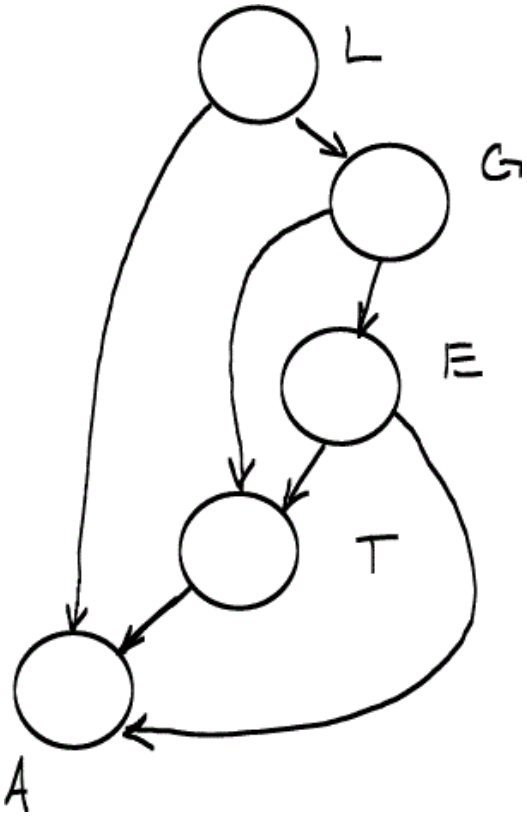
Lee Sutton
301145106

CMPT 726 – Assignment 3

Problem 1 Bayesian Network

1. A Bayesian network was drawn for the given variables. It can be seen below. The initial ordering chosen was L, G, E, T, A. The justification is as follows. Referring to the network on the following page.
    a. L was placed at the top of the page
    b. G was placed and an arrow was drawn connecting L to G. The parents' education level likely has an influence on who they voted for. Parents with university educations likely voted for the liberal party due to their middle wing ideologies. Parents' with no university education likely voted for NDP due to their left wing ideologies.
    c. Next E was drawn with an arrow connecting G to E. The government has an influence on the size of economy. Therefore, we must include some sort of dependency. However, it is most likely independent of the whether or not the parents went to university given that you know about the current government in power.
    d. Next T was drawn with an arrow drawn from G to T and E to T. The current government and the size of the economy will have an impact on the price of tuition.
    e. Lastly A was drawn and arrows were drawn from L to A, from E to A, and from T to A. Whether or not the student attends SFU will depend on the cost of tuition, and the parents will also influence the student depending on whether or not they attended university. Lastly, the size of the economy might relate to how many students are admitted to SFU.



Lee Sutton
301145106

2. Using the Bayesian network constructed above, the factored joint distribution is:

$$P(A, L, G, E, T, A) = P(L) * P(G|L) * P(E|G) * P(T|E, G) * P(A|L, T, E)$$

3. The probability distributions given below were formulated to give reasonable results given the parents are either discrete or continuous.
    a. P(L) = Discrete output. Using an educated guess:

P(L = u)	0.4
P(L = o)	0.6

    b. P(G|L) = Discrete outputs. Using an educated guess:

	L = o	L = u
G = Liberal	0.3	0.1
G = NDP	0.2	0.4

    c. P(E|G) = continuous output with discrete parents. The resulting system could have 2 different Gaussian distributions based on the values of the parents. The table below gives two possible normal distributions of the GDP (measured in billions) of British Columbia given the current government.

G = liberal	G = NDP
P(E|G = liberal) = N(μ = 275, σ = 20)	P(E|G = NDP) = N(μ = 250, σ = 15)

    d. P(T|E,G) = continuous output with discrete parents and continuous parents. The resultant distribution could include 2 linear Gaussians. One for G = liberal and one for G = NDP. As the size of the economy increases, the tuition might also increase and there is a probability associated with this point. One possibility is shown below in the table.

G = liberal	G = NDP
P(T|E, G) =N(T; 3000E; 100)	P(T|E, G) =N(T; 4000E; 120)

    e. P(A|L, A, T) = Discrete output with continuous parents and discrete parents. To model this distribution we can use a two multi-variate sigmoids. One for L = u and and for L = o. This would allow for two continuous inputs and the value of the sigmoid could be used to give a discrete output. One possibility is shown below. The following equations assume:
        i. If the parents went to university, the students are more likely to go to university regardless of the price and the current state of the economy.
        ii. An increase in tuition will decrease the probability of the student attending university.

iii. If the economy is doing well, people are more likely to invest in an education.

For L = u:

$$P(A|L,T,E) = \frac{1}{1 + \exp(\boldsymbol{w}^T * \boldsymbol{x})}$$

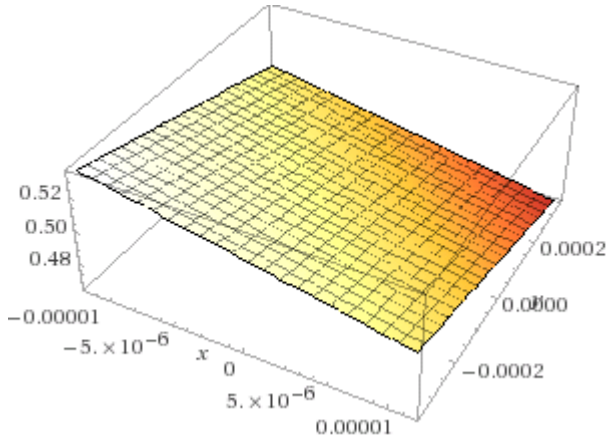
Where:

$$\boldsymbol{w} = \begin{pmatrix} 6000 \\ 250 \end{pmatrix}$$

$$\boldsymbol{x} = \begin{pmatrix} t \\ e \end{pmatrix}$$

- $t = tuition$ ($)
- $e = GDP$ $(billions\ of$ $)$

The resulting equation was plotted using Wolfram Alpha:



For L = 0:

$$P(A|L,T,E) = \frac{1}{1 + \exp(\boldsymbol{w}^T * \boldsymbol{x})}$$

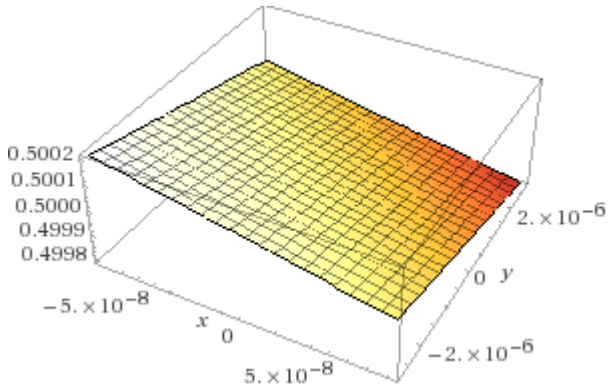
Where:

$$\boldsymbol{w} = \begin{pmatrix} 5000 \\ 230 \end{pmatrix}$$

$$\boldsymbol{x} = \begin{pmatrix} t \\ e \end{pmatrix}$$

- $t = tuition$ (\$)
- $e = GDP$ $(billions\ of$ \$)

The equation was then plotted using Wolfram Alpha:



4. To find the maximum likelihood estimates for the parameters, the maximum likelihood equation must first be generated. This function is given below. The maximum likelihood estimate examines the probability of the data, given the observed training set.

$$L(\theta : D) = P(D|\theta) = \prod_{k=1}^{N} P(x_k : \theta)$$

The probability of each data point can be factored out using the Bayesian network above to give:

$$P(x_k) = P(A=a_k, L=l_k, E=e_k, T=t_k, G=g_k)$$

$$P(x_k) = P(L=l_k)\,P(G=g_k|L=l_k)\,P(E=e_k|G=g_k)\cdots$$

$$\cdots P(T=t_k|E=e_k, G=g_k)\,P(A=a_k|L=l_k, E=e_k, T=t_k)$$

Plugging this back into the maximum likelihood equation we obtain:

Lee Sutton
301145106

$$L(D:\theta) = \prod_M \overbrace{P(A[m]:\theta_A)}^{\text{LOCAL LIKELIHOOD FUNCTIONS}} \overbrace{P(G[m]\mid L[m]:\theta_{G|M})}^{\text{LOCAL LIKELIHOOD FUNCTIONS}} \ldots$$
$$\ldots P(E[m]\mid G[m]:\theta_{E|G}) P(T[m]\mid E[m],G[m]:\theta_{T|E,G}) \ldots$$
$$\ldots P(A[m]\mid L[m],T[m],E[m]:\theta_{A|L,T,E})$$

Where m indicates the function spans over all the x values from $x_1 to\ x_m$. The local likelihood function for A is then:

$$L(x_A|pa(x_A)) = \prod_M P(A[m]\mid L[m],T[m],E[m]:\theta_{A|L,T,E})$$

When we plug these equations back into the likelihood equation, we see that local likelihood functions arise which do not depend on all the given parameters of **x**. As a result, we only have to maximize the likelihood of each parameter locally. When examining P(A|parents(A)) we see that it is only a function of $a, l, t, and\ m$. As a result, only the a_n, t_n, e_n, l_n parameters of **x** are needed. To learn the parameters for $\theta_{A|L,T,E}$ one would have to maximize the likelihood function with respect to $\theta_{A|L,T,E}$. This would likely involve taking the logarithm of the function and maximizing that by taking the gradient and solving where it equals 0.

Lee Sutton
301145106

Problem 2 Neural Networks

1. After filling in the values for dW1 and dW2 the training accuracy went to 1 while the testing accuracy rose to around 0.8. The results are shown below in the figure and the output was pasted below.

   Training neural network epoch 1/10: training accuracy = 0.8420, took 8.55 seconds

   Training neural network epoch 2/10: training accuracy = 0.9140, took 8.36 seconds

   Training neural network epoch 3/10: training accuracy = 0.9420, took 8.42 seconds

   Training neural network epoch 4/10: training accuracy = 0.9760, took 8.34 seconds

   Training neural network epoch 5/10: training accuracy = 1.0000, took 8.39 seconds

   Training neural network epoch 6/10: training accuracy = 1.0000, took 8.51 seconds

   Training neural network epoch 7/10: training accuracy = 1.0000, took 8.35 seconds

   Training neural network epoch 8/10: training accuracy = 1.0000, took 8.36 seconds

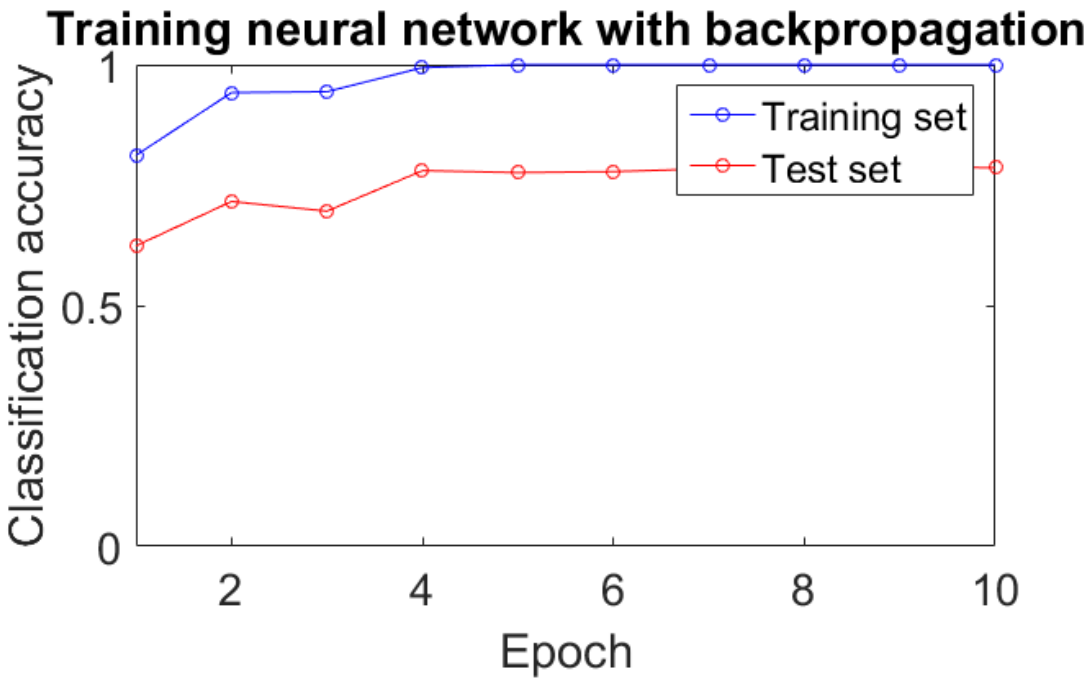
   Training neural network epoch 9/10: training accuracy = 1.0000, took 8.49 seconds

   Training neural network epoch 10/10: training accuracy = 1.0000, took 8.38 seconds

   Final test accuracy = 0.7520.

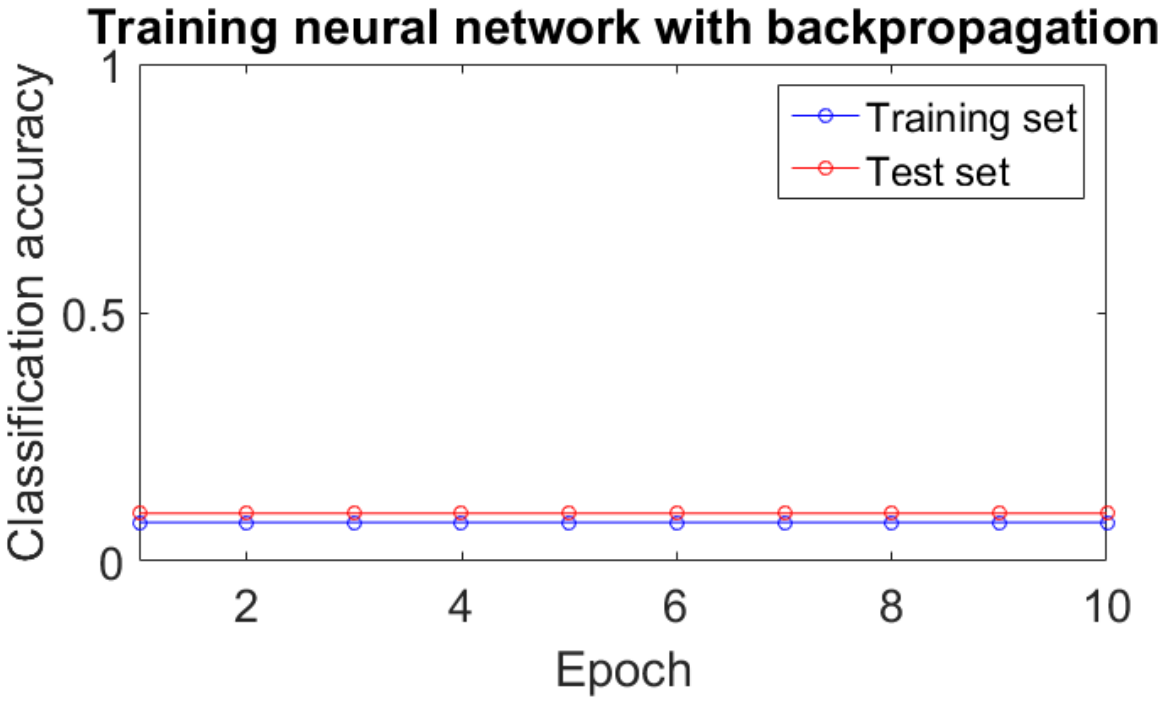
   Producing webpage of results... done.

   TRY OPENING output.html



Lee Sutton
301145106

2. When uncommenting line 40, the initial weights in layer 1 are changed from a normally distributed random set to a uniformly distributed random set. The initial weights in layer 1 with randn() were normally distributed and also contained negative numbers. With rand() all of the weights in layer 1 were assigned positive values. Since gradient descent methods are sensitive to initial conditions on the weights, this will inevitably change the output. After running the code, both the training accuracy and testing accuracy dropped to values less than 0.1 as shown below in the figure.



3. After uncommenting line 40, the variables were examined to determine why the stochastic gradient descent was not functioning properly. The values of dW2 and dW1 were examined after each iteration and they showed no change. In fact, there was no variation in the rows of dW2 even after the first iteration while dW1 contained only zeroes. When the weights in layer 1 were initialized to be positive and inputted into the first hidden activation function, all of the sigmoids became 1. This can be seen by examining the values of Z{1} and one can see that all values of Z{1} were 1. At this point the derivative of the sigmoid is nearly zero considering computational accuracy, it is essentially zero. In calculating δ_j we can see that its value will also go to zero.

$$\delta_j = h(a_j)\left(1 - h(a_j)\right) * \sum w_{kj} * \delta_k$$

Lee Sutton
301145106

The resulting stochastic gradient descent will make no changes to the values of the weights in layer 1. On the next iteration, all of the sigmoids again will be 1. By the time the x_i reaches the second layer of the neural network it looks no different than the previous x_i. The weights in the second layer can change as they are based on δ_k which is not zero however, they will have no chance to improve since the second layer will still see all 1's coming from the sigmoids in layer 1. Given that the sigmoids are all 1 in layer 1, all training inputs x will look identical by the time they get to the second layer. This was one of the issues brought up in class regarding the use of sigmoids. If given improper initial conditions, the derivatives of sigmoids approaches 0 when you move away from the steep portion. The result is a stochastic gradient descent method that makes essentially no improvement after each iteration.