EE 660 **Homework 2 (Week 5)** Posted: 9/23/2022

Due: Fri., 9/30/2022, 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:

   $$w^*_{m'} = \frac{1}{N_{\mathcal{R}_{m'}}} \sum_{\underline{x}_i \in \mathcal{R}_{m'}} y_i$$

   in which $N_{\mathcal{R}_{m'}}$ = number of data points in $\mathcal{R}_{m'}$.

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

   (b) Now let each data point $(\underline{x}_i, y_i)$ be associated with an importance value $v_i \geq 0$. The new cost function for region $\mathcal{R}_{m'}$ is:

   $$\text{cost}\,(\mathcal{R}_{m'}) = \sum_{\underline{x}_i \in \mathcal{R}_{m'}} v_i (y_i - w_{m'})^2$$

   Derive the optimal predicted value $w^*_{m'}$.

   (c) What algorithm or method would (b) be useful for? Be as specific as you can. A few words should be sufficient for your answer.

   Hint: Lecture 7 notes.

2. *Random Forest for Wifi localization*

   This problem is intended to give some hands on experience using random forest. You are given a dataset adapted from the WiFi user localization dataset :

   http://archive.ics.uci.edu/ml/datasets/Wireless+Indoor+Localization

   containing 2000 data points and 7 features, and has been partitioned into a training set of 1500 data points, and a testing set of 500 data points. Be sure to use the datasets posted on the D2L website, as partitioned into training and testing datasets.

   The goal is to use the WiFi signal strength received from the 7 routers, to predict the location of a user (who is in one of 4 rooms). Thus, there are 7 input variables (features) and 4 classes (user locations).

   The provided .csv files are in the usual format for Python use. No need to standardize the data.

(a) Before setting up the random forest classifier, we want to estimate the error of 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 for each prediction,

(ii) A classifier that always decides the same class (e.g. $\hat{y}_i = 2$).

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

For each of (i) and (ii) above, answer:

- What is the theoretical expected error?
- Use python to simulate the above classifiers. What are the resulting train and test errors?

(b) Then try a random forest classifier.

Use sklearn.ensemble.RandomForestClassifier with the following settings:

n_estimators = 1 to B, step size 5, B=101.
Max_samples (bag size) = 0.8
criterion = 'gini'
max_depth = 5
bootstrap = True
max_features = 2

Comment: bag size is the fraction of data points that are drawn randomly from the dataset to train each tree. bootstrap = True means it will do this random drawing of data.

For each value of number of trees (estimators), train the classifier 10 times, 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 21 by 1 vector.

For this problem, please:

(i) Run the above settings and plot your 4 sets of results against the number of trees as *x*-axis (plot mean error rates and std, 4 "curves" in total in the same plot). Report the minimum test error rate and its corresponding standard

deviation. (For example, B=21, your 'mean error rate on testing set' is a 21 by 1 vector E_{Test} , and the corresponding std is also a 21 by 1 vector E_{Std} . If you find that the best testing error rate is at index 18, then report $E_{\text{Test}}[18]$ and $E_{\text{Std}}[18]$.)

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

(ii) Now set max_features=[3,4] and try again. Plot the results and answer the questions as described in (b)(i) (2 plots, each with 4 "curves" for (b)(ii)). Also, compare the curves for max_features = 2, 3, and 4.

(iii) Finally try max_features = 7 (1 plot, 4 "curves" for (b)(iii)). Since we are using the total number of features, this method is equivalent to bagging. How does the bagging-only result compare with the random-forest results of (b)(i) and (ii)?

Tip: 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), (ii), and (iii), choose a best setting for the number of features and the number of trees. Train a model with these settings and change the maximum tree depth starting from max_depth = 2 to 25 with step 2. For each value of number of trees (estimators), train the classifier 10 times and calculate the train and test average error and standard deviation of each. How do the error rates change as a function of maximum tree depth? Plot each of these error metrics vs. max_depth, to give 4 "curves" on one plot.

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

Tip: use your measured test-error to answer question (v). No need to consider generalization-error bounds in this problem (that will be covered on a future homework).

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)

3. *Optimization in Adaboost*

(a) Starting from the Adaboost cost function at the m^{th} iteration (Lecture 7, top of p. 9 and Eq. 6):

$$L_m(\beta_m', \phi_m') = \sum_{i=1}^{N} w_{i,m} \exp\left[-\tilde{y}_i \beta_m' \phi_m'\right], \ \beta_m' > 0$$

in which we have used the shorthand notation $\phi_m' = \phi\left(\underline{x}_i, \underline{\gamma}_m'\right)$, derive an easily interpretable expression for the ϕ_m' that minimizes L_m . Your final expression should be in the form:

$$\phi_m = \text{argmin}_{\phi_m'} \sum_{i=1}^{N} A_{im} \ \text{II}(\tilde{y}_i \neq \phi_m')$$

In which II (..) denotes the indicator function of (..), and A_{im} is a coefficient that is a constant of β_m' and ϕ_m' . Please state what A_{im} is, in terms of given quantities.

Comment: Murphy does this but only gives one or two of the steps (an outline version); you are to show all the steps of the derivation.

Tip: First try to get L_m into the form:

(1) $L_m = \sum_{i=1}^{N} w_{i,m} \text{II}(\tilde{y}_i = \phi_m') e^{-\beta_m'} + \sum_{i=1}^{N} w_{i,m} \text{II}(\tilde{y}_i \neq \phi_m') e^{+\beta_m'}$

(b) Interpret in words the meaning of your result of part (a).

(c) Derive the optimal β_m' algebraically, starting from:

$$\beta_m = \text{argmin}_{\beta_m'} L_m(\beta_m', \phi_m)$$

in which we have substituted the optimal ϕ_m (obtained from (a)) for ϕ_m'.

Hint: L_m is a differentiable function of β_m' .

Show that your answer can be written in the form:

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

and give your expression for err_m in terms of given quantities.

(d) Is your err_m in (c) the same as that given in lecture?