Machine Learning

CS60050

k-Nearest Neighbour Classification

A Simple Species Classification Problem

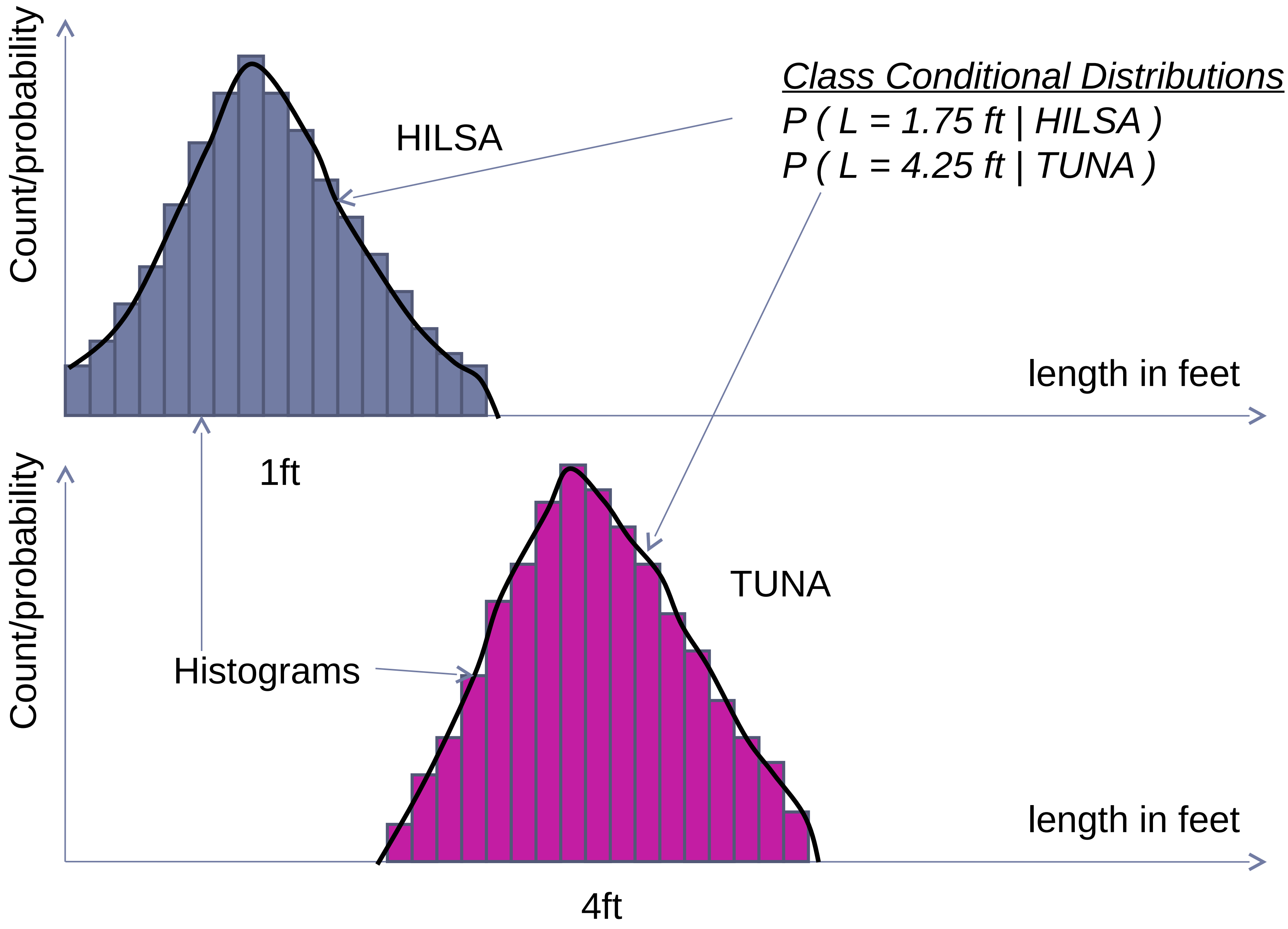
- Measure the *length* of a fish, and decide its <u>class</u>
  - Hilsa or Tuna

Collect Statistics …

Population for Class Hilsa

Population for Class Tuna

Distribution of “Fish Length”

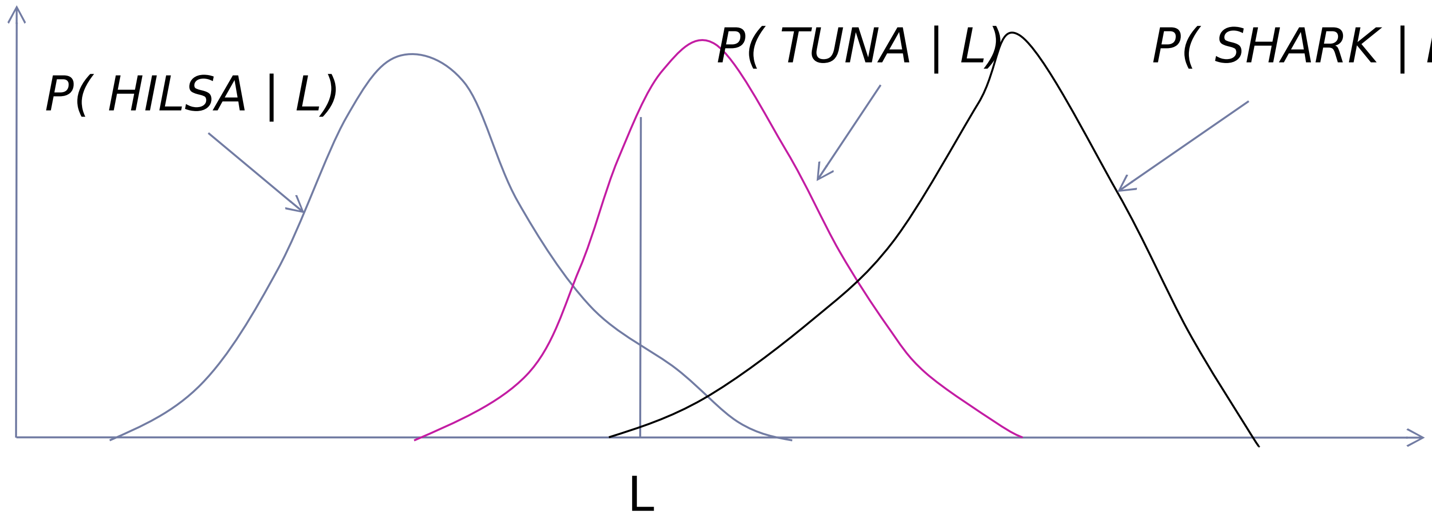


Class Conditional Distributions

P (L = 1.75 ft | HILSA)

P (L = 4.25 ft | TUNA)

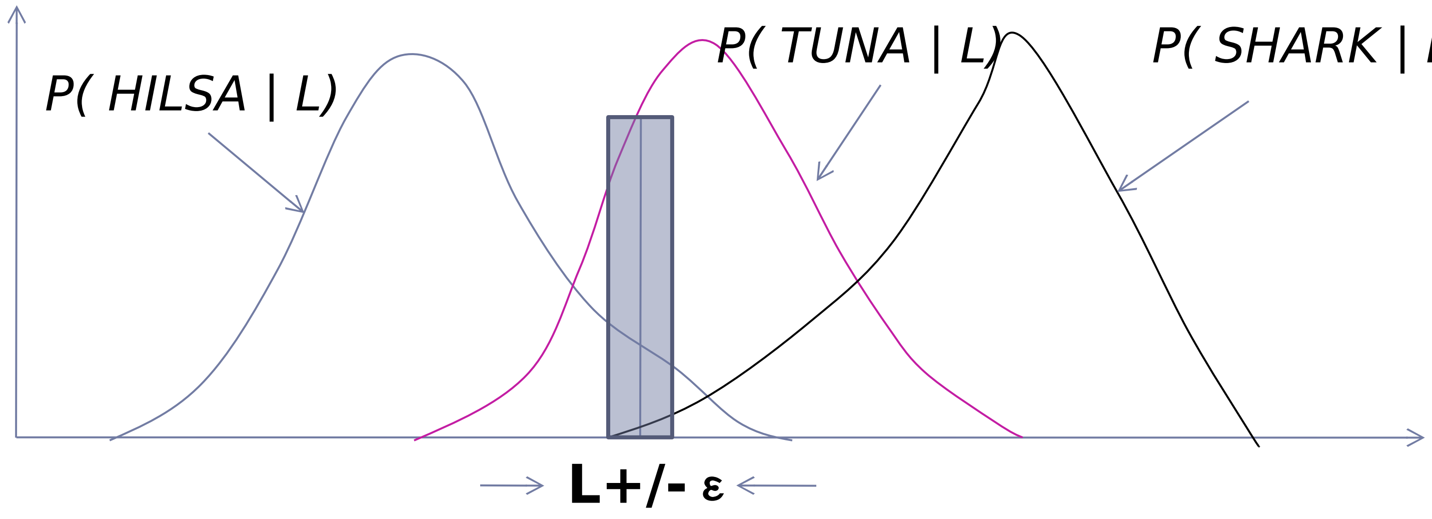
Bayes Classifier: Recap



Maximum Aposteriori (MAP) Rule

Distributions assumed to be of particular family (e.g., Gaussian), and parameters estimated from training data.

Bayes Classifier: Recap

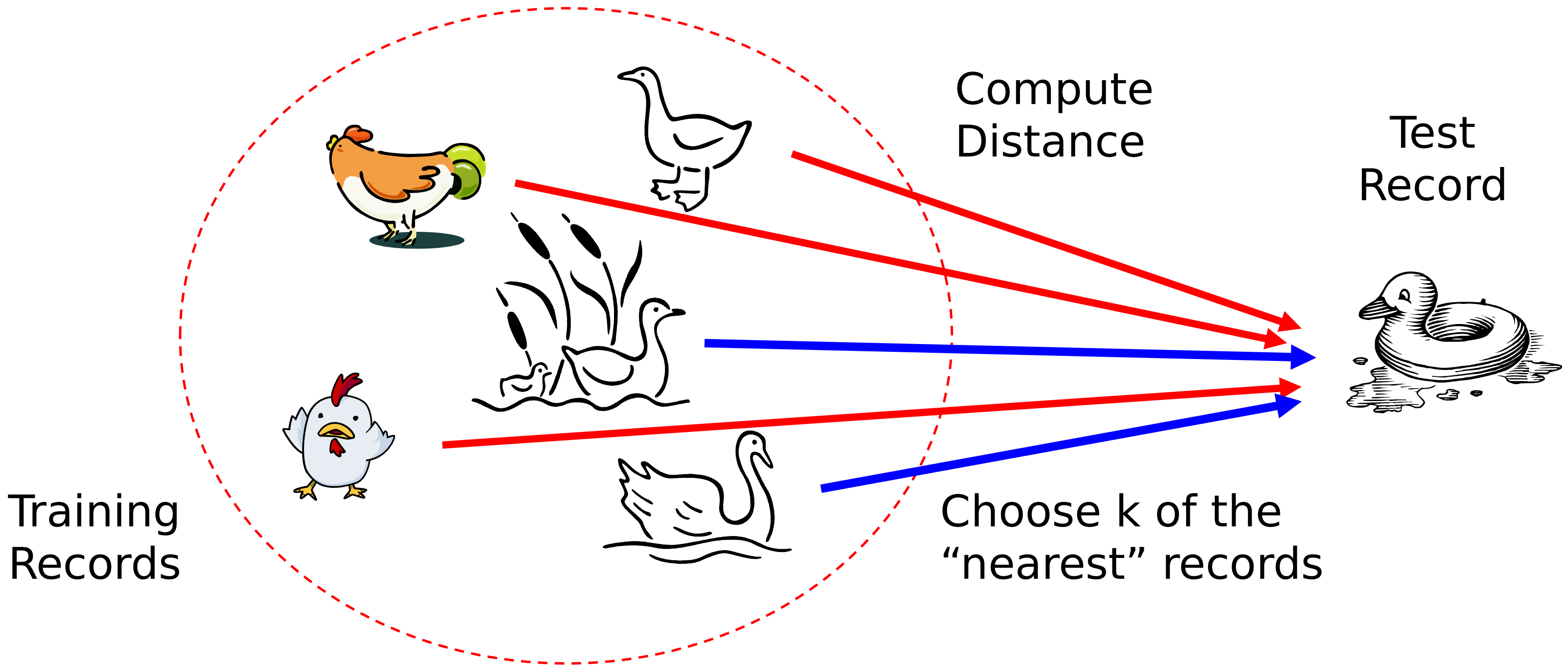


Approximate Maximum Aposteriori (MAP) Rule

Non-parametric (data driven) approach: consider a small window around L,
Find which class is most populous in that window.

Nearest Neighbor Classifiers

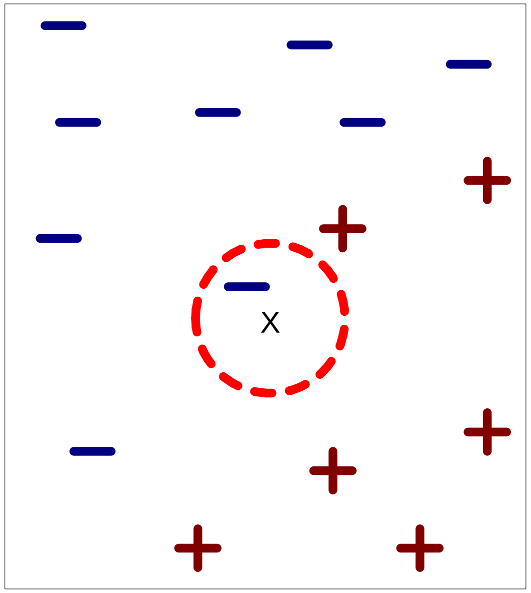
- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck



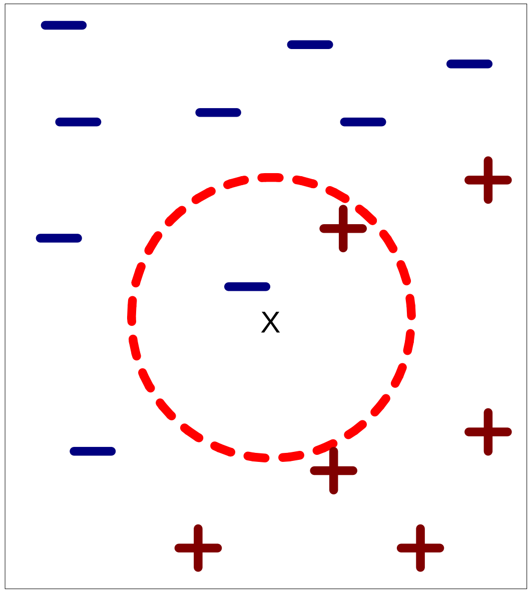
Basic Idea

- *k*-NN classification rule is to assign to a test sample the majority category label of its *k* nearest training samples
- In practice, *k* is usually chosen to be odd, so as to avoid ties
- The *k* = 1 rule is generally called the nearest-neighbor classification rule

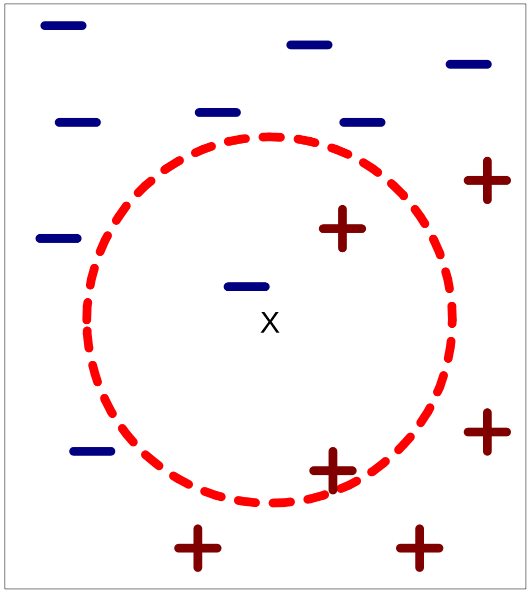
Definition of Nearest Neighbor



(a) 1-nearest neighbor



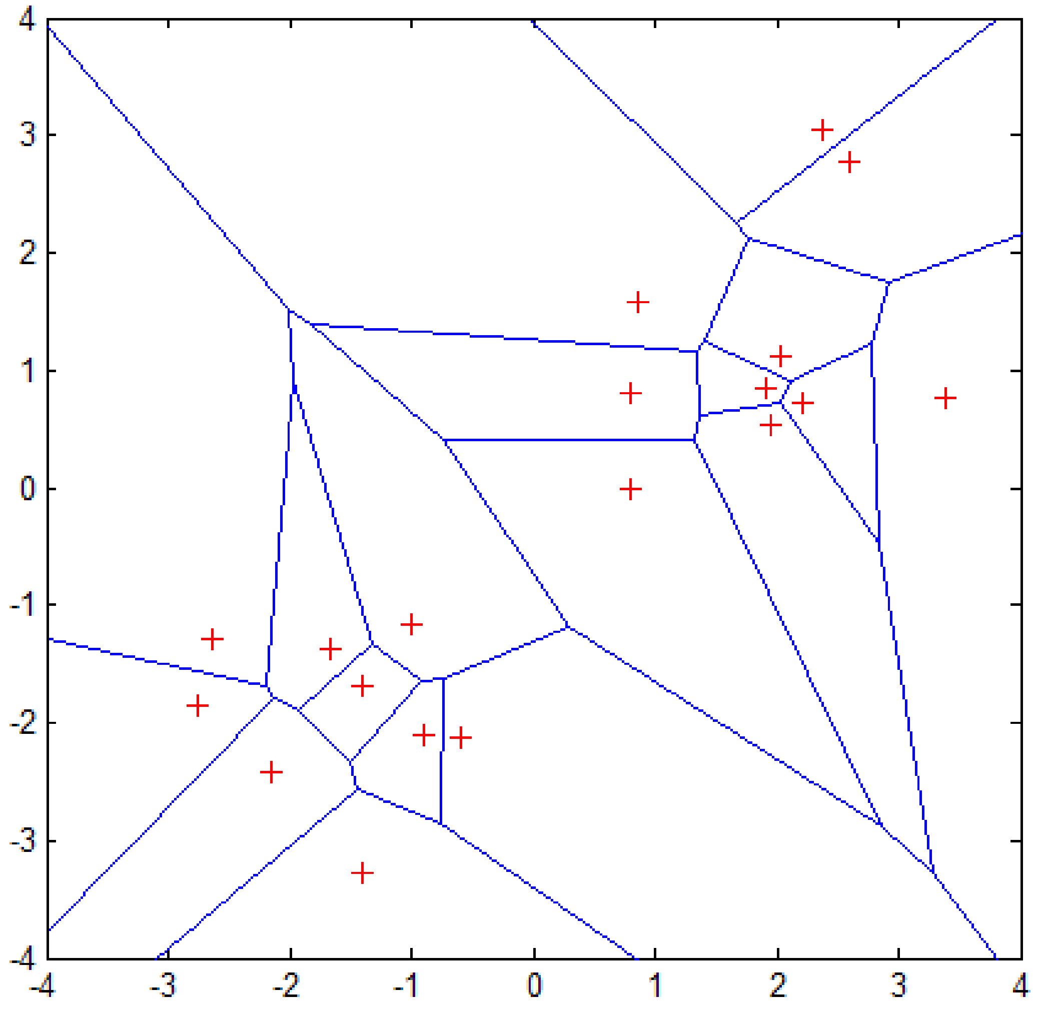
(b) 2-nearest neighbor



(c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

Voronoi Diagram



Properties:

1) All possible points within a sample's Voronoi cell are the nearest neighboring points for that sample
2) For any sample, the nearest sample is determined by the closest Voronoi cell edge

Distance-weighted *k*-NN

- Replace $\hat{f}(q) = \arg\max_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$ by:

$$\hat{f}(q) = \arg\max_{v \in V} \sum_{i=1}^{k} \frac{1}{d(x_i, x_q)^2} \delta(v, f(x_i))$$

General Kernel functions like Parzen Windows may be considered
Instead of inverse distance.

Predicting Continuous Values

- Replace $\hat{f}(q) = \arg\max_{v \in V} \sum_{i=1}^{k} w_i \delta(v, f(x_i))$ by:

$$\hat{f}(q) = \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

- Note: unweighted corresponds to w_i=1 for all *i*

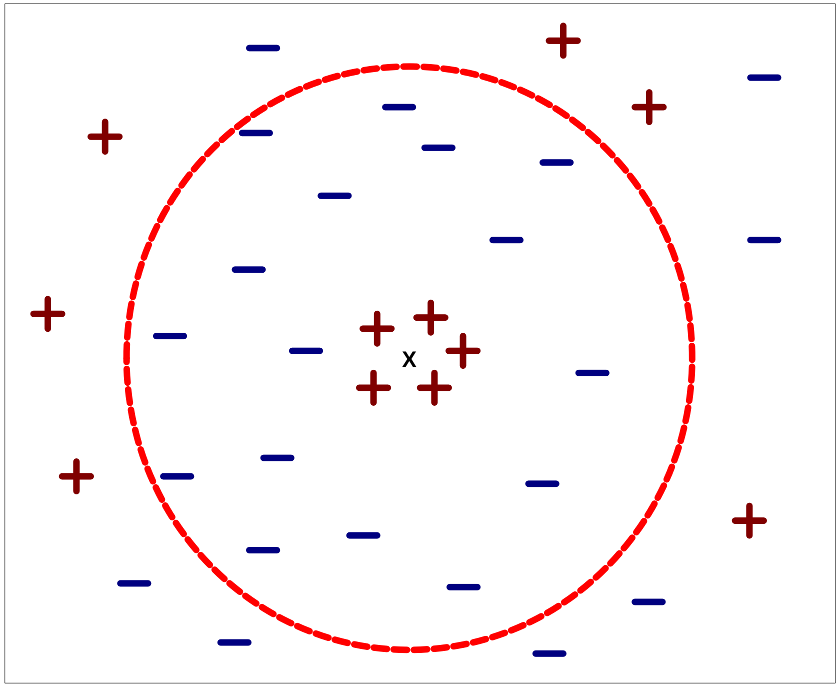
Nearest-Neighbor Classifiers: Issues

- The value of *k*, the number of nearest neighbors to retrieve
- Choice of Distance Metric to compute distance between records

- Computational complexity
  - Size of training set
  - Dimension of data

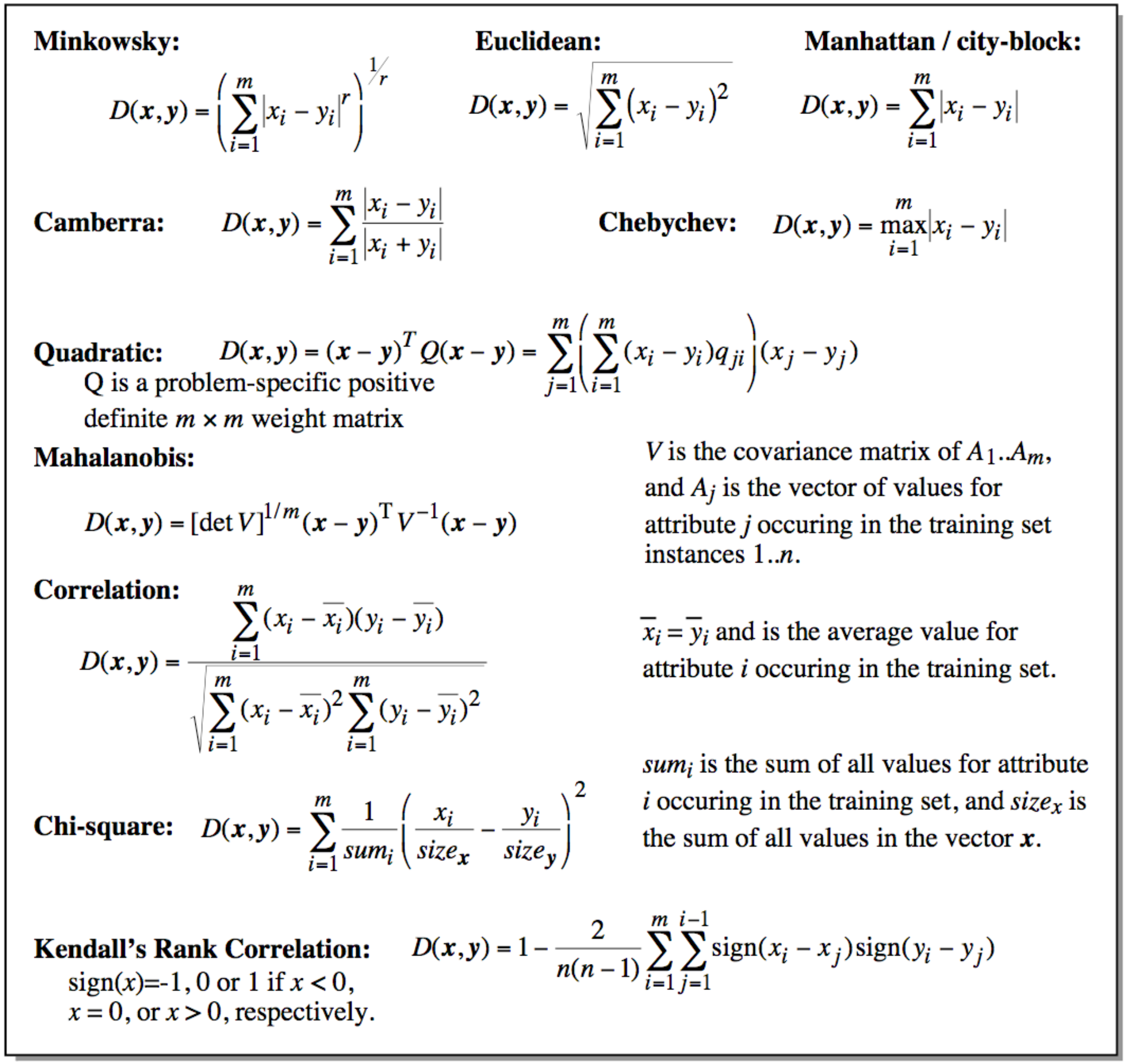
Value of K

- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes



Rule of thumb:
K = sqrt(N)
N: number of training points

Distance Metrics



Minkowsky:

$$D(\boldsymbol{x},\boldsymbol{y}) = \left(\sum_{i=1}^{m} |x_i - y_i|^r\right)^{1/r}$$

Euclidean:

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

Manhattan / city-block:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} |x_i - y_i|$$

Camberra:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \frac{|x_i - y_i|}{|x_i + y_i|}$$

Chebychev:

$$D(\boldsymbol{x},\boldsymbol{y}) = \max_{i=1}^{m} |x_i - y_i|$$

Quadratic:

$$D(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})^T Q (\boldsymbol{x} - \boldsymbol{y}) = \sum_{j=1}^{m}\left(\sum_{i=1}^{m} (x_i - y_i) q_{ji}\right)(x_j - y_j)$$

Q is a problem-specific positive definite $m \times m$ weight matrix

Mahalanobis:

$$D(\boldsymbol{x},\boldsymbol{y}) = [\det V]^{1/m} (\boldsymbol{x} - \boldsymbol{y})^{\mathrm{T}} V^{-1} (\boldsymbol{x} - \boldsymbol{y})$$

V is the covariance matrix of $A_1..A_m$, and A_j is the vector of values for attribute j occuring in the training set instances $1..n$.

Correlation:

$$D(\boldsymbol{x},\boldsymbol{y}) = \frac{\sum_{i=1}^{m} (x_i - \overline{x_i})(y_i - \overline{y_i})}{\sqrt{\sum_{i=1}^{m} (x_i - \overline{x_i})^2 \sum_{i=1}^{m} (y_i - \overline{y_i})^2}}$$

$\overline{x_i} = \overline{y_i}$ and is the average value for attribute i occuring in the training set.

Chi-square:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \frac{1}{sum_i}\left(\frac{x_i}{size_{\boldsymbol{x}}} - \frac{y_i}{size_{\boldsymbol{y}}}\right)^2$$

sum_i is the sum of all values for attribute i occuring in the training set, and $size_{\boldsymbol{x}}$ is the sum of all values in the vector $\boldsymbol{x}$.

Kendall's Rank Correlation:

sign(x)=-1, 0 or 1 if $x < 0$, $x = 0$, or $x > 0$, respectively.

$$D(\boldsymbol{x},\boldsymbol{y}) = 1 - \frac{2}{n(n-1)} \sum_{i=1}^{m} \sum_{j=1}^{i-1} \operatorname{sign}(x_i - x_j)\operatorname{sign}(y_i - y_j)$$

Figure 1. Equations of selected distance functions.
($\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors of m attribute values).

Distance Measure: Scale Effects

- Different features may have different measurement scales
  - E.g., patient weight in kg (range [50,200]) vs. blood protein values in ng/dL (range [-3,3])
- Consequences
  - Patient weight will have a much greater influence on the distance between samples
  - May bias the performance of the classifier

Standardization

- Transform raw feature values into z-scores

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}$$

  - x_{ij} is the value for the *i*th sample and *j*th feature
  - μ_j is the average of all x_{ij} for feature *j*
  - σ_j is the standard deviation of all x_{ij} over all input samples
- Range and scale of z-scores should be similar (providing distributions of raw feature values are alike)

Nearest Neighbor : Dimensionality

- Problem with Euclidean measure:
  - High dimensional data
    - curse of dimensionality
  - Can produce counter-intuitive results
  - Shrinking density – sparsification effect

1 1 1 1 1 1 1 1 1 1 1 0	vs	1 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1		0 0 0 0 0 0 0 0 0 0 0 1
d = 1.4142		d = 1.4142

Distance for Nominal Attributes

Value Difference Metric (VDM)

[Stanfill & Waltz, 1986]

Providing appropriate distance measurements for nominal attributes.

$$vdm_a(x,y) = \sum_{c=1}^{C} \left(\frac{N_{a,x,c}}{N_{a,x}} - \frac{N_{a,y,c}}{N_{a,y}} \right)^2$$

$N_{a,x}$ = # times attribute a had value x

$N_{a,x,c}$ = # times attribute a had value x and class was c

C = # output classes

Two values are considered closer if they have more similar classifications, i.e., if they have more similar correlations with the output classes.

Distance for Heterogeneous Data

In this section, we define a heterogeneous distance function *HVDM* that returns the distance between two input vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. It is defined as follows:

$$HVDM(\boldsymbol{x},\boldsymbol{y}) = \sqrt{\sum_{a=1}^{m} d_a^{\,2}(x_a, y_a)} \tag{11}$$

where m is the number of attributes. The function $d_a(x,y)$ returns a distance between the two values x and y for attribute a and is defined as:

$$d_a(x,y) = \begin{cases} 1, & \text{if } x \text{ or } y \text{ is unknown; otherwise...} \\ normalized\_vdm_a(x,y), & \text{if } a \text{ is nominal} \\ normalized\_diff_a(x,y), & \text{if } a \text{ is linear} \end{cases} \tag{12}$$

Wilson, D. R. and Martinez, T. R., Improved Heterogeneous Distance Functions, Journal of Artificial Intelligence Research, vol. **6**, no. 1, pp. 1-34, 1997

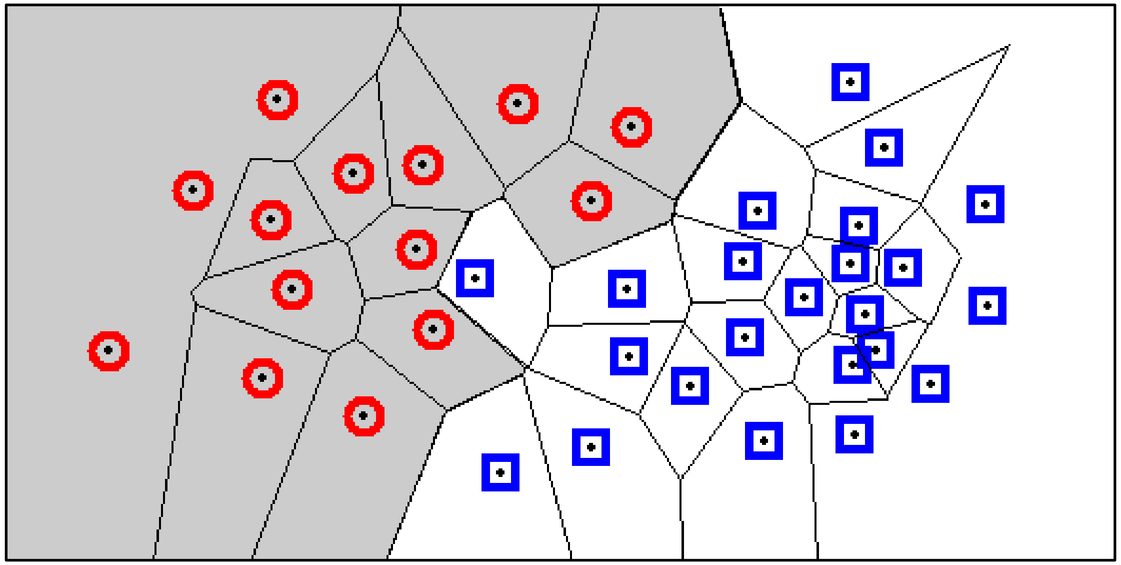
kNN : Computational Complexity

- Expensive
  - To determine the nearest neighbour of a query point *q*, must compute the distance to all *N* training examples
    - + Pre-sort training examples into fast data structures (kd-trees)
    - + Compute only an approximate distance (LSH)
    - + Remove redundant data (condensing)
- Storage Requirements
  - Must store all training data **P**
    - + Remove redundant data (condensing)
    - - Pre-sorting often increases the storage requirements
- High Dimensional Data
  - “Curse of Dimensionality”
    - Required amount of training data increases exponentially with dimension
    - Computational cost also increases dramatically
    - Partitioning techniques degrade to linear search in high dimension

Reduction in Computational Complexity

- Reduce size of training set
  - Condensation, editing
- Use geometric data structure for high dimensional search

Condensation: Decision Regions



Each cell contains one sample, and every location within the cell is closer to that sample than to any other sample.

A Voronoi diagram divides the space into such cells.

Every query point will be assigned the classification of the sample within that cell. The *decision boundary* separates the class regions based on the 1-NN decision rule.

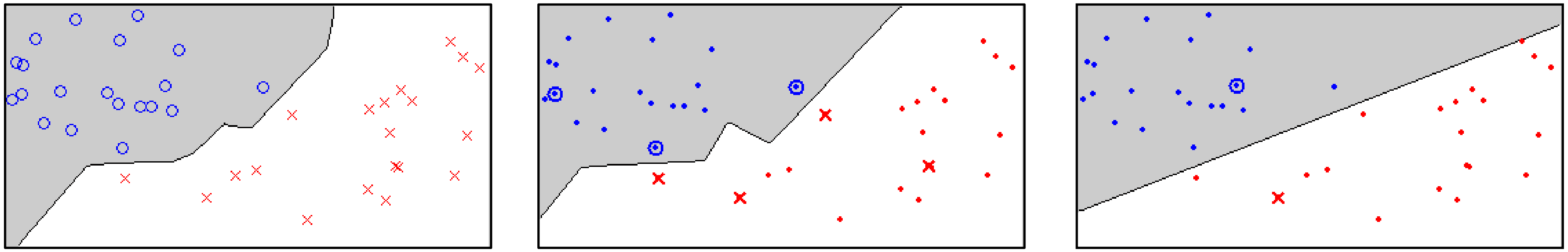
Knowledge of this boundary is sufficient to classify new points.

The boundary itself is rarely computed; many algorithms seek to retain only those points necessary to generate an identical boundary.

Condensing

- Aim is to reduce the number of training samples
- Retain only the samples that are needed to define the decision boundary

- Decision Boundary Consistent – a subset whose nearest neighbour decision boundary is identical to the boundary of the entire training set
- Minimum Consistent Set – the smallest subset of the training data that correctly classifies all of the original training data



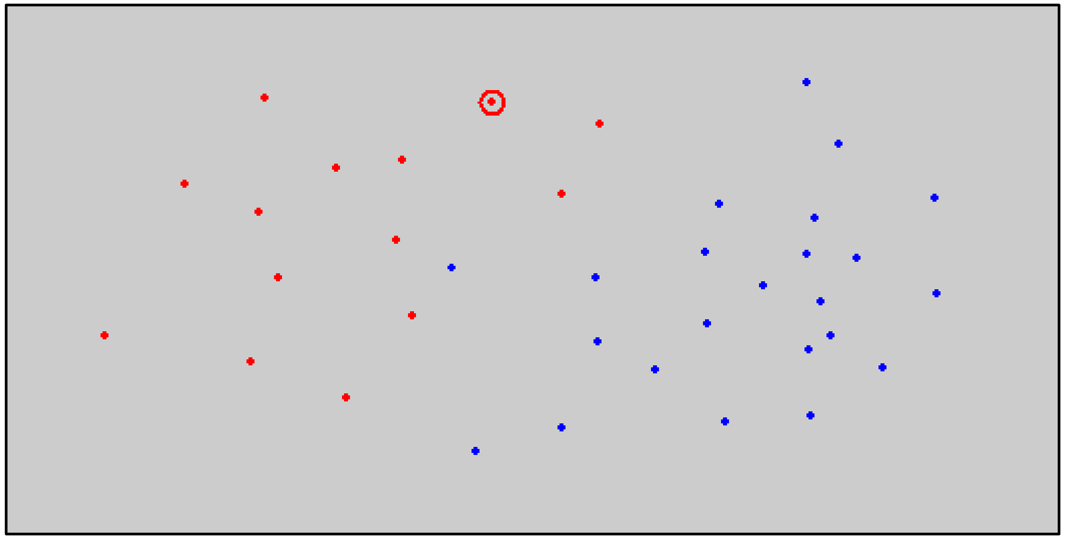
Original data   Condensed data   Minimum Consistent Set

Condensing

- Condensed Nearest Neighbour (CNN)

1. Initialize subset with a single (or K) training example
2. Classify all remaining samples using the subset, and transfer any incorrectly classified samples to the subset
3. Return to 2 until no transfers occurred or the subset is full

- Incremental
- Order dependent
- Neither minimal nor decision boundary consistent
- $O(n^3)$ for brute-force method

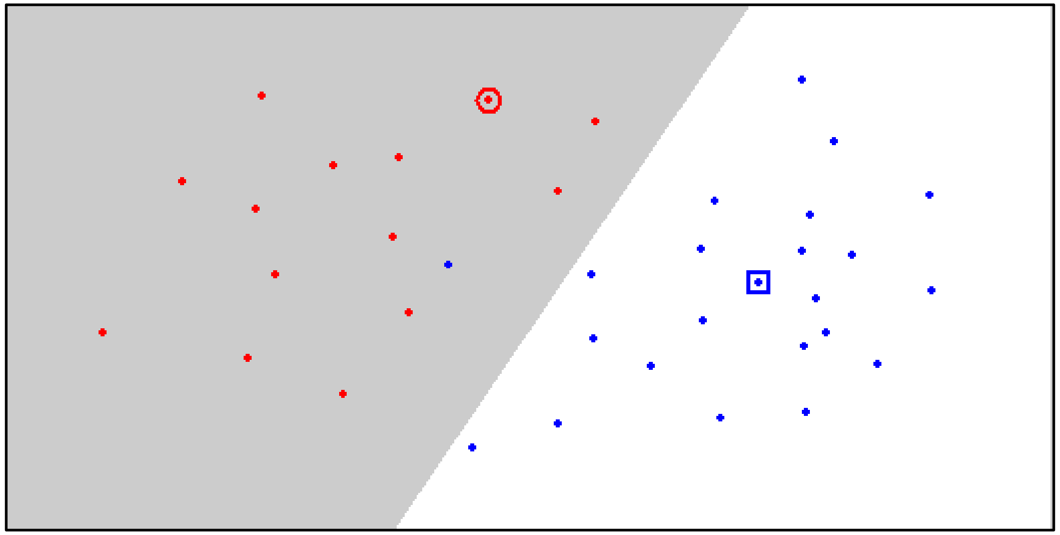


Condensing

- Condensed Nearest Neighbour (CNN)

1. Initialize subset with a single (or K) training example
2. Classify all remaining samples using the subset, and transfer any incorrectly classified samples to the subset
3. Return to 2 until no transfers occurred or the subset is full

- Incremental
- Order dependent
- Neither minimal nor decision boundary consistent
- $O(n^3)$ for brute-force method

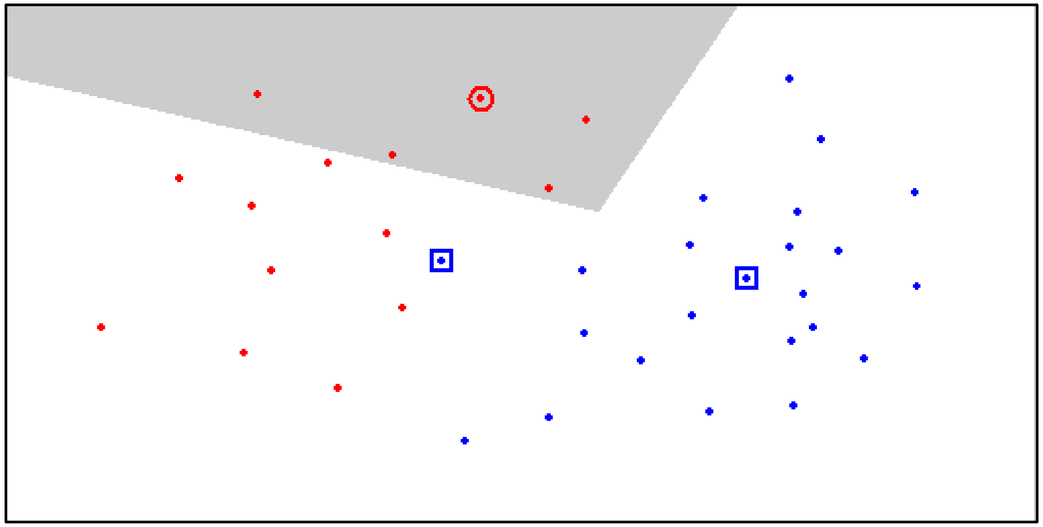


Condensing

- Condensed Nearest Neighbour (CNN)

1. Initialize subset with a single (or K) training example
2. Classify all remaining samples using the subset, and transfer any incorrectly classified samples to the subset
3. Return to 2 until no transfers occurred or the subset is full

- Incremental
- Order dependent
- Neither minimal nor decision boundary consistent
- $O(n^3)$ for brute-force method

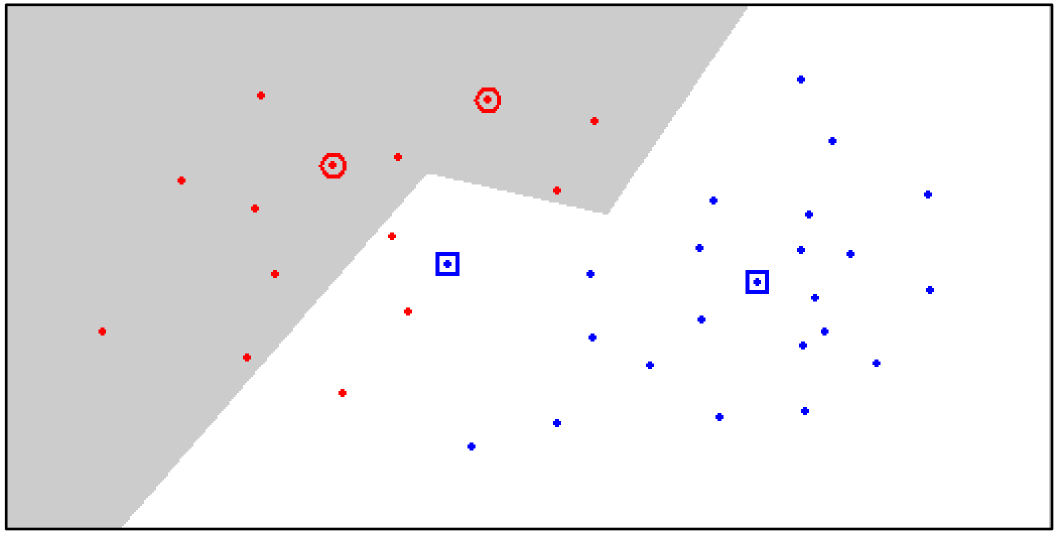


Condensing

- Condensed Nearest Neighbour (CNN)

1. Initialize subset with a single (or K) training example
2. Classify all remaining samples using the subset, and transfer any incorrectly classified samples to the subset
3. Return to 2 until no transfers occurred or the subset is full

- Incremental
- Order dependent
- Neither minimal nor decision boundary consistent
- $O(n^3)$ for brute-force method

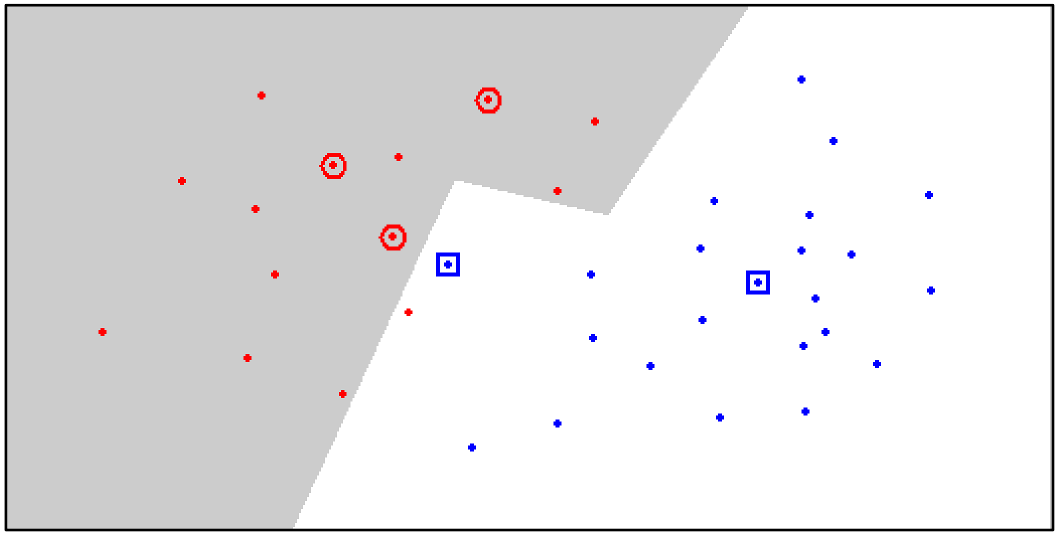


Condensing

- Condensed Nearest Neighbour (CNN)

1. Initialize subset with a single (or K) training example
2. Classify all remaining samples using the subset, and transfer any incorrectly classified samples to the subset
3. Return to 2 until no transfers occurred or the subset is full

- Incremental
- Order dependent
- Neither minimal nor decision boundary consistent
- $O(n^3)$ for brute-force method

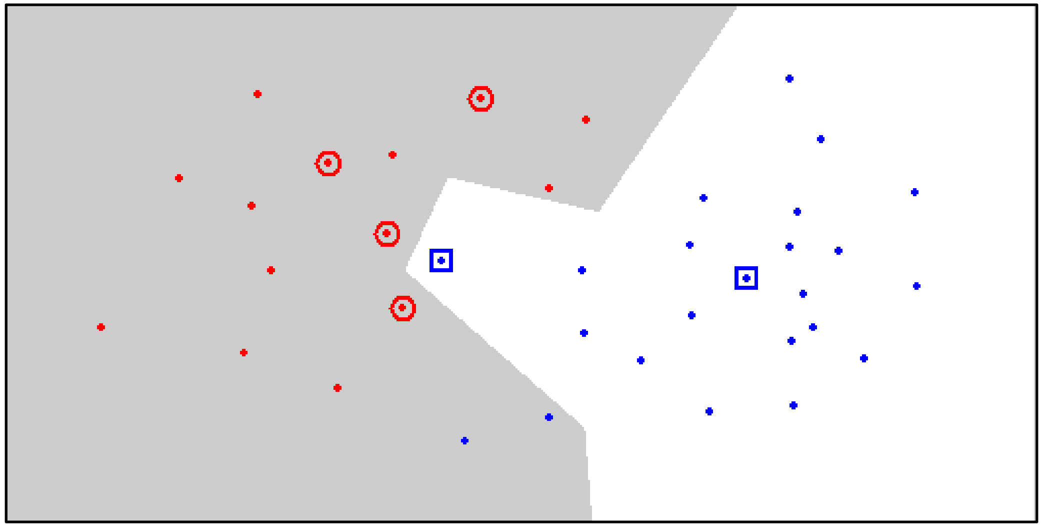


Condensing

- Condensed Nearest Neighbour (CNN)

1. Initialize subset with a single (or K) training example
2. Classify all remaining samples using the subset, and transfer any incorrectly classified samples to the subset
3. Return to 2 until no transfers occurred or the subset is full

- Incremental
- Order dependent
- Neither minimal nor decision boundary consistent
- $O(n^3)$ for brute-force method

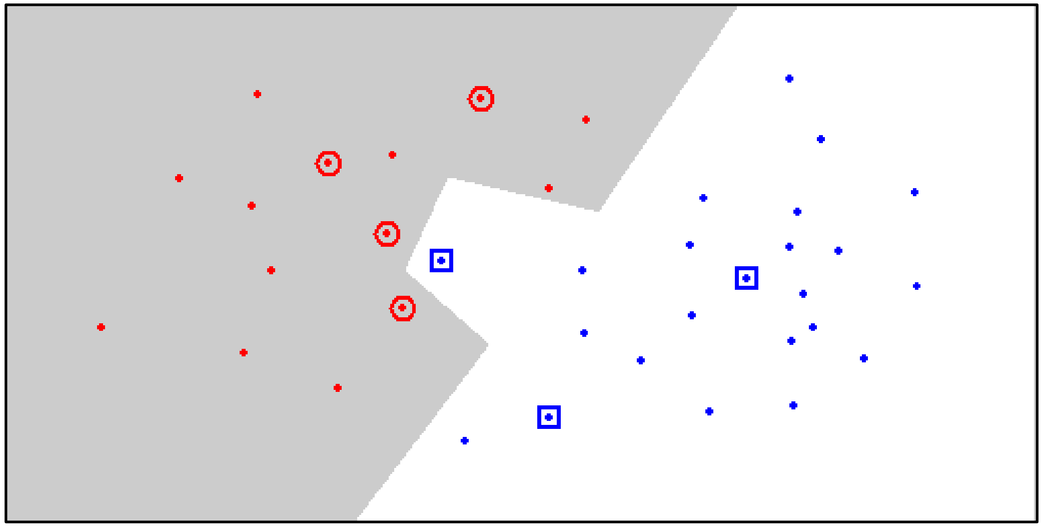


Condensing

- Condensed Nearest Neighbour (CNN)

1. Initialize subset with a single (or K) training example
2. Classify all remaining samples using the subset, and transfer any incorrectly classified samples to the subset
3. Return to 2 until no transfers occurred or the subset is full

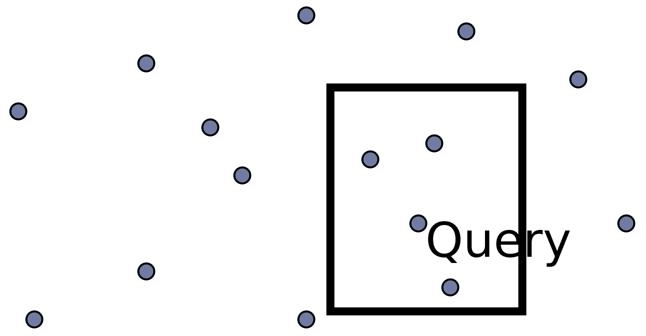
- Incremental
- Order dependent
- Neither minimal nor decision boundary consistent
- $O(n^3)$ for brute-force method



High dimensional search

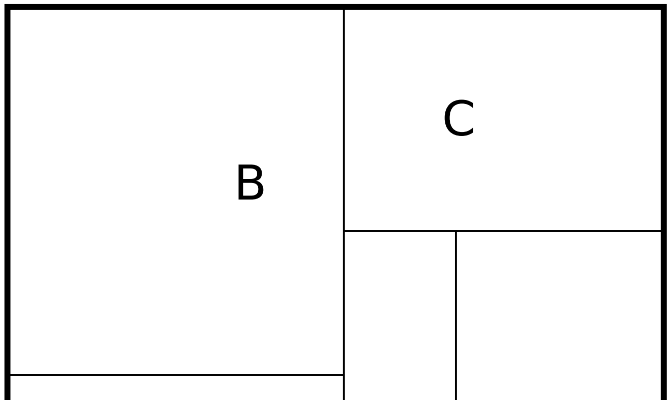
- Given a point set and a nearest neighbor query point

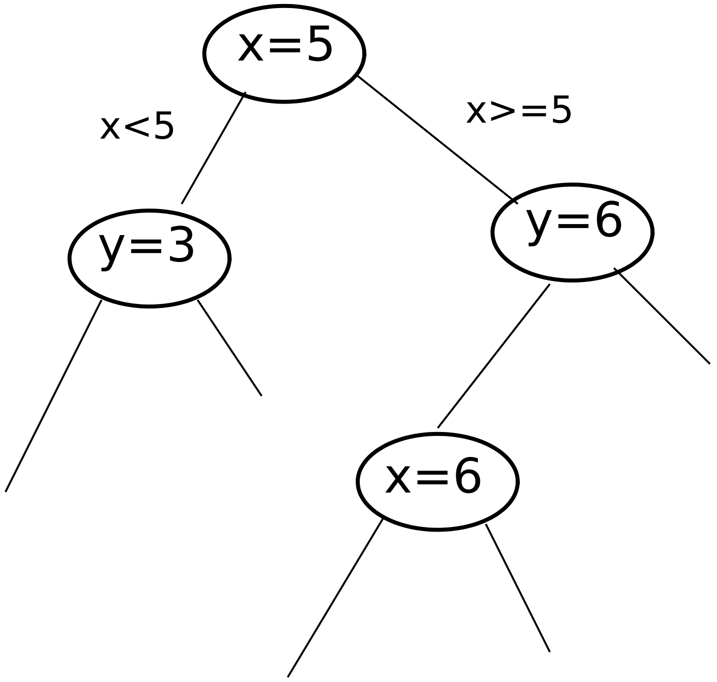
- Find the points enclosed in a rectangle (range) around the query
- Perform linear search for nearest neighbor only in the rectangle



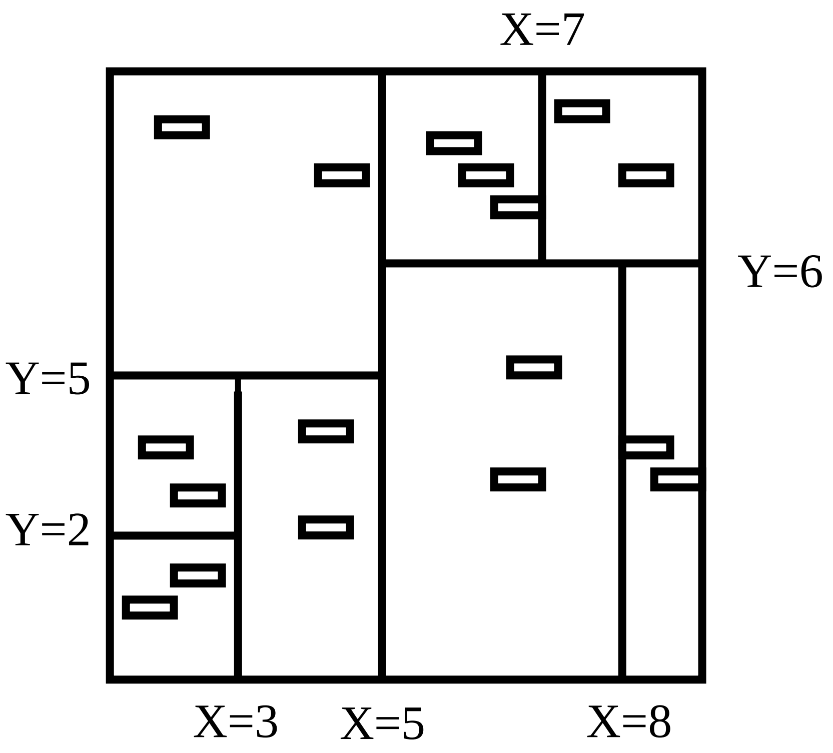
kd-tree: data structure for range search

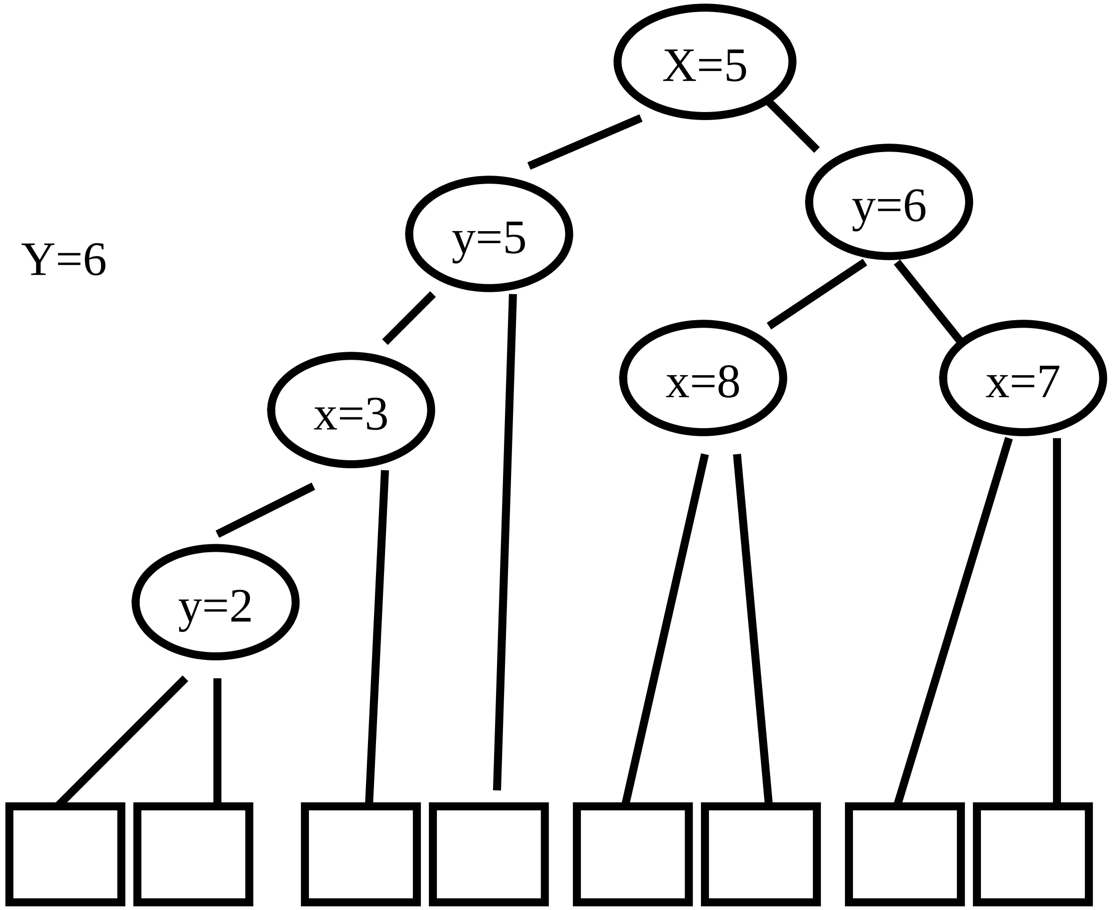
- Index data into a tree
- Search on the tree
- Tree construction: At each level we use a different dimension to split





kd-tree Example





KNN: Alternate Terminologies

- Instance Based Learning
- Lazy Learning
- Case Based Reasoning
- Exemplar Based Learning

Thank You!