EE 660 **Homework 2 (Week 3)** Posted: 9/12/2021

v2.2 Due: Mon., 9/20/2021, 11:59 PM PDT

1. *CART for regression*

   (a) For CART applied to regression, prove the relation for $w_{m'}^*$ (optimal prediction value for region $\mathcal{R}_{m'}$) given below:

   $$w_{m'}^* = \frac{1}{N_{\mathcal{R}_{m'}}} \sum_{x_i \in \mathcal{R}_{m'}} y_i$$

   [**Hint:** take the derivative of the cost function in that region, and set equal to 0.]

   (b) Also in a regression problem, for a given region $\mathcal{R}_{m'}$ containing $N_{\mathcal{R}_{m'}}$ data points, and a given feature to threshold x_j, suppose you want to find an optimal threshold value t_k by trying different values; how many values for t_k need to be tried? (Give an upper bound.) Justify your answer.

2. *Variance in a random forest*

   Let $v_i, i = 1,2,\cdots, B$, be identically distributed random variables, each with mean μ_v and variance σ_v^2. Let the average of these random variables be $s = \frac{1}{B}\sum_{i=1}^{B} v_i$.

   (a) You are given that the v_i are not independent, and have positive pairwise correlation coefficient ρ, given by

   $$\rho \triangleq \frac{\mathrm{E}\{v_i v_j\} - \mu_v^2}{\sigma_v^2}, \qquad \rho \geq 0 .$$

   Prove that the variance of *s* is:

   $$\sigma_s^2 = \rho\sigma_v^2 + (1-\rho)\frac{\sigma_v^2}{B}$$

   (b) Show that $\rho \leq 1$ always.

   (c) For a given *B* and σ_v^2, what is the smallest variance *s* could have? What is the largest variance *s* could have? What does this imply about desirable properties of decision trees in a random forest?

3. *Random Forest for yeast data classification*

This problem is intended to give some hands on experience using random forest. You are given a dataset adapted from the Yeast Data Set on UCI repository:

https://archive.ics.uci.edu/ml/datasets/Yeast

containing 1484 data points and 8 features, and has been partitioned into a training set of 1000 data points, and a testing set of 484 data points. Be sure to use the datasets posted on the D2L website, as partitioned into training and testing datasets.

The goal is to estimate the **Protein Localization Site** of each instance. The **Protein Localization Site** is a categorical label that takes 10 different values in form of strings. The provided .csv files are in the usual format for Python use.

(a) Before setting up the random forest classifier, try a couple base (trivial) classifiers for comparison, as follows. Note that both classifiers output a label prediction $\hat{y}_i$ without looking at the input feature values $\underline{x}_i$.

(i) A classifier that randomly picks an output label, with probability given by the label's frequency of occurrence in the training data set.

For example, let $\pi_c = \frac{N_c^{(Tr)}}{N^{(Tr)}}$. Then the classifier chooses label c with probability π_c, for $c = 1,2,\cdots,10$.

Run this classifier 10 times, with different seed each time, and give the resulting mean percent classification error on the training set and separately on the test set; also give the standard deviation of the percent classification error on the training set and on the test set.

(ii) A classifier that always outputs the label of the most populated class (class with the largest π_c as defined above). Give its percent classification error on the training set and separately on the test set.

The above classifiers provide examples of systems that haven't learned anything from the data features $\underline{x}$. They can be used as a basis for comparison.

(b) Then try the following experiment setting for random forest:

At each iteration, first create a smaller training set, 'bag", randomly drawn from the given training set. The size of bag means the percent of training samples that are used to grow a tree (for example, bag_size=1/3 means sampling 1/3 data from the training set). Use bag_size=1/2. You can use "train_test_split()" or any other technique for this purpose.

Then train a random forest with the "bag" set and the following parameters:

n_estimators = 1 to B, step size 1, B=30.

criterion = 'entropy'

max_depth = 5

bootstrap = True

max_features = 3

For each value of number of trees (estimators), repeat the experiment 10 times (selecting different bag samples every time), and calculate the following results:

- Mean error rate on testing set
- Mean error rate on training set
- Standard deviation of error rate on testing set
- Standard deviation of error rate on training set

Each of the 4 sets of results should be a 30 by 1 vector.

For this problem, please:

(i) Run the above setting, and plot your 4 sets of results against the number of trees as *x*-axis (one plot for mean error rates and another plot for std, 2 plots in total); report the minimum test error rate and its corresponding standard deviation. (For example, B=30, your 'mean error rate on testing set' is a 30 by 1 vector $\boldsymbol{x}$, and the corresponding std is also a 30 by 1 vector $\boldsymbol{y}$, you find that the best testing error rate is at index 28, then report $\boldsymbol{x}[28]$ and $\boldsymbol{y}[28]$.)

Also answer: how do the error rates change as a function of number of trees?

(ii) Now set B=100 and try again. Plot the figures as mentioned in (b)(i). Does the curve show convergence?

(iii) Try bag_size=$\left[\frac{1}{3}, \frac{1}{2}, \frac{2}{3}\right]$ and max_depth=$[1,5,None]$ (None means unlimited or unspecified) and max_features=$[1,3,8]$. When trying different values for a specific parameter, you can keep other parameters fixed as the values in (i). For each setting, draw a set of plots as in (i). So you'll have 9 sets of plots. Answer: how does the performance change as you adjust those parameters?

Notes:

(1) if nothing changes significantly when parameter values are changed, that could be a positive result because it means the classifier is robust (stable) to changes in the parameter values;

(2) use the standard deviation as a guide to what changes in error rate are statistically significant (for example, setting P has minimum error rate 0.41 and std 0.02, setting Q has 0.39 and std 0.02, then Q is not necessarily better than P since the difference is within one std).

(iv) Based on your results in (i) and (iii), choose a best setting. Compare it with the two trivial classifiers of part (a): has it learned from the data?

Hints:

Functions you can use:

sklearn.ensemble.RandomForestClassifier
and its method fit(). For more info, see the example in
http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

You may also find the following functions useful (feel free to use them):

sklearn.metrics.accuracy_score (to measure accuracy on training and test data)
matplotlib.pyplot (to show the results)

4. *Forward stagewise additive modeling*

   (a) Show that forward stagewise additive modeling (Eq. (16.34)) is an adaptive basis function model (ABM), by applying Eq. (16.34) iteratively, and putting it in the form of an additive basis function model (Eq. (16.3)); also give the resulting expressions for w_0, w_m, and $\phi_m(\underline{x})$ in terms of the quantities in Eq. (16.34).

   (b) If shrinkage is used in the stagewise additive modeling (Eq. (16.35) instead of Eq. (16.34)), show that this also be an additive basis function model, and give expressions for w_0, w_m, and $\phi_m(\underline{x})$ in terms of the quantities in Eq. (16.35).

5. *Boosting.* For the Adaboost.M1 algorithm:

   (a) Plot α_m vs. err_m, for $0 < \text{err}_m < 1$.

   (b) Let the weight update for data point *i* at iteration *m* be represented by:

   $$w_{i,m+1} = g_m w_{i,m}$$

   in which *g* denotes a gain factor. Plot g_m vs. err_m for $0 < \text{err}_m < 1$, for data points misclassified by ϕ_m. Also plot (or state the value of, if constant) g_m for data points correctly classified by ϕ_m.

   (c) If we require each base classifier ϕ_m to have a weighted error $\text{err}_m < 0.5$, then what is the resulting range of α_m? Of β_m? Of g_m for misclassified points?