The role of dimensionality reduction in classification

Weiran Wang and **Miguel Á. Carreira-Perpiñán**

Electrical Engineering and Computer Science

University of California, Merced

http://eecs.ucmerced.edu

Low-dimensional classifiers

We consider constructing a nonlinear classifier by first reducing dimension nonlinearly and then classifying linearly.

Three questions we ask in this paper:

1. How to train a nonlinear classifier of the form

$$y = g(\mathbf{F}(\mathbf{x})) \begin{cases} g\text{:} & \text{linear SVM, } g(\mathbf{z}) = \mathbf{w}^T\mathbf{z} + b \\ \mathbf{F}\text{:} & \text{nonlinear dimensionality reduction} \\ & \qquad \text{RBF, deep net, GP}\ldots \end{cases}$$

   optimizing jointly over g and $\mathbf{F}$?

2. What is the role of the nonlinear dimensionality reduction $\mathbf{F}$ in the resulting classifier?

3. How good and fast is the resulting nonlinear, low-dimensional classifier?

❶ Learning low-dim features for a linear SVM

Consider binary classification throughout (see the multiclass case in the paper).

Objective function over the parameters of g and $\mathbf{F}$ given a dataset $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$ of (inputs,labels):

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

- If $\mathbf{F} =$ identity then $y = g(\mathbf{F}(\mathbf{x})) = \mathbf{w}^T\mathbf{x} + b$ is a linear SVM and the problem is a convex quadratic program, easy to solve.
- If $\mathbf{F}$ is nonlinear then $y = g(\mathbf{F}(\mathbf{x})) = \mathbf{w}^T\mathbf{F}(\mathbf{x}) + b$ is a nonlinear classifier and the problem is nonconvex, difficult to solve.

❶ Learning low-dim features for a linear SVM (cont.)

- Simple but suboptimal approach often used in practice ("filter" approach):
  1. First reduce dimension of the inputs ignoring the classifier. This requires optimizing a proxy objective function over $\mathbf{F}$ only.

     unsupervised, e.g. PCA; supervised, e.g. LDA
  2. Then fix $\mathbf{F}$ and train g: fit a linear SVM with inputs $\{\mathbf{F}(\mathbf{x}_n)\}$ and labels $\{y_n\}$.

  The features are not optimal for the linear SVM classifier.

  They are still helpful to remove out-of-manifold noise and speed up the classifier.
- We want a "wrapper" approach, i.e., optimize the overall classification error jointly over the features $\mathbf{F}$ and classifier g.
- We apply the method of auxiliary coordinates (MAC) for nested systems

  Carreira-Perpinan & Wang, AISTATS, 2014

❶ Solution with auxiliary coordinates (MAC)

The problem is "nested" because of $g(\mathbf{F}(\cdot))$ in the constraints:

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

Break the nesting by introducing an auxiliary coordinate $\mathbf{z}_n$ per point:

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

The auxiliary coordinates correspond to the low-dim projections but are separate parameters.

❶ Solution with auxiliary coordinates (MAC) (cont.)

Solve this constrained problem with a quadratic-penalty method: optimize for fixed penalty parameter $\mu > 0$ and drive $\mu \to \infty$:

$$\min_{\mathbf{F},\mathbf{g},\boldsymbol{\xi},\mathbf{Z}} \quad \lambda R(\mathbf{F}) + \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n + \frac{\mu}{2}\sum_{n=1}^{N}\|\mathbf{z}_n - \mathbf{F}(\mathbf{x}_n)\|^2$$

$$\text{s.t.} \quad y_n(\mathbf{w}^T\mathbf{z}_n + b) \geq 1 - \xi_n,\ \xi_n \geq 0,\ n = 1,\dots,N.$$

Alternating optimization of the penalty function for fixed μ:

- g step: fit a linear SVM to $\{(\mathbf{z}_n, y_n)\}_{n=1}^N$
  A classifier using as inputs the auxiliary coordinates.
- $\mathbf{F}$ step: fit a nonlinear mapping $\mathbf{F}$ to $\{(\mathbf{x}_n, \mathbf{z}_n)\}_{n=1}^N$
  A regressor using as outputs the auxiliary coordinates.

} in parallel

- $\mathbf{Z}$ step: set $\mathbf{z}_n = \mathbf{F}(\mathbf{x}_n) + \gamma_n y_n \mathbf{w}$, $n = 1,\dots,N$
  A closed-form update for each point's auxiliary coordinate $\mathbf{z}_n$.

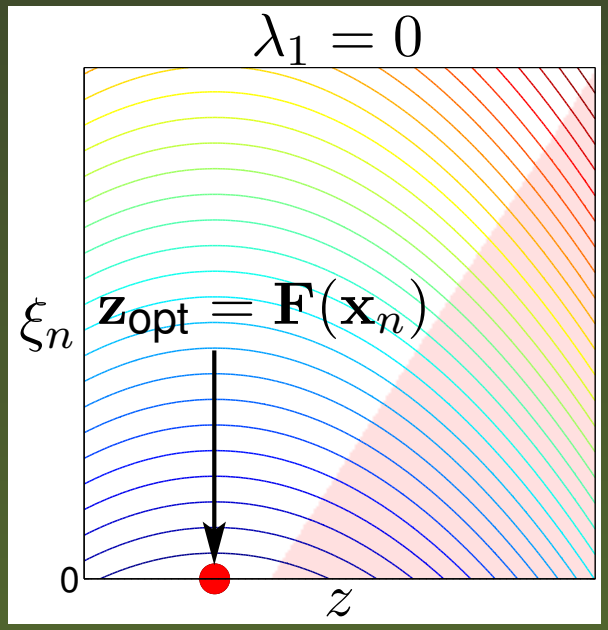
} in parallel

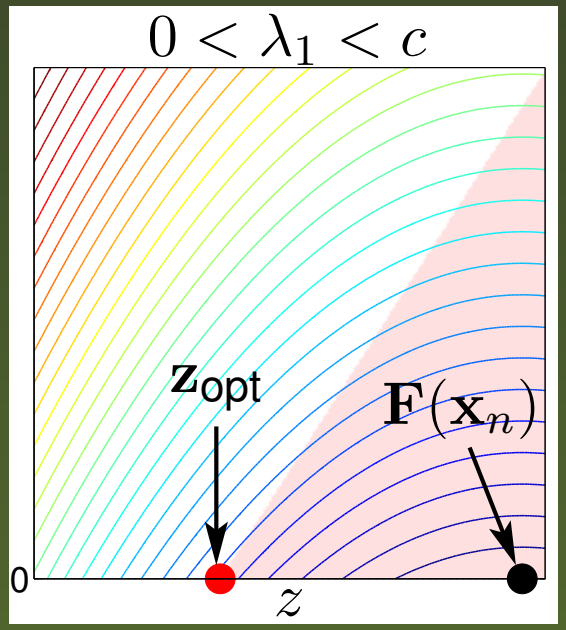
❶ MAC algorithm: solution of the Z step

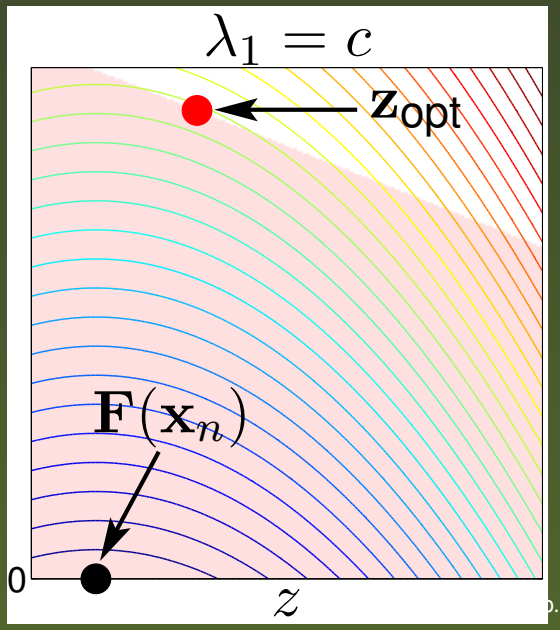
The Z step decouples into N independent problems, each a quadratic program on L parameters ($L =$ dimension of latent space):

$$\min_{\mathbf{z}_n, \xi_n} \quad \|\mathbf{z}_n - \mathbf{F}(\mathbf{x}_n)\|^2 + \frac{2C}{\mu}\xi_n$$
$$\text{s.t.} \quad y_n(\mathbf{w}^T\mathbf{z}_n + b) \geq 1 - \xi_n,\ \xi_n \geq 0, \qquad \mathbf{z}_n \in \mathbb{R}^L$$

with closed-form solution $\mathbf{z}_n = \mathbf{F}(\mathbf{x}_n) + \gamma_n y_n \mathbf{w}$, where γ_n takes one of three possible values. Cost: $\mathcal{O}(L)$.







❶ MAC algorithm: advantages

In summary, all the algorithm does is repeatedly fit a regressor and a linear SVM, and update each point's coordinates. It is like iterating a "filter" but correcting the coordinates so we converge to a true optimum.

- No complicated gradients obtained from the chain rule.
- Instead, it reuses algorithms:
  - classifier g: linear SVM
  - regressor $\mathbf{F}$: RBF, deep net, GP...
- Parallel:
  - classifier, regressor trained in parallel

    multiclass case: K SVMs trained in parallel (one per class)
  - N coordinates updated in parallel.
- Converges to a local optimum as $\mu \to \infty$.
- Fast iterations if using standard optimization techniques

  warm starts, caching matrix factorizations, inexact steps, etc.
- Large progress in first few iterations. ☞ ☞

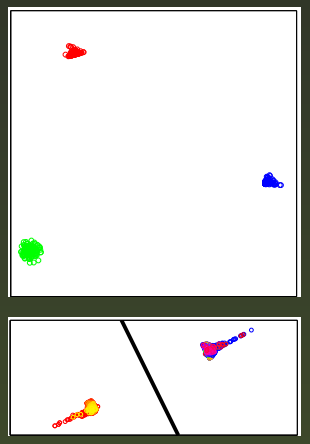
❷ Role of nonlinear dimension reduction in the linSVM

The linear SVM g is less flexible than the nonlinear mapping $\mathbf{F}$, so, intuitively, $\mathbf{F}$ will do most of the work:

- Ideal case: collapse classes onto maximally linearly separable centroids

  corners of a simplex, if using $L \geq K - 1$ dimensions

- In practice: the amount of collapse depends on how flexible $\mathbf{F}$ is (number of parameters, regularization).



This is very different from what PCA/LDA dimension reduction does. The optimal dimension reduction for linear classification destroys all manifold structure. It only cares about facilitating linear separability.

❷ Role of dimension reduction: experiments

We reduce dimension with radial basis function networks (RBFs):

$$\mathbf{F}(\mathbf{x}) = \sum_{m=1}^{M} \boldsymbol{\alpha}_m \phi_m(\mathbf{x})$$

each BF ϕ_m is a Gaussian.

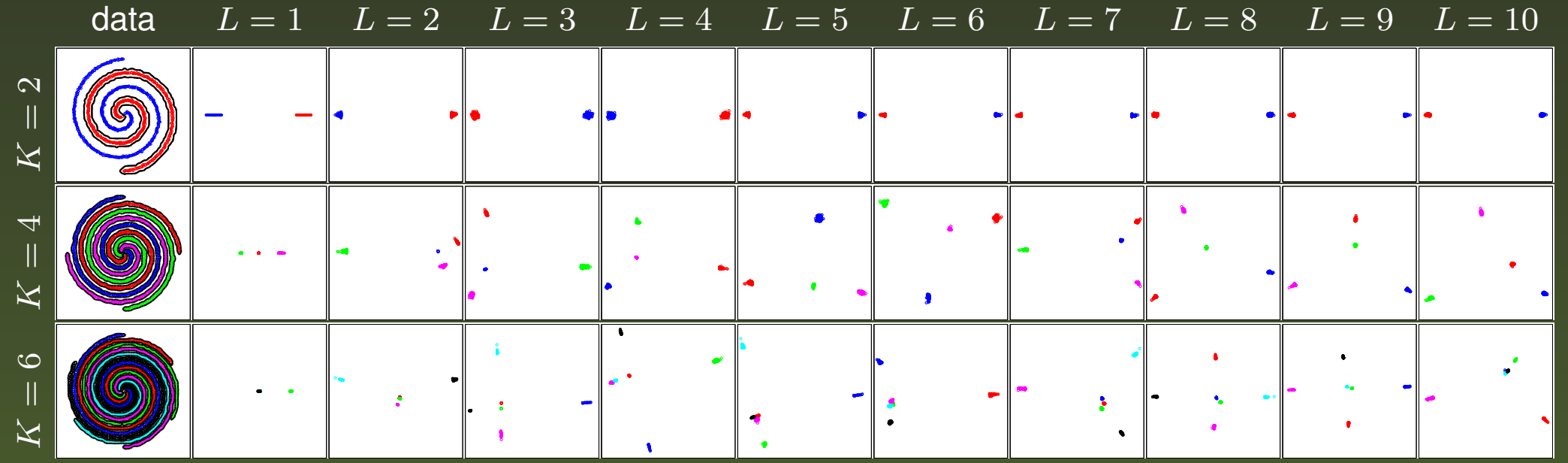
Configuration of the low-dim projections as a function of the number of basis functions M: the more flexible $\mathbf{F}$ is (larger M), the more the classes collapse and separate.

$\mathbf{F}$ linear | $M = 4$ | $M = 10$ | $M = 20$ | $M = 40$ | $M = 100$ | $M = 200$ | $M = 2000$

☞

❷ Role of dimension reduction: experiments (cont.)

Relation between the dimension L and the number of classes K: the classification accuracy improves drastically as L increases from 1 and stabilizes at $L \approx K - 1$, by which time the training samples are perfectly separated and the classes form point-like clusters approximately lying on the vertices of a simplex.



- This gives a recipe to choose L: starting from $L = K - 1$, increase L until the classification error does not improve.
- Initialize the latent projections $\mathbf{Z}$ to the corners of a simplex.

❸ How good/fast is the nonlinear low-dim classifier?

Consider dimension reduction with a radial basis function mapping

$$\mathbf{F}(\mathbf{x}) = \sum_{m=1}^{M} \boldsymbol{\alpha}_m \phi_m(\mathbf{x})$$

each BF ϕ_m is e.g. a Gaussian.

Then the nonlinear low-dim classifier has the form of a nonlinear SVM:

$$y = \sum_{m=1}^{M} \mathbf{v}_m \phi_m(\mathbf{x}) + b$$

- We have direct control on the classifier complexity through the number of basis functions M

  unlike in an SVM, where the number of support vectors M is often very large.

- We can trade off classification error vs runtime:
  - Training: linear on sample size N and number of BFs M.
  - Testing: linear on number of BFs M.
- Error comparable to an SVM's, but with far fewer basis functions.

In general, using the same MAC algorithm, we can use other forms for $\mathbf{F}$ (e.g. a neural net) and need not reduce dimension.

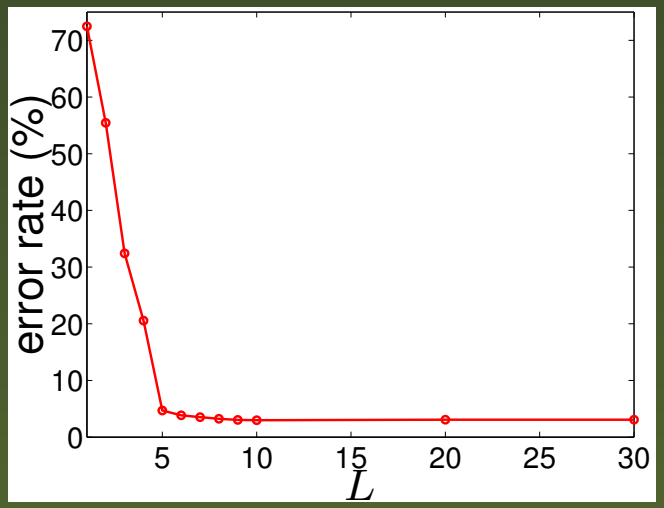
❸ How good/fast is the nonlinear low-dim classifier?

10k MNIST handwritten digit images of $28 \times 28 = 784$ dimensions with $K = 10$ classes (one-versus-all).

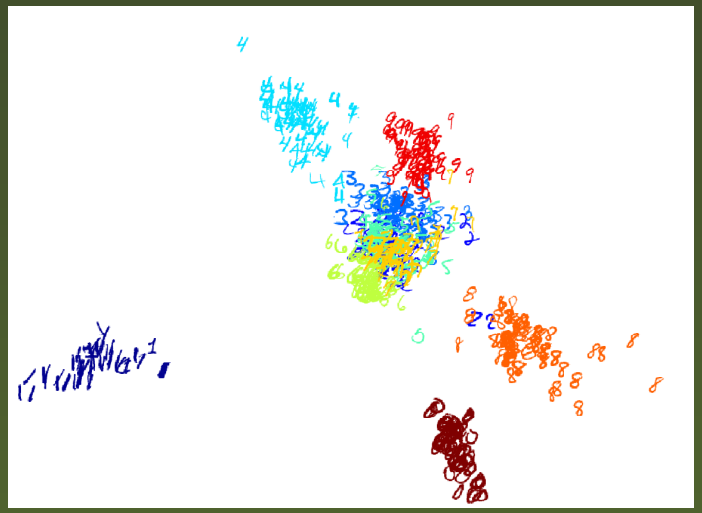
A low-dim SVM ($L = K$ dimensions) does as well as a kernel SVM but using 5.5 times fewer support vectors. The low-dim SVM is also faster to train and parallelizes trivially.

Method	Test error	# BFs
Nearest Neighbor	5.34	10 000
Linear SVM	9.20	–
Gaussian SVM	2.93	13 827
low-dim SVM	2.99	2 500
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
PCA (40) + low-dim SVM	2.60	2 500

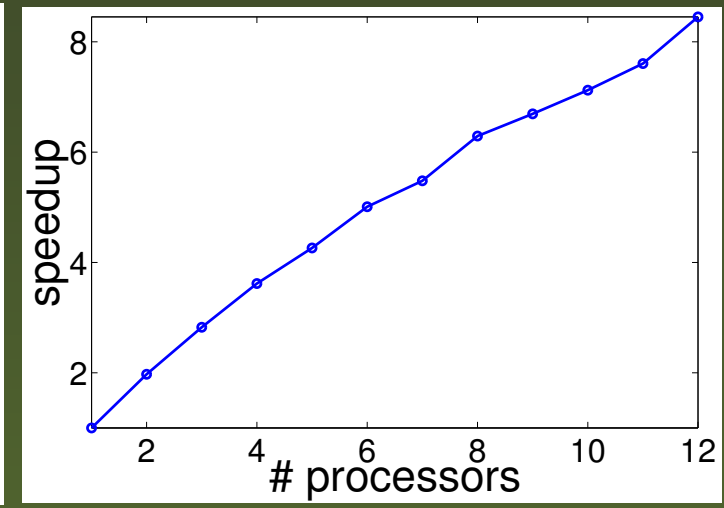
classification error wrt L



low-dim space (2D PCA)



parallel processing speedup



Conclusions

1. Learning optimal nonlinear low-dim features for a linear SVM is a nonconvex problem, but it can be solved easily and efficiently with the method of auxiliary coordinates.
   The algorithm is very intuitive: repeatedly fit a regressor and a linear SVM, and update each point's coordinates.
   It is like iterating a "filter" but correcting the coordinates so we converge to a true optimum.
2. It shows that classes collapse and maximally separate in latent space, so the optimal dimensionality reduction for linear classification destroys the manifold structure.
   This justifies filter approaches that maximize class separability and minimize intra-class scatter, but also obviates them—just train features and classifier jointly.
3. The resulting classifier is competitive with nonlinear SVMs but faster to train and at test time.
   Matlab code: `http://eecs.ucmerced.edu`.

Future work: what is the interplay of dimensionality reduction with a nonlinear classifier?

Formulation for the multiclass problem

- With K classes, we can use the one-versus-all scheme and have K SVMs, each of which classifies whether a point belongs to one class or not.
- The objective function is the sum of that of K SVMs.
- In the $\mathbf{Z}$ step, we solve for each point a quadratic program on $L + K$ variables and $2K$ constraints:

$$\min_{\mathbf{z}, \{\xi^k\}_{k=1}^K} \quad \|\mathbf{z} - \mathbf{F}(\mathbf{x})\|^2 + \sum_{k=1}^{K} C^k \xi^k$$

$$\text{s.t.} \quad y^k((\mathbf{w}^k)^T \mathbf{z} + b^k) \geq 1 - \xi^k, \; \xi^k \geq 0, \quad k = 1, \ldots, K.$$

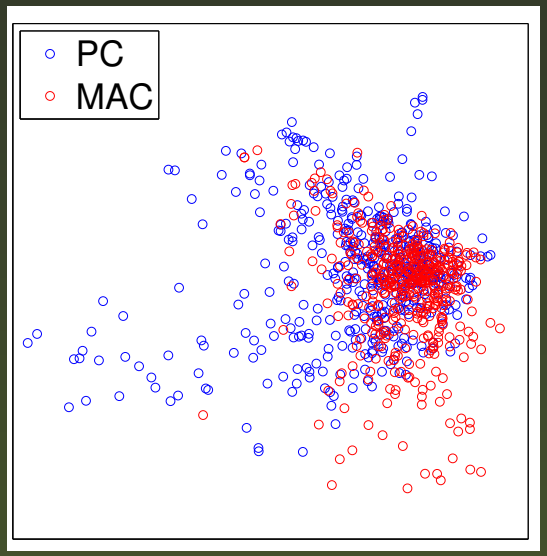
This has no closed-form solution, so we solve it numerically.

Document binary classification

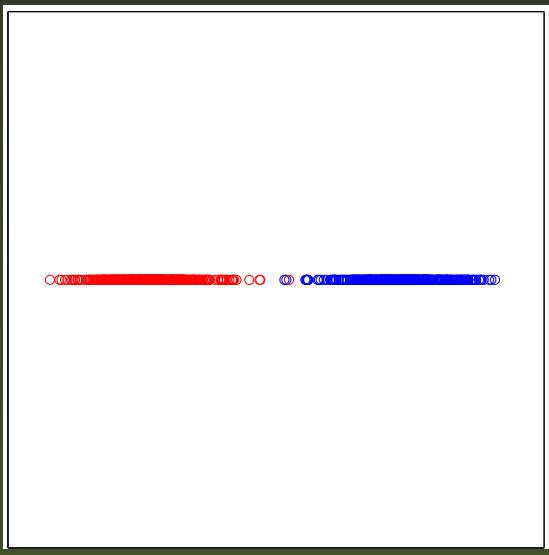
Results on the PC/MAC subset of 20 newsgroups.

methods	% error (std)
nearest neighbor	19.16 (0.74)
linear SVM	13.5 (0.72)
PCA ($L = 2$)	42.10 (1.22)
LDA ($L = 1$)	14.21 (1.63)
LMNN ($L = 2$)	15.91 (1.65)
low-dim SVM ($L = 1$)	13.12 (0.67)
low-dim SVM ($L = 2$)	12.94 (0.82)
low-dim SVM ($L = 20$)	12.76 (0.81)

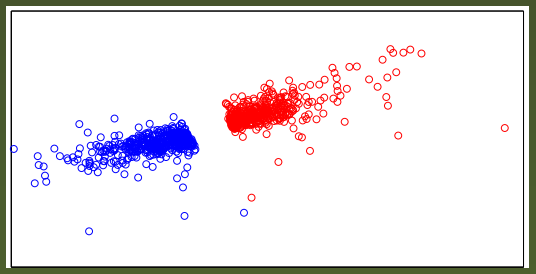
PCA



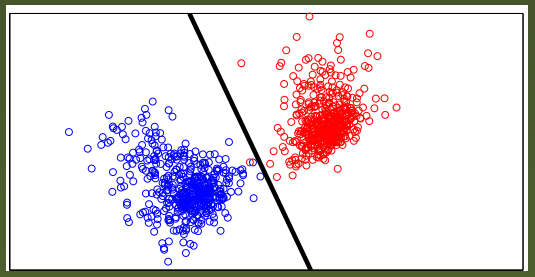
LDA



LMNN

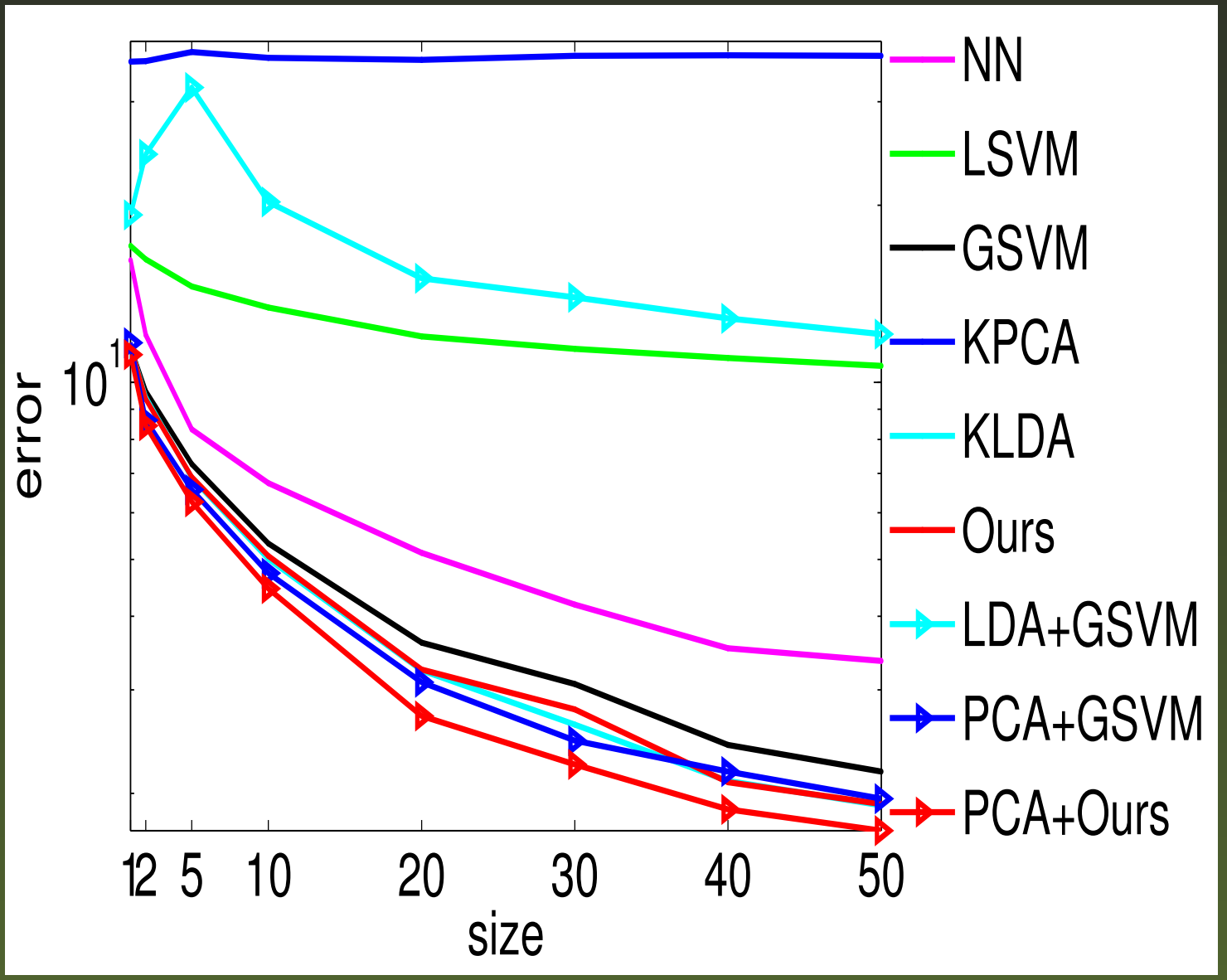


low-dim SVM

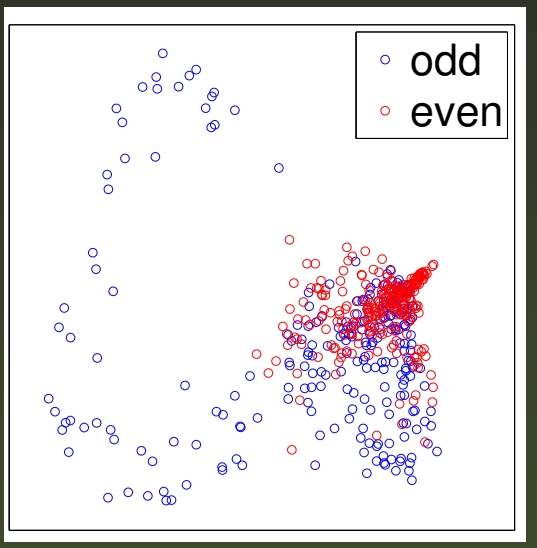


MNIST odd/even classification

errors of different algorithms wrt dataset size



KPCA



low-dim SVM