Clustering, K-Means, and K-Nearest Neighbors

CMSC 478

UMBC

Most slides courtesy Hamed Pirsiavash

Outline

Clustering basics

K-means: basic algorithm & extensions

Cluster evaluation

Non-parametric mode finding: density estimation

Graph & spectral clustering

Hierarchical clustering

K-Nearest Neighbor

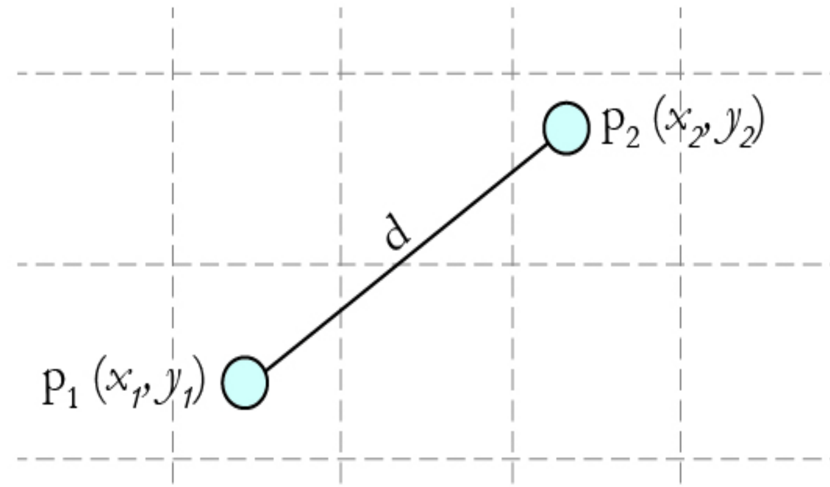
Nearest neighbor classifier

Will Alice like the movie?

Alice and James are similar

James likes the movie →

Alice must/might also like the movie



$$\text{Euclidean distance (d)} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Represent data as vectors of feature values

Find closest (Euclidean norm) points

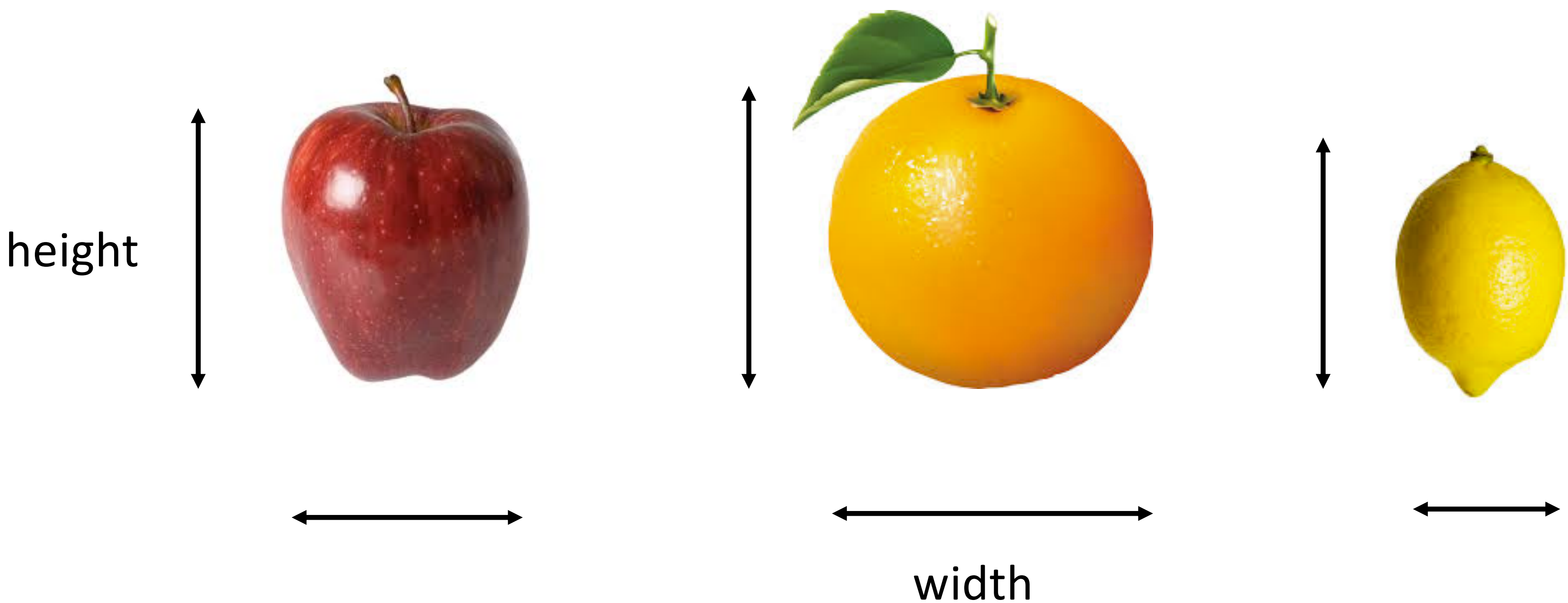
Nearest neighbor classifier

Training data is in the form of $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$

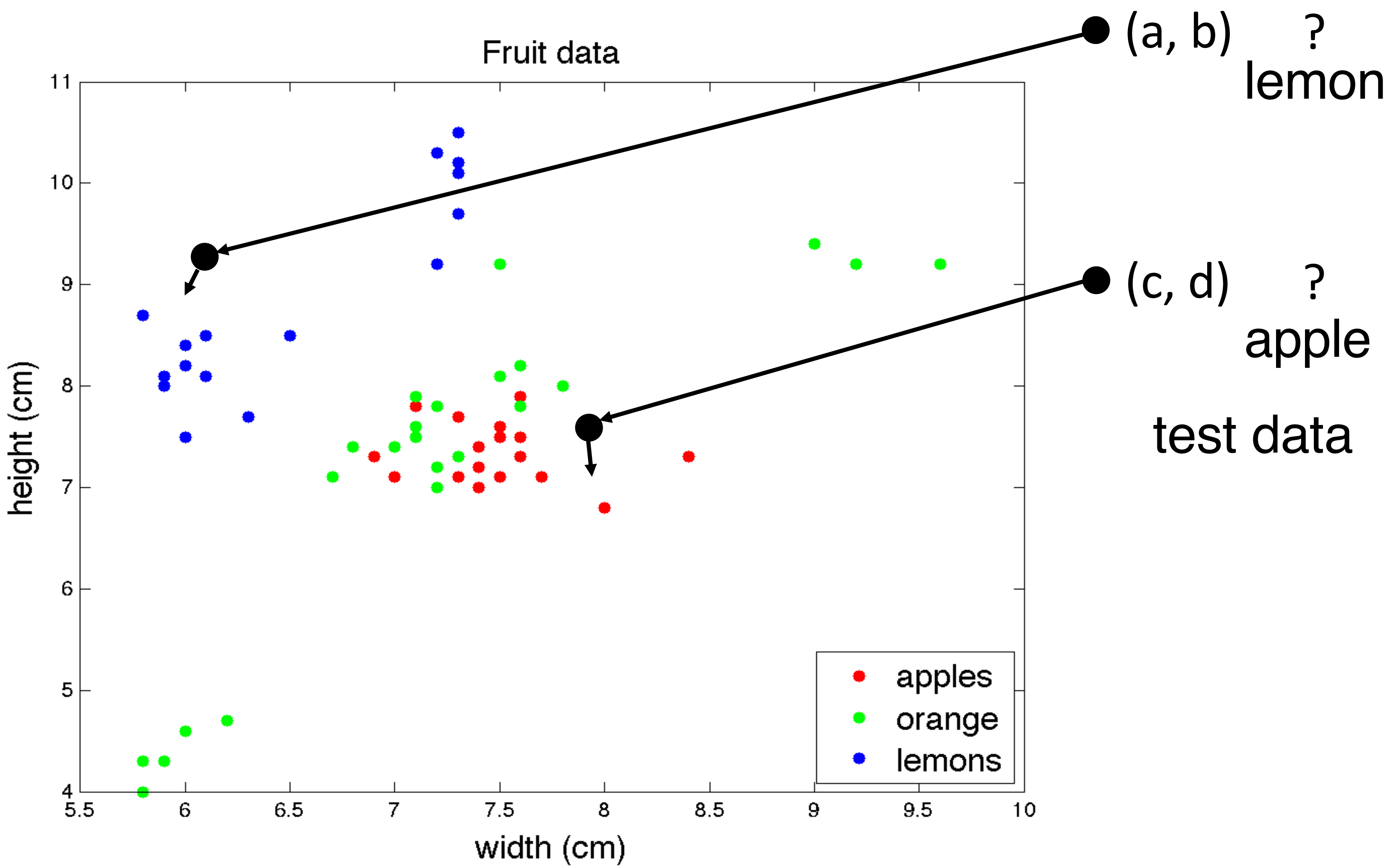
Fruit data:

label: {apples, oranges, lemons}

attributes: {width, height}

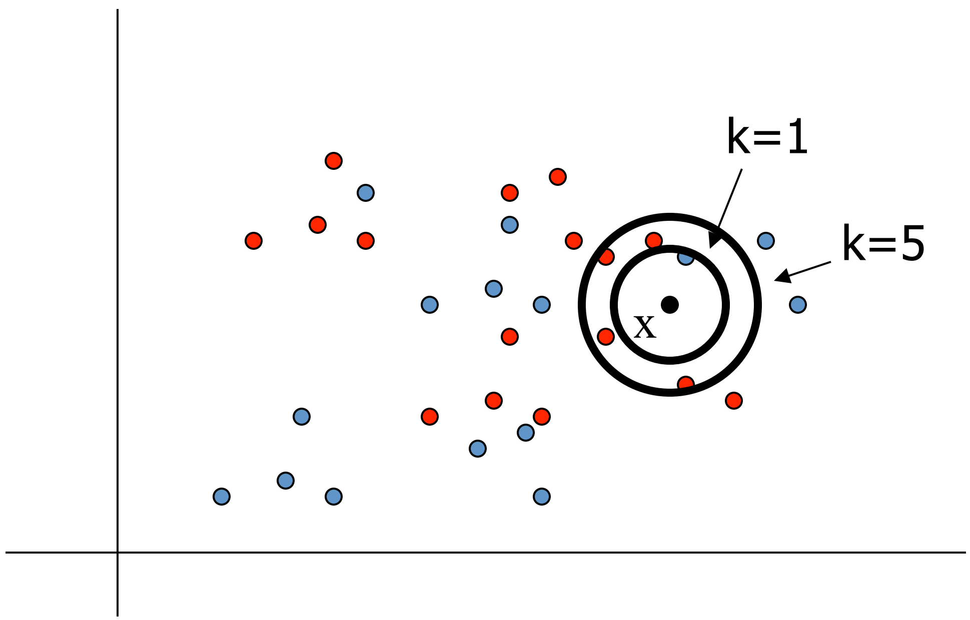


Nearest neighbor classifier



K-Nearest Neighbor Methods

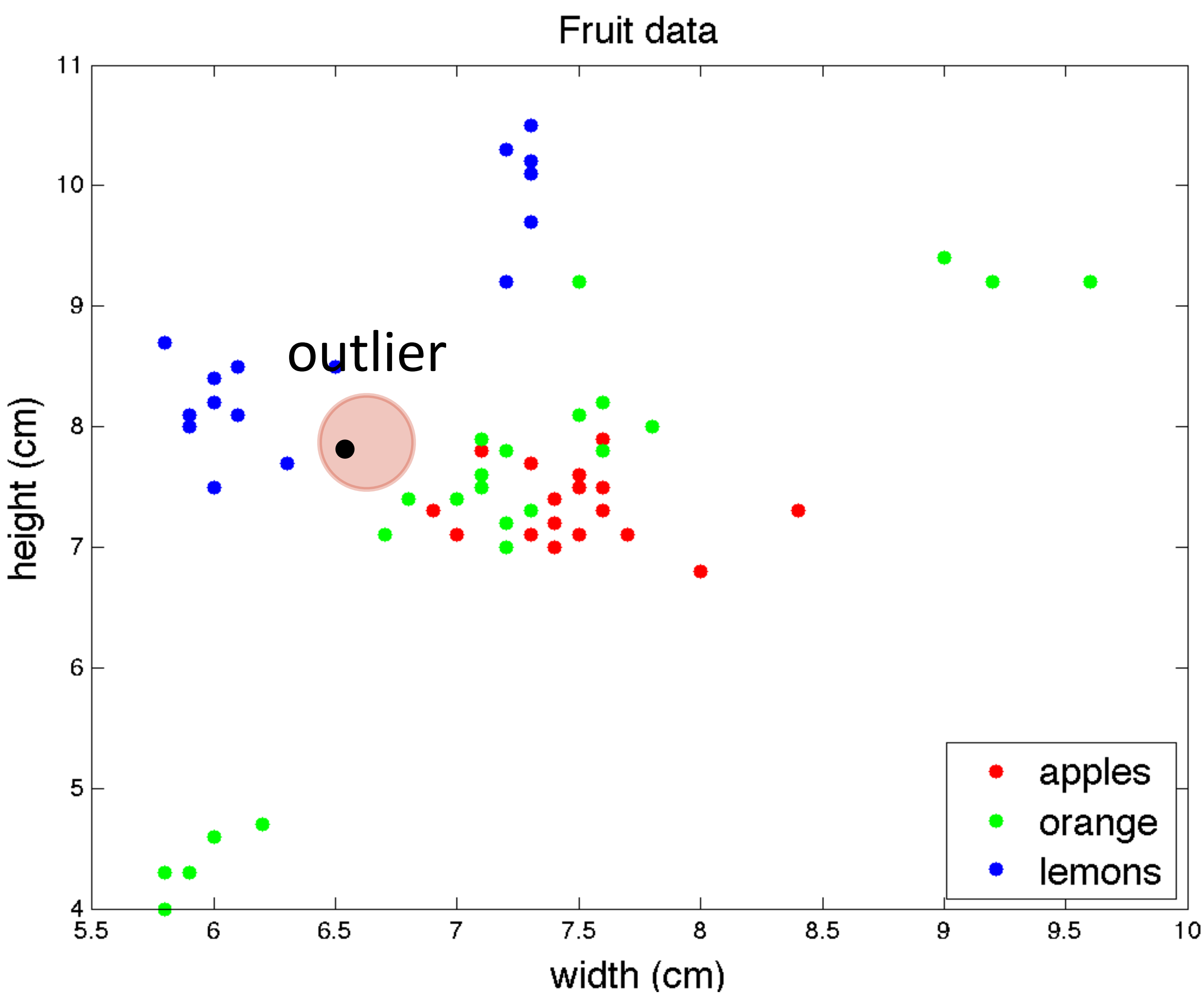
- To classify a new input vector x, examine the k-closest training data points to x and assign the object to the most frequently occurring class



common values for k: 3, 5

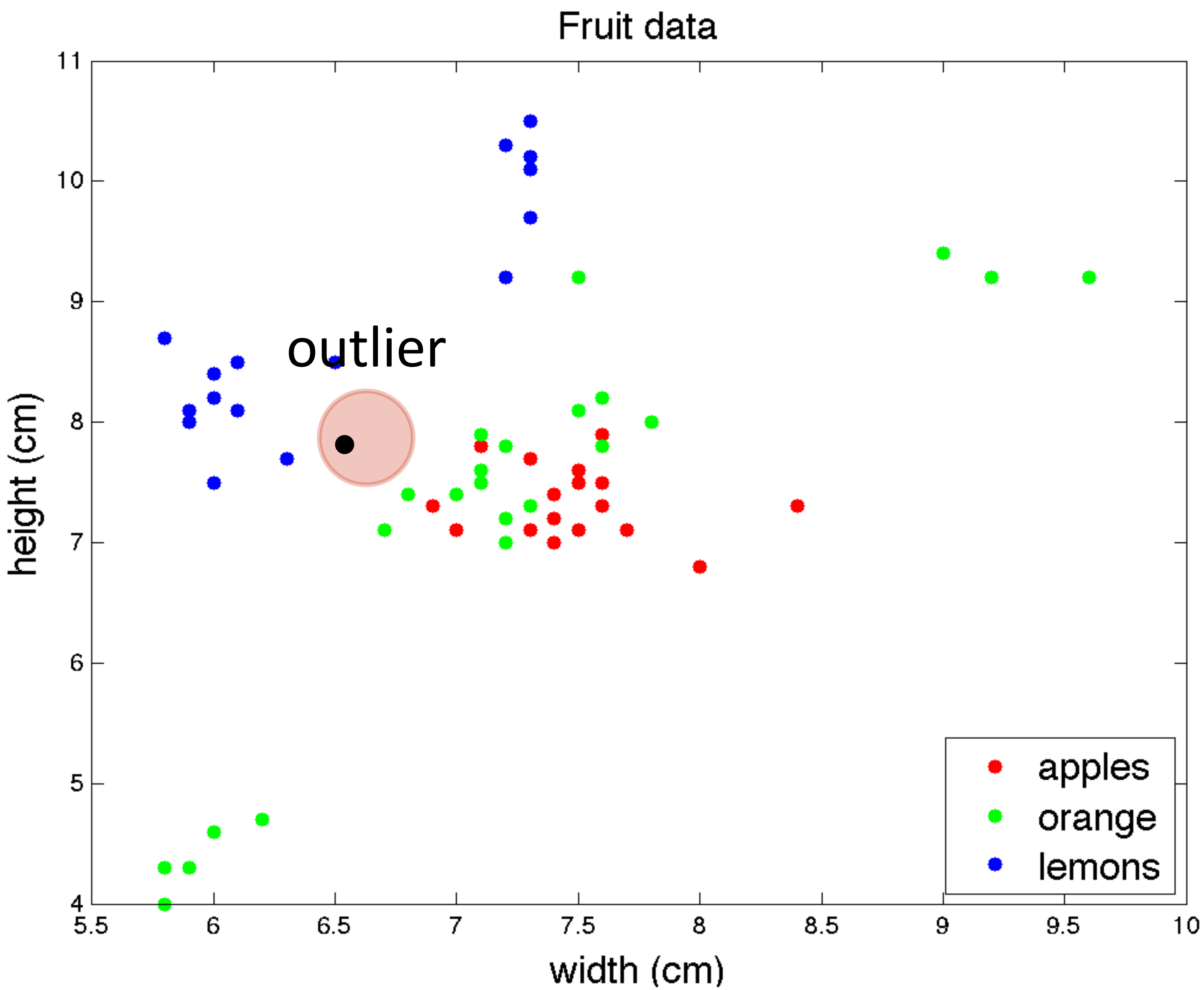
k-Nearest neighbor classifier

Take majority vote among the k nearest neighbors



k-Nearest neighbor classifier

Take majority vote among the k nearest neighbors

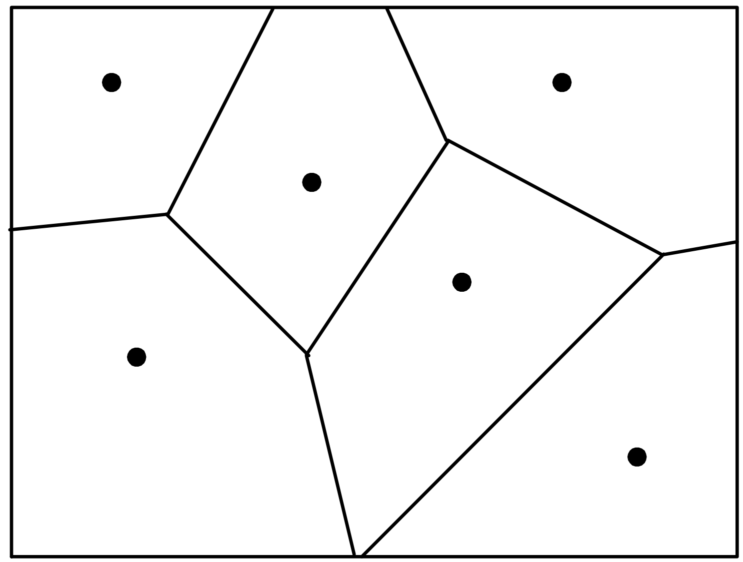


What is the effect of k?

Decision Boundaries

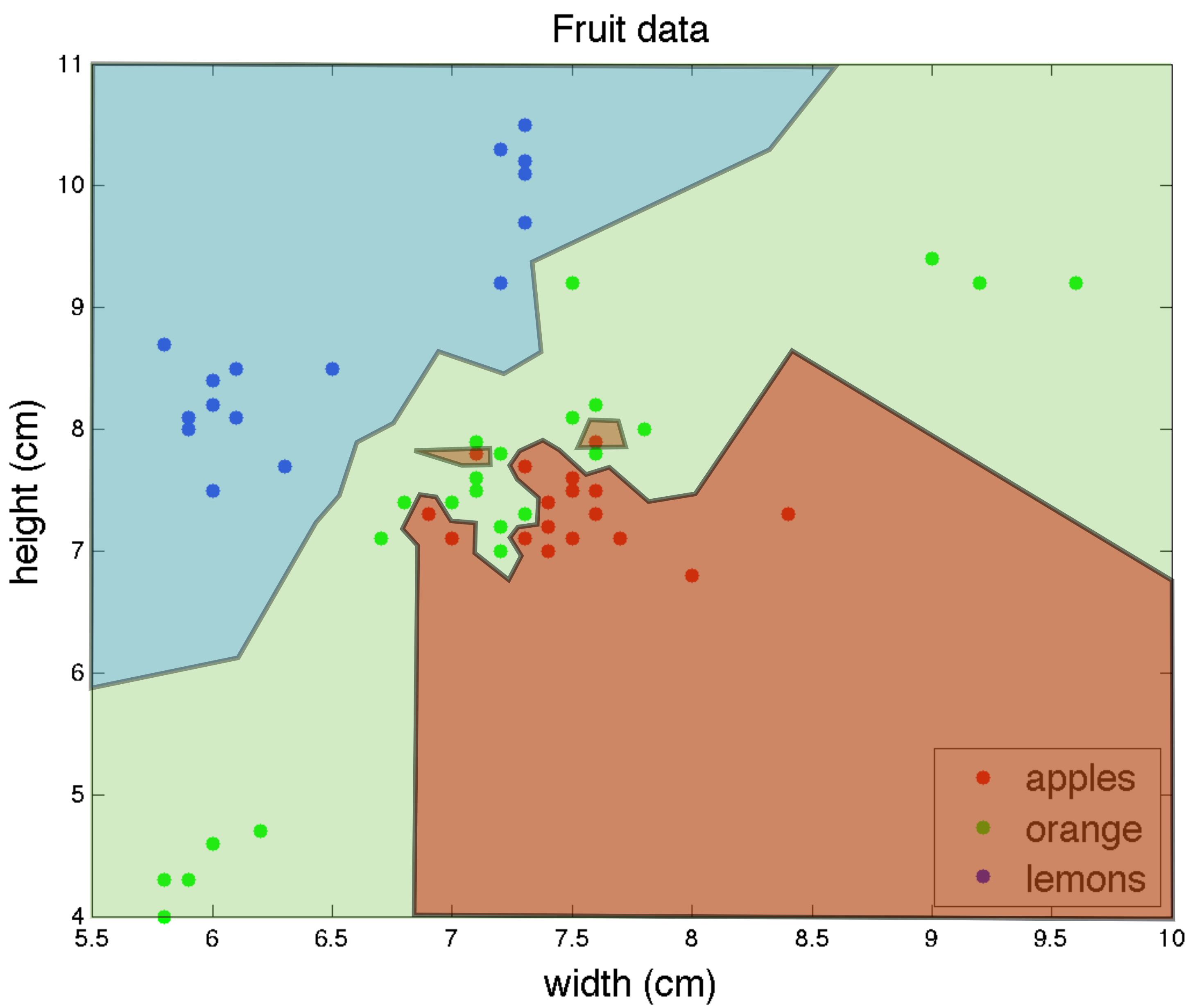
- The nearest neighbor algorithm does not explicitly compute decision boundaries. However, the decision boundaries form a subset of the Voronoi diagram for the training data.

1-NN Decision Surf ace

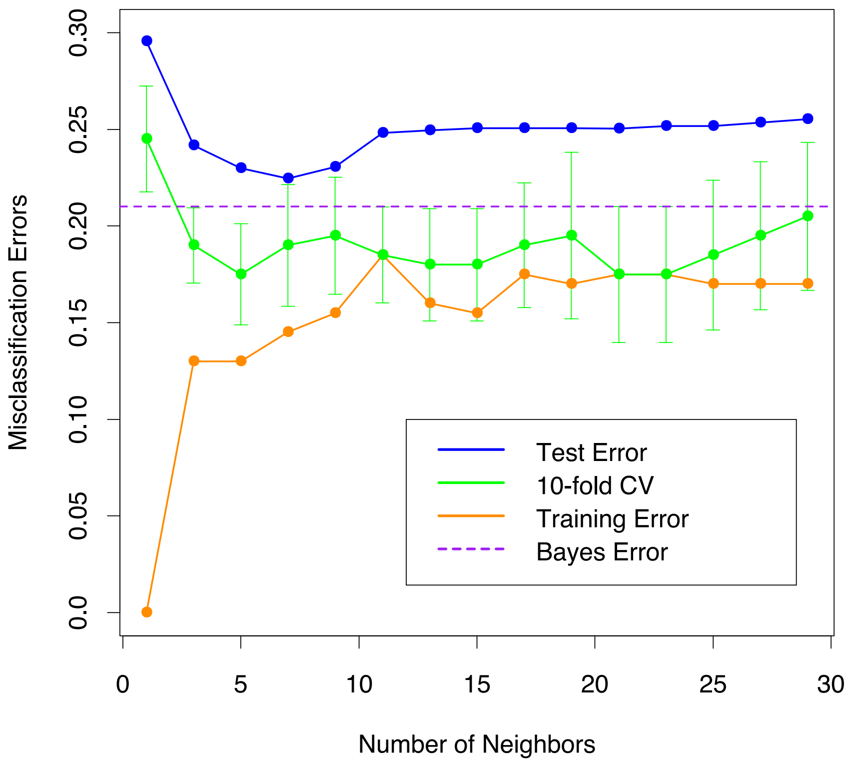


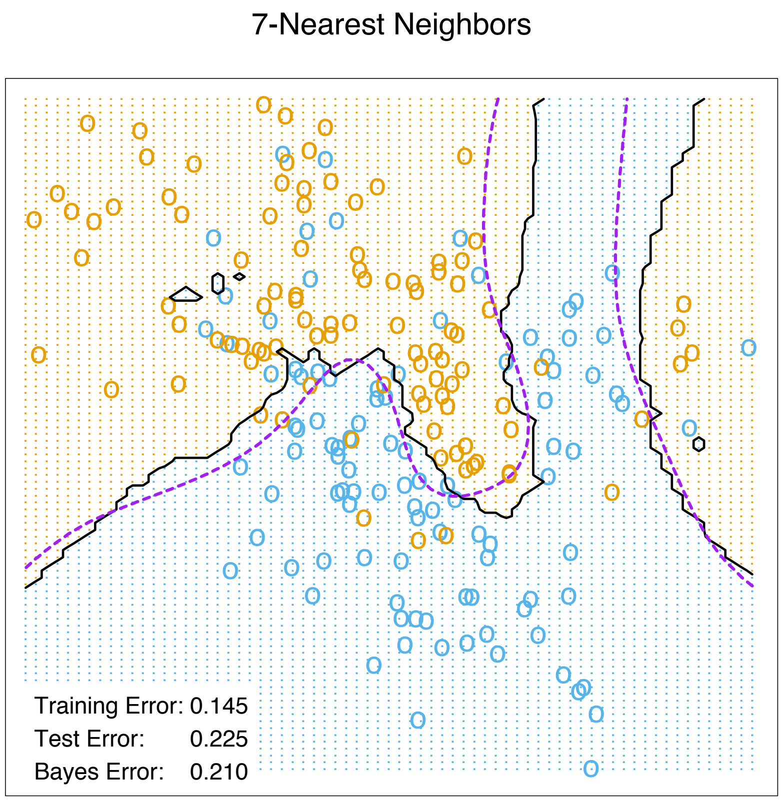
- The more examples that are stored, the more complex the decision boundaries can become

Decision boundaries: 1NN



Example results for k-NN





[Figures from Hastie and Tibshirani, Chapter 13]

Inductive bias of the kNN classifier

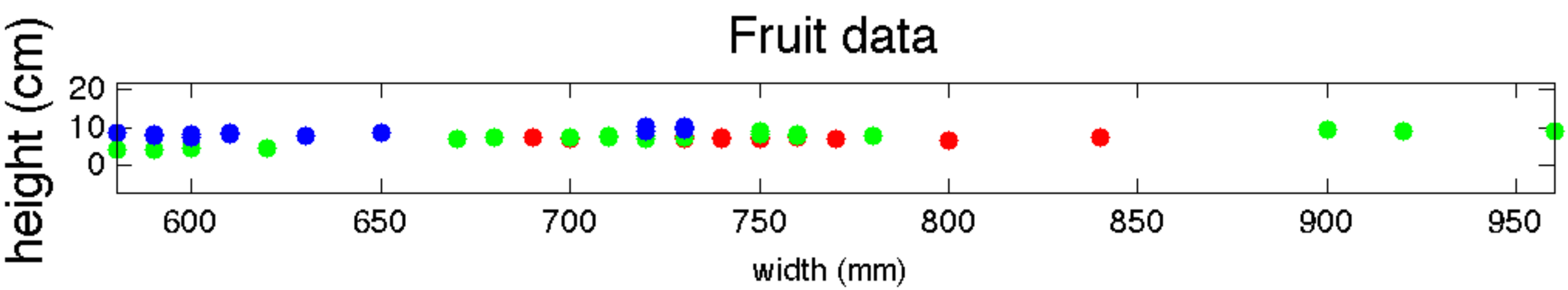
Choice of features

- We are assuming that all features are equally important
- What happens if we scale one of the features by a factor of 100?

Choice of distance function

- Euclidean, cosine similarity (angle), Gaussian, etc …
- Should the coordinates be independent?

Choice of k



Distance

- Notation: object with p measurements

$$x^i = (x^i_1, x^i_2, \ldots, x^i_p)$$

- Most common distance metric is *Euclidean* distance:

$$d_E(x^i, x^j) = \left(\sum_{k=1}^{p} (x^i_k - x^j_k)^2 \right)^{\frac{1}{2}}$$

- ED makes sense when different measurements are commensurate; each is variable measured in the same units.

- If the measurements are different, say length and weight, it is not clear.

Standardization

When variables are not commensurate, we can standardize them by dividing by the sample standard deviation. This makes them all equally important.

The estimate for the standard deviation of x_k :

$$\hat{\sigma}_k = \left(\frac{1}{n} \sum_{i=1}^{n} \left(x_k^i - \bar{x}_k \right)^2 \right)^{\frac{1}{2}}$$

where x_k is the sample mean:

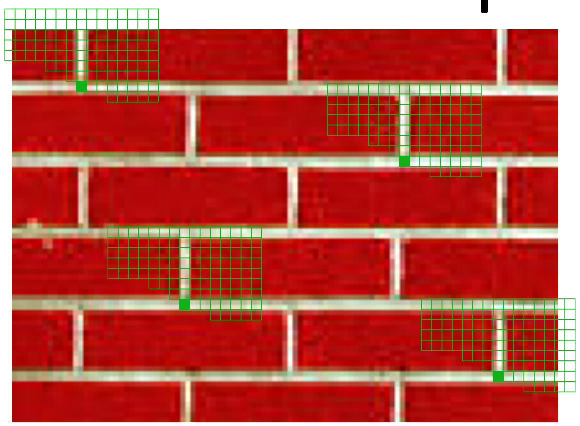
$$\bar{x}_k = \frac{1}{n} \sum_{i=1}^{n} x_k^i$$

Weighted Euclidean distance

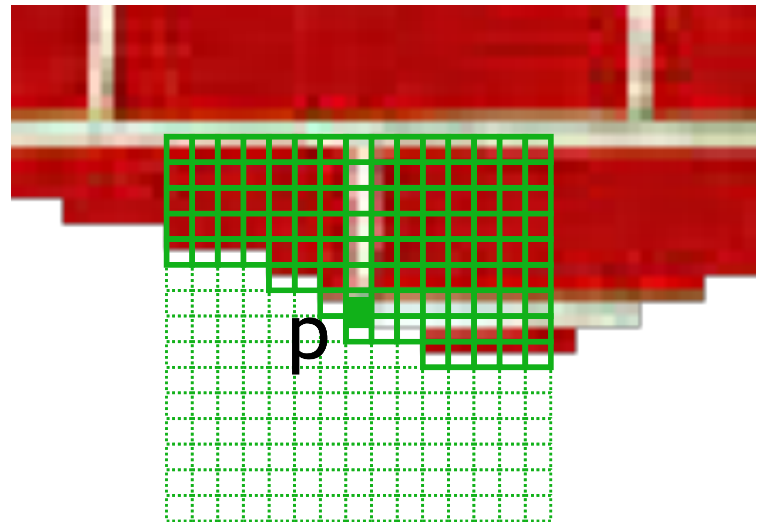
Finally, if we have some idea of the relative importance of each variable, we can weight them:

$$d_{WE}(i,j) = \left(\sum_{k=1}^{p} w_k (x_k^i - x_k^j)^2 \right)^{\frac{1}{2}}$$

An example: Synthesizing one pixel



input image



synthesized image

What is $P(\mathbf{x}|\text{neighborhood of pixels around x})$

Find all the windows in the image that match the neighborhood

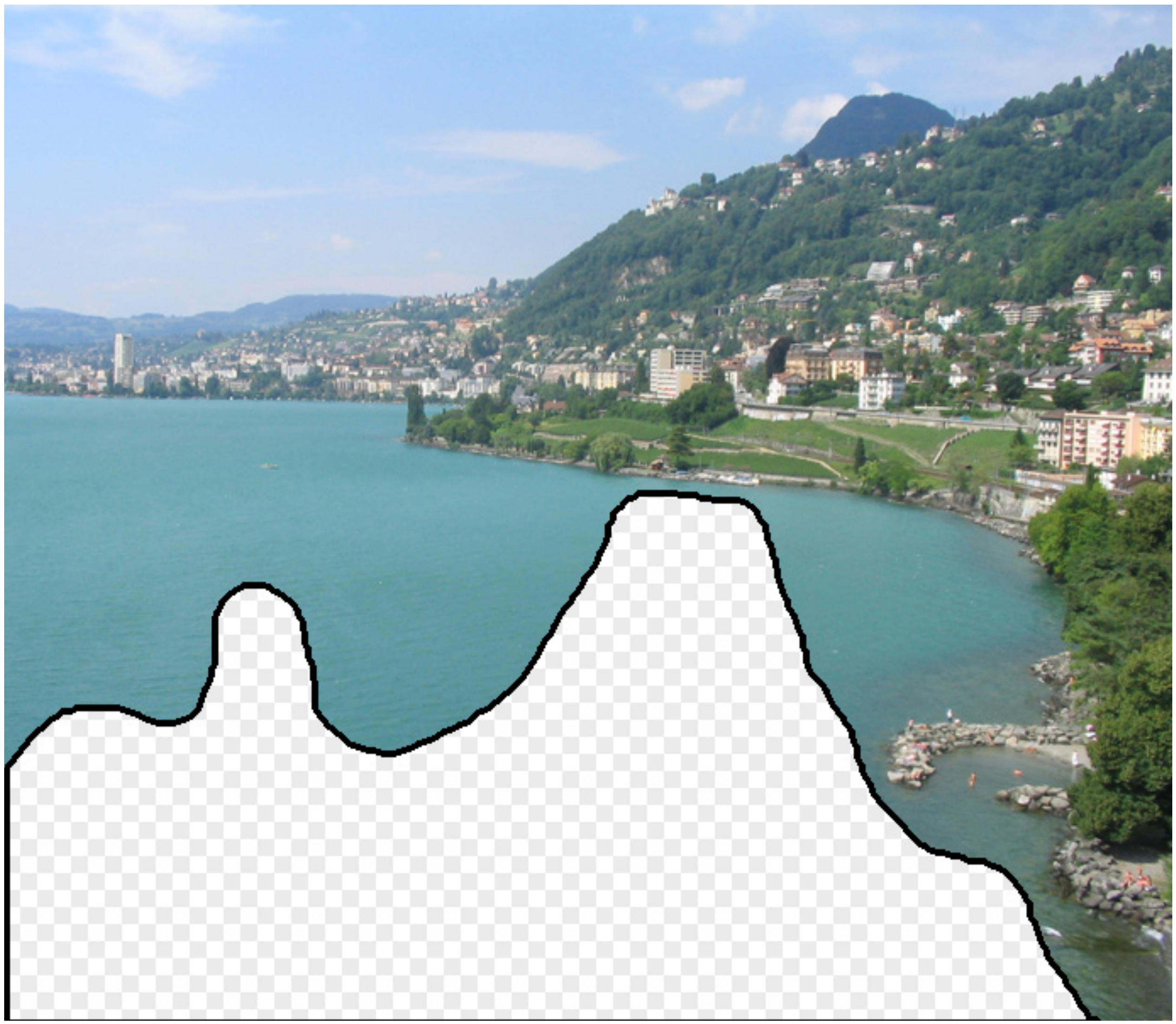
To synthesize **x**

- pick one matching window at random
- assign **x** to be the center pixel of that window

An **exact** match might not be present, so find the **best** matches using **Euclidean distance** and randomly choose between them, preferring better matches with higher probability

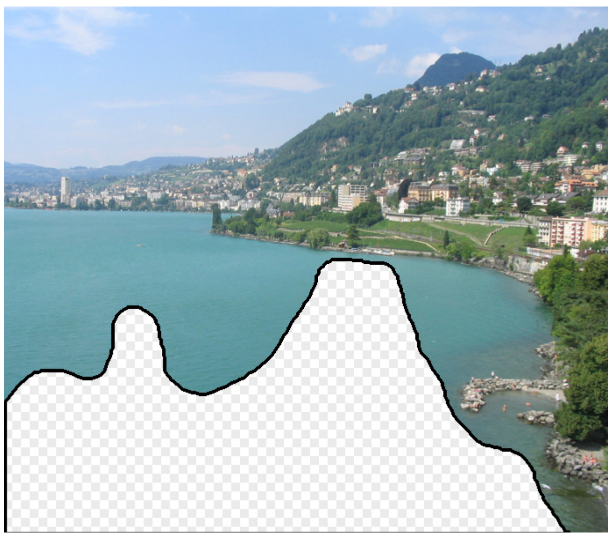
Slide from Alyosha Efros, ICCV 1999

kNN: Scene Completion

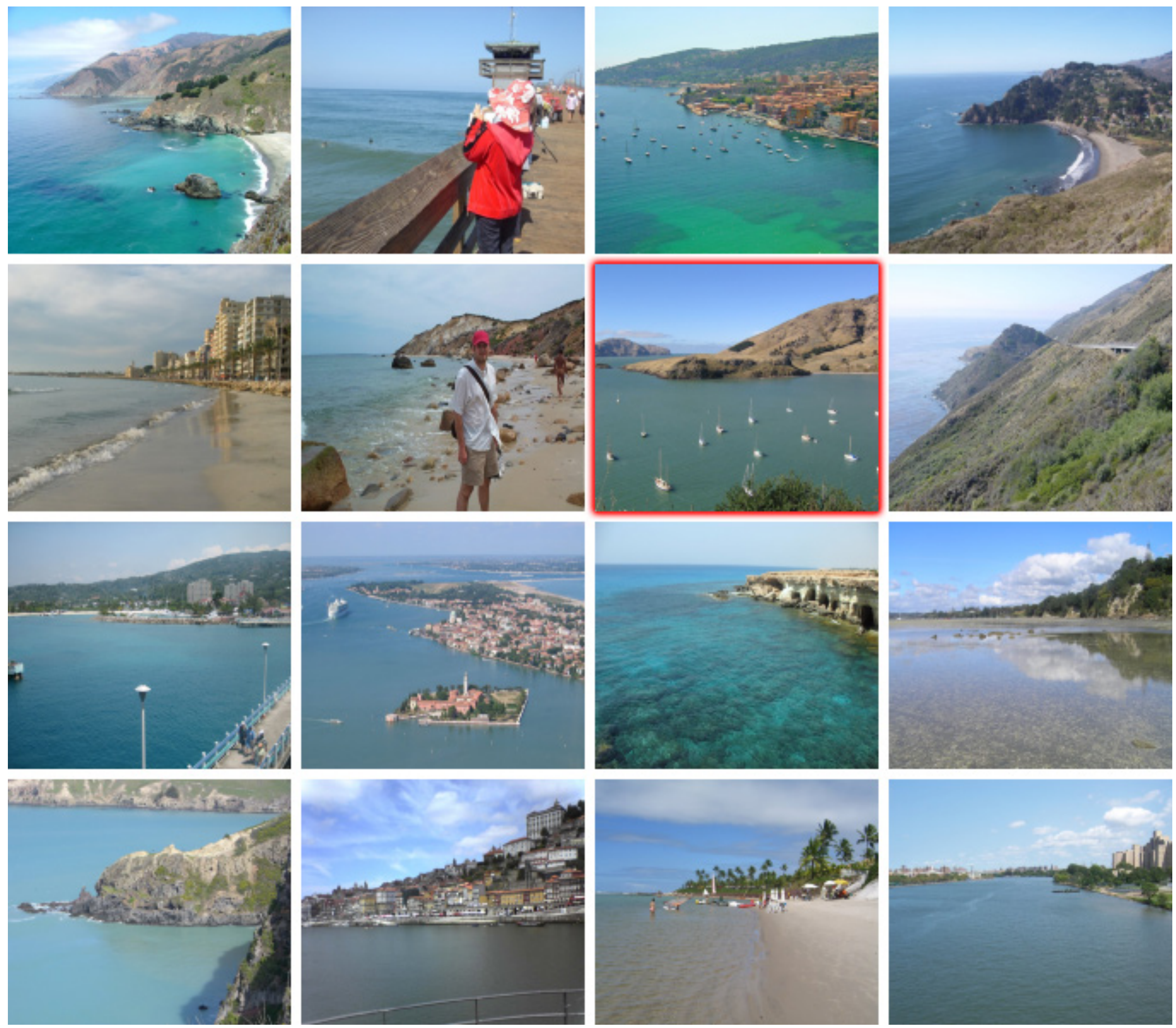


“Scene completion using millions of photographs”, Hayes and Efros, TOG 2007

kNN: Scene Completion

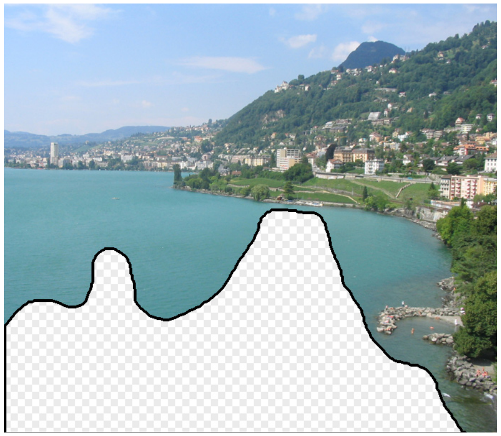


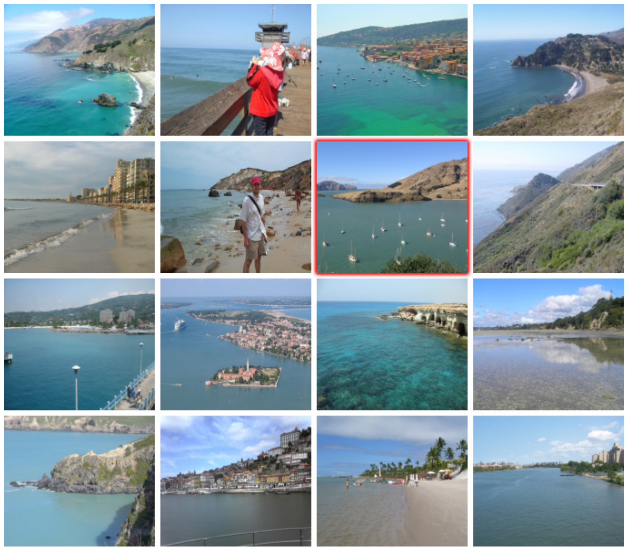
Nearest neighbors



“Scene completion using millions of photographs”, Hayes and Efros, TOG 2007

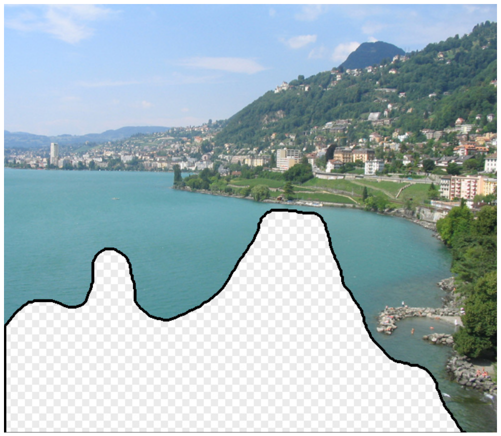
kNN: Scene Completion

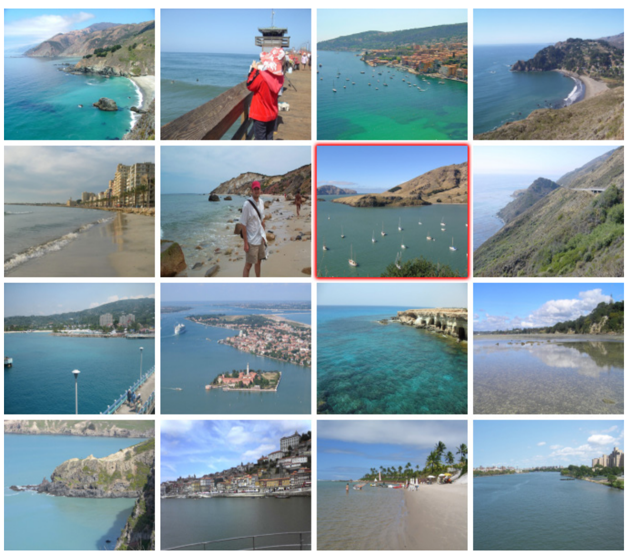




“Scene completion using millions of photographs”, Hayes and Efros, TOG 2007

kNN: Scene Completion





“Scene completion using millions of photographs”, Hayes and Efros, TOG 2007

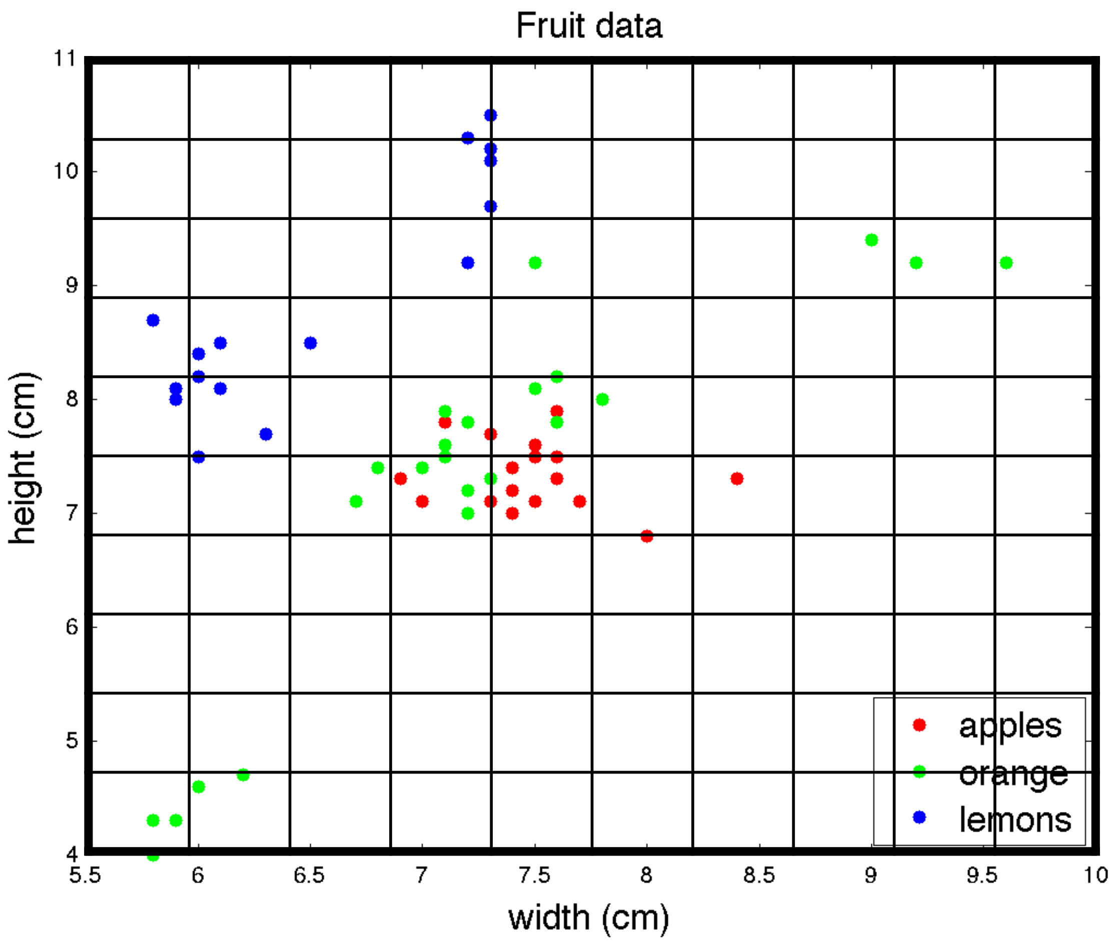
Practical issue when using kNN: speed

Time taken by kNN for N points of D dimensions

- time to compute distances: $O(ND)$
- time to find the k nearest neighbor
  - $O(k\,N)$: repeated minima
  - $O(N \log N)$: sorting
  - $O(N + k \log N)$: min heap
  - $O(N + k \log k)$: fast median
- Total time is dominated by distance computation

We can be faster if we are willing to sacrifice exactness

Practical issue when using kNN: Curse of dimensionality



#bins = 10x10

d = 2

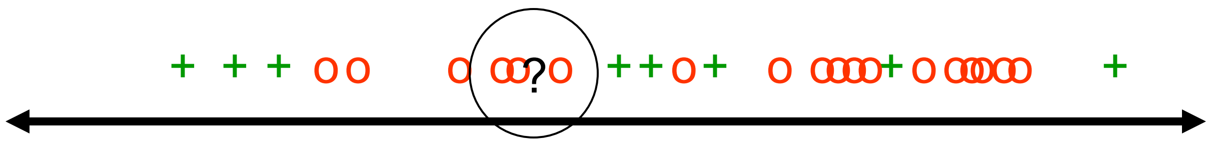
#bins = 10^d

d = 1000

Atoms in the universe: ~10^{80}

How many neighborhoods are there?

K-NN and irrelevant features



Nearest Neighbor

When to Consider

- Instance map to points in R^n
- Less than 20 attributes per instance
- Lots of training data

Advantages

- Training is very fast
- Learn complex target functions
- Do not lose information

Disadvantages

- Slow at query time
- Easily fooled by irrelevant attributes

KNN Advantages

- Easy to program
- No optimization or training required
- Classification accuracy can be very good; can outperform more complex models

Slides credit

Slides are closely following and adapted from Hal Daume's book and Subranshu Maji's course.

The fruit classification dataset is from Iain Murray at University of Edinburgh

http://homepages.inf.ed.ac.uk/imurray2/teaching/oranges_and_lemons/.

The slides on texture synthesis are from Efros and Leung's ICCV 2009 presentation.

Many images are from the Berkeley segmentation benchmark

http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds

Normalized cuts image segmentation:

http://www.timotheecour.com/research.html