

The role of dimensionality reduction in linear classification

Weiran Wang Miguel Á. Carreira-Perpiñán

Electrical Engineering and Computer Science, University of California, Merced
<http://eecs.ucmerced.edu>

May 25, 2014

Abstract

Dimensionality reduction (DR) is often used as a preprocessing step in classification, but usually one first fixes the DR mapping, possibly using label information, and then learns a classifier (a filter approach). Best performance would be obtained by optimizing the classification error jointly over DR mapping and classifier (a wrapper approach), but this is a difficult nonconvex problem, particularly with nonlinear DR. Using the method of auxiliary coordinates, we give a simple, efficient algorithm to train a combination of nonlinear DR and a classifier, and apply it to a RBF mapping with a linear SVM. This alternates steps where we train the RBF mapping and a linear SVM as usual regression and classification, respectively, with a closed-form step that coordinates both. The resulting nonlinear low-dimensional classifier achieves classification errors competitive with the state-of-the-art but is fast at training and testing, and allows the user to trade off runtime for classification accuracy easily. We then study the role of nonlinear DR in linear classification, and the interplay between the DR mapping, the number of latent dimensions and the number of classes. When trained jointly, the DR mapping takes an extreme role in eliminating variation: it tends to collapse classes in latent space, erasing all manifold structure, and lay out class centroids so they are linearly separable with maximum margin.

1 Introduction

Dimensionality reduction (DR) is a common preprocessing step for classification and other tasks. Learning a classifier on low-dimensional inputs is fast (though learning the DR itself may be costly). More importantly, DR can help learn a better classifier, particularly when the data does have a low-dimensional structure, and with small datasets, where DR has a regularizing effect that can help avoid overfitting. The reason is that DR can remove two types of “noise” from the input: (1) independent random noise, which is uncorrelated with the input and the label, and mostly perturbs points away from the data manifold. Simply running PCA, or other unsupervised DR algorithm, with an adequate number of components, can achieve this to some extent. (2) Unwanted degrees of freedom, which are possibly nonlinear, along which the input changes but the label does not. This more radical form of denoising requires the DR to be informed by the labels, of course, and is commonly called supervised DR.

Call \mathbf{F} the DR mapping, which takes an input $\mathbf{x} \in \mathbb{R}^D$ and projects it to $L < D$ dimensions, and \mathbf{g} the classifier, which applies to the low-dimensional vector $\mathbf{F}(\mathbf{x})$ and produces a label y , so that the overall classifier is $\mathbf{g} \circ \mathbf{F}$. The great majority of supervised DR algorithms are “filter” approaches (Kohavi and John, 1998; Guyon and Elisseeff, 2003), where one first learns \mathbf{F} from the training set of pairs (\mathbf{x}_n, y_n) , fixes it, and then train \mathbf{g} on the pairs $(\mathbf{F}(\mathbf{x}_n), y_n)$, using a standard classification algorithm as if the inputs were $\mathbf{F}(\mathbf{x})$. An example of supervised DR is linear discriminant analysis (LDA), which learns the best linear DR \mathbf{F} in the sense of minimal intra-class scatter and maximal across-class scatter. *The key in filter approaches is the design of a proxy objective function over \mathbf{F} that leads to learning a good overall classifier $\mathbf{g} \circ \mathbf{F}$.* Although the particulars differ among existing supervised DR methods (e.g. Belhumeur et al., 1997; Globerson and Roweis, 2006), usually they encourage \mathbf{F} to separate inputs or manifolds having different labels from each other. While this makes intuitive sense, and filter methods (and even PCA) can often do a reasonable job, it is clear that the best \mathbf{g} and \mathbf{F} are not obtained by optimizing a proxy function over \mathbf{F} and then having \mathbf{g} minimize the classification error (our real objective), but by *jointly* minimizing the latter over \mathbf{g} and \mathbf{F}

(the “wrapper” approach). However, filter approaches (particularly using a linear DR) are far more popular than wrapper ones. With filters, the classifier is learned as usual once \mathbf{F} has been learned. With wrappers, learning \mathbf{F} and \mathbf{g} involve a considerably more difficult, nonconvex optimization, particularly with nonlinear DR, having many more parameters that are coupled with each other. At this point, an important question arises: *what is the real role of (nonlinear) DR in classification, and how does it depend on the choice of mapping \mathbf{F} and of latent dimensionality L ?* Guided by this overall goal, the contributions of this paper are as follows. (1) We propose a simple, efficient, scalable and generic way of jointly optimizing the classifier’s loss over (\mathbf{F}, \mathbf{g}) , and in this paper we apply it to the case where \mathbf{F} is a RBF network and \mathbf{g} a linear SVM. (2) Armed with this algorithm, we study the role of nonlinear DR in the classifier’s performance and the latent space representation, and find lessons that apply to filter design. (3) We obtain a nonlinear low-dimensional SVM classifier that achieves state-of-the-art performance while being fast at test time.

A shorter version of this work appears in a conference paper (Wang and Carreira-Perpiñán, 2014).

2 Joint optimization of mapping and classifier using auxiliary coordinates

We describe first the approach for binary classification and focus on the case where \mathbf{g} is a linear SVM. We give the multiclass case in section 2.4. Given a training set of N input patterns $\mathbf{x}_n \in \mathbb{R}^D$ and corresponding labels $y_n \in \{-1, +1\}$, $n = 1, \dots, N$, we want to learn a nonlinear low-dimensional classifier $\mathbf{g} \circ \mathbf{F}$ that optimizes the following objective:

$$\begin{aligned} \min_{\mathbf{F}, \mathbf{g}, \xi} \lambda R(\mathbf{F}) + \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n \\ \text{s.t. } \{y_n(\mathbf{w}^T \mathbf{F}(\mathbf{x}_n) + b) \geq 1 - \xi_n, \xi_n \geq 0\}_{n=1}^N. \end{aligned} \quad (1)$$

This is the usual linear SVM objective of finding a separating hyperplane with maximum margin, but with inputs given by \mathbf{F} , which has a regularization term $\lambda R(\mathbf{F})$, where $\mathbf{g} = \{\mathbf{w}, b\}$ are the weights and bias of the linear SVM \mathbf{g} , and with a slack variable ξ_n per point n , where C is a penalty parameter for the slackness. The difficulty is that the constraints are heavily nonconvex because of the nonlinear mapping \mathbf{F} . However, the problem can be significantly simplified if we use the recently introduced *method of auxiliary coordinates (MAC)* (Carreira-Perpiñán and Wang, 2012, 2014) for nested systems. The idea is to introduce auxiliary variables that break nested functional dependences $\mathbf{g}(\mathbf{F}(\cdot))$ into simpler shallow mappings $\mathbf{g}(\mathbf{z})$ and $\mathbf{F}(\cdot)$. In our case, we introduce one auxiliary vector per input pattern and define the following constrained problem, which can be proven (Carreira-Perpiñán and Wang, 2012) to be equivalent to (1):

$$\begin{aligned} \min_{\mathbf{F}, \mathbf{g}, \xi, \mathbf{Z}} \lambda R(\mathbf{F}) + \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n \\ \text{s.t. } \{y_n(\mathbf{w}^T \mathbf{z}_n + b) \geq 1 - \xi_n, \xi_n \geq 0, \mathbf{z}_n = \mathbf{F}(\mathbf{x}_n)\}_{n=1}^N. \end{aligned} \quad (2)$$

This seems like a notational trick, but now we solve this with a quadratic-penalty method (Nocedal and Wright, 2006). We optimize the following problem for fixed penalty parameter $\mu > 0$ and drive $\mu \rightarrow \infty$:

$$\begin{aligned} \min_{\mathbf{F}, \mathbf{g}, \xi, \mathbf{Z}} \lambda R(\mathbf{F}) + \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n + \frac{\mu}{2} \sum_{n=1}^N \|\mathbf{z}_n - \mathbf{F}(\mathbf{x}_n)\|^2 \\ \text{s.t. } \{y_n(\mathbf{w}^T \mathbf{z}_n + b) \geq 1 - \xi_n, \xi_n \geq 0\}_{n=1}^N. \end{aligned} \quad (3)$$

This defines a continuous path $(\mathbf{F}^*(\mu), \mathbf{g}^*(\mu), \mathbf{Z}^*(\mu))$ which, under mild assumptions, converges to a minimum of the constrained problem (2), and thus to a minimum of the original problem (1) (Carreira-Perpiñán and Wang, 2012). Although problem (3) has more parameters, all the terms are simple and partially separable. The auxiliary vector \mathbf{z}_n can be interpreted as a target in latent space for the mapping \mathbf{F} , but these targets are themselves coordinated with the classifier. Using alternating optimization of (3) over $(\mathbf{F}, \mathbf{g}, \mathbf{Z})$ results

Algorithm 1 Nonlinear low-dimensional SVM using the method of auxiliary coordinates (MAC).

Input: dataset of points \mathbf{X} and their labels \mathbf{Y}

```

1: initialize  $\mathbf{Z}$ 
2: fit  $\mathbf{g}$  to  $(\mathbf{Z}, \mathbf{Y})$  by SVM training
3: fit  $\mathbf{F}$  to  $(\mathbf{X}, \mathbf{Z})$  by regression
4: for  $\mu = \mu_0 < \mu_1 < \dots < \mu_\infty$  do
5:   repeat
6:     optimize over  $\mathbf{Z}$  (algorithm 2)
7:     fit  $\mathbf{g}$  to  $(\mathbf{Z}, \mathbf{Y})$  by SVM training
8:     fit  $\mathbf{F}$  to  $(\mathbf{X}, \mathbf{Z})$  by regression
9:   until stop
10: end for
11: return  $\mathbf{F}$  and  $\mathbf{g}$ 

```

in very simple, convex steps. The (\mathbf{F}, \mathbf{g}) step is a usual RBF regression and linear SVM classification done independently from each other reusing existing, well-developed algorithms. The \mathbf{Z} -step has a closed-form solution for each \mathbf{z}_n separately. We describe the steps next. The complete alternating optimization procedure is given in Algorithm 1.

2.1 The \mathbf{g} step

For fixed (\mathbf{F}, \mathbf{Z}) , optimizing over \mathbf{w}, b, ξ is just training an ordinary linear SVM with low-dimensional inputs \mathbf{Z} . Much work exists on fast, scalable SVM training. We use LIBSVM (Chang and Lin, 2011). With K classes, each SVM can be trained independently of the others.

2.2 The \mathbf{F} step

For fixed (\mathbf{g}, \mathbf{Z}) , optimizing over \mathbf{F} is just a regularized regression with inputs \mathbf{X} and low-dimensional outputs \mathbf{Z} . So far the approach is generic over \mathbf{F} , which could be a deep net or a Gaussian process, for example, and we would use its corresponding training method within this step. However, we now focus on a special case which results in a particularly efficient step over \mathbf{F} , and which includes linear DR as a particular case. We use radial basis function (RBF) networks, which are universal function approximators, $\mathbf{F}(\mathbf{x}) = \mathbf{W}\Phi(\mathbf{x})$, with $M \ll N$ Gaussian RBFs $\phi_m(\mathbf{x}) = \exp(-\frac{1}{2}\|\mathbf{x} - \mathbf{c}_m\|^2/\sigma^2)$, and $R(\mathbf{F}) = \|\mathbf{W}\|^2$ is a quadratic regularizer on the weights. As commonly done in practice (Bishop, 2006), we determine the centers \mathbf{c}_m by k -means on \mathbf{X} , and then the weights \mathbf{W} have a unique solution given by a linear system. The total cost is $\mathcal{O}(M(L + D))$ in memory and $\mathcal{O}(NM(M + D))$ in training time, mainly driven by setting up the linear system for \mathbf{W} (involving the Gram matrix $\phi_m(\mathbf{x}_n)$); solving it exactly is a negligible $\mathcal{O}(M^3)$ since $M \ll N$ in practice, which can be reduced to $\mathcal{O}(M^2)$ by using warm-starts or caching its Cholesky factor. Note we only need to run k -means and factorize the linear system once and for all, since its input \mathbf{X} does not change. Thus, the \mathbf{F} -step simply involves a linear system for \mathbf{W} .

2.3 The \mathbf{Z} step

For fixed (\mathbf{F}, \mathbf{g}) , eq. (3) decouples on each n , so instead of one large problem on NL parameters, we have N independent small problems each on L parameters, of the form (omitting subindex n):

$$\begin{aligned}
& \min_{\mathbf{z}, \xi} \|\mathbf{z} - \mathbf{F}(\mathbf{x})\|^2 + c\xi \\
& \text{s.t. } y(\mathbf{w}^T \mathbf{z} + b) \geq 1 - \xi, \quad \xi \geq 0 \quad \mathbf{z} \in \mathbb{R}^L
\end{aligned} \tag{4}$$

where we have also included the slack variable, since we find this speeds up the alternating optimization. This is a convex quadratic program, whose closed-form solution is $\mathbf{z} = \mathbf{F}(\mathbf{x}) + \gamma y \mathbf{w}$, where γ is a scalar which takes one of three possible values, and costs $\mathcal{O}(L)$.

This can be seen using the KKT theorem. The Lagrangian of (4) is

$$\mathcal{L}(z, \xi, \lambda_1, \lambda_2) = \|\mathbf{z} - \mathbf{F}(\mathbf{x})\|^2 + c\xi - \lambda_1(y(\mathbf{w}^T \mathbf{z} + b) + \xi - 1) - \lambda_2 \xi \quad (5)$$

where λ_1 and λ_2 are Lagrange multipliers for the two inequality constraints. Its KKT system is

$$\begin{aligned} \nabla \mathcal{L}_{\mathbf{z}} = \mathbf{0} &\implies \mathbf{z} = \mathbf{F}(\mathbf{x}) + \frac{\lambda_1}{2} y \mathbf{w} \\ \nabla \mathcal{L}_{\xi} = 0 &\implies \lambda_1 + \lambda_2 = c \\ y(\mathbf{w}^T \mathbf{z} + b) &\geq 1 - \xi, \quad \xi \geq 0 \\ \lambda_1 &\geq 0, \quad \lambda_2 \geq 0 \\ \lambda_1(y(\mathbf{w}^T \mathbf{z} + b) + \xi - 1) &= 0, \quad \lambda_2 \xi = 0. \end{aligned}$$

From the first equation, the optimal \mathbf{z} lies on a line which passes through $\mathbf{F}(\mathbf{x})$ and is parallel to the normal direction of \mathbf{g} . If $\lambda_1 = 0$, we have $\lambda_2 = c$, $\xi = 0$, and the KKT system reduces to $\mathbf{z} = \mathbf{F}(\mathbf{x})$, $y(\mathbf{w}^T \mathbf{z} + b) \geq 1$. We have three cases.

Case 1 This means, if $y(\mathbf{w}^T \mathbf{F}(\mathbf{x}) + b) \geq 1$ is satisfied (i.e., $\mathbf{F}(\mathbf{x})$ is classified correctly with a margin of 1), we should simply set $\mathbf{z} = \mathbf{F}(\mathbf{x})$ and get an objective function value of 0 (the ideal case). This situation is demonstrated in fig. 1 (left plot).

Otherwise, we must have $\lambda_1 > 0$. The KKT system reduces to (using the relation $\lambda_2 = c - \lambda_1$):

$$\begin{aligned} \mathbf{z} &= \mathbf{F}(\mathbf{x}) + \frac{\lambda_1}{2} y \mathbf{w}, \quad y(\mathbf{w}^T \mathbf{z} + b) + \xi = 1 \\ \xi &\geq 0, \quad 0 < \lambda_1 \leq c, \quad (c - \lambda_1)\xi = 0 \end{aligned}$$

and from the first two equations we get

$$\begin{aligned} \xi &= 1 - y(\mathbf{w}^T \mathbf{F}(\mathbf{x}) + b) - \frac{\lambda_1}{2} \mathbf{w}^T \mathbf{w} \geq 0 \\ \implies \lambda_1 &\leq \frac{2(1 - y(\mathbf{w}^T \mathbf{F}(\mathbf{x}) + b))}{\mathbf{w}^T \mathbf{w}}. \end{aligned}$$

Substituting the above KKT conditions into the Lagrangian \mathcal{L} , we can express the dual of (4) using only λ_1 as

$$\begin{aligned} \max_{\lambda_1} & -\frac{1}{4} (\mathbf{w}^T \mathbf{w}) \lambda_1^2 + (1 - y(\mathbf{w}^T \mathbf{F}(\mathbf{x}) + b)) \lambda_1 \\ \text{s.t. } & \lambda_1 \leq \min \left(\frac{2(1 - y(\mathbf{w}^T \mathbf{F}(\mathbf{x}) + b))}{\mathbf{w}^T \mathbf{w}}, c \right). \end{aligned}$$

The objective function is a concave parabola with maximum achieved at $\tilde{\lambda}_1 = \frac{2(1 - y(\mathbf{w}^T \mathbf{F}(\mathbf{x}) + b))}{\mathbf{w}^T \mathbf{w}}$.

Case 2 Thus if $\tilde{\lambda}_1 < c$, the optimum of (4) is achieved by $\lambda_1 = \tilde{\lambda}_1$, $\lambda_2 = c - \tilde{\lambda}_1$, and $\mathbf{z} = \mathbf{F}(\mathbf{x}) + \frac{\tilde{\lambda}_1}{2} y \mathbf{w}$.

Case 3 Otherwise the optimum of (4) is achieved by $\lambda_1 = c$, $\lambda_2 = 0$, and $\mathbf{z} = \mathbf{F}(\mathbf{x}) + \frac{c}{2} y \mathbf{w}$.

By now, we have found the solutions to (4) completely with a cost of $\mathcal{O}(L)$. We give illustrations of the three possible cases in figure 1. This procedure is summarized in Algorithm 2.

2.4 Formulation for the multiclass problem

Let K be the number of classes. There are several ways to construct a classifier for K classes. We use the one-vs-all scheme, which has been shown to perform as well as any other variants of multiclass SVM (Rifkin and Klautau, 2004), and which gives rise to a simpler \mathbf{Z} step. In the one-vs-all scheme we have K binary SVMs, each of which is trained to classify whether a point belongs to some class or not. The decision

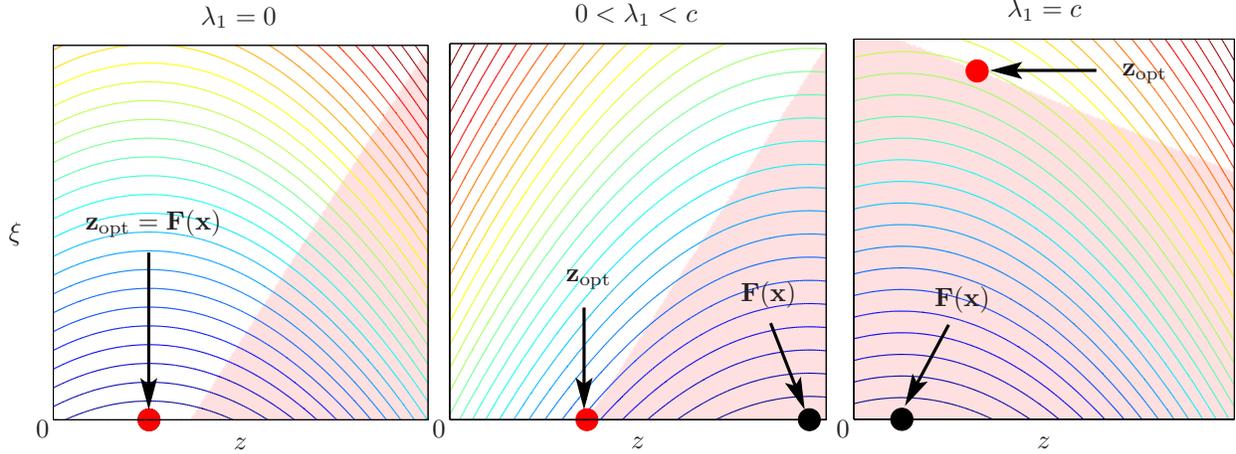


Figure 1: Three possible solutions of the quadratic program in the \mathbf{Z} step. The contours of the objective function are shown in color (increasing from blue to red) and the area not shaded denotes the feasible set. In each plot, the origin is in the bottom left corner, the black dot denotes $\mathbf{F}(\mathbf{x})$ and the red dot denotes the final solution \mathbf{z} .

Algorithm 2 \mathbf{Z} step optimization over $\{\mathbf{z}, \xi\}$.

Input: \mathbf{w} , b and c from \mathbf{g} , and $\mathbf{F}(\mathbf{x})$ from \mathbf{F}

- 1: $m = y(\mathbf{w}^T \mathbf{F}(\mathbf{x}) + b)$
 - 2: **if** $m \geq 1$ **then**
 - 3: $\mathbf{z} = \mathbf{F}(\mathbf{x})$, $\xi = 0$, $\lambda_1 = 0$
 - 4: **else**
 - 5: $\lambda_1 = 2(1 - m)/\mathbf{w}^T \mathbf{w}$
 - 6: **if** $\lambda_1 < c$ **then**
 - 7: $\mathbf{z} = \mathbf{F}(\mathbf{x}) + \frac{\lambda_1}{2} y \mathbf{w}$
 - 8: **else**
 - 9: $\mathbf{z} = \mathbf{F}(\mathbf{x}) + \frac{c}{2} y \mathbf{w}$
 - 10: **end if**
 - 11: $\xi = 1 - y(\mathbf{w}^T \mathbf{z} + b)$
 - 12: **end if**
 - 13: **return** \mathbf{z} and ξ
-

function on a test point \mathbf{x} is $\arg \max(\mathbf{w}_1^T \mathbf{x}, \dots, \mathbf{w}_K^T \mathbf{x})$, i.e., the final label is determined by the SVM with the largest decision value. The objective in (1) is replaced by the sum of the K SVMs' objective functions. In the \mathbf{Z} -step, we solve for each point a quadratic program of the following form (omitting subindex n):

$$\begin{aligned} \min_{\mathbf{z}, \{\xi^k\}_{k=1}^K} \quad & \|\mathbf{z} - \mathbf{F}(\mathbf{x})\|^2 + \sum_{k=1}^K C^k \xi^k \\ \text{s.t.} \quad & y^k ((\mathbf{w}^k)^T \mathbf{z} + b^k) \geq 1 - \xi^k, \quad \xi^k \geq 0, \quad \text{for } k = 1, \dots, K \end{aligned}$$

where C^k is the k th SVM's penalty parameter for the hinge loss and ξ^k is the slack variable of the k th SVM (associated with the point in consideration). This quadratic program contains $L + K$ variables and $2K$ inequality constraints. Since the size of this problem is typically not large, we use an active set algorithm for solving it (as implemented in Matlab's Optimization Toolbox). For the binary case, there exists only one SVM and the problem has a closed-form solution, as shown before.

It is also possible to use the one-vs-one scheme. With K classes, we have $\frac{K(K-1)}{2}$ binary SVMs, each trained on each pair of classes. The decision function on a test point is given by majority vote (i.e., the classifier that wins more times). This has a lower training time but a higher test time. Also, the one-vs-one

scheme involves a little more bookkeeping in the \mathbf{Z} step than the one-vs-all scheme. To see this, take for example $K = 10$ classes. The one-vs-all scheme uses 10 SVMs, and, in the \mathbf{Z} step, each data point is involved in all 10 SVMs, either as positive or negative example. The one-vs-one scheme uses 45 SVMs, and each point is involved in only 9 of them.

2.5 Summary and practicalities

Jointly optimizing the classification error over \mathbf{g} and \mathbf{F} becomes iterating simple convex subproblems: RBF regression, linear SVM and a closed-form update for \mathbf{Z} . Remarkably, we do not require any involved gradient computation, but simply reuse existing techniques for training \mathbf{F} and \mathbf{g} efficiently. Thus, although the problem (1) is nonconvex and has multiple local optima, the algorithm is deterministic given its initialization. We run the algorithm from an initial \mathbf{Z} . We observe that, given reasonable hyperparameters, even random values work well (section 3.1 shows how to construct a near-optimal initial \mathbf{Z}). In the first 1–2 iterations, the \mathbf{z}_n quickly reorganize in latent space, and they only get refined afterwards so as to enlarge the margin and reduce the bias. We use a initial value of $\mu = 2$ for the quadratic penalty parameter and increase it times 1.5 when the alternating scheme converges for fixed μ . We use early stopping, by exiting the iteration when the error in a validation set does not change or goes up. This helps to improve generalization and is faster: we observe a few iterations suffice, because each step updates very large, decoupled blocks of variables, and we need not drive $\mu \rightarrow \infty$ or achieve convergence. The hyperparameters are the usual ones: M , σ , λ for the RBF mapping \mathbf{F} , and C for the SVM \mathbf{g} , and can be determined by cross-validation.

Our algorithm affords massive parallelization and is suitable for large scale learning. The \mathbf{F} -step (a regression problem) decouples over latent dimensions, the \mathbf{g} -step (one-versus-all SVM) decouples over classes, and the \mathbf{Z} -step decouples over training samples. Indeed, our experiments show linear speedups with multiple processors.

The form of our final classifier is $y = \mathbf{g}(\mathbf{F}(\mathbf{x})) = \mathbf{v}^T \Phi(\mathbf{x}) + b = \sum_{m=1}^M v_m \phi_m(\mathbf{x}) + b$ where $\mathbf{v} = \mathbf{W}^T \mathbf{w} \in \mathbb{R}^M$, and the sign of y gives the label. The number of parameters (weights and centers) is $M(D+L) + L = \mathcal{O}(MD)$ and the runtime for a test point is $\mathcal{O}(MD)$.

3 Experiments

We explore three questions across a range of datasets: the role of dimension reduction on the classification error and the latent space representation; the performance of our classifier, compared to the state-of-the-art; and the training speed of our algorithm.

3.1 The role of dimension reduction

The ideal nonlinear dimensionality reduction + linear classifier Given that the classifier \mathbf{g} is linear, consider the ideal case where \mathbf{F} is infinitely flexible and can represent any desirable mapping from D to L dimensions. Then, a perfect wrapper classifier $\mathbf{g} \circ \mathbf{F}$ can be achieved by having \mathbf{F} map all the inputs having the same label k to a point $\boldsymbol{\mu}_k \in \mathbb{R}^L$, and then locating the K class centroids $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K$ in \mathbb{R}^L such that they are linearly separable and have maximum margin. How this can be achieved depends on the dimension L . With $L = 1$, only $K = 2$ classes may be linearly separable. With $L = 2$, all K classes are linearly separable if placing the centroids on the vertices of a regular K -polygon; however, this leads to a small margin as K grows. In $L > 2$, this generalizes to placing the centroids maximally apart on a hypersphere. However, when $L \geq K - 1$, the margin cannot be further improved, because the K points span a space of $K - 1$ dimensions, specifically a regular simplex, which provides linear separability and maximum margin. Is this ideal actually realized?

We investigate this question experimentally. The dataset in fig. 2 contains two spirals, each has 1000 samples and defines a class. We change the flexibility of \mathbf{F} to see what latent representations are obtained. We try a linear DR and a RBF DR with varying number of basis functions (M centers uniformly sampled from the training set) while keeping the other hyperparameters fixed.

We observe the following from the projections $\mathbf{F}(\mathbf{X})$ shown in fig. 2. (1) Since the dataset is not linearly separable, the two classes overlap severely in the latent space for linear \mathbf{F} (first column). (2) We see from the latent representations (shown in the second row) that, the more flexible \mathbf{F} is, the more classes collapse

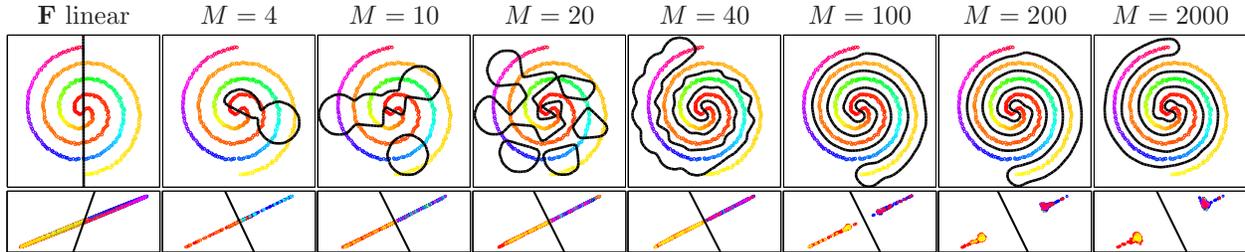


Figure 2: Results on the two spirals dataset. We vary the flexibility of \mathbf{F} to investigate the ideal dimensionality reduction for \mathbf{g} . *Top*: dataset and decision boundary (black curves) achieved with different \mathbf{F} . *Bottom*: representation achieved in the 2D latent space, i.e., the data projections $\mathbf{F}(\mathbf{X})$ and the classification boundary of the SVM \mathbf{g} (black line).

and training samples from different classes are pushed far apart (especially if using $M =$ all 2000 training samples as RBFs centers). Thus, they are easily separated by \mathbf{g} with a big margin (so the nearest neighbor classifier would do very well here). Thus, the overall classifier $\mathbf{g} \circ \mathbf{F}$ is capable of solving linearly non-separable problems given sufficient BFs. (3) To achieve a perfect classification, we need only a few basis functions ($M = 100$ more than suffice here). (4) The projections $\mathbf{F}(\mathbf{x})$ approximately form a line, implying that $L = 1$ is enough to separate two classes.

Relation between the latent dimensionality and the number of classes We now study the role of the latent dimension L , the number of classes K and the geometric configuration of the latent projections. We vary K in the spirals dataset from 2 to 6, with 500 samples in each spiral, and run our algorithm with \mathbf{F} being nonparametric RBFs and $L = 1, \dots, 10$, fixing other hyperparameters at reasonable values. Fig. 3 shows the classification results and projections. We find we do not always obtain a perfect classification using only $L = 2$ dimensions for the one-vs-all SVMs. Instead, we find a common behavior for different K : the classification performance improves drastically as L increases in the beginning, and then stabilizes after some critical L , by which time the training samples are perfectly separated. This is because once the classes are separable in \mathbb{R}^L , they can also be (more easily) separable in higher dimensions with more degrees of freedom to arrange them. We observe in this experiment that, typically with $L = K - 1$ dimensions, the classes all form a point-like cluster and approximately lie on vertices of a regular simplex, with zero classification error (decision boundaries shown in the first column). This gives a recipe to choose L : starting from $L = K - 1$, increase L until the classification error does not improve. It also gives a recipe to initialize \mathbf{Z} , namely to the ideal configuration of K centroids located in the corners of a simplex, so all the \mathbf{z}_n from the same class are set to the centroid of that class. Comparing this experimentally with random initial \mathbf{Z} , we observe that the simplex-based \mathbf{Z} leads to much faster convergence (usually 1 iteration), although interestingly the quality of the minimum found is similar to that of the random initial \mathbf{Z} .

In summary, these results are in approximate agreement with our ideal prediction, although in practice, the extent to which \mathbf{F} collapses classes depends on the number of basis functions. The more BFs, the more flexible \mathbf{F} is and the closer the latent projections are to K centroids in a maximally linearly separable arrangement as described above, and this behavior arises from the joint optimization of DR and classifier. Note that finding an \mathbf{F} that maps a set of points to the same point is trivial: it is constant. But what we seek is an \mathbf{F} that maps each class' points to the class centroid. This is a piecewise constant mapping which is much harder to learn.

3.2 Comparison with other classifiers

We compare with directly applying a nearest neighbor classifier (NN) and linear (LSVM) or Gaussian kernel SVMs (GSVM) on the original inputs, with unsupervised DR methods PCA/Gaussian Kernel PCA (KPCA) followed by nearest neighbor, and with filter approaches such as LDA/KLDA (Gaussian kernel) followed by nearest neighbor. Hyperparameters for these algorithms (kernel width σ for kernel SVM, KPCA, and KLDA, penalty parameter C for SVMs) are chosen by grid search on a separate validation set. In our algorithm, we

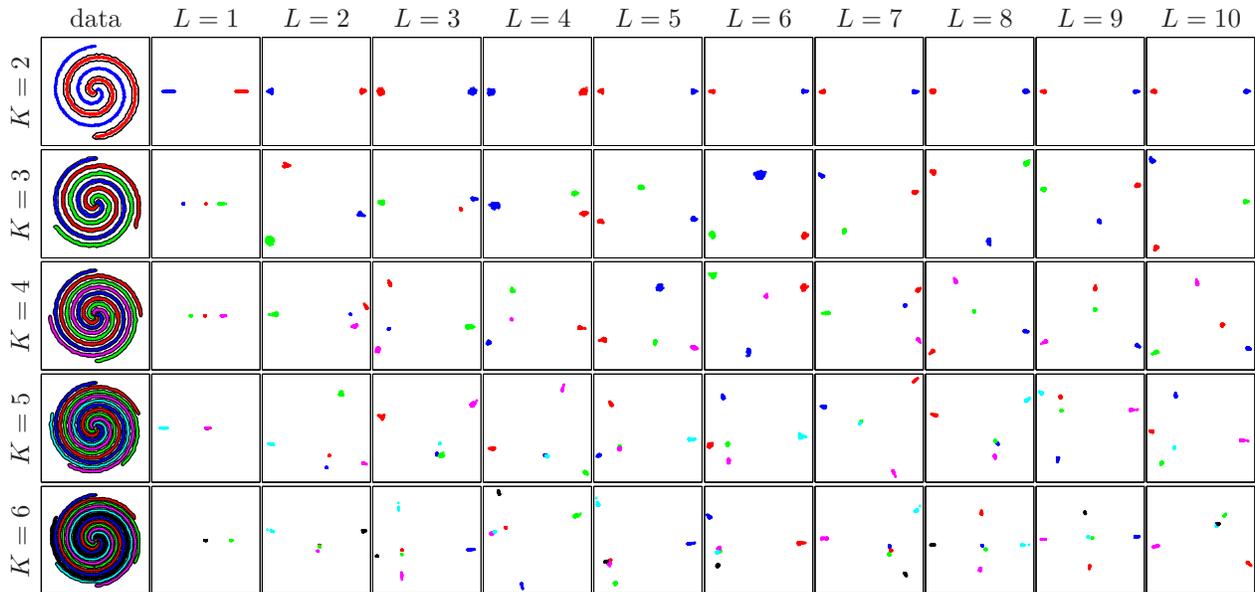


Figure 3: Results on the K -spirals dataset. Each row shows results for a different K . First column: datasets and decision boundaries (black curve, you may need to zoom in) in input space obtained at dimension $L = K - 1$. Columns $L = 1$ to 10: the latent projections $\mathbf{F}(\mathbf{X})$ obtained at dimension L (visualized in 2D using PCA).

initialize \mathbf{Z} randomly.

Document binary classification We pick two classes from the 20 newsgroup dataset (comp.sys.ibm.pc.hardware and comp.sys.mac.hardware), remove words appearing in 5 or fewer documents, reduce the dimension to 2989, and then extract the TFIDF features. We further reduce the input dimension to 1000 with PCA (keeping $> 98\%$ of the variance). For evaluation purposes, we create 10 different 80/20 splits of the 1162 training items into training and validation set. For each split, we let all algorithms pick the optimal hyperparameters based on validation error, and evaluate the optimal models on the test set. Due to the high dimensionality and scarcity of samples, we use linear \mathbf{F} for this problem (we did try RBFs for \mathbf{F} and obtained similar results). We also run Large Margin Nearest Neighbor (LMNN; Weinberger and Saul, 2009), a metric learning algorithm, with the recommended number of target neighbors 3. We report the mean and standard deviation of test error rates over 10 splits for all methods in fig. 4(left).

Our classification performance is quite robust to the choice of L , although in general more dimensions may bring a little improvement in the error rate at higher computational cost. We thus fix the latent dimensions to be 2 for other methods. The results show that using class information in dimensionality reduction is superior to not using it (e.g. PCA), and we outperform others consistently with different L . Fig. 4(right) shows the 2D projections of several algorithms, where supervised DR algorithms manage to separate the classes and PCA does not.

MNIST odd/even classification We perform a binary classification task of discriminating odd digits (1, 3, 5, 7, 9) from the even digits (0, 2, 4, 6, 8) on the MNIST benchmark. We vary the training set size to be 100, 200, 500, 1000, 2000, 3000, 4000, and 5000, including equal number of images randomly sampled for each digit. We use in all cases a balanced validation set of 10000 images from which we pick optimal hyperparameters for all methods. The MNIST test set (10000 samples) is used for evaluating the classification performance. Since this dataset is likely not linearly separable (as can be inferred from the error rate of the linear SVM), we fix the dimension of our latent space to be $L = 2$, and use nonparametric RBFs for \mathbf{F} . We compare our algorithm with nonlinear DR algorithms KPCA ($L = 2$) and KLDA ($L = 1$). We also explore

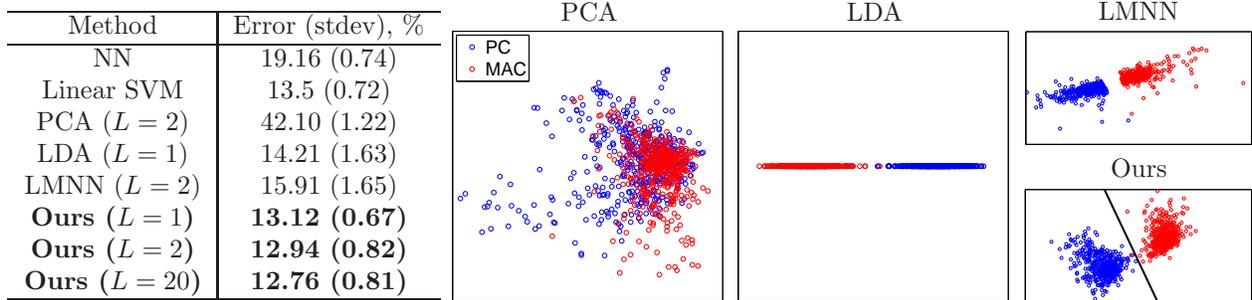


Figure 4: Results on the PC/MAC subset of 20 newsgroups. *Left*: mean test error rate over 10 splits (standard deviation in parenthesis). *Right*: projections by different algorithms.

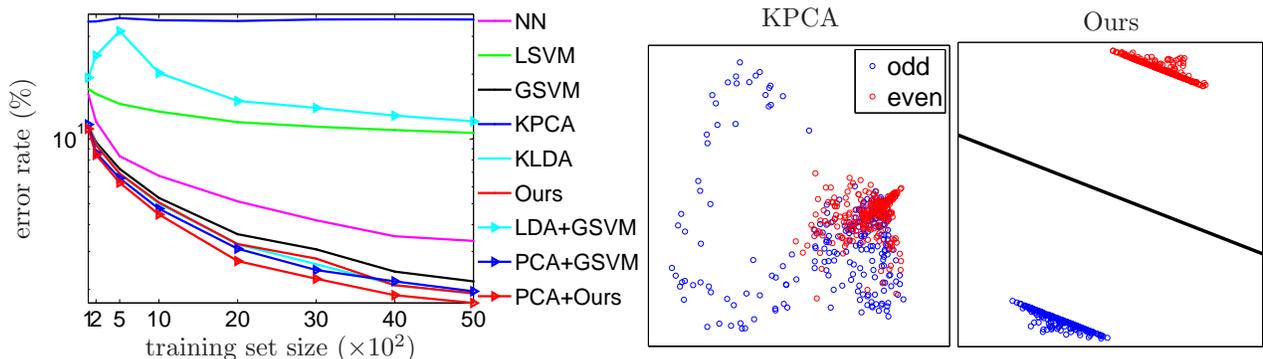


Figure 5: Results on MNIST odd/even classification. *Left*: errors of different algorithms over dataset sizes. *Right*: 2D projections by KPCA and our algorithm.

a two-step approach of using linear DR (PCA/LDA) followed by a Gaussian SVM. We cross-validate L for PCA, while LDA uses $L = 1$.

Figure 5 (top) shows the test errors of all methods over different training set sizes. On the original inputs, our algorithm and KLDA (given very fine grid search for optimal kernel width) perform the best. Their clear advantage over KPCA demonstrates again that class information should be incorporated in DR. We expect our model to have a regularization effect and improve generalization more in smaller training sets. Indeed, we find that we improve over the nearest neighbor classifier (known to achieve near optimal error for large training sets; Duda et al., 2001) most for 100-200 training samples. Fig. 5(bottom) shows the 2D projection of the training set of 500 samples obtained by KPCA and our algorithm. The classes are perfectly separated by our algorithm but not by KPCA, which uses no label information.

Due to the limited power of its DR mapping, LDA+Gaussian SVM performs poorly. PCA+Gaussian SVM performs well (slightly better than our algorithm on original inputs), though only at a much larger L (around 40). We hypothesize this is because PCA is able to remove some noise that is not learned by us from the inputs. We then trained our algorithm on the PCA projection, further reducing the dimension to $L = 2$, and obtained consistently better accuracy (shown in fig. 5 as PCA+Ours).

MNIST 10-classes We now consider the problem of classifying the 10 digit classes of MNIST. We randomly sample 10 000 images for training and 10 000 for validation. The original test set is used for evaluating different algorithms. We were not able to run KPCA and KLDA because the naive implementation of these algorithms requires a huge memory space to store the kernel matrix of $N \times N$ and solve a dense eigenvalue problem. Our algorithm uses 2 500 BFs with centers chosen by K -means.

We searched hyperparameters as follows in order to avoid unnecessary regions of the hyperparameter space. We first do a careful grid search for the kernel width and penalty parameter for the Gaussian SVM. The optimal kernel width ($\sigma = 4.0$) is also used as the RBF kernel width of \mathbf{F} . Then 2 500 centers are

chosen by K-means as RBF basis centers. The regularization parameter λ of \mathbf{F} is fixed to 10^{-3} based on experiments on a much smaller training set (there exists a wide range of λ for which our method works well). Thus we only search for the penalty parameter C of \mathbf{g} , which is common to all 10 SVMs, chosen from $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 10^1, 10^2, 10^3\}$.

We again explored the approach of PCA/LDA+Gaussian SVM as in the other MNIST experiment and obtained a similar result: LDA ($L = 9$)+Gaussian SVM performs poorly; PCA+Gaussian SVM performs well at a relatively larger L . We trained our model on the PCA projection, further reduced the dimension to $L = 10$, and obtained similar performance with much fewer basis functions.

As happened in the case of binary classification, starting from a random initialization, the \mathbf{Z} projections reorganize quickly in the latent space and are well classified after a few iterations.

Fig. 6(top left) shows the test error rates and the total number of support vectors (from the 10 Gaussian SVMs)/basis functions used in each algorithm. Our accuracy is similar to that of the kernel SVM but with 5.5 times fewer basis functions, obtaining a large speedup in testing time. We have again explored the approach of PCA/LDA+Gaussian SVM as in the previous experiment and obtained a similar result: LDA ($L = 9$)+Gaussian SVM performs poorly; PCA+Gaussian SVM performs well at a relatively larger L . We train our model on the PCA projection, further reduce the dimension to $L = 10$, and obtain similar performance with much fewer basis.

Fig. 6(bottom left) shows the performance of our algorithm using different values of the latent dimension L . Our error rates decrease quickly as L increases at first. After $L = 5$, it already does pretty well, and the performance does not change much for $L \geq 10$. We conjectured previously that the optimal latent configuration would be to arrange different classes on the vertices of a regular simplex. The projections achieved by our algorithm in fig. 6(bottom right) agree with that. Since we use PCA to visualize in 2D the latent representations lying in $L = 10$ dimensions, some classes appear to overlap, but they are all completely separated, as can be seen by using other 2D views.

3.3 Training runtime: comparison with alternating optimization without \mathbf{Z}

We compared our three-step alternating optimization scheme with a two-step alternating optimization scheme over only \mathbf{F} and \mathbf{g} , which optimizes the original problem (1) directly, where \mathbf{F} is a RBF network and \mathbf{g} a linear SVM. We are much faster in terms of both progress in objective function and actual runtime. The following experiment shows this in the MNIST odd/even classification problem for the case of 500 training samples. To minimize the nested objective function in eq. (1), the two-step algorithm alternates a \mathbf{g} -step which solves a linear SVM on $\mathbf{F}(\mathbf{X})$ and an \mathbf{F} -step that optimizes over the weights of \mathbf{F} by solving a quadratic program. Since the latent dimensionality is set to $L = 2$, there are 1000 weight parameters in the quadratic program, which we solve by calling the interior point solver of the CVX package (Grant and Boyd, 2012). Fig. 6(middle) shows the nested objective function value versus iteration number during training. The time spent per iteration for our algorithm is about 1/150 of that of alternating minimization. Therefore, the combined runtime of our \mathbf{Z} and \mathbf{F} steps is significantly lower than that of the \mathbf{F} step of alternating minimization, and besides our \mathbf{Z} step has a closed-form solution. Further, a few iterations of our algorithm suffice to find a near-optimal solution (with a small bias that continues to be eliminated as the penalty parameter μ is increased), while the progress of alternating minimization is hopelessly slow.

If instead of a RBF network, which is linear in the parameters of \mathbf{F} , we used a model which is nonlinear in the parameters (such as neural net), the gains of our algorithm would be even larger. The reason is that, in our algorithm, the nonlinearity over \mathbf{F} is confined to the \mathbf{F} -step in the form of a standard least-squares regression problem (section 2.2), thanks to the use of the auxiliary coordinates \mathbf{Z} . Regression problems are well-studied and can be solved with efficient algorithms for many classes of functions \mathbf{F} . In the direct optimization over \mathbf{F} , this nonlinearity is embedded in the constraints of (1), which is harder to deal with.

3.4 Parallel processing

We also ran a simple parallel version of our algorithm in the MNIST 10-classes problem. This solves in parallel for the 10 SVMs in the \mathbf{g} -step, and for the coordinates of all training points in the \mathbf{Z} -step (within each respective step, all these problems are independent). Given that our code is in Matlab, we used the Matlab Parallel Processing Toolbox. The programming effort is insignificant: all we do is replace the “for”

Method	Error (%)	# BFs
Nearest Neighbor	5.34	10 000
Linear SVM	9.20	—
Gaussian SVM	2.93	13 827
LDA (9) + Gaussian SVM	10.67	8 740
PCA (5) + Gaussian SVM	24.31	13 638
PCA (10) + Gaussian SVM	7.44	5 894
PCA (40) + Gaussian SVM	2.58	12 549
Ours (5 , 43)	4.69	2 500
Ours (10, 18)	2.99	2 500
PCA (40) + Ours (10, 17)	2.60	2 500

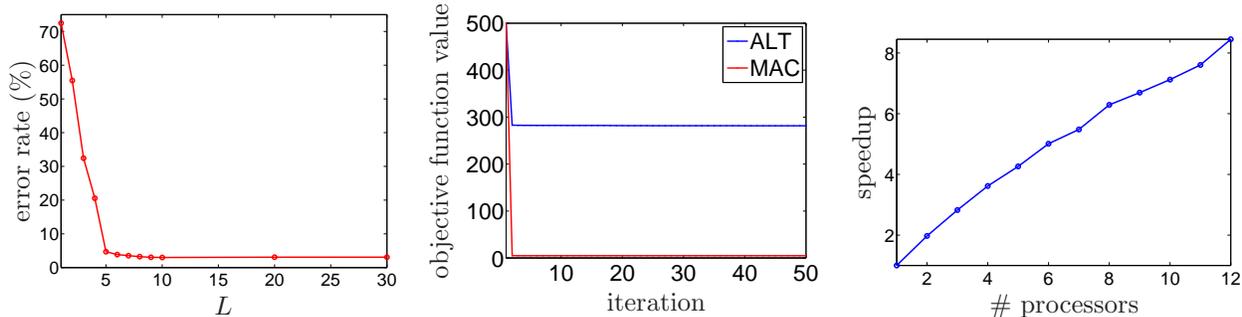
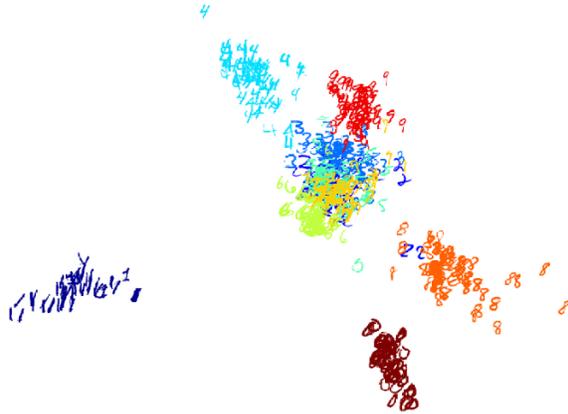


Figure 6: Results of MNIST 10-classes experiment. *Top left*: test error rates (%) and number of basis functions used in each method. In parenthesis, we specify L for LDA/PCA, and L and number of iterations used to reach early stopping for our algorithm. *Top right*: latent representations obtained at dimension $L = 10$ by our algorithm, visualized in 2D with PCA (to avoid cluttering, we plot a subset of 500 images). Although several class digits appear to overlap, this is a visual artifact of the projection to 2D. *Bottom left*: error rate of our algorithm over a range of latent dimensions. *Bottom middle*: comparison of our algorithm with alternating minimization of the nested error. We plot the objective function value vs the iteration number. *Bottom right*: speedups obtained by running our algorithm using the Matlab Parallel Processing Toolbox.

loop over SVMs (in the \mathbf{g} -step) or over data points (in the \mathbf{Z} -step) with a “parfor” loop. Matlab then sends each iteration of the loop to a different processor. We ran this in a shared-memory multiprocessor machine¹, using up to 12 processors (a limit imposed by our Matlab license). We obtain an impressive speedup of up to 8 times, as shown in fig. 6(bottom right). Even larger speedups may be possible if using other parallel computation models, since the Matlab Parallel Processing Toolbox is quite inefficient.

4 Discussion

Filters Our algorithm illuminates the behavior of filter approaches. Such approaches optimize a proxy objective function constructed from (\mathbf{x}_n, y_n) over \mathbf{F} and then learn the classifier \mathbf{g} by optimizing the classification error constructed from $(\mathbf{F}(\mathbf{x}_n), y_n)$ over \mathbf{g} . This is like our \mathbf{F} - and \mathbf{g} -steps, but, firstly, it uses the “wrong” objective for \mathbf{F} , and, secondly, without the coordination through the \mathbf{Z} variables, the process cannot be iterated to converge to a minimum of the joint problem. We can then view our algorithm as a *corrected, iterated filter* approach. Since in practice it converges in a few iterations, its cost is just a little larger, but one need not introduce a proxy objective and yet obtains a true (local) minimum. Thus, we learn the following two lessons: (1) if we want to use a filter \mathbf{F} , the ideal filter would consist of mapping the

¹An Aberdeen Stirling 148 computer having 4 physical CPUs (Intel Xeon CPU L7555@ 1.87GHz), each with 8 individual processing cores (thus a total of 32 actual processors), and a total RAM size of 64 GB.

inputs to class centroids located on the corners of a simplex in latent space; (2) we do not really need a filter approach, because we can train the optimal, wrapper approach nearly as efficiently and simply.

The role of dimensionality reduction in linear classification Being able to find true optima of the classification error allowed us to study the role of nonlinear DR as a preprocessing step for linear classification. With an ideally flexible DR mapping \mathbf{F} , the best possible preprocessing is precisely to remove all variation that is unrelated to the class label, including variation within a manifold—an extreme form of denoising. The input domains are “denoised” so they collapse to nearly zero-dimensional regions. In practice, \mathbf{F} belongs to a certain function class (given by the choice of model and number of parameters), and the ideal where classes collapse is only approached, but is clearly there. Using a latent space of $L = 2$ dimensions is theoretically sufficient but, with a limited \mathbf{F} , using up to $L = K - 1$ helps to improve the separation. Note that collapsing classes requires a genuinely nonlinear DR. The problem formulation of eq. (1) does not explicitly seek to collapse classes, but this behavior emerges anyway from the assumption of low-dimensional representation, if trained jointly with the classifier. Thus, rather than making the classifier work hard to approximate a possibly complex decision boundary, we help it by moving the data around in latent space so the boundary is simpler.

This clashes with the widely held view that a good supervised DR method should produce representations (often visualized in 2D) where the manifold structure of each class is displayed. In fact, with an optimal DR the entire manifold will collapse. This is different from unsupervised DR, where we do want to extract informative features that tell us something about the data variability; and from supervised regression, where only some of the input dimensions should be collapsed (those which do not alter the output).

SVMs and kernel learning Our method and kernel SVMs can be seen as constructing a classifier as an expansion in basis functions $y = \mathbf{g}(\mathbf{F}(\mathbf{x})) = \mathbf{v}^T \Phi(\mathbf{x}) + b$, with M BFs in our case and S support vectors for the SVM. The SVMs do this nonparametrically, at the cost of constructing an $N \times N$ kernel matrix and solving the corresponding problem, which is expensive—although much research work has sought to approximate this (Schölkopf and Smola, 2001) and reduce the number of SVs (Bi et al., 2003). The basis functions $\Phi(\mathbf{x})$ are given by the kernel, which is selected by the user, and the space they implicitly map to is typically infinite-dimensional. The number of SVs S is not set by the user but comes out automatically and can be quite large in practice. Our method is a parametric approach, where the user can set the number of BFs M , and the mapping \mathbf{F} (in this paper, an RBF mapping) maps to a low-dimensional space. The result is a competitive nonlinear classifier, with scalable training and efficient at test time. Having M as a user parameter also allows a simple, direct way for the user to trade off test runtime for accuracy, which is crucial in real-time problems, such as embedded systems, where a typical SVM classifier is too computationally demanding in both memory required to store the SVs and in runtime. As shown in our experiments, we can obtain a classification error comparable to the kernel SVM with $M \ll S$, thus much faster.

It is possible to train a linear SVM by learning \mathbf{v} and b directly given fixed basis functions $\phi(\mathbf{x})$, but this achieves a worse classification error, and does not do DR, as we seek. Our low-dimensional classifier can be seen as a special regularization structure on the classifier’s weights $\mathbf{v} = \mathbf{W}^T \mathbf{w}$, where \mathbf{W} and \mathbf{w} are regularized separately. This effect is more pronounced in the multiclass case since each one-vs-all SVM \mathbf{w}_k interacts with the same DR mapping \mathbf{W} . If using a different functional form for \mathbf{F} (e.g. deep nets), this resemblance with kernel SVMs disappears.

Our model can also be seen as learning a “low-dimensional kernel”, since we pass a pair of latent vectors $(\mathbf{F}(\mathbf{x}), \mathbf{F}(\mathbf{x}'))$ to the linear SVM kernel, rather than applying a kernel $K(\mathbf{x}, \mathbf{x}')$ directly to the high-dimensional inputs. If $\mathbf{F}(\mathbf{x}) = \mathbf{A}\mathbf{x}$ is a linear DR mapping (no need for bias in latent space) then this becomes a form of metric learning with SVMs, using a low-rank metric $\mathbf{A}^T \mathbf{A}$.

5 Related work

Filter approaches typically learn a DR mapping \mathbf{F} using the inputs and label information first, and then fit a classifier \mathbf{g} to the latent projections and labels $(\mathbf{F}(\mathbf{x}_n), y_n)$. They are quite popular due to the ease of optimization, but they rely on the choice of objective function for the filter. Linear discriminant analysis (LDA; Belhumeur et al., 1997) and its kernel version KLDA (Mika et al., 1999) look for a transformation of

the inputs such that, in the latent space, the within-class scatter is minimized while the between-class scatter is maximized. The solution for \mathbf{F} can be obtained by solving an eigenvalue problem. These two algorithms can only produce up to $L = K - 1$ latent dimensions for a K -class problem, due to the singularity of the between-class scatter matrix. Among other variations, Sugiyama (2007) modify LDA to work for manifold data, but the projection mapping is linear, and Urtasun and Darrell (2007) derive a prior distribution from LDA in latent space and use it for a GPLVM to achieve DR, and the latent representation is then fed to a Gaussian process classifier. A clear disadvantage of filter approaches is the heuristic nature of the objective for DR, which acts as a proxy for classification, and is therefore not optimal for the classifier learned afterwards. As we showed, a good filter objective would be to collapse all classes and place them in the corners of a simplex.

Metric learning algorithms (Xing et al., 2003; Goldberger et al., 2005; Globerson and Roweis, 2006; Weinberger and Saul, 2009; Davis et al., 2007) are closely related to DR for classification. Their goal is to find a Mahalanobis metric (or equivalently a linear transform) in input space such that samples in the same class are projected nearby in latent space while samples from different classes are pushed far apart. One achieves DR if the metric is low-rank. However, most metric learning algorithms first solve a positive semidefinite program without rank constraints, and then do a low-rank approximation of the learned Mahalanobis matrix to enforce dimensional reduction, thus the optimality of the projection is no longer guaranteed.

There also exist several wrapper approaches that train the DR mapping \mathbf{F} jointly with a classifier \mathbf{g} in a unified objective function. Pereira and Gordon (2006) (further generalized by Rish et al., 2008) use an objective function that combines the approximation error of the inputs using SVD and the hinge loss from applying a linear SVM to the coordinates, so that the representation extracted this way will be good for classification. This is closely related to supervised dictionary learning (Yang et al., 2012), only that the bias (approximation error) always exists in the model. Also, in this model the latent projections are an implicit function of the inputs, i.e., to project a new input, one needs to solve a optimization problem using learned basis. In contrast, our \mathbf{F} is an explicit mapping, directly applicable to test samples. Ji and Ye (2009) directly minimize the hinge loss of a SVM that operates on linearly transformed inputs (therefore it is a nested error, similar to us). They apply alternating optimization over both mappings. Due to the linearity of \mathbf{F} and \mathbf{g} , they are able to solve for \mathbf{g} in the dual and solve for \mathbf{F} using SVD. This would not be possible in general if \mathbf{F} has a different nonlinear form, unlike in our algorithm. Also, the SVD solution of \mathbf{F} limits the maximum meaningful latent dimension L to the number of classes. In contrast with these approaches, our algorithm is bias free, works with any L , and trains a nonlinear DR mapping fast.

Auxiliary variables were used previously for unsupervised dimensionality reduction (Carreira-Perpiñán and Lu, 2008, 2010) and regression (Wang and Carreira-Perpiñán, 2012). But the unconstrained objective function defined there, while jointly optimized over a dimension reduction mapping \mathbf{F} and a regressor \mathbf{g} , differs from a true wrapper objective function, and results in optima for the combined mapping $\mathbf{g} \circ \mathbf{F}$ that are biased.

6 Conclusion

We have proposed an efficient algorithm to train a nonlinear low-dimensional classifier jointly over the nonlinear DR mapping and the classifier (a wrapper approach). The algorithm is easy to implement, reuses existing regression and SVM procedures, and parallelizes well. The resulting classifier achieves state-of-the-art classification error with a small number of basis functions, which can be tuned by the user. The algorithm can be seen as an iterated filter approach with provable convergence to a local minimum of the joint objective. This justifies filter approaches that use a secondary criterion over the DR mapping such as class separability or intra-class scatter in an effort to construct a good classifier, but also obviates them, since one can ensure to get the best low-dimensional classifier (under the model assumptions) with just a little more computation.

Our experiments illuminate the role of nonlinear DR in linear classification. If we optimize the classification error—the figure of merit one really cares about—*jointly* over the projection mapping and the classifier, the best DR in fact erases all structure (manifold and otherwise) in the input other than class membership, and uses the latent space to place collapsed classes in such a way that they are maximally linearly separable. Future work should analyze the role of DR with nonlinear classifiers.

Our algorithm generalizes beyond the specific forms of DR mapping and classifier used here, and we are exploring other combinations. In particular, one can replace the DR mapping with a complex feature-extraction mapping that can handle invariances, such as convolutional neural nets, and jointly optimize this and the classifier.

Acknowledgments

Work funded in part by NSF CAREER award IIS-0754089.

References

- P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7):711–720, July 1997.
- J. Bi, K. Bennett, M. Embrechts, C. Breneman, and M. Song. Dimensionality reduction via sparse support vector machines. *J. Machine Learning Research*, 3:1229–1243, Mar. 2003.
- C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer Series in Information Science and Statistics. Springer-Verlag, Berlin, 2006.
- M. Á. Carreira-Perpiñán and Z. Lu. Dimensionality reduction by unsupervised regression. In *Proc. of the 2008 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'08)*, Anchorage, AK, June 23–28 2008.
- M. Á. Carreira-Perpiñán and Z. Lu. Parametric dimensionality reduction by unsupervised regression. In *Proc. of the 2010 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'10)*, pages 1895–1902, San Francisco, CA, June 13–18 2010.
- M. Á. Carreira-Perpiñán and W. Wang. Distributed optimization of deeply nested systems. Unpublished manuscript, arXiv:1212.5921, Dec. 24 2012.
- M. Á. Carreira-Perpiñán and W. Wang. Distributed optimization of deeply nested systems. In S. Kaski and J. Corander, editors, *Proc. of the 17th Int. Workshop on Artificial Intelligence and Statistics (AISTATS 2014)*, pages 10–19, Reykjavik, Iceland, Apr. 22–25 2014.
- C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Trans. Intelligent Systems and Technology*, 2(3):27, Apr. 2011.
- J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In Z. Ghahramani, editor, *Proc. of the 24th Int. Conf. Machine Learning (ICML'07)*, Corvallis, Oregon, June 20–24 2007.
- R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, New York, London, Sydney, second edition, 2001.
- A. Globerson and S. Roweis. Metric learning by collapsing classes. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 18, pages 451–458. MIT Press, Cambridge, MA, 2006.
- J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 17, pages 513–520. MIT Press, Cambridge, MA, 2005.
- M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.0 beta. <http://cvxr.com/cvx>, Sept. 2012.
- I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. Machine Learning Research*, 3:1157–1182, Mar. 2003.

- S. Ji and J. Ye. Linear dimensionality reduction for multi-label classification. In *Proc. of the 21st Int. Joint Conf. Artificial Intelligence (IJCAI'09)*, pages 1077–1082, Pasadena, California, July 11–17 2009.
- R. Kohavi and G. H. John. The wrapper approach. In H. Liu and H. Motoda, editors, *Feature Extraction, Construction and Selection. A Data Mining Perspective*. Springer-Verlag, 1998.
- S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In Y. H. Hu and J. Larsen, editors, *Proc. of the 1999 IEEE Signal Processing Society Workshop on Neural Networks for Signal Processing (NNSP'99)*, pages 41–48, Madison, WI, Aug. 23–25 1999.
- J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer-Verlag, New York, second edition, 2006.
- F. Pereira and G. Gordon. The support vector decomposition machine. In W. W. Cohen and A. Moore, editors, *Proc. of the 23rd Int. Conf. Machine Learning (ICML'06)*, pages 689–696, Pittsburgh, PA, June 25–29 2006.
- R. Rifkin and A. Klautau. In defense of one-vs-all classification. *J. Machine Learning Research*, 5:101–141, Jan. 2004.
- I. Rish, G. Grabarnilk, G. Cecchi, F. Pereira, and G. Gordon. Closed-form supervised dimensionality reduction with generalized linear models. In A. McCallum and S. Roweis, editors, *Proc. of the 25th Int. Conf. Machine Learning (ICML'08)*, pages 832–839, Helsinki, Finland, July 5–9 2008.
- B. Schölkopf and A. J. Smola. *Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning Series. MIT Press, Cambridge, MA, 2001.
- M. Sugiyama. Dimensionality reduction of multimodal labeled data by local Fisher discriminant analysis. *J. Machine Learning Research*, 8:1027–1061, May 2007.
- R. Urtasun and T. Darrell. Discriminative Gaussian process latent variable model for classification. In Z. Ghahramani, editor, *Proc. of the 24th Int. Conf. Machine Learning (ICML'07)*, pages 927–934, Corvallis, Oregon, June 20–24 2007.
- W. Wang and M. Á. Carreira-Perpiñán. Nonlinear low-dimensional regression using auxiliary coordinates. In N. Lawrence and M. Girolami, editors, *Proc. of the 15th Int. Workshop on Artificial Intelligence and Statistics (AISTATS 2012)*, pages 1295–1304, La Palma, Canary Islands, Spain, Apr. 21–23 2012.
- W. Wang and M. Á. Carreira-Perpiñán. The role of dimensionality reduction in classification. In C. E. Brodley and P. Stone, editors, *Proc. of the 28th National Conference on Artificial Intelligence (AAAI 2014)*, Quebec City, Canada, July 27–31 2014.
- K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *J. Machine Learning Research*, 10:207–244, Feb. 2009.
- E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 15, pages 521–528. MIT Press, Cambridge, MA, 2003.
- J. Yang, Z. Wang, Z. Lin, X. Shu, and T. Huang. Bilevel sparse coding for coupled feature spaces. In *Proc. of the 2012 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'12)*, pages 2360–2367, Providence, RI, June 16–21 2012.