# COL774: Assignment 2

Sayam Sethi

October, 2021

## Contents



## 1 Text Classification

All macro f1 scores are on test data.

### 1.1 Naïve Bayes

We used the multinomial classifier model (bag of words model). The observations made are:

1. Training accuracy is 72.086%
2. Test accuracy is 65.24285714285715%
3. The macro f1 score is 0.22283194818240615

### 1.2 Random and Mode Models

#### 1.2.1 Random Model

The probability of predicting each class is $\frac{1}{5}$ since there are 5 classes. Also, each example is independent. Therefore, the expected accuracy is $\frac{m \times \frac{1}{5}}{m}$ which is 20%. The accuracy obtained is 20.214285714285715% which is consistent with our calculations. The f1 score is 0.13801833427316684

#### 1.2.2 Mode Model

The accuracy obtained in this case is 66.08571428571428% which is even more than the Naïve Bayes model above. This is primarily because the test data is heavily skewed towards samples with rating 5. However the f1 score is very low at 0.15916050232238088

### 1.3 Confusion Matrix

The confusion matrix obtained is

$$\begin{pmatrix} 4 & 1 & 9 & 39 & 175 \\ 1 & 1 & 7 & 102 & 215 \\ 3 & 0 & 17 & 381 & 685 \\ 16 & 1 & 19 & 626 & 2446 \\ 81 & 16 & 69 & 600 & 8486 \end{pmatrix} \tag{1}$$

### 1.4 Cleaning and Stemming

The results on cleaning and stemming the `review text` are:

1. Training accuracy is 67.914%
2. Test accuracy is 65.31428571428571%
3. Macro f1 score is 0.30528097408076293

We observe that the training accuracy decreases and the testing accuracy very slightly improves.

### 1.5 Feature Engineering

#### 1.5.1 Bigram

1. The training accuracy is 85.608%
2. The test accuracy is 65.51428571428571%

3. The macro f1 score is 0.23173573788598736

4. The training accuracy greatly improves however there is a very slight increase in the accuracy and macro f1 score on test data.

#### 1.5.2 Trigram

1. The training accuracy shoots up to 99.596%

2. The test accuracy drops down to 43.164285714285716%

3. The macro f1 score increases to 0.25257681089990963

4. This model is a clear example of overfitting even though the macro f1 score is slightly better.

#### 1.5.3 Ignoring Smoothing for Unseen Words

In the default Naïve Bayes model, we perform Laplace smoothing for the unseen words. However, as a feature, we decide to completely ignore those words if encountered in the test data. The results are as follows:

1. Training accuracy obtained is 72.086%

2. Test accuracy improves 66.56428571428571%

3. However, the macro f1 score drops to 0.1933817935603243

#### 1.5.4 Bigram and Cleaning + Stemming

1. The training accuracy is very high at 93.32%

2. The test accuracy is at 63.85714285714286%

3. The macro f1 score is 0.27156312403441796

#### 1.5.5 Bigram and Ignoring

1. The training accuracy is 85.608%

2. The test accuracy is 66.51428571428571%

3. The macro f1 score is very low at 0.1809373323371783

#### 1.5.6 Analysis of Different Features (and accuracy vs f1 score)

The following observations have been made:

1. We observe that using bigram alone gives better accuracy than either of the default and cleaned + stemmed model.

2. However, the accuracy of bigram drops on using it along with cleaned + stemmed model, and with the "ignoring" model. The f1 score also decreases with respect to the maximum of individual features (cleaning + stemming and bigram respectively).

3. Trigram has a very low accuracy but a higher f1 score than the default model.

4. We claim that f1 score is more suited for this model since the data is highly skewed towards 5 rating examples. This is clearly visible from similar accuracy of the model which just predicts the class which occurs the most in the training set. However, there is significant variation in the f1 scores for each model and it makes sense to choose the model which reports the best f1 score.

5. However completely relying on f1 score might be a bad idea. Consider the trigram model. The training accuracy is above 99% and the test accuracy is below 50%. Therefore, the model doesn't do well at predicting the rating and predicts correctly at a rate of about 1 in 2. Thus, even though it has a higher macro f1 score, the model might perform poorly in practice. However this also shows that the model is better at predicting the less representative classes (classes with lower ratings).

### 1.6 Using the Summary Field

Summary field has been incorporated as weighted sum of the frequencies from `review text` and `summary` fields. Cleaning + stemming was done before computation. The accuracy obtained is 65.93571428571429% and the f1 score is 0.35236362262920623. This model performs the best compared to all other models since the accuracy is also relatively higher and the f1 score is the highest of all models.

## 2 MNIST Digit Classification

### 2.1 Binary Classification

My entry number ends with 9 and therefore the classes considered are 9 (label 1) and 0 (label $-1$) in the Support Vector Machine.

#### 2.1.1 Linear Kernel

To train using the CVXOPT package, we need to first transform the dual problem into the form

$$\begin{aligned} &\alpha^T P \alpha + q^T \alpha + d \\ &G\alpha \preceq H \\ &A\alpha = b \end{aligned} \tag{2}$$

The dual in summation format is given as:

$$\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y^i y^j (x^i)^T x^j - \sum_{i=1}^{m}\alpha_i \tag{3}$$

It is easy to see that $P_{ij}$ is the coefficient of $\alpha_i\alpha_j$. Therefore, $P_{ij}$ is given as:

$$\begin{aligned} P_{ij} &= y^i y^j (x^i)^T x^j \\ \implies P &= X_y \times X_y^T \\ \text{where } X_y &= \text{each row of } X \text{ multiplied by } Y \end{aligned} \tag{4}$$

Also, $d$ is 0 and $q$ is a vector with all $-1$:

$$\begin{pmatrix} -1 \\ -1 \\ \vdots \\ -1 \end{pmatrix} \tag{5}$$

The condition on $\alpha_i$ is $0 \leq \alpha_i \leq c$. Therefore $G$ and $H$ are given as:

$$\begin{aligned} G &= \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \\ -1 & 0 & \dots & 0 \\ 0 & -1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & -1 \end{pmatrix} \\ H &= \begin{pmatrix} c \\ c \\ \vdots \\ c \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \end{aligned} \tag{6}$$

The equality condition is $\sum_{i=1}^{m} \alpha_i y^i = 0$. Therefore $A$ and $b$ are given as:

$$\begin{aligned} A &= \begin{pmatrix} y_1 & y_2 & \dots & y_m \end{pmatrix} \\ b &= 0 \end{aligned} \tag{7}$$

We now compute these values and pass them to the CVXOPT quadratic program solver. The results obtained are:

$$\begin{aligned} \text{training time} &= 26.060819149017334s \\ nSV &= 126 \\ b &= 0.29323351252380514 \\ \text{training accuracy} &= 100\% \\ \text{test accuracy} &= 98.99446958270488\% \end{aligned} \tag{8}$$

### 2.1.2 Gaussian Kernel

The dual SVM problem is given as:

$$\frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y^i y^j \phi(x^i)^T \phi(x^j) - \sum_{i=1}^{m} \alpha_i \tag{9}$$

The only difference will be in the value of $P$, and $P$ is given as:

$$\begin{aligned} P_{ij} &= y^i y^j \phi(x^i)^T \phi(x)^j \\ \implies P_{ij} &= y^i y^j \exp(-\gamma ||x^i - x^j||^2) \\ \implies P_{ij} &= y^i y^j \exp(-\gamma (||x^i||^2 + ||x^j||^2 - 2x^i x^j)) \end{aligned} \tag{10}$$

We generalise this equation for computing the *product* of any two vectors $X, Z$ of sizes $n, m$ respectively as:

$$\mathcal{P}(X,Z) = \begin{pmatrix} ||X_1||^2 & ||X_1||^2 & (m \text{ times}) \dots & ||X_1||^2 \\ ||X_2||^2 & ||X_2||^2 & (m \text{ times}) \dots & ||X_2||^2 \\ \vdots & \vdots & \ddots & \vdots \\ ||X_n||^2 & ||X_n||^2 & (m \text{ times}) \dots & ||X_n||^2 \end{pmatrix} + \begin{pmatrix} ||Z_1||^2 & ||Z_2||^2 & \dots & ||Z_m||^2 \\ ||Z_1||^2 & ||Z_2||^2 & \dots & ||Z_m||^2 \\ \vdots & \vdots & & \vdots \\ n \text{ times} & n \text{ times} & \ddots & n \text{ times} \\ \vdots & \vdots & & \vdots \\ ||Z_1||^2 & ||Z_2||^2 & \dots & ||Z_m||^2 \end{pmatrix} - 2(X \otimes Z) \tag{11}$$

Here $\otimes$ is outer product. We can now compute $P$ as:

$$P = (Y \otimes Y) \circ \exp(-\gamma \mathcal{P}(X, X)) \tag{12}$$

$\circ$ is Hadamard product. The values are then again passed to the CVXOPT quadratic problem solver. We make predictions as:

$$(\alpha \circ Y) \circ \exp(-\gamma \mathcal{P}(X_{SV}, X_{data})) + b \tag{13}$$

The results obtained are:

$$\begin{aligned} \text{training time} &= 18.87721347808838 \\ nSV &= 1217 \\ b &= -0.2547934897245705 \\ \text{training accuracy} &= 100\% \\ \text{test accuracy} &= 99.14529914529915\% \end{aligned} \tag{14}$$

### 2.1.3 LIBSVM

**Linear Kernel** The results are:

$$\begin{aligned} \text{training time} &= 1.163637638092041 \\ nSV &= 126 \\ b &= 0.29308328093984604 \\ \text{training accuracy} &= 100\% \\ \text{test accuracy} &= 98.99446958270488\% \end{aligned} \tag{15}$$

**Gaussian Kernel** The results are:

$$\begin{aligned} \text{training time} &= 4.993328332901001 \\ nSV &= 1214 \\ b &= -0.0992945392255848 \\ \text{training accuracy} &= 100\% \\ \text{test accuracy} &= 99.14529914529915\% \end{aligned} \tag{16}$$

## 2.2 Multi-Class Classification

### 2.2.1 One-for-One Classifier using CVXOPT

The results are:

$$\begin{aligned} \text{training time} &= 843.7367413043976 \\ \text{prediction text} &= 14.868017673492432 \\ \text{test accuracy} &= 97.27\% \end{aligned} \tag{17}$$

### 2.2.2 Classification using LIBSVM

The results are:

$$\begin{aligned} \text{training time} &= 163.81674242019653 \\ \text{prediction text} &= 87.36125683784485 \\ \text{test accuracy} &= 97.23\% \end{aligned} \tag{18}$$

The training time is much lesser than using CVXOPT (takes $\frac{1}{5}$ the time), however the prediction time is larger with the model that is trained by LIBSVM (takes $6\times$ time).

### 2.2.3 Confusion Matrix

The confusion matrix using CVXOPT is:

$$\begin{pmatrix} 969 & 0 & 1 & 0 & 0 & 3 & 4 & 1 & 2 & 0 \\ 0 & 1121 & 3 & 2 & 1 & 2 & 2 & 0 & 3 & 1 \\ 4 & 0 & 1000 & 4 & 2 & 0 & 1 & 6 & 15 & 0 \\ 0 & 0 & 8 & 985 & 0 & 4 & 0 & 6 & 5 & 2 \\ 0 & 0 & 4 & 0 & 962 & 0 & 6 & 0 & 2 & 8 \\ 2 & 0 & 3 & 6 & 1 & 866 & 7 & 1 & 5 & 1 \\ 6 & 3 & 0 & 0 & 4 & 4 & 939 & 0 & 2 & 0 \\ 1 & 4 & 19 & 2 & 4 & 0 & 0 & 987 & 2 & 9 \\ 4 & 0 & 3 & 10 & 1 & 5 & 1 & 3 & 944 & 3 \\ 5 & 4 & 3 & 8 & 13 & 3 & 0 & 7 & 12 & 954 \end{pmatrix} \tag{19}$$

The confusion matrix using LIBSVM is:

$$\begin{pmatrix} 969 & 0 & 1 & 0 & 0 & 3 & 4 & 1 & 2 & 0 \\ 0 & 1121 & 3 & 2 & 1 & 2 & 2 & 0 & 3 & 1 \\ 4 & 0 & 1000 & 4 & 2 & 0 & 1 & 6 & 15 & 0 \\ 0 & 0 & 8 & 985 & 0 & 4 & 0 & 6 & 5 & 2 \\ 0 & 0 & 4 & 0 & 962 & 0 & 6 & 0 & 2 & 8 \\ 2 & 0 & 3 & 6 & 1 & 866 & 7 & 1 & 5 & 1 \\ 6 & 3 & 0 & 0 & 4 & 4 & 939 & 0 & 2 & 0 \\ 1 & 4 & 19 & 2 & 4 & 0 & 0 & 987 & 2 & 9 \\ 4 & 0 & 3 & 10 & 1 & 5 & 3 & 3 & 942 & 3 \\ 4 & 4 & 3 & 8 & 13 & 4 & 0 & 9 & 12 & 952 \end{pmatrix} \tag{20}$$

The misclassified data is submitted in the output directory of Q2. The reason for misclassification is because of the different styles of writing the number which leads to *confusion* in identifying which digit the image represents.

### 2.2.4 K-Fold Cross Validation

The average of K-Fold Cross Validation for different values of $c$ are:

$$\begin{aligned} c = 10^{-5} &: 9.525 \\ c = 0.001 &: 9.525 \\ c = 1 &: 97.45 \\ c = 5 &: 97.6 \\ c = 10 &: 97.6 \end{aligned} \tag{21}$$

The training accuracies are:

$$\begin{aligned} c = 10^{-5} &: 72.1 \\ c = 0.001 &: 72.1 \\ c = 1 &: 97.23 \\ c = 5 &: 97.29 \\ c = 10 &: 97.29 \end{aligned} \tag{22}$$

The accuracy for small values of $c$ are highly skewed between cross validation and testing accuracy. This is an example of underfitting. For larger values of $c$ there is overfitting since the model performs relatively much better on the validation data compared to the testing data.