Fast Bayesian Support Vector Machine Parameter Tuning with the Nystrom Method

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

Peter Sollich
Dept. of Mathematics
King's College London
Strand, London WC2R 2LS, U.K.
E-mail: peter.sollich@kcl.ac.uk

***Abstract*— We experiment with speeding up a Bayesian method for tuning the hyperparameters of a Support Vector Machine (SVM) classifier. The Bayesian approach gives the gradients of the evidence as averages over the posterior, which can be approximated using Hybrid Monte Carlo simulation (HMC). By using the Nystrom approximation to the SVM kernel, our method significantly reduces the dimensionality of the space to be simulated in the HMC. We show that this speeds up the running time of the HMC simulation from $O(n^2)$ (with a large prefactor) to effectively $O(n)$, where n is the number of training samples. We conclude that the Nystrom approximation has an almost insignificant effect on the performance of the algorithm when compared to the full Bayesian method, and gives excellent performance in comparison with other approaches to hyperparameter tuning.**

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.* [1]). The offset parameter b is treated as incorporated into $\mathbf{w}$ in the following, by augmenting feature space vectors to $\boldsymbol{\phi}(x) \to (\boldsymbol{\phi}(x), 1)$.

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

$$\frac{1}{2}||\mathbf{w}||^2 + C \sum_i l_p(y_i \mathbf{w} \cdot \boldsymbol{\phi}(x_i)) \qquad (1)$$

$$l_p(z) = \frac{1}{p}(1-z)^p H(1-z)$$

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 (see *e.g.* [1]) that the optimal weight vector is $\mathbf{w}^* = \sum_i y_i \alpha_i \boldsymbol{\phi}(x_i)$. For the linear penalty case $p = 1$, the α_i are found from

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

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 [1]. A common choice of kernel is the radial basis function (RBF) form

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

The l_a are length scales, one for each of the d input space dimensions. One of the main open challenges in SVM classification, and our focus in this paper, is then the *hyperparameter tuning problem*: find optimal – in terms of generalization performance – values for the (many, for large d) hyperparameters $\{l_a\}$, k_0, k_{off} and C, preferably without holding out parts of D as a test set.

II. BAYESIAN INTERPRETATION OF SVMS

In the probabilistic interpretation of SVM classification (see *e.g.* [2], [3] 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 [4], [5], [6]. 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

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

The constant factor $\kappa(C)$ is a normalization constant. The overall normalization of the probability model is somewhat subtle, and fully discussed in [2]; in line with other work on probabilistic interpretations of SVMs [4], [5], [6], [7], [8], we disregard this issue here.

A. Evidence gradients

With 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 $\boldsymbol{\theta} = (\theta_1 \dots \theta_n)$. 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

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

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 [9]

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

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 [9]

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

where $\mathbf{Y} = \mathrm{diag}(y_1 \dots y_n)$ and $l'_p(\mathbf{Y}\boldsymbol{\theta})$ is understood componentwise, ${l'}_p^{\mathrm{T}}(\mathbf{Y}\boldsymbol{\theta}) = (l'_p(y_1\theta_1) \dots l'_p(y_n\theta_n))$; $l'_p(z) \equiv dl_p/dz$.

The posterior averages required in the expressions (8,9) are still not analytically tractable; we therefore use Hybrid Monte Carlo (HMC, see *e.g.* [10]) 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

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

where $\mathbf{M}$ is the inverse covariance matrix of the momenta, chosen later to simplify the computation. The corresponding stationary "Boltzmann" distribution $P(\boldsymbol{\theta}, \mathbf{p}) \propto \exp[-\mathcal{H}(\boldsymbol{\theta}, \mathbf{p})] \propto \exp(-\frac{1}{2}\mathbf{p}^{\mathrm{T}}\mathbf{M}\mathbf{p})Q(\boldsymbol{\theta}|D)$ 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 $\exp(-\frac{1}{2}\mathbf{p}^{\mathrm{T}}\mathbf{M}\mathbf{p})$; 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

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

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 $\mathbf{M}$ [9]. 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 [9] 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. Nystrom Approximation to the Kernel

As demonstrated in [11], the *Nystrom method* can be used to reduce the run time of many kernel algorithms. One first

approximates the Gram or kernel matrix by a standard 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. The key idea of the Nystrom method is then to estimate the λ_i and u_i from a random subsample of the full training set, *i.e.* from a submatrix $\mathbf{K}_{m,m}$. If this gives eigenvalues $\lambda_i^{(m)}$ and eigenvectors $\mathbf{u}_i^{(m)}$, the Nystrom approximation to (14) is [11]

$$\tilde{\mathbf{K}} = \sum_{i=1}^{p} \tilde{\lambda}_i \tilde{\mathbf{u}}_i \tilde{\mathbf{u}}_i^{\mathrm{T}} \tag{15}$$

$$\tilde{\lambda}_i = \frac{n}{m} \lambda_i^{(m)} \tag{16}$$

$$\tilde{\mathbf{u}}_i = \sqrt{\frac{m}{n}} \mathbf{K}_{n,m} \mathbf{u}_i^{(m)} \tag{17}$$

where $\mathbf{K}_{n,m}$ is the $n \times m$ submatrix of $\mathbf{K}$ containing the columns from the selected subsample. If the $\lambda_i^{(m)}$ decay sufficiently quickly, one can choose $p < m$ in (14). The advantage of the Nystrom approach is that its run time is only $O(m^2 n)$ rather than the usual $O(n^3)$ required for a full eigensolution. Applications in [11] show that in *e.g.* Gaussian process classifiers values of m significantly smaller than n can often be used without impairing performance, and this motivated us to investigate the applicability of the method to SVM hyperparameter tuning by evidence gradient ascent.

To apply the Nystrom method to our HMC simulation, 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 $\langle \mathbf{b}\mathbf{b}^{\mathrm{T}} \rangle = \mathbf{1}$ and hence 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; $\mathbf{b}$ 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. When needed for evaluating gradients and $\mathcal{H}$, $\boldsymbol{\theta}$ is calculated from $\mathbf{b}$ according to (18). The resulting equations for the HMC trajectories are

$$\frac{db_i}{d\tau} = \frac{\partial \mathcal{H}}{\partial p_i} = p_i \tag{20}$$

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

For the average (9) we also need $\tilde{\mathbf{K}}^{-1}\boldsymbol{\theta} = \tilde{\mathbf{K}}^{-1}\mathbf{V}\mathbf{b}$. Even though $\tilde{\mathbf{K}}$ itself does not have full rank, $\tilde{\mathbf{K}}^{-1}\mathbf{V}$ is 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 using 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 [11], finding $(\tilde{\mathbf{K}} + \sigma\mathbf{1})^{-1}\mathbf{V}\mathbf{b}$ and taking $\sigma \to 0$.) Since 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, the entire $n \times p$ matrix $\mathbf{Y}(\partial\mathbf{K}/\partial\lambda)\mathbf{V}(\mathbf{V}^{\mathrm{T}}\mathbf{V})^{-1}$ is precalculated at the beginning of each HMC simulation, requiring $O(n^2 p + p^3)$ operations for each hyperparameter λ.

From (20) we read off that at each discretized trajectory step in the Nystrom 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 Nystrom approximation to the kernel thus reduces the computational complexity of the main loop of the HMC sampling procedure from effectively $O(n^2)$ to $O(np)$.

IV. Experiments and results

A. Evaluating the Nystrom HMC Method

To evaluate the performance of the Nystrom HMC estimate of the evidence gradients, we performed hyperparameter tuning experiments on a number of standard benchmark data sets (see [9] for details), varying the number of samples in the Nystrom approximation, m, and the number of eigenvalues in the HMC, p. Although only 30-50 gradient ascent steps can suffice for practical applications, the gradient ascent can result in some "hyperparameter overfitting" [9]. We therefore ran the experiments for 100 steps in order to asess the maximum extent of this overfitting. For comparison with previously published studies we used a fixed split of the data into training and test sets and evaluated the existence of local minima in the evidence by repeating trials of the gradient ascent from randomly chosen starting points; see [9] for details.

Fig. 1 shows the resulting minimum and final test errors discovered over the course of the gradient ascent for varying values of p and m. Both minimum and final errors are shown in order to illustrate the extent of the overfitting. Although the results vary somewhat for the different data sets, we see a small but progressive decline in performance from the full HMC as either p or m are decreased. The decrease in performance becomes pronounced only for $p < 10$ and $m < 100$. The impact of the approximation on the overfitting is somewhat harder to characterize: in some cases the approximation seems to reduce overfitting (*e.g.* for the Pima and WDBC data sets) while in others the reverse is true (*e.g.* for the Ringnorm data set). In any event overfitting can be minimized with an appropriate stopping criterion, discussed below.

Table I shows the run time for a 50 step gradient ascent with varying numbers of eigenvalues, p. For comparison, run times are also shown for the full HMC used in [9]. The results show an approximately linear scaling of the run time in the number of eigenvalues used, and a substantial time reduction over the full HMC. (Run times for different numbers of samples in the Nystrom approximation are not shown, as they are practically indistinguishable from each other.) This illustrates that the $O(np)$ scaling of the main loop completely dominates the $O(n^2 p + p^3)$ setup time in the HMC sampling process. Since we also find that p does not need to scale with the number of

hyperparameters or the number of training points, we conclude that the Nystrom HMC method described is effectively $O(n)$ for problems with up to a few dozen hyperparameters and several hundred training points. For larger n the $O(n^2 p)$ setup time per hyperparameter will eventually dominate, but even then the algorithm scales with n only in the same way as the best SVM training algorithms.

Data set	d	n	p 5	10	20	40	Full
Pima	7	200	0:5	0:10	0:17	0:32	5:01
Ringnorm	20	300	0:25	0:44	1:12	2:17	29:40
Twonorm	20	300	0:15	0:27	0:47	1:40	21:30
WDBC	30	300	0:20	0:37	0:50	1:24	26:00

TABLE I

RUN TIME VS. # OF EIGENVALUES IN NYSTROM APPROXIMATION

Average run time, h:mm, for a 50 step gradient ascent in the evidence using different numbers of eigenvalues, p. Run times of the full gradient ascent from [9] are also shown. These run times were obtained on a single processor 2.4 GHz Intel Pentium 4. The number of input dimensions d and sizes of training sets n are also shown.

B. Comparison with other hyperparameter tuning methods

We compared the performance of SVM parameter tuning via Nystrom HMC evidence gradient ascent to five other approaches proposed in the literature: Bayesian trigonometric SVMs [12], standard SVMs tuned by optimizing span or radius bounds [14] or by cross validation [13], and Adaboost [13].

Bayesian trigonometric SVMs are a modification of SVMs which replace the standard loss functions by the so called trigonometric loss: $l(z)_{-1<z<1} = 2\ln\sec(\frac{\pi}{4}(1-z))$; $l(z)_{z<-1} = \infty$; $l(z)_{z>1} = 0$. Use of the trigonometric loss function allows the definition of a normalized Bayesian evidence, which can be estimated along with its gradients using a Laplace approximation [12].

In Ref. [14] the tuning of the SVM hyperparameters is performed using smoothed approximations to the gradients of the radius-margin bound, $R^2/(n\gamma^2)$, and the span error bound. The radius-margin bound is defined by the minimum margin, $\gamma = \min_i y_i(\mathbf{w}\cdot\boldsymbol{\phi}(x_i)+b)/||\mathbf{w}||^2$, and the radius of the data set $R(D) = \min_{\mathbf{a},\mathbf{x_i}} \|\boldsymbol{\phi}(x_i)+\mathbf{a}\|$. The span error bound is an estimate of the leave-one-out CV error given by $T = (1/n)\sum_{p=1}^{n}[\alpha_i S_i^2 - 1]_+$, where S_i is the distance in feature space between the point $\boldsymbol{\phi}(x_i)$ and the set $\Lambda_i = \{\sum_{j\neq i,\alpha_j>0}\lambda_i\boldsymbol{\phi}(x_i), \sum_{j\neq i}\lambda_i = 1\}$.

Ref. [13] experiments with a version of the Adaboost algorithm, in which ensembles of weaker classifiers (e.g. radial basis function networks) are created by successively adjusting the sample distribution of the training data, using the margins of each classifier to calculate the re-weighting. The results are compared to a single length scale SVM with hyperparameters tuned by cross-validation, and both [12] and [14] use the data and method described in [13] for testing. In order to make all the results comparable we also follow this approach: Each data set was randomly split into 100 training and test set

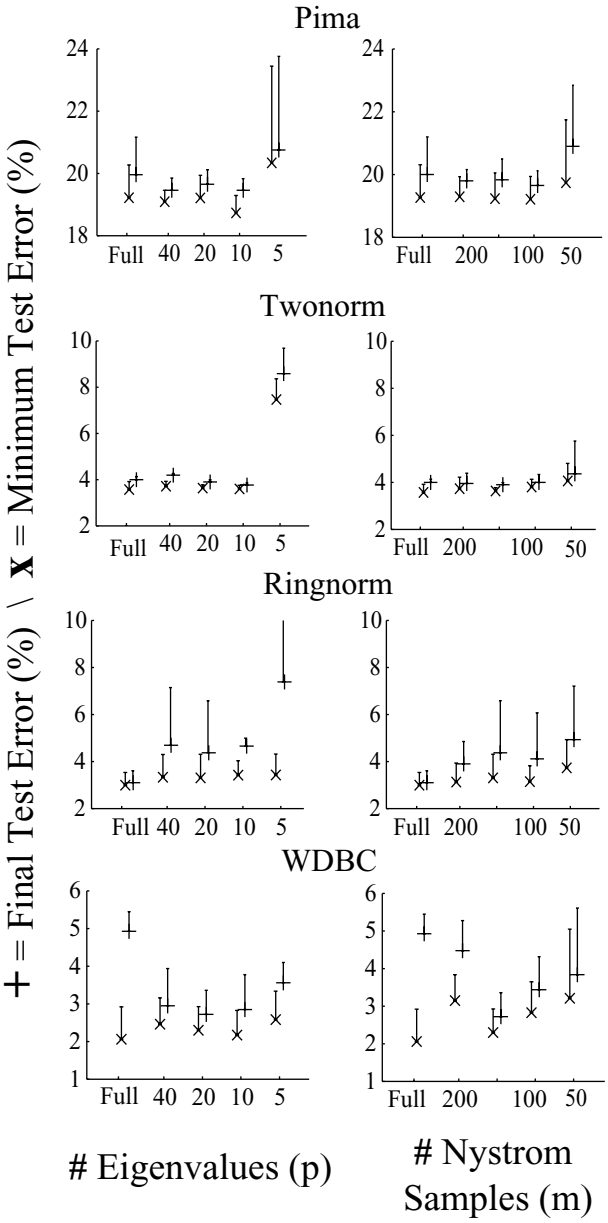

Fig. 1. Comparison of test errors (minimum and final) achieved for the Bayesian evidence gradient ascent Nystrom HMC hyperparameter tuning with varying numbers of eigenvalues, p (left column), and samples in the Nystrom approximation, m (right column). For the left column $m = 150$, and for the right column $p = 20$. For reference, results are also shown for the standard HMC with $p = m = n$ [9] ('Full') . Markers show averages over 25 trials. One-standard deviation error bars are shown "pessimistically", *i.e.* only upwards from the mean, because the error distributions are highly skewed in this direction [9].

Data set	Nys. Bayes. SVM	Trig. Bayes. SVM$^{(1)}$	CV SVM$^{(2)}$	R^2/γ^2 SVM$^{(3)}$	Span SVM$^{(3)}$	AB$^{(2)}$
Banana	10.5 ± 0.4	10.4 ± 0.5	11.5 ± 0.7	NA	NA	10.9 ± 0.4
Breast Cancer	23.6 ± 4.3	26.5 ± 4.6	26.0 ± 4.7	26.8 ± 4.7	25.6 ± 4.2	26.5 ± 4.5
Diabetes	23.1 ± 1.7	23.2 ± 1.8	23.5 ± 1.7	23.3 ± 1.7	23.2 ± 1.7	23.8 ± 1.8
Flare-Solar	32.3 ± 1.8	34.4 ± 1.8	32.4 ± 1.8	NA	NA	34.2 ± 2.2
German	23.9 ± 2.3	23.5 ± 2.1	23.6 ± 2.1	NA	NA	24.3 ± 2.1
Heart	16.7 ± 3.3	16.3 ± 2.9	16.0 ± 3.3	15.9 ± 3.2	16.1 ± 3.1	16.5 ± 3.5
Image	2.6 ± 0.4	2.6 ± 0.5	3.0 ± 0.6	NA	NA	2.7 ± 0.6
Ringnorm	1.8 ± 0.1	2.0 ± 0.3	1.7 ± 0.1	NA	NA	1.6 ± 0.1
Splice	5.1 ± 0.4	5.3 ± 0.7	10.9 ± 0.7	NA	NA	9.5 ± 0.7
Thyroid	4.6 ± 2.2	4.3 ± 2.9	4.8 ± 2.2	4.6 ± 2.0	4.6 ± 2.0	4.6 ± 2.2
Titanic	22.8 ± 1.1	22.7 ± 1.4	22.4 ± 1.0	22.9 ± 1.2	22.5 ± 0.9	22.6 ± 1.2
Twonorm	3.2 ± 0.3	2.9 ± 0.3	3.0 ± 0.2	NA	NA	2.7 ± 0.2
Waveform	10.9 ± 0.5	10.1 ± 0.4	9.9 ± 0.4	NA	NA	9.8 ± 0.8

TABLE II

TEST ERROR COMPARISON FOR SVM PARAMETER TUNING METHODS

Test errors $\pm$ standard deviation for different algorithms used to select SVM hyperparameters. The Bayesian results are for $p = 20$ and $m = 200$. $^{(1)}$Reproduced from [12]. $^{(2)}$Results for the AdaBoost$_{Reg}$ algorithm of Ref. [13], which gave the best results among the AdaBoost algorithms tested. $^{(3)}$Reproduced from [14].

partitions.[1] The tuning algorithm to be tested is then used to optimize hyperparameters on the first 5 of these partitions. Each hyperparameter is set to the median of the resulting 5 values, and this hyperparameter set is tested out of sample on the remaining partitions of the data set [13].

Table II shows the results for the Bayesian gradient ascent method using the Nystrom HMC and for the other methods described above. Because the Bayesian gradient ascent method is subject to overfitting we combined it with a stopping criterion to obtain these results. As described in [9], the gradients typically decline as the gradient ascent progresses to a local maximum in the evidence. Therefore a natural stopping criterion is to compare the magnitude of the gradients to their peak magnitude, and stop when this ratio falls below a given percentage. Tests found that good overall results are achieved when the algorithm is stopped when the average gradient magnitude is 15% of its peak value. However, [9] also showed that when the gradients are small they become noisy and because the algorithm includes adaptation of the ascent rate, the hyperparameters may settle to final values even if not all of the gradients are consistently close to zero. Consequently, we used a secondary criteria that a run terminates if all of the hyper-parameters have changed by less than 1% per step for the last five steps (i.e. the run terminates for whichever stopping criteria is met first.)

The results shown in Table II demonstrate that SVM hyper-parameter tuning by Nystrom HMC evidence gradient ascent achieves generalization performance that is as good as that of other state of the art methods. We note that the difference in the average errors on the test partitions are in all cases small compared to the variance. The only exception to this is the performance of the two Bayesian methods on the Splice data set. In this case the Bayesian methods correctly deduce that despite the high dimension of the inputs, only a small subset of the features are relevant to the classification problem. Consequently the Bayesian methods significantly outperform methods that do not incpororate some form of Automatic Relevance Detection. (Results on the Splice data set are not available for the methods described in [14].)

[1]The larger data sets, Splice and Image, are divided only into 20 partitions.

Data set	Stopping Criterion 30%	20%	10%
Banana	10.4	10.5	10.6
Heart	16.2	16.5	16.9
Thyroid	3.9	4.7	4.7
Twonorm	2.9	3.0	3.7
Waveform	9.8	10.1	11.2

TABLE III

IMPACT OF OVERFITTING IN GRADIENT ASCENT RESULTS

Average Test errors on the testing partitions when SVM parameter tuning is stopped when average normalized gradients reach the indicated percentage of their peak amplitude.

Data set	d	n	# Steps	Time
Banana	2	400	21 ± 6	0:25
Breast Cancer	9	200	87 ± 10	0:54
Diabetes	8	468	31 ± 4	0:58
Flare-Solar	9	666	33 ± 6	1:33
German	20	700	38 ± 8	2:18
Heart	13	170	29 ± 7	0:15
Image	18	1300	57 ± 24	6:10
Ringnorm	20	400	29 ± 9	0:42
Splice	60	1000	35 ± 9	4:36
Thyroid	5	140	41 ± 6	0:12
Titanic	3	150	52 ± 19	0:20
Twonorm	20	400	28 ± 27	0:33
Waveform	21	400	49 ± 4	1:11

TABLE IV
RUN TIMES ON TEST DATA SETS

The number of input dimensions, d, size of training set, n, and average number of steps (± standard deviation) for Bayesian gradient ascent hyperparameter tuning for the first 5 partitions of the data sets shown in Table II; the average number of steps to reach the stopping criteria is shown as well as the average run time, in h:mm. All results are for $p = 20$ and $m = 200$. These run times were obtained on a single processor 2.4 GHz Intel Pentium 4.

Although the stopping criterion of 15% of the normalized gradient amplitude gave good overall performance, it did not lead to optimal performance on all of the datasets, as detailed in Table III. For 5 out of 13 data sets, the optimal stopping point was somewhat earlier, when the gradients are at around 30% of the peak amplitude. Although the impact of the precise stopping criteria is small in most cases, it is the main reason why tuning SVM hyperparameters with Bayesian gradient ascent using the Nystrom method does not give overall superior performance in comparison to the other methods. Improvements to the stopping criteria are an area that will benefit from further research.

Table IV shows the average steps at which the Bayesian evidence gradient ascent tuning algorithm stopped for the first 5 partitions of the test data sets, along with the average run time to reach that point. The run times for the parameter tuning range from around a dozen minutes to around 6 hours, with the exact time depending on the input dimension, the number of training points, and the number of steps required for the gradient ascent algorithm to reach the stopping criterion. Although the computational cost of using the algorithm is considerably higher than a simpler method like choosing SVM hyperparameters with cross validation, it is practical to use with commonly available computing resources and gives superior performance. In comparison to the Bayesian trigonometric SVM, while the run time is somewhat longer and the test results comparable, the Bayesian evidence gradient ascent with the Nystrom HMC method has the advantage of not relying on a single, non-standard formulation of the loss function (which imposes hard constraints on the classifier due to its divergence at $z = -1$); instead, it can be used with all of the common loss functions.

V. CONCLUSION

We conclude that SVM hyperparameter tuning using gradient ascent on the Bayesian evidence can be significantly speeded up by using the Nystrom approximation in the HMC estimation of evidence gradients. Our experiments show that reasonably small numbers of samples in the Nystrom approximation and eigenvalues in the HMC can be used. The method thus gives excellent performance with a reasonable amount of computational effort, and has benign scaling with training set size n. We believe that the Bayesian interpretation of SVM classifier is an attractive area for further reseach, *e.g.* regarding selection between alternative kernel shapes, the effects of using a normalized version of the evidence [2] and the prediction of class probabilities rather than deterministic outputs.

The success of SVM hyperparameter tuning using gradient ascent on the Bayesian evidence at Automatic Relevence Determination may make the technique particularly valuable for data mining applications. In such applications there are typically a large number of input dimensions, but the relevance of the inputs is unkown. The main barrier to the use of the technique is the complexity of the algorithms required for the Hybrid Monte Carlo simulation using the Nystrom Approximation. Consequently, in order to assist further research in this area, we plan to make the C code with which this research was performed available to other researchers, at `http://www.mth.kcl.ac.uk/~psollich`.

REFERENCES

[1] N. Cristianini and J. Shawe-Taylor, *An introduction to Support Vector Machines*. Cambridge: Cambridge University Press, 2000.

[2] P. Sollich, "Bayesian methods for Support Vector Machines: Evidence and predictive class probabilities," *Machine Learning*, vol. 46, pp. 21–52, 2002.

[3] ——, "Probabilistic methods for Support Vector Machines," in *NIPS 12*, 2000, pp. 349–355.

[4] M. Seeger, "Bayesian model selection for Support Vector Machines, Gaussian processes and other kernel classifiers," in *NIPS 12*, 2000, pp. 603–609.

[5] M. Opper and O. Winther, "Gaussian process classification and SVM: Mean field results and leave-one-out estimator," in *Advances in Large Margin Classifiers*, A. J. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, Eds. Cambridge, MA: MIT Press, 2000, pp. 43–65.

[6] ——, "Gaussian processes for classification: Mean-field algorithms," *Neural Comput.*, vol. 12, no. 11, pp. 2655–2684, 2000.

[7] J. T. Y. Kwok, "Moderating the outputs of Support Vector Machine classifiers," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1018–1031, 1999.

[8] ——, "The evidence framework applied to Support Vector Machines," *IEEE Trans. Neural Netw.*, vol. 11, no. 5, pp. 1162–1173, 2000.

[9] C. Gold and P. Sollich, "Model selection for support vector machine classification," *Neurocomputing*, vol. 55, pp. 221–249, 2003.

[10] R. M. Neal, "Probabilistic inference using Markov chain Monte Carlo methods," University of Toronto, Tech. Rep. CRG-TR-93-1, 1993.

[11] C. K. I. Williams and M. Seeger, "Using the Nystrom method to speed up kernel machines," in *NIPS 13*, 2001.

[12] W. Chu, S. Keerthi, and O. C, "Bayesian trigonometric support vector classifier," *Neural Computation*, vol. 15, pp. 2227–2254, 2003.

[13] G. Rätsch, T. Onoda, and K.-R. Müller, "Soft margins for AdaBoost," *Machine Learning*, vol. 42, no. 3, pp. 287–320, Mar. 2001.

[14] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for Support Vector Machines," *Mach. Learn.*, vol. 46, no. 1-3, pp. 131–159, 2002.