PROBLEM SET 3 REPORT

Introduction to Pattern Recognition- CSE-555

Jayant Solanki

UBIT Name: jayantso
Person Number: 50246821
SPRING 2018 CSE

INTRODUCTION

This report contains the implementation details of Problem Set 2 i.e., training a Support Vector MAchine with using the MNIST training data-set and generation of the overall result and performance. Furthermore we also derived the primal-dual relationship of the 1-norm Soft-Margin classification problem which describes several important concepts in the Machine Learning such as Maximal Margin and Support Vector.

OBJECTIVES ACHIEVED

In the Part 1 of the Problem set 3, we successfully read the MNIST data using the Sklearn library. Then we partitioned the data into train and test sets and ran the gridsearch using Support Vector classifier on the different variation of the regulariser parameter C using 1-norm Soft margin with Linear (dot product) kernel.

In the Part 2 of the problem set 3, we identified the Lagrange Dual Problem of the following given primal problem:

Given features $(x_1, y_1), \cdots, (x_N, y_N)$, where $y_1, \cdots, y_N \in \{-1, 1\}$,

Minimize $w^T \cdot w + C \sum_{i=1}^{N} \xi_i$, the weighted sum between the squared length of the separating vector and the errors, where w is the separating vector, $w^T \cdot w$ is the dot product, and ξ_i is the error made by separating vector w on feature (x_i, y_i).

Furthermore we pointed out what is the “margin” in both primal and dual formulation. Then we mentioned the benefits of maximising the margi, characteristics of support vectors, and at last the benefit of solving the dual problem instead of the primal problem.

IMPLEMENTATION

BACKGROUND

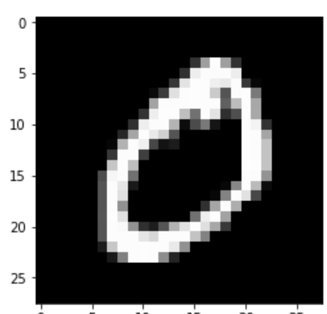
Hard Margin SVM vs. Soft margin SVM[1]

1. Hard margin given by Boser et al. 1992 in COLT and soft margin given by Vapnik et al. 1995.
2. Soft margin is extended version of hard margin SVM.
3. Hard margin SVM can work only when data is completely linearly separable without any errors (noise or outliers). In case of errors either the margin is smaller or hard margin SVM fails. On the other hand soft margin SVM was proposed by Vapnik to solve this problem by introducing slack variables.
4. As for as their usage is concerned since Soft margin is extended version of hard margin SVM so we use Soft margin SVM.

The cost or penalty parameter C is responsible for hard / soft margins in SVM.
Too high C = overfitting possible in the training set.
Too low C = overgeneralization / underfitting possible.
A low C can mean missed opportunity to fit a better model, whereas a high C can fit noise in the training dataset. By cross-validation using grid search, the optimal C is generally chosen.

Part 1:

- **Gathering Data**
  # The digits dataset
  digits = fetch_mldata('MNIST original', data_home="MNIST_Cache")
- **Creating Partitions of Train and test sets after normalising the data pixels:**
  #standardising the data
  digits.data = digits.data/255
  #image we have got are already flattened into 784 pixels, instead of 28x28 pixels
  Train_Images = digits.data[0:60000,:]
  Train_labels = digits.target[0:60000,]
  Test_Images = digits.data[60000:69999,:]
  Test_labels = digits.target[60000:69999,]
- **Visualized data:**
  Label 0



[1] https://www.researchgate.net/post/Can_anyone_explain_to_me_hard_and_soft_margin_Support_Vector_Machine_SVM

- **Creating Model using LinearSVM with C = 1**
  classifier = LinearSVC(C=1)
- **Fitting the Model with the training set**
  classifier.fit(Train_Images, Train_labels)
- **Predicting the labels for the test set**
  predicted = classifier.predict(Test_Images)
  #Output
  Classification report for classifier LinearSVC(C=1, class_weight=None, dual=True, fit_intercept=True,
      intercept_scaling=1, loss='squared_hinge', max_iter=1000,
      multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
      verbose=0):
  0.918091809181
- **From the result section we can find that the best value of C to be used is 1**

Part 2
Solution have been pasted in the form of a picture in the **Result** section.

DIRECTORY LAYOUT

There are 3 folders under **Assignment-3** directory:

- **code :**
  1. Contains the Problem-Statement-3.ipynb file which runs the part 1 of the Problem Set 3
- **Outputs:** Contains Part 2 solution image and some general outputs
- **Report:** Contains the word doc/pdf file of the documentation for Problem Set 3.

INSTALLATION

1. **For the Python code in Jupyter Notebook for Windows 10**

   In the anaconda prompt type:

   conda install matplotlib # for installing the matplotlib

   conda install -c anaconda scikit-learn #for installing scikit library
   All set now and open the Jupyter Notebook

RESULTS

PART 1

Tuning hyper-parameters for precision

Best parameters set found on development set:

{'C': 1}

Grid scores on development set:

0.906 (+/-0.001) for {'C': 1}
0.905 (+/-0.001) for {'C': 2}
0.900 (+/-0.004) for {'C': 5}
0.892 (+/-0.001) for {'C': 10}
0.883 (+/-0.010) for {'C': 20}

Detailed classification report:

The model is trained on the full development set.
The scores are computed on the full evaluation set.

	precision	recall	f1-score	support
0.0	0.95	0.98	0.96	980
1.0	0.96	0.98	0.97	1135
2.0	0.93	0.88	0.91	1032
3.0	0.90	0.91	0.90	1010
4.0	0.92	0.93	0.92	982
5.0	0.89	0.86	0.87	892
6.0	0.93	0.95	0.94	958
7.0	0.92	0.92	0.92	1028
8.0	0.88	0.87	0.87	974
9.0	0.90	0.89	0.89	1008
avg / total	0.92	0.92	0.92	9999

Tuning hyper-parameters for recall

Best parameters set found on development set:

{'C': 1}

Grid scores on development set:

0.906 (+/-0.001) for {'C': 1}

```
0.905 (+/-0.000) for {'C': 2}
0.901 (+/-0.001) for {'C': 5}
0.894 (+/-0.001) for {'C': 10}
0.885 (+/-0.006) for {'C': 20}

Detailed classification report:

The model is trained on the full development set.
The scores are computed on the full evaluation set.

             precision    recall  f1-score   support

        0.0       0.95      0.98      0.96       980
        1.0       0.96      0.98      0.97      1135
        2.0       0.93      0.88      0.91      1032
        3.0       0.90      0.91      0.90      1010
        4.0       0.92      0.93      0.92       982
        5.0       0.89      0.86      0.87       892
        6.0       0.93      0.95      0.94       958
        7.0       0.92      0.92      0.92      1028
        8.0       0.87      0.87      0.87       974
        9.0       0.90      0.89      0.89      1008

avg / total       0.92      0.92      0.92      9999
```

Best value of C is 1

Confusion Matrix for C = 1

```
Confusion matrix:
[[ 962    0    2    1    1    4    5    3    1    1]
 [   0 1112    3    2    0    1    5    1   11    0]
 [  11   11  913   18   10    4   13   12   37    3]
 [   4    0   19  920    2   20    5   12   20    8]
 [   1    4    5    4  913    0    9    3    5   38]
 [   9    2    0   40   12  767   17    7   30    8]
 [   7    4    7    2    5   21  909    1    2    0]
 [   2    8   22    6    7    1    1  947    4   30]
 [  10   13    8   23   14   31    8   13  842   12]
 [   7    8    2   15   31   12    0   26   12  895]]
```

Classification report for classifier LinearSVC(C=1, class_weight=None, dual=True, fit_intercept=True,
intercept_scaling=1, loss='squared_hinge', max_iter=1000,
multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
verbose=0):
0.918091809181

SVC(C=1, cache_size=200, class_weight=None, coef0=0.0,

```
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='linear',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)

Classification report for classifier SVC(C=1, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='linear',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False):
0.940394039404

Confusion matrix:
[[ 957    0    4    1    1    6    9    1    0    1]
 [    0 1122    3    2    0    1    2    1    4    0]
 [    8    6  967   11    3    3    7    8   17    2]
 [    4    3   16  947    1   16    0    9   12    2]
 [    1    1   10    1  942    2    4    2    3   16]
 [   10    4    3   36    6  803   13    1   14    2]
 [    9    2   13    1    5   16  910    1    1    0]
 [    1    8   21   10    8    1    0  957    3   19]
 [    8    4    6   25    7   26    6    7  877    8]
 [    7    7    2   11   33    4    0   18    5  921]]
```

PART 2
It is inside output folder

SOFTWARE/HARDWARE USED

- Anaconda
- Jupyter Notebook
- Python 3.6 Environment based on Anaconda
- Windows 10 System, Intel core i7 processor
- Python libraries: Scikit, MatplotLib

REFERENCES

1. Ublearns
2. Stackoverflow.com
3. Scikit Documentation
4. http://www.cs.tufts.edu/~roni/Teaching/CLT2008S/LN/lecture21-22.pdf
5. https://www.cs.cmu.edu/~tom/10701_sp11/slides/Kernels_SVM2_04_12_2011-ann.pdf
6. http://www.stat.ucdavis.edu/~chohsieh/teaching/ECS289G_Fall2015/lecture5.pdf

7. http://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf
8.