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

Maximum Likelihood learning

ML learning of Bayes net parameters for each variable $V$ with parents $pa(V)$, and each value those parents can take on $pa(V) = v$:

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

so that the ML learning of $\theta$ is:

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

Can add pseudocounts as priors

But what if some variable values are missing?

Clustering / Unsupervised Learning

1. Ignore hidden variables

2. Ignore records with missing values
   - does not work with true **latent** variables

Missing Data

- 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)
3. maximize likelihood directly
Suppose \( Z \) is hidden and \( E \) is observable, with values \( e \)

\[
h_{ML} = \arg \max_h P(e|h)
\]

\[
= \arg \max_h \left[ \sum_Z P(e, Z|h) \right]
\]

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

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

Problem: can’t push log inside the sum to linearize!

**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 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

**Augmented Data Method — E step – Naive Bayes with 4 observables**

Suppose \( k = 3 \), and \( \text{dom}(C) = \{1, 2, 3\} \).

\[
P(C = 1|X_1 = t, X_2 = f, X_3 = t, X_4 = t) = 0.4
\]

\[
P(C = 2|X_1 = t, X_2 = f, X_3 = t, X_4 = t) = 0.1
\]

\[
P(C = 3|X_1 = t, X_2 = f, X_3 = t, X_4 = t) = 0.5
\]

\[
\begin{array}{cccc|c}
X_1 & X_2 & X_3 & X_4 & C \\
: & : & : & : & \hline
: & : & : & : & 0.4 \\
t & f & t & t & 0.1 \\
t & f & t & t & 0.5 \\
: & : & : & : & \hline
\end{array}
\]

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 \( m \).
M step

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. $M_i[X_i, C] = s \times P(X_i; C)$.

General Bayes Network EM

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

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

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

**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) $z_j$ (using e.g. variable elimination)

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

**M Step:** Update parameters:

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

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)