Learning with Hidden Variables

November 18, 2021

Learning with Hidden Variables

This section is based on the 1997 paper *Adaptive Probabilistic Networks with Hidden Variables*, by John Binder, D. Koller, S. Russell, and K. Kanazawa, published in Machine Learning, 29, 213-244.

It poses (and provides one answer) to the question *"How can a probabilistic network be learned from data?"*.

In the following we assume a known network structure with hidden variables and that what we need to learn are the CP Tables.

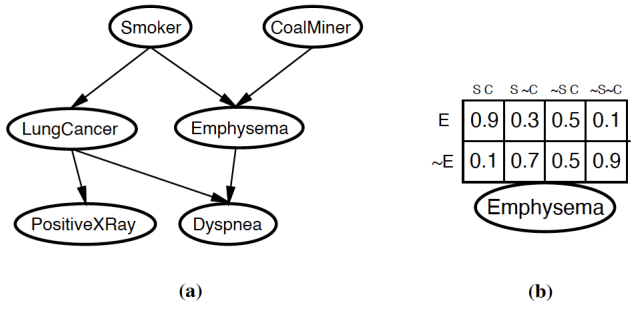

Figure 1: (a) A simple probabilistic network showing a proposed causal model. (b) A node with associated conditional probability table. The table gives the conditional probability of each possible value of the variable *Emphysema*, given each possible combination of values of the parent nodes *Smoker* and *CoalMiner*.

According to the table we have $P(Emphysema|Smoker, CoalMiner) = 0.9$, etc.

In general, there are several reasons why hidden variables are assumed:

1. Any particular variable may not be hidden in all the observed cases
2. The hidden variable might be one that we are interested in querying (several papers written in the '80s discuss this)

3. Networks with hidden variables can be *more compact* than the corresponding fully observable network (see Figure 2).

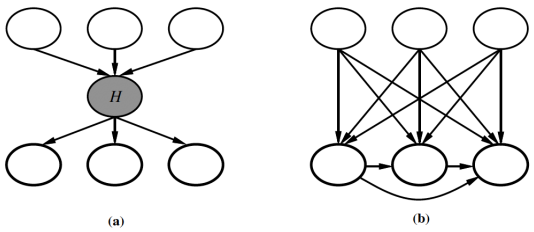

Figure 2: (a) A probabilistic network with a 2-valued hidden variable, labelled H; all other variables are 3-valued. Thus the network requires 45 independent parameters. (b) The corresponding fully observable network, following arc reversal and node removal. The network now requires 708 parameters.

4. With hidden variables it is possible to take advantage of local structure (if known)

   $\Longrightarrow$ more concise representation for the joint distribution on the observable variables

   $\Longrightarrow$ possible to learn from fewer examples.

TASK

GIVEN:

1 A network structure and initial (possibly randomly generated) values for the CPTs.

2 A set of cases $\mathbf{D} = \{D_1, \ldots, D_m\}$.

- Assume that the cases are generated independently from some underlying distribution.

3 Each data case provides the values of some subset of the variables;

4 This subset may differ from case to case.

OBJECTIVE: Find the CPT parameters w that *best model the data*, where w_{ijk} denotes a specific CPT entry: the probability that variable X_i takes on its jth possible value assignment given that its parents u_i take on their kth possible value assignment.

That is,

$$w_{ijk} = P(X_i = y_{ij} | Parents(Y_i) = \text{the list } u_{ik} \text{ of values})$$

e.g., if $X_i = Dyspnea$, then u_{ik} might be $\langle Emphysema = T, LungCancer = F \rangle$

To operationalize the phrase *best model the data*, assume that each possible setting of w is equally likely a priori, so that the maximum likelihood (ML) model is appropriate.

This means that the aim is to maximize $P_{\mathbf{w}}(\mathbf{D})$, that is, the probability assigned by the network to the observed data when the CPT parameters are set to $\mathbf{w}$.

- **Idea in Gradient Ascent**: Need to maximize a quantity. Use gradient - the vector of its partial derivatives with respect to its variables and adjust these in the direction of the gradient.

The Gradient Ascent Training for Bayesian networks

View the probability $P_{\mathbf{w}}(\mathbf{D})$ as a function of the CPT entries, that is the vector $\mathbf{w}$.

$\Longrightarrow$ Reduces the problem to one of finding the maximum of a multivariate nonlinear function.

Algorithms for solving this problem typically follow a path on a surface whose "coordinates" are the parameters and whose "height" is the value of the function, trying to get to the "highest" point on the surface.

It is easier[1] to maximize the log-likelihood function $\ln P_{\mathbf{w}}(\mathbf{D})$, since the two functions are monotonically related and hence maximizing one is equivalent to maximizing the other.

The simplest approach is *gradient ascent* (also known as "hill climbing").

- At each point $\mathbf{w}$, it computes $\nabla\mathbf{w}$, the gradient vector of partial derivatives with respect to the CPT entries.
- The algorithm then takes a small step in the direction of the gradient to the point $\mathbf{w} + \alpha\nabla\mathbf{w}$, where α is the step-size parameter. This simple algorithm converges to a local optimum for small enough α.

For our problem we need to modify the algorithm to account for the constraints on $\mathbf{w}$:

- Since $\mathbf{w}$ are probabilities, $w_{ijk} \in [0,1]$ and $\sum_j w_{ijk} = 1$, which is done by a renormalization of these values.
- The algorithm terminates when a local maximum is reached.

Derivation of the gradient formula

By independence of the data cases, we have

$$P_{\mathbf{w}}(\mathbf{D}) = \prod_{l=1}^{m} P_{\mathbf{w}}(D_l), \text{ by independence}$$

Hence

$$\ln P_{\mathbf{w}}(\mathbf{D}) = \sum_{l=1}^{m} \ln P_{\mathbf{w}}(D_l)$$

Therefore,

$$\begin{aligned} \frac{\partial \ln P_{\mathbf{w}}(\mathbf{D})}{\partial w_{ijk}} &= \sum_{l=1}^{m} \frac{\partial \ln P_{\mathbf{w}}(D_l)}{\partial w_{ijk}} \\ &= \sum_{l=1}^{m} \frac{1}{P_{\mathbf{w}}(D_l)} \times \frac{\partial P_{\mathbf{w}}(D_l)}{\partial w_{ijk}}. \end{aligned}$$

In order to get an expression in terms of information local to the parameter

[1] Because by independence, $P_{\mathbf{w}}(\mathbf{D})$ is a product of probabilities, so its ln is a sum of logs of these probabilities.

w_{ijk}, introduce $\mathbf{X}_i$ and $\mathbf{U}_i$ by averaging over their possible values:

$$\frac{1}{P_{\mathbf{w}}(D_l)} \times \frac{\partial P_{\mathbf{w}}(D_l)}{\partial w_{ijk}} = \text{averaging over } \mathbf{X}_i \text{ and } \mathbf{U}_i$$

$$= \frac{1}{P_{\mathbf{w}}(D_l)} \times \frac{\partial}{\partial w_{ijk}} \left(\sum_{j',k'} P_{\mathbf{w}}(D_l|x_{ij'}, \mathbf{u}_{ik'}) P_{\mathbf{w}}(x_{ij'}, \mathbf{u}_{ik'}) \right)$$

using the chain rule for probabilities

$$= \frac{1}{P_{\mathbf{w}}(D_l)} \times \frac{\partial}{\partial w_{ijk}} \left(\sum_{j',k'} P_{\mathbf{w}}(D_l|x_{ij'}, \mathbf{u}_{ik'}) P_{\mathbf{w}}(x_{ij'}|\mathbf{u}_{ik'}) P_{\mathbf{w}}(\mathbf{u}_{ik'}) \right)$$

$$= \frac{1}{P_{\mathbf{w}}(D_l)} \times \underbrace{\frac{\partial}{\partial w_{ijk}} \left(\sum_{j' \neq j, k' \neq k} P_{\mathbf{w}}(D_l|x_{ij'}, \mathbf{u}_{ik'}) P_{\mathbf{w}}(x_{ij'}|\mathbf{u}_{ik'}) P_{\mathbf{w}}(\mathbf{u}_{ik'}) \right)}_{\text{this is 0}}$$

$$\frac{1}{P_{\mathbf{w}}(D_l)} \times \frac{\partial}{\partial w_{ijk}} \left(P_{\mathbf{w}}(D_l|x_{ij}, \mathbf{u}_{ik}) P_{\mathbf{w}}(\mathbf{u}_{ik}) \underbrace{P_{\mathbf{w}}(x_{ij}|\mathbf{u}_{ik})}_{\text{this is } w_{ijk}} \right)$$

It is important for us that w_{ijk} appears only in linear form. Actually, it appears in only one one term in the summation, namely, where $j' = j$ and $k = k'$. Therefore, all other terms, i.e., where $j' \neq j$ and $k \neq k'$, when differentiated with respect to w_{ijk} will produce 0.

Hence,

$$\begin{aligned}
\frac{1}{P_{\mathbf{w}}(D_l)} \times \frac{\partial P_{\mathbf{w}}(D_l)}{\partial w_{ijk}} &= \frac{1}{P_{\mathbf{w}}(D_l)} \times P_{\mathbf{w}}(D_l|x_{ij}, \mathbf{u}_{ik}) P_{\mathbf{w}}(\mathbf{u}_{ik}) \\
&\quad \text{using Bayes Theorem, we obtain} \\
&= \frac{1}{P_{\mathbf{w}}(D_l) P_{\mathbf{w}}(x_{ij}, \mathbf{u}_{ik})} \times P_{\mathbf{w}}(x_{ij}, \mathbf{u}_{ik}|D_l) P_{\mathbf{w}}(D_l) P_{\mathbf{w}}(\mathbf{u}_{ik}) \\
&= \frac{P_{\mathbf{w}}(x_{ij}, \mathbf{u}_{ik}|D_l)}{P_{\mathbf{w}}(x_{ij}|\mathbf{u}_{ik})} \\
&= \frac{P_{\mathbf{w}}(x_{ij}, \mathbf{u}_{ik}|D_l)}{w_{ijk}}
\end{aligned}$$

The resulting algorithm for an Adaptive Probabilistic Network (APN) is then as follows:

The APN Algorithm.

APN(N, D, η): Given

(1) N: a (Bayesian) probabilistic network with CPT entries **w**

(2) **D**: a set of data cases

(3) η a small positive number (learning rate)

Initialize w randomly to positive values such that $\sum_j w_{ijk} = 1$

Repeat until $\Delta \mathbf{w} \approx \mathbf{0}$

$\Delta \mathbf{w} \leftarrow \mathbf{0}$

for each $D_l \in \mathbf{D}$

Set the evidence in N **from** D_l

For each variable i**, value** j**, and conditioning case** k

$$\Delta w_{ijk} \leftarrow \Delta w_{ijk} + \frac{P_{\mathbf{w}}(x_{ij}, \mathbf{u}_{ik} | D_l)}{w_{ijk}}$$

$\mathbf{w} \leftarrow \mathbf{w} + \eta \Delta \mathbf{w}$

Renormalize the w_{ijk} **to assure**

$\sum_j w_{ijk} = 1$

$0 \leq w_{ijk} \leq 1$
