John Larkin
12/22/16
Coursera: Computational Neuroscience
Class Notes

Computational Neuroscience

Course Highlights:

- Some light neurobiology
- PCA and eigenbases
- Backpropagation
- Circuit analysis for neuromodels
- Eigenfaces

WEEK 1 – Introduction to Computational Neuroscience

1.1 Course Introduction

- Descriptive Models
    - how do neurons respond to stimuli and how is that quantitatively encoded
    - how can we extract info from neurons (**decoding)**
- How can we simulate a single neuron?
- Why do brain circuits operate the way they do?

At the end of the course…

- should be able to quantitatively describe what is going on with a neuron or a network
- simulate behavior of neurons
- formulate computational neurons

1.2 Descriptive Models

- Goal: explain how brains generate behaviors
- Going to characterize what nervous systems do, how they function, and why they operate in particular ways
    - Descriptive models (what)
    - Mechanistic models (how the neural system does what it does)
    - Interpretative models (why)
- Output from brain cell → *action potential*
- **Def:** *receptive field:*
    - **Specific properties of a sensory stimulus that generate a strong response from the cell**
- Retina – layer of tissue at the back of the eyes
    - Inverted image projected onto back of the eyes
    - Retinal ganglion cells – conveying information about the image to other parts of the brain
- Information from the retina passed to the *Lateral Geniculate Nucleus (LGN)* which then passes information to the Primary Visual Cortex V1.

- Center surround LGN receptive fields are displaced because of the preferred orientation of the primary visual cortex

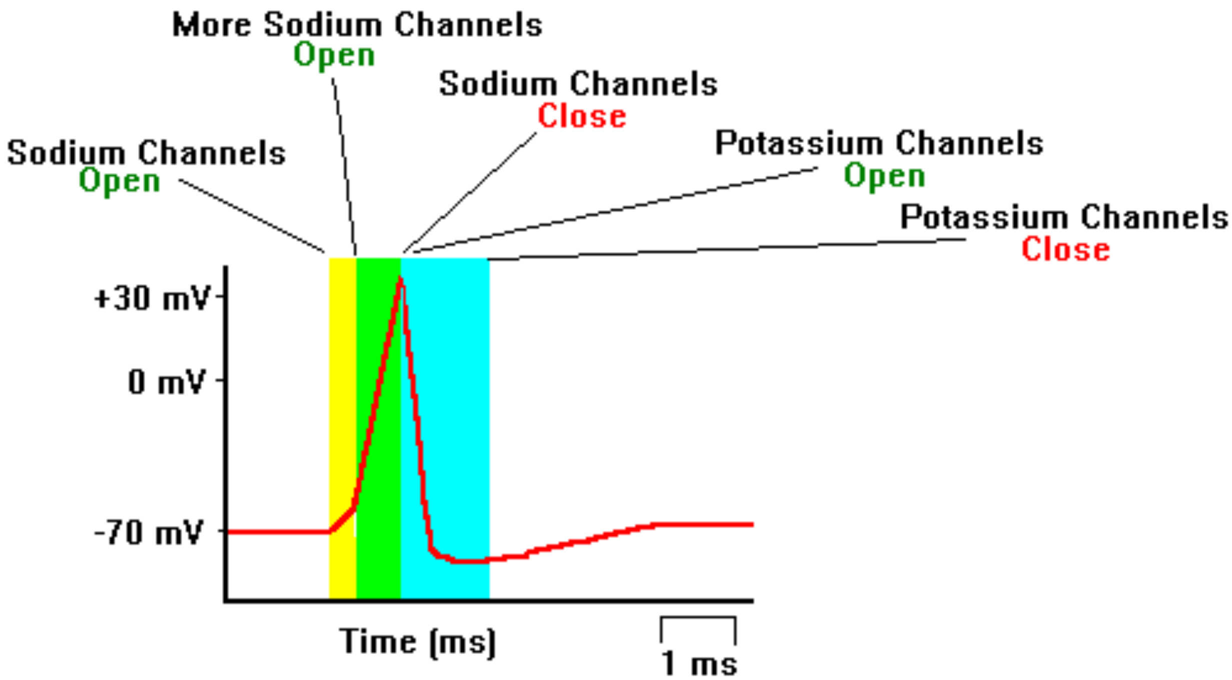
1.3 Mechanistic and Interpretive Models

- *Efficient coding hypothesis* – suppose goal is to represent images as faithfully as possible using neurons with receptive fields
- Given image I, we can reconstruct with a linear combination of receptive fields multiplied by the respective neural response
- We care about **minimizing** the total square pixel wise error and also making sure they're as independent as possible?
- Idea is like start with random receptive field and then run the coding algorithm on natural image patches
    - What is the efficient coding algorithm?
        - Sparse coding
        - Independent component analysis
        - Predictive coding
- Conclusion: the brain *may* be trying to find faithful and efficient representations of the natural environment

1.4 The Personality of Neurons
Essentially neurobio 101

- Main character: **cortical neuron**
    - Very small about 25 micron
- **Visual cortex**
    - **Axons form the pyramidal track in motor system**
- **Neuron doctrine**
    - Neuron is fundamental structural and functional unit
    - Neurons are discrete cells
    - Information flows from dendrites to the axon via cell body
- Dendrites are like the inputs
- EPSP – excitatory post-synaptic potential
- A bunch of these get fed into the dendrites and then essentially the summation of these is the action potential
- If some threshold is reached, then we have this *action potential* which is the output
- **Def** *neuron*
    - Leaky bag of charged liquid
    - Neuron insides enclosed within cell membrane
        - Cell membrane is a lipid bilayer
        - Impermeable to charged ion species
        - BUT there are ionic channels
            - The ionic channels let ions flow in and out
    - Maintains a potential difference across membrane

- Concentration of ion difference leads to -70 mV
- Ionic channels
    - Voltage-gated: prob of opening depends on membrane voltage
    - Chemically-gated: binding to a chemical causes channel to open
    - Mechanically-gated: sensitive to pressure or stretch
- Synapses
    - Junctions between neurons
    - Changes in local membrane potential
- Voltage gated channels cause action potentials
    - Depolarization opens sodium channels
    - Really about the sodium and potassium balance
    - Downward spike of action potential is from the sodium channels



- The wrapping of part of the axons is called *myelin sheath*
- The myelination of axons allows for *fast long-range spike communication*
- **Action potential hops from one non-myelated region to the next**
    - These non-myelinated regions are called *node of Ranvier*
    - This is essentially active wire → lossless signal propagation

1.5 Making Connections: Synapses

- Synapse – connection between two neurons
    - Electrical synapses – gap junctions
        - Helpful for when you need to synchronize
        - Neurons fire simultaneously
    - Chemical synapses – neurotransmitters

        - Basis for learning and memory
        - Changes the way the other neuron is affected simply by changing density
    - Can be excitatory or inhibitory
        - **Def** *excitatory*
            - Tends to increase the post synaptic membrane potential
            - Tends to excite membrane b
            - Neurotransmitters could be: glutamate
        - **Def** *inhibitory*
            - Tends to decrease the post synaptic membrane potential
        - So there is a spike, release of neurotransmitter, ion channels open, sodium influx, depolarization
- *Synapses are the basis for memory and learning*
- Allow for learning through: *synaptic plasticity*
    - Hebbian Plasticity
        - If a neuron repeatedly takes part in firing another neuron, then the synapse between those neurons is strengthened
        - *"Neurons that fire together, wire together!"*
        - Evidence: long term potentiation (LTP)
            - Experimentally observed *increase* in synaptic strength
        - Long term depression (LTD)
            - Experimentally observed *decrease* in synaptic strength
        - LTD and LTP are generally confirmed with decrease in EPSP size
    - Synaptic plasticity depends on spike timing!
    - If input is **after** output → LTD
    - If input is **before** output → LTP

1.6 Time to Network: Brain Areas and their Function

- Mainly two types of nervous systems
- **Peripheral Nervous System (PNS)**
    - Two main components
    - **Somatic** – nerves connecting to voluntary skeletal muscles and sensory receptors
    - Ex. Moving your arm and hand to shake a friends hand → utilized the SOMATIC nervous system
        - *Afferent Nerve Fibers (incoming)*
            - Axons that carry info away from the periphery to the CNS (central nervous system)
        - *Efferent Nerve Fibers*
            - Carry info from CNS to periphery
    - **Autonomic**
        - Nerves that connect to heart, blood vessels, etc.
        - Guilty of "fight or flight" reaction
- **Central Nervous System (CNS)**

- Spinal Cord + Brain
- **Spinal Cord**
    - Local feedback loops → reflex arc
        - Ex: jumping up when you step on a nail
        - Or jerking at a hot surface
    - Descending motor control signals → activate spinal motor neurons
        - Ex: brain tells your body to walk. Your spinal neurons are the ones that control this. So this way you can walk and also talk.
    - Ascending sensory axons
        - Convey *sensory* information from muscles and skin to the brain
- <u>**BRAIN**</u>
    - Region
    - **Hindbrain – Medulla oblongata, pons, cerebellum**
        - *Medulla oblongata*
            - Breathing, muscle ton
        - *Pons*
            - Connected to cerebellum
            - Involved in sleep and arousal
        - *Cerebellum*
            - EQUILLIBRIUM
            - Language and attention
            - Coordination and timing of voluntary movements
    - **Midbrain and Retic Formation**
        - *Midbrain*
            - Eye movements, visual and auditory reflexes
        - *Reticular Formation*
            - Modulates muscle reflexes
            - Regulates sleep
            - Wakefulness and arousal
    - (near center) **Thalamus and Hypothalamus**
        - *Thalamus*
            - "relay station" for all sensory information to the cortex
            - regulates sleep and wakefulness
        - *Hypothalamus*
            - Right below the thalamus
            - BASIC NEEDS (the four f's) <- lol:
                - **FIGHTING**
                - **FLEEING**
                - **FEEDING**
                - **MATING**
    - **Cerebrum**

        - Consists of cerebral cortex, basal ganglia, hippocampus, and amygdala
        - Perception, motor control, cognitive functions, emotions, memory and learning
        - *Cerebral Cortex*
            - Layered sheet of neurons
            - 1/8th of an inch thick
            - 30 billion neurons. 10,000 synapses each.
            - 300 trillion connections in total
            - Six layers of neurons
- Neural vs Digital Computing
    - The brain is **massively parallelized**
    - **Adaptive connectivity**
    - Digital computing:
        - More sequential via CPUs with fixed connectivity
    - Large computational analogs
        - Information storage: physical/chemical structure of neurons and synapses
        - Information transmission: electrical/chemical signaling
        - Primary computing elements: neurons
        - Computational basis: unknown

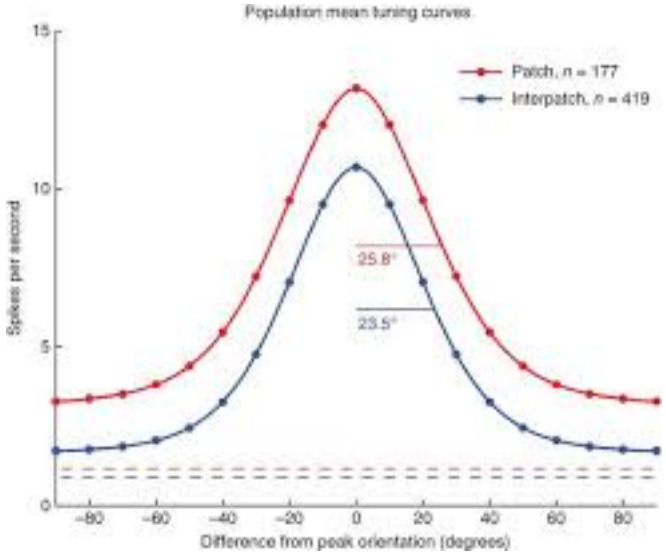
WEEK 2 – Neural Encoding and Decoding

2.1 What is the Neural Code?

- Tool for recording from the brain: fMRI
    - Functional magnetic resonance imaging
    - Measures spatial perturbations in the magnetic field
        - The changes are caused from blood oxygenation
        - As blood flows around you can see the underlying neural activity
- EEG's also just show activity for a bunch of neurons
- Calcium imaging is another way to read the neural code
- **What is the actual neural code?**
    - Let's look at the retina
    - *Retina* – sheet of cells at the back of the eyeball
        - Take light from the lens and converts to electrical signals
    - Raster plot – way of visualizing multiple iterations
    - Each neuron encodes a bit of the movie (from the experiment)
- Two questions:
    - Encoding: how does a stimulus cause a pattern of responses?
        - Stimulus → response
        - P(response | stimulus) *encoding*
    - Decoding: how do the responses tell us about stimulus?

- Response → stimulus
- P(stimulus | response) *decoding*

- Neuron response is some type of average firing rate of generating a spike
- Tuning curve
  - Frequency vs orientation of light
  - Looks about Gaussian



- There is higher order of spatial recognitions
- MRI's highlight different regions when shown faces vs houses
- Tuning curves can be difficult to record
- Building up complex selectivity
  - Brain areas build up the complexity of stimulus representation
  - Geometric in retina and thalamus, to V1 (orientated edges) and then V4.
  - Higher order areas are less sensitive to details such as color or location.
  - This is the idea behind hierarchical features in a feed forward way

2.1 Neural Encoding: Simple Models

- Basic coding model
  - linear response
    - r(t) = theta * s(t) (maybe – theta * s(t – tau))
    - just going to be delayed and scaled by a little bit
  - Temporal filtering (convolution)
    - We expect response to depend on the **combination of recent inputs**
    - r(t) = sum from k = 0 to n of s_{t-k} f_k
    - this is like convolution
    - in fact exact definition. See Cheever's page for refresher.
    - Example:
      - Running average
      - Leaky average

- Spatial filtering
  - Connected with receptive fields
  - So $r(t) = \sum s_{t-k} f_k$ temporal
  - $r(x,y) = \sum_{x'=-n,y'=-n} s_{x-x',y-y'} f_{x',y'}$
  - The receptive field is f. How similar is it to the receptive field is expressed by f
  - Often our receptive field f, is going to be a difference of Gaussians
  - Difference of Gaussians really just picks up the edges
- Spatiotemporal filtering
  - Both space and time are going to be best
  - We need a combination
- Another solution is to have a linear filter and a nonlinearity
  - Something like:
  - $r(t) = g(\int s(t-\tau) f(\tau) d\tau)$
  - How do you find the components of the model?

2.3 Neural Encoding: Feature Selection

- A good basic coding model: combination of a linear filter and a nonlinear input-output function
- One problem is of dimensionality
- Need to find the feature that drives the neuron
- Just enough so we can learn what really drives cell
- Start with s(t) and discretize
- What is the right stimulus to use?
  - Gaussian white noise
  - We choose a new Gaussian number at each frequency
  - The prior distribution is the distribution of the stimulus
  - Multivariate Gaussian – Gaussian no matter how we look at it
- Determining linear features -> one good way is to take the average
  - The vector through this average → spike triggered average
  - Then we can project all of the other points and project along that axis
- Linear filtering = convolution = projection
- Looking for stimulus feature *f* which is a vector in high dimensional stimulus space
- **Summary: find a feature by:**
  - **Stimulate with white noise**
  - **Reverse correlation to compute spike triggered average**
  - **This is good approximation to our feature**
- Still though how do we compute input / output w.r.t. feature
- P(spike | stimulus) → P(spike | component of the stimulus extracted by linear filter)
- Then use Bayes Rule
- P (spike | s1) = P(s1 | spike) P(spike) / P (s1)

- Denominator is called the prior remember
- And P (s1 | spike) is the spike conditional distribution
- P(spike) is independent of the stimulus

- P(spike | s1) = P(s1 | spike) P(spike) / P(s1)
  - Let's assume random → this means that is if the blue and red don't change
  - Then we might have filtered out the right feature
  - What we want to see is a nice difference between the prior and the spike conditional
  - This means that our input/output curve will be interesting and we can predict high firing rates
- Let's add the possibility of multiple features
- This essentially means there are several filters
- We could use PCA!! Ahhh
  - This way we get like the main dimensionality
  - As the video puts it, general, famous, and kind of magical tool for discovering low dimensional structure
  - The components correspond to orthogonal set of vectors that span the cloud
  - The important dimensions are some unknown linear combination of dimensions
  - Gives a new basis set to represent the data → lots of compression
  - Here, it is going to be some basis of our features
  - Tangent: eigenfaces!!
    - We can rep almost any new faces as sums of different eigenfaces
- PCA picks out the dimension with the largest amount of variance
- Then we project the rest of the data into the feature space
- We're trying to find interesting features in the retina
- We find an "on" and an "off" feature
- Using this technique, we can plot our data in the two feature axes and we can find the on and the off features
- NOTE: the two features are not the on and off feature themselves, but they allow a coordinate system where we can see the structure

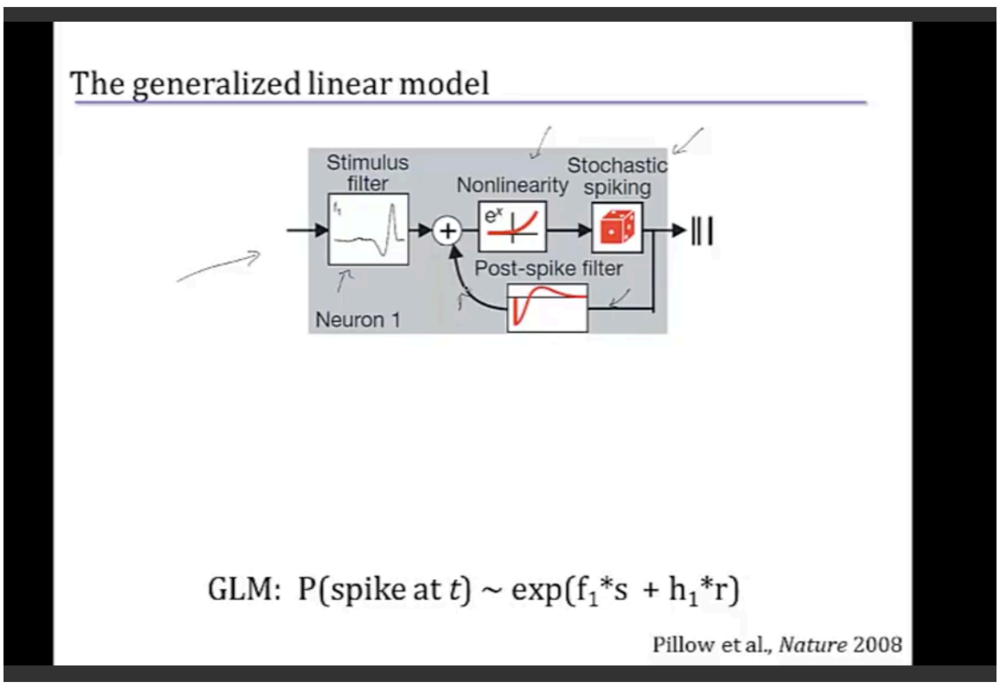
2.4 Neural Encoding: Variability

- Recall the Gaussian function:
- $P(x) = Ae^{-(\frac{(x-x_0)^2}{2\sigma^2})}$
- When we use something like PCA, making sure that we have a stimulus that's as symmetric as possible with respect to coordinate transformations
- But what if we don't use PCA, and we just look at the prior and the conditional distribution and say, can I find a filter? Meaning, like when I project the stimulus onto it are the distribution and prior as different as possible
  - Standard for measuring the difference between two probability distributions:
  - **KULLBACK-LEIBLER DIVERGENCE (DKL)**

- $D_{KL}\big(P(s), Q(s)\big) = \int dsP(s) \log_2 \frac{P(s)}{Q(s)}$
- So we just want to maximize this f
- Kind of turns into an optimization problem

- Maximally informative dimensions
  - Choose filter to maximize DKL between spike conditional and prior distributions
  - So we just vary our filter around, to maximize the DKL
  - Trying to find a stimulus component that is *as informative* as possible
  - This is a really powerful technique because it can generate
  - HOWEVER, A DOWNSIDE IS THAT THIS IS A VERY TOUGH OPTIMIZATION PROBLEM AND GLOBAL OPTIMIZATION IS TRICKY
- Finding relevant features
  - Single filter determined by conditional average
  - Family of filters from PCA
  - Information theoretic methods that use whole distribution
- Assumption that we make is that every spike is independent of others
  - Bernoulli trials
  - So kind of like a coin flipping
  - Dividing time sample into multiple time bins
  - Sequence of n time bins where n = T / Δt
  - ***Binomial distribution***
    - P = probability of firing
    - Distribution: $P_n[k] = \ p^k(1-p)^{n-k}\ (n \backslash choose\ k)$
    - The n choose k is because we don't care about the way we're arranging those k spikes
    - Average: np or rT
    - Variance: np (1-p)
    - Fano factor: F = 1
    - Interval distribution = P(T) = r exp(-rT)
    - Fano factor – tests if something is a Poisson distribution or not
    - If fano factor == 1: it is Poisson
    - Here, we have defined r as the rate or probability of per units of time
    - T is our time
    - We do some calculations from binomial and binomial → Poisson
    - Ex problem:
      - Suppose that while a stimulus is present, a neuron's mean firing rate is r=4 spikes / second. If this neuron's spiking is characterized by Poisson spiking, then the prob that the neuron fires *k* spikes in *T* seconds is given by:
      - $p(k) = \frac{(rT)^k e^{-rT}}{k!}$

- What is the prob that when this stimulus is shown for one second the neuron does not fire any spikes?
- e^-4 bc p(0) = 1 * e^-4 / 1
    - Intervals between spikes have exponential distribution
- Two strong traits of Poisson:
    - Fano factor == 1
    - Interval distribution: exponential distribution of times
- So then we can look at the slope of the number of spikes vs the mean count and then we can look at the slope
- If distributed Poisson, then the slopes should all be 1. So looking at the variance vs the mean count should have a slope of about 1
- Poisson nature of firing and randomness that we need takes care of random background noise
- Poisson assumes spike time independent
- Real neurons have refractory period that prevents the cell from spiking immediately
- Generalized linear model:



- Exponential non linearity → able to find all parameters of the model, using an optimization scheme that is globally convergent
- More generality but model now more complete in another way
- GLM = general linear model
- *Time rescaling theorem*
    - Use Poisson nature to test whether we have captured everything
    - We can predict our output spike intervals and scale them by firing rate that's predicted

- Take interval times and scale them by firing rate
- These new scaled intervals should be distributed like a pure Poisson process
- As a single clean exponential

QUIZ 2

1. A cosine function is not a linear filtering system
2. **The definition of a spike triggered average for a neuron is The set of stimuli preceding a spike, each averaged over time.**
    a. **I got this wrong. The correct answer is The averaged stimulus values over a given time before a spike that elicit a spike. That should have been obvious from the python script but alas…**
3. Sampling rate is 1sample/500s. so in 1s / 500 Hz = 0.002 sample period. Sampling period is the inverse of the sampling frequency. This is 2 ms.
4. # time steps in our average vector is 300 ms / width between interval = 300 ms / 2 ms from #3 = 150.
5. Just len(num_spikes) = 53583
6. See corresponding code
7. Leaky integration? Because we can see that things are decaying away prior to the spike
8. We can kind of think of this neuron like a capacitor. I had to look this one up because I wasn't sure. But yeah so it's kind of charging up right? So like the best thing is going to be a constant positive value because then it will gradually charge up and fire it's neuron.
9. PCA is the best of the ways

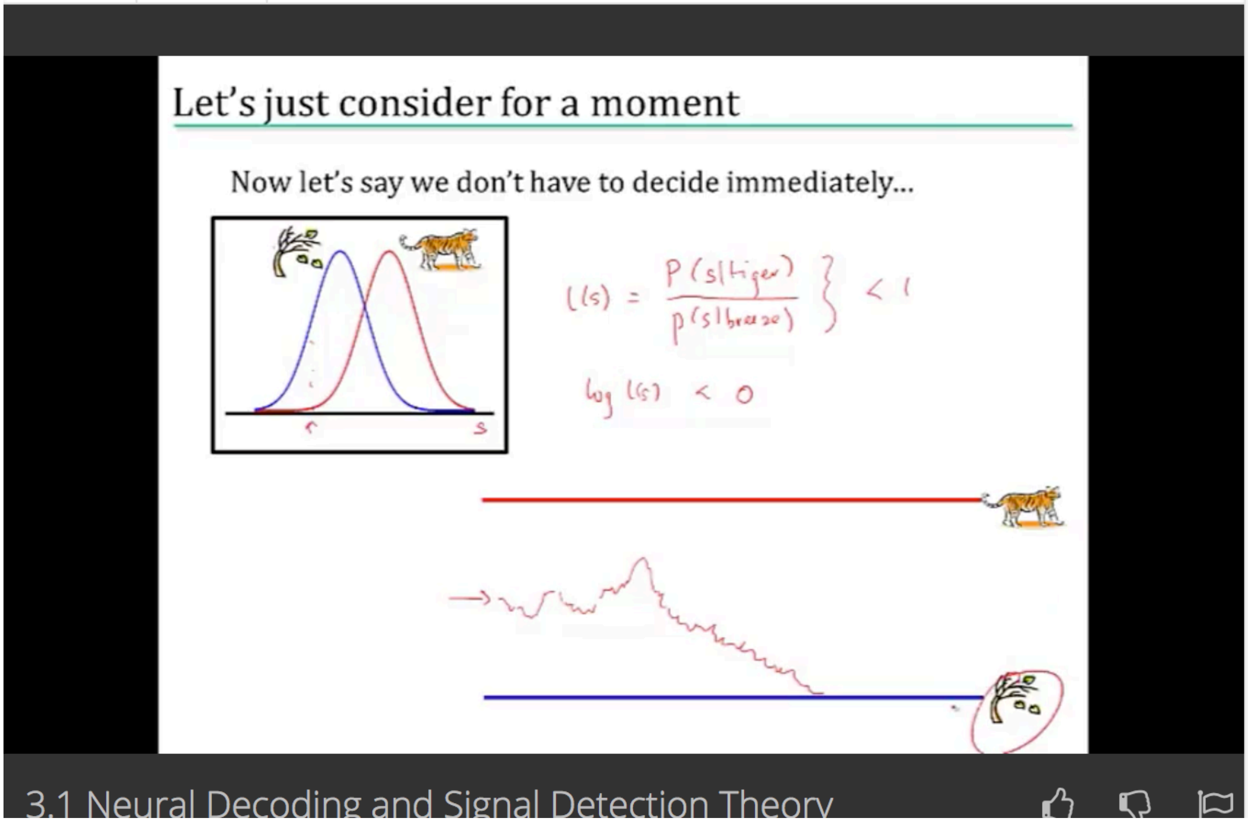
<u>WEEK 3 – Extracting Information from Neurons: Neural Decoding</u>

3.1 Neural Decoding and Signal Detection Theory

- Really going to choose between two cases:
    - Single neuron
    - Range of choices, where there are a few neurons that might be affected by the stimulus
- Also how do we decode in real time
- Famous experiment to determine how noisy sensory information was interpreted
    - Monkey would focus on a screen
    - Watch a pattern of random dots move across the screen
    - Monkey trained → follow the dots. Tracking the dot patterns.
    - Dot pattern is noisy. Hard to tell which way it's going.
    - Fraction that the monkey actually gets right is a function of the coherence. It looks like a sigmoidal function almost
- Signal Detection Theory
    - We can generate some graphs
    - R is the number of spikes in a single trial
    - Two probability distributions dist. Normal
    - P(r | -) and P (r | +)

- We want to map some range of r
- This means some threshold
- The intersection between the two Gaussians would maximize the percentage correct
- P_corr = P(+) P(r \geq z | +) + p(-) (1 – p(r \geq z | -)
- False alarms: P[r \geq z | -)
- Good calls: P(r \geq z | +)
- These probabilities p(r|-) and p(r|+) are known as the likelihoods
- Choosing the maximum likelihood

- Likelihood ratio
    - Putting a threshold on the likelihood ratio
    - $\frac{p(r|+)}{p(r|-)} > 1$ wheneeve we choose plus
    - This is the most efficient statistic to use, it has the most power for its size
    - This is called the Neyman-Pearson Lemma
    - https://en.wikipedia.org/wiki/Neyman%E2%80%93Pearson_lemma
    - Really cool lemma actually
- Seems to be a close correspondence between decoded neural response and monkey's behavior
- So why do we have so many neurons? Tbd
- Log odds!! Ah Zucker talked about this in mobile
- So we have
- $l(s) = \frac{P(s|tiger)}{p(s|breeze)}$
- $\log(l(s)) = \log p(s|tiger) + \log p(s|breeze)$

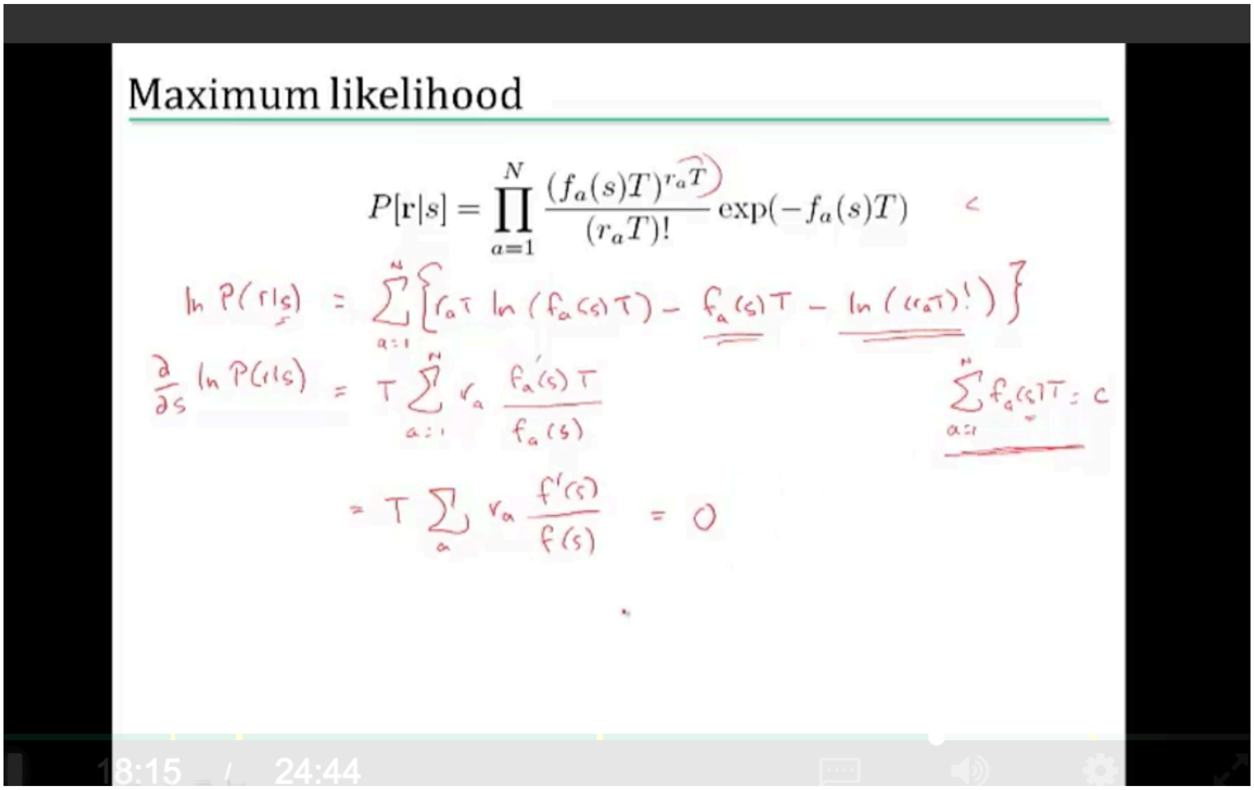


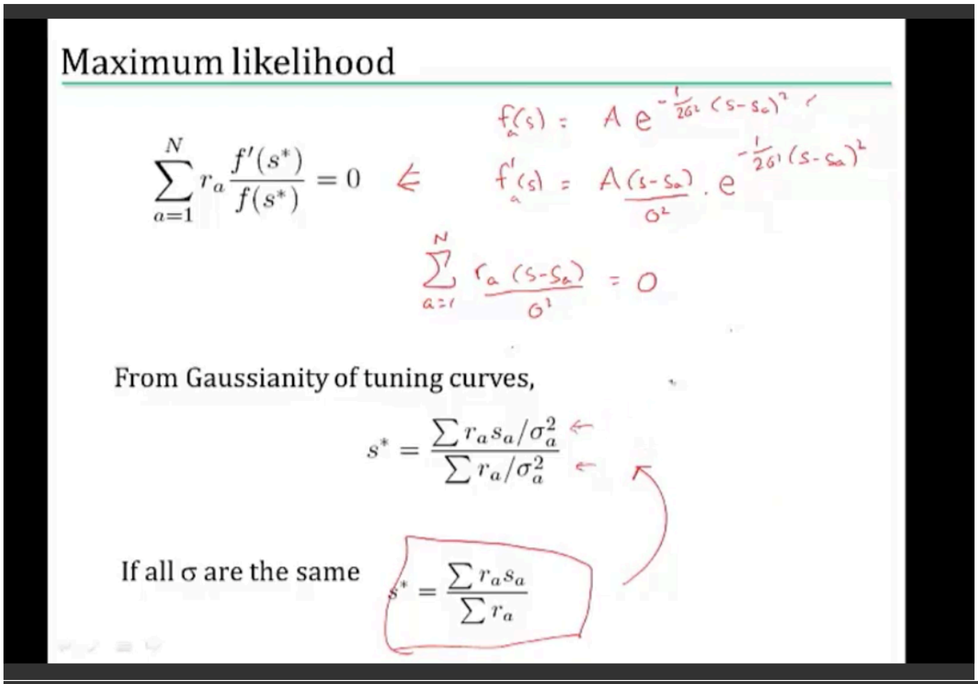
- Firing rates ramp up until a certain sure decision
- But back to our trial…
- What is the actual probability that we have a tiger? It's really low! We need to take into account the *priors.*
- The wind or a tiger?
    - Rods in your eyes can response to light and even a single photon
    - So if we adjust our probability distributions then we can pick out instances when there is a significant difference in firing rate
    - Building in cost
    - We have multiple loss functions
- Loss Functions
    - Loss_minus = L_minus P[+|r]
    - Loss_plus = L_plus P[-|r]
    - Cut your losses: answer plus when loss_plus < loss_minus
    - New criterion for the likelihood ratio:
        - $\frac{p[r|+]}{p[r|-]} > \frac{L_{+}P[-]}{L_{-}P[+]}$

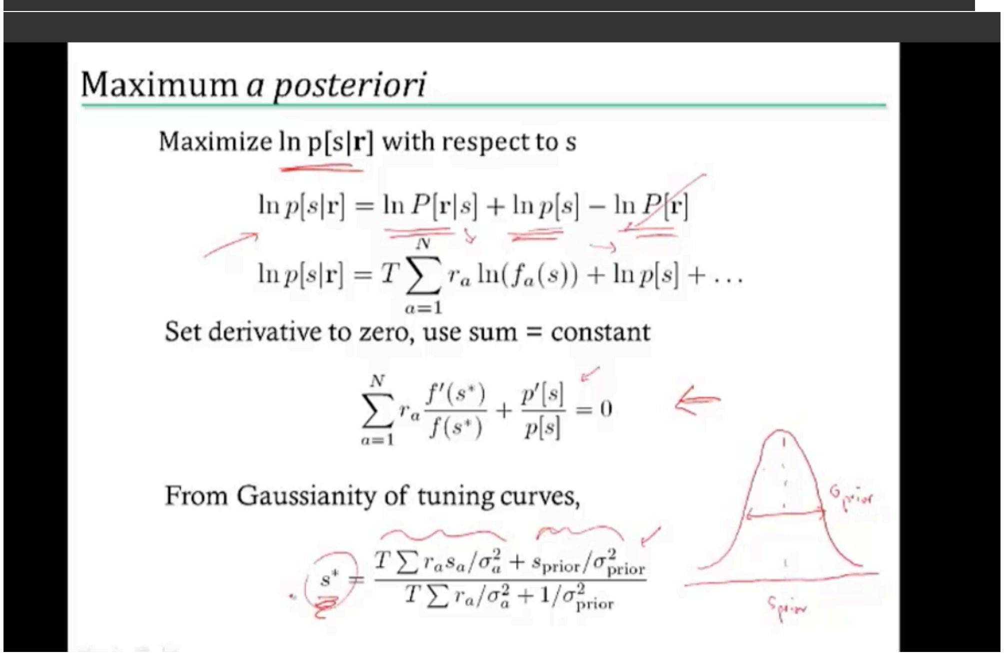
3.2 Population Coding and Bayesian Estimation (kind of a tough one to get through)

- Crickets are sensitive to wind. Like wicked sensitive.
- All because of cricket cercal cells.

- These neurons respond with peaks in one of the four cardinal directions, which is 45˚ to the animal. Left and right, front and back.
- The curves are approx. cosine, so that neurons respond to cosine of angle. Neuron's firing rate is proportional to the projection of the wind velocity.
- Bayesian Inference
  - $p[s|r] = \frac{p[r|s]p[s]}{p[r]}$
  - $a\ posteriori\ distribution = liklihood\ function * prior \frac{distribution}{marginal\ distribution}$
  - Maximum likelihood s* which maximizes p[r|s]
- Decoding an arbitrary continuous stimulus
  - Assume independence
  - Assume Poisson firing
    - Spikes are random and independent
    - $P_T[k] = \frac{(rT)^k \exp(-rT)}{k!}$
    - Then we want, r_a to stimulus s
    - That is the firing rate to a stimulus
    - $P(r_a|s) = \frac{(f_a(s)T)^{r_aT} \exp(-f_a(s)T)}{(r_aT)!}$
    - $P(r_a|s) = \Pi \frac{(f_a(s)T)^{r_aT} \exp(-f_a(s)T)}{(r_aT)!}$ because we're assuming independence and then we go from a = 1 to N
    - We can take the log
    - The math got pretty hair so here are some photos


Maximum likelihood
$P[\mathbf{r}|s] = \prod_{a=1}^{N} \frac{(f_a(s)T)^{r_a T}}{(r_a T)!} \exp(-f_a(s)T)$
$\ln P(r|s) = \sum_{a=1}^{N} \{ r_a T \ln(f_a(s)T) - f_a(s)T - \ln((r_a T)!) \}$
$\frac{\partial}{\partial s} \ln P(r|s) = T \sum_{a=1}^{N} r_a \frac{f_a'(s)T}{f_a(s)}$
$= T \sum_a r_a \frac{f'(s)}{f(s)} = 0$
$\sum_{a=1}^{N} f_a(s)T = c$
18:15 / 24:44





- And then we want to take the derivative and set that equal to zero to find the most likely value

    - Ok I didn't want to write all the equations out in a word doc so here are the pictures
    - This method takes care of weighting them based on the variance
- Limitations
  - Tuning curve / mean firing rate
  - Correlations

3.3 Reading Minds: Stimulus Reconstruction
… should go back and rewatch

- One day – play back our dreams?
- Extend model to handle varying continuously in time.
- We want to find estimator s_bayes that gives us best possible
- Introduce error function L(s, s_bayes)
- Least squares cost. So just L(s, s_bayes) = (s-s_bayes)^2
- Solution: s_bayes = int ds p[s|r] s
- Reading minds: fMRI
  - Output predicted on BOLD signals (blood oxygen signals)
  - It therefore has a delay
  - ^ that's one way
  - Another way is a motion energy filter

3.4 Fred Rieke on Visual Processing in the Retina

- A few rods out of 1000s are contributing signals
- All rods are generating noise
- Averaging would be a disaster
- Have access to rod signal and noise properties
- So we see evidence for a nonlinear threshold between rod and rod-bipolar cells
- Vision is working under conditions where the vast majority are generating noise
  - Want to scale the distributions to take into account the prior probability

QUIZ 3

- Stimulus *s*. Can be one of two values *s1 or s2.* Firing rate response *r.* Under stimulus s1 repose rate is roughly Gaussian ~ N(5,.5^2). S2 ~ N(7, 1).
- It is twice as bad to mistakenly think that it is s2 rather than s1.
  - So this is saying something about where we're thresholding.
- "The disease is very rare. The prior probability of being positive for the disease is therefore very low. MAP (maximum aPRIORi) takes this into account; MLE does not. The mathematics differ in that MAP includes a term for the prior."
  - From my stats.stackexchange question I asked about

John Larkin
12/22/16
Coursera: Computational Neuroscience
Class Notes

WEEK 4 – Information Theory and Neural Coding

4.1 Information and Entropy

- Going to start by talking about entropy and information
- How to compute information for neural spike trains
- And what can this tell us about coding
- Ok so back to the monkey example:
    - Information quantifies surprise
    - Some overall prob *p* that there's a spike
    - P(1) = *p*
    - P(0) = 1-*p*
    - Information(1) = -log_2 p
    - Information(0) = -log_2 (1-p)
- Why does the information have this form?
- Each *bit* of information specifies location by factor of 2
- What we're really doing is multiplying the probabilities
- **Entropy – average information of a random variable**
    - **Measures variability**
    - Units are in bits
    - Entropy counts the yes / no questions
    - Entropy = $-\sum p_i \log_2 p_i$
    - Or in continuous $-\int dx p(x) \log_2 p(x)$
- This is essentially just hunting for the binary search
- $H = -\sum p_i \log p_i$
- $p_i = \frac{1}{8}$
- $H = -\sum_{i=1\ to\ 8} \frac{1}{8} \log\left(\frac{1}{8}\right)$
- $\sum \frac{1}{8} * -3 = 3$
- Three questions to find car (in example) and that's exactly the entropy
- Maximize the entropy
    - Compute the entropy as a function of the probability p
    - What does having a large entropy do for a code?
    - Gives the most possibility for representing inputs
    - You want to find the value of p such that H has a max
    - If p == ½ then those two symbols are used equally as often
- Entropy tells about intrinsic variability of the outputs
- Week 2 was asking how do we know what our stimulus was
- But now, we need to incorporate our error chances
- Assume the same error
- How much of the entropy is accounted for by these errors?
- Total entropy: H[R] = -P(r_+) log P(r_+) – P(r_-) log P(r_-)
- Noise entropy: H[R|+] = -qlog q – (1-q) log (1-q)

- These stimulus driven entropies are called *noise entropies*
- Amount of entropy that is used in coding the stimulus
- MI(S, R) = Total entropy – average noise entropy
- $MI = -\sum_r p(r)\log p(r) - \sum_s p(s)[-\sum_r p(r|s)\log p(r|s)]$
- Entropy and information
    - Fixing p
    - Vary the noise probability
    - *When there is no error, the mutual information is 1 to 1. Information is just the entropy of th response.*
    - *As the error rate increases, error probability grows larger and larger.*
    - ***If p(r|s) = p(r), the mutual information MI of r and s is zero, because this is saying r and s are independent and therefore no information is gained***
    - ***If response is perfectly predicted, then the MI is 1, because total information is conveyed by 1.***
- Mutual information measures relationship
    - The information quantifies *how independent* R and S.
    - Going to use the Kullback Leibler divergence.
        - This is a measure of the difference between two prob distributions
        - Normally it is between a "true" distribution and a theoretical distribution
        - https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence
    - $D_{KL}(P,Q) = \int dx P(x)\log\frac{P(x)}{Q(x)}$
    - Going to generalize so that distributions are functions of s and r. So we would need to integrate over both s and r
    - $\int ds\, dr\, P(s,r)\log\frac{P(r,s)}{P(r)P(s)} = \int ds\, dr\, P(s,r)\log\frac{P(r|s)P(s)}{P(r)P(s)}$
    - $= \int ds\, dr\, P(s,r)\,[\log P(r|s) - \log P(r)]$
    - $= -\int ds\, dr P(s,r)\log P(r) + \int ds\, dr\, P(s)P(r|s)\log P(r|s)$
    - The first bit we can just integrate over s
    - The second term is going to be the entropy of P(r|s)
    - This gives us exactly what is expected
    - $I(S,R) = H[R] - \sum_s P(s)H[R|s]$
- Calculating mutual information
- Take one stimulus s and repeat many times → obtain P(R|s)
- Compute variability due to noise: noise entropy → H[R|s]
- Repeat for all s and average → \sum_s P(s) H[R|s]
- Compute P(R) = \ sum_s P(s) P(R|S) and theoretical entropy

4.2 Calculating Information in Spike Trains

- Two methods: single spikes, vs lots of spikes
- Mutual information = diff(total response entropy , mean noise entropy)
- Methodology:

- Divide up voltage train into letter size Δt and length T
- Essentially then just have a 1 if we have a spike 0 if not
- From this, compute p(w_i)
- $H[w] = -\sum p(w_i) \log p(w_i)$
- How to sample P(S) → average over time
- For each time, we're going to given set of words P(w|s(t))
- Then we have an average entropy
- Choose length of repeated sample long enough so that the sample the noise adequately

- Information in single spikes is similar to what we just saw in the previous lecture
- After a bit of math and some assumptions, the *information per spike is:*
- $I(r, s) = \frac{1}{T} \int_{0\ to\ T} dt \frac{r(t)}{r_{bar}} \log \frac{r(t)}{r_{bar}}$
- No explicit stimulus dependence (NO NEED FOR CODING / DECODING MODEL)
- The rate *r* does not have to mean rate of spikes → can be rate of any event
- Limitations of information:
  - Spike precision, blurs r(t)
  - Mean spike rate

4.3 Coding Principles

- Natural stimuli
  - Huge dynamic range
  - Power law scaling
- Efficient coding:
  - In order to have max entropy output, a good encoder should match its outputs to the distribution of its inputs
  - Should be able to stretch its input axis (IN REALTIME) so that it can accommodate the variations in the overall scaling
- Feature adaption
  - Power spectrum and signal to noise ratio are large factors for the predicted receptive field at certain light levels.
  - Center becomes broader in low light levels
  - Choose filter to maximize Kullback Leibler divergence between spike conditional and prior distributions
- Redundancy reduction
  - Neural systems should be trying to encode as efficiently as possible
  - Maximize the entropy should take into effect the marginal added together
  - Correlations can be good → error correction + correlations help discrimination
- Neuron populations should be as **SPARSE** as possible
  - Let's say we right down a set of basis functions, phi,
  - Any image can be expressed as a weighted sum
  - $I(x_{bar}) = \sum_i a_i \phi_i(x) + \epsilon(x)$

- Want to penalize having too many images
- Fourier basis represents in sin and cosine… but not necessarily sparse because the power spectrum is broad
- Spare code – excites a minimum number of image

- Classic and State of the Art Methods:
  - Models for how stimuli are coded in spikes
  - Models for decoding stimulus from neural
  - Information theory
  - A very quick glance at how coding strategies might shape other things

<u>WEEK 5 – Computing in Carbon</u>

5.1 – Modeling Neurons

- About to delve into circuit diagrams
- Differential equations (largely first order)
- Hodgkin Huxley model
  - Should be a review from biomedical signals
- Basic review of circuit diagrams
- Membrane patch
  - We have a lipid bilayer
    - Like a capacitor
  - Pores
  - Channel
- Cell battery
  - Outside the cell: higher sodium, chlorine and calcium contents
  - Inside higher potassium levels
  - Concentration gradient = battery
    - Nernst Equation $E = \frac{k_b T}{z\, q} \ln \frac{[inside]}{[outside]}$
- Currents flow through ion channel

5.2 – Spikes

- What makes a neuron compute?
- Neuron responds to steps and thresholds
  - Uncover the non-linearity
- Gate has subunits that need to be open for things to go through
- *Gating* depends on subunit state
  - *P_k = n^4*
  - *n is open prob*
  - *1-n is closed*
- Review of biomedical signals
- Independent probability of being open
- Hodgkin and Huxley's nobel equation

  - Specifies conductance for different channels
  - Time constant dictates how rapidly each variable corresponds to voltage change
- Hodgkin-Huxley Model
  - Two different places
  - Biophysical realm → ion channel physics, additional chennls
  - Simplified models → fundamental dynamics, analytical tractability

5.3 – Simplified Model Neurons

- Can one build a large model with lots of neurons
- Caputring the basic
  - Force the model to be linear
  - dV/dt = f(V) + I(t) # nonlinear because of f(V)
  - dV/dt = - a(V-V_0) + I(t)
  - Like a passive mebrane
  - C_m dV/dt = -g_L(V – V_e)
  - Integrated firing model ^
- Exponential integrate-and-fire neuron
- The theta neuron
  - Great for periodic neurons
  - One dimensionsal
  - $\frac{d\theta}{dt} = 1 - \cos\theta + (1 + \cos\theta)\, I(t)$
- Two dimensional models
  - Need a phase plate diagram
  - Can find the nullclines – the place where the derivative is equal to zero
  - Fixed point is going to be the intersection between two nullclines
- Various neurons have different firing rates and oscillations

5.4 – A Forest of Dendrites (should review)

- Real neurons are brutal to model
- Inject current at the cell body and record effect in dendrites
- So we're looking at the **soma** to see the response at some input
- Inputs that come in at different parts of the dendrite can have very different effects
- Theoretical basis for dendrite communication
  - PDEs!
  - Linear cables
  - Voltage V is a function of both x and t
  - Essentially a bunch of circuits distributed along a table
  - Now a spatial derivative that has to be taken into effect
  - Essentially, the diffusion equation, but we have an additional V_m / r_m
  - Time constant: $t_m = r_m c_m$
  - Space constant: $\lambda = \frac{r_m}{r_i}$

- R_m is membrane resistance
- Function decays rapidly as a function of space
    - Geometry can be extremely complicated → cable equation
    - Ion channels
    - Solution: divide and conquer
    - Each compartment = one dV/dt equation
    - If branches open a certain branching ratio, can replace each pair of branches with a single cable segment with equivalent surface area and electronic length
- Ion channels introduce the nonlinearity
- Dendrites can add a lot to neuronal computation
    - Logical operations
    - Low pass filter, attenuation
    - Coincidence detection
    - Segregation, amplification
- Example:
    - Delay lines in sound localization

Eric Shea-Brown on Neural Correlations and Synchrony

- This guy seems good
- Encoding via spikes
- Eye → optic nerve → lateral geniculate nucleus (LGN) → visual cortex
- Tuning curve – firing rates as a function of the angle of some stimulus
- Got a bunch of neurons, have a tuning curve, also variance around that mean
- Two statistics
- Still can quantify similar statistics
- **Pairwise correlation → departure from independence**
    - **Label the spike counts**
    - Pierson correlation coefficient
    - Or just correlation coefficient
    - Then you ask if that number is 0 or non-zero
    - **Correlation can degrade signal encoding**
- Turns out that you can apply this technique to numerous neurons
    - Compute the signal to noise ratio
        - Mean / variance
    - SNR → going to grow with *M (number of neurons)*
    - Then also observe the correlation coefficient
- Pairwise couples of entire population

<u>WEEK 6 – Computing with Networks</u>

6.1 Modeling Connections between Neurons

- Linear filter model of a synapse

- See online notes for this lecture
- Just listened to the audio

6.2 Introduction to Network Models

- Learned that neurons use synapses to connect
- Learned how to model with differential equations
- **FEEDFORWARD VS RECURRENT**
- Modelling networks
  - **Spiking Neurons**
    - Pro: Learning based on spike timing
    - Pro: Spike correlations
    - Con: computationally expensive
  - **Firing-rate outputs (real valued outputs)**
    - Greater efficiency, scales well to large networks
    - Ignore spike timing
  - How are they related?
- Synapse **b**
- Input spike train rho_b (t)
- $\rho_b(t) = \sum_i \delta(t - t_i)$
- $g_b(t) = g_{b,max} \sum_{t_i<t} K(t - t_i) = g_{b,max} \int_{-\backslash inf} K(t - \tau)p_b(\tau)dt$
- From single synapse to multiple synapses:
  - Each synapse has a synaptic weight
  - Assume no nonlinear interactions
  - Then total synaptic current
  - $I_s(t) = \sum_{b=1\ to\ N} w_b \int from - \inf to\ t\ K(t - \tau)p_b(\tau)d\tau$
  - We go from spike train, to firing rate
  - ***This would fail if there were correlations or synchronies***
- Suppose synaptic filter K is exponential
- Firing-rate-based network model
- **Output firing rate changes like:** $\boldsymbol{\tau_r \frac{dv}{dt} = -v + F(I_s(t))}$
- **Input current changes like:** $\boldsymbol{\frac{\tau_s dI_s}{dt} = -I_s + w \ * u}$
- Weights matrix **w**
- To get steady state, we need to set both of these equal to zero
- Static input: v_ss = F(w dot u)
- **THE RICH DYANMICS THAT ARE ACTUALLY IN THE SYANPTIC CURRENT ARE REPLACED WITH A SIGMOIDAL FUNCTION FOR ARTIFICAL NEURAL NETWORKS**
- **THAT'S ONE OF THE BIG DISTINCTIONS**
- **HENCE ARTIFICAL**
- BIG ASSUMPTIONS THAT THE SYANPSES ARE RELATIVELY FAST
- Multiple output neurons

- Then we have an input vector and an output vector
- V is now a vector. W becomes our weight matrix.
- This has all been **FEEDFORWARD NETWORKS**
- $\tau \frac{d\boldsymbol{v}}{dt} = -\boldsymbol{v} + F(W\boldsymbol{u} + M\boldsymbol{v})$
- For feedforward networks, M is a matrix of zeros
- There's no like passback without recurrent networks!!
- Linear Feedforward Network
  - Steady state: $v_{ss} = W\boldsymbol{u}$
- **Edge detectors in the brain**
  - **Primary visual cortex (V1)**
  - **Receptive fields in V1 have edge detection**

6.3 The Fascinating World of Recurrent Networks

- Want to find out how the output v(t) behaves for different M
- Eigen vectors to the rescue!
- $\tau \frac{dv}{dt} = -v + h + Mv$
- Idea use eigenvectors of M to solve differential equation for v
- Suppose N x N matrix M is symmetric
- IF m is symmetric, M has N orthogonal eigenvectors **e**_i and N eigenvalues lambda_i
- It is useful for them to be orthonomrla because then we can write our output vector using eigenvectors
  - $v(t) = \sum_{i=1\ to\ N} c_i(t) e_i$
  - Complete expression: $c_i(t) = h\ times\ e \frac{1}{1-\lambda_i}\ (1 - \exp\left(-\frac{t(1-\lambda_i)}{\tau}\right) + c_i(0) \exp\left(-\frac{t(1-\lambda_i)}{\tau}\right)$
  - If any of the lambda is greater than 1 → network explodes
  - If all of them are less than 1, network is stable and v(t) converges to some steady state value
- Network performs winner-takes-all input selection
- Gain modulation in the nonlinear network
  - Adding a constant amount to the input h **multiplies** the output
- Memory in nonlinear network
  - Network mantains some short term memory
- Nonsymmetrical recurrent networks
  - Network of excitatory and inhibitory neurons
- Linear stability analysis
  - Stability matrix
  - **THIS IS JUST THE JACOBIAN MATRIX**

NOTE: could not figure out one on the quiz. Posted on stack.

http://stackoverflow.com/questions/41492020/finding-the-steady-state-output-of-a-linear-recurrent-network

WEEK 7 – Networks that Learn: Plasticity in the Brain & Learning

7.1 Synaptic Plasticity, Hebb's Rule, and Statistical Learning

- Long term potentiation (LTP) – experimentally observed increase in synaptic strength that last for hours or days
- Long term depressison (LTD) – experimentally observed decrease in synaptic strength that last for hours or days
- **Hebb's Learning Rule**
  - If neuron A takes part in firing neuron B, then the synapse is *strengthened*
  - Formulation as a mathematical model
    - Let's start with linear feed forward model
    - We have a synaptic weight vector
    - Basic hebb rule
    - $\tau_w \frac{dw}{dt} = u\ v$
    - Discretization: $w_{i+1} = w_i + \epsilon * uv$
    - Hebb rule only increases synaptic weights (LTP)
  - Learning rules are NOT stable
  - W grows without bound
  - Covariance rule can both increase and decrease
- Start with the averaged Hebb rule: $\tau_w \frac{dw}{dt} = Q\ w$
- Solve this equation to find **w(t)** using eigenvectors
- Substitute in Hebb rule differential equation and simplify as before
- Synaptic weight vector is a linear combination
- Has terms that are exponentially dependent on the values of the correlation matrix
- *For large t, largest eigenvalue term dominates*
- *For Oja's Rule:* $w(t) = \frac{e_1}{\sqrt{\alpha}}$
- Thus we have shown the brain can do statistics
- Hebbian Learning implements ***principal component analysis (PCA)***
- **Hebbian learning learns a weigth vector aligned with the principal eigenvector of input correlation/covariance matrix**
  - DIRECTION OF MAXIMUM VARIANCE

7.2 Introduction to Unsupervised Learning

- Can neurons learn to represent clusters
- Feedforward network with two neurons
- Most active neuron in the network
  - The one whose weight vector is closest to an input
  - We can show that by looking at the Euclidean distance between vectors

- Given a new input, we can set the weight vector to the running average of all inputs IN THAT CLUSTER
- Then you pick the most active neuron

- Competitive learning and self organizing maps
  - Also known as Kohonen maps
  - Given an input, pick the winning neuron
  - Update weights for that neuron AND the other neurons in the neighborhood of the winning neuron
    - What do we mean by neighborhood?
    - We have locations assigned and the neighboring ones like literally on a 2d grid
- Unsupervised learning
  - We have causes *v*
  - Data points **u**
  - You kind of assume that there are multiple gaussians given by some prior
  - Mixture of gaussians model
  - **Goal:** learn a good generative model for the data you are seeing
    - Mimic the data generation process
  - **General approach:**
    - Given data **u,** need to
      - Estimate causes **v**
      - Learn parameters **G**
- Algorithm for learning the parameters
- Expectation-Maximization algorithm:
  - Iterating through expectation step
  - Then the maximization step
  - **E step – computing the posterior distribution of v for each u**
    - $\boldsymbol{p[v|u;G] = \frac{p[u|v;G]p[v;G]}{p[u;G]}}$
    - **soft competition**
  - **M step – charge parameters G using results from E**
    - Just updating the mean, variance, and the prior

7.3 Sparse Coding and Predictive Coding

- Hebb's learning rule implements principal component analysis
- How do we learn models of natural images?
  - Eigenvectors or principal compenents
  - Turk and Pentland
  - Eigenvectors of the input covariance matrix
  - Any face image can just be a linear summation of the eigenfaces
  - It's a basis

- This could be great for compression! But not great to extract the local components
  - Edges in a scene
  - Can't get that from an eigenvector analysis

- Define the generative model: likelihood
  - Linear model
  - u = Gv + noise
  - You're generating the likelihood based on a probabilistic model
  - **A lot of machine learning algorithms and things in engineering want to minimize the log of the likelihood**
  - $p[u|v;g] = -\frac{1}{2}|u - Gv|^2 + c$
  - if you MINIMIZE the squared reconstruction error you are MAXIMIZING the likelihood of the data
  - Prior
    - Can make some assumptions
    - Assume the causes v_i are independent
    - For any input, we want only a few causes v_i to be active
    - **SPARSE DISTRIBUTION**
      - **Also called *super-Gaussian distribution***
      - Very sharp
      - You're taking the exponential of a Gaussian
      - $p[v] = c \cdot \Pi \exp\left(g(v_i)\right)$
  - Bayesian approach to **finding v** and **learning G**
    - Going to maximize the posterior probability of causes
    - Equivalently, maximize the log posterior
    - $F(v, G) = -\frac{1}{2}|u - Gv|^2 + \sum_i g(v_i) + K$
      - maximize F with respect to v, keeping G fixed
      - maximize F with respect to G, keeping v from above
      - This is similar to the EM algorithm
      - *Normally, we just use gradient ascent*
      - $\frac{dF}{dv} = G^T(u - Gv) + g'(v)$
      - firing rate dynamics
      - $\tau\frac{dv}{dt} = G^T(u - Gv) + g'(v)$
      - First term is the error, Gv is the prediction
      - It converges to a stable value
  - Learning the synaptic weights G
    - $\tau_G\frac{dG}{dt} = (u - Gv)v^T$
    - This is the Hebbian term
    - This is almost identical to Oja's rule for learning

- **Why isn't this network just doing principal component analysis like Oja's rule?**
- Answer: Network is trying to compute a sparse representation of the image
- Learning G for Natural Images
  - **The basis vectors are a bunch of bars**
  - **Like one hot vectors**
  - **The g_i look like local edge or bar features SIMILAR TO RECEPTIVE FIELDS IN PRIMARY VISUAL CORTEX**

<u>**WEEK 8 – Learning from Supervision and Rewards**</u>

8.1 Neurons as Classifiers and Supervised Learning

- The classification problem
- Example: classifying images as faces
  - What is we just group them as +1 and -1. Could we draw a line to separate those groups?
- Recall: the idealized neuron
  - It's essentially thresholding
  - Inputs: u_i; synaptic weights: w_i, and if $\sum_i w_i u_i > \mu$ then we have an output spike
- **This is called the "perceptron"**
  - We have inputs that are either +1 or -1
  - We can build the equation $\sum_i w_i u_i - \mu = 0$ which is a hyperplane formula
  - *Perceptrons can classify*
- So the question becomes: how do we learn the weights and the threshold?
- Perceptron learning rule:
  - Adjust w_i and mu according to output error (v^d – v):
  - $\Delta w_i = \epsilon(v^d - v)u_i$ *for positive input; increases weight if error is positive decreases weight if error is negative*
  - $\Delta\mu = -\epsilon(v^d - v)$ *decreases threshold if error is positive and increases if error is negative*
- Great! So perceptrons learn any functions?
- Let's think about XOR:
  - Can't really do it
  - There's no line we can draw
  - **Perceptrons can only classify linearly separable data**
- However, **we can use multilayer perceptrons**
- What about continuous outputs?
  - Sigmoid functions!
  - Output: $v = g(w^T u) = g(\sum_i w_i u_i)$
  - Sigmoid output function: $g(a) = \frac{1}{1+e^{-\beta a}}$

  - Range of –inf to infinity and then compresses everything to one
  - Beta controls the slope
- Learning Multilayer sigmoid networks
  - You could learn weights that minimize the output error
  - $E(W, w) = \frac{1}{2}\sum_i (d_i - v_i)^2$
  - Use gradient descent!!
  - How do we change the weights for the hidden layer?
  - ***<u>Backpropagation learning rule</u>***
  - $\Delta w_{jk} = -\epsilon \frac{dE}{dw_{jk}}$
  - The answer essentially lies in the chain rule from calculus
  - Example:
  - $\frac{dE}{dw_{jk}} = \frac{dE}{dx_j} \cdot \frac{dx_j}{dw_{jk}}$
  - The error propagates down through the neural network
  - We should see the entire hidden layer affected

8.2 Reinforcement Learning: Predicting Rewards

- We learn by trial and error. Rewards are part of this
- We have some state, some reward, and some action
- We need to pick the action that will maximize our future reward
- ***Pavlov and his dog***
  - Classic conditioning experiments
  - Training: bell → food
  - After: bell → salivate
  - But how do we predict rewards delivered *sometime after* the stimulus?
- Want to have some neuron who predicts the expected total future reward?
- **Key idea: utilize dynamic programming**
  - We don't know our future rewards so we need to approximate
  - Learning the weights according to which v(t) is calculated
  - *Temporal difference (TD) learning rule*
    - $\Delta w\,(\tau) = \epsilon[r(t) + v(t+1) - v(t)]\,u(t-\tau)$
    - We have a temporal difference because we have our future prediction and our current prediction

8.3 Reinforcement Learning: Time for Action

- How does the brain use reward information to actually select the actions?
- Learn a state-to-action mapping or a **<u>policy</u>**
  - $\pi(u) = a$
  - Should maximize the expected total future reward
  - $< \sum_{\tau=0 \text{ to } T-t} r(t+\tau) >$
- However, that's using a random policy

- Values should act as surrogate immediate rewards → locally optimal choice leads to globally optimal policy
- **Markov Environment**
  - **The next state only depends on the current state and the current action**
  - **This is closely related to dynamic programming**
- **Putting it all together: <u>Actor - Critic Learning</u>**
  - Two separate components:
    - Actor (selects action and maintains policy)
    - Critic (mantains value of each state)
  - 1. Critic Learning (Policy Evaluation)
    - Value of state $u = v(u) = w(u)$
    - $w(u) \leftarrow w(u) + \epsilon[r(u) + v(u') - v(u)]$
  - 2. Actor Learning (Policy Improvement)
    - $P(a; u) = \frac{\exp(\beta Q_a(u))}{\sum_b \exp(\beta Q_b(u))}$
    - probabilistically select an action *a* at state *u*
    - This is like a softmax function
    - It lets us explore all possibilities
  - Then we repeat steps 1 and 2

<u>End of class</u>