The Role of Dimensionality Reduction in Classification

Weiran Wang and **Miguel Á. Carreira-Perpiñán**
Electrical Engineering and Computer Science, School of Engineering, University of California, Merced

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.

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 (Belhumeur, Hespanha, and Kriegman 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, our contributions are as follows. (1) We propose a simple, efficient, scalable and generic way of jointly optimizing the classifier's loss over $(\mathbf{F}, \mathbf{g})$, which we apply 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. (3) The resulting nonlinear low-dimensional SVM classifier achieves state-of-the-art performance while being fast at test time.

Joint optimization of mapping and classifier using auxiliary coordinates

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:

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

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 dependence $\mathbf{g}(\mathbf{F}(\cdot))$ 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 2014) to be equivalent to (1):

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

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 $\mu \to \infty$:

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

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). 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 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.

With K classes, we use one-vs-all scheme (Rifkin and Klautau 2004) and train K binary SVMs, one per class. The final label is given by the SVM with largest decision value. The objective in (1) is replaced by the sum of the K SVMs' objective functions (Wang and Carreira-Perpiñán 2014).

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

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. 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, $\mathbf{F}(\mathbf{x}) = \mathbf{W}\boldsymbol{\Phi}(\mathbf{x})$, with $M \ll N$ Gaussian RBFs $\phi_m(\mathbf{x}) = \exp(-\frac{1}{2}\|(\mathbf{x}-\mathbf{c}_m)/\sigma\|^2)$, 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 $\mathbf{W}$.

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

$$\min_{\mathbf{z},\xi} \ \left\|\mathbf{z} - \mathbf{F}(\mathbf{x})\right\|^2 + 2C/\mu\xi \tag{4}$$
$$\text{s.t. } y(\mathbf{w}^T\mathbf{z}+b) \geq 1-\xi, \quad \xi \geq 0$$

where we have also included the slack variable, since we find this speeds up the alternating optimization. This is a convex quadratic program with closed-form solution $\mathbf{z} = \mathbf{F}(\mathbf{x}) + \gamma y\mathbf{w}$, where γ is a scalar which takes one of three possible values (Wang and Carreira-Perpiñán 2014), and costs $\mathcal{O}(L)$.

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 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. 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 $\mu \to \infty$ 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.

The form of our final classifier is $y = \mathbf{g}(\mathbf{F}(\mathbf{x})) = \mathbf{v}^T\mathbf{\Phi}(\mathbf{x}) + b = \sum_{m=1}^{M} v_m\phi_m(\mathbf{x}) + b$ 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 $\mathcal{O}(MD)$.

Experiments

We explore three questions across a range of datasets: the role of dimension reduction; the performance of our classifier, compared to the state-of-the-art; and the training speed of our algorithm. 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. We initialize our algorithm randomly.

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 classifier $\mathbf{g} \circ \mathbf{F}$ can be achieved by having $\mathbf{F}$ map all 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 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. 1 contains two spirals, each has 1 000 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. 1: (1) since the dataset is not linearly separable, the two classes overlap severely in the latent space for linear $\mathbf{F}$ (first column); (2) the more flexible $\mathbf{F}$ is, the more classes collapse and training samples from different classes are pushed far apart (especially if using M = all 2000 training samples as RBFs centers), so as to be easily separated by $\mathbf{g}$ with big margin, and our classifier $\mathbf{g} \circ \mathbf{F}$ is capable of solving linearly non-separable problems given sufficient BFs; (3) to achieve a perfect classification, only a few basis functions are needed ($M = 100$ is more than sufficient here); (4) $\mathbf{F}(\mathbf{x})$ approximately form a line, implying that $L = 1$ is enough to separate two classes.

We then 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. 2 shows the classification results and projections $\mathbf{F}(\mathbf{X})$. We find we can not always obtain 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 also 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: increase L until classification error does not improve, and $K - 1$ seems to be a good starting point.

In summary, these results are in approximate agreement with our ideal prediction, although the extent to which $\mathbf{F}$ collapses classes depends on the number of basis functions M in practice. The larger M is, the more flexible $\mathbf{F}$ is and the closer the latent projections are to K centroids in a maximally linearly separable arrangement, and this behavior arises from the joint optimization of DR and classifier.

Document binary classification We perform binary classification on the two topics comp.sys.ibm.pc.hardware and comp.sys.mac.hardware from the 20 newsgroup dataset using the TFIDF features. For evaluation, we create 10 different 80/20 splits of the 1 162 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 (using RBFs achieves similar results). The mean and standard deviation of test error rates over 10 splits for all methods are shown in fig. 3 (top).

Our classification performance is quite robust to the choice of L, although in general larger L may bring a little improvement in the error rate at higher computational cost. We fix the latent dimensions to be 2 for other methods. The results show that using class information in dimensionality

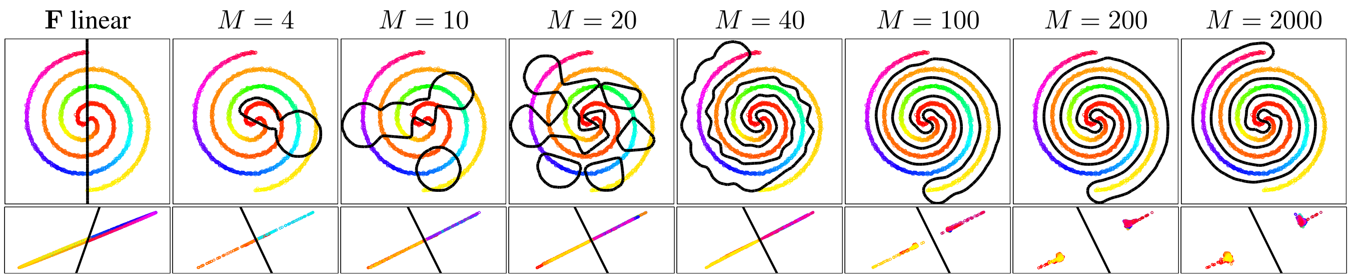

Figure 1: Experimental results on the two spirals dataset. We vary the flexibility of $\mathbf{F}$ to investigate the ideal dimensionality reduction for $\mathbf{g}$. *Top*: dataset and decision boundary (black curves) achieved with different $\mathbf{F}$. *Bottom*: 2D projections $\mathbf{F}(\mathbf{X})$ and the SVM $\mathbf{g}$ boundary (black line).

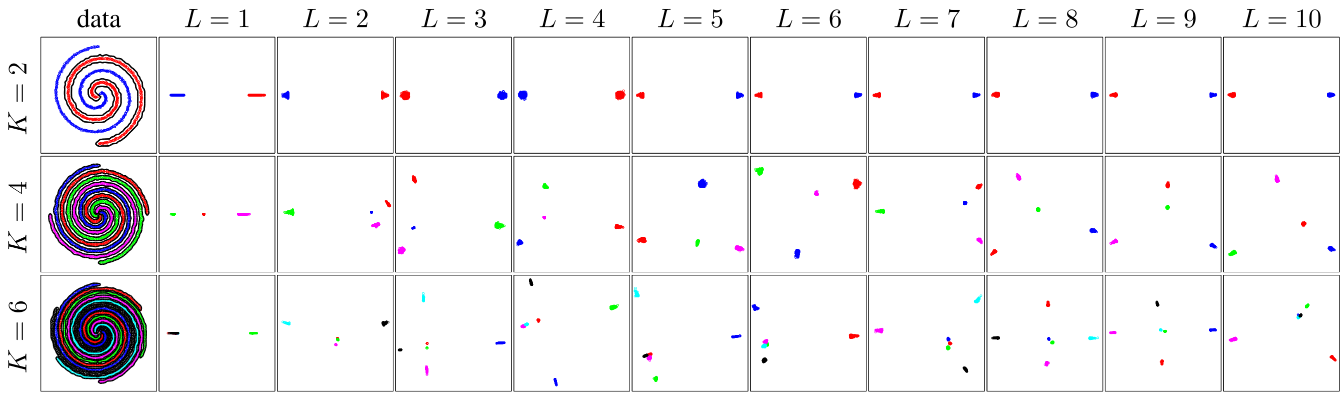

Figure 2: Experimental results on the K-spirals dataset. Each row shows results for a different K. First column: datasets and decision boundaries (black curve) obtained at $L = K - 1$. We apply PCA to visualize the latent representations in 2D.

Methods	Error (%)
NN	19.16 (0.74)
Linear SVM	13.5 (0.72)
PCA ($L = 2$)	42.10 (1.22)
LDA ($L = 1$)	14.21 (1.63)
Ours ($L = 1$)	**13.12 (0.67)**
Ours ($L = 2$)	**12.94 (0.82)**
Ours ($L = 20$)	**12.76 (0.81)**

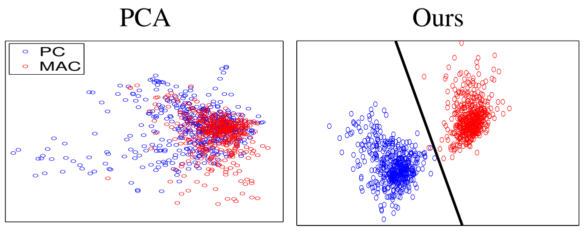

Figure 3: Results on the PC/MAC subset of 20 newsgroups. *Top*: mean mean test error rate (with Std in parenthesis) over 10 splits. *Bottom*: projections by different algorithms.

reduction is superior to not using it (e.g. PCA), and we outperform others consistently with different L. Fig. 3 (bottom) shows the 2D projections, where our algorithm manages to separate the classes and PCA can not.

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 are 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. Hyperparameters are searched in a way that avoids unnecessary regions of the space (Wang and Carreira-Perpiñán 2014).

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 4: Test error rates (%) and number of basis functions used in each method on MNIST 10-classes problem. We specify L for LDA/PCA, L and number of iterations used to reach early stopping for our algorithm in parenthesis.

Fig. 4 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,

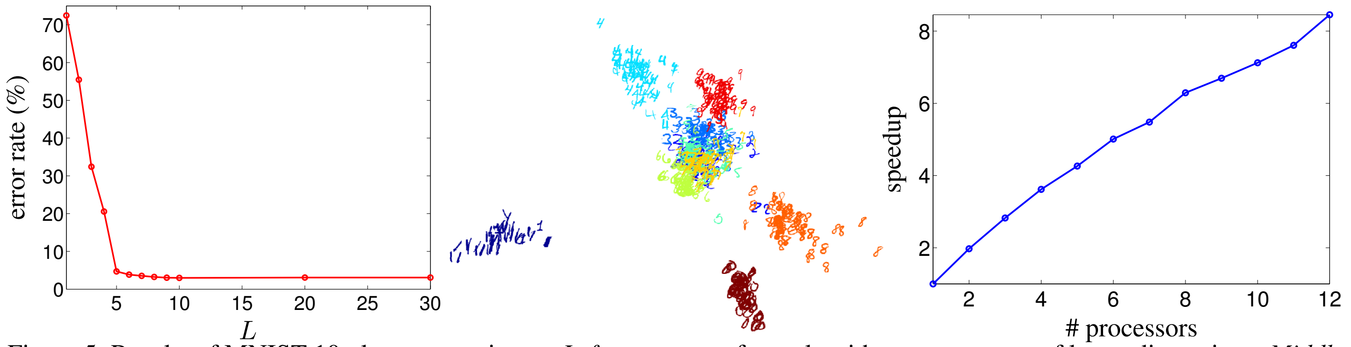

Figure 5: Results of MNIST 10-classes experiment. *Left*: error rate of our algorithm over a range of latent dimensions. *Middle*: 2D embedding of our algorithm, we apply PCA to the latent representations obtained at $L = 10$ (to avoid cluttering, we plot a subset of 500 images). *Right*: speedups obtained by running our algorithm using the Matlab Parallel Processing Toolbox.

obtaining a large speedup in testing time. We have explored the approach of PCA/LDA+Gaussian SVM 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. 5(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. This is supported in this experiment by the projections achieved by our algorithm in fig. 5(middle). PCA is used to visualize the latent representations at $L = 10$, though more powerful algorithms like t-SNE (van der Maaten and Hinton 2008) completely separate all classes in a 2D embedding.

Training runtime and parallel processing We compare our three-step alternating optimization scheme with a two-step alternating optimization scheme over only $\mathbf{F}$ and $\mathbf{g}$ which optimizes (1) directly. We are much faster in terms of both progress in objective function and actual runtime (Wang and Carreira-Perpiñán 2014). We also ran a simple parallel version of our algorithm in the MNIST 10-classes problem, which solves the 10 SVMs in the $\mathbf{g}$-step and the decoupled $\mathbf{Z}$-step for all points in parallel. Using up to 12 processors with the Matlab Parallel Processing Toolbox, we obtain a near-linear speedup of up to 8 times (fig. 5 right).

Discussion

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 (1) it uses the "wrong" objective for $\mathbf{F}$, and (2) 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 needs not introduce a proxy objective and yet obtains a true minimum.

Our method and kernel SVMs can be seen as constructing a classifier as an expansion in basis functions $y = \mathbf{g}(\mathbf{F}(\mathbf{x})) = \mathbf{v}^T\boldsymbol{\Phi}(\mathbf{x})+b$, 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 approximating this (Schölkopf and Smola 2001) and reducing the number of SVs (Bi et al. 2003)). The basis functions $\boldsymbol{\Phi}(\mathbf{x})$ 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 the experiments, we can obtain a classification error comparable to 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 $\boldsymbol{\phi}(\mathbf{x})$, but this achieves a worse classification error, and does not do DR. 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 dimensionality reduction mapping $\mathbf{W}$. If using a different functional form for $\mathbf{F}$ (e.g. deep nets), this resemblance with kernel SVMs disappears.

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.

Being able to find true optima of the classification error allowed us to study the role of nonlinear DR as a preprocessing step for 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. Note that collapsing classes requires a genuinely nonlinear DR. The problem formulation (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. 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).

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 $(\mathbf{F}(\mathbf{x}_n), y_n)$. Linear discriminant analysis (LDA) (Belhumeur, Hespanha, and Kriegman 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. LDA has also been modified to work for manifold data (Sugiyama 2007), where the projection mapping is still linear. The 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.

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, whose 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 metric to enforce DR, thus 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) propose 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 extracted representation is good for classification (see also generalization in Rish et al. (2008)). 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. In Ji and Ye (2009), one directly minimizes the hinge loss of a SVM that operates on linearly transformed inputs (therefore it is a nested error, similar to us). The authors 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. Also, the SVD solution of $\mathbf{F}$ limits the maximum meaningful latent dimension L to the number of classes. On the contrary, 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 2010) and regression (Wang and Carreira-Perpiñán 2012). But the objective 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.

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 joint minimum. 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 get the best low-dimensional classifier (under the model assumptions) with just a little more computation.

Our experiments illuminate the role of DR in classification. If we optimize the classification error—the figure of merit one really cares about—*jointly* over the DR 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 so 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 dimension reduction mapping with a complex feature-extraction mapping that can handle invariances, such as convolutional neural nets, and jointly optimize this and the classifier.

References

Belhumeur, P. N.; Hespanha, J. P.; and Kriegman, D. J. 1997. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Trans. Pattern Analysis and Machine Intelligence* 19(7):711–720.

Bi, J.; Bennett, K.; Embrechts, M.; Breneman, C.; and Song, M. 2003. Dimensionality reduction via sparse support vector machines. *J. Machine Learning Research* 3:1229–1243.

Bishop, C. M. 2006. *Pattern Recognition and Machine Learning*. Springer Series in Information Science and Statistics. Berlin: Springer-Verlag.

Carreira-Perpiñán, M. Á., and Lu, Z. 2010. Parametric dimensionality reduction by unsupervised regression. In *Proc. of the 2010 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'10)*, 1895–1902.

Carreira-Perpiñán, M. Á., and Wang, W. 2012. Distributed optimization of deeply nested systems. Unpublished manuscript, arXiv:1212.5921.

Carreira-Perpiñán, M. Á., and Wang, W. 2014. Distributed optimization of deeply nested systems. In Kaski, S., and Corander, J., eds., *Proc. of the 17th Int. Workshop on Artificial Intelligence and Statistics (AISTATS 2014)*, 10–19.

Chang, C.-C., and Lin, C.-J. 2011. LIBSVM: A library for support vector machines. *ACM Trans. Intelligent Systems and Technology* 2(3):27.

Davis, J. V.; Kulis, B.; Jain, P.; Sra, S.; and Dhillon, I. S. 2007. Information-theoretic metric learning. In Ghahramani, Z., ed., *Proc. of the 24th Int. Conf. Machine Learning (ICML'07)*.

Globerson, A., and Roweis, S. 2006. Metric learning by collapsing classes. In Weiss, Y.; Schölkopf, B.; and Platt, J., eds., *Advances in Neural Information Processing Systems (NIPS)*, volume 18, 451–458. MIT Press, Cambridge, MA.

Goldberger, J.; Roweis, S.; Hinton, G.; and Salakhutdinov, R. 2005. Neighbourhood components analysis. In Saul, L. K.; Weiss, Y.; and Bottou, L., eds., *Advances in Neural Information Processing Systems (NIPS)*, volume 17, 513–520. MIT Press, Cambridge, MA.

Guyon, I., and Elisseeff, A. 2003. An introduction to variable and feature selection. *J. Machine Learning Research* 3:1157–1182.

Ji, S., and Ye, J. 2009. Linear dimensionality reduction for multi-label classification. In *Proc. of the 21st Int. Joint Conf. Artificial Intelligence (IJCAI'09)*, 1077–1082.

Kohavi, R., and John, G. H. 1998. The wrapper approach. In Liu, H., and Motoda, H., eds., *Feature Extraction, Construction and Selection. A Data Mining Perspective*. Springer-Verlag.

Mika, S.; Rätsch, G.; Weston, J.; Schölkopf, B.; and Müller, K.-R. 1999. Fisher discriminant analysis with kernels. In Hu, Y. H., and Larsen, J., eds., *Proc. of the 1999 IEEE Signal Processing Society Workshop on Neural Networks for Signal Processing (NNSP'99)*, 41–48.

Nocedal, J., and Wright, S. J. 2006. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. New York: Springer-Verlag, second edition.

Pereira, F., and Gordon, G. 2006. The support vector decomposition machine. In Cohen, W. W., and Moore, A., eds., *Proc. of the 23rd Int. Conf. Machine Learning (ICML'06)*, 689–696.

Rifkin, R., and Klautau, A. 2004. In defense of one-vs-all classification. *J. Machine Learning Research* 5:101–141.

Rish, I.; Grabarnilk, G.; Cecchi, G.; Pereira, F.; and Gordon, G. 2008. Closed-form supervised dimensionality reduction with generalized linear models. In McCallum, A., and Roweis, S., eds., *Proc. of the 25th Int. Conf. Machine Learning (ICML'08)*, 832–839.

Schölkopf, B., and Smola, A. J. 2001. *Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning Series. Cambridge, MA: MIT Press.

Sugiyama, M. 2007. Dimensionality reduction of multimodal labeled data by local Fisher discriminant analysis. *J. Machine Learning Research* 8:1027–1061.

van der Maaten, L. J. P., and Hinton, G. E. 2008. Visualizing data using t-SNE. *J. Machine Learning Research* 9:2579–2605.

Wang, W., and Carreira-Perpiñán, M. Á. 2012. Nonlinear low-dimensional regression using auxiliary coordinates. In Lawrence, N., and Girolami, M., eds., *Proc. of the 15th Int. Workshop on Artificial Intelligence and Statistics (AISTATS 2012)*, 1295–1304.

Wang, W., and Carreira-Perpiñán, M. Á. 2014. The role of dimensionality reduction in classification. Unpublished manuscript, arXiv:1405.6444.

Weinberger, K. Q., and Saul, L. K. 2009. Distance metric learning for large margin nearest neighbor classification. *J. Machine Learning Research* 10:207–244.

Xing, E.; Ng, A.; Jordan, M.; and Russell, S. 2003. Distance metric learning with application to clustering with side-information. In Becker, S.; Thrun, S.; and Obermayer, K., eds., *Advances in Neural Information Processing Systems (NIPS)*, volume 15, 521–528. MIT Press, Cambridge, MA.

Yang, J.; Wang, Z.; Lin, Z.; Shu, X.; and Huang, T. 2012. Bilevel sparse coding for coupled feature spaces. In *Proc. of the 2012 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'12)*, 2360–2367.