Bayesian approach to feature selection and parameter tuning for Support Vector Machine classifiers

Carl Gold
Computation & Neural Systems
California Institute of Technology
139-74, Pasadena, CA 91125
E-mail: carlg@caltech.edu

Alex Holub
Computation & Neural Systems
California Institute of Technology
139-74, Pasadena, CA 91125

Peter Sollich
Dept. of Mathematics
King's College London
Strand, London WC2R 2LS, U.K.

***Abstract*— A Bayesian point of view of SVM classifiers allows the definition of a quantity analogous to the evidence in probabilistic models. By maximizing this one can systematically tune hyperparameters and, via automatic relevance determination (ARD), select relevant input features. Evidence gradients are expressed as averages over the associated posterior and can be approximated using Hybrid Monte Carlo (HMC) sampling. We describe how a Nyström approximation of the Gram matrix can be used to speed up sampling times significantly while maintaining almost unchanged classification accuracy. In experiments on classification problems with a significant number of irrelevant features this approach to ARD can give a significant improvement in classification performance over more traditional, non-ARD, SVM systems. The final tuned hyperparameter values provide a useful criterion for pruning irrelevant features, and we define a measure of relevance with which to determine systematically how many features should be removed. This use of ARD for hard feature selection can improve classification accuracy in non-ARD SVMs. In the majority of cases, however, we find that in data sets constructed by human domain experts the performance of non-ARD SVMs is largely insensitive to the presence of some less relevant features. Eliminating such features via ARD then does not improve classification accuracy, but leads to impressive reductions in the number of features required, by up to 75%.**[1]

I. SVM CLASSIFICATION

In the usual way we assume a set D of n training examples (x_i, y_i) with binary outputs $y_i = \pm 1$. The SVM maps the inputs x to vectors $\boldsymbol{\phi}(x)$ in some high-dimensional feature space and uses a maximal margin hyperplane, $\mathbf{w} \cdot \boldsymbol{\phi}(x) + b = 0$, to separate the training examples. This is equivalent to minimizing $||\mathbf{w}||^2$ subject to the constraints $y_i(\mathbf{w} \cdot \boldsymbol{\phi}(x_i) + b) \geq 1\ \forall i$; see *e.g.* (Cristianini and Shawe-Taylor, 2000). The offset parameter b is treated as incorporated into $\mathbf{w}$ in the following, by augmenting feature space vectors to $\boldsymbol{\phi}(x) \rightarrow (\boldsymbol{\phi}(x), 1)$.

To avoid fitting noise in the training data, 'slack variables' $\xi_i \geq 0$ are introduced to relax the margin constraints to $y_i \mathbf{w} \cdot \boldsymbol{\phi}(x_i) \geq 1 - \xi_i\ \forall i$ and the term $(C/p)\sum_i \xi_i^p$ is then added to the objective function, with a penalty coefficient C and typically $p = 1$ or 2. This gives the SVM optimization problem: Find $\mathbf{w}$ to minimize

$$\frac{1}{2}||\mathbf{w}||^2 + C\sum_i l_p(y_i \mathbf{w} \cdot \boldsymbol{\phi}(x_i)) \tag{1}$$
$$l_p(z) = \frac{1}{p}(1-z)^p H(1-z)$$

where the Heaviside step function $H(1-z)$ ensures that the loss function $l_p(z)$ is zero for $z > 1$. For $p = 1$, $l_p(z)$ is called (shifted) hinge loss or soft margin loss.

For a practical solution, one uses Lagrange multipliers α_i conjugate to the constraints $y_i \mathbf{w} \cdot \boldsymbol{\phi}(x_i) \geq 1 - \xi_i$ and finds in the standard way that the optimal weight vector is $\mathbf{w}^* = \sum_i y_i \alpha_i \boldsymbol{\phi}(x_i)$; see *e.g.* (Cristianini and Shawe-Taylor, 2000). For the linear penalty case $p = 1$, the α_i are found from

$$\max_{0 \leq \alpha_i \leq C} \left(\sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j K_{ij} \right) \tag{2}$$

Here $K_{ij} = K(x_i, x_j)$ are the elements of the Gram matrix $\mathbf{K}$, obtained by evaluating the *kernel* $K(x, x') = \boldsymbol{\phi}(x) \cdot \boldsymbol{\phi}(x')$ for all pairs of training inputs. The corresponding optimal latent or decision function is $\theta^*(x) = \mathbf{w}^* \cdot \boldsymbol{\phi}(x) = \sum_i y_i \alpha_i K(x, x_i)$. Only the x_i with $\alpha_i > 0$ contribute to this sum; these are called support vectors (SVs). A similar result can be found for the quadratic penalty case (Cristianini and Shawe-Taylor, 2000). A common choice of kernel is the radial basis function (RBF) form $K(x, x') = \exp(-c|x - x'|^2)$, and we use a version of this augmented for Automatic Relevance Determination (ARD):

$$K(x, x') = k_0 \exp\left[-\sum_{a=1}^{d} \frac{(x^a - x'^a)^2}{2l_a^2} \right] + k_{\text{off}} \tag{3}$$

The l_a are length scales, one for each of the d input dimensions or features. One of the challenges in SVM classification is the *hyperparameter tuning problem*: find optimal – in terms of generalization performance – values for the (many, for large

[1] An abbreviated version of some portions of this article appeared in (Gold and Sollich, 2005), published under the IEEE copyright.

d) hyperparameters $\{l_a\}$, k_0, $k_{\rm off}$ and C, preferably without holding out parts of D as a test set. A related challenge common to all classification systems is the *feature selection problem*: determining an optimal set of attributes of the input for making the classification. Tuning individual length scales for each feature via ARD allows both problems to be addressed simultaneously, as described below.

II. Bayesian Interpretation of SVMs

In the probabilistic interpretation of SVM classification (see *e.g.* (Sollich, 2002), (Sollich, 2000) and references below), one regards (1) as a negative log-posterior probability for the parameters $\mathbf{w}$ of the SVM, and the conventional SVM classifier as the maximum a posteriori (MAP) solution of the corresponding probabilistic inference problem. The first term in (1) gives the prior $Q(\mathbf{w}) \propto \exp(-\frac{1}{2}||\mathbf{w}||^2)$. This is a Gaussian prior on $\mathbf{w}$; the components of $\mathbf{w}$ are uncorrelated with each other and have unit variance. Because only the latent function values $\theta(x) = \mathbf{w} \cdot \boldsymbol{\phi}(x)$—rather than $\mathbf{w}$ itself—appear in the second, data dependent term of (1), it makes sense to express the prior directly as a distribution over these. The $\theta(x)$ have a joint Gaussian distribution because the components of $\mathbf{w}$ do, with covariances given by $\langle \theta(x)\theta(x')\rangle = \langle (\boldsymbol{\phi}(x)\cdot\mathbf{w})(\mathbf{w}\cdot\boldsymbol{\phi}(x'))\rangle = K(x,x')$. The SVM prior is therefore a *Gaussian process* (GP) over the functions θ, with zero mean and with the kernel $K(x,x')$ as covariance function (Seeger, 2000), (Opper and Winther, 2000a), (Opper and Winther, 2000b). The second term in (1) similarly becomes a (negative) log-likelihood if we define the probability of obtaining output y for a given x (and θ) as

$$Q(y=\pm 1|x,\theta) = \kappa(C)\exp[-Cl_p(y\theta(x))] \qquad (4)$$

The constant factor $\kappa(C)$ is a normalization constant. The overall normalization of the probability model is somewhat subtle, and fully discussed in (Sollich, 2002); in line with other work on probabilistic interpretations of SVM classifiers (Seeger, 2000), (Opper and Winther, 2000b), (Kwok, 2000) we disregard this issue here.

A. Evidence gradients

From the probabilistic point of view it becomes natural to tune hyperparameters to maximize the posterior likelihood of the data, or *evidence*, $Q(Y|X) = \int d\theta\, Q(Y|X,\theta)Q(\theta)$; the integration is over the latent function values $\theta(x)$ at all different input points x and X and Y are the training input and output sets respectively. Values of $\theta(x)$ at non-training inputs can be integrated out trivially, so that

$$\begin{aligned} Q(Y|X) &= \int d\boldsymbol{\theta}\, Q(Y|X,\boldsymbol{\theta})Q(\boldsymbol{\theta}) && (5)\\ &= \int d\boldsymbol{\theta} \prod_i Q(y_i|x_i,\theta)Q(\boldsymbol{\theta}) && (6)\end{aligned}$$

with $\boldsymbol{\theta} = (\theta_1 \ldots \theta_n)$. Because $Q(\theta)$ is a zero mean Gaussian process, the marginal distribution $Q(\boldsymbol{\theta})$ is a zero mean Gaussian with covariance matrix $\mathbf{K}$. The evidence is therefore

$$\begin{aligned} Q(Y|X) = \quad & |2\pi\mathbf{K}|^{-1/2}\kappa^n(C) \times && (7)\\ & \times \int d\boldsymbol{\theta}\, \exp\left[-\frac{1}{2}\boldsymbol{\theta}^{\rm T}\mathbf{K}^{-1}\boldsymbol{\theta} - \sum_i Cl_p(y_i\theta_i)\right] \end{aligned}$$

It is difficult to obtain accurate numerical estimates of the evidence itself, but one can estimate its gradients with respect to the hyperparameters and use these in a gradient ascent algorithm, without ever calculating the actual value of the evidence. Starting from (7) one finds for the derivative of the normalized log-evidence $E(Y|X) = n^{-1}\ln Q(Y|X)$ w.r.t. the penalty parameter C (Gold and Sollich, 2003)

$$\frac{\partial}{\partial C}E(Y|X) = \frac{\partial \ln\kappa(C)}{\partial C} - \left\langle \frac{1}{n}\sum_i l_p(y_i\theta_i)\right\rangle \qquad (8)$$

where the average is, as expected on general grounds, over the posterior $Q(\boldsymbol{\theta}|D) \propto Q(Y|X,\boldsymbol{\theta})Q(\boldsymbol{\theta})$. A little more algebra yields the derivative w.r.t. any parameter λ appearing in the kernel (Gold and Sollich, 2003)

$$\frac{\partial}{\partial\lambda}E(Y|X) = -\frac{C}{2n}\left\langle {l'}_p^{\rm T}(\mathbf{Y}\boldsymbol{\theta})\mathbf{Y}\frac{\partial\mathbf{K}}{\partial\lambda}\mathbf{K}^{-1}\boldsymbol{\theta}\right\rangle \qquad (9)$$

where $\mathbf{Y} = {\rm diag}(y_1\ldots y_n)$ while $l'_p(\mathbf{Y}\boldsymbol{\theta})$ is understood component-wise, i.e. ${l'}_p^{\rm T}(\mathbf{Y}\boldsymbol{\theta}) = (l'_p(y_1\theta_1)\ldots l'_p(y_n\theta_n))$; also $l'_p(z) \equiv dl_p/dz$.

The posterior averages required in the expressions (8,9) are still not analytically tractable; we therefore use Hybrid Monte Carlo (HMC, see *e.g.* (Neal, 1993)) to estimate them numerically. The HMC algorithm simulates a stochastic dynamics with a Hamiltonian "energy" defined by the target distribution plus a "momentum", or kinetic energy term. Denoting the momentum variables $\mathbf{p}$, a suitable Hamiltonian for our case is

$$\begin{aligned} \mathcal{H}(\boldsymbol{\theta},\mathbf{p}) &= \frac{1}{2}\mathbf{p}^{\rm T}\mathbf{M}\mathbf{p} + \frac{1}{2}\boldsymbol{\theta}^{\rm T}\mathbf{K}^{-1}\boldsymbol{\theta} + V(\boldsymbol{\theta}) && (10)\\ V(\boldsymbol{\theta}) &= C\sum_i l_p(y_i\theta_i) && (11)\end{aligned}$$

where $\mathbf{M}$ is the inverse covariance matrix of the momenta, chosen later to simplify the computation. The corresponding stationary "Boltzmann" distribution $P(\boldsymbol{\theta},\mathbf{p}) \propto \exp[-\mathcal{H}(\boldsymbol{\theta},\mathbf{p})] \propto \exp(-\frac{1}{2}\mathbf{p}^{\rm T}\mathbf{M}\mathbf{p})Q(\boldsymbol{\theta}|D)$ factorizes over $\boldsymbol{\theta}$ and $\mathbf{p}$, so that samples from $Q(\boldsymbol{\theta}|D)$ can be obtained by sampling from $P(\boldsymbol{\theta},\mathbf{p})$ and discarding the momenta $\mathbf{p}$. The only role of the $\mathbf{p}$ is to help ensure a representative sampling of the posterior. An update step in the HMC algorithm consists of: 1) updating a randomly chosen momentum variable p_i by Gibbs sampling according to the Gaussian distribution $\exp(-\frac{1}{2}\mathbf{p}^{\rm T}\mathbf{M}\mathbf{p})$; 2) changing both $\boldsymbol{\theta}$ and $\mathbf{p}$ by moving along a Hamiltonian trajectory for some specified "time" τ; the trajectory is determined by solving an appropriately discretized version of the differential equations

$$\begin{aligned} \frac{d\theta_i}{d\tau} &= \frac{\partial\mathcal{H}}{\partial p_i} = (\mathbf{M}\mathbf{p})_i && (12)\\ \frac{dp_i}{d\tau} &= -\frac{\partial\mathcal{H}}{\partial\theta_i} = -(\mathbf{K}^{-1}\boldsymbol{\theta})_i - Cy_i l'_p(y_i\theta_i) && (13)\end{aligned}$$

For an exact solution of these equations, $\mathcal{H}$ would remain constant; due to the discretization, small changes in $\mathcal{H}$ are possible and one accepts the update of $\boldsymbol{\theta}$ and $\mathbf{p}$ from the beginning to the end of the trajectory with the usual Metropolis acceptance rule.

The occurrence of the $O(n^3)$ matrix inversion $\mathbf{K}^{-1}$ in (13) may seem problematic, but can be avoided with an appropriate choice of $\mathbf{M}$ (Gold and Sollich, 2003). The multiplications of $n \times n$ matrices by n-dimensional vectors, requiring $O(n^2)$ operation, are a more serious problem because the HMC sampling process gives a large prefactor: averages over the posterior distribution are taken by sampling after each trajectory step, and repeating the procedure over some large number of steps. In practice the first half of the steps are discarded to allow for equilibration and we chose a total of 40,000 samples, giving 20,000 "production samples". The discretization of equations (13) into time intervals $\Delta\tau$ also means that all operations must be performed l times to approximate a trajectory of length $l\Delta\tau$. With our chosen values $\Delta\tau = 0.05$, $l = 10$, estimating the gradients by HMC then requires 400,000 matrix multiplications of $O(n^2)$. Results in (Gold and Sollich, 2003) showed that while tuning SVM hyperparameters using the evidence gradients calculated in this way gives impressive generalization performance, it requires an amount of computing time that would be impractical under most circumstances.

III. The Nyström Approximation to the Gram Matrix

As explained in e.g. (Williams and Seeger, 2001), the *Nyström method* is very useful for speeding up many kernel algorithms. We apply it here to the Gram matrix. This is first approximated by a truncated eigendecomposition as used in *e.g.* principal component analysis,

$$\mathbf{K} \approx \sum_{i=1}^{p} \lambda_i \mathbf{u}_i \mathbf{u}_i^{\mathrm{T}} \tag{14}$$

Here $\lambda_1 > \ldots > \lambda_p$ are the largest $p < n$ eigenvalues of $\mathbf{K}$ and $\mathbf{u}_i$ the corresponding eigenvectors. Implicitly, the truncation to $p < n$ assumes that the spectrum of $\mathbf{K}$ is dominated by a small fraction of large eigenvalues, with a negligible tail of smaller eigenvalues. This assumption is reasonable because for RBF kernels (3) the kernel *function* has a quickly decaying spectrum of this type; for large n, the leading eigenvalues of the Gram matrix should then exhibit the same fast decay.

The approximation (14) is still costly because it involves determination of many of the eigenvalues and associated eigenvectors of $\mathbf{K}$. The idea of the Nyström method is to avoid this by *estimating* the λ_i and u_i from a random subsample of size m of the full training set, *i.e.* from a submatrix $\mathbf{K}_{m,m}$. If the latter has eigenvalues $\lambda_i^{(m)}$ and eigenvectors $\mathbf{u}_i^{(m)}$, the Nyström approximation to (14) is (Williams and Seeger, 2001)

$$\begin{aligned} \tilde{\mathbf{K}} &= \sum_{i=1}^{p} \tilde{\lambda}_i \tilde{\mathbf{u}}_i \tilde{\mathbf{u}}_i^{\mathrm{T}} && (15) \\ \tilde{\lambda}_i &= \frac{n}{m} \lambda_i^{(m)} && (16) \\ \tilde{\mathbf{u}}_i &= \sqrt{\frac{m}{n}} \mathbf{K}_{n,m} \mathbf{u}_i^{(m)} && (17) \end{aligned}$$

where $\mathbf{K}_{n,m}$ is the $n \times m$ submatrix of $\mathbf{K}$ containing the columns from the selected subsample. The factor n/m relating the eigenvalues of the full Gram matrix and the subsample $\mathbf{K}_{m,m}$ is easy to understand for the case of an RBF kernel, where the diagonal entries of $\mathbf{K}$ are all equal to $k_0 + k_{\mathrm{off}}$; thus the trace of $\mathbf{K}$ is bigger by a factor n/m than that of $\mathbf{K}_{m,m}$ and the same proportionality must hold for the eigenvalues, if indeed the $\lambda_i^{(m)}$ are good estimators for the λ_i. The advantage of the Nyström approach is that its run time is only $O(m^2 n)$ rather than the usual $O(n^3)$ required for a full eigensolution.

Because the Nyström approximation estimates only m eigenvalues, at most m terms can be used in the expansion (14) of $\mathbf{K}$. However, if the $\lambda_i^{(m)}$ decay sufficiently quickly, then one can often get away with having p not just equal to m but in fact substantially smaller. Experiments in (Williams and Seeger, 2001) showed that for *e.g.* Gaussian process classifiers values of m (and thus p) significantly smaller than n can often be used without impairing performance. We therefore next describe how to adapt the Nyström approach to our aim of SVM hyperparameter tuning by evidence gradient ascent.

A. Application to evidence gradient estimation by HMC

To apply the Nyström method to our HMC sampling, we approximate $\boldsymbol{\theta}$ as

$$\boldsymbol{\theta} = \mathbf{V}\mathbf{b} \tag{18}$$

where $\mathbf{V}$ is an $n \times p$ matrix with columns $\tilde{\mathbf{u}}_i \tilde{\lambda}_i^{1/2}$, and use for the HMC the Hamiltonian

$$\mathcal{H}(\mathbf{b}, \mathbf{p}) = \frac{1}{2}\mathbf{p}^{\mathrm{T}}\mathbf{p} + \frac{1}{2}\mathbf{b}^{\mathrm{T}}\mathbf{b} + C \sum_i l_p((\mathbf{Y}\mathbf{V}\mathbf{b})_i) \tag{19}$$

The term $\frac{1}{2}\mathbf{b}^{\mathrm{T}}\mathbf{b}$ corresponds to a zero-mean Gaussian prior on $\mathbf{b}$ with $\langle \mathbf{b}\mathbf{b}^{\mathrm{T}} \rangle = \mathbf{1}$. The choice of the matrix $\mathbf{V}$ linking $\mathbf{b}$ to $\boldsymbol{\theta}$ was made precisely such that this simple prior, with i.i.d. unit Gaussian components of the vector $\mathbf{b}$, gives us the desired $\langle \boldsymbol{\theta}\boldsymbol{\theta}^{\mathrm{T}} \rangle = \mathbf{V}\mathbf{V}^{\mathrm{T}} = \tilde{\mathbf{K}}$. The primary variables for the simulation are now $\mathbf{p}$ and $\mathbf{b}$, both only p-dimensional vectors. The vector $\mathbf{b}$ is initialized using the pseudo-inverse $\mathbf{b} = (\mathbf{V}^{\mathrm{T}}\mathbf{V})^{-1}\mathbf{V}^{\mathrm{T}}\boldsymbol{\theta}^*$, with $\boldsymbol{\theta}^*$ from the conventional (MAP) SVM solution; this ensures that $\mathbf{V}\mathbf{b}$ is the closest approximation to $\boldsymbol{\theta}^*$ within the subspace accessible within our approximation (i.e. the space spanned by the columns of the matrix $\mathbf{V}$). When needed for evaluating gradients, $\boldsymbol{\theta}$ is calculated from $\mathbf{b}$ according to (18). This relation was already used in writing down the last term of $\mathcal{H}$ in (19). The HMC trajectories for $\mathbf{b}$ and $\mathbf{p}$ are again determined by solving Hamilton's equations, i.e.

$$\frac{db_i}{d\tau} = \frac{\partial \mathcal{H}}{\partial p_i} = p_i \quad (20)$$

$$\frac{dp_i}{d\tau} = -\frac{\partial \mathcal{H}}{\partial b_i} = -b_i + C\left(\mathbf{V}^{\mathrm{T}}\mathbf{Y}l'(\mathbf{Y}\mathbf{V}\mathbf{b})\right)_i \quad (21)$$

For the average (9) we also need $\tilde{\mathbf{K}}^{-1}\boldsymbol{\theta} = \tilde{\mathbf{K}}^{-1}\mathbf{V}\mathbf{b}$. This looks awkward because $\tilde{\mathbf{K}}$ does not have full rank. However, $\tilde{\mathbf{K}}^{-1}\mathbf{V}$ remains well-defined and can be expressed as the pseudo-inverse $\tilde{\mathbf{K}}^{-1}\mathbf{V} = \mathbf{V}(\mathbf{V}^{\mathrm{T}}\mathbf{V})^{-1}$. This can be seen by first regularizing $\tilde{\mathbf{K}}$ with a small "jitter", $\tilde{\mathbf{K}} \to \tilde{\mathbf{K}} + \sigma\mathbf{1}$. For the regularized inverse one has the result $(\tilde{\mathbf{K}} + \sigma\mathbf{1})^{-1} = \sigma^{-1}(\mathbf{1} - \mathbf{V}(\sigma + \mathbf{V}^{\mathrm{T}}\mathbf{V})^{-1}\mathbf{V}^{\mathrm{T}})$ from (Williams and Seeger, 2001). One can then find $(\tilde{\mathbf{K}}+\sigma\mathbf{1})^{-1}\mathbf{V}$ by right-multiplication with $\mathbf{V}$ and finally take $\sigma \to 0$ to get the desired result. To further speed up the numerics one notes that in (9) only the product $\mathbf{Y}(\partial\mathbf{K}/\partial\lambda)\tilde{\mathbf{K}}^{-1}\mathbf{V}\mathbf{b}$ is needed to calculate the gradients. We therefore pre-calculate the entire $n \times p$ matrix $\mathbf{Y}(\partial\mathbf{K}/\partial\lambda)\mathbf{V}(\mathbf{V}^{\mathrm{T}}\mathbf{V})^{-1}$ at the beginning of each HMC simulation, which requires $O(n^2p + p^3)$ operations for each hyperparameter λ.

Equation (20) tells us that at each discretized trajectory step in the Nyström HMC we must perform $O(np)$ multiplications to calculate the update of the momenta. The update (18) of $\boldsymbol{\theta}$ at the end of each trajectory is of the same complexity, as is the calculation of the quantity to be averaged in (9), which is needed for each production sample. Overall, the Nyström approximation to the kernel thus reduces the computational complexity of the main loop of the HMC sampling procedure from effectively $O(n^2)$ to $O(np)$. Although we focused on the Nyström method, it is likely that similar results could be achieved using other approaches to approximate the kernel matrix, such as those described in (Fine and Scheinberg, 2001).

IV. Feature Selection

Alternative methods for tuning the hyperparameters of an SVM were reviewed in (Gold and Sollich, 2005) and here we turn our attention to the problem of feature selection. Feature selection is used in classification problems in order to improve generalization accuracy, assist in the interpretation of the problem domain, and speed up computation times. Various specific definitions of feature selection have been proposed; see $e.g.$ (Weston et al., 2000). Generally speaking, if the original data set D contains feature vectors $x_i \in R^d$ then we would like to find some subset of the features, $x'_i \in R^{d'}$, $d' < d$, to accomplish one or more of the above goals. In the context of SVMs it is worth noting that the dimensionality of the input feature space has a relatively benign impact on the speed of computation and thus our focus is primarily on improving generalization performance and subsequently on translating that improvement into greater understanding of the problem domain.

Feature selection methods are often classified into wrappers and filters, the difference being in whether or not the method uses the output of the classifier in order to select the features. Wrapper methods usually work by evaluating the classifier on subsets of the feature space, using some sort of greedy algorithm to organize the search of the large (i.e. 2^d) number of possible feature combinations; see $e.g.$ (Kohavi and John, 1997). Filter methods, on the other hand, generally use unsupervised methods to select features.

Feature selection for SVMs was previously proposed in (Weston et al., 2000) and (Fröhlich and Zell, 2004). In (Weston et al., 2000) feature selection is performed using the radius-margin bound on the leave-one-out error. If M is the margin achieved on the training set and R the radius of the smallest sphere (in the kernel-induced feature space, not to be confused with the original input feature space) containing all training inputs, this bound is of the form $R^2/(M^2n)$. Which input features are picked is encoded in a binary valued vector $\sigma \in \{0,1\}^n$, so that $x \mapsto (x * \sigma) = x'$ where $*$ indicates elementwise multiplication. Feature selection is performed by gradient descent on the radius-margin bound with respect to a real-valued approximation of σ. In (Fröhlich and Zell, 2004) feature selection is performed using recursive feature elimination based on the regularized risk. The latter augments the error estimate produced by the test set with the norm of the margin and is an upper bound on the generalization error. The recursive feature selection is a greedy search algorithm which starts with all features and removes them one at a time. The regularized risk is used to guide the search by assuming the training error does not change with the removal of each feature, calculating the approximate change in the regularized risk as the change in margin, and removing those features for which the change in the margin is smallest.

In the ARD RBF kernel (3) the contribution of each input feature to the kernel function is divided by a separate length scale, l_a: the larger the length scale, the smaller the contribution which that feature will make to the kernel function. For this reason the length scales produced by the hyperparameter tuning algorithm can be used for feature selection simply by eliminating those features with the largest length scales. While feature selection via ARD is technically a wrapper approach because the output of the classifier on the training set is used in the hyperparameter tuning algorithm, it is distinct from traditional wrapper approaches because it avoids searching the space of feature combinations and instead proceeds directly to an appropriate feature set using principles designed to improve generalization performance.

Fetaure dectection using ARD was originally proposed for backprop and RBF networks; see $e.g.$ (MacKay, 1998). Performing feature selection by tuning the length scales in an ARD RBF kernel of an SVM has been suggested previously in (Chapelle et al., 2002), (Chu et al., 2003), (Gold and Sollich, 2005), and in (Gestel et al., 2002) ARD is performed using a Bayesian framework for Least Square SVM's, also using the Nyström approximation.

V. Experiments and results

In (Gold and Sollich, 2005) we tested tuning the hyperparameters with the Nyström approximation in the HMC calculation of the evidence gradients, and compared to results for the full HMC calculation presented in (Gold and Sollich, 2003). We found that the Nyström method resulted in a significant reduction of computation time at negligible cost in performance. While the quality of the results depended weakly on m, the number of samples for the Nyström approximation, and p, the number of eigenvalues used to construct the approximate Gram matrix $\tilde{\mathbf{K}}$, the requirements were modest ($m \sim 200$, $p \sim 20$) and crucially did not scale with the number n of training samples. Thus the use of the Nyström approximation improved the run time of the HMC simulation from $O(n^2)$ (with a large prefactor) to effectively $O(n)$.

We tested the generalization performance resulting from SVM hyperparameter tuning with the gradients of the Bayesian evidence on 13 benchmark data sets from the UCI Machine Learning Repository[2] (Gold and Sollich, 2005). The results were compared to an ARD SVM tuned using bounds on the generalization error (Chapelle et al., 2002) and to the Trigonometric Bayesian SVM (Chu et al., 2003), which also incorporates ARD. To assess the effects of ARD we compared further with Adaboost and a non-ARD SVM. The latter has the kernel (3) with $k_0 = 1$, $k_{\text{off}} = 0$ and all lengthscales l_a equal, and the two remaining parameters C and l are chosen by cross-validation (Rätsch et al., 2001). The experimental setup was to tune hyperparameters on 5 random subsets of the data, set each hyperparameter to the median value of the 5 trials and then assess the resulting classification performance on either 15 or 95 random divisions of the data into training and test sets, depending on the number of samples available.

We found that for most of the data sets the performance of the different systems was quite similar, with the best average performance usually within error bars of the worst. The one exception to this was the Splice data set, where the two Bayesian ARD SVMs significantly outperformed the other, non-ARD methods tested. The reason for this improvement in performance seems to be that the ARD SVMs correctly determine that the majority of the input features are not relevant to the classification task, while the presence of a large number of irrelevant features negatively affects the performance of the non-ARD systems. The process of tuning the length scales, l_a, using the gradients of the Bayesian evidence is illustrated in figure (1, top). The length scales are initialized to the square root of the number of input features because the sum in the exponent in equation (3) would otherwise grow in proportion to the number of features; note that the input features are all individually normalized to have zero mean and unit variance. The average of the magnitude of length scale gradients normalized by their peak amplitude is shown in figure (1, bottom). The gradients are largest at the beginning of the gradient ascent and decline quickly in the first few steps. Thereafter the gradient magnitudes continue to decrease in a noisy and non-monotonic fashion until their normalized average reaches a pre-defined stopping criterion, as described in (Gold and Sollich, 2005).

[2]http://www.ics.uci.edu/∼mlearn/MLRepository.html

The final length scales tuned by the Bayesian ARD SVM are shown in Figure (2). The top graph makes it readily apparent that the most relevant features are clustered around the middle of the input feature vector. A similar result was found in (Chu et al., 2003). In the Splice data set the task is to detect the presence of an exon/intron boundary in a DNA sequence and the features are a raw numeric representation of the nucleotides (i.e. $\{A, C, G, T\} \mapsto \{1, 2, 3, 4\}$). The sequences are aligned so that a boundary, if present, occurs in the middle of the sequence. The length scales tuned by the Bayesian ARD SVM then suggest that the nucleotides within approximately 5 positions of a hypothetical boundary are most significant to discriminating the presence or absence of such a boundary.

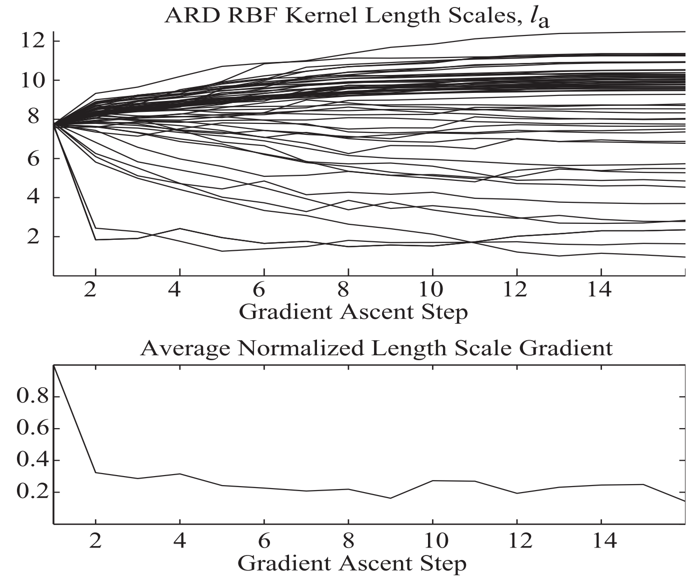

Fig. 1. Gradient ascent on Bayesian evidence for ARD RBF kernel length scales

Having reached these conclusions on the Splice data set we further tested the Bayesian ARD SVM on some of the other binary classification problems from the UCI repository, more particularly those with the highest numbers of features. We compared the performance of the ARD SVM tuned with gradients of the Bayesian evidence with a non-ARD SVM with hyperparameters tuned by tenfold cross-validation (CV SVM). The cross-validation employed a grid search in the logarithm of the noise parameter, C, and of the single length scale l. The data sets considered have been used in a variety of studies; as our interest was in comparing the Bayesian ARD SVM and the non-ARD CV SVM we did not attempt to replicate precisely previous experimental setups. Instead we adopted as a uniform procedure for all data sets 20 runs on random splits of the available data into 2/3 for training and 1/3 for testing. (For the larger Spam data set only 10 runs were used with 1/4 for training and 3/4 for testing.) No attempt was made to balance the number of examples from each category in either the training or the test data. Hyperparameters were re-tuned for every split of the data. The resulting average test errors

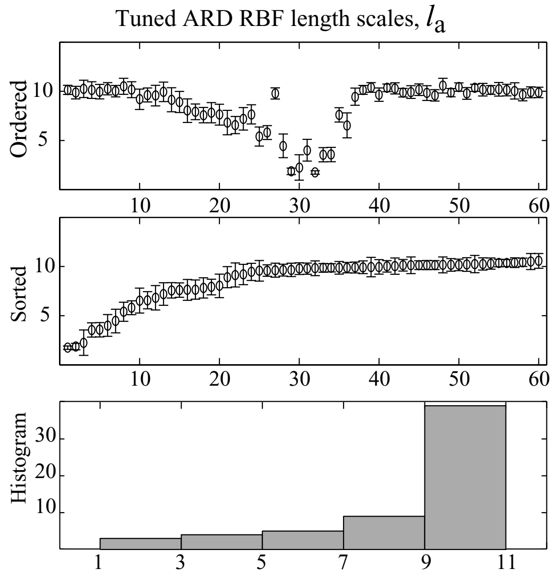

Fig. 2. Splice, final length scales as tuned by ARD

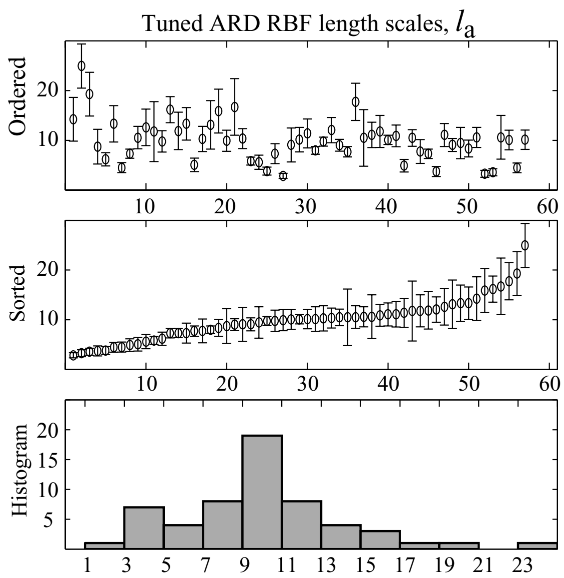

Fig. 3. Spam, final length scales as tuned by ARD

	d	n	Error: Bayes	Error: CV	Error: Previous
Ions	33	351	8.3 ± 2.7	6.4 ± 2.5	4.0[(1)]
Musk	166	476	10.4 ± 2.1	6.7 ± 2.1	7.6[(2)]
Spam	57	4601	7.3 ± 0.4	7.5 ± 0.5	9.2[(3)]
Spectf	44	349	17.1 ± 3.3	12.9 ± 3.2	23.0[(4)]
Splice	60	3175	6.3 ± 1.6	10.9 ± 0.8	5.3[(5)]
WDBC (new)	30	569	2.9 ± 1.2	2.9 ± 0.9	2.5[(6)]

TABLE I
COMPARISON OF ERROR RATES

Number of features, d, and number of samples, n, for high-dimensional data sets from the UCI Machine Learning Repository, along with test errors ± standard deviation across 20 independent runs. Previous results for each data set are reproduced from: [(1)] (Sigilito et al., 1989), [(2)] (Dietterich et al., 1997), [(3)] (Bursteinas and Long, 2000), [(4)] (Kurgan et al., 2001), [(5)] (Chu et al., 2003) , [(6)] (Mangasarian et al., 1995)

are summarized in table I along with examples of error rates found previously in the literature.

From the results in table I it appears that the advantage in generalization performance of the ARD over the non-ARD system may be the exception rather than the rule among the classification problems in the UCI repository. While the Bayesian ARD SVM has an advantage on the Splice data, the non-ARD CV SVM has equal or better performance on the other relatively high-dimensional data sets. In general the results for the Bayesian ARD SVM suffered from the drawback discussed previously in (Gold and Sollich, 2005): there was no single stopping criterion, in terms of the decrease of the average magnitude of the gradients from their peak (starting) value, that gave good results for all data sets. In particular, while the best result for Splice was achieved with a stopping criterion of 15% of peak gradient magnitude, for most other data sets 30% gave best results. We experimented with withholding a portion of the training data in order to use the error on this validation set as a stopping criterion but found that this did not improve the overall generalization performance of the Bayesian ARD SVM. In particular for the data sets with fewer training examples (i.e. Ions, Musk, Spectf) there was not enough data to both effectively train the SVM (including tuning the hyperparameters) and also estimate the generalization performance.

In order to better understand these results we analyzed the length scales tuned by the Bayesian ARD SVM, to see how the Splice data set differs from the others. Figure (3) illustrates the results using the Spam data set in which the input features are 57 statistics of word and character occurrences in an email and the task is to discriminate spam from non-spam emails. In contrast to the Splice data set, where the length scales give a relatively clear division into relevant and irrelevant features, in the Spam task the length scales vary gradually from the most relevant (frequency of the character '!') to the least relevant (frequency of the word 'address') feature, and the histogram shows a broad and largely featureless distribution of lengthscales; see figures 2, 3 (middle and bottom). The distribution of tuned length scales for the other data sets varied, although they were all more similar to the distribution for Spam than to that for Splice.

We next investigated the use of the length scales tuned by the Bayesian ARD SVM for feature selection. In order to have a systematic method for determining how many features to retain, we defined the overall relevance of a given subset of features as

$$\omega_I = \frac{\sum_{a \in I} 1/l_a^2}{\sum_{b=1}^{d} 1/l_b^2} \qquad (22)$$

where I indicates the set of features selected. This is mo-

ω_I	Ions			Splice			WDBC (new)		
	d'	% Features	Error	d'	% Features	Error	d'	% Features	Error
0.8	17	51.5	6.3 ± 2.7	25	41.7	8.7 ± 0.7	13	43.3	3.3 ± 1.1
0.7	12	36.4	5.6 ± 2.4	13	21.7	7.8 ± 0.5	9	30.0	3.3 ± 1.1
0.6	8	24.2	6.6 ± 2.2	7	11.7	6.3 ± 0.6	7	23.3	3.4 ± 1.1
0.5	5	15.2	8.2 ± 2.5	4	6.7	9.6 ± 0.6	4	13.3	4.2 ± 1.3
0.2	16	53.3	8.3 ± 2.8	35	58.3	36.4 ± 1.0	17	56.7	4.8 ± 1.3
0.1	9	27.2	10.4 ± 2.3	18	30.0	38.2 ± 1.3	12	40.0	7.8 ± 1.6

TABLE II

RESULTS OF FEATURE SELECTION FOR THREE SELECTED DATA SETS

Top: results of choosing the most relevant features to make up the indicated fractional relevance ω_I, equation (22). Bottom: results of choosing the *least* relevant features.

tivated by the form (3) of the ARD RBF kernel: since individual features x_a are normalized to have zero mean and unit variance, the typical contribution of each feature to the exponent in (3) is $1/l_a^2$. We therefore take this quantity as a measure of the relevance of the feature; ω_I is then just the fraction of the overall relevance captured by the chosen feature subset. This definition accounts for the potentially gradual variation of length scales across features, whereas a naive feature count would not, and accordingly one may hope that as ω_I is reduced from its maximal value of 1 similar changes in performance will be seen across a range of data sets. Of course, there is not necessarily a direct link between ω_I and the resulting classification performance, in a way that would be analogous to the fraction of total variance explained by selected components in a principal component analysis. One could argue, for example, that the Bayesian evidence for the selected subset (with the lengthscales for the subset specifically adapted to this selection) would be a better predictor of classification performance, but this would be very much more expensive to compute than the simple (22). We selected features by first fixing a threshold value for ω_I, and then adding the most relevant (large l_a) features until this threshold was reached or exceeded. A non-ARD SVM was then trained using this reduced feature set, with hyperparameters C and l tuned by CV as before; the quality of the selected feature subset was estimated by the average test set error over 20 random training/test splits. Effectively we are using ARD, with its continuously varying l_a, as a means of producing a "hard" feature selection for a non-ARD classifier; note that no explicit search over the 2^d different feature subsets is involved.

Table II (top) illustrates the results for feature selection on three of the sample data sets. We chose features subset with fractional relevance ω_I between 0.5 and 0.8. The performance of the non-ARD SVM on the Splice data set *improves* substantially as irrelevant features are removed, down to a fractional relevance ω_I of about 0.6 corresponding to the seven most relevant features. Comparison with table I shows that at this point the performance of the non-ARD SVM matches that of the Bayesian ARD SVM. This implies that the further freedom in ARD of tuning the individual length scales of the few features retained does not translate into additional gain in classification accuracy; in other words, the main effect of ARD is to effectively remove the irrelevant features. As the fractional relevance ω_I is decreased further to 0.5, performance of the CV SVM begins to decline as features directly relevant to the classification are removed.

In contrast, for the WDBC data set we observe from table II that removing features never improves the performance of the non-ARD SVM; even the least relevant features do contribute to the classification accuracy. More striking is the remarkable robustness of performance to the removal of a substantial fraction of the features: for a fractional relevance of $\omega_I = 0.6$ the corresponding subset of features chosen by ARD contains fewer than 25% of all features but classification accuracy remains almost unaffected. The Ions data set shows an intermediate scenario where removal of features does give a slight benefit, though this is small compared to the error bars. Data sets not shown had results similar to those for WDBC.

In order to confirm that the Bayesian ARD SVM was correctly choosing the most relevant features we reversed the selection criterion and instead chose the *least* relevant features in order to make up a fractional relevance $\omega_I = 0.1 \ldots 0.2$. This results in the selection of as many or more features as make up the top 50%-70% of the total relevance. If our method correctly discriminates relevant and irrelevant features then we expect to see a significant difference in performance in spite of this comparable number of features retained. The results in table II (bottom) show that when the least relevant features are chosen the performance does indeed drop substantially. This leads us to conclude that the Bayesian ARD SVM correctly ranks features according to their relevance even in cases where irrelevant features do not harm the generalization performance of a non-ARD SVM.

VI. Conclusion

We have described a Nyström-based method for significantly speeding up hyperparameter tuning in SVM classifiers. Our experiments show that the advantage of the resulting Bayesian ARD SVM over a non-ARD SVM, with its single length scale tuned by cross-validation, is most significant in cases where only a small minority of features are relevant to the classification problem. Otherwise performance differences to standard non-ARD SVMs are within error bars. This is partly due to the difficulty of finding a stopping criterion for the gradient ascent on the Bayesian evidence that works well for all data sets. Encouragingly, when using the ARD approach for feature selection we found that it correctly ranks features by relevance even in cases where it does not achieve performance superior to the non-ARD CV SVM.

As noted above, because the computational cost of training an SVM does not scale significantly with the number of input features there is less of an incentive to pursue feature selection unless it results in tangible improvements in the generalization performance. It should be emphasized, however, that the ARD approach manages to prune correctly up to 75% of all features as irrelevant, without significantly impairing performance. In the context of previous work on feature extraction for classification problems this would be considered a substantial success. What has changed the context for evaluation is the fact that while past classification systems such as decision trees and back-propagation networks suffered significant performance losses in the face of a moderate proportion of irrelevant features, the SVM classification algorithm seems to have rendered feature selection somewhat marginal since even with many features of low relevance performance is not necessarily impaired. Still, feature selection may be useful for knowledge discovery: if the goal is to improve classification performance by adding previously unused features it is clearly helpful to know which of the currently used features actually contribute to performance.

Furthermore, these observations pertain mainly to what we would consider "traditional" classification problems where the input features have been selected by human domain experts and consequently it is unlikely that so many features will be irrelevant as to impair classification. However, recent work has applied SVMs to classification tasks with inputs containing a large number of automatically generated features, $e.g.$ in machine vision or DNA analysis (Weston et al., 2000). In such situations there is no a priori guarantee that all or even most of the features generated will be relevant to the classification problem and feature selection with a technique like the Bayesian ARD SVM should remain beneficial. At present we are experimenting with this approach to evaluate the usefulness of various features generated for visual object classification, with encouraging preliminary results.

To facilitate further research in this area, we are planning to make the C code with our implementation of Hybrid Monte Carlo sampling using the Nyström approximation available to other researchers, at http://www.mth.kcl.ac.uk/$\sim$psollich.

Acknowledgment: This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

References

Bursteinas, B. and Long, J. A. (2000). Transforming supervised classifiers for feature extraction (extended version). 12th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'00).

Chapelle, O., Vapnik, V., Bousquet, O., and Mukherjee, S. (2002). Choosing multiple parameters for Support Vector Machines. *Mach. Learn.*, 46(1-3):131–159.

Chu, W., Keerthi, S., and C, O. (2003). Bayesian trigonometric Support Vector classifier. *Neural Computation*, 15:2227–2254.

Cristianini, N. and Shawe-Taylor, J. (2000). *An introduction to Support Vector Machines*. Cambridge University Press, Cambridge.

Dietterich, T. G., Lathrop, R. H., and Lozano-Perez, T. (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71.

Fine, S. and Scheinberg, K. (2001). Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264.

Fröhlich, H. and Zell, A. (2004). Feature subset selection for Support Vector Machines by incremental regularized risk minimization. volume 3, pages 2041 – 2046. IEEE Int. Joint Conf. on Neural Networks (IJCNN).

Gestel, T. V., Suykens, J. A. K., Moor, B. D., and Vandewalle, J. (2002). Bayesian inference for LS-SVMs on large data sets using the Nyström method. In *Proc. of the World Congress on Computational Intelligence - International Joint Conference on Neural Networks (WCCI-IJCNN 2002)*, pages 2779–2784.

Gold, C. and Sollich, P. (2003). Model selection for Support Vector Machine classification. *Neurocomputing*, 55:221–249.

Gold, C. and Sollich, P. (2005). Fast Bayesian Support Vector Machine parameter tuning with the nyström method. In *Proceedings of the International Joint Conference on Neural Networks,* July 31 - August 4, 2005, Montreal, Canada.

Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324.

Kurgan, L. A., Cios, K. J., Tadeusiewicz, R., Ogiela, M., and Goodenday, L. (2001). Knowledge discovery approach to automated cardiac spect diagnosis. *Artificial Intelligence in Medicine*, 23(2):149–169.

Kwok, J. T. Y. (2000). The evidence framework applied to Support Vector Machines. *IEEE Trans. Neural Netw.*, 11(5):1162–1173.

MacKay, D. (1998). Introduction to gaussian processes. volume 168, pages 133–165. Springer, Neural networks and machine learning: NATO-ASI Series F: Computer and Systems Sciences.

Mangasarian, O. L., Street, W. N., and Wolberg, W. H. (1995). Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4):570–577.

Neal, R. M. (1993). Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto.

Opper, M. and Winther, O. (2000a). Gaussian process classification and SVM: Mean field results and leave-one-out estimator. In Smola, A. J., Bartlett, P., Schölkopf, B., and Schuurmans, D., editors, *Advances in Large Margin Classifiers*, pages 43–65, Cambridge, MA. MIT Press.

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

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

Seeger, M. (2000). Bayesian model selection for Support Vector Machines, Gaussian processes and other kernel classifiers. In *NIPS 12*, pages 603–609.

Sigilito, V. G., Wing, S., Hutton, L., and Baker, K. (1989). Classification of rader returns from the ionosphere using neural networks. *John Hopkins APL Technical Digest*, 10:262–266.

Sollich, P. (2000). Probabilistic methods for Support Vector Machines. In *NIPS 12*, pages 349–355.

Sollich, P. (2002). Bayesian methods for Support Vector Machines: Evidence and predictive class probabilities. *Machine Learning*, 46:21–52.

Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T., and Vapnik, V. (2000). Feature selection for SVMs. In *NIPS*, pages 668–674.

Williams, C. K. I. and Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In *NIPS 13*.