CS464 Machine Learning, Spring 2015: Homework 2

Due: April 1 Wednesday, 17:00 pm

Instructions

- Submit a hard copy of your homework of all questions. Add the print out of your code at the end of the your submission and upload the code on Moodle. You may hand in the hard copy in classes to the instructor or may drop it in the box in room EA425 (please stick to this submission routes) and please STAPLE your write up.
- You may code in any programming language you would prefer. In submitting the code on Moodle, please package your code as a gzipped TAR file or a ZIP file with the name `CS464_HW1_Mustafa_Buyukozkan`, where you substitute in your first and last names into the filename in place of "Mustafa_Buyukozkan". The code you submit should be in a format easy to run, main script to call other functions. You must also provide us with a README file that tells us how we can execute/call your code.
- If you are submitting the homework *late* (see the late submission policy in syllabus), prepare a **soft copy** for all the parts of the homework, submit everything online on Moodle. Moodle will allow late submissions until after 4 days of submission, but we will grade your homework based on the time stamp and your remaining late days.
- Please refer to the syllabus for the late day policy.
- If you do not follow the submission routes, deadlines and specifications (codes, report, etc), it will lead to significant grade deduction.

1 Building a Webpage Classifier with Naive Bayes

Your job is to build a classifier that can accurately predict whether a webpage belongs to a student or a faculty. The questions summarizes the model therefore please read all questions before starting coding.

1.1 Dataset

Your dataset is a preprocessed and modified subset of the Web-Kb Dataset [1]. It is based on 1400 real (faculty or student) webpages collected from various computer science department. Webpages have been preprocessed in the following ways:

- Stop word removal: Words like "and","the", and "of", are very common in all English sentences and are therefore not very predictive in deciding student/faculty status. These words have been removed from the webpages.
- Lemmatization: Words that have the same meaning but different endings have been adjusted so that they all have the same form. For example, "include", "includes," and "included," would all be represented as "include." All words in the webpage body have also been converted to lower case.
- Removal of non-words: Numbers and punctuation have both been removed. All white spaces (tabs, newlines, spaces) have all been trimmed to a single space character.

The features have been generated for you. The `data.txt` file contains the features and labels. In `data.txt`, each row contains the feature vector for a webpage. The j-th term in a row i is the number of occurrences of the j-th vocabulary word in the i-th webpage. The size of the vocabulary is 1309. Also first row indicates the labels. (For the notations below, 0 stands for "Faculty" and 1 stands for "Student"'.)

1.2 Cross-Validation

You will employ 10-fold cross validation to evaluate each of the models. We split the data into 10 folds and they are provided in the `folds.txt`. The order of this examples is the same as the order of the `data.txt` and the number indicates in which test fold the example is in. Thus, set aside examples with fold id 1, train the model with the rest. Calculate calculate the validation error on these set aside examples and calculate the training accuracy on the examples you use to train the model. Repeat this 10 times, where in each case you set aside a fold. Average the training accuracies to get a CV training accuracy and average the validation accuracies to ge the CV validation accuracy.

1.3 Bag of Words Representation and Multinomial Naive Bayes Model

Recall the Bag of Words document representation makes the assumption that the probability that a word appears in document is conditionally independent of the word position given the class of the document. If we have a particular document D_i with n_i words in it, we can compute the probability that D_i comes from the class y_k as:

$$\mathbf{P}\left(D_i \,|\, Y = y_k\right) = \mathbf{P}\left(X_1 = x_1, X_2 = x_2, .., X_{n_i} = x_{n_i} \,|\, Y = y_k\right) = \prod_{j=1}^{n_i} \mathbf{P}\left(X_j = x_j \,|\, Y = y_k\right) \tag{1.1}$$

In Eq. (1.1), X_j represents the j^{th} position in document D_i and x_j represents the actual word that appears in the j^{th} position in the document, whereas n_i represents the number of positions in the document. As a concrete example, we might have the first webpage document (D_1) which contains 200 words ($n_1 = 200$). The document might be of student class ($y_k = 1$) and the 15th position in the document might have the word "office" ($x_j =$ "office").

In the above formulation, the feature vector $\vec{X}$ has a length that depends on the number of words in the webpage n_i. That means that the feature vector for each webpage will be of different sizes. As shown in the lecture slides, we can slightly change the representation, which makes it easier to implement:

$$\mathbf{P}\left(D_i \,|\, Y = y_k\right) = \prod_{j=1}^{V} \mathbf{P}\left(X_j \,|\, Y = y_k\right)^{t_{w_j,i}} \tag{1.2}$$

,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 document D_i. As a concrete example, we might have a vocabulary of size of 1309, $V = 1309$. The first webpage document (D_1) might be of Student class ($y_k = 1$) and the 80-th word in the vocabulary, w_{80}, is "click" and $t_{w_{80},1} = 2$, which says the word "click" appears 2 times in webpage D_1. Contemplate on why these two models (Eq. (1.1) and Eq. (1.2)) are equivalent.

In the classification problem, we are interested in the probability distribution over the document classes (in this case Student and Faculty) given a particular document D_i. We can use Bayes Rule to write:

$$\mathbf{P}\left(Y = y_k | D_i\right) = \frac{\mathbf{P}\left(Y = y_k\right) \prod_{j=1}^{V} \mathbf{P}\left(X_j \,|\, Y = y\right)^{t_{w_j,i}}}{\sum_k \mathbf{P}\left(Y = y_k\right) \prod_{j=1}^{V} \mathbf{P}\left(X_j \,|\, Y = y_k\right)^{t_{w_j,i}}} \tag{1.3}$$

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

$$\mathbf{P}\left(Y = y_k | D_i\right) \propto \mathbf{P}\left(Y = y_k\right) \prod_{j=1}^{V} \mathbf{P}\left(X_j \,|\, Y = y\right)^{t_{w_j,i}} \tag{1.4}$$

$$\hat{y}_i = \arg\max_{y_k} \mathbf{P}\left(Y = y_k \mid D_i\right) = \arg\max_{y_k} \mathbf{P}\left(Y = y_k\right) \prod_{j=1}^{V} \mathbf{P}\left(X_j \mid Y = y_k\right)^{t_{w_j,i}} \tag{1.5}$$

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 affect on which of the classes achieves a maximum. Taking the logarithm gives us:

$$\hat{y}_i = \arg\max_{y} \left(\log \mathbf{P}\left(Y = y_k\right) + \sum_{j=1}^{V} t_{w_j,i} * \log \mathbf{P}\left(X_j \mid Y = y_k\right) \right) \tag{1.6}$$

, where $\hat{y}_i$ is the predicted label for the i-th example. The above formal definition of a feature vector $\vec{x}$ for a document says that $x_j = k$ if the j-th word in this document 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 document i.

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

$$\theta_{j \mid y=0} \equiv \frac{T_{j,y=0}}{\sum_{j=1}^{V} T_{j,y=0}}$$

$$\theta_{j \mid y=1} \equiv \frac{T_{j,y=1}}{\sum_{j=1}^{V} T_{j,y=1}}$$

$$\pi_{y=1} \equiv \mathbf{P}\left(Y = 1\right) = \frac{N_1}{N}$$

- $T_{j,0}$ is the number of occurences of the word j in faculty documents in the trianing set including the multiple occurences of the word in a single document.
- $T_{j,1}$ is the number of occurences of the word j in student documents in the training set including the multiple occurences of the word in a single document.
- N_1 is the number of student documents in the training set.
- N is the total number of documents in the training set.
- $\pi_{y=1}$ estimates the probability that any particular webpage will be a student webpage.
- $\theta_{j \mid y=0}$ estimates the probability that a particular word in a faculty webpage will be the j-th word of the vocabulary, $\mathbf{P}\left(X_j \mid Y = 0\right)$
- $\theta_{j \mid y=1}$ estimates the probability that a particular word in a student webpage will be the j-th word of the vocabulary, $\mathbf{P}\left(X_j \mid Y = 1\right)$

Question 1.1 Multinomial Document Model (Coding) [35 points] Implement Multinomial Naive Bayes (MNB) classifier and report accuracies on the training data and on the validation data as well as corresponding confusion matrices (how many wrong predictions were made for each classes).

- 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 affect on which of the classes achieves a maximum. Taking the logarithm gives us:

$$\hat{y}_i = \arg\max_{y} \left(\log \mathbf{P}\left(Y = y_k\right) + \sum_{j=1}^{V} t_{w_j,i} * \log \mathbf{P}\left(X_j \mid Y = y_k\right) \right) \tag{1.7}$$

- Define $0 * \log 0 = 0$ (note that $a * \log 0$ is as it is, that is -inf) .
- In case of ties, you should predict the label in the “Faculty’ class.

In estimating the model parameters:

a) Use the above MLE estimator described above.
b) Use add-one smoothing (Laplace smoothing, which is equivalent to MAP estimation with Drichlet prior):

$$\hat{\theta}_{j\,|\,y=0} \equiv \frac{1+T_{j,y=0}}{|V|+\sum_{j=1}^{V} T_{j,y=0}}$$

$$\hat{\theta}_{j\,|\,y=1} \equiv \frac{1+T_{j,y=1}}{|V|+\sum_{j=1}^{V} T_{j,y=1}}$$

where $|V|$ = feature vector size (how many different word in the dictionary)

Discuss the results of the models. Explain in words why you see the change in accuracy that you observe.

Question 1.2 Bernoulli Model (Coding) [15 points] Now train a model with bernoulli document model. Your features should take value of 1 if the word is present in the corresponding document and 0 if it is absent. You may modify the earlier feature representation by simply (set $x_{i,j} = 1$, if $x_{i,j} > 0$) for classification. In estimating the model parameters, use MLE and add-one smoothing. Compare your results.

Question 1.3 TF-IDF Model (Coding) [25 points] TF-IDF score is commonly used in information retrieval tasks but is also useful for text categorization. TF-IDF is a numerical statistic that is intended to reflect how important a word is to a document in a collection: tf means term-frequency while idf means inverse document-frequency. A term occurring frequently in the document but rarely in the rest of the collection is given high weight. You may want to check out the wikipedia page.
Calculate the tf score for a word, t for a document d, as follows :

$$\text{tf(t, d)} = \begin{cases} 1 + \log f(t,d) & \text{if } f(t,d) > 0 \\ 0 & \text{if } f(t,d) = 0 \end{cases}$$

where$f(t,d)$ is the number of times the word t occurs in the document d. Note that tf can be formulated in different ways, here it is logaritmically scaled. Inverse document frequency measures whether the term is common or rare across all documents:

$$\text{idf}(t,D) = \log \frac{N}{1 + |\{d \in D : t \in d\}|}$$

where N is the number of documents in the training data and $|\{d \in D : t \in d\}|$: number of documents where the term t appears. The tf-idf score combines the two scores :

$$\text{tf} - \text{idf}(t,d,D) = \text{tf}(t,d) \times \text{idf}(t,D)$$

Train a multinomial document model where the features are weighted with tf-idf scores. Compare the accuracy of the best models of the three different models created with different feature representation i) word counts (multinomial model) ii) word presence and absence (bernoulli model) iii) tf-idf score weighted features. Report accuracies and the confusion matrices.

Question 1.4 (Coding) [25 points] You will modify the dataset and compare the results by doing modifications to the 668^{th} feature (669'th column in the dataset, first column is labels):

1. Remove this feature.
2. Duplicate this feature n times where you vary $n \in \{1, 2, 5, 10, 20\}$ (In Matlab you can do this by the `repmat` function.)

Rerun your best MNB classifier on the each of these datasets. Report confusion matrices and accuracies for each cases. Plot and discuss the distrubution of misclassified samples for the cases above. Explain in words why you see the change in accuracy that you observe. Which of the following the 668'th word could be : **a)** professor **b)**activity **c)** monday **d)** advisor **e)** computer? Justify your answer.

BONUS Question [10 points]Try to increase model performance, explain and justify your methodology.

References
1. Knowledge Base (Web-KB) project dataset. `http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/`