COMS 4771: Machine Learning Due: Apr. 5, 2016

Homework 3 - Solutions

Instructor: *Satyen Kale*

Part 1

See MATLAB code for full implementation. If y is the vector of labels of the examples, and w is the vector of their weights, then we can compute the cost of labeling a leaf 1 and the cost of labeling a leaf 0 as $c_0 = \langle y, w \rangle$ and $c_1 = \langle (1-y), w \rangle$, where $1 - y$ is the vector with the labels flipped.

Pseudocode:

Data: Training data $X \in \mathbb{R}^{n \times d}$ with corresponding weights $w \in \mathbb{R}^n$ and labels $Y \in \{0,1\}^n$
Result: Index j of decision stump feature, threshold t for that feature, class for left child y_l, class for right child y_r

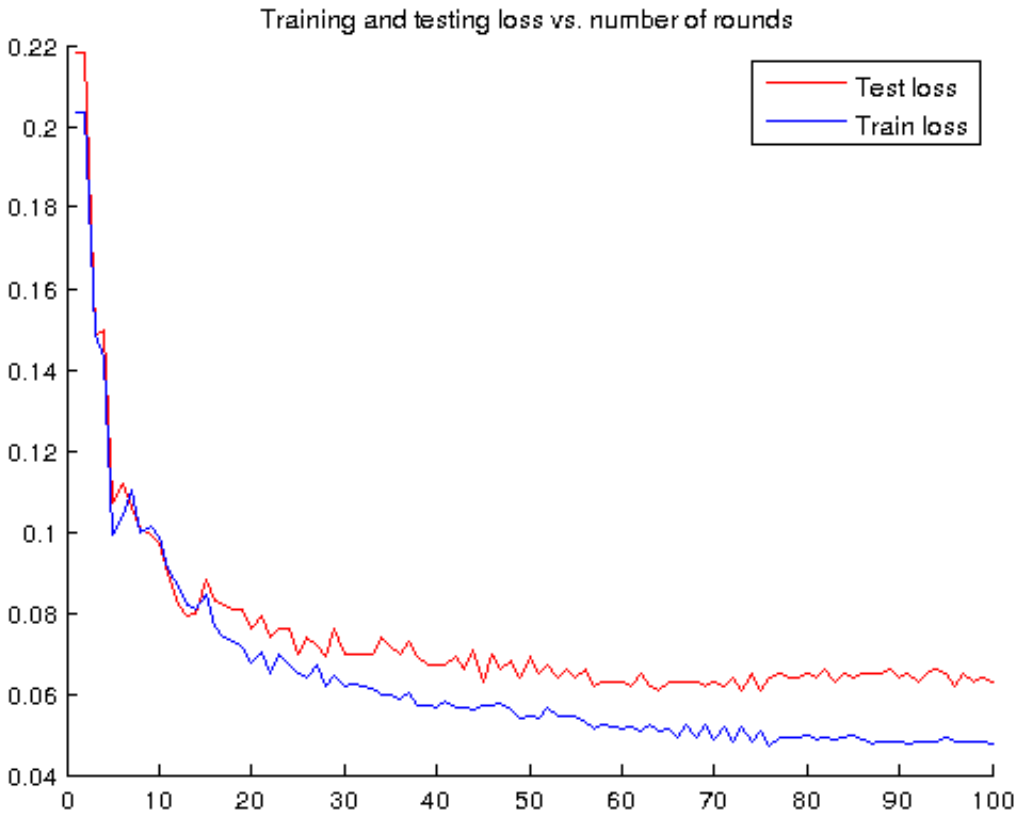
func `findLabel(`y, w`)`
  Let cost $c_1 = \langle y, w \rangle$, $c_1 = \langle (1-y), w \rangle$.
  if $c_1 < c_0$ **then**
    return $(1, c_1)$
  else
    return $(0, c_0)$
  end
end
func `train(`X, Y, w`)`
  Initialize $j = 1, t = -\infty, c = \infty$
  for *each feature* $i \in [1, d]$ **do**
    $t_i = -\infty, c_i = \infty$
    for *each potential split along feature* i *with threshold* t_i **do**
      Let w_l, Y_l and w_r, Y_r represent weight and label vectors of the left and right sides of the split, respectively.
      $y_{il}, c_l =$ `findLabel(`Y_l, w_l`)`
      $y_{ir}, c_r =$ `findLabel(`Y_r, w_r`)`
      if $c_l + c_r < c$ **then**
        $c = c_l + c_r, t = t_i, i = j, y_l = y_{il}, y_r = y_{ir}$
      end
    end
  end
  return j, t, y_l, y_r
end

Part 2

See MATLAB code for full implementation.

Results of computing training and test error with the uncertainty measure of classification error on the `spam_data.mat` dataset:

num_rounds	training error rate	test error rate
1	0.2033	0.2180
2	0.2033	0.2180
4	0.1427	0.1500
8	0.0997	0.1010
16	0.0766	0.0830
32	0.0619	0.0700
64	0.0508	0.0610
100	0.0478	0.0630



Part 3

Polynomial regression:

We take the class of regression functions

$$\mathcal{F}_d = \{c_0 + c_1 x + c_2 x^2 + \ldots + c_d x^d | c_0, c_1, \ldots, c_d \in \mathbb{R}\}$$

We note that, for any feature $x \in \mathbb{R}$, we can construct the expansion $\phi(x) = \langle 1, x, x^2, \ldots, x^d \rangle^\top \in \mathbb{R}^{d+1}$

This allows us to rewrite the problem as an ordinarly least-squares linear regression problem, as follows:

$$y_i = \langle \phi(x_i), w \rangle + \varepsilon_i$$

where ε_i represents the error $y_i - \langle \phi(x_i), w \rangle$.

We use the notation $\mathbf{y}$ for all labels and $\mathbf{X}$ for all the matrix of all expanded examples, i.e. the matrix whose i-th row is $\phi(x_i)^\top$.

The optimal coefficients $w^* = \langle c_0, c_1, \ldots, c_d \rangle^\top$ can then be found as follows:

$$w^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

using the same closed-form solution from the normal least-squares regression problem.

We note additionally that $\mathbf{X}$ is a Vandermonde matrix , i.e. a matrix with the terms of a geometric progression in each row (`https://en.wikipedia.org/wiki/Vandermonde_matrix`). Thus, as long as more than $d+1$ of the examples x_i are distinct, this solution is well-defined. This holds for $d < n$, which makes sense – we need at least as many points as the degree of the polynomial to fully constrain the regression problem.

Extra credit problem 1

First, we note that $D_{t+1}(\mathbf{x}, y) = \frac{1}{Z} D_t(\mathbf{x}, y) \cdot \exp(-\alpha_t y f_t(\mathbf{x}))$, where $\alpha_t = \frac{1}{2} \ln(\frac{1+z_t}{1-z_t})$ with $z_t = \sum_{(\mathbf{x},y) \in S} D_t(\mathbf{x}, y) \cdot y f_t(\mathbf{x})$, and $Z = \sum_{(\mathbf{x},y) \in S} D_t(\mathbf{x}, y) \cdot \exp(-\alpha_t y f_t(\mathbf{x}))$ is the normalization factor. Define

$$S_+ = \{(\mathbf{x}, y) \in S \mid y = f_t(\mathbf{x})\} \text{ and } S_- = \{(\mathbf{x}, y) \in S \mid y \neq f_t(\mathbf{x})\}.$$

For $(\mathbf{x}, y) \in S_+$, we have $y f_t(\mathbf{x}) = 1$, and for $(\mathbf{x}, y) \in S_-$, we have $y f_t(\mathbf{x}) = -1$. Furthermore, we have

$$z_t = \sum_{(\mathbf{x},y) \in S} D_t(\mathbf{x}, y) \cdot y f_t(\mathbf{x}) = \sum_{(\mathbf{x},y) \in S_+} D_t(\mathbf{x}, y) - \sum_{(\mathbf{x},y) \in S_-} D_t(\mathbf{x}, y).$$

Since $\sum_{(\mathbf{x},y) \in S_+} D_t(\mathbf{x}, y) + \sum_{(\mathbf{x},y) \in S_-} D_t(\mathbf{x}, y) = 1$, we conclude that $\sum_{(\mathbf{x},y) \in S_+} D_t(\mathbf{x}, y) = \frac{1}{2}(1 + z_t)$, and $\sum_{(\mathbf{x},y) \in S_-} D_t(\mathbf{x}, y) = \frac{1}{2}(1 - z_t)$.

Now, we have

$$\begin{aligned}
\sum_{(\mathbf{x},y)\in S} D_{t+1}(\mathbf{x},y)\cdot y f_t(\mathbf{x}) &= \frac{1}{Z}\sum_{(\mathbf{x},y)\in S} D_t(\mathbf{x},y)\cdot\exp(-\alpha_t y f_t(\mathbf{x}))\cdot y f_t(\mathbf{x}) \\
&= \frac{1}{Z}\sum_{(\mathbf{x},y)\in S_+} D_t(\mathbf{x},y)\cdot\exp(-\alpha_t) - \frac{1}{Z}\sum_{(\mathbf{x},y)\in S_-} D_t(\mathbf{x},y)\cdot\exp(\alpha_t) \\
&= \frac{1}{Z}\sum_{(\mathbf{x},y)\in S_+} D_t(\mathbf{x},y)\cdot\sqrt{\frac{1-z_t}{1+z_t}} - \frac{1}{Z}\sum_{(\mathbf{x},y)\in S_-} D_t(\mathbf{x},y)\cdot\sqrt{\frac{1+z_t}{1-z_t}} \\
&= \frac{1}{2Z}\sqrt{(1-z_t)(1+z_t)} - \frac{1}{2Z}\sqrt{(1-z_t)(1+z_t)} \\
&= 0.
\end{aligned}$$

This implies that for the distribution D_{t+1}, the classifier f_t is no better than random guessing.

Extra credit problem 2

In the experiments, we noticed that the training error was identical in rounds 1 and 2, and the same was true for the test error. The reason this happens is because the weight α_1 given to the first classifier f_1 is greater than the weight α_2 given to the second classifier, f_2 (we have $\alpha_1 = 0.6830$, and $\alpha_2 = 0.5796$). Since the final classifier constructed by AdaBoost after the second round outputs a weighted majority of the preditions of f_1 and f_2, the output is always going to be exactly the prediction of f_1 since f_1 has greater weight than f_2. Since f_1 is the final classifier constructed by AdaBoost after the first round, this means that the predictions of the final classifiers after rounds 1 and 2 will match perfectly on any example. Consequently, the error rates will be the same as well.