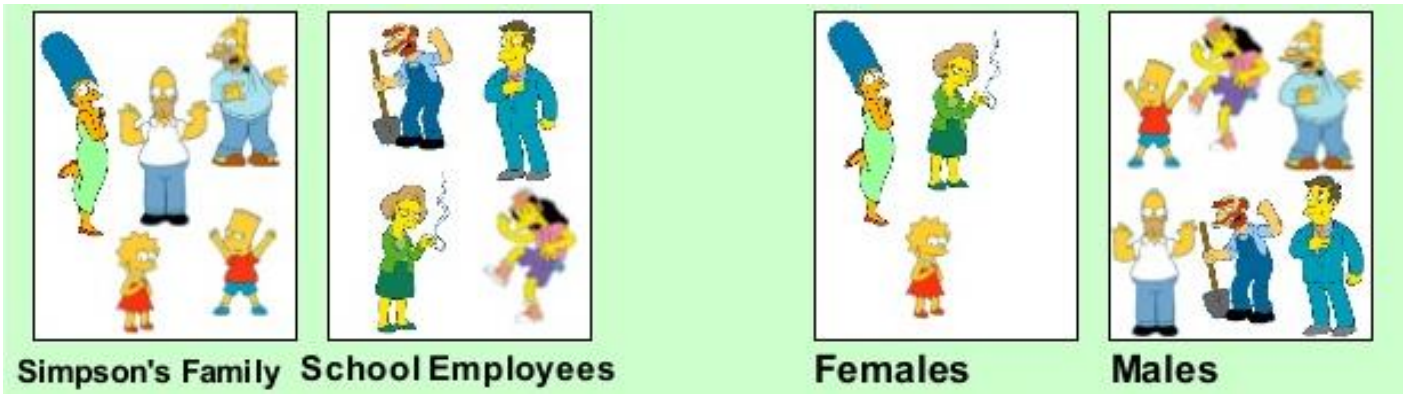
Document Clustering

Text and Web Mining (H6751)

School of Communication and Information

Clustering

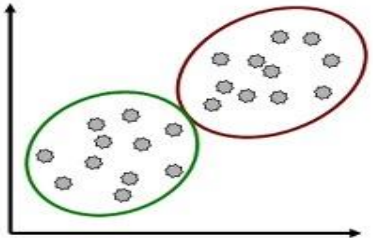


Finding Structure in a Document Collection

- **Prediction methods** (i.e. **document classification**) look at stored examples with **correct answers** and project answers for new examples.
- The assignment of **labels** is done **by humans**, and this can be a **tedious** and **expensive task**.
- Is there any way to assign labels automatically to a document collection? We will be looking at clustering.

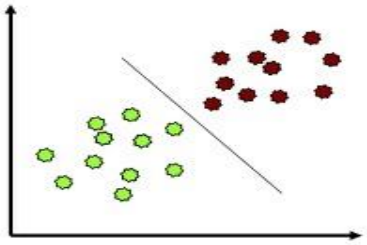
CLUSTERING

- Data is not labeled
- Group points that are "close" to each other
- Identify structure or patterns in data
- Unsupervised learning



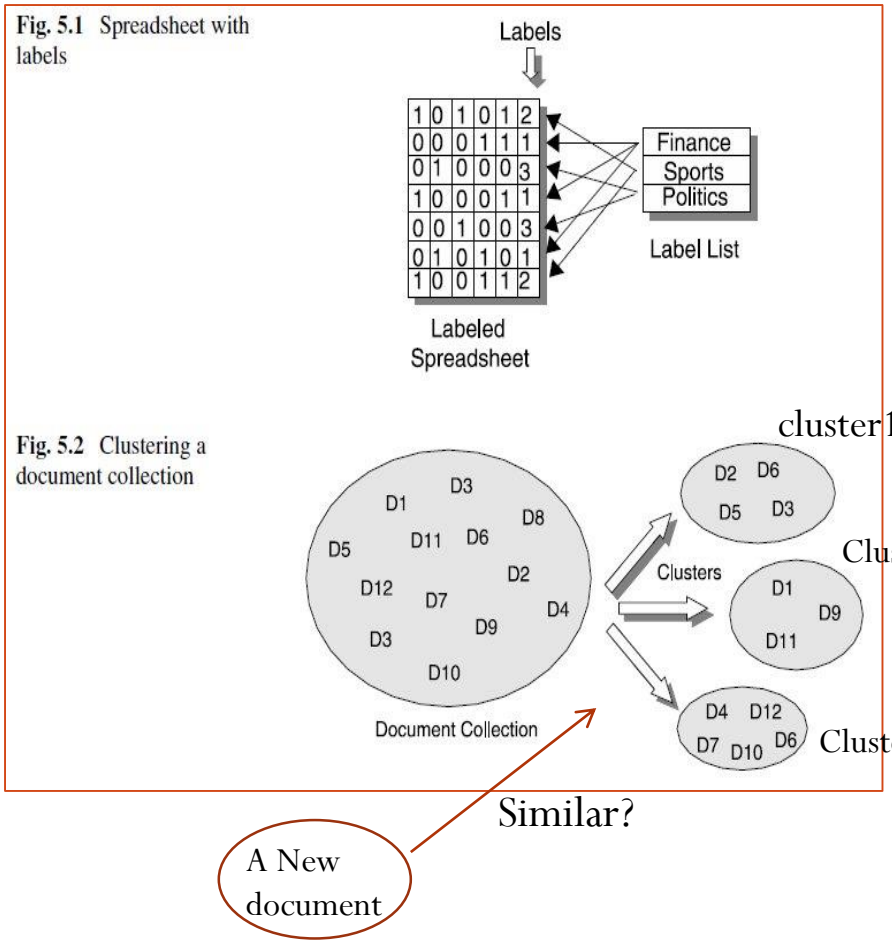
CLASSIFICATION

- Labeled data points
- Want a "rule" that assigns labels to new points
- Supervised learning



Finding Structure in a Document Collection

- Through Document Clustering, **similar documents** are grouped together (finding structure), and labels can be assigned to the clusters.
- Our expectations for **accurate predictive performance** should be **reduced** from standard prediction applications with labeled data.
  - **Unsupervised Learning**
  - Cluster labels can be used as a feature for a supervised learning.



k-Means Clustering

- **k-Means** is very widely used for document clustering and is relatively efficient.
- The documents start in one pile and are then distributed into smaller piles of similar documents.
- The name **k-Means** implies that **k clusters** will be used, and **the means of these clusters** will play an important role.

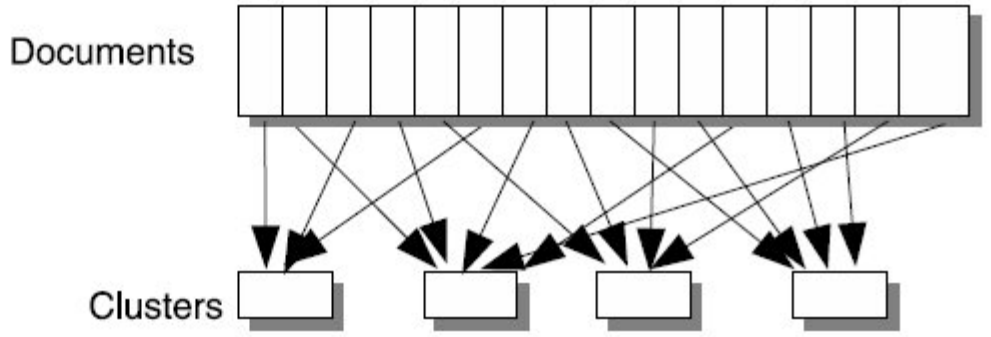


Fig. 5.6 Assigning documents to clusters

K-Means Clustering

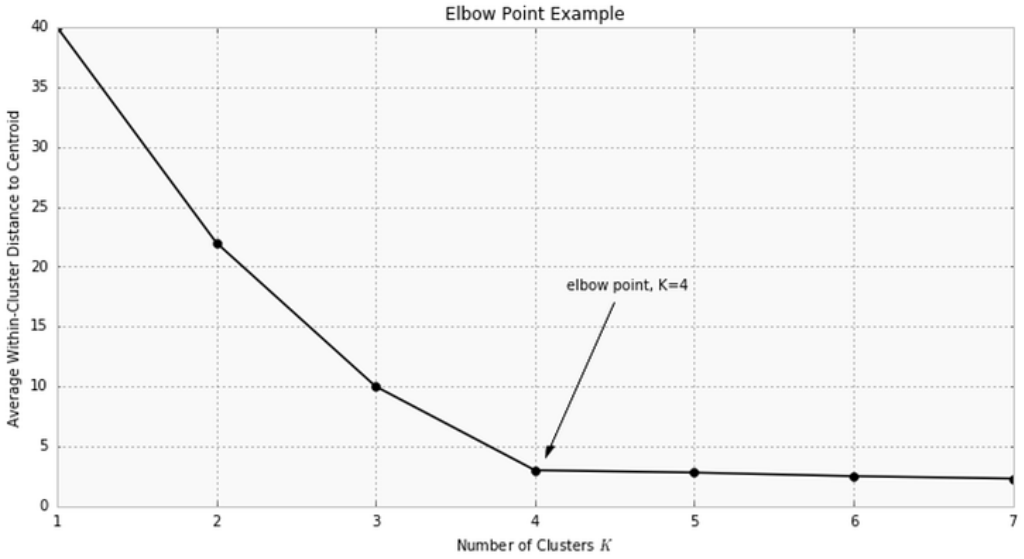
- Each cluster is associated with a centroid (center point) that is randomly selected.
- Each point assigned to closest centroid's cluster
- 'Closeness' measured by Euclidean distance, cosine similarity, correlation, etc.
- Specify K => number of clusters.
- Recompute centroids by taking the mean of all data points assigned to that centroid's cluster

Algorithm 1 Basic K-means Algorithm.

1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

K-Means

- One of the metrics that is commonly used to compare results across different values of *K* is the **mean distance** between data points and their cluster centroid.
- Since increasing the number of clusters will always reduce the distance to data points, increasing *K* will *always* decrease this metric to the extreme of reaching zero when *K* is the same as the number of data points.
- Plot the mean distance to the centroid as a function of *K* and the "elbow point," where the rate of decrease sharply shifts, can be used to roughly determine *K*.



k-Means Clustering: Centroid Classifier

- The following shows an example with **four** training documents partitioned into **two classes**, *C*1 and *C*2.
- The new document in this example is classified as *C*2.
- The closer the two documents are, the larger their inner product will be.
- The value of their inner product is a quantity in [0, 1], which can be used to measure the confidence of the classification result.

Class	Class Vectors	Normalized Centroids (m_{class})
*C*1	(1,2,0,0), (0,2,0,0)	(.24,.97,0,0)
*C*2	(1,0,1,1), (0,0,1,2)	(.27,0,.53,.80)

New document to be classified:	(1,0,1,0)
Normalized new document (x):	(.71,0,.71,0)
Similarity (inner product) between x and m_{C1}:	.17
Similarity (inner product) between x and m_{C2}:	.57
New Document classified as:	*C*2

(1+0)/2=0.5
(2+2)/2=2
(0.5,2,0,0)
Centroid = mean of vectors

0.5/sqrt(0.5^2+2^2)
= 0.24
2/sqrt(0.5^2+2^2)
=0.97
Normalization

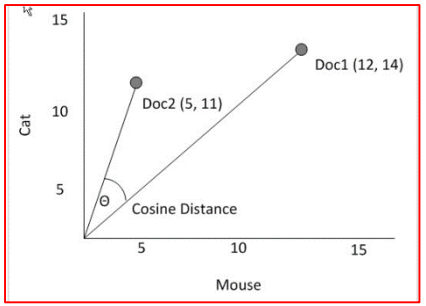
1/sqrt(2) Normalization

.71*.24 = 0.17 (inner product)

(.71*.27)+ 0 + (.71*.53) + 0 = 0.57 (inner product)

k-Means Clustering: Centroid Classifier

- How to generate **normalized centroids**?
- Firstly, convert vectors to **normalized vectors (unit vectors, their lengths are 1)**, and calculate a mean.



- Normalized Vector $= \frac{\vec{V}}{|\vec{V}|} = \frac{\vec{V}}{\sqrt{\sum_{i=1}^{n} \vec{V}_i^{\,2}}}$
- E.g., $(1, 0, 1, 0) = \frac{(1,0,1,0)}{\sqrt{1^2+0^2+1^2+0^2}} = \frac{(1,0,1,0)}{\sqrt{2}} = \frac{(1,0,1,0)}{1.41} = (0.71, 0, 0.71, 0)$

Class	Class Vectors	Normalized Centroids (m_{class})
$C1$	(1,2,0,0), (0,2,0,0)	(.24,.97,0,0)
$C2$	(1,0,1,1), (0,0,1,2)	(.27,0,.53,.80)

New document to be classified:	(1,0,1,0)
Normalized new document (x):	(.71,0,.71,0)
Similarity (inner product) between x and m_{C1}:	.17
Similarity (inner product) between x and m_{C2}:	.57
New Document classified as:	$C2$

Fig. 5.9 An example of centroid classification

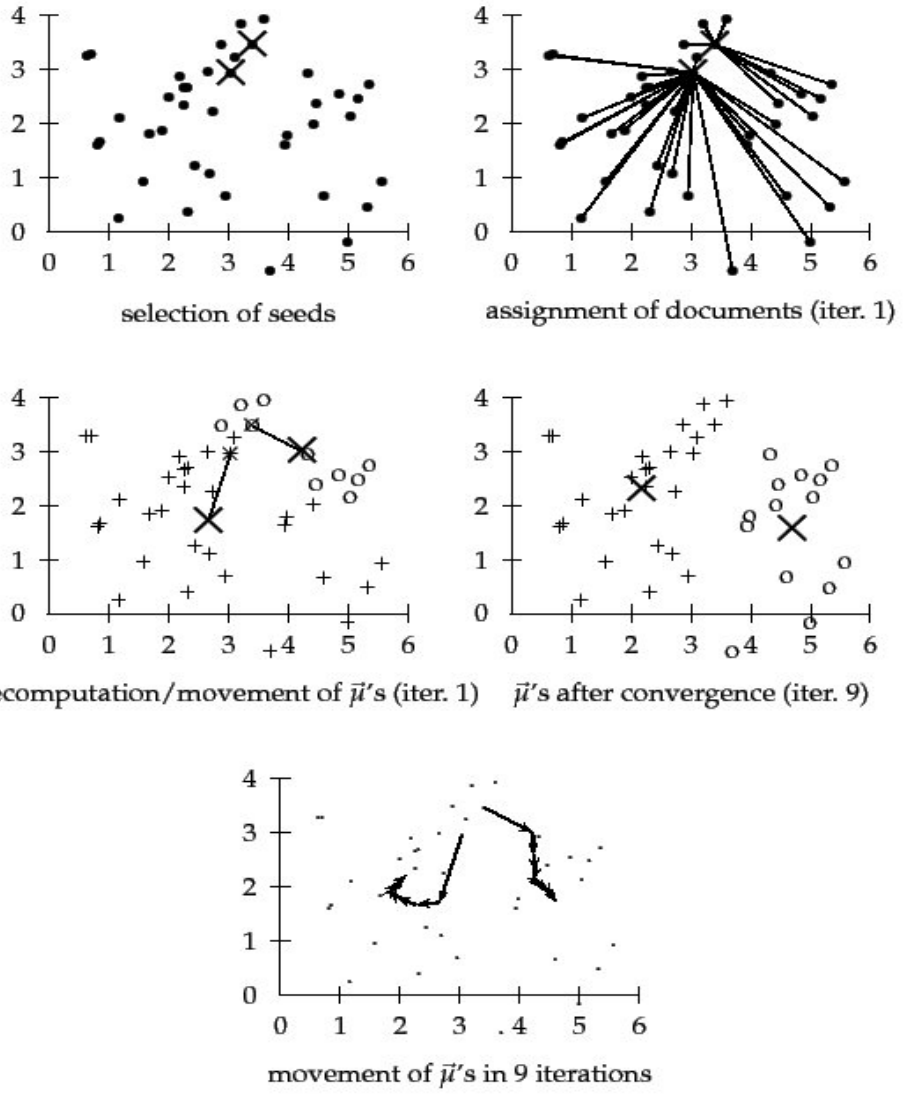
If the documents and the centroid vector are normalized, then the inner product corresponds to the **Cosine Similarity** measure.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

The cosine of 0° is 1, so becomes highest (i.e. 1) when A and B point to the same direction.

k-Means Clustering

- Another example with **k =2** (from Introduction to Information Retrieval, 2009)
- Selection of seeds: select any two as the two centroids.
- Movement of centroid mean.



► **Figure 16.3** A *K*-means example for $K = 2$ in $\mathbb{R}^2$. The position of the two centroids ($\vec{\mu}$'s shown as X's in the top four panels) converges after nine iterations.

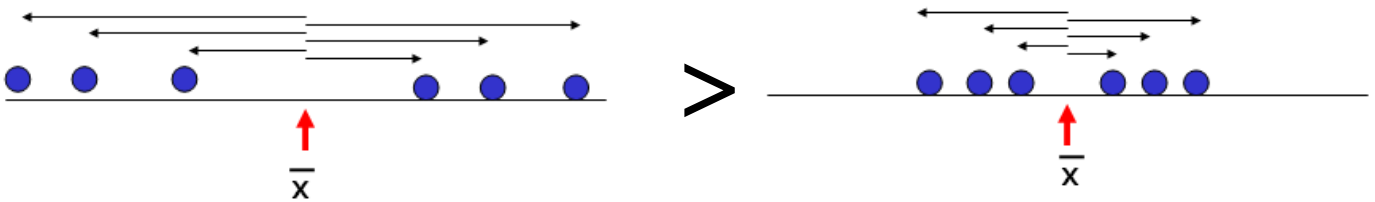
k-Means Clustering

- The main problem for **k-Means** clustering is to **determine *k***, the number of clusters.
- If we know something about the nature of the documents, we may also know something about a reasonable number of labels or categories.
- In general, the number of bins should be far smaller than the number of documents.
- Otherwise, prediction and generalization to new documents will be weakened.
- Is determining ***k*** strictly guesswork?

k-Means Clustering

- The **sample variance**, $\mathbf{s^2}$, is the arithmetic mean of the squared deviations from the sample mean:

$$s^2 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}$$



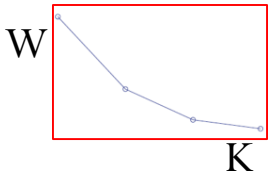
k-Means Clustering

- For ***k*-Means**, let x_i be the *i*-th document vector, and $c_i \in \{1, \ldots, k\}$ be its corresponding cluster index, a typical measure is **the total variance from the cluster means** as described below for ***k*** clusters.

$$E(k) = \sum_{i=1}^{n} \frac{(x_i - m_{c_i})^2}{n}$$

- For each document vector x_i and its cluster mean m_{c_i}, compute **the squared error averages** over all ***n*** documents, *E(k)*.
- Then vary ***k***, perhaps by starting with two clusters, then three, and continuing to rerun *k*-means, stopping when ***k*** is too large.
- Then compare the objective function values for varying ***k***.
- Choose ***k*** based on the point where increasing ***k*** does not commensurate (or reasonable) decrease in variance.
  - Called elbow method. But still not easy to decide k value.

k	W
2	19.998
3	19.814
4	19.462



k-Means Clustering: **Centroid Classifier**

- The mean of document vectors in a category is often referred to as **the centroid of the category**.
- It can be used for the purpose of **prediction**.
- Assume that we have $\boldsymbol{k}$ categories indexed by $c = 1, \ldots, k$, each with a centroid $\boldsymbol{m}_c$.
- When a future document $\boldsymbol{x}$ comes in, their inner product $\boldsymbol{x} \cdot \boldsymbol{m}_c$ can be computed, and we pick the category associated with **the largest inner product** as the label of the document.

Hierarchical Clustering

- **Hierarchical (agglomerative) clustering** produces clusters, but they are **organized in a hierarchy** much like a table of contents for a book.
- In **hierarchical clustering**, we do not know in advance how many clusters we want; in fact, we end up with **a tree-like** visual representation of the observations, called a **dendrogram**.
- The major weaknesses of hierarchical clustering are timing and computational complexity.
- The method is intended for clustering a relatively small number of documents.

Hierarchical Clustering

- Algorithm for **Hierarchical Agglomerative Clustering (HAC)**.

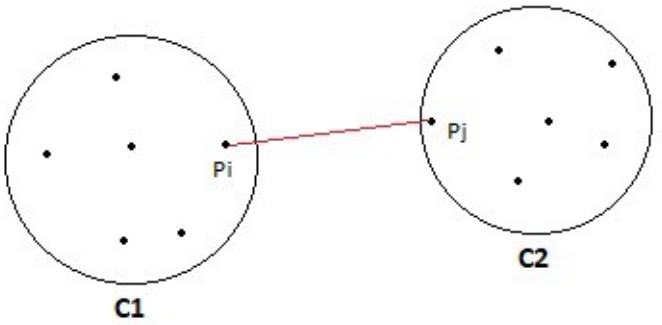
1. **Compute the proximity matrix**
2. **Let each data point be a cluster**
3. **Repeat**
4. **Merge the two closest clusters**
5. **Update the proximity matrix**
6. **Until only a single cluster remains**

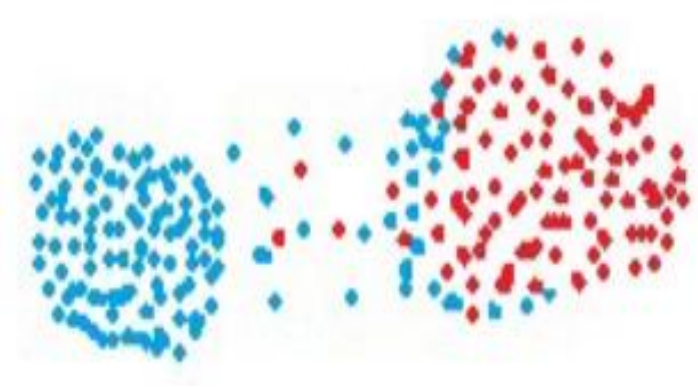
- The result will be a binary tree with links of parent clusters to their partitioned clusters.
- Pairs of clusters are recursively combined until only one cluster remains.
- Key operation is the computation of the proximity of two clusters
  - Different approaches to defining the **distance between clusters** distinguish the different algorithms.

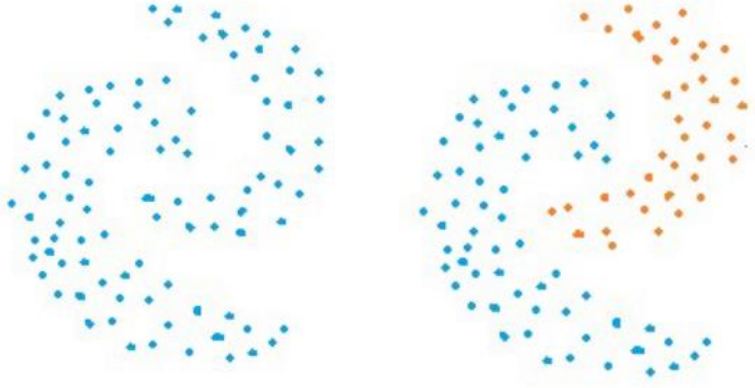
How to Define Inter-Cluster Similarity?

- (Single) - Minimal inter-cluster similarity. of cluster C1 and cluster C2 is the **minimum** of the similarities between points Pi and Pj such that Pi belongs to C1 and Pj belongs to C2.
  - Can separate non-elliptical shapes

  as long as the gap between two clusters is not small.
  - Sensitive to noise and outliers





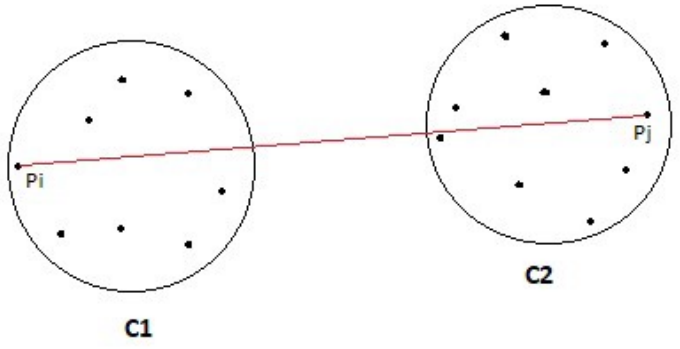


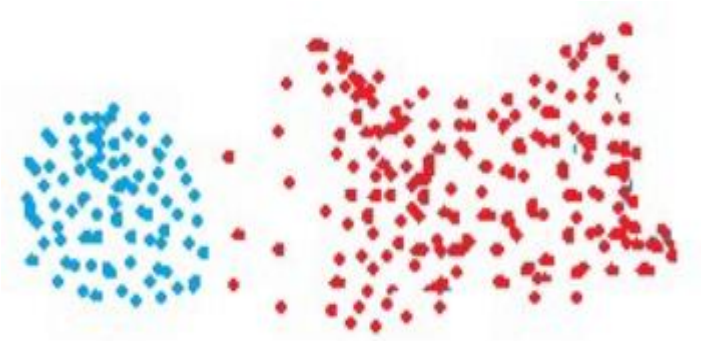
Original Clustered

non-elliptical shapes

How to Define Inter-Cluster Similarity?

- (Complete Linkage) – Maximal inter-cluster similarity. Similarity of clusters C1 and C2 is the **maximum** of the similarity between points Pi and Pj such that Pi belongs to C1 and Pj belongs to C2.
  - Does well in separating clusters if there is noise between clusters
  - Max approach is biased towards globular clusters.
  - Tends to break large clusters.





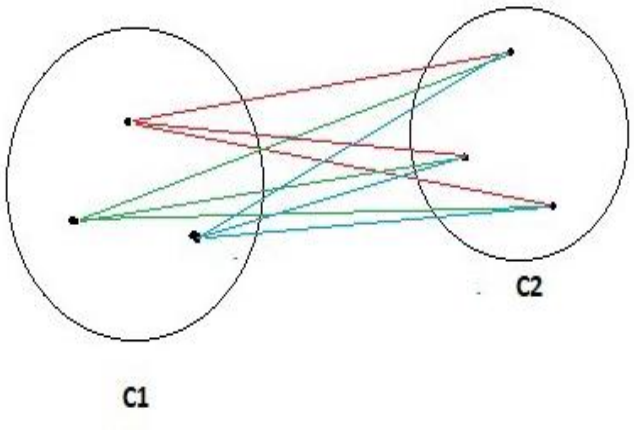
Does well in separating clusters if there is noise

How to Define Inter-Cluster Similarity?

- (Average) – Mean inter-cluster similarity. Compute all pairwise similarities between the observations in cluster C1 and the observations in cluster C2, and record the **average** of these similarities.

$$\text{sim}(C1,C2) = \sum_{i=1}^{n_{c1}} \sum_{j=1}^{n_{c2}} sim(Pi, Pj) / (|n_{C1}| * |n_{C2}|)$$

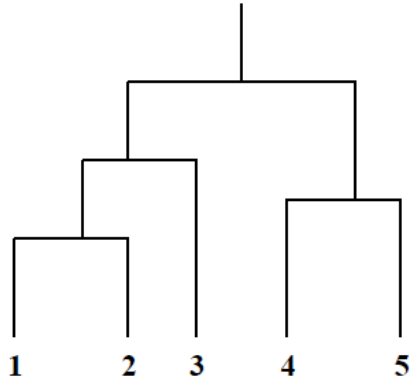
  - n_{C1} and n_{C2} are the number of observations in C1 and C2 respectively.
  - The group Average approach does well in separating clusters if there is noise between clusters.
  - The group Average approach is biased towards globular clusters.



Cluster Similarity

- **Single Link** – similarity of two clusters is based on the two **most** similar (closest) points in the different clusters - determined by one pair (strongest link) of points.

	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00

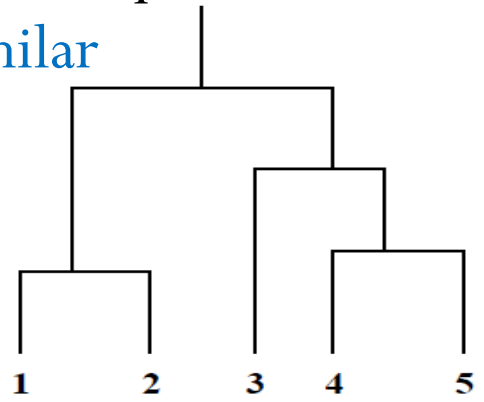


- **Complete Link** - similarity of two clusters is based on the two **least** similar (most distant) points – determined by all pairs of points in the two clusters => clustering is still based on most similar

Cluster (I1, I2) and I3 furthest point is I1 (0.1)

	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00

Cluster (I4,I5) and I3 similarity is furthest point I5 (**0.3**) => more distant than (0.1)



Hierarchical Clustering

- Types of Linkage

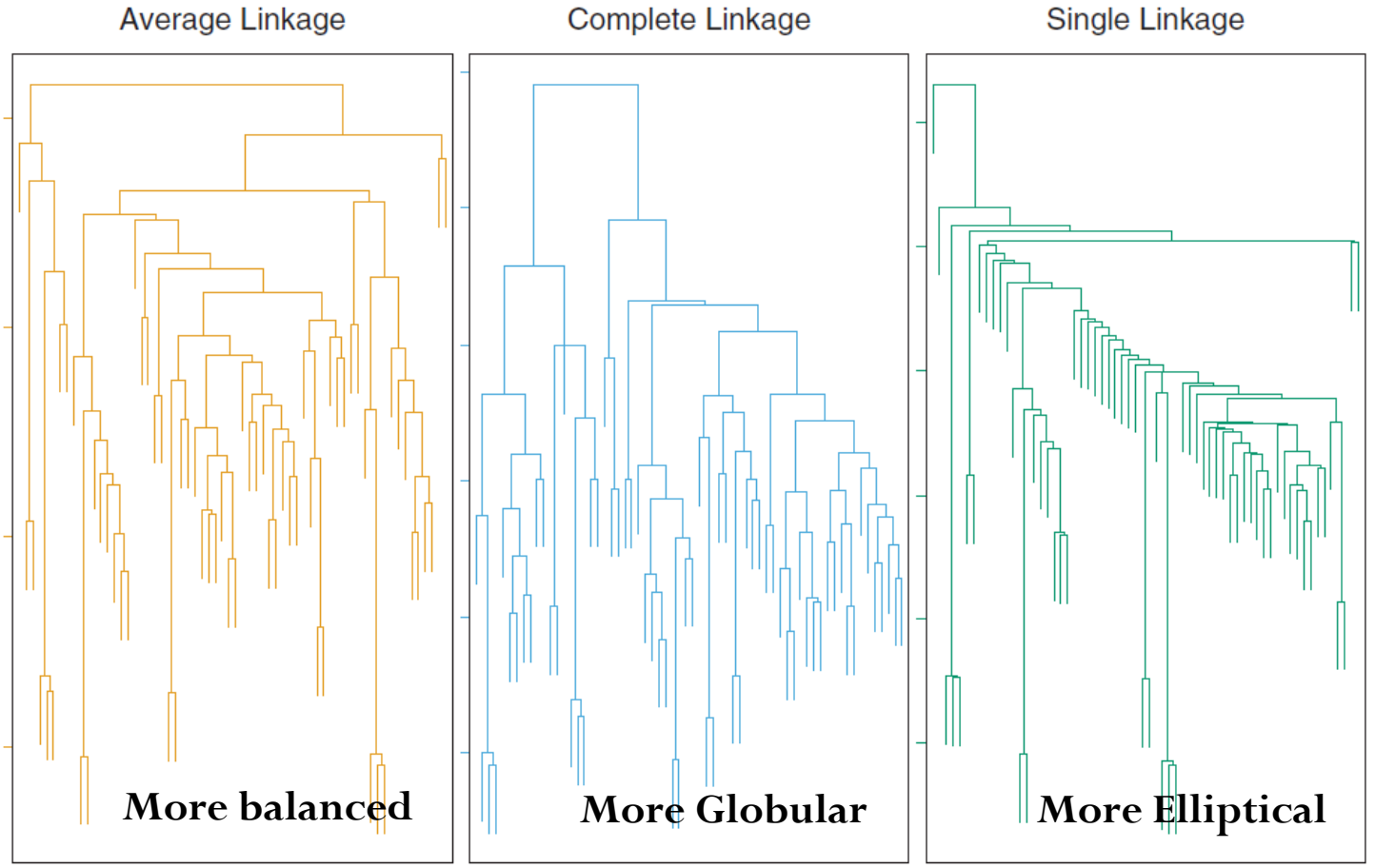
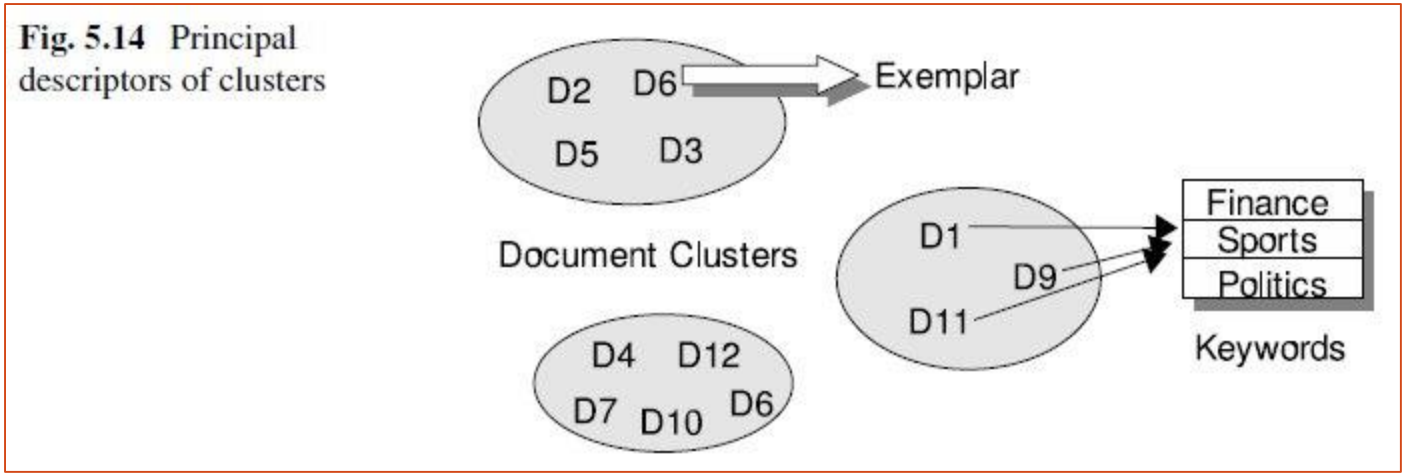


FIGURE 10.12. *Average, complete, and single linkage applied to an example data set. Average and complete linkage tend to yield more balanced clusters.*

What Do a Cluster's Labels Mean?

- At the end of the process, we have ***k*** clusters.
- We can assign numbers to correspond to different clusters.
- Yet, we need greater insight into the process than just assigning a label without expressing meaning.
- The principal descriptors of the automatically generated cluster labels are usually explained and motivated by key words or exemplar documents.



What Do a Cluster's Labels Mean?

- **Keyword Approach:**
  - The simplest approach is to take **the most frequent words** and then remove the stopwords.
  - Alternatively, the words with **the largest average tf-idf** measures might be used.
  - Instead of relying completely on automated procedures, **the human expert** can review the words generated by automated procedures and make a final determination.
- **Exemplar Approach:**
  - One simple approach is to select **the document** that is **most similar to the cluster mean vector**.
  - The **human expert** can read and review these retrieved documents to understand the results of the clustering process and to reach some decisions about their value.

Applications

- **The problems reported to help desks** are good examples of the potential benefits of clustering.
  - E.g., Clusters for Network issues, Operating Systems issues, etc.
- We have presented clustering as **the assignment of labels** to documents.
  - A problem with the spreadsheet that represents data for prediction—it was missing the last column, and now we have methods to compute and fill in the last column.
- The hierarchical methods give **structure to the data**, producing a **taxonomy** that maps relationships among the documents.
  - These are ways of summarizing and organizing documents whose connections are poorly understood.

Evaluation of Performance

- For prediction problems, evaluation is straightforward.
- The computed answers are compared with the correct answers, and accuracy, precision, or error rates are determined.
- **Clustering starts with unlabeled data**, and using the same approach to evaluation is not feasible.
- One way of evaluating clustering is to **compute a cluster mean and its variance or standard deviation**.
- Consider **a deviation from the mean** to be **a form of error**.
- If **documents within a cluster** are **similar**, **the variance of the mean** will be **low**.

Evaluation of Performance

- The overall measure of **variance** for all documents is computed, where *S(i)* is the similarity result for document *i* (e.g., against its cluster mean), and *MS(i)* is the average of the similarity measures for all documents in the same cluster as document *i*.
- $Variance = \sum_i \frac{(S(i) - MS(i))^2}{n}$
- $Standard\ Deviation = Error = \sqrt{Variance}$
- The variance can be computed over **all documents in the collection** to get a global indication of effectiveness.
- It can also be computed for an **individual cluster** to determine its performance and to compare it with other clusters.

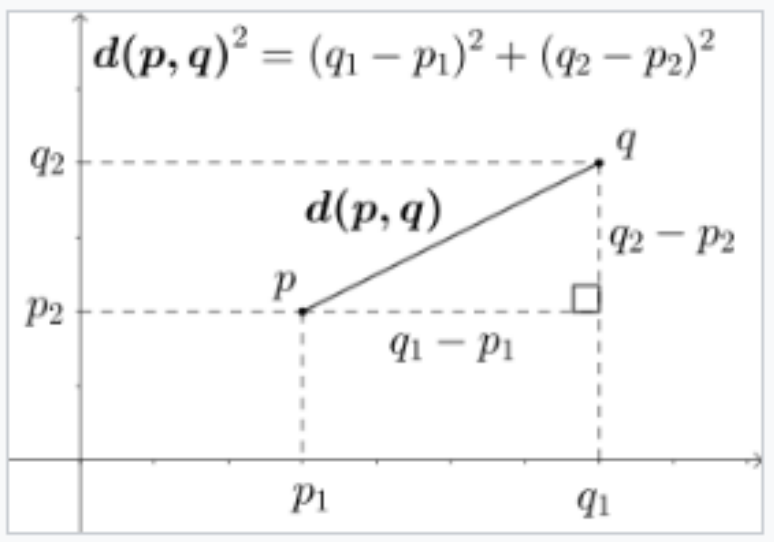
$$E(k) = \sum_{i=1}^{n} \frac{(x_i - m_{c_i})^2}{n}$$

Evaluation of Performance

- How is the similarity measure *S(i)* of a document computed?
- There are many choices for similarity measures.
- For example, we might simply **count shared words**.
  - For *k*-means, each document would be compared with the mean vector to see the number of shared words.
  - For instance, *S(i)* is the number of words shared between document *i* and the mean vector of its cluster, and the average of all *S(i)* in the same cluster is *MS(i)*: Use **Inner product of term presence vectors**.
- Alternatively, similarity can be computed using **cosine similarity**.
  - Use **Inner product of normalized tf or tf-idf vectors**.

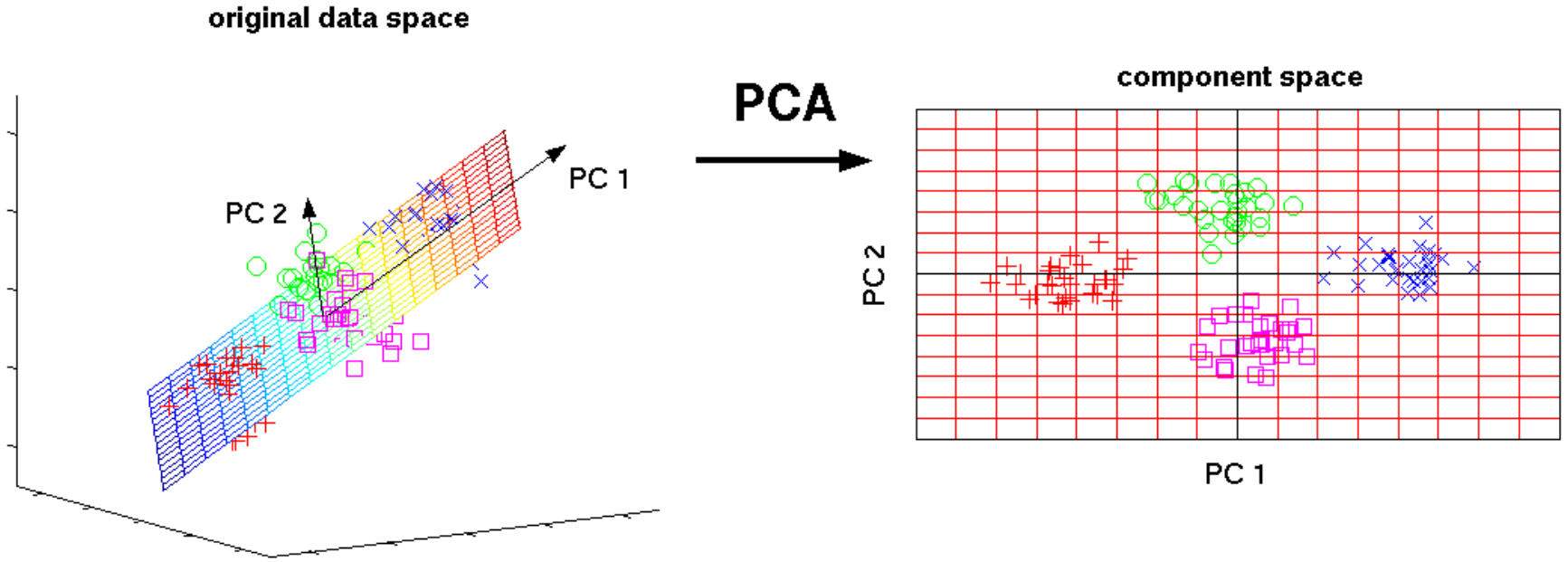
Evaluation of Performance

- There are many choices for similarity measures (cont.).
- Another measure of the similarity of two vectors is **Euclidean Distance** (or L_2 distance).
  - $| \vec{x} - \vec{y} | = \sqrt{\sum_{i=1}^{M}(x_i - y_i)^2}$



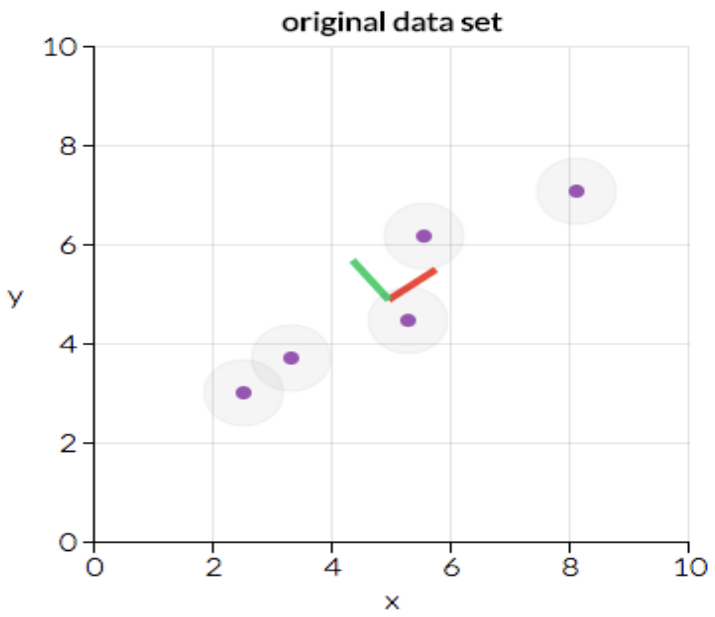
Principal Component Analysis

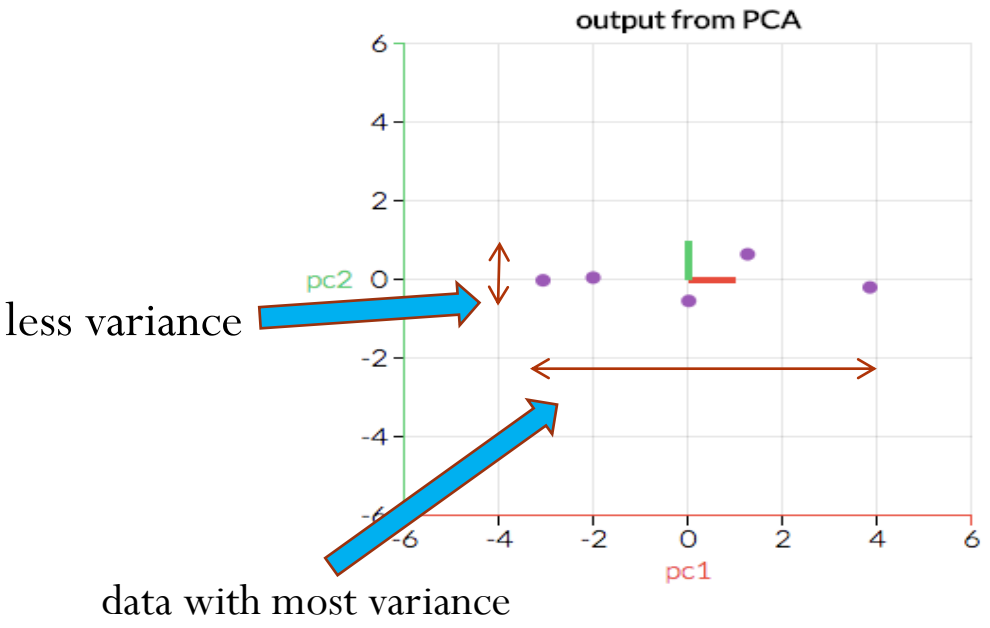
- When faced with a large set of correlated variables, principal components summarize this set with a **smaller number of representative** variables that collectively **explain most** of the **variability** in the original set.
- PCA produces a **low-dimensional** representation of a dataset by finding a linear combinations of the variables that have **maximal variance**, and are mutually uncorrelated (**independent**).



PCA

- Principal components are the underlying structure in the data showing the directions where there is the most variance (data is most spread out).
- In the two-dimensional example, by identifying which "directions" are most "important," we can compress or project our data into a smaller space by dropping the "directions" that are the "least important." (in this case pc2)





PCA

- Variance refers to the spread of a data set. It's a measurement used to identify **how far** each number in the data set is **from the mean**.

$$var(x) = \frac{\sum(x_i - \bar{x})^2}{N} \qquad cov(x,y) = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{N}$$

- Covariance provides insight into **how two variables are related** to one another.
- More precisely, covariance refers to the measure of how two random variables in a data set **will change together**. It is a measure of the extent to which corresponding elements from two sets of ordered data move in the same direction.
- A **positive** covariance means that the two variables at hand are positively related, and they move in the **same** direction.
- A **negative** covariance means that the variables are inversely related, or that they move in **opposite** directions.

PCA – Eigenvalues & Eigenvectors

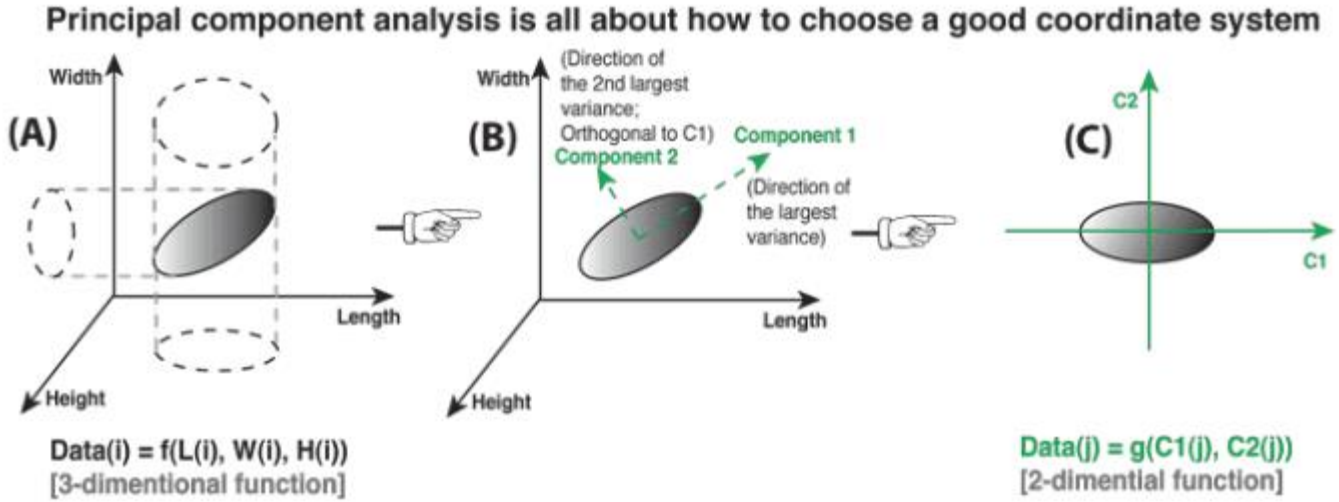
- **Eigenvectors** are vectors that the merely elongates or shrinks (without change in direction) a **matrix A** in a linear transformation, and the amount that they elongate/shrink by is the **eigenvalue**. Both are used to capture key information that is stored in a large matrix.
- A (non-zero) vector **v** of dimension *N* is an **eigenvector** of square N x N matrix **A** if it satisfies the linear equation: $\mathbf{A} * \mathbf{v} = \lambda * \mathbf{v}$

  where λ is a scalar, termed the **eigenvalue** corresponding to **v**.

- **Eigenvalue Decomposition (Factorization)**
  - Let **A** be a square $n \times n$ matrix with n linearly independent eigenvectors. Then **A** can be factorized as $\mathbf{A} = \mathbf{Q} * \lambda * \mathbf{Q}^{-1}$
    - where **Q** is a matrix comprised of the **eigenvectors**, λ is the diagonal matrix comprised of the **eigenvalues** along the diagonal, and Q^{-1} is the **inverse** of the matrix comprised of the eigenvectors.

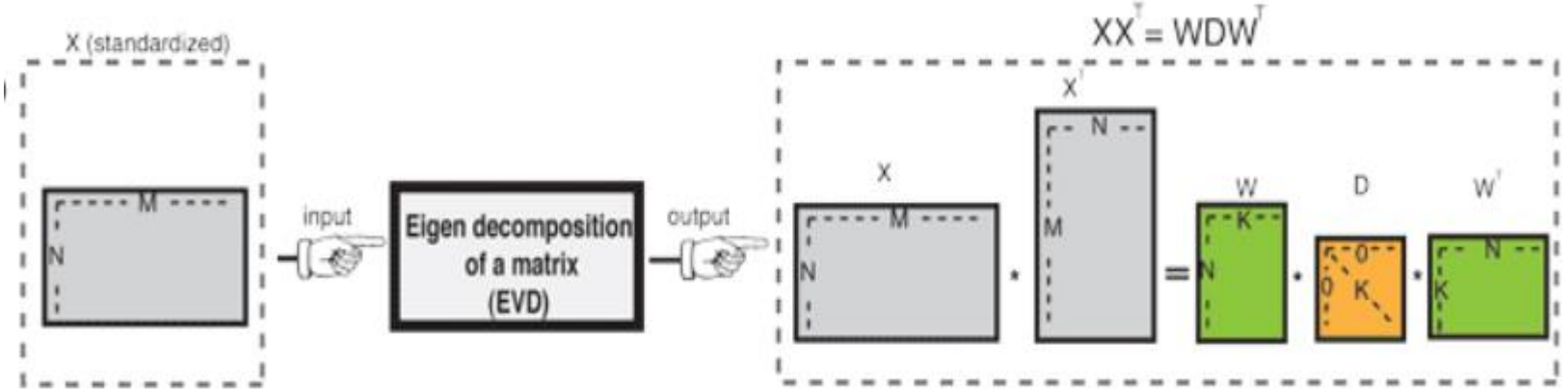
PCA

- From a high-level view PCA has three main steps:
- (1) Compute the covariance matrix of the data
- (2) Compute the eigenvalues and vectors of this covariance matrix (using Eigenvalue decomposition)
- (3) Use the eigenvalues and vectors to select only the most important feature vectors and then transform your data onto those vectors for reduced dimensionality.

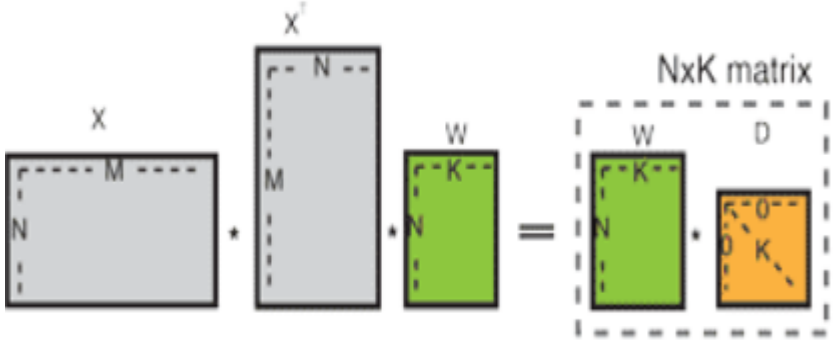


PCA

- Define a NxM matrix that can be transformed into a NxK (K<=M) matrix.
- Using Eigenvalue Decomposition output two new matrices W and D.
- **W** contains all principal component vectors (**eigenvectors**), **D** (**eigenvalues**) contains ranks of those vectors giving: $\mathbf{XX^T=WDW^T}$, where $\mathbf{XX^T}$ is the **covariance** matrix, X^T and W^T are transposes of X and W.



Since, WD is a NxK (K<=M) matrix which is exactly of the same format we want to transform our data set into. Re-write the equation as: $XX^TW=WD$

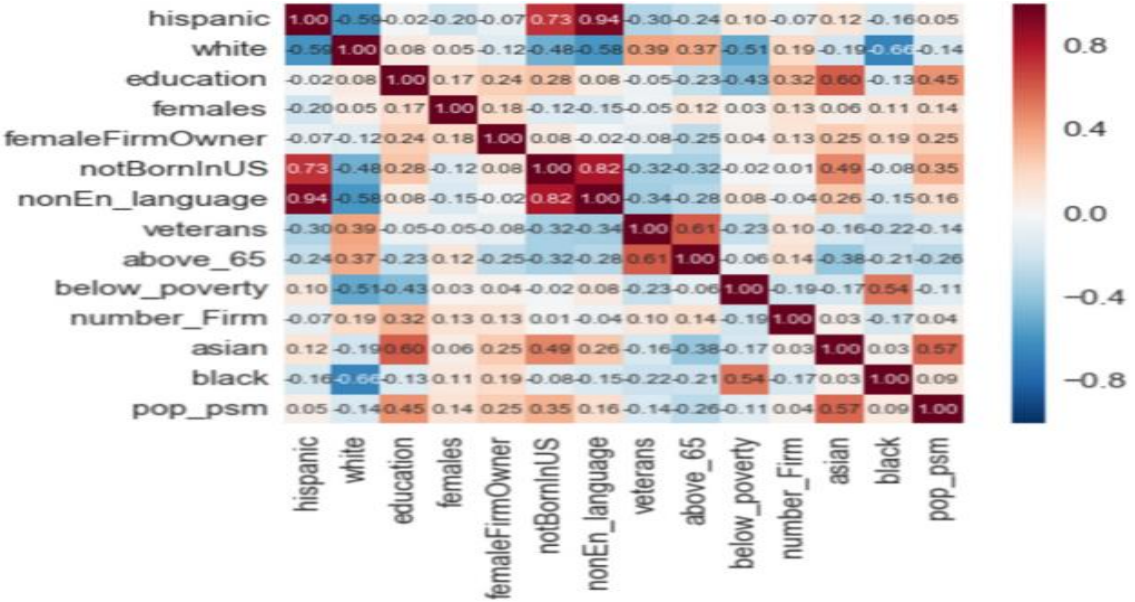


PCA

- The eigenvectors (principal components) of the covariance matrix represent the vector directions of the new feature space and the eigenvalues represent the magnitudes of those vectors.
- An eigenvalue is the number indicating how much variance (how much spread) there is in the data in that direction. The eigenvector with the highest eigenvalue is therefore the principal component.
- Thus, this vector holds a lot information about the data, since any movement along that vector causes large "variance". On the other hand vectors with small eigenvalues have low variance and thus the data does not vary greatly when moving along that vector. As this feature isn't very important it is removed.
- The essence of eigenvalues and vectors within PCA is to find the vectors that are the most important in representing the data and discard the rest.

PCA Example

- Predict the county democrat winner of USA Presidential election using the demographic information of each county.
- *Data Description:* It consists of county wise **demographic information** of all 50 states in USA and primary presidential election results of 28 states. https://www.kaggle.com/benhamner/2016-us-election
- Principal component analysis used to **find fewer uncorrelated components** which can be further used in logistic regression as independent variables.



PCA Example

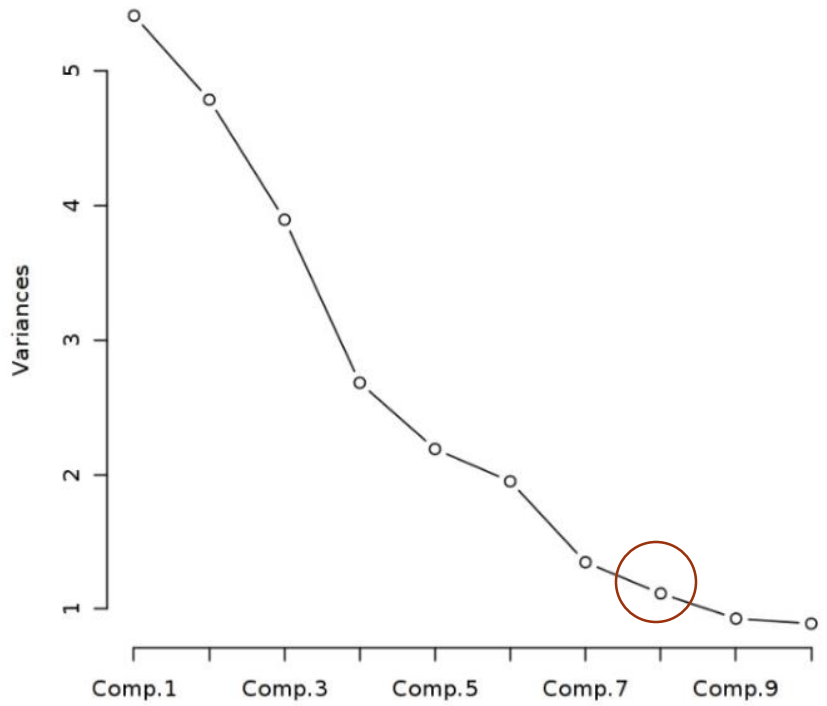
```
from sklearn.decomposition import PCA
# Standardizing the features
x = StandardScaler().fit_transform(x)

pca = PCA(n_components=10)
principalComponents = pca.fit_transform(x)
principalDf = pd.DataFrame(data = principalComponents, , columns = ['Comp.1', 'Comp.2', 'Comp.3', 'Comp.4', 'Comp.5', 'Comp.6', 'Comp.7', 'Comp.8', 'Comp.9', 'Comp.10'])
```

```
Importance of components:
                          Comp.1     Comp.2     Comp.3     Comp.4     Comp.5
Standard deviation     2.3268438 2.1883364 1.9737927 1.6379817 1.47981932
Proportion of Variance 0.1804734 0.1596272 0.1298619 0.0894328 0.07299551
Cumulative Proportion  0.1804734 0.3401006 0.4699625 0.5593953 0.63239084
                          Comp.6     Comp.7     Comp.8     Comp.9    Comp.10
Standard deviation     1.39620325 1.16079813 1.05600831 0.96341907 0.94399457
Proportion of Variance 0.06497945 0.04491508 0.03717179 0.03093921 0.02970419
Cumulative Proportion  0.69737029 0.74228536 0.77945715 0.81039636 0.84010055
```

Observing the summary result, 8 principal components were chosen which explained 80% variance of the dataset. (Look out for the cumulative proportion row to get the total variance explained)

PCA Example

- The number of components to be used for further analysis can also be determined using the following scree plot: (look out for the elbow shaped point)
- These 8 principal component scores can be used as the predictor variables to build logistic regression model which predicts binary outcome stating the winner.



Referenced Materials

- Fundamentals of Predictive Text Mining, Sholom M. Weiss, Nitin Indurkhya, and Tong Zhang, Springer.
  - Chapter 5
- An Introduction to Statistical Learning with Application in R, Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani.
  - Chapter 10 Unsupervised Learning