COL 774: Assignment 2

Due Date: 11:50 pm, March 12 (Tuesday), 2019. Total Points: 33+

Notes:

- This assignment has two parts - Text Classification using Naïve Bayes and Handwritten digit classification using SVM.
- 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/MATLAB 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. **(33 points) Text Classification**
   In this problem, we will use the Naïve Bayes algorithm for text classification. The dataset for this problem is a subset of the Yelp dataset and has been obtained from this website. Given a users review, task is to predict the stars given by the reviewer. Read the website for more details about the dataset. You have been provided with separate training and test files containing 534K reviews (samples) and 133K reviews respectively. This data can be downloaded from the course website. A review comes from one of the five categories (class label). Here, class label represents stars given by the user along with the review. Please refer to *README* in data directory for more details.

   (a) **(10 points)** Implement the Naïve Bayes algorithm to classify each of the articles into one of the given categories. Report the accuracy over the training as well as the test set.

   Notes:

   - Make sure to use the Laplace smoothing for Naïve Bayes (as discussed in class) to avoid any zero probabilities. Use $c = 1$.
   - 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 Matlab/Python modules.

   In the remaining parts below, we will only worry about test accuracy.

(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.
- Use the script provided with the data to you to perform stemming and remove the stop-words in the training as well as the test data. You are free to use other tools as well.
- 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) **(3 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) **(6 points)** You are also provided with full publicly available version of the Yelp dataset containing 5M training instances and 1M test instances. Train your model on this version of the dataset. Use the best performing model obtained in part (e) above. Note that this may take a while to train. Report your test set accuracy as well as Macro F1 score on the original test set (used in parts (a)- (f) above). What do you observe? You are free to run your model on the full test set (1M instances) though you do not have to report these numbers.

(h) **Extra Fun: No points** Till now we have been only using user reviews to predict stars, however the dataset also contains other fields such as number of votes the review received. Can the prediction accuracy be improved if you make use of other available fields? Experiment and report your findings.

Note: Full version of the data is only to be used for part (g), you should use subset of the data for parts (a)-(f).

2. Second question will be on SVMs will be posted later.

COL 774: Assignment 2 - Part B

Due Date: 11:50 pm, March 13 (Wednesday), 2019. Total Points: 36

Notes:

- This is the second part of Assignment 2 - Handwritten digit classification using SVM.
- 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/MATLAB 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**).

(36 points) MNIST Handwritten 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 grayscale 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 avaliable 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]).

1. **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.

   (a) **(8 points)** Download and install the CVXOPT package. Express the SVM dual problem (with a linear kernel) in the a form that the CVX package can take. You will have to think about how to express the SVM dual objective in the form $\alpha^T Q\alpha + b^T\alpha + c$ matrix where Q is an $m \times m$ matrix (m being the number of training examples), b 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 $\frac{1}{2}w^T w + C * \sum_i \xi_i$). Report the set of support vectors obtained 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 average test set accuracy obtained. You will need to carefully think about how to represent w and b in this case.

(b) **(6 points)** Use CVX package to solve the dual SVM problem using a Gaussian kernel. Think about how the Q 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 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?

(c) **(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.

2. **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.

(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 $\binom{k}{2}$ classifiers, k being the number of classes. During prediction time, we output the classifier which has the maximum number of wins among 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 MNIST dataset and report train and test accuracy. In case of ties, choose the label with the highest score.

(b) **(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 train as well as 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.

(c) **(4 points)** Draw the confusion matrix as done in the first Part (Naive Bayes) of this assignment. What do you observe? Which digits are mis-classified into which ones most often? Do the results make sense?

(d) **(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 and making predictions on validation set. This process is repeated for a range of model parameter values and the parameters which give best validation accuracy are reported as the best parameters. For a detailed introduction, you can refer to this video. You can check the correctness of your intuition by trying this test. We will use LIBSVM for this part.

For this problem, use a randomly sampled 10% of training data as your validation set to estimate the 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 validation accuracy for each value of C. Also, compute the corresponding accuracy on the test set. Now, plot both the validation set 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 validation accuracy? Does this value of the C also give the best test set accuracy? Comment on your observations.

3. **Extra Fun - no credits!**

(a) Use mini-batch version of Pegasos algorithm described in
"Pegasos: Primal Estimated sub-GrAdient SOlver for SVM" to optimize SVM problem and solve for w, b. You should start with Algorithm 1 given in the paper. Further, Algorithm 1 ignores the intercept term b. To incorporate b, use the description provided in Section 6 of the paper (use Equation 23). Your final implementation should compute the values of both w and b. Note that the paper uses a slightly different notation from the one covered in class and you should map the symbols accordingly.

(b) Several other approaches have been proposed to solve SVM optimization problem such as *LIBLINEAR* focussing on speed for linear kernel. Compare training time of LIBSVM, LIBLINEAR and Pegasos the given dataset. Which one will you choose and why? Would your answer change in case of a larger dataset like Yelp reviews from Q1?

(c) Multi-class problem can also be tackled using one-vs-rest strategy. Compare one-vs-rest and one-vs-one in terms of training time and accuracy. You can read more about Multiclass SVMs in Sec. 7.1.3 of PRML book [by Christopher Bishop] or on the the link given earlier.