10-601 Machine Learning: Homework Assignment 3: Solution to Purna's question

Professor Tom Mitchell
Carnegie Mellon University
February 3, 2009

- The assignment is due at 1:30pm (beginning of class) on **Wednesday, February 18, 2009**.
- Submit writeups to Problem 1 and Problem 2 *separately* with your name on each problem. Please do not staple the two writeups together.
- Write your name at the top right-hand corner of each page submitted.
- Each student must hand in their own answers to the following questions, and their own code. To submit your code, please send it as an attachment via email to `purnamritas` AT `gmail.com`. Package your code as a gzipped TAR file or a ZIP file with the prefix `601hw3-johndoe` where you substitute in your first and last names into the filename in place of '`johndoe`'. See the course webpage for the collaboration policies.
- Each question has the name of the TA who is the primary contact point for that question. Feel free to ask the other instructors about any question, but that TA is the authority on that question.

1 Logistic Regression (LR) and Naive Bayes (NB) [Purna: 65 points]

The data: In this assignment you will train a Naive Bayes and a Logistic Regression classifier to predict the class of a set of documents, represented by the words which appear in them. Please download the data from here[1]. The .data file is formatted "docIdx wordIdx count". Note that this only has words with nonzero counts. The .label file is simply a list of label id's. The ith line of this file gives you the label of the document with docIdx i. The .map file maps from label id's to label names. In this assignment you will classify documents into two classes: rec.sport.baseball (10) and rec.sport.hockey (11). The vocabulary.txt file contains the vocabulary for the indexed data. The line number in vocabulary.txt corresponds to the index number of the word in the .data file.

1.1 Implement Logistic Regression and Naive Bayes

1. Implement regularized Logistic Regression using gradient descent. Your instructors found that learning rate η around 0.0001, and regularization parameter λ around 1 works well for this dataset. This is just a rough point to begin your experiments with, please feel free to change the values based on what results you observe. Report the values you use. One way to determine convergence might be by stopping when the maximum entry in the absolute difference between the current and the previous weight vectors falls below a certain threshold. You can use other criteria for convergence if you prefer. Please specify what you are using. In each iteration report the log-likelihood, the training-set misclassification rate and the norm of weight difference you are using for determining convergence.

[1] These datasets were collected from Jason Rennie's webpage.

2. Implement the Naive Bayes classifier for text classification using the principles in Tom's lecture in class. You can use a "hallucinated" count of 1 for the MAP estimates.

1.2 Feature Selection

1. Train your Logistic Regression algorithm on the 200 randomly selected datapoints provided in here. Now look for the indices of the words "baseball", "hockey", "nfl" and "runs". If you sort the absolute values of the weight vector obtained from LR in descending order, where do these words appear? Based on this observation, how would you select interesting features from the parameters learnt from LR?
   $\star$ *Solution*: "baseball" is at rank 4. "hockey" is at rank 1. "nhl" is at rank 3, and "runs" is at rank 2. This shows that a simple feature selection algorithm is to pick the top k elements from a list of features, which are sorted in a descending order of their absolute w values. Some of the students pointed out that words which occur very often like "of,and,at" come up towards the top of the list. My understanding is that these words have very large count and hence pick up large weight values even if they are very common in both classes. One way to fix this will be to regularize different words differently. The way we would do is by introducing a penalization term $\sum_i \lambda_i w_i^2$ instead of $\lambda \sum_i w_i^2$ in the log-likelihood.

2. Use roughly $\frac{1}{3}^{rd}$ of the data as training and $\frac{2}{3}^{rd}$ of it as test. About half the number of documents are from one class. So pick the training set with a equal number of positive and negative points (198 of each in this case). Now using your feature selection scheme from the last question, pick the [20, 50, 100, 500, all] most interesting features and plot the error-rates of Naive Bayes and Logistic Regression. Remember to average your results on 5 random training-test partitions. What general trend do you notice in your results? How does the error rate change when you do feature selection? How would you pick the number of features based on this?
   $\star$ *Solution*: In figure 1.2 we see that the error-rate is high for a very small set of features (which means the top k features (for a small k) are missing some good discriminative features). The error rate goes down as we increase the number of interesting features. With about 500 good features we obtain as good classification accuracy as we can get with all the features included. This implies that feature selection helps. I would pick 500 words using this scheme, since that would help reduce both time and space consumption of the learning algorithms and at the same time give me small error-rate.

1.3 Highly Dependent Features: How do NB and LR differ?

In question 1 you considered the impact on Naive Bayes when the conditional independence assumption is violated (by adding a duplicate copy of a feature). Also question **??** formulates the discriminative analog of Naive Bayes, where we explicitly model the joint distribution of two features. In the current question, we introduce highly *dependent* features to our Baseball Vs. Hockey dataset and see the effect on the error rates of LR and NB. A simple way of doing this is by simply adding a few duplicate copies of a given feature to your dataset. First create a dataset D with the wordIds provided here. For each of the three words *baseball, hockey*, and *runs*:

1. Add 3 and 6 duplicate copies of it to the dataset D and train LR and NB again. Now report the respective average errors obtained by using 5 random train-test splits of the data (as in 1.1). For each feature report the average error-rates of LR and NB for the following:

   - Dataset with no duplicate feature added (D).

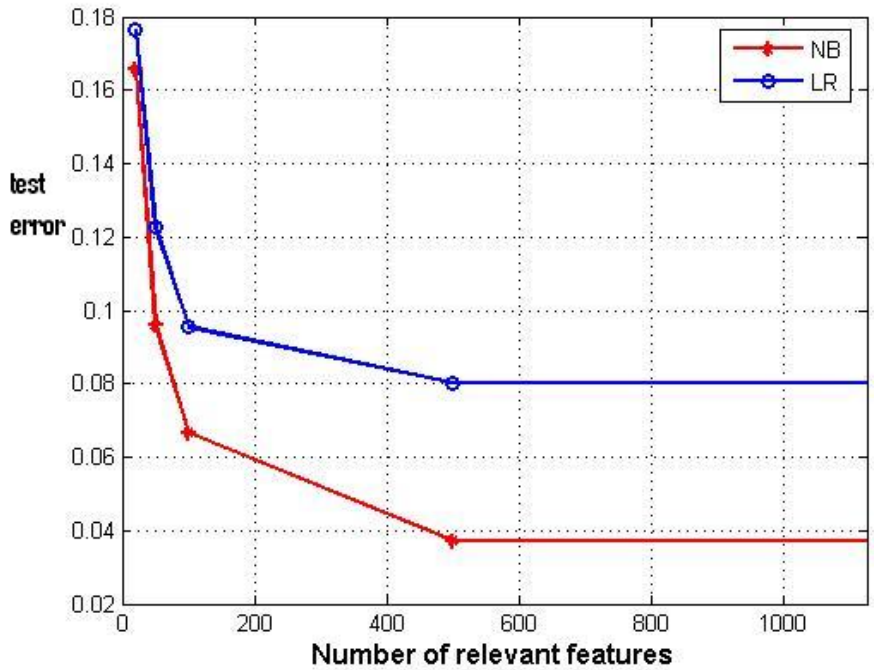

Figure 1: x axis is number of features and y axis has the test error.

- Dataset with 3 duplicate copies of feature added (D').
- Dataset with 6 duplicate copies added (D").

  ⋆ *Solution*:

Word=**baseball**

Dataset	*LR*	*NB*
D	0.1766	0.1615
D'	0.1751	0.1889
D''	0.1746	0.2252

Word=**hockey**

Dataset	*LR*	*NB*
D	0.1746	0.1618
D'	0.1668	0.1746
D''	0.1711	0.2242

Word=**runs**

Dataset	*LR*	*NB*
D	0.1635	0.1595
D'	0.1731	0.1965
D''	0.1728	0.2450

In order to have a fair comparison, use the same set of test-train splits for each of the above cases.

2. How do Naive Bayes and Logistic Regression behave in the presence of duplicate features?

   ⋆ *Solution*: The error-rate of Naive Bayes increases a lot compared to the error-rate of

Logistic regression, as we keep duplicating features.

3. Now compute the weight vectors for each of these datasets using logistic regression. Let W, W', and W'' be the weight vectors learned on the datasets D, D', and D'' respectively. You do not have to do any test-train splits. Compute these on the entire dataset. Look at the weights on the duplicate features for each case. Based on your observation can you find a relation between the weight of the duplicated feature in W', W'' and the same (not duplicated) feature in W? How would you use this observation to explain the behavior of NB and LR?

   $\star$ *Solution*: We see that each of the duplicated features in one dataset has identical weight values. Here is the table with the weights of different words for datasets D, D', and D''. I have excluded the weights of all 3 or 6 duplicates, since they are all identical.

Dataset	*baseball*	*hockey*	*runs*
D	1.2835	-3.1645	1.8859
D'	0.3279	-0.9926	0.5722
D''	0.1878	-0.5826	0.3302

   Note that in each case LR divides the weight of a feature in D *roughly equally* among its duplicates in D' and D''. For example for word *hockey* $3 * 0.5722 = 1.7$ and $6 * 0.3302 = 1.9$, whereas the original feature weight is 1.9. Since NB treats each duplicate feature as conditionally independent of each other given the class variable, its error rate goes up as the number of duplicates increases. As a result LR suffers less from double counting than NB does.