ML – CSL341

Nipun Shrivastava 2011cs50288

Assignment 2

1. Spam Classification

a. The SVM dual problem is of the type

$$\begin{aligned} \max_\alpha \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle. \\ \text{s.t.} \quad & \alpha_i \geq 0, \quad i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0, \end{aligned}$$

The SVM dual objective was written in the following form

$$\alpha^T Q \alpha + b^T \alpha + c$$

using the matrix form

```
x1 = bsxfun(@times,x,y);
Q = x1*(x1)';
Q = 0.5*Q;
b = ones(m,1);
% model = runSVM(y,x);
%
% [predict_label, accuracy, dec_values] = prediction(x,y, model, mailCount);

cvx_begin
variable aMatrix(m)
maximize(b'*aMatrix - aMatrix'*Q*aMatrix)
subject to
aMatrix >= 0
aMatrix <= 1
aMatrix'*y == 0
cvx_end
```

The set of support vectors obtained are all those x^i for which $a^i > 0$. They can be easily extracted from the code of first part.

b. The average accuracy obtained after optimising using CVX for linear kernel was 91.3 %.

c. For the case of Gaussian Kernel we used the following code to maximise the Lagrange

```
for i = 1:mailCount
    for j = 1:mailCount
        Q(i,j) = y(i).*y(j).*exp(-((norm(x(i,:)-x(j,:)))^2)*(gamma));
    end
end

b = ones(mailCount,1);

cvx_begin
variable aMatrix(mailCount)
maximize(b'*aMatrix - aMatrix'*(0.5*Q)*aMatrix)
subject to
aMatrix >= 0
aMatrix <= 1
aMatrix'*y == 0
cvx_end
```

With gamma = $2.5*10^{-4}$. The support vectors can be again found out by putting the $a^i > 0$ condition on Lagrange's multiplier. Q matrix was obtained using the kernel definition for Gaussian kernel.
The accuracy obtained for this case was 89.1 %.
As compared to the linear case, the accuracy has fallen most possibly because the data or the model in reality was linearly separable and hence, when we forced a Gaussian kernel on it, our accuracy was brought down.

d. When we trained the SVM using the LibSVM tool, we got the accuracies of 91.3 % and 89.2 % for the case of linear and Gaussian Kernel. These accuracies were in sync with what we got using CVX as the LibSVM internally performs the similar calculations as we did using CVX.
When we trained the LibSVM with the bigger training data, the accuracy jumped to 98.7% for both kernel, which is understandable.

2. Digit Recognition

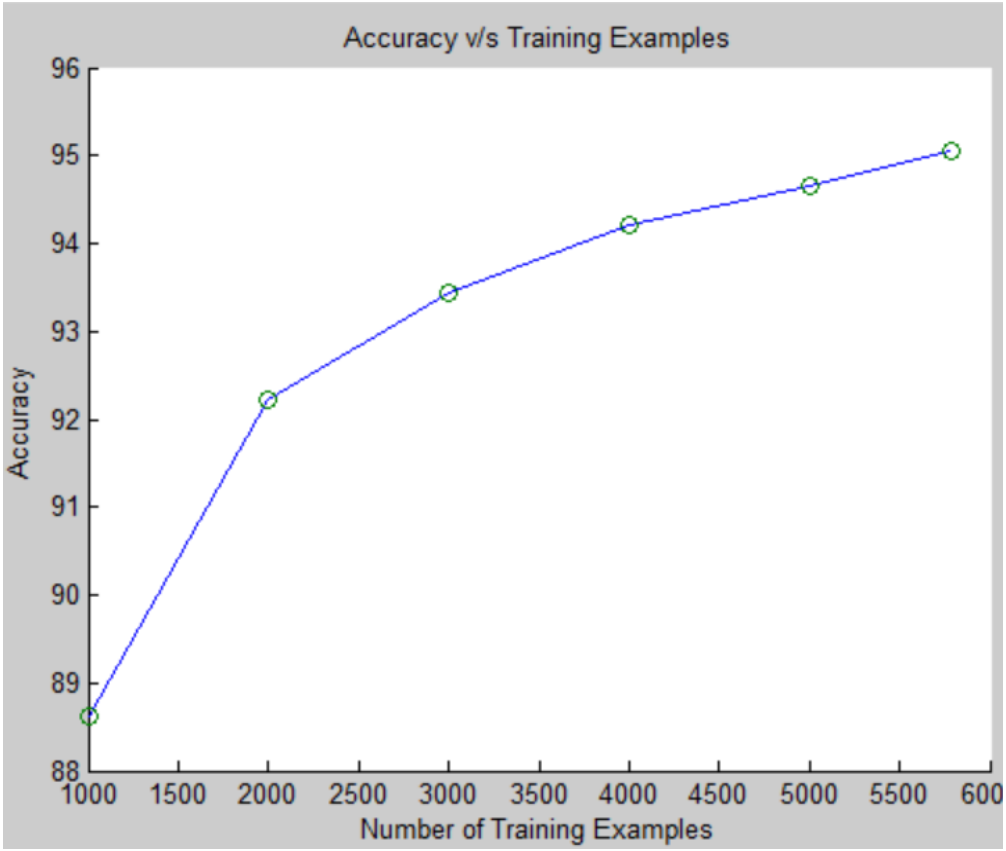
b. The stopping criteria chosen was change in total error < 0.8.
c. The algorithm took approximately 4 minutes to train and the accuracy over the test set came out to be 98.9 %.
d. Since there are 40 possible output, therefore, we need at least 4 binary values to depict them separately. Which is why we need 4 output units in this case. The accuracy over the test case when the neural network was trained for 7 rounds of 60K data came out to be around 92.5%. Error Criterion = Change in avg total error < 0.005
The time taken in this case was obviously more than the previous case. I would say it was around 8-10 times more. The best possible explanation for this is now we have 4 output units instead of 1.

3. Newsgroup Classification

b. By randomly guessing one of the newsgroups as the target class for each of the articles we can get on average an accuracy of 12.5% (since there are 8 newsgroup), but using Naïve Bayes we got an accuracy of 95% which is way more.

c. Yes cross-posting will create a problem for naïve Bayes learner as the confusion between the two classes will increase due to it. Though if there is only a small relative amount of cross-posting then no significant damage will be done.

d. The learning curve maximised around 5000 training examples and seemed to saturate for 6000+ training cases, which means that there is only a little gain for a lot of more computation. We have to decide the optimum size of the training set based on this trade off.



e.

rec.autos	**1**
talk.religion.misc	2
rec.motorcycles	3
talk.politics.mideast	4
talk.politics.misc	5
rec.sport.baseball	6
rec.sport.hockey	7
talk.politics.guns	8

This is the index mapping of various newsgroup. Refer to the confusion Matrix using this. But then we further divided the confusion matrix with the number of examples for that particular news group in order to get a relativistic measure of the confusion.

So from the confusion matrix we find the news group ‘rec.sports.baseball’ with the highest value of the diagonal entry.

5th and 8th newsgroup are the ones which provide maximum confusion, which can be also explained with a superficial analysis as they are similar news group and more often than not have the same words (which occur rarely in other newsgroup) in their corresponding articles. Naïve Bayes did not confuse these newsgroup with Mideast because articles on Mideast have some words which occur exclusively in those newsgroup often. Though (4,5) entry is also pretty high.

confusionMatrix =

861	3	14	2	1	2	2	3
4	626	1	5	5	0	0	5
18	7	947	3	3	1	6	13
2	7	0	927	15	1	0	2
5	18	6	15	742	5	6	21
2	0	1	1	3	977	12	0
0	0	1	3	0	12	966	0
11	31	8	6	62	3	0	827

confusionMatrix <8x8 double>

	1	2	3	4	5	6	7	8
1	0.9535	0.0043	0.0143	0.0021	0.0012	0.0020	0.0020	0.0034
2	0.0044	0.9046	0.0010	0.0052	0.0060	0	0	0.0057
3	0.0199	0.0101	0.9683	0.0031	0.0036	9.9900e-04	0.0060	0.0149
4	0.0022	0.0101	0	0.9636	0.0181	9.9900e-04	0	0.0023
5	0.0055	0.0260	0.0061	0.0156	0.8929	0.0050	0.0060	0.0241
6	0.0022	0	0.0010	0.0010	0.0036	0.9760	0.0121	0
7	0	0	0.0010	0.0031	0	0.0120	0.9738	0
8	0.0122	0.0448	0.0082	0.0062	0.0746	0.0030	0	0.9495