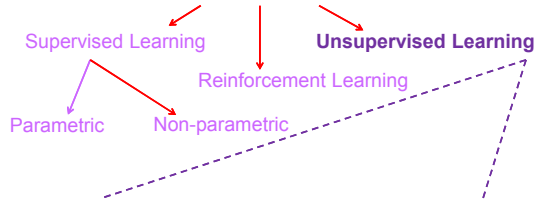
CSE 446: Expectation Maximization (EM)
Winter 2012

Daniel Weld

Slides adapted from Carlos Guestrin, Dan Klein & Luke Zettlemoyer

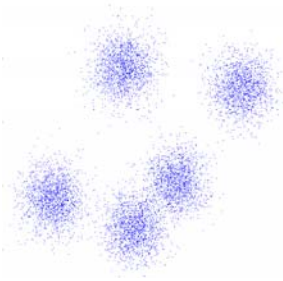
Machine Learning



Fri K-means & Agglomerative Clustering
Mon **Expectation Maximization (EM)**
Wed Principle Component Analysis (PCA)

K-Means

- An iterative clustering algorithm
  - Pick K random points as cluster centers (means)
  - Alternate:
    - Assign data instances to closest mean
    - Assign each mean to the average of its assigned points
  - Stop when no points' assignments change



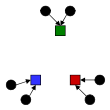
K-Means as Optimization

- Consider the total distance to the means:

$$\phi(\{x_i\},\{a_i\},\{c_k\}) = \sum_i \text{dist}(x_i, c_{a_i})$$

points, assignments, means

- Two stages each iteration:
  - Update assignments: fix means c, change assignments a
  - Update means: fix assignments a, change means c



- *Coordinate gradient ascent* on Φ
- Will it converge?
  - Yes!, change from either update can only decrease Φ

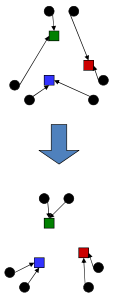
Phase I: Update Assignments (Expectation)

- For each point, re-assign to closest mean:

$$a_i = \arg\min_k \text{dist}(x_i, c_k)$$

- Can only decrease total distance phi!

$$\phi(\{x_i\},\{a_i\},\{c_k\}) = \sum_i \text{dist}(x_i, c_{a_i})$$

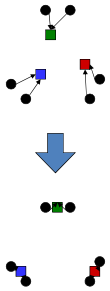


Phase II: Update Means (Maximization)

- Move each mean to the average of its assigned points:

$$c_k = \frac{1}{|\{i : a_i = k\}|} \sum_{i:a_i=k} x_i$$

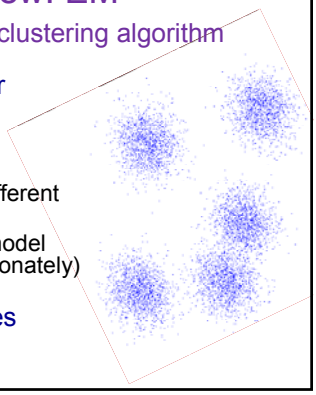
- Also can only decrease total distance… (Why?)
- Fun fact: the point y with minimum squared Euclidean distance to a set of points {x} is their mean



Preview: EM

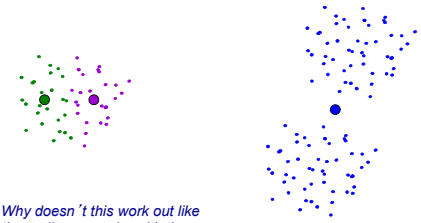
another iterative clustering algorithm

- Pick K random cluster models
- Alternate:
  - Assign data instances ***proportionately*** to different models
  - Revise each cluster model based on its (proportionately) assigned points
- Stop when no changes



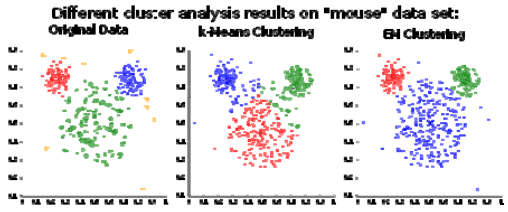
K-Means Getting Stuck

A local optimum:

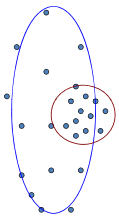


Why doesn't this work out like the earlier example, with the purple taking over half the blue?

Preference for Equally Sized Clusters

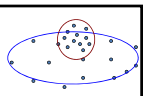


The Evils of "**Hard Assignments**"?



- Clusters may overlap
- Some clusters may be "wider" than others
- Distances can be deceiving!

Probabilistic Clustering



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

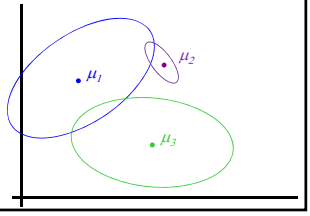
Y	X_1	X_2
??	0.1	2.1
??	0.5	-1.1
??	0.0	3.0
??	-0.1	-2.0
??	0.2	1.5
…	…	…

The General GMM assumption

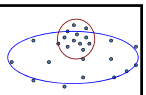
- P(Y): There are k components
- P(X|Y): Each component generates data from a **multivariate Gaussian** with mean μ_i and covariance matrix Σ_i

Each data point is sampled from a ***generative process***:

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



What Model Should We Use?



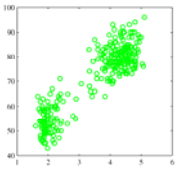
- Depends on X!
- Here, maybe Gaussian Naïve Bayes?
  - Multinomial over clusters Y
  - Gaussian over each X_i given Y

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
	...	...

$$p(Y_i = y_k) = \theta_k$$

$$P(X_i = x \mid Y = y_k) = \frac{1}{\sigma_{ik}\sqrt{2\pi}}\, e^{\frac{-(x-\mu_{ik})^2}{2\sigma_{ik}^2}}$$

Could we make fewer assumptions?

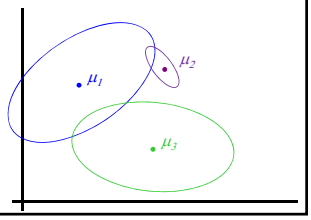


- What if the X_i co-vary?
- What if there are multiple peaks?
- Gaussian Mixture Models!
  - P(Y) still multinomial
  - P(**X**|Y) is a multivariate Gaussian dist'n

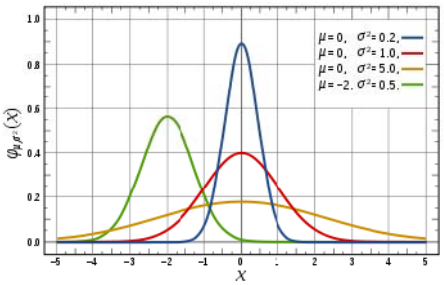
$$P(X=\mathbf{x}_j \mid Y=i) = \frac{1}{(2\pi)^{m/2}\,\|\Sigma_i\|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}_j-\mu_i)^T \Sigma_i^{-1}(\mathbf{x}_j-\mu_i)\right]$$

The General GMM assumption

1. What's a ***Multivariate*** Gaussian?
2. What's a ***Mixture Model***?



Review: Gaussians



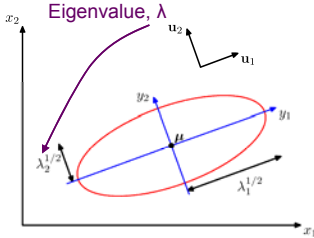
$$P(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

Learning Gaussian Parameters (given fully-observable data)

$$\hat{\mu}_{MLE} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

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

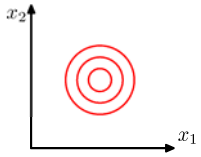
Multivariate Gaussians



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

18

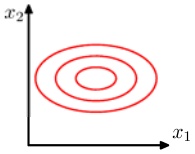
Multivariate Gaussians



$\Sigma \propto$ identity matrix

19

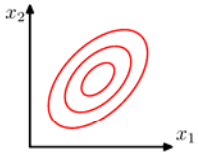
Multivariate Gaussians



Σ = diagonal matrix
X_i are independent *ala* Gaussian NB

20

Multivariate Gaussians

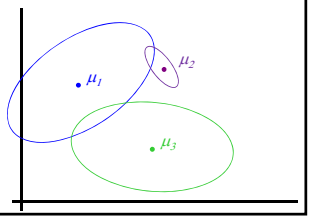


Σ = arbitrary (semidefinite) matrix
specifies rotation (change of basis)
eigenvalues specify relative elongation

21

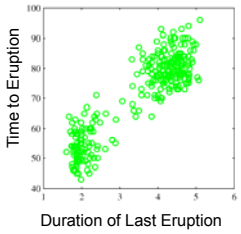
The General GMM assumption

1. What's a Multivariate Gaussian?
2. What's a ***Mixture Model***?



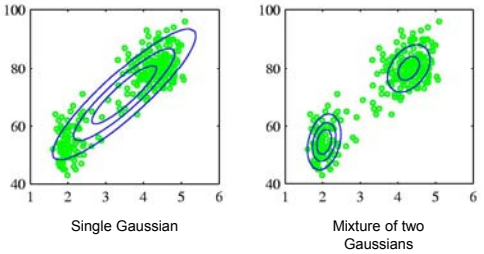
Mixtures of Gaussians (1)

Old Faithful Data Set



Mixtures of Gaussians (1)

Old Faithful Data Set



Mixtures of Gaussians (2)

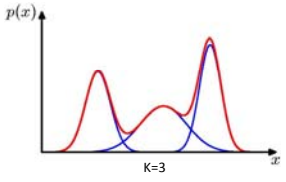
Combine simple models into a complex model:

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

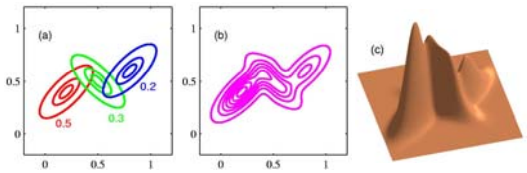
Component

Mixing coefficient

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

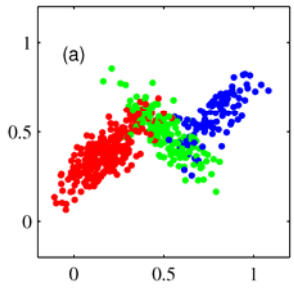


Mixtures of Gaussians (3)



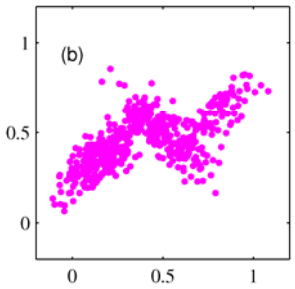
Eliminating Hard Assignments to Clusters

Model data as mixture of multivariate Gaussians



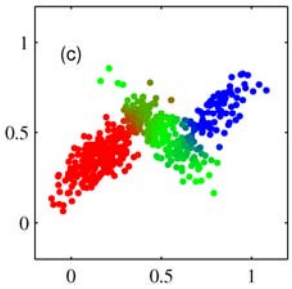
Eliminating Hard Assignments to Clusters

Model data as mixture of multivariate Gaussians



Eliminating Hard Assignments to Clusters

Model data as mixture of multivariate Gaussians



π_i = probability point was generated from i^{th} Gaussian

Detour/Review: Supervised MLE for GMM

- How do we estimate parameters for Gaussian Mixtures with fully supervised data?
- Have to define objective and solve optimization problem.

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

- For example, MLE estimate has closed form solution:

$$\mu_{ML} = \frac{1}{n}\sum_{j=1}^{n} x_n \qquad \Sigma_{ML} = \frac{1}{n}\sum_{j=1}^{n} (\mathbf{x}_j - \mu_{ML})(\mathbf{x}_j - \mu_{ML})^T$$

Compare

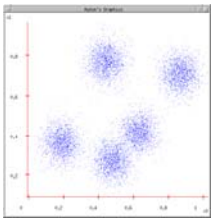
- Univariate Gaussian

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

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

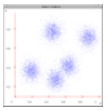
$$\mu_{ML} = \frac{1}{n}\sum_{j=1}^{n} x_n \qquad \Sigma_{ML} = \frac{1}{n}\sum_{j=1}^{n}(\mathbf{x}_j - \mu_{ML})(\mathbf{x}_j - \mu_{ML})^T$$

That was easy! But what if ***unobserved data***?



- MLE:
  - $\text{argmax}_\theta \prod_j P(y_j, x_j)$
  - θ: all model parameters
    - eg, class probs, means, and variance for naïve Bayes
- But we don't know y_j's!!!
- Maximize ***marginal likelihood***:
  - $\text{argmax}_\theta \prod_j P(x_j) = \text{argmax} \prod_j \sum_{i=1}^{k} P(y_j=i, x_j)$

How do we optimize? Closed Form?

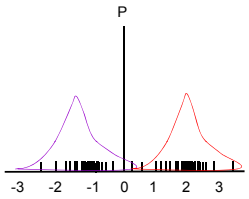


- Maximize ***marginal likelihood***:
  - $\text{argmax}_\theta \prod_j P(x_j) = \text{argmax} \prod_j \sum_{i=1}^{k} P(y_j=i, x_j)$
- Almost always a hard problem!
  - Usually no closed form solution
  - Even when P(X,Y) is convex, P(X) generally isn't…
  - For all but the simplest P(X), we will have to do gradient ascent, in a big messy space with lots of local optimum…

Simple example: learn means only!

Consider:

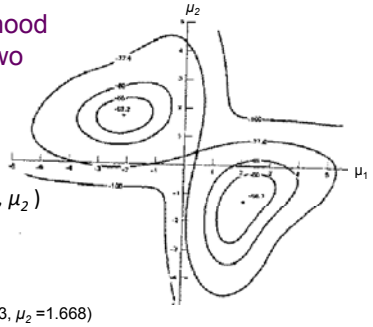
- 1D data
- Mixture of k=2 Gaussians
- Variances fixed to σ=1
- Dist'n over classes is uniform
- Just estimate μ_1 and μ_2



$$\prod_{j=1}^{m}\sum_{i=1}^{k} P(x, y=i) \propto \prod_{j=1}^{m}\sum_{i=1}^{k} \exp\left[-\frac{1}{2\sigma^2}\|x-\mu_i\|^2\right] P(y=i)$$

Marginal Likelihood for Mixture of two Gaussians

Graph of
$\log P(x_1, x_2 .. x_n \mid \mu_1, \mu_2)$
against μ_1 and μ_2



Max likelihood = (μ_1 =-2.13, μ_2 =1.668)

Local minimum, but very close to global at (μ_1 =2.085, μ_2 =-1.257)*

* corresponds to switching y_1 with y_2.

Learning general mixtures of Gaussian

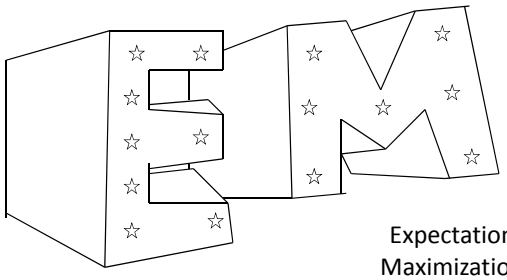
$$P(y=i \mid \mathbf{x}_j) \propto \frac{1}{(2\pi)^{m/2}\|\Sigma_i\|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}_j - \mu_i)^T \Sigma_i^{-1}(\mathbf{x}_j - \mu_i)\right] P(y=i)$$

- Marginal likelihood:

$$\prod_{j=1}^{m} P(\mathbf{x}_j) = \prod_{j=1}^{m}\sum_{i=1}^{k} P(\mathbf{x}_j, y=i)$$

$$= \prod_{j=1}^{m}\sum_{i=1}^{k} \frac{1}{(2\pi)^{m/2}\|\Sigma_i\|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}_j - \mu_i)^T \Sigma_i^{-1}(\mathbf{x}_j - \mu_i)\right] P(y=i)$$

- Need to differentiate and solve for μ_i, Σ_i, and P(Y=i) for i=1..k
- There will be no closed for solution, gradient is complex, lots of local optimum
- ***Wouldn't it be nice if there was a better way!?!***



Expectation
Maximization

The EM Algorithm

- A clever method for maximizing marginal likelihood:
  - $\text{argmax}_\theta \prod_j P(x_j) = \text{argmax}_\theta \prod_j \sum_{i=1}^k P(y_j=i,x_j)$
  - A type of gradient ascent that can be 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: $\text{argmax}_\theta \prod_j \sum_{i=1}^k P(y_j=i,x_j|\theta) = \sum_j \log \sum_{i=1}^k P(y_j=i,x_j|\theta)$

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

Notation a bit inconsistent
Parameters = θ=λ

- **E-step**: Compute expectations to "fill in" missing y values according to current parameters, θ
  - For all examples j and values i for y, compute: $P(y_j=i \mid x_j, \theta)$
- **M-step**: Re-estimate the parameters with "weighted" MLE estimates
  - Set $\theta = \text{argmax}_\theta \sum_j \sum_{i=1}^k P(y_j=i \mid x_j, \theta) \log P(y_j=i,x_j|\theta)$

Especially useful when the E and M steps have closed form solutions!!!

E.M. for General GMMs

$p_i^{(t)}$ is shorthand for estimate of $P(y=i)$ on t'th iteration

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

$$\lambda_t = \{ \mu_1^{(t)}, \mu_2^{(t)} \ldots \mu_k^{(t)}, \Sigma_1^{(t)}, \Sigma_2^{(t)} \ldots \Sigma_k^{(t)}, p_1^{(t)}, p_2^{(t)} \ldots p_k^{(t)} \}$$

E-step

Compute "expected" classes of all datapoints for each class

$$P(y=i|x_j,\lambda_t) \propto p_i^{(t)} p(x_j|\mu_i^{(t)},\Sigma_i^{(t)})$$

Just evaluate a Gaussian at x_j

M-step

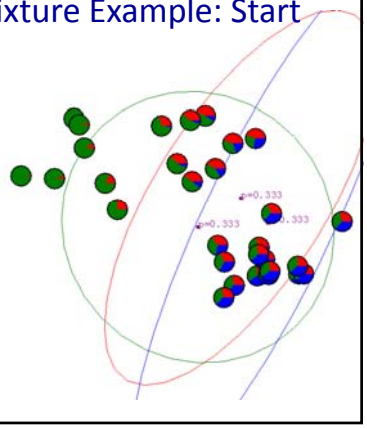
Compute weighted MLE for **μ** given expected classes above

$$\mu_i^{(t+1)} = \frac{\sum_j P(y=i|x_j,\lambda_t) x_j}{\sum_j P(y=i|x_j,\lambda_t)} \qquad \Sigma_i^{(t+1)} = \frac{\sum_j P(y=i|x_j,\lambda_t)[x_j-\mu_i^{(t+1)}][x_j-\mu_i^{(t+1)}]^T}{\sum_j P(y=i|x_j,\lambda_t)}$$

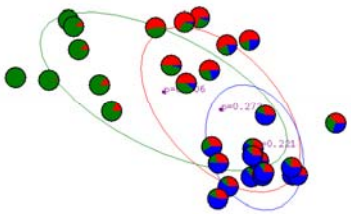
$$p_i^{(t+1)} = \frac{\sum_j P(y=i|x_j,\lambda_t)}{m}$$

m = #training examples

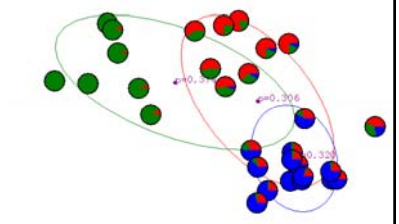
Gaussian Mixture Example: Start



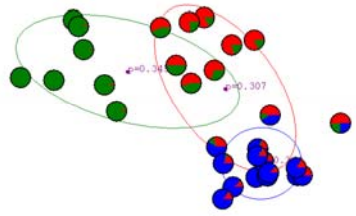
After first iteration



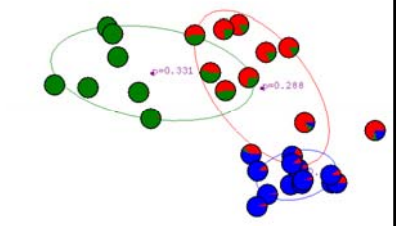
After 2nd iteration



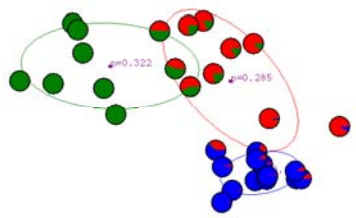
After 3rd iteration



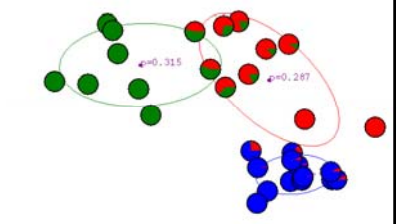
After 4th iteration



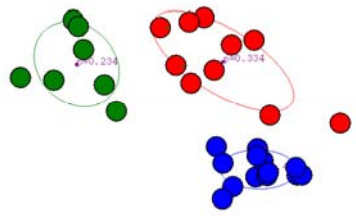
After 5th iteration



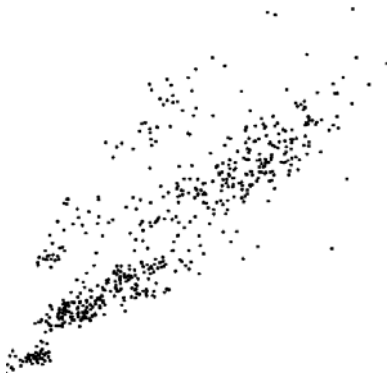
After 6th iteration



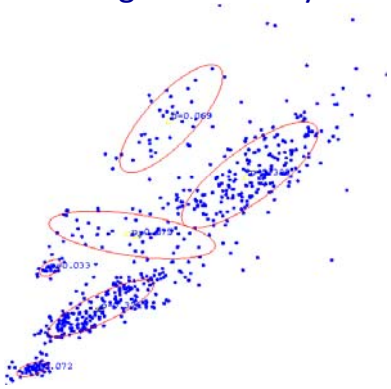
After 20th iteration



Some Bio Assay data



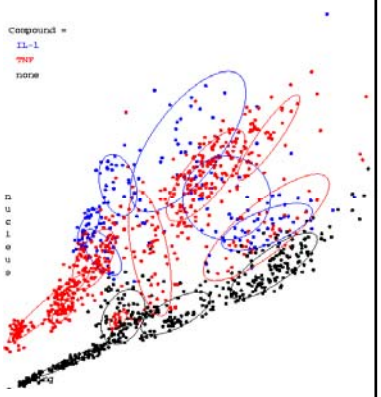
GMM clustering of the assay data



Resulting Density Estimator

Three classes of assay

(each learned with it's own mixture model)



What if we do hard assignments?

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

$$\theta_t = \{ \mu_1^{(t)}, \mu_2^{(t)} \ldots \mu_k^{(t)} \}$$

E-step

Compute "expected" classes of all datapoints

$$p(y=i|x_j,\mu_1 \ldots \mu_k) \propto \exp\left(-\frac{1}{2\sigma^2}\|x_j-\mu_i\|^2\right) P(y=i)$$

M-step

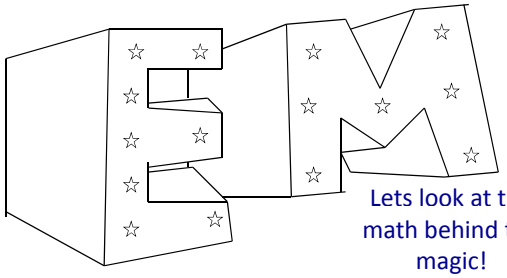
Compute most likely new **μ**s given class expectations

$$\mu_i = \frac{\sum_{j=1}^{m} P(y=i|x_j)\, x_j}{\sum_{j=1}^{m} P(y=i|x_j)}$$

$$\mu_i = \frac{\delta(y=i,x_j)\, x_j}{\sum_{j=1}^{m} \delta(y=i,x_j)}$$

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

Equivalent to k-means clustering algorithm!!!



Lets look at the math behind the magic!

We will argue that EM:

- Optimizes a bound on the likelihood
- Is a type of coordinate ascent
- Is guaranteed to converge to a (often local) optima

The general learning problem with missing data

- Marginal likelihood: **x** is observed, **z** (eg 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_θ l(θ:Data)

Skipping Gnarly Math

- EM Converges
  - E-step doesn't decrease F(θ, D)
  - M-step doesn't either
- EM is Coordinate Ascent

$$\ell(\theta : \mathcal{D}) \geq F(\theta, Q) = \sum_{j=1}^{m} \sum_{\mathbf{z}} Q(\mathbf{z} \mid \mathbf{x}_j) \log \frac{P(\mathbf{z}, \mathbf{x}_j \mid \theta)}{Q(\mathbf{z} \mid \mathbf{x}_j)}$$

60

What you should know

- K-means for clustering:
  - algorithm
  - converges because it's coordinate ascent
- Know what agglomerative clustering is
- EM for mixture of Gaussians:
  - Also coordinate ascent
  - How to "learn" maximum likelihood parameters (locally max. like.) in the case of unlabeled data
  - Relation to K-means
    - Hard / soft clustering
    - Probabilistic model
- Remember, E.M. can get stuck in local minima,
  - And empirically it ***DOES***

Acknowledgements

- K-means & Gaussian mixture models presentation contains material from excellent tutorial by Andrew Moore:
  - http://www.autonlab.org/tutorials/
- K-means Applet:
  - http://www.elet.polimi.it/upload/matteucc/Clustering/tutorial_html/AppletKM.html
- Gaussian mixture models Applet:
  - http://www.neurosci.aist.go.jp/%7Eakaho/MixtureEM.html