Instance Based Learning

- Learning Types
  - Model Based Learning
    - SVM,NN,……
  - Instance Based Learning
    - Non-Pararmetric Estimation of PDFs
      - Kernel Density Estimation (KDE)
    - K-Nearest Neighbor (kNN) as a Classifier

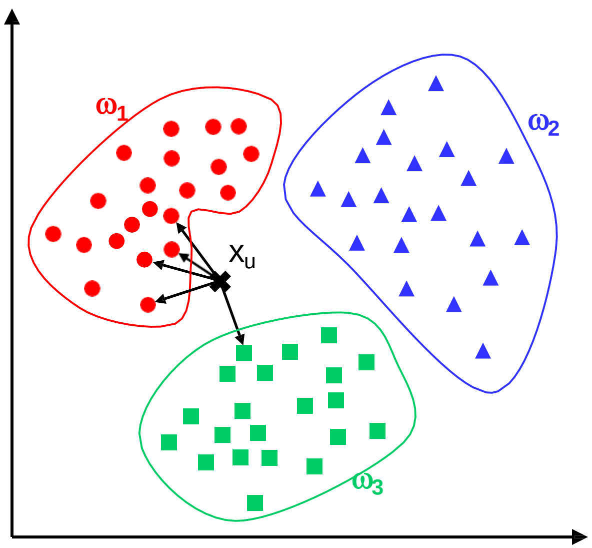
The kNN classifier

Definition

– The kNN rule is a very intuitive method that classifies unlabeled examples based on their similarity to examples in the training set

– For a given unlabeled example $x_u \in \Re^D$, find the k "closest" labeled examples in the training data set and assign x_u to the class that appears most frequently within the k-subset

– The kNN only requires
  - An integer k
  - A set of labeled examples (training data)
  - A metric to measure "closeness"

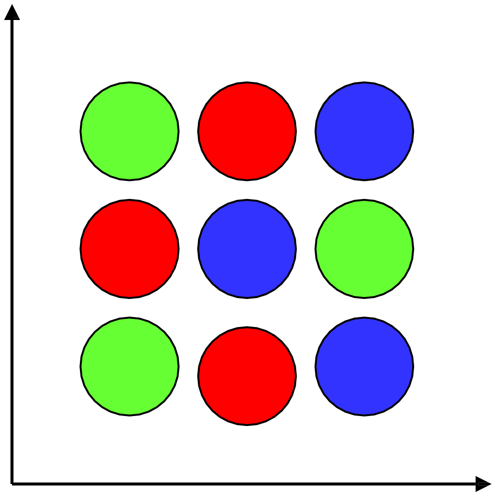


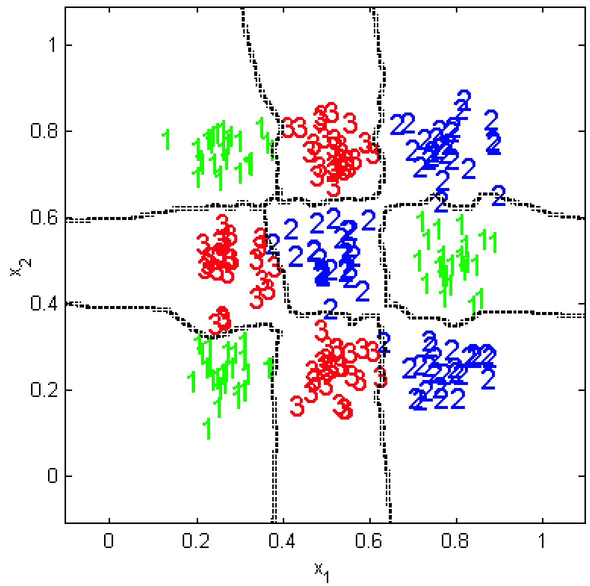
– Example
  - In the example here we have three classes and the goal is to find a class label for the unknown example x_u
  - In this case we use the Euclidean distance and a value of $k = 5$ neighbors
  - Of the 5 closest neighbors, 4 belong to ω_1 and 1 belongs to ω_3, so x_u is assigned to ω_1, the predominant class

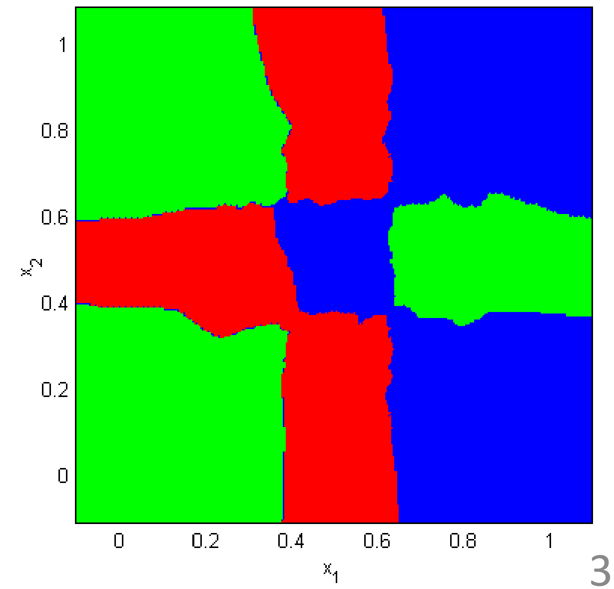
kNN in action

Example I

- Three-class 2D problem with non-linearly separable, multimodal likelihoods
- We use the kNN rule ($k = 5$) and the Euclidean distance
- The resulting decision boundaries and decision regions are shown below

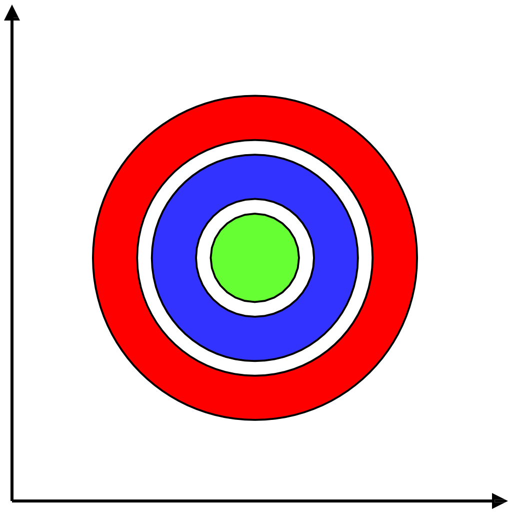


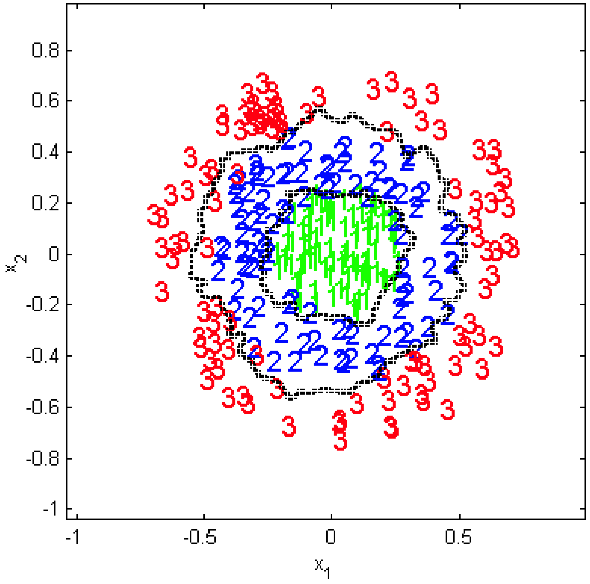


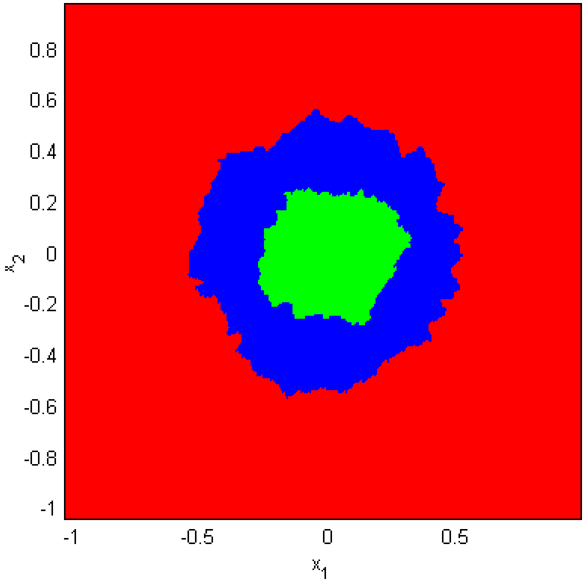


Example II

- Two-dim 3-class problem with unimodal likelihoods with a common mean; these classes are also not linearly separable
- We used the kNN rule ($k = 5$), and the Euclidean distance as a metric







kNN as a machine learning algorithm

kNN is considered a <u>lazy learning</u> algorithm

- Defers data processing until it receives a request to classify unlabeled data
- Replies to a request for information by combining its stored training data
- Discards the constructed answer and any intermediate results

This strategy is opposed to an <u>eager learning</u> algorithm which

- Compiles its data into a compressed description or model
  - A density estimate or density parameters (statistical PR)
  - A graph structure and associated weights (neural PR)
- Discards the training data after compilation of the model
- Classifies incoming patterns using the induced model, which is retained for future requests

Tradeoffs

- Lazy algorithms have fewer computational costs than eager algorithms during training
- Lazy algorithms have greater storage requirements and higher computational costs on recall

[Aha, 1997]

Characteristics of the kNN classifier

Advantages

- Analytically tractable
- Simple implementation
- Nearly optimal in the large sample limit ($N \rightarrow \infty$)

$$P_{Bayes}[error] < P_{1NN}[error] < 2P_{Bayes}[error]$$

- Uses local information, which can yield highly adaptive behavior
- Lends itself very easily to parallel implementations

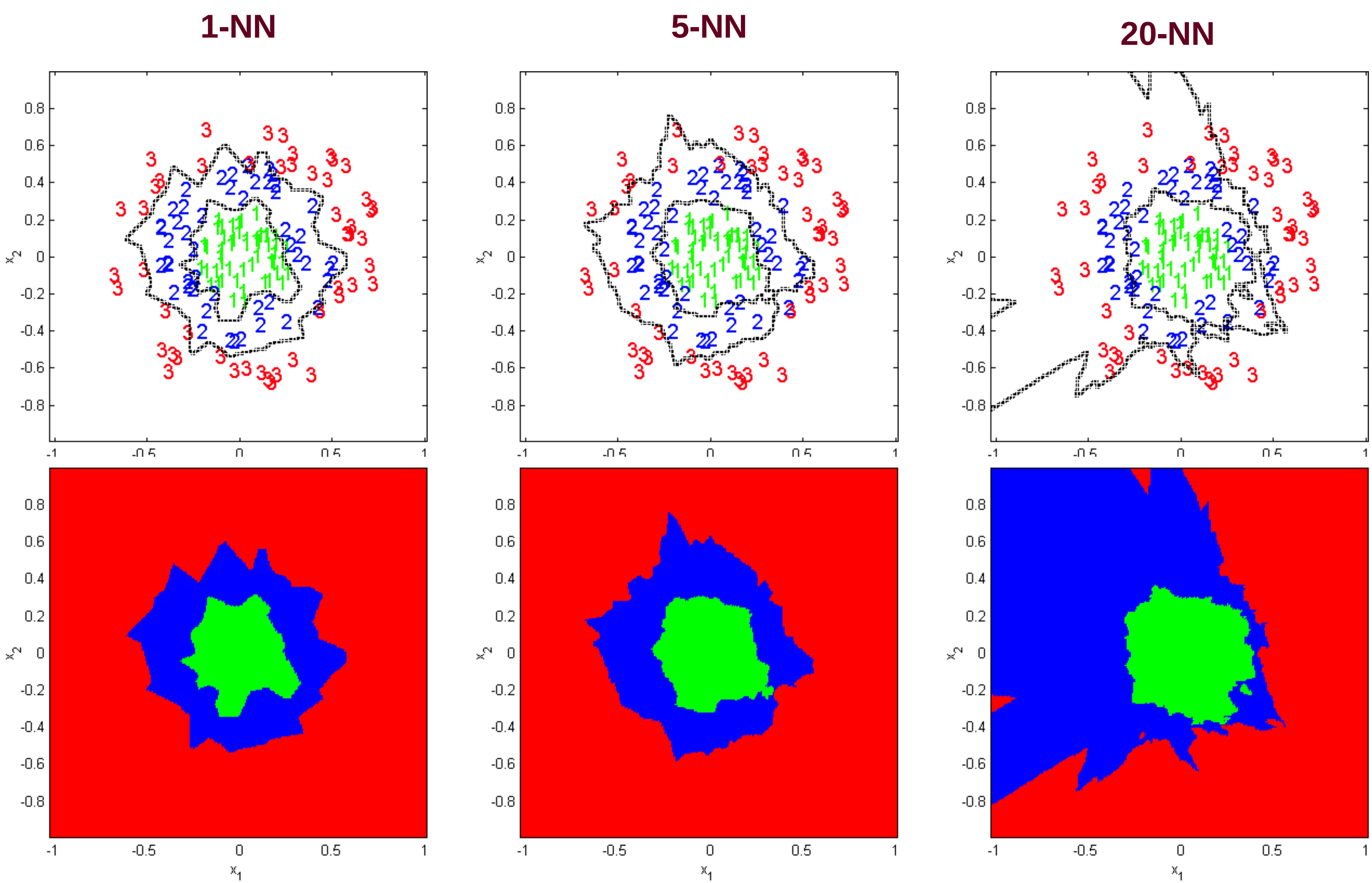
Disadvantages

- Large storage requirements
- Computationally intensive recall
- Highly susceptible to the curse of dimensionality

1NN versus kNN

- The use of large values of k has two main advantages
  - Yields smoother decision regions
  - Provides probabilistic information, i.e., the ratio of examples for each class gives information about the ambiguity of the decision
- However, too large a value of k is detrimental
  - It destroys the locality of the estimation since farther examples are taken into account
  - In addition, it increases the computational burden

kNN versus 1NN



Optimizing storage requirements

The basic kNN algorithm stores all the examples in the training set, creating high storage requirements (and computational cost)

- However, the entire training set need not be stored since the examples may contain information that is highly redundant
  - A degenerate case is the earlier example with the multimodal classes, where each of the clusters could be replaced by its mean vector, and the decision boundaries would be practically identical
- In addition, almost all of the information that is relevant for classification purposes is located around the decision boundaries

A number of methods, called edited kNN, have been derived to take advantage of this information redundancy

- One alternative [Wilson 72] is to classify all the examples in the training set and remove those examples that are misclassified, in an attempt to separate classification regions by removing ambiguous points
- The opposite alternative [Ritter 75], is to remove training examples that are classified correctly, in an attempt to define the boundaries between classes by eliminating points in the interior of the regions

A different alternative is to reduce the training examples to a set of prototypes that are representative of the underlying data

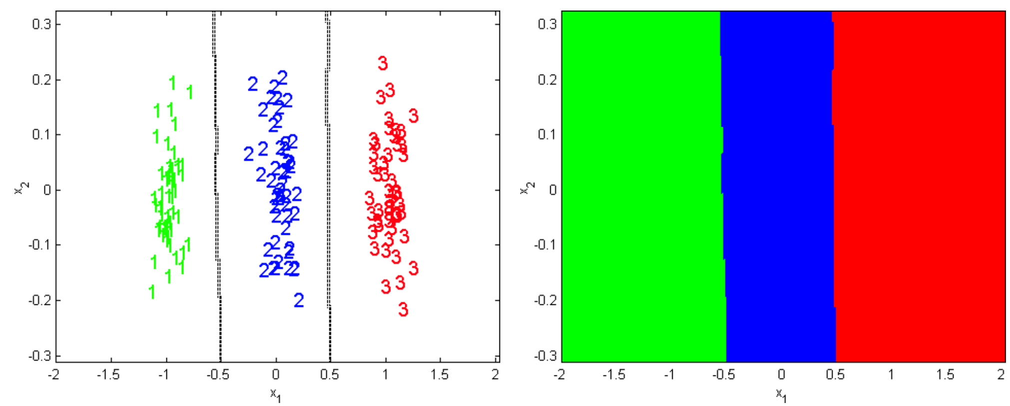
- The issue of selecting prototypes will be the subject of the lectures on clustering

kNN and feature weighting

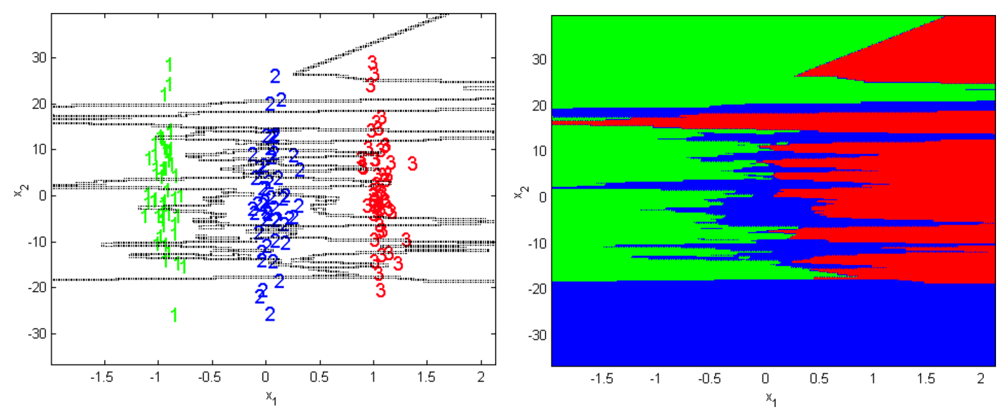
kNN is sensitive to noise since it is based on the Euclidean distance

- To illustrate this point, consider the example below
  - The first axis contains all the discriminatory information
  - The second axis is white noise, and does not contain classification information

- In a first case, both axes are scaled properly
  - kNN ($k = 5$) finds decision boundaries fairly close to the optimal



- In a second case, the scale of the second axis has been increased 100 times
  - kNN is biased by the large values of the second axis and its performance is very poor



Feature weighting

The previous example illustrated the Achilles' heel of kNN: its sensitivity to noisy features

- As a potential solution, one may attempt to normalize each feature to $N(0,1)$
- Unfortunately, the Euclidean distance (see below) becomes very noisy for high dimensional problems if only a few of the features carry the classification information

$$d(x_u, x) = \sqrt{\Sigma_{k=1}^{D}\left(x_{u,} \; - x_k \right)^2}$$

Feature weighting

- The solution is to modify the Euclidean metric by a set of weights that capture the information content or "goodness" of each feature

$$d_w(x_u, x) = \sqrt{\Sigma_{k=1}^{D}\left[w_k\left(x_{u,k} - x_k \right)\right]^2}$$

- Note this is equivalent to performing a linear transformation with a diagonal matrix
  - Hence, feature weighting is a special case of feature extraction where the features are not allowed to interact
  - In turn, feature subset selection can be viewed as a special case of feature weighting where the weights can only take binary [0,1] values

- Do not confuse feature-weighting with distance-weighting, a kNN variant that weights the contribution of each neighbor according to its distance to the unlabeled example
  - Distance-weighting distorts the kNN estimate of $P(\omega_i|x)$ and is NOT recommended
  - Studies have shown that distance-weighting does not improve kNN classification performance

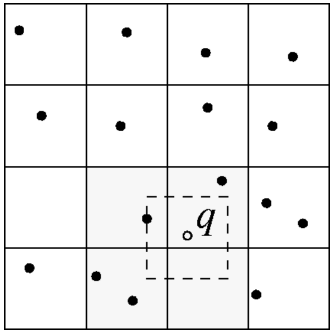
Improving the NN search procedure

The NN search procedure can be stated as follows

- Given a set of N points in D-dimensional space and an unlabeled example $x_u \in \Re^D$, find the point that minimizes the distance to x_u
- The naïve approach of computing a set of N distances, and finding the (k) smallest becomes impractical for large values of N and D

Two classical algorithms can be used to speed up the NN search

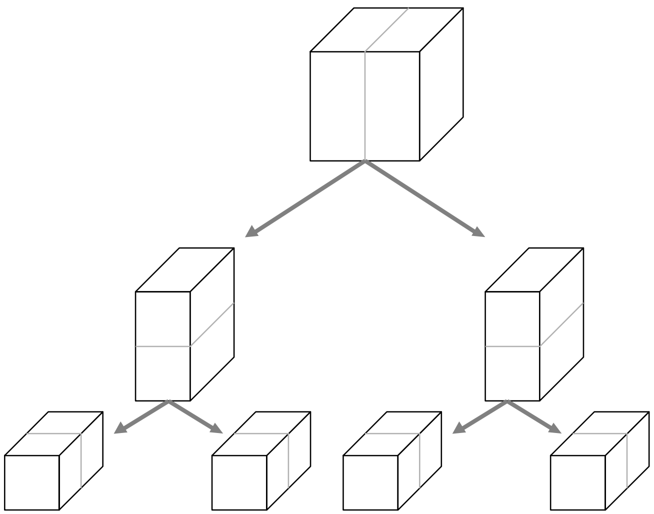
- **Bucketing** (a.k.a Elias's algorithm) [Welch 1971]
  - The space is divided into identical cells; for each cell, the data points inside it are stored in a list
  - Cells are examined in order of increasing distance from the query point; for each cell, the distance is computed between its internal data points and the query point
  - The search terminates when the distance from the query point to the cell exceeds the distance to the closest point already visited



- **k-d trees** [Bentley, 1975; Friedman et al, 1977]
  - A k-d tree is a generalization of a binary search tree in high dimensions
    - Each internal node in a k-d tree is associated with a hyper-rectangle and a hyper-plane orthogonal to one of the coordinate axis
    - The hyper-plane splits the hyper-rectangle into two parts, which are associated with the child nodes
    - The partitioning process goes on until the # data points in the hyper-rectangle falls below some given threshold
  - k-d trees partition the sample space according to the underlying distribution of the data: the partitioning being finer in regions where the density of data points is higher
    - For a given query point, the algorithm works by first descending the tree to find the data points lying in the cell that contains the query point
    - Then it examines surrounding cells if they overlap the ball centered at the query point and the closest data point so far

k-d tree example

Data structure (3D case)



Partitioning (2D case)