CSE 564
Visualization & Visual Analytics

Data and Dimension Reduction

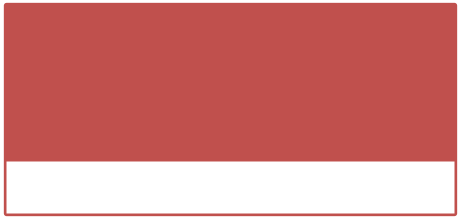
Klaus Mueller

Computer Science Department
Stony Brook University

Lecture	Topic	Projects
1	Intro, schedule, and logistics	
2	Applications of visual analytics	
3	Basic tasks, data types	Project #1 out
4	Data assimilation and preparation	
5	Introduction to D3	
6	Bias in visualization	
7	Data reduction and dimension reduction	
8	Visual perception	Project #2(a) out
9	Visual cognition	
10	Visual design and aesthetics	
11	Cluster analysis: numerical data	
12	Cluster analysis: categorical data	Project #2(b) out
13	High-dimensional data visualization	
14	Dimensionality reduction and embedding methods	
15	Principles of interaction	
16	Midterm #1	
17	Visual analytics	Final project proposal call out
18	The visual sense making process	
19	Maps	
20	Visualization of hierarchies	Final project proposal due
21	Visualization of time-varying and time-series data	
22	Foundations of scientific and medical visualization	
23	Volume rendering	Project 3 out
24	Scientific and medical visualization	Final Project preliminary report due
25	Visual analytics system design and evaluation	
26	Memorable visualization and embellishments	
27	Infographics design	
28	Midterm #2	

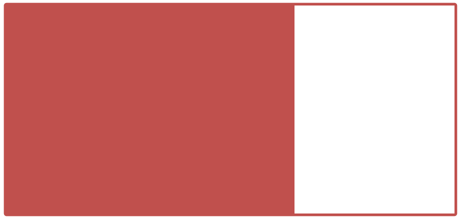
Data Reduction – How?

Reduce the number of data items (samples):



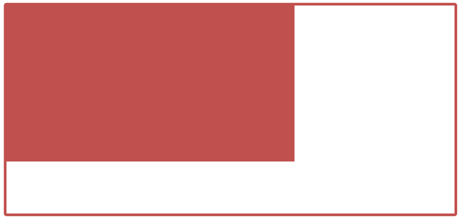
- random sampling
- stratified sampling

Reduce the number of attributes (dimensions):



- dimension reduction by transformation
- dimension reduction by elimination

Usually do both



Utmost goal

- keep the gist of the data
- only throw away what is redundant or superfluous
- it’s a one way street – once it’s gone, it’s gone

Data Reduction

Sampling

- random
- stratified

Data summarization

- binning (already discussed)
- clustering (see a future lecture)
- dimension reduction (see next lecture)

Data Reduction – Why?

Because…

- need to reduce the data so they can be feasibly stored
- need to reduce the data so a mining algorithm can be feasibly run

What else could we do

- buy more storage
- buy more computers or faster ones
- develop more efficient algorithms (look beyond O-notation)

However, in practice, all of this is happening at the same time

- unfortunately, the growth of data and complexities is always faster
- and so, data reduction will always be important

Which Samples to Discard?

Good candidates are *redundant* data

- how many cans of ravioli will you buy?

Sampling Principles

Keep a representative number of samples:

- pick one of each
- or maybe a few more depending on importance

How to Pick?

You are faced with collections of many different data

- they are usually not nicely organized like this:

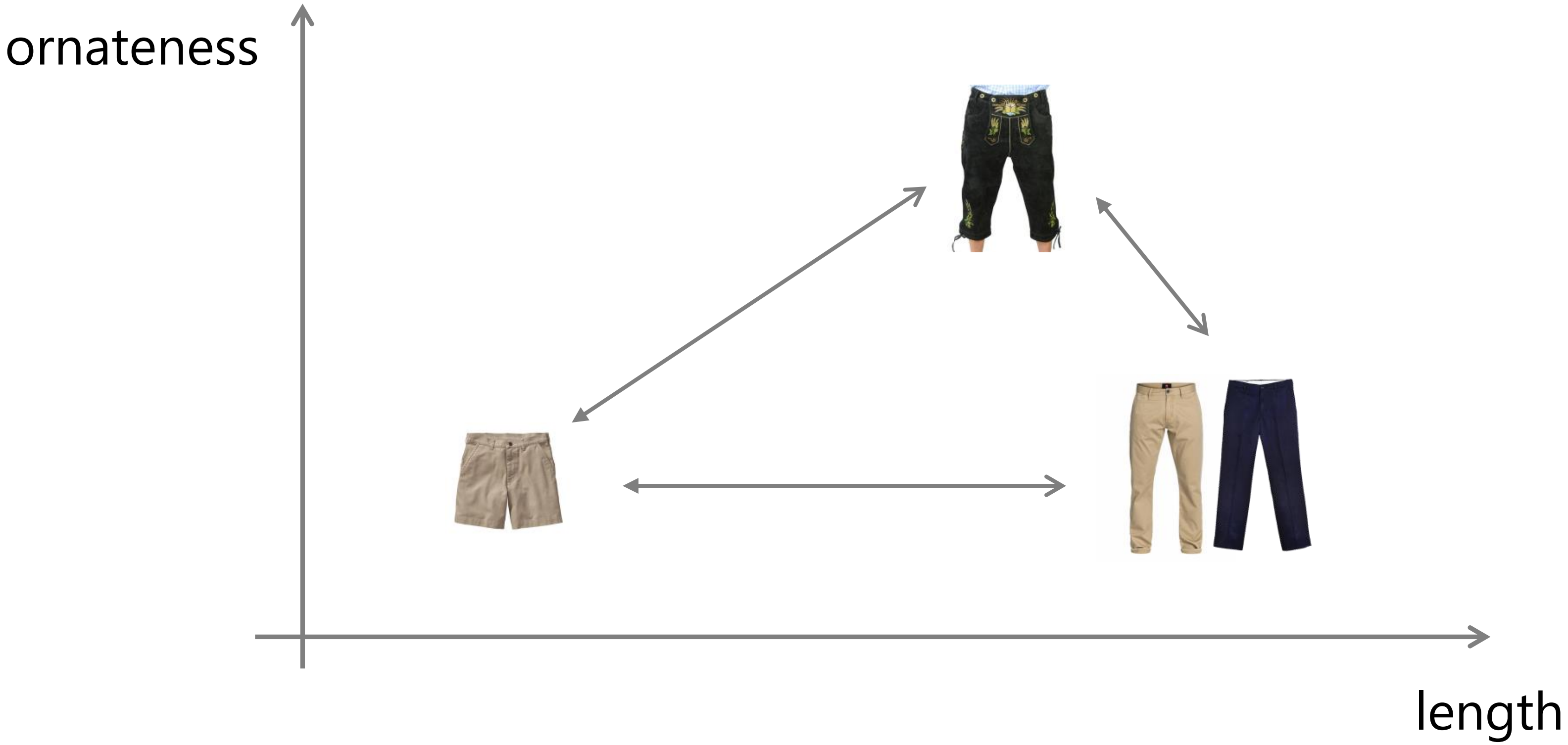
- but more like this:

Measure of Similarity

Are all of these items pants?

- need a measure of similarity
- it's a distance measure in high-dimensional feature space

Feature Space

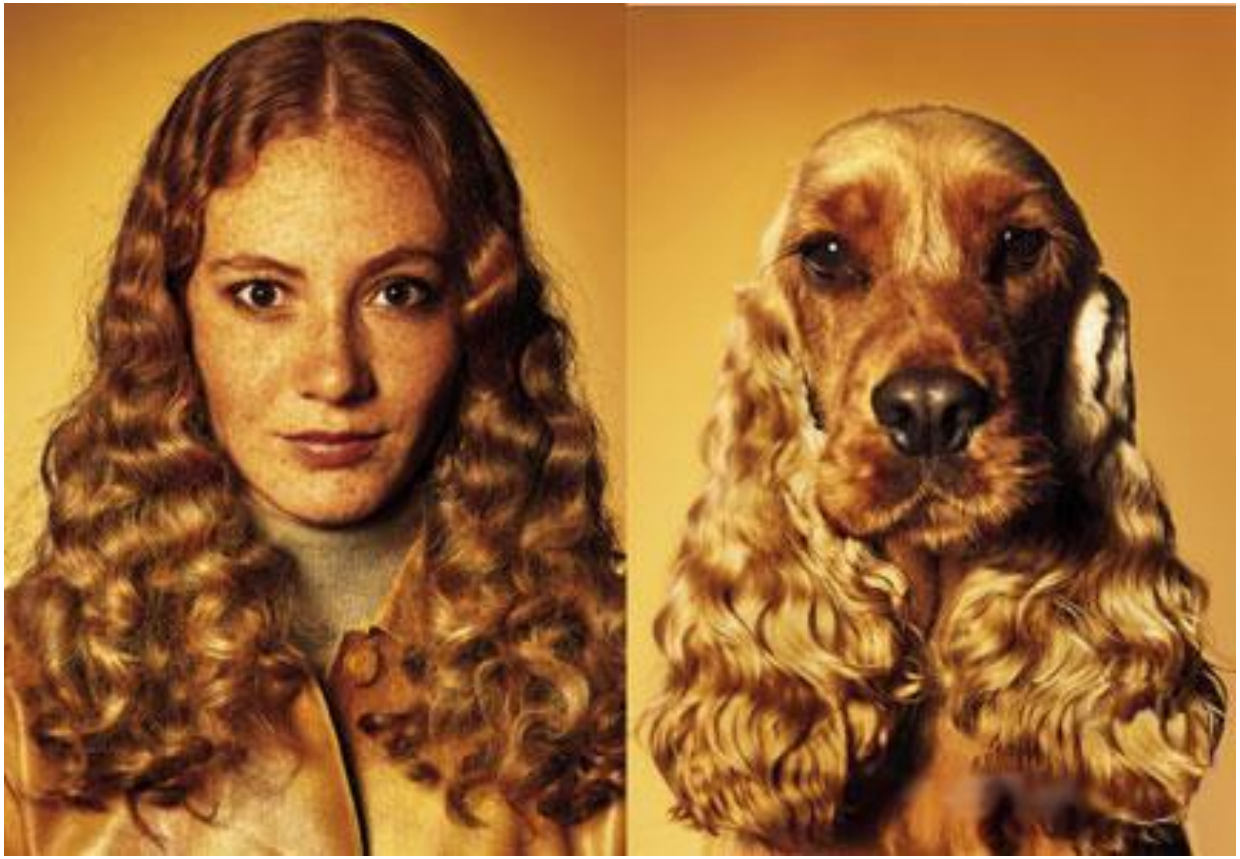


We did not consider color, texture, size, etc…

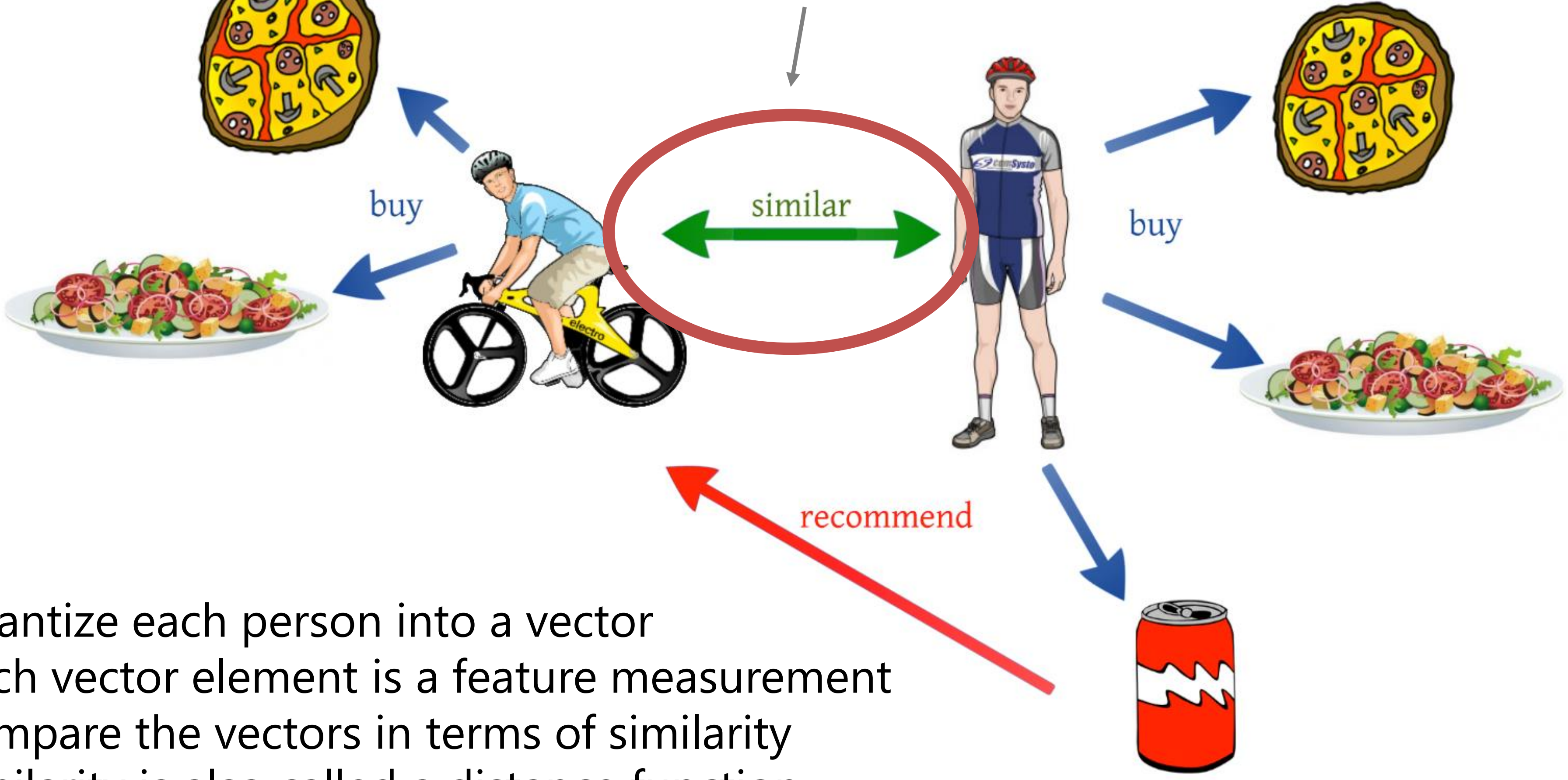
- this would have brought more differentiation (blue vs. tan pants)
- the more features, the better the differentiation

How Many Features Do We Need?

Measuring similarity can be difficult



Back To Similarity Functions



needs to be
accurately measured

quantize each person into a vector
each vector element is a feature measurement
compare the vectors in terms of similarity
similarity is also called a distance function

Data Vectors

Pant:

<length, ornateness, color>

Food delivery customer:

<type-pizza, type-salad, type-drink>

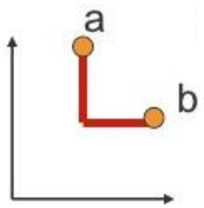
Examples:

- pants: <long, plain, tan>, <short, ornate, blue>, …
  expressed in numbers: <30”, 1, 2>, <15”, 2, 5>

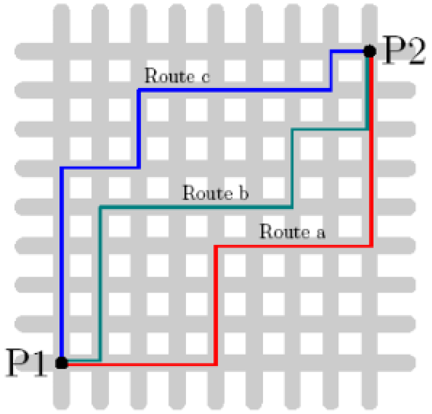
- food: <pepperoni, tossed, none>, <pepperoni, tossed, coke>, …
  expressed in numbers: <1, 1, 0>, <1, 1, 3>

Metric Distances

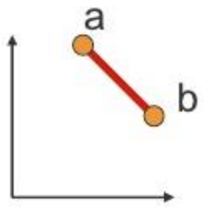
Manhattan distance



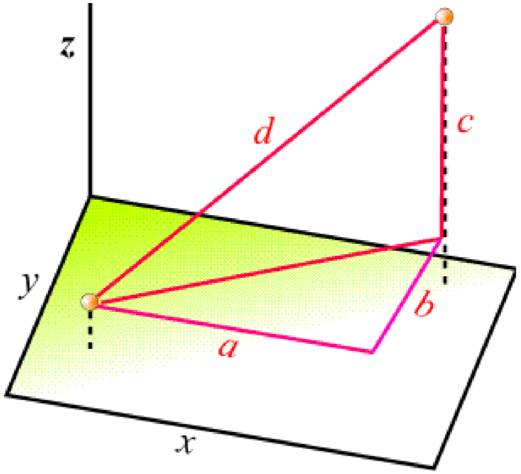
$$\text{dist(a,b)} = \|a - b\|_1 = \sum_i |a_i - b_i|$$



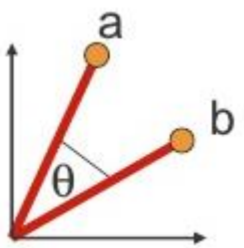
Euclidian distance



$$\text{dist(a,b)} = \|a - b\|_2 = \sqrt{\sum_i (a_i - b_i)^2}$$



Cosine Similarity



$$s(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x}\cdot\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} = \frac{\sum_{i=0}^{n-1} x_i y_i}{\sqrt{\sum_{i=0}^{n-1} (x_i)^2} \times \sqrt{\sum_{i=0}^{n-1} (y_i)^2}}$$

how is this related to correlation?

Pearson’s Correlation = correlation similarity

mean across all data values for attribute x, y

$$r = r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

e.g. the “average looking” pair of pants in terms of attribute x, y

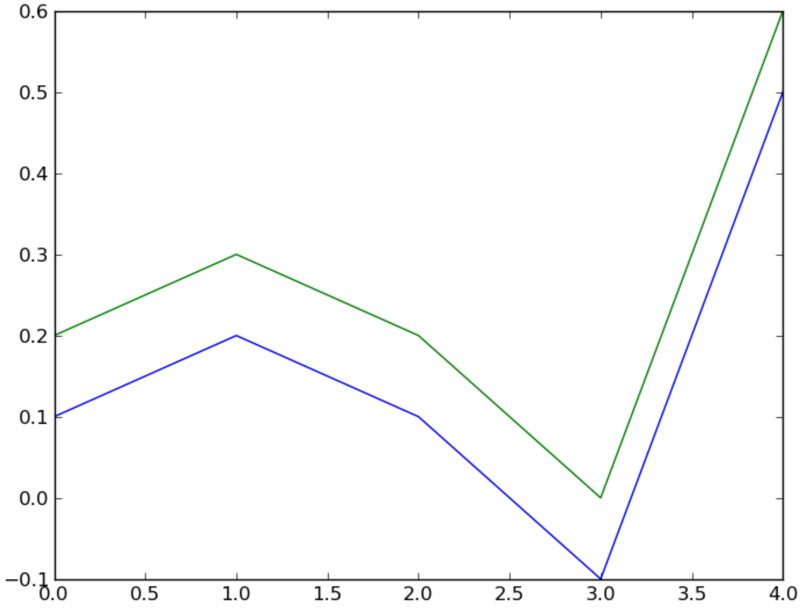
Correlation vs. Cosine Distance

Correlation distance is invariant to addition of a constant

- subtracts out by construction
- green and blue curve have correlation of 1
- but cosine similarity is < 1
- correlated vectors just vary in the same way
- cosine similarity is stricter

Both correlation and cosine similarity are invariant to multiplication with a constant

- invariant to scaling

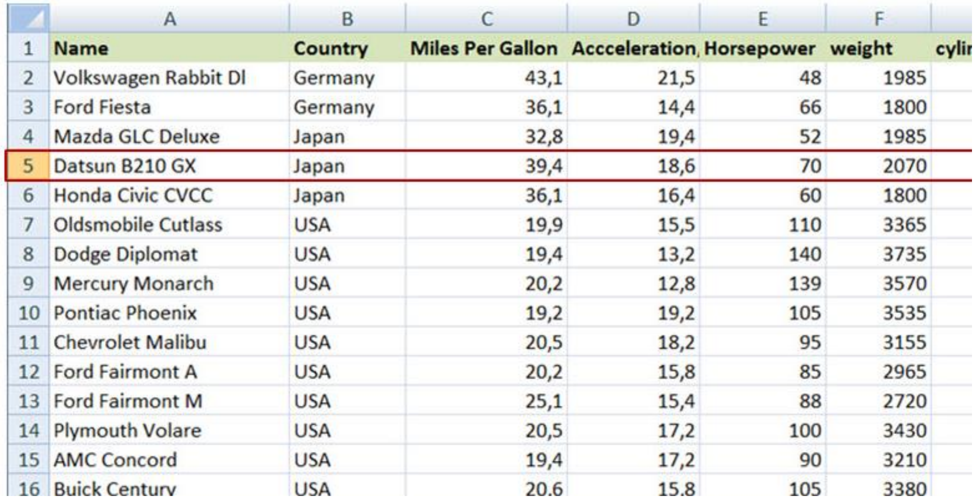


green = blue + 0.1

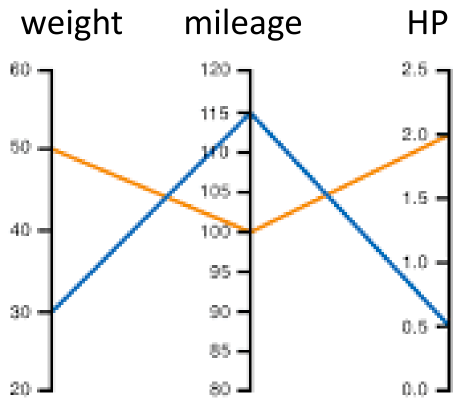
Variables vs. Data Items

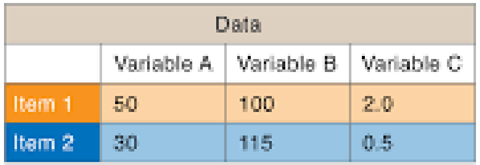
Distances can compare two attributes or two data items

- means and other stats are then measured correspondingly
- mean and std dev mileage and weight, resp. over all cars when computing correlation of weight and mileage
- mean of all attribute values for each car when computing the distance between two cars



	A	B	C	D	E	F	
1	Name	Country	Miles Per Gallon	Acceleration	Horsepower	weight	cylir
2	Volkswagen Rabbit Dl	Germany	43,1	21,5	48	1985	
3	Ford Fiesta	Germany	36,1	14,4	66	1800	
4	Mazda GLC Deluxe	Japan	32,8	19,4	52	1985	
5	Datsun B210 GX	Japan	39,4	18,6	70	2070	
6	Honda Civic CVCC	Japan	36,1	16,4	60	1800	
7	Oldsmobile Cutlass	USA	19,9	15,5	110	3365	
8	Dodge Diplomat	USA	19,4	13,2	140	3735	
9	Mercury Monarch	USA	20,2	12,8	139	3570	
10	Pontiac Phoenix	USA	19,2	19,2	105	3535	
11	Chevrolet Malibu	USA	20,5	18,2	95	3155	
12	Ford Fairmont A	USA	20,2	15,8	85	2965	
13	Ford Fairmont M	USA	25,1	15,4	88	2720	
14	Plymouth Volare	USA	20,5	17,2	100	3430	
15	AMC Concord	USA	19,4	17,2	90	3210	
16	Buick Century	USA	20.6	15.8	105	3380	

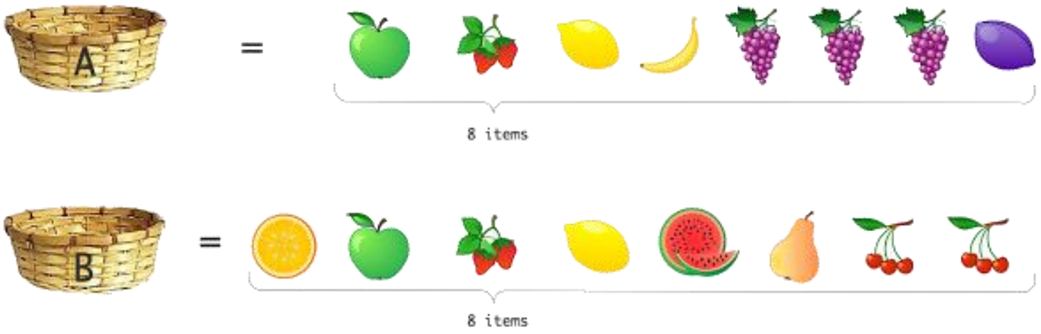


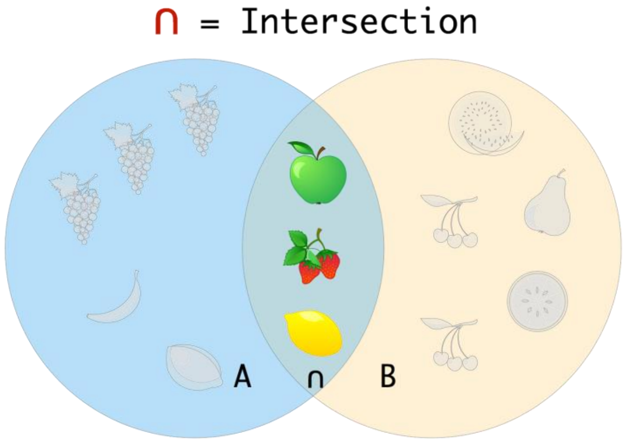


Data			
	Variable A	Variable B	Variable C
Item 1	50	100	2.0
Item 2	30	115	0.5

Jaccard Distance

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$





What’s the Jaccard similarity of the two baskets A and B?

Organizing the Shelf

(c) Gianni Di Loreto 2007

dansays/Flickr

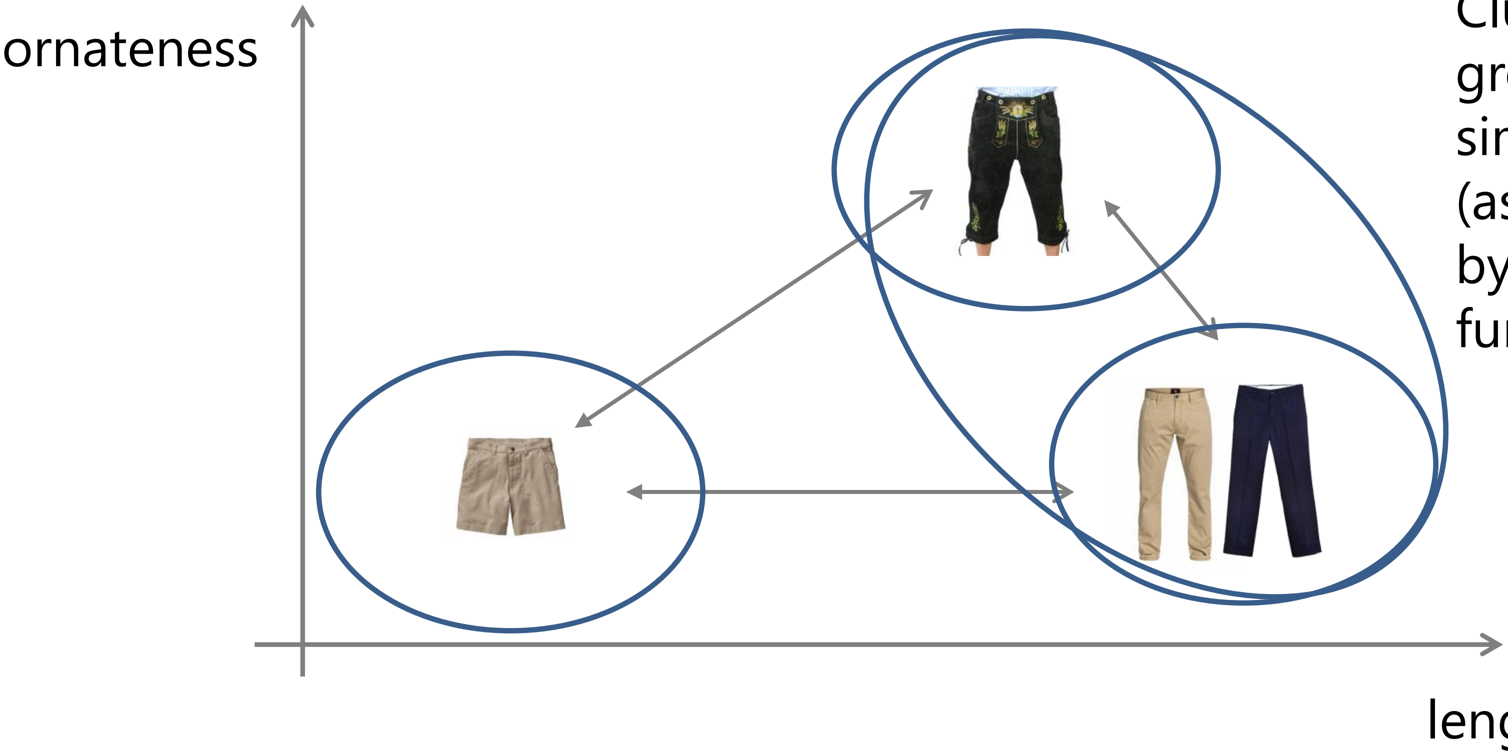
This process is called *clustering*

- and in contrast to a real store, we can make the computer do it for us

What is Clustering?

Note:

- in data mining similarity and distance are the same thing
- so we will use these terms interchangeably



Clustering = grouping of similar items (as determined by the distance function)

WHAT IS A GOOD CLUSTER?

A cluster is a group of objects that are similar

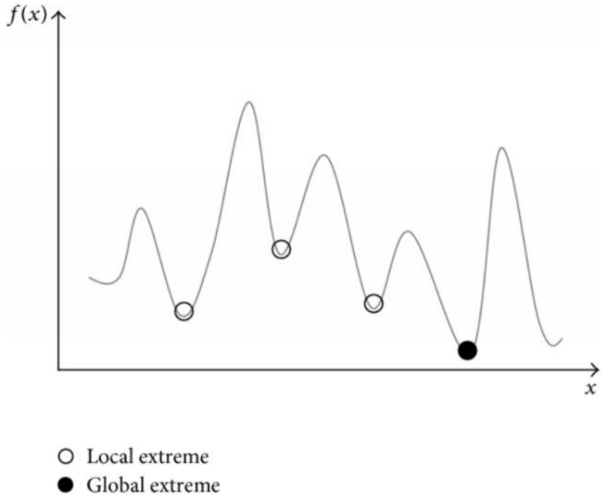
- and dissimilar from other groups of objects at the same time

We need an objective function to capture this mathematically

- the computer will evaluate this function within an algorithm
- one such function is the mean-squared error (MSE)
- and the objective is to minimize the MSE

It's not that easy in practice

- there is only one global minimum
- but often there are many local minima
- need to find the global minimum



Objective – Minimize Squared Error

number of clusters

number of cases

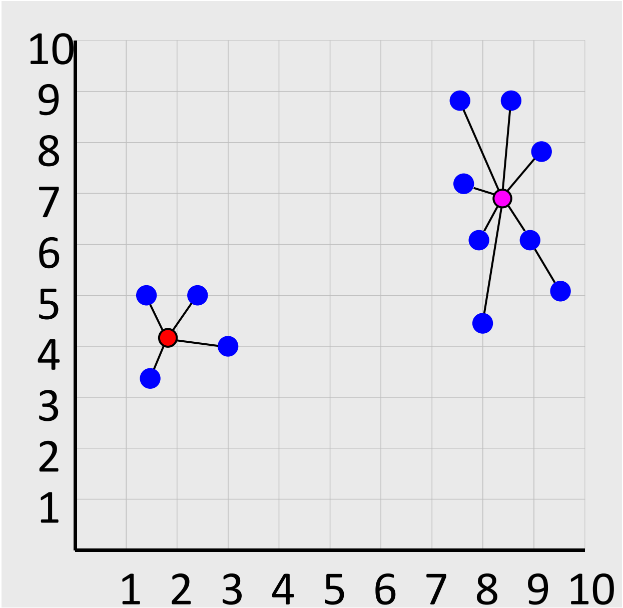
centroid for cluster *j*

case *i*

$$\text{objective function} \leftarrow J = \sum_{j=1}^{k}\sum_{i=1}^{n} \underbrace{\left\|x_i^{(j)} - c_j\right\|^2}_{\text{Distance function}}$$

In this case

- n=12 (blue points)
- k=2 (red points, the computed centroids)
- distance metric used: Euclidian
- minimization seems to be achieved



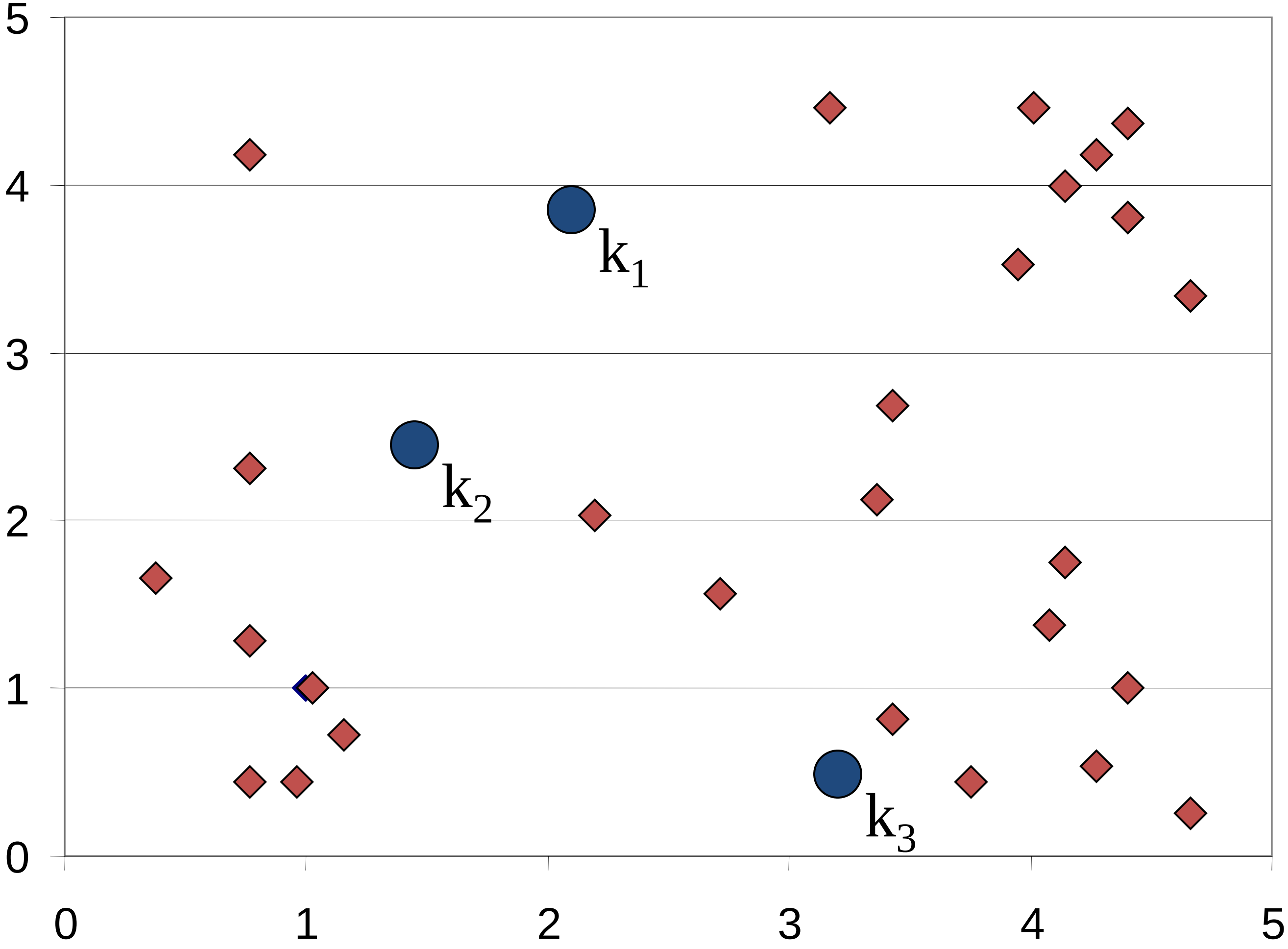
The K-Means Clustering Algorithm

1. Decide on a value for k
2. Initialize the k cluster centers (randomly, if necessary)
3. Decide the class memberships of the N objects by assigning them to the nearest cluster center
4. Re-estimate the k cluster centers, by assuming the memberships found above are correct
5. If none of the N objects changed membership in the last iteration, exit. Otherwise goto 3

The last slide and the next 8 slides contain figures courtesy of Eamonn Keogh, UC Riverside

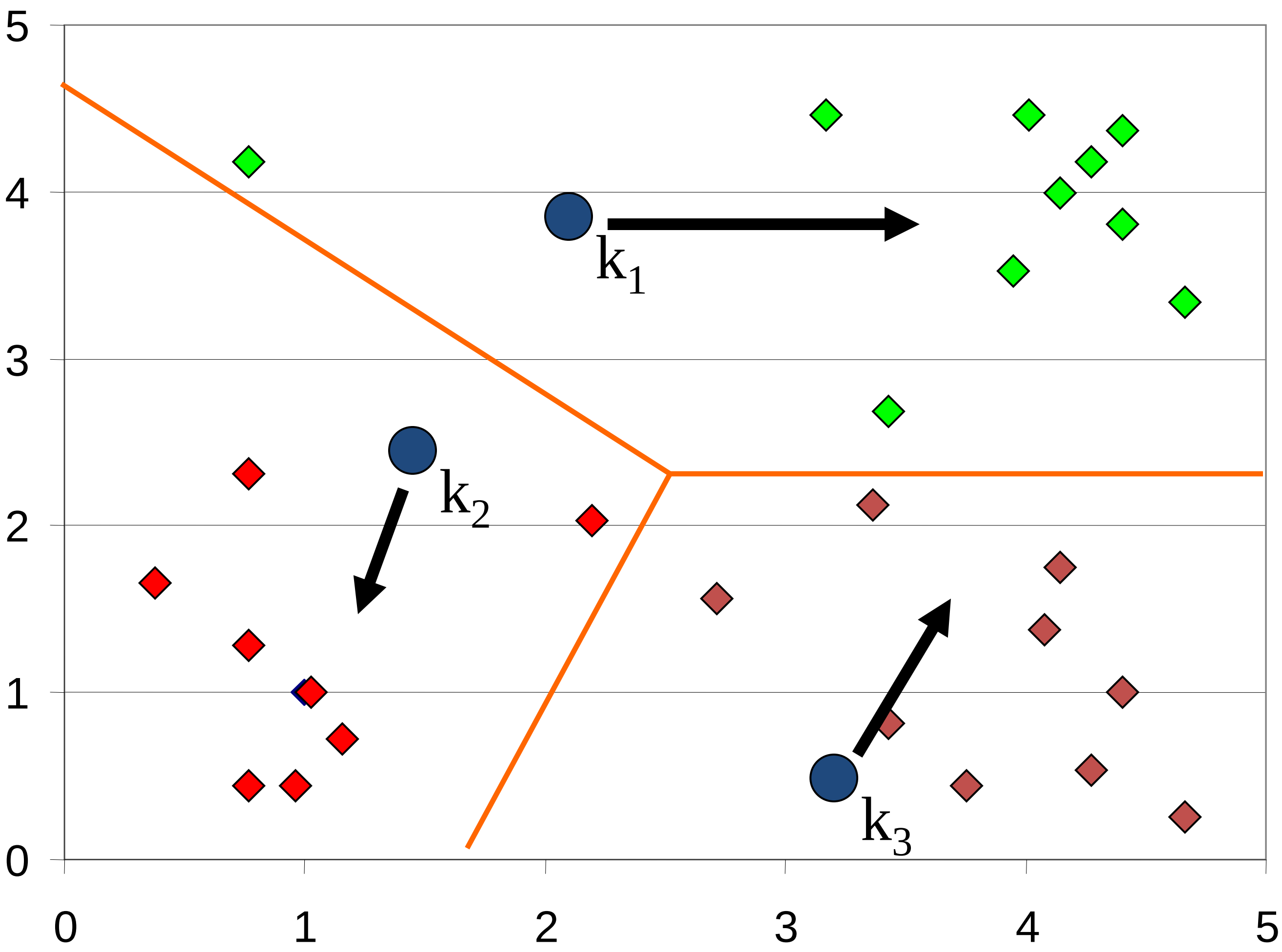
K-means Clustering: Step 1

Algorithm: k-means, Distance Metric: Euclidean Distance



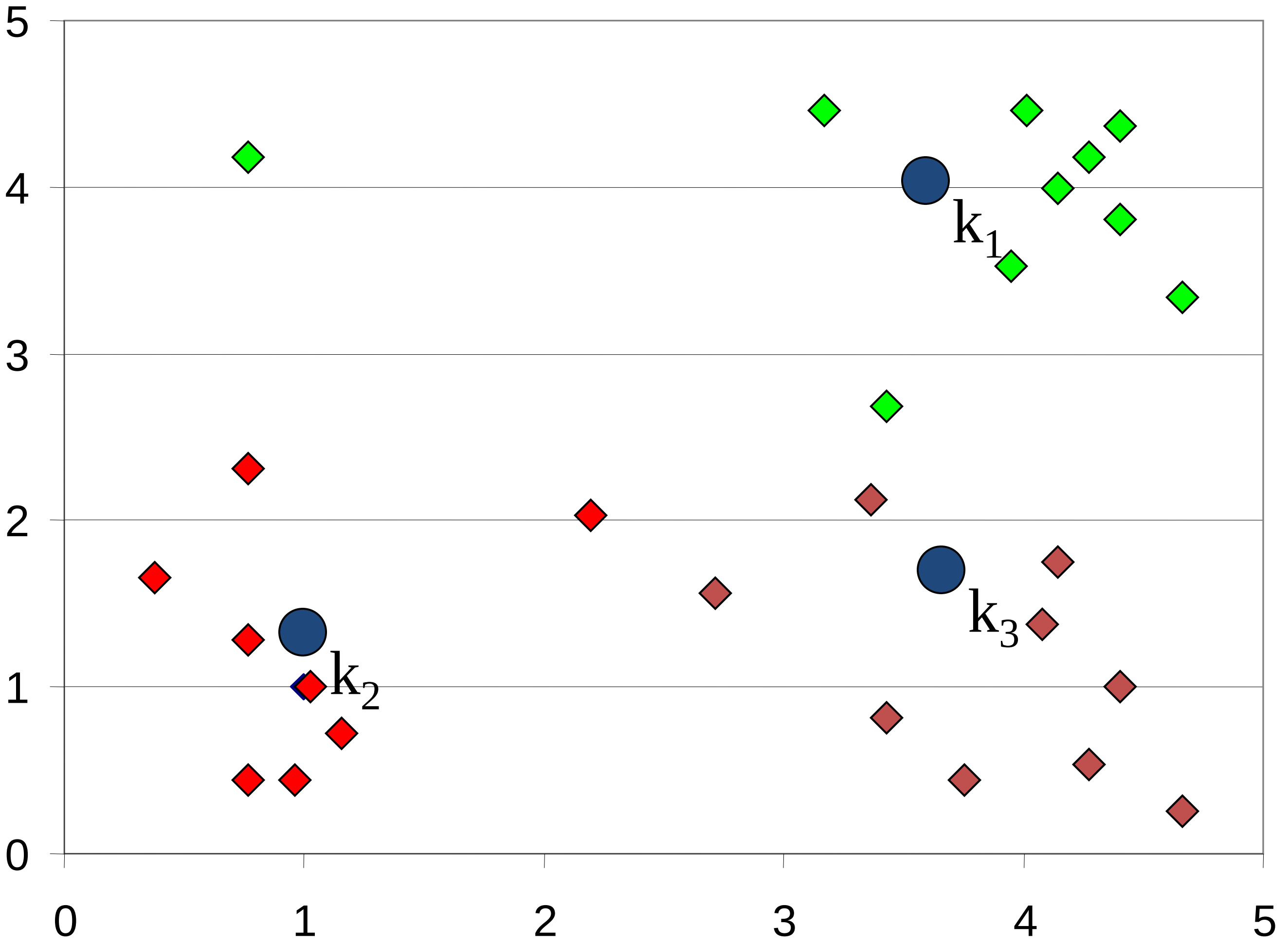
K-means Clustering: Step 2

Algorithm: k-means, Distance Metric: Euclidean Distance



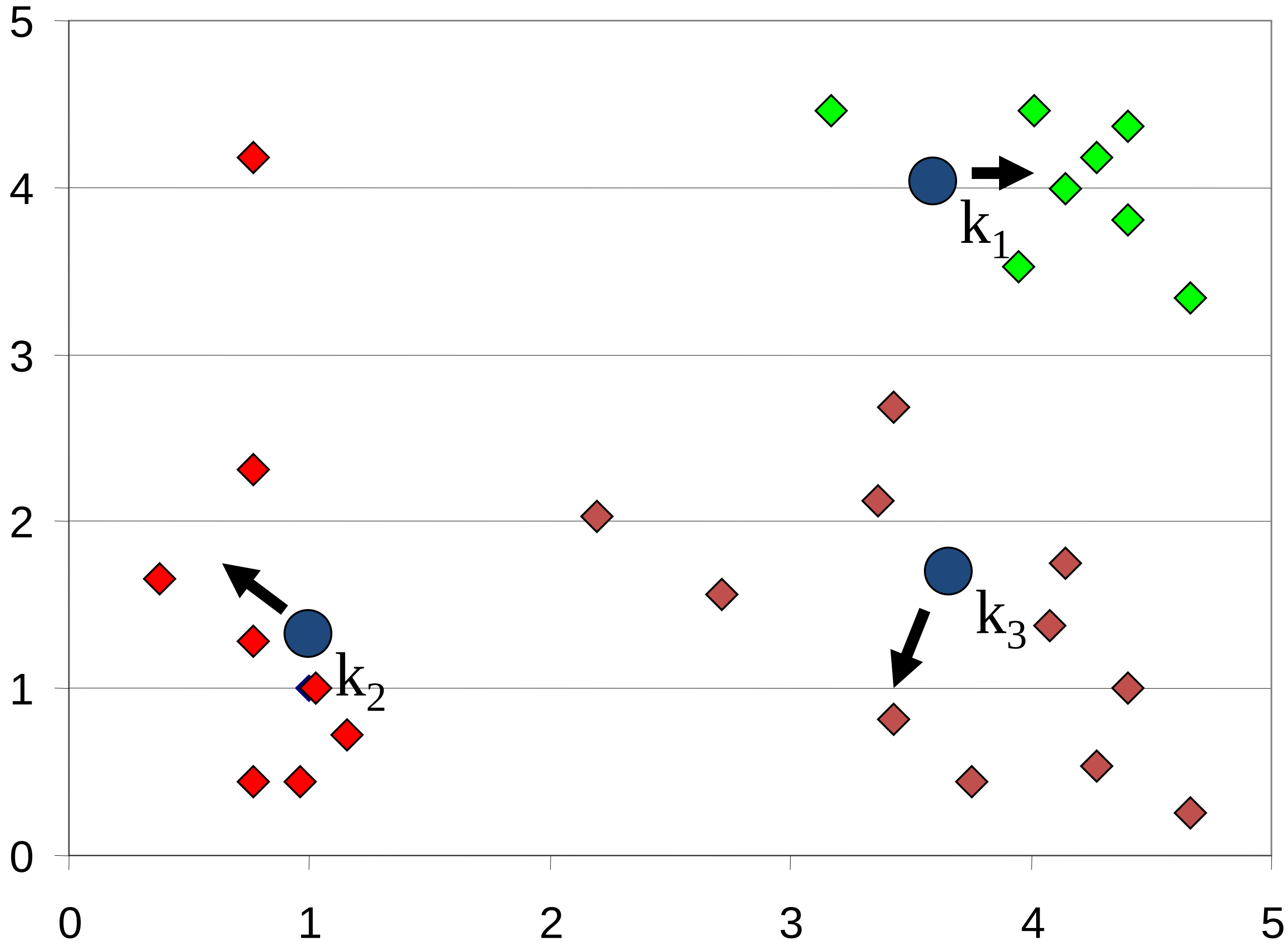
K-means Clustering: Step 3

Algorithm: k-means, Distance Metric: Euclidean Distance



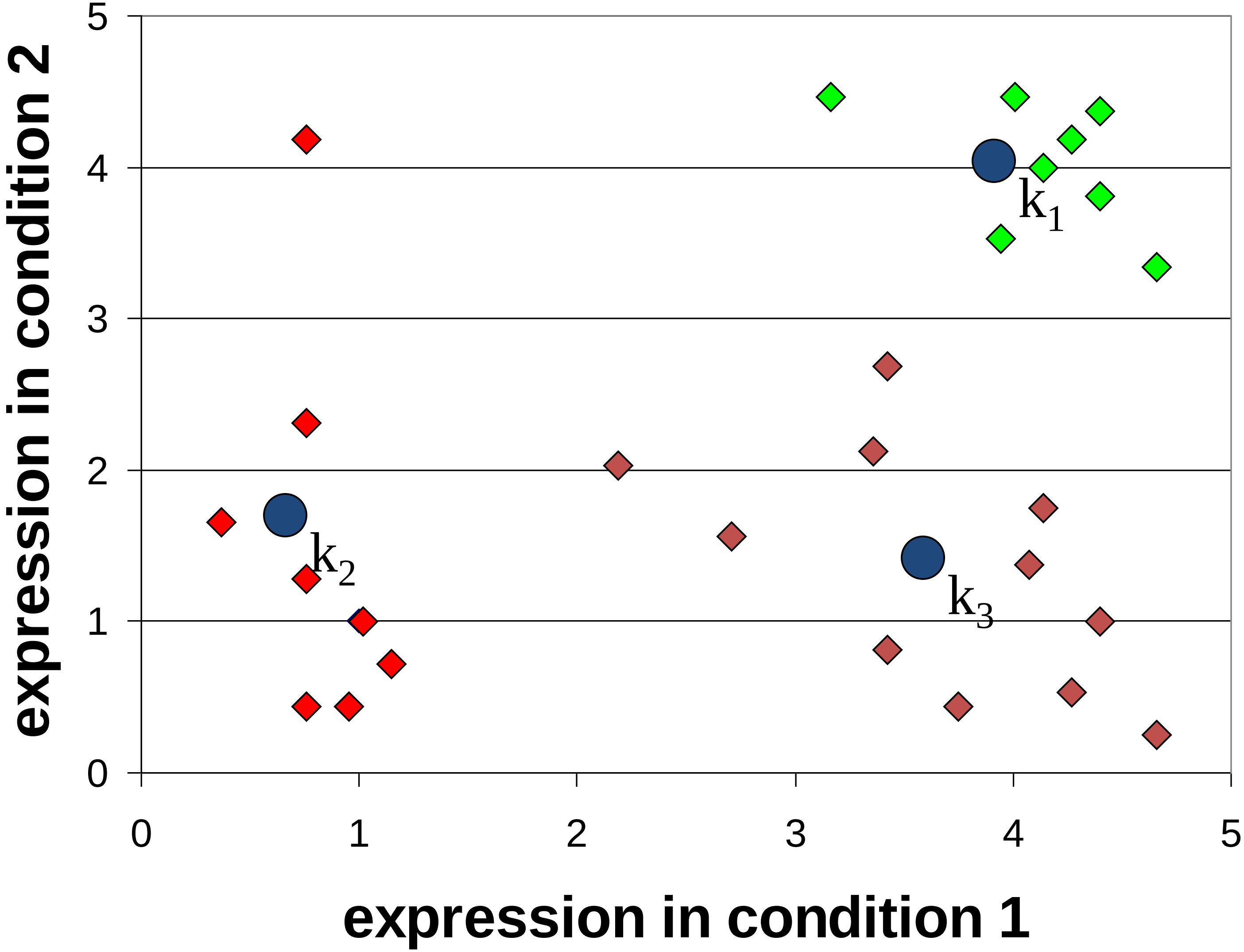
K-means Clustering: Step 4

Algorithm: k-means, Distance Metric: Euclidean Distance



K-means Clustering: Step 5

Algorithm: k-means, Distance Metric: Euclidean Distance



K-Means Algorithm – Comments

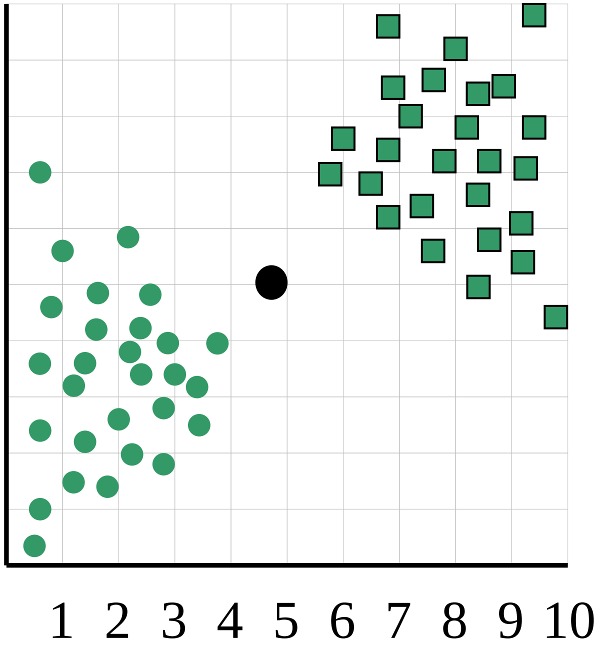
Strengths:

- *relatively efficient*: O(*tkn*), where *n* is # objects, *k* is # clusters, and *t* is # iterations. Normally, *k*, *t* << *n*.
- simple to code

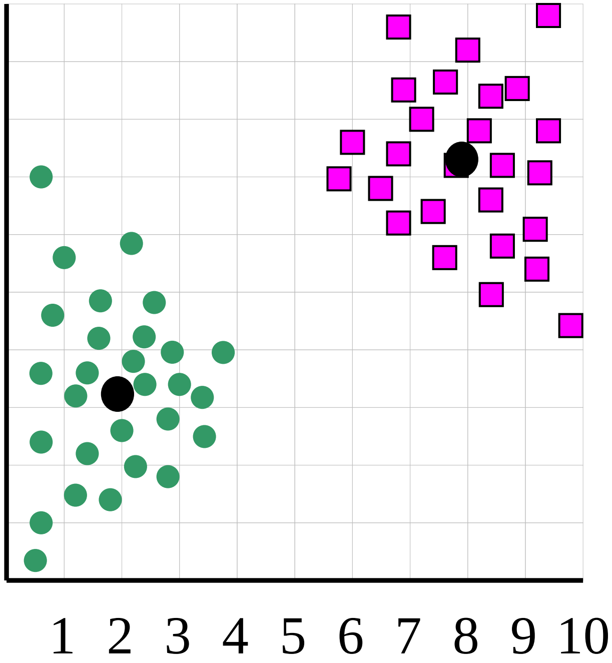
Weaknesses:

- need to specify *k* in advance which is often unknown
- find the best k by trying many different ones and picking the one with the lowest error
- often terminates at a *local optimum*
- the *global optimum* may be found by trying many times and using the best result

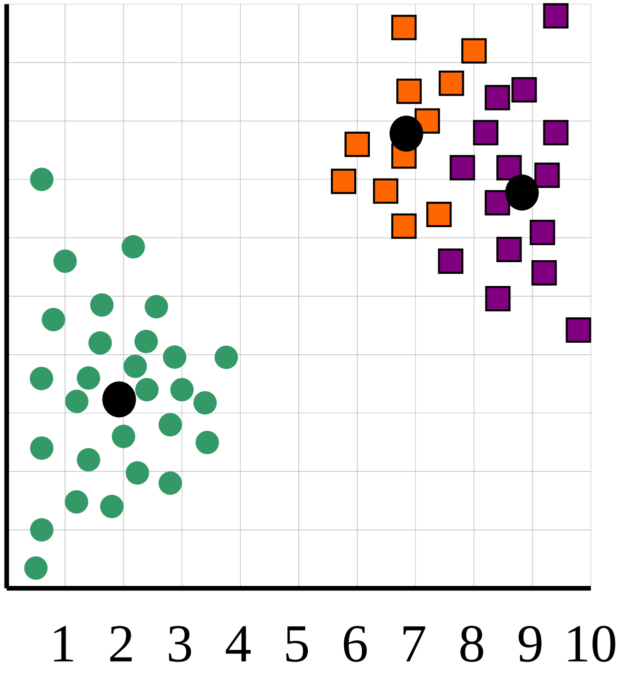
How Can We Find the Best K?



k=1, MSE=873.0



k=2, MSE=173.1



k=3, MSE=133.6

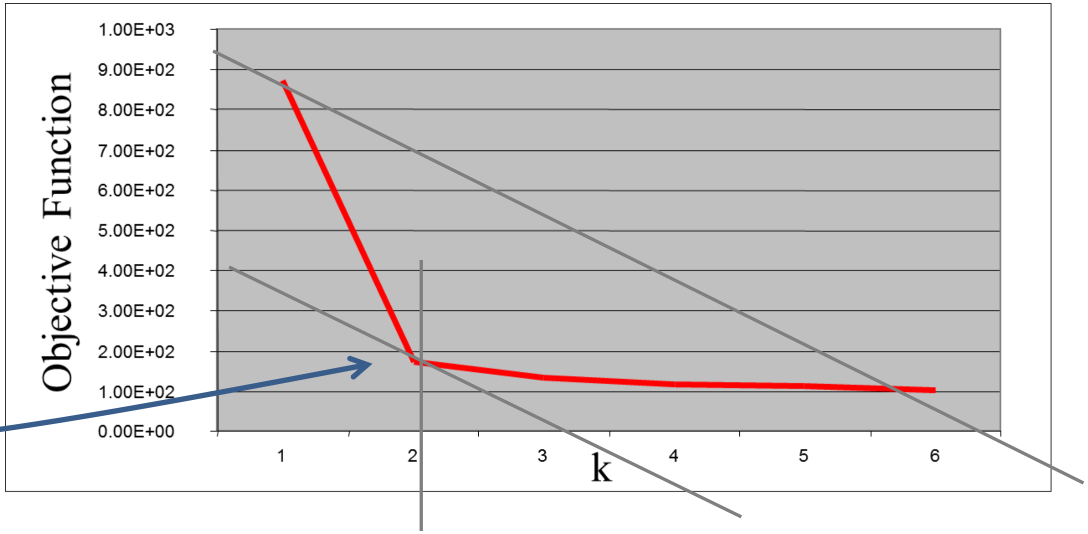
WE HAVE A WINNER

How About K=2?

Is there a principled way we can know when to stop looking?

Yes…

- we can plot the objective function values for k equals 1 to 6…
- then check for a flattening of the curve



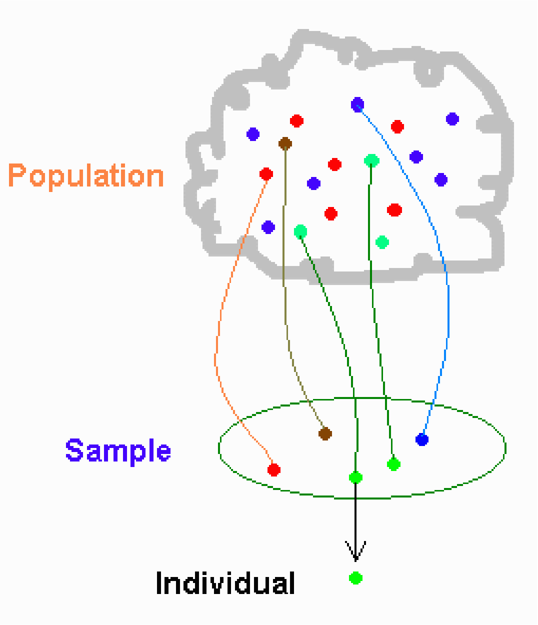
tangent at k=2

- the abrupt change at k = 2 is highly suggestive of two clusters
- this technique is known as “knee finding” or “elbow finding”

Back to Data reduction

What is sampling?

- pick a <u>representative</u> subset of the data
- discard the remaining data
- pick as many you can afford to keep
- recall: once it's gone, it's gone
- be smart about it



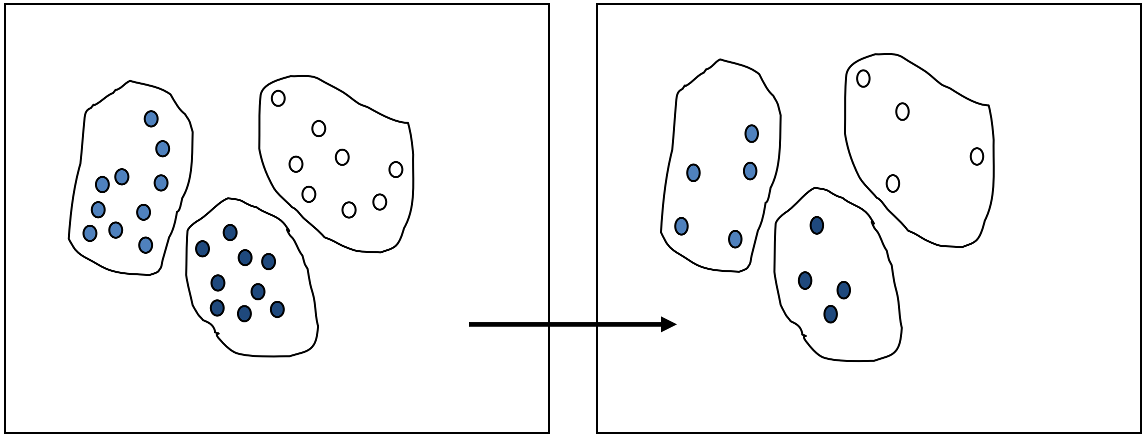
Simplest: random sampling

- pick sample points at random
- will work if the points are distributed uniformly
- this is usually not the case
- outliers will likely be missed
- so the sample will not be representative

Better: Adaptive Sampling

Pick the samples according to some knowledge of the data distribution

- cluster the data (outliers will form clusters as well)
- these clusters are also called *strata* (hence, stratified sampling)
- the size of each cluster represents its percentage in the population
- guides the number of samples – bigger clusters get more samples



sampling rate ~ cluster size

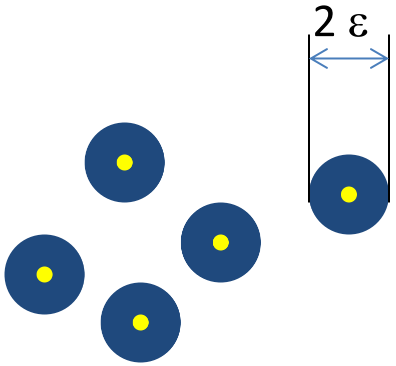
Redundancy Sampling

Eliminate redundant attributes

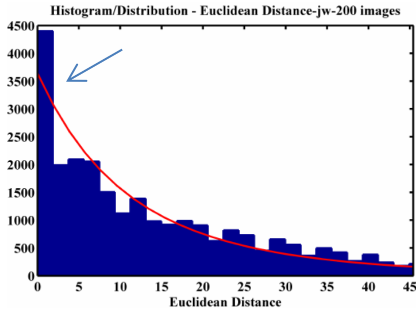
- eliminate highly correlated attributes
  - km vs. miles
  - a + b + c = d → can possibly eliminate ‘c’ (or ‘a’ or ‘b’)

Eliminate redundant data

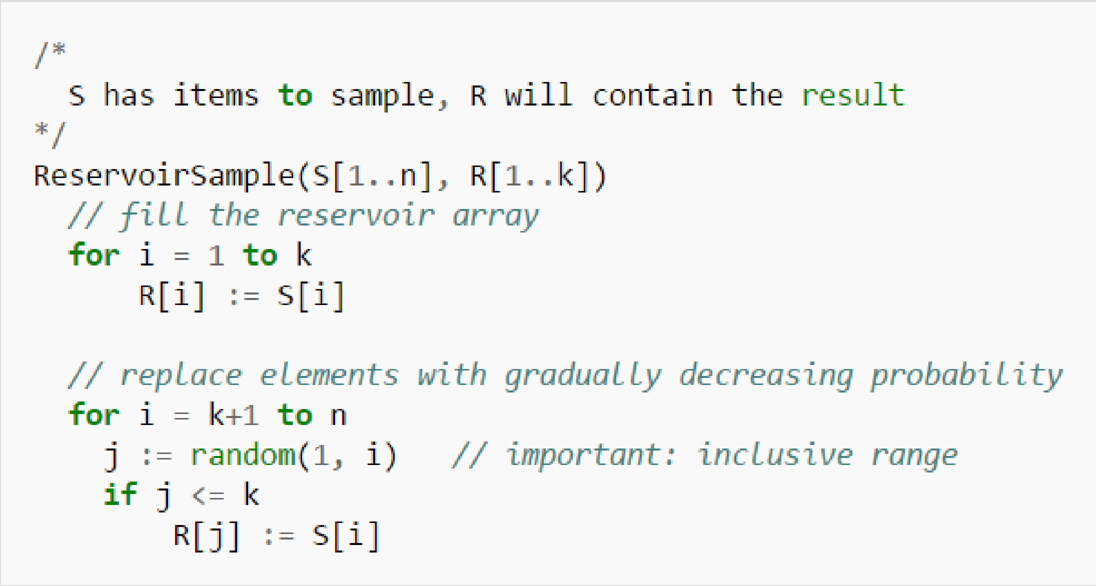
- cluster the data with small ranges ε
- only keep the cluster centroids
- store size of clusters along to keep importance



- question: how do we find a good ε?
- answer: compute histogram of distances
- choose a reasonable threshold from the left

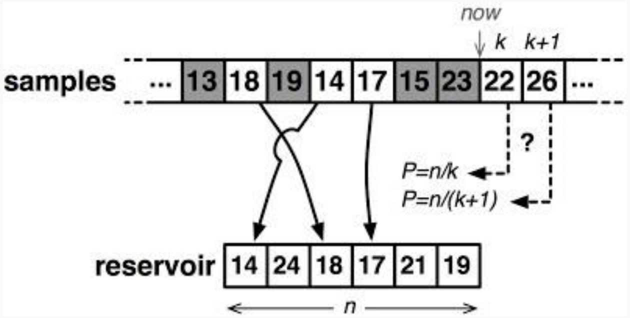


Reservoir Sampling



```
/*
  S has items to sample, R will contain the result
*/
ReservoirSample(S[1..n], R[1..k])
  // fill the reservoir array
  for i = 1 to k
      R[i] := S[i]

  // replace elements with gradually decreasing probability
  for i = k+1 to n
    j := random(1, i)   // important: inclusive range
    if j <= k
        R[j] := S[i]
```



Probabilities

- *k*/*i* for the *i*th sample to go into the reservoir
- 1/*k* · *k*/*i* = 1/*i* for the *j*th reservoir element to be replaced
- *k*/*n* for all elements in the reservoir after *n* has been reached
- can be shown via induction

A good algorithm to use for streaming data when *n* is growing

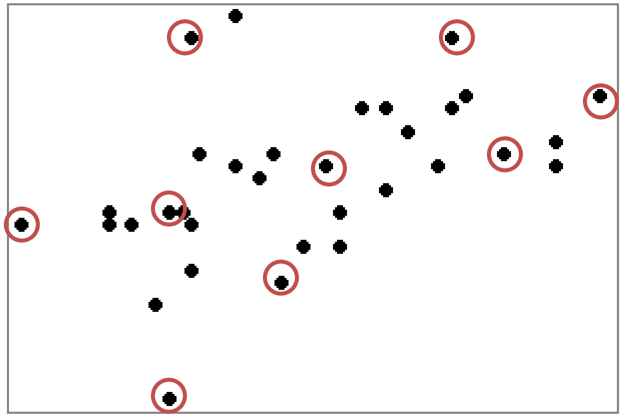
Sampling of Well-Scattered Points

Used in the CURE high-dimensional clustering algorithm

- S. Guha, R. Rajeev, and K. Shim. "CURE: an efficient clustering algorithm for large databases." *ACM SIGMOD,* 27(2): 73-84, 1998

Algorithm

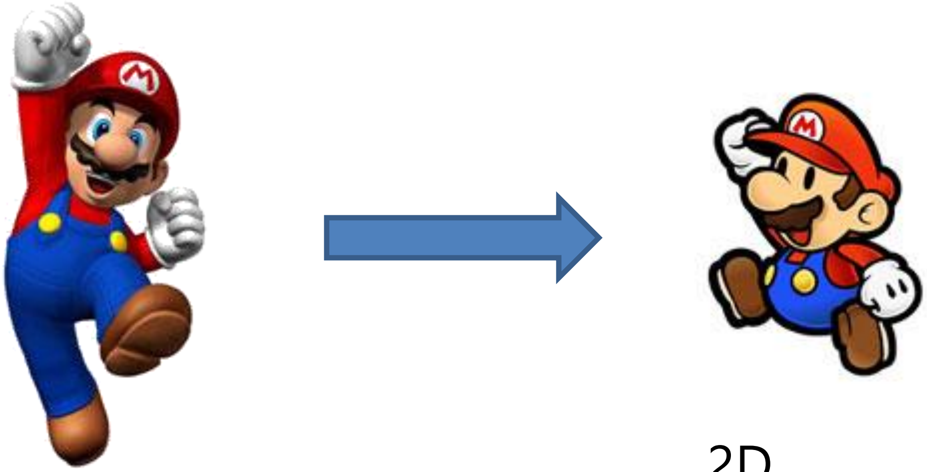
- initialize the point set *S* to empty
- pick the point farthest from the mean as the first point for *S*
- then iteratively pick points that are furthest from the points in S collected so far



Complexity is O($m \cdot n^2$)

- *n* is the total number of points, *m* is the number of desired points
- can find arbitrarily shaped clusters and preserve outliers, too
- need some good data structures to run efficiently: kd-tree, heap

Next Theme



3D

2D

Dimension Reduction

Measure of Attribute Similarity

Are there attributes that “go together”?

Can you name a few?

Feature Vector (1)

Physical attributes

- color
- number of doors
- number of wheels
- retractable roof
- height
- length
- frames around side windows

Which attributes are useful to distinguish SUVs from convertibles?

- number of doors (4 vs. 2) --> numerical, two levels
- retractable roof (no vs. yes) --> categorical, two levels
- frames around side windows (yes vs. no) --> categorical, two levels
- height (higher vs. lower) --> numerical, many levels

Feature Vector (2)

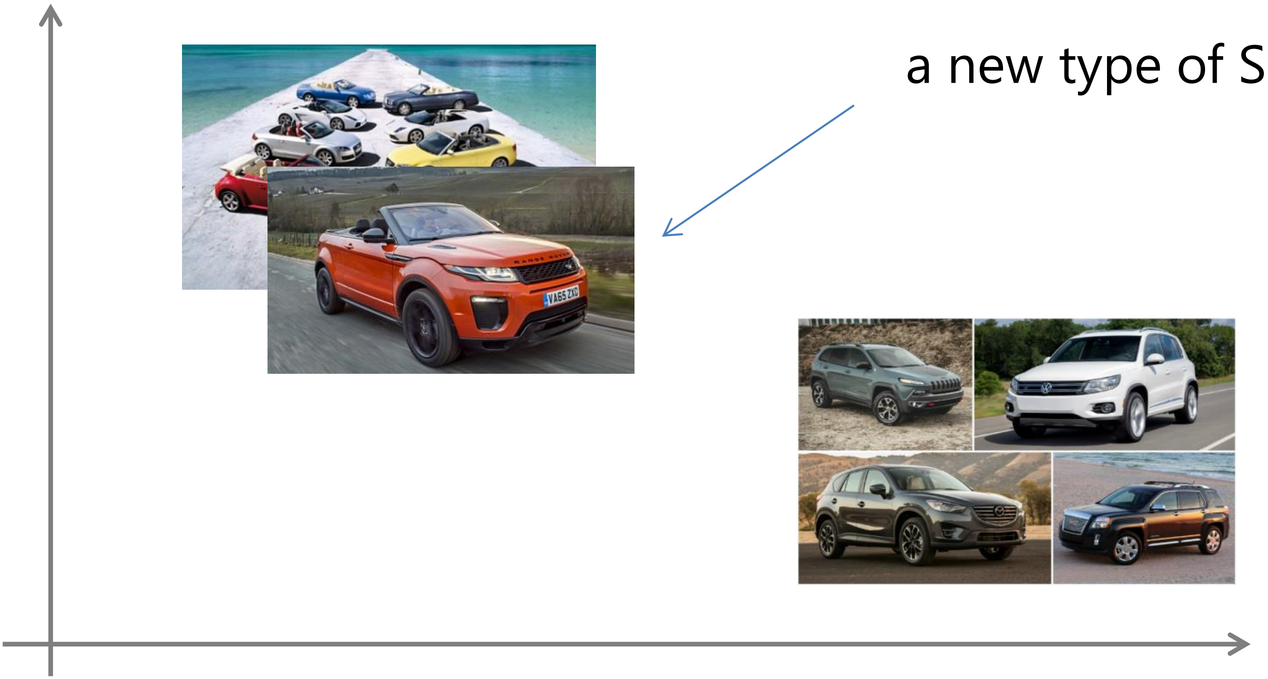
Which attributes are not so useful?

- number of wheels (constant 4) --> no discriminative power
- length (short and long SUVs, convertibles) --> confounding
- color (colors are seemingly random, or are they?)

Is color useful?

- the convertibles seem to have more vibrant colors (red, yellow, ...)
- so maybe we made a discovery

Attribute Space



retractable roof

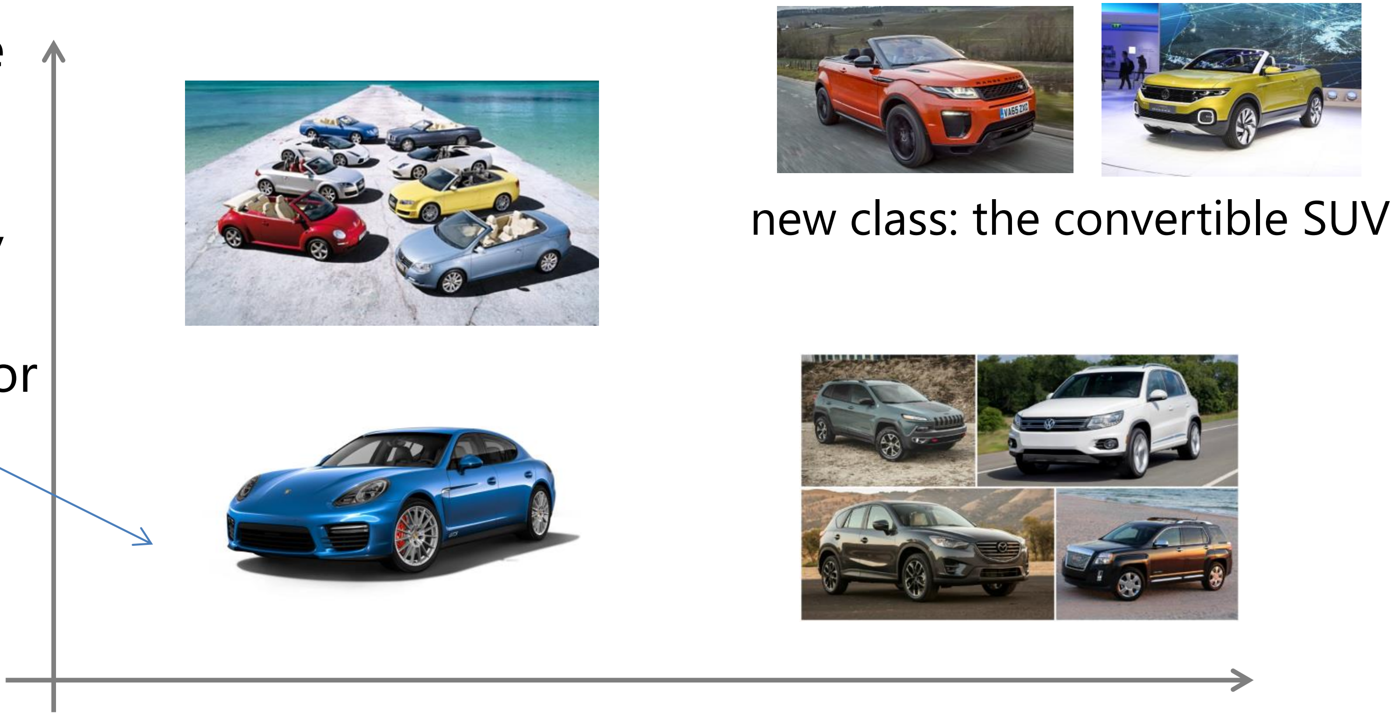
a new type of SUV

frames around side windows

Need to consider more than two attributes

- *height* attribute would have distinguished the Range Rover from the convertibles and caused it to be an outlier

Attribute Space



retractable roof

why can empty feature spaces be interesting or useful?

new class: the convertible SUV

height

New classes are constantly evolving over time

- this is known as *cluster evolution*
- measuring more features will increase the chance of discovery

How Many Data Do We Need?

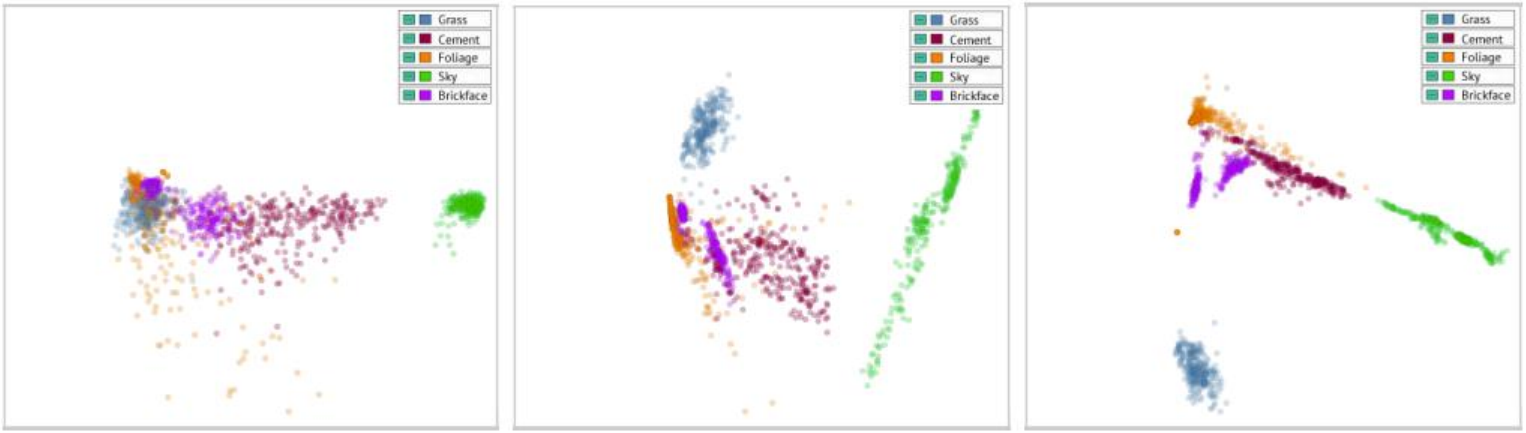
The more data (examples) the better

- increases the chances to discover the rare specimen

- but some attributes are useless
- we can cull them away
- perform attribute reduction or *dimension reduction*

How Many Attributes Do We Need?

Too many attributes can lead to obliteration of data patterns



(a) Full space (b) Subspace (c) Extended subspace

PCA projections of the Image Segmentation dataset generated from

- (a) the full 16D dataspace comprised of all feature dimensions
- (b) the 3D Raw Color semantic subspace
- (c) the 5D extended Raw Color semantic subspace.

The points are colored by their image class

Only (b) and (c) can separate the image classes well

Dimensionality Reduction

By axis rotation

- determine a more efficient basis
- Principal Component Analysis (PCA)
- Singular value decomposition (SVD)
- Latent semantic analysis (LSA)

By type transformation

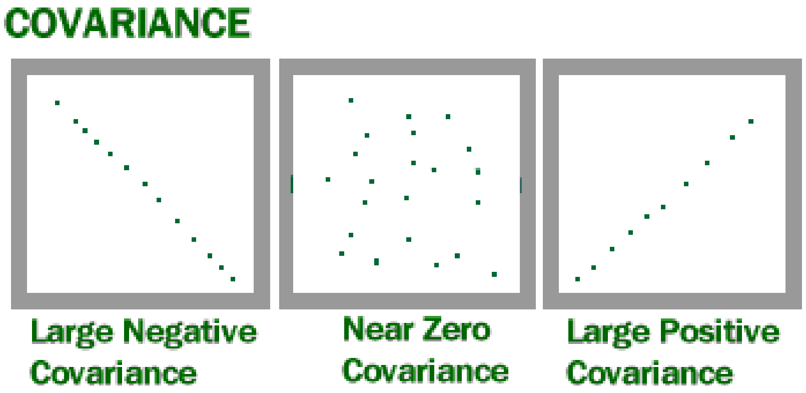
- determine a more efficient data type
- Fourier analysis and Wavelets for grids
- Multidimensional scaling (MSD) for graphs
- Locally Linear Embedding
- Isomap
- Self Organizing Maps (SOM)
- Linear Discriminant Analysis (LDA)

Principal Component Analysis (PCA)

Some Theory is Needed

Covariance

- measures how much two random variables change together



For N variable we have N^2 variable pairs

- we can write them in a matrix of size N^2 → the *covariance matrix*
- for two variables X_1 and X_2

$$\mathrm{Var}[X] = \begin{bmatrix} \mathrm{Var}[X_1] & \mathrm{Cov}[X_1, X_2] \\ \mathrm{Cov}[X_2, X_1] & \mathrm{Var}[X_2] \end{bmatrix}$$

Formulae

Covariance cov(X,Y)

mean of all data item values x_i and y_i for attributes X and Y, resp.

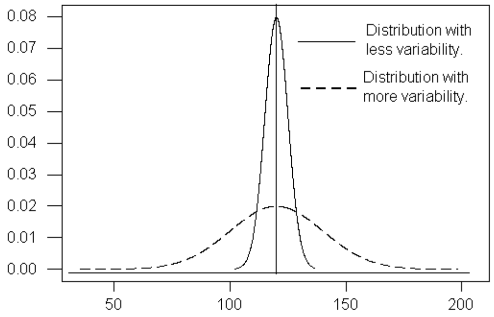
$$COV(X,Y) = \frac{\sum_{i=1}^{n}\left(X_i - \bar{X}\right)\left(Y_i - \bar{Y}\right)}{n-1}$$

Pearson's correlation r

- is covariance normalized by the individual variances for X and Y

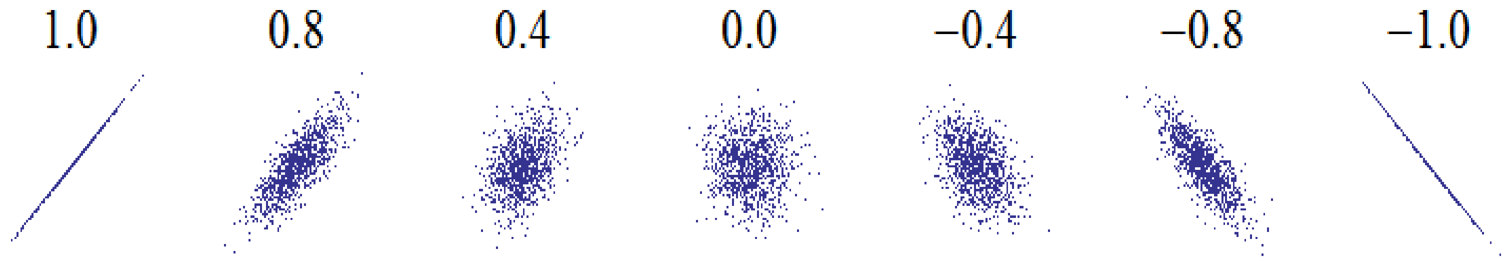
$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{x})^2}}$$

individual variances for attributes X and Y



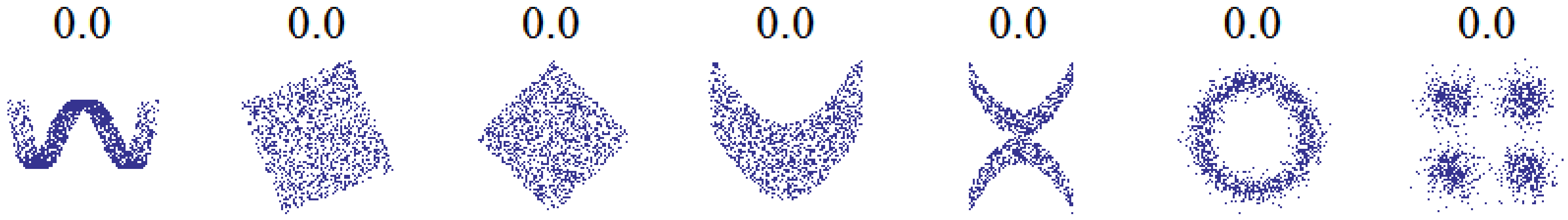
Correlation Patterns

Correlation rates between -1 and 1:



Important to note:

- correlation is defined for linear relationships
- visualization can help
- none of these point distributions have correlations:



Covariance Matrix

Analytical: $Cov(X,Y) = E[(X - \mu_x)(Y - \mu_y)]$

Samples: $\sigma_{xy} = \mathrm{cov}_{xy} = \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$

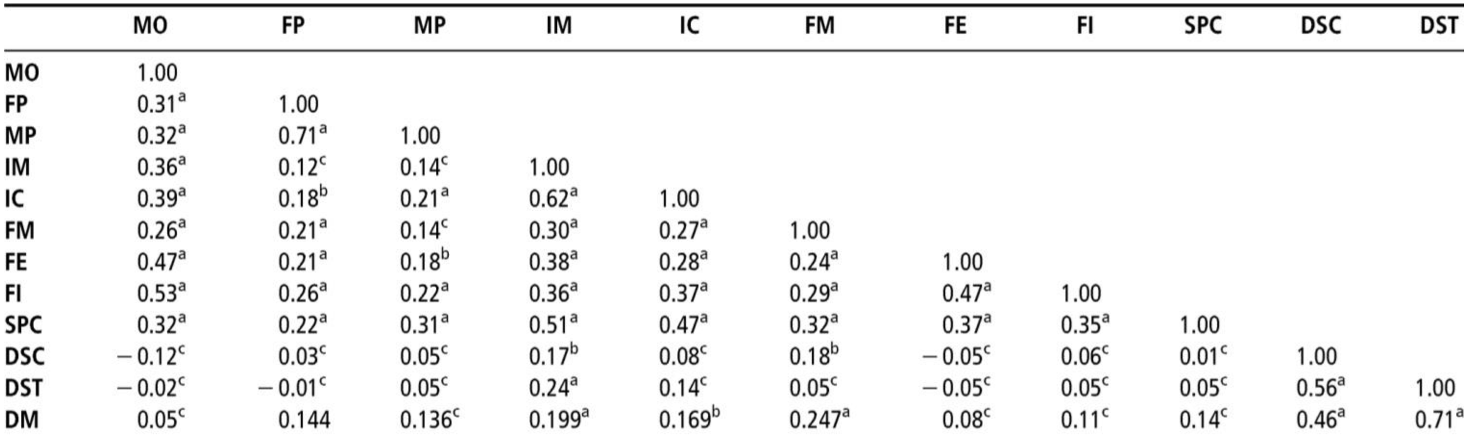
An n-D dataset has *n* variables x_1, x_2, … x_n

- define pairwise covariance among all of these variables
- construct a covariance matrix

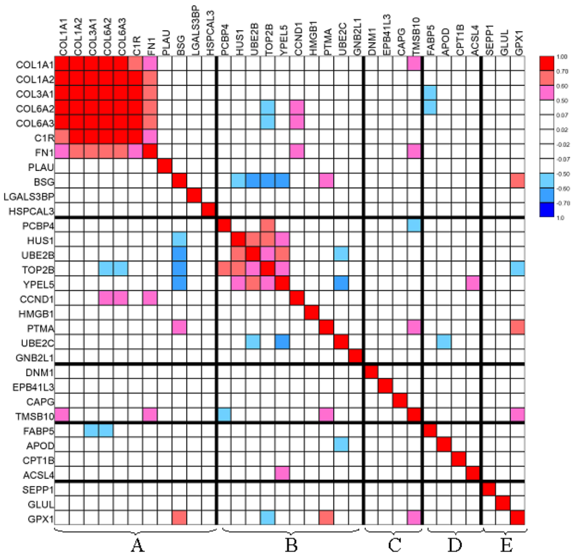
$$\Sigma = \mathrm{Cov}(X) = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_{nn} \end{bmatrix}$$

- a correlation matrix would just list the correlations instead

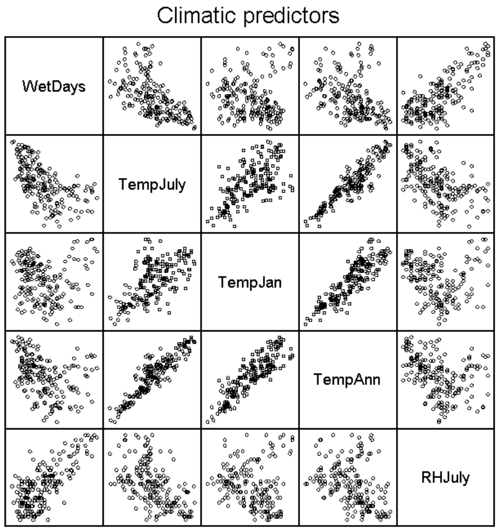
Correlation Matrix



	MO	FP	MP	IM	IC	FM	FE	FI	SPC	DSC	DST
MO	1.00										
FP	0.31[a]	1.00									
MP	0.32[a]	0.71[a]	1.00								
IM	0.36[a]	0.12[c]	0.14[c]	1.00							
IC	0.39[a]	0.18[b]	0.21[a]	0.62[a]	1.00						
FM	0.26[a]	0.21[a]	0.14[c]	0.30[a]	0.27[a]	1.00					
FE	0.47[a]	0.21[a]	0.18[b]	0.38[a]	0.28[a]	0.24[a]	1.00				
FI	0.53[a]	0.26[a]	0.22[a]	0.36[a]	0.37[a]	0.29[a]	0.47[a]	1.00			
SPC	0.32[a]	0.22[a]	0.31[a]	0.51[a]	0.47[a]	0.32[a]	0.37[a]	0.35[a]	1.00		
DSC	− 0.12[c]	0.03[c]	0.05[c]	0.17[b]	0.08[c]	0.18[b]	− 0.05[c]	0.06[c]	0.01[c]	1.00	
DST	− 0.02[c]	− 0.01[c]	0.05[c]	0.24[a]	0.14[c]	0.05[c]	− 0.05[c]	0.05[c]	0.05[c]	0.56[a]	1.00
DM	0.05[c]	0.144	0.136[c]	0.199[a]	0.169[b]	0.247[a]	0.08[c]	0.11[c]	0.14[c]	0.46[a]	0.71[a]



just value

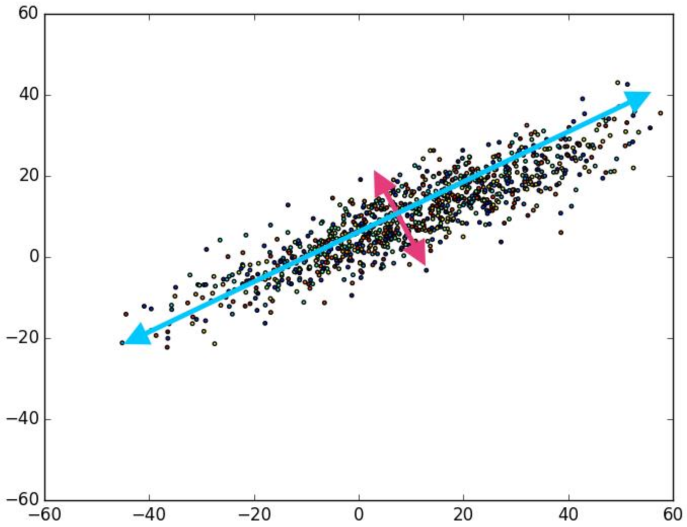


distribution (scatterplot matrix)

Principal Component Analysis

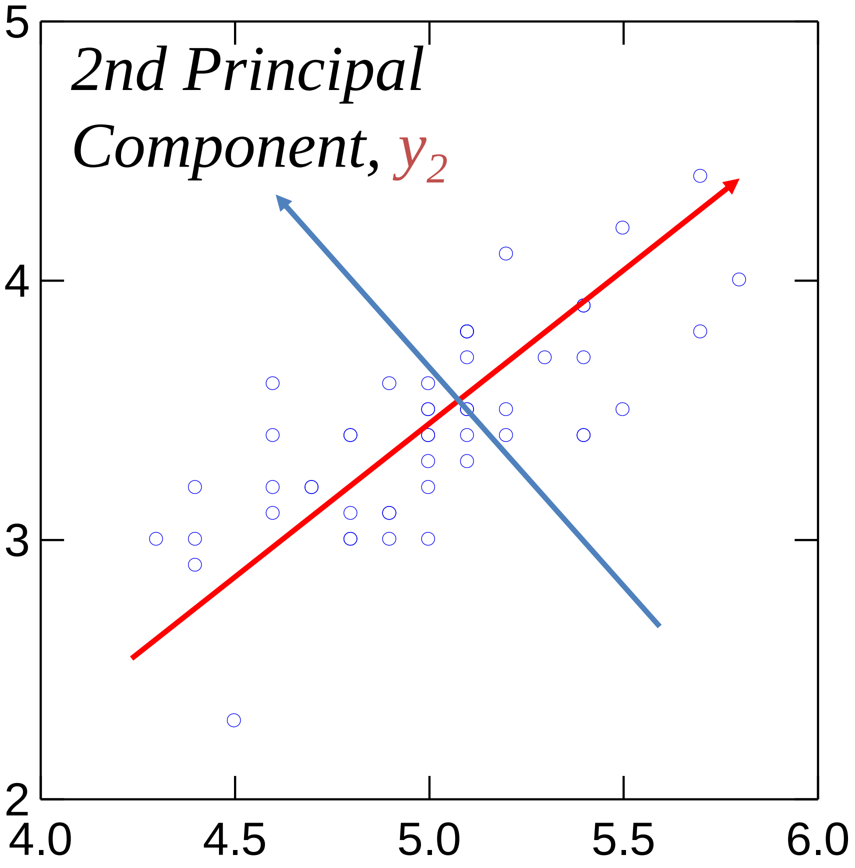
Ultimate goal:

- find a coordinate system that can represent the variance in the data with as few axes as possible



- rank these axes by the amount of variance (blue, red)
- drop the axes that have the least variance (red)

Principal Components



1st Principal Component, y_1

PCA – How To Do

Find the principal components (factors) of a distribution

First characterize the distribution by

- covariance matrix Cov
- correlation matrix Corr
- lets call it C

- perform QR factorization or LU decomposition on that matrix to get

$$\boldsymbol{C} = Q\Lambda Q^{-1}$$

Q: matrix with Eigenvectors

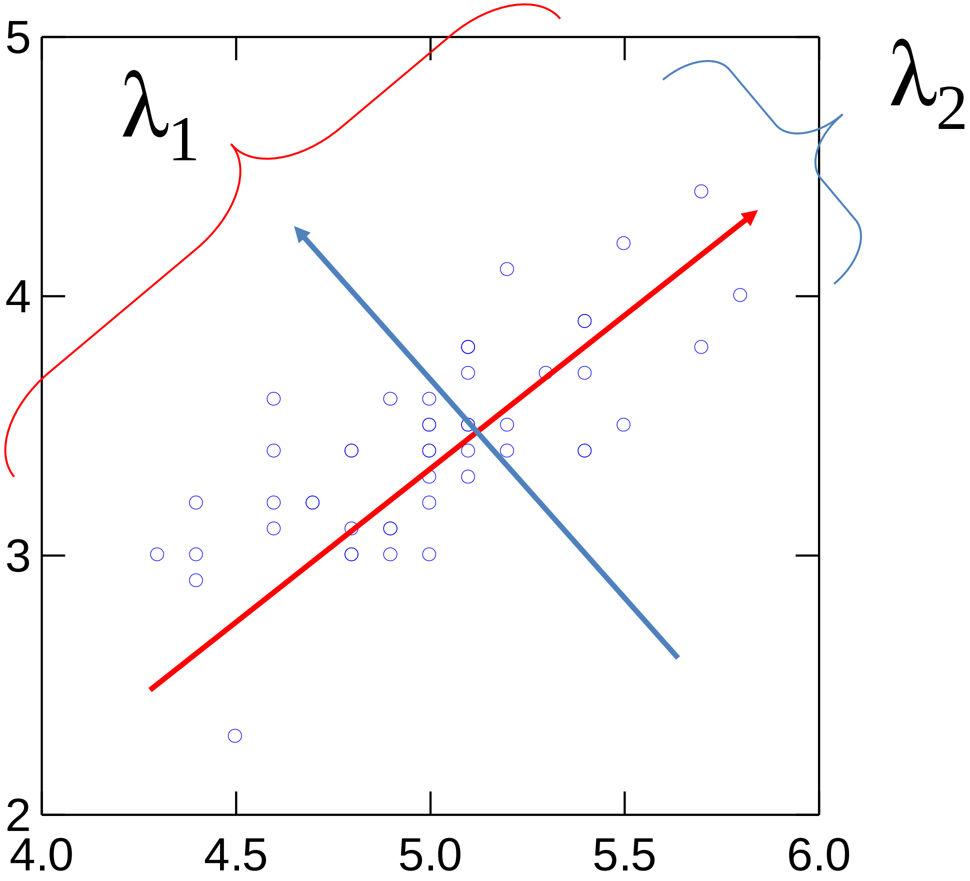
Λ: diagonal matrix with Eigenvalues λ

- now order the Eigenvectors in terms of their Eigenvalues λ

Eigenvectors and Eigenvalues

λ_1, λ_2 are the Eigenvalues

- encode the length (and therefore significance) of the Eigenvectors



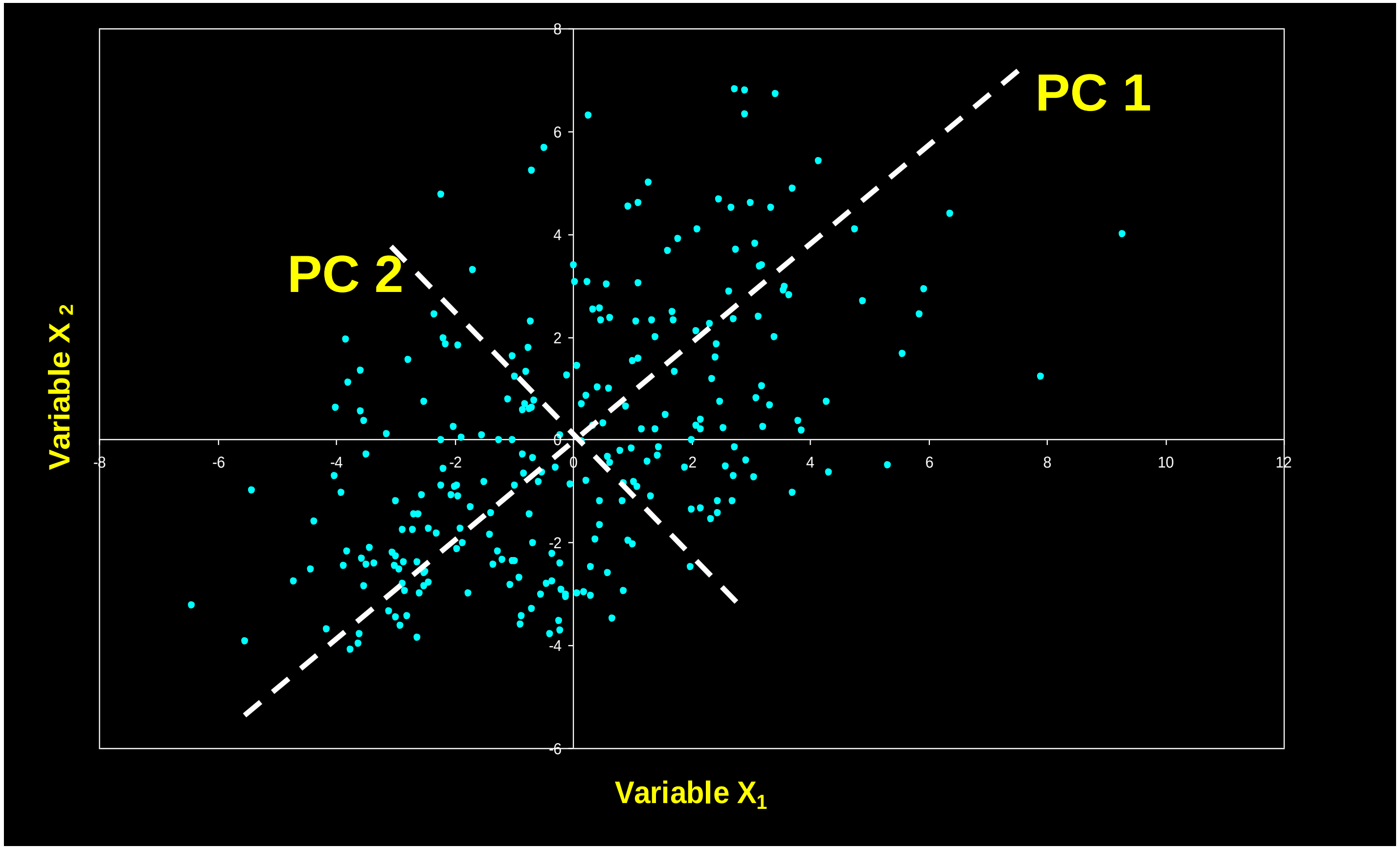
Covariance vs. Correlation

When to use what?

- use covariance matrix when the variable scales are similar
- use correlation matrix when the variables are on different scales
- the correlation matrix *standardizes* the data
- in general they give different results, especially when the scales are different

Example

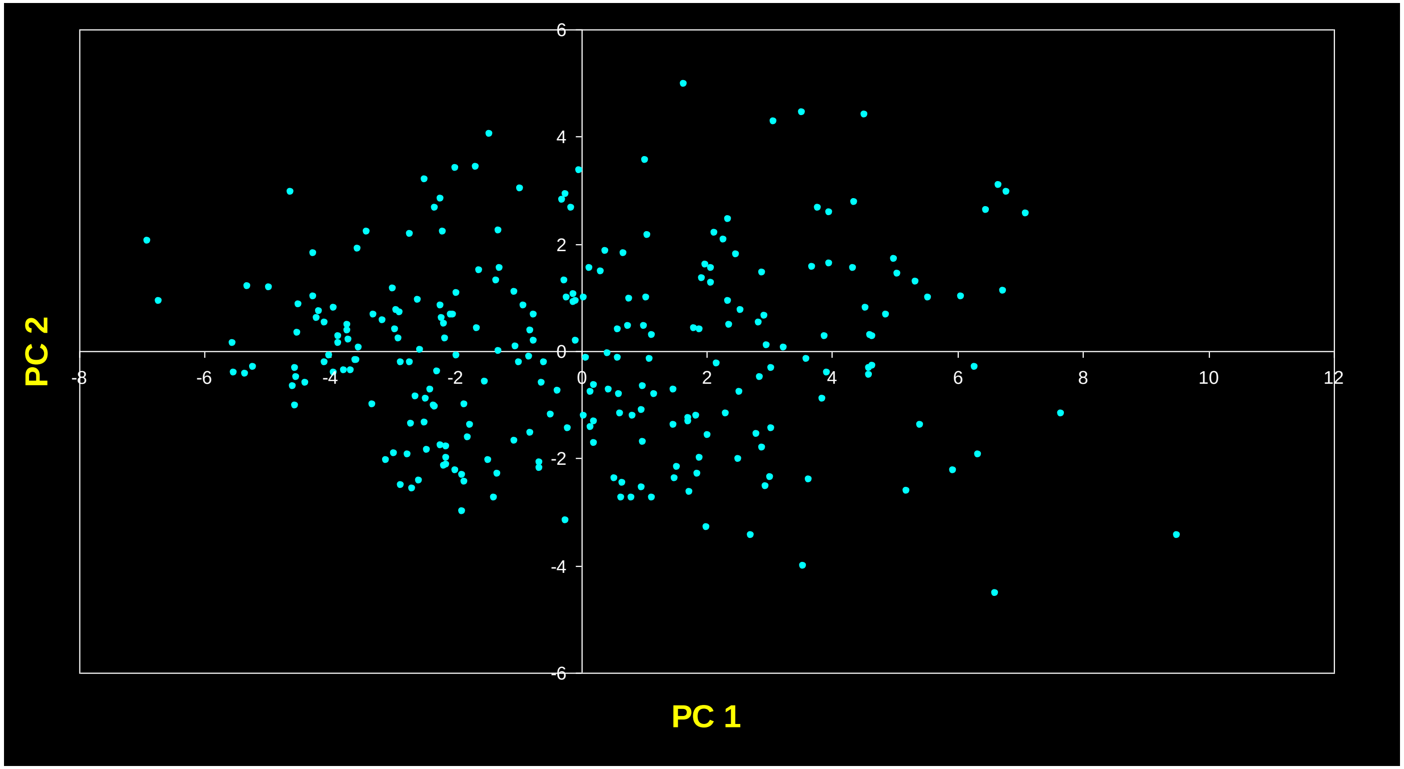
Before PCA



Example

After PCA

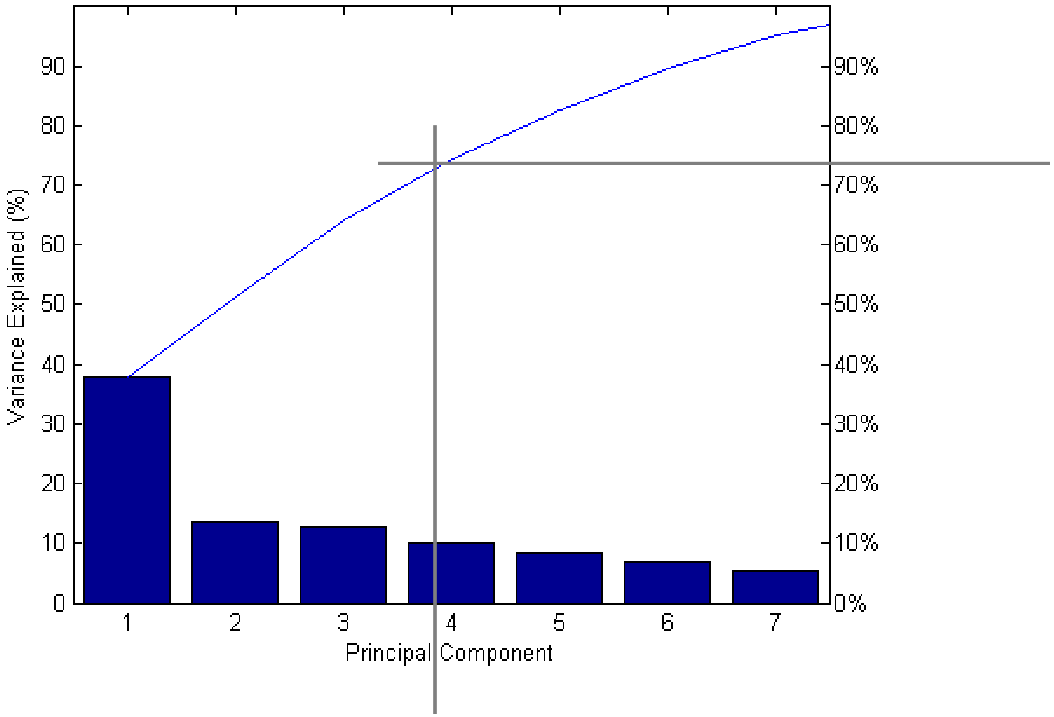
- λ_1 = 9.8783 λ_2 = 3.0308 Trace = 12.9091
- PC 1 displays ("explains") 9.8783/12.9091 = 76.5% of total variance



DIMENSION REDUCTION

Create a *scree plot*

- plots a histogram of the Eigenvalues ordered by magnitude
- plots the explained variance as a curve



possible threshold (explain 75% of data variance)

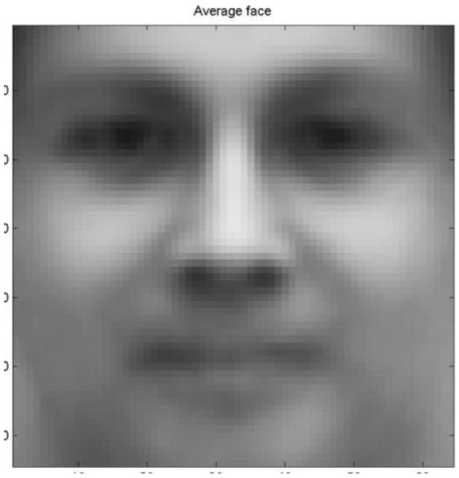
keep top 3 principal components → reduce dimensions by a factor of 4/7 = 57%

PCA Applied To Faces

Some familiar faces…

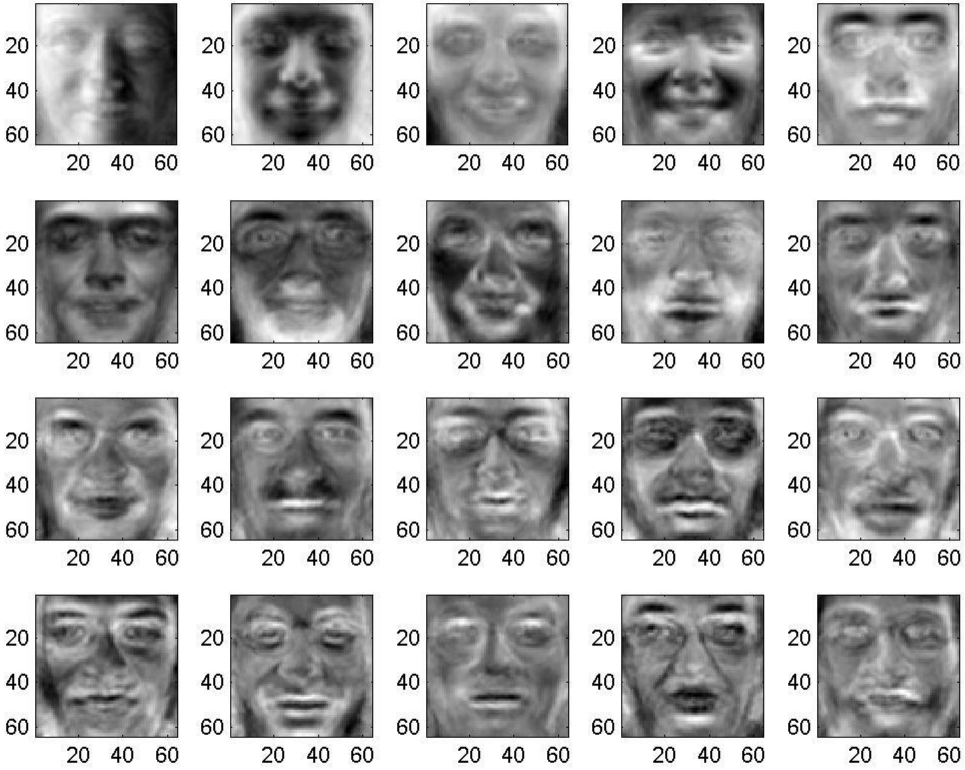
PCA Applied To Faces

We can reconstruct each face as a linear combination of "basis" faces, or Eigenfaces [M. Turk and A. Pentland (1991)]



Average Face

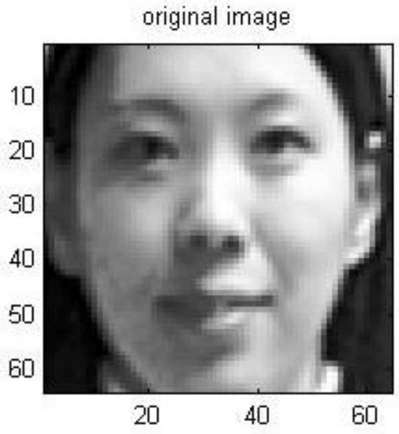
+

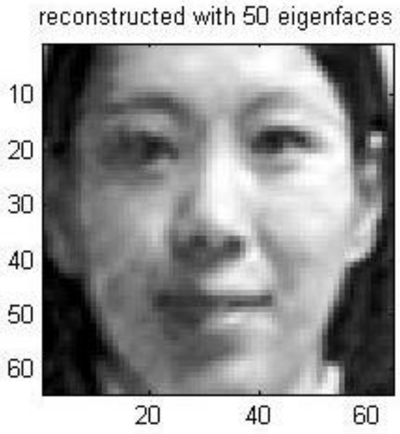


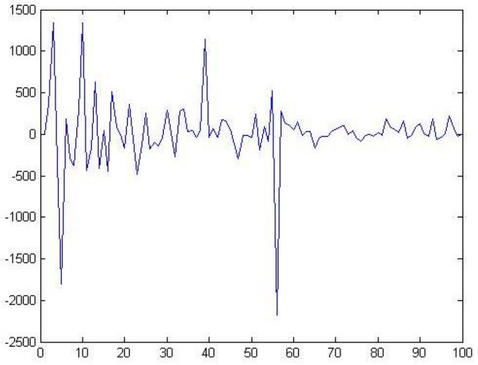
Eigenfaces

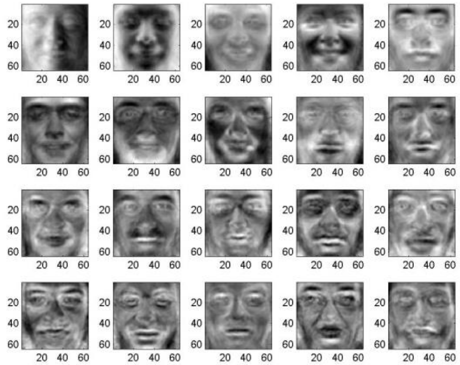
Reconstruction Using PCA

- 90% variance is captured by the first 50 eigenvectors
- Reconstruct existing faces using only 50 basis images
- We can also generate new faces by combining eigenvectors with different weights


original image


reconstructed with 50 eigenfaces





A More Challenging Example

- Data from research on habitat definition in the endangered Baw Baw frog
- 16 environmental and structural variables measured at each of 124 sites
- Correlation matrix used because variables have different units

Philoria frosti

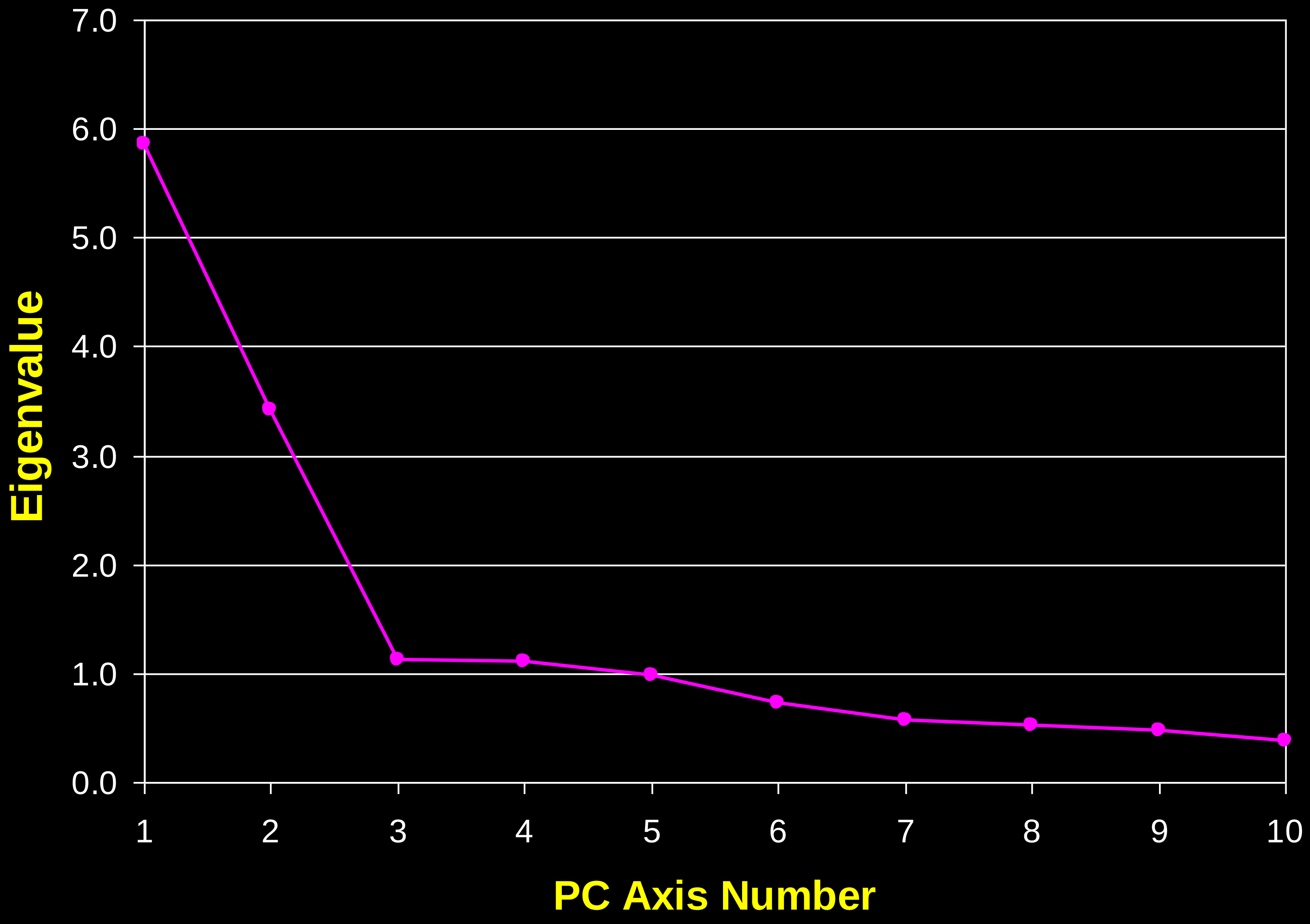
Eigenvalues

Axis	Eigenvalue	% of Variance	Cumulative % of Variance
1	5.855	36.60	36.60
2	3.420	21.38	57.97
3	1.122	7.01	64.98
4	1.116	6.97	71.95
5	0.982	6.14	78.09
6	0.725	4.53	82.62
7	0.563	3.52	86.14
8	0.529	3.31	89.45
9	0.476	2.98	92.42
10	0.375	2.35	94.77

How Many Axes Are Needed?

- Does the $(k+1)^{th}$ principal axis represent more variance than would be expected by chance?
- Several tests and rules have been proposed
- A common "rule of thumb" when PCA is based on correlations is that axes with eigenvalues > 1 are worth interpreting
- In our example 4 Eigenvectors fit this criterion (we shall keep 3 for simplicity)

Baw Baw Frog - PCA of 16 Habitat Variables



Interpreting Eigenvectors

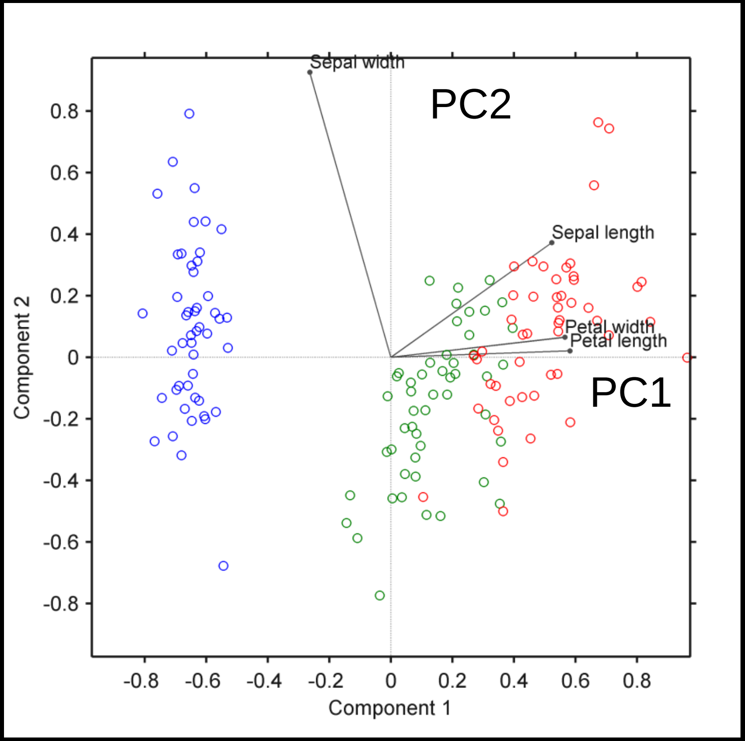
- Correlations between variables and the principal axes are known as loadings
- Each element of the eigenvectors represents the contribution of a given variable to a component
- The loadings of variables on the first three PCs are shown here

	PC 1	PC 2	PC 3
Altitude	0.3842	0.0659	-0.1177
pH	-0.1159	0.1696	-0.5578
Cond	-0.2729	-0.1200	0.3636
TempSurf	0.0538	-0.2800	0.2621
Relief	-0.0765	0.3855	-0.1462
maxERht	0.0248	0.4879	0.2426
avERht	0.0599	0.4568	0.2497
%ER	0.0789	0.4223	0.2278
%VEG	0.3305	-0.2087	-0.0276
%LIT	-0.3053	0.1226	0.1145
%LOG	-0.3144	0.0402	-0.1067
%W	-0.0886	-0.0654	-0.1171
H1Moss	0.1364	-0.1262	0.4761
DistSWH	-0.3787	0.0101	0.0042
DistSW	-0.3494	-0.1283	0.1166
DistMF	0.3899	0.0586	-0.0175

What's a "Loading"?

- The amount of weight a data dimension has on a principal component
  - – petal length/width have a high loading on PC1
  - – sepal width has a high loading on PC2

- Another observation
  - – projection into PC basis can also bring out clusters better
  - – since spread is maximized



Significance of Variables

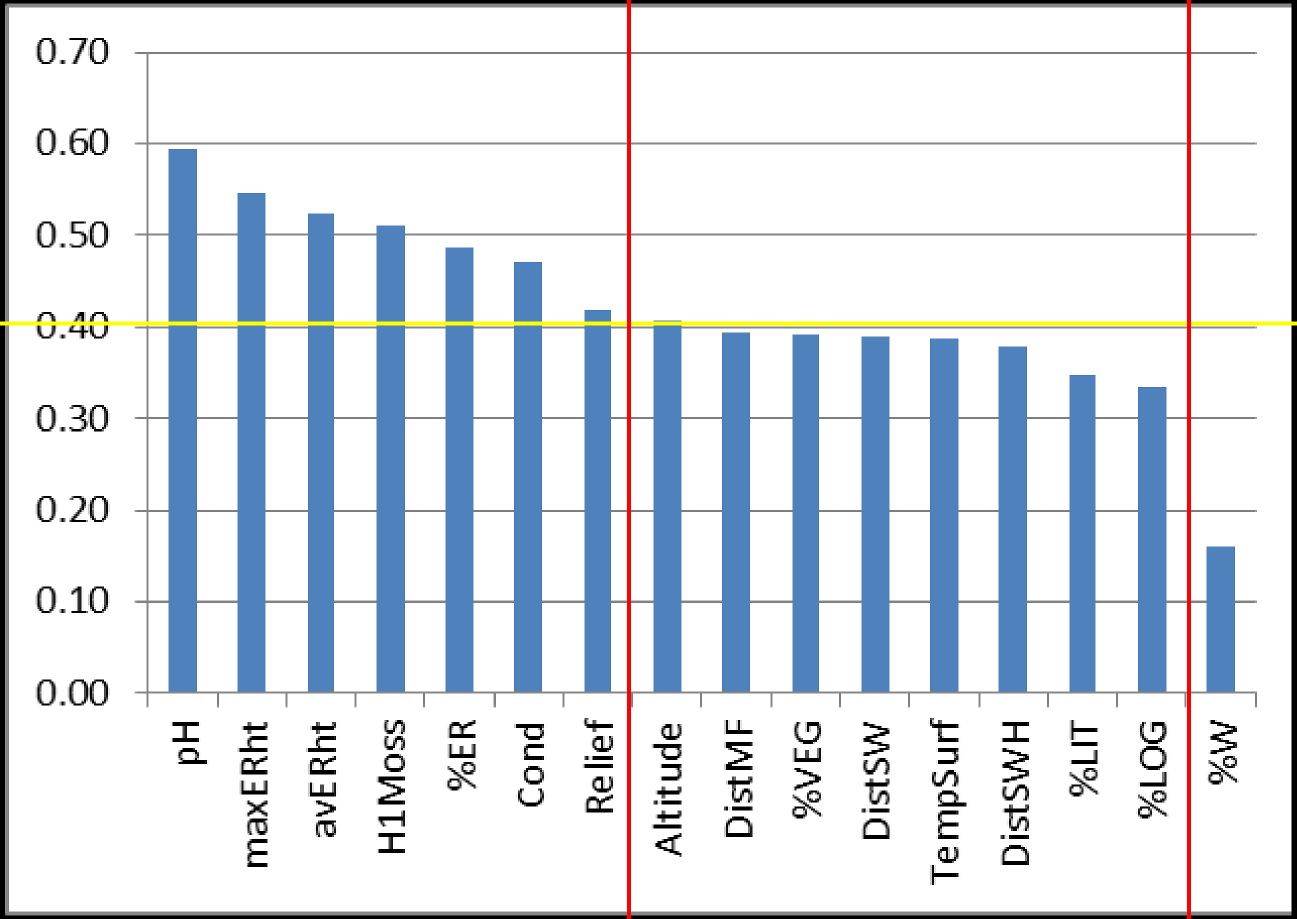
- We can compute the significance of the variables as the sum of squared loadings on to the most significant Eigenvectors we selected (3 in our example)
- The next slide shows the table of the last slide expanded with these squared loadings
- We can then sort the table by the squared loadings and make a scree plot
- The most significant variables are those above some chosen cutoff, for example 0.4 (marked in yellow in the table)

Significance of Variables

	PC 1	PC 2	PC 3	sum of squared loadings
Altitude	0.3842	0.0659	-0.1177	0.41
pH	-0.1159	0.1696	-0.5578	0.59
Cond	-0.2729	-0.1200	0.3636	0.47
TempSurf	0.0538	-0.2800	0.2621	0.39
Relief	-0.0765	0.3855	-0.1462	0.42
maxERht	0.0248	0.4879	0.2426	0.55
avERht	0.0599	0.4568	0.2497	0.52
%ER	0.0789	0.4223	0.2278	0.49
%VEG	0.3305	-0.2087	-0.0276	0.39
%LIT	-0.3053	0.1226	0.1145	0.35
%LOG	-0.3144	0.0402	-0.1067	0.33
%W	-0.0886	-0.0654	-0.1171	0.16
H1Moss	0.1364	-0.1262	0.4761	0.51
DistSWH	-0.3787	0.0101	0.0042	0.38
DistSW	-0.3494	-0.1283	0.1166	0.39
DistMF	0.3899	0.0586	-0.0175	0.39

Significance of Variables

- Scree plot



chosen significance threshold

variables considered significant

more aggressive reduction of variables

only eliminate very weak variables

Summary

Data reduction

- notions of similarity and distance in high-D data spaces
- clustering (k-means) and how to pick optimal k
- sampling

Dimension reduction

- important vs. irrelevant dimensions
- notion of principal components and Eigenvectors
- scree plots to visualize explained variance and threshold it
- principal component analysis (PCA)
- using PCA loadings to find most important data dimensions