Programming Assignment 6 Support Vector Machines

Part 1: Group member and contribution

Bin Zhang (5660329599): implantation of finding the fattest margin line including linear and kernel function, and report of implantation

Yihang Chen (6338254416): scikit-learn implementation of SVM including linear and nonlinear situation, and report of software familiarization

Hanzhi Zhang (4395561906): report of application

Part 2: Implementation

Linear separable

For this problem, we search for the Internet and find a QPP solver called CVXOPT to help us get the result. CVXOPT is a free software package for convex optimization based on the Python programming language and it has its own standard form to input the data and get the output of result.
The standard problem form of CVSXOPT is :

$$\begin{cases} min\ \frac{1}{2}x^T Px + q^T x \\ s.t.\ Gx \leqslant h \\ \quad Ax = b \end{cases}$$

So we need to define these parameters P, q, G, h, A, b with regard to the variable x (in our context that is α), here are the data structure we use to represent the parameters, we use $cvxopt.matrix$ to store them:
P is the Q matrix in our context:

$$Q(i,j) = y_i y_j x_i^T x_j \quad shape(100,100)$$

q is a matrix which every item in it is -1:

$$q = matrix(-np.ones((point\_num, 1)), tc = "d") \quad shape(100,1)$$

G is a minus identity matrix:

$$G = matrix(-np.eye(point\_num), tc = "d") \quad shape(100,100)$$

h is a matrix which all the item is 0:

$$h = matrix(np.zeros((point\_num, 1))) \quad shape\ (100,1)$$

A is a matrix which each item represents the corresponding label y:

$$A = matrix(y, (1, point_{num}), tc = \mathrm{d})shape\ (1,100)$$

b is a matrix which actually is a scalar 0:

$$b = matrix(0.0)$$

Once we prepare all the parameters, we can use the following function to get the α which is sol['x'] here. Then we can use α back to get the w and b .

$$from\ cvxopt\ import\ solvers,$$
$$sol = solvers.qp(P, q, G, h, A, b)$$

Once we prepare all the parameters, we can use the following function to get the α which is sol['x'] here. Then we can use α back to get the w and b .

When we calculate the b, we use the first α that exceeds 10^-6(which means is not 0) , and get the corresponding support vector to help us get the result.

The equation of the line is:

$$7.2500563x1 - 3.86188924x2 - 0.10698734 = 0$$

The support vectors are:

$$[0.24979414, 0.18230306]$$
$$[0.3917889\ , 0.96675591]$$
$$[0.02066458, 0.27003158]$$

Nonlinearly separable

The only difference between linear separable points and nonlinearly separable points is that we just need to change the product of X in Q with a kernel function and we can use the same QPP in the former operation.

In this assignment, we use Polynomial Kernel as kernel function, that is :

$$k(x, x') = (1 + x^T x')^2$$

Which could help turn the original data $x(x_1, x_2,)$ into 6 dimension

$$z(1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2)$$

The support vectors are :

$$[-8.47422847210542, 5.15621612964772]$$
$$[-10.2609690006984, 2.07391791404916]$$
$$[1.33933129592219, -10.2909882202102]$$
$$[9.67917723611986, 4.37595409614424]$$
$$[-6.80002273740268, -7.02384335261712]$$

$$[9.9014353820861, -0.31483148647915]$$

The equation of curve is :

$$g(x) = \sum \alpha_n y_n \, k(x_n, x) + y_m - \sum \alpha_n y_n \, k(x_n, x_m)$$

Since we have calculated the final result w and b, so the final equation is

$$g(x) = -8.65313390\text{e}-18 + 1.60702133\text{e}-01 * x_1^2 + 1.58698035\text{e}-01 * x_2^2$$
$$- 9.47654970\text{e}-03 * \sqrt{2}x_1 - 3.85759176\text{e}-02 * \sqrt{2}x_2$$
$$- 1.10221159\text{e}-03 * \sqrt{2}x_1x_2 - 16.66005058$$

w: [[-8.65313390e-18]

[1.60702133e-01]

[1.58698035e-01]

[-9.47654970e-03]

[-3.85759176e-02]

[-1.10221159e-03]]

b : -16.66005058

Part 3: Software Familiarization

Linear

We used sklearn python library for the implementation of SVM. Specifically, we used sklearn.svm.SVC for the first linearly separable dataset. Similarly, the mathematical formulation used in the SVC class is solving the QPP:

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i$$
$$\text{subject to} \quad y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i,$$
$$\zeta_i \geq 0, i = 1, \ldots, n$$

The dual is:

$$\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha$$
$$\text{subject to} \quad y^T \alpha = 0$$
$$0 \leq \alpha_i \leq C, i = 1, \ldots, n$$

e is the ones vector, C is the upper bound, Q is an n by n positive semidefinite matrix. $Q_{ij} \equiv y_i y_j K(x_i, x_j)$, where $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ is the kernel. And the decision function is:

$$\text{sgn}\left(\sum_{i=1}^{n} y_i \alpha_i K(x_i, x) + \rho\right)$$

For the linear separable data, we set the kernel to 'linear'.

$$\langle x, x' \rangle$$

More specifically, the linear kernel function is sum(x*x'), which is

summation of each attributes' product.

Need to be noticed here is that the mathematical methods used in *sklearn* for solving QPP has upper limit (C) to alpha. So we set C to the biggest number in python: float('inf'). Thus we can nearly remove the upper limit of alpha.

We got the score on the train set: 1.0, with coefficients: [[7.24837069 -3.86099178]] and constants: [-0.10703977]. Each side has 2 and 1 support vectors.

Nonlinear

The same as what has been mentioned. We use SVC class in the *sklearn* library and set kernel as 'rbf', which is the default value of the parameter 'kernel'.

The 'rbf' kernel function is like below:

$$\exp(-\gamma \parallel x - x' \parallel^2)$$

more specifically, the kernel function used in 'RBF' is

$$\mathrm{K}(\mathrm{x}, \mathrm{x}') = \exp\left(-gamma * sum\left(x - x'^2 \right)\right)$$

It can map an input space in infinite dimensional space. The parameter 'gamma' here ranges from 0 to 1. The higher value of gamma will perfectly fit the training dataset, which can cause over-fitting. Basically, the recommended value is 0.1

And we got score 1.0 on the train dataset. And each side has support vectors 45 and 50.

Part 4: Applications

Linear

Similar with logistic regression classier (LR), SVM can be a linear classifier. Therefore, lots of interesting applications discussed in assignment 4 can also be realized by SVM. These applications can be image recognition, text categorization and stock price direction prediction. Compared with SVM, LR has advantage on processing big data. For example, in stock price direction prediction, the financial company will provide millions of rows of data. In this case, LR has a higher computation speed. However, in some cases, we

might don’t have a large number of labelled instances. Under this circumstance, SVM is more likely to provide a more accurate prediction model than the one built by using LR. For example, SVM-based medical image classification. For some medical image of rare disease, the available image might be lower than 1,000. Furthermore, to judge one disease, tens of feature is included in one data. For small size high-dimensional spaces data, SVM is preferred.

Nonlinear

By applying kernel trick, SVM can become a nonlinear classifier. This nonlinear classifier is also applied on image recognition, text, categorization and stock price direction prediction which is the applications of linear classifier talked before. In reality, it is rare that a dataset is perfect linear separable. In linear classifier, SVM use a soft-margin with hinge loss function to realize an accurate model. For nonlinear classifier, prediction model can be more accurate with the help of kernel trick if overfitting can be avoided. Moreover, in some problems, the instances and results have a nonlinear relationship. In this case, we can only use SVM with kernel function. For example, climate forecast is a good example. By using forecast factor including the sea surface temperature, the southern Oscillation index, the subtropical high area index and the polar vortex area index, the nonlinear classification built by researchers to predict the climate change works well.