Imagine you’re walking in the forest and you have to estimate what the orientations of these branches are. You might want to do this for a number of reasons like determining whether a branch looks like it might fall on you. This might be complicated by the fact that there are many lines in the same location, that might be in the same receptive field. For example, there are two lines in this location, one is -20 and one is 20.

Lets think for a minute about how the brain does this. Each neuron in a orientation-coding population has a tuning curve which describes its mean response to a given stimulus. In addition, the height of these curves (the gain) is modulated by the contrast of the stimulus. I’ll use these words somewhat interchangeably but contrast refers to the stimulus and gain refers to the neural response to that stimulus. However, the neural response is variable (usually modeled by a Poisson distribution) so a neuron might not respond exactly as shown here. Any downstream brain area would have to do some sort of inference to reconstruct what the actual identity of the stimulus that generated that response is. Luckily these brain areas are helped by the fact that there are many of these neurons so it has statistics on its side

So let’s see what some of these neural representations look like. Here every dot is a single neuron and I’m labeling the neurons by their preferred stimulus (which is the maximum of the tuning curve) Here is -10 and 10. However, it most models of neural coding, responses to two stimuli in the same receptive field is the sum of the responses to the individual stimuli. So the combined response might look like this. We call this stimulus mixing

And what if the contrast in the stimulus is lower. In most models, this decreases the gain of the neural response so there are fewer spikes on an average trial. This can make identifying the orientations in the original stimulus even more difficult

So demixing is a problem for the brain. I mean how often do you see a single oriented line outside of a psychophysics experiment? And in fact it is much more pervasive than just orientation, there is lots of evidence to suggest that coding of motion in MT and objects in IT, among many other brain areas, have mixed stimuli representations. So how should the brain deal with it? The Bayesian approach would be to get the probability distribution over the stimuli given the neural response you are trying to decode. Here we write $s\_1$ and $s\_2$ and for most of the talk, we will be dealing with the case where you know that there are two stimuli to be estimated. And because we need to label these, we are always going to say that s\_2 will be the bigger stimulus. If we use the mean of this as estimates for the stimuli, this provably is the best we could do given the inherent neural variability. In addition, we might be interested in the precision of our estimate which allows us to know how confident we should be or compute other probabilities like “is this branch going to fall on me?” Lets see what these posteriors actually look like.

Here is a neural response to a stimulus pair of -5, 5. Here is the distribution over possible stimuli that generated that response. The blue dot is that point and the red dot is the posterior mean, given that neural response which is the best estimate possible of the blue dot.

Across many trials there will be variability in the posterior mean estimate as reflected by this red cloud of dots. This is still provably the best we could possibly do, given neural variability.

But in complex situations, it’s pretty much impossible. Even for these simple case, my computer got pretty warm computing it.

We know (to a gross approximation) that many neurons in cortex have firing rates that are thresholded linear sums of their inputs. So not doing anything complex. But it turns out that groups of these approximate neurons can do state of the art computation in many machine learning problems. In fact, theorems have shown that there exists a network with two layers and a finite number of these type of neurons that can approximate any function arbitrarily well. So we’re going to try and use the the computational power that the brain already has to approximate Bayesian inference and train some biologically plausible neural networks to do optimal demixing.

First I will show how well this approximation works in practice in this problem (since it is not trivial to find the weights or use a small number of units). In the later part of the talk I will examine aspects of this problem that the theorem doesn’t say anything about

So in order to figure out how the brain does it, we take a neural network approach. You see here the input population with the mixed stimulus representation from earlier. On top of that we use a layer of units that compute rectified linear combination. We use these both because neurons can’t have negative firing rates and this has had a lot of success in machine learning community. Finally, we have two output neurons that compute estimates of the two orientations as linear combinations of the hidden layer activity. We trained this network by giving it 270,000 input responses and having it minimize the Mean Squared Error of the estimates it produces. There are a few things that might affect demixing ability: how close the orientations of s1 and s2 are and what the contrasts of the stimuli are. The contrast affects the gain so we are going to model that here by changes in gain.

To start investigating how well this works, we picked a set of gains and gave the network training examples with only that gain combination. We then tested the network on trials generated with that same gain combination. This is as easy as possible for the network since it knows what gains it is going to be seeing during test

In order to analyze our networks performance, we looked at different sources of error. Here is our slide of posterior mean estimates from before. We can see that although they are all centered around the true stimulus, there is a distribution of estimates and standard deviation is a measure of the width of this distribution.

Another source is bias. This deals with the mean so I’ve made the estimates a lot lighter. Again, this is the true stimulus that generated the neural responses. And here is the mean of the posterior mean estimates. This is averaged over a large number of trials. They are not exactly the same and this difference is called the bias.

For visualization purposes I’m going to show you trials where we varied the distance between the two stimuli so the x axis is the distance between the two stimuli. First we’ve plotted how the posterior mean estimate performs. We tested it on several trails at each distance and

Since you can get different networks every time you use a different ordering of the training set, we trained 200 networks. We can now look at average performance for both of these measures so the purple here represents plus/minus one SD of both of these measures for the network. And here is what it looks like as you increase gain. In addition, we can see what it looks like when the two stimuli have different gains.

OK so far so good but in the real world, you experience a variable range of contrasts and your brain should be able to deal with that. So lets see what happens when we train the network with all combinations of 1, 2, 4 and then test it on that.

For plotting, we’ll split it up into different gain combinations.

So far the network has seen everything that it’s been tested on before. But what about for examples the network has never seen before? What if we only train the network on 1, 1 and 4, 4 and see what happens?

Here it is broken out by gain like above. It seems to do a pretty reasonable job.

So far we’ve been treating the posterior as just a point estimate. Not to be confused with the standard deviation of the error distribution (average over trials vs single trial uncertainty). But let us read out other quantities related to whatever task we might be doing. For example we might want to know how confident we are in our estimate, which we might think is related to the width of our posterior. Now that the neural network can get reasonable estimates, maybe it automatically learned something about the whole posterior, even if we didn’t actually train it to do that. We can take the hidden units (second layer) as the networks internal representation so we will see if it’s possible to easily readout confidence from that layer

We’ll start by looking at the network trained with all gain combinations. The posterior width is related to the gain so it saw a wide variety of different posterior widths so it is the most likely to work. Every spike gives you some information so you might think that more spikes equals greater precision. Many constructed models of neural codes with specific computations/weights allow you to recover the precision by just summing the activations of the hidden units. So lets see if that works here. Nope.

But maybe if we do a weighted sum where we optimize the weights it’ll work. Correlation of .83! So we can just do a simple linear readout of the precision

And the just 4, 4 network actually is still pretty good.

But how good is this really? If we train a neural network to go from the inputs directly to precisions, wouldn’t that do much better? Turns out only slightly.

In the real world you don’t always know how many stimuli there are. We wanted to see how our network performs when there are are either one or two stimuli present. We trained the network to estimate the highest contrast stimulus from a population or the orientation if the single stimulus (if there was only one). This is very preliminary work but we wanted to see if our networks could do this.

For these demixing problems, it seems that we can approximate a very hard computation with a relatively easy one for the brain. When trained correctly, we get near optimal estimation even when gains are variable or never before seen and when the number of stimuli are variable. And we can read out the posterior precision linearly, basically for free

This fits into a broader agenda of showing that Bayesian computations that are mathematically very complicated can be well approximated by generic, biologically plausible neural networks

We would like to see how this generalizes to tasks where the networks don’t know how many actual stimuli there are that generated the neural response. We have some preliminary evidence that this is possible. Also, we would like to understand what the networks are doing but so far, they’ve been pretty inscrutable.