Week 4 Lecture Notes

Points of clarification and fun facts re: video lectures

L1: Information and Entropy

- Slide: Information and Coding Principles
  - Info. theory provides a precise mathematical framework for talking about how much information a code can contain.
  - If you find this stuff interesting, check out this half-hour documentary on Claude Shannon, the father of information theory: http://www.youtube.com/watch?v=z2Whj_nL-x8
  - Information is a term that is used all the time in colloquial English, but in Info. Theory, it has a precise definition that gives a mathematical handle on many complex processing systems
- Slide: What is entropy?
  - Consider the following situation: you have some system with a hidden state, such as a box with five coins in it that are each either heads or tails. There are two easy ways to think about the entropy of this system, and they are equivalent.
    - 1. The entropy is a measure of the number of different states the system can take on. A box with five head/tail coins can take on $2^5 = 32$ states. The entropy is just the log of the number of states: $\log_2(2^5) = 5$ (for fans of statistical mechanics, this should sound familiar). We use logarithms so that the entropy only grows linearly as the number of coins increases. All of the math and neuroscience would be just as consistent if we didn't use logarithms, but it would be a lot messier.
    - 2. It's the number of bits (if we use $\log_2$) we need to specify the state, which is equivalent to the number of yes/no questions you need to ask about it. With five coins, you need to ask five questions to know the state: is the first coin heads or tails? Is the second coin heads or tails? And so on.
  - Intuitively, entropy can be thought of as a measure of uncertainty. If there is high entropy, there are many possible states the system could take on, so you're very uncertain about it.
  - Information is gained when the entropy of the system decreases, i.e., when the uncertainty decreases, i.e., when the number of possible states the system could be in decreases.
    - For example, if you originally had 5 coins in the box and you didn't know anything about them, the entropy is $\log_2(2^5)$. If someone gives you a message telling you that the first four coins are all heads, then the number of possible states decreases: it could either be all heads, or 4 heads and one tail. Thus, the entropy goes from $\log_2(2^5) = 5$ to $\log_2(2^1) = 1$. The difference between the initial entropy (5) and the final entropy (1) is 4, which is a quantification of the *information* gained

        from the message. In neuroscience, the uncertain system might be some stimulus, and the message would be the spike-train of one or more neurons.

    - We don't have to use $\log_2$. We could have used base 3 or 4 or 10 or 100, but with base 2 our natural units for counting entropy and information are bits, which we are familiar with because these are the fundamental information carrying units in a standard computer.
    - If all states are equally probable, then the entropy is just the log of the number of states (you should verify this yourself using the formula that follows). If certain states are more probable than others, then we have the more general formula for entropy $H = -\sum_i p_i \log(p_i)$, where i indexes the different states. For an explanation of why this formula is the natural (and only) generalization to states with non-equal probabilities, see the appendix to Shannon 1948 or the derivation included in the supplementary materials.
    - Keep in mind that we never talk about the entropy of a single state. We always talk about the entropy of a group/distribution of states, or rather a group/distribution of possible states.
- Slide: How much of the variability in *r* is encoding *s*?
    - When we gain information from a message (such as a spike or spike-train *r*) about a stimulus *s*, we talk about two quantities: the total entropy and the noise entropy.
        - The total entropy represents the total number of states the message *r* can take on regardless of what stimulus there is. If there are 100 time bins and each bin can equally probably contain a spike (before we know anything about the stimulus) then the total entropy of the message is $\log_2(2^{100}) = 100$.
        - The noise entropy represents the total number of states the message *r* can take on after/when the stimulus occurs. If the neuron encodes the stimulus really well, then *r* will only be able to take on a small number states. For example, maybe the stimulus always causes the neuron to spike in each of the first 95 time bins, but in the last 5 time bins the neuron still spikes with probability ½ per time bin. Thus, there are only 2^5 states available to the neuron after/when the stimulus is presented, which is much smaller than the original 2^{100}. The log of the number of states, conditioned on the stimulus, is called the noise entropy (because the variability in the state distribution is caused only by noise, and not by differing stimuli), and in this case is $\log(2^5) = 5$.
            - Another stimulus might cause the neuron to be silent in each of the first 95 time bins, and then spike with probability ½ in each of the last five. If this were the case, the neuron would encode the same amount of information about both stimuli. Intuitively, this means

that by reading out the neuron's spiking pattern you can figure out what stimulus caused it.

    - Thus, when a neuron encodes a stimulus, the stimulus causes the neuron to "choose" a certain firing pattern or set of firing patterns and to have only a small variability around that choice. The nature of the chosen sets of firing pattern then presumably can be interpreted as meaning something about the stimulus.
    - The information gained from a message/firing pattern about a stimulus is the total entropy minus the noise entropy. Since different stimuli might yield different information gain, we often look at the average information gain (averaged over the different stimuli and their prior probabilities), which is called the *mutual information*. This is characteristic of the message-transmitting system as a whole, rather than of a specific stimulus.

- Slide: Mutual information
  - Intuitively it makes sense that if there is a high mutual information between two random variables then the two random variables are not independent.
    - Suppose your two random variables are the daily temperature and the cloud cover in your city. By knowing what the cloud cover is you could narrow your guess about what the temperature is. Thus, the cloud cover contains some information about the temperature, i.e., there is a high mutual information between the cloud cover and temperature. This is quite consistent with the fact that these are not very independent random variables. When it's cloudy out, the temperature is more probably low. When it's not cloudy out the temperature is more probably high.

L2: Calculating information in spike trains

- Slide: Calculating information in spike trains (*information*)
  - We calculate mutual information by computing the reduction in the entropy of the spike-train distribution given a stimulus, averaged over the stimuli. Interestingly, there is a very nice symmetry here. If we'd computed the reduction in the stimulus distribution entropy given a spike-train, averaged over spike trains, we would have gotten the same result. Since in an experimental setting we are forced to approximate distributions, one method is often much more useful than the other.
- Slide: Apply grandma's recipe
  - In P(s), s is a vector representing the stimulus over some length of time
  - Approximating P(s) by sampling at several points in time, rather than by sampling over the ensemble of possible stimulus, can be done because of the ergodicity of the stimulus. Roughly speaking, a random

process is ergodic if its characteristics do not change over time. Thus, a white noise stimulus is ergodic.
        - Presenting a long stimulus each trial allows us to sample P(s).
        - Presenting that same long stimulus over many trials allows us to sample P(w|s)
    - In the equation on the slide, *i* indexes the different stimuli we've sampled in our attempt to approximate P(s)
- Slide: Learning about the LGN's code
    - When calculating mutual information between a spike train and a stimulus, you have to choose some definition of your spike train (called a word).
        - One parameter is how long each word is. If you only use words that are extremely short, so that they can only fit one spike, for example, then the maximum possible entropy of the word distribution is $\log_2(2) = 1$. Since entropy can only decrease to zero, you can therefore get at most one bit of information from each word, which isn't very much.
        - The other parameter is the width of the time bin you use to discretize your spike train. How long your time bin is tells you something about which words are the same, and which aren't. For example, if a bunch of words contain one spike each, but the time of that spike jitters a little bit, a very tiny choice of time bin will tell you that all those words are different, whereas a larger choice will tell you that a lot of them are the same. Thus, the choice of time bin is kind of like choosing how much of the message you want to attribute to signal and how much you want to attribute to noise.
        - What Reinagel and Reid did was to quantify mutual information between the words and stimuli as a function of those two parameters.
            - One problem that you'll run into if you do this yourself is that for very long words, it's really hard to get a sample that adequately approximates the probability distribution. Thus, you have to use an approximation when calculating the entropy.
- Slide: Information in single spikes
    - This is important: we use mutual information because it quantifies how much a neural response could theoretically be telling us about a stimulus, but without having to make a model (like a GLM, etc.) for what the neuron is doing.
- Slide: Information in single spikes
    - To get the information about a stimulus from a single spike, we need to calculate H(SPIKE) – H(SPIKE|STIMULUS). H(SPIKE) is computed from the prior distribution over whether or not there is a spike in an arbitrary bin. This distribution has only two values: probability of spike and probability of no spike. The probability of the value being

spike is the average firing rate times the width of the time bin. When a stimulus is presented, this probability distribution might change, thus the entropy might change. Since it might change in different ways for different stimuli, we take the average conditional entropy, averaged over the stimulus distribution.

- Slide: Information in single spikes
    - o Don't be scared of this formula, the first term is just the total entropy written out instead of left in sum form. The second term might be confusing at first. The stuff inside the integral is the entropy of the response for a given stimulus (conditional entropy), but aren't we supposed to average over stimulus? Yes, we are. However, since the stimulus is *ergodic*, an average over different stimuli is equivalent to an average over time, so the integral, with the 1/T in front, takes care of the stimulus average.

L3: Coding principles

- Slide: Efficient coding
    - o When we map a stimulus onto a "symbol", that symbol may be many things. It could be the instance of a single spike or no spike, or it could be a spike train. In this slide, it is the instantaneous firing rate of a neuron at a moment in time.
    - o If we have a monotonic input-output function, then technically each stimulus will map to only one output, and you will have a perfect code. However, if each stimulus maps somewhat noisily/probabilistically to an output, then we want our input-output function to be constructed in such a way that there is "as little overlap as possible" between the response distributions the different stimuli map to. The "best" input-output function will be the one that maximizes the mutual information between the distributions of response and stimulus.
- Slide: Variation in time
    - o Different stimulus distributions will have different optimal input-output functions. A good adaptive system should therefore change its input-output function when the stimulus distribution changes.
        - ▪ Remember that input-output functions can be computed straightforwardly from a dataset in which a white noise stimulus is presented and spikes are recorded from a neuron.
- Slide: Redundancy reduction
    - o If you want to make the perfect code, it may seem natural to have each neuron's response be independent of the rest. For example, if you have two neurons that can each be 0 or 1: if they are independent maybe the possible signals will be (0,0), (0,1), (1,0), (1,1), i.e., they can encode one of four different signals. If they are not independent, though, maybe the possible signals will be (0,0), (1,1), which means they can only encode one of two different signals. However, lack of independence also makes for fewer errors, so there is a balance to be achieved.

- Slide: Representing natural scenes sparsely
    - In this case, I(**x**) is I(x,y), i.e., a function of an x and y coordinate. Different images correspond to different functions, because their coordinates will have different pixel values. The goal of the Olshausen and Field experiment was to assume to find a set of “basis” images/functions, each also a function of (x,y) such that most images could be constructed pretty accurately (faithfulness) by adding only a few (efficiency) of the basis functions together. The first term in the cost function is a faithfulness term, and the second term is an efficiency term.