Overview

I Artificial Intelligence
II Problem Solving
III Knowledge, Reasoning, Planning
IV Uncertain Knowledge and Reasoning
V Machine Learning
    19. Learning from Examples
    20. **Learning Probabilistic Models**
    21. Deep Learning
    22. Reinforcement Learning
VI Communicating, Perceiving, and Acting
VII Conclusions

Learning Probabilistic Models

- Statistical learning
- Learning with complete data
- Learning with hidden variables: The EM algorithm

Statistical Learning

- **Given:**
  - Several hypotheses with apriori probabilities
  - Observations made by one of the hypotheses
- **Sought:**
  - Most probable hypothesis
  - Prediction of further observations

Example:

- We buy a large bag of surprise candies which has two sorts equally wrapped (C1 = cherry, C2 = lime), but we don‘t know the distribution.
- We know, that there are 5 bag types (hypotheses) with h_1: 100% cherry; h_2: 75% cherry, 25% lime; h_3: 50% cherry, 50% lime; h_4: 25% cherry, 75% lime; h_5: 100 lime.
- Their apriori probabilities are: <10%, 20%, 40%, 20%, 10%>.
- We take some candies from the bag, e.g. (C1, C1, C1, C2, C1). What kind of bag have we bought and what‘s the sort of the next candy?

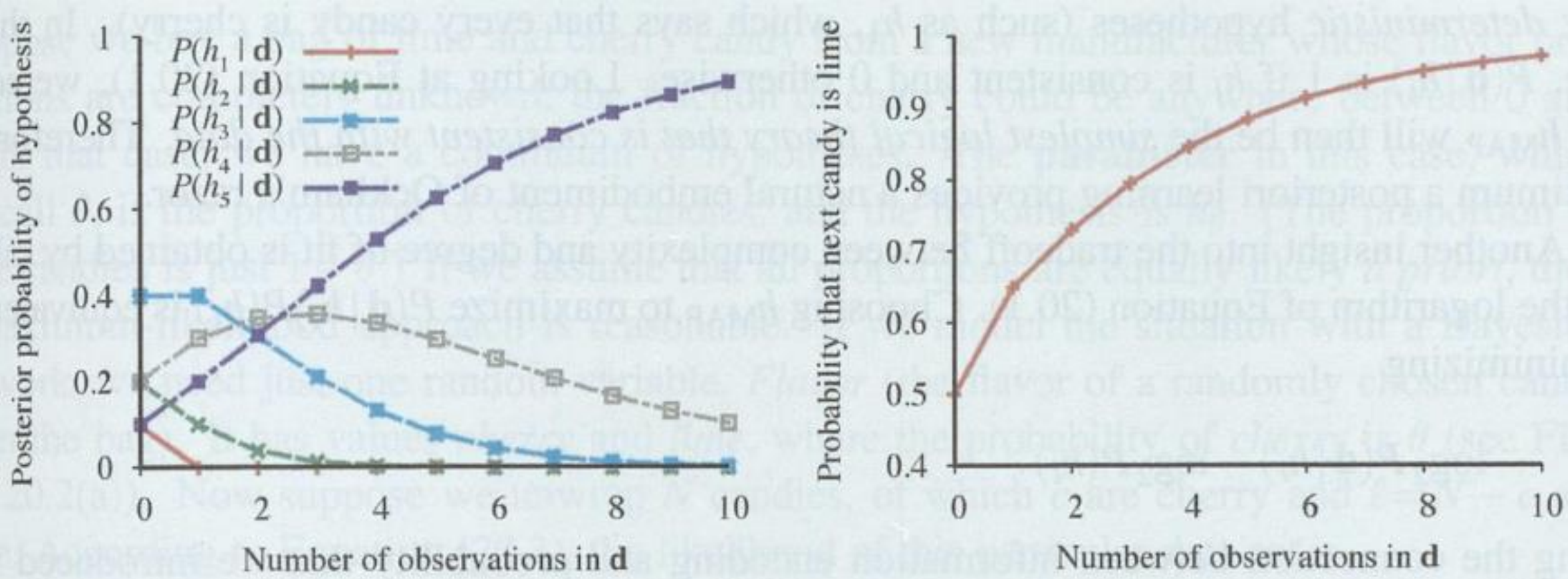
Bayesian Learning

Bayesian learning simply calculates the probability of each hypothesis, given the observed data d and makes predicitions on that basis

- Compute for each hypothesis h_i its probability:
  - $P(h_i|d) = \alpha\ P(d|h_i)\ \ P(h_i)$
  - $P(d|h_i) = \Pi_j\ P(d_j|h_i) = P(d_1|h_i) * P(d_2|h_i) * P(d_3|h_i) * \ldots * P(d_n|h_i)$
  - Normalize h_i
- Compute probability for next observation:
  - Compute for each hypothesis the probability of the next observed value (trivial) and add the probabilities for the values weighted by the probabilities of the hypotheses

Results

- (left) Probabilities of the 5 hypotheses, if n candies all of the sort C2 (lime) are taken (n: 1 .. 10)
- (right) Probabilities that the next candy is from type C2 (lime)



Simplifications to Bayesian learning:

- Approximation1 **(Maximum a Posteriori; MAP):**
  - Use in prediction of next observation only the most probable hypothesis (instead of all hyptheses)

- Approximation2 **(Maximum Likelihood; ML):**
  - Ignore in first computation the apriori probabilities $P(h_i)$ of the hypotheses

$$P(h_i|d) = \alpha\, P(d|h_i)\ \cancel{P(h_i)}$$

Justifications of Approximations

- **Maximum a Posteriori:**
  - Use only the most probable hypothesis
    - In example after the three observations with C2= lime, the most probable hypothesis is h_5: 100% lime. Therefore the predicition of the fourth candy is C2=lime with 100%, which has a real probability of 80%
  - Justification: Ockhams razor and pragmatism: In most applications, there are many hypotheses, whose computation and summation would be difficult (e.g. we don‘t know that there are 5 bag types)
- **Maximum Likelihood:**
  - Ignore the apriori probabilities $P(h_i)$ of the hypotheses
    - Justification: In most applications the apriori probabilities of hypotheses are unknown and difficult to compute. Their effects rapidly diminish with observations.

Learning with Complete Data

- **Density estimation** task: Learning a probability model from data generated by that model
  - Quite simple with complete data
  - Example with discrete model: Learning the fraction of cherry candies without any prior knowledge

Maximum Likelihood Parameter Learning

- A framework for explaing learning parameters of discrete and continous models
  - Learning discrete Bayesion probabilities (see next slides)
  - Learning continuous models like linear-Gaussian models including linear regression

Maximum Likelihood Parameter Learning

- Example: What is the most likely hypothesis, if we know, that there are two sorts of candies in a bag, but we don‘t know the proportion. We take N candies from the bag and unwrap them. We find c cherry candies and l lime candies (c + l = N).
- We have a continuum of hypotheses with one parameter, the proportion of cherry candies, called θ. We call our hypothesis h_θ.
- General: $P(d|h_\theta) = \prod_{j=1}^{N} P(dj|h_\theta) = \theta^c \cdot (1-\theta)^l$
- Using the Log Likelihood $L(d|h_\theta)$: $\log P(d|h_\theta) = \sum_{j=1}^{N} \log P(dj|h_\theta) = c \log \theta + l \log (1-\theta)$
- Differentiate and set the resulting expression = 0

$$\frac{dL(d|h_\theta)}{d_\theta} = \frac{c}{\theta} - \frac{l}{1-\theta} = 0 \Rightarrow \theta = \frac{c}{c+l} = \boldsymbol{\frac{c}{N}}$$

The maximum likelihood hypothesis h_{ML} asserts, that the actual proportion of cherry candies in the bag is equal to the observed proportion!

General Process of ML-Parameter Learning

1. Write down an expression of the likelihood of the data as a function of the parameter(s)

2. Write down the derivative of the log likelihood with respect to each parameter

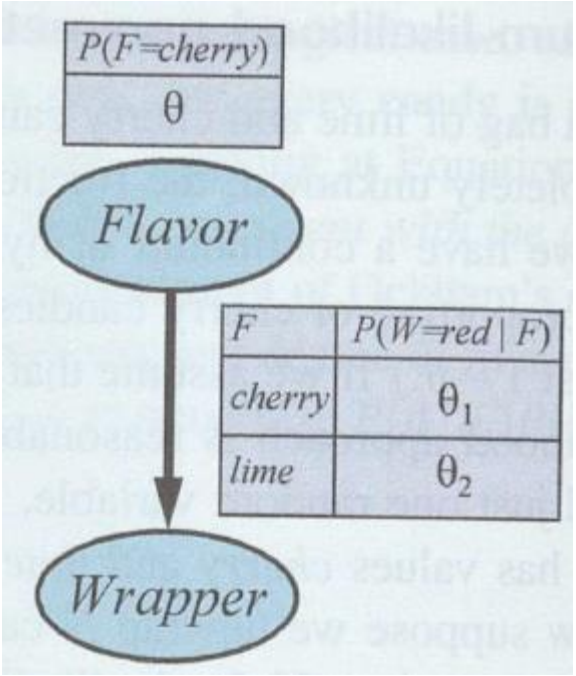
3. Find the parameter values such that the derivatives are zero

Comments:

- The last step might require iterative solution algorithms
- If we haven‘t observe something (e.g. no cherry candies), zero probabilities result
  - To avoid this, counts might be initialized with 1 instead of 0

Hypotheses with Conditional Probabilities (1)

Extension of example: Instead of computing just the apriori distribution of cherry and lime candies, we have additional information from red and green wrapper colors. The model now has three parameters θ, θ_1, θ_2



The likelihood of a green wrapped cherry candy is:

$P(\text{Flavor} = \text{cherry}, \text{Wrapper} = \text{green} \mid h_{\theta,\theta_1,\theta_2}) =$

$P(\text{Flavor} = \text{cherry} \mid h_{\theta,\theta_1,\theta_2}) \; P(\text{Wrapper} = \text{green} \mid \text{Flavor} = \text{Cherry}, h_{\theta,\theta_1,\theta_2}) =$

$\theta \cdot (1 - \theta_1)$

Hypotheses with Conditional Probabilities (2)

Now we unwrap N candies:

- c cherry candies, of which are r_c red and g_c green, and l lime candies, r_l red and g_l green, i.e. $c+l = N$; $r_c + g_c = c$; $r_l + g_l = l$

The likelihood of the data is: $P(d \mid h_{\theta,\theta_1,\theta_2}) = \theta^c(1-\theta)^l \cdot \theta_1^{r_c}(1-\theta_1)^{g_c} \cdot \theta_2^{r_l}(1-\theta_2)^{g_l}$

Taking the logarithm: $L = [c \log\theta + l \log(1-\theta)] + [r_c \log\theta_1 + g_c \log(1-\theta_1)] + [r_l \log\theta_2 + g_l \log(1-\theta_2)]$

Taking the derivative of each of the three parameters θ, θ_1, θ_2 and set them to zero:

$$\frac{\partial l}{\partial \theta} = \frac{c}{\theta} - \frac{l}{1-\theta} = 0 \quad \Rightarrow \theta = \frac{c}{c+l}$$

$$\frac{\partial l}{\partial \theta_1} = \frac{r_c}{\theta_1} - \frac{g_c}{1-\theta_1} = 0 \quad \Rightarrow \theta_1 = \frac{r_c}{r_c+g_c}$$

$$\frac{\partial l}{\partial \theta_2} = \frac{r_l}{\theta_2} - \frac{g_l}{1-\theta_2} = 0 \quad \Rightarrow \theta_2 = \frac{r_l}{r_l+g_l}$$

The solutions are again the observed proportions: cherry candies from all candies, red cherry candies from all cherry candies and red lime candies from all lime candies

Summary

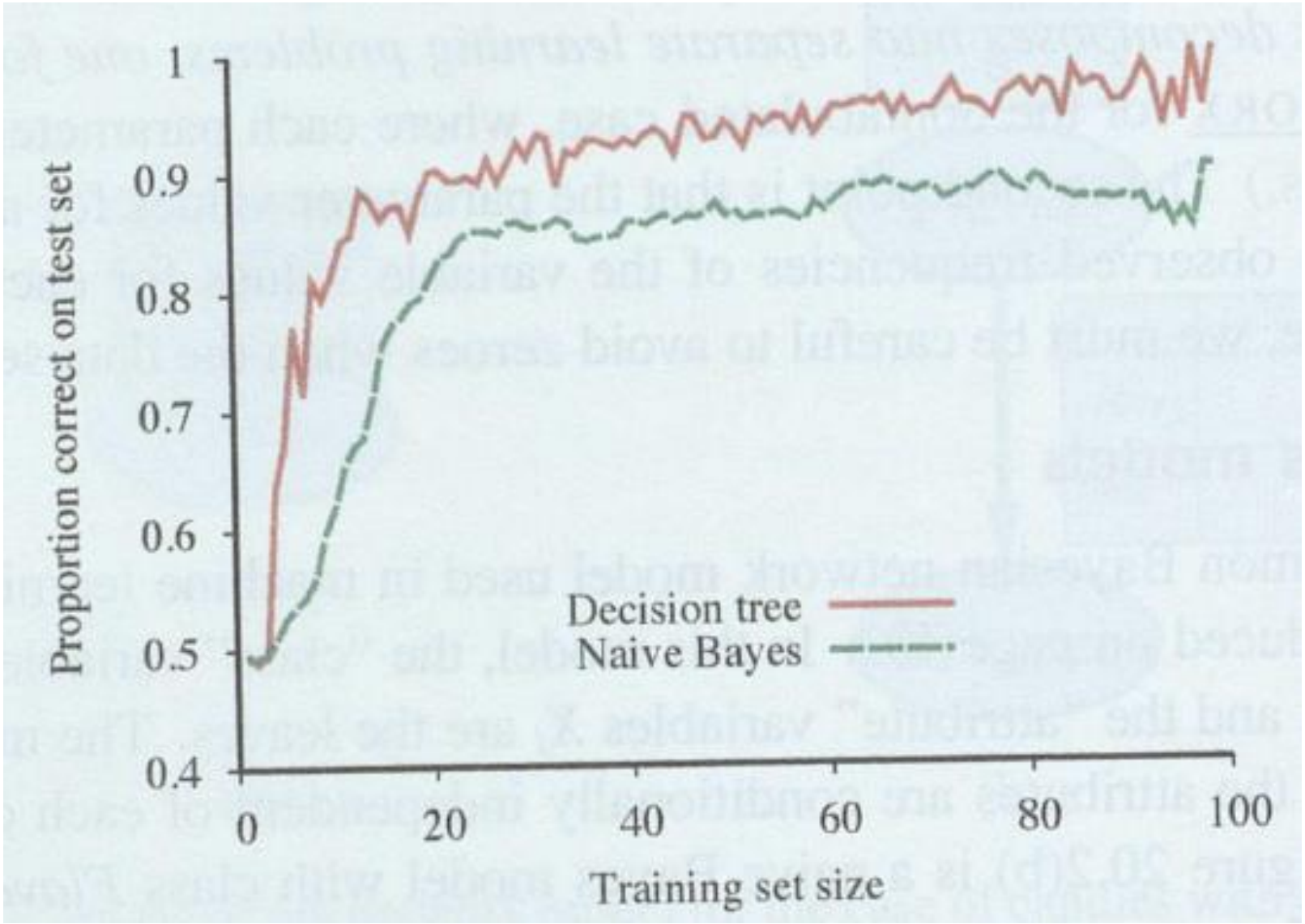
- Parameter learning with complete data can be divided in separate learning problems for each parameter under the independance assumption
- Probability values for a variable with known parents are the observed proportions of the observations (for each value of the parents)
- Problems with zero-probabilities (solution usually initializing with small numbers)

➢ Learning of probabilities for Naive Bayes Theorem in this manner is very easy and efficient for data collections

- Does surprisingly well in a wide range of application
- Boosted version is one of the most effective general-purpose learning algorithm
  - Similar to random forests
- Deals well with missing and noisy data
- Primary drawback is independance assumption (might need feature engineering)

Comparison for Decision tree and Naive Bayes Learning

- With restaurant data:



Generative and Discriminative Models

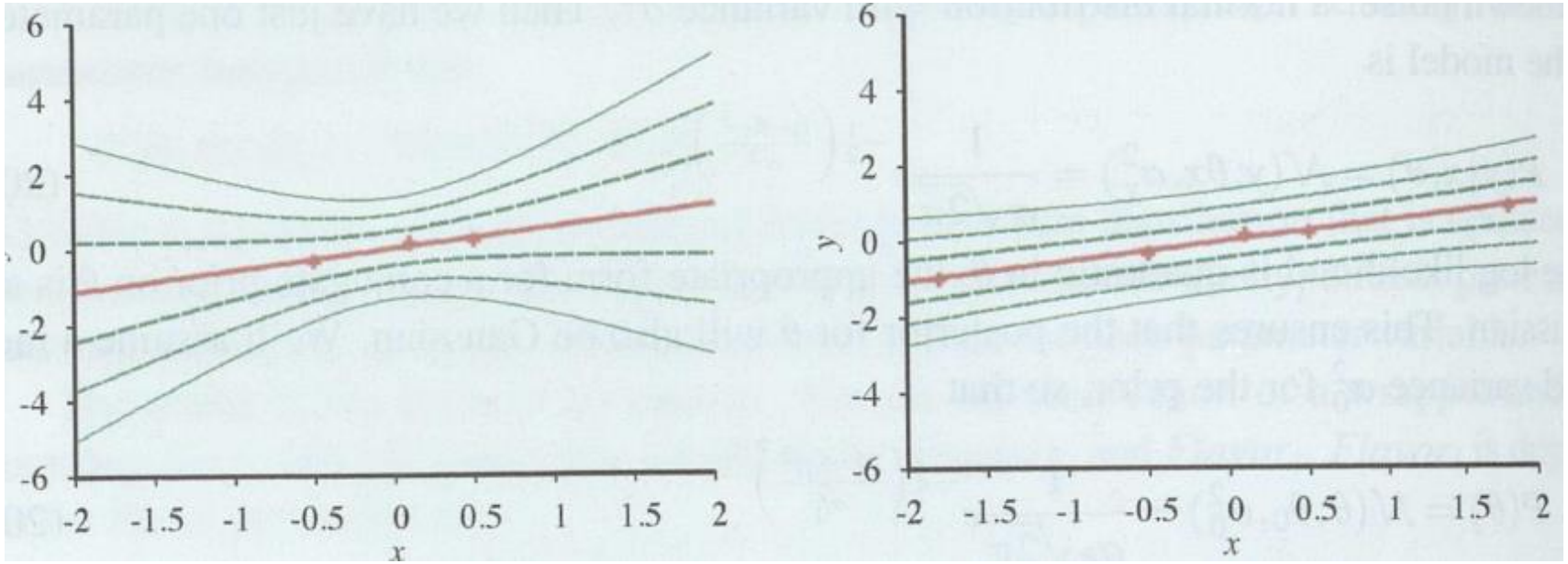
Two kinds of ML models for classification:

- **Generative models**: Model the probability distribution for each class
  - e.g. a naive Bayes text classifier creates a separate model for each category (weather, sports, etc.)
  - Each model needs all parameters, i.e. a prior probability and the conditional probabilities
  - For classification the model with highest probability given the data is chosen
  - The model can also be used to generate a random selection of words representative for its category (e.g. weather)
- **Discriminative models**: directly learn the decision boundary between classes
  - Examples: Decision trees, logistic regression, support vector machines
  - Cannot be used for generating random words for the categories
  - Usually performs slightly better for classification

Bayesian Parameter Learning

- Bayesian maximum likelihood models – both for discrete and continuous values – provide hypotheses making predicitions with the same certainty regardless whether the data set for training was small or large
- Better models would include hyperparameters for probability density functions that reflect the amount of training data in some way
  - E.g. supplement a probability θ by two hyperparameters a and b representing the virtual count of corresponding observations
  - For continuous models like linear functions, hyperparameters might represent the range of data points on the linear functions

Example: Red line is the hypothesis based on 3 (left) and 5 (right) data points. The other curves represent uncertainty intervals



Learning Bayes Net Structures

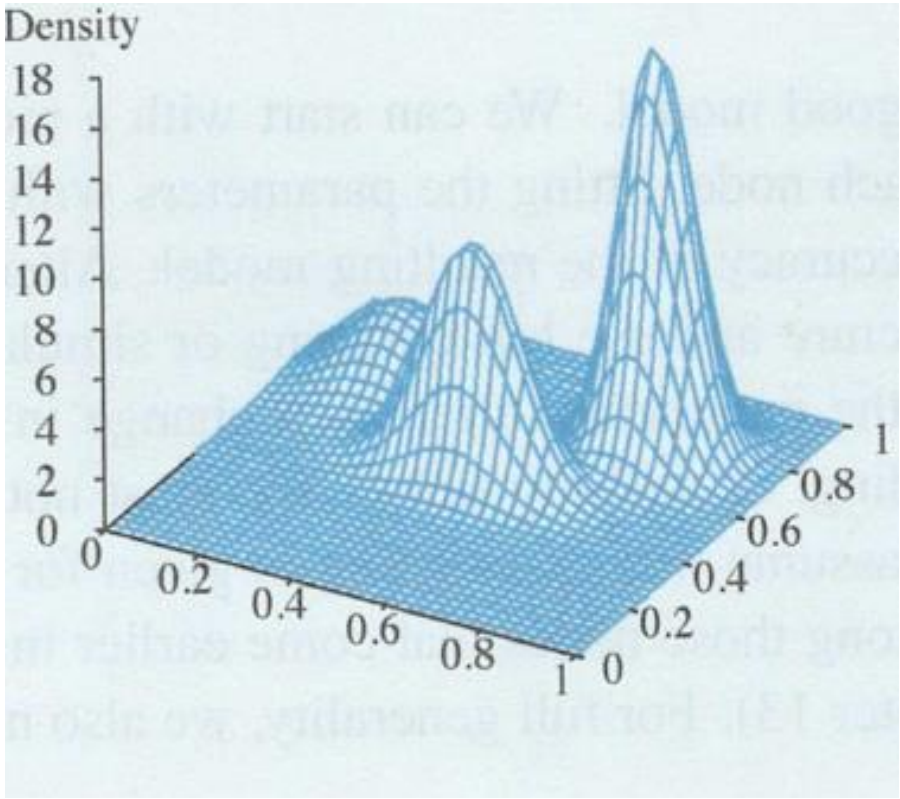
- The structure of Bayesian Networks is usually specified by background knowlege (e.g. provided by a human or literature) based on causal knowledge
- Learning Bayes net structures can be implemented as a search for a good model
  - Start from scratch with variables (nodes) without links and sequentially add parents to the nodes
  - Start with a full model and make modifications (reversing, adding, deleting links)
  - Sometimes an order is prespecied, in which the nodes are processed
- The critical question is, how to measure the quality of the resulting network structures
  - Total fit of the data to the final model is a candidate, but then complex hypotheses are preferred (penalty for complexitiy?)
  - Criteria for conditional independance of two nodes: What amount of noise can be tolerated?

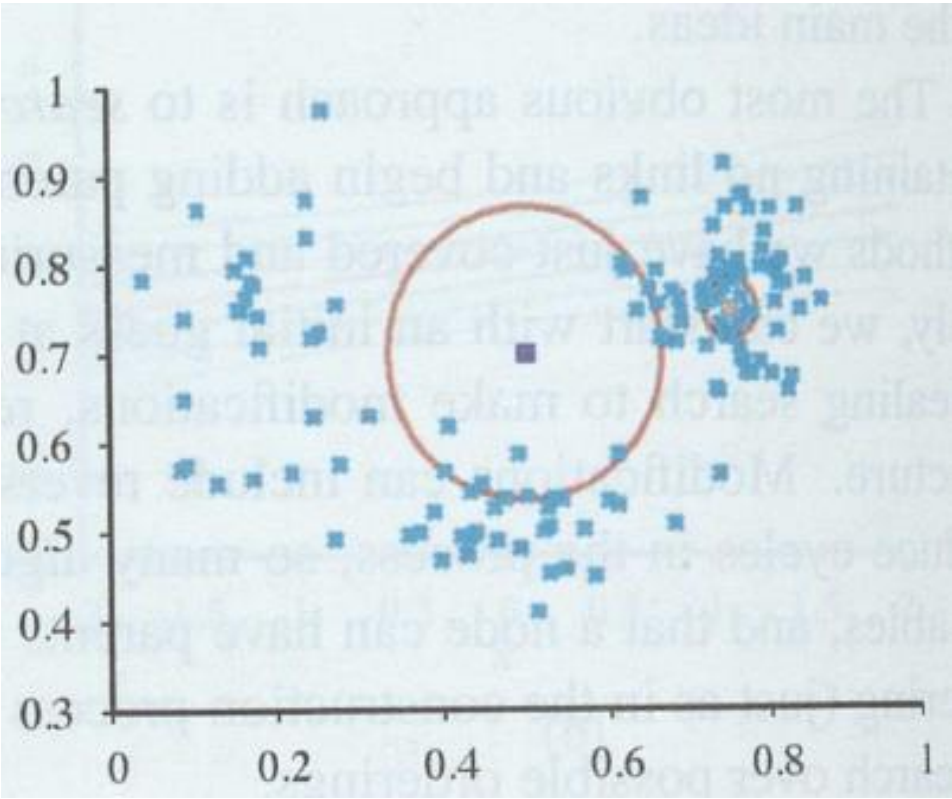
Densitity Estimation with Nonparametric Models

- Learn a probability model without making any assumptions about its structure
- Example: k-nearest-neighbor models (can be used in addition to classification and regression also for density estimation):
  - Estimate the density for a query point by the number of nearest neighbors around the query point

Example:

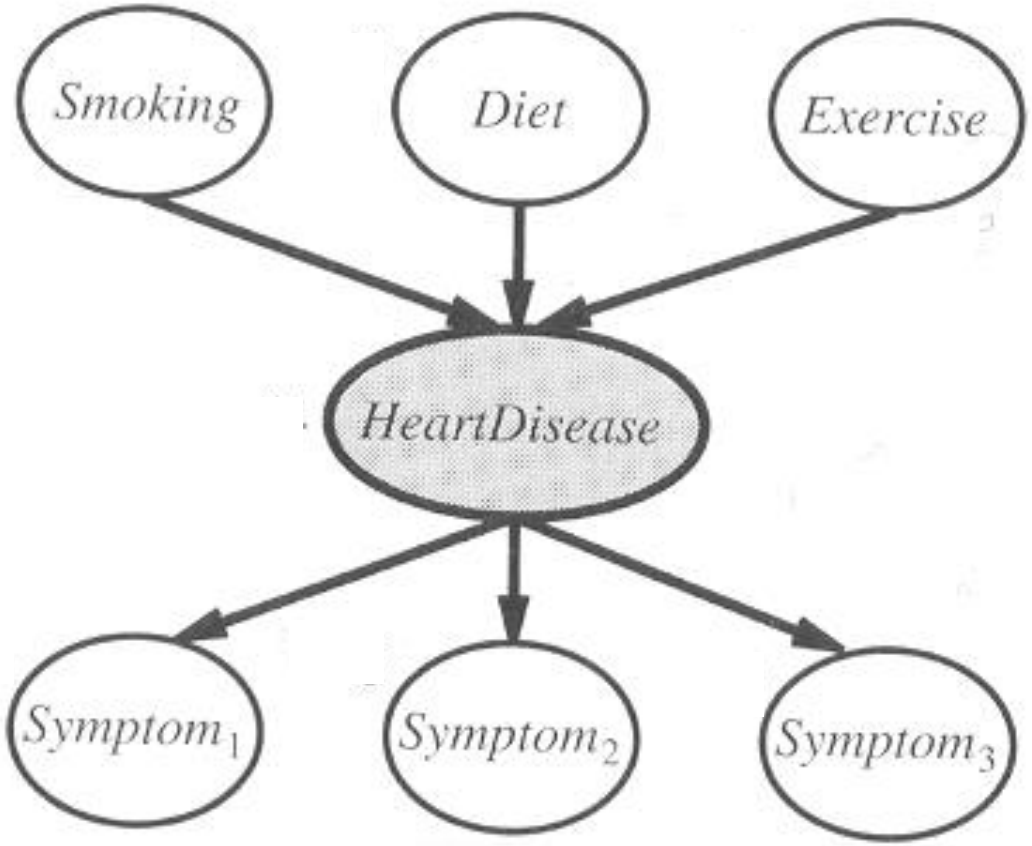
- *Left: 3-D plot of a Gaussian mixture of 3 components*
- *Right: sample points generated by the mixture and 2 query points with their 10-neigborhood density (red circles)*





Learning with Hidden Variables

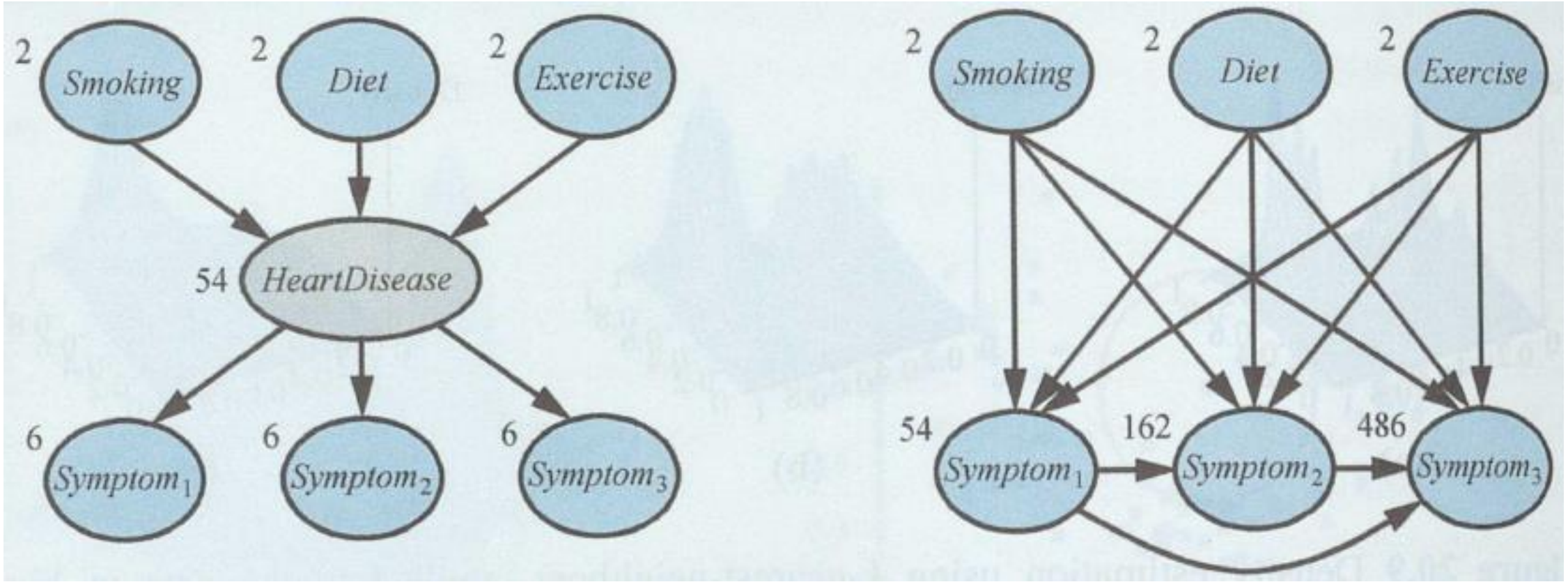
- Till now, all variables were observable
- Many real-world problems have hidden variables (latent variables), which are not observable in the data
- Example: In medical diagnosis, the values of the causes (smoking, diet, exercise) and the effects ($symptom_1$, $symptom_2$, $symptom_3$) are observable, but not disease (HeartDisease), which is a hidden variable



Advantages of Hidden Variables

- If each variable has 3 possible values, the size of the probability tables and the resulting number of parameters in a Bayesian network is shown.

➢With hidden variables (left), the number of necessary parameters to be learnt is substantially smaller than without (right): 78 versus 708 in total



Expectation-Maximization (EM) Algorithm (1)

The EM algorithm is a family of algorithms for learning approximations of the parameters of the hidden variables in a model. Applications include:

- Bayesian networks (see medical example)
- Hidden Markov Models
- Gaussian distributions
- Clustering
- …

Expectation-Maximization (EM) Algorithm (2)

EM iterates between two steps till convergence:

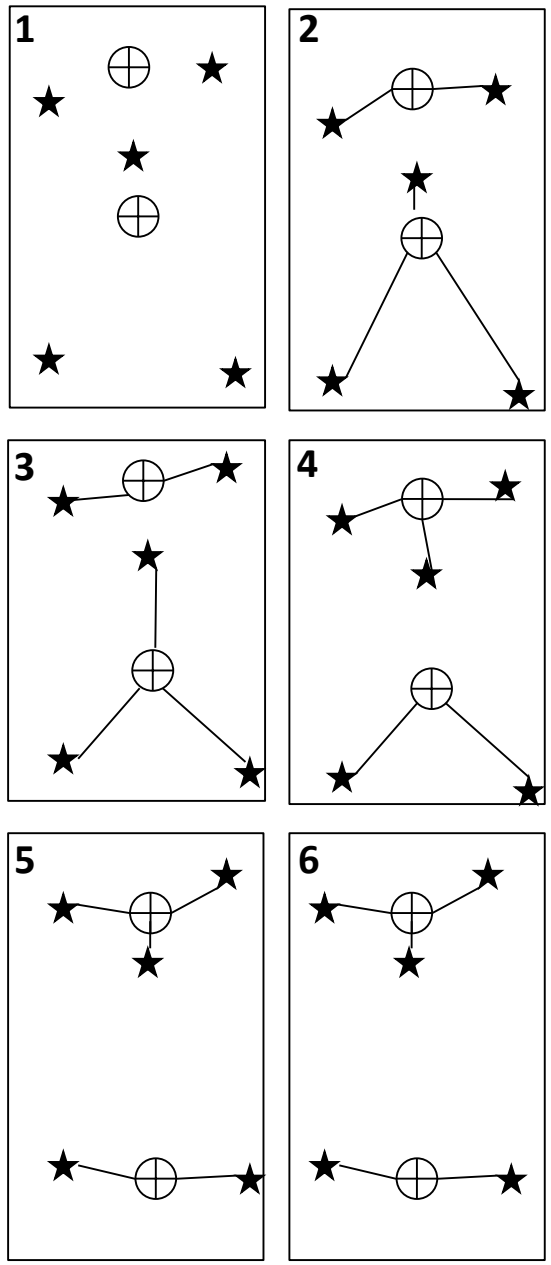
- **Initialisation:** Start with a model, i.e. a specification of all parameters (random or guessed)
- **E-Step (Expectation):** From model and given data, hypothetical data for the hidden variables are computed
- **M-Step (Maximation):** From hypothetical values of hidden variables and given data, new parameters of the model are computed
- **Termination:** Iterate E- and M-step until no relevant changes of the model occur

EM converges to a local maximum, which is often but not always good

- Depends a.o. from the inital parameter guess in the initialization.

Standard Example for EM: Clustering

- **Clustering:**
  - Given: A set of points in an n-dimensional space
  - Sought: A set of c clusters (represented by the cluster centroids)
- **EM-Clustering-Algorithm:**
  1. **Initialization:** Choose c cluster centroids randomly (1)
  2. **Expectation**: Assign each point to the next cluster centroid (2, 4)
  3. **Maximation**: Recompute centroids based on the assigned points (3, 5)
  4. **Terminination:** Repeat E- and M-Steps, until no point changes its centroid (6)
- Variants for computing the cluster centroids:
  - Computing the mean (K-Means-algorithm)
  - Computing the median (K-Median-algorithm)
  - Choosing data points as centroids (K-Mediods-algorithm)
  - Density distribution (DB-SCAN)



Example for EM for Bayesian Networks

- Task: From two bags are 1000 candies taken (s. table). The candies have 3 attributes: Flavor (cherry, lime), Wrapper (red, green), Hole (yes, no). The bags have a different distribution of candy attributes, but it is unknown.
- The following 7 parameters of the model should be learned:

  θ: P(Bag=1)

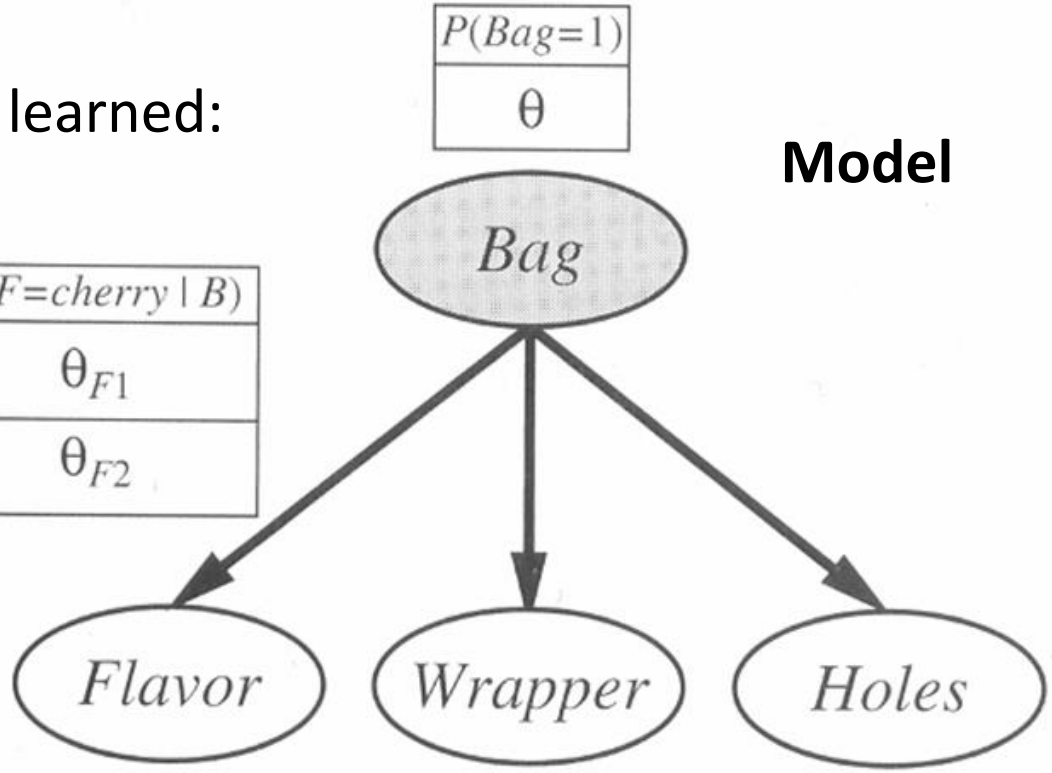
  θ_{F1}: P(F=cherry | Bag1)

  θ_{F2}: P(F=cherry | Bag2)

  θ_{W1}: P(W=red | Bag1) θ_{H1}: P(H=1 | Bag1)

  θ_{W2}: P(W=red | Bag2) θ_{H2}: P(H=1 | Bag2)

Model



Bag	P(F=cherry | B)
1	θ_{F1}
2	θ_{F2}

Data	$W = red$		$W = green$	
	$H = 1$	$H = 0$	$H = 1$	$H = 0$
$F = cherry$	273	93	104	90
$F = lime$	79	100	94	167

Solution with EM (1)

1. **Initialization:** Guess the parametes, e.g. $\theta = 0.6$; $\theta_{F1}=\theta_{W1}=\theta_{H1}=0.6$; $\theta_{F2}=\theta_{W2}=\theta_{H2}=0.4$
2. **Expectation**: Compute, how many candies with different attributes came from bag1
   - Distribute the 273 cherry red candies with hole between bag1 and bag2:
   - bag1: 0.6*0.6*0.6*0.6 = 0.1296
   - bag2: 0.4*0.4*0.4*0.4 = 0.0256
   - proportion bag1 of cherry red candies with hole = 0.1296 / (0.1296+0.0256) ≈ 83,5%
   - Guessed number of cherry red candies with hole from bag1 = 0.835 * 273 ≈ 227,97
   - Repeat for all 8 candy combinations; results for bag1:

bag1	W=red		W=green	
	H=1	H=0	H=1	H=0
F=cherry	227,97	64,38	72,00	45,00
F=lime	54,69	50,00	47,00	51,38

Data	$W = red$		$W = green$	
	$H = 1$	$H = 0$	$H = 1$	$H = 0$
$F = cherry$	273	93	104	90
$F = lime$	79	100	94	167

Results for bag2 (difference of bag1 and data):

bag2	W=red		W=green	
	H=1	H=0	H=1	H=0
F=cherry	45,03	28,62	32,00	45,00
F=lime	24,31	50,00	47,00	115,62

Solution with EM (2)

bag1	W=red		W=green	
	H=1	H=0	H=1	H=0
F=cherry	227,97	64,38	72,00	45,00
F=lime	54,69	50,00	47,00	51,38

3. **Maximization:** From these numbers, recompute the parameters of the model:
   - θ = sum bag1/1000 ≈ 0.61243
   - θ_{F1} = sum (F=cherry from bag1) / sum (bag1) = 409.35/612.43 = 0.6684
   - θ_{W1}= sum (W=red from bag1) / sum (bag1) = 397,05/612,43 = 0.6483
   - θ_{H1}= sum (H=1 from bag1) / sum (bag1) = 401,66/612,43 = 0.6558
   - 1-θ = sum bag2/1000 ≈ 0.3876
   - θ_{F2}= sum (F=cherry from bag2) / sum (bag2) = 150.65/387.57 = 0.3887
   - θ_{W2}= sum (W=red from bag2) / sum (bag2) = 147.95/387.57 = 0.3817
   - θ_{H2}= sum (H=1 from bag2) / sum (bag2) = 148.34/387.57 = 0.3827

	$W = red$		$W = green$	
	$H = 1$	$H = 0$	$H = 1$	$H = 0$
$F = cherry$	273	93	104	90
$F = lime$	79	100	94	167

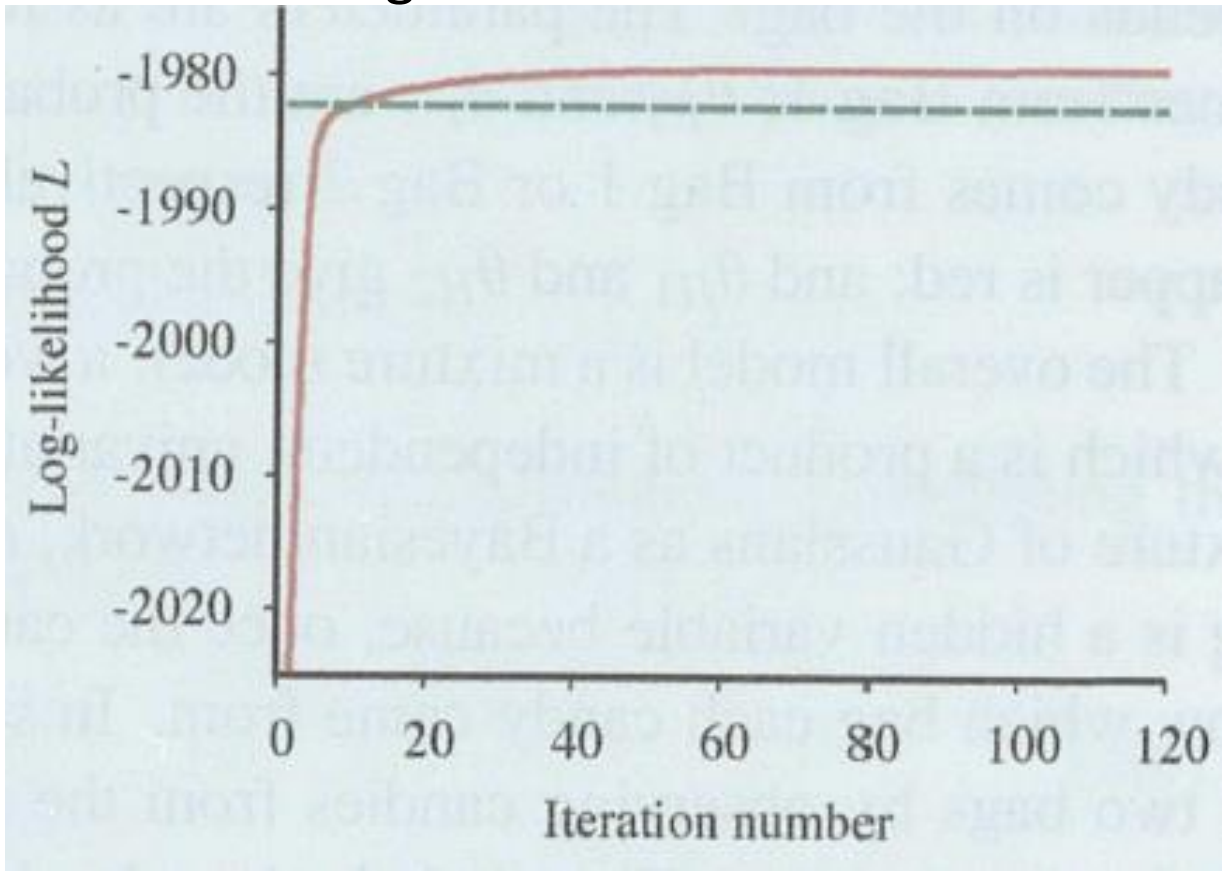
bag2	W=red		W=green	
	H=1	H=0	H=1	H=0
F=cherry	45,03	28,62	32,00	45,00
F=lime	24,31	50,00	47,00	115,62

Solution with EM (3)

- Iterate the expectation and maximization steps until no relevant changes of the model occur

- We can measure the fit of the model to the data by computing the likelihood, that the exact numbers of the data are generated by the model.
- Since this is a very small probability, we take the logarithm of the likelihood (Log-likelihood)
  - Log-likelihood of initial model = -2044
  - Log-likelihood after 1. iteration = -2021
  - Log-likelihood after 10. iteration > -1982
    - Better fit than original model (-1982): $\theta = 0.5$; $\theta_{F1}=\theta_{W1}=\theta_{H1}=0.8$; $\theta_{F2}=\theta_{W2}=\theta_{H2}=0.3$

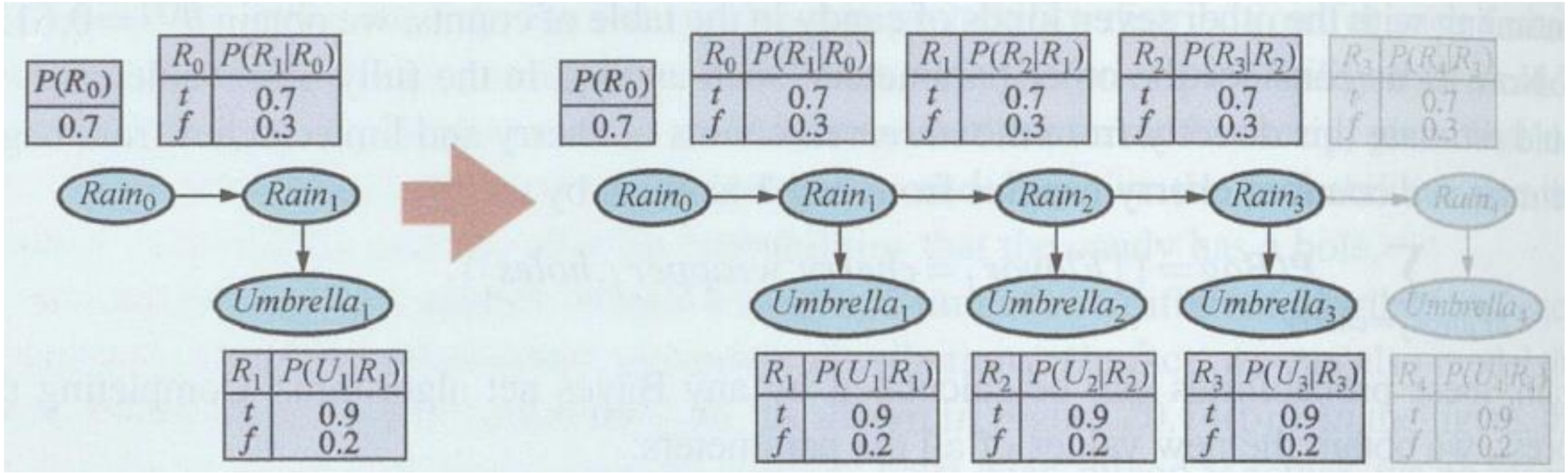
Red line: learned model after n iterations
Green line: original model



Limits of EM Algorithm

- In the example, the EM-algorithm recovered 7 parameters θ, θ_{F1},θ_{W1},θ_{H1}, θ_{F2},θ_{W2},θ_{H2} from 7 (2^3 -1) observed counts in the data (the 8^{th} count can be computed from the others).
- If each candy is described by two attributes (e.g. omitting the holes), 5 parameters (θ, θ_{F1},θ_{W1}, θ_{F2},θ_{W2}) must be derived from 3 (2^2 -1) counts, which is impossible
  - The two-attribute model is not **identifiable**
- Even for the three-attribute model, there are two symmetric solution, if bag1 and bag2 are exchanged. This kind of non-identifiability is unavoidable with variables that are never observed.

EM for Hidden Markov Models



- HMMs can be represented by a dynamic Bayesian net with a single discrete state variable
- When transfering the Bayesian EM algorithm to HMMs, there is one complication: the transition probabilities from state i to state j are repeated across time
- Therefore, as parameter the average of the individual transitions must be computed
- The expected counts are computed by an HMM inference algorithm (smoothing)

General Form of EM-Algorithms

$$\theta^{i+1} = \underset{\theta}{\operatorname{argmax}} \sum_{z} P(Z=z \mid x, \theta^{i}) \; L(x, Z=z \mid \theta)$$

$P(Z=z \mid x, \theta^{i})$: A-Posteriori probabilities of hidden variables

$L(x, Z=z \mid \theta)$: Log Likelihood of the values of hidden variable based on parameters

x = observed values of variables
z = hidden variables
θ = parameter for probability model
L = Log Likelihood: $L(a \mid b) = \log P(a \mid b)$

In practical use, this basic form is varied and the E-step is often approximated (e.g. by MCMC)

E-Step: **Computation of hidden variables Z = z**,
Computation of summation which is the expectation of Log-likelihood of the „completed“ data with respect to parameters and observed data $P(Z=z \mid x, \theta^{i})$

M-Step: **Computation of new values for model parameter θ** by maximization of the expected Log-probabilities for the parameters:

- For Gaussian mixtures: median, variance, (weigths), usw.
- For Bayesian nets: probability tables
- For HMM: transition tables

Learning Bayes Net Structures with Hidden Variables

- Learning Bayes net structures with complete variables is already very difficult
- Learning them with hidden variables is even more difficult
- Simplest case: An expert tell the learning algorithm, that certain hidden variables exist, leaving it to the algorithm to find a place for them in the network structure
  - The overall algorithm is an outer loop that searches over structures and an inner loop that fits the networks parameter given the structure
- If the learning algorithm is not told, which hidden variables variables exist, it has 2 choices:
  - Learn a parameter-intensive model
  - Or invent new hidden variables to simplify the model
  - Metric for explainability of data necessary (problem: overfitting; penalty for complexity)
- Both options have an extremly high computation effort $\rightarrow$ active research area