k – Nearest Neighbor rule

- Exemplar characterised by a set of features;
- decide class to which exemplar belongs
What is Lazy Learning

- Compare ANNs and CBR or $k$-NN classifier
  - Artificial Neural Networks are *eager* learners
  - training examples compiled into a model at training time
  - not available at runtime
  - CBR or $k$-Nearest Neighbour are *lazy*
  - little offline learning done
  - work deferred to runtime
Distance/Similarity Function

- For query $q$ and training set $X$ (described by features $F$)
- compute $d(x,q)$ for each $x$ in $X$, where $d(x,q)$ is a function calculating the similarity or distance between $x$ and $q$
- Category of $q$ decided by its $k$ Nearest Neighbours
Requirements of a distance function

• Most employed functions are the so called metrics or distance function

• Any distance function must satisfy the following requirements:
  • - no negativity: \( f(x,y) \geq 0 \)
  • - reflexitivity: \( f(c,y) = f(y,x) \)
  • - triangular unequality:
  
  \[ f(x,y) \leq f(x,z) + f(z,y) \]
Dimension reduction in $k$-NN

- Feature selection
- Not all features required
- Noisy features a hindrance
- Condensed NN
- Some examples redundant
- Retrieval time depends on number of examples
Condensed NN

- $D$ set of training samples
- Find $E$ where $E \subset D$; NN rule used with $E$
- should be as good as with $D$
- choose $x \in D$ randomly,
- $D \leftarrow D \setminus \{x\}$, $E \leftarrow \{x\}$,
- DO
  - learning? $\leftarrow$ FALSE,
  - FOR EACH $x \in D$
    - classify $x$ by NN using $E$,
    - if classification incorrect
      - then $E \leftarrow E \cup \{x\}$,
      - $D \leftarrow D \setminus \{x\}$,
    - learning $\leftarrow$ TRUE,
- WHILE (learning? $\neq$ FALSE)
Improving Condensed NN

• Sort data based on distance to nearest unlike neighbour
• Different outcomes depending on data order
  • that’s a bad thing in an algorithm
• identify exemplars near decision surface
Condensing illustration: resulting samples
Feature selection is NP

- $2n$ possible feature combinations
- powerset of all features
- two evaluation strategies
  - filtering
  - wraper
Filtering: Evaluation function

- Score *predictiveness* of features, e.g.
  - eigenvalues of covariance matrix of features
  - information theoretic analysis
  - all features appearing in a decision tree
  - some other statistical tests
  - Learning bias of evaluation different to that of classifier
  - classifier is best evaluation function
Feature Weighting

• Use *introspective* learning
• – Test training data on itself
• • For a *correct* retrieval
• – *increase* weight of matching features
• – *decrease* weight of un-matching features
Editing illustration: samples and decision boundary
Editing illustration:
Edited training sample
Basic editing procedure

• Given a training sample $R$ (of known classification), let $S$ be the set of samples misclassified by the classification 1-NN rule.

• Remove these from the training sample to form $R = R - S$ and repeat the procedure until a stopping criterion is met.

• Thus, we end up with a set of samples correctly classified by the 1-NN rule.
Computer time saving

• Development of data structures and non-exhaustive search algorithms
• Friedman algorithm:
• Preprocessing:
• samples are ordered according to their value in one of the features. It is recommended to use that feature with the greatest variability
Computer time saving (cont)

• Friedman algorithm
• Search (for the nearest neighbor of a new pattern X, in the training sample).
  • --samples in the training sample are examined according to their projected distances to X, in the chosen feature
  • --When this projected distance gets a value greater than the full distance from X to its current nearest neighbor no more samples are examined.