TP4 SVM: support vector machines

Yan Chen & Dajing GU

October 2020

In this TP, the algorithms of Support Vector Machines are analyzed and implemented in the program of Handwritten Digit Recognition. The MINIST Dataset is used. Code associated can be found in the GitHub repository: TP4 SVM: support vector machines .

1 Principle of PCA and SVM

Support Vector Machines belong to supervised learning methods for classification and regression. It's relatively new class of successful learning methods. They can represent non-linear functions and they have an efficient training algorithm.

Given training sample $D = \{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}, y_i \in \{-1, 1\}$. The main purpose of SVM is to find a hyperplane to divide samples of different classifications, as shown in Figure 1.1 [1].

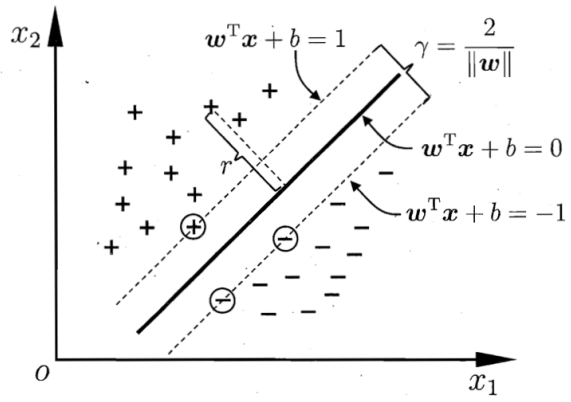

Figure 1.1: SVM and Margin

In the sample space, hyperplane can be described by the following equation:

$$w^T x + b = 0 \tag{1}$$

where $w = (w_1; w_2; ...; w_d)$ is the normal vector, which determines the direction of the hyperplane. b determine the distance between the hyperplane and the origin. $\oplus$ and $\ominus$

are the Supported Vectors. The sum of the distance between the two support vectors and the hyperplane is

$$\gamma = \frac{2}{\|w\|} \tag{2}$$

So the main target of SVM is to find the parameters w and b that make γ maximum, that is,

$$\begin{aligned} \max_{w,b} \frac{2}{\|w\|} &=> \min_{w,b} \frac{1}{2}\|w\|^2 \\ s.t.\ \ y_i(w^T x_i + b) &\geqslant 1, i = 1, 2, ..., m \end{aligned} \tag{3}$$

In order to solve the equation (3) to obtain w and b, Lagrange multiplier is a better method. And the dual problem appear. So the equation (3) can be described as the equation bellow:

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 + \sum_{i=1}^{m} \alpha_i (1 - y_i(w^T x_i + b)) \tag{4}$$

where $\alpha = (\alpha_1; \alpha_2; ...; \alpha_m)$. Let L take the partial derivative of w and b.

$$w = \sum_{i=1}^{m} \alpha_i y_i x_i \tag{5}$$

$$0 = \sum_{i=1}^{m} \alpha_i y_i \tag{6}$$

Combine the equation (4), (5), (6), we obtain the dual problem bellow:

$$\begin{aligned} \max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j {x_i}^T x_j \\ s.t.\ \ \sum_{j=1}^{m} \alpha_i y_i = 0 \\ \alpha_i \geqslant 0, i = 1, 2, ..., m \end{aligned} \tag{7}$$

So we just find the α to fit the equation (7), and we can obtain the value of w and b according to equation (5), (6). Finally, the hyperplane could be determined.

For the linear separable sample, SMO algorithm is a efficient method to find the best α. But for the non-linear separable sample, it's not useful. However, in real tasks, samples are often non-linear. For such problems, the samples can be mapped from the original space to a higher-dimension feature space. Than, we can transfer the equation (7) to the equation bellow:

$$\begin{aligned} \max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j) \\ s.t. \sum_{j=1}^{m} \alpha_i y_i = 0 \\ \alpha_i \geqslant 0, i = 1, 2, ..., m \end{aligned} \tag{8}$$

where $\phi(x)$ represents the feature vector after mapping.

But the dimension of the feature space is usually very large, or even infinite, it is very difficult to calculate $\phi(x_i)^T\phi(x_j)$. Kernel function has been a popular method to solve this problem. Finally, the equation (8) can be converted into the equation bellow:

$$\begin{aligned} \max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \kappa(x_i, x_j) \\ s.t. \quad \sum_{j=1}^{m} \alpha_i y_i = 0 \\ \alpha_i \geqslant 0, i = 1, 2, ..., m \end{aligned} \tag{9}$$

where $\kappa(x_i, x_j)$ is the kernel function.

$$\begin{aligned} f(x) &= w^T \kappa(x) + b \\ &= \sum_{i=1}^{m} \alpha_i y_i \phi(x_i)^T \phi(x) + b \\ &= \sum_{i=1}^{m} \alpha_i y_i \kappa(x, x_j) + b \end{aligned} \tag{10}$$

By solving the equation above, the multidimensional hyperplane can then be determined.

2 Inplementation of the algorithms

The MNIST handwritten digit recognition data set is used in our project, which comes from the National Institute of Standards and Technology (NIST). This data set consists of 250 handwritten numbers by different people, of which 50% are from high school students and 50% are from the staff of the Census Bureau.

The entire data set consists of three parts: the training set has 55,000 data, the validation set has 5,000 data, and the test set has 10,000 data. Although this data set sounds quite large, in fact, in machine learning, this is only an entry-level case. There are 55,000 images in the training set, each with a size of $784 = 28 \times 28$; the training set has 55,000 labels, and each label is a one-dimensional array of length 10 (10 numbers from 0 to 9).

The associated code is constituted by three parts: a simple implementation of PCA and SVM, an implementation with *make_pipeline* and finally a try with the *Grid_search*. The code and results can be found in the repository of GitHub .

In *Pipeline* we ensemble all the models needed and make them into one entity to have a clearer architecture of the models. In this way all the models can also be trained together.

Grid_search is a method to find the best parameters. Before training, all the hyperparameters are given in the grid, and the best models will be found during the training. In our code, the parameter **C** and different kernel choices are given for the *Grid_search*.

1) **C**: The C parameter tells the SVM optimization how much we want to avoid misclassifying each training example. For large values of C (Lower bias, high variance), the optimization will choose a smaller margin hyperplane. A very small value of C (higher bias, low variance) will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.

2) **Kernel**: The function of the kernel function is mapping from low-dimensional space to high-dimensional space. Two types of linearly inseparable points in the low-dimensional space can be turned into linearly separable in the high-dimensional space. Using 'linear' will use a linear hyperplane (a line in the case of 2D data). 'rbf' and 'poly' uses a non linear hyper-plane.

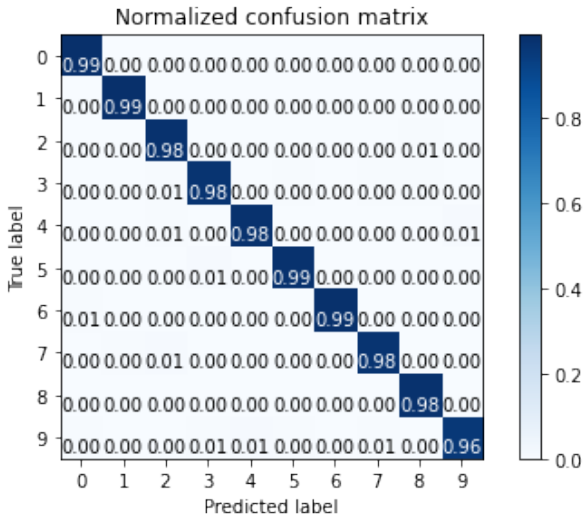

Figure 2.1: Confusion Matrix of the algorithms

The result of the classification is shown in figure 2.1, whose accuracy can reach 98.24%.

References

[1] Zhihua Zhou. *Machine Learning*. Qing hua da xue chu ban she, 2016.