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

Maximum Likelihood learning

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

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

so that the ML learning of θ is:

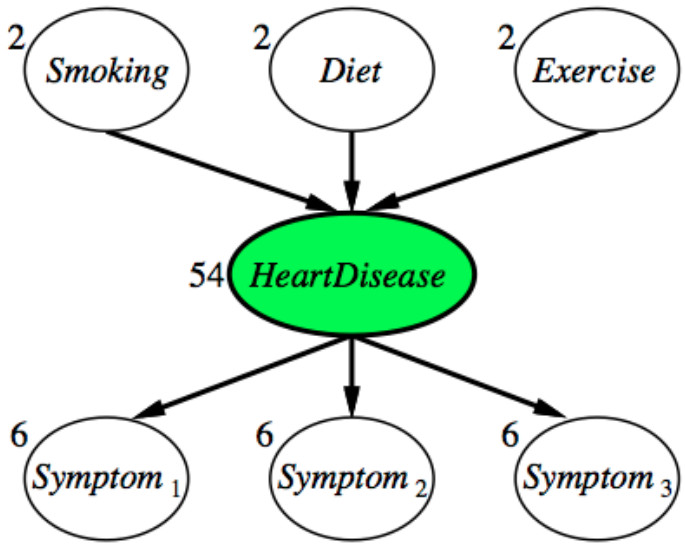
$$\theta_{V=true,pa(V)=\mathbf{v}} = \frac{\text{number with } (V = true \wedge pa(V) = \mathbf{v})}{\text{number with } pa(V) = \mathbf{V}}$$

Can add pseudocounts as priors
But what if some variable values are missing?

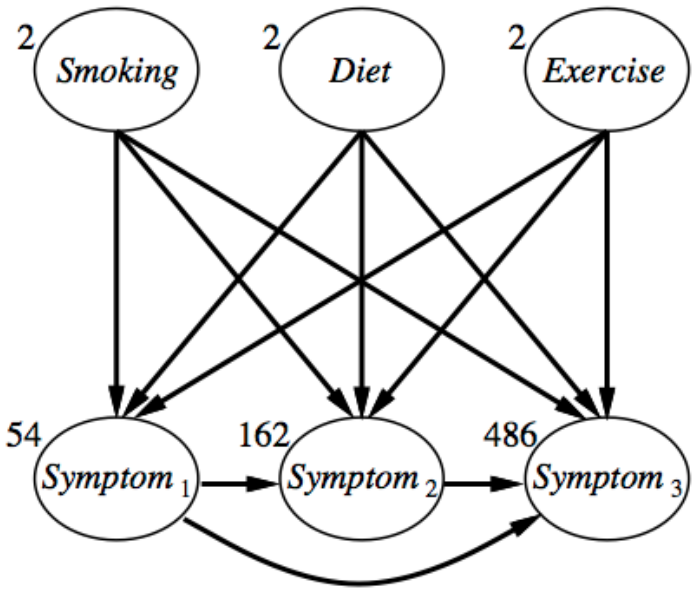
Clustering / Unsupervised Learning

1. Ignore hidden variables

   number of parameters shown (variables have 3 values):



(a)



(b)

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

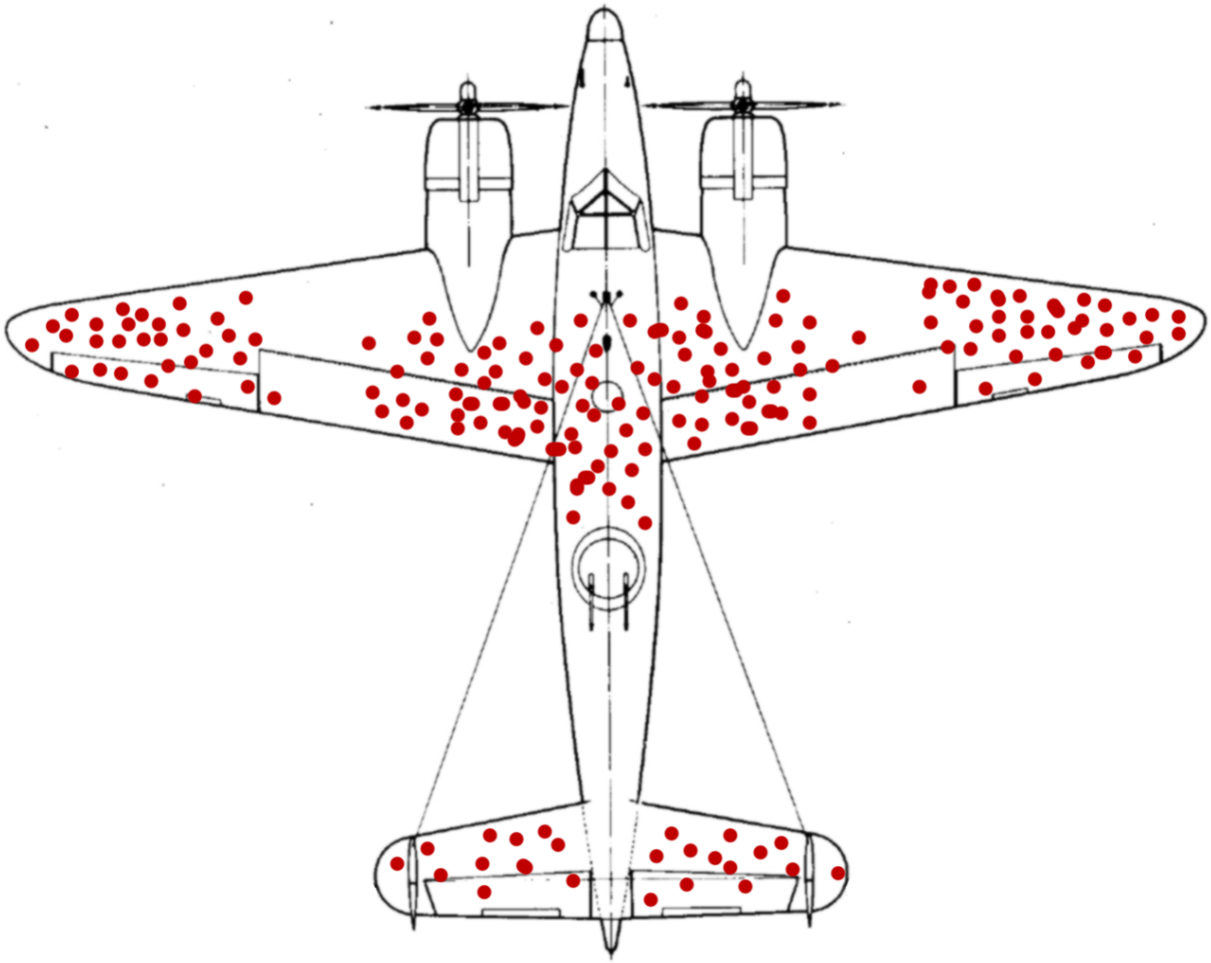
Missing Data

- 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)

"Direct" maximum likelihood

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

$$
\begin{aligned}
h_{ML} &= \arg\max_h P(\mathbf{e}|h) \\
&= \arg\max_h \left[\sum_Z P(\mathbf{e}, \mathbf{Z}|h) \right] \\
&= \arg\max_h \left[\sum_Z \prod_{i=1}^{n} P(X_i|parents(X_i), h)_{\mathbf{E}=\mathbf{e}} \right] \\
&= \arg\max_h \left[\log \sum_Z \prod_{i=1}^{n} P(X_i|parents(X_i), h)_{\mathbf{E}=\mathbf{e}} \right]
\end{aligned}
$$

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

Expectation-Maximization Algorithm

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 $P(\mathbf{Z}|h_{ML}, \mathbf{e})$
   - 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 $\arg\max_{\mathbf{Z}} P(\mathbf{Z}|h_{ML}, \mathbf{e})$
   - maximization: based on those missing values, you now have complete data, so compute new estimate of h_{ML} using ML learning as before

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

Expectation Maximization

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

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

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

Expectation Maximization

Can show that:

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

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

log inside sum can linearize the product

$$\begin{aligned} h_{i+1} &= \arg\max_h \sum_Z P(\mathbf{Z}|h_i, \mathbf{e}) \log P(\mathbf{e}, \mathbf{Z}|h) \\ &= \arg\max_h \sum_Z P(\mathbf{Z}|h_i, \mathbf{e}) \log \prod_j CPT_j \\ &= \arg\max_h \sum_Z P(\mathbf{Z}|h_i, \mathbf{e}) \sum_j \log CPT_j \end{aligned}$$

EM monotonically improves the likelihood

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

Suppose $k = 3$, and $dom(C) = \{1, 2, 3\}$.
$P(C = 1|X_1 = t, X_2 = f, X_3 = t, X_4 = t) = 0.4$
$P(C = 2|X_1 = t, X_2 = f, X_3 = t, X_4 = t) = 0.1$
$P(C = 3|X_1 = t, X_2 = f, X_3 = t, X_4 = t) = 0.5$:

X_1	X_2	X_3	X_4
⋮	⋮	⋮	⋮
t	f	t	t
⋮	⋮	⋮	⋮

$\longrightarrow$

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

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

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

$$P(X_i|C) = M_i[X_i, C]/M[C]$$

$$P(C) = M[C]/s$$

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 $\mathbf{x}_i$ to observed variables $\mathbf{X}$, leaving $\mathbf{Z}$ latent variables (with values $\mathbf{z}_j$) unassigned, then

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

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

Now: Bayes Net Expectation Maximization (incomplete data)

Start with some guess for θ,

E Step: Compute weights for each data $\mathbf{x}_i$ and latent variable(s) value(s) $\mathbf{z}_j$ (using e.g. variable elimination)

$$w_{ij} = P(\mathbf{z}_j | \theta, \mathbf{x}_i)$$

M Step: Update parameters:

$$\theta_{V=true,pa(V)=\mathbf{v}} = \frac{\sum_{ij} w_{ij} | V = true \wedge pa(V) = \mathbf{v} \text{ in } \{\mathbf{x}_i, \mathbf{z}_j\}}{\sum_{ij} w_{ij} | pa(V) = \mathbf{v} \text{ in } \{\mathbf{x}_i, \mathbf{z}_j\}}$$

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.

Belief network structure learning (II)

- 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)