# Course. Introduction to Machine Learning

# Work 3. Lazy Learning exercise

## Session 2

Dr. Maria Salamó Llorente

Dept. Mathematics and Informatics,

Faculty of Mathematics and Informatics,
University of Barcelona

# Contents

1. Introduction (Session 1)
2. Parameters in k-IBL (Session 1)
    1. Distance metric
    2. K parameter
    3. Voting schemes
    4. Retention policies
3. Feature Selection techniques (Session 2)
4. Instance Selection techniques (Session 2)

# Feature Selection

# Feature Selection

- Feature selection is a process where you automatically select those features in your data that contribute most to the prediction variable or output in which you are interested
  - Having irrelevant features in your data can decrease the accuracy of many models, especially linear algorithms like linear and logistic regression

- Three benefits of performing feature selection before modeling your data are:
  - **Reduces the risk of Overfitting**: Less redundant data means less opportunity to make decisions based on noise
  - **Improves Accuracy**: Less misleading data means modeling accuracy improves
  - **Reduces Training Time**: Less data means that algorithms train faster
  - **Reduces model's complexity:** such that interpretation becomes easier

# Filter methods

- Features are selected independently from any machine algorithms
- Mainly used as pre-processing steps of any feature selection pipeline.

- Examples:
  - ANOVA F-value
    - `from sklearn.feature_selection import f_classif`
  - Variance Threshold
    - `from sklearn.feature_selection import VarianceThreshold`
  - Mutual Information
    - `from sklearn.feature_selection import mutual_info_classif`

# Wrapper methods

- Wrapper methods try to find a subset of features that yield the best performance for a model by training, evaluating, and comparing the model with different combinations of features
- Can be computationally expensive, especially if the number of features is high
- The risk of overfitting is high if the number of instances in the dataset is insufficient
- Examples:
  - Exhaustive feature selection (EFS)
    - from mlxtend.feature_selection import ExhaustiveFeatureSelector as EFS
  - Sequential forward selection (SFS)
    - from mlxtend.feature_selection import SequentialFeatureSelector as SFS
  - Sequential backward seletion (SBS)
    - from mlxtend.feature_selection import SequentialFeatureSelector as SFS, set the K_features parameter to False

# Feature Selection methods

To perform feature selection, there are different ways to do it in Python:

https://scikit-learn.org/stable/modules/feature_selection.html

Some examples are:

- Univariate Selection
- Recursive Feature Elimination
- Principle Component Analysis
- Feature Importance

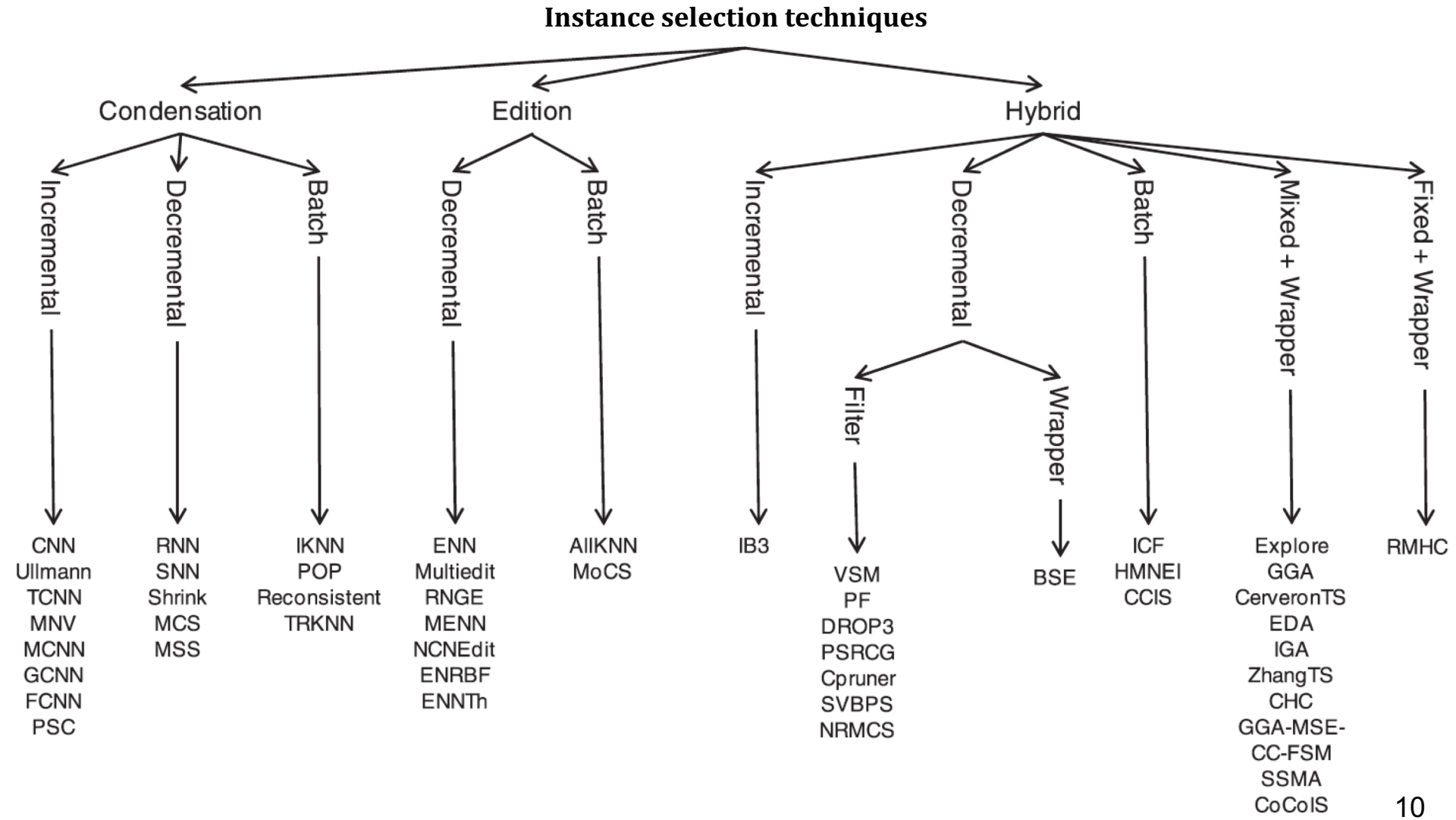
# Feature Selection in NN

You can use metrics to decide which features are not necessary to the dataset and filter them, here are some examples:

- **Information Gain or Mutual information**
  - `mutual_info_classif` available in `scikit.feature_selection` module
  - `from sklearn.feature_selection import mutual_info_classif`
- **ReliefF**
  - Class `Relief` in `sklearn_relief`
  - It implements `Relief` and `ReliefF` feature selection techniques returning a weight vector with values between 0 and 1
- **Chi2 test**
  - `from sklearn.feature_selection import chi2`

# Instance selection techniques

# Taxonomy


Instance selection techniques
Condensation
Edition
Hybrid
Incremental
Decremental
Batch
Decremental
Batch
Incremental
Decremental
Batch
Mixed + Wrapper
Fixed + Wrapper
Filter
Wrapper
CNN
Ullmann
TCNN
MNV
MCNN
GCNN
FCNN
PSC
RNN
SNN
Shrink
MCS
MSS
IKNN
POP
Reconsistent
TRKNN
ENN
Multiedit
RNGE
MENN
NCNEdit
ENRBF
ENNTh
AllKNN
MoCS
IB3
VSM
PF
DROP3
PSRCG
Cpruner
SVBPS
NRMCS
BSE
ICF
HMNEI
CCIS
Explore
GGA
CerveronTS
EDA
IGA
ZhangTS
CHC
GGA-MSE-
CC-FSM
SSMA
CoCoIS
RMHC

# Types of selection

- **Condensation**
  - Aim to retain points which are closer to the decision boundaries, also called border points
  - The idea behind these methods is to preserve the accuracy over the training set, but the generalization accuracy over the test set can be negatively affected
  - Reduction rate is normally high
- **Edition**
  - Seek to remove border points
  - Remove points that are noisy or do not agree with their neighbors
  - Reduction rate is low
- **Hybrid**
  - Try to find the smallest subset S which maintains or even increases the generalization accuracy in test data
  - It allows the removal of internal and border points

# Direction of search

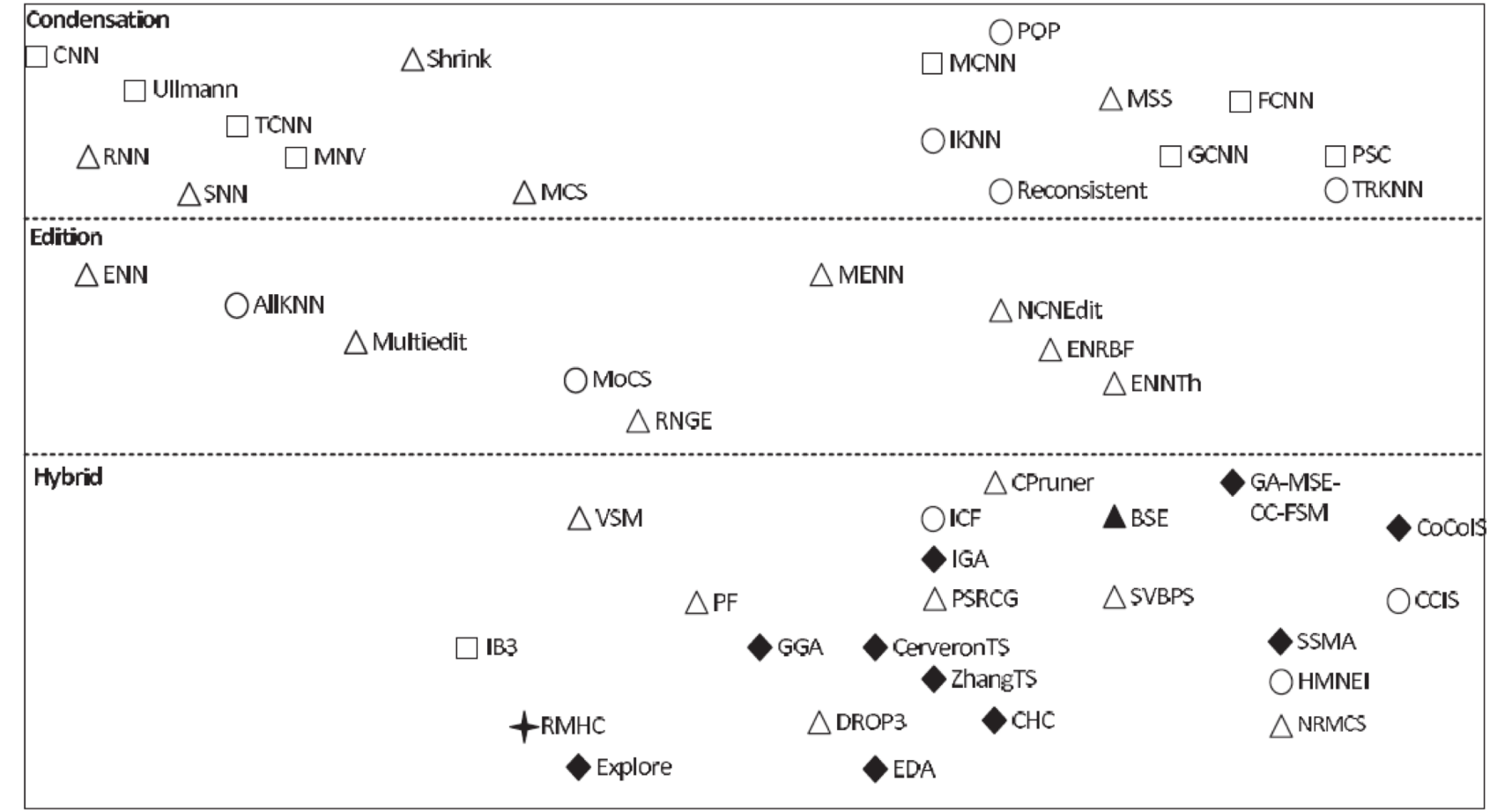
- **Incremental**
  - Starts with an empty subset S, and adds each case in the training set to S if it fulfills some criteria
  - These algorithms depend on the order of presentation
- **Decremental**
  - Begins with S= training set, and then searches for instances to remove from S
  - These algorithms depend on the order of presentation
- **Batch**
  - Use a batch mode. This involves deciding if each instance meets the removal criteria before removing any of them.
  - Then, all those that do not meet the criteria are removed at once
- **Mixed**
  - Begins with a preselected subset S (randomly or selected by an incremental or decremental process) and can iteratively add or remove any instance which meets the specific criterion

# Evaluation of search

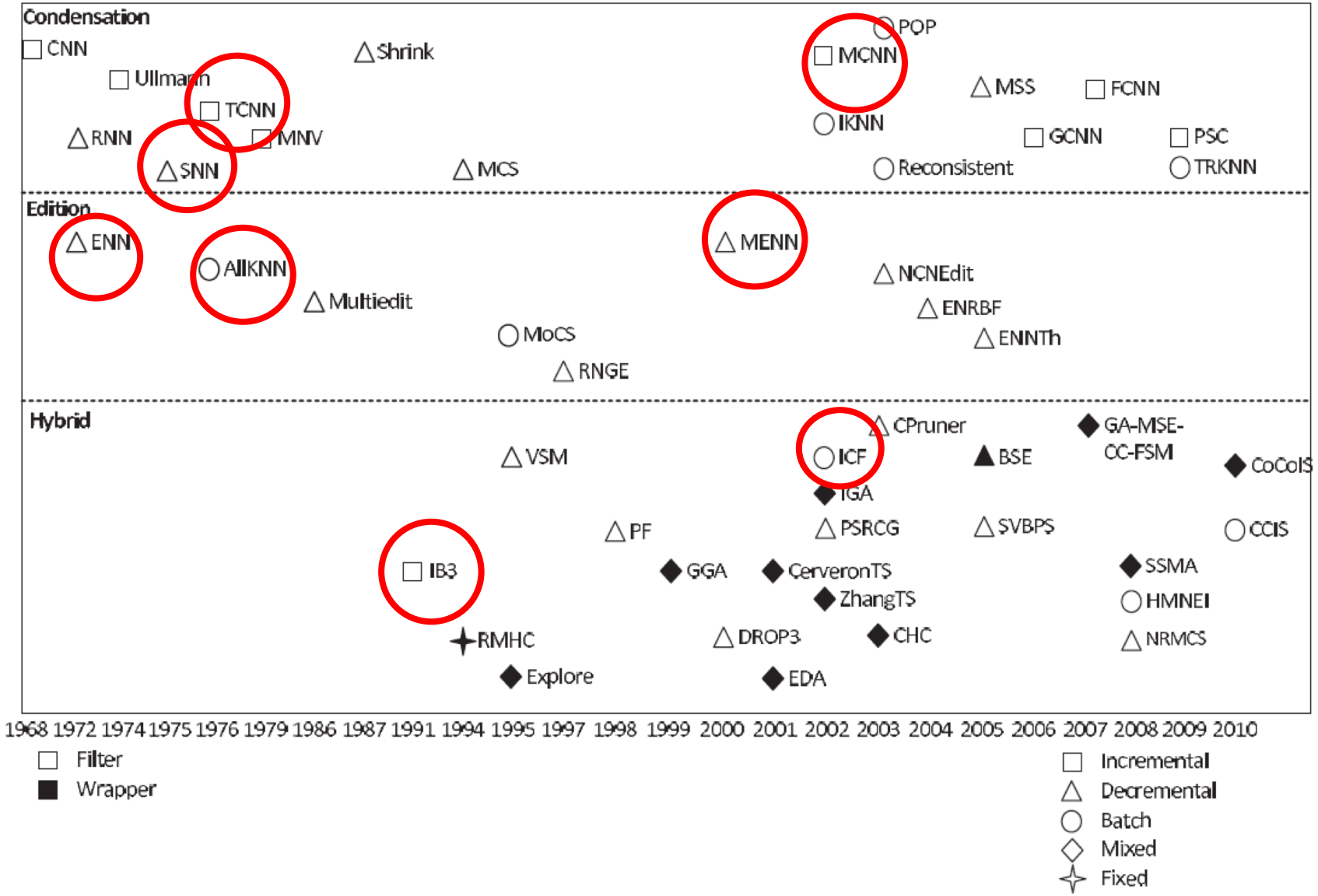
- **Filter**
  - When the kNN rule is used for partial data to determine the criteria of adding or removing and no leave-one-out validation scheme is used to obtain a good estimation of generalization accuracy
- **Wrapper**
  - When the kNN rule is used for the complete training set with the leave-one-out validation scheme
  - The conjunction in the use of the two mentioned factors allows us to get a great estimator of generalization accuracy, which helps to obtain better accuracy over test data
  - It can be computationally expensive

UNIVERSITAT DE BARCELONA

# Instance selection techniques map



# Instance selction techniques map



# Criteria to compare methods

- **Storage reduction**
  - The main goal of reduction techniques is to reduce storage requirements
  - Another goal is to speed up classification
- **Noise tolerance**
  - Two main problems may occur in the presence of noise:
    1. Few instances will be removed because many instances are needed to maintain noisy decision boundaries
    2. The generalization accuracy can suffer, especially if noisy instances are retained instead of good instances
- **Generalization accuracy**
  - A successful algorithm will be often able to significantly reduce the size of the training set without significantly reducing the accuracy
- **Time requirements**
  - If the learning phase takes too long it can become impractical for real applications

# IS techniques in Work 3

- **SNN or TCNN or MCNN**
  - **Filter** approaches based on **condensation** with **incremental** or **decremental** direction of search

- **SNN**: G.L. Ritter, H.B. Woodruff, S.R. Lowry, and T.L. Isenhour, "An Algorithm for a Selective Nearest Neighbor Decision Rule," IEEE Trans. Information Theory, vol. 21, no. 6, pp. 665-669, Nov. 1975

- **TCCN**: I. Tomek, "Two Modifications of CNN," IEEE Trans. Systems, Man, and Cybernetics, vol. 6, no. 6, pp. 769-772, Nov. 1976.

- **MCNN**: V.S. Devi and M.N. Murty, "An Incremental Prototype Set Building Technique," Pattern Recognition, vol. 35, no. 2, pp. 505-513, 2002
  https://www.sciencedirect.com/science/article/abs/pii/S0031320300001849

# CNN

- **CNN Family**
  - ***Condensed Nearest-Neighbor rule* (CNN)**
    - build an edited set from scratch by adding instances that cannot be successfully solved by the edited set built so far
    - tends to select training instances near the class boundaries
    - consistent
    - not minimal edited set (redundant instances) : order dependent
  - ***Reduced Nearest-Neighbor* (RNN) method**
    - adaptation of CNN
    - postprocess to contract the edited set by identifying and deleting redundant instances

# CNN

- **CNN Family**
  - ***SNN***
    - Extended CNN such that every member of T must be closer to a member of S of the same class than to any member of T (instead of S) of a different class
  - ***TCNN***
    - Modifies the CNN method by considering only points close to the boundary
    - Two variants were proposed, **implement the first one**.
  - **MCNN**
    - It is an incremental algorithm that takes misclassified patterns at each stage and finding instances to classify these patterns correctly
    - The algorithm is order independent and fast
    - A deletion operation is also used to reduce the number of instances by removing instances that do not participate very much in the classification procedure

# IS techniques in Work 3

- **ENN or allKNN or MENN**
  - **Filter** approaches based on **edition** with **decremental** direction of search
  - **ENN**: D.L. Wilson, “Asymptotic Properties of Nearest Neighbor Rules Using Edited Data,” IEEE Trans. Systems, Man, and Cybernetics, vol. 2, no. 3, pp. 408-421, July 1972
  - **allKNN:** I. Tomek, “An Experiment with the Edited Nearest-Neighbor Rule,” IEEE Trans. Systems, Man, and Cybernetics, vol. 6, no. 6, pp. 448-452, June 1976.
  - **MENN**: K. Hattori and M. Takahashi, “A New Edited K-Nearest Neighbor Rule in the Pattern Classification Problem,” Pattern Recognition, vol. 33, no. 3, pp. 521-528, 2000

# ENN

- **Edited Nearest Neighbor**
  - perfect counterpoint to CNN
  - filter out incorrectly classified instances in order to remove boundary instances (and noise) and preserve interior instances that are representative of the class being considered

**Procedure**

- begin with all training instances
- removed if its classification is not the same as the majority classification of its *k* nearest neighbors (edits out the noisy and boundary instances)
- suffer from redundancy problem

# RENN and All-kNN

- **RENN (repeated ENN)**
  - repeatedly applying ENN until all instances have the majority classification of their neighbors
  - the effect of widening the gap between classes and smoothing the decision boundaries
- **All-kNN**
  - It increases the value of k for each iteration of RENN
  - For i=0 to k flags as bad any instance not classified correctly by its i nearest neighbors. When the loop is completed k times, it removes the instances flagged as bad
  - the effect of removing boundary instances and preserving interior

# IB-Algorithms

- **IB1**: store all examples
  - High noise tolerance
  - High memory demands
- **IB2**: Store examples that are misclassified by current example set
  - Low noise tolerance
  - Low memory demands
- **IB3**: like IB2 but,
  - Maintain a counter for the number of times the example participated in correct and incorrect classifications
  - Use a significant test for filtering noisy examples
    - Improved noise tolerance
    - Low memory demands

# IB3

- IB3 is an extension of IB1
  - Deal with noise, keep only good classifier data points
  - Discard instances that do not perform well
- Algorithm:
  - Keep a record of the number of correct and incorrect classification decisions that each saved data point makes
  - Two predetermined thresholds are set on success ratio
  - An instance is selected to be used for training:
    - If the number of incorrect classifications is ≤ the first (lower) threshold and,
    - If the number of correct classifications is ≥ the second (upper) threshold

# ICF

- Iterative Case Filtering (ICF)
  - It identifies instances that should be deleted
  - Defines local set L(X) which contain all cases inside largest hypersphere centered in $X_i$ such that the hypersphere contains only cases of the same class as instance $X_i$
  - Two properties: **reachability** and **coverage**

$$Coverage(X_i) = \{X'_i \in TR : X_i \in L(X'_i)\},$$

$$Reachability(X_i) = \{X'_i \in TR : X'_i \in L(X_i)\},$$

*TR is the training set*

  - **ICF uses the reachable and coverage sets**

# ICF

```
ICF(T)
1   > Perform Wilson Editing
2   for all x ∈ T do
3       if x classified incorrectly by k nearest neighbours then
4           flag x for removal
5   for all x ∈ T do
6       if x flagged for removal then T = T − {x}
7   > Iterate until no cases flagged for removal:
8   repeat
9       for all x ∈ T do
10          compute reachable(x)
11          compute coverage(x)
12      progress = false
13      for all x ∈ T do
14          if |reachable(x)| > |coverage(x)| then
15              flag x for removal
16              progress = true
17      for all x ∈ T do
18          if x flagged for removal then T = T − {x}
19  until not progress
20  return T
```

# ICF

- Iterative Case Filtering (ICF)
  - In the first phase ICF uses the ENN algorithm to remove the noise from the training set.
  - In the second phase the ICF algorithm removes each instance $x_i$ for which the Reachability($x_i$) is bigger than the Coverage($x_i$). This procedure is repeated for each instance in the training set.
  - After that ICF recalculates reachability and coverage properties and restarts the second phase.

# Reduction techniques in Work 3

- **IB3 or ICF**
  - **Filter** hybrid approaches
  - **IB3**: D.W. Aha, D. Kibler, and M.K. Albert, “Instance-Based Learning Algorithms,” Machine Learning, vol. 6, no. 1, pp. 37-66, 1991
  - **ICF**: H. Brighton and C. Mellish, “Advances in Instance Selection for Instance-Based Learning Algorithms,” Data Mining and Knowledge Discovery, vol. 6, no. 2, pp. 153-172, 2002.

# References


- Janez Demšar. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. J. Mach. Learn. Res. 7 (December 2006), 1-30 (**MANDATORY READING**)

- Wilson, D.R., Martínez, T.R., 1997. Improved heterogeneous distance functions. Journal of Artificial Intelligence Research 6, 1–34
  - **This paper details the basis of the CNN family, ENN family, IB family and drop family**

- David W. Aha, Dennis Kibler, and Marc K. Albert. 1991. Instance-Based Learning Algorithms. Machine Learning. 6, 1 (January 1991), 37-66

- R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In Proceedings of the International Joint Conferences on Artificial Intelligence IJCAI-95. 1995

# Course. Introduction to Machine Learning

# Work 3. Lazy Learning exercise

## Session 2

Dr. Maria Salamó Llorente

Dept. Mathematics and Informatics,

Faculty of Mathematics and Informatics,
University of Barcelona