Assignment 1 -- Introduction to AI

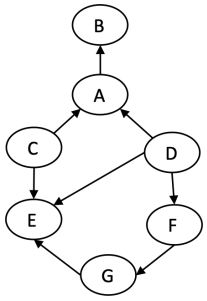
2021-22 Term 2

Note: Your solution for this assignment should have two parts---a pdf document and code files.

- Have a **single** pdf document that shows your solution for different questions (show either numerical values if the question asks for it, and/or theoretical justification as required). Include in this pdf, the code you wrote for the solution for the respective question (if coding is required).
- Upload your real code files that you used to solve the particular question. Make sure your code is neatly organized per question, runs correctly, and has comments that highlight the part you implemented so that your TA can easily understand it
- Combine your solution pdf and code files in a single zip folder and upload it on the eLearn assignment folder
- Solution should be typeset using a professional software (word, keynote, latex etc). Figures should also be made using software such as power point. **No handwritten solutions are allowed, and will not be graded.**

Question 1 [10 points]:

Consider the following Bayesian network:



Write True/False for the following conditional independence statements. Justify clearly your answer. For example, show the active/blocked trails as necessary and the reason for them to be active or blocked (e.g., use rules such as cascade, common cause or v-structure). [No coding required for this question. Each sub-part has **2.5 points**]

1. $A \perp E \mid \{C\}$
2. $A \perp E \mid \{C, D\}$
3. $F \perp E \mid \{G, D, A\}$
4. $B \perp D \mid \{A\}$

Question 2 [10 points]

A box contains three lottery machines. Each machine has four values: 1 ("*First Prize*"), 2 ("*Second Prize*"), 3 ("*Third Prize*"), and 0 ("*No-Win*"). Each machine has different configurations:

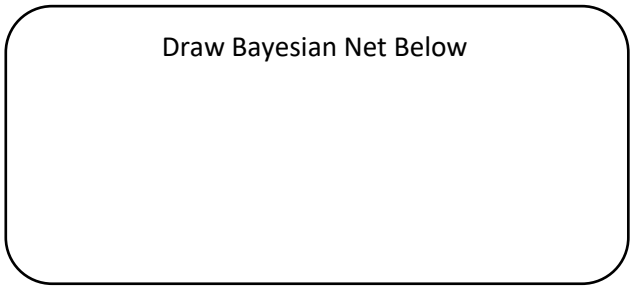
- Machine 1 (M1): has outcome 0 with probability 0.8, 1 with probability 0.05, and the rest of the outcomes with equal probability.
- Machine 2 (M2): has outcome 0 with probability 0.9, 1 with probability 0.01, 2 with probability 0.02, and 3 with probability 0.07.
- Machine 3 (M3): has outcome 0 with probability 0.7 and the rest of the outcomes with equal probability

We pick a machine from the box with the following probabilities:

- Machine M1 with probability 0.2
- Machine M2 with probability 0.7
- Machine M3 with probability 0.1

Then, we draw three outcomes (O1, O2, O3) from it.

(i) Draw the Bayes net corresponding to this setup. Explain what each random variable of this Bayes represents, and show the domain of each random variable **[3 points]**
Hint: There should be 4 random variables.



Variable Name	**Domain (Set of Values)**	**Interpretation** *(intuitive explanation of what the variable represents)*

(ii) Write conditional probabilities (numerical values) associated with each node of this Bayes net. As there are 4 variables, please specify one conditional probability table (CPT) for each variable **[3 points]**

CPT for variable 1

CPT for variable 2

CPT for variable 3

CPT for variable 4

(iii) Assume that the observed outcome was O1=0, O2=1, O3=3. Calculate which machine (M1 or M2 or M3) was most likely to have been picked from the box. Show numerical calculations (based on conditional probabilities) to justify your answer (do not use pgmpy to just write the final answer). **[4 points]**

Question 3 [10 points]

In this question below, we will construct a small Bayesian network in the pgmpy toolbox. This network models the relationship between *yellow fingers*, *smoking*, *cancer*, *skin burn*, *radiation*, *solar flares*, and using a *microwave*. In this model, smoking can cause yellow fingers and cancer. Solar flares and making microwave popcorn can cause radiation. Radiation can cause cancer and skin burn.

Consider "0" represents variable is false, "1" represents variable is true. Each variable in this problem is binary or can only take two values (0 or 1).

The prior probability of smoking P(S=1) is 0.1. The prior probability of solar flares P(F=1) is 0.001. The prior probability of using the microwave P(M=1) is 0.999.

Conditional probabilities for skin burn (B) are given below:

R	P(B=1|R)
0	0.01
1	0.1

Conditional probabilities for radiation (R) are given below:

F	M	P(R=1|F, M)
0	0	0.01
0	1	0.2
1	0	0.3
1	1	0.5

Conditional probabilities table for cancer (C) are given as:

S	R	P(C=1|S, R)
0	0	0.1
0	1	0.6
1	0	0.2
1	1	0.9

The conditional probability table for yellow fingers (Y) is given as:

S	P(Y=1|S)
0	0.1
1	0.9

Implement the above Bayes net with the specified conditional probabilities into pgmpy (Install it from http://pgmpy.org/).

- **Show screenshot of your code denoting the implementation in pgmpy and attach as part of your solution pdf.** [**2.5 points**]

Answer the following questions. You can use functions implemented in pgmpy to compute the numerical answers as required for some of the below questions. Show also snippets of your code used. [Each sub-part has **1.5 points**]

1. Draw the Bayesian network clearly showing the nodes and arrows showing relationship among all the variables (you can use drawing tools in PowerPoint or Keynote)
2. What is the probability of radiation given cancer = 1 (show values both for R=0, 1)?
3. What is the probability of cancer given SkinBurn=1 (show values both for C=0, 1)?
4. Are Smoking and Solar Flare independent? Justify your answer.
5. What is the probability of cancer (=1) if you never use a microwave?

Question 4 [10 points]

In this question, you will create an artificial neural-based classifier to classify the **k_mnist** dataset which is available at https://www.tensorflow.org/datasets/catalog/kmnist. More details about this dataset are here: https://github.com/rois-codh/kmnist.

You can build your solution on top of the python notebook covered in class to classify the standard (fashion_)MNIST dataset. The solution you provide should perform the following tasks:

a) Download and load the k_mnist dataset. **[1 point]**
b) Create an ANN with 1 input layer, **at least** one hidden layer with at least 2 nodes per layer. You can use relu or any other activation function for hidden layers. You are free to create additional hidden layers or increase the number of nodes to maximize the final accuracy (it would require some trial and error). Design your neural network so that you get an overall accuracy of 80% or more on the testing dataset after training. **[2 points]**
c) Create 1 output layer. What is the size of output layer? What should be the activation function for output layer? **[2 points]**
d) Compile and train the neural network with the appropriate loss function (what should be the loss function type?) **[2 points]**

For each of the above parts, also **show the relevant code snippets** in your solution pdf.

e) Plot the average training error (sum of the error over the training dataset divided by the total number of training examples) on the y-axis vs. the epoch number **[1 point]**
f) What is the final accuracy for different classes and overall accuracy on the testing data? Try to get an overall accuracy of 80% or more. **[2 points]**

For this question, please limit yourselves to using dense ANNs (rather than using advanced CNNs).

Question 5 [10 points]

In this question, we will learn how to use transfer learning in the context of CNNs. More hints to solve this question are at: https://www.tensorflow.org/tutorials/images/transfer_learning

We will use a pre-trained network (MobileNetV2) to do image classification on the CIFAR-10 dataset. More information on the dataset is available here: https://www.cs.toronto.edu/~kriz/cifar.html

We have provided you a code skeleton (skeletonCode.ipynb) to help you finish this problem. You are free to change this file as required.

Please do the following tasks in your solution:

a) Download the pre-trained MobileNetV2 network from Keras Applications (https://keras.io/applications/) **[1 point]**
b) Remove the final output layer of the network you have downloaded (to do this, please check the flag "include_top" in Keras and the tensorflow link for transfer learning noted earlier) **[1 point]**
c) After removing the final output layer, extend the MobileNetV2 model by adding at least one hidden layer (dense, convolution or any other type of layer). Also attach one final output layer. In this part, you are free to explore and decide how many hidden layers to add, their type, the number of nodes in each layer and the activation function yourself.

   Keep in mind, output layer must have number of nodes and activation function that matches the given task. **[2 points]**
d) Add the appropriate loss function, compile and train the modified network from part c) to classify the CIFAR-10 dataset. You can either fix the weights of the downloaded MobileNetV2 model and train only the layers you have added, or train the whole network again. Choose the setting that gives you higher accuracy given the computational resources. Check link https://keras.io/getting-started/faq/#how-can-i-freeze-keras-layers. **[2 points]**

Try to design your CNN so that you achieve 85% accuracy or more on the test dataset of CIFAR-10 after training.

For each part (a)-(d), **copy and paste the code snippets** in your solution pdf that shows how you're doing the particular tasks.

Show also the following results in your solution pdf:

1) Write how you extended the MobileNetV2 model (how many layers you added, what type of layers, how many nodes per layer, their activation function etc). **[2 points]**

2) Plot the loss function value with respect to the epoch number on the training data. How did you decide when to terminate training? How did you decide the mini-batch size for training? **[1 point]**
3) Show accuracy of the trained classifier over the entire testing dataset. **[1 point]**

Also attach the complete code file as part of your zip file. The code should be well commented, correctly compiles, and should produce the required outputs.

Please use tensorflow and Keras libraries for this question (rather than pytorch or other deep learning packages) as our testing setup has only these environments. You may have to install opencv package also for doing some image processing for this question (opencv should be already installed if you use Google Collab).

Programming hints: Make sure to save your network's parameters while training every once in a while (find out how you can do it in Keras). That way if your program crashes, you don't have to re-train from scratch. How many epochs are sufficient for convergence? You may need to enter some threshold condition so that training stops when that condition is satisfied. It is not good (or necessary) to train your network until the loss is zero (that may never happen). Furthermore, this may overfit the training dataset, and may lead to poor accuracy on the testing dataset.

Useful pointers for running on Colab:

1. Note that resizing the training and testing images will consume a relatively large amount of RAM. Please make sure the usage of memory is low before running your program otherwise your program may crash. What you can do is to restart runtime (*Runtime->Restart runtime*) before running your program.
2. To use GPU for training, you will need to enable GPUs for the notebook (*Runtime->Change runtime type->Hardware accelerator->GPU*). Please change the runtime type to CPU when you are not using GPU resources as GPUs will be prioritized for users who have recently used less resources.
3. If the Colab still runs out of memory, you can decrease the size of the training dataset. However, it might reduce your accuracy on the test dataset also.