Probabilistic & Unsupervised Learning

Lecture 2: Latent Variable Models

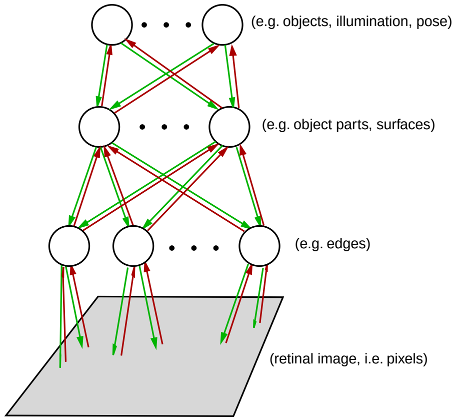
Maneesh Sahani
maneesh@gatsby.ucl.ac.uk

Gatsby Computational Neuroscience Unit, and
MSc ML/CSML, Dept Computer Science
University College London

Term 1, Autumn 2013

Latent variable models

Explain correlations in **x** by assuming dependence on latent variables **y**


(e.g. objects, illumination, pose)
(e.g. object parts, surfaces)
(e.g. edges)
(retinal image, i.e. pixels)

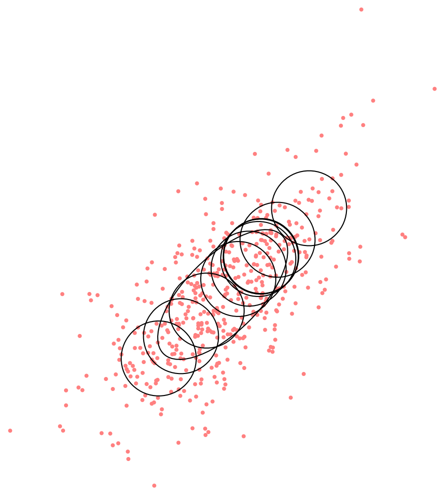
$$\mathbf{y} \sim \mathcal{P}[\theta_y]$$
$$\mathbf{x} \mid \mathbf{y} \sim \mathcal{P}[\theta_x]$$
$$p(\mathbf{x}, \mathbf{y}; \theta_x, \theta_y) = p(\mathbf{x} \mid \mathbf{y}; \theta_x) p(\mathbf{y}; \theta_y)$$
$$p(\mathbf{x}; \theta_x, \theta_y) = \int d\mathbf{y}\ p(\mathbf{x} \mid \mathbf{y}; \theta_x) p(\mathbf{y}; \theta_y)$$

Latent variable models

- Describe structured distributions.
  - Correlations in high-dimensional **x** may be captured by fewer parameters.
- Capture an underlying generative process.
  - **y** may describe *causes* of **x**.
  - help to separate signal from noise.
- Combine exponential family distributions into richer, more flexible forms.
  - $P(\mathbf{y})$, $P(\mathbf{x}|\mathbf{y})$ and even $P(\mathbf{x}, \mathbf{y})$ may be in the exponential family
  - $P(\mathbf{x})$ rarely is. (Exception: Linear Gaussian models).

Latent variables and Gaussians

Gaussian correlation can be composed from latent components and uncorrelated noise.



$$\mathbf{x} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix}\right) \quad \Leftrightarrow \quad y \sim \mathcal{N}(0, 1) \quad \mathbf{x} \sim \mathcal{N}\left(\sqrt{2}\begin{bmatrix} 1 \\ 1 \end{bmatrix} y, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right)$$

Probabilistic Principal Components Analysis (PPCA)

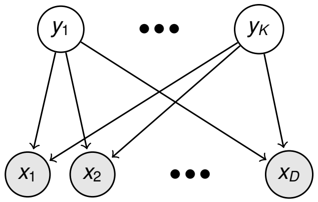
If the uncorrelated noise is assumed to be isotropic, this model is called PPCA.

Data: $\mathcal{D} = \mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}; \mathbf{x}_i \in \mathbb{R}^D$

Latents: $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\}; \mathbf{y}_i \in \mathbb{R}^K$

Linear generative model: $x_d = \sum_{k=1}^{K} \Lambda_{dk}\, y_k + \epsilon_d$

- y_k are independent $\mathcal{N}(0,1)$ Gaussian factors
- ϵ_d are independent $\mathcal{N}(0,\psi)$ Gaussian noise
- $K < D$



Model for observations $\mathbf{x}$ is a correlated Gaussian:

$$p(\mathbf{y}) = \mathcal{N}(0, I)$$

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\Lambda\mathbf{y}, \psi I)$$

$$p(\mathbf{x}) = \int p(\mathbf{y})p(\mathbf{x}|\mathbf{y})d\mathbf{y} = \mathcal{N}\left(\mathrm{E}\left[\Lambda\mathbf{y}\right], \mathrm{E}\left[\Lambda\mathbf{y}\mathbf{y}^\mathsf{T}\Lambda^\mathsf{T}\right] + \psi I\right) = \mathcal{N}\left(0, \Lambda\Lambda^\mathsf{T} + \psi I\right)$$

where Λ is a $D \times K$ matrix.

PPCA likelihood

The marginal distribution on $\mathbf{x}$ gives us the PPCA likelihood:

$$\log p(\mathcal{X}|\Lambda,\psi) = -\frac{N}{2}\log\left|2\pi(\Lambda\Lambda^\mathsf{T} + \psi I)\right| - \frac{1}{2}\mathrm{Tr}\left[(\Lambda\Lambda^\mathsf{T} + \psi I)^{-1}\underbrace{\sum_n \mathbf{x}\mathbf{x}^\mathsf{T}}_{NS}\right]$$

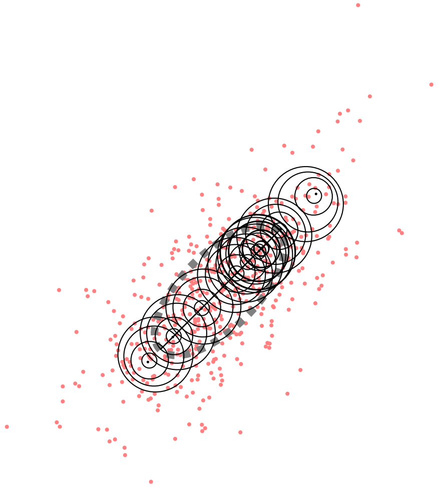
To find the ML values of (Λ, ψ) we could optimise numerically (gradient ascent / Newton's method), or we could use a different iterative algorithm called EM which we'll introduce soon.

In fact, however, ML for PPCA is more straightforward in principle, as we will see by first considering the limit $\psi \to 0$.

[Note: We may also add a constant mean $\boldsymbol{\mu}$ to the output, so as to model data that are not distributed around 0. In this case, the ML estimate $\widehat{\boldsymbol{\mu}} = \frac{1}{N}\sum_n \mathbf{x}_n$ and we can define $S = \frac{1}{N}\sum_n(\mathbf{x} - \widehat{\boldsymbol{\mu}})(\mathbf{x} - \widehat{\boldsymbol{\mu}})^\mathsf{T}$ in the likelihood above.]

The $\psi \to 0$ limit

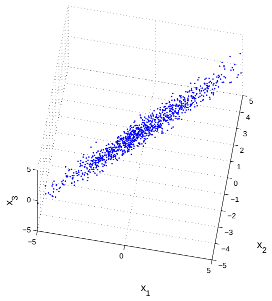
As $\psi \to 0$, the latent model can only capture K dimensions of variance.



In a Gaussian model, the ML parameters will find the K-dimensional space of most variance.

Principal Components Analysis

This leads us to an (old) algorithm called Principal Components Analysis (PCA).



Assume data $\mathcal{D} = \{\mathbf{x}_i\}$ have zero mean (if not, subtract it).

- Find direction of greatest variance – $\boldsymbol{\lambda}_{(1)}$.

$$\boldsymbol{\lambda}_{(1)} = \operatorname*{argmax}_{\|\mathbf{v}\|=1} \sum_n (\mathbf{x}_n^\mathsf{T}\mathbf{v})^2$$

- Find direction orthogonal to $\boldsymbol{\lambda}_{(1)}$ with greatest variance – $\boldsymbol{\lambda}_{(2)}$

$\vdots$

- Find direction orthogonal to $\{\boldsymbol{\lambda}_{(1)}, \boldsymbol{\lambda}_{(2)}, \ldots, \boldsymbol{\lambda}_{(n-1)}\}$ with greatest variance – $\boldsymbol{\lambda}_{(n)}$.
- Terminate when remaining variance drops below a threshold.

Eigendecomposition of a covariance matrix

The eigendecomposition of a covariance matrix makes finding the PCs easy.
Recall that $\mathbf{u}$ is an eigenvector, with scalar eigenvalue ω, of a matrix S if

$$S\mathbf{u} = \omega\mathbf{u}$$

$\mathbf{u}$ can have any norm, but we will define it to be unity (i.e., $\mathbf{u}^\mathsf{T}\mathbf{u} = 1$).
For a covariance matrix $S = \left\langle \mathbf{x}\mathbf{x}^\mathsf{T} \right\rangle$ (which is $D \times D$, symmetric, positive semi-definite):

- In general there are D eigenvector-eigenvalue pairs $(\mathbf{u}_{(i)}, \omega_{(i)})$, except if two or more eigenvectors share the same eigenvalue (in which case the eigenvectors are degenerate — any linear combination is also an eigenvector).
- The D eigenvectors are orthogonal (or orthogonalisable, if $\omega_{(i)} = \omega_{(j)}$). Thus, they form an orthonormal basis. $\sum_i \mathbf{u}_{(i)}\mathbf{u}_{(i)}{}^\mathsf{T} = I$.
- Any vector $\mathbf{v}$ can be written as

$$\mathbf{v} = \left(\sum_i \mathbf{u}_{(i)}\mathbf{u}_{(i)}{}^\mathsf{T}\right)\mathbf{v} = \sum_i (\mathbf{u}_{(i)}{}^\mathsf{T}\mathbf{v})\mathbf{u}_{(i)} = \sum_i v_{(i)}\mathbf{u}_{(i)}$$

- The original matrix S can be written:

$$S = \sum_i \omega_{(i)}\mathbf{u}_{(i)}\mathbf{u}_{(i)}{}^\mathsf{T} = UWU^\mathsf{T}$$

where $U = [\mathbf{u}_{(1)}, \mathbf{u}_{(2)}, \ldots, \mathbf{u}_{(D)}]$ collects the eigenvectors and
$W = \mathrm{diag}\left[(\omega_{(1)}, \omega_{(2)}, \ldots, \omega_{(D)})\right]$.

PCA and eigenvectors

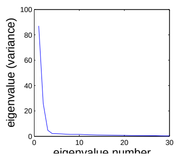
- The variance in direction $\mathbf{u}_{(i)}$ is

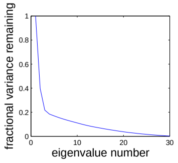
$$\left\langle (\mathbf{x}^\mathsf{T}\mathbf{u}_{(i)})^2 \right\rangle = \left\langle \mathbf{u}_{(i)}{}^\mathsf{T}\mathbf{x}\mathbf{x}^\mathsf{T}\mathbf{u}_{(i)} \right\rangle = \mathbf{u}_{(i)}{}^\mathsf{T} S\mathbf{u}_{(i)} = \mathbf{u}_{(i)}{}^\mathsf{T}\omega_{(i)}\mathbf{u}_{(i)} = \omega_{(i)}$$

- The variance in an arbitrary direction $\mathbf{v}$ is

$$\left\langle (\mathbf{x}^\mathsf{T}\mathbf{v})^2 \right\rangle = \left\langle \left(\mathbf{x}^\mathsf{T}\left(\sum_i v_{(i)}\mathbf{u}_{(i)}\right)\right)^2 \right\rangle = \sum_{ij} v_{(i)}\mathbf{u}_{(i)}{}^\mathsf{T} S\mathbf{u}_{(j)} v_{(j)}$$
$$= \sum_{ij} v_{(i)}\omega_{(j)} v_{(j)}\mathbf{u}_{(i)}{}^\mathsf{T}\mathbf{u}_{(j)} = \sum_i v_{(i)}^2\omega_{(i)}$$

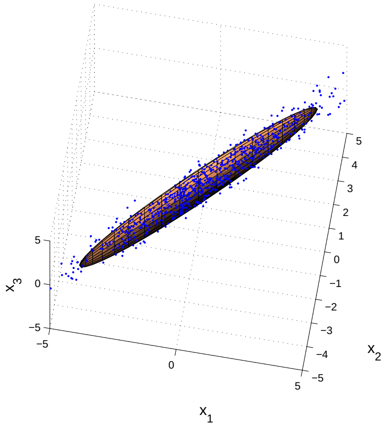
- If $\mathbf{v}^\mathsf{T}\mathbf{v} = 1$, then $\sum_i v_{(i)}^2 = 1$ and so $\mathrm{argmax}_{\|\mathbf{v}\|=1} \left\langle (\mathbf{x}^\mathsf{T}\mathbf{v})^2 \right\rangle = \mathbf{u}_{(\max)}$
The direction of greatest variance is the eigenvector the largest eigenvalue.
- In general, the PCs are exactly the eigenvectors of the empirical covariance matrix, ordered by decreasing eigenvalue.
- The eigenspectrum shows how the variance is distributed across dimensions; can identify transitions that might separate signal from noise, or the number of PCs that capture a predetermined fraction of variance.





PCA subspace

The K principle components define the K-dimensional subspace of greatest variance.

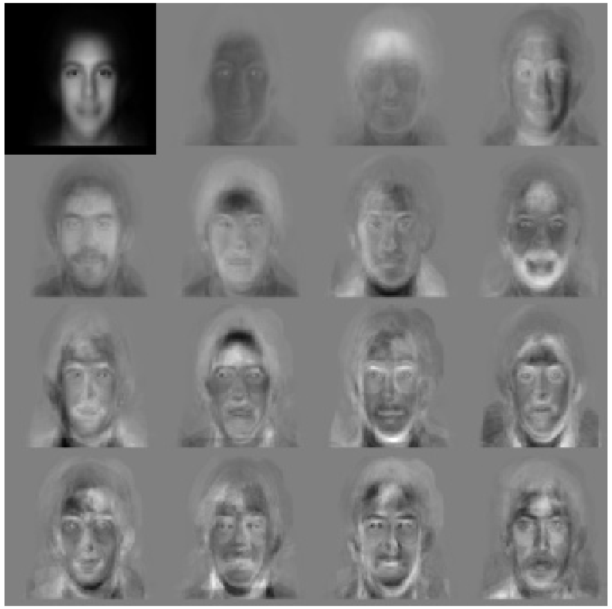


- Each data point $\mathbf{x}_n$ is associated with a projection $\hat{\mathbf{x}}_n$ into the principle subspace.

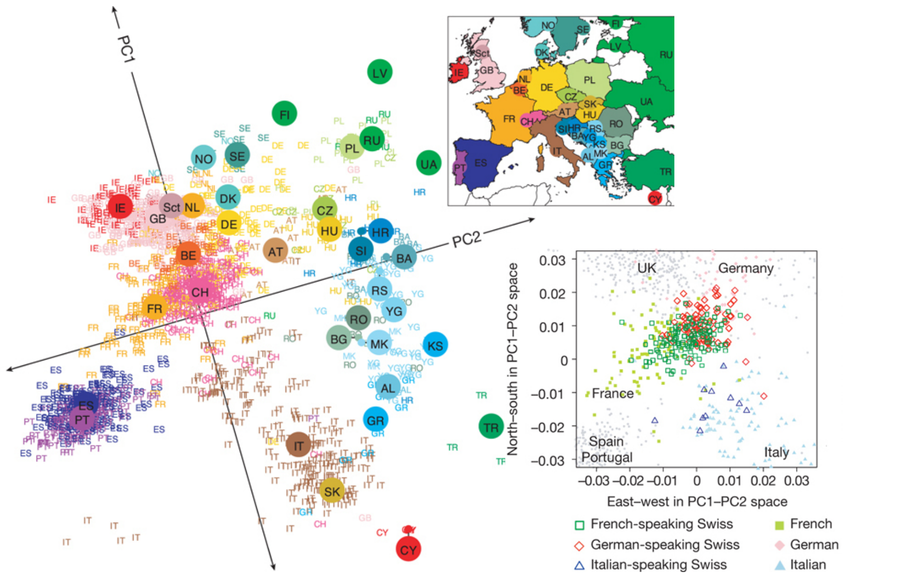
$$\hat{\mathbf{x}}_n = \sum_{k=1}^{K} x_{n(k)}\boldsymbol{\lambda}_{(k)}$$

- This can be used for lossy compression, denoising, recognition, ...

Example of PCA: Eigenfaces

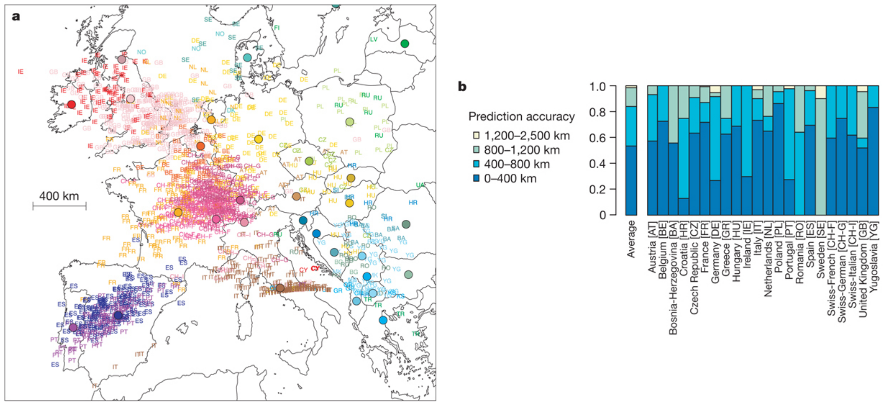


Example of PCA: Genetic variation within Europe



Novembre et al. (2008) Nature 456:98-101

Example of PCA: Genetic variation within Europe



Novembre et al. (2008) Nature 456:98-101

Equivalent definitions of PCA

- Find K directions of greatest variance in data.
- Find K-dimensional orthogonal projection that *preserves* greatest variance.
- Find K-dimensional vectors $\mathbf{y}_i$ and matrix Λ so that $\hat{\mathbf{x}}_i = \Lambda \mathbf{y}_i$ is as close as possible (in squared distance) to $\mathbf{x}_i$.
- . . . (many others)

Another view of PCA: Mutual information

Problem: Given $\mathbf{x}$, find $\mathbf{y} = A\mathbf{x}$ with columns of A unit vectors, s.t. $I(\mathbf{y}; \mathbf{x})$ is maximised (assuming that $P(\mathbf{x})$ is Gaussian).

$$I(\mathbf{y}; \mathbf{x}) = H(\mathbf{y}) + H(\mathbf{x}) - H(\mathbf{y}, \mathbf{x}) = H(\mathbf{y})$$

So we want to maximise the entropy of $\mathbf{y}$. What is the entropy of a Gaussian?

$$H(\mathbf{z}) \quad = \quad -\int d\mathbf{z}\, p(\mathbf{z}) \ln p(\mathbf{z}) = \frac{1}{2} \ln |\Sigma| + \frac{D}{2}(1 + \ln 2\pi)$$

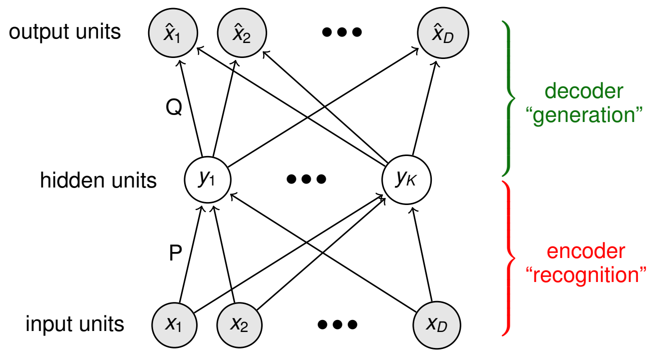
Therefore we want the distribution of $\mathbf{y}$ to have largest volume (i.e. det of covariance matrix).

$$\Sigma_y = A \Sigma_x A^\mathsf{T} = A U W_x U^\mathsf{T} A^\mathsf{T}$$

So, A should be aligned with the columns of U which are associated with the largest eigenvalues (variances).

Projection to the principal component subspace preserves the most information about the (Gaussian) data.

Linear autoencoders: From supervised learning to PCA



$$\text{Learning: } \underset{P,Q}{\operatorname{argmin}} \|\hat{\mathbf{x}} - \mathbf{x}\|^2 \qquad \hat{\mathbf{x}} = Q\mathbf{y} \quad \mathbf{y} = P\mathbf{x}$$

At the optimum, P and Q perform the projection and reconstruction steps of PCA. (Baldi & Hornik 1989).

ML learning for PPCA

$$\ell = -\frac{N}{2} \log |2\pi C| - \frac{N}{2} \mathrm{Tr}\left[C^{-1}S\right] \qquad \text{where } C = \Lambda\Lambda^{\mathsf{T}} + \psi I$$

$$\frac{\partial \ell}{\partial \Lambda} = \frac{N}{2}\left(-\frac{\partial}{\partial \Lambda}|C| - \frac{\partial}{\partial \Lambda}\mathrm{Tr}\left[C^{-1}S\right]\right) = \frac{N}{2}\left(-C^{-1}\Lambda + C^{-1}SC^{-1}\Lambda\right)$$

So at the stationary points we have $SC^{-1}\Lambda = \Lambda$. This implies either:

- $\Lambda = 0$, which turns out to be a minimum.
- $C = S \Rightarrow \Lambda\Lambda^{\mathsf{T}} = S - \psi I$. Now $\mathrm{rank}(\Lambda\Lambda^{\mathsf{T}}) \leq K \Rightarrow \mathrm{rank} S - \psi I \Rightarrow S$ has $D - K$ eigenvalues $= \psi$ and Λ aligns with space of remaining eigenvectors.
- or, taking the SVD: $\Lambda = ULV^{\mathsf{T}}$:

$$\begin{array}{rrl} & S(ULV^{\mathsf{T}}VLU^{\mathsf{T}} + \psi I)^{-1}ULV^{\mathsf{T}} = ULV^{\mathsf{T}} & \times VL^{-1} \\ \Rightarrow & S(UL^2U^{\mathsf{T}} + \psi I)^{-1}U = U & U(L^2 + \psi I) = (UL^2U^{\mathsf{T}} + \psi I)U \\ & & \Rightarrow (UL^2U^{\mathsf{T}} + \psi I)^{-1}U = U(L^2 + \psi I)^{-1} \\ \Rightarrow & SU(L^2 + \psi I)^{-1} = U & \times(L^2 + \psi I) \\ \Rightarrow & SU = U\underbrace{(L^2 + \psi I)}_{\text{diagonal}} & \end{array}$$

$\Rightarrow$ columns of U are eigenvectors of S with eigenvalues given by $l_i^2 + \psi$.
Thus, $\Lambda = ULV^{\mathsf{T}}$ spans a space defined by K eigenvectors of S; and the lengths of the column vectors of L are given by the eigenvalues $-\psi$ (V selects an arbitrary basis in the latent space).

Remains to show (we won't, but it's intuitively reasonable) that the global ML solution is attained when Λ aligns with the K *leading* eigenvectors.

PPCA latents

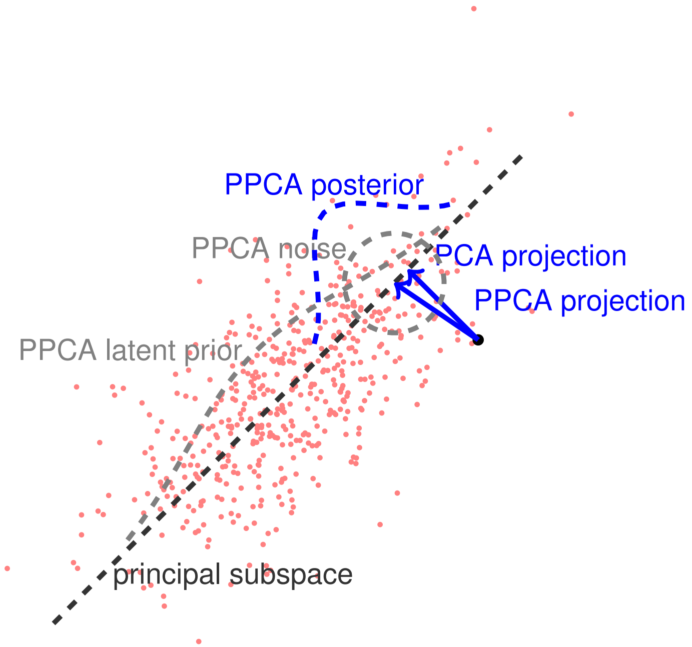
- In PCA the "noise" is orthogonal to the subspace, and we can project $\mathbf{x}_n \to \tilde{\mathbf{x}}_n$ trivially.
- In PPCA, the noise is more sensible (equal in all directions). But what is the projection?

  Find the expected value of $\mathbf{y}_n|\mathbf{x}_n$ and then take $\tilde{\mathbf{x}}_n = \Lambda\bar{\mathbf{y}}_n$.
- **Tactic:** write $p(\mathbf{y}_n, \mathbf{x}_n|\theta)$, consider $\mathbf{x}_n$ to be fixed. What is this as a function of $\mathbf{y}_n$?

$$\begin{aligned} p(\mathbf{y}_n, \mathbf{x}_n) &= p(\mathbf{y}_n)p(\mathbf{x}_n|\mathbf{y}_n) \\ &= (2\pi)^{-\frac{K}{2}} \exp\{-\frac{1}{2}\mathbf{y}_n^{\mathsf{T}}\mathbf{y}_n\} \, |2\pi\Psi|^{-\frac{1}{2}} \exp\{-\frac{1}{2}(\mathbf{x}_n - \Lambda\mathbf{y}_n)^{\mathsf{T}}\Psi^{-1}(\mathbf{x}_n - \Lambda\mathbf{y}_n)\} \\ &= \mathrm{c} \times \exp\{-\frac{1}{2}[\mathbf{y}_n^{\mathsf{T}}\mathbf{y}_n + (\mathbf{x}_n - \Lambda\mathbf{y}_n)^{\mathsf{T}}\Psi^{-1}(\mathbf{x}_n - \Lambda\mathbf{y}_n)]\} \\ &= \mathrm{c'} \times \exp\{-\frac{1}{2}[\mathbf{y}_n^{\mathsf{T}}(I + \Lambda^{\mathsf{T}}\Psi^{-1}\Lambda)\mathbf{y}_n - 2\mathbf{y}_n^{\mathsf{T}}\Lambda^{\mathsf{T}}\Psi^{-1}\mathbf{x}_n]\} \\ &= \mathrm{c''} \times \exp\{-\frac{1}{2}[\mathbf{y}_n^{\mathsf{T}}\Sigma^{-1}\mathbf{y}_n - 2\mathbf{y}_n^{\mathsf{T}}\Sigma^{-1}\mu + \mu^{\mathsf{T}}\Sigma^{-1}\mu]\} \end{aligned}$$

  So $\Sigma = (I + \Lambda^{\mathsf{T}}\Psi^{-1}\Lambda)^{-1} = I - \beta\Lambda$ and $\mu = \Sigma\Lambda^{\mathsf{T}}\Psi^{-1}\mathbf{x}_n = \beta\mathbf{x}_n$. Where $\beta = \Sigma\Lambda^{\mathsf{T}}\Psi^{-1}$.
- Thus, $\tilde{\mathbf{x}}_n = \Lambda(I + \Lambda^{\mathsf{T}}\Psi^{-1}\Lambda)^{-1}\Lambda^{\mathsf{T}}\Psi^{-1}\mathbf{x}_n = \mathbf{x}_n - \Psi(\Lambda\Lambda^{\mathsf{T}} + \Psi)^{-1}\mathbf{x}_n$
- This is not the same projection. PPCA takes into account noise in the principal subspace.
- As $\psi \to 0$, the PPCA estimate $\to$ the PCA value.

PPCA latents



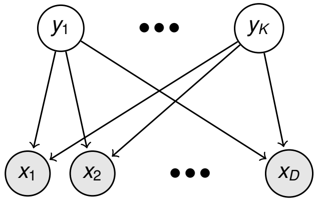
Factor Analysis

If dimensions are not equivalent, equal variance assumption is inappropriate.

Data: $\mathcal{D} = \mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}; \mathbf{x}_i \in \mathbb{R}^D$
Latents: $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\}; \mathbf{y}_i \in \mathbb{R}^K$

Linear generative model: $x_d = \sum_{k=1}^{K} \Lambda_{dk}\, y_k + \epsilon_d$

- y_k are independent $\mathcal{N}(0, 1)$ Gaussian factors
- ϵ_d are independent $\mathcal{N}(0, \Psi_{dd})$ Gaussian noise
- $K < D$



Model for observations $\mathbf{x}$ is still a correlated Gaussian:

$$p(\mathbf{y}) = \mathcal{N}(0, I)$$

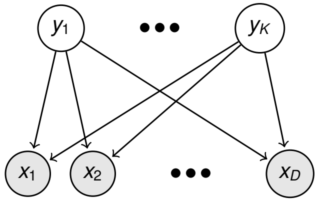
$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\Lambda\mathbf{y}, \Psi)$$

$$p(\mathbf{x}) = \int p(\mathbf{y}) p(\mathbf{x}|\mathbf{y}) d\mathbf{y} = \mathcal{N}\left(0, \Lambda\Lambda^{\mathsf{T}} + \Psi\right)$$

where Λ is a $D \times K$, and Ψ is $K \times K$ and diagonal.

Dimensionality Reduction: Finds a low-dimensional projection of high dimensional data that captures the correlation structure of the data.

Factor Analysis (cont.)



- ML learning finds Λ ("common factors") and Ψ ("unique factors" or "uniquenesses") given data
- parameters (corrected for symmetries): $DK + D - \frac{K(K-1)}{2}$
- If number of parameters $> \frac{D(D+1)}{2}$ model is not identifiable (even after accounting for rotational degeneracy discussed later)
- no closed form solution for ML params: $\mathcal{N}(0, \Lambda\Lambda^{\mathsf{T}} + \Psi)$

Factor Analysis projections

Our analysis for PPCA still applies:

$$\tilde{\mathbf{x}}_n = \Lambda(I + \Lambda^{\mathsf{T}}\Psi^{-1}\Lambda)^{-1}\Lambda^{\mathsf{T}}\Psi^{-1}\mathbf{x}_n = \mathbf{x}_n - \Psi(\Lambda\Lambda^{\mathsf{T}} + \Psi)^{-1}\mathbf{x}_n$$

but now Ψ is diagonal but not spherical.

Note, though, that Λ is generally different from that found by PPCA.

And Λ is not unique: the latent space may be transformed by an arbitrary orthogonal transform U ($U^{\mathsf{T}}U = UU^{\mathsf{T}} = I$) without changing the likelihood.

$$\tilde{\mathbf{y}} = U\mathbf{y} \quad \text{and} \quad \tilde{\Lambda} = \Lambda U^{\mathsf{T}} \quad \Rightarrow \quad \tilde{\Lambda}\tilde{\mathbf{y}} = \Lambda U^{\mathsf{T}} U\mathbf{y} = \Lambda\mathbf{y}$$

$$\begin{aligned} -\ell &= \frac{1}{2}\log\left|2\pi(\Lambda U^{\mathsf{T}} U \Lambda^{\mathsf{T}} + \Psi)\right| + \frac{1}{2}\mathbf{x}^{\mathsf{T}}(\Lambda U^{\mathsf{T}} U \Lambda^{\mathsf{T}} + \Psi)^{-1}\mathbf{x} \\ &= \frac{1}{2}\log\left|2\pi(\tilde{\Lambda}\tilde{\Lambda}^{\mathsf{T}} + \Psi)\right| + \frac{1}{2}\mathbf{x}^{\mathsf{T}}(\tilde{\Lambda}\tilde{\Lambda}^{\mathsf{T}} + \Psi)^{-1}\mathbf{x} \end{aligned}$$

Gradient methods for learning FA

Optimise negative log-likelihood:

$$-\ell = \frac{1}{2}\log|2\pi(\Lambda\Lambda^{\mathsf{T}} + \Psi)| + \frac{1}{2}\mathbf{x}^{\mathsf{T}}(\Lambda\Lambda^{\mathsf{T}} + \Psi)^{-1}\mathbf{x}$$

w.r.t. Λ and Ψ (need matrix calculus) subject to constraints.

- No spectral short-cut exists.
- Likelihood can have more than one (local) optimum, making it difficult to find the global value.
- For some data ("Heywood cases") likelihood may grow unboundedly by taking one or more $\Psi_{dd} \to 0$. Can eliminate by assuming a prior on Ψ with zero density at $\Psi_{dd} = 0$, but results sensitive to precise choice of prior.

Expectation maximisation (next week) provides an alternative approach to maximisation, but doesn't solve these issues.

FA vs PCA

- PCA and PPCA are rotationally invariant; FA is not

$$\text{If } \mathbf{x} \to U\mathbf{x} \text{ for unitary } U, \quad \text{then } \boldsymbol{\lambda}_{(i)}^{\mathsf{PCA}} \to U\boldsymbol{\lambda}_{(i)}^{\mathsf{PCA}}$$

- FA is measurement scale invariant; PCA and PPCA are not

$$\text{If } \mathbf{x} \to S\mathbf{x} \text{ for diagonal } S, \quad \text{then } \boldsymbol{\lambda}_{(i)}^{\mathsf{FA}} \to S\boldsymbol{\lambda}_{(i)}^{\mathsf{FA}}$$

- FA and PPCA define a probabilistic model; PCA does not

[Note: it may be tempting to try to eliminate the scale-dependence of (P)PCA by pre-processing data to equalise total variance on each axis. But P(PCA) assume equal *noise* variance. Total variance has contributions from both $\Lambda\Lambda^{\mathsf{T}}$ and noise, so this approach does not exactly solve the problem.]

Canonical Correlations Analysis

Data vector pairs: $\mathcal{D} = \{(\mathbf{u}_1, \mathbf{v}_1), (\mathbf{u}_2, \mathbf{v}_2) \dots\}$ in spaces $\mathcal{U}$ and $\mathcal{V}$.

Classic CCA

- Find unit vectors $\boldsymbol{\upsilon}_1 \in \mathcal{U}$, $\boldsymbol{\phi}_1 \in \mathcal{V}$ such that the correlation of $\mathbf{u}_i^{\mathsf{T}}\boldsymbol{\upsilon}_1$ and $\mathbf{v}_i^{\mathsf{T}}\boldsymbol{\phi}_1$ is maximised.
- As with PCA, repeat in orthogonal subspaces.

Probabilistic CCA

- Generative model with latent $\mathbf{y}_i \in \mathbb{R}^K$:

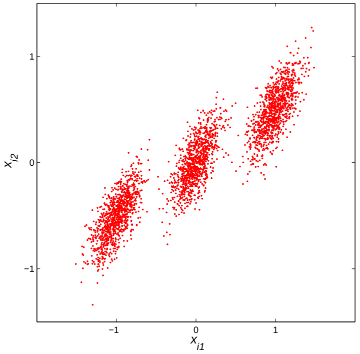
$$\begin{aligned} \mathbf{y} &\sim \mathcal{N}(0, I) \\ \mathbf{u} &\sim \mathcal{N}(\Upsilon\mathbf{y}, \Psi_u) \quad \Psi_u \succcurlyeq 0 \\ \mathbf{v} &\sim \mathcal{N}(\Phi\mathbf{y}, \Psi_v) \quad \Psi_v \succcurlyeq 0 \end{aligned}$$

- Block diagonal noise.

Limitations of Gaussian, FA and PCA models

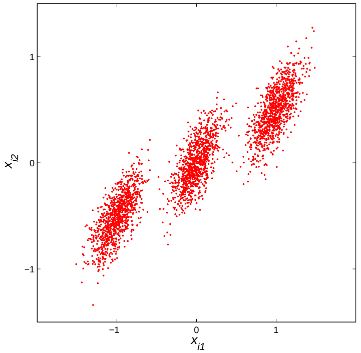
- Gaussian, FA and PCA models are easy to understand and use in practice.
- They are a convenient way to find interesting directions in very high dimensional data sets, eg as preprocessing
- Their problem is that they make very strong assumptions about the distribution of the data, only the mean and variance of the data are taken into account.

The class of densities which can be modelled is too restrictive.



By using *mixtures* of simple distributions, such as Gaussians, we can expand the class of densities greatly.

Mixture Distributions



A mixture distribution has a single discrete latent variable:

$$\begin{aligned} s_i &\overset{\text{iid}}{\sim} \mathit{Discrete}[\boldsymbol{\pi}] \\ \mathbf{x}_i \mid s_i &\sim \mathcal{P}_{s_i}[\theta_{s_i}] \end{aligned}$$

Mixtures arise naturally when observations from different sources have been collated. They can also be used to *approximate* arbitrary distributions.

The Mixture Likelihood

The mixture model is

$$s_i \overset{\text{iid}}{\sim} \textit{Discrete}[\boldsymbol{\pi}]$$
$$\mathbf{x}_i \mid s_i \sim \mathcal{P}_{s_i}[\theta_{s_i}]$$

Under the discrete distribution

$$P(s_i = m) = \pi_m; \qquad \pi_m \geq 0, \ \sum_{m=1}^{k} \pi_m = 1$$

Thus, the probability (density) at a single data point $\mathbf{x}_i$ is

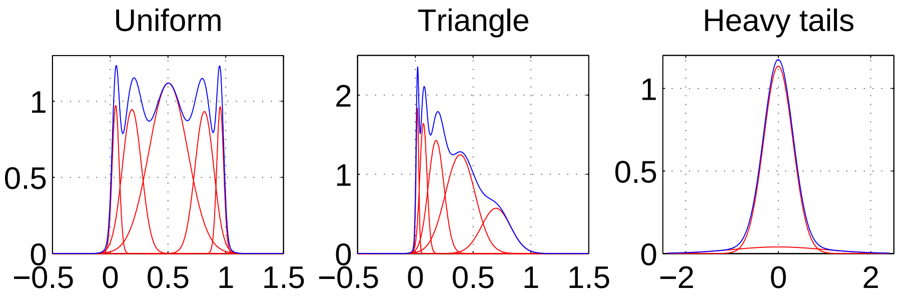
$$P(\mathbf{x}_i) = \sum_{m=1}^{k} P(\mathbf{x}_i \mid s_i = m) P(s_i = m)$$
$$= \sum_{m=1}^{k} \pi_m P_m(\mathbf{x}_i; \theta_m)$$

The mixture distribution (density) is a convex combination (or *weighted average*) of the component distributions (densities).

Approximation with a Mixture of Gaussians (MoG)

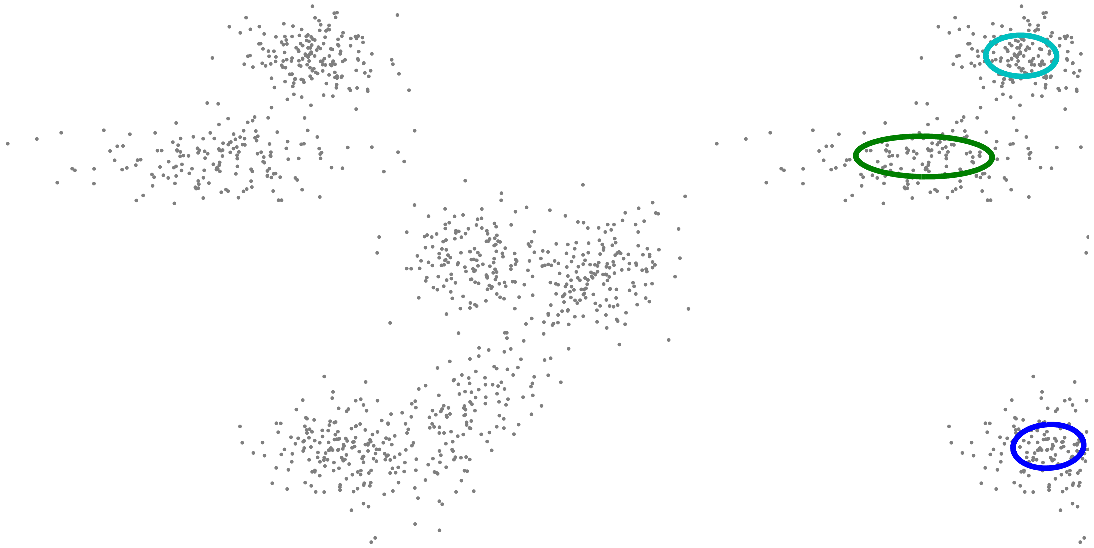
The component densities may be viewed as elements of a *basis* which can be combined to approximate arbitrary distributions.

Here are examples where non-Gaussian densities are modelled (aproximated) as a mixture of Gaussians. The red curves show the (weighted) Gaussians, and the blue curve the resulting density.



Given enough mixture components we can model (almost) any density (as accurately as desired), but still only need to work with the well-known Gaussian form.

Clustering with a MoG



Clustering with a MoG

In clustering applications, the latent variable s_i represents the (unknown) identity of the cluster to which the ith observation belongs.

Thus, the latent distribution gives the *prior* probability of a data point coming from each cluster.

$$P(s_i = m \mid \pi) = \pi_m$$

Data from the mth cluster are distributed according to the mth component:

$$P(\mathbf{x}_i \mid s_i = m) = P_m(\mathbf{x}_i)$$

Once we observe a data point, the *posterior* probability distribution for the cluster it belongs to is

$$P(s_i = m \mid \mathbf{x}_i) = \frac{P_m(\mathbf{x}_i)\pi_m}{\sum_m P_m(\mathbf{x}_i)\pi_m}$$

This is often called the responsibility of the mth cluster for the ith data point.

The MoG likelihood

Each component of a MoG is a Gaussian, with mean $\boldsymbol{\mu}_m$ and covariance matrix Σ_m. Thus, the probability density evaluated at a set of n iid observations, $\mathcal{D} = \{\mathbf{x}_1 \dots \mathbf{x}_n\}$ (i.e. the likelihood) is

$$\begin{aligned} p(\mathcal{D} \mid \{\boldsymbol{\mu}_m\}, \{\Sigma_m\}, \boldsymbol{\pi}) &= \prod_{i=1}^{n} \sum_{m=1}^{k} \pi_m \mathcal{N}\left(\mathbf{x}_i \mid \boldsymbol{\mu}_m, \Sigma_m\right) \\ &= \prod_{i=1}^{n} \sum_{m=1}^{k} \pi_m \frac{1}{\sqrt{|2\pi\Sigma_m|}} e^{-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_m)^{\mathsf{T}} \Sigma_m^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_m)} \end{aligned}$$

The log of the likelihood is

$$\log p(\mathcal{D} \mid \{\boldsymbol{\mu}_m\}, \{\Sigma_m\}, \boldsymbol{\pi}) = \sum_{i=1}^{n} \log \sum_{m=1}^{k} \pi_m \frac{1}{\sqrt{|2\pi\Sigma_m|}} e^{-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_m)^{\mathsf{T}} \Sigma_m^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_m)}$$

Note that the logarithm fails to simplify the component density terms. A mixture distribution does not lie in the exponential family. Direct optimisation is not easy.

Maximum Likelihood for a Mixture Model

The log likelihood is:

$$\mathcal{L} = \sum_{i=1}^{n} \log \sum_{m=1}^{k} \pi_m P_m(\mathbf{x}_i; \theta_m)$$

Its partial derivative wrt θ_m is

$$\frac{\partial \mathcal{L}}{\partial \theta_m} = \sum_{i=1}^{n} \frac{\pi_m}{\sum_{m=1}^{k} \pi_m P_m(\mathbf{x}_i; \theta_m)} \frac{\partial P_m(\mathbf{x}_i; \theta_m)}{\partial \theta_m}$$

or, using $\partial P / \partial \theta = P \times \partial \log P / \partial \theta$,

$$\begin{aligned} &= \sum_{i=1}^{n} \underbrace{\frac{\pi_m P_m(\mathbf{x}_i; \theta_m)}{\sum_{m=1}^{k} \pi_m P_m(\mathbf{x}_i; \theta_m)}} \frac{\partial \log P_m(\mathbf{x}_i; \theta_m)}{\partial \theta_m} \\ &= \sum_{i=1}^{n} \qquad r_{im} \qquad \frac{\partial \log P_m(\mathbf{x}_i; \theta_m)}{\partial \theta_m} \end{aligned}$$

And its partial derivative wrt π_m is

$$\frac{\partial \mathcal{L}}{\partial \pi_m} = \sum_{i=1}^{n} \frac{P_m(\mathbf{x}_i; \theta_m)}{\sum_{m=1}^{k} \pi_m P_m(\mathbf{x}_i; \theta_m)} = \sum_{i=1}^{n} \frac{r_{im}}{\pi_m}$$

MoG Derivatives

For a MoG, with $\theta_m = \{\boldsymbol{\mu}_m, \Sigma_m\}$ we get

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_m} &= \sum_{i=1}^{n} r_{im} \Sigma_m^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_m) \\ \frac{\partial \mathcal{L}}{\partial \Sigma_m^{-1}} &= \frac{1}{2} \sum_{i=1}^{n} r_{im} \left(\Sigma_m - (\mathbf{x}_i - \boldsymbol{\mu}_m)(\mathbf{x}_i - \boldsymbol{\mu}_m)^{\mathsf{T}} \right) \end{aligned}$$

These equations can be used (along with the derivatives wrt to π_m) for gradient based learning; e.g., taking small steps in the direction of the gradient (or using conjugate gradients).

The K-means Algorithm

The K-means algorithm is a limiting case of the mixture of Gaussians (c.f. PCA and Factor Analysis).

Take $\pi_m = 1/k$ and $\Sigma_m = \sigma^2 I$, with $\sigma^2 \to 0$. Then the responsibilities become binary

$$r_{im} \to \delta(m, \operatorname*{argmin}_{l} \|\mathbf{x}_i - \boldsymbol{\mu}_l\|^2)$$

with 1 for the component with the closest mean and 0 for all other components. We can then solve directly for the means by setting the gradient to 0.

The k-means algorithm iterates these two steps:

- assign each point to its closest mean $\left(\text{set } r_{im} = \delta(m, \operatorname*{argmin}_{l} \|\mathbf{x}_i - \boldsymbol{\mu}_l\|^2) \right)$
- update the means to the average of their assigned points $\left(\text{set } \boldsymbol{\mu}_m = \frac{\sum_i r_{im} \mathbf{x}_i}{\sum_i r_{im}} \right)$

This usually converges within a few iterations, although the fixed point depends on the initial values chosen for $\boldsymbol{\mu}_m$. The algorithm has no learning rate, but the assumptions are quite limiting.

A preview of the EM algorithm

We wrote the k-means algorithm in terms of binary responsibilities. Suppose, instead, we used the fractional responsibilities from the full (non-limiting) MoG, but still neglected the dependence of the responsibilities on the parameters. We could then solve for both $\boldsymbol{\mu}_m$ and Σ_m.

The EM algorithm for MoGs iterates these two steps:

- Evaluate the responsibilities for each point given the current parameters.
- Optimise the parameters assuming the responsibilities stay fixed:

$$\boldsymbol{\mu}_m = \frac{\sum_i r_{im}\mathbf{x}_i}{\sum_i r_{im}} \quad \text{and} \quad \Sigma_m = \frac{\sum_i r_{im}(\mathbf{x}_i - \boldsymbol{\mu}_m)(\mathbf{x}_i - \boldsymbol{\mu}_m)^\top}{\sum_i r_{im}}$$

Although this appears *ad hoc*, we will see (later) that it is a special case of a general algorithm, and is actually guaranteed to increase the likelihood at each iteration.

Issues

There are several problems with the new algorithms:

- slow convergence for the gradient based method
- gradient based method may develop invalid covariance matrices
- local minima; the end configuration may depend on the starting state
- how do you adjust k? Using the likelihood alone is no good.
- singularities; components with a single data point will have their covariance going to zero and the likelihood will tend to infinity.