Askesis: Positive Pathway

Eric Purdy

August 3, 2014

1 Overview

The cerebellum as a whole tries to predict the output of the cerebrum, and takes over the performance of activities that are sufficiently predictable. The positive pathway is responsible for producing the outputs of the cerebellum, some of which will subsequently be filtered out by the negative pathway.

One of the cerebral outputs that is predicted by the cerebellum is the input that the cerebrum sends to the cerebellum. One function of the positive pathway is thus to predict its own input at the next time step.

As the cerebellum does its work to predict the output of the cerebrum, it relies on three kinds of information: commands from the cerebrum, context from the proprioceptive system, and its own internal model of the state of the cerebrum and the body.

We will refer to sequences of cerebellar outputs as actions, and the individual cerebellar outputs as elementary actions.

1.1 Kinds of cells

The positive pathway makes use of several kinds of neurons: mossy fibers, deep nuclear cells, and the cells of the inferior olive.

We classify mossy fibers into three kinds:

- Command mossy fibers (those coming from the cerebrum via the pontine nuclei),
- Context mossy fibers (those coming from the spinal cord),
- State mossy fibers (those coming from the deep nuclear cells).

These classes are depicted in Figure 1. This classification is novel, as far as we are aware. These names are meant to be suggestive, but the true situation is probably more complex, in that some command cells are probably used mainly to provide context for other cells' decision making, and some context cells are interpreted as commands. Nevertheless, we will use this classification in our discussion.

We classify the deep nuclear cells into three kinds:

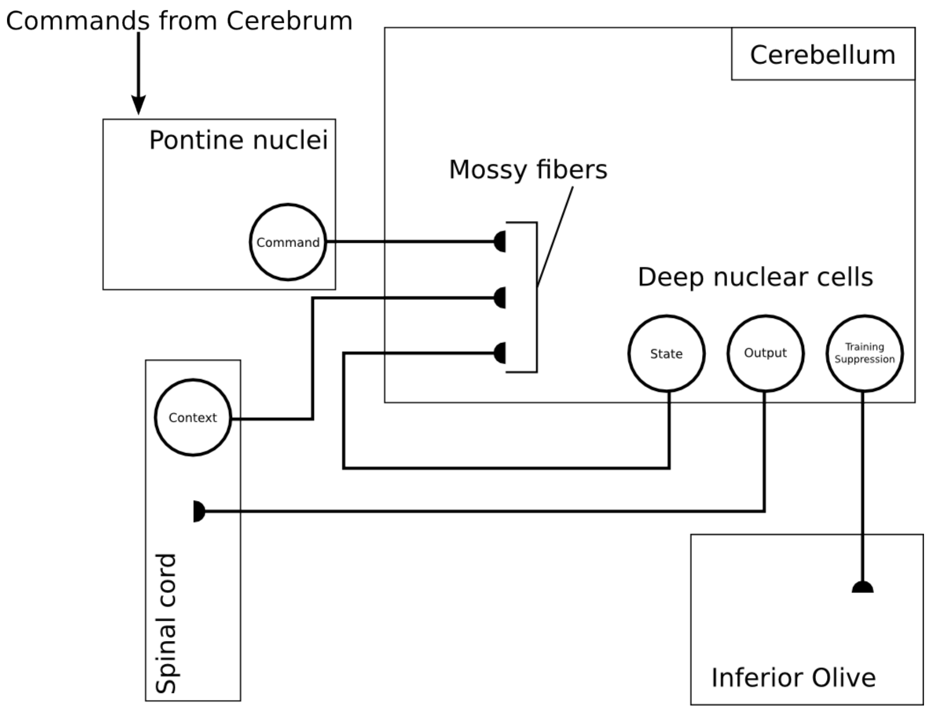

Figure 1: Classification of mossy fibers and deep nuclear cells. We have omitted intermediate cells between the output deep nuclear cells and the neurons of the spinal cord.

- State cells (those whose outputs reenter the cerebellum as mossy fibers),
- Output cells (those whose output goes to the spinal cord, possibly via intermediate neurons),
- Training suppression cells (those whose output goes to the inferior olive).

excitatory

Both inhibitory and excitatory, possibly depending on which segment of the cerebellum we're looking at.

Look these up - they use a different neurotransmitter and are inhibitory. They are also smaller than the other cells.

These classes are again depicted in Figure 1. The classification of state cells vs. output cells is again novel, as far as we are aware. The difference between these cells and training suppression cells is already known, although we have not seen the purpose of training suppression attributed to them. We will discuss training suppression cells in Chapter **??**.

Note that the state cells occur both as mossy fibers and as deep nuclear cells - they receive input as deep nuclear cells and send output as mossy fibers. We hypothesize that the cerebellum's model of the state of the cerebrum, the state of the body, and the state of the world is encoded by firings of the state cells.

In our models below, we assume a one-to-one mapping between the command cells and the state cells. We would also be OK with a one-to-one mapping between the command cells and a subset of the state cells, i.e., it is OK for some state cells not to receive input from command cells. We assume that this mapping is completely arbitrary, and may be fixed at development. The rest of the system is set up to learn around this mapping.

Word this more formally. What does OK mean? What happens if it's not one-to-one? More work, but doesn't invalidate approach. If it's just a subset, this has no effect on the mathematical model.

Finally, we have the cells of the inferior olive, which signal that an unexpected movement has occurred. In general, they just signal that a movement command has been issued; the unexpected part comes from the fact that the training suppression cells suppress the inferior olive cells coding a movement that the cerebellum has predicted.

1.2 Competent Performance

We now give an informal description of the model during performance of an action. We will give a more formal mathematical model later. The model is depicted in Figure 2.

Move figure slightly earlier?

Consider the act of walking. We need to make a conscious decision to start walking, but once we start, our body seems to continue the process on its own. People with damaged cerebellums report having to plan out such movements every time they execute them.

The cerebrum initiates actions via the command mossy fibers. Each command cell corresponds to some suite of learned actions. After the command cell fires, the cerebellum takes over and executes the desired action.

Every action can be thought of as a series of stages. For instance, during walking, we push off with one foot, shift our balance slightly to the other leg, move the foot forward, and then bring it down. In order to perform the action correctly, the cerebellum has to know what part of the walking cycle it is in. Such information is encoded by the state cells, which are the deep nuclear cells

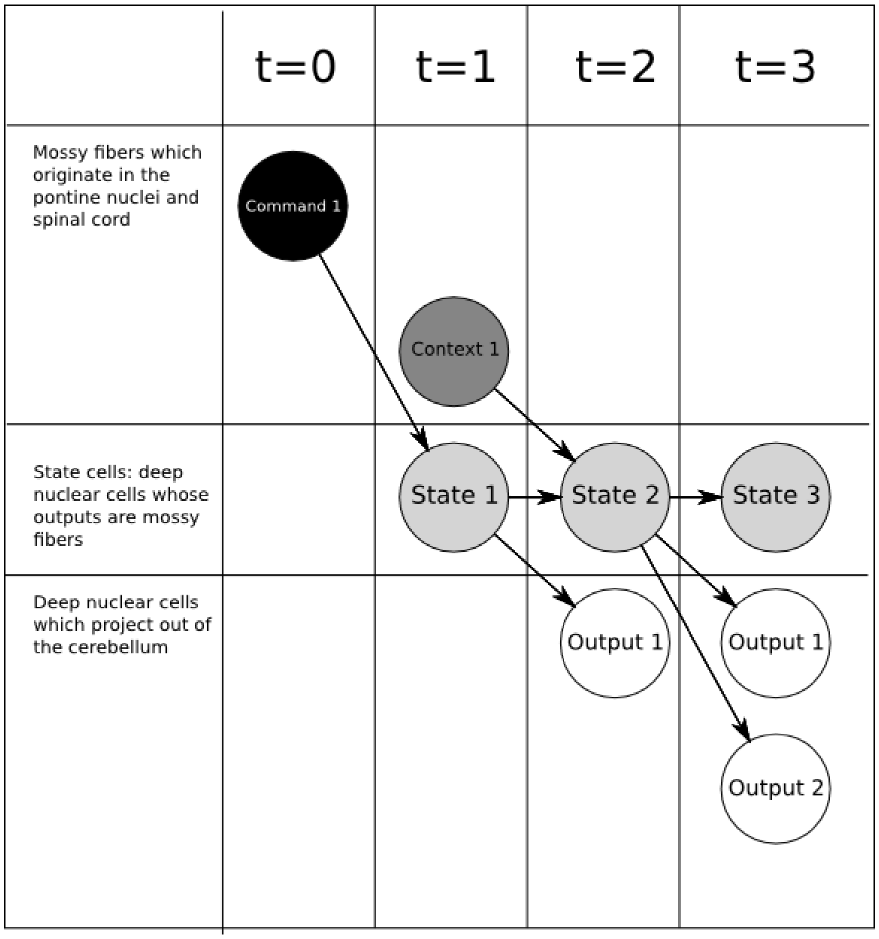
t=0
t=1
t=2
t=3
Mossy fibers which originate in the pontine nuclei and spinal cord
Command 1
Context 1
State cells: deep nuclear cells whose outputs are mossy fibers
State 1
State 2
State 3
Deep nuclear cells which project out of the cerebellum
Output 1
Output 1
Output 2

Figure 2: Overview of the positive pathway during performance. The system receives input through the mossy fibers, which is used to maintain a state coded by the activation of a state cell. State cells trigger other state cells to fire subsequently. State cells also trigger output cells to fire, producing the output of the positive pathway as deep nuclear cell firings. From $t = 2$ to $t = 3$, we have a timed transition from state cell 2 to state cell 3. From $t = 1$ to $t = 2$, we have a waypoint transition, since the context cell is used to guide the decision to transition from state cell 1 to state cell 2. See text for more discussion of the two types of transition.

whose outputs are mossy fibers. A state cell is active during a particular moment in the action.

Each state cell is responsible for two things, intiating the muscle contractions appropriate to the moment for which the cell is responsible (called "output"), and ensuring that the next state cell in the sequence fires at the next moment in time (called "transition"). The information necessary to perform output is encoded in the synapses between state cells and output cells: the stronger the synapse, the more likely it is that the state cell will induce the output cell to fire. This firing will be relayed to the spinal cord, and will eventually result in a muscle contraction. The information necessary to perform transition is encoded in the synapses between the current state cell and other state cells: the stronger the synapse, the more likely it is that the current state cell will induce the next state cell to fire at the next moment in time.

There are two different ways to decide that a transition is needed: either we know that we want to be in a particular state at the next moment in time, or some particular information comes in that suggests that we need to change to a particular state. We call the first way a "timed transition", and the second way a "waypoint transition". A timed transition simply encodes the information that one state should follow another at a particular later time. For instance, during a blink, the eye should close and then open; the commands to open the eyelid should start a fixed time after the commands to close the eyelid. A waypoint transition encodes the information that the state of the body in the world has changed. For instance, during walking, once the leg has successfully been raised the appropriate amount, this information can be relayed back to the cerebellum, and the system can advance to the next state, corresponding, say, to the point in the walking process where the leg begins to be brought back down.

The end of a movement can be encoded by simply having no strong synapses from the last state cell in the chain to any subsequent state cell. Alternatively, there can be another command to stop the motion. Stop commands require a more complex model than the other phenomena discussed here, so we will defer discussion until later.

Give a reference to when. Also really need to think about stop commands.

The organization of the positive pathway during competent performance is described in Figure 2.

1.3 Learning mechanisms

We now give an informal description of the model during learning. We will give a more formal mathematical model later.

There are two different kinds of learning that take place in the positive pathway: learning of transitions, and learning of outputs.

Transition learning allows the cerebellum to merge two skills that it has already learned. If one command is consistently issued before another command, and the delay between them is consistently t steps later, then the cerebellum will learn to transition between them. After this has taken place, the first command will automatically cause the same effects as if the second command

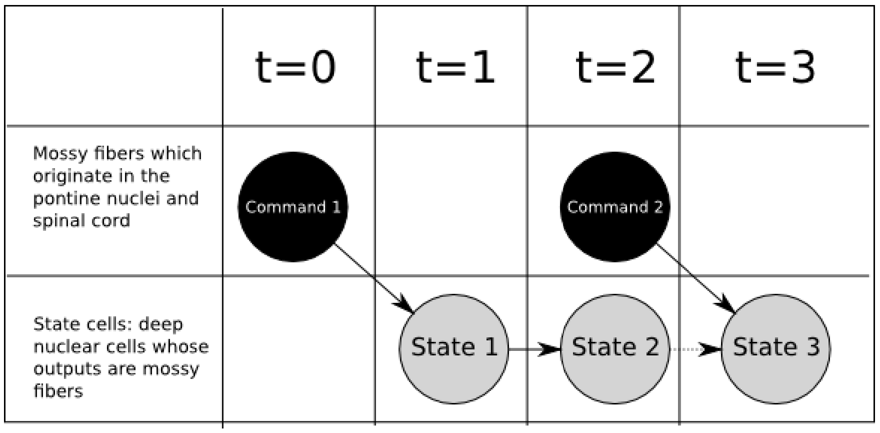
t=0
t=1
t=2
t=3
Mossy fibers which originate in the pontine nuclei and spinal cord
Command 1
Command 2
State cells: deep nuclear cells whose outputs are mossy fibers
State 1
State 2
State 3

Figure 3: Learning of positive pathway transition functions. If command 1 is consistently followed by command 2 two time steps later, the system will learn the transition represented by the dashed arrow.

had followed *t* steps later. We can think of this as "movement chaining": the movement coded for by one command cell will be automatically followed by the movement coded for by a second command cell.

Transition learning takes place at the synapses between two state cells. If one state cell is consistently active at the time step directly before another state cell is active, then the synapse between them will be strengthened. This coincidence can be caused by the consistent issuance of two commands a fixed distance apart, as shown in Figure 3. There are other ways to cause the coincidence. For instance, if a particular context cell consistently fires at the time step before a particular command cell fires, then a transition will be learned between the context cell and the command cell - this can be thought of as a "learned reflex", since it will result in the cerebellum automatically initiating an action in response to some sensory input, without the need for involvement of the cerebrum.

Output learning establishes the mapping between states and outputs, allowing the cerebellum to issue commands to the muscles via the spinal cord. Each state cell has the ability to trigger several output cells, which code for particular muscle contractions. The cerebellum is taught which output cells should be fired at a particular movement by the firing of the inferior olive. The climbing fiber collateral causes the firing of the output cell that corresponds to the movement that the inferior olive cell codes for. Output learning takes place when the firing of a particular inferior olive cell consistently coincides with the firing of a particular state cell, as shown in Figure 4.

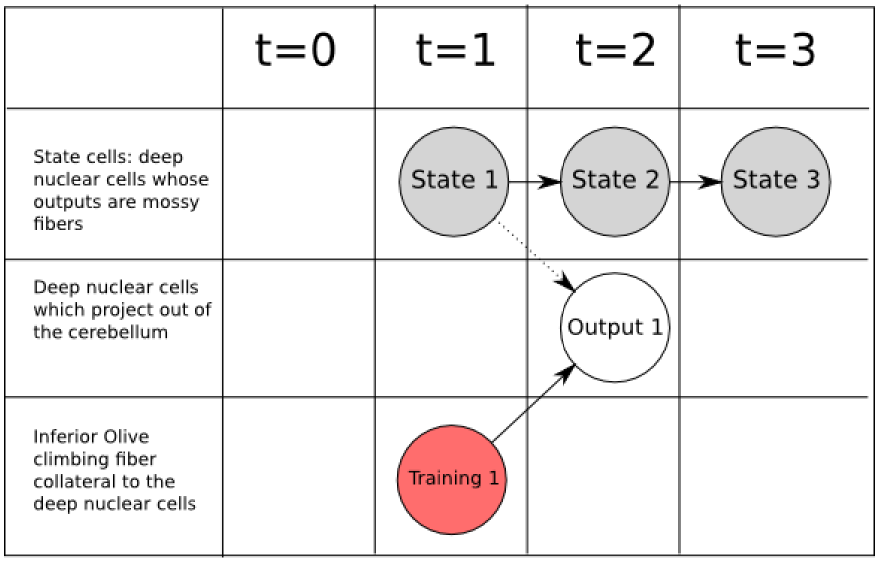
t=0
t=1
t=2
t=3
State cells: deep nuclear cells whose outputs are mossy fibers
Deep nuclear cells which project out of the cerebellum
Inferior Olive climbing fiber collateral to the deep nuclear cells
State 1
State 2
State 3
Output 1
Training 1

Figure 4: Learning of positive pathway output functions. If state cell 1 is consistently active at the same time as training cell 1, the system will learn that state cell 1 should trigger output 1, as represented by the dashed arrow. Note that learning should actually be maximized when the training cell fires slightly after the state cell fires (100 msec?).

2 Mathematical models

First, some notation. We break time into discrete points labeled by integers. If k is the name of a neuron, then we let $k(t)$ be 1 if neuron k fires at time t, and 0 if neuron k does not fire at time t.

We will assume that time is broken up into discrete, regularly spaced steps, and that all firings occur exactly at the time of a particular time step. This is obviously not realistic, but it makes everything simpler to write down and discuss. The model should still work with the messy firing timing that the brain actually uses.

What is the relationship between the various models?

We give four mathematical models of the positive pathway, largely for the sake of understanding. Each model is more complex than the model before it. Except for the third and fourth models, each model is a special case of the next model. The fourth and last model is the only one which seems biologically feasible. Even more complex models are possible, which might perform better; We have stopped at the simplest model which is biologically feasible and likely to perform reasonably well.

2.1 Preliminaries: Conditional Probability

We will be working with conditional probabilities. The *conditional probability of A given B*, written $P(A|B)$, is the probability that event A happens given that we already know that B has happened. It is defined as the probability that both A and B happen (written $A \cap B$) divided by the probability that B happens.

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

If A and B are random variables, then the events we are interested in are the event that they take on a particular value. We write the probability of the event that A takes the value α as $P(A = \alpha)$. If we write just $P(A)$, this should be understood as a function that maps values of A to probabilities, i.e., $P(A)(\alpha) = P(A = \alpha)$. A conditional probability $P(A|B)$, where A and B are random variables, should be understood as a function that maps values of A and B to probabilities, i.e., $P(A|B)(\alpha, \beta) = P(A = \alpha|B = \beta)$.

It is always the case that, for every β, $\sum_\alpha P(A = \alpha|B = \beta) = 1$, where the sum is taken over all possible values of A.

2.2 Model 0: Linearly ordered state cells

The simplest possible model that is worth discussing is the linearly ordered model. This model is able to learn a list of elementary actions to perform in a given order. It is not able to learn a looping activity, like walking. It is not able to merge two actions into a single action - it can only grow an action by adding additional elementary actions at the end of it.

In this model, the state cells are arranged so that each one receives input from one state cell and gives output to one other state cell, and they are arranged in a line. Let the state cells be denoted by $s_1, \ldots, s_n$. If a state cell s_i fires at time t, then it is more likely that s_{i+1}, the next state cell in the line, fires at time $t+1$. For the time being, it is impossible for two state cells to fire at the same time; we will remove this restriction later.

For this simple model we will assume that the command cells are in one-to-one correspondence with the state cells. Let the command cells be denoted by $c_1, \ldots, c_n$. We will assume that if the command cell c_i fires at time t, then the state cell s_i is guaranteed to fire at time $t+1$. If c_i does not fire at time t, then it is still possible for s_i to fire at time $t+1$, but only if state cell s_{i-1} fires at time t. For the time being, it is impossible for two command cells to fire at the same time.

We have the cells of the inferior olive, which we will denote by $t_1, \ldots, t_m$ (t for training).

Finally, we have the output cells. We will denote the output cells by $o_1, \ldots, o_m$. We do not assume any correspondence between output cells and state cells, but we do assume that the output cells are in one-to-one correspondence with the cells of the inferior olive. If t_i fires at time t, then o_i is guaranteed to fire at time $t+1$. t_i fires when the overall system should produce output o_i, but failed to.

The system follows the following rules:

$$\begin{aligned}
P\left[s_i(t)=1|c_i(t-1)=1\right] &= 1 \\
P\left[s_i(t)=1|s_{i-1}(t-1)=0, c_i(t-1)=0\right] &= 0 \\
P\left[s_i(t)=1|s_{i-1}(t-1)=1, c_i(t-1)=0\right] &= p_i \\
P\left[o_j(t)=1|t_j(t-1)=1\right] &= 1 \\
P\left[o_j(t)=1|t_j(t-1)=0, s_i(t-1)=1\right] &= q_{ij} \\
P\left[o_j(t)=1|t_j(t-1)=0, (\forall i) s_i(t-1)=1\right] &= 0,
\end{aligned}$$

where the p_i and q_{ij} are parameters that are learned over time. Recall that only one s_i may be active at any one time in this model, and that only one c_i can fire at a particular time.

How do we learn p_i? Given a set of observations of $s_1, \ldots, s_n$ over time, it is simply the observed conditional probability that s_{i-1} fired immediately before s_i, conditioned on the firing of s_{i-1}.

$$p_i = \frac{\#\ t \text{ such that } s_i(t)=1, s_{i-1}(t-1)=1}{\#\ t \text{ such that } s_{i-1}(t-1)=1}.$$

Note that this is not the maximum likelihood setting of parameters, since we don't look at whether a firing of s_i was caused by s_{i-1} firing or by c_i firing. This is important, because it allows the system to change its parameters in response to the sequence of commands observed.

How do we learn q_{ij}? Given a set of observations of $s_1, \ldots, s_n$ and $o_1, \ldots, o_m$ over time, it is simply the observed conditional probability that s_i fired immediately before o_j, conditioned on the firing of s_i.

$$q_{ij} = \frac{\#\ t \text{ such that } o_j(t) = 1, s_i(t-1) = 1}{\#\ t \text{ such that } s_i(t-1) = 1}.$$

Note that this is not the maximum likelihood setting of parameters, since we don't look at whether a firing of o_j was caused by s_i firing or by t_j firing. This is important, because it allows the system to change its parameters in response to the training signals from the inferior olive.

2.3 Model 1: Hidden Markov Model

This model is the same as the above, except that we don't assume that the states are linearly ordered, and allow any state cell to influence any other state cell. (We are only modeling cells within a single microzone, so really we are only allowing state cells to influence related state cells.) At every time step, if a command cell fired at the last time step, we activate the corresponding state cell at this time step. If no command cell fired at the last time step, we pick a random state cell to make active based only on which state cell was active at the previous time step. If s_i was active at the previous time step, we pick s_j with probability p_{ij}. Here $\sum_j p_{ij} = 1$, so the p_{ij} (called the "transition probabilities") give a probability distribution over j for each fixed i.

This model is a slight modification of the Hidden Markov Model in machine learning. The word "hidden" is slightly inaccurate for our case, since we can observe the states via seeing which state cell fires, and the data we are learning from includes the activity of the command cells, which are in one-to-one correspondence with the state cells. Also, we allow multiple outputs to be active at a given time, while the standard HMM formalism has a single output at each time step.

This model is able to learn sequences of outputs, and also to merge together two actions that it already knows. The merging happens by learning a transition between the state at end of one action and the state at the beginning of the next action. The model is not able to learn waypoint transitions, as it does not have input from context cells. All transitions must either be timed, or take place because of input from command cells.

The system follows the following rules:

$$\begin{aligned} P\left[s_i(t) = 1 | c_i(t-1) = 1\right] &= 1 \\ P\left[s_i(t) = 1 | c_i(t-1) = 0, s_j(t-1) = 1\right] &= p_{ji} \\ P\left[o_j(t) = 1 | t_j(t-1) = 1\right] &= 1 \\ P\left[o_j(t) = 1 | t_j(t-1) = 0, s_i(t-1) = 1\right] &= q_{ij}, \end{aligned}$$

where the transition probabilities p_{ji} and output probabilities q_{ij} are learned over time.

How do we learn the p_{ij}? Given a set of observations $s_1, \ldots, s_n$ over time, it is simply the observed conditional probability that s_i fired immediately before s_j, given the firing of s_i.

$$p_{ij} = \frac{\#\ t \text{ such that } s_j(t) = 1, s_i(t-1) = 1}{\#\ t \text{ such that } s_i(t-1) = 1}.$$

Note that this is not the maximum likelihood setting of parameters, since we don't look at whether a firing of s_i was caused by s_j firing or by c_i firing. This is important, because it allows us to learn transitions that we don't already know.

How do we learn the q_{ij}? Given a set of observations of $s_1, \ldots, s_n$ and $o_1, \ldots, o_m$ over time, it is simply the observed probability that s_i fired immediately before o_j.

$$q_{ij} = \frac{\#\ t \text{ such that } o_j(t) = 1, s_i(t-1) = 1}{\#\ t \text{ such that } s_i(t-1) = 1}.$$

Note that this is not the maximum likelihood setting of parameters, since we don't look at whether a firing of o_j was caused by s_i firing or by t_j firing. This is important, because it allows us to learn outputs that we don't already know.

2.4 Model 2: Hidden Markov Model with Context

This model is the same as above, except that our transition probabilities and output probabilities change based on context from the context mossy fibers. Let the context mossy fibers be denoted by $m_1, \ldots, m_\ell$. We still only allow one state cell to be active at a given time. We allow at most one command cell to be active at a given time. We allow any number of context cells to be active at a given time.

This model can be thought of as an average of multiple copies of the previous model, one for each context mossy fiber that is active. This allows us to learn waypoint transitions: if we have a context mossy fiber that is active when the leg is in a certain position, then we can learn to transition between the state we want to be in while walking before the leg reaches that position, and the state we want to be in after the leg reaches that position.

The system follows the following rules:

$$\begin{aligned}
P\left[s_i(t) = 1 | c_i(t-1) = 1\right] &= 1 \\
P\left[s_i(t) = 1 | c_i(t-1) = 0, s_j(t-1) = 1\right] &= \frac{1}{\#k : m_k(t-1) = 1} \sum_{k : m_k(t-1)=1} p_{kji} \\
P\left[o_j(t) = 1 | t_j(t-1) = 1\right] &= 1 \\
P\left[o_j(t) = 1 | t_j(t-1) = 0, s_i(t-1) = 1\right] &= \frac{1}{\#k : m_k(t-1) = 1} \sum_{k : m_k(t-1)=1} q_{kij},
\end{aligned}$$

where the transition probabilities p_{kji} and output probabilities q_{kij} are learned over time.

How do we learn the p_{kij}? Given a set of observations of $s_1, \ldots, s_n$ and $m_1, \ldots, m_\ell$ over time, it is simply the observed conditional probability that s_i fired immediately before s_j, given the firing of s_i and that the mossy fiber m_k was active at time $t-1$.

$$p_{kij} = \frac{\#\ t \text{ such that } s_j(t) = 1, s_i(t-1) = 1, m_k(t-1) = 1}{\#\ t \text{ such that } s_i(t-1) = 1, m_k(t-1) = 1}.$$

Note that this is not the maximum likelihood setting of parameters, since we don't look at whether a firing of s_i was caused by s_j firing or by c_i firing.

How do we learn the q_{kij}? Given a set of observations of $s_1, \ldots, s_n$, $m_1, \ldots, m_\ell$, and $o_1, \ldots, o_m$ over time, it is simply the observed conditional probability that s_i fired immediately before o_j, given the firing of s_i and that the mossy fiber m_k was active at time $t-1$.

$$q_{kij} = \frac{\#\ t \text{ such that } o_j(t) = 1, s_i(t-1) = 1, m_k(t-1) = 1}{\#\ t \text{ such that } s_i(t-1) = 1, m_k(t-1) = 1}.$$

Note that this is not the maximum likelihood setting of parameters, since we don't look at whether a firing of o_j was caused by s_i firing or by t_j firing.

2.4.1 Sparsity in the Context Mossy Fibers

This model works by essentially averaging together multiple copies of the previous model, one for each mossy fiber that is active. It is important to note that this version of the model cannot work well unless each context mossy fiber is sparse, that is, fires rarely and only in a narrow band of situations. If this is not the case, then we will be averaging together models that are not specific to the situation, and this will make it impossible for the transition function to be appropriate - we will wind up transitioning to a random state, essentially.

Fortunately, we have access to sparse context signals, since muscles contain nerves that fire predominantly when the relevant muscle is a particular length.

Look this up, I think they're called muscle spindles.

2.5 Model 3: Naive Bayes

We now relax the restriction that only one state cell in a microzone can be active at a particular time. The state of the system (not counting inputs) is therefore represented by which subset of the state cells are active at a given time step. This means that we have an exponential number of states that the system can be in.

We maintain the one-to-one correspondence between the command cells and the state cells, and we continue to assume that a state cell will definitely fire when its corresponding command cell fired in the previous time step.

In this model, we assume that each state cell makes a choice of whether to fire at each time step. It makes this choice based on its inputs: command cells, other state cells, and context cells. Let s be a state cell. We assume that it performs a relatively straightforward computation: for each of its inputs u,

This paragraph needs to be finished

s considers what fraction of the time it has previously fired when that cell has fired in the previous time step. Specifically, we compare the two conditional probabilities $P[u(t-1)=1|s(t)=1]$ and $P[u(t-1)=1|s(t)=0]$.

There are three kinds of deep nuclear cells. One projects to the inferior olive, and can be ignored for now. The other two are excitatory and inhibitory, respectively. We assume that there are state cells of both kinds. An excitatory state cell can only increase the probability that other cells fire, and thus an excitatory synapse from cell s_1 to cell s_2 can only be used to encode the information that $P[s_2(t)=1|s_1(t)=1] > P[s_2(t)=1|s_1=0]$ (and the strength of the synapse can encode the relative difference between the two). If the reverse holds (i.e., if the event $s_1=1$ should be taken as evidence that we shouldn't fire s_2), then the best the synapse can do is to erode completely, so that s_1 does not excite s_2 at all. Similarly, an inhibitory state cell can only decrease the probability that another state cell fires, and can only encode the information that $P[s_2=1|s_1=1] < P[s_2=1|s_1=0]$.

following paragraph has conditioning backwards?

In the rest of this section, we show that our rule for whether to fire a state cell makes sense.

We will assume that the probability that $s_i(t)=1$ depends only on the activity of $s_1(t-1),\ldots,s_n(t-1)$ and $c_i(t-1)$. If $c_i(t-1)=1$, then we assume that $s_i(t)=1$. For the time being, let us assume that $c_i(t-1)=0$. We then have (by assumption)

$$P[s_i(t)] = P[s_i(t)|s_1(t-1),\ldots,s_n(t-1)]$$

By Bayes' rule

$$\propto P[s_i]\cdot P[s_1(t-1),\ldots,s_n(t-1)|s_i(t)].$$

Here $P[s_i]$ (called the "prior") is the probability that represents how often s_i fires, independent of what inputs it receives.

A common assumption (the "naive Bayes" assumption) is that the $s_i(t-1)$ are independent given $s_i(t)$. In this case, we have

$$P[s_i(t)=\epsilon] = P[s_i=\epsilon]\cdot\prod_j P[s_j(t-1)|s_i(t)=\epsilon].$$

Dividing, we have

$$\frac{P[s_i(t)=1]}{P[s_i(t)=0]} = \frac{P[s_i=1]}{P[s_i=0]}\cdot\prod_j\frac{P[s_j(t-1)|s_i(t)=1]}{P[s_j(t-1)|s_i(t)=0]}$$

We can learn the parameters easily:

$$P[s_j(t-1)=1|s_i(t)=1] = \frac{\#\ t \text{ such that } s_j(t-1)=1, s_i(t)=1}{\#\ t \text{ such that } s_i(t)=1},$$

and analogously for the other parameters.

3 Psychophysical phenomena

When you scratch a dog behind the ears, they will sometimes start to move their leg in a way that looks like an abbreviated form of the motion they would use to scratch their own ear. We believe that this phenomenon is explained by the organization of the positive pathway - basically, the context cells are relaying the information that the ear is being scratched, which results in firing of the state cells that code for this activity (remember that, in the naive Bayes model, there is no distinction between commands and context), which results in the firing of output cells appropriate to scratching the ear, which results in the observed motion.

4 Predictions

Prediction 4.1. The synapse between the mossy fibers and the deep nuclear cell should have a classical Hebbian style of learning, in which the synapse is strengthened when one cell fires immediately before the other.

Prediction 4.2. Learning at a state cell - output cell synapse should be maximized when there is some delay between the firings. The delay should be on the order of 100 msec, analogously with LTD in the parallel fiber - Purkinje cell synapse.

5 Thoughts

- Stop commands in NB model
- Think about zero denominators in all of the bernoulli estimates
- No difference between commands and context in the NB model
- Why are we using probabilities? Why do we want a probabilistic model?
- Output probabilities should actually be higher than observed, since some outputs will be filtered out.
- Transition probabilities should also be higher than observed, or at least should be taken over a small time window. Otherwise it would be difficult to learn new skills.
- Positive pathway should learn from the negative pathway - rebound firing should be taken as a training signal