Lecture 9c - Unsupervised Learning under Uncertainty

Jesse Hoey
School of Computer Science
University of Waterloo

March 13, 2020

Readings: Poole & Mackworth (2nd. Ed.) Chapt. 10.2, 10.3, 10.5
Incomplete Data

- So far:
  - values of all attributes are known
  - learning is easy

- But many real-world problems have **hidden** variables (aka **latent** variables)
  - Incomplete data
  - Values of some attributes missing

- Incomplete data $\rightarrow$ unsupervised learning
Recall: ML learning of Bayes net parameters for each variable $V$ with parents $pa(V)$, and each value those parents can take on $pa(V) = v$:

$$θ_{V=\text{true},pa(V)=v} = P(V = \text{true}|pa(V) = v)$$

so that the ML learning of $θ$ is:

$$θ_{V=\text{true},pa(V)=v} = \frac{\text{number with } (V = \text{true} \land pa(V) = v)}{\text{number with } pa(V) = V}$$

Can add pseudocounts as priors
But what if some variable values are missing?
1. Ignore hidden variables
   number of parameters shown (variables have 3 values):

2. Ignore records with missing values
   - does not work with true latent variables
You cannot just ignore missing data unless you know it is missing at random.

Often missing data is not missing at random, and the reason it is missing is correlated with something of interest.

For example: data in a clinical trial to test a drug may be missing because:

- the patient dies,
- the patient dropped out because of severe side effects,
- they dropped out because they were better, or
- the patient had to visit a sick relative.

— ignoring some of these may make the drug look better or worse than it is.

In general you need to model why data is missing.
Survivorship Bias

Bullet holes on planes returning from battle:
where should the extra armour be installed?

Abraham Wald (WWII)
“Direct” maximum likelihood

3. maximize likelihood directly
   Suppose $\mathbf{Z}$ is hidden and $\mathbf{E}$ is observable, with values $\mathbf{e}$

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

$$= \arg\max_h \left[ \sum_{\mathbf{Z}} P(\mathbf{e}, \mathbf{Z}|h) \right]$$

$$= \arg\max_h \left[ \sum_{\mathbf{Z}} \prod_{i=1}^n P(X_i|\text{parents}(X_i), h)_{\mathbf{E} = \mathbf{e}} \right]$$

$$= \arg\max_h \left[ \log \sum_{\mathbf{Z}} \prod_{i=1}^n P(X_i|\text{parents}(X_i), h)_{\mathbf{E} = \mathbf{e}} \right]$$

**Problem:** can’t push log inside the sum to linearize!
4. If we knew the missing values, computing $h_{ML}$ would be easy again!

**EM:**
A). Guess $h_{ML}$
B). iterate:
   - **expectation**: based on $h_{ML}$, compute expectation of missing values $P(Z|h_{ML}, e)$
   - **maximization**: based on expected missing values, compute new estimate of $h_{ML}$

5. Really simple version (e.g. K-means algorithm):
   - **expectation**: based on $h_{ML}$, compute most likely missing values $\arg\max_Z P(Z|h_{ML}, e)$
   - **maximization**: based on those missing values, you now have complete data, so compute new estimate of $h_{ML}$ using ML learning as before
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 target feature for each class
- an assignment of examples to classes

**Algorithm:**
1. pick $k$ means, one per class
2. iterate until means stop changing:
   a. assign examples to $k$ classes (e.g. as closest to current means)
   b. re-estimate $k$ means based on assignment
Expectation Maximization

- Approximate the maximum likelihood
- Start with a guess $h_0$
- Iteratively compute:

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

- can show that $P(e|h_{i+1}) \geq P(e|h_i)$ when computed with these two
Expectation Maximization

Can show that:

$$\log P(e|h) \geq \sum_Z P(Z|e,h) \log P(e,Z|h)$$

EM finds a local maximum of right side, which is a lower bound of the left side.

Log inside sum can linearize the product

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

$$= \arg \max_h \sum_Z P(Z|h_i, e) \log \prod_j CPT_j$$

$$= \arg \max_h \sum_Z P(Z|h_i, e) \sum_j \log CPT_j$$

EM monotonically improves the likelihood
Suppose $k = 3$, and $\text{dom}(C) = \{1, 2, 3\}$.

$P(C = 1|X_1 = t, X_2 = f, X_3 = t, X_4 = t) = 0.4$

$P(C = 2|X_1 = t, X_2 = f, X_3 = t, X_4 = t) = 0.1$

$P(C = 3|X_1 = t, X_2 = f, X_3 = t, X_4 = t) = 0.5$:

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_4$</th>
<th>$C$</th>
<th>$\text{val}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>:</td>
<td>:</td>
</tr>
<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>0.4</td>
</tr>
<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>2</td>
<td>0.1</td>
</tr>
<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>3</td>
<td>0.5</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>

Call this $A[X_1,\ldots, X_4, C]$ If there are $m$ copies of the tuple, the values in $A$ should be multiplied by $m$. 
Let $s$ be the number of tuples in data set. Compute the statistics for each feature and class:

$$M_i[X_i, C] = \sum_{X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n} A[X_1, \ldots, X_n, C]$$

$$M[C] = \sum_{X_i} M_i[X_i, C]$$

$M$ is unnormalized marginal. $M_i[X_i, C] = s \times P(X_i, C)$. 

Pseudo-counts can also be added.
Let $s$ be the number of tuples in data set.
Compute the statistics for each feature and class:

$$M_i[X_i, C] = \sum_{X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n} A[X_1, \ldots, X_n, C]$$

$$M[C] = \sum_{X_i} M_i[X_i, C]$$

$M$ is unnormalized marginal. $M_i[X_i, C] = s \times P(X_i, C)$.
Compute probabilities by normalizing:

$$P(X_i|C) = \frac{M_i[X_i, C]}{M[C]}$$

$$P(C) = \frac{M[C]}{s}$$

Pseudo-counts can also be added.
General Bayes Network EM

Suppose we have a dataset $\mathbf{e}$, where the $i^{th}$ data assigns a value $x_i$ to observed variables $\mathbf{X}$, leaving $\mathbf{Z}$ latent variables (with values $z_j$) unassigned, then

**Recall**: Bayes Net Maximum Likelihood (Complete data - $Z = \{\}$)

$$\theta_{V=\text{true}, pa(V)=v} = \frac{\text{number in } \mathbf{e} \text{ with } (V = \text{true} \land pa(V) = v)}{\text{number in } \mathbf{e} \text{ with } pa(V) = v}$$

**Now**: Bayes Net Expectation Maximization (incomplete data)

Start with some guess for $\theta$,

**E Step**: Compute weights for each data $x_i$ and latent variable(s) value(s) $z_j$ (using e.g. variable elimination)

$$w_{ij} = P(z_j | \theta, x_i)$$

**M Step**: Update parameters:

$$\theta_{V=\text{true},pa(V)=v} = \frac{\sum_{ij} w_{ij} | V = \text{true} \land pa(V) = v \text{ in } \{x_i, z_j\}}{\sum_{ij} w_{ij} | pa(V) = v \text{ in } \{x_i, z_j\}}$$
Belief network structure learning (I)

\[ P(model|data) = \frac{P(data|model) \times P(model)}{P(data)}. \]

- A model here is a belief network.
- A bigger network can always fit the data better.
- \( P(model) \) lets us encode a preference for smaller networks (e.g., using the description length).
- You can search over network structure looking for the most likely model.
Given a total ordering, can do independence tests to determine which features should be the parents.

XOR problem: just because features do not give information individually, does not mean they will not give information in combination.

Search over total orderings of variables.
Next:

- Planning with uncertainty (Poole & Mackworth (2nd. Ed.) chapter 9.1-9.3,9.5)
- Reinforcement Learning (Poole & Mackworth (2nd. Ed.) chapter 12.1,12.3-12.9)