B555 - Machine Learning - Homework 4

Enrique Areyan
April 28, 2015

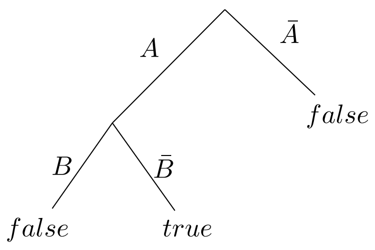
Problem 1: Give decision trees to represent the following Boolean functions

a) $A \wedge \bar{B}$
b) $A \vee [B \wedge \bar{C}]$
c) $\bar{A} \oplus B$
d) $[A \wedge B] \vee [C \wedge D]$
e) $[A \vee B] \wedge [C \vee \bar{D}] \oplus B$

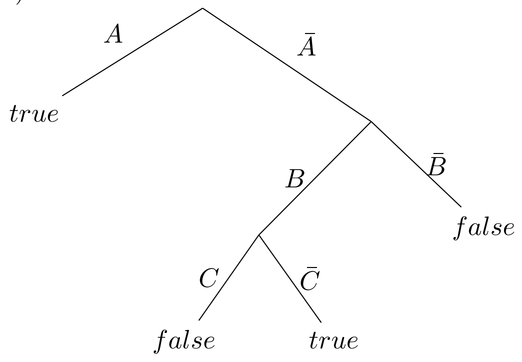
where $\bar{A}$ is a negation of A and $\oplus$ is an exclusive OR operation.

Solution: I will use the convention that a path from the left of a node for attribute A corresponds to that attribute being true (A), and a path to the right corresponds to the attribute being false ($\bar{A}$).

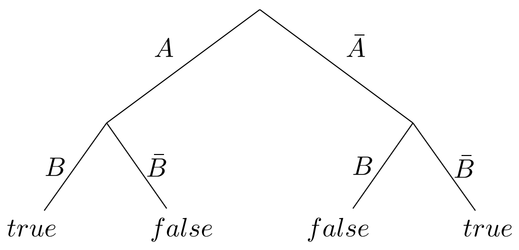
a)



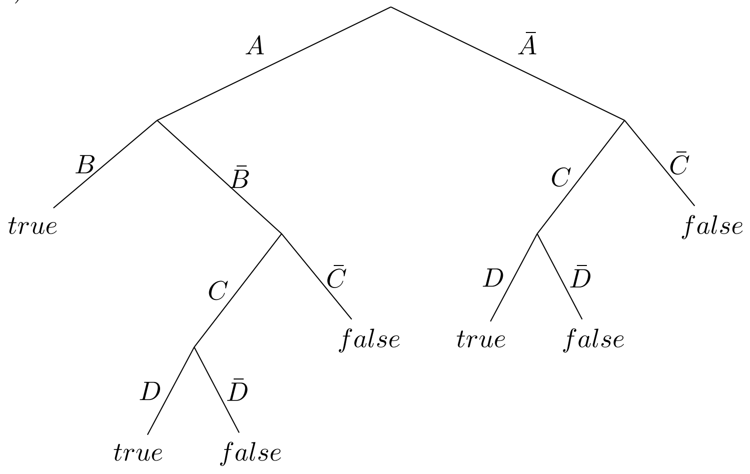
b)



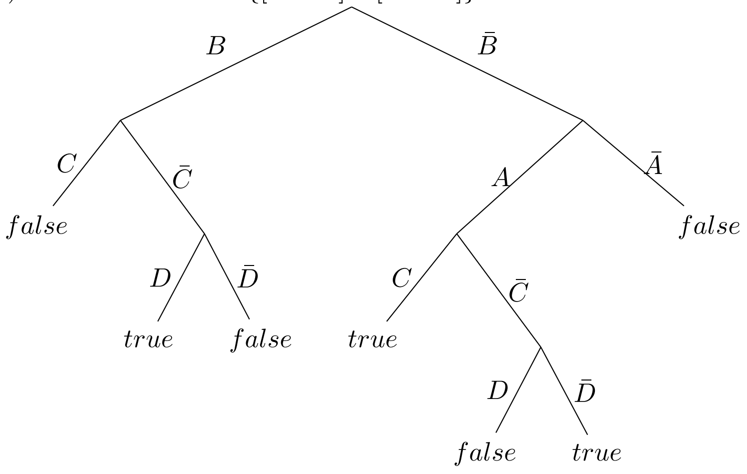
c)



d)



e) **Note:** read it like $\{[A \vee B] \wedge [C \vee \bar{D}]\} \oplus B$



Problem 2: Consider the data set from the following table

	Sky	Temperature	Humidity	Wind	Water	Forecast	Enjoy Sport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

a) Using *Enjoy Sport* as the target, show the decision tree that would be learned if the splitting criterion was information gain

b) Add the training example from the table below and compute the new decision tree. This time, show the value of the information gain for each candidate attribute at each step in growing the tree.

	Sky	Temperature	Humidity	Wind	Water	Forecast	Enjoy Sport
1	Sunny	Warm	Normal	Weak	Warm	Same	No

Solution: I will use the notation from the Tom Mitchel's book to denote a set with positive and negative examples, e.g., $S = [3+, 1-]$ in the first data set (there are 3 positive examples corresponding to *Enjoy Sport* = yes, and 1 negative example corresponding to *Enjoy Sport* = no).

a) Let $S = [3+, 1-]$. Then, $Entropy(S) = -\frac{3}{4}log_2\left(\frac{3}{4}\right) - \frac{1}{4}log_2\left(\frac{1}{4}\right) \approx 0.8113$.

Clearly, if the splitting criterion is information gain we will split either by attribute *Sky* or *Temperature* since both of these partition the training set immediately. Nonetheless, let us check that the math works. We will use the following formula to compute information gain for attribute A:

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

i. $Values(Sky) = \{Sunny, Rainy\}$.
$S_{Sunny} \leftarrow [3+, 0-] \Longrightarrow Entropy(S_{Sunny}) = 0$.
$S_{Rainy} \leftarrow [0+, 1-] \Longrightarrow Entropy(S_{Rainy}) = 0$.
$Gain(S, Sky) = 0.8113 - 0 - 0 \Longrightarrow \boxed{Gain(S, Sky) = 0.8113}$

ii. $Values(Temperature) = \{Warm, Cold\}$.
$S_{Warm} \leftarrow [3+, 0-] \Longrightarrow Entropy(S_{Warm}) = 0$.
$S_{Cold} \leftarrow [0+, 1-] \Longrightarrow Entropy(S_{Cold}) = 0$.
$Gain(S, Temperature) = 0.8113 - 0 - 0 \Longrightarrow \boxed{Gain(S, Temperature) = 0.8113}$

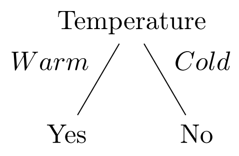
iii. $Values(Humidity) = \{Normal, High\}$.
$S_{Normal} \leftarrow [1+, 0-] \Longrightarrow Entropy(S_{Normal}) = 0$.
$S_{High} \leftarrow [2+, 1-] \Longrightarrow Entropy(S_{High}) = -\frac{2}{3}log_2\left(\frac{2}{3}\right) - \frac{1}{3}log_2\left(\frac{1}{3}\right) \approx 0.9183$.
$Gain(S, Humidity) = 0.8113 - 0 - \frac{3}{4}0.9183 \Longrightarrow \boxed{Gain(S, Humidity) = 0.1226}$

iv. $Values(Wind) = \{Strong\}$.
$S_{Strong} \leftarrow [3+, 1-] \Longrightarrow Entropy(S_{Strong}) = -\frac{3}{4}log_2\left(\frac{3}{4}\right) - \frac{1}{4}log_2\left(\frac{1}{4}\right) \approx 0.8113$.
$Gain(S, Wind) = 0.8113 - 0.8113 \Longrightarrow \boxed{Gain(S, Wind) = 0}$

v. $Values(Water) = \{Warm, Cool\}$.
$S_{Warm} \leftarrow [2+, 1-] \Longrightarrow Entropy(S_{Warm}) = -\frac{2}{3}log_2\left(\frac{2}{3}\right) - \frac{1}{3}log_2\left(\frac{1}{3}\right) \approx 0.9183$.
$S_{Cool} \leftarrow [1+, 0-] \Longrightarrow Entropy(S_{Cool}) = 0$.
$Gain(S, Water) = 0.8113 - 0 - \frac{3}{4}0.9183 \Longrightarrow \boxed{Gain(S, Water) = 0.1226}$

vi. $Values(Forecast) = \{Same, Change\}$.
$S_{Same} \leftarrow [2+, 0-] \Longrightarrow Entropy(S_{Same}) = 0$.
$S_{Change} \leftarrow [1+, 1-] \Longrightarrow Entropy(S_{Change}) = -\frac{1}{2}log_2\left(\frac{1}{2}\right) - \frac{1}{2}log_2\left(\frac{1}{2}\right) = 1$.
$Gain(S, Forecast) = 0.8113 - 0 - \frac{1}{2}1 \Longrightarrow \boxed{Gain(S, Forecast) = 0.3113}$

We have a tie between *Sky* and *Temperature* for the attributes with higher Information Gain. Choose one randomly, say *Temperature* to get the decision tree for the concept *Enjoy Sport*:



b) Let us go through the same process as before but adding the new training example.
Let $S = [3+, 2-]$. Then, $Entropy(S) = -\frac{3}{5}log_2\left(\frac{3}{5}\right) - \frac{2}{5}log_2\left(\frac{2}{5}\right) \approx 0.9710$.

i. $Values(Sky) = \{Sunny, Rainy\}$.
$S_{Sunny} \leftarrow [3+, 1-] \Longrightarrow Entropy(S_{Sunny}) = -\frac{3}{4}log_2\left(\frac{3}{4}\right) - \frac{1}{4}log_2\left(\frac{1}{4}\right) \approx 0.8113$.
$S_{Rainy} \leftarrow [0+, 1-] \Longrightarrow Entropy(S_{Rainy}) = 0$.
$Gain(S, Sky) = 0.9710 - \frac{4}{5}0.8113 - 0 \Longrightarrow \boxed{Gain(S, Sky) = 0.3220}$

ii. $Values(Temperature) = \{Warm, Cold\}$.
$S_{Warm} \leftarrow [3+, 1-] \Longrightarrow Entropy(S_{Warm}) = -\frac{3}{4}log_2\left(\frac{3}{4}\right) - \frac{1}{4}log_2\left(\frac{1}{4}\right) \approx 0.8113$.
$S_{Cold} \leftarrow [0+, 1-] \Longrightarrow Entropy(S_{Cold}) = 0$.
$Gain(S, Temperature) = 0.9710 - \frac{4}{5}0.8113 - 0 \Longrightarrow \boxed{Gain(S, Temperature) = 0.3220}$

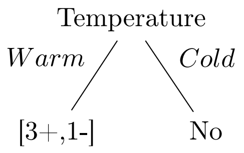
iii. $Values(Humidity) = \{Normal, High\}$.
$S_{Normal} \leftarrow [1+, 1-] \Longrightarrow Entropy(S_{Normal}) = -\frac{1}{2}log_2\left(\frac{1}{2}\right) - \frac{1}{2}log_2\left(\frac{1}{2}\right) = 1$.
$S_{High} \leftarrow [2+, 1-] \Longrightarrow Entropy(S_{High}) = -\frac{2}{3}log_2\left(\frac{2}{3}\right) - \frac{1}{3}log_2\left(\frac{1}{3}\right) \approx 0.9183$.
$Gain(S, Humidity) = 0.9710 - \frac{2}{5}1 - \frac{3}{5}0.9183 \Longrightarrow \boxed{Gain(S, Humidity) = 0.0200}$

iv. $Values(Wind) = \{Strong, Weak\}$.
$S_{Strong} \leftarrow [3+, 1-] \Longrightarrow Entropy(S_{Strong}) = -\frac{3}{4}log_2\left(\frac{3}{4}\right) - \frac{1}{4}log_2\left(\frac{1}{4}\right) \approx 0.8113$.
$S_{Weak} \leftarrow [0+, 1-] \Longrightarrow Entropy(S_{Strong}) = 0$.
$Gain(S, Wind) = 0.9710 - \frac{4}{5}0.8113 - 0 \Longrightarrow \boxed{Gain(S, Wind) = 0.3220}$

v. $Values(Water) = \{Warm, Cool\}$.
$S_{Warm} \leftarrow [2+, 2-] \Longrightarrow Entropy(S_{Warm}) = -\frac{2}{4}log_2\left(\frac{2}{4}\right) - \frac{2}{4}log_2\left(\frac{2}{4}\right) = 1$.
$S_{Cool} \leftarrow [1+, 0-] \Longrightarrow Entropy(S_{Cool}) = 0$.
$Gain(S, Water) = 0.9710 - \frac{4}{5}1 - 0 \Longrightarrow \boxed{Gain(S, Water) = 0.1710}$

vi. $Values(Forecast) = \{Same, Change\}$.
$S_{Same} \leftarrow [2+, 1-] \Longrightarrow Entropy(S_{Same}) = -\frac{2}{3}log_2\left(\frac{2}{3}\right) - \frac{1}{3}log_2\left(\frac{1}{3}\right) \approx 0.9183$.
$S_{Change} \leftarrow [1+, 1-] \Longrightarrow Entropy(S_{Change}) = -\frac{1}{2}log_2\left(\frac{1}{2}\right) - \frac{1}{2}log_2\left(\frac{1}{2}\right) = 1$.
$Gain(S, Forecast) = 0.9710 - \frac{3}{5}0.9183 - \frac{2}{5}1 \Longrightarrow \boxed{Gain(S, Forecast) = 0.0200}$

There is a three way tie between attributes *Sky*, *Temperature* and *Wind*. I am choosing *Temperature* first:



Now we have to repeat the process but considering the S to contain only points $S = \{1, 2, 4, 5\}$
In this case $S = [3+, 1-]$. Then, $Entropy(S) = -\frac{3}{4}log_2\left(\frac{3}{4}\right) - \frac{1}{4}log_2\left(\frac{1}{4}\right) \approx 0.8113.$

i. $Values(Sky) = \{Sunny, \}.$
$S_{Sunny} \leftarrow [3+, 1-] \Longrightarrow Entropy(S_{Sunny}) = -\frac{3}{4}log_2\left(\frac{3}{4}\right) - \frac{1}{4}log_2\left(\frac{1}{4}\right) \approx 0.8113.$
$Gain(S, Sky) = 0.8113 - \frac{4}{4}0.8113 \Longrightarrow \boxed{Gain(S, Sky) = 0}$

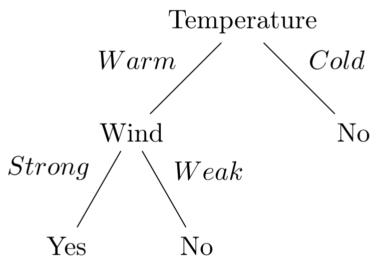
ii. $Values(Humidity) = \{Normal, High\}.$
$S_{Normal} \leftarrow [1+, 1-] \Longrightarrow Entropy(S_{Normal}) = -\frac{1}{2}log_2\left(\frac{1}{2}\right) - \frac{1}{2}log_2\left(\frac{1}{2}\right) = 1.$
$S_{High} \leftarrow [2+, 0-] \Longrightarrow Entropy(S_{High}) = 0.$
$Gain(S, Humidity) = 0.8113 - \frac{2}{4}1 - 0 \Longrightarrow \boxed{Gain(S, Humidity) = 0.3113}$

iii. $Values(Wind) = \{Strong, Weak\}.$
$S_{Strong} \leftarrow [3+, 0-] \Longrightarrow Entropy(S_{Strong}) = 0.$
$S_{Weak} \leftarrow [0+, 1-] \Longrightarrow Entropy(S_{Strong}) = 0.$
$Gain(S, Wind) = 0.8113 - 0 - 0 \Longrightarrow \boxed{Gain(S, Wind) = 0.8113}$

iv. $Values(Water) = \{Warm, Cool\}.$
$S_{Warm} \leftarrow [2+, 1-] \Longrightarrow Entropy(S_{Warm}) = -\frac{2}{3}log_2\left(\frac{2}{3}\right) - \frac{1}{3}log_2\left(\frac{1}{3}\right) \approx 0.9183.$
$S_{Cool} \leftarrow [1+, 0-] \Longrightarrow Entropy(S_{Cool}) = 0.$
$Gain(S, Water) = 0.8113 - \frac{3}{4}0.9183 - 0 \Longrightarrow \boxed{Gain(S, Water) = 0.1226}$

v. $Values(Forecast) = \{Same, Change\}.$
$S_{Same} \leftarrow [2+, 1-] \Longrightarrow Entropy(S_{Same}) = -\frac{2}{3}log_2\left(\frac{2}{3}\right) - \frac{1}{3}log_2\left(\frac{1}{3}\right) \approx 0.9183.$
$S_{Change} \leftarrow [1+, 0-] \Longrightarrow Entropy(S_{Change}) = 0.$
$Gain(S, Forecast) = 0.9710 - \frac{3}{4}0.9183 - 0 \Longrightarrow \boxed{Gain(S, Forecast) = 0.2823}$

Attribute *Wind* has the highest information gain, so we separate by this attribute:



All training examples are correctly classified. Hence, this is the final tree.

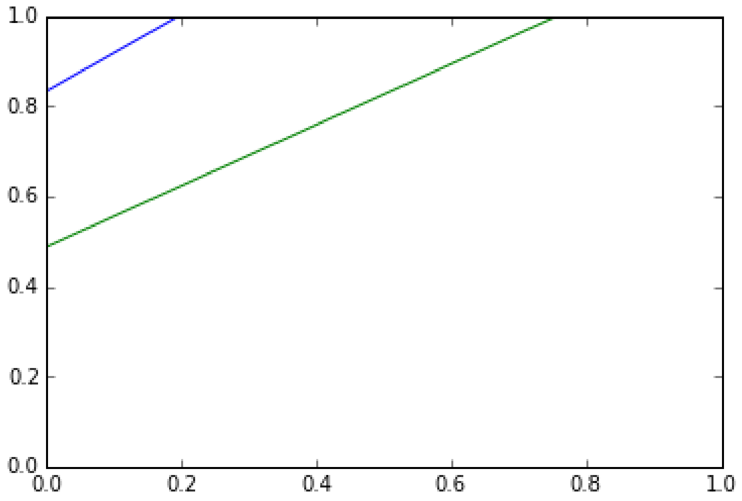
Problem 3: Consider a two-layer feed-forward neural network with two inputs x_1 and x_2, one hidden unit z, and one output unit f. This network has five weights $(w_{1z}, w_{2z}, w_{0z}, w_{zf}, w_{0f})$ where w_{0z} represents the bias for unit z and where w_{0f} represents the bias for unit f. Initialize these weights to the values (0.1, -0.1, 0.1, -0.1, 0.1), then give their values after each of the first two training iterations of the backpropagation algorithm. Assume learning rate $\eta = 0.3$, momentum $\mu = 0.9$, incremental weight updates, and the following training examples

$$\{((x_1, x_2)_i, y_i)\}_{i=1}^{2} = \{((1,0),1), ((0,1),0)\}$$

Solution: Using the tool pybrain in python we can obtain the weights of two iterations of Backpropagation:

#	w_{1z}	w_{2z}	w_{0z}	w_{zf}	w_{0f}
1	0.087	-0.102	0.085	0.137	0.520
2	0.083	-0.123	0.060	0.266	0.710

Note that this neural network has only one hidden unit, so it will learn a linear concept analogous to logistic regression. It is interesting to see the lines generated by the hidden unit for the first two iterations:



We start to see how the neural network adapts to the data, which in this case is linearly separable. The green line correspond to the second iteration and the blue to the first iteration.

Problem 4: Consider minimizing the following regularized error function using the backpropagation algorithm

$$E = \sum_{i=1}^{n}\sum_{j=1}^{m}(y_{ij} - f_{ij})^2 + \gamma \sum_{i,j} w_{i,j}^2$$

Derive the gradient descent update rule for this error function. Show that it can be implemented by multiplying each weight by some constant before performing the standard gradient descent update as shown in class.

Solution: Following the notation used in class, let us minimize the above regularized error function. Note:
We will use the notation b_{uv} for the weights from hidden unit Z_u to output f_v.
We will use the notation a_{uv} for the weights from input X_u to hidden unit Z_v.
Now we optimize. First for $b's$ and later for $a's$:

$$\begin{aligned}
\frac{\partial E}{\partial b_{uv}} &= \frac{\partial}{\partial b_{uv}}\left[\sum_{i=1}^{n}\sum_{j=1}^{m}(y_{ij} - f_{ij})^2\right] + \frac{\partial}{\partial b_{uv}}\left[\gamma \sum_{i,j} w_{i,j}^2\right] \\
&= \sum_{i=1}^{n}\frac{\partial}{\partial b_{uv}}\left[\sum_{j=1}^{m}(y_{ij} - f_{ij})^2\right] + \gamma \sum_{i,j}\frac{\partial}{\partial b_{uv}}\left[w_{i,j}^2\right] \\
&= \left[\sum_{i=1}^{n}\frac{\partial}{\partial b_{uv}}\left((y_{i1} - f_{i1})^2 + (y_{i2} - f_{i2})^2 + \cdots + (y_{im} - f_{im})^2\right)\right] + 2\gamma b_{uv} \\
&= \left[\sum_{i=1}^{n} -2(y_{iu} - f_{iu}) \cdot \frac{\partial}{\partial b_{uv}}\varphi(\mathbf{b}_u^T \mathbf{z}_i)\right] + 2\gamma b_{uv} = (1)
\end{aligned}$$

Choose $\varphi(x) = \dfrac{1}{1+e^{-x}}$. Then $\varphi'(x) = \dfrac{e^{-x}}{(1+e^{-x})^2} = \dfrac{1}{1+e^{-x}} \cdot \dfrac{e^{-x}}{1+e^{-x}} = \varphi(x)(1-\varphi(x))$

$$(1) \quad = \quad \left[\sum_{i=1}^{n} -2(y_{iu} - f_{iu})\varphi(\mathbf{b}_u^T \mathbf{z}_i)(1-\varphi(\mathbf{b}_u^T \mathbf{z}_i))z_{iv}\right] + 2\gamma b_{uv} = (2)$$

Note that $f_{iu} = \varphi(\mathbf{b}_u^T \mathbf{z}_i)$ which means that, $1 - f_{iu} = (1-\varphi(\mathbf{b}_u^T \mathbf{z}_i))$ Thus,

$$(2) \quad = \quad \left[\sum_{i=1}^{n} -2(y_{iu} - f_{iu})f_{iu}(1-f_{iu})z_{iv}\right] + 2\gamma b_{uv}$$

Let $\beta_{iu} = -2(y_{iu} - f_{iu})f_{iu}(1-f_{iu})$ to conclude that

$$\frac{\partial E}{\partial b_{uv}} = \left[\sum_{i=1}^{n} \beta_{iu} z_{iv}\right] + 2\gamma b_{uv}$$

The update rule for gradient descent is

$$b_{uv}^{(t+1)} = b_{uv}^{(t)} - \eta \left\{\left[\sum_{i=1}^{n} \beta_{iu} z_{iv}\right] + 2\gamma b_{uv}^{(t)}\right\} = b_{uv}^{(t)} - \eta \left[\sum_{i=1}^{n} \beta_{iu} z_{iv}\right] - 2\eta\gamma b_{uv}^{(t)} = (1-2\eta\gamma)b_{uv}^{(t)} - \eta \left[\sum_{i=1}^{n} \beta_{iu} z_{iv}\right]$$

Optimization for $a's$:

$$\begin{aligned}
\frac{\partial E}{\partial a_{uv}} &= \frac{\partial}{\partial a_{uv}}\left[\sum_{i=1}^{n}\sum_{j=1}^{m}(y_{ij} - f_{ij})^2\right] + \frac{\partial}{\partial a_{uv}}\left[\gamma \sum_{i,j} w_{i,j}^2\right] \\
&= \sum_{i=1}^{n} \frac{\partial}{\partial a_{uv}}\left[\sum_{j=1}^{m}(y_{ij} - f_{ij})^2\right] + \gamma \sum_{i,j} \frac{\partial}{\partial a_{uv}}\left[w_{i,j}^2\right] \\
&= \sum_{i=1}^{n} -2\left[\sum_{j=1}^{m}(y_{ij} - f_{ij}) \cdot \frac{\partial}{\partial a_{uv}}\left(\varphi(\mathbf{b}_j^T \mathbf{z}_i)\right)\right] + 2\gamma a_{uv} = (3)
\end{aligned}$$

By definition $\mathbf{b}_j^T \mathbf{z}_i = \sum_{l=1}^{h} b_{jl} z_{il}$. Thus,

$$\begin{aligned}
\frac{\partial}{\partial a_{uv}}\left(\varphi(\mathbf{b}_j^T \mathbf{z}_i)\right) &= \varphi'(\mathbf{b}_j^T \mathbf{z}_i)\frac{\partial}{\partial a_{uv}}\mathbf{b}_j^T \mathbf{z}_i \\
&= \varphi'(\mathbf{b}_j^T \mathbf{z}_i)\frac{\partial}{\partial a_{uv}}\left(\sum_{l=1}^{h} b_{jl} z_{il}\right) \\
&= \varphi'(\mathbf{b}_j^T \mathbf{z}_i) b_{ju} \frac{\partial}{\partial a_{uv}}\left(z_{iu}\right) \\
&= \varphi'(\mathbf{b}_j^T \mathbf{z}_i) b_{ju} \frac{\partial}{\partial a_{uv}}\left(\varphi(\mathbf{a}_u^T \mathbf{x}_i)\right) \\
&= \varphi'(\mathbf{b}_j^T \mathbf{z}_i) b_{ju} \varphi'(\mathbf{a}_u^T \mathbf{x}_i) x_{iv} = (4)
\end{aligned}$$

Replacing (4) into (3) and rearranging:

$$\begin{aligned}
\frac{\partial E}{\partial a_{uv}} &= \sum_{i=1}^{n} -2\left[\sum_{j=1}^{m}(y_{ij}-f_{ij})\varphi'(\mathbf{b}_j^T\mathbf{z}_i)b_{ju}\varphi'(\mathbf{a}_u^T\mathbf{x}_i)x_{iv}\right]+2\gamma a_{uv} \\
&= \sum_{i=1}^{n}\left[\sum_{j=1}^{m}\left\{-2(y_{ij}-f_{ij})\varphi'(\mathbf{b}_j^T\mathbf{z}_i)\right\}b_{ju}\varphi'(\mathbf{a}_u^T\mathbf{x}_i)x_{iv}\right]+2\gamma a_{uv} \\
&= \sum_{i=1}^{n}\left[\sum_{j=1}^{m}\beta_{ij}b_{ju}\varphi'(\mathbf{a}_u^T\mathbf{x}_i)x_{iv}\right]+2\gamma a_{uv} = (5)
\end{aligned}$$

Let $\alpha_{iu} = \beta_{ij}b_{ju}\varphi'(\mathbf{a}_u^T\mathbf{x}_i)x_{iv} = \beta_{ij}b_{ju}z_{iu}(1-z_{iu})x_{iv}$, and replace in (5) to conclude that

$$\frac{\partial E}{\partial a_{uv}} = \left[\sum_{i=1}^{n}\alpha_{iu}x_{iv}\right]+2\gamma a_{uv}$$

The update rule for gradient descent is

$$a_{uv}^{(t+1)} = a_{uv}^{(t)} - \eta\left\{\left[\sum_{i=1}^{n}\alpha_{iu}x_{iv}\right]+2\gamma a_{uv}^{(t)}\right\} = a_{uv}^{(t)} - \eta\left[\sum_{i=1}^{n}\alpha_{iu}x_{iv}\right] - 2\eta\gamma a_{uv}^{(t)} = (1-2\eta\gamma)a_{uv}^{(t)} - \eta\left[\sum_{i=1}^{n}\alpha_{iu}x_{iv}\right]$$

The gradient descent update rule for this error function is:

$$a_{uv}^{(t+1)} = (1-2\eta\gamma)a_{uv}^{(t)} - \eta\sum_{i=1}^{n}\alpha_{iu}x_{iv}$$

$$b_{uv}^{(t+1)} = (1-2\eta\gamma)b_{uv}^{(t)} - \eta\sum_{i=1}^{n}\beta_{iu}z_{iv}$$

b : for every $u \in \{1,2,\ldots,m\}$ and for every $v \in \{1,2,\ldots,h\}$
a : for every $u \in \{1,2,\ldots,h\}$ and for every $v \in \{1,2,\ldots,k\}$

This shows that the minimization of the regularized error function can be implemented by multiplying each weight by the constant $(1-2\eta\gamma)$ before performing the standard gradient descent update.

Problem 5: Using the neural network code from class as a starting point, compare the following ensemble methods in a binary classification scenario: 30 bagged neural networks with 30 bagged regression trees. In each case, the final decision is made by averaging the raw outputs from the ensemble members. Use at least 3 different classification data sets to provide comparisons. Select appropriate data sets from the UCI Machine Learning Repository. Proper comparisons should be made using 10-fold cross-validation experiments. Use your knowledge about model comparison to formally conclude which of the two algorithms is better. Normalize the variables on the training set, then normalize the test set using the normalization parameters from the training set (e.g. mean and standard deviation for each feature). If data sets have categorical variables, provide a proper approach to convert them into numerical variables. Use classification error to evaluate quality of classifiers.

Solution: All the code for this problem was produced using MatLab. As usual, you can find the code attached to the OnCourse submission.

I tested the following three data sets:

1. Spambase: https://archive.ics.uci.edu/ml/datasets/Spambase
2. Transfusion: https://archive.ics.uci.edu/ml/datasets/Blood+Transfusion+Service+Center
3. Ionosphere: https://archive.ics.uci.edu/ml/datasets/Ionosphere

Results on each of the sets follow:

Spambase

Run #	NN	Trees
1	0.084547	0.012606
2	0.082591	0.013475
3	0.079548	0.012823
4	0.082156	0.0097805
5	0.081287	0.012171
6	0.084982	0.010867
7	0.079331	0.012171
8	0.081287	0.011737
9	0.082156	0.012606
10	0.081504	0.0099978
11	0.083895	0.010867
12	0.084112	0.01391
13	0.082591	0.013041
14	0.084764	0.012823
15	0.084112	0.012171
16	0.082808	0.013693
17	0.078461	0.011085
18	0.084112	0.013475
19	0.079548	0.012389
20	0.083243	0.012171
21	0.086286	0.012606
22	0.084764	0.012389
23	0.08759	0.011302
24	0.081939	0.012171
25	0.080852	0.012171

Transfusion

Run #	NN	Trees
1	0.2139	0.13503
2	0.21123	0.12567
3	0.21524	0.1377
4	0.20455	0.13636
5	0.21257	0.13636
6	0.21257	0.13235
7	0.21791	0.13369
8	0.21524	0.12701
9	0.20722	0.13235
10	0.2139	0.13503
11	0.21524	0.13235
12	0.2139	0.13235
13	0.21257	0.13102
14	0.21123	0.13235
15	0.21257	0.12299
16	0.21257	0.12834
17	0.20989	0.13235
18	0.21123	0.12968
19	0.21524	0.12567
20	0.21123	0.13904
21	0.2139	0.12032
22	0.21257	0.12299
23	0.21524	0.13235
24	0.2139	0.13235
25	0.20856	0.12968

Ionosphere

Run #	NN	Trees
1	0.048433	0.014245
2	0.045584	0.005698
3	0.045584	0.011396
4	0.037037	0.005698
5	0.039886	0.014245
6	0.045584	0.008547
7	0.051282	0.002849
8	0.037037	0.011396
9	0.045584	0.014245
10	0.051282	0.008547
11	0.025641	0.011396
12	0.054131	0.008547
13	0.039886	0.011396
14	0.054131	0.014245
15	0.045584	0.008547
16	0.039886	0.005698
17	0.039886	0.008547
18	0.039886	0.008547
19	0.05698	0.017094
20	0.042735	0.011396
21	0.051282	0.008547
22	0.054131	0.008547
23	0.039886	0.014245
24	0.048433	0.008547
25	0.045584	0.014245

These results corresponds to 25 runs of the corresponding bagged algorithm. Each bag consisted of 30 models, i.e., either 30 Neural networks or 30 Trees respectively. Neural Networks are feedforwad networks with 1 hidden layers of 10 neurons and trained using Scaled Conjugate Gradient. The models for the bags were selected by 10-fold cross-validation. Note that all the data was normalized before training the models.

All of the datasets are binary and thus, the final decision was made by taking a majority vote of individual decisions from the 30 constituents of a single bag. In case of a tie, one class was chosen randomly and with equal probability. For each run we have the classification error defined as:

$$E = \frac{\#\text{examples incorrectly classified}}{\#\text{total data set}}$$

For example, for the Spambase, run 1, the bagged Neural Network (NN) misclassified 0.084547 (about 8.4% of examples) whereas the bag of Trees misclassified 0.012606 (about 1.2% of examples).

Just inspecting the data one can conclude that Trees outperformed NN in every case. For mathematical sound evidence, let us perform two different statistical tests:

1. Let us hypothesize that NN perform the same as Trees, i.e.,

$$H_0 : T = NN \text{ vs. } H_1 : T > NN$$

   Suppose that H_0 is true. In this case the probability that Trees won 25 times out of 25 is given by:

$$p = \sum_{i=25}^{25} \binom{25}{i} \left(\frac{1}{2}\right)^{i} \left(\frac{1}{2}\right)^{25-i} = \binom{25}{25} \left(\frac{1}{2}\right)^{25} \left(\frac{1}{2}\right)^{25-25} = \left(\frac{1}{2}\right)^{25} = 2.98023223876953125 \times 10^{-8}$$

   Clearly, this is a small enough probability at any reasonable level. Conclude that it is not the case that $T = NN$. Note also that this test apply to all 3 data sets since in all of these Trees won every time.

2. The next test is a T-test for the means of two independent samples. The samples in this case are the results for NN and trees. For this I used the following function provided by the Python package SciPy:

$$(ttest, pvalue) = ttest\_ind(data\_nn[i], data\_trees[i])$$

   Where $data\_nn[i]$ is a vector of 25 numbers with the results for bagged neural networks. Likewise, $data\_trees[i]$ is a vector of 25 numbers with the bagged results for trees.

   According to the function documentation (see [4.]), this function: "Calculates the T-test for the means of TWO INDEPENDENT samples of scores. This is a two-sided test for the null hypothesis that 2 independent samples have identical average (expected) values. This test assumes that the populations have identical variances ... We can use this test, if we observe two independent samples from the same or different population, e.g. exam scores of boys and girls or of two ethnic groups. The test measures whether the average (expected) value differs significantly across samples. If we observe a large p-value, for example larger than 0.05 or 0.1, then we cannot reject the null hypothesis of identical average scores. If the p-value is smaller than the threshold, e.g. 1%, 5% or 10%, then we reject the null hypothesis of equal averages."

   The results of the test for each data set are:

dataset	*ttest*	*pvalue*
spambase	142.1723269	$1.12071685 \times 10^{-64}$
transfusion	73.26719721	$6.34244720 \times 10^{-51}$
ionosphere	22.03061111	$9.55472889 \times 10^{-27}$

   Clearly and in all cases we reject the hypothesis that the mean of the classification error of the algorithms are the same.

Together tests (1) and (2) provide strong evidence that bagged Trees outperform bagged Neural Networks for all 3 datasets tested. This may be due to the fact that these are small datasets where Trees have a greater chance of fitting the data. An alternative reason is that we would need more than 10 neurons to obtain better results for neural networks.

References

[1.] www.pybrain.org

[2.] Tom Mitchel's book chapter on decision trees

[3.] UC Irvine Machine Learning Repository: https://archive.ics.uci.edu/ml/index.html

[4.] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html#scipy.stats.ttest_ind