Journal of Machine Learning Research 5 (2004) 101-141 Submitted 4/03; Revised 8/03; Published 1/04

In Defense of One-Vs-All Classification

Ryan Rifkin RIF@ALUM.MIT.EDU
Honda Research Institute USA
145 Tremont Street
Boston, MA 02111-1208, USA

Aldebaro Klautau A.KLAUTAU@IEEE.ORG
UC San Diego
La Jolla, CA 92093-0407, USA

Editor: John Shawe-Taylor

Abstract

We consider the problem of multiclass classification. Our main thesis is that a simple "one-vs-all" scheme is as accurate as any other approach, assuming that the underlying binary classifiers are well-tuned regularized classifiers such as support vector machines. This thesis is interesting in that it disagrees with a large body of recent published work on multiclass classification. We support our position by means of a critical review of the existing literature, a substantial collection of carefully controlled experimental work, and theoretical arguments.

Keywords: Multiclass Classification, Regularization

1. Introduction

We consider the problem of multiclass classification. A training set consisting of data points belonging to N different classes is given, and the goal is to construct a function which, given a new data point, will correctly predict the class to which the new point belongs.[1]

Over the last decade, there has been great interest in classifiers that use *regularization* to control the capacity of the function spaces they operate in. These classifiers—the best-known example of which is the support vector machine (SVM) (Boser et al., 1992)—have proved extremely successful at binary classification tasks (Vapnik, 1998, Evgeniou et al., 2000, Rifkin, 2002). It therefore seems interesting to consider whether the advantages of regularization approaches for binary classifiers carried over to the multiclass situation.

One of the simplest multiclass classification schemes built on top of real-valued binary classifiers is to train N different binary classifiers, each one trained to distinguish the examples in a single class from the examples in all remaining classes. When it is desired to classify a new example, the N classifiers are run, and the classifier which outputs the largest (most positive) value is chosen. This scheme will be referred to as the "one-vs-all" or OVA

[1] In our framework, each data point is required to belong to a single class. We distinguish this from the case when there are more than two classes, but a given example can be a member of more than one class simultaneously. In the latter case, if the labels are independent, the problem very naturally decomposes into N *unlinked* binary problems, where the ith binary learner simply learns to distinguish whether or not an example is in class i. If the labels are dependent, then how best to perform multiclass classification is an interesting research problem, but is beyond the scope of this paper.

scheme throughout this paper. The one-vs-all scheme is conceptually simple, and has been independently discovered numerous times by different researchers.

One might argue that OVA is the first thing thought of when asked to come up with an approach for combining binary classifiers to solve multiclass problems. Although it is simple and obvious, the primary thesis of this paper is that the OVA scheme is extremely powerful, producing results that are often at least as accurate as other methods. This thesis seems quite innocuous and hardly worth writing a paper about, until one realizes that this idea is in opposition to a large body of recent literature on multiclass classification, in which a number of more complicated methods have been developed and their superiority over OVA claimed. These methods can be roughly divided between two different approaches—the "single machine" approaches, which attempt to construct a multiclass classifier by solving a *single* optimization problem (Weston and Watkins, 1998, Lee et al., 2001a,b, Crammer and Singer, 2001) and the "error correcting" approaches (Dietterich and Bakiri, 1995, Allwein et al., 2000, Crammer and Singer, 2002, Fürnkranz, 2002, Hsu and Lin, 2002), which use ideas from error correcting coding theory to choose a collection of binary classifiers to train and a method for combining the binary classifiers.

A substantial portion of this paper is devoted to a detailed review and discussion of this literature. What we find is that although a wide array of more sophisticated methods for multiclass classification exist, experimental evidence of the superiority of these methods over a simple OVA scheme is either lacking or improperly controlled or measured.

One scheme that is particularly worthy of attention is the "all-pairs", or AVA ("all-vs-all") scheme. In this approach, $\binom{N}{2}$ binary classifiers are trained; each classifier separates a pair of classes. This scheme, like the OVA scheme, has a simple conceptual justification, and can be implemented to train faster and test as quickly as the OVA scheme. Several authors have reported that the AVA scheme offers better performance than the OVA scheme (Allwein et al., 2000, Fürnkranz, 2002, Hsu and Lin, 2002). Our results disagree with the ones presented in all three of these papers, essentially because we feel their experiments were not as carefully controlled and reported as ours.

For an experiment to be carefully *controlled* means that a reasonable effort was made to find correct settings for the hyperparameters of the algorithm (in a way that does not involve mining the test set, obviously) and that the best available binary classifiers are used. This point is critical; it is easy to show (and many authors have) that if relatively weak binary learners are used, then a wide variety of clever methods for combining them will exploit the independence in the error rates of these weak classifiers to improve the overall result. In contrast, we demonstrate empirically in a substantial collection of well-controlled experiments that when well-tuned SVMs (or regularized least squares classifiers) are used as the binary learners, there is little to no independence in the errors of the binary classifiers, and therefore nothing to be gained from sophisticated methods of combination. The crucial question for the practitioner then becomes whether sophisticated methods of combining weak classifiers can achieve stronger results than a simple method of combining strong classifiers. The empirical evidence supports the notion that they cannot.

For an experiment to be carefully *reported* indicates that the results are presented in a way that is easy to understand, and that reasonable conclusions are drawn from them. This is obviously a subjective notion, but we wish to point out a specific area which we feel is problematic: the notion that a lower absolute error rate is strongly indicative of the superiority of a classifier *when these absolute error rates are very close.* In other words, we feel it is not appropriate simply to present results where the best-performing classifier has its score given in bold on each experiment, and the classifier with the most bold scores is declared the strongest, because this ignores the very real possibility (see for example Hsu and Lin, 2002) that on nearly all the experiments, the actual experimental differences are tiny. Therefore, it is worthwhile to assess statistically the relative performance of classification systems.

One possible test for comparing two schemes is McNemar's test (McNemar, 1947, Everitt, 1977) (see Appendix A). A difficulty with McNemar's test is that it is insensitive to the number of points that the two schemes *agree* on; directly related to this, McNemar's test simply tells *whether* or not two scores are statistically significantly different (according to the assumptions inherent in the test), but gives no indication of *how* different they are. For this reason, we advocate and use a simple bootstrap method for computing confidence intervals of the difference in performance between a pair of classifiers. This method is described in Appendix A.[2]

Additionally, in order to allow for comparisons by different researchers, we feel that it is crucial to present the actual error rates of various classification schemes, rather than (or in addition to) the relative differences in error between two schemes (see, for example, Dietterich and Bakiri, 1995, Crammer and Singer, 2001, 2002); this is particularly important given that different researchers are often unable to produce identical results using identical methodologies (see Appendix B).

In general, we are *not* stating that an OVA scheme will perform substantially better than other approaches. Instead, we are stating that it will perform *just as well* as these approaches, and therefore it is often to be preferred due to its computational and conceptual simplicity.

In Section 2 we describe support vector machines and regularized least squares classifiers, which are the binary classifiers used throughout this paper. In Section 3, we describe previous work in multiclass classification using binary classifiers. In Section 4, we present a large, carefully controlled and carefully measured set of experiments, as well as analysis of these experiments. In Section 5, we present theoretical arguments that help to indicate why OVA approaches perform well. Finally, in Section 6, we discuss our results and outline open questions. Note that notation which is used in Sections 4 and 5 is introduced in Section 3.2, as this seemed the most natural approach to the presentation.

It is our hope that this paper will be of equal interest to machine learning researchers and general practitioners who are actually faced with multiclass classification problems in

[2] Other statistical tests are of course possible. Dietterich (1998) compares a number of approaches to testing the statistical difference between classifiers. Dietterich derives and recommends a test known as a 5x2 CV test, suggesting that it is more powerful than McNemar's test while having essentially equivalent probability of incorrectly finding a distinction where none exists. We must confess that at the time our experiments were performed, we were not aware of this work. However, there are several advantages to our current protocol. The 5x2 CV test, like McNemar's test, only gives an estimate of whether two classifiers have different performance, whereas the bootstrap method we advocate naturally produces an estimate of the *size* of this difference. Additionally, in order to use the 5x2 CV test, it would have been necessary to use training sets whose size was exactly half the size of the total training set; this would have made comparisons to previous work (especially that of Fürnkranz, 2002) much more difficult.

engineering applications. In particular, we hope to demonstrate that for practical purposes, simple approaches such as one-vs-all classification work as well as more complicated schemes, and are therefore to be preferred.

2. Regularized Kernel Classifiers

A broad class of classification (and regression) algorithms can be derived from the general approach of Tikhonov regularization. In our case, we consider Tikhonov regularization in a reproducing kernel Hilbert space:

$$\min_{f \in \mathcal{H}} \sum_{i=1}^{\ell} V(f(\mathbf{x_i}), y_i) + \lambda ||f||_K^2.$$

Here, ℓ is the size of the training set $S \equiv \{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$. $V(f(\mathbf{x}, y))$ represents the cost we incur when the algorithm sees $\mathbf{x}$, predicts $f(\mathbf{x})$, and the actual value is y. The parameter λ is the "regularization parameter" which controls the tradeoff between the two conflicting goals of minimizing the training error and ensuring that f is smooth. A detailed discussion of RKHSs is beyond the scope of this paper (for details, see Evgeniou et al., 2000, and the references therein). For our purposes, there are essentially only two key facts about RKHS. The first is the "Representer Theorem" (Wahba, 1990), which states that under very general conditions on the loss function V, the solution to a Tikhonov minimization problem can be written as a sum of kernel products on the training set:

$$f(\mathbf{x}) = \sum_{i=1}^{\ell} c_i K(\mathbf{x_i}, \mathbf{x_j}). \tag{1}$$

The goal of a Tikhonov minimization procedure is to find the c_i. The other fact is that for functions represented in this form,

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

Given these facts, a range of different algorithms can be derived by choosing the function V.

If we choose $V(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2$, the so-called "square loss", the Tikhonov minimization problem becomes regularized least squares classification (RLSC): [3]

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

[3] Regularized least-squares classification is not a new algorithm. The mathematics of finding a linear function that minimizes the square loss over a set of data was first derived by Gauss (1823). The idea of regularization is apparent in the work of Tikhonov and Arsenin (1977), who used least-squares regularization to restore well-posedness to ill-posed problems. Schönberg's seminal article on smoothing splines (Schönberg, 1964) also used regularization. These authors considered regression problems rather than classification, and did not use reproducing kernel Hilbert spaces as regularizers.

In 1971, Wahba and Kimeldorf (1971) considered square-loss regularization using the norm in a reproducing kernel Hilbert space as a stabilizer. Only regression problems were considered in this work.

In 1989, Girosi and Poggio considered regularized classification and regression problems with the square loss (Girosi and Poggio, 1989, Poggio and Girosi, 1990). They used pseudodifferential operators as their stabilizers; these are essentially equivalent to using the norm in an RKHS.

If we choose $V(f(\mathbf{x}), y) = \max(1 - y_i f(\mathbf{x_i}), 0) \equiv (1 - y_i f(\mathbf{x_i}))_+$, the so-called "hinge loss", we arrive at the standard support vector machine:[4]

$$\begin{array}{lll} \min\limits_{\mathbf{c}\in\mathbb{R}^\ell, \xi\in\mathbb{R}^\ell} & C\sum_{i=1}^{\ell} \xi_i + \frac{1}{2}\mathbf{c}^T K\mathbf{c} & \\ \text{subject to :} & y_i(\sum_{j=1}^{\ell} c_j K(\mathbf{x_i}, \mathbf{x_j}) + b) \geq 1 - \xi_i & i = 1, \ldots, \ell, \\ & \xi_i \geq 0 & i = 1, \ldots, \ell. \end{array}$$

We note that RLSC and SVM are *both* instances of Tikhonov regularization, and will both have solutions in the form given by Equation 1. From the standpoint of theoretical bounds on the generalization of these algorithms using measures of the size of the function class such as covering numbers, the choice of the loss function is almost irrelevant and the two methods will provide very similar bounds (Vapnik, 1998, Bousquet and Elisseeff, 2002).

Intuitively, it seems that the square loss may be less well suited to classification than the hinge loss—if a point $\mathbf{x_i}$ is in the positive class ($y_i = 1$) and we observe $f(\mathbf{x_i}) = 5$, we pay nothing under the hinge loss but we pay $(5 - 1)^2 = 16$ under the square loss, the same penalty we would pay if $f(\mathbf{x_i})$ were -3. However, in practice, we have found that the *accuracy* of RLSC is essentially equivalent to that of SVMs (Rifkin, 2002), and substantial additional evidence of that claim will be presented in Section 4; while the performance differs substantially on a few data sets (in both directions), on most data sets the difference in the accuracy of the methods is very small.

For this reason, we believe that the choice between RLSC and SVMs should be made on the basis of computational tractability rather than accuracy. For linear kernels (or other cases where the kernel matrix K will be sparse or can be decomposed in a way that is known a priori), RLSC will be substantially faster than SVM both at training and test times. On the other hand, for a general nonlinear kernel, the first step in solving RLSC is computing the matrix K; for large problems, an SVM can be trained in less time than it takes to

These earlier works tended to belong to the statistical rather than the machine learning community. As such, the technique called RLSC in the present work was not given a name *per se* in these works.

More recently, the algorithm (or a minor variant) has been rediscovered independently by many authors who were not fully aware of the above literature. Saunders et al. (1998) rederives the algorithm as "kernel ridge regression"; he derives it by means of applying the "kernel trick" to ridge regression, rather than directly via regularization, and does not consider the use of this algorithm for classification. Mika et al. (1999) present a similar algorithm under the name kernel fisher discriminant, but in this work, the algorithm *without* regularization is presented as primary, with regularization added "to improve stability"; in our view, the regularization is central to both theory and practice. Fung and Mangasarian, under the name "proximal support vector machines" (Fung and Mangasarian, 2001b,a), and Suykens et al., under the name "least-squares support vector machines" (Suykens and Vandewalle, 1999a,b, Suykens et al., 1999), both derive essentially the same algorithm (we view the presence or absence of a bias term b in either the function or the cost function as a relatively minor detail) by modifying the cost function of SVMs. We strongly prefer to view *both* RLSC and SVM as instantiations of Tikhonov regularization, on an equal footing, rather than viewing RLSC as a "modified" SVM. Although it is regrettable to have to introduce yet another name for the same algorithm, we do not find any of the above names to be satisfactory. We believe that regularized least-squares classification is a highly appropriate name, as it draws attention to all the key features of the algorithm, and we hope (likely in vain) that future users of this algorithm will make use of this name.

[4] In order to arrive at the standard SVM, we modify our notation slightly, defining $C = \frac{1}{2\lambda\ell}$, and also add an unregularized bias term b to the formulation. Details of this derivation, as well as the derivation of RLSC, can be found in Rifkin's PhD thesis (Rifkin, 2002).

compute K. This is done by solving the SVM *dual problem*:

$$
\begin{array}{lll}
\max\limits_{\alpha\in\mathbb{R}^\ell} & \sum_{i=1}^{\ell}\alpha_i - \frac{1}{(2\lambda)^2}\alpha^T Q\alpha & \\
\text{subject to :} & \sum_{i=1}^{\ell} y_i\alpha_i = 0, & \\
 & 0 \le \alpha_i \le \frac{1}{\ell} & i = 1,\dots,\ell.
\end{array}
$$

Here, Q is the matrix defined by the relationship

$$Q = YKY \Longleftrightarrow Q_{ij} = y_i y_j K(\mathbf{x_i},\mathbf{x_j}),$$

where Y is a diagonal matrix whose satisfying $Y_{i,i} = y_i$. The SVM dual has only simple box constraints and a single equality constraint. For this reason, a large SVM problem can be *decomposed* and solved as a sequence of smaller problems. (Osuna et al., 1997, Osuna, 1998) If a data point never has a nonzero coefficient over the course of this procedure (the point is not a support vector and the algorithm never conjectures that it might be), then the associated row of K (equivalently Q) need never be computed at all. Very often, this condition holds for a large majority of the data points, and the time required to train an SVM is substantially less than the time required to compute all of K; this is what makes it possible to solve large SVM problems (relatively) quickly. It is also important to note that in state-of-the-art implementations of SVMs (Rifkin, 2002, Collobert and Bengio, 2001, Joachims, 1998), the idea of *caching* kernel products which were needed previously and will probably be needed again is crucial; if the data is high-dimensional, the time required to obtain a kernel product from a cache is much less than the time required to compute it anew.

Furthermore, the SVM will exhibit *sparsity*—generally only a small percentage of the c_i will be non-zero, making it much faster at test time as well. Therefore, for large problems with nonlinear kernels, the SVM is preferred to RLSC for computational reasons. For further discussion of this point, see Rifkin's PhD thesis (Rifkin, 2002).

In our paper, we use only the Gaussian kernel

$$K(\mathbf{x_1},\mathbf{x_2}) = \exp^{-\gamma||\mathbf{x_1}-\mathbf{x_2}||^2},$$

making the SVM the preferred algorithm. However, we also perform a large number of experiments with RLSC, both in order to support our claim that the accuracy of RLSC and SVM are essentially the same, and to motivate the theoretical results in Section 5, which only apply to the RLSC algorithm.

It is worth noting that in many "classic" derivations of SVMs, the primal problem is derived for the case of a linear hyperplane and separable data, using the idea of "maximizing margin". Non-separability is handled by introducing slack variables, the dual is taken, and only then is it observed that the $\mathbf{x_i}$ appear only as dot products $\mathbf{x_i}\cdot\mathbf{x_j}$, which can be replaced by kernel products $K(\mathbf{x_i},\mathbf{x_j})$ (the so-called "kernel trick"). Developing SVMs and RLSC in a unified framework from the perspective of Tikhonov regularization makes clear that we can use kernel products directly in the primal formulations, taking the dual only when it is useful for computational purposes. Many authors of the papers discussed in the next section instead take the more "classical" approach of deriving their algorithm for the linear case, taking the dual, and *then* nonlinearizing by means of kernel functions. This issue is discussed in more detail in Rifkin's PhD thesis (Rifkin, 2002).

3. Previous Work

The central thesis of this chapter is that one-vs-all classification using SVMs or RLSC is an excellent choice for multiclass classification. In the past few years, many papers have been presented that claim to represent an advance on this technique. We will review these papers in detail, directly considering the hypothesis that the new techniques outperform a simple OVA approach. These papers fall into two main categories. The first category attempts to solve a single optimization problem rather than combine the solutions to a collection of binary problems. The second category attempts to use the power of error-correcting codes to improve multiclass classification. We deal with these two approaches separately.

3.1 Single Machine Approaches

We now discuss the single-machine approaches that have been presented in the literature.

3.1.1 Vapnik and Blanz, Weston and Watkins

The single machine approach was introduced simultaneously by Vapnik (1998) and Weston and Watkins (1998). The formulations introduced in these two sources are essentially identical. The approach is a multiclass generalization of support vector machines. A standard SVM finds a function

$$f(x) = \sum_{j=1}^{\ell} c_j K(\mathbf{x}, \mathbf{x_j}) + b.$$

The multiclass SVM of Weston and Watkins finds N functions $f_1, \ldots, f_N$ simultaneously, where

$$f_i(x) = \sum_{j=1}^{\ell} c_{ij} K(\mathbf{x}, \mathbf{x_j}) + b_i.$$

The basic idea behind the multiclass SVM of Weston and Watkins (as well as all other single machine approaches, with slight modifications, as we shall see) is that instead of paying a penalty for each machine separately based on whether each machine satisfies its margin requirements for a given point, we pay a penalty based on the *relative* values output by the different machines. More concretely, given a single data point x belonging to class i, in the one-vs-all scheme we pay a penalty for machine i if $f_i(x) < 1$, and for all other classes j we pay a penalty if $f_j(x) > -1$. In the Weston and Watkins scheme, for each pair $i \neq j$, we pay a penalty if $f_i(x) < f_j(x) + 2$. If $f_i(x) < 1$, we may not pay a penalty, as long as $f_j(x)$ is sufficiently small for $i \neq j$; similarly, if $f_j(x) > 1$, we will not pay a penalty for x if $f_i(x)$ is sufficiently large. To facilitate this, we will use $\ell(N-1)$ slack variables ξ_{ij}, where $i \in \{1, \ldots, \ell\}$ and $j \in \{1, \ldots, N\} \backslash y_i$. Using these slack variables, the optimization problem being solved can be expressed (using our notation) as

$$\begin{array}{rl} \min\limits_{\mathbf{f_1}, \ldots, \mathbf{f_N} \in \mathcal{H}, \xi \in \mathbb{R}^{\ell(\mathbf{N-1})}} & \sum_{i=1}^{N} ||f_i||_K^2 + C \sum_{i=1}^{\ell} \sum_{j \neq y_i} \xi_{ij} \\ \text{subject to :} & f_{y_i}(\mathbf{x_i}) + b_{y_i} \geq f_j(\mathbf{x_i}) + b_j + 2 - \xi_{ij}, \\ & \xi_{ij} \geq 0. \end{array}$$

where the constraints all run over $i \in \{1, \ldots, \ell\}$ and $j \in \{1, \ldots, N\} \backslash y_i$. As in Section 2, we can write for each f_i

$$||f_i||_K^2 = \mathbf{c_{i \cdot}}^T K \mathbf{c_{i \cdot}},$$

where $\mathbf{c_{i \cdot}}$ is the vector whose jth entry is c_{ij}. Doing so leads to a single quadratic programming problem with $N\ell$ function defining variables c_{ij}, $(N-1)\ell$ slack variables ξ_{ij}, and N bias terms b_i. The dual of this problem can be taken using the standard Lagrangian approach. Weston and Watkins define α_{ij} to be the dual variables associated with the first set of constraints (including "dummy" variables α_{i,y_i}), and β_{ij} to be the dual variables associated with the second set of constraints. Introducing the notation

$$A_i = \sum_{j=1}^{N} \alpha_{ij},$$

and skipping intermediate algebra, the dual problem derived by Weston and Watkins is

$$\begin{array}{ll} \max\limits_{\alpha \in \mathbb{R}^{\ell N}} & 2 \sum_{ij} \alpha_{ij} + \sum_{i,j,k} \left[-\frac{1}{2} c_{j,y_i} A_i A_j + \alpha_{i,k} \alpha j, y_i - \frac{1}{2} \alpha_{i,k} \alpha_{j,k} \right] K(\mathbf{x_i}, \mathbf{x_j}) \\ \text{subject to :} & \sum_{i=1}^{\ell} \alpha_{ij} = \sum_{i=1}^{\ell} c_{ij} A_i, \\ & 0 \leq \alpha_{ij} \leq C, \\ & \alpha_{i,y_i} = 0. \end{array}$$

The first set of constraints holds for $j \in \{1, \ldots, N\}$, the second over $i \in \{1, \ldots, \ell\}$ and $j \in \{1, \ldots, N\}$, and the third over $i \in \{1, \ldots, \ell\}$.

It is not clear whether this is useful or not, as it is unknown whether the resulting dual problem can be decomposed in any useful way. Weston and Watkins mention in passing that "decomposition techniques can be used, as in the usual SV case," but provide no mathematical derivation or implementation. Unlike the SVM, which has box constraints and a single equality constraint over all the variables, this system has N equality constraints, where the equality constraint for class j involves $\ell + JN$ terms, and J is the number of points in class j. The relative complexity of the constraints makes it likely that the decomposition algorithm would have to be substantially more complicated to maintain feasibility of the generated solutions at each iteration. Also, unlike the SVM scenario, the "dual" problem does not succeed in fully eliminating the primal variables c_{ij}. Weston and Watkins consider nonlinear kernels only in the dual formulation.

Weston and Watkins perform two different sorts of experiments. In the first set of experiments, they work with toy examples where several classes in the plane are classified using their algorithm. They show examples which are both separable and nonseparable, but they do not compare their algorithm to any other method, so these experiments only serve as a proof of concept that the algorithm works in some reasonable way.

In the second set of experiments, they compare their algorithm to a one-vs-all scheme on five data sets from the UCI repository (Merz and Murphy, 1998); the data sets used were `iris, wine, glass, soy,`[5] and `vowel`. The authors conclude that their method seems to be approximately equivalent in accuracy to a one-vs-all or an all-pairs scheme, and suggest

[5] It is unclear to us what `soy` refers to. The UCI Repository contains a directory titled `soybean`, which contains two data sets, `soybean-large` and `soybean-small`. The `soy` data set considered by Weston

that the single-machine approach may have an advantage as regards the number of support vectors needed. However, the authors also state that "to enable comparison, for each algorithm $C = \infty$ was chosen (the training data must be classified without error)." Setting C to ∞ implies that the regularization is very weak; although there is some regularization, we are only able to select the smoothest function from among those functions having zero loss. This is rarely desirable in realistic applications, so it is hard to draw any conclusions about accuracy from these experiments. Furthermore, setting C to ∞ will tend to induce a much less smooth function and greatly increase the number of support vectors, making it difficult to use these experiments to draw conclusions about the number of support vectors required for different methods. There is an additional problem with the claim that the single-machine approach requires fewer support vectors, which is that in the OVA (or AVA) case, it is computationally easy to "reuse" support vectors that appear in multiple machines, leading to a large reduction in the total computational costs.

3.1.2 Lee, Lin and Wahba

Lee, Lin and Wahba present a substantially different single-machine approach to multiclass classification (Lee et al., 2001a,b). The work has its roots in an earlier paper by Lin (1999) on the asymptotic properties of support vector machine regularization for binary classification. If we define $p(\mathbf{x})$ to be the probability that a data point located at $\mathbf{x}$ is in class 1, Lin proved using elegant elementary arguments that the minimizer of $E[(1 - yf(\mathbf{x}))_+)]$ is $f(\mathbf{x}) = \text{sign}(p(\mathbf{x}) - \frac{1}{2})$. In other words, if we consider solving an SVM problem and let the number of data points tend to infinity, the minimizer of the loss functional (ignoring the regularization term $\lambda||f||_K^2$, and the fact that the functional f has to live in the RKHS $\mathcal{H}_K$) tends to $\text{sign}(p(\mathbf{x}) - \frac{1}{2})$. Lin refers to this function as the Bayes-optimal solution.

Considering this to be a useful property, Lee et al. design a multiclass classification technique with similar behavior. They begin by noting that a standard one-vs-all SVM approach does not have this property. In particular, defining $p_i(\mathbf{x})$ to be the probability that a point located at $\mathbf{x}$ belongs to class i, the Lin's results show that $f_i(\mathbf{x}) \rightarrow \text{sign}(p_i(\mathbf{x}) - \frac{1}{2})$ as $\ell \rightarrow \infty$. For all points for which $\arg\max_i p_i(\mathbf{x}) \geq \frac{1}{2}$, we will recover the correct result: asymptotically, $f_i(\mathbf{x}) = 1$, and $f_j(\mathbf{x}) = -1$ for $j \neq i$. However, if $\arg\max_i p_i(\mathbf{x}) < \frac{1}{2}$, then asymptotically, $f_i(\mathbf{x}) = -1 \ \forall i$, and we will be unable to recover the correct class. Lee et al. note that for other formulations such as the one of Weston and Watkins (1998), the asymptotic behavior is hard to analyze.

Lee, Lin and Wahba proceed to derive a multiclass formulation with the desired correct asymptotic behavior. For $1 \leq i \leq N$, they define v_i to be an N dimensional vector with a 1 in the ith coordinate and $\frac{1}{N-1}$ elsewhere.[6] The v_i vector plays the role of a "target" for points in class i—we try to get function outputs that are very close to the entries of v_i. However, for technical reasons, instead of worrying about all N functions, they only worry about $f_j(\mathbf{x_i})$ for $j \neq y_{\mathbf{x}_i}$, and ensure (approximate) correctness of $f_i(\mathbf{x_i})$ by requiring that

and Watkins does not match (in size or number of classes) either of these. There is also a note in this directory indicating the existence of other versions of the data set; from this note it seems that this may have been a version used separately by Mooney, Stepp and Reinke. Since this version does not seem to be publicly available, it is difficult to compare against it directly.

[6] We have taken some liberties with the notation in order to shorten the presentation and keep notation consistent with the rest of the paper.

for all $\mathbf{x}$, $\sum_{i=1}^{N} f_i(\mathbf{x}) = 0$. This leads to the following optimization problem:

$$\begin{array}{lcr} \min\limits_{f_1,\ldots,f_N \in \mathcal{H}_K} & \frac{1}{\ell}\sum_{i=1}^{\ell}\sum_{j=1,j\neq y_i}^{N}(f_j(\mathbf{x}_i)+\frac{1}{N-1})_+ + \lambda\sum_{j=1}^{C}||f_j||_K^2 & \\ \text{subject to :} & \sum_{j=1}^{C} f_j(\mathbf{x}) = 0, & \forall \mathbf{x}. \end{array}$$

Using arguments along the same lines of those in Lin (1999), it is shown that the asymptotic solution to this regularization problem (again ignoring the λ term and the fact that the functions must live in the RKHS) is $f_i(\mathbf{x}) = 1$ if $i = \arg\max_{j=1,\ldots,N} p_j(\mathbf{x})$ and $f_i(\mathbf{x}) = -\frac{1}{N-1}$ otherwise; $f_i(\mathbf{x})$ is one if and only if i is the most likely class for a point located at $\mathbf{x}$. They point out that this is a natural generalization of binary SVMs, if we view binary SVMs as producing two functions, one for each class, constrained so that $f_1(\mathbf{x}) = f_{-1}(\mathbf{x})$ for all $\mathbf{x}$. A Lagrangian dual is derived, and it is noted that the approach retains some of the sparsity properties of binary SVMs. The resulting optimization problem is approximately $N-1$ times as large as a single SVM problem, and no decomposition method is provided.

Although this approach is interesting, there are a number of problems with it. The primary difficulty is that the analysis is entirely asymptotic, holding only as the number of data points goes to infinity and the regularization term is ignored. In this framework, any method which asymptotically estimates densities accurately will also perform optimally. However, such density estimation methods have been shown to be grossly inferior to discriminative methods such as SVMs in real-world classification tasks using limited amounts of data. Therefore, it is difficult to argue the superiority of a method based only on its asymptotic behavior. In the Lee, Lin and Wahba analysis, no information is provided about the rate of convergence to the Bayes-optimal solution. In order to arrive at this Bayes-optimal solution, we must also let $\lambda \to 0$ and $\ell \to \infty$; although this is of course the right thing to do, the result is not at all surprising viewed in this context, and no information about rates is provided. Additionally, comparing this method to one-vs-all SVMs, the only points $\mathbf{x}$ for which this approach (asymptotically) makes a difference are points for which $\arg\max_i p_i(\mathbf{x}) < \frac{1}{2}$. In other words, if a single class is more than 50% likely at a given $\mathbf{x}$, this approach and the computationally much simpler one-vs-all approach will make the same prediction (asymptotically). We expect this to be the case for the vast majority of the probability mass of many real-world problems, although this is an intuition rather than a known fact.

If the class densities are highly overlapping in some region (one class is only slightly more likely than all the others), there are two additional problems. The first is that classification accuracy is inherently limited to the likelihood of the most likely class, indicating that our problem is too difficult to be usefully solved or that we have not represented our problem in a manner that allows good classification. There may be exceptions to this, such as problems involving financial data (which are notoriously hard to achieve good performance on), but in general, we are most interested in problems which can be solved fairly accurately. The second, more important difficulty is that in high dimensions, if two class densities are similar over a region, we expect that we will need a large number of points to capture this distinction.

Another intimately related problem with this approach is that it ignores the fundamental characteristics of the regularization approach, which is the attempt to find *smooth* functions that fit the data accurately. Although the optimization problem suggested does include a regularization term, the analysis of the technique is completely dependent on ignoring the regularization term. The Bayes-optimal asymptotic solution can be arbitrarily non-smooth, and convergence to it relies on the use of an RKHS that is dense in L_2 (such as the one obtained when the kernel K is Gaussian).

Two toy examples illustrating the method are presented. In one example, the method is illustrated graphically, and no comparisons to other methods are made. In the other example, a comparison to one-vs-all is made. The training data consists of 200 one-dimensional points (in the interval $[0,1]$) from three overlapping classes, and the test data consists of $10,000$ independent test points from the distribution. The distributions are chosen so that class 2 never has a conditional probability of more than 50%. In the example, the method of Lee, Lin and Wahba is able to predict class 2 over the region where it is more likely than any other class, and a one-vs-all system is not. On the test set, the one-vs-all system has an error rate of .4243 and the Lee, Lin and Wahba method has an error of .389. However, it is difficult to understand how the parameter settings were chosen, possibly indicating that different parameter settings would help the one-vs-all system. Additionally, the example is only one dimensional, and involved a relatively large number of points for one dimension. Furthermore, the difference in test error rates was not especially large. Nevertheless, this experiment is somewhat interesting, and it would be good to see a number of better-controlled experiments on more realistic data sets.

Some additional insights can be gained from taking another look at the original Lin paper. The Lee, Lin and Wahba paper was based on Lin's results for the SVM hinge loss function: $V(f(\mathbf{x}), y) = (1 - yf(\mathbf{x}))_+$. The Lin paper also includes easily proved results stating that for any $q > 1$, if the loss function is either $(1 - yf(\mathbf{x})_+)^q$ or $|y - f(\mathbf{x})|^q$, then the asymptotic minimizer is given by (recalling that $p(\mathbf{x})$ is the conditional probability of a point at $\mathbf{x}$ being in class 1):

$$f(\mathbf{x}) = \frac{(p(\mathbf{x}))^{\frac{1}{q-1}} - (1 - p(\mathbf{x}))^{\frac{1}{q-1}}}{(p(\mathbf{x}))^{\frac{1}{q-1}} + (1 - p(\mathbf{x}))^{\frac{1}{q-1}}}.$$

In the specific case of regularized least squares classification discussed in Section 2, $V(f(\mathbf{x}), y) = (y - f(\mathbf{x}))^2$, so the asymptotic discrimination function is

$$f(\mathbf{x}) = \frac{(p(\mathbf{x}))^{\frac{1}{2}} - (1 - p(\mathbf{x}))^{\frac{1}{2}}}{(p(\mathbf{x}))^{\frac{1}{2}} + (1 - p(\mathbf{x}))^{\frac{1}{2}}}.$$

Now, instead of SVM, let's consider the use of RLSC in a one-vs-all framework. We will (asymptotically) arrive at N functions, where

$$f_i(\mathbf{x}) = \frac{(p_i(\mathbf{x}))^{\frac{1}{2}} - (1 - p_i(\mathbf{x}))^{\frac{1}{2}}}{(p_i(\mathbf{x}))^{\frac{1}{2}} + (1 - p_i(\mathbf{x}))^{\frac{1}{2}}}.$$

Now assume $p_i(\mathbf{x}) > p_j(\mathbf{x})$. We will show that this implies that $f_i(\mathbf{x}) > f_j(\mathbf{x})$. Specifically, we consider the notationally simplified quantity

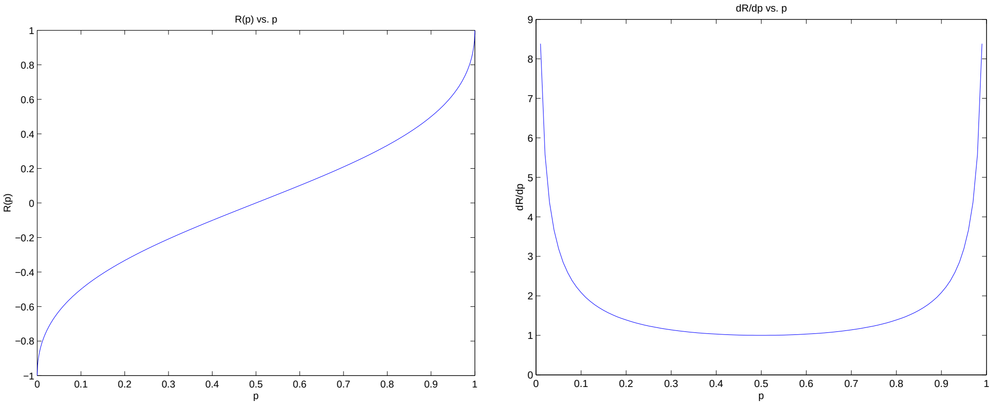
$$R(p) = \frac{p^{\frac{1}{2}} - (1 - p)^{\frac{1}{2}}}{p^{\frac{1}{2}} + (1 - p)^{\frac{1}{2}}},$$

and show that $R(p)$ is increasing as a function of p, for $p \in [0,1]$. We first note that $R(0) = -1$ and $R(1) = 1$. Next, for $p \in (0,1)$, we find that

$$\begin{aligned}
\frac{dR}{dp} &= \frac{\frac{d(p^{\frac{1}{2}}-(1-p)^{\frac{1}{2}})}{dp}(p^{\frac{1}{2}}+(1-p)^{\frac{1}{2}}) - (p^{\frac{1}{2}}-(1-p)^{\frac{1}{2}})\frac{d(p^{\frac{1}{2}}+(1-p)^{\frac{1}{2}})}{dp}}{(p^{\frac{1}{2}}+(1-p)^{\frac{1}{2}})^2} \\
&= \frac{\frac{1}{2}\left[(p^{-\frac{1}{2}}+(1-p)^{-\frac{1}{2}})(p^{\frac{1}{2}}+(1-p)^{\frac{1}{2}})\right]}{(p^{\frac{1}{2}}+(1-p)^{\frac{1}{2}})^2} \\
&+ \frac{\frac{1}{2}\left[(p^{\frac{1}{2}}-(1-p)^{\frac{1}{2}})(p^{-\frac{1}{2}}-(1-p)^{-\frac{1}{2}})\right]}{(p^{\frac{1}{2}}+(1-p)^{\frac{1}{2}})^2} \\
&= \frac{\frac{1}{2}\left[1+(\frac{1-p}{p})^{\frac{1}{2}}+(\frac{p}{1-p})^{\frac{1}{2}}+1-1+(\frac{1-p}{p})^{\frac{1}{2}}+(\frac{p}{1-p})^{\frac{1}{2}}-1\right]}{(p^{\frac{1}{2}}+(1-p)^{\frac{1}{2}})^2} \\
&= \frac{(\frac{1-p}{p})^{\frac{1}{2}}+(\frac{p}{1-p})^{\frac{1}{2}}}{(p^{\frac{1}{2}}+(1-p)^{\frac{1}{2}})^2} \\
&> 0.
\end{aligned}$$

In other words, $R(p)$ is a strictly increasing function of p (Figure 1 shows both $R(p)$ and $\frac{dR}{dP}$ as a function of p), which implies that if class i is more likely than class j at point $\mathbf{x}$, $f_i(\mathbf{x}) > f_j(\mathbf{x})$. This in turn implies that if we use a one-vs-all RLSC scheme, and classify test points using the function with the largest output value (which is of course the common procedure in one-vs-all classification), the error of our scheme will asymptotically converge to the Bayes error, just as the multiclass SVM of Lee, Lin and Wahba does. Put differently, *the need for a single-machine approach with a sum-to-zero constraint on the functions in order to asymptotically converge to the Bayes function was a specific technical requirement associated with the use of the SVM hinge loss.* When we change the loss function to the square loss, another commonly used loss function that gives equivalent accuracy, the one-vs-all approach has precisely the same asymptotic convergence properties.

We are not claiming that this analysis is a strong argument in favor of the one-vs-all RLSC scheme as opposed to the one-vs-all SVM. The argument is an asymptotic one, applying only in the limit of infinitely many data points. There are a large number of schemes that will work equivalently with infinite amounts of data, and it is something of a technical oddity that the one-vs-all SVM appears not to be one of them. However, we do not believe that this asymptotic analysis tells us anything especially useful about the performance of a multiclass scheme on finite, limited amounts of high-dimensional data. In this regime, both a one-vs-all SVM scheme and a one-vs-all RLSC scheme have been demonstrated to behave quite well empirically. However, the fact that the one-vs-all RLSC scheme has equivalent asymptotic behavior to the Lee, Lin and Wahba scheme casts further doubt on the idea that their scheme will prove superior to one-vs-all on real applications.



(a) (b)

Figure 1: An analysis of the quantity $R(p)$. (a): $R(p)$ vs. p. (b): $\frac{dR}{dp}$ vs. p. We see that $R(p)$ is a strictly increasing function of p, implying that if class i is more likely than class j at point $\mathbf{x}$, then, asymptotically, $f_i(\mathbf{x}) > f_j(\mathbf{x})$.

3.1.3 Bredensteiner and Bennett

Bredensteiner and Bennett (1999) also suggest a single-machine approach to multiclass classification.[7] Like Weston and Watkins, they begin by stating the invariant that they want the functions generated by their multiclass system to satisfy:

$$\mathbf{w_{y_i}}^T \cdot \mathbf{x_i} + b_i \geq \mathbf{w_j}^T \cdot \mathbf{x_i} + b_j + 1 - \xi_{ij},$$

where $\mathbf{x_i}$ is a member of class y_i and $j \neq y_i$. They rewrite this equation as

$$(\mathbf{w_{y_i}} - \mathbf{w_j})^T \cdot \mathbf{x_i} \geq (b_j - b_i) + 1 - \xi_{ij}.$$

They then argue that a good measure of the separation between class i and j is $\frac{2}{||w_i - w_j||}$, and suggest maximizing this quantity by minimizing $||w_i - w_j||$ over all pairs i and j. They also add the regularization term $\frac{1}{2}\sum_{i=1}^{N} ||w_i||^2$ to the objective function. The resulting optimization problem (where we have adjusted the notation substantially to fit with our development) is

$$\begin{array}{cc} \min\limits_{(\mathbf{w_i}, b_i \in \mathbb{R}^{d+1})} & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} ||w_i - w_j||^2 + \frac{1}{2}\sum_{i=1}^{N} ||w_i||^2 + C\sum_{i=1}^{\ell}\sum_{j \neq y_i} \xi_{ij} \\ \text{subject to :} & \mathbf{w_{y_i}} - \mathbf{w_j}^T \cdot \mathbf{x_i} \geq (b_j - b_i) + 1 - \xi_{ij}, \\ & \xi_{ij} \geq 0. \end{array}$$

Using standard but rather involved techniques, they derive the Lagrangian dual problem, and observe that the dot products can be replaced with kernel products.

[7] The Bredensteiner and Bennett formulation has been shown to be equivalent to the Weston and Watkins formulation (Guermeur, 2002, Hsu and Lin, 2002).

Two sets of experiments were performed. The first set involved two data sets from the UCI repository (Merz and Murphy, 1998), (wine and glass). Ten-fold cross validation was performed on each data set, and polynomials of degree one through five are used as models. On both data sets, the highest performance reported is for a one-vs-all SVM system rather than the multiclass system they derived (their multiclass SVM does perform better than an unregularized system which merely finds an arbitrary separating hyperplane).

In the second set of experiments, two separate subsets of the USPS data were constructed. Subsets of the training data were used, because training their multiclass method on the full data set was not computationally feasible.[8] On both data sets, a one-vs-all SVM system performs (slightly) better than their single-machine system.

3.1.4 Crammer and Singer

Crammer and Singer consider a similar but not identical single-machine approach to multiclass classification (Crammer and Singer, 2001). This work is a specific case of a general method for solving multiclass problems, presented in several papers (Crammer and Singer, 2000b,a, 2002) and discussed in Section 3.2 below. The method can be viewed as a simple modification of the approach of Weston and Watkins (1998). Weston and Watkins start from the idea that if a point $\mathbf{x}$ is in class i, we should try to make $f_i(\mathbf{x}) \geq f_j(\mathbf{x}) + 2$ for $i \neq j$, and arrive at the following formulation:

$$\begin{array}{rl} \min\limits_{\mathbf{f_1},\ldots,\mathbf{f_N}\in\mathcal{H},\xi\in\mathbb{R}^{\ell(\mathbf{N-1})}} & \sum_{i=1}^{N} \|f_i\|_K^2 + C\sum_{i=1}^{\ell}\sum_{j\neq y_i}\xi_{ij} \\ \text{subject to :} & f_{y_i}(\mathbf{x_i}) + b_{y_i} \geq f_j(\mathbf{x_i}) + b_j + 2 - \xi_{ij}, \\ & \xi_{ij} \geq 0. \end{array}$$

Crammer and Singer begin with the same condition, but instead of paying for *each* class $j \neq i$ for which $f_i(\mathbf{x}) < f_j(\mathbf{x}) + 1$,[9] they pay only for the *largest* $f_j(\mathbf{x})$. This results in a *single* slack variable for each data point, rather than the $N-1$ slack variables per point in the Weston and Watkins formulation. The resulting mathematical programming problem is (as usual, placing the formulation into our own notations for consistency):

$$\begin{array}{rl} \min\limits_{\mathbf{f_1},\ldots,\mathbf{f_N}\in\mathcal{H},\xi\in\mathbb{R}^{\ell}} & \sum_{i=1}^{N} \|f_i\|_K^2 + C\sum_{i=1}^{\ell}\xi_i \\ \text{subject to :} & f_{y_i}(\mathbf{x_i}) \geq f_j(\mathbf{x_i}) + 1 - \xi_i, \\ & \xi_i \geq 0. \end{array}$$

The majority of the paper is devoted to the development of an efficient algorithm for solving the above formulation. The Lagrangian dual is taken, and the standard observations that

[8] For example, their method takes over 10,000 seconds to train on a data set of 1,756 examples in three classes. Modern, freely available SVM solvers such as SVMTorch (Collobert and Bengio, 2001) routinely solve problems on 5,000 or more points in 30 seconds or less.

[9] The choice of 1 rather than 2 as a "required difference" is arbitrary. Crammer and Singer quite reasonably choose 1 for simplicity. The choice of 2 in Weston and Watkins seems to be motivated from a desire to make the system as similar to standard binary SVMs as possible, where we require a margin of 1 for points in the positive class and -1 for points in the negative class. This choice is arbitrary: if we required 1 for points in the positive class and 0 for points in the negative class, the details of the algorithm would change, but the function found by the algorithm would not.

the dot products can be replaced with kernels are made. An elegant dual decomposition algorithm is developed, in which a single data point is chosen at each step and an iterative algorithm is used to solve the reduced N-variable quadratic programming problem associated with the chosen data point. A number of additional implementation tricks are used, including using the KKT conditions for example selection, caching kernel values, maintaining an active set from which the example to be optimized in a given iteration is chosen, and cooling of the accuracy parameter.

In the experimental section of the paper, Crammer and Singer considered a number of data sets from the UCI repository. They produce a chart showing the difference in error rate between their one-machine system and an OVA system, but not the actual error rates. There are two data sets for which the difference between their system and OVA seems to be large: `satimage` with a difference of approximately 6.5% in performance, and `shuttle` with a difference of approximately 3% in performance. In personal communication, Crammer indicated that the actual error rates for his system on these two data sets were 8.1% and 0.1%, respectively. In our own one-vs-all experiments on the `satimage` data (see Section 4, we observed an error rate of 8.2%. Although we did not do experiments on the `shuttle` data set for this paper, we note that Fürnkranz (2002) achieved an error of 0.3% on this data set using a simple OVA system with Ripper as the binary learner. These numbers for an OVA system are in sharp contrast to the results implied by the paper; we have no explanation for the differences, and Crammer and Singer do not provide enough information to precisely reproduce their experiments.

3.1.5 Summary

When we apply the one-vs-all strategy, we solve a separate optimization problems for each of the N classes. The single machine approaches solve a single optimization problem to find N functions simultaneously. Of the papers considered here, only Crammer and Singer claimed that their single-machine approach outperformed OVA across realistic (non-toy) data sets, and as we show below in Section 4, performance equivalent to the best results they achieved can also be achieved by an OVA scheme when the underlying binary classifiers are properly tuned. Therefore, although these approaches may have theoretical interest, it does not appear that they offer any advantages over a simple OVA (or AVA) scheme in the solution of multiclass classification problems. Additionally, the methods are generally complicated to implement and slow to train, indicating that they would have to have some other compelling advantage, such as higher accuracy or a much sparser representation, to make them worth using in applications.

3.2 Error-Correcting Coding Approaches

We now turn to error-correcting code approaches, a second major approach to combining binary classifiers into a multiclass classification system.

3.2.1 Dietterich and Bakiri

Dietterich and Bakiri (1995) first popularized the idea of using error-correcting codes for multiclass classification. We will describe the method using notation introduced later by Allwein et al. (2000).

Dietterich and Bakiri suggested the use of a $\{-1, 1\}$-valued matrix M of size N by F, that is $M \in \{-1, 1\}^{N \times F}$, where N is the number of classes and F is the number of binary classifiers to be trained. We let M_{ij} refer to the entry in the ith row and the jth column of M. The ith column of the matrix induces a partition of the classes into two "metaclasses", where a point $\mathbf{x_i}$ is placed in the positive metaclass for the jth classifier if and only if $M_{y_i j} = 1$. In our framework, in which the binary classifiers implement Tikhonov regularization, the jth machine solves the following problem:

$$\min \sum_{i=1}^{\ell} V(f_j(\mathbf{x_i}), M_{y_i j}) + \lambda ||f_j||_K^2.$$

When faced with a new test point $\mathbf{x}$, we compute $f_1(\mathbf{x}), \ldots, f_F(\mathbf{x})$, take the signs of these values, and then compare the Hamming distance between the resulting vector and each row of the matrix, choosing the minimizer

$$f(\mathbf{x}) = \arg \min_{r \in 1, \ldots, N} \sum_{i=1}^{F} \left(\frac{1 - \text{sign}(M_{ri} f_i(\mathbf{x}))}{2} \right).$$

This representation had been previously used by Sejnowski and Rosenberg (1987), but in their case, the matrix M was chosen so that a column of M corresponded to the presence or absence of some specific feature across the given classes. For example (taken from Dietterich and Bakiri, 1995), in a digit recognizer, one might build a binary classifier that learned whether or not the digit contained a vertical line segment, placing the examples in classes 1, 4, and 5 in the positive metaclass for this classifier, and the remaining classes in the negative metaclass.

Dietterich and Bakiri take their cue from the theory of error-correcting codes (Bose and Ray-Chaudhuri, 1960), and suggest that the M matrix be constructed to have good error-correcting properties. The basic observation is that if the minimum Hamming distance between rows of M is d, then the resulting multiclass classification will be able to correct any $\lfloor \frac{d-1}{2} \rfloor$ errors. They also note that good *column* separation is important when using error-correcting codes for multiclass classification; if two columns of the matrix are identical (or are opposites of each other, assuming an algorithm that treats positive and negative examples equivalently), they will make identical errors.

After these initial observations, the majority of the paper is devoted to experimental results. A number of data sets from various sources are used, including several data sets from the UCI Machine Learning Repository (Merz and Murphy, 1998), and a subset of the NETtalk data set used by Sejnowski and Rosenberg (1987). Two learning algorithms were tested: decision trees using a modified version of the C4.5 algorithm (Quinlan, 1993), and feed-forward neural networks. The parameters were often tuned extensively to improve performance. In some cases, the algorithms were modified for individual data sets. They considered four different methods for constructing good error-correcting codes: an exhaustive method, a method that selects a subset of the columns generated by the exhaustive method, a method based on randomized hill climbing, and a method based on using BCH codes or a subset of BCH codes (sometimes selected using manual intervention). In summary, although the description of the experimental work is quite lengthy, it would be essentially impossible to replicate the work exactly due to its complexity and the level of detail at which it was reported.

A large variety of experimental results are reported. It appears that in general, with the data sets and algorithms tried, the error-correcting code approach performs better than a one-vs-all approach. However, the difference is often small, and it is difficult to know how good the underlying binary classifiers are. In many instances, only relative performance results are given—the difference between the error-correcting and a one-vs-all method is given, but the actual performance numbers are not given, making comparison to alternate approaches (such as a one-vs-all SVM scheme) impossible.

3.2.2 Allwein, Schapire and Singer

In 2000, Allwein, Schapire, and Singer (2000) extended the earlier work of Dietterich and Bakiri in several directions. They were specifically interested in *margin-based* classifiers, where the underlying classifier is attempting to minimize an expression of the form

$$\frac{1}{\ell}\sum_{i=1}^{\ell} L(y_i f(\mathbf{x_i})),$$

where L is an arbitrary (chosen) function (while also possibly trying to minimize a regularization term). The quantity $y_i f(\mathbf{x_i})$ is referred to as the *margin*. In the case of the SVM, $L(yf(\mathbf{x})) = (1-yf(\mathbf{x_i}))_+$, and in the case of RLSC, $L(yf(\mathbf{x})) = (1-yf(\mathbf{x}))^2 = (y-f(\mathbf{x}))^2$; we see that both SVM and RLSC are margin-based classifiers, and that we can easily relate the margin loss function $L(yf(\mathbf{x}))$ to the loss function $V(f(\mathbf{x}), y)$ we considered in Section 2. Allwein et al. are also very interested in the AdaBoost algorithm (Freund and Schapire, 1997, Schapire and Singer, 1999), which builds a function $f(\mathbf{x})$ that is a weighted linear combination of base hypotheses h_t:

$$f(\mathbf{x}) = \sum_t \alpha_t h_t(\mathbf{x}),$$

where the h_t are selected by a (weak) base learning algorithm, and reference numerous papers indicating that AdaBoost is approximately greedily minimizing

$$\sum_{i=1}^{\ell} e^{-y_i f(\mathbf{x_i})},$$

demonstrating that AdaBoost is a margin-based classifier with $L(yf\mathbf{x}) = e^{-y_i f(\mathbf{x_i})}$.

Allwein et al. chose the matrix $M \in \{-1,0,1\}^{N\times F}$, rather than only allowing 1 and -1 as entries in the matrix as Dietterich and Bakiri did. If $M_{y_i j} = 0$, then example i is simply not used when the jth classifier is trained. With this extension, they were able to place one-vs-all classification, error-correcting code classification schemes, and all-pairs classification schemes (Hastie and Tibshirani, 1998) in a single theoretical framework.

If the classifiers are combined using Hamming decoding (taking the signs of the real values output by the classifiers, then finding the closest match among the rows of M), we again have

$$f(\mathbf{x}) = \arg\min_{r\in 1,\ldots,N} \sum_{i=1}^{F}\left(\frac{1-\operatorname{sign}(M_{ri}f_i(\mathbf{x}))}{2}\right),$$

where it is now understood that if $M_{ri} = 0$ (class r was not used in the ith classifier), class r will contribute $\frac{1}{2}$ to the sum. Allwein et al. note that the major disadvantage of Hamming decoding is that it completely ignores the magnitude of the predictions, which can often be interpreted as a measure of "confidence" of a prediction. If the underlying classifiers are margin-based classifiers, they suggest using the loss function L instead of the Hamming distance. More specifically, they suggested that the prediction for a point $\mathbf{x}$ should be the class r that minimizes the total loss of the binary predictions under the assumptions that the label for point $\mathbf{x}$ for the ith machine is M_{ri}:

$$f(\mathbf{x}) = \arg \min_{r \in 1,\ldots,N} \sum_{i=1}^{F} L(M_{ri} f_i(\mathbf{x})).$$

This procedure is known as *loss-based decoding.* If the matrix M represents a one-vs-all coding scheme ($M_{ri} = 1$ if $r = i$, $M_{ri} = -1$ otherwise), the above equation simplifies to

$$\begin{aligned} f(\mathbf{x}) &= \arg \min_{r \in 1,\ldots,N} \sum_{i=1}^{F} L(M_{ri} f_i(\mathbf{x})) \\ &= \arg \min_{r \in 1,\ldots,N} L(f_r(\mathbf{x})) - \sum_{i \neq r}^{F} L(-f_i(\mathbf{x})). \end{aligned}$$

It is easy to check that for both SVM and RLSC, the prediction in the one-vs-all scheme will be chosen so that

$$f(\mathbf{x}) = \arg \max_{r} f_r(\mathbf{x_i}).$$

Allwein et al. provide an elegant analysis of the training error of multiclass error-correcting code based systems using both Hamming decoding and loss-based decoding. They also provide an analysis of the generalization performance of multiclass loss-based schemes in the particular case when the underlying binary classifier is AdaBoost. The arguments are extensions of those given by Schapire et al. (1998), and are beyond the scope of this paper.

The remainder of the paper is devoted to experiments on both toy and UCI Repository data sets, using AdaBoost and SVMs as the base learners. The two stated primary goals of the experiments are to compare Hamming and loss-based decoding and to compare the performance of different output codes.

The toy experiment considers $100k$ one-dimensional points generated from a single normal distribution, selecting the class boundaries so that each class contains 100 training points. AdaBoost is used as the weak learner, and comparisons are made between Hamming and loss-based decoding, and between a one-vs-all code and a complete code. The authors find that the loss-based decoding substantially outperforms the Hamming decoding, and that the one-vs-all and complete codes perform essentially identically.

Allwein et al. next consider experiments on a number of data sets from the machine learning repository. For SVMs, eight data sets are used: `dermatology, satimage, glass, ecoli, pendigits, yeast, vowel,` and `soybean`. Five different codes are considered: the one-vs-all code (OVA), the all-pairs code (omitted when there were too many classes) (AVA), the complete code (omitted when there were too many classes) (COM), and two types of random codes. The first type had $\lceil 10log_2(N) \rceil$ columns, and each entry was chosen to be 1 or -1 with equal probabilities. The codes were picked by considering $10,000$ random matrices, and picking the one with the highest value of ρ which did not have any identical columns. These codes were refereed to as *dense* (DEN) codes. They also considered *sparse* (SPA) codes, which had $\lceil 15log_2(N) \rceil$ columns, and each entry was 0 with probability $\frac{1}{2}$, and 1 or -1 with probability $\frac{1}{4}$ each. Again, 10,000 random matrices were considered, and the one with the best ρ with no identical columns and no columns or rows containing only zeros was chosen.

Direct numerical results of the experiments are presented, as well as bar graphs showing the relative performance of the various codes. The authors conclude that "For SVM, it is clear that the widely used one-against-all code is inferior to all the other codes we tested." However, this conclusion is somewhat premature. All the SVM experiments were performed using a polynomial kernel of degree 4, and no justification for this choice of kernel was given. Additionally, the regularization parameter used (λ) was not specified by the authors. Looking at the bar graphs comparing relative performance, we see that there are two data sets on which the one-vs-all SVMs seem to be doing particularly badly compared to the other codes: `satimage` and `yeast`. We performed our own SVM experiments on this data, using a Gaussian kernel with γ and C tuned separately for each scheme (for details and actual parameter values, see Section 4).[10]

The results are summarized in Table 1 and Table 2. We find that while other codes do sometimes perform (slightly) better than one-vs-all, that none of the differences are large. This is in stark contrast to the gross differences reported by Allwein et al. Although we did not test the other data sets from the UCI repository (on which Allwein and Schapire found that all the schemes performed very similarly), this experiment strongly supported the hypothesis that the differences observed by Allwein et al. result from a poor choice of kernel parameters, which makes the SVM a much weaker classifier than it would be with a good choice of kernel. In this regime, it is plausible that the errors from the different classifiers will be somewhat decorrelated, and that a scheme with better error-correcting properties than the one-vs-all scheme will be superior. However, given that our goal is to solve the problem as accurately as possible, it appears that choosing the kernel parameters to maximize the strength of the individual binary classifiers, and then using a one-vs-all multiclass scheme, performs as well as the other coding schemes, and, in the case of the

[10] We replicated the dense and sparse random codes as accurately as possible, but the information in Allwein et al. is incomplete. For both codes, we added the additional constraint that each column had to contain at least one $+1$ and at least one -1; one assumes that Allwein et al. had this constraint but did not report it, as without it, the individual binary classifiers could not be trained. For the sparse random code, the probability that a random column of length six (the number of classes in the `satimage` data set) generated according to the probabilities given fails to contain both a $+1$ and a -1 is more than 35%, and the procedure as defined in the paper fails to generate a single usable matrix. Personal communication with Allwein et al. indicate that it is likely that columns not satisfying this constraint were thrown out immediately upon generation. Additionally, there are only 601 possible length six columns containing at least one $+1$ and one -1 entry, and if these columns were chosen at random, only 28% of the matrices generated (the matrices have $\lceil 15log_2(6) \rceil = 39$ columns) would not contain duplicate columns. Because there was no mention in either case of avoiding columns which were opposites of each other (which is equivalent to duplication if the learning algorithms are symmetric), we elected to allow duplicate columns in our sparse codes, in the belief that this would have little effect on the quality of the outcome.

	OVA	AVA	COM	DEN	SPA
Allwein et al.	40.9	27.8	13.9	14.3	13.3
Rifkin & Klautau	8.2	7.8	7.8	7.7	8.9

Table 1: Multiclass classification error rates for the `satimage` data set. Allwein et al. used a polynomial kernel of degree four and an unknown value of C. Rifkin and Klautau used a Gaussian kernel with σ and C tuned separately for each scheme; see Section 4 for details. We see that with the Gaussian kernel, overall performance is much stronger, and the differences between coding schemes disappear.

	OVA	AVA	COM	DEN	SPA
Allwein et al.	72.9	40.9	40.4	39.7	47.2
Rifkin & Klautau	40.3	41.0	40.3	40.1	38.6

Table 2: Multiclass classification error rates for the `yeast` data set. Allwein et al. used a polynomial kernel of degree four, an unknown value of C, and ten-fold cross-validation. Rifkin and Klautau used a Gaussian kernel with σ and C tuned separately for each scheme, and ten-fold cross-validation; see Section 4 for details. We see that with the Gaussian kernel, the performance of the one-vs-all scheme jumps substantially, and the differences between coding schemes disappear.

`satimage` data, noticeably better than *any* of the coding schemes when the underlying classifiers are weak.[11]

3.2.3 Crammer and Singer

Crammer and Singer develop a formalism for multiclass classification using *continuous* output coding (Crammer and Singer, 2000a,b, 2002). This formalism includes the single-machine approach discussed in Section 3.1.4 as a special case.

The Crammer and Singer framework begins by assuming that a collection of binary classifiers $f_1, \ldots, f_F$ is provided. The goal is then to *learn* the N-by-F error-correcting code matrix M. Crammer and Singer show (under some mild assumptions) that finding an optimal discrete code matrix is an NP-complete problem, so they relax the problem and allow the matrix M to contain real-valued entries. Borrowing ideas from regularization, they argue that we would like to find a matrix M that has good performance on the training set but also has a small norm. To simplify the presentation, we introduce the following notation. We let $\bar{f}(\mathbf{x})$ denote the vector $f_1(\mathbf{x}), \ldots, f_F(\mathbf{x})$, and we let M_i denote the ith row of the matrix M. Given a matrix M, we let $K(\bar{f}(\mathbf{x}), M_i)$ denote our confidence that point x is in class i; here K is an arbitrary positive definite kernel function satisfying Mercer's

[11] In this context, weak is not used in a formal sense, but merely as a stand-in for poorly tuned.

Theorem. Then, the Crammer and Singer approach is:

$$\min_{M \in \mathbb{R}^{N \times F}} \quad \lambda ||M||_p + \sum_{i=1}^{\ell} \xi_i$$
$$K(\bar{f}(\mathbf{x_i}), M_{y_i}) \geq K(\bar{f}(\mathbf{x_i}), M_r) + 1 - \xi_i.$$

In the above formulation, the constraints range over all points $\mathbf{x_i}$, and all classes $r \neq y_i$. Simply put, we try to find a matrix with small norm with the property that the confidence for the correct class is greater by at least one than the confidence for any other class. Note that as in the approach discussed in Section 3.1.4, there is only a single slack variable ξ_i for each data point, rather than $N-1$ slack variables per data point as in many of the other formulations we discuss.

In their general formulation, the norm in which the matrix is measured ($||M||_p$) is left unspecified. Crammer and Singer briefly show that if $p = 1$ or $p = \infty$ and $K(\mathbf{x_i}, \mathbf{x_j}) = \mathbf{x_i} \cdot \mathbf{x_j}$, the resulting formulation is a linear program. They spend the majority of the paper considering $p = 2$ (technically, they penalize $||M||_2^2$, not $||M||_2$), and showing that this choice results in a quadratic programming problem. They take the dual (in order to introduce kernels; again, they start with the linear formulation in the primal), and indicate some algorithmic approaches to solving the dual problem.

In an interesting twist, Crammer and Singer also show that we can derive the one-machine multiclass SVM formulation used in a different paper of theirs (Crammer and Singer, 2001, discussed in Section 3.1.4) by taking $\bar{f}(\mathbf{x}) = \mathbf{x}$. In this case, the implicit assumption is that our "given" binary classifiers are d (the dimensionality of the input space) machines, where $f_i(\mathbf{x})$ is equal to the value of the ith dimension at point x. In the linear case ($K(\mathbf{x_i}, \mathbf{x_j}) = \mathbf{x_i} \cdot \mathbf{x_j}$), the code matrix M becomes the N separating hyperplane functions $w_1, w_2, \ldots, w_N$. This formulation is discussed in greater detail in the previous section.

Experiments are performed on seven different data sets from the UCI repository, as well as a subset of the MNIST data set. The experiments compare the performance of the continuous output codes to discrete output codes, including the one-vs-all code, BCH codes, and random codes. Personal communication with Crammer indicates that the "base learners" for the continuous codes are linear SVMs. Seven different kernels are tested, although their identities are not disclosed (personal communication indicates that they were homogeneous and nonhomogeneous polynomials of degree one through three, and a Gaussian kernel with a γ that was not recorded). No performance results for individual experiments are given. Instead, for each data set, we find the improvement in performance for the "best kernel" (presumably the kernel with the largest difference in performance), and the average improvement in performance across the seven kernels. It is important to note that his comparison was not against an OVA system, but against using the error-correcting coding approach directly with *linear SVMs* as the underlying binary classifiers. Therefore, he was comparing the classification ability of nonlinear and linear systems on data sets for which it is well known that Gaussian classifiers perform very strongly. In this context, his results are unsurprising.

Crammer communicated to us personally the actual performance numbers, which allows us to compare their continuous codes to an OVA approach. For the `satimage` data, the *best* error rate they achieved (over all seven kernels and three coding schemes) was 9.8%, compared to 8.2% for OVA in the current experiments (see Section 4). For the `shuttle` data, their best error rate was 0.5%; while we did not do experiments on this data set because of its unwieldy size, we note that Fürnrkanz (see below) achieved an error rate of 0.3% on this data set using an OVA system with Ripper as the underlying binary learner. We see that although the nonlinear system developed here greatly outperformed a linear system, it did not allow us to actually achieve better multiclass classification error rates than a simple well-tuned OVA system.

3.2.4 Fürnkranz

Relatively recently, Fürnkranz published a paper on round robin classification (Fürnkranz, 2002), which is another name for all-vs-all or all-pairs classification. He used Ripper (Cohen, 1995), a rule-based learner, as his underlying binary learner. He experimentally found that across a number of data sets from the UCI repository, an all-vs-all system had improved performance compared to a one-vs-all scheme. Fürnkranz used McNemar's test (McNemar, 1947) to decide when two classifiers were different. We hypothesize that Ripper is not as effective a binary learner as SVMs, and is therefore able to benefit from a scheme such as all-pairs; however, the scheme does not enable Fürnkranz to obtain better results than an OVA SVM scheme. In Section 4, we compare a variety of error-correcting schemes on the *specific data sets on which Fürnkranz found that AVA performed much better than OVA*; when the underlying classifiers are well-tuned SVMs, we find that the improved performance of AVA over OVA, observed by Fürnkranz, disappears.

3.2.5 Platt, Cristianini and Shawe-Taylor

The DAG (Directed Acyclic Graph) method of Platt, Cristianini, and Shawe-Taylor (2000) does not fit easily into the error-correcting code framework, but has much more in common with these methods than with the single-machine approaches, so is presented here. The DAG method is identical to AVA at training time—one SVM is trained for each pair of classes. At test time, an acyclic graph is used to determine which classifiers to test on a given point. First, classes i and j are compared, and whichever class achieves a lower score is removed from further consideration. By repeating this process $N-1$ times, $N-1$ classes are removed from consideration, and the final remaining class is predicted. Although the order in which classes are compared can affect the results, the authors observe that empirically, the ordering does not seem to affect the accuracy. The authors conduct well-controlled experiments on two data sets (`USPS` and `letter`), and observe that the OVA, AVA and DAG approaches have essentially identical accuracy, but that the DAG approach is substantially faster than OVA at testing time.

3.2.6 Hsu and Lin

Hsu and Lin (2002) present an empirical study comparing various methods of multiclass classification using SVMs as the binary learners. They conclude that "one-against-one and DAG methods are more suitable for practical use than the other methods." However, although they run a substantial number of experiments and their experimental protocol is sound, their data do not seem to support their conclusions. In particular, the table of

	Best Score	Worst Score	Difference	Size	Size * Diff
iris	97.333	96.667	.666	150	1.000
wine	99.438	98.876	.562	178	1.000
glass	73.832	71.028	2.804	214	6.000
vowel	99.053	98.485	.568	528	3.000
vehicle	87.470	86.052	1.418	746	10.58
segment	97.576	97.316	.260	2310	6.006
dna	95.869	95.447	.422	1186	5.005
satimage	92.35	91.3	1.050	2000	20.1
letter	97.98	97.68	.300	5000	15.0
shuttle	99.938	99.910	.028	14500	4.06

Table 3: A view of the multiclass results of Hsu and Lin (2002) for RBF kernels. The first three columns show the performance of the best and worst performing classifier for each data set, the third column shows the difference in performance between the best and worst, the fourth column the size of the data set, and the fifth column the difference expressed as a number of data points. Note that for the first six data sets, there is no training set and CV was used, so the fourth column reports the entire size of the data set. Columns 1-3 are percentages, columns four and five are numbers of points.

results for tuned RBF classifiers[12] shows That among the five methods they tried (OVA, AVA, DAG, the method of Crammer and Singer (2000b), and the method proposed by Vapnik (1998) and Weston and Watkins (1998)) are essentially identical. Table 3 presents one view of this data—we show, for each data set, the performance of the best and worst performing systems, and the difference between the best and worst both as a percentage and as a number of data points.

We see that the vast majority of these differences are quite small. Furthermore, visual inspection of Hsu and Lin's results show no clear pattern of which system is actually better across different data sets. Therefore, we must conclude that the Hsu and Lin results support the notion that at least as far as accuracy is concerned, when well-tuned binary classifiers (in this case SVMs with RBF kernels) are used as the underlying classifiers, a wide variety of multiclass classification schemes are essentially indistinguishable.

Hsu and Lin also examine the training and testing times of the various systems. Here again they find that the AVA and DAG systems have an advantage; although they are training $O(N^2)$ classifiers rather than $O(N)$ for an OVA system, the individual classifiers are much smaller, and given that the time required to train on ℓ points is generally superlinear in ℓ, we expect that AVA and DAG systems will train faster; this point is also explored in detail by Fürnkranz (2002). Their implementation is not heavily optimized, so it is difficult

[12] Hsu and Lin also ran experiments where the underlying binary classifiers were linear, but this again corresponds to a situation where the underlying binary classifiers are poorly tuned, and the overall accuracy of all methods in this regime is lower than with RBF kernels on several data sets, and better on none.

to draw conclusions from this; in particular, their implementation does *not* share kernel products between different classifiers.[13] At testing time, the systems are relatively close together in speed (within a factor of 2), indicating that this argument is mostly about the time required for training. This is an interesting point, but it should be explored in a larger study involving a heavily optimized implementation on very large data sets. It is crucial, when comparing training times, to compare them on large data sets; for small data sets, all training times are short, so relative differences in training times are unimportant from a practical standpoint. Furthermore, differences on small training sets are not necessarily indicative of what will happen as larger problems are considered. In Hsu and Lin's study, the largest two problems are `letter`, with 15,000 training points in 26 dimensions, and `shuttle`, with 43,500 training points in 7 dimensions. For `letter`, the OVA approach trained 6 times slower than the AVA or DAG approaches, but for the much larger `shuttle`, the difference was only about 15%. Again, this indicates that a larger study with a more heavily optimized implementation would be necessary to untangle issues concerning the relative training times. However, it does seem likely that for large-scale problems, AVA will enjoy a speed advantage over OVA.

3.2.7 Summary

In the first paper exploring the use of error-correcting code approaches to multiclass classification, Dietterich and Bakiri were already fully aware of both the promise and the difficulty of this approach, which obviously relies heavily on the errors produced by different binary classifiers being (at least somewhat) decorrelated. In a companion paper, they address this issue for the specific case where the underlying binary classifiers are decision trees (Kong and Dietterich, 1995). We believe (and show experimentally in Section 4) that when the underlying classifiers are appropriately tuned regularization systems such as SVM or RLSC, that the errors produced by different binary classifiers will be extremely highly correlated, implying that a one-vs-all scheme will be as effective as an error-correcting code scheme. Furthermore, we will show that across several data sets, using SVM (or RLSC) in a one-vs-all framework yields results as good as error-correcting approaches.

4. Experimental Work

In previous work (Rifkin, 2002), we found that a simple OVA approach was extremely effective at multiclass classification. However, those experiments were on relatively few data sets, and furthermore, the kernel parameters were "illegally" tuned to the data sets. Therefore, we decided to perform a much larger set of more carefully controlled experiments, in order to better gauge the actual differences between multiclass data schemes. Fürnkranz (2002) reported that an AVA scheme substantially outperformed an OVA scheme over a wide variety of data sets; we decided to focus specifically on those data sets that Fürnkranz had found a large difference on. These data sets are all publicly available as part of the UCI Machine Learning Repository (Merz and Murphy, 1998). We tested the five error-correcting coding schemes suggested by Allwein et al. (2000): a one-vs-all scheme (OVA),

[13] We note that the freely available SVM solver SvmFu (`http://fpn.mit.edu/SvmFu`) *does* share kernel products between SVMs while performing multiclass training (or testing).

an all-vs-all or all-pairs scheme (AVA), a complete code (COM), a dense code (DEN), and a sparse code (SPA). For further descriptions of these codes, see Section 3.2.2 or the paper by Allwein et al. (2000). In some cases, experiments could not be run because they were too computationally expensive.

4.1 Experimental Protocol

Of the data sets found by Fürnkrnaz to have a large performance difference between OVA and AVA schemes, we selected ten on which to perform experiments: `soybean-large`, `letter`, `satimage`, `abalone`, `optdigits`, `glass`, `car`, `spectrometer`, `yeast`, and `page-blocks`. For all data sets which have test sets, we use only the given train/test split (Fürnkranz tested three data sets in both the original train/test and cross-validation settings). We omitted the `covertype` data set because its very large test set made it unwieldy to work with, and we omitted the `vowel` data set because, although Fürnkranz found a significant performance difference under cross-validation, this difference disappeared under the original train/test split.[14]

By 10-fold cross-validation (CV), we refer to randomly breaking a data set into ten equal-sized (as closely as possible) subsets, respecting as closely as possible the class percentages in the original data, and then considering the ten train/test splits obtained by taking nine of the subsets as training data and the tenth as test data.

The only data set in our experiments which had any missing values was the `soybean-large` data set. For this data set, we first filled in missing elements in the data using the training set modes (all the attributes are nominal; we could use means if the attributes were numeric).

For data sets with a train/test split (`soybean-large`, `letter`, `satimage`, `abalone` and `optdigits`), each numeric attribute was normalized to have mean 0 and variance 1 on the training set (the same scaling was applied to the training and test sets, but the scaling was determined using only the training set). Finally, each nominal attribute taking on k different values was converted to k binary (0-1 valued) attributes, where the ith variable is set to 1 if and only if the nominal attribute takes on the ith possible value. The parameters γ and C (or, equivalently, λ for RLSC) were found by doing 10-fold cross-validation on the training set. We first set $C = 1$ and $\gamma = 1$ (a reasonable "rough guess" given the mean 0 variance 1 normalization), then increased or decreased γ by a factor of 2 until no improvements were seen for three consecutive attempts. Then γ was held fixed at the best value found and an identical optimization was performed over C. It would have been better to jointly optimize over C and γ, but this would have been computationally prohibitively expensive.

For data sets without a standard train/test split, we split the original data set using 10-fold CV, and then performed the procedure described in the previous paragraph. Note that because of this, different CV "splits" of a data set could end up with slightly different normalizations, C and γ values.

[14] It is interesting to note that for the `vowel` data set, the standard train/test split is based on the speaker. Under cross-validation, instances of the same speaker are mixed into both the training and test sets, and we might expect performance in this scenario to be very different: indeed, in Fürnkranz's study, all methods perform much better under cross-validation than under the original train/test split.

Basic information about the data sets is summarized in Table 4. The baseline error for each data set is the error rate for a classification scheme which always chooses the class containing the largest number of examples. The number of classifiers required for each scheme is shown in Table 5. The large number of classifiers required for the COM code indicates that this approach is not feasible for data sets with many classes. In Section 5, we will show that when RLSC is used as the underlying binary classifier, the OVA and the COM approaches will produce *identical* classification decisions, indicating that the inability to directly implement this scheme is of little concern.

The simulations were implemented using a modified version of the WEKA package (Witten and Frank, 1999). The multiclass schemes were implemented as a new WEKA classifier and integrated into the original package. The underlying binary SVMs were trained using the SVMTorch package (Collobert and Bengio, 2001). The binary RLSCs were implemented directly in Java.

4.2 Results

				# attributes /	average / min / max	baseline
Name	# train	# test	# classes	# nominal attr.	# examples per class	error (%)
soybean-large	307	376	19	35 / 35	16.2 / 1 / 40	87.2
letter	16000	4000	26	16 / 0	615.4 / 576 / 648	96.4
satimage	4435	2000	6	36 / 0	739.2 / 409 / 1059	77.5
abalone	3133	1044	29	8 / 1	108.0 / 0 / 522	84.0
optdigits	3823	1797	10	64 / 0	382.3 / 376 / 389	89.9
glass	214	-	7	9 / 0	30.6 / 0 / 76	64.5
car	1728	-	4	6 / 6	432.0 / 65 / 1210	30.0
spectrometer	531	-	48	101 / 0	11.1 / 1 / 55	89.6
yeast	1484	-	10	8 / 0	148.4 / 5 / 463	68.8
page-blocks	5473	-	5	10 / 0	1094.6 / 28 / 4913	10.2

Table 4: Data sets used for the experiments.

Name	OVA	AVA	COM	DEN	SPA
soybean-large	19	171	262143	43	64
letter	26	325	33554431	48	71
satimage	6	15	31	26	39
abalone	29	406	268435455	49	73
optdigits	10	45	511	34	50
glass	6	15	31	26	39
car	4	6	7	20	30
spectrometer	48	1128	1.407e+014	56	84
yeast	10	45	511	34	50
page-blocks	5	10	15	24	35

Table 5: Number of possible binary classifiers for each code matrix.

Tables 6 through 9 show a comparison between the OVA method and each of the methods AVA, DEN, SPA, and COM. For each data set, we show the error rates of each system, the

Data Set	AVA	OVA	DIFF	AGREE	BOOTSTRAP
soybean-large	6.38	5.85	0.530	0.971	[-0.008, 0.019]
letter	3.85	2.75	1.09	0.978	[0.008, 0.015]
satimage	8.15	7.80	0.350	0.984	[-5E-4, 0.008]
abalone	72.32	79.69	-7.37	0.347	[-0.102, -0.047]
optdigits	3.78	2.73	1.05	0.982	[0.006, 0.016]
glass	30.37	30.84	-.470	0.818	[-0.047, 0.037]
car	0.41	1.50	-1.09	0.987	[-0.016, -0.006]
spectrometer	42.75	53.67	-10.920	0.635	[-0.143, -0.075]
yeast	41.04	40.30	0.740	0.855	[-0.006, 0.021]
page-blocks	3.38	3.40	-.020	0.991	[-0.002, 0.002]

Table 6: SVM test error rate (%), OVA vs. AVA.

Data Set	DEN	OVA	DIFF	AGREE	BOOTSTRAP
soybean-large	5.58	5.85	-0.270	0.963	[-0.019, 0.013]
letter	2.95	2.75	0.200	0.994	[5E-4, 0.004]
satimage	7.65	7.80	-0.150	0.985	[-0.006, 0.003]
abalone	73.18	79.69	-6.51	0.393	[-0.092, -0.039]
optdigits	2.61	2.73	-0.12	0.993	[-0.004, 0.002]
glass	29.44	30.84	-1.40	0.911	[-0.042, 0.014]
car	-	1.50	-	-	-
spectrometer	54.43	53.67	-0.760	0.866	[-0.011, 0.026]
yeast	40.30	40.30	0.00	0.900	[-0.011, 0.011]
page-blocks	-	3.40	-	-	-

Table 7: SVM test error rate (%), OVA vs. DENSE.

difference in error rates, the percentage of data points on which the two classifiers agree (predict the same output class), and finally a 90% bootstrap confidence interval for the difference in performance of the two methods (see Appendix A).[15]

Our primary observation for tables 6 through 9 is that in nearly all cases the results of the two methods compared are very close. In a majority of the comparisons, 0 is in the bootstrap interval, meaning we cannot conclude that the classifiers are statistically significantly different. For many of the remaining comparisons, the bootstrap interval includes numbers which are quite small. The main counterexample seems to be the `abalone` data set: on this data set we find that the OVA method performs noticeably worse than any of the other methods (except COM, which we could not run for computational reasons). It is perhaps worth noting that on this data set, the performance in general is quite poor—although there are 29 classes, a single class contains 16% of the data, yielding a baseline error rate of 84%,

[15] As an additional check, we also ran McNemar's test for these experiments. With no exceptions, we found that McNemar's test reported a significant different (at the 5% level) between two classifiers if and only if the confidence interval for our bootstrap test did not include 0.

Data Set	SPA	OVA	DIFF	AGREE	BOOTSTRAP
soybean-large	6.12	5.85	0.270	0.968	[-0.011, 0.016]
letter	3.55	2.75	0.800	0.980	[0.005, 0.011]
satimage	8.85	7.80	1.05	0.958	[0.003, 0.018]
abalone	75.67	79.69	-4.02	0.352	[-0.067, -0.014]
optdigits	3.01	2.73	0.280	0.984	[-0.002, 0.008]
glass	28.97	30.84	-1.87	0.738	[-0.070, 0.033]
car	0.81	1.50	-0.69	0.988	[-0.011, -0.003]
spectrometer	52.73	53.67	-0.940	0.744	[-0.038, 0.019]
yeast	40.16	40.30	-0.140	0.855	[-0.015, 0.013]
page-blocks	3.84	3.40	0.440	0.979	[0.001, 0.007]

Table 8: SVM test error rate (%), OVA vs. SPARSE.

Data Set	COM	OVA	DIFF	AGREE	BOOTSTRAP
soybean-large	-	5.85	-	-	-
letter	-	2.75	-	-	-
satimage	7.80	7.80	0.00	0.999	[-1E-3, 1E-3]
abalone	-	79.69	-	-	-
optdigits	2.67	2.73	-0.060	0.996	[-0.003, 0.002]
glass	29.44	30.84	-1.340	0.911	[-0.042, 0.014]
car	1.68	1.50	-0.180	0.998	[5.79E-4, 0.003]
spectrometer	-	53.67	-	-	-
yeast	38.61	40.30	-1.690	0.906	[-0.028, -0.005]
page-blocks	3.49	3.40	-0.090	0.983	[-0.002, 0.004]

Table 9: SVM test error rate (%), OVA vs. COMPLETE.

and none of the systems achieve an error rate lower than 72%, indicating that this data set is very difficult to classify well. Additionally, on the `spectrometer` data set, the AVA method performs better than any of the remaining methods. On the remaining 8 of the 10 data sets, OVA performs essentially as well as the other methods.

Also interesting to note is the AGREE column of the classifiers. In many cases, the AGREE number is substantially higher than the error rates, indicating that not only do the classifiers make errors on the same point, *but that different systems often make identical "incorrect" predictions.* This is in keeping with the authors' intuition that when well-tuned SVMs are used as binary classifiers, points become errors not because of deficiencies in the method of combining binary classifiers, but simply because for all practical purposes, the points "look" more like a member of an incorrect class than their true class. These types of errors will in general be extremely difficult to correct.

It is critical to keep in mind that these results were obtained using only data sets on which Fürnkranz (2002) found a substantial difference in performance between OVA and AVA approaches. From this perspective, we feel that our results strongly support the

Data Set	FUR	OVA	DIFF	AGREE	BOOTSTRAP
soybean-large	13.3	5.85	7.45	0.891	[.056, .109]
letter	7.7	2.75	4.95	0.922	[.043, .057]
satimage	12.2	7.80	4.40	0.906	[.0345, .055]
abalone	74.1	79.69	-5.59	0.335	[-.083, -0.029]
optdigits	7.5	2.73	4.77	0.920	[0.035, 0.056]
glass	26.2	30.84	-4.64	0.734	[-0.098, 0.005]
car	2.8	1.50	1.3	0.969	[0.006, 0.020]
spectrometer	51.2	53.67	-2.47	0.488	[-0.060, 0.017]
yeast	41.6	40.3	1.29	0.765	[-0.005, -0.032]
page-blocks	2.6	3.40	-0.80	0.978	[-0.012, -0.005]

Table 10: SVM test error rate (%), OVA vs. Fürnkranz's R^3.

hypothesis that when the underlying binary classifiers are properly tuned SVMs, there is little difference between competing methods of multiclass classification. On the other hand, we fully expect that when weaker or poorly tuned binary classifiers are used, that combining the classifiers using an error-correcting approach will improve performance. The crucial question to the practitioner then becomes how to get the best performance. In Table 10 we compare our OVA SVM results to Fürnkranz's R^3 (essentially AVA) method. The code was obtained directly from Fürnkranz. Details of the comparison are given in Appendix B. We had to run our own experiments (rather than using Fürnkranz's numbers directly) because we needed the actual predictions at each data point in order to compute the AGREE and BOOTSTRAP columns of the table. Our results were similar to Fürnkranz's (error rates within 2%)[16] on 8 of the 10 data sets. We have no real explanation of the relatively large differences on `soybean` and `optdigits`.

Again, the primary impression from this table is that the methods are quite similar in performance; if anything, the SVMs seem to have a slight advantage. Certainly, these experiments do not support the idea that by intentionally using weaker classifiers and then combining them using an error-correcting approach, we can exceed the accuracy of an approach that begins with well-tuned binary classifiers and a simple OVA combination.

In Section 5, we present theoretical results on multiclass classification using RLSC as the underlying binary classifier. For this reason, it is important to gauge the empirical performance of RLSC on multiclass classification tasks. These results are presented in Tables 11 through 14. Note that because RLSC is substantially more computationally expensive than SVMs both to train and to test, we were only able to perform a subset of the SVM experiments; in particular, no experiments with the COM code were performed. In almost all cases, the differences between RLSC and SVM performance were very small. The vast majority of bootstrap intervals contained 0. Therefore, although the arguments we make in Section 5 apply directly to RLSC rather than SVMs, we can "infer experimentally" that the results are useful as well for describing SVM behavior.

[16] It is interesting that the difference in the choice of random seed can lead to differences in accuracy of 1% or so. This is in keeping with our impressions that differences in performance of this magnitude can often be caused by "nuisance" issues such as random seeds, tolerance and accuracy parameters, etc.

Data Set	RLSC	SVM	DIFF	AGREE	BOOTSTRAP
soybean-large	6.12	5.85	0.270	0.984	[-0.008, 0.013]
letter	-	2.75	-	-	-
satimage	7.9	7.80	0.010	0.979	[-0.004, 0.006]
abalone	72.7	79.69	-7.000	0.284	[-0.099, -0.041]
optdigits	2.5	2.73	-0.230	0.980	[-0.007, 0.003]
glass	31.3	30.84	0.460	0.808	[-0.037, 0.047]
car	2.9	1.50	1.40	0.980	[0.009, 0.020]
spectrometer	52.3	53.67	-1.370	0.821	[-0.036, 0.009]
yeast	40.0	40.30	-0.300	0.872	[-0.016, 0.011]
page-blocks	3.25	3.40	-0.150	0.983	[-0.004, 0.001]

Table 11: OVA test error rate (%), RLSC vs. SVM.

Data Set	RLSC	SVM	DIFF	AGREE	BOOTSTRAP
soybean-large	8.2	6.38	1.820	0.941	[0.000, 0.037]
letter	-	3.85	-	-	-
satimage	7.4	8.15	-.750	0.974	[-0.013, -0.001]
abalone	73.66	72.32	1.340	0.560	[-0.009, 0.034]
optdigits	3.0	3.78	-.780	0.974	[-0.013, -0.002]
glass	29.4	30.37	-0.970	0.864	[-0.047, 0.028]
car	2.3	0.41	1.89	0.980	[0.013, 0.024]
spectrometer	49.1	42.75	6.350	0.738	[0.036, 0.092]
yeast	40.0	41.04	-1.040	0.838	[-0.025, 0.005]
page-blocks	3.4	3.38	0.020	0.981	[-0.003, 0.003]

Table 12: AVA test error rate (%), RLSC vs. SVM.

The parameter settings for SVM and RLSC, found via cross-validation over the training set, are given in Appendix C.

5. Multiclass Classification with RLSC: Theoretical Arguments

In this section, we make some simple yet powerful arguments concerning multiclass classification with RLSC as the underlying binary classifier. Recall that to solve an RLSC problem, we solve a linear system of the form

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

Because this is a *linear* system, the vector $\mathbf{c}$ is a linear function of the right hand side $\mathbf{y}$. Define the vector $\mathbf{y^i}$ as

$$y_j^i = \begin{cases} 1 & \text{if } y_j = i, \\ 0 & \text{otherwise.} \end{cases}$$

Data Set	RLSC	SVM	DIFF	AGREE	BOOTSTRAP
soybean-large	8.0	5.58	2.41	0.971	[0.011, 0.040]
letter	8.0	7.65	0.350	0.976	[-0.002, 0.009]
abalone	72.8	73.18	-0.380	0.663	[-0.025, 0.017]
optdigits	2.5	2.61	-0.110	0.982	[-0.006, 0.003]
glass	29.9	29.44	-.460	0.864	[-0.037, 0.042]
car	-	-	-	-	-
spectrometer	52.9	54.43	-1.530	0.825	[-0.038, 0.008]
yeast	40.0	40.30	-.300	0.888	[-0.016, 0.009]
page-blocks	-	-	-	-	-

Table 13: DENSE test error rate (%), RLSC vs. SVM.

Data Set	RLSC	SVM	DIFF	AGREE	BOOTSTRAP
soybean-large	7.4	6.12	1.280	0.973	[0.000, 0.027]
letter	-	3.55	-	-	-
satimage	8.4	8.85	-0.450	0.958	[-0.011, 0.003]
abalone	73.3	75.67	-2.370	0.621	[-0.043, -0.005]
optdigits	3.6	3.01	0.590	0.977	[0.001, 0.011]
glass	29.4	28.97	-0.430	0.841	[-0.037, 0.047]
car	4.3	0.81	3.490	0.963	[0.028, 0.043]
spectrometer	52.5	52.73	-0.230	0.827	[-0.024, 0.021]
yeast	40.9	40.16	0.740	0.877	[-0.005, 0.020]
page-blocks	3.5	3.84	-0.340	0.980	[-0.006, 1.83E-4]

Table 14: SPARSE test error rate (%), RLSC vs. SVM.

Now, suppose that we solve N RLSC problems of the form

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

and denote the associated functions $f^{c_1}, \ldots, f^{c_N}$. Now, for *any* possible right hand side $\mathbf{y}^*$ for which the y_i and y_j are equal, whenever $\mathbf{x_i}$ and $\mathbf{x_j}$ are in the same class, we can calculate the associated $\mathbf{c}$ vector from the $\mathbf{c^i}$ *without solving a new RLSC system.* In particular, if we let m_i ($i \in \{1, \ldots, N\}$) be the y value for points in class i, then the associated solution vector $\mathbf{c}$ is given by

$$\mathbf{c} = \sum_{i=1}^{N} \mathbf{c^i} m_i.$$

For any code matrix M not containing any zeros, we can compute the output of the coding system using only the entries of the coding matrix and the outputs of the underlying one-vs-all classifiers. In particular, we do not need to actually train the classifiers associated with the coding matrix. We can simply use the appropriate linear combination of the "underlying" one-vs-all classifiers. For any $r \in \{1, \ldots, N\}$,

$$\begin{aligned}
& \arg \min_{r\in\{1,\dots,N\}} \sum_{i=1}^{F} L(M_{ri} f_i(\mathbf{x})) \\
= & \arg \min_{r\in\{1,\dots,N\}} \sum_{i=1}^{F} L(M_{ri} \sum_{j=1}^{N} M_{ji} f^j(\mathbf{x})) \\
= & \arg \min_{r\in\{1,\dots,N\}} \sum_{i=1}^{F} (M_{ri} - \sum_{j=1}^{N} M_{ji} f^j(\mathbf{x}))^2 \\
= & \arg \min_{r\in\{1,\dots,N\}} \sum_{i=1}^{F} (1 - 2M_{ri} \sum_{j=1}^{N} M_{ji} f^j(\mathbf{x}) + \sum_{j=1}^{N} \sum_{k=1}^{N} M_{ji} M_{ki} f^j(\mathbf{x}) f^k(\mathbf{x})) \\
= & \arg \min_{r\in\{1,\dots,N\}} \sum_{i=1}^{F} (-M_{ri} \sum_{j=1}^{N} M_{ji} f^j(\mathbf{x})) \\
= & \arg \min_{r\in\{1,\dots,N\}} \sum_{j=1}^{N} D_{rj} f^j(\mathbf{x}) \\
= & \arg \min_{r\in\{1,\dots,N\}} -F f^r(\mathbf{x}) + \sum_{\substack{j=1 \\ j\neq r}} D_{rj} f^j(\mathbf{x}), \qquad (2)
\end{aligned}$$

where we define define $D_{ri} \equiv -\sum_{j=1}^{F} M_{jr} M_{ji}$, and note that $D_{ii} = -F$ for all i.

We define a coding matrix to be *class-symmetric* if D_{ij} (and therefore C_{ij} as well) is independent of i and j (assuming $i \neq j$). Note that a sufficient (but not necessary) condition for a coding-matrix to be class-symmetric is if, whenever it contains a column containing k 1's and $n-k$ -1's, all $\frac{N!}{k!(N-k)!}$ such columns are included. The OVA and COMPLETE schemes are class-symmetric, while the DENSE scheme is in general not (the AVA and SPARSE schemes include zeros in the coding matrix, and are not addressed by this analysis). Other schemes generated by means of explicit error-correcting codes may be class-symmetric in this sense as well.

For class-symmetric schemes, D_{rj} is independent of the choice of r and j, and can be denoted simply as D^*. For these schemes, noting that $D^* > -F$ (assuming no identical rows in M),

$$\begin{aligned}
f(\mathbf{x}) &= \arg \min_{r\in 1,\dots,N} -F f^r(\mathbf{x}) + D^* \sum_{\substack{j=1 \\ j\neq r}}^{N} f^j(\mathbf{x}) \\
&= \arg \max_{r\in 1,\dots,N} f^r(\mathbf{x}).
\end{aligned}$$

We have shown that *when RLSC is used as the underlying binary learner for class-symmetric coding matrices containing no zeros, the predictions generated are identical to those of the one-vs-all scheme.*

For matrices which are not class-symmetric, we cannot use the above argument directly. However, we believe that Equation 2 is still highly indicative. If the D_{rj} are all close to equal (which is generally the case for the matrices generated), we can see that if $f^i(\mathbf{x})$ is substantially larger than $f^j(\mathbf{x})$ for $j \neq i$, then i will be chosen by the multiclass RLSC system. Although this notion could potentially be quantified, it would probably not provide much additional insight.

It is important to note that the above arguments *only* apply to schemes in which all the data is used for every classifier, and there are no zeros in the coding matrix. This includes the OVA, DEN, and COM schemes of Allwein et al., but not their AVA or SPA schemes. Whether, in practice, the addition of zeros to the coding scheme can make a big difference is an interesting question. The experimental results of this paper seem to indicate that for most data sets, there is no difference, but on some small data sets with many classes and high error rates, there may be a modest improvement.

6. Discussion and Conclusions

It is hoped that this work will help practitioners. The viewpoint presented in this paper is that the most important step in good multiclass classification is to use the best binary classifier available. Once this is done, it seems to make little difference what multiclass scheme is applied, and therefore a simple scheme such as OVA (or AVA) is preferable to a more complex error-correcting coding scheme or single-machine scheme.

There seems to be some evidence that on small data sets with large numbers of classes (in our experiments, `abalone` and `spectrometer`) where the overall error rate is high, an AVA scheme can offer a moderate performance boost over an OVA scheme. However, this evidence is not strong—on the majority of data sets, we found the performance of all the schemes to be essentially identical. On the data sets where AVA did show an improvement over OVA, the error rates of all approaches were quite high, bringing into question the suitability of the task and representation to machine learning approaches.

We have also presented, using RLSC as the underlying binary classifiers, what we believe is the beginning of a theory demonstrating why simple approaches may perform just as well as error-correcting approaches. In Section 4 we showed empirically that across a number of data sets, RLSC and SVMs produce very similar results, thereby motivating the use of RLSC as a theoretical tool to study approaches to multiclass classification.[17] When Dietterich and Bakiri wrote their paper introducing error-correcting coding approaches (Dietterich and Bakiri, 1995), they were already aware that these methods would be useful only if the correlation between different binary classifiers was not too high. But the use of RLSC as the underlying classifier, with its simple linear structure, makes clear that the correlation is extremely high—once we've trained the OVA RLSC classifiers, we already know the RLSC outputs for any classifier that uses all the classes. It would be nice to extend this result to a situation where not every class is used by each binary classifier (for example, the AVA scheme), although how to do so is an open question.

This paper has focussed on multiclass classification accuracy, but a brief word on speed is in order. Comparing different machine learning algorithms for speed is notoriously difficult; we are no longer judging mathematical algorithms but are instead judging specific implementations. However, some possibly useful general observations can be made. Empir-

[17] We are not suggesting the use of RLSC in practice, because on nonlinear problems, it is much slower than the SVM.

ically, SVM training time tends to be superlinear in the number of training points. Armed only with this assumption, it can be shown (see the analysis by Fürnkranz (2002) for an in-depth discussion) that an OVA scheme will train more slowly than an AVA scheme. This is certainly observed by Hsu and Lin (2002); however, on their largest data set (`shuttle`, 43,500 training points), they observe approximately a 15% difference in training times, whereas for their second largest data set (`letter`, 15,000 training points), OVA is six times slower, indicating that the size of the data set alone is not highly predictive of the size of the difference.

In many applications, the time required to test is of far greater importance than the time required to train. It is clear that, for any training scheme discussed here except the DAG scheme, the testing time is directly proportional to the number of unique support vectors. Hsu and Lin (2002) present the average number of unique SVs for their experiments, and the OVA scheme is often the worst performing from this perspective, but the differences are often not large. Nevertheless, it appears that from a speed perspective, an AVA or DAG scheme will likely have the advantage. Further work, especially on very large data sets using heavily-optimized implementations, could be useful here.

Additonally, we note that our belief that an OVA scheme is as accurate as any other scheme is predicated on the assumption that the classes are "independent"—that the classes do not belong to a natural hierarchy, and that we do not necessarily expect examples from class "A" to be closer to those in class "B" than those in class "C". In the latter situation, especially when few examples were available, we might suspect that an algorithm which exploited the relationships between classes could offer superior performance. This is an interesting open question.

Finally, we do not wish to overstate our claims. In particular, all of our experiments were done on data sets from the UCI repository, and some might well view these data sets as "toy data sets". In particular, it is unknown how OVA will compare to other schemes on very large, difficult problems that include label noise. Rather than view our experiments as proof that OVA is necessarily an ideal method in all possible situations, we instead view our experiments as demonstrating that there is no compelling evidence to date that either error-correcting coding or single-machine approaches outperform OVA when the underlying binary classifiers are properly tuned. This is interesting because it is in contrast to the work of Allwein et al. (2000) and Fürnkranz (2002), who concluded (using the *same* data sets that were used in this paper), respectively, that error-correcting approaches and AVA strongly outperformed OVA; in our own experiments, we show that when more care is taken to use properly tuned binary classifiers, the performance of all methods improves, and the difference in performance between methods largely disappears. While of course it remains open what will happen on more difficult tasks, we believe the burden should be on the developer of a new multiclass algorithm to show that it outperforms a naive approach such as OVA. To date, neither the single-machine approach nor the error-correcting code approach has resulted in an algorithm that demonstrably outperforms OVA in terms of accuracy. We welcome future work that offers practitioners a better algorithm than OVA—whether it is an entirely new algorithm, or a demonstration that any of the currently known algorithms outperforms OVA on more difficult data sets.

Appendix A. A Simple Bootstrap Approach to Comparing Classifiers

We briefly discuss McNemar's statistical test (McNemar, 1947, Everitt, 1977), and also present a simple bootstrap approach for comparing two different classifiers ("A" and "B") that were tested on the same test set. In both cases, by examining the paired outputs, we compute the empirical probabilities of the following four events over an individual data point x:

- both classifiers were correct on (CC),
- both classifiers were incorrect (II),
- A was correct, B incorrect (CI),
- B was correct, A incorrect (IC).

We let $n(CC)$ denote the number of observations falling into "class" CC, and similarly for the other "classes." McNemar's test rests on the assumption that under the null hypothesis (classifiers "A" and "B" have equivalent Bayes error rates), $n(CI)$ (as well as $n(IC)$) is binomially distributed with $p = 1/2$ and $N = n(IC) + n(CI)$. In practice, the test is carried out by means of a χ^2-approximation.

Instead of (or in addition to) using McNemar's test, we can use the following simple bootstrap procedure. We generate a large number (in our experiments, 10,000) of standard bootstrap data sets: samples of size ℓ where each data point independently belongs to the "class" CC, II, CI, or IC with probability equal to the empirical probability of the associated "class" on the original data set. For each bootstrap sample, we compute the difference in performance of the two classifiers. We generate a $k\%$ confidence interval (in our experiments, $k = 90\%$) by reporting the $\frac{1-k}{2}$'th and $\frac{1+k}{2}$'th percentiles of this distribution.

We feel that this technique represents an improvement over McNemar's test for two reasons. The first is that it takes into account the number of examples on which the two classifiers agree (and therefore the total size of the data set): with McNemar's test, only the number of points contributing to events CI and IC matter. Suppose we consider two scenarios. In the first scenario, there are 100 data points, and classifier "A" gets them all right and classifier "B" gets them all wrong. In the second scenario, there are 1,000,000 points, both classifiers get 999,900 points right, classifier "A" gets the remaining points right and classifier "B" gets them wrong. As far as McNemar's test is concerned, these two situations are *identical*.

The second, more important issue is that McNemar's test provides only an indication of whether the two classifiers are significantly different, but provides no indication of the *size* of the difference. This is obviously of crucial importance to practitioners—even if two methods are significantly different, if the actual difference in performance is quite small, a practitioner may well choose to implement the method which is less computationally intensive.

It is important to note that both this technique and McNemar's test require the paired predictions of the classifiers, and neither can be used if only the absolute error rates of the two systems are available.

Data Set	Current Experiments	Furnkranz's paper
soybean-large	13.3	6.30
letter	7.7	7.85
satimage	12.2	11.15
abalone	74.1	74.34
optdigits	7.5	3.74
glass	26.2	25.70
car	2.8	2.26
spectrometer	51.2	53.11
yeast	41.6	41.78
page-blocks	2.6	2.76

Table 15: Results for Furnkranz's R^3 ("double" all-pairs) using Ripper as the base learner.

Appendix B. Fürnkranz's Experiments Revisited

In this appendix, we describe attempts to "replicate" the results of Fürnkranz (2002). There are at least two good reasons for doing this. First, we wanted to see to what extents the results *were* replicable. Secondly, we wanted to obtain actual predictions of his algorithm on each data set, in order to calculate bootstrap confidence intervals against our other predictors (see Appendix A). Fürnkranz was kind enough to make his code available to us; we performed our own preprocessing as described in Section 4.1.

The results are shown in Table 15. The results are quite close on many of the data sets. The numbers are not identical because the Ripper (Cohen, 1995) implementation uses random numbers internally, and our random seed and Fürnkranz's differ—on our own machine, repeated runs give identical results. It is unclear what is causing the relatively large discrepancies on the `soybean` and `optdigits` data sets.

We are unable to compute bootstrap confidence intervals on the difference in performance between Fürnkranz's original results and our attempts to reproduce these results because we do not have his actual predictions available.

We were able to get broad agreement with Fürnkranz on the majority of the data sets. It is worth noting that in general, differing choices of the random seed seem to lead to differences in performance on the order of 1%; because many of the differences in performance between different classification systems on these data sets are of the same order, this is further evidence that such differences must be carefully tested for significance.

Appendix C. SVM and RLSC Parameter Settings

In this appendix we present the parameters selected for our experiments. All parameters were selected via 10-fold cross-validation. For each experiment, the Gaussian parameter γ was first selected, then the regularization parameter (C for SVMs, λ for RLSC) was selected. For those experiments with no test set, we performed 10-fold cross-validation on each of the 10 "train/test" splits; we did not constrain each split to have the same parameters. For

Data Set	OVA	AVA	DEN	SPA	COM
soybean-large	4	32	2	16	-
letter	4	32	4	16	-
satimage	4	4	2	16	4
abalone	2	4	4	4	-
optdigits	2	8	4	32	2
glass	8.4 (91.8)	26.0 (489.6)	16.4 (583.8)	11.2 (25.0)	18.0 (1.4e3)
car	3.2 (1.0)	6.8 (3.4)	-	10.0 (55.2)	3.6 (3.0)
spectrometer	3.0 (3.4)	46.4 (1.1E3)	5.2 (20.2)	5.8 (26.8)	-
yeast	1.4 (1.4)	5.8 (33.4)	3.0 (3.9)	5.0 (81.3)	2.0 (0.6)
page-blocks	36.0 (553.6)	82.4 (4.81E3)	-	50.4 (1.1E3)	14.6 (296.0)

Table 16: SVM configuration: the C parameter.

Data Set	OVA	AVA	DEN	SPA	COM
soybean-large	2^{-7}	2^{-7}	2^{-5}	2^{-6}	-
letter	2^{-1}	2^{-1}	2^{-1}	2^{-1}	-
satimage	2^{-2}	2^{-2}	2^{-2}	2^{-3}	2^{-2}
abalone	2^{4}	2^{-2}	2^{-1}	2^{-1}	-
optdigits	2^{-5}	2^{-6}	2^{-5}	2^{-6}	2^{-5}
glass	$2^{-1.8}$ (0.1)	$2^{-2.9}$ (1.4e-3)	$2^{-2.7}$ (4.1e-3)	$2^{-1.3}$ (1.5e-2)	$2^{-2.1}$ (3.4e-2)
car	2^{-3} (0.0)	2^{-4} (0.0)	-	2^{-4} (0.0)	2^{-3} (0.0)
spectrometer	$2^{-3.6}$ (8.2E-4)	2^{-6} (3.7E-5)	$2^{-3.9}$ (9.3E-4)	$2^{-3.9}$ (1.1E-3)	-
yeast	$2^{-1.2}$ (1.3E-2)	$2^{-3.5}$ (1.3E-3)	$2^{-1.5}$ (1.5E-2)	$2^{-3.1}$ (7.9E-4)	$2^{-1.8}$ (2.2E-2)
page-blocks	$2^{-2.4}$ (2.5E-2)	$2^{-2.9}$ (6.6E-3)	-	$2^{-1.2}$ (1.3E-2)	$2^{-0.8}$ (9.7E-2)

Table 17: SVM configuration: γ for the Gaussian kernel.

Data Set	OVA	AVA	DEN	SPA
soybean-large	0.5	2.0	2.0	2.0
letter	-	-	-	-
satimage	0.2	0.2	0.2	0.2
abalone	0.2	16.0	2.0	16.0
optdigits	0.1	1.56E-2	3.12E-2	0.1
glass	2.8 (20.0)	2.4 (20.8)	2.0 (5.2)	0.2 (3.3E-2)
car	0.2 (7.12E-3)	1.38E-2 (3.39E-4)	-	-
spectrometer	3.5 (18.8)	0.5 (1.4)	5.1 (84.6)	0.9 (0.2)
yeast	7.1 (88.4)	17.4 (559.2)	28.1 (1.68E3)	17.4 (559.2)
page-blocks	1.1 (0.6)	1.1 (9E-2)	-	1.0 (0.1)

Table 18: RLSC configuration: $\lambda * \ell$.

these data sets, we report the mean and variance of the parameters. The results are shown in Tables 16 to 18.

It is worth noting that many of the variances are quite large relative to the means, indicating that small changes in the data set (each pair of "training folds" shares 80% of the total data set) can lead to relatively large changes in the parameter settings. We tend to believe that changes of this magnitude do not induce a *large* difference in classifier accuracy (with SVMs or RLSC, parameter settings that are "close" will give classifier performances that are "close"), but this is a question for further study.

Data Set	OVA	AVA	DEN	SPA
soybean-large	2^{-7}	2^{-2}	2^{-2}	2^{-2}
letter	-	-	-	-
satimage	2^{-2}	2^{-1}	2^{-2}	2^{0}
abalone	2^{-4}	2^{-5}	2^{-3}	2^{-3}
optdigits	2^{-6}	2^{-4}	2^{-5}	2^{-4}
glass	$2^{2.5}$ (88.1)	$2^{-0.9}$ (0.2)	$2^{-1.5}$ (0.1)	$2^{-0.5}$ (6E-2)
car	$2^{-3.6}$ (8.2E-4)	$2^{-3.0}$ (0.0)	-	-
spectrometer	$2^{-3.5}$ (4.19E-3)	$2^{-3.9}$ (9.37E4)	$2^{-3.4}$ (3.91E-3)	$2^{-2.9}$ (1.78E-2)
yeast	$2^{-1.5}$ (6.7E-2)	$2^{-0.6}$ (0.4)	$2^{-1.3}$ (6.58E-2)	$2^{-0.6}$ (0.4)
page-blocks	$2^{1.0}$ (0.6)	$2^{-2.8}$ (3.16E-3)	-	$2^{-0.4}$ (8.06E-2)

Table 19: RLSC configuration: γ for the Gaussian kernel.

References

E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.

R. C. Bose and D. K. Ray-Chaudhuri. On a class of error-correcting binary group codes. *Information and Control*, 3:68–79, 1960.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *5th Annual ACM Workshop on COLT*, pages 144–152, 1992.

O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

E. Bredensteiner and K. P. Bennett. Multicategory classification by support vector machines. In *Computational Optimizations and Applications*, volume 12, pages 53–79, 1999.

W. W. Cohen. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, 1995.

R. Collobert and S. Bengio. SVMTorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, 2001.

K. Crammer and Y. Singer. Improved output coding for classification using continuous relaxation. In *Proceedings of the Thirteenth Annual Conference on Neural Information Processing Systems*, 2000a.

K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. In *Computational Learning Theory*, pages 35–46, 2000b.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-basd vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. *Machine Learning*, 47(2):201–233, 2002.

T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1998.

T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

B. Everitt. *The analysis of contingency tables.* Chapman and Hall, 1977.

Theodoros Evgeniou, Massimiliano Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances In Computational Mathematics*, 13(1):1–50, 2000.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

G. Fung and O. L. Mangasarian. Proximal support vector classifiers. In Provost and Srikant, editors, *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 77–86. ACM, 2001a.

G. Fung and O. L. Mangasarian. Proximal support vector machine classifiers. Technical report, Data Mining Institue, 2001b.

J. Fürnkranz. Round robin classification. *Journal of Machine Learning Research*, 2:721–747, 2002.

C. F. Gauss. Theoria combinationis obsevationum erroribus minimis obnoxiae. *Werke*, 1823.

F. Girosi and T. Poggio. Networks and the best approximation property. Technical Report A.I. Memo No. 1164, C.B.C.L Paper No. 45, Massachusetts Institute of Technology, Artificial Intelligence Laboratory and Center for Biological and Computational Learning, Department of Brain and Cognitive Sciences, October 1989.

Y. Guermeur. Combining discriminant models with new multi-class svms. *Pattern Analysis and Applications*, 5(2):168–179, 2002.

T. Hastie and R. Tibshirani. Classification by pairwise coupling. *The Annals of Statistics*, 26(2):451–471, 1998.

C. Hsu and C. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13:415–425, 2002.

T. Joachims. Making large-scale SVM learning practical. Technical Report LS VIII-Report, Universität Dortmund, 1998.

E. B. Kong and T. G. Dietterich. Why error-correcting output coding works with decision trees. Technical report, Department of Computer Science, Oregon State University, Corvallis, OR, 1995.

Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines. Technical Report 1043, Department of Statistics, University of Wisconsin, 2001a.

Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines. In *Proceedings of the 33rd Symposium on the Interface*, 2001b.

Y. Lin. Support vector machines and the Bayes rule in classification. Technical Report Technical Report Numberr 1014, Department of Statistics, University of Wisconsin, 1999.

Q. McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12:153–157, 1947.

C. J. Merz and P. M. Murphy. UCI repository of machine learning databases. http://www.ics.uci.edu/ mlearn/MLRepository.html, 1998.

S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In Hu, Larsen, Wilson, and Douglas, editors, *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing IX*, pages 41–48, 1999.

E. Osuna. *Support Vector Machines: Training and Applications*. PhD thesis, Massachusetts Institute of Technology, 1998.

E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. In *Proceedings of the 1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 130–136, Los Alamitos, CA, USA, June 1997. IEEE Computer Society Technical Committee on Pattern Analysis and Machine Intelligence, IEEE Computer Society.

J. C. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classification. In *Advances in Neural Information Processing Systems*, volume 12, pages 547–553. MIT Press, 2000.

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

J. R. Quinlan. *C4.5: Programs for Empirical Learning*. Morgan Kaufmann, 1993.

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

C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the 15th International Conference on Machine Learning*, 1998.

R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5): 1651–1686, 1998.

R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

I. Schönberg. Spline functions and the problem of graduation. *Proceedings of the National Academy of Science*, pages 947–950, 1964.

T. J. Sejnowski and C. R. Rosenberg. Parallel networks that learn to pronounce English text. *Journal of Complex Systems*, 1(1):145–168, 1987.

J. A. K. Suykens, L. Lukas, P. Van Dooren, B. De Moor, and J. Vandewalle. Least squares support vector machine classifiers: a large scale algorithm. In *Proceedings of the European Conference on Circuit Theory and Design*, 1999.

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

J. A. K. Suykens and J. Vandewalle. Multiclass least squares support vector machines. In *Proceedings of the International Joint Conference on Neural Networks*, 1999b.

A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-posed Problems.* W. H. Winston, 1977.

V. N. Vapnik. *Statistical Learning Theory.* John Wiley and Sons, 1998.

G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics.* Society for Industrial and Applied Mathematics, 1990.

G. Wahba and G. Kimeldorf. Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis Applications*, 33(1):82–95, 1971.

J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, Royal Holloway, University of London, Department of Computer Science, 1998.

I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufmann, 1999.