Machine Learning

K-Nearest Neighbor (kNN)

Many slides/pictures thanks to Öznur Taştan Okan\ others as noted.

Nearest Neighbor Classifiers

- Basic idea:
  - Label a given test sample according to the labels of its k nearest neighbors.

1. Compute distance from the test sample to all observations
2. Find k of the “nearest” samples
3. Classify according to the majority label of these nearest neighbors

  - Special case with k=1

Test sample

Training set

Nearest (k=1) Neighbor Algorithm

- The training data D = $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$

- **Learning step:**
  - Store training examples ☺
  - Called a lazy algorithm for this reason

- **Prediction step:**
  - Classify a new example **x** by finding the training example $(\mathbf{x}_i, y_i)$ that is nearest to **x**
  - Predict the class y as y_i

K-Nearest Neighbour(kNN)

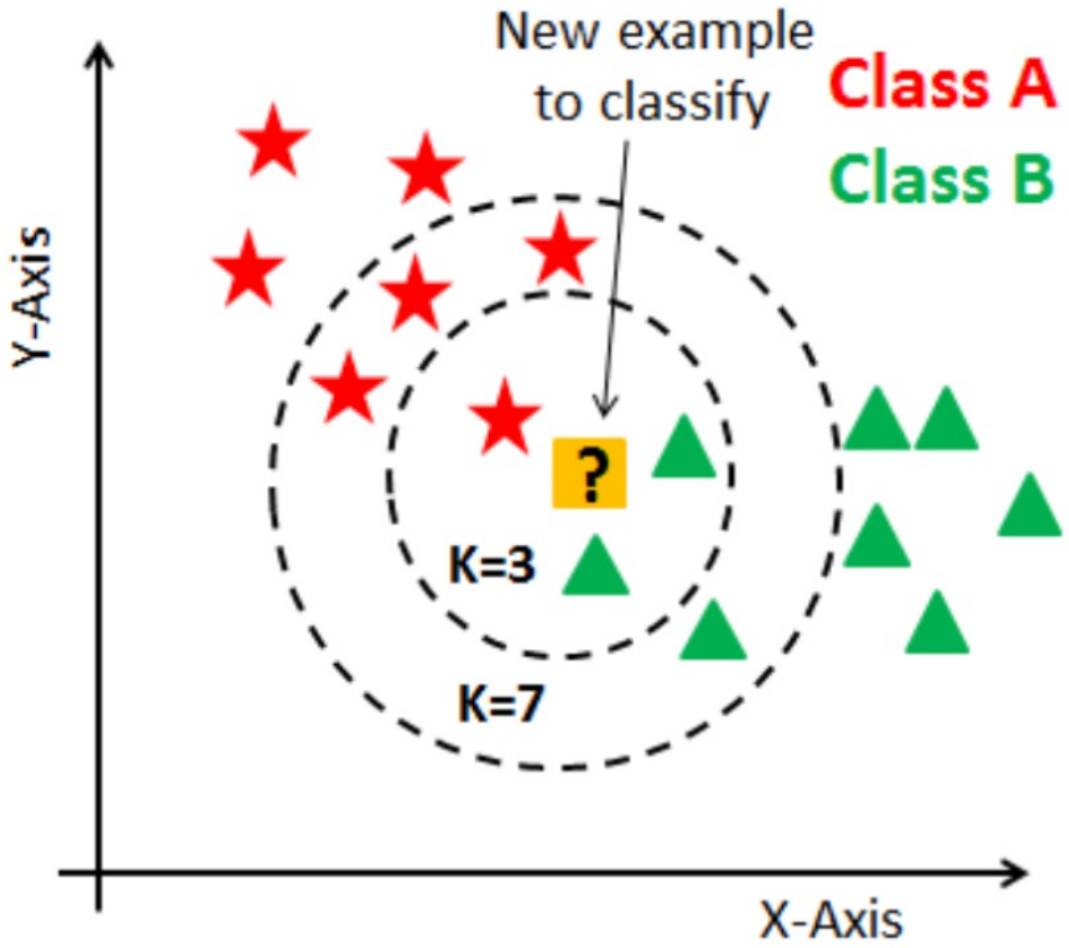


Photo by Sarang Anil Gotke on Kdnuggets

K-Nearest Neighbor (k-NN)

Formally:

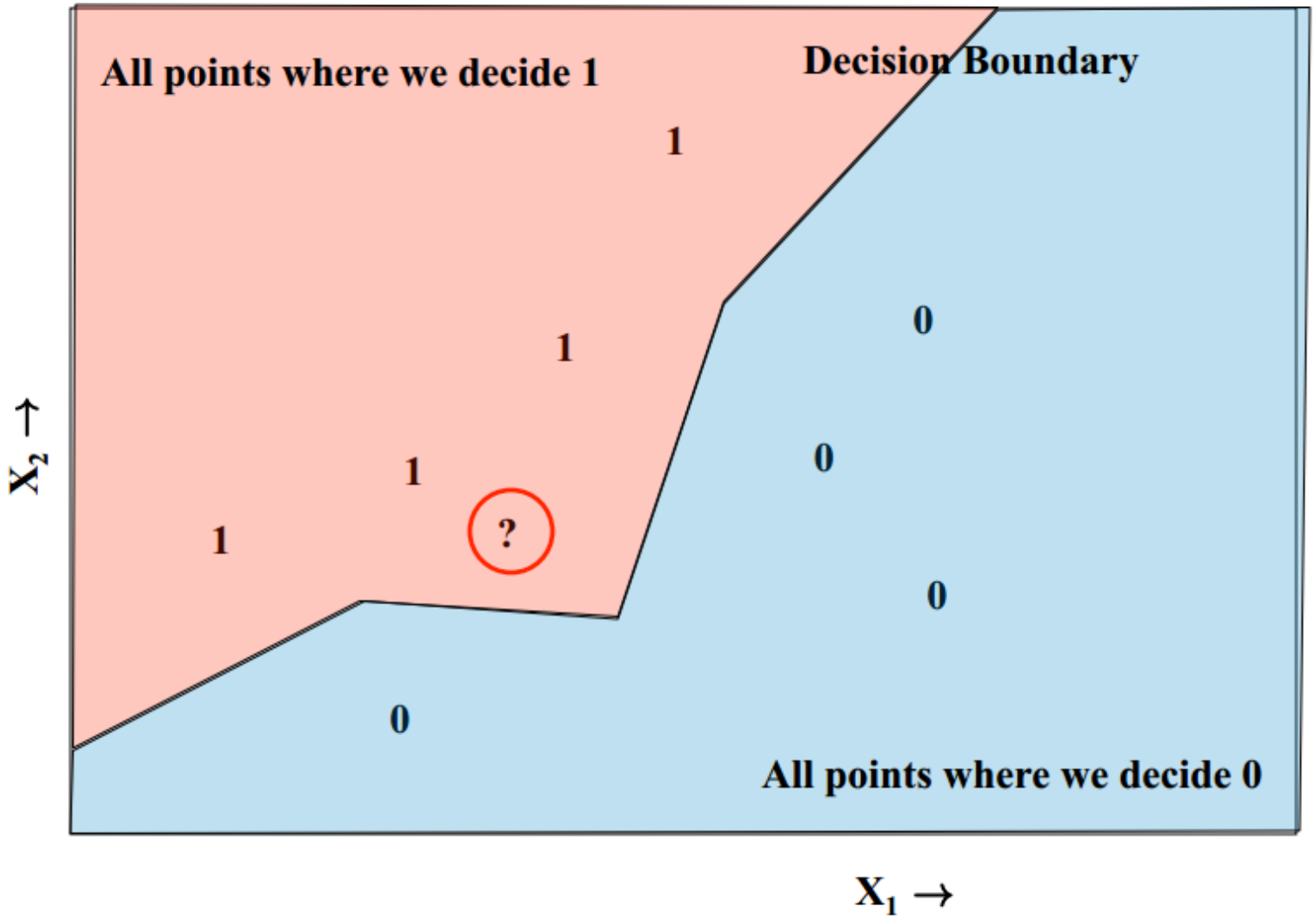
- Given training data $D=\{(\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_N,y_N)\}$ and a test sample **x**
- Find the k most-similar training examples
  - For classification, choose the **majority label** among those neighbors and assign to the test sample
  - For regression, choose the **average target** among those neighbors and assign to the test sample
- Algorithm requires meta-parameter k and a distance function to compute similarities between x and training samples

- Special case: 1-NN Nearest neighbor

Important Decisions

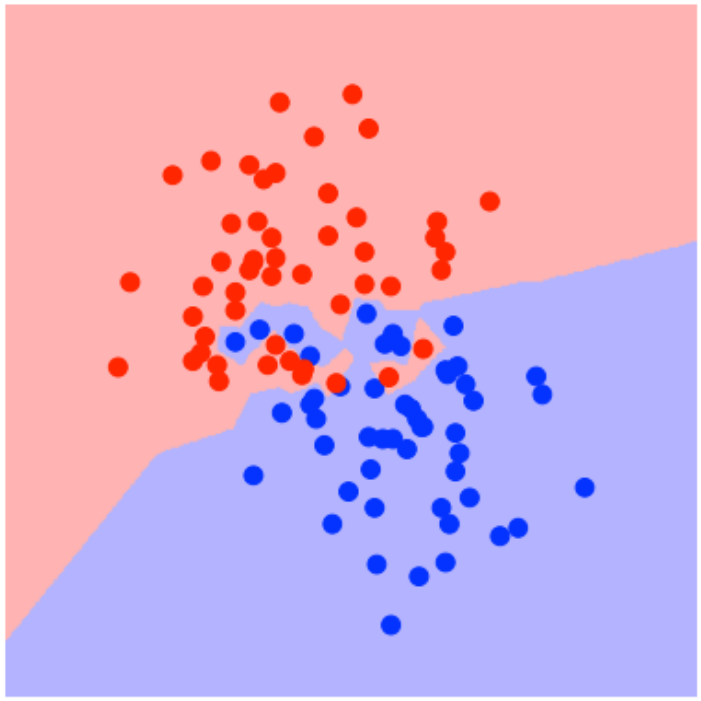
- Value of k
- Distance measure
- Voting mechanism

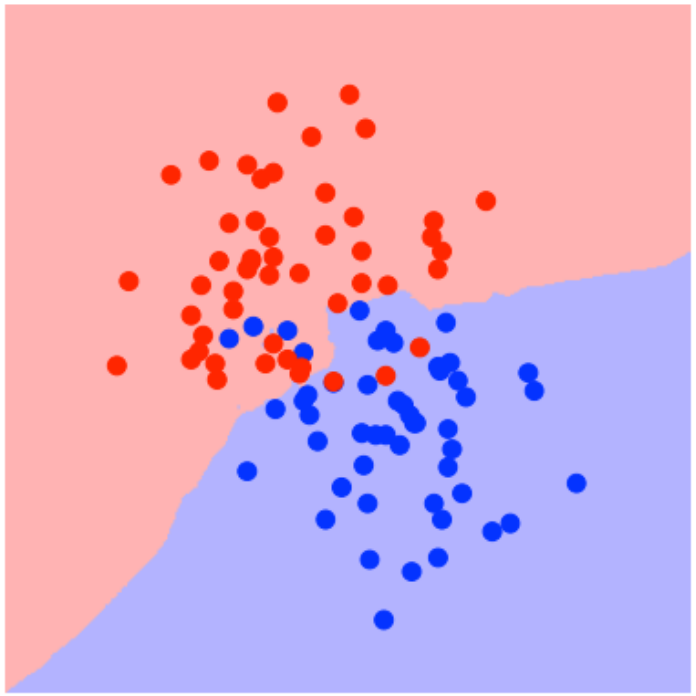
Classification – Decision Boundary



k-NN Decision Boundary

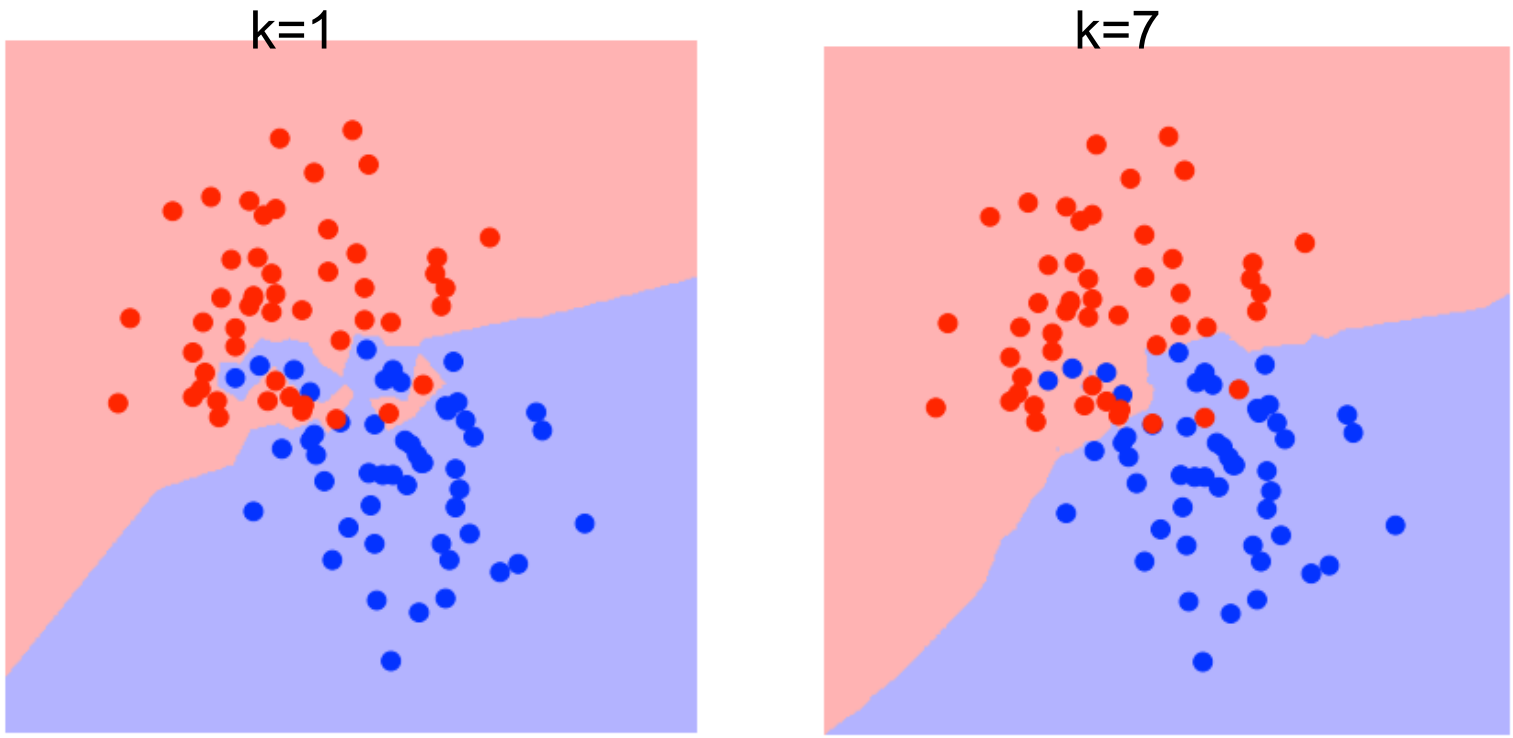
- How does the decision boundary changes with k?
  - Which one is k=1?
  - Which one is k=7?





k-NN Decision Boundary

- How does the decision boundary changes with k?
  - Which ones is k=1?
  - Which one is k=7?



k-NN Decision Boundary

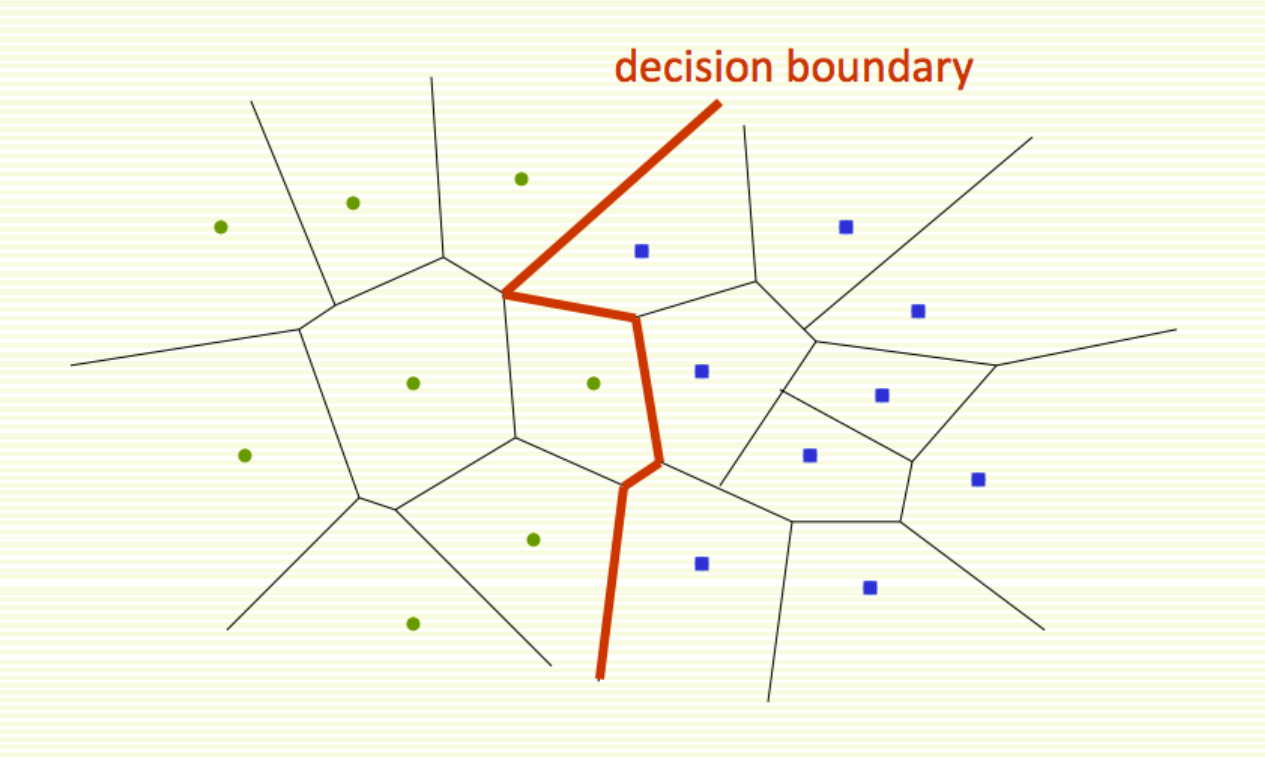
- Increasing *k* smooths the decision boundary
  - Smoother predictions, since we average over more data
  - Majority voting means less emphasis on individual points
- But could also be too smooth.
  - Think of the extreme case k = N, where N is number of training examples. What happens then?

Effect of k

- Very large value of k
  - Everything is classified as the most probable class
- Small value
  - Highly variable, small change to the training data will lead to large changes in the classifier

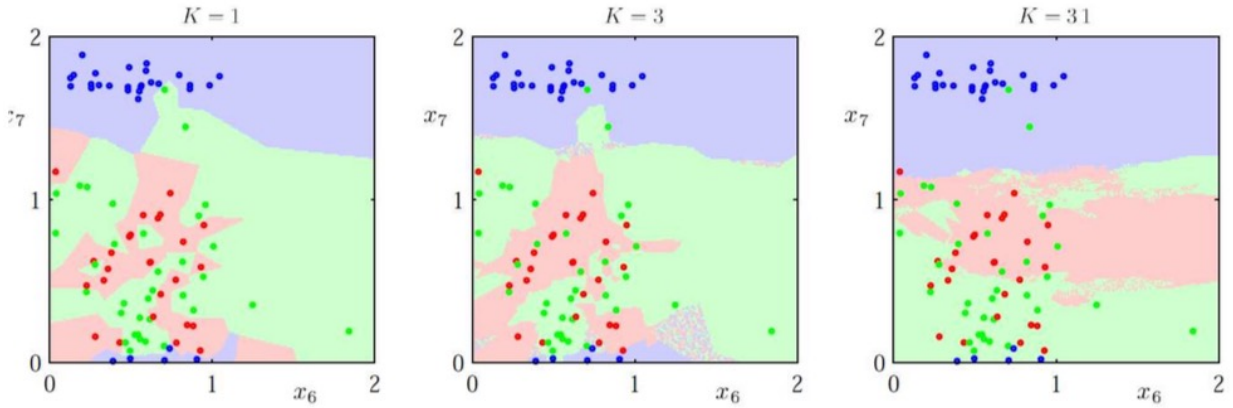
1- NN Decision Boundary

- The decision boundaries form a subset of the Voronoi diagram for the training data.
- Nearest-neighbor classifier produces piecewise linear decision boundaries



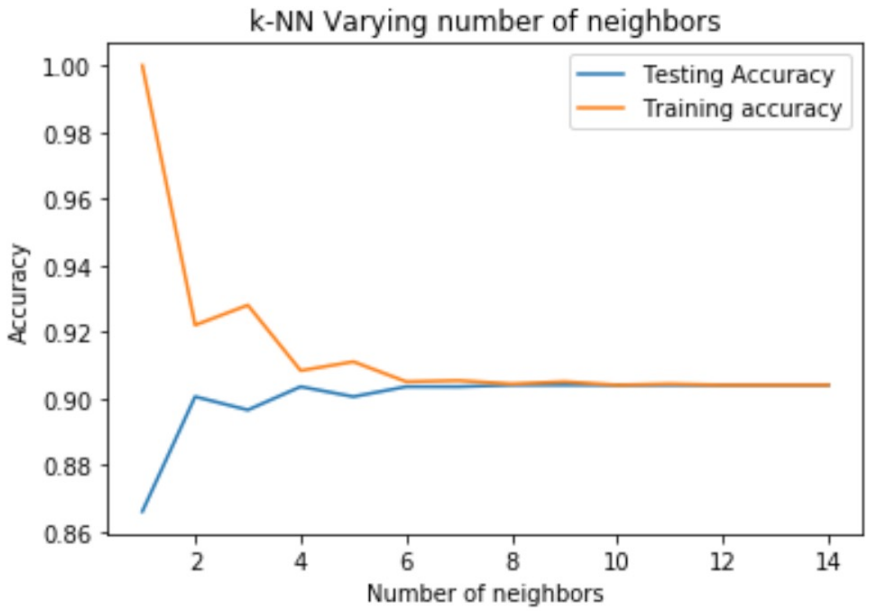
Choice of K

- Small K
  - Creates many small regions for each class
  - May lead to non-smooth decision boundaries and overfit
- Large K
  - Creates fewer larger regions
  - Usually leads to smoother decision boundaries (caution: too smooth decision boundary can underfit)
- Choosing K
  - Often data dependent and heuristic based
  - Or using cross-validation (using some held-out data)
  - In general, a K that is too small or too big is bad!



Selecting k

- Set aside a portion of your training data (validation set)
- Vary k, observe validation set error
- Pick k that gives best generalization performance as measured on the validation set



K-Nearest Neighbors

IMPORTANT DECISIONS

- Value of k (usually odd)
- Distance measure
- Voting mechanism

Distance Measures

- Key component of k-NN algorithm
  - Defines which examples are similar and which are not
  - Can have strong effect on performance

Distance - d(x,y)

- Numerical measure of how different (dissimilar) two data samples x and y are
  - A function that maps pairs of objects to real values
  - Lower when objects are more alike
- Minimum distance is 0, when comparing an object with itself.

Distance Metric

- A distance function d is a distance metric if it is a function from pairs of objects to real numbers such that:
  1. $d(x,y) \geq 0$ (non-negativity)
  2. $d(x,y) = 0$ iff $x = y$ (identity)
  3. $d(x,y) = d(y,x)$ (symmetry)
  4. $d(x,y) \leq d(x,z) + d(z,y)$ (triangle inequality)

Distances for Real Vectors

- Euclidean distance:

$$dist(x,y) = \sqrt{\sum_{i=1}(x_i - y_i)^2}$$

- Manhattan distance:

$$dist(x,y) = \sum_{i=1}|x_i - y_i|$$

- L_p norm or Minkowski distance:

$$dist(x,y) = (\sum_{i=1}|x_i - y_i|^p)^{1/p}$$

- The Euclidian distance between x and y is the L_2-norm of Δ = **x** – **y** (the difference vector)

- The Manhattan distance between x and y is the L_1-norm of Δ= **x** – **y** (the difference vector)

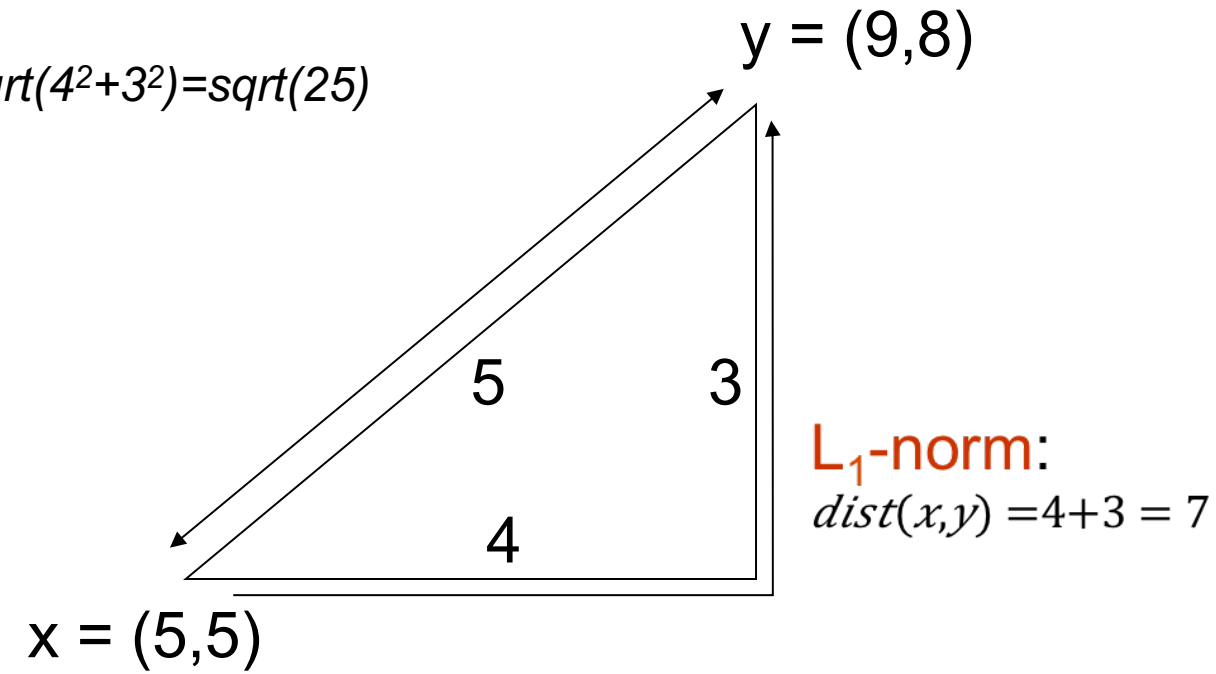
Viewed as Vector Norms

- l_2-norm $\|\mathbf{x}\|_2 = \sqrt{\sum_i |x_i|^2}$
- l_1-norm $\|\mathbf{x}\|_1 = \sum_i |x_i|$
- l_p-norm $\|\mathbf{x}\|_p = \sqrt[p]{\sum_i |x_i|^p}$

Example of Distances

L_2-norm:

dist(x,y) =sqrt(4^2+3^2)=sqrt(25)



L_1-norm:

$dist(x,y) = 4+3 = 7$

L_∞-norm:

$dist(x,y) = \max\{3,4\} = 4$

Distance Between Strings

- How do we define similarity between strings?

weird	wierd
intelligent	unintelligent
Athena	Athina

- Important for recognizing and correcting typing errors and analyzing DNA sequences

Hamming Distance

- Hamming distance is the number of positions in which bit-vectors differ.
  - Example: p_1 = 10**1**0**1**
    p_2 = 10**0**1**1**

    $d(p_1, p_2) = 2$ because the bit-vectors differ in the 3rd and 4th positions.

- Hamming distance between two vectors of categorical attributes is the number of positions in which they differ.
  - Example: p_1 = (red, tall, heavy),
    p_2 = (green, tall, light)
    $d(p_1, p_2) = 2$

Scale Issues

- Attributes may have to be scaled/normalized to prevent distance measures from being dominated by one of the attributes

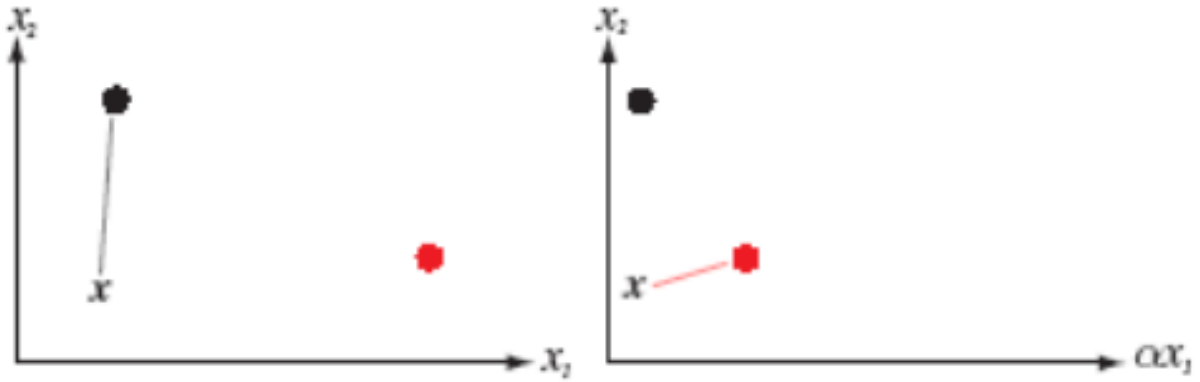
- Example:
  - height of a person may vary from 1.5 m to 1.8 m
  - weight of a person may vary from 40kg to 200kg
  - income of a person may vary from 1,000 to 10,000TL

Scale Issues

- For real-valued feature vectors, we can use Euclidean distance

$$D(u,v)^2 = ||u-v||^2 = (u-v)^T(u-v) = \sum_{i=1}^{d}(u_i - v_i)^2$$

- If we scale x1 by 1/3, NN changes!



Scaling/Normalizing Features

For a feature $x \in \mathcal{R}$:

- **Linear scaling to unit range**
  - Given a lower and upper bound, [a,b]

$$\tilde{x} = \frac{x - a}{b - a}$$

- **Linear scaling to unit variance**
  - Transform to zero mean and unit variance as

$$\tilde{x} = \frac{x - \mu}{\sigma}$$

where μ is the sample mean and σ is the sample standard deviation of the feature.

IMPORTANT DECISIONS

- Value of k (usually odd)
- Distance measure
- Voting mechanism

Combining of Neighbor Labels

- Options for determining the class from nearest neighbor list
  - Take **majority** vote of class labels among the *k*-nearest neighbors
  - **Weight** the votes according to distance
    - example: weight factor $w = 1 / d^2$

Summary

- Pros:
  - **Simple and intuitive; easily implementable**
  - *Asymptotically consistent:*
    - **With infinite training data and large enough K, K-NN approaches the best possible classifier (''Bayes optimal'')**
- Cons:
  - **Store all the training data in memory even at test time**
    - Can be memory intensive for large training datasets
    - An example of non-parametric, or memory/instance-based methods
      in contrast to parametric, model-based learning models
  - **Expensive at test time: O(ND) computations for each test point!**
    - Have to search through all training data to find nearest neighbors
    - Distance computations with N training points (D features each)
  - **Sensitive to noisy features**
  - **May perform badly in high dimensions (curse of dimensionality)**
    - In high dimensions, distance notions can be counter-intuitive!

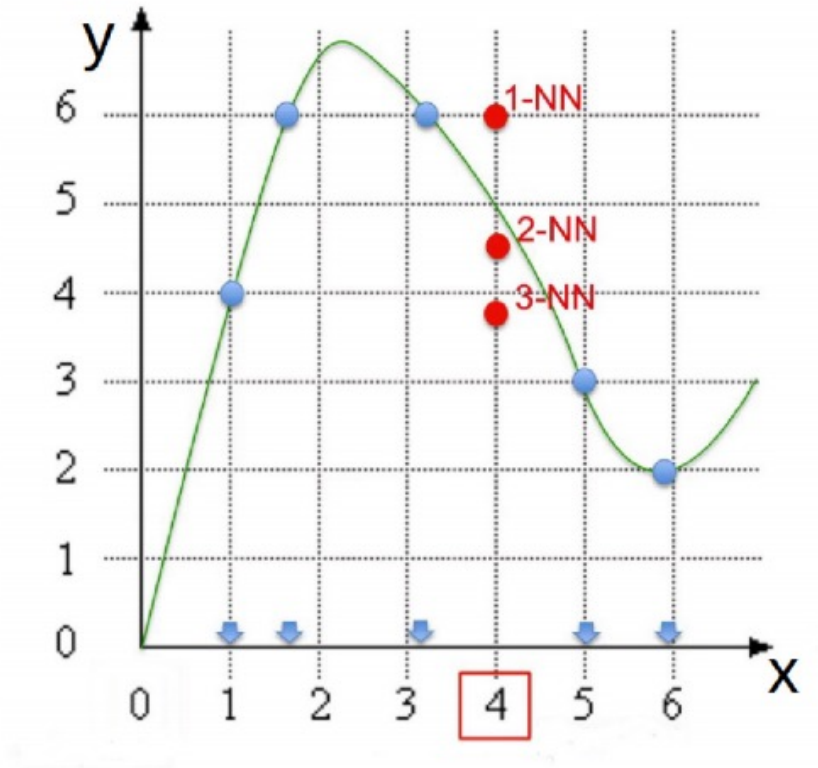
K-NN REGRESSION

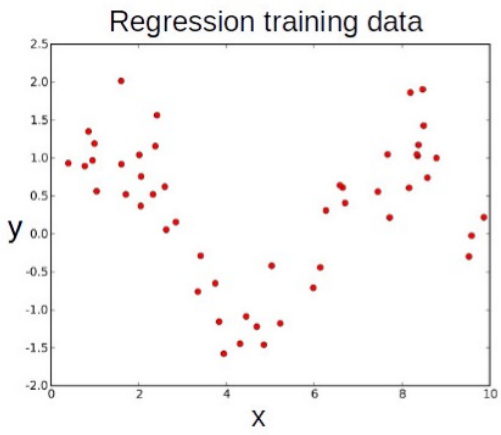
k-NN for regression

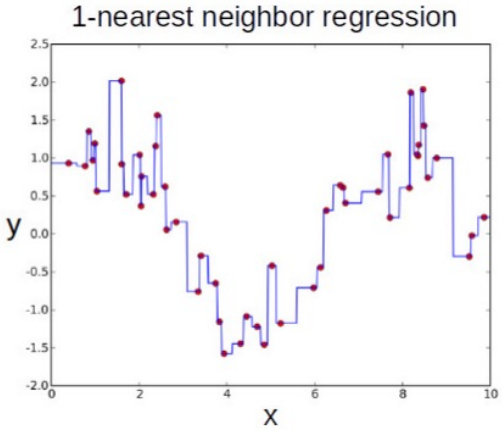
- As in classification, we find the label of the k nearest data points
- But instead of using the majority label, we will take the average of the labels of the k neighbors

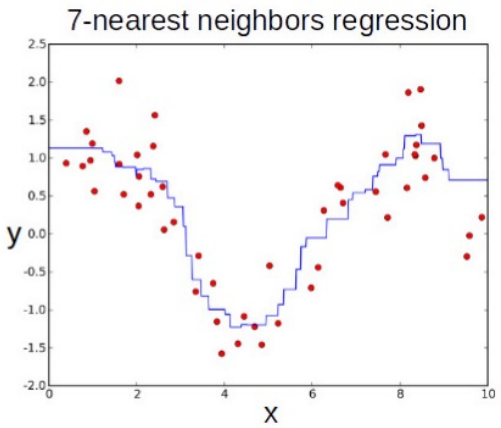
k-NN for regression

- Query is x=4.
  - When k=1, we find the point to the left as closest (around x=3.2). Thus, the predicted y(4)=6.
  - When k=2, we find the points to the left and right closest (around x=3.2 and x=5). Thus, the predicted y(4)=average(6,3).
  - Thus, the 3 red dots are the predicted y values for x=4, when k=1..3




Regression training data
y
x


1-nearest neighbor regression
y
x


7-nearest neighbors regression
y
x