Lecture 2 Report

EnHsien Chou
zex18@mails.tsinghua.edu.cn

Abstract

Reading Notes for Lecture 2, "Support Vector Machine"

1 Introduction

Support Vector Machine = Support Vector + Kernel Function. SVM is used on "classifying". We first consider dichotomy, i.e. separate data into two classes. No matter in what dimension, we can always use a hyperplane to cut the space into two parts, but what is the best one ?

2 Optimization Goal

Assume all data are in R^m. Given a training set, we want to find the best function $g(x)$ that matches the data (Supervised learning).

- Training Set : $S = \{(x_1, y_1), ..., (x_n, y_n)\}$, where y_i is the label of x_i, $y_i \in \{-1, 1\}$
- Linear Classifier : $g(x) = \omega^T x + b$, the hyperplane used for classification.
- Decision function : $f(x) = sgn(g(x))$, used to classify the input x into two classes

2.1 Functional Margin

For every input training data, we can check whether it is correctly classified or not by the sign of its functional margin:

$$\hat{\gamma}^{(i)} = y^{(i)}(\omega^T x^{(i)} + b)$$

The functional margin would be positive if and only if y^i and $g(x^i)$ are of same sign, meaning the data is correctly classified.

2.2 Geometric Margin

By normalizing function margin, we obtain geometric margin:

$$\gamma^{(i)} = y^{(i)} \frac{1}{||\omega||}(\omega^T x^{(i)} + b)$$

Geometric Margin can be viewed as the distance from the input data point to the hyperplane $g(x)$.

2.3 Optimization Problem

We define margin, the minimum of all the geometric margin.

$$\gamma = \min \gamma^{(i)}, i = 1, 2, 3..., n$$

A better function is the one with bigger margin. We can set $||\omega|| = 1$ for simplification. Therefore, our goal is:

$$\max_{\gamma,\omega,b} \gamma$$

with constrains:

$$y^{(i)}(\omega^T x^{(i)} + b) \geq \gamma$$

Notice that the solution forms a ball in space, difficult to optimize. By transforming the geometric margin to functional margin, and set the functional margin into 1, our goal is transformed into:

$$\min_{\omega,b} \frac{1}{2}||\omega||^2$$

with constrains:

$$y^{(i)}(\omega^T x^{(i)} + b) \geq 1$$

2.4 Lagrangian Form and Dual Problem

We can rewrite our problem into Lagrange form:

$$L(\omega, b, \alpha) = \frac{1}{2}||\omega||^2 - \sum_{i=0}^{N} \alpha_i [y^{(i)}(\omega^T x^{(i)} + b) - 1]$$

with constrains:

$$\alpha_i \geq 0, i = 1, 2, ..., N$$

By evaluating dual problems and KKT conditions, our problem can be transformed into:

$$\max_{\alpha}(-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y^{(i)}y^{(j)}x^{(i)}x^{(j)} + \sum_{i=1}^{N}\alpha_i)$$

with constrains:

$$\alpha_i \geq 0, i = 1, ..., N; \quad \sum_{i=1}^{N}\alpha_i y_i = 0$$

3 Support Vector

Simply solving the dual problem, we know ω has the form :

$$\omega = \sum_{i=1}^{N}\alpha_i y^{(i)} x^{(i)}$$

Notice that not all $\alpha_i > 0$. Those x_i s.t. $\alpha_i > 0$ are called support vectors, which determine the decision boundary.

4 Kernel Function

4.1 Feature space mapping

Sometimes, the data distribution can't be simply divided by easy hyperspace in a space V. Therefore, we can use a map to transform data into a feature space U.

$$\Phi : V \to U$$

Usually, U is of higher dimension than V. And we replace all x with $\Phi(x)$ in previous formulas.

4.2 Kernel Function

Kernel Function is the inner product on feature space of two vector $x_i, x_j \in V$:

$$K(x_i, x_j) =< \Phi(x_i), \Phi(x_j) >$$

Most of the time, we can define a kernel function as long as it has the properties of inner product. It is not necessary to find out what the map Φ is.

One commonly-used kernel function is the RBF kernel:

$$K(x, y) = e^{-\frac{||x-y||_2^2}{2\sigma^2}}$$

which is similar to normal distribution.

5 Implementation

See Appendix for SVM codes.

Acknowledgments

Thanks for the speecher, Yao XingCheng, and every one in our reading group. Additionally, thanks for all the learning materials provided from our leading teacher, Pro. Su Hang.

References

[1] Murphy, K. P. (2012). Machine learning: a probabilistic perspective. MIT press.

Appendix

Python Code—SVM

```
from sklearn import svm
clf = svm.SVC(C = 0.8,
              kernel = 'rbf',
              gamma = 20,
              decision_function_shape = 'ovo')
clf.fit(x_train, y_train.ravel() )
```