Bayesian learning

Applied artificial intelligence (EDA132)
Lecture 06
2014-02-07
Elin A. Topp

Material based on course book, chapter 20,
and on Tom M. Mitchell, "Machine Learning", McGraw-Hill, 1997

Bayes' Rule

Bayes' Rule $P(a \mid b) = \dfrac{P(b \mid a)\, P(a)}{P(b)}$

or in distribution form:

$$\boldsymbol{P}(Y \mid X) = \frac{\boldsymbol{P}(X \mid Y)\, \boldsymbol{P}(Y)}{\boldsymbol{P}(X)} = \alpha\, \boldsymbol{P}(X \mid Y)\, \boldsymbol{P}(Y)$$

Useful for assessing *diagnostic* probability from *causal* probability

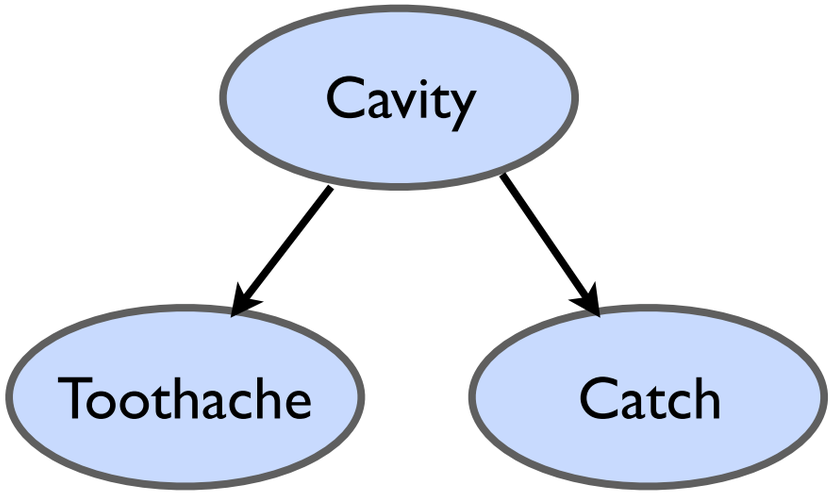
$$P(\mathit{Cause} \mid \mathit{Effect}) = \frac{P(\mathit{Effect} \mid \mathit{Cause})\, P(\mathit{Cause})}{P(\mathit{Effect})}$$

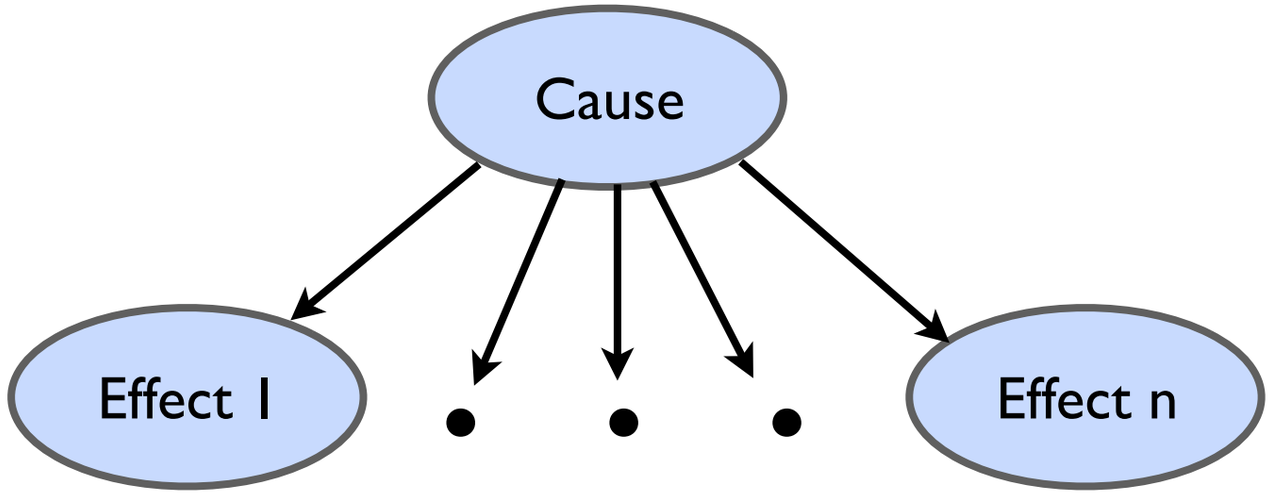
Bayes' Rule and conditional independence

$$\boldsymbol{P}(Cavity \mid toothache \wedge catch)$$
$$= \alpha\ \boldsymbol{P}(toothache \wedge catch \mid Cavity)\ \boldsymbol{P}(Cavity)$$
$$= \alpha\ \boldsymbol{P}(toothache \mid Cavity)\ \boldsymbol{P}(catch \mid Cavity)\ \boldsymbol{P}(Cavity)$$

An example of a *naive Bayes* model:

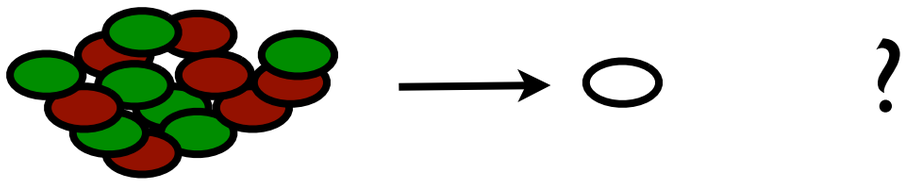
$$\boldsymbol{P}(Cause, Effect_1, \ldots, Effect_n) = \boldsymbol{P}(Cause) \prod_i \boldsymbol{P}(Effect_i \mid Cause)$$





The total number of parameters is *linear* in *n*

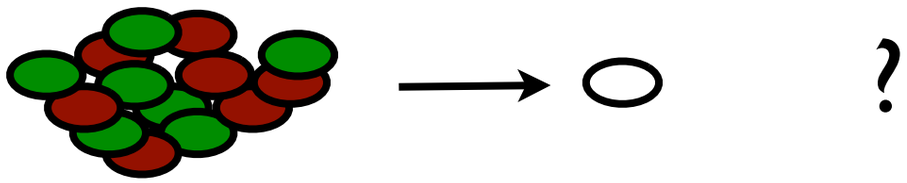
Predicting flavour



Hypotheses for which type of bag is given, with their *priors*:

h1:100% Cherry	P(h_1) = 0.1
h2: 75% Cherry, 25% Lime	P(h_2) = 0.2
h3: 50% Cherry, 50% Lime	P(h_3) = 0.4
h4: 25% Cherry, 75% Lime	P(h_4) = 0.2
h5: 100% Lime	P(h_5) = 0.1

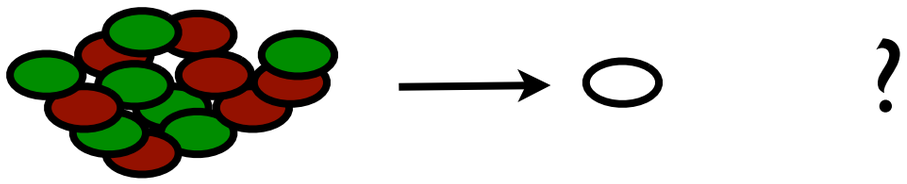
Maximum Likelihood



We can predict (probabilities) by maximizing the likelihood of having observed some particular data with the help of the ***Maximum Likelihood*** hypothesis:

$$h_{ML} = \underset{h}{argmax}\ P(D \mid h)$$

Maximum A Posteriori



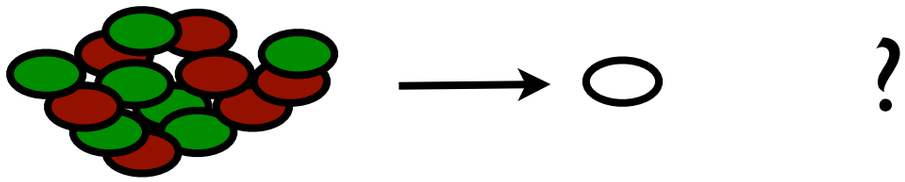
Finding the ***Maximum A Posteriori*** hypothesis:

$$h_{MAP} = \underset{h}{argmax}\ P(h \mid D)$$

Then predict by assuming the MAP-hypothesis (quite bold)

$$\mathbf{P}(X \mid \mathbf{D}) = P(X \mid h_{MAP})$$

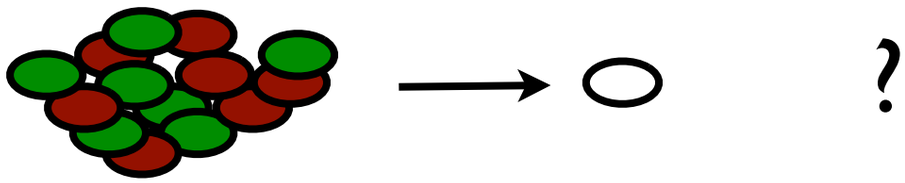
Optimal Bayes learner



Prediction for X, given some observations D = <d_0, d_1 .... d_n>

$\mathbf{P}(X \mid \mathbf{D}) = \sum_i \mathbf{P}(X \mid h_i)\, P(h_i \mid \mathbf{D})$ in first step, $P(h_i \mid \mathbf{D}) = P(h_i)$...

Learning from experience



Prediction for the first piece picked, assuming e.g., h_3, and no observations are made:

$$P(d\_0 = \text{Cherry} \mid h\_3) = P(d\_0 = \text{Lime} \mid h\_3) = 0.5$$

First piece is Lime flavour, now we know:

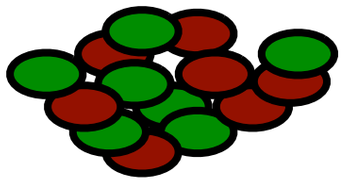
$$P(h\_1 \mid d\_0) = 0$$ (as $P(d\_0 \mid h\_1) = 0$), etc...

After 10 pieces that all turn out to be Lime, assuming that outcomes for d_i are *i.i.d. (independent and identically distributed)*:

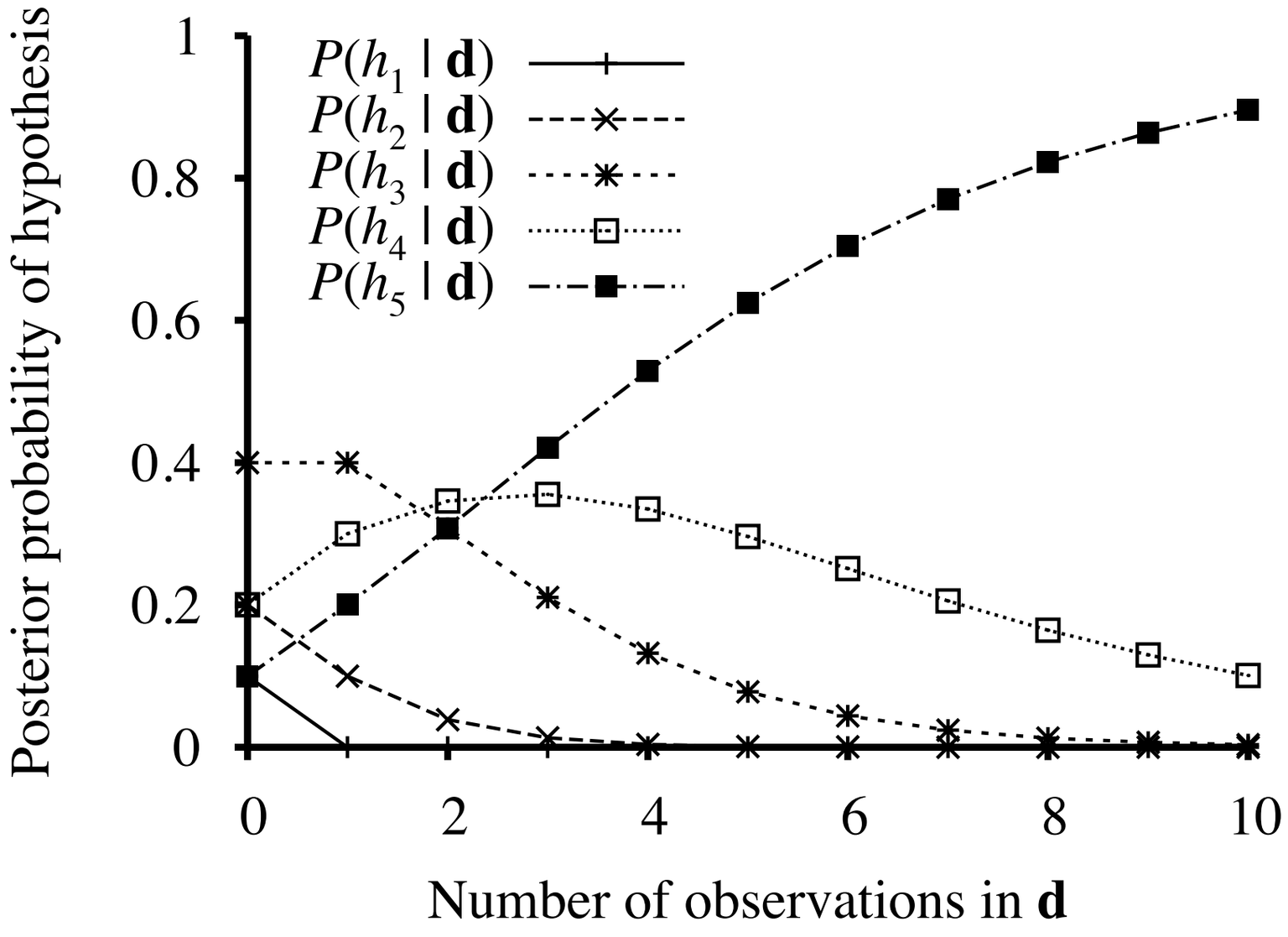
$$P(\mathbf{D} \mid h\_k) = \prod_i P(d\_i \mid h\_k)$$

$$P(h\_k \mid \mathbf{D}) = \mathbf{P}(\mathbf{D} \mid h\_k) P(h\_k) / \mathbf{P}(\mathbf{D}) = \alpha \ \mathbf{P}(\mathbf{D} \mid h\_k) P(h\_k)$$

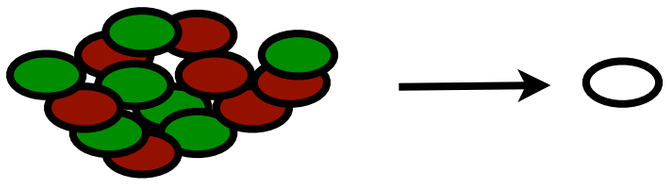
Posterior probabilities



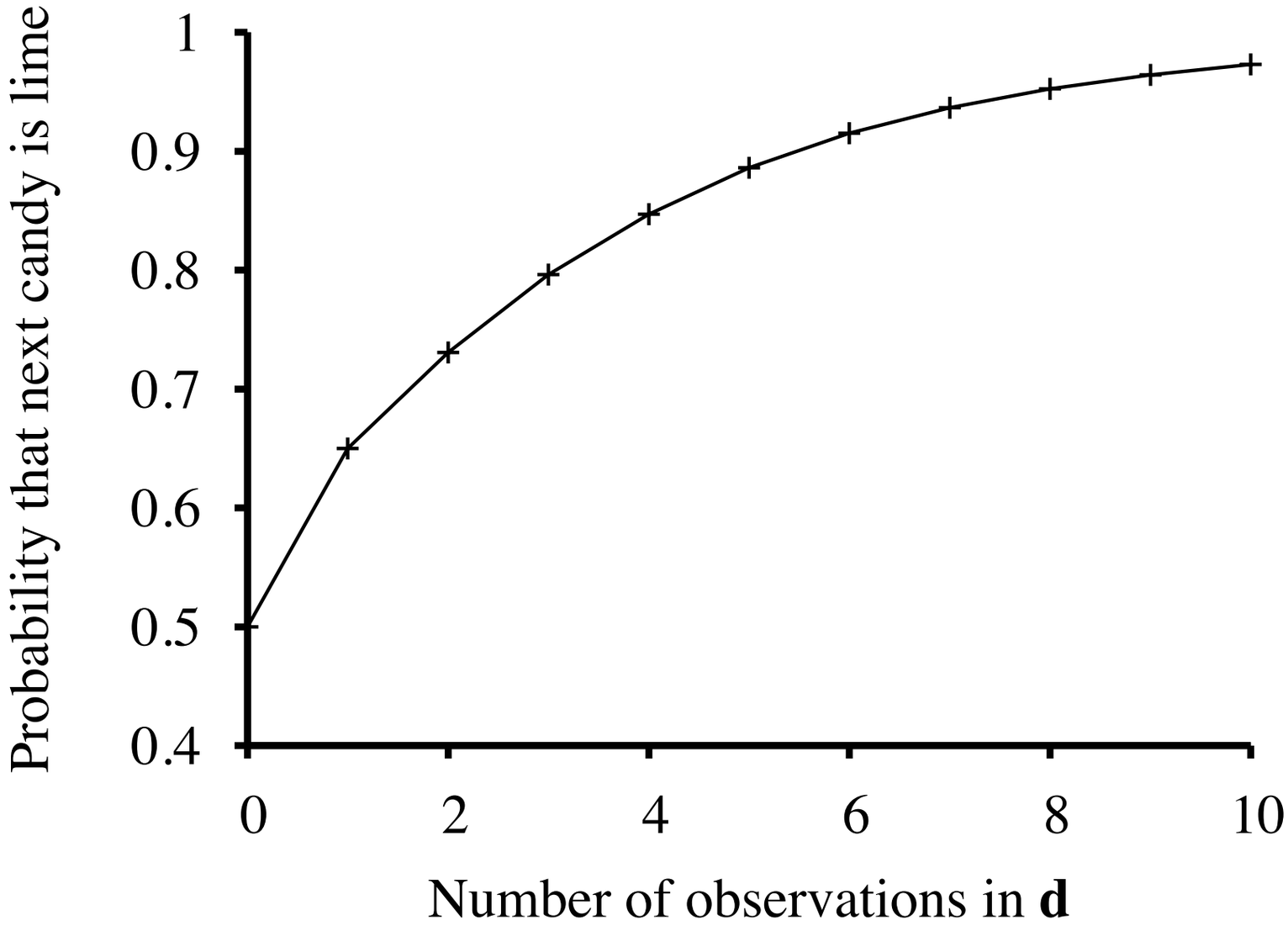
?



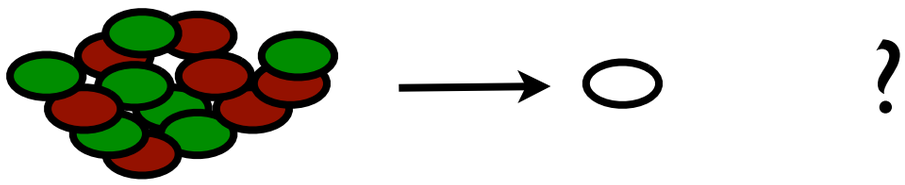
Prediction after sampling



?



Optimal learning vs estimating



Predict by assuming the MAP-hypothesis:

$\mathbf{P}(X \mid \mathbf{D}) = P(X \mid h_{MAP})$

P_h_{MAP}(d_4 = LIME | d_1 = d_2 = d_3 = Lime) = P(X | h_5) = 1

While the optimal classifier / learner predicts

P(d_4 = Lime | d_1=d_2=d_3=Lime) = ... = 0.76875

However, they will grow closer! Consequently, the MAP-learner should not be considered for small sets of training data!

The Gibbs Algorithm

Optimal Bayes Learner is costly, MAP-learner might be as well.

Gibbs algorithm (surprisingly well working under certain conditions regarding the a posteriori distribution for H):

1. Choose a hypothesis *h* from H *at random*, according to the posterior probability distribution over H (i.e., rule out "impossible" hypotheses)
2. Use *h* to predict the classification of the next instance x.

Naive Bayes classifier

Each instance with a value v_j from a fixed set V in a training set is described by several attributes $<a_1, \ldots, a_i, \ldots, a_n>$

Now we try to maximise:

$$v_{MAP} = \underset{v_j}{argmax}\ P(\ v_j \mid a_1, a_2, \ldots a_n)$$

$$= \underset{v_j}{argmax}\ \frac{P(a_1, a_2, \ldots a_n \mid v_j)\ P(v_j)}{P(a_1, a_2, \ldots a_n)}$$

$$= \underset{v_j}{argmax}\ P(\ a_1, a_2, \ldots a_n \mid v_j)\ P(v_j)$$

And (by assuming independence) end up with the Naive Bayes Classifier (corresponding in some sense to the MAP-hypothesis):

$$v_{NB} = \underset{v_j}{argmax}\ P(v_j) \prod_i P(\ a_i \mid v_j\ )$$

Classifying text

Our approach to representing arbitrary text is disturbingly simple: Given a text document, such as this paragraph, we define an attribute for each word position in the document and define the value of that attribute to be the English word found in that position. Thus, the current paragraph would be described by 111 attribute values, corresponding to the 111 word positions. The value of the first attribute is the word "our", the value of the second attribute is the word "approach", and so on. Notice that long text documents will require a larger number of attributes than short documents. As we shall see, this will not cause us any trouble. (*)

$$v_{NB} = \underset{v_j \in \{like, dislike\}}{argmax}\ P(v_j) \prod_i^{111} P(a_i \mid v_j) = P(v_j)\, P(a_1 = \text{“our”} \mid v_j) * .... * P(a_{111} = \text{“trouble”} \mid v_j)$$

(*)[Tom M. Mitchell, "Machine Learning", p 180]

Naive Bayes Classifier for text

Given a test person who classified 1000 text samples into the categories "like" and "dislike" (i.e., the target value set *V)* and those text samples (*Examples*), the text from the previous slide is to be classified with the help of the Naive Bayes Classifier. This algorithm (from Tom M. Mitchell, "Machine Learning", p 183) assumes (and learns) the *m-estimate* for $P(w_k \mid v_j)$, the term describing the probability that a randomly drawn word from a document in class v_j will be the word w_k.

LEARN_NAIVE_BAYES_TEXT(*Examples*, *V*)

/* learn probability terms $P(w_k \mid v_j)$ and the class prior probabilities $P(v_j)$ */

1. Collect all words, punctuation, and other tokens that occur in *Examples*
   - *Vocabulary* ⟵ the set of all distinct words and other tokens occurring in any text document from *Examples*
2. calculate the required $P(v_j)$ and $P(w_k \mid v_j)$ terms
   - $docs_j$ ⟵ the subset of documents from *Examples* for which the target value is v_j
   - $P(v_j) \longleftarrow | docs_j | / | Examples |$
   - $Text_j$ ⟵ a single document created by concatenating all members of $docs_j$
   - n ⟵ total number of distinct word positions in $Text_j$
   - for each word w_k in *Vocabulary*
     - n_k ⟵ number of times word w_k occurs in $Text_j$
     - $P(w_k \mid v_j) \longleftarrow (n_k + 1) / (n + | Vocabulary |)$ /* m-estimate */

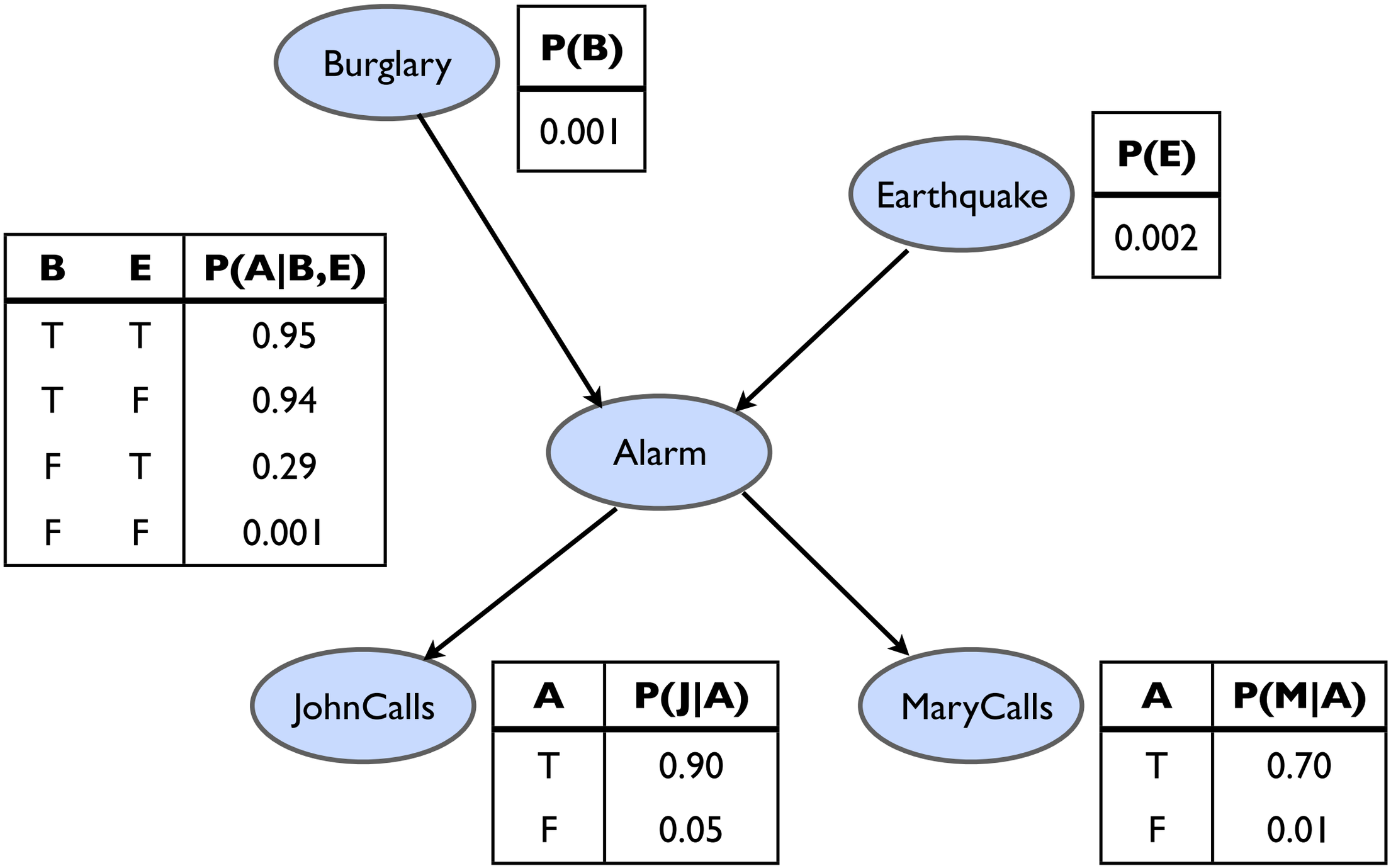
CLASSIFY_NAIVE_BAYES_TEXT(*Doc*)

/* Return the estimated target value for the document *Doc*. a_i denotes the word found in *i*th position within *Doc*.

- *positions* ⟵ all word positions in *Doc* that contain tokens found in *Vocabulary*
- Return v_{NB}, where

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} \; P(v_j) \prod_{i \in positions} P(a_i \mid v_j)$$

Bayesian Belief Networks (recap)



Learning Bayesian Belief Networks

Two issues:

Learning the CPTs given a suitable structure AND all variables are observable:

Estimate the CPTs as for a Naive Bayes Classifier / Learner (relatively easy)

Learning the CPTs given a network structure with only partially observable variables:

Corresponds to learning the weights of hidden units in a neural network (ascent gradient or EM)

Learning the network structure

Difficult. Bayesian scoring method for choosing among alternative networks.

Expectation maximization - EM algorithm

A situation with some variables being sometimes unobservable, sometimes observable is quite common.

Use the observations that are available to predict in cases where there is none.

Step 1: Estimate value for the hidden variable given some parameters (observed, initial...)

Step 2: Maximize parameters assuming this estimate

Learning useful stuff

Video from http://lasa.epfl.ch/videos/control.php

Summary

Maximum likelihood hypothesis and MAP-hypothesis / learning

Optimal Bayes learner / classifier

Gibbs algorithm

Naive Bayes classifier

Learning Bayesian Belief Networks

- EM algorithm

(Example: Mixtures of Gaussians -> Movie)