COL 774: Assignment 2. (Semester I, 2020-21)

Due Date: 11:50 pm, Wednesday Oct 6, 2021. Total Points: 66

Notes:

- This assignment has three parts - Text Classification using Naïve Bayes (Part A) and MNIST Classification using SVM (Part B). Comparison of Naïve Bayes and SVM on large scale text classification (Part C). All parts will be due at the same time.
- 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 have provided 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.
- You should also submit run files required for auto-grading. Auto-grading instructions will be released separately.
- 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 Amazon Digital Music review dataset and has been obtained from this website. Given a users review, task is to predict the 'overall' rating given by the reviewer. Read the website for more details about the dataset. You have been provided with separate training and test files containing 50,000 reviews (samples) and 14,000 reviews respectively. Data is available at this link. A review comes from one of the five categories (class label). Here, class label represents 'overall' rating given by the user along with the 'reviewText'.

   (a) **(10 points)** Implement the Naïve Bayes algorithm to classify each of the 'reviewText' into one of the given categories. Report the accuracy over the training as well as the test set. In the remaining parts below, we will only worry about test accuracy.

   Notes:

   - Make sure to use the Laplace smoothing for Naïve Bayes (as discussed in class) to avoid any zero probabilities. Use $\alpha = 1$, where α is the parameter which controls the strength of the smoothing.
   - 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)** What is the test set accuracy that you would obtain by randomly guessing one of the categories as the target class for each of the review (random prediction). What accuracy would you obtain if you simply predicted the class which occurs most of the times in the training data (majority prediction)? How much improvement does your algorithm give over the random/majority baseline?

(c) **(3 points)** Read about the confusion matrix. Draw the confusion matrix for your results in the part (a) above (for the test data only). Which category has the highest value of the diagonal entry? What does that mean? What other observations can you draw from the confusion matrix? Include the confusion matrix in your submission and explain your observations.

(d) **(4 points)** The dataset provided to 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.
- Perform stemming and remove the stop-words in the training as well as the test data.
- Learn a new model on the transformed data. Again, report the accuracy.
- How does your accuracy change over test set? Comment on your observations.

(e) **(5 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 on the prediction task. For example, instead of using each word as a feature, you may treat bi-grams (two consecutive words) as a feature. Come up with at least two alternative features and learn a new model based on those features. Add them on top of your model obtained in part (d) above. Compare with the test set accuracy that you obtained in parts (a) and parts (d). Which features help you improve the overall accuracy? Comment on your observations.

(f) **(2 points)** Read about another performance metric referred to as F1-score. For your best performing model obtained above, report the F1-score for each class in the test set. Also report the average of these numbers referred to as macro F1-score. Which metric, test error or macro-F1 score, do you think is more suited for this kind of dataset? Why?

(g) **(4 points)** Till now we have been using 'reviewText' to predict 'overall' rating, however the dataset also contains a 'summary' field. Can the prediction accuracy be improved if you make use of summarized review? Experiment and report your findings. Feel free to experiment with innovative ways of incorporating 'summary' field in your classification model.

2. **(36 points) MNIST Digit Classification**
In this problem, we will use Support Vector Machines (SVMs) to build a handwritten digit classifier. We will be solving the SVM optimization problem using a general purpose convex optimization package as well as using a customized solver known as LIBSVM. You are provided with separate training and test example files. Each row in the (train/test) data file corresponds to an image of size 28x28, represented as a vector of grayscale pixel intensities followed by the label associated with the image. Every column represents a feature where the feature value denotes the gray-scale value (0-255) of the corresponding pixel in the image. There is a feature for every pixel in the image. Last column gives the corresponding label. The entire dataset (along with description) is available at this webpage. We will work with a subset of the MNIST data for the purpose of this assignment. This subset can be downloaded from this link. Since the features represent grayscale values (on the same metric), we may not want to normalize the data to zero mean and unit variance as described in the class. But for this problem, you may find it helpful to simply scale all the values to the range [0,1] (down from [0 255]).

(a) **Binary Classification:**
Let d be the last digit of your entry number. Then, we would start with binary classification problem over the images of digits (d vs (d+ 1) mod 10) in this Section. In particular, you should take the subset of images for the digits d and (d + 1) mod 10) from the train/test data provided to you and perform the following experiments.

i. **(8 points)** Download and install the CVXOPT package. Express the SVM dual problem (with a linear kernel) in the 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. Obtain the set of support vectors from your optimization for the binary classification problem. Furthermore, calculate the weight vector w and the intercept term b and classify each of the examples in the test file into one of the two labels. Report the validation and test set accuracy obtained. You will need to carefully think about how to represent w and b in this case.

ii. **(6 points)** Use CVXOPT package to solve the dual SVM problem using a Gaussian kernel. Think about how the P matrix will be represented. What are the set of support vectors in this case? 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 and test examples and report the accuracies obtained. Use $C = 1.0$ and $\gamma = 0.05$ (i.e. γ in $K(x, z) = \exp^{-\gamma * \|x-z\|^2}$) for this part. How do these compare with the ones obtained with in the linear kernel?

iii. **(6 points)** Repeat part-a & b with LIBSVM package available for download from this link. Report accuracy on test set for both linear and Gaussian kernel. Furthermore, compare weight (w), bias (b) and nSV (# of Support Vectors) with your implementation in part (a) for linear kernel and part (b) for Gaussian kernel. Also compare the computational cost (training time) of the two implementations.

(b) **Multi-Class Classification:**

In this section, we will work with the entire subset of the data provided to you focusing on a multi-class classification problem. We will work with a Gaussian kernel for this section.

i. **(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 $\binom{k}{2}$ classifiers, k being the number of classes. 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 solver from previous section, implement one-vs-one multi-class SVM. Use a Gaussian Kernel with $C = 1.0$ and $\gamma = 0.05$. Classify given dataset and report test set accuracy. In case of ties, choose the label with the highest score.

ii. **(4 points)** Now train a multi-class SVM on this dataset using the LIBSVM library . Repeat part (a) using a a Gaussian kernel with $\gamma = 0.05$. Use $C = 1.0$ as earlier. Report the test set accuracies. How do your results compare with those obtained in part (a) above. As earlier, compare the computational cost (training time) of the two implementations? Comment.

iii. **(4 points)** Draw the confusion matrix as done in the first Part (Naive Bayes) of this assignment for both of the above parts (2(a) and 2(b)). What do you observe? Which digits are miss-classified into which ones most often? Visualize (and report) 10 examples of mis-classified digits. Do the results make sense? Comment.

iv. **(4 points)** Validation set is typically used to estimate the best value of the model parameters (e.g., C in our problem with linear kernel) by randomly selecting small subset of data as validation set, training on train set (minus the validation set) and making predictions on 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 techniques 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 parameter values and the parameters which give best K-fold cross validation accuracy are reported as the best parameters. We will use LIBSVM 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. Test data should not be touched. Fix γ as 0.05 and vary the value of C in the set $\{10^{-5}, 10^{-3}, 1, 5, 10\}$ and compute the 5-fold cross validation accuracy for each value of C. Also, compute the corresponding accuracy on the test set. Now, plot both the 5-fold cross validation accuracy as well as the test 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 test 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