COL774 Assignment 2

Soham Gaikwad, 2018CS10394

November 2020

1. **Text Classification**

   (a) Implemented Naive Bayes by first constructing a vocabulary V over training data. The input data x takes values in $\{1, 2, ..., |V|\}$; parameterized by θ and the class labels y takes values in $\{1, 2, ..., 5\}$; parameterized by ϕ. After calculating ϕ and θ, prediction was done by taking argmax over all class probabilities. Train set accuracy was 64.64% and Test set accuracy was 60.52%

   (b) Random prediction accuracy was 11.21% and Majority prediction accuracy 0.43.99% with class 5 as the majority class. As can be seen, the implementation of Naive Bayes gives much better accuracy.

   (c) Confusion matrix is given below predictions on rows and actual data on columns. Class 5 has the largest value of the diagonal entry which is also the majority class which means that most number of correct predictions belong to class 5. It can be observed that for every class, the correct predicted class (e.g 4 star review predicted as 4 star) has the highest entry and the classes adjacent to it (e.g. 4 star review predicted as 3 or 5 star) has a lower value and so on. This tells us that the model gets "confused" between nearby stars which is somewhat expected.

	1	**2**	**3**	**4**	**5**
1	14751	3018	1461	1115	3046
2	3173	2758	1316	490	215
3	1204	3154	4468	1820	387
4	626	1470	6276	18523	14741
5	415	438	1010	7410	40433

Table 1: Confusion matrix for Naive Bayes

   (d) Used the given PorterStemmer for stemming along with some modifications to remove punctuation etc. and achieved Test set accuracy of 60.17% which is very close to previous accuracy. Stemming will provide better results in general but there was a slight drop in accuracy for this particular test set. Also, stemming helps to reduce the number of features and hence computational time.

   (e) Feature engineering

   - Bi-gram: Used two consecutive words as feature. Test set accuracy 63.29%.
   - Tri-gram: Used three consecutive words as feature. Test set accuracy 55.99%.
   - Incorporating Inverse document frequency: Transformed counts of words to have a factor of inverse document frequency (IDF). Test set accuracy 59.02%.
     $IDF(word) = log(\#documents/1 + \#documents\ in\ which\ word\ appears)$

2. **Fashion MNIST Article Classification**

(a) **Binary classification**: My entry number is 2018CS10394, so d = 4. Binary classification was done between class 4 (coats) and class 5 (sandals).

i. Since CVXOPT solves a problem of minimizing, the dual problem of $max_\alpha W(\alpha)$ was converted to $min_\alpha - W(\alpha)$. So P matrix was defined as $P_{ij} = y_i * y_j * dot(x_i, x_j)$ and q was m sized vector of -1. Matrices G, h, A, b were also constructed as required by CVXOPT.
From the SVM training, 71 support vectors were found by checking for $\alpha_i > \epsilon = 10^{-3}$. The validation and test set accuracy were 99.6% and 99.8% respectively.

ii. In this case $dot(x_i, x_j)$ was replaced by $K(x_i, x_j)$ i.e. the Gaussian kernel in the definition of P. The validation and test set accuracy were 99.6% and 99.6% respectively, similar to linear kernel in the previous case.

(b) **Multi-Class classification**

i. As computing on the original dataset was taking a huge time, I used a subset (around a fifth) of the data to train. This was tested on 400 test vectors (again due to computation cost) and an accuracy of 80-85% was achieved.

ii. I used sklearn's `svm.SVC` model for this problem. The validation and test set accuracy were 87.92% and 88.08% respectively. The training time was about 3-4 minutes as sklearn does optimizations like multi-processing/multi-threading.
Note: I had set the `max_iter` parameter to 4000 in the model which limits the maximum iterations in SVC. Without this, the model was taking a large time to train.

iii. Confusion matrix for parts 2.b.i (my SVM implementation) and 2.b.ii (sklearn SVM).

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	31	0	0	2	0	0	4	0	0	0
1	0	41	0	0	0	0	0	0	0	0
2	1	0	27	0	2	0	3	0	0	0
3	1	3	1	38	2	0	2	0	0	0
4	0	0	7	0	25	0	4	0	0	0
5	0	0	0	0	0	37	0	1	1	3
6	11	0	6	3	1	0	22	0	1	0
7	0	0	0	0	0	3	0	29	0	0
8	2	0	2	1	0	1	0	0	36	1
9	0	0	0	0	0	1	0	3	0	40

Table 2: Confusion matrix with my SVM implementation (2.b.i)

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	433	1	5	12	3	0	80	0	1	0
1	0	482	0	0	1	0	0	0	0	0
2	5	4	411	3	41	0	55	0	1	0
3	11	9	7	457	13	0	9	0	1	0
4	3	0	37	9	399	0	34	0	2	0
5	0	0	0	0	0	473	0	14	2	11
6	38	4	32	14	38	0	315	0	2	0
7	0	0	0	0	0	16	0	471	2	14
8	10	0	8	5	5	5	7	1	489	1
9	0	0	0	0	0	6	0	14	0	474

Table 3: Confusion matrix with sklearn's svm.SVC (2.b.ii)

As can be seen class 0 (T-Shirt/top) was "confused" the most with class 6 (Shirts) which seems fairly natural and makes sense as T-shirts and shirts are quite similar.

iv. A graph for 5-fold cross validation for C is shown in Figure 1 below. This was done using a subset of the original data (around one-fifth) due to computational costs.
It can be observed that C=5 gives the best validation as well as training accuracy though it is very similar to C=1 and C=10.

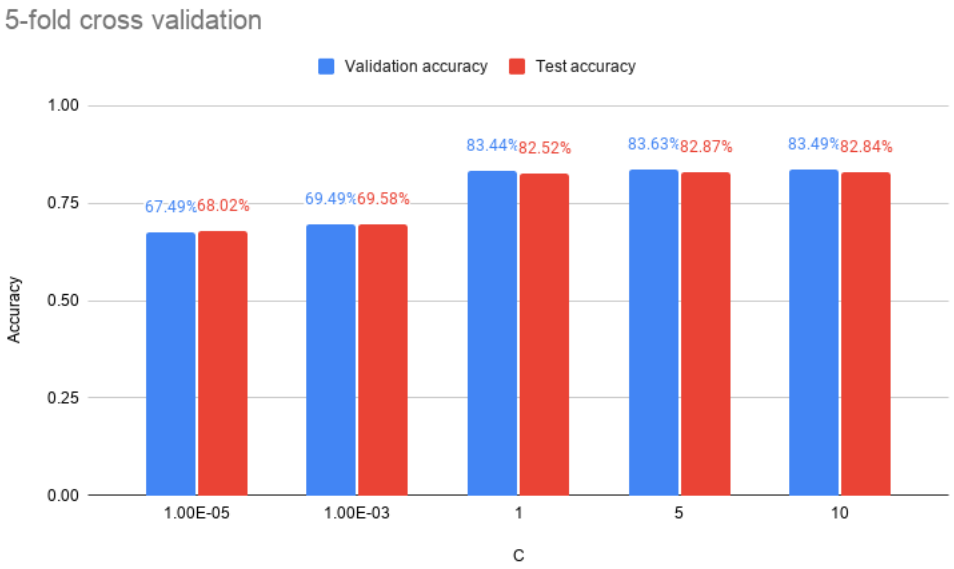

Figure 1: 5-fold cross validation for C

3. **Large scale text classification**

   For text preprocessing, I used the CountVectorizer to create the Bag of words and used TfidfTransformer on top of it to transform counts to tf-idf (similar to part 1.e). Optimal parameters for each model were found using sklearn's GridSearchCV over a validation set (subset of the training set).

   (a) i. **Naive Bayes**: Used the MultinomialNB from sklearn and achieved an accuracy of 57.7% with a training time of around 2-3 minutes.

   ii. **SVM (with LIBLINEAR)**: Used the LinearSVC from sklearn and achieved an accuracy of 69.03% with a training time of around 5-6 minutes. Optimal parameters found for this model were C=1 and using bigrams.
   SVM (with LIBLINEAR) took comparatively large time but also achieved significantly better accuracy.

   (b) **SVM (with SGD algorithm)**: Used the SGDClassifier from sklearn and achieved an accuracy of 63.34% with a training time of around 3-4 minutes. Optimal parameters found for this model were alpha=0.0001 and using bigrams.
   In comparision with LIBLINEAR, the SGD algorithm took less time to train but also achieved a lower accuracy.