CS 534: Machine Learning
Homework 3

(Due Oct 26th at 11:59 PM on Gradescope)

Submission Instructions:

The homework must be submitted electronically on Gradescope as a single submission. Each question must be tagged appropriately (you need to set which page the problem is on), otherwise you may lose credit. The answer must also be explicit, if you are exporting an iPython notebook, you will need to add remarks to denote your final answer for the question.

The code can be done in Python, R, or MATLAB. The use of any other programming language must be confirmed with Huan (TA). Source code must be submitted separately on Canvas (the deadline is 15 minutes after the Gradescope submission) and zipped into a single file. There is no need to attach the data, and you can assume the data will be in the same directory as the code. The source code should be executable and there should be no explicit path names for the data files (`joyce/CS534/HW3/train.csv` is an example of hard-coding).

1. **(10 pts) AdaBoost Update**

   Derive the expression for the update parameter in AdaBoost (Exercise 10.1 in HTF).

$$\beta_m = \frac{1}{2}\log\frac{1-\mathrm{err}_m}{\mathrm{err}_m}$$

2. **(4+8+2+3+6+2=25 pts) Decision Tree to Predict Loan Defaults**

   We will consider a simplified subset of the Lending Club Loan Data available on Kaggle (`https://www.kaggle.com/wendykan/lending-club-loan-data`). Using ideas from a Kaggle Kernel by Kacper Wozniak[1], we have preprocessed the dataset the following steps:

   - Removed any column that has more than 50% of the observations missing
   - Remove any row with at least one missing value
   - Removed any loans that are not Fully Paid or Charged Off (defaulted)
   - Only used individual applicants (i.e., join types are ignored)
   - Removed columns referring to "Active" loan accounts, joint applicants, or with too many unique values that preprocessing may take longer than necessary (i.e., policy code, employee title)
   - Convert the loan status to binary variable (`class`) as Fully Paid (0) vs Charged Off (1)
   - Subsample the data for a smaller dataset

   The resulting dataset (`loan_default.csv`) contains 2850 loans with 26 features. Use your favorite decision tree package, or implement your own, to predict the loan status.

   (a) Pre-processing: Split the data into a train / test split (justify your selection). Specify any additional pre-processing you performed on the dataset.

[1]The kernel is available at `https://www.kaggle.com/kacpiw/who-default-on-a-loan-logistic-regression`

(b) There is debate amongst the community about the use of pruning to control overfitting. Instead, you will tune the decision tree by limiting the depth of the decision tree and the minimum number of samples that each leaf must have (this maps to the `scikit-learn` parameters `max_depth` and `min_samples_leaf`, respectively). Specify how you determined the best hyperparameters for the decision tree and plot the estimated AUC on a single plot (either do a 3D plot or use size of the point to encode the classification accuracy) as a function of the two parameters. What is the best choice of the two parameters?

(c) Re-train a decision tree on the entire training data using the optimal max depth and minimum number of samples from (b). What is the AUC on the test set?

(d) Create a visualization of the top 3 levels of your decision tree using the optimal max depth and minimum number of samples.

(e) Analyze the effects of three different filtering methods (independently) on the decision tree[2]. After filtering the variables, re-determine the optimal parameters and report the AUC on the test set as a direct comparison to part (c).

   i. Remove "highly" correlated variables. Note what metric you decide to use as correlation, and what your threshold for high is.

   ii. Rank the features based on the correlation criteria discussed in class. Based on the correlation results, justify your selection of the top k features.

   iii. Rank the features based on the mutual information criteria. Based on the results, justify your selection of the top k features.

(f) Comment on the effect of features selection for decision trees based on your results from (c) and (e).

3. **(2+10+4+2=18 pts) Bootstrapping to Predict Loan Defaults**

   Use the pre-processed data you created from problem (2a).

   (a) Train a ridge logistic regression model (use a standard toolkit) using all the training data you created. Specify how you chose the optimal parameter λ. How well does your model do on the test data?

   (b) Using 100 bootstrap samples, train 100 different ridge logistic regression models using all the training data you created. For ease, you will want to use the same λ as (a). Compute the 95% confidence intervals for each coefficient estimate (you should have 26). Generate a plot that shoes the ridge regression estimates with error bars to denote the range (x-axis should be coefficient, y-axis should be value). How does it compare against the estimated parameters in (a)?

   (c) Using your 100 bootstrap models, generate the estimated prediction by (1) averaging the predicted response, and (2) averaging the predicted probabilities. What are the AUCs for the averaged versions? How does it compare to (a)?

   (d) Comment on the effect of bootstrapping on the predictive performance based on the results from (a) and (c).

4. **(20+10+10+5+2=47 pts) Stochastic Gradient Tree Boosting to Predict Appliance Energy Usage**

[2]It is important that you use only the training set to run these filtering methods

Consider the Energy dataset (`energydata.zip`) from Homework #1. As a reminder, each sample contains measurements of temperature and humidity sensors from a wireless network, weather from a nearby airport station, and the recorded energy use of lighting fixtures to predict the energy consumption of appliances in a low energy house. For this problem, you will implement a variant of stochastic gradient tree boost to predict the energy usage [3]. There are two major changes from the gradient tree boosting algorithm that was discussed in class that each model is dampened by a factor ν (i.e., model prediction is $y^{(t)} = y^{(t-1)} + \nu f_t(\mathbf{x}_i)$), and there is random subsampling of the dataset (hence the name stochastic).

(a) Implement the stochastic gradient tree boosting algorithm (see Algorithm 1) using an existing decision tree implementation. Your boosting model should support the following parameters:

- M: number of boosting iterations
- ν: shrinkage parameter
- q: subsampling rate

(b) Using all the training data ($q = 1$), plot the validation error as a function of the parameter $\nu \in [0, 1]$ and the number of boosting iterations (you can choose however you plan to present the information, whether using different lines, 3D plot, etc.) Note that $\nu = 0.1$ is a common parameter value of ν and maybe helpful to have in your grid search. What conclusions can you draw from the plots in terms of how the shrinkage parameter relates to the number of boosting iterations? What would be optimal parameters for $q = 1$?

(c) Tune for the best validation error including different subsampling rates ($q \in [0.6, 1]$), ν, and M. You can use part (b) to reduce the parameter search for ν and M (convince us that there might be certain values we can consider omitting and how you decided it). What are the optimal parameters for all three?

(d) For the optimal parameters you found in (b) and (c), train your "final model". What are the test errors? How does this compare to your results from homework #1?

(e) Comment on the stochastic gradient tree boosting versus gradient tree boosting in terms of computational time, performance results, and hyperparameter tuning.

Algorithm 1 Stochastic Gradient Tree Boosting with Shrinkage for Regression

```
1: Initialize $f_0(\mathbf{x})$ to target mean
2: for $m = 1 : M$ do
3:     Subsample training set at rate $q$
4:     Compute gradient residual for all $i$ in subsample, $r_{im} = -\left[\frac{\partial L(y_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x_i})}\right]_{f(\mathbf{x}_i) = f_{m-1}(\mathbf{x}_i)}$
5:     Fit tree model $h_m$ to residual $\mathbf{r}_m$
6:     Update model $f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \nu h_m$
7: end for
8: Return $f(\mathbf{x}) = f_M(\mathbf{x})$
```

[3] In gradient tree boosting, you re-calculate the weight of each terminal tree node. This is not easy to do if you use most standard tree implementations (i.e., `scikit-learn`), and thus has been omitted for implementation ease. The sacrifice is that your implementation is less likely to do as well as other packages.