Lecture 2a:

Latent Variables Models and Learning with the EM Algorithm

Sam Roweis

Thursday August 17, 2006
CIAR Summer School, Toronto

Partially Unobserved (Missing) Variables

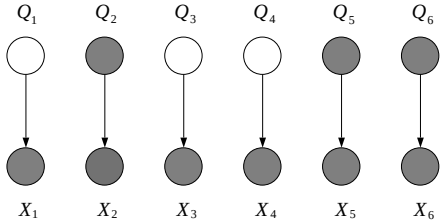
- If variables are occasionally unobserved they are *missing data*.
  e.g. undefinied inputs, missing class labels, erroneous target values
- In this case, we can still model the joint distribution, but we define a new cost function in which we *sum out* or *marginalize* the missing values at training or test time:

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

[Recall that $p(x) = \sum_q p(x,q)$.]

Unobserved Variables

- We have been assuming that we observe all the random variables in our model at training time, and all the "inputs" at test time.
- But certain variables Q in our models may be *unobserved*, either some of the time or always, either at training time or at test time.



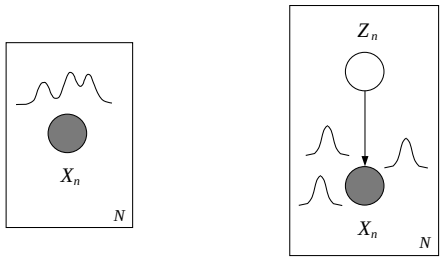
(Graphically, we will use shading to indicate observation.)

Latent Variable Models

- What to do when a variable $\mathbf{z}$ 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 $\mathbf{y}, \mathbf{x}$ fit the model $p(\mathbf{z}, \mathbf{y}|\mathbf{x}) = p(\mathbf{z}|\mathbf{y})p(\mathbf{y}|\mathbf{x}, \mathbf{w})p(\mathbf{w})$.
  (In other words if it is a leaf node.)
- But if $\mathbf{z}$ is conditioned on, we need to model it:
  e.g. given $\mathbf{y}, \mathbf{x}$ fit the model $p(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{y}|\mathbf{x}, \mathbf{z})p(\mathbf{z})$

Where Do Latent Variables Come From?

- Latent variables may appear naturally, from the structure of the problem, because something wasn't measured, because of faulty sensors, occlusion, privacy, etc.
- 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 (e.g. mixtures).



(a) (b)

Why is Learning Harder?

- In fully observed iid settings, the probability model is a product thus the log likelihood is a sum where terms decouple.
  (At least for directed models.)

$$\begin{aligned}\ell(\theta;\mathcal{D}) &= \log p(\mathbf{x},\mathbf{z}|\theta) \\ &= \log p(\mathbf{z}|\theta_z) + \log p(\mathbf{x}|\mathbf{z},\theta_x)\end{aligned}$$

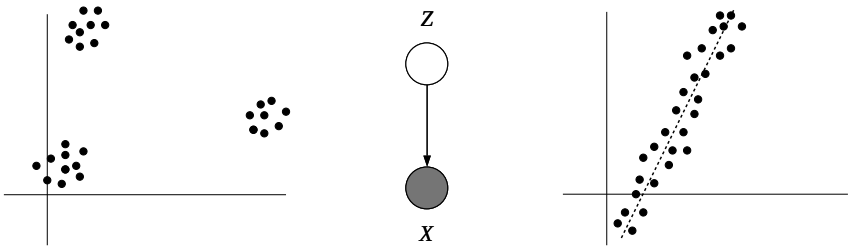
- With latent variables, the probability already contains a sum, so the log likelihood has all parameters coupled together via $\log\sum()$:

$$\begin{aligned}\ell(\theta;\mathcal{D}) &= \log \sum_{\mathbf{z}} p(\mathbf{x},\mathbf{z}|\theta) \\ &= \log \sum_{\mathbf{z}} p(\mathbf{z}|\theta_z)p(\mathbf{x}|\mathbf{z},\theta_x)\end{aligned}$$

  (Just as with the partition function in undirected models.)

Clustering vs. Classification
Latent Factor Models vs. Regression

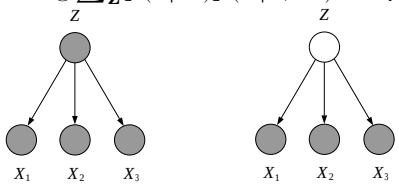
- You can think of clustering as the problem of classification with missing class labels.



- You can think of factor models (such as factor analysis, PCA, ICA, etc.) as linear or nonlinear regression with missing inputs.

Learning with Latent Variables

- Likelihood $\ell(\theta;\mathcal{D}) = \log\sum_{\mathbf{z}} p(\mathbf{z}|\theta_z)p(\mathbf{x}|\mathbf{z},\theta_x)$ couples parameters:

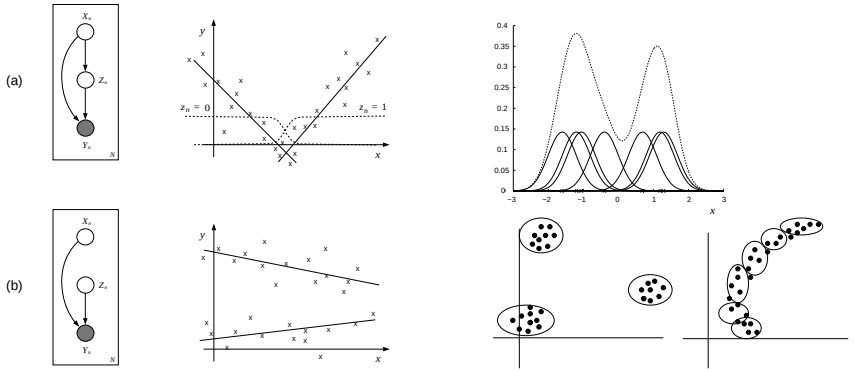


(a) (b)

- We can treat this as a black box probability function and just try to optimize the likelihood as a function of θ (e.g. gradient descent). However, sometimes taking advantage of the latent variable structure can make parameter estimation easier.
- Good news: soon 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.

Mixture Models (1 discrete latent var)

- Most basic latent variable model with a single discrete node z.
- Allows different submodels (experts) to contribute to the (conditional) density model in different parts of the space.
- Divide and conquer idea: use simple parts to build complex models. (e.g. multimodal densities, or piecewise-linear regressions).



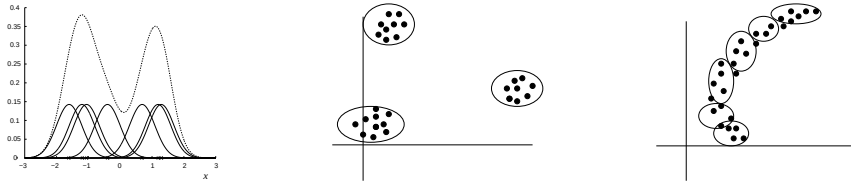
Clustering Example: Gaussian Mixture Models

- Consider a mixture of K Gaussian components:

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

$$p(z = k|\mathbf{x}, \theta) = \frac{\alpha_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)}{\sum_j \alpha_j \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)}$$

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



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

Mixture Densities

- Exactly like a classification model but the class is unobserved and so we sum it out. What we get is a perfectly valid density:

$$p(\mathbf{x}|\theta) = \sum_{k=1}^{K} p(z = k|\theta_z) p(\mathbf{x}|z = k, \theta_k)$$
$$= \sum_k \alpha_k p_k(\mathbf{x}|\theta_k)$$

  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:

$$p(z = k|\mathbf{x}, \theta) = \frac{\alpha_k p_k(\mathbf{x}|\theta_k)}{\sum_j \alpha_j p_j(\mathbf{x}|\theta_j)}$$

  these quantities are sometimes called the *responsibilities*.

Example: Mixture of Linear Regression Experts

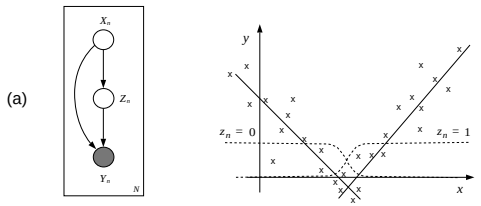
- Each expert generates data according to a linear function of the input plus additive Gaussian noise:

$$p(y|\mathbf{x}, \theta) = \sum_k \alpha_k(\mathbf{x}) \mathcal{N}(y|\beta_k^\top \mathbf{x}, \sigma_k^2)$$

- The "gating function" can be a softmax classification machine:

$$\alpha_k(\mathbf{x}) = p(z = k|\mathbf{x}) = \frac{e^{\eta_k^\top \mathbf{x}}}{\sum_j e^{\eta_j^\top \mathbf{x}}}$$

- Remember: we are not modeling the density of the inputs $\mathbf{x}$.



Gradient Learning with Mixtures

- We can learn mixture densities using gradient descent on the likelihood as usual. The gradients are quite interesting:

$$\ell(\theta) = \log p(\mathbf{x}|\theta) = \log \sum_k \alpha_k p_k(\mathbf{x}|\theta_k)$$

$$\frac{\partial \ell}{\partial \theta} = \frac{1}{p(\mathbf{x}|\theta)} \sum_k \alpha_k \frac{\partial p_k(\mathbf{x}|\theta_k)}{\partial \theta}$$

$$= \sum_k \alpha_k \frac{1}{p(\mathbf{x}|\theta)} p_k(\mathbf{x}|\theta_k) \frac{\partial \log p_k(\mathbf{x}|\theta_k)}{\partial \theta}$$

$$= \sum_k \alpha_k \frac{p_k(\mathbf{x}|\theta_k)}{p(\mathbf{x}|\theta)} \frac{\partial \ell_k}{\partial \theta_k} = \sum_k \alpha_k r_k \frac{\partial \ell_k}{\partial \theta_k}$$

- In other words, the gradient is the *responsibility weighted sum* of the individual log likelihood gradients.

Expectation-Maximization (EM) Algorithm

- Iterative algorithm with two linked steps:
  E-step: fill in values of $\hat{\mathbf{z}}^t$ using $p(\mathbf{z}|\mathbf{x}, \theta^t)$.
  M-step: update parameters using $\theta^{t+1} \leftarrow \operatorname{argmax} \ell(\theta; \mathbf{x}, \hat{\mathbf{z}}^t)$.
- 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 ℓ (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".

Recap: Learning with Latent Variables

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

$$\ell(\theta; \mathcal{D}) = \log \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}|\theta) = \log \sum_{\mathbf{z}} p(\mathbf{z}|\theta_z) p(\mathbf{x}|\mathbf{z}, \theta_x)$$

  (we can also consider continuous $\mathbf{z}$ and replace $\sum$ with $\int$)
- If the latent variables were observed, parameters would decouple again and learning would be easy:

$$\ell(\theta; \mathcal{D}) = \log p(\mathbf{x}, \mathbf{z}|\theta) = \log p(\mathbf{z}|\theta_z) + \log p(\mathbf{x}|\mathbf{z}, \theta_x)$$

- One idea: ignore this fact, compute $\partial \ell / \partial \theta$, and do learning with a smart optimizer like conjugate gradient.
- Another idea: what if we use our current parameters to *guess* the values of the latent variables, and then do fully-observed learning? This back-and-forth trick might make optimization easier.

Expected Complete Log Likelihood

- For *any* distribution $q(\mathbf{z})$ define *expected complete log likelihood*:

$$\ell_q(\theta; \mathbf{x}) = \langle \ell_c(\theta; \mathbf{x}, \mathbf{z}) \rangle_q \equiv \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log p(\mathbf{x}, \mathbf{z}|\theta)$$

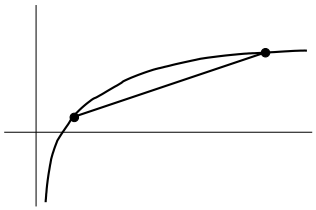
- Amazing fact: $\ell(\theta) \geq \ell_q(\theta) + \mathcal{H}(q)$ because of concavity of $\log$:

$$\ell(\theta; \mathbf{x}) = \log p(\mathbf{x}|\theta)$$

$$= \log \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}|\theta)$$

$$= \log \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \frac{p(\mathbf{x}, \mathbf{z}|\theta)}{q(\mathbf{z}|\mathbf{x})}$$

$$\geq \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log \frac{p(\mathbf{x}, \mathbf{z}|\theta)}{q(\mathbf{z}|\mathbf{x})}$$

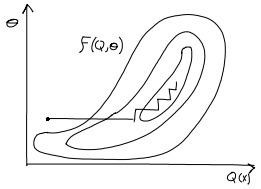


- Where the inequality is called *Jensen's inequality*. (It is only true for distributions: $\sum q(\mathbf{z}) = 1;\ q(\mathbf{z}) > 0$.)

Lower Bounds and Free Energy

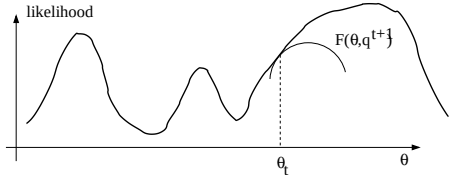
- For fixed data $\mathbf{x}$, define a functional called the *free energy*:

$$F(q,\theta) \equiv \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log \frac{p(\mathbf{x},\mathbf{z}|\theta)}{q(\mathbf{z}|\mathbf{x})} \quad \leq \ell(\theta)$$



- The EM algorithm is coordinate-ascent on F:
  E-step: $q^{t+1} = \text{argmax}_q \quad F(q,\theta^t)$
  M-step: $\theta^{t+1} = \text{argmax}_\theta \quad F(q^{t+1},\theta^t)$

- EM Constructs Sequential Convex Lower Bounds



E-step: inferring latent posterior

- Claim: the optimum setting of q in the E-step is:

$$q^{t+1} = p(\mathbf{z}|\mathbf{x},\theta^t)$$

- 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 $\ell(\theta;\mathbf{x}) \geq F(q,\theta)$

$$\begin{aligned} F(p(\mathbf{z}|\mathbf{x},\theta^t),\theta^t) &= \sum_{\mathbf{z}} p(\mathbf{z}|\mathbf{x},\theta^t) \log \frac{p(\mathbf{x},\mathbf{z}|\theta^t)}{p(\mathbf{z}|\mathbf{x},\theta^t)} \\ &= \sum_{\mathbf{z}} p(\mathbf{z}|\mathbf{x},\theta^t) \log p(\mathbf{x}|\theta^t) \\ &= \log p(\mathbf{x}|\theta^t) \textstyle\sum_{\mathbf{z}} p(\mathbf{z}|\mathbf{x},\theta^t) \\ &= \ell(\theta;\mathbf{x}) \cdot 1 \end{aligned}$$

- Can also show this result using variational calculus or the fact that $\ell(\theta) - F(q,\theta) = \text{KL}[q||p(\mathbf{z}|\mathbf{x},\theta)]$

M-step: maximization of expected ℓ_c

- Note that the free energy breaks into two terms:

$$\begin{aligned} F(q,\theta) &= \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log p(\mathbf{x},\mathbf{z}|\theta) - \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log q(\mathbf{z}|\mathbf{x}) \\ &= \ell_q(\theta;\mathbf{x}) + \mathcal{H}(q) \end{aligned}$$

  (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 θ, is the entropy.
- Thus, in the M-step, maximizing with respect to θ for fixed q we only need to consider the first term:

$$\theta^{t+1} = \text{argmax}_\theta \, \ell_q(\theta;\mathbf{x}) = \text{argmax}_\theta \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log p(\mathbf{x},\mathbf{z}|\theta)$$

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

Example: Mixtures of Gaussians

- Recall: a mixture of K Gaussians:
  $p(\mathbf{x}|\theta) = \sum_k \alpha_k \mathcal{N}(\mathbf{x}|\mu_k,\Sigma_k)$
  $\ell(\theta;\mathcal{D}) = \sum_n \log \sum_k \alpha_k \mathcal{N}(\mathbf{x}^n|\mu_k,\Sigma_k)$
- Learning with EM algorithm:

$$\mathbf{E-step}: \quad p_{kn}^t = \mathcal{N}(\mathbf{x}^n|\mu_k^t,\Sigma_k^t)$$

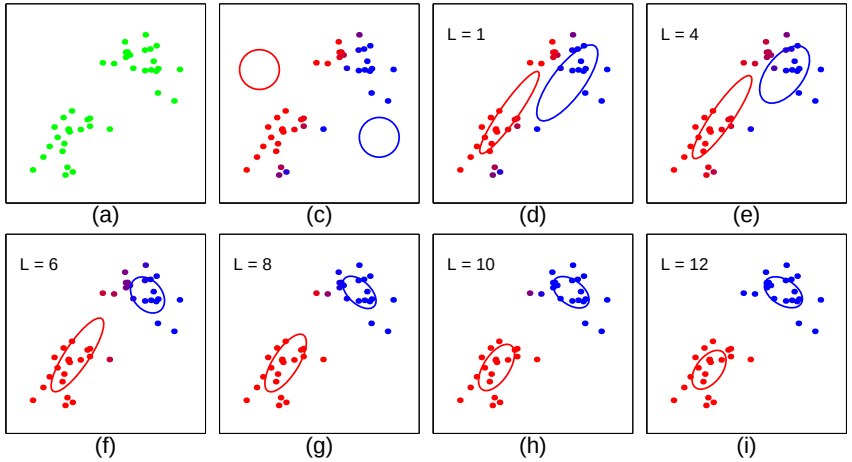
$$q_{kn}^{t+1} = p(z=k|\mathbf{x}^n,\theta^t) = \frac{\alpha_k^t p_{kn}^t}{\sum_j \alpha_j^t p_{kn}^t}$$

$$\mathbf{M-step}: \quad \mu_k^{t+1} = \frac{\sum_n q_{kn}^{t+1} \mathbf{x}^n}{\sum_n q_{kn}^{t+1}}$$

$$\Sigma_k^{t+1} = \frac{\sum_n q_{kn}^{t+1} (\mathbf{x}^n - \mu_k^{t+1})(\mathbf{x}^n - \mu_k^{t+1})^\top}{\sum_n q_{kn}^{t+1}}$$

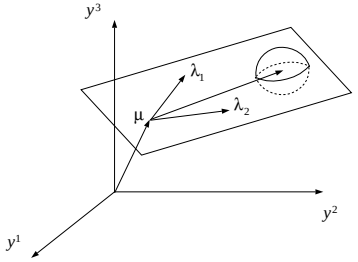
$$\alpha_k^{t+1} = \frac{1}{M} \sum_n q_{kn}^{t+1}$$

EM for MOG



Continuous Latent Variables

- In many models there are some *underlying causes* of the data.
- Mixture models use a discrete class variable: clustering.
- Sometimes, it is more appropriate to think in terms of continuous *factors* which control the data we observe. Geometrically, this is equivalent to thinking of a data *manifold* or subspace.



- To generate data, first generate a point within the manifold then add noise. Coordinates of point are components of latent variable.

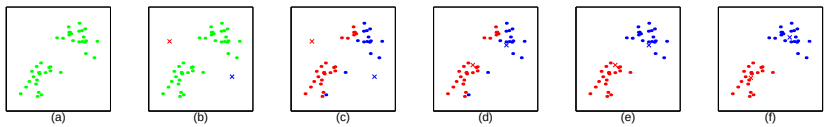
Compare: K-means

- The EM algorithm for mixtures of Gaussians is just like a soft version of the K-means algorithm.
- In the K-means "E-step" we do hard assignment:

$$c_n^{t+1} = \text{argmin}_k (\mathbf{x}^n - \mu_k^t)^\top \Sigma_k^{-1} (\mathbf{x}^n - \mu_k^t)$$

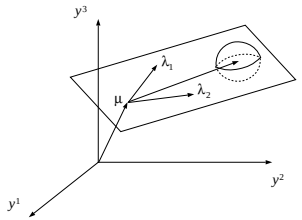
- In the K-means "M-step" we update the means as the weighted sum of the data, but now the weights are 0 or 1:

$$\mu_k^{t+1} = \frac{\sum_n [c_k^{t+1} = n] \mathbf{x}^n}{\sum_n [c_k^{t+1} = n]}$$



Factor Analysis

- When we assume that the subspace is *linear* and that the underlying latent variable has a Gaussian distribution we get a model known as *factor analysis*:
  — data $\mathbf{y}$ (p-dim);
  — latent variable $\mathbf{x}$ (k-dim)



$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|0, I)$$
$$p(\mathbf{y}|\mathbf{x}, \theta) = \mathcal{N}(\mathbf{y}|\mu + \Lambda\mathbf{x}, \Psi)$$

where μ is the mean vector, Λ is the p by k *factor loading matrix*, and Ψ is the *sensor noise covariance* (ususally diagonal).

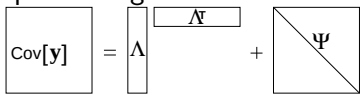
- Important: since the product of Gaussians is still Gaussian, the joint distribution $p(\mathbf{x}, \mathbf{y})$, the other marginal $p(\mathbf{y})$ and the conditional $p(\mathbf{x}|\mathbf{y})$ are also Gaussian.

FA = Constrained Covariance Gaussian

- Marginal density for factor analysis ($\mathbf{y}$ is p-dim, $\mathbf{x}$ is k-dim):

$$p(\mathbf{y}|\theta) = \mathcal{N}(\mathbf{y}|\mu\ ,\ \Lambda\Lambda^\top + \Psi) \quad \text{(integrate out } \mathbf{x})$$

- So the effective covariance is the low-rank outer product of two long skinny matrices plus a diagonal matrix:



- In other words, factor analysis is just a constrained Gaussian model. (If Ψ were not diagonal then we could model any Gaussian and it would be pointless.)
- Learning: how should we fit the ML parameters?
- It is easy to find μ: just take the mean of the data.
  From now on assume we have done this and re-centred $\mathbf{y}$.
- What about the other parameters?

EM for Factor Analysis

- We will do maximum likelihood learning using the EM algorithm.
  E-step: $q_n^{t+1} = p(\mathbf{x}^n|\mathbf{y}^n, \theta^t)$
  M-step: $\theta^{t+1} = \text{argmax}_\theta \sum_n \int_{\mathbf{x}} q^{t+1}(\mathbf{x}^n|\mathbf{y}^n) \log p(\mathbf{y}^n, \mathbf{x}^n|\theta) d\mathbf{x}^n$
- For E-step we need the conditional distribution (inference)
  For M-step we need the expected log of the complete data.

$$\begin{aligned}
\mathbf{E-step}: q_n^{t+1} &= p(\mathbf{x}^n|\mathbf{y}^n, \theta^t) = \mathcal{N}(\mathbf{x}^n|\mathbf{m}^n, \mathbf{V}^n) \\
\mathbf{M-step}: \Lambda^{t+1} &= \text{argmax}_\Lambda \sum_n \langle \ell_c(\mathbf{x}^n, \mathbf{y}^n) \rangle_{q_n^{t+1}} \\
\Psi^{t+1} &= \text{argmax}_\Psi \sum_n \langle \ell_c(\mathbf{x}^n, \mathbf{y}^n) \rangle_{q_n^{t+1}}
\end{aligned}$$

- Inferring the posterior mean is just a linear operation, and theposterior covariance does not depend on observed data:

$$\mathbf{m}^n = \beta(\mathbf{y}^n - \mu) \qquad \mathbf{V} = (I + \Lambda^\top \Psi^{-1} \Lambda)^{-1}$$

  where β can be computed beforehand given the model parameters.

EM Algorithm for Factor Analysis

- First, set μ equal to the sample mean $(1/N)\sum_n \mathbf{y}_n$, and subtract this mean from all the data.
- Now run the following iterations:

$$\begin{aligned}
\mathbf{E-step}: q^{t+1} &= p(\mathbf{x}|\mathbf{y}, \theta^t) = \mathcal{N}(\mathbf{x}^n|\mathbf{m}^n, \mathbf{V}^n) \\
\mathbf{V}^n &= (I + \Lambda^\top \Psi^{-1} \Lambda)^{-1} \\
\mathbf{m}^n &= \mathbf{V}^n \Lambda^\top \Psi^{-1} (\mathbf{y} - \mu) \\
\mathbf{M-step}: \Lambda^{t+1} &= \left(\sum_n \mathbf{y}^n \mathbf{m}^{n\top} \right) \left(\sum_n \mathbf{V}^n \right)^{-1} \\
\Psi^{t+1} &= \frac{1}{N} \text{diag} \left[\sum_n \mathbf{y}^n \mathbf{y}^{n\top} + \Lambda^{t+1} \sum_n \mathbf{m}^n \mathbf{y}^{n\top} \right]
\end{aligned}$$

Principal Component Analysis

- In Factor Analysis, we can write the marginal density explicitly:

$$p(\mathbf{y}|\theta) = \int_{\mathbf{x}} p(\mathbf{x}) p(\mathbf{y}|\mathbf{x}, \theta) d\mathbf{x} = \mathcal{N}(\mathbf{y}|\mu\ ,\ \Lambda\Lambda^\top + \Psi)$$

- Noise Ψ mut be restricted for model to be interesting. (Why?)
- In Factor Analysis the restriction is that Ψ is *diagonal* (axis-aligned).
- What if we further restrict $\Psi = \sigma^2 I$ (ie *spherical*)?
- We get the Probabilistic Principal Component Analysis (PPCA) model:

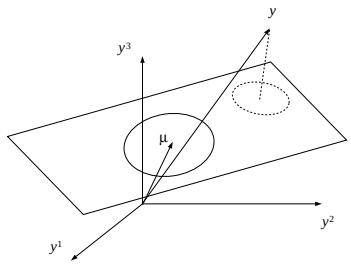
$$\begin{aligned}
p(\mathbf{x}) &= \mathcal{N}(\mathbf{x}|0, I) \\
p(\mathbf{y}|\mathbf{x}, \theta) &= \mathcal{N}(\mathbf{y}|\mu + \Lambda\mathbf{x}, \sigma^2 I)
\end{aligned}$$

  where μ is the mean vector,
  columns of Λ are the *principal components* (usually orthogonal),
  and σ^2 is the *global sensor noise*.

PCA: Zero Noise Limit

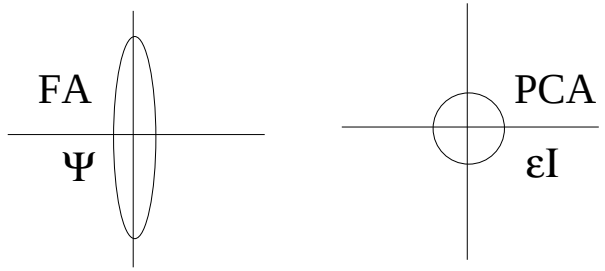
- The traditional PCA model is actually a limit as $\sigma^2 \to 0$.
  The model we saw is actually called "probabilistic PCA".
- However, the ML parameters Λ^* are the same.
  The only difference is the global sensor noise σ^2.
- In the zero noise limit inference is easier: orthogonal projection.

$$\lim_{\sigma^2 \to 0} \Lambda^\top (\Lambda\Lambda^\top + \sigma^2 I)^{-1} = (\Lambda^\top \Lambda)^{-1} \Lambda^\top$$



Gaussians are Footballs in High-D

- Recall the intuition that Gaussians are hyperellipsoids.
- Mean == centre of football
  Eigenvectors of covariance matrix == axes of football
  Eigenvalues == lengths of axes
- In FA our football is an axis aligned cigar.
  In PPCA our football is a sphere of radius σ^2.



Direct Fitting

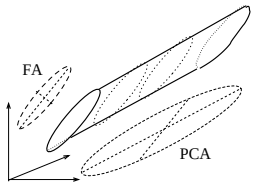
- For FA the parameters are coupled in a way that makes it impossible to solve for the ML params directly.
  We must use EM or other nonlinear optimization techniques.
- But for (P)PCA, the ML params can be solved for directly:
  The k^{th} column of Λ is the k^{th} largest eigenvalue of the sample covariance $\mathbf{S}$ times the associated eigenvector.
- The global sensor noise σ^2 is the sum of all the eigenvalues smaller than the k^{th} one.
- This technique is good for initializing FA also.
- Actually PCA is the limit as the ratio of the noise variance on the output to the prior variance on the latent variables goes to zero.
  We can either achieve this with zero noise or with infinite variance priors.

Scale Invariance in Factor Analysis

- In FA the *scale* of the data is unimportant: we can multiply $\mathbf{y}_i$ by α_i without changing anything:

$$\begin{aligned} \mu_i &\leftarrow \alpha_i \mu_i \\ \Lambda_{ij} &\leftarrow \alpha_i \Lambda_{ij} \quad \forall j \\ \Psi_i &\leftarrow \alpha_i^2 \Psi_i \end{aligned}$$

- However, the *rotation* of the data *is* important.
- FA looks for directions of large correlation in the data, so it is not fooled by large variance noise.

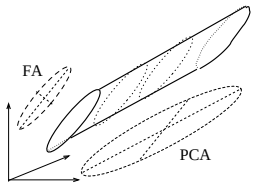


Rotational Invariance in PCA

- In PCA the *rotation* of the data is unimportant: we can multiply the data $\mathbf{y}$ by and rotation $\mathbf{Q}$ without changing anything:

$$\mu \leftarrow \mathbf{Q}\mu$$
$$\Lambda \leftarrow \mathbf{Q}\Lambda$$
$$\Psi \leftarrow \text{unchanged}$$

- However, the *scale* of the data *is* important.
- PCA looks for directions of large variance, so it will chase big noise directions.



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 paramters based on this guess:
  E-step: $q^{t+1} = p(\mathbf{z}|\mathbf{x}, \theta^t)$
  M-step: $\theta^{t+1} = \text{argmax}_\theta \sum_\mathbf{z} q(\mathbf{z}|\mathbf{x}) \log p(\mathbf{x}, \mathbf{z}|\theta)$
- In the M-step we optimize a lower bound on the likelihood. In the E-step we close the gap, making bound=likelihood.