CS464 Introduction to Machine Learning
Fall 2020
Homework 1

Due: November 08, 2020, 5:00 PM

Instructions

- Submit a soft copy of your homework of all questions to the related assignment section on Moodle. Submitting a hard copy or scanned files is NOT allowed. You have to prepare your homework digitally (using Word, Excel, Latex etc.).
- This is an individual assignment for each student. That is, you are NOT allowed to share your work with your classmates. Your homeworks will be checked against plagiarism and violation of the Bilkent University Code of Academic Integrity.
- **You may ask questions regarding this homework to your TA, Oğuzhan Karakahya (o.karakahya@bilkent.edu.tr).**
- For this homework, you may code in any programming language you would prefer. In submitting the homework file, please package your file as a gzipped TAR file or a ZIP file with the name `CS464_HW1_Section#_Firstname_Lastname`.

  As an example, if your name is Sheldon Cooper and you are from Section 1 for instance, then you should submit a file with name `CS464_HW1_1_sheldon_cooper`. Do NOT use Turkish letters in your package name.

  Your compressed file should include the following:

  - report.pdf : The report file where you have written your calculations, plots, discussions and other related work for the first and fourth questions.
  - main.*: The main code file of your work. It should be in a format easy to run and must include a main script serving as an entry point. The extension of this file depends on the programming language of your choice. For instance, if you are using Python, your code file should end with ".py" extension. If you are using a notebook editor, do not forget to save your file as a Python file at the end. If you are using MATLAB, your file should end with extension ".m". For other programming languages, your file should have the extension of the main executable file format for that language. Your code must generate requested files while running. DO NOT add them to your compressed file.
  - README.txt : You must also provide us with a README file that tells us how we can execute/call your program. README file should include which parameters are the default values, what is the terminal command to execute your file and how to read the outputs.
- You are NOT allowed to use any machine learning packages, libraries or toolboxes for this assignment (such as scikit-learn, tensorflow, keras, theano, MATLAB Statistics and Machine Learning Toolbox functions, e1071, nnet, kernlab etc.).
- Your codes will be evaluated in terms of efficiency as well. Make sure you do not have unnecessary loops and obvious inefficient calculations in your code.

- Create the requested files in the same directory as your source code. Do not provide absolute paths when creating requested files.
- We follow no extension policy. Any assignments turned in late will be penalized and will incur a reduction of 25% in the final score, for each day (or part thereof) it is late.
- If you do not follow the submission routes, deadlines and specifications (codes, report, etc.), it will lead to significant grade deduction.

1 Probability Questions [15 pts]

Question 1.1 [5 pts] A basketball team has 8 matches left for the remainder of the season. The probability of winning a match is 0.6 and it is independent for each match. A turnover is defined as losing after a win or winning after a lose. What is the probability of having exactly 1 turnover for the remainder of the season?

Question 1.2 [10 pts] In tennis, after 40-40, a player wins when s/he scores 2 successive points. As an example, assume that the score is 40-40. Then, player A scores, so s/he is in advantage. At this time, if player A scores once more, he wins. Otherwise, if player B scores, the score is equal once more and 2 successive points are required again.

The probability of scoring a point is p for player A and it is independent of the current score of the game. What is the probability of player A winning the game if the score is 40-40? Plot this probability as a function of p for $p \in [0, 1]$

2 Spam SMS Detection [45 pts]

Spam mail and SMS detection is an important task to enhance user experience and avoid frauds. For this homework, your task is to create a feature set using a tokenized SMS dataset [1]. Then, you will use this curated feature set to train a Multinomial Naive Bayes model, then test it on your test set. While working on this question, assume that there is another test set that you cannot use.

Dataset

The SMS spam collection dataset contains 5572 SMS texts labeled as either ham or spam. This file is pre-processed to obtain word tokens for each SMS. The corresponding comma separated file is named as `tokenized_corpus.csv`. In this file, each row corresponds to an SMS. Column values denote tokens, separated by commas.

Using the `tokenized_corpus.csv` file, construct your feature set by counting the occurrence of each token in each SMS. First, you have to create an array of all unique tokens, which is called as vocabulary. Then, based on the length of this vocabulary array, fill the frequency values of each token for each document. For instance, SMS i (S_i) will have d features where d is the length of vocabulary. If d_j of S_i is m, then it shows that j^{th} word in vocabulary occurs m times in S_i. Considering your feature matrix as a $2D$ matrix, we can denote the example above as $M[i][j] = m$ where M is your feature matrix. In this case, M will be a $N \times d$ matrix where N is the number of SMS instances.

Question 2.1 (Coding) [5 points] Construct the feature set as described above and save it as `feature_set.csv` using comma as the separator. While constructing your vocabulary, make sure you start counting from the first token of the first SMS. The very first token of the first SMS must be the first word in the vocabulary, and therefore the first feature in your dataset. Similarly, the first SMS should be the first instance in your feature set. Your feature matrix size must be $N \times d$. Your code must generate and save this file while running. Please DO NOT submit pre-generated files.

Once you have constructed the feature set, the next step is to split it into training and test sets. Without shuffling, divide your feature set into 2. First 4460 instances will be in your training set, and the remaining 1112 instances will be in your test set. Labels are provided in the file with name `labels.csv`. The i^{th} label in `labels.csv` depicts the ground truth value of SMS i. 1 denotes spam SMS whereas 0 denotes ham SMS in `labels.csv`.

Bag-of-Words Representation and Multinomial Naive Bayes Model

Notice that the bag-of-words document representation assumes that the probability of a word appearing in an SMS is conditionally independent of the word position given the class of the SMS. If we have a particular SMS document S_i with n_i words in it, we can compute the probability that S_i comes from the class y_k as:

$$\mathbf{P}\left(S_i \,|\, Y = y_k\right) = \mathbf{P}\left(X_1 = x_1, X_2 = x_2, .., X_{n_i} = x_{n_i} \,|\, Y = y_k\right) = \prod_{j=1}^{n_i} \mathbf{P}\left(X_j = x_j \,|\, Y = y_k\right) \tag{2.1}$$

In Eq. (2.1), X_j represents the j^{th} position in SMS S_i and x_j represents the actual word that appears in the j^{th} position in the SMS, whereas n_i represents the number of positions in the SMS. As a concrete example, we might have the first SMS (S_1) which contains 200 words ($n_1 = 200$). The document might be a spam SMS ($y_k = 1$) and the 15th position in the SMS might have the word "free" ($x_j =$ "free").
In the above formulation, the feature vector $\vec{X}$ has a length that depends on the number of words in the SMS n_i. That means that the feature vector for each SMS will be of different sizes. Also, the above formal definition of a feature vector $\vec{x}$ for a SMS says that $x_j = k$ if the j-th word in this SMS is the k-th word in the dictionary. This does not exactly match our feature files, where the j-th term in a row i is the number of occurrences of the j-th dictionary word in that SMS i. As shown in the lecture slides, we can slightly change the representation, which makes it easier to implement:

$$\mathbf{P}\left(S_i \,|\, Y = y_k\right) = \prod_{j=1}^{V} \mathbf{P}\left(X_j \,|\, Y = y_k\right)^{t_{w_j,i}} \tag{2.2}$$

where V is the size of the vocabulary, X_j represents the appearing of the j-th vocabulary word and $t_{w_j,i}$ denotes how many times word w_j appears in an SMS S_i. As a concrete example, we might have a vocabulary of size of 1309, $V = 1309$. The first SMS (S_1) might be spam ($y_k = 1$) and the 80-th word in the vocabulary, w_{80}, is "now" and $t_{w_{80},1} = 2$, which says the word "now" appears 2 times in the SMS S_1. Contemplate on why these two models (Eq. (2.1) and Eq. (2.2)) are equivalent.

In the classification problem, we are interested in the probability distribution over the SMS classes (in this it is either ham or spam) given a particular SMS S_i. We can use Bayes Rule to write:

$$\mathbf{P}\left(Y = y_k | S_i\right) = \frac{\mathbf{P}\left(Y = y_k\right) \prod_{j=1}^{V} \mathbf{P}\left(X_j \,|\, Y = y\right)^{t_{w_j,i}}}{\sum_k \mathbf{P}\left(Y = y_k\right) \prod_{j=1}^{V} \mathbf{P}\left(X_j \,|\, Y = y_k\right)^{t_{w_j,i}}} \tag{2.3}$$

Note that, for the purposes of classification, we can actually ignore the denominator here and write:

$$\mathbf{P}\left(Y = y_k | S_i\right) \propto \mathbf{P}\left(Y = y_k\right) \prod_{j=1}^{V} \mathbf{P}\left(X_j \,|\, Y = y\right)^{t_{w_j,i}} \tag{2.4}$$

$$\hat{y}_i = \arg\max_{y_k} \mathbf{P}\left(Y = y_k \,|\, D_i\right) = \arg\max_{y_k} \mathbf{P}\left(Y = y_k\right) \prod_{j=1}^{V} \mathbf{P}\left(X_j \,|\, Y = y_k\right)^{t_{w_j,i}} \tag{2.5}$$

We can ignore the denominator in Eq. (2.3) since it is the same whether $y_k = 0$ or $y_k = 1$. Therefore, it does not affect the relative order of the probabilities.

Probabilities are floating point numbers between 0 and 1, so when you are programming it is usually not a good idea to use actual probability values as this might cause numerical underflow issues. As the logarithm is a strictly monotonic function on [0,1] and all of the inputs are probabilities that must lie in [0,1], it does not have an effect on which of the classes achieves a maximum. Taking the logarithm gives us:

$$\hat{y}_i = \arg\max_{y} \left(\log \mathbf{P}\left(Y = y_k\right) + \sum_{j=1}^{V} t_{w_j,i} * \log \mathbf{P}\left(X_j \,|\, Y = y_k\right) \right) \tag{2.6}$$

where $\hat{y}_i$ is the predicted label for the i-th example.

The parameters to learn and their MLE estimators are as follows:

$$\theta_{j \mid y=spam} \equiv \frac{T_{j,y=spam}}{\sum_{j=1}^{V} T_{j,y=spam}}$$
$$\theta_{j \mid y=ham} \equiv \frac{T_{j,y=ham}}{\sum_{j=1}^{V} T_{j,y=ham}}$$
$$\pi_{y=spam} \equiv \mathbf{P}\left(Y = spam\right) = \frac{N_{spam}}{N}$$

- $T_{j,spam}$ is the number of occurrences of the word j in spam SMSs in the training set including the multiple occurrences of the word in a single SMS.
- $T_{j,ham}$ is the number of occurrences of the word j in ham SMSs in the training set including the multiple occurrences of the word in a single SMS.
- N_{spam} is the number of spam SMSs in the training set.
- N is the total number of SMSs in the training set.
- $\pi_{y=spam}$ estimates the probability that any particular SMS will be spam.
- $\theta_{j \mid y=spam}$ estimates the probability that a particular word in a spam SMS will be the j-th word of the vocabulary, $\mathbf{P}\left(X_j \mid Y = spam\right)$
- $\theta_{j \mid y=ham}$ estimates the probability that a particular word in a ham SMS will be the j-th word of the vocabulary, $\mathbf{P}\left(X_j \mid Y = ham\right)$

Question 2.2 (Coding) [35 points] Train a Multinomial Naive Bayes classifier using all of the data in the training set you have curated and the ground truth values provided in `labels.csv`. Test your classifier on the test data you have separated, and report the **testing accuracy**. In estimating the model parameters, use the above MLE estimator. If it arises in your code, define $0 * \log 0 = 0$ (note that $a * \log 0$ is as it is, that is -inf). In case of ties, you should predict "spam". Report your **testing accuracy** by writing it into a file with name `test_accuracy.csv`. There must be a single accuracy value in the generated file. Your code must generate and save this file while running. Please DO NOT submit pre-generated files.

Question 2.3 (Coding) [5 points] Extend your classifier so that it can compute an MAP estimate of θ parameters using a fair Dirichlet prior. This corresponds to additive smoothing, or Laplace smoothing. The prior is fair in the sense that it assumes that each word appears additionally α times in the train set.

$$\theta_{j \mid y=spam} \equiv \frac{T_{j,y=spam}+\alpha}{\sum_{j=1}^{V} T_{j,y=spam}+\alpha*V}$$
$$\theta_{j \mid y=ham} \equiv \frac{T_{j,y=ham}+\alpha}{\sum_{j=1}^{V} T_{j,y=ham}+\alpha*V}$$
$$\pi_{y=spam} \equiv \mathbf{P}\left(Y = spam\right) = \frac{N_{spam}}{N}$$

In the equation above, V is the vocabulary size. For this question, set $\alpha = 1$. Train your classifier using all of the training set and have it classify all of the test set and report **testing accuracy** by writing it to a file with name `test_accuracy_laplace.csv`. There must be a single accuracy value in the generated file. Your code must generate and save this file while running. Please DO NOT submit pre-generated files.

3 Feature Selection [25 pts]

One way of improving the performance of your model is to perform feature selection. There are many different ways of performing feature selection. For this section, you are asked to perform feature selection using forward selection and frequency of words. To ease the computational burden, slightly change your feature set generation code to only consider words that occur at least 10 times across the whole dataset. This way, your vocabulary will shrink and the amount of features you have will diminish. This new vocabulary is referred as V_r for the remainder of this section. You don't need to save this new feature set to a file. Use this filtered feature set and $\alpha = 1$ for questions 3.1 and 3.2.

Hint: In these questions, you are required to train your model from scratch thousands of times. Therefore, your train and test functions must be as efficient as possible. As a reference, each question (Q3.1 and Q3.2) should take no more than 5 minutes to complete on an average CPU.

Question 3.1 (Coding) [12.5 points] Perform forward selection based on test set accuracy values until no performance gain is obtained. Write the indices of selected words to a file with name `forward_selection.csv`. In this file, you must report indices of selected features line by line so that each index will be in a separate line. Your code must generate and save this file while running. Please DO NOT submit pre-generated files.

Question 3.2 (Coding) [12.5 points] Calculate the frequency of each word in the training set. Sort your features in descending order with respect to their frequencies. Then, train your model using the most k frequent words for $k \in [1, V_r]$ where V_r is your new vocabulary size for reduced feature set. In other words, you are asked to train your model V_r times using the most frequent k words, increasing k by 1 each time. Write the testing accuracy for each k value (starting from $k = 1$) line by line to a file with name `frequency_selection.csv`. Your code must generate and save this file while running. Please DO NOT submit pre-generated files.

4 Principal Component Analysis [15 pts]

Principal component analysis (PCA) is a dimensionality reduction method for data visualization or feature extraction. The aim of PCA is to maximize variance of z_1, which is the projection of original features x on w_1 vector. We know that $Var(z_1) = w_1^T \Sigma w_1$ where $Cov(X) = \Sigma$. We also require that $\|w_1\| = 1$. Using this information,
Question 4.1 [5 points] Show that the first principal component is the eigenvector of the covariance matrix with the largest eigenvalue.

Question 4.2 [10 points] Show that the second principal component is the eigenvector of the covariance matrix with the second largest eigenvalue (Hint: You have to add orthogonality constraint).

References

1. UCI - SMS Spam Collection https://archive.ics.uci.edu/ml/datasets/sms+spam+collection
2. ”On Discriminative vs. Generative Classifiers: A comparison of logistic regression and Naive Bayes” by Andrew Ng and Michael I. Jordan.
3. Manning, C. D., Raghavan, P., and Schutze, H. (2008). Introduction to information retrieval. New York: Cambridge University Press.
http://nlp.stanford.edu/IR-book/html/htmledition/mutual-information-1.html
4. CMU Lecture Notes.
http://www.cs.cmu.edu/~epxing/Class/10701-10s/Lecture/lecture5.pdf