Final Review

Topics we covered

Machine Learning

Graphical Models

- Basics
  - Encode independence
  - Bayes ball, markov blanket
- Inference
  - Exact
  - Expectation Maximization
  - Gibbs sampling
- Dynamical Systems
  - HMMs, SSMs

Non-parametrics

- Kernels
- Gaussian process

Neural Networks

- Perceptrons
- Back prop

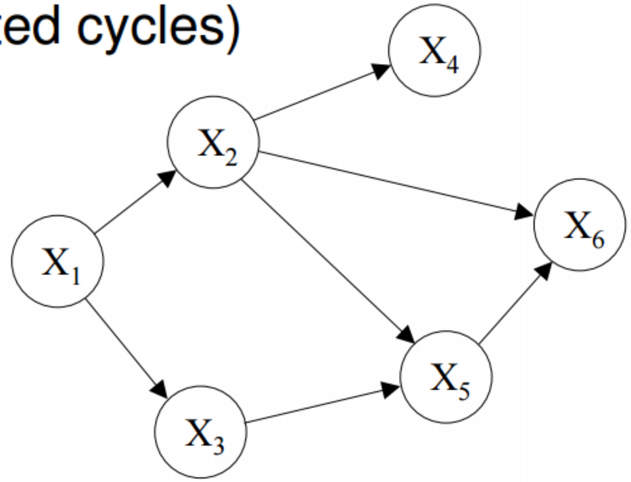
Optimization!

Visualization always helps!

Algebra is boring, so let's draw this

- Let's represent variables as circles
- Let's draw an arrow from j to i if $j \in pa_i$
- The resulting drawing will be a Directed Graph
- Moreover it will be Acyclic (no directed cycles)

Llatent variable / latent parameter	var
Observed variable	obs
Constant / hyper parameter	const



Bayesian Network

- A **Bayesian network** is specified by a directed *acyclic* graph $G = (V, E)$ with:
  1. One node $i \in V$ for each random variable X_i
  2. One conditional probability distribution (CPD) per node, $p(x_i \mid \mathbf{x}_{\mathrm{Pa}(i)})$, specifying the variable's probability conditioned on its parents' values
- Corresponds 1-1 with a particular factorization of the joint distribution:

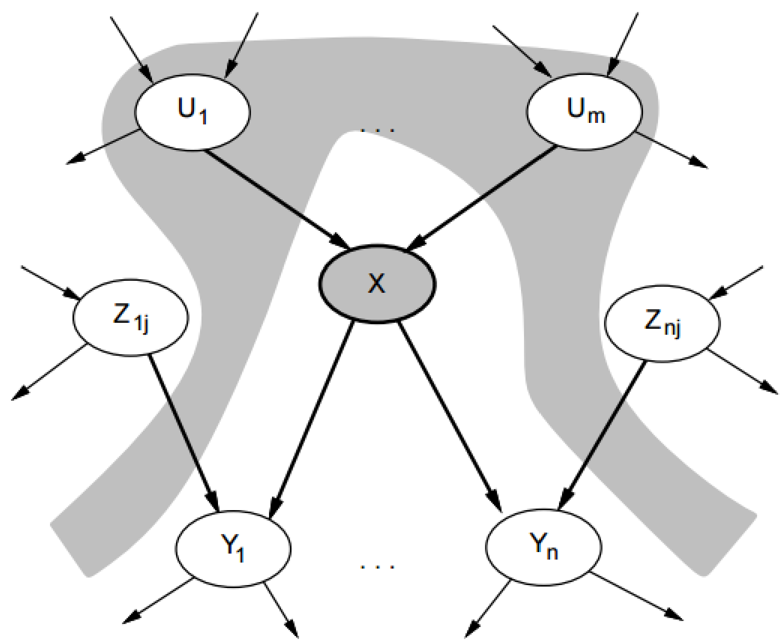
$$p(x_1, \ldots x_n) = \prod_{i \in V} p(x_i \mid \mathbf{x}_{\mathrm{Pa}(i)})$$

- Powerful framework for designing *algorithms* to perform probability computations

Copied from: http://cs.nyu.edu/~dsontag/courses/pgm13/slides/lecture1.pdf

More properties

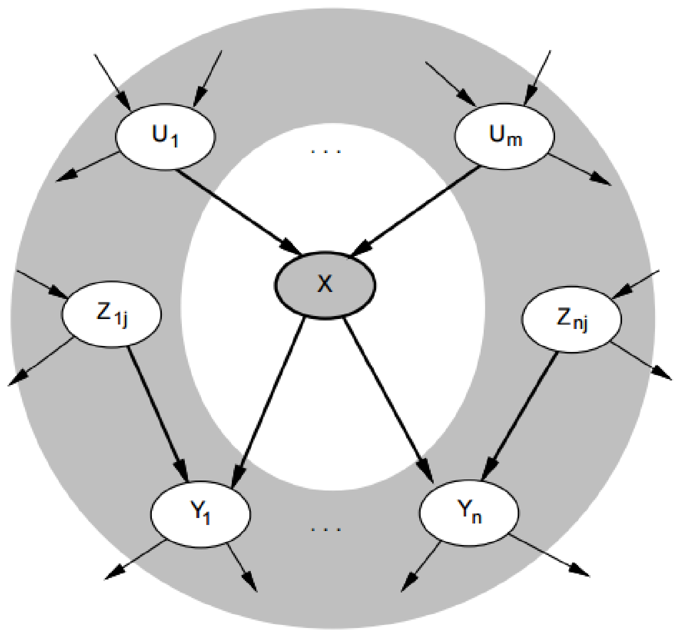
Local semantics: each node is conditionally independent of its nondescendants given its parents



Copied from: http://courses.cs.washington.edu/courses/cse515/15wi/slides/bnets.pdf

More Properties

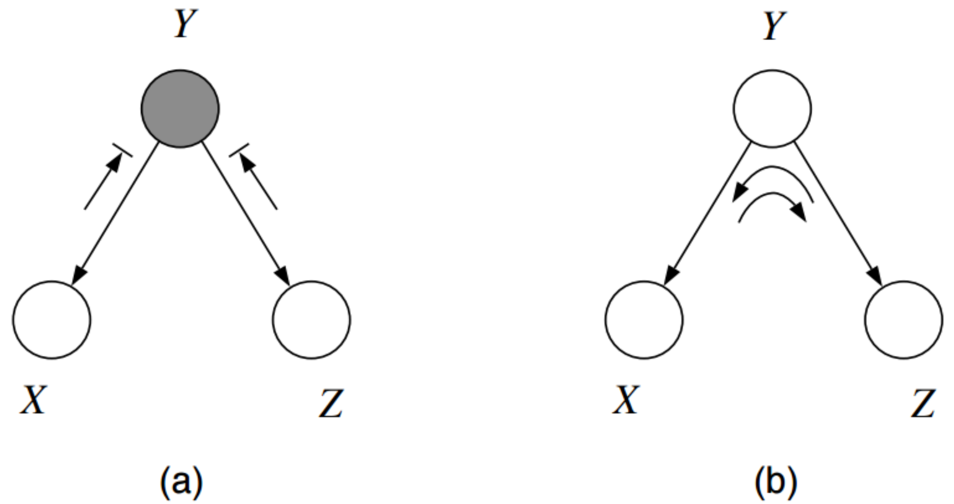
Each node is conditionally independent of all others given its Markov blanket: parents + children + children's parents



Copied from: http://courses.cs.washington.edu/courses/cse515/15wi/slides/bnets.pdf

Bayes Ball

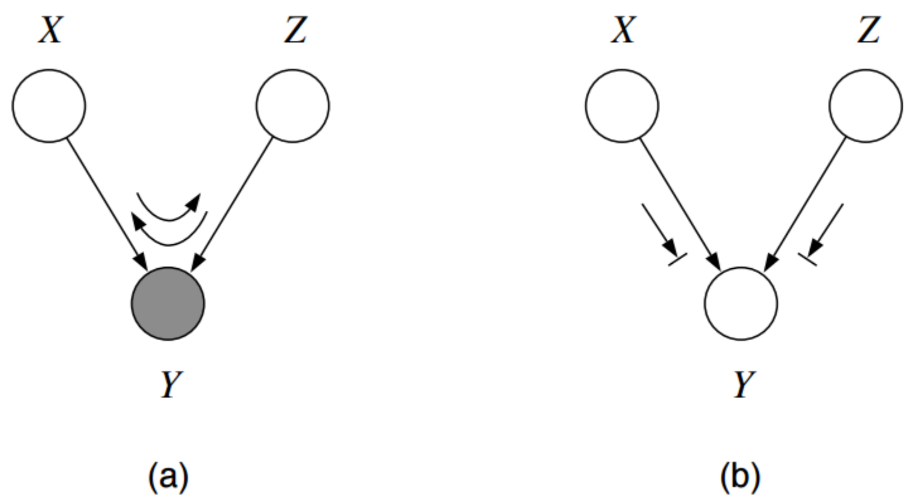
- Algorithm to calculate whether $X \perp Z \mid \mathbf{Y}$ by looking at graph separation
- Look to see if there is **active path** between X and Z when variables $\mathbf{Y}$ are observed:



Copied from: http://cs.nyu.edu/~dsontag/courses/pgm13/slides/lecture1.pdf

Bayes Ball

- Algorithm to calculate whether $X \perp Z \mid \mathbf{Y}$ by looking at graph separation
- Look to see if there is **active path** between X and Z when variables $\mathbf{Y}$ are observed:



Copied from: http://cs.nyu.edu/~dsontag/courses/pgm13/slides/lecture1.pdf

Bayes Ball

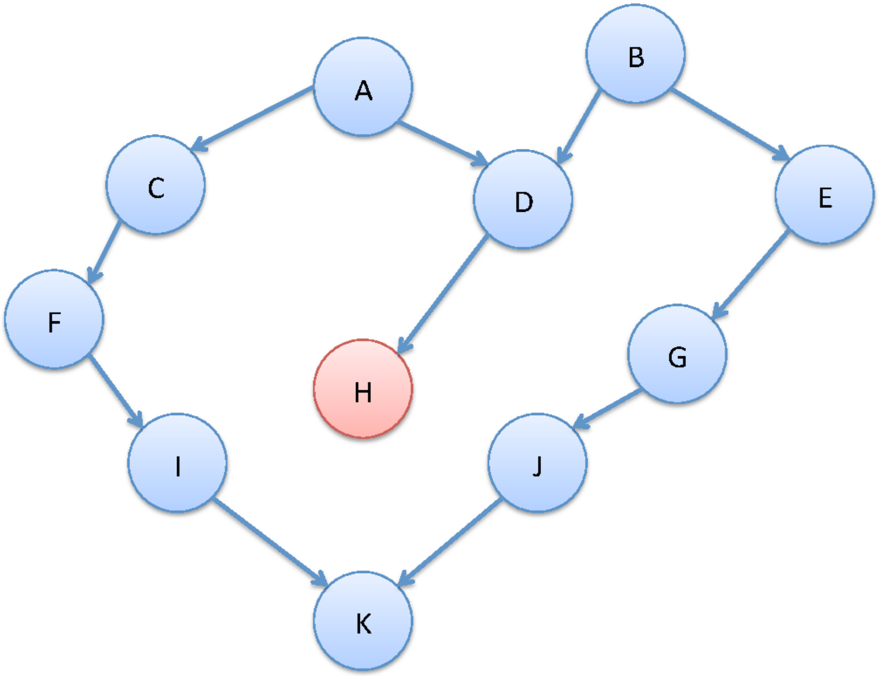
- Algorithm to calculate whether $X \perp Z \mid \mathbf{Y}$ by looking at graph separation
- Look to see if there is **active path** between X and Z when variables $\mathbf{Y}$ are observed:

(a)

(b)

Copied from: http://cs.nyu.edu/~dsontag/courses/pgm13/slides/lecture1.pdf

A More Complex Example



Easy:

- ¬(A ⊥ B | D)

Harder:

- ¬(F ⊥ G | D)
- ¬(F ⊥ G | H)

Flow of influence, again?

Copied from: https://www.ark.cs.cmu.edu/PGM/index.php/Current_events_(2010)

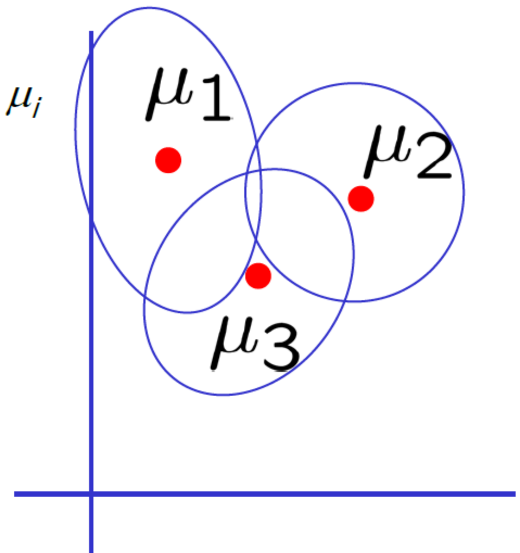
Such exact inference is hopeless in general.

We have to approximate.

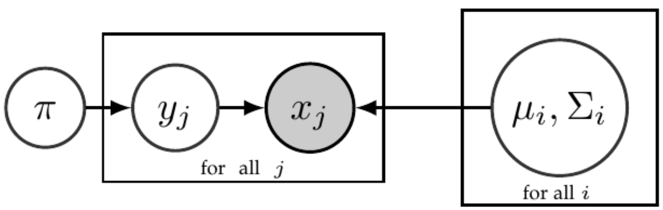
Gaussian Mixture Model

Mixture of K Gaussians distributions: (Multi-modal distribution)

- There are K components
- Component *i* has an associated mean vector μ_i



Component *i* generates data from $N(\mu_i, \Sigma_i)$



Each data point is generated using this process:

1) Choose component i with probability $\pi_i = P(y = i)$
2) Datapoint $x \sim N(\mu_i, \Sigma_i)$

Gaussian Mixture Model

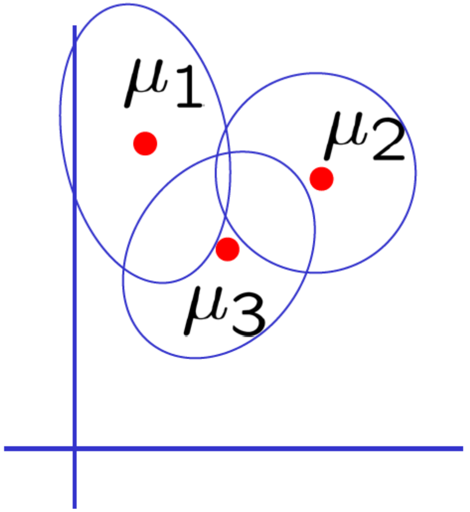
Mixture of K Gaussians distributions: (Multi-modal distribution)

Hidden variable

$$p(x|y = i) = N(\mu_i, \Sigma_i)$$

$$p(x) = \sum_{i=1}^{K} p(x|y = i)P(y = i)$$

Observed data — **Mixture component** — **Mixture proportion**

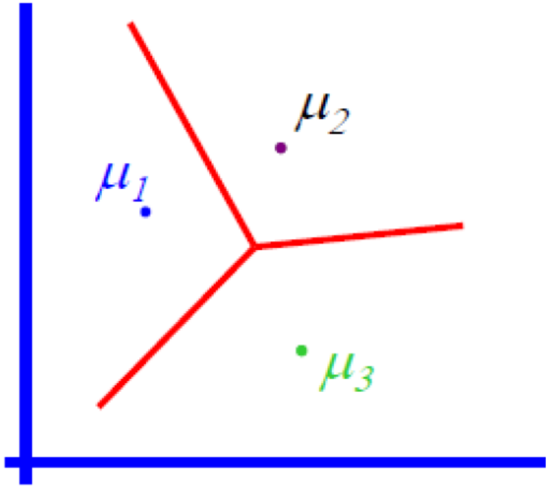


Inference on GMM

What if we don't know $\mu_1, \ldots, \mu_K, \sigma^2, \pi_1, \ldots, \pi_K$?

$\Rightarrow$ **Maximum Likelihood Estimate (MLE)**

$$\theta = [\mu_1, \ldots, \mu_K, \sigma^2, \pi_1, \ldots, \pi_K]$$



$$\arg\max_{\theta} \prod_{j=1}^{n} P(x_j|\theta)$$

$$= \arg\max_{\theta} \prod_{j=1}^{n} \sum_{i=1}^{K} P(y_j = i, x_j|\theta)$$

$$= \arg\max_{\theta} \prod_{j=1}^{n} \sum_{i=1}^{K} P(y_j = i|\theta)p(x_j|y_j = i|\theta)$$

$$= \arg\max_{\theta} \prod_{j=1}^{n} \sum_{i=1}^{K} \pi_i \frac{1}{\sqrt{2\pi\sigma^2}} \exp(\frac{-1}{2\sigma^2}||x_j - \mu_i||^2)$$

Inference on GMM

What if we don't know $\theta = [\mu_1, \ldots, \mu_K, \Sigma_1, \ldots, \Sigma_K, \pi_1, \ldots, \pi_K]$?

$\Rightarrow$ **Maximize marginal likelihood (MLE):**

$$\arg\max_\theta \prod_{j=1}^{n} P(x_j|\theta) = \arg\max_\theta \prod_{j=1}^{n} \sum_{i=1}^{K} P(y_j = i, x_j|\theta)$$

$$= \arg\max_\theta \prod_{j=1}^{n} \sum_{i=1}^{K} \pi_i \frac{1}{\sqrt{2\pi|\Sigma_i|}} \exp\left[-\frac{1}{2}(x_j - \mu_i)^T \Sigma_i^{-1} (x_j - \mu_i)\right]$$

How do we find $\theta = [\mu_1, \ldots, \mu_K, \Sigma_1, \ldots, \Sigma_K, \pi_1, \ldots, \pi_K]$ which gives max. marginal likelihood?

* Set $\frac{\partial}{\partial \mu_i} \log \text{Prob}(\ldots) = 0$, and solve for μ_i. Non-linear, non-analytically solvable
* Use gradient descent. Doable, but often slow
* Use EM.

Expectation-Maximization (EM)

A general algorithm to deal with hidden data, but we will study it in the context of unsupervised learning (hidden class labels = clustering) first.

- EM is an optimization strategy for objective functions that can be interpreted as likelihoods in the presence of missing data.
- EM is much simpler than gradient methods:
  No need to choose step size.
- EM is an iterative algorithm with two linked steps:
  - E-step: fill-in hidden values using inference
  - M-step: apply standard MLE/MAP method to completed data
- We will prove that this procedure monotonically improves the likelihood (or leaves it unchanged). EM always converges to a local optimum of the likelihood.

A simple case:

- We have unlabeled data $x_1, x_2, \ldots, x_m$
- We know there are K classes
- We know $P(y=1)=\pi_1$, $P(y=2)=\pi_2$ $P(y=3)$ … $P(y=K)=\pi_K$
- We know common variance σ^2
- We **don't** know $\mu_1, \mu_2, \ldots \mu_K$, and we want to learn them

We can write

$$p(x_1, \ldots, x_n | \mu_1, \ldots \mu_K) = \prod_{j=1}^{n} p(x_j | \mu_1, \ldots, \mu_K) \quad \text{Independent data}$$

$$= \prod_{ij=1}^{n} \sum_{i=1}^{K} p(x_j, y_j = i | \mu_1, \ldots, \mu_K) \quad \text{Marginalize over class}$$

$$= \prod_{ij=1}^{n} \sum_{i=1}^{K} p(x_j | y_j = i | \mu_1, \ldots, \mu_K) p(y_j = i)$$

$$\propto \prod_{ij=1}^{n} \sum_{i=1}^{K} \exp(-\frac{1}{2\sigma^2} \|x_j - \mu_i\|^2) \pi_i \quad \Rightarrow \text{learn } \mu_1, \mu_2, \ldots \mu_K$$

E-step

We want to learn: $\theta = [\mu_1, \ldots, \mu_K]$

Our estimator at the end of iteration t-1: $\theta^{t-1} = [\mu_1^{t-1}, \ldots, \mu_K^{t-1}]$

At iteration t, construct function Q:

$$Q(\theta^t|\theta^{t-1}) = \sum_{j=1}^{n} \sum_{i=1}^{K} P(y_j = i|x_j, \theta^{t-1}) \log P(x_j, y_j = i|\theta^t)$$

E step

$$\begin{aligned} P(y_j = i|x_j, \theta^{t-1}) &= P(y_j = i|x_j, \mu_1^{t-1}, \ldots, \mu_K^{t-1}) \\ &\propto P(x_j|y_j = i, \mu_1^{t-1}, \ldots, \mu_K^{t-1}) P(y_j = i) \\ &\propto \exp(-\frac{1}{2\sigma^2}\|x_j - \mu_i^{t-1}\|^2)\pi_i \\ &= \frac{\exp(-\frac{1}{2\sigma^2}\|x_j - \mu_i^{t-1}\|^2)\pi_i}{\sum_{i=1}^{K} \exp(-\frac{1}{2\sigma^2}\|x_j - \mu_i^{t-1}\|^2)\pi_i} \end{aligned}$$

Equivalent to assigning clusters to each data point in K-means in a soft way

M-step

$$Q(\theta^t|\theta^{t-1}) = \sum_{j=1}^{n}\sum_{i=1}^{K} P(y_j = i|x_j, \theta^{t-1}) \log P(x_j, y_j = i|\theta^t)$$

$$= \sum_{j=1}^{n}\sum_{i=1}^{K} P(y_j = i|x_j, \theta^{t-1})[\log \underbrace{P(x_j|y_j = i, \theta^t)}_{\propto \exp(-\frac{1}{2\sigma^2}\|x_j - \mu_i^t\|^2)} + \log \underbrace{P(y_j = i|\theta^t)}_{\pi_i}]$$

We calculated these weights in the E step

$$R_{i,j}^{t-1} = P(y_j = i|x_j, \theta^{t-1})$$

Joint distribution is simple

M step At iteration t, maximize function Q in θ^t:

$$Q(\mu_i^t|\theta^{t-1}) \propto \sum_{j=1}^{n} R_{i,j}^{t-1}(-\frac{1}{2\sigma^2}\|x_j - \mu_i^t\|^2)$$

$$\frac{\partial}{\partial \mu_i^t} Q(\mu_i^t|\theta^{t-1}) = 0 \Rightarrow \sum_{j=1}^{n} R_{i,j}^{t-1}(x_n - \mu_i^t) = 0$$

$$\mu_i^t = \sum_{j=1}^{n} w_j x_j \text{ where } w_j = \frac{R_{i,j}^{t-1}}{\sum_{j=1}^{n} R_{i,j}^{t-1}} = \frac{P(y_j=i|x_j,\theta^{t-1})}{\sum_{l=1}^{n} P(y_l=i|x_l,\theta^{t-1})}$$

Equivalent to updating cluster centers in K-means

Summary

E-step

Compute "expected" classes of all datapoints for each class

$$P(y_j = i|x_j, \theta^{t-1}) = \frac{\exp(-\frac{1}{2\sigma^2}\|x_j - \mu_i^{t-1}\|^2)\pi_i}{\sum_{i=1}^K \exp(-\frac{1}{2\sigma^2}\|x_j - \mu_i^{t-1}\|^2)\pi_i}$$

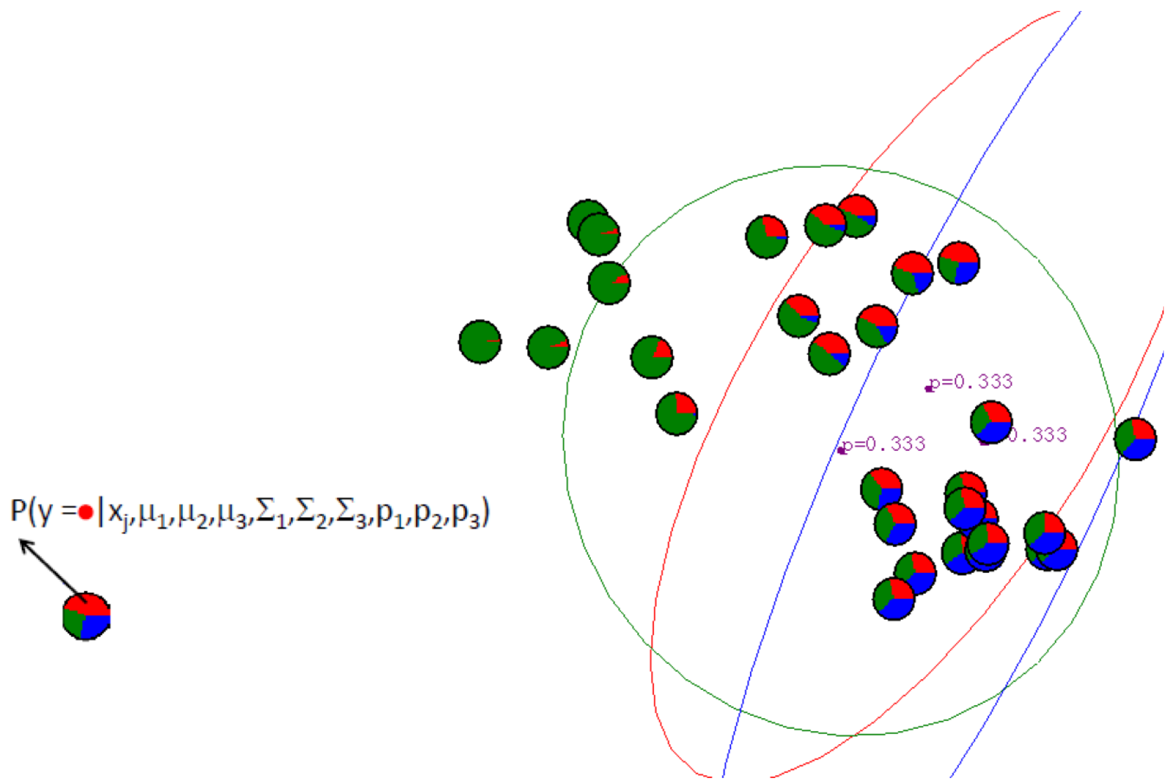
In K-means "E-step" we do hard assignment. EM does soft assignment

M-step

Compute Max. like **μ** given our data's class membership distributions (weights)

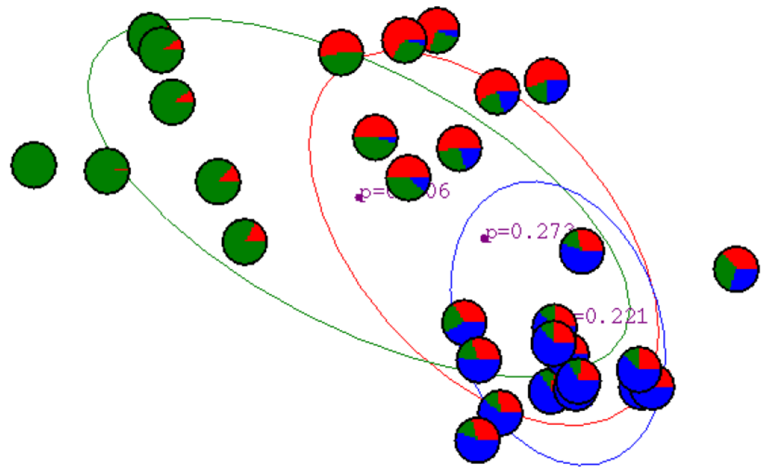
$$\mu_i^t = \sum_{j=1}^n w_j x_j \quad \text{where } w_j = \frac{P(y_j=i|x_j,\theta^{t-1})}{\sum_{l=1}^n P(y_l=i|x_l,\theta^{t-1})}$$

Iterate. Exactly the same as MLE with weighted data.

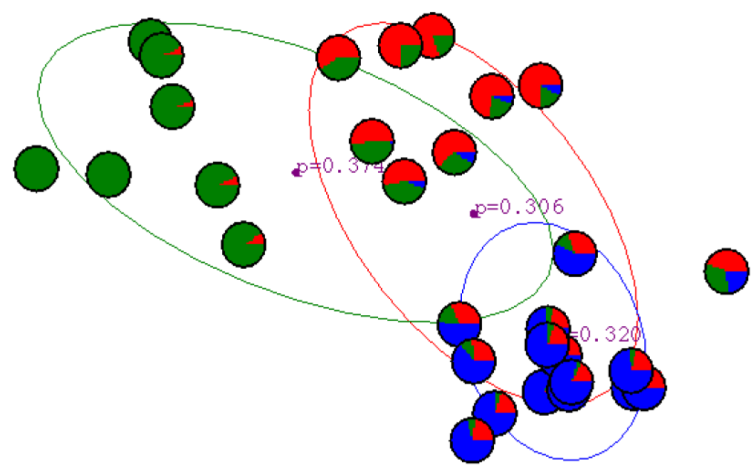


$P(y = \bullet \mid x_j, \mu_1, \mu_2, \mu_3, \Sigma_1, \Sigma_2, \Sigma_3, p_1, p_2, p_3)$
p=0.333
p=0.333
0.333

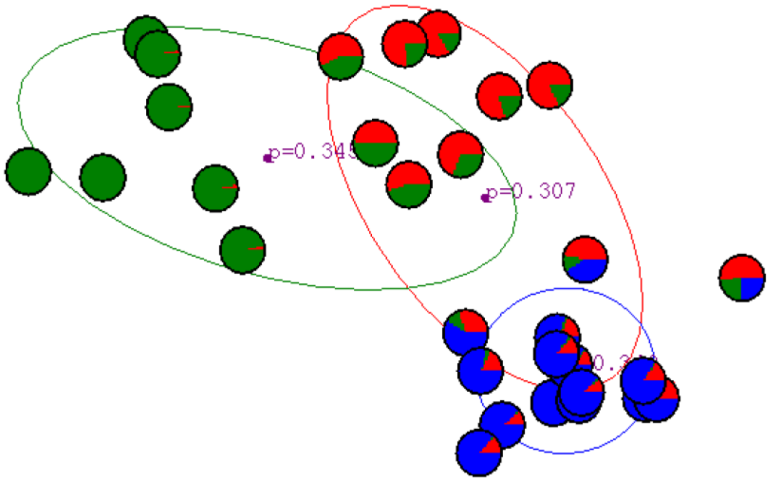
After 1st iteration



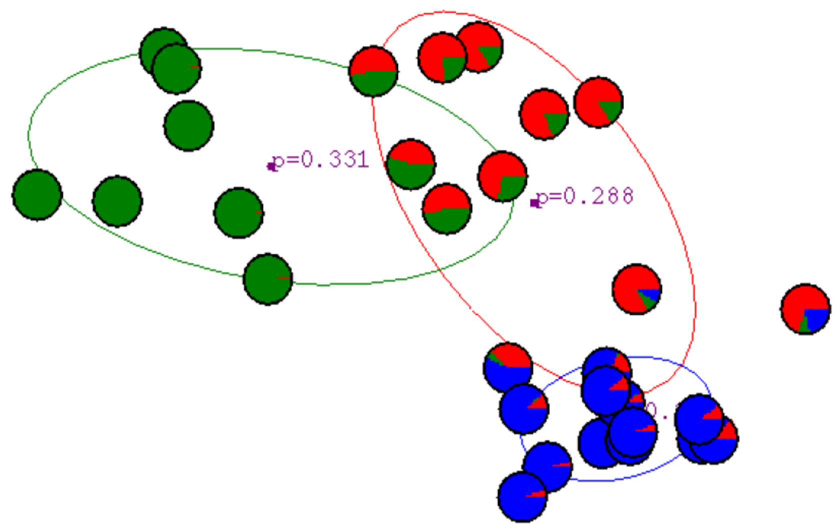
After 2nd iteration



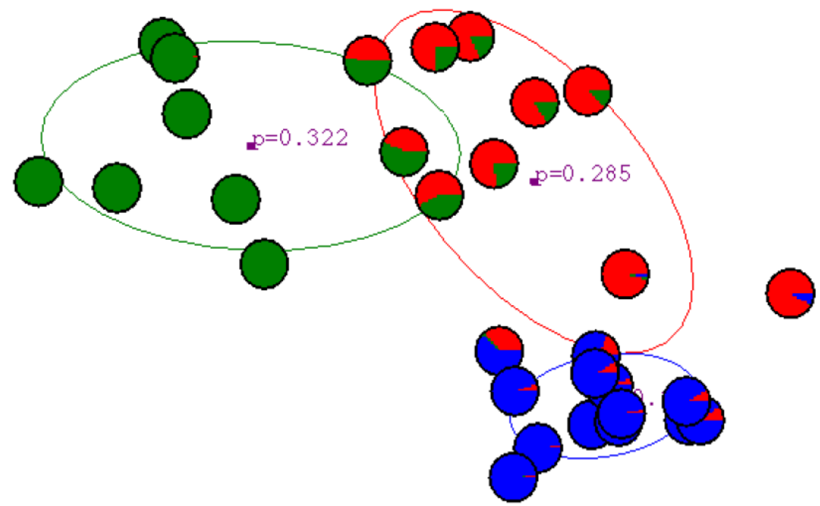
After 3rd iteration



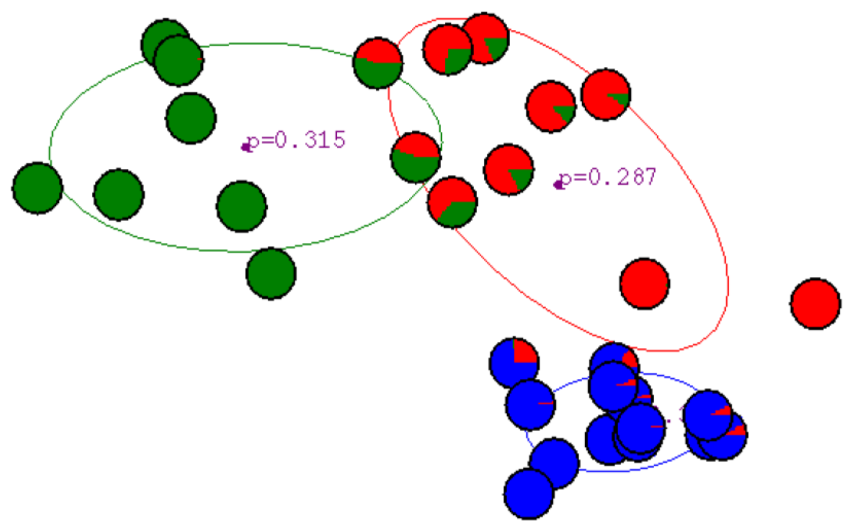
After 4th iteration



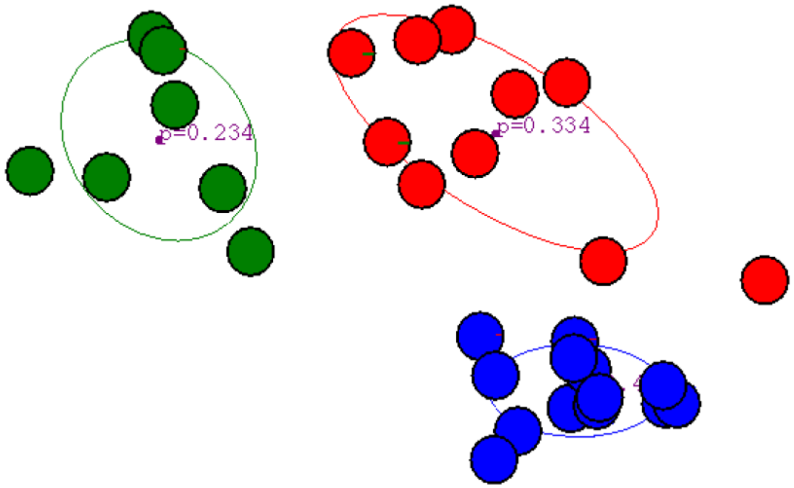
After 5th iteration



After 6th iteration



After 20th iteration



General EM - Frequentist

Notation

Observed data: $D = \{x_1, \ldots, x_n\}$

Unknown variables: y

For example in clustering: $y = (y_1, \ldots, y_n)$

Paramaters: θ

For example in MoG: $\theta = [\mu_1, \ldots, \mu_K, \pi_1, \ldots, \pi_K, \Sigma_1, \ldots, \Sigma_K]$

Goal: $\widehat{\theta}_n = \arg\max_{\theta} \log P(D|\theta)$

General EM - Bayesian

Notation

Observed data: $D = \{x_1, \ldots, x_n\}$

Unknown variables: y

For example in clustering: $y = (y_1, \ldots, y_n)$

Paramaters: θ

Prior: $P(\theta)$

Goal: $\widehat{\theta}_n = \arg\max_{\theta} \log P(D|\theta) + \log P(\theta)$

Goal: $\max_{\theta} p(X|\theta)$

$$
\begin{aligned}
\log p(X|\theta) &= \log \sum_{Z} p(X, Z|\theta) = \sum_{i} \log \sum_{k} p(x_i, z_i = k|\theta) \\
&= \sum_{i} \log \sum_{k} \frac{q(z_i = k|x_i)}{q(z_i = k|x_i)} p(x_i, z_i = k|\theta) \\
&= \sum_{i} \log \sum_{k} q(z_i = k|x_i) \frac{p(x_i, z_i = k|\theta)}{q(z_i = k|x_i)} \\
&\geq \sum_{i} \sum_{k} q(z_i = k|x_i) \log \frac{p(x_i, z_i = k|\theta)}{q(z_i = k|x_i)} \\
&= \sum_{i} \sum_{k} q(z_i = k|x_i) \log \frac{p(x_i|\theta) p(z_i|x_i, \theta)}{q(z_i = k|x_i)} \\
&:= F(q, \theta)
\end{aligned}
$$

Goal: $\max_{\theta} p(\theta|X)$

$$
\begin{aligned}
\log p(X,\theta) &= \log \sum_Z p(X,Z,\theta) \\
&= \log \sum_Z p(X,Z|\theta) + \sum_k \log p(\theta_k) \\
&= \sum_i \log \sum_k p(x_i, z_i = k|\theta) + \sum_k \log p(\theta_k) \\
&= \sum_i \log \sum_k \frac{q(z_i = k|x_i)}{q(z_i = k|x_i)} p(x_i, z_i = k|\theta) + \boxed{\sum_k \log p(\theta_k)} \\
&= \sum_i \log \sum_k q(z_i = k|x_i) \frac{p(x_i, z_i = k|\theta)}{q(z_i = k|x_i)} + \boxed{\sum_k \log p(\theta_k)} \\
&\geq \sum_i \sum_k q(z_i = k|x_i) \log \frac{p(x_i, z_i = k|\theta)}{q(z_i = k|x_i)} + \boxed{\sum_k \log p(\theta_k)} \\
&:= F(q,\theta)
\end{aligned}
$$

$$F(q,\theta) = \sum_i \sum_k q(z_i = k|x_i) \log \frac{p(x_i, z_i = k|\theta, \pi)}{q(z_i = k|x_i)} + \sum_k \log p(\theta_k)$$

$$= -\sum_i D_{KL}(q(z_i|x_i)||P(z_i|x_i,\theta)) + \log P(x_i|\theta) + \sum_k \log p(\theta_k)$$

- **E-Step:** Maximize over q keeping θ fixed

$$q^{(t)} = \arg\max_q F(q, \theta^{(t-1)}) = p(z_i|x_i, \theta^{(t-1)})$$

- **M-Step:** Maximize over θ keeping q fixed

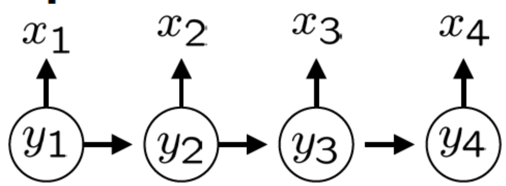
$$q^{(t)} = \arg\max_\theta F(q^{(t)}, \theta) = \arg\max_\theta \sum_k \sum_i q_{ik}^{(t)} \log p(x_i|\theta_k) + \log p(\theta_k)$$

MLE or MAP on weighted data!

Theorem: During the EM algorithm the marginal likelihood is not decreasing!

$$p(X|\theta^{(t-1)}) \leq p(X|\theta^{(t)})$$

Other Examples: Hidden Markov Models



Observed data: $D = \{x_1, \ldots, x_n\}$

Unknown variables: $y = (y_1, \ldots, y_n)$

Paramaters: θ $\theta = [\pi_1, \ldots, \pi_K, A, B]$

Initial probabilities: $P(x_1 = i) = \pi_i,\ i = 1, \ldots, K$

Transition probabilities: $P(y_{t+1} = j | y_t = i) = A_{ij}$

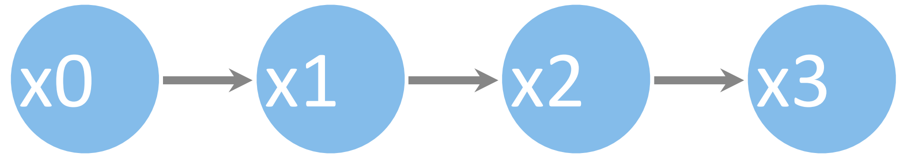
Emission probabilities: $P(x_{t+1} = l | x_t = i) = B_{il}$

Goal:

$$\widehat{\theta}_n = \arg\max_{\theta} \log P(D|\theta) = \arg\max_{\pi, A, B} \log P(x_1, \ldots, x_n|\theta)$$

Chains

$$p(x;\theta) = p(x_0;\theta)\prod_{i=1}^{n-1} p(x_{i+1}|x_i;\theta)$$



Transition Matrices

x0	
0	0.4
1	0.6

x1 \ x0	0	1
0	0.2	0.1
1	0.8	0.9

x2 \ x1	0	1
0	0.8	0.5
1	0.2	0.5

x3 \ x2	0	1
0	0	1
1	1	0

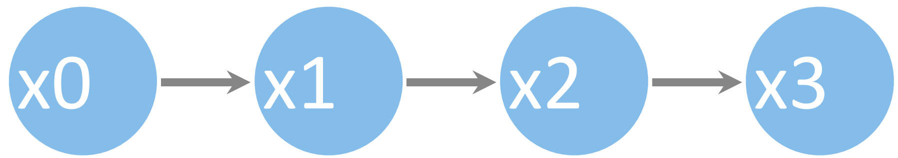
Unraveling the chain

$$p(x_1) = \sum_{x_0} p(x_1|x_0)p(x_0) \iff \pi_1 = \Pi_{0\to 1}\pi_0$$

$$p(x_2) = \sum_{x_1} p(x_2|x_1)p(x_1) \iff \pi_2 = \Pi_{1\to 2}\pi_1 = \Pi_{1\to 2}\Pi_{0\to 1}\pi_0$$

Chains

$$p(x;\theta) = p(x_0;\theta)\prod_{i=1}^{n-1} p(x_{i+1}|x_i;\theta)$$



- Transition matrices

x0	
0	0.4
1	0.6

x1 \ x0	0	1
0	0.2	0.1
1	0.8	0.9

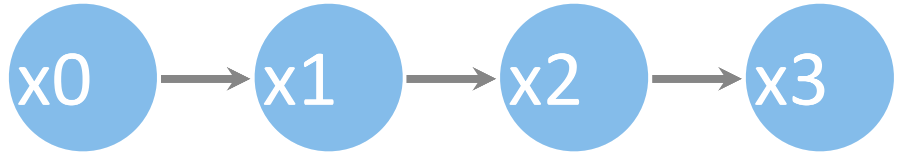
x2 \ x1	0	1
0	0.8	0.5
1	0.2	0.5

x3 \ x2	0	1
0	0	1
1	1	0

```
x0  = [0.4; 0.6];
Pi1 = [0.2 0.1; 0.8 0.9];
Pi2 = [0.8 0.5; 0.2 0.5];
Pi3 = [0 1; 1 0];
x3  = Pi3 * Pi2 * Pi1 * x0 = [0.45800; 0.54200]
```

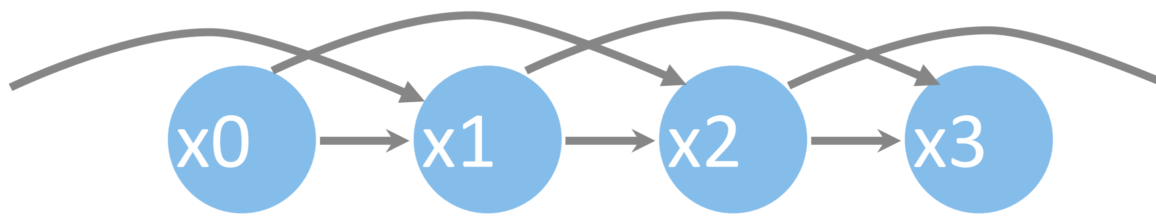
Markov Chains

- First order chain



$$p(X) = p(x_0) \prod_i p(x_{i+1}|x_i)$$

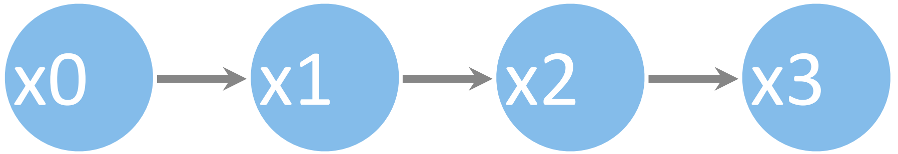
- Second order

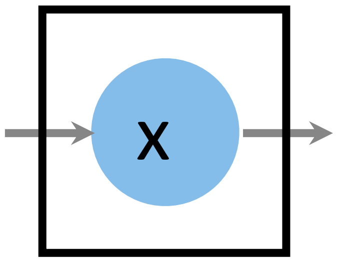


$$p(X) = p(x_0, x_1) \prod_i p(x_{i+1}|x_i, x_{i-1})$$

Chains

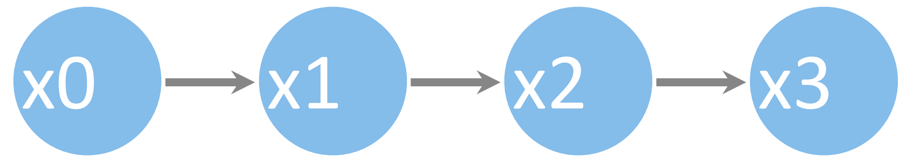
$$p(x;\theta) = p(x_0;\theta)\prod_{i=1}^{n-1} p(x_{i+1}|x_i;\theta)$$





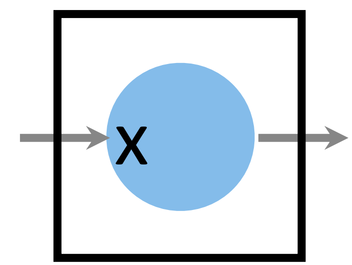
Chains

$$p(x;\theta) = p(x_0;\theta)\prod_{i=1}^{n-1} p(x_{i+1}|x_i;\theta)$$



$p(x_i) = l_i(x$ … $n-1$

$= l_i(x$ …



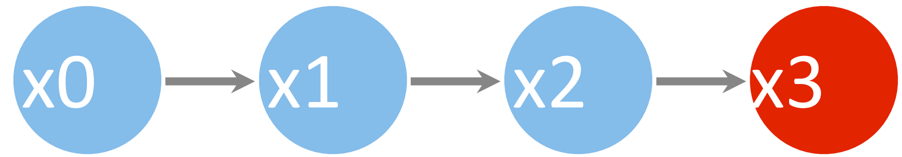
$= l_i(x_i)$ … $\sum_{x_{i+1}\ldots x_{n-2}} \prod_{j=i}$ … $\sum_{x_{n-1}}$ … $_1(x_{n-1})$

$:= r_{n-2}(x_{n-2})$

not needed for directed graphs that are already normalized
... but good to know ...

$$\pi_i = \prod_{j=1}^{i} \Pi_{j-1\to j}\pi_0$$

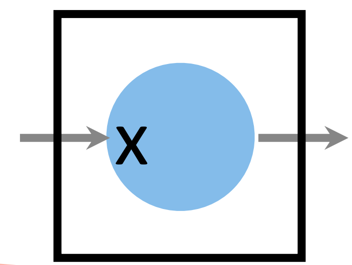
Chains



$$p(x_{1\dots n-1}|x_n;\theta) = p(x_0|\theta)\prod_{i=1}^{n-1} p(x_{i+1}|x_i;\theta)$$

$$p(x_i|x_n) = l_i(x_i)\sum_{x_{i+1}\dots x_{n-1}}\prod_{j=i}^{n-1} p(x_{j+1}|x_j)$$

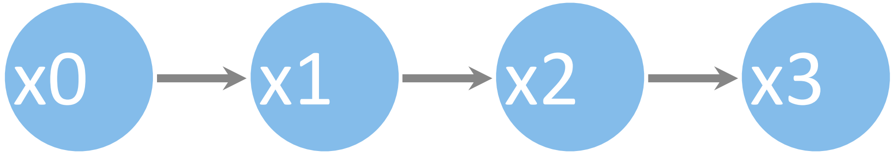
$$= l_i(x_i)\sum_{x_{i+1}\dots x_{n-1}}\prod_{j=i}^{n-2} p(x_{j+1}|x_j)\underbrace{p(x_n|x_{n-1})}_{:=r_{n-1}(x_{n-1})}$$



$$= l_i(x_i)\sum_{x_{i+1}\dots x_{n-2}}\prod_{j=i}^{n-3} p(x_{j+1}|x_j)\underbrace{\sum_{x_{n-1}} p(x_{n-1}|x_{n-2})\,r_{n-1}(x_{n-1})}_{:=r_{n-2}(x_{n-2})}$$

Chains

$$p(x;\theta) = p(x_0;\theta) \prod_{i=1}^{n-1} p(x_{i+1}|x_i;\theta)$$



- Forward recursion

$$l_0(x_0) := p(x_0) \text{ and } l_i(x_i) := \sum_{x_{i-1}} l_{i-1}(x_{i-1}) p(x_i|x_{i-1})$$

- Backward recursion

$$r_n(x_n) := 1 \text{ and } r_i(x_i) := \sum_{x_{i+1}} r_{i+1}(x_{i+1}) p(x_{i+1}|x_i)$$

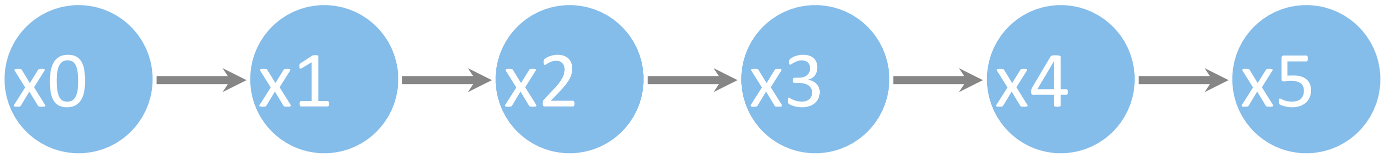
- Marginalization & conditioning

$$p(x_i) = l_i(x_i) r_i(x_i)$$

$$p(x_{-i}|x_i) = \frac{p(x)}{p(x_i)}$$

$$p(x_i, x_{i+1}) = l_i(x_i) p(x_{i+1}|x_i) r_i(x_{i+1})$$

Chains



$$l_i = \Pi_i l_{i-1}$$
$$r_i = \Pi_i^\top r_{i+1}$$

- Send forward messages starting from left node

$$m_{i-1\to i}(x_i) = \sum_{x_{i-1}} m_{i-2\to i-1}(x_{i-1}) f(x_{i-1}, x_i)$$

- Send backward messages starting from right node

$$m_{i+1\to i}(x_i) = \sum_{x_{i+1}} m_{i+2\to i+1}(x_{i+1}) f(x_i, x_{i+1})$$

Example - inferring lunch

current

		
	0.9	0.2
	0.1	0.8

- Initial probability p(x0=t)=p(x0=b) = 0.5
- Stationary transition matrix
- On fifth day observed at Tazza d'oro p(x5=t)=1
- Distribution on day 3
  - Left messages to 3
  - Right messages to 3
  - Renormalize

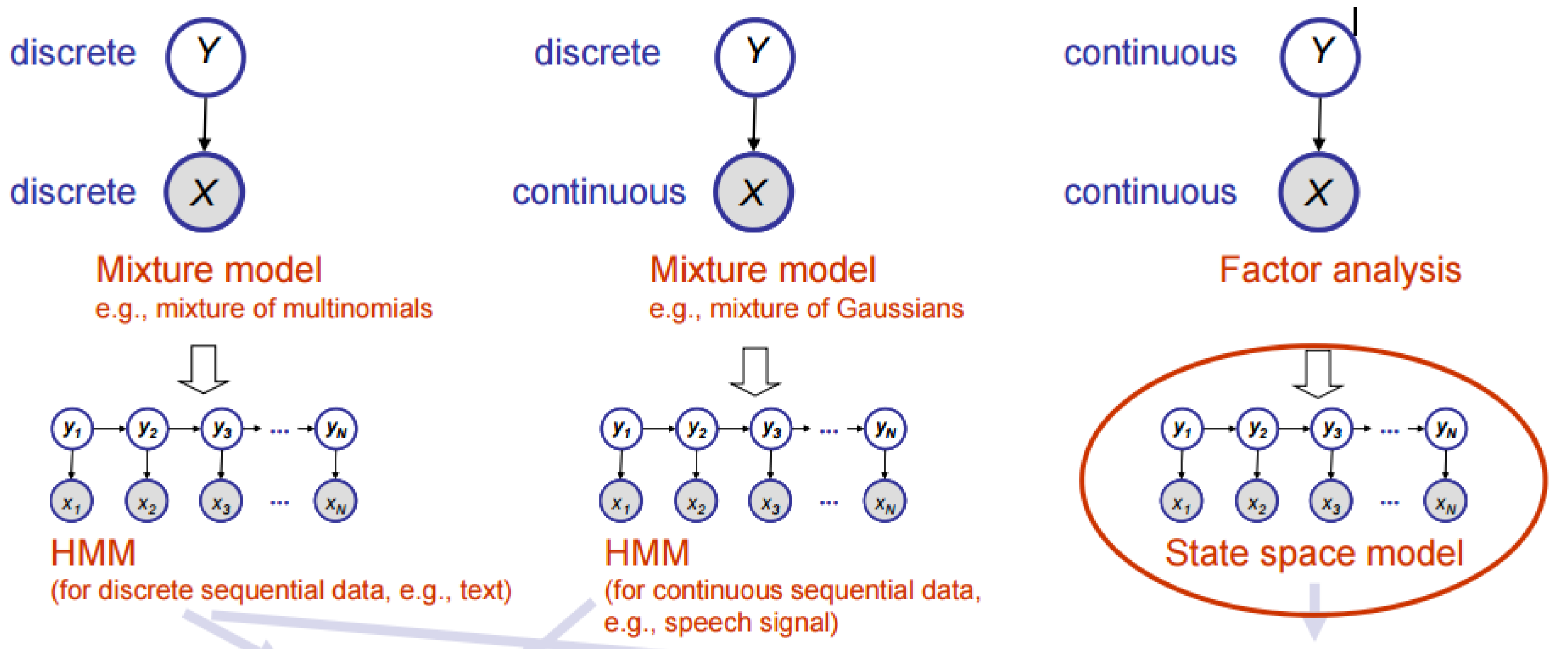
Example - inferring lunch

current

		
	0.9	0.2
	0.1	0.8

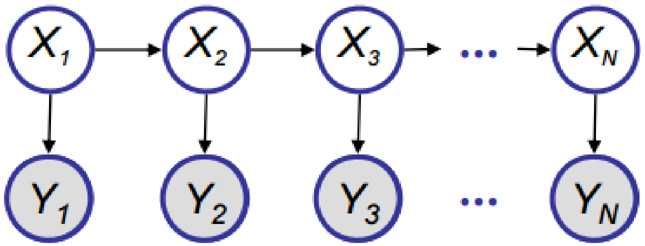
```
> Pi = [0.9, 0.2; 0.1 0.8]
Pi =
   0.90000   0.20000
   0.10000   0.80000
> l1 = [0.5; 0.5];
> l3 = Pi * Pi * l1
l3 =
   0.58500
   0.41500
> r5 = [1; 0];
> r3 = Pi' * Pi' * r5
r3 =
   0.83000
   0.34000
> (l3 .* r3) / sum(l3 .* r3)
ans =
   0.77483
   0.22517
```

Generalizing



State Space Model

- A sequential FA or a continuous state HMM



$$\mathbf{x}_t = A\mathbf{x}_{t-1} + Gw_t$$
$$\mathbf{y}_t = C\mathbf{x}_{t-1} + v_t$$
$$w_t \sim \mathcal{N}(0; Q), \quad v_t \sim \mathcal{N}(0; R)$$
$$\mathbf{x}_0 \sim \mathcal{N}(0; \Sigma_0),$$

This is a linear dynamic system.

- In general,

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}) + Gw_t$$
$$\mathbf{y}_t = g(\mathbf{x}_{t-1}) + v_t$$

where f is an (arbitrary) dynamic model, and g is an (arbitrary) observation model

Markov Chains

Markov chain:

$$P(X_{t+1}|X_t, \ldots, X_1) = P(X_{t+1}|X_t)$$

Homogen Markov chain:

$P(X_{t+1}|X_t)$ is invariant for all t.

Definitions

- Assume that the state space is finite:

$$\mathcal{X} = \{1, \ldots, k\}.$$

- 1-Step state transition matrix:

$$T_{ij} = P(X_{t+1} = j | X_t = i)$$

Lemma: The state transition matrix is stochastic:

$$\sum_j T_{ij} = 1 \;\; \forall i$$

j

$i \quad T_{ij}$

- t-Step state transition matrix:

$$Q_{ij} \doteq P(X_{k+t} = j | X_k = i)$$

Lemma:

$$P(X_{k+t} = j | X_k = i) = Q_{ij} = [T^t]_{ij}, \; \forall (k, i, j)$$

Limit behaviour

$$T = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0.1 & 0.9 \\ 0.6 & 0.4 & 0 \end{bmatrix}$$

If the probability vector for the initial state is $\mu(x^{(1)}) = (0.5, 0.2, 0.3)$

it follows that $\mu(x^{(1)})T = (0.2, 0.6, 0.2)$

and, after several iterations (multiplications by T)

$\mu(x^{(1)})T^t \rightarrow p(x) = (0.22, 0.41, 0.37)$ **stationary distribution**

no matter what initial distribution $\mu(x^1)$ was.

$$T^{\infty} = \begin{bmatrix} 0.22 & 0.41 & 0.37 \\ 0.22 & 0.41 & 0.37 \\ 0.22 & 0.41 & 0.37 \end{bmatrix}$$

The chain has forgotten its past.

Definition: [stationary distribution, invariant distribution]

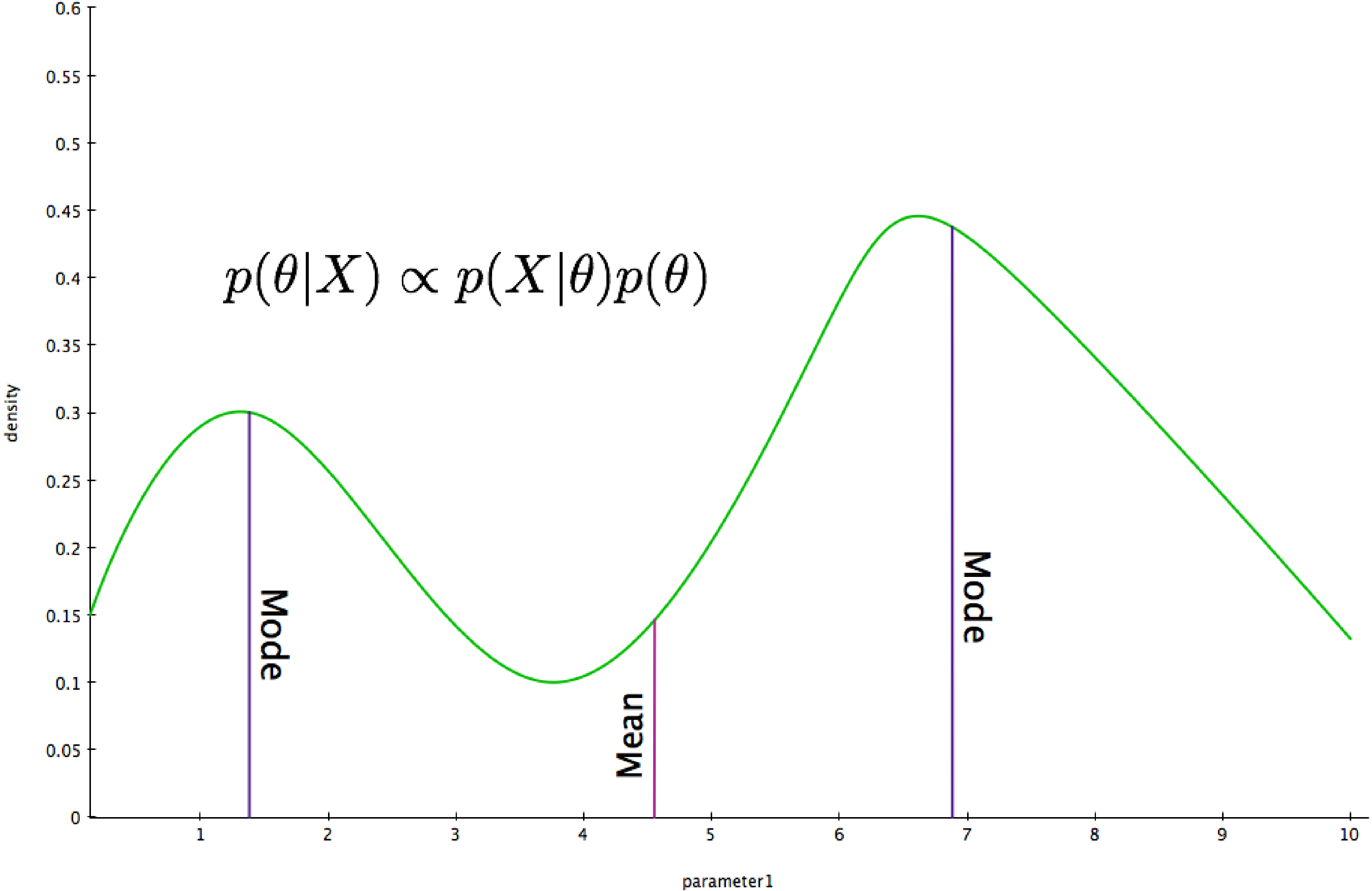
The distribution $\pi = (\pi_1, \ldots, \pi_k)$ is **stationary** distribution if $\pi_i \geq 0 \; \forall i$, $\sum_{i=1}^{T} \pi_i = 1$, and $\pi \mathbf{T} = \pi$.

Theorem:

$$\pi \mathbf{T} = \pi.$$

- π is the left eigenvector of the matrix *T* with eigenvalue 1.
- The Perron-Frobenius theorem from linear algebra tells us that the remaining eigenvalues have absolute value less than 1.
- The second largest eigenvalue, therefore, determines the rate of convergence of the chain, and should be as small as possible.

Is maximization (always) good?



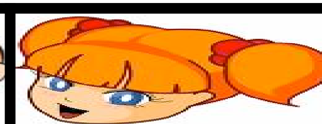
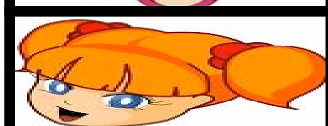
Sampling

- Key idea
  - Want accurate distribution of the posterior
  - Sample from posterior distribution rather than maximizing it
- Problem - direct sampling is usually intractable
- Solutions
  - Markov Chain Monte Carlo (complicated)
  - Gibbs Sampling (somewhat simpler)

$$x \sim p(x|x') \text{ and then } x' \sim p(x'|x)$$

Gibbs sampling

- Gibbs sampling:
  - In most cases direct sampling not possible
  - Draw one set of variables at a time

		
	0.45	0.05
	0.05	0.45

(b,g) - draw p(.,g)
(g,g) - draw p(g,.)
(g,g) - draw p(.,g)
(b,g) - draw p(b,.)
(b,b) ...

Gibbs Sampling

- The basic idea is to split the multidimensional $\boldsymbol{\theta}$ into blocks (often scalars) and sample each block separately, conditional on the most recent values of the other blocks
- The beauty of Gibbs sampling is that it simplifies a complex high-dimensional problem by breaking it down into simple, low-dimensional problems

Gibbs Sampling

- Formally, the algorithm proceeds as follows, where θ consists of k blocks $\theta_1, \theta_2, \ldots, \theta_k$: at iteration (t),
  - Draw $\theta_1^{(t+1)}$ from

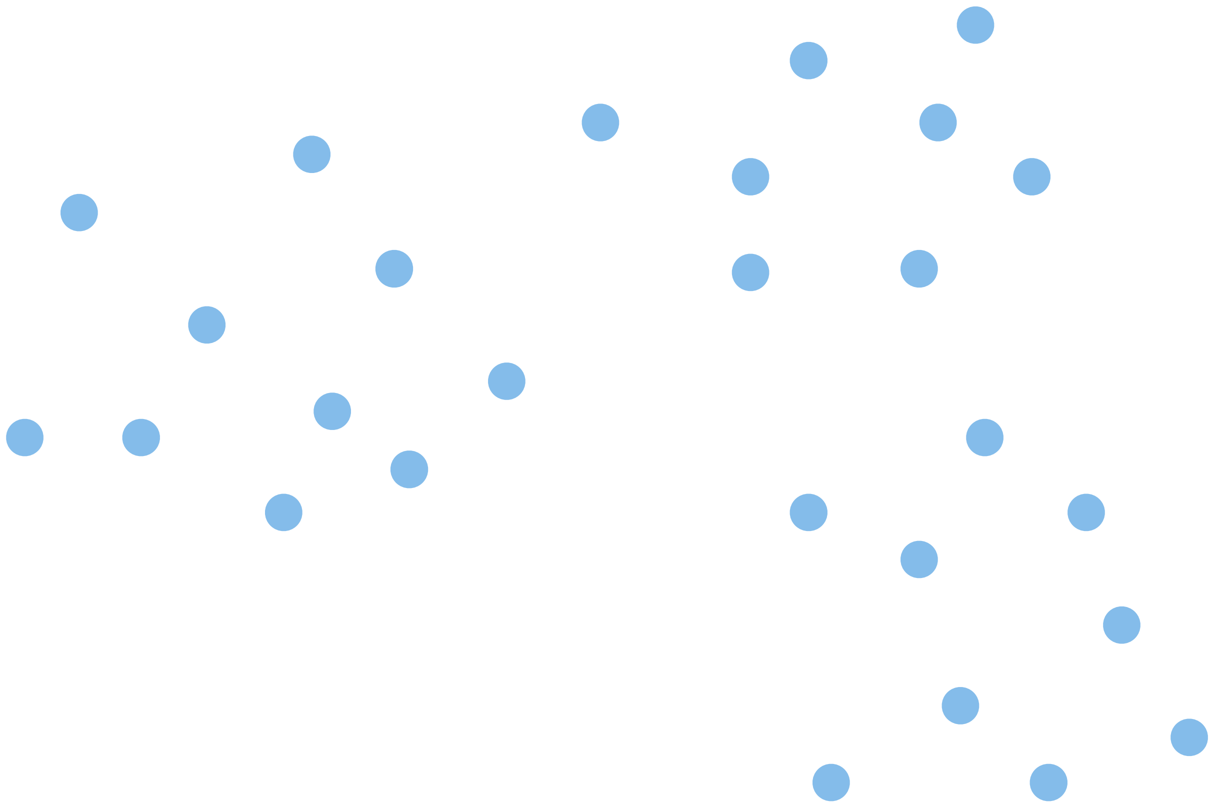
$$p(\theta_1|\theta_2^{(t)}, \theta_3^{(t)}, \ldots, \theta_k^{(t)})$$

  - Draw $\theta_2^{(t+1)}$ from

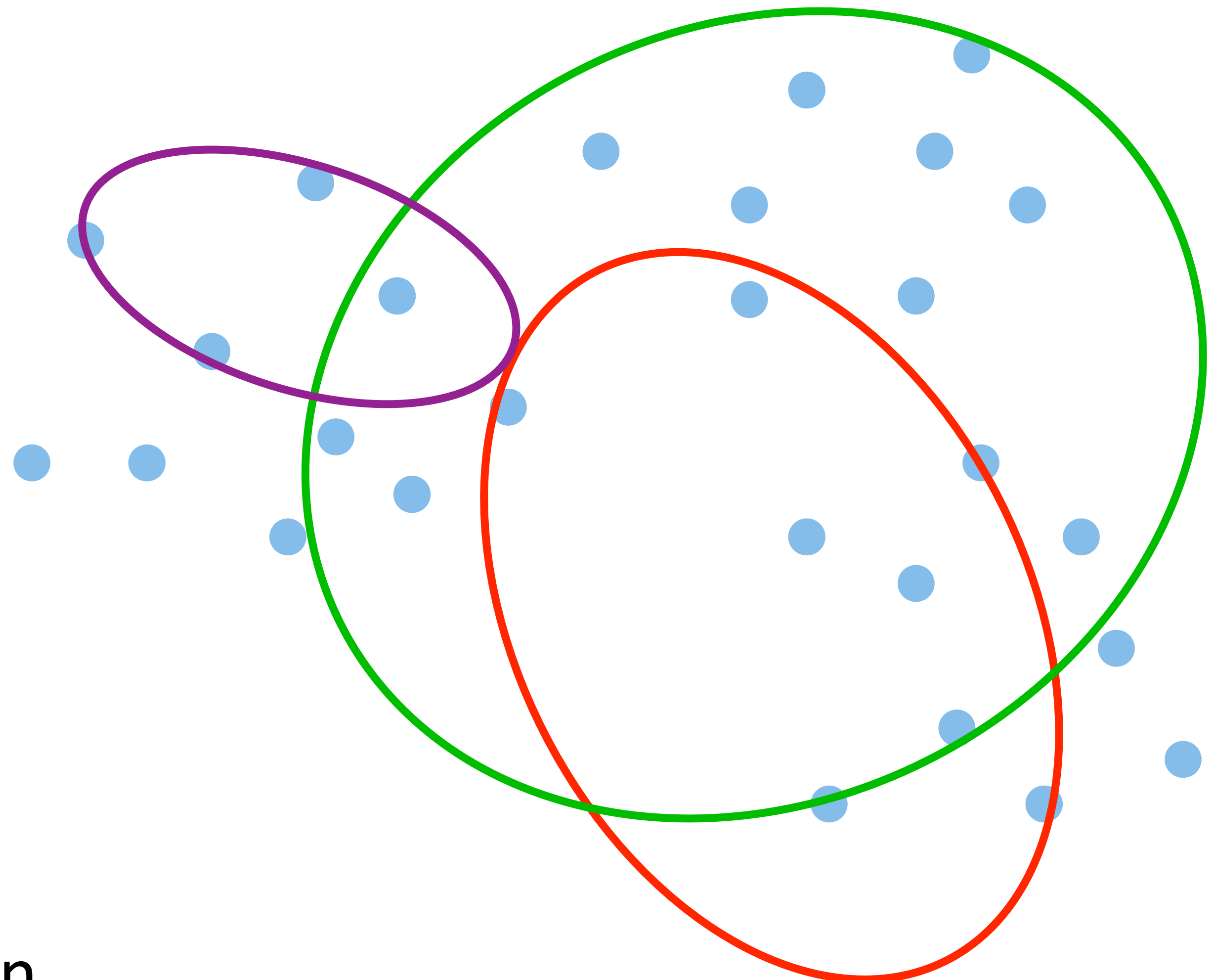
$$p(\theta_2|\theta_1^{(t+1)}, \theta_3^{(t)}, \ldots, \theta_k^{(t)})$$

  - $\ldots$
- This completes one iteration of the Gibbs sampler, thereby producing one draw $\theta^{(t+1)}$; the above process is then repeated many times
- The distribution $p(\theta_1|\theta_2^{(t)}, \theta_3^{(t)}, \ldots, \theta_k^{(t)})$ is known as the *full conditional* distribution of θ_1

Gibbs sampling for clustering

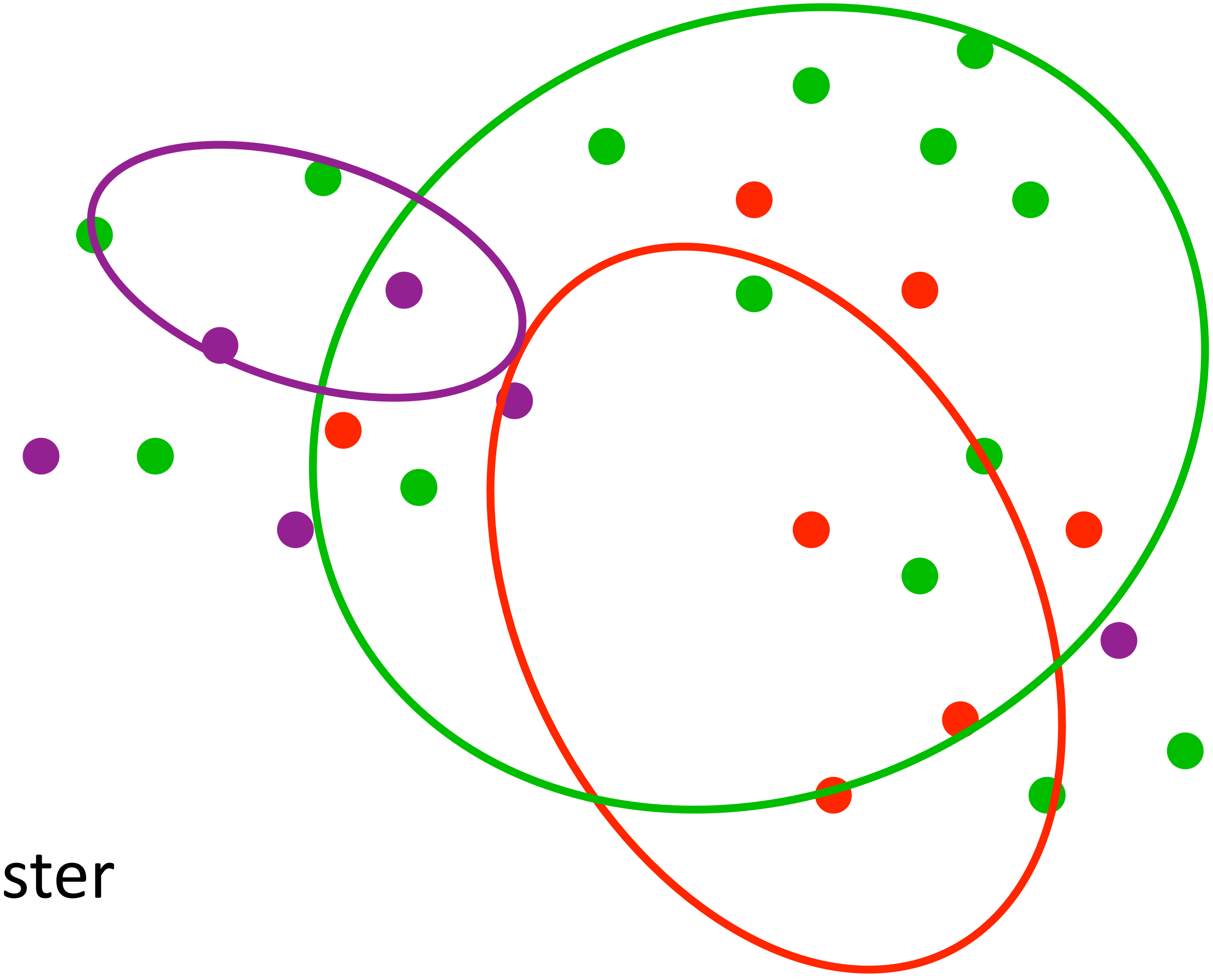


Gibbs sampling for clustering



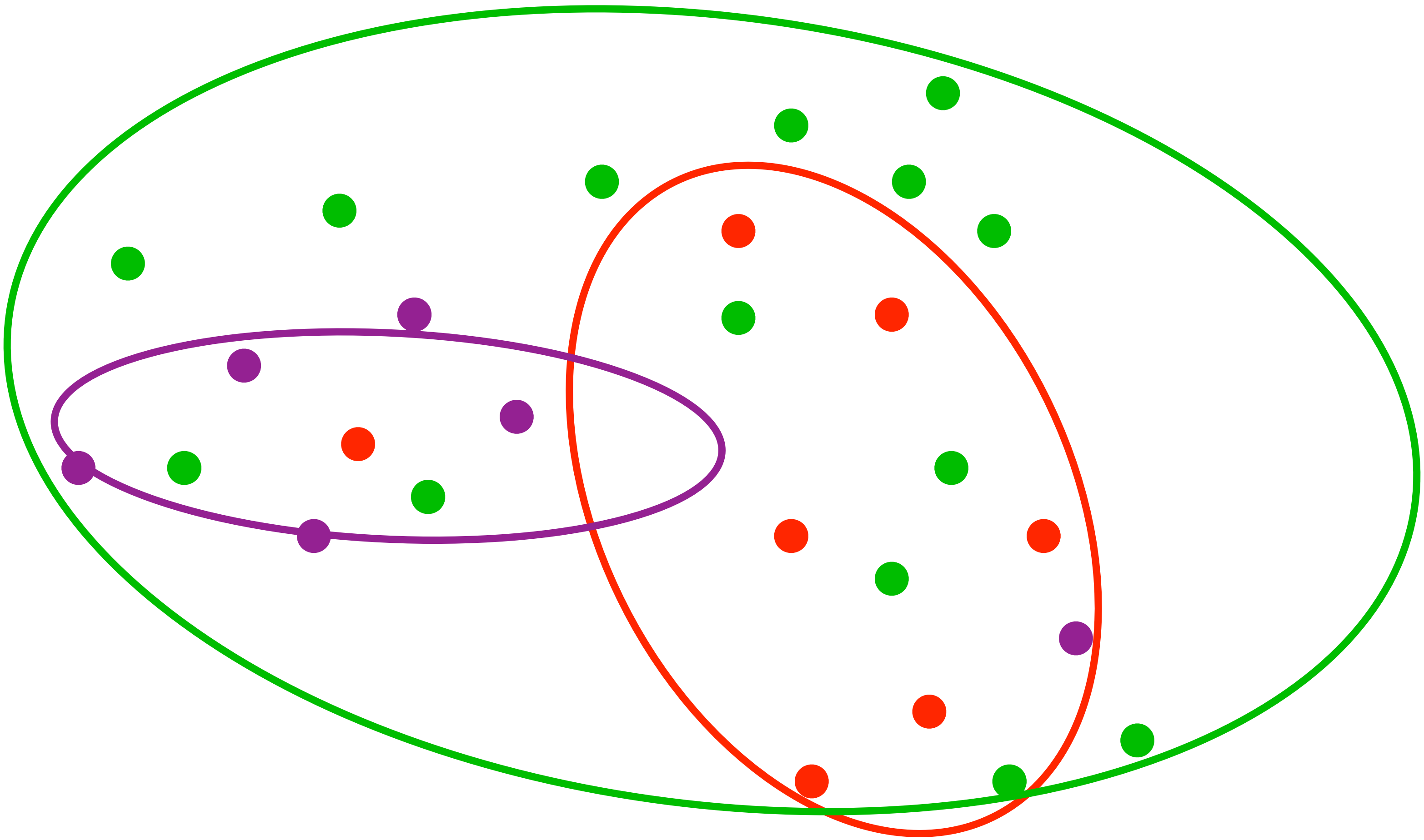
random
initialization

Gibbs sampling for clustering



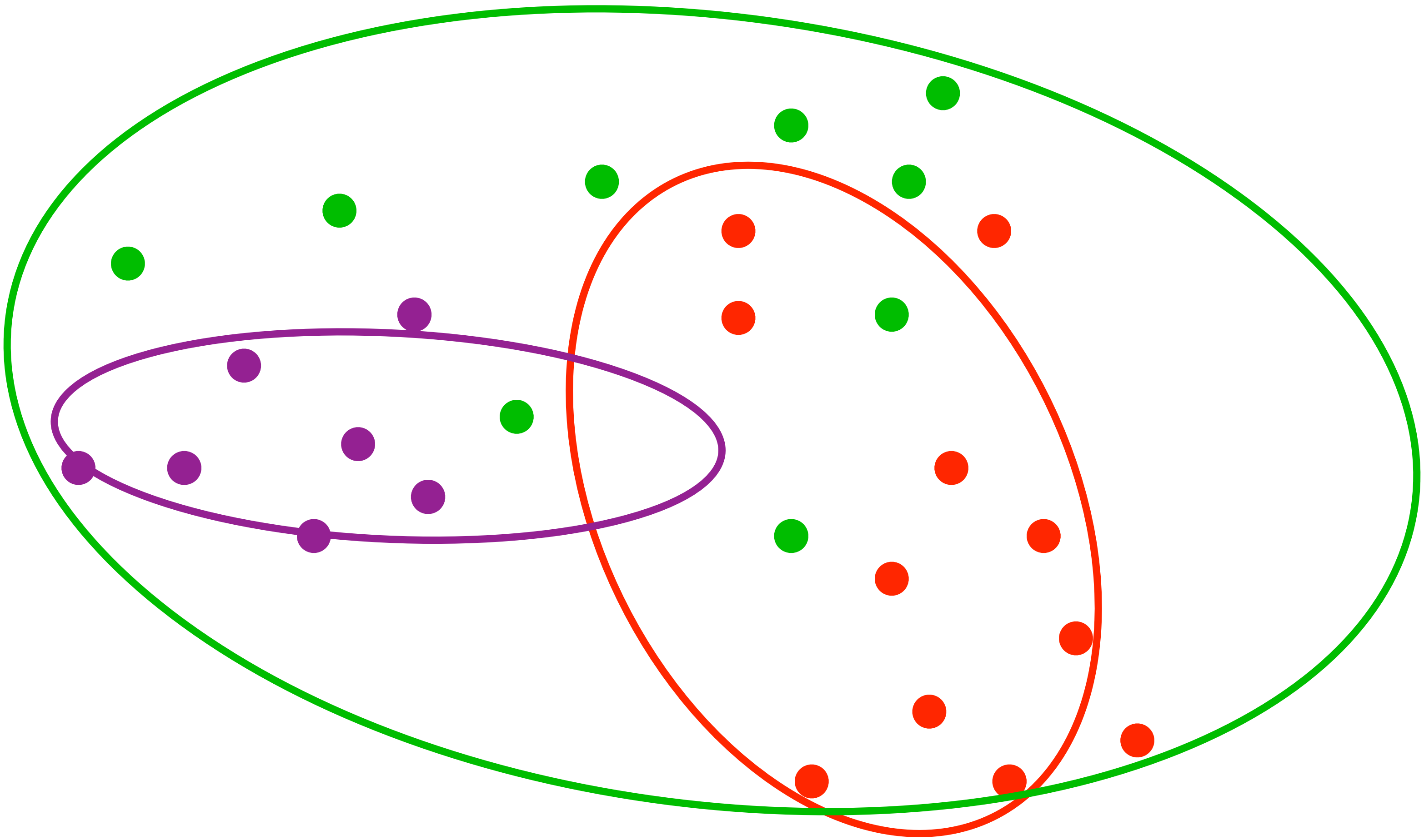
sample cluster labels

Gibbs sampling for clustering



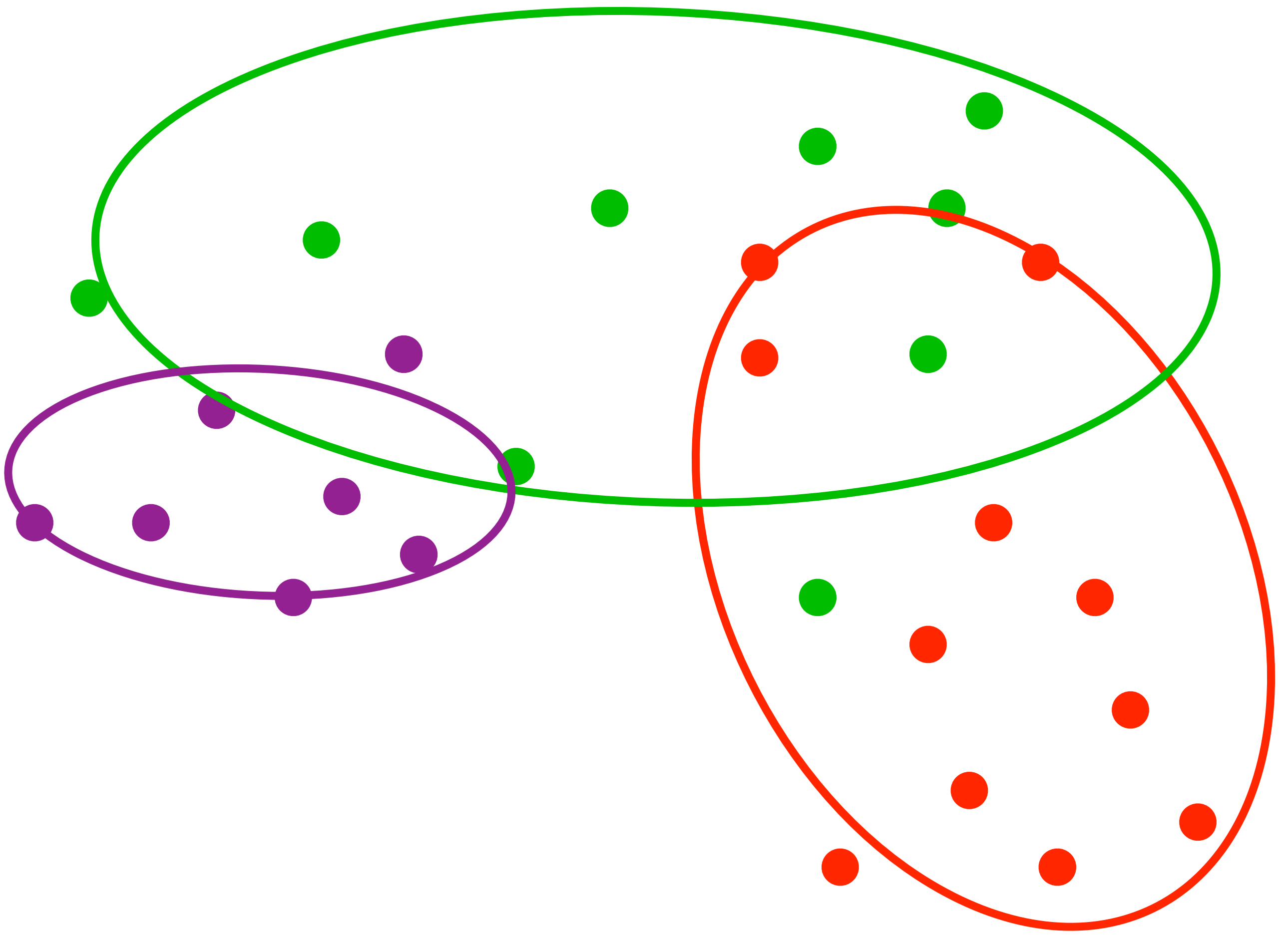
resample
cluster model

Gibbs sampling for clustering



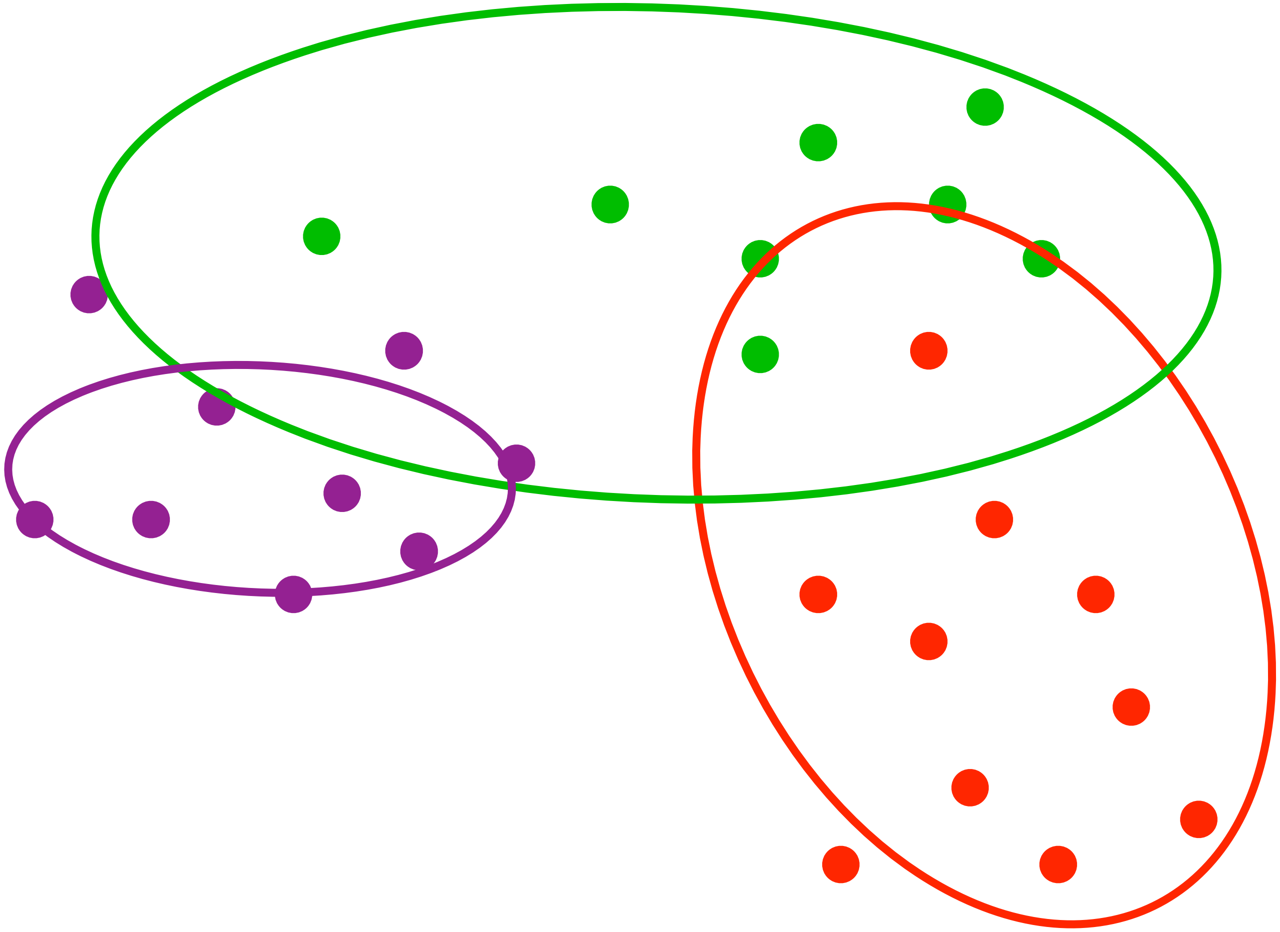
resample
cluster labels

Gibbs sampling for clustering



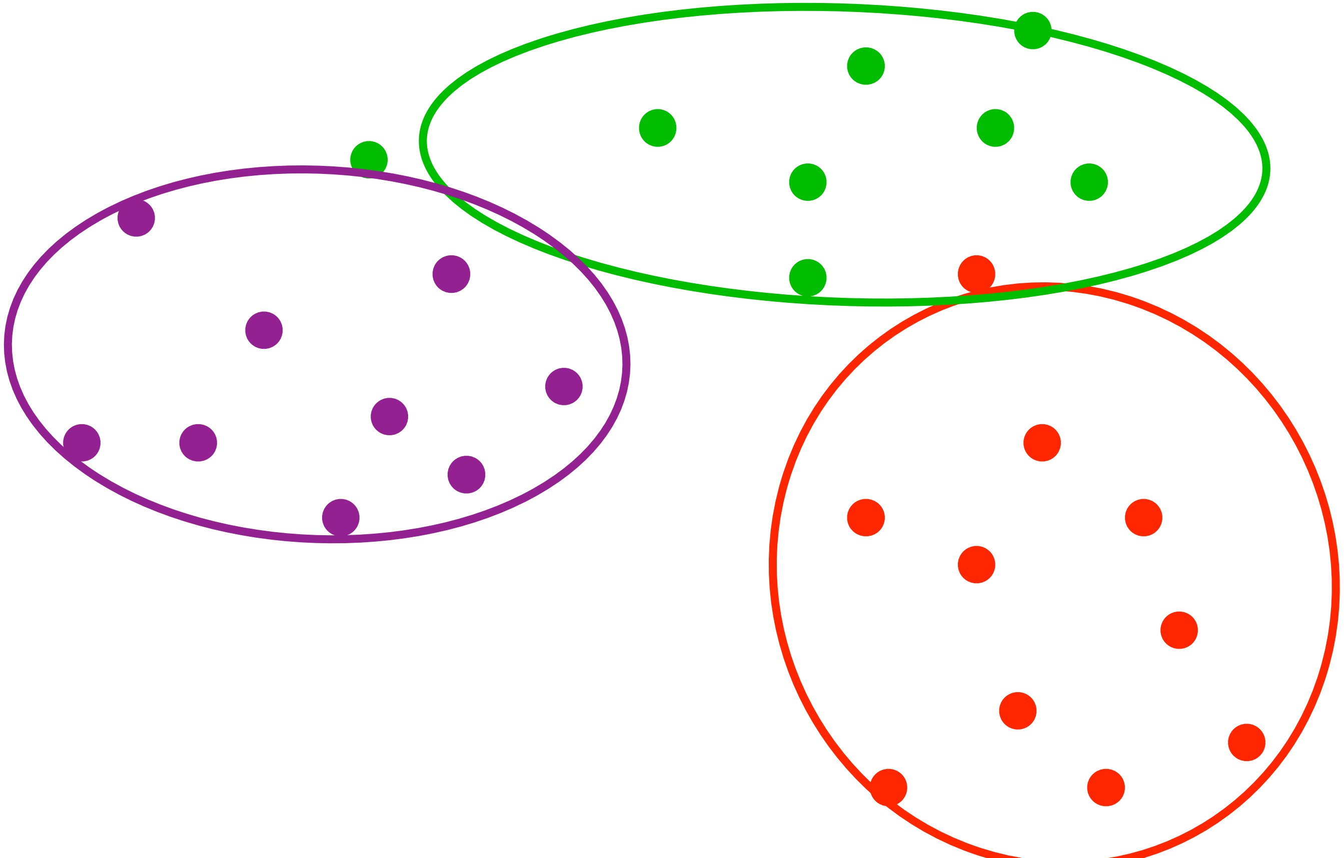
resample
cluster model

Gibbs sampling for clustering



resample
cluster labels

Gibbs sampling for clustering



resample
cluster model   e.g. Mahout Dirichlet Process Clustering

Inference Algorithm ≠ Model

Corollary: EM ≠ Clustering

… but some algorithms and models are good match …

Reminder on Kernels

- Remember Kernels are nothing but implicit feature maps

$$\phi : \mathcal{X} \to \mathbb{R}^d$$

- Gram Matrix
  - $G_{ij} = K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \qquad \forall i, j \in 1 \ldots n$
  - of a set of vectors $x_1 \ldots x_n$ in the inner product space defined by the kernel K

- Gram Matrix is always positive definite

Gaussian Process

Correlated Observations

Assume that the random variables $t \in \mathbb{R}^n, t' \in \mathbb{R}^{n'}$ are jointly normal with mean (μ, μ') and covariance matrix K

$$p(t, t') \propto \exp\left(-\frac{1}{2}\begin{bmatrix} t-\mu \\ t'-\mu' \end{bmatrix}^\top \begin{bmatrix} K_{tt} & K_{tt'} \\ K_{tt'}^\top & K_{t't'} \end{bmatrix}^{-1} \begin{bmatrix} t-\mu \\ t'-\mu' \end{bmatrix}\right).$$

Inference

Given t, estimate t' via $p(t'|t)$. Translation into machine learning language: we learn t' from t.

Practical Solution

Since $t'|t \sim \mathcal{N}(\tilde{\mu}, \tilde{K})$, we only need to collect all terms in $p(t, t')$ depending on t' by matrix inversion, hence

$$\tilde{K} = K_{t't'} - K_{tt'}^\top K_{tt}^{-1} K_{tt'} \text{ and } \tilde{\mu} = \mu' + K_{tt'}^\top \underbrace{\left[K_{tt}^{-1}(t-\mu)\right]}_{\text{independent of } t'}$$

Handbook of Matrices, Lütkepohl 1997 (big timesaver)

Carnegie Mellon University

Additive Noise

Indirect Model

Instead of observing $t(x)$ we observe $y = t(x) + \xi$, where ξ is a nuisance term. This yields

$$p(Y|X) = \int \prod_{i=1}^{m} p(y_i|t_i) p(t|X) dt$$

where we can now find a maximum a posteriori solution for t by maximizing the integrand (we will use this later).

Additive Normal Noise

- If $\xi \sim \mathcal{N}(0, \sigma^2)$ then y is the sum of two Gaussian random variables.
- Means and variances **add up**.

$$y \sim \mathcal{N}(\mu, K + \sigma^2 \mathbf{1}).$$

Posterior is also Gaussian

Covariance Matrices

- Additive noise

$$K = K_{\text{kernel}} + \sigma^2 \mathbf{1}$$

- Predictive mean and variance

$$\tilde{K} = K_{t't'} - K_{tt'}^{\top} K_{tt}^{-1} K_{tt'} \text{ and } \tilde{\mu} = K_{tt'}^{\top} K_{tt}^{-1} t$$

With Noise

$$\tilde{K} = K_{t't'} + \sigma^2 \mathbf{1} - K_{tt'}^{\top} \left(K_{tt} + \sigma^2 \mathbf{1} \right)^{-1} K_{tt'}$$

$$\text{and } \tilde{\mu} = \mu' + K_{tt'}^{\top} \left[\left(K_{tt} + \sigma^2 \mathbf{1} \right)^{-1} (y - \mu) \right]$$

Optimization

1. Convexity
   - Convex Sets
   - Convex Functions
2. Unconstrained Convex Optimization
   - First-order Methods
   - Newton's Method
3. Constrained Optimization
   - Primal and dual problems
   - KKT conditions

Copied from: Xuezhi Wang

Convex Sets

- Definition
  For $x, x' \in X$ it follows that $\lambda x + (1-\lambda)x' \in X$ for $\lambda \in [0,1]$
- Examples
  - Empty set $\emptyset$, single point $\{x_0\}$, the whole space $\mathbb{R}^n$
  - Hyperplane: $\{x \mid a^\top x = b\}$, halfspaces $\{x \mid a^\top x \leq b\}$
  - Euclidean balls: $\{x \mid ||x - x_c||_2 \leq r\}$
  - Positive semidefinite matrices: $\mathbf{S}_+^n = \{A \in \mathbf{S}^n | A \succeq 0\}$ ($\mathbf{S}^n$ is the set of symmetric $n \times n$ matrices)
- Convex Set C, D
  - Translation $\{x + b \mid x \in C\}$
  - Scaling $\{\lambda x \mid x \in C\}$
  - Affine function $\{Ax + b \mid x \in C\}$
  - Intersection $C \cap D$
  - Set sum $C + D = \{x + y \mid x \in C, y \in D\}$

Copied from: Xuezhi Wang

Gradient Descent

given a starting point $x \in \mathbf{dom} f$.
repeat
1. $\Delta x := -\nabla f(x)$
2. Choose step size t via exact or backtracking line search.
3. update. $x := x + t\Delta x$.

Until stopping criterion is satisfied.

- Key idea
  - Gradient points into descent direction
  - Locally gradient is good approximation of objective function

Copied from: Xuezhi Wang

Newton's Method

Goal: $\phi : \mathbb{R} \to \mathbb{R}$

$$\phi(x^*) = 0$$

$$x^* = ?$$

Linear Approximation (1st order Taylor approx**):**

$$\underbrace{\phi(\underbrace{x + \Delta x}_{x^*})}_{\phi(x^*) = 0} = \phi(x) + \phi'(x)\Delta x + \underbrace{o(|\Delta x|)}_{\text{NEGLIGABLE}}$$

Therefore,

$$0 \approx \phi(x) + \phi'(x)\Delta x$$

$$x^* - x = \Delta x = -\frac{\phi(x)}{\phi'(x)}$$

$$x_{k+1} = x_k - \frac{\phi(x)}{\phi'(x)}$$

Copied from: Prof Barnabas

Newton's Method

$f : \mathbb{R}^n \to \mathbb{R},\ f$ is differentiable.

$$\min_{x \in \mathbb{R}^n} f(x)$$

We need to find the roots of $\nabla f(x) = 0_n$

$$\nabla f : \mathbb{R}^n \to \mathbb{R}^n$$

Newton system: $\nabla f(x) + \nabla^2 f(x) \Delta x = 0_n$

Newton step: $\Delta x = x_{k+1} - x_k = -[\nabla^2 f(x)]^{-1} \nabla f(x)$

Iterate until convergence, or max number of iterations exceeded

Copied from: Prof Barnabas

Duality

Primal problem:

$$\min_{x \in \mathbb{R}^n} f(x)$$
$$\text{subject to } h_i(x) \leq 0, i = 1, \ldots, m$$

Lagrangian:

$$L(x, u) = f(x) + \sum_{i=1}^{m} u_i h_i(x)$$

where $u \in \mathbb{R}^m$ and $u \geq 0$.
Lagrange dual function:

$$g(u) = \min_{x \in \mathbb{R}^n} L(x, u)$$

Back to Optimization

- A typical machine learning problem has a penalty/regularizer + loss form

$$\min_w F(w) = g(w) + \frac{1}{n}\sum_{i=1}^{n} f(w; y_i, x_i),$$

$x_i, w \in \mathbb{R}^p$, $y_i \in \mathbb{R}$, both g and f are convex

- Today we only consider differentiable f, and let $g = 0$ for simplicity
- For example, let $f(w; y_i, x_i) = -\log p(y_i|x_i, w)$, we are trying to maximize the log likelihood, which is

$$\max_w \frac{1}{n}\sum_{i=1}^{n} \log p(y_i|x_i, w)$$

Gradient Descent

- choose initial $w^{(0)}$, repeat

$$w^{(t+1)} = w^{(t)} - \eta_t \cdot \nabla F(w^{(t)})$$

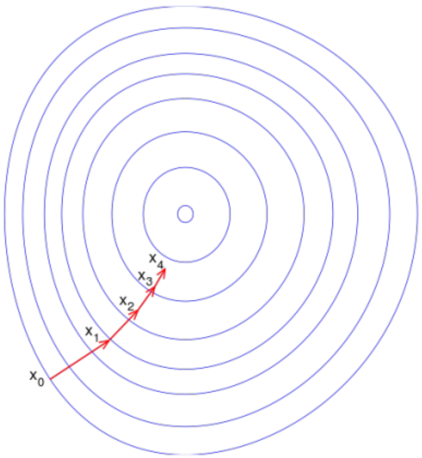
  until stop

- η_t is the learning rate, and

$$\nabla F(w^{(t)}) = \frac{1}{n} \sum_i \nabla_w f(w^{(t)}; y_i, x_i)$$

- How to stop? $\|w^{(t+1)} - w^{(t)}\| \leq \epsilon$ or $\|\nabla F(w^{(t)})\| \leq \epsilon$

Two dimensional example:



Stochastic Gradient Descent

- We name $\frac{1}{n}\sum_i f(w; y_i, x_i)$ the empirical loss, the thing we hope to minimize is the expected loss

$$f(w) = \mathbb{E}_{y_i, x_i} f(w; y_i, x_i)$$

- Suppose we receive an infinite stream of samples (y_t, x_t) from the distribution, one way to optimize the objective is

$$w^{(t+1)} = w^{(t)} - \eta_t \nabla_w f(w^{(t)}; y_t, x_t)$$

- On practice, we simulate the stream by randomly pick up (y_t, x_t) from the samples we have
- Comparing the average gradient of GD $\frac{1}{n}\sum_i \nabla_w f(w^{(t)}; y_i, x_i)$

SGD and Perceptron

- Recall Perceptron: initialize w, repeat

$$w = w + \begin{cases} y_i x_i & \text{if } y_i \langle w, x_i \rangle < 0 \\ 0 & \text{otherwise} \end{cases}$$

- Fix learning rate $\eta = 1$, let $f(w; y, x) = \max(0, -y_i \langle w, x_i \rangle)$, then

$$\nabla_w f(w; y, x) = \begin{cases} -y_i x_i & \text{if } y_i \langle w, x_i \rangle < 0 \\ 0 & \text{otherwise} \end{cases}$$

  we derive Perceptron from SGD