UPPSALA
UNIVERSITET

- Practicalities
- Introduction to unsupervised learning
  - — Latent variables
- Clustering
  - — k-means
- Mixture models
- Expectation-Maximization
- Probabilistic PCA

Machine learning – Block 6

Måns Magnusson
Department of Statistics, Uppsala University

Autumn 2025

This week's lectures

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

Practicalities

- Last push this week!

Section 2

Introduction to unsupervised learning

Supervised and Unsupervised learning

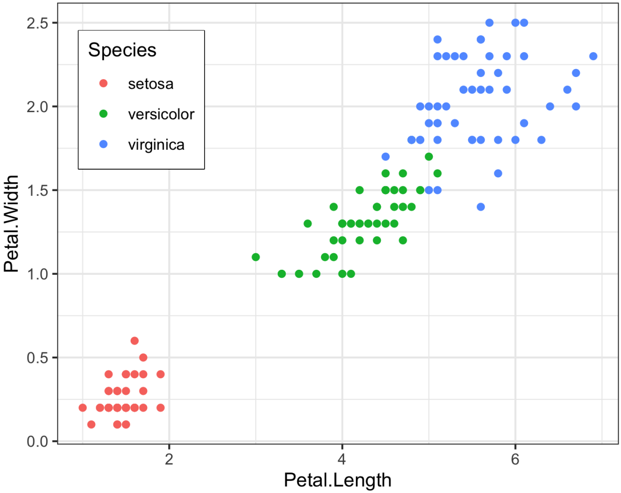


Figure: The Supervised Problem

Supervised and Unsupervised learning

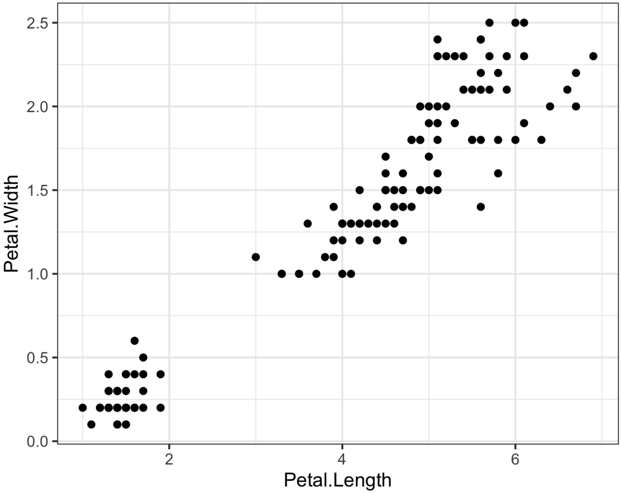


Figure: The Unsupervised Problem

Supervised and Unsupervised learning

In supervised learning:

- We have *training* data

$$\mathcal{D}_{yx} = \{(y_i, x_i)\}_{i=1}^{n}$$

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

Supervised and Unsupervised learning

In supervised learning:

- We have *training* data

$$\mathcal{D}_{yx} = \{(y_i, x_i)\}_{i=1}^{n}$$

- 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

$$\mathcal{D}_{x} = \{(x_i)\}_{i=1}^{n}$$

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

Unsupervised learning

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

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 $p(x)$.

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

The Learning Problem

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

The Learning Problem

- Goal: A model that can "explain" the data well
- Two main (basic) 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

Example: Autoencoder

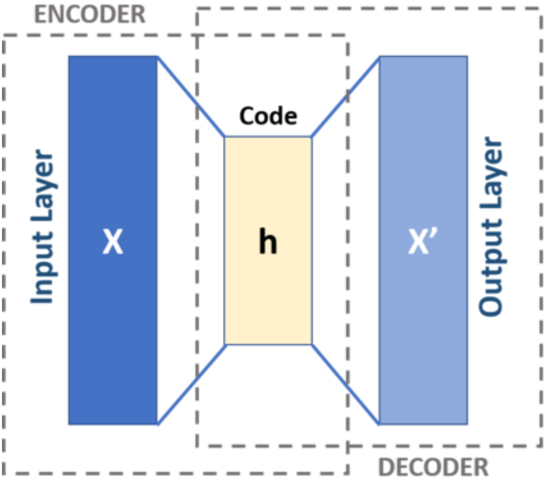


Figure: A Neural Autoencoder (Wikipedia)

Loss functions and evaluation

- For a deterministic autoencoder a common reconstruction loss is:

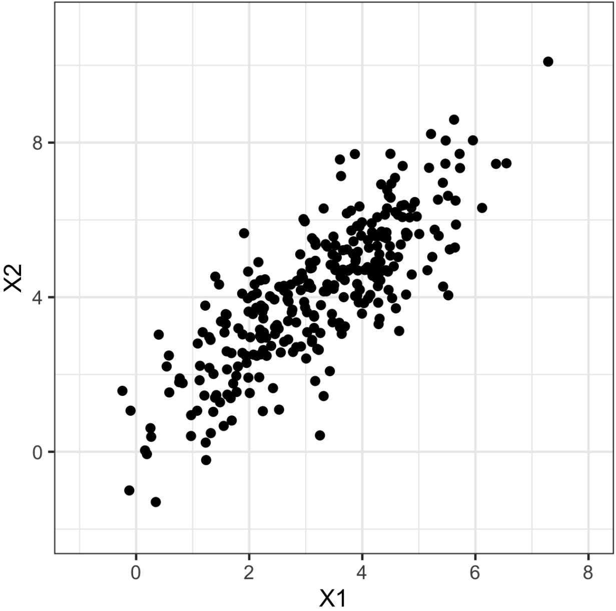
$$L(x) = \|x - \hat{x}\|^2 = \|x - d(e(x))\|^2$$

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

Loss functions and evaluation

- For a deterministic autoencoder a common reconstruction loss is:
$$L(x) = \|x - \hat{x}\|^2 = \|x - d(e(x))\|^2$$
where $d(h)$ is the decoder and $e(x)$ is the encoder.
- In probabilistic models we can use the log-likelihood, $\mathcal{L}(x) = \log p(x)$, or perplexity (a function of the log-likelihood).
  - High $\mathcal{L}(x)$: The observation is well explained by the model
  - Low loss $\mathcal{L}(x)$: The observation is badly explained by the model
- Evaluate log-likelihood on a held-out validation set

Example: Bivariate Gaussian model



Example: Bivariate Gaussian model

We assume a $p(x)$ is a Multivariate Gaussian model and estimate μ, Σ from data.

Example: Bivariate Gaussian model

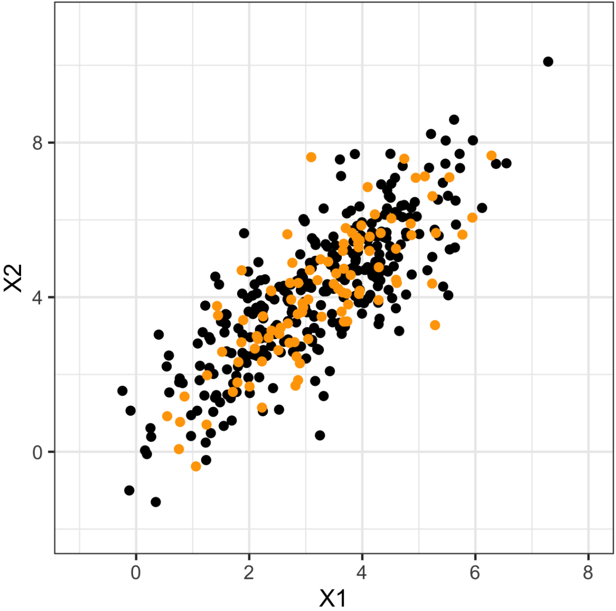
We assume a $p(x)$ is a Multivariate Gaussian model and estimate μ, Σ from data.

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

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

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

Example: Bivariate Gaussian model



Latent variables

- An unobserved or hidden variable or ”factor”

Latent variables

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

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 main interest

Example: Hidden Markov Model

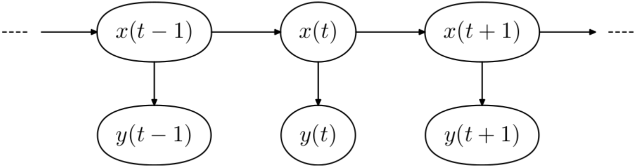


Figure: A Hidden Markov Model (Wikipedia).

Here: x is unobserved/latent and y is observed.

Example: Factor Analysis

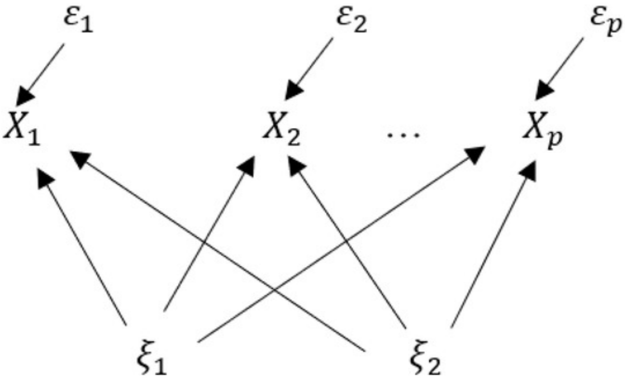


Figure: A Factor Analysis Model (Eshima, Tabata and Borroni, 2018, edited).

Section 3

Clustering

Clustering

- Separate observations x_i into groups or segments

Clustering

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

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

(This assumes component-wise additive dissimilarity).

Clustering

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

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

(This assumes component-wise additive dissimilarity).

- A common dissimilarity is the squared distance

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

Clustering

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

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

(This assumes component-wise additive dissimilarity).

- 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

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

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

(This assumes component-wise additive dissimilarity).

- 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:
  - Hierarchical clustering
  - Flat clustering

Clustering

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

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

(This assumes component-wise additive dissimilarity).

- 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:
  - Hierarchical clustering
  - Flat clustering
- There is a ton of different algorithms and methods...

k-means

- Popular in practice and a classic in unsupervised machine learning

k-means

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

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 $m_1, \dots, m_K$ vectors
- Loss function:

$$L(m) = \sum_{i=1}^{n} \min_{k \in \{1,\dots,K\}} \|x_i - m_k\|^2$$

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 $m_1, \ldots, m_K$ vectors
- Loss function:

$$L(m) = \sum_{i=1}^{n} \min_{k \in \{1,\ldots,K\}} \|x_i - m_k\|^2$$

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

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 $m_1, \ldots, m_K$ vectors
- Loss function:

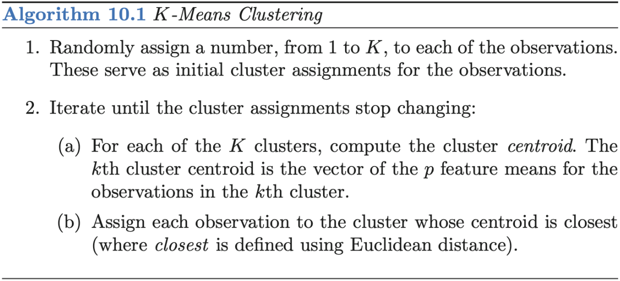
$$L(m) = \sum_{i=1}^{n} \min_{k \in \{1,\ldots,K\}} \|x_i - m_k\|^2$$

- Hyperparameter: K (the number of clusters)
- Parameters: $\mathbf{m}$ (a $K \times P$ matrix).
- The combinatorial assignment problem has K^n possible clusterings, making global optimization infeasible.

UPPSALA
UNIVERSITET

- Practicalities
- Introduction to unsupervised learning
  - Latent variables
- Clustering
  - k-means
- Mixture models
- Expectation-Maximization
- Probabilistic PCA

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).

k-means clustering

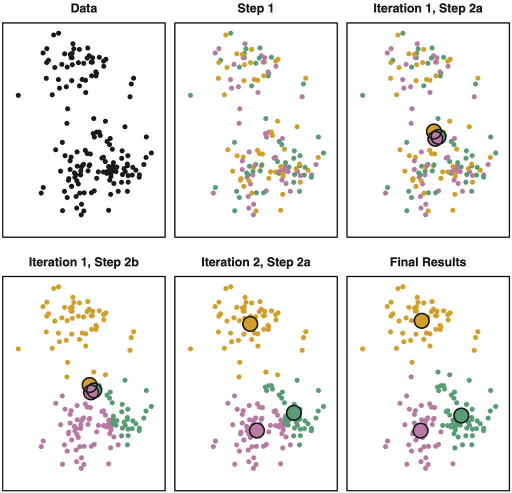


Figure: The k-means cluster algorithm (Garreth et al, 2013, Fig. 10.6).

k-means clustering

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

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

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

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
- Choosing the number of clusters K, common approaches:
  - Elbow method (within-cluster variance)
  - Information criteria: AIC, BIC (mixture models)
  - Cross-validated log-likelihood
  - Bayesian nonparametrics (e.g. Dirichlet process mixtures)

k-means clustering

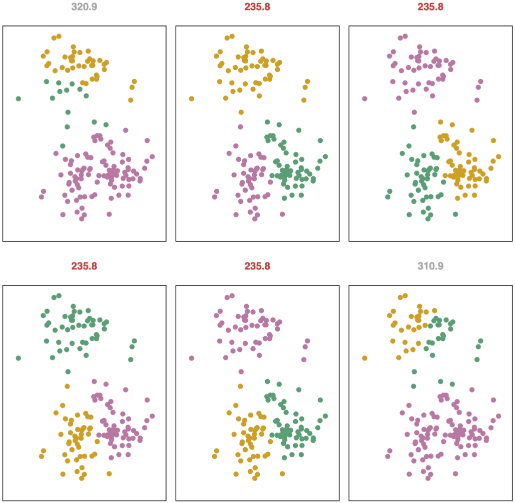


Figure: The k-means cluster algorithm (Garreth et al, 2013, Fig. 10.7).

Problems with k-means

- Clusters might
  - overlap
  - have different forms

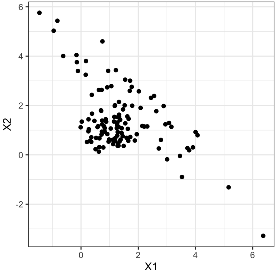


Figure: Two clusters with different shapes.

Problems with k-means

- Clusters might
  - overlap
  - have different forms

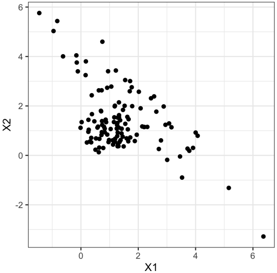


Figure: Two clusters with different shapes.

We can solve these problems using probabilistic models

Section 4

Mixture models

Finite Mixture Models

The finite mixture model density can be expressed as:

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

where $\pi_k \geq 0$, $\sum_{k=1}^{K} \pi_k = 1$, and $\phi(\cdot \mid \theta_k)$ is a density with parameters θ_k.

Finite Mixture Models

The finite mixture model density can be expressed as:

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

where $\pi_k \geq 0$, $\sum_{k=1}^{K} \pi_k = 1$, and $\phi(\cdot \mid \theta_k)$ is a density with parameters θ_k. Equivalent latent-variable representation:

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

$$y_i \mid z_i = k \sim \phi(\cdot \mid \theta_k).$$

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

Finite Mixture Models

The finite mixture model density can be expressed as:

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

where $\pi_k \geq 0$, $\sum_{k=1}^{K} \pi_k = 1$, and $\phi(\cdot \mid \theta_k)$ is a density with parameters θ_k. Equivalent latent-variable representation:

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

$$y_i \mid z_i = k \sim \phi(\cdot \mid \theta_k).$$

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

Finite Mixture Models

The finite mixture model density can be expressed as:

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

where $\pi_k \geq 0$, $\sum_{k=1}^{K} \pi_k = 1$, and $\phi(\cdot \mid \theta_k)$ is a density with parameters θ_k. Equivalent latent-variable representation:

$$z_i \sim \mathrm{Categorical}(\pi),$$

$$y_i \mid z_i = k \sim \phi(\cdot \mid \theta_k).$$

- 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

Finite Mixture Models

The finite mixture model density can be expressed as:

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

where $\pi_k \geq 0$, $\sum_{k=1}^{K} \pi_k = 1$, and $\phi(\cdot \mid \theta_k)$ is a density with parameters θ_k. Equivalent latent-variable representation:

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

$$y_i \mid z_i = k \sim \phi(\cdot \mid \theta_k).$$

- 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

Finite Mixture Models

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

Finite Mixture Models

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

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

Finite Mixture Models

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

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

  2. Generate y_i conditioned on z_i:

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

Finite Mixture Models

- Usually, we
  - set K, and
  - use the same density for all k.
- We can simulate data from the model as compound 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

Gaussian Mixture Models (GMM)

- The (finite) Gaussian mixture model:

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

where μ_k and Σ_k depend on the dimensionality of y_i.

Gaussian Mixture Models (GMM)

- The (finite) Gaussian mixture model:

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

where μ_k and Σ_k depend on the dimensionality of y_i.

GMM as Universal Approximators

Let $p(x)$ be any continuous probability density on $\mathbb{R}^d$ with compact support. For any $\varepsilon > 0$, there exists a Gaussian mixture model

$$p_K(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x \mid \mu_k, \sigma_k^2 \mathbf{I})$$

such that

$$\int |p(x) - p_K(x)|\, dx < \varepsilon.$$

Hence, Gaussian mixture models are **universal approximators** of probability densities.

For an indepth proof, see Nguyen et. al (2020). *Approximation by finite mixtures of continuous density functions that vanish at infinity* Cogent Mathematics and Statistics

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

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

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

Simulated data from a GMM

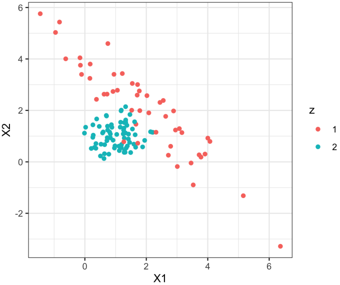


Figure: Simulated mixture data with the latent variable z.

Simulated data from a GMM

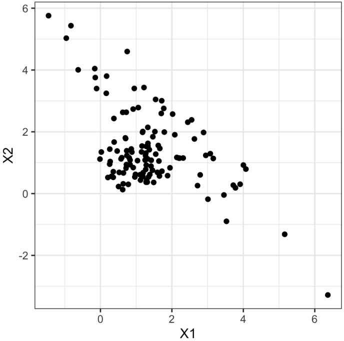


Figure: Simulated mixture data with the latent variable z.

Label switching

Finite mixture models are *not identifiable* up to label permutation.

The likelihood is invariant under relabeling of components:

$$(\pi_k, \theta_k)_{k=1}^K \equiv (\pi_{\sigma(k)}, \theta_{\sigma(k)})_{k=1}^K.$$

This is known as *label switching* and implies that cluster labels have no intrinsic meaning.

Mixtures of Multinomial distributions

What distribution (ϕ) should I use?

Mixtures of Multinomial distributions

What distribution (ϕ) should I use?

Depends on your data (y).

UPPSALA
UNIVERSITET

- Practicalities
- Introduction to unsupervised learning
  — Latent variables
- Clustering
  — k-means
- Mixture models
- Expectation-Maximization
- Probabilistic PCA

Mixtures of Multinomial distributions

What distribution (ϕ) should I use?

Depends on your data (y).

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

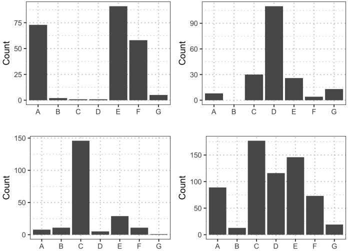


Figure: Mixture of Multinomials.

Estimating Mixture Models

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

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

- If we add a cluster indicators **z** it is simpler...

Estimating Mixture Models

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

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

- If we add a cluster indicators $\mathbf{z}$ it is simpler...
- Two approaches:
  - Gibbs sampler (Bayesian)

$$p(\mathbf{z}, \theta, \pi | \mathbf{y})$$

Estimating Mixture Models

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

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

- If we add a cluster indicators $\mathbf{z}$ it is simpler...
- Two approaches:
  - Gibbs sampler (Bayesian)

$$p(\mathbf{z}, \theta, \pi | \mathbf{y})$$

  - Expectation-Maximization (Frequentist)

Estimating Mixture Models

- 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**...

Estimating Mixture Models

- 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**...

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

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

The Expectation

- But, we dont know $\mathbf{z}$.

The Expectation

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

$$\gamma_{i,k}^{(t)} = \Pr(z_i = k \mid y_i, \theta^{(t)}) = \frac{\pi_k^{(t)}\, \phi(y_i \mid \theta_k^{(t)})}{\sum_{j=1}^{K} \pi_j^{(t)}\, \phi(y_i \mid \theta_j^{(t)})}$$

The Expectation

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

$$\gamma_{i,k}^{(t)} = \Pr(z_i = k \mid y_i, \theta^{(t)}) = \frac{\pi_k^{(t)}\,\phi(y_i \mid \theta_k^{(t)})}{\sum_{j=1}^{K} \pi_j^{(t)}\,\phi(y_i \mid \theta_j^{(t)})}$$

- γ_i can be seen as observation is weights or probability for each cluster

The Expectation

- But, we dont know $\mathbf{z}$.
- Although, we could compute the expected cluster assignment

$$\gamma_{i,k}^{(t)} = \Pr(z_i = k \mid y_i, \theta^{(t)}) = \frac{\pi_k^{(t)}\,\phi(y_i \mid \theta_k^{(t)})}{\sum_{j=1}^{K} \pi_j^{(t)}\,\phi(y_i \mid \theta_j^{(t)})}$$

- γ_i can be seen as observation *i*s weights or probability for each cluster
- γ_i is sometimes referred to as the responsibility.

The Maximization

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

The Maximization

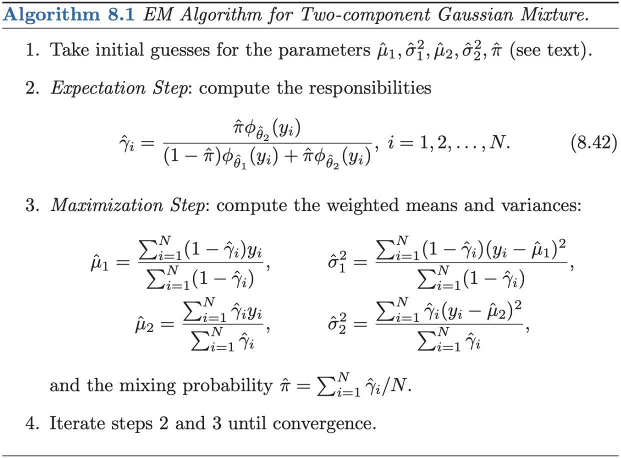
- Now, given γ we can (hopefully) easier maximize (π, θ).

$$\theta^{(t+1)} = \arg\max_{(} \pi, \theta) \, \mathbb{E}_{z|y,\pi^{(t)},\theta^{(t)}} \left[\log p(y, z \mid \theta, \pi)\right]$$

- We usually choose ϕ (the density) so the maximization
  - is a nice analytical expression.
  - end up with a weighted MLE.

Note! The E-step computes posterior probabilities of the latent variables. The M-step maximizes an expected log-likelihood using these probabilities as weights.

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)

The EM algorithm

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

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 straightforward using Jensen's inequality

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 straightforward using Jensen's inequality
  - We can interpret the final γ_i as the expected cluster
    Hence, the EM algorithm is a soft clustering approach.

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 straightforward using Jensen's 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.

Clarification

In EM, data augmentation refers to introducing latent variables (e.g., z_i) to make optimization easier. This is unrelated to "data augmentation" in deep learning, where new training examples are synthetically generated.

Connections to other approaches

- k-means can be derived as a limiting case of GMM: Gaussian mixture with equal, spherical covariances

  $$\Sigma_k = \sigma^2 I$$

  where $\sigma^2 \to 0$. Then EM assignments become hard clustering, recovering k-means.
- Hence, if we set $z_i = \text{argmax}(\gamma_i)$: k-means

Connections to other approaches

- k-means can be derived as a limiting case of GMM: Gaussian mixture with equal, spherical covariances

$$\Sigma_k = \sigma^2 I$$

  where $\sigma^2 \to 0$. Then EM assignments become hard clustering, recovering k-means.
- Hence, if we set $z_i = \text{argmax}(\gamma_i)$: k-means

Connections to other approaches

- k-means can be derived as a limiting case of GMM: Gaussian mixture with equal, spherical covariances

  $$\Sigma_k = \sigma^2 I$$

  where $\sigma^2 \to 0$. Then EM assignments become hard clustering, recovering k-means.
- Hence, if we set $z_i = \text{argmax}(\gamma_i)$: k-means
- Similarly, if we sample z_i according to γ: stochastic EM

Connections to other approaches

- k-means can be derived as a limiting case of GMM: Gaussian mixture with equal, spherical covariances

  $$\Sigma_k = \sigma^2 I$$

  where $\sigma^2 \to 0$. Then EM assignments become hard clustering, recovering k-means.
- Hence, if we set $z_i = \text{argmax}(\gamma_i)$: k-means
- Similarly, if we sample z_i according to γ: stochastic EM
- If we sample z_i conditional on θ: Gibbs sampling

UPPSALA UNIVERSITET

- Practicalities
- Introduction to unsupervised learning
  - Latent variables
- Clustering
  - k-means
- Mixture models
- Expectation-Maximization
- Probabilistic PCA

Section 6

Probabilistic PCA

Dimensionality reduction

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

Dimensionality reduction

- So far focus has been on (clustering) observations
- Now, we will address 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)

Dimensionality reduction

- So far focus has been on (clustering) observations
- Now, we will address 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.

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 $H \in \mathbb{R}^{n \times k}$, $W \in \mathbb{R}^{p \times k}$, $b \in \mathbb{R}^p$ and $X \in \mathbb{R}^{n \times p}$.
- H can be seen as latent factors

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 $H \in \mathbb{R}^{n \times k}$, $W \in \mathbb{R}^{p \times k}$, $b \in \mathbb{R}^p$ and $X \in \mathbb{R}^{n \times p}$.

- H can be seen as latent factors
- W can be seen as factor loadings

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 $H \in \mathbb{R}^{n \times k}$, $W \in \mathbb{R}^{p \times k}$, $b \in \mathbb{R}^p$ and $X \in \mathbb{R}^{n \times p}$.

- H can be seen as latent factors
- W can be seen as factor loadings
- We assume that W is orthogonal: $W^T W = I$

Principal Component Analysis

- The PCA model

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

- The loss function, also called reconstruction error:

$$J(b, W, H) = \sum_{i=1}^{n} \|x_i - (b + Wh_i)\|^2.$$

Principal Component Analysis

- The PCA model
$$X \approx b + WH^T ,$$
- The loss function, also called reconstruction error:
$$J(b, W, H) = \sum_{i=1}^{n} \|x_i - (b + Wh_i)\|^2.$$
- This can be minimized using Singular Value Decomposition

PCA: Conceptual depiction

$$\begin{bmatrix} \quad X \quad \\ (n \times p) \end{bmatrix} \approx \begin{bmatrix} W \\ (p \times k) \end{bmatrix} \times \begin{bmatrix} \quad H^T \quad \\ (k \times n) \end{bmatrix}$$

Figure: Conceptual depiction of PCA.

Probabilistic PCA (pPCA)

- PCA is not a probabilistic model

Probabilistic PCA (pPCA)

- PCA is not a probabilistic model
- Probabilistic PCA

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

  where $\epsilon \sim N(\mathbf{0}, \Psi)$
- In pPCA, we assume $\Psi = \sigma^2 \mathbf{I}$
- We also assume that $h_i \sim N(0, I)$

Probabilistic PCA (pPCA)

- PCA is not a probabilistic model
- Probabilistic PCA

$$x_i = b + W h_i^T + \epsilon_i$$

where $\epsilon \sim N(\mathbf{0}, \Psi)$

- In pPCA, we assume $\Psi = \sigma^2 \mathbf{I}$
- 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

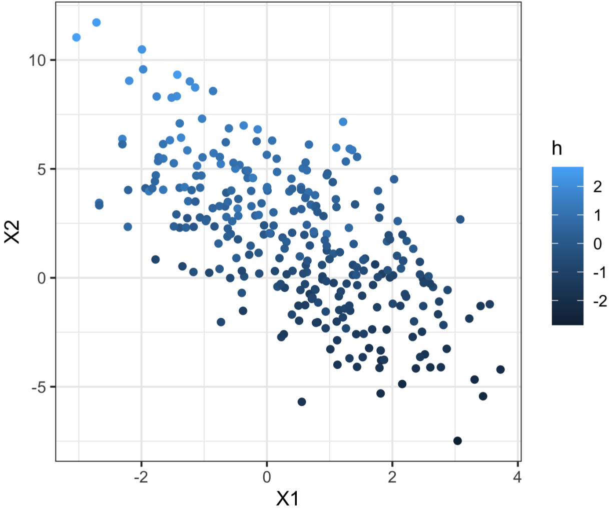


Figure: Data from a pPCA model with $W = (-1, 3)^T$, $b = (0.5, 2)$ and $\sigma^2 = 1$

Probabilistic PCA

- Probabilistic PCA

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

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)

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

Connections to PCA and Factor Analysis

- Probabilistic PCA

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

  where $\epsilon \sim N(\mathbf{0}, \Psi)$
- pPCA is closely connected to PCA and Factor Analysis:
  - As $\sigma^2 \rightarrow 0$, noise vanishes and the latent-variable model reduces to finding *directions of maximal variance*, i.e. PCA.
  - $\Psi = \text{diag}(\sigma_1, ..., \sigma_p, ..., \sigma_P)$: pPCA → Factor Analysis, where
    - H can be seen as latent factors
    - W can be seen as factor loadings

Latent-variable models (general form)

A latent-variable model assumes:

$$z_i \sim p(z), \qquad x_i \mid z_i \sim p(x \mid z_i).$$

Examples:

- Mixture models: $z_i =$ cluster
- PCA / pPCA: $z_i =$ latent factor
- HMMs / State-space models: $z_t =$ hidden state