Probabilistic Machine Learning

Clustering: k-means and Mixtures of Gaussians
The EM Algorithm and (perhaps some) Factor Analysis

Joaquin Quiñonero Candela
joaquinc@microsoft.com

Applied Games Group
Microsoft Research Cambridge, UK

PASCAL Bootcamp in Machine Learning
Vilanova i la Geltrú, July 5 and 6, 2007

Three Types of Learning

Imagine an organism or machine which experiences a series of sensory inputs:

$$x_1, x_2, x_3, x_4, \ldots$$

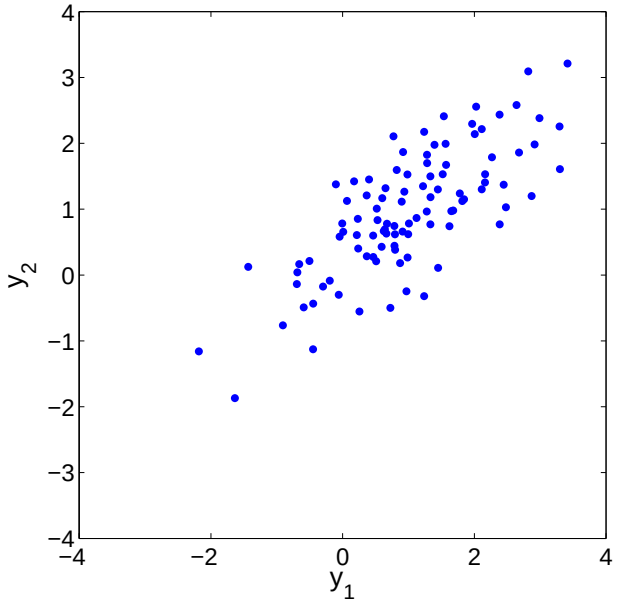
Supervised learning: The machine is also given desired outputs $y_1, y_2, \ldots$, and its goal is to learn to produce the correct output given a new input.

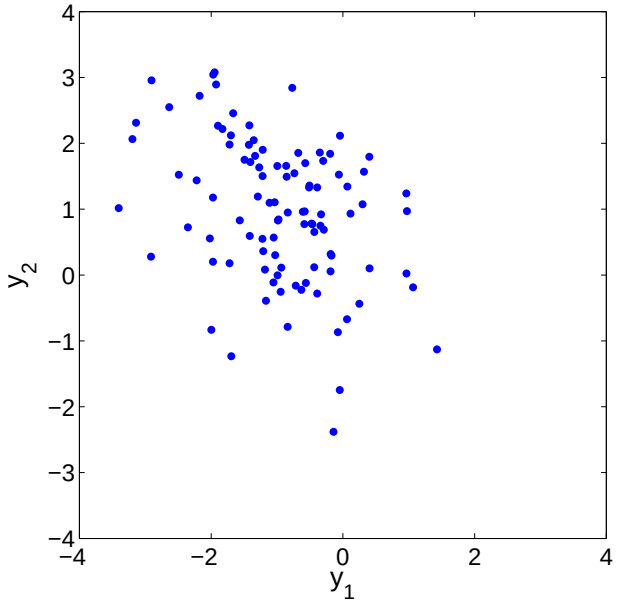
Unsupervised learning: The goal of the machine is to build a model of x that can be used for reasoning, decision making, predicting things, communicating etc.

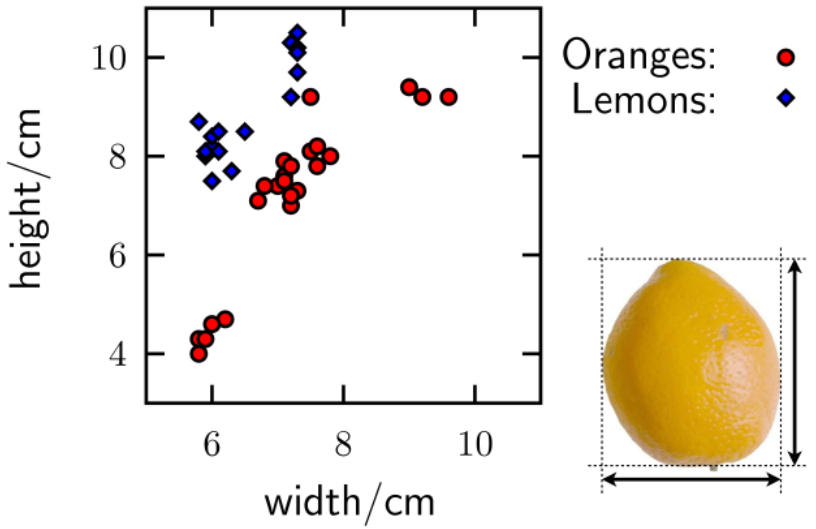
Reinforcement learning: The machine can also produce actions $a_1, a_2, \ldots$ which affect the state of the world, and receives rewards (or punishments) $r_1, r_2, \ldots$. Its goal is to learn to act in a way that maximises rewards in the long term.

Datasets

Some simple datasets in $\mathbb{R}^2$:





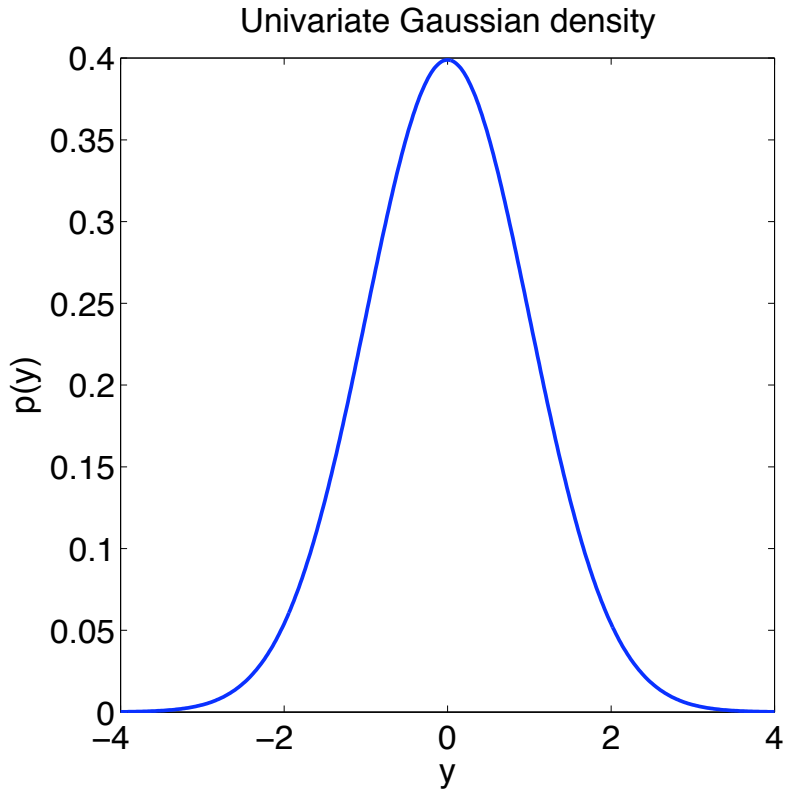


A slightly less boring dataset in $\mathbb{R}^2$:

A Very Simple Model

Univariate Gaussian Density $y \in \mathbb{R}$:

$$p(y|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)$$



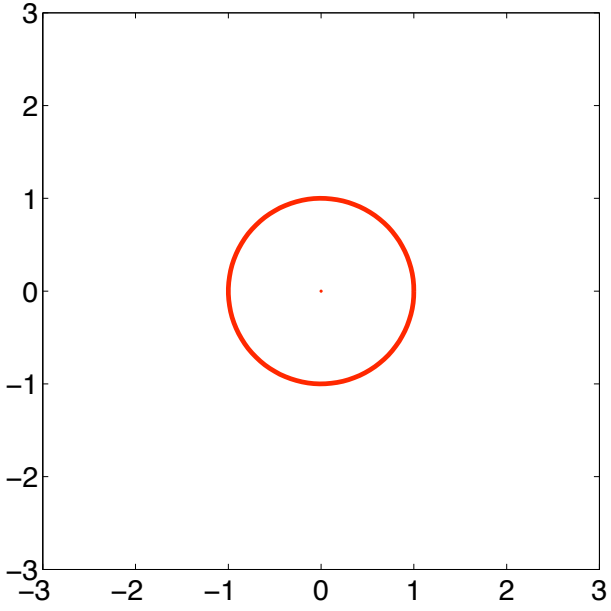
This model has parameters $\boldsymbol{\theta} = \{\mu, \sigma\}$, which model the mean and the standard deviation of the data, respectively.

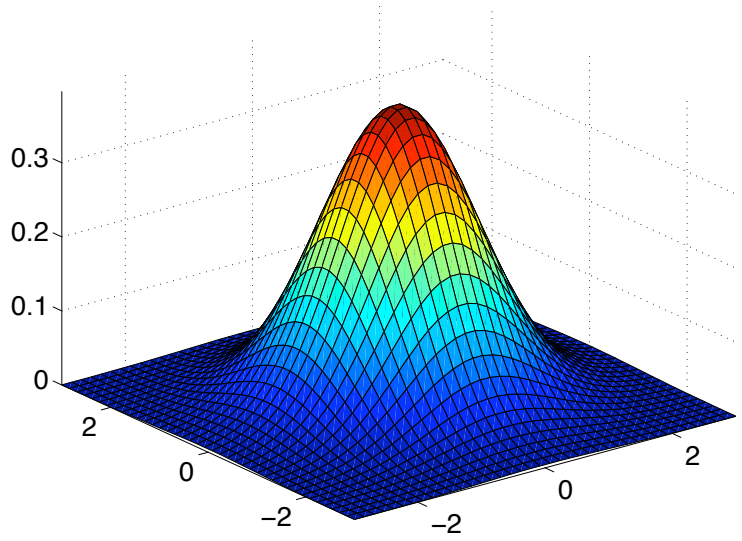
A Less Simple Model

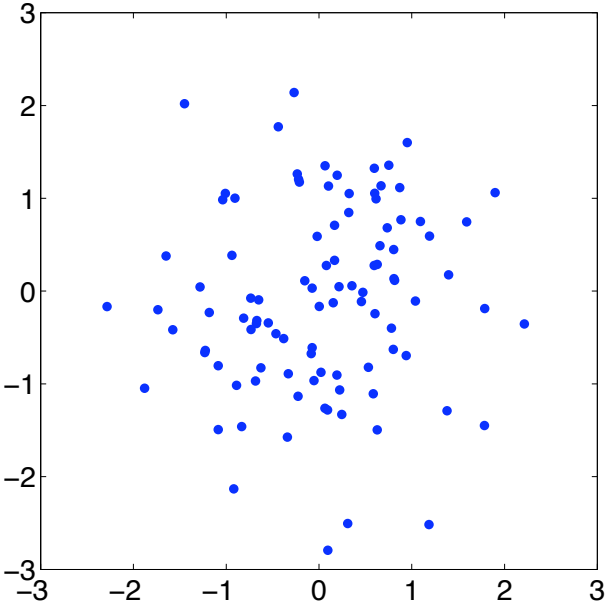
Multivariate Gaussian Density $y \in \mathbb{R}^D$:

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

$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



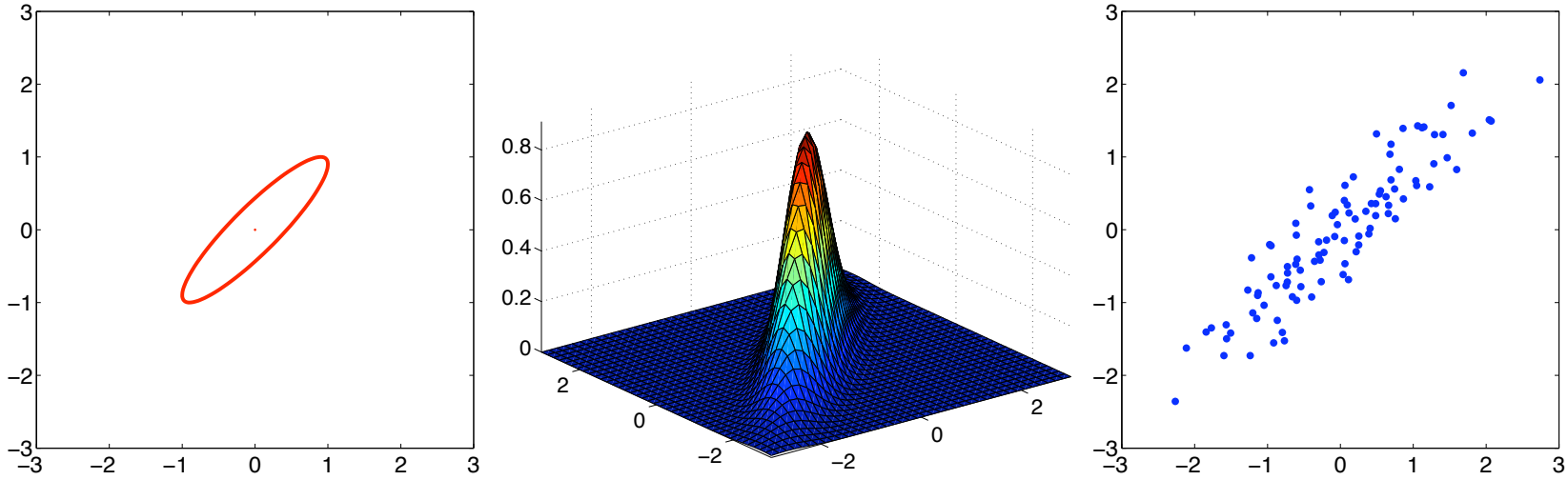




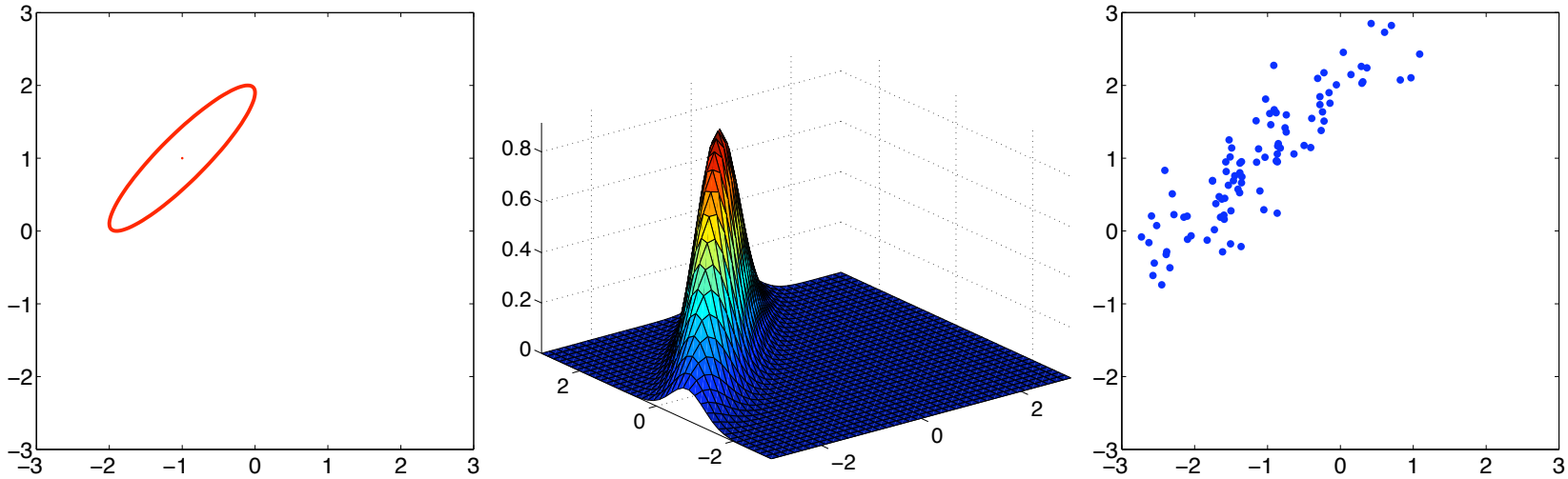
This model has parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \Sigma\}$, which model the mean and covariance matrix

A Less Simple Model: A Multivariate Gaussian

$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix}$$



$$\boldsymbol{\mu} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix}$$



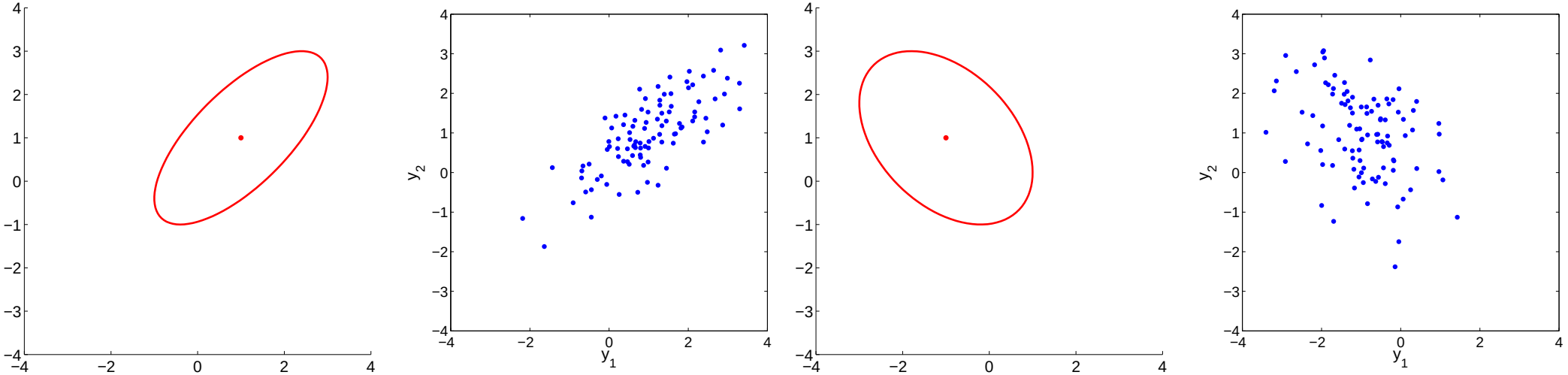
Generating Samples from a Multivariate Gaussian

We know how to generate independent samples: $\mathbf{z} \sim \mathcal{N}(0, I)$.
How do we generate samples with covariance Σ? Find the appropriate rotation.

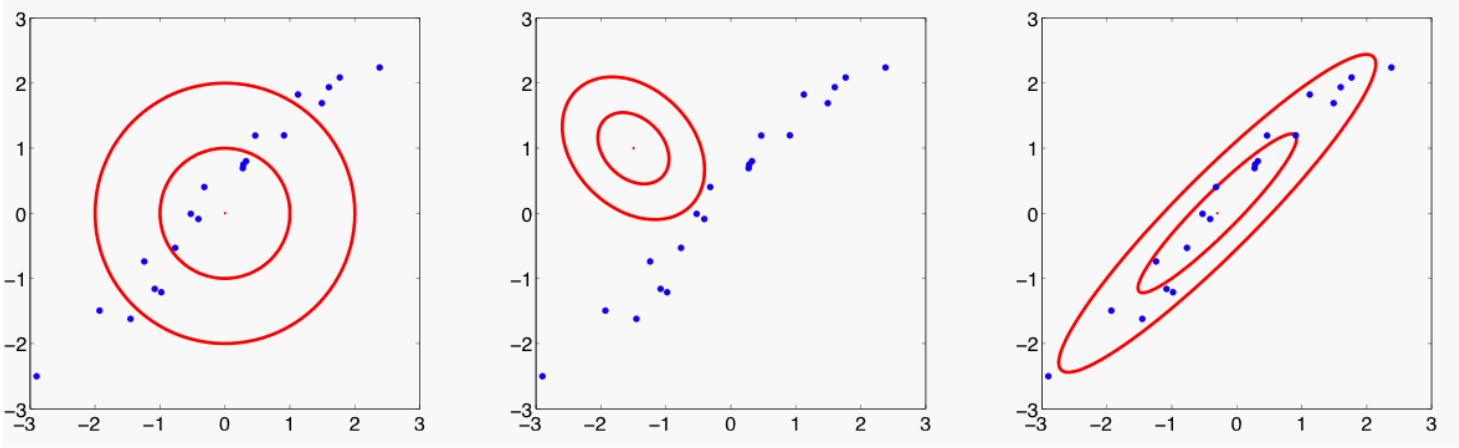
Take the eigen-decomposition of the covariance matrix $\Sigma = U\,D\,U^\top$.
U contains the eigenvectors as columns: $U^\top U = I$. Orthogonal base: $L = U D^{\frac{1}{2}}$.

Sample independently and then rotate:

$$\mathbf{z} \sim \mathcal{N}(0, I) \quad \text{and} \quad \mathbf{y} = L\mathbf{z};, \quad \Rightarrow \quad \text{cov}(\mathbf{y}) = \mathbb{E}(\mathbf{y}\mathbf{y}^\top) = LL^\top = \Sigma\,.$$



Fitting the Model to Data



Assuming the data were generated independently, the likelihood of the model is:

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{i=1}^{n} p(\mathbf{y}_i|\boldsymbol{\theta})$$

From left to right: clearly the 3rd model is the best fit to the data

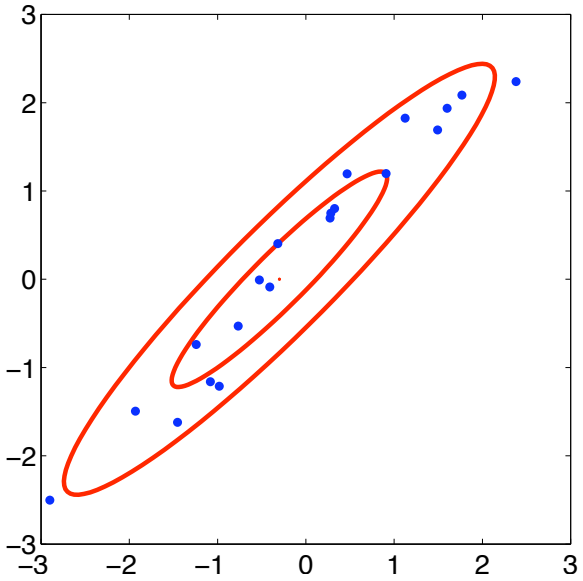
$$\begin{aligned} \log p(\mathcal{D}|\boldsymbol{\theta}_1) &= -55.38 \\ \log p(\mathcal{D}|\boldsymbol{\theta}_2) &= -238.29 \\ \log p(\mathcal{D}|\boldsymbol{\theta}_3) &= -22.14 \end{aligned}$$

The Likelihood Function

The likelihood $p(\mathcal{D}|\boldsymbol{\theta}) = p(\{\mathbf{y}_1, \ldots, \mathbf{y}_n\}|\boldsymbol{\mu}, \Sigma) = \prod_{i=1}^{n} p(\mathbf{y}_i|\boldsymbol{\mu}, \Sigma)$ is a function of the model parameters $\boldsymbol{\theta}$

The maximum likelihood (ML) procedure finds parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \Sigma\}$ such that:

$$\boldsymbol{\theta}_{\mathrm{ML}} = \mathrm{argmax}_{\boldsymbol{\theta}}\, p(\mathcal{D}|\boldsymbol{\theta})$$



Maximum Likelihood Estimate for a Gaussian

Likelihood is p(data|model): $p(\mathcal{D}|\boldsymbol{\mu}, \Sigma) = \prod_{n=1}^{N} p(\mathbf{y}_n|\boldsymbol{\mu}, \Sigma)$

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

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

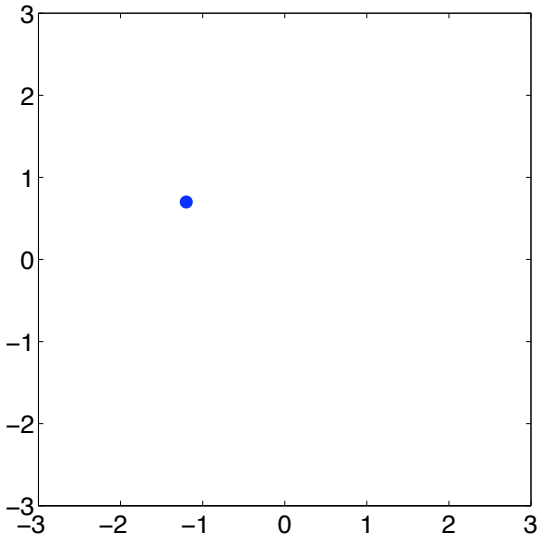
Note: equivalently, minimise $-\mathcal{L}$, which is *quadratic* in $\boldsymbol{\mu}$
Procedure: take derivatives and set to zero:

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

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

Dangers of the Maximum Likelihood Procedure

What is the ML estimate of the model parameters for this dataset?



Is this reasonable?

Bayesian Learning

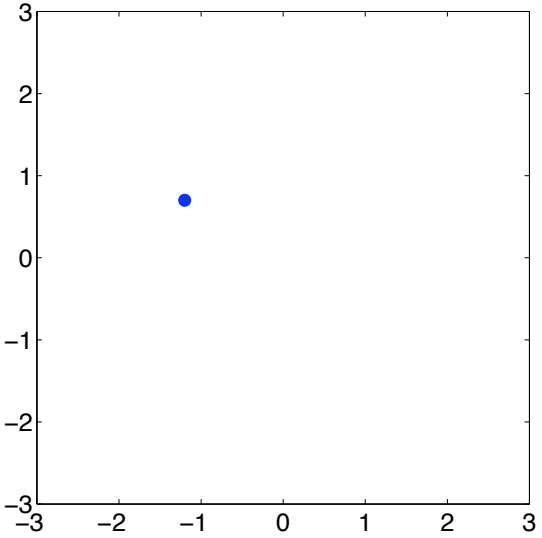
We make prior assumptions on the value of the parameters before we see the data.
We then apply the basic rules of probability theory.

- Prior distribution over the parameters: $p(\boldsymbol{\theta})$

- Model of the data given the parameters, likelihood function: $p(\mathcal{D}|\boldsymbol{\theta})$

- Posterior distribution of model parameters:

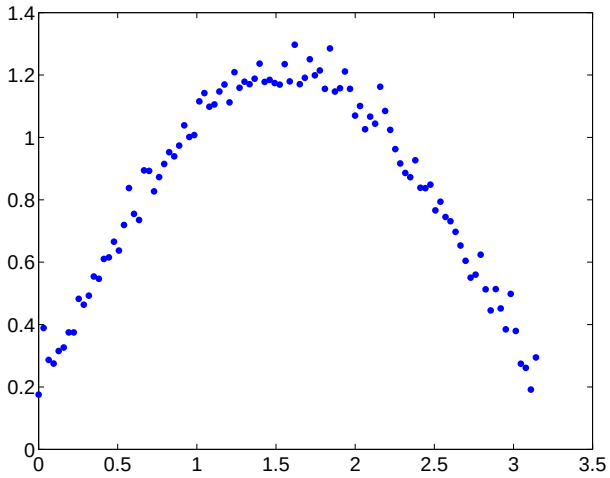
$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})}$$



Limitations of Multivariate Gaussians

Gaussians are fundamental and widespread, but not all datasets conform to a Gaussian distribution.

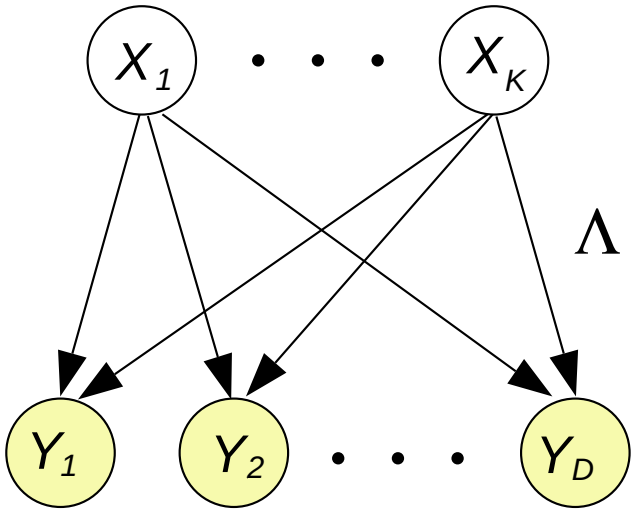
- Restriction to linear relations plus Gaussian noise
- There might exist outliers in the data (heavy-tailed noise)
- The data might have non-linear structure



- Even if data is Gaussian, if D is large the full multivariate Gaussian model might be difficult to handle: $D(D+1)/2$ parameters! dimensionality reduction

Factor Analysis

Model the observed data $\mathbf{y}$ as a linear combination of a smaller number of generating, latent factors $\mathbf{x}$.



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
- $K < D$

Properties:

- $p(\mathbf{x}) \sim \mathcal{N}(0, I)$ and $\mathbf{y} = \Lambda\mathbf{x} + \epsilon$
- Since $p(\epsilon) = \mathcal{N}(0, \Psi)$, we get $p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\Lambda\mathbf{x}, \Psi)$
- $p(\mathbf{y}) = \int p(\mathbf{x})p(\mathbf{y}|\mathbf{x})d\mathbf{x} = \mathcal{N}(0, \Lambda\Lambda^\top + \Psi)$ where Λ is a $D \times K$ matrix, and Ψ is diagonal.

latent = hidden = unobserved = missing

Digesting Factor Analysis

$$\mathbf{y} = \Lambda\mathbf{x} + \boldsymbol{\epsilon}$$

$\mathbf{y}$ is one $D \times 1$ observed vector from our dataset $\mathcal{D} = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$.
$\mathbf{x}$ is a $K \times 1$ vector of latent factors with $K < D$.

The prior over factors is $p(\mathbf{x}) = \mathcal{N}(0, I)$.

The noise is independent across dimensions: $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \Psi)$, where Ψ is diagonal.
Therefore the conditional likelihood given the factors is:

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

The (marginal) likelihood is obtained by **integrating over** the unknown factors:

$$p(\mathbf{y}|\Lambda, \Psi) = \int p(\mathbf{y}|\mathbf{x}, \Lambda, \Psi)p(\mathbf{x})\mathrm{d}\mathbf{x} = \mathcal{N}(0, \Lambda\Lambda^\top + \Psi)$$

This is the likelihood we would use to **learn** the parameters $\boldsymbol{\theta} = \{\Lambda, \Psi\}$.

Digesting Factor Analysis (2)

The posterior distribution over the factor $\mathbf{x}$, given an observation $\mathbf{y}_i$ is given by:

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

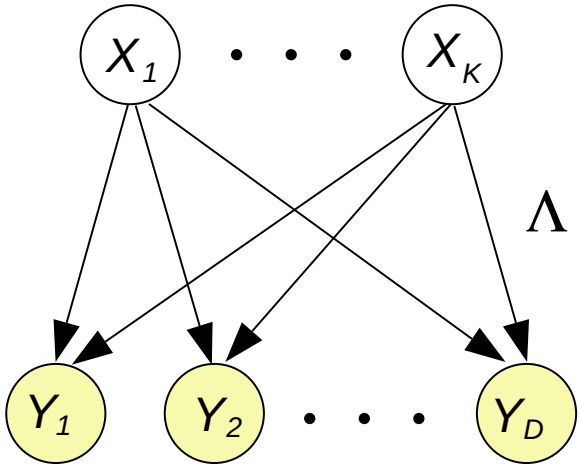
where $\beta = \Lambda^\top(\Lambda\Lambda^\top + \Psi)^{-1}$.

Ways of Thinking about Factor Analysis (FA)

- FA models high dimensional data in terms of a linear transformation of a smaller number (K) of latent factors (white sources of randomness)
- FA is a way of parameterizing a covariance matrix in terms of a smaller number of parameters: $\Sigma = \Lambda\Lambda^\top + \Psi$ ($K \times D + D$ instead of $D \times D$). This allows modelling high dimensional data.
- FA is a method for finding correlations between the observed variables.
- FA is a model for linear regression, where the inputs are hidden (latent)
- FA performs dimensionality reduction: the posterior distribution over factors is probabilistic low-dimensional projection of high dimensional data that captures the correlation structure of the data:

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})} = \mathcal{N}(\beta\mathbf{y}, I - \beta\Lambda) \quad \text{where} \quad \beta = \Lambda^\top(\Lambda\Lambda^\top + \Psi)^{-1}$$

More on Factor Analysis



Relation to Multivariate Gaussian $\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$

- $\boldsymbol{\mu} = 0$
- $\Sigma \approx \Lambda\Lambda^\top + \Psi$

- The covariance of the data does not change if we rotate the sources $\Lambda \rightarrow \Lambda R$, where $RR^\top = I$. We can only hope to find Λ up to a rotation of the factors.
- Number of free parameters (corrected for symmetries):

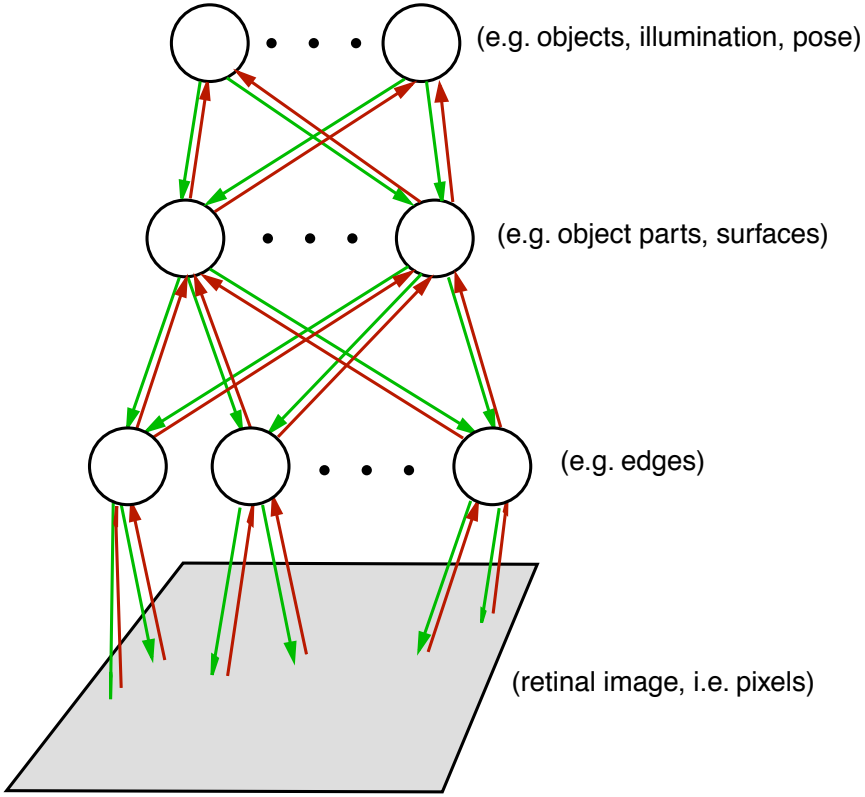
$$DK + D - \frac{K(K-1)}{2} < \frac{D(D+1)}{2}$$

- A Bayesian treatment of FA would start with priors over Λ and Ψ and infer their posterior given the data.

$$p(\Lambda, \Psi|\mathcal{D}) = \frac{p(\mathcal{D}|\Lambda, \Psi)p(\Lambda, \Psi)}{p(\mathcal{D})}$$

Latent Variable Models

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



$$\mathbf{x} \sim p(\mathbf{x}|\boldsymbol{\theta}_{\mathbf{x}})$$

$$\mathbf{y} \sim p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}_{\mathbf{y}})$$

$$p(\mathbf{x}, \mathbf{y}|\boldsymbol{\theta}_{\mathbf{x}}, \boldsymbol{\theta}_{\mathbf{y}}) = p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}_{\mathbf{y}})p(\mathbf{x}|\boldsymbol{\theta}_{\mathbf{x}})$$

$$p(\mathbf{y}|\boldsymbol{\theta}_{\mathbf{x}}, \boldsymbol{\theta}_{\mathbf{y}}) = \int p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}_{\mathbf{y}})p(\mathbf{x}|\boldsymbol{\theta}_{\mathbf{x}})\mathrm{d}\mathbf{x}$$

Gradient Methods of Learning FA

Write down negative log likelihood:

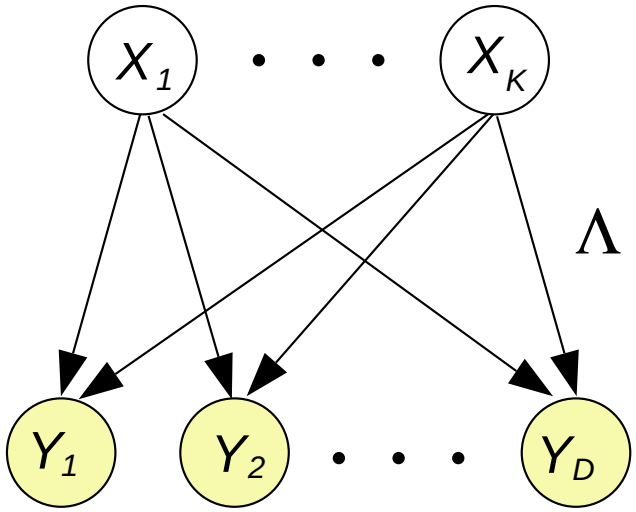
$$\frac{1}{2}\log|2\pi(\Lambda\Lambda^\top+\Psi)|+\frac{1}{2}\mathbf{y}^\top(\Lambda\Lambda^\top+\Psi)^{-1}\mathbf{y}$$

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

There is an easier way to learn latent variable models...

... the Expectation-Maximization (EM) algorithm we will study tomorrow!

Probabilistic Principal Components Analysis (PPCA)



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

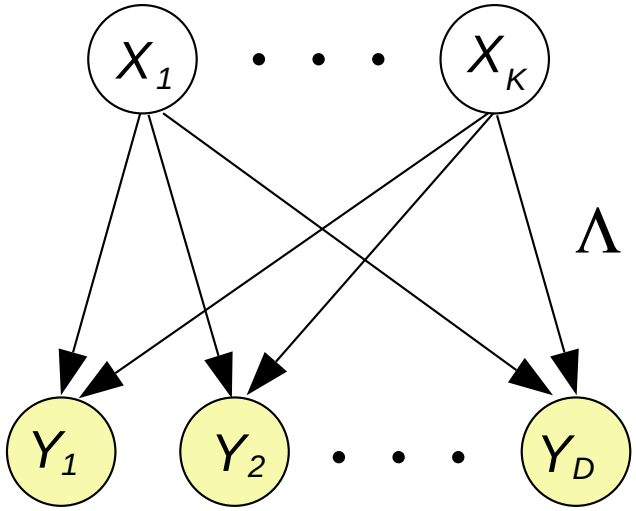
- x_k are independent $\mathcal{N}(0,1)$ Gaussian factors
- ϵ_d are independent $\mathcal{N}(0,\sigma^2)$ Gaussian noise
- $K < D$

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.

(Mike Tipping and Chris Bishop, J. of the Royal Statistical Society, Series B, 1999)

Principal Components Analysis



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

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

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

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

In PCA we choose the columns of Λ to be orthonormal, $\Lambda^\top\Lambda = I$, therefore:

$$\beta = \Lambda^\top$$

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.

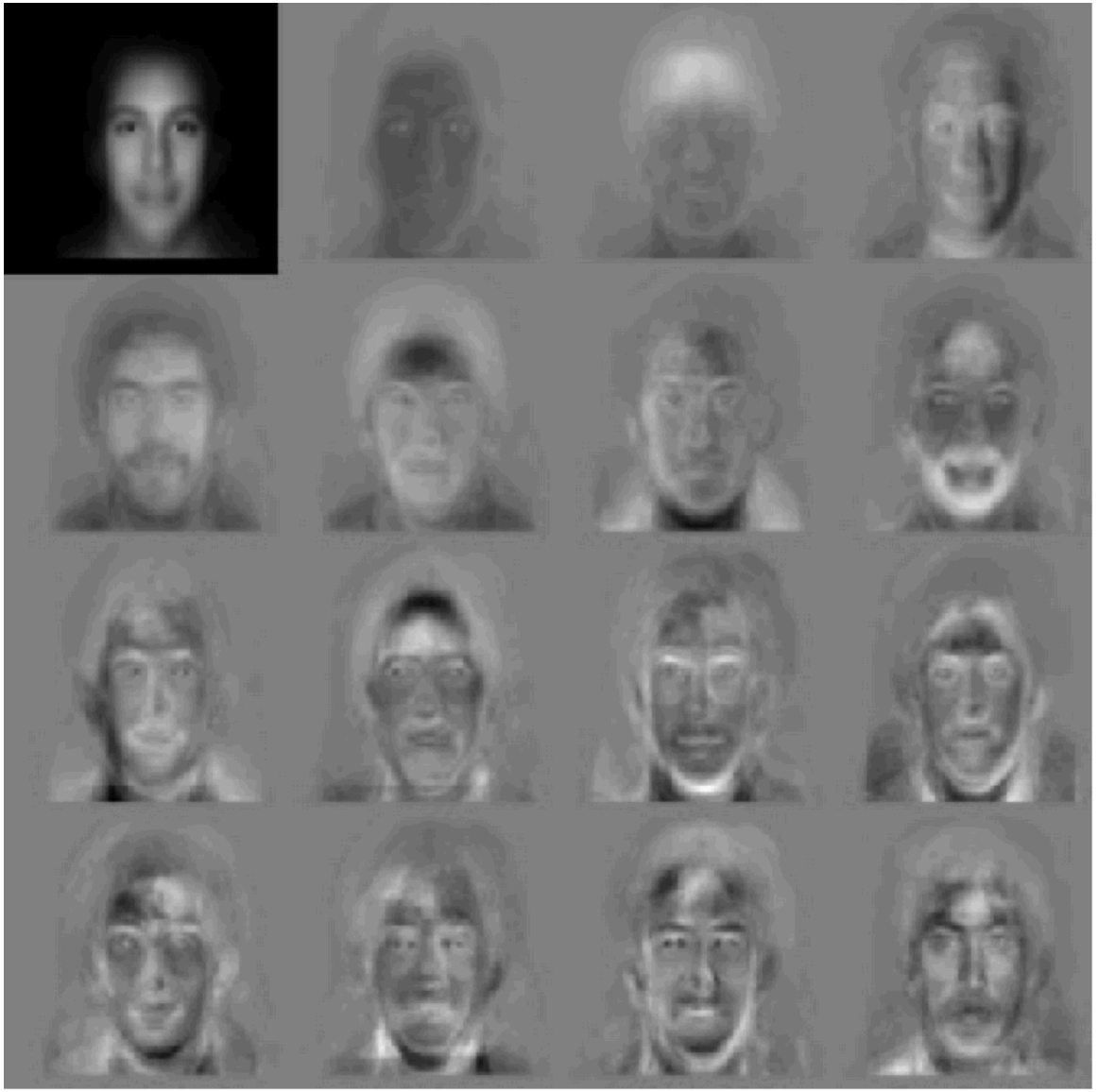
The K Principal Components are the K eigenvectors with the largest eigenvalues of the data covariance matrix (i.e. K directions with the largest variance).

Note: Σ can be decomposed:

$$\Sigma = USU^\top$$

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

Example of PCA: Eigenfaces



from www.media.mit.edu

Another View on PCA

PCA finds the optimal linear projection, that minimizes a linear reconstruction mean squared error:

$$E = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{y}_i - A\mathbf{x}_i\|^2 \qquad \mathbf{x}_i = P^\top \mathbf{y}_i$$

Spring model (Roweis, NIPS*98)

- project the data onto the current estimate of the subplane
- fix the subplane at the origin so that it still can rotate
- attach springs between the projected points and the original samples and relax the system: it will equilibrate at the newly estimated position of the subplane.

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 maximize the entropy of $\mathbf{x}$. 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 $\mathbf{x}$ to have largest volume (i.e. det of covariance matrix).

$$\Sigma_x = A\Sigma_y A^\top = AUS_yU^\top A^\top$$

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

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

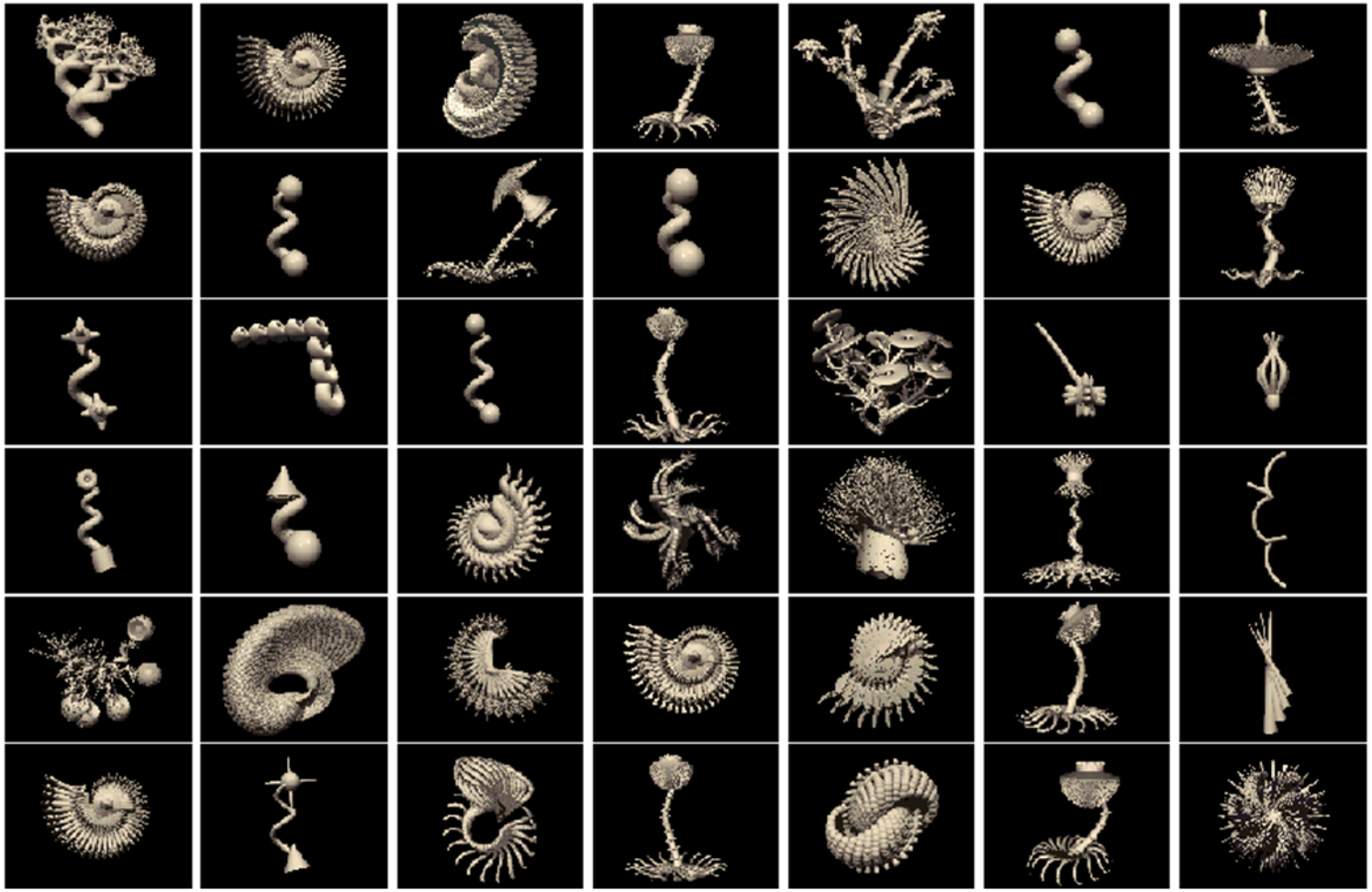
Limitations of Gaussian, FA and PCA models

- 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.

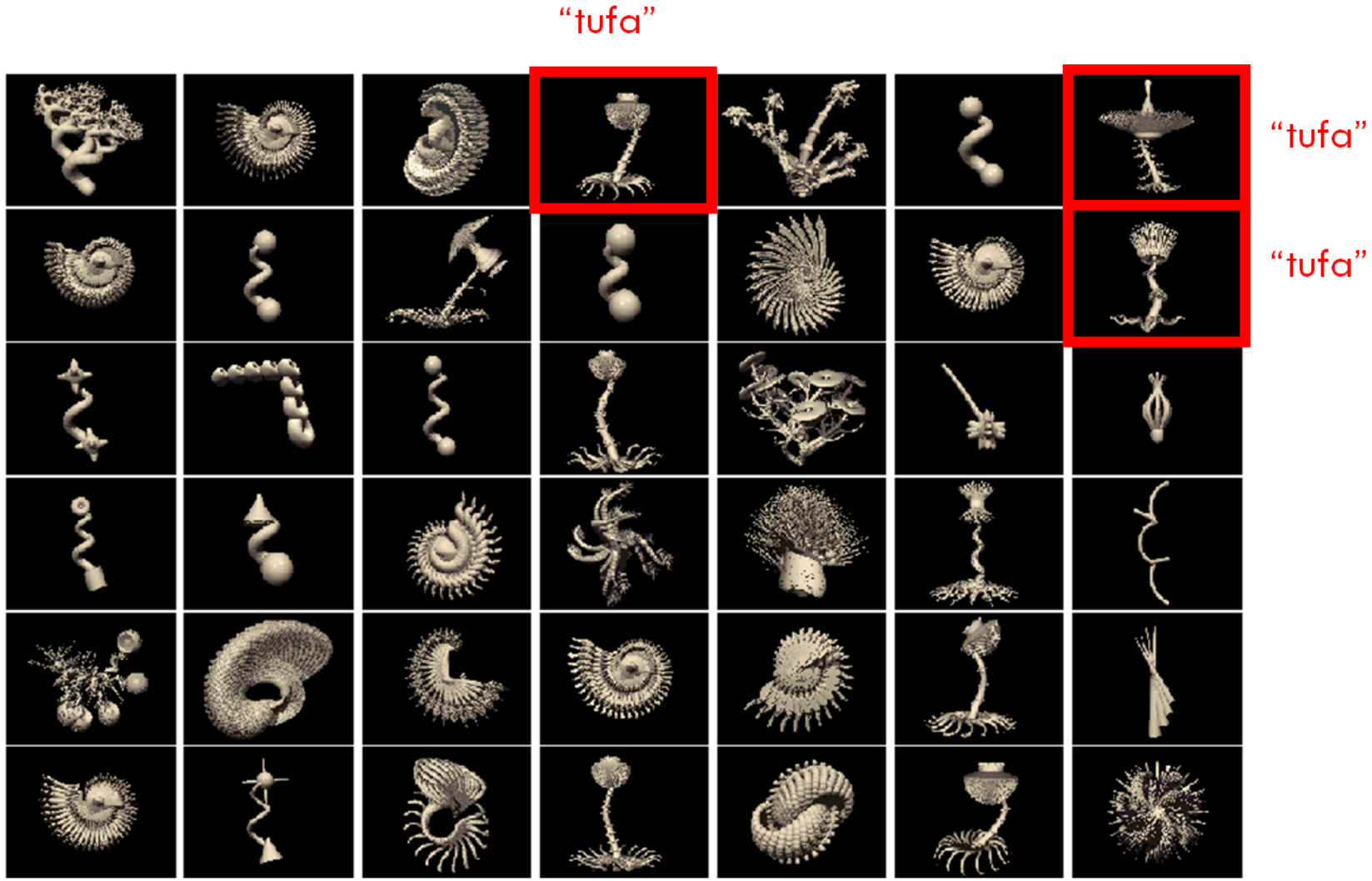
The class of densities which can be modelled is too restrictive.

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

Clustering



Clustering



(thanks to Josh Tenenbaum)

Prologue to Mixtures of Gaussians

Clustering: put a set of N objects into K groups that are similar to each other.
(things that are brown and run away, and things that are green and don't run away)

Why cluster?:
- predictive power: better description of our data, better actions
- compression for communications (vector quantization, K-means clustering)
- detection of relevant (surprising) objects
- models for learning processes in neural systems

Measure of expected distorsion

$$D = \sum_{\mathbf{y}} p(\mathbf{y}) \|\mathbf{m}_{k(\mathbf{y})} - \mathbf{y}\|^2$$

where $k(\mathbf{y}) \in \{1, \ldots, K\}$ is the cluster $\mathbf{y}$ is assigned to, and $\mathbf{m}_{k(\mathbf{y})}$ its mean.

Note that we could use other distances, and other prototypes.

The K-Means Algorithm

Initialization: set the K means $\mathbf{m}_k$ to random values.

Assignment step: Assign each data point i to the nearest mean.

$$k(i) = \text{argmin}_k \, \|\mathbf{y}_i - \mathbf{m}_k\|^2$$

Update the responsibility indicators r_{ij} of cluster j for point i:

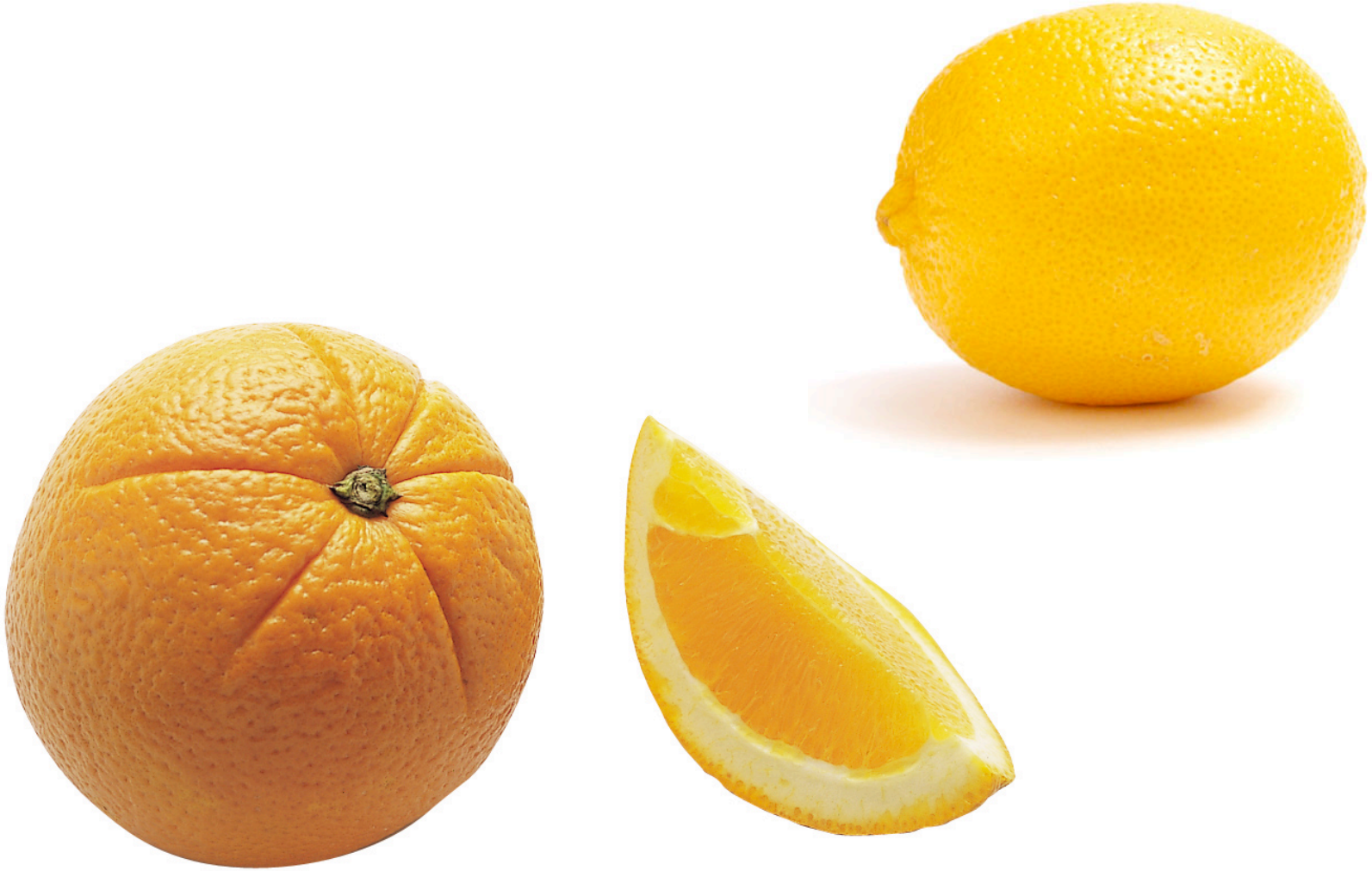
$$r_{ij} = \begin{cases} 1 & \text{if} \quad k(i) = j \\ 0 & \text{if} \quad k(i) \neq j \end{cases}$$

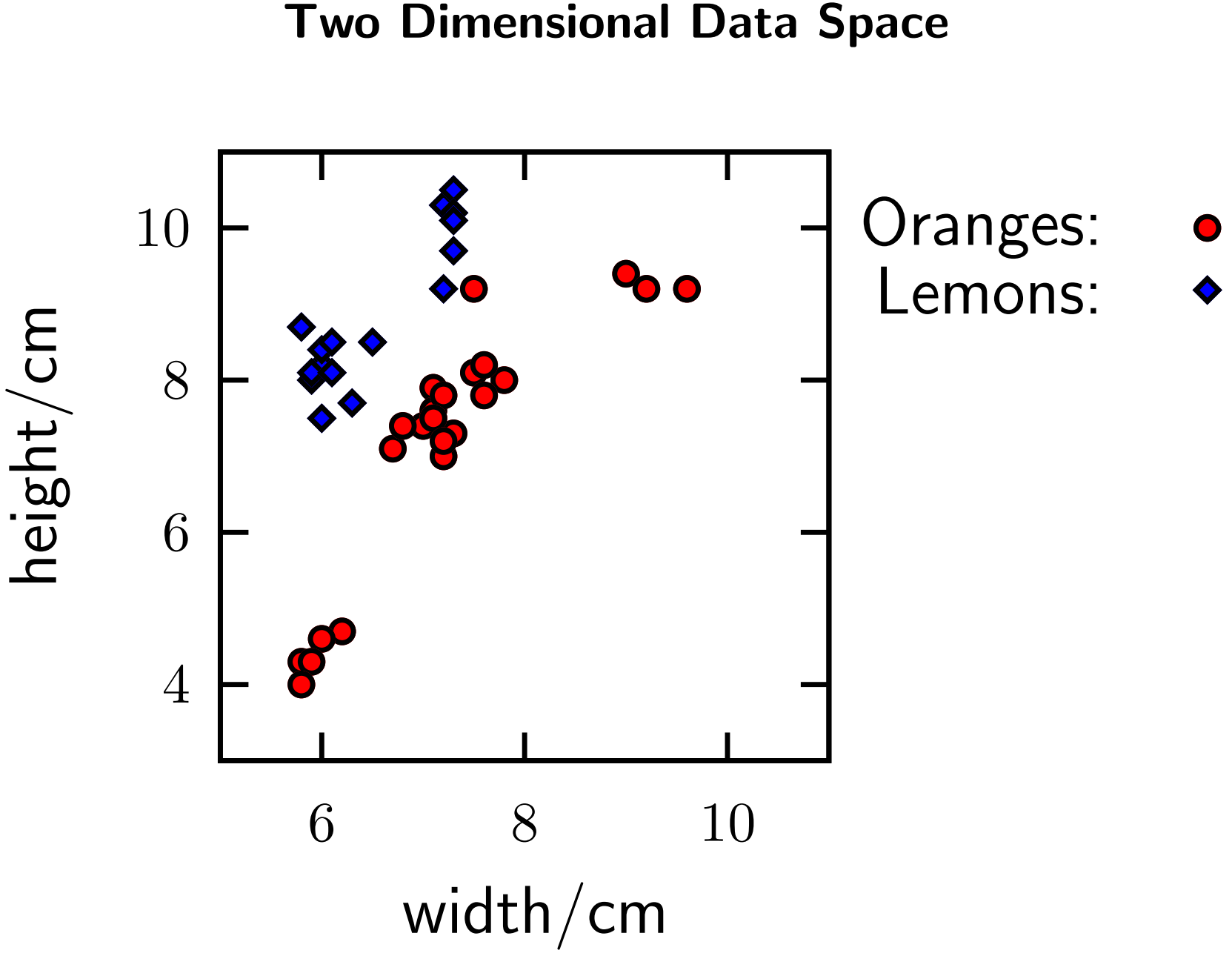
Update step: Compute the means of the clusters

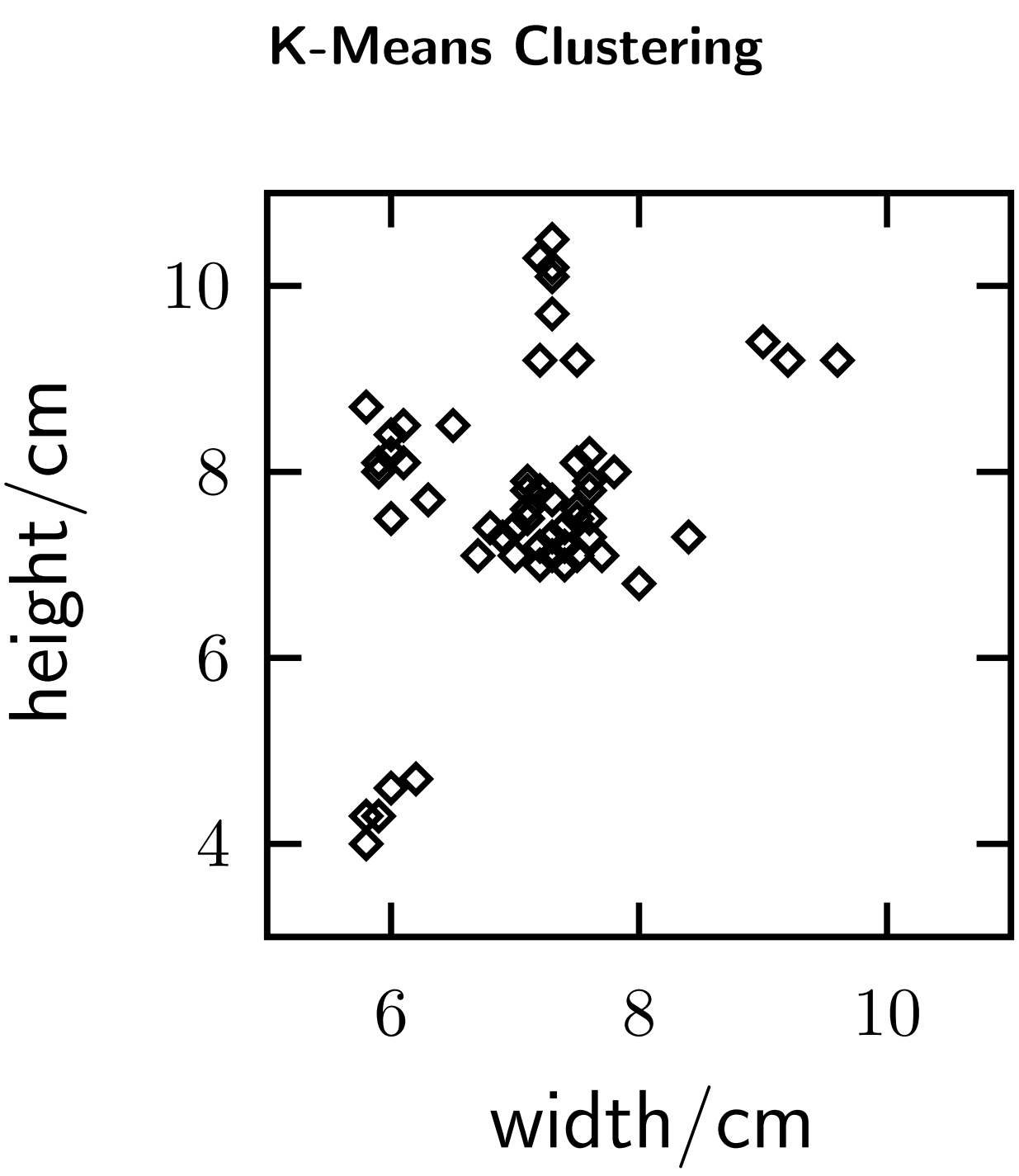
$$\mathbf{m}_k = \frac{\sum_{i=1}^{n} r_{ik} \mathbf{y}_i}{\sum_{i=1}^{n} r_{ik}}$$

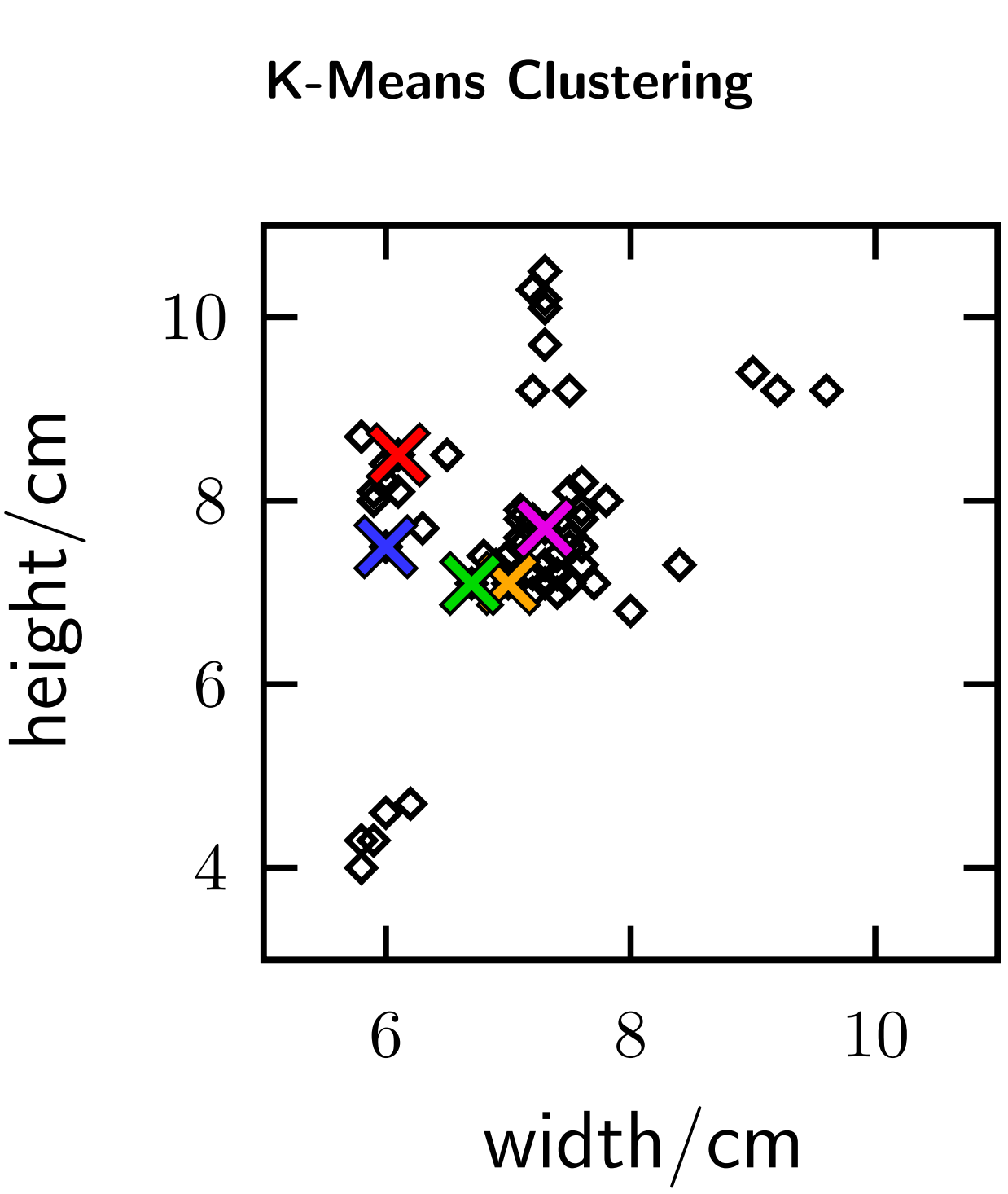
where $\sum_{i=1}^{n} r_{ik} = R(k)$ is the total number of points assigned to cluster k.

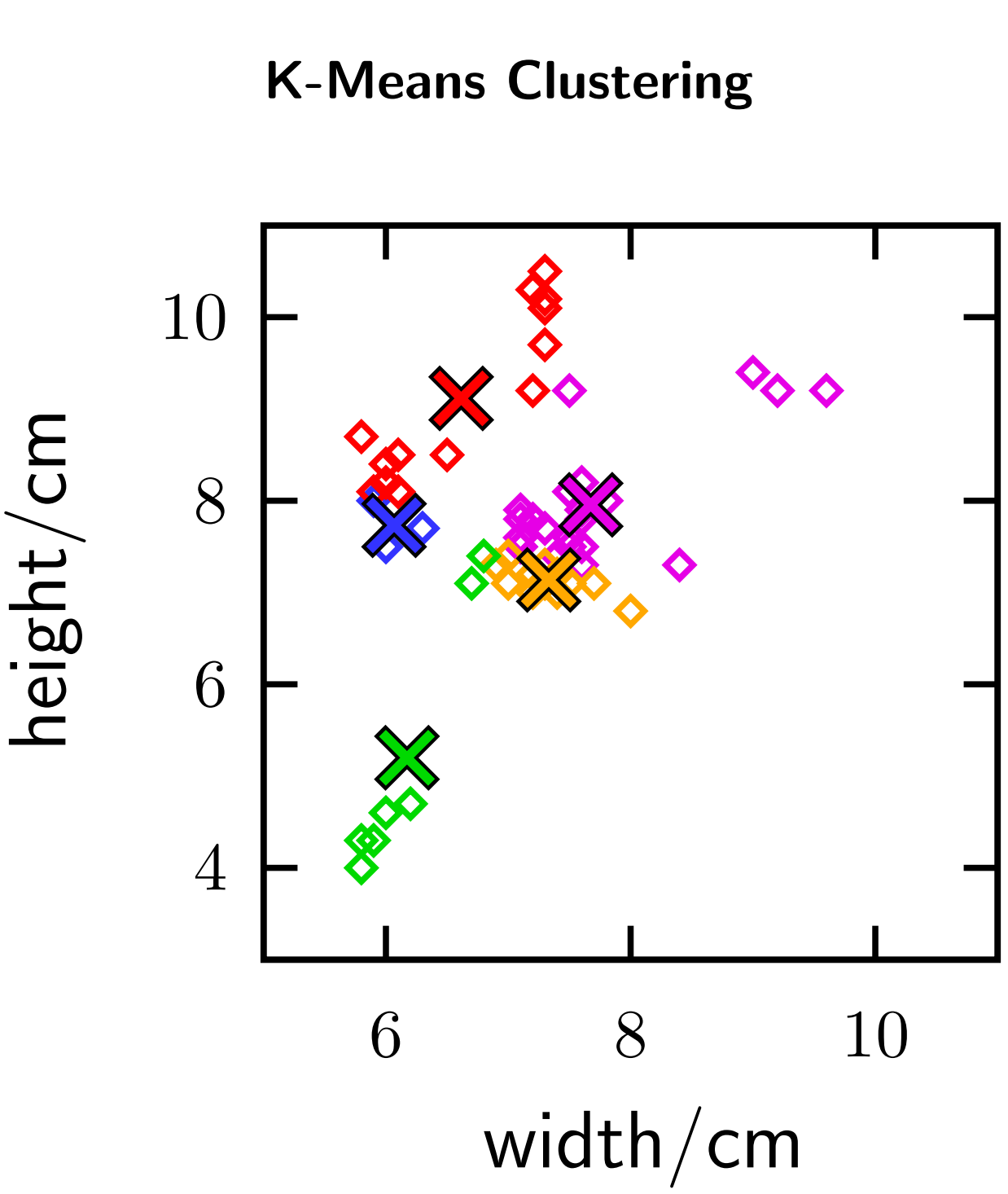
Oranges and Lemons (thanks to Iain Murray)

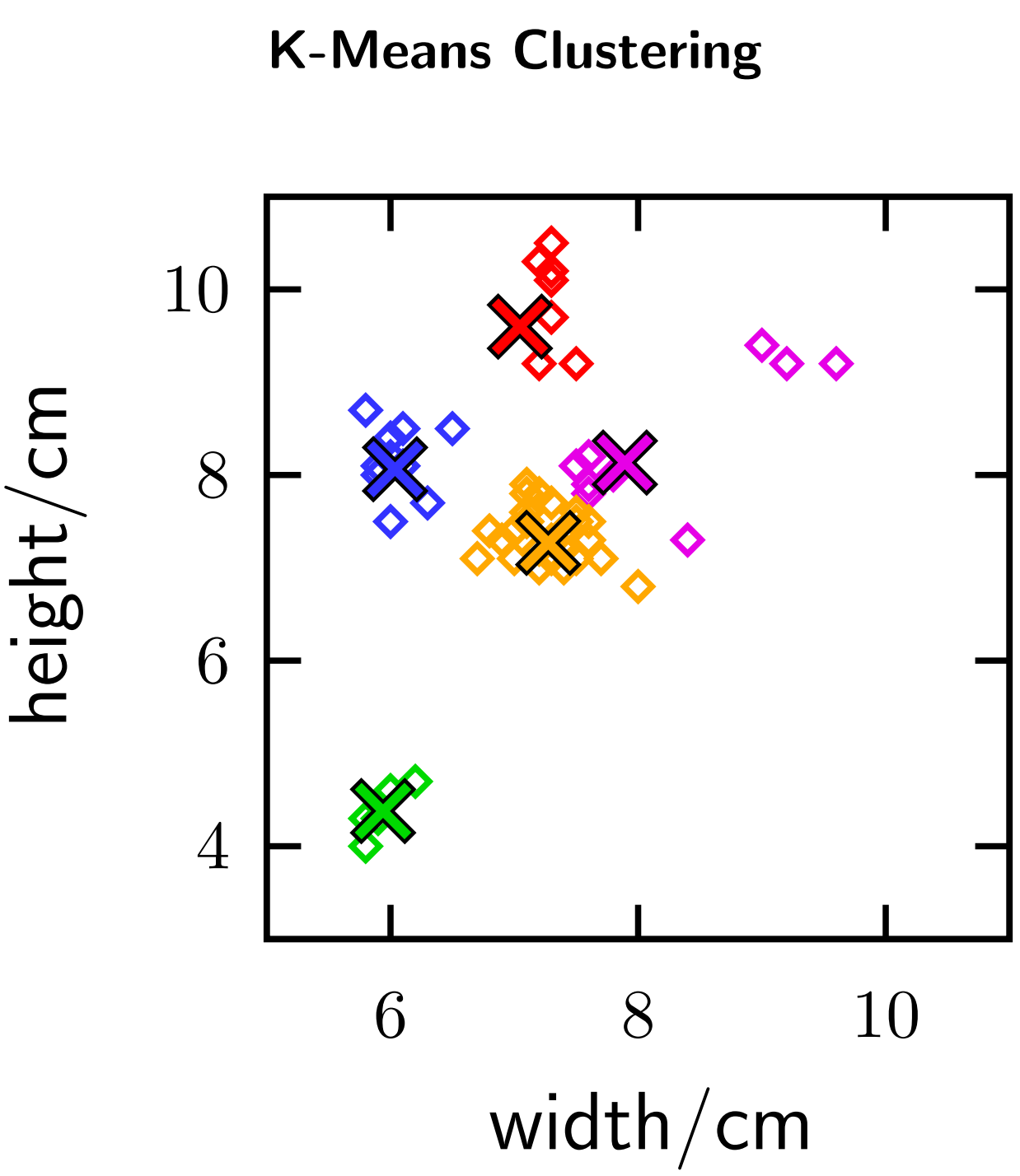


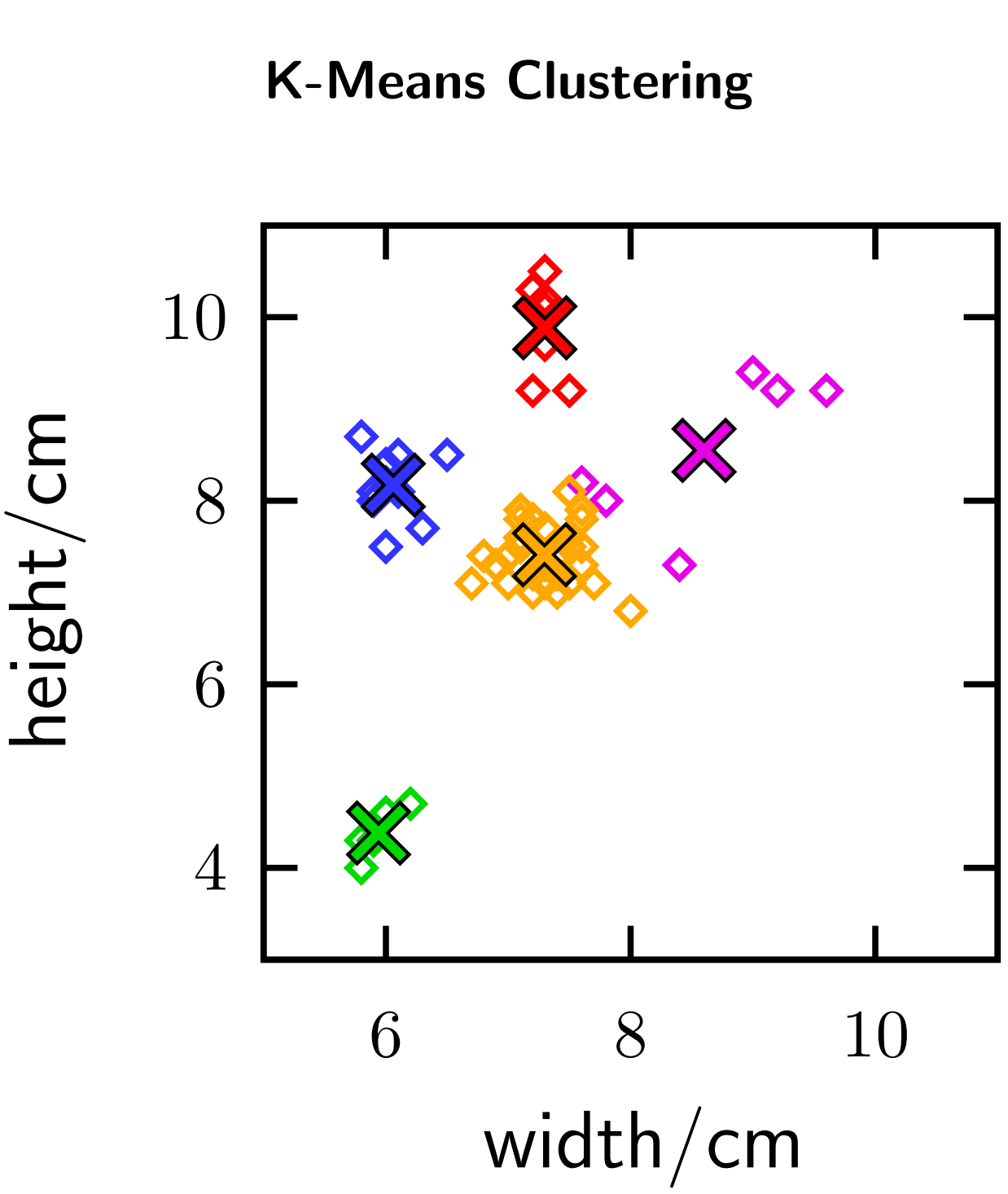

Two Dimensional Data Space
height/cm
10
8
6
4
6
8
10
width/cm
Oranges:
Lemons:

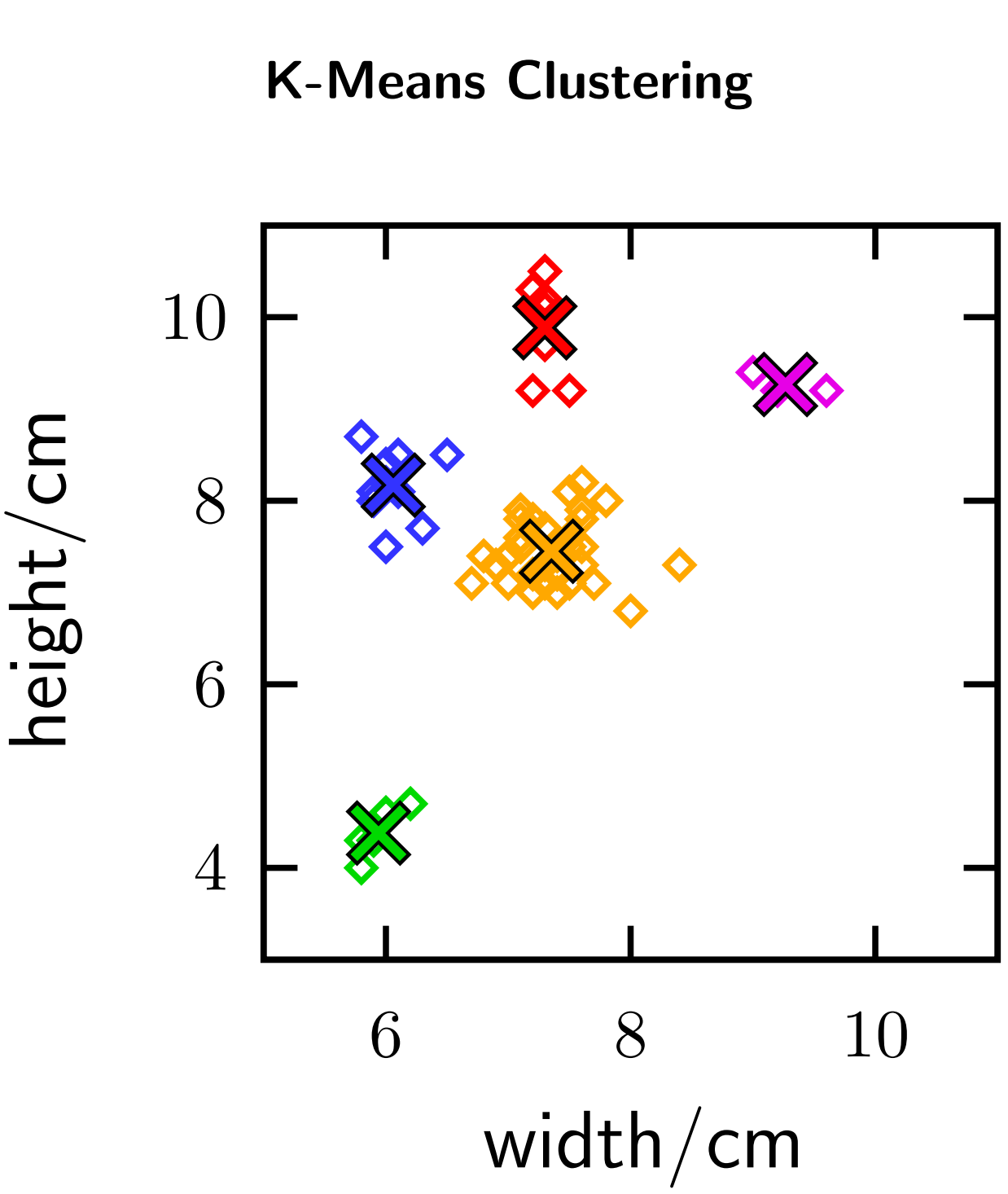

K-Means Clustering
height/cm
width/cm
10
8
6
4
6
8
10


K-Means Clustering
height/cm
width/cm
10
8
6
4
6
8
10


K-Means Clustering
height/cm
width/cm
10
8
6
4
6
8
10


K-Means Clustering
height/cm
width/cm
10
8
6
4
6
8
10


K-Means Clustering
height/cm
width/cm
10
8
6
4
6
8
10


K-Means Clustering
height/cm
10
8
6
4
6
8
10
width/cm

Limitations of K-Means Clustering

- Solution depends heavily on initialization
- Why are the cluster means equal to the empirical means of the assigned points?
- Why are distances computed the way they are?
- How do we find K?
- Responsibilities assign points to clusters in a hard way.

K-Means Clustering as a Mixture of Gaussians

Think of a cluster as of one Gaussian distribution with mean $\mathbf{m}_k$ and unit variance:

$$p(\mathbf{y}|\mathbf{m}_k) = (2\pi)^{-\frac{D}{2}} \exp\left(-\frac{1}{2}\|\mathbf{y} - \mathbf{m}_k\|^2\right)$$

We define a Gaussian mixture model with equal mixing proportions $\pi_k = 1/K$:

$$p(\mathbf{y}) = \sum_{k=1}^{K} \pi_k p(\mathbf{y}|\mathbf{m}_k)$$

If we knew the responsibilities r_{ij} of cluster j for point i, the likelihood of the model would be:

$$p(\{\mathbf{y}_i\}, \{r_{ij}\}|\{\mathbf{m}_k\}) = \prod_{i=1}^{n} [\pi_k p(\mathbf{y}_i|\mathbf{m}_k)]^{r_{ij}}$$

K-Means Clustering as a Mixture of Gaussians (2)

If we knew the responsibilities r_{ij} of cluster j for point i, the likelihood of the model would be:

$$p(\{\mathbf{y}_i\}, \{r_{ij}\}|\{\mathbf{m}_k\}) = \prod_{i=1}^{n} [\pi_k p(\mathbf{y}_i|\mathbf{m}_k)]^{r_{ij}}$$

Taking logarithms:

$$\log p(\{\mathbf{y}_i\}, \{r_{ij}\}|\{\mathbf{m}_k\}) = -\frac{1}{2} \sum_{i=1}^{n} r_{ik} \|\mathbf{y}_i - \mathbf{m}_k\|^2 + \text{Constant}$$

Maximizing this expression is equivalent to minimizing the K-Means cost function.
It is better to treat the r_{ik} as latent variables.

Mixture of Gaussians (MoG)

The discrete indicator variable $s_i = k$ means that data point i is assigned to cluster k.

The prior probability of being assigned to cluster k is $\pi_k = p(s = k)$.

Model: latent variables s_i assign points to one of L Gaussian components:

$$p(\mathbf{y}_i|\boldsymbol{\theta}) = \sum_{k=1}^{K} p(s_i = k)\, p(\mathbf{y}_i|s_i = k, \boldsymbol{\mu}_k, \Sigma_k) = \sum_{k=1}^{K} \pi_k P_{ik}$$

with $P_{ik} = p(\mathbf{y}_i|s_i = k, \boldsymbol{\mu}_k, \Sigma_k) = \mathcal{N}(\mathbf{y}_i|\boldsymbol{\mu}_k, \Sigma_k)$.

Goal: learn the parameters $\{\pi_k\}$ and $\{\boldsymbol{\mu}_k, \Sigma_k\}$.

Maximum Likelihood for MoG

Assuming independent data $\mathcal{D} = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$:

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k \, p(\mathbf{y}_i | s_i = k, \boldsymbol{\mu}_k, \Sigma_k)$$

Taking the logarithm:

$$\mathcal{L} = \sum_{i=1}^{n} \log \sum_{k=1}^{K} \pi_k \, p(\mathbf{y}_i | s_i = k, \boldsymbol{\mu}_k, \Sigma_k)$$

Maximum Likelihood for MoG (2)

Remember, log likelihood is $\mathcal{L} = \sum_{i=1}^{n} \log \sum_{k=1}^{K} \pi_k P_{ik}$.

Derivative with respect to $\theta = \{\boldsymbol{\mu}, \Sigma\}$

$$\frac{\partial \mathcal{L}}{\partial \theta_k} = \sum_{i=1}^{n} \frac{\pi_k}{\sum_{j=1}^{K} \pi_j P_{ij}} \frac{\partial P_{ik}}{\partial \theta_k} = \sum_{i=1}^{n} \frac{\pi_k P_{ik}}{\sum_{j=1}^{K} \pi_j P_{ij}} \frac{\partial \log P_{ik}}{\partial \theta_k} = \sum_{i=1}^{n} r_{ik} \frac{\partial \log P_{ik}}{\partial \theta_k}$$

We have used the identity $\partial p / \partial \theta = p \times \partial \log p / \partial \theta$, and defined the responsibilities r_{ik}:

$$r_{ik} = \frac{p(\mathbf{y}_i, s_i = k | \boldsymbol{\mu}_k, \Sigma_k)}{p(\mathbf{y}_i | \boldsymbol{\mu}_k, \Sigma_k)} = p(s_i = k | \mathbf{y}_i, \boldsymbol{\mu}_k, \Sigma_k)$$

which are posterior class-membership probabilities, that normalize $\sum_{k=1}^{K} r_{ik} = 1$.

Maximum Likelihood for MoG (3)

Derivative with respect to the π_k:
We need to add $\lambda(1 - \sum_{k=1}^{K} \pi_k)$ to the objective function: $\tilde{\mathcal{L}} = \mathcal{L} + \lambda(1 - \sum_{k=1}^{K} \pi_k)$

$$\frac{\partial \tilde{\mathcal{L}}}{\partial \pi_k} = \sum_{i=1}^{n} \frac{P_{ik}}{\sum_{j=1}^{n} \pi_j P_{ij}} - \lambda = 0 \iff \sum_{i=1}^{n} r_{ik} - \lambda \pi_k = 0$$

$$\frac{\partial \tilde{\mathcal{L}}}{\partial \lambda} = 1 - \sum_{i=k}^{K} \pi_K = 0$$

Using the fact that $\sum_{k=1} \sum_{i=1}^{n} r_{ik} - \lambda \pi_k = n - \lambda = 0$ we get $\lambda = n$ and

$$\pi_k = \frac{1}{n} \sum_{i=1}^{n} r_{ik}$$

(what is $\sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik}$?)

Maximum Likelihood for MoG (4)

Derivative with respect to the mean:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_k} = \sum_{i=1}^{n} r_{ik} \Sigma_k^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_k)$$

Derivative with respect to the inverse covariance:

$$\frac{\partial \mathcal{L}}{\partial \Sigma_k^{-1}} = \sum_{i=1}^{n} r_{ik} [\Sigma_k - (\mathbf{y}_i - \boldsymbol{\mu}_k)(\mathbf{y}_i - \boldsymbol{\mu}_k)^\top]$$

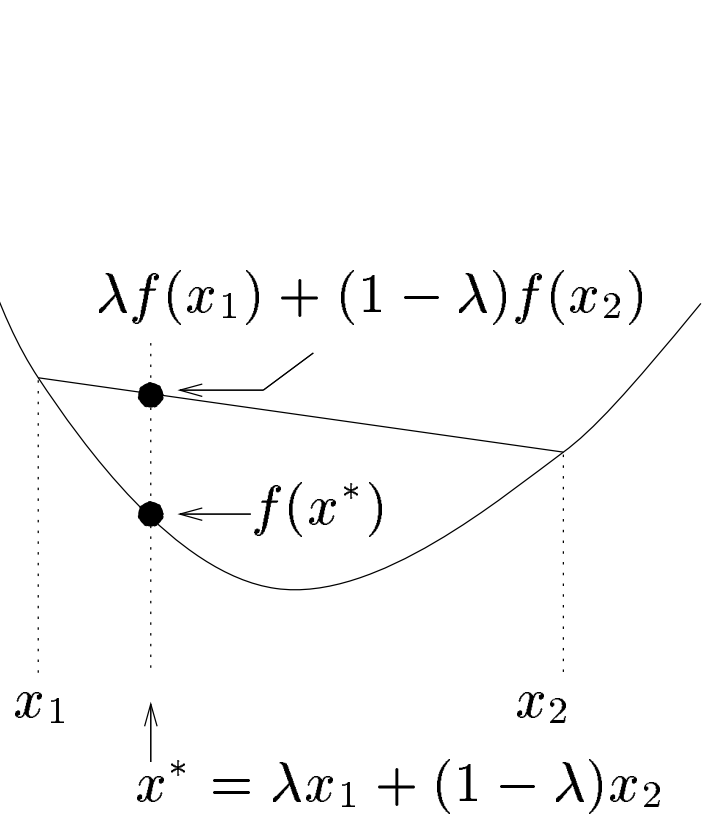
Setting to zero we get the **update equations**:

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{n} r_{ik} \mathbf{y}_i}{\sum_{i=1}^{n} r_{ik}} \quad \text{and} \quad \Sigma_k = \frac{\sum_{i=1}^{n} r_{ik} (\mathbf{y}_i - \boldsymbol{\mu}_k)(\mathbf{y}_i - \boldsymbol{\mu}_k)^\top}{\sum_{i=1}^{n} r_{ik}}$$

Notice that the denominators $\sum_{i=1}^{n} r_{ik} = n\pi_k$ are the **effective** total number of points assigned to cluster k.

Jensen's Inequality

If $f(x)$ is convex then $\lambda f(x_1) + (1-\lambda) f(x_2) \geq f\Big[\lambda x_1 + (1-\lambda) x_2\Big]$ with $\lambda \in [0, 1]$



If $f(x)$ is concave then $\lambda f(x_1) + (1-\lambda) f(x_2) \leq f\Big[\lambda x_1 + (1-\lambda) x_2\Big]$

The Expectation Maximization (EM) algorithm

Assume a model with observed (visible) variables $\mathbf{y}$, unobserved (hidden / latent / missing) variables $\mathbf{x}$, and model parameters θ

Goal: Maximize the log likelihood (i.e. ML learning) wrt θ:

$$\mathcal{L}(\theta) = \log p(\mathbf{y}|\theta) = \log \int p(\mathbf{x}, \mathbf{y}|\theta) d\mathbf{x},$$

Any distribution, $q(\mathbf{x})$, over the hidden variables can be used to obtain a lower bound on the log likelihood:

$$\mathcal{L}(\theta) = \log \int q(\mathbf{x}) \frac{p(\mathbf{x}, \mathbf{y}|\theta)}{q(\mathbf{x})} d\mathbf{x} \geq \int q(\mathbf{x}) \log \frac{p(\mathbf{x}, \mathbf{y}|\theta)}{q(\mathbf{x})} \, d\mathbf{x} \stackrel{\text{def}}{=} \mathcal{F}(q, \theta),$$

This lower bound is called Jensen's inequality and comes from the fact that the log function is concave ("log of average is greater than average of logs").

In the EM algorithm, we alternately optimize $\mathcal{F}(q, \theta)$ wrt $q(\mathbf{x})$ and θ, and we can prove that this will never decrease $\mathcal{L}(\theta)$.

The E and M steps of EM

The lower bound on the log likelihood:

$$\mathcal{F}(q,\theta) = \int q(\mathbf{x}) \log \frac{p(\mathbf{x},\mathbf{y}|\theta)}{q(\mathbf{x})} d\mathbf{x} = \int q(\mathbf{x}) \log p(\mathbf{x},\mathbf{y}|\theta) d\mathbf{x} + \mathcal{H}(q),$$

where $\mathcal{H}(q) = -\int q(\mathbf{x}) \log q(\mathbf{x}) d\mathbf{x}$ is the entropy of $q(\mathbf{x})$. EM alternates between:

E step: optimize $\mathcal{F}(q,\theta)$ wrt distribution over hidden variables holding parameters fixed:

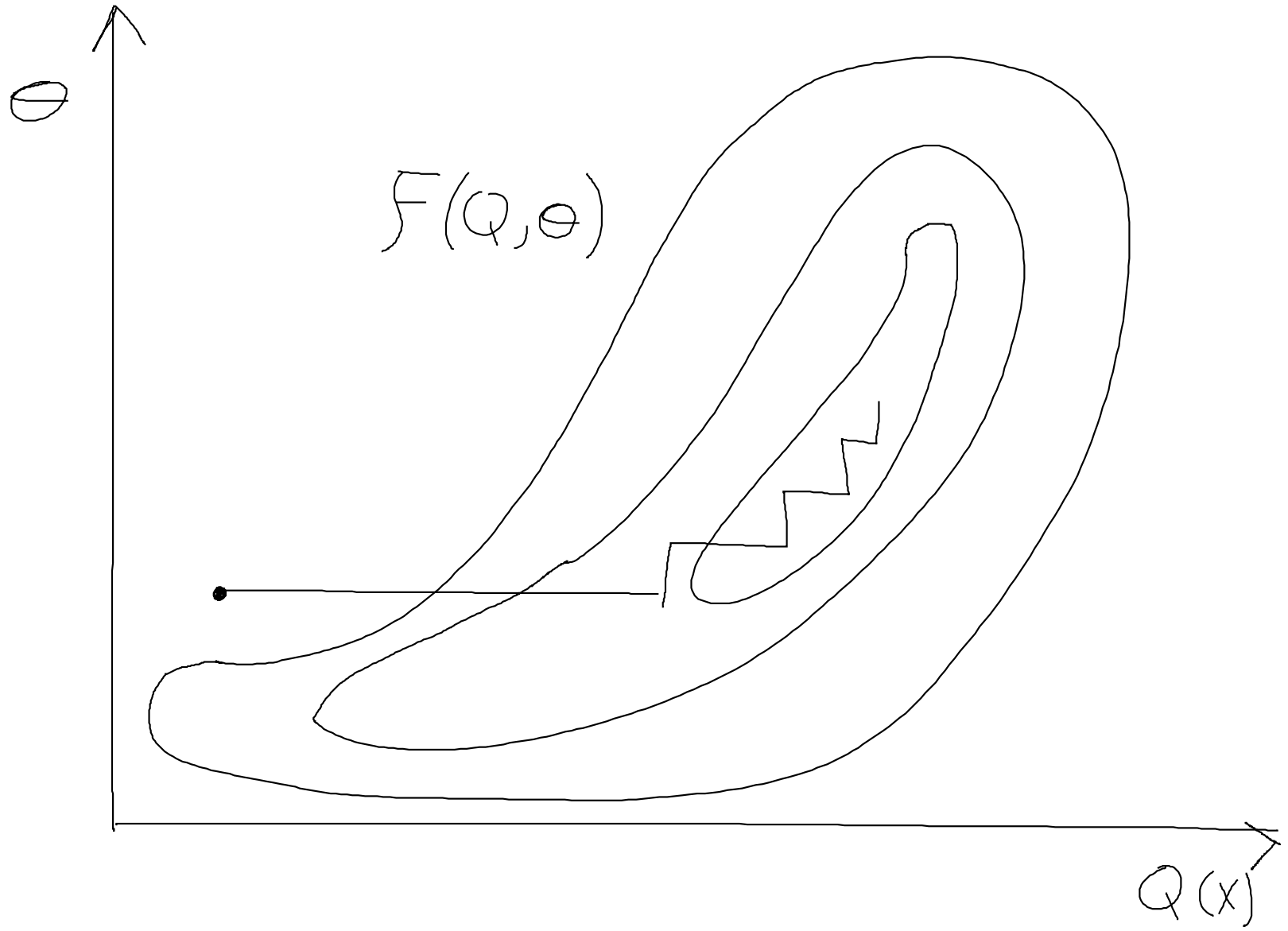
$$q^{(k)}(\mathbf{x}) := \underset{q(\mathbf{x})}{\operatorname{argmax}} \; \mathcal{F}\big(q(\mathbf{x}), \theta^{(k-1)}\big).$$

M step: maximize $\mathcal{F}(q,\theta)$ wrt parameters holding hidden distribution fixed:

$$\theta^{(k)} := \underset{\theta}{\operatorname{argmax}} \; \mathcal{F}\big(q^{(k)}(\mathbf{x}), \theta\big) = \underset{\theta}{\operatorname{argmax}} \; \int q^{(k)}(\mathbf{x}) \log p(\mathbf{x},\mathbf{y}|\theta) d\mathbf{x}.$$

The second equality comes from the fact that the entropy of $q(\mathbf{x})$ does not depend directly on θ.

EM as Coordinate Ascent in $\mathcal{F}$



The Intuition Behind EM

E step: fill in values for the hidden variables according to their posterior probabilities

M step: learn model as if hidden variables were not hidden

- EM is useful because in many models, if the hidden variables were no longer hidden, learning would be easy (e.g. consider a mixture of Gaussians).

- EM breaks up a hard learning problem into a sequence of easy learning problems.

The EM algorithm never decreases the log likelihood

The difference between the log likelihood and the lower bound:

$$\begin{aligned}\mathcal{L}(\theta) - \mathcal{F}(q,\theta) &= \log p(\mathbf{y}|\theta) - \int q(\mathbf{x}) \log \frac{p(\mathbf{x},\mathbf{y}|\theta)}{q(\mathbf{x})} d\mathbf{x} \\ &= \log p(\mathbf{y}|\theta) - \int q(\mathbf{x}) \log \frac{p(\mathbf{x}|\mathbf{y},\theta)p(\mathbf{y}|\theta)}{q(\mathbf{x})} d\mathbf{x} \\ &= -\int q(\mathbf{x}) \log \frac{p(\mathbf{x}|\mathbf{y},\theta)}{q(\mathbf{x})} d\mathbf{x} = \mathcal{KL}\big(q(\mathbf{x}), p(\mathbf{x}|\mathbf{y},\theta)\big),\end{aligned}$$

This is the Kullback-Liebler divergence; it is zero if and only if $q(\mathbf{x}) = p(\mathbf{x}|\mathbf{y},\theta)$.

Therefore, the E step simply sets $q(\mathbf{x}) \leftarrow p(\mathbf{x}|\mathbf{y},\theta)$.

The E and M steps together increase the log likelihood:

$$\mathcal{L}\big(\theta^{(k-1)}\big) \underset{\text{E step}}{=} \mathcal{F}\big(q^{(k)}, \theta^{(k-1)}\big) \underset{\text{M step}}{\leq} \mathcal{F}\big(q^{(k)}, \theta^{(k)}\big) \underset{\text{Jensen}}{\leq} \mathcal{L}\big(\theta^{(k)}\big),$$

where the first equality holds because of the E step, and the first inequality comes from the M step and the final inequality from Jensen.

EM converges to a local optimum of $\mathcal{L}(\theta)$.

The $\mathcal{KL}\big(q(x), p(x)\big)$ is non-negative and zero iff $\forall x:\ p(x) = q(x)$

First let's consider discrete distributions; the Kullback-Liebler divergence is:

$$\mathcal{KL}(q, p) = \sum_i q_i \log \frac{q_i}{p_i}.$$

To find the distribution q which minimizes $\mathcal{KL}(q, p)$ we add a Lagrange multiplier to enforce the normalization constraint:

$$E \stackrel{\text{def}}{=} \mathcal{KL}(q, p) + \lambda\big(1 - \sum_i q_i\big) = \sum_i q_i \log \frac{q_i}{p_i} + \lambda\big(1 - \sum_i q_i\big)$$

We then take partial derivatives and set to zero:

$$\left.\begin{array}{rcl} \dfrac{\partial E}{\partial q_i} &=& \log q_i - \log p_i + 1 - \lambda = 0 \Rightarrow q_i = p_i \exp(\lambda - 1) \\ \dfrac{\partial E}{\partial \lambda} &=& 1 - \sum_i q_i = 0 \Rightarrow \sum_i q_i = 1 \end{array}\right\} \Rightarrow q_i = p_i.$$

Why $\mathcal{KL}(q,p)$ is non-negative and zero iff $p(x) = q(x)$ . . .

Check that the curvature (Hessian) is positive (definite), corresponding to a minimum:

$$\frac{\partial^2 E}{\partial q_i \partial q_i} = \frac{1}{q_i} > 0, \qquad \frac{\partial^2 E}{\partial q_i \partial q_j} = 0,$$

showing that $q_i = p_i$ is a genuine minimum.

At the minimum is it easily verified that $\mathcal{KL}(p,p) = 0$.

A similar proof holds for $\mathcal{KL}$ between continuous densities, the derivatives being substituted by functional derivatives.

The Gaussian mixture model (E-step)

In a univariate Gaussian mixture model, the density of a data point y is:

$$p(y|\theta) = \sum_{k=1}^{K} p(s=k|\theta)p(y|s=k,\theta) \propto \sum_{k=1}^{K} \frac{\pi_k}{\sigma_k} \exp\Big\{ -\frac{1}{2\sigma_k^2}(y-\mu_k)^2 \Big\},$$

where θ is the collection of parameters: means μ_k, variances σ_k^2 and mixing proportions $\pi_k = p(s=k|\theta)$.

The hidden variable s_i indicates which component observation y_i belongs to.
The E-step computes the posterior for s_i given the current parameters:

$$\begin{aligned} q(s_i) &= p(s_i|y_i,\theta) \propto p(y_i|s_i,\theta)p(s_i|\theta) \\ r_{ik} \stackrel{\text{def}}{=} q(s_i=k) &\propto \frac{\pi_k}{\sigma_k} \exp\Big\{ -\frac{1}{2\sigma_k^2}(y_i-\mu_k)^2 \Big\} \quad \text{(responsibilities)} \end{aligned}$$

with the normalization such that $\sum_k r_k^{(c)} = 1$.

The Gaussian mixture model (M-step)

In the M-step we optimize the sum (since s is discrete):

$$E = \sum q(s) \log[p(s|\theta)\ p(y|s,\theta)] = \sum_{i,k} r_{ik}\big[\log \pi_k - \log \sigma_k - \frac{1}{2\sigma_k^2}(y_i - \mu_k)^2\big].$$

Optimization is done by setting the partial derivatives of E to zero:

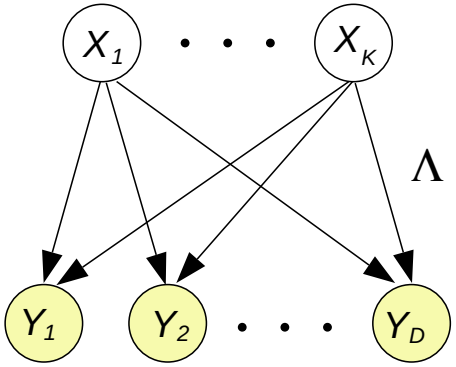
$$\frac{\partial E}{\partial \mu_k} = \sum_i r_{ik} \frac{(y_i - \mu_k)}{2\sigma_k^2} = 0 \Rightarrow \quad \mu_k = \frac{\sum_i r_{ik} y_i}{\sum_i r_{ik}},$$

$$\frac{\partial E}{\partial \sigma_k} = \sum_i r_{ik}\Big[-\frac{1}{\sigma_k} + \frac{(y_i - \mu_k)^2}{\sigma_k^3}\Big] = 0 \Rightarrow \quad \sigma_k^2 = \frac{\sum_i r_{ik}(y_i - \mu_k)^2}{\sum_i r_{ik}},$$

$$\frac{\partial E}{\partial \pi_k} = \sum_i r_{ik} \frac{1}{\pi_k}, \qquad \frac{\partial E}{\partial \pi_k} + \lambda = 0 \Rightarrow \quad \pi_k = \frac{1}{n}\sum_i r_{ik},$$

where λ is a Lagrange multiplier ensuring that the mixing proportions sum to unity.

EM for Factor Analysis



The model for $\mathbf{y}$:
$p(\mathbf{y}|\theta) = \int p(\mathbf{x}|\theta)p(\mathbf{y}|\mathbf{x},\theta)d\mathbf{x} = \mathcal{N}(0, \Lambda\Lambda^\top + \Psi)$
Model parameters: $\theta = \{\Lambda, \Psi\}$.

E step: For each data point $\mathbf{y}_n$, compute the posterior distribution of hidden factors given the observed data: $q_n(\mathbf{x}) = p(\mathbf{x}|\mathbf{y}_n, \theta_t)$.

M step: Find the θ_{t+1} that maximises $\mathcal{F}(q, \theta)$:

$$\begin{aligned}\mathcal{F}(q,\theta) &= \sum_n \int q_n(\mathbf{x}) \left[\log p(\mathbf{x}|\theta) + \log p(\mathbf{y}_n|\mathbf{x},\theta) - \log q_n(\mathbf{x})\right] d\mathbf{x} \\ &= \sum_n \int q_n(\mathbf{x}) \left[\log p(\mathbf{x}|\theta) + \log p(\mathbf{y}_n|\mathbf{x},\theta)\right] d\mathbf{x} + \text{c}.\end{aligned}$$

The E step for Factor Analysis

E step: For each data point $\mathbf{y}_n$, compute the posterior distribution of hidden factors given the observed data: $q_n(\mathbf{x}) = p(\mathbf{x}|\mathbf{y}_n, \theta) = p(\mathbf{x}, \mathbf{y}_n|\theta)/p(\mathbf{y}_n|\theta)$

Tactic: write $p(\mathbf{x}, \mathbf{y}_n|\theta)$, consider $\mathbf{y}_n$ to be fixed. What is this as a function of $\mathbf{x}$?

$$
\begin{aligned}
p(\mathbf{x}, \mathbf{y}_n) &= p(\mathbf{x})p(\mathbf{y}_n|\mathbf{x}) \\
&= (2\pi)^{-\frac{K}{2}} \exp\{-\frac{1}{2}\mathbf{x}^\top\mathbf{x}\} \, |2\pi\Psi|^{-\frac{1}{2}} \exp\{-\frac{1}{2}(\mathbf{y}_n - \Lambda\mathbf{x})^\top\Psi^{-1}(\mathbf{y}_n - \Lambda\mathbf{x})\} \\
&= \text{c} \times \exp\{-\frac{1}{2}[\mathbf{x}^\top\mathbf{x} + (\mathbf{y}_n - \Lambda\mathbf{x})^\top\Psi^{-1}(\mathbf{y}_n - \Lambda\mathbf{x})]\} \\
&= \text{c'} \times \exp\{-\frac{1}{2}[\mathbf{x}^\top(I + \Lambda^\top\Psi^{-1}\Lambda)\mathbf{x} - 2\mathbf{x}^\top\Lambda^\top\Psi^{-1}\mathbf{y}_n]\} \\
&= \text{c''} \times \exp\{-\frac{1}{2}[\mathbf{x}^\top\Sigma^{-1}\mathbf{x} - 2\mathbf{x}^\top\Sigma^{-1}\mu + \mu^\top\Sigma^{-1}\mu]\}
\end{aligned}
$$

So $\Sigma = (I + \Lambda^\top\Psi^{-1}\Lambda)^{-1} = I - \beta\Lambda$ and $\mu = \Sigma\Lambda^\top\Psi^{-1}\mathbf{y}_n = \beta\mathbf{y}_n$. Where $\beta = \Sigma\Lambda^\top\Psi^{-1}$. Note that μ is a linear function of $\mathbf{y}_n$ and Σ does not depend on $\mathbf{y}_n$.

The M step for Factor Analysis

M step: Find θ_{t+1} maximising $\mathcal{F} = \sum_n \int q_n(\mathbf{x}) \left[\log p(\mathbf{x}|\theta) + \log p(\mathbf{y}_n|\mathbf{x},\theta)\right] d\mathbf{x} + \mathsf{c}$

$$\begin{aligned}
\log p(\mathbf{x}|\theta) + \log p(\mathbf{y}_n|\mathbf{x},\theta) &= \mathsf{c} - \frac{1}{2}\mathbf{x}^\top\mathbf{x} - \frac{1}{2}\log|\Psi| - \frac{1}{2}(\mathbf{y}_n - \Lambda\mathbf{x})^\top\Psi^{-1}(\mathbf{y}_n - \Lambda\mathbf{x}) \\
&= \mathsf{c}' - \frac{1}{2}\log|\Psi| - \frac{1}{2}[\mathbf{y}_n{}^\top\Psi^{-1}\mathbf{y}_n - 2\mathbf{y}_n{}^\top\Psi^{-1}\Lambda\mathbf{x} + \mathbf{x}^\top\Lambda^\top\Psi^{-1}\Lambda\mathbf{x}] \\
&= \mathsf{c}' - \frac{1}{2}\log|\Psi| - \frac{1}{2}[\mathbf{y}_n{}^\top\Psi^{-1}\mathbf{y}_n - 2\mathbf{y}_n{}^\top\Psi^{-1}\Lambda\mathbf{x} + \mathsf{tr}(\Lambda^\top\Psi^{-1}\Lambda\mathbf{x}\mathbf{x}^\top)]
\end{aligned}$$

Taking expectations over $q_n(\mathbf{x})$. . .

$$= \mathsf{c}' - \frac{1}{2}\log|\Psi| - \frac{1}{2}[\mathbf{y}_n{}^\top\Psi^{-1}\mathbf{y}_n - 2\mathbf{y}_n{}^\top\Psi^{-1}\Lambda\mu_n + \mathsf{tr}(\Lambda^\top\Psi^{-1}\Lambda(\mu_n\mu_n{}^\top + \Sigma))]$$

Note that we don't need to know everything about q, just the expectations of $\mathbf{x}$ and $\mathbf{x}\mathbf{x}^\top$ under q (i.e. the expected sufficient statistics).

The M step for Factor Analysis (cont.)

$$\mathcal{F} = \text{c'} - \frac{N}{2}\log|\Psi| - \frac{1}{2}\sum_n \left[\mathbf{y}_n{}^\top \Psi^{-1}\mathbf{y}_n - 2\mathbf{y}_n{}^\top \Psi^{-1}\Lambda\mu_n + \text{tr}(\Lambda^\top\Psi^{-1}\Lambda(\mu_n\mu_n{}^\top + \Sigma))\right]$$

Taking derivatives w.r.t. Λ and Ψ^{-1}, using $\frac{\partial \text{tr}(AB)}{\partial B} = A^\top$ and $\frac{\partial \log|A|}{\partial A} = A^{-\top}$:

$$\frac{\partial \mathcal{F}}{\partial \Lambda} = \Psi^{-1}\sum_n \mathbf{y}_n\mu_n{}^\top - \Psi^{-1}\Lambda\left(N\Sigma + \sum_n \mu_n\mu_n{}^\top\right) = 0$$

$$\hat{\Lambda} = \left(\sum_n \mathbf{y}_n\mu_n{}^\top\right)\left(N\Sigma + \sum_n \mu_n\mu_n{}^\top\right)^{-1}$$

$$\frac{\partial \mathcal{F}}{\partial \Psi^{-1}} = \frac{N}{2}\Psi - \frac{1}{2}\sum_n \left[\mathbf{y}_n\mathbf{y}_n{}^\top - \Lambda\mu_n\mathbf{y}_n{}^\top - \mathbf{y}_n\mu_n{}^\top\Lambda^\top + \Lambda(\mu_n\mu_n{}^\top + \Sigma)\Lambda^\top\right]$$

$$\hat{\Psi} = \frac{1}{N}\sum_n \left[\mathbf{y}_n\mathbf{y}_n{}^\top - \Lambda\mu_n\mathbf{y}_n{}^\top - \mathbf{y}_n\mu_n{}^\top\Lambda^\top + \Lambda(\mu_n\mu_n{}^\top + \Sigma)\Lambda^\top\right]$$

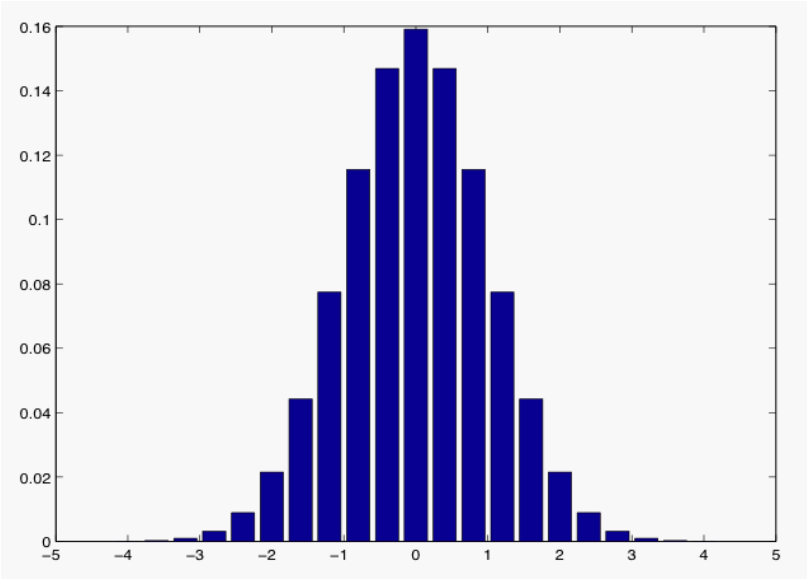
$$\hat{\Psi} = \Lambda\Sigma\Lambda^\top + \frac{1}{N}\sum_n (\mathbf{y}_n - \Lambda\mu_n)(\mathbf{y}_n - \Lambda\mu_n)^\top \qquad \text{(squared residuals)}$$

Note: we should actually only take derivarives w.r.t. Ψ_{dd} since Ψ is diagonal.
When $\Sigma \to 0$ these become the equations for linear regression!

Appendix: Coding Interpretation of Factor Analysis: Coding Under a Gaussian

Remember, from Shannon's source coding theorem:

$$\begin{aligned} l(x) &= -\log P(x) \approx -\log[p(x)\Delta] = -\log p(x) - \log \Delta \\ &= \frac{(x-\mu)^2}{2\sigma^2} + \frac{1}{2}\log 2\pi + \log \sigma - \log \Delta \end{aligned}$$



Note as $\Delta \Rightarrow 0$ then $l(x) \Rightarrow \infty$.

Coding Interpretation of Factor Analysis

Multivariate: $l(y) = \frac{1}{2}\sum_d \frac{(y_d - \mu_d)^2}{\sigma_d^2} + \frac{D}{2}\log 2\pi + \sum_d \log \sigma_d - D \log \Delta$

Alternative, two-stage code...

First code the K factors: $l(x) = \frac{1}{2}\sum_k x_k^2 + \frac{K}{2}\log 2\pi - K \log \Delta$

Then code the data given the factors:

$$l(y|x) = \frac{1}{2}\sum_d \frac{(y_d - \sum_k \Lambda_{dk} x_k)^2}{\Psi_d^2} + \frac{D}{2}\log 2\pi + \sum_d \log \Psi_d - D \log \Delta$$

How should we choose the x ?

Deterministic vs stochastic codes and "bits back"

Coding Interpretation of PCA

First code the K factors:

$$l(x) = \frac{1}{2}\sum_k x_k^2 + \frac{K}{2}\log 2\pi - K\log\Delta$$

Then code the data given the factors:

$$l(y|x) = \frac{1}{2}\sum_d \frac{(y_d - \sum_k \Lambda_{dk}x_k)^2}{\sigma^2} + \frac{D}{2}\log 2\pi + D\log\sigma - D\log\Delta$$

Since $\sigma \to 0$ the cost of coding the factors is negligible
compared to the cost of coding the data.

Additional Suggested Readings

- David MacKay's Textbook Chapters 20 and 22 http://www.inference.phy.cam.ac.uk/mackay/itprnn/
- Hinton and Zemel (1994) Autoencoders, minimum description length, and the Helmholtz free energy. In *Advances in Neural Information Processing Systems 6*. Morgan Kaufmann. See http://www.cs.toronto.edu/∼hinton/absps/cvq.html
- Minka, T. Tutorial on linear algebra. http://www.stat.cmu.edu/∼minka/papers/matrix/
- Roweis and Ghahramani (1999) A Unifying Review of Linear Gaussian Models. Neural Computation 11(2). Sections 1-5.3 and 6-6.1. See also Appendix A.1-A.2. http://www.gatsby.ucl.ac.uk/∼zoubin/papers/lds.ps.gz
- Wallace, C. S. and Dowe, D. L. (1999) Minimum message length and Kolmogorov complexity. *The Computer Journal* 42(4):270–283.
- Welling, M. (2000) Linear models. class notes. http://www.gatsby.ucl.ac.uk/∼zoubin/course01/PCA.ps