Expectation Maximisation (EM)

CS 486/686: Introduction to Artificial Intelligence
University of Waterloo
Incomplete Data

So far we have seen problems where

- Values of all attributes are known
- Learning is relatively easy

Many real-world problems have hidden variables

- Incomplete data
- Missing attribute values
Maximum Likelihood Learning

Learning of Bayes nets parameters

- $\Theta_{V=True, \ Par(V)=x} = P(V=True | \ Par(V)=x)$
- $\Theta_{V=True, \ Par(V)=x} = (#\text{Insts } V=True)/(\text{Total } #V=x)$

Assumes all attributes have values

- What if some values are missing?
Naïve Solutions

• Ignore examples with missing attribute values
  - What if all examples have missing attribute values?

• Ignore hidden variables
  - Model might become much more complex
a) Uses a Hidden Variable, simpler (fewer CPT parameters)
b) No Hidden Variable, complex (many CPT parameters)
“Direct” ML

Maximize likelihood directly where $E$ are the evidence variables and $Z$ are the hidden variables

$$h_{ML} = \arg \max_h P(E|h)$$

$$= \arg \max_h \sum_Z P(E, Z|h)$$

$$= \arg \max_h \sum_Z \prod_i \text{CPT}(V_i)$$

$$= \arg \max_h \log \sum_Z \prod_i \text{CPT}(V_i)$$
Expectation-Maximization (EM)

If we knew the missing values computing $h_{ML}$ is trivial

- Guess $h_{ML}$
- Iterate
  - **Expectation**: based on $h_{ML}$ compute expectation of (missing) values
  - **Maximization**: based on expected (missing) values compute new $h_{ML}$
Expectation-Maximization (EM)

Formally

- Approximate maximum likelihood
- Iteratively compute:

\[ h_{i+1} = \arg\max_h \sum Z P(Z|h_i, e) \log P(e, Z|h_i) \]

Expectation

Maximization
EM Derivation

\[
\log P(e|h) = \log \left( \frac{P(e, Z|h)}{P(Z|e, h)} \right)
\]

\[
= \log P(e, Z|h) - \log P(Z|e, h)
\]

\[
= \sum_z P(Z|e, h) \log P(e, Z|h) - \sum_z P(Z|e, h) \log P(Z|e, h)
\]

\[
\geq \sum_z P(Z|e, h) \log P(e, Z|h)
\]

EM finds a local maxima of \( \sum_z P(Z|e, h) \log P(e, Z|h) \)

which is a lower bound of \( \log P(e|h) \)
EM

Log inside can linearize the product

\[ h_{i+1} = \arg \max_h \sum_Z P(Z|h, e) \log P(e, Z|h) \]

\[ = \arg \max_h \sum_Z P(Z|h, e) \log \prod_j \text{CPT}_j \]

\[ = \arg \max_h \sum_Z P(Z|h, e) \sum_j \log \text{CPT}_j \]

Monotonic improvement of likelihood

\[ P(e|h_{i+1}) \geq P(e|h_i) \]
Example

• Assume we have two coins, A and B
• The probability of getting heads with A is $\theta_A$
• The probability of getting heads with B is $\theta_B$
• We want to find $\theta_A$ and $\theta_B$ by performing a number of trials

Example from S. Zafeiriohu, Advanced Statistical Machine Learning, Imperial College
Example

**Coin A** and **Coin B**

- H T T H H T H T H
- H H H T H H H H H H
- H T H H H H H T H H
- H T H T T T H H T T
- T H H H T H H H H T H

<table>
<thead>
<tr>
<th>Coin A</th>
<th>Coin B</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 H, 5 T</td>
<td></td>
</tr>
<tr>
<td>9 H, 1 T</td>
<td></td>
</tr>
<tr>
<td>8 H, 2 T</td>
<td></td>
</tr>
<tr>
<td>4 H, 6 T</td>
<td></td>
</tr>
<tr>
<td>7 H, 3 T</td>
<td></td>
</tr>
<tr>
<td><strong>24 H, 6 T</strong></td>
<td><strong>9 H, 11 T</strong></td>
</tr>
</tbody>
</table>
### Example

<table>
<thead>
<tr>
<th>Coin A</th>
<th>Coin B</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 H, 5 T</td>
<td>5 H, 5 T</td>
</tr>
<tr>
<td>9 H, 1 T</td>
<td>9 H, 1 T</td>
</tr>
<tr>
<td>8 H, 2 T</td>
<td>9 H, 1 T</td>
</tr>
<tr>
<td>4 H, 6 T</td>
<td>9 H, 11 T</td>
</tr>
<tr>
<td>24 H, 6 T</td>
<td>9 H, 11 T</td>
</tr>
</tbody>
</table>

\[ \theta_A = \frac{24}{24 + 6} = 0.8 \]

\[ \theta_B = \frac{9}{9 + 11} = 0.45 \]
Example

Now assume we do not know which coin was used in which trial (hidden variable)

• H T T T H H T H T H
• H H H H T H H H H H
• H T H H H H H T H H
• H T H T T T H H T T
• T H H H T H H H T H
Example

Initialization: $\theta_A^0 = 0.60$
$\theta_B^0 = 0.50$

E Step: Compute the Expected counts of Heads and Tails

**Trial 1: H T T T H H T H T H T H H T H**

\[
P(A|\text{Trial 1}) = \frac{P(\text{Trial 1}|A)P(A)}{\sum_{i \in \{A,B\}} P(\text{Trial 1}|i)P(i)} = 0.45
\]

\[
P(B|\text{Trial 1}) = \frac{P(\text{Trial 1}|B)P(B)}{\sum_{i \in \{A,B\}} P(\text{Trial 1}|i)P(i)} = 0.55
\]

<table>
<thead>
<tr>
<th>Coin A</th>
<th>Coin B</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2 H,</td>
<td>2.8 H,</td>
</tr>
<tr>
<td>2.2 T</td>
<td>2.8 T</td>
</tr>
</tbody>
</table>
## Example

- **H T T T H H T H H**
  
  \[(0.55 \text{ A}, 0.45 \text{ B})\]

- **H H H H T H H H H**
  
  \[(0.80 \text{ A}, 0.20 \text{ B})\]

- **H T H H H H H T H H**
  
  \[(0.73 \text{ A}, 0.27 \text{ A})\]

- **H T H T T T H H T T**
  
  \[(0.35 \text{ A}, 0.65 \text{ B})\]

- **T H H N T H H H T H**
  
  \[(0.65 \text{ A}, 0.35 \text{ B})\]

### Table

<table>
<thead>
<tr>
<th>Coin A</th>
<th>Coin B</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2H, 2.2T</td>
<td>2.8H, 2.8T</td>
</tr>
<tr>
<td>7.2H, 0.8T</td>
<td>1.8H, 0.2T</td>
</tr>
<tr>
<td>5.9H, 1.5T</td>
<td>2.1H, 0.5T</td>
</tr>
<tr>
<td>1.4H, 2.1T</td>
<td>2.6H, 3.9T</td>
</tr>
<tr>
<td>4.5H, 1.9T</td>
<td>2.5H, 1.1T</td>
</tr>
<tr>
<td><strong>21.3H, 8.6T</strong></td>
<td><strong>11.7H, 8.4T</strong></td>
</tr>
</tbody>
</table>
Example

M Step: Compute parameters based on expected counts

<table>
<thead>
<tr>
<th>Coin A</th>
<th>Coin B</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2H, 2.2T</td>
<td>2.8H, 2.8T</td>
</tr>
<tr>
<td>7.2H, 0.8T</td>
<td>1.8H, 0.2T</td>
</tr>
<tr>
<td>5.9H, 1.5T</td>
<td>2.1H, 0.5T</td>
</tr>
<tr>
<td>1.4H, 2.1T</td>
<td>2.6H, 3.9T</td>
</tr>
<tr>
<td>4.5H, 1.9T</td>
<td>2.5H, 1.1T</td>
</tr>
<tr>
<td>21.3H, 8.6T</td>
<td>11.7H, 8.4T</td>
</tr>
</tbody>
</table>

\[
\theta^1_A = \frac{21.3}{21.3 + 8.6} = 0.71 \\
\theta^1_B = \frac{11.7}{11.7 + 8.4} = 0.58
\]

Repeat

\[
\theta^{10}_A = 0.80 \\
\theta^{10}_B = 0.52
\]
EM: k-means Algorithm

**Input**
- Set of examples, $E$
- Input features $X_1, \ldots, X_n$
- $val(e, X_j) =$ value of feature $j$ for example $e$
- $k$ classes

**Output**
- Function $\text{class}: E \rightarrow \{1, \ldots, k\}$ where $\text{class}(e) = i$ means example $e$ belongs to class $i$
- Function $pval$ where $pval(i, X_j)$ is the predicted value of feature $X_j$ for each example in class $i$
k-means Algorithm

• Sum-of-squares error for class i and pval is

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

• Goal: Final \textit{class} and \textit{pval} that minimizes sum-of-squares error.
Minimizing the error

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

• Given \textit{class}, the \textit{pval} that minimizes sum-of-square error is the mean value for that class

• Given \textit{pval}, each example can be assigned to the \textit{class} that minimizes the error for that example
**k-means Algorithm**

- Randomly assign the examples to classes
- Repeat the following two steps until E step does not change the assignment of any example
  - **M**: For each class $i$ and feature $X_j$
    \[
pval(i, X_j) = \frac{\sum_{e: \text{class}(e) = i} \text{val}(e, X_j)}{|\{e : \text{class}(e) = i\}|}
    \]
  - **E**: For each example $e$, assign $e$ to the class that minimizes
    \[
    \sum_{j=1}^{n} (pval(\text{class}(e), X_j) - \text{val}(e, X_j))^2
    \]
k-means Example

- Data set: (X,Y) pairs
  - (0.7,5.1) (1.5,6), (2.1, 4.5), (2.4, 5.5), (3, 4.4), (3.5, 5), (4.5, 1.5), (5.2, 0.7), (5.3, 1.8), (6.2, 1.7), (6.7, 2.5), (8.5, 9.2), (9.1, 9.7), (9.5, 8.5)
Example Data
Random Assignment to Classes
Assign Each Example to Closest Mean
Reassign each example
Properties of k-means

• An assignment is stable if both M step and E step do not change the assignment
  - Algorithm will eventually converge to a stable local minimum
  - No guarantee that it will converge to a global minimum

• Increasing k can always decrease error until k is the number of different examples