CSC 417 Midterm Study Guide

1. Quiz 1
   1. A drawback of rote learning is that AI learns a good sequence of moves for a game, but does not understand how to win
   2. Generalized learning techniques do not result in an AI that plays strong endgames.
   3. Humans and AI tend to not use similar representations of board game states (e.g. visual representation).
   4. Human beings seem to be built to employ heuristics rather than algorithms
   5. In conjunction with Checkers, Arthur Samuel observed that developing a heuristic for a task can be harder than performing the task.
   6. When employing a Finite State machine, additional states are not to be created as needed during execution
   7. All Finite State Machines do not need to contain an ending state.
   8. Behavior Trees tend to be more transparent than Finite State Machines (that is, it is easier for a human to understand exactly what the AI is doing).
   9. A perceptron learns by adjusting the weights associated with specific inputs.
   10. A Convolutional Neural Network analyzes a group of pixels in a source image by means of one or more filter(s).
   11. Heuristics are most appropriate for analyzing the midgame of a board game (such as checkers or chess).
   12. A chess AI which employs a Shannon Type B approach will extend the searching depth for interesting moves.
   13. In a behavior tree, a leaf node specifies the actual action to be taken.
   14. The following is true of backpropagation in neural networks—they typically requires large training data sets, they distribute error across weights in the output layer and all hidden layers and they require a continuous activation function.
   15. It is not true of backpropagation in neural network that it allows networks to learn training data in a single epoch.
   16. One strength of Convolutional Neural Networks compared to “traditional” ANNs is that they can identify subject regardless of its position in an image.
2. Quiz 2
   1. In a Convolutional Neural Network, a tensor is not a one-dimensional array.
   2. The output from a convolutional layer is smaller than the input.
   3. The purpose of a pooling layer in a CNN is to reduce spatial size of features (and thus require the network to process less data).
   4. In order to achieve high accuracy, a Recurrent Neural Network must include more than one recurrent layer.
   5. In an RNN, the vanishing gradient problem refers to the fact that later inputs have a substantially larger impact on the output than early inputs.
   6. In an RNN, one purpose of the hidden state is not to preserve sequence relationship information.
   7. The goal of a Generative Adversarial Network is not to create an exact duplicate of some set of input data.
   8. Because the input is typically random noise, a GAN typically requires many epochs to create “good” output.
   9. Generative algorithms are not used to classify input data.
   10. A “standard” CNN would not be an appropriate choice for the generative component of a GAN.
   11. In a Convolutional Neural Network, max pooling is used to perform both de-noising and dimensionality reduction on the input data.
   12. Padding is used to address this shortcoming of CNNs—that information “on the edges” of the input is not captured well.
   13. In an RNN, processing all inputs in sequence is referred to as unfolding.
   14. In a GAN, the following is a job of the generative network—modle the distribution of data in a category, determine the probability of a feature existing in a given category, and produce a synthetic data item which fits a given category.
   15. In a GAN, to learn the decision boundary separating categories is not a job of the generative network.
   16. Within a GAN, no training data set is required, the training data set is passed into the discriminative network, and the training data is unrelated to the random noise input to the generative network.
3. Advanced Computer Games
   1. Checkers (Arthur Samuel)
      1. Why checkers?
         1. Non-deterministic – different outputs possible for identical inputs
         2. Clear goal – remove opponent’s pieces
         3. Clear rules – easily described
         4. Established knowledge – many human experts
         5. Understandable to even “causal” observers
      2. Minimax – good at representing a game like checkers
         1. System defaulted to looking 3 moves ahead
         2. Exceptions that increased search depth:
            1. Considering a jump
            2. Last move was a jump
            3. Piece exchange possible
      3. Notation – light colored squares are numbered (this allows for easy input into computers)
      4. Heuristics
         1. Piece advantage
         2. Denial of occupancy (board control)
         3. Mobility
         4. Hybrid measure of control of board center plus piece advancement
      5. Moves that were not referenced for a long period were “forgotten” – storage space was expensive
      6. Rote Learning
         1. Lacked sense of direction (*how* do we achieve the goal)
         2. Program could memorize a good sequence of moves but did not “understand” how to win
         3. Resulted in strong openings and endgames, but a poor midgame
         4. Good for situations requiring specific actions
      7. Generalized Learning
         1. Alpha program – updated evaluation of moves during a game
         2. Beta program – updated evaluation of moves after a game
         3. Programs played each other repeatedly to develop a collection of good moves
         4. Resulting program was above average, but had limitations:
            1. Easily fooled by deliberate bad play (opponent could make a non-optimal move to “throw off” the system)
            2. Evaluation function changed too quickly
            3. Overvalued “flashy plays” – favored large score swings over basic moves that set up later wins
         5. Strong midgame, but poor openings and endgames
         6. Useful when there are many possible actions – aka the midgames of chess and checkers
   2. Non-search-based techniques
      1. Turing Machine
         1. Components
            1. Infinitely long tape with “boxes” (memory locations)
            2. Each box may contain a symbol
            3. Each box may be read, written, or erased by the read/write head
            4. Instructions on specifying how to modify squares
            5. Listing of states (which define actions) and transitions
      2. Finite State Machines
         1. Consist of one or more states (finite number)
         2. One state active at a time
         3. FSM transitions between states
         4. States must be known in advance