

Week 2: Latent Variable Models

Maneesh Sahani

`maneesh@gatsby.ucl.ac.uk`

**Gatsby Computational Neuroscience Unit
University College London**

Term 1, Autumn 2005

Unsupervised Learning

The story so far . . .

- beliefs about data represented by parameterised distribution

$$P(\mathcal{D} \mid \theta, m) = \prod_{i=1}^n P(x_i \mid \theta, m)$$

- learning involves estimating a value (or distribution) for θ

Bayes
$$P(\theta \mid \mathcal{D}, m) = \frac{P(\mathcal{D} \mid \theta, m)P(\theta \mid m)}{P(\mathcal{D} \mid m)}$$

MAP
$$\theta^* = \operatorname{argmax}_{\theta} P(\theta \mid \mathcal{D}, m)$$

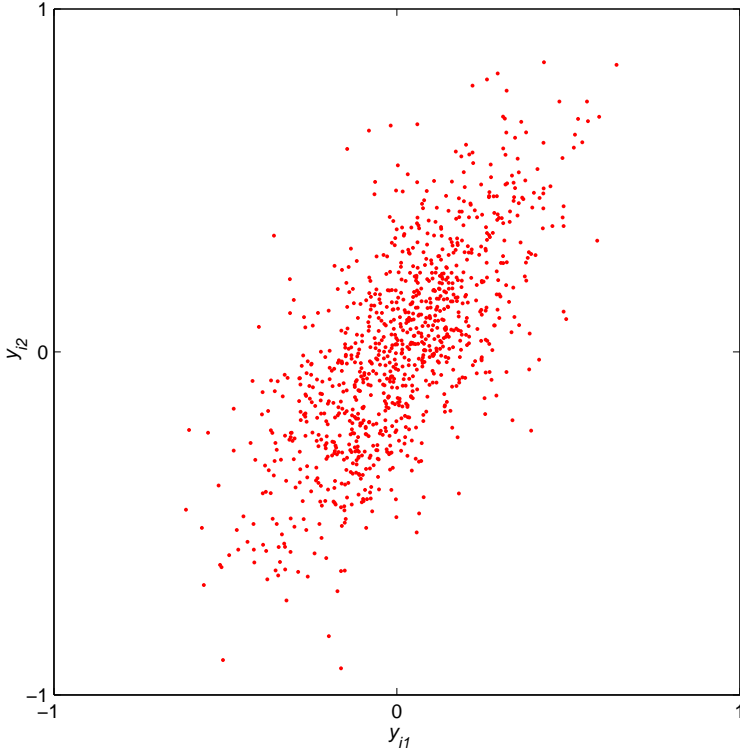
ML
$$\theta^* = \operatorname{argmax}_{\theta} P(\mathcal{D} \mid \theta, m)$$

Aren't We Done?

No!

- direct application of Bayes' rule is intractable for all but the simplest models.
- even maximum-likelihood (or similar) learning may be prohibitive.
- algorithms (and approximations) are dictated by form of distribution.

Correlated Continuous-Valued Data



Assume:

- we have a data set $Y = \{\mathbf{y}_1, \dots, \mathbf{y}_N\}$
- each data point is a vector of D features:
 $\mathbf{y}_i = [y_{i1} \dots y_{iD}]$
- the data points are i.i.d. (independent and identically distributed).

One of the simplest forms of unsupervised learning: model the **mean** of the data and the **correlations** between the D features in the data

We can use a multi-variate Gaussian (or **M**ulti-**V**ariate **N**ormal) model:

$$p(\mathbf{y}|\boldsymbol{\mu}, \Sigma) = |2\pi\Sigma|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2}(\mathbf{y} - \boldsymbol{\mu})^\top \Sigma^{-1}(\mathbf{y} - \boldsymbol{\mu}) \right\}$$

ML Estimation of a Gaussian

Data set $Y = \{\mathbf{y}_1, \dots, \mathbf{y}_N\}$, likelihood: $p(Y|\boldsymbol{\mu}, \Sigma) = \prod_{n=1}^N p(\mathbf{y}_n|\boldsymbol{\mu}, \Sigma)$

Maximise likelihood \Leftrightarrow maximise log likelihood

Goal: find $\boldsymbol{\mu}$ and Σ that maximise log likelihood:

$$\begin{aligned}\mathcal{L} &= \log \prod_{n=1}^N p(\mathbf{y}_n|\boldsymbol{\mu}, \Sigma) = \sum_n \log p(\mathbf{y}_n|\boldsymbol{\mu}, \Sigma) \\ &= -\frac{N}{2} \log |2\pi\Sigma| - \frac{1}{2} \sum_n (\mathbf{y}_n - \boldsymbol{\mu})^\top \Sigma^{-1} (\mathbf{y}_n - \boldsymbol{\mu})\end{aligned}\tag{1}$$

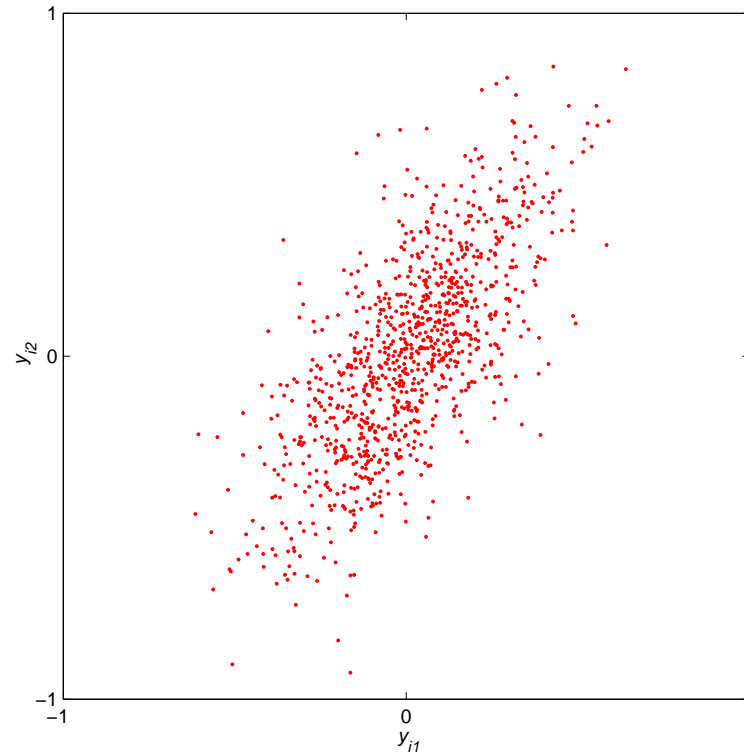
Note: equivalently, minimise $-\mathcal{L}$, which is *quadratic* in $\boldsymbol{\mu}$

Procedure: take derivatives and set to zero:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}} = 0 \quad \Rightarrow \quad \hat{\boldsymbol{\mu}} = \frac{1}{N} \sum_n \mathbf{y}_n \quad (\text{sample mean})$$

$$\frac{\partial \mathcal{L}}{\partial \Sigma} = 0 \quad \Rightarrow \quad \hat{\Sigma} = \frac{1}{N} \sum_n (\mathbf{y}_n - \hat{\boldsymbol{\mu}})(\mathbf{y}_n - \hat{\boldsymbol{\mu}})^\top \quad (\text{sample covariance})$$

Note



modelling correlations



maximising likelihood of a Gaussian model



minimising a squared error cost function



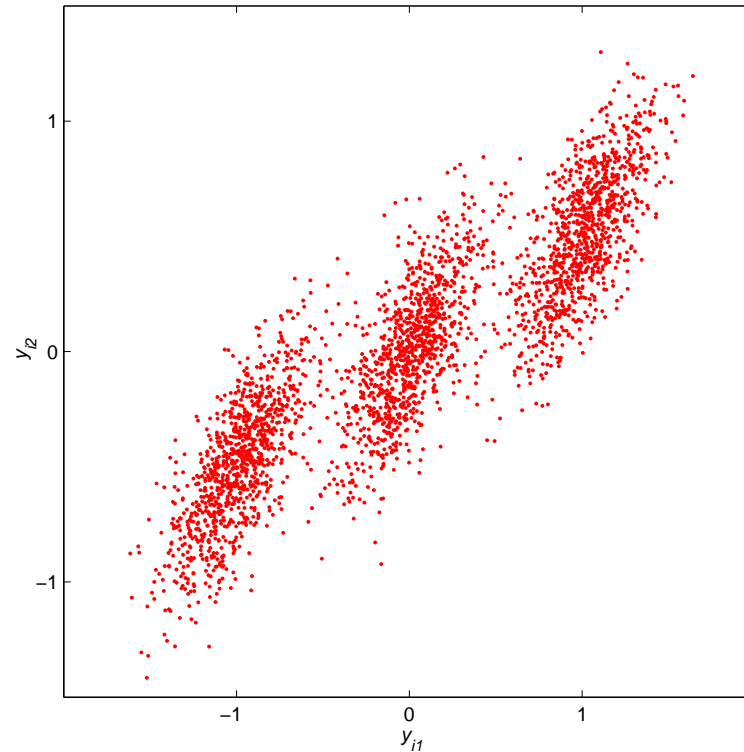
minimising data coding cost in bits (assuming Gaussian distributed)

MVN Limitations

Gaussians are fundamental and widespread, but not every distribution of interest is Gaussian.

- Not all random processes fit the central limit theorem.
- Some processes produce outliers.
- Some data has higher-order or non-linear structure.
- Even if data are Gaussian, if D is large the full MVN model may be difficult to handle. There are $D(D + 1)/2$ parameters in the MVN covariance.

What about these data?

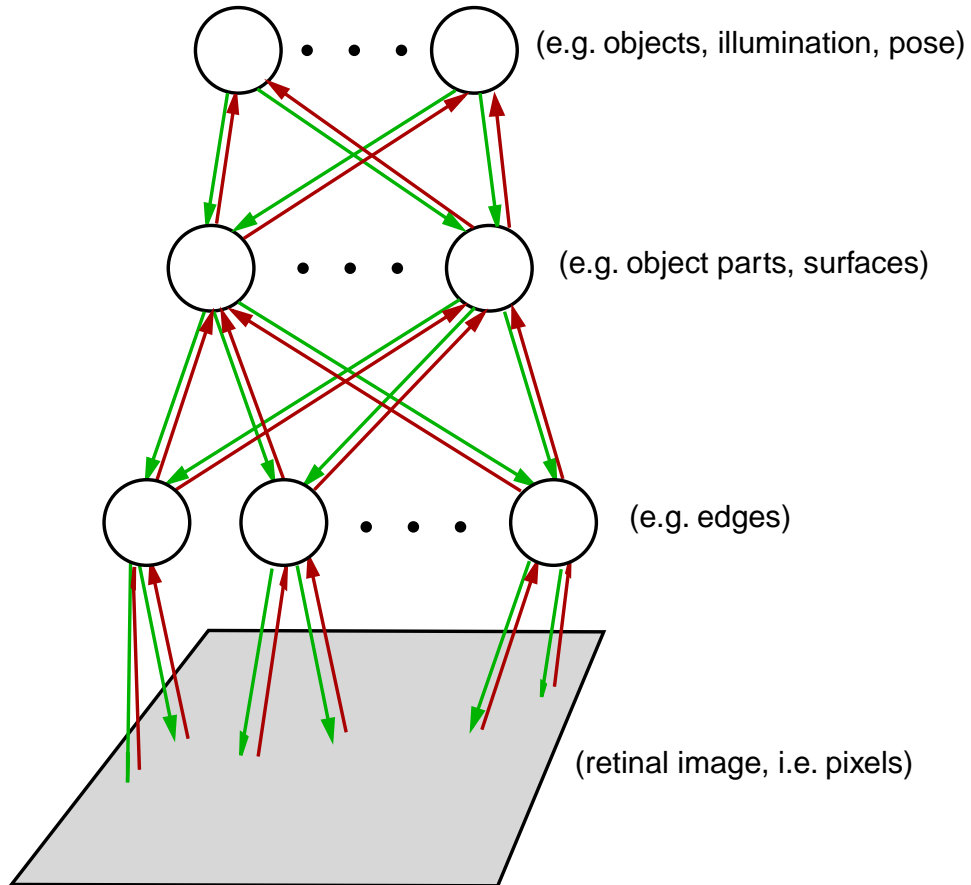


- joint distribution $p(y_{i1}, y_{i2})$ is not Normal.
- conditional distributions $p(y_{i1} | y_{i2})$ and $p(y_{i2} | y_{i1})$ vary, and are difficult to codify.
- easier to describe by an **latent** tri-valued discrete process

$$s_i \sim \text{Discrete}[1/3, 1/3, 1/3]$$
$$\mathbf{y}_i \sim \mathcal{N}[\boldsymbol{\mu}_{s_i}; \Sigma]$$

Latent Variable Models

Explain correlations in \mathbf{y} by assuming some latent variables \mathbf{x}



$$\mathbf{x} \sim \mathcal{P}[\theta_x]$$

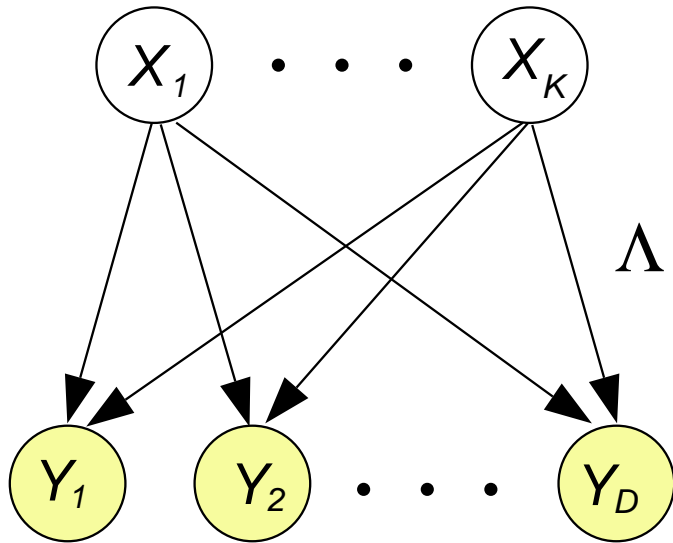
$$\mathbf{y} \mid \mathbf{x} \sim \mathcal{P}[\theta_y]$$

$$p(\mathbf{x}, \mathbf{y}; \theta_x, \theta_y) = p(\mathbf{y} \mid \mathbf{x}; \theta_y)p(\mathbf{x}; \theta_x)$$

$$p(\mathbf{y}; \theta_x, \theta_y) = \int d\mathbf{x} p(\mathbf{y} \mid \mathbf{x}; \theta_y)p(\mathbf{x}; \theta_x)$$

Factor Analysis

Latent variable models are useful even when both latent and observed variables are Gaussian.



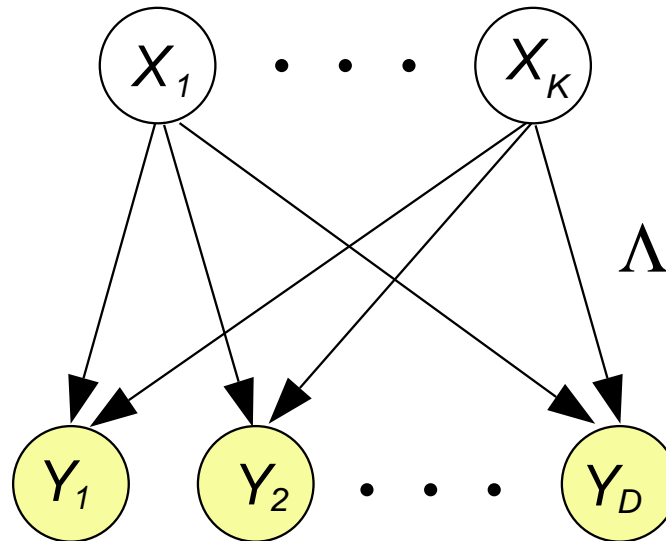
Linear generative model:
$$y_d = \sum_{k=1}^K \Lambda_{dk} x_k + \epsilon_d$$

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

So, \mathbf{y} is Gaussian with:
$$p(\mathbf{y}) = \int p(\mathbf{x})p(\mathbf{y}|\mathbf{x})d\mathbf{x} = \mathcal{N}(0, \Lambda\Lambda^\top + \Psi)$$
 where Λ is a $D \times K$ matrix, and Ψ is diagonal.

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

Factor Analysis (cont.)



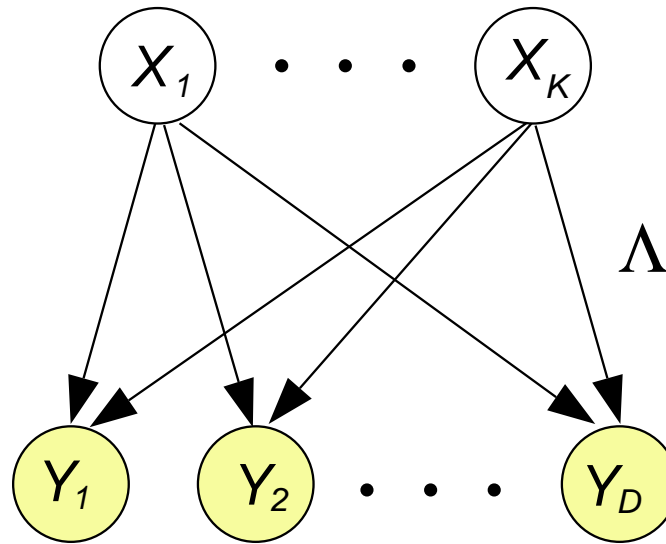
- ML learning finds Λ and Ψ given data

- parameters (corrected for symmetries):
$$DK + D - \frac{K(K-1)}{2} < \frac{D(D+1)}{2}$$

- no closed form solution for ML params: $\mathcal{N}(0, \Lambda\Lambda^\top + \Psi)$

- [Bayesian treatment would also have priors over Λ and Ψ and would average over them for prediction.]

Principal Components Analysis



Noise variable becomes infinitesimal compared to the scale of the data: $\Psi = \lim_{\sigma^2 \rightarrow 0} \sigma^2 I$

Equivalently: reconstruction cost becomes infinite compared to the cost of coding the hidden units under the prior.

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

$$\beta = \lim_{\sigma^2 \rightarrow 0} \Lambda^T (\Lambda\Lambda^T + \sigma^2 I)^{-1} = (\Lambda^T \Lambda)^{-1} \Lambda^T$$

Eigenvalues and Eigenvectors

λ is an **eigenvalue** and \mathbf{x} is an **eigenvector** of A if:

$$A\mathbf{x} = \lambda\mathbf{x}$$

and \mathbf{x} is a unit vector ($\mathbf{x}^\top \mathbf{x} = 1$).

Interpretation: the operation of A in direction \mathbf{x} is a scaling by λ .

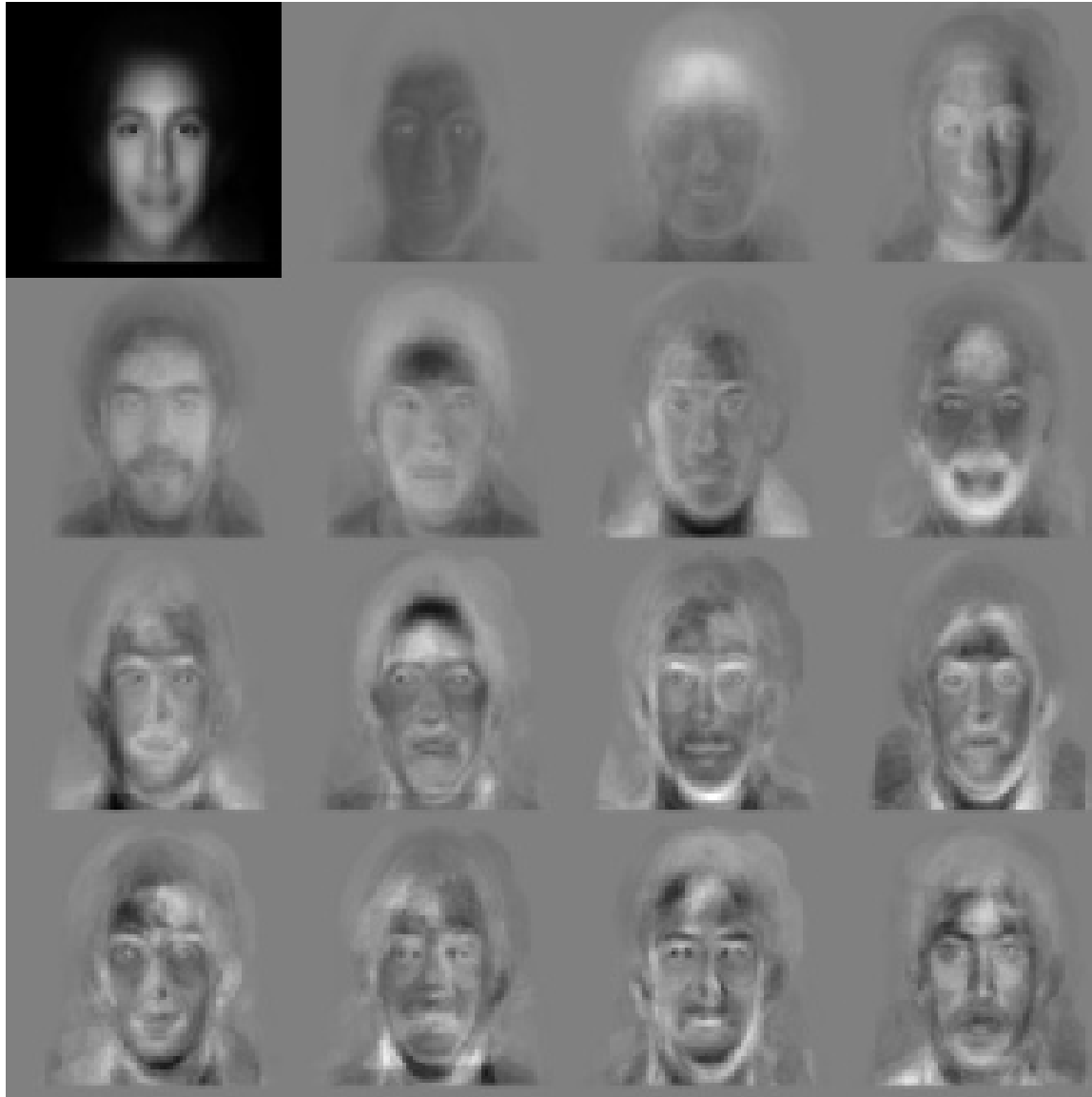
The K Principal Components are the K eigenvectors with the largest eigenvalues of the data covariance matrix (i.e. K directions with the largest variance).

Note: Σ can be decomposed:

$$\Sigma = USU^\top$$

where S is $\text{diag}(\sigma_1^2, \dots, \sigma_D^2)$ and U is a an orthonormal matrix.

Example of PCA: Eigenfaces



from www-white.media.mit.edu/vismod/demos/facerec/basic.html

Mutual Information and PCA

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

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

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

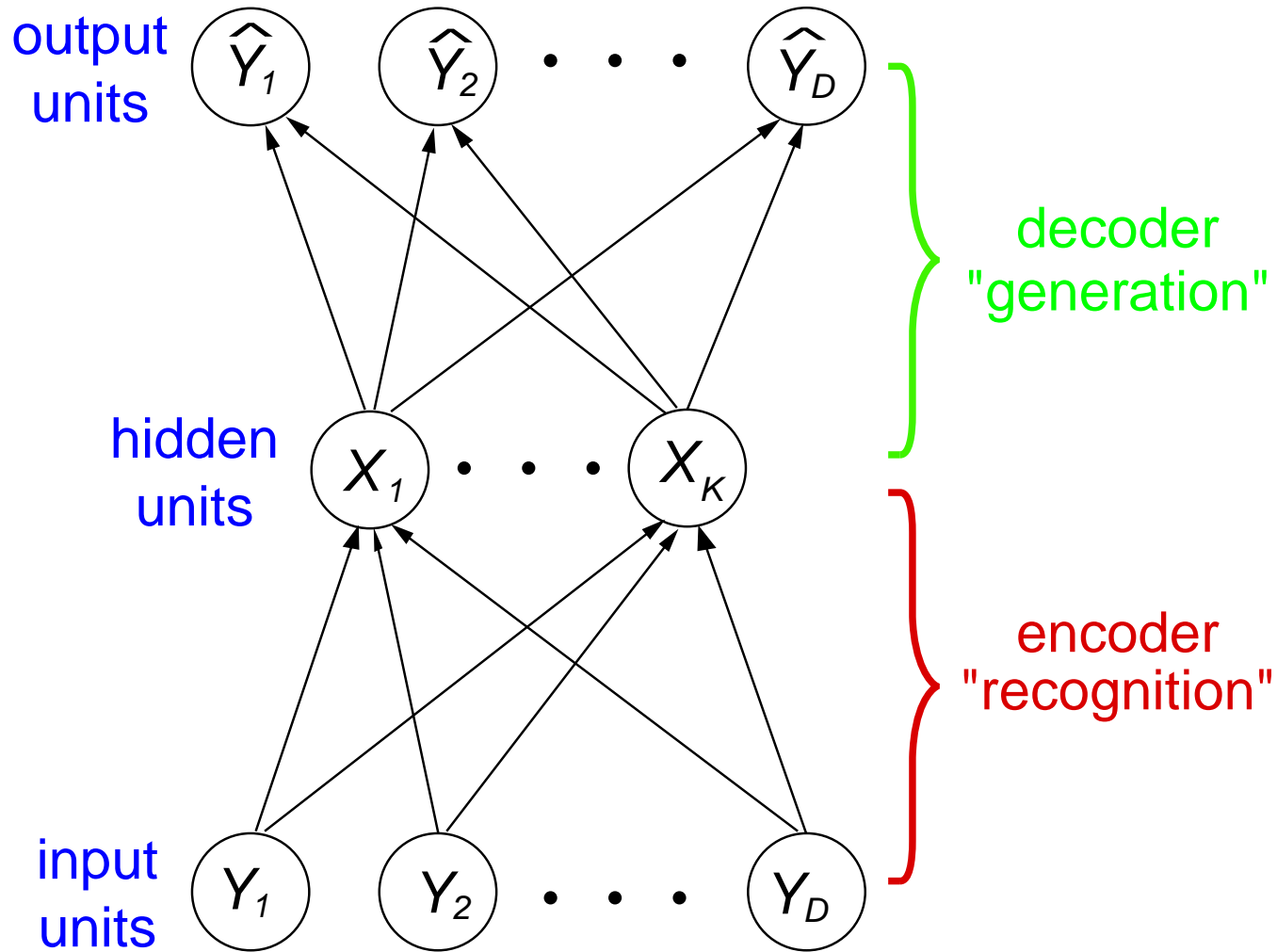
$$H(\mathbf{z}) = - \int d\mathbf{z} p(\mathbf{z}) \ln p(\mathbf{z}) = \frac{1}{2} \ln |\Sigma| + \frac{D}{2} (1 + \ln 2\pi) \quad (2)$$

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

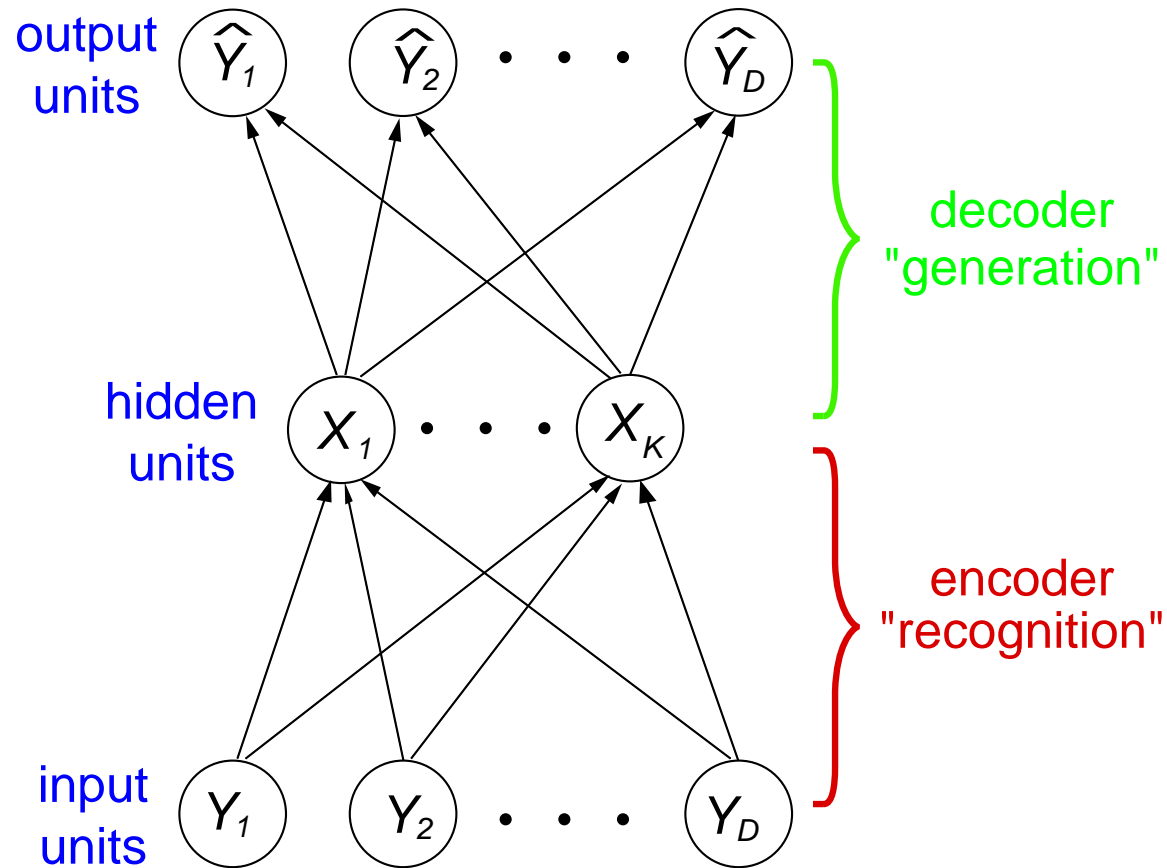
$$\Sigma_x = A\Sigma_y A^\top = A U S_y U^\top A^\top$$

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

Network Interpretations and Encoder-Decoder Duality

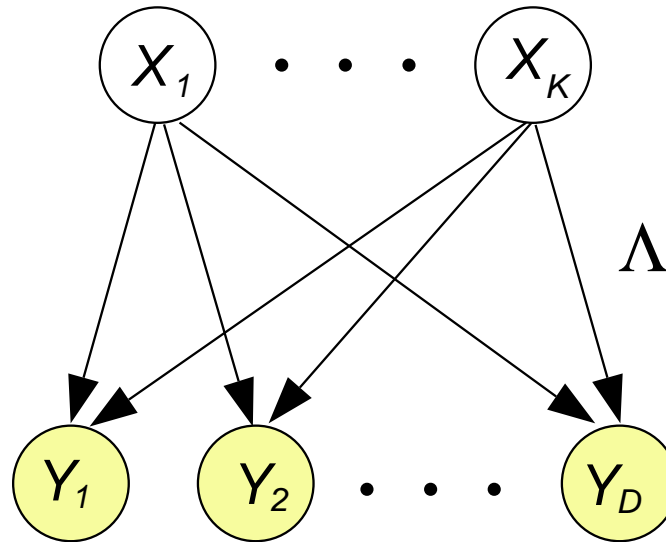


From Supervised Learning to PCA



A linear autoencoder neural network trained to minimise squared error learns to perform PCA (Baldi & Hornik, 1989).

Probabilistic PCA



Linear generative model: $y_d = \sum_{k=1}^K \Lambda_{dk} x_k + \epsilon_d$

- x_k are independent $\mathcal{N}(0, 1)$ Gaussian **factors**
- ϵ_d are independent $\mathcal{N}(0, \sigma^2)$ Gaussian **noise**
- $K < D$

PPCA is factor analysis with isotropic noise: $\Psi = \sigma^2 I$

Finds the same principal subspace as PCA but provides a well-defined probabilistic model.

Gradient Methods of Learning FA

Write down negative log likelihood:

$$\frac{1}{2} \log |2\pi(\Lambda\Lambda^\top + \Psi)| + \frac{1}{2} \mathbf{y}^\top (\Lambda\Lambda^\top + \Psi)^{-1} \mathbf{y}$$

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

We will soon see an easier way to learn latent variable models...

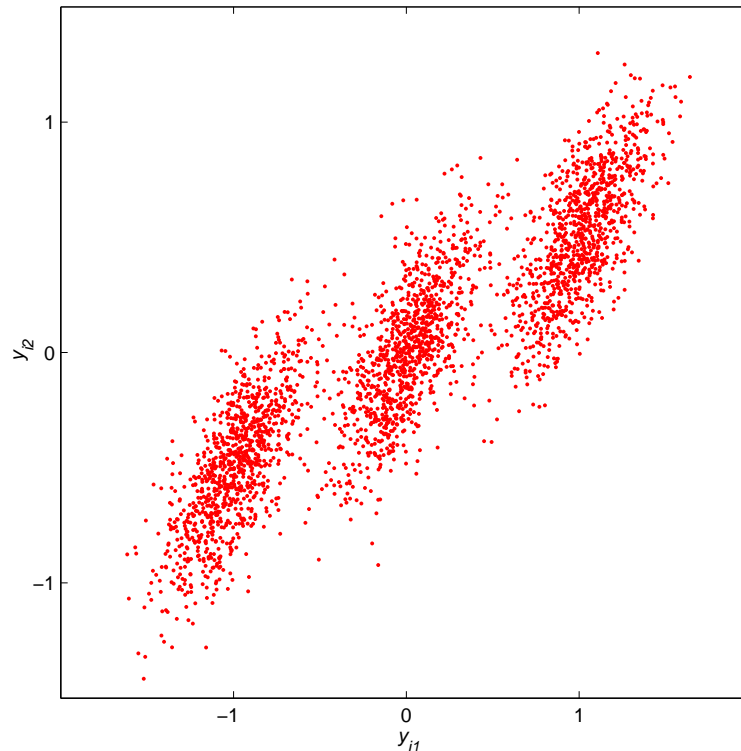
FA vs PCA

- PCA is rotationally invariant; FA is not
- FA is measurement scale invariant; PCA is not
- FA defines a probabilistic model; PCA does not

Limitations of Gaussian, FA and PCA models

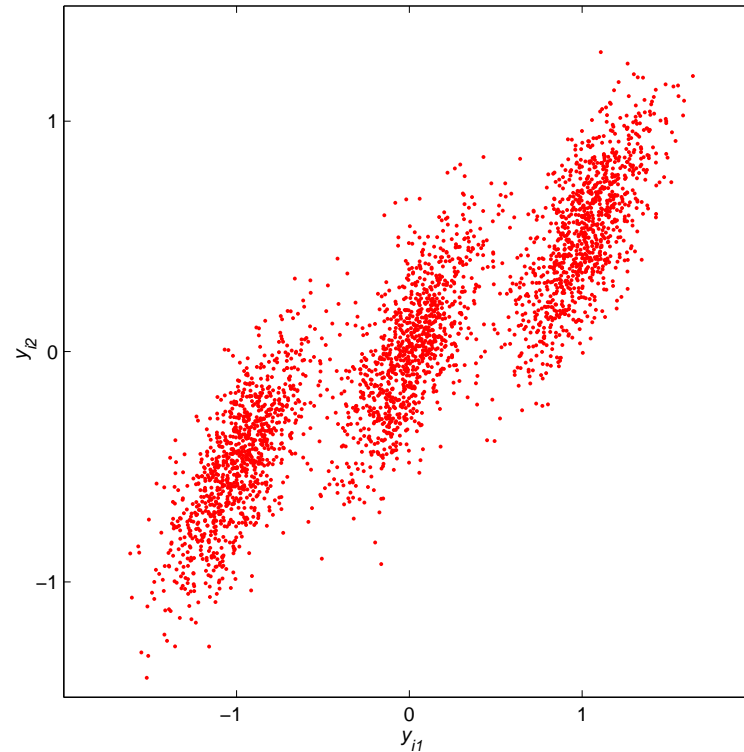
- Gaussian, FA and PCA models are easy to understand and use in practice.
- They are a convenient way to find interesting directions in very high dimensional data sets, eg as preprocessing
- Their problem is that they make very strong assumptions about the distribution of the data, only the mean and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

$$s_i \stackrel{\text{iid}}{\sim} \text{Discrete}[\boldsymbol{\pi}]$$
$$\mathbf{y}_i \mid s_i \sim \mathcal{P}_{s_i}[\boldsymbol{\theta}_{s_i}]$$

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

$$s_i \stackrel{\text{iid}}{\sim} \text{Discrete}[\boldsymbol{\pi}]$$
$$\mathbf{y}_i \mid s_i \sim \mathcal{P}_{s_i}[\boldsymbol{\theta}_{s_i}]$$

Under the discrete distribution

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

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

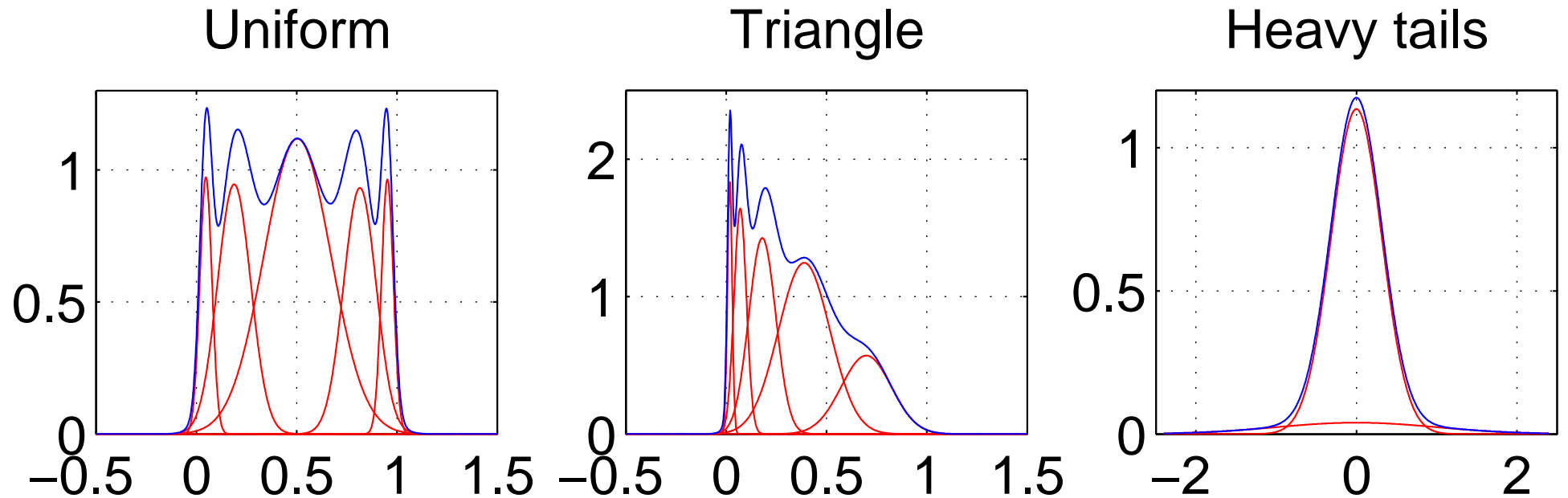
$$P(\mathbf{y}_i) = \sum_{m=1}^k P(\mathbf{y}_i \mid s_i = m) P(s_i = m)$$
$$= \sum_{m=1}^k \pi_m P_m(\mathbf{y}; \boldsymbol{\theta}_m)$$

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

Approximation with a Mixture of Gaussians (MoG)

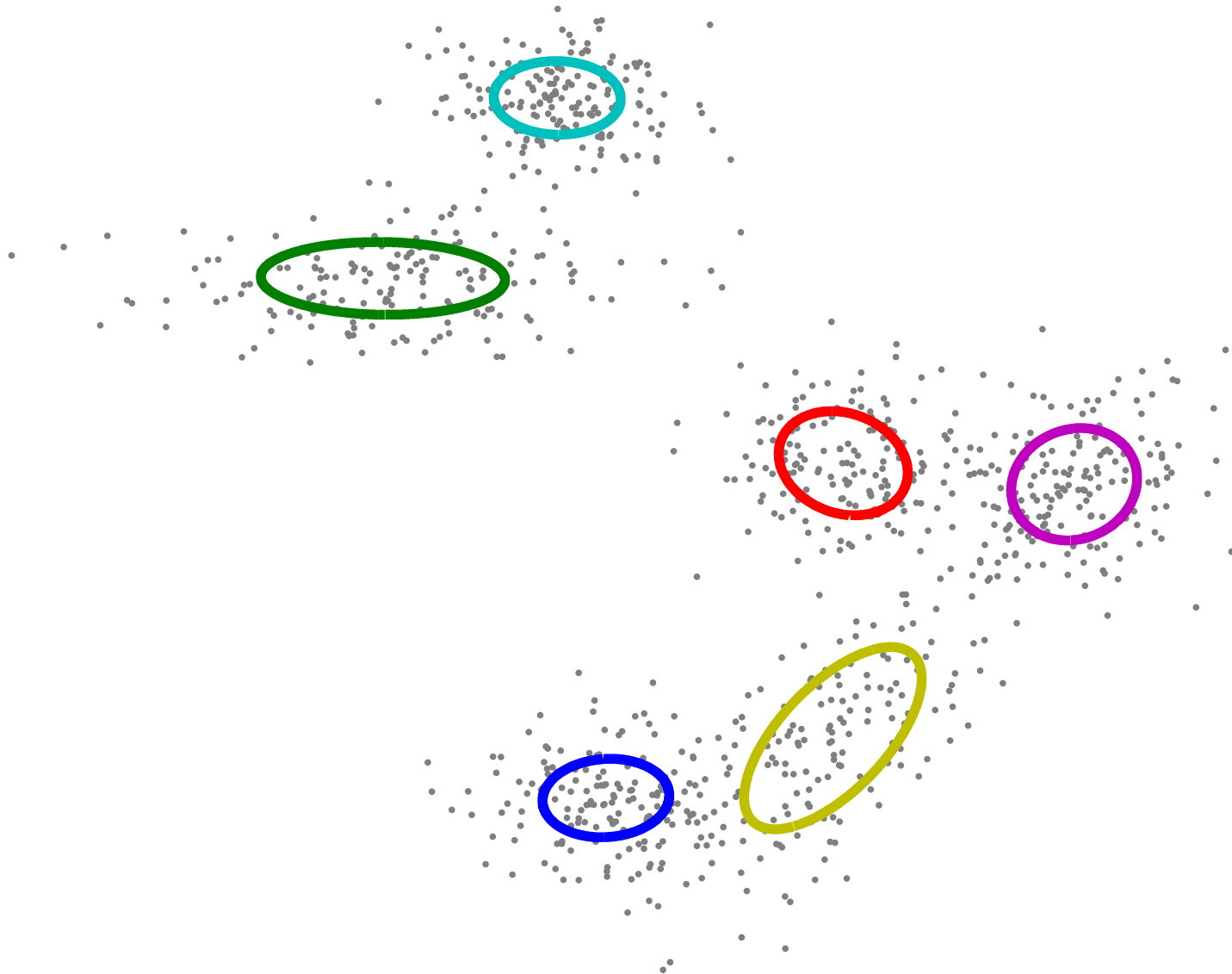
The component densities may be viewed as elements of a *basis* which can be combined to approximate arbitrary distributions.

Here are examples where non-Gaussian densities are modelled (approximated) 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 i th 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 m th cluster are distributed according to the m th component:

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

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

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

This is often called the **responsibility** of the m th cluster for the i th 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, $\mathcal{Y} = \{\mathbf{y}_1 \dots \mathbf{y}_n\}$ (i.e. the likelihood) is

$$\begin{aligned} p(\mathcal{Y} \mid \{\boldsymbol{\mu}_m\}, \{\Sigma_m\}, \boldsymbol{\pi}) &= \prod_{i=1}^n \sum_{m=1}^k \pi_m \mathcal{N}[\mathbf{y}_i \mid \boldsymbol{\mu}_m; \Sigma_m] \\ &= \prod_{i=1}^n \sum_{m=1}^k \pi_m |2\pi\Sigma_m|^{-1/2} \exp \left[-\frac{1}{2}(\mathbf{y}_i - \boldsymbol{\mu}_m)^\top \Sigma_m^{-1}(\mathbf{y}_i - \boldsymbol{\mu}_m) \right] \end{aligned}$$

The log of the likelihood is

$$\log p(\mathcal{Y} \mid \{\boldsymbol{\mu}_m\}, \{\Sigma_m\}, \boldsymbol{\pi}) = \sum_{i=1}^n \log \sum_{m=1}^k \pi_m |2\pi\Sigma_m|^{-1/2} \exp \left[-\frac{1}{2}(\mathbf{y}_i - \boldsymbol{\mu}_m)^\top \Sigma_m^{-1}(\mathbf{y}_i - \boldsymbol{\mu}_m) \right]$$

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

Its partial derivative wrt θ_m is

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

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

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

And its partial derivative wrt π_m is

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

MoG Derivatives

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

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_m} = \sum_{i=1}^n r_{im} \Sigma_m^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_m)$$

$$\frac{\partial \mathcal{L}}{\partial \Sigma_m^{-1}} = \frac{1}{2} \sum_{i=1}^n r_{im} (\Sigma_m - (\mathbf{y}_i - \boldsymbol{\mu}_m)(\mathbf{y}_i - \boldsymbol{\mu}_m)^\top)$$

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 $\sigma^2 \rightarrow 0$. Then the responsibilities become binary

$$r_{im} \rightarrow \delta(m, \underset{l}{\operatorname{argmin}} \|\mathbf{y}_i - \boldsymbol{\mu}_l\|^2)$$

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 $\left(\text{set } r_{im} = \delta(m, \underset{l}{\operatorname{argmin}} \|\mathbf{y}_i - \boldsymbol{\mu}_l\|^2) \right)$
- update the means to the average of their assigned points $\left(\text{set } \boldsymbol{\mu}_m = \frac{\sum_i r_{im} \mathbf{y}_i}{\sum_i r_{im}} \right)$

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:

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

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.