Week 7 Lecture Notes

Points of clarification and fun facts re: video lectures

L1: Synaptic Plasticity, Hebb's Rule, and Statistical Learning

- Slide: Hippocampus picture
    - This is a wonderful drawing done by Ramon y Cajal. Keep in mind, however, that there are many more neurons in the hippocampus than the ones he drew!
- Slide: Long term potentiation
    - Recall what an EPSP is: an excitatory post-synaptic potential. In these plots the x-axis is time and the y-axis is voltage
- Slide: Long term depression
    - Remember that while synapses can change their strength they do not switch from excitatory to inhibitory or vice versa, as this would require them to change which neurotransmitters they release
    - You might ask what determines whether a synapse will undergo LTP or LTD. In reality it is not fully understood, but the timing of the spikes arriving at the synapse seem to play a large role in determining the resultant change in plasticity.
- Slide: Hebb's Learning Rule
    - This basically says that neurons who have very correlated activity should link up more strongly so that their activity becomes even more correlated
- Slide: Formalizing Hebb's Rule
    - This equation makes sense because v is essentially the activity of the downstream neuron
    - Notice that in the discrete implementation, the fraction to the left of the equals sign is just an approximation to the derivative
- Slide: What is the average effect of the Hebb rule?
    - **o** Here we average over all input activity **u**
    - Think about the matrix/vector expression $Q\mathbf{w}$. This yields a vector that determines how much each of the weights change. What aspect of Q and $\mathbf{w}$ would cause the i-th weight to increase? The i-th weight would increase if the other inputs with which the i-th input neuron is highly correlated are the same inputs that have strong connection weights onto the downstream neuron. Further, if all the weights start out equal, the weights that will increase the most will be those of the neurons that have correlated activity. So two neurons that are active simultaneously as they talk to a downstream neuron will increase their synaptic strengths, something observed in experiments.
- Slide: Covariance rule
    - The covariance rule is quite similar to the correlation matrix strategy, but the main difference is that now all quantities of interest are looked at with respect to their average. This allows synaptic weight changes to take on both positive and negative values.
- Slide: Are these learning rules stable?

    - When we say "length" of **w**, we are not talking about how many elements are in the vector. Instead we're referring to the magnitude, or *norm*, of **w**, which is the square root of the sum of the elements squared
    - One biophysical mechanism that might play a role in normalizing the strength of synaptic weights is called homeostatic plasticity
- Slide: The brain can do statistics!
    - The covariance rule does what it does in this case because everything is first centered around the mean before correlations are taken

L2: Introduction to Unsupervised Learning

- Slide: Using neurons to represent clusters
    - Remember that the vector $\mathbf{w}_A$ is the list of synaptic weights onto neuron v_A and likewise for $\mathbf{w}_B$
- Slide: Updating weights given a new input
    - Presumably the activity of the input neurons represent something about some stimulus. The whole purpose of this sort of modeling is to design a system where certain neurons respond to certain aspects of the stimulus. Thus, we are beginning to make a model that goes hand-in-hand with our original models about single stimulus-selective neurons, except that now a fair bit of the computation is implemented in the structure of a network.
- Slide: Competitive Learning Example
    - There are lots of clustering techniques out there in the world, but this one is very cool because it allows the system to learn the clusters one data point at a time, and it does so in a way that is biophysically plausible
- Slide: Enter… Unsupervised Learning
    - Unsupervised learning is a fascinating field, and those of you who have taken the Andrew Ng's Coursera course on machine learning are probably somewhat familiar with it
    - It is basically the field of finding patterns in datasets without using any hints (or without any training data), and in general, it is a very hard problem, because it's hard to know what patterns you should be looking for! However, there are quite a few principled methods that are starting to appear rather useful.
    - In the prior and likelihood example, *v* represents some set of hidden variables (the index of the Gaussian in the mixture-of-gaussians model), whereas **u** represents the data you actually see. Often the goal is to figure out what the hidden variable associated with a data point is given only the data point itself and some knowledge of the structure of how the hidden variables generate the observations. The other, deeper and more difficult goal, is to figure out what sort of generative structure best describes your dataset (e.g., learning the mean and variance of the Gaussians in the Gaussian mixture model), i.e., for a truly unsupervised problem one would like to learn both the structure and hidden variables.
- Slide: EM algorithm for Unsupervised Learning
    - If you remember nothing else about EM, just know that it is a very useful algorithm when you have three things: observed data points, hidden

variables, and parameters of the generative model. The EM algorithm helps you figure out the hidden variables and model parameters in an iterative way: first it holds the hidden variables fixed and learns the model parameters; second it holds the model parameters fixed and infers the hidden variables
    - It has been proved that if the E and M step can be successfully performed at each iteration, the EM algorithm will necessarily converge

L3: Sparse Coding and Predictive Coding

- Slide: Eigenvectors strike again?
    - Recall that the eigenvectors of the covariance matrix of a data set with the largest eigenvalues are the "directions" along which most of the variance of the data lies. In a 2D cloud of data points, the covariance matrix would have two orthogonal eigenvectors, and the one with the higher eigenvalue would point in the direction of maximum spread of the data. With many dimensional data points (such as images), they do the same thing, but it's harder to visualize things as a cloud of data. However, in this case, each eigenvector is just a vector with the same dimensionality as all of your image vectors, so you can represent it as an image.
- Slide: A linear model for natural images
    - Dimensionality reduction allows image compression, but this is not necessarily what the brain is doing at each level of visual processing. Thus, we need to generalize our understanding of how the image might be represented.
    - These two equations are equal, but it is often nicer to write things out as a vector/matrix equation rather than as a summation
    - Here, we can think of **v** as being our hidden variable—the hidden weights of the component functions **g**—and as $\mathbf{g}_i$ as being the parameters of the model. Using the hidden variables and the parameters, we get a data point, which is an image.
- Slide: Defining the Generative Model: Likelihood
    - Hopefully this doesn't sound so crazy. In English, we're just finding the most likely set of hidden variables/causes **v**. And in fact, this terminology translates fairly well over to math.
- Slide: Bayesian approach to finding **v** and learning *G*
    - In short, Bayes' rule allows us to write out the probability of **v** in terms of an evidence based portion (what we think **v** should be just looking at the data point **u** i.e., image) and a prior portion (some previously assumed constraints—in this case, that most of the elements of **v** should be zero [sparsity])
    - The gradient is just the generalization of the derivative to scalar valued functions of multiple arguments (it tells you the change in the function with respect to changes in any of the arguments)
- Slide: Finding **v** for a given input
    - This is really cool! It provides an example of how a recurrent neural network might actually implement this exact learning rule. That is, as soon

as an image is presented, the network finds its neural representation **v** using something quite akin to the gradient ascent algorithm you might write up on a computer.

- Slide: Learning the synaptic weights G
    - Similarly, there is a realistic model that implements the learning rule for *G*! And this is just synaptic plasticity. Of course, the synaptic weights will change much more slowly than the electrical activity of the neurons. One might imagine that the synaptic weights are learned during development, when your visual system is first exposed to the various natural images in the world, and then, while they can still change when you're older, it's primarily the activity of the neurons that changes in order to represent new images.
- Slide: Predictive coding model of the visual cortex
    - This provides a relatively simple explanation for one of the purposes of feedback networks in the brain: it allows comparison between a predicted signal and an actual signal. This in turn allows things like context to play a large role in your visual experience of a scene. For example, knowing the context of a scene might cause "edge-detector" neurons in V1 to respond more strongly when they are detecting the edge of an object you believe to be in the scene.