# Machine Learning
# Unsupervised Learning

dsai.asia

Asian Data Science and Artificial Intelligence Master's Program

Co-funded by the
Erasmus+ Programme
of the European Union

# Readings

Readings for these lecture notes:

- Bishop, C. (2006), *Pattern Recognition and Machine Learning*, Springer, Chapter 9.
- Hastie, T., Tibshirani, R., and Friedman, J. (2016), *Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, Chapters 13, 14.
- Ng, A. (2017), *Learning Theory*. Lecture note set 7 for CS229, Stanford University.

These notes contain material © Bishop (2006), Hastie et al. (2016), and Ng (2017).

# Outline

1. Introduction
2. $k$-means
3. EM for Gaussian mixture models
4. Other unsupervised learning algorithms

# Introduction

Now we consider the problem of unsupervised learning in which, rather than trying to predict some target given x, we want to understand the relationship of x to the examples in our unlabeled training set.

We begin with $k$-means clustering then move to other models.

# Outline

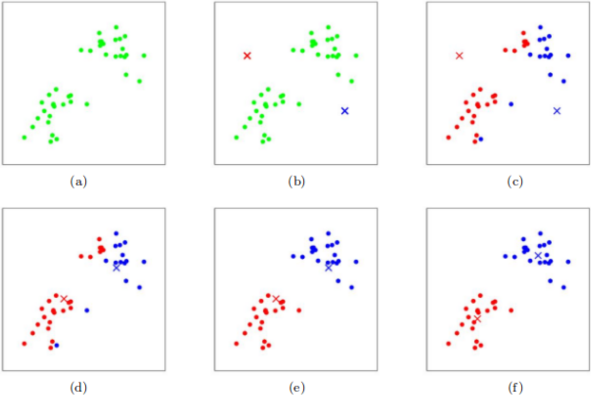
In a clustering problem, we are given a training set $S = \{\mathrm{x}^{(1)}, \ldots, \mathrm{x}^{(m)}\}$, $\mathrm{x}^{(i)} \in \mathbb{R}^n$ and we want to group the $\mathrm{x}^{(i)}$ into cohesive clusters.

The $k$-means clustering algorithm is as follows:

1. Randomly initialize $k$ cluster centroids $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k \in \mathbb{R}^n$.
2. Repeat until convergence:
    1. For $i \in 1..m$, $c^{(i)} \leftarrow \mathrm{argmin}_j \|\mathrm{x}^{(i)} - \boldsymbol{\mu}_j\|^2$.
    2. For $j \in 1..k$,

$$\boldsymbol{\mu}_j \leftarrow \frac{\sum_{i=1}^{m} \delta(c^{(i)} = j)\mathrm{x}^{(i)}}{\sum_{i=1}^{m} \delta(c^{(i)} = j)}.$$

# $k$-means
## Example



Ng (2017), CS229 lecture notes

(a) Training data. (b) Initial mean assignment. (c) Iteration 1 cluster assignment. (d) Iteration 1 mean computation. (e) Iteration 2 cluster assignment. (f) Iteration 2 mean computation.

As an exercise, generate data from three Gaussians with different means and covariances and produce an animation of $k$-means in action.

*k*-means is equivalent to coordinate descent on c and the $\boldsymbol{\mu}_i$'s for cost function or distortion function

$$J(\mathrm{c}, \Theta) = \sum_{i=1}^{m} \|\mathrm{x}^{(i)} - \boldsymbol{\mu}_{c^{(i)}}\|^2$$

with

$$\Theta = \begin{bmatrix} \boldsymbol{\mu}_1 & \cdots & \boldsymbol{\mu}_k \end{bmatrix}.$$

Prove this by finding the minimum with respect to c holding the $\boldsymbol{\mu}_i$ fixed then finding the minimum with respect to the $\boldsymbol{\mu}_i$ holding the c fixed.

The method is thus guaranteed to converge in the sense that on every iteration, $J$ will monotonically decrease.

As $J$ is not convex, *k*-means is not guaranteed to converge to a global minimum. Local minima are possible.

# Outline

# EM for Gaussian mixture models

## Setting

Similar to the $k$-means setting, suppose we are given a training set $S = \{\mathrm{x}^{(1)}, \ldots, \mathrm{x}^{(m)}\}$ without labels.

Rather than simply find cluster centers, however, now our goal is to estimate a joint distribution $p(\mathrm{x}^{(i)}, z^{(i)}) = p(\mathrm{x}^{(i)} \mid z^{(i)})p(z^{(i)})$.

The latent (hidden) variable $z^{(i)}$ indicates which of $k$ clusters or components the example $\mathrm{x}^{(i)}$ was drawn from.

Assumptions:

- $z^{(i)} \sim \text{Multinomial}(\phi)$ with $\phi_j \geq 0$ and $\sum_{j=1}^{k} \phi_j = 1$.
- $\mathrm{x}^{(i)} \mid (z^{(i)} = j) \sim \mathcal{N}(\boldsymbol{\mu}_j, \Sigma_j)$.

# EM for Gaussian mixture models

## Likelihood

As usual, we try to estimate the parameters $\phi, \boldsymbol{\mu}_1, \ldots, \Sigma_1, \ldots$ via maximum likelihood, first by writing down the (log) likelihood function:

$$
\begin{aligned}
\ell(\phi, \boldsymbol{\mu}_1, \ldots, \Sigma_1, \ldots) &= \sum_{i=1}^{m} \log p(\mathrm{x}^{(i)}; \phi, \boldsymbol{\mu}_1, \ldots, \Sigma_1, \ldots) \\
&= \sum_{i=1}^{m} \log \sum_{j=1}^{k} p(\mathrm{x}^{(i)} \mid z^{(i)} = j; \boldsymbol{\mu}_j, \Sigma_j) p(z^{(i)} = j; \phi) \\
&= \sum_{i=1}^{m} \log \sum_{j=1}^{k} \phi_j \mathcal{N}(\mathrm{x}^{(i)}; \boldsymbol{\mu}_j, \Sigma_j)
\end{aligned}
$$

There is no closed form solution to this maximization problem unfortunately...

But, what if we knew the $z^{(i)}$'s? Could we find the maximimum likelihood solution for the $\boldsymbol{\mu}$'s and $\Sigma$'s?

The answer is yes...

$$
\begin{aligned}
\ell(\phi, \boldsymbol{\mu}_1, \ldots, \Sigma_1, \ldots) &= \sum_{i=1}^{m} \left[ \log p(\mathrm{x}^{(i)} \mid \mathrm{z}^{(i)}; \boldsymbol{\mu}_{z^{(i)}}, \Sigma_{z^{(i)}}) + \log p(z^{(i)}; \phi) \right] \\
&= \sum_{i=1}^{m} \left[ \log \phi_{z^{(i)}} + \log \mathcal{N}(\mathrm{x}^{(i)}; \boldsymbol{\mu}_{z^{(i)}}, \Sigma_{z^{(i)}}) \right]
\end{aligned}
$$

Take the derivatives with respect to each parameter, set them to 0, and solve for the parameters. What do you get?

# EM for Gaussian mixture models

## Maximization with respect to $\phi_j$

If we set up the equation $\frac{\partial \ell}{\partial \phi_j} = 0$, we get close to a solution as $\phi_j \to \infty$.

This violates the constraint $\sum_{j=1}^{k} \phi_j = 1$.

Try instead, then, to introduce a Lagrange multiplier for the constraint and find a stationary point of

$$\ell(\phi, \boldsymbol{\mu}_1, \ldots, \Sigma_1, \ldots) + \lambda \left( \sum_{j=1}^{k} \phi_k - 1 \right).$$

Maximizing this with the full likelihood expression, you should be able to obtain

$$\sum_{i=1}^{m} \frac{\mathcal{N}(\mathrm{x}^{(i)}; \boldsymbol{\mu}_{z^{(i)}}, \Sigma_{z^{(i)}})}{\sum_{l=1}^{k} \phi_l \mathcal{N}(\mathrm{x}^{(i)}; \boldsymbol{\mu}_{z^{(i)}}, \Sigma_{z^{(i)}})} + \lambda = 0.$$

Here's bit of a magic trick: try multiplying both sides of the equation by $\phi_j$ then summing over $j$.

Can you get $\lambda = -m$?

Next, substitute $\lambda = -m$ and use the form of $\ell()$ assuming known $z^{(i)}$ and solve for $\phi_j$.

# EM for Gaussian mixture models

## Maximization with respect to $\boldsymbol{\mu}_j$ and $\Sigma_j$

Setting up the equation $\frac{\partial \ell}{\partial \boldsymbol{\mu}_j} = 0$ with the likelihood assuming known $z^{(i)}$, and using Equation 81[1] from *The Matrix Cookbook*, we get

$$\sum_{i=1}^{m} \delta(z^{(i)} = j) \Sigma_j^{-1} (\mathrm{x}^{(i)} - \boldsymbol{\mu}_j) = 0.$$

Doing the same with $\frac{\partial \ell}{\partial \Sigma_j} = 0$ and Equations 72[2] and 49[3] from the *Cookbook*, we get

$$\sum_{i=1}^{m} \delta(z^{(i)} = j) \Sigma_j^{-1} \left( (\mathrm{x}^{(i)} - \boldsymbol{\mu}_j)(\mathrm{x}^{(i)} - \boldsymbol{\mu}_j)^\top - \Sigma_j \right) \Sigma_j^{-1} = 0.$$

1 $\frac{\partial \mathrm{x}^\top \mathtt{B} \mathrm{x}}{\partial \mathrm{x}} = (\mathtt{B} + \mathtt{B}^\top)\mathrm{x}$

2 $\frac{\partial \mathrm{a}^\top \mathtt{X} \mathrm{a}}{\partial \mathtt{X}} = \mathrm{a}\mathrm{a}^\top$

3 $\frac{\partial |\mathtt{X}|}{\partial \mathtt{X}} = |\mathtt{X}|\mathtt{X}^{-T}$

Finally, we obtain

$$\begin{aligned}
\phi_j &= \frac{1}{m}\sum_{i=1}^{m}\delta(z^{(i)}=j), \\
\boldsymbol{\mu}_j &= \frac{\sum_{i=1}^{m}\delta(z^{(i)}=j)\mathrm{x}^{(i)}}{\sum_{i=1}^{m}\delta(z^{(i)}=j)}, \\
\Sigma_j &= \frac{\sum_{i=1}^{m}\delta(z^{(i)}=j)(\mathrm{x}^{(i)}-\boldsymbol{\mu}_j)(\mathrm{x}^{(i)}-\boldsymbol{\mu}_j)^\top}{\sum_{i=1}^{m}\delta(z^{(i)}=j)}.
\end{aligned}$$

If you look back, you'll see this is very similar to the Gaussian discriminant analysis model...

OK, that's great, but we don't know the $z^{(i)}$'s!!

To address this issue, we use the EM (Expectation Maximization) algorithm.

EM is a general approach to solving maximum likelihood problems when there are latent (unobserved/hidden) variables.

In the E step, we try to guess values of the hidden variables, then in the M step, we update the parameters of the model based on those guesses.

Here's the EM algorithm, made specific for our Gaussian mixtures problem:

1. Repeat until convergence:
    1. (E-step) For each $i \in 1..m, j \in 1..k$,

    $$w_j^{(i)} \leftarrow p(z^{(i)} = j \mid \mathrm{x}^{(i)}; \phi, \boldsymbol{\mu}_1, \ldots, \Sigma_1, \ldots)$$

    2. (M-step) Update the parameters

    $$\begin{aligned}
    \phi_j &\leftarrow \frac{1}{m} \sum_{i=1}^{m} w_j^{(i)}, \\
    \boldsymbol{\mu}_j &\leftarrow \frac{\sum_{i=1}^{m} w_j^{(i)} \mathrm{x}^{(i)}}{\sum_{i=1}^{m} w_j^{(i)}}, \\
    \Sigma_j &\leftarrow \frac{\sum_{i=1}^{m} w_j^{(i)} (\mathrm{x}^{(i)} - \boldsymbol{\mu}_j)(\mathrm{x}^{(i)} - \boldsymbol{\mu}_j)^\top}{\sum_{i=1}^{m} w_j^{(i)}}
    \end{aligned}$$

The $w_j^{(i)}$'s are our soft guesses as to the membership of each example $\mathrm{x}^{(i)}$ in component $j$ of the mixture.

How to calculate them? We have

$$w_j^{(i)} \leftarrow p(z^{(i)} = j \mid \mathrm{x}^{(i)}; \phi, \boldsymbol{\mu}_1, \ldots, \Sigma_1, \ldots).$$

Let's use Bayes rule to turn this into something we know how to calcluate:

$$p(z^{(i)} = j \mid \mathrm{x}^{(i)}; \phi, \boldsymbol{\mu}_1, \ldots, \Sigma_1, \ldots) = \frac{p(\mathrm{x}^{(i)} \mid z^{(i)} = j; \boldsymbol{\mu}_j, \Sigma_j) p(z^{(i)} = j; \phi)}{\sum_{l=1}^{k} p(\mathrm{x}^{(i)} \mid z^{(i)} = l; \boldsymbol{\mu}_l, \Sigma_l) p(z^{(i)} = l; \phi)}.$$

By the way, Bishop (2006) calls these $w_j^{(i)}$'s responsibilities, denoted $\gamma_{ij}$.

$\gamma_{ij}$ is the degree to which example $i$ is compatible with or could have been generated by Gaussian $j$.

You should compare the Gaussian mixture model to $k$-means.

Just like $k$-means, EM GMM is susceptible to local minima. We should perform several random restarts and retain the best model.

Exercise: implement the Gaussian mixture EM for a small dataset.

As mentioned earlier, this is a specialization of the general EM algorithm for estimating parameters when we have latent variables.

Next we'll look at the general EM algorithm and its guarantees on convergence.

Then we'll look at different forms of unsupervised learning algorithms.

# Outline

In this brief overview, we've seen $k$-means and GMMs.

What else is out there? Most unsupervised learning algorithms have one or more of these aims:

- Clustering, for purposes of coding, discretizing, or detecting anomalies.
- Probability density estimation, for purposes of positive detection, anomaly detection, or interpolation/synthesis of new data distributed similarly to the training set.
- Latent space discovery, for purposes of dimensionality reduction, positive detection, or interpolation/synthesis of new data distributed similarly to the training set.

We already saw how $k$-means and GMM can perform data clustering.

This style of clustering is sometimes called vector quantization. The goal is coding of inputs as members of a discrete set according to spatial locality.

A second major group of clustering algorithms are pairwise clustering methods that use pairwise similarity or distance (affinity) to group inputs. Two types:

- Hierarchical clustering is top-down.
- Agglomerative clustering is bottom-up.

# Other unsupervised learning algorithms
## Density estimation

We saw the GMM as an example of probability density estimation.

Density estimators can be used for detection (any input whose probability density is larger than some value is classified as a positive).

Density estimators can be used for anomaly detection (any input whose probability density is below some value is classified as anomalous).

The GMM is a parametric density estimator and is a universal approximator.

Non-parametric density estimators like kernel density estimation do not try to model the exact form of the probability density but instead estimate density directly based on sparsity/density of training data near the target input.

# Other unsupervised learning algorithms

## Latent space discovery

Latent space methods map the input space to a lower-dimensional representation, called a latent or semantic representation.

Principle components analysis (PCA) models the latent space as Gaussian distribution in the $k$-dimensional linear subspace of the original space in which the input data have the most variance.

Locally-linear embedding (LLE) and other nonlinear dimensionality reduction methods model the data as generated from a non-linear submanifold of the input space. LLE uses a piecewise linear model.

Latent Dirichlet allocation (LDA) is a topic model in NLP that imposes a prior on the data distribution in the latent space.

Generative adversarial networks (GANs) represent the mapping from the latent space to the data space by a neural network, usually a deconvolutional neural network. This generative model is trained alongside a discriminative adversary.

Unsupervised learning is an extremely rich area, deserving an entire course on its own!