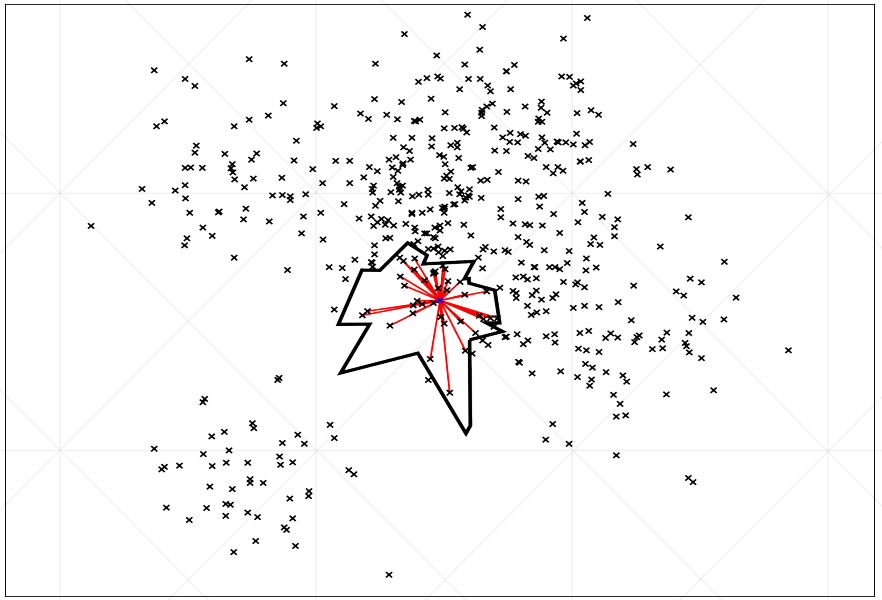


Nearest Neighbor Method

~Abhishek Kumar

Overview

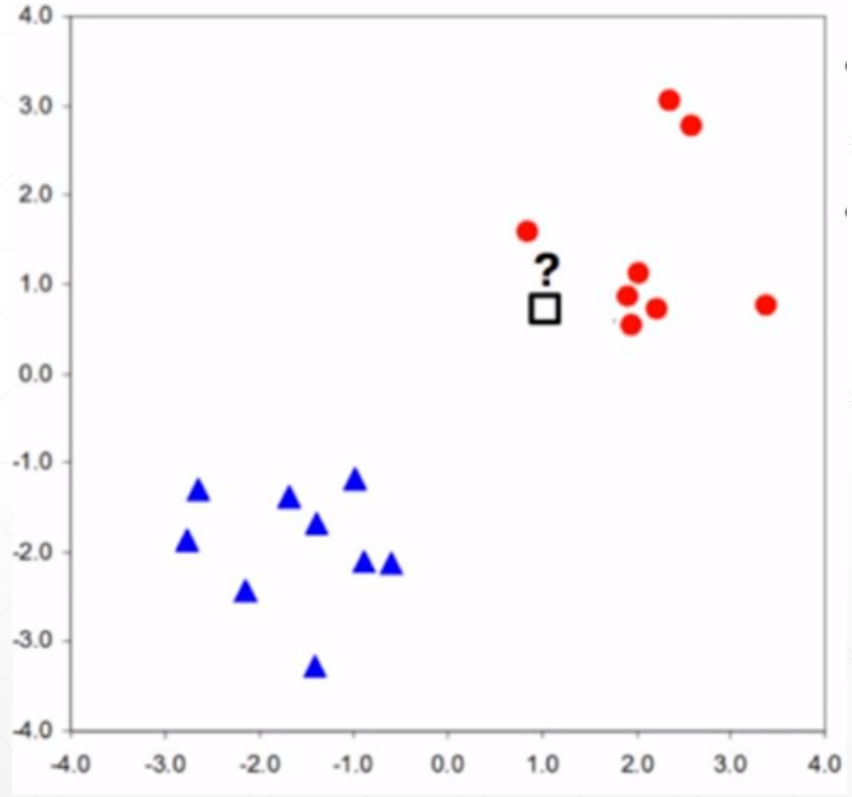
- What is Nearest Neighbor method?
  - Classification and regression
  - Practical issues: k distance, ties, missing values
  - Optimality
  - Assumptions
- Making KNN fast
  - K-D trees
  - Inverted indices
  - fingerprinting

Nearest Neighbor Method

- Supervised
- Classification algorithm

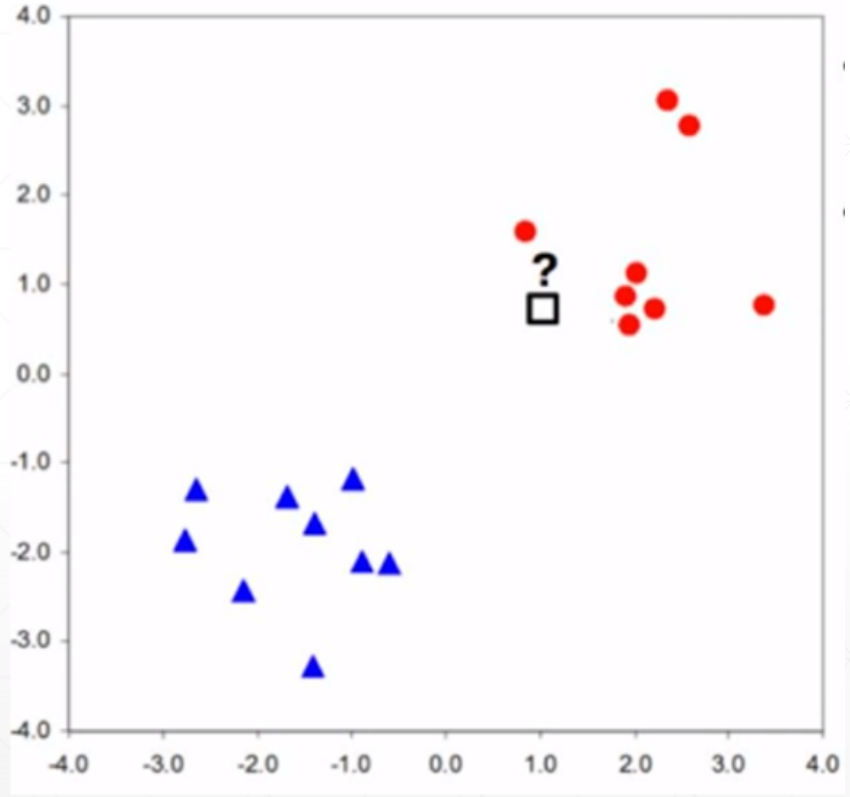
Intuition for kNN

- Set points (x,y)
  - -two classes
- Is the box red or blue?

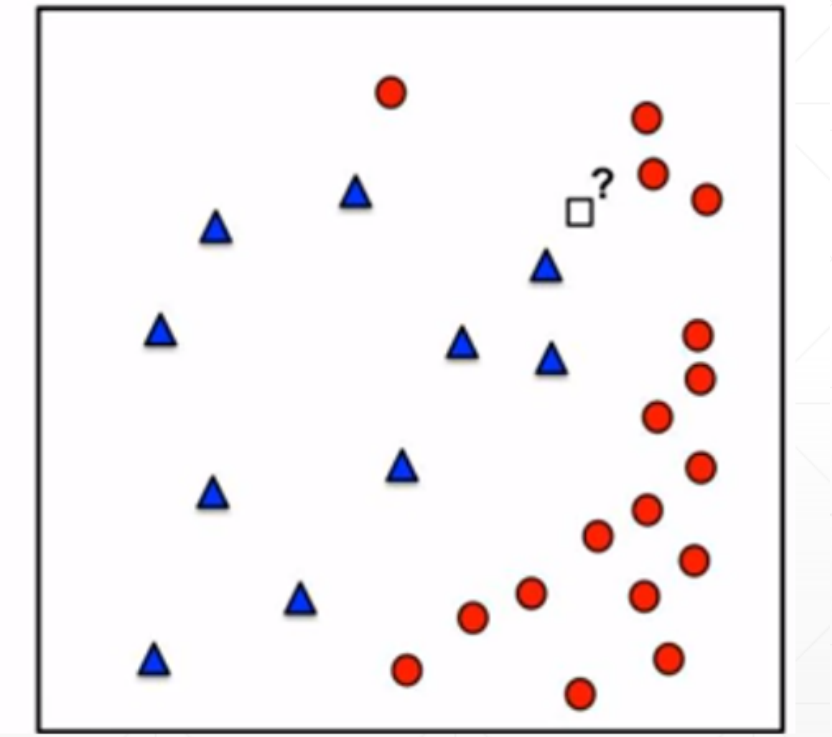


Intuition for kNN

- Set points (x,y)
  - -two classes
- Is the box red or blue?
  - How did you do it?
  - Use Bayes rule
  - Or fit a decision boundary
- Nearby points are red
  - Use this as a basis for a learning algorithm

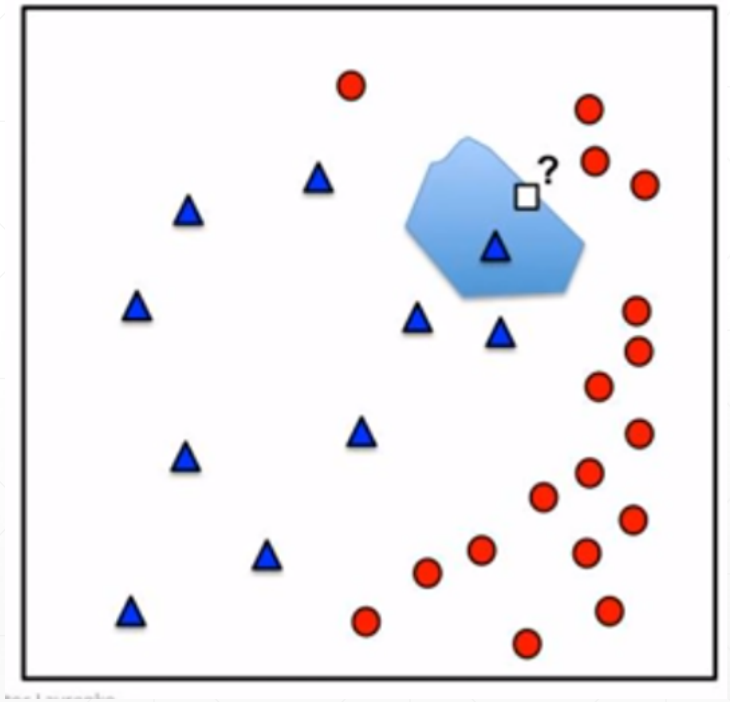


Nearest Neighbor Classification



Nearest Neighbor Classification

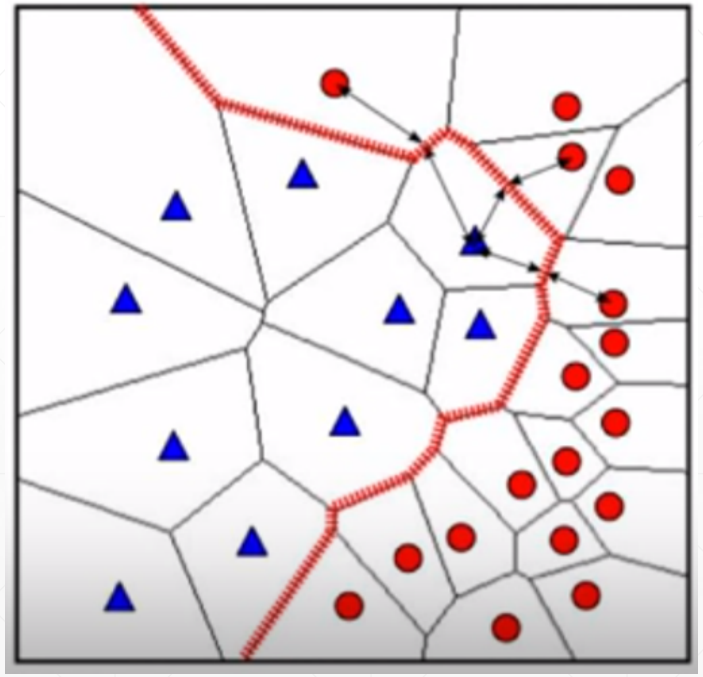
- Use the intuition to classify a new point x:
  - find the most similar training example x’
  - Predicts its class y’



Nearest Neighbor Classification

- Voronoi tesselation
  - partition into non overlapping regions where each region is dominated by one training example
  - B ig if data is sparse.
- classification boundary:
  - Broken line that follows voronoi cells
  - Non linear, reflects classes well
  - Compare to NB, Dt, logistic
  - Impressive for simple method
  - Simplicity comes at cost – flexible methods –

  -- leads to overfitting

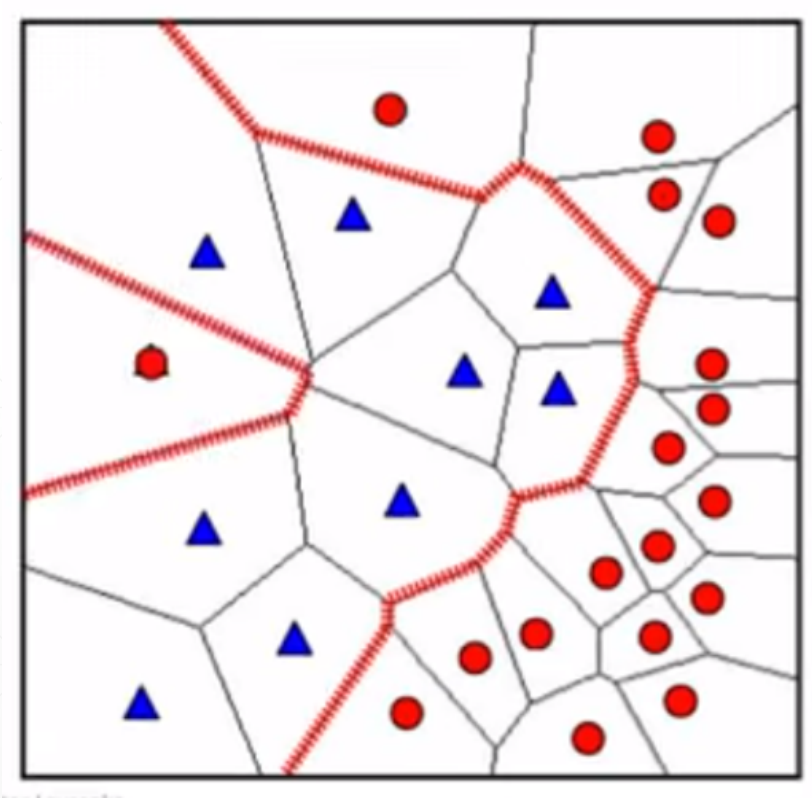


Nearest Neighbor: Outliers

Mislabeled data

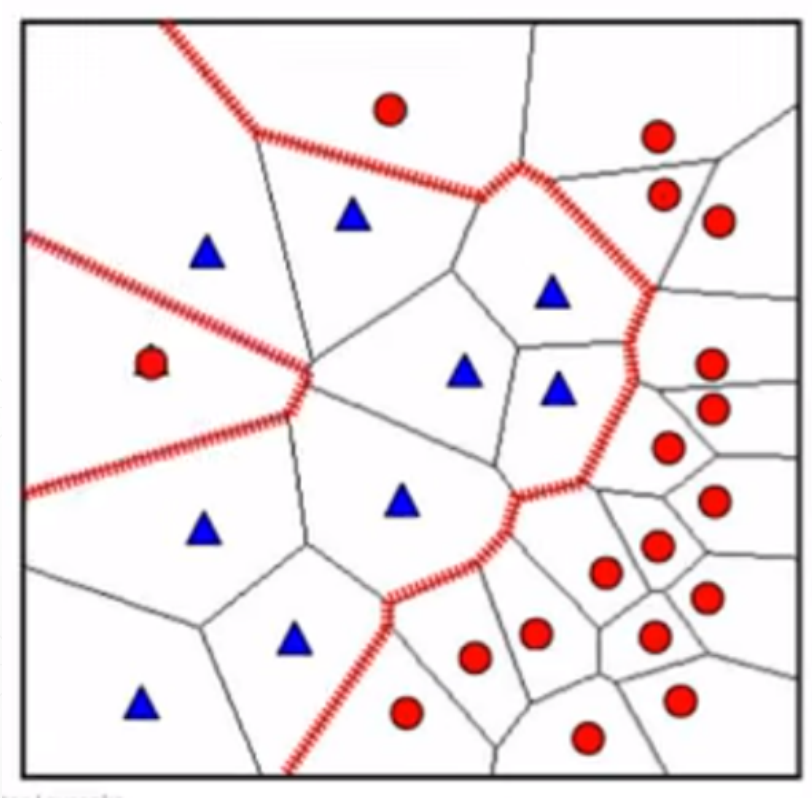
DB changes very rapidly

Not desirable that small changes shouldn't affect the classifier that much



Nearest Neighbor: Outliers

- Algorithm is sensitive to outliers
  - Single mislabeled example dramatically changes boundary
- No confidence P(x/y)
- Insensitive to class prior
- Idea:
  - Use more than one nearest neighbor to make the decision
  - Count class label in k most similar training example :
    - Many “triangles” will outweigh single inner “circle”

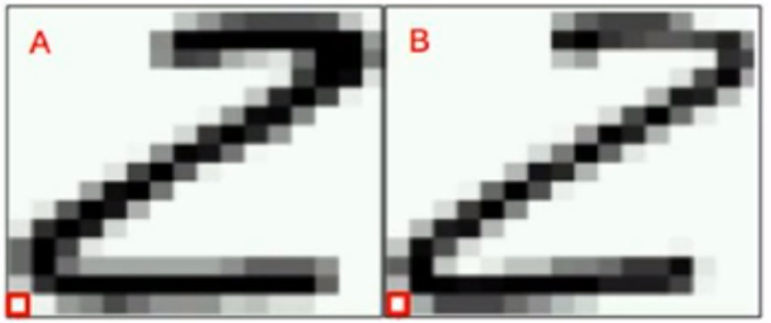


kNN classification algorithm

- Given
  - - training example(x,y)
    - x …attribute – value representation of examples
    - y ...class label: {ham,spam}, digit{0,1,…9} etc
  - -testing point x that we want to classify
- Algorithm:
  - Compute distance $D(x,x_i)$ to every training example x_i
  - Select k closest instances $x….x_i$ and their label $y…y_i$
  - Output the y^* which is most frequent in $y….y_i$

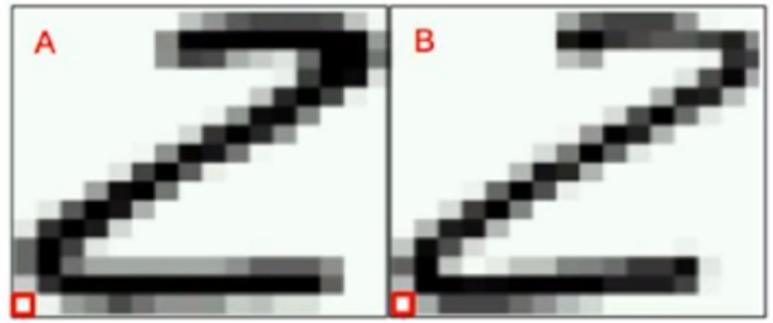
Example: handwritten digits

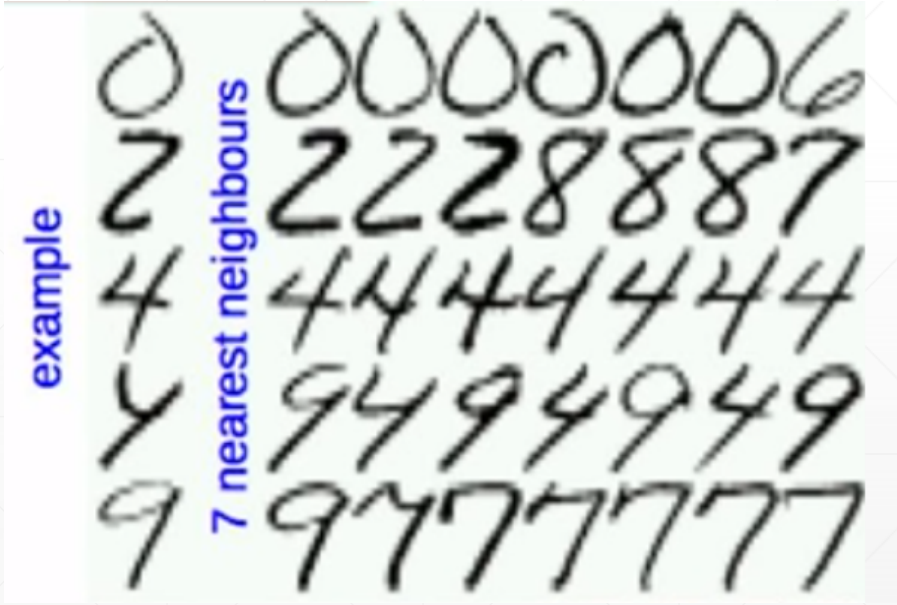
- 16 X 16 bitmaps
- 8-bit grayscale
- Euclidean distance
  - - over raw pixels
- Accuracy:
  - -- 7-NN ~ 95.2%
  - -- SVM ~ 95.8%
  - -- humans ~ 97.5%



Example: handwritten digits

- 16 X 16 bitmaps
- 8-bit grayscale
- Euclidean distance
  - - over raw pixels
- Accuracy:
  - -- 7-NN ~ 95.2%
  - -- SVM ~ 95.8%
  - -- humans ~ 97.5%

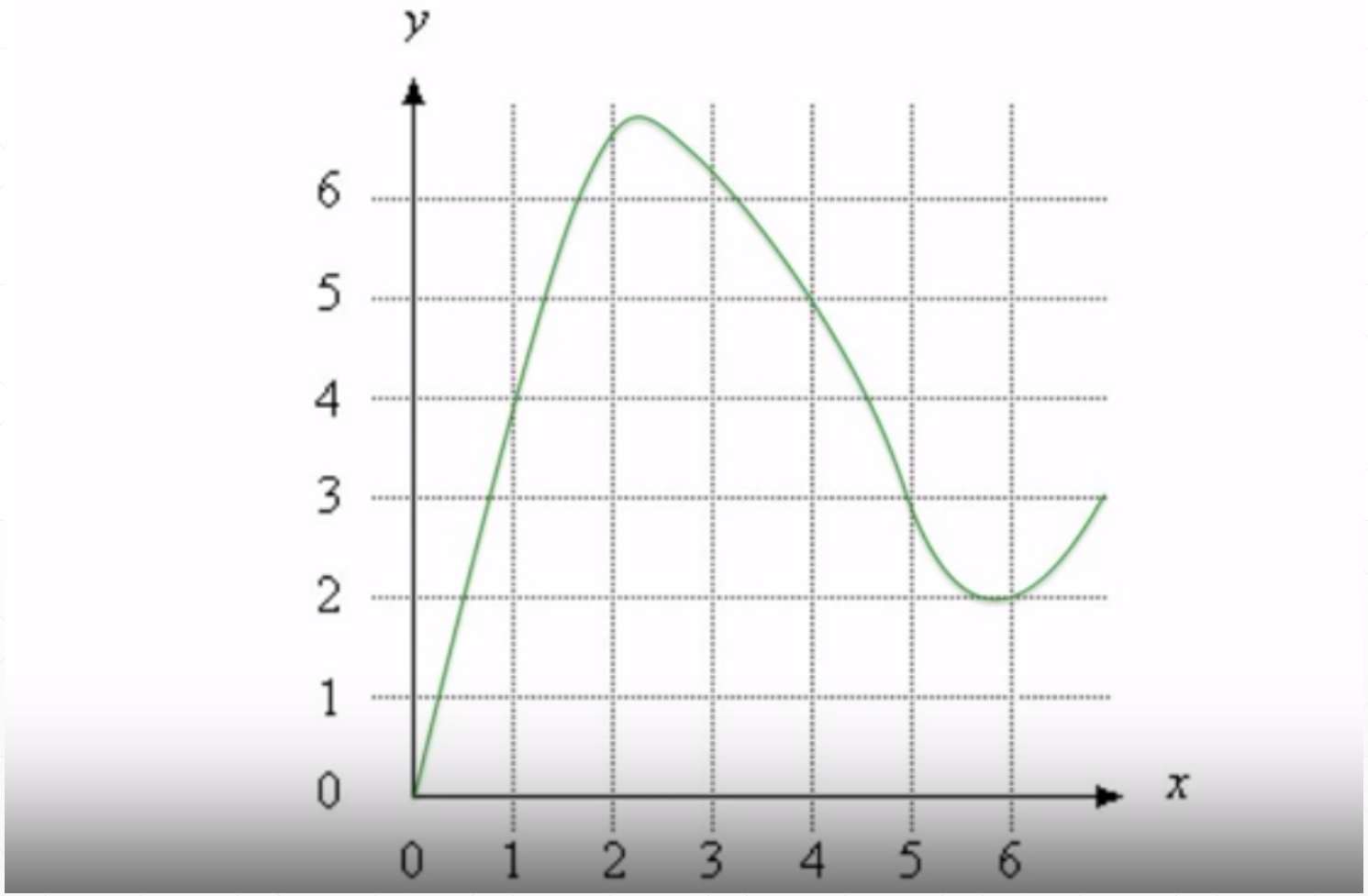




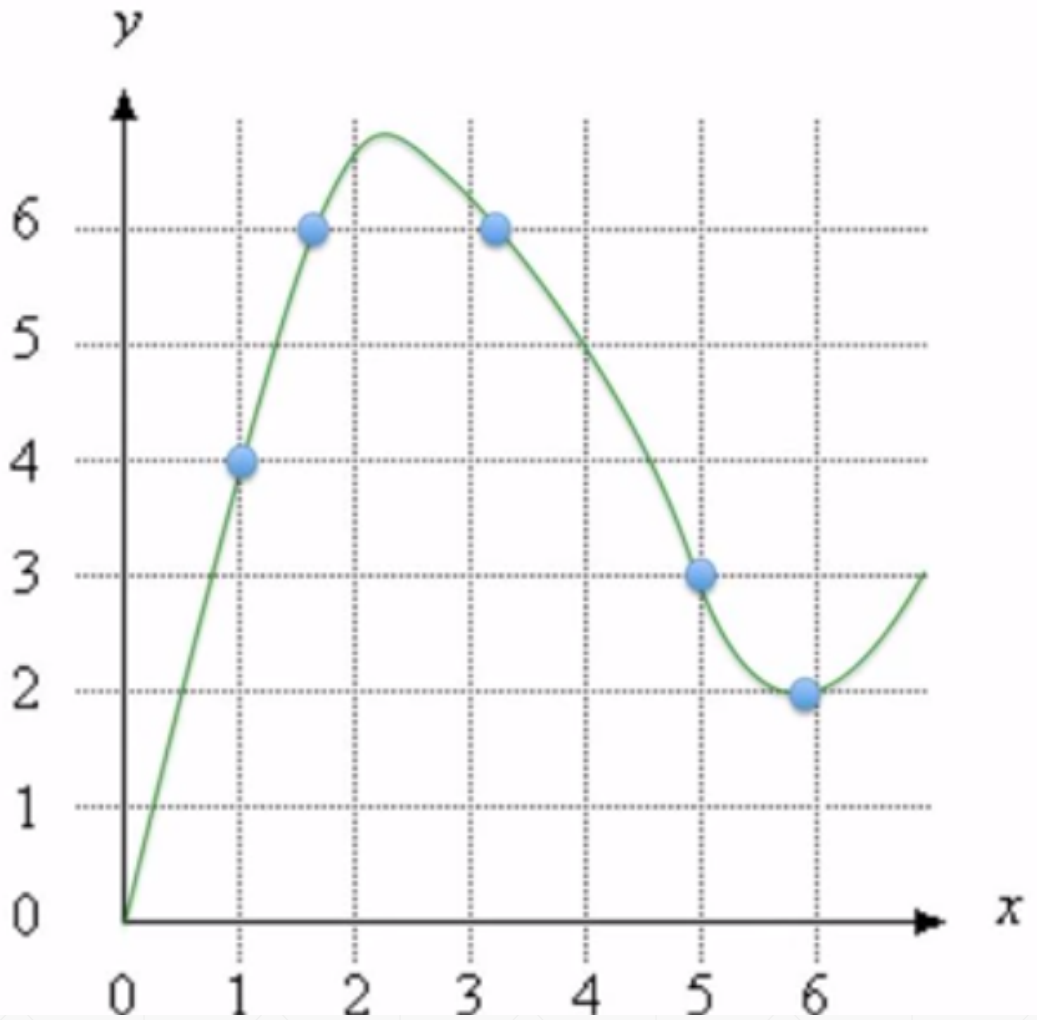
kNN regression algorithm

- Given
  - - training example(x,y)
    - x …attribute – value representation of examples
    - y ...real valued target
  - -testing point x that we want to predict the target
- Algorithm:
  - Compute distance $D(x,x_i)$ to every training example x_i
  - Select k closest instances $x….x_i$ and their labels $y…y_i$
  - Output the mean of $y….y_i$

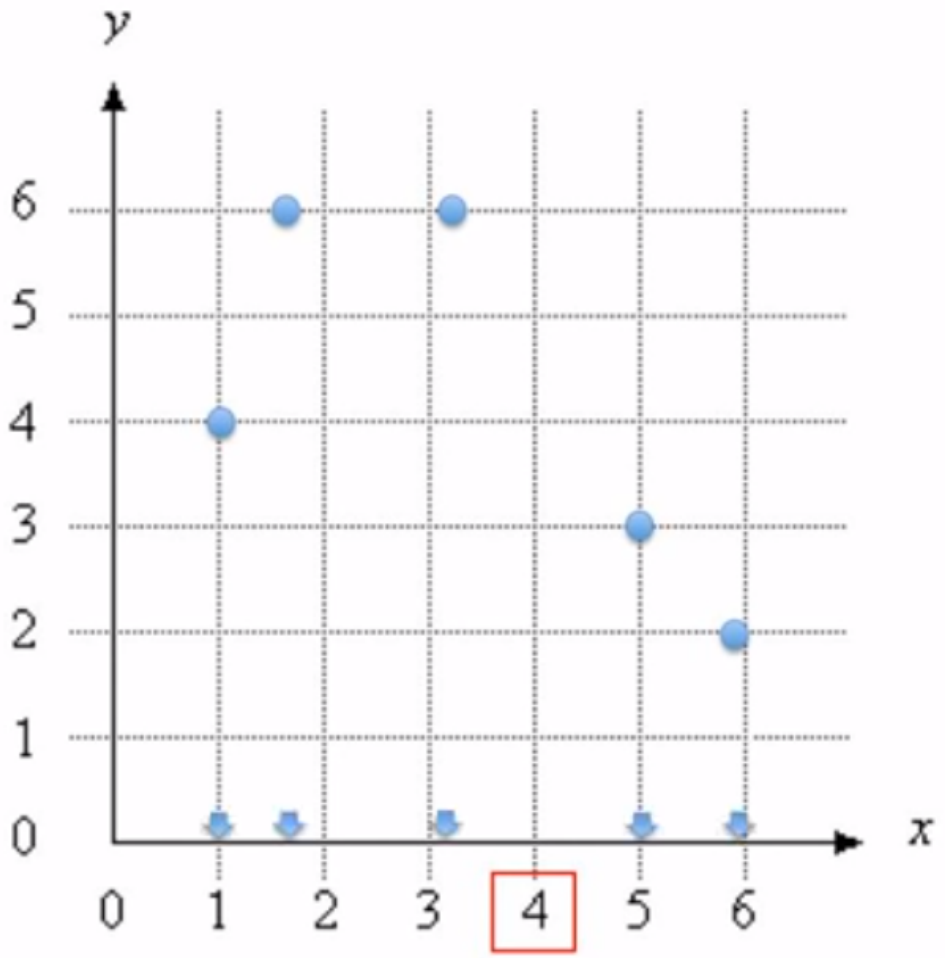
Example: kNN regression in 1-d



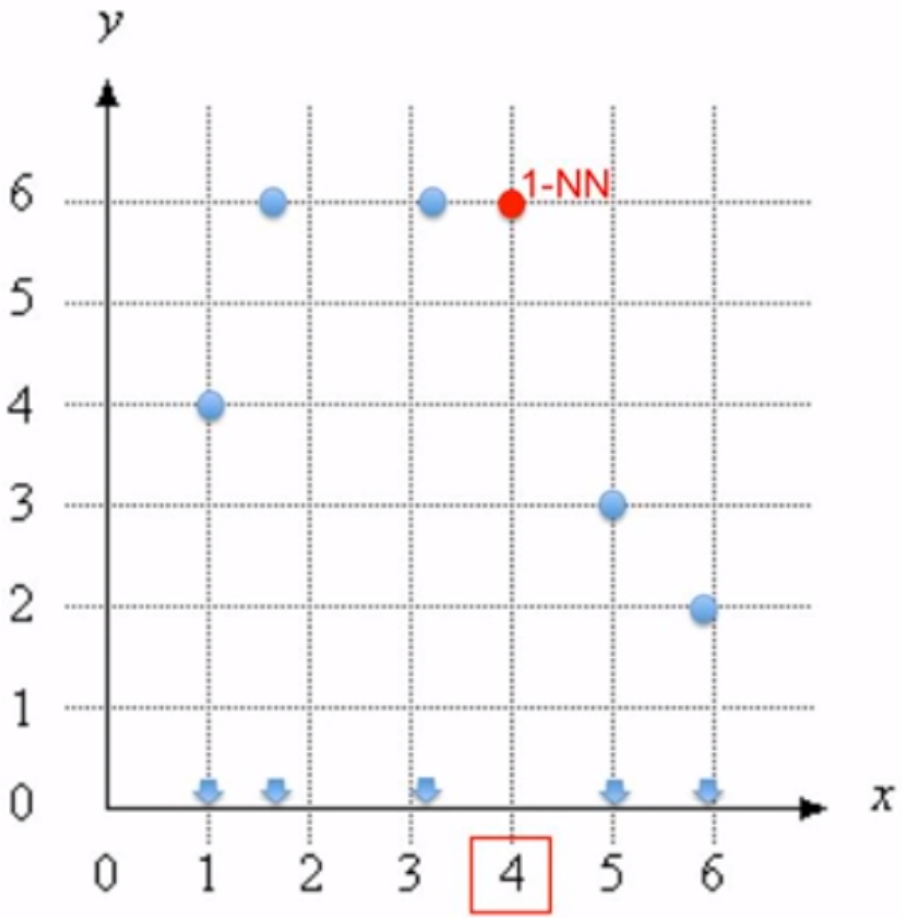
Example: kNN regression in 1-d



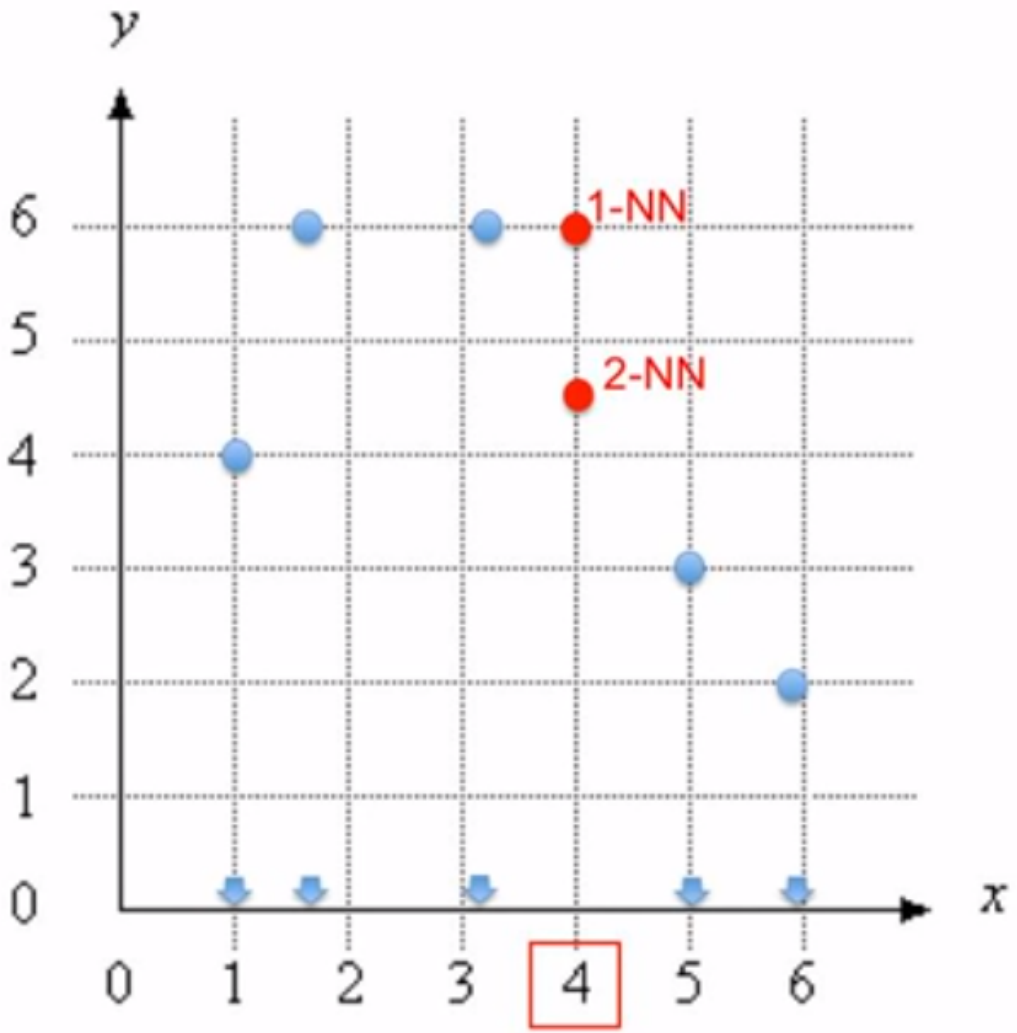
Example: kNN regression in 1-d



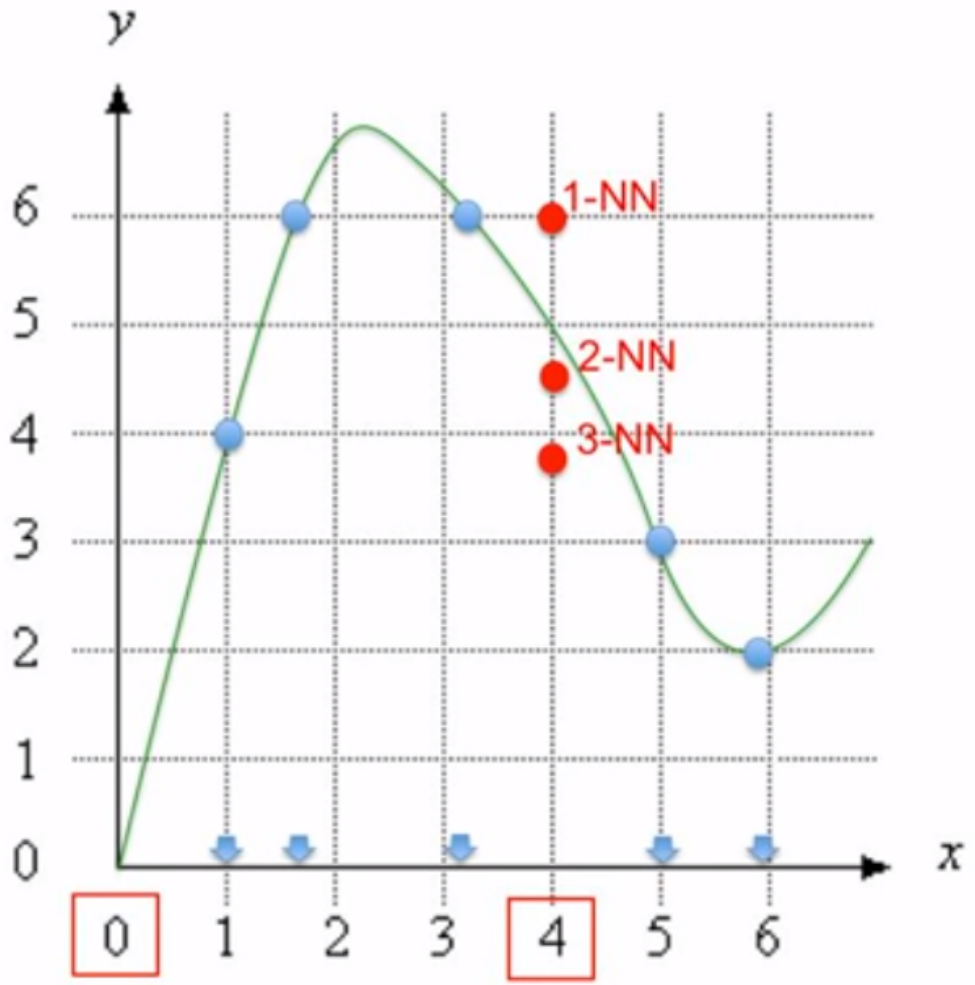
Example: kNN regression in 1-d



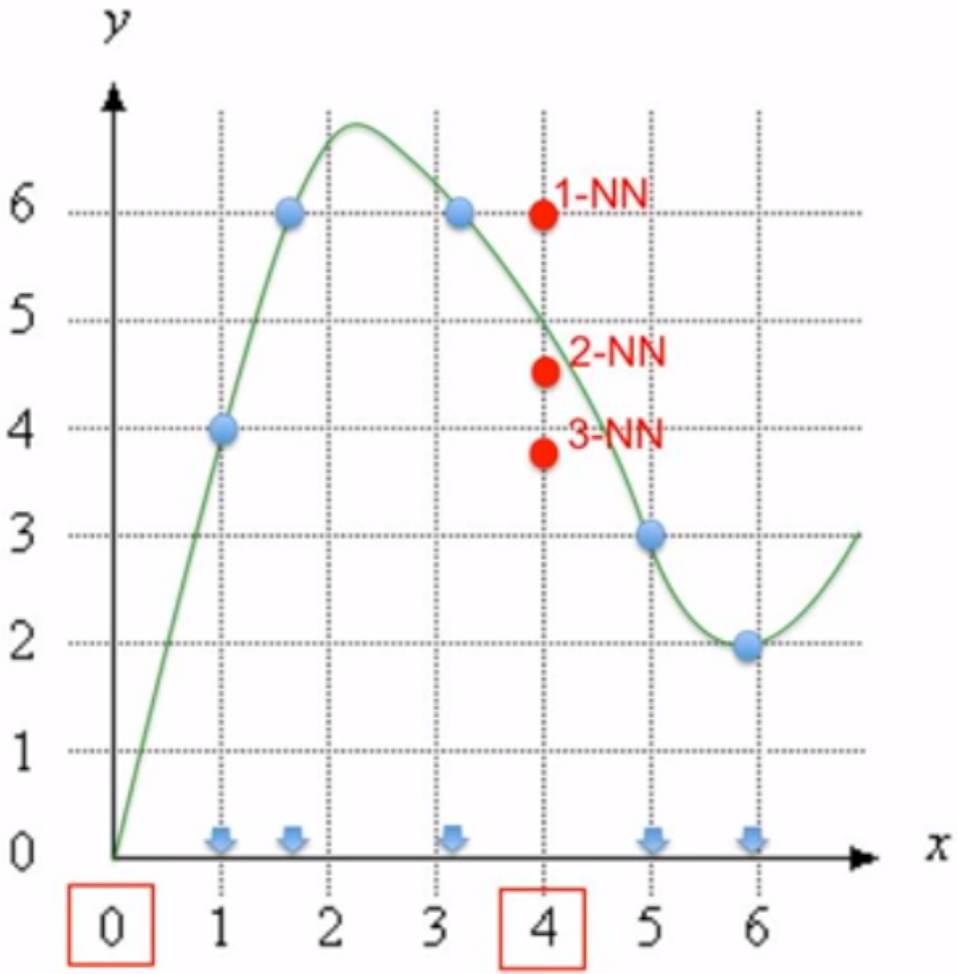
Example: kNN regression in 1-d



Example: kNN regression in 1-d

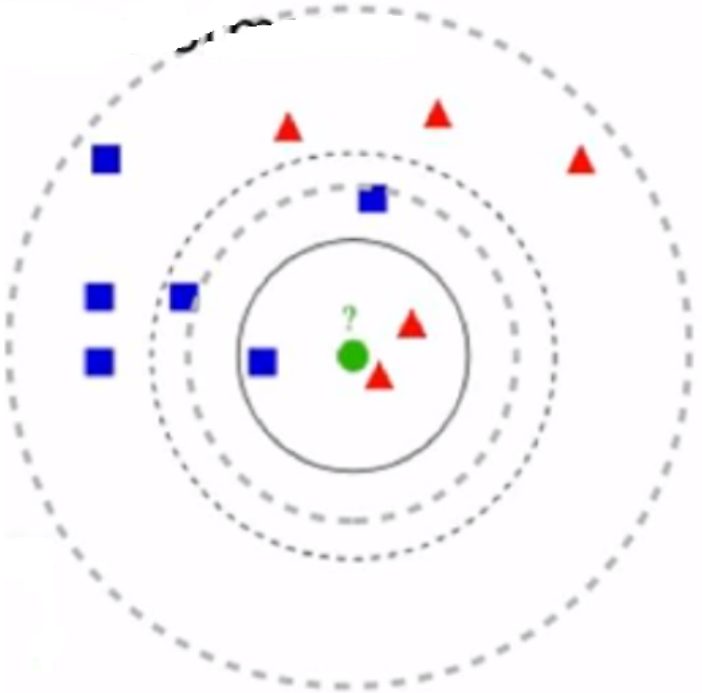


Example: kNN regression in 1-d



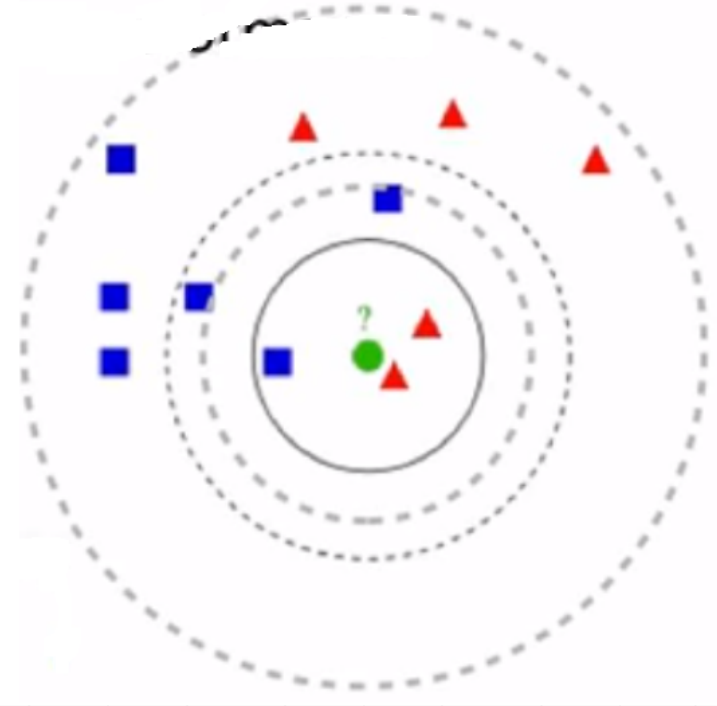
Choosing the value of k

- Value of k has strong effect on kNN performance
  - - large value → everything classified as the most probable class
  - - small value → highly variable, unstable decision boundary
    - Small changes to training set→ large changes in classification
- -affects “smoothness” of the boundary



Choosing the value of k

- Selecting value of k
  - Set aside a portion of training data (validation set)
  - Vary k, observe training → validation error
  - Pick k that gives the best generalization performance

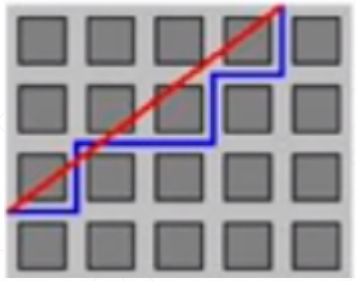


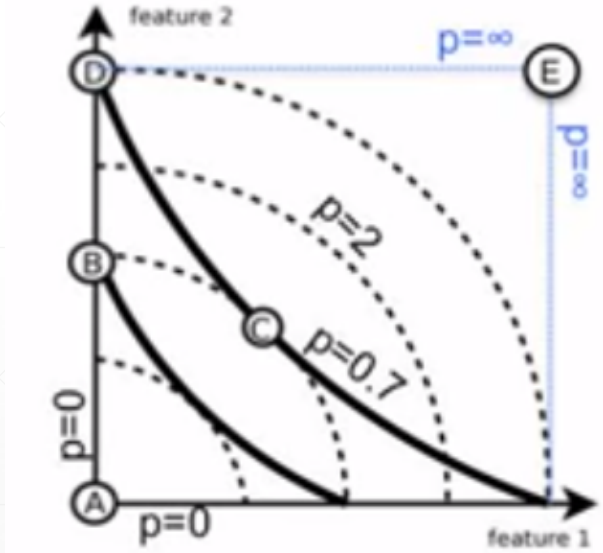
Distance measures

- Key component of the kNN algorithm
  - -- defines which examples are similar & which aren't
  - -- can have a strong effect on performance
- Euclidian (numerical attributes): $D(x,x') = \sqrt{\sum_d |x_d - x'_d|^2}$
  - -- symmetric, spherical, treat all dimensions equally
  - -- sensitive to extreme differences in single attribute
- Hamming (categorical attributes): $D(x,x') = \sum_d 1_{x_d \neq x'_d}$
  - -- number of attributes where x, x' differ

Distance measures

- Minkowski distance (p-norm): $D(x,x') = \sqrt[p]{\sum_d |x_d - x'_d|^p}$
  - p = 2 Euclidian
  - p = 1 Manhattan
  - p = 0 Hamming
  - p = ∞ $\max_d |x_d - x'_d|$



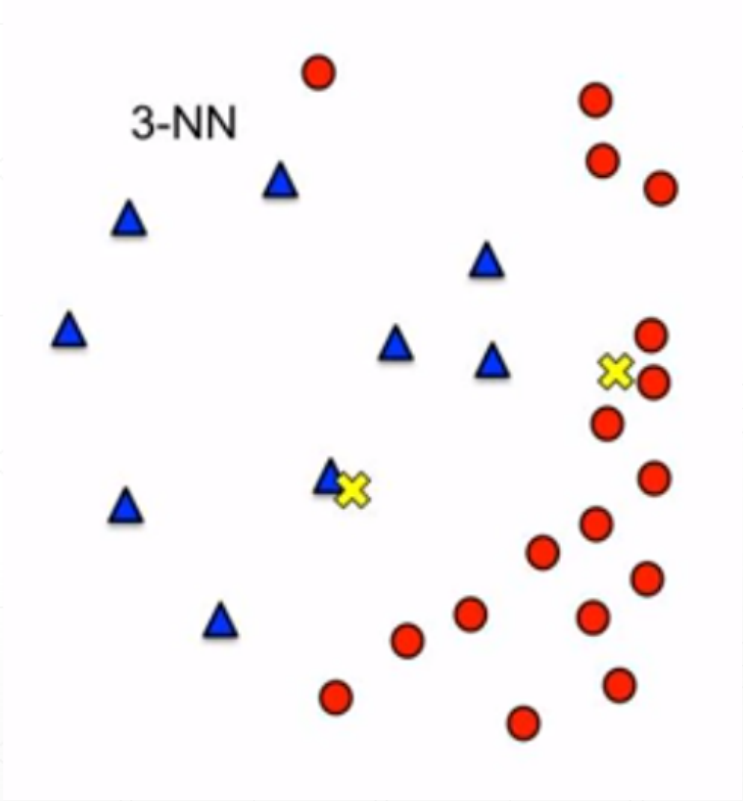


- Kullback Leibler (KL) divergence
  - For histograms
- Custom distance measures (BM25 for text)

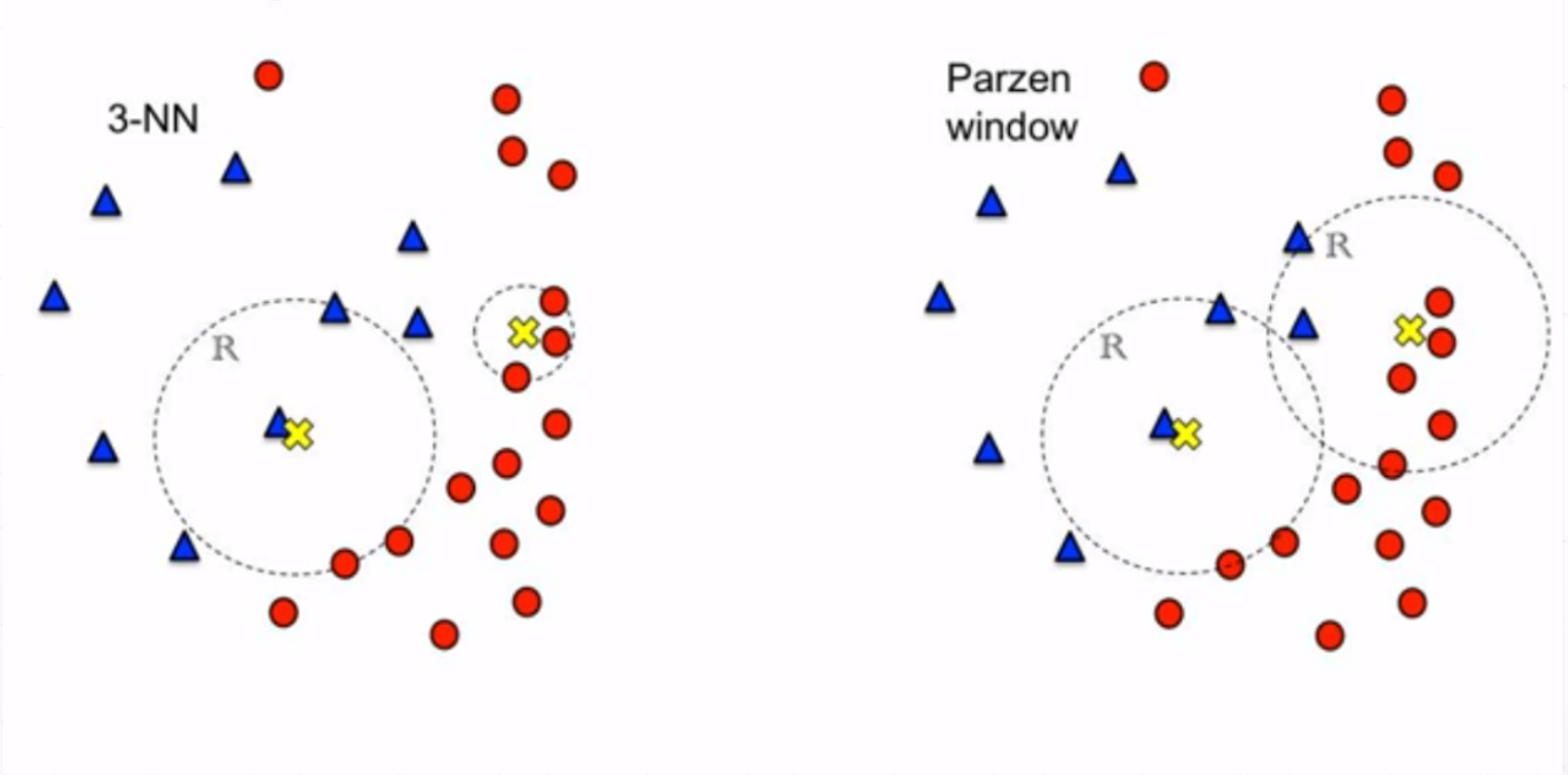
Issues

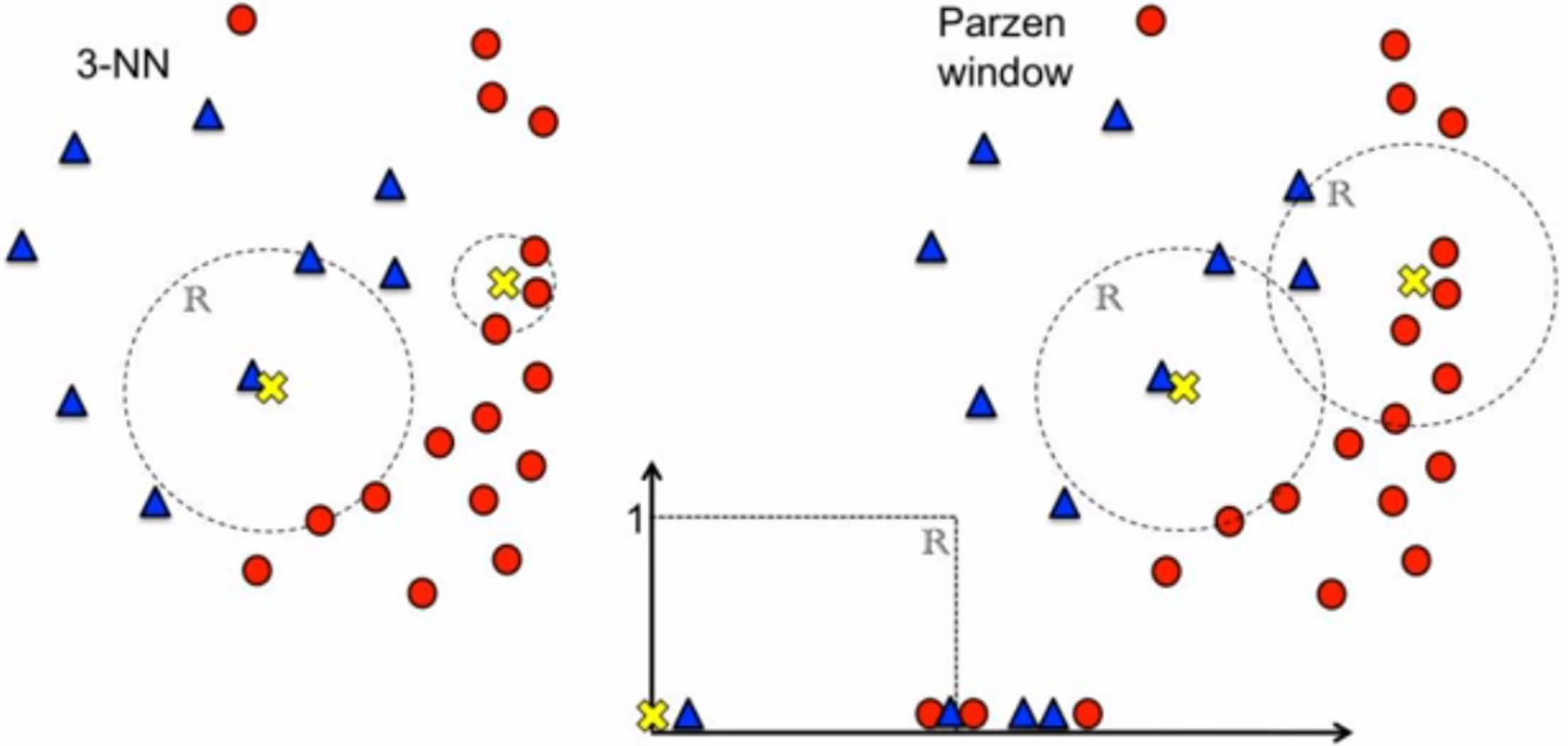
- Resolving ties:
  - -- equal number of positive/negative neighbors
  - -- use odd k (doesn't solve multi class)
  - -- breaking ties:
    - Random: flip a coin to decide positive / negative
    - Prior: pick class with greater prior
    - Nearest; use 1-nn classifier to decide
- Missing values
  - -- have to "fill in", otherwise cant compute distance
  - -- key concern: should affect distance as little as possible
  - -- reasonable choice: average value across all dataset

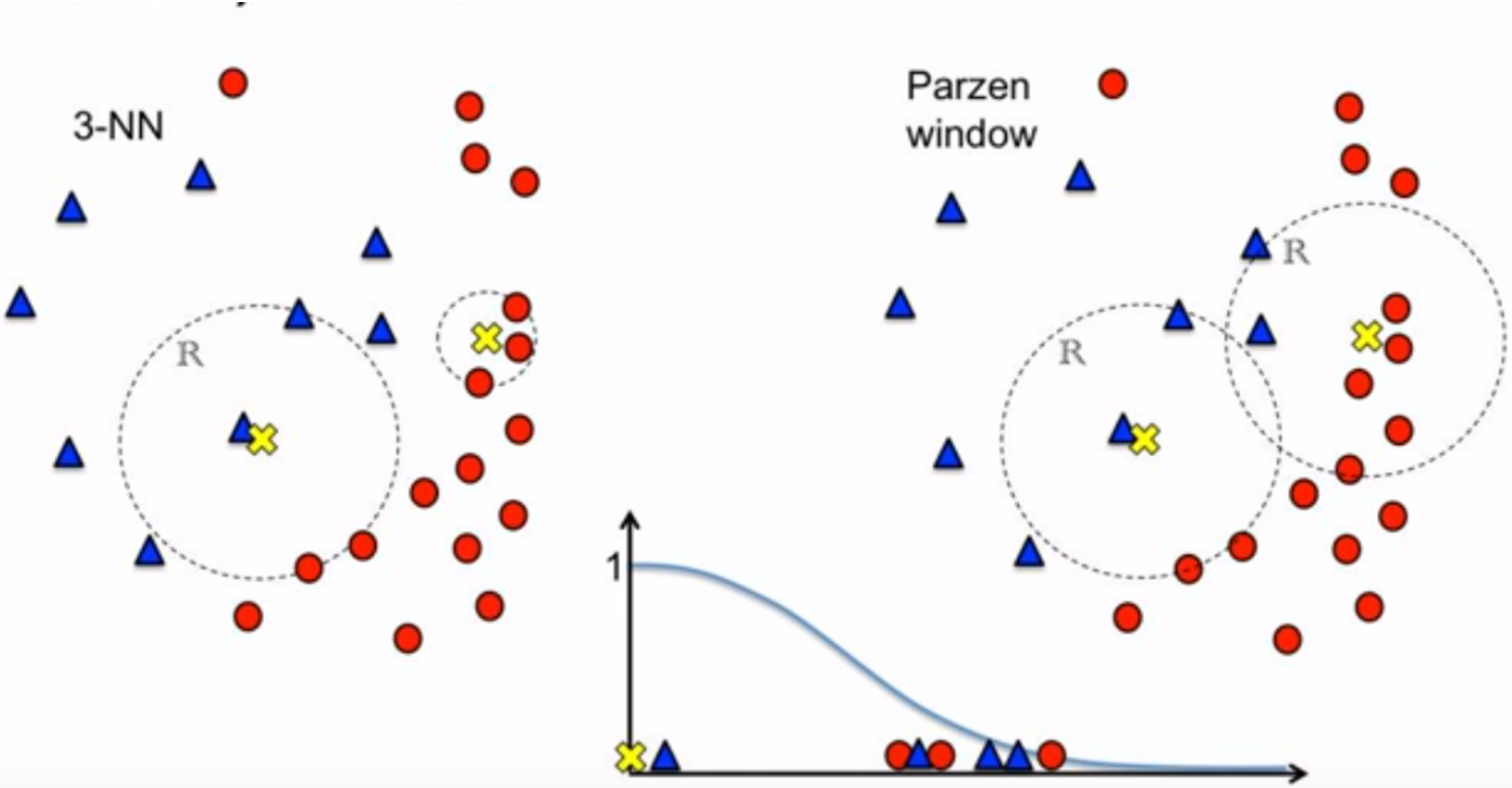
kNN, Parzen Windows and Kernels



kNN, Parzen Windows and Kernels

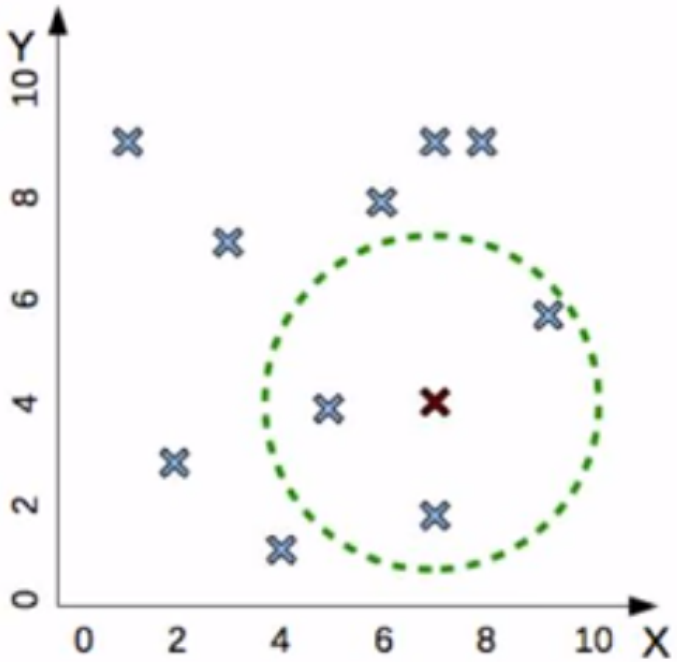



3-NN
R
Parzen
window
R
R
1
R


3-NN
R
Parzen
window
R
R
1

Why is kNN slow?

What you see



Find nearest neighbors
of the testing point (red)

What algorithm sees

What you see

What algorithm sees

- Training set:

  {(1,9), (2,3), (4,1),
  (3,7), (5,4), (6,8),
  (7,2), (8,8), (7,9), (9,6)}

- Testing instance:

  (7,4)

What you see

Find nearest neighbors
of the testing point (red)

What algorithm sees

- Training set:

  {(1,9), (2,3), (4,1),
  (3,7), (5,4), (6,8),
  (7,2), (8,8), (7,9), (9,6)}

- Testing instance:

  (7,4)

- Nearest neighbors?

  compare one-by-one
  to each training instance

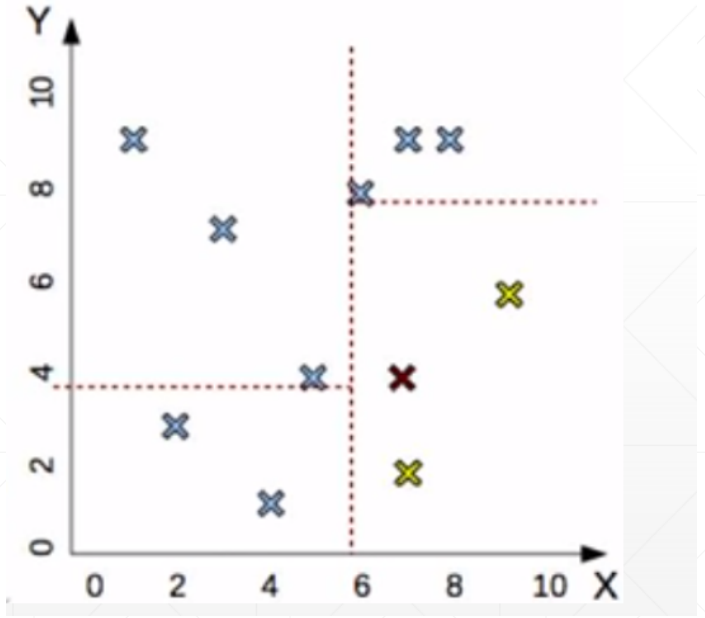
- n comparisons
- each takes d operations

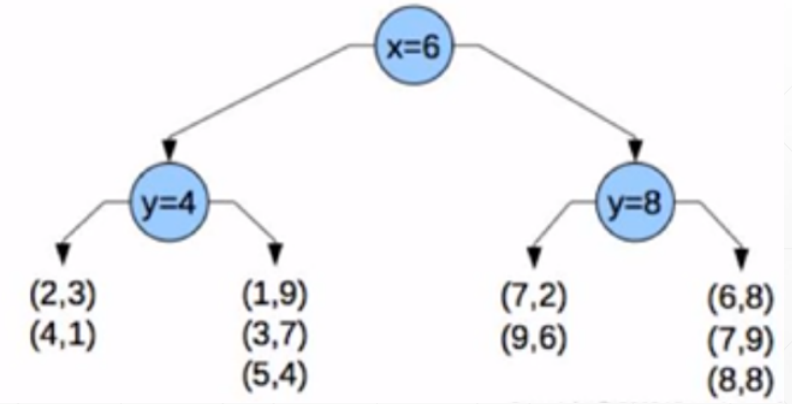
Making kNN fast

- Training: O(1), but testing: O(nd)
- Reduce d: dimensionality reduction
  - - simple feature selection, other methods $O(d^3)$
- Reduce n: don't compare to all training example
  - -- idea: quickly identify m<<n potential neighbors
    - Compare only to those, pick k nearest neighbors → O(md) time
  - -- K-D trees: low dimensional, real valued data
    - O (d log n), only works when d << n, inexact: may miss neighbors
    - -- aproximate
  - -- inverted lists: high dimensional, discrete data
    - O (d'n') where d'<<d, n'<<n, only for sparse data(e.g. text)
    - -- exact

K-D tree example

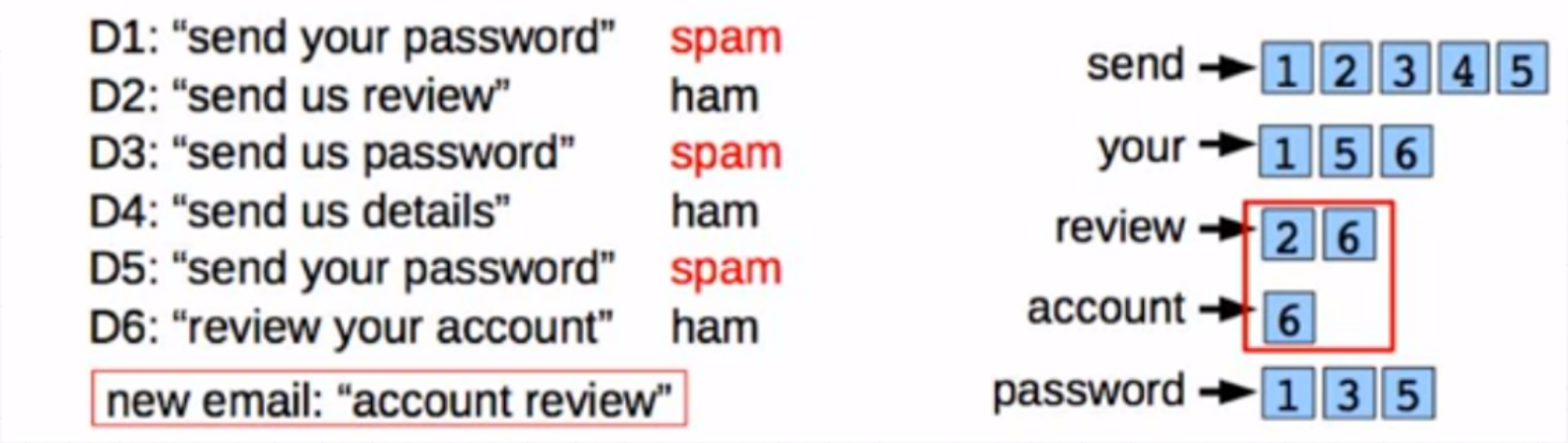
- Building a K-D tree from a training data:
  - -{(1,9),(2,3),(4,1),(3,7),(5,4),(6,8),(7,2),(8,8,),(7,9),(9,6)}
  - - pick random dimension, find median, split data, repeat
- Find NN for new point (7,4)
  - Find the region containing (7,4)
  - Compare to all points in region





Inverted List example

- Data structure used by search engines
  - -- list all training example that contains particular attribute
  - -- assumption: most attribute value are zero (sparseness) Zipf's law
- Given a new testing example:
  - -- merge inverted lists for attributes present in new example
  - -- O(dn): d.. nonzero attributes, n is avg length of inverted list



Pros and Cons

- Almost no assumptions about the data
  - Assumption implied by distance function
  - Non parametric approach: “let the data speak for itself”
- Need to handle missing value
- Sensitive to outliers (mislabeled training example)
- Sensitive to a lot of irrelevant features (affect distances)
- Computationally expensive
  - High testing time

Summary

- Key idea: nearby points → same class
  - -- important to select good distance function
- Can be used for classification and regression
- Simple, non-linear, asymptomatically optimal
  - -- does not make assumption about the data
  - -- “let the data speak for itself”
- Select k by optimization error on held-out set
- Naïve implementations slow for big datasets
  - -- use K-D trees (low-d) or inverted lists (high-d)

Discussion

Thank you!