CSC 180 – Final Study Guide

Lecture #1: Introduction:

- **What is AI:**
  - Artificial Intelligence (AI) is the study on how to program computers to solve hard problems that traditionally require human intelligence to solve.
- **Agents & Smart Agents:**
  - **Agent:** A piece of software that observes input through sensors and acts upon an environment using actuators.
  - **Smart Agent:** Given input from a sensor, take an action that will maximize its performance measure.
- **PEAS Model** (Ex's in Lecture #1 Slides):
  - **P**erformance Measure: An objective function the agent is maximizing.
  - **E**nvironment States: a formal representation for states (a group of variables).
  - **A**ctuators: Actions that change the agent state according to a transition model.
  - **S**ensors: Observations that allow the agent to infer the world state.

Lecture #2: Data Preprocessing & Model Evaluation:

- **Attribute Types:**
  - **Categorical**: Can be nominal (no order) or ordinal (ordered but not comparable). Examples include eye color, rankings, height (tall, medium, short), etc.
  - **Numerical**: Can be discrete or continues. Examples included temperature, dates, time, length, etc.
- **Categorical Feature Normalization:**
  - **Label Encoding:**
    - Assigns one unique number to each distinct value (*calabar -> 0, uyo -> 1, etc).* It is a one-to-one mapping from a string to an integer. Some Scikit functions require this as input, though it is never to be used on input features.
  - **OHE (One-Hot Encoding):**
    - For each distinct value, create a new binary feature. (Red -> 001, Green -> 010, Blue -> 100). Always used to encode categorical features (input & output).
- **Numerical Feature Normalization** (Used to make the values comparable):
  - Divide raw values by the max value. Everything within [0, 1] range.
    - *New = (raw value) / (max column value)*
  - Subtract the min value and divide by the difference of the max and min value. Everything within [0, 1] range.
    - *New = [(raw value) – (min column value)] / [(max column value) – (min column value)]*
  - MOST POPULAR - Use the standard score (z-score) of a raw value X. This is negative when the raw value is below the mean, and positive when above.
    - *Z-score = (x – mean) / (stdDev)*
- **Regression Vs Classification:**
  - Regression is used for predicting a continuous output.
    - Supervised: Regression (Linear Regression, Neural Networks, etc).
    - Unsupervised: Dimensionality Reduction
  - Classification is used to predict a discrete output. Often a label.
    - Supervised: Classification (KNN, SVM, Decision Tree, Bayes Classifier, Logistic Regression, Neural Networks, etc).
    - Unsupervised: Clustering (K-Means, Hierarchical Clustering, DBSCAN, GMM, etc).
- **Underfitting:**
  - Occurs when a model is too simple and has large errors on the training and test set.
- **Overfitting:**
  - Occurs when a model fails to generalize. This is seen by high scores on the training set, but low scores on the test set.
- **Training Goal:**
  - Minimize test-set error by using checkpoints, and monitors. Also use a train-test split.

- **Classification Metrics:**
  - **Precision:** Out of labeled as positive how many are positive?
    - *Prediction = (#true positive predictions)/(#true positive + #false positive)*
  - **Recall**: How many positive records are labelled corrected.
    - *Recall = (#true positive predictions)/(#truee positive + #false negative)*
  - **F1-Score**: Combines both precision and recall in a single summary score to determine the model quality.
    - *F1 = (2 * recall * precision) / (recall + precision)*
  - **ROC Curve**: Plots the true positive rate against the false positive rate. The area under the curve (AOC) is the best model. A larger area = better model. A perfect model has an area of 1, with a random guess model being 0.5.
- **Regression Metrics:**
  - **RMSE (Root-Mean-Squared-Error):** Essentially the square of the mean-squared-error (MSE). The lower the value, the better the model. It is in the same units as the training data outcome.
  - **R2 Score:** The proportion of variance in the dependent variable that is predictable from the independent variable.

Lecture #3: Neural Networks

- **What are Neural Networks:**
  - Neural networks are a collection of neurons. Each neuron is a weighted sum followed by an activation function.
    - Pros:
      - Flexible and general functional approximation framework.
      - Can build extremely powerful models by adding more layers.
    - Cons:
      - Training is prone to local optimum.
      - Huge amount of training data and computing power required.
      - Implementation choices are huge (hyper-parameter options).
- **Loss Function to Minimize:**
  - The goal of training a NN is to minimize the error between true and predicted labels by finding the best weights & biases for the NN.
- **Gradient Descent (back propagation):**
  - Weights are updated according to the rules of gradient descent and computed from output to input layers.
    - **Batch Size:**
      - The number of samples per update (default 32).
    - **Epoch:**
      - At each epoch, the entire training set was passed over once.
    - **Step:**
      - At each step (iteration). One batch was processed.
    - **How often weights are updated:**
      - Weights are updated according to the batch size (step). This can be specified and is by default 32. If done by epoch, the weights are updated each pass through the entire training set.
    - **How should the weights be updated:**
      - Weights are updated according to the rules defined by the optimizer. The most common being *Adam* & *SGD* using back propagation.
    - **Learning Rate:**
      - A high learning rate runs the risk of passing over the optimum while a low learning rate takes forever to find it. The best learning rate is somewhere in-between.

Lecture #4: Deep Learning

- **Deep Learning Vs "Regular" Neural Networks:**
  - "Regular" neural networks have a small number of hidden layers and are only used for supervised learning. In addition, they only include fully connected layers (every neuron in adjacent layers is connected). "Deep" learning networks have a larger number of hidden layers and can be used for either supervised or unsupervised learning. They can include layers such as convolution, pooling, and dense (fully connected).

- **Why Deep Learning is Generally Better:**
    - Deep learning is generally better because its greater (and more diverse) hidden layers can detect true underlying features and train to detect those features. Stacked hidden layers capture features from a lower to a higher level.
- **CNN Vs RNN:**
    - CNN's are good for image classification while RNN's (Recurrent Neural Networks) work well with temporal or sequential structures. CNN's are good where spatially closer data is more correlated.
- **Three Stages of CNN:**
    - **Convolution** (Single or Multiple Kernels):
        - Produces a weighted sum (compressed) image.
        - Uses a kernel of a defined size. This scans the image and produces a feature map.
        - Number of features maps are equal to the number of kernels.
        - Outputs a feature map, based on the weighted sum of the scanned image.
    - **Nonlinear Transformation:**
        - Same as an activation function in Neural Networks.
    - **Pooling:**
        - Max-Pooling reports the max output within a rectangular region.
        - Average-Pooling reports the average output with a rectangular region.
        - Pool size, the vertical and horizontal size of the rectangular region.
        - Stride value, the vertical then horizontal space the rectangular region travels.
        - Padding, valid (no zeros) or same (pad with zeros).

Lecture #5: Transfer Learning

- **Transfer Learning:**
    - The process of adapting an existing model, trained for a similar task, to perform our desired task. It essentially acts as a shortcut, saving time and boosting performance.
- **Steps to Implement Transfer Learning:**
    - Step 1: Load the pre-trained model, excluding the top layer.
    - Step 2: Freeze the weights of the pre-trained model.
    - Step 3: Add other layers on top of the pre-trained model.
- **Where to Use Transfer Learning:**
    - Use if you do not have much data on your own task, but you can identify a related task with an existing pre-trained model.

Lecture #6, 7: Informed Search

- **What is Search:**
    - Search involves starting from an initial state and finding a path to the goal state. Our goal is to try to expand as few states as possible while finding a least-cost path.
        - **Frontier:** A list of unexpanded states
        - **Search Strategy:** The strategy we use to determine which node on the frontier to expand next.
- **Search Criteria:**
    - **Completeness:** Does the algorithm always find a solution if one exists?
    - **Optimality:** Does it always find a least-cost solution?
- **Informed Search:**
    - Search using a heuristic function to rank nodes on the frontier. Expanding only the nodes with the lowest heuristic value.
- **Admissibility:**
    - A search strategy is admissible if and only if the heuristic value is always <= to the true cost of a node.
- **Search Algorithms:**
    - **Dijkstra:** Orders all nodes on their G(N) values (distance travelled).
        - Optimal!
    - **Greedy Search:** Orders all nodes on their H(N) values (distance remaining).
        - Fast, but not complete (loops). Also, not optimal.
    - **A*:** Balances the above using the F(N) = G(N) + H(N) to order all nodes (distance travelled + distance remaining).

        - If the heuristic value is admissible, A* is optimal. Also, complete.

Lecture #8: Uninformed Search

- **Uninformed Search:**
    - Only use the information available in the problem definition. As a result, we cannot produce a heuristic function and simply expand nodes according to some order. This is generally slower than informed search.
- **Search Algorithms:**
    - **Breadth-First Search (BFS):** Expand the shallowest unexpanded node. Visitor all your neighbors before your neighbor's neighbors. Implemented via a FIFO queue. If at same depth expand left to right.
        - BFS is complete. Also, it is optimal only if the costs are all equal.
    - **Depth-First Search (DFS):** Expand the deepest unexpanded node. Visit nodes adjacent to the last visited node first. Implemented via a stack.
        - DFS is not complete as it fails in infinite-depth trees, or states with loops. It is also not optimal as it returns the first solution it finds.

Lecture #9: Adversarial Search (Minimax)

- **Minimax Search:**
    - An adversarial search algorithm that computes each nodes minimax value. The highest reward value Max can achieve if Max plays against an optimal adversary.
- **Alpha-Beta Pruning:**
    - Used to prune nodes whose values are worse than the current Alpha-Beta value for Max & Min respectively.
- **AlphaGo:**
    - Treats the gameboard of Go as an image. It is composed of two CNN networks.
        - **Policy Network**: Trained to predict human moves.
        - **Value Network:** Trained to predict the reward for each state.

Lecture #10: Genetic Algorithms

- **What are Genetic Algorithms (GA):**
    - GA is search based optimization based on Darwin's evolutionary biology. It selects good individuals and extends them by mating. After several generations you have better items.
- **GA Advantages:**
    - Does not require any derivative information (failure or gradient based methods).
    - Optimizes both continuous and discrete functions and multi-objective problems.
    - Provides near-optimal solutions in short time.
    - Provides a list of "good" solutions and not just a single solution.
    - Useful when the search space is large, with many parameters.
- **GA Limitations:**
    - Not suited for all problems where derivative information is available.
    - Fitness value is calculated repeatedly, which can be computationally expensive.
    - No guarantees for optimality or solution quality.
- **Basic GA Structure:**
    - **Structure:** Population Init -> Fitness Assignment -> Loop(Selection, Crossover, Mutation, Survivor Selection) -> Terminate & Return
    - **Static State:** Generate a few offspring (1 or 2) in each iteration. These will replace an equal number of individuals in the current population.
    - **Generational GA:** Generate N offspring where N is equal to the population size. The entire population is replaced with their children at the end of each iteration.
- **Parent Selection:**
    - **Fitness Proportionate Selection:** Every individual can become a parent with a probability proportionate to its fitness.
        - **Roulette Wheel Selection:** Divide a wheel into N pies, each individual's wheel portion is proportionate to its fitness. With 1 fixed point, spin the wheel. The location where the point lands is the selected parent.

        - **Stochastic Universal Sampling:** Using 2 or more fixed points, spin the wheel. Each point selects a parent for a child.
    - **K-Way Tournament Selection:** Randomly select K individuals. Select a champion among them (best fitness). This becomes the parent.
    - **Rank Selection:** Every individual ranked according to their fitness. A higher rank equals a higher selection chance.
- **Mating:**
    - **Crossover 1 or 2 point:** Randomly crossover the selected segments. Swapped between parent to get new offspring.
    - **Uniform Crossover:** Each gene is considered separately. Randomly swap genes or not.
    - **Arithmetic Recombination Crossover:** Take the weighted average of the 2 parents. Produces identical children. However, a different weight can be used to produce unique children.
- **Mutation:**
    - **Bit Flip:** Randomly select 1 or more positions and flip the value.
    - **Swap:** Randomly select 2 positions and interchange the values.
    - **Scramble:** Choose a subset of genes and randomly shuffle their values.
- **Survivor Selection** (Static State GA Only):
    - **Age Based:** Always kick out the oldest individuals.
    - **Fitness Based:** Always kick out the individuals with the lowest fitness.
- **Avoiding Invalid Individuals:**
    - Invalid individuals can easily be avoided by assigning a high penalty to individuals that are invalid. This should automatically exclude them from the next generation.

Lecture #11: Bayesian Inference

- **Inference:**
    - Given the value of some evidence variable, find the value of the query variable that maximizes the posterior probability. Essentially an educated guess…
- **Bayesian Inference:**
    - An inference tool that uses conditional independence to simplify the calculation of joint probabilities.
- **Chain Rule:**
    - Derives the joint probability from the conditional probability.
        - *P(X1, X2, …, Xn) = P(X1)P(X2|X1)P(X3|X1, X2)…*
- **Independence:**
    - Two events (A, B) are independent if and only if:
        - *P(A, B) = P(A) * P(B)* or *P(A|B) = P(A)* or *P(B|A) = P(B)*
- **Conditional Independence:**
    - Two events (A, B) are conditionally independent given C, if and only if:
        - *P(A, B|C) = P(A|C) * P(B|C)* or *P(A|B, C) = P(A|C)* or *P(B|A, C) = P(B|C)*

Lecture #12: Fuzzy Logic & Expert Systems:

- **Fuzzy Set Operations:**
    - **Complement (NOT):** $\bar{B}_1 = 1 - \mu_b(x)$
    - **Union (OR):** $B_1 \cup B_2 = \max[\mu_a(x), \mu_b(x)]$
    - **Intersection (AND):** $B_1 \cap B_2 = \min[\mu_a(x), \mu_b(x)]$
    - **Difference:** $B_1 \mid B_2 = B_1 \cap \bar{B}_2$
    - **Sum:** $B_1 + B_2 = [\mu_a(x) + \mu_b(x)] - [\mu_a(x) \cdot \mu_b(x)]$
- **Fuzzy Numbers:**
    - Fuzzy sets defined on real numbers.