For HW 8 Boosting

Zishen Xu (TA)

The algorithm for the homework on logit boost

1) Data pre-processing:
   - Add a column of 1s to the left of the design matrices (both X_{train} and X_{test});
   - Change the labels to -1 and 1 (both Y_{train} and Y_{test});

   (Note that you don't need to normalize (remove mean and divide standard deviation) this time)
   - Denote N=number of rows for X_{train} and M= number of columns for X_{train} (after adding the column of 1s);

2) The iteration:
   - Initialize:

     β_{all} = an M*1 vector of 0s;

     loss = an empty vector that has 300 (max of k values given in the assignment) elements;

     $error_{train}$ and $error_{test}$ each is an empty vector that has 4 (number of k values given in the assignment) elements;

     $ik = 1$;
   - Loop i from 1 to 300 (max of k values given in the assignment, this loops over the number of weak classifiers)
     - Calculate current H(X), the overall function values on X:

       $$H = X_{train}\,\beta_{all}$$

       (matrix product, return a N*1 vector)
     - Calculate $p(x_i)$, w_i and z_i:

       $$p = \frac{1}{1 + e^{-2*H}}$$

       $$w = p * (1 - p)$$

       $$z = \frac{\frac{1}{2}(Y_{train} + 1) - p}{w}$$

       If w has 0 elements, then the corresponding z elements are 0;

       Note that all calculations in this step are made element-wise, thus all p, w and z are N*1 vectors.

- Initialize: coef = an empty 2*(M-1) matrix and newloss = an empty (M-1)*1 vector.
- Loop j from 1 to (M-1) (this loops over the number of features, i.e. non-intercept columns)
  - Obtain the j-th feature values for each sample (N*1 vector):
    $$X_j = X_{train}[:, j+1]$$
  - Do weighted least square (WLS) estimation on $\{X_j, z\}$ with weight w manually:
    $$a = sum(w)$$
    $$b = sum(w * X_j)$$
    $$c = sum(w * X_j{}^2)$$
    $$d = sum(w * z)$$
    $$e = sum(w * X_j * z)$$
    All the multiplication (*) and square (^{2}) are done element-wise (all X_j, z, w are N*1, result a to e are all scalers).
    Then obtain the WLS estimation (2*1 vector):
    $$\beta_j = \frac{1}{a * c - b^2}\begin{bmatrix} c * d - b * e \\ a * e - b * d \end{bmatrix}$$
    If $a * c - b^2 = 0$, we have:
    $$\beta_j = \begin{bmatrix} \frac{d}{a} \\ 0 \end{bmatrix}$$
  - Calculate new H(X) if β_j is chosen to be the update for the i-th iteration (N*1 vector):
    $$H_j = H + \frac{1}{2}\left(\beta_j[1] + \beta_j[2] * X_j\right)$$
    Note here $\beta_j[1], \beta_j[2]$ are scaler, X_j and H are N*1 vectors.
  - Calculate new loss function value if if β_j is chosen to be the update for the i-th iteration (scaler):
    $$Loss_j = sum[\log(1 + e^{-2 * Y_{train} * H_j})]$$
  - Record β_j and $Loss_j$:
    $$coef[:, j] = \beta_j$$
    $$newloss\ [j] = Loss_j$$
- Sort newloss to get the smallest element's position:
  $$\hat{J} = argmin(newloss)$$

- Update β_{all} and loss accordingly:

$$\beta_{all}[1] = \beta_{all}[1] + \frac{1}{2} coef\left[1, \hat{J}\right]$$
$$\beta_{all}\left[\hat{J}+1\right] = \beta_{all}\left[\hat{J}+1\right] + \frac{1}{2} coef\left[2, \hat{J}\right]$$
$$loss[i] = newloss\left[\hat{J}\right]$$

- When i is in {10, 30, 100, 300}, calculate misclassification error:
  - Make prediction (sign[H(X)]):

$$\hat{Y}_{train} = sign(X_{train}\, \beta_{all})$$
$$\hat{Y}_{test} = sign(X_{test}\, \beta_{all})$$

  - Calculate error rate:

$$error_{train}[ik] = mean\left(\hat{Y}_{train} \neq Y_{train}\right)$$
$$error_{test}[ik] = mean\left(\hat{Y}_{test} \neq Y_{test}\right)$$
$$ik = ik + 1$$

- When i is 300, plot the loss value vs number of iteration:

Plot(1:300, loss)

3) Make output table and plot
   - Make the table of misclassification error:

No. Iteration	Train Error	Test Error
10	$error_{train}[1]$	$error_{test}[1]$
30	$error_{train}[2]$	$error_{test}[2]$
100	$error_{train}[3]$	$error_{test}[3]$
300	$error_{train}[4]$	$error_{test}[4]$

   - Make plot of misclassification error vs No. iteration:

Plot({10, 30, 100, 300}, $error_{train}$)
Plot({10, 30, 100, 300}, $error_{test}$)

After your code reaches here, you have done the logit-boost homework.
Check whether your result is reasonable or not and do the debugging.

Q&A

(You don't need to understand what I'm saying below if you find it hard to follow)

1) Why the loss and update on overall classifier have different formulas from the assignment file? In the assignment file, there is no 2* and ½.
   Ans:
   The formulas in the assignment file and lecture note are different / not consistent and here the algorithm uses the one in the lecture note. These two will lead to the same result but if you use the formulas in the assignment file calculation for p should remove the 2* operation. To see why it is true, firstly you need to know there is a serious typo in the lecture note on slide 27:

   - Logistic Loss

$$\lambda(y, h(\mathbf{x})) = -\log(1 + e^{-2yh(\mathbf{x})})$$

   There shouldn't be a minus sign on the left. Actually this one is the log-likelihood rather than the loss and for the loss that simply takes negative log-likelihood values, the correct one should be without the minus sign. Below is what is stated in the original paper:

Let $y^* = (y + 1)/2$, taking values 0, 1, and parametrize the binomial probabilities by

$$p(x) = \frac{e^{F(x)}}{e^{F(x)} + e^{-F(x)}}.$$

The binomial log-likelihood is

$$l(y^*, p(x)) = y^* \log(p(x)) + (1 - y^*)\log(1 - p(x)) \quad (27)$$
$$= -\log(1 + e^{-2yF(x)}).$$

   From here we see another serious issue for the lecture note: the labels used are {-1,1}. There is a procedure to transform to {0,1} labels when calculating p on slide 28. Therefore, **the logit-boost in the lecture note is based on labels to be {-1,1}**, and **the y_i below should be $y_i^* = (y_i + 1)/2$**:

   - Repeat m times:
     1. Compute $p(\mathbf{x}_i) = \frac{e^{h(\mathbf{x}_i)}}{e^{h(\mathbf{x}_i)} + e^{-h(\mathbf{x}_i)}}$

$$w_i = p(\mathbf{x}_i)(1 - p(\mathbf{x}_i))$$
$$z_i = \frac{y_i - p(\mathbf{x}_i)}{p(\mathbf{x}_i)(1 - p(\mathbf{x}_i))}$$

After learning these, let's come back to the inconsistency between lecture note and assignment. Below is the algorithm from the original paper:

LogitBoost (two classes)

1. Start with weights $w_i = 1/N$ $i = 1, 2, \ldots, N$, $F(x) = 0$ and probability estimates $p(x_i) = \frac{1}{2}$.
2. Repeat for $m = 1, 2, \ldots, M$:

   (a) Compute the working response and weights

$$z_i = \frac{y_i^* - p(x_i)}{p(x_i)(1 - p(x_i))},$$

$$w_i = p(x_i)(1 - p(x_i)).$$

   (b) Fit the function $f_m(x)$ by a weighted least-squares regression of z_i to x_i using weights w_i.

   (c) Update $F(x) \leftarrow F(x) + \frac{1}{2} f_m(x)$ and $p(x) \leftarrow (e^{F(x)})/(e^{F(x)} + e^{-F(x)})$.

3. Output the classifier $\text{sign}[F(x)] = \text{sign}[\sum_{m=1}^{M} f_m(x)]$.

Since initial value of F is set to 0, we have:

$$F(x) = \frac{1}{2} \sum_{m=1}^{M} f_m(x)$$

If we make a change below and use T to replace F:

$$T(x) = 2 * F(x) = \sum_{m=1}^{M} f_m(x)$$

Then the algorithm will become:

- Update: $T(x) \leftarrow T(x) + f_m(x)$
- $p(x) \leftarrow \frac{\left(e^{\frac{T(x)}{2}}\right)}{e^{\frac{T(x)}{2}} + e^{-\frac{T(x)}{2}}} = \frac{1}{1+e^{-T(x)}}$
- Loss function: $\lambda[y, T(x)] = \log\left[1 + e^{-2*y*\frac{T(x)}{2}}\right] = \log\left[1 + e^{-y*T(x)}\right]$

Thus at this point, we can see that the assignment file just uses T(x) to replace F(x) in the algorithm, as stated below:

on the training set $D = \{(\mathbf{x}_i, y_i), i = 1, ..., N\}$, with $y_i \in \{0, 1\}$:

$$L = \sum_{i=1}^{N} \ln(1 + \exp[-\tilde{y}_i h(\mathbf{x}_i)]) \quad (1)$$

where $h(\mathbf{x}) = h_1(\mathbf{x}) + ... + h_k(\mathbf{x})$ is the boosted classifier and $\tilde{y}_i = 2y_i - 1$ take values ± 1. Please note that the Logitboost algorithm presented in class uses $y_i \in \{0, 1\}$.

(see the loss function and update on h(x), the same as the T(x) above)

However the difference in p is not mentioned and the label type {0,1} creates more ambiguities. Therefore my algorithm follows the original paper's symbols and expressions which I think is more clear than the assignment's.

2) I don't understand what you are doing for the beta, beta_j .etc.
Ans:
First of all, you need to understand what the homework means by 1D regressor. The model is simply below:

$$z = \beta_1 x_j + \beta_0$$

Where (x,z) is an observation with z=scaler and x=M*1 vector.
In other words, this is just a regression of z on x which only uses one element of x (simple linear regression).
By the way, logit-boost is a little confusing that, although each step we are adding 'weak classifier' h(x), actually those are not classifier – they don't return labels but numerical values. In the end the overall classifier H(x) also is not 'classifier'. Only sign[H(x)] can be said 'classifier'.
After understanding what the weak classifier is, based on the algorithm, we know that we should find the best WLS on x, z with weight w, according to the change in loss. This means we need to fit each column of X to get the beta. Then using the beta, we can calculate the loss value. After obtaining all the loss values, we are able to make comparison and know which one is the 'best' WLS. This is what is done in the inside loop.
For the overall classifier, as stated above, it is just the half of all weak classifiers' summation, thus it is still a linear function on x, which can be represented by the slope parameter. See it in this way:
The i-th weak classifier is:

$$h(x, \beta_{i0}, \beta_{i1}) = \beta_{i1} x_j + \beta_{i0}$$

Let $\beta_i = [\beta_{i0}, 0,0 \dots \beta_{i1} \dots 0,0 \dots 0]^T$ with only the 1st and (j+1)-th element to be nonzero, i.e.

$$\beta_i[t] = \begin{cases} \beta_{i0} & t = 1 \\ \beta_{i1} & t = j+1 \\ 0 & otherwise \end{cases}$$

then

$$h(x, \beta_{i0}, \beta_{i1}) = h(x, \beta_i) = [1, x^T]\, \beta_i$$

Note that x is the feature for one observation, so it is a M*1 vector and after adding 0s, β_i is (M+1)*1 vector. $[1, x^T] = [1, x_1, x_2 \dots x_M]$ is a 1*(M+1) vector.

Then we have for the overall classifier:

$$H(x) = \frac{1}{2}\sum_{i=1}^{Niter} h(x, \beta_i) = \frac{1}{2}\sum_{i=1}^{Niter} [1, x^T]\,\beta_i = [1, x^T]\frac{1}{2}\sum_{i=1}^{Niter} \beta_i$$

So we can see that the overall beta is just:

$$\beta_{all} = \frac{1}{2}\sum_{i=1}^{Niter} \beta_i$$

Each iteration it is updated by:

$$\beta_{all} \leftarrow \beta_{all} + \frac{1}{2}\beta_i$$

Which is equivalent to:

$\beta_{all}[1] \leftarrow \beta_{all}[1] + \frac{1}{2}\beta_i[1]$ and $\beta_{all}[j+1] \leftarrow \beta_{all}[j+1] + \frac{1}{2}\beta_i[j+1]$

To this point I think you can understand the idea of the whole algorithm.

3) What is WLS? Why we need to do it manually?

Ans:

Weighted least square is a special case of linear regression, for the classical linear regression model:

$$y = X\beta + e$$

Where X=N*M design matrix, y=N*1 response, β=M*1 parameters and e=N*1 random error. If e is assume to be:

$$e \sim N(0, W^{-1})$$

Where W is diagonal: $W = diag[w_1, w_2 \ldots w_N]$. W is called weight matrix. The solution is the following:

$$\hat{\beta} = (X^T W X)^{-1} X^T W y$$

To learn more, ask other students who have taken STA5167 or search online.

For why to calculate manually, the main reason is computation cost. Matrix inverse will cost a lot of time to implement and in the algorithm, we need to do it for each column of X during each outer loop (also the data we use all have large number of columns). However, since we only need to do simple linear regression in the algorithm, which means the matrix to take inverse is just 2*2, we can easily get the result by hand. Thus in this way we can save a huge amount of time. (current algorithm already runs for a long time for more than 10 min, you can have a try how long it will take if using matrix form calculation)

4) Where can I find the original paper?
   Check the reference page slide of the lecture note. This time I have uploaded the original paper to Canvas/Files/unfiled.