1 Hidden Markov Models

A **Hidden Markov Model** (HMM) is defined by an underlying system that is a Markov process with states that cannot be observed, but gives outputs that are dependent on the hidden state of the system. If we let random variable X_i represent the i-th state of the system, and Y_i represent the i-th output of the system, we have:

$$X_1 \sim \mu(x_1) \text{ and } X_n|(X_{n-1} = x_{n-1}) \sim f(x_n|x_{n-1}) \tag{1}$$

$$Y_n|(X_n = x_n) \sim g(y_n|x_n) \tag{2}$$

where X_1 is defined by some initial conditions $\mu(x_1)$.

2 Particle Filters

Basically, the big idea is to use our observations of the system to determine the hidden state of the system. That is to say, we want to figure out $p(x_n|y_{1:n})$. Recognizing that this is a marginal is useful because then we can just say:

$$p(x_n|y_{1:n}) = \int p(x_{1:n}|y_{1:n})dx_{1:n-1} \tag{3}$$

Setting this as a marginal of $p(x_{1:n}|y_{1:n})$ is helpful because we can refactor this conditional distribution to see how it comes from our prior distributions of the hidden state and observations. With some refactoring and using our initial HMM conditions we can show:

$$p(x_n|y_{1:n}) = \int \frac{\mu(x_1)\prod_{i=2}^{n} f(x_1|x_{i-1})\prod_{i=1}^{n} g(y_i|x_i)}{\int \mu(x_1)\prod_{i=2}^{n} f(x_1|x_{i-1})\prod_{i=1}^{n} g(y_i|x_i)dx_{1:n}} dx_{1:n-1} \tag{4}$$

This looks really messy, but it's helpful to see that this is just integration of functions we already know f and g. In other words, we already have all the information we need to arrive at the answer.

The issue now is that this is sometimes intractable. If functions are nice, like guassians or linear distributions, this is integrable and we can arrive at a solution analytically. However, this is not always the case, which is why we sometimes need to derive an answer from numeric approximations.

3 Kalman Filter

So let's say you're really lucky and your initial distribution is something nice. This allows us to get a solution analytically through a method called the Kalman Filter. We can reformulate our goal as:

$$\text{Prediction step: } p(x_n|y_{1:n}) = \int f(x_n|x_{1:n-1})p(x_{n-1}|y_{1:n-1})dx_{n-1} \tag{5}$$

$$\text{Update step: } p(x_n|y_{1:n}) \propto g(y_n|x_n)p(x_{n-1}|y_{1:n-1}) \tag{6}$$

Essentially, we are now assuming that $p(x_{n-1}|y_{1:n-1})$ is known due to recursion. To see how this relates to our earlier definition of $p(x_n|y_{1:n})$ in Eq. (4), note that this recursion is what gives us the numerator, while the normalization we do in the update step essentially gives us the denominator.

4 Sequential Monte Carlo Methods

So what if our distribution isn't nice? To find the our desired distribution we will use an approximation:

$$\text{Prediction step: } \pi_{x_{1:t}|y_{1:t-1}}(x_{1:t}|y_{1:t-1}) = f(x_t|x_{t-1})\pi_{x_{1:t-1}|y_{1:t-1}}(x_{1:t-1}|y_{1:t-1}) \tag{7}$$

$$\text{Update step: } \pi_{x_{1:t}|y_{1:t}}(x_{1:t}|y_{1:t}) \propto g(y_t|x_t)\pi_{x_{1:t}|y_{1:t-1}}(x_{1:t}|y_{1:t-1}) \tag{8}$$

5 SIS

1. *Initialization:*

   For $i = 1, \dots N$:

   Sample $x_0^{(i)} \sim \mu(x_0)$
   Assign weights $\widetilde{w}_0^{(i)} = g(y_0|x_0^{(i)})$
   Normalize weights $w_0^{(i)} = \frac{\widetilde{w}_0^{(i)}}{\sum_{i=1}^n \widetilde{w}_0^{(i)}}$

2. *Importance Sampling:*

   For $t = 1, \dots, T$:

   Sample $x_t^{(i)} \sim f(x_t|x_{t-1}^i)$
   Assign weights $\widetilde{w}_t^{(i)} = g(y_t|x_t^{(i)})$
   Normalize weights $w_t^{(i)} = \frac{\widetilde{w}_t^{(i)}}{\sum_{i=1}^n \widetilde{w}_t^{(i)}}$

3. Return $\{x_T^{(i)}, w_T^{(i)}\}_{i=1}^N$

We never use this is real life beacuse of the degeneracy problem.

6 Degeneracy Problem

When we are sampling from our reweighted distribution, points with more mass tend to get more mass, while points with fewer mass tend to get less mass. Over a large number of iterations, this manifests as a handful of points having nearly all the weight while the rest of our points become negligable.

7 Bootstrap (SIR)

We resample to deal with the degeneracy problem. Modifying step 2 gives us:

2. *Importance Sampling and Resampling:*

   For $t = 1, \dots, T$:

   Sample $x_t^{(i)} \sim f(x_t|x_{t-1}^i)$
   Assign weights $\widetilde{w}_t^{(i)} = g(y_t|x_t^{(i)})$
   Normalize weights $w_t^{(i)} = \frac{\widetilde{w}_t^{(i)}}{\sum_{i=1}^n \widetilde{w}_t^{(i)}}$
   Resample $x_t^{(i)}$ according to the weight distribution.

Except that there's a whole new problem called the sampling impoverishment problem. There are methods that address both of these issues but they require fancy sounding things like the "Epanechnikov Kernel".