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 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 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.
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 $X_1, \ldots, X_n$
  - $X_j(e)$ 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 function \( \text{class} : E \rightarrow \{1, \ldots, k\} \).

\[ \text{class}(e) = i \] means $e$ is in class $i$.

The sum-of-squares error for class $i$ and prediction $\hat{X}_j(i)$ for feature $X_j$ is

\[ \sum_{e \in E} \sum_{j=1}^{n} (\hat{X}_j(\text{class}(e)) - X_j(e))^2. \]

Aim: find class and prediction function that minimize sum-of-squares error.
**k-means algorithm formalized**

- $E$ is the set of all examples
- the input features are $X_1, \ldots, X_n$
  - $X_j(e)$ 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

- function $\text{class} : E \rightarrow \{1, \ldots, k\}$.
  - $\text{class}(e) = i$ means $e$ is in class $i$.
- prediction $\hat{X}_j(i)$ for each feature $X_j$ and class $i$. 

\[ \sum_{e \in E} \sum_{j=1}^{n} (\hat{X}_j(\text{class}(e)) - X_j(e))^2 \]

Aim: find $\text{class}$ and prediction function that minimize sum-of-squares error.
k-means algorithm formalized

- $E$ is the set of all examples
- the input features are $X_1, \ldots, X_n$
  - $X_j(e)$ 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

- function $\text{class} : E \rightarrow \{1, \ldots, k\}$.
  - $\text{class}(e) = i$ means $e$ is in class $i$.
- prediction $\hat{X}_j(i)$ for each feature $X_j$ and class $i$.

The sum-of-squares error for $\text{class}$ and $\hat{X}_j(i)$ is
\textit{k-means algorithm formalized}

- $E$ is the set of all examples
- the input features are $X_1, \ldots, X_n$
  $X_j(e)$ 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

- function $\text{class} : E \rightarrow \{1, \ldots, k\}$.
  $\text{class}(e) = i$ means $e$ is in class $i$.
- prediction $\hat{X}_j(i)$ for each feature $X_j$ and class $i$.

The sum-of-squares error for $\text{class}$ and $\hat{X}_j(i)$ is

$$\sum_{e \in E} \sum_{j=1}^{n} \left( \hat{X}_j(\text{class}(e)) - X_j(e) \right)^2.$$ 

Aim: find $\text{class}$ and prediction function that minimize sum-of-squares error.
Minimizing the error

The sum-of-squares error for class and \( \hat{X}_j(i) \) is

\[
\sum_{e \in E} \sum_{j=1}^{n} \left( \hat{X}_j(\text{class}(e)) - X_j(e) \right)^2.
\]

- Given class, the \( \hat{X}_j \) that minimizes the sum-of-squares error is
Minimizing the error

The sum-of-squares error for class and $\hat{X}_j(i)$ is

$$\sum_{e \in E} \sum_{j=1}^{n} \left( \hat{X}_j(\text{class}(e)) - X_j(e) \right)^2.$$ 

- Given class, the $\hat{X}_j$ that minimizes the sum-of-squares error is the mean value of $X_j$ for that class.
Minimizing the error

The sum-of-squares error for class and $\hat{X}_j(i)$ is

$$\sum_{e \in E} \sum_{j=1}^{n} \left( \hat{X}_j(class(e)) - X_j(e) \right)^2.$$ 

- Given class, the $\hat{X}_j$ that minimizes the sum-of-squares error is the mean value of $X_j$ for that class.
- Given $\hat{X}_j$ for each $j$, each example can be assigned to the class that
Minimizing the error

The sum-of-squares error for class and $\hat{X}_j(i)$ is

$$\sum_{e \in E} \sum_{j=1}^{n} \left( \hat{X}_j(\text{class}(e)) - X_j(e) \right)^2.$$ 

- Given class, the $\hat{X}_j$ that minimizes the sum-of-squares error is the mean value of $X_j$ for that class.
- Given $\hat{X}_j$ for each $j$, 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 $X_j$,
  \[ \hat{X}_j(i) \leftarrow \frac{\sum_{e: \text{class}(e) = i} X_j(e)}{|\{e: \text{class}(e) = i\}|}, \]

- For each example $e$, assign $e$ to the class $i$ that minimizes
  \[ \sum_{j=1}^{n} \left( \hat{X}_j(i) - X_j(e) \right)^2. \]

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

- $cc[c]$ is the number of examples in class $c$,
- $fs[j, c]$ is the sum of the values for $X_j(e)$ for examples in class $c$.

Then define $pn(j, c)$, current estimate of $\hat{X}_j(c)$

$$pn(j, c) = \frac{fs[j, c]}{cc[c]}$$
Sufficient statistics:

- $cc[c]$ is the number of examples in class $c$,
- $fs[j, c]$ is the sum of the values for $X_j(e)$ for examples in class $c$.

Then define $pn(j, c)$, current estimate of $\hat{X}_j(c)$

$$pn(j, c) = \frac{fs[j, c]}{cc[c]}$$
**k-means algorithm**

Sufficient statistics:
- $cc[c]$ is the number of examples in class $c$,
- $fs[j, c]$ is the sum of the values for $X_j(e)$ for examples in class $c$.

Then define $pn(j, c)$, current estimate of $\hat{X}_j(c)$

$$pn(j, c) = \frac{fs[j, c]}{cc[c]}$$

$\text{class}(e) =$
**k-means algorithm**

Sufficient statistics:

- $cc[c]$ is the number of examples in class $c$,
- $fs[j, c]$ is the sum of the values for $X_j(e)$ for examples in class $c$.

Then define $pn(j, c)$, current estimate of $\hat{X}_j(c)$

$$pn(j, c) = \frac{fs[j, c]}{cc[c]}$$

$$class(e) = \arg \min_c \sum_{j=1}^n (pn(j, c) - X_j(e))^2$$

These can be updated in one pass through the training data.
procedure \textit{k-means}(Xs, Es, k)

2: Initialize \(fs\) and \(cc\) randomly (based on data)

3: \textbf{def} \(pn(j, c) = fs[j, c]/cc[c]\)

4: \textbf{def} \(\text{class}(e) = \arg\min_c \sum_{j=1}^n (pn(j, c) - X_j(e))^2\)

5: \textbf{repeat}

6: \(fsn\) and \(ccn\) initialized to be all zero

7: \textbf{for each} example \(e \in Es\) \textbf{do}

8: \(c := \text{class}(e)\)

9: \(ccn[c] += 1\)

10: \textbf{for each} feature \(X_j \in Xs\) \textbf{do}

11: \(fsn[j, c] += X_j(e)\)

12: \(stable := (fsn=fs) \text{ and } (ccn=cc)\)

13: \(fs := fsn\)

14: \(cc := ccn\)

15: \textbf{until} \(stable\)

16: \textbf{return} \(\text{class}, \ pn\)
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

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t$</td>
<td>$f$</td>
<td>$t$</td>
<td>$t$</td>
</tr>
<tr>
<td>$f$</td>
<td>$t$</td>
<td>$t$</td>
<td>$f$</td>
</tr>
<tr>
<td>$f$</td>
<td>$f$</td>
<td>$t$</td>
<td>$t$</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Probabilities

$P(C)$

$P(X_1 | C)$

$P(X_2 | C)$

$P(X_3 | C)$

$P(X_4 | C)$
EM Algorithm

\[
\begin{align*}
X_1 & \quad X_2 & \quad X_3 & \quad X_4 & \quad C & \quad \text{count} \\
& : & : & : & : & : \\
t & f & t & t & t & 1 \quad 0.4 \\
t & f & t & t & t & 2 \quad 0.1 \\
t & f & t & t & t & 3 \quad 0.5 \\
& : & : & : & : & : \\
\end{align*}
\]

M-step

\[
P(C) \\
P(X_1|C) \\
P(X_2|C) \\
P(X_3|C) \\
P(X_4|C)
\]

E-step

\[
\begin{align*}
X_1 & \quad X_2 & \quad X_3 & \quad X_4 & \quad C & \quad \text{count} \\
& : & : & : & : & : \\
t & f & t & t & t & 1 \quad 0.4 \\
t & f & t & t & t & 2 \quad 0.1 \\
t & f & t & t & t & 3 \quad 0.5 \\
& : & : & : & : & : \\
\end{align*}
\]

©D. Poole and A. Mackworth 2019

Artificial Intelligence, Lecture 10.2
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 a posteriori 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 $\text{dom}(C) = \{1, 2, 3\}$.

$P(C = 1|X_1 = t, X_2 = f, X_3 = t, X_4 = t) = 0.407$
$P(C = 2|X_1 = t, X_2 = f, X_3 = t, X_4 = t) = 0.121$
$P(C = 3|X_1 = t, X_2 = f, X_3 = t, X_4 = t) = 0.472$:

\[
A[X_1, \ldots, X_4, C]
\]

\[
\begin{array}{cccc|c|c}
X_1 & X_2 & X_3 & X_4 & \text{Count} \\
\hline
\vdots & \vdots & \vdots & \vdots & \vdots \\
t & f & t & t & 100 \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\end{array}
\]

\[
\begin{array}{ccccc|c|c}
X_1 & X_2 & X_3 & X_4 & C & \text{Count} \\
\hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
t & f & t & t & t & 1 & 40.7 \\
t & f & t & t & t & 2 & 12.1 \\
t & f & t & t & t & 3 & 47.2 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\end{array}
\]
M step

<table>
<thead>
<tr>
<th>X_1</th>
<th>X_2</th>
<th>X_3</th>
<th>X_4</th>
<th>C</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>f</td>
<td>t</td>
<td>t</td>
<td>1</td>
<td>40.7</td>
</tr>
<tr>
<td>t</td>
<td>f</td>
<td>t</td>
<td>t</td>
<td>2</td>
<td>12.1</td>
</tr>
<tr>
<td>t</td>
<td>f</td>
<td>t</td>
<td>t</td>
<td>3</td>
<td>47.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
P(C = c)
\]

\[
P(X_i = v | C = c)
\]
M step

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_4$</th>
<th>$C$</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td><em>t</em></td>
<td><em>f</em></td>
<td><em>t</em></td>
<td><em>t</em></td>
<td>1</td>
<td>40.7</td>
</tr>
<tr>
<td><em>t</em></td>
<td><em>f</em></td>
<td><em>t</em></td>
<td><em>t</em></td>
<td>2</td>
<td>12.1</td>
</tr>
<tr>
<td><em>t</em></td>
<td><em>f</em></td>
<td><em>t</em></td>
<td><em>t</em></td>
<td>3</td>
<td>47.2</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>

$P(C=c)$

$P(X_i = v | C=c)$
EM sufficient statistics

- $cc$, a $k$-valued array, $cc[c]$ is the sum of the counts for $\text{class}=c$.
- $fc$, a 3-dimensional array such that $fc[i, v, c]$, is the sum of the counts of the augmented examples $t$ with $X_i(t) = \text{val}$ and $\text{class}(t) = c$. 

$\text{P}(\text{class}=c) = cc[c] / \text{Es}$

$\text{P}(X_i(t) = v | \text{class}=c) = fc[i, v, c] / cc[c]$
EM sufficient statistics

- $cc$, a $k$-valued array, $cc[c]$ is the sum of the counts for $\text{class}=c$.

- $fc$, a 3-dimensional array such that $fc[i, v, c]$, is the sum of the counts of the augmented examples $t$ with $X_i(t) = \text{val}$ and $\text{class}(t) = c$.

- The probabilities can be computed by:
EM sufficient statistics

- $cc$, a $k$-valued array, $cc[c]$ is the sum of the counts for $class=c$.
- $fc$, a 3-dimensional array such that $fc[i, v, c]$, is the sum of the counts of the augmented examples $t$ with $X_i(t) = val$ and $class(t) = c$.
- The probabilities can be computed by:

$$P(C=c) = \frac{cc[c]}{|Es|}$$

$$P(X_i = v | C=c) = \frac{fc[i, v, c]}{cc[c]}$$
1: procedure EM(Xs, Es, k)
2:   cc[c] := 0; fc[i, v, c] := 0
3:   repeat
4:     cc_new[c] := 0; fc_new[i, v, c] := 0
5:     for each example ⟨v₁, ..., vₙ⟩ ∈ Es do
6:       for each c ∈ [1, k] do
7:         dc := P(C = c | X₁ = v₁, ..., Xₙ = vₙ)
8:         cc_new[c] := cc_new[c] + dc
9:       for each i ∈ [1, n] do
10:          fc_new[i, vᵢ, c] := fc_new[i, vᵢ, c] + dc
11:       stable := (cc ≈ cc_new) and (fc ≈ fc_new)
12:   cc := cc_new
13:   fc := fc_new
14: until stable
15: return cc, fc

©D. Poole and A. Mackworth 2019 Artifical Intelligence, Lecture 10.2 19 / 19