class probabilities. *In Machine Learning*, 46(1), pages 21-52, 2002.

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

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

C.-C. Chang, C.-J. Lin. LIBSVM: A library for support vector machines. *In ACM Transactions on Intelligent Systems and Technology*, 2, pages 1-27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm