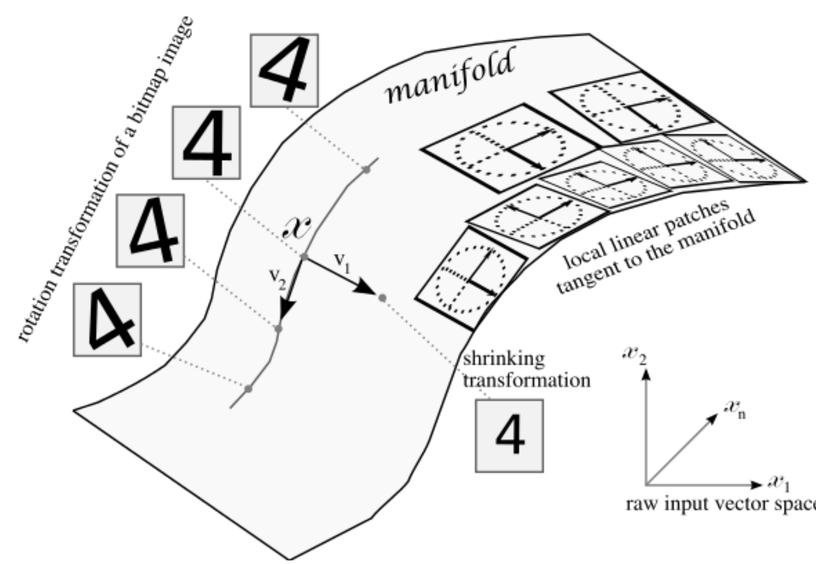
CSI 436/536
Introduction to Machine Learning

Kernel SVM

Professor Siwei Lyu
Computer Science
University at Albany, State University of New York

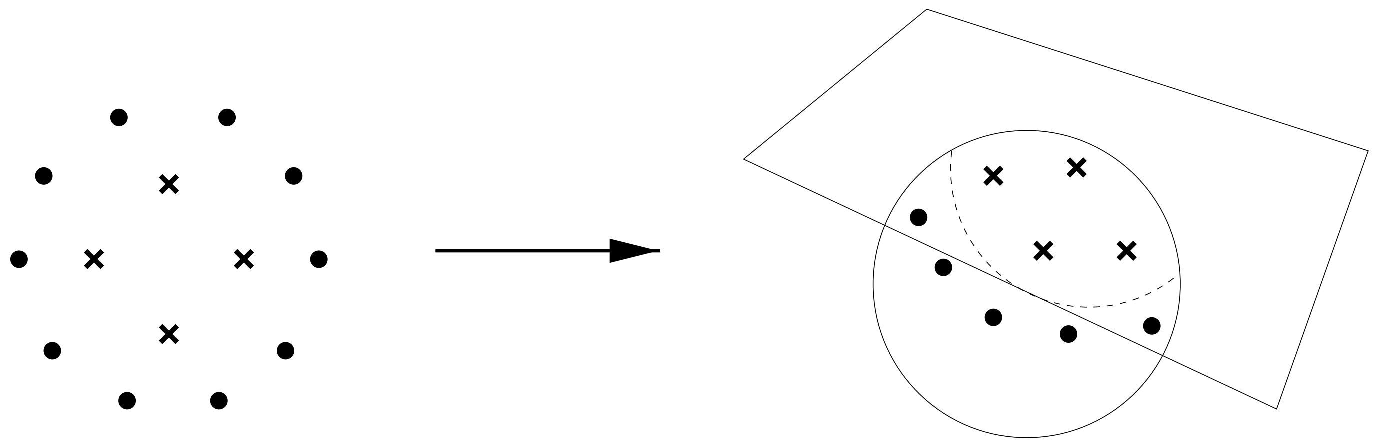
Leap from linear to nonlinear techniques

- So far we have mostly focused on linear techniques
- We need nonlinear analysis
- There are two general approaches to obtain nonlinear models
  - Directly design a nonlinear model
  - Convert a linear model via the “kernel trick” to get a nonlinear model



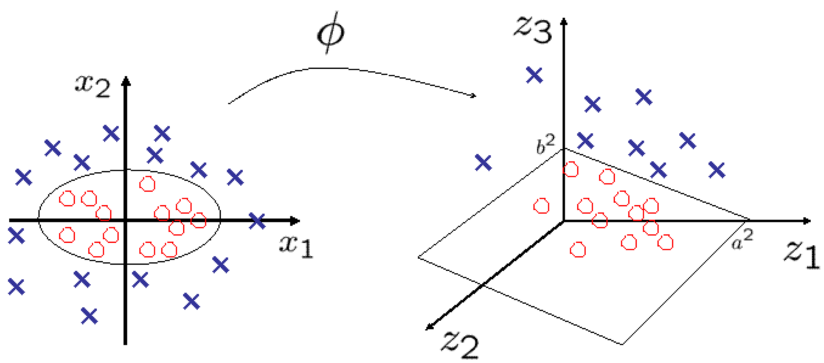
How to build nonlinear models?

- Consider classification problem
  - Nonlinearly transform data into a feature space
  - Build non-linear linear separation surface in the feature space
  - Transform back to the original space to obtain a nonlinear transform



Why this approach may work?

- Linearly non separable data can become linearly separable in a higher dimensional space



$$\phi : [x_1, x_2]^T \rightarrow [x_1^2, \sqrt{2}x_1x_2, x_2^2]^T$$

Elliptical decision boundary in the input space becomes linear in the feature space $\mathbf{z} = \phi(\mathbf{x})$:

$$\frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} = c \Rightarrow \frac{z_1}{a^2} + \frac{z_3}{b^2} = c.$$

What is the problem

- We may raise to very high dimension

Consider the mapping:
$\phi: [x_1, x_2]^T \rightarrow [1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2]^T$.
The (linear) SVM classifier in the feature space:

$$\hat{y} = \text{sign}\left(\hat{w}_0 + \sum_{\alpha_i > 0} \alpha_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x})\right)$$

The dot product in the feature space:

$$\begin{aligned}
\phi(\mathbf{x})^T \phi(\mathbf{z}) &= 1 + 2x_1z_1 + 2x_2z_2 + x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 \\
&= \left(1 + \mathbf{x}^T\mathbf{z}\right)^2.
\end{aligned}$$

The kernel trick

- Finding a feature map then a linear SVM classifier may not work when the feature map involves very high dimension (curse of dimensionality)
- The SVM training and testing only requires inner product between data points in the feature space
- That inner product can be computed using a function in the original space between a pair of training data, this is the kernel function
- Many algorithms can be “kernelized”
    - If we can covert them into a formulation only depend on inner products

Kernel SVM

- data linearly separable in the (infinite-dimensional) feature space
- We don’t need to explicitly compute dot products in that feature space – instead we simply evaluate the RBF kernel
  - avoid curse of dimensionality
- need to design kernel with domain knowledge
  - “no free lunch theorem: no universal kernel

Kernel functions

- kernel function computes inner product in the feature space from an implicit feature mapping
- can any function be a kernel function?
  - it has to be symmetric
  - it has to be positive when two inputs are same
  - it has to be zero when one input is zero
- It needs to satisfy the Mercer's condition

Mercer's condition

What kind of function K is a valid kernel, i.e. such that there exists a feature space $\Phi(\mathbf{x})$ in which $K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{z})$?

Theorem due to Mercer (1930s): K must be

- Continuous;
- symmetric: $K(\mathbf{x}, \mathbf{z}) = K(\mathbf{z}, \mathbf{x})$;
- positive definite: for any $\mathbf{x}_1, \ldots, \mathbf{x}_N$, the *kernel matrix*

$$K = \begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) & K(\mathbf{x}_1, \mathbf{x}_N) \\ . \quad . \quad . \quad . \quad . & . \quad . \quad . \quad . \quad . & . \quad . \quad . \quad . \quad . \\ K(\mathbf{x}_N, \mathbf{x}_1) & K(\mathbf{x}_N, \mathbf{x}_2) & K(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}$$

  must be positive definite.

★ Reproducing kernel Hilbert space

- A Hilbert space is an abstract vector space with a proper definition of inner product
- Defined properly, a Mercer kernel induces a space like that for functions $f_K(x) = K(.,x)$, with $<f_K(x),f_K(y)> = K(x,y)$, such a space is known as an RKHS with K being the reproducing kernel
  - This is a vector space with inf dimension
- On training dataset, a finite vector space is formed by $K(x_1,.),\ldots, K(x_m,.)$
- We have the representer's theorem stating that solutions to regularized LSE in such space is a vector in that space

Useful kernels

The linear kernel:

$$K(\mathbf{x}, \mathbf{z}) \;=\; \mathbf{x}^T\mathbf{z}.$$

This leads to the original, linear SVM.

The polynomial kernel:

$$K(\mathbf{x}, \mathbf{z};\, c, d) \;=\; (c + \mathbf{x}^T\mathbf{z})^d.$$

We can write the expansion explicitly, by concatenating powers up to d and multiplying by appropriate weights.

Radial basis function (RBF) kernels

$$K(\mathbf{x}, \mathbf{z}; \sigma) \; = \; \exp\left(-\frac{1}{\sigma^2}\|\mathbf{x} - \mathbf{z}\|^2\right).$$

The RBF kernel is a measure of similarity between two examples.

- The feature space is infinite-dimensional!

What is the role of parameter σ? Consider $\sigma \to 0$.

$$K(\mathbf{x}_i, \mathbf{x}; \sigma) \; \to \; \begin{cases} 1 & \text{if } \mathbf{x} = \mathbf{x}_i, \\ 0 & \text{if } \mathbf{x} \neq \mathbf{x}_i. \end{cases}$$

All examples become SVs $\Rightarrow$ likely overfitting.

Special kernel functions

- string kernels
  - texts, DNA sequences, etc
- Fisher kernels
  - probability distributions
- tree kernels
  - tree structures
- building kernels from similarity measures
  - Shoenberg's theorem
- Combining kernels to generate new kernels*

* topic of my first CVPR paper in 2005

Kernel SVM

The optimization problem:

$$\max \left\{ \sum_{i=1}^{N} \alpha_i \; - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \right\}$$

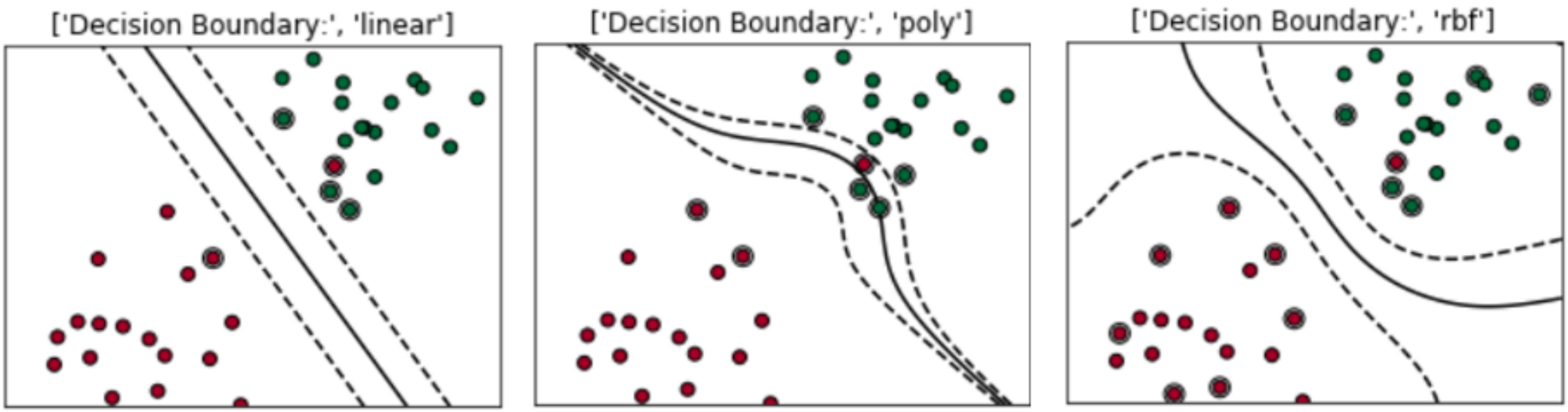
- Need to compute the *kernel matrix* for the training data

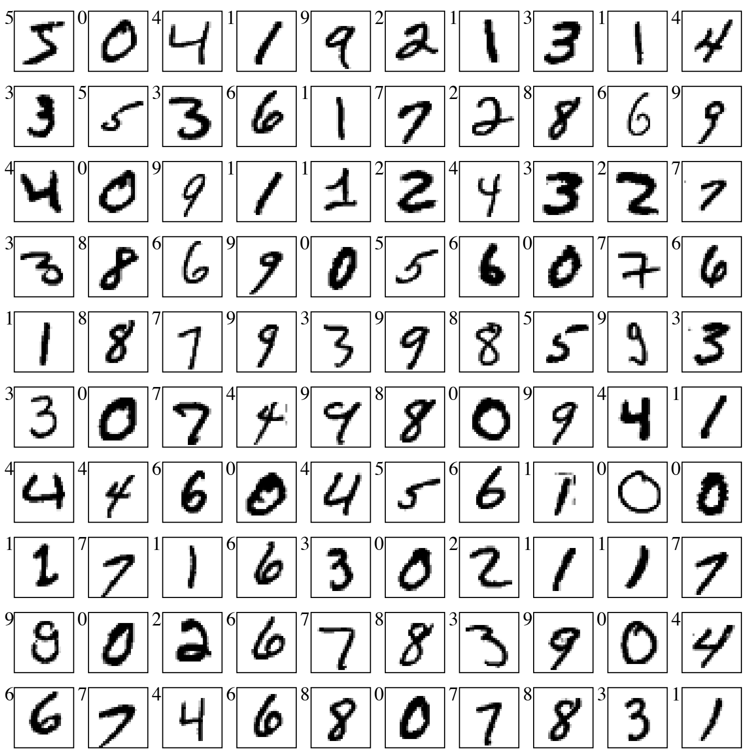
The classifier:

$$\hat{y} = \text{sign}\left(\hat{w}_0 + \sum_{\alpha_i > 0} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) \right)$$

- Need to compute $K(\mathbf{x}_i, \mathbf{x})$ for all SVs $\mathbf{x}_i$.

Kernel SVM



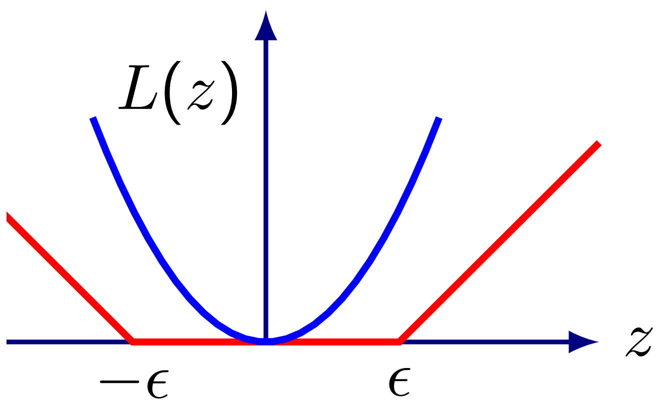


Classifier	Test Error
linear	8.4%
3-nearest-neighbor	2.4%
RBF-SVM	1.4 %
Tangent distance	1.1 %
LeNet	1.1 %
Boosted LeNet	0.7 %
Translation invariant SVM	0.56 %

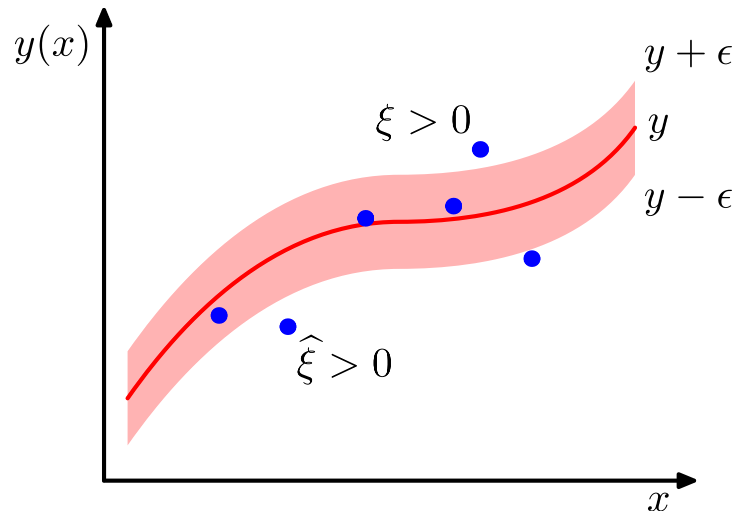
SV regression

The key ideas:

ϵ-insensitive loss



ϵ-tube



Two sets of slack variables:

$$y_i \leq f(\mathbf{x}_i) + \epsilon + \xi_i,$$
$$y_i \geq f(\mathbf{x}_i) - \epsilon - \tilde{\xi}_i, \qquad \xi_i \geq 0,\ \tilde{\xi}_i \geq 0.$$

Optimization: $\min C \sum_i \left(\xi_i + \tilde{\xi}_i\right) + \frac{1}{2}\|\mathbf{w}\|^2$

Kernel SVM

- Performance depends on the design of the kernel
- May lose the generalization guarantee as linear SVM — kernels may lead to infinite VC dimensions
- More recent trend focuses on designing good high dimensional features and then use linear SVM

Kernelizing other algorithms

- linear algorithms that can be re-written in the form of depending only on inner products
  - PCA/kernel PCA
    - ISOMAP and MDS is an instance of kernel PCA
  - LDA/kernel LDA
  - k-means/kernel k-means
  - CCA/kernel CCA
  - LSE/Kernel LSE