SVM Report

Zhongwei Wan
11612201
School of Computer Science and Engineering
Southern University of Science and Technology
Email: 11612201@mail.sustc.edu.cn

1. Preliminaries

1.1 Problem Description

In this project, we will use a support vector machine to classify a data set whose feature quantity is ten dimensions. The support vector machine is a two-class model. The basic model is defined as the linear classifier with the largest interval in the feature space. The learning strategy of SVM is to maximize the interval, and the problem we want to solve is transformed into a convex quadratic programming problem. In this lab, I will first do some research in how to define a margins and separate data with a large "gap", then , I try to transform the optimal margin classifier into a digression on Largrange duality[1]. Finally, I use kernels method to apply to high-dimensional feature spaces, and use SMO algorithm to accomplish this project.

1.2 Problem Application

Since the introduction of the classic SVM in the 1990s, due to its complete theoretical framework and practical applications, it has achieved a lot of good results. It has received extensive attention in the field of machine learning, and it has great development in theory and application.In practical applications, face detection, generator fault diagnosis, classification, regression, clustering, time series prediction, system identification, financial engineering, biomedical signal processing, data mining, biological information, text mining, adaptive signals Processing, splice site identification, database learning algorithm based on support vector machine, handwritten similarity word recognition, support vector machine function fitting in fractal interpolation, support vector machine based inertial navigation initial alignment system, rock-burst prediction Support vector machine, defect recognition, computer keyboard user authentication, video subtitle automatic positioning for extraction, speaker confirmation so on.

2. Methodology

2.1 Notation

In this SVM project, according to the mathematical definition of the SMO algorithm in the support vector machine, the variables I use are feature, label, C, t, alphas, b, error, error_cache, K_matrix, m, n. Next, I will introduce the specific meaning of my variables in the code separately.

Variables:

-feature: the matrix of data feature xi.
-label: the matrix of data label yi.
-C: constant C related to slack variable.
-t: tolerance coefficient, which modifies variables when KTT condition is violated.
-alphas: the matrix of Largrange coefficient.
-error : the error between its predict yi and label.
-error_cache: use a matrix to record a sign bit and error.
-K_matrix: kernel matrix.
-m: row number of feature label
-n: column number of feature label

2.2 Software and data structure

In this SVM project, I used python 3.6 and pycharm IDE to complete the code portion of the experiment. In addition, I will continue to use Numpy to implement some data structures. In this project, the main data structures are **matrix** and **linked list**, in which the matrix shows superior performance in point multiplication and cross multiplication.

In this experiment, I stored the data in the feature matrix and in the label matrix. And when the SMO algorithm is implemented, the two-dimensional matrix error_cache is used to store data flag bits and error information. In addition, I use the Kernel matrix to store the results of Gaussian kernel function calculations, mapping low-dimensional feature data to high dimensions. In the stochastic gradient descent algorithm, I will use the predict list to store the predicted results and compare the values with the label list to calculate the accuracy.

2.3 Model design

2.3.1 Formulate the problem

At the beginning of the experiment, I first clarified the requirements of the topic and used the support vector machine to classify the ten-dimensional data set within a limited time. After clarifying this problem, I will use the standard SMO algorithm and the kernel support vector machine to solve this high-dimensional data set problem.

2.3.2 Method to solve the problem

The classical algorithms in the SVM problem are the gradient descent method and the convex quadratic programming algorithm. In this lab, I use John Platt's SMO standard algorithm. The sequence minimum optimization problem is to convert the optimization problem into multiple small problems, in order to find the value of alpha and intercept b. The core idea of this algorithm is to find two alpha values in each loop. They satisfy two conditions. One is that the two alphas must be outside the interval boundary, and the other two aphas values are not subject to standard interval processing. Or not on the border. The best alphas pair to be optimized is then determined in the outer loop using the standard SMO algorithm.

After completing the SMO algorithm, I used the pegasos stochastic gradient descent algorithm. The core of the algorithm is to randomly select a training sample to calculate the gradient of the objective function and then go a specific step in the opposite direction. The complexity of the overall algorithm is $O(n/\lambda\ \varepsilon)$, and n is the dimension of the data set. The algorithm can be optimized to $O(d/\varepsilon\ \lambda)$.

2.3.3 Algorithm step by step

The framework for the specific steps of this SVM project is:

First, the ten-dimensional data set in this project is imported into the matrix of feature and label for storage. And create a class that stores variables, which contain important variables such as feature, label, C, t, alphas, b, error, error_cache, K_matrix, m, n.

Second, the values of alpha and intercept b are solved using a standard SMO algorithm. In this standard SMO algorithm, the parameter data is first initialized. In the outer loop, initialize the matrix of alphas, set the number of iterations and the convergence condition of the loop out, and iteratively select the first alpha value. Then, use the optimized SMO algorithm to select the second alpha value.

Third, in the inner loop, the data error value Error i calculated by the alpha_i obtained by the outer loop is first used. Second, according to the condition of the fault tolerance rate t, use the heuristic method to select alpha j . Save the previous alpha_i and alpha_j, and calculate the values of the upper and lower bounds L and H and the eta value, and choose whether to exit the loop according to the judgment condition. Second, update the values of alpha_i and alpha_j and calculate the value of the intercept b. Returns the newly obtained alpha and b values to the outer loop.

Fourth , According to the prediction function and the values of alpha and b, the predicted value is obtained, and the predicted value is compared with the real label value to calculate the correct rate of training.

Fifth, complete the comparing codes according to the pegasos stochastic gradient descent algorithm. Compare the accuracy and time of the SMO and Pegasos algorithms on the same data set. Compare and analyze two algorithms

2.3.4 Detail of algorithm

2.3.4.1 Architecture

Main function in SVM project :

--**load_data:** to get the data from command line.
--**calculateError**: calculate the error of label and predict
--**Guass_kerne**l: calculate the guass_kernel matrix.
--**heuristic_select**: select the second alpha with max step.
--**adjust_a**: adjust alphas according to H and L
--**outer_smo**: choose alpha_i
--**inner_smo**:choose alpha_j and calculate b.

2.3.4.2 pseudo codes of Algorithm

In this part, I will detail the main four algorithm pseudo-codes, they are **outer_smo, inner_smo** respectively. In addition, there is the **Pegasos** algorithm used in comparative analysis.

Outer_smo algorithm:

The purpose of this algorithm is to select the first alpha value under defined conditions.

Algorithm 1 : **outer_smo**

Input:feature, labels, C, t, kernel,iteration
Output:alpha, b
1: **initialize** class data, count, change = 0;
2: **initialize** flag = ture;
3: **while** (count < iteration **and** change>0 **or** flag):
4: change = 0

```
5:          if flag == true:
6:              for i in range(m):
7:                  change+ = inner_smo()
8:              count+=1
9:          else:
11:             choose index not in bound line
12:             for i in range(m):
13:                 change+=  inner_smo()
14:             count+=1
15:       if flag = true
16:          flag = false
17:       else if (change == 0):
18:           flag = true
19:    return b, alphas
```

Inner_smo algorithm:

The algorithm uses a heuristic method to select the second alpha value that satisfies the interval condition, and calculates the error value and updates the first and second alpha values to obtain the value of the intercept b.

Algorithm 2 : inner_smo

```
Input: i, data
Output:1 or 0
1: error_i = calculate_error(data, i)
2: if (error_i*label[i] satisfies t):
3:     j = heuristic_select(i, data, error_i)
4:     record_alpha_i = data.alphas[i]
5:     record_alpha_j = data.alphas[j]
6:     if label[i]!=label[j]:
7:         L = max{0,alphas[j]-alphas[i]}
8:         H = min{C, C+alpha[j]-alphas[i]}
9:     else:
11:        L = max{0,alphas[j]+alphas[i] - C}
12:        H = min{C,alphas[j]+alphas[i]}
13:    if  L == H
14:        return 0
15:    calculate_eta()
16:    if eta>=0:
17:        return 0
18:    update_alphas_j()
19:    update_error()
20:    if (alphas[j] - record_alphas_j < t):
21:        return 0
22:    update_alphas_i()
23:    update_error()
24:    b1 = calculate_b1()
25:    b2 = calculate_b2()
26:    b = (b1+b2)/2
27:    return 1
28:  else:
19:    return 0
```

Pegasos algorithm:

Using random gradient algorithm to obtain the value of w, and the prediction result is calculated and compared with SMO algorithm.

Algorithm 3: Pegasos

```
Input: S, lambda, t
Output: w(t+1)
1: for i in range(t):
2:    choose i(t)∈{1,...|S|}uniformly at random
3:    set n = 1 / lambda
4:    if y(it)<w(t),x(it)>1:
5:        set w(t+1) = (1-n*lambda)*wt + n*y(it)*x(it)
6:    else if (y(it)<w(t),x(it)>>=1):
7:        set w(t+1) = (1-n*lambda)*wt
8:    w(t+1) = min{1,1/lambda^(1/2)/||w(t+1)||}
9: return w(t+1)
```

3. Empirical Verification

In Empirical Verification, In this experiment, the data set I used is train_data.txt provided on sakai, using the SMO algorithm to find the values of alphas and b, and then comparing the predicted values obtained from the prediction function with the actual label values. And then I get the correct rate of training data.

3.1 Data set

The train_data.txt data set is a ten-dimensional data set with a size of 1300. In this project, I split the data set into 80% training set and 20% test set.

3.2 Test environment

The test environment for this experiment is pycharm community version 2018.2, python version is 3.6, then, the hardware part is, the host is Intel(R) Core I7-6700HQ, the benchmark speed is @2.60GHz, the kernel is 4, the logical processors are eight, L1 caches 256KB, L2 caches 1.0MB, and L3 caches 6.0MB. No multi-process was used in this experiment.

3.3 Performance measure

In this svm project, I use the standard SMO algorithm and the Gaussian kernel function to solve the alpha and b values, traverse all index values and select alphai according to the conditions, and then use the heuristic method to select the second alphaj. The data of 1300 converges at 40s and exits the loop, resulting in an average correct rate of 92%.

The stochastic gradient descent algorithm has better

results. In the ten-dimensional data set with 1300, the training time is only 0.75s, but the average training accuracy is 99.997%, and the time and accuracy are higher than SMO. algorithm. The reason for this result is that the ten-dimensional data set is a linear data set, and I use the Gaussian kernel function in the SMO algorithm. Due to the random selection of the inner loop of the SMO, the data accuracy is unstable.

SMO algorithm performance

	time	accuracy
train_70%	20.12s	92.74%
test_30%	0.52s	94.75%
train_80%	25.74s	93.31%
Test_20%	0.35s	95.54%

Pegasos algorithm performance

	time	accuracy
train_70%	0.75s	99.994%
test_30%	< 0.1s	99.997%
train_80%	0.84s	99.975%
Test_20%	< 0.1s	99.98%

I randomly shuffled the data set, dividing it into 70% for training and 30% for testing, and 80% for training and 20% for testing. Two tables are the performance of the two algorithms

3.4 Hype-parameters

In SMO, the hyper-parameter used in the experiment is the denominator value k in the Gaussian kernel calculation process. The meaning of this value is the reach rate, which is a speed hyper-parameter whose function value falls to 0., I set it to 150. And the other hyper-parameter is slack variable C. This parameter is used to control the maximum interval and to ensure that the function interval for most points is less than 1.0. the value I set it as 15. The value of the tolerance is used to determine whether the second alpha value is modified. If it is less than the set value 0.0001, it means that the second alpha value is not improved, then it jumps out of the inner loop .And the value I set is 0.0001.

In pegasos stochastic gradient descent algorithm, the hyper-parameters are learning rate and epochs. The learning rate plays an important role in updating the weight. If it is set too small, it will cause the training to be too slow. If it is set too large, the function will oscillate back and forth. Here, I set the learning rate to .0.1 and set the epochs to 200.

4.Conclusion

4.1 Result

The comparison between the final results of the SMO algorithm and the stochastic gradient descent algorithm shows that the training set is a linear training set, and the random gradient descent algorithm can obtain better accuracy and reduce the training time. And if the training set is a nonlinear high-dimensional data set, using the standard SMO algorithm may lead to better results.

4.2 Analysis

The purpose of this experiment is to master how to use svm to solve a high-dimensional data problem. The main algorithm used in this experiment is the standard SMO algorithm. The heuristic method in the standard SMO algorithm selects the second alpha value to speed up. The speed of the algorithm. The **advantage** of the algorithm I use is that for nonlinear data, using Gaussian kernel mapping to high-dimensional space can better deal with the two-class problem. The **disadvantage** is that if the algorithm uses a linear data set, the accuracy will be lower than a gradient descent method which doesn't use any kernel.

5.Acknowledgments

I am grateful to the teaching assistant Zhao because she helps me a lot. She gave me an enlightenment on the algorithm and answered some questions that are crucial to me.

6.References

[1] Schuldt, C. , Laptev, I. , & Caputo, B. . (2004). Recognizing human actions: a local SVM approach. *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*. IEEE.