**CPSC 4610 Artificial Intelligence Homework #4**

1. **Ethics (25 points)**

Suggest two feasible measures to remediate *bias* in Machine Learning

1. Explain why each measure would be effective
2. Identify potential costs for each approach.

* **Diverse and Representative Training Data**
  + The key point here is that ML models learn from the data they are trained on. If that data is unbalanced/skewed the model may develop biases that reflect those imbalances. Ensuring we have diverse and representative data, the model can learn a more equitable distribution of features, making it less likely to favor one “group” over the other
  + **Example:**
    - **Facial Recognition System**: If the model is trained on primarily light-skinned faces, it may struggle to recognize darker-skinned individuals (Similar to the Tesla case in 2019). A more diverse dataset would eliminate that bias
  + **Costs:**
    - Since we want a more diverse data set, we need to do some more data collecting. This can be expensive in some cases, but it all depends on the application. Now that we have more data, this also requires a lot more processing, serializing, labeling, …
    - The costs here are primarily “the work” rather than financial. Although some people say “time is money” so…
* **Bias Detection & Correction Algorithms**
  + The key point here is to detect and mitigate bias in some stage of our algorithm/model. The most common techniques are pre-processing, where we resample the biased data. In-processing, where we add fairness constraints during model training, and post-processing, where we adjust model outputs to make them fair.
  + **Example:**
    - **AI Hiring System**: Going back to the bank example we covered in class. We could’ve changed the input data to where the ratio between black people and white people would’ve been equal. We could’ve adjusted certain parameters during the training to obtain non-biased results, or we could’ve added more black people to the final pool of candidates.
  + **Costs:**
    - We’re now a Buridan’s donkey. We have to pick between having a model that is somewhat efficient but biased, or a model that is slow but unbiased. By adding correction algorithms, we slow down the model pretty significantly and we risk accuracy. On top of that, if our fairness constraints are not clear or well designed – we just created a model that is even more biased.

1. **Temporal Difference and QLearning (25 points)**

When asked the question “What is the difference between Temporal Difference Learning and QLearning?”, Chatgpt provides a fairly good summary, citing Wikipedia, of each approach. Key differences noted though focus on the update process and policy type. To fill out Chatgpt’s definitional response, for each of the following, provide an explanation that a novice would understand.

1. The differences in intent and usage for QLearning and TD learning.
2. Expected drawbacks of each approach

* **TD Learning**
  + **Intent:**
    - TD Learning is designed for learning value functions by updating estimates based on observed rewards and future estimated values. It does not require a complete model of the environment.
  + **Example:**
    - A real life example might be easier for one to comprehend. Think of a dog learning how to sit for a treat. The dog is trained to sit when given the command “sit”. Each time it sits, it gets a small treat (reward) immediately. If it doesn’t sit, it doesn’t get a reward. Over time, the dog associates sitting with a positive outcome and adjusts it’s behavior.
    - The learning process updates step-by-step as the dog experiences small rewards for correct behavior. The dog does not need to wait until the end of training to understand rewards, each interaction updates its value estimate of sitting.
    - *This is in some way similar to Pavlov’s experiment, both of the dogs learned through experience and reinforcement, and adjusted behavior based on feedback ☺*
  + **Usage:**
    - It is used when an agent needs to estimate the value of states over time
    - It’s also often applied in prediction problems, where future rewards are uncertain.
* **Q-Learning**
  + **Intent:**
    - Q-Learning is a specific type of TD learning focused on learning an optimal policy. Instead of estimating state values, it estimates the optimal action to take in each state.
  + **Example:**
    - Here the dog example isn’t enough, because the optimal action is to obviously sit as he wants a treat. Let’s look at a Roomba vacuum instead. A Roomba needs to decide which direction to move in a house to get the whole house vacuumed. It’s available actions are **L, R, F, B**, and each action results in a score based on how much dirt it picks up. If it reaches a dead end it gets a score of -10, if it finds a dusty area it gets a score of 10 points. Over time it learns the best sequence of moves to clean efficiently.
    - The vacuum stores values for each action in each state (Q-vals). It then learns the best path by continuously updating which moves give the highest future rewards. Unlike TD Learning, which estimates state values, Q-learning focuses on actions and decision-making.
  + **Usage:**
    - It most commonly used for control tasks, where an agent needs to learn how to act optimally, and in decision making scenarios, like games or robotics
* **Drawbacks:**
  + **TD Learning:**
    - According to [this article](https://botpenguin.com/glossary/temporal-difference-learning), TD Learning is known to be biased in early estimates, since TD updates are based on previous estimates rather than final outcomes, errors can propagate if the model starts with poor initial values. In complex environments, TD learning can be “unstable”, especially when combined function approximation (NN)
  + **Q-Learning:**
    - According to [this article](https://www.techtarget.com/searchenterpriseai/definition/Q-learning#:~:text=What%20are%20the%20disadvantages%20of,ways%20to%20approach%20a%20problem.), Q -learning requires sufficient exploration of all possible actions; otherwise, it may settle on a suboptimal policy (Explore vs Exploit tradeoff). Q-Learning is also very slow in large search spaces since it maintains a table of Q-vals for all state-action pairs, it becomes inefficient in high-dimensional environments.

1. **Bayesian Games (25 points)**

The StagHunt and the Pig as Game Theorists were two examples explored in L18 (3/10/25)

1. Identify and explain the key differences between the problems
2. Briefly explain how each problem is OR is not similar to a POMDP

* **Stag Hunt**
  + Stag Hunt is a coordination games where two players can hunt stag, cooperate, and earn a large reward. If one defects and chooses rabbit while the other chooses stag, the stag hunter gets nothing while the rabbit hunter gets a small meal. The key challenge is trust. Both players must commit to cooperation for the best outcome.
* **Pig as Game Theorists**
  + PGT is a game where players decide when to stop and claim a reward, knowing that if they wait longer, they can get a bigger reward but also introduce more risk. The key challenge in this game is strategic stopping. Both players must balance risk vs. reward rather than simple cooperation as seen in Stag Hunt. An example is two people deciding when to stop a dice game. Each turn, they can roll again to accumulate more points or stop, and keep their points. If they roll a 6 for example, they lose all points for that round. The challenge is deciding when to stop.
* **Relation to POMDPs**
  + **Stag Hunt**
    - Stag Hunt is similar to a POMDP as both of the players do not have full visibility into the other player’s decision-making process. They need to infer whether cooperation is likely to happen or not.
  + **PGT**
    - PGT is not similar to a POMDP. The decision is primarily about risk timing rather than state uncertainty. There is no hidden state (fully observable), just uncertainty about future rewards

1. **Bayesian Sampling (25 points)**

Bayesian networks are important for predictive modeling yet often are missing data and usually are too large for exact inference. L19 provides a quick review of sampling techniques used. Consider the techniques of Prior Sampling and Likelihood Weighting

1. Identify and explain the key differences between the two techniques
2. Give a different, short example for each technique that displays its effective use

* **Prior Sampling:**
  + **Concept:**
    - Prior sampling generates samples directly from the Bayesian network’s probability distribution, without considering any observed evidence. It follows the network structure and assigns values to variables in topological order (parent to children). It is useful for exploring general behavior but not efficient if we need to condition on some specific evidence.
  + **Example:**
    - Let’s say we have a BN predicting the weather in Ljubljana. The network includes the variables like: Season S, which affects temperature and precipitation. Temperature T, which depends on the season. Precipitation P, which depends on the season and temperature.
    - **Prior Sampling:**
      * First, randomly sample Season (S) (60% chance of summer, 40% chance of winter)
      * Sample Temperature (T) based on the season (summer ⇒ high temperature, winter ⇒ low temperature)
      * Sample Precipitation (P) based on the temperature and season
    - It’s important to note, that if we need to know the probability of rain given that the temperature is 35°C, prior sampling is pretty inefficient
    - Most samples will not have 35°C, so we’re wasting computation on irrelevant cases.
* **Likelihood Weighting:**
  + **Concept:**
    - Likelihood weighting focuses sampling on relevant cases where we already have some evidence. Instead of sampling all variables randomly, it fixes the observed evidence and adjusts the probability weights of samples accordingly. This is very useful when we need to compute probabilities given specific observations.
  + **Example:**
    - In this case, let’s say we have a BN for diagnosing whether a patient has a version of COVID or not. The network includes variables like Disease D (some strand of COVID), which affects symptoms like loss of smell and loss of taste. Loss of smell S, which depends on disease D. Loss of taste T, which depends on disease D.
    - To keep things simple, let’s just assume that the two symptoms S & T are the only symptoms associated with COVID-19
    - **Likelihood Weighting:**
      * Suppose the patient already lost his sense of taste (T = True)
      * Instead of randomly sampling all cases (including T or ¬T), we only generate cases where T = True. We then assign weights based on how likely it is a given disease would produce the loss of taste.
      * The samples where loss of taste is more likely will be weighted higher, while other strands of COVID where loss of taste is rare will have low weight
    - Likelihood Weighting is farm more computationally efficient than prior sampling because it focuses on relevant samples.
    - On the other hand, prior sampling outperforms likelihood weighting if the evidence is very unlikely in the network. In that case, all weights become very small, reducing accuracy.