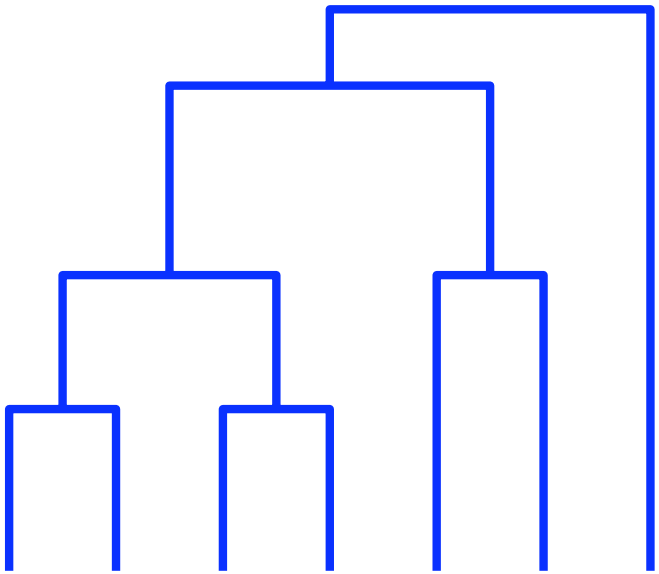
COMP 652: Machine Learning

Lecture 15

Today

Hierarchical clustering

- Organizes data instances into trees.
- For visualization, exploratory data analysis.
- **Agglomerative methods** build the tree bottom-up, successively grouping together the clusters deemed most similar.
- **Divisive methods** build the tree top-down, recursively partitioning the data.

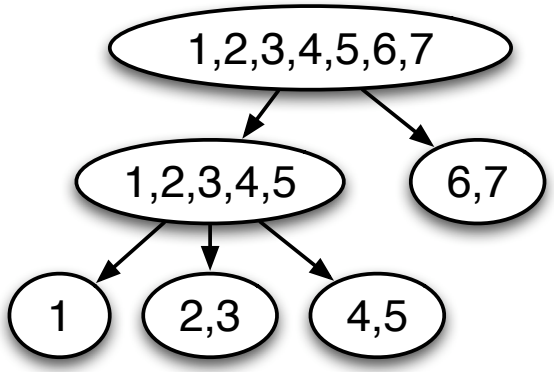


What is a hierarchical clustering?

- Given instances $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$.
- A hierarchical clustering is a set of subsets (clusters) of D, $C = \{C_1, \ldots, C_K\}$, where
  - Every element in D is in at least one set of C (the root)
  - The C_j can be assigned to the nodes of a tree such that the cluster at any node is precisely the union of the clusters at the node's children (if any).

Example of a hierarchical clustering

- Suppose $D = \{1, 2, 3, 4, 5, 6, 7\}$.
- One hierarchical clustering is
  $C = \{\{1\}, \{2, 3\}, \{4, 5\}, \{1, 2, 3, 4, 5\}, \{6, 7\}, \{1, 2, 3, 4, 5, 6, 7\}\}$.



- In this example:
  - Leaves of the tree need not correspond to single instances.
  - The branching factor of the tree is not limited.
- However, most hierarchical clustering algorithms produce binary trees, and take single instances as the smallest clusters.

Agglomerative clustering

- Input: Pairwise distances $d(\mathbf{x}, \mathbf{x}')$ between a set of data objects $\{\mathbf{x_i}\}$.
- Output: A hierarchical clustering
- Algorithm:
  - Assign each instance as its own cluster on a working list W.
  - Repeat
    - Find the two clusters in W that are most "similar".
    - Remove them from W.
    - Add their union to W.

    Until W contains a single cluster with all the data objects.
  - The hierarchical clustering contains *all* clusters appearing in W at any stage of the algorithm.

How do we measure dissimilarity between clusters?

- Distance between nearest objects ("Single-linkage" agglomerative clustering, or "nearest neighbor"):

$$\min_{\mathbf{x} \in C, \mathbf{x}' \in C'} d(\mathbf{x}, \mathbf{x}')$$

- Distance between farthest objects ("Complete-linkage" agglomerative clustering, or "furthest neighbor"):

$$\max_{\mathbf{x} \in C, \mathbf{x}' \in C'} d(\mathbf{x}, \mathbf{x}')$$

- Average distance between objects ("Group-average" agglomerative clustering):

$$\frac{1}{|C||C'|} \sum_{\mathbf{x} \in C, \mathbf{x}' \in C'} d(\mathbf{x}, \mathbf{x}')$$

Examples

Show examples!

Intuitions about cluster similarity

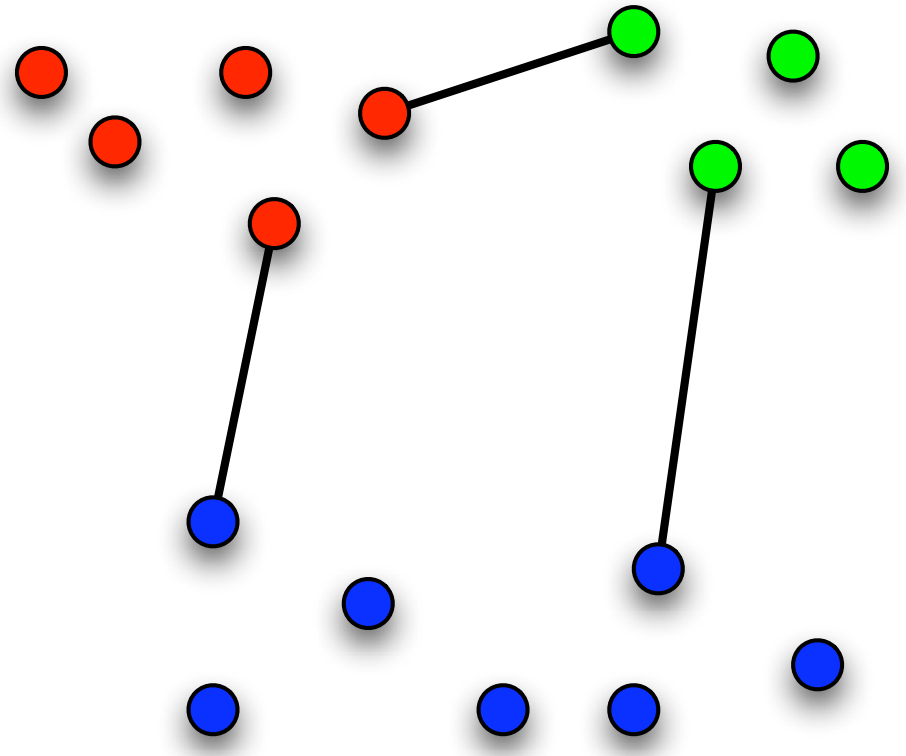
- Single-linkage
  - Favors spatially-extended / filamentous clusters
  - Often leaves singleton clusters until near the end
- Complete-linkage favors compact clusters
- Average-linkage is somewhere in between

Monotonicity

- Single-linkage, complete-linkage and group-average dissimilarity measure all share a monotonicity property:
  - Let A, B, C be clusters.
  - Let d be one of the dissimilarity measures.
  - If $d(A, B) \leq d(A, C)$ and $d(A, B) \leq d(B, C)$, then $d(A, B) \leq d(A \cup B, C)$.

Monotonicity of Single-linkage criterion

Proof by picture:



Monotonicity of Single-linkage criterion

More formal proof:

- We are given that $d(A, B) \leq d(A, C)$ and $d(A, B) \leq d(B, C)$
- Then:

$$\begin{aligned} d(A \cup B, C) &= \min_{x \in A \cup B, x' \in C} d(x, x') \\ &= \min \left(\min_{x_a \in A, x' \in C} d(x_a, x'), \min_{x_b \in B, x' \in C} d(x_a, x') \right) \\ &= \min \left(d(A, C), d(B, C) \right) \\ &\geq \min \left(d(A, B), d(A, B) \right) \\ &= d(A, B) \end{aligned}$$

- Proofs for group-average and complete-linkage are similar.

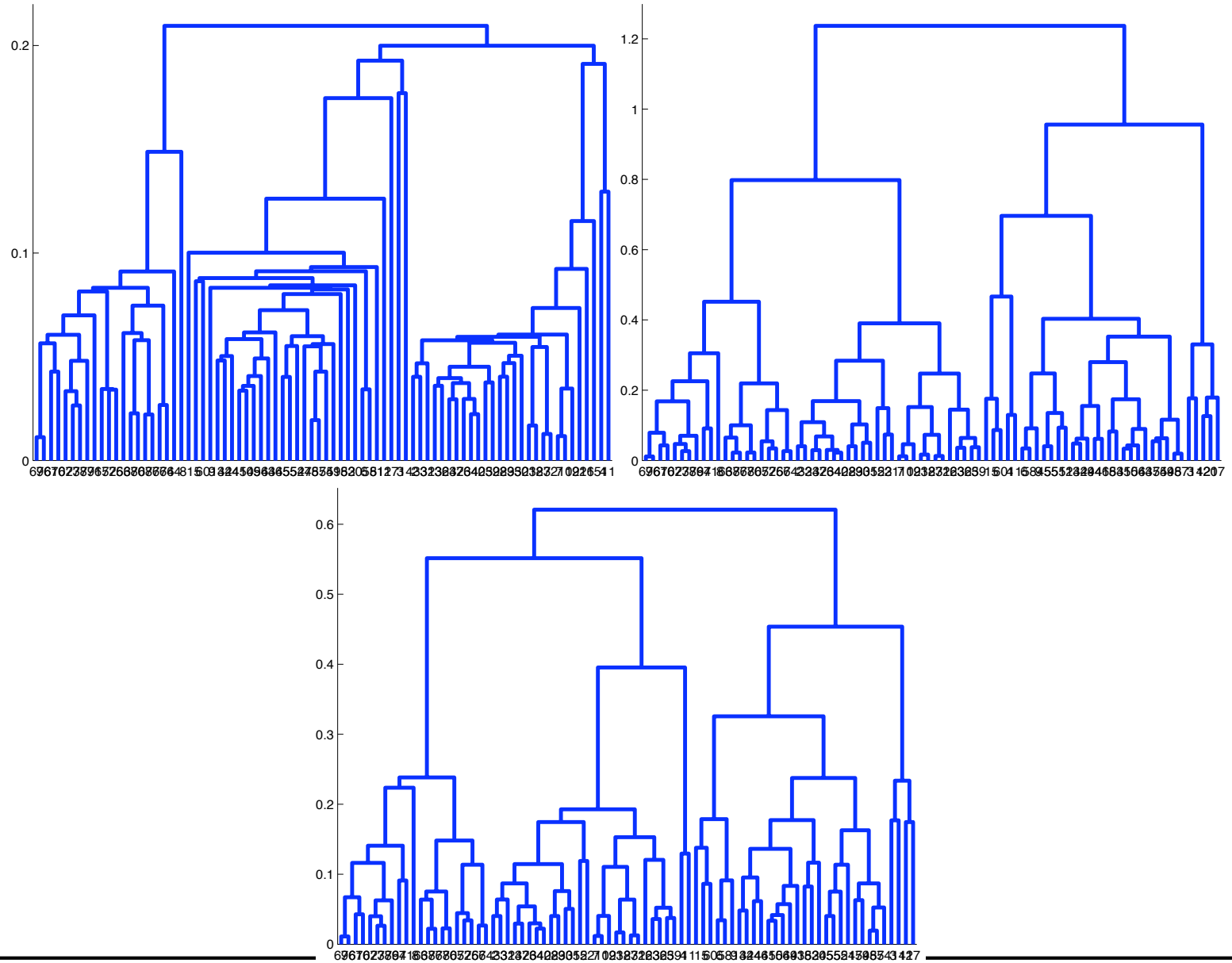
Dendrograms

- The monotonicity property implies that every time agglomerative clustering merges two clusters, the dissimilarity of those clusters is $\geq$ the dissimilarity of all previous merges.
- Why?

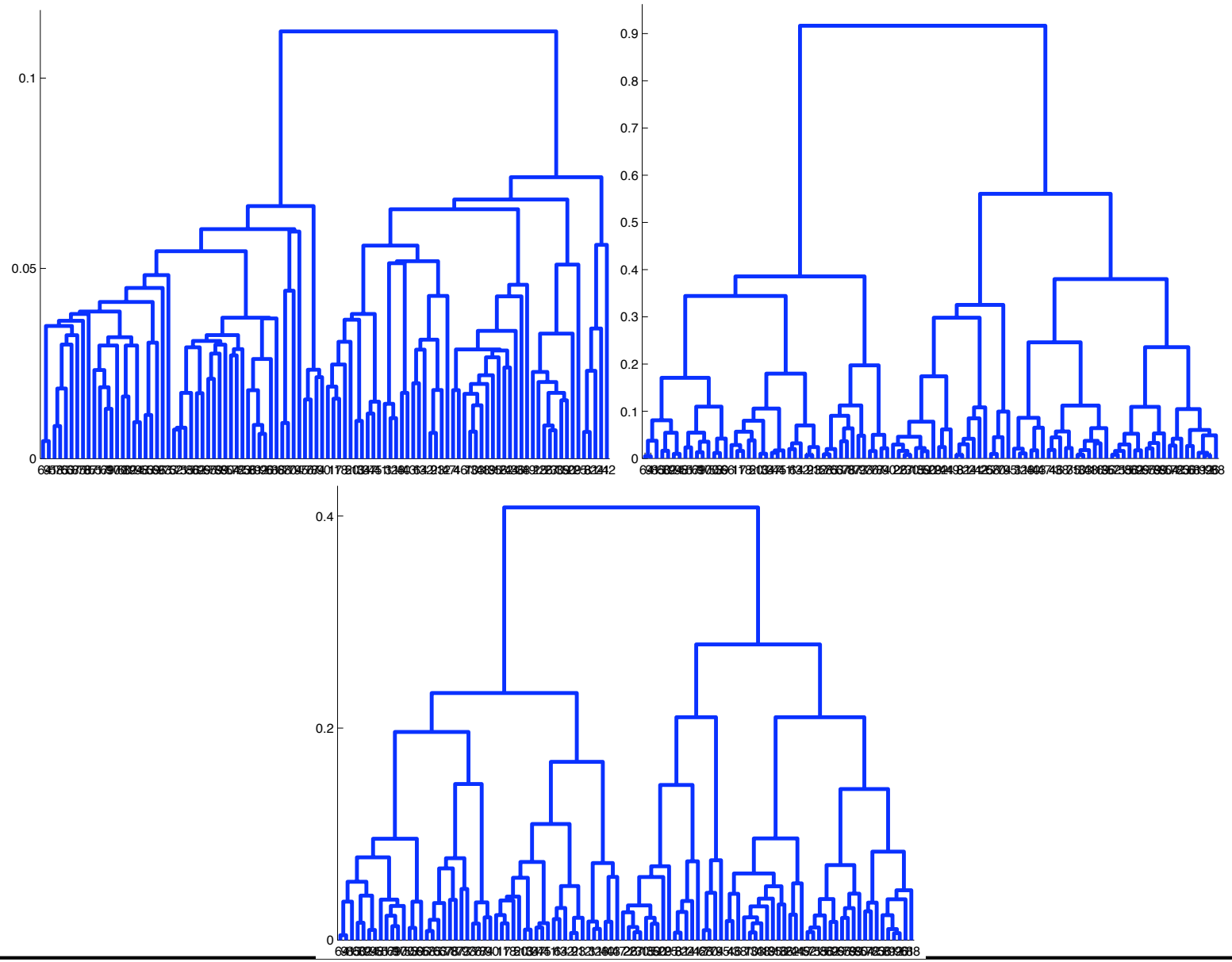
Dendrograms

- The monotonicity property implies that every time agglomerative clustering merges two clusters, the dissimilarity of those clusters is $\geq$ the dissimilarity of all previous merges.
- Why?
- Dendrograms (trees depicting hierarchical clusterings) are often drawn so that the height of a node corresponds to the dissimilarity of the merged clusters.

Dendrograms for Example 1 data



Dendrograms for Example 2 data



Remarks

- We can form a flat clustering by cutting the tree at any height.
- Jumps in the height of the dendrogram can suggest natural cutoffs.

How many clusters?

- How many clusters are generated by the agglomerative clustering algorithm?

How many clusters?

- How many clusters are generated by the agglomerative clustering algorithm?
- Answer: $2m - 1$, where m is the number of data objects.
- Why? A binary tree with m leaves has $m - 1$ internal nodes, thus $2m - 1$ nodes total.
- More explicitly:
  - The working list W starts with m singleton clusters
  - Each iteration removes two clusters from W and adds one new one
  - The algorithm stops when W has one cluster, which is after $m - 1$ iterations

Divisive clustering

- Works by recursively partitioning the instances.
- How might you do that?

Divisive clustering

- Works by recursively partitioning the instances.
- How might you do that?
  - K-means?
  - Max weighted cut on graph where edges are weighted by pairwise distances?
  - Maximum margin?
- Many heuristics for partitioning the instances have been proposed . . . but many are computationally hard and/or violate monotonicity, making it hard to draw dendrograms.

Hierarchical clustering summary

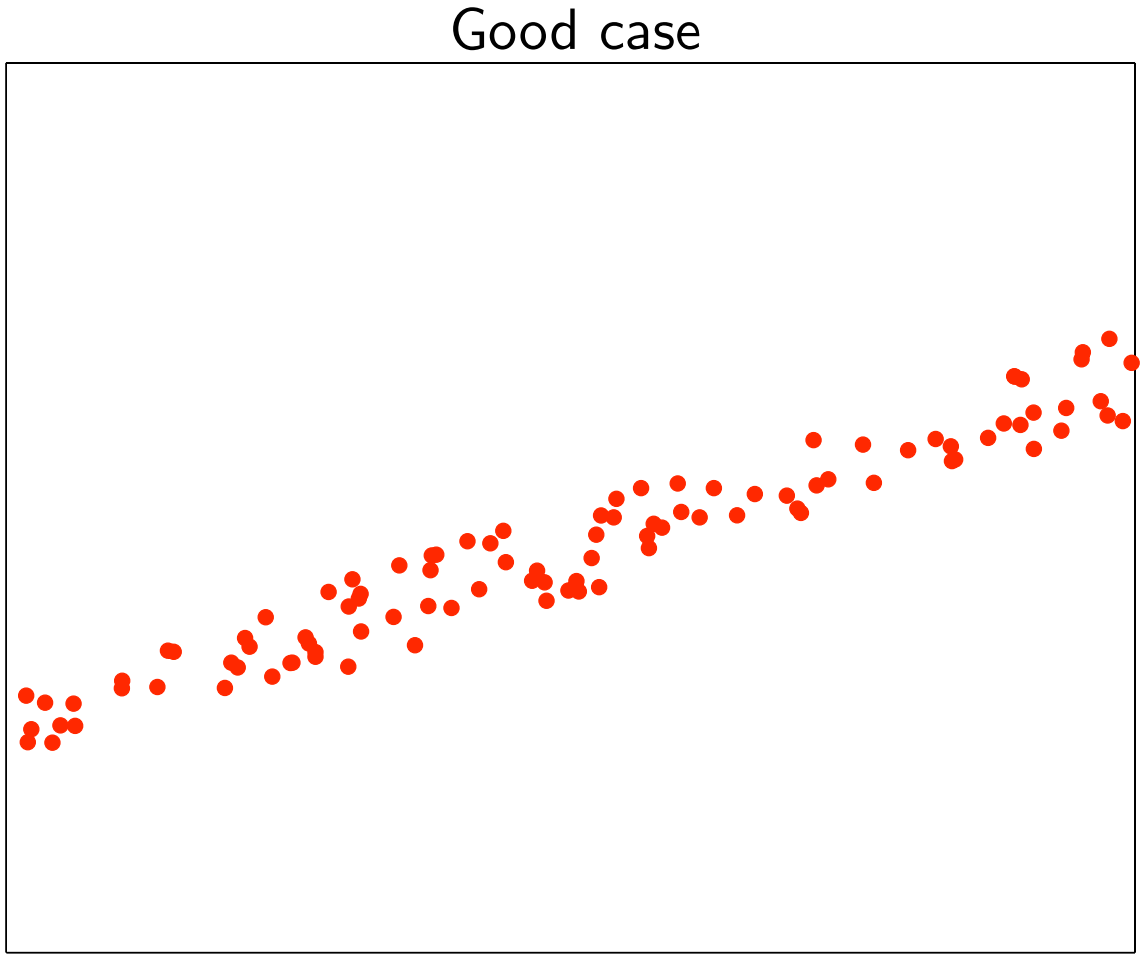
- Hierarchical clustering organizes data objects into a tree based on similarity.
- Agglomerative (bottom-up) tree construction is most popular.
- There are several choices of linkage criterion.
- Monotonicity allows us to draw dendrograms in which the height of a node corresponds to the dissimilarity of the clusters merged.
- Trees can be cut off at some level, to generate a flat partitioning of the data.

Dimensionality reduction

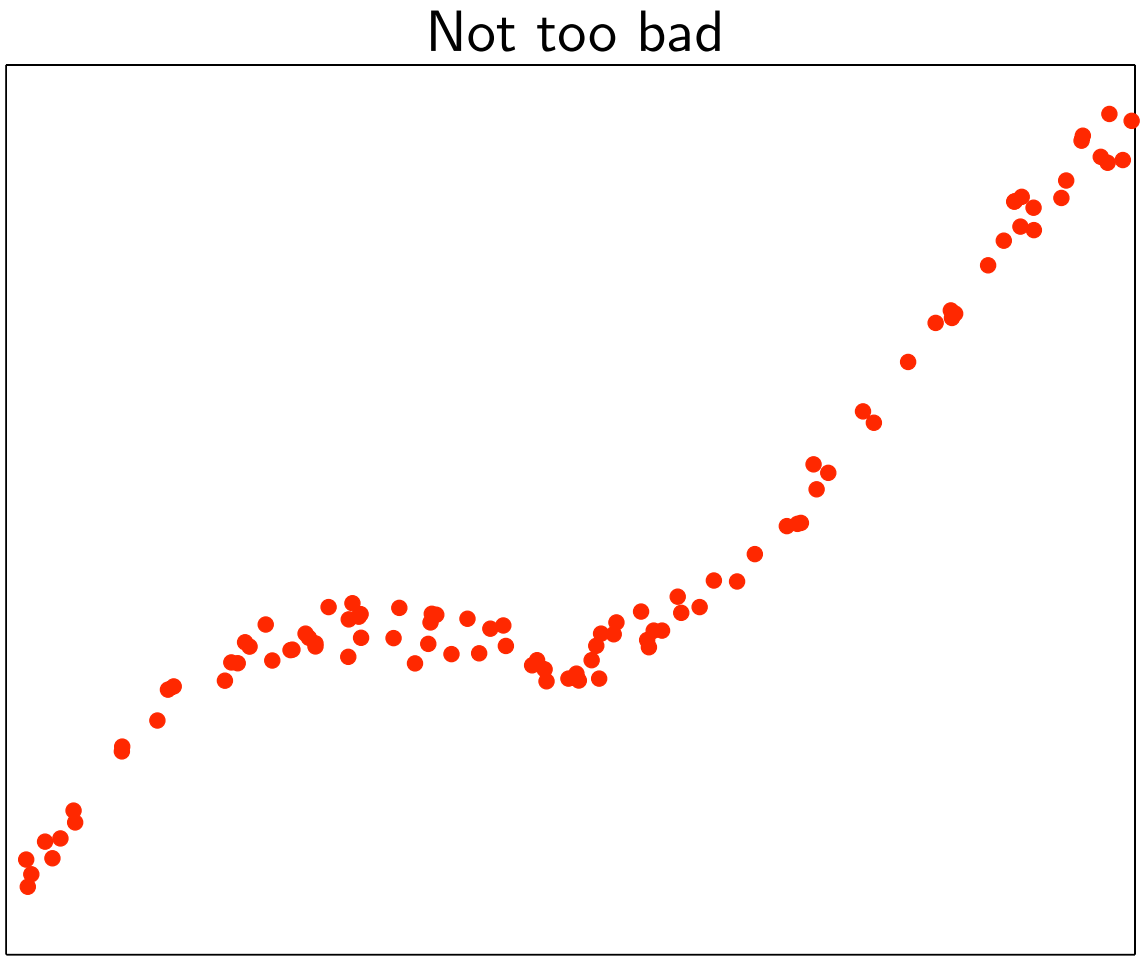
What is dimensionality reduction?

- Mapping data objects to (short) real vectors
- For visualization, comparison, outlier detection
- For further machine learning
- Some techniques:
  - Principal components analysis (linear)
  - Kernel PCA (nonlinear)
  - Independent components analysis (linear or nonlinear)
  - Self-organizing maps (nonlinear)
  - Multi-dimensional scaling (nonlinear, allows non-numeric data objects)

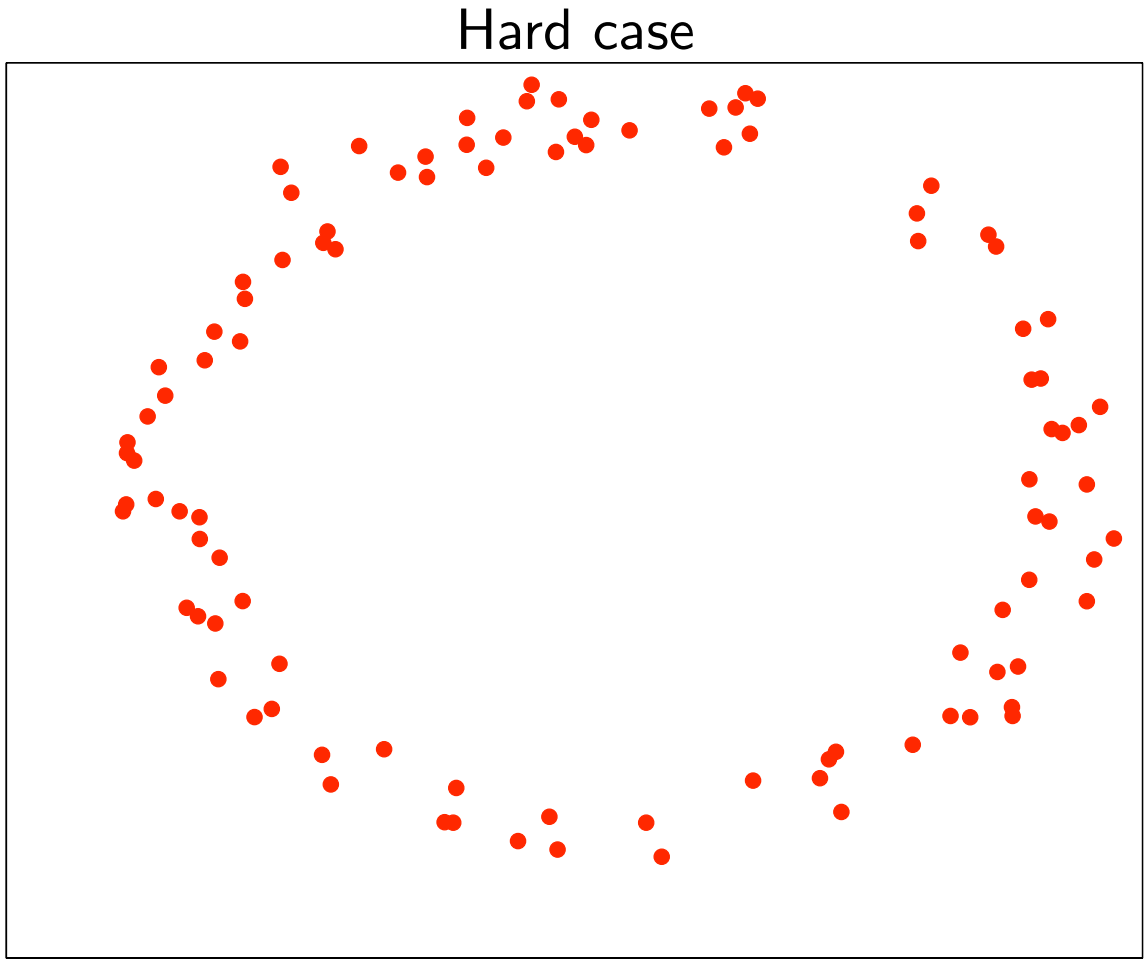
When is dimensionality reduction possible? (I)



When is dimensionality reduction possible? (II)

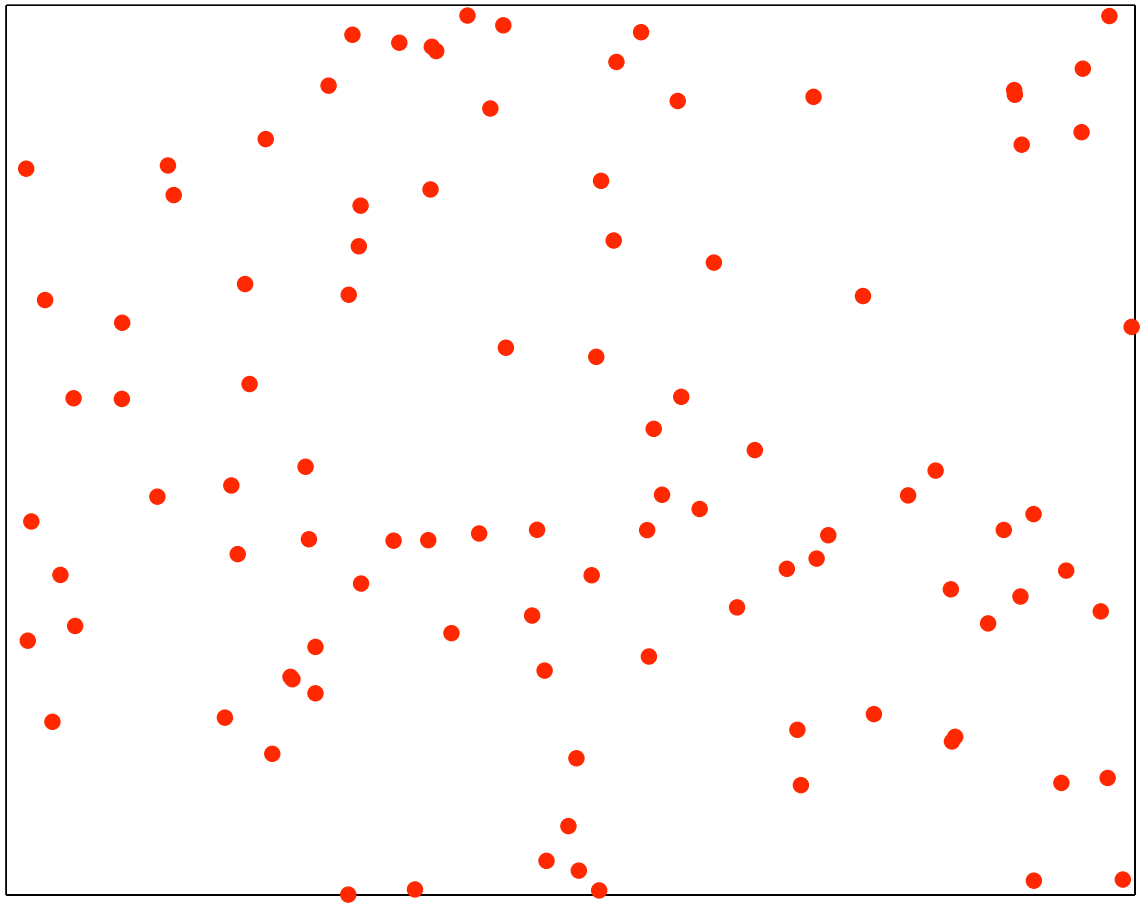


When is dimensionality reduction possible? (III)



When is dimensionality reduction possible? (IV)

Forget it!



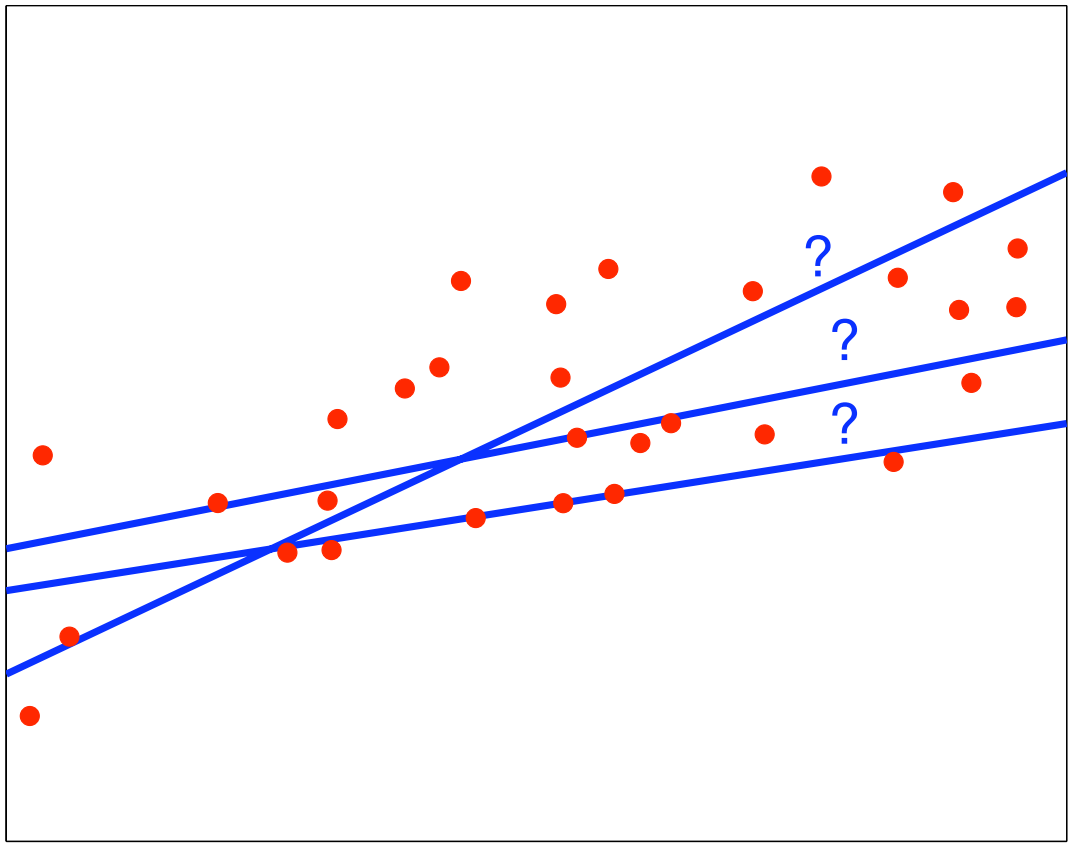
Remarks

- All dimensionality reduction techniques are based on an implicit assumption that the data lies along some *low-dimensional manifold*
- This is the case for the first three examples, which pretty much lie along a 1-dimensional manifold despite being plotted in 2D
- In the last example, the data has been generated randomly in 2D, so no dimensionality reduction is possible without losing information
- The first three cases are in increasing order of difficulty, from the point of view of existing techniques.

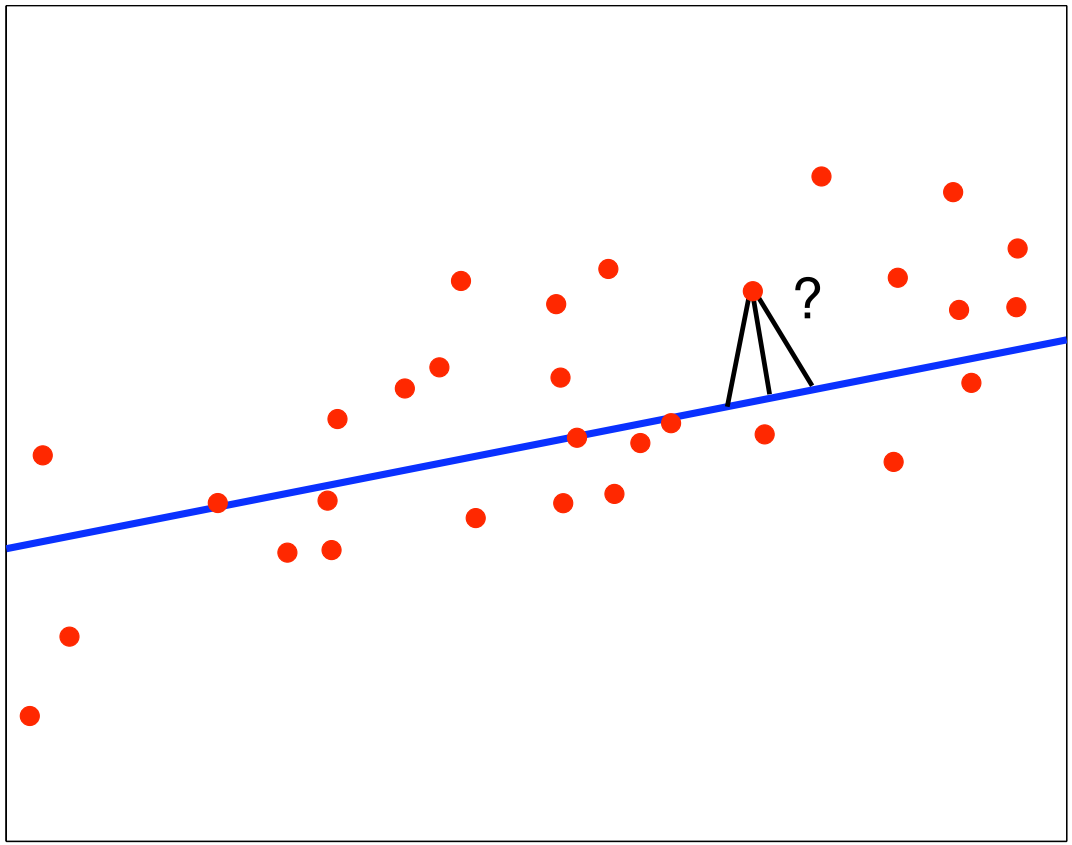
Simple Principal Component Analysis (PCA)

- Given: m data objects, each a length-n real vector.
- Suppose we want a 1-dimensional representation of that data, instead of n-dimensional.
- Specifically, we will:
  - Choose a line in $\Re^n$ that "best represents" the data.
  - Assign each data object to a point along that line.

Which line is best?



How do we assign points to lines?



Reconstruction error

- Let our line be represented as $\mathbf{b} + \alpha\mathbf{v}$ for $\mathbf{b}, \mathbf{v} \in \Re^n$, $\alpha \in \Re$.
  For later convenience, assume $\|\mathbf{v}\| = 1$.
- Each instance $\mathbf{x}_i$ is assigned a point on the line $\hat{\mathbf{x}}_i = \mathbf{b} + \alpha_i\mathbf{v}$.
- We want to choose $\mathbf{b}$, $\mathbf{v}$, and the α_i to minimize the total reconstruction error over all data points, measured using Euclidean distance:

$$\begin{aligned} R &= \sum_{i=1}^{m} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2 \\ &= \sum_{i=1}^{m} \|\mathbf{x}_i - (\mathbf{b} + \alpha_i\mathbf{v})\|^2 \end{aligned}$$

A constrained optimization problem!

$$\begin{array}{rl} \text{min} & \sum_{i=1}^{m} \|\mathbf{x}_i - (\mathbf{b} + \alpha_i \mathbf{v})\|^2 \\ \text{w.r.t.} & \mathbf{b}, \mathbf{v}, \alpha_i, i = 1, \ldots m \\ \text{s.t.} & \|\mathbf{v}\|^2 = 1 \end{array}$$

We write down the Lagrangian:

$$\begin{aligned} L(\mathbf{b}, \mathbf{v}, \lambda, \alpha_1, \ldots \alpha_m) &= \sum_{i=1}^{m} \|\mathbf{x}_i - (\mathbf{b} + \alpha_i \mathbf{v})\|^2 + \lambda(\|\mathbf{v}\|^2 - 1) \\ &= \sum_{i=1}^{m} \|\mathbf{x}_i\|^2 + m\|\mathbf{b}\|^2 + \|\mathbf{v}\|^2 \sum_{i=1}^{m} \alpha_i^2 \\ &- 2\mathbf{b}^T \sum_{i=1}^{m} \mathbf{x}_i - 2\mathbf{v}^T \sum_{i=1}^{m} \alpha_i \mathbf{x}_i + 2\mathbf{b}^T \mathbf{v} \sum_{i=1}^{m} \alpha_i \\ &- \lambda\|\mathbf{v}\|^2 + \lambda \end{aligned}$$

Solving the optimization problem

- The most straightforward approach would be to write the KKT conditions and solve the resulting equations
- Unfortunately, we get equations which have multiple variables in them, and the resulting system is not linear (you can check this)
- Instead, we will fix $\mathbf{v}$.
- For a given $\mathbf{v}$, finding the best $\mathbf{b}$ and α_i is now an unconstrained optimization problem:

$$\min R = \min \sum_{i=1}^{m} \|\mathbf{x}_i - (\mathbf{b} + \alpha_i \mathbf{v})\|^2$$

Solving the optimization problem (II)

- We write the gradient of R wrt to α_i and set it to 0:

$$\frac{\partial R}{\partial \alpha_i} = 2\|\mathbf{v}\|^2 \alpha_i - 2\mathbf{v}^T \mathbf{x}_i + 2\mathbf{b}^T \mathbf{v} = 0 \Rightarrow \alpha_i = \mathbf{v}^T(\mathbf{x}_i - \mathbf{b})$$

  where we take into account that $\|\mathbf{v}\|^2 = 1$.

- We write the gradient of R wrt $\mathbf{b}$ and set it to 0:

$$\nabla_{\mathbf{b}} R = 2m\mathbf{b} - 2\sum_{i=1}^{m} \mathbf{x}_i + 2\left(\sum_{i=1}^{m} \alpha_i\right)\mathbf{v} = 0 \tag{1}$$

- From above:

$$\sum_{i=1}^{m} \alpha_i = \sum_{i=1}^{m} \mathbf{v}^T(\mathbf{x}_i - \mathbf{b}) = \mathbf{v}^T\left(\sum_{i=1}^{m} \mathbf{x}_i - m\mathbf{b}\right) \tag{2}$$

Solving the optimization problem (III)

- By plugging (2) into (1) we get:

$$\mathbf{v}^T \left(\sum_{i=1}^{m} \mathbf{x}_i - m\mathbf{b} \right) \mathbf{v} = \left(\sum_{i=1}^{m} \mathbf{x}_i - m\mathbf{b} \right)$$

- This is satisfied when:

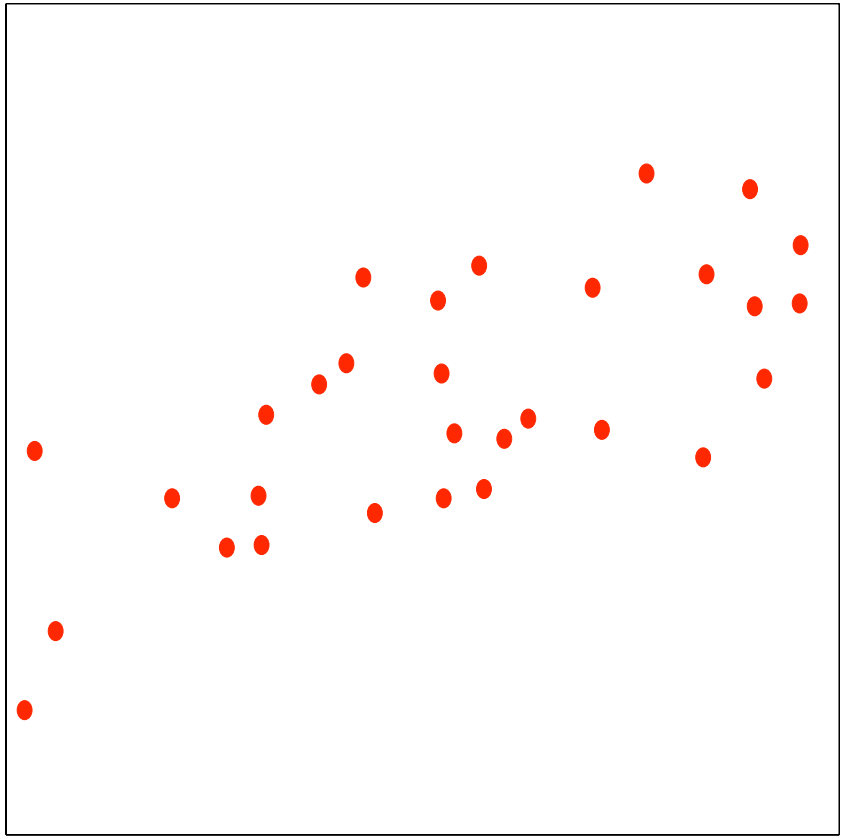
$$\sum_{i=1}^{m} \mathbf{x}_i - m\mathbf{b} = 0 \Rightarrow \mathbf{b} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{x}_i$$

- This means that the line goes through the mean of the data
- By substituting α_i, we get:

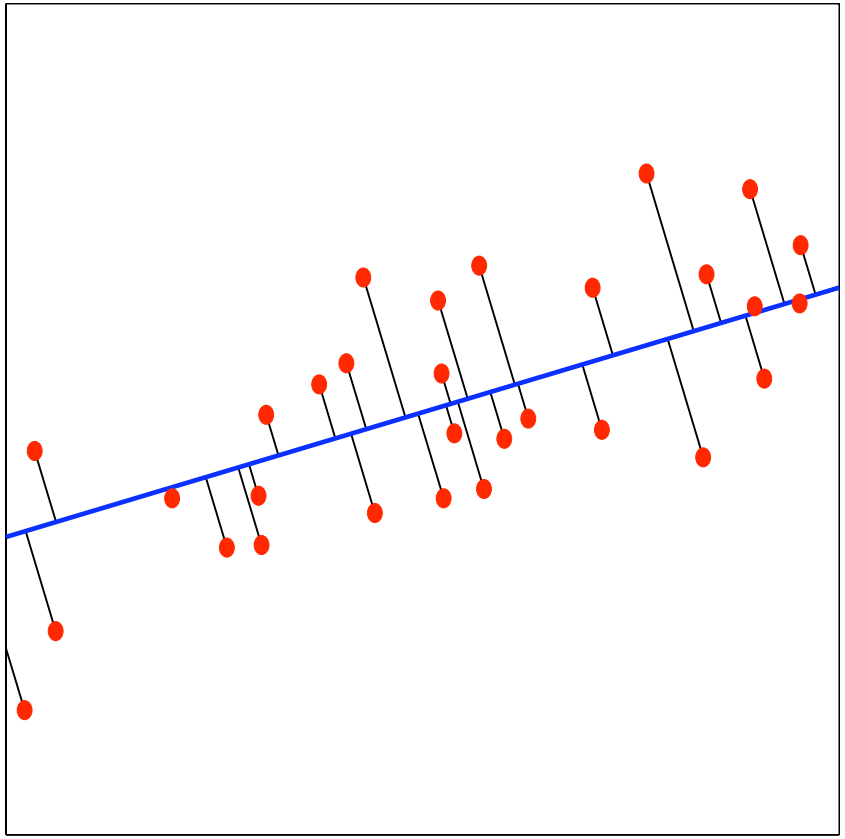
$$\hat{\mathbf{x}}_i = \mathbf{b} + (\mathbf{v}^T(\mathbf{x}_i - \mathbf{b}))\mathbf{v}$$

- This means that instances are projected orthogonally on the line to get the associated point.

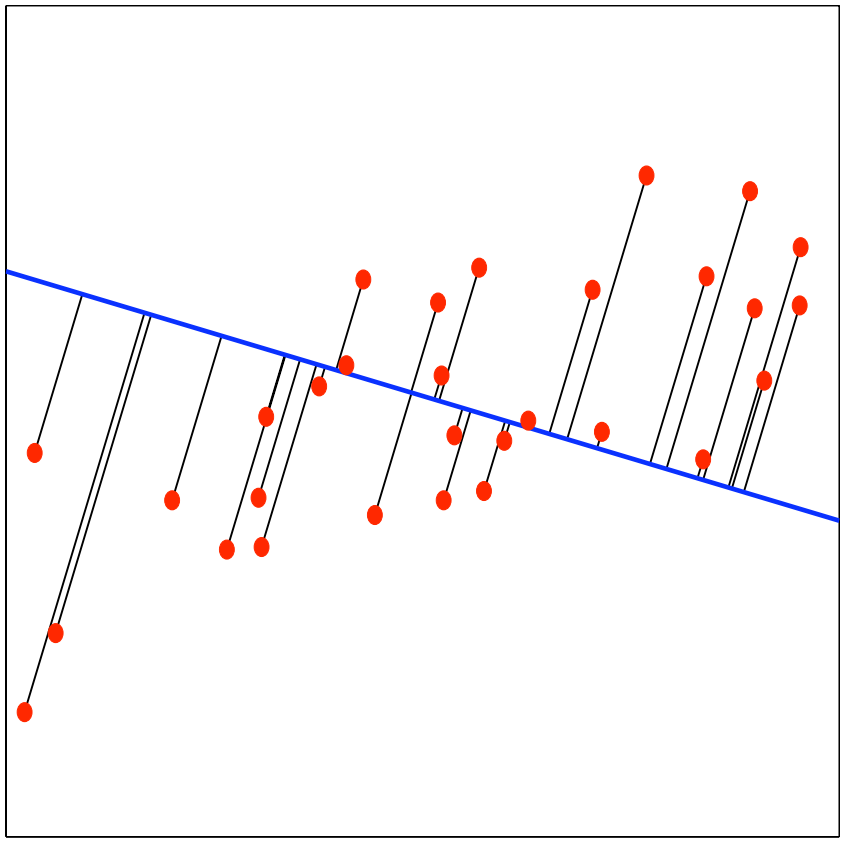
Example data



Example with $\mathbf{v} \propto (1, 0.3)$



Example with $\mathbf{v} \propto (1, -0.3)$



Finding the direction of the line

- Recall the formulation:

$$\begin{array}{rl} \min & \sum_{i=1}^{m} \|\mathbf{x}_i - (\mathbf{b} + \alpha_i \mathbf{v})\|^2 \\ \text{w.r.t.} & \mathbf{b}, \mathbf{v}, \alpha_i, i = 1, \ldots m \\ \text{s.t.} & \|\mathbf{v}\|^2 = 1 \end{array}$$

- Substituting $\alpha_i = \mathbf{v}^T(\mathbf{x}_i - \mathbf{b}) = (\mathbf{x}_i - \mathbf{b})^T\mathbf{v}$ into our optimization problem we obtain a new optimization problem:

$$\begin{array}{rl} \min_{\mathbf{v}} & \sum_{i=1}^{m} \|\mathbf{x}_i - \mathbf{b} - (\mathbf{v}^T(\mathbf{x}_i - \mathbf{b}))\mathbf{v}\|^2 \\ \text{s.t.} & \|\mathbf{v}\|^2 = 1 \end{array}$$

- Expanding the norm squared, we can simplify further ...

Finding the direction of the line

- Optimization problem re-written:

$$\begin{array}{rl} \max_{\mathbf{v}} & \sum_{i=1}^{m} \mathbf{v}^T(\mathbf{x}_i - \mathbf{b})(\mathbf{x}_i - \mathbf{b})^T\mathbf{v} \\ \text{s.t.} & \|\mathbf{v}\|^2 = 1 \end{array}$$

- The Lagrangian is:

$$L(\mathbf{v}, \lambda) = \sum_{i=1}^{m} \mathbf{v}^T(\mathbf{x}_i - \mathbf{b})(\mathbf{x}_i - \mathbf{b})^T\mathbf{v} + \lambda - \lambda\|\mathbf{v}\|^2$$

- Let $S = \sum_{i=1}^{m}(\mathbf{x}_i - \mathbf{b})(\mathbf{x}_i - \mathbf{b})^T$ be an n-by-n matrix, which we will call the **scatter matrix**
- The solution to the problem, obtained by setting $\nabla_{\mathbf{v}} L = 0$, is: $S\mathbf{v} = \lambda\mathbf{v}$.

Optimal choice of $\mathbf{v}$

- Recall: an **eigenvector** $\mathbf{u}$ of a matrix A satisfies $A\mathbf{u} = \lambda\mathbf{u}$, where $\lambda \in \Re$ is the **eigenvalue**.
- Fact: the scatter matrix, S, has n non-negative eigenvalues (except in certain degenerate cases) and n orthogonal eigenvectors.
- The equation obtained for $\mathbf{v}$ tells us that it should be an eigenvector of S.

- The $\mathbf{v}$ that maximizes $\mathbf{v}^T S\mathbf{v}$ is the eigenvector of S with the largest eigenvalue

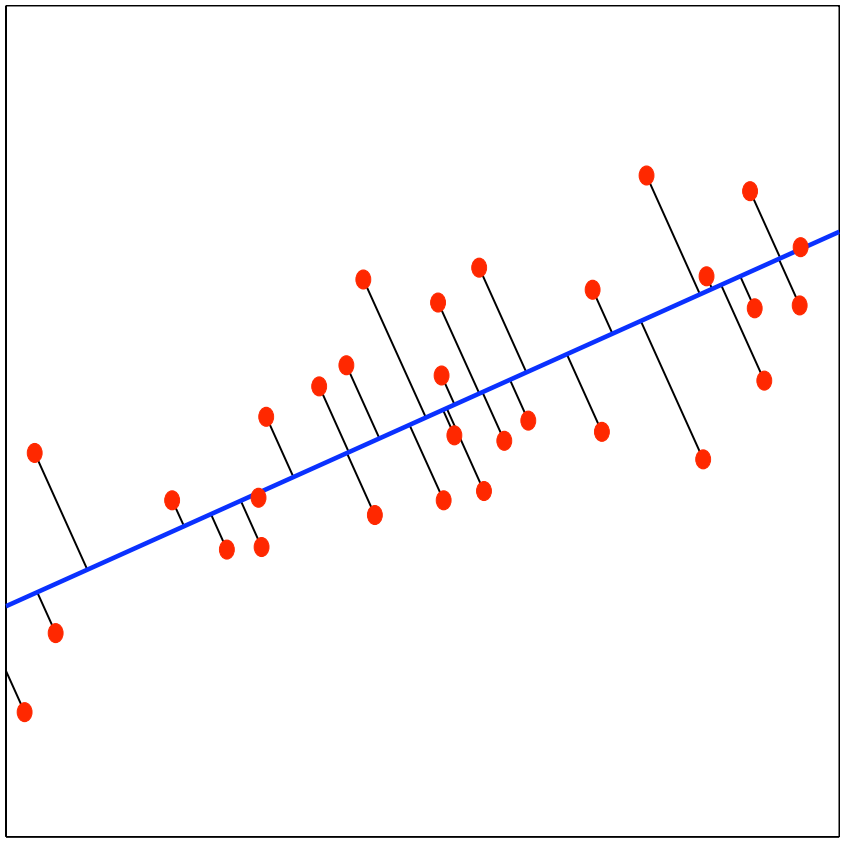
What is the scatter matrix?

- S is an $n \times n$ matrix with

$$S(k,l) = \sum_{i=1}^{m} (\mathbf{x}_i(k) - \mathbf{b}(k))(\mathbf{x}_i(l) - \mathbf{b}(l))$$

- Hence, $S(k,l)$ is proportional to the *estimated covariance* between the kth and lth dimension in the data.

Example with optimal line: $\mathbf{b} = (0.54, 0.52)$, $\mathbf{v} \propto (1, 0.45)$



Remarks

- The line $\mathbf{b} + \alpha\mathbf{v}$ is the **first principal component**.
- The variance of the data along the line $\mathbf{b} + \alpha\mathbf{v}$ is as large as along any other line.
- $\mathbf{b}$, $\mathbf{v}$, and the α_i can be computed easily in polynomial time.

Reduction to d dimensions

- More generally, we can create a d-dimensional representation of our data by projecting the instances onto a hyperplane $\mathbf{b} + \alpha^1 \mathbf{v}_1 + \ldots + \alpha^d \mathbf{v}_d$.
- If we assume the $\mathbf{v}_j$ are of unit length and orthogonal, then the optimal choices are:
  - $\mathbf{b}$ is the mean of the data (as before)
  - The $\mathbf{v}_j$ are orthogonal eigenvectors of S corresponding to its d largest eigenvalues.
  - Each instance is projected orthogonally on the hyperplane.

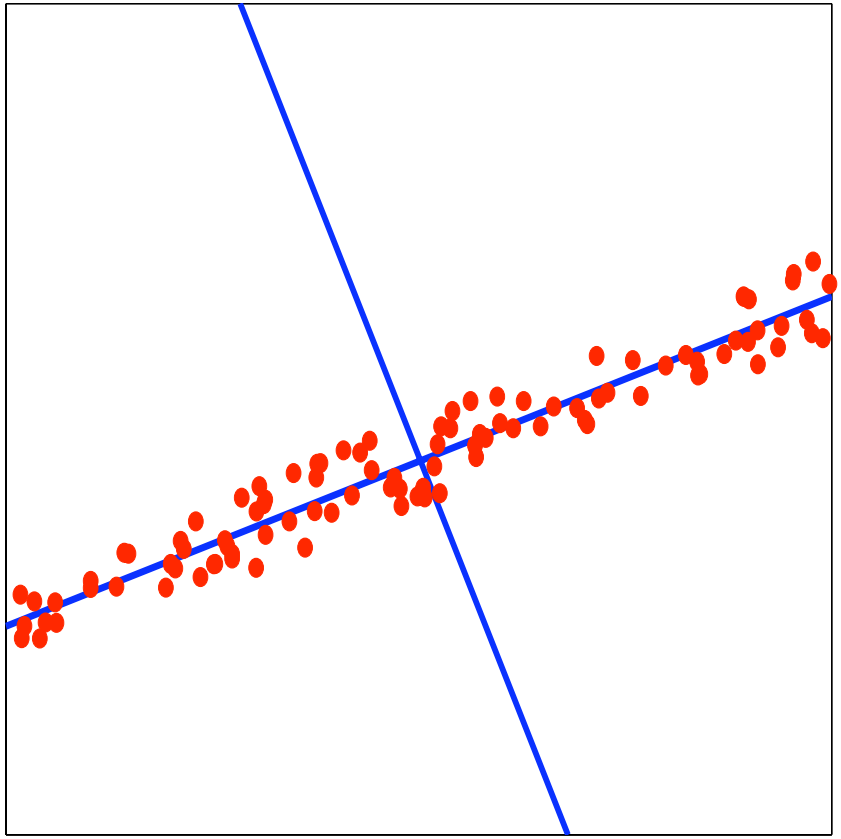
Remarks

- $\mathbf{b}$, the eigenvalues, the $\mathbf{v}_j$, and the projections of the instances can all be computed in polynomial time.
- The magnitude of the j^{th}-largest eigenvalue, λ_j, tells you how much variability in the data is captured by the j^{th} principal component
- So you have feedback on how to choose d!
- When the eigenvalues are sorted in decreasing order, the proportion of the variance captured by the first d components is:

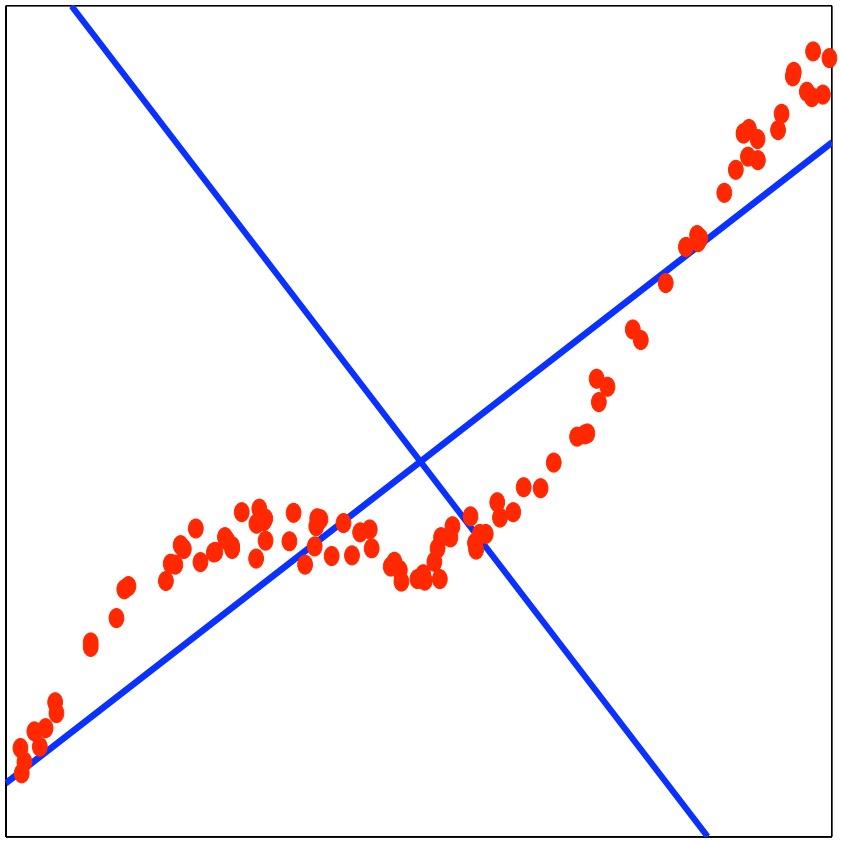
$$\frac{\lambda_1 + \ldots + \lambda_d}{\lambda_1 + \ldots + \lambda_d + \lambda_{d+1} + \ldots + \lambda_n}$$

- So if a "big" drop occurs in the eigenvalues at some point, that suggests a good dimension cutoff

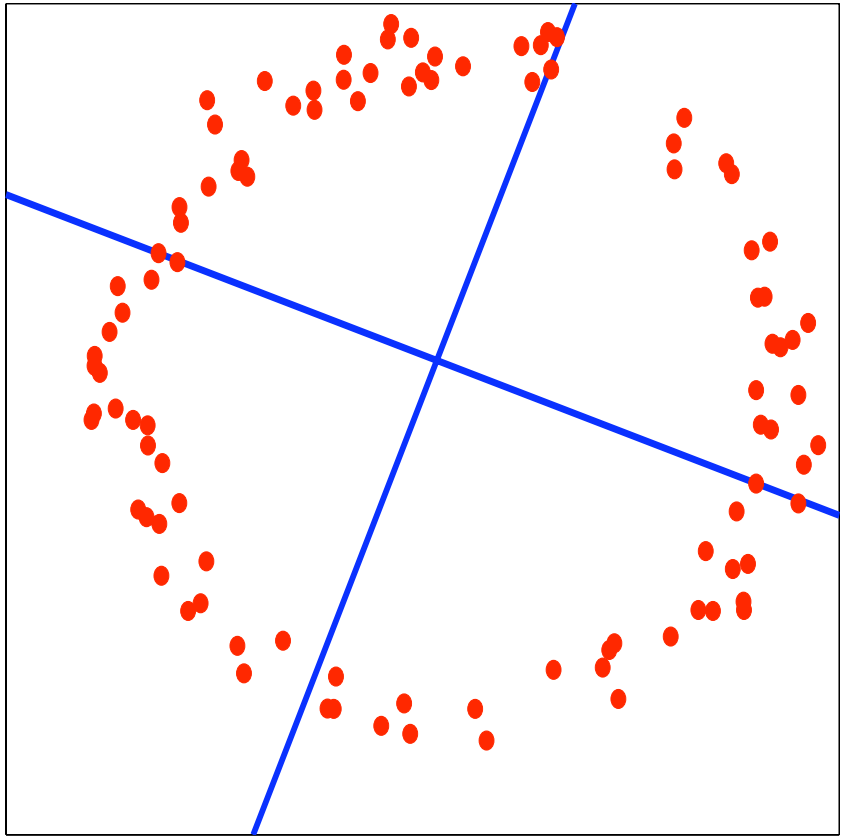
Example: $\lambda_1 = 0.0938, \lambda_2 = 0.0007$



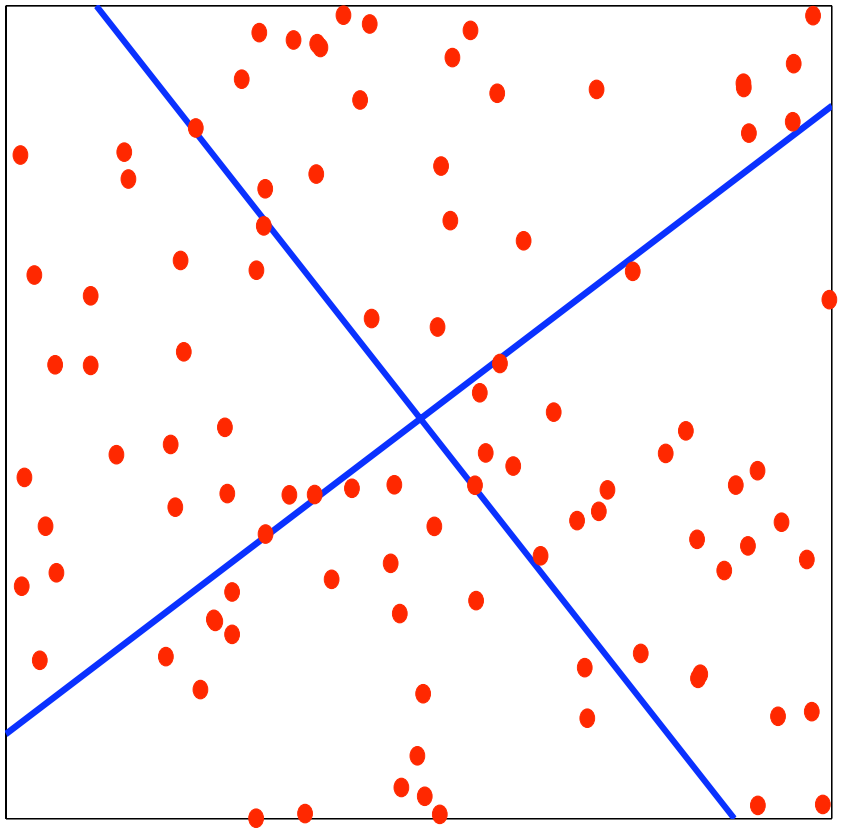
Example: $\lambda_1 = 0.1260, \lambda_2 = 0.0054$



Example: $\lambda_1 = 0.0884, \lambda_2 = 0.0725$



Example: $\lambda_1 = 0.0881, \lambda_2 = 0.0769$

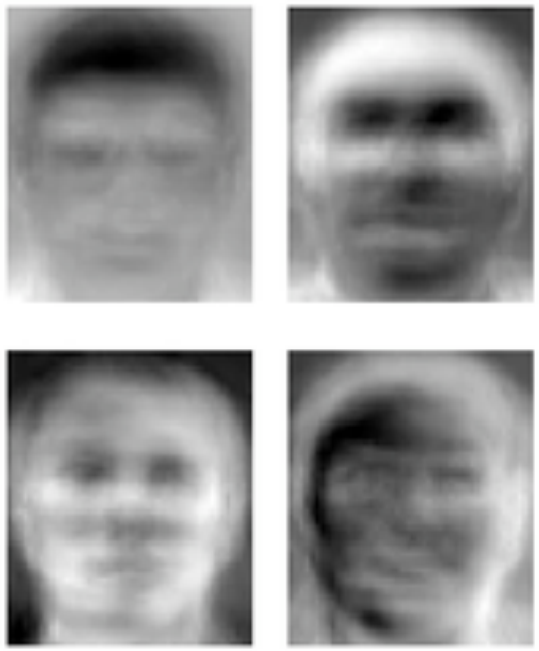


Remarks

- Outliers have a big effect on the covariance matrix, so they can affect the eignevectors quite a bit
- A simple examination of the pairwise distances between instances can help discard points that are very far away (for the purpose of PCA)
- If the variances in the original dimensions vary considerably, they can "muddle" the true correlations. There are two solutions:
  - work with the correlation of the original data, instead of covariance matrix
  - normalize the input dimensions individually before PCA

Remarks (II)

- In certain cases, the eigenvectors are meaningful; e.g. in vision, they can be displayed as images ("eigenfaces")



Uses of PCA

- Pre-processing for a supervised learning algorithm, e.g. for image data, robotic sensor data
- Used with great success in image and speech processing
- Visualization
- Exploratory data analysis
- Removing the linear component of a signal (before fancier non-linear models are applied)