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.

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:
\[ \hat{y} = \text{sign} \left( \hat{w}_0 + \sum_{\alpha_i > 0} \alpha_i y_i \phi(x_i)^T \phi(x) \right) \]

The dot product in the feature space:
\[ \phi(x)^T \phi(z) = 1 + 2x_1 z_1 + 2x_2 z_2 + x_1^2 z_1^2 + x_2^2 z_2^2 + 2x_1 x_2 z_1 z_2 \]
\[ = (1 + x^T z)^2. \]
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 $\Phi(x)$ in which $K(x, z) = \phi(x)^T \phi(z)$?

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

- Continuous;
- symmetric: $K(x, z) = K(z, x)$;
- positive definite: for any $x_1, \ldots, x_N$, the kernel matrix

$$K = \begin{bmatrix} K(x_1, x_1) & K(x_1, x_2) & K(x_1, x_N) \\ \vdots & \vdots & \vdots \\ K(x_N, x_1) & K(x_N, x_2) & K(x_N, x_N) \end{bmatrix}$$

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:

\[
K(x, z) = x^T z.
\]

This leads to the original, linear SVM.

The polynomial kernel:

\[
K(x, z; c, d) = (c + x^T z)^d.
\]

We can write the expansion explicitly, by concatenating powers up to \(d\) and multiplying by appropriate weights.
Radial basis function (RBF) kernels

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

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

- The feature space is infinite-dimensional!

What is the role of parameter \( \sigma \)? Consider \( \sigma \to 0 \).

\[ K(x_i, x; \sigma) \to \begin{cases} 1 & \text{if } x = x_i, \\ 0 & \text{if } x \neq x_i. \end{cases} \]

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
  - My first CVPR paper
Kernel SVM

The optimization problem:

$$\max \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \right\}$$

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

The classifier:

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

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

Kernel SVM

For example MNIST hand-writing recognition. 60,000 training examples, 10,000 test examples, 28x28. Linear SVM has around 8.5% test error. Polynomial SVM has around 1% test error.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Test Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>linear</td>
<td>8.4%</td>
</tr>
<tr>
<td>3-nearest-neighbor</td>
<td>2.4%</td>
</tr>
<tr>
<td>RBF-SVM</td>
<td>1.4%</td>
</tr>
<tr>
<td>Tangent distance</td>
<td>1.1%</td>
</tr>
<tr>
<td>LeNet</td>
<td>1.1%</td>
</tr>
<tr>
<td>Boosted LeNet</td>
<td>0.7%</td>
</tr>
<tr>
<td>Translation invariant SVM</td>
<td>0.56%</td>
</tr>
</tbody>
</table>
SV regression

The key ideas:

- $\varepsilon$-insensitive loss

$$L(z) = \begin{cases} 0, & |z| \leq \varepsilon \\ |z| - \varepsilon, & \text{otherwise} \end{cases}$$

- $\varepsilon$-tube

Two sets of slack variables:

$$y_i \leq f(x_i) + \varepsilon + \xi_i,$$

$$y_i \geq f(x_i) - \varepsilon - \tilde{\xi}_i,$$

$$\xi_i \geq 0, \tilde{\xi}_i \geq 0.$$

Optimization:

$$\min C \sum_i \left( \xi_i + \tilde{\xi}_i \right) + \frac{1}{2} \| w \|^2$$
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
  - LDA/kernel LDA
  - k-means/kernel k-means
  - CCA/kernel CCA