Lecture 4 — April 9

Lecturer: Lester Mackey *Scribe: Yishun Dong, Sam Paglia*

4.1 Recap

In the last lecture, we developed general approach using EM to approximate the maximum likelihood estimator (MLE) in latent variable models. The EM algorithm we studied in last lecture was explained and given its name in the seminal work by Arthur Dempster, Nan Laird, and Donald Rubin in 1977. They note that the method has been proposed many times in various special cases in earlier works, with the earliest reference dating back to 1926.

In particular, the EM algorithm gives us a general purpose tool for probabilistic mixture modeling, which in turn enables a great variety of applications. The following examples demonstrate the diversity of domains and data types to which mixture modeling can be applied.

Example 1. Email clustering for legal document review is one such example, which we studied last lecture using the Multinomial Mixture Model (MMM).

Example 2. Modeling uncertain disease state (cured/uncured) in studies of cancer patient survival (Yu/Tiwari 2007).

Example 3. Inferring community structure from network interactions (Newman 2007).

Example 4. Inferring test-taking behaviors from GRE responses (Schnipke/Scrams 1997).

4.2 Models with Temporal or Sequential Structure

While these generic mixture models are broadly applicable, they all assume that data points are generated **independently** from a common density. Such an assumption is inappropriate (or at least, wasteful) if our data has a known dependence structure. For instance, it is not uncommon for data to have a known **temporal** or **sequential** structure, and we would like to leverage this structure in performing unsupervised learning.

Let us consider a few examples of sequentially structured. Some instances arise in the common setting in which we have a small amount of labeled data (e.g., transcribed recordings of speech), which is expensive to obtain, and a large amount of cheap, unlabeled data. Incorporating both labeled and unlabeled data into our learning procedure will generally yield more accurate and actionable results than either set used individually. This setting is called the *semi-supervised learning* setting, and all of our unsupervised methods naturally extend to ths setting (e.g., we can incorporate this label information into our probabilistic models, by treating the class indicator z_i as observed for any labeled datapoint x_i).

Example 5 (Speech Recognition)**.** We observe an audio signal, segment the signal, and extract features. For instance, Figure 4.1 might be the audio wave generated by someone who's trying to pronounce an "x". The goal is to infer the latent words or phonemes for each temporally indexed audio segment. In this setting, our inferences can benefit from modeling the dependence between the current word or sound under consideration and adjacent words or sounds.

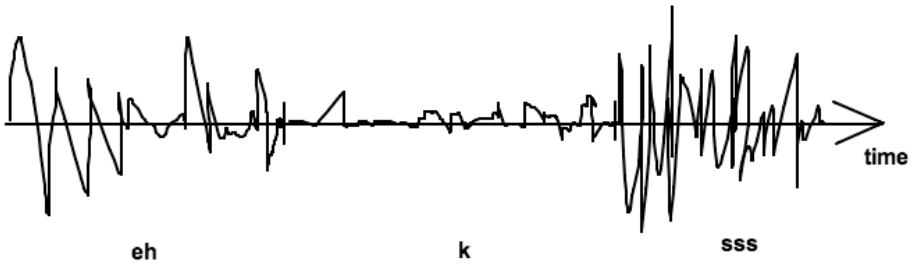

Figure 4.1. Audio signal of someone pronouncing "x".

Example 6 (Handwriting Recognition)**.** Figure 4.2 shows an example of an old/unreadable text, and we would like to identify words or characters from the handwritten text. It often helps to use surrounding context in determining the word/letter to identity. For example, the letter immediately following a character 't' is not uniformly distributed among the alphabet. It might be much more likely to have an 'i' following a 't' than a 'z' following a 't'.

Figure 4.2. Handwriting of "Lester M"

Example 7 (Copy Number Segmentation)**.** In this setting, we observe noisy intensity measurements at each genome locus, and the goal is to estimate the true DNA copy number at each location from normal or tumor samples. The copies often occur in large contiguous regions (see the picture from online slide for this lecture). Therefore, the information from neighboring copy numbers is highly informative in this task.

Such sequential data examples call for methods that can both:

1. Uncover the clustered structure associated with our data

2. Take advantage of the known sequential dependence

Today, we will take a probabilisIc modeling approach to clustering sequential data with discrete hidden Markov models (HMMs); this approach has found great success in each of the settings just described.

4.3 Discrete Hidden Markov Models (HMMs)

Discrete hidden Markov models are probabilistic models for clustered **sequential** data and can be viewed sequential generalizations of independent mixture models. We view our sequence of observed data points $x_0, x_1, ..., x_T$ as a random draw from the following generative process:

Step One: First, we sample the initial latent **state** (or cluster) $z_0 \in \{1, 2, ..., k\}$, according to

$$z_0 \sim \text{Mult}(\pi, 1),$$

where π is an unknown probability vector in $\mathbb{R}^k$.

Step Two: For each $t > 0$, we sample the new state z_t based on the prior state according to the following **transition probability**

$$z_t | z_{t-1} = j \overset{\text{ind}}{\sim} \text{Mult}(a_j, 1)$$

where a_j is an (unknown) transition probability vector in $\mathbb{R}^k$.

The matrix of all transition probability vectors

$$A = \begin{bmatrix} - & a_1 & - \\ - & a_2 & - \\ & \vdots & \\ - & a_k & - \end{bmatrix}$$

is called the transition matrix.

Note: we could generalize to different transition matrices (A^t) at each time step t, but the **homogeneous** case ($A^t = A$, $\forall t$) is the most common scenario.

Step Three: Finally, we independently generate the observed data from state specific densities

$$x_t | z_t \overset{\text{ind}}{\sim} p(x_t | z_t; \eta)$$

where $p(x_t | z_t; \eta)$ are called the **emission probabilities** and η is a vector of unknown parameters.

Figure 4.3 shows a graphical depiction of the Hidden Markov Model we described above. Note that since the distribution of z_t is determined fully by z_{t-1}, the hidden sequence $(z_0, z_1, ..., z_T)$ is a **Markov Chain**, hence the name Hidden Markov Model.

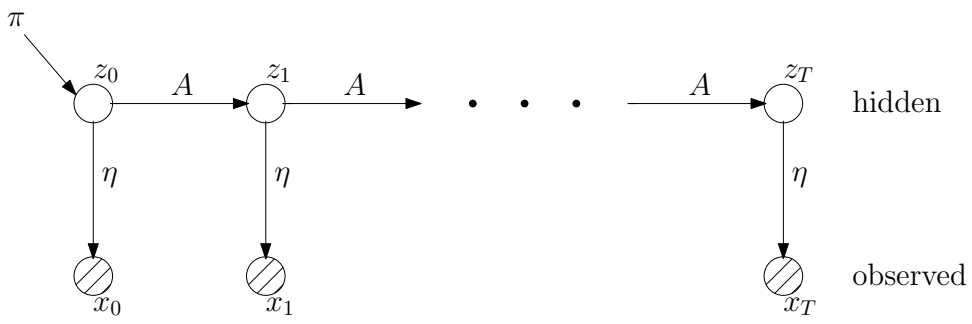

Figure 4.3. Graphical depiction of the hidden markov model (HMM).

4.3.1 Conditional independence in the HMM

To ease our future calculations in carrying out probabilistic inference, let us record an important conditional independence relation that exists amongst the variables in this model. Because past variables are connected to future variables only through the links between hidden states z_t, knowledge of the future (hidden or observed) is independent of knowledge of the past (hidden or observed) given knowledge of the current hidden state z_t.

More formally, given the structure of our HMM, the following holds for all times t and collections of preceding and succeeding states:

$$p(x_{t+1:T}, z_{t+1:T}|z_t, z_{1:t-1}, x_{1:t}) = p(x_{t+1:T}, z_{t+1:T}|z_t).$$

Readers further interested in probabilistic inference in graphical models could refer to Koller and Friedman's work, *Probabilistic Graphical Models: Principals and Techniques*, MIT Press, 2009

4.3.2 Joint density of $X_{0:T}$

Let $\theta = (\pi, A, \eta)$ denote all the HMM parameters, $x = (x_0, x_1, ..., x_T)$, and $z = (z_0, z_1, ..., z_T)$. Then the joint density for x takes the form

$$\begin{aligned} p(x;\theta) &= \sum_z p(x, z; \theta) \\ &= \sum_z p(z_0;\pi) \prod_{t=0}^{T-1} p(z_{t+1}|z_t; A) \prod_{t=0}^{T} p(x_t|z_t;\eta) \\ &= \sum_z \pi_{z_0} \prod_{t=0}^{T-1} a_{z_t, z_{t+1}} \prod_{t=0}^{T} p(x_t|z_t;\eta). \end{aligned}$$

4.3.3 Unsupervised learning goal

We want to infer the hidden structure z from the sequence x, which is more complicated than in the i.i.d. setting. To simplify matters, let us first suppose all the parameters θ are

known. How would we infer a single hidden state z_t from the sequence x? Let us consider the usual conditional distribution

$$\begin{aligned} p(z_t|x) &= \frac{p(x|z_t)p(z_t)}{p(x)} \\ &\overset{(a)}{=} \frac{p(x_0, ..., x_t|z_t)p(x_{t+1}, ..., x_T|z_t)p(z_t)}{p(x)} \\ &= \frac{p(x_{0:t}, z_t)p(x_{t+1:T}|z_t)}{p(x)} \\ &= \frac{\alpha(z_t)\beta(z_t)}{p(x)}. \end{aligned}$$

where (a) follows from the conditional independence given z_t between the data up to time t and the future data. Note that we have defined two new quantitates in our last line:

$$\alpha(z_t) \triangleq p(x_{0:t}, z_t)$$

$$\beta(z_t) \triangleq p(x_{t+1:T}|z_t).$$

We see that $\alpha(z_t) \triangleq p(x_{0:t}, z_t)$ represents the probability of emitting all past and present data ($x_{0:t}$) and ending up in state z_t, while $\beta(z_t) \triangleq p(x_{t+1:T}|z_t)$ represents the probability of emitting all future data ($x_{t+1:T}$) given the current state z_t.

Since we know

$$1 = \sum_{z_t} p(z_t|x) = \sum_{z_t} \frac{\alpha(z_t)\beta(z_t)}{p(x)}$$

we have the following relation

$$\boxed{p(x) = \sum_{z_t} \alpha(z_t)\beta(z_t)}.$$

Therefore, if we knew $\alpha(z_t), \beta(z_t)$ for all states z_t, we could calculate the conditional distribution $p(z_t|x)$. What is the cost of computing these quantities $\alpha(z_t), \beta(z_t)$?

Bad news: Inference for z_t at a single time-point t will take order k^2T work to carry out, so, seemingly, if we were to carry out inference for each time point individually, the total computation cost of inference over all states z_t will be $O(k^2T^2)$, which is problematic for large T (long sequences).

However, all is not lost.

Good news: We can in fact leverage the structure of our model to compute $\alpha(z_t)$, $\beta(z_t)$ recursively in time, so that total time for computing $p(z_t|x)$ **for all t** is $O(k^2T)$.

Next, we will illustrate how this is recursive procedure is carried out

4.4 Recursions to calculate $\alpha(z_t), \beta(z_t)$

We now show how to recursively calculate $\{\alpha(z_t)\}_{t=0}^T, \{\beta(z_t)\}_{t=0}^T$ in $O(k^2T)$ time. We begin with our computation of $\alpha(z_t)$.

4.4.1 Alpha recursion / forward pass

Our calculation of $\alpha(z_t)$ can also be termed a "forward pass", as we will first define $\alpha(z_0)$ and subsequently calculate $\alpha(z_{t+1})$ in terms of $\alpha(z_t)$ for all $t \in \{1, ..., T\}$.

Given that our general definition of α was $\alpha(z_t) = p(x_{0:t}, z_t)$, for $t = 0$ we have $\alpha(z_0) = p(x_0, z_0) = p(x_0|z_0)p(z_0) = p(x_0|z_0)\pi_{z_0}$, which is easily computable given our known quantities. We now develop the recursive formula for $\alpha(z_{t+1})$:

$$\begin{aligned}
\alpha(z_{t+1}) &= p(x_{0:t+1}, z_{t+1}) && (4.1)\\
&= \sum_{z_t} p(x_{0:t+1}, z_{t+1}, z_t) && (4.2)\\
&= \sum_{z_t} p(x_{0:t}, z_t)p(x_{t+1}, z_{t+1}|z_t, x_{0:t}) && (4.3)\\
&= \sum_{z_t} \alpha(z_t)p(x_{t+1}, z_{t+1}|z_t) && (4.4)\\
&= \sum_{z_t} \alpha(z_t)p(x_{t+1}|z_{t+1})p(z_{t+1}|z_t) && (4.5)\\
&= \sum_{z_t} \alpha(z_t)a_{z_t,z_{t+1}}p(x_{t+1}|z_{t+1}) && (4.6)
\end{aligned}$$

Going through this derivation in more detail, we see that (4.2) is an application of the law of total probability, while (4.3) is an application of the chain rule. In (4.4), we substitute based on the equality $\alpha(z_t) = p(x_{0:t}, z_t)$ and leverage the dependence structure of our HMM to drop the $x_{0:t}$ from the $p(x_{t+1}, z_{t+1}|z_t, x_{0:t})$ term given that z_t is known. Finally, we split our $p(x_{t+1}, z_{t+1}|z_t)$ term according to the dependence structure of our HMM in (4.5) and substitute terms in line (4.6).

Thus, we have the following forward recurrence relation for $\alpha(z_t)$:

$$\boxed{\alpha(z_{t+1}) = \sum_{z_t} \alpha(z_t)a_{z_t,z_{t+1}}p(x_{t+1}|z_{t+1})}$$

Let us turn to the computational cost of computing $\{\alpha(z_t)\}_{t=0}^T$ using the above recurrence relation. Since z_t can take any value in $\{1, 2, ..., k\}$, it takes $O(k)$ operations to compute the sum in the boxed equation for one value of z_{t+1}. As z_{t+1} can also take on value in $\{1, 2, ..., k\}$, updating from t to $t + 1$ is an $O(k^2)$ operation. Finally, since it takes T such steps to finish computing $\{\alpha(z_t)\}_{t=0}^T$, the total computation cost is $O(k^2T)$ as mentioned earlier. We now move on to our β terms.

4.4.2 Beta recursion / backward pass

In contrast to our calculation of $\alpha(z_t)$, we compute $\beta(z_t)$ via a "backward pass", as will first define $\beta(z_T)$ and subsequently calculate $\beta(z_t)$ in terms of $\beta(z_{t+1})$ for all $t < T$. We begin by defining $\beta(z_T) = 1$, and now develop a recursive formula for $\beta(z_t)$. Consider the calculations below

$$\begin{aligned}
\beta(z_t) &= p(x_{t+1:T}|z_t) && (4.7)\\
&= \sum_{z_{t+1}} p(x_{t+1}, x_{t+2:T}, z_{t+1}|z_t) && (4.8)\\
&= \sum_{z_{t+1}} p(z_{t+1}|z_t)p(x_{t+1}|z_{t+1}, z_t)p(x_{t+2:T}|x_{t+1}, z_{t+1}, z_t) && (4.9)\\
&= \sum_{z_{t+1}} a_{z_t,z_{t+1}} p(x_{t+1}|z_{t+1})p(x_{t+2:T}|z_{t+1}) && (4.10)\\
&= \sum_{z_{t+1}} a_{z_t,z_{t+1}} p(x_{t+1}|z_{t+1})\beta(z_{t+1}) && (4.11)
\end{aligned}$$

Note that (4.8) is due to the law of total probability, while (4.9) is an application of the chain rule. In (4.10) we substitute terms and leverage the dependence structure of our model to drop terms. Finally, in (4.11), we substitute our existing identity for $\beta(z_{t+1})$.

Thus, we have the following backward recurrence relation for $\beta(z_t)$:

$$\boxed{\beta(z_t) = \sum_{z_{t+1}} a_{z_t,z_{t+1}} p(x_{t+1}|z_{t+1})\beta(z_{t+1})}$$

Since again, both z_{t+1} and z_t can take on values in $\{1, 2, ..., k\}$, it costs $O(k)$ to compute the sum in the above boxed equation for one value of z_t. There are k such $\beta(z_t)$'s we need compute at each step, so the cost per step is $O(k^2)$. Finally, since there are total T steps, in much the same way as for the $\alpha(z_t)$ case, we conclude computing $\{\beta(z_t)\}_{t=0}^T$ is an $O(k^2T)$ operation.

Because we can compute $\{\alpha(z_t)\}_{t=0}^T, \{\beta(z_t)\}_{t=0}^T$ in $O(k^2T)$ time, we can calculate $p(z_t|x)$ for all t fairly efficiently.

We can now perform inference on z_t in isolation via $p(z_t|x)$. However, this is not the same as making inference surrounding groups of z_t. This is the case because our hidden states z are dependent given x, e.g., $p(z_t, z_{t+1}|x) \neq p(z_t|x)p(z_{t+1}|x)$.

This leaves us with several open questions:

- How do we compute these conditional co-occurrence probabilities?
- How do we estimate θ?

We will find out in the next lecture.