Support Vector Machine

YingZhou 11610701
Computer Science and Engineering
SUSTech
11610701@mail.sustc.edu.cn

1. Preliminaries

1.1. Problem Description

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.

1.2. Problem Application[1]

- 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.
- **Classification of images**. Experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback. This is also true of image segmentation systems, including those using a modified version SVM that uses the privileged approach as suggested by Vapnik.
- **Hand-written** characters can be recognized using SVM

2. Methodology

2.1. Notation

TABLE 1: Notation

Notation	Description
X	the matrix of training data vectors
Y	the label of training data vector in X
λ	learning rate
$\vec{w}$	the coefficient vector of the hyperplane

2.2. Data Structure

Only NumPy array is used in vector operation.

2.3. Model Design

As the requirement of project 4 is clear, I only need use any related methods to train my model, for example *SMO* or *Gradient Decent Algorithm*. I decide to implement *Pegasos* approach a stochastic gradient descent way. For measuring the performance of *Pegasos*, I simply learn and import existing library to complete *SMO*.

2.4. Detail of Algorithm

2.4.1. Pegasos. It is repeated and redundant to copy the mathematical conversion and proof of the gradient descent method. What *Pegasos* does is to apply an optimization algorithm to find the $\vec{w}$ that minimizes the objective function f.

Algorithm 1 Pegasos

```
function PEGASOS(IT, λ)
    w1 = 0, t = 0
    for iter 1 to iter do
        for j 1 to |data| do
            t = t + 1
            η = 1/(tλ)
            if y_j(w_t x_j) < 1 then
                w_{t+1} = (1 − ηλ)w_t + ηy_j x_j
            else w_{t+1} = (1 − ηλ)w_t
            end if
        end for
    end for
end function
```

2.4.2. Sequential minimal optimization. *Sequential minimal optimization* (SMO) works by breaking down the dual form of the SVM optimization problem into many smaller optimization problems that are more easily solvable.

Since SMO approach is just used to measure the performance of *Pegasos* algorithm, I simply implement it by importing *sklearn.svm.LinearSVC* and adjust the hyperparameters.

3. Empirical Verification

3.1. Dataset

TABLE 2: Datasets

Name	number of attributes	number of instances	missing values
train_data	10	1.33K	No
breast_cancer	9	2.77K	No

3.2. Hyperparameters

Pegasos

- **iter** The number of iterations in algorithm 1. According to David Sontag's SVM powerpoint, fixed number of iterations $T = 20 * |data|$, I defaultly set $iter = 20$.
- **learning rate** The learning rate of *Pegasos*. I found that 0.05 is better than 0.01, so I just roughly set $learning\_rate = 0.05$.

SMO

- $max\_iter = 10000$
- $kernel_type = ' linear'$
- $C = 1.0$
- $epsilon = 0.001$

All the setting are referred LasseRegin's work.[2]

3.3. Performance Measure

The code is written in Python, compiled using Python 3.6.5 :: Anaconda. Only NumPy package is extra imported. Ubuntu 18.04.1, one processor is used, 32G RAM, Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz.

3.4. Experimental Result

Figure 1: The estimation error varying proportion of training data sets

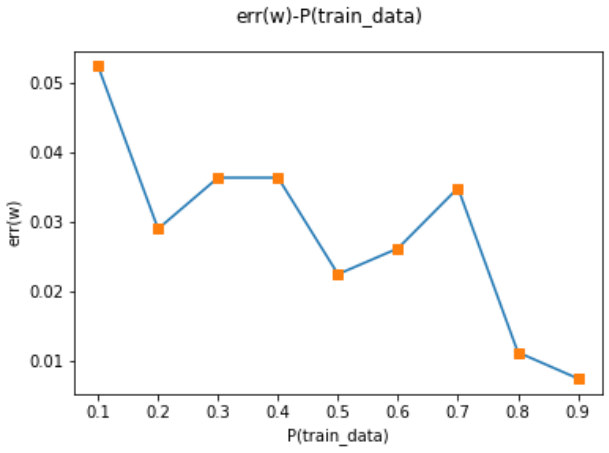

TABLE 3: The performance of SMO and Pegasos

dataset	S_err(w)	S_time(s)	P_err(w)	P_time(s)	learning rate
train	0.011	0.017	0.26	0.025	0.01
	0.011	0.017	0.26	0.022	0.05
breast cancer	0.027	0.168	0.37	0.283	0.01
	0.031	0.126	0.37	0.195	0.05

3.5. Conclusion

The *Pegasos* algorithm may have problems finding the minimum if the step length *learning rate* is not set properly. But it is boring and meaningless to test it under only two datasets. In general, *Pegasos* is a simple and efficient approach according to its code and performances. However, it seems that *SMO* is better than *Pegasos* during experiments.

References

[1] "Support vector machine", En.wikipedia.org, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Support_vector_machine#Applications. [Accessed: 31- Dec- 2018].

[2] "LasseRegin/SVM-w-SMO", GitHub, 2018. [Online]. Available: https://github.com/LasseRegin/SVM-w-SMO. [Accessed: 31- Dec- 2018].