Recall: Missing Outputs

- Remember that you can think of unsupervised learning as supervised learning in which all the outputs are missing:
  - Clustering == classification with missing labels.
  - Dimensionality reduction == regression with missing targets.
- Density estimation is actually very general and encompasses the two problems above and a whole lot more.
- Today, let’s focus on the idea of missing (unobserved) variables...

Partially Unobserved Variables

- Certain variables $q$ in our models may be unobserved, either at training time or at test time or both.
- If the are occasionally unobserved they are missing data.
  e.g. undefined inputs, missing class labels, erroneous target values
- In this case, we define a new cost function in which we integrate out the missing values at training or test time:

  $$
  \ell(\theta; D) = \sum_{\text{complete}} \log p(x^c, y^c | \theta) + \sum_{\text{missing}} \log p(x^m | \theta)
  = \sum_{\text{complete}} \log p(x^c, y^c | \theta) + \sum_{\text{missing}} \log \sum_y p(x^m, y | \theta)
  $$

- Variables which are always unobserved are called latent variables or sometimes hidden variables.

Latent Variables

- What should we do when a variable $z$ is always unobserved?
  Depends on where it appears in our model. If we never condition on it when computing the probability of the variables we do observe, then we can just forget about it and integrate it out.
  e.g. given $y, x$ fit the model $p(z, y | x) = p(z | y)p(y | x, w)p(w)$.
- But if $z$ is conditioned on, we need to model it:
  e.g. given $y, x$ fit the model $p(y | x) = \sum_z p(y | x, z)p(z)$
- Latent variables may appear naturally, from the structure of the problem. But also, we may want to intentionally introduce latent variables to model complex dependencies between variables without looking at the dependencies between them directly.
- This can actually simplify the model, and the most common example of this is in mixture modeling.
Mixture Models

- Mixture models are the most basic possible latent variable model, having only a single discrete latent variable $z$.
- Idea: allow different submodels (experts) to contribute to the (conditional) density model in different parts of the space.
- Divide and conquer: use simple parts to build complex models. (e.g. multimodal densities, or piecewise-linear regressions).

Mixture Densities

- Exactly like a class-conditional model but the class is unobserved and so we sum it out. What we get is a perfectly valid density:
  \[ p(x|\theta) = \sum_k p(z = k|\theta_k) p(x|z = k, \theta_k) \]
  \[ = \sum_k \alpha_k p_k(x|\theta_k) \]
  where the “mixing proportions” add to one: $\sum_k \alpha_k = 1$.
- We can use Bayes’ rule to compute the posterior probability of the mixture component given some data:
  \[ r_k(x) = p(z = k|x, \theta) = \frac{\alpha_k p_k(x|\theta_k)}{\sum_j \alpha_j p_j(x|\theta_j)} \]
  these quantities are called responsibilities. You’ve seen them many times before; now you know their names!

Why is Learning Harder with Hidden Variables?

- In fully observed settings, the probability model is a product, thus the log likelihood is a sum where terms decouple.
  \[ \ell(\theta; D) = \sum_n \log p(y_n, x_n|\theta) \]
  \[ = \sum_n \log p(x_n|\theta_x) + \sum_n \log p(y_n|x_n, \theta_y) \]
- With latent variables, the probability already contains a sum, so the log likelihood has all parameters coupled together:
  \[ \ell(\theta; D) = \sum_n \log \sum_z p(x_n, z|\theta) \]
  \[ = \sum_n \log \sum_z p(z|\theta_z) p(x_n|z, \theta_x) \]

Learning with Latent Variables

- Likelihood $\ell(\theta) = \log \sum_z p(z|\theta_z) p(x|z, \theta_x)$ couples parameters:
- We can treat this as a black box probability function and just try to optimize the likelihood as a function of $\theta$.
  We did this many times before by taking gradients.
- Remember Mixtures of Experts...?
- However, sometimes taking advantage of the latent variable structure can make parameter estimation easier.
- Good news: today we will see the EM algorithm which allows us to treat learning with latent variables using fully observed tools.
- Basic trick: guess the values you don’t know. Basic math: use convexity to lower bound the likelihood.
We can learn mixture densities using gradient descent on the likelihood as usual. The gradients are quite interesting:

\[ \ell(\theta) = \log p(x|\theta) = \log \sum_k \alpha_k p_k(x|\theta_k) \]

\[ \frac{\partial \ell}{\partial \theta} = \frac{1}{p(x|\theta)} \sum_k \alpha_k \frac{\partial p_k(x|\theta_k)}{\partial \theta} = \sum_k \alpha_k \frac{1}{p(x|\theta)} p_k(x|\theta_k) \frac{\partial \log p_k(x|\theta_k)}{\partial \theta} = \sum_k \alpha_k \frac{p_k(x|\theta_k)}{p(x|\theta)} \frac{\partial \ell}{\partial \theta} = \sum_k \alpha_k r_k \frac{\partial \ell}{\partial \theta} \]

In other words, the gradient is the responsibility weighted sum of the individual log likelihood gradients. (cf. MOEs)

Clustering Example: Gaussian Mixture Models

Consider a mixture of K Gaussian components:

\[ p(x|\theta) = \sum_k \alpha_k \mathcal{N}(x|\mu_k, \Sigma_k) \]

\[ p(z = k|x, \theta) = \frac{\alpha_k \mathcal{N}(x|\mu_k, \Sigma_k)}{\sum_j \alpha_j \mathcal{N}(x|\mu_j, \Sigma_j)} \]

\[ \ell(\theta; D) = -\sum_n \log \sum_k \alpha_k \mathcal{N}(x^n|\mu_k, \Sigma_k) \]

Density model: \( p(x|\theta) \) is a familiarity signal.
Clustering: \( p(z|x, \theta) \) is the assignment rule, \( -\ell(\theta) \) is the cost.

An Aside: Parameter Constraints

If we want to use general optimizations (e.g. conjugate gradient) to learn latent variable models, we often have to make sure parameters respect certain constraints. (e.g. \( \sum_k \alpha_k = 1, \Sigma_k \) pos.definite).

A good trick is to reparameterize these quantities in terms of unconstrained values. For mixing proportions, use the softmax:

\[ \alpha_k = \frac{\exp(q_k)}{\sum_j \exp(q_j)} \]

For covariance matrices, use the Cholesky decomposition:

\[ \Sigma^{-1} = A^T A \]

\[ |\Sigma|^{-1/2} = \prod_i A_{ii} \]

where \( A \) is upper diagonal with positive diagonal:

\[ A_{ii} = \exp(r_i) > 0 \quad A_{ij} = a_{ij} \quad (j > i) \quad A_{ij} = 0 \quad (j < i) \]
Conditional Mixtures: MOEs Revisited

- Mixtures of Experts are also called conditional mixtures. Exactly like a class-conditional classification model, except the class is unobserved and so we sum it out:

\[ p(y|x, \theta) = \sum_{k=1}^{K} \alpha_k(x|\theta_k)p_k(y|x, \theta_k) \]

- Harder: must learn \( \alpha_k(x) \) (unless chose \( z \) independent of \( x \)). The \( \alpha_k(x) \) are exactly what we called the gating function.

- We can still use Bayes’ rule to compute the posterior probability of the mixture component given some data:

\[ p(z = k|x, y, \theta) = \frac{\alpha_k(x|\theta_k)p_k(y|x, \theta_k)}{\sum_{j} \alpha_j(x)p_j(y|x, \theta_j)} \]

Recap: Learning with Latent Variables

- With latent variables, the probability contains a sum, so the log likelihood has all parameters coupled together:

\[ \ell(\theta; D) = \log \sum_z p(x, z|\theta) = \log \sum_z p(z|\theta)p(x|z, \theta_x) \]

- If the latent variables were observed, parameters would decouple again and learning would be easy:

\[ \ell(\theta; D) = \log p(x, z|\theta) = \log p(z|\theta) + \log p(x|z, \theta_x) \]

An Idea about Mixtures of Experts...

- What if instead of using the gradient to adjust MOE parameters, we just used the posterior weightings to “softly label” the data.

- Then we could solved a (weighted) least-squares problem for each expert and a (soft-target) logistic-regression problem for the gate.

- Both of these problems are convex (even with weighted data), so we can solve them exactly without needing to do any gradient descent.

- Then we could alternate: find the “optimum” parameters given the current posterior weightings and then recalculate the posteriors (weights) given the new parameters, and repeat.

- We will explore this idea today. It is called Expectation Maximization or EM, and is a form of bound optimization as opposed to gradient methods.

Expectation-Maximization (EM) Algorithm

- Iterative algorithm with two linked steps:

  - **E-step**: fill in values of \( z_t^l \) using \( p(z|x, \theta_t^l) \).

  - **M-step**: update parameters using \( \theta_t^{l+1} \leftarrow \text{argmax} \ell(\theta; x, z_t^l) \).

- E-step involves inference, which we need to do at runtime anyway. M-step is no harder than in fully observed case.

- We will prove that this procedure monotonically improves \( \ell \) (or leaves it unchanged). Thus it always converges to a local optimum of the likelihood (as any optimizer should).

- Note: EM is an optimization strategy for objective functions that can be interpreted as likelihoods in the presence of missing data.

- EM is not a cost function such as “maximum-likelihood”. EM is not a model such as “mixture-of-Gaussians”. 
Complete & Incomplete Log Likelihoods

- Observed variables \( x \), latent variables \( z \), parameters \( \theta \):
  \[
  \ell_c(\theta; x, z) = \log p(x, z|\theta)
  \]
is the complete log likelihood.

- Usually optimizing \( \ell_c(\theta) \) given both \( z \) and \( x \) is straightforward.
  (e.g. class conditional Gaussian fitting, linear regression)

- With \( z \) unobserved, we need the log of a marginal probability:
  \[
  \ell(\theta; x) = \log p(x|\theta) = \log \sum_z p(x, z|\theta)
  \]
  which is the incomplete log likelihood.

Expected Complete Log Likelihood

- For any distribution \( q(z) \) define expected complete log likelihood:
  \[
  \ell_q(\theta; x) = \langle \ell_c(\theta; x, z) \rangle_q = \sum_z q(z|x) \log p(x, z|\theta)
  \]

- Amazing fact: \( \ell(\theta) \geq \ell_q(\theta) + \mathcal{H}(q) \) because of concavity of log:
  \[
  \ell(\theta; x) = \log p(x|\theta) = \log \sum_z p(x, z|\theta)
  = \log \sum_z q(z|x) \frac{p(x, z|\theta)}{q(z|x)}
  \geq \sum_z q(z|x) \log \frac{p(x, z|\theta)}{q(z|x)}
  \]
  \[
  \geq \sum_z q(z|x) \log \frac{p(x, z|\theta)}{q(z|x)}
  \]
  where the inequality is called Jensen’s inequality.
  (It is only true for distributions: \( \sum q(z) = 1 \); \( q(z) > 0 \).)

Lower Bounds and Free Energy

- For fixed data \( x \), define a functional called the free energy:
  \[
  F(q, \theta) = \sum_z q(z|x) \log \frac{p(x, z|\theta)}{q(z|x)} \leq \ell(\theta)
  \]

- The EM algorithm is coordinate-ascent on \( F \):
  
  **E-step:** \( q^{t+1} = \arg\max_q F(q, \theta^t) \)
  
  **M-step:** \( \theta^{t+1} = \arg\max_\theta F(q^{t+1}, \theta^t) \)

M-step: maximization of expected \( \ell_c \)

- Note that the free energy breaks into two terms:
  \[
  F(q, \theta) = \sum_z q(z|x) \log \frac{p(x, z|\theta)}{q(z|x)}
  = \sum_z q(z|x) \log p(x, z|\theta) - \sum_z q(z|x) \log q(z|x)
  = \ell_q(\theta; x) + \mathcal{H}(q) = \text{average energy + entropy}
  \]
  (this is where its name comes from)
  
- The first term is the expected complete log likelihood (energy) and the second term, which does not depend on \( \theta \), is the entropy.
  
- Thus, in the M-step, maximizing with respect to \( \theta \) for fixed \( q \) we only need to consider the first term:
  \[
  \theta^{t+1} = \arg\max_\theta \ell_q(\theta; x) = \arg\max_\theta \sum_z q(z|x) \log p(x, z|\theta)
  \]
E-step: inferring latent posterior

- Claim: the optimum setting of \( q \) in the E-step is:
  \[
  q^{t+1} = p(z|x, \theta^t)
  \]
- This is the posterior distribution over the latent variables given the data and the parameters. Often we need this at test time anyway (e.g. to perform classification).
- Proof (easy): this setting saturates the bound \( \ell(\theta; x) \geq F(q, \theta) \)
  \[
  F(p(z|x, \theta^t), \theta^t) = \sum_z p(z|x, \theta^t) \log \frac{p(x, z|\theta^t)}{p(z|x, \theta^t)}
  = \sum_z p(z|x, \theta^t) \log p(x|\theta^t)
  = \log p(x|\theta^t) \sum_z p(z|x, \theta^t)
  = \ell(\theta; x) \cdot 1
  \]
- Can also show this result using variational calculus or the fact that \( \ell(\theta) - F(q, \theta) = \text{KL}[q||p(z|x, \theta)] \)

EM Constructs Sequential Convex Lower Bounds

- Consider the likelihood function and the function \( F(q^{t+1}, \cdot) \).

Be careful: logsum

- Often you can easily compute \( b_k = \log p(x|z = k, \theta_k) \), but it will be very negative, say \(-10^6\) or smaller.
- Now, to compute \( \ell = \log p(x|\theta) \) you need to compute \( \log \sum_k e^{b_k} \).
- Careful! Do not compute this by doing \( \log(\text{sum}(\exp(b))) \).
  You will get underflow and an incorrect answer.
- Instead do this:
  - Add a constant exponent \( B \) to all the values \( b_k \) such that the largest value comes close to the maximum exponent allowed by machine precision: \( B = \text{MAXEXponent} - \log(K) - \max(b) \).
  - Compute \( \log(\text{sum}(\exp(b+B)))-B \).

Example: Mixtures of Gaussians

- Recall: a mixture of \( K \) Gaussians:
  \[
  p(x|\theta) = \sum_k \alpha_k N(x|m_k, \Sigma_k)
  \]
  \[
  \ell(\theta; D) = \sum_n \log \sum_k \alpha_k N(x^n|m_k, \Sigma_k)
  \]
- Learning with EM algorithm:
  \[
  \begin{align*}
  &\textbf{E} - \text{step} : \\
  &p_{kn} = N(x^n|m_k, \Sigma_k) \\
  &q_{kn}^{t+1} = p(z = k|x^n, \theta^t) = \frac{\alpha_k p_{kn}^t}{\sum_j \alpha_j p_{jn}^t}
  \\
  &\textbf{M} - \text{step} : \\
  &\mu_k^{t+1} = \frac{\sum_n q_{kn}^{t+1} x^n}{\sum_n q_{kn}^{t+1}} \\
  &\Sigma_k^{t+1} = \frac{\sum_n q_{kn}^{t+1} (x^n - \mu_k^{t+1})(x^n - \mu_k^{t+1})^T}{\sum_n q_{kn}^{t+1}}
  \\
  &\alpha_k^{t+1} = \frac{1}{N} \sum_n q_{kn}^{t+1}
  \end{align*}
  \]
**Derivation of M-step**

- Expected complete log likelihood $\ell_q(\theta; D)$:

$$\sum_n \sum_k q_{kn} \left[ \log \alpha_k - \frac{1}{2} (x^n - \mu_k^{t+1})^\top \Sigma_k^{-1} (x^n - \mu_k^{t+1}) - \frac{1}{2} \log |2\pi \Sigma_k| \right]$$

- For fixed $q$ we can optimize the parameters:

$$\frac{\partial \ell_q}{\partial \mu_k} = \Sigma_k^{-1} \sum_n q_{kn} (x^n - \mu_k)$$

$$\frac{\partial \ell_q}{\partial \Sigma_k} = \frac{1}{2} \sum_n q_{kn} \left[ \Sigma_k^\top - (x^n - \mu_k^{t+1})(x^n - \mu_k^{t+1})^\top \right]$$

$$\frac{\partial \ell_q}{\partial \alpha_k} = \frac{1}{\alpha_k} \sum_n q_{kn} - \lambda \quad (\lambda = N)$$

- Fact: $\frac{\partial \log |A^{-1}|}{\partial A} = A^\top$ and $\frac{\partial x^\top A x}{\partial A} = xx^\top$

**Recap: EM Algorithm**

- A way of maximizing likelihood function for latent variable models. Finds ML parameters when the original (hard) problem can be broken up into two (easy) pieces:

1. Estimate some “missing” or “unobserved” data from observed data and current parameters.
2. Using this “complete” data, find the maximum likelihood parameter estimates.

- Alternate between filling in the latent variables using our best guess (posterior) and updating the parameters based on this guess:

**E-step:**

$$q_k^{t+1} = \frac{\exp(-d_{kn}^t)}{\sum_j \exp(-d_{jn}^t)} = p(c_n^t = k | x_n^t, \mu_t^t)$$

**M-step:**

$$\mu_k^{t+1} = \frac{\sum_n q_k^{t+1} x_n}{\sum_n q_k^{t+1}}$$

- In the M-step we optimize a lower bound on the likelihood. In the E-step we close the gap, making bound=likelihood.

**Compare: K-Means**

- The EM algorithm for mixtures of Gaussians is just like a soft version of the K-means algorithm with fixed priors and covariance.
- Instead of “hard assignment” in the E-step, we do “soft assignment” based on the softmax of the squared distance from each point to each cluster.
- Each centre is then moved to the weighted mean of the data, with weights given by soft assignments. In K-means, the weights are 0 or 1.

**E-step:**

$$d_{kn}^t = \frac{1}{2} (x^n - \mu_k^t)^\top \Sigma_k^{-1} (x^n - \mu_k^t)$$

$$q_k^{t+1} = \frac{\exp(-d_{kn}^t)}{\sum_j \exp(-d_{jn}^t)} = p(c_n^t = k | x_n^t, \mu_t^t)$$

**M-step:**

$$\mu_k^{t+1} = \frac{\sum_n q_k^{t+1} x_n}{\sum_n q_k^{t+1}}$$
A Report Card for EM

• Some good things about EM:
  – no learning rate parameter
  – very fast for low dimensions
  – each iteration guaranteed to improve likelihood
  – adapts unused units rapidly

• Some bad things about EM:
  – can get stuck in local minima
  – both steps require considering all explanations of the data which is an exponential amount of work in the dimension of $\theta$

• EM is typically used with mixture models, for example mixtures of Gaussians or mixtures of experts. The “missing” data are the labels showing which sub-model generated each datapoint.
  Very common: also used to train HMMs, Boltzmann machines,...

Partially Hidden Data

• Of course, we can learn when there are missing (hidden) variables on some cases and not on others.

• In this case the cost function was:

$$
\ell(\theta; D) = \sum_{\text{complete}} \log p(x^c, y^c|\theta) + \sum_{\text{missing}} \log \sum_{y} \log p(x^m, y|\theta)
$$

• Now you can think of this in a new way: in the E-step we estimate the hidden variables on the incomplete cases only.

• The M-step optimizes the log likelihood on the complete data plus the expected likelihood on the incomplete data using the E-step.

Variants

• Sparse EM:
  Do not recompute exactly the posterior probability on each data point under all models, because it is almost zero.
  Instead keep an “active list” which you update every once in a while.

• Generalized (Incomplete) EM: It might be hard to find the ML parameters in the M-step, even given the completed data. We can still make progress by doing an M-step that improves the likelihood a bit (e.g. gradient step).