Artificial Intelligence—Spring 2022

Homework 6

Issued: May. 9th, 2022 Due: May. 16th, 2022

Problem 1

Solutions:

∵ When it comes to the information gain from the attribute test on A_i, we have

$$Gain(A_i) = B(\frac{p}{p+n}) - Remainder(A_i)$$

where p and n are the number of positive examples and negative examples respectively, and:

$$Remainder(A_i) = \Sigma_{k=1}^{d} \frac{p_k + n_k}{p+n} B(\frac{p_k}{p_k + n_k})$$

where p_k and n_k are the number of positive and negative examples in subset E_k, obtained by dividing set according to A_i's value.

∴ According to the title, we can obtain that:

$$\begin{aligned} Remainder(A_1) &= \frac{4}{5}B(\frac{2}{4}) + \frac{1}{5}B(\frac{0}{1}) = 0.8000 \\ Remainder(A_2) &= \frac{3}{5}B(\frac{2}{3}) + \frac{2}{5}B(\frac{0}{2}) = 0.5510 \\ Remainder(A_3) &= \frac{2}{5}B(\frac{1}{2}) + \frac{3}{5}B(\frac{1}{3}) = 0.9510 \end{aligned}$$

So the information gains are:

$$\begin{aligned} Gain(A_1) &= B(\frac{2}{5}) - Remainder(A_1) = 0.9710 - 0.8000 = 0.1710 \\ Gain(A_2) &= B(\frac{2}{5}) - Remainder(A_2) = 0.9710 - 0.5510 = 0.4200 \\ Gain(A_3) &= B(\frac{2}{5}) - Remainder(A_3) = 0.9710 - 0.9510 = 0.0200 \end{aligned}$$

In this case, A_2 is a better attribute to spilt on and should be chosen as the root.

Note that when A_2 is 0, the outputs are 0, too. So we can only consider the cases when $A_2 = 1$, that is, $\boldsymbol{x}_3$ to $\boldsymbol{x}_5$.

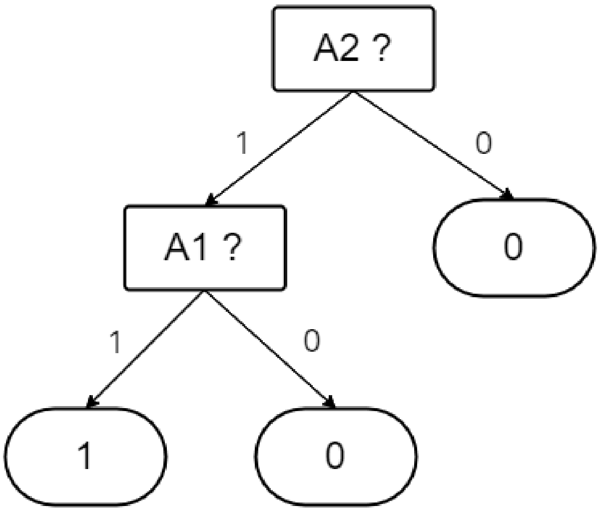
The remainders are:

$$Remainder(A_1) = \frac{2}{3}B(\frac{2}{2}) + \frac{1}{3}B(\frac{0}{1}) = 0$$
$$Remainder(A_3) = \frac{1}{3}B(\frac{1}{1}) + \frac{2}{3}B(\frac{1}{2}) = 0.6667$$

So the information gains are:

$$Gain(A_1) = B(\frac{2}{3}) - Remainder(A_1) = 0.9183 - 0 = 0.9183$$
$$Gain(A_3) = B(\frac{2}{3}) - Remainder(A_2) = 0.9183 - 0.6667 = 0.2516$$

$\therefore A_1$ is a better attribute to split on, and the decision tree can be sketched as follows:



Problem 2

Solutions:

a. According to the title, the inputs are $\boldsymbol{x}$ and the hidden layer has a weight matrix $\boldsymbol{w}$, and we have a linear activation function $g(v) = cv + d$

$\therefore$ Suppose that $w_{i,j}$ is element at the i^{th} row and the j^{th} col of the matrix $\boldsymbol{w}$, we then have:
The value of the j^{th} hidden layer unit before activation is:

$$v_{1j} = \Sigma_i w_{i,j} x_i$$

and its output is:

$$y_{1j} = g(v_{1,j}) = c\Sigma_i w_{i,j} x_i + d$$

At the output layer, we have:

$$v_k = \Sigma_j w_{j,k} y_{1j} = \Sigma_j w_{j,k} \left(c\Sigma_i w_{i,j} x_i + d\right)$$

So the final output is:

$$y_k = g(v_k) = c^2\Sigma_i x_i \Sigma_j w_{i,j} w_{j,k} + d\big(c\Sigma_j w_{j,k} + 1\big)$$

Obviously, a no-hidden-layer network with a weight of $w_{i,k} = \Sigma_j w_{i,j} w_{j,k}$ and an activate function of $g(v) = c^2 v + d\big(c\Sigma_j w_{j,k} + 1\big)$ can compute the same function.

b. When it comes to a network with an arbitrary number of hidden layers, we can reduce the network's layers like what we have done in part (a):

Assume we have a 2-layer network, so

$$y_{2k} = g(v_{2k}) = c^2\Sigma_i x_i \Sigma_j w_{i,j} w_{j,k} + d\big(c\Sigma_j w_{j,k} + 1\big)$$

Then at the output layer,

$$v_m = \Sigma_k w_{k,m} y_{2k}$$

and

$$\begin{aligned} y_m = g(v_m) &= c\Sigma_k w_{k,m} y_{2k} + d \\ &= c^3\Sigma_i x_i \Sigma_j \Sigma_k w_{i,j} w_{j,k} w_{k,m} + d\big(c^2\Sigma_j \Sigma_k w_{j,k} w_{k,m} + c\Sigma_k w_{k.m} + 1\big) \end{aligned}$$

In this case, a no-hidden-layer network with a weight of $w_{i,m} = \Sigma_j \Sigma_k w_{i,j} w_{j,k} w_{k,m}$ and an activate function of $g(v) = c^3 v + d\big(c^2\Sigma_j \Sigma_k w_{j,k} w_{k,m} + c\Sigma_k w_{k.m} + 1\big)$ will perform the same as this 2-layer net.

Similarly, we may generalize this theory to n-layer network, that is, a no-hidden-layer network with a weight of $w_{i,o} = \Sigma_{h1} \Sigma_{h2} \ldots \Sigma_{hn} w_{i,h1} w_{h1,h2} \ldots w_{hn,o}$ and an activate function of $g(v) = c^{(n+1)} v + d\big[\Sigma_{m=0}^{n} c^{(n-m)} \Sigma_{h1} \ldots \Sigma_{hn-m} w_{h1,h2} \ldots w_{hn-m,o}\big]$ can compute the same function as it, and the output is:

$$y_o = g\big(\Sigma_{i,o} w_{i,o} x_i\big)$$

which is still a linear function of the inputs.

c. For the one-layer network, the total number of weights is:

$$m_1 = n * h + h * n = 2nh$$

If we transform this one-layer net to a no-hidden-layer network, we will obtain that:

$$m_0 = n * n = n^2$$

In particular the case $h \ll n$, it is apparent that $m_1 \ll m_0$, so the original network has far less weights and will compute faster. Additionally, we will understand the mapping relationship between the inputs and the outputs more easily. As a result, the original network has its advantages as well.