Unsupervised learning (part 1)
Lecture 19

David Sontag

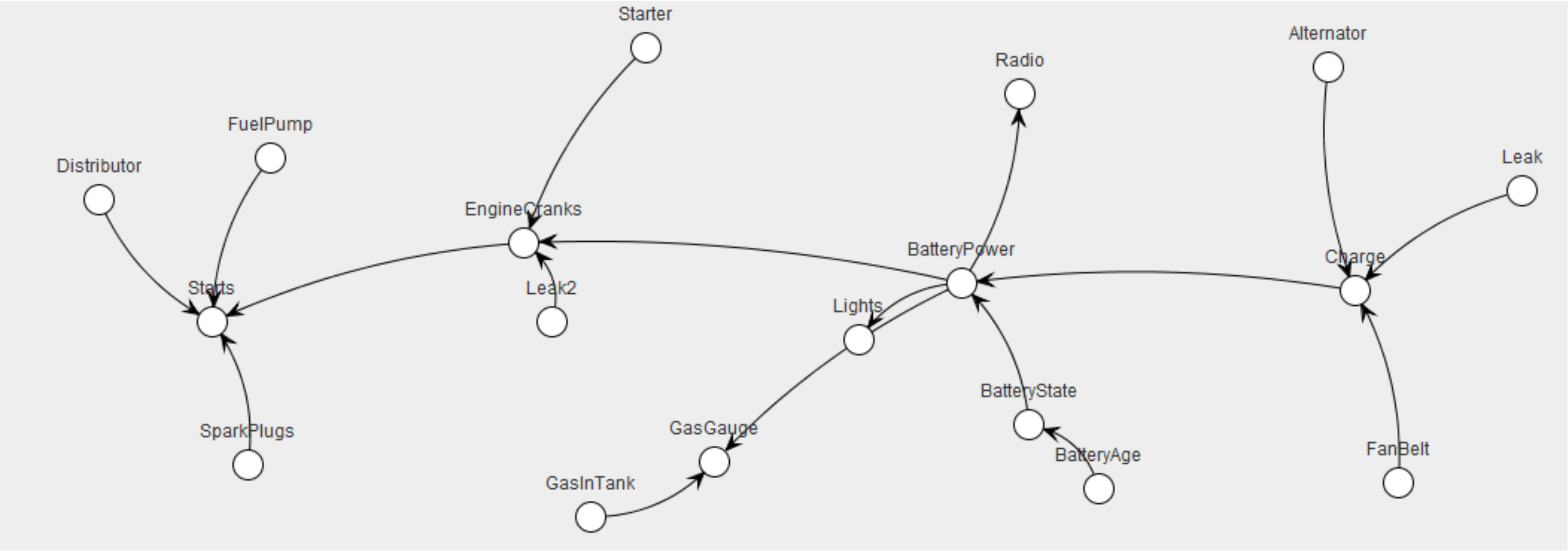
New York University

Slides adapted from Carlos Guestrin, Dan Klein, Luke Zettlemoyer, Dan Weld, Vibhav Gogate, and Andrew Moore

Bayesian networks enable use of domain knowledge

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

Will my car start this morning?



Heckerman *et al.*, Decision-Theoretic Troubleshooting, 1995

Bayesian networks enable use of domain knowledge

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

What is the differential diagnosis?

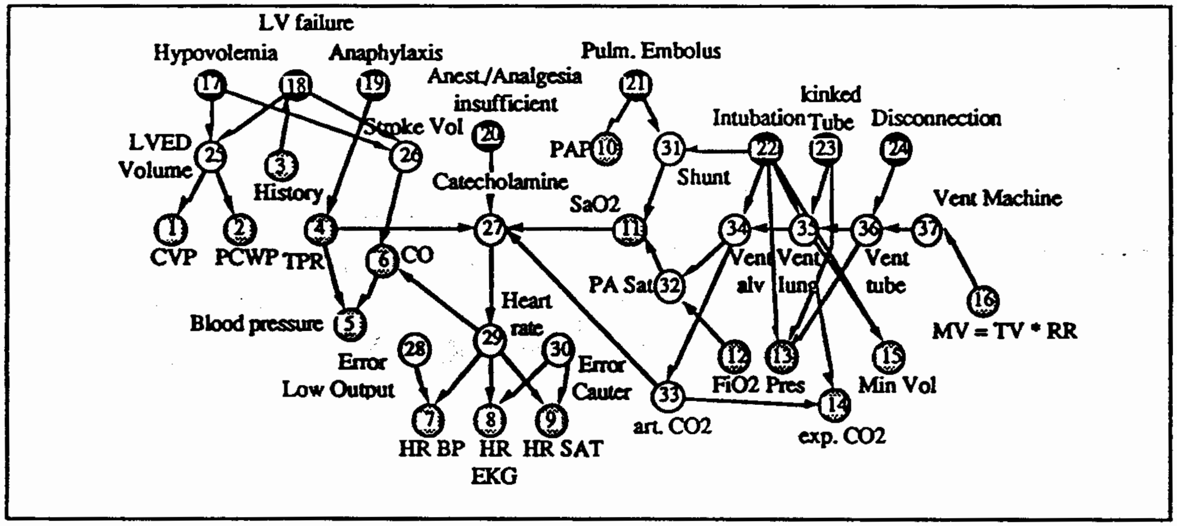
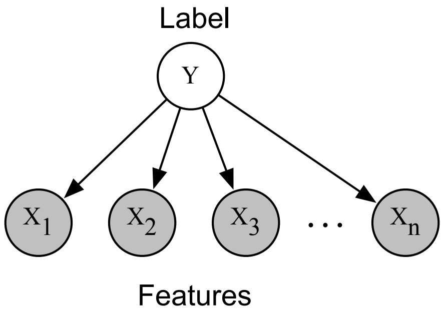


Fig. 1 The ALARM network representing causal relationships is shown with diagnostic (●), intermediate (○) and measurement (◎) nodes. CO: cardiac output, CVP: central venous pressure, LVED volume: left ventricular end-diastolic volume, LV failure: left ventricular failure, MV: minute ventilation, PA Sat: pulmonary artery oxygen saturation, PAP: pulmonary artery pressure, PCWP: pulmonary capillary wedge pressure, Pres: breathing pressure, RR: respiratory rate, TPR: total peripheral resistance, TV: tidal volume

Beinlich *et al.*, The ALARM Monitoring System, 1989

Bayesian networks are *generative models*

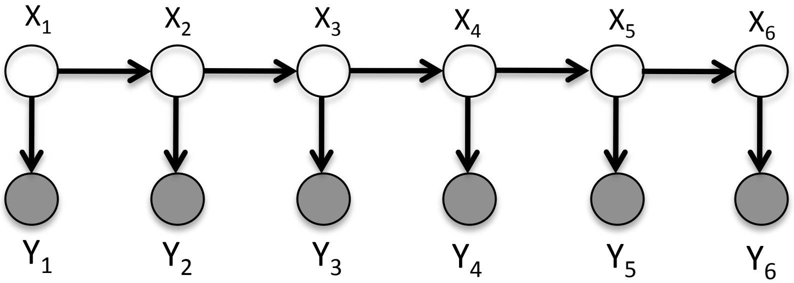
- Can sample from the joint distribution, top-down
- Suppose Y can be "spam" or "not spam", and X_i is a binary indicator of whether word *i* is present in the e-mail
- Let's try generating a few emails!

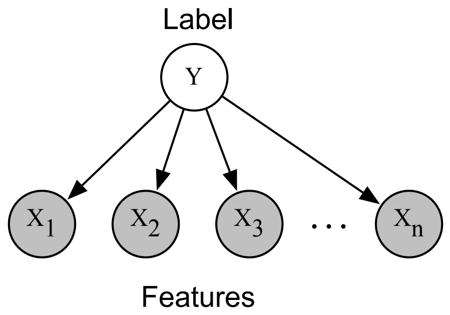


- Often helps to think about Bayesian networks as a generative model when constructing the structure and thinking about the model assumptions

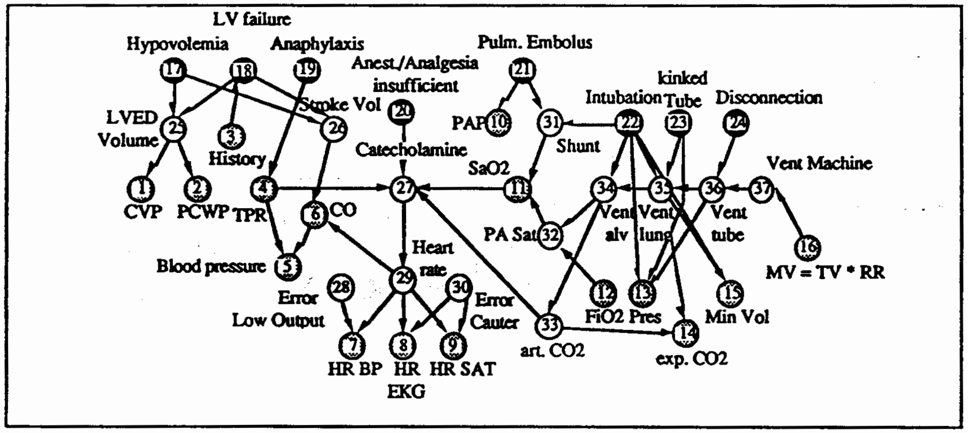
Inference in Bayesian networks

- Computing marginal probabilities in **tree** structured Bayesian networks is easy
  - The algorithm called “belief propagation” generalizes what we showed for hidden Markov models to arbitrary trees



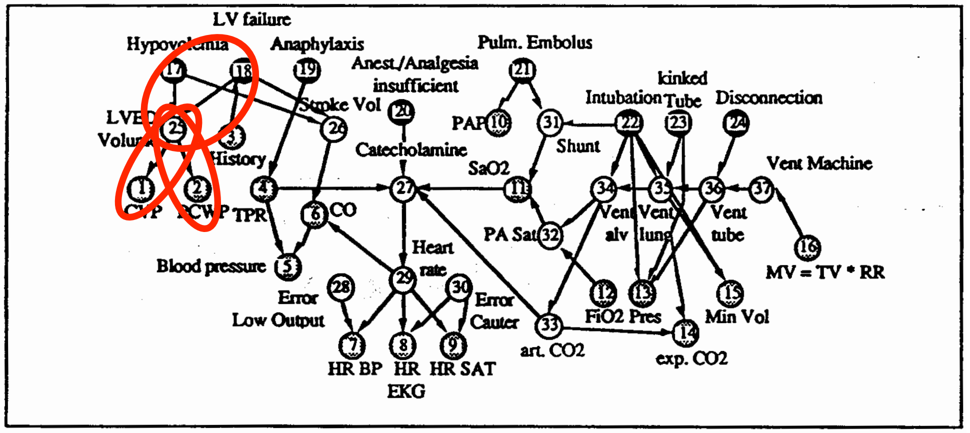


- Wait… this isn’t a tree! What can we do?



Inference in Bayesian networks

- In some cases (such as this) we can *transform* this into what is called a “junction tree”, and then run belief propagation



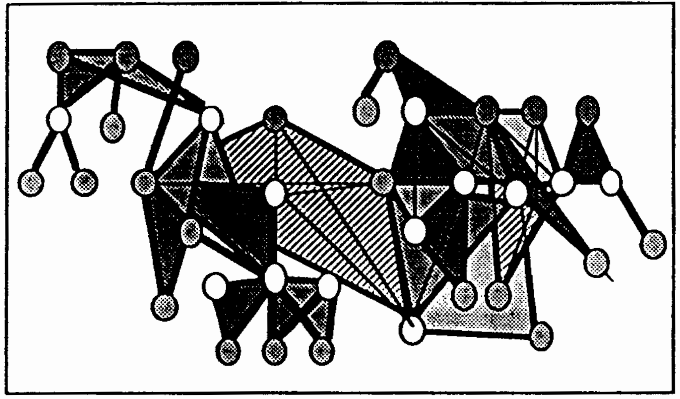
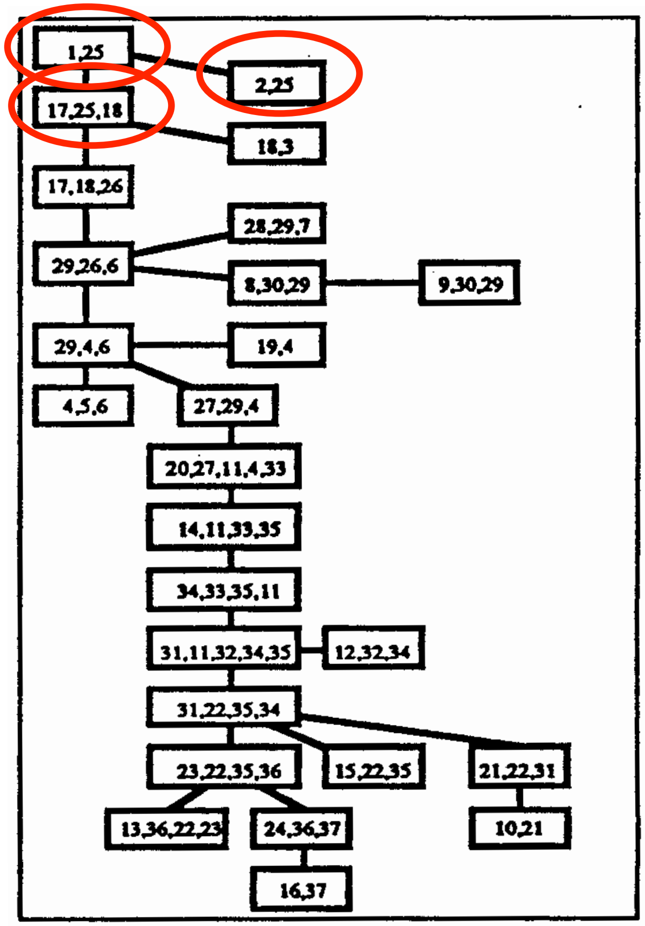


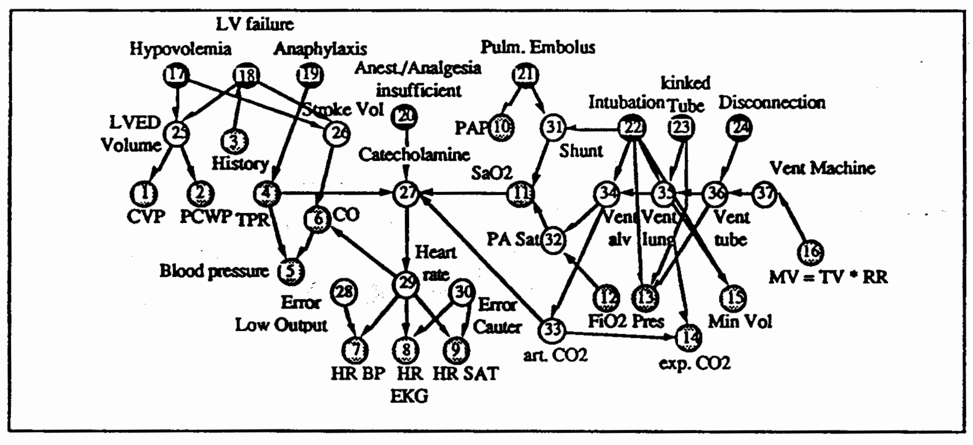
Fig. 7

Spiegelhalter's algorithm rearranges the ALARM network by triangulation and clique formation. The cliques are shaded differently to make them visible.



Approximate inference

- There is also a wealth of **approximate** inference algorithms that can be applied to Bayesian networks such as these



- ***Markov chain Monte Carlo algorithms*** repeatedly sample assignments for estimating marginals
- ***Variational inference algorithms*** (deterministic) find a simpler distribution which is "close" to the original, then compute marginals using the simpler distribution

Maximum likelihood estimation in Bayesian networks

- Suppose that we know the Bayesian network structure G
- Let $\theta_{x_i|\mathbf{x}_{pa(i)}}$ be the parameter giving the value of the CPD $p(x_i \mid \mathbf{x}_{pa(i)})$
- Maximum likelihood estimation corresponds to solving:

$$\max_{\theta} \frac{1}{M} \sum_{m=1}^{M} \log p(\mathbf{x}^M; \theta)$$

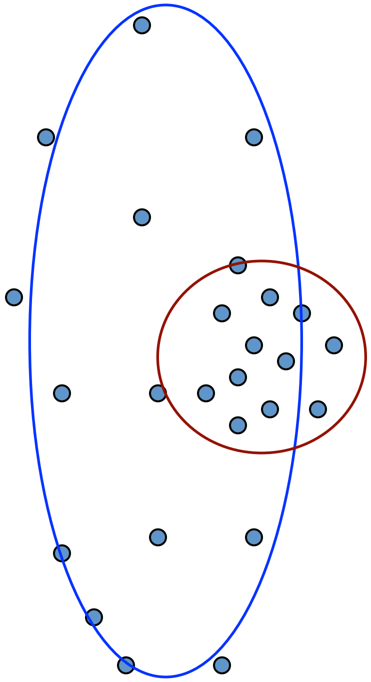
subject to the non-negativity and normalization constraints

- This is equal to:

$$\begin{aligned} \max_{\theta} \frac{1}{M} \sum_{m=1}^{M} \log p(\mathbf{x}^M; \theta) &= \max_{\theta} \frac{1}{M} \sum_{m=1}^{M} \sum_{i=1}^{N} \log p(x_i^M \mid \mathbf{x}_{pa(i)}^M; \theta) \\ &= \max_{\theta} \sum_{i=1}^{N} \frac{1}{M} \sum_{m=1}^{M} \log p(x_i^M \mid \mathbf{x}_{pa(i)}^M; \theta) \end{aligned}$$

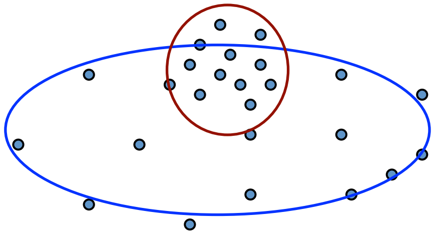
- The optimization problem decomposes into an independent optimization problem for each CPD! Has a simple closed-form solution.

Returning to clustering…



- Clusters may overlap
- Some clusters may be "wider" than others
- Can we model this explicitly?
- With what **probability** is a point from a cluster?

Probabilistic Clustering



- Try a probabilistic model!
  - allows overlaps, clusters of different size, etc.
- Can tell a *generative story* for data
  - – P(Y)P(X|Y)
- Challenge: we need to estimate model parameters without labeled Ys

Y	X_1	X_2
??	0.1	2.1
??	0.5	-1.1
??	0.0	3.0
??	-0.1	-2.0
??	0.2	1.5
…	…	…

Gaussian Mixture Models

- P(Y): There are k components
- P(X|Y): Each component generates data from a **multivariate Gaussian** with mean μ_i and covariance matrix Σ_i

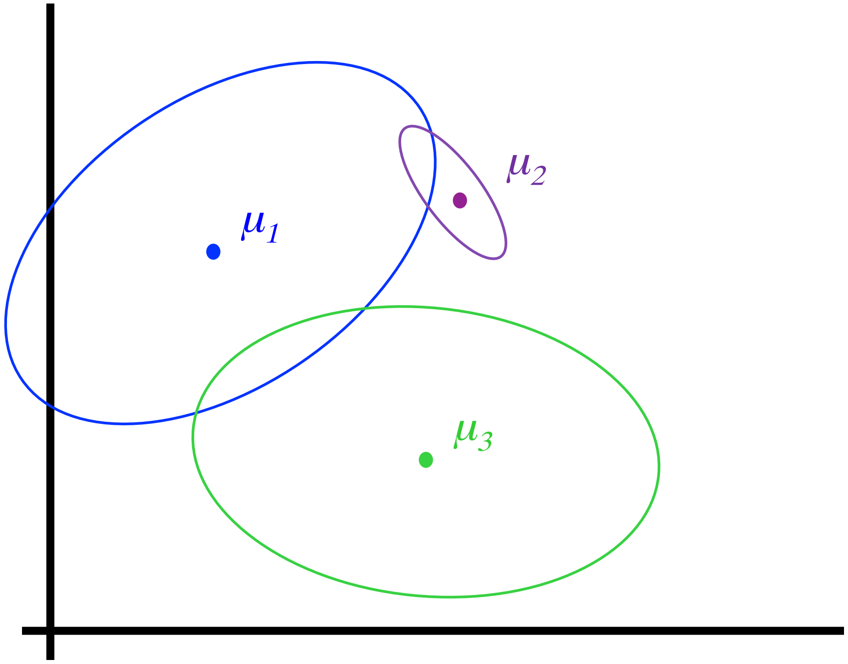
Each data point assumed to have been sampled from a ***generative process***:

1. Choose component i with probability *P(y=i)*   *[Multinomial]*
2. Generate datapoint ~ N(m_i, Σ_i)

$$P(X = \mathbf{x}_j \mid Y = i) =$$

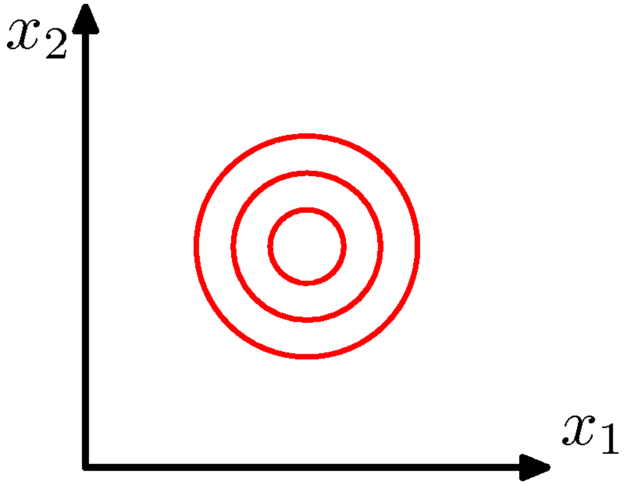
$$\frac{1}{(2\pi)^{m/2} \|\Sigma_i\|^{1/2}} \exp\left[-\frac{1}{2}\left(\mathbf{x}_j - \mu_i\right)^T \Sigma_i^{-1}\left(\mathbf{x}_j - \mu_i\right)\right]$$

By fitting this model (unsupervised learning), we can learn new insights about the data



Multivariate Gaussians

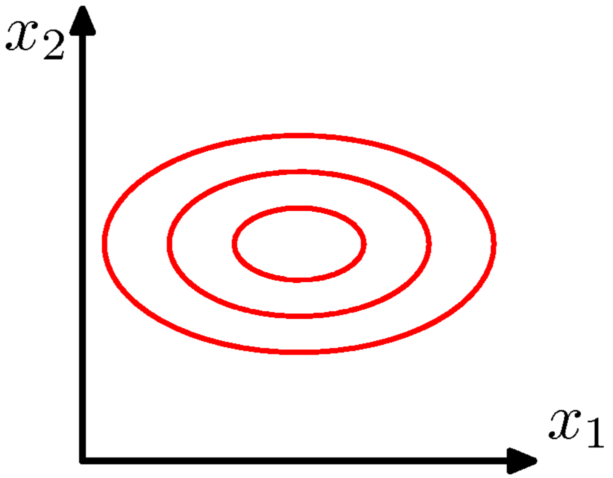
$$P(X{=}\boldsymbol{x}_j) = \frac{1}{(2\pi)^{m/2} \,||\, \Sigma \;||^{1/2}} \exp\left[-\frac{1}{2}\left(\mathbf{x}_j - \mu\right)^T \Sigma^{-1} \left(\mathbf{x}_j - \mu\right)\right]$$



Σ ∝ identity matrix

Multivariate Gaussians

$$P(X=\boldsymbol{x}_j)= \frac{1}{(2\pi)^{m/2} \|\Sigma\|^{1/2}} \exp\left[-\frac{1}{2}\left(\mathbf{x}_j - \mu\right)^T \Sigma^{-1}\left(\mathbf{x}_j - \mu\right)\right]$$

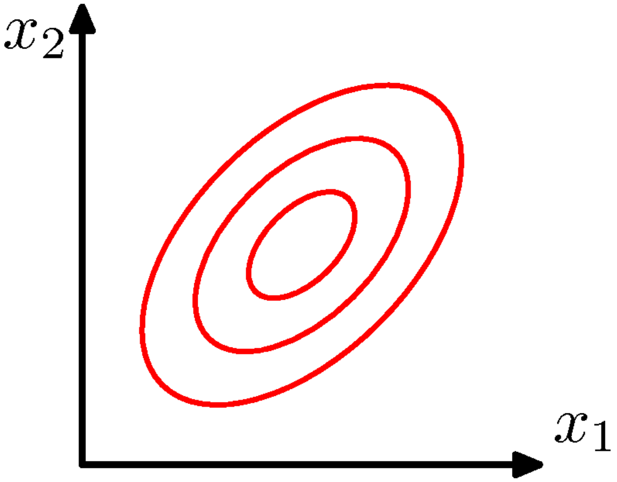


Σ = diagonal matrix

X_i are independent *ala* Gaussian NB

Multivariate Gaussians

$$P(X=\boldsymbol{x}_j)= \frac{1}{(2\pi)^{m/2} \,||\Sigma\;||^{1/2}} \exp\left[-\frac{1}{2}\left(\mathbf{x}_j-\mu\right)^T \Sigma^{-1}\left(\mathbf{x}_j-\mu\right)\right]$$

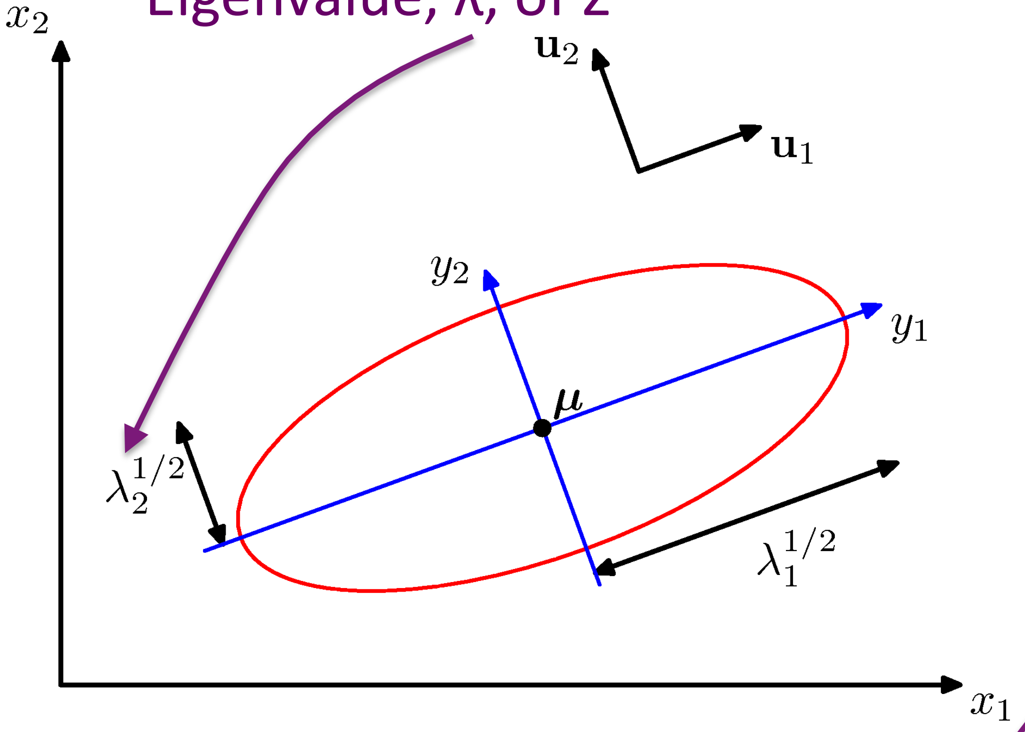


Σ = arbitrary (semidefinite) matrix:

- specifies rotation (change of basis)
- eigenvalues specify relative elongation

Multivariate Gaussians

Eigenvalue, λ, of Σ

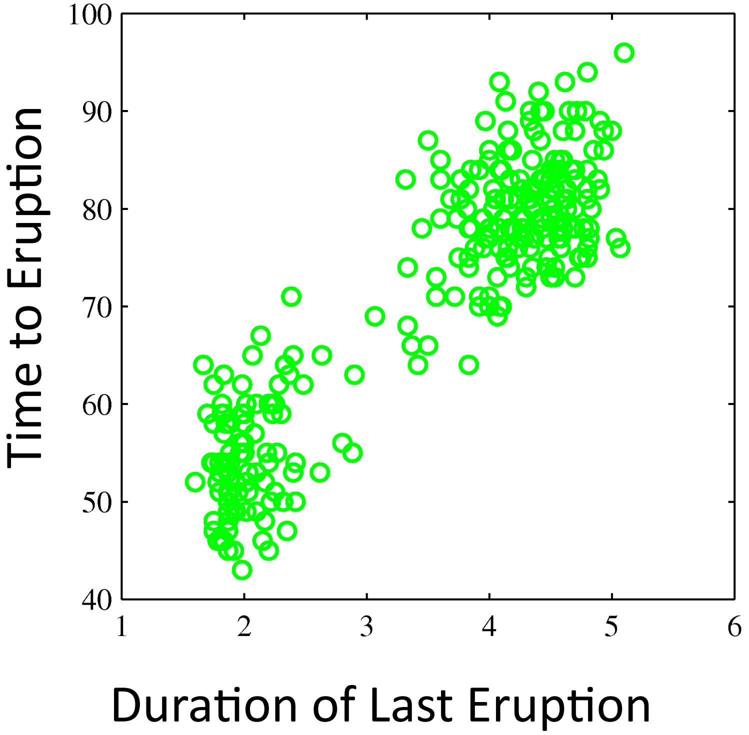


Covariance matrix, Σ, = degree to which x_i vary together

$$P(X=\boldsymbol{x}_j)= \frac{1}{(2\pi)^{m/2} \, || \Sigma \; ||^{1/2}} \exp\left[-\frac{1}{2}\left(\mathbf{x}_j-\mu\right)^T \Sigma^{-1}\left(\mathbf{x}_j-\mu\right)\right]$$

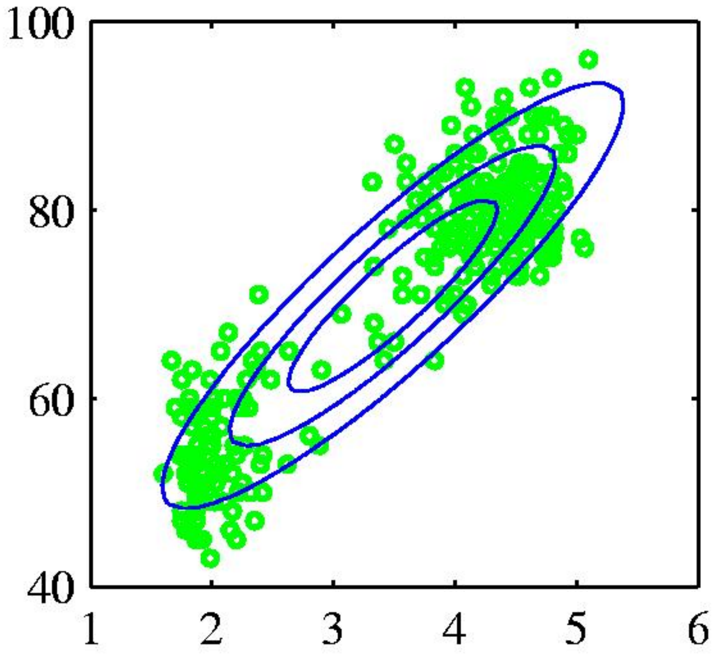
Modelling eruption of geysers

Old Faithful Data Set

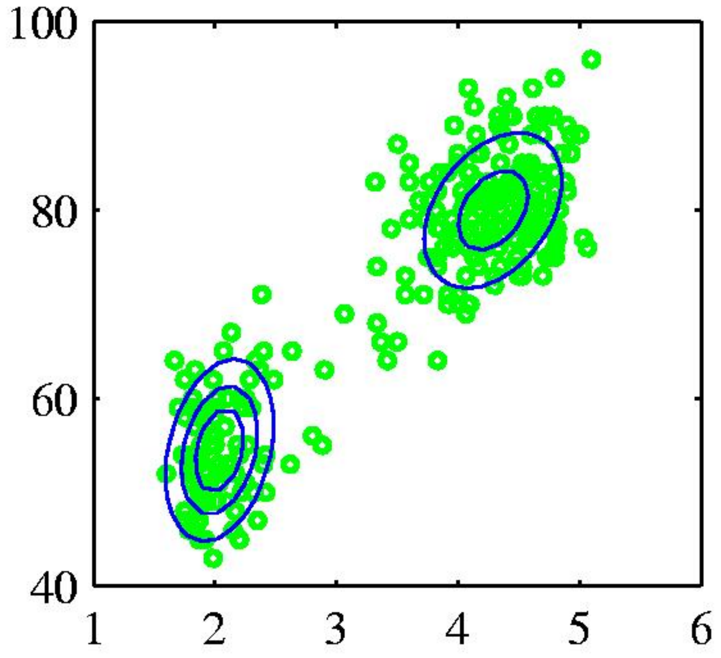


Modelling eruption of geysers

Old Faithful Data Set



Single Gaussian



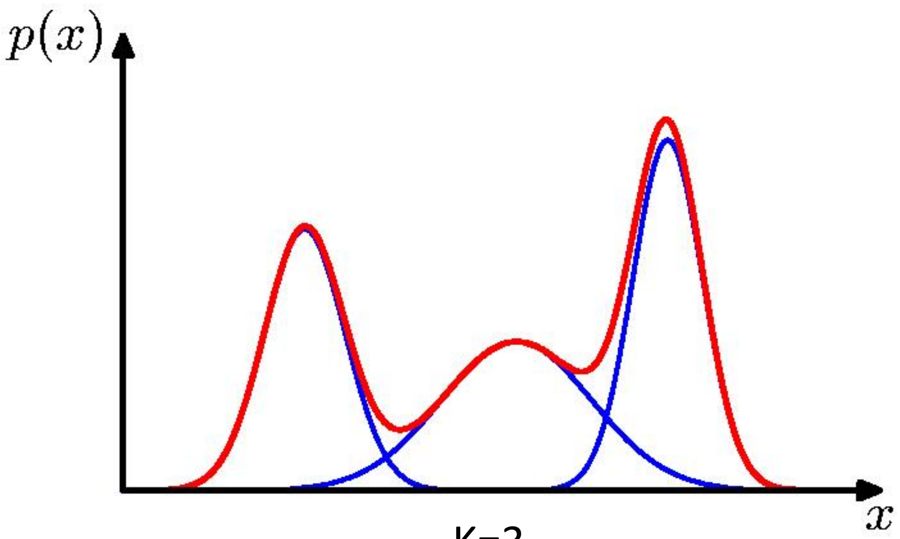
Mixture of two Gaussians

Marginal distribution for mixtures of Gaussians

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \underbrace{\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}_{\text{Component}}$$

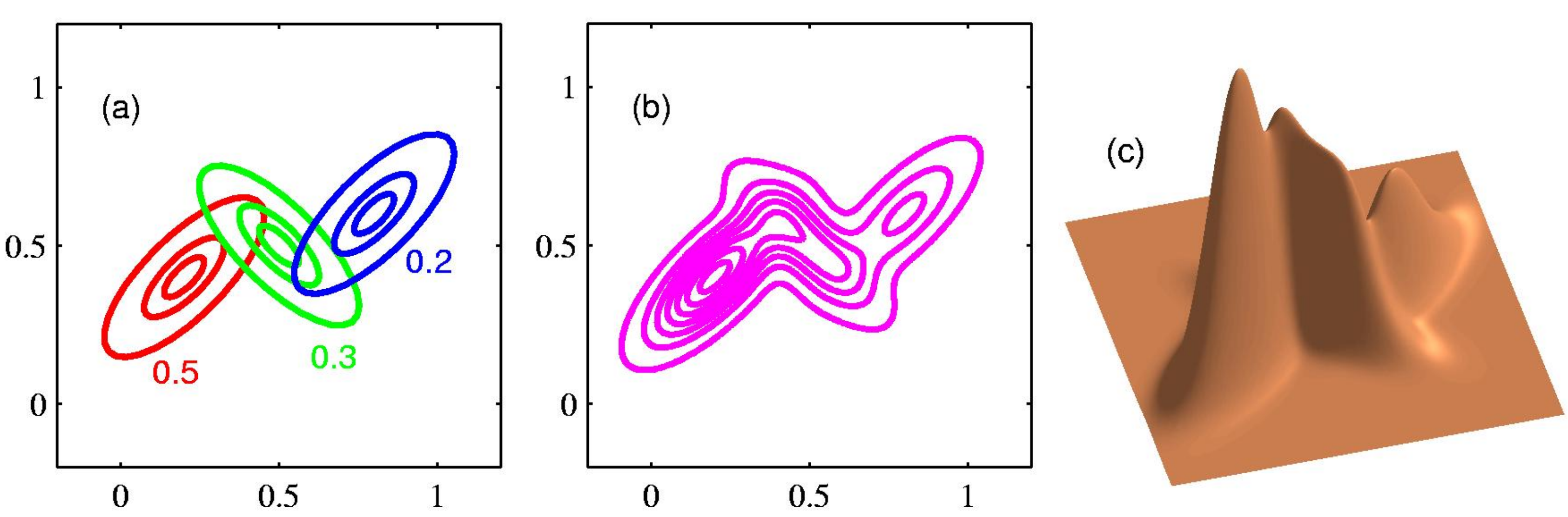
Mixing coefficient

$$\forall k : \pi_k \geqslant 0 \qquad \sum_{k=1}^{K} \pi_k = 1$$

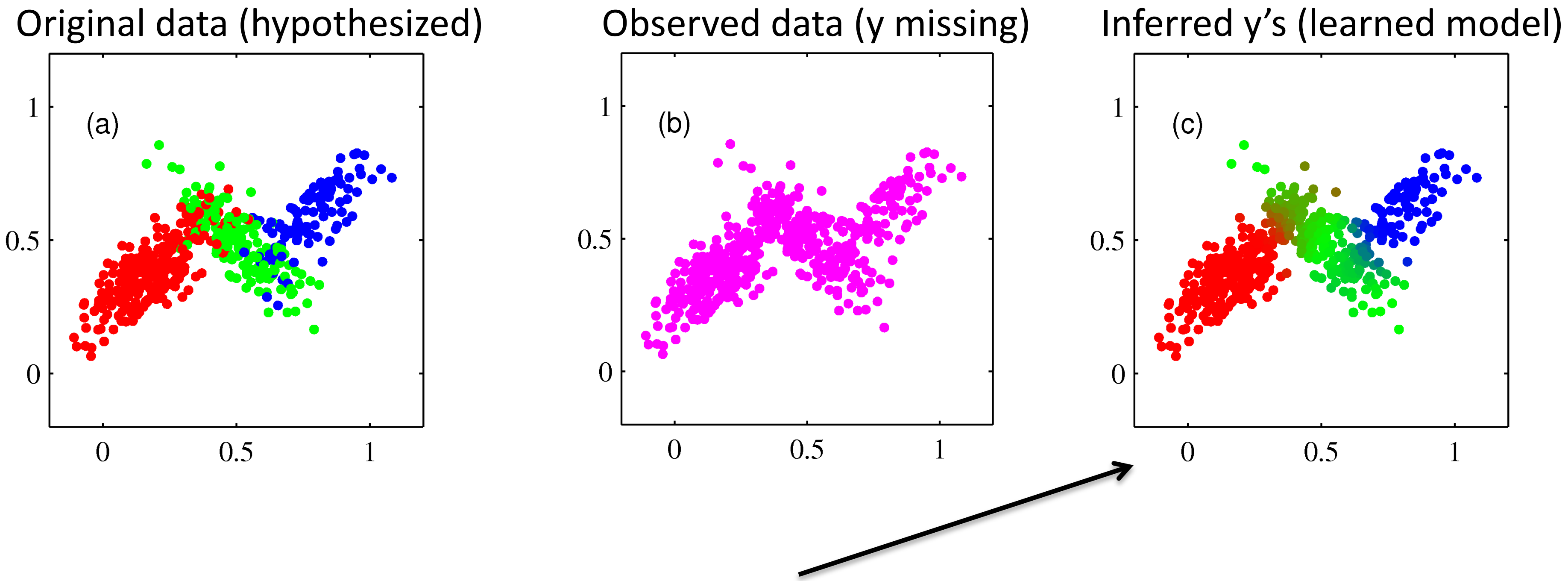


K=3

Marginal distribution for mixtures of Gaussians



Learning mixtures of Gaussians



Original data (hypothesized)

Observed data (y missing)

Inferred y’s (learned model)

Shown is the *posterior probability* that a point was generated from ith Gaussian: $\Pr(Y = i \mid x)$

ML estimation in **supervised** setting

- Univariate Gaussian

$$\mu_{MLE} = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \sigma^2_{MLE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \widehat{\mu})^2$$

- ***Mixture*** of ***Multi***variate Gaussians

ML estimate for each of the Multivariate Gaussians is given by:

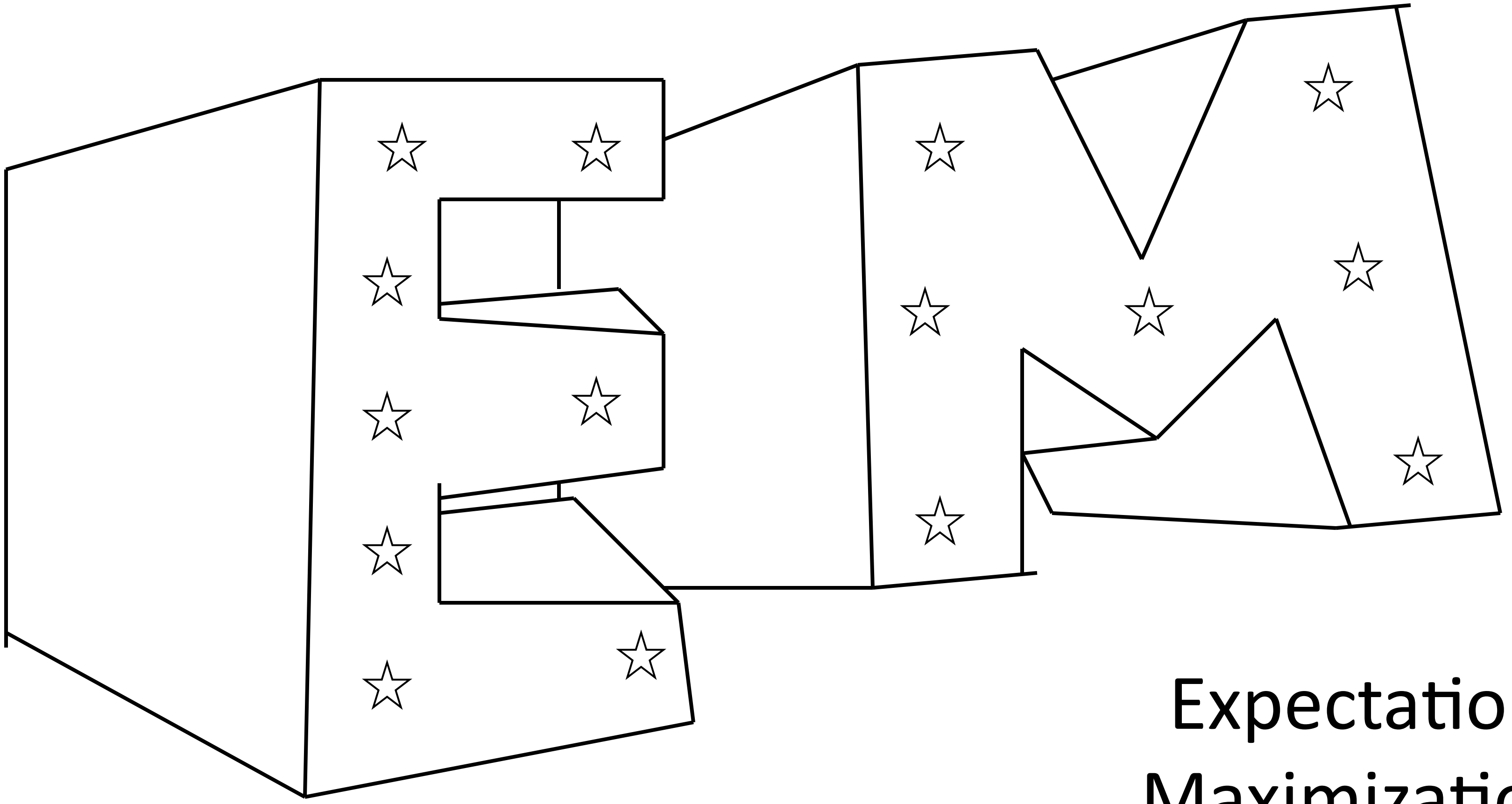
$$\mu^k_{ML} = \frac{1}{n}\sum_{j=1}^{n} x_n \qquad \Sigma^k_{ML} = \frac{1}{n}\sum_{j=1}^{n}\left(\mathbf{x}_j - \mu^k_{ML}\right)\left(\mathbf{x}_j - \mu^k_{ML}\right)^T$$

Just sums over *x* generated from the *k*'th Gaussian

What about with unobserved data?

- Maximize ***marginal likelihood***:
  - $\text{argmax}_{\theta} \prod_j P(x_j) = \text{argmax} \prod_j \sum_{k=1}^{K} P(Y_j{=}k, x_j)$

- Almost always a hard problem!
  - Usually no closed form solution
  - Even when lgP(X,Y) is convex, lgP(X) generally isn't…
  - Many local optima



Expectation
Maximization

1977: Dempster, Laird, & Rubin

The EM Algorithm

- A clever method for maximizing marginal likelihood:
  - $\text{argmax}_\theta \prod_j P(x_j) = \text{argmax}_\theta \prod_j \sum_{k=1}^{K} P(Y_j=k, x_j)$
  - Based on coordinate descent. Easy to implement (eg, no line search, learning rates, etc.)
- Alternate between two steps:
  - Compute an expectation
  - Compute a maximization
- Not magic: ***still optimizing a non-convex function with lots of local optima***
  - The computations are just easier (often, significantly so)

EM: Two Easy Steps

Objective: $\text{argmax}_\theta \lg \prod_j \sum_{k=1}^{K} P(Y_j=k, x_j ; \theta) = \sum_j \lg \sum_{k=1}^{K} P(Y_j=k, x_j; \theta)$

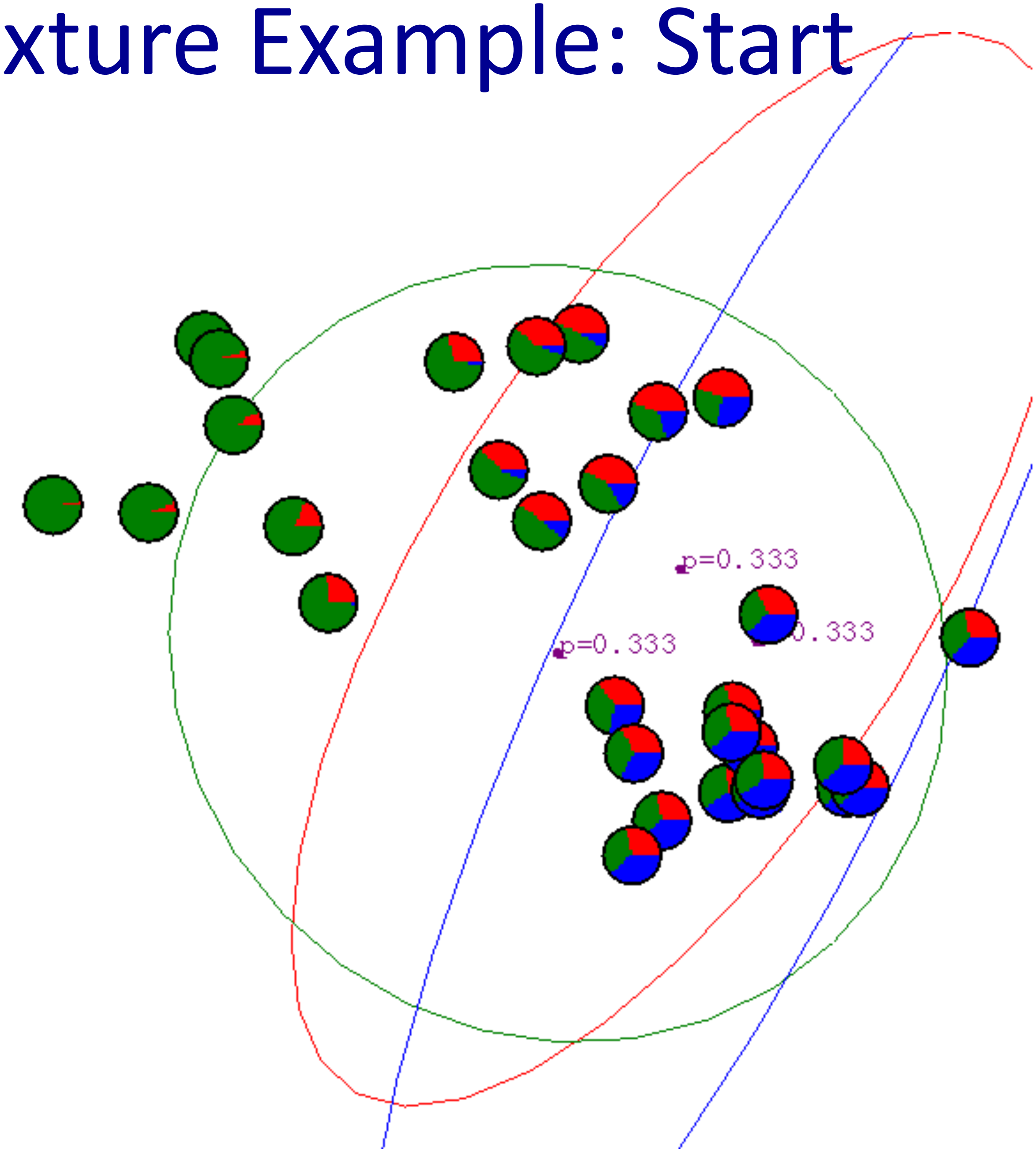
Data: $\{x_j \mid j=1 .. n\}$

- **E-step**: Compute expectations to "fill in" missing y values according to current parameters, θ
  - For all examples j and values k for Y_j, compute: $P(Y_j=k \mid x_j; \theta)$

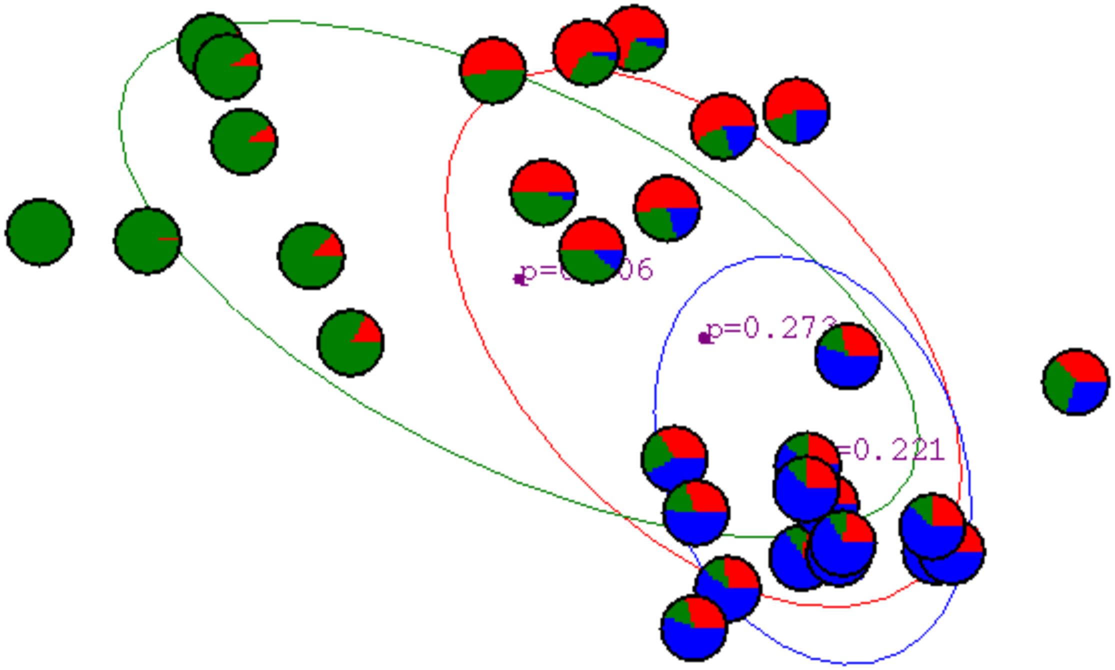
- **M-step**: Re-estimate the parameters with "weighted" MLE estimates
  - Set $\theta^{new} = \text{argmax}_\theta \sum_j \sum_k P(Y_j=k \mid x_j ;\theta^{old}) \log P(Y_j=k, x_j ; \theta)$

Particularly useful when the E and M steps have closed form solutions

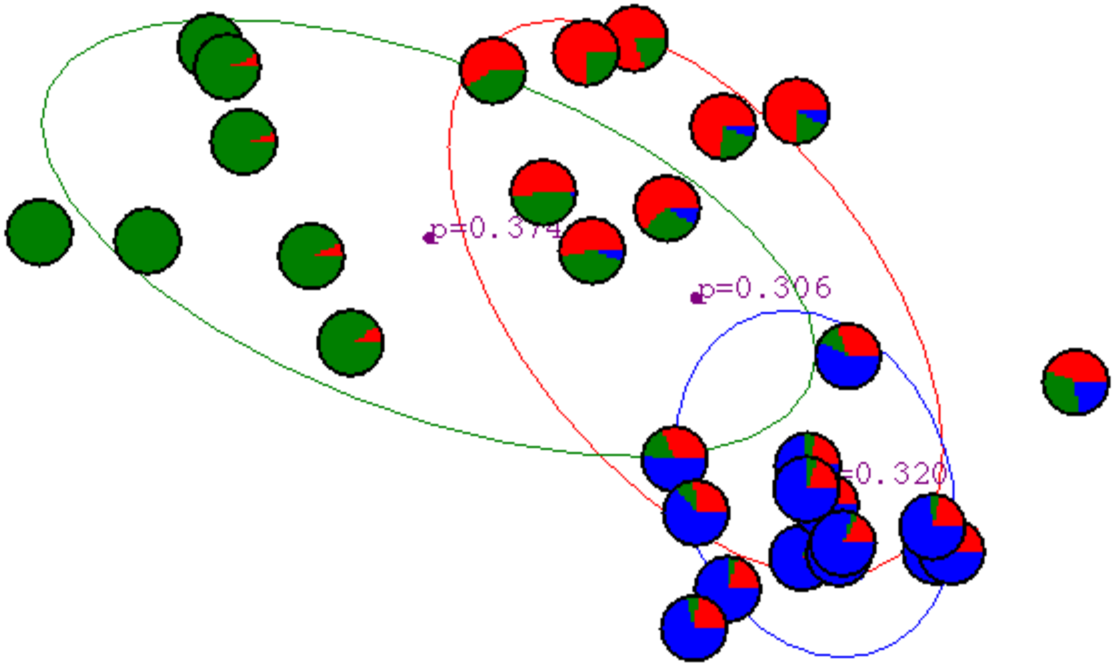
Gaussian Mixture Example: Start



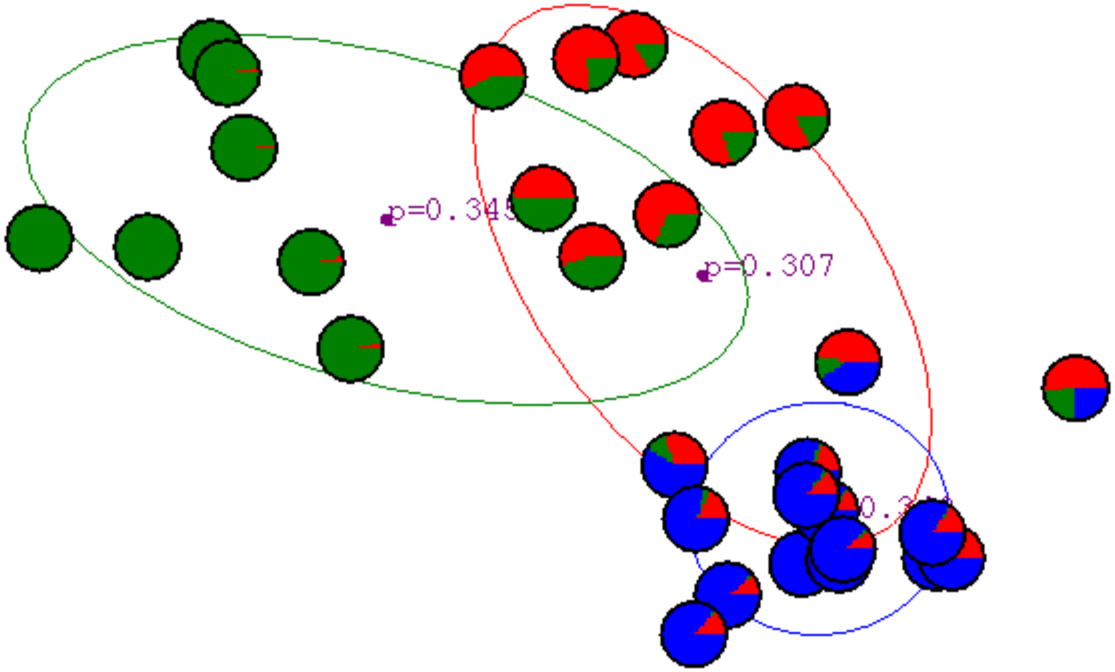
After first iteration



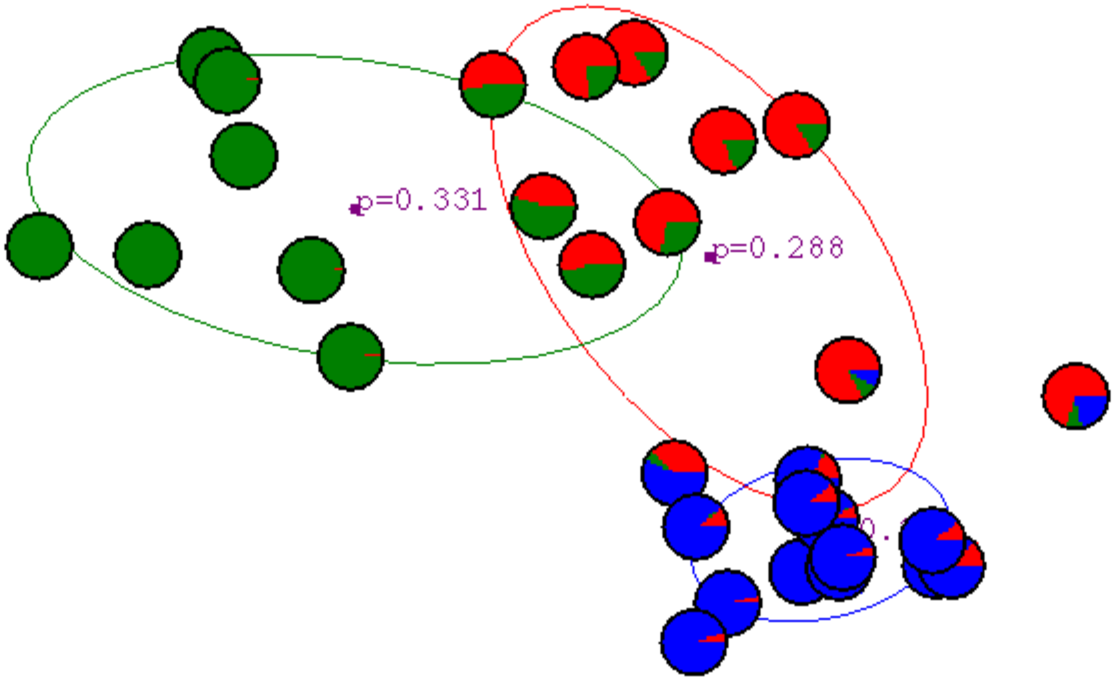
After 2nd iteration



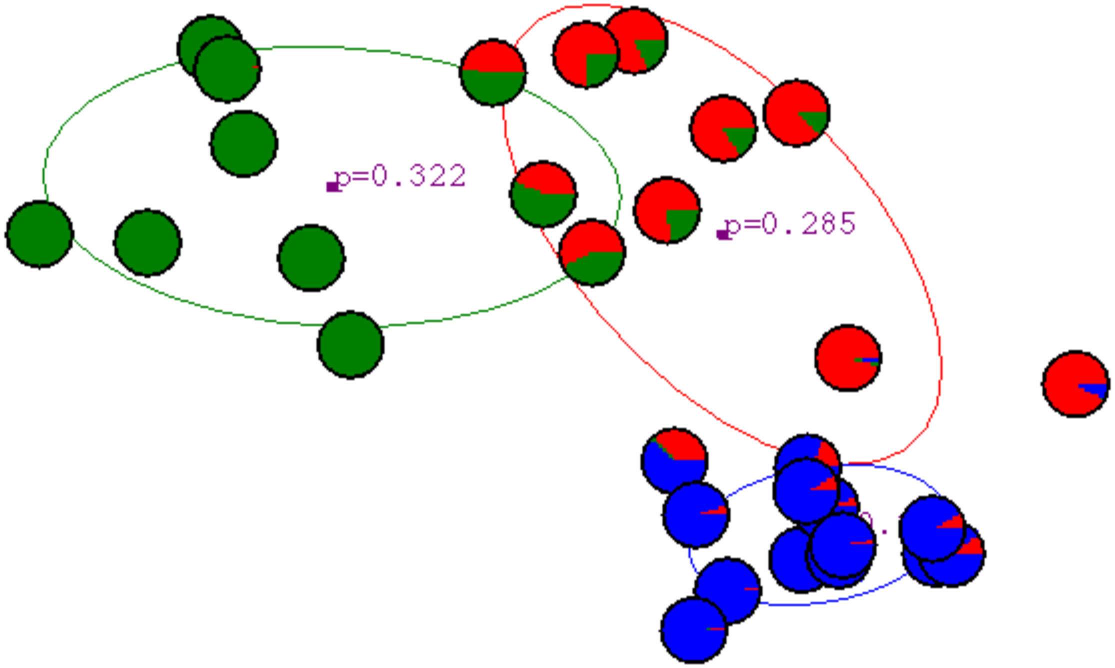
After 3rd iteration



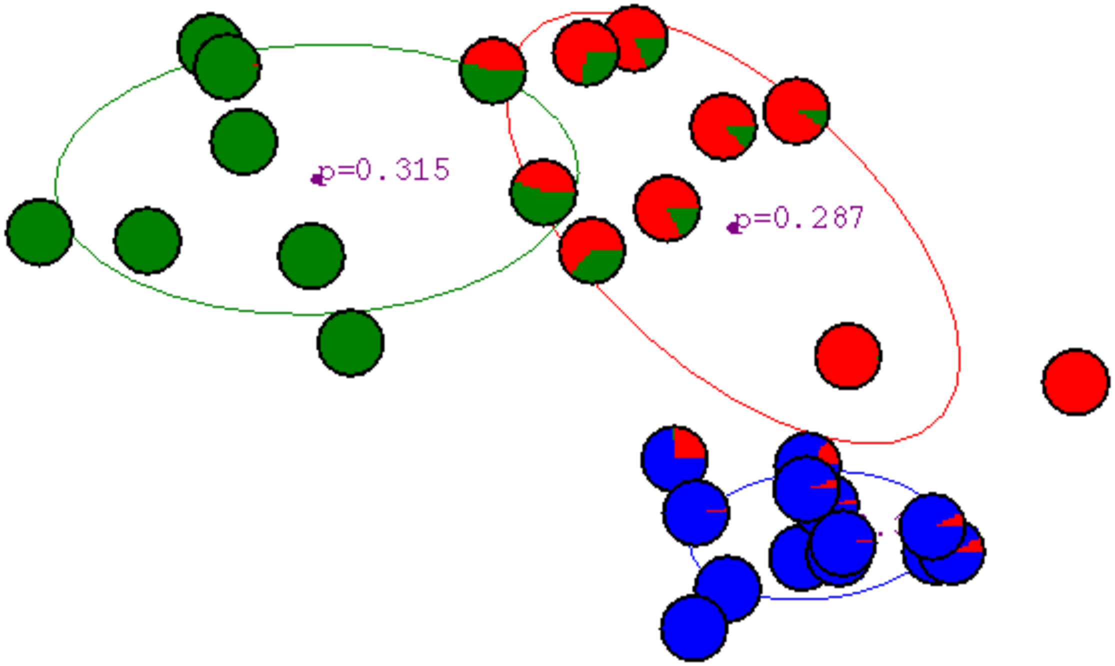
After 4th iteration



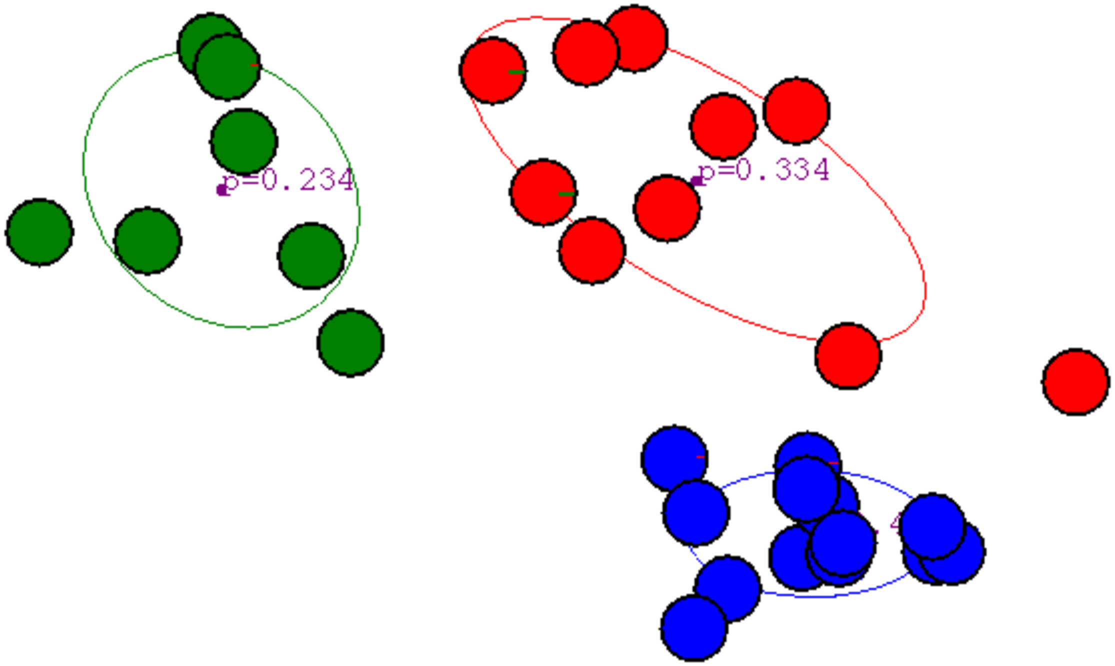
After 5th iteration



After 6th iteration



After 20th iteration



EM for GMMs: only learning means (1D)

Iterate: On the *t*'th iteration let our estimates be

$$\lambda_t = \{ \mu_1^{(t)}, \mu_2^{(t)} \ldots \mu_K^{(t)} \}$$

E-step

Compute "expected" classes of all datapoints

$$P\left(Y_j = k \middle| x_j, \mu_1 \ldots \mu_K\right) \propto \exp\left(-\frac{1}{2\sigma^2}(x_j - \mu_k)^2\right) \mathrm{P}\left(Y_j = k\right)$$

M-step

Compute most likely new **μ**s given class expectations

$$\mu_k = \frac{\sum_{j=1}^{m} P\left(Y_j = k \middle| x_j\right) x_j}{\sum_{j=1}^{m} P\left(Y_j = k \middle| x_j\right)}$$

What if we do hard assignments?

Iterate: On the *t*'th iteration let our estimates be

$$\lambda_t = \{ \mu_1^{(t)}, \mu_2^{(t)} \ldots \mu_K^{(t)} \}$$

E-step

Compute "expected" classes of all datapoints

$$P\left(Y_j = k \middle| x_j, \mu_1 \ldots \mu_K\right) \propto \exp\left(-\frac{1}{2\sigma^2}(x_j - \mu_k)^2\right) \xcancel{\mathrm{P}\left(Y_j = k\right)}$$

M-step

Compute most likely new **μ**s given class expectations

δ represents hard assignment to "most likely" or nearest cluster

$$\mu_k = \xcancel{\frac{\sum_{j=1}^{m} P\left(Y_j = k \middle| x_j\right) x_j}{\sum_{j=1}^{m} P\left(Y_j = k \middle| x_j\right)}} \qquad \mu_k = \frac{\sum_{j=1}^{m} \delta\left(Y_j = k, x_j\right) x_j}{\sum_{j=1}^{m} \delta\left(Y_j = k, x_j\right)}$$

Equivalent to k-means clustering algorithm!!!

E.M. for General GMMs

Iterate: On the *t*'th iteration let our estimates be

$$\lambda_t = \{ \mu_1^{(t)}, \mu_2^{(t)} \ldots \mu_K^{(t)}, \Sigma_1^{(t)}, \Sigma_2^{(t)} \ldots \Sigma_K^{(t)}, p_1^{(t)}, p_2^{(t)} \ldots p_K^{(t)} \}$$

$p_k^{(t)}$ is shorthand for estimate of *P(y=k)* on t'th iteration

E-step

Compute "expected" classes of all datapoints for each class

$$\mathrm{P}\left(Y_j = k \middle| x_j; \lambda_t\right) \propto p_k^{(t)} \mathrm{p}\left(x_j; \mu_k^{(t)}, \Sigma_k^{(t)}\right)$$

Evaluate probability of a multivariate a Gaussian at x_j

M-step

Compute weighted MLE for **μ** given expected classes above

$$\mu_k^{(t+1)} = \frac{\sum_j \mathrm{P}\left(Y_j = k \middle| x_j; \lambda_t\right) x_j}{\sum_j \mathrm{P}\left(Y_j = k \middle| x_j; \lambda_t\right)} \qquad \Sigma_k^{(t+1)} = \frac{\sum_j \mathrm{P}\left(Y_j = k \middle| x_j; \lambda_t\right) \left[x_j - \mu_k^{(t+1)}\right]\left[x_j - \mu_k^{(t+1)}\right]^T}{\sum_j \mathrm{P}\left(Y_j = k \middle| x_j; \lambda_t\right)}$$

$$p_k^{(t+1)} = \frac{\sum_j \mathrm{P}\left(Y_j = k \middle| x_j; \lambda_t\right)}{m}$$

m = #training examples

The general learning problem with missing data

- Marginal likelihood: **X** is observed,

  Z (e.g. the class labels **Y**) is missing:

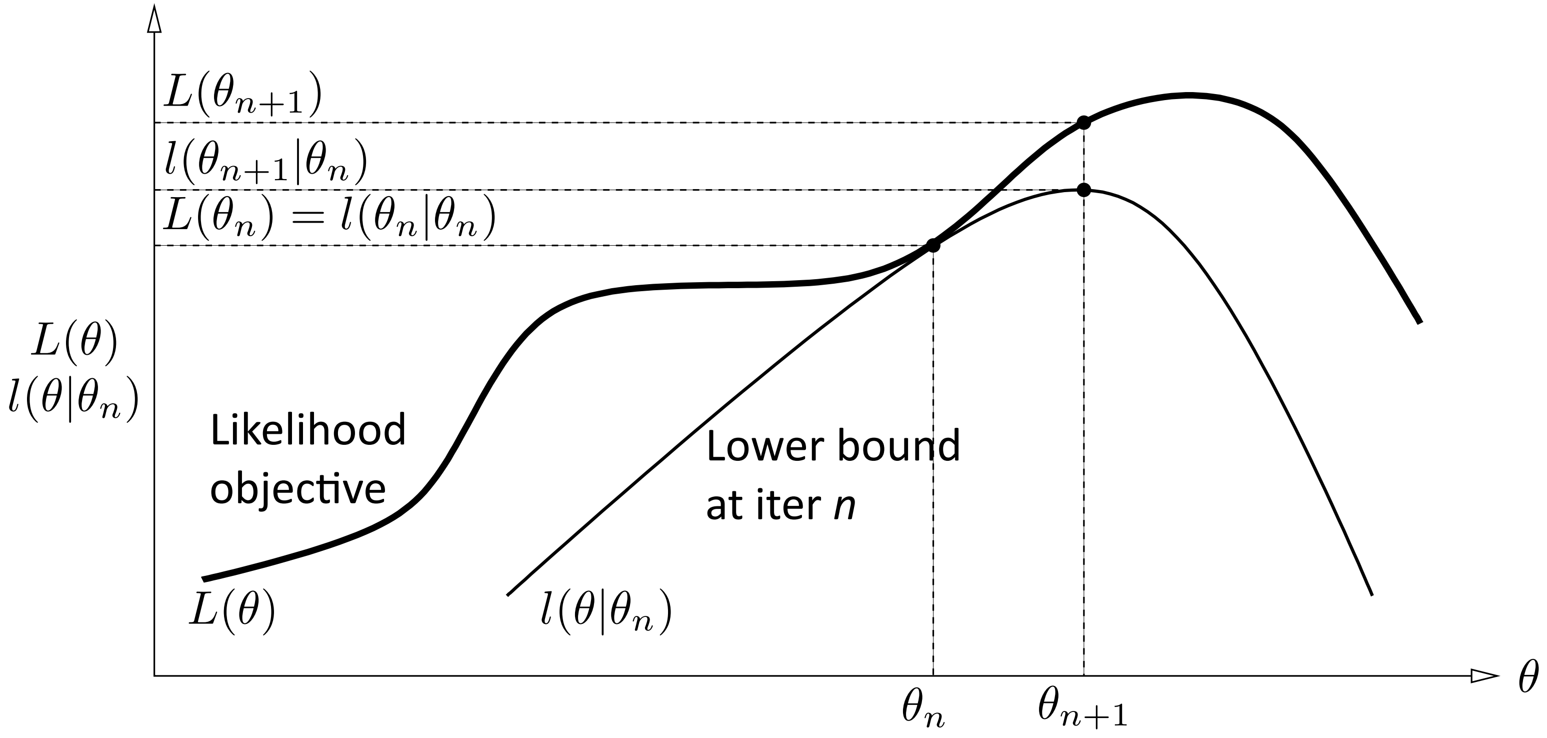
$$\begin{aligned}\ell(\theta : \mathcal{D}) &= \log \prod_{j=1}^{m} P(\mathbf{x}_j \mid \theta)\\ &= \sum_{j=1}^{m} \log P(\mathbf{x}_j \mid \theta)\\ &= \sum_{j=1}^{m} \log \sum_{\mathbf{z}} P(\mathbf{x}_j, \mathbf{z} \mid \theta)\end{aligned}$$

- Objective: Find argmax$_\theta$ l(θ:Data)
- **Assuming hidden variables are *missing completely at random* (otherwise, we should explicitly model *why* the values are missing)**

Properties of EM

- One can prove that:
  - EM converges to a local maxima
  - Each iteration improves the log-likelihood
- How? (Same as k-means)
  - Likelihood objective instead of k-means objective
  - M-step can never decrease likelihood

EM pictorially



(Figure from tutorial by Sean Borman)

What you should know

- Mixture of Gaussians
- EM for mixture of Gaussians:
  - How to learn maximum likelihood parameters in the case of unlabeled data
  - Relation to K-means
    - Two step algorithm, just like K-means
    - Hard / soft clustering
    - Probabilistic model
- Remember, EM can get stuck in local minima,
  - And empirically it ***DOES***