Lecture 13

Learning in Graphical Models

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Course Overview

- ✔ Introduction
  - ✔ Artificial Intelligence
  - ✔ Intelligent Agents
- ✔ Search
  - ✔ Uninformed Search
  - ✔ Heuristic Search
- ✔ Uncertain knowledge and Reasoning
  - ✔ Probability and Bayesian approach
  - ✔ Bayesian Networks
  - ✔ Hidden Markov Chains
  - ✔ Kalman Filters

- Learning
  - ✔ Supervised
    Decision Trees, Neural Networks
    Learning Bayesian Networks
  - ✔ Unsupervised
    EM Algorithm
- Reinforcement Learning
- Games and Adversarial Search
  - Minimax search and Alpha-beta pruning
  - Multiagent search
- Knowledge representation and Reasoning
  - Propositional logic
  - First order logic
  - Inference
  - Plannning

Outline

Outline

Methods:

1. Bayesian learning
2. Maximum *a posteriori* and maximum likelihood learning

Bayesian networks learning with complete data

a. ML parameter learning
b. Bayesian parameter learning

Full Bayesian learning

- View learning as Bayesian updating of a probability distribution over the hypothesis space
- H hypothesis variable, values $h_1, h_2, \ldots$, prior $\Pr(h)$
- d_j gives the outcome of random variable D_j (the jth observation) training data $\mathbf{d} = d_1, \ldots, d_N$
- Given the data so far, each hypothesis has a posterior probability:

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

  where $P(\mathbf{d}|h_i)$ is called the likelihood

- Predictions use a likelihood-weighted average over the hypotheses:

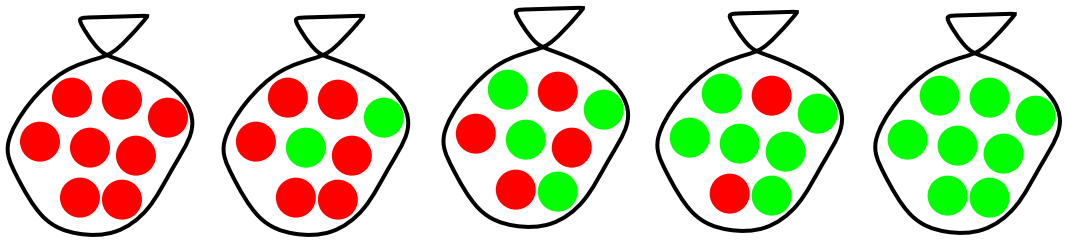
$$\Pr(X|\mathbf{d}) = \sum_i \Pr(X|\mathbf{d}, h_i)P(h_i|\mathbf{d}) = \sum_i \Pr(X|h_i)P(h_i|\mathbf{d})$$

  Or predict according to the most probable hypothesis (maximum a posteriori)

Example

Suppose there are five kinds of bags of candies:

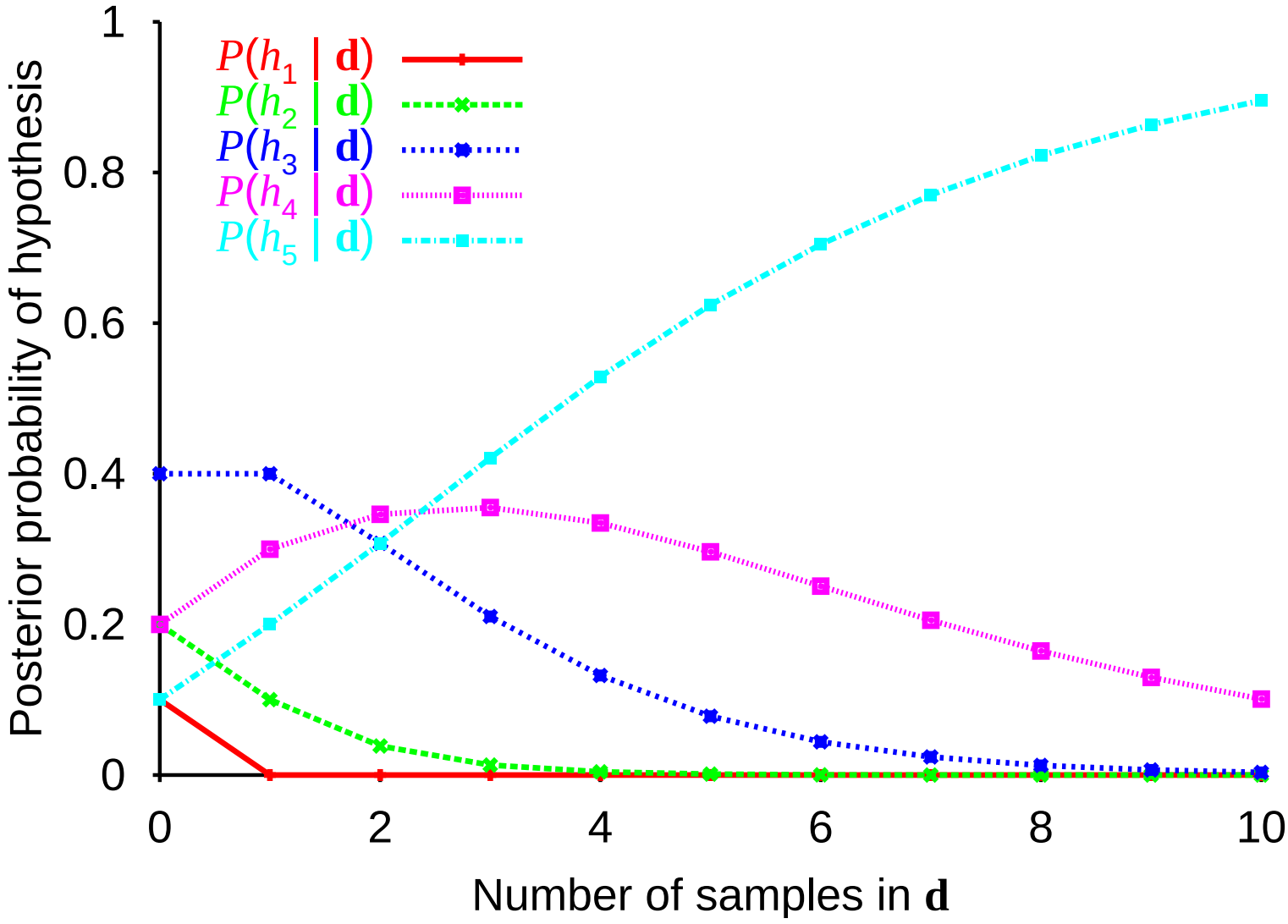
- 10% are h_1: 100% cherry candies
- 20% are h_2: 75% cherry candies + 25% lime candies
- 40% are h_3: 50% cherry candies + 50% lime candies
- 20% are h_4: 25% cherry candies + 75% lime candies
- 10% are h_5: 100% lime candies



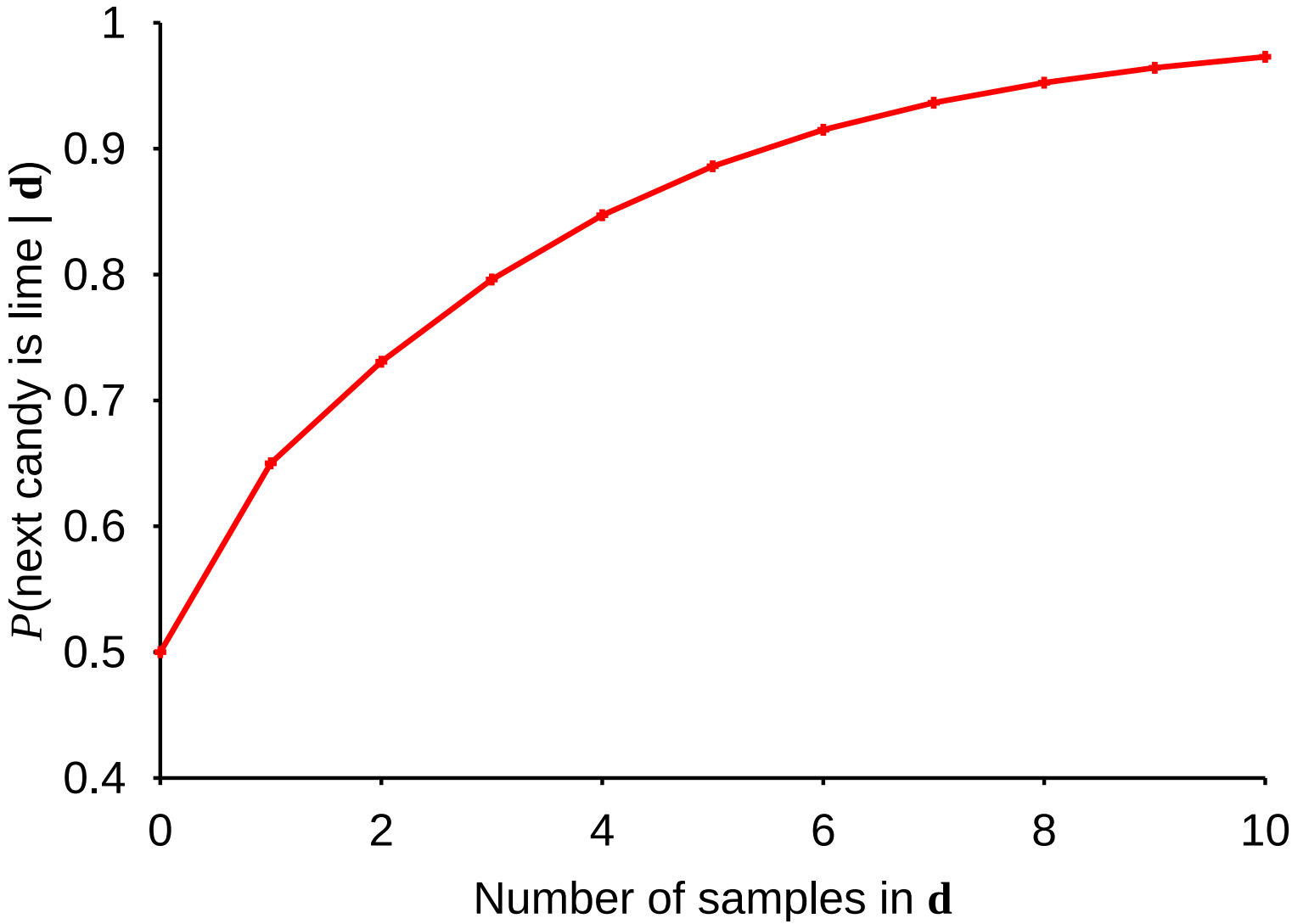
Then we observe candies drawn from some bag: 

What kind of bag is it? What flavour will the next candy be?

Posterior probability of hypotheses



Prediction probability



MAP approximation

- Summing over the hypothesis space is often intractable
  (e.g., 18,446,744,073,709,551,616 Boolean functions of 6 attributes)

- Maximum a posteriori (MAP) learning: choose h_{MAP} maximizing $P(h_i|\mathbf{d})$
  I.e., maximize $P(\mathbf{d}|h_i)P(h_i)$ or $\log P(\mathbf{d}|h_i) + \log P(h_i)$
  Log terms can be viewed as (negative of)
  bits to encode data given hypothesis + bits to encode hypothesis
  This is the basic idea of minimum description length (MDL) learning

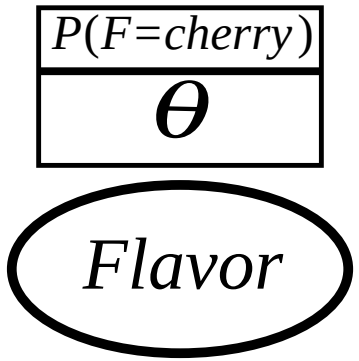
- For deterministic hypotheses, $P(\mathbf{d}|h_i)$ is 1 if consistent, 0 otherwise
  $\Longrightarrow$ MAP = simplest consistent hypothesis

ML approximation

- For large data sets, prior becomes irrelevant
- Maximum likelihood (ML) learning: choose h_{ML} maximizing $P(\mathbf{d}|h_i)$
  I.e., simply get the best fit to the data; identical to MAP for uniform prior
  (which is reasonable if all hypotheses are of the same complexity)
- ML is the "standard" (non-Bayesian) statistical learning method

Parameter learning by ML

Bag from a new manufacturer; fraction θ of cherry candies?



Any θ is possible: continuum of hypotheses h_θ
θ is a parameter for this simple (binomial) family of models

Suppose we unwrap N candies, c cherries and $\ell = N - c$ limes
These are i.i.d. (independent, identically distributed) observations, so

$$P(\mathbf{d}|h_\theta) = \prod_{j=1}^{N} P(d_j|h_\theta) = \theta^c \cdot (1-\theta)^\ell$$

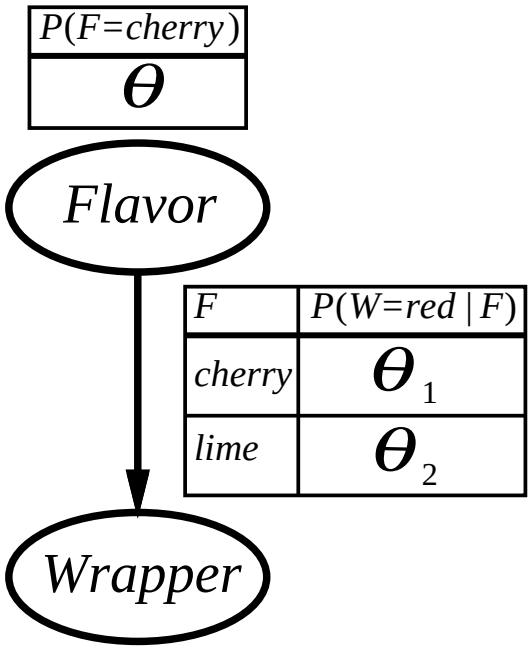
Maximize this w.r.t. θ—which is easier for the log-likelihood:

$$L(\mathbf{d}|h_\theta) = \log P(\mathbf{d}|h_\theta) = \sum_{j=1}^{N} \log P(d_j|h_\theta) = c\log\theta + \ell\log(1-\theta)$$

$$\frac{dL(\mathbf{d}|h_\theta)}{d\theta} = \frac{c}{\theta} - \frac{\ell}{1-\theta} = 0 \quad \Longrightarrow \quad \theta = \frac{c}{c+\ell} = \frac{c}{N}$$

Seems sensible, but causes problems with 0 counts!

Multiple parameters



Red/green wrapper depends probabilistically on flavor:
Likelihood for, e.g., cherry candy in green wrapper:

$$\begin{aligned} P(F = cherry, W = green | h_{\theta,\theta_1,\theta_2}) \\ = \quad P(F = cherry | h_{\theta,\theta_1,\theta_2}) P(W = green | F = cherry \\ = \quad \theta \cdot (1 - \theta_1) \end{aligned}$$

N candies, r_c red-wrapped cherry candies, etc.:

$$P(\mathbf{d} | h_{\theta,\theta_1,\theta_2}) = \theta^c (1-\theta)^\ell \cdot \theta_1^{r_c} (1-\theta_1)^{g_c} \cdot \theta_2^{r_\ell} (1-\theta_2)^{g_\ell}$$

$$\begin{aligned} L &= [c \log \theta + \ell \log(1-\theta)] \\ &+ [r_c \log \theta_1 + g_c \log(1-\theta_1)] \\ &+ [r_\ell \log \theta_2 + g_\ell \log(1-\theta_2)] \end{aligned}$$

Multiple parameters contd.

Derivatives of L contain only the relevant parameter:

$$\frac{\partial L}{\partial \theta} = \frac{c}{\theta} - \frac{\ell}{1-\theta} = 0 \qquad \Longrightarrow \quad \theta = \frac{c}{c+\ell}$$

$$\frac{\partial L}{\partial \theta_1} = \frac{r_c}{\theta_1} - \frac{g_c}{1-\theta_1} = 0 \qquad \Longrightarrow \quad \theta_1 = \frac{r_c}{r_c+g_c}$$

$$\frac{\partial L}{\partial \theta_2} = \frac{r_\ell}{\theta_2} - \frac{g_\ell}{1-\theta_2} = 0 \qquad \Longrightarrow \quad \theta_2 = \frac{r_\ell}{r_\ell+g_\ell}$$

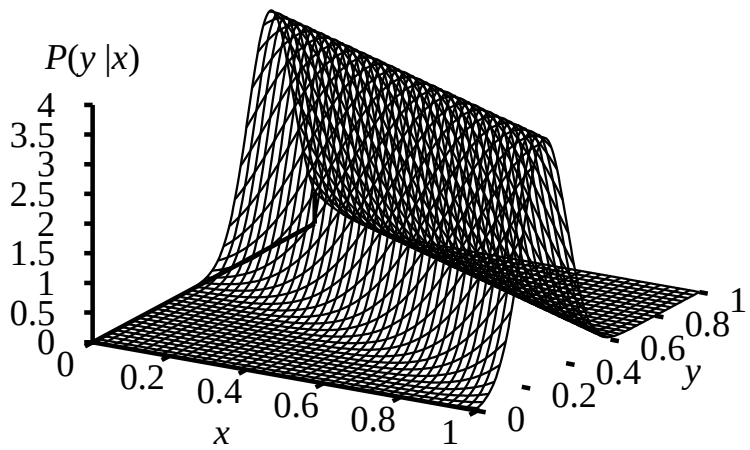
With complete data, **parameters can be learned separately**

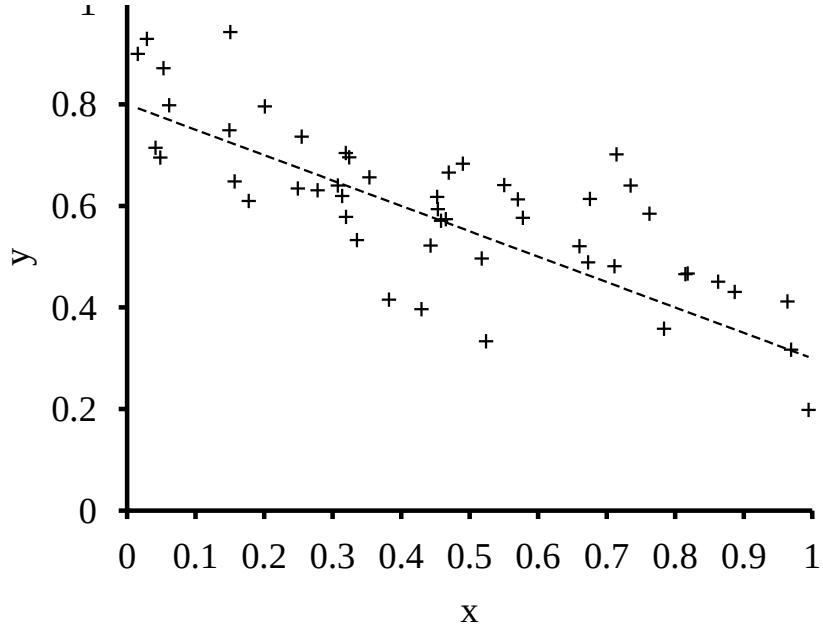
Continuous models

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Parameters μ and σ^2
Maximum likelihood:

Continuous models, Multiple param.





Maximizing $P(y|x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y-(\theta_1 x+\theta_2))^2}{2\sigma^2}}$ w.r.t. θ_1, θ_2

= minimizing $E = \sum_{j=1}^{N} (y_j - (\theta_1 x_j + \theta_2))^2$

That is, minimizing the sum of squared errors gives the ML solution for a linear fit **assuming Gaussian noise of fixed variance**

Summary

- Full Bayesian learning gives best possible predictions but is intractable
- MAP learning balances complexity with accuracy on training data
- Maximum likelihood assumes uniform prior, OK for large data sets

1. Choose a parameterized family of models to describe the data
   requires substantial insight and sometimes new models
2. Write down the likelihood of the data as a function of the parameters
   may require summing over hidden variables, i.e., inference
3. Write down the derivative of the log likelihood w.r.t. each parameter
4. Find the parameter values such that the derivatives are zero
   may be hard/impossible; gradient techniques help

Bayesian Parameter Learning

If small data set the ML method leads to premature conclusions:
From the Flavor example:

$$P(\mathbf{d}|h_\theta) = \prod_{j=1}^{N} P(d_j|h_\theta) = \theta^c \cdot (1-\theta)^\ell \qquad \Longrightarrow \qquad \theta = \frac{c}{c+\ell}$$

If $N = 1$ and $c = 1$, $l = 0$ we conclude $\theta = 1$. Laplace adjustment can mitigate this result but it is artificial.

Bayesian approach:

$$P(\theta|\mathbf{d}) = \alpha P(\mathbf{d}|\theta)P(\theta)$$

we saw the likelihood to be

$$p(X = 1|\theta) = \text{Bern}(\theta) = \theta$$

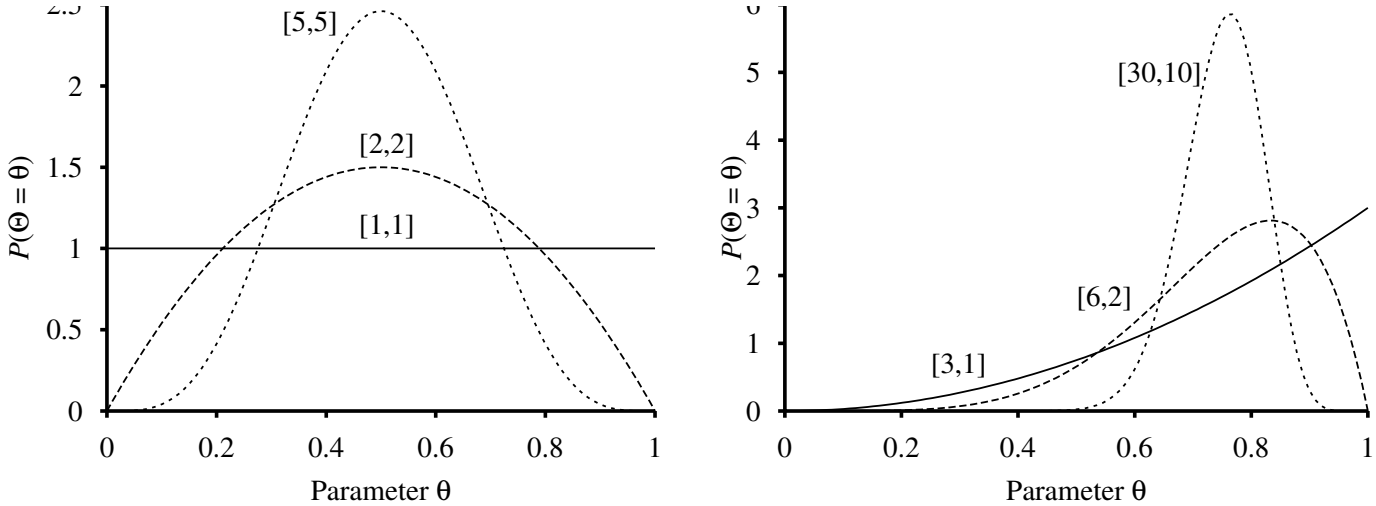
which is known as Bernoulli distribution. Further, for a set of n observed outcomes $\mathbf{d} = (x_1, \ldots, x_n)$ of which s are 1s, we have the **binomial sampling model**:

$$p(\mathbf{D} = \mathbf{d}|\theta) = p(s|\theta) = \text{Bin}(s|\theta) = \binom{n}{s}\theta^s(1-\theta)^{n-s} \quad (1)$$

The Beta Distribution

We define the prior probability $p(\theta)$ to be Beta distributed

$$p(\theta) = \text{Beta}(\theta|a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\theta^{a-1}(1-\theta)^{b-1}$$



Reasons for this choice:

- provides flexiblity varying the hyperparameters a and b
  Eg. the uniform distribution is included in this family with $a = 1$, $b = 1$
- conjugancy property

Eg: we observe $N = 1$, $c = 1$, $l = 0$:

$$\begin{aligned} p(\theta|d) &= \alpha p(\mathbf{d}|\theta)p(\theta) \\ &= \alpha \mathrm{Bin}(d|\theta)p(\theta) \\ &= \alpha \mathrm{Beta}(\theta|a+c, b+l). \end{aligned}$$

In Presence of Parents

- Denote by $\mathbf{Pa}_i^j$ the jth parent variable/node of X_i

$$p(x_i|\mathbf{pa}_i^j, \boldsymbol{\theta}_i) = \theta_{ij},$$

where $\mathbf{pa}_i^1, \ldots, \mathbf{pa}_i^{q_i}$, $q_i = \prod_{X_i \in \mathbf{Pa}_i} r_i$, denote the configurations of $\mathbf{Pa}_i$, and $\boldsymbol{\theta}_i = (\theta_{ij})$, $j = 1, \ldots, q_i$, are the local parameters of variable i.

- In the case of no missing values, that is, all variables of the network have a value in the random sample $\mathbf{d}$, and independence among parameters, the parameters remain independent given $\mathbf{d}$, that is,

$$p(\boldsymbol{\theta}|\mathbf{d}) = \prod_{i=1}^{d} \prod_{j=1}^{q_i} p(\theta_{ij}|\mathbf{d})$$

- In other terms, we can update each vector parameter $\boldsymbol{\theta}_{ij}$ independently, just as in the one-variable case. Assuming each vector has the prior distribution $\mathrm{Beta}(\theta_{ij}|a_{ij}, b_{ij})$, we obtain the posterior distribution

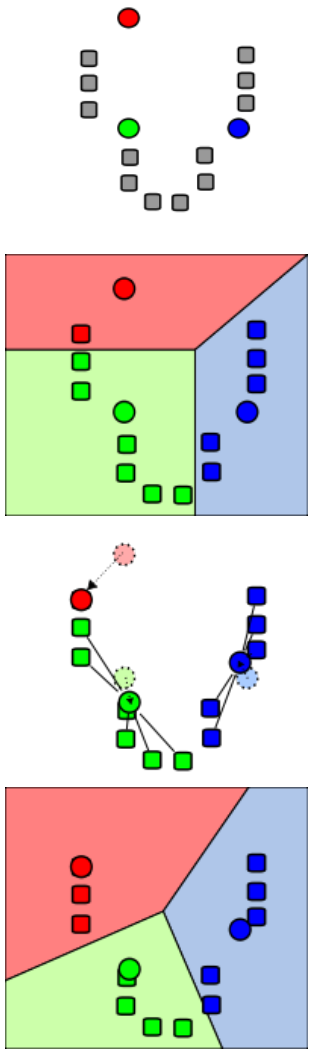
$$p(\theta_{ij}|\mathbf{d}) = \mathrm{Beta}(\theta_{ij}|a_{ij} + s_{ij}, b_{ij} + n - s_{ij})$$

where s_{ij} is the number of cases in $\mathbf{d}$ in which $X_i = 1$ and $\mathbf{Pa}_i = \mathbf{pa}_i^j$.

Outline

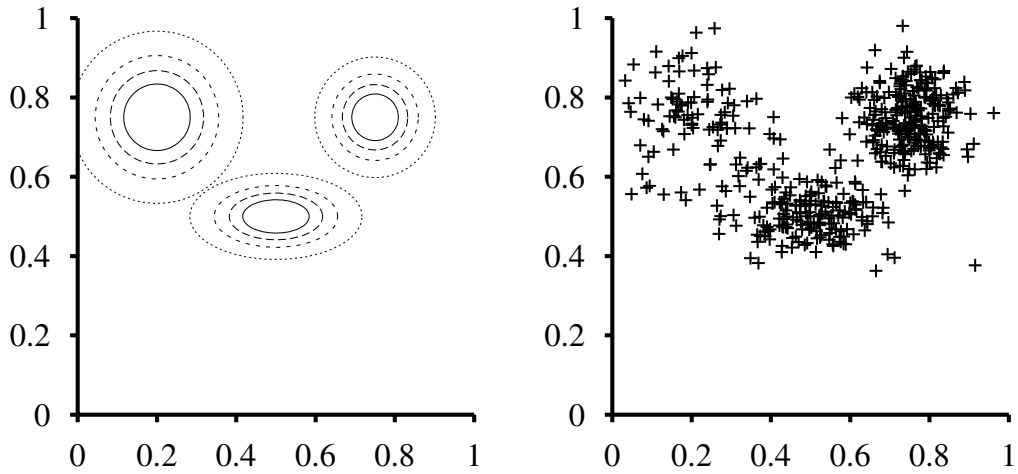
K-means clustering

Init: select k cluster centers at random
repeat
  assign data to nearest center.
  update cluster center to the centroid of assigned data points
until no change ;



Expectation-Maximization Algorithm

Generalization of k-means that uses soft assignments



Mixture model: exploit an hidden variable $\mathbf{z}$

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}) = \sum_{\mathbf{z}} p(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z})$$

Both $p(\mathbf{x} \mid \mathbf{z})$ and $p(\mathbf{z})$ are unknown:

- assume $p(\mathbf{x} \mid \mathbf{z})$ is multivariate Gaussian distribution $N(\mu_i, \sigma_i)$
- assume $p(\mathbf{z})$ is multinomial distribution with parameter θ_i

$\rightsquigarrow$ $\mu_i, \sigma_i, \theta_i$ are unkown

E-step: Assume we know $\mu_i, \sigma_i, \theta_i$,
calculate for each sample j the probability of coming from i

$$p_{ij} = \alpha\theta_i(2\pi)^{-N/2}|\Sigma|^{-1}\exp\{-1/2(\mathbf{x}-\boldsymbol{\mu})\Sigma(\mathbf{x}-\boldsymbol{\mu})^T\}$$

M-step: update $\mu_i, \sigma_i, \theta_i$:

$$\pi_i = \sum_j \frac{p_{ij}}{N}$$

$$\mu_i = \sum_j \frac{p_{ij}x_j}{\sum_j p_{ij}}$$

$$\Sigma_i = \frac{\sum_j p_{ij}(x_j-\mu_i)(x_j-\mu_j)^T}{\sum_j p_{ij}}$$

- the ML method on $\prod_j p(\mathbf{x}_j \mid \mu_i, \sigma_i, \theta_i)$ does not lead to a closed form. Hence we need to proceed by assuming values for some parameters and deriving the others as a consequence of these choices.
- The procedure finds local optima
- It can be proven that the procedure converges
- p_{ij} are soft guesses as opposed to hard links in the k-means algorithm