COL 774: Assignment 2 (Semester I, 2023-24)

Due Date: 11:50 pm, Wednesday Oct 4, 2023. Total Points: 64

Notes:

- This assignment has two main parts - Text Classification using Naïve Bayes (Part 1), and Image Classification using SVMs (Part II).
- You should submit all your code (including any pre-processing scripts written by you) and any graphs that you might plot.
- Do not submit the datasets. Do not submit any code that we will be providing to you for processing.
- Include a single write-up (pdf) file which includes a brief description for each question explaining what you did. Include any observations and/or plots required by the question in this single write-up file.
- You should use Python for all your programming solutions.
- Your code should have appropriate documentation for readability.
- You will be graded based on what you have submitted as well as your ability to explain your code.
- Refer to the course website for assignment submission instructions.
- This assignment is supposed to be done individually. You should carry out all the implementation by yourself.
- We plan to run Moss on the submissions. We will also include submissions from previous years since some of the questions may be repeated. Any cheating will result in a zero on the assignment, a penalty of -10 points and possibly much stricter penalties (including a fail grade and/or a DISCO).

1. **(30 points) Text Classification**
   In this problem, we will use the Naïve Bayes algorithm for text classification. The dataset for this problem is the Coronavirus tweets Dataset. Given a tweet related to coronavirus, the task is to predict the sentiment of the review. There are three possible sentiments (labels) - positive, neutral or negative. You have been provided with a subset of the Coronavirus tweets Dataset, with the training and validation splits containing 37,864 reviews (samples) and 32,93 reviews respectively. We will be testing your code on a held-out test split. Data is available at this link.

(a) (10 points) Implement the Naïve Bayes Multiclass classification algorithm to classify each sample into one of the given three categories. You should implement the model where for each word position in a document, we generate the word using a single (fixed across word positions) Multinoulli distribution.

i. Report the accuracy over the training as well as the validation set.

ii. Read about word cloud. Construct a word cloud representing the most frequent words for each class.

Notes:

- Make sure to use the Laplace smoothing for Naïve Bayes (as discussed in class) to avoid any zero probabilities.
- You should implement your algorithm using logarithms to avoid underflow issues.
- You should implement Naïve Bayes from the first principles and not use any existing Python modules.

(b) (2 points)

i. What is the validation set accuracy that you would obtain by randomly guessing one of the categories as the target class for each of the reviews (random prediction)?

ii. What accuracy would you obtain if you simply predicted each sample as positive?

iii. How much improvement does your algorithm give over the random/positive baseline?

(c) (2 points) Read about the confusion matrix.

i. Draw the confusion matrix for your results in parts (a) and (b) above (for both training and validation data).

ii. For each confusion matrix, which category has the highest value of the diagonal entry? What does that mean?

(d) (4 points) The dataset provided to you is in the raw format i.e., it has all the words appearing in the original set of articles. This includes words such as 'of', 'the', 'and' etc. (called stopwords). Presumably, these words may not be relevant for classification. In fact, their presence can sometimes hurt the performance of the classifier by introducing noise in the data. Similarly, the raw data treats different forms of the same word separately, e.g., 'eating' and 'eat' would be treated as separate words. Merging such variations into a single word is called stemming. Read about stopword removal and stemming (for text classification) online.

i. Perform stemming and remove the stop-words in the training as well as the validation data.

ii. Construct word clouds for both classes on the transformed data.

iii. Learn a new model on the transformed data. Report the validation set accuracy.

iv. How does your accuracy change over the validation set? Comment on your observations.

(e) (6 points) Feature engineering is an essential component of Machine Learning. It refers to the process of manipulating existing features/constructing new features in order to help improve the overall accuracy of the prediction task. For example, instead of using each word as a feature, you may treat bi-grams (two consecutive words) as a feature.

i. You can read here about Bi-grams. Use word-based bi-grams to construct new features. You should construct these features after doing the pre-processing described

in part d(i) above. Further, these should be added as additional features, on top of existing unigram (single word) based features. Learn your model again, and report validation set accuracy.

ii. Come up with at least one additional set of features to further enhance your model. Learn the model after doing pre-processing as described in part d(i) above, and report the validation set accuracy.

iii. For both your variations tried above, compare with the validation set accuracy that you obtained in parts (a) and parts (d). Do the additional set of features constructed in each case help you improve the overall accuracy? Comment on your observations.

(f) (6 points) In Domain Adaptation, given a model trained on data from the *source domain*, we apply it on data from the *target domain*. The idea is to exploit the similarity between source and target domains, and leverage the parameters learned on the target domain, making up for lack of sufficient training data in the target domain. In our setting, we will assume the source domain is coronavirus tweets, and the target domain is represented by another dataset of general purpose tweets. You are given subsets of different sizes of the training data (in the target domain), varying from $1\%, 2\%, 5\%, 10\%, 25\%, 50\%, 100\%$ of the total training set.

i. Learn a Naïve bayes model, on entire source training data, combined with the partial target training data for each of the splits above. Your vocabulary should be constructed on the combined dataset. You should use the model in part (d) above, i.e., after removing stopwords, and performing stemming but before doing any additional feature engineering. Compute (target) validation set accuracy for each split.

ii. Do the same thing as in the part above, but without adding any data from source domain. This represents learning from scratch on the target data. Compute the accuracy on target validation set for each of the splits.

iii. Plot on a single graph the validation set accuracy for the two algorithms described above, as we vary the size of target training data.

iv. What do you observe? Explain.

2. **Image Classification** In this problem, we will use Support Vector Machines (SVMs) to build a binary image classifier. We will be solving the SVM optimization problem using a general-purpose convex optimization package as well as using a scikit-learn library function based on a customized solver known as LIBSVM. This problem is based on Intel image classification dataset. The original dataset has 25000 images belonging to 6 classes. However, we will be using a subset of this, containing 2380 training samples and 200 validation samples each for 6 classes. We will be testing your code on held-out test set. Data is available at this link. There is one folder each for images in one of the 6 classes. Each sample is an RGB image of size $150 \times 150 \times 3$. Resize the images to size $16 \times 16 \times 3$ and normalize the data by dividing each pixel value by 255. For working with SVM's, you should flatten each image to create a vector of length 768.

(20 points) Binary Classification: Let d be the last digit of your entry number. Take the subset of images for the classes (d mod 6) and (($d+1$) mod 6) from the train data provided to you and perform the following experiments in the context of binary classification.

(a) (8 points) Download and install the CVXOPT package. Express the SVM dual problem (with a linear kernel) in a form that the CVXOPT package can take. You will have to think about how to express the SVM dual objective in the form $\alpha^T P\alpha + q^T\alpha + c$ matrix

where P is an $m \times m$ matrix (m being the number of training examples), q is an m-sized column vector and c is a constant. For your optimization problem, remember to use the constraints on α_i 's in the dual. Use the SVM formulation which can handle noise and use $C = 1.0$ (i.e. C in the expression $\frac{1}{2}w^T w + C * \sum_i \xi_i$). You can refer this link to get a working overview of cvxopt module and it's formulation.

i. How many support vectors do you get in this case? What percentage of training samples constitute the support vectors?

ii. Calculate the weight vector w and the intercept term b and classify each of the examples in the validation file into one of the two labels. Report the validation set accuracy. You will need to carefully think about how to represent w and b in this case.

iii. Reshape the support vectors corresponding to the top- 6 coefficients to get images of $16 \times 16 \times 3$ and plot these. Similarly, reshape and plot the weight vector w.

(b) (6 points) Again use the CVXOPT package to solve the dual SVM problem using a Gaussian kernel. Think about how the P matrix will be represented. Use $C = 1.0$ and $\gamma = 0.001$ (i.e. γ in $K(x, z) = \exp^{-\gamma * \|x-z\|^2}$) for this part.

i. How many support vectors do you get in this case as compared to the linear case above? How many support vectors obtained here match with the linear case above?

ii. Note that you may not be able to explicitly store the weight vector (w) or the intercept term (b) in this case. Use your learned model to classify the validation examples and report the validation accuracy.

iii. Reshape the support vectors corresponding to the top- 6 coefficients to get images of $16 \times 16 \times 3$ and plot these.

iv. Compare the validation accuracy obtained here with part (a).

(c) (6 points) Repeat parts -(a) & (b) with the scikit-learn SVM function, which is based on the LIBSVM package.

i. Compare the nSV (Number of Support Vectors) obtained here with part (a) for linear kernel and part (b) for Gaussian kernel. How many of the support vectors obtained here match with the support vectors obtained in both these cases?

ii. Compare weight (w), bias (b) obtained here with part (a) for linear kernel.

iii. Report the validation accuracy for both linear and Gaussian kernel.

iv. Compare the computational cost (training time) of the CVXOPT with the sklearn implementation in both the linear and Gaussian case.

(d) **(Extra fun! No Credits)** Resize the original $150 \times 150 \times 3$ sized image to size $32 \times 32 \times 3$ and normalize the data by dividing each pixel value by 255. Then flatten each image to create a vector of length 3072. Repeat part (c) on this and compare the results with that of part (c). What do you observe?

(14 points) Multi-Class Image Classification: In this section, we will work with the entire subset of the data provided to you in this question focusing on a multi-class classification problem using SVMs. We will work with Gaussian kernel for this section.

(a) (4 points) In class, we described the SVM formulation for a binary classification problem. In order to extend this to the multi-class setting, we train a model on each pair of classes to get $\begin{pmatrix} k \\ 2 \end{pmatrix}$ classifiers, k being the number of classes (here, k = 5). During prediction

time, we output the class which has the maximum number of votes from all the $\binom{k}{2}$ classifiers. You can read more about one-vs-one classifier setting at the following link. Using your CVXOPT solver from the previous section, implement one-vs-one multi-class SVM. Use a Gaussian Kernel with $C = 1.0$ and $\gamma = 0.001$.

i. Classify the validation examples and report validation set accuracy. In the case of ties, choose the label with the highest score.

(b) (3 points) Now train a multi-class SVM on this dataset using the scikit-learn SVM function, which is based on the LIBSVM package. Repeat part (a) using a Gaussian kernel with $\gamma = 0.001$. Use $C = 1.0$ as earlier.

i. Classify the validation examples and report validation set accuracy.

ii. How do the validation set accuracy and the training time obtained here compare with part (a) above?

(c) (4 points) Draw the confusion matrix for both of the above parts (2(a) and 2(b)). What do you observe? Which classes are miss-classified into which ones most often? Visualize (and report) 12 examples of miss-classified objects. Do the results make sense? Comment.

(d) (3 points) Validation set is typically used to estimate the best value of the model hyper-parameters (e.g., C in our problem with the Gaussian kernel) by randomly selecting a small subset of the training data as the validation set and then training on the modified training data (original training data minus the validation set) and making predictions on the validation set. For a detailed introduction, you can refer to this video. You can check the correctness of your intuition by trying this test. K-fold cross-validation is another such technique in this regard that we use in practice. In this technique, we divide our training data into K-folds or parts and then treat each part as our validation set once and train on the remaining K-1 parts. You can read more about cross-validation here [1] (see Section 1) for more details. This process is repeated for a range of model hyper-parameter values and the parameters which give best K-fold cross-validation accuracy are reported as the best hyper-parameters. We will use scikit-learn SVM function for this part.

For this problem, we will do a 5-fold cross-validation to estimate the best value of the C parameter for the Gaussian kernel case. Validation data should not be touched.

i. Fix γ as 0.001 and vary the value of C in the set $\{10^{-5}, 10^{-3}, 1, 5, 10\}$ and compute the 5 -fold cross-validation accuracy and the validation accuracy for each value of C.

ii. Now, plot both the 5-fold cross-validation accuracy as well as the validation set accuracy on a graph as you vary the value of C on x-axis (you may use log scale on x-axis). What do you observe? Which value of C gives the best 5 -fold cross-validation accuracy? Does this value of the C also give the best validation set accuracy? Comment on your observations.

[1] These are from Andrew Ng notes posted on the course website, and the link is available only from the internal IIT Delhi network. Feel free to read additional material online about this topic.