Unsupervised Learning

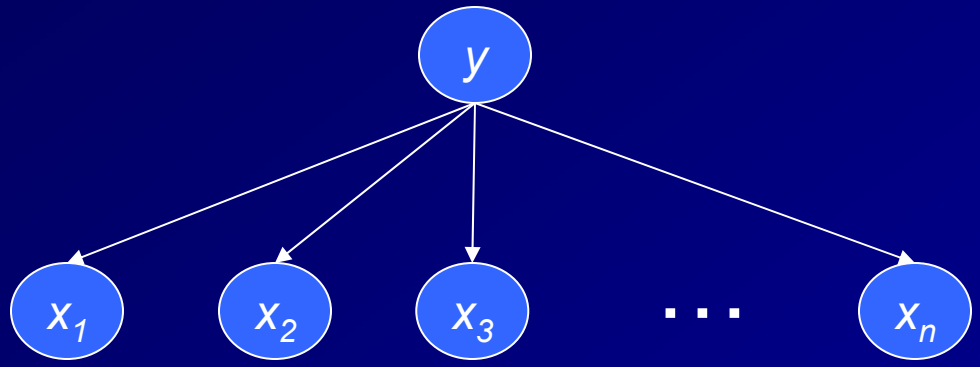
- Learning without Class Labels (or correct outputs)
  - Density Estimation
    - Learn P(X) given training data for X
  - Clustering
    - Partition data into clusters
  - Dimensionality Reduction
    - Discover low-dimensional representation of data
  - Blind Source Separation
    - Unmixing multiple signals

Density Estimation

- Given: $S = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$
- Find: $P(\mathbf{x})$
- Search problem:

  $\text{argmax}_h P(S|h) = \text{argmax}_h \sum_i \log P(\mathbf{x}_i|h)$

Unsupervised Fitting of the Naïve Bayes Model



- y is discrete with K values

  $P(\mathbf{x}) = \sum_k P(y=k) \prod_j P(x_j \mid y=k)$
- finite mixture model
- we can think of each y=k as a separate "cluster" of data points

The Expectation-Maximization Algorithm (1): Hard EM

- Learning would be easy if we knew y_i for each $\mathbf{x}_i$
- Idea: guess them and then iteratively revise our guesses to maximize P(S|h)

$\mathbf{x}_i$	y_i
$\mathbf{x}_1$	y_1
$\mathbf{x}_2$	y_2
…	…
$\mathbf{x}_N$	y_N

Hard EM (2)

1. Guess initial *y* values to get "complete data"
2. M Step: Compute probabilities for hypotheses (model) from complete data [Maximum likelihood estimate of the model parameters]
3. E Step: Classify each example using the current model to get a new *y* value [Most likely class $\hat{y}$ of each example]
4. Repeat steps 2-3 until convergence

$\mathbf{x}_i$	y_i
$\mathbf{x}_1$	y_1
$\mathbf{x}_2$	y_2
…	…
$\mathbf{x}_N$	y_N

Special Case: k-Means Clustering

1. Assign an initial y_i to each data point $\mathbf{x}_i$ at random
2. M Step. For each class k = 1, …, K compute the mean:

$$\mu_k = 1/N_k \sum_i \mathbf{x}_i \cdot I[y_i = k]$$

3. E Step. For each example x_i, assign it to the class k with the nearest mean:

$$y_i = \text{argmin}_k \|\mathbf{x}_i - \mu_k\|$$

4. Repeat steps 2 and 3 to convergence

Gaussian Interpretation of K-means

- Each feature x_j in class k is gaussian distributed with mean μ_{kj} and constant variance σ^2

$$P(x_j|y=k) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\frac{(\|x_j-\mu_{kj}\|)^2}{\sigma^2}\right]$$

$$\log P(x_j|y=k) = -\frac{1}{2}\frac{\|x_j-\mu_{kj}\|^2}{\sigma^2} + C$$

$$\underset{y}{\operatorname{argmax}}\, P(\mathbf{x}|y=k) = \underset{y}{\operatorname{argmax}}\, \log P(\mathbf{x}|y) = \underset{y}{\operatorname{argmin}} \|\mathbf{x}-\mu_{kj}\|^2 = \underset{y}{\operatorname{argmin}} \|\mathbf{x}-\mu_{kj}\|$$

- This could easily be extended to have general covariance matrix Σ or class-specific Σ_k

The EM algorithm

- The true EM algorithm augments the incomplete data with a probability distribution over the possible *y* values

1. Start with initial naive Bayes hypothesis
2. E step: For each example, compute $P(y_i)$ and add it to the table
3. M step: Compute updated estimates of the parameters
4. Repeat steps 2-3 to convergence.

$\mathbf{x}_i$	y_i
$\mathbf{x}_1$	$P(y_1)$
$\mathbf{x}_2$	$P(y_2)$
…	…
$\mathbf{x}_N$	$P(y_N)$

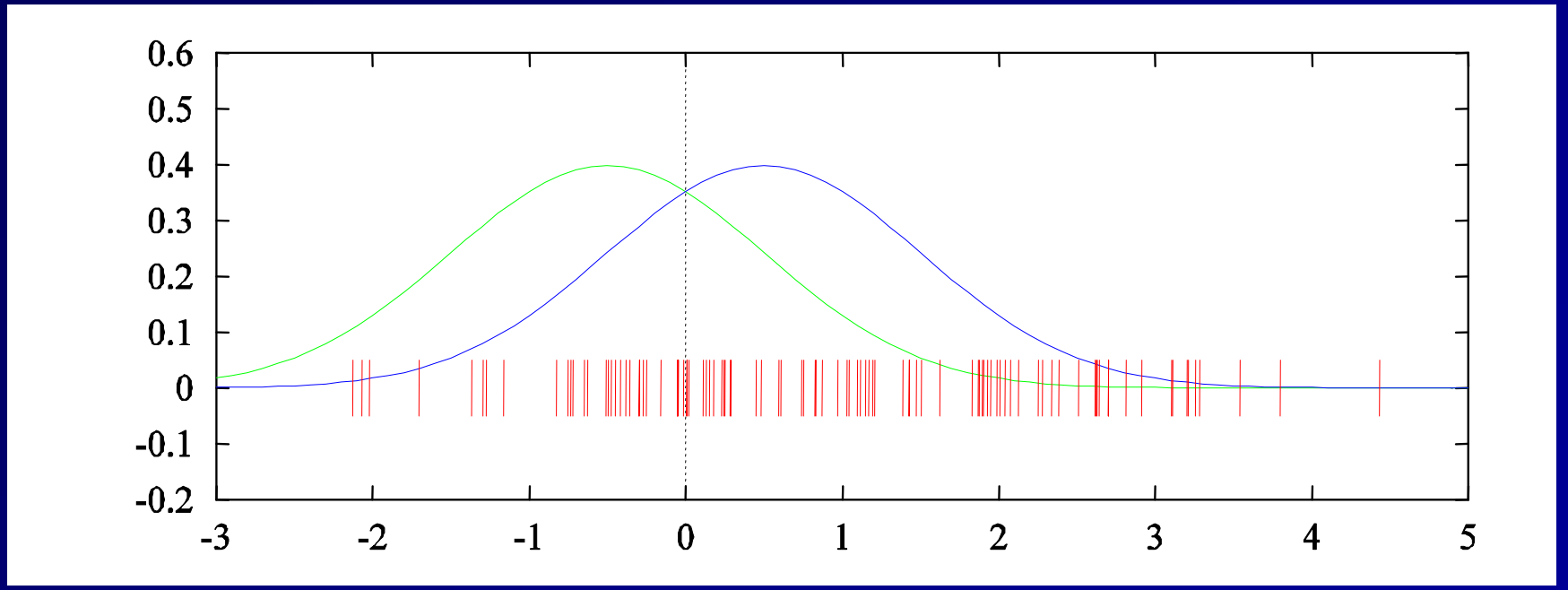
Details of the M step

- Each example $\mathbf{x}_i$ is treated as if $y_i=k$ with probability $P(y_i=k \mid \mathbf{x}_i)$

$$P(y=k) := \frac{1}{N}\sum_{i=1}^{N} P(y_i=k|\mathbf{x}_i)$$

$$P(x_j=v|y=k) := \frac{\sum_i P(y_i=k|\mathbf{x}_i)\cdot I(x_{ij}=v)}{\sum_{i=1}^{N} P(y_i=k|\mathbf{x}_i)}$$

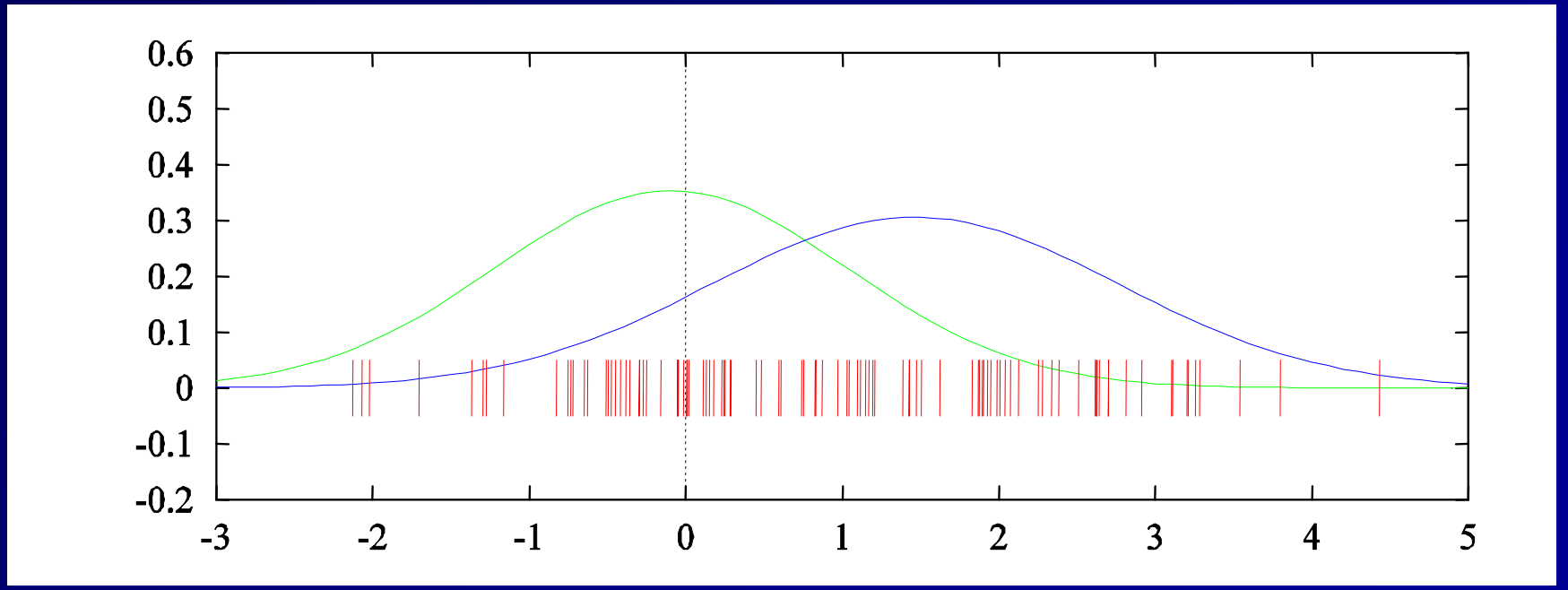
Example: Mixture of 2 Gaussians



Initial distributions

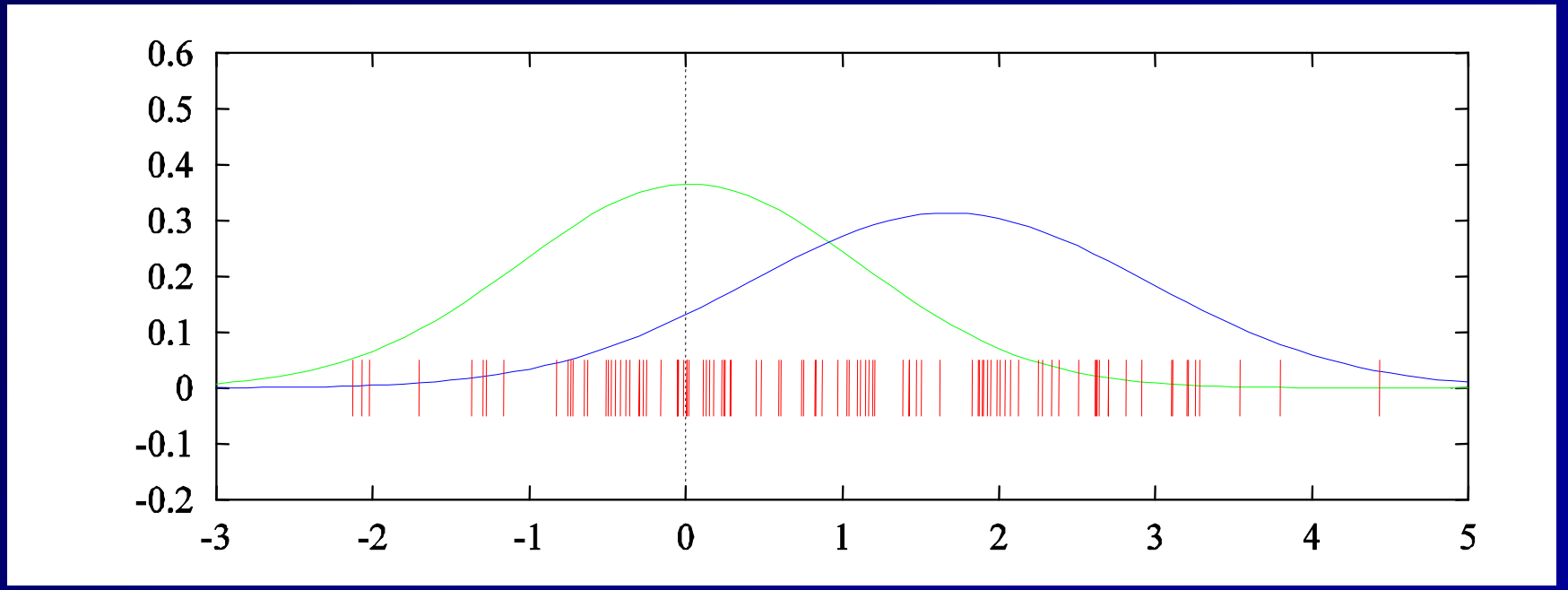
means at -0.5, +0.5

Example: Mixture of 2 Gaussians



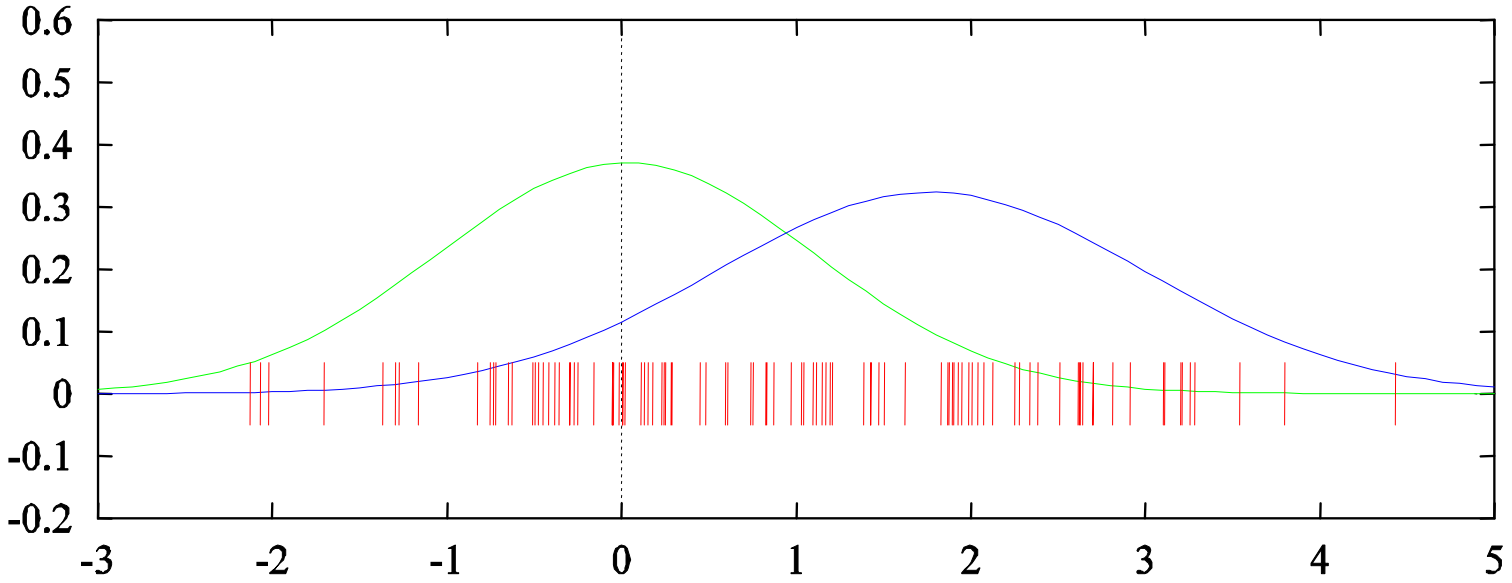
Iteration 1

Example: Mixture of 2 Gaussians



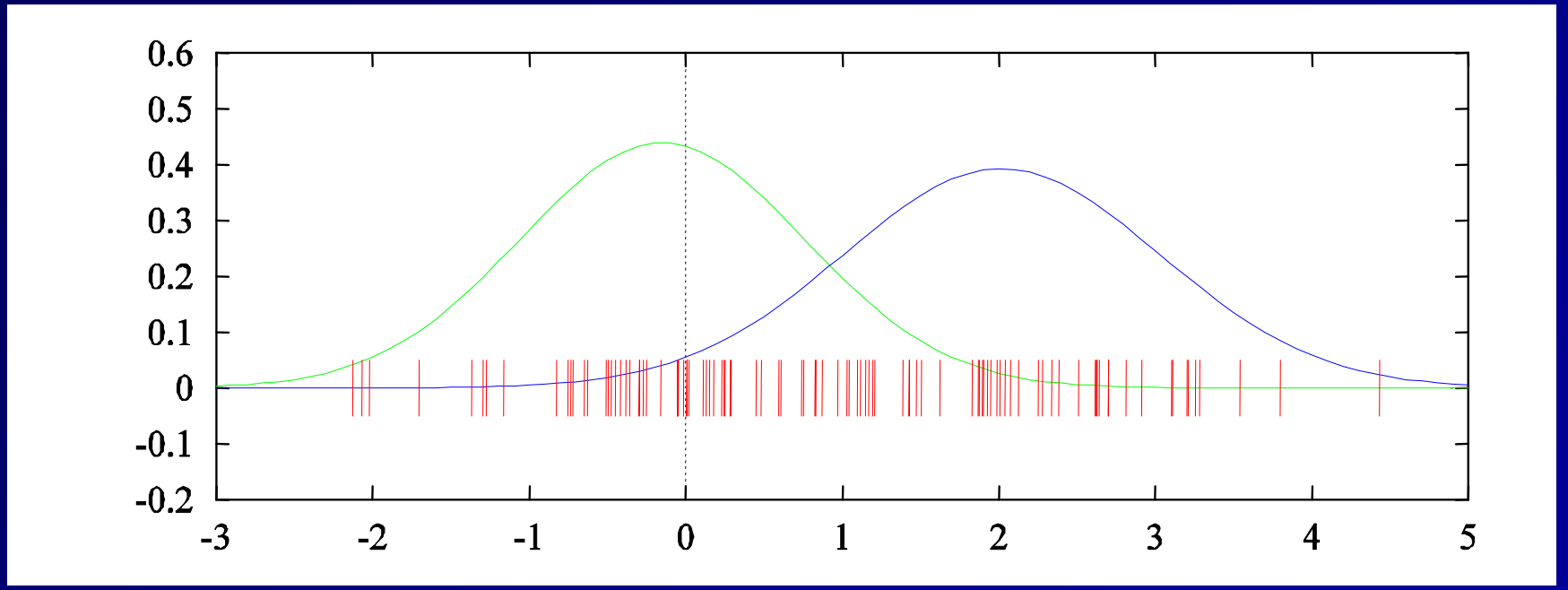
Iteration 2

Example: Mixture of 2 Gaussians



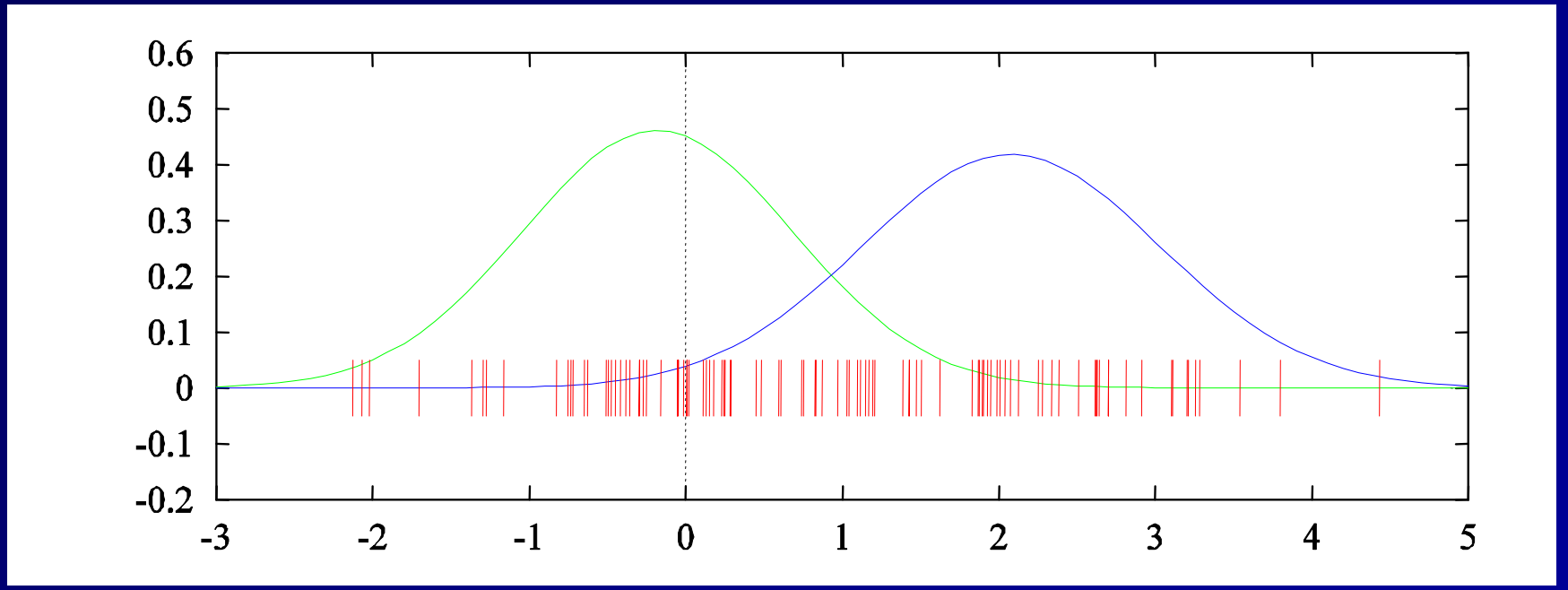
Iteration 3

Example: Mixture of 2 Gaussians



Iteration 10

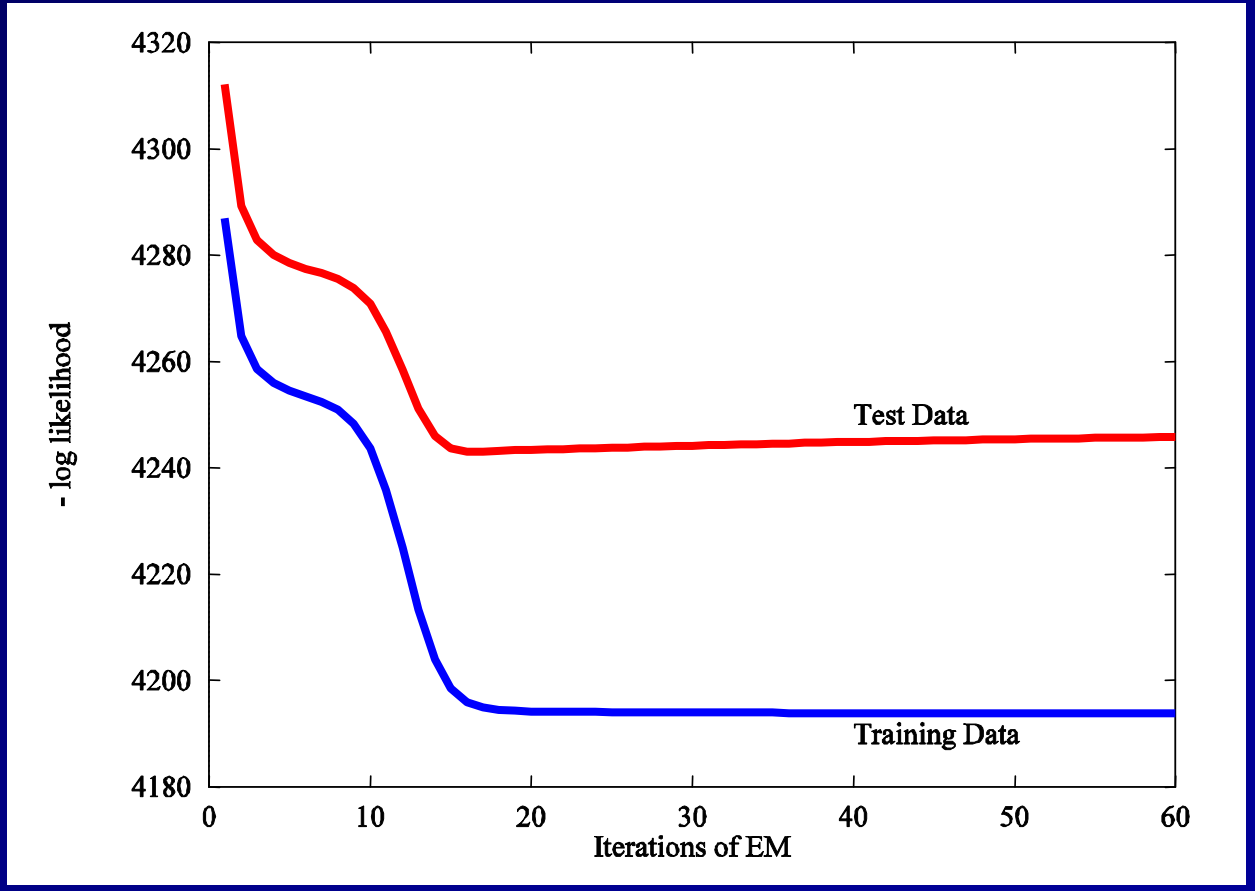
Example: Mixture of 2 Gaussians



Iteration 20

Evaluation: Test set likelihood

- Overfitting is also a problem in unsupervised learning



Potential Problems

- If σ_k is allowed to vary, it may go to zero, which leads to infinite likelihood
- Fix by placing an overfitting penalty on $1/\sigma$

Choosing K

- Internal holdout likelihood

Unsupervised Learning for Sequences

- Suppose each training example $\mathbf{X}_i$ is a sequence of objects:

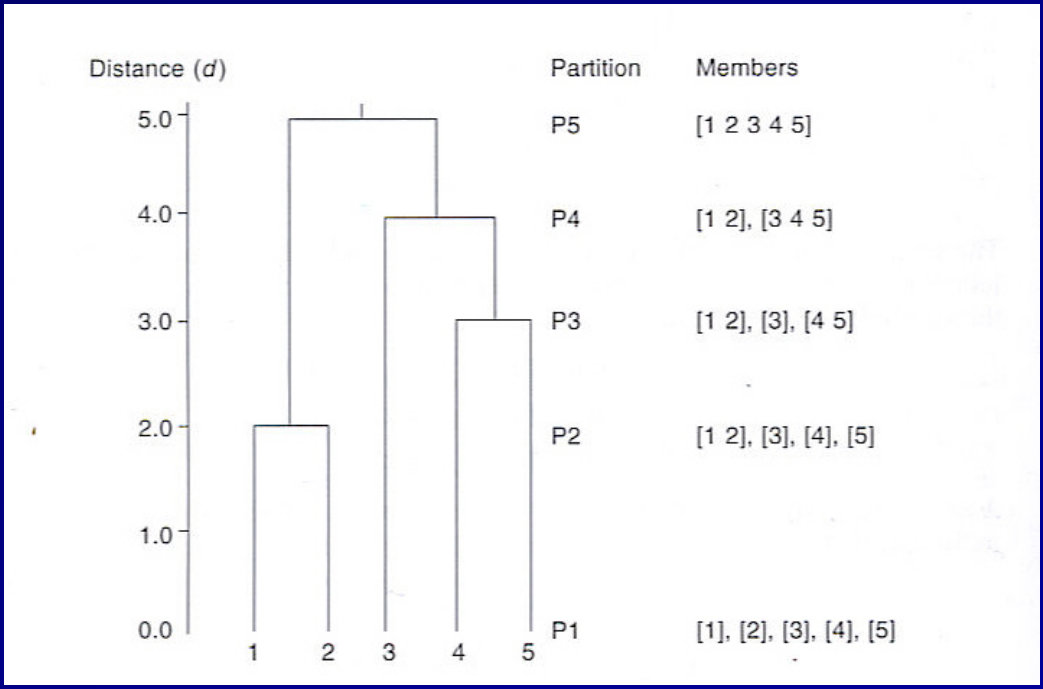
$$\mathbf{X}_i = (\mathbf{x}_{i1}, \mathbf{x}_{i2}, \ldots, \mathbf{x}_{i,T_i})$$

- Fit HMM by unsupervised learning

1. Initialize model parameters
2. E step: apply forward-backward algorithm to estimate $P(y_{it} \mid \mathbf{X}_i)$ at each point t
3. M step: estimate model parameters
4. Repeat steps 2-3 to convergence

Agglomerative Clustering

- Initialize each data point to be its own cluster
- Repeat:
  - Merge the two clusters that are most similar
  - Build dendrogram with height = distance between the most similar clusters
- Apply various intuitive methods to choose number of clusters
  - Equivalent to choosing where to “slice” the dendrogram



Source: Charity Morgan

http://www.people.fas.harvard.edu/~rizem/teach/stat325/CharityCluster.ppt

Agglomerative Clustering

- Each cluster is defined only by the points it contains (not by a parameterized model)
- Very fast (using priority queue)
- No objective measure of correctness

- Distance measures
  - distance between nearest pair of points
  - distance between cluster centers

Probabilistic Agglomerative Clustering = Bottom-up Model Merging

- Each data point is an initial cluster but with penalized σ_k
- Repeat:
  - Merge the two clusters that would most increase the penalized log likelihood
  - Until no merger would further improve likelihood
- Note that without the penalty on σ_k, the algorithm would never merge anything

Dimensionality Reduction

- Often, raw data have very high dimension
  - Example: images of human faces
- Dimensionality Reduction:
  - Construct a lower-dimensional space that preserves information important for the task
  - Examples:
    - preserve distances
    - preserve separation between classes
    - etc.

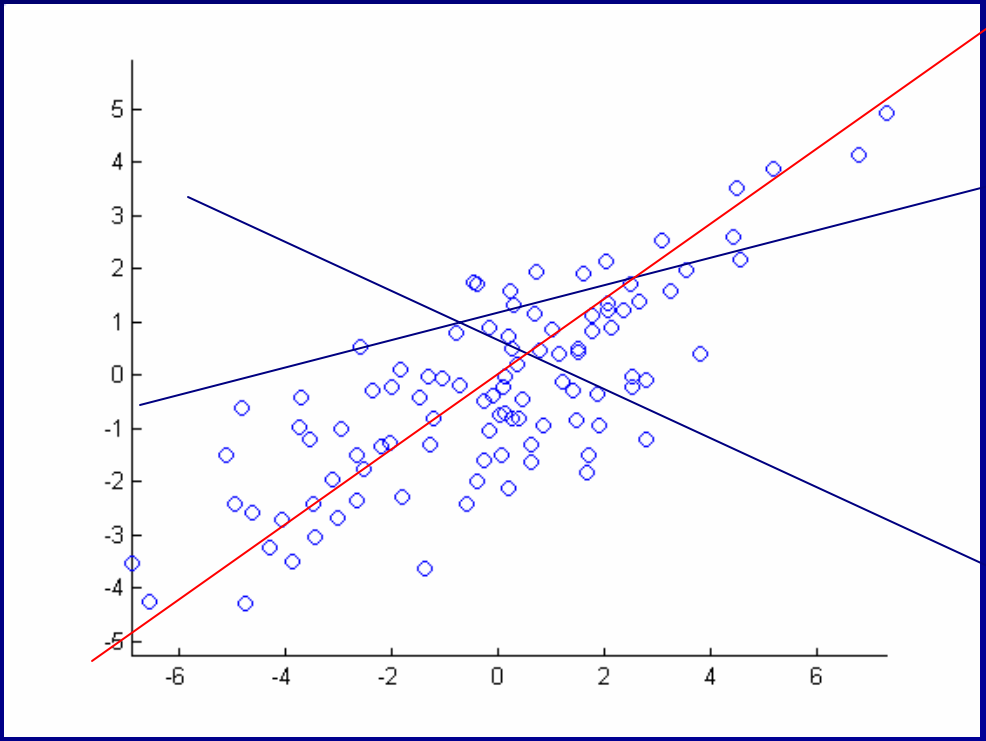
Principal Component Analysis

- Given:
  - Data: n-dimensional vectors $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$
  - Desired dimensionality m
- Find an m x n orthogonal matrix A to minimize

  $\sum_i ||A^{-1}A\mathbf{x}_i - \mathbf{x}_i||^2$
- Explanation:
  - $A\mathbf{x}_i$ maps $\mathbf{x}_i$ into an m-dimensional matrix $\mathbf{x'}_i$
  - $A^{-1}A\mathbf{x}_i$ maps $\mathbf{x'}_i$ back to n-dimensional space
  - We minimize the "squared reconstruction error" between the reconstructed vectors and the original vectors

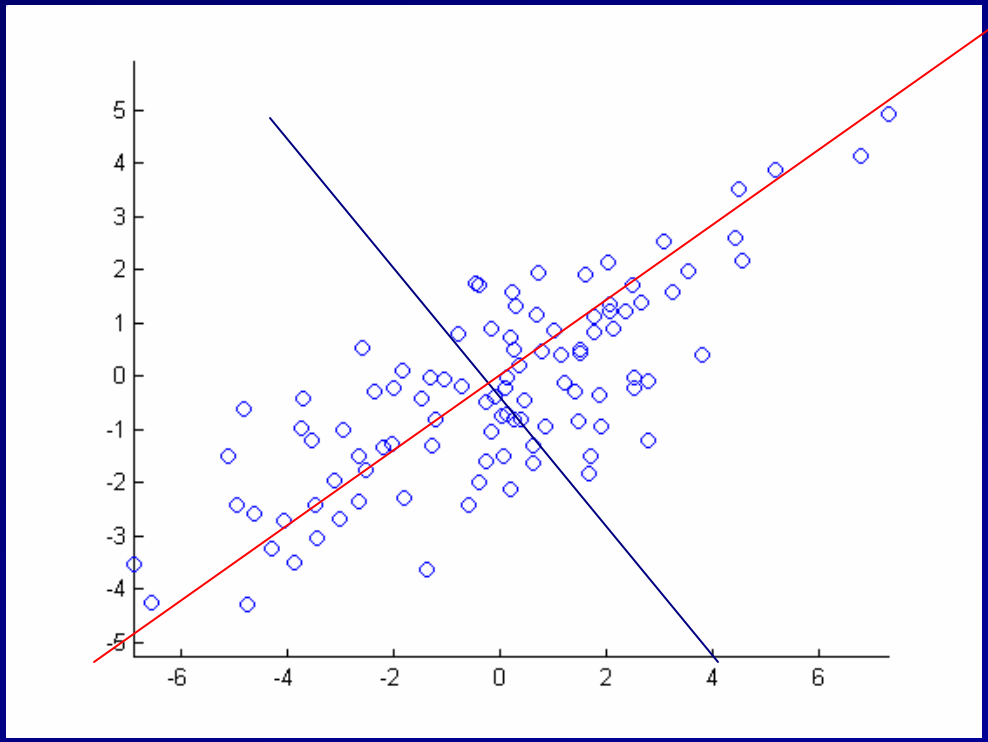
Conceptual Algorithm

- Find a line such that when the data is projected onto that line, it has the maximum variance:



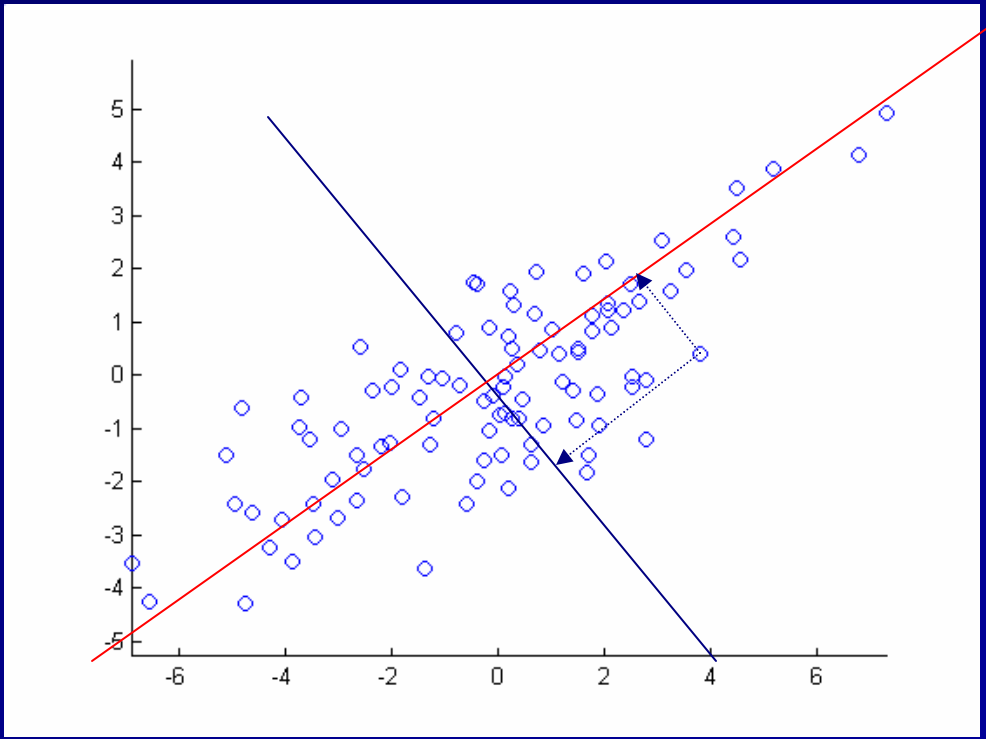
Conceptual Algorithm

- Find a new line, orthogonal to the first, that has maximum projected variance:



Repeat Until m Lines Have Been Found

- The projected position of a point on these lines gives the coordinates in the m-dimensional reduced space



A Better Method Numerical Method

- Compute the co-variance matrix

  $\Sigma = \Sigma_i (\mathbf{x}_i - \mu) \cdot (\mathbf{x}_i - \mu)^T$

- Compute the singular value decomposition

  $\Sigma = U\ D\ V^T$

  where

  – the columns of U are the eigenvectors of Σ

  – D is a diagonal matrix whose elements are the square roots of the eigenvalues of Σ in descending order

  – V^T are the projected data points

- Replace all but the m largest elements of D by zeros

Example: Eigenfaces

- Database of 128 carefully-aligned faces
- Here are the first 15 eigenvectors:

Face Classification in Eigenspace is Easier

- Nearest Mean classifier

  $\hat{y} = \text{argmin}_k \| A\mathbf{x} - A\mu_k \|$

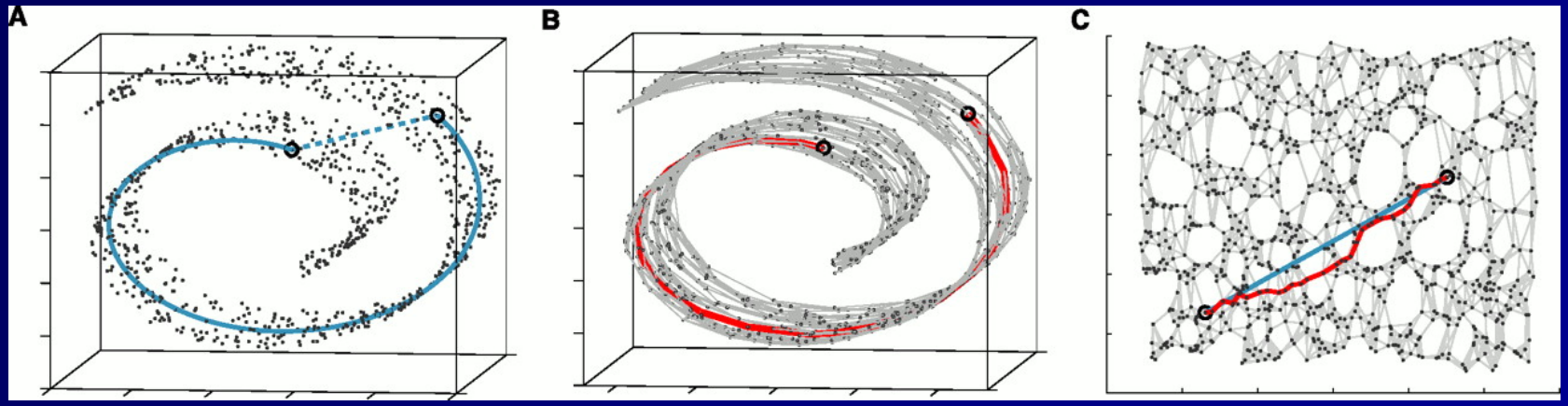
- Accuracy
  - variation in lighting: 96%
  - variation in orientation: 85%
  - variation in size: 64%

PCA is a useful preprocessing step

- Helps all LTU algorithms by making the features more independent
- Helps decision tree algorithms
- Helps nearest neighbor algorithms by discovering the distance metric

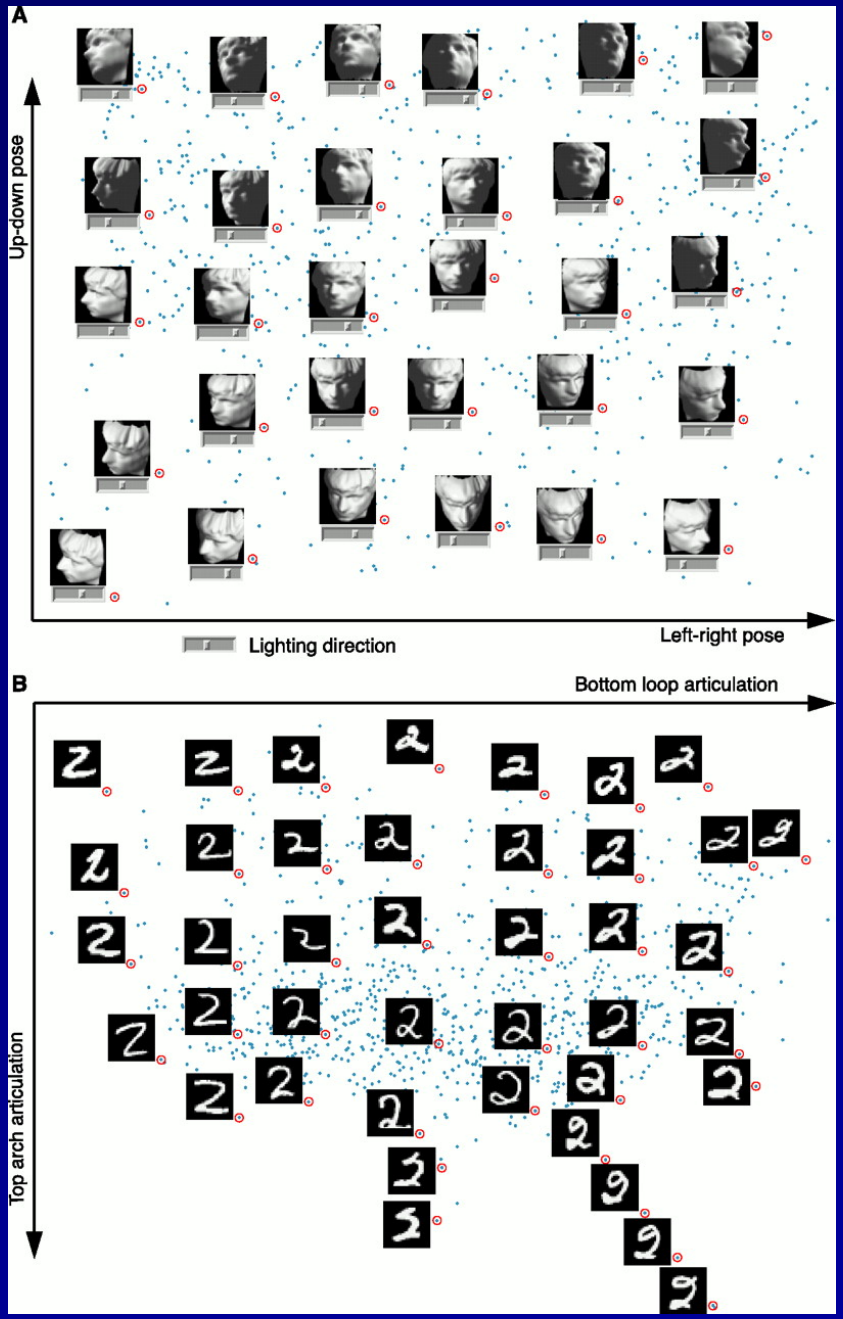
- Fails when data consists of multiple separate clusters
  - mixtures of PCAs can be learned too

Non-Linear Dimensionality Reduction: ISOMAP



- Replace Euclidean distance by geodesic distance
  - Construct a graph where each point is connected to its k nearest neighbors by an edge AND any pair of points are connected if they are less than ε apart
  - Construct an N x N matrix D in which D[i,j] is the shortest path in the graph connecting $\mathbf{x}_i$ to $\mathbf{x}_j$
  - Apply SVD to D and keep the m most important dimensions

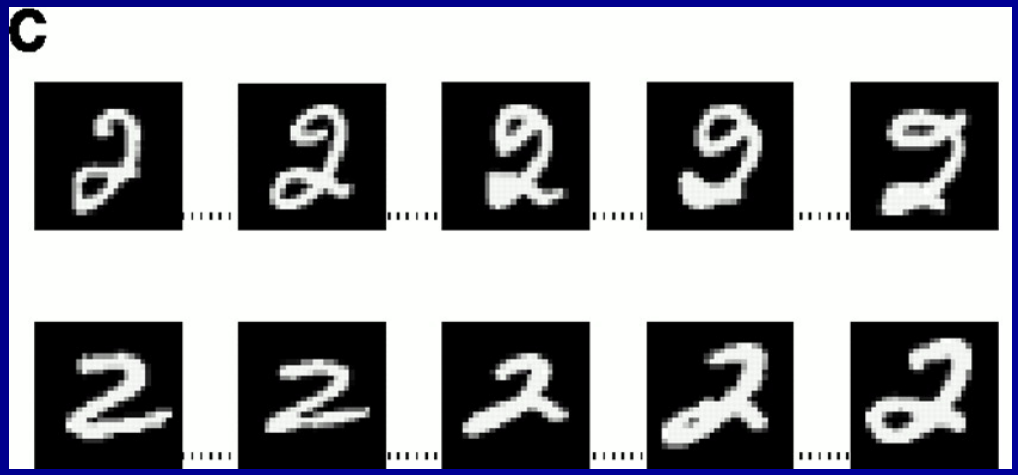
- Two more ISOMAP examples



Linear Interpolation Between Points in ISOMAP Space

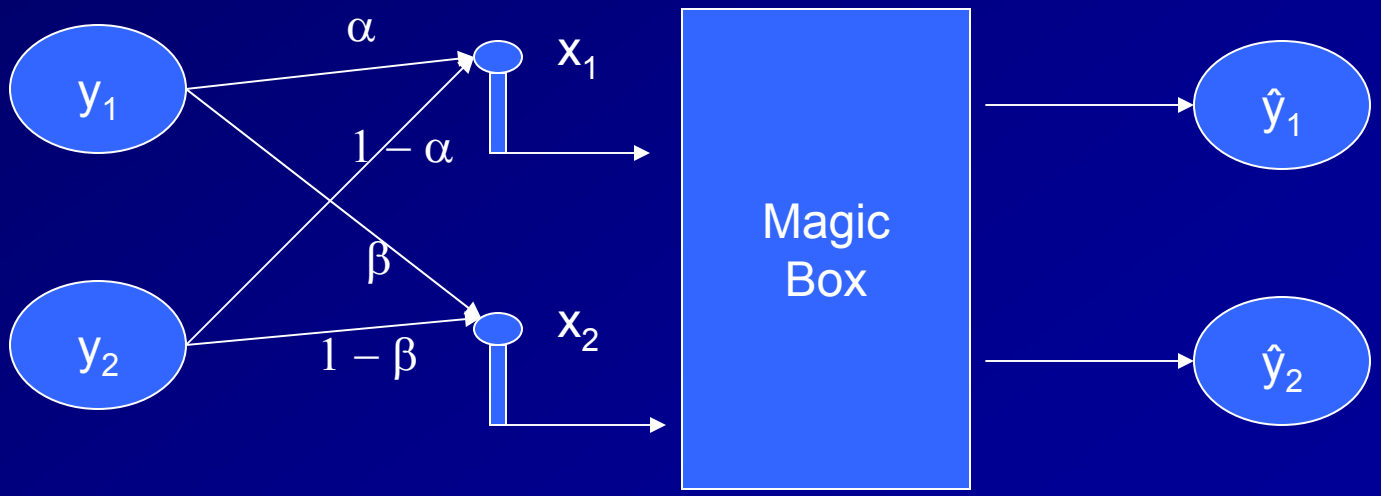
- Algorithm generates new poses

- and new 2's



Blind Source Separation

- Suppose we have two sound sources that have been linearly mixed and recorded by two microphones. Given the two microphone signals, we want to recover the two sound sources



Minimizing Mutual Information

- If the input sources are independent, then they should have zero mutual information.
- Idea: Minimize the mutual information between the outputs while maximizing the information (entropy) of each output separately:

  $\max_W \; H(\hat{y}_1) + H(\hat{y}_2) - I(\hat{y}_1; \hat{y}_2)$

  where $[\hat{y}_1, \hat{y}] = F_W(x_1, x_2)$

  and F_W is a sigmoid neural network

Independent Component Analysis (ICA)

- Microphone 1
- Microphone 2
- Reconstructed source 1
- Reconstructed source 2

source: http://www.cnl.salk.edu/~tewon/Blind/blind_audio.html

Unsupervised Learning Summary

- Density Estimation: Learn P(X) given training data for X
  - Mixture models and EM
- Clustering: Partition data into clusters
  - Bottom up aggomerative clustering
- Dimensionality Reduction: Discover low-dimensional representation of data
  - Principal Component Analysis
  - ISOMAP
- Blind Source Separation: Unmixing multiple signals
  - Many algorithms

Objective Functions

- Density Estimation:
  - Log likelihood on training data
- Clustering:
  - ????
- Dimensionality Reduction
  - Minimum reconstruction error
  - Maximum likelihood (gaussian interpretation of PCA)
- Blind Source Separation
  - Information Maximization
  - Maximum Likelihood (assuming models of the sources)