Unsupervised Learning with Mixtures of Gaussians

EM Algorithm - continued

Machine Learning – CSE446
Carlos Guestrin
University of Washington

May 20, 2013

Supervised Learning of Mixtures of Gaussians

K components (y^j, x^j)

- Mixtures of Gaussians:
  - Prior class probabilities: P(y) ← K-1 params (multinomial)
  - Likelihood function per class: P(**x**|y=i) ← $N(\mu_i, \Sigma_i)$

$x \to m$ dims

m ↑ symmetric $m \times m$ matrix $\frac{m^2}{2} + \frac{m}{2}$ params

Total: $K-1 + K\left(\frac{m^2+m}{2}\right) + Km$

- Suppose, for each data point, we know location **x** and class y
  - Learning is easy… ☺
  - For prior P(y) $P(y=i) = \frac{\text{Count}(y=i) \text{ in data}}{N}$
  - For likelihood function:

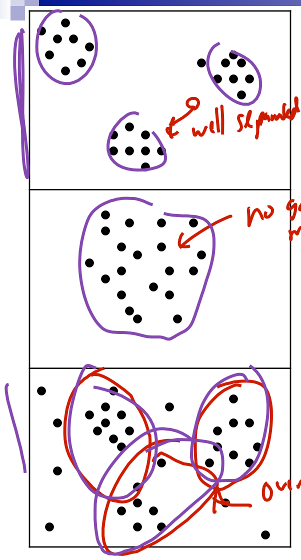
$P(x \mid y=i)$

μ_i is average of x^j for points in class i

$\Sigma_i \leftarrow \sigma_{uv} = \frac{\sum_{j \text{ in cluster } i} (x^j_u - \mu_{iu})(x^j_v - \mu_{iv})}{\text{num points in cluster } i}$

Unsupervised Learning: not as hard as it looks

we don't have y^j



Sometimes easy

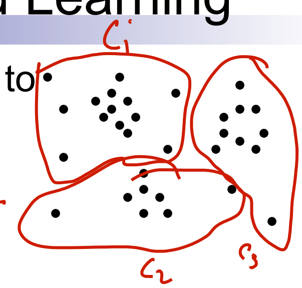
Sometimes impossible

and sometimes in between

*IN CASE YOU'RE WONDERING WHAT THESE DIAGRAMS ARE, THEY SHOW 2-d UNLABELED DATA (**X** VECTORS) DISTRIBUTED IN 2-d SPACE. THE TOP ONE HAS THREE VERY CLEAR GAUSSIAN CENTERS*

EM: "Reducing" Unsupervised Learning to Supervised Learning

- If we knew assignment of points to classes ➔ Supervised Learning!



- Expectation-Maximization (EM)
  - Guess assignment of points to classes or clusters
  - Recompute model parameters
  - Iterate

Back to Unsupervised Learning of Mixtures of spherical Gaussians – a simple version

- A simple case:
  - We have unlabeled data $x_1\ x_2 \ldots x_m$ (annotated as $x^1\ x^2 \ldots x^N$)
  - We know there are k classes
  - We know $P(y_1)\ P(y_2)\ P(y_3) \ldots P(y_k)$ ← prior
  - We <u>don't</u> know $\mu_1\ \mu_2 \ldots \mu_k$

also know σ^2 and same for all classes

We can write P(data | $\mu_1 \ldots \mu_k$)

want to max μ →

$$= \mathrm{p}\left(x^1 \ldots x^N \middle| \mu_1 \ldots \mu_k\right)$$

iid

$$= \prod_{j=1}^{N} \mathrm{p}\left(x_j \middle| \mu_1 \ldots \mu_k\right)$$

don't know y^j, so avg.

prob of $y=i$ — prior prob of cluster

$$= \prod_{j=1}^{N} \sum_{i=1}^{k} \mathrm{p}\left(x^j \middle| \mu_i\right) \mathrm{P}(y=i)$$

$$\propto \prod_{j=1}^{N} \sum_{i=1}^{k} \underbrace{\exp\left(-\frac{1}{2\sigma^2}\left\|x^j - \mu_i\right\|^2\right)}_{\text{plug in spherical gaussian}} \mathrm{P}(y=i)$$

optimize objective wrt. μ

©Carlos Guestrin 2005-2013

5

EM for simple version of Mixtures of Gaussians: The E-step

- If we know $\mu_1,\ldots,\mu_k$ → easily compute prob. point x^j belongs to class y=i

x^j = (GPA: 3.99, 446 Grade: 3.95)

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

$$\left.\begin{array}{l} P(y=0 \mid x^j, \mu) \propto 3.7 \\ P(y=1 \mid x^j, \mu) \propto 3.2 \end{array}\right\} \Rightarrow$$

$$P(y=0 \mid x^j, \mu) = \frac{3.7}{3.7+3.2} \approx 0.6$$

$$P(y=1 \mid x^j, \mu) = \frac{3.2}{3.7+3.2} \approx 0.4$$

it's like 2 data points: $(y=0, x_j)$ weight 0.6

$(y=1, x_j)$ weight 0.4

©Carlos Guestrin 2005-2013

6

EM for simple version of Mixtures of Gaussians: The M-step

- If we know prob. point x^j belongs to class y=i
  → MLE for μ_i is weighted average
  - imagine k copies of each x^j, each with weight $P(y=i|x^j)$:

$$\mu_i = \frac{\sum_{j=1}^{N} P\left(y=i \middle| x^j\right) x^j}{\sum_{j=1}^{N} P\left(y=i \middle| x^j\right)}$$

	x^0	x^1	x^2	...
$P(Y=0 \mid x^j)$	0.7	0.8	0.1	
$P(Y=1 \mid x^j)$	0.3	0.2	0.9	

weighted average

E.M. for Simple version of Mixtures of Gaussians

E-step

Compute "expected" classes of all datapoints for each class

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

Just evaluate a Gaussian at x^j

M-step

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

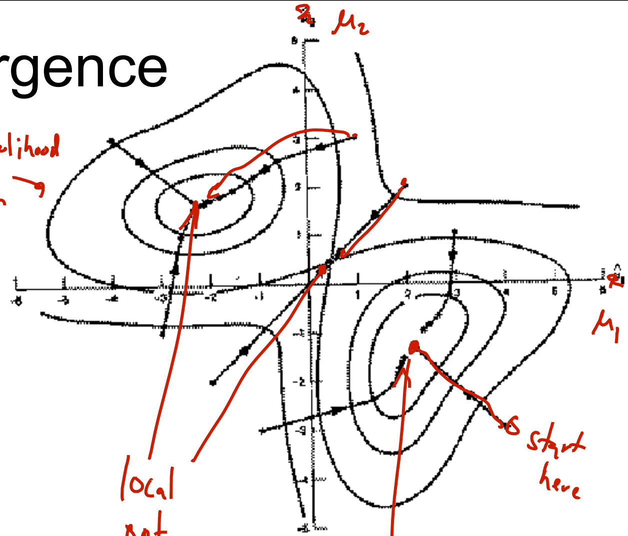
$$\mu_i = \frac{\sum_{j=1}^{m} P\left(y=i \middle| x^j\right) x^j}{\sum_{j=1}^{m} P\left(y=i \middle| x^j\right)}$$

E.M. Convergence

- EM is coordinate ascent on an interesting potential function
- Coord. ascent for bounded pot. func. ⇒ convergence to a local optimum guaranteed

not a global optima, necessarily

likelihood fn



- This algorithm is REALLY USED. And in high dimensional state spaces, too. E.G. Vector Quantization for Speech Data

E.M. for axis-aligned GMM

$$\Sigma = \begin{pmatrix} \sigma_1^2 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \sigma_2^2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \sigma_3^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_{m-1}^2 & 0 \\ 0 & 0 & 0 & \cdots & 0 & \sigma_m^2 \end{pmatrix}$$

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

$$\lambda_t = \{ \underbrace{\mu_1^{(t)}, \mu_2^{(t)} \ldots \mu_k^{(t)}}_{\text{means of Gaussian per class}}, \underbrace{\Sigma_1^{(t)}, \Sigma_2^{(t)} \ldots \Sigma_k^{(t)}}_{\text{Cov. gaussian per class}}, \underbrace{p_1^{(t)}, p_2^{(t)} \ldots p_k^{(t)}}_{\text{Priors}} \}$$

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

E-step

Compute "expected" classes of all datapoints for each class

$$P\left(y=i \middle| x^j, \lambda_t\right) \propto p_i^{(t)} p\left(x^j \middle| \mu_i^{(t)}, \Sigma_i^{(t)}\right)$$

Just evaluate a Gaussian at x^j

M-step

Compute Max. like μ, Σ, p given our data's class membership distributions

$$\mu_i^{(t+1)} = \frac{\sum_j P\left(y=i \middle| x^j, \lambda_t\right) x^j}{\sum_j P\left(y=i \middle| x^j, \lambda_t\right)}$$

weighted average

$$p_i^{(t+1)} = \frac{\sum_j P\left(y=i \middle| x^j, \lambda_t\right)}{N}$$

weighted count

~~m~~ N = #records

covariance: same as usual, but with weighted data

E.M. for General GMMs

$p_i^{(t)}$ is shorthand for estimate of prior $P(y=i)$ 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

update

$$P\left(y=i \middle| x^j, \lambda_t\right) \propto p_i^{(t)} p\left(x^j \middle| \mu_i^{(t)}, \Sigma_i^{(t)}\right)$$

Just evaluate a Gaussian at x^j

compact version of earlier slide

M-step

Compute Max. like μ given our data's class membership distributions

general learning of cov matrices

$$\mu_i^{(t+1)} = \frac{\sum_j P\left(y=i \middle| x^j, \lambda_t\right) x^j}{\sum_j P\left(y=i \middle| x^j, \lambda_t\right)}$$

$$\Sigma_i^{(t+1)} = \frac{\sum_j P\left(y=i \middle| x^j, \lambda_t\right) \left[x^j - \mu_i^{(t+1)}\right]\left[x^j - \mu_i^{(t+1)}\right]^T}{\sum_j P\left(y=i \middle| x^j, \lambda_t\right)}$$

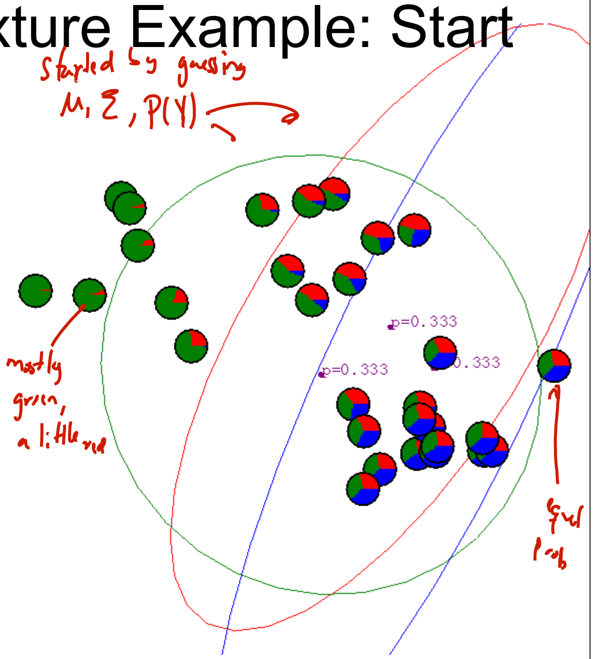
weighted data

$$p_i^{(t+1)} = \frac{\sum_j P\left(y=i \middle| x^j, \lambda_t\right)}{\cancel{m}\, N}$$

~~m~~ N = #records

Gaussian Mixture Example: Start

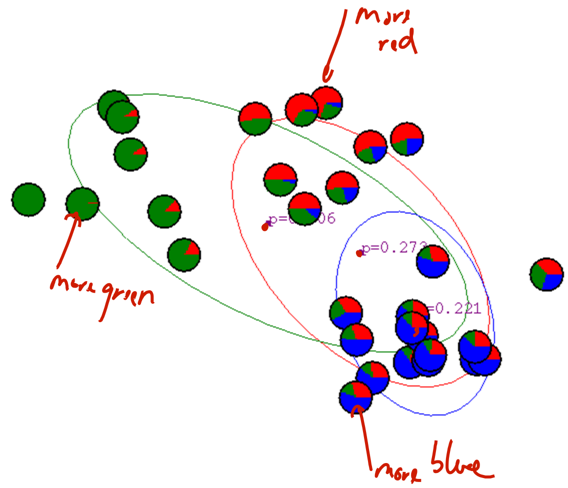
Started by guessing $\mu, \Sigma, P(Y)$



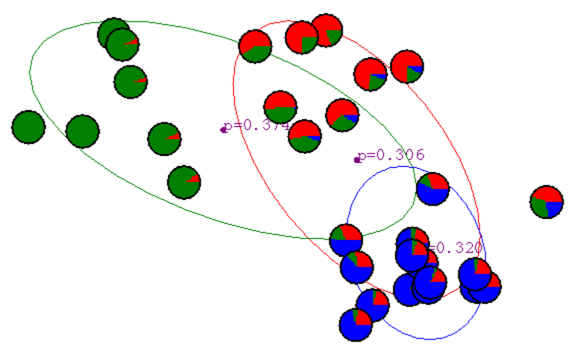
mostly green, a little red

equal prob

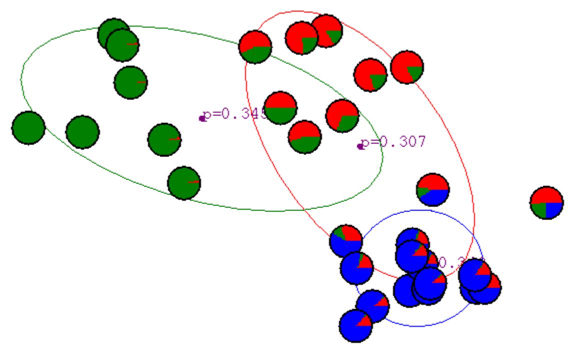
After first iteration



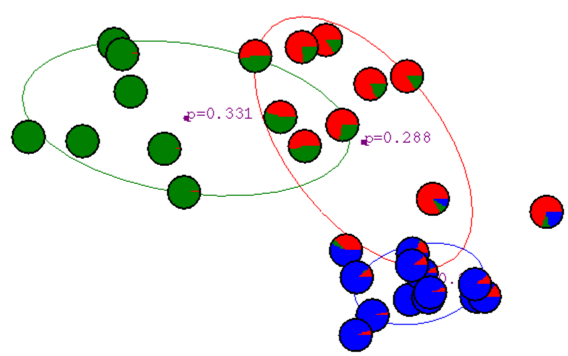
After 2nd iteration



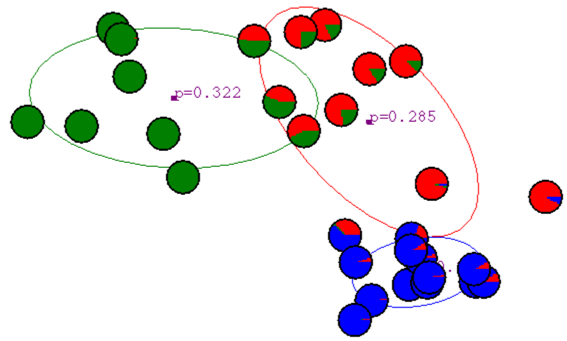
After 3rd iteration



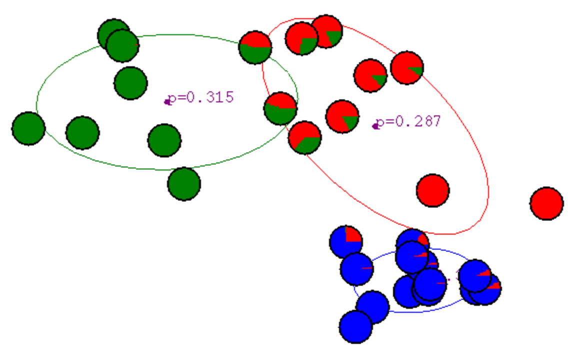
After 4th iteration



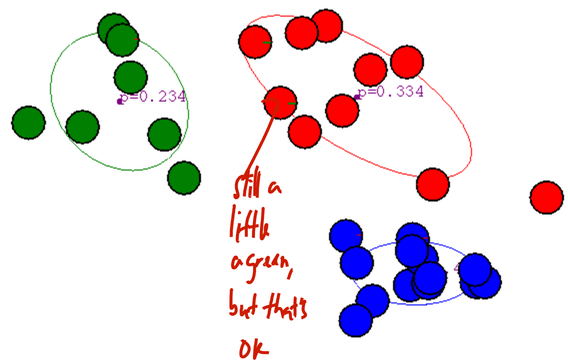
After 5th iteration



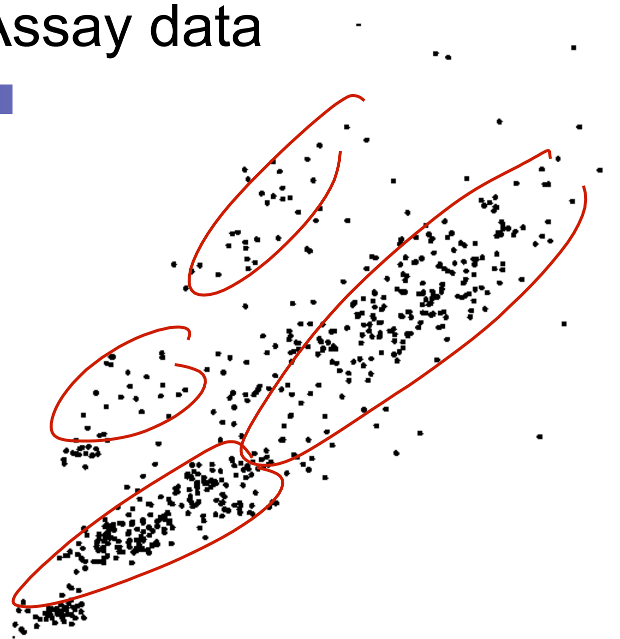
After 6th iteration



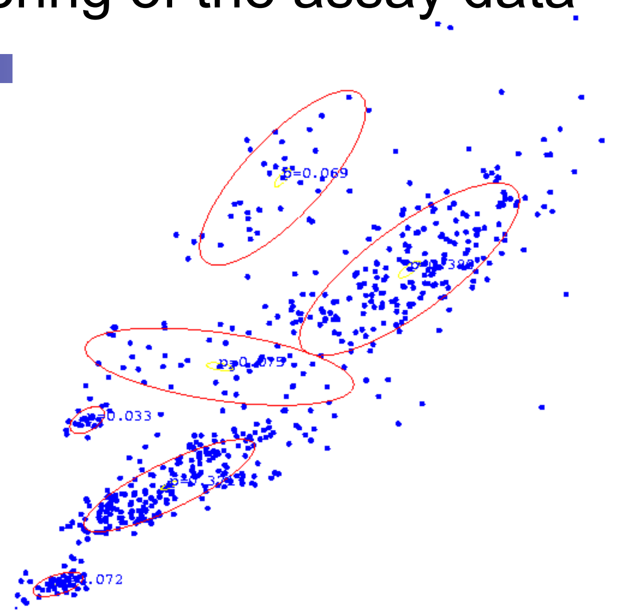
After 20th iteration



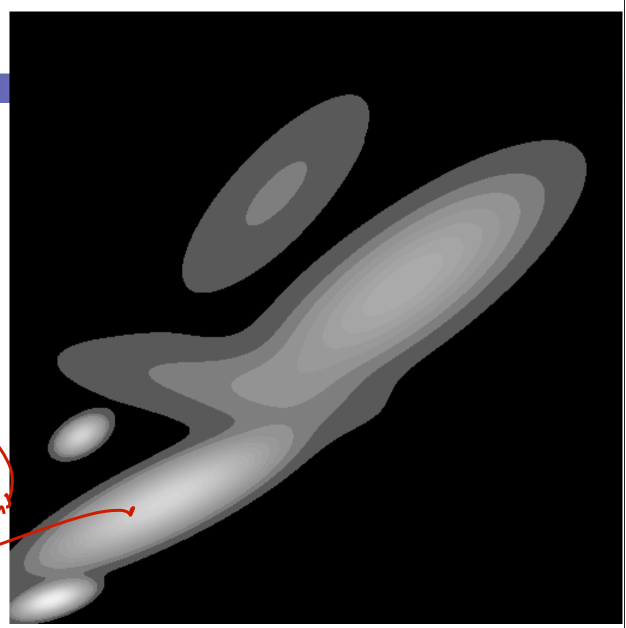
Some Bio Assay data



GMM clustering of the assay data



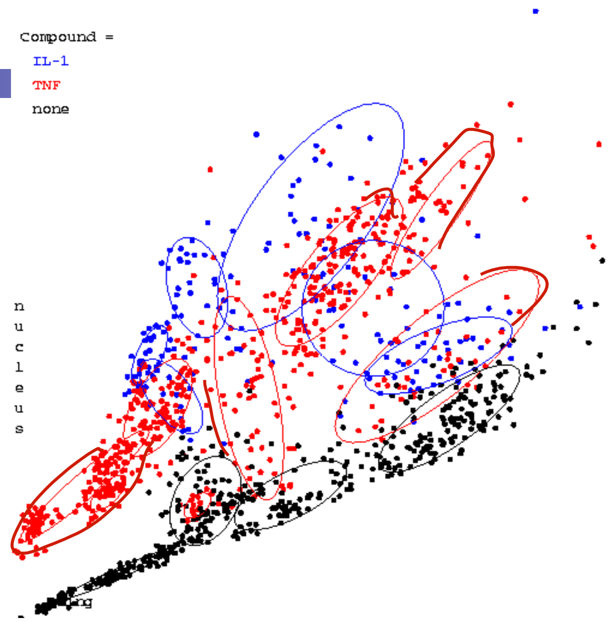
Resulting Density Estimator



P(x | mixture of Gaussians)

Three classes of assay

(each learned with it's own mixture model)

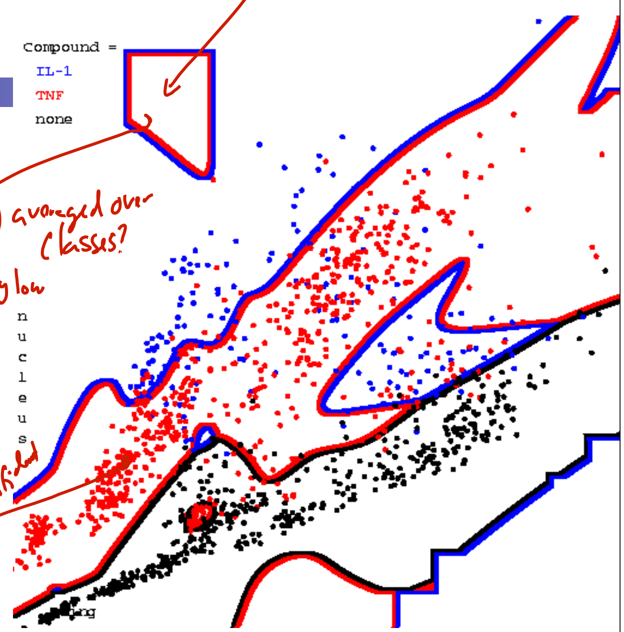


Resulting Bayes Classifier

predict red!!

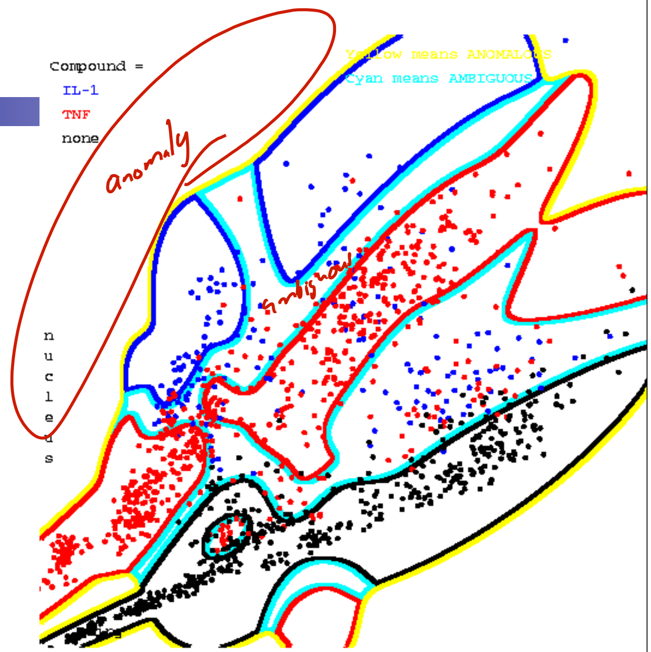
P(X) averaged over classes?

very low



predicted red pretty confident

Resulting Bayes Classifier, using posterior probabilities to alert about ambiguity and anomalousness



Yellow means anomalous

Cyan means ambiguous

E.M.: The General Case

- E.M. widely used beyond mixtures of Gaussians
  - The recipe is the same…
- Expectation Step: Fill in missing data, given current values of parameters, $\theta^{(t)}$
  - If variable y is missing (could be many variables)
  - Compute, for each data point $\mathbf{x}^j$, for each value i of y:
    - $P(y=i|\mathbf{x}^j,\theta^{(t)})$
- Maximization step: Find maximum likelihood parameters for (weighted) "completed data":
  - For each data point $\mathbf{x}^j$, create k weighted data points
    - $(y=i, x_j)$ weight $P(y=i|x^j,\theta^{(t)})$
  - Set $\theta^{(t+1)}$ as the maximum likelihood parameter estimate for this weighted data
- Repeat

What you should know

- K-means for clustering:
  - algorithm
  - converges because it's coordinate ascent
- EM for mixture of Gaussians:
  - How to "learn" maximum likelihood parameters (locally max. like.) in the case of unlabeled data
- Be happy with this kind of probabilistic analysis
- Remember, E.M. can get stuck in local minima, and empirically it <u>DOES</u>
- EM is coordinate ascent

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

Dimensionality Reduction
PCA

Machine Learning – CSE446
Carlos Guestrin
University of Washington

May 20, 2013

Dimensionality reduction

- Input data may have thousands or millions of dimensions!
  - e.g., text data has

x with 10 000 — 10 000 000 dims

- **Dimensionality reduction**: represent data with fewer dimensions
  - easier learning – fewer parameters
  - visualization – hard to visualize more than 3D or 4D
  - discover "intrinsic dimensionality" of data
    - high dimensional data that is truly lower dimensional

Lower dimensional projections

- Rather than picking a subset of the features, we can new features that are combinations of existing features

$$z_7 = 2.5x_1 + 2.9x_2 - 3.2x_3 \ldots\ldots$$

model linear

$z = Ax$

X 1000 learn

z 100 1000

logs/accounts

reconstruction error

- Let's see this in the unsupervised setting
  - just **X**, but no Y

Linear projection and reconstruction