CSE446 Machine Learning, Spring 2013: Homework 2

Due: Friday, May 3^{rd}, beginning of class

Instructions There are 3 written questions on this assignment, plus a fifth coding question. Submit both your written answers (as a .txt or a .pdf) and your implementation to the Dropbox at `https://catalyst.uw.edu/collectit/assignment/darylh/26829/108641`

1 Boosting [30 Points]

We learned about boosting in lecture and the topic is covered in Murphy 16.4. On page 555 Murphy claims that "it was proved that one could boost the performance (on the training set) of any weak learner arbitrarily high, provided the weak learned could always perform slightly better than chance." We will now verify this in the AdaBoost framework.

1. (*10 points*) Given a set of N observations (x^j, y^j) where y^j is the label $y^j \in \{-1, 1\}$, let $h_t(x)$ be the weak classifier at step t and let α_t be its weight. First we note that the final classifier after T steps is defined as:

$$H(x) = sgn\left\{\sum_{t=1}^{T} \alpha_t h_t(x)\right\} = sgn\{f(x)\}$$

Where

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

Show that:

$$\epsilon_{\text{Training}} = \frac{1}{N}\sum_{j=1}^{N} 1_{\{H(x^j)\neq y^j\}} \leq \frac{1}{N}\sum_{j=1}^{N} \exp(-f(x^j)y^j)$$

Where $1_{\{H(x^j)\neq y^j\}}$ is 1 if $H(x^j) \neq y^j$ and 0 otherwise.

hint: Consider an arbitrary y^j and x^j and given that $f(x^j) < 0$ or $f(x^j) > 0$ and $y^j = 1$ or $y^j = -1$ consider all possible pairs $(1_{\{H(x^j)\neq y^j\}}, \exp(-f(x^j)y^j))$. You may also want to recall that $e^{-x} \geq 1 \Leftrightarrow x \leq 0$.

2. (*10 points*) The weight for each data point j at step $t+1$ can be defined recursively by:

$$w_j^{(t+1)} = \frac{w_j^{(t)} \exp(-\alpha_t y^j h_t(x^j))}{Z_t}$$

Where Z_t is a normalizing constant ensuring the weights sum to 1:

$$Z_t = \sum_{j=1}^{N} w_j^t \exp(-\alpha_t y^j h_t(x^j))$$

Show that:

$$\frac{1}{N}\sum_{j=1}^{N} \exp(-f(x^j)y^j) = \prod_{t=1}^{T} Z_t$$

hint: Consider $w_j^{(T+1)}$ and continue the recursion (i.e. plug in the recursive formula for $w_j^{(T)}$ in to that of $w_j^{(T+1)}$ until using that $w_j^1 = \frac{1}{N}$) and that $e^{\sum_t a_t} = \prod_t e^{a_t}$. Then use that $\sum_{j=1}^N w_j^{(T+1)} = 1$

3. (*10 points*) We showed above that training error is bounded above by $\prod_{t=1}^T Z_t$. At step t the values $Z_1, Z_2, \ldots, Z_{t-1}$ are already fixed therefore at step t we can choose α_t to minimize Z_t. Let

$$\epsilon_t = \sum_{i=1}^m w_j^t 1_{\{h_t(x^j) \neq y^j\}}$$

be the weighted training error for weak classifier $h_t(x)$ then we can re-write the formula for Z_t as:

$$Z_t = (1 - \epsilon_t)\exp(-\alpha_t) + \epsilon_t \exp(\alpha_t)$$

(a) First find the value of α_t that minimizes Z_t then show that

$$Z_t^{opt} = 2\sqrt{\epsilon_t(1 - \epsilon_t)}$$

(b) Assume we choose Z_t this way. Then re-write $\epsilon_t = \frac{1}{2} - \gamma_t$ where $\gamma_t > 0$ implies better than random and $\gamma_t < 0$ implies worse than random. Then show that:

$$Z_t \leq \exp(-2\gamma_t^2)$$

You may want to use the fact that $\log(1 - x) \leq -x$ for $0 < x \leq 1$

Thus we have:

$$\epsilon_{\text{training}} \leq \prod_{t=1}^T Z_t \leq \exp(-2\sum_{t=1}^T \gamma_t^2)$$

(c) Finally, show that if each classifier is better than random (e.g. $\gamma_t \geq \gamma$ for all t and $\gamma > 0$) that:

$$\epsilon_{\text{training}} \leq \exp(-2T\gamma^2)$$

Which shows that the training error can be made arbitrarily small with enough steps.

2 KNN [5 Points]

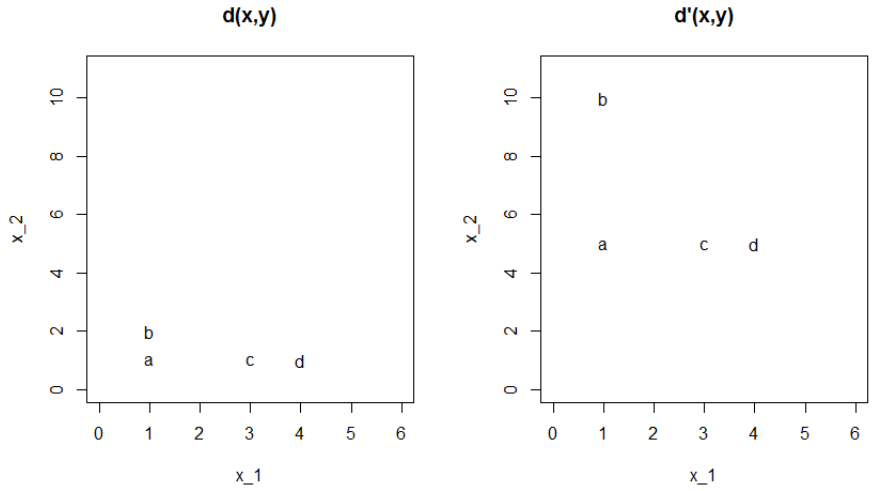
The KNN algorithm is highly sensitive to different distance metrics. Consider fitting a KNN algorithm to four data points labeled a, b, c, d using the standard euclidean distance in $\mathbb{R}^2$. Let $x = (x_1, x_2)$ and $y = (y_1, y_2)$ be points in $\mathbb{R}^2$ then the Euclidean distance is defined as:

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

Now consider fitting those same data points but using the following distance metric:

$$d'(x, y) = \sqrt{(x_1 - y_1)^2 + (5x_2 - 5y_2)^2}$$

We could, instead of changing the distance metric, change the point locations to reflect the change in metric. To do this we would multiply the second coordinate of all the points by 5. Consider the following plots.



The plot on the left should be read in the usual sense and represents the points under the first distance metric. The effect of changing our distance metric to $d'(x, y)$ is equivalent to taking all points (x_1, x_2) and plotting $(x_1, 5x_2)$ and calculating distance using the Euclidean metric (so we can "see" how far they are apart). The plot on the left is of the same four points, however, the second coordinate for all points was multiplied by 5 . Under the standard metric (the left hand plot) the $1NN$ of a is b (it's the closest point). Under the new metric, however, the $1NN$ of a is c (it's the closest point).

1. (*1 pt*) Can you define points a, b, c, and d such that exactly **zero** points will change $1NN$? If yes, provide the coordinates of each point and a brief explanation (you may include a diagram).

2. (*1 pt*) Can you define points a, b, c, and d such that exactly **one** point will change $1NN$? If yes, provide the coordinates of each point and a brief explanation (you may include a diagram).

3. (*1 pt*) Can you define points a, b, c, and d such that exactly **two** points will change $1NN$? If yes, provide the coordinates of each point and a brief explanation (you may include a diagram).

4. (*1 pt*) Can you define points a, b, c, and d such that exactly **three** points will change $1NN$? If yes, provide the coordinates of each point and a brief explanation (you may include a diagram).

5. (*1 pt*) Can you define points a, b, c, and d such that all **four** points will change $1NN$? If yes, provide the coordinates of each point and a brief explanation (you may include a diagram).

3 Decision Trees [30 Points]

3.1 Two Dimensions

The use of decision trees for classification is discussed in section 16.2 of the Kevin Murphy textbook. Suppose we are trying to use decision trees to learn the following two-class function:

X_1	X_2	Y
0	0	-
0	1	+
1	0	+
1	1	-

Table 1: $X_1 \oplus X_2$

1. *(3 points)* What is the classification error of a decision tree of depth 0? *(That is, a single leaf that predicts the majority class of the whole dataset)*

2. *(3 points)* What is the best possible classification error of a decision tree of depth 1? Draw such a tree.

3. *(3 points)* What is the best possible classification error of a decision tree of depth 2? Draw such a tree.

4. *(3 points)* Suppose our algorithm for building a decision tree followed KM's Algorithm 16.1, where we decide a node is not worth splitting if its classification error is not δ better than its parent, for some $\delta > 0$.[1] What kind of tree will this algorithm produce, and what will its classification error be?

5. Suppose instead our criterion was that we would stop if none of our splits resulted in any information gained. Define the entropy for the two-class case as follows:

   If there are P positive examples and N negative examples in a branch, let $\pi_P = \frac{P}{P+N}, \pi_N = \frac{N}{P+N}$.
   Then the entropy is defined as $H[P,N] = \pi_P \log_2(\frac{1}{\pi_P}) + \pi_N \log_2(\frac{1}{\pi_N})$.
   Suppose a parent node has training data D, which a split partitions into D_L and D_R. Then the information gain is the difference between the entropy of the parent and the entropy of the children. That is, $IG = H[P,N] - (\frac{|D_L|}{|D|}H[P_L,N_L] + \frac{|D_R|}{|D|}H[P_R,N_R])$.

   (a) *(3 points)* For the tree you drew of depth 2, calculate the entropy at each split point (as well as the root).

   (b) *(3 points)* Calculate the information gain at each split point of this tree.

   (c) *(3 points)* If we use this as a criterion for termination, what will our tree look like? What will be its classification error?

3.2 Multiple Dimensions

Having learned from our mistake in the previous problem, we now decide to change our stopping criterion to be when the depth of the tree has reached a certain value k. We set $k = 100$ for now, since that seems like more than enough. Feeling confident, we've decided to attempt to learn a more complicated function, which turns out to be $F = X_1 \oplus X_2 \oplus \ldots \oplus X_{100} \oplus X_{101}$. Assume that we have an equal number of training examples for each possible value of $X \in \{0,1\}^{101}$.

1. *(3 points)* What is the classification error of a decision tree of depth 0?

2. *(3 points)* What is the best possible classification error of a decision tree of depth 100? *(This is the best that our algorithm could hope to achieve)*

3. *(3 points)* What is the best possible classification error of a decision tree of depth 101?

4 Programming Question [35 Points]

In this problem, you will train a logistic regression model to predict the Click Through Rate (CTR) on a dataset with about 10,000 examples. The CTR provides a measure of the popularity of an advertisement, and the features we will use for prediction include attributes of the ad and the user.

4.1 Unprocessed Dataset

The dataset we will consider comes from the 2012 KDD Cup Track 2. Here, a user types a query and a set of ads are displayed and we observe which ad was clicked.

For example:

[1] See Equation 16.9 on page 547

1. Alice went to the famous search engine Elgoog, and typed the query "big data".
2. Besides the search result, Elgoog displayed 3 ads each with some short text including its title, description, etc.
3. Alice then clicked on the first advertisement

This completes a SESSION. At the end of this session Elgoog logged 3 records:

Clicked = 1 | Depth = 3 | Position = 1 | Alice | Text of Ad1 |
Clicked = 0 | Depth = 3 | Position = 2 | Alice | Text of Ad2 |
Clicked = 0 | Depth = 3 | Position = 3 | Alice | Text of Ad3 |

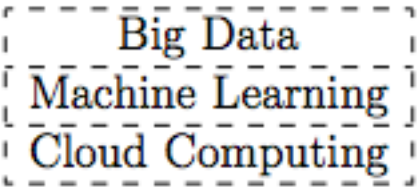
In addition, the log contains information about Alice's age and gender. Here is the format of a complete row of our training data:

Clicked | Depth | Position | Userid | Gender | Age | Text Tokens of Ad

Let's go through each field in detail:

- "Clicked" is either 0 or 1, indicating whether the ad is clicked.
- "Depth" takes a value in $\{1, 2, ..., \}$ specifying the number of ads displayed in the session.
- "Position" takes a value in $\{1, 2, ..., Depth\}$ specifying the rank of the ad among all the ads displayed in the session.
- "Userid" is an integer id of the user.
- "Age" takes a value in $\{1, 2, 3, 4, 5, 6\}$, indicating different ranges of a user's age: ''1' for (0, 12], '2' for (12, 18], '3' for (18, 24], '4' for (24, 30], '5' for (30, 40], and '6' for greater than 40.
- "Gender" takes a value in $\{-1, 1\}$, where -1 stands for male and 1 stands for female.
- "Text Tokens" is a comma separated list of token ids. For example: "15,251,599" means "token_15", "token_251", and "token_599". (Note that due to privacy issues, the mapping from token ids to words is not revealed to us in this dataset, e.g., "token_32" to "big".)

Here is an example that illustrates the concept of features "Depth" and "Position". Suppose the list below was returned by Elgoog as a response to Alice's query. The list has $depth = 3$. "Big Data" has $position = 1$, "Machine Learning" has $position = 2$ and so forth.



Here is a sample from the training data:
$0\,|\,2\,|\,2\,|\,280151\,|\,1\,|\,2\,|\,0, 1, 154, 173, 183, 188, 214, 234, 26, 3, 32, 36, 37, 4503, 51, 679, 7740, 8, 94$
The test data are in the same format except that they do not have the first label field, which is stored in a separate file named "test_label.txt". Some data points do not have user information. In these cases, the userid, age, and gender are set to zero.

4.2 Processed Dataset (that you will use...)

In class, we simply denote

$$x^t = [x_1^t, ..., x_d^t] \tag{1}$$

as an abstract feature vector. In the real world, however, constructing the feature vector requires some thought. We have preprocessed the dataset for you, making the following modifications.

- First of all, not everything in the data should be treated as a feature. In this dataset, "Userid" should not be treated as feature. We have removed this column from the dataset for you.
- Similarly, we cannot directly use the list of token ids as features in Equation 1 since the numbers are arbitrarily assigned and thus meaningless for the purposes of regression. Instead, we should think of the list of token ids $L = [l_1, l_2, l_3, ...]$ as a compact representation of a sparse binary vector b where $b[i] = 1 \forall i \in L$. We have replaced the list of words with a list of binary variables indicating token presence.
- As for the rest of the features: "Depth", "Position", "Age", and 'Gender", they are scalar variables, so we maintain their original values in the dataset.

The dataset that we are giving you has the following form:
Clicked, Depth, Position, Gender, Age, Word1, Word2, Word3, Word4, ..., Word50

So a sample datapoint would be:
0, 1, 2, 1, 12, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0

4.3 Accessing and Processing the Data

1. Download the processed version of the dataset "clickprediction_data.zip" from the course website.
2. After unzipping the folder, there should be three files: train.txt, test.txt and test label.txt.
3. The following set of commands should load the training data to get you started:

```
trainingMatrix <- read.csv("train.txt", header=FALSE)

y <- as.matrix(trainingMatrix[,1])

trainingX <- trainingMatrix[,-1]

x <- as.matrix(cbind(rep(1, dim(trainingX)[1]),trainingX))
```

4.4 Stochastic Gradient Descent

Recall that stochastic gradient descent (SGD) performs a gradient descent using a noisy estimate of the full gradient based on just the current example.

1. Write down the equation for the weight update step. That is, how to update weights w^t using the data point (x^t, y^t), where $x^t = [x_1^t, x_2^t, ..., x_d^t]$ is the feature vector for example t, and $y^t \in \{0, 1\}$ is the label.
2. For stepsizes $\eta = \{0.5, 0.25, 0.125\}$ and without regularization, implement SGD and train the weights by making one pass over the dataset. You may assume that the data is randomly ordered. Use only one pass over the data on all subsequent questions as well. For each step size:

(a) Plot the average loss $\bar{L}$ as a function of the number of steps T, where

$$\bar{L}(T) = \frac{1}{T}\Sigma_{t=1}^{T}(\hat{y}^t - y^t)^2 \tag{2}$$

where $\hat{y}^t$ (either 0 or 1) is the predicted label of example x^t using the weights w^{t-1}. Record the average loss every 100 steps, e.g. [100, 200, 300, ...].

(b) Report the l_2 norm of the weights at the end of the pass. Note that if your l_2 norm is not small and keeps growing as you get more and more data, adding l_2 regularization to each update state becomes necessary. However, implementing the addition of l_2 regularization is beyond the scope of this problem set, so you are not expected to do so.

(c) Use the weights to predict the CTRs for the test data. Recall that "test_label.txt" contains the labels for the test data. Report the SSE (sum of squared erros) of your predicted CTR. Make sure to use the SSE for the 0/1 prediction. (Do not expect a huge improvement since the label distribution is biased. Elgoog still makes a huge profit even with a 0.1% improvement in accuracy.)

3. For $\eta = 0.125$, report the weights for the following features: intercept, "Position", "Depth", "Gender", and "Age". Provide an interpretation of the effect of each feature on the probability of a click based on these inferred weights.