CS57300: Homework 2 - SOLUTION

Your homework must be typed and submitted as a PDF. Use of Latex is recommended, but not required.

Programming assignment

1. Write code to transform a given input file to a bag-of-words representation. (5 pts)

   (a) Compute binary features for the 101-600 most frequently occurring words as described above. Note that the frequencies should be computed (and selected) from the training set, not the entire data.

   Please see the attached code.

   (b) Output the top ten selected words with highest frequency (i.e., words 101-210). (Make sure you count every word once in each review (even though it appear multiple times) since we are using bag of words representation.)

   ```
   ive
   definitely
   people
   do
   worth
   delicious
   burger
   again
   did
   than
   ```

   (c) Construct the bag of words features each review in both the training and the test set, based on the words selected from the training set.

   Please see the attached code.

2. Implement a Naive Bayes Algorithm. (20 pts)

   (a) Write code to read in training data and learn the NBC model.

   Please see the attached code.

   (b) Write code to read in test data to apply the learned NBC and evaluate the resulting predictions with zero-one loss.

   Please see the attached code.

3. Learn and apply the algorithm (15 pts)

   (a) For each % in [1, 5, 10, 20, 50, 90]:

   - Randomly sample % of the data to use for training.
   - Use the remaining (1-%) of the data for testing.
   - Learn a model from the training data and apply it to test data.

- Measure the performance on the test data using zero-one loss.
- Repeat ten times with different random samples (using the same %).
- Record avg. and st. dev. of zero-one loss across the ten trials.

(b) Plot a learning curve for the results (training set size vs. zero-one loss). Compare to the baseline *default* error that would be achieved if you just predicted the most frequent class label. Discuss the results.

Please see Figure 1.

Discussion:

- NBC is always better than the baseline in this experiment. (Good news.)
- When we increase the training percentage, the error decreases, this means our classifier learns better with more data. (Good news.)
- After around 0.2-0.3 percentage training size, increasing the training percentage does not help a lot.
- The variance seems to be increases with more data, that is counter intuitive, I believe this is because the test set size decreases with more training data.

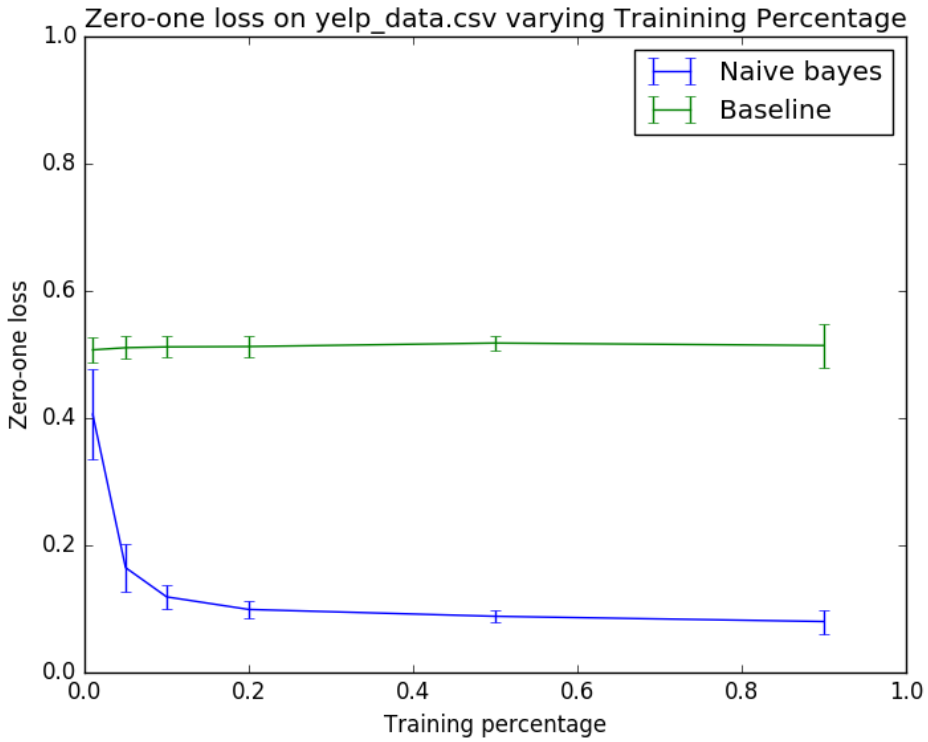

Figure 1: Zero-one loss on yelp_data.csv varying training percentage

4. Explore effect of feature space (10 pts)

(a) For each W in [10, 50, 250, 500, 1000, 4000]:

- Drop the 100 most frequent words and construct features for the next W most frequent words (i.e., you will vary how many features are used in the model). Randomly sample 50% of the data to use for training.
- Use the remaining (50%) of the data for testing.
- Learn a model from the training data and apply it to test data.
- Measure the performance on the test data using zero-one loss.
- Repeat ten times with different random samples (using the same %).
- Record avg. and st. dev. of zero-one loss across the ten trials.

(b) Plot a learning curves for the results (feature size vs. zero-one loss). Again compare to the baseline *default* error that would be achieved if you just predicted the most frequent class label. Discuss the results.

Please see Figure 2.

Discussion:

- NBC is always better than the baseline in this experiment. (Good news.)
- When we increase the feature space size (W), the error decreases, this means our classifier learns better with more features. (Good news.) This is probably because we consider more words that has some information in them and use that information as well.
- Using a bigger feature space is computationally more expensive, so there is a tradeoff between accuracy vs computational complexity here.

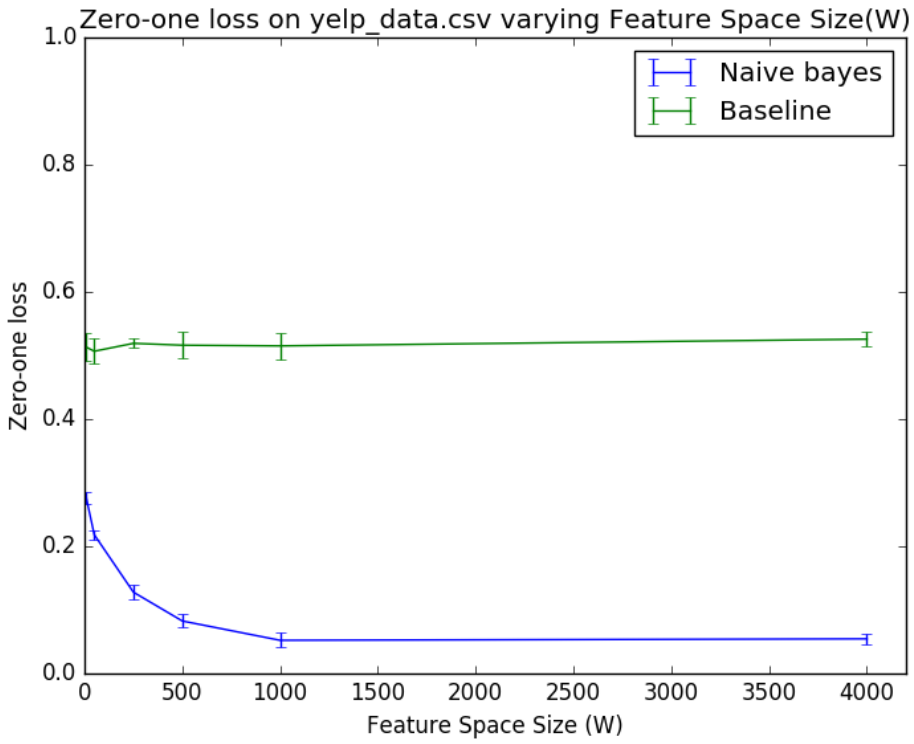

Figure 2: Zero-one loss on yelp_data.csv varying feature space size