Probabilistic Models

Ludwig Krippahl

Probabilistic Models

Summary

- Probabilistic Clustering
- Graphical Models
- Markov Processes
- Hidden Markov Models

- Baum-Welch and Viterbi algorithms

- Lots of equations

- But it's the idea that matters

- Next tutorials and lectures

Probabilistic Models

Probabilistic Clustering

- Suppose we have a probabilistic model for the distribution of examples X:

$$P(X, Y) = P(X|Y)P(Y)$$

- This would not only provide a way of clustering X over Y but also to generate examples from that distribution.
- The problem is that we do not know Y

Probabilistic Clustering

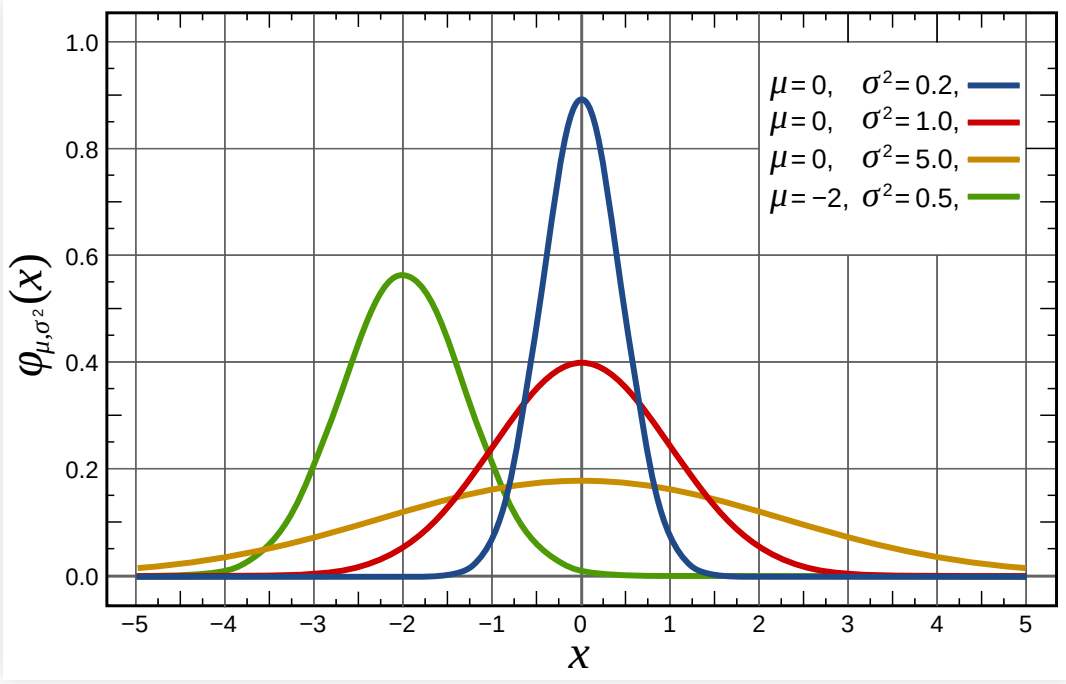
Recall, from EM:

- Set X of observed data
- Set Z of missing data or latent variables
- Unknown parameters θ
- Likelihood function $L(\theta; X, Z) = p(X, Z|\theta)$

Probabilistic Clustering

- Let us assume Gaussian distributions. In 1D:

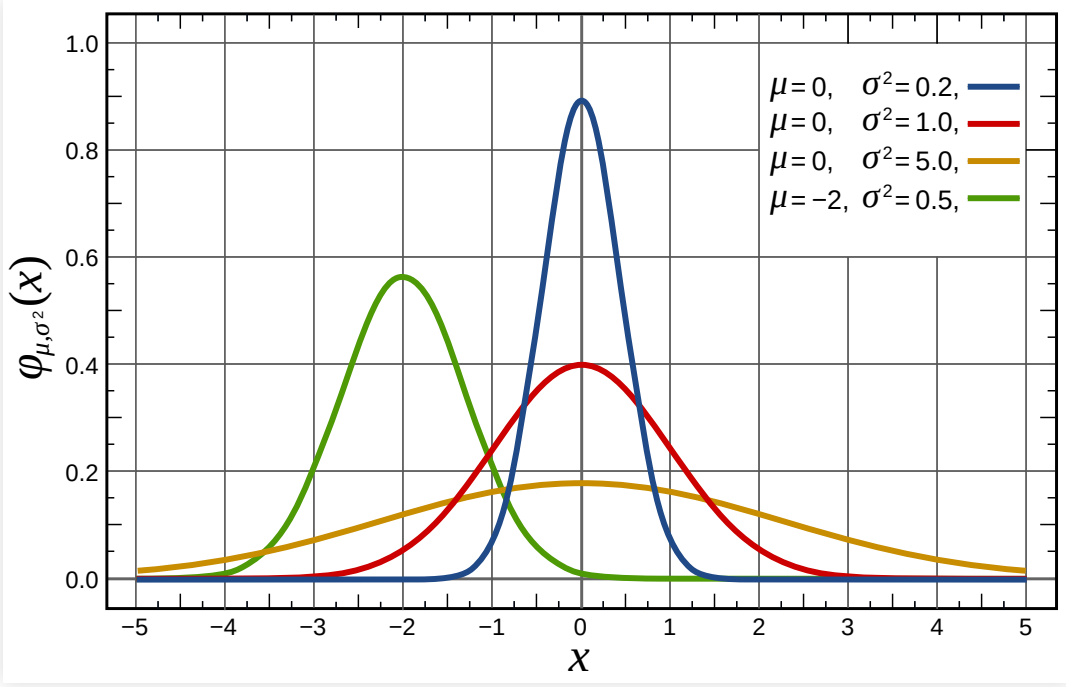
$$\mathcal{N}(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



Source: Wikimedia, public domain

Probabilistic Clustering

- We can create a mixture of gaussians by adding the different distributions with weights



Source: Wikimedia, public domain

Probabilistic Clustering

- For multivariate Gaussian distributions:

$$\mathcal{N}(x|\mu, \Sigma) = \frac{1}{(2\pi|\Sigma|)^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

  - Where μ is the vector of means and Σ the covariance matrix

- A a mixture model of k gaussians is:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$$

- where π_k is the mixing coefficient, such that

$$\sum_{k=1}^{K} \pi_k = 1 \qquad 0 \leq \pi_k \leq 1 \; \forall k$$

Probabilistic Clustering

- If we create a variable $\mathbf{z}$ so for each $\mathbf{x}$ that:

$$z_k \in \{0, 1\} \qquad \sum_k z_k = 1$$

- And we define the distribution:

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

- (z_k are the latent variables assigning each example to each cluster)
- The marginal probability $p(z_k = 1) = \pi_k$
- And $p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)$
- Which brings us back to:

$$p(\mathbf{x}) = \sum_z p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)$$

- but now explicitly with the latent variables

Probabilistic Clustering

- Using Bayes rule and the previous equations, we get the posterior prob. that k responsible for x_n (we'll call it $\gamma(z_{nk})$):

$$p(z_k = 1|\mathbf{x}) = \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\mathbf{x}|z_j = 1)} = \frac{\pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\mu_j, \Sigma_j)} = \gamma(z_{nk})$$

- Since we want to maximize the log-likelihood of our parameters:

$$\ln p(\mathbf{X}|\pi, \mu, \mathbf{\Sigma}) = \sum_{n=1}^{N} \ln\left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k) \right)$$

- We want to find the maximum, where the derivative is 0

Probabilistic Clustering

- Setting to zero the derivative w.r.t. μ_k:

$$0 = -\sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\mu_j, \Sigma_j)} \sum_k (\mathbf{x}_n - \mu_k)$$

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n$$

- Where N_k is the effective number of points in component k:

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

Probabilistic Clustering

- Setting to zero the derivative w.r.t. μ_k:

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n$$

- Doing the same for Σ_k:

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) (\mathbf{x}_n - \mu_k)(\mathbf{x}_n - \mu_k)^T$$

- Which is the sum of the covariances weighted by the $\gamma(z_{nk})$
- Finally, the π_k:

$$\pi_k = \frac{N_k}{N}$$

Probabilistic Clustering

- Now we do EM. First, expectation, computing the posterior probabilities $\gamma(z_{nk})$ from an initial guess of the parameters π, μ, Σ:

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\mu_j, \Sigma_j)}$$

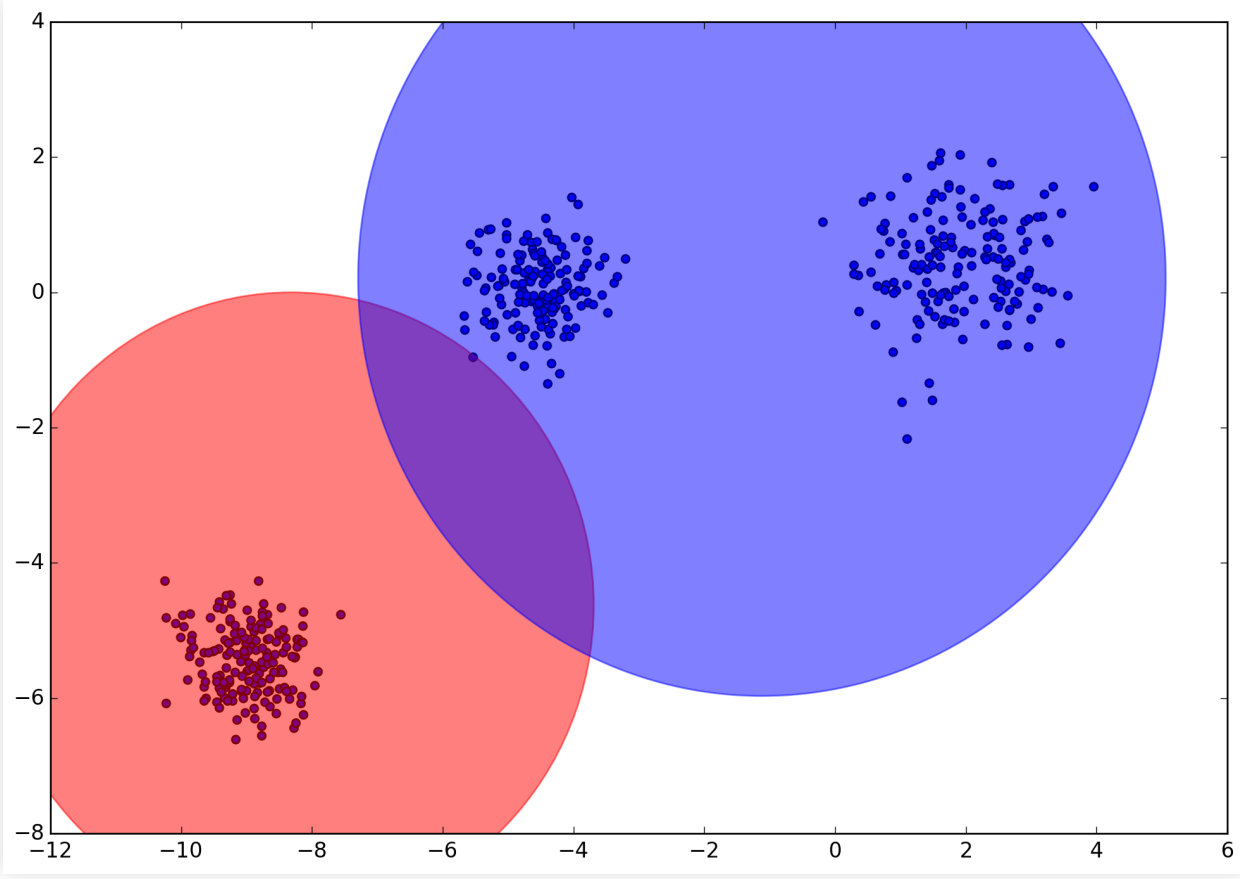
- Then we use the posterior $\gamma(z_{nk})$ in the likelihood function and maximize w.r.t. π, μ, Σ:

$$\pi_k = \frac{N_k}{N} \qquad \mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \qquad \Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \mu_k)$$

- And repeat until convergence...

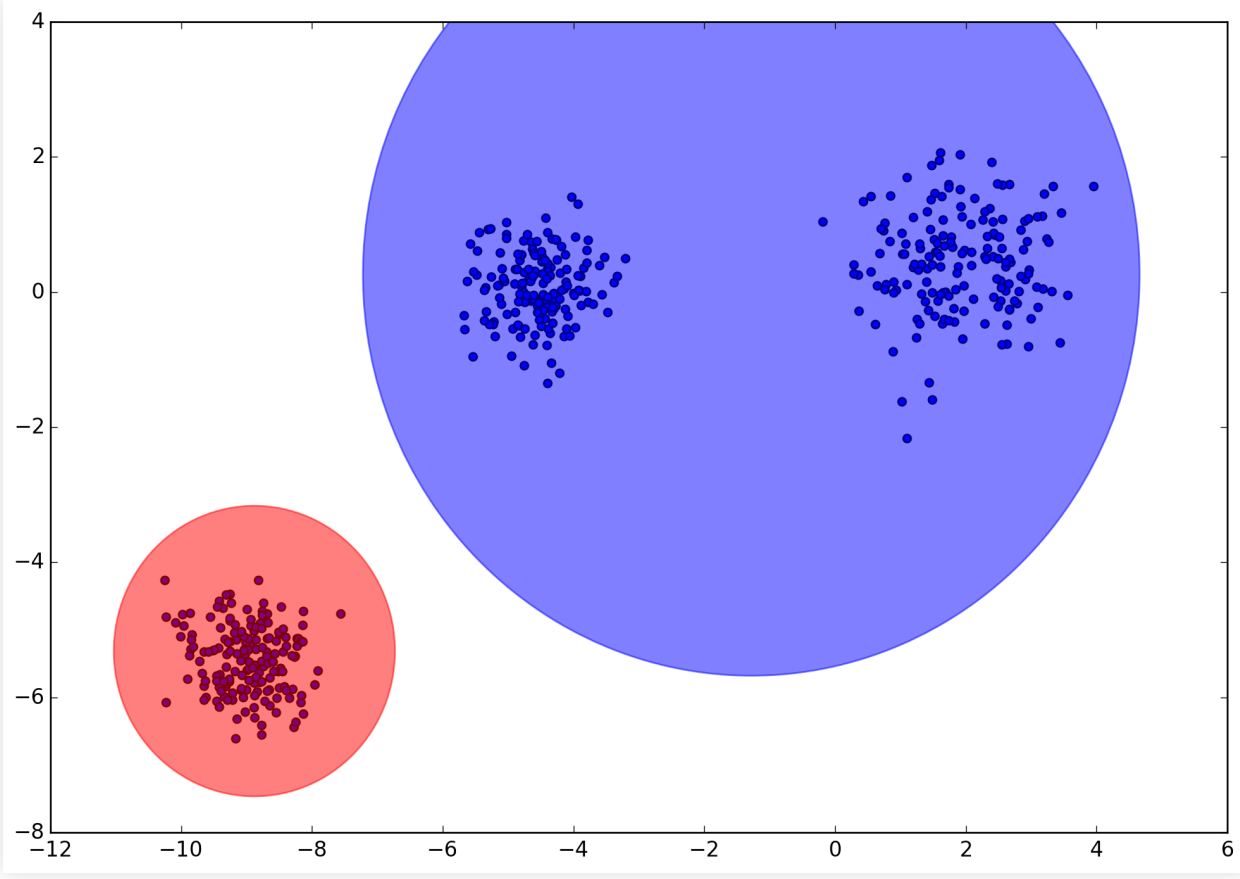
Probabilistic Clustering

- Example, 2 Gaussian components, after 1 pass:



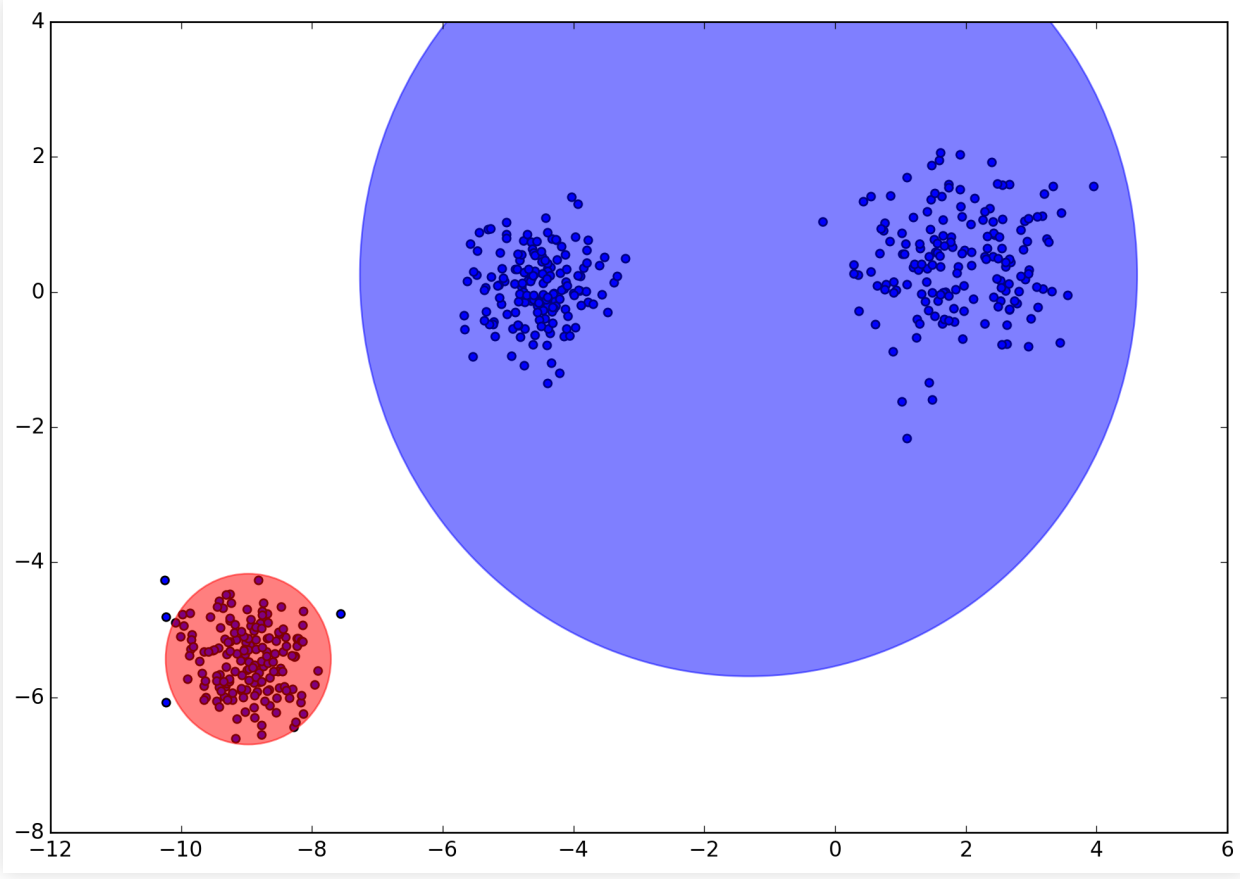
Probabilistic Clustering

- Example, 2 Gaussian components, after 2 passes:



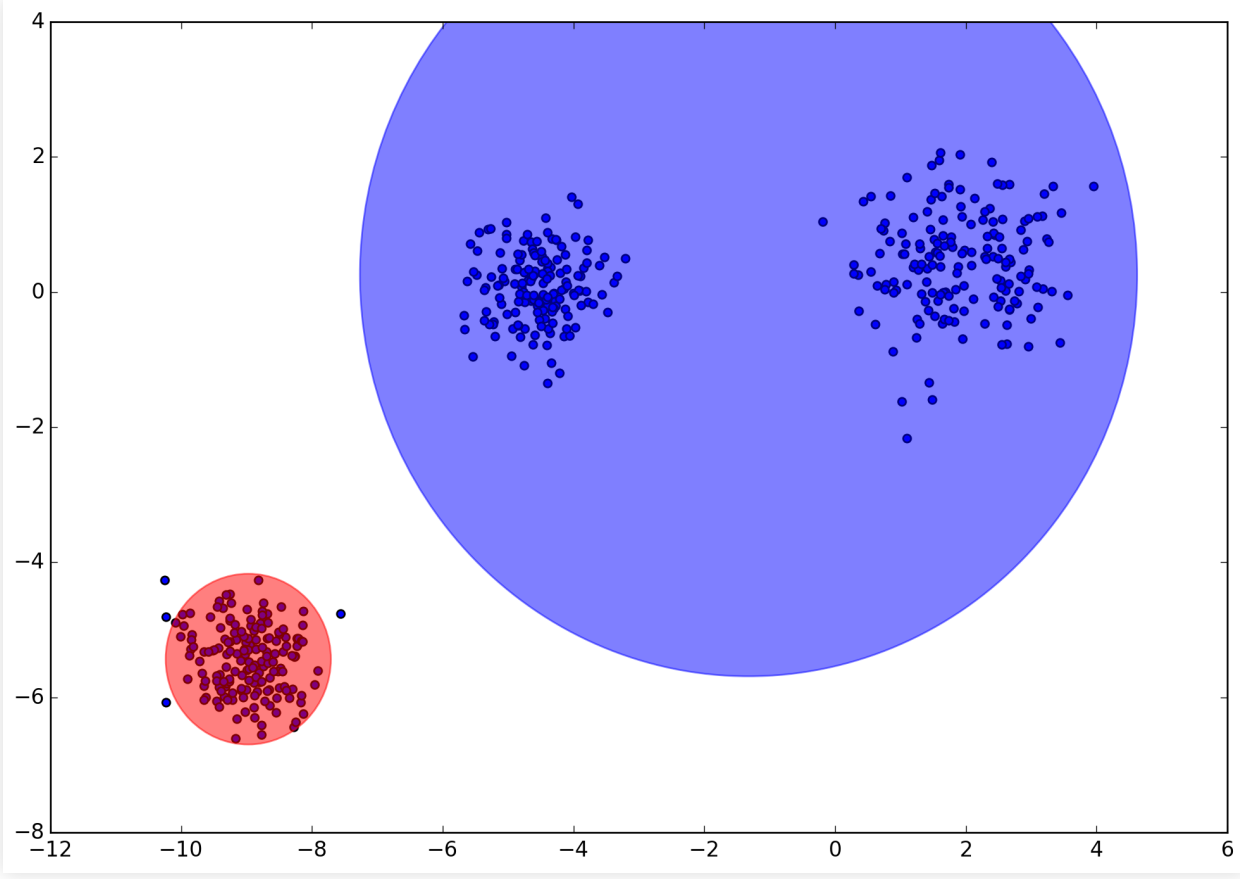
Probabilistic Clustering

- Example, 2 Gaussian components, after 3 passes:



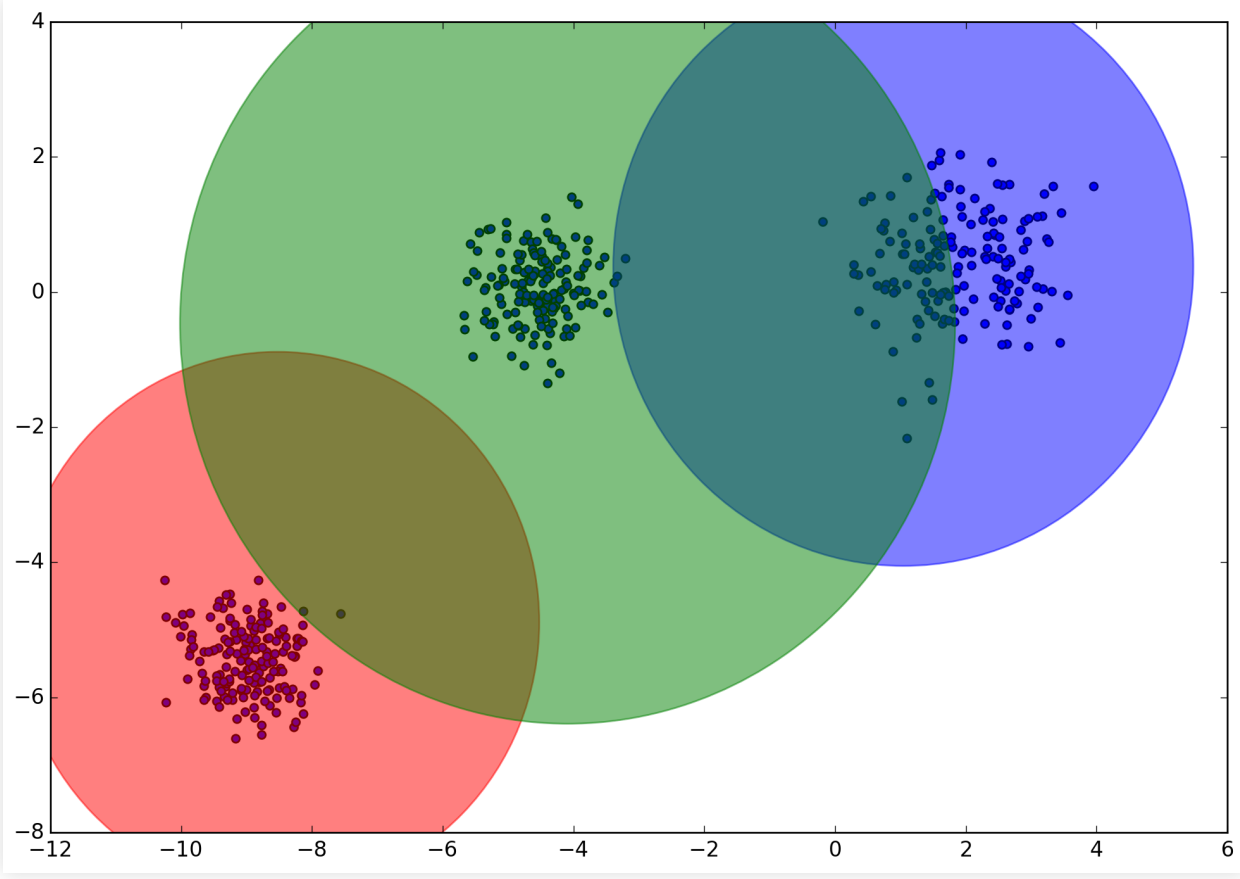
Probabilistic Clustering

- 2 Gaussian components not good for these data



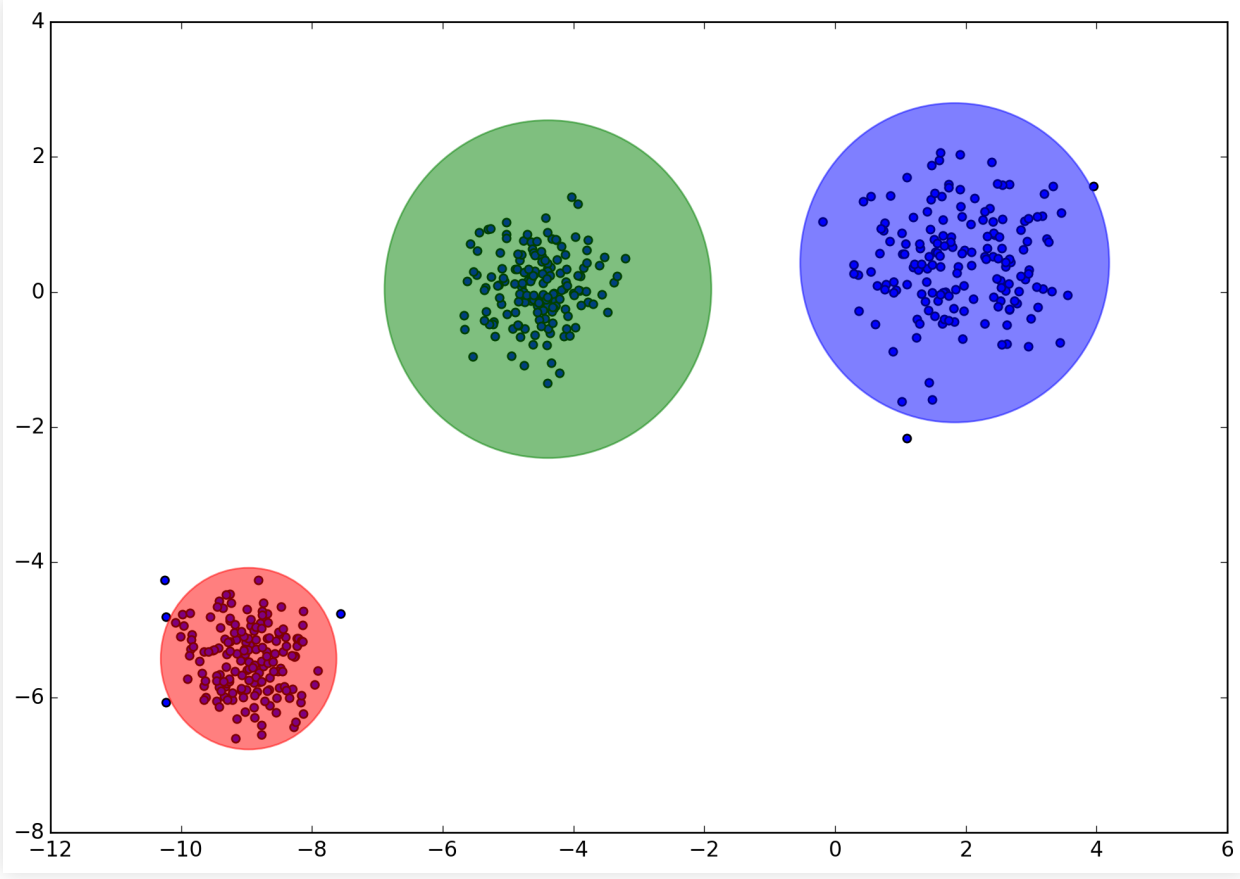
Probabilistic Clustering

- Example, 3 Gaussian components, after 1 pass:



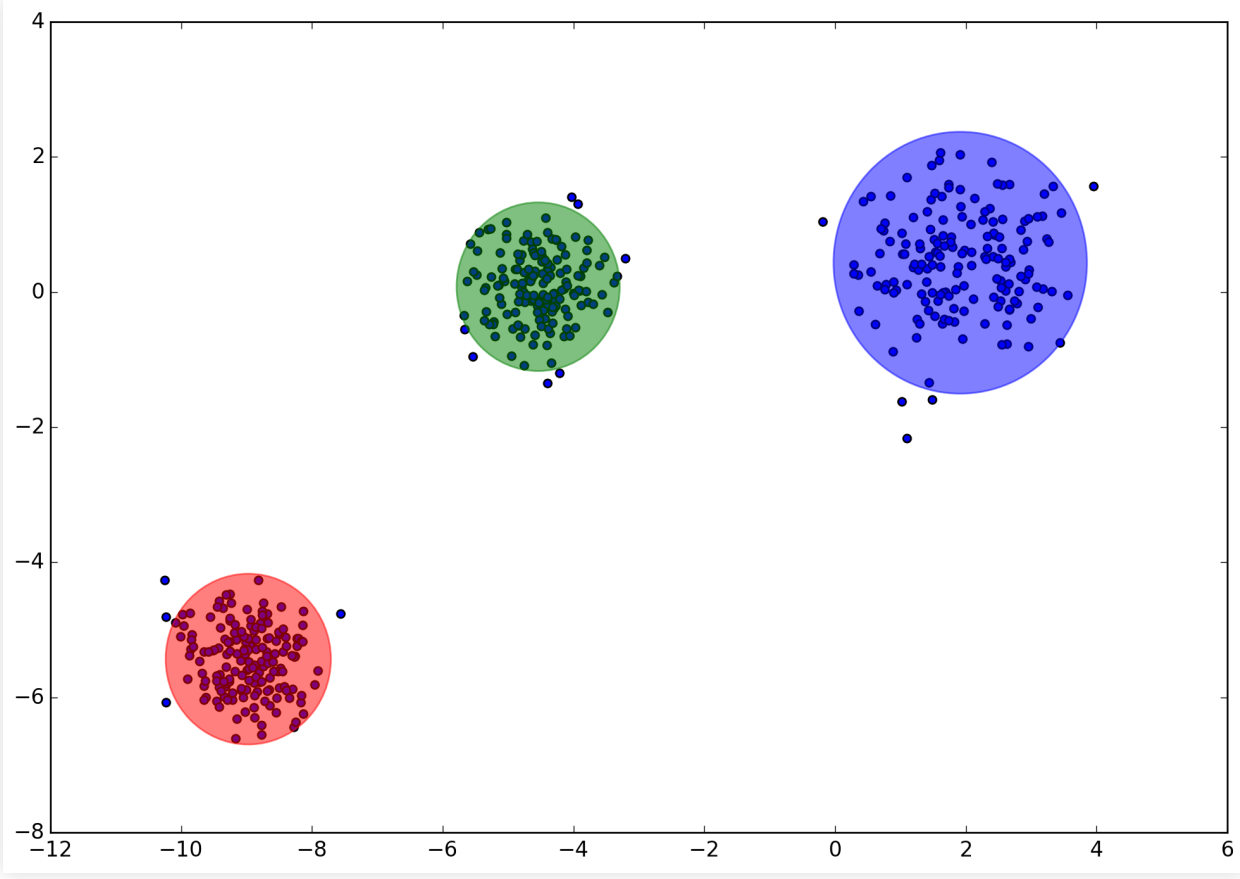
Probabilistic Clustering

- Example, 3 Gaussian components, after 2 passes:



Probabilistic Clustering

- Example, 3 Gaussian components, after 3 passes:



Gaussian mixtures with Scikit-Learn:

```
class sklearn.mixture.GaussianMixture:
#arguments
n_components=1
covariance_type='diag' #constraints on covariance
n_iter=100

#attributes
weights_
means_
covars_
```

Graphical Models

Probabilistic Graphical Models

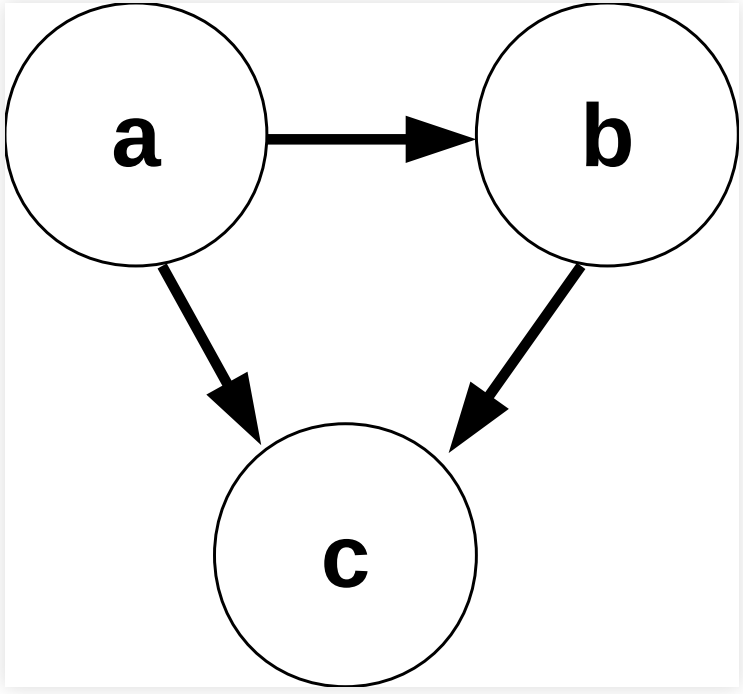
- Convenient representation of probabilistic relations

- Bayesian networks : directed acyclic graphs representing conditional probability dependencies
- Markov Random Fields : undirected graphs representing dependencies
- Dynamic Bayesian networks : Bayesian networks representing time series

Bayesian networks

- Variables, a, b, c, factorize joint probability

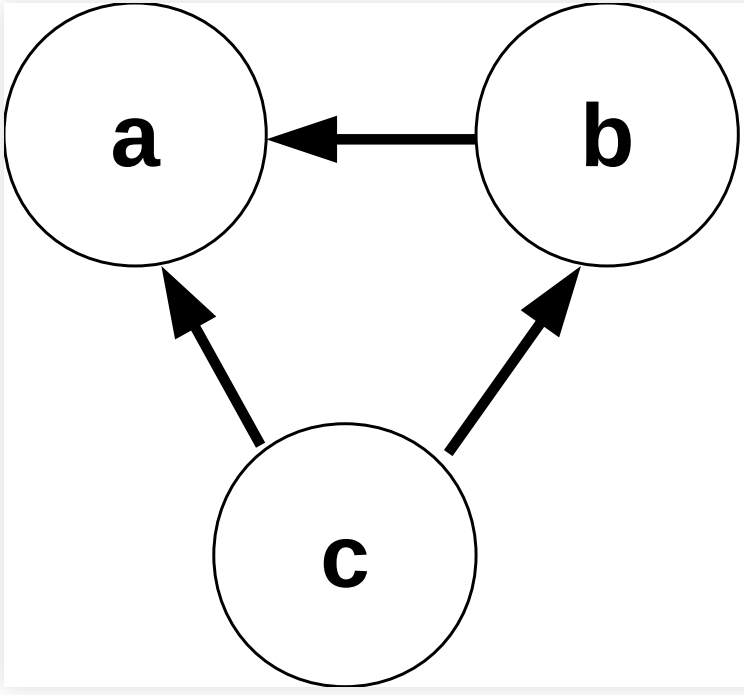
$$P(a, b, c) = P(c|a, b)P(b|a)P(a)$$



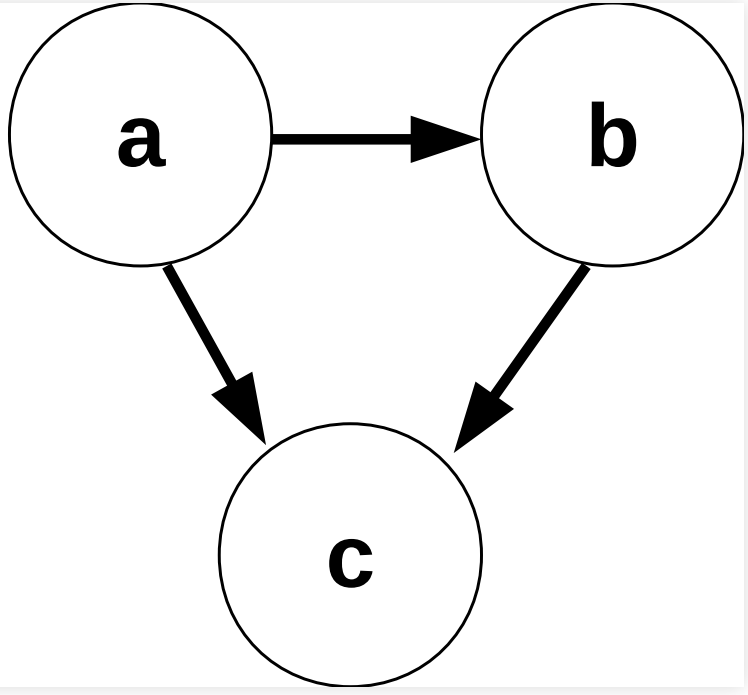
Bayesian networks

- Different possible factorizations:

$$P(a,b,c) = P(a|b,c)P(b|c)P(c)$$

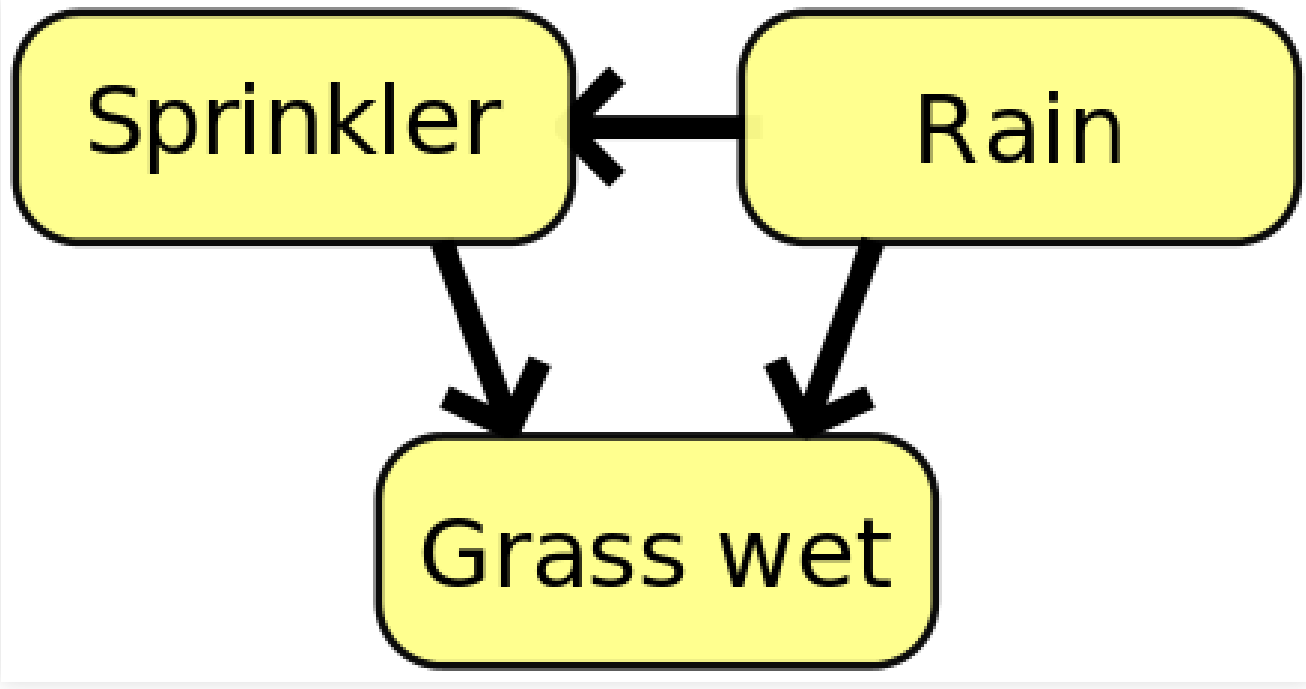


$$P(a,b,c) = P(c|a,b)P(b|a)P(a)$$



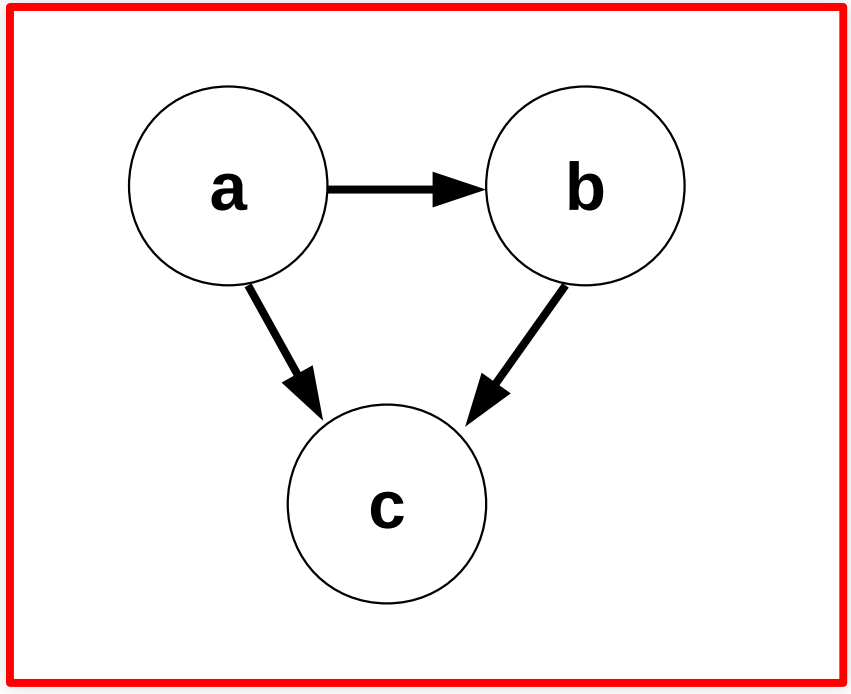
Bayesian networks

- We can use it to represent causal knowledge:



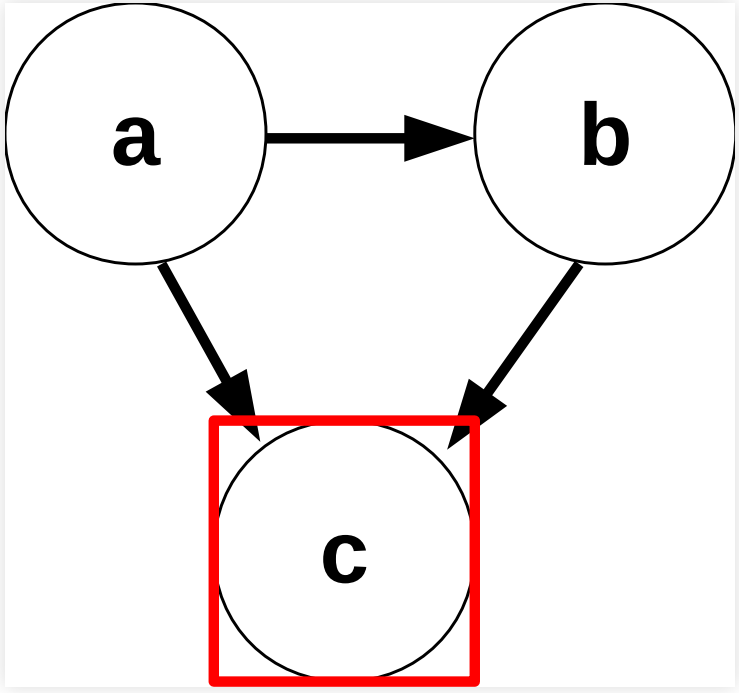
Bayesian networks

- The graph:



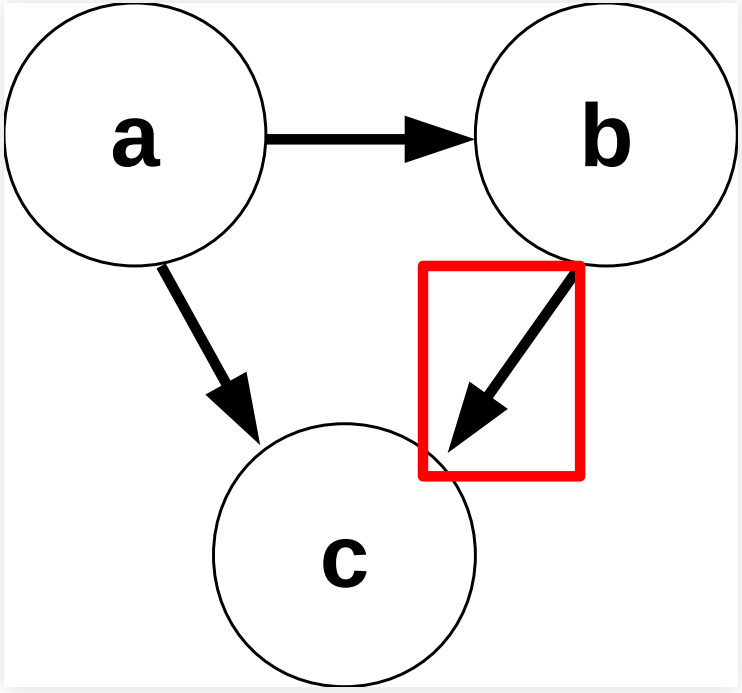
Bayesian networks

- Node, one variable:



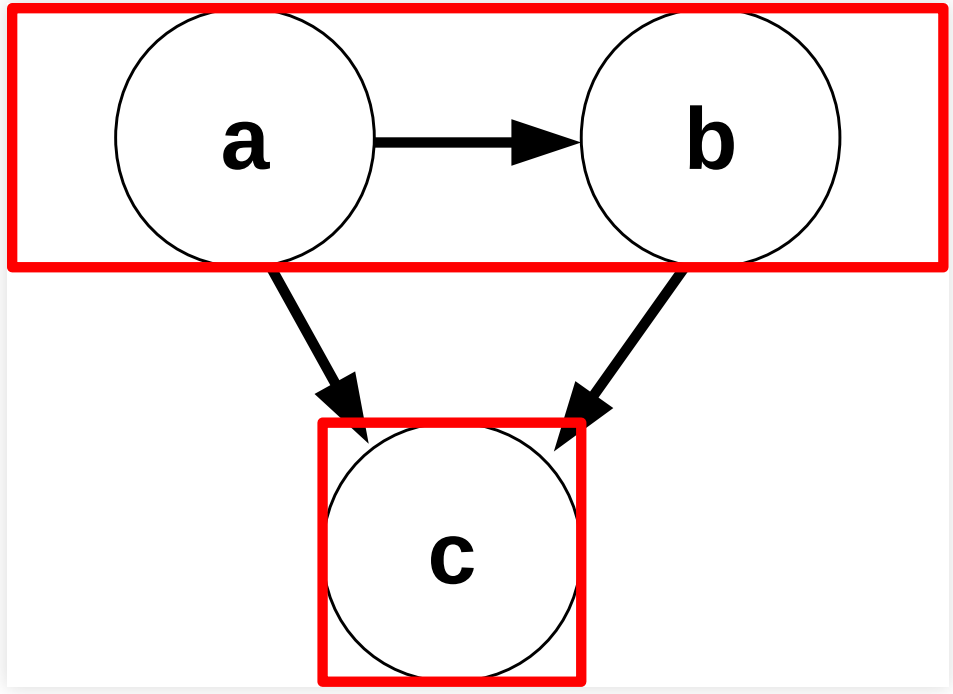
Bayesian networks

- Arc representing conditional dependency:



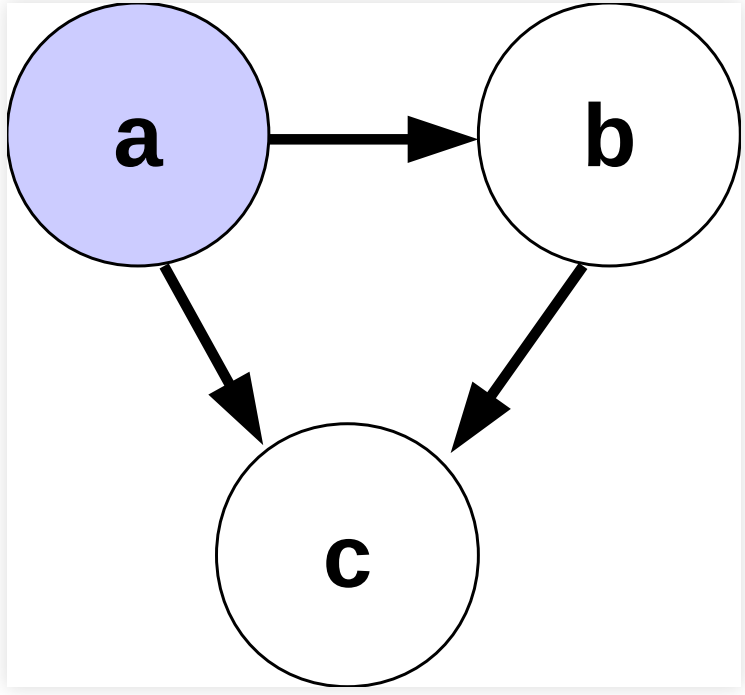
Bayesian networks

- Parents and offspring:



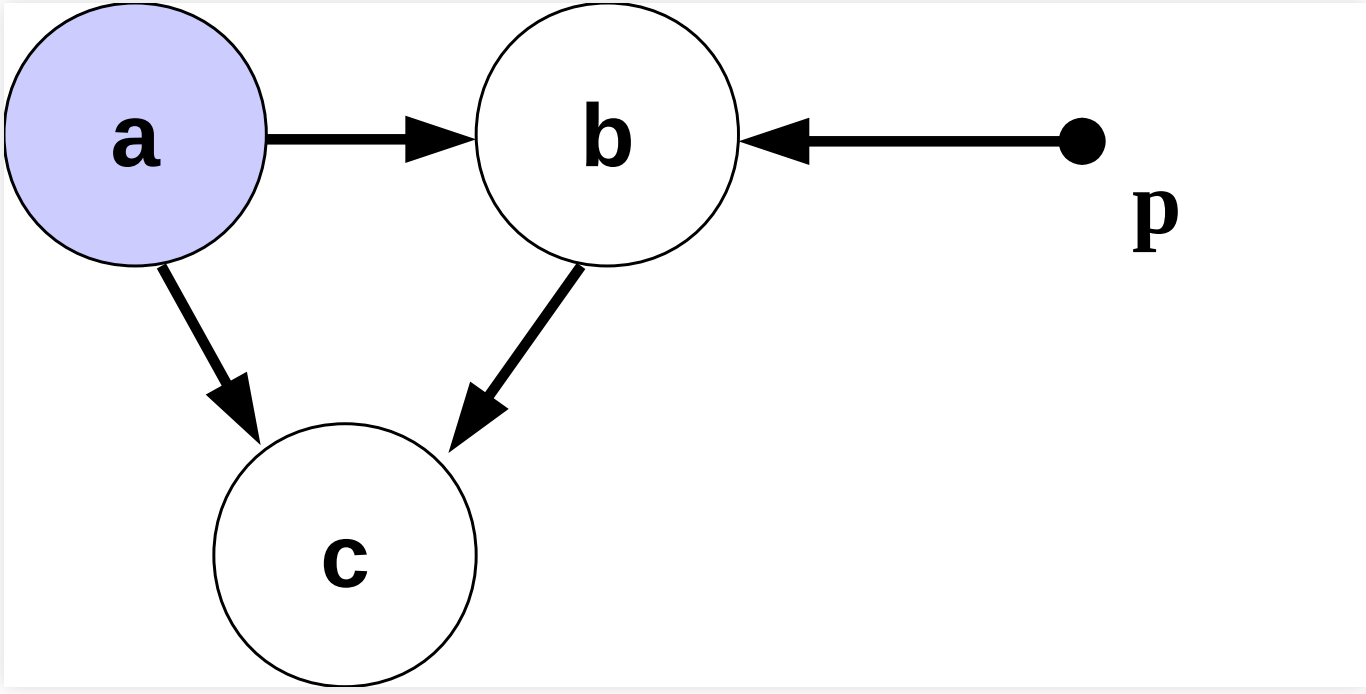
Bayesian networks

- We can indicate an observed variable by shading the corresponding node



Bayesian networks

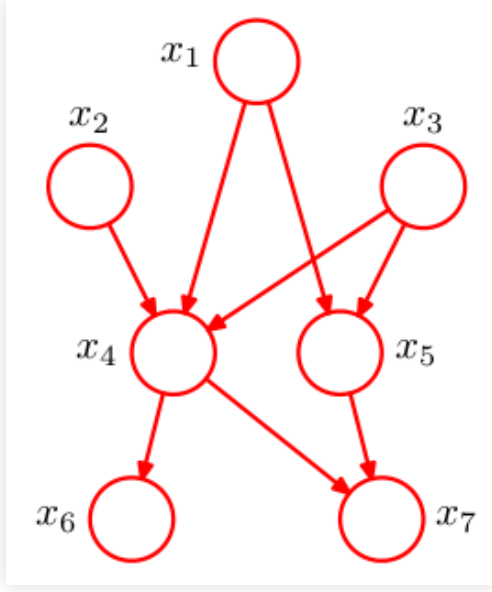
- Or parameters as special nodes



Bayesian networks

$$P(\mathbf{x}) = \prod_{n=1}^{n} P\left(x_n | \text{parents}(x_n)\right)$$

- Product of probabilities of all variables conditioned on the joint probabilities of their parents (e.g. Bishop, p.362)



$$p(x_1, \ldots, x_7) = p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5)$$

Markov Random Fields

Markov Random Fields

- Directed graphical models dependence separation more complex (head-to-tail, tail-to-tail, head-to-head)
- Undirected graph can represent conditional independency better

Markov Random Fields

- Undirected graph $G = (\mathbf{N}, \mathbf{A})$
- Random variables $\mathbf{X} = x_n \ , n \in \mathbf{N}$
- Non-adjacent independent given the rest

$$x_u \perp\!\!\!\perp x_v \, | X_{\mathbf{N} \setminus \{u,v\}} \ , \ u, v \notin \mathbf{A}$$

- Each independent of others given neighbours

$$x_v \perp\!\!\!\perp X_{\mathbf{N} \setminus cl(v)} \, | X_{ne(v)}$$

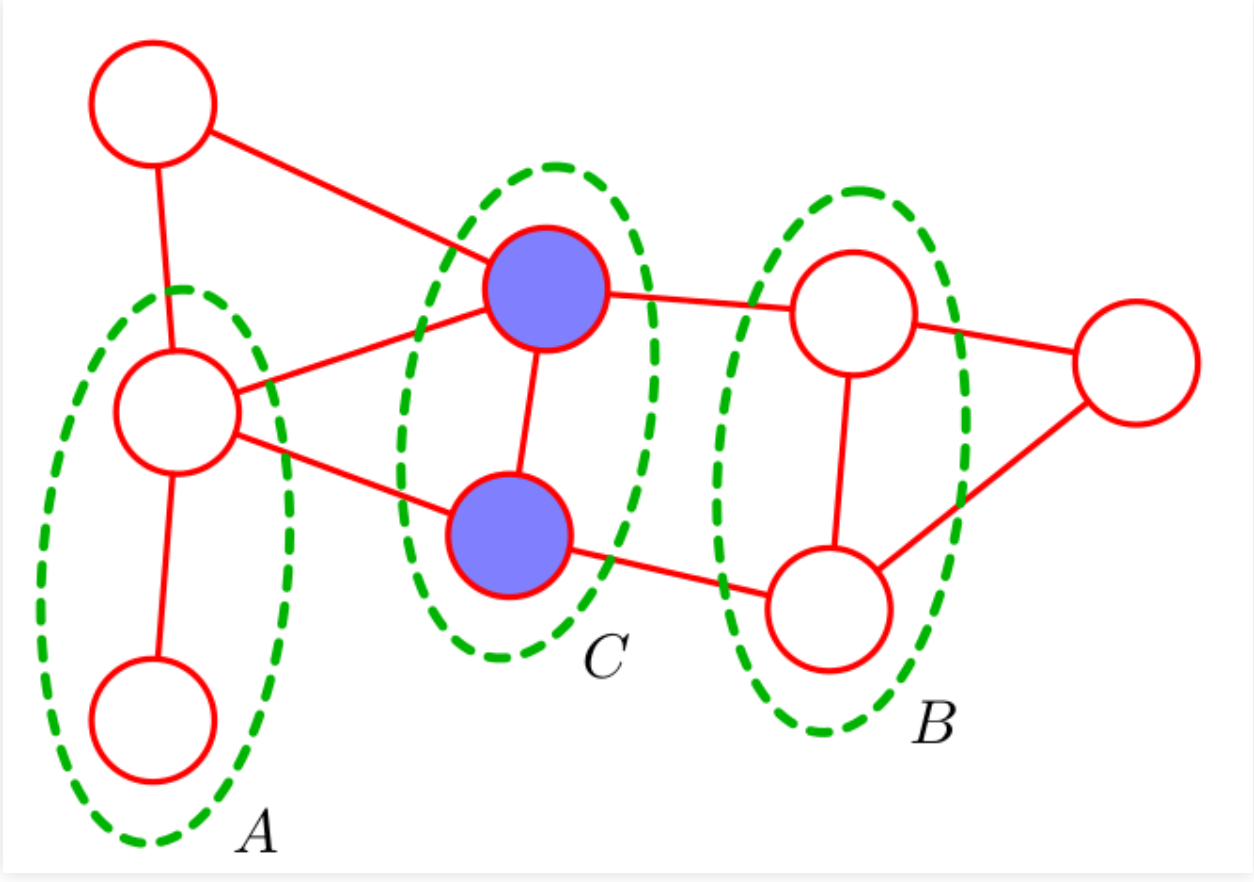
- if all paths between $\mathbf{U}$ and $\mathbf{V}$ go through $\mathbf{S}$

$$\mathbf{U} \perp\!\!\!\perp \mathbf{V} \, | \mathbf{S}$$

Markov Random Fields

- Example (Bishop, p.384)

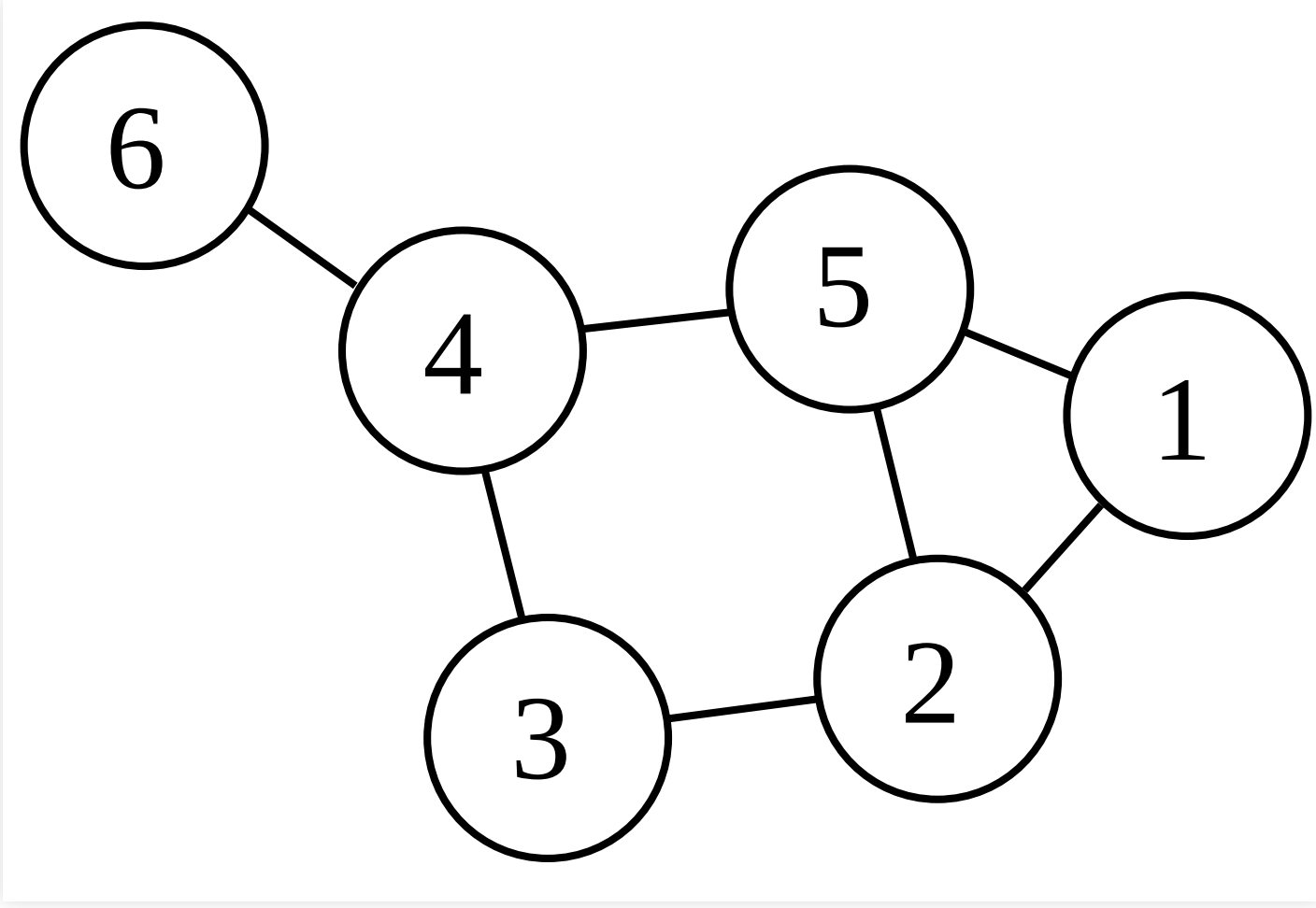
$$\mathbf{A} \perp\!\!\!\perp \mathbf{B} \,|\mathbf{C}$$



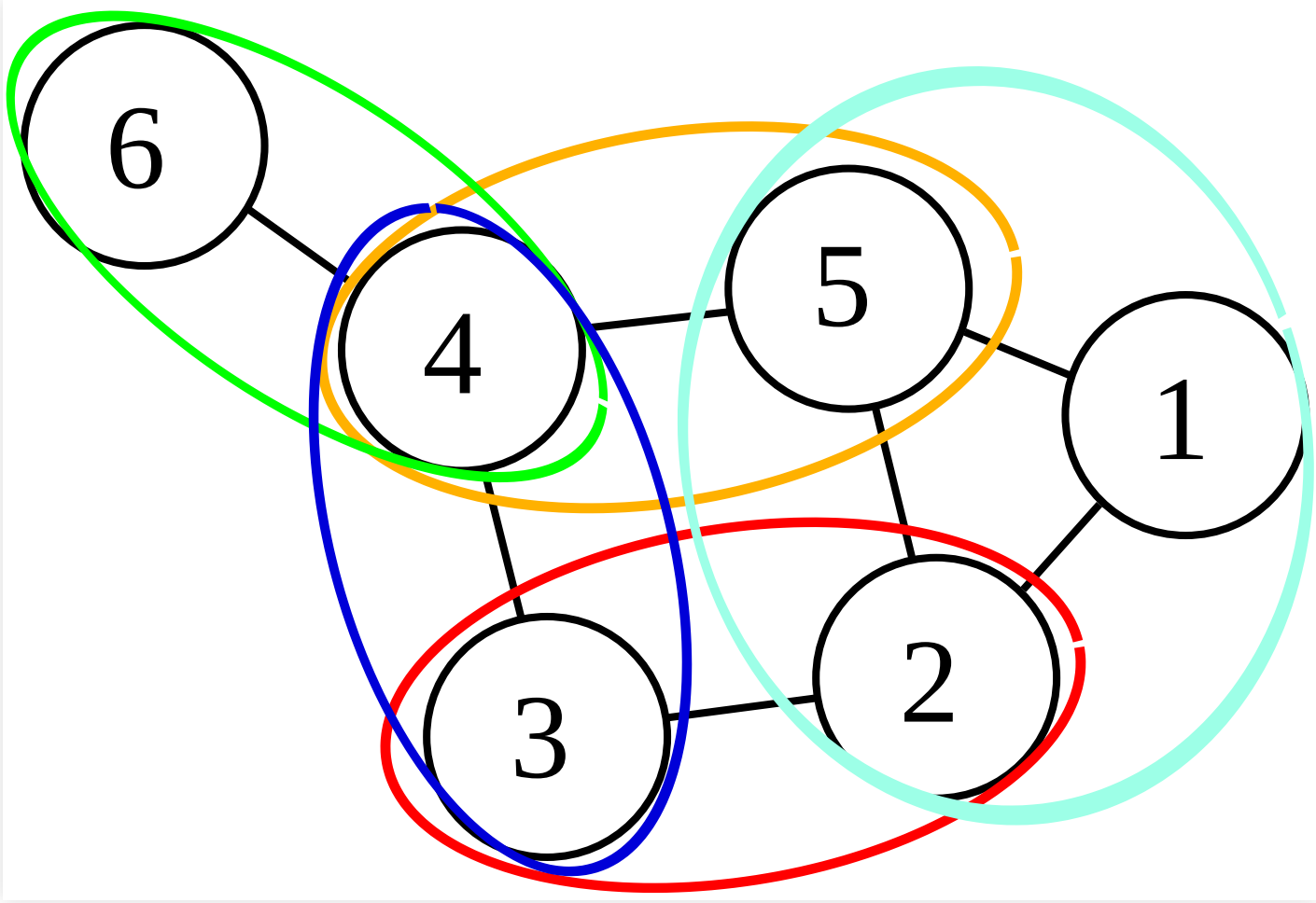
Markov Random Fields

- We can represent conditional independence in this way if the joint distribution is the product of functions of the cliques of the graph.
- A clique is a set of nodes that are all interconnected
- A maximal clique is a clique to which no nodes can be added and still remain a clique. We can write the joint distribution as a product of functions of maximal cliques.

Markov Random Fields



Markov Random Fields



Markov Random Fields

- The joint probability is:

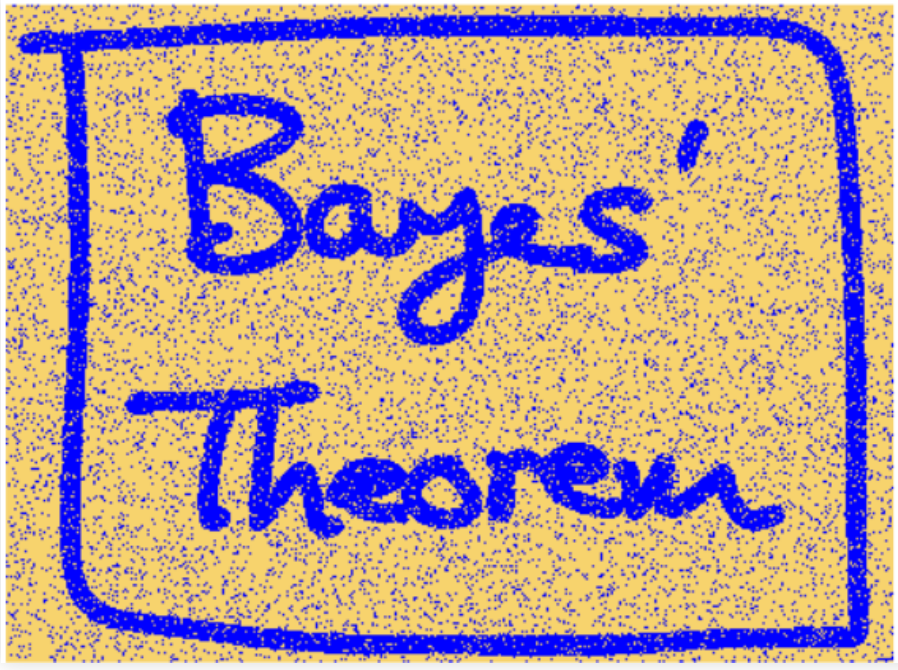
$$P(\mathbf{X}) = \frac{1}{Z} \prod_{C} \psi_C(\mathbf{x}_C)$$

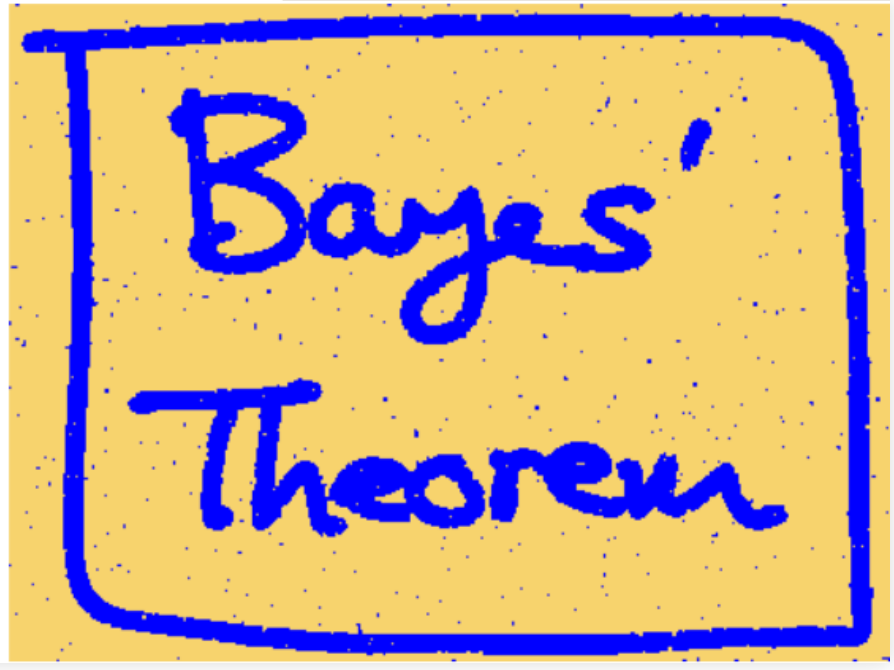
- Where ψ_C is a potential function for each (maximal) clique and Z is a normalization constant

$$Z = \sum_{x} \prod_{C} \psi_C(\mathbf{x}_C)$$

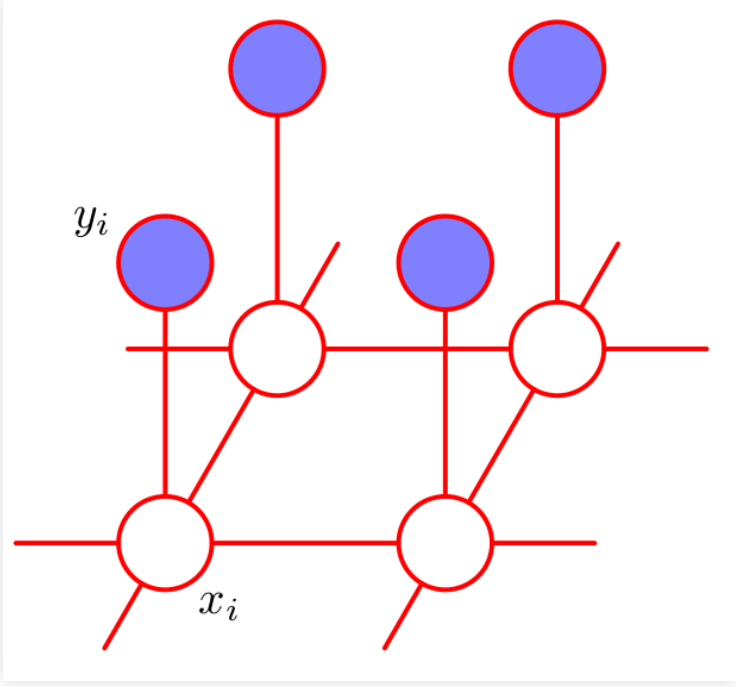
Markov Random Fields

- Example, denoising image (Bishop, 388)



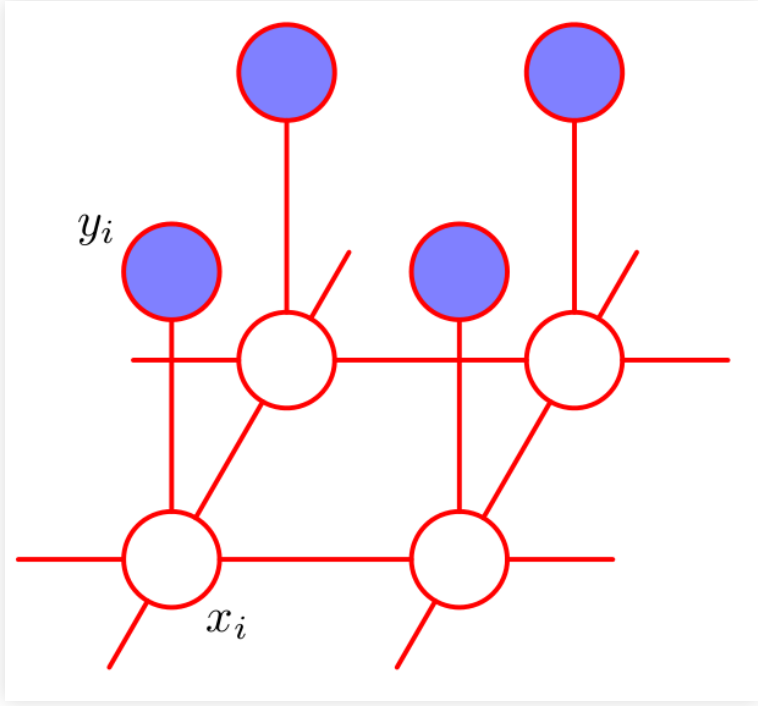


Markov Random Fields



$$x_i, y_i \in \{-1, 1\}$$

Markov Random Fields



- "Energy" function

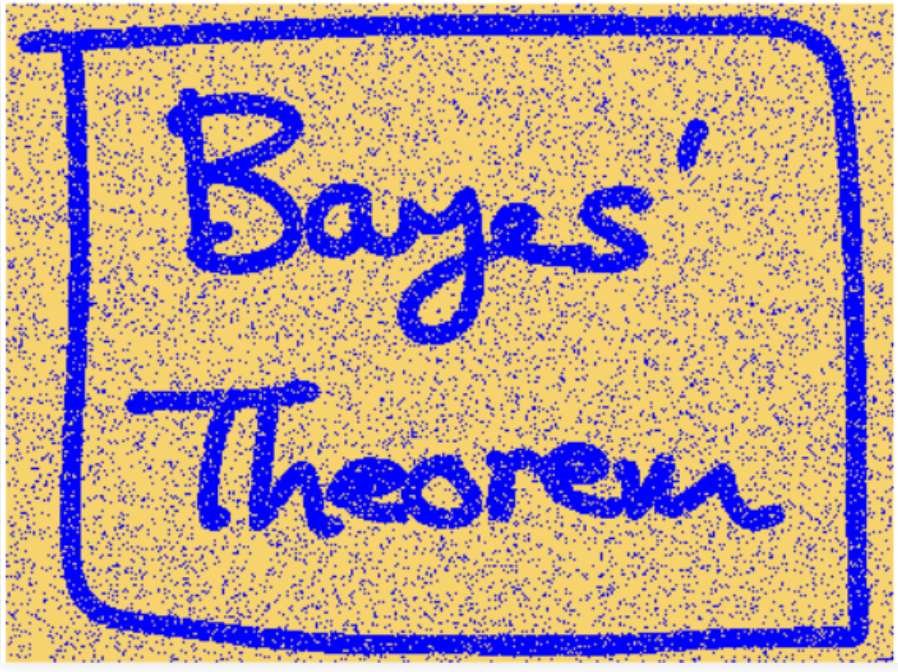
$$E(\mathbf{x}, \mathbf{y}) =$$

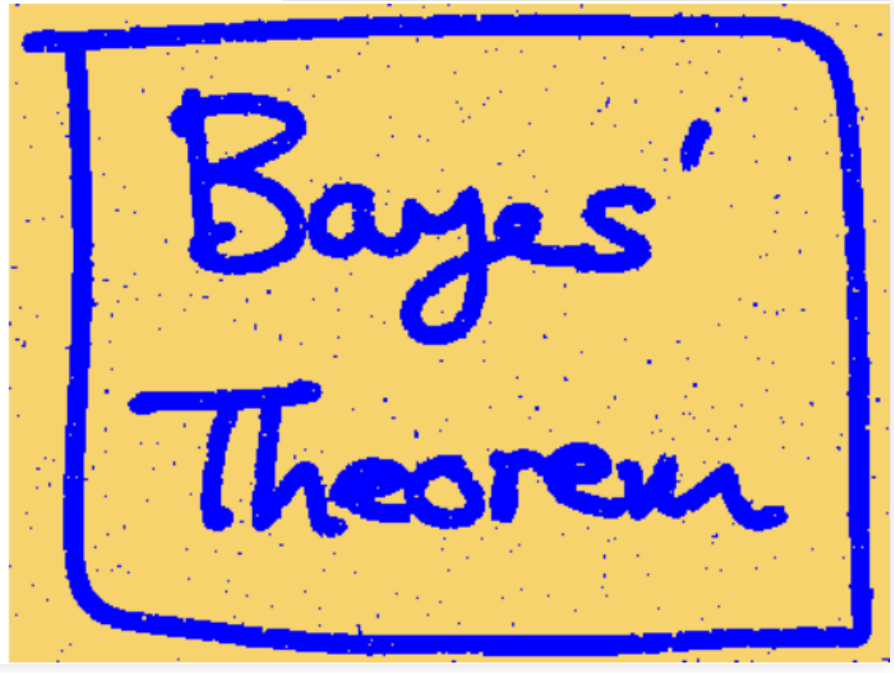
- x_i, x_j pairs: $-\beta \sum x_i x_j$
- x_i, y_i pairs: $-\eta \sum x_i y_i$
- x bias: $+h \sum x_i$

$$P(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} e^{-E(\mathbf{x},\mathbf{y})}$$

Markov Random Fields

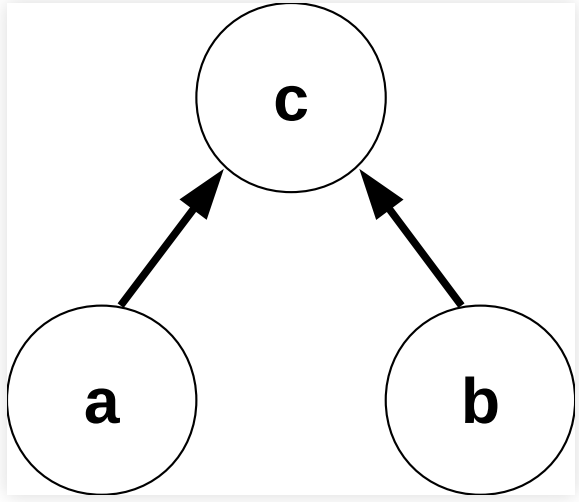
- Find $\mathbf{x}$ that maximizes $\frac{1}{Z} e^{-E(\mathbf{x},\mathbf{y})}$





Bayesian Networks vs Markov Random Fields

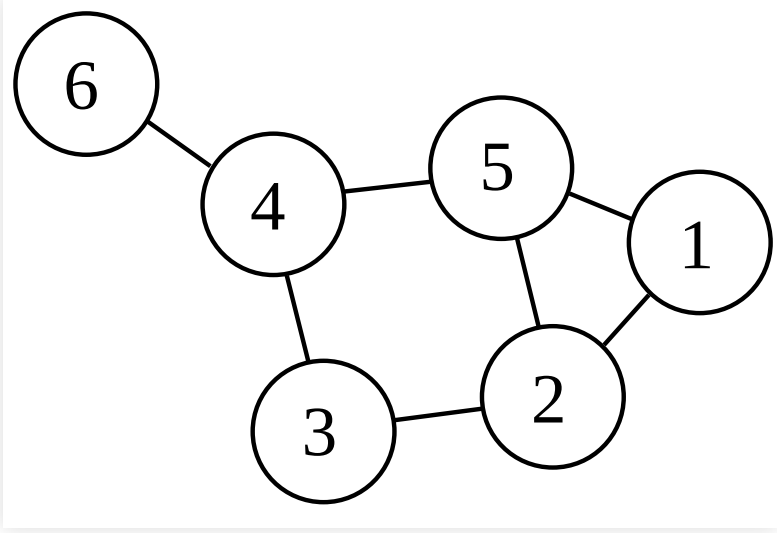
- These graph models represent dependencies differently
- Bayesian Networks are directed acyclical graphs of conditional dependencies



- Bayesian Networks can represent some dependencies that MRF cannot represent

Bayesian Networks vs Markov Random Fields

- These graph models represent dependencies differently
- Random Markov Fields are undirected graphs



- Random Markov Fields can include cyclical dependencies that Bayesian Networks cannot represent

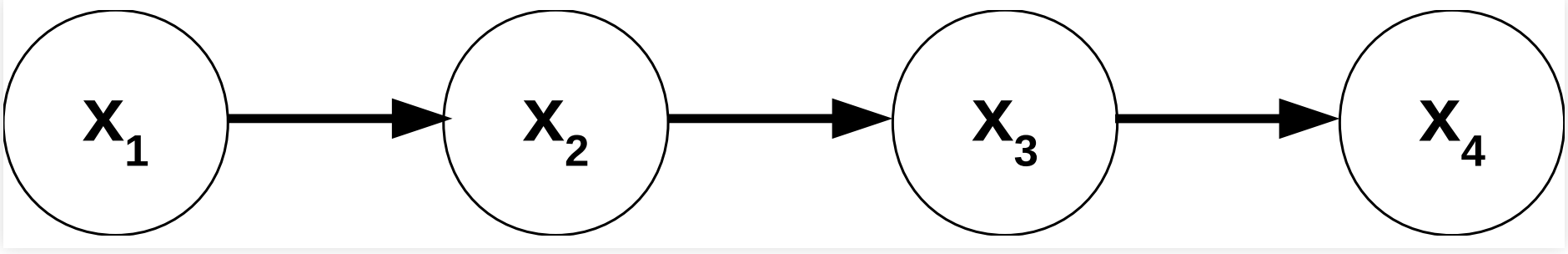
Dynamic Bayesian Networks

- Dynamic Bayesian Networks relate variables over time steps.

DBN and Markov Process

- One thing we can represent with a DBN is a Markov process.
- In a Markov process , the probability distribution of x_t depends only on $x_t - 1$.

$$p(x_1, \ldots, x_T) = p(x_1) \prod_{t=2}^{T} p(x_t | x_{t-1})$$

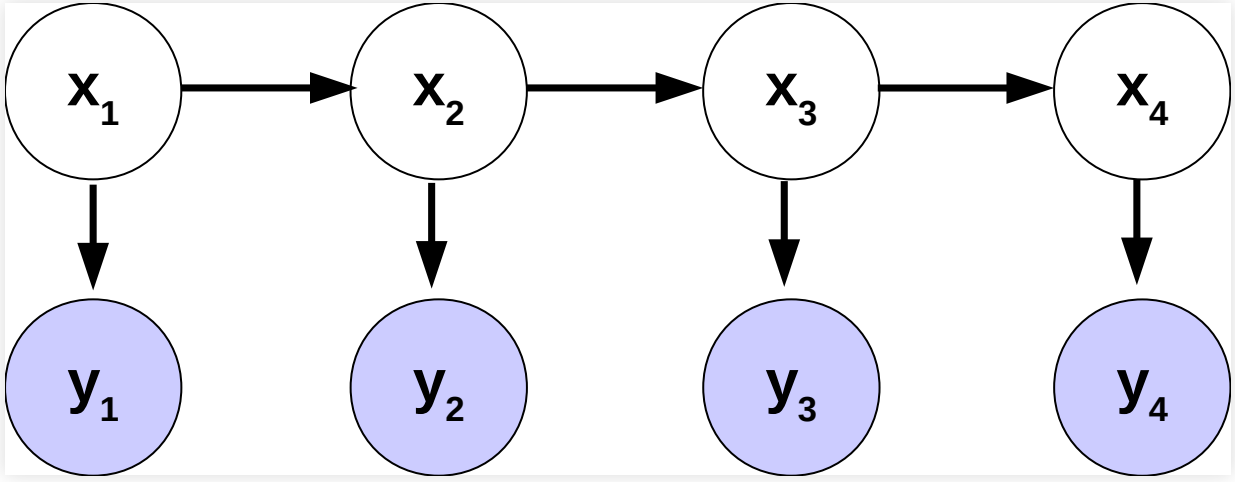


Hidden Markov Models

Hidden Markov Models

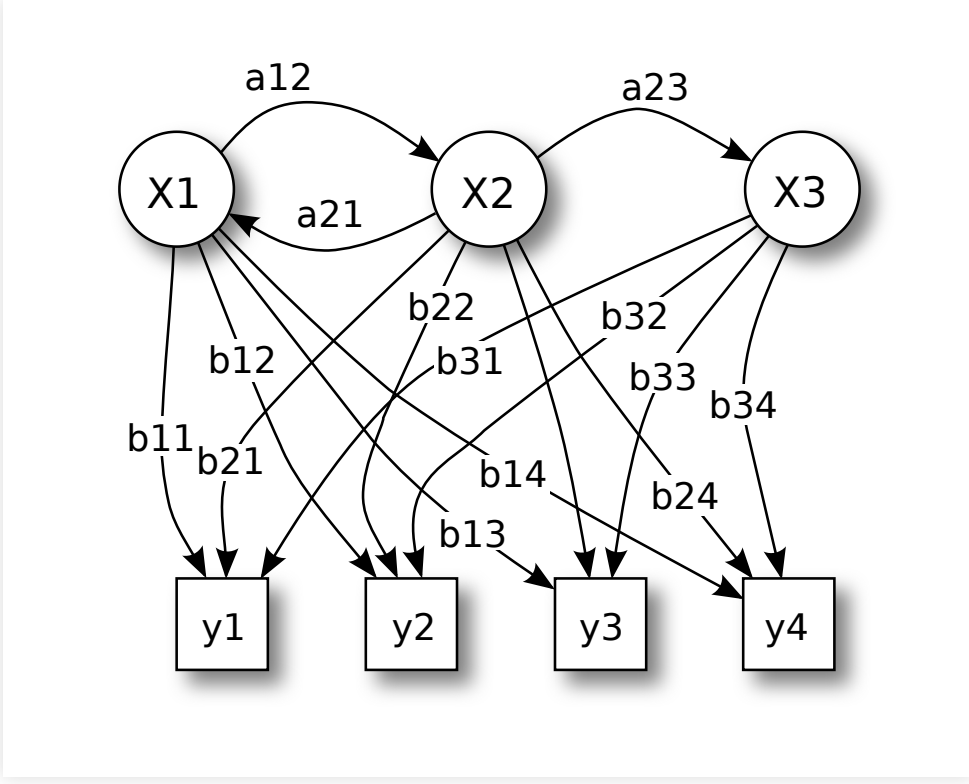
- A Hidden Markov Model is a Markov chain of hidden variables each with an observation conditioned on its state.

$$p(x_1, \ldots, x_T, y_1, \ldots, y_T) = p(x_1) \prod_{t=2}^{T} p(x_t|x_{t-1}) \prod_{t=1}^{T} p(y_t|x_t)$$



HMM

- We can also represent a HMM with a graph showing state transitions and emissions.



Tdunning CC BY 3.0, Wikimedia

Hidden Markov Model parameters

- To specify a HMM with N states, assuming $P(x_t|x_{t-1})$, we need the $N \times N$ matrix:

$$A_{i,j} = P(x_t = j|x_{t-1} = i)$$

- $N \times (N-1)$ independent parameters, because they are probabilities

- If y_t can take one of K values, the $K \times N$ matrix:

$$B_{k,j} = P(y_t = k|x_t = j)$$

- $N \times (K-1)$ independent parameters

- Finally, the N probabilities for x_1:

$$\pi_i = P(x_1 = i)$$

- $N - 1$ independent parameters

Uses for Hidden Markov Models

- Temporal pattern recognition

- Voice recognition, sign language, gait recognition

- Pattern recognition in sequences

- Text tagging, sequence comparison in bioinformatics, genome annotation (finding genes)

HMM

- Predict next value (filtering)
- With A, B, π, the belief state at t after $y_{1:t}$:

$$\alpha_t(x_t) = p(x_t, y_{1:t})$$

- (The joint probability of the state at time t and all previous observations)

- Too hard to compute all possible combinations
- Use the forward algorithm:

- Since y_t only depends on x_t and x_t on x_{t-1}:
- $\alpha_t(x_t) = p(y_t|x_t) \sum_{x_{t-1}}^{N} p(x_t|x_{t-1})\alpha_{t-1}(x_{t-1})$
- $\alpha_1(x_1) = p(y_1|x_1 = i)\pi_i$

HMM

- Knowing A, B, π we can also compute the most likely sequence of $x_1, \ldots, x_t$ given $y_1, \ldots, y_t$ (Viterbi algorithm)
- Two matrices of $N \times T$:

- Probability of the most likely path for each state x_j
- Previous state of the most likely path for each state x_j

- These matrices can be built in order, from previous state
- Then we trace back the states of x_j used at each step to get the most likely path
- Applications of the Viterbi algorithm

- Speech synthesis and recognition
- Speaker identification (diarization)
- Wi-Fi 802.11 (Viterbi decoding)
- Bioinformatics (sequence analysis)

Note: forward algorithm and Viterbi algorithm

- Not the same thing!
- Forward algorithm:
  - Likelihood of sequence of observations
- Viterbi algorithm:
  - Most likely sequence of states given observations

HMM

- To train the model and find (A, B, π) we use the Baum-Welch algorithm, which is another example of Expectation-Maximization
- Start with a guess for (A, B, π)
- The forward pass: $\alpha_i(t) = p(x_t = i, y_{1:t}|A, B, \pi)$

$$\alpha_i(1) = \pi_i B_{i,y_1} \qquad \alpha_i(t) = B_{i,y_t} \sum_{n=1}^{N} \alpha_n(t-1) A_{n,i}$$

- The backward pass: $\beta_i(t) = p(y_{t+1:T}|x_t = i, A, B, \pi)$

$$\beta_i(T) = 1 \qquad \beta_i(t) = \sum_{n=1}^{N} \beta_n(t+1) A_{i,n} B_{n,y_{t+1}}$$

Expectation step

- From this we can compute how to maximize the estimated likelihood function, based on previous (A, B, π):

$$\gamma_i(t) = p(x_t = i|\mathbf{y}, A, B, \pi) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{n=1}^{N} \alpha_n(t)\beta_n(t)}$$

$$\xi_{ij}(t) = p(x_t = i, x_{t+1} = j|\mathbf{y}, A, B, \pi) = \frac{\alpha_i(t)A_{i,j}\beta_j(t+1)B_{j,y_{t+1}}}{\sum_{n=1}^{N}\sum_{m=1}^{N} \alpha_m(t)A_{m,n}\beta_m(t+1)B_{n,y_{t+1}}}$$

Maximization step

- Now we update π, fraction of each state at beginning:

$$\pi_i = \gamma_i(1)$$

- A, fraction of each transition w.r.t fraction of each state:

$$A_{i,j} = \frac{\sum_T \xi_{i,j}(t)}{\sum_T \gamma_i(t)}$$

- B, Fraction of each emmission w.r.t fraction of each state:

$$B_{i,k} = \frac{\sum_T 1_{y_t=k} \gamma_i(t)}{\sum_T \gamma_i(t)}$$

Baum-Welch algorithm (Expectation-Maximization)

- Compute joint probability of hidden state and emissions so far at each time.
- Compute probability of future emissions given hidden state at each time.
- Compute probability hidden state at each time given all emissions.
- Compute joint probability of consecutive hidden states at each time given all emissions.
- Use this to find new parameter values
- (Does not guarantee optimum and may overfit)

HMM

- HMM are generative models (compute joint probabilities) so can be used to generate and classify
- To classify, we can compute the probability of the most likely state sequences for each HMM. Example: Pfam, speech recognition, time-series analysis, ...
- To generate, we can use the joint probabilities. E.g speech synthesis, generating values for frequency and duration.

This week and the next

- Today:

- Lecture on deep learning (preview, not for tests)
- Tutorial: P1

- Thursday, 17

- Officially, there are no more tutorials
- But I will be available at the times of P5 and P6 (11:00-16:00) for questions.

- Next week, 22 of December

- No streamed lecture but I'll be on the Zoom meeting for questions

First assignment grades

- Nearly done. I should publish them tomorrow.

Summary

Summary

- Probabilistic Clustering
- Graphical models: BN and RMF
- Markov process
- Hidden Markov Model

- Viterbi algorithm: most likely sequence
- Baum-Welch algorithm: EM training of HMM

Further reading

- Bishop, Sections 8-8.1.3; 8.2.1; 8.3-8.3.3; 13-13.2.2