Programming Assignment 5 – Basic Probability, Computing and Statistics 2015

Fall 2015, Master of Logic, University of Amsterdam

Submission deadline: Monday, October 5th, 9 a.m.

Note: if the assignment is unclear to you or if you get stuck, do not hesitate to contact Philip.

1 Naïve Bayes for Text Classification

This week you will achieve two of the big goals of this course. You will implement your first machine learning algorithm and run it on a real data set. The algorithm will be naïve Bayes. You will have to implement parameter learning by maximum likelihood estimation and prediction.

Data set The data set is the 20 new groups data set. It consists of roughly 1000 post for each of 20 newsgroups. The posts are of varying length but usually do not exceed the size of an e-mail. The data set is a classic in text classification and much research had been done with it in the beginning and mid 2000s. Nowadays it is considered too small and not very representative (after all, you want to learn to classify text beyond what people talk about in the 20 newsgroups.

We have pre-processed the data for you. We removed all punctuation and also split it up into a training set, a development set and a test set. This split is standard in machine learning. The training set contains the largest bulk of data and you use it to train your algorithm. The development set is usually rather small and allows you to check your algorithms performance on unseen data while developing. The test set is the final criterion of success. In a research paper you report your results on the test set. It is meant to represent new, unseen data. This means that you should not run you algorithm on it till the very end.

You can find the training data here and the dev set here. The correct labels for the dev set are supplied here. The test set will only be made available next week to simulate a situation in which you really do not know the data on which you are going to apply your algorithm.

The model The model you need to implement is a naïve Bayes model. Recall that for some value $y$and n i.i.d. observations of x_1^n this is a model of the form

$$P(Y = y|X_1^n = x_1^n) \propto \prod_{i=1}^{n} P(X_i = x_i|Y = y) \times P(Y = y) \ .$$

In our case the random variable Y will range over labels and X will range over **features** (observations). The labels are the names of the news groups and the features are the words in the texts. The goal of the classifier is to learn the θ parameters the distributions using maximum likelihood (when you calculate the probabilities in the prediction step, you can neglect the multinomial constant since we are only looking for a proportional quantity anyway). We assume that all distributions are multinomials. The MLE for θ_i is the $\frac{\#o_i}{n}$, where $\#o_i$ is the count of the associated outcome (a label for Y and a word for X) and n is the total number of observations (the number of documents for Y and the number of words in all documents that carry the same label for X).

Implementation When implementing naïve Bayes, we have to take care of two things:

1. Most documents in the test and development sets will contain words that our classifier has never seen before. Hence, the maximum likelihood estimate for such words will be word $= 0$. This means the entire document will have probability 0. This is obviously not what we want. Therefore we apply the following trick: after having counted the words in the entire training set, we remove the words that have only have a count of 1, add up their counts and assign them to the unknown word `__UNK__`. In the prediction step, whenever we encounter a word that we have not seen before, we just pretend it is `__UNK__`. This way we can assign probabilities to all word.

2. The probabilities of each document will be rather low, so low in fact that they might get zeroed out. For this reason we will work with log-probs instead. Notice that this turns all multiplications into addtions.

2 Grading

The grading is going to be based on the performance of your classifier rather than on your code alone. Please do not stress out about this. Try to work together and test your methods frequently (e.g. write your onw unit tests).

8 points if the classifier achieves an accuracy of TBD% on the test set. You should use the dev set accuracy as an indicator for your classifiers

performance. The classifier should achieve TBD% accuracy on the dev set.

2 points if you add a command line option that allows to modify the threshold on the word count that determines what words get mapped to the unknown word.

3 extra points if you manage to improve the performance of your classifier on the test set. This can achieved by incorporating more features. For example, you may want to estimate the probability of upper case letters or include pairs of words (so-called bigrams) as features. Notice that if you include bigrams you also want to include the unknown bigram, that fulfills the same function as the unknown word. Furthermore, you may want to introduce a third random variable Z over the additional features. This way, you will keep the distributions over words and your additional features separate. The model then becomes

$$P(Y = y|X_1^n = x_1^n, Z_1^m = z_1^m) \propto \\ P(Y = y)\prod_{i=1}^{n} P(X_i = x_i|Y = y)\prod_{j=1}^{m} P(Z_j = z_j|Y = y)$$

where we assume conditional independence between all X and Z.

Both for development and grading purposes, we will soon provide a script that checks the accurracy of your output.