Unsupervised Machine Learning

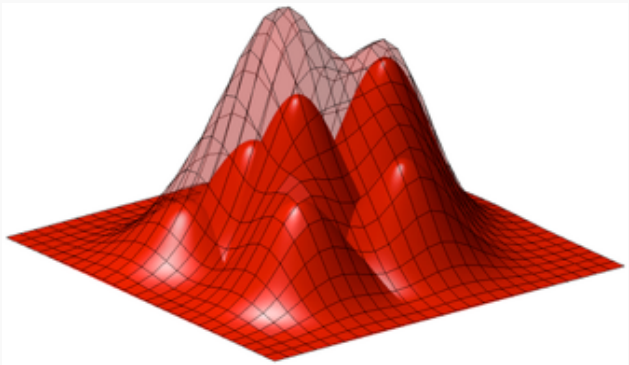
Unsupervised Machine Learning

Unsupervised machine learning deals with **unlabeled data**

$$\mathcal{D} = \{x_1, \ldots, x_N\}$$

Learning tasks aim to identify structure in the data, e.g.,

- groups (→ clustering)
- patterns (→ pattern mining)
- anomalies (→ anomaly detection)
- densities (→ density estimation)
- embeddings (→ autoencoders)
- ...



Unsupervised machiner learning shares many ideas, approaches, and algorithms with **data mining**

Clustering

Clustering

Clustering is one of the oldest problems in machine learning. The task is to identify groups in the data such that instances in a group are similar while instances in different groups are dissimilar.

Approaches thus, directly or indirectly, maximize the **intra-cluster** similarity or minimize the **inter-cluster** similarity, or a combination of both, e.g.,

$$\max \underbrace{\sum_{\substack{n,m=1 \\ n \neq m}}^{N} \mathbf{1}_{nm} sim(x_n, x_m)}_{\text{intra-cluster similarity}} \quad \text{or} \quad \min \underbrace{\sum_{\substack{n,m=1 \\ n \neq m}}^{N} (1 - \mathbf{1}_{nm}) sim(x_n, x_m)}_{\text{inter-cluster similarity}}$$

where $\mathbf{1}_{nm} = 1$ if x_n and x_m are in the same cluster and 0 otherwise.

Clustering is intrinsically an NP-hard problem! (why?)

K-Means

K-means (MacQueen, 1967; Forgy, 1965) is a simple clustering algorithm using Euclidean distances, though other distance measures (cosine, ...) can be incorporated as well.

Idea: Divide data into K disjoint partitions (clusters) and identify representatives (cluster centers) for each partition

Limitation: NP-hard

Given a set of N data points

$$\mathcal{D} = \{x_1, \ldots, x_N\} \quad \text{with} \quad x_n \in \mathbb{R}^D$$

and K centers

$$c_1, \ldots, c_K \quad \text{with} \quad c_k \in \mathbb{R}^D$$

the K-means optimization problem is to minimize the distance to the closest center for all $x_n \in \mathcal{D}$

$$\min_{c_1,\ldots,c_K} \quad \sum_{n=1}^{N} \min_{1 \le k \le K} (x_n - c_k)^2$$

Every point is only associated with a single center, which acts as a prototype or representative for all points in its vicinity.

The prototypes induce a **Voronoi partitioning** of the input space into K disjoint cells. Define candidate sets of points that are assigned to center k by

$$C_k = \{x : x \in \mathcal{D} \wedge c_k = \operatorname*{argmin}_{c \in \{c_1, \ldots, c_K\}} (x - c)^2\}$$

to rewrite the objective as

$$\min_{c_1, \ldots, c_K} \sum_{k=1}^{K} \sum_{x \in C_k} (x - c_k)^2.$$

Optimal center c_k^* is the mean of the associated points

$$c_k^* = \frac{1}{|C_k|} \sum_{x \in C_k} x$$

But: Assignments C_k may change when centers are changed!!

Hence, K-means is a non-convex optimization problem

Optimal clusters c_k^* can be computed using candidate sets C_k but the optimization changes the candidate sets

Need for an iterative algorithm that alternates between assignment and maximization phase

K-means Algorithm

Input: Instances $x_1, \ldots, x_N$

Initialize centers $c_1, \ldots, c_K$
repeat
    for $1 \leq k \leq K$:
        $C_k = \{x : x \in \mathcal{D} \wedge c_k = \operatorname{argmin}_{c \in \{c_1, \ldots, c_K\}} (x - c)^2\}$
    end
    for $1 \leq k \leq K$:
        $c_k = \frac{1}{|C_k|} \sum_{x \in C_k} x$
    end
until convergence

Output: Clustering $C_1, \ldots, C_K$ and centers $c_1^*, \ldots, c_K^*$

- alternate between assignment and maximization phase
- runtime in $\mathcal{O}(NK)$

K-means Algorithm

Input: Instances $x_1, \ldots, x_N$

Initialize centers $c_1, \ldots, c_K$
repeat
    for $1 \leq k \leq K$:
        $C_k = \{x : x \in \mathcal{D} \wedge c_k = \text{argmin}_{c \in \{c_1, \ldots, c_K\}} (x - c)^2\}$
    end
    for $1 \leq k \leq K$:
        $c_k = \frac{1}{|C_k|} \sum_{x \in C_k} x$
    end
until convergence

Output: Clustering $C_1, \ldots, C_K$ and centers $c_1^*, \ldots, c_K^*$

- alternate between assignment and maximization phase
- runtime in $\mathcal{O}(NK)$

K-means Algorithm

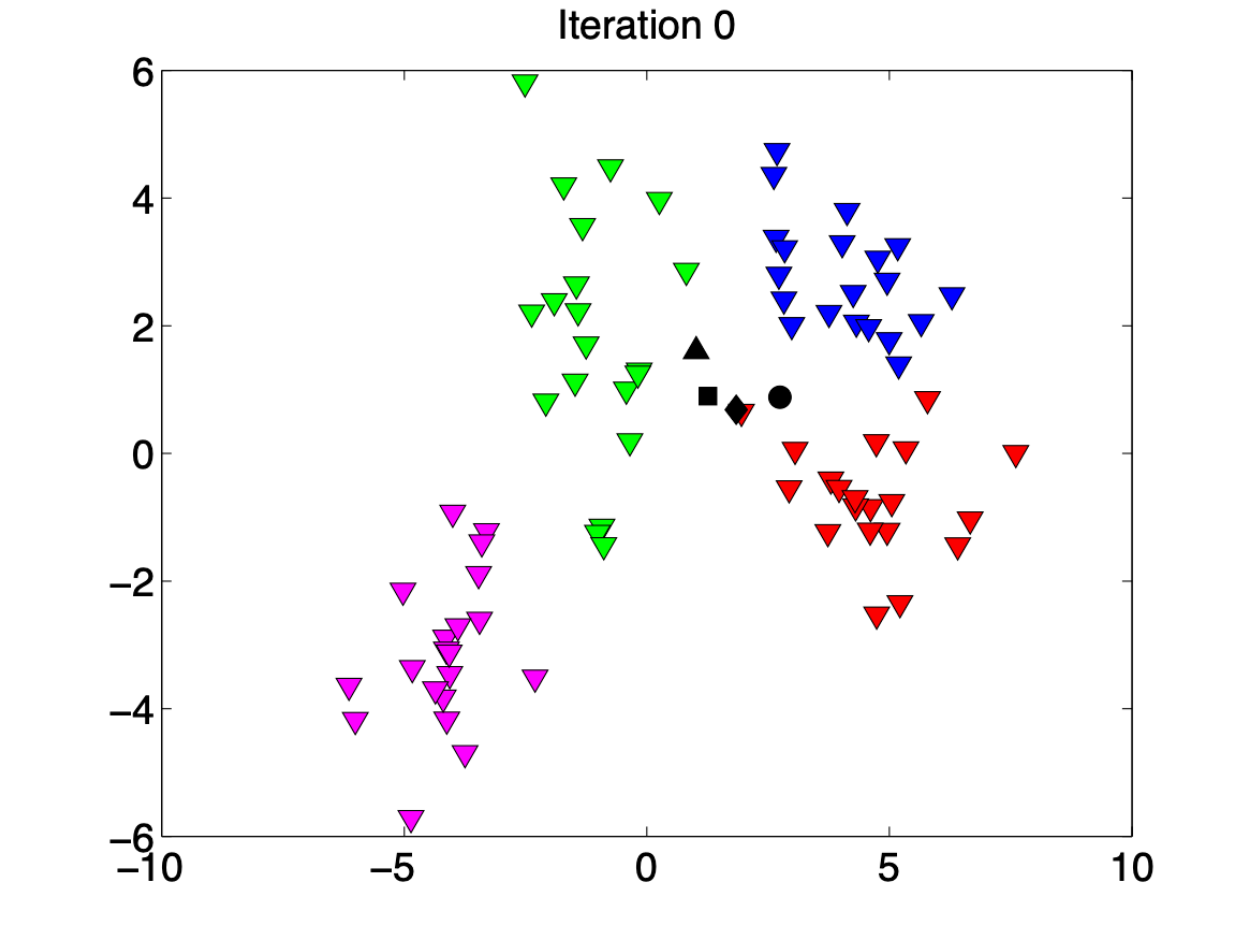
Input: Instances $x_1, \ldots, x_N$

Initialize centers $c_1, \ldots, c_K$
repeat
    for $1 \leq k \leq K$:
        $C_k = \{x : x \in \mathcal{D} \wedge c_k = \text{argmin}_{c \in \{c_1, \ldots, c_K\}} (x - c)^2\}$
    end
    for $1 \leq k \leq K$:
        $c_k = \frac{1}{|C_k|} \sum_{x \in C_k} x$
    end
until convergence

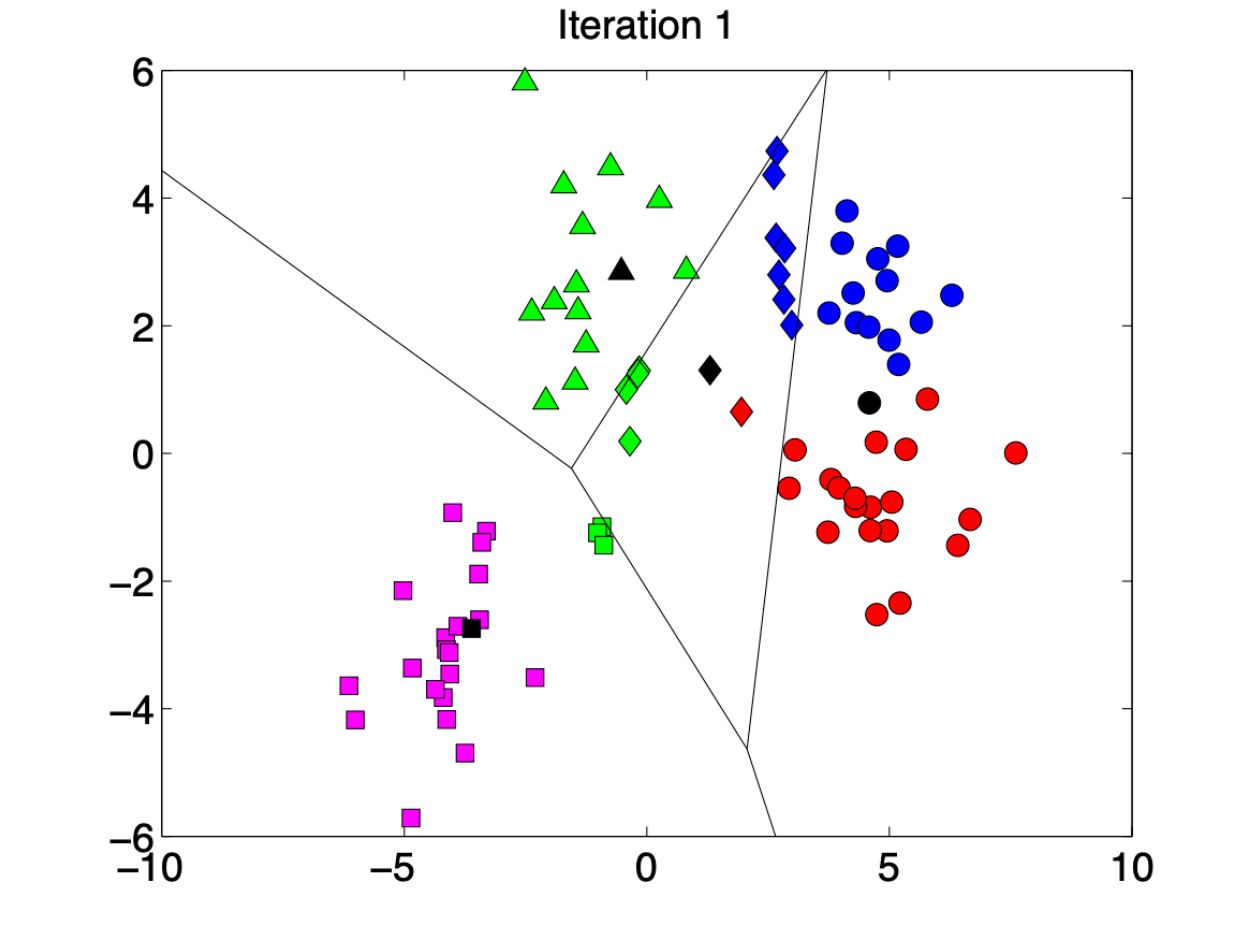
Output: Clustering $C_1, \ldots, C_K$ and centers $c_1^*, \ldots, c_K^*$

- alternate between assignment and maximization phase
- runtime in $\mathcal{O}(NK)$

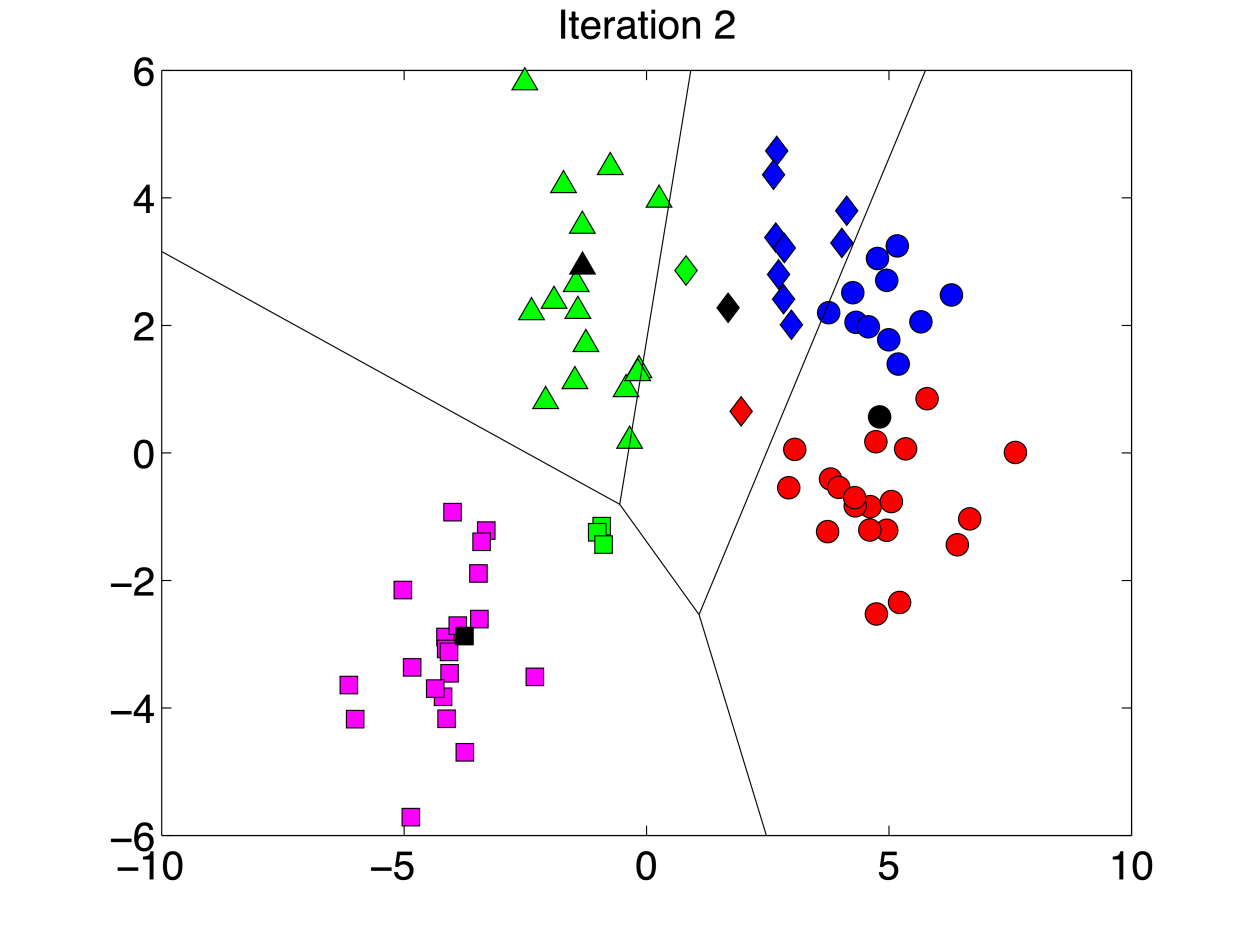
Example



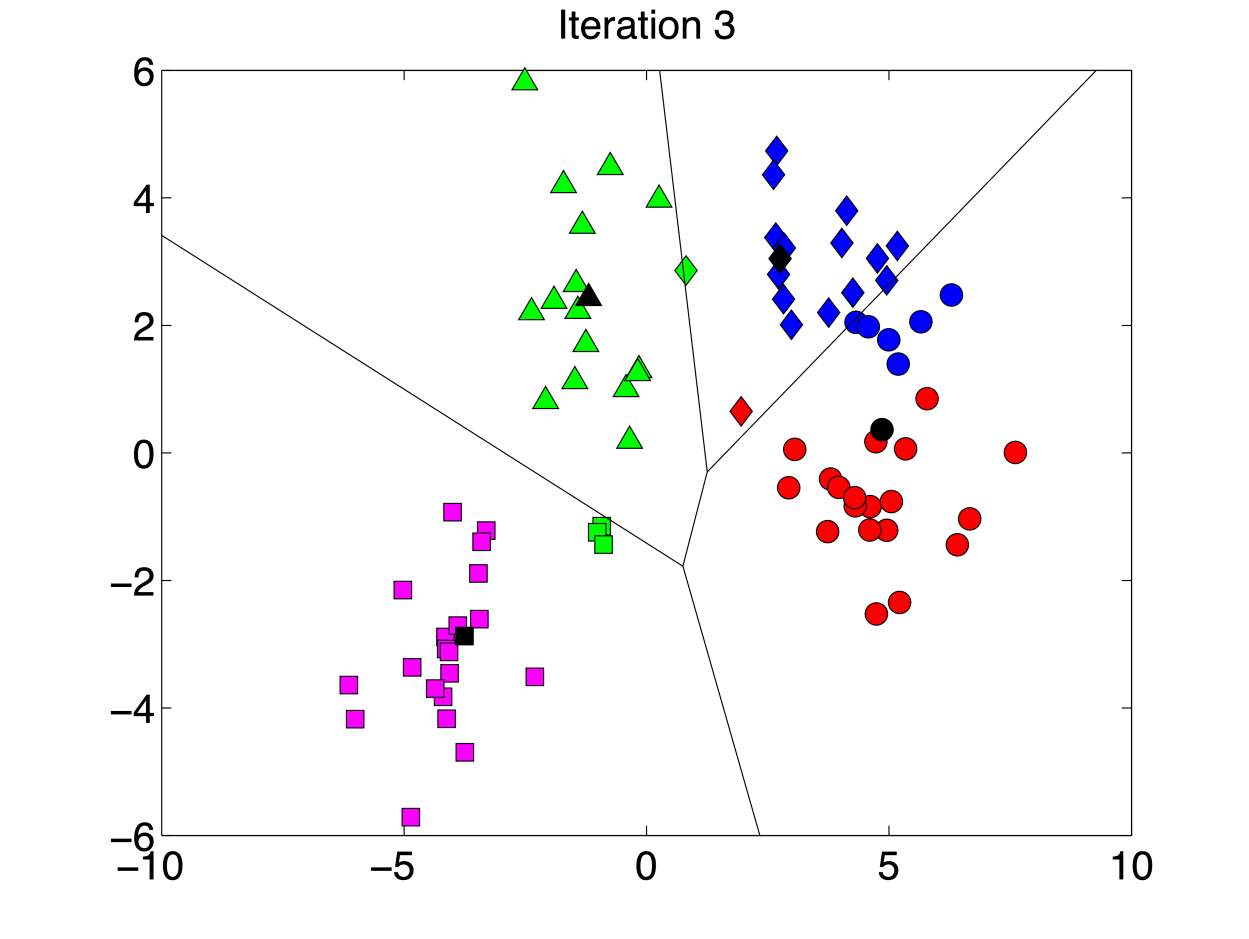
Example



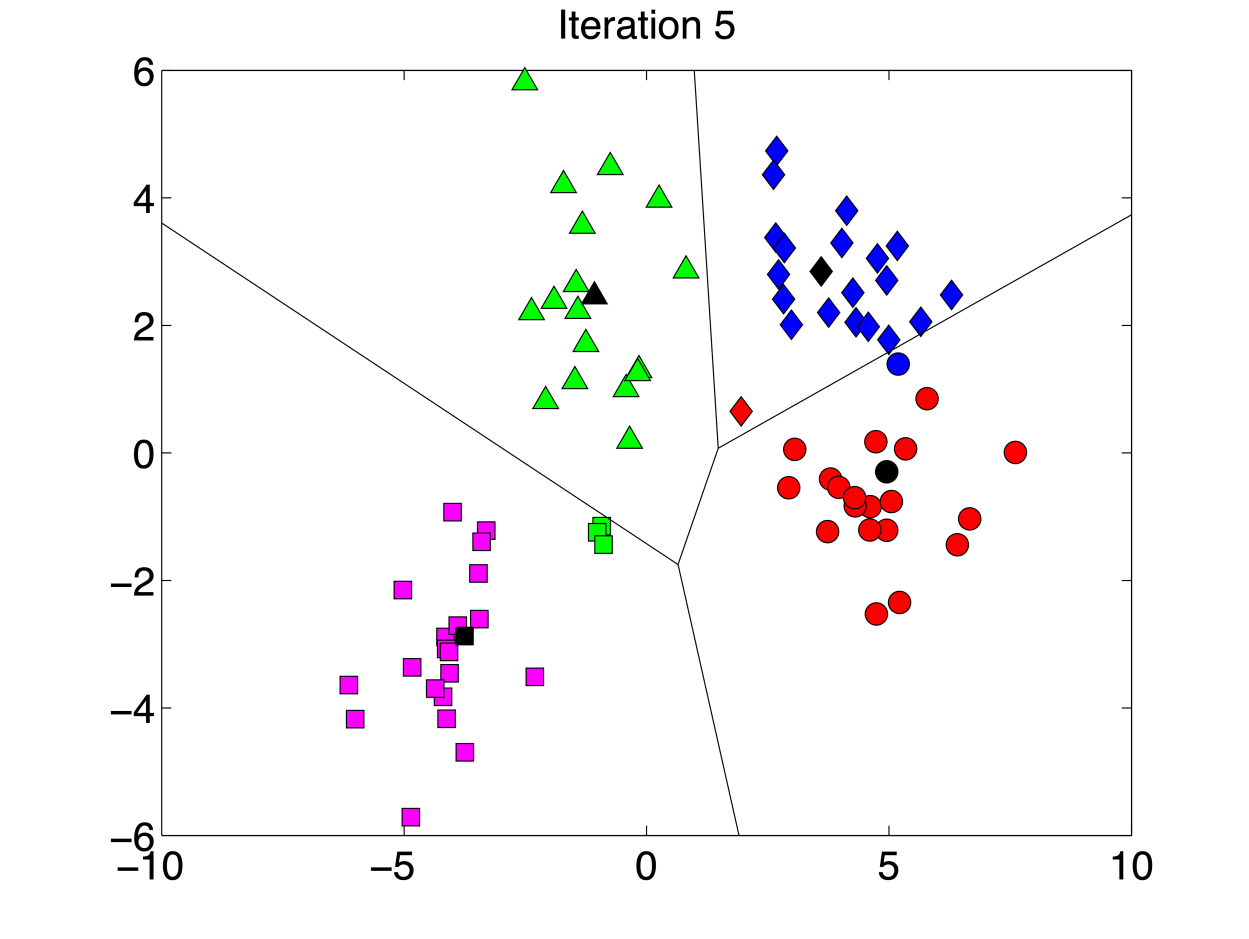
Example



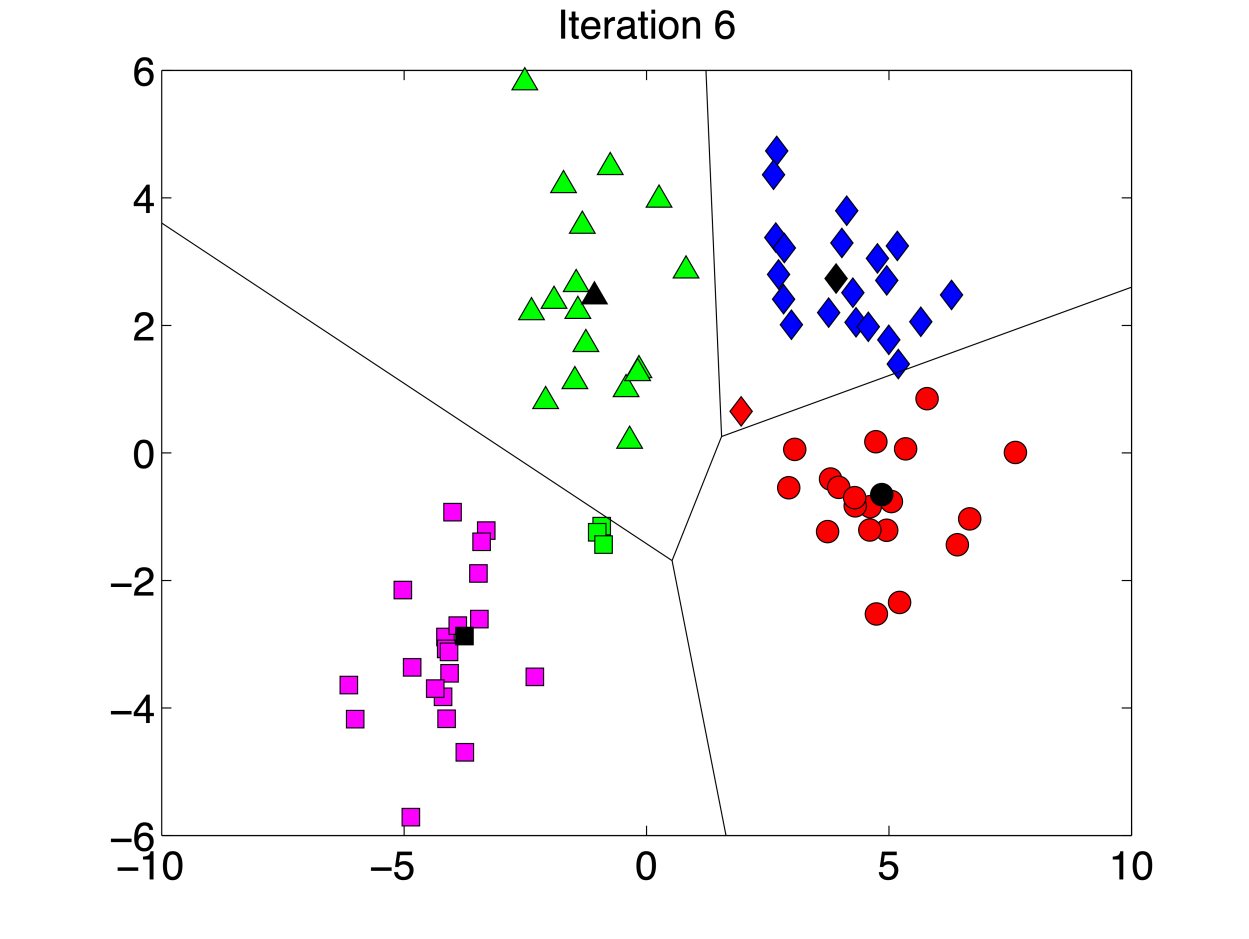
Example



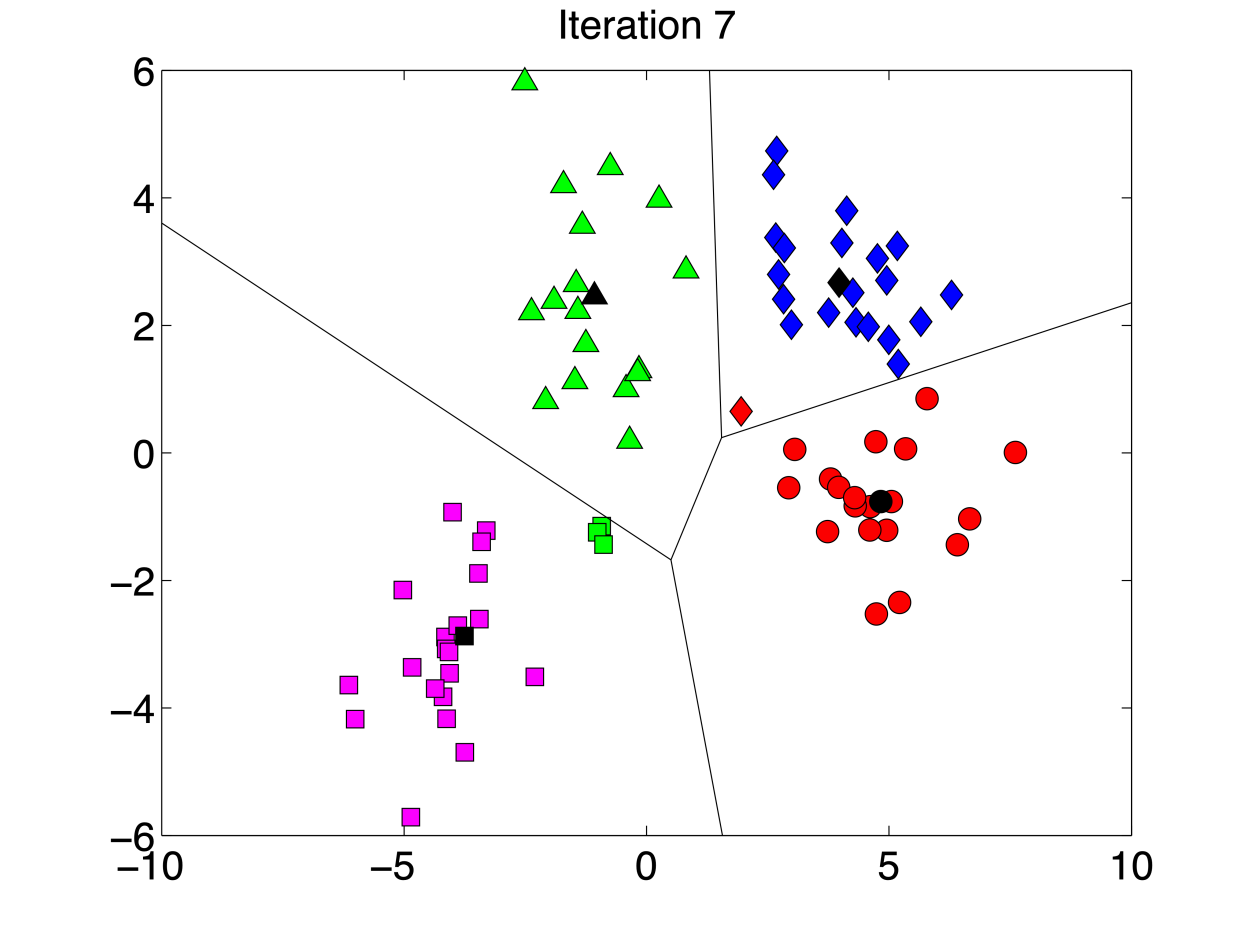
Example



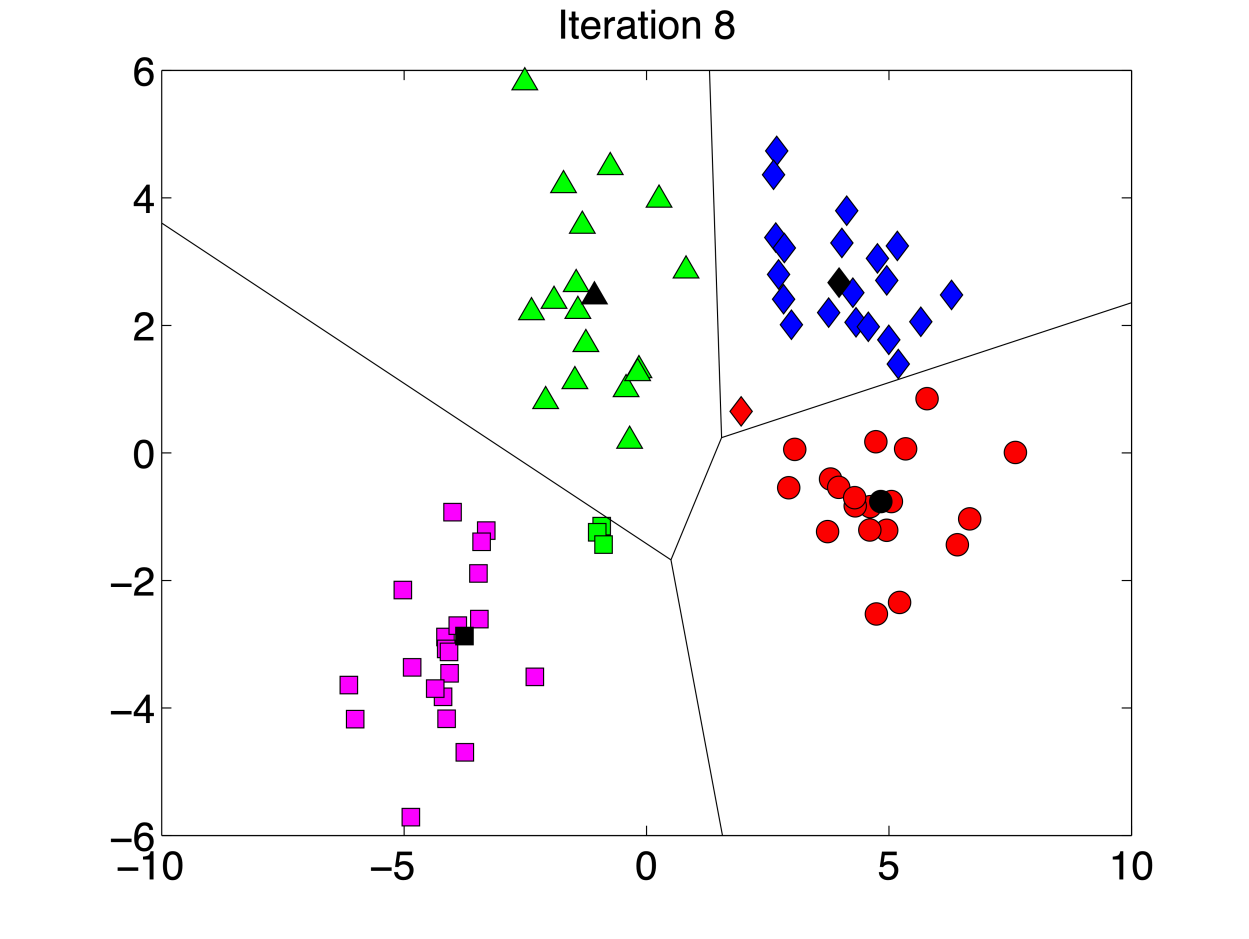
Example



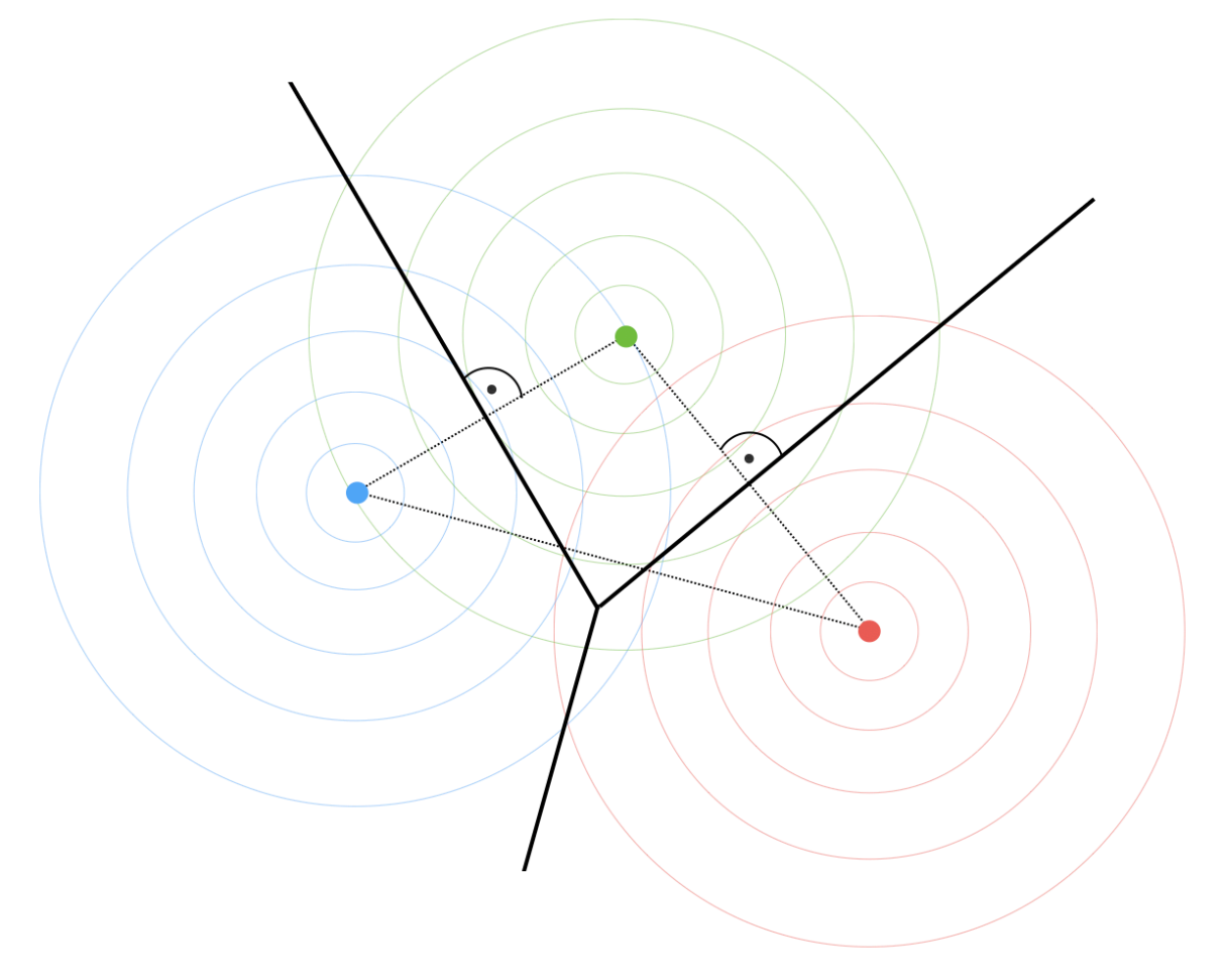
Example



Example

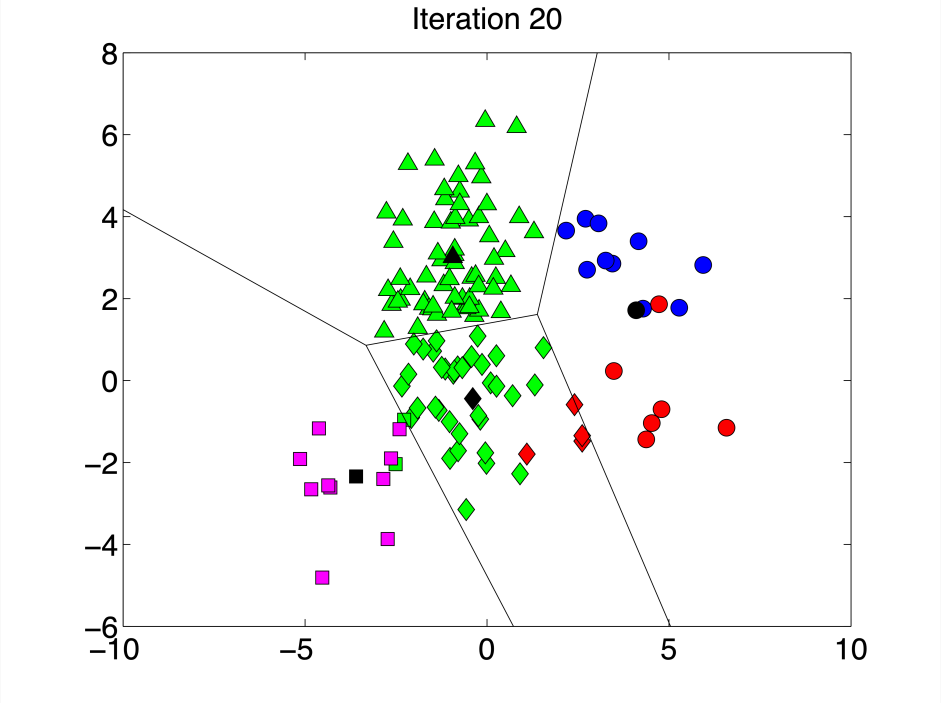


Euclidean distance and the hard assignment of points to centers induce piece-wise linear separations of the input space (Voronoi cells) favoring **non-overlapping globular/isotropic** clusters



Problems with K-means

- Greedy assignments
- Overlapping regions
- Different cluster sizes
- Non-globular shapes
- Curse of dimensionality
- How many clusters?
- How to initialize centers?



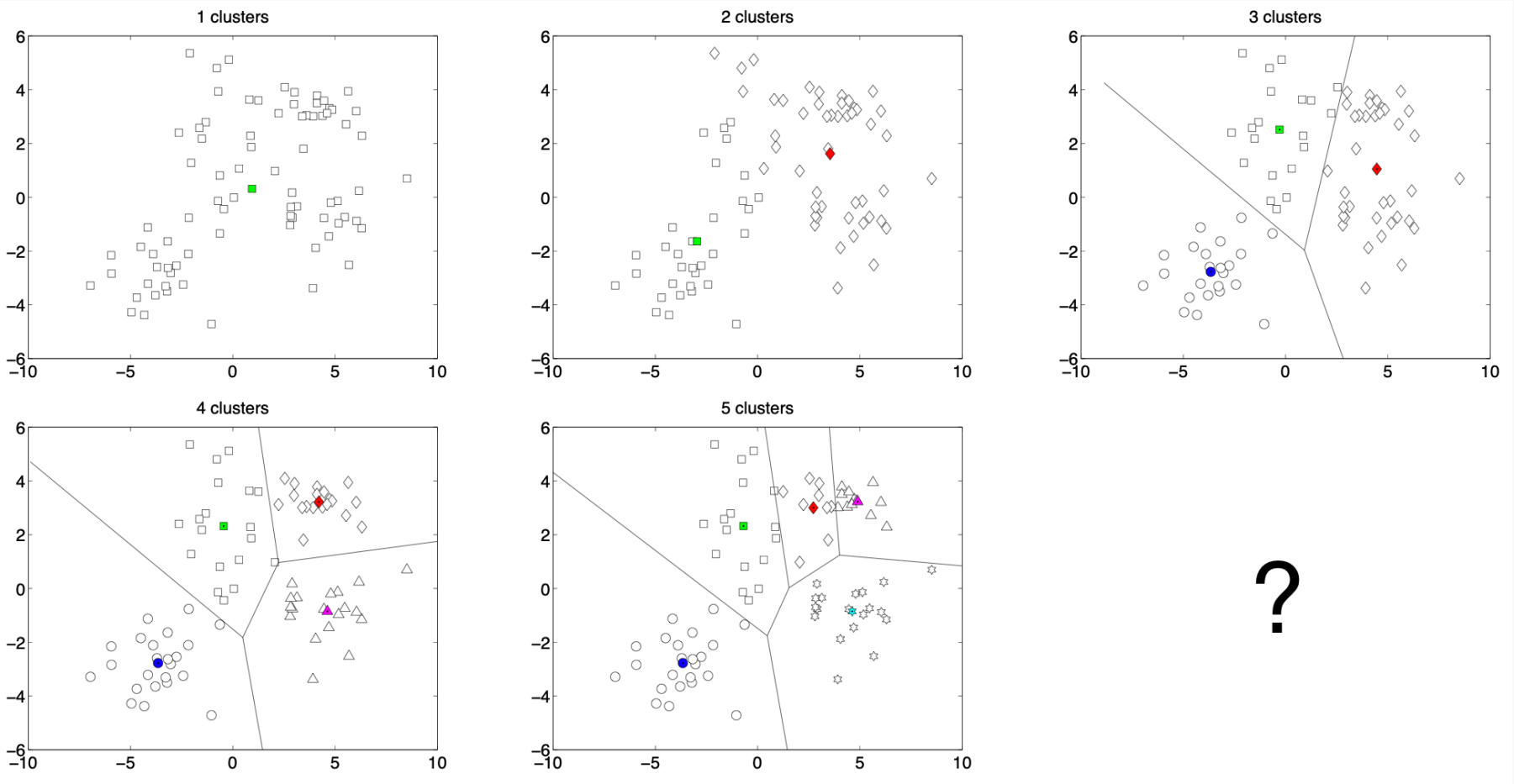
In the present form, K-means is also easily kernelizable (Dhillon et al., 2004). Substituting the optimal center (mean of points in cluster) into the objective yields

$$\sum_{k=1}^{K} \sum_{x \in C_k} (x - c_k)^2 = \sum_{k=1}^{K} \sum_{x \in C_k} \langle x, x \rangle - 2\langle x, c_k \rangle + \langle c_k, c_k \rangle$$

$$= \sum_{k=1}^{K} \sum_{x \in C_k} \left[\langle x, x \rangle - \frac{2}{|C_k|} \sum_{\bar{x} \in C_k} \langle x, \bar{x} \rangle + \frac{1}{|C_k|^2} \sum_{\bar{x}, \bar{\bar{x}} \in C_k} \langle \bar{x}, \bar{\bar{x}} \rangle) \right]$$

which now depends only on inner products of input examples and allows for the use of kernels

How to choose K?



How to choose k?

In general difficult, often choose k that minimises the Schwarz Criterion [] given by

$$\min_K obj(K) + \lambda DK \log N$$

where obj is the value of the objective function with D dimensions, K clusters, and N traninig instances; λ is a trade-off variable

Schwarz criterion is related to Bayesian Information Criterion (BIC)

An alternative is to learn K (Hamerly & Elkan, 2003)

How to initialize the centers?

So far, centers are randomly distributed around the mean of the data. Is there a better way?

How to initialize the centers?

So far, centers are randomly distributed around the mean of the data. Is there a better way?

Straight forward:

- pick a random point as first center
- pick farthest data point as second center
- pick farthest data point from the previous two as third center
- ...

However, this does not work in the presence of outliers (why?)

K-means++

The K-Means++ initialization (Arthur & Vassilvitskii, 2004) remedies this drawback and proceeds as follows:

- initialise the first center with a uniform random example
- draw the remaining centers from the set of input instances where each instance has probability proportional to

$$\min_{1 \leq k \leq K} (x - c_k)^2$$

Although the run-time is $\mathcal{O}(NK)$ (identical to a single K-means iteration!), it usually produces good initial clusterings

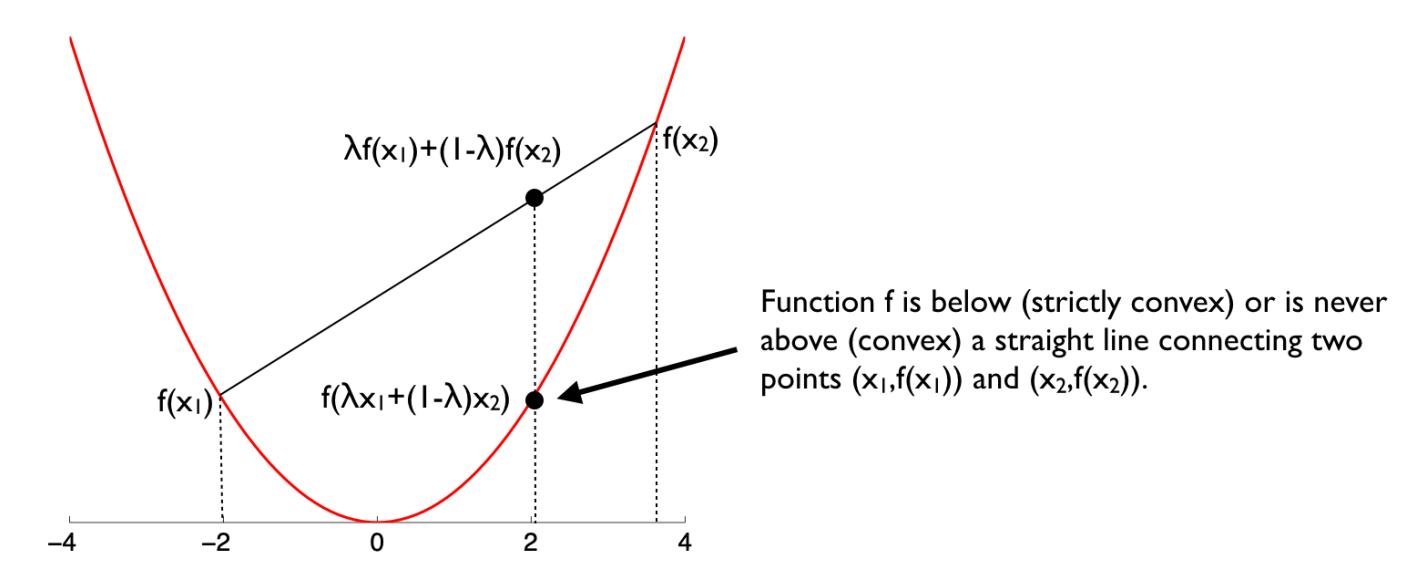
Alternatively, comparing the results from several repetitions of k-means with randomly chosen centers may help

Expectation Maximization

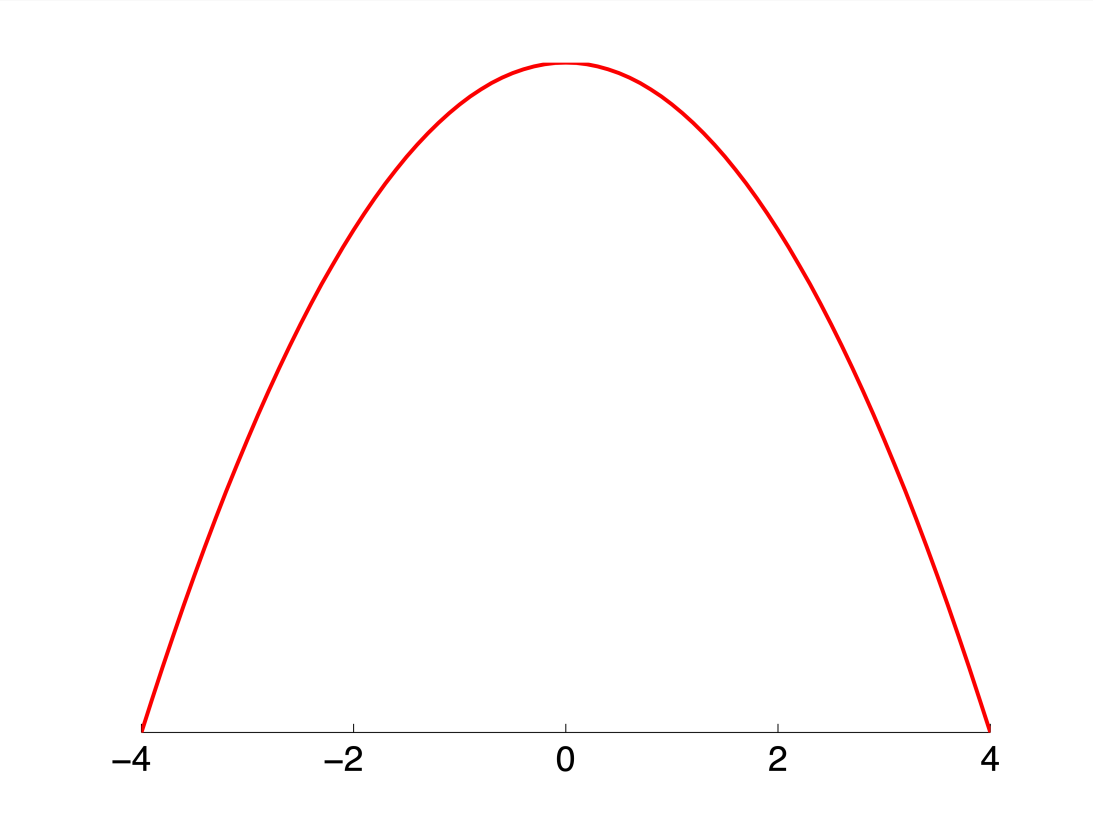
Definition (Convex function)
Let f be a real-valued function defined on an interval $\mathcal{I} = [a, b]$. f is said to be *convex* on $\mathcal{I}$ if $\forall x_1, x_2 \in \mathcal{I}, \lambda \in [0, 1]$,

$$f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda) f(x_2).$$

If the inequality is strict, f is said to be *strictly convex*.



Definition (Concave function)
A function f is *(strictly) concave* if $-f$ is (strictly) convex.



More things to recall...

Theorem
Every function $f(x)$ which is twice differentiable on $\mathcal{I} = [a, b]$ with $f''(x) \geq 0$ on $\mathcal{I}$ is convex on $\mathcal{I}$

We now show that $f(x) = -\log(x)$ is strictly convex on $\mathcal{I} = (0, \infty)$ by verifying that $\frac{\partial^2 - \log(x)}{\partial^2 x} = \frac{1}{x^2} > 0$ for $x \in (0, \infty)$. According to the above Theorem $-\log(x)$ is strictly convex on $(0, \infty)$ and, using the definition from the previous slide, $\log(x)$ is concave on $(0, \infty)$.

A very useful and often deployed inequality is Jensen's Inequality that naturally extends the notion of convexity to N points

Theorem (Jensen's inequality)
Let $f(x)$ be a convex function defined on an interval $\mathcal{I}$. Let $x_1, \ldots, N \in \mathcal{I}$ and $\lambda_1, \ldots, \lambda_N \geq 0$ with $\sum_{n=1}^{N} \lambda_n = 1$, then

$$f\left(\sum_{n=1}^{N} \lambda_n x_n\right) \leq \sum_{n=1}^{N} \lambda_n f(x_n)$$

Proof by induction: case $N = 1$ is trivial, case $N = 2$ is precisely the definition of convexity. Assume the theorem is true for some N, then...

$$
\begin{aligned}
f\left(\sum_{n=1}^{N+1} \lambda_n x_n\right) &= f\left(\lambda_{N+1} x_{N+1} + \sum_{n=1}^{N} \lambda_n f(x_n)\right) \\
&= f\left(\lambda_{N+1} x_{N+1} + \frac{1-\lambda_{n+1}}{1-\lambda_{N+1}} \sum_{n=1}^{n} \lambda_n f(x_n)\right) \\
&\leq \lambda_{N+1} f(x_{N+1}) + (1-\lambda_{N+1}) f\left(\frac{1}{1-\lambda_{N+1}} \sum_{n=1}^{N} \lambda_n x_n\right) \\
&= \lambda_{N+1} f(x_{N+1}) + (1-\lambda_{N+1}) f\left(\sum_{n=1}^{N} \frac{\lambda_i}{1-\lambda_{N+1}} x_i\right) \\
&\leq \lambda_{N+1} f(x_{N+1}) + (1-\lambda_{N+1}) \sum_{n=1}^{N} \frac{\lambda_i}{1-\lambda_{N+1}} f(x_n) \\
&\leq \lambda_{N+1} f(x_{N+1}) + \sum_{n=1}^{N} \lambda_n f(x_n) \\
&= \sum_{n=1}^{N+1} \lambda_n f(x_n)
\end{aligned}
$$

$\square$

Thus, since $\log(x)$ is concave, we may apply Jensen's inequality to obtain a very useful result,

$$\log\left(\sum_{n=1}^{N} \lambda_n x_n\right) \geq \sum_{n=1}^{N} \lambda_n \log(x_n)$$

gives us a lower-bound on the logarithm of a sum, which we will need later in the derivation of the Expectation Maximization algorithm

Maximum likelihood

In **maximum likelihood** estimation, we wish to estimate the model parameter(s) for which the observed data are the most likely.

Given a random variable X with i.i.d. realizations $x_1, \ldots, x_N$, we aim at finding parameters θ such that $p(X|\theta)$ is maximal. To estimate θ we introduce the likelihood function L given by

$$L(\theta|X) \equiv p(X|\theta) = \prod_{n=1}^{N} p(x_n|\theta)$$

The likelihood is a function of θ, given the data. Since $\log(x)$ is a strictly increasing function, we focus on the log-likelihood $\log L$. The goal is to find the parameters θ^* that maximize the log-likelihood

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \log L(\theta|X) = \sum_{n=1}^{N} \log p(x_i|\theta)$$

Example

Assume an N-sample drawn from a single but unknown one-dimensional Gaussian distribution $X \sim G(\mu, \sigma^2)$, such that

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

The likelihood function is given by

$$L(\mu, \sigma^2|X) = \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x_n-\mu)^2\right\}$$

The goal of maximum likelihood is to compute

$$\frac{\partial \log L}{\partial \mu} \stackrel{!}{=} 0 \quad \text{and} \quad \frac{\partial \log L}{\partial \sigma^2} \stackrel{!}{=} 0$$

to derive point estimators for the unknown parameters

Example (cont'd)

Computing the log likelihood:

$$\log L(\mu, \sigma^2|X) = \log\left\{\prod_{n=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x_n - \mu)^2\right\}\right\}$$

$$\log L(\mu, \sigma^2|X) = \sum_{n=1}^{N} \log\left\{\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x_n - \mu)^2\right\}\right\}$$

$$\log L(\mu, \sigma^2|X) = \sum_{n=1}^{N} -\frac{1}{2}\log 2\pi\sigma^2 + \log\left\{\exp\left\{-\frac{1}{2\sigma^2}(x_n - \mu)^2\right\}\right\}$$

$$\log L(\mu, \sigma^2|X) = -\frac{N}{2}\log 2\pi\sigma^2 + \sum_{n=1}^{N} \log\left\{\exp\left\{-\frac{1}{2\sigma^2}(x_n - \mu)^2\right\}\right\}$$

$$\log L(\mu, \sigma^2|X) = -\frac{N}{2}\log 2\pi\sigma^2 + \sum_{n=1}^{N} \left\{-\frac{1}{2\sigma^2}(x_n - \mu)^2\right\}$$

$$\log L(\mu, \sigma^2|X) = -\frac{N}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n - \mu)^2$$

Example (cont'd)

Computing the derivatives and setting to zero:

$$\frac{\partial \log L(\mu, \sigma^2|X)}{\partial \mu} = \frac{\partial}{\partial \mu}\left\{-\frac{N}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2\right\}$$
$$= \frac{\partial}{\partial \mu}\left\{-\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2\right\}$$

and thus

$$0 \stackrel{!}{=} \frac{\partial}{\partial \mu}\left\{-\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2\right\} = \frac{\partial}{\partial \mu}\left\{\sum_{n=1}^{N}(x_n-\mu)^2\right\}$$

Note the relation of ML and ERM: we want to minimize a sum of squared errors, the objective is quadratic in μ, hence, there is a unique optimum given by

$$0 = \sum_{n=1}^{N} 2(x_n-\mu) = \sum_{n=1}^{N}(x_n-\mu) = -N\mu + \sum_{n=1}^{N} x_n \quad \Rightarrow \quad \mu = \frac{1}{N}\sum_{n=1}^{N} x_n$$

Example (cont'd)

Conducting the corresponding derivation also for σ^2, maximum likelihood recovers the standard estimators for mean and variance of Gaussian distributions

$$\hat{\mu} = \frac{1}{N}\sum_{n=1}^{N} x_n \qquad \hat{\sigma}^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n - \hat{\mu})^2$$

Note that in pracice you may want to normalize the variance by $N-1$ instead of N

Limitations of ML

Maximum likelihood is intractable in the presence of missing values/observations in the data which render the likelihood function incomplete.

Differentiate between **observed variables** that can be directly measured and **latent (hidden) variables** that influence data but are not directly observable

Example: There is a fire alarm. Infer whether there is actually a fire or whether the smoke detector has a malfunction

Observed vs hidden variables

Example: There is a fire alarm. Infer whether there is actually a fire or whether the smoke detector has a malfunction

Examples for observed variables:

- the waveform values of a speech recording
- is it raining today?
- did the smoke alarm go off?

Examples for hidden variables:

- the phonemes that produce a given speech recording
- $P(\text{rain today}|\text{rain yesterday})$
- is the smoke alarm malfunctioning?

Expectation Maximization

Expectation Maximization (EM) (Dempster et al., 1977) is an efficient iterative procedure to compute the maximum likelihood estimates in the presence of missing or hidden data.

Each iteration of the EM algorithm consists of two processes:

- The **E-step** estimates missing data given the observed data and current estimate of the model parameters
- The **M-step** maximizes the likelihood function under the assumption that the missing data are known.

Convergence (to a local optimum) is assured since the algorithm is guaranteed to increase the likelihood at each iteration.

Assume the observed data X is incomplete, and the missing parts are denoted by Z so that

$$p(x, z|\theta) = p(z|x, \theta)p(x|\theta)$$

Accordingly, we differentiate the **complete-data likelihood** given by

$$L(\theta|X, Z) = p(X, Z|\theta)$$

and the **incomplete-data likelihood** denoted by

$$L(\theta|X)$$

Due to the incompleteness of the liklihood, a direct analytical solution is infeasible.

Instead, we resort to an iterative procedure that computes incremental parameter updates $\theta_0, \theta_1, \theta_2, \ldots, \theta^*$ so that the final θ^* maximizes $L(\theta|X)$.

Consider the situation at time $t+1$ with the current paramter estimation θ_t. We aim to find better parameters θ such that

$$L(\theta|X) > L(\theta_t|X)$$

Equivalently, we use the log likelihood and maximize the difference

$$\log L(\theta|X) - \log L(\theta_t|X) = \log \sum_{n=1}^{N} p(x_n|\theta) - \log \sum_{n=1}^{N} p(x_n|\theta_t)$$

Idea: incorporate hidden variables and marginalise them out:

$$p(X|\theta) = \sum_z p(X, Z = z|\theta) = \sum_z p(X|z, \theta)p(z|\theta)$$

The difference becomes

$$\log L(\theta|X) - \log L(\theta_t|X) = \log \left(\sum_z p(X|z, \theta)p(z|\theta) \right) - \log p(X|\theta_t)$$

Idea: incorporate hidden variables and marginalise them out:

$$p(X|\theta) = \sum_z p(X, Z = z|\theta) = \sum_z p(X|z, \theta)p(z|\theta)$$

The difference becomes

$$\log L(\theta|X) - \log L(\theta_t|X) = \log\left(\sum_z p(X|z, \theta)p(z|\theta)\right) - \log p(X|\theta_t)$$

(fixed from previous iteration)

Idea: incorporate hidden variables and marginalise them out:

$$p(X|\theta) = \sum_z p(X, Z = z|\theta) = \sum_z p(X|z, \theta)p(z|\theta)$$

The difference becomes

$$\log L(\theta|X) - \log L(\theta_t|X) = \log\left(\sum_z p(X|z, \theta)p(z|\theta)\right) - \log p(X|\theta_t)$$

Jensen's inequality guarantees that

$$\log \sum_{i=1}^{n} \lambda_i x_i \geq \sum_{i=1}^{n} \lambda_i \log(x_i)$$

for $\lambda_i > 0$ and $\sum_{i=1}^{n} \lambda_i = 1$. Identify $p(z|X, \theta_t)$ with λ.

Putting things together:

$$
\begin{aligned}
&\log\left(\sum_z p(X|z,\theta)p(z|\theta)\right) - \log p(X|\theta_t) \\
&= \log\left(\sum_z p(X|z,\theta)p(z|\theta)\frac{p(z|X,\theta_t)}{p(z|X,\theta_t)}\right) - \log p(X|\theta_t) \\
&= \log\left(\sum_z p(z|X,\theta_t)\frac{p(X|z,\theta)p(z|\theta)}{p(z|X,\theta_t)}\right) - \log p(X|\theta_t) \\
&\geq \sum_z p(z|X,\theta_t)\log\left(\frac{p(X|z,\theta)p(z|\theta)}{p(z|X,\theta_t)}\right) - \log p(X|\theta_t)
\end{aligned}
$$

This is Jensen's inequality!

Putting things together:

$$
\begin{aligned}
&\log\left(\sum_z p(X|z,\theta)p(z|\theta)\right) - \log p(X|\theta_t) \\
&= \log\left(\sum_z p(X|z,\theta)p(z|\theta)\frac{p(z|X,\theta_t)}{p(z|X,\theta_t)}\right) - \log p(X|\theta_t) \\
&= \log\left(\sum_z p(z|X,\theta_t)\frac{p(X|z,\theta)p(z|\theta)}{p(z|X,\theta_t)}\right) - \log p(X|\theta_t) \\
&\geq \sum_z p(z|X,\theta_t)\log\left(\frac{p(X|z,\theta)p(z|\theta)}{p(z|X,\theta_t)}\right) - \log p(X|\theta_t) \\
&= \sum_z p(z|X,\theta_t)\log\left(\frac{p(X|z,\theta)p(z|\theta)}{p(z|X,\theta_t)p(X|\theta_t)}\right)
\end{aligned}
$$

Use $\sum_z p(z|X,\theta_t) = 1$ so that $p(X|\theta_t) = \sum_z p(z|X,\theta_t)\log p(X|\theta_t)$ and $\log p(X|\theta_t)$ could be brought into the sum.

Putting things together:

$$
\begin{aligned}
&\log\left(\sum_z p(X|z,\theta)p(z|\theta)\right) - \log p(X|\theta_t) \\
&= \log\left(\sum_z p(X|z,\theta)p(z|\theta)\frac{p(z|X,\theta_t)}{p(z|X,\theta_t)}\right) - \log p(X|\theta_t) \\
&= \log\left(\sum_z p(z|X,\theta_t)\frac{p(X|z,\theta)p(z|\theta)}{p(z|X,\theta_t)}\right) - \log p(X|\theta_t) \\
&\geq \sum_z p(z|X,\theta_t)\log\left(\frac{p(X|z,\theta)p(z|\theta)}{p(z|X,\theta_t)}\right) - \log p(X|\theta_t) \\
&= \sum_z p(z|X,\theta_t)\log\left(\frac{p(X|z,\theta)p(z|\theta)}{p(z|X,\theta_t)p(X|\theta_t)}\right) \\
&\equiv \Delta(\theta|\theta_t)
\end{aligned}
$$

We now have

$$\log L(\theta|X) \geq \log L(\theta_t|X) + \Delta(\theta|\theta_t)$$

and define

$$Q(\theta|\theta_t) \equiv \log L(\theta_t|X) + \Delta(\theta|\theta_t)$$

so that we may write

$$\log L(\theta|X) \geq Q(\theta|\theta_t)$$

The function $Q(\theta|\theta_t)$ is bounded from above by the log likelihood $L(\theta|X)$. Additionally, observe that

$$\begin{aligned}
Q(\theta_t|\theta_t) &= \log L(\theta_t|X) + \Delta(\theta_t|\theta_t) \\
&= \log L(\theta_t|X) + \sum_z p(z|X,\theta_t) \log \left(\frac{p(X|z,\theta_t)p(z|\theta_t)}{p(z|X,\theta_t)p(X|\theta_t)} \right) \\
&= \log L(\theta_t|X) + \sum_z p(z|X,\theta_t) \log \left(\frac{p(X,z|\theta_t)}{p(X,z|\theta_t)} \right) \\
&= \log L(\theta_t|X) + \sum_z p(z|X,\theta_t) \log 1 \\
&= \log L(\theta_t|X)
\end{aligned}$$

so if $\theta = \theta_t$ the functions $Q(\theta|\theta_t)$ and $\log L(\theta|X)$ are equal.

Our objective is to choose a θ that maximizes $\log L(\theta)$.

We have shown that $Q(\theta|\theta_t)$ is bounded above by $\log L(\theta|X)$ and that $Q(\theta|\theta_t) = \log L(\theta|X)$ holds for $\theta = \theta_t$.

Thus, any θ that increases $Q(\theta|\theta_t)$ will also increase $\log L(\theta|X)$. The greatest possible increase is achieved by selecting θ such that $Q(\theta|\theta_t)$ is maximized.

We denote this updated value as θ_{t+1}:

$$\begin{aligned}\theta_{t+1} &= \operatorname*{argmax}_{\theta} Q(\theta|\theta_t) \\ &= \operatorname*{argmax}_{\theta} \log L(\theta_t|X) + \sum_z p(z|X,\theta_t) \log\left(\frac{p(X|z,\theta)p(z|\theta)}{p(z|X,\theta_t)p(X|\theta_t)}\right)\end{aligned}$$

Our objective is to choose a θ that maximizes $\log L(\theta)$.

We have shown that $Q(\theta|\theta_t)$ is bounded above by $\log L(\theta|X)$ and that $Q(\theta|\theta_t) = \log L(\theta|X)$ holds for $\theta = \theta_t$.

Thus, any θ that increases $Q(\theta|\theta_t)$ will also increase $\log L(\theta|X)$. The greatest possible increase is achieved by selecting θ such that $Q(\theta|\theta_t)$ is maximized.

We denote this updated value as θ_{t+1}:

$$\begin{aligned}\theta_{t+1} &= \operatorname*{argmax}_{\theta} Q(\theta|\theta_t) \\ &= \operatorname*{argmax}_{\theta} \log L(\theta_t|X) + \sum_z p(z|X,\theta_t) \log\left(\frac{p(X|z,\theta)p(z|\theta)}{p(z|X,\theta_t)p(X|\theta_t)}\right)\end{aligned}$$

constant terms can be dropped
(they don't change the argmax)

Our objective is to choose a θ that maximizes $\log L(\theta)$.

We have shown that $Q(\theta|\theta_t)$ is bounded above by $\log L(\theta|X)$ and that $Q(\theta|\theta_t) = \log L(\theta|X)$ holds for $\theta = \theta_t$.

Thus, any θ that increases $Q(\theta|\theta_t)$ will also increase $\log L(\theta|X)$. The greatest possible increase is achieved by selecting θ such that $Q(\theta|\theta_t)$ is maximized.

We denote this updated value as θ_{t+1}:

$$\begin{aligned}
\theta_{t+1} &= \operatorname*{argmax}_{\theta} \sum_z p(z|X, \theta_t) \log \left(p(X|z, \theta) p(z|\theta) \right) \\
&= \operatorname*{argmax}_{\theta} \sum_z p(z|X, \theta_t) \log \left(\frac{p(X, z, \theta)}{p(z, \theta)} \frac{p(z, \theta)}{p(\theta)} \right) \\
&= \operatorname*{argmax}_{\theta} \sum_z p(z|X, \theta_t) \log p(X, z|\theta) \\
&= \operatorname*{argmax}_{\theta} \mathbb{E}_{Z|X,\theta_t} \left[\log p(X, z|\theta) \right]
\end{aligned}$$

In the solution

$$\theta_{t+1} = \underset{\theta}{\operatorname{argmax}}\, \mathbb{E}_{Z|X,\theta_t}\left[\log p(X, z|\theta)\right]$$

the expectation and maximisation steps are evident: the e-step is to determine the conditional expectation $\mathbb{E}_{Z|X,\theta_t}\left[\log p(X, z|\theta)\right]$ and the m-step is to maximise this expression wrt θ.

Convergence of EM

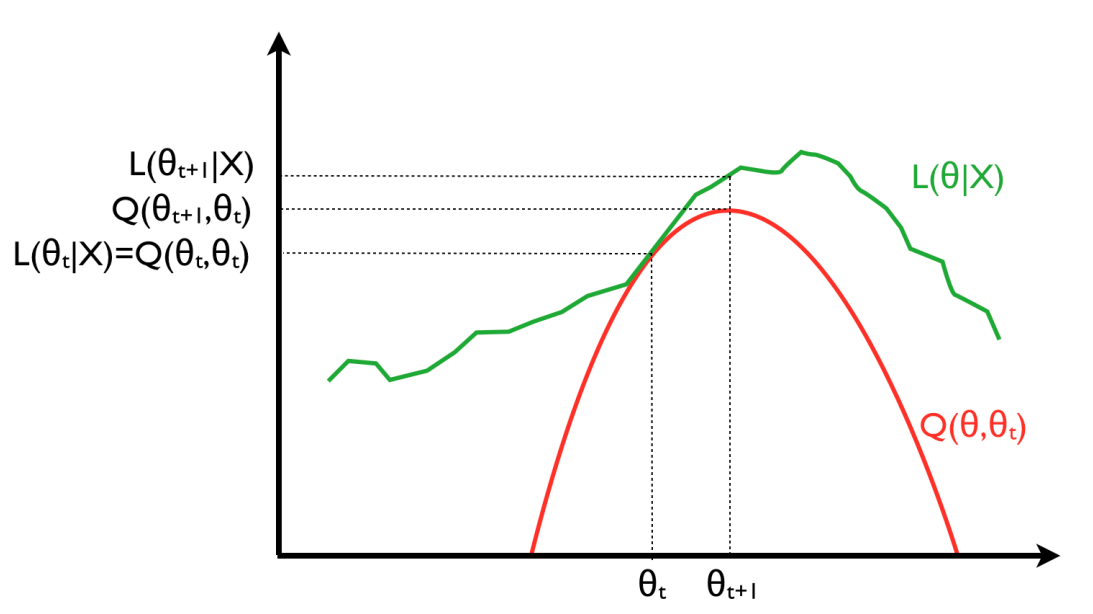
For an analysis of convergence see McLachlan & Krishnan (1996). In short:

θ_{t+1} is an estimate for θ that maximizes $\Delta(\theta, \theta_t)$

Since $\Delta(\theta_t, \theta_t) = 0$, we have $\Delta(\theta_{t+1}, \theta_t) \geq \Delta(\theta_t, \theta_t) = 0$

Thus, the log likelihood is non-decreasing.

Once θ_t maximizes $Q(\theta, \theta_t)$, θ_t is a stationary point



Example: $2D$ GMM

Consider a $2D$ Gaussian mixture model (GMM)

$$p(X|\theta) = \sum_{k=1}^{K} \pi_k G(X|\mu_k, \Sigma_k)$$

with $\sum_k \pi_k = 1$ and G is the Gaussian distribution. We write $\theta_k = (\pi_k, \mu_k, \Sigma_k)$ and $\theta = (\theta_1, \ldots, \theta_K)$. Since p is a density it must be non-negative and integrate to one. That is,

$$\begin{aligned}
1 &= \int_{\mathbb{R}^2} p(x|\theta)\,dx \\
&= \int_{\mathbb{R}^2} \sum_{k=1}^{K} \pi_k G(X|\mu_k, \Sigma_k)\,dx \\
&= \sum_{k=1}^{K} \pi_k \int_{\mathcal{X}} G(X|\mu_k, \Sigma_k)\,dx \\
&= \sum_{k=1}^{K} \pi_k
\end{aligned}$$

Example: $2D$ GMM (cont'd)

The GMM

$$p(X|\theta) = \sum_{k=1}^{K} \pi_k G(X|\mu_k, \Sigma_k)$$

is a **generative model** because it can be used to generate data as follows: select a component k according to π_k and draw a point x from the Gaussian parameterized with θ_k

Assume we generated N data points and now aim to recover the parameters θ of the GMM from the generated data

If we knew which data point was drawn from which component, we could estimate parameters θ_k using maximum likelihood. However, this information is hidden and we need to resort to EM

Example: $2D$ GMM (cont'd)

The incomplete-data likelihood is given by

$$L(\theta|X) = \prod_{j=1}^{n} p(x_j|\theta)$$
$$= \prod_{j=1}^{n} \sum_{i=1}^{k} \pi_i G(x_j|\mu_i, \Sigma_i)$$

Since we don't know which component generated which point, introduce latent variables z. The complete-data likelihood is then given by

$$L(\theta|X, Z) = \prod_{n=1}^{N} p(x_n, z_n|\theta) = \prod_{n=1}^{N} p(x_n|z_n, \theta) p(z_n|\theta)$$

Latent variables $z_n \in \{1, \ldots, K\}$ indicate the (possible) generating components of point n. E.g., $p(z_n = k|\theta)$ encodes the probability that x_n was generated by the k-th Gaussian

Example: $2D$ GMM (cont'd)

The latent variables z_n are independent from other samples, thus

$$p(Z|X,\theta) = \prod_{n=1}^{N} p(z_n|x_n,\theta)$$

Bayes' rule gives us

$$\begin{aligned} p(z_n = k|x_n,\theta) &= \frac{p(x_n|z_n = k,\theta_k)p(z_n = k|\theta_k)}{\sum_{\ell=1}^{K} p(x_n|z_n,\theta_m)p(z_n = \ell|\theta_\ell)} \\ &= \frac{p(x_n|\theta_k)\pi_k}{\sum_{\ell=1}^{K} p(x_n|\theta_\ell)\pi_\ell} \\ &= \frac{\pi_k G(x_n|\mu_k,\Sigma_k)}{\sum_{\ell=1}^{K} \pi_\ell G(x_n|\mu_\ell,\Sigma_\ell)} \equiv \gamma_{kn} \end{aligned}$$

The γ_{kn} can be viewed as membership probabilities of point n and cluster k.

Example: $2D$ GMM (cont'd)

E-Step:

$$\gamma_{kn}^{(t+1)} \leftarrow \frac{\pi_k^{(t)} G(x_n|\mu_k^{(t)}, \Sigma_k^{(t)})}{\sum_{\ell=1}^{K} \pi_\ell^{(t)} G(x_n|\mu_\ell^{(t)}, \Sigma_\ell^{(t)})}$$

M-Step:

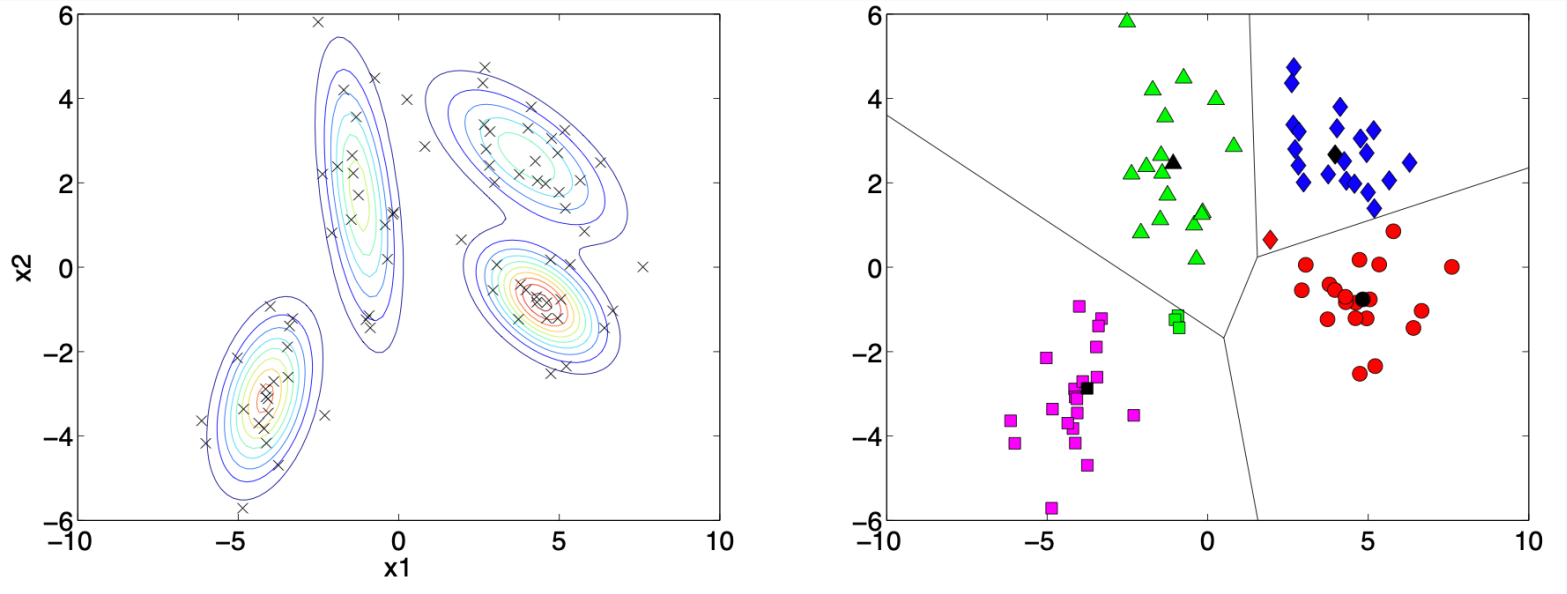
$$\pi_k^{(t+1)} \leftarrow \frac{1}{N} \sum_{n=1}^{N} p(z_n = k|x_n, \theta)$$

$$\mu_k^{(t+1)} \leftarrow \frac{1}{\sum_{n=1}^{N} \gamma_{kn}^{(t+1)}} \sum_{n=1}^{N} \gamma_{kn}^{(t+1)} x_n$$

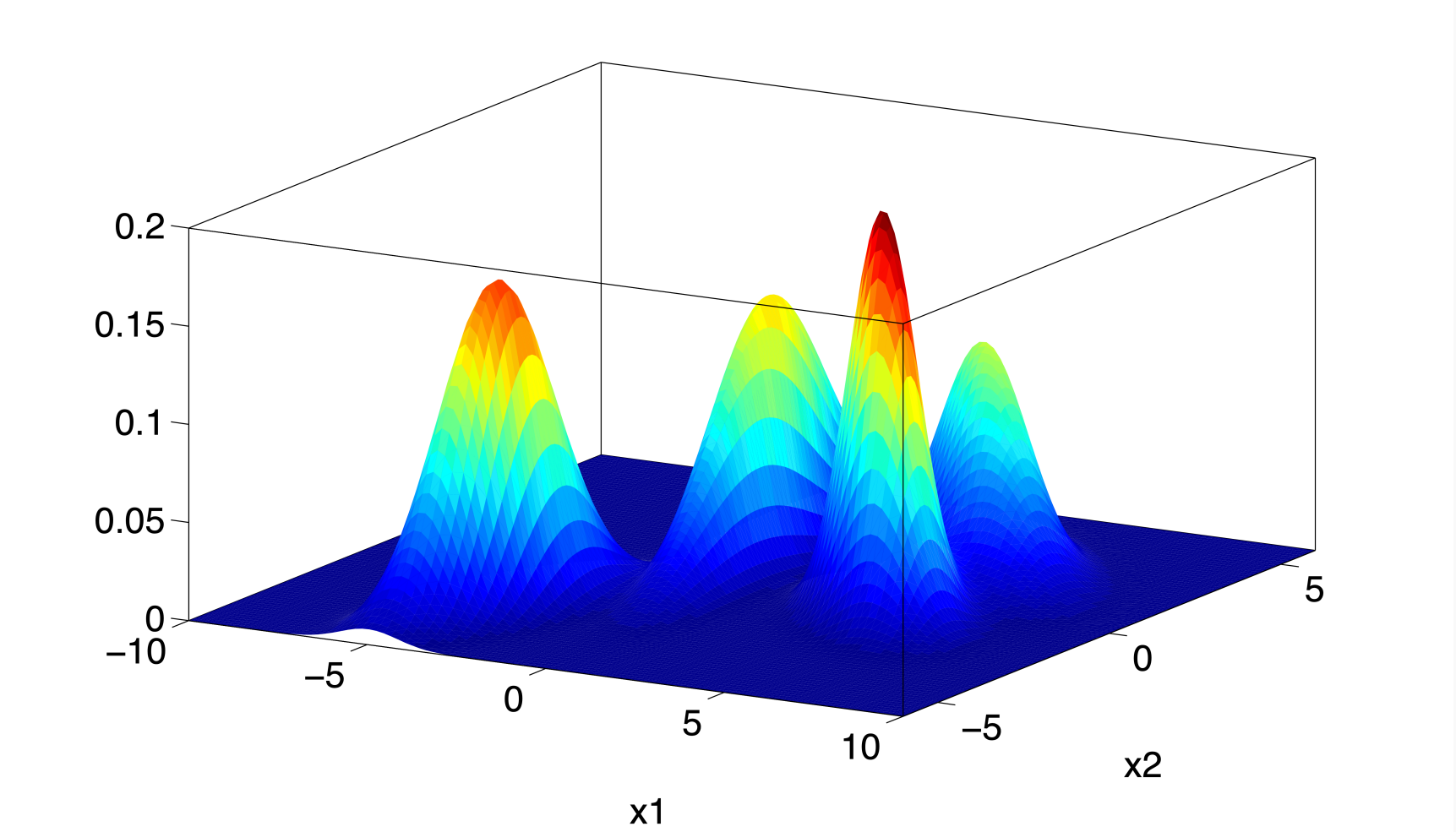
$$\Sigma_k^{(t+1)} \leftarrow \frac{1}{\sum_{n=1}^{N} \gamma_{kn}^{(t+1)}} \sum_{n=1}^{N} \gamma_{kn}^{(t+1)} (x_n - \mu_k^{(t+1)})(x_n - \mu_k^{(t+1)})^\top$$

Example: 2D GMM (cont'd)

For clustering tasks, EM (left) works like ML with estimated latent variables and leads to nice cluster boundaries comparted to hard K-means clusterings (right)



Example: 2D GMM (cont'd)



References

References (still incomplete)

W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5(4):115–133, 1943.

F. Rosenblatt. The perceptron – A perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory, 1957.

References

H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. Biological Cybernetics, 59(4):291–294, 1988.

Ahmed Elgammal, Bingchen Liu, Mohamed Elhoseiny, and Marian Mazzone. CAN: Creative Adversarial Networks Generating "Art" by Learning About Styles and Deviating from Style Norms, arXiv:1706.07068v1, 2017.

F. A. Gers, J. A. Schmidhuber, and F. A. Cummins. Learning to forget: Continual prediction with lstm. Neural Computation, 12(10):2451–2471, 2000.

F. A. Gers, N. N. Schraudolph, and J. Schmidhuber. Learning precise timing with lstm recurrent networks. Journal of Machine Learning Research (JMLR), 3:115–143, 2003.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. CoRR, abs/1406.2661, 2014.

References (still incomplete)

G. E. Hinton and R. S. Zemel. Autoencoders, minimum description length and helmholtz free energy. In Neural Information Processing Systems (NIPS), pages 3–10, 1994.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. CoRR, abs/1512.03385, 2015.

S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural Computation, 9(8): 1735–1780, 1997.

R. Jozefowicz, W. Zaremba, and I. Sutskever. An empirical exploration of recurrent network architectures. In International Conference on Machine Learning (ICML), pages 2342–2350, 2015.

A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. CoRR, abs/1511.06434, 2015.