

CS840a
Machine Learning in Computer
Vision
Olga Veksler

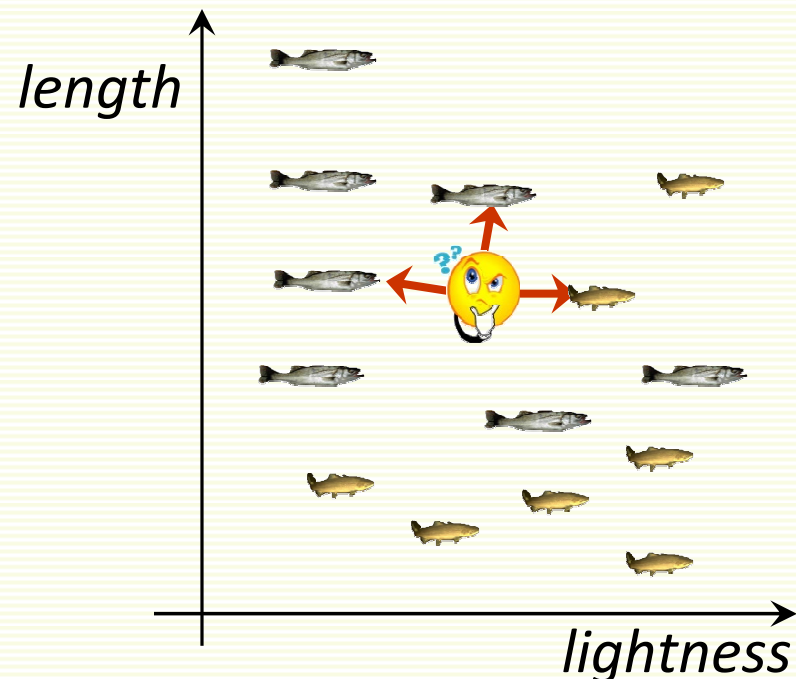
Lecture 2
k Nearest Neighbors

k-Nearest Neighbors

- classify an unknown example with the most common class among k closest examples
- *“tell me who your neighbors are, and I’ll tell you who you are”*

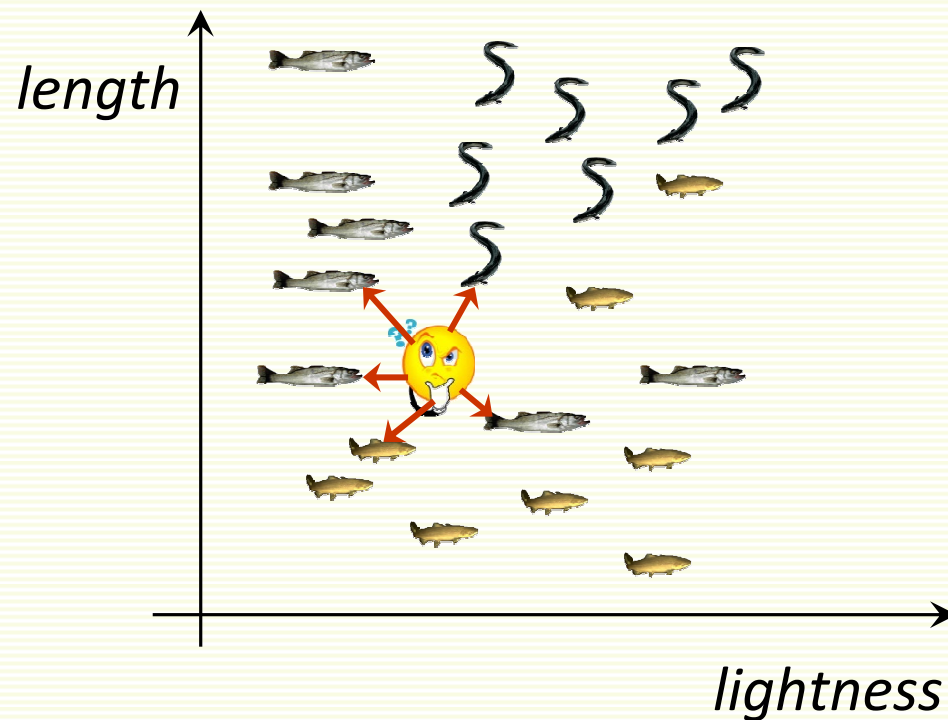
- Example:

- $k = 3$
- 2 sea bass, 1 salmon
- Classify as sea bass



kNN: Multiple Classes

- Easy to implement for multiple classes
- Example for $k = 5$
 - 3 fish species: salmon, sea bass, eel
 - 3 sea bass, 1 eel, 1 salmon \Rightarrow classify as sea bass



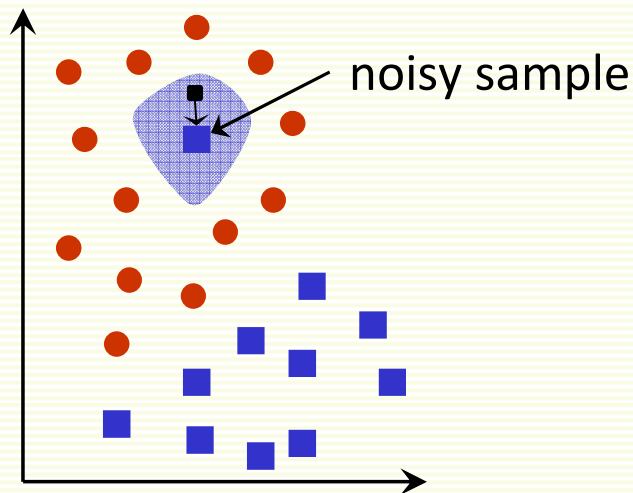
kNN: How to Choose k ?

- In theory, if infinite number of samples available, the larger is k , the better is classification
- The caveat is that all k neighbors have to be close
 - Possible when infinite # samples available
 - Impossible in practice since # samples is finite

kNN: How to Choose k?

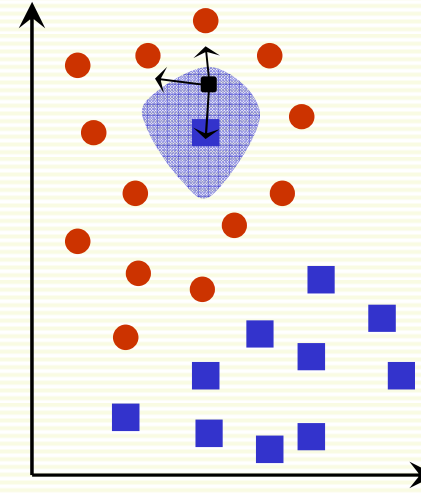
- Rule of thumb is $k < \sqrt{n}$, n is number of examples
- interesting theoretical properties
- In practice, $k = 1$ is often used for efficiency, but can be sensitive to “noise”

1 NN



every example in the blue shaded area will be misclassified as the blue class

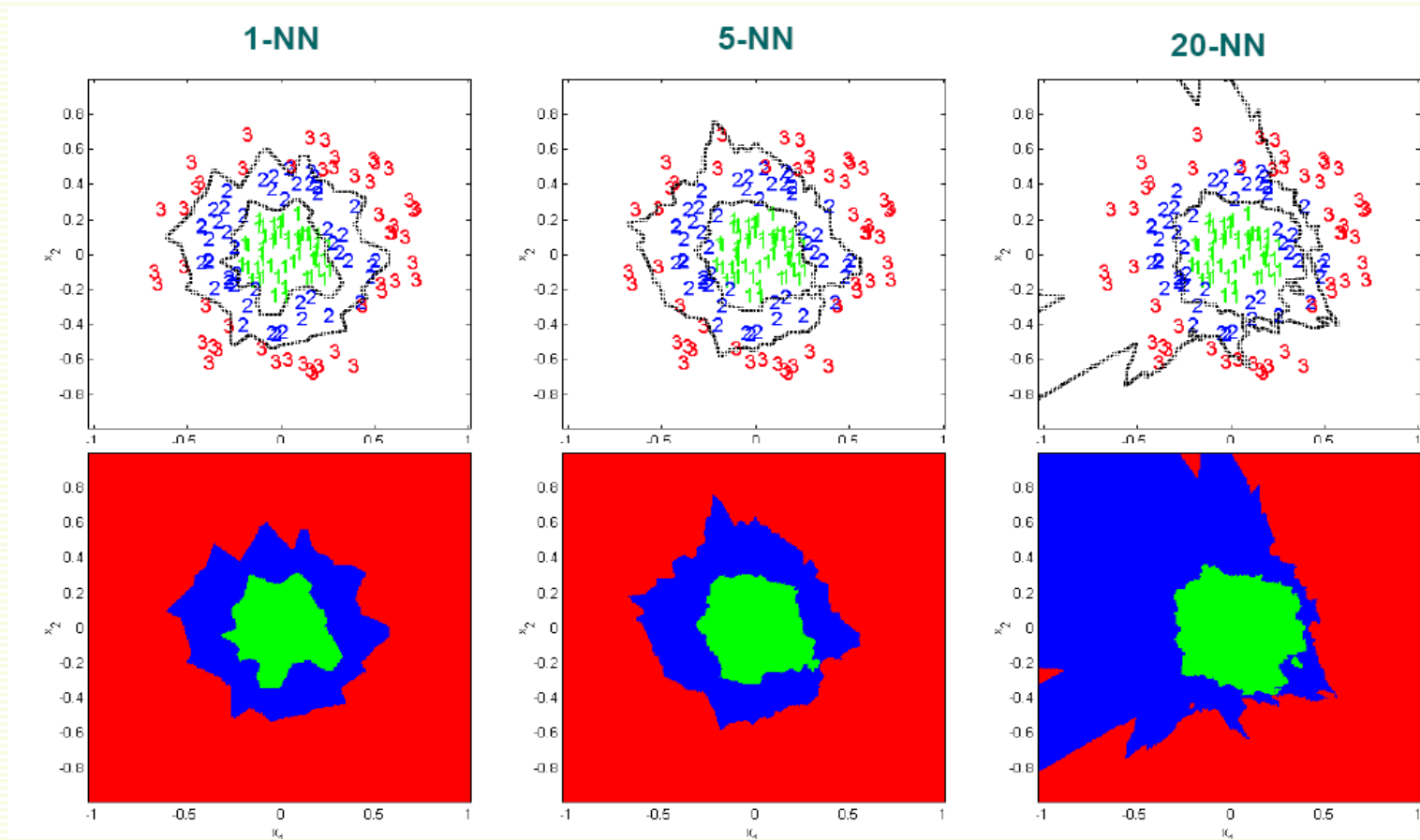
3 NN



every example in the blue shaded area will be classified correctly as the red class

kNN: How to Choose k ?

- larger k may improve performance, but too large k destroys *locality*, i.e. end up looking at samples that are not neighbors
- cross-validation (study later) may be used to choose k



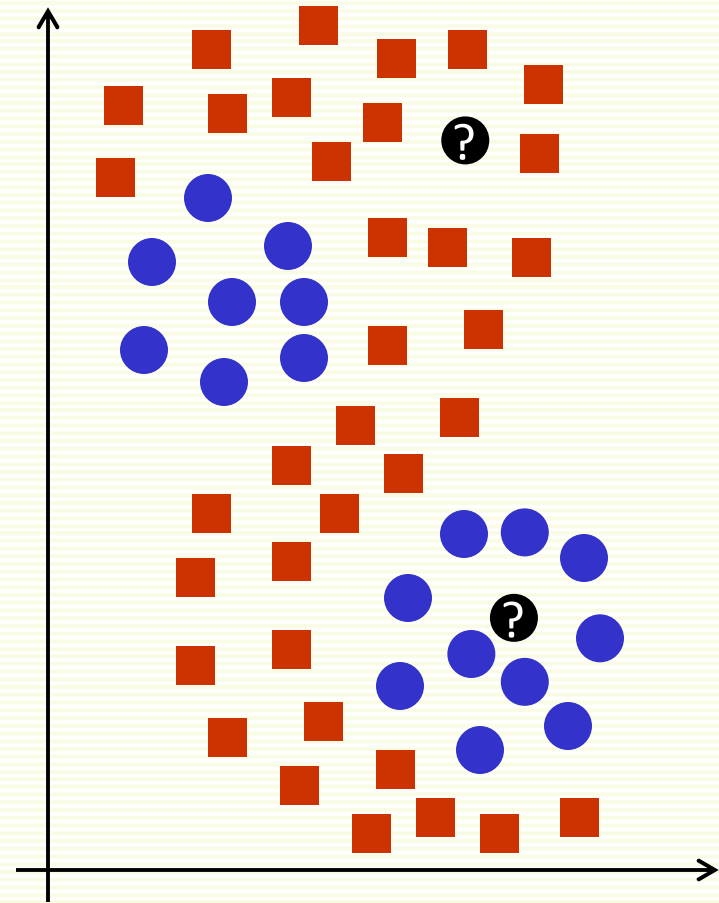
picture from R. Gutierrez-Osuna

kNN: How Well does it Work?

- kNN is simple and intuitive, but does it work?
- Theoretically, the best error rate is the Bayes rate E^*
 - Bayes error rate is the best (smallest) error rate a classifier can have, for a given problem, but we do not study it in this course
- Assume we have an unlimited number of samples
- kNN leads to an error rate greater than E^*
- But even for $k=1$, as $n \rightarrow \infty$, it can be shown that kNN error rate is smaller than $2E^*$
- As we increase k , the upper bound on the error gets better, that is the error rate (as $n \rightarrow \infty$) for the **kNN** rule is smaller than cE^* , with smaller c for larger k
- **If we have lots of samples, kNN works well**

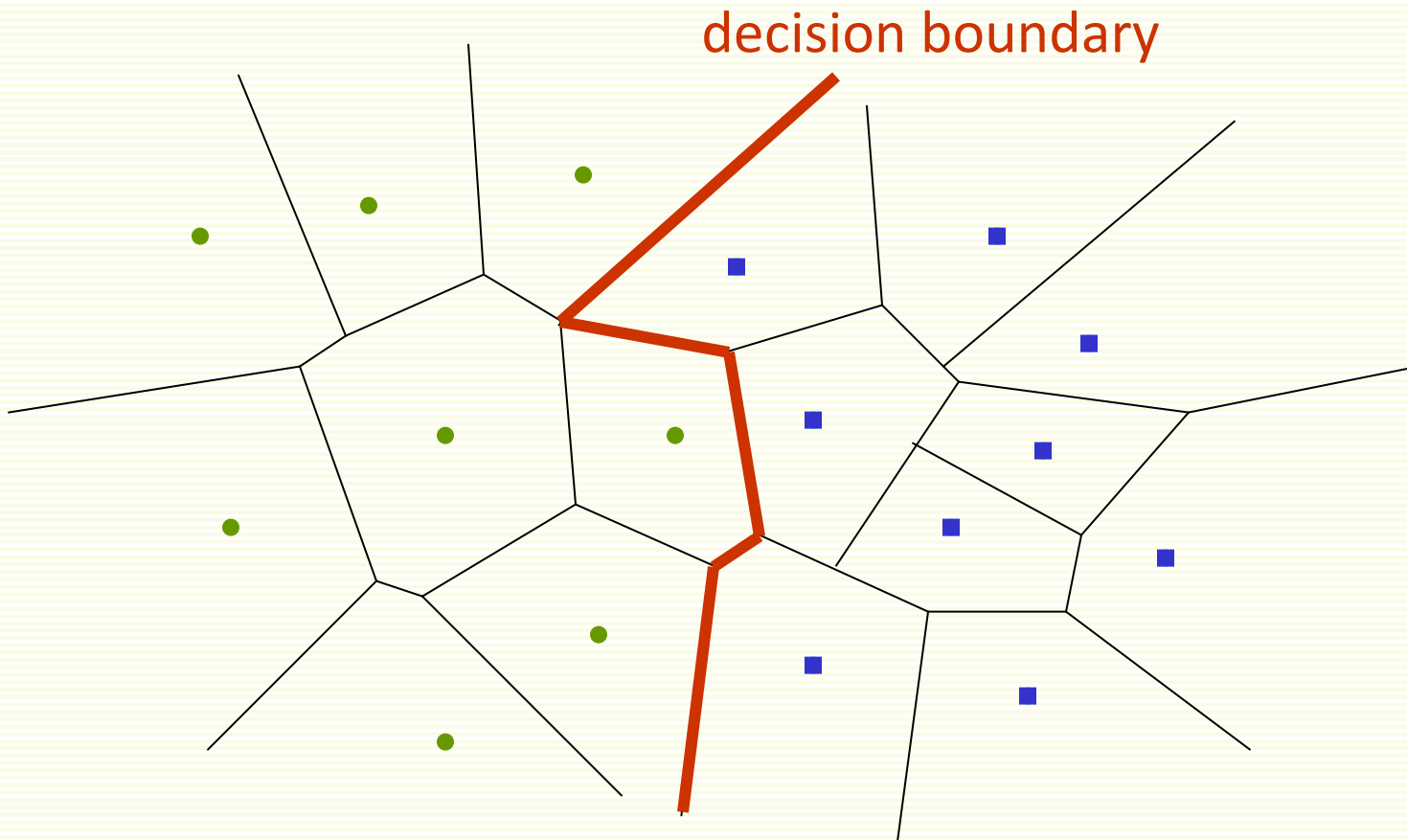
kNN: Multi-Modal Distributions

- Most parametric distributions would not work for this 2 class classification problem:
 -
- Nearest neighbors will do reasonably well, provided we have a lot of samples



1NN Visualization

- Voronoi tessellation is useful for visualization



kNN Selection of Distance

- So far we assumed we use Euclidian Distance to find the nearest neighbor:

$$D(a,b) = \sqrt{\sum_k (a_k - b_k)^2} = \sqrt{a \cdot b}$$

- Euclidean distance treats each feature as equally important
- However some features (dimensions) may be much more discriminative than other features

kNN Distance Selection: Extreme Example

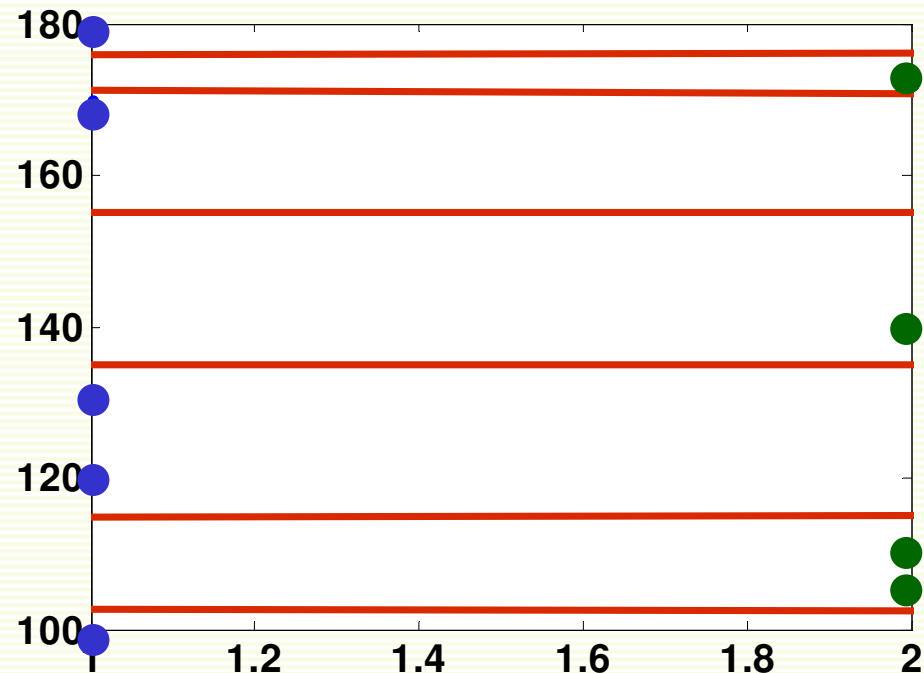
- feature 1 gives the correct class: 1 or 2
- feature 2 gives irrelevant number from 100 to 200
- dataset: **[1 150]**
[2 110]
- classify **[1 100]**

$$D\left(\begin{bmatrix} 1 \\ 100 \end{bmatrix}, \begin{bmatrix} 1 \\ 150 \end{bmatrix}\right) = \sqrt{(1-1)^2 + (100-150)^2} = 50$$

$$D\left(\begin{bmatrix} 1 \\ 100 \end{bmatrix}, \begin{bmatrix} 2 \\ 110 \end{bmatrix}\right) = \sqrt{(1-2)^2 + (100-110)^2} = 10.5$$

- **[1 100]** is misclassified!
- The denser the samples, the less of this problem
- But we rarely have samples dense enough

kNN Distance Selection: Extreme Example



- Decision boundary is in red, and is really wrong because
 - feature 1 is discriminative, but it's scale is small
 - feature 2 gives no class information but its scale is large, it dominates distance calculation

kNN: Feature Normalization

- Notice that 2 features are on different scales:
- First feature takes values between 1 or 2
- Second feature takes values between 100 to 200
- **Idea:** normalize features to be on the same scale
- Different normalization approaches
- Linearly scale the range of each feature to be, say, in range [0,1]

$$f_{new} = \frac{f_{old} - f_{old}^{\min}}{f_{old}^{\max} - f_{old}^{\min}}$$

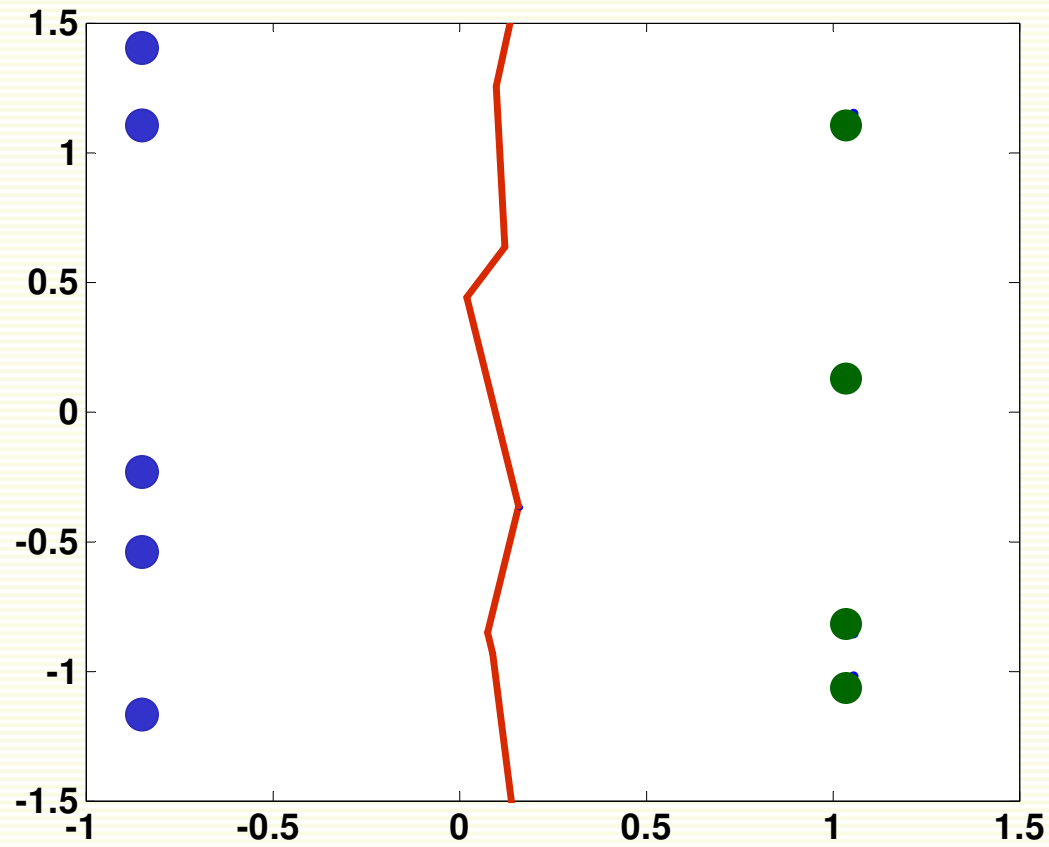
kNN: Feature Normalization

- Linearly scale to **0** mean variance **1**:
- If **Z** is a random variable of mean **m** and variance **σ^2** , then **$(Z - m)/\sigma$** has mean **0** and variance **1**
- For each feature **f** let the new rescaled feature be

$$f_{new} = \frac{f_{old} - \mu}{\sigma}$$

- Let us apply this normalization to previous example

kNN: Feature Normalization



kNN: Selection of Distance

- Feature normalization does not help in high dimensional spaces if most features are irrelevant

$$D(a,b) = \sqrt{\sum_k (a_k - b_k)^2} = \sqrt{\underbrace{\sum_i (a_i - b_i)^2}_{\text{discriminative features}} + \underbrace{\sum_j (a_j - b_j)^2}_{\text{noisy features}}}$$

- If the number of useful features is smaller than the number of noisy features, Euclidean distance is dominated by noise

kNN: Feature Weighting

- Scale each feature by its importance for classification

$$D(a, b) = \sqrt{\sum_k w_k (a_k - b_k)^2}$$

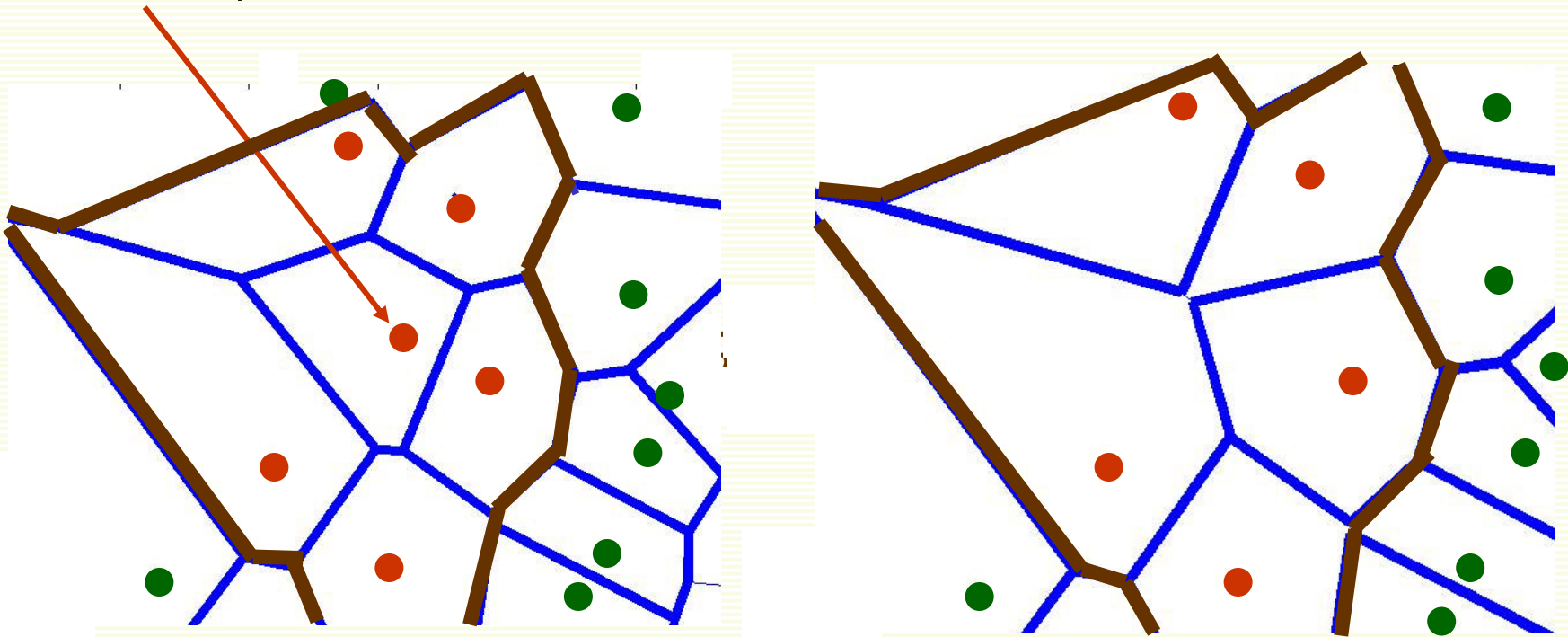
- Can use our prior knowledge about which features are more important
- Can learn the weights w_k using cross-validation (to be covered later)

kNN: Computational Complexity

- Basic kNN algorithm stores all examples
- Suppose we have n examples each of dimension d
- $O(d)$ to compute distance to one example
- $O(nd)$ to find one nearest neighbor
- $O(knd)$ to find k closest examples
- Thus total complexity is $O(knd)$
- Very expensive for a large number of samples
- But we need a large number of samples for kNN to work well!

Reducing Complexity: editing 1NN

- If all Voronoi neighbors have the same class, a sample is useless, remove it



- Number of samples decreases
- Decision boundary does not change

Reducing Complexity: Partial Distance

- Have current k closes samples
- Abort distance computation if partial distance is already greater than the full distance to the current k closest samples
- Advantages:
 - complexity decreases
 - we are guaranteed to find closes neighbor(s)
- Disadvantages:
 - how much complexity decreases depends on our luck and data layout

Reducing Complexity

- Other methods for reducing complexity
 - reduce dimensionality of the data
 - find projection to a lower dimensional space so that the distances between samples are approximately the same
 - PCA
 - Projection to a Random subspace
 - use smart data structures, like kd trees

kNN Summary

- Advantages
 - Can be applied to the data from any distribution
 - for example, data does not have to be separable with a linear boundary
 - Very simple and intuitive
 - Good classification if the number of samples is large enough
- Disadvantages
 - Choosing k may be tricky
 - Test stage is computationally expensive
 - No training stage, all the work is done during the test stage
 - This is actually the opposite of what we want. Usually we can afford training step to take a long time, but we want fast test step
 - Need large number of samples for accuracy