CS57300: Homework 3 - SOLUTION

Due date: Tuesday March 14, midnight (submit via turnin)

Comparing Methods for Yelp Review Classification

In this programming assignment you will run further experiments with the classification task that you explored in HW2.

Consider the following setup from HW2:

- Data: Use the `yelp_data.csv` dataset (which we refer to as D) with 2000 reviews.
- Features: Compute binary word features as before; discard the top 100 words, use **4000** features (i.e., 101-4100 most frequent words).
- Class label: Use the same class label *isPositive*.
- Evaluation: Use same evaluation measure: zero-one loss.
- Use the following proportions of D as training set sizes: $[0.01, 0.03, 0.05, 0.08, 0.1, 0.15]$.
  NB: these are different proportions than we used for HW2.

New for this assignment:

- Implement Logistic Regression and Linear SVM to compare to NBC.
  1. Logistic Regression: Use L2 regularization, with $\lambda = 0.01$. Optimize with gradient descent, using step size of 0.01.
  2. SVM: Use hinge loss. Optimize with subgradient descent, using step size of 0.5 and regularization parameter $\lambda = 0.01$.
  3. In both methods, use a cutoff on the number of iterations ($max = 100$) and cutoff on the difference between new and old weights ($tol = 1e-6$) to stop the optimization.
- Use *incremental* 10-fold cross validation to compute learning curves with training sets of varying size, but constant test set size.
  1. Randomly partition the examples from the dataset D into 10 disjoint sets $\mathbf{S} = [S_1, ..., S_{10}]$, where each set has 200 examples.
  2. For each training set size TSS (e.g., $0.05|D| \implies TSS = 100$):
     (a) For $idx = [1..10]$
        i. Let $test\_set = S_{idx}$.
        ii. Let $\mathbf{S}_C = \bigcup \mathbf{S}_{i=[1..10], i \neq idx}$.
        iii. Construct $train\_set$ by randomly selecting TSS examples from $\mathbf{S}_C$.
        iv. Learn each model (i.e., NBC, LR, SVM) from $train\_set$.
        v. Apply each model to $test\_set$; measure loss $L_{0/1}$.
     (b) Record the average $L_{0/1}$ for each model over the ten trials, and its *standard error* (see below). Plot on learning curve of performance vs. TSS, for each model.

- Report the average performance (e.g., $L_{0/1}$) over the ten-fold cross validation trials and the *standard error*. For each training set size you will have 10 measures of performance (one for each fold). For example, you may have $\mathbf{L} = [0.16, 0.18, 0.19, 0.15, 0.19, 0.21, 0.21, 0.16, 0.18, 0.16]$. Then you can compute the mean and standard deviation of $\mathbf{L}$ to be $\mu_{\mathbf{L}} = 0.179$ and $\sigma_{\mathbf{L}} = 0.021$. The standard error is the standard deviation divided by the square root of the number of trials (in our case 10):

$$sterr_{\mathbf{L}} = \frac{\sigma_{\mathbf{L}}}{sqrt(num\_trials)} = 0.007$$

Programming assignment

You should implement your solution using Python. You can use supporting libraries like numpy, scipy as before, but you may not use scikit-learn. As before, you should submit your typed HW report as a pdf along with your source code file.

Code (20 pts)

Name your file hw3.py. Your python script should take three arguments as input.

1. *trainingDataFilename*: corresponds to a subset of the Yelp data (in the same format as `yelp_data.csv`) that should be used as the *training set* in your algorithm.

2. *testDataFilename*: corresponds to another subset of the Yelp data (again in the same format) that should be used as the *test set* in your algorithm.

3. *modelIdx*: an integer to specify the model to use for classification (LR= 1 and SVM= 2).

Use the same output format as for HW2, but (i) no need to output the top 10 words, and (ii) append the model name to the score, E.g.:

```
ZERO-ONE-LOSS-LR 0.3106
```

or

```
ZERO-ONE-LOSS-SVM 0.2947
```

Note that your submitted code should include both the basic code described above to test your models from the command line and the code you use for the analysis below.

Refer to attached code.

Analysis (20 pts)

1. *Assess whether choice of model improves performance.*

   (a) Plot the learning curves for the three models (in the same plot), including error bars that indicate ±1 standard error, from the evaluation based on incremental CV as described above.

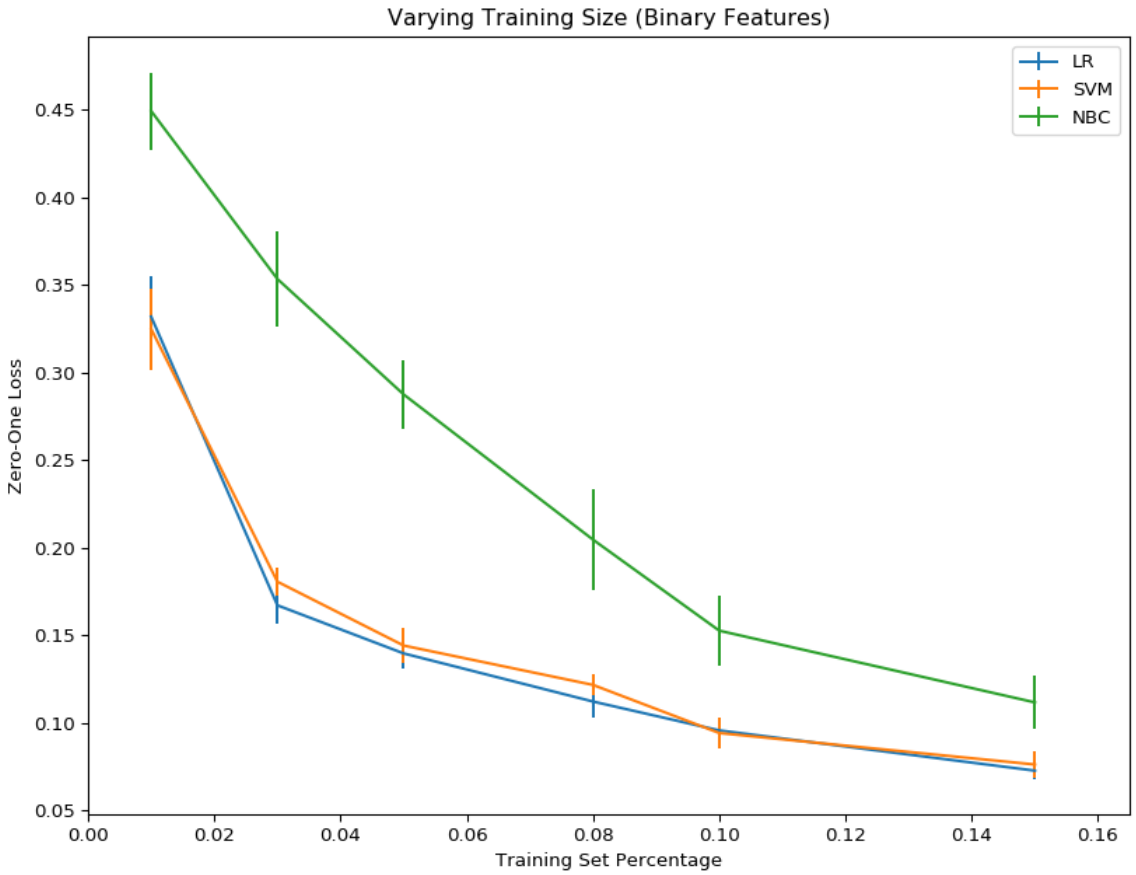

Figure 1: Varying Training Set Size with Binary Features.

   (b) Formulate a hypothesis about the performance difference between at least two of the models.
   Let μ_{LR} refer to mean zero-one loss of the Logistic Regression Classifier and μ_{NBC} refer to the mean zero-one loss of the Naive-Bayes Classifier.

   Null Hypothesis (H_0): $\mu_{LR} = \mu_{NBC}$

   Alternative Hypothesis (H_1): $\mu_{LR} < \mu_{NBC}$

   (Replacing Logistic Regression with SVM is another possible hypothesis.)

(c) Discuss whether the observed data support the hypothesis (i.e., are the observed differences significant).

From the graph, we see that the Naive-Bayes classifier has a higher 0/1 loss for all training set sizes compared to Logistic Regression (or SVM). This difference is significant because the standard error bars of Logistic Regression (or SVM) do not overlap and are sufficiently far away from that of the Naive-Bayes classifier.

Alternatively, we can perform a one-tailed paired t-test for each training set size. We will choose our significance $\alpha = 0.05$. In order to correct for testing multiple hypotheses, we apply Bonferroni's correction. We reject the null hypothesis if the p-value is less than $\frac{\alpha}{6} = 0.0083$.

Training Set Percentage	**p-value**
0.01	0.000683340036864
0.03	8.8843043622e-06
0.05	2.44761017431e-06
0.08	0.00129865351959
0.10	0.00468702417935
0.15	0.019906633468

From the above table, we see that we can reject the null hypothesis for all training set percentages except for 0.15.

2. *Assess whether feature construction affects performance.*
Instead of constructing binary word features, construct features with three values (0: word does not exist, 1: word occurs once, 2: word occurs 2 or more times). Use the new features in each of the three models.

   (a) Plot the learning curves for the three models (in the same plot), including error bars that indicate ±1 standard error, from the evaluation based on incremental CV as described above.

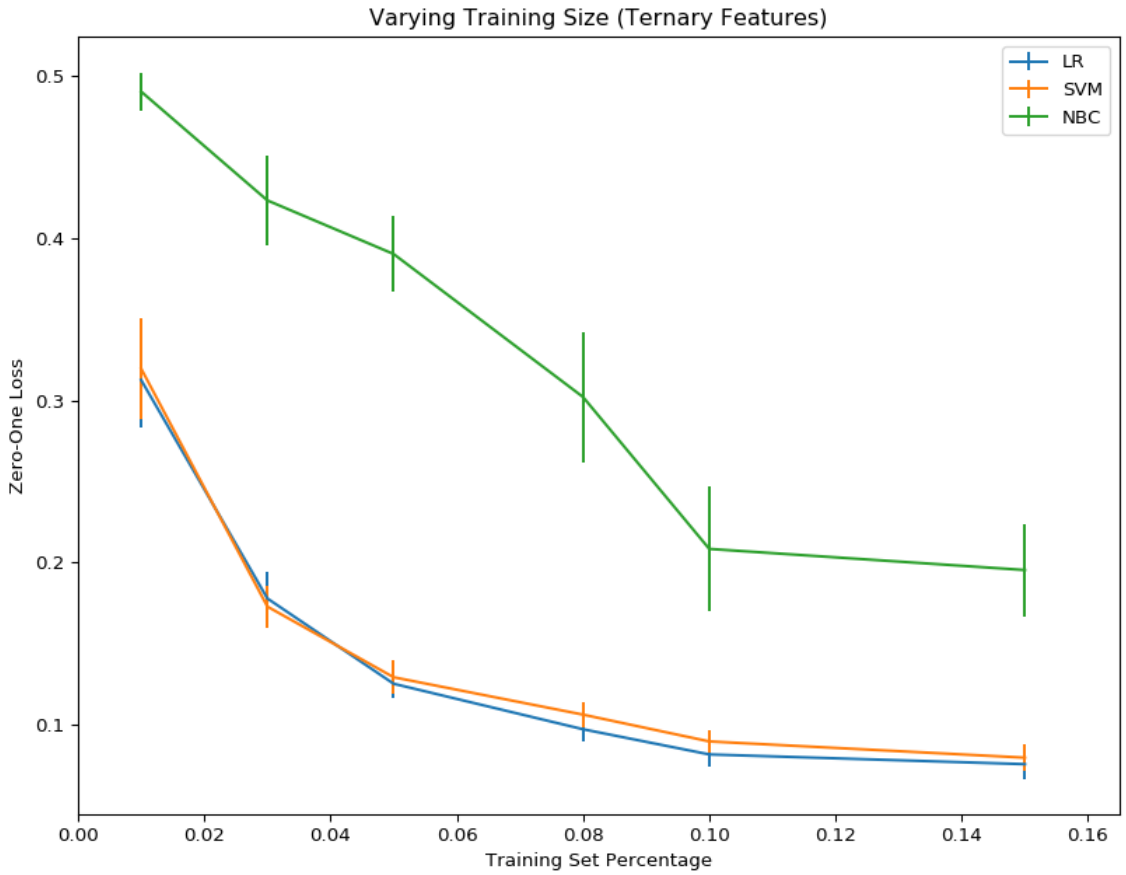

Figure 2: Varying Training Set Size with 0, 1, 2 Features.

   (b) Formulate a hypothesis about the performance difference you observe for at least one model (comparing results from this experiment and (1)).

   Let $\mu_{NBC,B}$ refer to mean zero-one loss of the Naive-Bayes Classifier using Binary Features and $\mu_{NBC,T}$ refer to the mean zero-one loss of the Naive-Bayes Classifier using Ternary Features.

   Null Hypothesis (H_0): $\mu_{NBC,B} = \mu_{NBC,T}$

   Alternative Hypothesis (H_1): $\mu_{NBC,B} < \mu_{NBC,T}$

(c) Discuss whether the observed data support the hypothesis (i.e., are the observed differences significant).

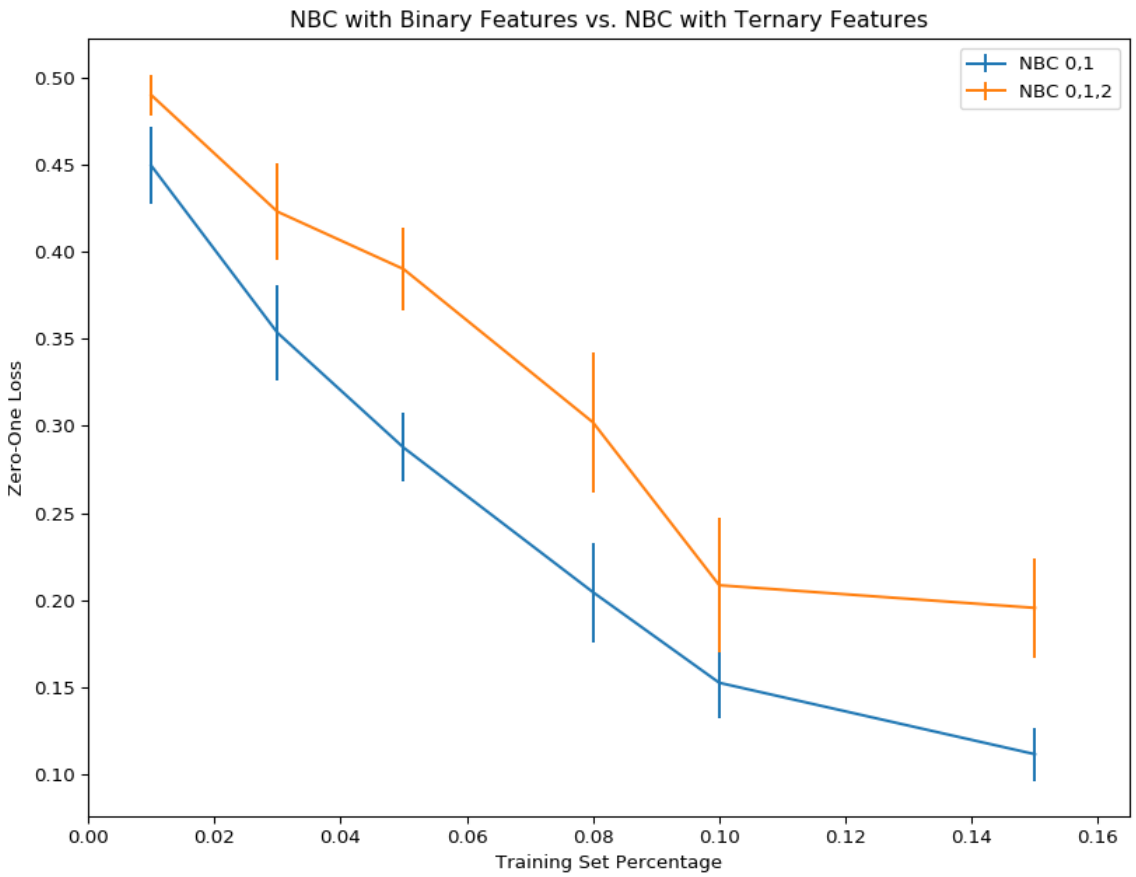

Figure 3: Comparing Naive Bayes Performance Across Different Features.

From Figure 3, we see that the Naive-Bayes classifier using 0,1,2 features has a higher 0/1 loss for all training set sizes compared to Naive-Bayes with 0,1 features. For most training sizes, the difference appears to be significant as their standard error bars do not overlap (except for training percentage 0.1).

Alternatively, we can perform a one-tailed paired t-test for each training set size. We will choose our significance $\alpha = 0.05$. In order to correct for testing multiple hypotheses, we apply Bonferroni's correction. We reject the null hypothesis if the p-value is less than $\frac{\alpha}{6} = 0.0083$.

Training Set Percentage	p-value
0.01	0.0957612399967
0.03	0.0847687002996
0.05	0.005585860394
0.08	0.0509294150103
0.10	0.132847041034
0.15	0.0166865529227

From the above table, we see that we can reject the null hypothesis only for training set percentages 0.05.

Submission Instructions:

After logging into data.cs.purdue.edu, please follow these steps to submit your assignment:

1. Make a directory named *'yourName_yourSurname'* and copy all of your files there.
2. While in the upper level directory (if the files are in /homes/neville/jennifer_neville, go to /homes/neville), execute the following command:

   turnin -c cs57300 -p HW3 your_folder_name

   (e.g. your prof would use: turnin -c cs57300 -p HW3 jennifer_neville to submit her work) Keep in mind that old submissions are overwritten with new ones whenever you execute this command.

   You can verify the contents of your submission by executing the following command:

   turnin -v -c cs57300 -p HW3

   Do not forget the -v flag here, otherwise your submission will be replaced with an empty one.

Your submission should include the following files:

1. The source code in python, named `hw3.py`.
2. Your evaluation & analysis in named `report.pdf`. Note that your analysis should include learning curve graphs as well as a discussion of results.
3. A README file containing your name, instructions to run your code and anything you would like us to know about your program (like errors, special conditions, etc).