Clustering / Unsupervised Learning

- The target features are not given in the training examples
- The aim is to construct a natural classification that can be used to predict features of the data.

Clustering / Unsupervised Learning

- The target features are not given in the training examples
- The aim is to construct a natural classification that can be used to predict features of the data.
- The examples are partitioned in into clusters or classes. Each class predicts feature values for the examples in the class.
  - In hard clustering each example is placed definitively in a class.
  - In soft clustering each example has a probability distribution over its class.
- Each clustering has a prediction error on the examples. The best clustering is the one that minimizes the error.

k-means algorithm

The **k-means algorithm** is used for hard clustering.

Inputs:

- training examples
- the number of classes, k

Outputs:

- a prediction of a value for each feature for each class
- an assignment of examples to classes

k-means algorithm formalized

- E is the set of all examples
- the input features are $X_1, \ldots, X_n$
- $val(e, X_j)$ is the value of feature X_j for example e.
- there is a class for each integer $i \in \{1, \ldots, k\}$.

The k-means algorithm outputs

- a function $class : E \to \{1, \ldots, k\}$.
  $class(e) = i$ means e is in class i.
- a $pval$ function where $pval(i, X_j)$ is the prediction for each example in class i for feature X_j.

The sum-of-squares error for $class$ and $pval$ is

$$\sum_{e \in E} \sum_{j=1}^{n} \left(pval(class(e), X_j) - val(e, X_j)\right)^2 .$$

Aim: find $class$ and $pval$ that minimize sum-of-squares error.

Minimizing the error

The sum-of-squares error for *class* and *pval* is

$$\sum_{e \in E} \sum_{j=1}^{n} \left(pval(class(e), X_j) - val(e, X_j)\right)^2 .$$

- Given *class*, the *pval* that minimizes the sum-of-squares error is

Minimizing the error

The sum-of-squares error for *class* and *pval* is

$$\sum_{e \in E} \sum_{j=1}^{n} \left(pval(class(e), X_j) - val(e, X_j) \right)^2 .$$

- Given *class*, the *pval* that minimizes the sum-of-squares error is the mean value for that class.
- Given *pval*, each example can be assigned to the class that

Minimizing the error

The sum-of-squares error for *class* and *pval* is

$$\sum_{e \in E} \sum_{j=1}^{n} (pval(class(e), X_j) - val(e, X_j))^2 .$$

- Given *class*, the *pval* that minimizes the sum-of-squares error is the mean value for that class.
- Given *pval*, each example can be assigned to the class that minimizes the error for that example.

k-means algorithm

Initially, randomly assign the examples to the classes.
Repeat the following two steps:

- For each class i and feature X_j,

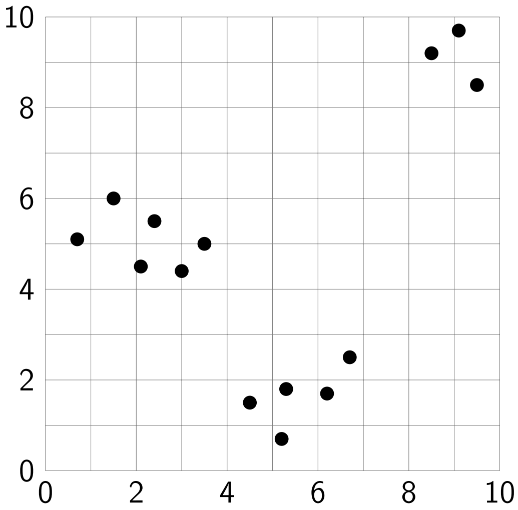
$$pval(i, X_j) \leftarrow \frac{\sum_{e:class(e)=i} val(e, X_j)}{|\{e : class(e) = i\}|},$$

- For each example e, assign e to the class i that minimizes

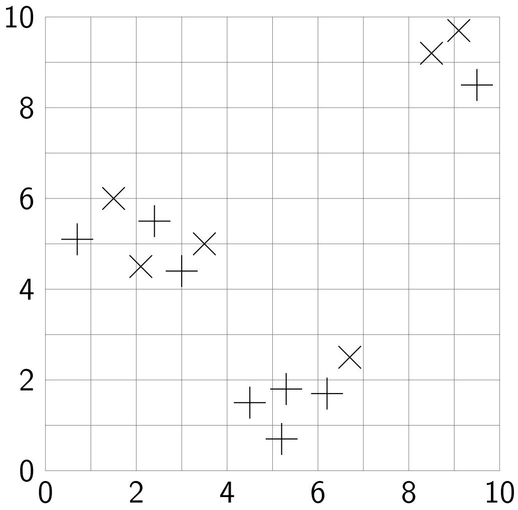
$$\sum_{j=1}^{n} \left(pval(i, X_j) - val(e, X_j)\right)^2 .$$

until the second step does not change the assignment of any example.

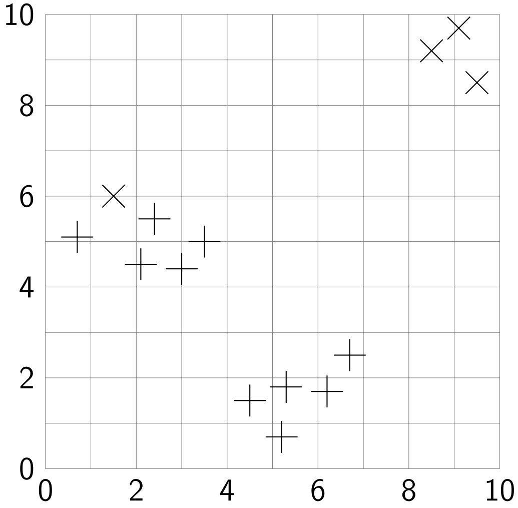
Example Data



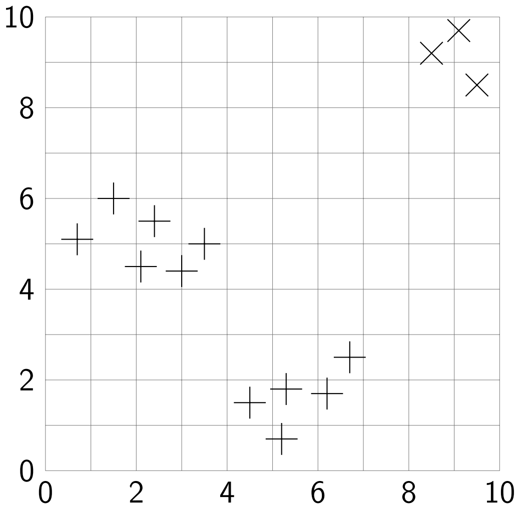
Random Assignment to Classes



Assign Each Example to Closest Mean



Ressign Each Example to Closest Mean



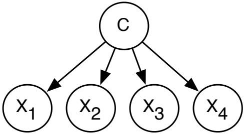
Properties of k-means

- An assignment of examples to classes is stable if running both the M step and the E step does not change the assignment.
- This algorithm will eventually converge to a stable local minimum.
- Any permutation of the labels of a stable assignment is also a stable assignment.
- It is not guaranteed to converge to a global minimum.
- It is sensitive to the relative scale of the dimensions.
- Increasing k can always decrease error until k is the number of different examples.

EM Algorithm

- Used for soft clustering — examples are probabilistically in classes.
- k-valued random variable C

Model



Data

X_1	X_2	X_3	X_4
t	f	t	t
f	t	t	f
f	f	t	t
	$\dots$		

➪ Probabilities

$P(C)$
$P(X_1|C)$
$P(X_2|C)$
$P(X_3|C)$
$P(X_4|C)$

EM Algorithm

X_1	X_2	X_3	X_4	C	*count*
⋮	⋮	⋮	⋮	⋮	⋮
t	f	t	t	1	0.4
t	f	t	t	2	0.1
t	f	t	t	3	0.5
⋮	⋮	⋮	⋮	⋮	⋮

M-step

$P(C)$
$P(X_1|C)$
$P(X_2|C)$
$P(X_3|C)$
$P(X_4|C)$

E-step

EM Algorithm Overview

- Repeat the following two steps:
  - E-step give the expected number of data points for the unobserved variables based on the given probability distribution.
  - M-step infer the (maximum likelihood or maximum aposteriori probability) probabilities from the data.
- Start either with made-up data or made-up probabilities.
- EM will converge to a local maxima.

Augmented Data — E step

Suppose $k = 3$, and $dom(C) = \{1, 2, 3\}$.
$P(C = 1 | X_1 = t, X_2 = f, X_3 = t, X_4 = t) = 0.407$
$P(C = 2 | X_1 = t, X_2 = f, X_3 = t, X_4 = t) = 0.121$
$P(C = 3 | X_1 = t, X_2 = f, X_3 = t, X_4 = t) = 0.472$:

X_1	X_2	X_3	X_4	*Count*
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
t	f	t	t	100
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

$\longrightarrow$

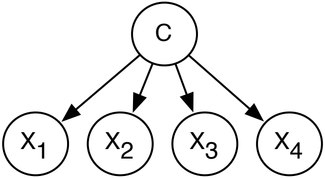
$\overbrace{\hspace{6em}}^{A[X_1, \ldots, X_4, C]}$

X_1	X_2	X_3	X_4	C	*Count*
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
t	f	t	t	1	40.7
t	f	t	t	2	12.1
t	f	t	t	3	47.2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

M step

X_1	X_2	X_3	X_4	C	$Count$
⋮	⋮	⋮	⋮	⋮	⋮
t	f	t	t	1	40.7
t	f	t	t	2	12.1
t	f	t	t	3	47.2
⋮	⋮	⋮	⋮	⋮	⋮

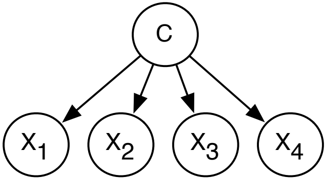
$\longrightarrow$



M step

X_1	X_2	X_3	X_4	C	$Count$
⋮	⋮	⋮	⋮	⋮	⋮
t	f	t	t	1	40.7
t	f	t	t	2	12.1
t	f	t	t	3	47.2
⋮	⋮	⋮	⋮	⋮	⋮

$\longrightarrow$



$$P(C{=}v_i) = \frac{\sum_{t \models C=v_i} Count(t)}{\sum_t Count(t)}$$

$$P(X_k = v_j | C{=}v_i) = \frac{\sum_{t \models C=v_i \wedge X_k=v_j} Count(t)}{\sum_{t \models C=v_i} Count(t)}$$

...perhaps including pseudo-counts