Cover page for answers.pdf
CSE512 Fall 2018 - Machine Learning - Homework 4

Your Name: Sriram Reddy Kalluri

Solar ID: 111878857

NetID email address: skalluri@cs.stonybrook.edu sriram.kalluri@stonybrook.edu

Names of people whom you discussed the homework with:

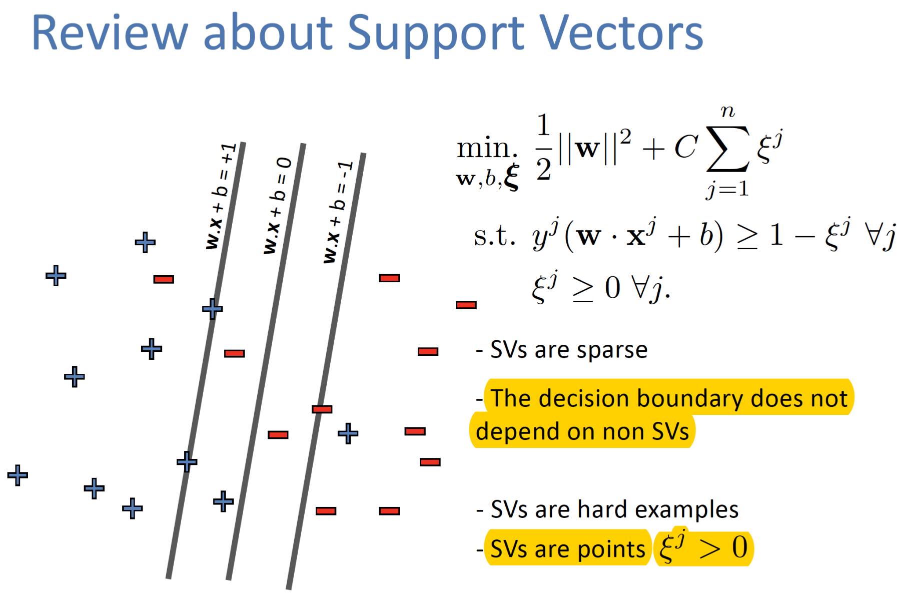
CSE512: Machine Learning HW4

Sriram Reddy Kalluri (skalluri@cs.stonybrook.edu)

September 27 2018

1 Question 1 – Support Vector Machines

1.1 Linear case



Review about Support Vectors

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{j=1}^{n}\xi^j$$

$$\text{s.t. } y^j(\mathbf{w}\cdot\mathbf{x}^j + b) \geq 1 - \xi^j \ \forall j$$

$$\xi^j \geq 0 \ \forall j.$$

- SVs are sparse

- The decision boundary does not depend on non SVs

- SVs are hard examples

- SVs are points $\xi^j > 0$

Figure 1: Decision boundary depends on Support Vector

As discussed in the class, the decision boundary does not depend on non-support vectors. Even if you remove a non-support vector, it is not going to change the decision boundary and the decision hypothesis is only dependent on the support vectors. More-over, it is stated that all the points are linear separable. Now, consider if you are performing the $LOOCV$ error, i.e you are removing the i^{th} example from the data and you are generating the decision boundary, and then you are computing the $LOOCV_i$ for the set. Since the data is linearly seperable, $LOOCV_i$ is only dependent on the i^{th} example error. Now consider two cases: a)if I'm removing a non-support vector, it is not going to add any error as the hypothesis is not changed. b)But in case if I'm removing the support vector, it may rise to error and the error is atmost 1. So over all $LOOCV_i$ error is 0 if i is support vector and $LOOCV_i \leq 1$ if i is support vector. So, the over all $LOOCV$ error can be computed as given below.

$$LOOCV = \frac{1}{n}(\sum_i^n LOOCV_i) \leq \frac{m}{n} \tag{1}$$

So $LOOCV$ error is bounded by m/n.

1.2 General Kernel

As discussed in 1.1, for SVM's one needs to conduct the leave-one-out procedure only for support vectors: non-support vectors will be recognized correctly since removing a point which is not a support vector does not change the decision function. Now the data is linearly separable in the high dimensional feature space corresponding to the kernel, it is totally dependent on the number of the support vectors in the high dimensional feature space. In general, the idea is to gain linearly separation by mapping the data to higher dimensional data for non linear SVM. Since all the examples are projected to the higher dimension, there will be margin to seperate the data points effectively and the number of the support vectors will not be more than m which we have in the Linear Kernel.

So, the over all $LOOCV$ error can be computed as given below.

$$LOOCV = \frac{1}{n}(\sum_i^n LOOCV_i) \leq \frac{|SupportVectors|}{n} \leq \frac{m}{n} \tag{2}$$

LOOCV error bound still holds good for other kernel as the number of support vector is bounded by m.

2 References

[1] http://olivier.chapelle.cc/pub/span_lmc.pdf

2)
Please use q24.m and q26.m files to generate the results.
Number of support vectors is dependent on the precision used. I'm using the threshold value for alpha > 10^-6 to be considered as Support Vector as matlab is generating the values in the range of 10^(-25) to denote 0. The values which I have reported is for the mentioned threshold. Accuracy reported is in the range of (0 to 1). It is not multiplied by 100.

2.4)
C used in this experiment 0.1
Accuracy is 9.073569e-01
Dual Objective function value is 2.476482e+01
Number of support Vectors are 339
Confusion Matrix is

```
  152    32
    2   181
```

2.5)

C used in this experiment 10
Accuracy is 9.782016e-01
Dual Objective function value is 1.121461e+02
Number of support Vectors are 126
Confusion Matrix is

```
  180     4
    4   179
```

2.6)
Initially, I used only linear kernel. Then I used two different kernel, i.e RBF kernel

```
  for i=1:n
  for j=1:n
      rbf=exp(-sum((trD(:,i)-trD(:,j)).^2)*gamma);
      H(i,j) = trLb(i)*trLb(j)*rbf;
  end
  end
```

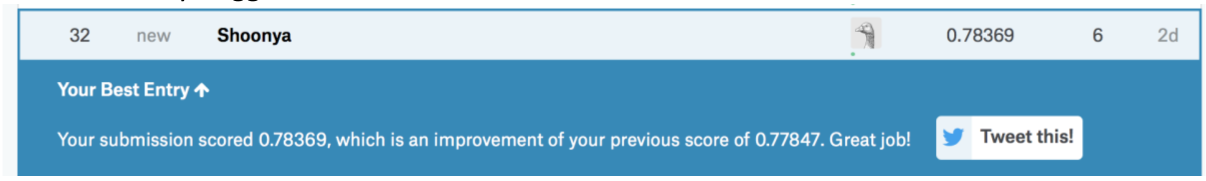
I performed the grid search to find the best values for gamma and C.
Please find the results for gamma and C below:

Gamma on right	0.001	0.01	0.1	1	10	100	1000
C=0.01	0.8033	0.8929	0.8849	0.8849	0.8849	0.8849	0.8849
C=0.1	0.9448	0.9009	0.8679	0.8844	0.8844	0.8844	0.8844
C=1	0.9330	0.8797	0.8679	0.8844	0.8844	0.8844	0.8844
C=10	0.9330	0.9019	0.8844	0.8844	0.8844	0.8844	0.8844
C=100	0.9330	0.9019	0.8844	0.8844	0.8844	0.8844	0.8844
C=1000	0.9330	0.9019	0.8844	0.8844	0.8844	0.8844	0.8844

Then I have used the two methodologies to compute for multi class classification. I have created the programs for one v/s all as well as one v/s one classifiers. Then, I tried different types of normalization for the features and found out that the normalization is improving the results. I have tried adding the extra features by taking the sum and the squares of the features, it is not improving any results. I have calculated the individual class accuracies as well. Since my data set is imbalanced, I checked with one vs one method as well. I have gone through the SMOTE analysis which involves generation of positive examples so that both positive and negative examples are of the same size.

Please use the following files for running:

Please find my Kaggle result below which is same as the baseline.



32 new **Shoonya** 0.78369 6 2d

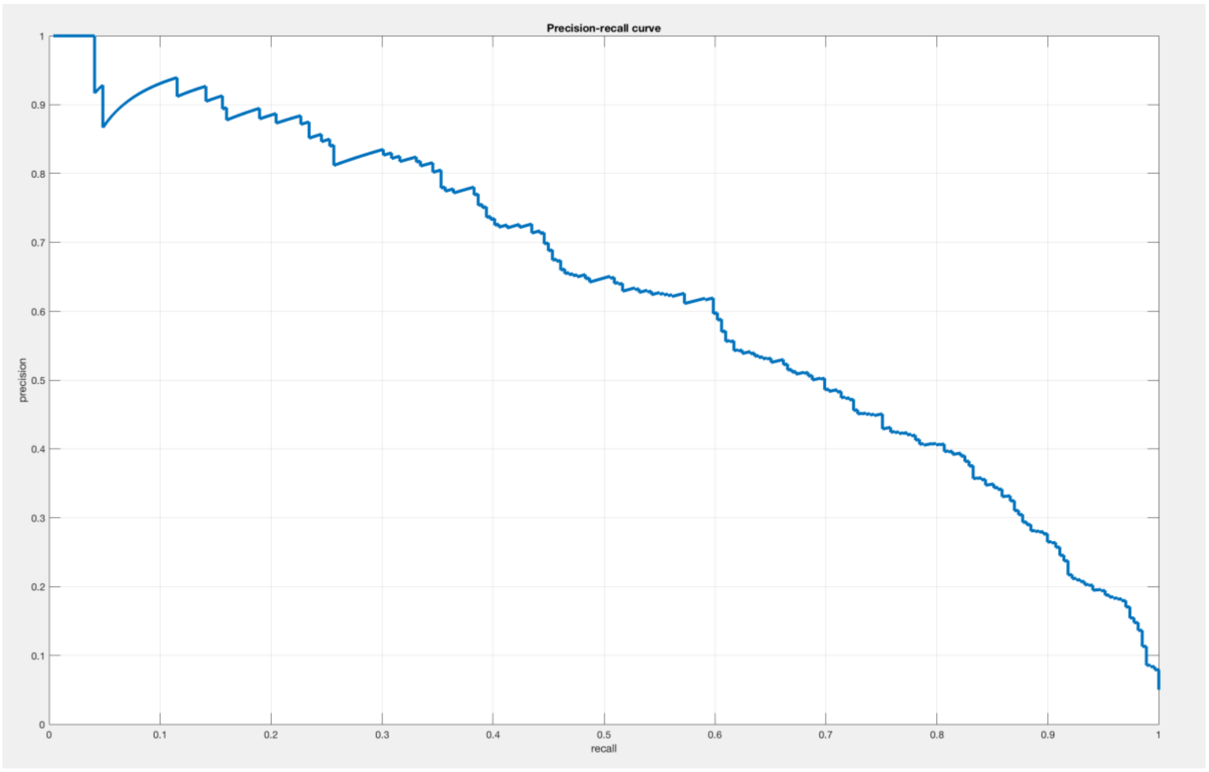
Your Best Entry

Your submission scored 0.78369, which is an improvement of your previous score of 0.77847. Great job!

3.4.1)

Please use the following files to generate the data:
q31.m
SVM_hardmining.m

The Average Precision value which I received is 0.6351.
Please find the Precision Recall curve below:



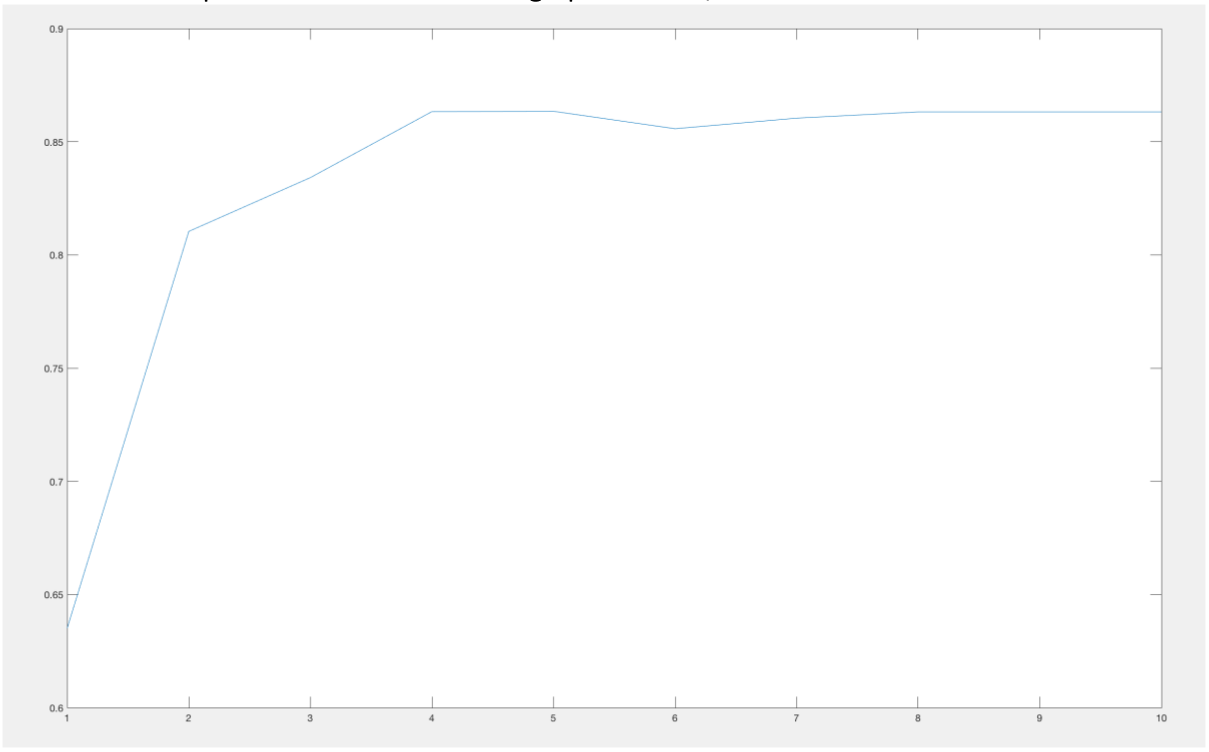
3.4.2)
Justification for the Hardest Negative Examples:
I’m picking the hardest negative examples as the ones with the positive SVM scores from the detect function and then the overlap area as 0.3. These are the hardest negative examples. Moreover, to increase the accuracy, I’m generating the rectangular boxes for negative examples by picking the best 1000 images out of all the negative examples satisfying the above criteria based on the sorted order of SVM scores with respect to the corresponding image and then adding these hardest negative examples to the training test.

3.4.3)
Please use the following files to run the program:
q3_4_3.m
SVM_final10.m
q3_4_3_v1.m

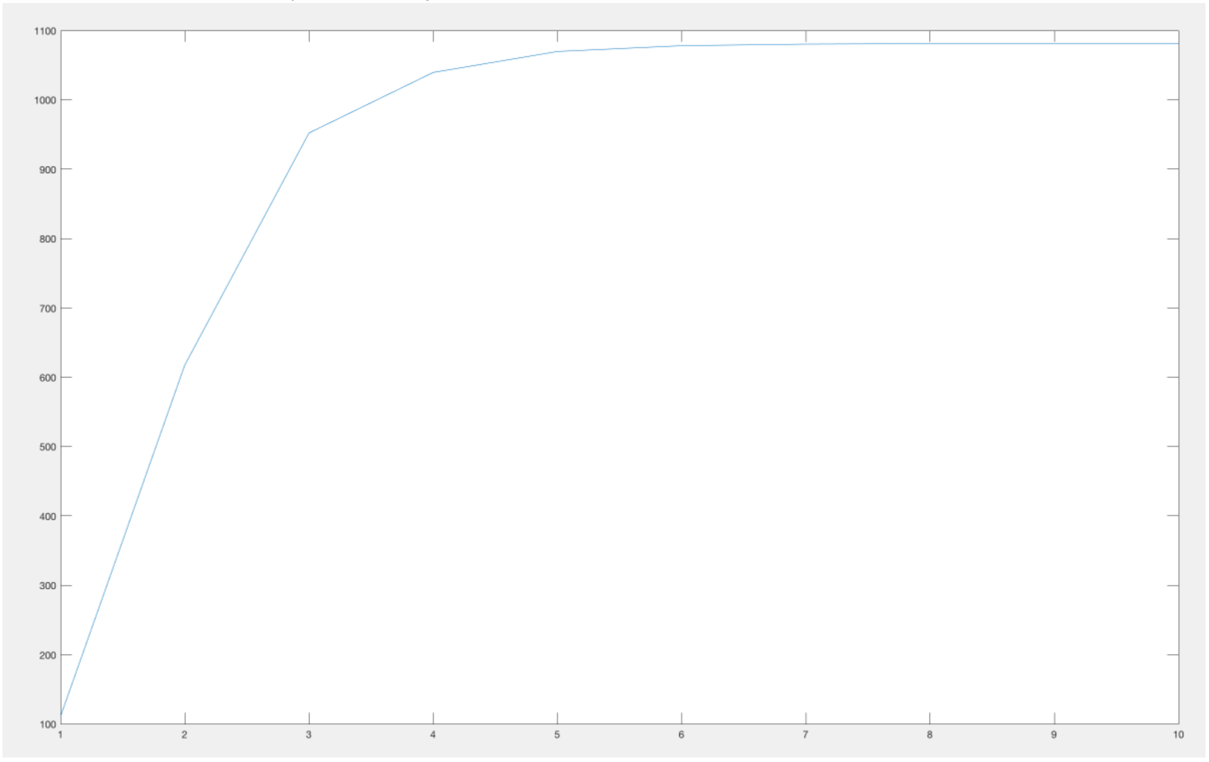
Please find the objective function values and average precision value:

Iteration	Average Precision	Objective function value
1	0.6351	111.2
2	0.8104	617.6
3	0.8342	952.3
4	0.8633	1039.6
5	0.8635	1069.8
6	0.8558	1078.2
7	0.8604	1080.5
8	0.8632	1081.2
9	0.8632	1081.2
10	0.8632	1081.2

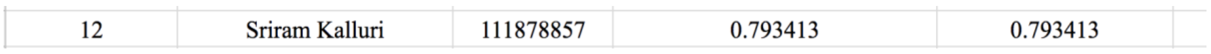
Please find the plots below for the average precision v/s iterations on x axis.



Please find the below plot for objective function v/s iterations on x axis.



3.4.4)
To improve the accuracy, I have tried various things to make the classifier strong by properly choosing the hardest negative examples by increasing the number of examples to 1500 and then increasing the over lap area to 0.33. All these things helped in increasing the average precision value.
Please find the average precision value which is above the base line.



12	Sriram Kalluri	111878857	0.793413	0.793413