Programming Assignment 2

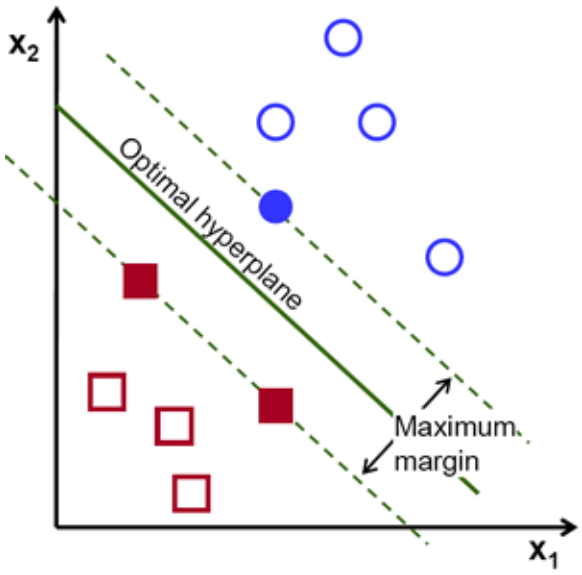
Varun Nagaraj

13th March, 2019

1 Support Vector Machines

1.1 Theory

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N—the number of features) that distinctly classifies the data points.



To separate the ten classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of ten classes.

Below is the confusion matrix when I ran the code with Radial Basis Function Kernel.

MNIST dataset Test

Python code to generate Confusion Matrix

```
[[ 967   0   1   0   0   5   4   1   2   0]
 [  0 1120   2   3   0   1   3   1   5   0]
 [  9   1 962   7  10   1  13  11  16   2]
 [  1   1  14 950   1  17   1  10  11   4]
 [  1   1   7   0 937   0   7   2   2  25]
 [  7   4   5  33   7 808  11   2  10   5]
 [ 10   3   4   1   5  10 924   0   1   0]
 [  2  13  22   5   7   1   0 954   4  20]
 [  4   6   6  14   8  24  10   8 891   3]
 [ 10   6   0  12  33   5   1  14   6 922]]
```

MNIST Accuracy is: 0.9435

Hyper parameter variation: Gamma defines how much of variance can be accepted by the model. If gamma is large, then variance is small implying the support vector does not have wide-spread influence. Technically speaking, large gamma leads to high bias and low variance models, and vice-versa.

Kernel	Accuracy
Linear Kernel-Default	93.9
RBF Kernel - Default	94.35
RBF Kernel -gamma=1.0	97.57

1.2 Identifying the Lagrangian Dual of a Primal Problem

Given features $(X1, Y1)$ .....(Xn, Yn), where $Y1...Yn$ belong to class (-1,1) and Xi is the set of features for which the corresponding result is Yi.

Minimize:

$$w^T.w + c\sum_{i=1}^{N} \epsilon_i \tag{1}$$

the weighted sum between the squared length of the separating vectors and the errors.

Where,

w = is the separating vector;

$w^T w$ = is the dot product of the transpose of the separating vector and itself;

ϵ_i = is the amount of error incurred for choosing the separating vector w for the features Xi,Yi.

Subject to,

$$Y_i.(w^T.X_i) \geq 1 - \epsilon_i \tag{2}$$

$$\text{Where, } \epsilon_i \geq 0 \text{ for i=1....n} \tag{3}$$

The following inequalities hold for all the n points in the training set.

$$X_i^T.w \geq 1 - \epsilon_i \tag{4}$$

$$X_i^T.w \geq -1 + \epsilon_i \tag{5}$$

$$\epsilon_i \geq 0 \tag{6}$$

where $Y_i = 1$ for equation 4 and Y_i = -1 for equation 5. This can be combined into two constraints as shown in equation 6 and 7.

$$Y_i(X_i^T w) \geq 1 - \epsilon_i \tag{7}$$

For an error to occur, the corresponding ϵ_i must be greater than or equal to 1. So $\sum_i \epsilon_i$ is an upper bound on the number of mistakes that can be done while training phase is happening. We find a natural approach to assigning an error cost. We can change the objective function to be minimized in the equation 8.

$$L_p = \frac{1}{2} * (w^T.w) + C \sum_{i=1}^{n} \epsilon_i - \sum_{i=1}^{n} (\alpha_i (Y_i (X_i^T w) - 1) + \epsilon_i) - \sum_{i=1}^{n} \mu_i \epsilon_i \tag{8}$$

Where L_p is the Lagrangian Primal, C is the parameter chosen by the user for assigning the penalty for the wrong choice of separating vector, α_i and μ_i are the Lagrangian multipliers.

$$L_p = min_{w,b,\epsilon} max_{\alpha} (\frac{1}{2} * (w^T.w) + C \sum_{i=1}^{n} \epsilon_i - \sum_{i=1}^{n} (\alpha_i (Y_i (X_i^T w) - 1) + \epsilon_i) - \sum_{i=1}^{n} \mu_i \epsilon_i) \tag{9}$$

Where b is the threshold that has been applied.
Applying Karush-Kuhn-Tucker conditions it can be stated as below.

$$\frac{\partial L_P}{\partial w} = w_i - \sum_i (\alpha_i Y i X i) = 0 \tag{10}$$

$$\frac{\partial L_P}{\partial b} = - \sum_i (\alpha_i Y i X i) = 0 \tag{11}$$

$$\frac{\partial L_P}{\partial \epsilon} = C - \alpha_i - \mu_i = 0 \tag{12}$$

This implies, W = $\sum \alpha_i Y_i X i$

For the duality problem, we can substitute equations 10 and 11 back to equation 9. We get,

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha j Y_i Y_j X_i^T X_i \tag{13}$$

This can be presented as a dual problem as $\max_{\alpha_i} L_D$
Subject to : $0 \leq \alpha_i \leq C, \sum_{i=1}^{N} \alpha_i Y_i = 0$
Similar to before we can use the KKT conditions to find the threshold B.

1.3 Margin determination

The margin for both the primal and dual problem is given by,

$$\gamma = \frac{1}{w^T w} \tag{14}$$

Using the concept of hinge loss, we can arrive at the soft margin for the dual problem.

$$J = max(0, 1 - Y_i(w.X_i)) \tag{15}$$

Where J is the actual distance from the correct place.

$$w* = argmin_w \frac{1}{C} \sum_i w_i^2 + \sum_i J_i(Y_i, w.X_i + b) \tag{16}$$

Substitute w in equation 14, we get the actual margin for the dual problem. A large margin for a problem specifies the regularization of the weights being utilized in order to prevent overfitting. Hence we prefer choosing the right margin chosen by the cross-validation since it helps us generalize the predictions and helps accurately decide the class for the testing data.

1.4 Why the dual better than primal solution?

Every minimization problem is associated with a maximization problem and vice-versa. The original linear programming problem is known as primal problem, and the derived problem is known as its dual problem. The optimal solutions for the primal and dual problems are equivalent. Conversion of primal to dual has to be done because of many reasons. The dual form of the problem, in many cases, is simple and can be solved with ease.

The kernel trick helps solving the dual problem easier. The Dual optimized problem lends itself to the kernel trick, except that you compute the kernel function instead of the ordinary dot-product. This is not possible in the primal problem since we have to explicitly calculate the mapping of each and every feature in the data set. The concave dual objective gets maximized, which is a convex optimization problem just like the primal. The structure encountered for a quadratic objective subject to a single quadratic constraint. The dual is not only convex, but the duality gap is 0 even if the primal problem is not convex.

1. Any feasible solution to the dual problem gives a bound on the optimal objective function value in the primal problem.
2. Understanding the dual problem leads to specialized algorithms for some important classes of linear programming problems.
3. Sometimes finding an initial feasible solution to the dual is much easier than finding one for the primal. For example, if the primal is a minimization problem, the constraints are often of the form A $\mathbf{x} \geq \mathbf{b}, \mathbf{x} \geq \mathbf{0}, for \mathbf{b} \geq \mathbf{0}$. The dual constraints would then likely be of the form $A^T\mathbf{y} \leq \mathbf{c}, \mathbf{y} \geq \mathbf{0}, for \mathbf{c} \geq \mathbf{0}$. The origin is feasible for the latter problem but not for the former.

4. A primal problem with many constraints and few variables can be converted into a dual problem with few constraints and many variables. Fewer constraints are nice in linear programs because the basis matrix is an n $\times n$ matrix, where n is the number of constraints. Thus the fewer the constraints, the smaller the size of the basis matrix, and thus the fewer computations required in each iteration of the simplex method.

1.5 Code for SVM classification

```
1  import numpy as np
2  import time
3  import pickle
4  import gzip
5
6  def generate_confusion_matrix(y_actual, y_pred):
7      """
8      Generates the confusion matrix
9      :param y_actual: True values of the target
10     :param y_pred: predicted values of the target
11     """
12     print("Python code to generate Confusion Matrix")
13     cm =np.zeros((10,10), dtype=int)
14     for i in range(y_pred.shape[0]):
15         cm[y_actual[i]][y_pred[i]] += 1
16     print(np.asmatrix(cm))
17
18
19 def support_vector_machine(dataset, mnist_test, gamma="
       auto_deprecated"):
20     """
21     Implements the Support Vector Machine Classification algorithm
22     :param dataset: Dataset
23     :param mnist_test: MNIST Test dataset
24     :param usps_data: USPS Feature Set
25     :param usps_test: USPS Target
26     :param gamma: Gamma value
27     """
28     # Fitting classifier to the Training set
29     from sklearn.svm import SVC
30     if gamma == 1.0:
31         classifier = SVC(kernel='rbf', gamma=gamma, random_state
               =0, max_iter=2000, C=1.75)
32     else:
33         classifier = SVC(kernel='linear', random_state=0)
34     classifier.fit(dataset[0], dataset[1])
```

```
35
36      from sklearn.metrics import confusion_matrix, accuracy_score
37      # Testing with MNIST test dataset
38      print("MNIST dataset Test")
39      mnist_pred = classifier.predict(mnist_test[0])
40      cm = confusion_matrix(mnist_test[1], mnist_pred)
41      generate_confusion_matrix(mnist_test[1], mnist_pred)
42      score = accuracy_score(mnist_test[1], mnist_pred)
43      print("SKlearn method to generate Confusion Matrix")
44      print(cm)
45      print("MNIST Accuracy is: {}".format(score))
46
47  filename = 'mnist.pkl.gz'
48  f = gzip.open(filename, 'rb')
49  minst_training_data, minst_validation_data, minst_test_data =
        pickle.load(f, encoding='latin1')
50  f.close()
51  y_mnist_svm = support_vector_machine(minst_training_data,
        minst_test_data)
```

References

1. http://sfb649.wiwi.hu-berlin.de/fedc$_h$*omepage/xplore/tutorials/stfhtmlnode*64.*htmleqn* : *constr*1

2.*https* : *//www.engr.mun.ca/ baxter/Publications/LagrangeForSVMs.pdf*

3.*https* : *//www.quora.com/Why − is − solving − in − the − dual − easier − than − solving − in − the − primal − What − advantages − do − we − get − from − solving − in − the − dual*

4.*https* : *//mikespivey.wordpress.com*/2012/12/04/*dualitylp/*

5.*https* : *//math.stackexchange.com/questions*/2846066/*advantages − of − dual − over − primal − solution − of − an − optimization − problem*