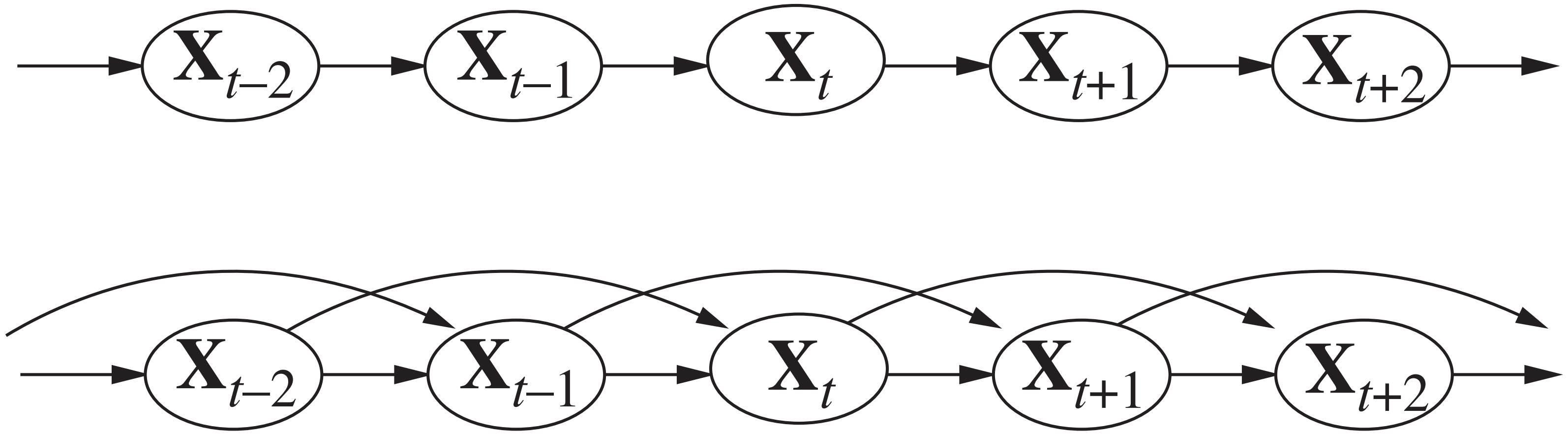
Bayesian learning
(with a recap of HMMs)

Applied artificial intelligence (EDAF70)
Lecture 10
2018-02-16
Elin A. Topp

Material based on course book, chapters 14.1-3, 20,
and on Tom M. Mitchell, "Machine Learning", McGraw-Hill, 1997

The Markov assumption

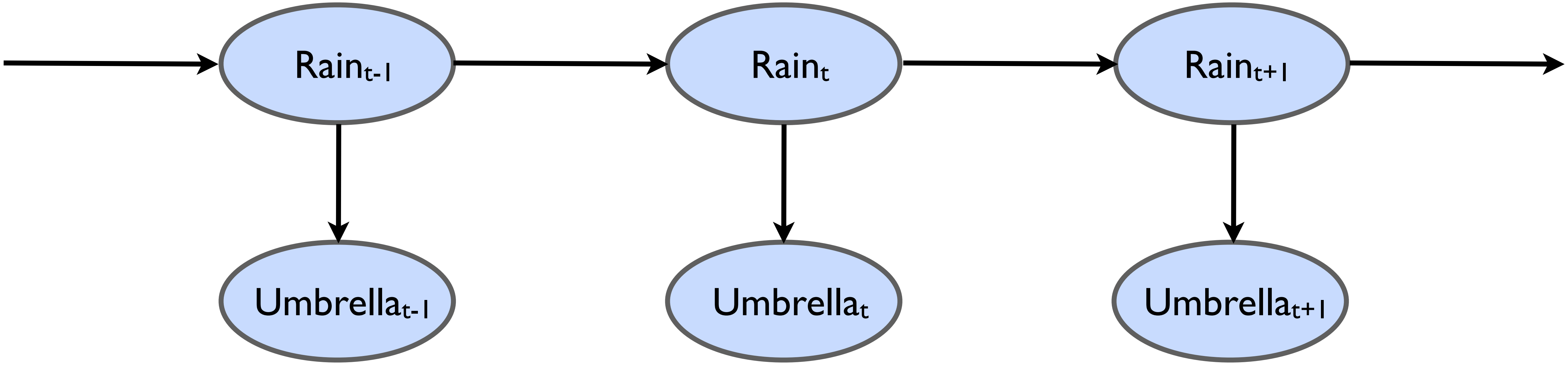
A process is *Markov* (i.e., complies with the Markov assumption), when any given state $\boldsymbol{X}_t$ depends only on a *finite and fixed number of previous states.*



A first-order Markov chain as Bayesian network

R_{t-1}	$P(R_t \mid R_{t-1})$
T	0.7
F	0.3

“cause” / state



“effect” / evidence

R_t	$P(U_t \mid R_t)$
T	0.9
F	0.2

“HMM”
Hidden Markov models

A specific class of models (sensor and transition) to be plugged into algorithms for filtering, predicting, learning - which makes the algorithms more specific as well!

Main idea:

The state is represented by a *single discrete random variable*, taking on values that represent the (all) possible states of the world.

Complex states, e.g., the location and the heading of a robot in a grid world can be merged into one variable; the possible values are then all possible tuples of the values for each original “single” variable.

“HMM”
State transition and sensor model

We get the following notation:

X_t the state at time t, taking on values $1 \dots S$, with S the number of possible states / values.

E_t the observation at time t

The ***transition*** model $P(X_t \mid X_{t-1})$ is then expressed as S x S matrix $\boldsymbol{T}$:

$$\boldsymbol{T}_{ij} = P(X_t = j \mid X_{t-1} = i) \text{ in time step } t$$

The ***sensor*** model for the corresponding observations depending on the current state, i.e., $P(e_t \mid X_t = i)$ is then expressed as S x S diagonal matrix $\boldsymbol{O}$ in time step t with

$$\boldsymbol{O}_{\boldsymbol{e\_t}ij} = P(e_t \mid X_t = i) \quad \text{for } i = j \qquad \text{and}$$

$$\boldsymbol{O}_{\boldsymbol{e\_t}ij} = 0 \qquad \text{for } i \neq j$$

Forward-backward equations as matrix-vector operations

Forward-equation (recap)

$$\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1}) = f(\mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t}), \mathbf{e}_{t+1}) = \mathbf{f}_{1:t+1} = \alpha\ \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}) \sum_{\mathbf{x}_t} \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t)\, P(\mathbf{x}_t \mid \mathbf{e}_{1:t})$$

becomes $\mathbf{f}_{1:t+1} = \alpha\ \mathbf{O}_{t+1} \mathbf{T}^{T} \mathbf{f}_{1:t}$

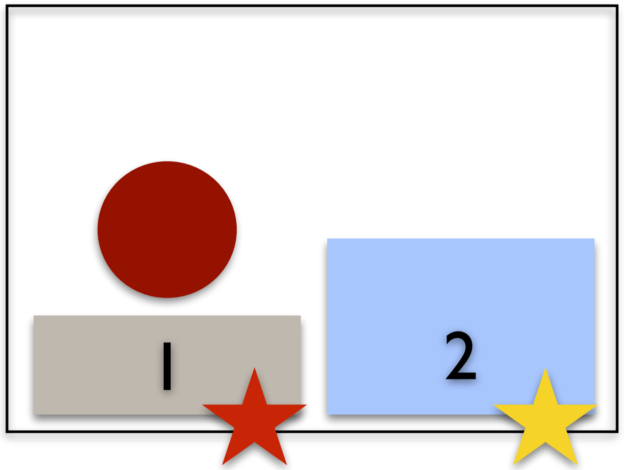
Backward-equation (recap)

$$\mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k) = \mathbf{b}_{k+1:t} = \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1} \mid \mathbf{x}_{k+1})\, P(\mathbf{e}_{k+2:t} \mid \mathbf{x}_{k+1})\, \mathbf{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k)$$

becomes $\mathbf{b}_{k+1:t} = \mathbf{T}\mathbf{O}_{k+1}\, \mathbf{b}_{k+2:t}$

Forward-Backward-equation is then still $\alpha\ \mathbf{f}_{1:k} \times \mathbf{b}_{k+1:t}$

Example matrix setup for a two-state world and three sensor readings



Ball behaviour:

P(step t: ball in 1 | step t-1: ball in 1) = 0.7

P(step t: ball in 2 | step t-1: ball in 1) = 0.3

P(step t: ball in 1 | step t-1: ball in 2) = 0.4

P(step t: ball in 2 | step t-1: ball in 2) = 0.6

Sensor correct:
"red" in state 1, "yellow" in state 2

P(sensor correct) = 0.8

P(sensor incorrect) = 0.15

P(sensor fails) = 0.05

$$T = \begin{pmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{pmatrix} \quad O_r = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 0.15 \end{pmatrix} \quad O_y = \begin{pmatrix} 0.15 & 0.0 \\ 0.0 & 0.8 \end{pmatrix} \quad O_f = \begin{pmatrix} 0.05 & 0.0 \\ 0.0 & 0.05 \end{pmatrix}$$

forward filtering with $\boldsymbol{f}_{1:0} = \mathbb{P}(\boldsymbol{X}_0)$ becomes then: $\boldsymbol{f}_{1:t+1} = \alpha \; \mathbf{O}_{t+1} \, \boldsymbol{T}^T \boldsymbol{f}_{1:t}$

Inference in temporal models - what can we use all this for?

- **Filtering**: Finding the **belief state**, or doing **state estimation**, i.e., computing the posterior distribution over the *most recent state*, using evidence up to this point:
  $\mathbb{P}(\boldsymbol{X}_t \mid \mathbf{e}_{1:t})$

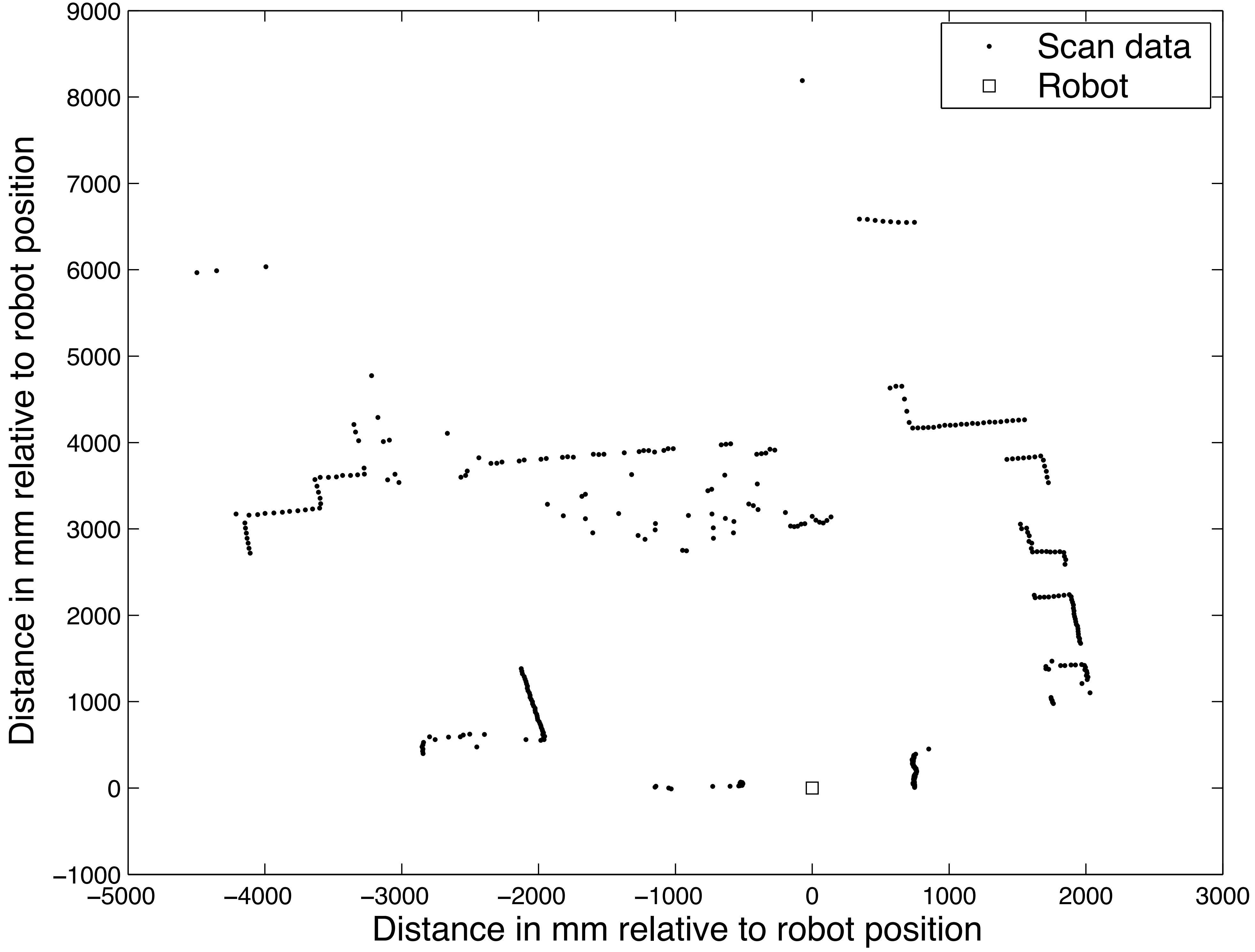
- **Predicting**: Computing the posterior over a *future* state, using evidence up to this point: $\mathbb{P}(\boldsymbol{X}_{t+k} \mid \mathbf{e}_{1:t})$ for some $k>0$ (can be used to evaluate course of action based on predicted outcome)

- **Smoothing**: Computing the posterior over a past state, i.e., understand the past, given information up to this point: $\mathbb{P}(\boldsymbol{X}_k \mid \mathbf{e}_{1:t})$ for some k with $0 \leq k < t$

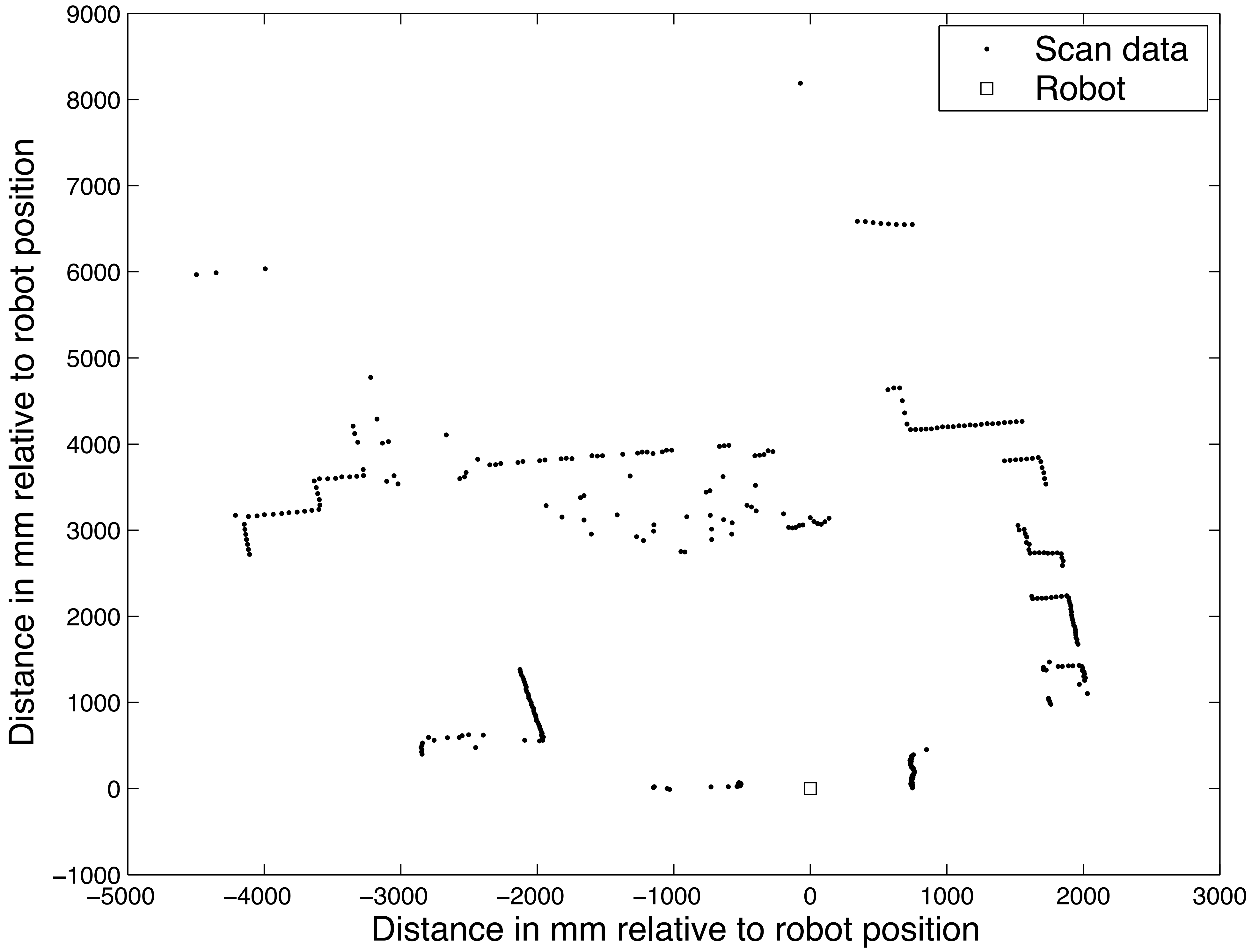
- (**Explaining**: Find the best explanation for a series of observations, i.e., computing
  $argmax_{\mathbf{x}_{1:t}} P(\mathbf{x}_{1:t} \mid \mathbf{e}_{1:t})$)

- **Learning**: If sensor and / or transition model are not known, they can be learned from observations (by-product of inference in Bayesian network - both static or dynamic). Inference gives estimates, estimates are used to update the model, updated models provide new estimates (by inference). Iterate until converging - and you have an instance of the EM-algorithm.

A robot’s view of the world...



A robot’s view of the world...



Which combination of point group features corresponds to person-leg, which to furniture?

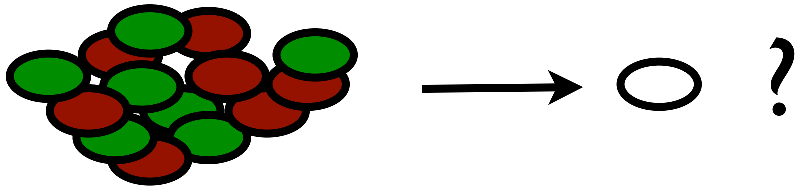
Bayesian learning.

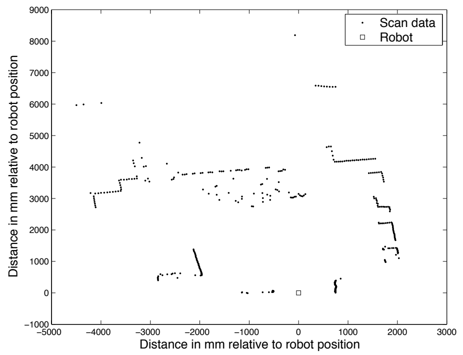
We want to classify / categorize / label new observations based on experience

More general: We want to *predict* and *explain* based on (limited) experience, to find categories / labels for observations or even the model for "how things work" (transition models, sensor models) *given a series of (explained) observations*.

Limitation of this lecture: Reinforcement learning, which also builds upon probabilistic methods, is not discussed.

Predicting the next outcome



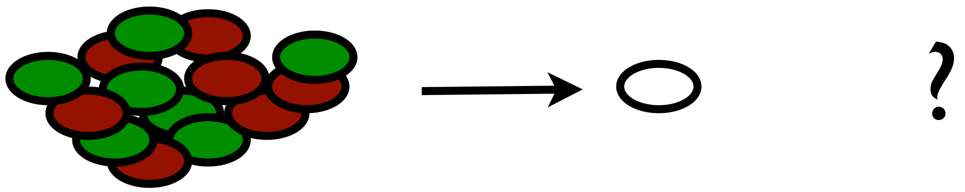


Candy bags with different percentages of flavours "lime" and "cherry".
A bag is opened, you can take the candies, but the label of the bag is gone. Which type was it?
And, more interestingly, what will the next candy flavour be, if I pick one at random?

Hypotheses for types of pattern collection (i.e., images from a certain situation) are still available, with their *priors*:

h_1: 100% Cherry	$P(h_1) = 0.1$
h_2: 75% Cherry, 25% Lime	$P(h_2) = 0.2$
h_3: 50% Cherry, 50% Lime	$P(h_3) = 0.4$
h_4: 25% Cherry, 75% Lime	$P(h_4) = 0.2$
h_5: 100% Lime	$P(h_5) = 0.1$

Maximum Likelihood

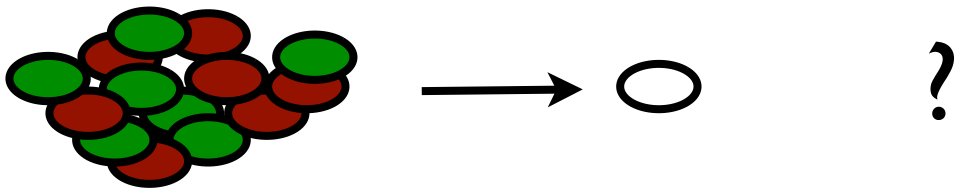


We can predict (probabilities) by maximizing the likelihood of having observed some particular data with the help of the ***Maximum Likelihood*** hypothesis:

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

… which is a strong simplification disregarding the priors…

“Maximum A Posteriori” - MAP



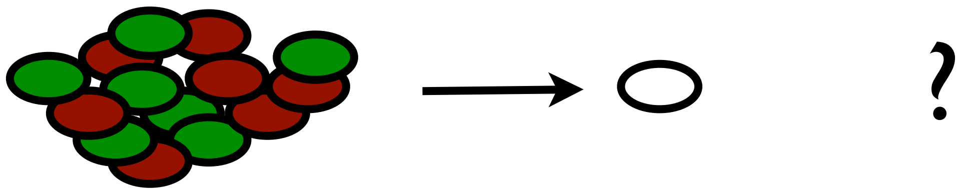
Finding the slightly more sophisticated ***Maximum A Posteriori*** hypothesis:

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

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

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

Optimal Bayes learner



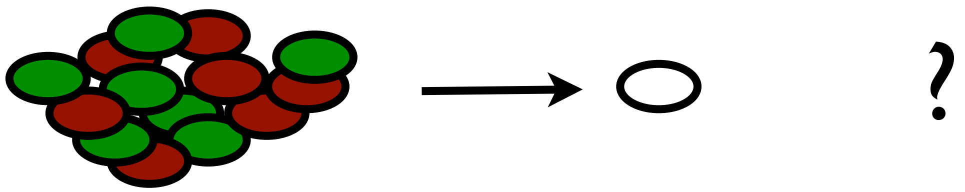
Prediction for X, given some observations **D** = <d_0, d_1 .... d_n>

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

For comparison (look at the prediction step in forward filtering):

$$\mathbb{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1}) = \alpha\ \ \mathbb{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1})\ \sum_{\mathbf{x}_t} \mathbb{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t)\ P(\mathbf{x}_t \mid \mathbf{e}_{1:t})$$

Learning from experience



Prediction for the first pattern 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 candy is of type *Lime*, now we know:

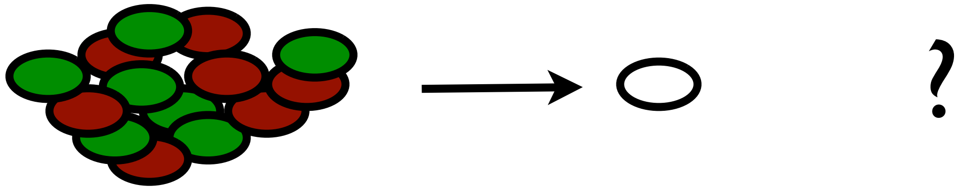
$P(\, h_1 \mid d_0) = 0$ (as $P(\, d_0 \mid h_1) = 0$), etc...

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

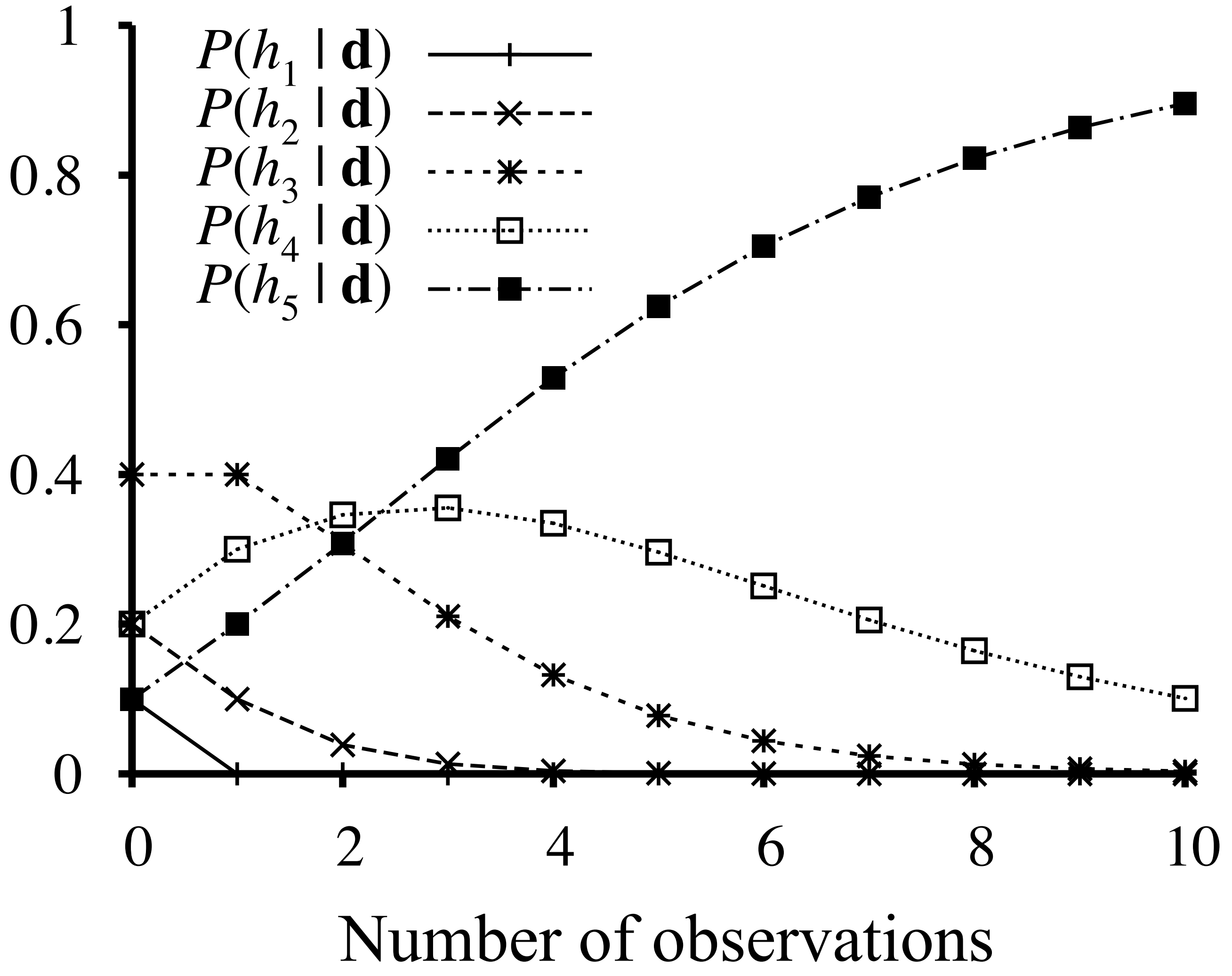
$$P(\, \mathbf{D} \mid h_k) = \prod_i P(\, d_i \mid h_k)$$

$$\mathbb{P}(\, h_k \mid \mathbf{D}) = \mathbb{P}(\, \mathbf{D} \mid h_k)\, P(\, h_k) \,/\, \mathbb{P}(\, \mathbf{D}) = \alpha\ \ \mathbb{P}(\, \mathbf{D} \mid h_k)\, P(\, h_k)$$

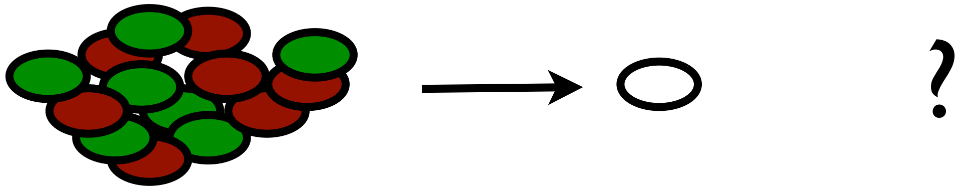
Posterior probabilities

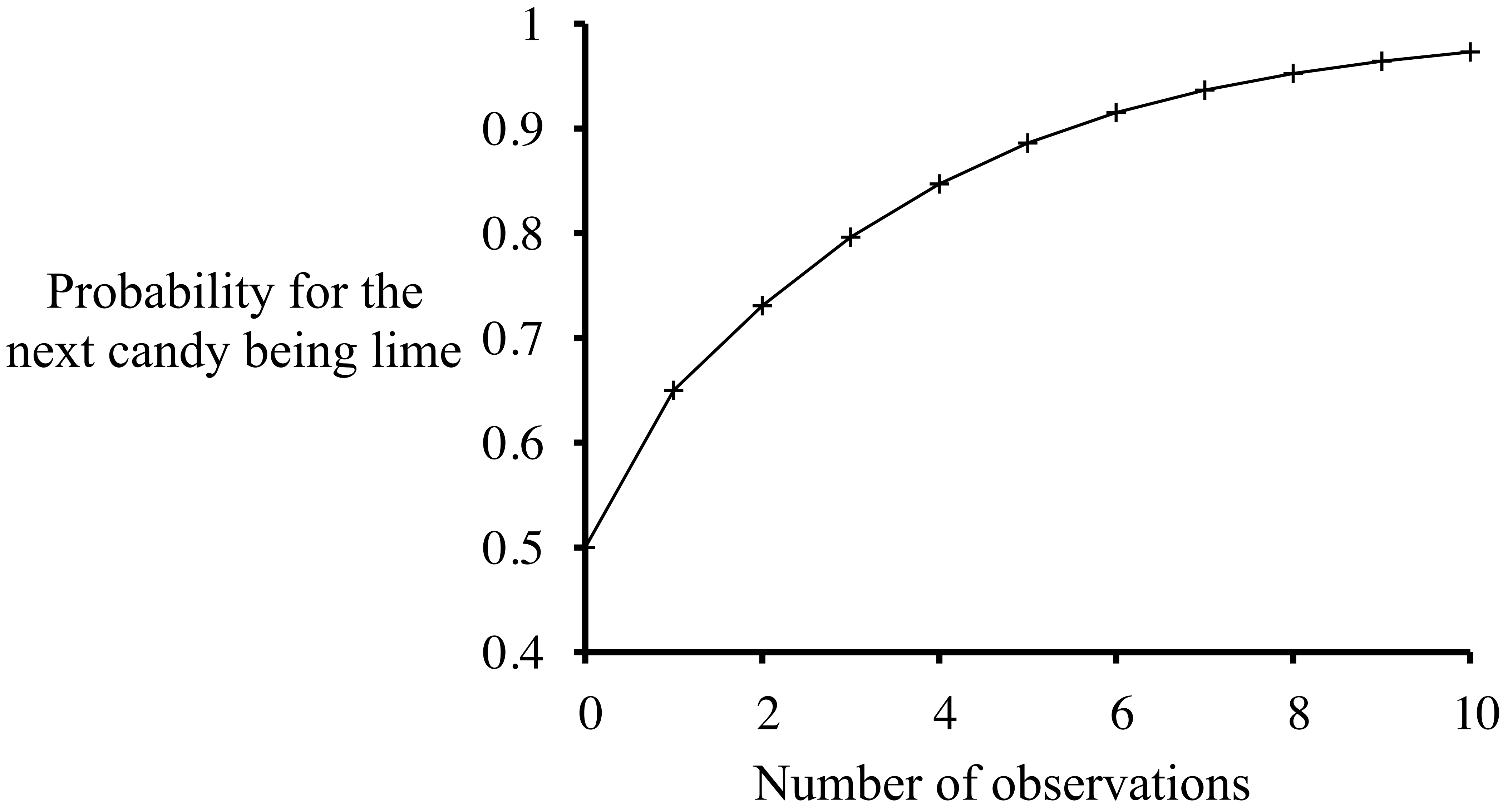


Posterior probability for hypothesis h_k after *i* observations

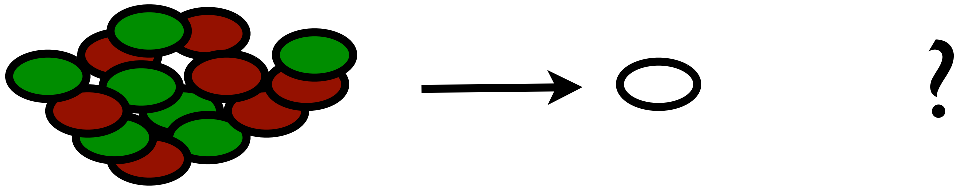


Prediction after sampling, OBC





Optimal learning vs MAP-estimating



Predict by assuming the MAP-hypothesis:

$\mathbb{P}(X \mid \mathbf{D}) = P(X \mid h_{MAP})$ with $h_{MAP} = \underset{h}{argmax}\, P(h \mid D)$

i.e., $P\_h_{MAP}(d_4 = \mathit{Lime} \mid d_1 = d_2 = d_3 = \mathit{Lime}) = P(X \mid h_5) = 1$

While the optimal classifier / learner predicts

$P(d_4 = \mathit{Lime} \mid d_1 = d_2 = d_3 = \mathit{Lime}) = \ldots = 0.7961$

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.

Bayes' Rule

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

or in distribution form:

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

Useful for assessing *diagnostic* probability from *causal* probability
- assume hypothesis / class as causing the observations / features

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

And, if independence (at least conditional such) can be assumed:

Naive Bayes model: $\mathbb{P}(Cause, Effect_1, ..., Effect_n) = \mathbb{P}(Cause) \prod_i \mathbb{P}(Effect_i \mid Cause)$

Naive Bayes classifier

Each instance (pattern) with a value v_j from a fixed set V (= {furniture, person}) in a training set (all patterns registered and annotated) is described by several attributes $<a_1, \dots, a_i, \dots, a_n>$ (e.g., number of laser data points, curvature of the "arc", distance from first to last point)

Now we try to maximise:

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

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

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

And (by assuming independence) end up with the Naive Bayes Classifier (corresponding to the MAP-hypothesis, if the observations are seen as features):

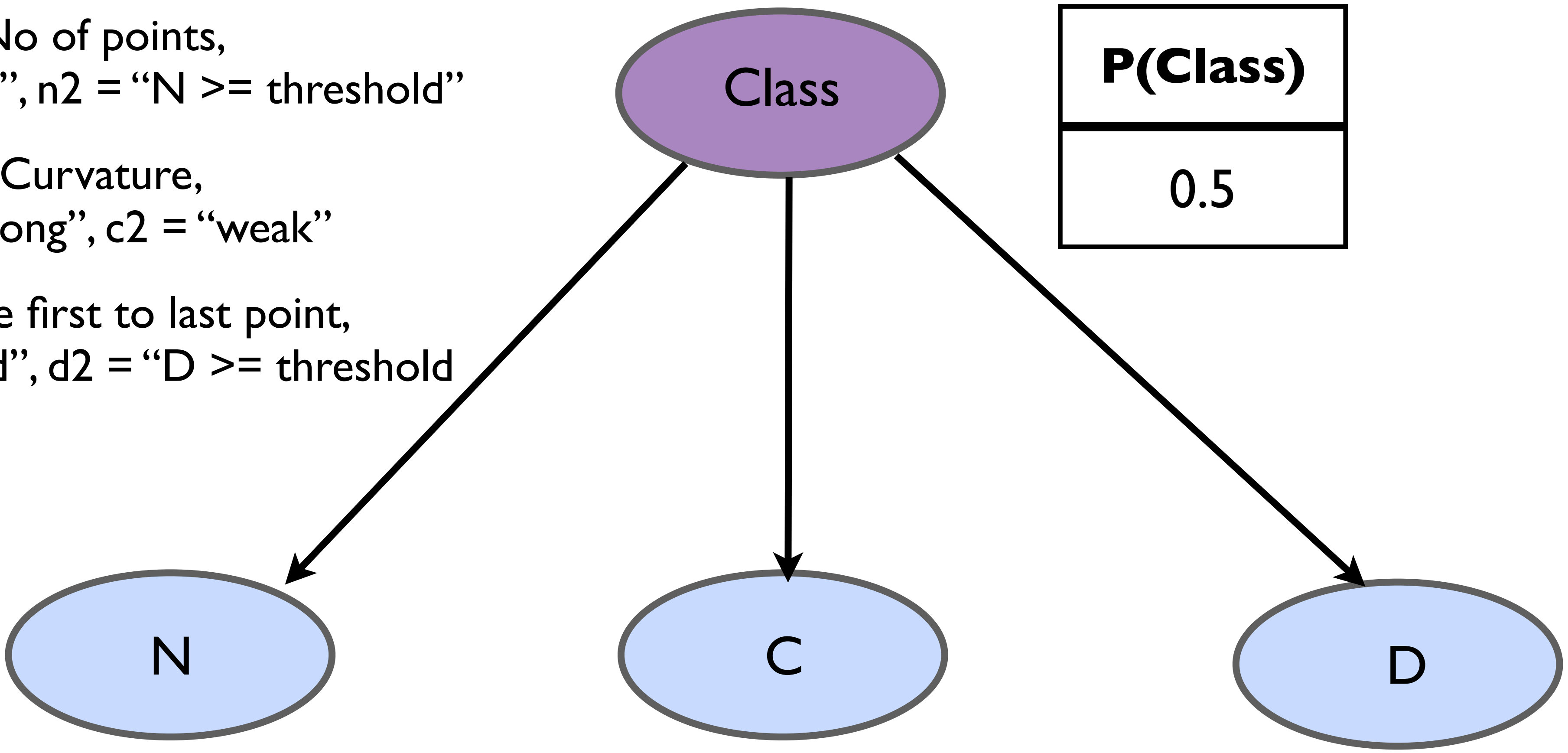
$$v_{NB} = \underset{v_j}{argmax}\ P(v_j) \prod_i P(\ a_i \mid v_j\ )$$

Expressed as a BN:
(true model)

N = No of points,
n1 = “N<threshold”, n2 = “N >= threshold”

C = Curvature,
c1 = “C=strong”, c2 = “weak”

D = Distance first to last point,
d1 = “D<threshold”, d2 = “D >= threshold



P(Class)
0.5

Class	P(N=n1|Class) = P(C = c1 | Class) = P(D = d1| Class)
Furniture	0.8
Person	0.3

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 not any observation.

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

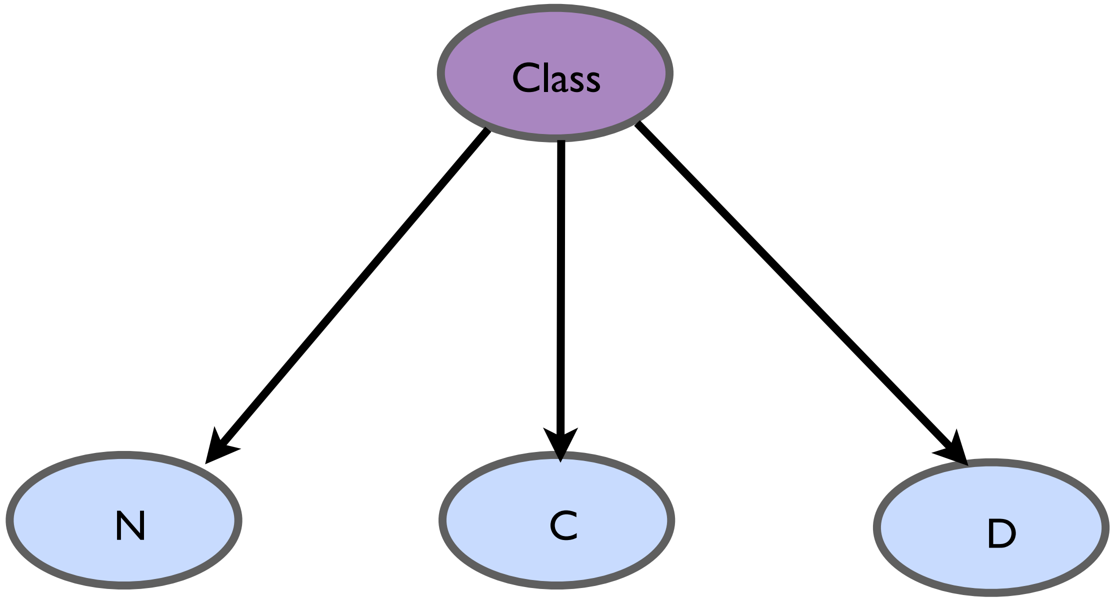
Step 2: Maximize parameters assuming this estimate

Finding the numbers … (model lost)

N = No of points,
n1 = “N<threshold”, n2 = “N >= threshold”

C = Curvature,
c1 = “C=strong”, c2 = “weak”

D = Distance first to last point,
d1 = “D<threshold”, d2 = “D >= threshold



	C = c1		C = c2	
	D = d1	D = d2	D = d1	D = d2
N = n1	273	93	104	90
N = n2	79	100	94	167

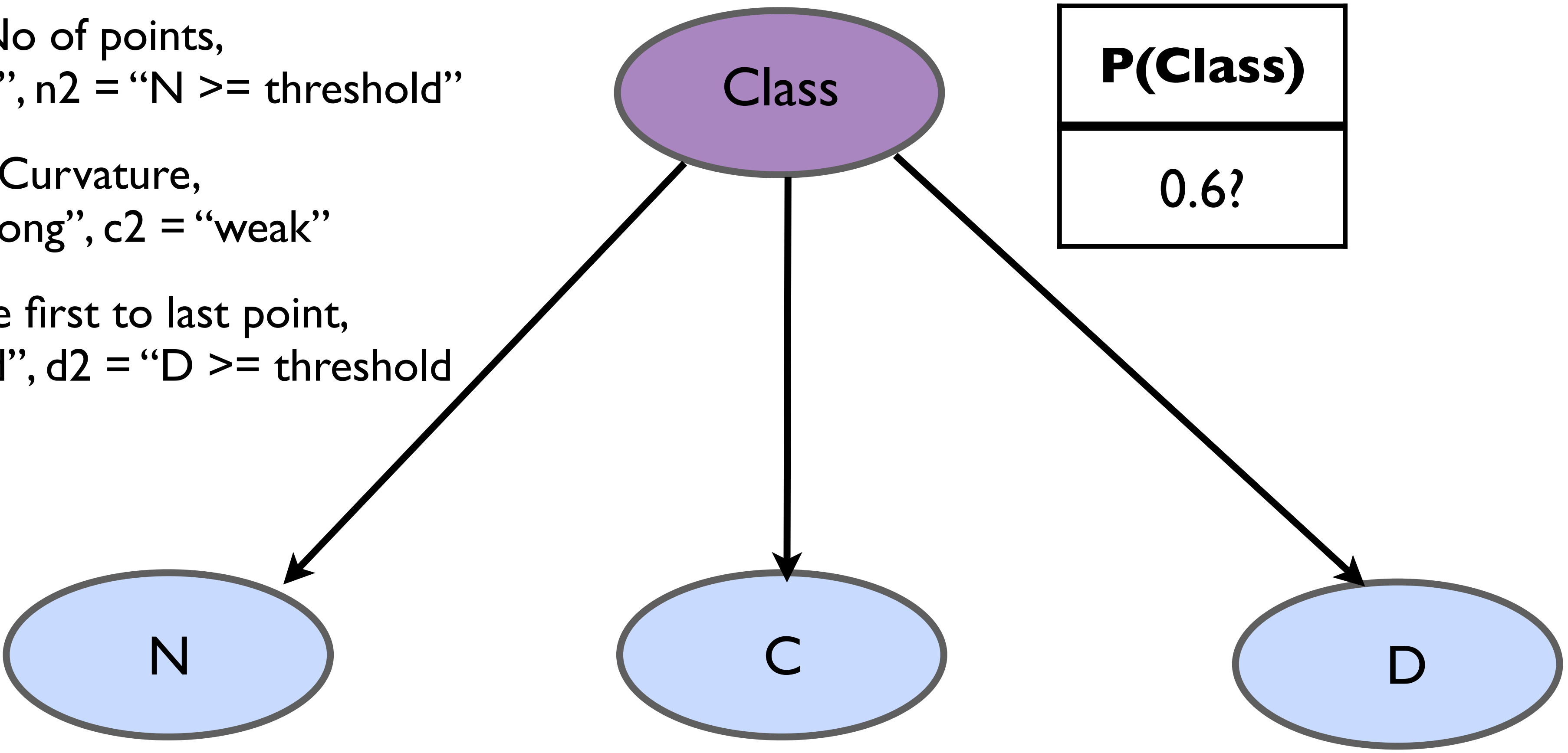
Now, we can do EM

Starting guess

N = No of points,
n1 = "N<threshold", n2 = "N >= threshold"

C = Curvature,
c1 = "C=strong", c2 = "weak"

D = Distance first to last point,
d1 = "D<threshold", d2 = "D >= threshold



P(Class)
0.6?

Class	P(N=n1|Class) = P(C = c1 | Class) = P(D = d1| Class)
Furniture	0.6?
Person	0.4?

Excourse: 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. (*)

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

(*)[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 ith position within Doc.
    • positions ⟵ all word positions in Doc that contain tokens found in Vocabulary
    • Return $v_{NB}$, where
```

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

Summary

Maximum likelihood hypothesis and MAP-hypothesis / learning

Optimal Bayes learner / classifier

Gibbs algorithm

Naive Bayes classifier

Learning Bayesian Belief Networks
- EM algorithm

(Example: The GeNIe network for interaction patterns)