KNN Instance-Based Learning

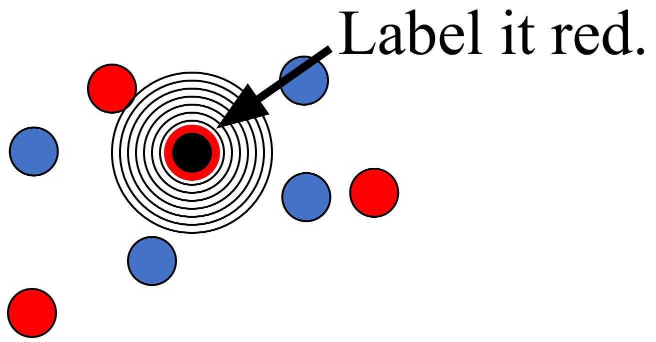
- Idea:
  - Datapoints with similar attributes belong to same class.
  - Classify new examples by comparing similar training examples.
- Algorithm:
  - Given some new case to predict its class y
  - Find similar training examples
  - Count how many similar examples are in each class and assign to max membership
- Consequence:
  - Memory based Learning
  - No need for weight parameters’ training !

Issues

- How to determine similarity?
- How many similar training examples to consider?
- How to avoid noisy classification, i.e. avoid overfitting?
- How to resolve clashes of classes?
- How to reduce complexity for large datasets?
- How to manage the curse of dimensionality – too many features?

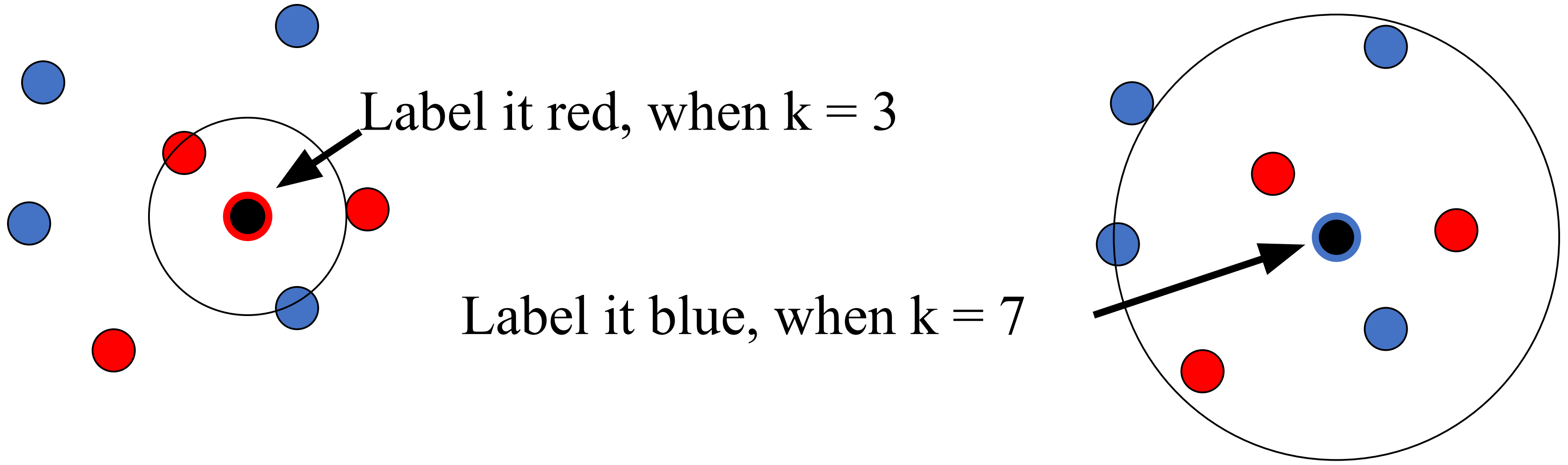
1-Nearest Neighbor

- One of the simplest of all machine learning classifiers
- May wrongly classify to a noisy example



k – Nearest Neighbor

- KNN (k>1) Generalizes 1-NN to **smooth away noise in the labels**
- Conduct majority voting among all K neighbors to select thee final class



Distance Metric

- Euclidean distance
- Two-dimensional: Dist(a,b) = sqrt($(a_1 - b_1)^2 + (a_2 - b_2)^2$
- Multivariate : Dist(a,b) = sqrt($\sum (a_i - b_i)^2$)
- Hamming distance

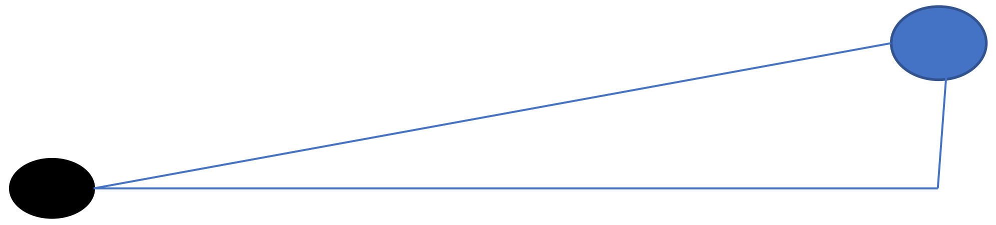
- When different units are used for each dimension
  □ normalize each attribute (j) by standard deviation

$$z_{ij} = x_{ij} - \mu_j)/\sigma_j,$$

Distance Metric

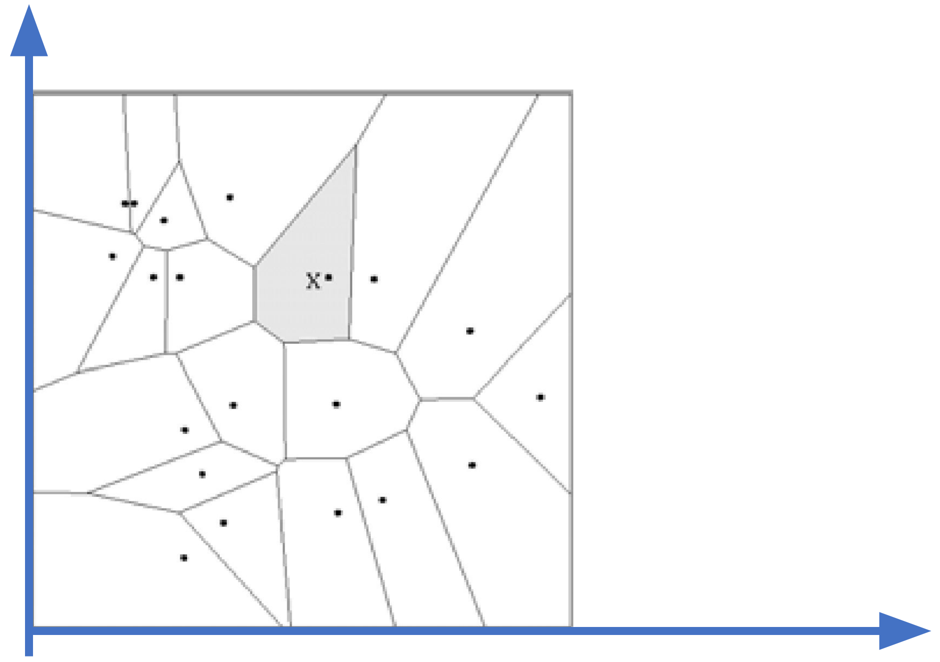
- For discrete data, can use hamming distance
   D(a,b) =number of features on which a and b differ

- Other distances - normal, cosine, Manhatten
- Using centroid distances (Mahalanobis Distance) regularizes KNN



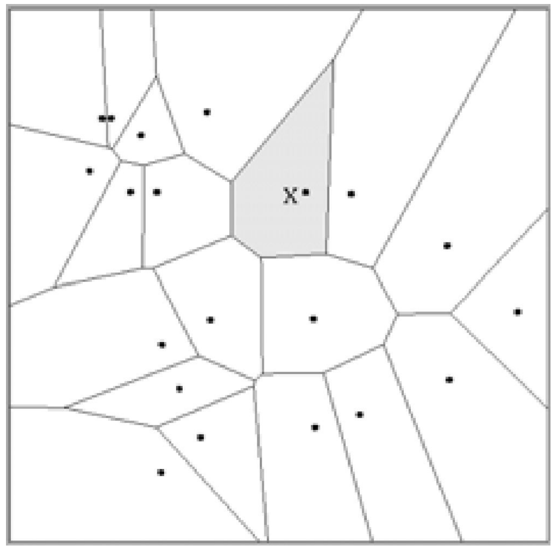
1-NN Voronoi tessalation

- Forms a Voronoi tessellation of the instance feature space

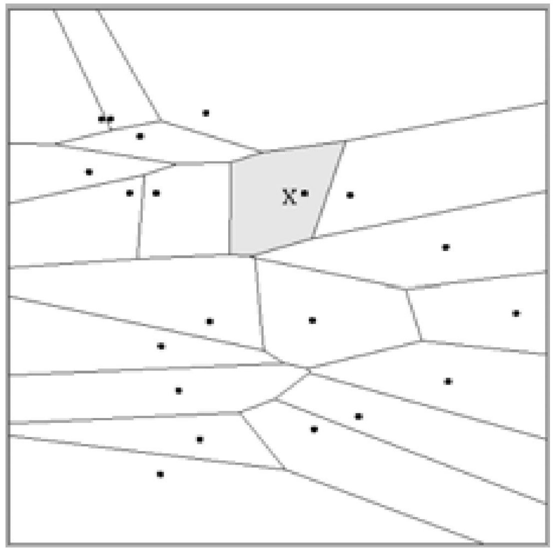


Distance Metrics

- Different metrics can change the decision surface



Dist($\mathbf{a,b}$) = sqrt($(a_1 - b_1)^2 + (a_2 - b_2)^2$)



Dist($\mathbf{a,b}$) = sqrt($(a_1 - b_1)^2 + (3a_2 - 3b_2)^2$)

KNN Example

	Food (3)	Chat (2)	Fast (2)	Price (3)	Bar (2)	BigTip
1	great	yes	yes	normal	no	yes
2	great	no	yes	normal	no	yes
3	mediocre	yes	no	high	no	no
4	great	yes	yes	normal	yes	yes

Similarity metric: Number of matching attributes (k=2)

•New examples:

- Example 1 (great, no, no, normal, no) Ye
  s

  □ most similar: number 2 (1 mismatch, 4 match) □ yes

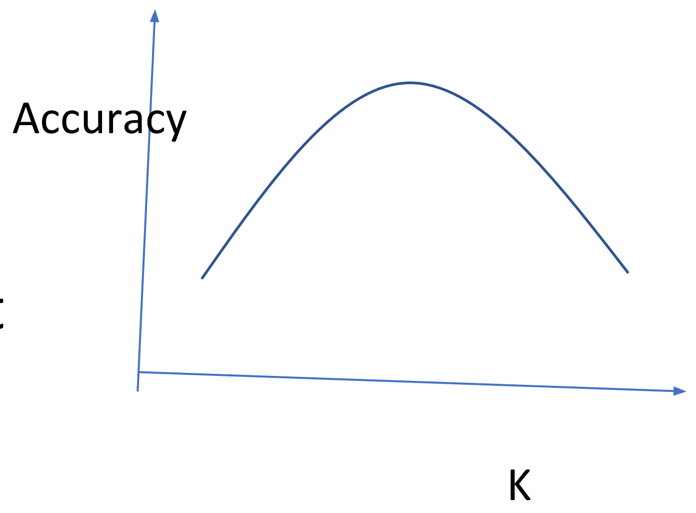
  □Second most similar example: number 1 (2 mismatch, 3 match) □ yes

- Example 2 (mediocre, yes, no, normal, no) Yes/No

  □ Most similar: number 3 (1 mismatch, 4 match) □ no

  □Second most similar example: number 1 (2 mismatch, 3 match) □ yes

Selecting K



- Increase k:
  - Makes KNN less sensitive to noise till a certain point where irrelevant classes interfere
- Decrease k:
  - Allows capturing finer structure of space till random noise dominates

- ⍰Pick k not too large, but not too small (depends on data)

Issues

- How to determine similarity? Euclidean distance/ Hamming distance
- How many similar training examples to consider?
  - Has to be odd for majority voting in classification.
  - Neither too small nor too big.
- How to avoid noisy classification, i.e. avoid overfitting?
- How to resolve clashes of classes?
- How to reduce complexity for large datasets? O(n*b)
- How to manage the curse of dimensionality?

Curse-of-Dimensionality

- Prediction accuracy can quickly degrade when number of attributes grows.
  - Irrelevant attributes easily “swamp” information from relevant attributes
  - When many irrelevant attributes, similarity/distance measure becomes less reliable

- Remedy
  - Try to remove irrelevant attributes in pre-processing step
  - Weight attributes differently
  - Increase k (but not too much)

Complexity

- For a given new test case, using brute force:

  O(n*d), n: no of training examples, d: no of attributes
- Parallelize distance calculations
- Space partitioning – Structure the data points as a tree
- Approximate nearest neighbor search – fuzzify attributes and defuzzify to target class
- Feature selection and elimination, dimension reduction
- Nearest prototypes
- Combination methods