

COL774 – Machine Learning (Assignment-2)

Prakhar Aggarwal

2019CS50441

Part 1 (Text Classification)

(a) Implementing Naive Bayes Algorithm

For this part, we simply computed the θ_{jl} values where " l " $\in \{1, 2, \dots, |V|\}$, V being the vocabulary. We have also performed parameter typing i.e.

$\theta_{jl} = \theta_{j+1} \forall j, j'$ where j denotes the j^{th} feature.

Using θ_{jl} , we compute the conditional probability $P(y|x; \Theta)$ for all classes and take the argmax to predict the label class.

Train Set Accuracy = 72.086 %

Test Set Accuracy = 66.564 %

(b) Random/Majority Prediction

Random Prediction Accuracy over Test Set = 20.17%

Majority Prediction Accuracy over Test Set = 66.09%

(High value of majority prediction is because of large number of examples for class 5)

On comparing the accuracy for random prediction, majority prediction with naive bayes prediction; naive bayes perform slightly better than the majority baseline.

Improvement over Random Prediction: 69.70%

Improvement over Majority Baseline: 0.72%

(c) Confusion Matrix

	Predicted 1	Predicted 2	Predicted 3	Predicted 4	Predicted 5
Actual 1	2	0	0	21	205
Actual 2	0	0	0	63	263
Actual 3	2	0	1	231	852
Actual 4	3	0	0	280	2825
Actual 5	14	0	0	202	9036

The highest value of the Diagonal Entry is corresponding to the 5 star rating i.e. 5 star review predicted as a 5 star review. This can be attributed to the fact that class 5 has the largest number of examples. Also, the accuracy of a 5 star review to be predicted correctly is $\sim 97\%$, so, we can say that, given a 5 star example, probability of it being predicted correctly is quite high.

Other observations:

- Sum over k^{th} row in the confusion matrix denotes number of examples belonging to that class.
- The model performs badly over other classes $\{1, 2, 3, 4\}$ majorly because of very high number of 5 star examples.
- For obtaining $P(y' = k | x, y = k) = (\text{diagonal entry in } k^{\text{th}} \text{ row}) / (\text{sum of } k^{\text{th}} \text{ row})$ i.e. given a k star review, probability of it being predicted correctly by the model is given by the diagonal entry in k^{th} row by sum of k^{th} row.

(d) Stopword removal and Stemming

To remove the stopwords and for performing stemming, I iterated over the reviews, for the i^{th} review, I split it and clean the words obtained.

Cleaning is done as follows:

- Special characters (characters other than A-Za-z0-9 are removed)
- Contiguous '?' are replaced with ' '

After cleaning the data, I have checked for the stopwords and removed them from the data. In last, stemming is performed i.e. obtain the root word.

Train Set Accuracy (stemmed data) = 72.09 %

Test Set Accuracy (stemmed data) = 66.564 %

Thus, here I find that the accuracy for the training set is reduced but the accuracy for the test set seems to be slightly improved! So, such data processing will have a positive influence on the accuracies of the test set since they are removing all the stop words and all which bring around substantial amount of noise in the measurements!

(e) Stopword removal and Stemming

Part 2 (MNIST Digit Classification)

Part a (Binary Classification)

For this part, I used the CVXOPT Package to solve the convex constrained optimization problem (SVM problem).

SVM's dual objective is given by:

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

$W(\alpha)$ is represented in the form $(1/2)\alpha^T P \alpha + q^T \alpha$.

SVM's optimization problem changes to: (replace x with α)

$$\begin{aligned} &\text{minimize} \quad (1/2)x^T P x + q^T x \\ &\text{subject to} \quad Gx \leq h \\ &\quad \quad \quad Ax = b \end{aligned}$$

with parameters:

P: $M * M$ array, where $P[i,j] = y^{(i)} y^{(j)} \langle x^{(i)}, x^{(j)} \rangle$

q: $M * 1$ array of -1

G: $2M * M$ array (the condition, $0 \leq \alpha_i \leq C$ is visualized as $(-\alpha_i) \leq 0$ and $\alpha_i \leq C$)
First half is a diagonal matrix with all entries -1
Second half is a diagonal matrix with all entries 1

h: $2M * 1$ array, first M entries are 0, remaining M entries are C .

A: $1 * M$ array, y^T

b: 0

Since my entry number ends in 1, I performed the binary classification problem over class 1 vs class 2.

After formulating the SVM problem, we can solve it using CVXOPT package and obtain the α_i values. Using a certain threshold value ($1e-5$), we can obtain the support vectors (values that determine the hyperplane).

For the Linear Kernel, we can easily compute w as feature vector is finite. 'b' can also be computed using the expression derived in class.

$$w = \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)} \quad b = -\frac{\max_{i: y^{(i)} = -1} w^{*T} x^{(i)} + \min_{i: y^{(i)} = 1} w^{*T} x^{(i)}}{2}$$

Formula for prediction: $w^T x + b$

For the Gaussian kernel, we cannot compute w because of feature vector being infinite dimensional. Instead, we will use $w^T \phi(x)$ directly to predict y as well as to compute b .

$$w = \sum \alpha_i y^{(i)} \phi(x^{(i)})$$

$$\text{So, } w^T \phi(x) = \sum \alpha_i y^{(i)} \phi(x^{(i)})^T \phi(x)$$

$\Rightarrow w^T \phi(x)$ can be computed using inner product over $\langle \phi(x), \phi(x^{(i)}) \rangle$ which further can be expressed as $K(x, x^{(i)})$

For the gaussian case, kernel value is given by:

$$K(x, z) = \exp \left(-\frac{\|x - z\|^2}{2\sigma^2} \right)$$

(i) SVM problem using a Linear Kernel (CVXOPT Package)

No. of Support Vectors:

weight matrix (w) \rightarrow (784,1) matrix

intercept term (b) \rightarrow

Test set accuracy:

Validation set accuracy:

(ii) SVM problem using a Gaussian Kernel (CVXOPT Package)

No. of Support Vectors:

intercept term (b) \rightarrow

Test set accuracy:

Validation set accuracy:

(ii) SVM problem using Linear/Gaussian Kernel (LIBSVM Package)

Linear:

	LIBSVM	CVXOPT
Test set accuracy		
Validation set accuracy		
Number of Support Vectors		
Intercept Term		
Running Time		

Gaussian:

	LIBSVM	CVXOPT
Test set accuracy		
Validation set accuracy		
Number of Support Vectors		
Intercept Term		
Running Time		

So we see, the accuracy of LIBSVM and our CVXOPT implementation is similar, but the training + prediction time is significantly better in LIBSVM.

This is due to a highly optimised and parallelised code running in LIBSVM

Part b (Multi Class Classification)

For this part, we will extend the SVM formulation for a binary classification problem. We will train a model on each pair of classes.

This method is also said to be one vs one classifier.

(i) Multi-Class SVM using Gaussian kernel (CVXOPT package)

Test set accuracy:

Training set accuracy: (Tried running model 5 times for more than 4 hrs but laptop hanged)

Running time for test set prediction:

Running time for validation set prediction:

Confusion Matrix for the test set is given by:

	Predicted 0	Predicted 1	Predicted 2	Predicted 3	Predicted 4	Predicted 5	Predicted 6	Predicted 7	Predicted 8	Predicted 9
Actual 0										
Actual 1										
Actual 2										
Actual 3										
Actual 4										
Actual 5										
Actual 6										
Actual 7										
Actual 8										
Actual 9										

(ii) Multi-Class SVM using Gaussian kernel (LIBSVM package)

In this part, one vs one classifier is implemented using LIBSVM package

Test set accuracy : 97.23%

Validation set accuracy : 99.92%

Running time (test set + training set prediction) : 760 seconds

Confusion matrix for the test set is:

	Predicted 0	Predicted 1	Predicted 2	Predicted 3	Predicted 4	Predicted 5	Predicted 6	Predicted 7	Predicted 8	Predicted 9
Actual 0	969	0	1	0	0	3	4	1	2	0
Actual 1	0	1121	3	2	1	2	2	0	3	1
Actual 2	4	0	1000	4	2	0	1	6	15	0
Actual 3	0	0	8	985	0	4	0	6	5	2
Actual 4	0	0	4	0	962	0	6	0	2	8
Actual 5	2	0	3	6	1	866	7	1	5	1
Actual 6	6	3	0	0	4	4	939	0	2	0
Actual 7	1	4	19	2	4	0	0	987	2	9
Actual 8	4	0	3	10	1	5	3	3	942	3
Actual 9	4	4	3	8	13	4	0	9	12	952

Confusion matrix for the validation set is:

	Predicted 0	Predicted 1	Predicted 2	Predicted 3	Predicted 4	Predicted 5	Predicted 6	Predicted 7	Predicted 8	Predicted 9
Actual 0	2000	0	0	0	0	0	0	0	0	0
Actual 1	0	1997	1	0	1	0	0	1	0	0
Actual 2	0	0	2000	0	0	0	0	0	0	0
Actual 3	0	0	0	1999	0	0	0	1	0	0
Actual 4	0	0	0	0	1999	0	0	0	0	1
Actual 5	0	0	0	0	0	2000	0	0	0	0
Actual 6	0	0	0	0	1	0	1999	0	0	0
Actual 7	0	2	1	0	1	0	0	1995	0	1
Actual 8	0	1	0	0	0	0	0	0	1999	1
Actual 9	0	0	0	0	2	0	0	2	0	1996

(iii) Comparison between Mutli-Class CVXOPT vs Multi-Class LIBSVM

So we see, the accuracy of LIBSVM and our CVXOPT implementation is similar, but the training + prediction time is significantly better in LIBSVM.

This is due to a highly optimised and parallelised code running in LIBSVM and SMO algorithm which works extremely faster than the CVXOPT matrix solver.




On observing the confusion matrix for the test set, we can see that examples for digit 9 (~5.6%) are the ones that are most missclassified into digit 4 and digit 8.

Whereas, digit 0 and digit 1 are the ones that are most correctly classified (~98.87%, ~98.76%).

Misclassified digits: 2 and 7, 2 and 8, 4 and 9, 8 and 9

Results makes sense as these digits look similar to each other and such errors are also very easy to be made by the human eye by just observing images.

Some examples of miss-classified digits:

- 1.)  Correct label: 2, Predicted label: 7 (Example: 322 in test set (indexing from 1))
- 2.)  Correct label: 9, Predicted label: 8 (Example: 242)
- 3.)  Correct label: 5, Predicted label: 3 (Example: 341)

(iv) K-fold cross validation

For this part, we had to divide the training data into 5 parts and then run the 5-fold cross validation.

Validation accuracies obtained for different values of C:

$C = 10^{-5}$ ->

$C = 10^{-3}$ ->

$C = 1$ ->

$C = 5$ ->

$C = 10$ ->

Test set accuracies obtained for different values of C:

$C = 10^{-5}$ ->

$C = 10^{-3}$ ->

$C = 1$ ->

$C = 5$ ->

$C = 10$ ->

We can see that for the lower values of C (10^{-5} and 10^{-3}), both, validation set accuracy and test set accuracy are quite low. This can be understood from the expression:

$$(1/2)\mathbf{w}^T\mathbf{w} + C * \sum_i \xi_i$$

For lower values of C, we allow noise to be high thus affecting the accuracies. Also, we can see that for higher values of C (1, 5 and 10) the model works fine with high accuracies over validation as well as test set.

C = __ gives the maximum accuracy for the cross validation

C = __ gives the maximum accuracy over the test set