CSE 564
Introduction to Visualization

Data Reduction & Similarity metrics

Klaus Mueller

Computer Science Department
Stony Brook University

Lecture	Topic	Projects
1	Intro, schedule, and logistics	
2	Applications of visual analytics, basic tasks, data types	
3	Introduction to D3, basic vis techniques for non-spatial data	Project #1 out
4	Data assimilation and preparation	
5	Data reduction and notion of similarity and distance	
6	Visual perception and cognition	
7	Visual design and aesthetics	Project #1 due
8	Statistics foundations	Project #2 out
9	Data mining techniques: clusters, text, patterns, classifiers	
10	Data mining techniques: clusters, text, patterns, classifiers	
11	Computer graphics and volume rendering	
12	Techniques to visualize spatial (3D) data	Project #2 due
13	Scientific and medical visualization	Project #3 out
14	Scientific and medical visualization	
15	Midterm #1	
16	High-dimensional data, dimensionality reduction	Project #3 due
17	Big data: data reduction, summarization	
18	Correlation and causal modeling	
19	Principles of interaction	
20	Visual analytics and the visual sense making process	Final project proposal due
21	Evaluation and user studies	
22	Visualization of time-varying and time-series data	
23	Visualization of streaming data	
24	Visualization of graph data	Final Project preliminary report due
25	Visualization of text data	
26	Midterm #2	
27	Data journalism	
	Final project presentations	Final Project slides and final report due

Today's Theme

Data Reduction

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

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)

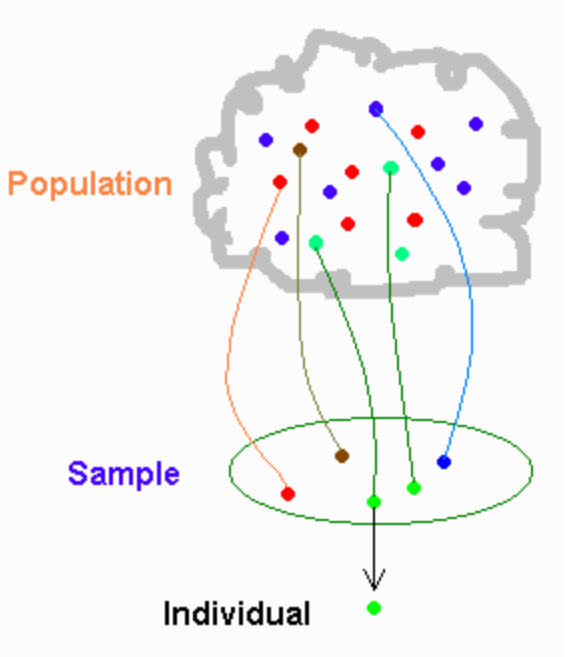
Sampling

The goal

- pick a representative subset of the data

Random sampling

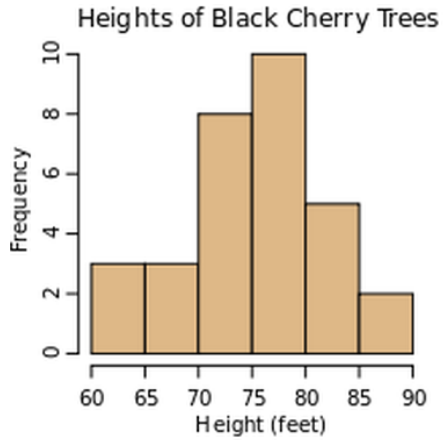
- pick sample points at random
- will work if the points are distributed uniformly
- this is usually not the case
- outliers will likely be missed
- so the sample will not be representative

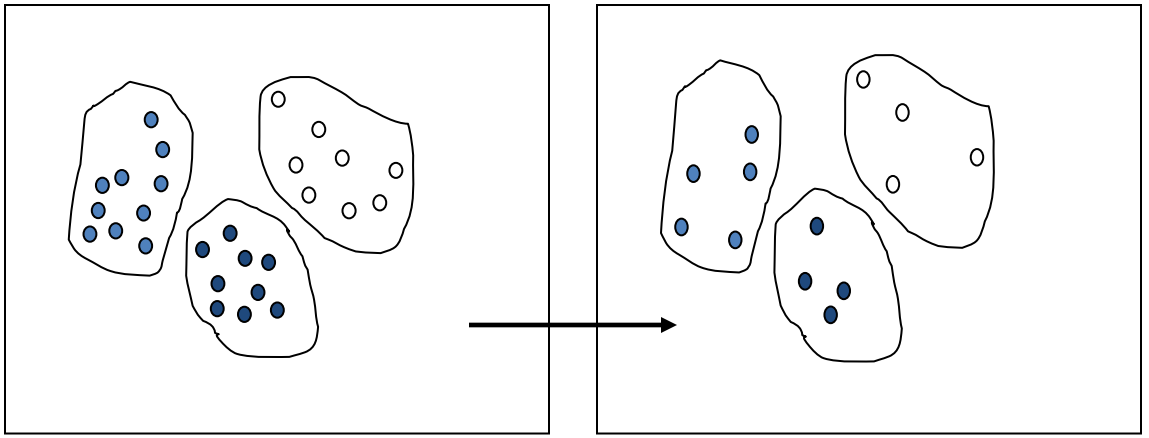


Adaptive Sampling

Pick the samples according to some knowledge of the data distribution

- create a binning of some sort (outliers will form bins as well)
- also called *strata* (stratified sampling)
- the size of each bin represents its percentage in the population
- it guides the number of samples – bigger bins get more samples





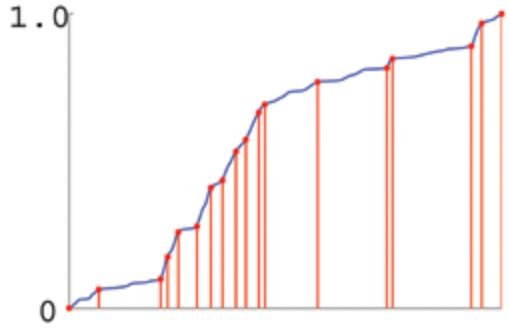
sampling rate ~ bin height

sampling rate ~ cluster size

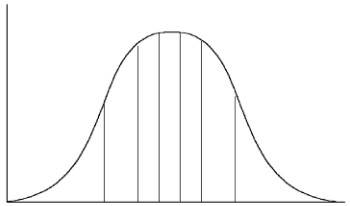
Importance Sampling

Estimate the statistical properties of a distribution

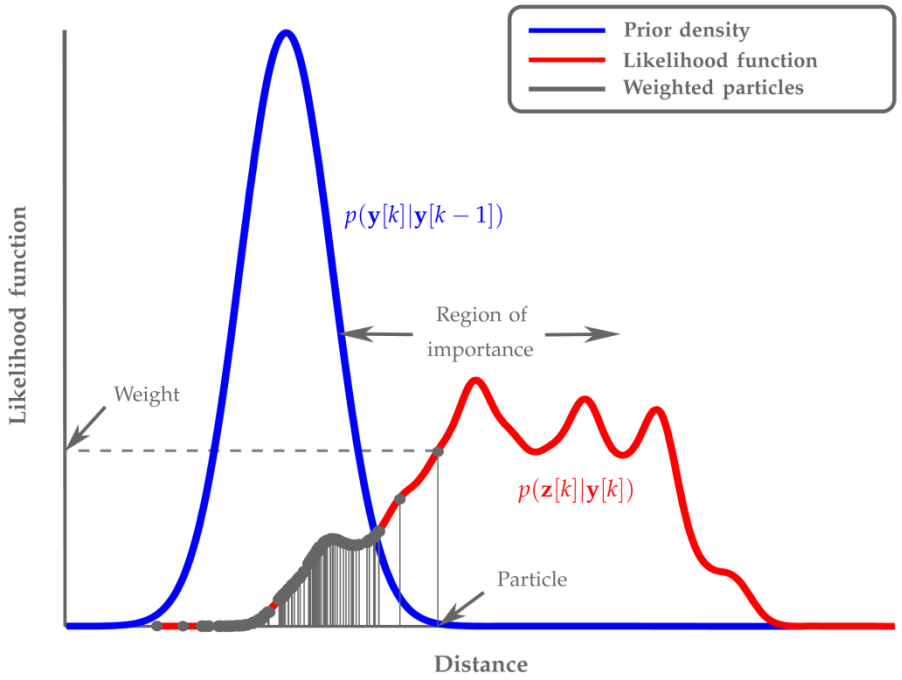
- then sample the distribution according to this distribution
- define the importance



sample in high slopes



sample in high densities



sample according to a user-defined function

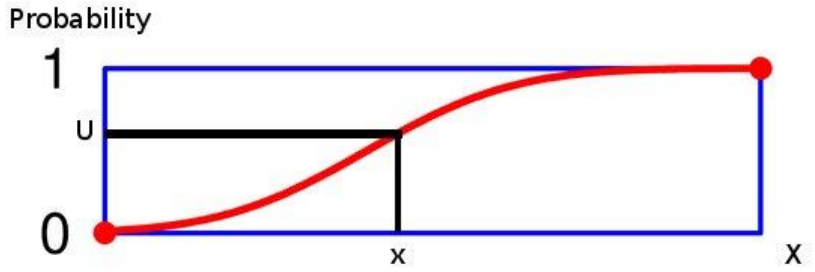
The Inversion Method

Easy way to making your own sampling algorithm

- find the cumulative distribution function (CDF) of your desired probability density function (PDF)

$$F(x) = \int_{-\infty}^{x} f(t)dt$$

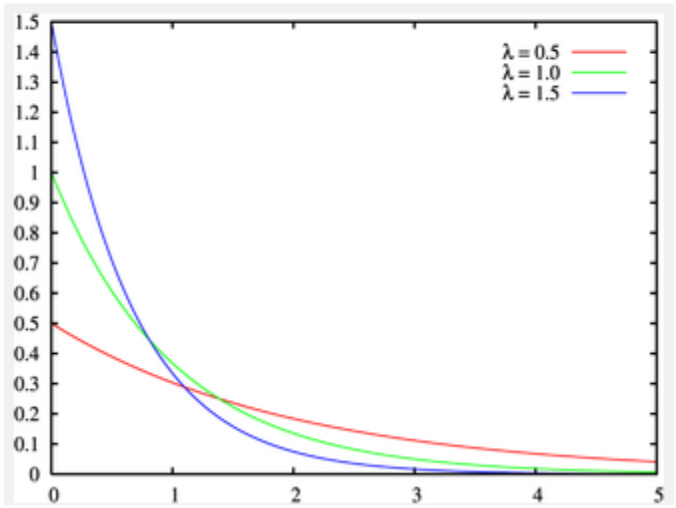
- if f(x) has an inverse then we can use the inversion method to create a sampling method



- generate a random u-value between [0.1] and look up the x-value
- region with higher f(x) have a steeper CDF and get sampled more

Inversion Method Example

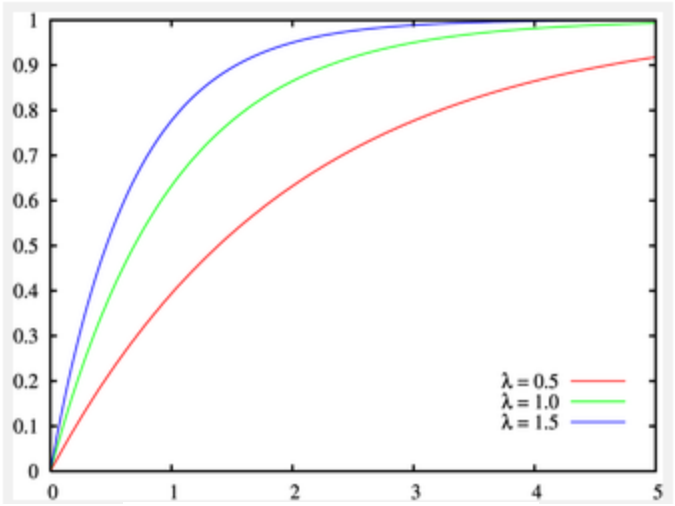
$$f(x) = \lambda e^{-\lambda x}$$



The cumulative probability function of this PDF is:

$$\int_{-\infty}^{x} f(t)dt = F(x) = 1 - e^{-\lambda x}$$

u



$$F^{-1}(u) = -\frac{\ln(u)}{\lambda}$$

Redundancy Sampling

Good candidates for elimination are *redundant* data

- how many cans of ravioli will you buy?

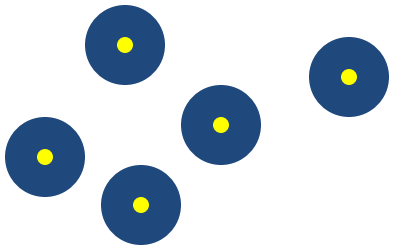
Redundancy Sampling

Eliminate redundant attributes

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

Eliminate redundant data

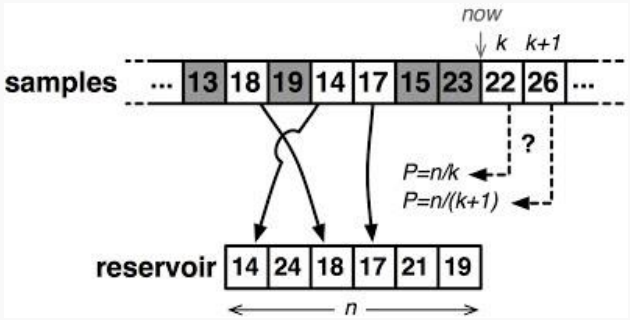
- cluster the data with small ranges
- only keep the cluster centroids
- store size of clusters along to keep importance



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
- $1/k \cdot k/i = 1/i$ 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

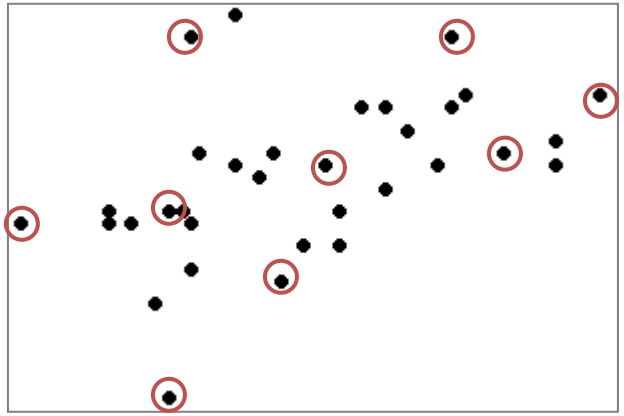
Sampling of Well-Scattered Points

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

What's When Your data Is Too Small

Can you "hallucinate" or "invent" realistic data?

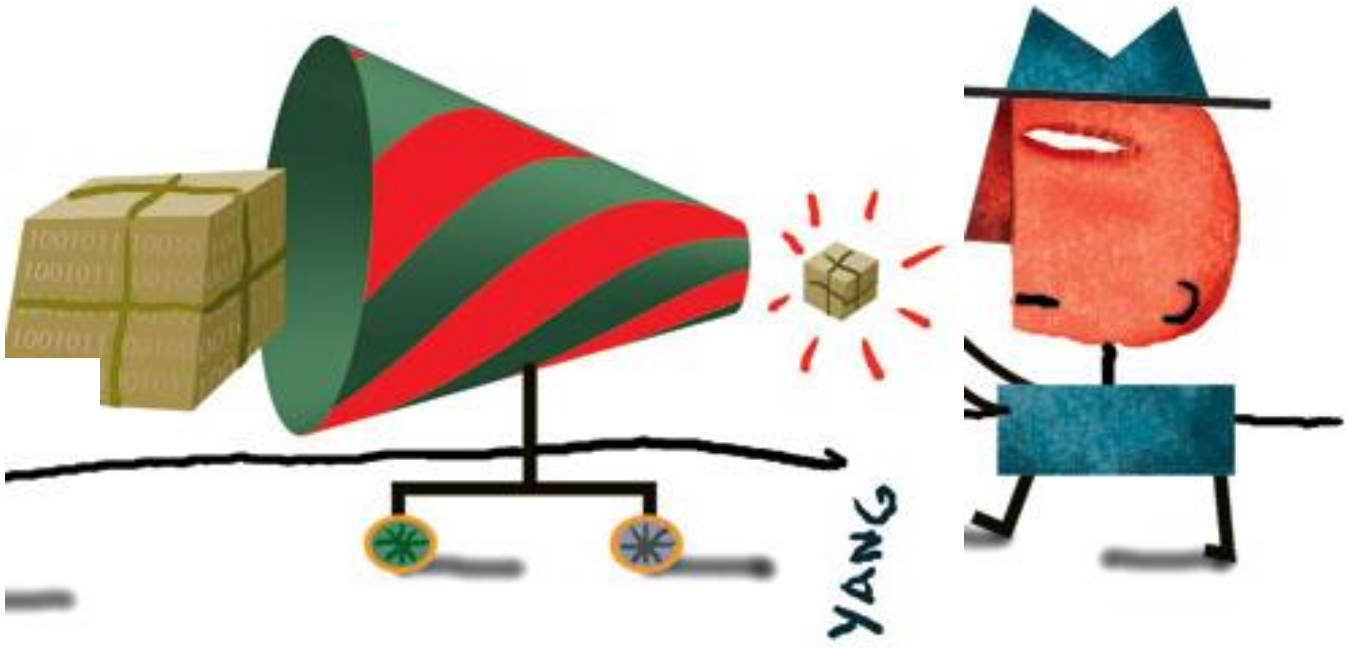
And if so, how would you go about this?

How to Hallucinate More data...

Data Augmentation

Strategy to artificially synthesize new data from existing data

- go from small data to big data



Data Augmentation in Machine learning

Important topic in deep learning

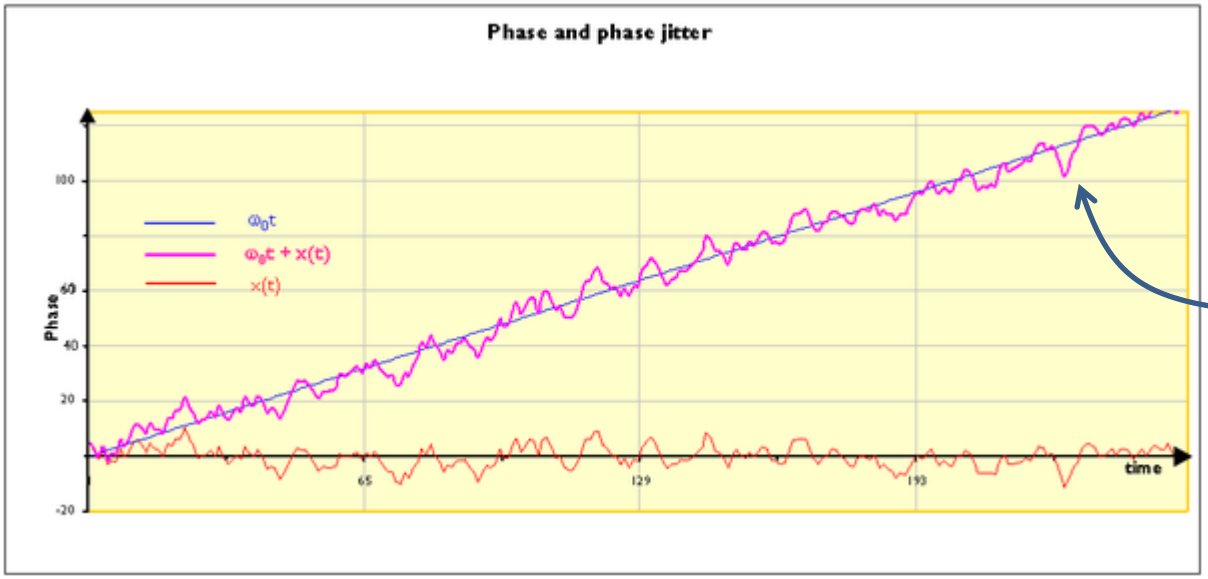
Common techniques are (for images)

- rotations
- translations
- zooms
- flips
- color perturbations
- crops
- add noise by *jittering*

What's Jittering?

Definition from dictionary

- act nervously
- "an anxious student who jittered at any provocation"



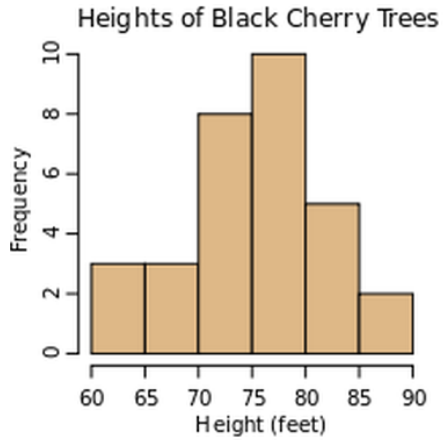
Jitter (noise)

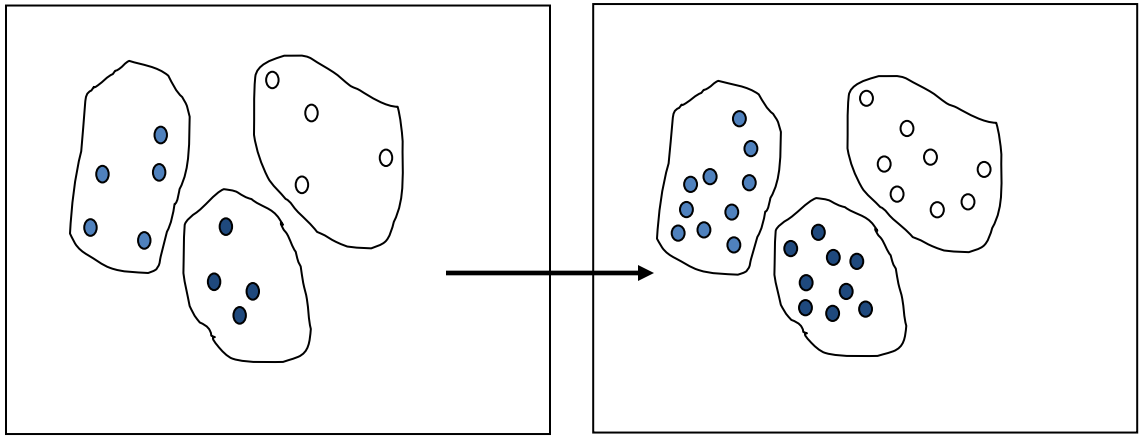
- small random noise about a steady signal

Data Augmentation for Visualization & Visual Analytics

Generate new samples according to the data distributions

- cluster the data (outliers will form clusters as well)
- the size of each cluster represents its percentage in the population
- randomize new samples – bigger clusters get more samples
- add a small randomized value to either the mean or an existing sample
- do this for every dimension of the chosen mean or sample

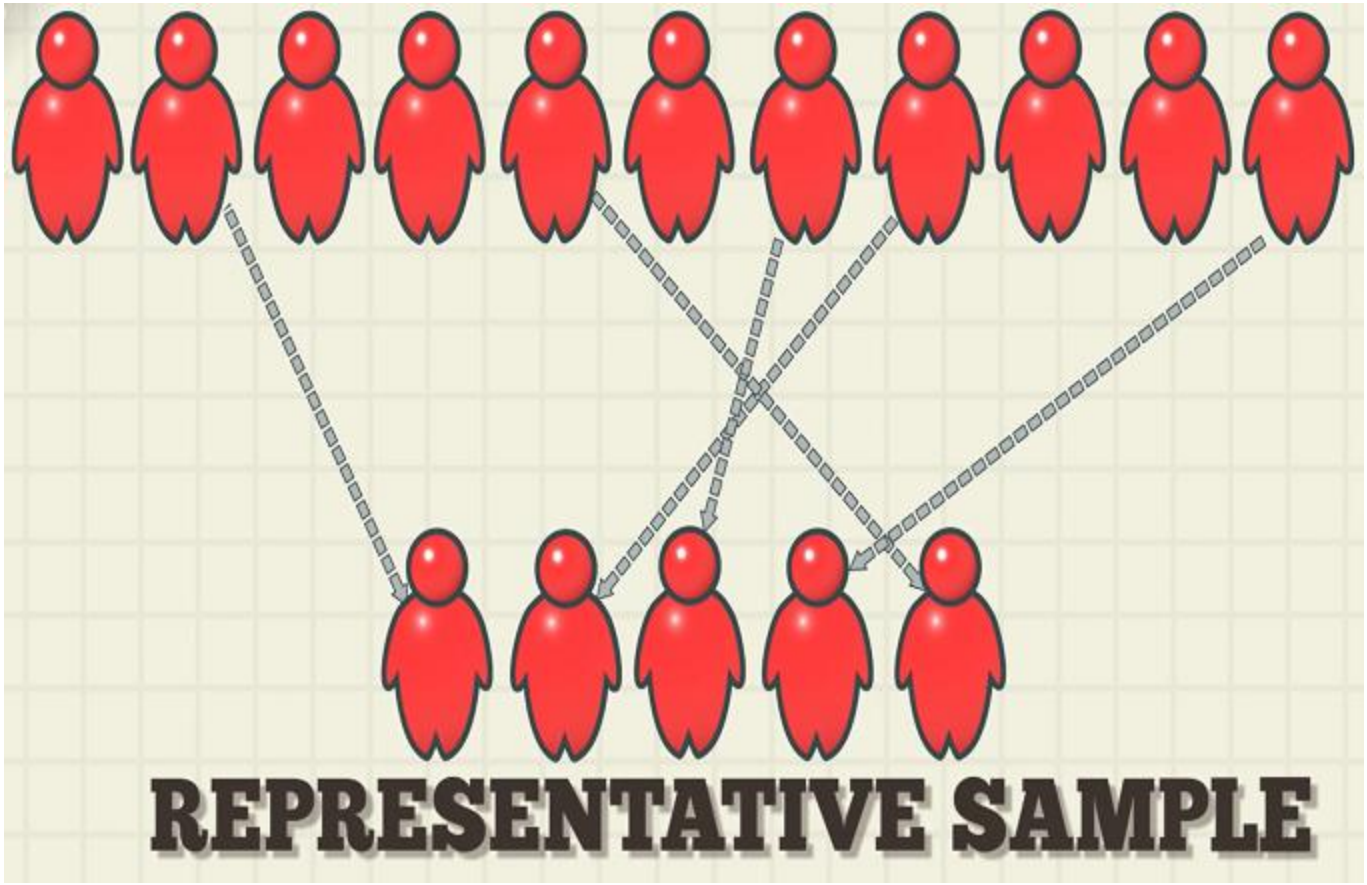




sampling rate ~ bin height

augmentation rate ~ cluster size

Sampling



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:

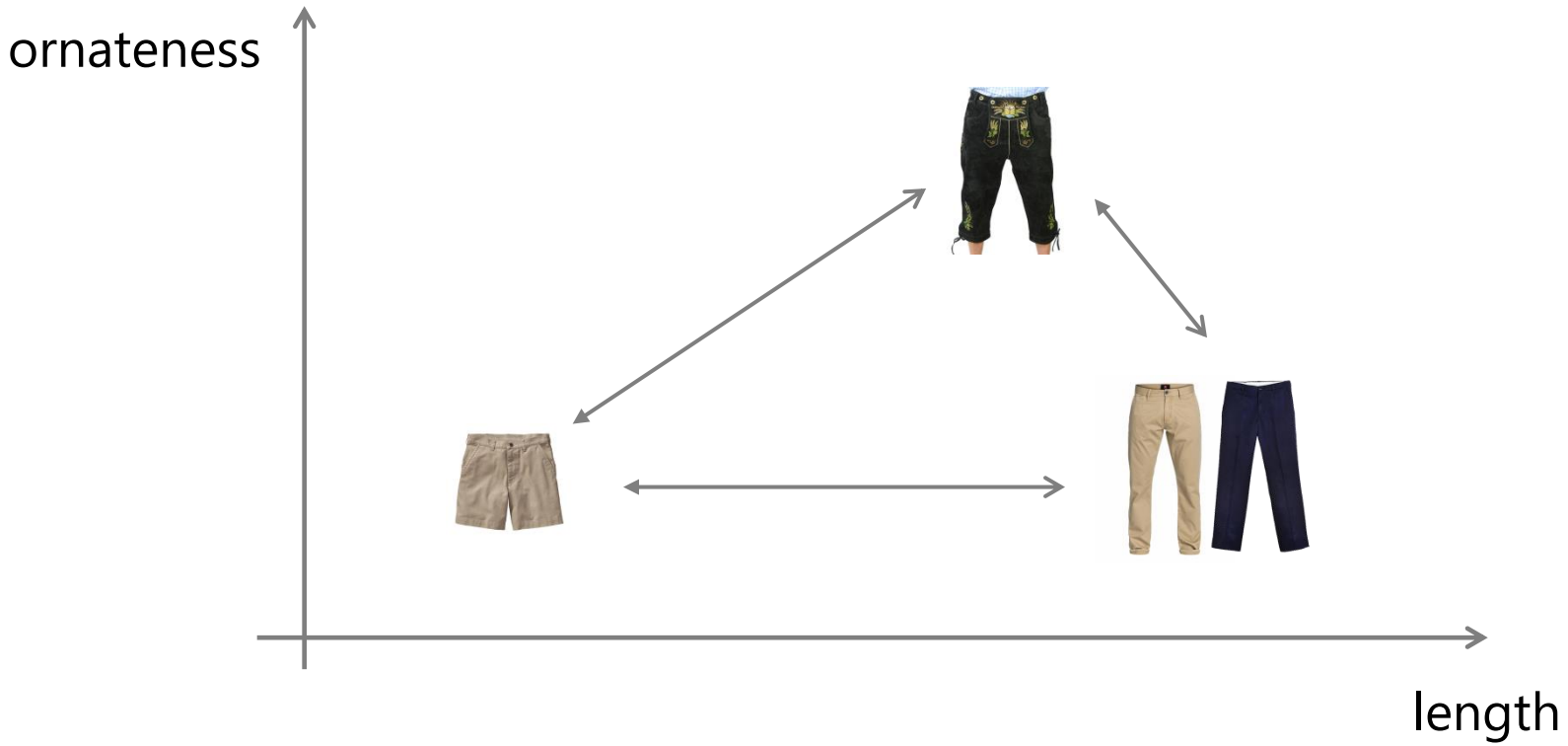
- but more like this:

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



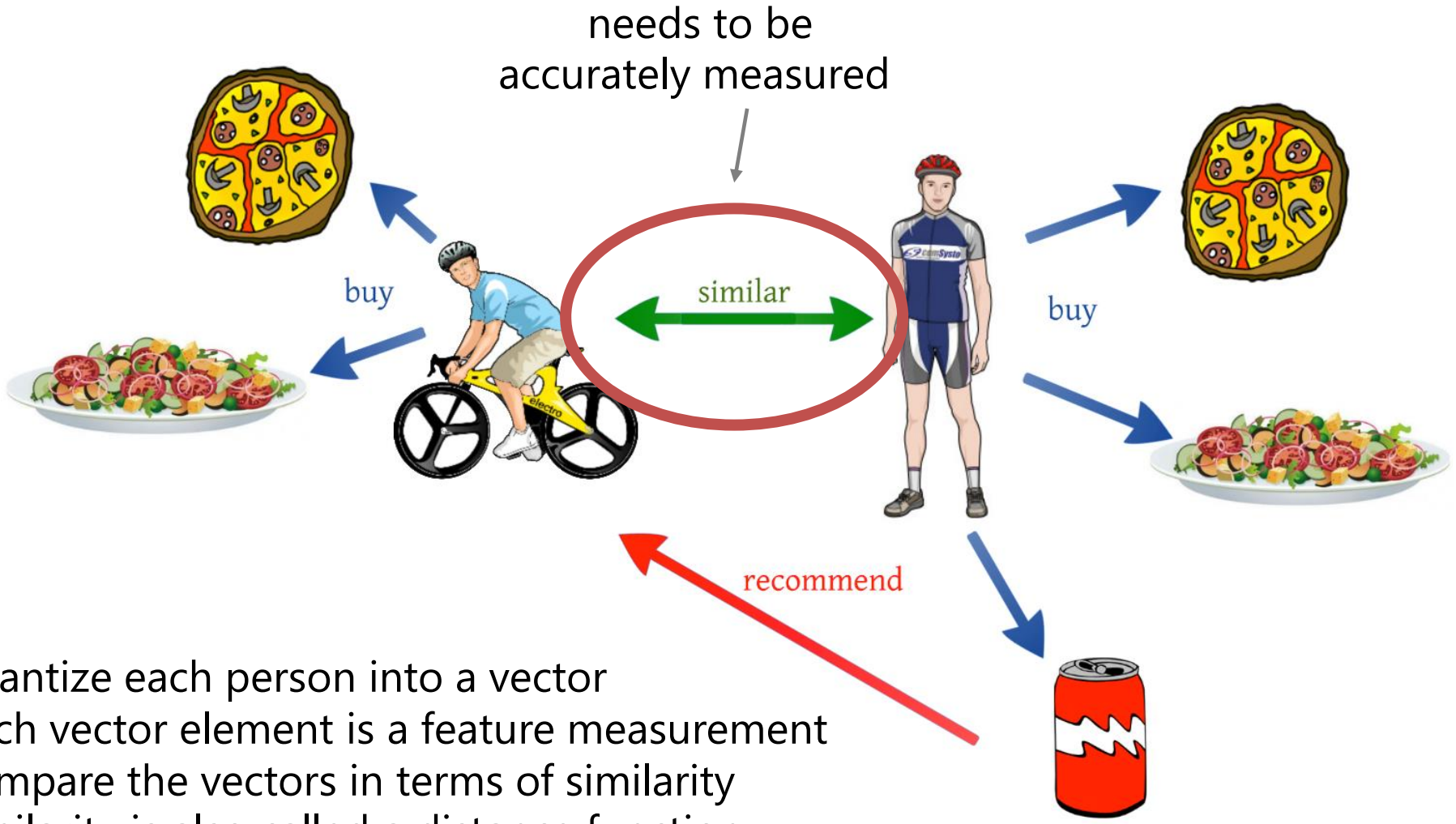
We did not consider color, texture, size, etc…

- 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



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>

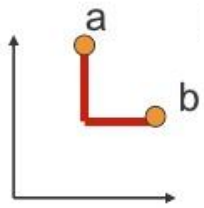
Examples:

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

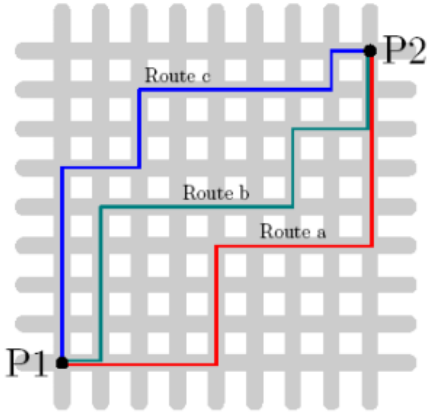
- food: <pepperoni, tossed, none>, <pepperoni, tossed, coke>, …
  expressed in numbers: <1, 1, 0>, <1, 1, 3>

Metric Distances

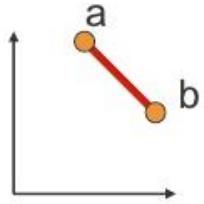
Manhattan distance



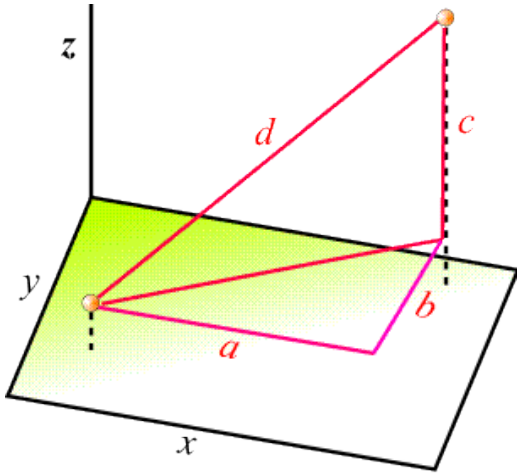
$$\text{dist(a,b)} = \|a - b\|_1 = \sum_i |a_i - b_i|$$



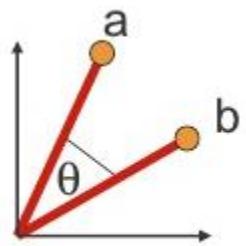
Euclidian distance



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



Cosine Similarity



$$\text{dist(a,b)} = cos^{-1} \frac{\langle a, b \rangle}{\|a\| \|b\|}$$

how is this related to correlation?

Pearson's Correlation = correlation similarity

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

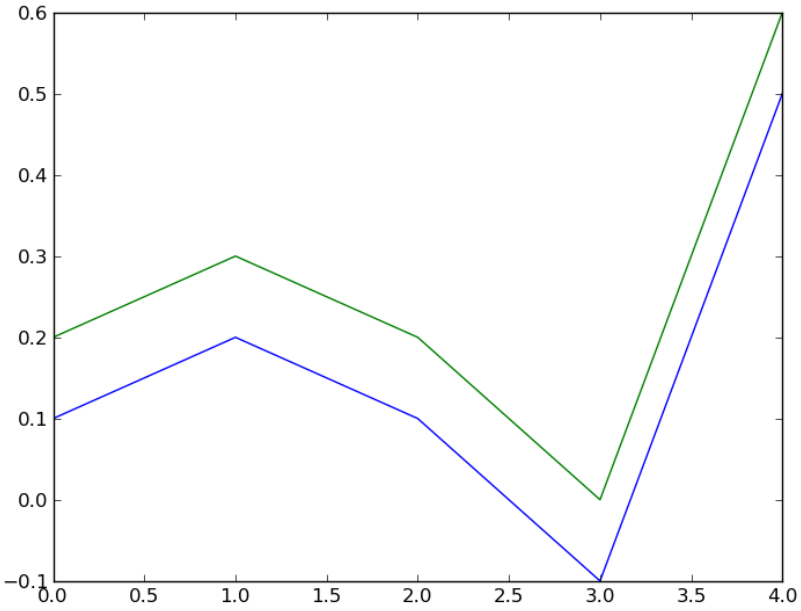
mean across all variable values for data items x, y

e.g. the "average looking" pair of pants or shoes

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



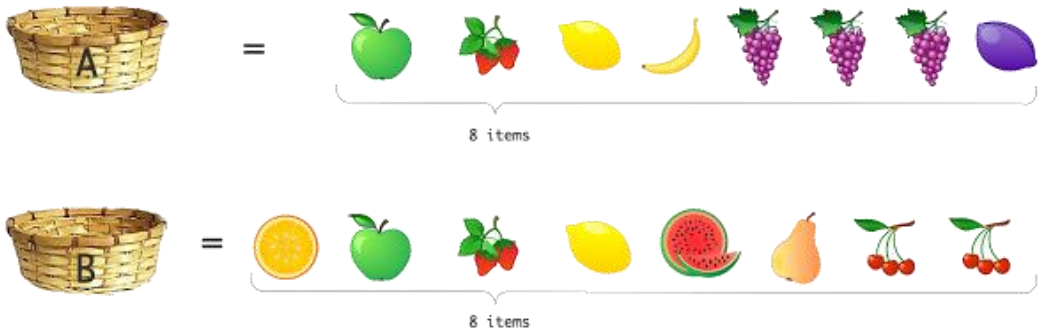
green = blue + 0.1

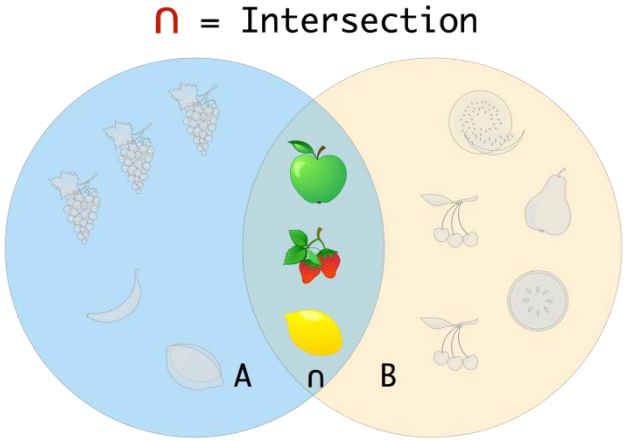
Both correlation and cosine similarity are invariant to multiplication with a constant

- invariant to scaling

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

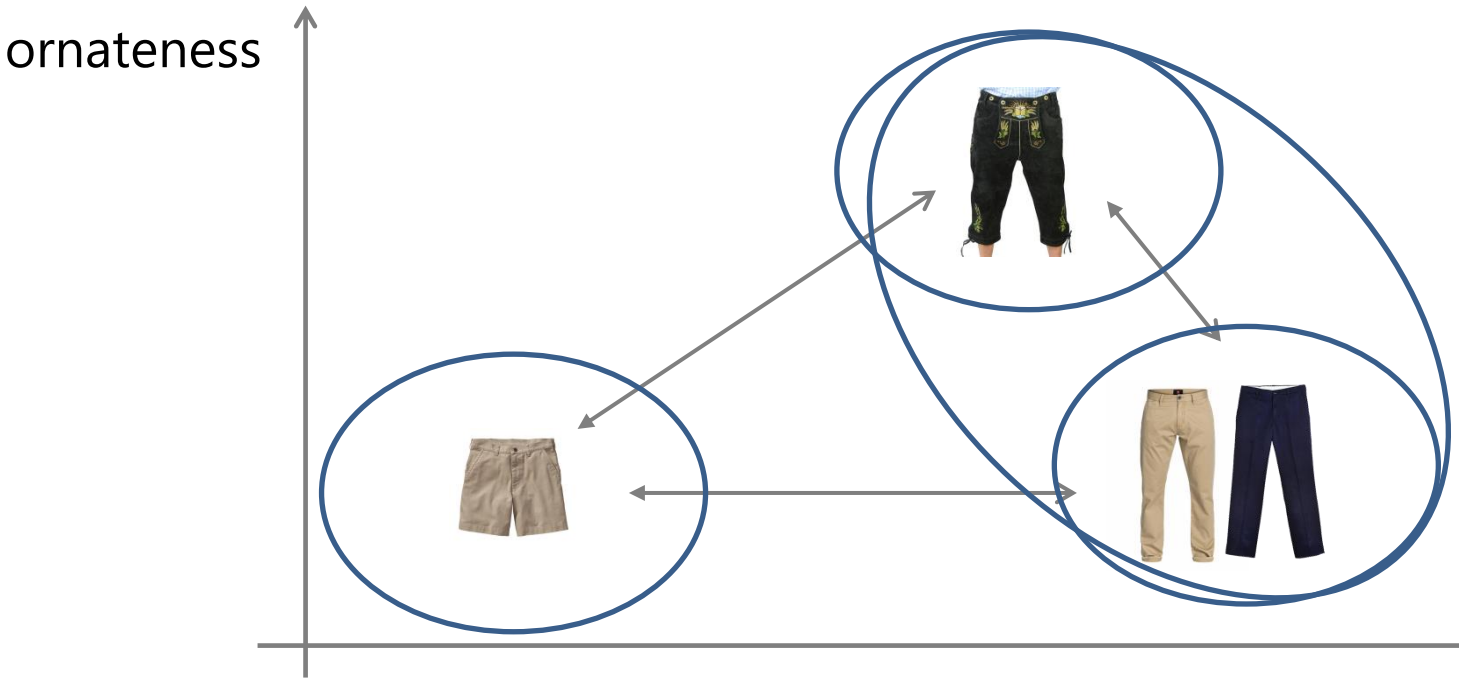
This process is called *clustering*

- 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

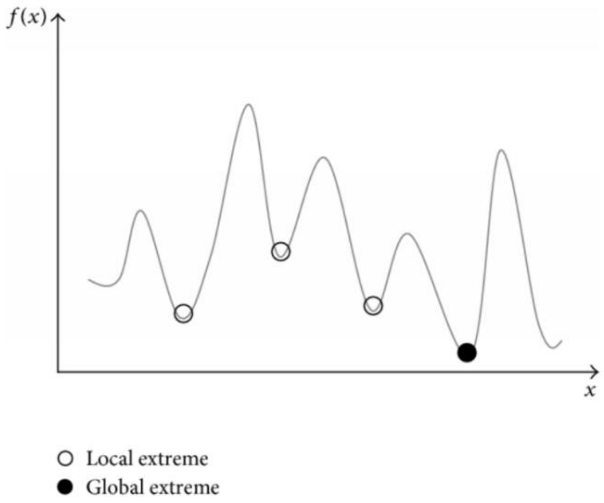
- and dissimilar from other groups of objects at the same time

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*

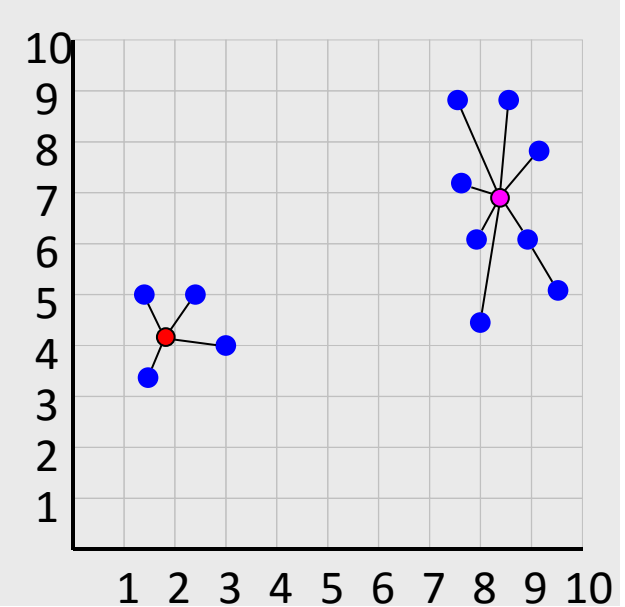
objective function ←

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2$$

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



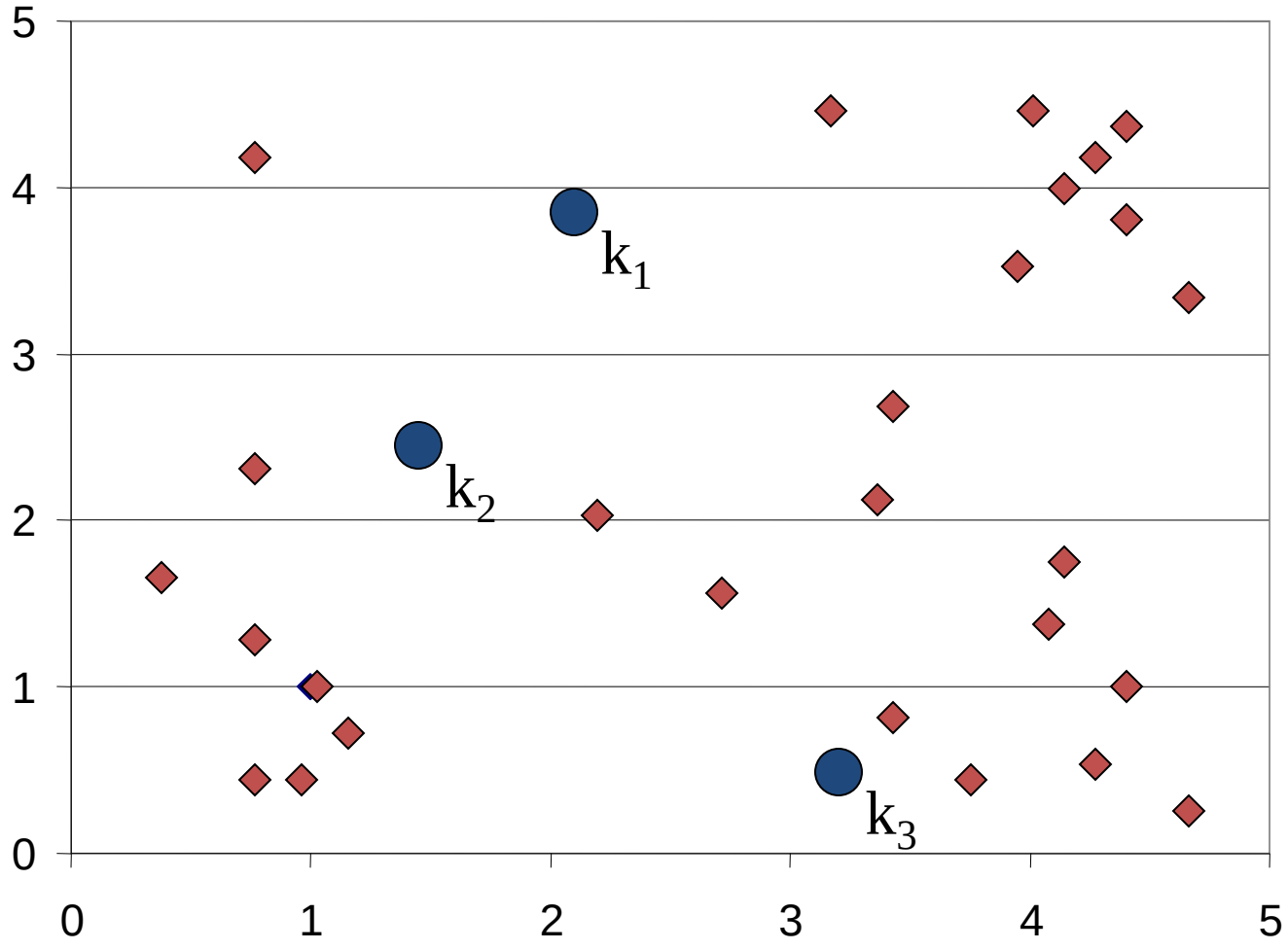
The K-Means Clustering Algorithm

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

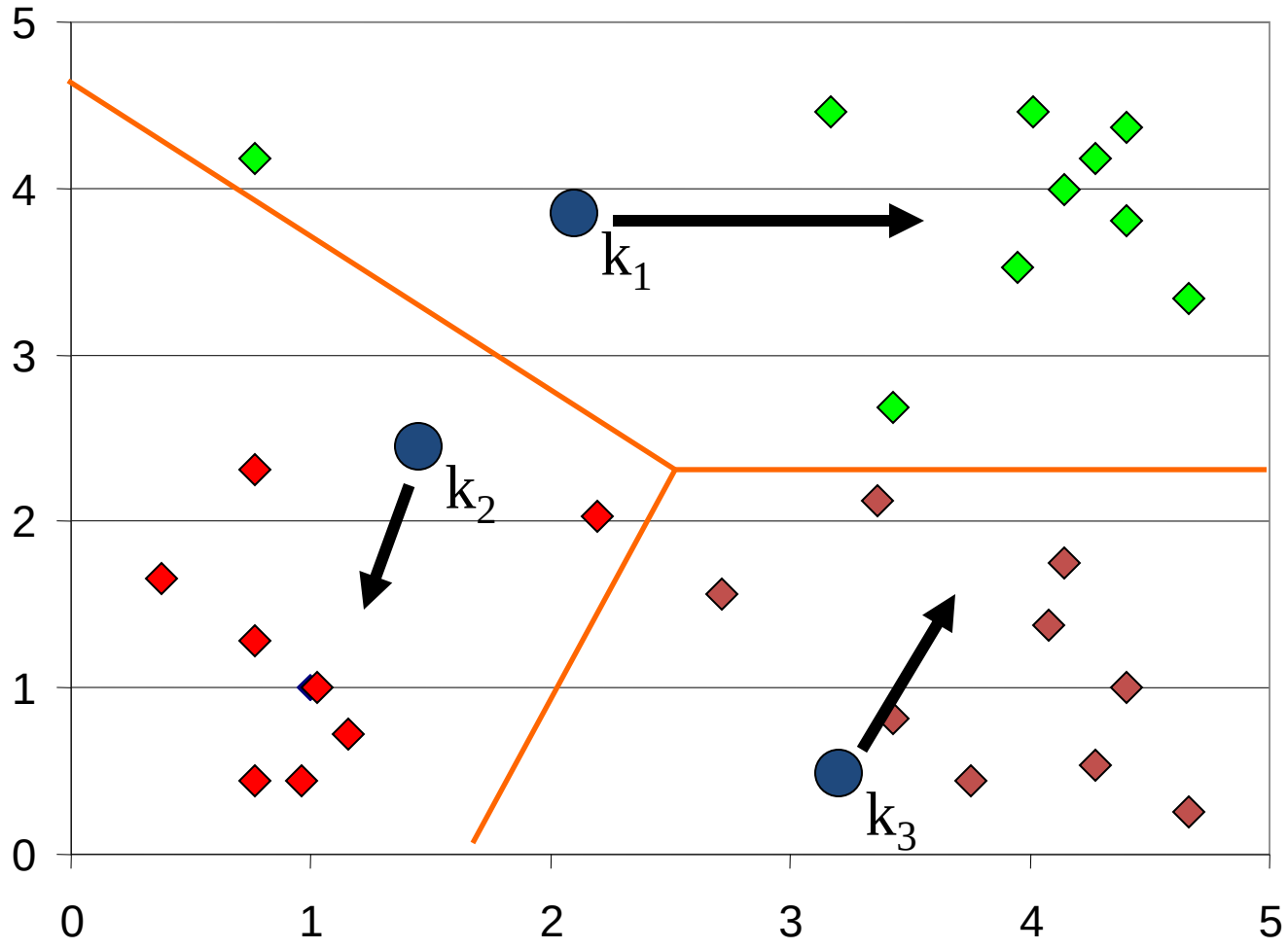
K-means Clustering: Step 1

Algorithm: k-means, Distance Metric: Euclidean Distance



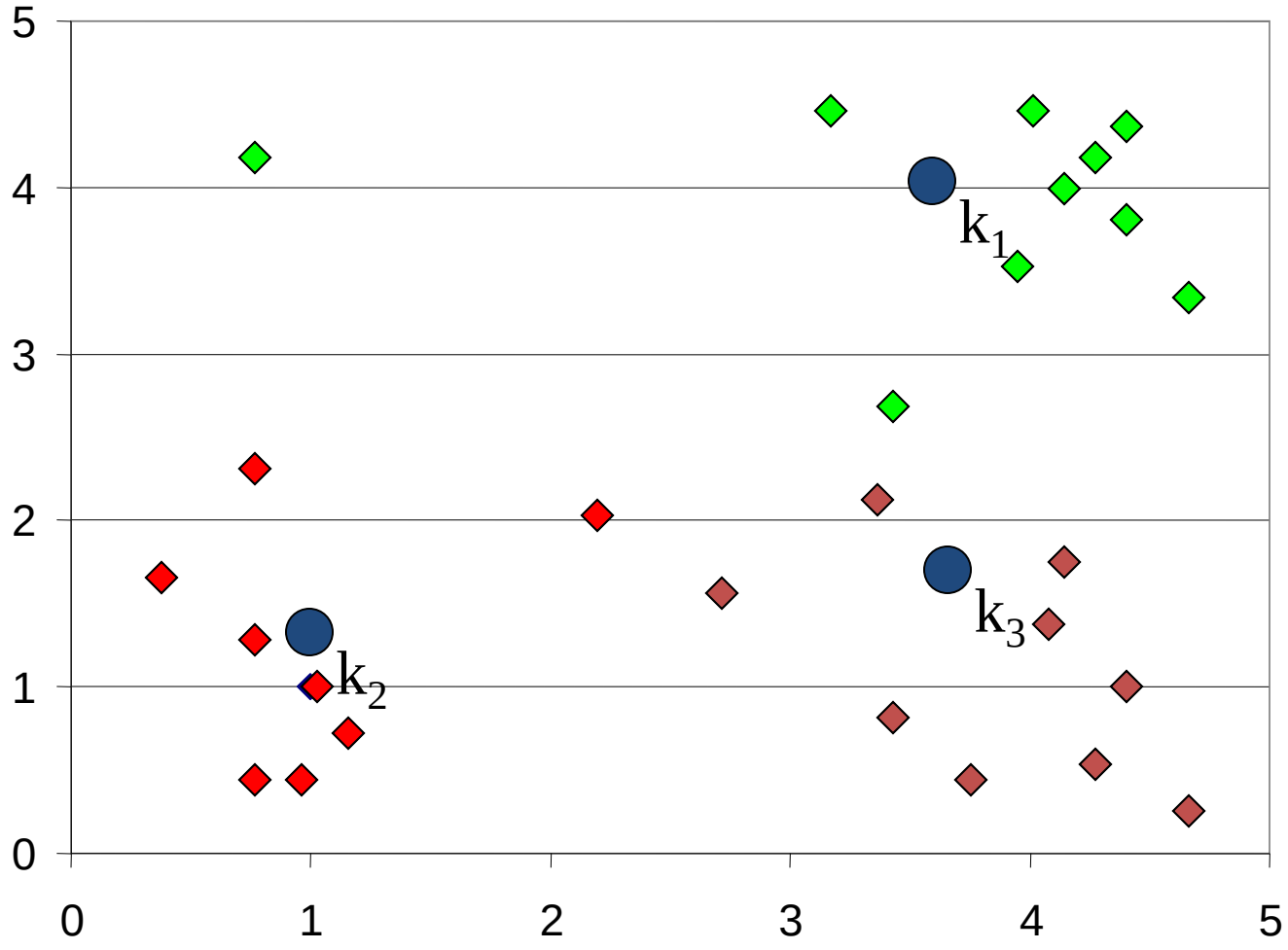
K-means Clustering: Step 2

Algorithm: k-means, Distance Metric: Euclidean Distance



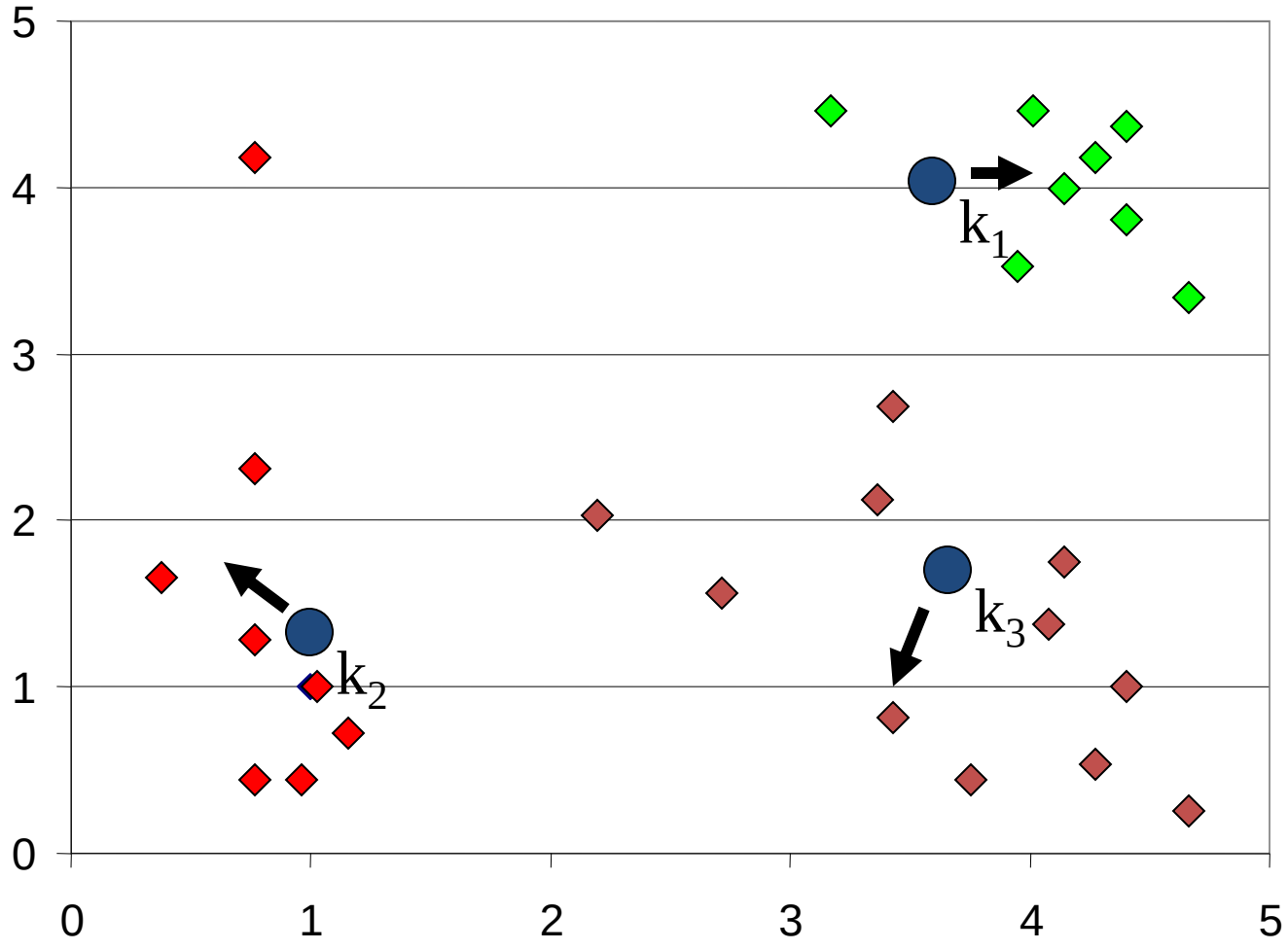
K-means Clustering: Step 3

Algorithm: k-means, Distance Metric: Euclidean Distance



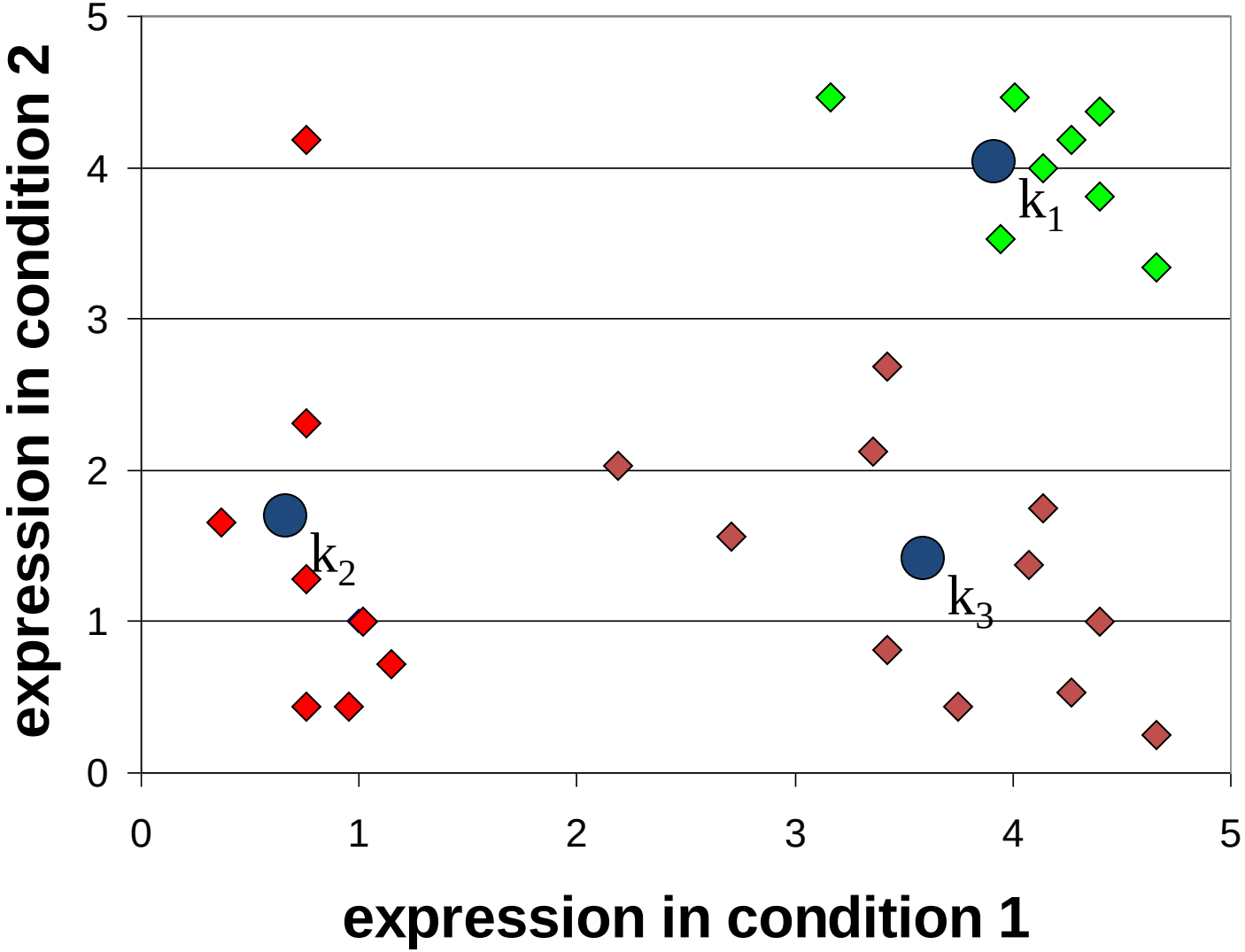
K-means Clustering: Step 4

Algorithm: k-means, Distance Metric: Euclidean Distance



K-means Clustering: Step 5

Algorithm: k-means, Distance Metric: Euclidean Distance



K-Means Algorithm – Comments

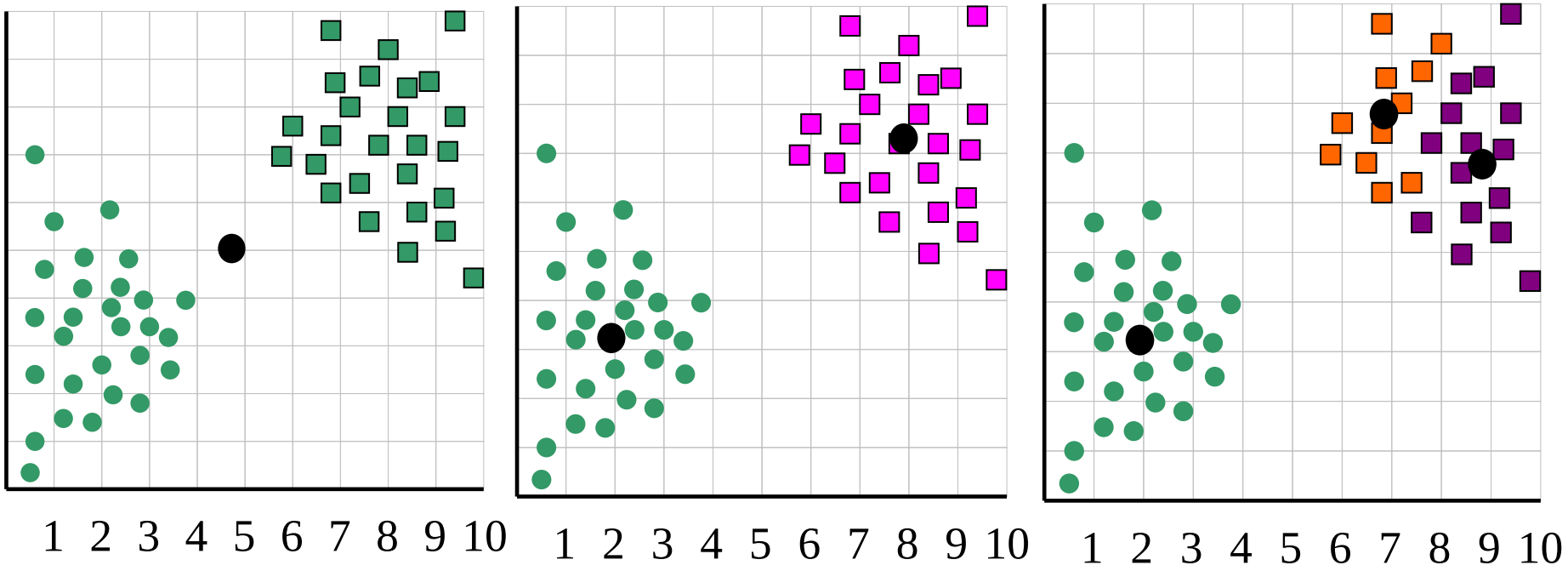
Strengths:

- *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

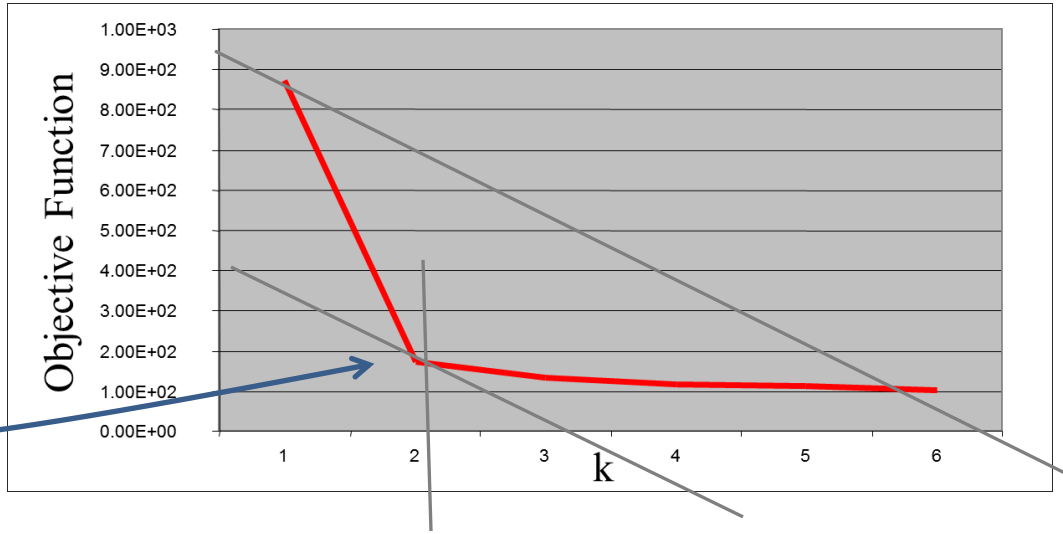
How About K=2?

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”