Unsupervised Learning 101: the EM for the HMM

Karl Stratos

1 What Does it Mean to Learn Unsupervised?

It means you have only the question but not the answer. Have you ever learned substantially merely by trying to solve a math problem, without even knowing the answer? You'd start with a guess based on your intuition, and align the question with your guess. If they're perfectly coherent, great! Otherwise, you'll change your guess and do this alignment again. How fast depends heavily on your mathematical talent, but you'll converge to your best bet (hopefully the right answer) most consistent with what the problem seems to ask for. In the process, you will have learned how this particular principle that you're dealing with in this problem works in general, and the next time you see it you'll have better "intuition" to start with.

2 Learning an HMM Unsupervised

This is exactly how the Expectation Maximization (EM) algorithm is used to learn the hidden parameters of a hidden Markov model (HMM). We see sequences of form $x_1, \ldots, x_l$ where $x_i \in \mathcal{X}$ for some observation domain $\mathcal{X}$ of size n. We assume there are hidden labels h_i in some hidden domain $\mathcal{H}$ of size m that generated these sequences by "emitting and moving". We can't see the labels though.

We want to learn about the hidden parameters $h \in \mathcal{H}$ of the HMM.

1. $P(h_1 = i)$: How likely is i the first hidden state of a sequence?
2. $P(x|h = i)$: How likely is the hidden state i to emit x?
3. $P(h' = j|h = i)$: How likely is the hidden state i move to the next j?

Note that if we have labeled data (i.e., sequences of form $(x_1, h_1), \ldots, (x_l, h_l)$), this task is really easy. Following the maximum likelihood estimation (MLE) principle, we can estimate

$$\hat{P}(h_1 = i) = \frac{C(h_1 = i)}{C(h_1)} \tag{1}$$

$$\hat{P}(x|h = i) = \frac{C(h = i, x)}{C(h = i)} \tag{2}$$

$$\hat{P}(h' = j|h = i) = \frac{C(h = i, h' = j)}{C(h = i)} \tag{3}$$

by counting the events in our data. This is not possible when we can't see the labels. We need to start with a guess, and keep improving it to make it coherent with the data.

2.1 A Top-Down Derivation of the EM

Guess at $\hat{P}(h_1), \hat{P}(x|h), \hat{P}(h'|h)$.[1] Implementation-wise, this means we'll fill up vectors and matrices for all possible $x \in \mathcal{X}, h \in \mathcal{H}$ with whatever values that form proper probability distributions.

In light of the pontification in Section 1, now we've guessed $\hat{P}(h_1), \hat{P}(x|h), \hat{P}(h'|h)$, we wish to see how well they explain the data. This means we'd like to have high probabilities for the events that occur frequently in the data—so we want go back to the MLE and use Eq. (1–3). But the problem seems unchanged; how do we *count* these *specific* events when all we have guessed are *probabilities* of *general* events (instantiation, emission, and transition)?

2.1.1 Counting with Probabilities

The idea is to use expected counts $\hat{C}$ instead of real counts C. The intuition is that when we sum up the probability of an event happening in our data, we get the "count" of the event in the data. For each sequence $\mathbf{x}$ of length $l_{\mathbf{x}}$ in our data, we'll sum over the probabilities of the events to get their expected counts.

$$\hat{C}(h_1 = i) = \sum_{\mathbf{x}} \hat{P}(h_1 = i|\mathbf{x}) \tag{4}$$

$$\hat{C}(h = i, x) = \sum_{\mathbf{x}} \sum_{t=1}^{l_{\mathbf{x}}} \hat{P}(h_t = i, x_t = x|\mathbf{x}) \tag{5}$$

$$\hat{C}(h = i, h' = j) = \sum_{\mathbf{x}} \sum_{t=1}^{l_{\mathbf{x}}-1} \hat{P}(h_t = i, h_{t+1} = j|\mathbf{x}) \tag{6}$$

Then $\hat{C}(h_1) = \sum_{h\in\mathcal{H}} \hat{C}(h_1 = h)$ and $\hat{C}(h = i) = \sum_{h'\in\mathcal{H}} \hat{C}(h = i, h')$.[2] Realize that these are all we need to estimate Eq. (1–3). So the only remaining question is, how do we get the probabilities of the specific events (e.g., $\hat{P}(h_t = i, x_t = x|\mathbf{x})$ or $\hat{P}(h_t = i, h_{t+1} = j|\mathbf{x})$ for a specific t in the data), when we have the general probabilities $\hat{P}(h_1), \hat{P}(x|h), \hat{P}(h'|h)$?

[1] It's a huge deal (and rightly so, for good performance) how to make this initial guess, but we don't worry about it here for simplicity.

[2] Be careful in evaluating Eq. (2,3) with this as the denominator. $\hat{C}(h = i) = \sum_{h'\in\mathcal{H}} \hat{C}(h = i, h')$ means the expected number of transition from $h = i$, what we want for (2). But for (3), we want $\hat{C}(h = i) = \sum_{x\in\mathcal{X}} \hat{C}(h = i, x)$, the expected number of emission from $h = i$. Now, they may be different depending on our convention. E.g., when we don't have the STOP symbol, the latter is greater than the former. If our handling of this issue is not proper, neither will the distribution be.

2.1.2 Probabilities of Specific Events

The idea is to inspect the hidden state probabilities at particular points in each sequence $\mathbf{x}$. This is, seemingly fortuitously, done as a subroutine of the dynamic programming (DP) procedure that computes the probability of a sequence $P(\mathbf{x})$ (called the "inside probability") by considering all possible hidden states that could have generated $\mathbf{x}$.

Denote the subsequence $x_i, \ldots, x_j$ by $\mathbf{x}_{i:j}$. Usually, when one just wants the inside probability, doing a left-to-right DP suffices (the "forward algorithm"). There, for each point $t = 1, \ldots, l_{\mathbf{x}}$, we compute $P(\mathbf{x}_{1:t}, h_t)$ for all $h_t \in \mathcal{H}$, finally summing over the last state to obtain the desired quantity: $\sum_{h_{l_{\mathbf{x}}} \in \mathcal{H}} P(\mathbf{x}, h_{l_{\mathbf{x}}}) = P(\mathbf{x})$. Another way of obtaining it is to do a right-to-left DP (the "backward algorithm"). Here, for each point $t = l_{\mathbf{x}}, \ldots, 1$, we compute $P(\mathbf{x}_{t+1:l_{\mathbf{x}}} | h_t)$ for all $h_t \in \mathcal{H}$, finally evaluating $\sum_{h_1 \in \mathcal{H}} P(h_1) P(x_1|h_1) P(\mathbf{x}_{2:l_{\mathbf{x}}}|h_1) = P(\mathbf{x})$.

We're interested in the intermediate values $P(\mathbf{x}_{1:t}, h_t)$ and $P(\mathbf{x}_{t+1:l_{\mathbf{x}}}|h_t)$ because they allow us to exactly infer $\hat{P}(h_1 = i|\mathbf{x})$, $\hat{P}(h_t = i, x_t = x|\mathbf{x})$, and $\hat{P}(h_t = i, h_{t+1} = j|\mathbf{x})$—the three quantities we need for computing the expected counts. To highlight their role, define $\alpha_t(i) = P(\mathbf{x}_{1:t}, h_t = i)$ and $\beta_t(i) = P(\mathbf{x}_{t+1:l_{\mathbf{x}}}|h_t = i)$. The key is to exploit the Markovian assumption to derive joint probabilities, from which we can compute

$$\hat{P}(h_1 = i|\mathbf{x}) = \frac{\hat{P}(h_1 = i, \mathbf{x})}{\hat{P}(\mathbf{x})} \tag{7}$$

$$\hat{P}(h_t = i, x_t = x|\mathbf{x}) = \frac{\hat{P}(h_t = i, x_t = x, \mathbf{x})}{\hat{P}(\mathbf{x})} \tag{8}$$

$$\hat{P}(h_t = i, h_{t+1} = j|\mathbf{x}) = \frac{\hat{P}(h_t = i, h_{t+1} = j, \mathbf{x})}{\hat{P}(\mathbf{x})} \tag{9}$$

The easiest way to see how the joint probabilities $\hat{P}(h_1 = i, \mathbf{x})$, $\hat{P}(h_t = i, x_t = x, \mathbf{x})$, and $\hat{P}(h_t = i, h_{t+1} = j, \mathbf{x})$ may be derived is to "break up" the expression into known quantities. Realize that computing $\hat{P}(h_t, \mathbf{x})$ for all values of h_t is sufficient for $\hat{P}(h_1, \mathbf{x})$ (a special case where $t = 1$) and $\hat{P}(h_t, x_t = x, \mathbf{x})$ ($\hat{P}(h_t, \mathbf{x})$ if $x_t = x$, 0 otherwise).

$$\begin{aligned}
\hat{P}(h_t = i, \mathbf{x}) &= \hat{P}(\mathbf{x}_{1:t}, h_t = i, \mathbf{x}_{t:l_{\mathbf{x}}}) \\
&= \hat{P}(\mathbf{x}_{1:t}|h_t = i)\hat{P}(\mathbf{x}_{t+1:l_{\mathbf{x}}}|h_t = i) \\
&= \alpha_t(i)\beta_t(i)
\end{aligned} \tag{10}$$

Finally, $\hat{P}(h_t = i, h_{t+1} = j, \mathbf{x})$ can be broken up likewise. To clarify the decomposition process, define $M(i,j) = \hat{P}(h_{t+1} = j|h_t = i)\hat{P}(x_{t+1}|h_{t+1} = j)$ as the probability of moving from i to j at time t, and then emitting x_{t+1}. Note that we know this value by our guess.

$$\begin{aligned}\hat{P}(h_t=i,h_{t+1}=j,\mathbf{x}) &= \hat{P}(\mathbf{x}_{1:t},h_t=i,h_{t+1}=j,\mathbf{x}_{t+1:l\_x})\\ &= \hat{P}(\mathbf{x}_{1:t}|h_t=i)M(i,j)\hat{P}(\mathbf{x}_{t+2:l\_x}|h_{t+1}=j)\\ &= \alpha_t(i)M(i,j)\beta_{t+1}(j) \end{aligned} \tag{11}$$

The denominator $\hat{P}(\mathbf{x})$ is given for free, as a consequence of running the forward or backward algorithm. Thus at this point, we have all the ingredients to compute $\hat{P}(h_1), \hat{P}(x|h), \hat{P}(h'|h)$ that are "better" for this particular training data, from their previous guess. We can compute even better values from these improved values, again and again, until we reach some local optimum (usually verified in the convergence of $\hat{P}(\mathbf{x})$—or its log value—which represents how well the data is explained by the parameters). This is at the heart of the EM machinery.

Remark: It may feel unsatisfying to simply accept that the forward and backward algorithm provide us with the necessary quantities, but this is really a corollary of the sequential nature of an HMM. The technical leap is made in the conditional probabilities $P(\mathbf{x}_{t+1:l_{\mathbf{x}}}|h_t)$ of the backward algorithm, which together with $P(\mathbf{x}_{1:t},h_t)$ from the forward algorithm lets us specify particular hidden states at particular times, $\hat{P}(h_t=i,\mathbf{x})$ and $\hat{P}(h_t=i,h_{t+1}=j,\mathbf{x})$, which will yield what we are after when conditioned over $\mathbf{x}$.

2.2 The Algorithm

We assume we are given unlabeled sequences $\mathbf{x}^1,\dots,\mathbf{x}^N$. We suspect that there is a hidden HMM responsible for generating these sequences, and try to estimate the model parameters $P(h_1), P(x|h), P(h'|h)$ from the data.

1. Initialize $\hat{P}(h_1), \hat{P}(x|h), \hat{P}(h'|h)$ randomly.
2. Until convergence,
   (a) *E Step*: For each sequence $\mathbf{x}$, compute $\alpha_t(i), \beta_t(i), \hat{P}(\mathbf{x})$ from the forward-backward algorithm, and get $\hat{P}(h_1|\mathbf{x}), \hat{P}(h_t,x_t|\mathbf{x}), \hat{P}(h_t,h_{t+1}|\mathbf{x})$ based on these values [Eq. (7–11)]. Then compute the expected counts $\hat{C}(h_1), \hat{C}(h,x), \hat{C}(h,h')$ by summing over all sequences [Eq. (4–6)]
   (b) *M Step*: Update the model parameters $\hat{P}(h_1), \hat{P}(x|h), \hat{P}(h'|h)$ with these counts [Eq. (1–3)].

The two steps are named so because the *E Step* obtains the expected counts, and the *M Step* maximizes the likelihood of the data with the MLE update of the parameters. Any algorithm called the "EM" necessarily follows a similar procedure.