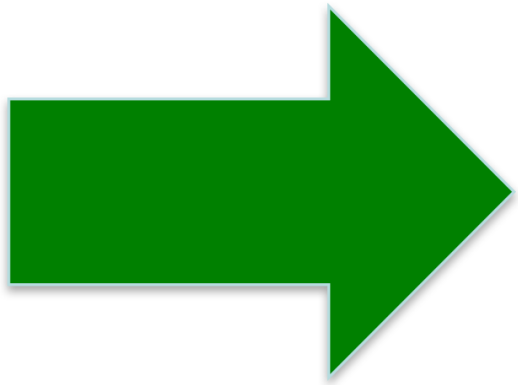
Unsupervised Learning

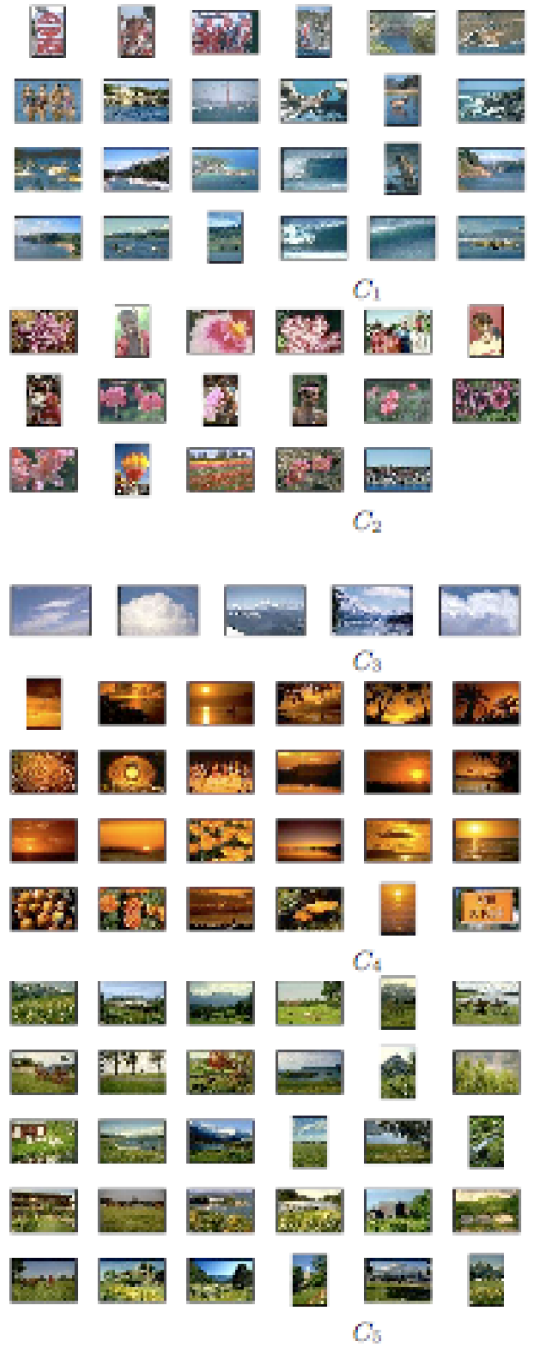
ECE 449

Unsupervised Learning

- Clustering: group similar things
  - Finding structure in data
  - Applications: group search results or customers, find anomalies
- Feature projections: dimensionality reduction
  - Feature reduction that preserves structure in data
  - Applications: improved learning, visualizing data
- General theme: no labels

Clustering Images





[Goldberger et al.]

Clustering for Segmentation

$K = 2$

$K = 3$

$K = 10$

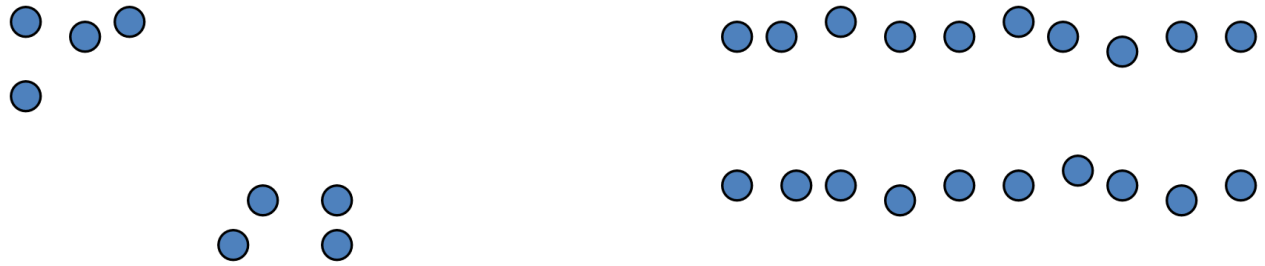
Original image

Unsupervised Learning: Clustering

- (Dis)similarity
- Hierarchical clustering
- K-means (partitioning) and its variants
- Soft clustering (EM for GMM)
- Density based clustering
- Evaluating clusters
- Principle components analysis (PCA)

Clustering

- Basic idea: group together similar instances
- Example: 2D point patterns



- What could “similar” mean?
  - For two different samples
  - Between a sample and a cluster

What could Similar Mean

- With respect to another sample
  - Small distance
    - Euclidean distance (L2), city block (L1)
  - High match
    - Correlation, cosine distance (equivalent to L2)
    - Feature overlap
- With respect to the cluster
  - Close to (all, some, avg) members of the group

Other Clustering Questions

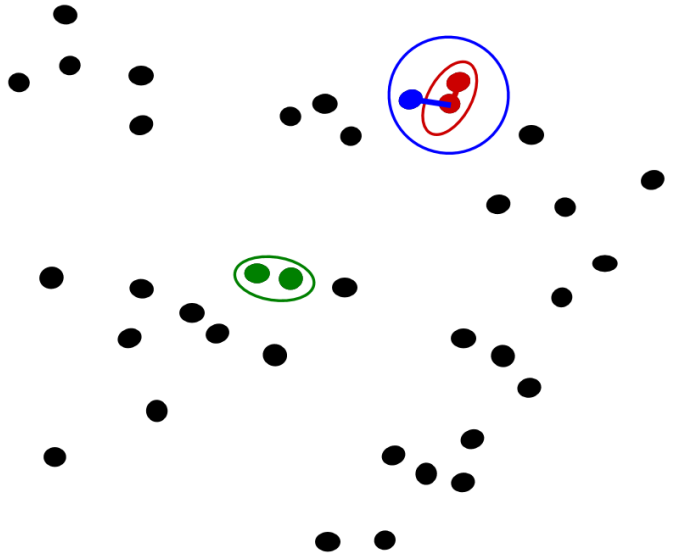
- Will the algorithm converge?
- Will it find the true patterns in the data?
- How many clusters to pick?
- How good are the clusters?

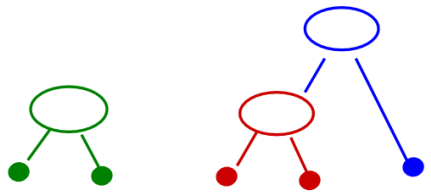
Hierarchical Clustering

- Agglomerative
  - Iteratively group samples
  - Good for different linkages, but expensive
- Divisive clustering
  - Iteratively divide samples (iterative K-means)
  - Better decisions for a small number of clusters
- Both produce a hierarchy –good for when you don’t know the # of clusters

Agglomerative Clustering

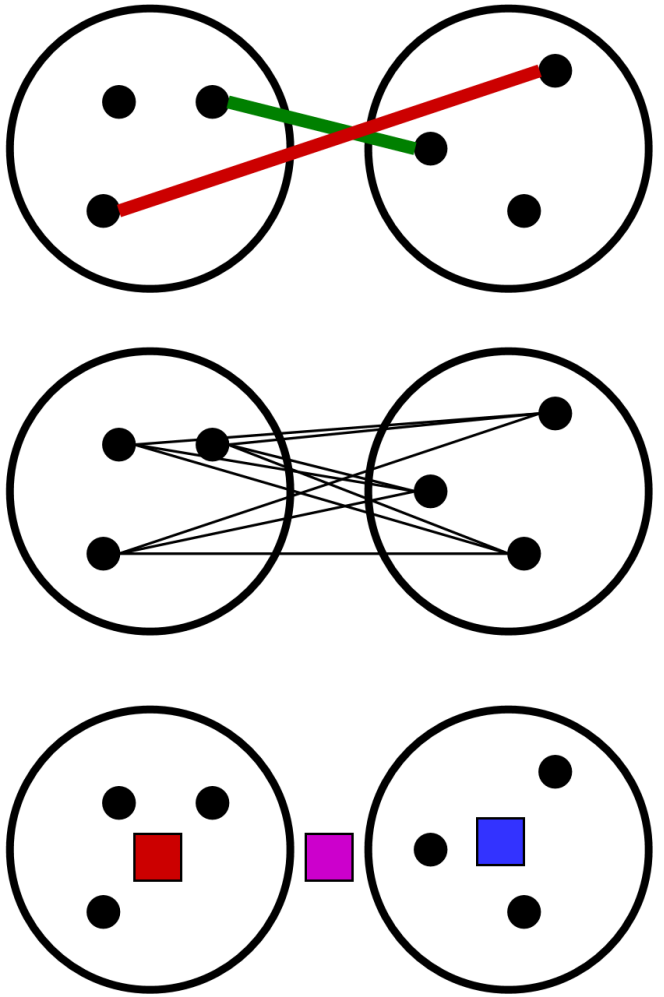
- Agglomerative clustering
  - First merge very similar instances
  - Incrementally build larger clusters out of smaller clusters
- Algorithm
  - Maintain a set of clusters
  - Initially, each instance in its own cluster
  - Repeat:
    - Pick the two closest clusters
    - Merge them into a new cluster
- Stop when there's only one cluster left
- Produces not one clustering, but a family of clusterings represented by a dendrogram



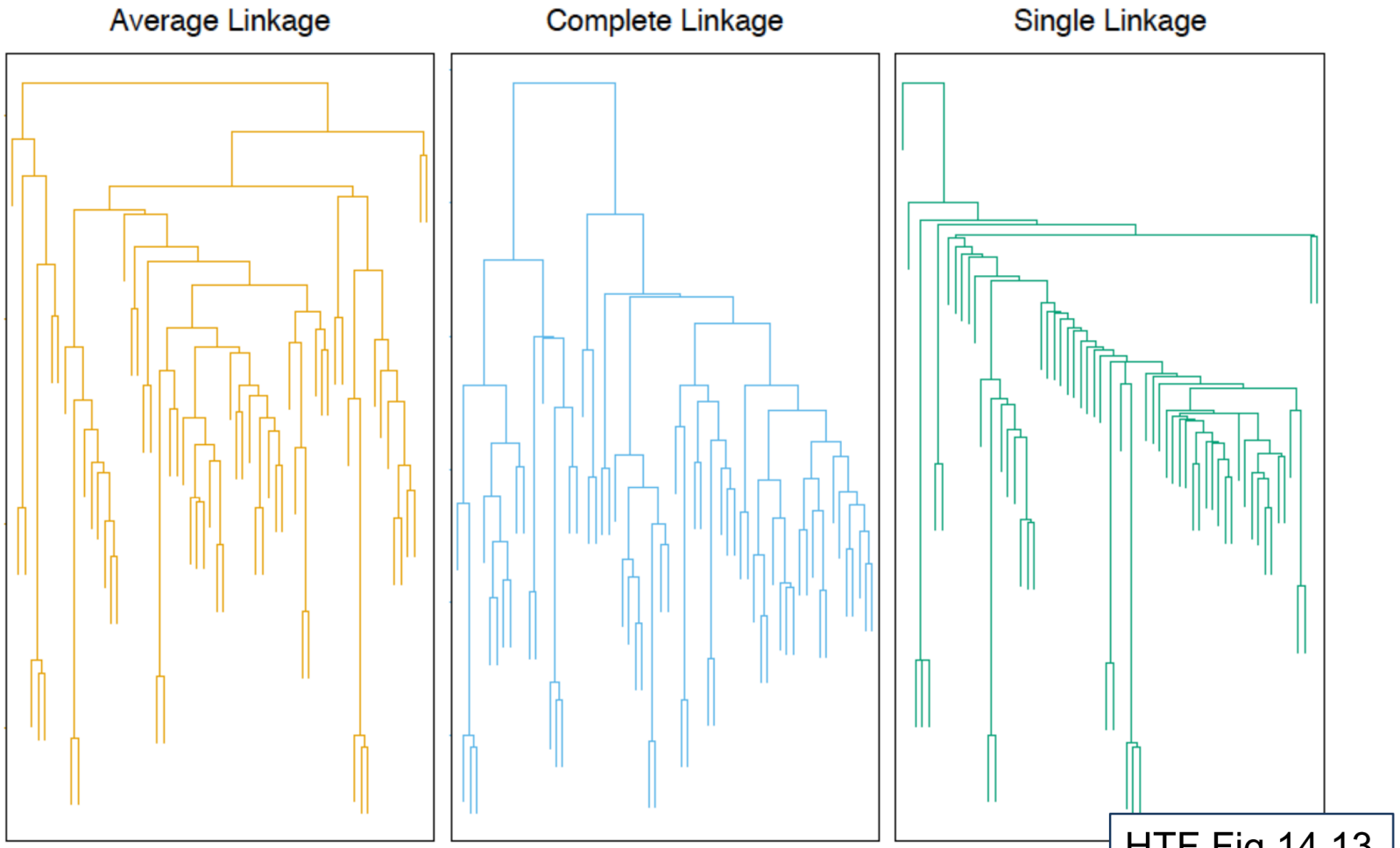


Agglomerative Clustering

- How should we define “closest” for clusters with multiple elements?
- Many options:
  - Closest pair (single-link clustering)
  - Farthest pair (complete-link clustering)
  - Average of all pairs
  - Ward’s method (min variance, like k-means)
- Different choices create different clustering behaviors



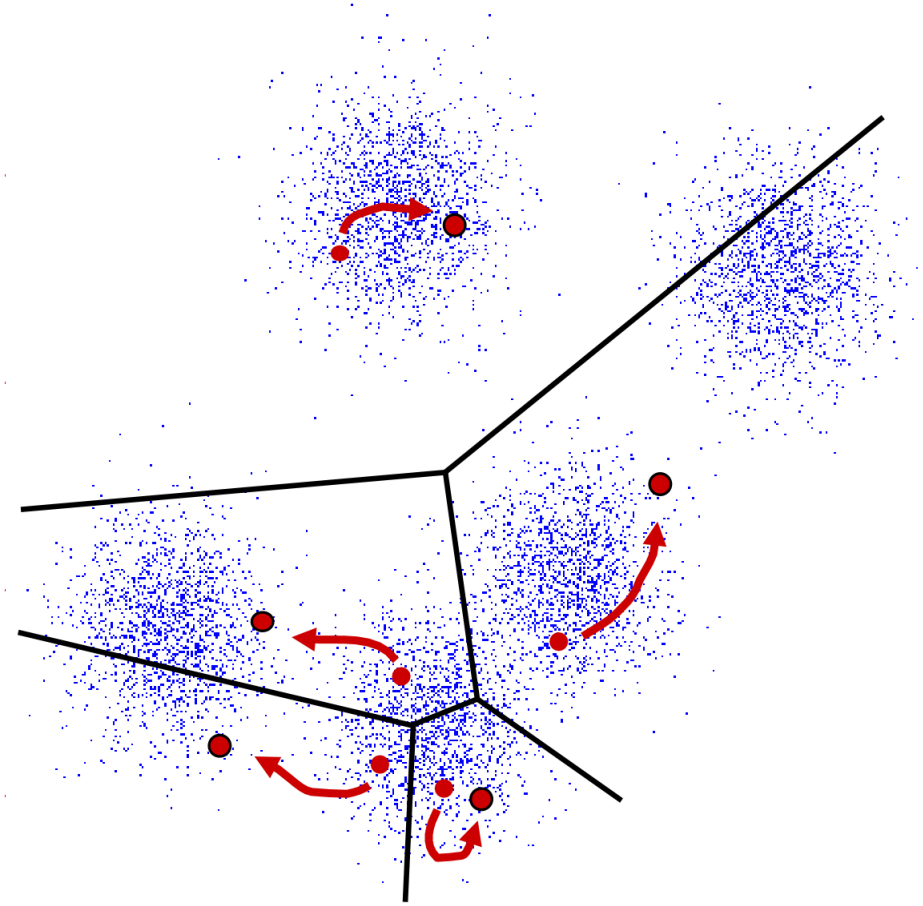
Agglomerative Clustering: Diff Objectives



HTF Fig 14.13

K-Means

- An iterative clustering algorithm
  - Pick K random points as cluster centers (means), $c^1 \ldots c^k$
  - Alternate:
    - Assign each example x^i to the mean c^j that is closest to it
    - Set each mean c^j to the average of its assigned points
  - Stop when no points' assignments change



K-Means

- Data:{x^j| j=1..n}
- An iterative clustering algorithm
  - Pick K random cluster centers, c^1…c^k
  - For t=1..T: [or, stop if assignments don't change]
    - for j = 1.. n: [recompute cluster assignments]

$$a^j = \arg\min_i \text{dist}(x^j, c^i)$$

    - for j= 1…k: [recompute cluster centers]

$$c^j = \frac{1}{|\{i|a^i = j\}|} \sum_{\{i|a^i=j\}} x^i$$

K-Means as Optimization

- Consider the total distance to the means

$$L\underbrace{\left(\{x^i\}\right.}_{\text{points}},\underbrace{\{a^i\}}_{\text{assignments}},\underbrace{\left.\{c^k\}\right)}_{\text{means}} = \sum_i \text{dist}(x^i, c^{a^i})$$

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

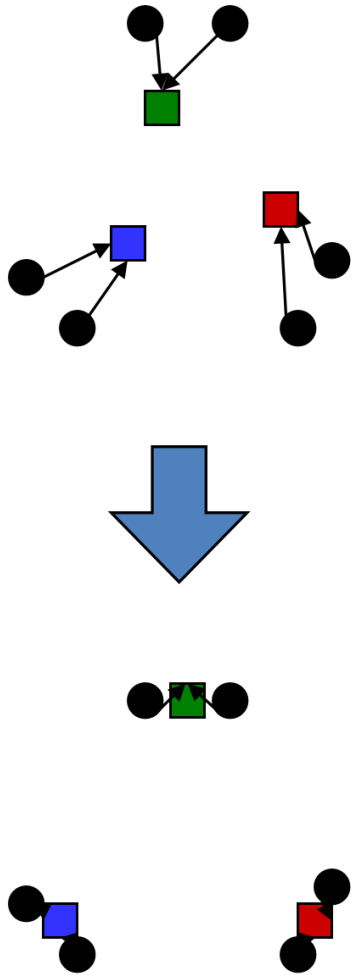
Phase I: Update Assignments

- For each point, re-assign to closest mean

$$a^i = \arg\min_j \mathrm{dist}\left(x^i, c^j\right)$$

- Can only decrease total distance L

$$L\left(\{x^i\}, \{a^i\}, \{c^k\}\right) = \sum_i \mathrm{dist}(x^i, c^{a^i})$$



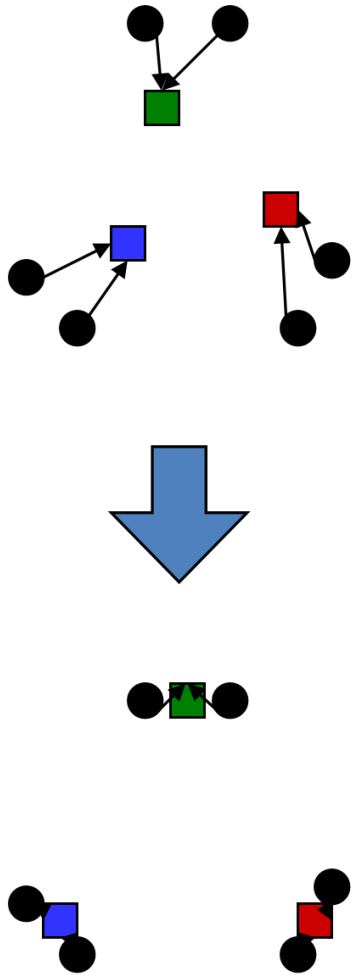
Phase II: Update Means

- Move each mean to the average of its assigned points

$$c^j = \frac{1}{|\{i|a^i = j\}|} \sum_{\{i|a^i=j\}} x^i$$

- Also can only decrease total distance

- Fun fact: the point y with minimum squared Euclidean distance to a set of points {x} is their mean

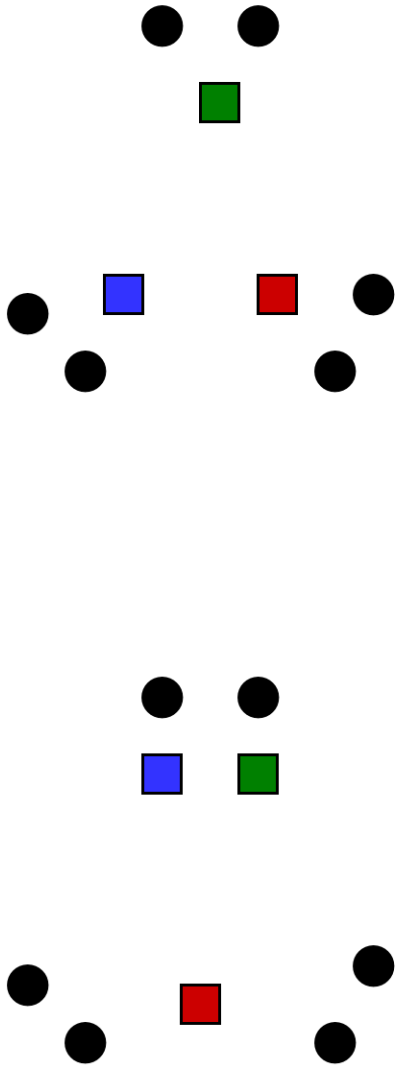


Questions

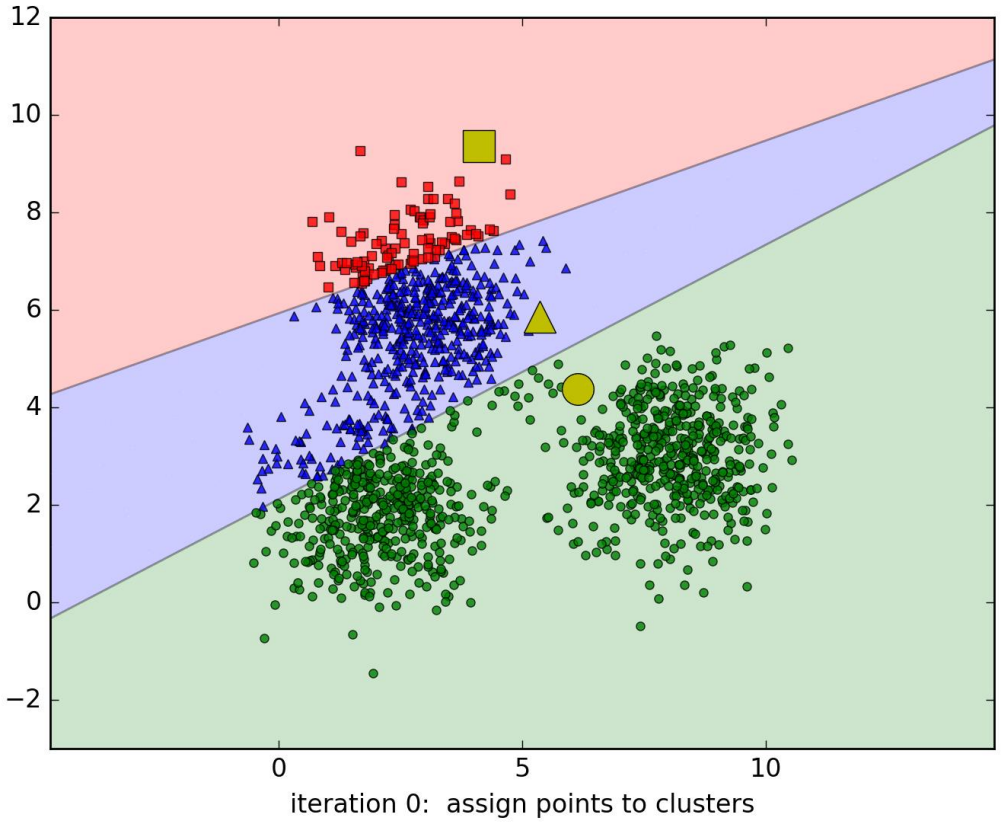
- Optimal solution?

Initialization

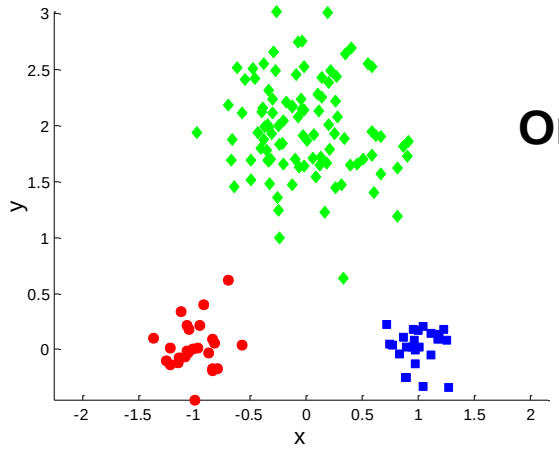
- K-means is non-deterministic
  - Requires initial means
  - It does matter what you pick!
- Various schemes for preventing this kind of thing:
  - Multiple random starts
  - Divisive clustering



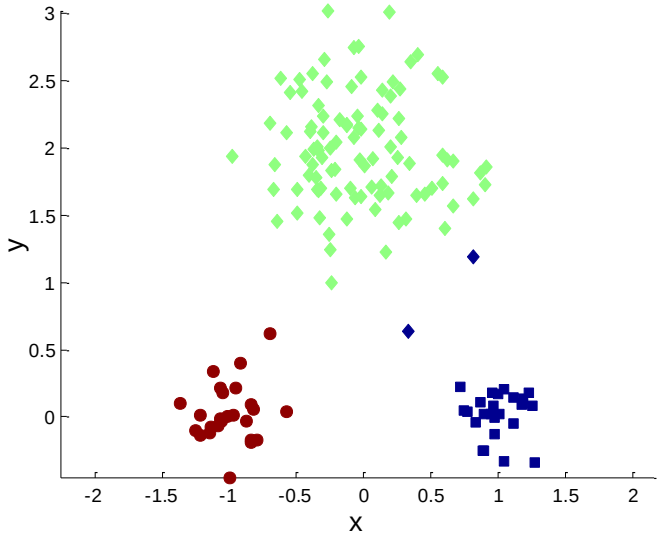
Examples



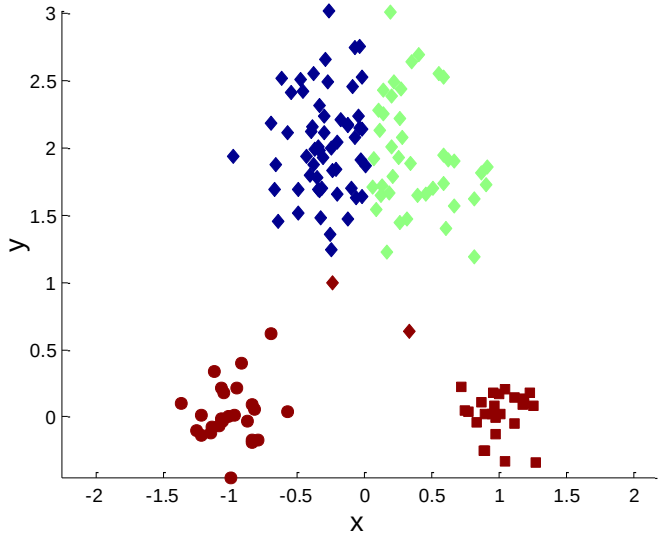
Importance of Choosing Initial Centroids



Original Points

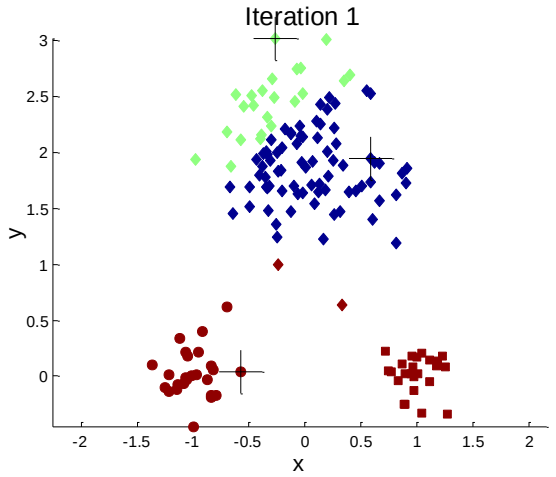


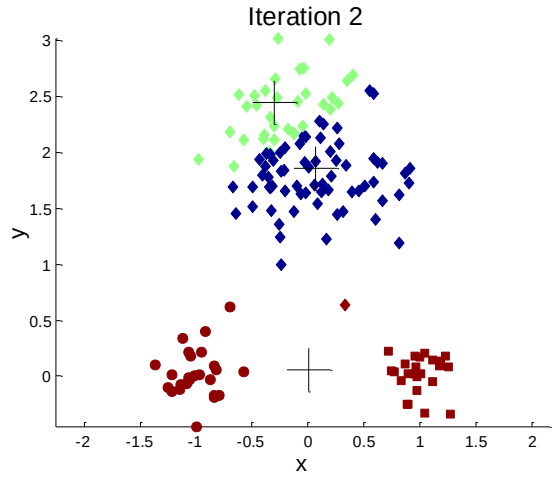
Optimal Clustering

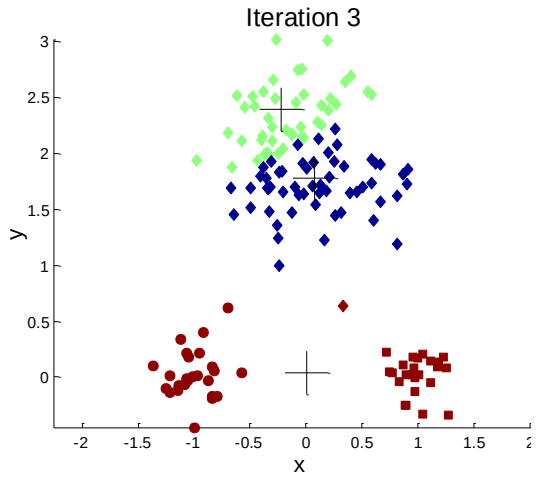


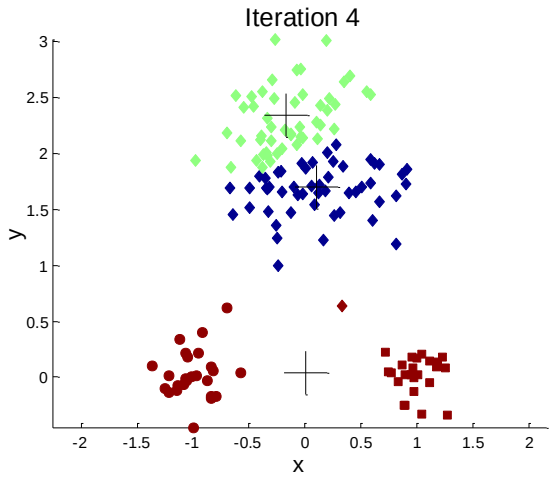
Sub-optimal Clustering

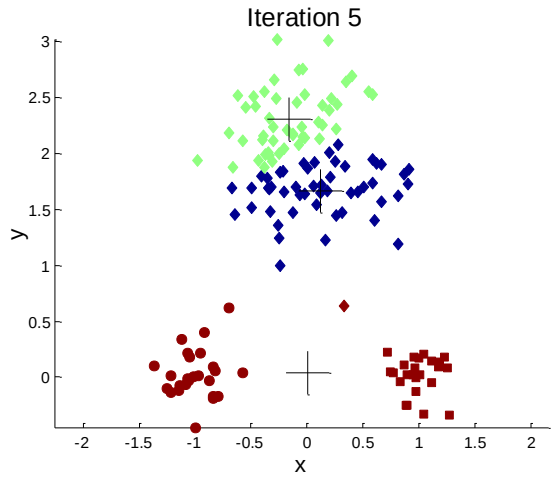
Importance of Choosing Initial Centroids











Solutions to Initial Centroids Problem

- Multiple runs
  - Helps, but probability is not on your side
- Use some strategy to select the k initial centroids and then select among these initial centroids
  - Select most widely separated
    - K-means++ is a robust way of doing this selection
  - Use hierarchical clustering to determine initial centroids

K-means++

- This approach can be slower than random initialization, but very consistently produces better results in terms of SSE
- To select a set of initial centroids, C, perform the following

Select an initial point at random to be the first centroid

For k – 1 steps

For each of the N points, x_i, $1 \leq i \leq \mathrm{N}$, find the minimum squared distance to the currently selected centroids, C_1, …, C_j, $1 \leq j < \mathrm{k}$, i.e., $\min_j d^2(C_j, x_i)$

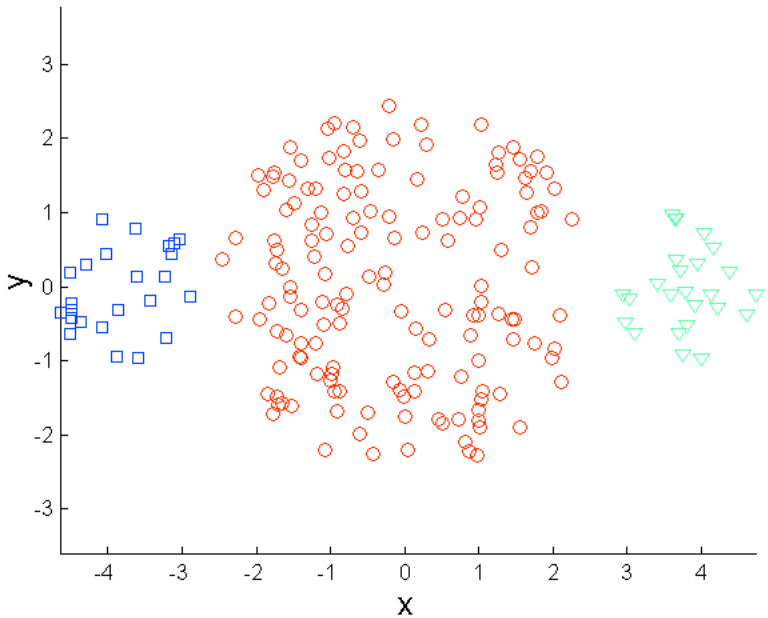
Randomly select a new centroid by choosing a point with probability proportional to $\frac{\min_j d^2(Cj, xi)}{\sum_i \min_j d^2(Cj, xi)}$

End For

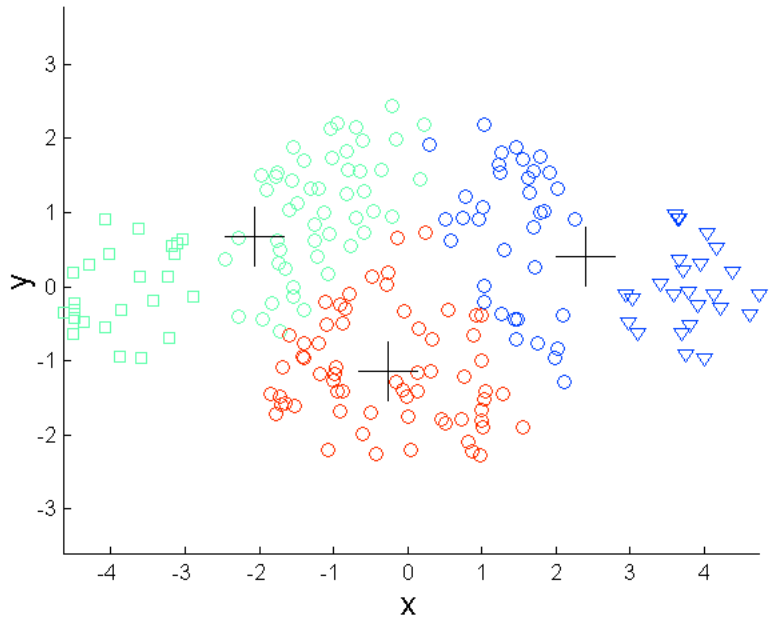
Limitations

- K-means has problems when clusters are of differing
  - Sizes
  - Densities
  - Non-globular shapes

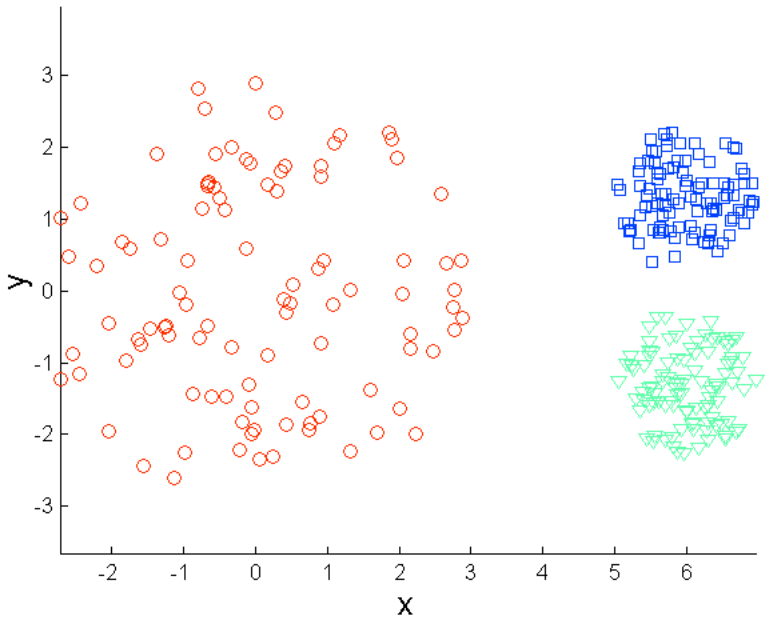
- K-means has problems when the data contains outliers.



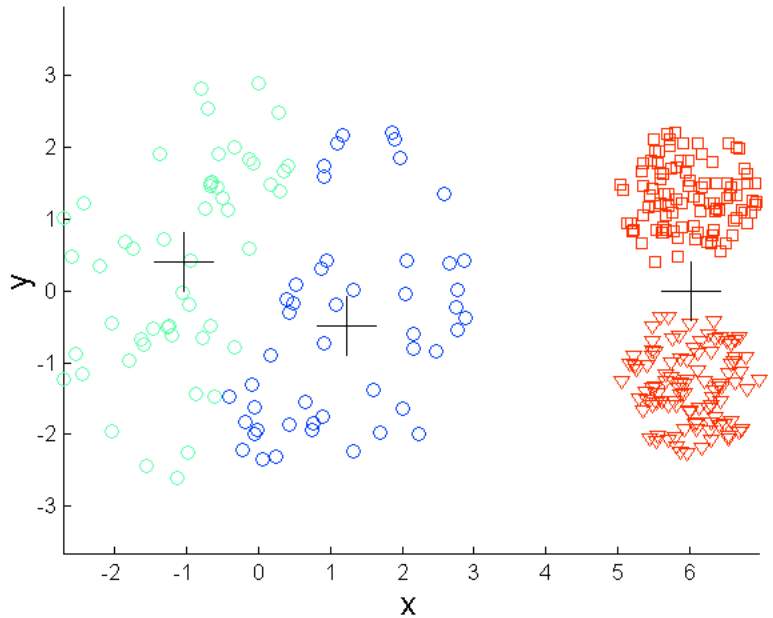
Original Points



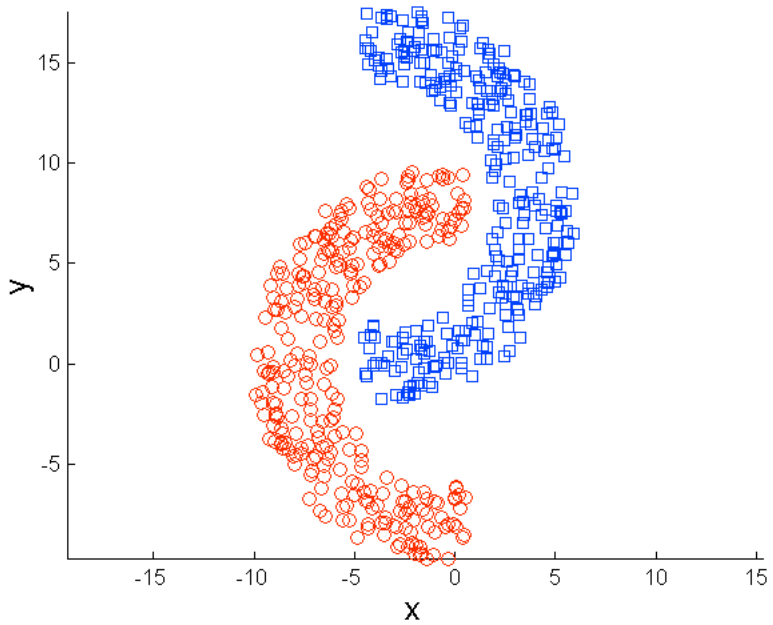
K-means (3 Clusters)



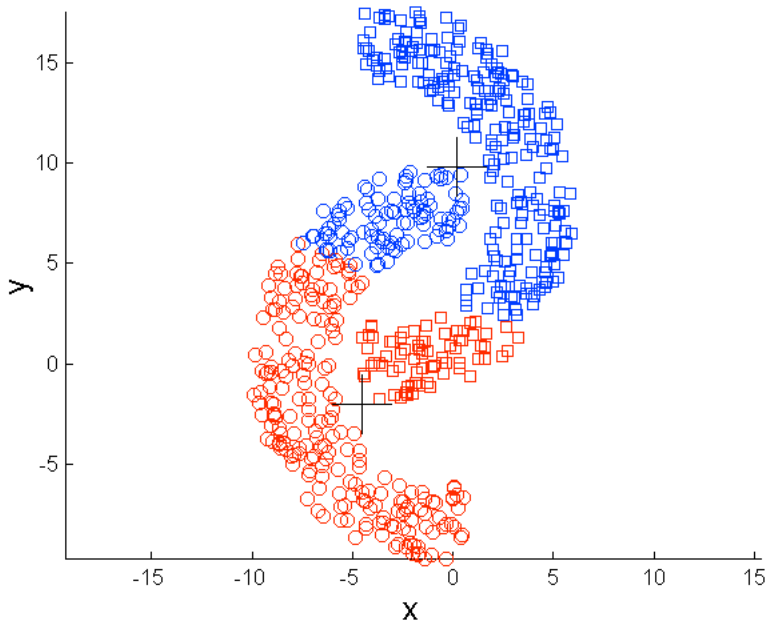
Original Points



K-means (3 Clusters)



Original Points

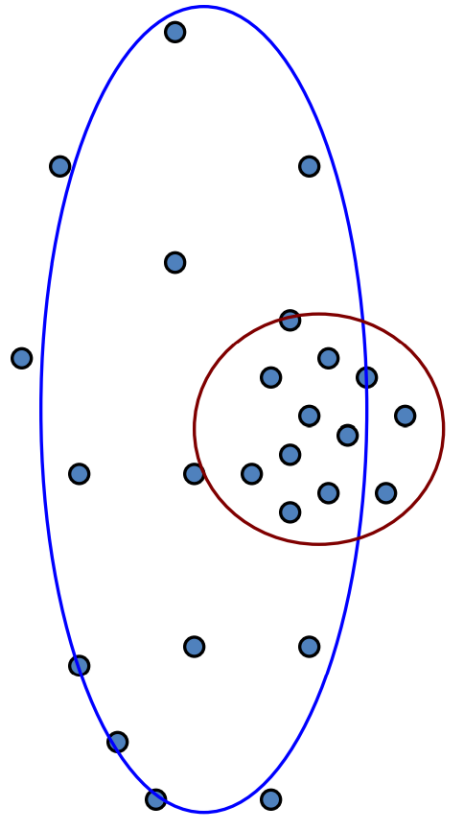


K-means (2 Clusters)

A Bad Case for “Hard” Assignments

- Clusters may overlap
- Some clusters may be “wider” than others
- Distances can be deceiving

- We can use a probabilistic model
  - Allows overlaps, clusters of different size, etc.



Hard Assignment vs Soft Assignment

K-Means

EM for GMMs

(with known Σ)

Iterate: p=0,1,2, …

For i=1,2, …, n

Hard decision

$$z^i = \underset{j}{\operatorname{argmin}}\, d(x^i, c_j | C^{(p)})$$

Soft decision

$$\gamma^i(j) = P(z^i = j | x^i, \theta^{(p)})$$

For j=1,2, …, K

$$c_j^{(p+1)} = \frac{1}{N_j} \sum_{i:z^i=j} x^i$$

$$N_j = \sum_{i=1}^{n} I(z^i = j)$$

$$\mu_j^{(p+1)} = \frac{1}{N_j} \sum_{i=1}^{n} \gamma^i(j) x^i$$

$$N_j = \sum_{i=1}^{n} \gamma^i(j)$$

Expectation-Maximization (EM) for Gaussian Mixture Models (GMM)

- GMM model

$$p(x|\theta) = \sum_{j=1}^{k} p_j N(x; \mu_j, \Sigma_j)$$

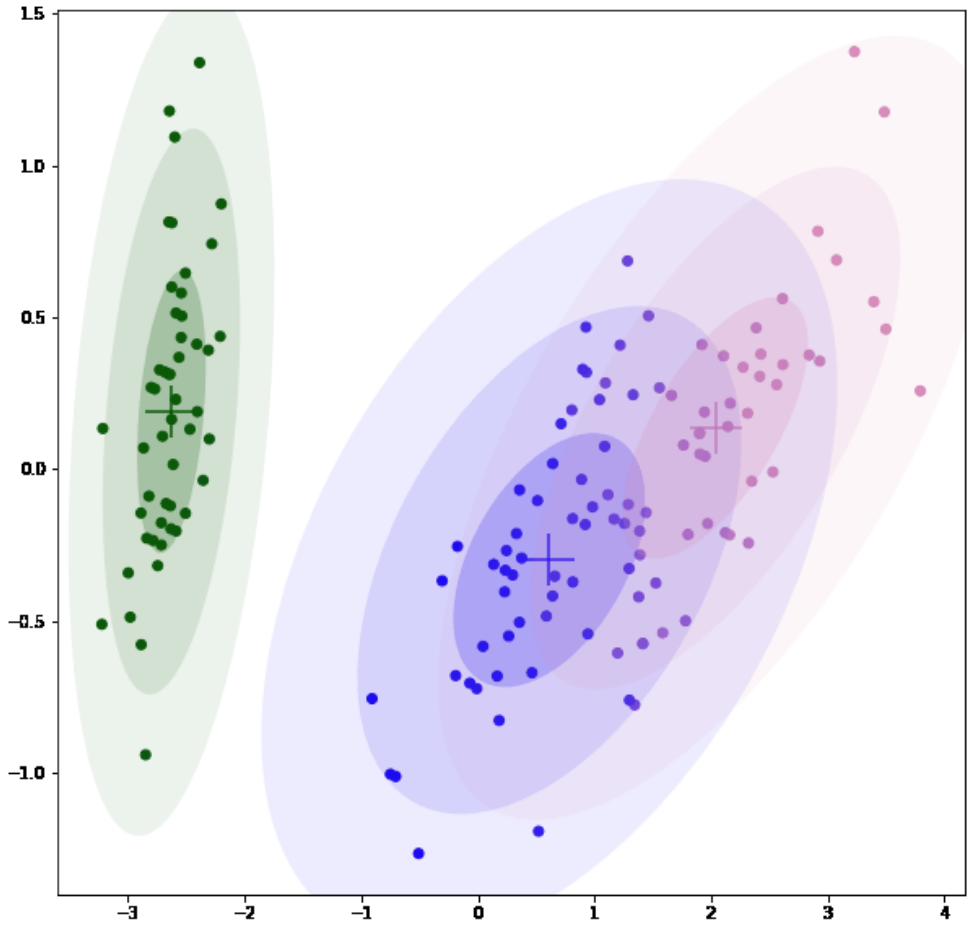
- Expectation

$$q_{i,j} = \frac{p_j N(x; \mu_j, \Sigma_j)}{\sum_{j=1}^{k} p_j N(x; \mu_j, \Sigma_j)}$$

- Maximization

$$\mu_j = \frac{\sum_i q_{i,j} x_i}{\sum_i q_{i,j}}, \Sigma_j = \frac{\sum_i q_{i,j} (x_i - \mu_j)(x_i - \mu_j)^T}{\sum_i q_{i,j}}, p_j = \frac{1}{m} \sum_i q_{i,j}$$

Example



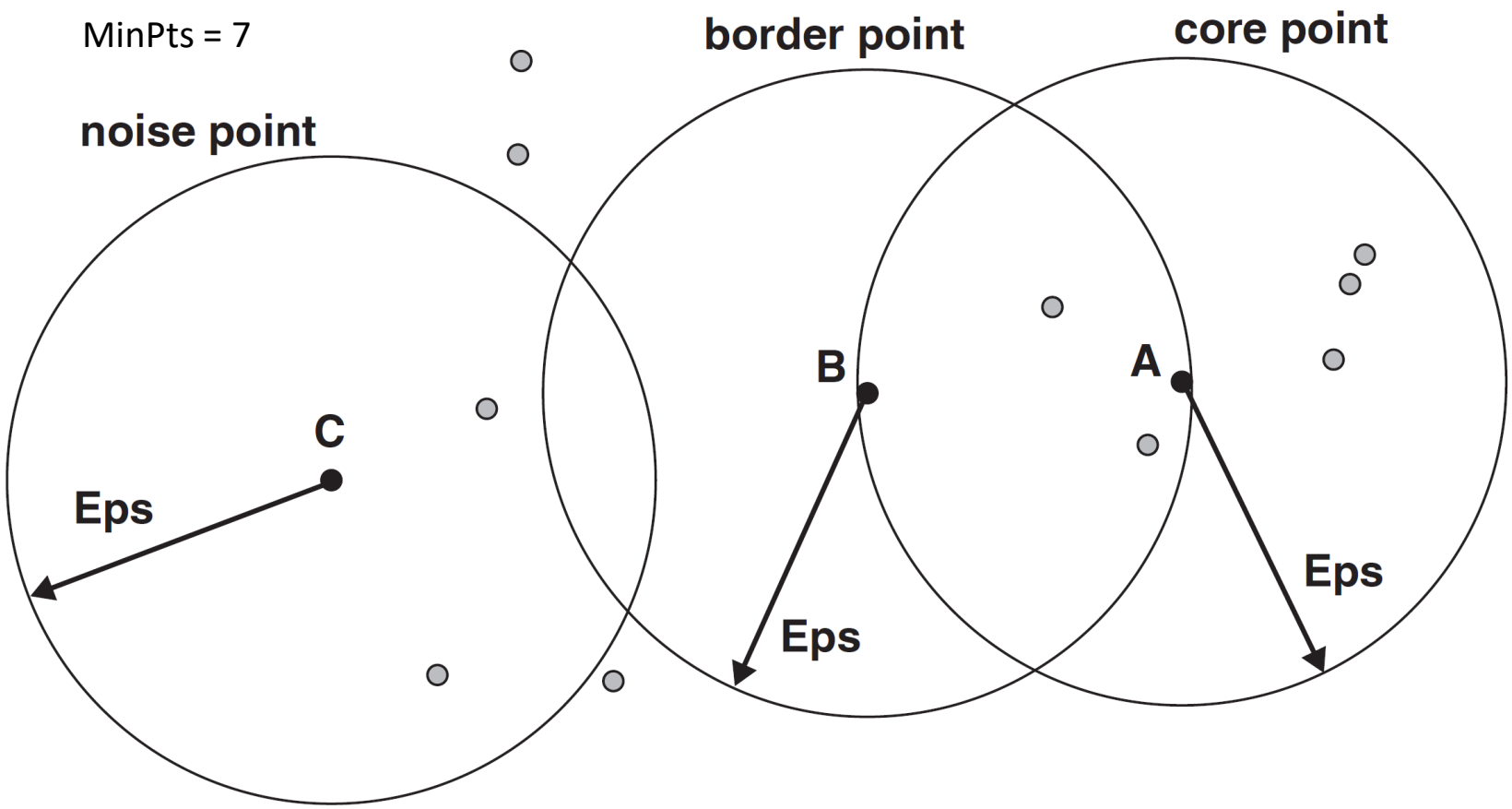
Density Based Clustering

- Clusters are regions of high density that are separated from one another by regions on low density.

DBSCAN

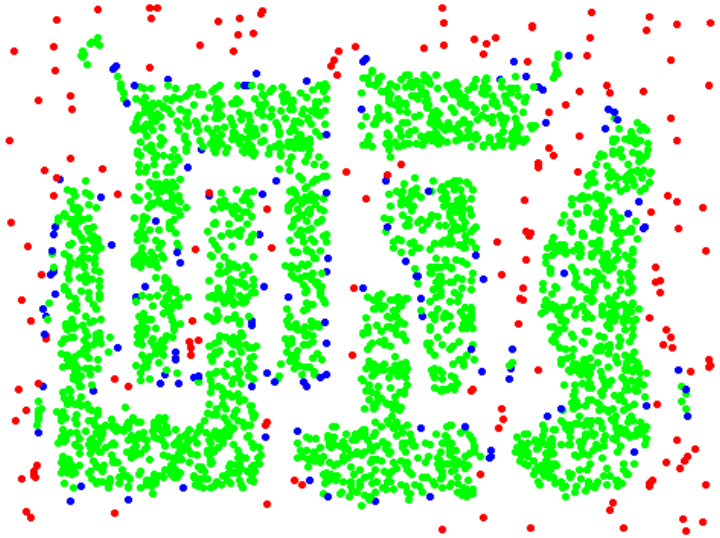
- DBSCAN is a density-based algorithm.
  - Density = number of points within a specified radius (Eps)
  - A point is a core point if it has at least a specified number of points (MinPts) within Eps
    - These are points that are at the interior of a cluster
    - Counts the point itself
  - A border point is not a core point, but is in the neighborhood of a core point
  - A noise point is any point that is not a core point or a border point

DBSCAN: Core, Border, and Noise Points



DBSCAN: Core, Border and Noise Points

Original Points



Point types: core, border and noise

Eps = 10, MinPts = 4

DBSCAN Algorithm

- Form clusters using core points, and assign border points to one of its neighboring clusters

1: Label all points as core, border, or noise points.

2: Eliminate noise points.

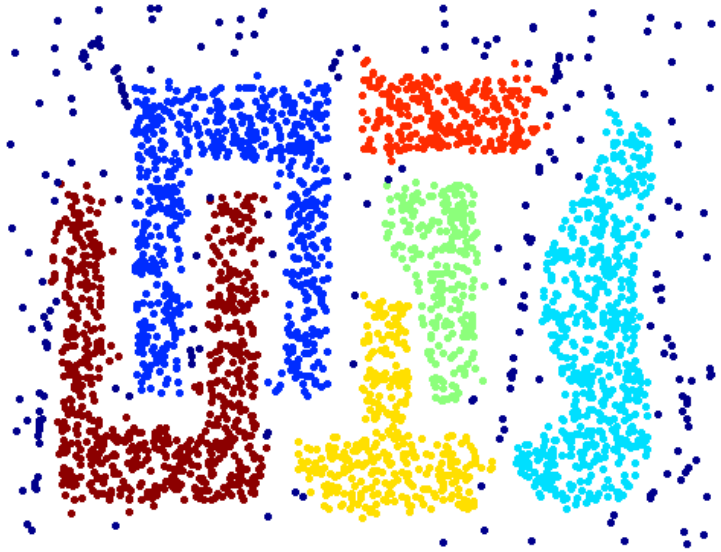
3: Put an edge between all core points within a distance *Eps* of each other.

4: Make each group of connected core points into a separate cluster.

5: Assign each border point to one of the clusters of its associated core points

When DBSCAN Works Well

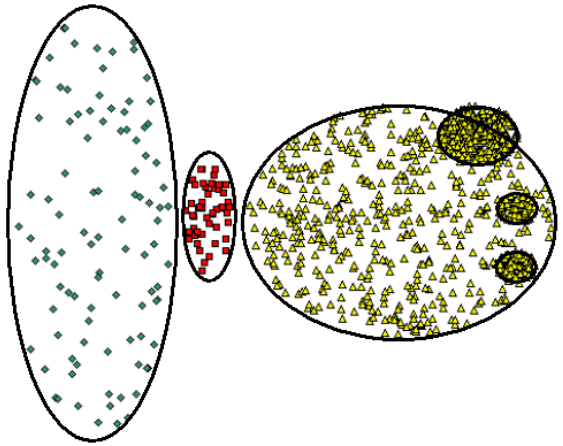
Original Points



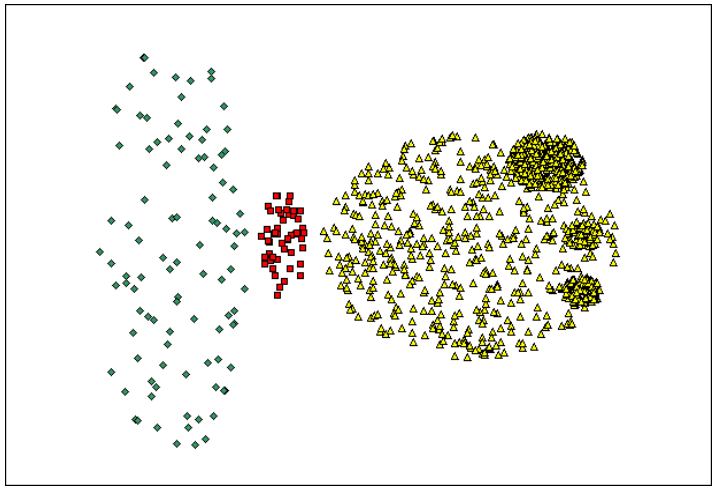
Clusters

- **Resistant to Noise**
- **Can handle clusters of different shapes and sizes**

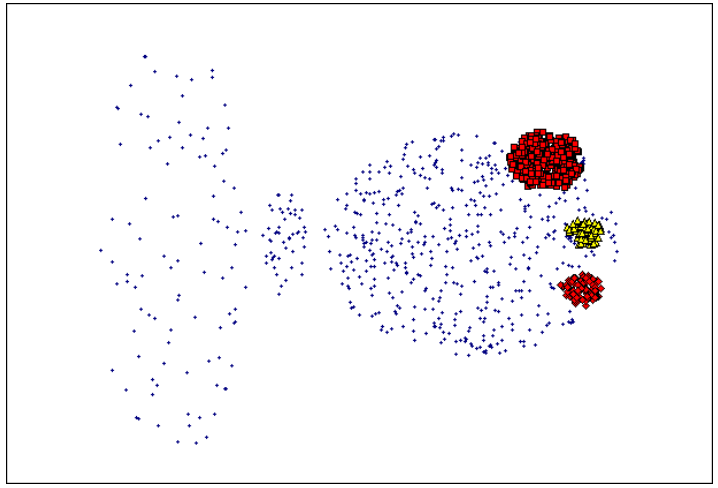
When DBSCAN Does NOT Work Well



Original Points



(MinPts=4, Eps=9.92).



(MinPts=4, Eps=9.75)

- **Varying densities**
- **High-dimensional data**

Evaluating Clusters

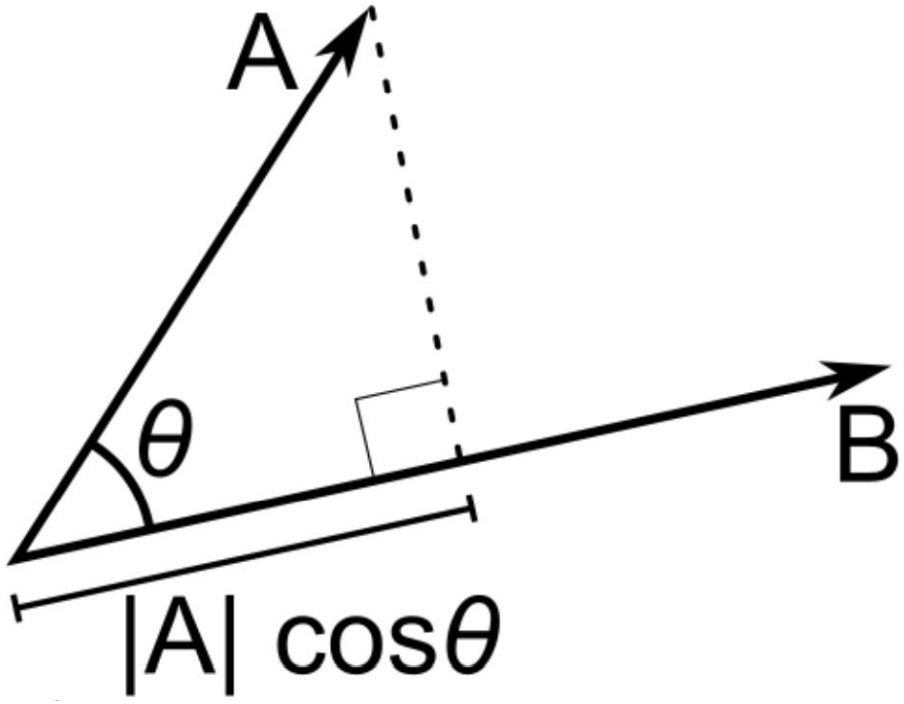
- Two options:
  - Compare to labeled data (if you have it)
  - Plug into some application & show improvement
- Labeled data options
  - Cluster purity: Label cluster by most likely class
    - $p_i = \max_j p_{ij}$, $P = \sum_i \frac{N_i}{N} p_i$
  - Rand Index: count pairs in same/diff clusters
    - $R = \frac{TP+TN}{TP+TN+FP+FN}$
  - …

Feature Projections

- General idea: project n-dim data into k-dim space while preserving information
- Projections to min reconstruction error
  - Principle Components Analysis (PCA)
  - Neural network embeddings

Reminder: Vector Projections

- Basic definitions:
  - A.B = |A||B|cos θ
- Assume |B|=1 (unit vector)
  - A.B = |A|cos θ
- So, dot product is length of projection



Linear Projections

- Project a point into a (lower dimensional) space
  - point: $\mathbf{x} = (x_1,\ldots,x_n)$
  - Select a basis – set of unit (length 1) basis vectors $(\mathbf{u}_1,\ldots,\mathbf{u}_k)$
    - We consider orthonormal basis
      - $\mathbf{u}_i^t\mathbf{u}_i = 1, \mathbf{u}_j^t\mathbf{u}_i = 0, \forall i \neq j$
  - Select a center – $\bar{\mathbf{x}}$, defines offset of space
  - Best coordinates in lower dimensional space
    - Defined by dot-products: $(z_1,\ldots,z_k)$, $z_j^i = \left(\mathbf{x}^i - \bar{\mathbf{x}}\right)^t \mathbf{u}_j$

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

PCA Finds Projection that Minimizes Reconstruction Error

- Given m data points: $\mathbf{x}^i = \left(x_1^i, \ldots, x_n^i\right), i = 1, \ldots, m$
- Will represent each point as a projection
  - $\hat{\mathbf{x}}^i = \bar{\mathbf{x}} + \sum_{j=1}^{k} z_j^i \mathbf{u}_j$
  - $\bar{\mathbf{x}} = \frac{1}{m}\sum_{i=1}^{m} \mathbf{x}^i \, , z_j^i = \left(\mathbf{x}^i - \bar{\mathbf{x}}\right)^t \mathbf{u}_j$
- PCA
  - Given k<n, find (u1,…,uk) minimizing reconstruction error

$$err_k = \sum_{i=1}^{m} \left\| \mathbf{x}^i - \hat{\mathbf{x}}^i \right\|^2$$

Understanding the Reconstruction Error

- Note that $\mathbf{x}^i$ can be represented exactly by n-dimensional projection
  - $\mathbf{x}^i = \bar{\mathbf{x}} + \sum_{j=1}^{n} z_j^i \mathbf{u}_j$
- Rewriting error
  - $err_k = \sum_{i=1}^{m} \left\| \mathbf{x}^i - \hat{\mathbf{x}}^i \right\|^2 = \sum_{i=1}^{m} \left\| \mathbf{x}^i - \left(\bar{\mathbf{x}} + \sum_{j=1}^{k} z_j^i \mathbf{u}_j \right) \right\|^2 = \sum_{i=1}^{m} \left\| \left(\bar{\mathbf{x}} + \sum_{j=1}^{n} z_j^i \mathbf{u}_j \right) - \left(\bar{\mathbf{x}} + \sum_{j=1}^{k} z_j^i \mathbf{u}_j \right) \right\|^2 = \sum_{i=1}^{m} \left\| \sum_{j=k+1}^{n} z_j^i \mathbf{u}_j \right\|^2 = \sum_{i=1}^{m} \sum_{j=k+1}^{n} \left(z_j^i \right)^2 = \sum_{i=1}^{m} \sum_{j=k+1}^{n} \left(\left(\mathbf{x}^i - \bar{\mathbf{x}} \right)^t \mathbf{u}_j \right)^2$
- Note
  - $\mathbf{u}_j^t \mathbf{u}_i = 1$ if i==j, and zero otherwise, because $\mathbf{u}$'s are an orthonormal basis
  - Error is sum of squared weights that would have be used for dimensions that are cut

Reconstruction Error and Covariance Matrix

- $err_k = \sum_{i=1}^{m} \sum_{j=k+1}^{n} \left(\left(\mathbf{x}^i - \bar{\mathbf{x}} \right)^t \mathbf{u}_j \right)^2 = \sum_{i=1}^{m} \sum_{j=k+1}^{n} \mathbf{u}_j^t \left(\mathbf{x}^i - \bar{\mathbf{x}} \right) \left(\mathbf{x}^i - \bar{\mathbf{x}} \right)^t \mathbf{u}_j = \sum_{j=k+1}^{n} \mathbf{u}_j^t \left[\sum_{i=1}^{m} \left(\mathbf{x}^i - \bar{\mathbf{x}} \right) \left(\mathbf{x}^i - \bar{\mathbf{x}} \right)^t \right] \mathbf{u}_j = m \sum_{j=k+1}^{n} \mathbf{u}_j^t \Sigma \mathbf{u}_j$
- To find the $\mathbf{u}_j$, we minimize
  - $\mathbf{u}^t \Sigma \mathbf{u} + \lambda (1 - \mathbf{u}^t \mathbf{u})$
- Take derivative, set equal to 0, solutions are eigenvectors
  - $\Sigma \mathbf{u}_i = \lambda_i \mathbf{u}_i$

Minimizing Reconstruction Error

- Minimizing reconstruction error equivalent to picking orthonormal basis ($\mathbf{u}_1,\ldots,\mathbf{u}_k$) minimizing
  - $err_k = m\sum_{j=k+1}^{n} \mathbf{u}_j^t \Sigma \mathbf{u}_j$
- Solutions: eigenvectors
  - $\Sigma \mathbf{u}_i = \lambda_i \mathbf{u}_i$
- So, minimizing reconstruction error is equivalent to picking ($\mathbf{u}_{k+1},\ldots,\mathbf{u}_n$) to be eigenvectors with smallest eigenvalues
- And, our projection should be onto the ($\mathbf{u}_1,\ldots,\mathbf{u}_k$) with the largest values

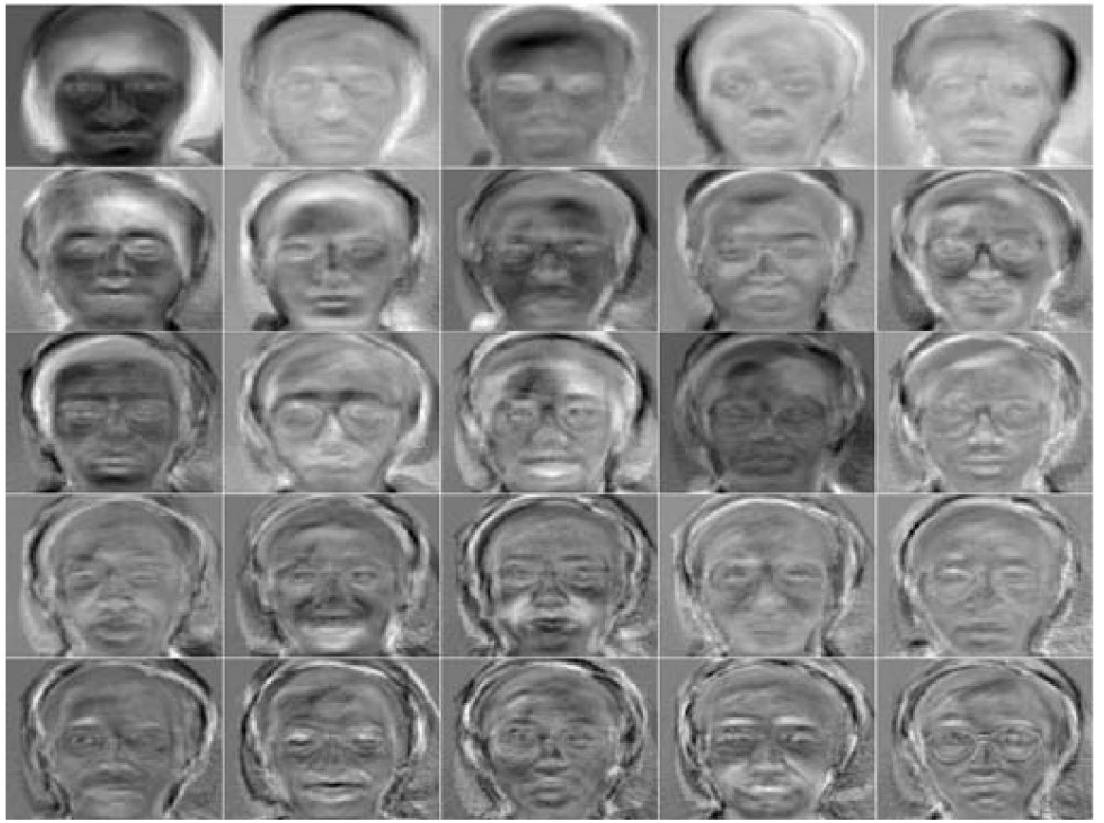
Basic PCA algorithm

- Start from m by n data matrix **X**
- Recenter: subtract mean from each row of **X**
  - $\mathbf{X}_c \leftarrow \mathbf{X} - \widehat{\mathbf{X}}$
- Compute covariance matrix:
  - $\Sigma = \frac{1}{m}\mathbf{X}_c^T\mathbf{X}$
- Find eigenvectors and values of Σ
- Principal components: k eigenvectors with highest eigenvalues

Eigenfaces

Input Images

Principal components



Eigenfaces Reconstruction

- Each image corresponds to adding together the principal components