CMSC 25025 / STAT 37601

Machine Learning and Large Scale Data Analysis

Assignment 6 Due: Tuesday June 4, 2019 at 2:00 pm.

Please hand in this Homework in 5 files:
1. A pdf of your solution to problems 1 and 2.
2. A pdf of your jupyter notebook for problem 3.
3. The ipynb file for problem 3.

1. *Multiclass boosting* (20 points)

   In this problem we show how the reweighting for boosting is derived from a particular optimization problem. If g_m is the m'th classifier, the weight update scheme for step m of boosting is given by

$$\widetilde{w}_i = \begin{cases} (C-1)\frac{1-e_m}{e_m} w_i & X_i \text{ misclassified by } g_m \\ w_i & \text{otherwise} \end{cases},$$
$$w_i^{new} = \frac{\widetilde{w}_i}{\sum_i \widetilde{w}_i}. \tag{1}$$

   Assume we have C classes and we code class c as:

$$Y = \mathcal{Y}_c = (\underbrace{-\frac{1}{C-1}, \ldots, 1, \ldots, -\frac{1}{C-1}}_{\uparrow\, c}),$$

   and assume we create multiclass classifiers $g(X)$ that produce an output $g(X) = \mathcal{Y}_c$ if $g(X) = c$.

   (a) Show that if a point is correctly classified $Y^t g(X) = \frac{C}{C-1}$, and if it is misclassified $Y^t g(X) = -C/(C-1)^2$.

   (b) Define the following loss on a collection $g_1, \ldots, g_m$ of m classifiers:

$$L(X, Y, \beta_1, \ldots, \beta_m, g_1, \ldots, g_m) = \sum_{i=1}^{n} \exp\left[-\frac{1}{C}\left(\beta_1 Y_i^t g_1(X_i) + \beta_2 Y_i^t g_2(X_i) + \ldots + \beta_m Y_i^t g_m(X_i)\right)\right] \tag{2}$$

   Assume we've already trained the classifiers $g_1, \ldots, g_{m-1}$ and we want to find the best 'next' classifier g_m. So we want to compute

$$\begin{aligned}\beta_m, g_m &= \arg\min_{\beta,g} \sum_{i=1}^{n} \exp\left[-\frac{1}{C}\left(\sum_{r=1}^{m-1} \beta_r Y_i^t g_r(X_i) + \beta_m Y_i^t g_m(X_i)\right)\right] \\ &= \arg\min_{\beta,g} \sum_{i=1}^{n} \widetilde{u}_i \exp\left[-\frac{1}{C}\left(\beta_m Y_i^t g_m(X_i)\right)\right],\end{aligned}$$

where $\widetilde{u}_i = \exp\left[-\frac{1}{C}\left(\sum_{r=1}^{m-1} \beta_r Y_i^t g_r(X_i)\right)\right]$. Denote

$$e_m(g) = \sum_{i=1}^{n} u_i \mathbf{1}[Y_i^t g(X_i) < 0].$$

Show that g_m that minimizes the above loss satisfies:

$$g_m = \arg\min_{g} \sum_{i=1}^{n} \widetilde{u}_i \mathbf{1}[Y_i^t g(X_i) < 0] = \arg\min_{g} \sum_{i=1}^{n} u_i \mathbf{1}[Y_i^t g(X_i) < 0] \doteq \arg\min e_m(g),$$

where $u_i = \widetilde{u}_i / \sum_i \widetilde{u}_i$. In other words g_m is chosen to minimize the weighted error rate, and show that

$$\beta_m = \frac{(C-1)^2}{C}\left[\log\left(\frac{1-e_m}{e_m}\right) + \log(C-1)\right].$$

where $e_m = \min_g e_m(g)$

Hint: For fixed β write the sum over the losses as

$$\sum_{i:Y_i^t g(X_i)>0} u_i \exp[-\beta/(C-1)] + \sum_{i:Y_i^t g(X_i)<0} u_i \exp[\beta/(C-1)^2].$$

(c) The new weights for the next classifier will now be

$$\widetilde{u}_i^{new} = \widetilde{u}_i \exp\left[-\frac{(C-1)^2}{C^2}\left(\log\frac{1-e}{e} + \log(C-1)\right) Y_i^t g_m(X_i)\right],$$

and $u_i = \widetilde{u}_i^{new} / \sum_i \widetilde{u}_i^{new}$.

Show inductively that $u_i = w_i$ defined in (1).

Thus the reweighting scheme for boosting is equivalent to minimizing the exponential loss in (2).

Note that we never find g_m that actually minimzes the weighted error loss, we find some classifier based on the weighted error for example by growing a tree.

2. *EM* (30 points)

Let $X = (X_1, \ldots, X_d)$ be a random vector where $X_j \in \{0, 1, 2\}$. Let Y be a random variable taking values in the set $\{1, \ldots, d-3\}$. Write $P(Y = k) = \pi_k, k = 1, \ldots, d-3$. Conditional on Y we assume the components of X are independent with the following distribution:

$P(X_j = r|Y = k) = a_r, r = 0, 1, 2$, for $j = k, k+1$.

$P(X_j = r|Y = k) = b_r, r = 0, 1, 2$, for $j = k+2, k+3$.

$P(X_j = r|Y = k) = c_r, r = 0, 1, 2$, for all $j < k$ and $j > k+3$.

(a) If a_2, b_1, c_0 are close to 1 describe what X would look like for different values of Y. Try to do it visually. What do the variables Y represent.

(b) Write the log-joint distribution - $\log P(X_1, \ldots, X_d, Y)$.

(c) If you had a fully observed sample $X_i, Y_i, i = 1, \ldots, n$ what would be the maximum likelihood estimates of $a, b, c, \pi_k, k = 1, \ldots, d-3$. Remember that $a_2 = 1 - a_0 - a_1$. If you can't derive it analytically try to come up with an intuitive solution and explain.

(d) Write pseudo code for the EM iterations to estimate these parameters if Y_i are unobserved.

3. *Multiple trees on MNIST* (50 points)

In python use the following two functions:

```
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
```

Using the mnist dataset split the data into 50,000 training, 10000 validation, 10000 test.

(a) With the function `RandomForestClassifier` experiment with various stopping rules

`min_samples_split` - minimum number of samples in node to allow it to split. `max_features` - number of random features from which to select the best, and `n_estimators`- number of trees. Also use the option `criterion="entropy"`.

Setting up the classifier is done with this command:

```
clf = RandomForestClassifier(n_estimators=num_trees,
min_samples_split=...,criterion="entropy",
                             max_features=...)
```

Then you can call the functions `clf.fit`, `clf.predict`, and `clf.predict_proba` - which gives the output average probabilities.

Plot error rates on both training and validation data as a function of the number of trees.

Once you have found the best protocol check it on the test set.

(b) The function `AdaBoostClassifier` implements SAMME, the boosting protocol described in problem 1. You define a classifier `clfb` (for example `RandomForestClassifier` with 1 tree), and pass it on to the function

```
AdaBoostClassifier(clfb,n_estimators=...,algorithm='SAMME')
```

Experiment with the same three parameters in this setting (stopping rule and number of splits in the definition of `clfb`) and number of trees. Compare the results to the random forest results in part (a).