# (Un)Supervised (Bayesian) Learning

## (Clustering)

Applied Machine Learning (EDAN95)
Lecture 11
2019-12-09
Elin A. Topp

Material based on Lecture Slides on Probabilistic Representation and Bayesian Learning, EDAF70, Spring 2018, Goodfellow et al, "Deep Learning", and Russel/Norvig, "AI - A Modern Approach", Murphy, "Machine Learning - A Probabilistic Perspective", Lecture Slides on EM-algorithm by S. Zafeiriou at https://ibug.doc.ic.ac.uk/media/uploads/documents/expectation_maximization-1.pdf

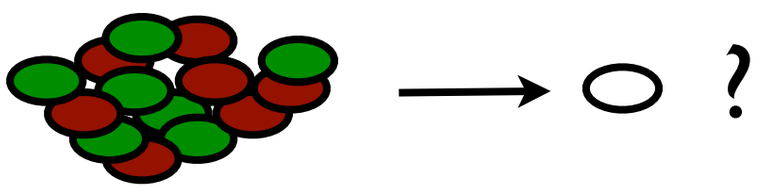
# Today's agenda

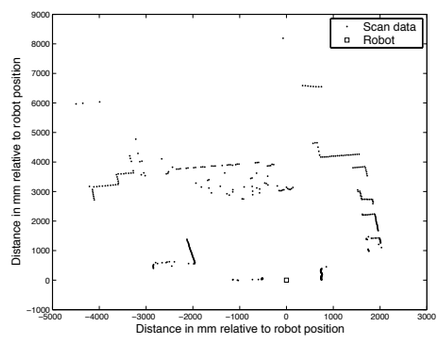
- Bayesian learning with complete data - MLE, MAP, Optimal Bayes (revisited)
- Bayesian learning with hidden variables - unsupervised Bayesian learning, EM
- Bayesian Networks (revisited), Conditional Indepence and the Markov assumption

# Today's agenda

- Bayesian learning with complete data - MLE, MAP, Optimal Bayes (revisited)
- Bayesian learning with hidden variables - unsupervised Bayesian learning, EM
- Bayesian Networks (revisited), Conditional Indepence and the Markov assumption

# Constructing explanations for predictions



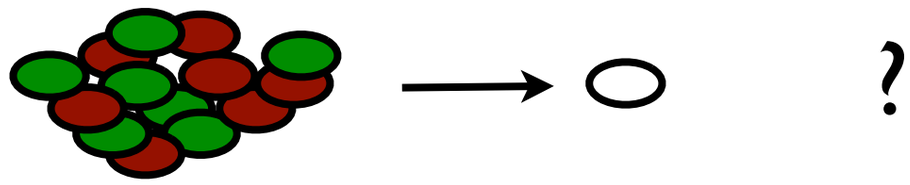



Game chips / coins with different known percentages of outcome "Head (=green)" and "Tail (=red)" when tossed.
You have a coin, but you do not know which type it was, i.e., what its bias is for getting "Head".
Can you figure that out?
And, more interestingly, what will the next tossing outcome be?

Assume you know of five coin types (which form your hypotheses), with following bias each:

| | |
|---|---|
| $h_1$: 100% Heads | $P(h_1) = 0.1$ |
| $h_2$: 75% Heads, 25% Tails | $P(h_2) = 0.2$ |
| $h_3$: 50% Heads, 50% Tails | $P(h_3) = 0.4$ |
| $h_4$: 25% Heads, 75% Tails | $P(h_4) = 0.2$ |
| $h_5$: 100% Tails | $P(h_5) = 0.1$ |

# Maximum Likelihood Estimate




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

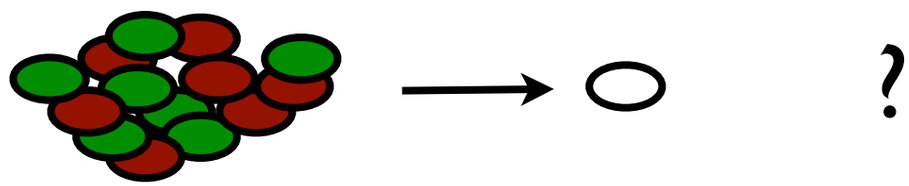
$$h_{ML} = \arg\max_{h} P(\mathbf{D} \mid h)$$

… which is a strong simplification disregarding the priors… but a good estimate over large amounts of data and within more elaborate methods - and the basis that makes NBCs and EM work.

To solve the maximisation, it is often suitable to maximise the log-likelihood instead:

$$h_{ML} = \arg\max_{h} logP(\mathbf{D} \mid h)$$

# “Maximum A Posteriori” - MAP




Finding the slightly more sophisticated ***Maximum A Posteriori*** hypothesis given a set of observations $\mathbf{D}$:

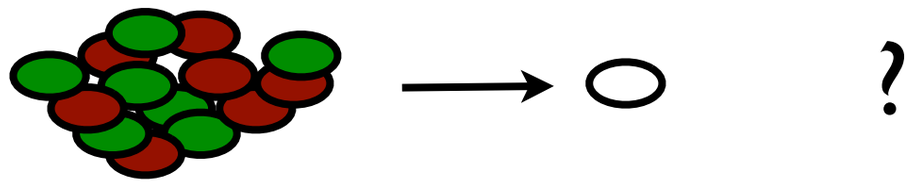
$$h_{MAP} = \arg\max_{h} P(h \mid \mathbf{D})$$

Then predict by assuming the MAP-hypothesis (quite bold) for new observation $\mathbf{X}$

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

Compare also to applying an NBC…

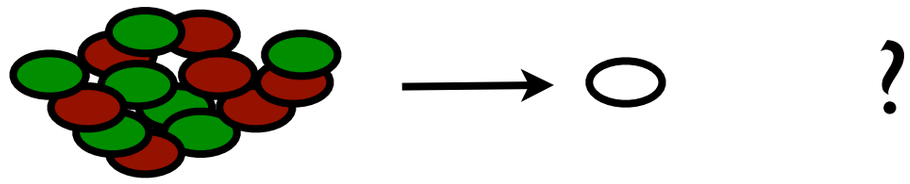
# Optimal Bayes learner

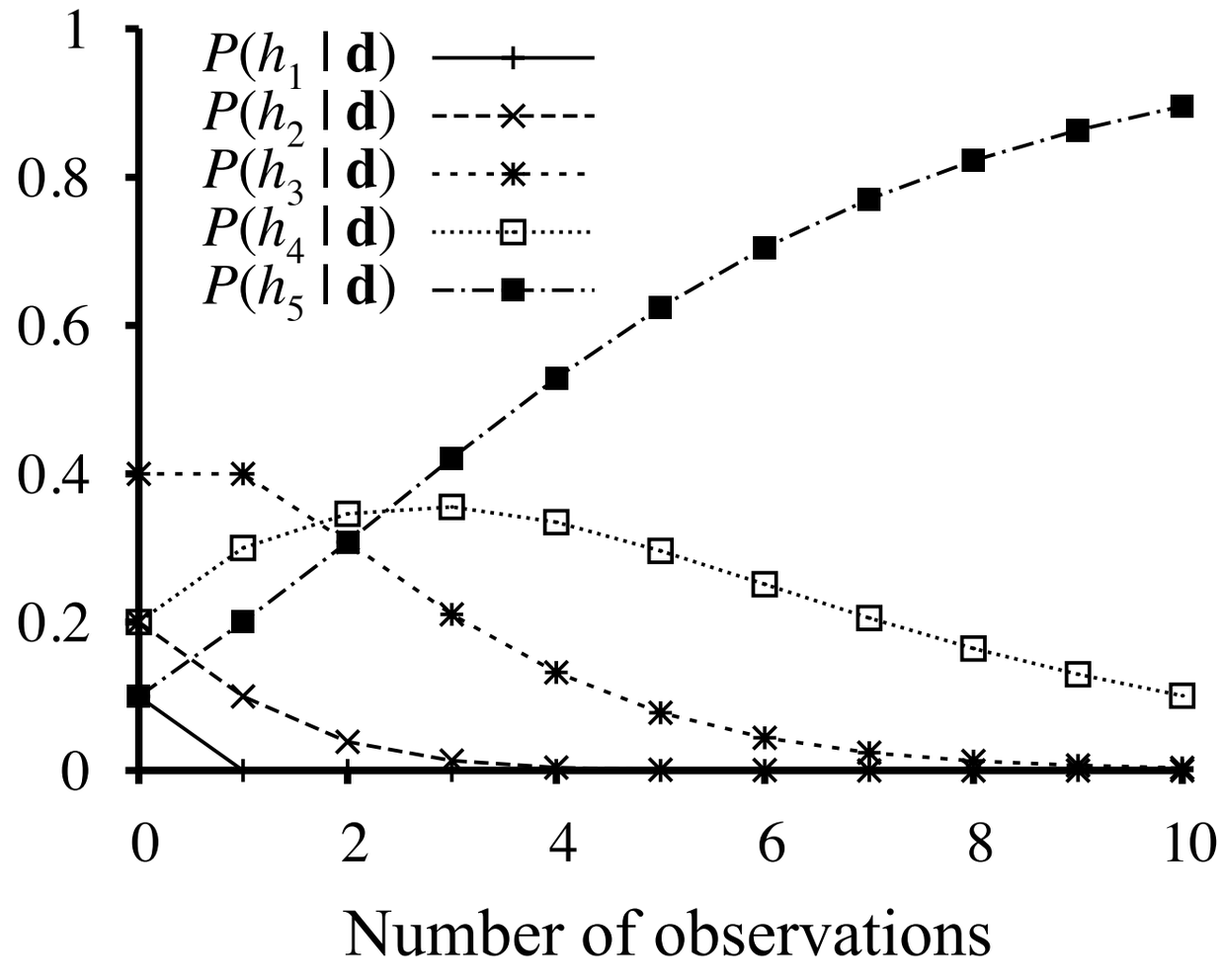



Prediction for $\mathbf{X}$, given some observations $\mathbf{D} = < \mathbf{D_0}, \mathbf{D_1}, \ldots, \mathbf{D_n} >$

$$\mathbb{P}(\mathbf{X} \mid \mathbf{D}) = \sum_{\mathbf{i}} \mathbb{P}(\mathbf{X} \mid h_i) P(h_i \mid \mathbf{D}) \qquad \text{in first step, } P(h_i \mid \mathbf{D}) = P(h_i \mid \oslash) = P(h_i)$$

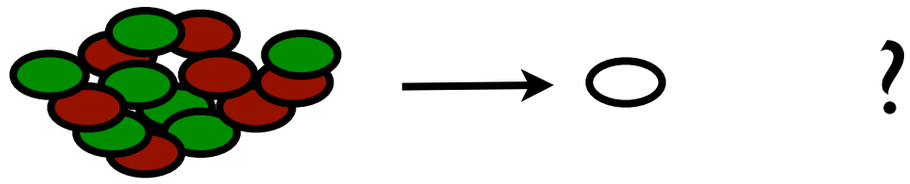
# Posterior probabilities

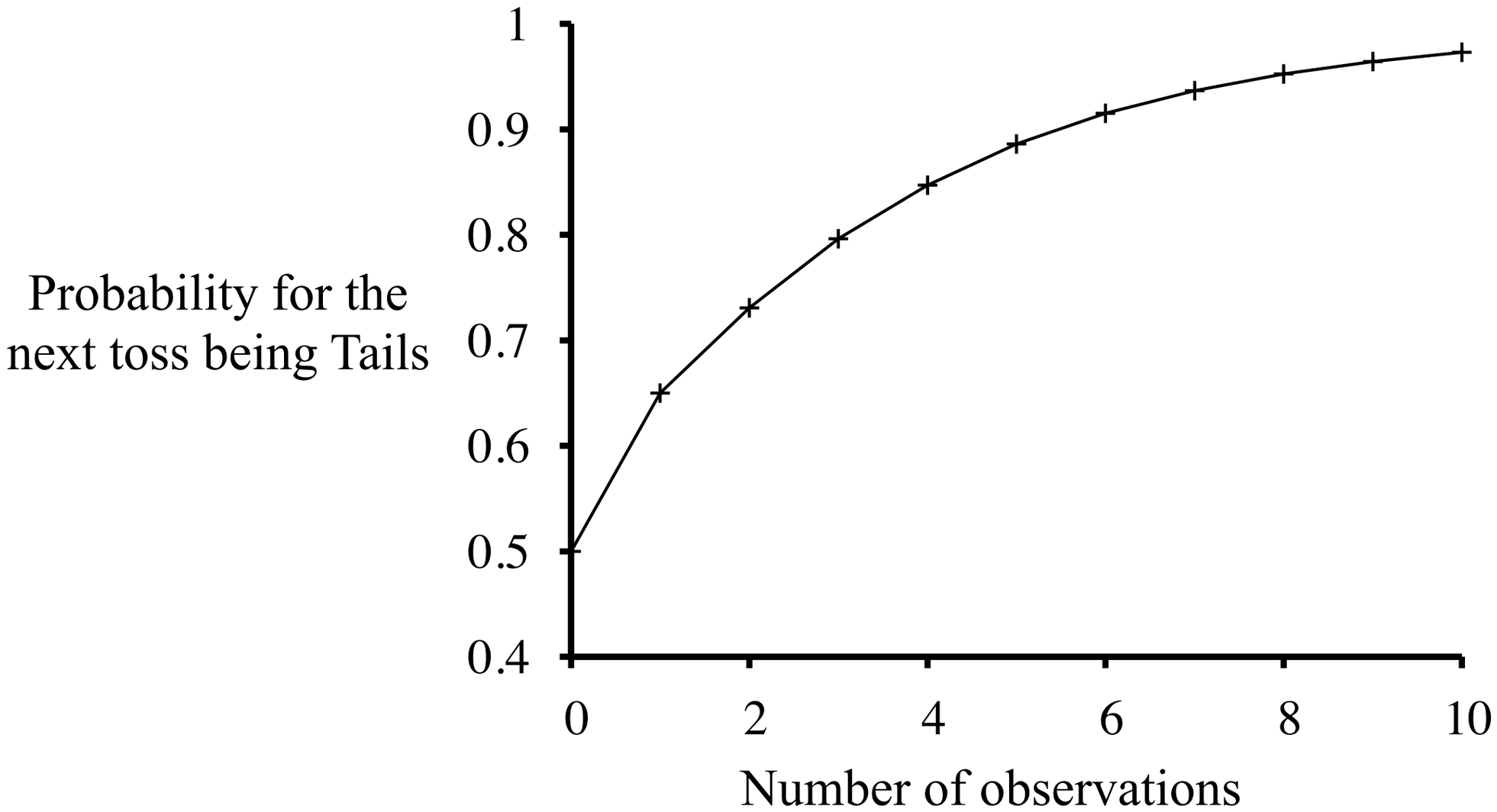




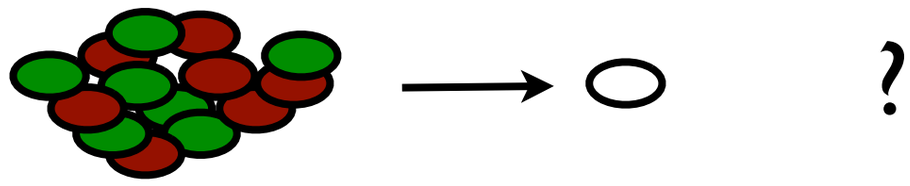

Posterior probability for hypothesis $h_k$ after *i* observations

# Prediction after sampling, OBC





# Optimal learning vs MAP-estimating




Predict by assuming the MAP-hypothesis, after 3 observations of "Tails":

$$\mathbb{P}(\mathbf{X}\,|\,\mathbf{D}) = \mathbb{P}(\mathbf{X}\,|\,h_{MAP}) \qquad \text{with } h_{MAP} = \arg\max_h P(h\,|\,\mathbf{D})$$

i.e., $P(X = Tails\,|\,D_1 = D_2 = D3)$

$= P_{h_{MAP}}(D_4 = Tails\,|\,D_1 = D_2 = D3 = Tails) = P(X\,|\,h_5) = 1$

While the optimal classifier / learner predicts

$$P_{OB}(D_4 = Tails\,|\,D_1 = D_2 = D3 = Tails) = \ldots = 0.7961$$

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 by definition, MAP-learner might be as well, both MAP and MLE-learners need a lot of data to actually produce something useful.

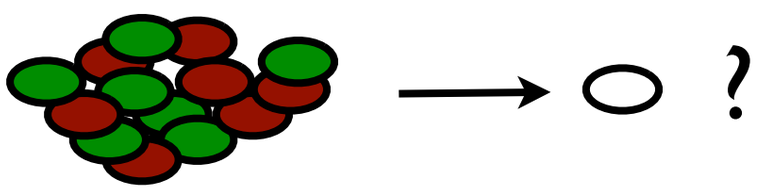
Gibbs algorithm (surprisingly well working under certain conditions regarding the a posteriori distribution for the set of hypotheses *H*):

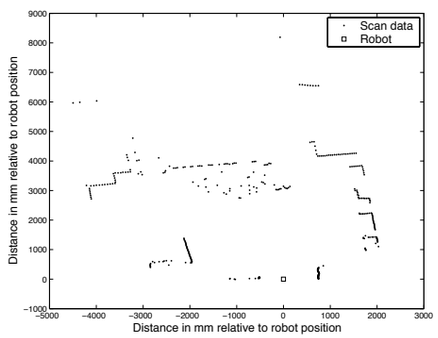
1. Choose a hypothesis *h* from *H at random*, according to the posterior probability distribution over *H* (i.e., rule out "impossible" hypotheses)
2. Use *h* to predict the classification of the next instance x.

# Today's agenda

- Bayesian learning with complete data - MLE, MAP, Optimal Bayes (revisited)
- Bayesian learning with hidden variables - unsupervised Bayesian learning, EM
- Bayesian Networks (revisited), Conditional Indepence and the Markov assumption

# No model / labels






Game chips / coins with different unknown likelihoods for outcome “Head (=green)” and “Tail (=red)” when tossed.
You have a coin, toss it, and try to find the model (hypothesis / likelihood) and thus the posterior probabilities for observing certain data.

Maximising the MLE (and thus any of the other estimates) for an unknown model is in general not possible.

But, we can compute all estimates iteratively with the EM-algorithm.

# EM intuition (coin tossing example)

Assume two coins and a series of tossing experiments, in each of which you observe a sequence of "Heads" and "Tails". You do not know which coin was used for which experiment.

Find the likelihoods $P(\mathbf{D} \mid h_i)$ and priors $P(h_i)$ for the two hypotheses $h_i$ that an observed sequence was generated with coin *i* to then be able to reason according to the resulting model.

EM works in general as follows:

- E-step: Compute the expectation for the observation given an initial estimate for priors and likelihoods.
- M-step: Update the priors and the estimate for the likelihoods for the respective hypotheses according to the expected distribution of the data from the E-step. Repeat E and M until convergence.

# EM intuition (coin tossing example)

A widely cited example:

https://ibug.doc.ic.ac.uk/media/uploads/documents/expectation_maximization-1.pdf

Additional explanation to the assumed priors on page 6:

for sequence 1, calculate $\mathbb{E}_{Coin}[5H,5T] = \alpha * \theta_{Coin}^5 * (1-\theta_{Coin})^5$ based on the assumption that we have a repeated Bernoulli experiment of drawing 10 times with replacement (thus $Bin(k|n,\theta) = \binom{n}{k} \theta_{Coin}^k * (1-\theta_{Coin})^{n-k}$ and $n$ = 10, $k$ = 5)

The expected counts from the binomial term $\binom{n}{k} = \frac{n!}{(n-k)!k!}$ become part of the normalisation factor $\alpha$.

for sequence 2, this means to calculate $\mathbb{E}_{Coin}[9H,1T] = \alpha * \theta_{Coin}^9 * (1-\theta_{Coin})$, etc.

# EM-algorithm for GMMs

This is the EM-algorithm as given by Murphy, "Machine Learning - A Probabilistic Approach", p 353.
assume data set $\mathbf{X}$ with examples $\overrightarrow{x_i}, i = 1,...,N$ and $K$ classes you want to cluster $\mathbf{X}$ into.

EM-for-GMM( $\mathbf{X}, \mathbf{K}$)

1. Initialize $\theta_k^0 = (\pi_k^0, \overrightarrow{\mu_k}^0, \Sigma_k^0)$, where

   $\pi_k$ is the class prior for class *k* (e.g., assume uniform distribution here initially)

   $\overrightarrow{\mu_k}$ are the means for the attribute values *j* in class *k* (e.g., the means over a random subset of the data)

   $\Sigma_k$ is the covariance for the attribute values in class *k* (can be simplified to variance $\sigma_{jk}^2$ for each attribute *j* if a G-NBC is assumed as the model)

2. Iterate over E and M steps as follows:

   E-step:

   compute $r_{ik}^t = \frac{\pi_k^{t-1} P(\overrightarrow{x}_i \mid \theta_k^{t-1})}{\sum_{k'} \pi_{k'} P(\overrightarrow{x}_i \mid \theta_{k'}^{t-1})}$ where $P(\overrightarrow{x}_i \mid \theta_k^{t-1}) = \prod_j \frac{1}{\sqrt{2\pi\sigma_{kj}^2}} e^{-\frac{1}{2\sigma_{kj}^2}(x-\mu_{kj}^{t-1})^2}$, assuming that the covariance can be substituted with $\sigma_{jk}^2$ for attribute *j* and class *k*.

   M-step:

   compute $r_k^t = \sum_i r_{ik}^t \quad and \quad \pi_k^t = \frac{r_k^t}{N}$, then update the means and variances:

   $\overrightarrow{\mu_k}^t = \frac{\sum_i r_{ik}^t \overrightarrow{x_i}}{r_k^t} \quad and \quad \Sigma_k^t = \frac{\sum_i r_{ik}^t \overrightarrow{x}_i \overrightarrow{x}_i^T}{r_k^t} - \overrightarrow{\mu_k^t} \overrightarrow{\mu_k^t}^T$ (from which the new $\sigma_{jk}^2$ can be extracted)

3. Stop, when the $\overrightarrow{\mu_k}$ and $\Sigma_k$ are not changing significantly anymore.

# EM-algorithm for k-Means

This is the EM-algorithm as given by Murphy, "Machine Learning - A Probabilistic Approach", p 356.
assume data set $\mathbf{X}$ with examples $\overrightarrow{x_i}, i = 1,...,N$ and $K$ classes you want to cluster $\mathbf{X}$ into.

k-Means( $\mathbf{X}, \mathbf{K}$)
1. Initialize $\overrightarrow{\mu_k}^0$, assume fixed class priors $\pi_k$
2. *Iterate* over E and M steps as follows:

E-step:

Assign each data point to its closest cluster centre: $z_i = \arg\min_k \| \overrightarrow{x}_i - \overrightarrow{\mu}_k \|_2^2 = L_2(\overrightarrow{x}_i - \overrightarrow{\mu}_k)^2$

M-step:

Update each cluster centre by computing the means of all points assigned to it:

$$\overrightarrow{\mu}_k = \frac{1}{N_k} \sum_{i:z_i=k} \overrightarrow{x}_i$$

*Until converged*

# Outlook on lab 6

- Implementation of EM-algorithm to find the Gaussians for the GNB for the digits data
- Compare to k-Means clustering

- Write report (note: there are some aspects of labs 2 and 5 to be considered, refresh your memory!)

# Today's agenda

- Bayesian learning with complete data - MLE, MAP, Optimal Bayes (revisited)
- Bayesian learning with hidden variables - unsupervised Bayesian learning, EM
- Bayesian Networks (revisited), Conditional Indepence and the Markov assumption

# Conditional independence

$\mathbb{P}$*( Number, Pixel[0], …, Pixel[63])* has $10*17^{64} - 1 \approx 5.61e^{78}$ independent entries

But: If looking at a specific number, the probability distribution for "Pixel[36]" does not depend on whether Pixel[4] has a certain value or not (this dependency is now "implicit" in some sense):

(1) $\mathbb{P}$*( Pixel[36] | Pixel[4] = 0.0, number = 0)* = $\mathbb{P}$*( Pixel[36] | number = 0)*

The same holds when we are looking at another number:

(2) $\mathbb{P}$*( Pixel[36] | Pixel[4] = 0.0, number != 0)* = $\mathbb{P}$*( Pixel[36] | number != 0)*

*Pixel[36]* is *conditionally independent* of *Pixel[4]* (and all other Pixels) given *Number*:

$\mathbb{P}$*( Pixel[36] | Pixel[4], Number)* = $\mathbb{P}$*( Pixel[36] | Number)*

Writing out the full joint distribution using chain rule and conditional independence assumption:

$\mathbb{P}$*( Pixel[0], …, Pixel[63], Number)*
= $\mathbb{P}$*( Pixel[0] | Pixel[1], …, Pixel[63], Number)* $\mathbb{P}$*( Pixel[1], …, Pixel[63], Number)*
= $\mathbb{P}$*( Pixel[0] | Pixel[1], …, Pixel[63], Number)* $\mathbb{P}$*( Pixel[1] | Pixel[2], … Pixel[63], Number)* $\mathbb{P}$*( Pixel[2], …, Pixel[63], Number)*
= … =
= $\mathbb{P}$*( Pixel[0] | Number)* … $\mathbb{P}$*( Pixel[63] | Number)* $\mathbb{P}$*( Number)*

gives thus 64*((17-1)*10) + (10-1) = *10249* independent entries

# Bayesian networks

A simple, graphical notation for *conditional independence assertions* and hence for compact specification of full joint distributions

Syntax:

a set of nodes, one per random variable

a directed, acyclic graph (link ≈ "directly influences")

a conditional distribution for each node given its parents:

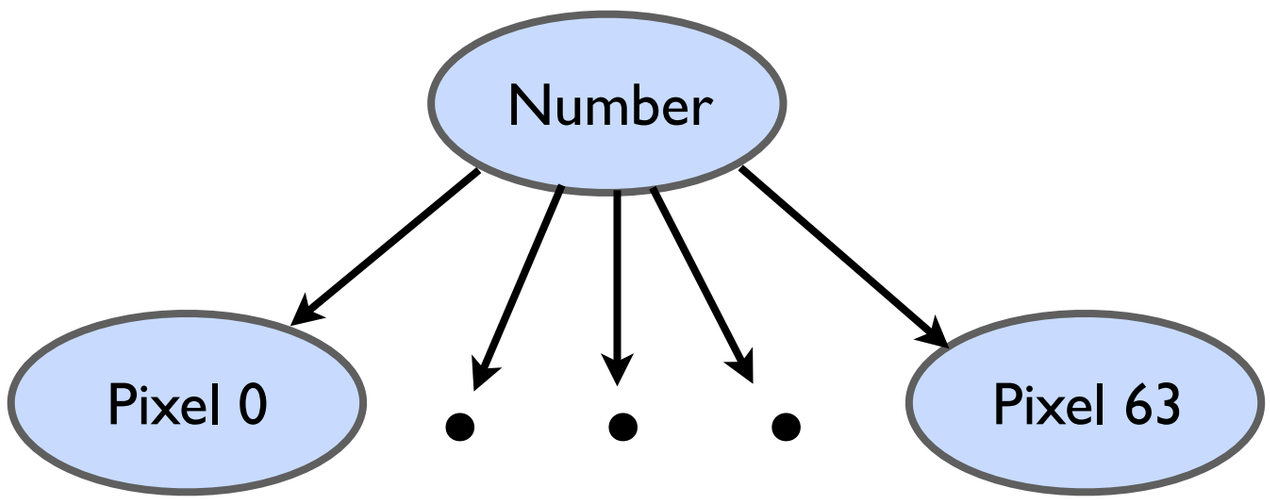
$\mathbb{P}( X_i \mid Parents( X_i))$

In the simplest case, conditional distribution represented as a

*conditional probability table* ( CPT)

giving the distribution over $X_i$ for each combination of parent values

# The BN for the digits data

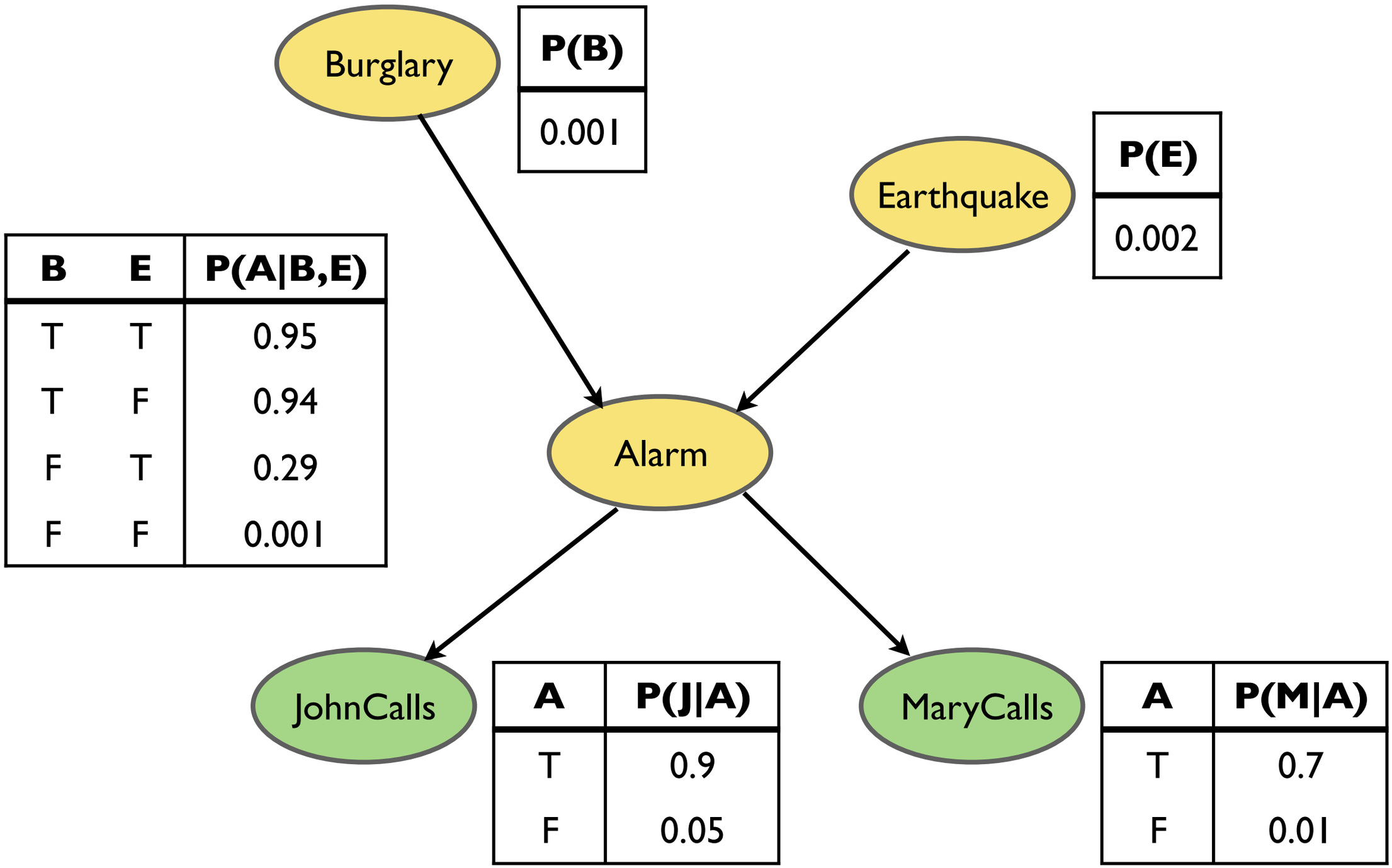



$CPT_{jk} = P(Pixel_0 = v_j \,|\, Number = k)$ $\qquad$ $CPT_{jk} = P(Pixel_{63} = v_j \,|\, Number = k)$

$$P(X = Number) = \arg\max_k [P(Number = k, Pixel_0, \ldots, Pixel_{n-1})]$$

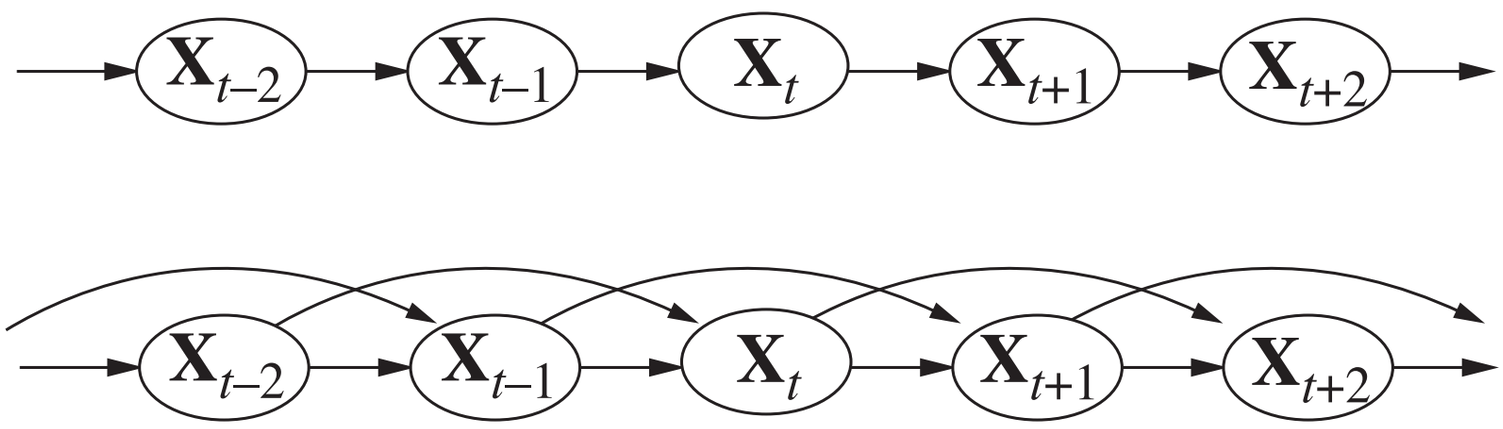
$$= \arg\max_k [P(Number = k) \prod_i P(Pixel_i = X_i \,|\, Number = k)]$$

# Observable and "hidden" variables



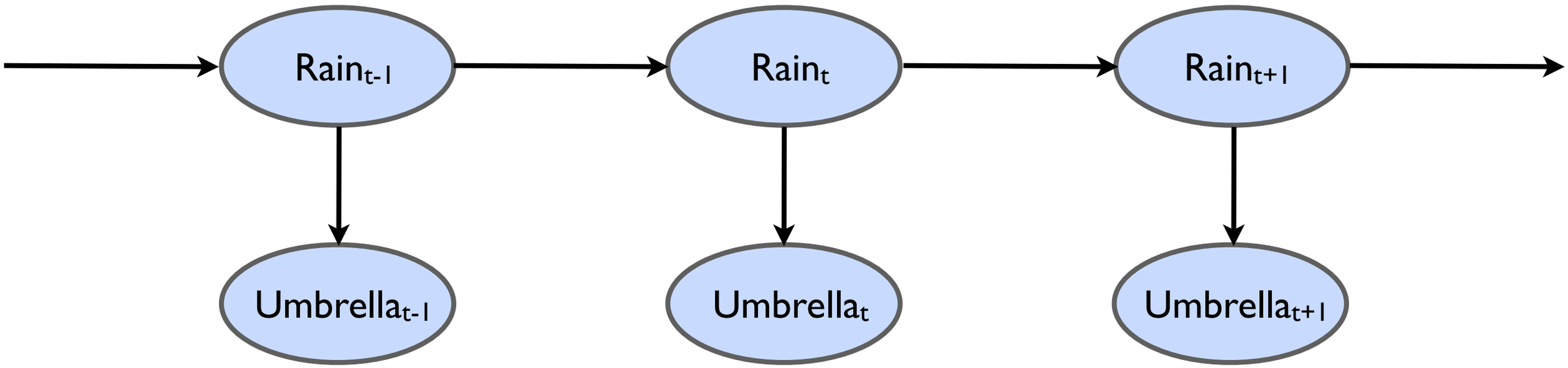
# The Markov assumption

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 |




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

# Outlook on lectures 12-14 and beyond

- From Markov Chains to Markov Decision Processes (MDP)
- MDP and Reinforcement Learning, MCMC (briefly)
- Policy Search, Policy / Value Iteration, Q-Learning, SARSA-learning

- Lab 6 is already online (in a provisional version)

- Save the date: January 7, 13-15, double guest seminar on ML
  with Marc Deisenroth (University College, London) and Shakir Mohamed (DeepMind)
  to be found in the calendar on http://rss.cs.lth.se
  PLEASE REGISTER if you are interested!!!

# Today's summary

- Summarised Bayesian learning approaches based on MLE, MAP, and Optimal Bayes
- Introduced Naive Bayesian Classifiers
- Introduced Gaussian (Mixture) Models and GNBs (very briefly)

- Reading:
  - Murphy, ch 11
  - https://ibug.doc.ic.ac.uk/media/uploads/documents/expectation_maximization-1.pdf
  - Mitchell, chapter 6
  - Bayesian Networks: see also Russel / Norvig, ch 14)