Classification

JIE-YI LI

National Cheng Kung University Minn Wu School of Computing

nn6114069@gs.ncku.edu.tw

***Abstract*—This report shows various well-known methods of classification. Including linear classifier from scratch, linear classifier with least-squared manner, voted perception and SVM. Finally, I make a comparison by calculating accuracy between above mentioned method . Code is uploaded to GitHhub[1]**

Keywords—Linear classifier, SVM

I. INTRODUCTION

1. Linear Classifier from scratch
2. Linear classifier with least-squared manner : When the "J=WX+b" could be represented as the matrix form for the linear classifier, find the solution by solving this equation using least-squared manner.
3. Voted perceptron :
4. SVM (Hard margin) : With minimizing the ||w||^2, it should drive the marginal to be maximized as well.
5. SVM (Soft margin) : With minimizing the ||w||^2 and adding of slack variable, it should drive the marginal to be maximized as well.
6. SVM by sklearn

I adopted the strateries mentioned above to deal with the given 2 dataset : "data.csv" and "crx.csv" and evaluated their performances.

II. METHODOLOGY

A. *Problem Setting and Notation*

The dataset "data.csv" contains 569 cases, each case has its own ID and 30 features with label marked as "Diagnosis".

The dataset "crx.csv" contains 690 cases, each case has 15 features with label.

The problem is that I need to make a comparison between these methods to evaluate their performances.

For the input data, I use x_i to denote the features, y_i to denote the labels, w_i to denote the weights and b_i to denote the bias.

B. *Framework*

- Data Preprocessing

  1. Remove rows with missing value (Nan or '?') and drop the useless column (ex : ID in "data.csv")
  2. Modify the values or symbols in label column :
     For "data.csv", replaced 'M' with 1 and 'B' with -1.
     For "crx.csv", replaced '+' with 1 and '-' with -1.
  3. For "crx.csv", since some features were not recorded as numeric type, the strategy Onehot-encoding is proposed to encode those features with float type.

- Linear Classifier from Scratch

  The strategy provided in class :

repeat until convergence (or for some # of iterations):
for each training example ($f_1, f_2, \ldots, f_n$, *label*):

$$prediction = b + \sum_{i=1}^{n} w_i f_i$$

if *prediction* * *label* ≤ 0: // they don't agree
for each w_i:
$w_i = w_i + f_i$*label
$b = b$ + *label*

Figure 1 : Update rule of w_i

  In my code, *iteration* is set to 500 since I found that after about 500 times of iterations the performance can converge and get little or no improvement. For the predicted value, if value>0, then it will be categorized to 1, otherwise , -1, then the *prediction* will be 1 or -1.

- Linear Classifier with Least-Squared Manner

  The "J=WX+b" could be represented as the matrix form for the linear classifier. Thus the weight set W can be obtained by Least-squares solution method using linear algebra :

$$\hat{w} = (X^T X)^{-1} X^T Y \qquad (1)$$

- Voted perception

  1. Training (#iteration=20)

     Every time a mistake is made on an example:

     (1) Store the weights (i.e. before changing for current example)

     (2) Store the number of examples that set of

[1] Github : https://github.com/podo47/ML-Classification.git

weights got correct

2. Classify

(1) Calculate the prediction from all saved weights

(2) Multiply each prediction by the number it got correct and take the sum over all predictions

(3) Said another way: pick whichever prediction has the most votes

- SVM (Hard margin)

1. With minimizing the ||w||^2, it should drive the marginal to be maximized as well.

2. It can be interpreted as an optimize problem that we need to solve a quadratic program :

$$\min_x \frac{1}{2} x^T Q x + p^T x$$
$$\text{Subject to } Gx \leq h \quad (2)$$

3.

$$x = [\, w_0 \quad w_1 \quad w_2 \quad \cdots \quad w_M \,] \quad (3)$$

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \quad (4)$$

$$p = [\, 0 \quad 0 \quad 0 \quad \cdots \quad 0 \,] \quad (5)$$

$$G = -\begin{bmatrix} y_1 & y_1 x_1^1 & y_1 x_1^2 & \cdots & y_1 x_1^M \\ y_2 & y_2 x_2^1 & y_2 x_2^2 & \cdots & y_2 x_2^M \\ y_3 & y_3 x_3^1 & y_3 x_3^2 & \cdots & y_3 x_3^M \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_n & y_n x_n^1 & y_n x_n^2 & \cdots & y_n x_N^M \end{bmatrix} \quad (6)$$

Where M denotes number of features and N denotes number of cases.

- SVM (Soft margin)

1. With minimizing the ||w||^2 and adding of slack variable, it should drive the marginal to be maximized as well.

2. It can be interpreted as an optimize problem that we need to solve a quadratic program :

$$\min_{w,b} \|x\|^2 + C \sum_i \xi_i$$
$$\text{Subject to } y_i(w \cdot x_i + b) \geq 1\text{-}\xi \ \ \forall i$$
$$\xi_i \geq 0$$

3. Linear kernel : $K(x, x') = x^T x'$

- SVM by sklearn

SVC(kernel='linear') # Linear Kernel

III. Experiment Result

A. Accuracy

TABLE I. Accuracy of Each Method

Data Set	Methods					
	LC	***LS***	***VP***	***SVM (Hard)***	***SVM (Soft)***	***SVM (sklearn)***
data	0.9156	0.9649	0.9104	1.0	0.9824	0.9591
crx	0.6937	0.5345	0.6217	0.6018	0.8760	0.8827

From above table, “data.csv” got better results with all methods than “crx.csv. The reason might be that the features in “data.csv” are all numeric originally, but some features in “crx.csv” are nominal at the beginning. After Onehot-encoding, the additional transformation may cause some fallacious prediction.

For “data.csv”, hard margin SVM got greatest performance, and Voted perceptron got worsest performance. However, even the Voted perceptron did not get good result compared with other methods, it still get the good enough accuracy. Finally, based on the results, SVM is the greatest methods for “data.csv”.

For “crx.csv”, SVM by sklearn got greatest performance, and Linear classifier with least-squared manner got worsest performance as it only predict about half of the case correctly . Still, based on the results, SVM is the greatest methods for “crx.csv”. And for three kinds of SVM methods, Soft margin Svm and SVM by sklearn got better performance than Hard margin SVM obviously.

B. Margin

TABLE II. Comparison of Margin

Data Set	Methods	
	LC (Scratch)	***SVM (Hard)***
data	0.0001	4.1371e-05
crx	0.0017	0.4905

For “data.csv”, hard margin *SVM* has smaller margin than Linear classifier. However, for “crx.csv”, hard margin *SVM* has larger margin than Linear classifier.

C. *Effective weighting value C*

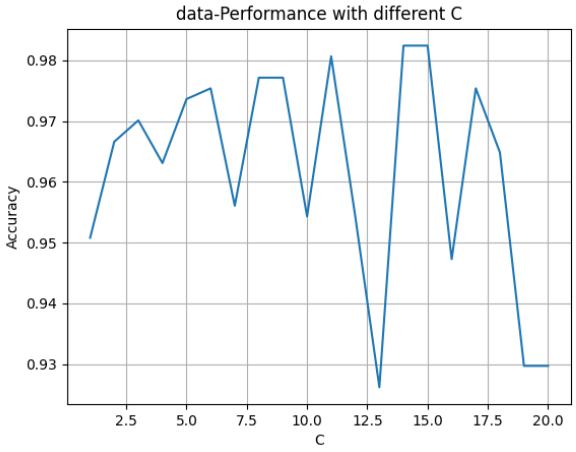

Figure 2 : Performance with different C (“data.csv”)

For “data.csv”, according to Figure 2, when using Soft margin SVM, the most effective weighting value C is 14 or 15, and the accuracy is 0.9824. I also nitice that even range of C is from 1 to 20, it seems that accuracy does not converge to a steady value.

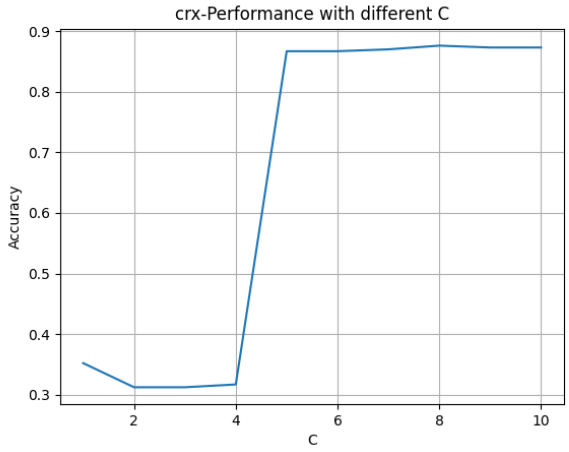

Figure 3 : Performance with different C (“crx.csv”)

For “crx.csv”, according to Figure 3, when using Soft margin SVM, the most effective weighting value C is 8, and the accuracy is 0.8760. I also nitice that after C reaches 5, different than fluctuation shown in “data.csv”, the accuracy converge to a steady value.

REFERENCES

[1] Tommy Huang, 機器學習-支撐向量機(support vector machine, SVM)詳細推導 [Website]
https://chih-sheng-huang821.medium.com/%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92-%E6%94%AF%E6%92%90%E5%90%91%E9%87%8F%E6%A9%9F-support-vector-machine-svm-%E8%A9%B3%E7%B4%B0%E6%8E%A8%E5%B0%8E-c320098a3d2e

[2] 劉智皓 (Chih-Hao Liu), 機器學習_學習筆記系列(15)：支撐向量機 (Support Vector Machine) [Website]
https://tomohiroliu22.medium.com/%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92-%E5%AD%B8%E7%BF%92%E7%AD%86%E8%A8%98%E7%B3%BB%E5%88%97-15-%E6%94%AF%E6%92%90%E5%90%91%E9%87%8F%E6%A9%9F-support-vector-machine-22512c96d947

[3] Emile Mathieu, An Efficient Soft-Margin kernel SVM Implementation In Python [Website]
https://emilemathieu.fr/posts/2018/08/svm/

[4] Support Vector Machine introduction
https://pythonprogramming.net/support-vector-machine-intro-machine-learning-tutorial/

[5] 子風的知識庫 [Website]
https://zwindr.blogspot.com/2017/06/ml-soft-margin-svm.html

Github: https://github.com/podo47/ML-Classification.git