Machine learning, big data and artificial intelligence – Block 7

Måns Magnusson
Department of Statistics, Uppsala University

HT 2021

This week's lectures

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

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

Practicalities

- Remember the project proposition deadline the 15th of December

- An additional guest lecture in January on fairness in AI and law (Holli Sargeant, Cambride University)

Assignment 4

- Not much

Assignment 5: Evaluation

• More variation

• More example code

• Would have liked to play around with BERT

• Still not superclear what is the weights in the RNN: Keras confusion

Recurrent Neural Networks

$$a_t = b + Wh_{t-1} + Ux_t$$
$$h_t = \sigma_1(a_t)$$
$$o_t = c + Vh_t$$
$$\hat{y}_t = \sigma_{\text{output}}(o_t) = \text{softmax}(o_t)$$

Think of h_t as the "state" at timepoint t

The embedding is X.

Supervised and Unsupervised learning

Figure: The Supervised Problem

Supervised and Unsupervised learning

- Previous assignments
- Introduction to unsupervised learning
  — Latent variables
- Clustering
- Mixture models
- Expectation Maximization
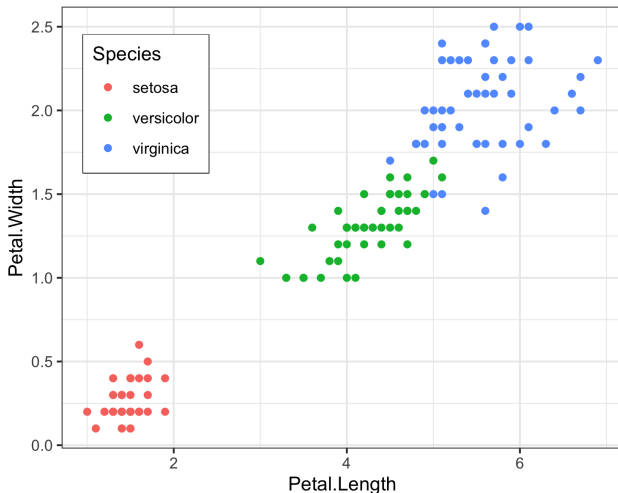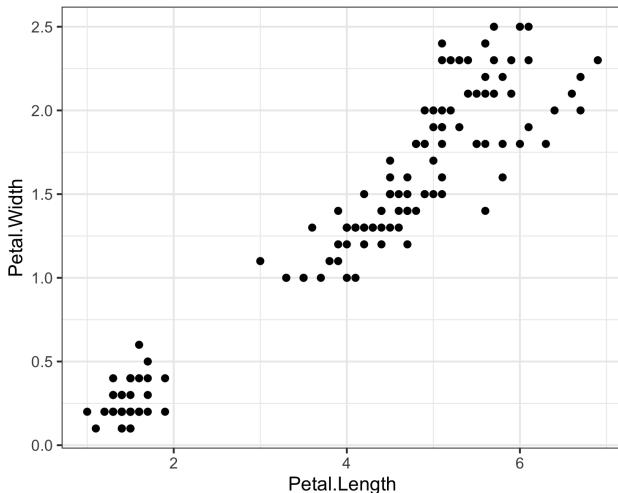- probabilistic PCA

Figure: The Unsupervised Problem

Supervised and Unsupervised learning

In supervised learning:

- We have *training* data

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

- 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

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

- 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

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

- 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 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 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
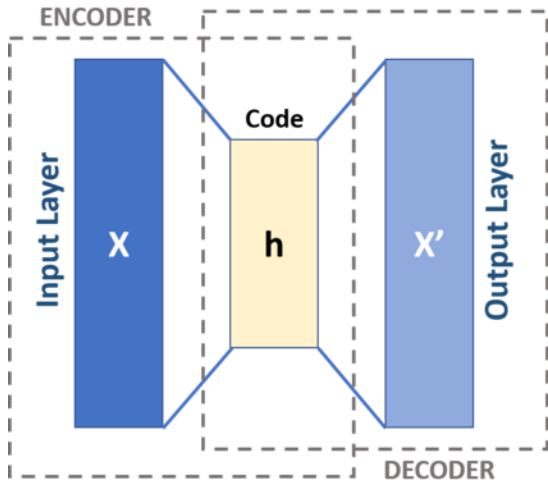
Figure: A Neural Autoencoder (Wikipedia)

Loss functions and evaluation

- Autoencoder uses the difference between the original and reconstructed output

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

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

Loss functions and evaluation

- Autoencoder uses the difference between the original and reconstructed output

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

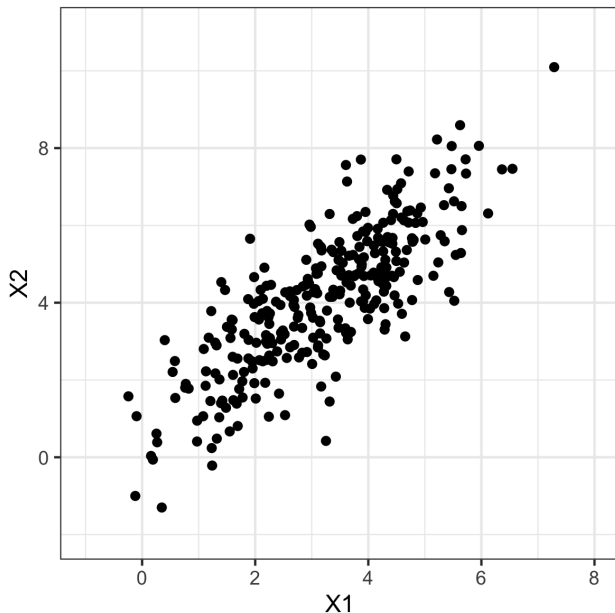 where $d(x|h)$ is the decoder and $e(h|x)$ is the encoder.

- 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

Example: Bivariate Gaussian model

Example: Bivariate Gaussian model

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

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

$$\hat{\mu} = [3.19, 4.11]$$
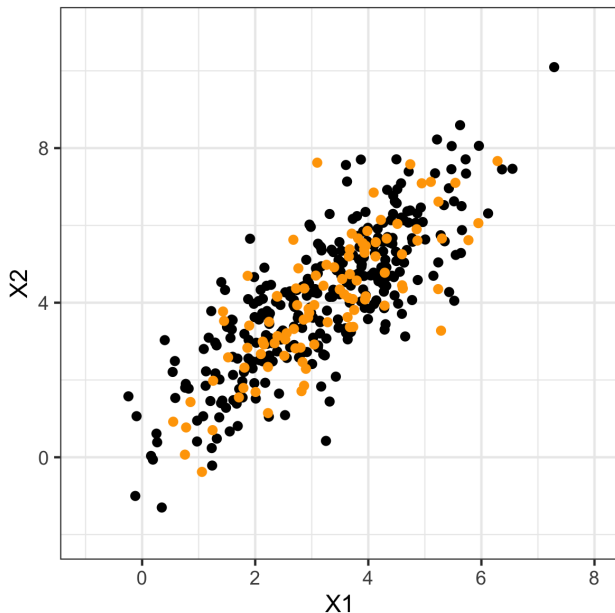
$$\hat{\Sigma} = \begin{bmatrix} 1.95 & 2.05 \\ 2.05 & 3.36 \end{bmatrix}$$

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

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 interest

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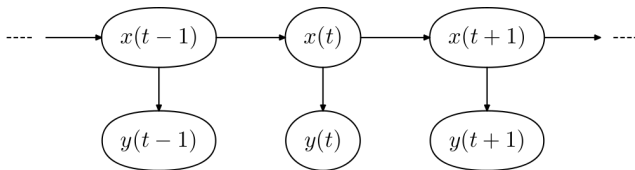
Figure: A Hidden Markov Model (Wikipedia). Note that x is unobserved and y is observed.

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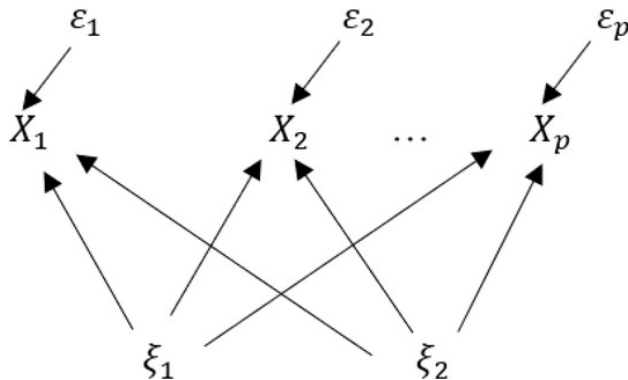
Figure: A Factor Analysis Model (Eshima, Tabata and Borroni, 2018, edited).

Clustering

- Separate observations x_i into groups or segments

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

- 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

- 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

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...

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 $m_1, ..., m_K$ vectors

- Loss function:

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

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 $m_1, ..., m_K$ vectors
- Loss function:

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

- 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 $m_1, ..., m_K$ vectors

- Loss function:

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

- Hyperparameter: K (the number of clusters)

- Parameters: $\mathbf{m}$ (a $K \times P$ matrix).

- A difficult problem: K^n possibilities

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 cluser 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
  - generalized loss
  - approaches to find a good K

k-means clustering

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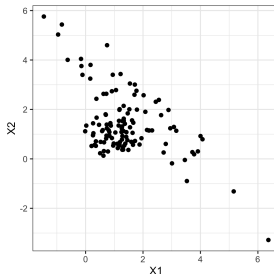
Problems with k-means

- Clusters might
    - overlap
    - have different forms

Figure: Two clusters with different shapes.

Problems with k-means

- Clusters might
  - overlap
  - have different forms
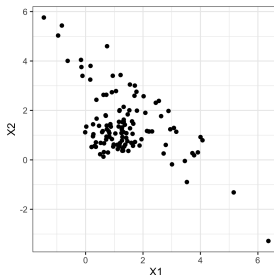
Figure: Two clusters with different shapes.

We can solve these problems using probabilistic models

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

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

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

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

- 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 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

- 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.

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

Example: Simulate data from a GMM

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

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
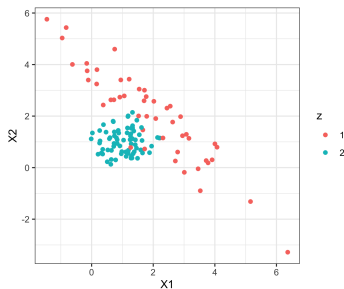
Figure: Simulated mixture data with the latent variable z.

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).

Mixtures of Multinomial distributions

What distribution (ϕ) should I use?

Depends on your data (y).

$$y_i = \sum_{k=1}^{K} \pi_k \text{Multinomial}(\mathbf{p}_k)$$
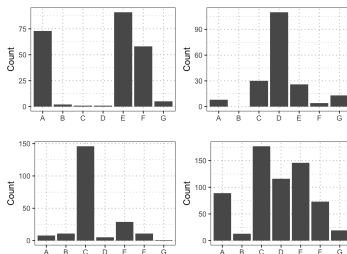
Figure: Mixture of Multinomials.

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**...

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**...

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

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

UPPSALA
UNIVERSITET

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

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.

The Maximization

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

The Maximization

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

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.

Example: EM for a Gaussian Mixture

UPPSALA
UNIVERSITET

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

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

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

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 straight-forward using Jensens 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 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.

Connections to other approaches

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

UPPSALA
UNIVERSITET

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

Dimensionality reduction

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

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)

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.

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: $W^T W = I$

Principal Component Analysis

- The PCA model
$$X \approx b + WH^T ,$$

- The loss function, also called reconstruction error:
$$J(b, W, H) = \sum_{i}^{N} ||x_i - b + Wh_i^T||^2$$

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

PCA: Conceptual depiction

UPPSALA
UNIVERSITET

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

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

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

$$x_i \sim N(b, W W^T + \Psi)$$
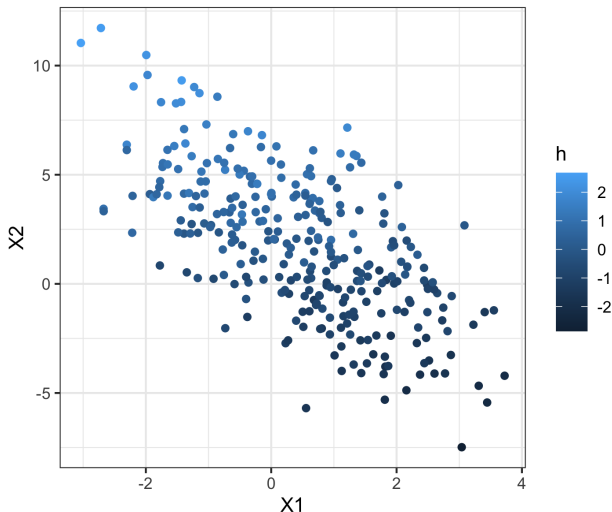
probabilistic PCA

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

probabilistic PCA

- probabilistic PCA

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

probabilistic PCA

- probabilistic PCA

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

- 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:
  - $\sigma I \rightarrow 0$: pPCA $\rightarrow$ PCA
  - $\Psi = \text{diag}(\sigma_1, ..., \sigma_p, ..., \sigma_P)$: pPCA $\rightarrow$ Factor Analysis