Machine Learning

CUNY Graduate Center, Spring 2013

Lectures 11-12: Unsupervised Learning I

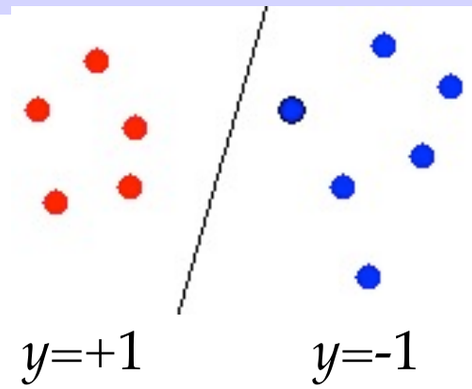
(Clustering: *k*-means, EM, mixture models)

Professor Liang Huang

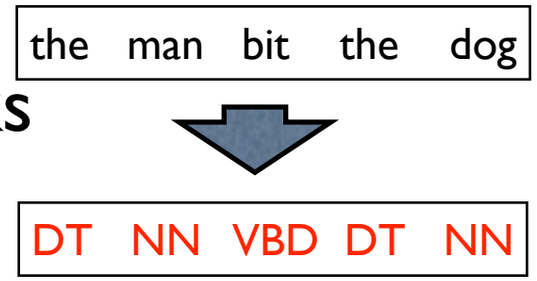
huang@cs.qc.cuny.edu

http://acl.cs.qc.edu/~lhuang/teaching/machine-learning

Roadmap



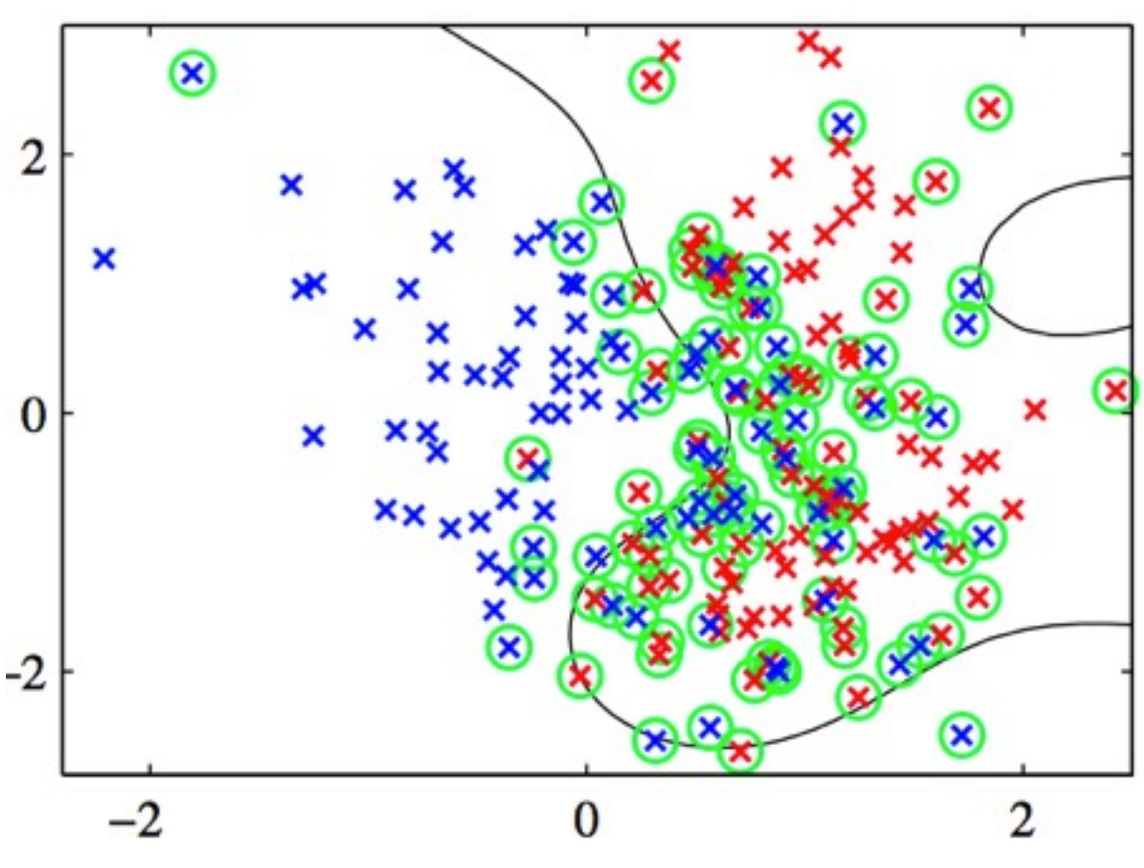
- so far: (large-margin) supervised learning
  - binary, multiclass, and structured classifications
  - online learning: avg perceptron/MIRA, convergence proof
  - kernels and kernelized perceptron in dual
  - SVMs: formulation, KKT, dual, convex optimization, slacks

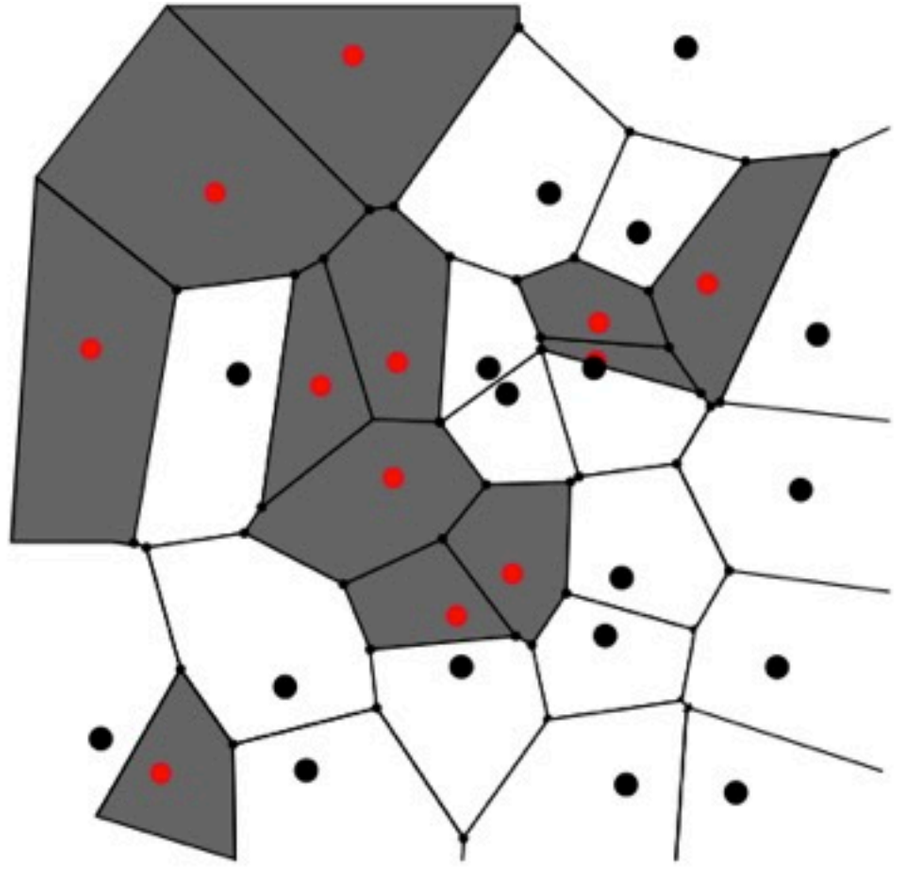


  - structured perceptron, HMM, and Viterbi algorithm
- what we left out: many classical algorithms
  - nearest neighbors (instance-based), decision trees, logistic regression...
- next up: unsupervised learning
  - clustering: k-means, EM, mixture models, hierarchical
  - dimensionality reduction: linear (PCA/ICA, MDS), nonlinear (isomap)

Sup=>Unsup: Nearest Neighbor=> k-means

- let's look at a supervised learning method: nearest neighbor
  - SVM, perceptron (in dual) and NN are all instance-based learning
  - instance-based learning: store a subset of examples for classification
    - compression rate: SVM: very high, perceptron: medium high, NN: 0





k-Nearest Neighbor

- one way to prevent overfitting => more stable results

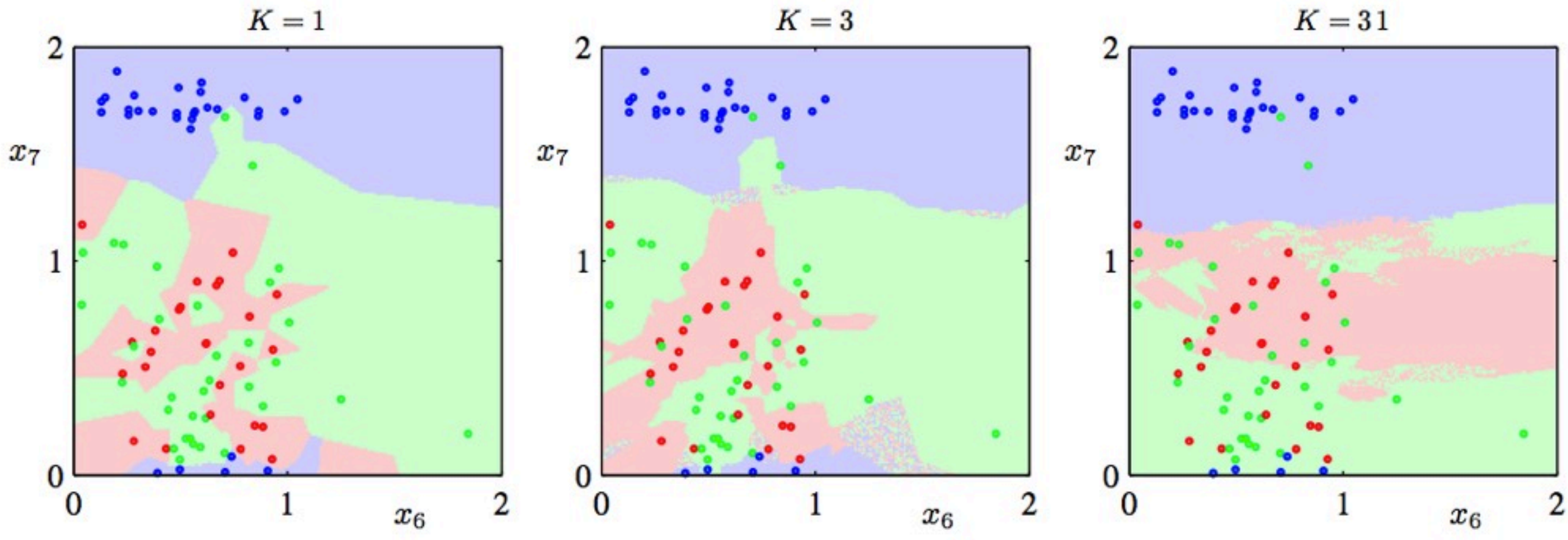
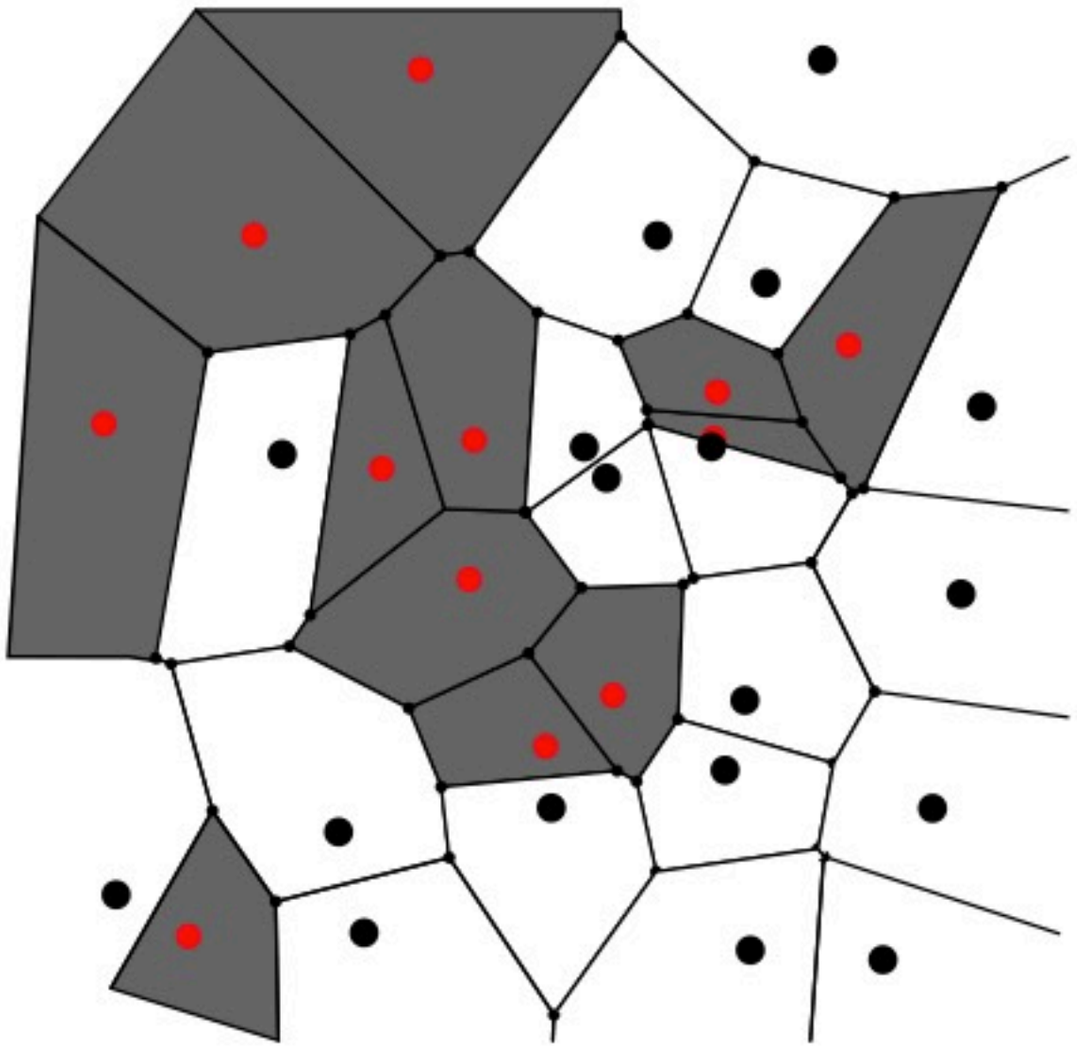
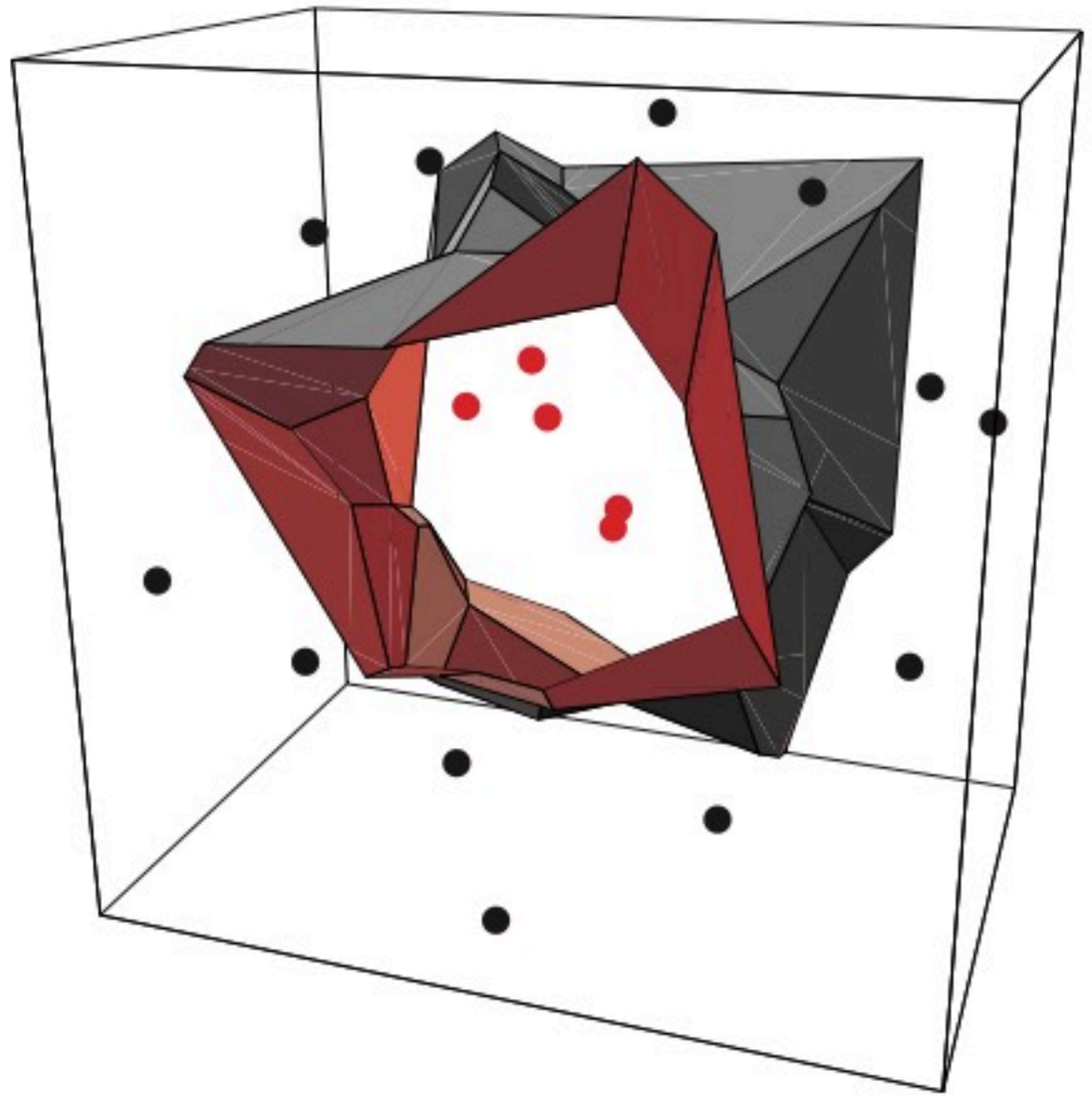


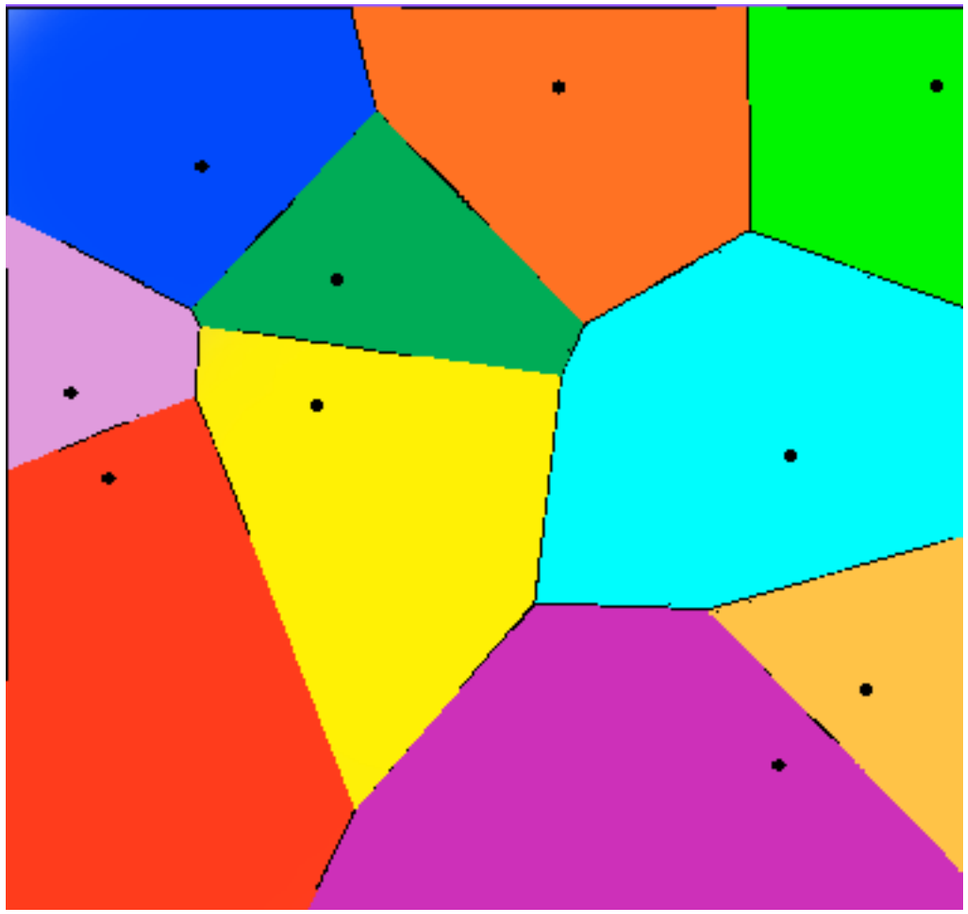
Figure 2.28 Plot of 200 data points from the oil data set showing values of x_6 plotted against x_7, where the red, green, and blue points correspond to the 'laminar', 'annular', and 'homogeneous' classes, respectively. Also shown are the classifications of the input space given by the K-nearest-neighbour algorithm for various values of K.

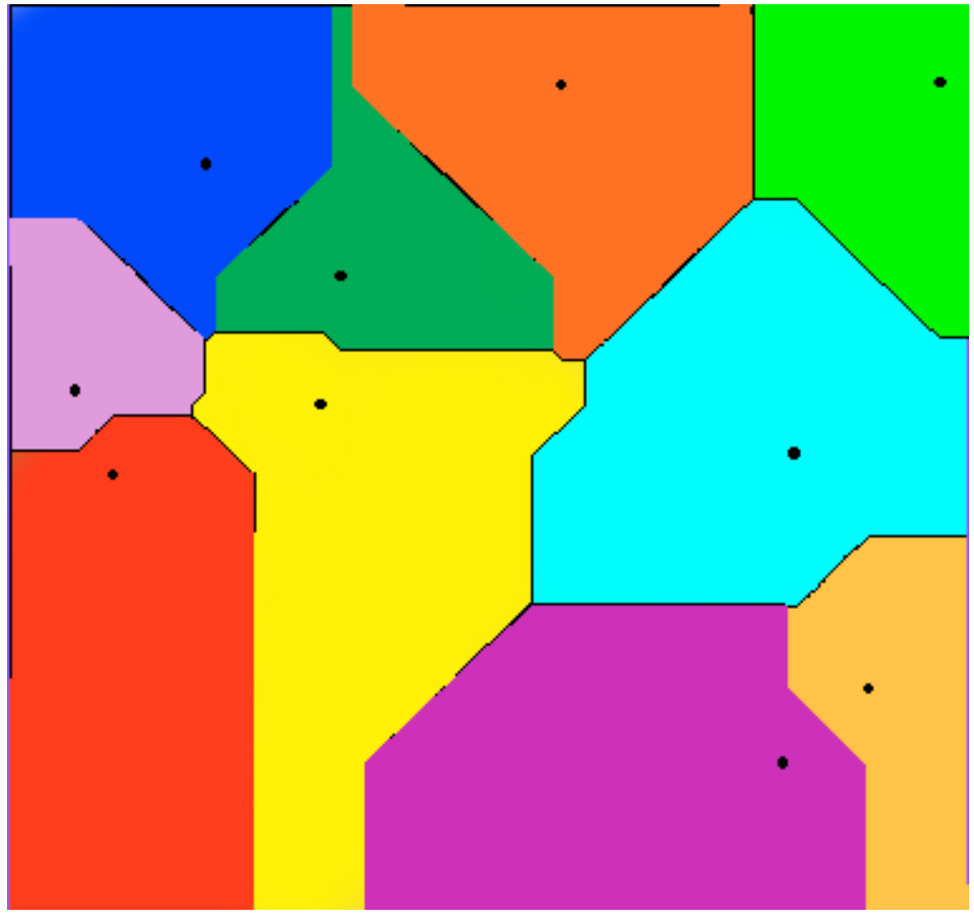
NN Voronoi in 2D and 3D





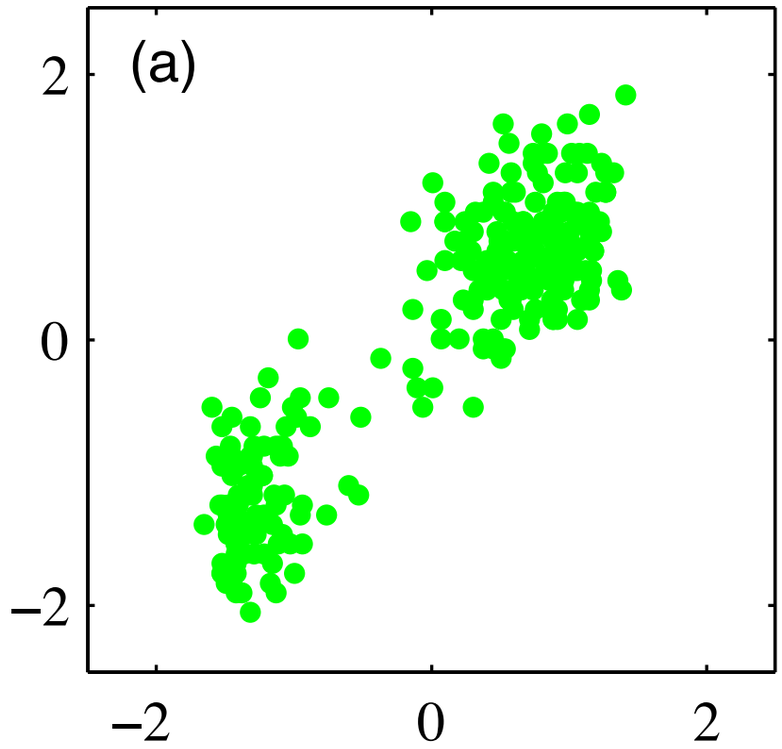
Voronoi for Euclidian and Manhattan





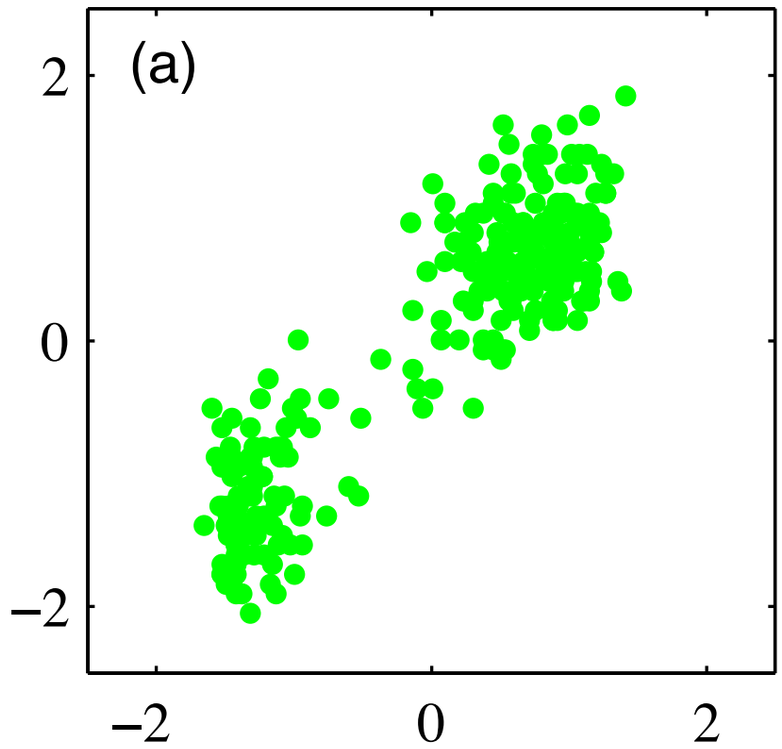
Unsupervised Learning

- cost of supervised learning
  - labeled data: expensive to annotate!
  - but there exists huge data w/o labels
- unsupervised learning
  - can only hallucinate the labels
  - infer some "internal structures" of data
    - still the "compression" view of learning
    - too much data => reduce it!
    - clustering: reduce # of examples
    - dimensionality reduction: reduce # of dimensions



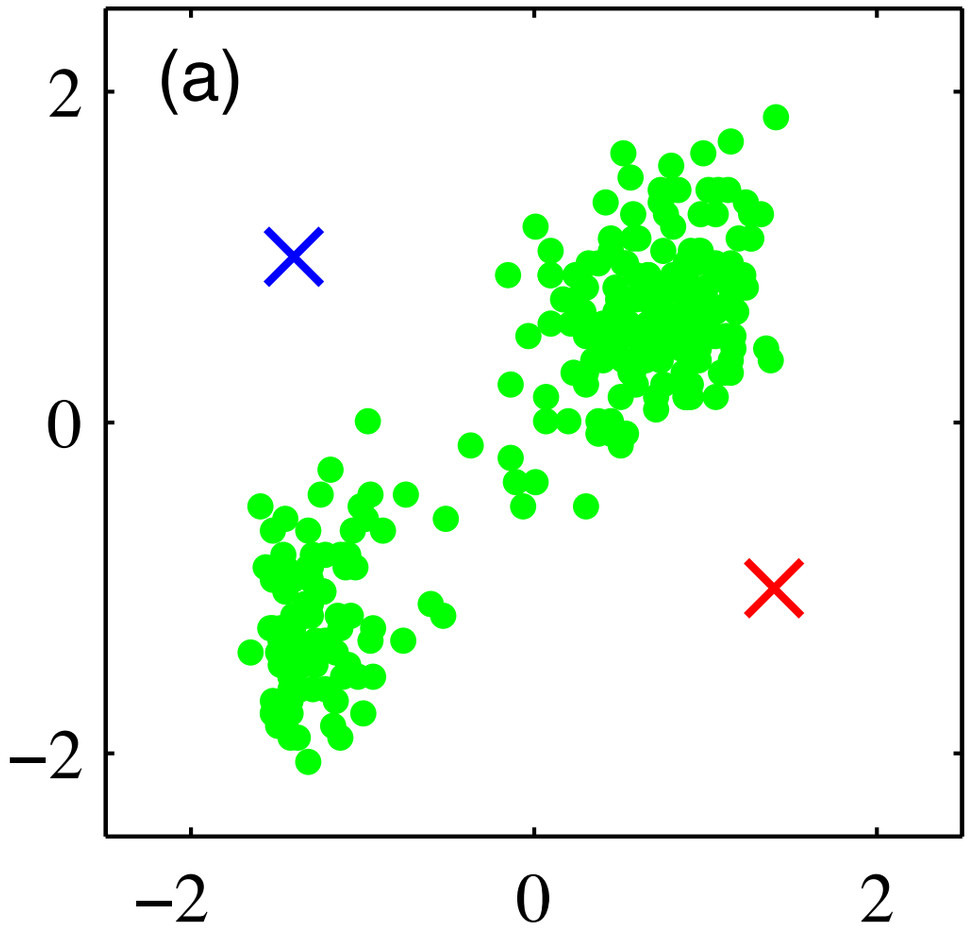
Challenges in Unsupervised Learning

- how to evaluate the results?
  - there is no gold standard data!
  - internal metric?
- how to interpret the results?
  - how to "name" the clusters?
- how to initialize the model/guess?
  - a bad initial guess can lead to very bad results
  - unsup is very sensitive to initialization (unlike supervised)
- how to do optimization => in general no longer convex!



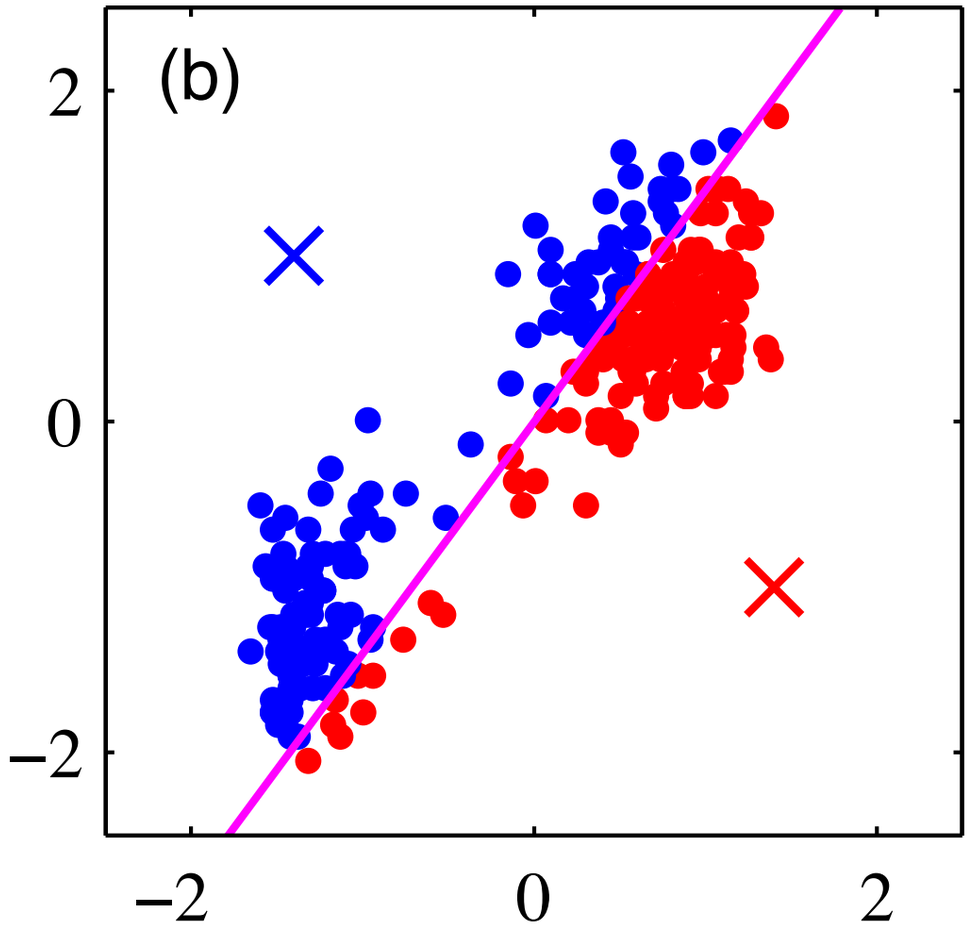
k-means

- (randomly) pick *k* points to be initial centroids
- repeat the two steps until convergence
  - assignment to centroids: voronoi, like NN
  - recomputation of centroids based on the new assignment



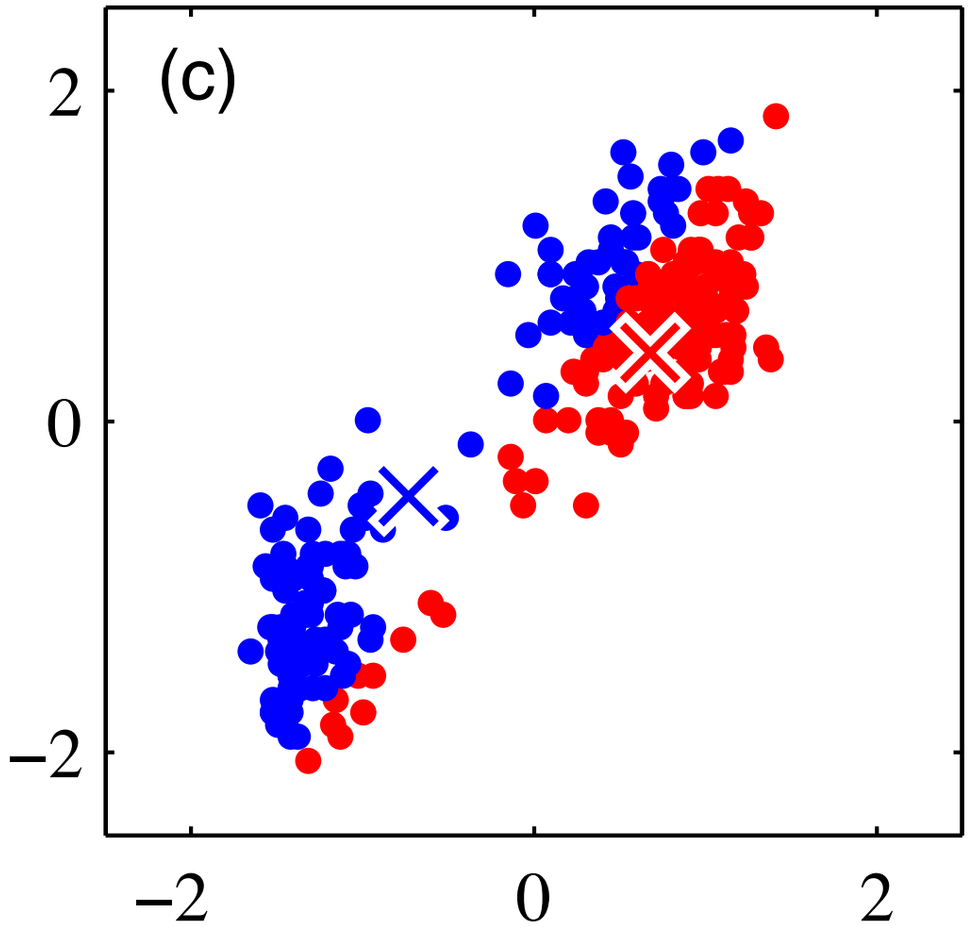
k-means

- (randomly) pick *k* points to be initial centroids
- repeat the two steps until convergence
  - assignment to centroids: voronoi, like NN
  - recomputation of centroids based on the new assignment



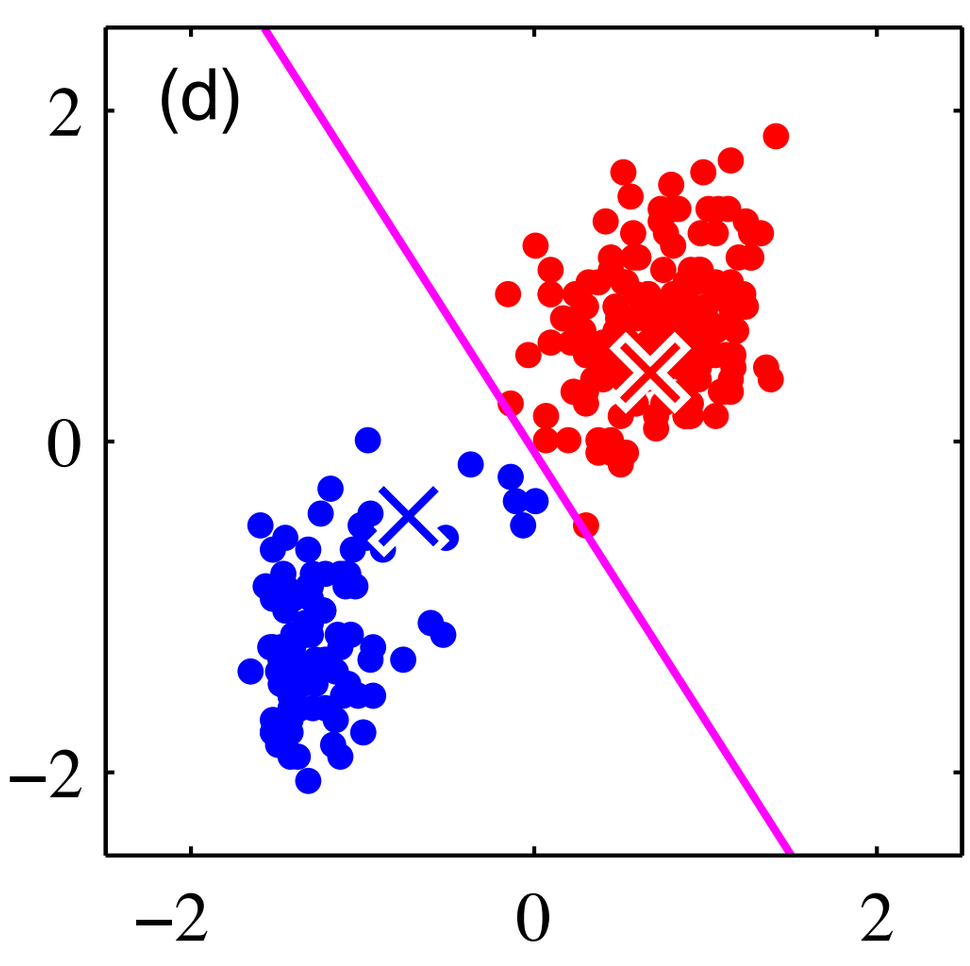
k-means

- (randomly) pick *k* points to be initial centroids
- repeat the two steps until convergence
  - assignment to centroids: voronoi, like NN
  - recomputation of centroids based on the new assignment



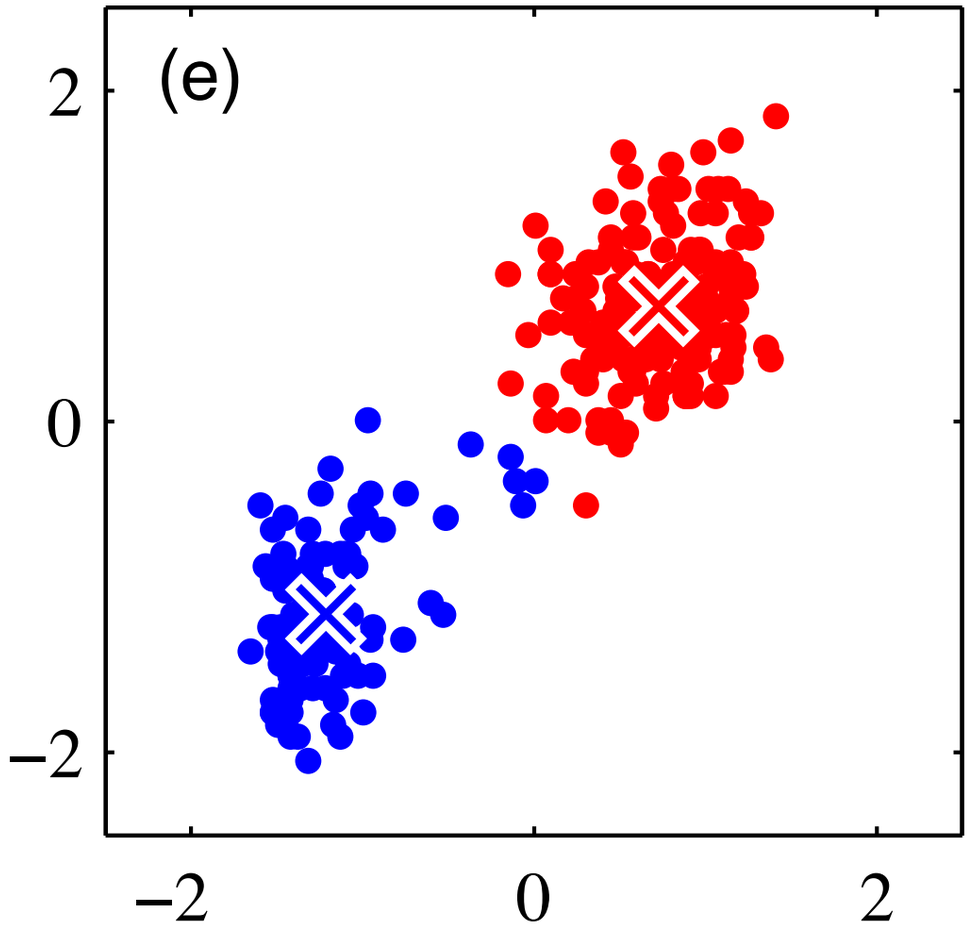
k-means

- (randomly) pick *k* points to be initial centroids
- repeat the two steps until convergence
  - assignment to centroids: voronoi, like NN
  - recomputation of centroids based on the new assignment



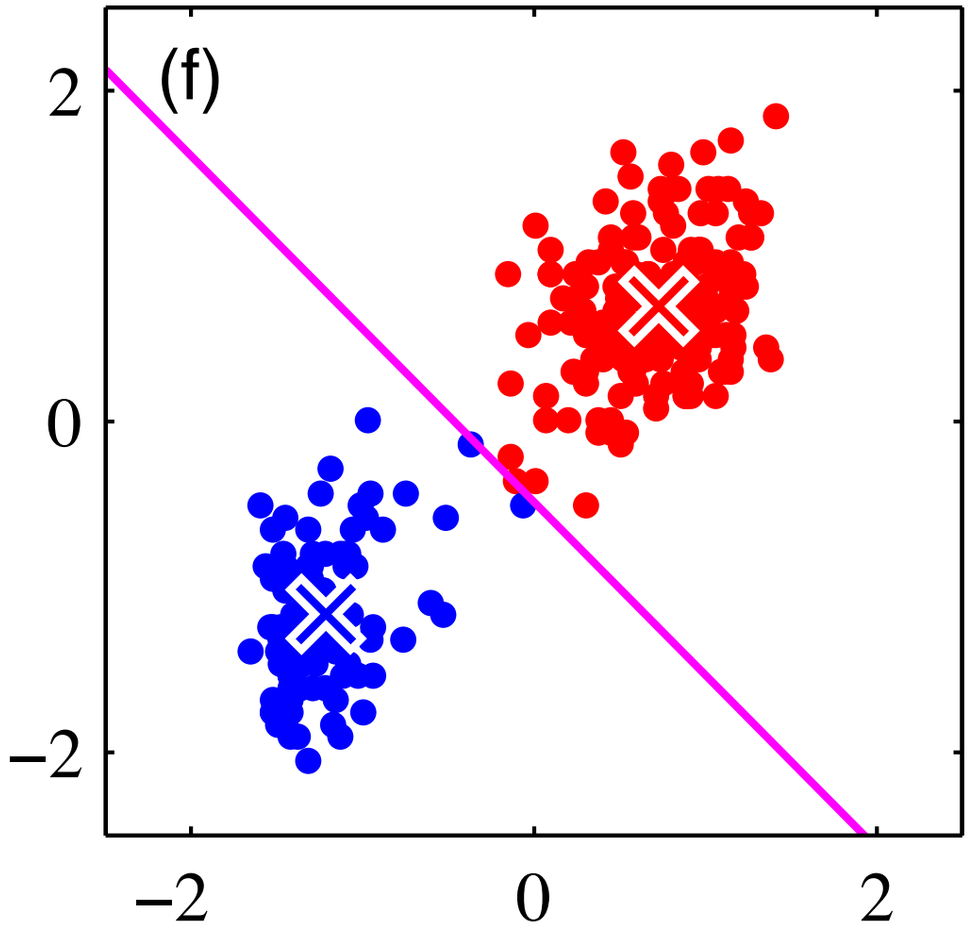
k-means

- (randomly) pick *k* points to be initial centroids
- repeat the two steps until convergence
  - assignment to centroids: voronoi, like NN
  - recomputation of centroids based on the new assignment



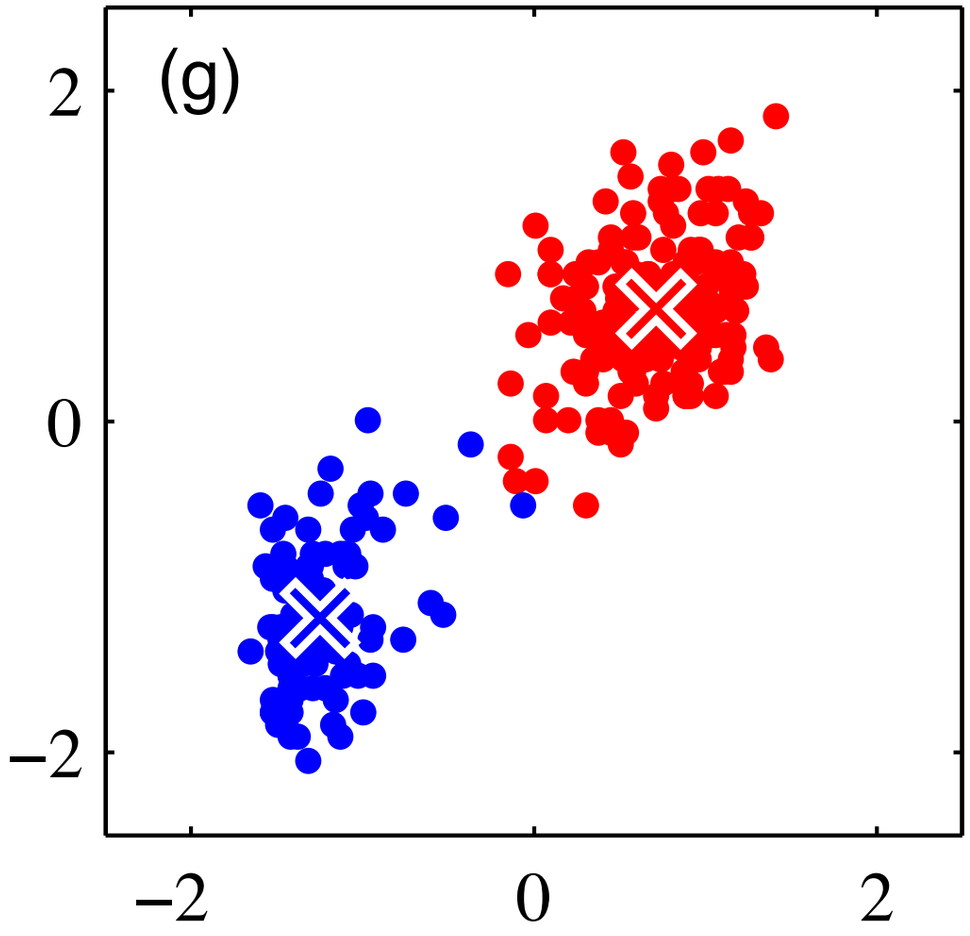
k-means

- (randomly) pick *k* points to be initial centroids
- repeat the two steps until convergence
  - assignment to centroids: voronoi, like NN
  - recomputation of centroids based on the new assignment



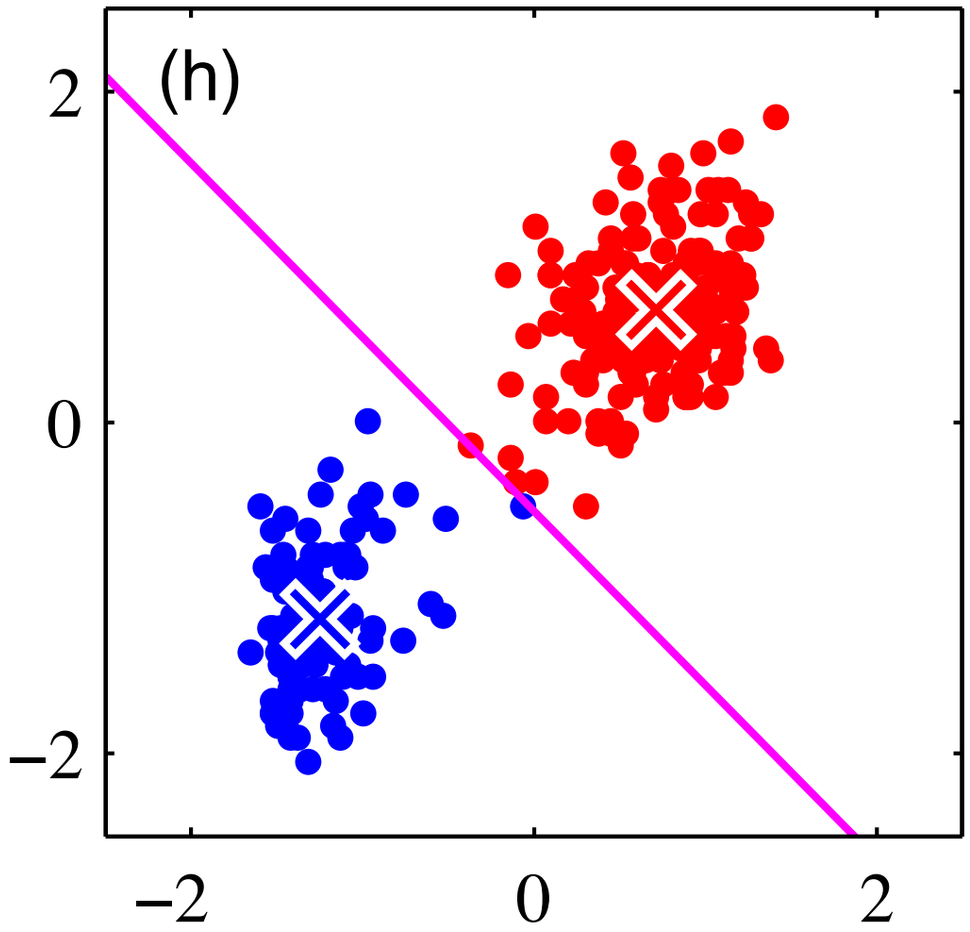
k-means

- (randomly) pick *k* points to be initial centroids
- repeat the two steps until convergence
  - assignment to centroids: voronoi, like NN
  - recomputation of centroids based on the new assignment



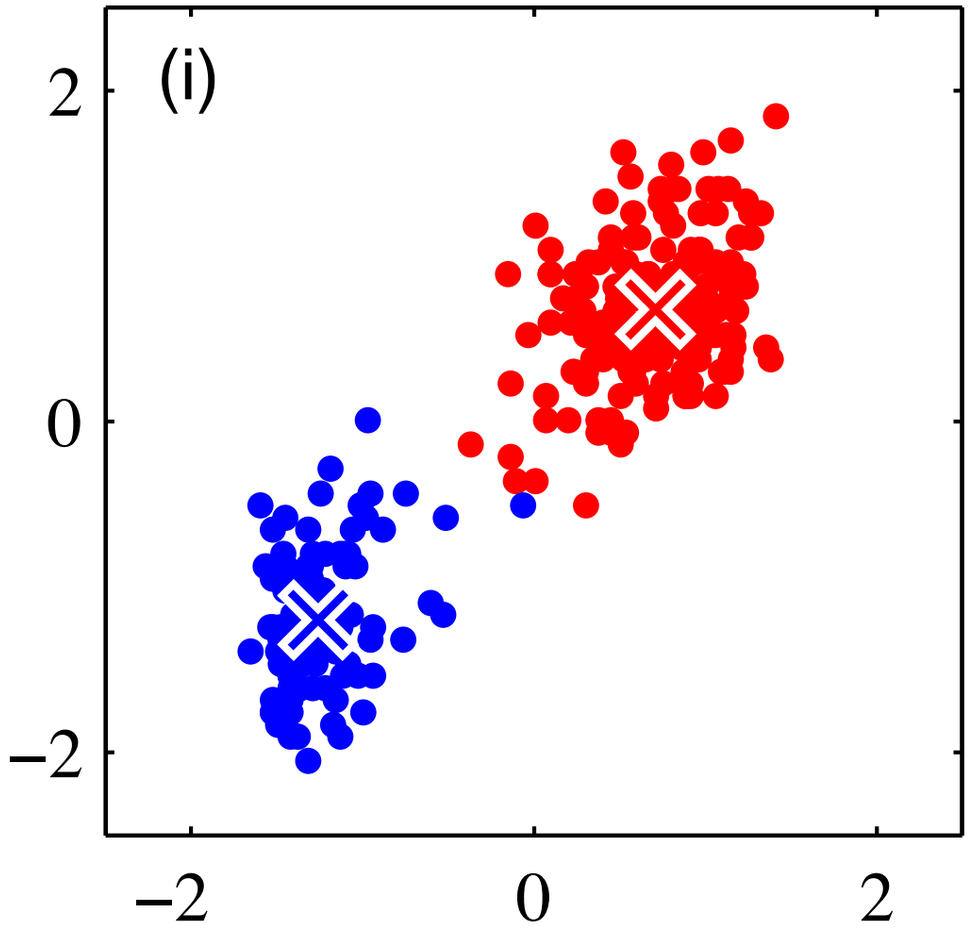
k-means

- (randomly) pick *k* points to be initial centroids
- repeat the two steps until convergence
  - assignment to centroids: voronoi, like NN
  - recomputation of centroids based on the new assignment



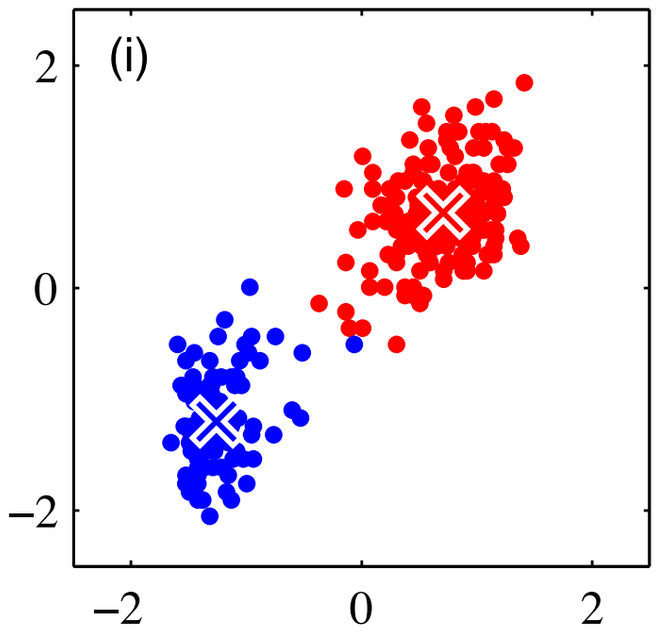
k-means

- (randomly) pick *k* points to be initial centroids
- repeat the two steps until convergence
  - assignment to centroids: voronoi, like NN
  - recomputation of centroids based on the new assignment



k-means

- (randomly) pick *k* points to be initial centroids
- repeat the two steps until convergence
  - assignment to centroids: voronoi, like NN
  - recomputation of centroids based on the new assignment
- how to define convergence?
  - after a fixed number of iterations, or
  - assignments do not change, or
  - centroids do not change (equivalent?) or
  - change in objective function falls below threshold

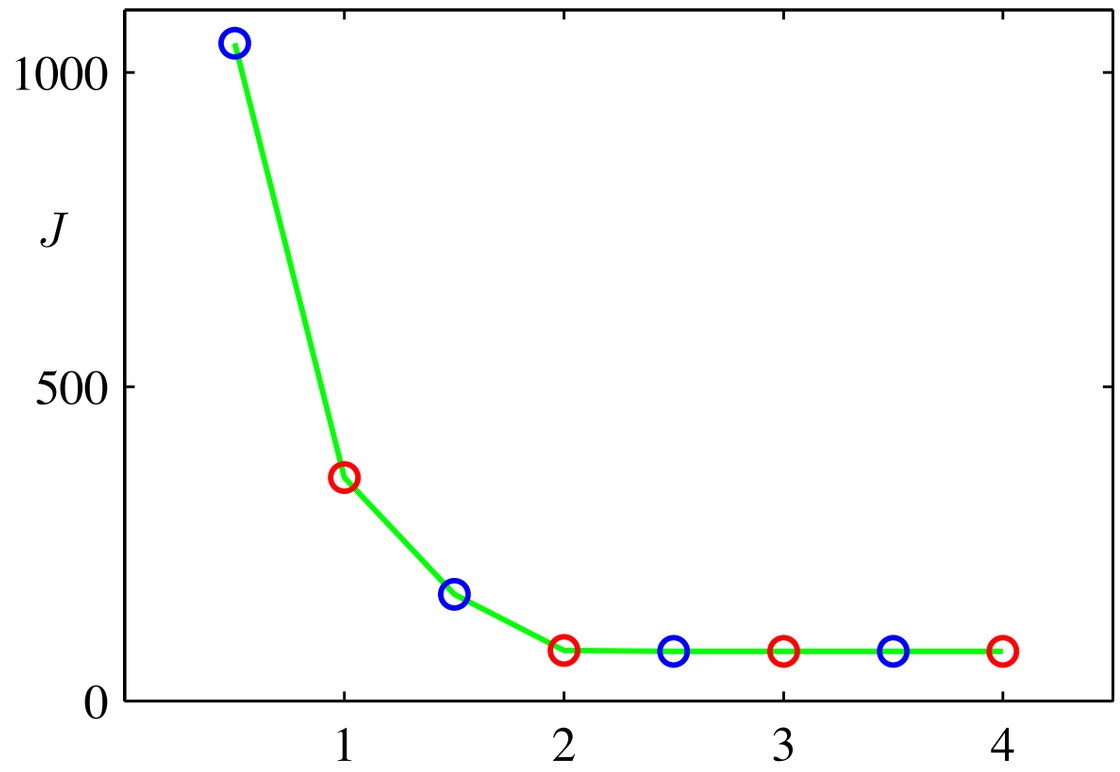


k-means objective function

- residual sum of squares (RSS)
  - sum of distances from points to their centroids
  - guaranteed to decrease monotonically
  - convergence proof: decrease + finite # of clusterings

$$\mathrm{RSS}_k = \sum_{\vec{x} \in \omega_k} |\vec{x} - \vec{\mu}(\omega_k)|^2$$

$$\mathrm{RSS} = \sum_{k=1}^{K} \mathrm{RSS}_k$$



k-means for image segmentation

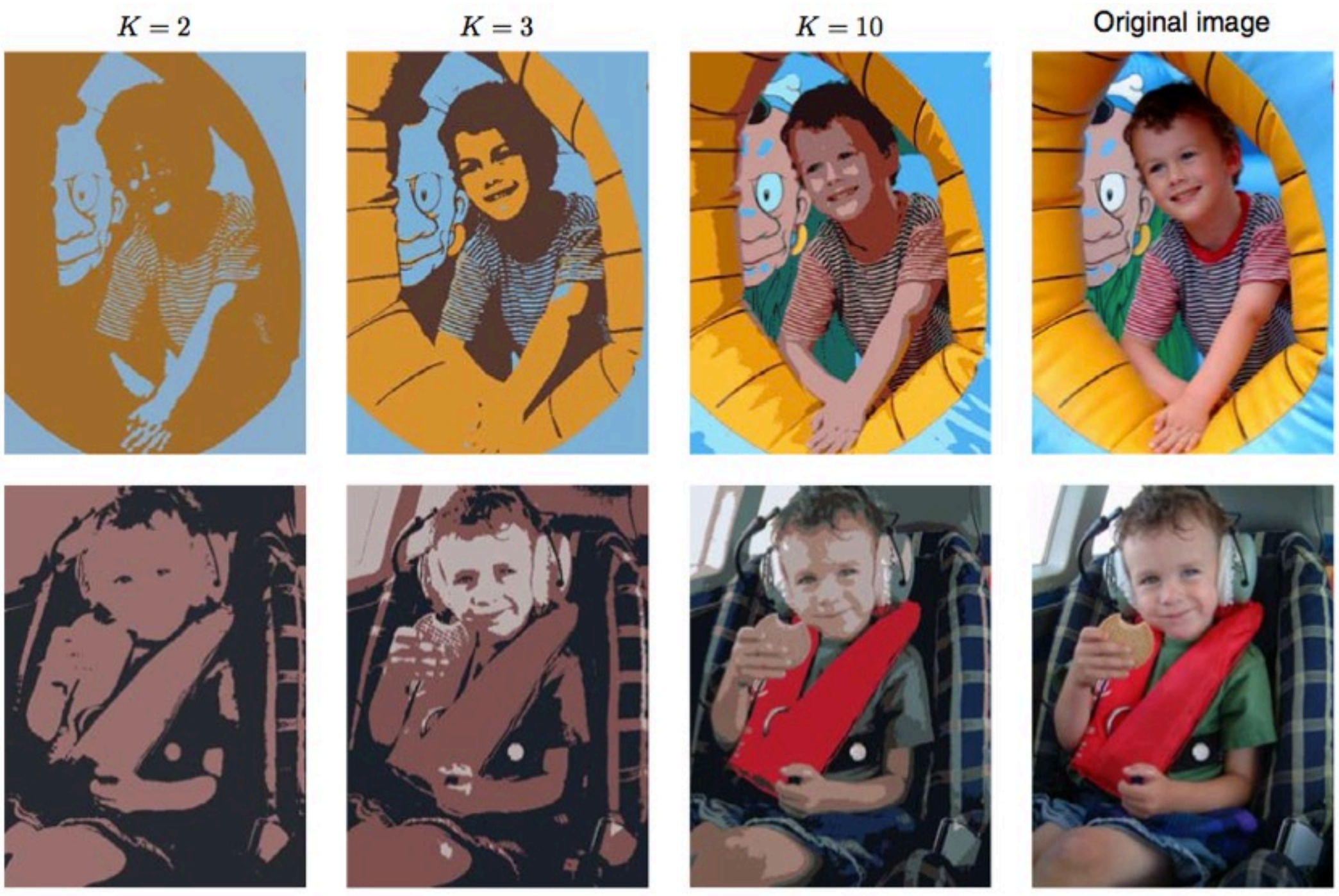


Figure 9.3 Two examples of the application of the K-means clustering algorithm to image segmentation showing the initial images together with their K-means segmentations obtained using various values of K. This also illustrates of the use of vector quantization for data compression, in which smaller values of K give higher compression at the expense of poorer image quality.

Problems with *k*-means

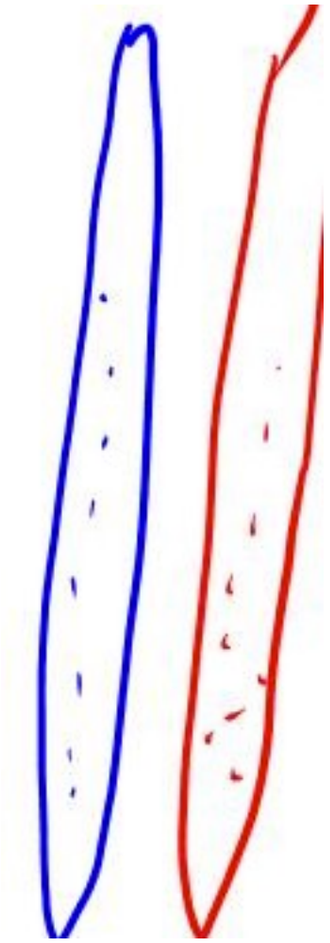
- problem: sensitive to initialization
  - the objective function is non-convex: many local minima
  - why?

$$\mathrm{RSS}_k = \sum_{\vec{x} \in \omega_k} |\vec{x} - \vec{\mu}(\omega_k)|^2$$

- k-means works well if
  - clusters are spherical

$$\mathrm{RSS} = \sum_{k=1}^{K} \mathrm{RSS}_k$$

  - clusters are well separated
  - clusters of similar volumes
  - clusters have similar # of examples



Problems with *k*-means

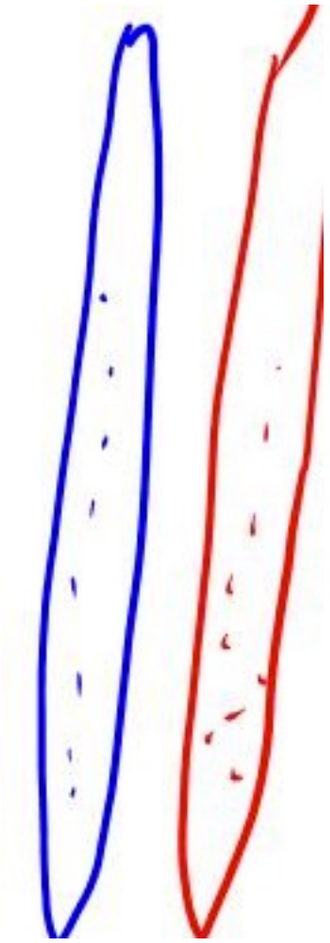
- problem: sensitive to initialization
  - the objective function is non-convex: many local minima
  - why?

$$\mathrm{RSS}_k = \sum_{\vec{x} \in \omega_k} |\vec{x} - \vec{\mu}(\omega_k)|^2 \qquad = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

- k-means works well if
  - clusters are spherical

$$\mathrm{RSS} = \sum_{k=1}^{K} \mathrm{RSS}_k$$

  - clusters are well separated
  - clusters of similar volumes
  - clusters have similar # of examples



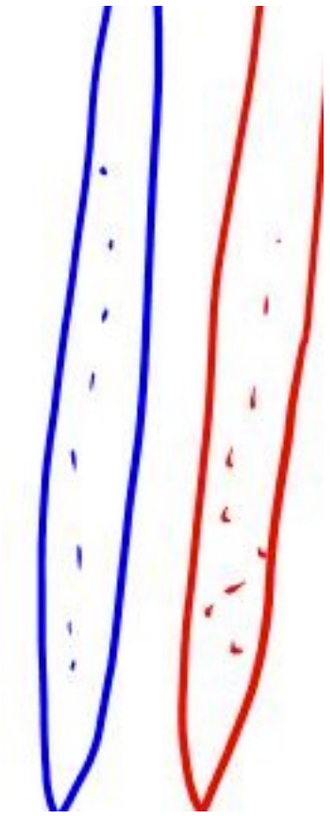
Problems with *k*-means

- problem: sensitive to initialization
  - the objective function is non-convex: many local minima
  - why?
- k-means works well if
  - clusters are spherical
  - clusters are well separated
  - clusters of similar volumes
  - clusters have similar # of examples

$$\mathrm{RSS}_k = \sum_{\vec{x} \in \omega_k} |\vec{x} - \vec{\mu}(\omega_k)|^2 \qquad = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

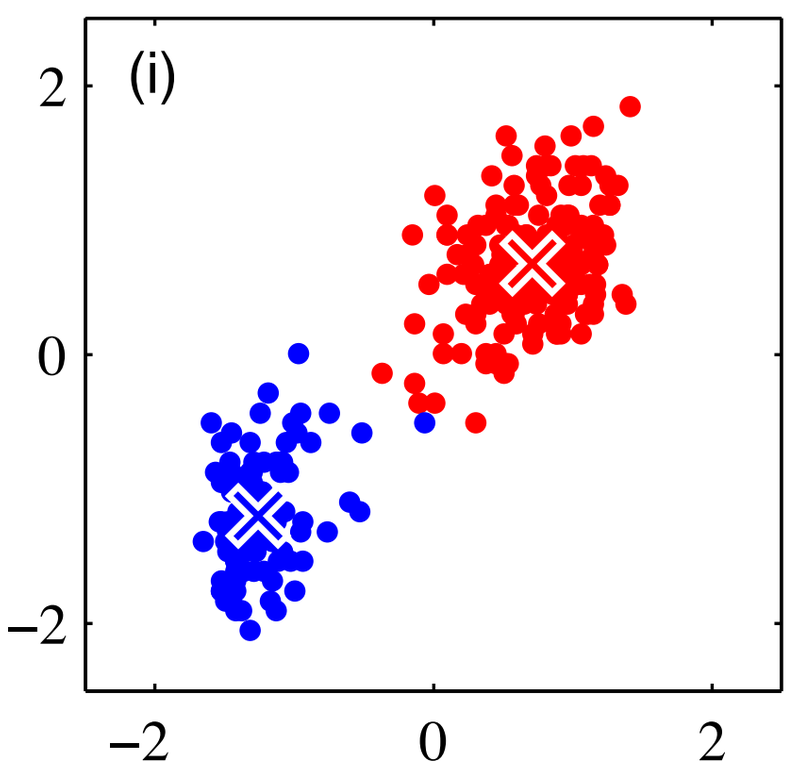
$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2 \\ 0 & \text{otherwise.} \end{cases}$$

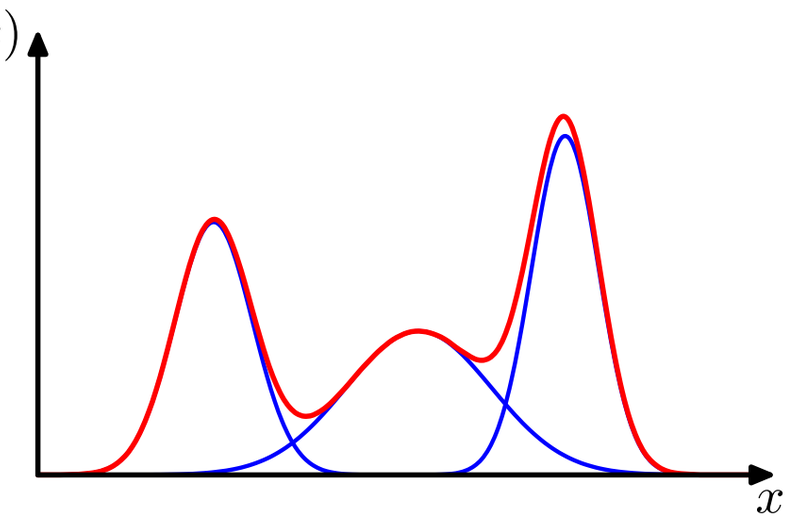
$$\mathrm{RSS} = \sum_{k=1}^{K} \mathrm{RSS}_k$$

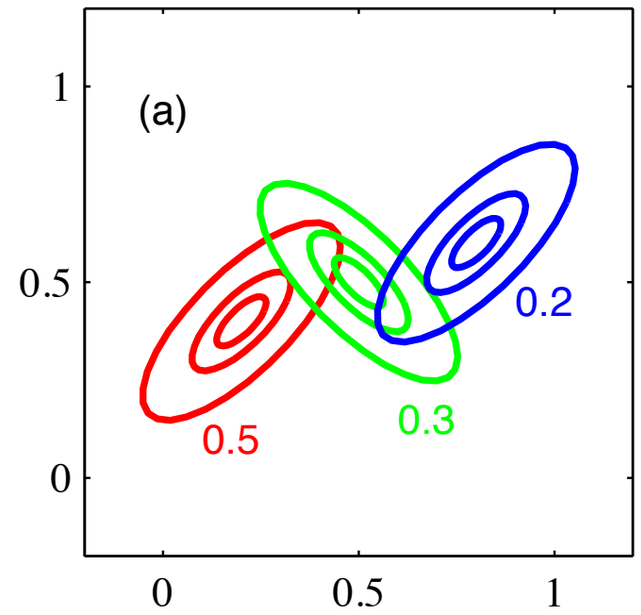


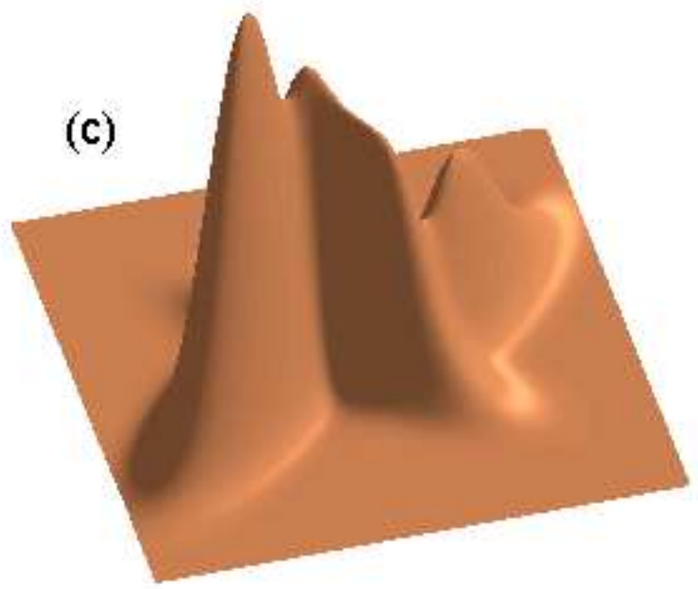
Better ("soft") *k*-means?

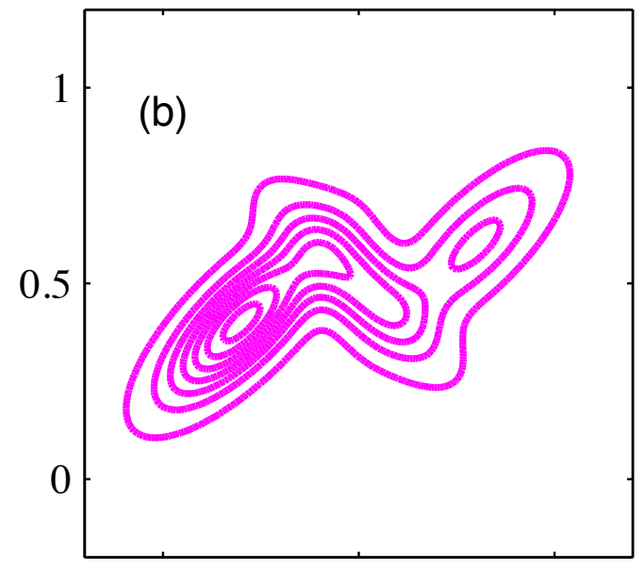
- random restarts -- definitely helps
- soft clusters => EM with Gaussian Mixture Model





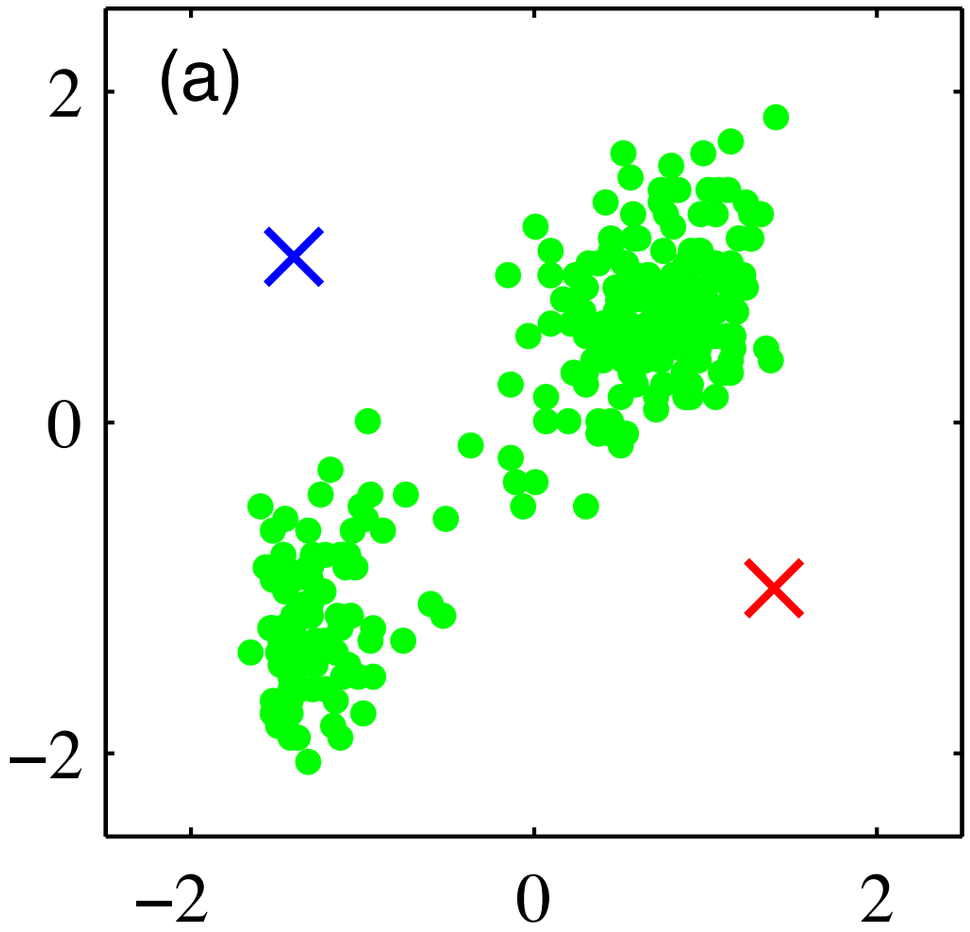






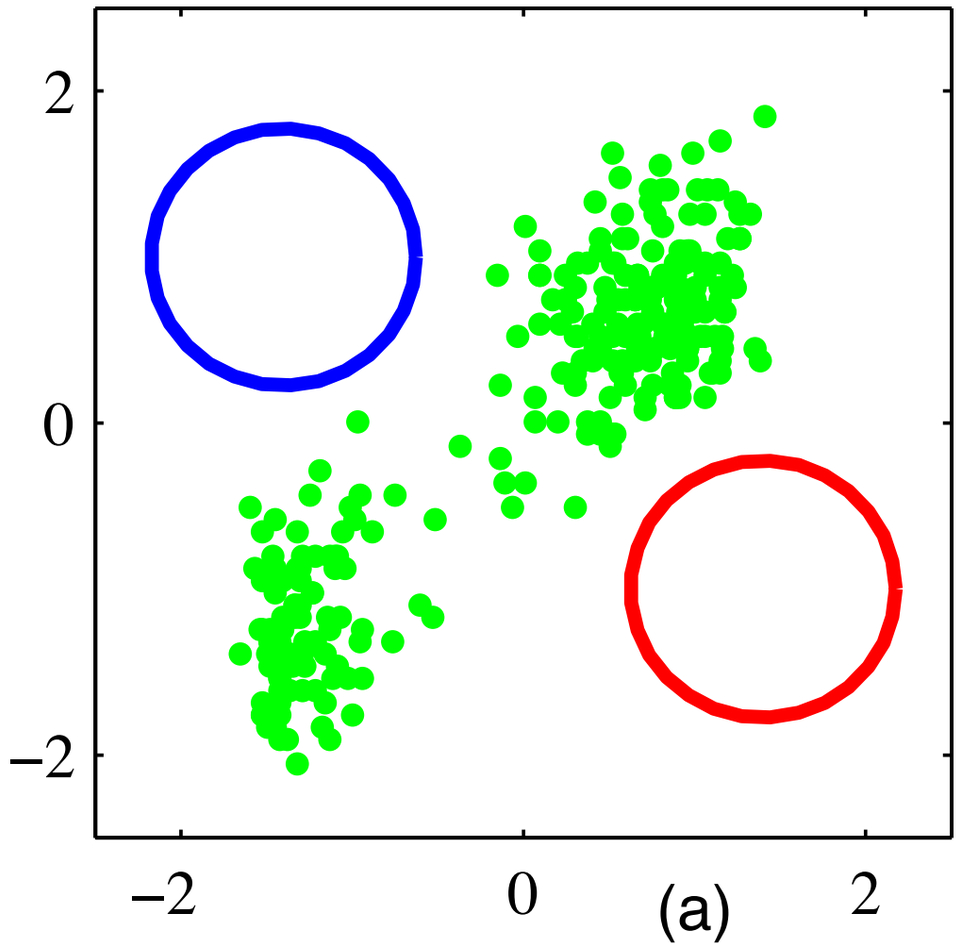
k-means

- randomize *k* initial centroids
- repeat the two steps until convergence
  - E-step: assignment each example to centroids (Voronoi)
  - M-step: recomputation of centroids (based on the new assignment)



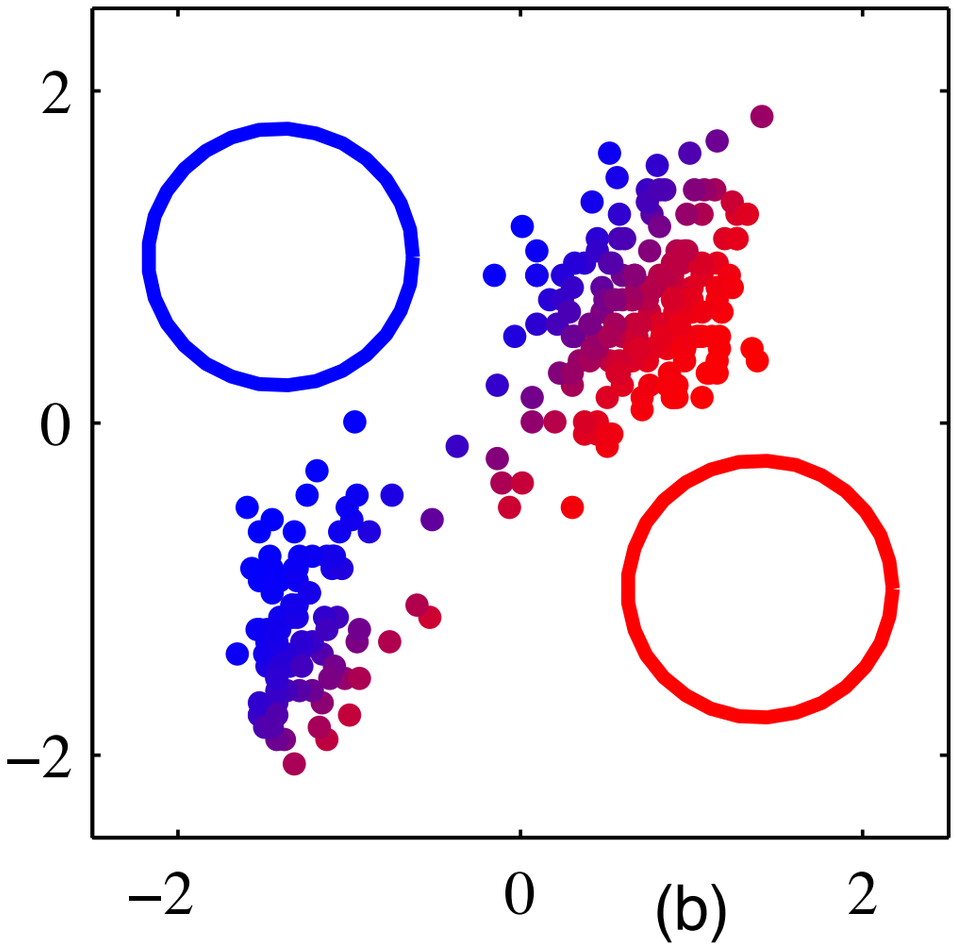
EM for Gaussian Mixtures

- randomize *k* means, covariances, mixing coefficients
- repeat the two steps until convergence
  - E-step: evaluate the responsibilities using current parameters
  - M-step: reestimate parameters using current responsibilities



EM for Gaussian Mixtures

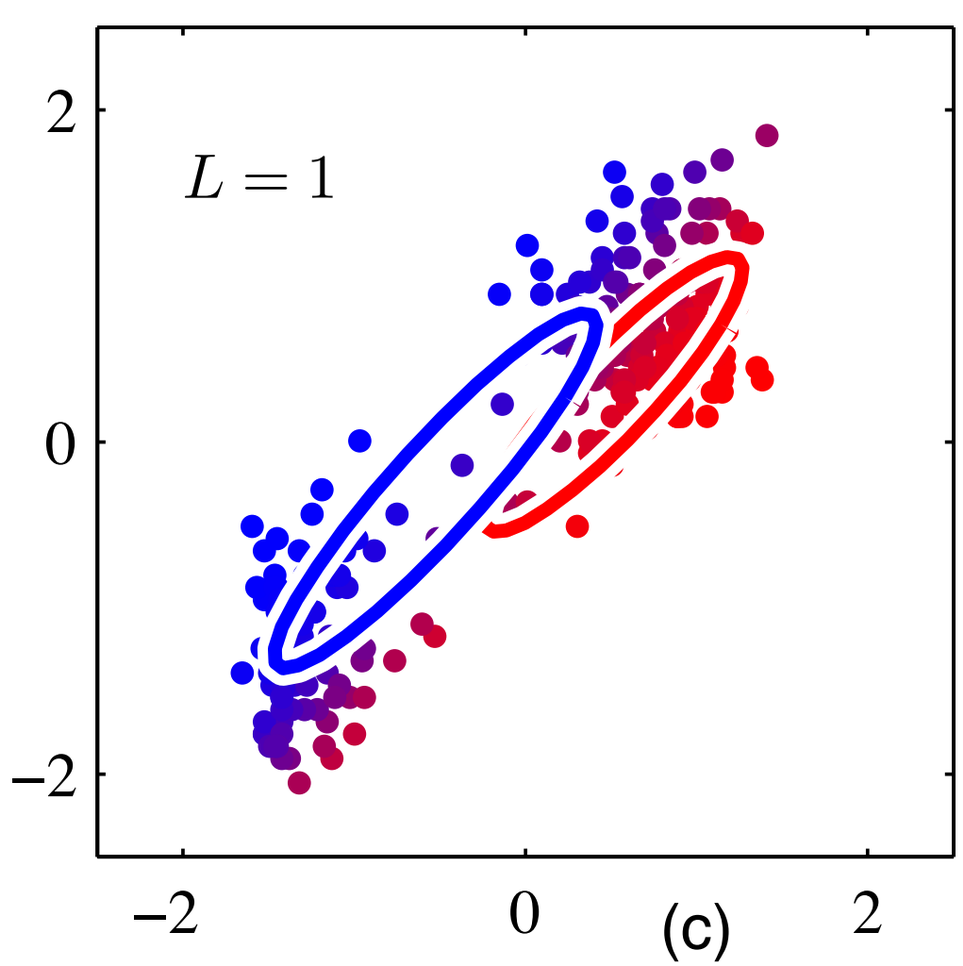
- randomize *k* means, covariances, mixing coefficients
- repeat the two steps until convergence
  - E-step: evaluate the responsibilities using current parameters
  - M-step: reestimate parameters using current responsibilities



"fractional assignments"

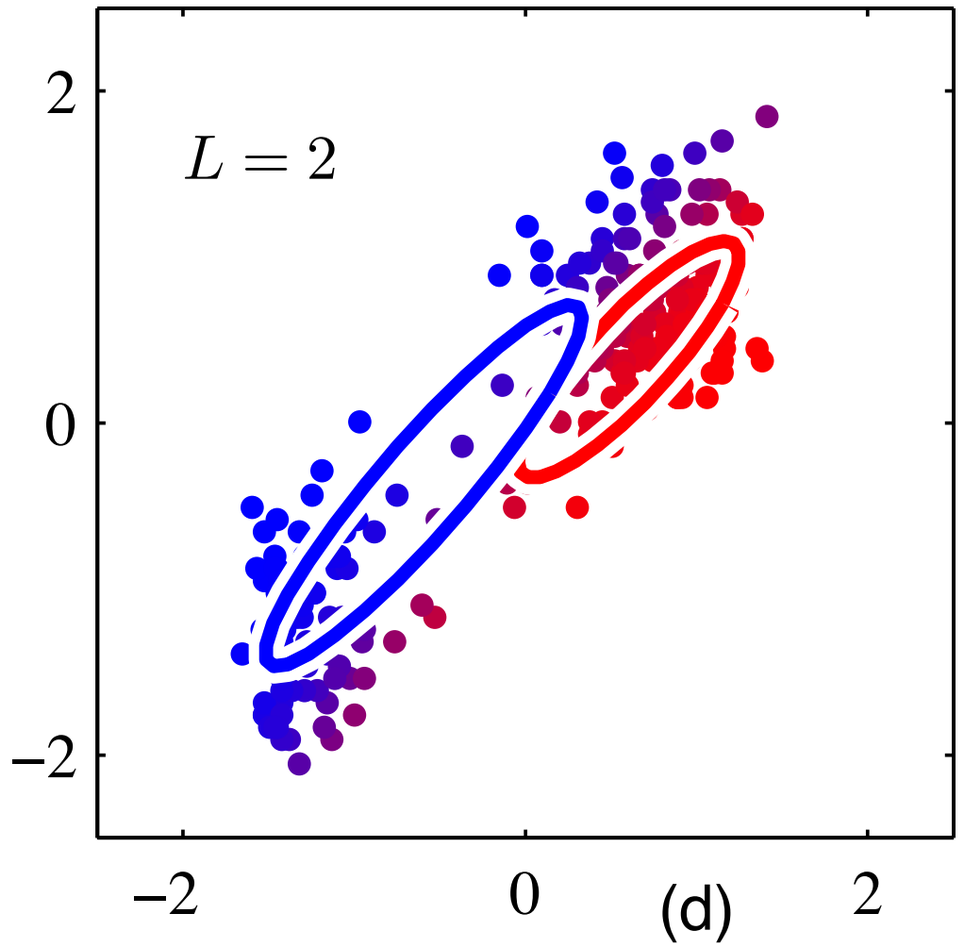
EM for Gaussian Mixtures

- randomize *k* means, covariances, mixing coefficients
- repeat the two steps until convergence
  - E-step: evaluate the responsibilities using current parameters
  - M-step: reestimate parameters using current responsibilities



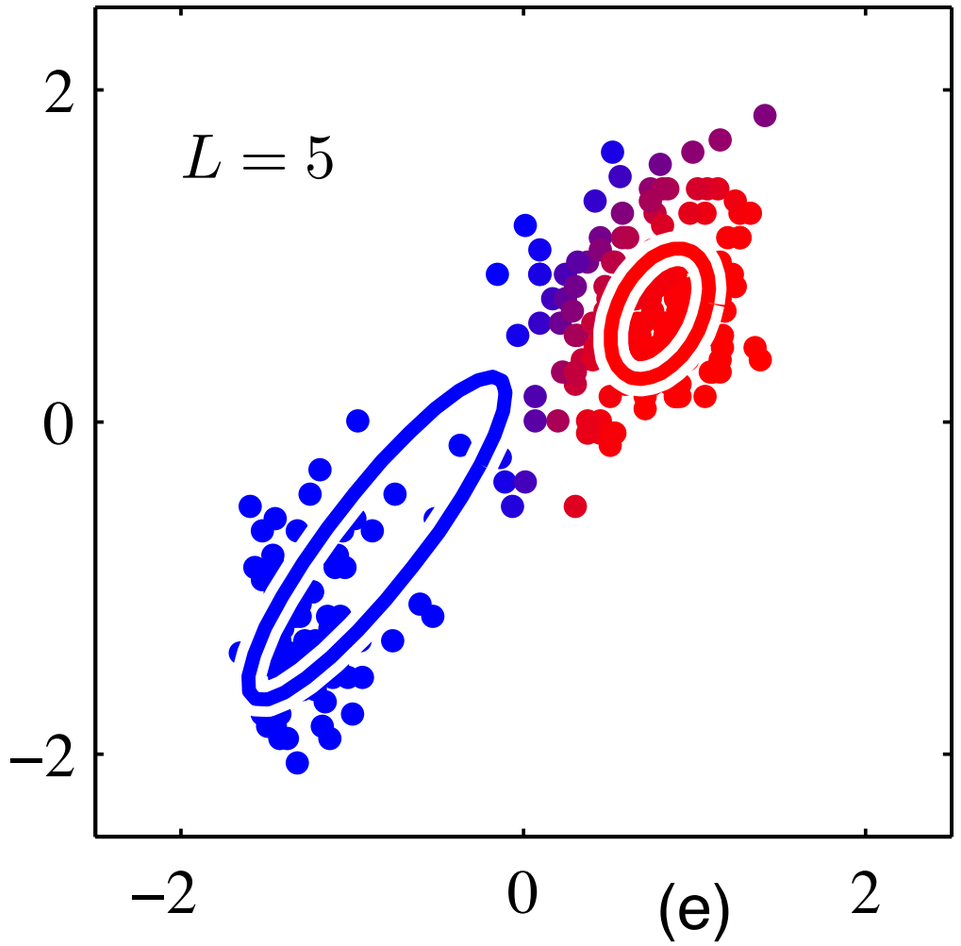
EM for Gaussian Mixtures

- randomize *k* means, covariances, mixing coefficients
- repeat the two steps until convergence
  - E-step: evaluate the responsibilities using current parameters
  - M-step: reestimate parameters using current responsibilities



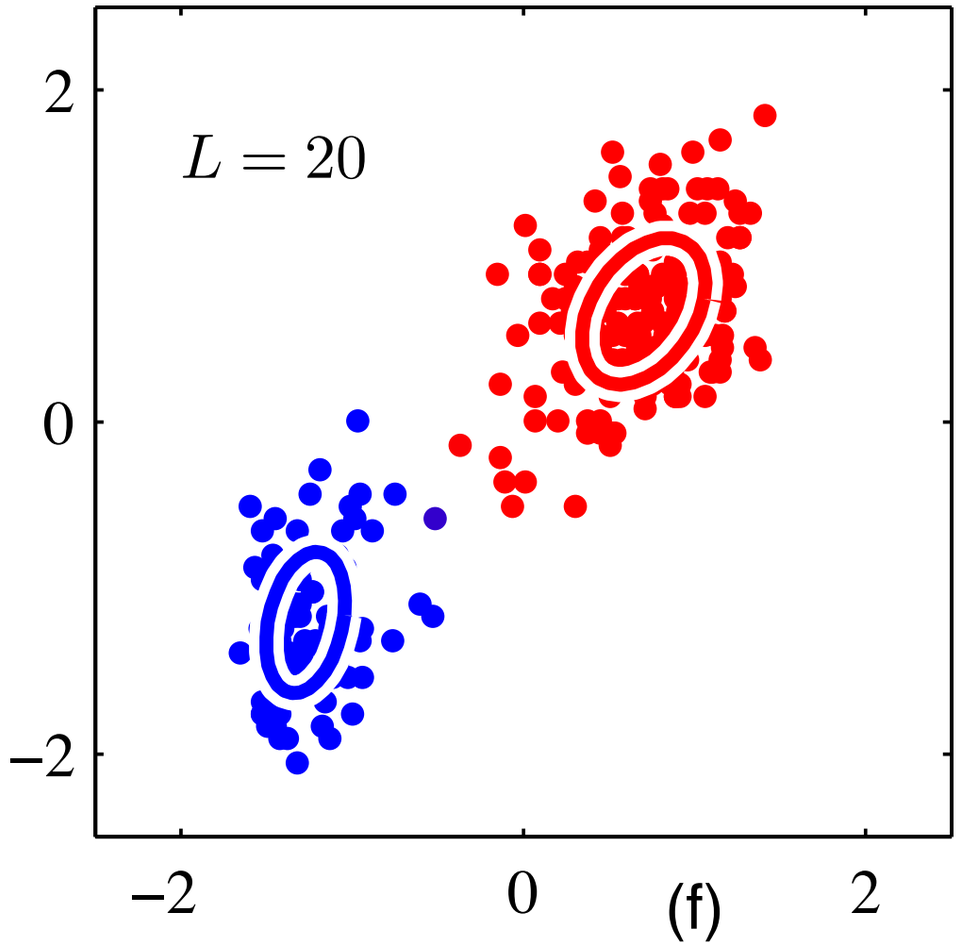
EM for Gaussian Mixtures

- randomize *k* means, covariances, mixing coefficients
- repeat the two steps until convergence
  - E-step: evaluate the responsibilities using current parameters
  - M-step: reestimate parameters using current responsibilities



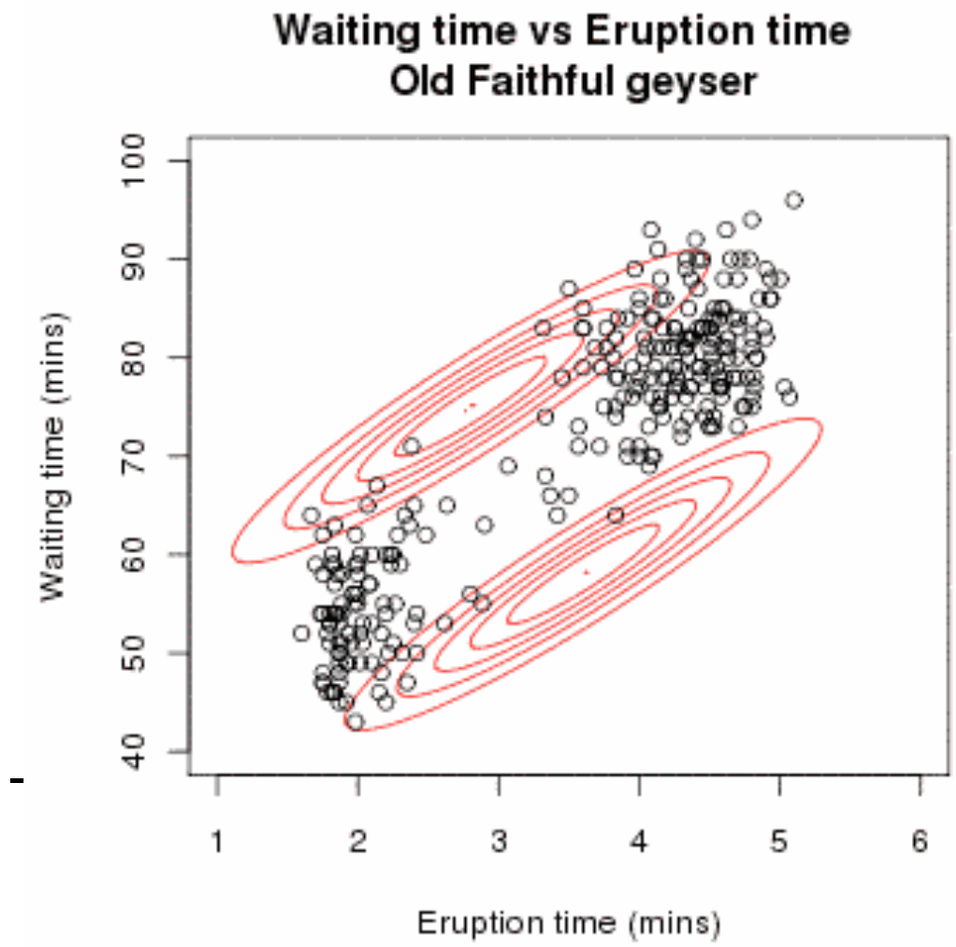
EM for Gaussian Mixtures

- randomize *k* means, covariances, mixing coefficients
- repeat the two steps until convergence
  - E-step: evaluate the responsibilities using current parameters
  - M-step: reestimate parameters using current responsibilities



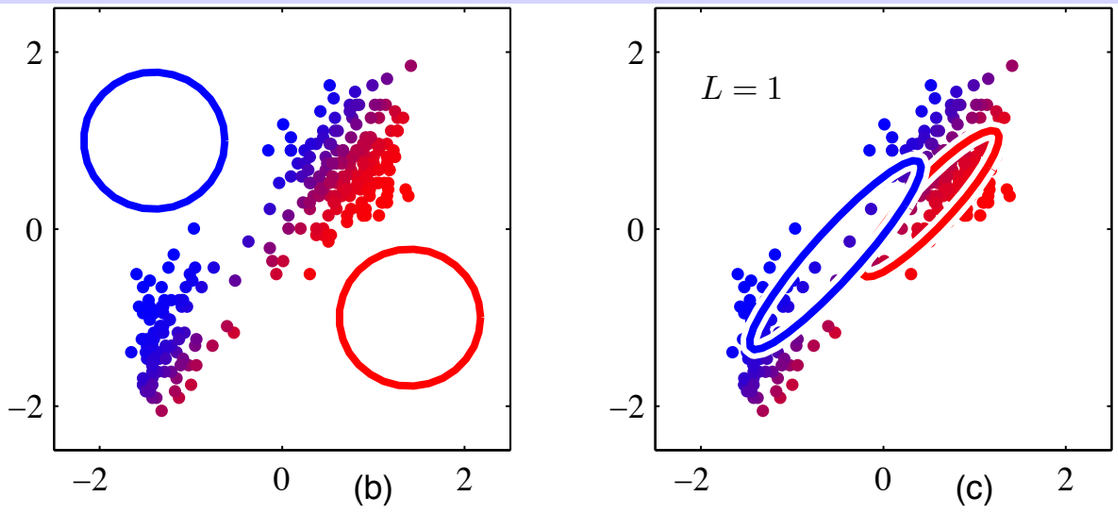
EM for Gaussian Mixtures

- randomize *k* means, covariances, mixing coefficients
- repeat the two steps until convergence
  - E-step: evaluate the responsibilities using current parameters
  - M-step: reestimate parameters using current responsibilities



EM for Gaussian Mixtures

$$P(x) = \sum_{i=1}^{n_c} P(c_i)P(x|c_i)$$



Initialization: Choose means at random, etc.

E step: For all examples x_k:

$$P(\mu_i|x_k) = \frac{P(\mu_i)P(x_k|\mu_i)}{P(x_k)} = \frac{P(\mu_i)P(x_k|\mu_i)}{\sum_{i'} P(\mu_{i'})P(x_k|\mu_{i'})}$$

M step: For all components c_i:

$$\begin{aligned} P(c_i) &= \frac{1}{n_e}\sum_{k=1}^{n_e} P(\mu_i|x_k) \\ \mu_i &= \frac{\sum_{k=1}^{n_e} x_k\, P(\mu_i|x_k)}{\sum_{k=1}^{n_e} P(\mu_i|x_k)} \\ \sigma_i^2 &= \frac{\sum_{k=1}^{n_e} (x_k - \mu_i)^2\, P(\mu_i|x_k)}{\sum_{k=1}^{n_e} P(\mu_i|x_k)} \end{aligned}$$

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.$$

$$\pi_k^{\text{new}} = \frac{N_k}{N} \qquad N_k = \sum_{n=1}^{N} \gamma(z_{nk}).$$

$$\begin{aligned} \boldsymbol{\mu}_k^{\text{new}} &= \frac{1}{N_k}\sum_{n=1}^{N} \gamma(z_{nk})\mathbf{x}_n \\ \boldsymbol{\Sigma}_k^{\text{new}} &= \frac{1}{N_k}\sum_{n=1}^{N} \gamma(z_{nk})\,(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})\,(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})^{\mathrm{T}} \end{aligned}$$

Convergence

- EM converges much slower than *k*-means
  - can't use "assignment doesn't change" for convergence
- use log likelihood of the data
  - stop if increase in log likelihood smaller than threshold
  - or a maximum # of iterations has reached

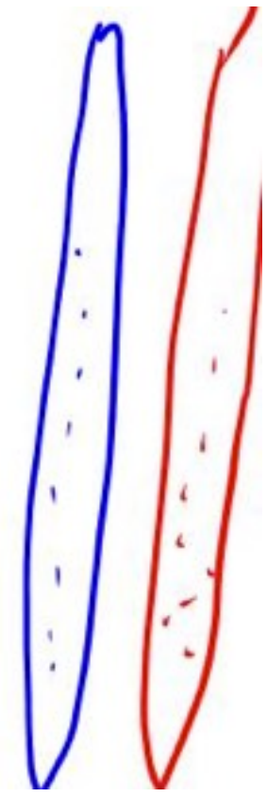
$$P(x) = \sum_{i=1}^{n_c} P(c_i)P(x|c_i)$$

$$\begin{aligned} L &= \log P(\text{data}) \\ &= \log \Pi_j P(x_j) \\ &= \sum_j \log P(x_j) \\ &= \sum_j \log \sum_i P(c_i)P(x_j \mid c_i) \end{aligned}$$

EM: pros and cons (vs. *k*-means)

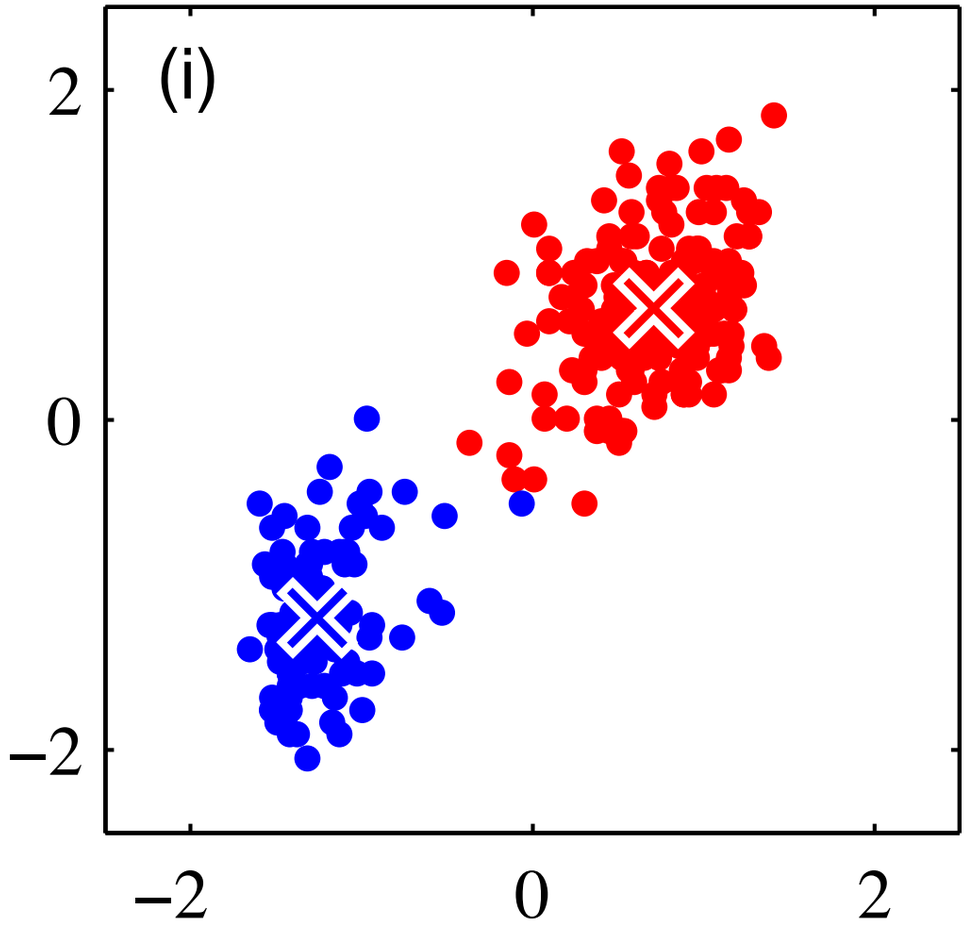
- EM: pros
  - doesn't need the data to be spherical
  - doesn't need the data to be well-separated
  - doesn't need the clusters to be in similar sizes/volumes

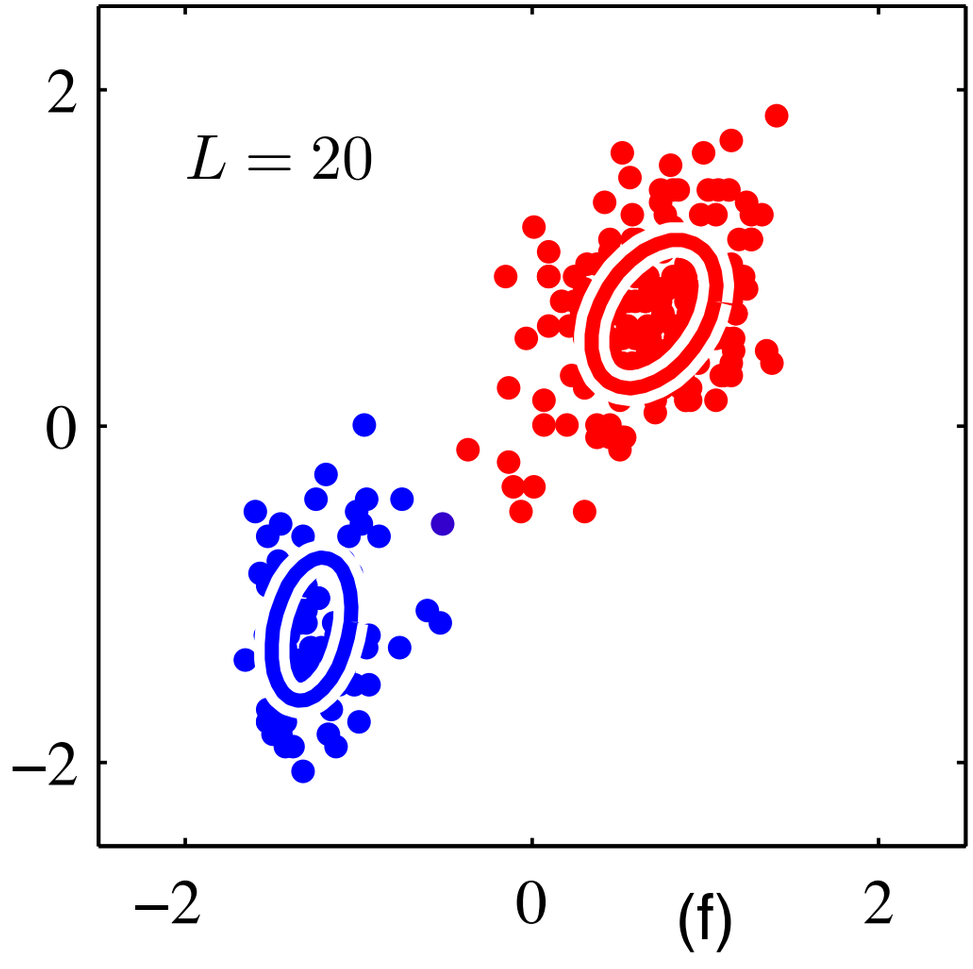
- EM: cons
  - converges much slower than *k*-means
  - per-iteration computation also slower
  - (to speedup EM): use *k*-means to burn-in
  - (same as *k*-means) local minimum!



k-means is a special case of EM

- *k*-means is "hard" EM
  - covariance matrix is diagonal -- i.e., spherical
  - diagonal variances are approaching 0





Why EM increases p(data) iteratively?

$$D = \log p(x;\theta) = \log \sum_z p(x,z;\theta)$$

Why EM increases p(data) iteratively?

$$D = \log p(x;\theta) = \log \sum_z p(x,z;\theta) \frac{p(z|x;\theta_t)}{p(z|x;\theta_t)}.$$

Why EM increases p(data) iteratively?

$$D = \log p(x;\theta) = \log \sum_z p(x,z;\theta) \frac{p(z|x;\theta_t)}{p(z|x;\theta_t)}.$$

Note that $\sum_z p(z|x;\theta_t) = 1$ and $p(z|x;\theta_t) \geq 0$ for all z. Therefore D is the logarithm of a weighted sum, so we can apply Jensen's inequality, which says $\log \sum_j w_j v_j \geq \sum_j w_j \log v_j$, given $\sum_j w_j = 1$ and each $w_j \geq 0$. Here, we let the sum range over the values z of Z, with the weight w_j being $p(z|x;\theta_t)$. We get

$$D \geq E = \sum_z p(z|x;\theta_t) \log \frac{p(x,z;\theta)}{p(z|x;\theta_t)}.$$

Why EM increases p(data) iteratively?

$$D = \log p(x;\theta) = \log \sum_z p(x,z;\theta) \frac{p(z|x;\theta_t)}{p(z|x;\theta_t)}.$$

Note that $\sum_z p(z|x;\theta_t) = 1$ and $p(z|x;\theta_t) \geq 0$ for all z. Therefore D is the logarithm of a weighted sum, so we can apply Jensen's inequality, which says $\log \sum_j w_j v_j \geq \sum_j w_j \log v_j$, given $\sum_j w_j = 1$ and each $w_j \geq 0$. Here, we let the sum range over the values z of Z, with the weight w_j being $p(z|x;\theta_t)$. We get

$$D \geq E = \sum_z p(z|x;\theta_t) \log \frac{p(x,z;\theta)}{p(z|x;\theta_t)}.$$

Separating the fraction inside the logarithm to obtain two sums gives

$$E = \Big(\sum_z p(z|x;\theta_t) \log p(x,z;\theta)\Big) - \Big(\sum_z p(z|x;\theta_t) \log p(z|x;\theta_t)\Big).$$

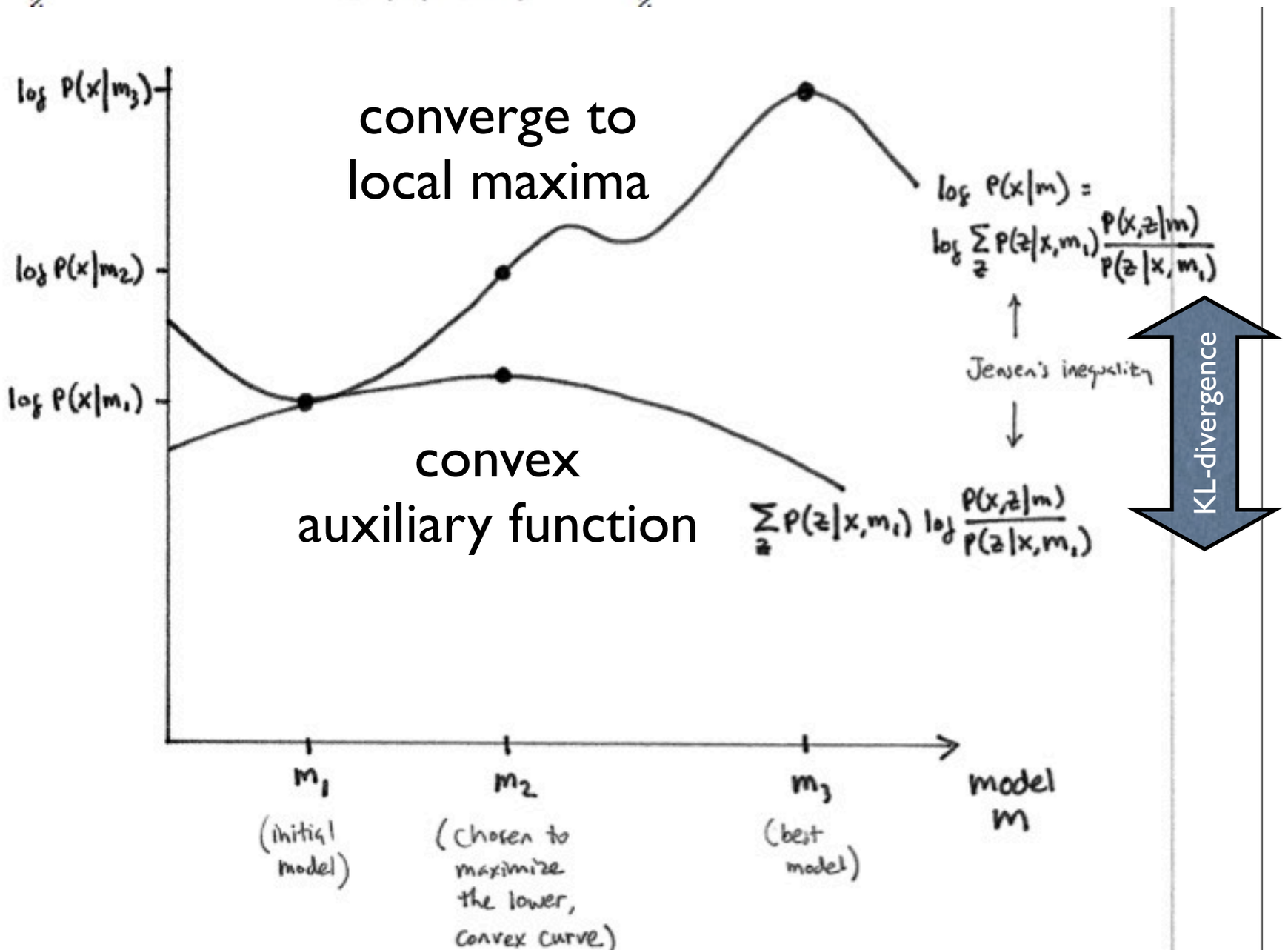
Since $E \leq D$ and we want to maximize D, consider maximizing E. The weights $p(z|x;\theta_t)$ do not depend on θ, so we only need to maximize the first sum, which is

$$\sum_z p(z|x;\theta_t) \log p(x,z;\theta).$$

Why EM increases p(data) iteratively?

How do we know that maximizing E actually leads to an improvement in the likelihood? With $\theta = \theta_t$,

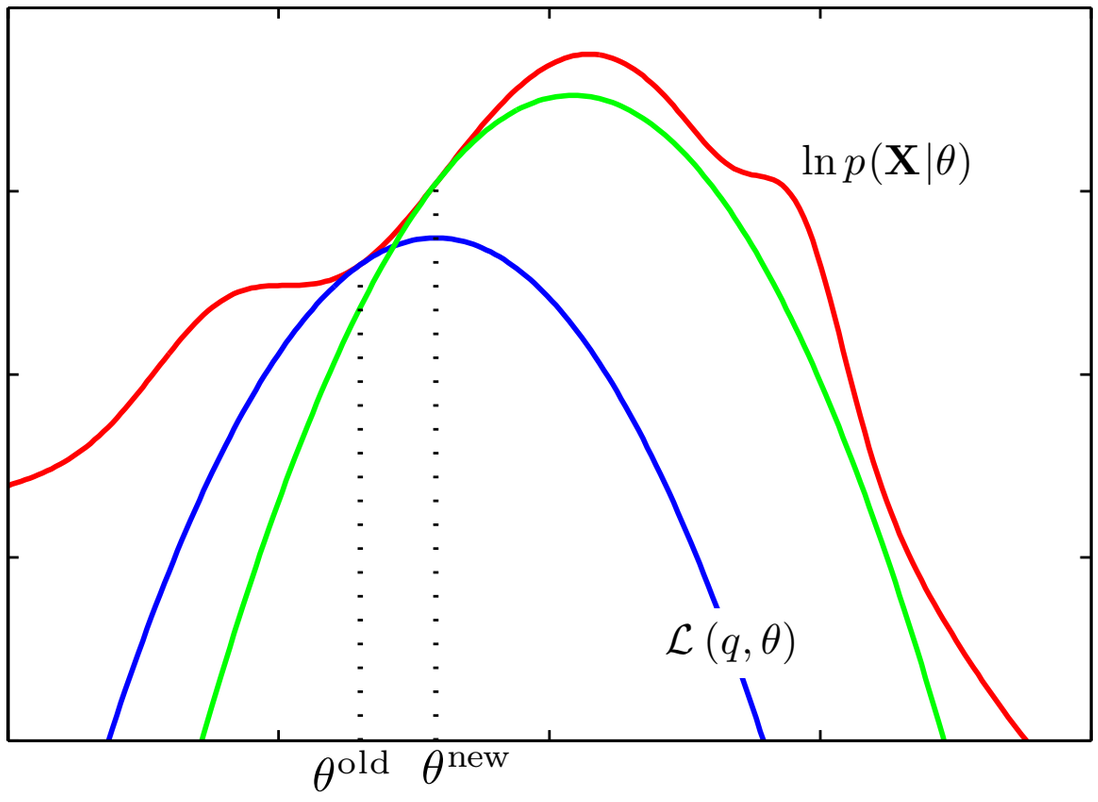
$$E = \sum_z p(z|x;\theta_t) \log \frac{p(x,z;\theta_t)}{p(z|x;\theta_t)} = \sum_z p(z|x;\theta_t) \log p(x;\theta_t) = \log p(x;\theta_t) = D$$

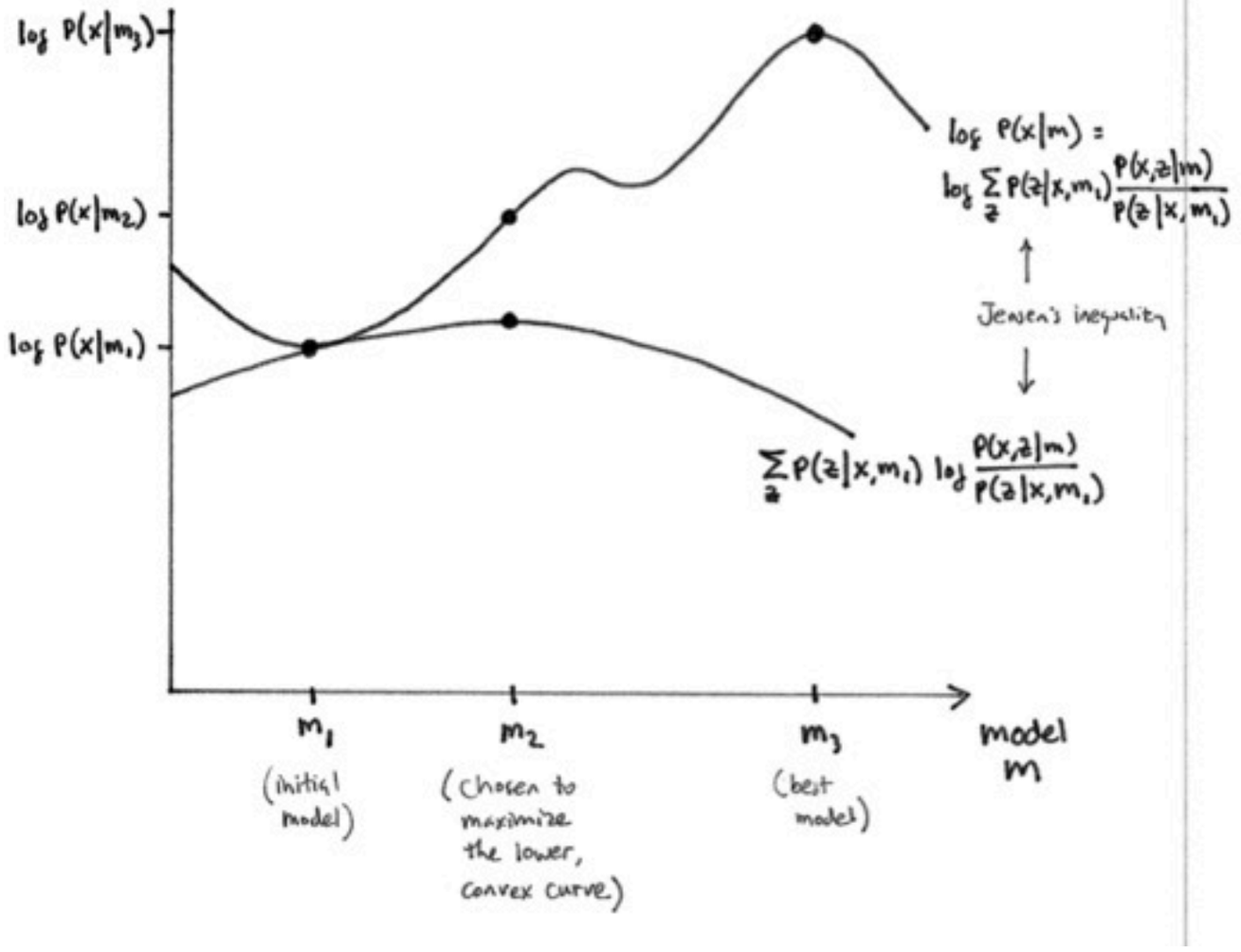


How to maximize the auxiliary?

$$\sum_z p(z|x;\theta_t)\log p(x,z;\theta).$$

In general, the E-step of an EM algorithm is to compute $p(z|x;\theta_t)$ for all z. The M-step is then to find θ to maximize $\sum_z p(z|x;\theta_t)\log p(x,z;\theta)$.





Machine Learning

A Geometric Approach

CUNY Graduate Center, Spring 2013

Lectures 13: Unsupervised Learning 2

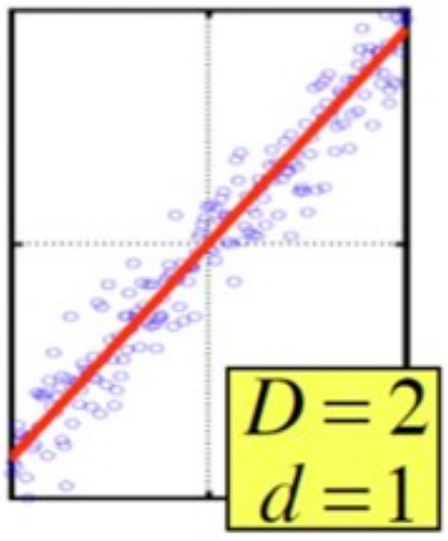
(dimensionality reduction: linear: PCA, ICA, CCA, LDA[1], LDA[2]; non-linear: kernel PCA, isomap, LLE, SDE)

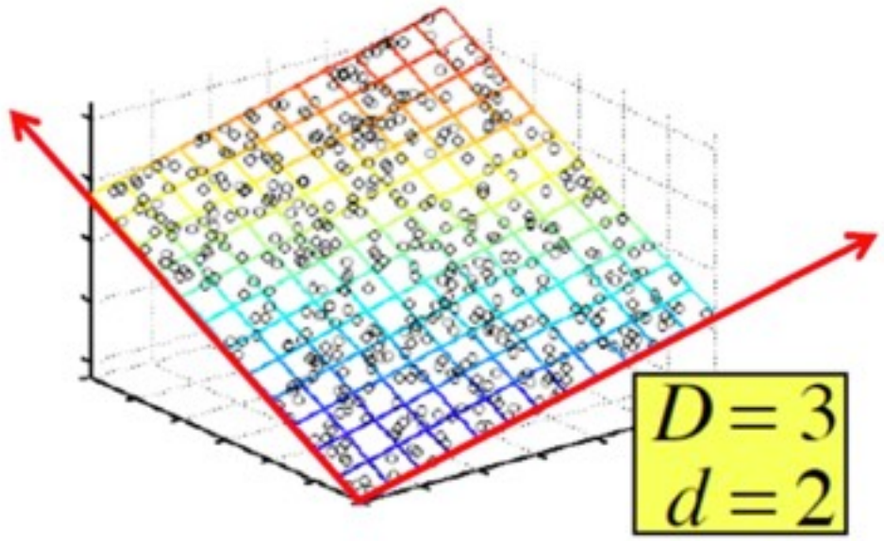
Professor Liang Huang

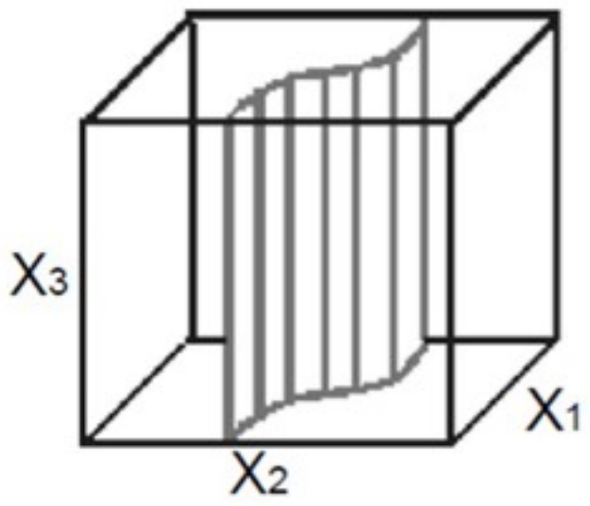
huang@cs.qc.cuny.edu

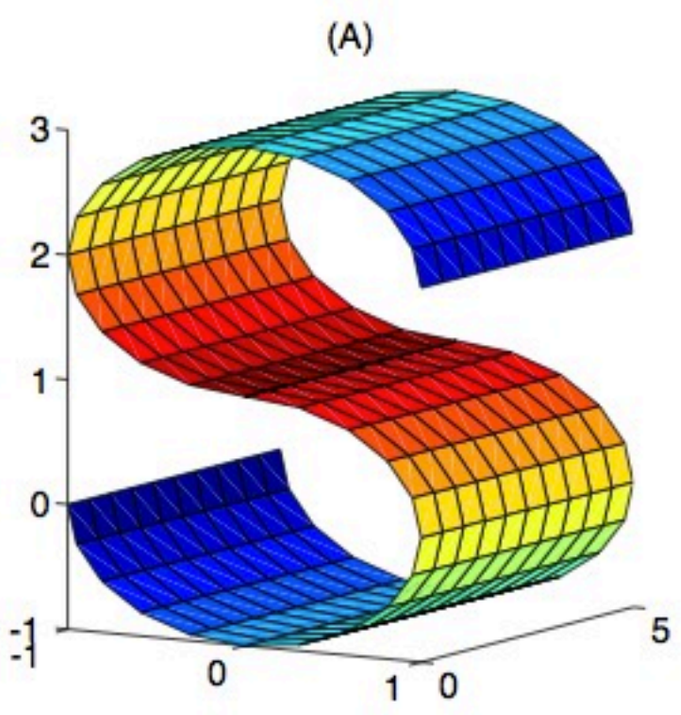
http://acl.cs.qc.edu/~lhuang/teaching/machine-learning

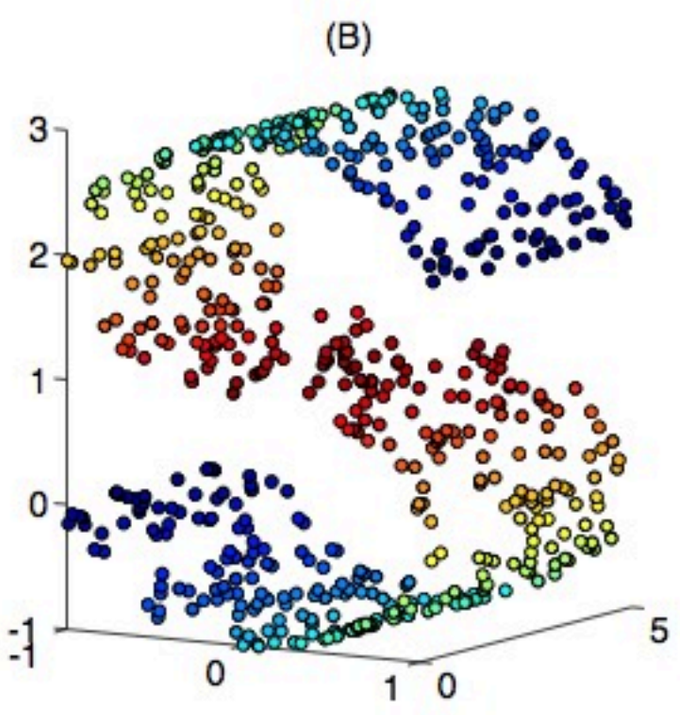
Dimensionality Reduction

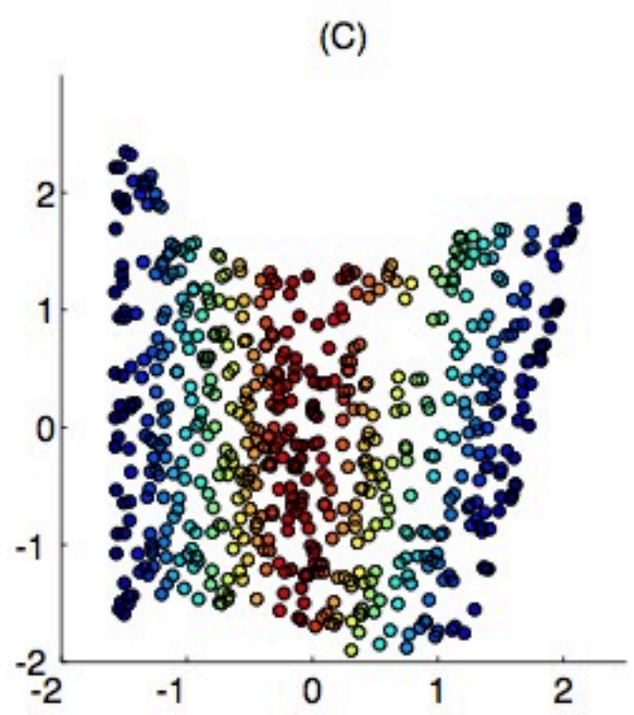




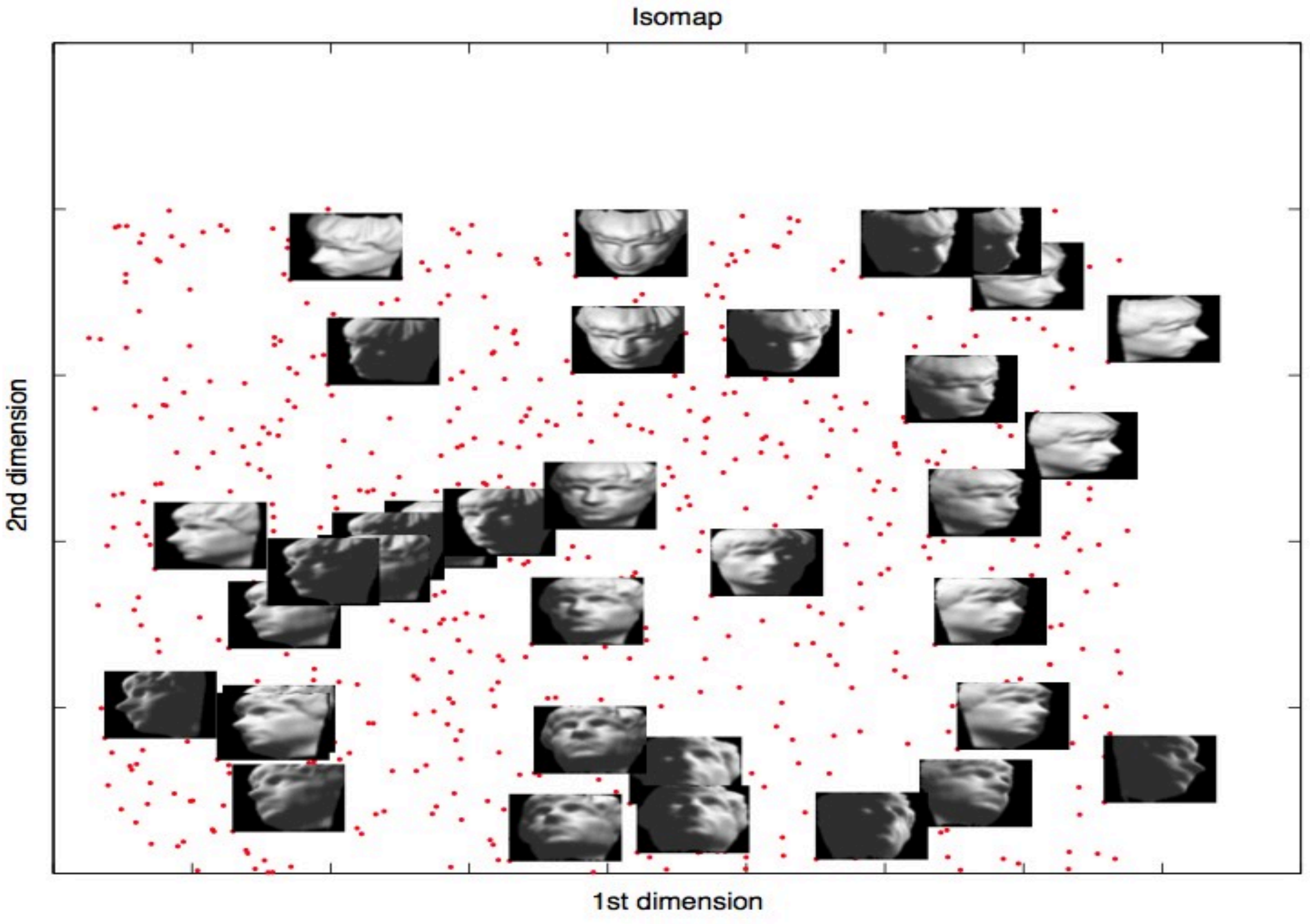






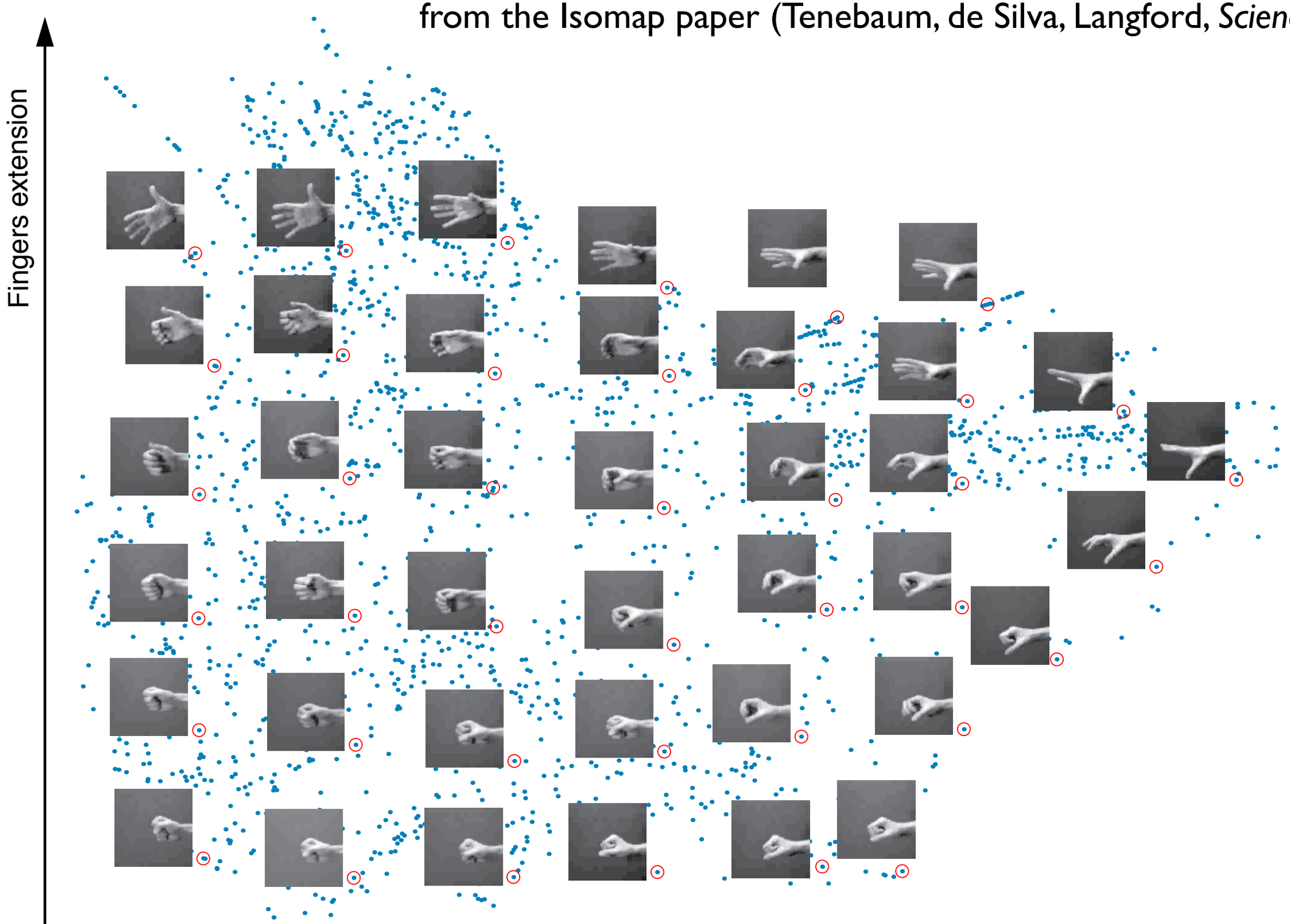


Dimensionality Reduction



Dimensionality Reduction

from the Isomap paper (Tenebaum, de Silva, Langford, *Science* 2000)



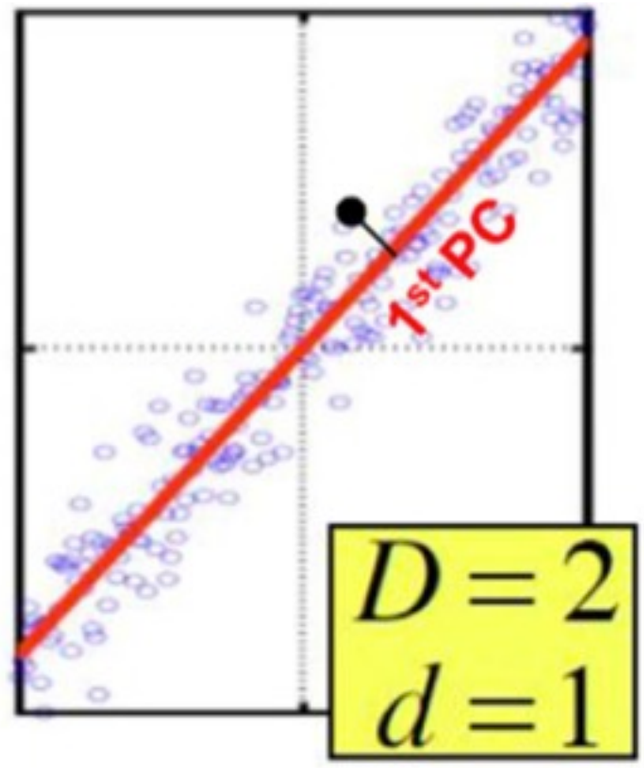
Algorithms

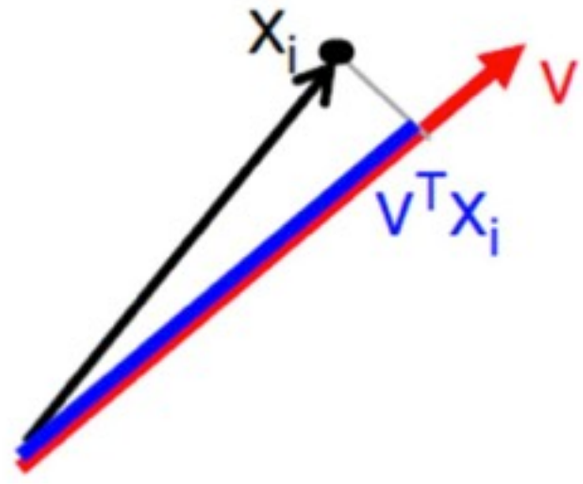
- linear methods
  - PCA - principle ...
  - ICA - independent ...
  - CCA - canonical ...
  - MDS - multidim. scaling
  - LEM - laplacian eignen maps
  - LDA1 - linear discriminant analysis
  - LDA2 - latent dirichlet allocation
- non-linear methods
  - kernelized PCA
  - isomap
  - LLE - locally linear embedding
  - SDE - semidefinite embedding

all are spectral methods! -- i.e., using eigenvalues

PCA

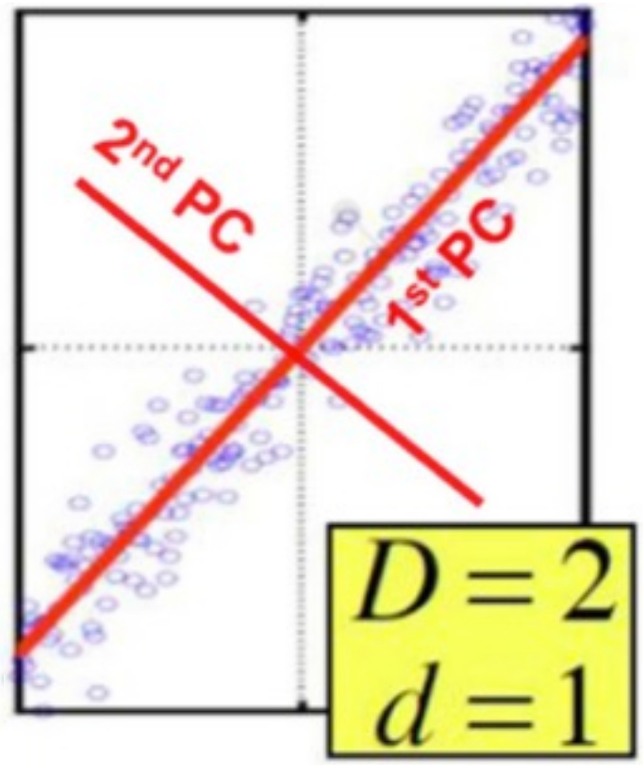
- greedily find *d* orthogonal axes onto which the variance under projection is maximal
  - the "max variance subspace" formulation
  - 1st PC: direction of greatest variability in data
  - 2nd PC: the next *unrelated* max-var direction
    - remove all variance in 1st PC, redo max-var
- another equivalent formulation: "minimum reconstruction error"
  - find orthogonal vectors onto which the projection yields min MSE reconstruction

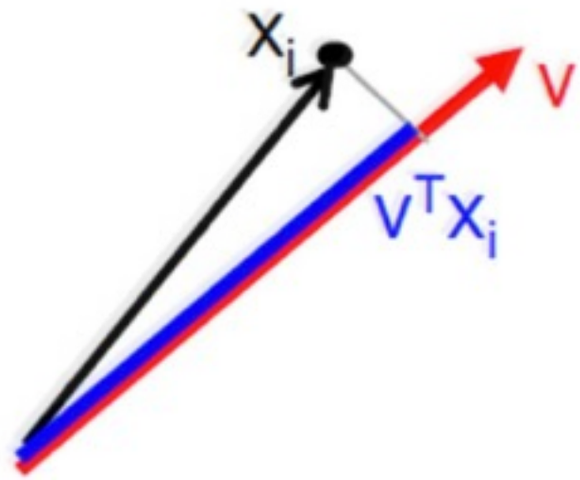




PCA

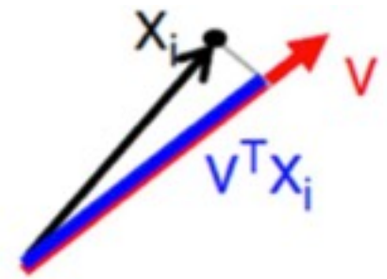
- greedily find *d* orthogonal axes onto which the variance under projection is maximal
  - the "max variance subspace" formulation
  - 1st PC: direction of greatest variability in data
  - 2nd PC: the next *unrelated* max-var direction
    - remove all variance in 1st PC, redo max-var
- another equivalent formulation: "minimum reconstruction error"
  - find orthogonal vectors onto which the projection yields min MSE reconstruction





PCA optimization: max-var proj.

- first translate data to zero mean
- compute co-variance matrix
- find top *d* eigenvalues and eigenvectors of covar matrix
- project data onto those eigenvectors



$$\max_{\mathbf{v}} \ \mathbf{v}^T \mathbf{X}\mathbf{X}^T \mathbf{v} \quad \text{s.t.} \quad \mathbf{v}^T\mathbf{v} = 1$$

Lagrangian: $\max_{\mathbf{v}} \mathbf{v}^T \mathbf{X}\mathbf{X}^T \mathbf{v} - \lambda \mathbf{v}^T\mathbf{v}$

Wrap constraints into the objective function

$$\partial/\partial \mathbf{v} = 0 \qquad (\mathbf{X}\mathbf{X}^T - \lambda \mathbf{I})\mathbf{v} = 0 \qquad \Rightarrow \boxed{(\mathbf{X}\mathbf{X}^T)\mathbf{v} = \lambda \mathbf{v}}$$

Therefore, v is the eigenvector of sample correlation/ covariance matrix XX^T

PCA for *k*-means and whitening

- rescaling to zero mean and unit variance as preprocessing
  - we did that in perceptron HW1/2 also!
- but PCA can do more: whitening (sphereication)

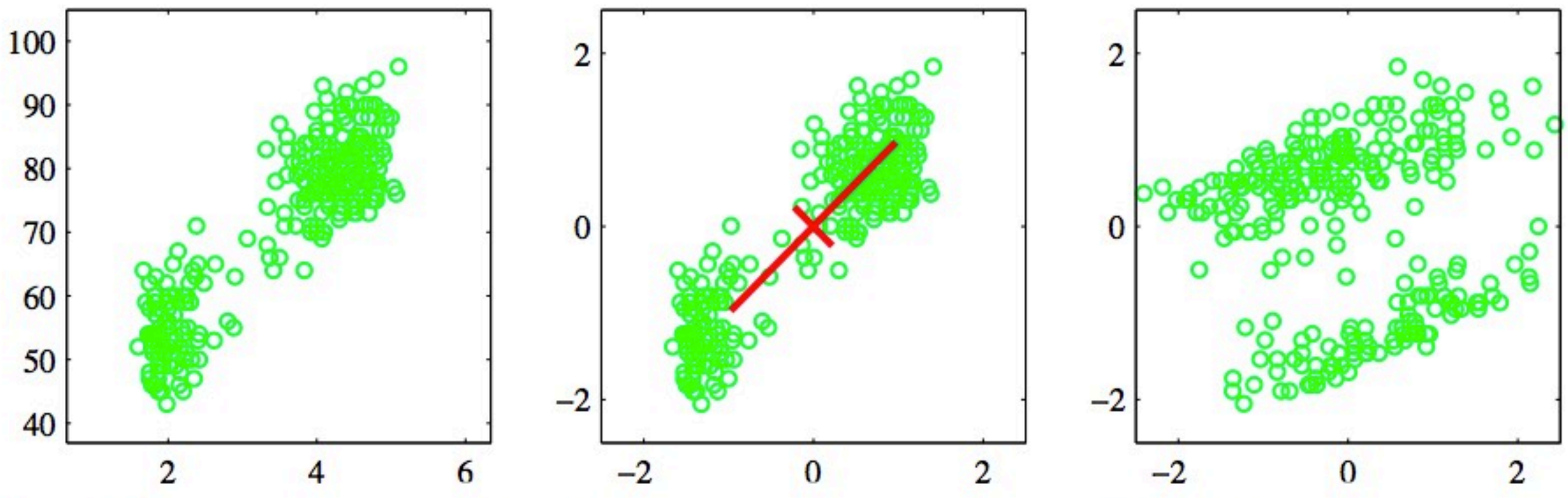


Figure 12.6 Illustration of the effects of linear pre-processing applied to the Old Faithful data set. The plot on the left shows the original data. The centre plot shows the result of standardizing the individual variables to zero mean and unit variance. Also shown are the principal axes of this normalized data set, plotted over the range $\pm\lambda_i^{1/2}$. The plot on the right shows the result of whitening of the data to give it zero mean and unit covariance.

Eigendigits

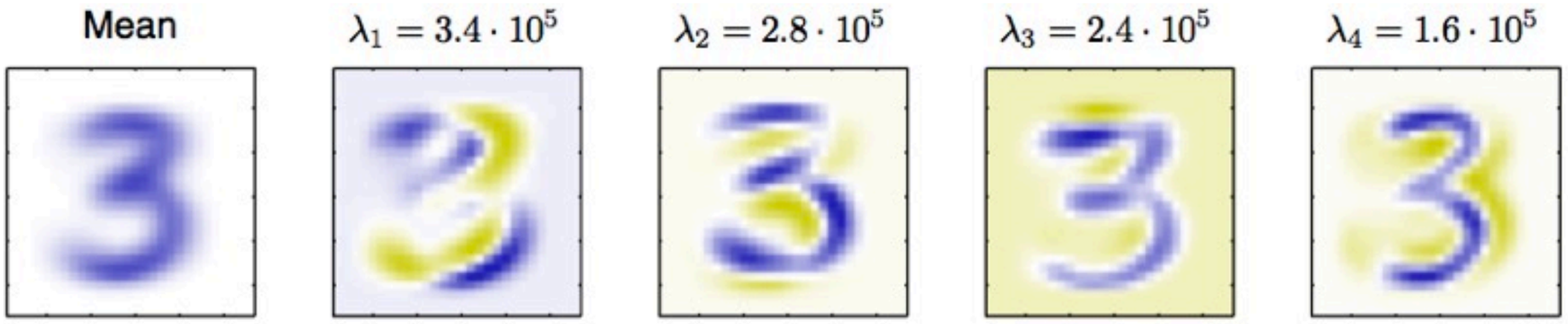


Figure 12.3 The mean vector $\overline{\mathbf{x}}$ along with the first four PCA eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_4$ for the off-line digits data set, together with the corresponding eigenvalues.

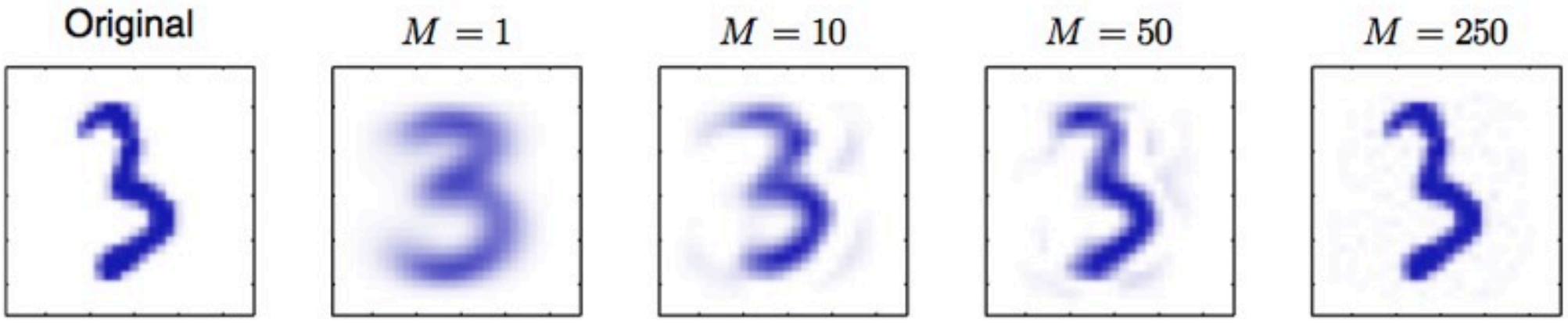


Figure 12.5 An original example from the off-line digits data set together with its PCA reconstructions obtained by retaining M principal components for various values of M. As M increases the reconstruction becomes more accurate and would become perfect when $M = D = 28 \times 28 = 784$.

Eigenfaces

Figure 1. (a) Face images used as the training set.

Eigenfaces

Figure 1. **(b)** The average face Ψ.

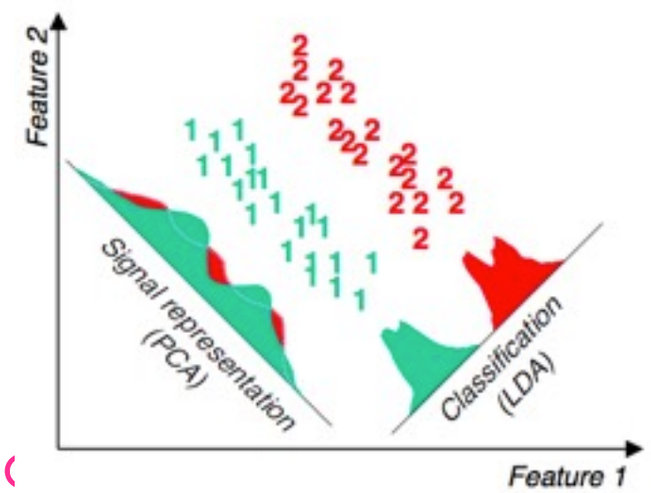
Eigenfaces

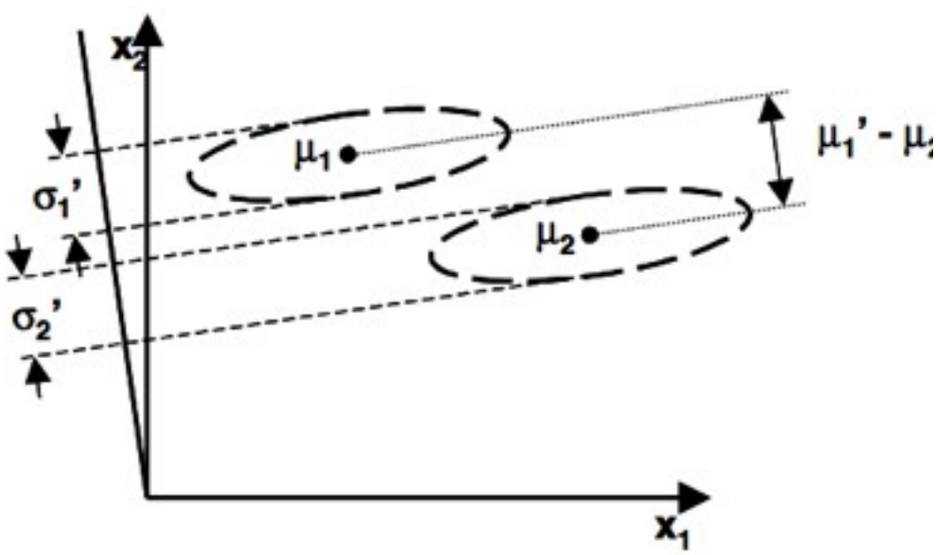
Figure 1. **(b)** The average face Ψ.

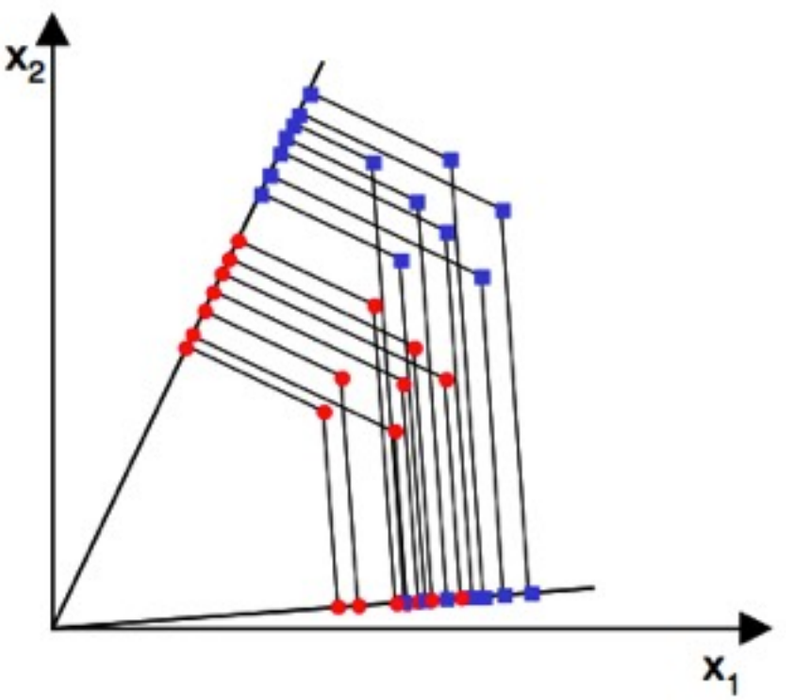
Figure 2. Seven of the eigenfaces calculated from the input images of Figure 1.

LDA: Fisher's linear discriminant analysis

- PCA finds a small representation of the data
- LDA performs dimensionality reduction while preserving as much the given class discrimination info as possible
  - it's a linear classification method (like perceptron)
  - find a scalar projection that maximizes separability
    - max separation b/w projected means
    - min variance within each projected class

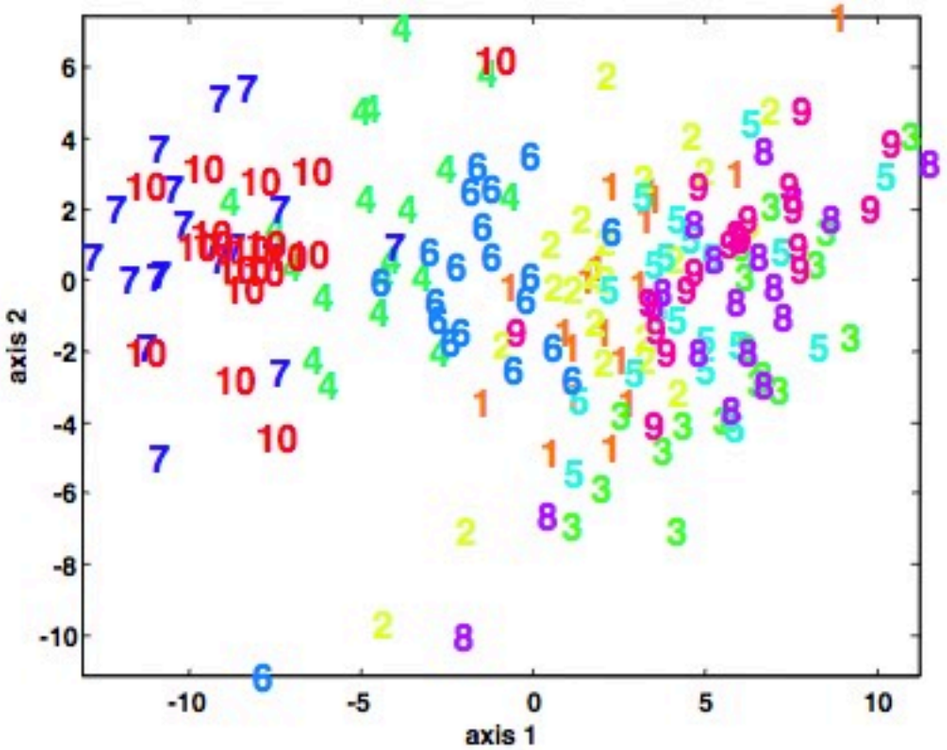


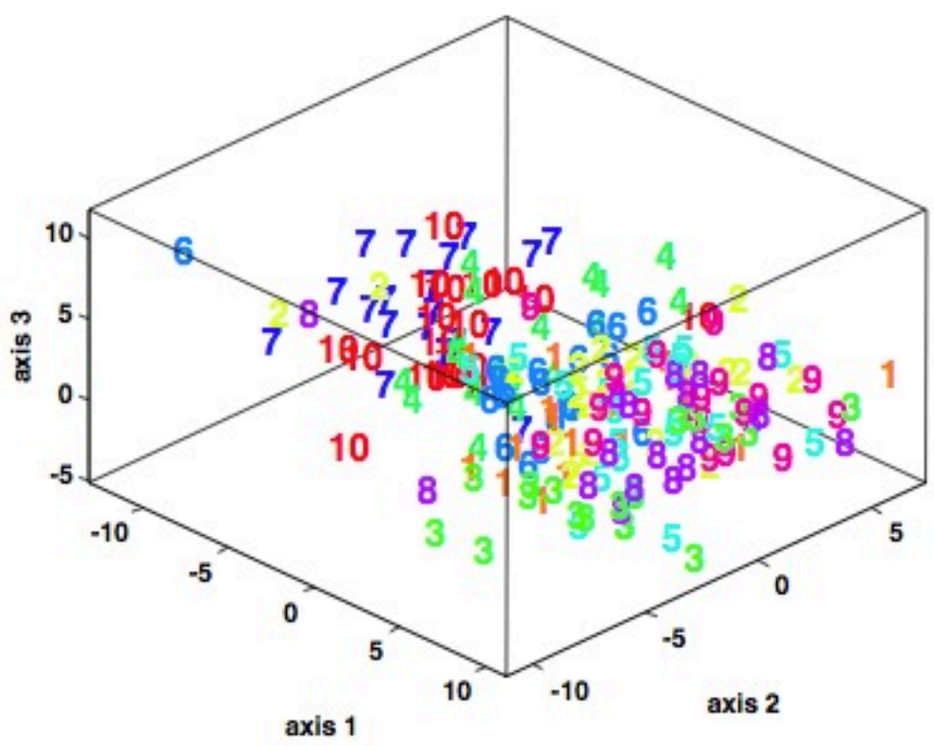




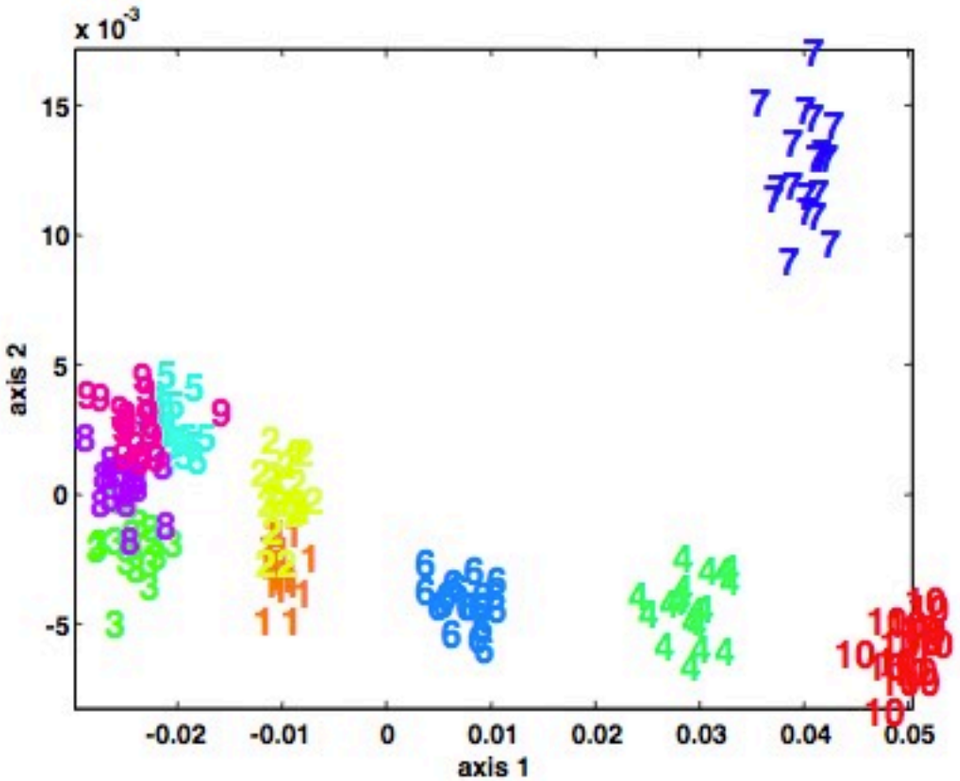
PCA vs LDA

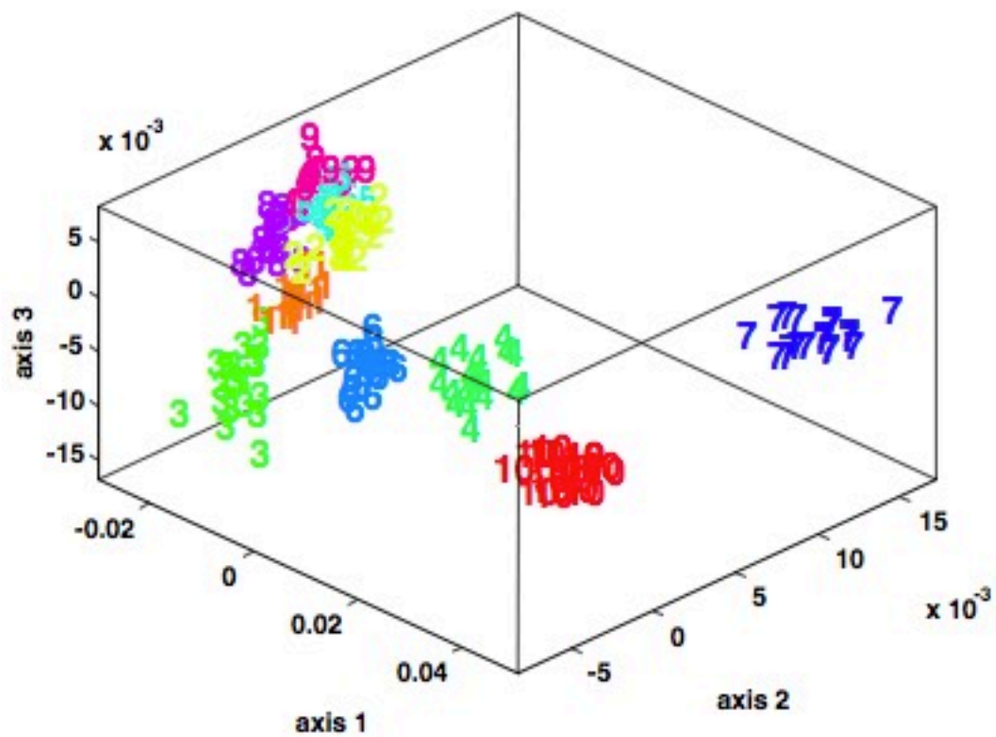
PCA





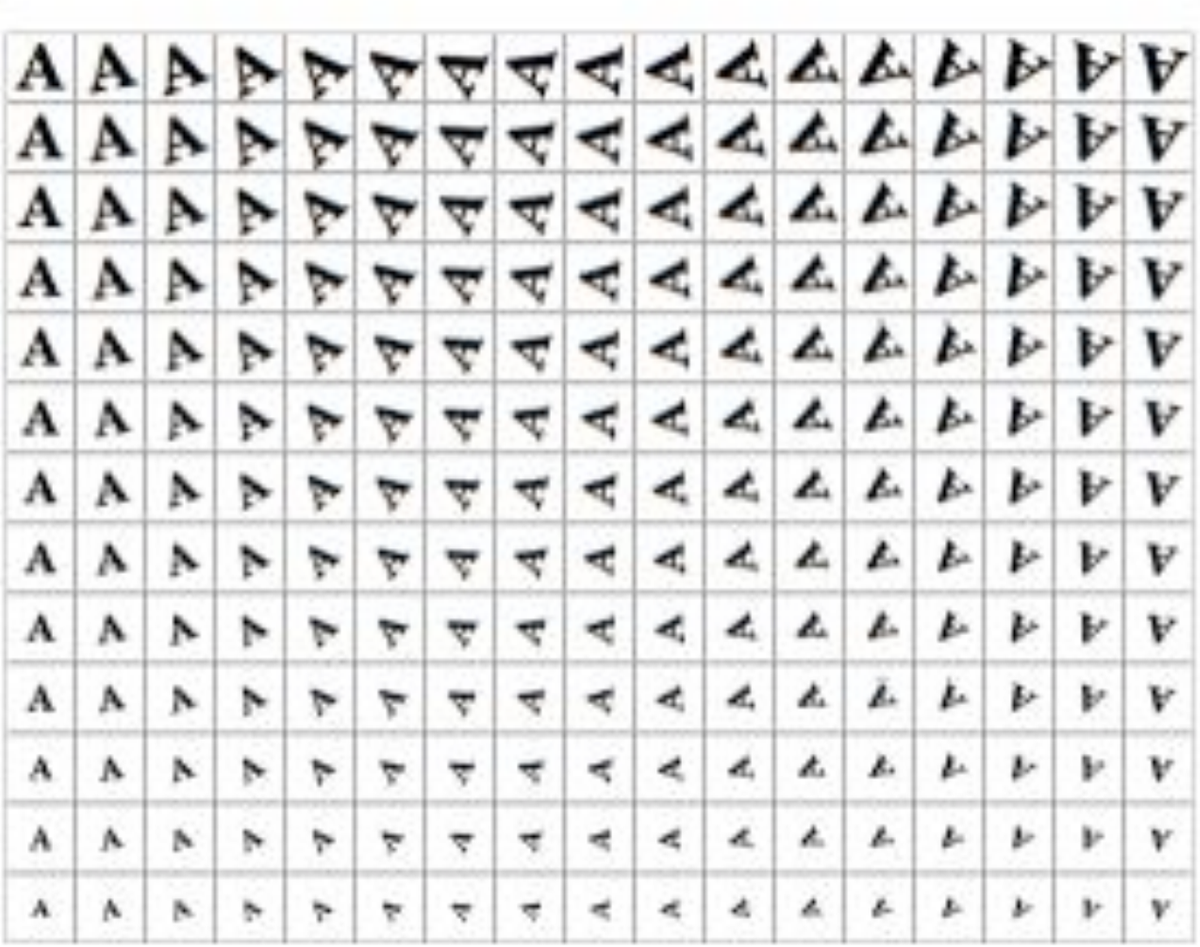
LDA

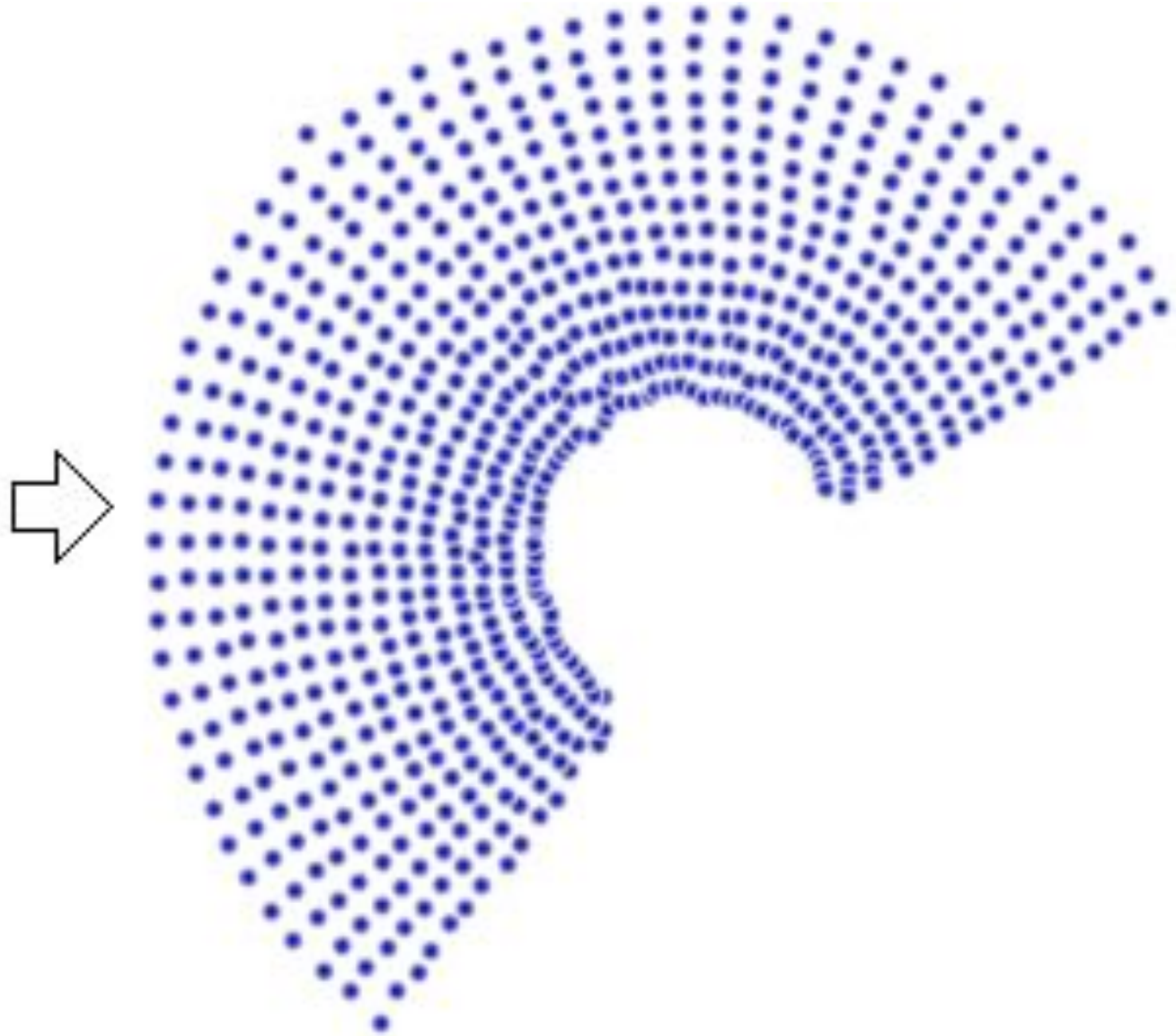




Linear vs. non-Linear

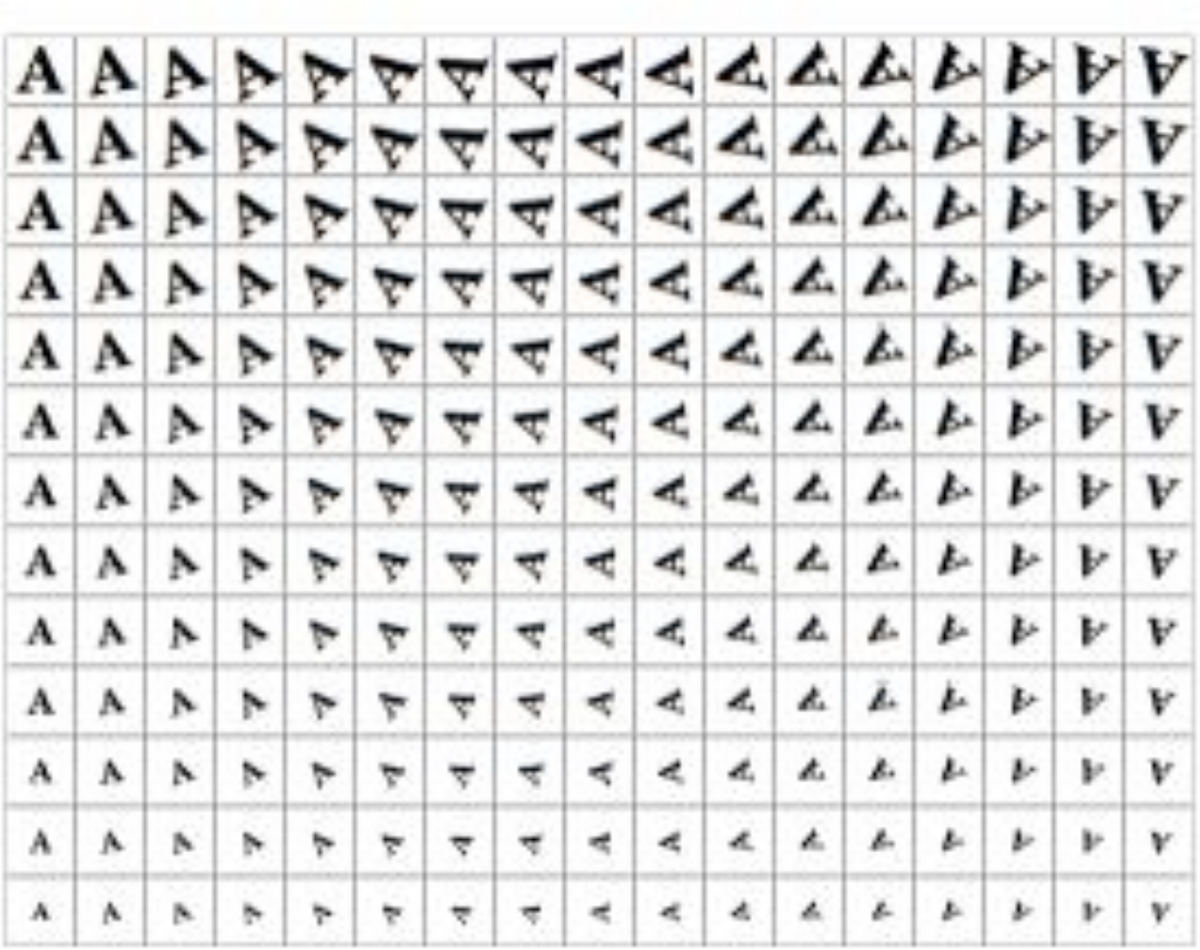
LLE or isomap

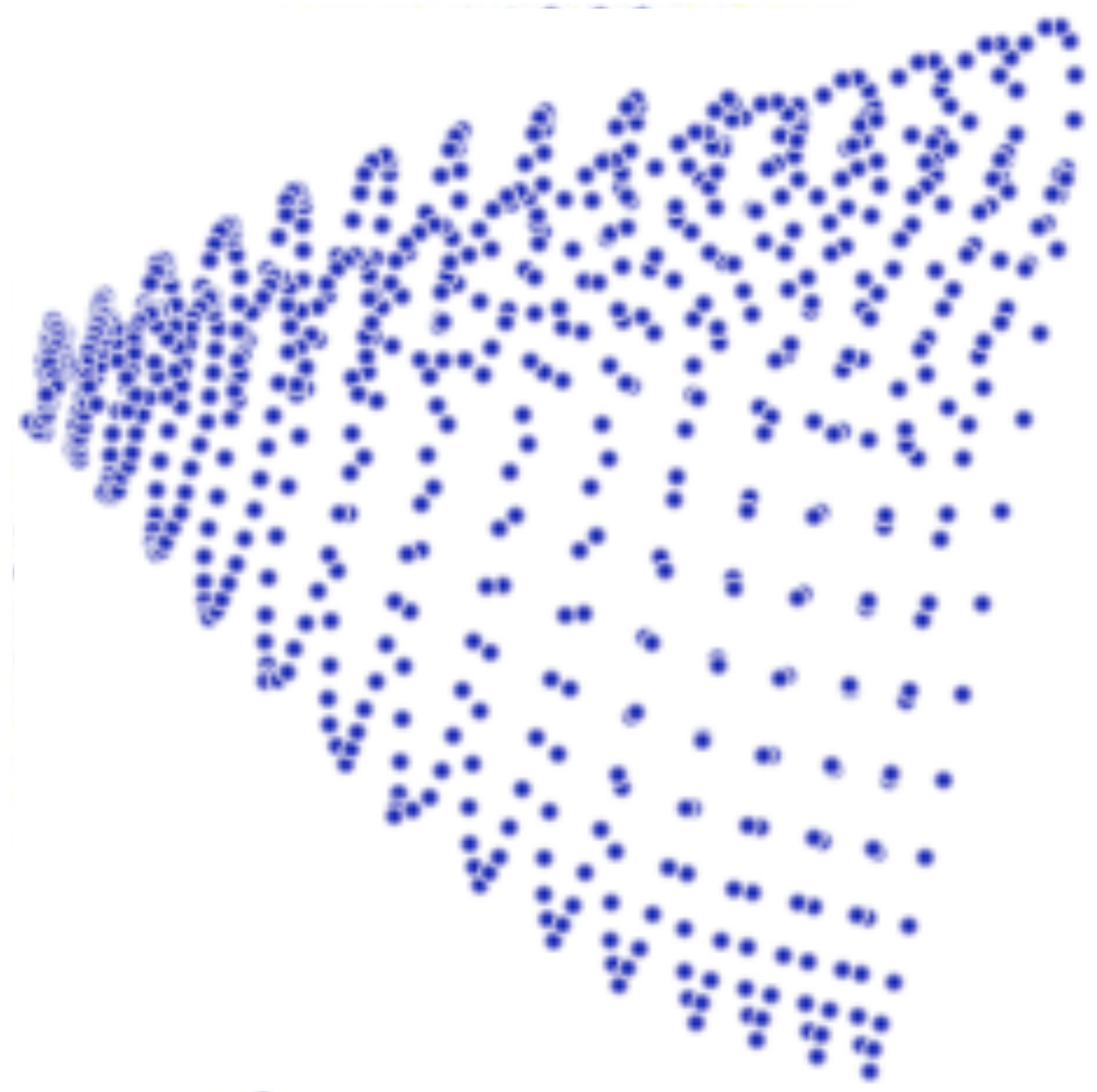




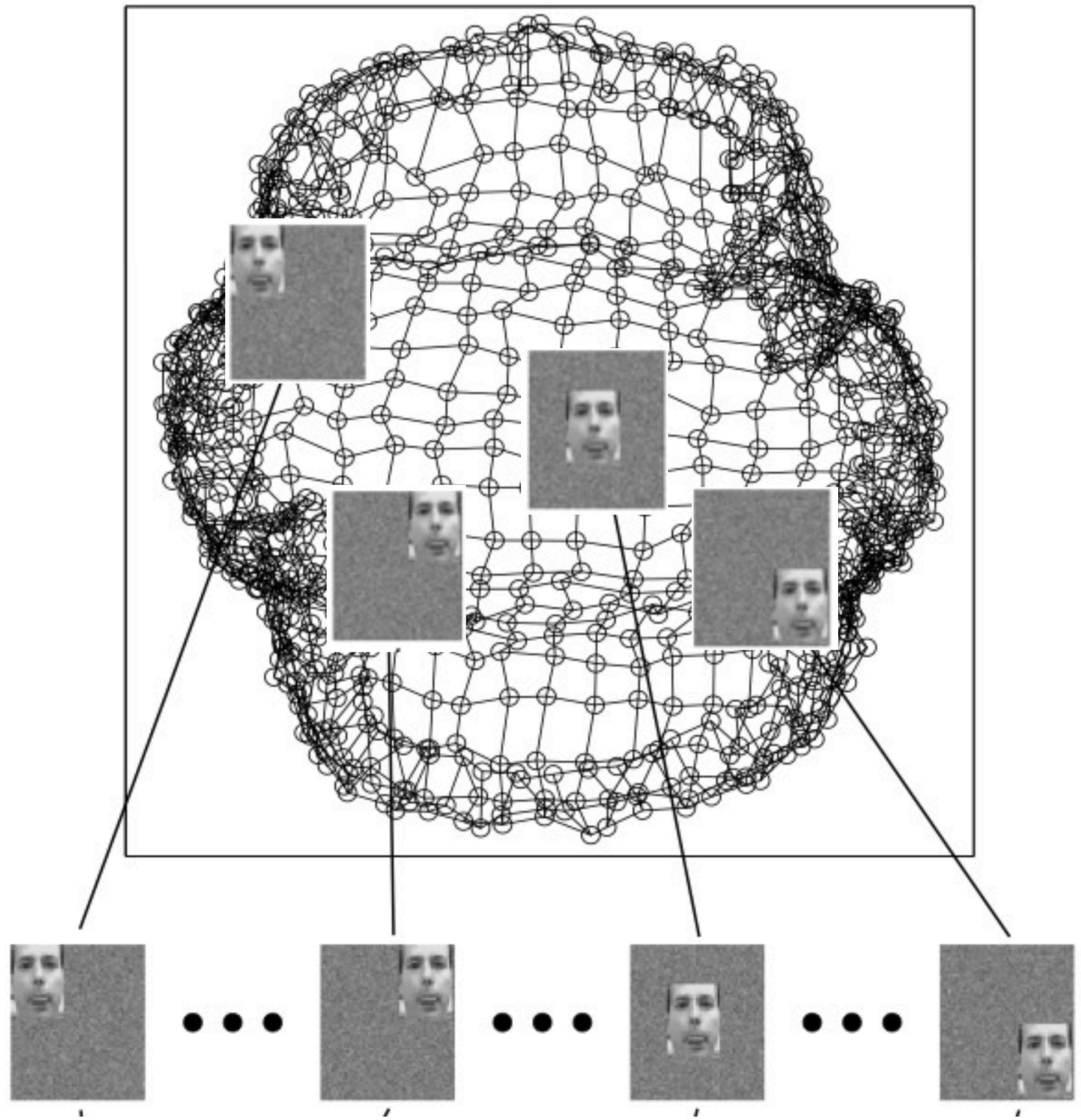
Linear vs. non-Linear

PCA



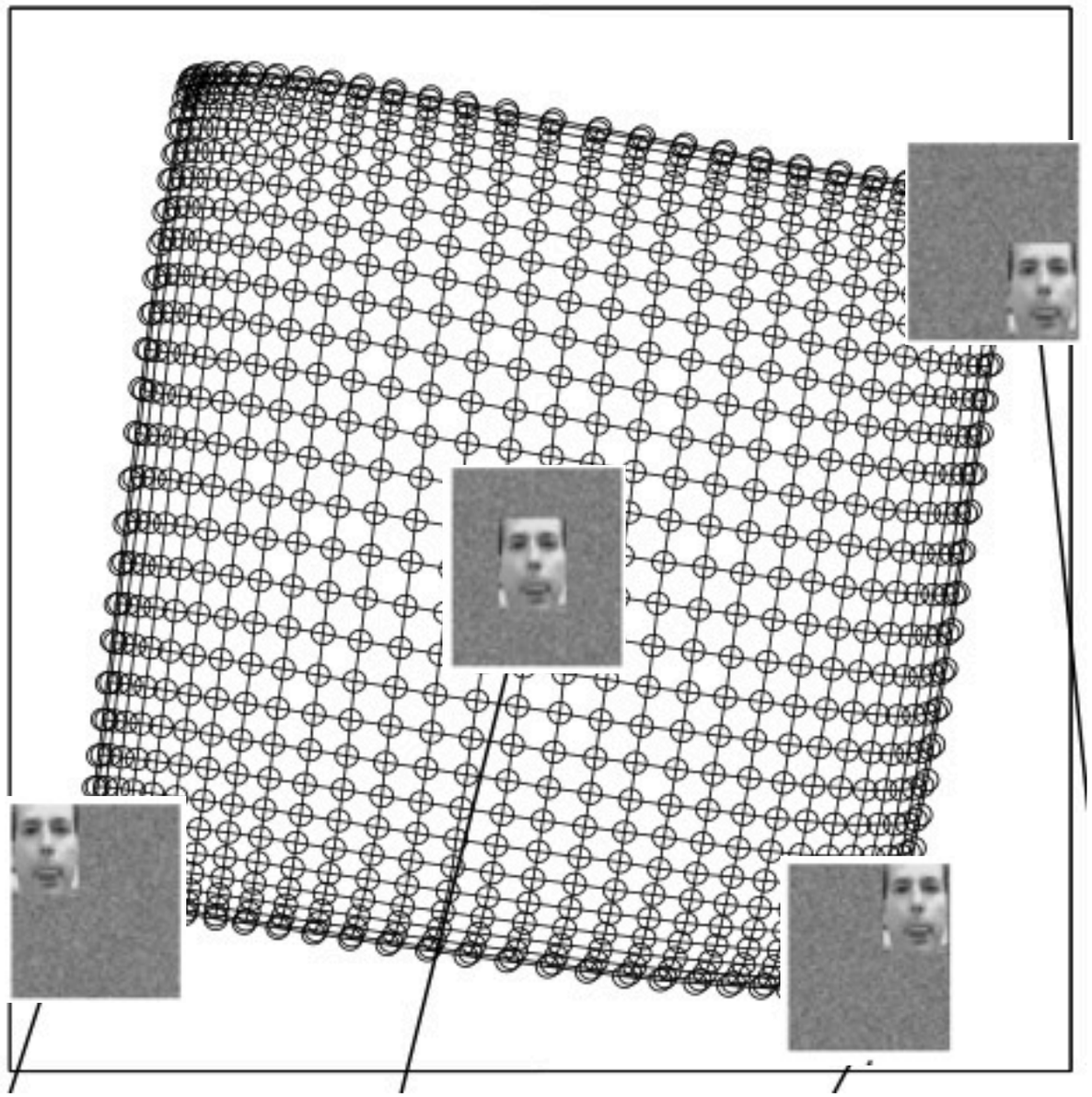


Linear vs. non-Linear



PCA

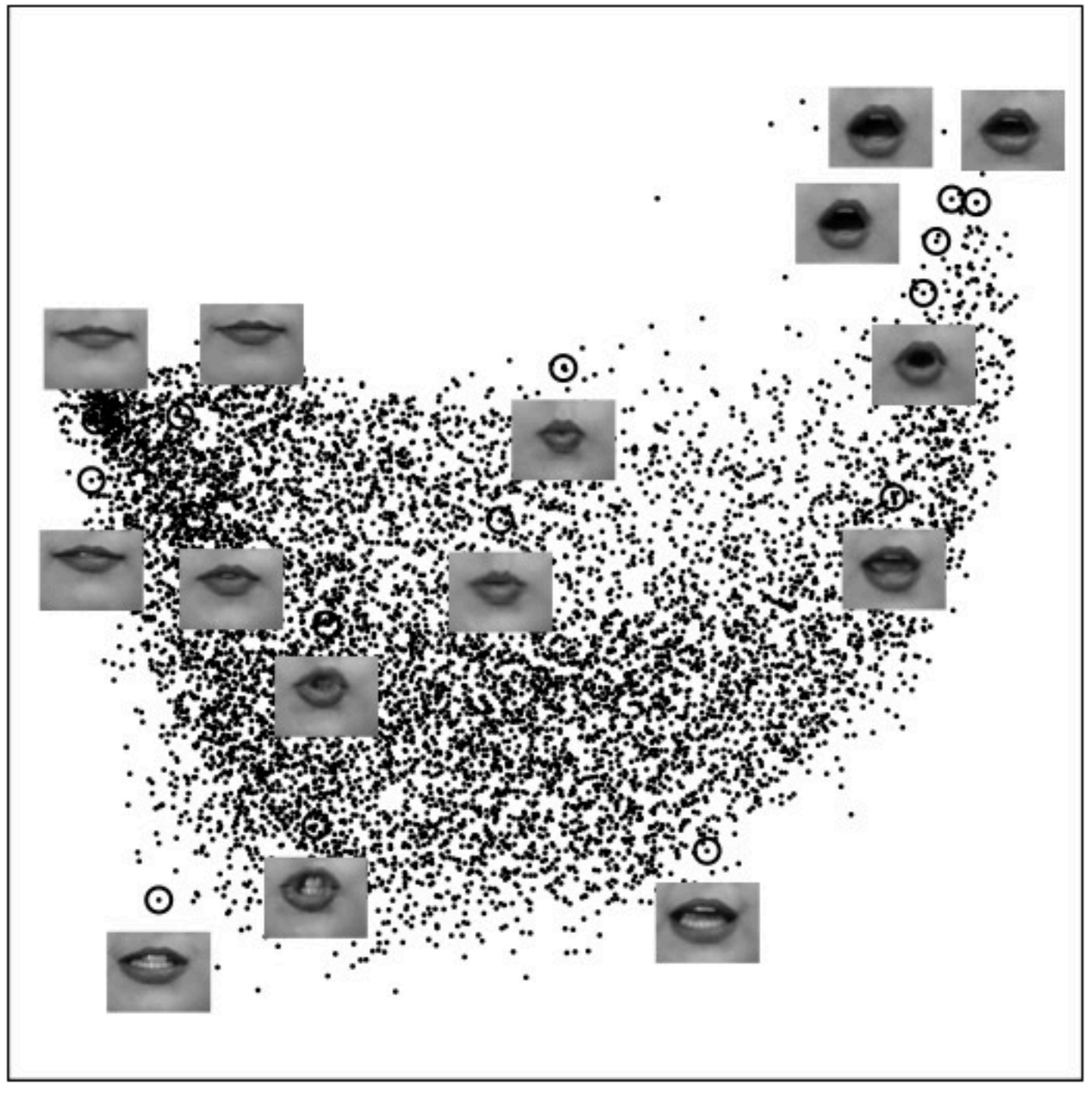
Linear vs. non-Linear



LLE

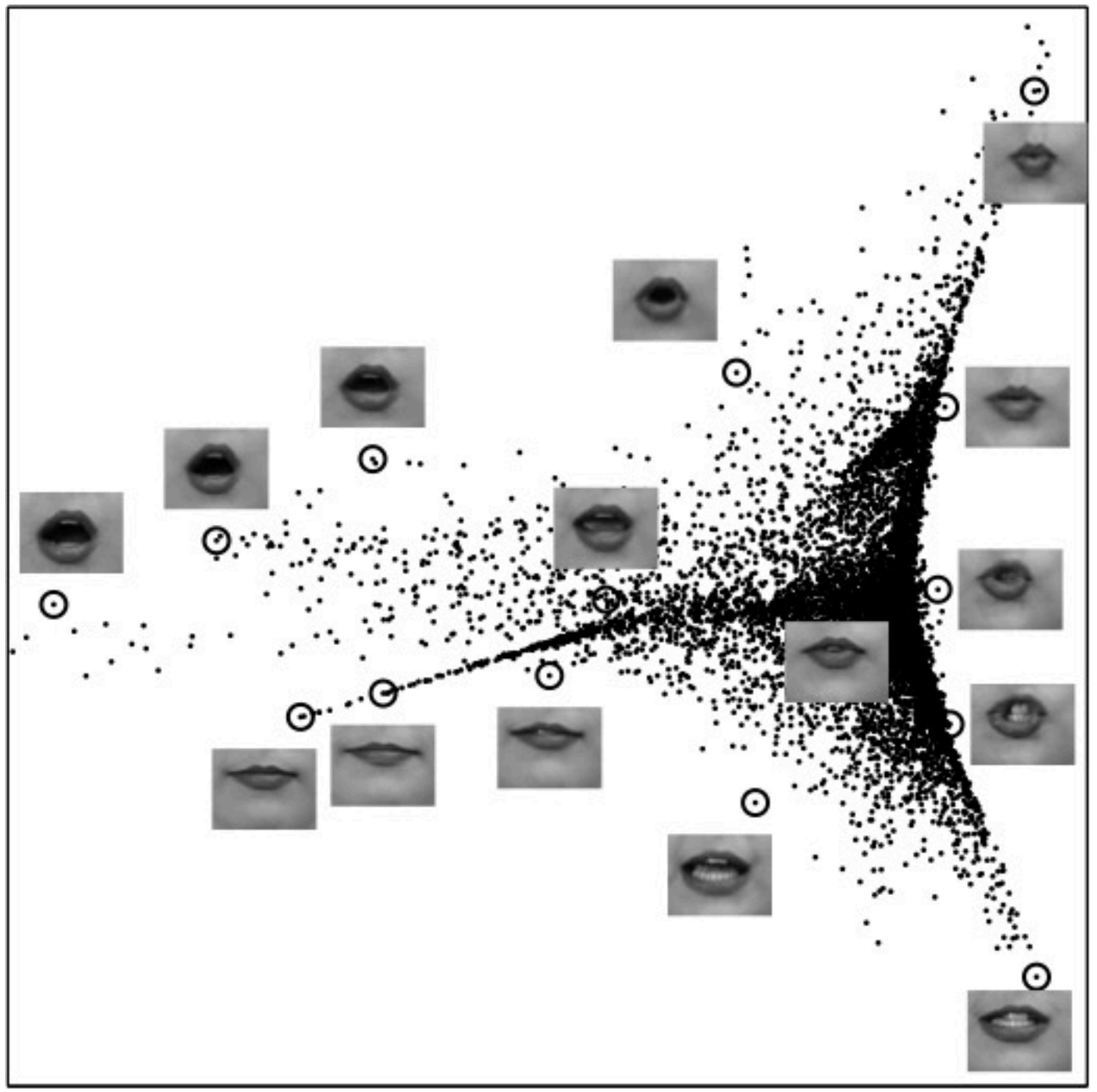
Linear vs. non-Linear

PCA

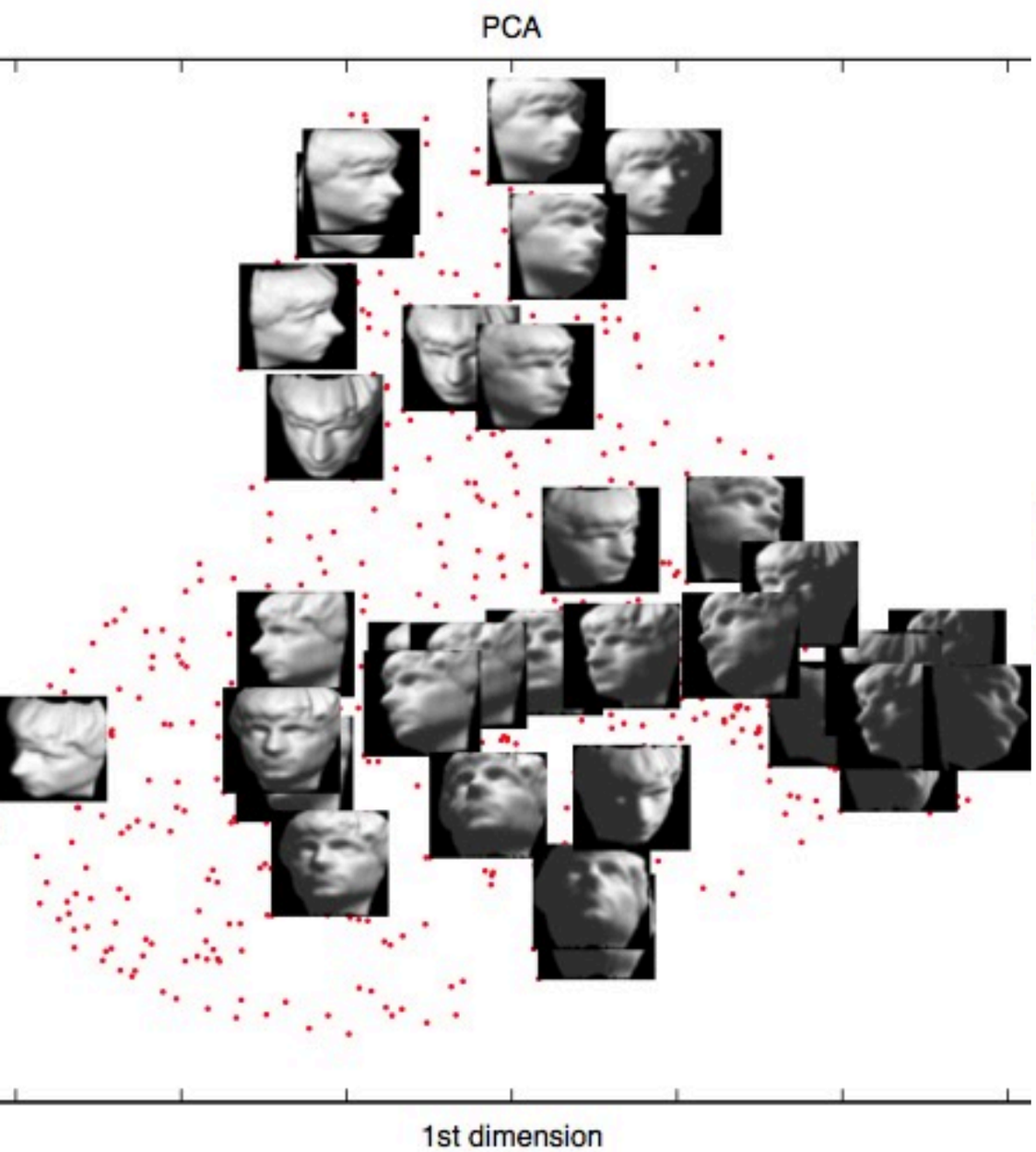


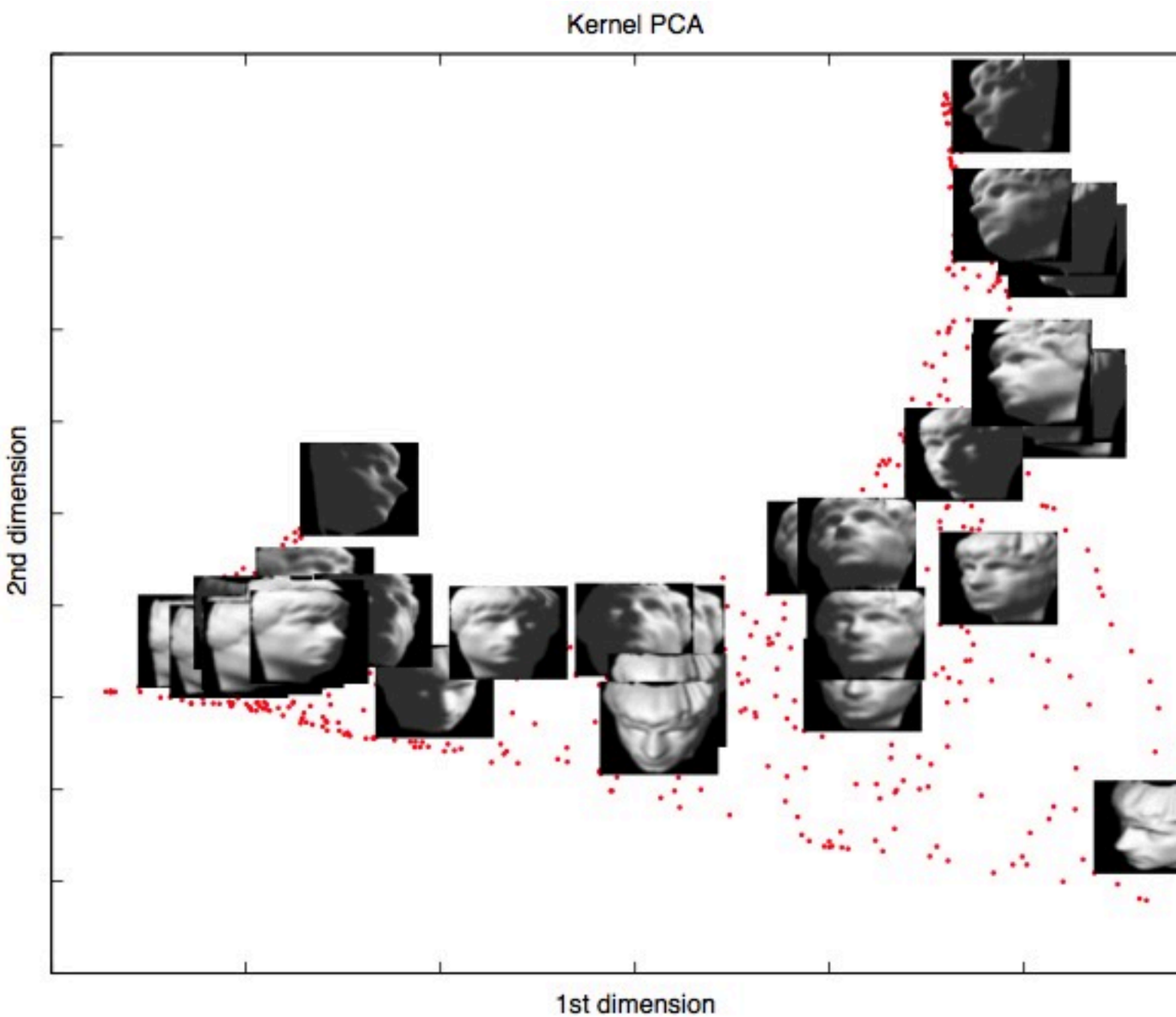
Linear vs. non-Linear

LLE

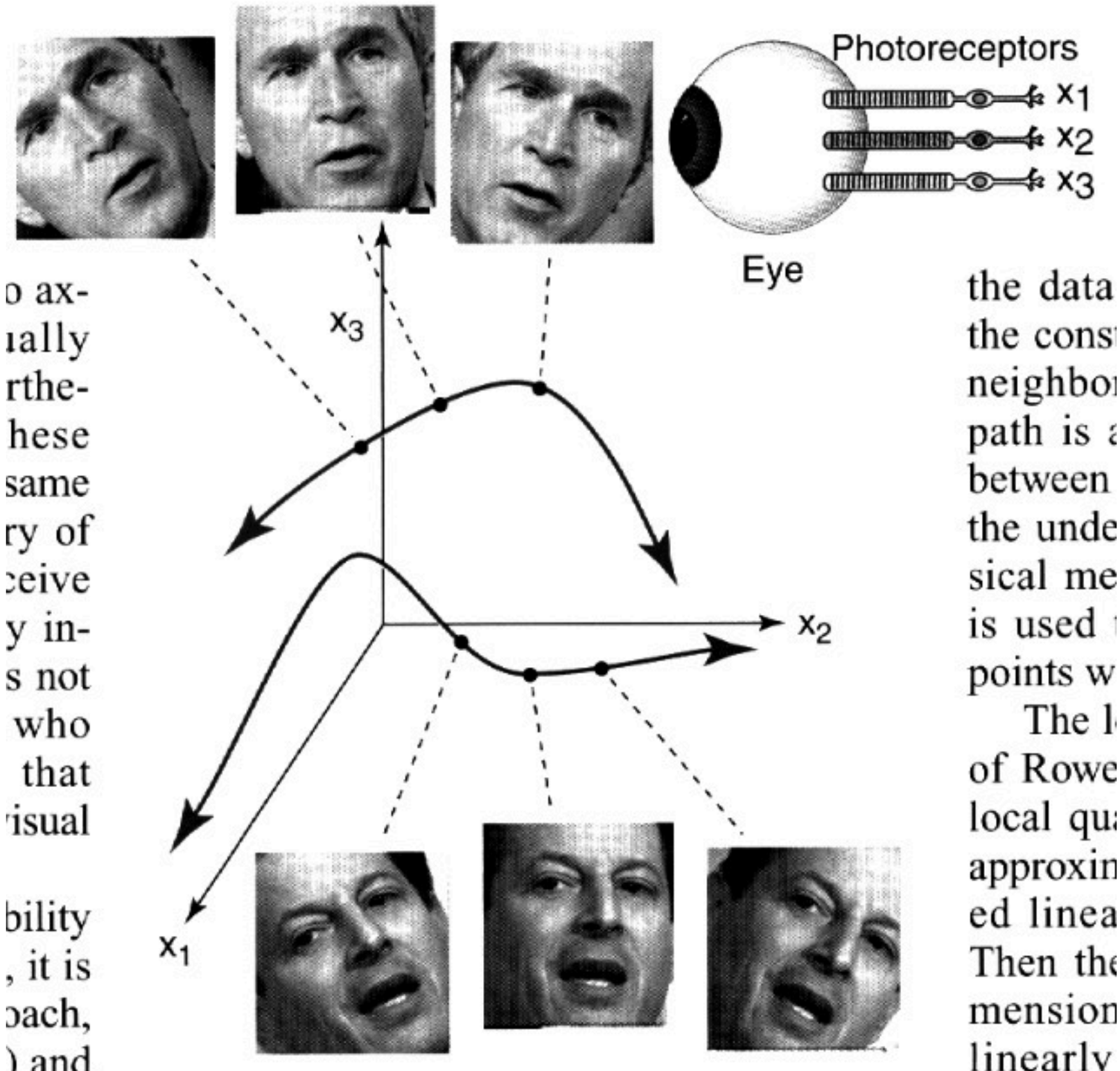


PCA vs Kernel PCA





Brain does non-linear dim. redux



o ax-
ially
rthe-
hese
same
ry of
ceive
y in-
s not
who
that
visual

bility
, it is
oach,
) and
pages

the data
the cons
neighbo
path is
between
the unde
sical me
is used
points w
The l
of Rowe
local qu
approxi
ed linea
Then the
mension
linearly
with the

Manifolds in visual perception. The retinal im-