Week 8 Lecture Notes

Points of clarification and fun facts re: video lectures

Much of this lecture overlaps with parts of Andrew Ng's machine learning class. Here is a link to his lecture notes: http://www.holehouse.org/mlclass/ . Neural networks are covered in lectures 08 and 09.

L1: Neurons as Classifiers and Supervised Learning

- Slide: The Classification Problem
  - In order to perform classification, you first need a representation of your object/face/ etc. as a vector in a vector space. The elements of these vectors correspond to values of features of the things you're trying to classify. For example, one feature of an image could be the average brightness, whereas another feature could be the ratio of reds to greens, etc. Properly choosing a low-dimensional feature space is an important step in building a successful classifier.
  - A hyperplane in an N-dimensional vector space is an N-1-dimensional "sheet" described by $a_1x_1 + a_2x_2 + \ldots + a_Nx_N = c$, that is, the equation for the hyperplane is some linear function of the elements in the vector space set equal to a constant. In 2-D, one example would be $3x_1 + 2x_2 = 5$, which is just a line.
- Slide: The "Perceptron"
  - Theta(x), often called the step-function or Heaviside function, is normally equal to 0 if x is less than or equal to 0. Here, however, we set it to -1 for convenience.
- Slide: Perceptron Learning Rule
  - In short: for positive inputs, increasing the weight moves the output upwards and decreasing the weight moves the output downwards. Ideally, you want to move upwards/downwards until the perceptron output are on the correct side of the threshold. Further, moving the threshold downwards makes more inputs exceed the threshold, and moving the threshold upwards makes fewer inputs exceed the threshold.
- Slide: Multilayer Perceptrons
  - This is somewhat analogous to linking together logic gates to build an XOR function
- Slide: What if you want continuous outputs?
  - Here we are referring to continuity in the output space, and not necessarily in time.
- Slide: Continuous outputs with Sigmoid Networks
  - The sigmoid function generalizes the step function; the limit of the sigmoid function as the slope goes to infinity is the step function
- Slide: Learning Multilayer Sigmoid Networks
  - These networks are feed-forward, meaning that there are no recurrent/feedback connections. That is, signals propagate in only one direction. This causes the networks to be memoryless. However, when you invoke a learning rule, this allows

the network properties to change, and these changes will depend on the previous state of the network.

- Keep in mind, though, that the delta and back propagation learning rules are entirely supervised. This means that the connection weights are changed in order to move the network output toward the desired output, i.e., the amount that each weight needs to change depends on the global state of the network. In biologically realistic neural networks, however, an individual synapse does not necessarily have access to the global state of the network. Hence, there is much research investigating how unsupervised learning rules could give rise to the network architectures we see in real systems.

L2: Reinforcement learning: predicting rewards

- Slide: Predicting Delayed Rewards
  - This is actually quite similar to the previous lecture, in which we tried to train a network to imitate some input-output function. In this slide, we're trying to get a neuron (or network) to mimic the function whose input is a stimulus and whose output is a predicted reward. In this learning problem, however, training occurs one trial at a time.
  - In the equation, we are summing over the rewards from time *t* to time *T*, where *T-t* is the amount of time left in the trial. That is, v(t) is the expected total reward the animal will receive from t until the end of the trial.
- Slide: Learning to Predict Future Rewards
  - A tapped delay line maps a temporal signal (a signal varying in time) to a spatial signal (a pattern of simultaneous neural activations). For example, a signal over three time steps that started off at 10, then decreased to 5, and then to 0, might get mapped to one neuron of three being very active, one neuron being moderately active, and the last neuron being inactive.
  - Remember that the animal only has access to r(t), not to r(t+1), r(t+2), etc., as these occur in the future. Thus, we need an approximation in our equation.
-
- Slide: Temporal difference learning
  - Remember, the parameters of the function v(t) are the weights w(tau).
  -
- Slide: Predicting future rewards: TD learning
  - Right-hand plots (columns represent two different trials, i.e., points in the learning process):
    - 1. Stimulus
    - 2. Reward
    - 3. Prediction
    - 4. Derivative of prediction (v(t+1) - v(t)) -> remember that v(t) is the sum/integral of the reward over time, so the derivative of v(t) is the reward itself
    - 5. Error between predicted reward and actual reward
- Slide: Possible Reward Prediction Error Signal in the Primate Brain

- The moral of this story is that it seems the neurons in VTA don't just tell you what the reward was, and they don't tell you what the prediction was. Rather, they tell you how much the true reward differed from what was expected.

L3: Reinforcement Learning: Time for Action!

- Slide: Policy evaluation
  - Previously, when we associated a reward with each state, the value was equal to the reward (so we didn't call it a value). Now we have to take into account actions. To get a scalar value for each state in which actions can occur, we take the average reward, where we average over all possible actions and their associated probabilities. Different policies will lead to different values for each state.
- Slide: Selecting actions based on values
  - The big point here is that values can take the place of rewards when making a decision. This is because the values are just the expected rewards. Importantly, this leads to the fact that global optimization can be obtained using local policies. That is, you don't have to know about the entire system when you make a choice, you only have to know about the system immediately around you.
- Slide: Putting it all together: Actor-Critic Learning
  - The exploration vs. exploitation problem is a common one in decision-making. Exploration essentially means gathering more information to reduce your uncertainty about a system, whereas exploitation means acting on all the information you have. Either one alone will lead to a poor decision-making policy, as pure exploration would mean never actually using the information you gather, whereas pure exploitation would mean acting on everything, regardless of how uncertain you are.
- Slide: Autonomous Helicopter Flight
  - Well this is pretty dang cool. Although it may look like the helicopter is just going crazy, it is actually sequencing through a set of very complex maneuvers. Each of these maneuvers would be incredibly difficult to program directly into the helicopter without having it crash. However, when we use learning algorithms, we can actually have humans who are expert model helicopter flyers teach the helicopter how to fly.