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

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

- 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 → m dims

μ_i: m

Σ_i: ↑ symmetric m×m matrix, $\frac{m^2}{2}+\frac{m}{2}$ params

- 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:

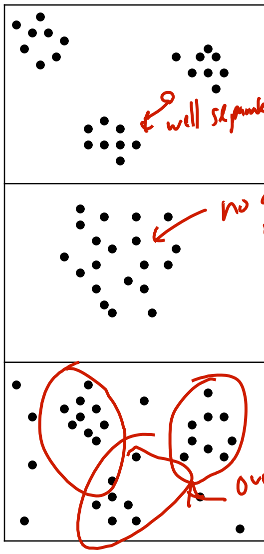
$P(x \mid y=i)$

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

no good mixture of Gaussians

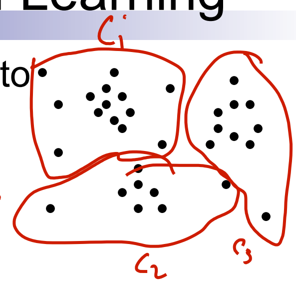
Sometimes impossible

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 Gaussians – a simple version

- A simple case:
  - We have unlabeled data $\boldsymbol{x}_1\ \boldsymbol{x}_2 \ldots \boldsymbol{x}_m$
  - We know there are k classes
  - We know $P(y_1)\ P(y_2)\ P(y_3) \ldots P(y_k)$
  - We <u>don't</u> know $\boldsymbol{\mu}_1\ \boldsymbol{\mu}_2 .. \boldsymbol{\mu}_k$

We can write P(data | $\boldsymbol{\mu}_1 \ldots \boldsymbol{\mu}_k$)

$$= \mathrm{p}\left(x_1 \ldots x_m \middle| \mu_1 \ldots \mu_k\right)$$

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

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

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

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

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

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 $P(y=i|x^j)$:

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

E.M. for Simple version of Mixtures of Gaussians

E-step

Compute "expected" classes of all datapoints for each class

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

Just evaluate a Gaussian at x^j

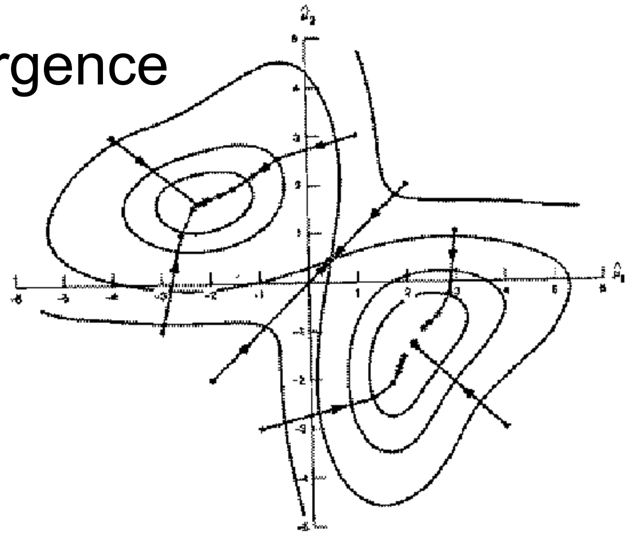
M-step

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



- 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

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

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

E-step

Compute "expected" classes of all datapoints for each class

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

Just evaluate a Gaussian at x^j

M-step

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

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

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

m = #records

E.M. for General GMMs

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

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

$p_i^{(t)}$ 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

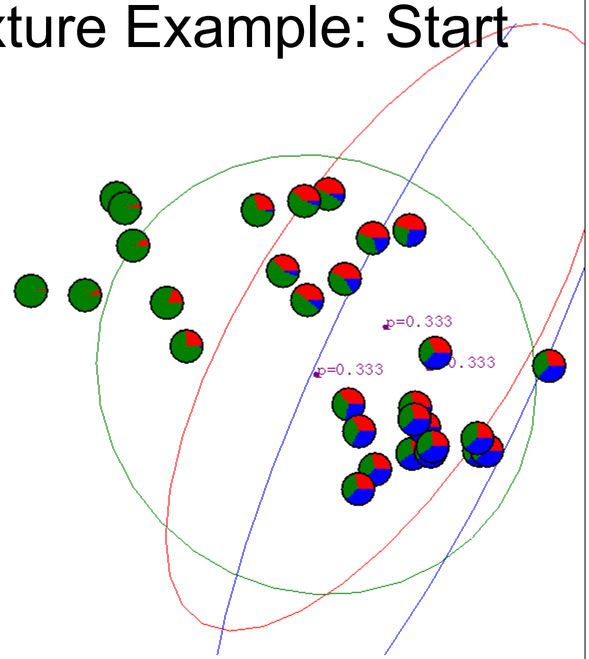
Compute Max. like **μ** given our data's class membership distributions

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

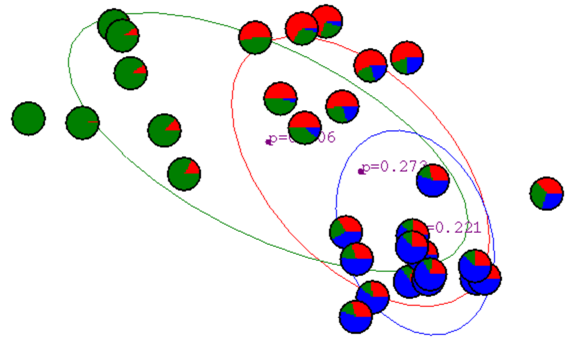
$$p_i^{(t+1)} = \frac{\sum_j P\left(y=i \middle| x^j, \lambda_t\right)}{m}$$

m = #records

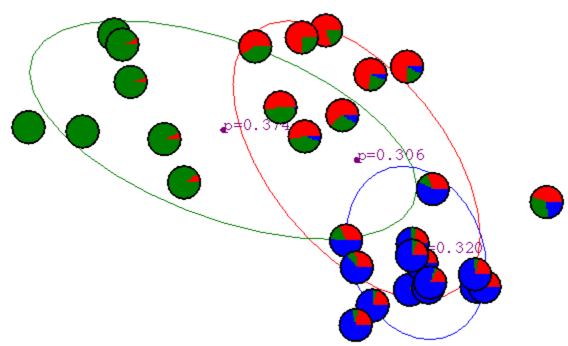
Gaussian Mixture Example: Start



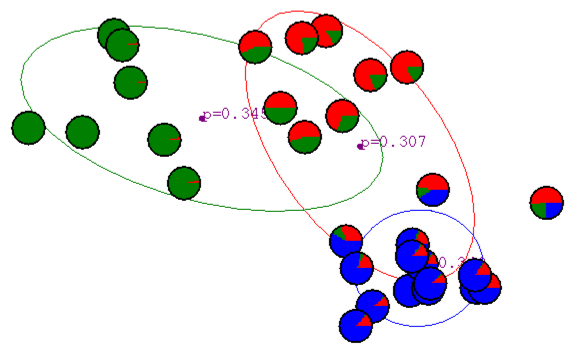
After first iteration



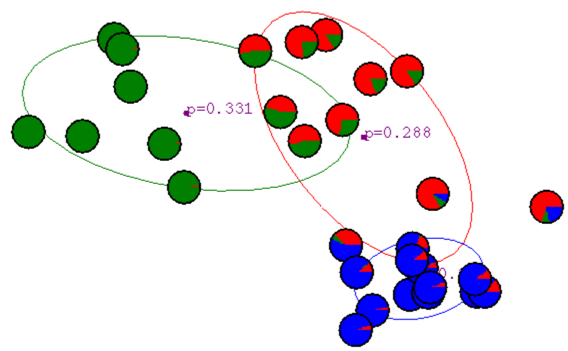
After 2nd iteration



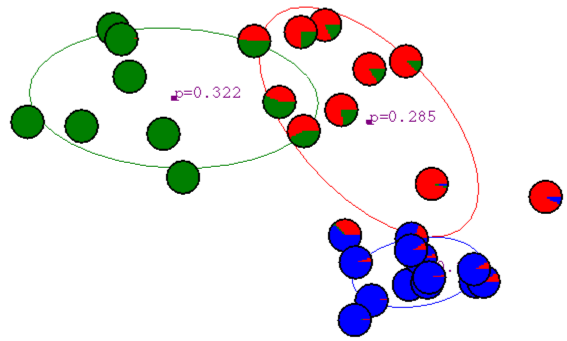
After 3rd iteration



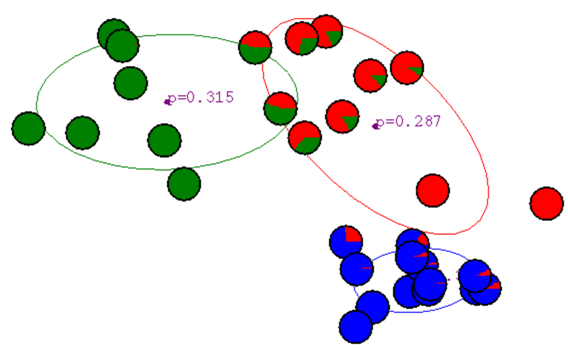
After 4th iteration



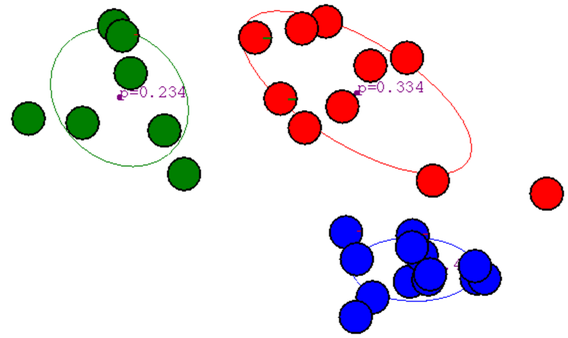
After 5th iteration



After 6th iteration



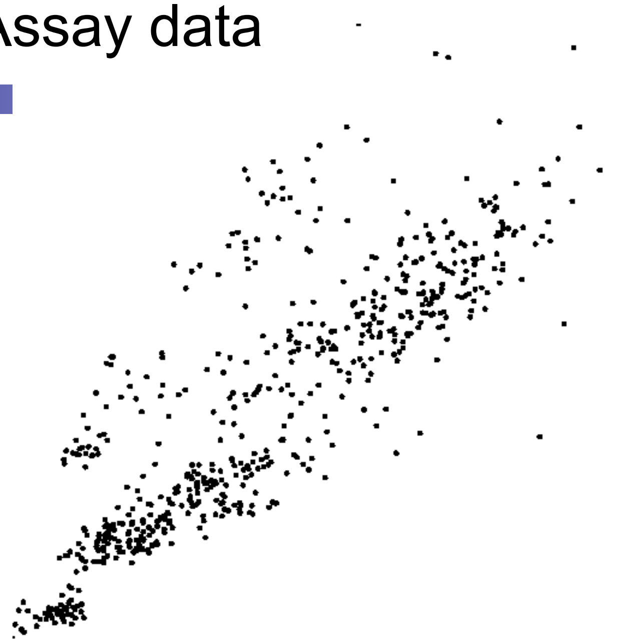
After 20th iteration



©Carlos Guestrin 2005-2013

19

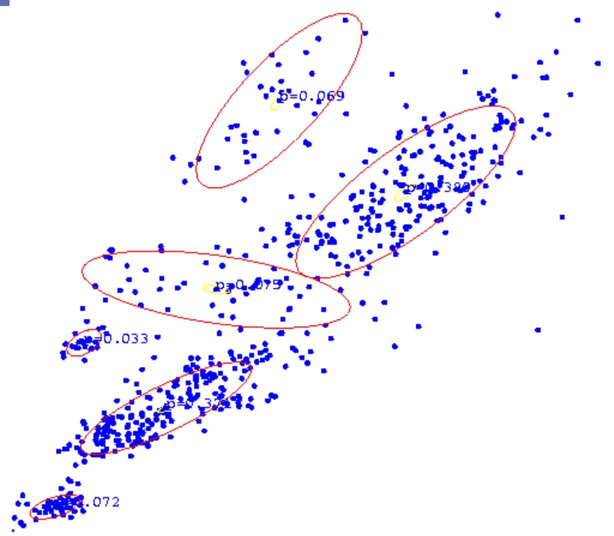
Some Bio Assay data



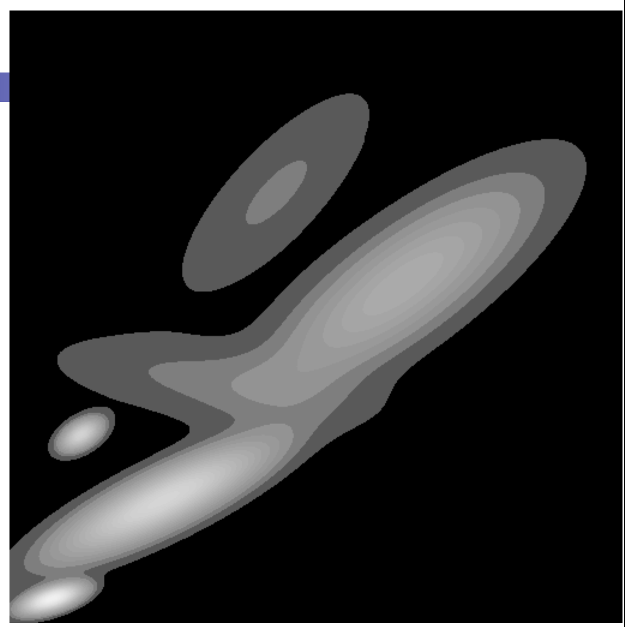
©Carlos Guestrin 2005-2013

20

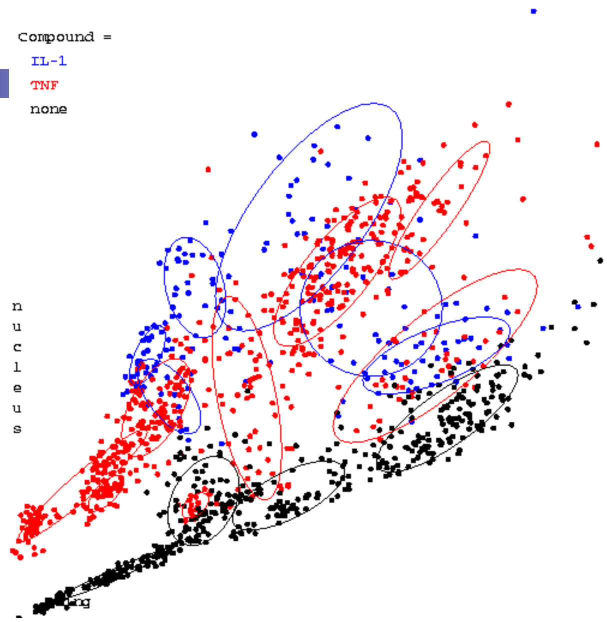
GMM clustering of the assay data



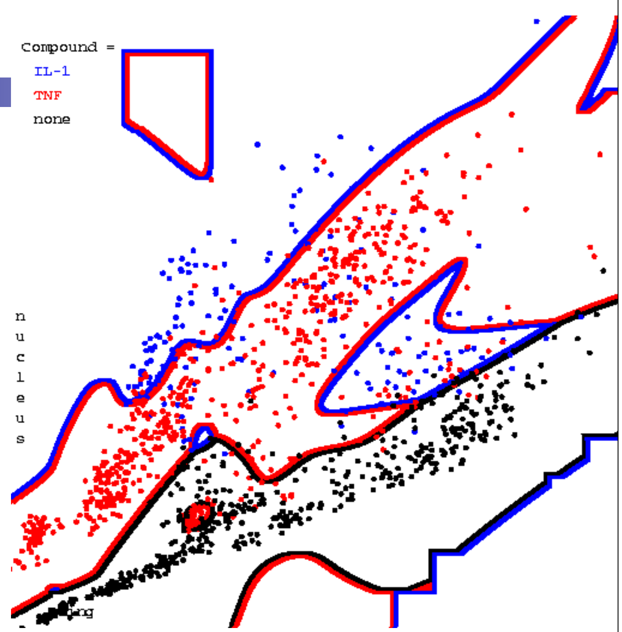
Resulting Density Estimator



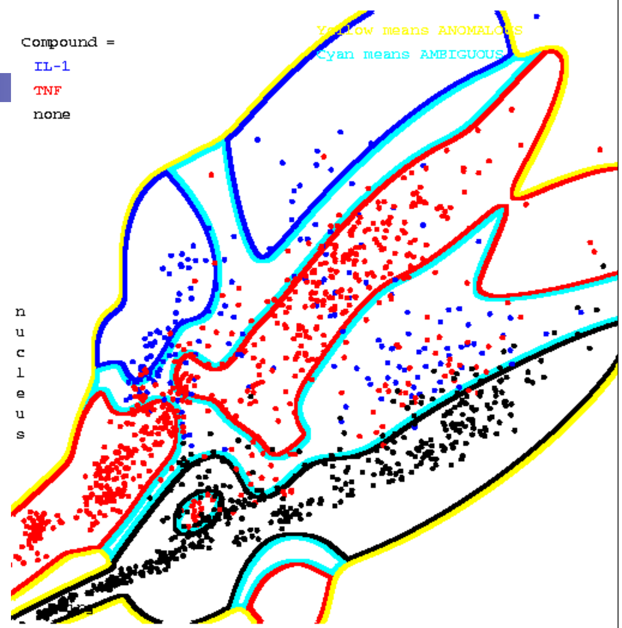
Three classes of assay
(each learned with it's own mixture model)



Resulting Bayes Classifier



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
    -
  - 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 DOES
- 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

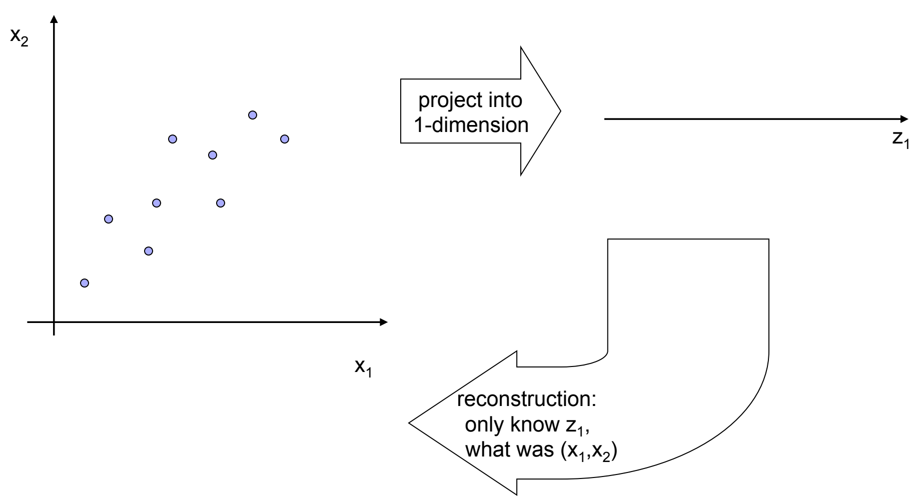
Dimensionality reduction

- Input data may have thousands or millions of dimensions!
  - e.g., text data has
- **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
- Let's see this in the unsupervised setting
  - just **X**, but no Y

Linear projection and reconstruction



Principal component analysis – basic idea

- Project n-dimensional data into k-dimensional space while preserving information:
  - e.g., project space of 10000 words into 3-dimensions
  - e.g., project 3-d into 2-d
- Choose projection with minimum reconstruction error

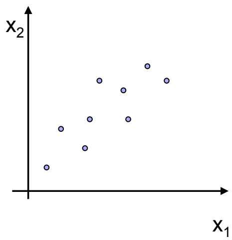
Linear projections, a review

- Project a point into a (lower dimensional) space:
  - **point**: $\mathbf{x} = (x_1,\ldots,x_d)$
  - **select a basis** – set of basis vectors – $(\mathbf{u}_1,\ldots,\mathbf{u}_k)$
    - we consider orthonormal basis:
      - $\mathbf{u}_i \cdot \mathbf{u}_i = 1$, and $\mathbf{u}_i \cdot \mathbf{u}_j = 0$ for $i \neq j$
  - **select a center** – $\bar{\mathbf{x}}$, defines offset of space
  - **best coordinates** in lower dimensional space defined by dot-products: $(z_1,\ldots,z_k)$, $z_i = (\mathbf{x}-\bar{\mathbf{x}}) \cdot \mathbf{u}_i$
    - minimum squared error

PCA finds projection that minimizes reconstruction error

- Given m data points: $\mathbf{x}^i = (x_1^i,\ldots,x_d^i)$, i=1…N
- Will represent each point as a projection:
  - $\hat{\mathbf{x}}^i = \bar{\mathbf{x}} + \sum_{j=1}^{k} z_j^i \mathbf{u}_j$ where: $\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}^i$ and $z_j^i = (\mathbf{x}^i - \bar{\mathbf{x}}) \cdot \mathbf{u}_j$
- PCA:
  - Given k<<d, find ($\mathbf{u}_1,\ldots,\mathbf{u}_k$) minimizing reconstruction error:

$$error_k = \sum_{i=1}^{N} (\mathbf{x}^i - \hat{\mathbf{x}}^i)^2$$



Understanding the reconstruction error

$$\hat{\mathbf{x}}^i = \bar{\mathbf{x}} + \sum_{j=1}^{k} z_j^i \mathbf{u}_j$$

$$z_j^i = (\mathbf{x}^i - \bar{\mathbf{x}}) \cdot \mathbf{u}_j$$

- Given k<<d, find ($\mathbf{u}_1,\ldots,\mathbf{u}_k$) minimizing reconstruction error:

$$error_k = \sum_{i=1}^{N} (\mathbf{x}^i - \hat{\mathbf{x}}^i)^2$$

- Note that $\mathbf{x}^i$ can be represented exactly by d-dimensional projection:

$$\mathbf{x}^i = \bar{\mathbf{x}} + \sum_{j=1}^{d} z_j^i \mathbf{u}_j$$

- Rewriting error:

Reconstruction error and covariance matrix

$$error_k = \sum_{i=1}^{N} \sum_{j=k+1}^{d} [\mathbf{u}_j \cdot (\mathbf{x}^i - \bar{\mathbf{x}})]^2$$

$$\Sigma = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}^i - \bar{\mathbf{x}})(\mathbf{x}^i - \bar{\mathbf{x}})^T$$

Minimizing reconstruction error and eigen vectors

- Minimizing reconstruction error equivalent to picking orthonormal basis ($\mathbf{u}_1,\ldots,\mathbf{u}_d$) minimizing:

$$error_k = N \sum_{j=k+1}^{d} \mathbf{u}_j^T \Sigma \mathbf{u}_j$$

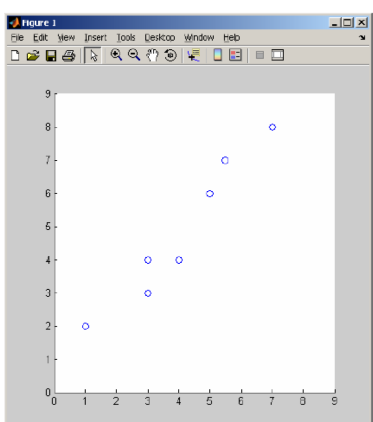
- Eigen vector:
- Minimizing reconstruction error equivalent to picking ($\mathbf{u}_{k+1},\ldots,\mathbf{u}_d$) to be eigen vectors with smallest eigen values

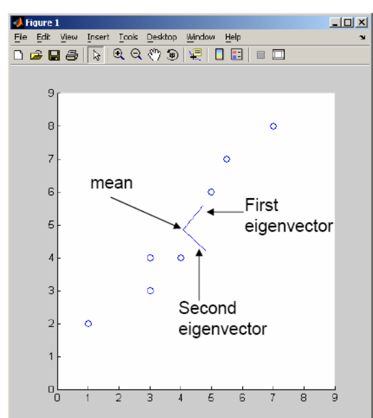
Basic PCA algoritm

- Start from m by n data matrix **X**
- **Recenter**: subtract mean from each row of **X**
  - $\mathbf{X_c} \leftarrow \mathbf{X} - \overline{\mathbf{X}}$
- **Compute covariance matrix**:
  - $\Sigma \leftarrow 1/N\ \mathbf{X_c}^T\ \mathbf{X_c}$
- Find **eigen vectors and values** of Σ
- **Principal components:** k eigen vectors with highest eigen values

PCA example

$$\hat{\mathbf{x}}^i = \bar{\mathbf{x}} + \sum_{j=1}^{k} z_j^i \mathbf{u}_j$$

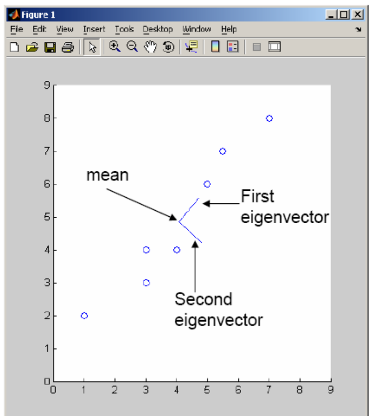


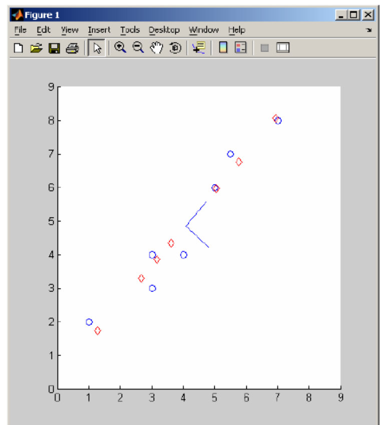


PCA example – reconstruction

$$\widehat{\mathbf{x}}^i = \bar{\mathbf{x}} + \sum_{j=1}^{k} z_j^i \mathbf{u}_j$$

only used first principal component

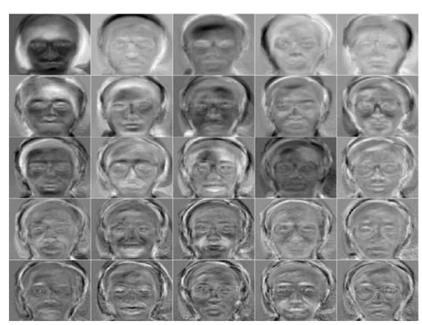




Eigenfaces [Turk, Pentland '91]

- Input images:

- Principal components:



Eigenfaces reconstruction

- Each image corresponds to adding 8 principal components:

Scaling up

- Covariance matrix can be really big!
  - Σ is d by d
  - Say, only 10000 features
  - finding eigenvectors is very slow…
- Use singular value decomposition (SVD)
  - finds to k eigenvectors
  - great implementations available, e.g., R or Matlab svd

SVD

- Write $\mathbf{X} = \mathbf{W}\ \mathbf{S}\ \mathbf{V}^T$
  - $\mathbf{X} \leftarrow$ data matrix, one row per datapoint
  - $\mathbf{W} \leftarrow$ weight matrix, one row per datapoint – coordinate of $\mathbf{x}^i$ in eigenspace
  - $\mathbf{S} \leftarrow$ singular value matrix, diagonal matrix
    - in our setting each entry is eigenvalue λ_j
  - $\mathbf{V}^T \leftarrow$ singular vector matrix
    - in our setting each row is eigenvector $\mathbf{v}_j$

PCA using SVD algoritm

- Start from m by n data matrix **X**
- **Recenter**: subtract mean from each row of **X**
  - $\mathbf{X_c} \leftarrow \mathbf{X} - \overline{\mathbf{X}}$
- Call SVD algorithm on $\mathbf{X_c}$ – ask for k singular vectors
- **Principal components:** k singular vectors with highest singular values (rows of $\mathbf{V}^T$)
  - **Coefficients** become:

What you need to know

- Dimensionality reduction
  - why and when it's important
- Simple feature selection
- Principal component analysis
  - minimizing reconstruction error
  - relationship to covariance matrix and eigenvectors
  - using SVD