PS3: Decision Trees, Ensemble Methods

DS 5220: Supervised Machine Learning and Learning Theory
Omer Seyfeddin Koc

April 3, 2023

1 Decision Trees on a Toy Dataset (20%)

Master Yoda is concerned about the number of Jedi apprentices that have turned to the Dark Side, so he's decided to train a decision tree on some historical data to help identify problem cases in the future. The following table summarizes whether or not each of the 12 initiates turned to the Dark Side based on their age when their Jedi training began, whether or not they completed their training, their general disposition, and their species.

Dark Side	Age Started Training	Completed Training	Disposition	Species
0	5	1	Happy	Human
0	9	1	Happy	Gungan
0	6	0	Happy	Wookiee
0	6	1	Sad	Mon Calamari
0	7	0	Sad	Human
0	8	1	Angry	Human
0	5	1	Angry	Ewok
1	9	0	Happy	Ewok
1	8	0	Sad	Human
1	8	0	Sed	Human
1	6	0	Angry	Wookiee
1	7	0	Angry	Mon Calamari

1. What is the initial entropy of Dark Side?

   Solution : To calculate the initial entropy of a dataset, we first calculate the proportion of instances in each class. We can then use these proportions to calculate the entropy using the following formula:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_1)$$

   where $H(X)$ is the entropy of the variable X, $P(x_i)$ is the probability of each possible value x_i of X.

$$p(Dark\ Side = 0) = \frac{7}{12}$$
$$p(Dark\ Side = 1) = \frac{5}{12}$$

$$\begin{aligned} H(X) &= -\frac{5}{12}\log_2(\frac{5}{12}) - \frac{7}{12}\log_2(\frac{7}{12}) \\ &= -[0.4167 * (-1.2630)] - [0.5833 * (-0.7776)] \\ &= -[-0.5264] - [-0.4536] = 0.5264 + 0.4536 \\ &= \mathbf{0.9799} \end{aligned}$$

2. Which attribute would the decision tree building algorithm choose to use for the root of the tree?

   Solution : Idea is a good attribute splits the examples into purer subsets. Considering all the attribute, the most striking attribute is the **Completed Training**. Dark side=1 in 5 of 7 cases where completed training was 0, while dark side was always 0 when completed training was 1. That's why root of the tree should be Completed Training. Also, Information Gain is calculated for each feature below, and the highest IG value in these calculations is found in the 'Completed Training' feature (this prove our approach).

   As a mathematical approach, to determine which attribute to use as the root of the decision tree, we need to calculate information gain for each feature. The attribute with the highest information gain is chosen as the root of the tree. The formula for information gain is:

$$\Delta_{info} = H(p) - \sum_{i=1}^{k} \frac{n_i}{n} H(v_i)$$

   In the given dataset, the target variable is Dark Side attribute. We calculate the information gain of each attribute as follows ($p \log_2 p$ is considered to be 0 whenever $p = 0$):

$$\begin{aligned}
IG(Age\ Started\ Training) &= 0.9799 - [\frac{2}{12}(\frac{-2}{2}\log_2\frac{2}{2} - \frac{0}{2}\log_2\frac{0}{2})] - [\frac{3}{12}(\frac{-2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3})] \\
&\quad - [\frac{2}{12}(\frac{-1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2})] - [\frac{3}{12}(\frac{-1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3})] \\
&\quad - [\frac{2}{12}(\frac{-1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2})] \\
&= 0.9799 - 0.7925 \\
&= \mathbf{0.1874} \\
IG(Completed\ Training) &= 0.9799 - [\frac{5}{12}(-\frac{0}{5}\log_2(\frac{0}{5}) - \frac{5}{5}\log_2(\frac{5}{5})) + (\frac{7}{12}(-\frac{5}{7}\log_2(\frac{5}{7}) - \frac{2}{7}\log_2(\frac{2}{7}))] \\
&= 0.9799 - [0 + \frac{7}{12}(0.8631)] \\
&= 0.9799 - 0.5035 \\
&= \mathbf{0.4764*} \\
IG(Disposition) &= 0.9799 - [\frac{4}{12}(\frac{-3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4})] - [\frac{4}{12}(\frac{-2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4})] \\
&\quad - [\frac{4}{12}(\frac{-2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4})] \\
&= 0.9799 - 0.9371 \\
&= \mathbf{0.0428} \\
IG(Species) &= 0.9799 - [\frac{5}{12}(\frac{-3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5})] - [\frac{1}{12}(\frac{-1}{1}\log_2\frac{1}{1} - \frac{0}{1}\log_2\frac{0}{1})] \\
&\quad - [\frac{2}{12}(\frac{-1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2})] - [\frac{2}{12}(\frac{-1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2})] \\
&\quad - [\frac{2}{12}(\frac{-1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}) \\
&= 0.9799 - 0.9046 \\
&= \mathbf{0.0753}
\end{aligned}$$

   Therefore, the **Completed Training** attribute has the highest information gain and would be chosen as the **root of the decision tree.**

3. What is the information gain of the attribute you chose to split on in the previous question?

Solution :

$$IG(Completed\ Training) = 0.9799 - [\frac{5}{12}(-\frac{0}{5}\log_2(\frac{0}{5}) - \frac{5}{5}\log_2(\frac{5}{5})) + (\frac{7}{12}(-\frac{5}{7}\log_2(\frac{5}{7}) - \frac{2}{7}\log_2(\frac{2}{7}))]$$
$$= 0.9799 - [0 + \frac{7}{12}(0.8631)] = 0.9799 - 0.5035$$
$$= \mathbf{0.4764}$$

4. Draw the full decision tree that would be learned for this data (with no pruning).
   Hint: The correct split at each point should be clear from just the groups it splits the data into, without having to actually compute the information gain for each possible split.

   Solution :

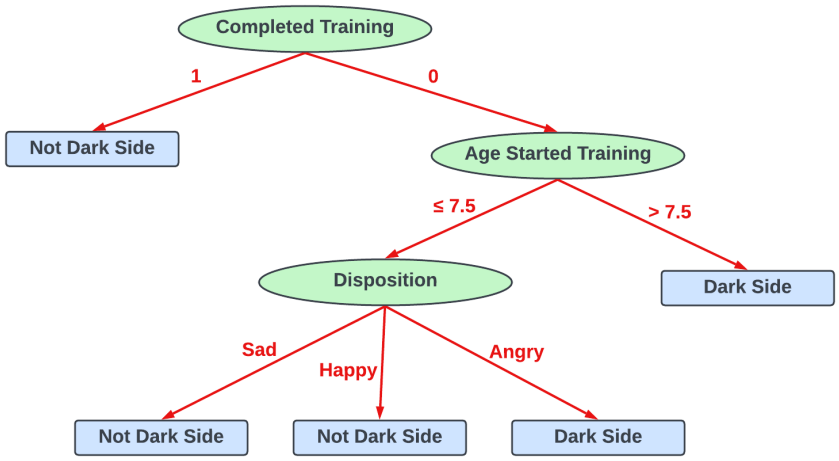

Figure 1: Decision Tree

2 Decision Trees Implementation (30%)

1. Implement the decision tree learning algorithm described in class(without post-pruning). Your program can assume that all features/attributes are real-valued attributes (no categorical attributes), except for the target attribute that is discrete and may take on a finite set of possible values. You may also assume that the data contains no missing attributes. Use information gain as your splitting criterion and allow only binary splits.
   Your algorithm should accept an optional parameter that sets the limit of the depth of the tree that will be generated when training. If the depth limit has not been set, the entire tree should be learned. If the depth limit has been set, only nodes up to that depth limit should be built.

2. Run your algorithm on the Iris flower data set (available at *sklearn.datasets*), and display the resulting decision tree.

3. Run $DecisionTreeClassifier$ from scikit-learn on the same data set, and compare its generated decision tree to the one built by your algorithm.

Solution : The answers to the related question are shown in the Python notebook named Omer_3_2.ipynb.

3 AdaBoost on a Toy Dataset (20%)

Now we will apply AdaBoost to classify a toy data set. Consider the data set shown in Figure 1a. The data set consists of 4 points: $(0, 1, -), (1, 0, +), (1, 0, +)$ and $(0, 1, -)$. For this part, you may find it helpful to use

python as a calculator rather than doing the computations by hand. (You do not need to submit any code though.)

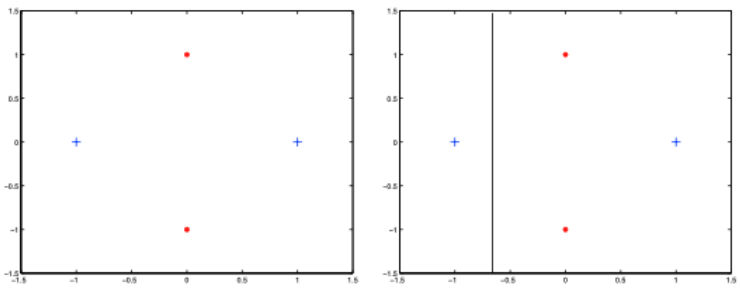

Figure 2: a) Toy data set b)Decision stump h_1.

1. For T = 4, show how AdaBoost works for this data set, using simple decision stumps (depth-1 decision trees that simply split on a single variable once) as weak classifiers. For each time step compute the following:

$$\epsilon_t, \alpha_t, Z_t, D_t(i)\ \forall i$$

Also for each time step draw your weak classifier. For example, h_1 can be as shown on Figure 2b.

Solution : Python is used for calculations. The table for values of the parameters at each timestep is given below:

t	ϵ	α	Z_t	$D_t(1)$	$D_t(2)$	$D_t(3)$	$D_t(4)$	$h_t(1)$, $h_t(2)$, $h_t(3)$, $h_t(4)$
1	0.2500	0.5493	0.8660	0.2500	0.2500	0.2500	0.2500	$-1, -1, 1, -1$
2	0.1667	0.8047	0.7454	0.1667	0.5000	0.1667	0.1667	$1, 1, 1, -1$
3	0.1000	1.0986	0.6000	0.5000	0.3000	0.1000	0.1000	$-1, 1, -1, -1$
4	0.0556	1.4166	0.4581	0.2778	0.1667	0.5000	0.0556	$-1, 1, 1, -1$

The output of the Python code is as follows:

```
---------------
Iteration 1
epsilon_t = 0.25
alpha_t = 0.5493061443340549
Z_t = 0.8660254037844386
D_(t+1)(i) = [0.16666667 0.5        0.16666667 0.16666667]
---------------
Iteration 2
epsilon_t = 0.16666666666666666
alpha_t = 0.8047189562170503
Z_t = 0.74535599249993
D_(t+1)(i) = [0.1 0.3 0.5 0.1]
---------------
Iteration 3
epsilon_t = 0.09999999999999998
alpha_t = 1.0986122886681098
Z_t = 0.5999999999999999
D_(t+1)(i) = [0.05555556 0.16666667 0.27777778 0.5       ]
---------------
Iteration 4
epsilon_t = 0.055555555555555546
alpha_t = 1.416606672028108
Z_t = 0.4581228472908511
D_(t+1)(i) = [0.5        0.08823529 0.14705882 0.26470588]
Predictions: [ 1. -1. -1.  1.]
```

Figure 3: Output of Python's Computation Code

The weak classifier for each time step is as follows:

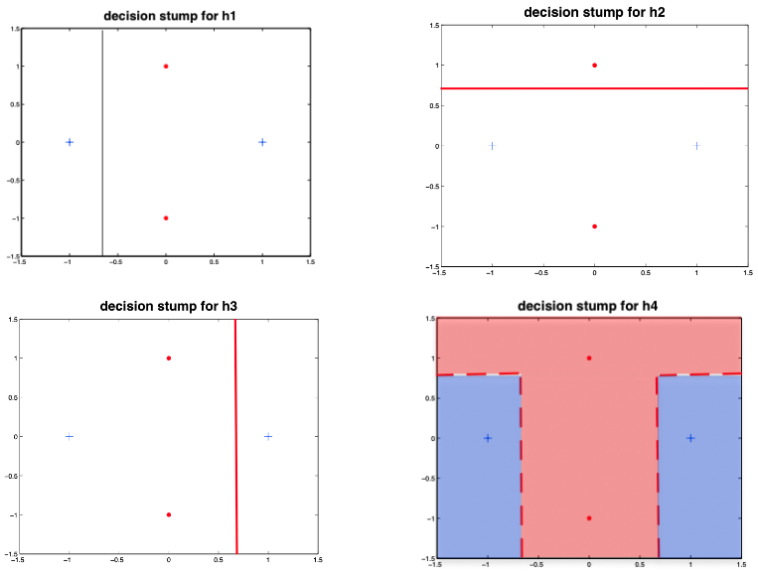

Figure 4: Weak classifier

2. What is the training error of AdaBoost for this toy data set?

   Solution : After $T = 4$ rounds, the Adaboost classifier estimated the classes x_i as -1, +1 , +1 , and -1, respectively. This estimation matches with our data exactly, so we have a perfect fit and our training error value is **0.**

3. Is the above data set linearly separable? Explain why AdaBoost does better than a decision stump on the above data set.

   Solution : The data in this dataset cannot be separated linearly and a simple decision stump is incapable of perfectly classifying it. Nevertheless, Adaboost merges the outputs of multiple weak learners to produce a nonlinear boundary for classification. That's why Adaboost does better than a decision stump.

4 Training Error of Boosting (30%)

Consider the AdaBoost algorithm you saw in class. In this question we will try to analyze the training error of boosting.

1. Given a set of n examples, (x_i, y_i) (y_i is the class label of x_i), $i = 1, \ldots, n$, let $h_t(x)$ be the weak classifier obtained at step t, and let α_t be its weight. Recall that the final classifier is

$$H(x) = \text{sign}(f(x)\}, \text{ where } f(x) = \sum_{i=1}^{T} a_t h_t(x).$$

   Show that the training error of the final classifier can be bounded from above by an exponential loss function.

$$\frac{1}{n}\sum_{i=1}^{n} I\left(H\left(\mathbf{x}_i\right) \neq y_i\right) \leq \frac{1}{n}\sum_{i=1}^{n} \exp\left(-f\left(\mathbf{x}_i\right) y_i\right)$$

where $I(a = b)$ is the indicator function. It is equal to 1 if $a = b$, and 0 otherwise.
Hint: $e^{-x} \geq 1 \Leftrightarrow x \leq 0$.

Solution :

Let's start by examining the e^{-x} function first.

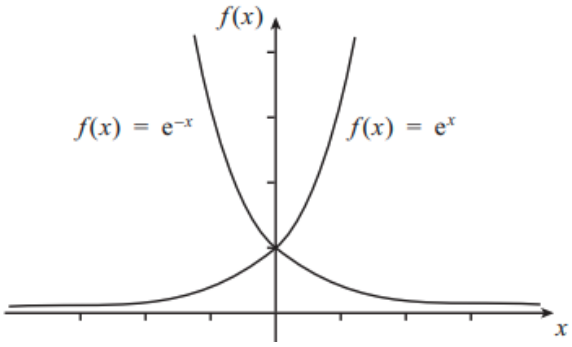

Figure 5: e^{-x} function

As can be seen from the graph, two conditions are valid for e^{-x}:

$$e^{-x} \geq 1 \Leftrightarrow x \leq 0 \; and \; e^{-x} \geq 0 \Leftrightarrow x \geq 0$$

Then create a comparison chart of all these values and functions in the equation.

$f(x_i)$	y_i	$H(x_i)$	$I(H(x_i) \neq y_i)$	$e^{-f(x_i)y_i}$
> 0	1	1	0	≥ 0
> 0	-1	1	1	≥ 1
< 0	1	-1	1	≥ 1
< 0	-1	-1	0	≥ 0

(1)

According to the comparison in the table, $e^{-f(x_i)y_i}$ is always ≥ 0. Also, $e^{-f(x_i)y_i} \geq 1$ when $f(x_i)\, y_i \leq 0$. Regardless of $f(x_i)$ and y_i values, we reach the following equation from the table:

$$I(H(x_i) \neq y_i) \leq e^{-f(x_i)y_i}$$

Summing over all i, we get:

$$\sum_{i=1}^{n} I(H(x_i) \neq y_i) \leq \sum_{i=1}^{n} e^{-f(x_i)y_i}$$

Dividing both sides by n, we get:

$$\frac{1}{n} \sum_{i=1}^{n} I(H(x_i) \neq y_i) \leq \frac{1}{n} \sum_{i=1}^{n} e^{-f(x_i)y_i}$$

Thus, it is proved that the training error of the final classifier can be bounded from above by an exponential loss function.

2. Remember that

$$D_{t+1}(i) = \frac{D_t(i) \exp\left(-\alpha_t y_i h_t\left(\mathbf{x}_i\right)\right)}{Z_t}$$

where Z_t is the normalization factor for distribution D_{t+1} :

$$Z_t = \sum_{i=1}^{n} D_t(i) \exp\left(-\alpha_t y_i h_t\left(\mathbf{x}_i\right)\right)$$

Use this recursive definition to prove the following:

$$\frac{1}{n} \sum_{i=1}^{n} \exp\left(-f\left(\mathbf{x}_i\right) y_i\right) = \prod_{t=1}^{T} Z_t.$$

Hint: Remember that $e^{\sum_i g_i} = \prod_i e^{g_i}$, $D_1(i) = \frac{1}{n}$, and that $\sum_i D_{t+1}(i) = 1$.

Solution : Equation for D_{t+1}:

$$D_{t+1}(i) = \frac{D_t(i) \exp\left(-\alpha_t y_i h_t\left(\mathbf{x}_i\right)\right)}{Z_t}$$

We will update the equation for D_t:

$$D_t(i) = \frac{D_{t-1}(i) \exp\left(-\alpha_{t-1} h_{t-1}\left(\mathbf{x}_i\right) y_i\right)}{Z_{t-1}}$$

Let's substitute the D_t equation in the D_{t+1} equation:

$$D_{t+1}(i) = \frac{D_{t-1}(i) \exp\left(-\alpha_t h_t\left(x_i\right) y_i\right) \exp\left(-\alpha_{t-1} h_{t-1}\left(x_i\right) y_i\right)}{Z_t Z_{t-1}}$$

By doing this transformation up to $D_1(i)$, we get the following equation:

$$\begin{aligned} D_{t+1}(i) &= \frac{1}{Z_1, \cdots Z_t} D_1(i) \exp[-y_i(\alpha_t h_t(x_i) + \cdots + \alpha_1 h_1(x_1))] \\ &= \frac{D_1(i) \exp\left(\sum_{t=1}^{T} -\alpha_t h_t\left(x_i\right)\right)}{\prod_t Z_t} \\ &= \frac{D_1(i) \exp\left(-f\left(x_i\right) y_i\right)}{\prod_t Z_t} \end{aligned}$$

Since, $\mathrm{D}_1(i) = \frac{1}{n}$, and $\sum_i D_{t+1}(i) = 1$, let's put both sides in the sum operation and substitute $D_1(i)$

$$\begin{aligned} \sum_{i=1}^{n} D_{t+1}(i) &= \frac{\sum_{i=1}^{n} \exp\left(-f\left(x_i\right) y_i\right)}{n \prod_t Z_t} \\ 1 &= \frac{\sum_{i=1}^{n} \exp\left(-f\left(x_i\right) y_i\right)}{n \prod_t Z_t} \\ \prod_{t=1}^{T} Z_t &= \frac{\sum_{i=1}^{n} \exp\left(-f\left(x_i\right) y_i\right)}{n} \end{aligned}$$

Finally, we get:

$$\prod_{t=1}^{T} Z_t = \frac{1}{n} \sum_{i=1}^{n} \exp\left(-f\left(\mathbf{x}_i\right) y_i\right)$$

3. Equation (4) suggests that the training error can be reduced rapidly by greedily optimizing Z_t at each step. You have shown that the error is bounded from above:

$$\epsilon_{\text{training}} \leq \prod_{t=1}^{T} Z_t$$

Observe that $Z_1, \ldots, Z_{t-1}$ are determined by the first $(t-1)$ rounds of boosting, and we cannot change them on round t. A greedy step we can take to minimize the training error bound on round t is to minimize Z_t.

In this question, you will prove that for binary weak classifiers, Z_t from Equation (3) is minimized by picking α_t as:

$$\alpha_t^* = \frac{1}{2} \log \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

where ϵ_t is the training error of weak classifier h_t for the weighted data set:

$$\epsilon_t = \sum_{i=1}^{n} D_t(i) I(h_t(\mathbf{x}_i) \neq y_i)$$

For this proof, only consider the simplest case of binary classifiers, i.e. the output of $h_t(x)$ is binary, $\{-1, +1\}$.

For this special class of classifiers, first show that the normalizer Z_t can be written as:

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

Hint: Consider the sums over correctly and incorrectly classified examples separately.

Solution :

$$\begin{aligned} Z_t &= \sum_{i=1}^{n} D_t(i) \exp(-\alpha_t h_t(x_i) y_i) \\ &= \sum_{i=1}^{n} D_t(i) \exp(-\alpha_t) I(h_t(x_i) = y_i) + \sum_{i=1}^{n} D_i(i) \exp(\alpha_i) I(h_t(x_i) \neq y_i) \end{aligned}$$

Then, $I(h_t(x_i) = y_i) = 1 - I(h_t(x_i) \neq y_i)$

$$\begin{aligned} Z_t &= \sum_{i=1}^{n} D_t(i) \exp(-\alpha_t) (1 - I(h_t(x_i) \neq y_i)) + \left(\sum_{i=1}^{n} D_t(i) I(h_t(x_i) \neq y_i) \right) \exp(\alpha_i) \\ &= \left(\sum_{i=1}^{n} D_t(i) - \sum_{i=1}^{n} D_t(i) I(h_t(x_i) \neq y_i) \right) \exp(-\alpha_t) + \left(\sum_{i=1}^{n} D_t(i) I(h_t(x_i) \neq y_i) \right) \exp(\alpha_t) \end{aligned}$$

Let's call ϵ_t and change it with summing operation:

$$\epsilon_t = \sum_{i=1}^{n} D_t(i) I(h_t(\mathbf{x}_i) \neq y_i)$$

Finally, we get:

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

4. Now, prove that the value of α_t that minimizes this definition of Z_t is given by Equation (3).

Solution : To minimize Z_t, set the derivative of Z_t to 0:

$$\frac{\partial Z_t}{\partial \alpha_t} = 0$$

$$\frac{\partial(\exp(-\alpha_t))}{\partial\alpha_t}(1-\epsilon_t)+\frac{\partial(\exp(\alpha_t))}{\partial\alpha_t}(\epsilon_t)=0$$
$$-\exp(-\alpha_t)(1-\epsilon_t)+\exp(\alpha_t)\epsilon_t=0$$
$$\exp(\alpha_t)\epsilon_t=\exp(-\alpha_t)(1-\epsilon_t)$$
$$\exp(\alpha_t)\epsilon_t=\frac{1-\epsilon_t}{\exp(\alpha_t)}$$
$$(\exp\alpha_t)^2=\frac{1-\epsilon_t}{\epsilon_t}$$
$$(\exp\alpha_t)=[\frac{1-\epsilon_t}{\epsilon_t}]^{1/2}$$
$$a_t^*=\frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$$

5. Prove that for the above value of α_t we have $Z_t=2\sqrt{\epsilon_t(1-\epsilon_t)}$.

   Solution : Call α_t:

$$a_t^*=\frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$$

   Substituting this into the expression for Z_t derived in the previous question, we get:

$$\begin{aligned}
Z_t &= (1-\epsilon_t)\exp(-\alpha_t)+\epsilon_t\exp(\alpha_t)\\
&= (1-\epsilon_t)\exp(-\frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t})+\epsilon_t\exp(\frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t})\\
&= (1-\epsilon_t)[\frac{\epsilon_t}{(1-\epsilon_t)}]^{1/2}+\epsilon_t[\frac{1-\epsilon_t}{(\epsilon_t)}]^{1/2}\\
&= \sqrt{1-\epsilon_t}\sqrt{\epsilon_t}+\sqrt{\epsilon_t}\sqrt{1-\epsilon_t}\\
&= 2\sqrt{\epsilon_t}\sqrt{1-\epsilon_t}\\
&= 2\sqrt{\epsilon_t(1-\epsilon_t)}
\end{aligned}$$

6. Furthermore, let $\epsilon_t=\frac{1}{2}-\gamma_t$ and prove that $Z_t\le\exp\left(-2\gamma_t^2\right)$. Therefore, we will know

$$\epsilon_{\text{training}}\le\prod_t Z_t\le\exp\left(-2\sum_t\gamma_t^2\right).$$

   Hint: $\log(1-x)\le-x$ for $0<x\le1$.

   Solution : Call ϵ_t and Z_t

$$\epsilon_t=\frac{1}{2}-\gamma_t \text{ and } Z_t=2\sqrt{\epsilon_t(1-\epsilon_t)}$$

   Then,

$$\begin{aligned}
Z_t &= 2\sqrt{(\frac{1}{2}-\gamma_t)(1-(\frac{1}{2}-\gamma_t))}\\
&= 2\sqrt{(\frac{1-2\gamma_t}{2})(\frac{1+2\gamma_t}{2})}\\
&= \sqrt{1-4\gamma_t^2}
\end{aligned}$$

   Take the log for both sides of the equation to be proven

$$Z_t\le\exp\left(-2\gamma_t^2\right)$$
$$\log(Z_t)\le\log(\exp\left(-2\gamma_t^2\right))$$

We get,

$$\frac{1}{2}\log(1-4\gamma_t^2) \leq -2\gamma_t^2$$

Therefore, we will know

$$Z_t \leq \exp\left(-2\gamma_t^2\right) \text{ and } \epsilon_{\text{training}} \leq \prod_t Z_t \leq \exp\left(-2\sum_t \gamma_t^2\right)$$

So, $Z_t \leq \exp\left(-2\gamma_t^2\right)$ is proved by the above operations.

7. If each weak classifier is slightly better than random, so that $\gamma_t \geq \gamma$, for some $\gamma > 0$, then the training error drops exponentially fast in T, i.e.

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

Argue that in each round of boosting, there always exists a weak classifier h_t such that its training error on the weighted data set $\epsilon_t \leq 0.5$.

Solution : Suppose a weak classifier h_t with error ≥ 0.5 is defined. However, we can define a new h_t' classifier that will be the exact opposite of the h_t classifier. In this way, the error value for h_t' will be < 0.5. Thus, we choose the weak classifier that has the minimum error at iteration t. So, $\epsilon_t \leq 0.5$.

8. Show that for $\epsilon_t = 0.5$ the training error can get "stuck" above zero. Hint: $D_t(i)$s may get stuck.

Solution : Call α_t:

$$a_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$$

Put $\epsilon_t = 0.5$,

$$\begin{aligned} a_t &= \frac{1}{2}\log(\frac{1-0.5}{0.5}) \\ &= \frac{1}{2}\log(\frac{0.5}{0.5}) \\ &= \frac{1}{2}\log(1) \\ &= 0 \end{aligned}$$

Then we get a_t is zero. Also, remember that

$$D_{t+1}(i) = \frac{D_t(i)\exp\left(-\alpha_t y_i h_t\left(\mathbf{x}_i\right)\right)}{Z_t}$$

where Z_t is the normalization factor for distribution D_{t+1} :

$$Z_t = \sum_{i=1}^{n} D_t(i)\exp\left(-\alpha_t y_i h_t\left(\mathbf{x}_i\right)\right)$$

In all cases where a_t is 0, Z_t and $D_{t+1}(i)$ are:

$$D_{t+1}(i) = D_t(i) \text{ and } Z_t = 1$$

The parameters and predictions from the classifier will not change in subsequent iterations. Consequently, the training error can be stuck above 0.