CSCI 590: Machine Learning

Lecture 15:

Nonlinear SVM, VC theory revisited, Support vector regression

Instructor: Murat Dundar

Acknowledgements:
1. SVM Tutorial by C. Burges (http://research.microsoft.com/pubs/67119/svmtutorial.pdf)
2. PRML by C. Bishop

An observation in SVM formulation

$$L_d = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\mu_i\mu_j y_i y_j {x_i}^T x_j + \sum_{i=1}^{N}\mu_i$$

All the data appears in the training in the dot product form

Dual problem:

Maximize L_d

s. t. $\sum_{i=1}^{N}\mu_i y_i = 0$

$C \geq \mu_i \geq 0$

Kernel Trick (1)

Suppose we first map the data to some (possibly infinite dimensional) Euclidean space H, using a mapping ϕ

$$\phi: R^d \rightarrow H$$

Imagine solving the dual problem in H

$$L_d = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\mu_i\mu_j y_i y_j \phi(x_i)^T\phi(x_j) + \sum_{i=1}^{N}\mu_i$$

Kernel Trick (2)

If there were a “kernel function” K such that $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ we would only need K to evaluate L_d

There won’t be any need for explicit knowledge and execution of the mapping $\phi: R^d \rightarrow H$.

Kernel Trick (3)

The training phase is fine but how about testing? After all , w will live in H as well.

This is not an issue at all because we don't need to explicitly evaluate w during testing phase.

Recall that $w = \sum_{i=1}^{N} \mu_i y_i \phi(x_i)$

To evaluate SVM for a test sample $\phi(x_j)$ we need

$$f(x_j) = w^T x_j + w_0 =$$
$$\sum_{i=1}^{N} \mu_i y_i \phi(x_i)^T \phi(x_j) + w_0 =$$
$$\sum_{i=1}^{N} \mu_i y_i K(x_i, x_j) + w_0$$

Kernel Trick (4)

Suppose $x_i \in R^2$ and we choose $K(x_i, x_j) = (x_i{}^T x_j)^2$

For this kernel function it is easy to find the mapping $\phi: R^2 \rightarrow H$, such that $K(x_i, x_j) = (x_i{}^T x_j)^2 = \phi(x_i)^T \phi(x_j)$

In this case $H=R^3$ and $\phi(x) = \begin{bmatrix} x_1{}^2 \\ \sqrt{2}x_1x_2 \\ x_2{}^2 \end{bmatrix}$

Instead of explicitly mapping the data to $H=R^3$ through $\phi(x)$ we can produce the same effect on the classifier by replacing all the dot products with $K(x_i, x_j)$

Which functions qualify as a kernel?

Mercer's condition: There exists a mapping ϕ and an expansion and an expansion

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

if and only if, for any $g(x)$ such that

$\int g(x)^2 dx$ is finite then

$$\int K(x_i, x_j)\, g(x_i) g(x_j) dx_i dx_j \geq 0$$

This tells us whether or not a kernel function is actually a dot product in some space.

Isn't high dimension a "curse" for a classifier?

Why would mapping data to a "feature space" with a potentially infinite number of dimensions be good?

A standard classifier with a potentially infinite number of parameters can fit any training data we want but this usually comes at the cost of a dismal performance on test data.

Why would SVM be any different?

VC dimension of SVM

The VC dimension of SVM can be potentially as high as $\dim(H) + 1$, where $\dim(H)$ is the dimensionality of the embedding space H.

Recall VC bound, higher VC dimension implies a higher VC bound on test error.

Structural risk minimization does not quite rigorously explain why SVMs often have good generalization performance.

However, we have a theorem than links the probability of test error to the minimal enclosing sphere and the margin.

VC dimension of SVM

The VC dimension of SVM can be potentially as high as $\dim(H) + 1$, where $\dim(H)$ is the dimensionality of the embedding space H.

Recall VC bound, higher VC dimension implies a higher VC bound on test error.

Structural risk minimization does not quite rigorously explain why SVMs often have good generalization performance.

However, we have a theorem than links the probability of test error to the minimal enclosing sphere and the margin.

Expected test error

Theorem: $E[P(error)] \leq \frac{E\left[\frac{D^2}{M^2}\right]}{N}$

N: number of training samples
D: diameter of the smallest enclosing sphere
M: margin

Computing D requires solving a convex quadratic programming problem.

Alternative bound

Alternative bound on the actual risk of SVMs

$$E[P(error)] \leq \frac{E[Number\ of\ support\ vectors]}{N}$$

Interpretation: We can get an estimate of the test error by evaluating leave-one-out error with the training samples. If a left-out sample is not a support vector than w will not change and hence it will be correctly classified. Only those points that are support vectors have the risk of being misclassified. The worst case scenario is to have all the support vectors misclassified.

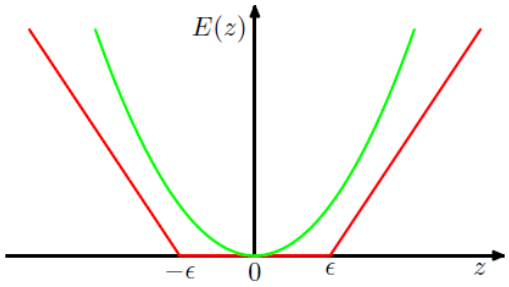
Support vector regression (1)

Regularized Regression problem:

$$\frac{1}{2}\sum_{i=1}^{N}\{f(x_i)-y_i\}^2+\frac{\lambda}{2}\|w\|^2$$

ε-insensitive Regularized Regression:

$$C\sum_{i=1}^{N}E_\varepsilon(f(x_i)-y_i)+\frac{\lambda}{2}\|w\|^2$$



$$E_\varepsilon(f(x_i)-y_i)=\begin{cases}0 & if\,|f(x_i)-y_i|<\varepsilon\\ |f(x_i)-y_i|-\varepsilon & otherwise\end{cases}$$

Support vector regression (2)

We introduce slack variables ξ_i and $\hat{\xi}_i$ and optimize the following problem

$$\min C \sum_{i=1}^{N} (\hat{\xi}_i + \xi_i) + \frac{1}{2}\|w\|^2$$
$$f(x_i) - y_i < \varepsilon + \hat{\xi}_i$$
$$f(x_i) - y_i > -\varepsilon - \xi_i$$
$$\xi_i \geq 0, \hat{\xi}_i \geq 0$$

Solving this problem generates a sparse solution to w

$$w = \sum_{i=1}^{N} (a_i - \hat{a}_i) x_i$$

a_i and $\hat{a}_i$ are both KKT multipliers.