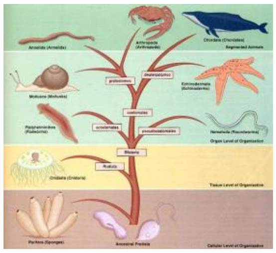
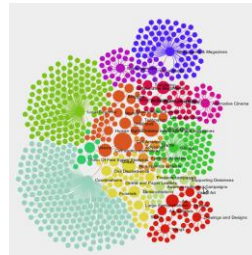




Generic Machine Learning

Clustering and Classification

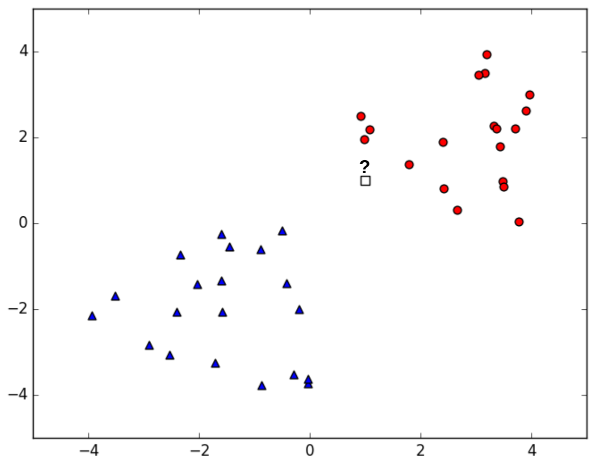
TCTI-VKAAI-17: Applied Artificial Intelligence

Classification

- The problem of identifying to which set of categories a new observation belongs
  - On the basis of a training data set containing observations whose category is known

- Examples:
  - Spam filtering, Optical Character Recognition (OCR), Search Engines, Computer Vision

Intuition of Classification

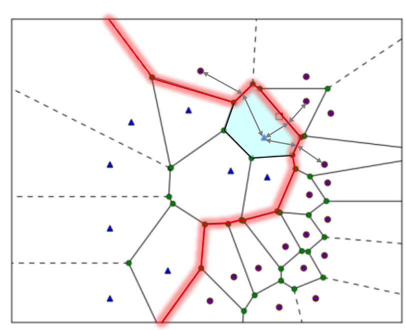


- Set of points (x,y)
  - Two classes (red / blue)
- Is the box red or blue?
- How did you do it?
  - Calculate prior?
  - Gaussian probability?
- Nearby points are red
  - Use as basis for learning algorithm

3

Nearest Neighbor Classification

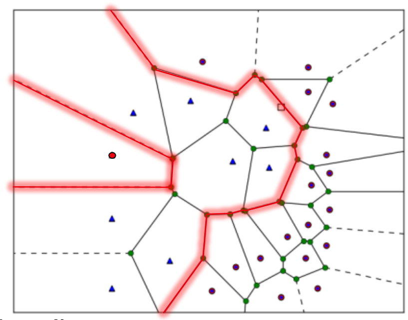
- Use the intuition to classify new point x:
  - Find the most similar training example x'
  - Predict its class y'
- Voronoi tesselation
  - Partitions in space into regions
  - Boundary: points at same distance from two different training examples
- Classification boundary
  - Non-linear, reflects classes well
  - Impressive for simple method



4

Nearest Neighbor: Outliers

- Algorithm sensitive to outliers
  - Single mislabeled example dramatically changes boundary
- Overfitting
  - Boundary exactly follows your training data
- Idea:
  - Use more than one nearest neighbor to make decision
  - Count class labels in k most similar training examples
    - Many "triangles" will outweigh single "circular" outlier



5

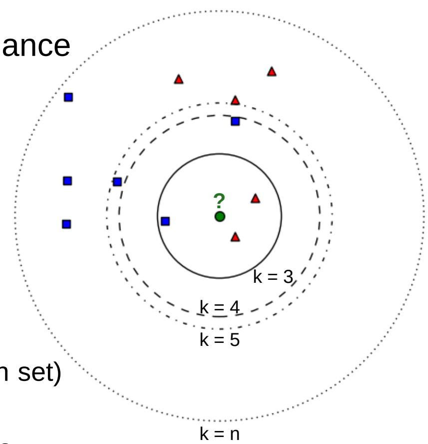
kNN Classification Algorithm

- Given:
  - Training examples $\{x_i, y_i\}$
    - x_i . . . attribute-value representation of examples
    - y_i . . . class labels: {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 $x_{i_1}, \ldots, x_{i_k}$ and their labels $y_{i_1}, \ldots, y_{i_k}$
  - Output the class $y*$ which is most frequent in $y_{i_1}, \ldots, y_{i_k}$

6

Choosing value of k

- Value of k has strong effect on kNN performance
  - Large value → everything classified as most frequent class in training set
  - Small value → highly variable, unstable decision boundaries
    - Small changes to training set → large changes in classification
  - Affects 'smoothness' of the boundary
- Selecting the value of k
  - Set aside a portion of the training data (validation set)
  - Vary k, observe training → validation error
  - Pick k that gives best generalisation performance



Training & Validation Data

- A common method to validate the performance of machine learning algorithms
  - Split available set of data into
    1. Training Data
    2. Validation Data
  - Train algorithm on Training Data
  - Use Validation Data to see how well the algorithm performs on 'new' cases.
- Why should you never validate on training data?

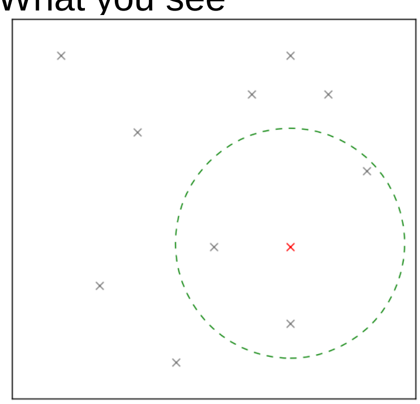
Practical Issues of kNN

- Resolving ties:
  - Equal number of positive/negative neighbors
  - Use odd k (does not solve multi-class)
  - Breaking ties:
    - Random: flip a coin
    - Prior: pick the most common class
    - Nearest: use 1-nn classifier to decide
- K-Nearest Neighbors is a **slow** mechanism!

9

Why is kNN slow?

- What you see



Find nearest neighbors
of testing point (red)

- What the algorithm sees

```
data = np.array([
    (1, 9), (2, 3), (4, 1), (3,7),
    (5, 4), (6, 8), (7, 2), (8, 8),
    (7, 9), (9, 6)
])
testPoint = (7, 4)
```

- Nearest neighbors?
  - Compare one-by-one to each training instance
- n comparisons
- Each takes d operations

10

Making kNN fast

- Reduce dimensions
  - Simple feature selection (throw away attributes that do not look promising)
- Reduce number of examples
  - Idea: quickly identify m << n potential near neighbors
  - Compare only to those neighbors

11