CS 475 Machine Learning: Homework 3
Non-linear Methods
Due: Wednesday October 12, 2016, 11:59pm
100 Points Total Version 1.0

Jonathan Liu
jliu118

1 Analytical (50 points)

1) Decision Trees (12 points) Consider a binary classification task with the following set of training examples:

x_1	x_2	x_3	x_4	x_5	classification
0	1	1	-1	-1	+
0	1	1	-1	-1	+
0	1	1	1	1	-
0	-1	1	1	1	+
0	-1	1	-1	-1	-
0	-1	1	-1	-1	-

(a) Can this function be learned using a decision tree? If so, provide such a tree (describe each node in the tree). If not, prove it.

(b) Can this function be learned using a logistic regression classifier? If yes, give some example parameter weights. If not, why not.

(c) What is the entropy of the labels in the training data?

(d) What is the information gain of x_2 relative to labels?

Answer:

(a) See attached page for tree.

(b) see attached page for graph. Looking at the graph, you cannot create a hyperplane decision rule that can separate the data.

(c) $H(Y) = -P(+)\log_2 P(+) - P(-)\log_2 P(-) = -\frac{1}{2}(-1) - \frac{1}{2}(-1) = 1$

(d) $-\frac{2}{6}\log\frac{2/6}{1/2} - \frac{1}{6}\log\frac{1/6}{1/2} - \frac{1}{6}\log\frac{1/6}{1/2} - \frac{2}{6}\log\frac{2/6}{1/2}$
$= 0.918$
$IG(Y|X) = H(Y) - H(Y|X) = 1 - 0.918 = 0.0817$

2) Decision Tree (12 points) Let's investigate the accuracy of decision tree learning. We start by constructing a unit square ($[0;1] \times [0;1] \times [0;1]$). We select n samples from the cube, each with a binary label ($+1$ or -1), such that no two samples share either x, y, or z coordinates. Each feature can be used multiple times in a decision tree. At each node we can only conduct a binary threshold split using one single feature.

(a) Prove that we can find a decision tree of depth at most $\log_2 n$, which perfectly labels all n samples.

(b) If the samples can share either two coordinates but not all three, can we still learn a decision tree which perfectly labels all n samples with the same depth as (a)? Why or why not?

Answer:

(a) Since no two samples share either x, y, or z coordinates, there must be at most n leaves in the decision tree. And since the labels have only two options, there are two branches per node, so the depth is $\log_b n$ where b = 2 so it becomes $\log_2 n$.

(b) Since now you can share coordinates, there will be more options for paths to leaves so you'll end up having more leaves for the same number of samples n, so naturally depth will be increased.

3) Adaboost (14 points) There is one good example at $x = 0$ and two negative examples at $x = \pm 1$. There are three weak classifiers are

$$h_1(x) = 1 \cdot \mathbf{1}(x > 1/2) - 1 \cdot \mathbf{1}(x \leq 1/2),$$
$$h_2(x) = 1 \cdot \mathbf{1}(x > -1/2) - 1 \cdot \mathbf{1}(x \leq -1/2)$$
$$h_3(x) = 1.$$

Show that this data can be classified correctly by a strong classifier which uses only three weak classifiers. Calculate the first two iterations of AdaBoost for this problem. Are they sufficient to classify the data correctly?

Answer:

$$h_1(0) = -1$$
$$h_1(-1) = -1$$
$$h_1(1) = 1$$
$$Accuracy : 1/3$$

$$h_2(0) = 1$$
$$h_2(-1) = -1$$
$$h_2(1) = 1$$
$$Accuracy : 2/3$$

$$h_3(0) = 1$$
$$h_3(-1) = 1$$
$$h_3(1) = 1$$
$$Accuracy : 1/3$$

Since for h_1 and h_3 the accuracy was less than 50%, we can just flip the inequalities for those two classifiers and we can classify the data correctly.

$D_1(i) = 1/n = 1/3$
$\epsilon_1(h_1) = 2/3$
$\alpha = -0.347$
$Z = 0.943$
$D_2(0) = 0.25$
$D_2(-1) = 0.5$
$D_2(1) = 0.25$
$\epsilon_2(h_2) = 0.25$
$\alpha_2 = 0.5 \log 3 = 0.549$
Predicting:
$f(0) = 1$ because if it were 1, then the alpha value would be -0.347 which is less than 0.25
$f(-1) = -1$ because from h1 and h2 that was the only label
$f(1) = 1$ because from h1 and h2 that was the only label

4) Ensemble Methods (12 points) Consider the following binary classification Boosting algorithm.

1. Given $\{\mathbf{x}_i, y_i\}_{i=1}^N$, number of iterations T, weak learner f.
2. Initialize $\mathcal{D}_0$ to be a uniform distribution over examples.
3. For each iteration $t = 1 \dots T$:
   (a) Train a weak learner f on the data given $\mathcal{D}_t$ to produce hypothesis h_t.
   (b) Compute the error of h_t as $\epsilon_t = P_{\mathcal{D}_t}[h_t(\mathbf{x}_i) \neq y_i]$
   (c) Compute $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$

(d) Update $\mathcal{D}$ as:

$$\mathcal{D}_{t+1}(i) = \frac{\mathcal{D}_t(i)}{Z_t} \times \begin{cases} \exp(-\alpha_t + (T-t)/T) & \text{if } h_t(\mathbf{x}_i) = y_i \\ \exp(\alpha_t + (T-t)/T) & \text{otherwise} \end{cases}$$

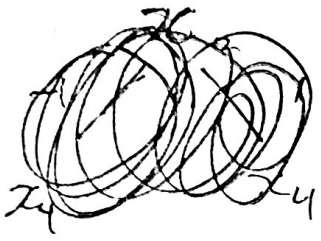
4. Output final hypothesis $H(\mathbf{x}) = \text{sign}\left\{\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right\}$

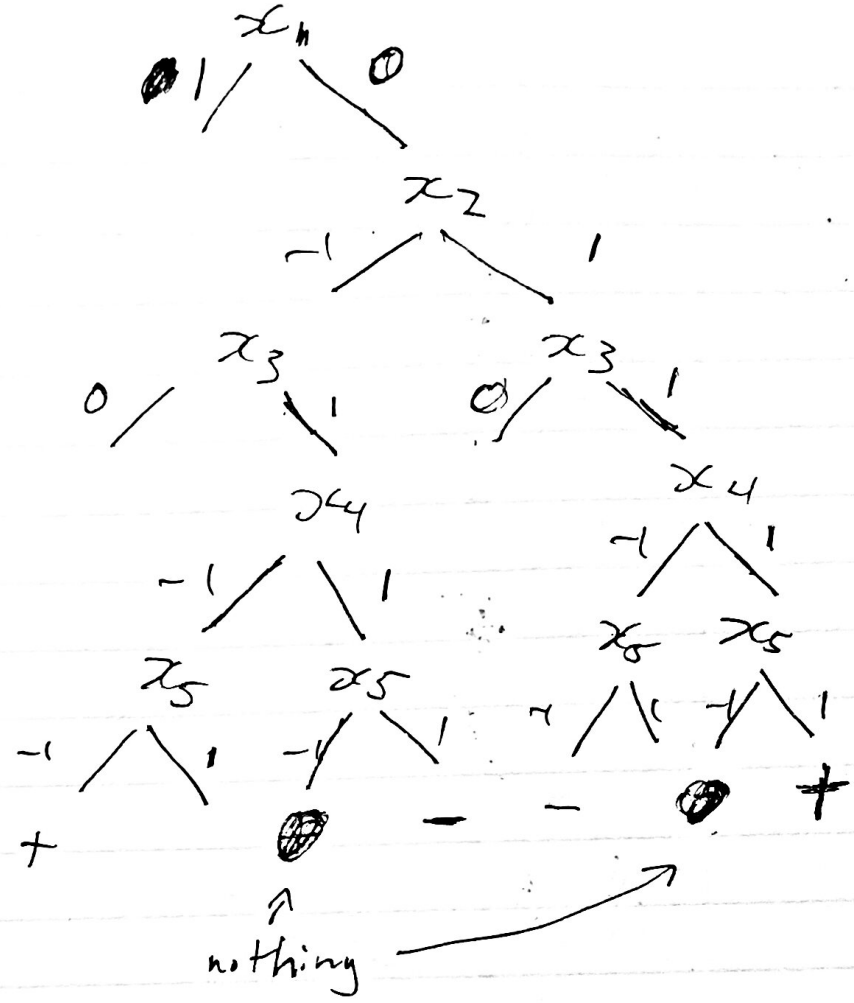
Z_t is a normalization constant so that $\mathcal{D}$ is a valid probability distribution.

Describe the difference between this algorithm and the AdaBoost algorithm we learned about in class. What problem of AdaBoost is this change designed to fix? How does changing the algorithm's user provided parameter affect this behavior?

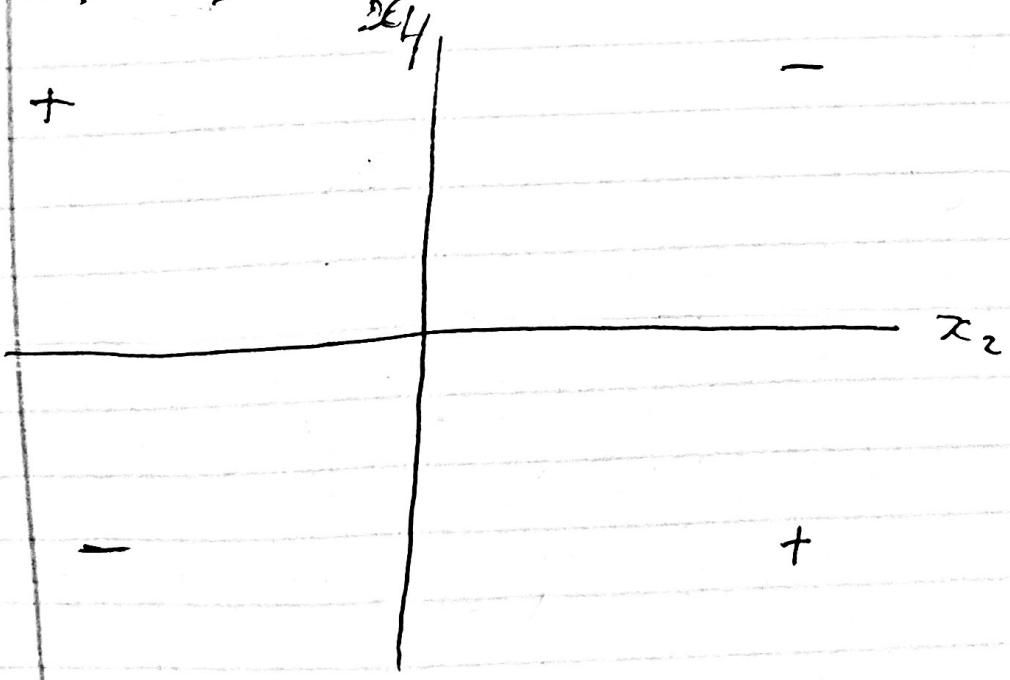
Answer: The only difference between the new algorithm and AdaBoost is how the distribution is updated. Looking at the new algorithm, as the number of iterations increases, the smaller the impact of the alpha values for the later iterations. So this would have the effect of reducing the possibility of overfitting.

(1 a)




x_1
1
0
x_2
-1
1
x_3
x_3
0
1
0
1
x_4
x_4
x_4
-1
1
-1
1
-1
1
x_5
x_5
x_6
x_5
-1
1
-1
1
-1
1
+
-
-
+
nothing

b) x_2 and x_4


x_4
+
-
x_2
-
+