1 Latent variable models

In the next section we will discuss latent variable models for unsupervised learning, where instead of trying to learn a mapping from regressors to responses (e.g. from stimuli to responses), we are simply trying to capture structure in a set of observed responses.

The word \textit{latent} simply means \textit{unobserved}. Latent variables are simply random variables that we posit to exist underlying our data. We could also refer to such models as \textit{doubly stochastic}, because they involve two stages of noise: noise in the latent variable and then noise in the mapping from latent variable to observed variable.

Specifically, we will specify latent variable models in terms of two pieces

- Prior over the latent: \( z \sim p(z) \)
- Conditional probability of observed data: \( x|z \sim p(x|z) \)

The probability of the observed data \( x \) is given by an integral over the latent variable:

\[
p(x) = \int p(x|z)p(z)dz
\]

(1)

or a sum in the case of discrete latent variables:

\[
p(x) = \sum_{i=1}^{m} p(x|z=\alpha_i)p(z=\alpha_i),
\]

(2)

where the latent variable takes on a finite set of values \( z \in \{\alpha_1, \alpha_2, \ldots, \alpha_m\} \).

2 Two key things we want to do with latent variable models

1. \textbf{Recognition / inference} - refers to the problem of inferring the latent variable \( z \) from the data \( x \). The posterior over the latent given the data is specified by Bayes’ rule:

\[
p(z|x) = \frac{p(x|z)p(z)}{p(x)},
\]

(3)

where the model is specified by the terms in the denominator, and the denominator is the marginal probability obtained by integrating the numerator, by \( p(x) = \int p(x|z)p(z)dz \).
2. **Model fitting** - refers to the problem of learning the model parameters, which we have so far suppressed. In fact we should write the model as specified by

\[ p(x, z|\theta) = p(x|z, \theta)p(z|\theta) \]  

(4)

where \( \theta \) are the parameters governing both the prior over the latent and the conditional distribution of the data.

Maximum likelihood fitting involves computing and maximizing the marginal probability:

\[ \hat{\theta} = \arg \max_{\theta} p(x|\theta) = \arg \max_{\theta} \int p(x, z|\theta)dz. \]  

(5)

3. **Example: binary mixture of Gaussians (MoG)**

(Also commonly known as a Gaussian mixture model (GMM)).

This model is specified by:

\[ z \sim \text{Ber}(p) \]  

(6)

\[ x|z \sim \begin{cases} 
    \mathcal{N}(\mu_0, C_0), & \text{if } p = 0 \\
    \mathcal{N}(\mu_1, C_1), & \text{if } p = 1 
\end{cases} \]  

(7)

So \( z \) is a binary random variable that takes value 1 with probability \( p \) and value 0 with probability \((1 - p)\). The datapoint \( x \) is then drawn from either Gaussian \( \mathcal{N}_0(x) = \mathcal{N}(\mu_0, C_0) \) if \( p = 0 \) or a different Gaussian \( \mathcal{N}_1(x) = \mathcal{N}(\mu_1, C_1) \) if \( p = 1 \).

For this simple model the recognition distribution (conditional distribution of the latent):

\[ p(z = 0|x) = \frac{(1 - p)\mathcal{N}_0(x)}{(1 - p)\mathcal{N}_0(x) + p\mathcal{N}_1(x)} \]  

(8)

\[ p(z = 1|x) = \frac{p\mathcal{N}_1(x)}{(1 - p)\mathcal{N}_0(x) + p\mathcal{N}_1(x)} \]  

(9)

The likelihood (or marginal likelihood) is simply the normalizer in the expressions above:

\[ p(x|\theta) = (1 - p)\mathcal{N}_0(x) + p\mathcal{N}_1(x), \]  

(10)

where the model parameters are \( \theta = \{p, \mu_0, C_0, \mu_1, C_1\} \).

For an entire dataset, likelihood would be the product of independent terms, since we assume each latent \( z_i \) is drawn independently from the prior, giving:

\[ p(X|\theta) = \prod_{i=1}^{N} \left( (1 - p)\mathcal{N}_0(x_i) + p\mathcal{N}_1(x_i) \right) \]  

(11)

and hence

\[ \log p(X|\theta) = \sum_{i=1}^{N} \log \left( (1 - p)\mathcal{N}_0(x_i) + p\mathcal{N}_1(x_i) \right). \]  

(12)
Clearly we could write a function to compute this sum and use an off-the-shelf algorithm to optimize it numerically if we wanted to. However, we will next discuss an alternative iterative approach to maximizing the likelihood.

4 The Expectation-Maximization (EM) algorithm

4.1 Jensen’s inequality

Before we proceed to the algorithm, let’s first describe one of the tools used in its derivation.

**Jensen’s inequality:** for any concave function $f$ and $p \in [0, 1]$, 

$$f((1-p)x_1 + px_2) \geq (1-p)f(x_1) + pf(x_2).$$  \hspace{1cm} (13)

The left hand side is the function $f$ evaluated at a point somewhere between $x_1$ and $x_2$, while the right hand side is a point on the straight line (a chord) connecting $f(x_1)$ and $f(x_2)$. Since a concave function lies above any chord, this follows straightforwardly from the definition of concave functions. (For convex functions the inequality is reversed!)

In our hands we will use the function $f(x) = \exp(x)$, in which case we can think of Jensen’s inequality as equivalent to the statement that “The log of the average is greater than or equal to the average of the logs”.

The inequality can be extended to any continuous probability distribution $p(x)$ and implies that:

$$f(\int p(x)g(x)dx) \geq \int p(x)f(g(x))dx$$  \hspace{1cm} (14)

for any concave $f(x)$, or in our case:

$$\log \int p(x)g(x)dx \geq \int p(x)\log g(x).$$  \hspace{1cm} (15)

4.2 EM

The expectation-maximization algorithm is an iterative method for finding the maximum likelihood estimate for a latent variable model. It consists of iterating between two steps ("Expectation step" and "Maximization step", or "E-step" and "M-step" for short) until convergence. Both steps involve maximizing a lower bound on the likelihood.

Before deriving this lower bound, recall that $p(x|z, \theta)p(z|\theta) = p(x, z|\theta) = p(z|x, \theta)p(x|\theta)$. This is a quantity known in the EM literature as the total data likelihood.

The log-likelihood can be lower-bounded through a straightforward application of Jensen’s inequal-
\[
\log p(x|\theta) = \log p(x, z|\theta)dz 
= \log q(z|\phi)p(x, z|\theta)q(z|\phi)dz 
\geq \int q(z|\phi) \log \left[ \frac{p(x, z|\theta)}{q(z|\phi)} \right] dz 
\triangleq F(\phi, \theta) 
\]

Here \(q(z|\phi)\) is an arbitrary distribution over the latent \(z\), with parameters \(\phi\). The quantity we have obtained in equation (eq. 18) is known as the negative free energy \(F(\phi, \theta)\).

We will now write the negative free energy in two different forms. First:

\[
-F(\phi, \theta) = \int q(z|\phi) \log \left[ \frac{p(x, z|\theta)}{q(z|\phi)} \right] dz 
= \int q(z|\phi) \log \left[ \frac{p(x|\theta)p(z|x, \theta)}{q(z|\phi)} \right] dz 
= \int q(z|\phi) \log p(x|\theta) + \int q(z|\phi) \log \left[ \frac{p(z|x, \theta)}{q(z|\phi)} \right] dz 
= \log p(x|\theta) - KL\left(q(z|\phi) || p(z|x, \theta)\right) 
\]

This last line makes clear that the NFE is indeed a lower bound on \(\log p(x|\theta)\) because the KL divergence is always non-negative. Moreover, it shows how to make the bound tight, namely by setting \(\phi\) such that the \(q\) distribution is equal to the conditional distribution over the latent given the data and the current parameters \(\theta\), i.e., \(q(z|\phi) = p(z|x, \theta)\).

A second way to write the NFE that will prove useful is:

\[
-F(\phi, \theta) = \int q(z|\phi) \log \left[ \frac{p(x, z|\theta)}{q(z|\phi)} \right] dz 
= \int q(z|\phi) \log p(x, z|\theta)dz - \int q(z|\phi) \log q(z|\phi)dz. 
\]

Here we observe that the second term is independent of \(\theta\). We can therefore maximize the NFE for \(\theta\) by simply maximizing the first term.

We are now ready to define the two steps of the EM algorithm:

- **E-step**: Update \(\phi\) by setting \(q(z|\phi) = p(z|x, \theta)\) (eq. 23), with \(\theta\) held fixed.

- **M-step**: Update \(\theta\) by maximizing the expected total data likelihood, \(\int q(z|\phi) \log p(x, z|\theta)dz\) (eq. 25), with \(\phi\) held fixed.

Note that the lower bound on the log-likelihood will be tight after each E-step.