**CIS 9**

**Reinforcement Learning and Beyond**

**Reinforcement learning**

Supervised learning is learning by examples. Unsupervised learning is learning by observations.

Reinforcement learning is learning by feedback:

state

reward / penalty

Agent

(algorithm)

Environment

action

* The agent starts at state S0 in the environment.
* The agent takes action A0.
* The action results in the agent changing to state S1, and getting a reward R1 from the environment.
* Based on the resulting state and reward, the agent continues with action A1, which results in the new state S2 and reward R2.
* The cycle continues until the agent either solves the problem or gets stuck.
* During the training stage, the agent then tries solving the problem again with the added knowledge from the previous times.

There are several methods that determines the next action that the agent takes:

* Some rewards have a higher score than others.
* The agent also needs to figure in any penalties that are present.
* At each stage, the agent can figure out the total reward for each path, and choose the action which goes down the path with the most reward value. This is the *value based* approach.
* At each stage, the agent can also choose the next action based on the probability that it's the best action. The probability of the best action, or the *policy*, can come from the set of rules that are defined for the problem, or it can be deduced from previous tries during the training stage. This is the *policy based* approach.

Steps to train a reinforcement learning algorithm:

1. Create an environment that the agent will work with.

Example: a tic-tac-toe game, where the algorithm will play against an opponent (human or machine)

 (image [source](https://en.wikipedia.org/wiki/Tic-tac-toe))

1. Specify the rewards and penalties.

Example of rewards: adding an X (or O) is +1, adding it in the same column or row is +2, adding it in the center is +3

1. Create an algorithm and policies for problem solving.

Example: build the algorithm from an existing model or from scratch

Examples of policy: get 3 X's (or O's) in a row, block opponent if the opponent has 2 X's (or O's) in a row.

1. Train and test the agent.

Example: let the agent play the game repeatedly until the agent always ends with a win or a draw.

Steps 2 - 4 are part of an iterative process. It takes some experimenting to set the rewards and penalties and tune the algorithm in order to have an optimized algorithm, just like with supervised and unsupervised learning.

**Neural network**

The algorithms we've studied are considered *classic machine learning* algorithms. They are relatively (emphasis on 'relatively') simple and easy to implement, and they provide a good foundation for understanding the concepts and challenges of machine learning. When these algorithms are tuned and used correctly, they are good at predicting, classifying, or detecting patterns in data. However, they are limited to a specific kind of problem that they can solve. For example, if we have a good classification algorithm to help us identify the types of banking transactions for a business, we cannot use the same algorithm to detect frauds in the banking transactions. We would need to apply a different algorithm for the detection.

Because of this limitation of classic machine learning algorithms, the majority of large tasks that use ML today need to be solved with *neural networks*. For researchers and industries of all kinds that heavily use ML (self-driving cars, facial recognition, deep fake, etc), the only ML model to use is neural networks.

Does this mean neural networks are the answer to all ML applications? No, not really. If we have a small budget, small datasets, or want to get insights on how the algorithm works, then the classic machine learning models are the answer. Neural networks are for large and complex tasks because they can solve multi-faceted problems, but they are difficult and expensive to build, and they need extremely large datasets in order to be accurate.

Neural networks are black boxes, researchers who build them to run algorithms actually don't know what goes on inside the hidden layers of the neural networks. A neural network is made up of 3 main layers:

Shallow network

Input layer

Hidden layer

Output layer

Deep network

Output layer

Hidden layer 2

Hidden layer 3

Input layer

Hidden layer 1

If a neural network has multiple hidden layers, then it's considered a deep network, the building block of *deep learning*.

Neural networks have been a major area of research in both neuroscience and computer science, which explains the terminologies used in neural network. A neural network is made of: neurons, synapses, weights, biases, and functions.

* A *neuron* is a basic unit that can accept one or more input; it runs an activation *function* on the input, and passes the result forward to another neuron.
* The neurons are connected together by *synapses*. Each synapse has a *weight* assigned to it. A large weight means the data coming from that synapse will be considered more important to the neuron receiving the data. During training and tuning, the designer adjusts the weights for the synapses.
* Each layer can contain *bias* neurons that adjust the weight of the synapses within the layer. This allows the designer to tune the neural network further.

For an overview visualization of different neural networks, check out this [page](https://www.asimovinstitute.org/neural-network-zoo/) where you can see diagrams of neural networks, showing the neurons, synapses, and layers. And if you're more curious, the author described each network and provided links to the research paper behind each neural network. To dive deeper into how a simple neural network is implemented in Python, see this [page](https://towardsdatascience.com/an-introduction-to-neural-networks-with-implementation-from-scratch-using-python-da4b6a45c05b).

The biggest difference between classic machine learning and neural network is in how the learning takes place. With classic machine learning, the model needs input from the human designers, whereas with neural networks, the model learns by taking in many, many input and using the hidden layers to extract information from this input, similar to how the human brain learns.

Neural networks have been a field of research since the 1970's but it has come in and out of favor and never quite gained traction until the recent years. Neural network's popularity has soared in recent time due to:

* the advancement of hardware: massively parallel processors with powerful computing performance.
* sophisticated software framework (Tensorflow, Keras, Pytorch, etc.): libraries with many machine learning models and algorithms and a platform that makes it easier to process input data, and train and tune the models.

Whether we use classical machine learning or neural net for deep learning, the steps are the same:

* gather and clean the data
* set up X and y training ant testing set
* select an ML model
* train and test the model
* evaluate the model's accuracy

[Here](https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/) are the steps using Keras and Tensorflow for deep learning. You can see that it follows the same thought process as the ML work that you've done.

Finally, we end our module on [machine learning](https://www.facebook.com/mldcmu/photos/a.2311822062430292/2042421699370331) with some humorous quotes, courtesy of Carnegie-Mellon University, one the many powerhouses of machine learning / AI research.