Report for Support Vector Machine (SVM) Project

Wentao Ning 11610914
Computer Science and technology
Southern University of Science and Technology
11610914@mail.sustc.edu.cn

1. Preliminaries

Description: In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.[1]

Application: SVMs can be used to solve various real world problems. SVMs are helpful in text and hypertext categorization as their application can significantly reduce the need for labeled training instances in both the standard inductive and transductive settings. Hand-written characters can be recognized using SVM. The SVM algorithm has been widely applied in the biological and other sciences. They have been used to classify proteins with up to 90% of the compounds classified correctly. Permutation tests based on SVM weights have been suggested as a mechanism for interpretation of SVM models. Support vector machine weights have also been used to interpret SVM models in the past. Posthoc interpretation of support vector machine models in order to identify features used by the model to make predictions is a relatively new area of research with special significance in the biological sciences.[1]

1.1. Software

I wrote this project in Python 3.6 using editor Sublime Text 3 and IDE PyCharm.

1.2. Algorithm

At first I wrote a loadfile function to preprocess the file and get the train data and test data. Then I used pegasos algorithm to train our model and test. I also write SMO algorithm, but I find the accuracy of SMO algorithm is much lower than pegasos algorithm. Therefore I choose pegasos algorithm in the end.

2. Methodology

2.1. Representation

There are several variables in SVM.py.

- **data**: the data x1, x2, x3 ... x10 from given file.
- **labels**: labels (1 or -1) from given file.
- **lam**: parameter λ in the pegasos algorithm.
- **w**: vector w in the pegasos algorithm.

2.2. Architecture

There are 3 primary methods in my SVM.py file.

- **load**: Preproccess the given file and get the data and labels.
- **pegasos**: Use pegasos algorithm to get the better vector w
- **SMO**: Use SMO algorithm to get the α and b

2.3. Details of algorithm

There is one primary function in my SVM.py file.

- **pegasos**: Using pegasos algorithm to get a better performance by training data

Algorithm 1 pegasos

$w_1 \leftarrow 0$, $t \leftarrow 0$
for $i = 1$ to T **do**
  for $j = 1$ to $len(data)$ **do**
    $t \leftarrow t + 1$
    $\eta \leftarrow \lambda * t$
    $p \leftarrow w * transpose(data[i, :])$
    if $labels[i] * p < 1$ **then**
      $w = (1.0 - 1/t) * w + \eta * labels[i] * data[i, :]$
    else
      $w = (1.0 - 1/t) * w$
    end if
  end for
end for
return w

3. Empirical Verification

3.1. Design

In the Empirical Verification part, I use the train_data.txt which is given by teacher and divide it into 2 parts. One part for training and the other part for testing. I split it in different proportions to get different trainning data and testing data set. Further more, I compared the efficiency between pegasos algorithm and SMO algorithm.

3.2. Data and data structure

- **Data Set**: train_data.txt file which is given by teacher.
- **Data Structure**: numpy matrix, list.

3.3. Performance & Hyperparameters

Performance Measurement: I run my algorithm on the given file which is divided into 2 parts, train_data.txt and test_data.txt. I test my SVM.py file to calculate the correctness rate using these two files. I test my SVM.py file using different train and test dataset. In the empirical part, I test my code in Windows Platform and run it in Python 3.6

Hyperparameters setting: In pegasos algorithm, the hyperparameter which I need to set is the value of λ. I set the λ to different values and get different results.

λ	**Accuracy**
1	97.27%
2	97.82%
3	98.09%
4	97.27%
5	97.00%

TABLE 1. ACCURACY OF DIFFERENT λ IN **PEGASOS**

3.4. Result

In the test part, I set λ in pegasos to 3 and using different dataset to get the results. I test the pegasos algorithm and SMO algorithm for 5 times and the accuracy is below. The average accuracy of pegasos algorithm is 97.75% and the average of SMO algorithm is 93.74%. Therefore pegasos algorithm is better than SMO alogorithm.

3.5. Conclusion

This project is to write a support vector machine algorithm which is a kind of machine learning algorithm. I used an algorithm called pegasos to solve this problem.

Advantage: The advantage of this algorithm is that it is very fast, it costs only 2 seconds to run 100 times iteration.

Disadvantage: The disadvantage of this algorithm is that it's accuracy is not high enough. Also, it need to take a long time to adjust the parameter to get a better performance.

Times	**Accuracy of pegasos**	**Accuracy of SMO**
1st	97.82%	92.35 %
2nd	98.24%	90.00 %
3rd	98.65%	95.88 %
4th	97.35%	95.88 %
5th	96.67%	94.57 %

TABLE 2. ACCURACY OF DIFFERENT DATASET BETWEEN 2 ALGORITHMS

What I need to do is that I need to adjust the hyperparameter to get a better performance. And compare this algorithm with other algorithms like Gradient Descent.

References

[1] "Support vector machine", En.wikipedia.org, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Support_vector_machine. [Accessed: 21- Dec- 2018].