Advanced Statistical Inference
Clustering

Maurizio Filippone
Maurizio.Filippone@eurecom.fr

Department of Data Science
EURECOM

Unsupervised learning

- Everything we've seen so far has been *supervised*
- We were given a set of $\mathbf{x}_n$ **and** associated t_n.

Unsupervised learning

- Everything we've seen so far has been *supervised*
- We were given a set of $\mathbf{x}_n$ **and** associated t_n.
- What if we just have $\mathbf{x}_n$?
- For example:
  - $\mathbf{x}_n$ is a binary vector indicating products customer n has bought.
  - Can group customers that buy similar products.
  - Can group products bought together.

Unsupervised learning

- Everything we've seen so far has been *supervised*
- We were given a set of $\mathbf{x}_n$ **and** associated t_n.
- What if we just have $\mathbf{x}_n$?
- For example:
  - $\mathbf{x}_n$ is a binary vector indicating products customer n has bought.
  - Can group customers that buy similar products.
  - Can group products bought together.
- Known as **Clustering**
- And is an example of *unsupervised* learning.

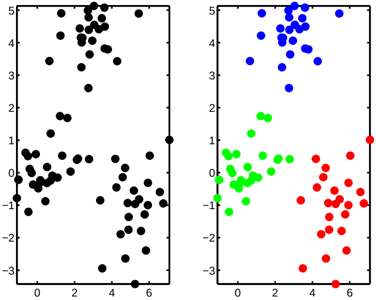
Unsupervised learning

- Everything we've seen so far has been *supervised*
- We were given a set of $\mathbf{x}_n$ **and** associated t_n.
- What if we just have $\mathbf{x}_n$?
- For example:
  - $\mathbf{x}_n$ is a binary vector indicating products customer n has bought.
  - Can group customers that buy similar products.
  - Can group products bought together.
- Known as **Clustering**
- And is an example of *unsupervised* learning.
- We'll also cover *projection* (later in the course)

Aims

- This is the only lecture on clustering.
- By the end, you should:
    - Understand what clustering is.
    - Understand the K-means algorithm.
    - Understand the idea of mixture models.
    - Be able to derive the update expression for mixture model parameters.

Clustering



- In this example each object has two attributes:

$$\mathbf{x}_n = [x_{n1}, x_{n2}]^{\mathsf{T}}$$

- Left: data.
- Right: data after clustering (points coloured according to cluster membership).

What we'll cover

- 2 algorithms:
  - K-means
  - Mixture models
- The two are somewhat related.
- We'll also see how K-means can be kernelized.

What we'll cover

- 2 algorithms:
  - **K-means**
  - Mixture models
- The two are somewhat related.
- We'll also see how K-means can be kernelized.

K-means

- Assume that there are K clusters.
- Each cluster is defined by a position in the input space:

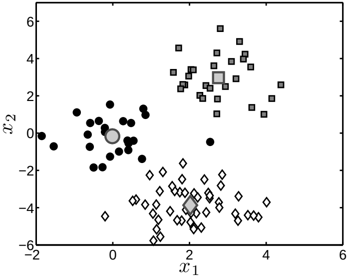
$$\boldsymbol{\mu}_k = [\mu_{k1}, \mu_{k2}]^\mathsf{T}$$

K-means

- Assume that there are K clusters.
- Each cluster is defined by a position in the input space:

$$\boldsymbol{\mu}_k = [\mu_{k1}, \mu_{k2}]^\mathsf{T}$$

- Each $\mathbf{x}_n$ is assigned to its closest cluster:

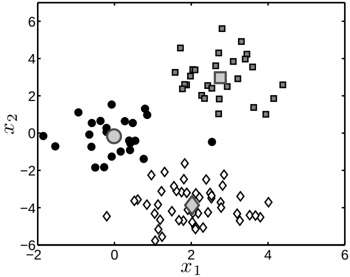


K-means

- Assume that there are K clusters.
- Each cluster is defined by a position in the input space:

$$\boldsymbol{\mu}_k = [\mu_{k1}, \mu_{k2}]^\mathsf{T}$$

- Each $\mathbf{x}_n$ is assigned to its closest cluster:



- Distance is normally Euclidean distance:

$$d_{nk} = (\mathbf{x}_n - \boldsymbol{\mu}_k)^\mathsf{T}(\mathbf{x}_n - \boldsymbol{\mu}_k)$$

How do we find $\boldsymbol{\mu}_k$?

- No analytical solution – we can't write down $\boldsymbol{\mu}_k$ as a function of $\mathbf{X}$.
- Use an iterative algorithm:

How do we find $\boldsymbol{\mu}_k$?

- No analytical solution – we can't write down $\boldsymbol{\mu}_k$ as a function of $\mathbf{X}$.
- Use an iterative algorithm:
  1. Guess $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$

How do we find $\boldsymbol{\mu}_k$?

- No analytical solution – we can't write down $\boldsymbol{\mu}_k$ as a function of $\mathbf{X}$.
- Use an iterative algorithm:
    1. Guess $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$
    2. Assign each $\mathbf{x}_n$ to its closest $\boldsymbol{\mu}_k$

How do we find $\boldsymbol{\mu}_k$?

- No analytical solution – we can't write down $\boldsymbol{\mu}_k$ as a function of $\mathbf{X}$.
- Use an iterative algorithm:
  1. Guess $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$
  2. Assign each $\mathbf{x}_n$ to its closest $\boldsymbol{\mu}_k$
  3. $z_{nk} = 1$ if $\mathbf{x}_n$ assigned to $\boldsymbol{\mu}_k$ (0 otherwise)

How do we find $\boldsymbol{\mu}_k$?

- No analytical solution – we can't write down $\boldsymbol{\mu}_k$ as a function of $\mathbf{X}$.
- Use an iterative algorithm:
  1. Guess $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$
  2. Assign each $\mathbf{x}_n$ to its closest $\boldsymbol{\mu}_k$
  3. $z_{nk} = 1$ if $\mathbf{x}_n$ assigned to $\boldsymbol{\mu}_k$ (0 otherwise)
  4. Update $\boldsymbol{\mu}_k$ to average of $\mathbf{x}_n$s assigned to $\boldsymbol{\mu}_k$:

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} z_{nk}\mathbf{x}_n}{\sum_{n=1}^{N} z_{nk}}$$

How do we find $\boldsymbol{\mu}_k$?

- No analytical solution – we can't write down $\boldsymbol{\mu}_k$ as a function of $\mathbf{X}$.
- Use an iterative algorithm:
  1. Guess $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$
  2. Assign each $\mathbf{x}_n$ to its closest $\boldsymbol{\mu}_k$
  3. $z_{nk} = 1$ if $\mathbf{x}_n$ assigned to $\boldsymbol{\mu}_k$ (0 otherwise)
  4. Update $\boldsymbol{\mu}_k$ to average of $\mathbf{x}_n$s assigned to $\boldsymbol{\mu}_k$:

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} z_{nk}\mathbf{x}_n}{\sum_{n=1}^{N} z_{nk}}$$

  5. Return to 2 until assignments do not change.

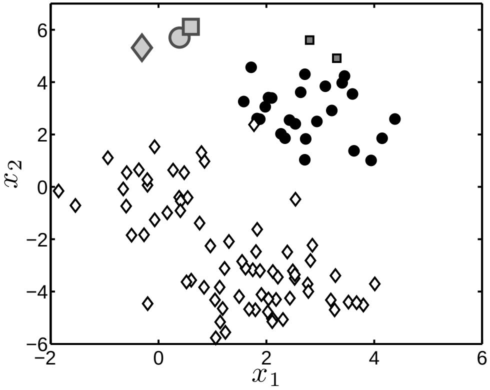
How do we find $\boldsymbol{\mu}_k$?

- No analytical solution – we can't write down $\boldsymbol{\mu}_k$ as a function of $\mathbf{X}$.
- Use an iterative algorithm:
  1. Guess $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$
  2. Assign each $\mathbf{x}_n$ to its closest $\boldsymbol{\mu}_k$
  3. $z_{nk} = 1$ if $\mathbf{x}_n$ assigned to $\boldsymbol{\mu}_k$ (0 otherwise)
  4. Update $\boldsymbol{\mu}_k$ to average of $\mathbf{x}_n$s assigned to $\boldsymbol{\mu}_k$:

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^N z_{nk}\mathbf{x}_n}{\sum_{n=1}^N z_{nk}}$$

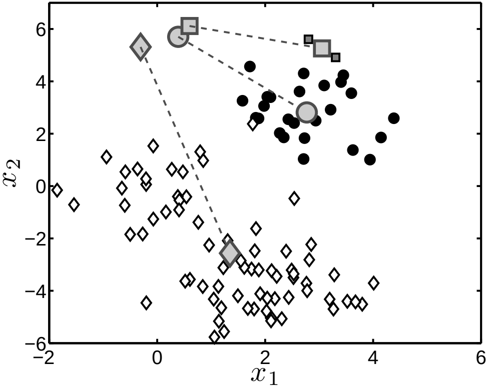
  5. Return to 2 until assignments do not change.
- Algorithm will converge....it will reach a point where the assignments don't change.

K-means – example



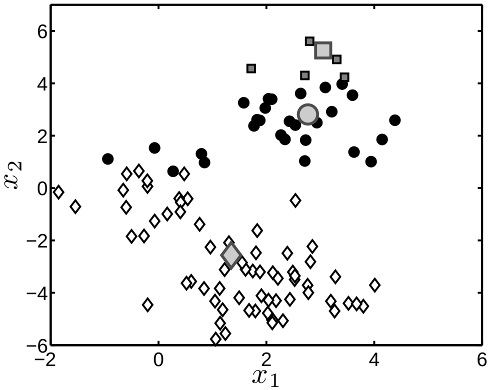
- Cluster means randomly assigned (top left).
- Points assigned to their closest mean.

K-means – example



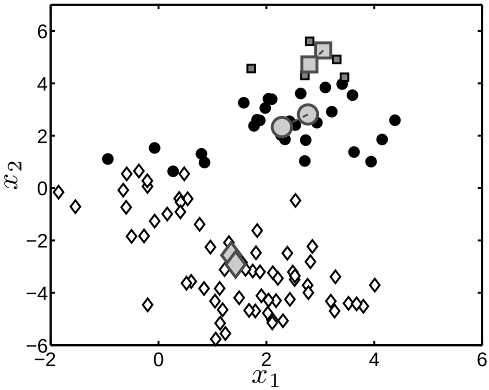
- Cluster means updated to mean of assigned points.

K-means – example



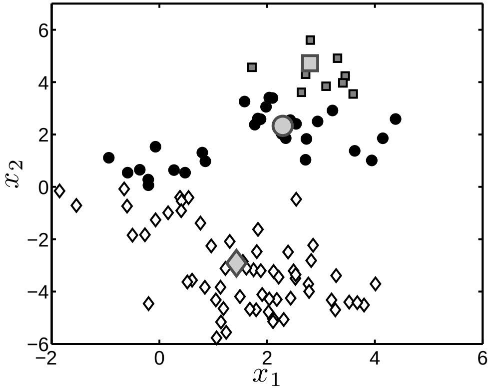
- Points re-assigned to closest mean.

K-means – example



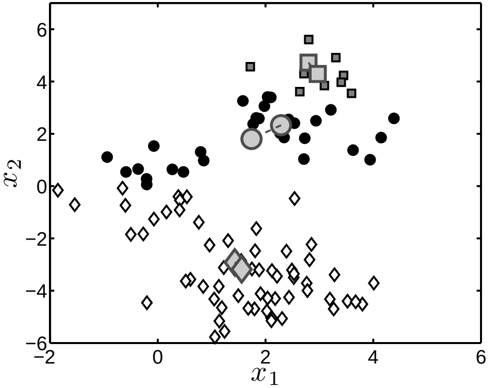
- Cluster means updated to mean of assigned points.

K-means – example



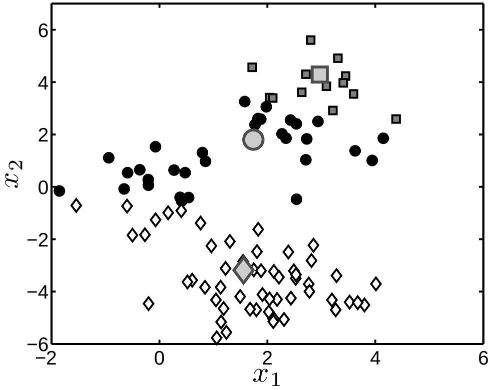
- Assign point to closest mean.

K-means – example



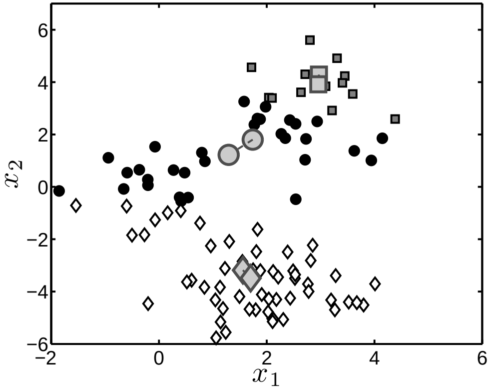
- Update mean.

K-means – example



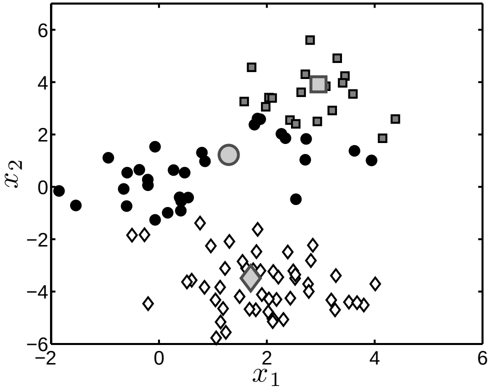
- Assign point to closest mean.

K-means – example



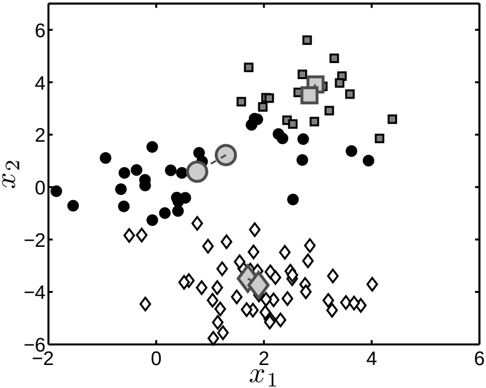
- Update mean.

K-means – example



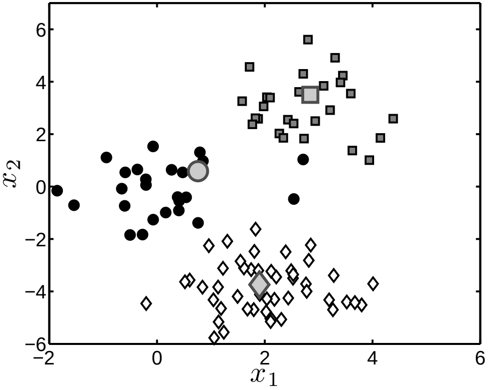
- Assign point to closest mean.

K-means – example



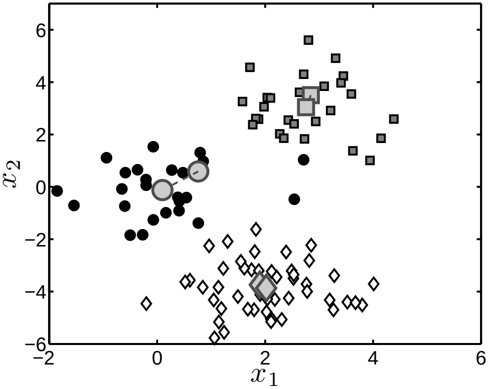
- Update mean.

K-means – example



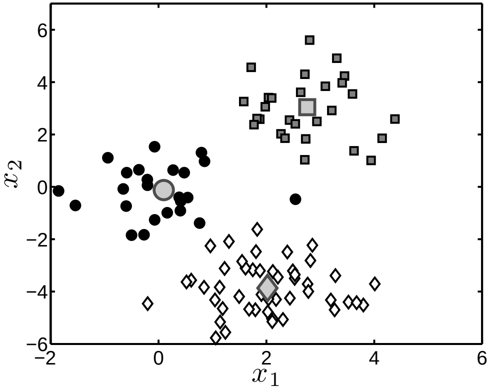
- Assign point to closest mean.

K-means – example



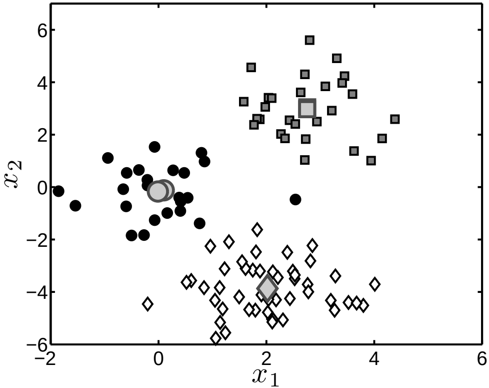
- Update mean.

K-means – example



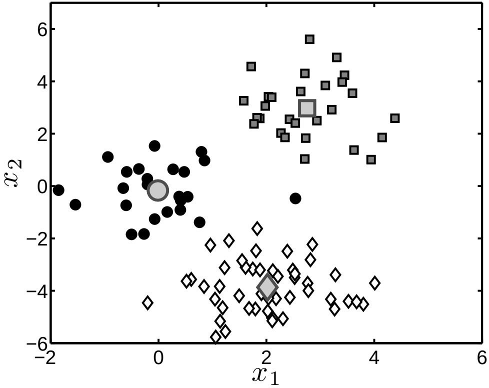
- Assign point to closest mean.

K-means – example



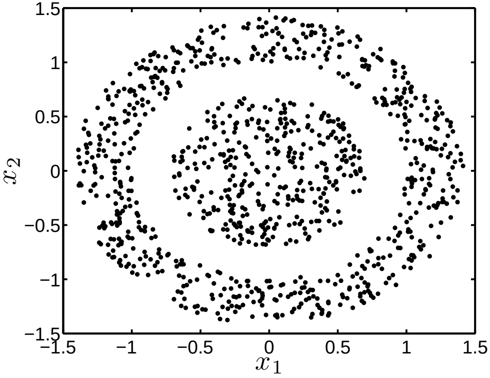
- Update mean.

K-means – example



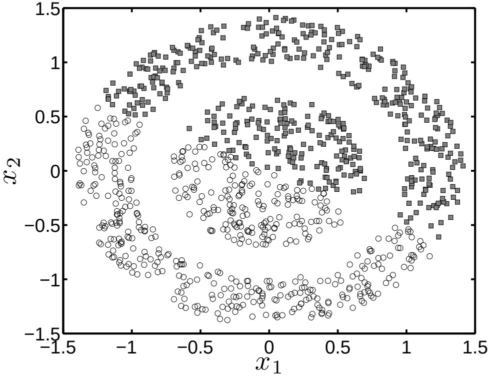
- Solution at convergence.

When does K-means break?



- Data has clear cluster structure.
- Outer cluster can not be represented as a single point.

When does K-means break?



- Data has clear cluster structure.
- Outer cluster can not be represented as a single point.

Kernelizing K-means

- Maybe we can kernelize K-means?
- Distances:

$$(\mathbf{x}_n - \boldsymbol{\mu}_k)^\mathsf{T}(\mathbf{x}_n - \boldsymbol{\mu}_k)$$

Kernelizing K-means

- Maybe we can kernelize K-means?
- Distances:

$$(\mathbf{x}_n - \boldsymbol{\mu}_k)^\mathsf{T}(\mathbf{x}_n - \boldsymbol{\mu}_k)$$

- Cluster means:

$$\boldsymbol{\mu}_k = \frac{\sum_{m=1}^{N} z_{mk}\mathbf{x}_m}{\sum_{m=1}^{N} z_{mk}}$$

Kernelizing K-means

- Maybe we can kernelize K-means?
- Distances:

$$(\mathbf{x}_n - \boldsymbol{\mu}_k)^\mathsf{T}(\mathbf{x}_n - \boldsymbol{\mu}_k)$$

- Cluster means:

$$\boldsymbol{\mu}_k = \frac{\sum_{m=1}^N z_{mk}\mathbf{x}_m}{\sum_{m=1}^N z_{mk}}$$

- Distances can be written as (defining $N_k = \sum_n z_{nk}$):

$$(\mathbf{x}_n - \boldsymbol{\mu}_k)^\mathsf{T}(\mathbf{x}_n - \boldsymbol{\mu}_k) = \left(\mathbf{x}_n - N_k^{-1}\sum_{m=1}^N z_{mk}\mathbf{x}_m\right)^\mathsf{T}\left(\mathbf{x}_n - N_k^{-1}\sum_{m=1}^N z_{mk}\mathbf{x}_m\right)$$

Kernelizing K-means

- Multiply out:

$$\mathbf{x}_n^\mathsf{T}\mathbf{x}_n - 2N_k^{-1}\sum_{m=1}^{N} z_{mk}\mathbf{x}_m^\mathsf{T}\mathbf{x}_n + N_k^{-2}\sum_{m,l} z_{mk}z_{lk}\mathbf{x}_m^\mathsf{T}\mathbf{x}_l$$

Kernelizing K-means

- Multiply out:

$$\mathbf{x}_n^{\mathsf{T}}\mathbf{x}_n - 2N_k^{-1}\sum_{m=1}^{N} z_{mk}\mathbf{x}_m^{\mathsf{T}}\mathbf{x}_n + N_k^{-2}\sum_{m,l} z_{mk}z_{lk}\mathbf{x}_m^{\mathsf{T}}\mathbf{x}_l$$

- Kernel substitution:

$$k(\mathbf{x}_n,\mathbf{x}_n) - 2N_k^{-1}\sum_{m=1}^{N} z_{mk}k(\mathbf{x}_n,\mathbf{x}_m) + N_k^{-2}\sum_{m,l=1}^{N} z_{mk}z_{lk}k(\mathbf{x}_m,\mathbf{x}_l)$$

Kernel K-means

- Algorithm:
  1. Choose a kernel and any necessary parameters.

Kernel K-means

- Algorithm:
  1. Choose a kernel and any necessary parameters.
  2. Start with random assignments z_{nk}.

Kernel K-means

- Algorithm:
  1. Choose a kernel and any necessary parameters.
  2. Start with random assignments z_{nk}.
  3. For each $\mathbf{x}_n$ assign it to the nearest 'center' where distance is defined as:

$$k(\mathbf{x}_n, \mathbf{x}_n) - 2N_k^{-1} \sum_{m=1}^{N} z_{mk} k(\mathbf{x}_n, \mathbf{x}_m) + N_k^{-2} \sum_{m,l=1}^{N} z_{mk} z_{lk} k(\mathbf{x}_m, \mathbf{x}_l)$$

Kernel K-means

- Algorithm:
  1. Choose a kernel and any necessary parameters.
  2. Start with random assignments z_{nk}.
  3. For each $\mathbf{x}_n$ assign it to the nearest 'center' where distance is defined as:

$$k(\mathbf{x}_n, \mathbf{x}_n) - 2N_k^{-1} \sum_{m=1}^{N} z_{mk} k(\mathbf{x}_n, \mathbf{x}_m) + N_k^{-2} \sum_{m,l=1}^{N} z_{mk} z_{lk} k(\mathbf{x}_m, \mathbf{x}_l)$$

  4. If assignments have changed, return to 3.

Kernel K-means

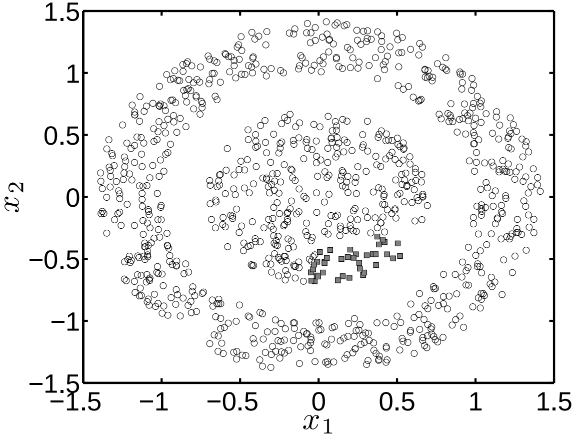
- Algorithm:
  1. Choose a kernel and any necessary parameters.
  2. Start with random assignments z_{nk}.
  3. For each $\mathbf{x}_n$ assign it to the nearest 'center' where distance is defined as:

$$k(\mathbf{x}_n, \mathbf{x}_n) - 2N_k^{-1} \sum_{m=1}^{N} z_{mk} k(\mathbf{x}_n, \mathbf{x}_m) + N_k^{-2} \sum_{m,l=1}^{N} z_{mk} z_{lk} k(\mathbf{x}_m, \mathbf{x}_l)$$

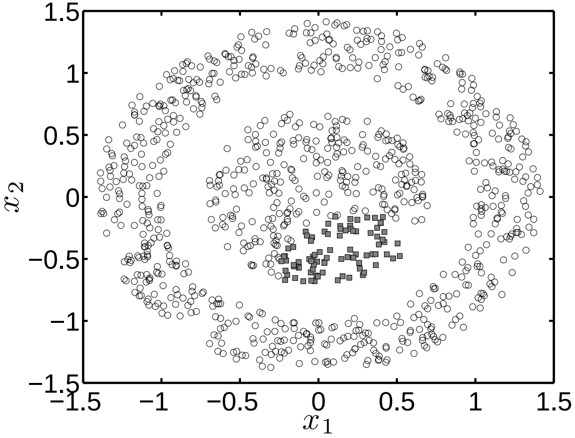
  4. If assignments have changed, return to 3.

- Note – no $\boldsymbol{\mu}_k$. This would be $N_k^{-1} \sum_n z_{nk} \phi(\mathbf{x}_n)$ but we don't know $\phi(\mathbf{x}_n)$ for kernels. We only know $\phi(\mathbf{x}_n)^\mathsf{T} \phi(\mathbf{x}_m)$...

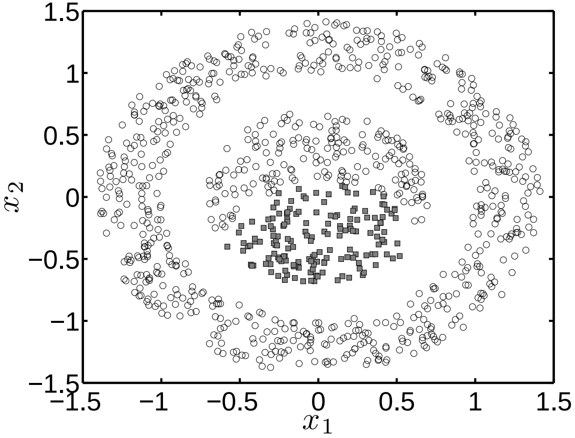
Kernel K-means – example



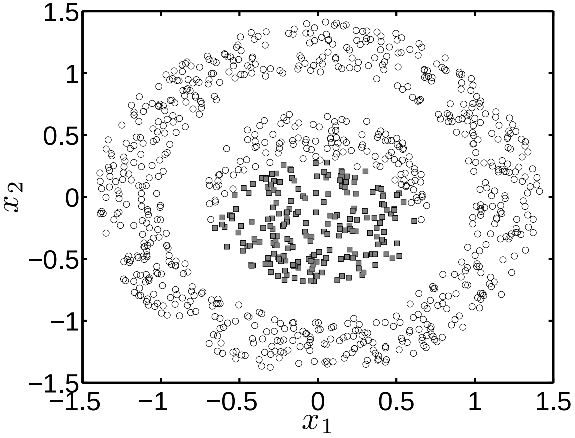
Kernel K-means – example



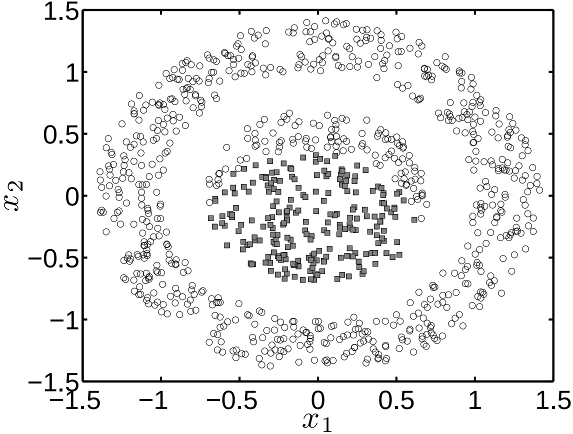
Kernel K-means – example



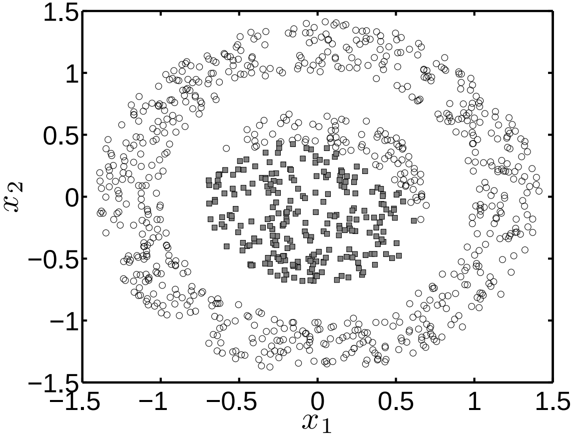
Kernel K-means – example



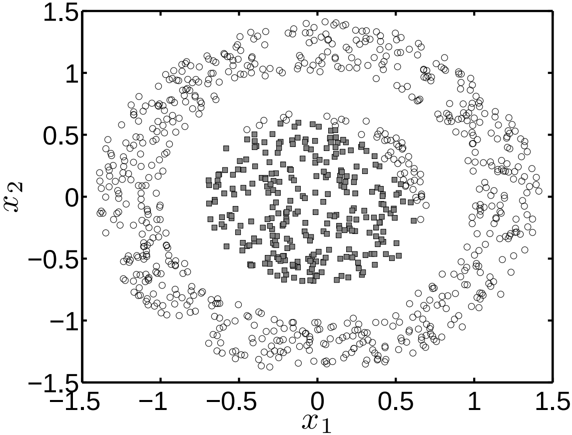
Kernel K-means – example



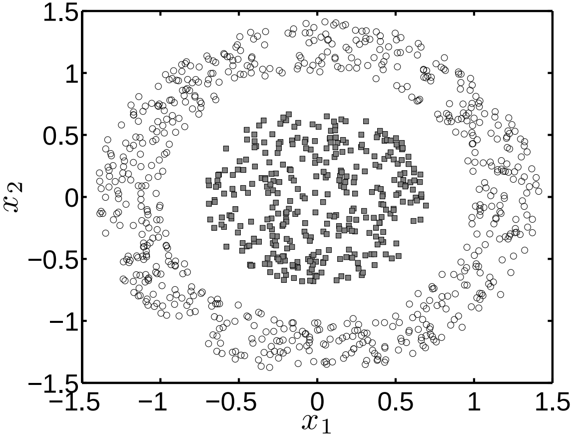
Kernel K-means – example



Kernel K-means – example



Kernel K-means – example



- Solution at convergence.

Kernel K-means

- Makes simple K-means algorithm more flexible.
- But, have to now set additional parameters.
- Very sensitive to initial conditions – lots of local optima.

K-means – summary

- Simple (and effective) clustering strategy.

K-means – summary

- Simple (and effective) clustering strategy.
- Converges to (local) minima of:

$$\sum_n \sum_k z_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k)^\mathsf{T} (\mathbf{x}_n - \boldsymbol{\mu}_k)$$

K-means – summary

- Simple (and effective) clustering strategy.
- Converges to (local) minima of:

$$\sum_{n}\sum_{k} z_{nk}(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathsf{T}}(\mathbf{x}_n - \boldsymbol{\mu}_k)$$

- Sensitive to initialization.

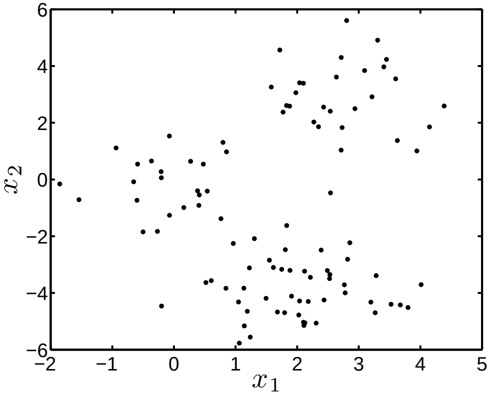
K-means – summary

- Simple (and effective) clustering strategy.
- Converges to (local) minima of:

$$\sum_n \sum_k z_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k)^\mathsf{T} (\mathbf{x}_n - \boldsymbol{\mu}_k)$$

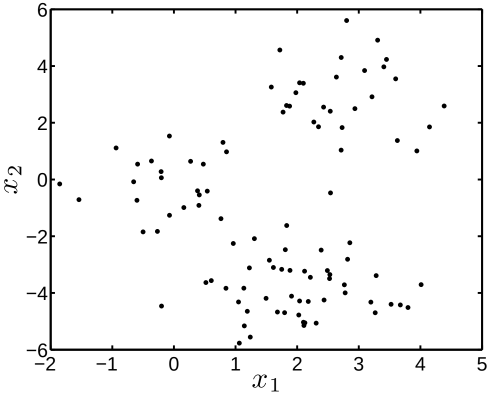
- Sensitive to initialization.
- How do we choose K?
  - Tricky: Quantity above always decreases as K increases.
  - Can use CV if we have a measure of 'goodness'.
  - For clustering these will be application specific.

Mixture models – thinking generatively



- Could we hypothesis a model that could have created this data?

Mixture models – thinking generatively



- Could we hypothesis a model that could have created this data?
- Each $\mathbf{x}_n$ seems to have come from one of three distributions.

A generative model

- Assumption:Each $\mathbf{x}_n$ comes from one of different K distributions.
- To generate $\mathbf{X}$:
- For each n:
  1. Pick one of the K components.
  2. Sample $\mathbf{x}_n$ from this distribution.

A generative model

- Assumption:Each $\mathbf{x}_n$ comes from one of different K distributions.
- To generate $\mathbf{X}$:
- For each n:
    1. Pick one of the K components.
    2. Sample $\mathbf{x}_n$ from this distribution.

- We already have $\mathbf{X}$

A generative model

- Assumption:Each $\mathbf{x}_n$ comes from one of different K distributions.
- To generate $\mathbf{X}$:
- For each n:
  1. Pick one of the K components.
  2. Sample $\mathbf{x}_n$ from this distribution.

- We already have $\mathbf{X}$
- Define parameters of all these distributions as Δ.
- We'd like to reverse-engineer this process learn Δ which we can then use to find which component each point came from.

A generative model

- Assumption:Each $\mathbf{x}_n$ comes from one of different K distributions.
- To generate $\mathbf{X}$:
- For each n:
  1. Pick one of the K components.
  2. Sample $\mathbf{x}_n$ from this distribution.

- We already have $\mathbf{X}$
- Define parameters of all these distributions as Δ.
- We'd like to reverse-engineer this process learn Δ which we can then use to find which component each point came from.
- Maximize the likelihood!

Mixture model likelihood

- Let the kth distribution have pdf: $p(\mathbf{x}_n | z_{nk} = 1, \Delta_k)$

Mixture model likelihood

- Let the kth distribution have pdf: $p(\mathbf{x}_n | z_{nk} = 1, \Delta_k)$
- We want the likelihood:

$$p(\mathbf{X}|\Delta)$$

Mixture model likelihood

- Let the kth distribution have pdf: $p(\mathbf{x}_n | z_{nk} = 1, \Delta_k)$
- We want the likelihood:

$$p(\mathbf{X}|\Delta)$$

- First, factorize:

$$p(\mathbf{X}|\Delta) = \prod_{i=1}^{N} p(\mathbf{x}_n|\Delta)$$

Mixture model likelihood

- Let the kth distribution have pdf: $p(\mathbf{x}_n|z_{nk}=1,\Delta_k)$
- We want the likelihood:

$$p(\mathbf{X}|\Delta)$$

- First, factorize:

$$p(\mathbf{X}|\Delta)=\prod_{i=1}^{N}p(\mathbf{x}_n|\Delta)$$

- Then, un-marginalize k:

$$\begin{aligned}p(\mathbf{X}|\Delta) &= \prod_{i=1}^{N}\sum_{k=1}^{K}p(\mathbf{x}_n,z_{nk}=1|\Delta)\\ &= \prod_{i=1}^{N}\sum_{k=1}^{K}p(\mathbf{x}_n|z_{nk}=1,\Delta_k)p(z_{nk}=1|\Delta)\end{aligned}$$

Mixture model likelihood

- So, we have a likelihood:

$$p(\mathbf{X}|\Delta) = \prod_{i=1}^{N} \sum_{k=1}^{K} p(\mathbf{x}_n|z_{nk} = 1, \Delta_k) p(z_{nk} = 1|\Delta)$$

- And we want to find Δ.

Mixture model likelihood

- So, we have a likelihood:

$$p(\mathbf{X}|\Delta) = \prod_{i=1}^{N} \sum_{k=1}^{K} p(\mathbf{x}_n|z_{nk} = 1, \Delta_k)p(z_{nk} = 1|\Delta)$$

- And we want to find Δ.
- So:

$$\operatorname*{argmax}_{\Delta} \prod_{i=1}^{N} \sum_{k=1}^{K} p(\mathbf{x}_n|z_{nk} = 1, \Delta_k)p(z_{nk} = 1|\Delta)$$

Mixture model likelihood

- So, we have a likelihood:

$$p(\mathbf{X}|\Delta) = \prod_{i=1}^{N} \sum_{k=1}^{K} p(\mathbf{x}_n | z_{nk} = 1, \Delta_k) p(z_{nk} = 1 | \Delta)$$

- And we want to find Δ.
- So:

$$\underset{\Delta}{\text{argmax}} \prod_{i=1}^{N} \sum_{k=1}^{K} p(\mathbf{x}_n | z_{nk} = 1, \Delta_k) p(z_{nk} = 1 | \Delta)$$

- Logging made this easier before, so let's try it:

$$\underset{\Delta}{\text{argmax}} \sum_{n=1}^{N} \log \sum_{k=1}^{K} p(\mathbf{x}_n | z_{nk} = 1, \Delta_k) p(z_{nk} = 1 | \Delta)$$

Mixture model likelihood

- So, we have a likelihood:

$$p(\mathbf{X}|\Delta) = \prod_{i=1}^{N} \sum_{k=1}^{K} p(\mathbf{x}_n | z_{nk} = 1, \Delta_k) p(z_{nk} = 1 | \Delta)$$

- And we want to find Δ.
- So:

$$\underset{\Delta}{\operatorname{argmax}} \prod_{i=1}^{N} \sum_{k=1}^{K} p(\mathbf{x}_n | z_{nk} = 1, \Delta_k) p(z_{nk} = 1 | \Delta)$$

- Logging made this easier before, so let's try it:

$$\underset{\Delta}{\operatorname{argmax}} \sum_{n=1}^{N} \log \sum_{k=1}^{K} p(\mathbf{x}_n | z_{nk} = 1, \Delta_k) p(z_{nk} = 1 | \Delta)$$

- Log of a sum is bad – we need some help....

Jensen's inequality

$$\log \mathbf{E}_{p(x)} \{f(x)\} \geq \mathbf{E}_{p(x)} \{\log f(x)\}$$

- How does this help us?
- Our log likelihood:

$$L = \sum_{n=1}^{N} \log \sum_{k=1}^{K} p(\mathbf{x}_n | z_{nk} = 1, \Delta_k) p(z_{nk} = 1 | \Delta)$$

- Add a (arbitrary looking) distribution $q(z_{nk} = 1)$ (s.t. $\sum_k q(z_{nk} = 1) = 1$):

$$L = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \frac{q(z_{nk} = 1)}{q(z_{nk} = 1)} p(\mathbf{x}_n | z_{nk} = 1, \Delta_k) p(z_{nk} = 1 | \Delta)$$

Jensen's inequality

$$L = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \frac{q(z_{nk} = 1)}{q(z_{nk} = 1)} p(\mathbf{x}_n | z_{nk} = 1, \Delta_k) p(z_{nk} = 1 | \Delta)$$

- We now have an expectation:

$$L = \sum_{n=1}^{N} \log \mathbf{E}_{q(z_{nk}=1)} \left\{ \frac{1}{q(z_{nk} = 1)} p(\mathbf{x}_n | z_{nk} = 1, \Delta_k) p(z_{nk} = 1 | \Delta) \right\}$$

Jensen's inequality

$$L = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \frac{q(z_{nk}=1)}{q(z_{nk}=1)} p(\mathbf{x}_n | z_{nk}=1, \Delta_k) p(z_{nk}=1|\Delta)$$

- We now have an expectation:

$$L = \sum_{n=1}^{N} \log \mathbf{E}_{q(z_{nk}=1)} \left\{ \frac{1}{q(z_{nk}=1)} p(\mathbf{x}_n | z_{nk}=1, \Delta_k) p(z_{nk}=1|\Delta) \right\}$$

- So, using Jensen's:

$$\begin{aligned} L &\geq \sum_{n=1}^{N} \mathbf{E}_{q(z_{nk}=1)} \left\{ \log \frac{1}{q(z_{nk}=1)} p(\mathbf{x}_n | z_{nk}=1, \Delta_k) p(z_{nk}=1|\Delta) \right\} \\ &= \sum_{n=1}^{N} \sum_{k=1}^{K} q(z_{nk}=1) \log \left\{ \frac{1}{q(z_{nk}=1)} p(\mathbf{x}_n | z_{nk}=1, \Delta_k) p(z_{nk}=1|\Delta) \right\} \end{aligned}$$

Lower bound on log-likelihood

$$
\begin{aligned}
L &\geq \sum_{n=1}^{N}\sum_{k=1}^{K} q(z_{nk}=1)\log\left\{\frac{1}{q(z_{nk}=1)}p(\mathbf{x}_n|z_{nk}=1,\Delta_k)p(z_{nk}=1|\Delta)\right\} \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} q(z_{nk}=1)\log p(z_{nk}=1|\Delta) + \dots \\
&\quad \sum_{n=1}^{N}\sum_{k=1}^{K} q(z_{nk}=1)\log p(\mathbf{x}_n|z_{nk}=1,\Delta_k) - \dots \\
&\quad \sum_{n=1}^{N}\sum_{k=1}^{K} q(z_{nk}=1)\log q(z_{nk}=1)
\end{aligned}
$$

Lower bound on log-likelihood

$$
\begin{aligned}
L &\geq \sum_{n=1}^{N}\sum_{k=1}^{K} q(z_{nk}=1)\log\left\{\frac{1}{q(z_{nk}=1)} p(\mathbf{x}_n|z_{nk}=1,\Delta_k)p(z_{nk}=1|\Delta)\right\} \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} q(z_{nk}=1)\log p(z_{nk}=1|\Delta) + \ldots \\
&\quad \sum_{n=1}^{N}\sum_{k=1}^{K} q(z_{nk}=1)\log p(\mathbf{x}_n|z_{nk}=1,\Delta_k) - \ldots \\
&\quad \sum_{n=1}^{N}\sum_{k=1}^{K} q(z_{nk}=1)\log q(z_{nk}=1)
\end{aligned}
$$

- Define $q_{nk} = q(z_{nk}=1)$, $\pi_k = p(z_{nk}=1|\Delta)$ (both just scalars).

Lower bound on log-likelihood

$$
\begin{aligned}
L &\geq \sum_{n=1}^{N}\sum_{k=1}^{K} q(z_{nk}=1)\log\left\{\frac{1}{q(z_{nk}=1)}p(\mathbf{x}_n|z_{nk}=1,\Delta_k)p(z_{nk}=1|\Delta)\right\} \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} q(z_{nk}=1)\log p(z_{nk}=1|\Delta)+\ldots \\
&\quad \sum_{n=1}^{N}\sum_{k=1}^{K} q(z_{nk}=1)\log p(\mathbf{x}_n|z_{nk}=1,\Delta_k)-\ldots \\
&\quad \sum_{n=1}^{N}\sum_{k=1}^{K} q(z_{nk}=1)\log q(z_{nk}=1)
\end{aligned}
$$

- Define $q_{nk}=q(z_{nk}=1)$, $\pi_k=p(z_{nk}=1|\Delta)$ (both just scalars).
- Differentiate lower bound w.r.t q_{nk}, π_k and Δ_k and set to zero to obtain **iterative** update equations.

Optimizing lower bound

- Updates for Δ_k, π_k will depend on q_{nk}.
- Update q_{nk} and then use these values to update Δ_k and π_k etc.

Optimizing lower bound

- Updates for Δ_k, π_k will depend on q_{nk}.
- Update q_{nk} and then use these values to update Δ_k and π_k etc.
- This is a form of the Expectation-Maximization algorithm (EM) but we've derived it differently.

Optimizing lower bound

- Updates for Δ_k, π_k will depend on q_{nk}.
- Update q_{nk} and then use these values to update Δ_k and π_k etc.
- This is a form of the Expectation-Maximization algorithm (EM) but we've derived it differently.
- Best illustrated with an example....

Gaussian mixture model

- Assume component distributions are Gaussians with diagonal covariance:

$$p(\mathbf{x}_n | z_{nk} = 1, \boldsymbol{\mu}_k, \sigma_k^2) = \mathcal{N}(\boldsymbol{\mu}, \sigma^2 \mathbf{I})$$

- Update for π_k. Relevant bit of bound:

$$\sum_{n,k} q_{nk} \log(\pi_k)$$

- Now, we have a constraint: $\sum_k \pi_k = 1$. So, add a Lagrangian:

$$\sum_{n,k} q_{nk} \log \pi_k - \lambda \left(\sum_k \pi_k - 1 \right)$$

- Differentiate and set to zero:

$$\frac{\partial}{\partial \pi_k} = \frac{1}{\pi_k} \sum_n q_{nk} - \lambda = 0$$

- Re-arrange:

$$\sum_n q_{nk} = \lambda \pi_k$$

- Re-arrange:

$$\sum_n q_{nk} = \lambda \pi_k$$

- Sum both sides over k to find λ:

$$\sum_{n,k} q_{nk} = \lambda \times 1$$

- Re-arrange:

$$\sum_n q_{nk} = \lambda \pi_k$$

- Sum both sides over k to find λ:

$$\sum_{n,k} q_{nk} = \lambda \times 1$$

- Substitute and re-arrange:

$$\pi_k = \frac{\sum_n q_{nk}}{\sum_{n,j} q_{nj}} = \frac{1}{N} \sum_k q_{nk}$$

Update for q_{nk}

- Now for q_{nk}. Whole bound is relevant.
- Add Lagrange term $-\lambda(\sum_k q_{nk} - 1)$
- Differentiate:

$$\frac{\partial}{\partial q_{nk}} = \log \pi_k + \log p(\mathbf{x}_n | z_{nk} = 1, \Delta_k) - (\log q_{nk} + 1) - \lambda$$

- Re-arranging ($\lambda' = f(\lambda)$):

$$\pi_k p(\mathbf{x}_n | z_{nk} = 1, \Delta_k) = \lambda' q_{nk}$$

- Sum over k to find λ' and re-arrange:

$$q_{nk} = \frac{\pi_k p(\mathbf{x}_n | z_{nk} = 1, \Delta_k)}{\sum_{j=1}^{K} \pi_j p(\mathbf{x}_n | z_{nj} = 1, \Delta_j)}$$

Updates for $\boldsymbol{\mu}_k$ and σ_k^2

- These are easier – no constraints.
- Differentiate the following and set to zero (D is dimension of $\mathbf{x}_n$):

$$\sum_{n,k} q_{nk} \log \left\{ \frac{1}{(2\pi\sigma_k^2)^{D/2}} \exp\left(-\frac{1}{2\sigma_k^2} (\mathbf{x}_n - \boldsymbol{\mu}_k)^\mathsf{T} (\mathbf{x}_n - \boldsymbol{\mu}_k) \right) \right\}$$

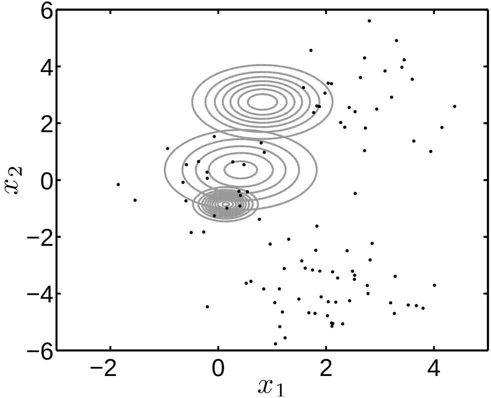
- Result:

$$\begin{aligned} \boldsymbol{\mu}_k &= \frac{\sum_n q_{nk} \mathbf{x}_n}{\sum_n q_{nk}} \\ \sigma_k^2 &= \frac{\sum_n q_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k)^\mathsf{T} (\mathbf{x}_n - \boldsymbol{\mu}_k)}{D \sum_n q_{nk}} \end{aligned}$$

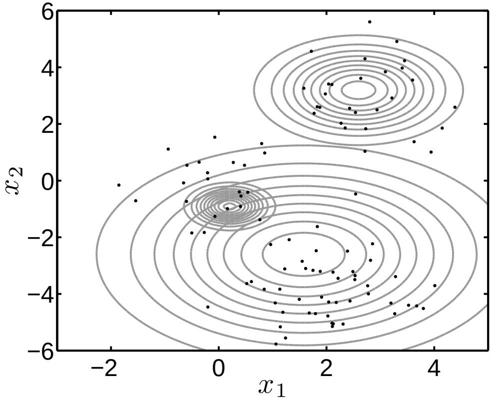
Mixture model optimization – algorithm

- Following optimization algorithm:
  1. Guess $\boldsymbol{\mu}_k, \sigma_k^2, \pi_k$
  2. Compute q_{nk}
  3. Update $\boldsymbol{\mu}_k, \sigma_k^2$
  4. Update π_k
  5. Return to 2 unless parameters are unchanged.
- Guaranteed to converge to a local maximum of the lower bound.
- Note the similarity with kmeans.

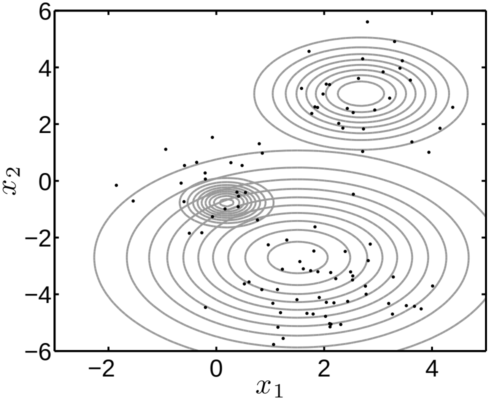
Algorithm in operation



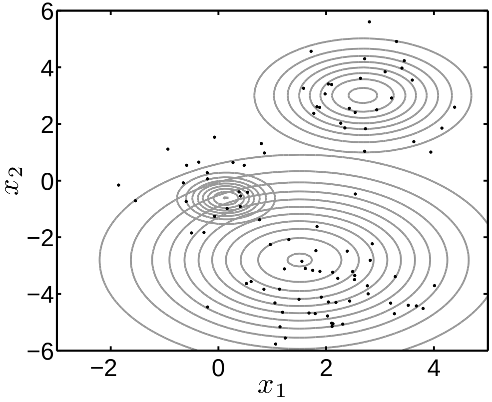
- Initial parameter values.



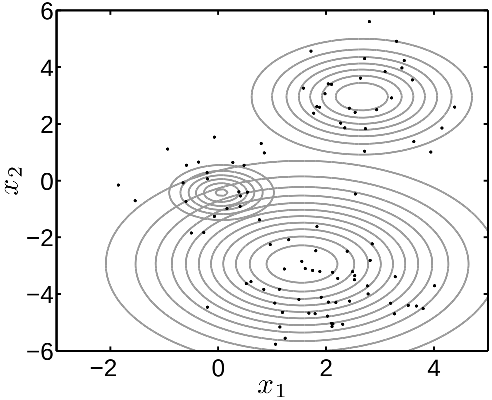
- Update q_{nk} and then other parameters.



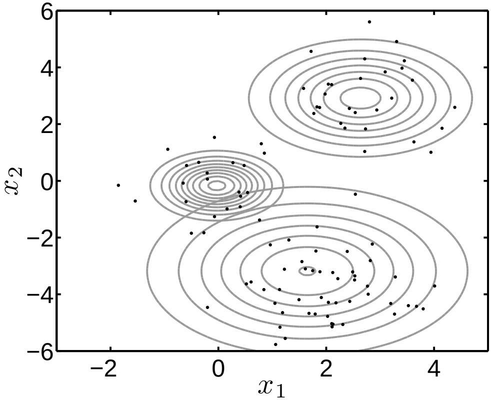
- Update q_{nk} and then other parameters.



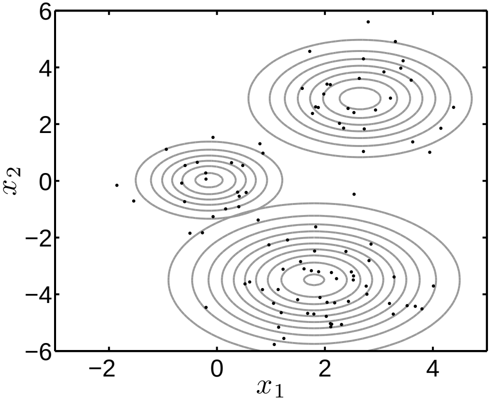
- Update q_{nk} and then other parameters.



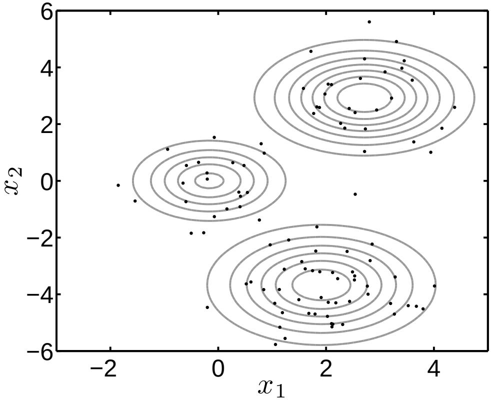
- Update q_{nk} and then other parameters.



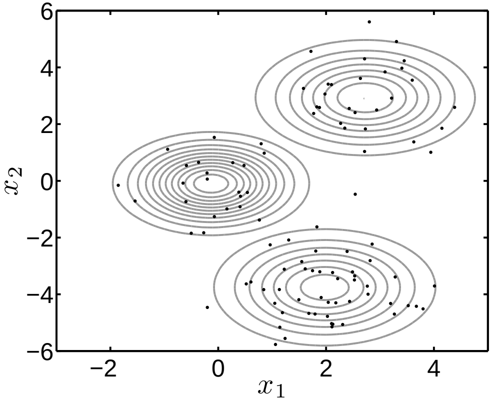
- Update q_{nk} and then other parameters.



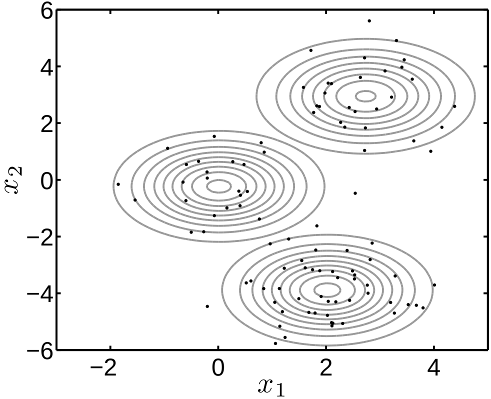
- Update q_{nk} and then other parameters.



- Update q_{nk} and then other parameters.



- Update q_{nk} and then other parameters.



- Solution at convergence.

Mixture model clustering

- So, we've got the parameters, but what about the assignments?
- Which points came from which distributions?

Mixture model clustering

- So, we've got the parameters, but what about the assignments?
- Which points came from which distributions?
- q_{nk} is the probability that $\mathbf{x}_n$ came from distribution k.

$$q_{nk} = P(z_{nk} = 1 | \mathbf{x}_n, \mathbf{X})$$

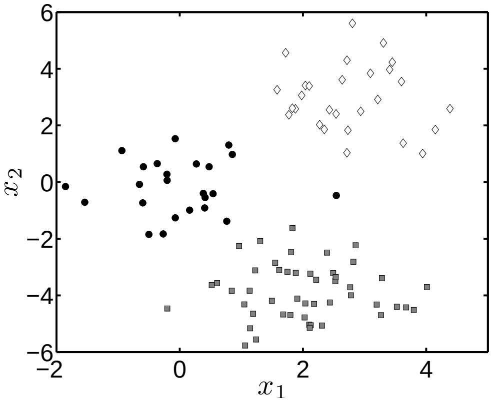
Mixture model clustering

- So, we've got the parameters, but what about the assignments?
- Which points came from which distributions?
- q_{nk} is the probability that $\mathbf{x}_n$ came from distribution k.

$$q_{nk} = P(z_{nk} = 1|\mathbf{x}_n, \mathbf{X})$$

- Can stick with probabilities or assign each $\mathbf{x}_n$ to it's most likely component.

Mixture model clustering



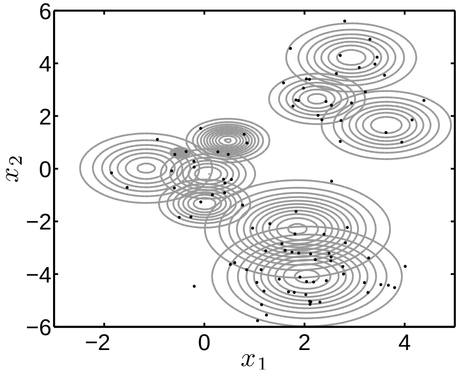
- Points assigned to the cluster with the highest q_{nk} value.

Mixture model – issues

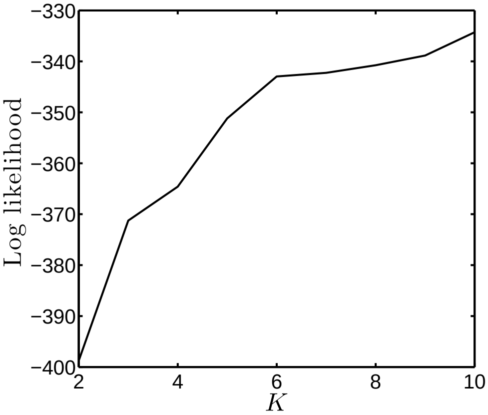
- How do we choose K?
- What happens when we increase it?

Mixture model – issues

- How do we choose K?
- What happens when we increase it?
- $K = 10$



Likelihood increase



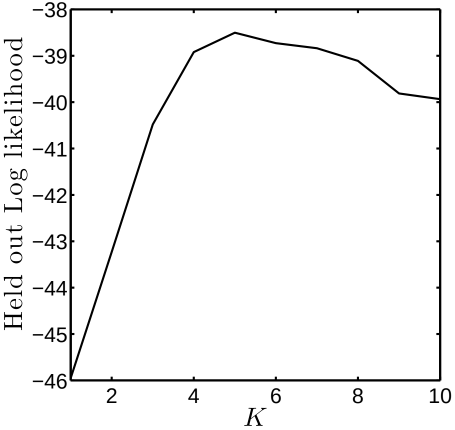
- Likelihood always increases as σ_k^2 decreases.

What can we do?

- What can we do?

What can we do?

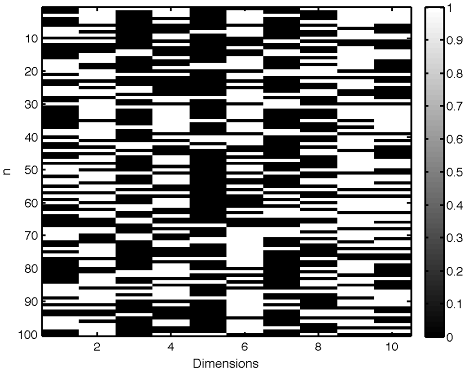
- What can we do?
- Cross-validation...



- 10-fold CV. Maximum is close to true value (3)

Mixture models – other distributions

- We've seen Gaussian distributions.
- Can actually use anything....
- As long as we can define $p(\mathbf{x}_n | z_{nk} = 1, \Delta_k)$
- e.g. Binary data:



Binary example

- $\mathbf{x}_n = [0, 1, 0, 1, 1, \ldots, 0, 1]^\mathsf{T}$ (D dimensions)
- $p(\mathbf{x}_n | z_{nk} = 1, \Delta_k) = \prod_{d=1}^{D} p_{kd}^{x_{nd}} (1 - p_{kd})^{1 - x_{nd}}$

Binary example

- $\mathbf{x}_n = [0, 1, 0, 1, 1, \ldots, 0, 1]^\mathsf{T}$ (D dimensions)
- $p(\mathbf{x}_n | z_{nk} = 1, \Delta_k) = \prod_{d=1}^{D} p_{kd}^{x_{nd}} (1 - p_{kd})^{1 - x_{nd}}$
- Updates for p_{kd} are:

$$p_{kd} = \frac{\sum_n q_{nk} x_{nd}}{\sum_n q_{nk}}$$

Binary example

- $\mathbf{x}_n = [0, 1, 0, 1, 1, \ldots, 0, 1]^\mathsf{T}$ (D dimensions)
- $p(\mathbf{x}_n | z_{nk} = 1, \Delta_k) = \prod_{d=1}^{D} p_{kd}^{x_{nd}} (1 - p_{kd})^{1 - x_{nd}}$
- Updates for p_{kd} are:

$$p_{kd} = \frac{\sum_n q_{nk} x_{nd}}{\sum_n q_{nk}}$$

- q_{nk} and π_k are the same as before...

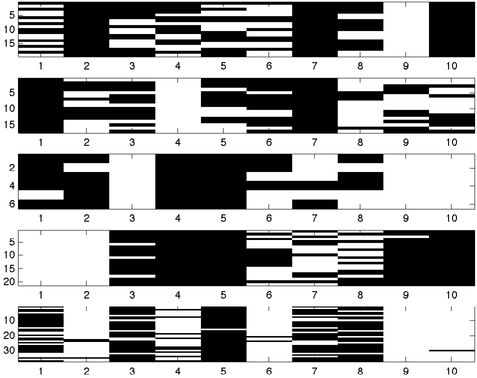
Binary example

- $\mathbf{x}_n = [0,1,0,1,1,\ldots,0,1]^\mathsf{T}$ (D dimensions)
- $p(\mathbf{x}_n | z_{nk} = 1, \Delta_k) = \prod_{d=1}^{D} p_{kd}^{x_{nd}} (1 - p_{kd})^{1 - x_{nd}}$
- Updates for p_{kd} are:

$$p_{kd} = \frac{\sum_n q_{nk} x_{nd}}{\sum_n q_{nk}}$$

- q_{nk} and π_k are the same as before...
- Initialize with random p_{kd} ($0 \leq p_{kd} \leq 1$)

Results



- $K = 5$ clusters.
- Clear structure present.

Summary

- Introduced two clustering methods.
- K-means
  - Very simple.
  - Iterative scheme.
  - Can be kernelized.
  - Need to choose K.
- Mixture models
  - Create a model of each class (similar to Bayes classifier)
  - Iterative sceme (EM)
  - Can use any distribution for the components.
  - Can set K by cross-validation (held-out likelihood)
  - State-of-the-art: Don't need to set K – treat as a variable in a Bayesian sampling scheme.