The (Markov) Hidden Connection: Quantifying Neuronal Spikes and Forest Fires

by Fabiana Ferracina and Jacob Pennington

MATH 583 - Assignment 3 - Spring 2020

Abstract

The probability that a neuron (or forest) will transition between three states — active, quiescent and refractory — is influenced both by its previous state and the current states of its neighbors. These state transitions represent neurons transmitting spikes via synapses or forest fires spreading to nearby areas. In the work that follows, we will initially establish a simple discrete-time Markov chain (DTMC) model that will unify the quantification of neuronal spikes and forest fires. Once suitable transition probabilities between the three states have been established as a starting point, we will hide the DTMC and look at how images can reveal those states by introducing a hidden Markov Model. The ultimate goal is to train the model on apriori labeled images — either of two-photon calcium data or satellite images of forests — and then use the model to predict states of the two systems based on unlabelled images.

1 Introduction

Two decades ago Jack Cowan, a neuroscience researcher at the University of Chicago, inspired by Per Bak's work on self-organized criticality, cited the dramatic similarities between the dynamics of cortical neurons and forest fires as the inspiration for a novel model of neural activity (Lanceta, 2003, Benayoun et al., 2010). The analogy between these disparate fields revolves around the characterization of both neurons and forested land as existing in one of three states: active, either a neuron spiking or a forest burning; refractory, a neuron that just spiked or a forest that just burned; or quiescent, a neuron that is ready to fire again or a forest with new un-burned growth.

The probability that a neuron (or forest) will transition between the three states is in turn influenced by the current state of its neighbors: neurons transmit spikes via synapses, and forest fires spread to nearby areas. Many other factors can also affect these transition probabilities. For example, the spread of a forest fire may be encouraged by strong winds or dry weather, and neural activity can be modulated by arousal or sensory context (David, 2018). These factors may trigger any of the states at a particular time step depending on their probability of occurrence and the stochastic dynamics of signal spread. Ultimately, a realistic model of either system should incorporate these additional factors. However, for this project we decided to focus on modeling the influence of neighboring neurons/forest plots.

Our goal is to use a unified hidden Markov model (HMM) to classify the behaviors exhibited by both neuronal activity and forest fire spread. In particular, we are interested in inferring the systems' switches between critical states. We aim to use satellite and remote sensed images of forest fires as well as images of neuronal activity obtained via *in-vivo* two-photon calcium imaging. The latter images would also need to be paired with simultaneously collected electrophysiological recordings for model validation. Figure 1 shows examples of the types of images that might be used to train and test our model.

2 Background and Motivation

There are several reasons why one should care about forest fires and brain activity. In the area of forest management, understanding how fires spread is of utmost importance for preventing forest fires and for efficiently stopping further spread of active fires. In neuroscience research, modeling neural responses evoked by visual stimuli like moving gratings or changes in network dynamics in response to anesthesia can help explain how the brain processes images or pain.

In both forest fire and neuroscience research, many deterministic models have been proposed to describe various aspects of these respective systems. Boychuk et al. (2009) specifically point out the unrealistic nature of such models to explain the phenomenon of fire spread that is intrinsically stochastic. In that same paper, they modeled a forest fire as a continuous time Markov chain on a lattice where each node represents a square area of forested land and can be in one of three familiar states: combustible fuel, burning fuel and burned out fuel, depending on covariates that are node-specific.

Agee (1998) explains that disturbance effects are due to pattern, that is, factors such as fuel amounts, size classes and landscape arrangement. A similar idea might be occurring in the brain, where the likelihood that a group of neurons fire together is linked by a power law to the number of neurons

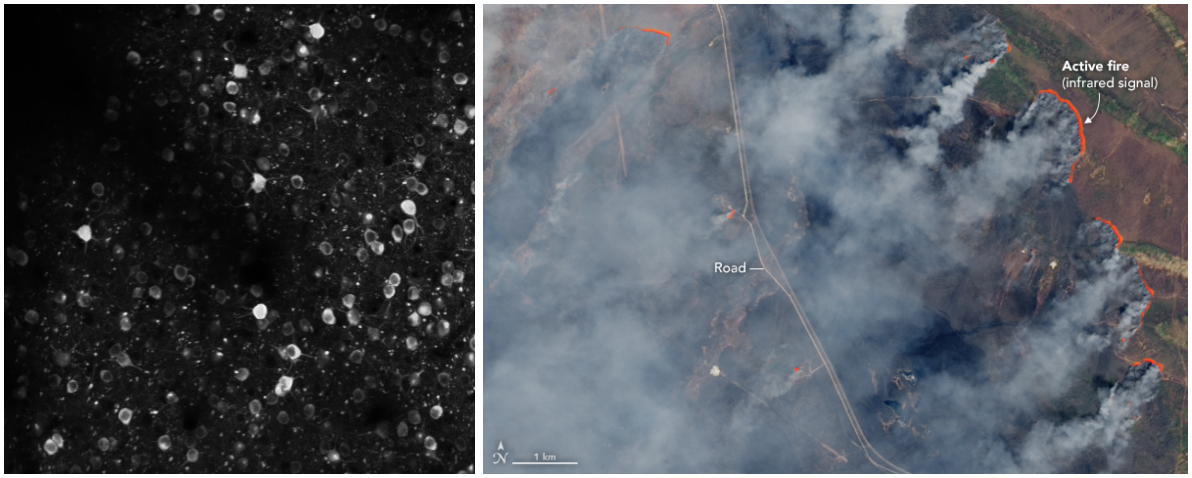

Figure 1: *[left]* Two-photon calcium imaging data from primary visual cortex, collected during the presentation of drifting gratings. From the Allen Institute Brain Observatory (Observatory, 2020). *[right]* Landsat 8 image of Sweltering, Smoky Fires in Siberia on May 7, 2018 (NASA, 2018).

Figure 2: A fire at Yellowstone National Park (nps.gov/articles/fires-yell.htm).

involved - a striking statistical rule explaining signal "avalanches" in the brain that manifest as bursts of spontaneous activity (Benayoun et al., 2010).

This unseeming connection between two disparate phenomena, neuronal spiking and forest fires, might be hidden in the dynamics of the tiny particles that compose matter. Over thirty years ago, Bak et al. (1987) came up with the idea of dynamical systems naturally evolving to a critical state. They called this process, in which the behavior of extended dissipative systems assumes a critical steady state independent of the initial state and without tuning of parameters to a special value, *self-organized criticality*.

An instance of criticality as applied to forest fire models is given by Albano (1994). Here the author suggests a lattice with nodes that are either empty, containing a green tree or containing a burning tree. At each time step, the following rules apply to the nodes: (i) a burning tree becomes an empty site; (ii) trees grow with probability p from empty sites; (iii) a green tree becomes a burning tree with probability $(1 - g)$ if at least one neighbor is burning; and finally (iv) a green tree becomes a burning tree with probability $f << 1$ if no neighbor is burning. An alternative model may be constructed if we consider neurons firing instead of trees burning.

Yet another connection between brain activity and fire seems to be the idea of directed percolation. Von Niessen and Blumen (1986) explored this idea in forest fires using similar model states to those described above. Cowan et al. (2016) explored a similar modelling framework but applied to neocortical dynamics.

Cowan in the same work with Benayoun et al. (2010) derived the master equation governing the stochastic evolution of the states of a network of k active excitatory and l active inhibitory neuron clusters, an "ancestral" dynamic to fire spread that has not yet been borrowed to explain and predict

the spread of fire in a two-dimensional lattices.

In the forest fire literature, there are several models for fire detection from images. Wang et al. (2011) use flame flicker frequency to discriminate candidate fire regions in images. They use a HMM based on spatial and temporal factors to determine the flame flicker process. They create a luminance map and run the HMM to make the final decision. Teng et al. (2010) use a HMM to create a fire detection system off of video surveillance. They analyse pixel data to find fire characteristics related to the RGB scale and the frequency domain. Based on these values they can determine the states of pixels. In all these works the focus is usually on providing the ability for autonomous systems to detect fire, but using these techniques to get at the dynamics of fire spread and ecological change due to fire is a novel idea we wish to explore.

The literature of Markovian approaches to neural modeling is also rich. Makarenko et al. (1997) used a Markov random field model and considered neuron activity for every time t on a rectangular lattice, with each rectangle described by a Gaussian random variable. The probability of the lattice site (i, j) being in some particular state depends only on the states of some neighborhood of the site (i, j). Vogelstein et al. (2009) tackled he difficult problem of inferring the spike trains and calcium concentrations from a fluorescence signal by starting with spike train data and predicting the images and then inverting their model to go from image to spike trains. Camproux et al. (1996), as well as Delescluse and Pouzat (2006) independently presented a hidden Markov model for a single neuron's electrical activity using successive occurrence of spikes and interspike intervals.

In our research we wish to use two-photon calcium imaging data to infer spike trains. Calcium ions serve as versatile intracellular signalling molecules, whose influx at presynaptic terminals triggers exocytosis of vesicles containing neurotransmitters. In quiescence, neurons have a calcium concentration of about 50-100nM. During activity these levels are 10 to 100 times higher (Grienberger and Konnerth, 2012). This dramatic shift can be translated into a recording of neural activity by transfecting the target neural population with a flourescent calcium indicator. An increase in flourescence can then be associated with spiking activity in the targeted neurons. However, this association is not perfect: determining accurate spike times from calcium flourescence data is an active area of research (Deneux et al., 2016). We believe using a HMM to infer spikes from calcium data may be a fruitfuil approach.

3 The Stochastic Model: DTMC and HMM

3.1 Laying the DTMC Foundations

We initially construct a simple discrete-time Markov chain before expanding it to a hidden Markov model. Let X_n be a discrete random variable defined on the finite state space $\{a, r, q\}$ (for active, refractory, quiescent) such that the probability to transition from any state i to a new state j is $P(X_{n+1} = j | X_n = i) = p_{ji}$. For now we will consider the DTMC to be time homogeneous, however a later modification that may be more realistic for our application would be for the transitions that are not triggered by particle interactions or criticality to decay over time.

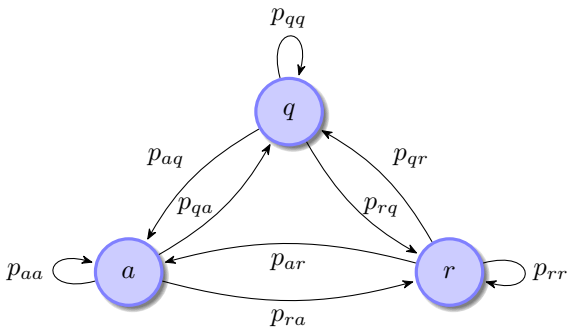

Figure 3: A DTMC for the three possible states of neuron/tree clusters. At each time step, the system transitions from its current state $i \in \{a, r, q\}$ to a new state j in the same set of states by travelling along an edge with a probability of p_{ji}.

Given the Markov chain described in 3, we have the following transition matrix P:

$$P = \begin{pmatrix} p_{aa} & p_{ar} & p_{aq} \\ p_{ra} & p_{rr} & p_{rq} \\ p_{qa} & p_{qr} & p_{qq} \end{pmatrix} \tag{1}$$

Although simple, the DTMC described above can serve as a foundation for more complex models, such as hidden Markov models (HMM), which is what we wish to explore in this work. An HMM is a discrete-time finite state homogeneous Markov chain observed through a discrete-time channel characterized by a finite set of transition densities (such as Gaussian or Poisson) that are indexed by the states of the Markov chain (Ephraim and Merhav, 2002). The underlying process is not observable and referred to as the regime. The second process is a sequence of independent random variables that are conditional on the underlying Markov chain. The value obtained from the distribution of each random variable depends on the Markov chain at that time. These random variables are referred to as the observation sequence (Ephraim and Merhav, 2002).

Our hidden Markov model will take the Markov chain introduced above and place it behind a "curtain" so that we only observe noisy states that are functions of each of the original states. Thus observations $\{\mathcal{O}_n\}_{n=0}^{\infty}$ is a process (with a distribution yet to be determined) that is dependent on the respective underlying state for the particular time step of the original DTMC process, i.e. $\{X_n\}_{n=0}^{\infty}$. Realistically the observation sequence is time bounded by some time length T. Note that the number of unique symbols from the observation sequence need not be the same as the number of unique states occurring in the hidden Markov process, as noise and other phenomena might alter observed symbols pertaining to the same hidden state. Ideally, the distribution that connects observed to hidden states would capture these deviations with a high level of confidence.

By expanding (or rather hiding) the DTMC from 3 we obtain the following Markov model from 4, where for each state $i \in \{a, r, q\}$ (the state-space as before), we have $\mathcal{O}_i$ an observation of state i through a filter and/or with some noise.

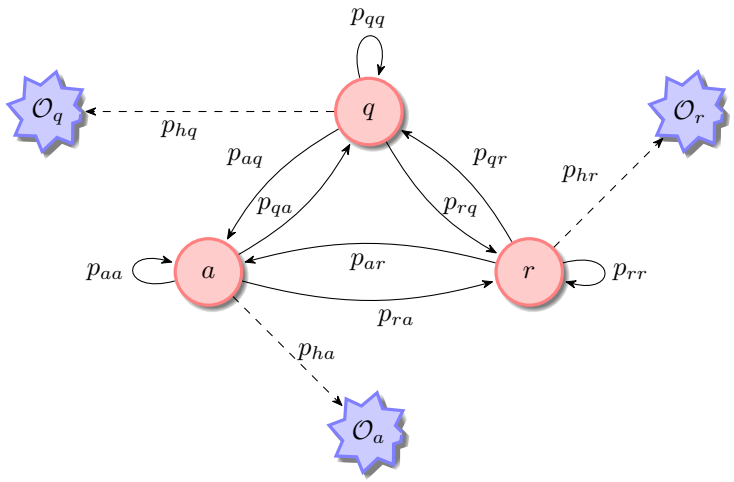

Figure 4: Hidden Markov Model for the three possible states of neuron/tree clusters.

3.2 Hidden Markov Model Specification

In this section we completely specify the HMM while noting some implementation details for our current model. Let

> T = the time length of the observation sequence. For our initial implementation we will observe 12 months of NASA's MODIS Images (see the Model Implementation section 4 for further details), however this can change based on the data and will be modeled here as T.
>
> $N = |S| = 3$ be the number of states in the model.
>
> $M = |V| =$ the number of observation symbols. As with T, this number will depend on the data. For the initial implementation, we will use $M = 2$.

$S = \{a, r, q\}$ be the distinct states in the hidden DTMC.

$V = \{0, 1, \ldots, M-1\}$ be the distinct symbols that can be possibly observed. In our initial implementation, we only consider two possible symbols: 0 and 1.

$P =$ the $N \times N$ matrix of state transition probabilities in 1.

$B =$ the $N \times M$ matrix of observation probabilities, that is, the probability of observing symbol $k \in V$ given the hidden state $j \in S$.

$\pi =$ a 3×1 vector of initial state probabilities.

$\mathcal{O} = (\mathcal{O}_0, \mathcal{O}_1, \ldots, \mathcal{O}_{T-1})$ be the observation sequence. In the model implementation, this sequence is of length 12.

Therefore, a hidden Markov model λ is the 3-tuple (P, B, π). As in the DTMC and in order to be consistent with the transition matrix description provided in class $P = \{p_{ji}\}$ is $N \times N$ and column stochastic, with $p_{ji} = P(\text{state } j \text{ at } t+1 \mid \text{state } i \text{ at } t)$. In the model implementation however, we find it easier to transpose the transition matrix making it effectively row stochastic. The matrix $B = \{b_j(k)\}$ is $N \times M$ and row stochastic, with $b_j(k) = P(\text{observing } k \text{ at } t \mid \text{state } j \text{ at } t)$. Both matrices, P and B have probabilities that are time homogeneous.

Thus we must solve the following problem: given an observation sequence $\{\mathcal{O}_n\}_{n=0}^{T}$, the number of states we wish to describe $N = 3$, and a number M of observed symbols (from our image data), we wish to find the model $\lambda = (P, B, \pi)$ that maximizes the likelihood of the given observation sequence. For this purpose, we will use the Baum-Welch reestimation formulas described in Rabiner and Juang (1986).

Although the number of states N and the number of symbols that can be observed M are fixed (thus making the dimensions of (P, B, π) fixed), we can still attempt to find the best model λ that maximizes the likelihood of observing $\{\mathcal{O}_n\}_{n=0}^{T}$. As in Rabiner and Juang (1986) we define "di-gammas"

$$\gamma_t(i, j) = P(x_t = i, x_{t+1} = j \mid \mathcal{O}, \lambda), \ t = 0, 1, \ldots, T-2, \ i, j \in \{a, r, q\}$$

which are the probabilities of being at state i at time t and transitioning to state j at time $t+1$. The probability described by $\gamma_t(i, j)$ can be re-written as a conditional probability ratio that can be further manipulated algebraically to obtain sums efficiently solved by the forward and backward algorithms as follows.

The forward algorithm solves the probability of observing a sequence of symbols given a model λ, or $P(\mathcal{O}|\lambda)$. We can see how this probability is useful for scoring and re-estimating the initial guess for the model λ. Following Stamp (2018), we have

$$P(\mathcal{O}|\lambda) = \sum_X P(\mathcal{O}, X \mid \lambda), \text{where } X \text{ denotes a state sequence.}$$

Thus the sum is over all possible state sequences,

$$\Rightarrow P(\mathcal{O}|\lambda) = \sum_X P(\mathcal{O} \mid X, \lambda) P(X \mid \lambda).$$

Clearly, we cannot compute the above directly in an efficient manner due to the exponential number of possible sequences of states matched to possible observations. The clever approach used is called the forward algorithm, or α-pass. We define the probability of a partial observation sequence up to time t, where the underlying DTMC is in state i at time t, given the model λ. This probability can be recursively computed as follows:

$$P(\mathcal{O} \mid \lambda) = \sum_{i \in \{a,r,q\}} \alpha_{T-1}(i),$$

$$\text{where for } t = 1, 2, \ldots, T-1 \text{ and } i \in \{a, r, q\},$$

$$\alpha_t(i) = \left[\sum_{j \in \{a,r,q\}} \alpha_{t-1}(j) p_{ij} \right] b_i(\mathcal{O}_t) \text{ and } \alpha_0(i) = \pi_i b_i(\mathcal{O}_0).$$

The backward algorithm is very similar to the forward algorithm, with the difference being that it goes backward in time. The base case for the recursion is defined as 1 for the last time step observed, and for $t = T-2, T-3, \ldots, 0$ and $i \in \{a, r, q\}$, the recursive step is defined as

$$\beta_t(i) = \sum_{j \in \{a,r,q\}} p_{ji} b_j(\mathcal{O}_{t+1}) \beta_{t+1}(j).$$

Thus returning to our formulation for the "di-gammas", we can re-write them in terms of α and β as defined above

$$\gamma_t(i,j) = \frac{\alpha_t(i)p_{ji}b_j(\mathcal{O}_{t+1})\beta_{t+1}(j)}{P(\mathcal{O} \mid \lambda)}.$$

Thus we accomplish the same probability from the originally defined $\gamma_t(i,j)$ through computing the probability to reach state i by marching forward in time jointly with the probability of reaching state j by marching backward in time. By also defining the total probability of transitioning from i to any state, i.e $\gamma_t(i) = \sum_{j \in \{a,r,q\}} \gamma_t(i,j)$ for $i \in \{a,r,q\}$, and using this probability along with $\gamma_t(i,j)$, we can perform one estimate of the model $\lambda = (P, B, \pi)$ by taking the appropriate ratios.

We estimate the initial probability π_i for state i with $\gamma_0(i)$; the transition probabilities p_{ji} with $\sum_{t=0}^{T-2} \gamma_t(i,j) \Big/ \sum_{t=0}^{T-2} \gamma_t(i)$, which can be seen as the ratio of the expected number of transitions from i to j and the expected number of transitions from i to any state; and finally the observation k given state j probabilities $b_j(k)$ with $\sum_{\mathcal{O}_t=k} \gamma_t(j) \Big/ \sum_{t=0}^{T-1} \gamma_t(j)$, which can be seen as the ratio of the expectation of being in state j by observing k and the expectation of being in state j regardless of the observation (Stamp, 2018).

This estimation is repeated for as long as the probability of the observation sequence given the current model estimate λ increases. When the increase is marginal or non-existent, the local maximum has been reached and the algorithm can terminate. In the following section (4), we will provide the implementation of this algorithmic solution and apply it to satellite imagery in order to detect fire.

4 Model Implementation

4.1 DTMC

Our first exploration of the model involved implementing a simple DTMC with the three states described above: active, refractory, and quiescent. The code below implements the basic machinery of the DTMC in Python3. Some documentation and details are omitted for brevity, but see our github repository for the full source code.

```
class Node():
    def __init__(self, initial_distribution, transition_matrix):
        self.initial_distribution = initial_distribution
        self.transition_matrix = transition_matrix
        self.state = self._sample_state(self.initial_distribution)

    def transition(self):
        distribution = self.transition_matrix[:, self.state]
        self.state = self._sample_state(distribution)

    def _sample_state(self, distribution):
        arq = np.random.sample(1)
        if arq <= distribution[0]:
            state = 0
        elif arq <= distribution[0] + distribution[1]:
            state = 1
        else:
            state = 2

        return state
```

To apply the model to both neurons and forest fires, we simply used different default values for the transition probabilities. In order to test our model and obtain some preliminary results, we initialize the distribution and the transition matrix probabilities to some intuitive guesses. Note that initially both neuronal activity and forest fires are in the quiescent state, that is, they are ready to spike or burn, but still inactive. Also, note that we allow for the possibility of transitions between all states, including quiescent to refractory and refractory to active. Although, these transitions are very improbable, they are not impossible and allows us to better mimic natural stochastic phenomena with only three states.

These phenomena can be thought as a hard to detect gradient between the states or a unrelated event affecting neurons or trees (e.g. loss of a quiescent neuron or tree would make its location a refractory neuron or tree).

```
initial_distribution = np.array([0.005, 0.005, 0.990])
forest_matrix = np.array([[0.500, 0.000, 0.090],
                          [0.250, 0.900, 0.010],
                          [0.250, 0.090, 0.90]])
neuron_transition = np.array([[0.050, 0.010, 0.045],
                              [0.900, 0.190, 0.005],
                              [0.050, 0.800, 0.950]])
```

Where the 0th, 1st and 2nd indices correspond to the active, refractory, and quiescent states, respectively. Initially, these values were chosen by hand to represent our prior expectations of what the dynamics should look like. For example, we expect forest fires to have a moderate probability of remaining in an active (burning) state for several time steps while neurons should almost always transition from an active state to a refractory state in one step.

4.2 HMM

Our next goal was to be able to refine the model parameters based on real data following a HMM framework. The first step in this process was to translate some available data into a HMM-friendly format: namely, a sequence whose entries are drawn from some finite number of symbols. We decided to apply the model to forestry data first. For this preliminary implementation, we used NASA's Moderate Resolution Imaging Spectroradiometer (MODIS) Images in which each pixel reports whether a fire took place within a 500m square plot of land at some time during a 30 day period. The indication of fire is encoded in a single data layer using the ordinal day of the calendar year (in our first implementation the year is 2018); values indicating unburned land, missing data and water are also assigned (Giglio et al., 2015).

On the one hand, this labeling made the estimation of hidden states partially redundant - the labels on the images already tell us whether a plot is in the "active" state or not (but cannot distinguish between refractory versus quiescent). However, this also made the images ideal for initial sanity checks. For example, if our model estimation process works as intended then a hidden "active" state should correspond to an observed "fire" state with very high probability. The steps taken to process the images into labels were as follows:

1. Treat all pixels labeled as either "water" or "not enough data" as equivalent to "no burn."
2. Collapse the color axis by summing, so that all days of the month on which fires occurred are treated roughly equally.
3. Cut out a neighborhood of size r extending out from a point of interest, resulting in a square array with shape $(2r + 1, 2r + 1)$.
4. Compute the mean over the entire neighborhood, representing the density of fires in the surrounding area over the month.
5. Set a threshold equal to the 50th percentile of the means (over all images). Images with a processed mean that is less than or equal to this threshold are labeled "no fire", while all other images are labeled "fire."

The second step toward refining the model was to incorporate the state-conditional probability distribution of the two possible observations for each of the three state variables. Following the simplicity of the current labeling scheme, we assigned the probabilities as follows:

```
observation_distributions = np.array([[0.10, 0.90], # active
                                      [0.90, 0.10], # refractory
                                      [0.90, 0.10]]) # quiescent
```

Where the first column is for an observation of "no fire" and the second column is for an observation of "fire." These probabilities were based on our initial guess that NASA's labeling correctly detected the absence or presence of a fire 90% of the time.

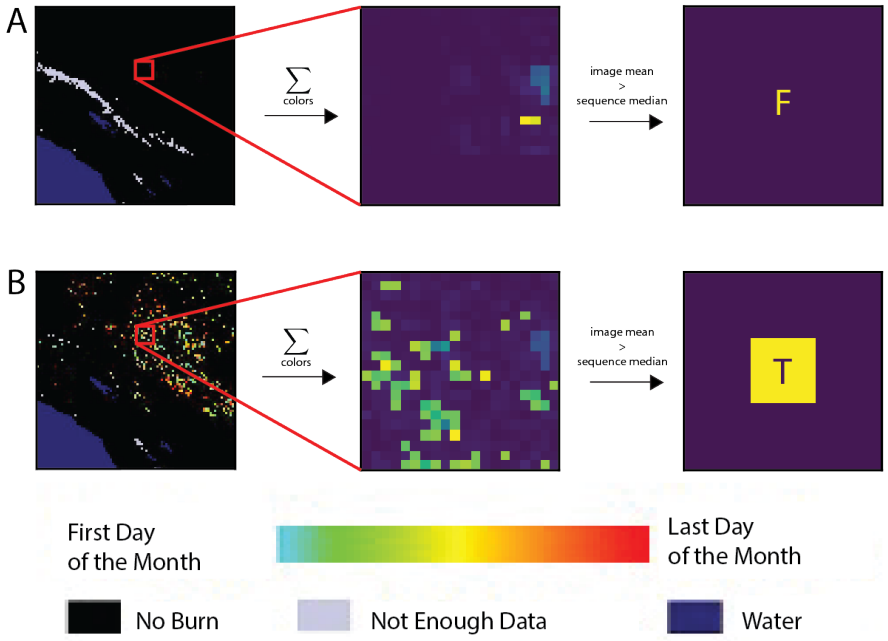

Figure 5: Example NASA MODIS images and their processed counterparts. (A) An example that got labeled as "no fire." (B) An example that was labeled as "fire."

Our next step was to assign any given set of model parameters a score so that the best possible score can be searched for. For this purpose, we followed a standard approach: to score models according to the probability of a sequence of observations given the set of model parameters, where this probability is computed using the "forward-backward" algorithm (Rabiner and Juang, 1986, Stamp, 2018). As the name suggests, this algorithm can be implemented in either a "forward" or "backward" manner to achieve the same result. We present both versions since both are necessary in a subsequent computation.

Note that the implemented algorithms in this section uses the transpose of the transition matrices described above. This changes the matrices to be row stochastic instead of column stochastic and changes the interpretation of the order of indices (p_{ij} is the one-step transition probability from i to j instead of from j to i). This choice was made to keep the notation consistent with the published descriptions of the algorithms that we referenced (Rabiner and Juang, 1986, Stamp, 2018), which helped maintain our sanity while debugging.

The forward portion of the algorithm recursively computes the probability, $\alpha_t(i)$, of a sequence of observations truncated at time t, and that the model is in state i at time t, given the set of model parameters. This relies on assigning $\alpha_0(i)$ to be the product of the initial probability of state i and the probability of the first observation in the sequence given that the model is in state i for each $i \in N$. Each other $a_t(i)$, $t \leq T-1$, then depends on the probabilities computed at the previous time step (hence the "forward" terminology). The probability of the full sequence of observations given the model parameters is then the sum of $\alpha_{T-1}(i)$ for each i (Stamp, 2018).

```
    def _forward(tmax):
        alphas = np.empty((tmax, n_states))
        for i in range(n_states):
            a = initial_state_distribution[i] * observation_distributions[i,
                observation_sequence[0]]
            alphas[0][i] = a

        t = 1
        while t < tmax:
            for i in range(n_states):
                sum_i = 0
                for j in range(n_states):
                    sum_i += alphas[t-1][j]*transition_matrix[j, i]
                a = sum_i * observation_distributions[i, observation_sequence[t]]
```

```
            alphas[t, i] = a
        t += 1

    return alphas
```

The backward portion is similar, and computes the same final probability. However, the process begins at $T-1$ with $\beta_{T-1}(i) = 1$ for each i. The computation of each other $\beta_t i$ then depends on $\beta_{t+1}(i)$ while t decreases (hence the "backward" terminology) (Stamp, 2018).

```
def _backward(tmax):
    betas = np.empty((tmax, n_states))
    betas[-1, :] = 1
    t = tmax - 2
    while t >= 0:
        for i in range(n_states):
            b = 0
            for j in range(n_states):
                a_ij = transition_matrix[i, j]
                beta_j = betas[t+1, j]
                b_j = observation_distributions[j, observation_sequence[t+1]]
                b += a_ij * beta_j * b_j
            betas[t, i] = b
        t -= 1

    return betas
```

With a score in hand, our next step was to determine how to adjust the model parameters to maximize that score. We used a standard iterative approach for this that takes advantage of the probabilities computed above, referred to as the Baum-Welch algorithm (see Rabiner and Juang (1986), Stamp (2018) for more details).

```
model = HMM()
def fit_baum_welch(self, iteration_count=100):
    time_steps = len(observation_sequence)
    for k in range(iteration_count):
        di_gammas = np.zeros((time_steps, n_states, n_states))
        gammas = np.zeros((time_steps, n_states, n_states))
        alphas = self._forward(time_steps)
        betas = self._backward(time_steps)
        pr = np.sum(alphas[-1])
        t = 0
        while t < time_steps-1:
            for i in range(n_states):
                alpha_i = alphas[t, i]
                for j in range(n_states):
                    beta_j = betas[t+1, j]
                    b_j = observation_distributions[j, observation_sequence[t+1]]
                    di_gamma_ij = (alphas[t, i] * transition_matrix[i, j] * b_j * betas[t+1,
                        j])/pr
                    di_gammas[t, i, j] = di_gamma_ij
                gammas[t, i, :] = np.sum(di_gammas[t, i, :])
            t += 1

        model.initial_state_distribution = gammas[0, :, 0]
        transition_matrix = np.sum(di_gammas, axis=0) / np.sum(gammas, axis=0)
        model.state_transition = np.transpose(transition_matrix)

        # update observation distributions
        bs = np.empty((n_states, n_observations))
        for j in range(n_states):
            for k in range(n_observations):
                numer = np.sum(gammas[:, j, 0] * (observation_sequence == k).astype(np.int))
                denom = np.sum(gammas[:, j, 0], axis=0)
```

```
                bs[j, k] = numer/denom
            model.observation_distributions = bs

        return
```

5 Preliminary Results and Discussion

We began by simulating a single "node" (representing either a neuron or a plot of forest) as a standard DTMC, recording the state of the node at each time step. Figure 6 shows the results of 100-step simulations for one neuron and one forest plot using the initial distributions and transition probabilities described above. The simulations reflected dynamics that looked promisingly like what we might expect

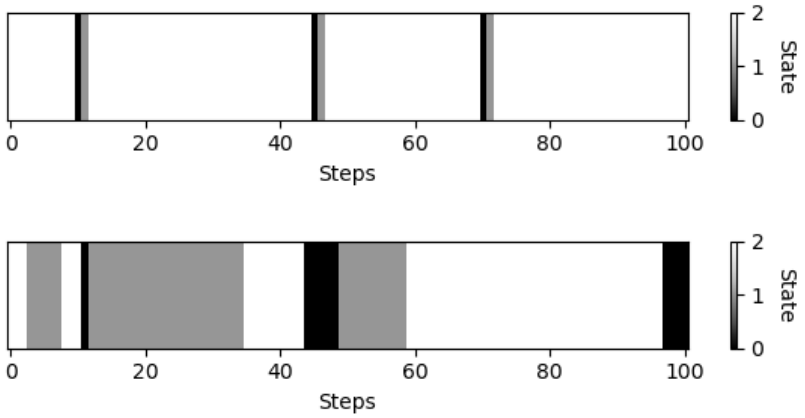

Figure 6: DTMC simulations using two sets of transition probabilities. States 0, 1, and 2 represent quiescent, refractory, and active, respectively. *[top]* Simulation with transition probabilities intended to emulate a single neuron. *[bottom]* As above, but to emulate a plot of forest.

for spiking neurons and forest fires. For example, the top simulation represents a neuron with sparse activity, as it stays in the quiescent state for the majority of the time. After transitioning to an active state, it immediately switches to refractory and then back to quiescent, representing a single spike. Meanwhile, the bottom simulation resembles a plot of forest intermittently burning down and then recovering. The typical case is switching from quiescent to active and then to refractory after some number of steps, and then staying in the refractory state for a prolonged period. This represents the forest catching fire, exhausting all of its fuel, and then requiring many years to recover. We also left in the possibility of a forest transitioning immediately from quiescent to refractory (an example of which is seen after a few steps in the simulation). This could represent an exhaustion of the forest's fuel for reasons other than a fire, like logging.

Next, we trained the forest model's parameters on a small NASA MODIS Image dataset using our HMM implementation. The Node implementation was left unchanged for simulating hidden states, while the new code described in 4 was used to optimize the transition matrix, initial state distribution, and state-conditional observation distributions. For the preliminary twelve-image dataset we used (representing one year of one-month recordings from the same geographical region), we obtained the following optimized parameters (approximated):

```
initial_state_distribution = np.array([0.0, 0.0, 1.0])
transition_matrix = np.array([[0.667, 0.000, 0.200],
                              [0.333, 0.667, 0.000],
                              [0.000, 0.333, 0.800]])
observation_distributions = np.array([[0.000, 1.000],
                                      [0.000, 1.000],
                                      [1.000, 0.000]])
```

Using these parameters, we generated ten single-node simulations each over 12 time steps: (7).

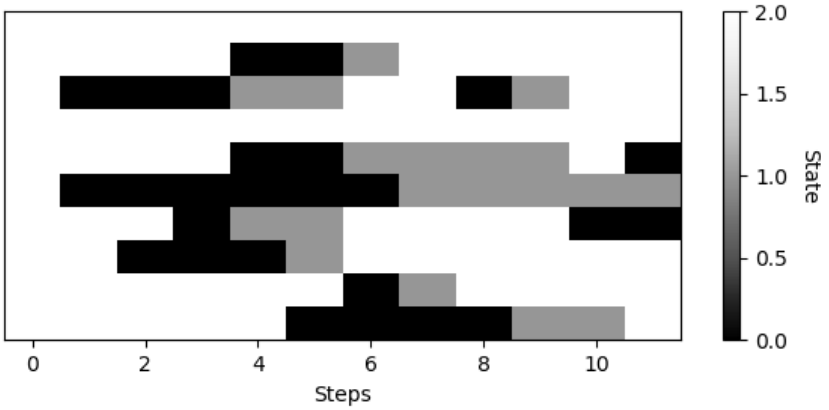

Figure 7: Ten simulations from one DTMC node using parameters fit to a sample NASA MODIS Image set

6 Work in Progress and Future Work

Our next steps fall into two categories: fitting the model to data and simplifying the spatial component. To approach both of these issues, our plan is to use a hidden Markov model (HMM) framework. In this implementation, we would treat images as observed states and use them to infer the state (quiescent, refractory, or active) of a neuron or forest. In the former case, we plan to use this approach to infer spiking activity from two-photo calcium data. In the latter case, we would infer the presence of forest fires based on satellite images. Since HMMs are well-studied, many optimization algorithms have already been developed that we can use as a starting point (Rabiner and Juang, 1986).

However, the multi-node model that we described in a previous draft (see Math 583 Assignment 2 document) likely would not fit into a classical HMM framework: the proposed mechanism for the influence of neighboring nodes on transition probabilities is difficult to analyze from a mathematical perspective. Instead, we plan to incorporate spatial information by fitting a model for each node individually but using two-dimensional observations.

References

Agee, J. K. (1998). The landscape ecology of western forest fire regimes. *Northwest Science*, 72(17):24–34.

Albano, E. V. (1994). Critical behaviour of a forest fire model with immune trees. *Journal of Physics A: Mathematical and General*, 27(23):L881.

Bak, P., Tang, C., and Wiesenfeld, K. (1987). Self-organized criticality: An explanation of the 1/f noise. *Physical review letters*, 59(4):381.

Benayoun, M., Cowan, J. D., van Drongelen, W., and Wallace, E. (2010). Avalanches in a stochastic model of spiking neurons. *PLoS computational biology*, 6(7).

Boychuk, D., Braun, W. J., Kulperger, R. J., Krougly, Z. L., and Stanford, D. A. (2009). A stochastic forest fire growth model. *Environmental and Ecological Statistics*, 16(2):133–151.

Camproux, A.-C., Saunier, F., Chouvet, G., Thalabard, J.-C., and Thomas, G. (1996). A hidden markov model approach to neuron firing patterns. *Biophysical journal*, 71(5):2404–2412.

Cowan, J. D., Neuman, J., and van Drongelen, W. (2016). Wilson–cowan equations for neocortical dynamics. *The Journal of Mathematical Neuroscience*, 6(1):1–24.

David, S. V. (2018). Incorporating behavioral and sensory context into spectro-temporal models of auditory encoding. *Hearing Research*, 360:107–123.

Delescluse, M. and Pouzat, C. (2006). Efficient spike-sorting of multi-state neurons using inter-spike intervals information. *Journal of neuroscience methods*, 150(1):16–29.

Deneux, T., Kaszas, A., Szalay, G., Katona, G., Lakner, T., Grinvald, A., Rozsa, B., and Vanzetta, I. (2016). Accurate spike estimation from noisy calcium signals for ultrafast three-dimensional imaging of large neuronal populations in vivo. *Nature Communications*, 7.

Ephraim, Y. and Merhav, N. (2002). Hidden markov processes. *IEEE Transactions on information theory*, 48(6):1518–1569.

Giglio, L., Justice, C., Boschetti, L., and Roy, D. (2015). Mcd64a1 modis/terra+ aqua burned area monthly l3 global 500 m sin grid v006 [data set]. *NASA EOSDIS Land Processes DAAC: Sioux Falls, SD, USA*.

Grienberger, C. and Konnerth, A. (2012). Imaging calcium in neurons. *Neuron*, 73(5):862–885.

Lanceta, J. (2003). Brain scientist models neural networks on forest fire patterns. chicagomaroon.com/2003/02/25/brain-scientist-models-neural-networks-on-forest-fire-patterns/. Accessed: 2020-02-19.

Makarenko, V. I., Welsh, J. P., Lang, E. J., and Llinas, R. (1997). A new approach to the analysis of multidimensional neuronal activity: markov random fields. *Neural Networks*, 10:785–789.

NASA (2018). Landsat 8 image gallery. landsat.visibleearth.nasa.gov/view.php?id=92165. Accessed: 2020-02-19.

Observatory, A. I. B. (2020). Allen brain atlas, experiment id 511507650. observatory.brain-map.org/visualcoding/search/overview. Accessed: 2020-02-19.

Rabiner, L. and Juang, B. (1986). An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16.

Stamp, M. (2018). A revealing introduction to hidden markov models.

Teng, Z., Kim, J.-H., and Kang, D.-J. (2010). Fire detection based on hidden markov models. *International Journal of Control, Automation and Systems*, 8(4):822–830.

Vogelstein, J. T., Watson, B. O., Packer, A. M., Yuste, R., Jedynak, B., and Paninski, L. (2009). Spike inference from calcium imaging using sequential monte carlo methods. *Biophysical journal*, 97(2):636–655.

Von Niessen, W. and Blumen, A. (1986). Dynamics of forest fires as a directed percolation model. *Journal of Physics A: Mathematical and General*, 19(5):L289.

Wang, L., Ye, M., Ding, J., and Zhu, Y. (2011). Hybrid fire detection using hidden markov model and luminance map. *Computers & Electrical Engineering*, 37(6):905–915.