UPPSALA UNIVERSITET

Machine learning, big data and artificial intelligence – Block 7

Måns Magnusson
Department of Statistics, Uppsala University

HT 2020

UPPSALA UNIVERSITET

This week's lectures

- Introduction to unsupervised learning
- k-means
- Mixture of Gaussians
- Expectation-Maximization
- Probabilistic PCA

UPPSALA UNIVERSITET

Supervised and Unsupervised learning

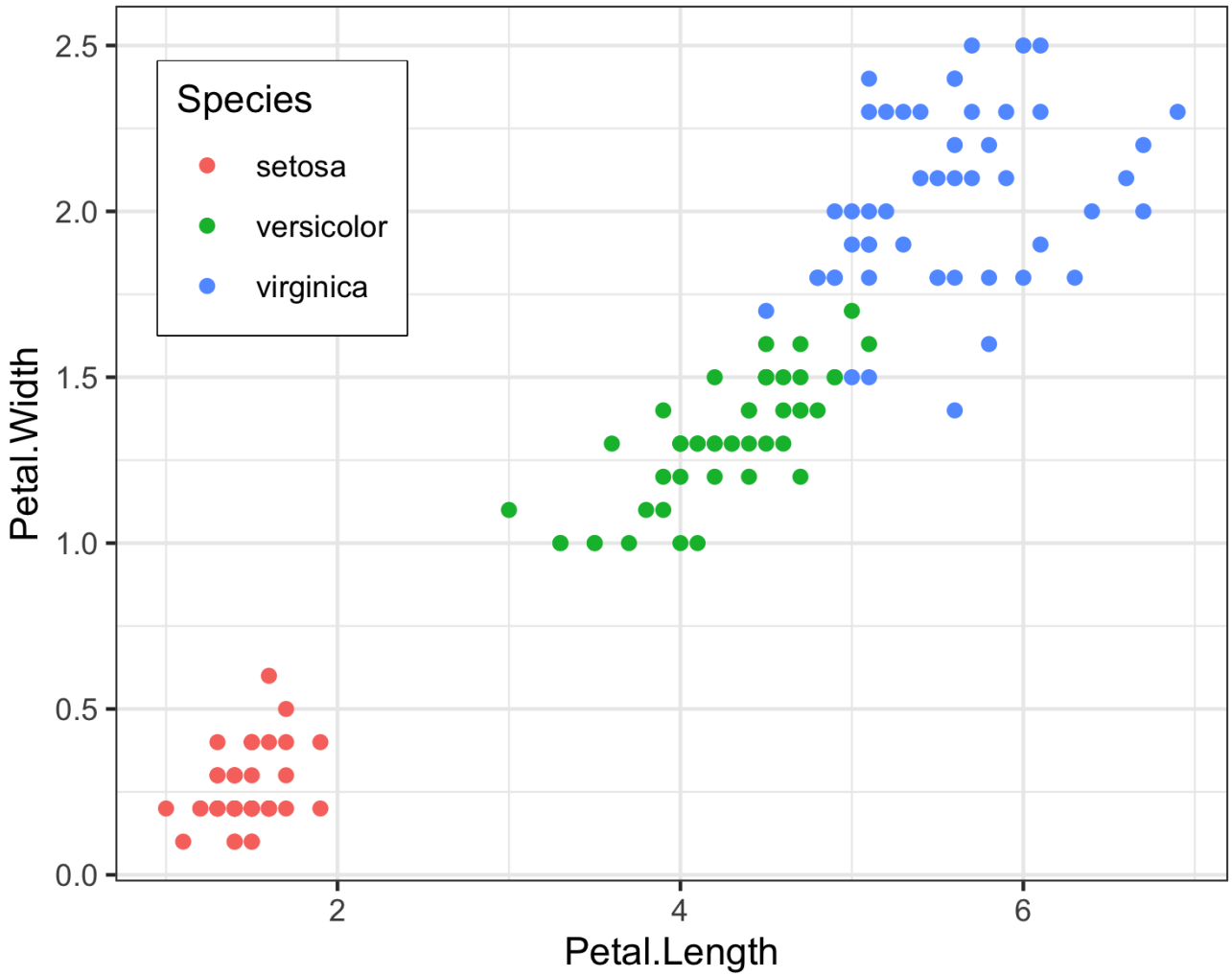


Figure: The Supervised Problem

UPPSALA UNIVERSITET

Supervised and Unsupervised learning

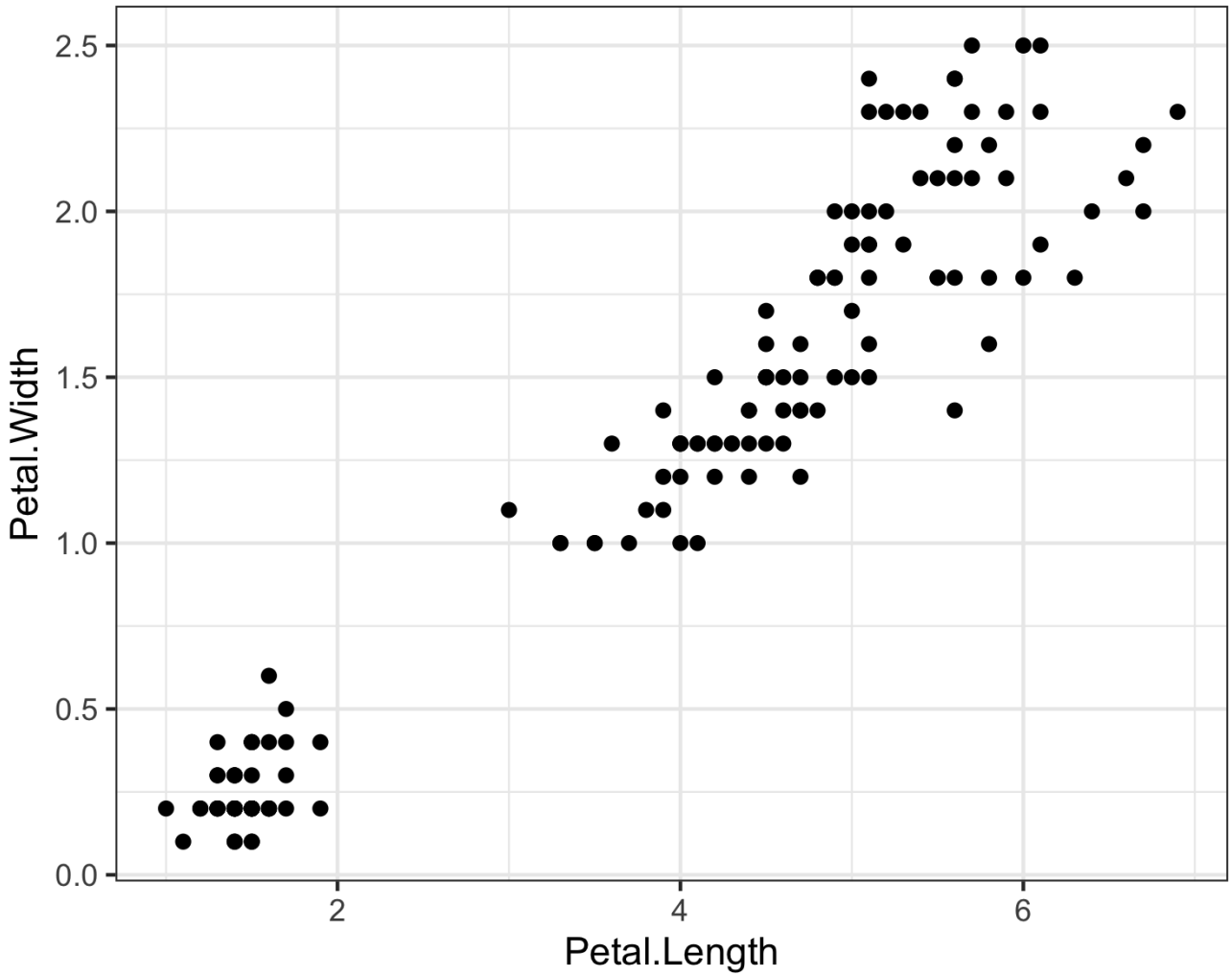


Figure: The Unsupervised Problem

UPPSALA UNIVERSITET

Supervised and Unsupervised learning

In supervised learning:

- We have *training* data

$$\mathrm{d} = \{(y_i, \mathrm{x}_i), i = 1, ..., n\}\,.$$

- We train a model $p(y|x)$ to predict y
- We only care about the loss function during training

UPPSALA UNIVERSITET

Supervised and Unsupervised learning

In supervised learning:

- We have *training* data

$$\mathrm{d} = \{(y_i, \mathrm{x}_i), i = 1, ..., n\}\,.$$

- We train a model $p(y|x)$ to predict y
- We only care about the loss function during training

In unsupervised learning:

- We have *training* data

$$\mathrm{d} = \{(\mathrm{x}_i), i = 1, ..., n\}\,.$$

- We train a model $p(x)$ to explain/model x
- Our loss function (or model) can be the goal

UPPSALA UNIVERSITET

Unsupervised learning

Goal: Build a good (probabilistic) model $p(x)$ for x

UPPSALA UNIVERSITET

Unsupervised learning

Goal: Build a good (probabilistic) model $p(x)$ for x

Other names for $p(x)$:

- Data model
  $p(x)$ is our *data* generating mechanism
- Generative model
  We can *generate* samples from $p(x)$.

UPPSALA UNIVERSITET

Unsupervised learning

Goal: Build a good (probabilistic) model $p(x)$ for x

Other names for $p(x)$:

- Data model
  $p(x)$ is our *data* generating mechanism
- Generative model
  We can *generate* samples from $p(x)$.

Common use cases for unsupervised learning:

- Generate new observations from $p(x)$
- Study structure in large data
- Anomaly detection
- Create representations for downstream tasks

UPPSALA UNIVERSITET

The Learning Problem

- Goal: A model that can "explain" the data well
- Two main approaches:
  - Clustering: Finding similar observations (rows)
  - Dimensionality reduction: Finding similar variables (columns)

UPPSALA UNIVERSITET

The Learning Problem

- Goal: A model that can "explain" the data well
- Two main approaches:
  - Clustering: Finding similar observations (rows)
  - Dimensionality reduction: Finding similar variables (columns)
- Commonly, we use parametric probabilistic models $p(x|\theta)$ where θ is unknown
- Learning problem: Learn θ to explain the data as good as possible

UPPSALA UNIVERSITET

Example: Autoencoder

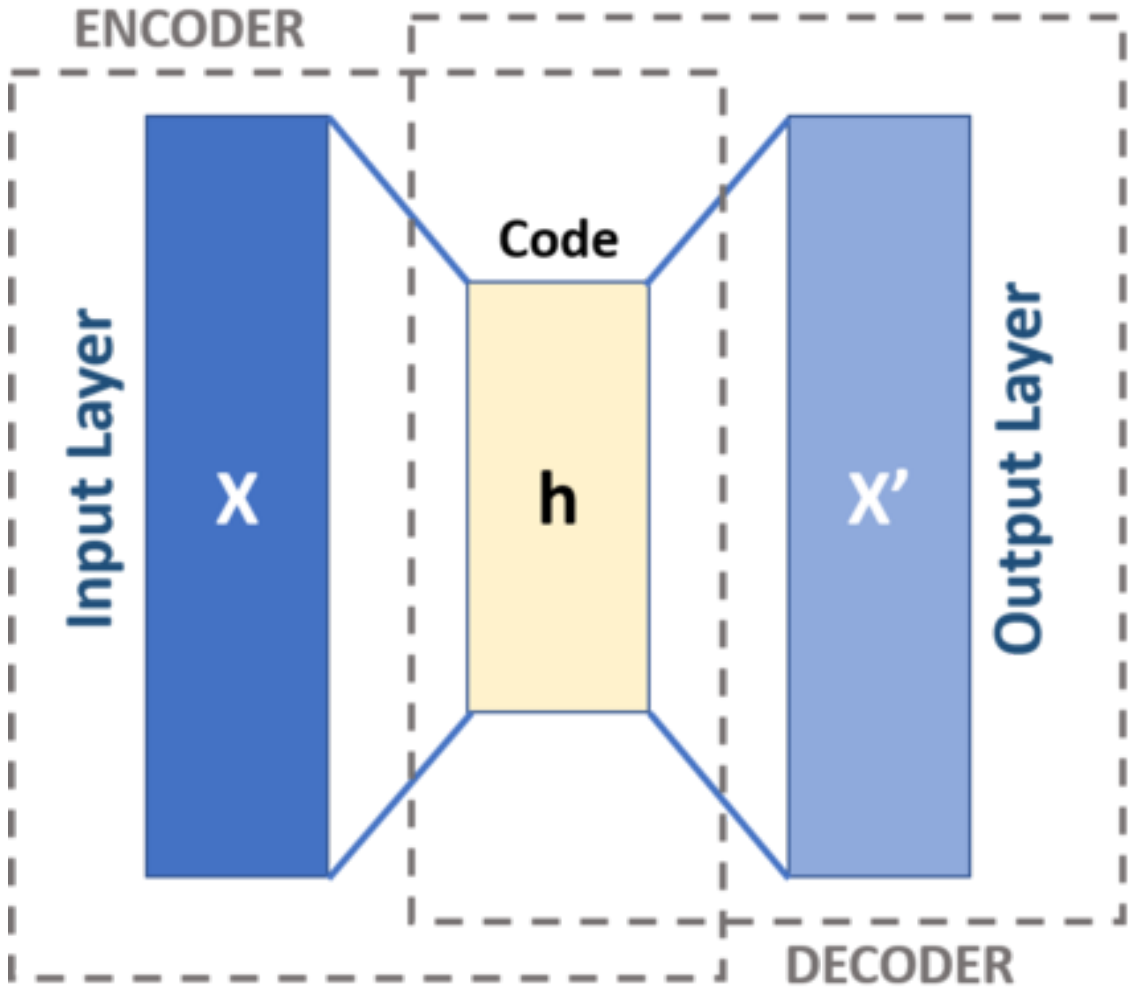


Figure: A Neural Autoencoder (Wikipedia)

UPPSALA UNIVERSITET

Loss functions and evaluation

- Autoencoder uses the difference between the original and reconstructed output

$$L(x) = (d(e(h|x)|h) - x)^2 ,$$

where $d(x|h)$ is the decoder and $e(h|x)$ is the encoder.

UPPSALA UNIVERSITET

Loss functions and evaluation

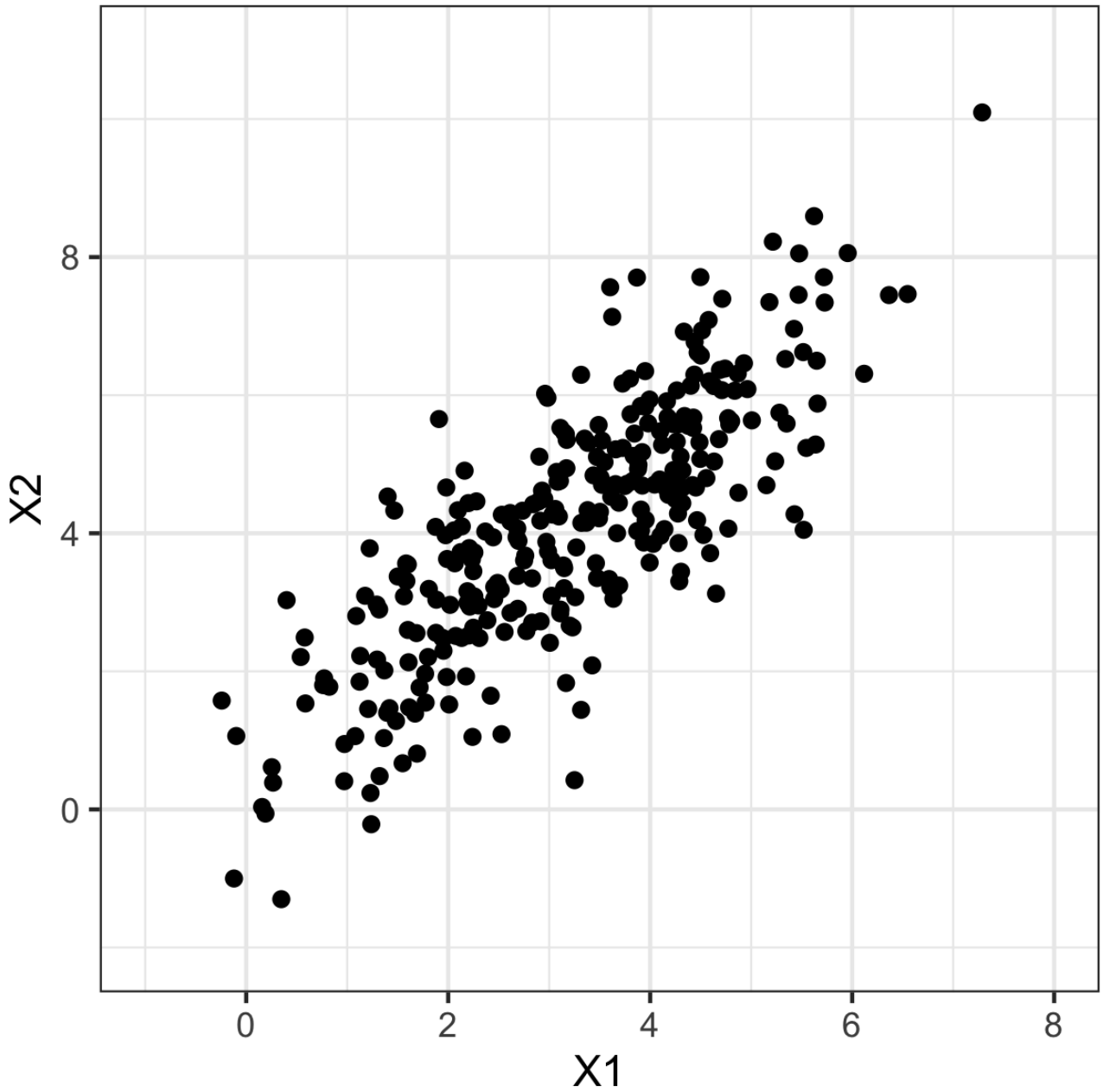
- Autoencoder uses the difference between the original and reconstructed output

$$L(x) = (d(e(h|x)|h) - x)^2 ,$$

  where $d(x|h)$ is the decoder and $e(h|x)$ is the encoder.
- In probabilistic models we can use the log-likelihood ($\mathcal{L}$) Sometimes called perplexity or surprise.
  - High $\mathcal{L}$: The observation is well explained by the model
  - Low $\mathcal{L}$: The observation is badly explained by the model
- Common approach: Evaluate log-likelihood on a held-out set

UPPSALA UNIVERSITET

Example: Bivariate Gaussian model



UPPSALA UNIVERSITET

Example: Bivariate Gaussian model

We assume a Multivariate Gaussian model and estimate μ, Σ from data.

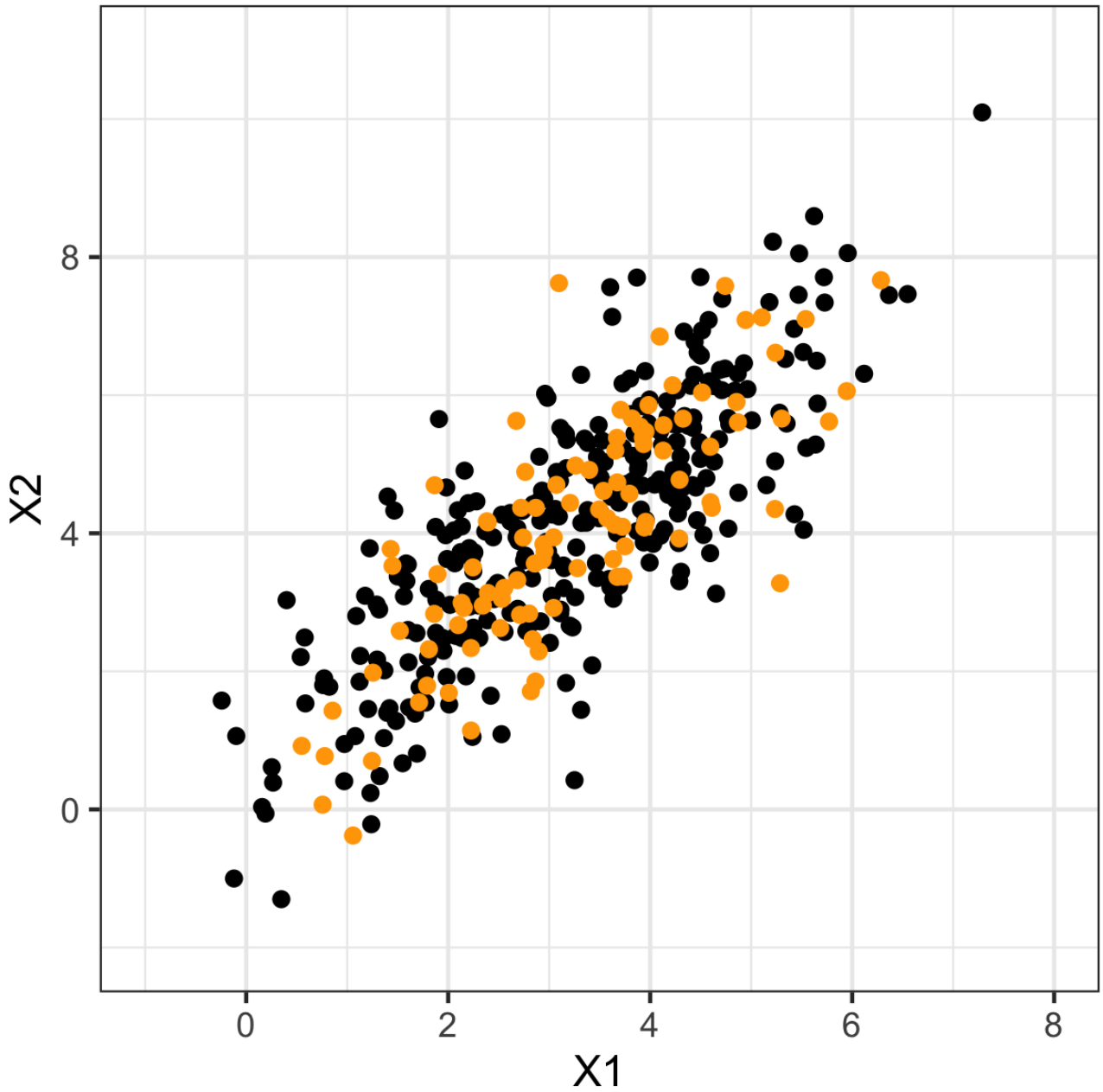
$$\hat{\mu} = [3.19, 4.11]$$

$$\hat{\Sigma} = \begin{bmatrix} 1.95 & 2.05 \\ 2.05 & 3.36 \end{bmatrix}$$

We can now generate new data from MVN$(\hat{\mu}, \hat{\Sigma})$.

UPPSALA UNIVERSITET

Example: Bivariate Gaussian model



UPPSALA UNIVERSITET

Latent variables

- An unobserved or hidden variable or "factor"

UPPSALA UNIVERSITET

Latent variables

- An unobserved or hidden variable or "factor"
- A parameter specific to some or a few observations or features

UPPSALA UNIVERSITET

Latent variables

- An unobserved or hidden variable or "factor"
- A parameter specific to some or a few observations or features
- Often these latent variables can be of interest

UPPSALA UNIVERSITET

Example: Hidden Markov Model

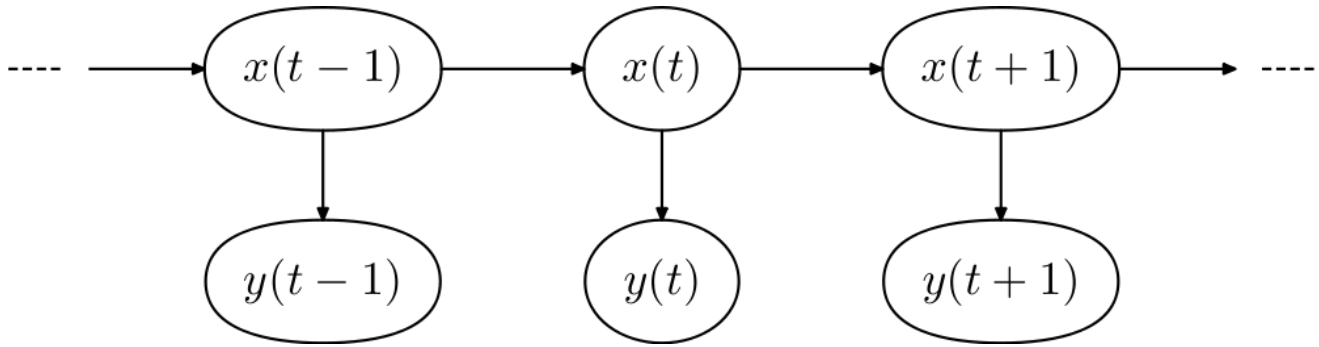


Figure: A Hidden Markov Model (Wikipedia). Note that x is unobserved and y is observed.

UPPSALA UNIVERSITET

Example: Factor Analysis

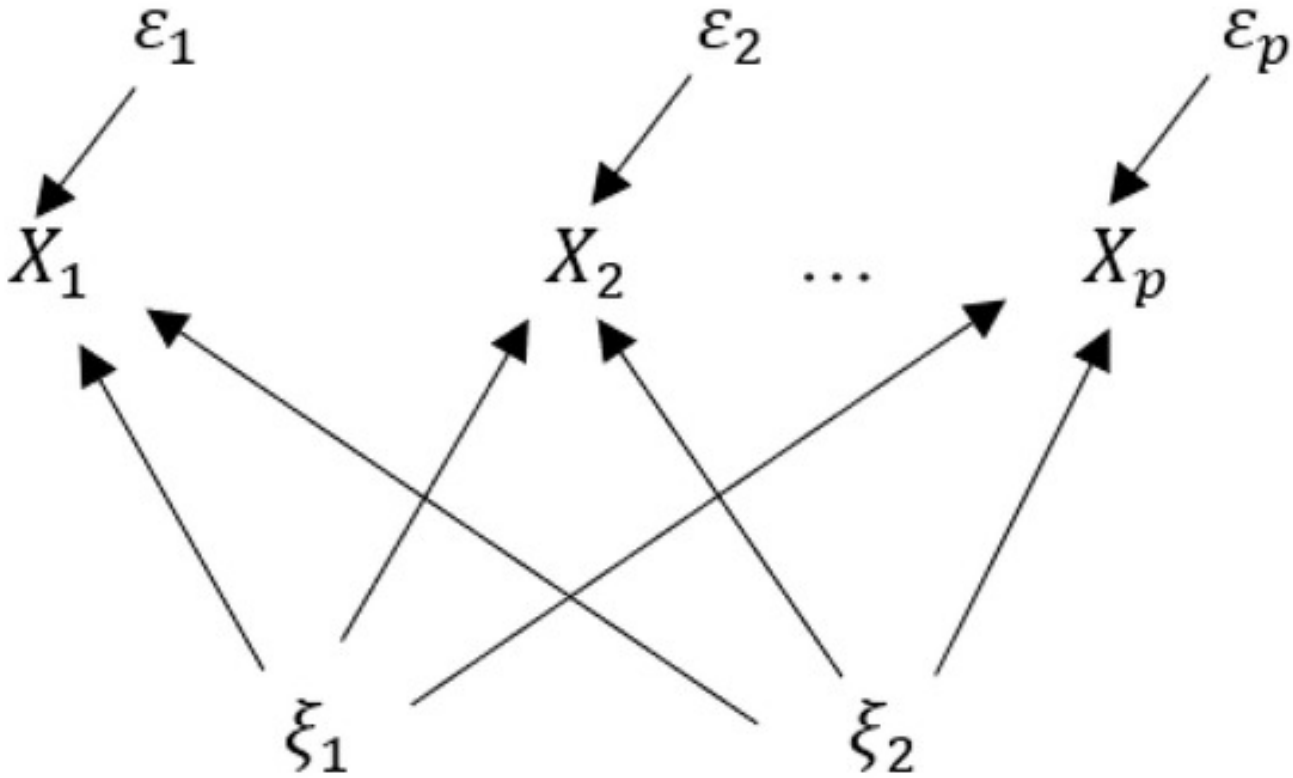


Figure: A Factor Analysis Model (Eshima, Tabata and Borroni, 2018, edited).

UPPSALA UNIVERSITET

Clustering

- Separate observations x_i into groups or segments

UPPSALA UNIVERSITET

Clustering

- Separate observations x_i into groups or segments
- What is a cluster "is" depends on the model/(dis)similarity.
- (Dis)similarity:

$$D(x_i, x_j) = \sum_{k=1}^{P} d_k(x_{i,k}, x_{j,k})$$

- A common dissimilarity is the squared distance

$$d_k(x_{i,k}, x_{j,k}) = (x_{i,k} - x_{j,k})^2$$

UPPSALA UNIVERSITET

Clustering

- Separate observations x_i into groups or segments
- What is a cluster "is" depends on the model/(dis)similarity.
- (Dis)similarity:

$$D(x_i, x_j) = \sum_{k=1}^{P} d_k(x_{i,k}, x_{j,k})$$

- A common dissimilarity is the squared distance

$$d_k(x_{i,k}, x_{j,k}) = (x_{i,k} - x_{j,k})^2$$

- Clustering can be divided into:
  - Hard clustering
  - Soft clustering

UPPSALA UNIVERSITET

Clustering

- Separate observations x_i into groups or segments
- What is a cluster "is" depends on the model/(dis)similarity.
- (Dis)similarity:

$$D(x_i, x_j) = \sum_{k=1}^{P} d_k(x_{i,k}, x_{j,k})$$

- A common dissimilarity is the squared distance

$$d_k(x_{i,k}, x_{j,k}) = (x_{i,k} - x_{j,k})^2$$

- Clustering can be divided into:
  - Hard clustering
  - Soft clustering
- Clustering can also be divided into:
  - Hiearchical clustering
  - Flat clustering

UPPSALA UNIVERSITET

Clustering

- Separate observations x_i into groups or segments
- What is a cluster "is" depends on the model/(dis)similarity.
- (Dis)similarity:

$$D(x_i, x_j) = \sum_{k=1}^{P} d_k(x_{i,k}, x_{j,k})$$

- A common dissimilarity is the squared distance

$$d_k(x_{i,k}, x_{j,k}) = (x_{i,k} - x_{j,k})^2$$

- Clustering can be divided into:
  - Hard clustering
  - Soft clustering
- Clustering can also be divided into:
  - Hiearchical clustering
  - Flat clustering
- There is a ton of different algorithms and methods...

UPPSALA UNIVERSITET

k-means

- Popular in practice and a classic in unsupervised machine learning

UPPSALA UNIVERSITET

k-means

- Popular in practice and a classic in unsupervised machine learning
- Hard, flat clustering
- Simple and effective

UPPSALA UNIVERSITET

k-means

- Popular in practice and a classic in unsupervised machine learning
- Hard, flat clustering
- Simple and effective
- Model: x_i is close to one of $m_1, ..., m_K$ vectors
- Loss function:

$$l_{\mathrm{m}}(x) = \min_{\mathrm{m}} x - m_k)^2$$

UPPSALA UNIVERSITET

k-means

- Popular in practice and a classic in unsupervised machine learning
- Hard, flat clustering
- Simple and effective
- Model: x_i is close to one of $m_1, ..., m_K$ vectors
- Loss function:

$$l_{\mathrm{m}}(x) = \min_{\mathrm{m}} x - m_k)^2$$

- Hyperparameter: K (the number of clusters)
- Parameters: m (a $K \times P$ matrix).

UPPSALA UNIVERSITET

k-means

- Popular in practice and a classic in unsupervised machine learning
- Hard, flat clustering
- Simple and effective
- Model: x_i is close to one of $m_1, ..., m_K$ vectors
- Loss function:

$$l_{\mathrm{m}}(x) = \min_{\mathrm{m}} x - m_k)^2$$

- Hyperparameter: K (the number of clusters)
- Parameters: m (a $K \times P$ matrix).
- A difficult problem: K^n possibilities

UPPSALA UNIVERSITET

k-means algorithm

Algorithm 10.1 *K-Means Clustering*

1. Randomly assign a number, from 1 to K, to each of the observations. These serve as initial cluster assignments for the observations.
2. Iterate until the cluster assignments stop changing:
   (a) For each of the K clusters, compute the cluster *centroid*. The kth cluster centroid is the vector of the p feature means for the observations in the kth cluster.
   (b) Assign each observation to the cluster whose centroid is closest (where *closest* is defined using Euclidean distance).

Figure: The k-means cluster algorithm (Garreth et al, 2013, Alg. 10.1).

UPPSALA UNIVERSITET

k-means clustering

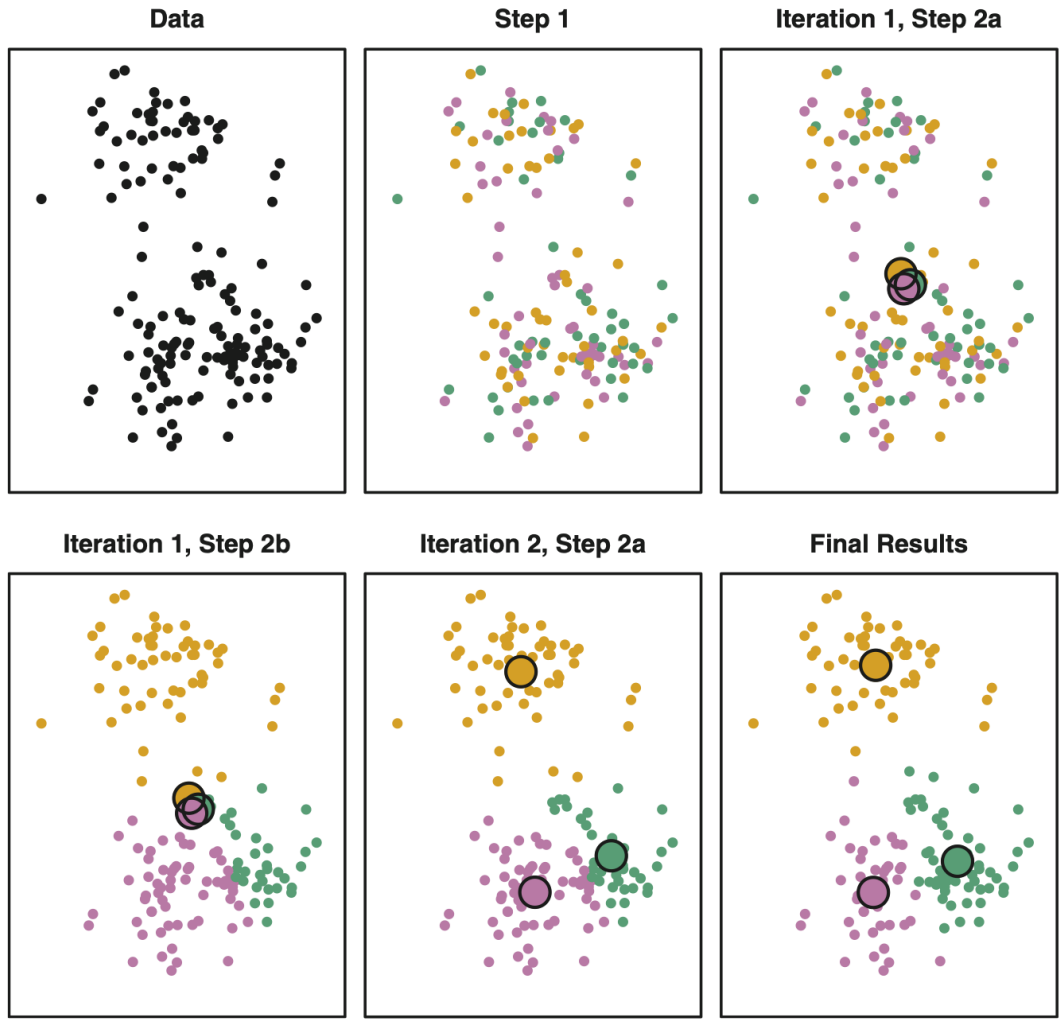


Figure: The k-means cluser algorithm (Garreth et al, 2013, Fig. 10.6).

UPPSALA UNIVERSITET

k-means clustering

- k-means finds local modes
- Re-run algorithm with many different starting values
- Choose the best by the best loss

UPPSALA UNIVERSITET

k-means clustering

- k-means finds local modes
- Re-run algorithm with many different starting values
- Choose the best by the best loss
- There exists many developments
  - scaling to large data
  - generalized loss
  - approaches to find a good K

UPPSALA UNIVERSITET

k-means clustering

Figure: The k-means cluser algorithm (Garreth et al, 2013, Fig. 10.7).

UPPSALA UNIVERSITET

Problems with k-means

- Clusters might
  - overlap
  - have different forms

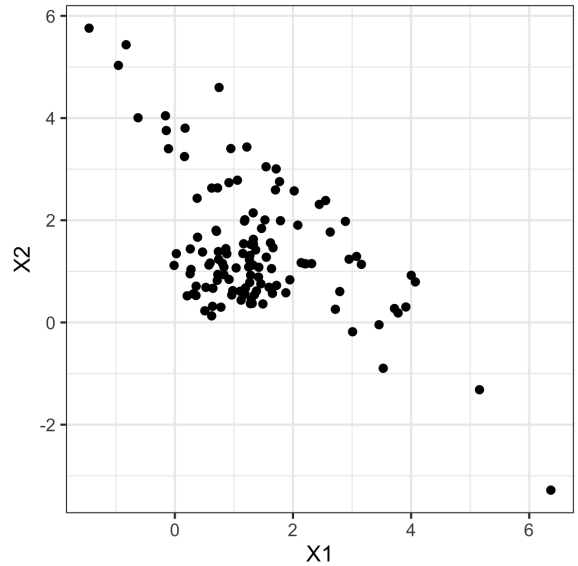


Figure: Two clusters with different shapes.

UPPSALA UNIVERSITET

Problems with k-means

- Clusters might
  - overlap
  - have different forms

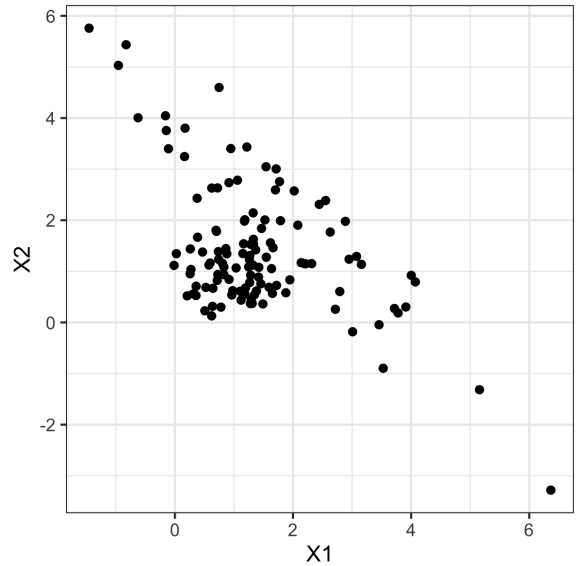


Figure: Two clusters with different shapes.

We can solve these problems using probabilistic models

UPPSALA UNIVERSITET

Finite Mixture Models

The finite mixture model can be expressed as:

$$y_i = \sum_{k=1}^{K} \pi_k \phi_k(\theta_k)\,,$$

- The parts of a (finite) mixture model:
  - The number of components: K

UPPSALA UNIVERSITET

Finite Mixture Models

The finite mixture model can be expressed as:

$$y_i = \sum_{k=1}^{K} \pi_k \phi_k(\theta_k)\,,$$

- The parts of a (finite) mixture model:
  - The number of components: K
  - The proportions of observation from component k: π_k

UPPSALA UNIVERSITET

Finite Mixture Models

The finite mixture model can be expressed as:

$$y_i = \sum_{k=1}^{K} \pi_k \phi_k(\theta_k)\,,$$

- The parts of a (finite) mixture model:
  - The number of components: K
  - The proportions of observation from component k: π_k
  - The density of component k: ϕ_k

UPPSALA UNIVERSITET

Finite Mixture Models

The finite mixture model can be expressed as:

$$y_i = \sum_{k=1}^{K} \pi_k \phi_k(\theta_k)\,,$$

- The parts of a (finite) mixture model:
  - The number of components: K
  - The proportions of observation from component k: π_k
  - The density of component k: ϕ_k
  - The parameters of component k: θ_k

UPPSALA UNIVERSITET

Finite Mixture Models

- Usually, we
  - set K, and
  - use the same density for all k.

UPPSALA UNIVERSITET

Finite Mixture Models

- Usually, we
  - set K, and
  - use the same density for all k.
- We can simulate data from the model as compund probability distribution:
  1. Simulate cluster assignments for all i:

$$z_i \sim \text{Categorical}(\pi)$$

  2. Generate y_i conditioned on z_i:

$$y_i \sim \phi_{z_i}(\theta_{z_i})$$

- Cluster assignments z_i are the latent variables

UPPSALA UNIVERSITET

Gaussian Mixture Models (GMM)

- The (finite) Gaussian mixture model:

$$y_i = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mu_k, \Sigma_k)\,,$$

where μ_k and Σ_k depend on the dimensionality of y_i.

UPPSALA UNIVERSITET

Gaussian Mixture Models (GMM)

- The (finite) Gaussian mixture model:

$$y_i = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mu_k, \Sigma_k)\,,$$

  where μ_k and Σ_k depend on the dimensionality of y_i.
- GMM is a universial approximator of densities

UPPSALA UNIVERSITET

Example: Simulate data from a GMM

1. Generate cluster assignments:

$$z_i \sim \text{Categorical}(\pi = [0.4, 0.6])$$

2. Generate observation conditioned on cluster assignment:

$$y_i \sim \mathcal{N}(\mu_k, \Sigma_k)\,,$$

where

$$\mu_1 = [2, 2]\,, \mu_2 = [1, 1] \text{ and}$$

$$\Sigma_1 = \begin{bmatrix} 3 & -2.7 \\ -2.7 & 3 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 0.2 & 0 \\ 0 & 0.2 \end{bmatrix}$$

UPPSALA UNIVERSITET

- Introuction to unsupervised learning
  - Latent variables
- Clustering
- Mixture models
- Expectation Maximization
- probabilistic PCA

Simulated data from a GMM

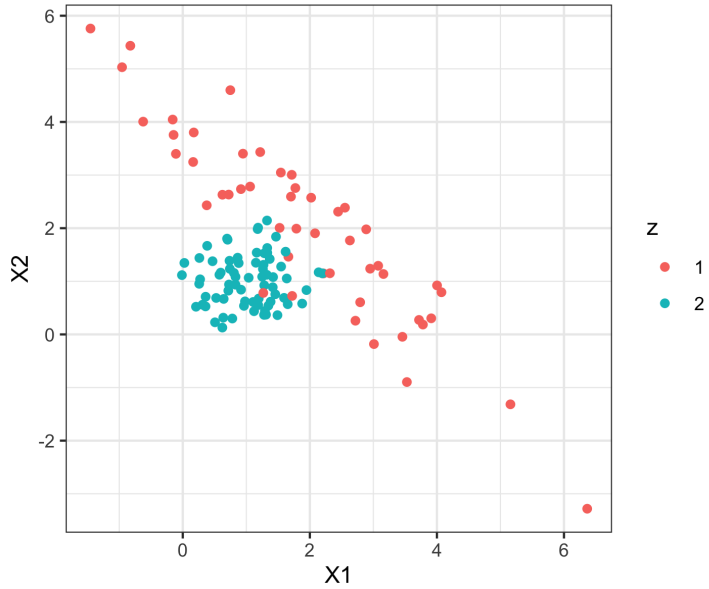


Figure: Simulated mixture data with the latent variable z.

UPPSALA UNIVERSITET

Mixtures of Multinomial distributions

What distribution (ϕ) should I use?

UPPSALA UNIVERSITET

Mixtures of Multinomial distributions

What distribution (ϕ) should I use?

Depends on your data (y).

UPPSALA UNIVERSITET

Mixtures of Multinomial distributions

What distribution (ϕ) should I use?

Depends on your data (y).

$$y_i = \sum_{k=1}^{K} \pi_k \text{Multinomial}(\text{p}_k)$$

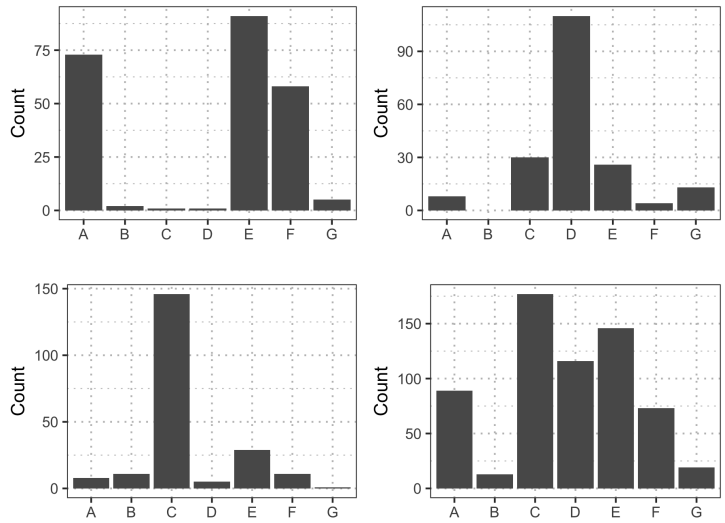


Figure: Mixture of Multinomials.

UPPSALA UNIVERSITET

Estimating Mixtures Models

- We are interested in estimating θ_k and π_k for the model

$$y_i = \sum_{k=1}^{K} \pi_k \phi(\theta_k)\,,$$

- Hence we want to maximize the log-likelihood

$$\mathcal{L}(\pi, \theta) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \phi(y_i|\theta_k) \right)$$

- This is difficult, although if we only knew z...

UPPSALA UNIVERSITET

Estimating Mixtures Models

- We are interested in estimating θ_k and π_k for the model

$$y_i = \sum_{k=1}^{K} \pi_k \phi(\theta_k)\,,$$

- Hence we want to maximize the log-likelihood

$$\mathcal{L}(\pi, \theta) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \phi(y_i|\theta_k) \right)$$

- This is difficult, although if we only knew z...

$$\mathcal{L}_{\text{full}}(\pi, \theta, \mathrm{z}) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} I(z_i = k) \phi(y_i|\theta_k) \right) + \log(\pi_k^I(z_i = k))$$

$$= \sum_{i=1}^{N} \sum_{k=1}^{K} I(z_i = k) \log \phi(y_i|\theta_k) + I(z_i = k) \log(\pi_k$$

- So if we knew z it is essentially just maximizing $\mathcal{L}$ for each cluster separately.

UPPSALA UNIVERSITET

The Expectation

- But, we dont know z.

UPPSALA UNIVERSITET

The Expectation

- But, we dont know z.
- Although, we could compute the expected cluster assignment

$$\gamma_i = E_{z_i}(\mathcal{L}_{\text{full}}|\theta, y_i)\,.$$

- γ_i can be seen as observation is weights for each cluster
- γ_i is sometimes refered to as the responsibility.

UPPSALA UNIVERSITET

The Maximization

- Now, given γ we can (hopefully) easier maximize θ.

UPPSALA UNIVERSITET

The Maximization

- Now, given γ we can (hopefully) easier maximize θ.
- We maximize $\mathcal{L}_{\text{full}}$ given γ and y.

UPPSALA UNIVERSITET

The Maximization

- Now, given γ we can (hopefully) easier maximize θ.
- We maximize $\mathcal{L}_{\text{full}}$ given γ and y.
- We usually choose ψ so the maximization
  - is a nice analytical expression.
  - end up with a weighted MLE.

UPPSALA UNIVERSITET

Example: EM for a Gaussian Mixture

Algorithm 8.1 *EM Algorithm for Two-component Gaussian Mixture.*

1. Take initial guesses for the parameters $\hat{\mu}_1, \hat{\sigma}_1^2, \hat{\mu}_2, \hat{\sigma}_2^2, \hat{\pi}$ (see text).

2. *Expectation Step*: compute the responsibilities

$$\hat{\gamma}_i = \frac{\hat{\pi}\phi_{\hat{\theta}_2}(y_i)}{(1-\hat{\pi})\phi_{\hat{\theta}_1}(y_i) + \hat{\pi}\phi_{\hat{\theta}_2}(y_i)}, \; i = 1, 2, \ldots, N. \qquad (8.42)$$

3. *Maximization Step*: compute the weighted means and variances:

$$\hat{\mu}_1 = \frac{\sum_{i=1}^N (1-\hat{\gamma}_i) y_i}{\sum_{i=1}^N (1-\hat{\gamma}_i)}, \qquad \hat{\sigma}_1^2 = \frac{\sum_{i=1}^N (1-\hat{\gamma}_i)(y_i - \hat{\mu}_1)^2}{\sum_{i=1}^N (1-\hat{\gamma}_i)},$$

$$\hat{\mu}_2 = \frac{\sum_{i=1}^N \hat{\gamma}_i y_i}{\sum_{i=1}^N \hat{\gamma}_i}, \qquad \hat{\sigma}_2^2 = \frac{\sum_{i=1}^N \hat{\gamma}_i (y_i - \hat{\mu}_2)^2}{\sum_{i=1}^N \hat{\gamma}_i},$$

and the mixing probability $\hat{\pi} = \sum_{i=1}^N \hat{\gamma}_i / N$.

4. Iterate steps 2 and 3 until convergence.

Figure: The EM algorithm for a two component Gaussian mixture (Hastie et al 2008, Alg. 10.1)

UPPSALA UNIVERSITET

The EM algorithm

- Properties of the EM algorithm:
  - The EM-algorithm will converge to a local mode

UPPSALA UNIVERSITET

The EM algorithm

- Properties of the EM algorithm:
  - The EM-algorithm will converge to a local mode
  - Each iteration will always increase the likelihood
    - Can be proven straight-forward using Jensens inequality

UPPSALA UNIVERSITET

The EM algorithm

- Properties of the EM algorithm:
  - The EM-algorithm will converge to a local mode
  - Each iteration will always increase the likelihood
    - Can be proven straight-forward using Jensens inequality
  - We can interpret the final γ_i as the expected cluster
    Hence, the EM algorithm is a soft clustering approach.
- Expanding the likelihood with latent variables (z) is called data augmentation.
  Note! Not the same as data augmentation in CNNs.

UPPSALA UNIVERSITET

Connections to other approaches

- If we set $z_i = \text{argmax}(\gamma_i)$: k-means

UPPSALA UNIVERSITET

Connections to other approaches

- If we set $z_i = \text{argmax}(\gamma_i)$: k-means
- If we sample z_i according to γ: stochastic EM

UPPSALA UNIVERSITET

Dimensionality reduction

- So far focus has been on (clustering) observations
- Now, we will adress the other large area of UL: dimensionality reduction

UPPSALA UNIVERSITET

Dimensionality reduction

- So far focus has been on (clustering) observations
- Now, we will adress the other large area of UL: dimensionality reduction
- The starting point is Principal Component Analysis (PCA)
- PCA can be used for
  - Reduce the dimensionality of our data
  - Produce lower-dimensional features in a prediction model
  - Discover underlying latent variables (factors)

UPPSALA UNIVERSITET

Dimensionality reduction

- So far focus has been on (clustering) observations
- Now, we will adress the other large area of UL: dimensionality reduction
- The starting point is Principal Component Analysis (PCA)
- PCA can be used for
  - Reduce the dimensionality of our data
  - Produce lower-dimensional features in a prediction model
  - Discover underlying latent variables (factors)
- More details in the multivariate course.

UPPSALA UNIVERSITET

Principal Component Analysis

- Basic idea: We can summarize our data using K principal components (PC)
- The PCA "model" can be expressed as

$$X \approx b + WH^T,$$

  where $H \in \mathbb{R}^{n \times k}$, $W \in \mathbb{R}^{k \times p}$, $b \in \mathbb{R}^p$ and $X \in \mathbb{R}^{n \times p}$.
- H can be seen as a latent factors
- W can be seen as a factor loadings
- We assume that W is orthogonal: $W^T W = I$

UPPSALA UNIVERSITET

Principal Component Analysis

- The PCA model

$$X \approx b + WH^T,$$

- The loss function, also called reconstruction error:

$$J(b, W, H) = \sum_{i}^{N} ||x_i - b + Wh_i^T||^2$$

UPPSALA UNIVERSITET

Principal Component Analysis

- The PCA model

$$X \approx b + WH^T,$$

- The loss function, also called reconstruction error:

$$J(b, W, H) = \sum_{i}^{N} ||x_i - b + Wh_i^T||^2$$

- This can be minmized using Singular Value Decomposition

UPPSALA UNIVERSITET

PCA: Conceptual depiction

$$\underset{(n \times p)}{X} \approx \underset{(p \times k)}{W} \times \underset{(k \times n)}{H^T}$$

Figure: Conceptual depiction of PCA.

UPPSALA UNIVERSITET

probabilistic PCA (pPCA)

- PCA is not a probabilistic model

UPPSALA UNIVERSITET

probabilistic PCA (pPCA)

- PCA is not a probabilistic model
- probabilistic PCA

$$x_i = b + Wh_i^T + \epsilon_i$$

  where $\epsilon \sim N(0, \Psi)$
- In pPCA, we assume $\Psi = \sigma \mathrm{I}$
- We also assume that $h_i \sim N(0, I)$

UPPSALA UNIVERSITET

probabilistic PCA (pPCA)

- PCA is not a probabilistic model
- probabilistic PCA

$$x_i = b + Wh_i^T + \epsilon_i$$

  where $\epsilon \sim N(0, \Psi)$
- In pPCA, we assume $\Psi = \sigma \mathrm{I}$
- We also assume that $h_i \sim N(0, I)$
- We can integrate out H and get the model

$$x_i \sim N(b, WW^T + \Psi)$$

UPPSALA UNIVERSITET

probabilistic PCA

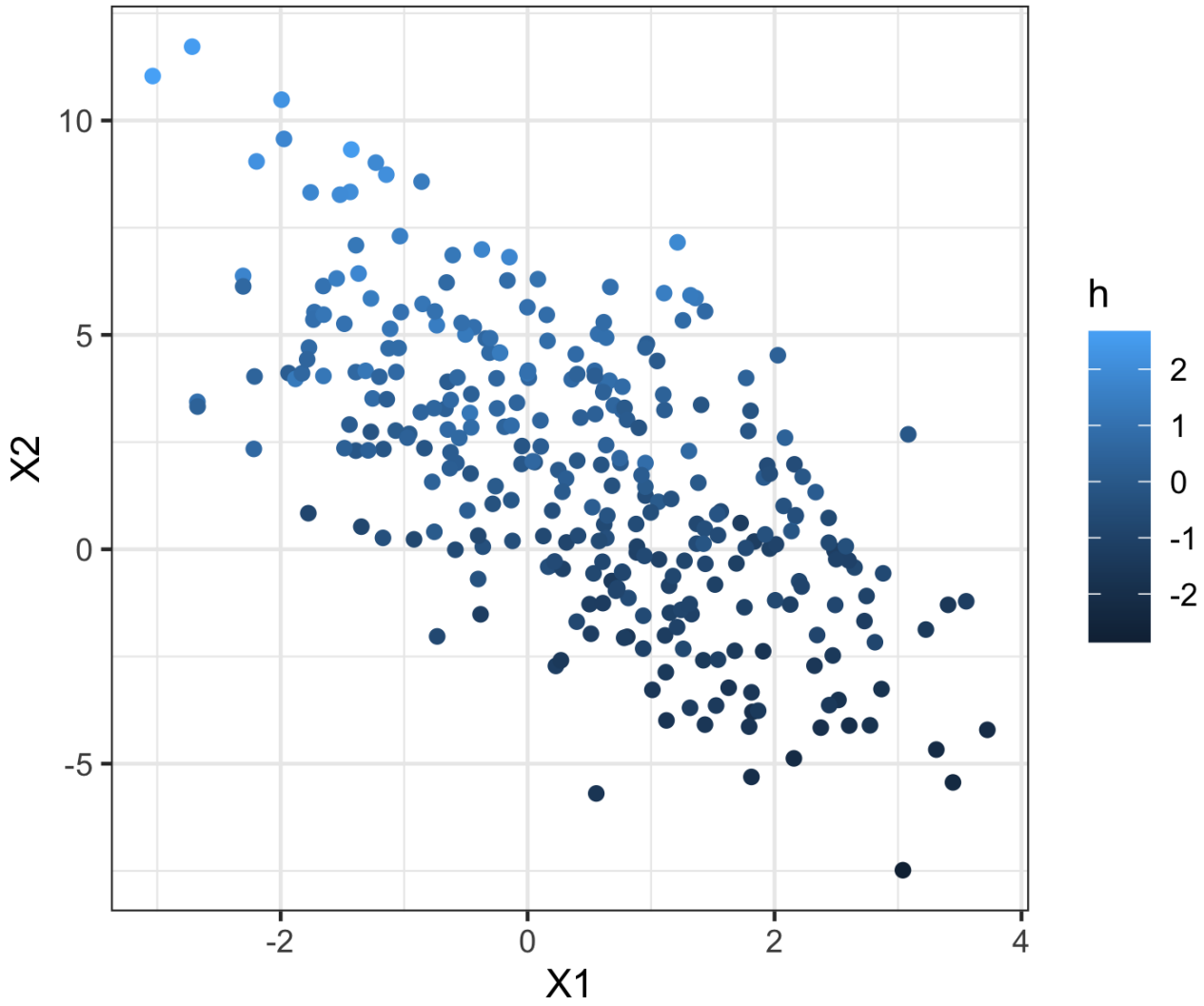


Figure: Data from a pPCA model with $W = (-1, 3)^T$, $b = (0.5, 2)$ and $\sigma^2 = 1$

UPPSALA UNIVERSITET

probabilistic PCA

- probabilistic PCA

$$x_i = b + Wh_i^T + \epsilon_i$$

UPPSALA UNIVERSITET

probabilistic PCA

- probabilistic PCA

$$x_i = b + Wh_i^T + \epsilon_i$$

- We can now estimate our parameters using EM (or Bayesian methods)
- Enables us to combine with other models (e.g. mixture of pPCA)

UPPSALA UNIVERSITET

probabilistic PCA

- probabilistic PCA

$$x_i = b + Wh_i^T + \epsilon_i$$

- We can now estimate our parameters using EM (or Bayesian methods)
- Enables us to combine with other models (e.g. mixture of pPCA)
- And as we will see next week, is the basic building block for many high-dimensional problems

UPPSALA UNIVERSITET

Connections to PCA and Factor Analysis

- probabilistic PCA

$$x_i = b + Wh_i^T + \epsilon_i$$

  where $\epsilon \sim N(0, \Psi)$
- pPCA is closely connected to PCA and Factor Analysis:
  - $\sigma I \to 0$: pPCA $\to$ PCA
  - $\Psi = \text{diag}(\sigma_1, ..., \sigma_p, ..., \sigma_P)$: pPCA $\to$ Factor Analysis