COMP 790-125: Goals for today

- Intro to Information Theory
- Mixture models
- Our first Expectation Maximization algorithm
- Numerical tricks, EM init

Intro to Information Theory

Suppose you have 4 messages (**a**,**b**,**c** and **d**) you may want to communicate to a friend.

You might opt to encode messages as[1]

- $\text{Enc}(\mathbf{a}) = 00$
- $\text{Enc}(\mathbf{b}) = 01$
- $\text{Enc}(\mathbf{c}) = 10$
- $\text{Enc}(\mathbf{d}) = 11$

Note that the length of each message is 2 bits (notation $|\mathrm{Enc}(\mathbf{a})| = 2$)

Regardless of the message you want to communicate it always takes 2 bits.

[1] this table is called a codebook

Expected/average message length

The expected length of message is

$$E_p[|\mathrm{Enc}(m)|] = \sum_{m \in \{\mathbf{a,b,c,d}\}} p(m)|\mathrm{Enc}(m)|$$

so with your codebook that uses 2 bits per message, on average for each message you will use ... 2 bits

Intro to Information Theory

Suppose you knew something extra: the probability that a particular message will need to be transmitted.

$$p(m) = \begin{cases} 1/2, & m = \mathbf{a} \\ 1/4, & m = \mathbf{b} \\ 1/8, & m = \mathbf{c} \\ 1/8, & m = \mathbf{d} \end{cases}$$

Could you then take advantage of this information?

Intro to Information Theory

Short codewords for frequent messages, longer codewords for infrequent messages[2]

- $p(m = a) = 1/2$, $\text{Enc}(\mathbf{a}) = 0$
- $p(m = b) = 1/4$, $\text{Enc}(\mathbf{b}) = 10$
- $p(m = c) = 1/8$, $\text{Enc}(\mathbf{c}) = 110$
- $p(m = a) = 1/8$, $\text{Enc}(\mathbf{d}) = 111$

and expected message length is

$$1/2 * 1 + 1/4 * 2 + 1/8 * 3 + 1/8 * 3 = 1.75$$

so shorter than 2 bits we had earlier.

[2] reordering inequality.

Entropy

You can show that the codeword length assignment that minimizes the expected message length is $-\log_2 p(m)$.

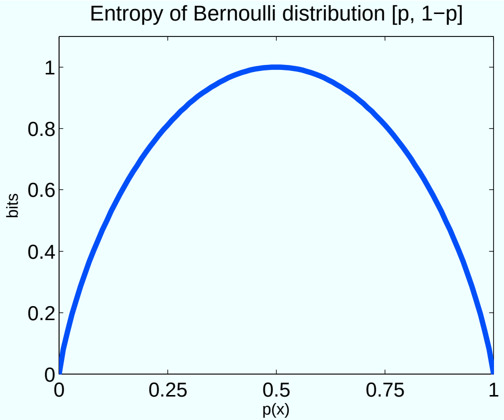
This optimal expected message length is called entropy $H(p)$

$$H(p) = \sum_m p(m) \left[-\log_2 p(m)\right]$$

The more uniform the distribution the higher the entropy (I need 2 bits for 4 messages with prob. $1/4$).

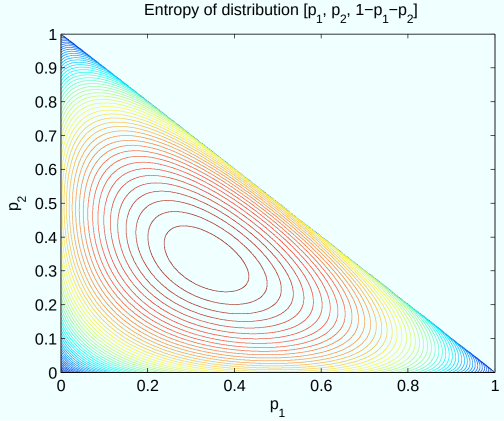
The less uniform (or more concentrated) the distribution the lower the entropy (I need 0 bits if I always say the same thing).

Entropy



$$H\left(\left[\frac{1}{2}\,\frac{1}{2}\right]\right) = \frac{1}{2}\left(-\log_2\frac{1}{2}\right) + \frac{1}{2}\left(-\log_2\frac{1}{2}\right) = \frac{1}{2}\log_2 2 + \frac{1}{2}\log_2 2 = 1$$

Entropy



Where is the maximum of entropy in this case? What is the value of entropy there?

$$H\left(\left[\frac{1}{3}\frac{1}{3}\frac{1}{3}\right]\right) = \frac{1}{3}\left(-\log_2\frac{1}{3}\right) + \frac{1}{3}\left(-\log_2\frac{1}{3}\right) + \frac{1}{3}\left(-\log_2\frac{1}{3}\right) = \log_2(3)$$

Cross-entropy

Suppose we were given message probabilities q.

We built our codebook based on q and for all we know the optimal average message length is $H(q)$.

But it turns out the q is incorrect and the true message probabilities are p.

Cross-entropy is the average message length under these circumstances

$$H(p, q) = -\sum_m p(m) \log q(m)$$

Kulback Leibler divergence

Had we known the true distribution p our average message length would be

$$H(p) = -\sum_m p(m) \log p(m)$$

we didn't and we are now on average using a *longer* message

$$H(p, q) = -\sum_m p(m) \log q(m).$$

How much longer?

$$H(p, q) - H(p) = -\sum_m p(m) \log q(m) + \sum_m p(m) \log p(m)$$

This difference is called Kullback-Leibler divergence

$$\mathrm{KL}(p||q) = H(p, q) - H(p) = -\sum_m p(m) \log q(m) + \sum_m p(m) \log p(m)$$

A couple of observations about KL-divergence

1. $\mathrm{KL}(p||q) \geq 0$
2. $\mathrm{KL}(p||q) = 0$ if and only if $p = q$
3. it is not symmetric $\mathrm{KL}(p||q) \neq \mathrm{KL}(q||p)$

Information Theory recap

- Entropy as a measurement of the number of bits needed to communicate efficiently.
- KL-divergence as a number of bits that could be saved with the right distribution
- KL-divergence as a distance between distributions.

See, MacKay's book Information Theory, Inference, and Learning Algorithms
`http://www.cs.toronto.edu/~mackay/itprnn/book.pdf`

Approximation

Generally, we build approximations of functions because they are easy to optimize, analyze, reason about, etc.

We learned about quadratic approximations.

There are other approaches to building approximations.

Jensen's inequality enter stage right

If $f(\mathbf{x})$ is a concave function then

$$\begin{aligned} f(\mathbf{E}[\mathbf{x}]) &\geq \mathbf{E}[f(\mathbf{x})] \\ f\left(\sum_{\mathbf{v}} p(\mathbf{x}=\mathbf{v})\mathbf{v}\right) &\geq \sum_{\mathbf{v}} p(\mathbf{x}=\mathbf{v}) f(\mathbf{v}) \end{aligned}$$

To give a **very** particular example, if $p(\mathbf{x})$ is an uniform distribution then

$$f\left(\sum_{\mathbf{v}} \frac{1}{n}\mathbf{v}\right) \geq \sum_{\mathbf{v}} \frac{1}{n} f(\mathbf{v}).$$

Mixtures

Let's start with demos, since you had to sit through the information theory bit.

Before we dive in

Marginalization

$$p(\mathbf{x}) = \sum_h p(\mathbf{x}, h)$$

Chain rule

$$p(\mathbf{x}, h) = p(\mathbf{x}|h)p(h) = p(\mathbf{x})p(h|\mathbf{x})$$

Bayes theorem

$$p(h|\mathbf{x}) = \frac{p(\mathbf{x}, h)}{p(\mathbf{x})}$$

Generative process for mixture models

Data is assumed to be generated from a preset number of components (say k):

1. randomly choose a mixture component
2. generate a data instance using the components' parameters

Probabilistic model

$$\begin{aligned} p(h = c) &= \pi_c \\ p(\mathbf{x}|h = c) &= f(\mathbf{x}|\theta_c) \end{aligned}$$

For example in the case of a Mixture-of-Gaussians model each mixture component would be a Gaussian

$$f(\mathbf{x}|\mu, \Sigma) \propto \exp\left\{-(1/2)(\mathbf{x} - \mu)'\Sigma^{-1}(\mathbf{x} - \mu)\right\}$$

Likelihood for a mixture model

We will use θ to denote the whole collection of parameters across all classes

$$\theta = (\underbrace{\theta_1}_{\text{class 1 params}}, \ldots \underbrace{\theta_k}_{\text{class k params}})$$

Then we can write likelihood

$$\begin{aligned}
\mathrm{LL}(\theta) = \sum_i \log p(\mathbf{x}_i) &= \sum_i \log \sum_{c=1}^{k} p(\mathbf{x}_i, h_i = c) \\
&= \sum_i \log \sum_{c=1}^{k} p(\mathbf{x}_i | h_i = c) p(h_i = c) \\
&= \sum_i \log \sum_{c=1}^{k} f(\mathbf{x}_i | \theta_c) \pi_c
\end{aligned}$$

Our goal is to maximize likelihood with respect to θ.

A manipulation

We will perform a small manipulation dividing and multiplying by a term[3]

$$
\begin{aligned}
\sum_i \log\left\{\sum_{h_i} p(\mathbf{x}_i|h_i)\pi_{h_i}\right\} &= \sum_i \log\left\{\sum_{h_i} \frac{q(h_i)}{q(h_i)} p(\mathbf{x}_i|h_i)\pi_{h_i}\right\} \\
&= \sum_i \log\left\{\sum_{h_i} \frac{q(h_i)}{q(h_i)} p(\mathbf{x}_i, h_i)\right\}
\end{aligned}
$$

To make derivation shorter, I will drop sum over instances $\sum_i$ and focus just on a single instance

$$
\log\left\{\sum_{h} \frac{q(h)}{q(h)} p(\mathbf{x}, h_i)\right\}
$$

[3] not unlike add 1 and substract 1 trick.

Applying Jensen's inequality

If $f(\mathbf{x})$ is a concave function then

$$f(\mathbf{E}[\mathbf{x}]) \geq \mathbf{E}[f(\mathbf{x})].$$

Here is a specialized version of Jensen's inequality and in this context f is a log which is a concave function.

$$\underbrace{\log\left\{\sum_h q(h)\frac{p(\mathbf{x},h)}{q(h)}\right\}}_{f\left(E\left[\frac{p(\mathbf{x},h)}{q(h)}\right]\right)} \geq \underbrace{\sum_h q(h)\log\left\{\frac{p(\mathbf{x},h)}{q(h)}\right\}}_{E\left[f\left(\frac{p(\mathbf{x},h)}{q(h)}\right)\right]}$$

When is the bound tight?

If we use $q(h) = p(h|\mathbf{x})$

$$\begin{aligned}
\log p(\mathbf{x}) = \log\left\{\sum_h q(h)\frac{p(\mathbf{x},h)}{q(h)}\right\} \quad &\geq \quad \sum_h p(h|\mathbf{x}) \log\left\{\frac{p(\mathbf{x},h)}{p(h|\mathbf{x})}\right\} \\
&= \quad \sum_h p(h|\mathbf{x}) \log p(\mathbf{x}) \\
&= \quad \log p(\mathbf{x})
\end{aligned}$$

so if $q(h)$ is equal to the true posterior $p(h|\mathbf{x})$ the bound is tight – the approximation is exact.

Upshot: we can convert a challenging log-sum term into a sum-log term and retain exactness if $q(h) = p(h|\mathbf{x})$

Likelihood lower bound optimization – recap

$$
\begin{aligned}
\mathrm{LL}(\theta) &= \sum_i \log\{p(\mathbf{x}_i, h_i)\} = \sum_i \log\left\{\sum_{h_i} q(h_i)\frac{p(\mathbf{x}_i, h_i|\theta)}{q(h_i)}\right\} \\
&\geq \sum_i \sum_{h_i} q(h_i) \log\left\{\frac{p(\mathbf{x}_i, h_i|\theta)}{q(h_i)}\right\} \\
&= \sum_i \sum_{h_i} q(h_i) \log\{p(\mathbf{x}_i, h_i|\theta)\} - \sum_i \sum_{h_i} q(h_i) \log\{q(h_i)\} \\
&= \mathcal{F}(q, \theta)
\end{aligned}
$$

Sometimes, this last expression is called *free energy*, thus $\mathcal{F}$.

Optimizing the lower bound

$$\mathrm{LL}(\theta) \geq \mathcal{F}(q, \theta)$$

We know that the bound is tight if $q(h_i) = p(h_i|\mathbf{x}, \theta)$ for all i.

So, in principle if θ gives a local maximum of $\mathcal{F}$ then it also gives a local maximum of $\mathrm{LL}(\theta)$.

Hence, we can proceed to optimize $\mathcal{F}$ with respect to q and θ.

Optimizing the lower bound with respect to q

$$
\begin{aligned}
& \operatorname*{argmax}_{q_i} \mathcal{F}(q, \theta) \\
= \quad & \operatorname*{argmax}_{q_i} \sum_h q_i(h) \log \{p(\mathbf{x}, h|\theta)\} - \sum_h q(h) \log \{q(h)\} \\
= \quad & \operatorname*{argmax}_{q_i} -\mathrm{KL}(q_i(h)||p(h|\mathbf{x}, \theta)) + \log \{p(\mathbf{x}|\theta)\} \\
= \quad & \operatorname*{argmin}_{q_i} \mathrm{KL}(q_i(h)||p(h|\mathbf{x}, \theta)) \\
= \quad & p(h_i|\mathbf{x}_i, \theta)
\end{aligned}
$$

Computing gradients with respect to θ

Gradient of $\mathcal{F}$ with respect to θ is easy to compute thanks to introduction of q

$$\nabla_\theta \mathcal{F}(q, \theta) = \sum_i \sum_{h_i} q(h_i)[\nabla \log p(\mathbf{x}, h_i|\theta)]$$

How to compute posterior $p(h|\mathbf{x})$ for a mixture model?

We want to be able to compute $q(h_i)$ in order to compute those gradients.

We opted to use the true posterior $p(h_i|\mathbf{x})$ in place of $q(h_i)$ but how is this computed?

$$
\begin{aligned}
p(h_i = m|\mathbf{x}_i, \theta) &= \frac{p(\mathbf{x}_i, h_i = m)}{p(\mathbf{x}_i)} \\
&= \frac{p(\mathbf{x}_i, h_i = m)}{\sum_c p(\mathbf{x}_i, h_i = c)} \\
&= \frac{p(\mathbf{x}_i|h_i = m)p(h_i = m)}{\sum_c p(\mathbf{x}_i|h_i = c)p(h_i = c)} \\
&= \frac{f(\mathbf{x}_i|\theta_m)\pi_m}{\sum_c f(\mathbf{x}_i|\theta_c)\pi_c}
\end{aligned}
$$

An algorithm

1. Compute $q(h_i) = p(h_i|\mathbf{x}_i)$ given some set of parameters θ
2. Solve for θ^{new} by equating $\sum_i \sum q(h_i)[\nabla \log p(\mathbf{x}, h_i)] = 0$

This is an Expectation Maximization algorithm.

The first part, the E-step, computes the posterior $p(h|\mathbf{x})$ (like we did on the last slide).

The second part, the M-step, performs maximization of $\mathbf{E}[\log p(\mathbf{x}_i, h_i)]$ with respect to parameters.

Mixture of Gaussians - E-step

The E-step is relatively simple: we evaluate the probability of the data point under each component $p(\mathbf{x}_i | h_i = c)$ and compile a matrix of probabilities

$$q(h_i = m) = \frac{f(\mathbf{x}_i | \theta_m)\pi_m}{\sum_c f(\mathbf{x}_i | \theta_c)\pi_c}$$

For a dataset with N instances and K classes q could be seen as a matrix of $N \times K$ probabilities.

It may not be necessary to store this matrix, however, since it is only used to compute expectations, and contributions from each instance can be collected incrementally.

Mixture of Gaussians - M-step for μ

$$\log\left\{p(\mathbf{x}_i|h_i=c)\right\} = -\frac{D}{2}\log\left\{2\pi\right\} - \frac{1}{2}\log|\Sigma_c| - \frac{1}{2}(\mathbf{x}_i-\mu)'\Sigma^{-1}(\mathbf{x}_i-\mu)$$

and further

$$\sum_i \sum_h q(h_i)\frac{\partial \log p(\mathbf{x}, h_i)}{\partial \mu_c} = \sum_i q(h_i = c)\Sigma^{-1}(\mathbf{x}_i - \mu)$$

and setting this to 0 yields

$$\mu_c^{\text{new}} = \frac{\sum_i q(h_i = c)\mathbf{x}_i}{\sum_i q(h_i = c)}$$

Mixture of Gaussians - M-step for Σ

$$\log\{p(\mathbf{x}_i|h_i=c)\} = -\frac{D}{2}\log\{2\pi\} - \frac{1}{2}\log|\Sigma_c| - \frac{1}{2}(\mathbf{x}_i-\mu)'\Sigma^{-1}(\mathbf{x}_i-\mu)$$

and

$$\sum_i\sum_h q(h_i)\frac{\partial \log p(\mathbf{x},h_i)}{\partial \Sigma_c^{-1}} = \sum_i q(h_i=c)(\Sigma_c - (\mathbf{x}_i-\mu_c^{\text{new}})(\mathbf{x}_i-\mu_c^{\text{new}})'$$

and setting this to 0 yields

$$\Sigma_c^{\text{new}} = \frac{\sum_i q(h_i=c)\mathbf{x}_i\mathbf{x}_i'}{\sum_i q(h_i=c)} - \mu_c^{\text{new}}\mu_c^{\text{new}\prime}$$

Mixture of Gaussians - M-step for π

The part of the lower bound on log likelihood that involves mixing proportions π_c is given by

$$\sum_i \sum_{h_i} q(h_i) \log \pi_{h_i}$$

but by definition π_c are probabilities and we cannot simply take derivatives and set them to zero.

Mixture of Gaussians - M-step

Instead we formulate a constrained optimization problem

$$\begin{array}{ll} \underset{\pi \in \mathbf{R}_K^+}{\text{maximize}} & \sum_i \sum_{h_i} q(h_i) \log \pi_{h_i} \\ \text{subject to} & \sum_c \pi_c = 1 \end{array}$$

and its Lagrangian is

$$L(\pi, \lambda) = \sum_i \sum_{h_i} q(h_i) \log \pi_{h_i} + \lambda(\sum_c \pi_c - 1)$$

at optimum the following conditions must hold

$$\begin{aligned} \sum_i \frac{q(h_i = c)}{\pi_c} + \lambda &= 0 \\ \sum_c \pi_c &= 1 \end{aligned}$$

Mixture of Gaussians - M-step for π

Solving these equations gives

$$\pi_c = \frac{\sum_i q(h_i = c)}{\sum_i \sum_{h_i} q(h_i)} = \frac{1}{N} \sum_i q(h_i = c)$$

Demo second look

A second look at the MoG demo and some code perusal.

We did ...

- A little information theory
- First EM for MoG
- A bit on practicalities of implementing EM