# Chapter 7

# Regularized Least-Squares Classification

Ryan Rifkin, Gene Yeo and Tomaso Poggio

**Abstract.** We consider the solution of binary classification problems via Tikhonov regularization in a Reproducing Kernel Hilbert Space using the square loss, and denote the resulting algorithm Regularized Least-Squares Classification (RLSC). We sketch the historical developments that led to this algorithm, and demonstrate empirically that its performance is equivalent to that of the well-known Support Vector Machine on several datasets. Whereas training an SVM requires solving a convex quadratic program, training RLSC requires only the solution of a single system of linear equations. We discuss the computational tradeoffs between RLSC and SVM, and explore the use of approximations to RLSC in situations where the full RLSC is too expensive. We also develop an elegant leave-one-out bound for RLSC that exploits the geometry of the algorithm, making a connection to recent work in algorithmic stability.

## 7.1 Introduction

We assume that $X$ and $Y$ are two sets of random variables. We are given a **training set** $S = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)$, consisting of $\ell$ independent identically distributed samples drawn from the probability distribution on $X \times Y$. The joint and conditional probabilities over $X$ and $Y$ obey the following relation:

$$p(x, y) = p(y|x) \cdot p(x)$$

It is crucial to note that we view the joint distribution $p(x, y)$ as **fixed** but **unknown**, since we are only given the $\ell$ examples.

In this chapter, we consider the $n$-dimensional binary classification problem, where the $\ell$ training examples satisfy $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i \in \{-1, 1\}$ for all $i$. Denoting the training set by $S$, our goal is to learn a function $f_S$ that will generalize well on new examples. In particular, we would like

$$Pr(\text{sgn}(f_S(\mathbf{x}))) \neq y$$

to be as small as possible, where the probability is taken over the distribution $X \times Y$, and $\text{sgn}(f(\mathbf{x}))$ denotes the sign of $f(\mathbf{x})$. Put differently, we want to minimize the *expected risk*, defined as

$$I_{exp}[f] = \int_{X \times Y} i_{\text{sgn}(f(x)) \neq y} dP,$$

where $i_p$ denotes an indicator function that evaluates to 1 when $p$ is true and 0 otherwise. Since we do not have access to $p(x, y)$, we cannot minimize $I_{exp}[f]$ directly. We instead consider the *empirical risk minimization* problem, which involves the minimization of:

$$I_{emp}[f] = \frac{1}{\ell} \sum_{i=1}^{\ell} i_{\text{sgn}(f(\mathbf{x}_i)) \neq y_i}.$$

This problem is ill-defined, because we have not specified the set of functions that we consider when minimizing $I_{emp}[f]$. If we require that the solution $f_S$ lie in a bounded convex subset of a Reproducing Kernel Hilbert Space $H$ defined by a positive definite kernel function $K$ [5, 6] (the norm of a function in this space is denoted by $||f||_K$), the regularized empirical risk minimization problem (also known as Ivanov Regularization) is well-defined:

$$\min_{f_S \in H, ||f_S||_K \leq R} \frac{1}{\ell} \sum_{i=1}^{\ell} i_{f(\mathbf{x}_i) \neq y_i}.$$

For Ivanov regularization, we can derive (probabilistic) generalization error bounds [7] that bound the difference between $I_{exp}[f_S]$ and $I_{emp}[f_S]$; these bounds will depend on $H$ and $R$ (in particular, on the VC-dimension of $\{f : f \in H \wedge ||f||_K^2 \leq R\}$) and $\ell$. Since $I_{emp}[f]$ can be measured, this in turn allows us to (probabilistically) upper-bound $I_{exp}[f]$. The minimization of the (non-smooth, non-convex) 0-1 loss $i_{f(\mathbf{x}) \neq y}$ induces an NP-complete optimization problem, which motivates replacing $i_{\text{sgn}(f(\mathbf{x})) \neq y}$ with a smooth, convex loss function $V(y, f(\mathbf{x}))$, thereby making the problem well-posed [1, 2, 3, 4]. If $V$ *upper bounds* the 0-1 loss function, then the upper bound we derive on any $I_{exp}[f_S]$ with respect to $V$ will also be an upper bound on $I_{exp}[f_S]$ with respect to the 0-1 loss.

In practice, although Ivanov regularization with a smooth loss function $V$ is not necessarily intractable, it is much simpler to solve instead the closely related (via Lagrange multipliers) Tikhonov minimization problem

$$\min_{f \in H} \frac{1}{\ell} \sum_{i=1}^{\ell} V(y_i, f(\mathbf{x}_i)) + \lambda ||f||_K^2. \tag{7.1}$$

Whereas the Ivanov form minimizes the empirical risk subject to a bound on $||f||_K^2$, the Tikhonov form smoothly trades off $||f||_K^2$ and the empirical risk; this tradeoff is controlled by the regularization parameter $\lambda$. Although the algorithm does not explicitly include a bound on $||f||_K^2$, using the fact that the all-0 function $f(\mathbf{x}) \equiv 0 \in H$, we can easily show that the function $f^*$ that solves the Tikhonov regularization problem satisfies $||f||_K^2 \leq \frac{B}{\ell}$, where $B$ is an upper bound on $V(y, 0)$, which will always exist given that $y \in \{-1, 1\}$. This allows us to immediately derive bounds for Tikhonov regularization that have the same form as the original Ivanov bounds (with weaker constants). Using the notion of uniform stability developed by Bousquet and Elisseef [8], we can also more directly derive bounds that apply to Tikhonov regularization in an RKHS.

For our purposes, there are two key facts about RKHS's that allow us to greatly simplify (7.1). The first is the Representer Theorem [9, 10], stating that, under very general conditions on the loss function $V$, the solution $f^*$ to the Tikhonov regularization problem can be written in the following form:

$$f^*(\mathbf{x}) = \sum_{i=1}^{\ell} c_i K(\mathbf{x}, \mathbf{x}_i).$$

The second is that, for functions in the above form,

$$||f||_K^2 = \mathbf{c}^T K \mathbf{c},$$

where $K$ now denotes the $\ell$-by-$\ell$ matrix whose $(i,j)$'th entry is $K(\mathbf{x}_i, \mathbf{x}_j)$.[1] The Tikhonov regularization problem becomes the problem of finding the $c_i$:

$$\min_{\mathbf{c} \in \mathbb{R}^\ell} \frac{1}{\ell} \sum_{i=1}^{\ell} V(y_i, \sum_{j=1}^{\ell} c_i K(\mathbf{x}_i, \mathbf{x}_j)) + \lambda \mathbf{c}^T K \mathbf{c}. \tag{7.2}$$

A specific learning scheme (algorithm) is now determined by the choice of the loss function $V$. The most natural choice of loss function, from a pure learning theory perspective, is the $L_0$ or misclassification loss, but as mentioned previously, this results in an intractable NP-complete optimization problem. The Support Vector Machine [7] arises by choosing $V$ to be the hinge loss

$$V(y, f(\mathbf{x})) = \begin{cases} 0 & \text{if } yf(\mathbf{x}) \geq 1 \\ 1 - yf(\mathbf{x}) & \text{otherwise} \end{cases}$$

[1]This overloading of the term $K$ to refer to both the kernel function and the kernel matrix is somewhat unfortunate, but the usage is clear from context, and the practice is standard in the literature.

The Support Vector Machine leads to a convex quadratic programming problem in $\ell$ variables, and has been shown to provide very good performance in a wide range of contexts.

In this chapter, we focus on the simple square-loss function

$$V(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2. \tag{7.3}$$

This choice seems very natural for regression problems, in which the $y_i$ are real-valued, but at first glance seems a bit odd for classification — for examples in the positive class, large positive predictions are almost as costly as large negative predictions. However, we will see that empirically, this algorithm performs as well as the Support Vector Machine, and in certain situations offers compelling practical advantages.

## 7.2 The RLSC Algorithm

Substituting (7.3) into (7.2), and dividing by two for convenience, the RLSC problem can be written as:

$$\min F(\mathbf{c}) = \min_{\mathbf{c}\in\mathbb{R}^\ell} \frac{1}{2\ell} \sum_{i=1}^{\ell} (\mathbf{y} - K\mathbf{c})^T(\mathbf{y} - K\mathbf{c}) + \frac{\lambda}{2}\mathbf{c}^T K\mathbf{c}.$$

This is a convex differentiable function, so we can find the minimizer simply by taking the derivative with respect to $\mathbf{c}$:

$$\begin{aligned} \nabla F_{\mathbf{c}} &= \frac{1}{\ell}(\mathbf{y} - K\mathbf{c})^T(-K) + \lambda K\mathbf{c} \\ &= -\frac{1}{\ell}K\mathbf{y} + \frac{1}{\ell}K^2\mathbf{c} + \lambda K\mathbf{c}. \end{aligned}$$

The kernel matrix $K$ is positive semidefinite, so $\nabla F_{\mathbf{c}} = 0$ is a necessary and sufficient condition for optimality of a candidate solution $\mathbf{c}$. By multiplying through by $K^{-1}$ if $K$ is invertible, or reasoning that we only need a single solution if $K$ is not invertible, the optimal $K$ can be found by solving

$$(K + \lambda\ell I)\mathbf{c} = \mathbf{y}, \tag{7.4}$$

where $I$ denotes an appropriately-sized identity matrix. We see that a Regularized Least-Squares Classification problem can be solved by solving a single system of linear equations.

Unlike the case of SVMs, there is no *algorithmic* reason to define the dual of this problem. However, by deriving a dual, we can make some interesting connections. We rewrite our problems as:

$$\begin{aligned} \min_{\mathbf{c}\in\mathbb{R}^\ell} \quad & \tfrac{1}{2\ell}\xi^T\xi + \tfrac{\lambda}{2}\mathbf{c}^T K\mathbf{c} \\ \text{subject to :} \quad & K\mathbf{c} - \mathbf{y} = \xi \end{aligned}$$

We introduce a vector of dual variables $\mathbf{u}$ associated with the equality constraints, and form the Lagrangian:

$$L(\mathbf{c}, \xi, \mathbf{u}) = \frac{1}{2\ell}\xi^T\xi + \frac{\lambda}{2}\mathbf{c}^T K\mathbf{c} - \mathbf{u}^T(K\mathbf{c} - \mathbf{y} - \xi).$$

We want to minimize the Lagrangian with respect to $\mathbf{c}$ and $\xi$, and maximize it with respect to $\mathbf{u}$. We can derive a "dual" by eliminating $\mathbf{c}$ and $\xi$ from the problem. We take the derivative with respect to each in turn, and set it equal to zero:

$$\frac{\partial L}{\partial \mathbf{c}} = \lambda K\mathbf{c} - K\mathbf{u} = 0 \implies \mathbf{c} = \frac{\mathbf{u}}{\lambda} \tag{7.5}$$

$$\frac{\partial L}{\partial \xi} = \frac{1}{\ell}\xi + \mathbf{u} = 0 \implies \xi = -\ell\mathbf{u} \tag{7.6}$$

Unsurprisingly, we see that both $\mathbf{c}$ and $\xi$ are simply expressible in terms of the dual variables $\mathbf{u}$. Substituting these expressions into the Lagrangian, we arrive at the reduced Lagrangian

$$\begin{aligned} L^R(\mathbf{u}) &= \frac{\ell}{2}\mathbf{u}^T\mathbf{u} + \frac{1}{2\lambda}\mathbf{u}^T K\mathbf{u} - \mathbf{u}^T(K\frac{\mathbf{u}}{\lambda} - \mathbf{y} + \ell\mathbf{u}) \\ &= -\frac{\ell}{2}\mathbf{u}^T\mathbf{u} - \frac{1}{2\lambda}\mathbf{u}^T K\mathbf{u} + \mathbf{u}^T\mathbf{y}. \end{aligned}$$

We are now faced with a differentiable concave maximization problem in $\mathbf{u}$, and we can find the maximum by setting the derivative with respect to $\mathbf{u}$ equal to zero:

$$\nabla L^R_{\mathbf{u}} = -\ell\mathbf{u} - \frac{1}{\lambda}K\mathbf{u} + y = 0 \Longrightarrow (K + \lambda\ell I)\mathbf{u} = \lambda\mathbf{y}.$$

After we solve for $\mathbf{u}$, we can recover $\mathbf{c}$ via Equation (7.5). It is trivial to check that the resulting $\mathbf{c}$ satisfies (7.4). While the exercise of deriving the dual may seem somewhat pointless, its value will become clear in later sections, where it will allow us to make several interesting connections.

## 7.3 Previous Work

The square loss function is an obvious choice for regression. Tikhonov and Arsenin [3] and Schönberg [11] used least-squares regularization to restore well-posedness to ill-posed regression problems. In 1988, Bertero, Poggio and Torre introduced regularization in computer vision, making use of Reproducing Kernel Hilbert Space ideas [12]. In 1989, Girosi and Poggio [13, 14] introduced classification and regression techniques with the square loss in the field of supervised learning. They used pseudodifferential operators as their stabilizers; these are essentially equivalent to using the norm in an RKHS. In 1990, Wahba [4] considered square-loss regularization for regression problems using the norm in a Reproducing Kernel Hilbert Space as a stabilizer.

More recently, Fung and Mangasarian considered *Proximal Support Vector Machines* [15]. This algorithm is essentially identical to RLSC in the case of the linear kernel, although the derivation is very different: Fung and Mangasarian begin with the SVM formulation (with an unregularized bias term $b$, as is standard for SVMs), then modify the problem by penalizing the bias term and changing the inequalities to equalities. They arrive at a system of linear equations that is identical to RLSC up to sign changes, but they define the right-hand-side to be a vector of all 1's, which somewhat complicates the intuition. In the nonlinear case, instead of penalizing $\mathbf{c}^T K \mathbf{c}$, they penalize $\mathbf{c}^T \mathbf{c}$ directly, which leads to a substantially more complicated algorithm. For linear RLSC where $n << \ell$, they show how to use the Sherman-Morrison-Woodbury to solve the problem rapidly (see Section 7.4).

Suykens also uses the square loss in his *Least-Squares Support Vector Machines* [16], but allows the unregularized free bias term $b$ to remain, leading to a slightly different optimization problem.

We chose a new name for this old algorithm, Regularized Least-Squares Classification, to emphasize both the key connection with regularization, and the fact RLSC is not a standard Support Vector Machine in the sense that there is no sparsity in the $\mathbf{c}$ vector. The connection between RLSC and SVM is that *both* are instances of Tikhonov regularization, with different choices of the loss function $V$.

## 7.4 RLSC vs. SVM

Whereas training an SVM requires solving a convex quadratic program, training RLSC requires only the solution of a single system of linear equations, which is conceptually much simpler. However, this does not necessarily mean that training RLSC is faster. To solve a general, nonlinear RLSC problem, we must first compute the $K$ matrix, which requires time $O(\ell^2 n)$ and space $O(\ell^2)$.[2] We must then solve the linear system, which requires $O(\ell^3)$ operations. The constants associated with the $O$ notation in RLSC are small and easily measured. The resource requirements of the SVM are not nearly so amenable to simple analysis. In general, convex quadratic programming problems are solvable in polynomial time [17], but the bounds are not tight enough to be practically useful. Empirically, Joachims [18] has observed a scaling of approximately $O(\ell^{2.1})$, although this ignores the dependence on the dimensionality $n$. Modern SVM algorithms are not amenable to direct formal analysis, and depend on a complex tradeoff between the number of data points, the number of Support Vectors, and the amount of memory available. However, as a rule of thumb, for large nonlinear problems, we expect the SVM to be much more tractable than the direct approach to the RLSC problem. In particular, for large problems, we can often solve the SVM problem in less time than it takes to compute every entry in $K$. Furthermore, if we do not have $O(\ell^2)$ memory available to store $K$, we cannot solve the RLSC problem at all via direct methods (we can still solve it using conjugate gradient methods, recomputing the kernel matrix $K$ at every iteration, but this will be intractably slow for large problems). Even if we do manage to train the RLSC system, using the resulting function at test time will be much slower — for RLSC, we will have to compute all $\ell$ kernel products of training points with a new test point, whereas for SVM, we will only have to compute the kernel products with the support vectors, which are usually a small fraction of the training set.

[2]We make the assumption (satisfied by all frequently used kernel functions) that computing a single kernel product takes $O(n)$ operations.

In Section 7.6, we consider various approximations to the RLSC problem [19, 20, 21], but whether these approximations allow us to recover a high quality solution using an amount of time and memory equivalent to the SVM is an intriguing open question.

In the case of linear RLSC, the situation is very different. If we place our data in the $\ell$-by-$n$ matrix $A$, (7.4) becomes

$$(AA^T + \lambda \ell I)\mathbf{c} = \mathbf{y}.$$

If $n << \ell$, we can solve the problem in $O(\ell n^2)$ operations using either the Sherman-Morrison-Woodbury formula [15] or a product-form Cholesky factorization (suggested by Fine and Scheinberg in the context of interior point methods for SVMs [22]); the essential observation is that we can transform the rank $\ell$ problem into a rank $n$ problem. If $n$ is small compared to $\ell$, this leads to a huge savings in computational cost. If $n$ is large compared to $\ell$, this technique is not helpful in general. However, in the important subcase where the data points are *sparse* (have very few nonzero elements per vector), for example in the case of a text categorization task, we can still solve the problem very rapidly using the Conjugate Gradient algorithm [23]. In particular, an explicit representation of $(AA^T + \lambda \ell I)$, which would require $O(\ell^2)$ storage, is not required — instead, we can form a matrix-vector product $(AA^T + \lambda \ell I)\mathbf{x}$ in $O(\ell \overline{n})$ operations, where $\overline{n}$ is the *average* number of nonzero entries per data point by first computing $\mathbf{i} = (\lambda \ell I)\mathbf{x}$ and $\mathbf{t} = A^T\mathbf{x}$, then using

$$(AA^T + \lambda \ell I)\mathbf{x} = A\mathbf{t} + \mathbf{i}.$$

As an example of the speed advantages offered by this technique, we compared SVMs and RLSCs trained using the Conjugate Gradient approach on the `20newsgroups` data set (described in Section 7.5 below) containing 15,935 data points. It took 10,045 seconds (on a Pentium IV running at 1.5 GhZ) to train 20 one-vs-all SVMs, and only 1,309 seconds to train the equivalent RLSC classifiers.[3]. At test time, both SVM and RLSC yield a linear hyperplane, so testing times are equivalent.

It is also worth noting that the same techniques that can be used to train linear RLSC systems very quickly can be used to train linear SVMs quickly as well. Using an interior point approach to the quadratic optimization problem [22], we can solve the SVM training problem as a sequence of linear systems of the form $(K + dI)\mathbf{x} = \mathbf{b}$, where $d$ changes with each iteration. Each system can be solved using the techniques discussed above. In this case, the RLSC system will be faster than the SVM system by a factor of the number of systems the SVM has to solve (as the RLSC approach only needs to solve a single system); whether this approach to linear SVM will be faster than the current approaches is an open question.

## 7.5 Empirical Performance of RLSC

Fung and Mangasarian [15] perform numerous experiments on benchmark datasets from the UCI Machine Learning Repository [24], and conclude that RLSC performs as well as SVMs. We consider two additional, large-scale examples. The first example is

[3]The SVMs were trained using the highly-optimized SvmFu system (`http://fpn.mit.edu/SvmFu`), and kernel products were stored between different SVMs. The RLSC code was written in Matlab.

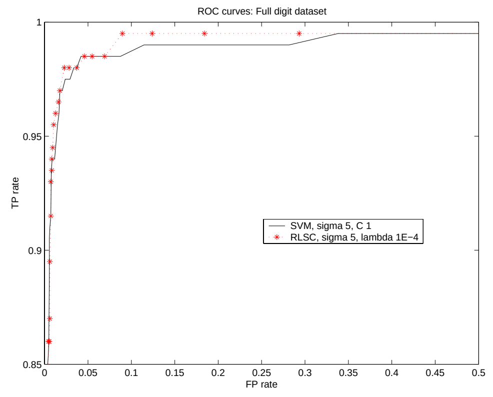

Figure 7.1: ROC curves for SVM and RLSC on the `usps` dataset.

| | 800 | | 250 | | 100 | | 30 | |
|---|---|---|---|---|---|---|---|---|
| | SVM | RLSC | SVM | RLSC | SVM | RLSC | SVM | RLSC |
| OVA | 0.131 | 0.129 | 0.167 | 0.165 | 0.214 | 0.211 | 0.311 | 0.309 |
| BCH 63 | 0.125 | 0.129 | 0.164 | 0.165 | 0.213 | 0.213 | 0.312 | 0.316 |

Table 7.1: A comparison of SVM and RLSC accuracy on the `20newsgroups` multiclass classification task. The top row indicates the number of documents/class used for training. The left column indicates the multiclass classification scheme. Entries in the table are the fraction of misclassified documents.

the US Postal Service handwritten database (referred to here as `usps`), consisting of 7,291 training and 2,007 testing points; the task we consider is to discriminate images of the digit “4” from images of other digits. We first optimized the parameters (C and $\sigma$) for the SVM, then optimized the $\lambda$ parameter of RLSC using the same $\sigma$ ($\sigma = 5$). In Figure 7.1, we see that the full RLSC performed as well or better than the full SVM across the entire range of the ROC curve.

The second example is a pair of multiclass text categorization tasks, referred to as `20Newsgroups` and `Sector105`. Rennie and Rifkin [25] used linear SVMs on these datasets, using both a one-vs-all and a BCH coding scheme for combining the binary classifiers into a multiclass classification system, obtaining the best published results on both datasets. The datasets were split into nested training and testing sets 10 times; details of the preprocessing scheme are available in [25]. Rifkin [21] used linear RLSC with $\lambda\ell = 1$ (solved using Conjugate Gradient) instead of SVMs; the experimental setup is otherwise identical. Tables 7.1 and 7.2 give the average accuracy of the SVM and RLSC schemes, for differing numbers of training documents per class. In all cases, the accuracy of the RLSC scheme is essentially identical to that of the SVM scheme; we note again that these results are better than all previously published results on these datasets. Given that linear RLSC can be trained very fast, it seems that linear RLSC shows great promise as a method of choice for very large-scale text categorization problems.

| | 52 | | 20 | | 10 | | 3 | |
|---|---|---|---|---|---|---|---|---|
| | SVM | RLSC | SVM | RLSC | SVM | RLSC | SVM | RLSC |
| OVA | 0.072 | 0.066 | 0.176 | 0.169 | 0.341 | 0.335 | 0.650 | 0.648 |
| BCH 63 | 0.067 | 0.069 | 0.176 | 0.178 | 0.343 | 0.344 | 0.653 | 0.654 |

Table 7.2: A comparison of SVM and RLSC accuracy on the `sector105` multiclass classification task. The top row indicates the number of documents/class used for training. The left column indicates the multiclass classification scheme. Entries in the table are the fraction of misclassified documents.

## 7.6 Approximations to the RLSC Algorithm

Although the RLSC algorithm is conceptually simple and yields high accuracy, for general nonlinear classification, it is often many orders of magnitude slower than an SVM. For this reason, we explore here the idea of solving approximations to RLSC. Hopefully, these approximations will yield a vast increase in speed with very little loss in accuracy.

Several authors have suggested the use of low-rank approximations to the kernel matrix in order to avoid explicit storage of the entire kernel matrix [19, 26, 20, 27, 22]. These techniques can be used in a variety of methods, including Regularized Least Squares Classification, Gaussian Process regression and classification, and interior point approaches to Support Vector Machines.

The approaches all rely on choosing a subset of $m$ of the training points (or a subset of size $m$, abusing notation slightly) representing those points exactly in the Hilbert space, and representing the remaining points approximately as a linear combination of the points in $m$. The methods differ in the approach to choosing the subset, and in the matrix math used to represent the hypothesis. Both Smola and Schölkopf [19] and Williams and Seeger [20] suggest the use of the approximate kernel matrix

$$\widetilde{K} = K_{\ell m} K_{mm}^{-1} K_{m\ell}.$$

The assumption is that $m$ is small enough so that $K_{mm}$ is invertible. If we use a method that does not need the entries of $\widetilde{K}$, but only needs to multiply vectors by $\widetilde{K}$, we can form the matrix-vector produce by taking advantage of the $\widetilde{K}$'s representation as three separate matrices:

$$\begin{aligned}\widetilde{K}\mathbf{x} &= (K_{\ell m} K_{mm}^{-1} K_{m\ell})\mathbf{x} \\ &= (K_{\ell m}(K_{mm}^{-1}(K_{m\ell}\mathbf{x}))).\end{aligned}$$

This approximation is known as the *Nyström approximation* and is justified theoretically in [20]. If we approximate the kernel matrix using a subset of size $m$, we can show that we are actually approximating the first $m$ eigenfunctions of the integral operator induced by the kernel (evaluated at all the data points). The quality of this approximation will of course depend on the rate of decay of the eigenvalues of the kernel matrix.

The two sets of authors differ in their suggestions as to how to choose the subset. Williams and Seeger simply pick their subset randomly. Smola and Schölkopf suggest a number of greedy approximation methods that iteratively reduce some measure of the difference between $K$ and $\widetilde{K}$, such as the trace of $K - \widetilde{K}$. They also suggest approximations to the full greedy approach in order to speed up their method. However, all their methods are likely to involve computing nearly all of the entries of $K$ over the course of their operation (for reasonable subset sizes), implying that forming a matrix approximation in this manner will already be slower than training an SVM.

It is easy to show that (however $m$ is selected) $\widetilde{K}$ has the property that $\widetilde{K}_{mm} = K_{mm}$, $\widetilde{K}_{m\ell} = K_{m\ell}$, and (trivially by symmetry) $\widetilde{K}_{\ell m} = K_{\ell m}$. In other words, the approximated kernel product $\widetilde{K}(\mathbf{x_1}, \mathbf{x_2})$ for a pair of examples $\mathbf{x_1}$ and $\mathbf{x_2}$ will be exact if either of $\mathbf{x_1}$ or $\mathbf{x_2}$ are in $m$. It is also easy to show that $\widetilde{K}_{nn} = K_{\ell m} K_{mm}^{-1} K_{m\ell}$.

Inverting the matrix $K_{mm}$ explicitly takes $O(m^3)$ steps formally, and in practice, performing the inversion may be ill-conditioned, so this approach does not not seem to be a good choice for real-world applications.

Fine and Scheinberg [22] instead suggest the use of an incomplete Cholesky factorization. Noting that a positive semidefinite kernel matrix can always be represented as $K = R^T R$ where $R$ is an upper-triangular matrix, the incomplete Cholesky factorization is formed by using only the first $m$ rows of $R$. Fine and Scheinberg cleverly choose the subset on the fly — at each iteration, they pivot a pair of rows of the matrix so that the largest diagonal element becomes the next element in the subset. The total running time of their approach is only $O(mk^2)$.[4] In terms of the feature space, this pivoting technique is equivalent to, at each step, adding to the subset the data point which is most poorly represented by the current subset. Although the final matrix contains only $m$ nonzero rows, it is still upper-triangular, and we write

$$\widetilde{K} = R_m^T R_m.$$

However, there is a serious problem with this approach. Although Fine and Scheinberg claim that their method "takes the full advantage of the greedy approach for for the best reduction in the approximation bound $\operatorname{tr}(\Delta Q)$", this is not the case. To reduce the approximation bound, we need to consider which element not in the basis will best represent (the currently unrepresented portion of) all remaining non-basis elements. This is what Smola and Schölkopf attempt in [19], at too large a computational cost. The Fine and Scheinberg approach, in contrast, adds to the basis the element which is most poorly represented by the current basis elements. If we believe that the trace of $K - \widetilde{K}$ is a useful measure of the quality of the approximation, this turns out to be a very poor choice, at least for Gaussian kernels. In particular, on two different datasets (see Section 7.6.2), we find that the portion of the trace accounted for by the Fine and Scheinberg approximation is consistently smaller than the portion of the trace accounted for by selecting the subset at random. This is not too hard to explain. Because the Fine and Scheinberg approach adds the most poorly represented element to the basis, under the Gaussian kernel, it will tend to add outliers — data points that are far from any other data point. The trace would be reduced much more by adding elements which are not as far away, but which are themselves close to a large number

[4]We have not formally defined "operations" (is a multiplication more expensive than an addition or a memory access?), but it is clear that the constant involved in this approach is small compared to the methods suggested in [19].

of additional points. Speaking informally, we want to add the points which are centers of clusters to the basis, not point which are outliers.

We now consider the use of a low-rank approximation, obtained by any of the above means, in RLSC.

### 7.6.1 Low-rank approximations for RLSC

The most obvious approach (and indeed, the one suggested in [20] in the context of Gaussian process classification and [22] in the context of Support Vector Machine classification) is to simply use the matrix $\widetilde{K}$ in place of the original $K$, resulting in the system of linear equations:

$$(\widetilde{K} + \lambda \ell I) = \mathbf{y}.$$

These equations can be solved using the conjugate gradient method, taking advantage of the factorization of $\widetilde{K}$ to avoid having to store an $\ell$-by-$\ell$ matrix. From a machine learning standpoint, this approach consists of taking a subset of the points, estimating the kernel values at the remaining points, then treating the estimate as correct and solving the original problem over all the data points. In practice, we found that this worked quite poorly, because the matrix eigenvalues do not decay sufficiently quickly (see Section 7.6.2).

Instead, we consider the following modified *algorithm*, alluded to (among other places) in [19]. We minimize the empirical risk over all points, but allow the $c_i$ to be nonzero only at a specific subset of the points (identical to the points used in the low-rank kernel approximation). This leads to the following modified problem:

$$\begin{aligned}
&\min F(\mathbf{c_m}) \\
&= \min_{\mathbf{c_m} \in \mathbb{R}^m} \quad \frac{1}{\ell}(\mathbf{y} - K_{\ell m}\mathbf{c_m})^T(\mathbf{y} - K_{\ell m}\mathbf{c_m}) + \lambda \mathbf{c_m}^T K \mathbf{c_m} \\
&= \min_{\mathbf{c_m} \in \mathbb{R}^m} \quad \frac{1}{2\ell}\left(\mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T K_{\ell m}\mathbf{c_m} + \mathbf{c}^T K_{m\ell} K_{\ell m}\mathbf{c}\right) + \frac{\lambda}{2}\mathbf{c_m}^T K \mathbf{c_m}.
\end{aligned}$$

We take the derivative with respect to $\mathbf{c_m}$ and set it equal to zero:

$$\begin{aligned}
\nabla F_{\mathbf{c_m}} &= \frac{1}{\ell}\left(K_{m\ell}\mathbf{y} + K_{m\ell}K_{\ell m}\mathbf{c_m}\right) + \lambda K_{mm}\mathbf{c_m} = 0 \\
\Longrightarrow \quad & (K_{m\ell}K_{\ell m} + K_{mm}\lambda\ell)\mathbf{c_m} = K_{m\ell}\mathbf{y}.
\end{aligned}$$

We see that when we only allow a subset of size $m$ points to have nonzero coefficients in the expansion, we can solve a $m$ by $m$ system of equations rather than an $\ell$-by-$\ell$ system. As with the standard full RLSC, we were able to find the system of equations that defines the optimal solution by setting the derivative equal to zero. Again, suppose for the sake of argument we decide to derive a "dual" problem. We introduce a vector of dual variables $\mathbf{u}$ — it is important to note that this vector is of length $\ell$, not $m$. We form the Lagrangian:

$$L(\mathbf{c_m}, \xi, \mathbf{u}) = \frac{1}{2\ell}\xi^T\xi + \frac{\lambda}{2}\mathbf{c_m}^T K_{mm}\mathbf{c_m} - \mathbf{u}^T(K_{\ell m}\mathbf{c_m} - \mathbf{y} - \xi).$$

We take the derivative with respect to $\xi$:

$$\begin{aligned} \frac{\partial L}{\partial \xi} &= \frac{1}{\ell}\xi + \mathbf{u} = 0 \\ \Longrightarrow \quad \xi &= -\ell \mathbf{u} \end{aligned}$$

We take the derivative with respect to $\mathbf{c_m}$:

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{c_m}} &= \lambda K_{mm}\mathbf{c_m} - K_{m\ell}\mathbf{u} = 0 \\ \Longrightarrow \quad \mathbf{c_m} &= \frac{1}{\lambda} K_{mm}^{-1} K_{m\ell}\mathbf{u}. \end{aligned}$$

Substituting these equations into $L$, we reduce the Lagrangian to:

$$\begin{aligned} & L(\mathbf{u}) \\ = \;& \frac{\ell}{2}\mathbf{u}^T\mathbf{u} + \frac{1}{2\lambda}\mathbf{u^T} K_{\ell m} K_{mm}^{-1} K_{m\ell}\mathbf{u} - \frac{1}{\lambda}\mathbf{u^T} K_{\ell m} K_{mm}^{-1} K_{m\ell}\mathbf{u} + \mathbf{u}^T\mathbf{y} - \ell \mathbf{u}^T\mathbf{u} \\ = \;& \frac{\ell}{2}\mathbf{u}^T\mathbf{u} + \frac{1}{2\lambda}\mathbf{u^T}\widetilde{K}\mathbf{u} - \frac{1}{\lambda}\mathbf{u^T}\widetilde{K}\mathbf{u} + \mathbf{u}^T\mathbf{y} - \ell\mathbf{u}^T\mathbf{u} \\ = \;& -\frac{\ell}{2}\mathbf{u}^T\mathbf{u} - \frac{1}{2\lambda}\mathbf{u^T}\widetilde{K}\mathbf{u} + \mathbf{u}^T\mathbf{y}. \end{aligned}$$

Setting the derivative to zero yields

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{u}} &= -\ell\mathbf{u} - \frac{1}{\lambda}\widetilde{K}\mathbf{u} + \mathbf{y} = 0 \\ \Longrightarrow \quad (\widetilde{K} + \lambda\ell I)\mathbf{u} &= \lambda\mathbf{y} \end{aligned}$$

If we solve this system for $\frac{\mathbf{u}}{\lambda}$, *we are solving exactly the same system of equations as if we had used the Nyström approximation at the subset $m$ directly.* However, in order to find the optimal solution, we need to recover the vector $\mathbf{c_m}$ using the equation

$$\mathbf{c_m} = \frac{1}{\lambda} K_{mm}^{-1} K_{m\ell}\mathbf{u}. \tag{7.7}$$

To summarize, we suggest that instead of using the matrix $\widetilde{K}$ directly in place of $K$, we consider a modified problem in which we require our function to be expressed in terms of the points in the subset $m$. This leads to an algorithm that doesn't directly involve $\widetilde{K}$, but uses only the component matrices $K_{mm}$, $K_{\ell m}$ and $K_{m\ell}$. Although it is not strictly necessary, we can take the Lagrangian dual of this problem, at which point we solve a system that is identical to the original, full RLSC problem with $K$ replaced with $\widetilde{K}$. However, we do not use the resulting $\mathbf{u}$ vector directly, instead recovering the $c_m$ by means of (7.7).

### 7.6.2 Nonlinear RLSC application: image classification

To test the ideas discussed above, we compare various approaches to nonlinear RLSC on two different datasets. The first dataset is the US Postal Service handwritten database, used in [20] and communicated to us by the authors, and referred to here as `usps`. This dataset consists of 7,291 training and 2,007 testing points; the task we consider is to discriminate images of the digit "4" from images of other digits. The training set contains 6,639 negative examples and 652 positive examples. The testing set contains 1,807 negative examples and 200 positive example. The second data set is a face recognition data set that has been used numerous times at the Center for Biological and Computational Learning at MIT [28, 29], referred to here as `faces`. The training set contains 2,429 faces and 4,548 non-faces. The testing set contains 472 faces and 23,573 non-faces.

Although RLSC on the entire dataset will produce results essentially equivalent to those of the SVM [21], they will be substantially slower. We therefore turn to the question of whether an *approximation* to RLSC results can produce results as good as the full RLSC algorithm. We consider three different approximation methods. The first method, which we call `subset`, involves merely selecting (at random) a subset of the points, and solving a reduced RLSC problem on only these points. The second method, which we call `rectangle`, is the primary algorithm discussed in Section 7.6.1: we choose a subset of the points, allow only those points to have nonzero coefficients in the function expansion, but minimize the loss over all points simultaneously. The third method is the `Nyström` method, also discussed briefly in Section 7.6.1 and presented more extensively in [20] (and in a slightly different context in [22]): in this method we choose a subset of the points, use those points to approximate the entire kernel matrix, and then solve the full problem using this approximate kernel matrix. In all experiments, we try four different subset sizes (1,024, 512, 256, and 128 for the `usps` dataset, and 1,000, 500, 250 and 125 for the `faces` dataset), and the results presented are the average of ten independent runs.

The results for the `usps` data set are given in Figure 7.2. We see that `rectangle` performs best, followed by `subset`, followed by `nystrom`. We suggest in passing that the extremely good results reported for `nystrom` in [20] may be a consequence of looking only at the error rate (no ROC curves are provided) for a problem with a highly skewed distribution (1,807 negative examples, 200 positive examples). For `rectangle` performance is very good at both the 1,024 and 512 sizes, but degrades noticeably for 256 samples. The results for the `faces` dataset, shown in Figure 7.3, paint a very similar picture. Note that the overall accuracy rates are much lower for this dataset, which contains many difficult test examples.

In all the experiments described so far, the subset of points was selected at random. Fine and Scheinberg [22] suggests selecting the points iteratively using an optimal greedy heuristic: at each point, the example is selected which will minimize the trace of the difference between the approximate matrix and the true matrix. Because the method simply amounts to running an incomplete Cholesky factorization for some number of steps (with pivoting at each step), we call this method `ic`. As mentioned earlier, if we believe that smaller values of $\mathrm{tr}(K - \widetilde{K})$ are indicative of a better approximation, this method appears to produce a worse approximation than choosing the subset at random (see Figure 7.4 and Figure 7.5). As pointed out by Scheinberg in personal communication, a smaller value of $\mathrm{tr}(K - \widetilde{K})$ is not necessarily indicative of a better matrix for learning, so we conducted experiments comparing `ic` to choosing a subset of the data at random randomly. Figure 7.6 shows the results for the `usps`

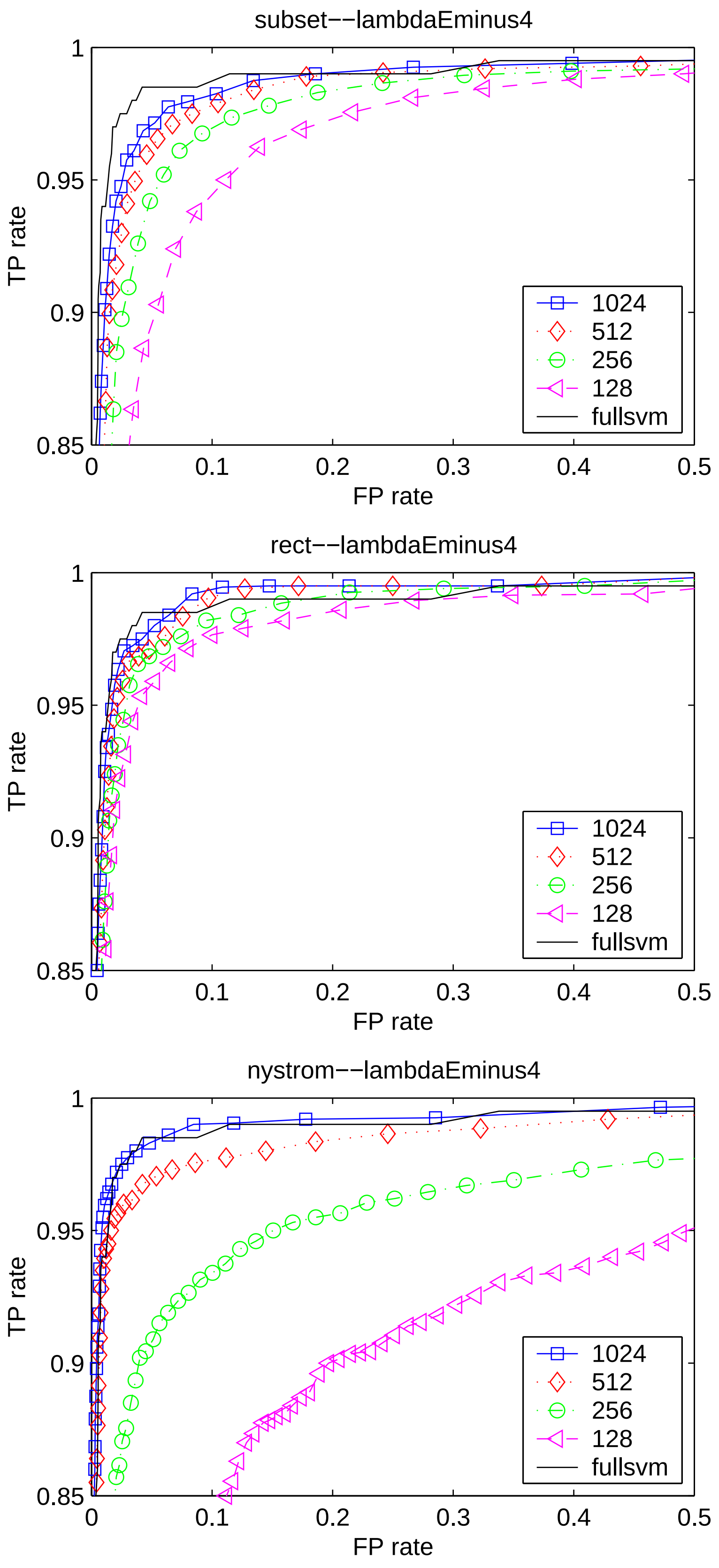

Figure 7.2: A comparison of the `subset`, `rectangle` and `nystrom` approximations to RLSC on the `usps` dataset. Both SVM and RLSC used $\sigma = 5$.

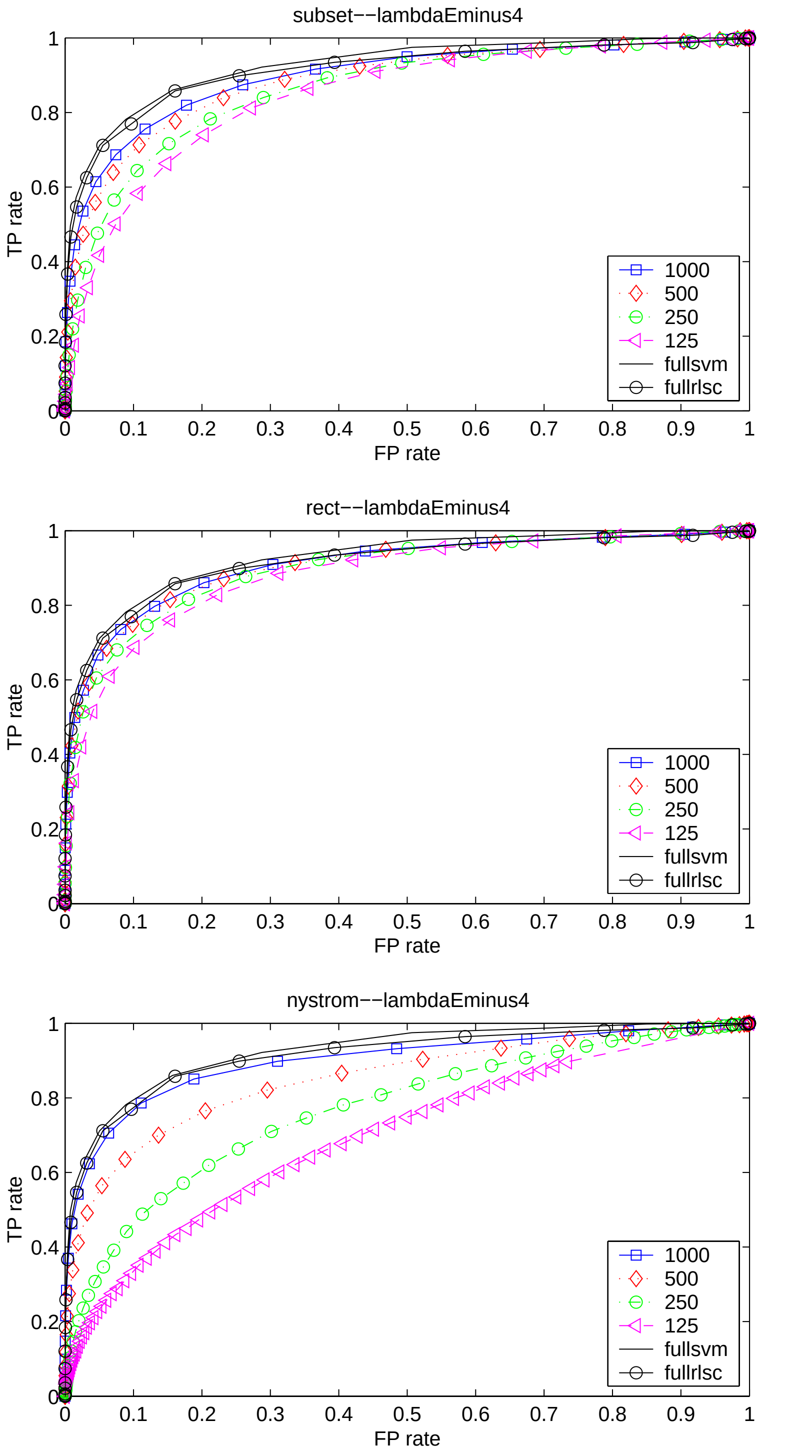

Figure 7.3: A comparison of the `subset`, `rectangle` and `nystrom` approximations to RLSC on the `faces` dataset. SVM used $\sigma = 5$, RLSC used $\sigma = 2$.

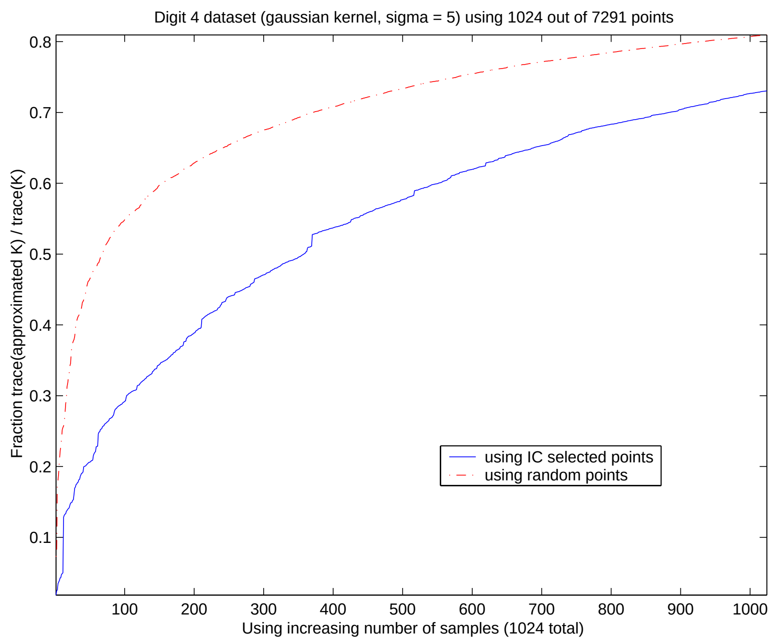

Figure 7.4: A comparison between the portion of the trace of $K$ accounted for by using a subset of the points chosen using `ic` and a random subset, for the `usps` dataset.

data, and Figure 7.7 shows the results for the `faces` data. In all these experiments, we compare the average performance of the ten classifiers trained on a random subset to the performance of a single classifier trained on the subset of the same size obtained using the `ic` method. In these experiments, it does not seem that selecting the "optimal" subset using the `ic` method improves performance on learning problems. In the interests of computational efficiency, we may be better off simply choosing a random subset of the data.

The experiments contained in this section are somewhat preliminary. In particular, although using subsets of size 128 is faster than running the SVM, using subsets of size 1024 are already slower. Further experiments in which both techniques are optimized more thoroughly are necessary to fully answer the question of whether these approximations represent a viable approach to solving large-scale optimization problems.

## 7.7 Leave-one-out Bounds for RLSC

Leave-one-out bounds allow us to estimate the generalization error of a learning system when we do not have access to an independent test set. In this section, we define $f_S$ to be the function obtained when the entire data set is used, and $f_{S^i}$ to be the function obtained when the $i$th data point is removed and the remaining $\ell - 1$ points are used. We use $G$ to denote $(K + \lambda \ell I)^{-1}$. Classical results (for example see [30]) yield the following exact computation of the leave-one-out value:

$$y_i - f_{S^i}(\mathbf{x}_i) = \frac{y_i - f_S(\mathbf{x}_i)}{1 - G_{ii}}.$$

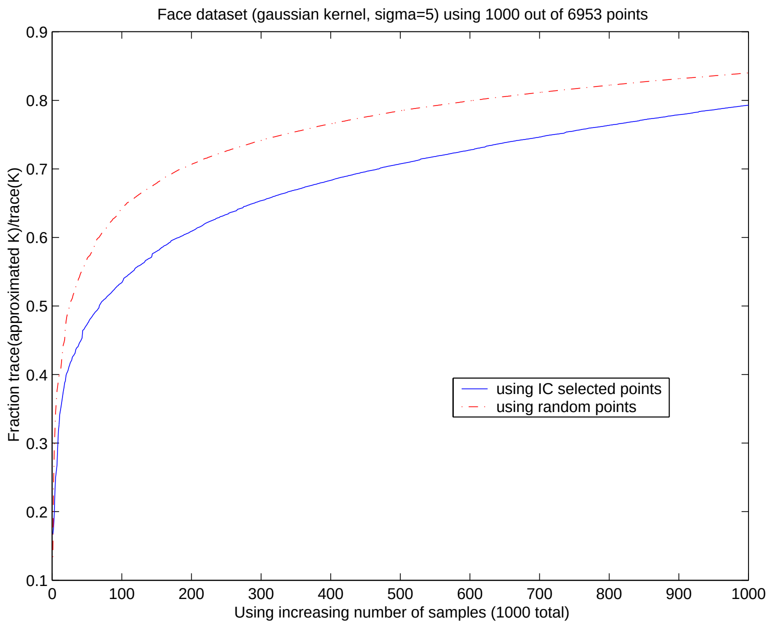

Figure 7.5: A comparison between the portion of the trace of $K$ accounted for by using a subset of the points chosen using `ic` and a random subset, for the `faces` dataset.

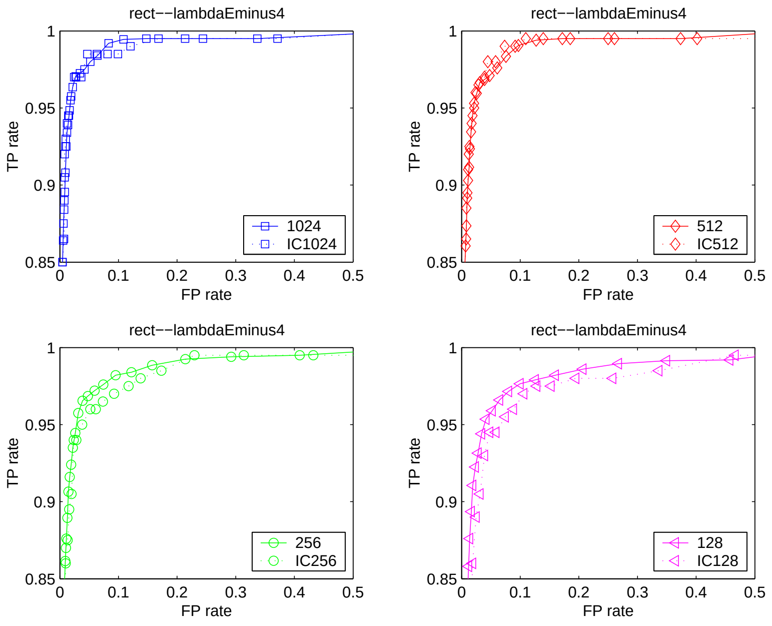

Figure 7.6: A comparison between random subset selection and the `ic` method, classifying the `usps` dataset using the `rectangle` method.

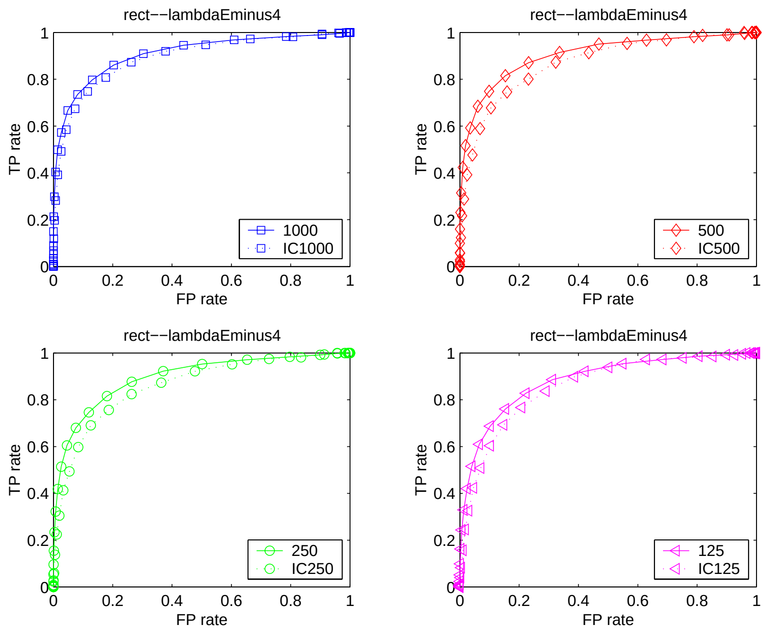

Figure 7.7: A comparison between random subset selection and the `ic` method, classifying the `faces` dataset using the `rectangle` method.

Unfortunately, using this equation requires access to the inverse matrix $G$, which is often computationally intractable and numerically unstable.

An alternative approach, introduced by Craven and Wahba [31], is known as the *generalized approximate cross-validation*, or GACV for short. Instead of actually using the entries of the inverse matrix $G$ directly, an approximation to the leave-one-out value is made using the *trace* of $G$:

$$y_i - f_{S^i}(\mathbf{x}_i) \approx \frac{y_i - f_S(\mathbf{x}_i)}{1 - \frac{1}{\ell}\mathrm{tr}(G)}$$

This approach, while being only an approximation rather than an exact calculation, has an important advantage. The trace of a matrix is the sum of its eigenvalues. If the eigenvalues of $K$ are known, the eigenvalues of $G$ can be computed easily for any value of $\lambda$, and we can then easily select the value of $\lambda$ which minimizes the leave-one-out bound. Unfortunately, computing all the eigenvalues of $K$ is in general much too expensive, so again, this technique is not practical for large problems.

Jaakkola and Haussler introduced an interesting class of simple leave-one-bounds [32] for kernel classifiers. In words, their bound says that if a given training point $\mathbf{x}$ can be classified correctly by $f_S$ *without using the contribution of* $\mathbf{x}$ *to* $f_S$, then $\mathbf{x}$ cannot be a leave-one-out error — $f_{S^i}(\mathbf{x}_i) \geq 0$. Although the Jaakkola and Haussler bound does not apply directly to RLSC (because the $c_i$ are not sign-constrained), Rifkin [21] showed that their bound is valid for RLSC via a slightly more involved proof, and that the number of leave-one-out errors is bounded by

$$|\mathbf{x}_i : y_i \sum_{j \neq i} c_j K(\mathbf{x}_i, \mathbf{x}_j) \leq 0|$$

This bound can be computed directly given the $c_i$; no retraining is required. The same bound holds for SVMs (without an unregularized $b$ term), via the Jaakkola and Haussler proof. However, there is a simple geometric condition in RLSC that allows us to provide a more elegant bound.

Using (7.5) and (7.6), we see that

$$\xi_i = -\ell\lambda c_i.$$

Combining this with the definition of the $\xi_i$,

$$\xi_i = f(\mathbf{x}_i) - y_i,$$

we find that

$$c_i = \frac{y_i - f(\mathbf{x}_i)}{\ell\lambda}.$$

Using this, we can eliminate $c_i$ from the bound, arriving at

$$\begin{aligned} & |\mathbf{x}_i : y_i \left( f(\mathbf{x}_i) - \left( \frac{y_i - f(\mathbf{x}_i)}{\ell\lambda} \right) K(\mathbf{x}_i, \mathbf{x}_i) \right) \leq 0| \\ \Longrightarrow \quad & |\mathbf{x}_i : y_i \left( (1 + \frac{K(\mathbf{x}_i, \mathbf{x}_i)}{\ell\lambda}) f(\mathbf{x}_i) - \frac{y_i}{\ell\lambda} K(\mathbf{x}_i, \mathbf{x}_i) \right) \leq 0|. \end{aligned}$$

Supposing $y_i = 1$, for some $i$, the condition reduces to

$$\begin{aligned} & (1 + \frac{K(\mathbf{x}_i, \mathbf{x}_i)}{\ell\lambda}) f(\mathbf{x}_i) \leq \frac{1}{\ell\lambda} K(\mathbf{x}_i, \mathbf{x}_i) \\ \Longrightarrow \quad & f(\mathbf{x}_i) \leq \frac{K(\mathbf{x}_i, \mathbf{x}_i)}{\ell\lambda + K(\mathbf{x_i}, \mathbf{x_i})}. \end{aligned}$$

Reasoning similarly for the case where $y_i = -1$, we find that we can bound the number of leave-one-out errors for RLSC via

$$|\mathbf{x}_i : y_i f(\mathbf{x}_i) \leq \frac{K(\mathbf{x_i}, \mathbf{x_i})}{K(\mathbf{x_i}, \mathbf{x_i}) + \ell\lambda}|.$$

In this form, there is a nice connection to the recent algorithmic stability work of Bousquet and Elisseef [8], where it is shown that Tikhonov regularization algorithms have "stability" of $O(\frac{1}{\ell\lambda})$ — very loosely, that when a single data point is removed from the training set, the change in the function obtained is $O(\frac{1}{\lambda\ell})$. For RLSC, we are able to exploit the geometry of the problem to obtain a leave-one-out bound that directly mirrors the stability theory results.

# Bibliography

[1] T. Evgeniou, M. Pontil and T. Poggio, Regularization networks and support vector machines, *Advances in Computational Mathematics* **13** (2000) 1–50.

[2] F. Cucker and S. Smale, On the mathematical foundations of learning, *Bulletin of the American Mathematical Society* **39** (2002) 1–49.

[3] A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-posed problems*, W. H. Winston, Washington D.C. (1977).

[4] G. Wahba, *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*, Society for Industrial & Applied Mathematics (1990).

[5] N. Aronszajn, Theory of reproducing kernels, *Transactions of the American Mathematical Society* **68** (1950) 337–404.

[6] G. Wahba, Support vector machines, reproducing kernel hilbert spaces and the randomized gacv, Technical Report 984rr, University of Wisconsin, Department of Statistics (1998).

[7] V.N. Vapnik, *Statistical Learning Theory*, John Wiley & Sons (1998).

[8] O. Bousquet and André Elisseeff, Stability and generalization, *Journal of Machine Learning Research* **2** (2002) 499–526.

[9] F. Girosi, An equivalence between sparse approximation and support vector machines, *Neural Computation* **10**(6) (1998) 1455–1480.

[10] B. Schölkopf, R. Herbrich and Alex J. Smola, A generalized representer theorem. In *Proceedings of the 14th Annual Conference on Computational Learning Theory* (2001) 416–426.

[11] I. Schönberg, Spline functions and the problem of graduation, *Proceedings of the National Academy of Science* (1964) 947–950.

[12] M. Bertero, T. Poggio and V. Torre, Ill-posed problems in early vision, *Proceedings of the IEEE* **76** (1988) 869–889.

[13] T. Poggio and F. Girosi, A theory of networks for approximation and learning, Technical Report A.I. Memo No. 1140, C.B.C.L Paper No. 31, Massachusetts Institute of Technology, Artificial Intelligence Laboratory and Center for Biological and Computational Learning, Department of Brain and Cognitive Sciences, July (1989).

[14] T. Poggio and F. Girosi, Networks for approximation and learning, *Proceedings of the IEEE* **78**(9) (1990) 1481–1497.

[15] G. Fung and O.L. Mangasarian, Proximal support vector machine classifiers, Technical report, Data Mining Institute (2001).

[16] J.A.K. Suykens and J. Vandewalle, Least squares support vector machine classifiers, *Neural Processing Letters* **9**(3) (1999) 293–300.

[17] Y. Nesterov and A. Nemirovskii, *Interior Point Polynomial Algorithms in Convex Programming*, volume 13 of *Studies In Applied Mathematics.* SIAM (1994).

[18] T. Joachims, Making large-scale svm learning practical, Technical Report LS VIII-Report, Universität Dortmund (1998).

[19] A.J. Smola and B. Schölkopf, Sparse greedy matrix approximation for machine learning, In *Proceedings of the International Conference on Machine Learning* (2000).

[20] C.K.I. Williams and M. Seeger, Using the Nyström method to speed up kernel machines, In *Neural Information Processing Systems* (2000).

[21] R.M. Rifkin, *Everything Old Is New Again: A Fresh Look at Historical Approaches to Machine Learning*, PhD thesis, Massachusetts Institute of Technology (2002).

[22] S. Fine and K. Scheinberg, Efficient application of interior point methods for quadratic problems arising in support vector machines using low-rank kernel representation, *Submitted to Mathematical Programming* (2001).

[23] J.R. Shewchuk, An introduction to the conjugate gradient method without the agonizing pain, http://www-2.cs.cmu.edu/∼jrs/jrspapers.html (1994).

[24] C.J. Merz and P.M. Murphy, Uci repository of machine learning databases, http://www.ics.uci.edu/∼mlearn/MLRepository.html (1998).

[25] J. Rennie and R. Rifkin, Improving multiclass classification with the support vector machine. Technical Report A.I. Memo 2001-026, C.B.C.L. Memo 210, MIT Artificial Intelligence Laboratory, Center for Biological and Computational Learning (2001).

[26] Y.-J. Lee and O. Mangasarian, Rsvm: Reduced support vector machines, In *SIAM International Conference on Data Mining* (2001).

[27] T. Van Gestel, J. Suykens, B. De Moor, and J. Vandewalle. Bayesian inference for ls-svms on large data sets using the Nyström method. In *International Joint Conference on Neural Networks* (2002).

[28] B. Heisele, T. Poggio, and M. Pontil, Face detection in still gray images, Technical Report A.I. Memo No. 2001-010, C.B.C.L. Memo No. 197, MIT Center for Biological and Computational Learning (2000).

[29] M. Alvira and R. Rifkin, An empirical comparison of snow and svms for face detection, Technical Report A. I. Memo No. 2001-004, C.B.C.L. Memo No. 193, MIT Center for Biological and Computational Learning (2001).

[30] P.J. Green and B.W. Silverman, *Nonparametric Regression and Generalized Linear Models*, Number 58 in Monographs on Statistics and Applied Probability. Chapman & Hall (1994).

[31] P. Craven and G. Wahba, Smoothing noisy data with spline functions, *Numerical Mathematics* **31** (1966) 377–390.

[32] T. Jaakkola and D. Haussler, Probabilistic kernel regression models, In *Advances in Neural Information Processing Systems 11* (1998).