Home Work Assignment 3: Support Vector Machines

Fatemeh Lotfi
University of Colorado, Colorado Springs
Colorado Springs, CO 80918
flotfi@uccs.edu

Introduction

Support Vector Machine (SVM) is a supervised learning model for data classification. The SVM constructs a hyper-plane or set of hyper-planes in a high- or infinite-dimensional space, which can be used for classification. In this paper, we learn how to develop and evaluate the SVM classifier algorithm in Python.

1. Linear discriminator for binary classification

Linear classification is a classification method that make decision based on the value of linear combination of the feature values. To explain this, assume a dataset D contains N example. The features of D are $\vec{x}^{(i)} = \{x_1^{(i)}, \cdots, x_n^{(i)}\}, i \in \{1, \cdots, N\}$, and $y^{(i)} \in \{-1, 1\}$ are the classes labels associated with the examples. The classification decision is made by a linear combination of features with parameters $\vec{\theta} = \{\theta_1, \cdots, \theta_n\}$, such as :

$$f(x^{(i)}) = \vec{\theta}^T \vec{x}^{(i)} + \theta_0 = \sum_{j=1}^{n} \theta_j x_j^{(i)} + \theta_0, \ i \in \{1, \cdots, N\}. \tag{1}$$

Then, by considering linear separability between dataset samples and the (1), the samples that are above the $f(x)$ belong to the class $+1$ and the samples that are below the $f(x)$ belong to the class -1. By this explanation, the objective function for a linear classifier can be written as:

$$\begin{aligned} \text{Find} \quad & f(x) = \vec{\theta}^T \vec{x} + \theta_0 = 0, \\ \text{s.t.,} \quad & f(x) \geq 1, \quad \text{for each point in the } \{+1\} \text{ class}, \\ & f(x) \leq -1, \ \text{for each point in the } \{\text{-}1\} \text{ class}, \end{aligned} \tag{2}$$

where, $f(x)$ is the decision boundary hyper-plane.

2. SVM margin-based binary discriminator

A margin-based classifier is a classifier that can provide a related distance from the decision boundary for each example. For instance, if a linear classifier is used, an example's distance from the dividing hyper-plane is its margin. Margin is significant in some classification algorithms like SVM because it can be used to limit the classifier's generalization error. The SVM is a Margin-based binary classification that it's aim is to find the hyper-plane with the greatest difference between the training points for classes $\{+1\}$ and $\{-1\}$ according to Fig. 1. As is shown in Fig. 1, the best margin on both sides is the one that has the maximum distance between data points of both classes. Margin is defined as the distance between a line that is tangent to the green dots and a line that is tangent to the red dots. The optimization problem for SVM classifier, by considering M as a margin and (1), can be written as:

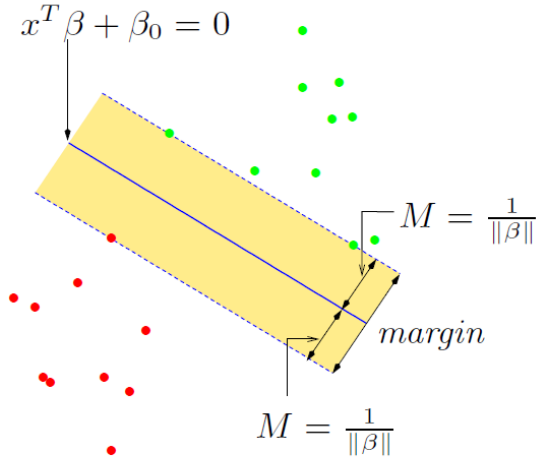

Figure 1: Support vector classifiers and their margins(Hastie, Tibshirani, and Friedman 2001)

$$\begin{aligned} \max \quad & M, \\ \text{s.t.,} \quad & f(x) \geq 1, \quad \text{for each point in the } \{+1\} \text{ class}, \\ & f(x) \leq -1, \ \text{for each point in the } \{\text{-}1\} \text{ class}. \end{aligned} \tag{3}$$

Now, to compute the M value, we assume a plane p is given as $\vec{\theta}\vec{x} + \theta_0 = 0$ and two arbitrary points p_1, p_2 in the plane p. Because these two points are in the plane, $\vec{\theta}\vec{p}_1 + \theta_0 = 0$ and $\vec{\theta}\vec{p}_2 + \theta_0 = 0$ which means that $\vec{\theta} \cdot (\vec{p}_1 - \vec{p}_2) = 0$. Therefore, $\vec{n} = \frac{\vec{\theta}}{|\theta|}$ is a unit normal to plane p given by $\vec{\theta}\vec{x} + \theta_0$.

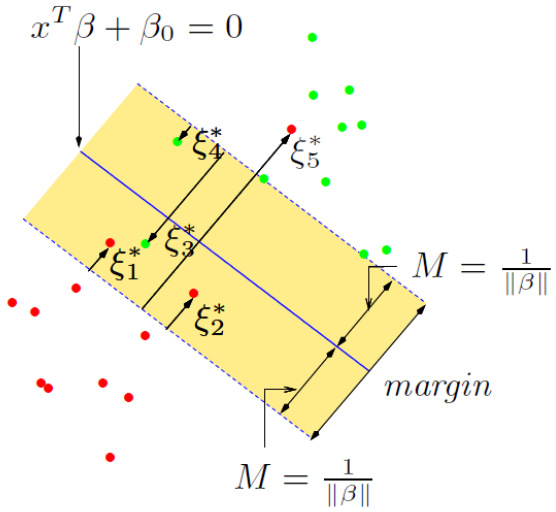

Figure 2: Miss-classified samples in support vector machine classifier(Hastie, Tibshirani, and Friedman 2001)

Let assume any x^+ points on the $+1$ class and any x^- points on the -1 class. We can define M as a project of $\vec{x}^+ - \vec{x}^-$ onto plane p given by $\vec{\theta}\vec{x} + \theta_0$. So, we have

$$\begin{aligned} M &= (\vec{x}^+ - \vec{x}^-) \cdot \vec{n}, \\ &= (\vec{x}^+ - \vec{x}^-) \cdot \frac{\vec{\theta}}{|\theta|}, \\ &= \frac{\vec{x}^+ \cdot \vec{\theta} - \vec{x}^- \cdot \vec{\theta}}{|\theta|} = \frac{2}{|\theta|}, \end{aligned} \tag{4}$$

where, M shows the total margin. By (4), the optimization problem (3) can be written as:

$$\max \quad \frac{2}{|\theta|}, \tag{5}$$

s.t.,

$$y^{(i)}\left(\vec{\theta}\vec{x}^{(i)} + \theta_0\right) \geq 1, \ \forall x^{(i)}, i \in \{1, \cdots, N\},$$

3. SVM regularized loss function

Given the real situation, we have some miss-classifications samples such as Fig. 2. To minimize the number of miss-classified samples, we measure a loss function by considering the error rate of the correct classification area for each sample. By considering the Fig. 2, the loss function L can be written as:

$$L = \begin{cases} \epsilon_i = 0, & \text{for points above the + margin,} \\ \epsilon_i = 1, & \text{for points on the classifier,} \\ \epsilon_i = 2, & \text{for points on the - margin,} \\ \epsilon_i = 1 - f(x^{(i)}), & \text{other points,} \end{cases} \tag{6}$$

where, it can be rewrite as a simple mode :

$$L = \begin{cases} \epsilon_i = 0, & \vec{\theta}\vec{x}^{(i)} + \theta_0 = 1, \\ \epsilon_i = 1 - \left(\vec{\theta}\vec{x}^{(i)} + \theta_0\right), & \text{O.W.} \end{cases} \tag{7}$$

Which (7) can be write as a general function

$$L = max\Big(0, 1 - y^{(i)}\left(\vec{\theta}\vec{x}^{(i)} + \theta_0\right)\Big), i \in \{1, \cdots, N\}. \tag{8}$$

The (8) is called hinge loss function.

As we mentioned in this section, this loss function is considered to minimize the number of miss-classified samples. On the other hand, the goal of the SVM is to find the widest margin. So, the general SVM loss function $\mathcal{L}$, by considering (8) and (5) would be written as:

$$\mathcal{L} = \frac{c}{2}|\theta|^2 + \frac{1}{N}\sum_{i=1}^{N} max\Big(0, 1 - y^{(i)}\left(\vec{\theta}\vec{x}^{(i)} + \theta_0\right)\Big), \tag{9}$$

where the first part is the loss component required to wide margin and the second part is the loss component to make a few error numbers as possible. Furthermore, as we know regularization is a procedure that penalizes massive coefficients in the solution vector to prevent overfitting. In (9), the first part is both the loss component for a wide margin and an L2 regularization element. Also, c is a hyper-parameter that controls the trade-off between a large margin and a small hinge-loss.

4. Stochastic gradient descent and SVM classification implementation

By performing the stochastic gradient descent (SGD) and decreasing the loss function, the SVM tries to find the widest margin and maximize the distance between the decision boundary and two class samples. As we mentioned in previous sections, the SVM loss function is according to (9). To implement the SGD algorithm and its update equation, we need to write a gradient of the loss function. Since the hinge loss part of the SVM loss function is not differentiable, we should consider a sub-gradient of the SVM objective function. So, this sub-gradient is written as:

$$\nabla\mathcal{L}^{(i)} = \begin{cases} c\theta, & \max\left(0, 1 - y^{(i)}f(\vec{x}^{(i)})\right) = 0, \\ c\theta - \frac{1}{N}y^{(i)}\vec{x}^{(i)}, & \text{O.W.} \end{cases} \tag{10}$$

where $f(x^{(i)}) = (\vec{\theta}\vec{x}^{(i)} + \theta_0)$. Now, the SGD update equation based on the (10) would be:

$$\theta_{t+1} = \theta_t - \alpha\nabla\mathcal{L}, \tag{11}$$

where α is the SGD learning rate.

5. Analysis of results

In this section, we show the SGD-SVM classification results for two datasets from the UCI Machine Learning Repository.

Dataset 1: Skin Segmentation

This dataset contains 245056 sample data points, three features, and one label. In this section, we derive the SVM classification problem with the SGD algorithm. Fig. 3 shows some parts of these samples for feature 1 and feature 2. To classify the data with SVM, we consider $\alpha = 0.01$ as a learnign rate and $c = 0.001$ as a regularization

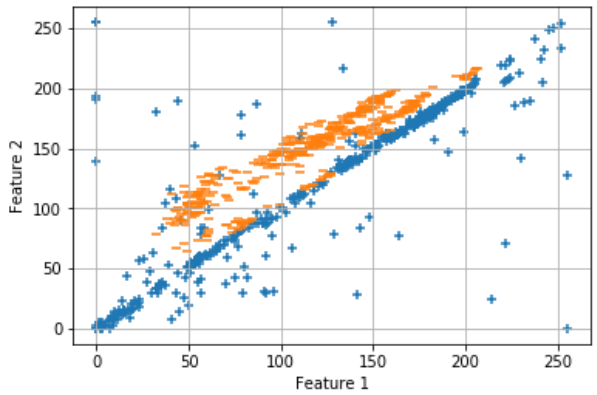

Figure 3: Some samples of feature 1 and feature 2 related to dataset 1

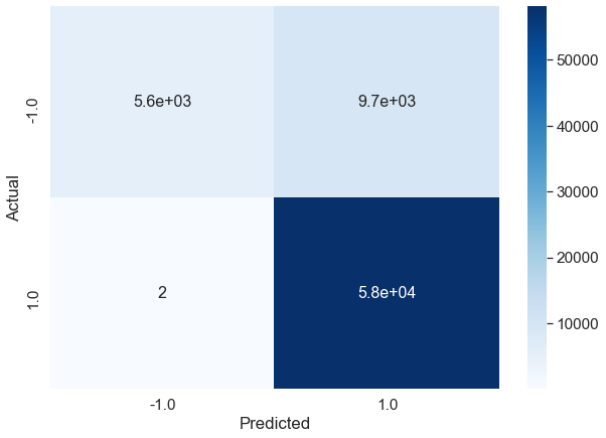

Figure 4: Confusion matrix of SVM classification for dataset 1

parameter. Finally, we find the learning parameters $\theta = \{0.022, 0.016, -0.03\}$ and $\theta_0 = -0.728$. Hence, the hyperplane model for the classification problem can be written as

$$f(x) = 0.022x_1 + 0.016x_2 - 0.03x_3 - 0.728. \quad (12)$$

Fig. 4 shows the confusion matrix of this binary classification problem.

Dataset 2: Planning Relax

This dataset contains 181 sample data points, 12 features, and one label. Fig. 5 shows some parts of these samples for feature 1 and feature 2. In this section we classified the data with SVM classification and SGD algorithm. To do this, we choose $\alpha = 0.01$ as a learning rate in SGD algorithm and $c = 0.001$ as a regularization parameter in SVM algorithm. Finally, we find the learning parameters $\theta = \{-2.99e-04, -6.02e-04, 3.35e-03, -7.553e-04, -2.31e-03, -1.17e-03, 3.31e-04, 4.59e-05, 1.06e-04, -4.17e-03, -5.38e-03, 3.16e-04\}$ and $\theta_0 = 1.0003$. Hence, the hyperplane model for the classification problem

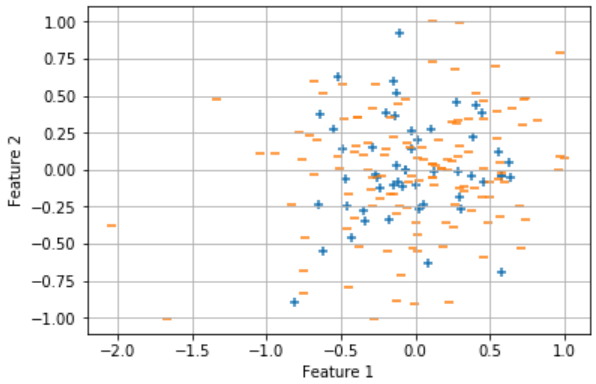

Figure 5: Some samples of feature 1 and feature 2 related to dataset 2

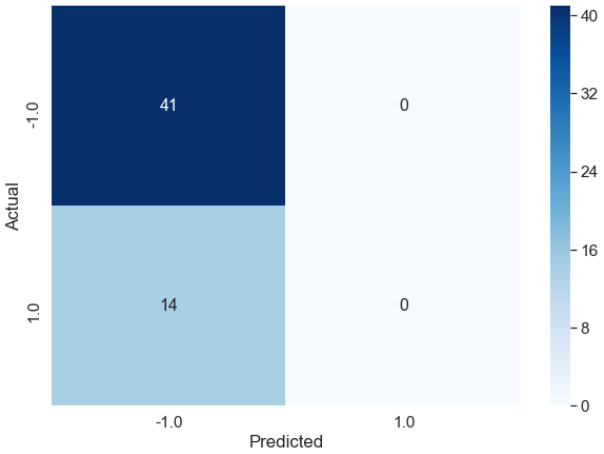

Figure 6: Confusion matrix of SVM classification for dataset 2

can be written as

$$\begin{aligned} f(x) = &-2.99e-04x_1 - 6.02e-04x_2 \\ &+ 3.35e-03x_3 - 7.553e-04x_4 \\ &- 2.31e-03x_5 - 1.17e-03x_6 \\ &+ 3.31e-04x_7 + 4.59e-05x_8 \\ &+ 1.06e-04x_9 - 4.17e-03x_{10} \\ &- 5.38e-03x_{11} + 3.16e-04x_{12} + 1.0003. \end{aligned} \quad (13)$$

Fig. 6 shows the confusion matrix of this binary classification problem.

6. SVM loss minimization problem with SGD

As we mentioned in section 4, we implement an SVM classification with stochastic gradient descent. In this section, we show that how the SGD solves the problem by minimizing the loss function according to (9). Fig. 7 and Fig 8, respectively, show the loss function of the SGD algorithm in each training iterations of dataset 1 and dataset 2.

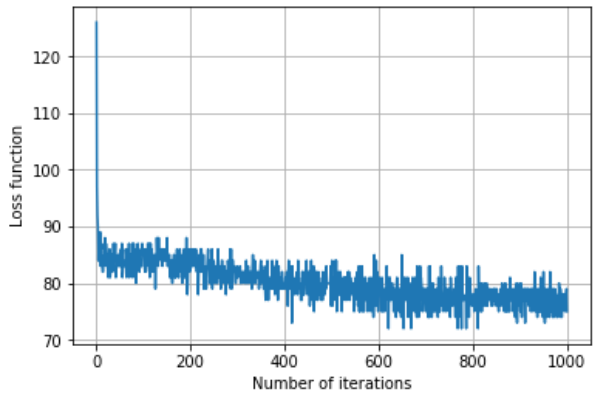

Figure 7: The loss function in each iteration of SVM classification with SGD algorithm of dataset 1

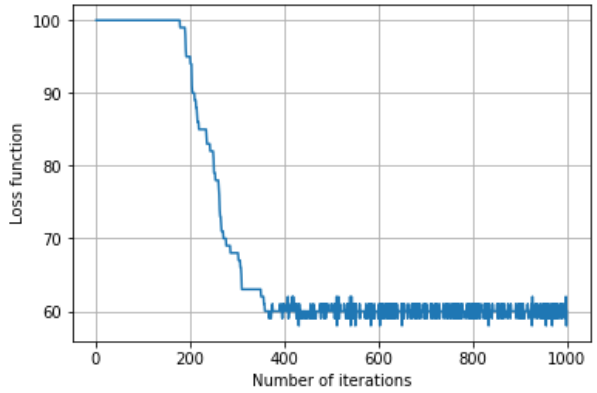

Figure 8: The loss function in each iteration of SVM classification with SGD algorithm of dataset 2

Effect of different values of regularization parameter (c) on the SVM performance

In this section we investigate the intuition of the regularization parameter c in (9) in SVM classification.

The regularization parameter c represents the importance assigned to miss-classifications. SVM poses a quadratic optimization problem that seeks to maximize the margin between two classes while minimizing miss-classifications. However, to find a solution for non-separable problems, the miss-classification constraint must be relaxed, which is done by setting the mentioned "regularization". As a result, it makes sense that as c grows larger, fewer miss-classified examples are allowed (or the highest price pay in the loss function). When c gets larger, the solution approaches the hard-margin (allow no miss-classification). More miss-classifications are allowed when c tends to 0. There is a tradeoff between the two, and smaller lambdas, but not too small lambdas, generally generalize well. Fig. 9 shows the effect of different values of c on the margin between two classes.

7. SVM implementation with Python library

In this section, we implement the SVM classification with the Python Sklearn library and compare their performance

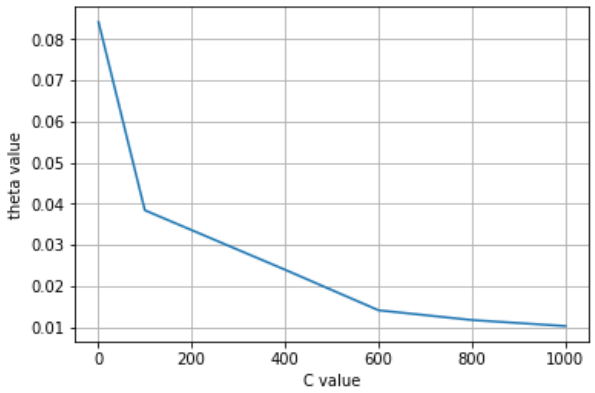

Figure 9: The effect of different values of c on the margin between two classes

with our implementation program. Also, we consider the accuracy of classification as a performance metric. Table 1 shows the results for dataset 1 and dataset 2.

Table 1: Comparison between our implemented SVM algorithm and Sklearn SVM algorithm

Accuracy	Dataset 1	Dataset 2
Our implemented SVM	93.48 %	73.03 %
Sklearn SVM	97.11 %	73.03 %

8. Compare SVM results with other algorithms

In this section, we want to compare the results of our SVM implementation, and library-based implementations of AdaBoost, Nearest Neighbor classification, and Logistic Regression. Table 2 shows the accuracy of each classification method. The classification methods have some hyper-

Table 2: Compare classification methods accuracy

Classification algorithms	**Accuracy of dataset 1**	**Accuracy of dataset 2**
Our implementation SVM	93.48 %	73.03 %
Sklearn AdaBoost	95.52 %	67.83 %
Sklearn nearest neighbor	99.95 %	65.45 %
Sklearn Logistic Regression	92.15 %	81.81 %

parameters that should be tuned correctly to achieve the best classification performance. In the follows we express the hyper-parameters of each method.

- AdaBoost
  The hyper-parameter in this method is the number of estimators or stumps. In Table 3 the effect of different number of estimators in classification accuracy are shown.

Table 3: The effect of AdaBoost hyper-parameter on the classification accuracy

Number of estimators	Dataset 1	Dataset 2
20	89.9 %	61.81 %
50	91.6 %	63.63 %
100	95.52 %	67.83 %

- Nearest neighbor
  The hyper-parameter in this method is the number of neighbors to participate in the KNN algorithm. Table 4 shows the effect of the different number of neighbors in classification accuracy.

Table 4: The effect of KNN hyper-parameter on the classification accuracy

Number of neighbors	Dataset 1	Dataset 2
2	99.95 %	64.54 %
3	99.95%	65.45 %
10	99.95 %	61.81 %

- Logistic regression
  Logistic regression hyper-parameters are similar to that of linear regression. The learning rate(α) and regularization parameter(λ) have to be tuned properly to achieve high accuracy. In this part, we show the effect of different values of regularization parameter as a hyper-parameter in classification accuracy in Table 5.

Table 5: The effect of logistic regression hyper-parameter on the classification accuracy

different values of λ	Dataset 1	Dataset 2
0.01	92.2 %	81.81 %
1	91.98%	72.7 %
100	91 %	69.09 %

Conclusion

In this report, we discuss the SVM classification method and learn how to implement SVM with SGD algorithm and classify the data. Also, we find the effect of different values of regularization parameter c on the margin between two classes. We implement the SVM with the Sklearn Python library and compare the accuracy of our implemented program with them to validate our implementation program. Finally, we compare results of our SVM implementation, and library-based implementations of AdaBoost, Nearest Neighbor classification, and Logistic Regression.

References

Hastie, T.; Tibshirani, R.; and Friedman, J. 2001. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc.