Numerosity Reduction

Department of Computer Languages and Systems

We already know …

General objective of data reduction: to obtain a reduced representation of the data that is much smaller in size and make it feasible for analysis

We need to reduce the data because …

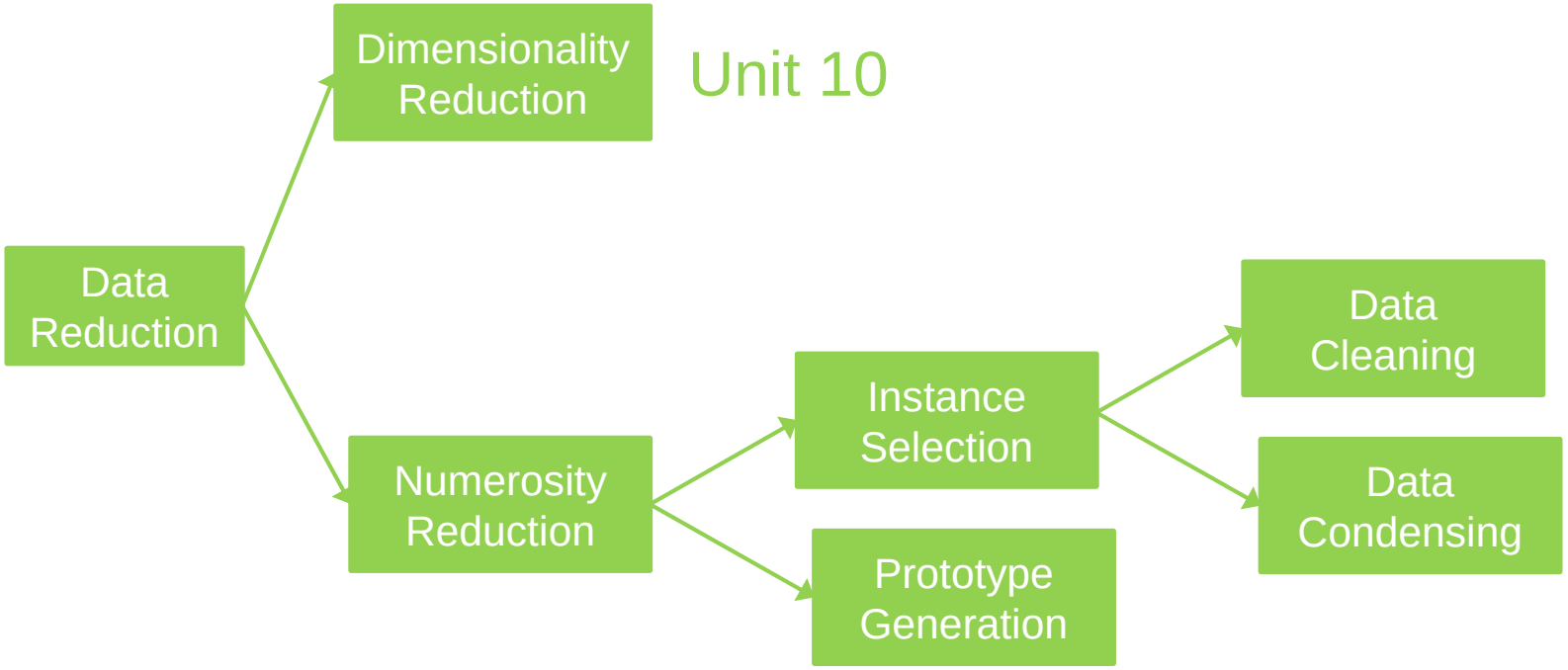
- data analysis could be too complex with huge data sets
- data analysis could be too computational expensive with huge data sets
- noisy instances can lead to misclassifications
- irrelevant and redundant attributes can hinder any machine learning process

We already know …

Data reduction is simply ..

- the reduction in the number of attributes or
- the reduction in the number of instances or ←
- the reduction in both the number of instances and the number of attributes ←

Categorization



Instance selection

According to the objectives, instance selection can be divided into two groups:

- To improve the performance of the model → Data cleaning
- To reduce the size of the data set → Data condensing

Instance selection (ii)

How to achieve those objectives?

- Data cleaning: by identifying the incorrect, incomplete and inaccurate objects and then modifying, replacing or deleting them
  - Noisy data (instances)
  - Missing data (attributes)
- Data condensing: by identifying and removing the irrelevant and redundant objects in a data set

Noisy data

What is noisy data?

- Errors that occur during the collection process. These errors can include both erroneous attribute values (attribute noise) in samples and/or erroneous class labels (class noise)
  - Class noise is generally considered more harmful to the learning process, and can be due to insufficient or imperfect information, subjectivity or the use of non-domain experts in data labelling, encoding or communication problems, etc.

Noisy data (ii)

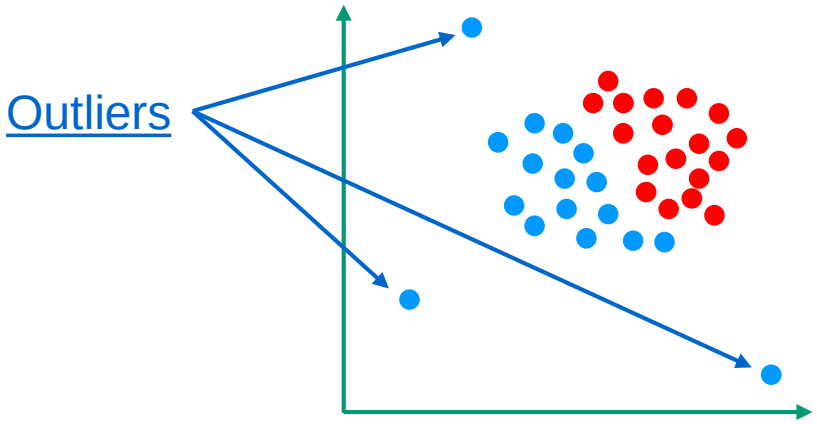
How can class-noisy data affect the model if not tackled properly?

- it may deteriorate the performance of classifiers
- it may disrupt the class boundaries and increase overlap
- it may lead to overfitting
- it makes the problem more complex and the solution less understandable

Noisy data (iii)

What is an outlier?

- It is data that differs considerably from all or most other data in a data set → they correspond to rare, abnormal or atypical cases
- It is more likely to occur outliers with large data sets, but their impact is less on the results



Noisy data (iv)

A simple method to detect outliers:

Compute the centroid of each class, C_i

for each x_i in T_{tra} :

Calculate the distance between each instance x_i and its centroid C_i, $d((x_i, C_i)$

Calculate the distance between the nearest instance nn_i to the centroid C_i, $d((nn_i, C_i)$

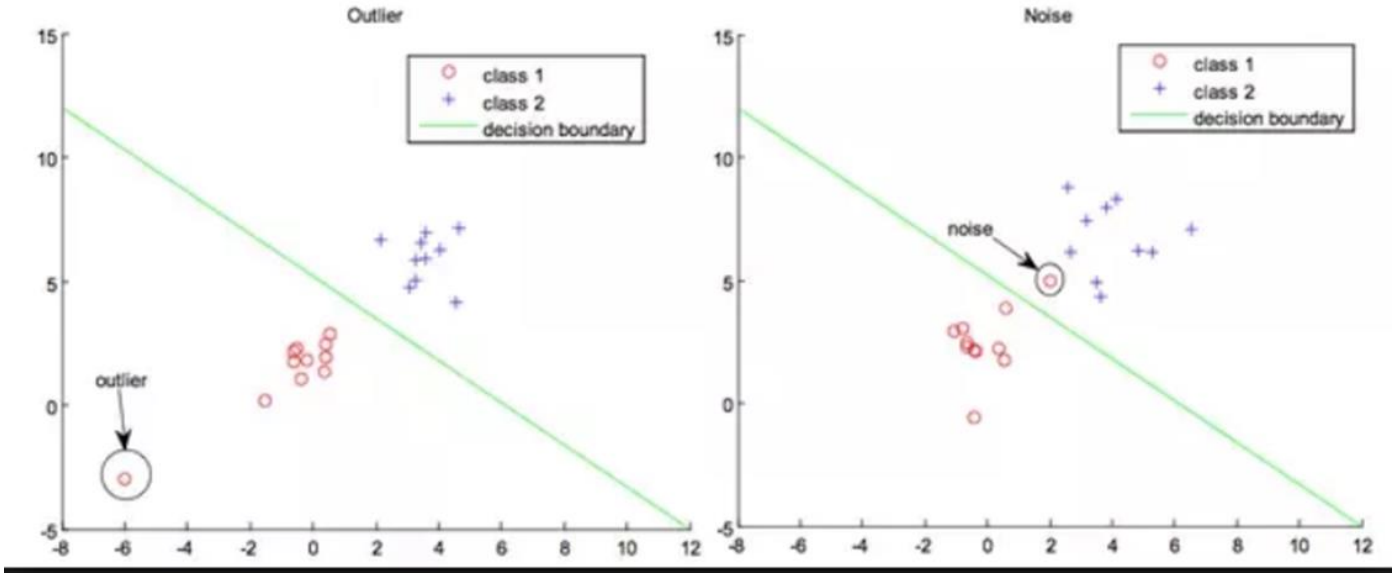
If $d((nn_i, C_i)/d((x_i, C_i) < \varepsilon$ (a threshold)

x_i is an outlier

Noisy data (v)

Difference between outlier and class-noise

- Both outliers and class-noise can be produced by errors. However, outliers can be also due to rare, but correct behavior that often results in interesting findings



Noisy data (vi)

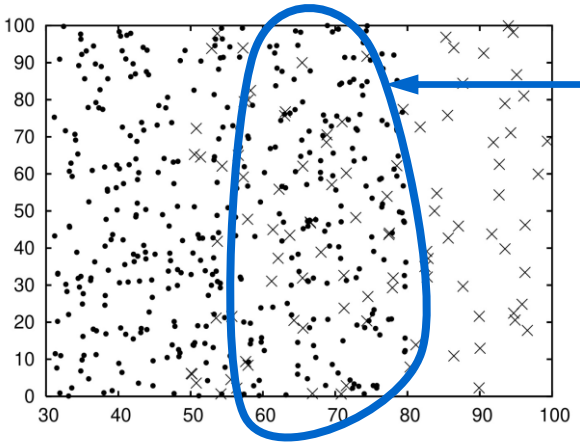
Strategies for coping with noise:

- Algorithm-level approaches aim to design robust (e.g., C4.5 decision tree) or noise-tolerant (e.g., boosting) classifiers. They do not require any previous noise treatment
- Data-level approaches attempt to remove or correct the noise present in the data before applying any standard classification algorithm. This is the best option if using a robust learner is unfeasible or inappropriate, or even aiming to improve results of robust learners
  - Filtering (or editing) methods
  - Relabelling methods

Noisy data (vii): Filtering

Data-level approaches:

- Filtering methods identify and remove noisy instances and 'clean' the possible overlap between classes → as a by-product, they obtain a reduction in the data set size
  - In general, the filtering or editing methods attempt to remove border instances (i.e., those close to the decision boundary) → this leads to smoothing the boundaries of the classification regions



Noisy data (viii): Filtering

Wilson's editing (T_{tra}, k)

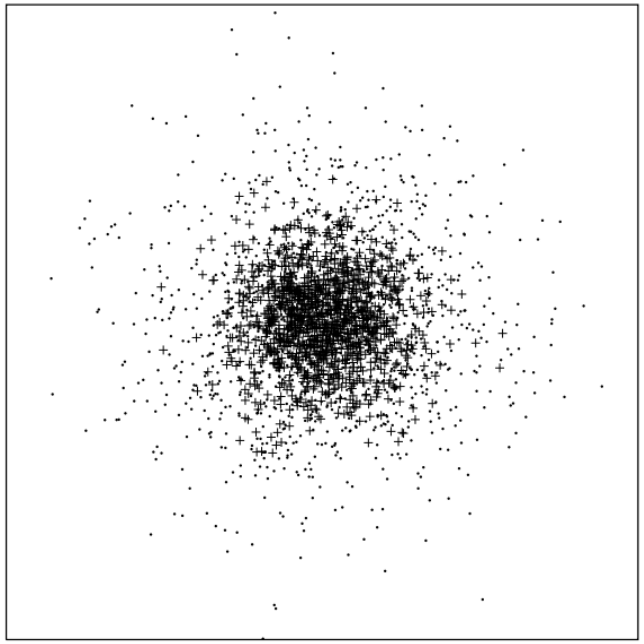
$S \leftarrow T_{tra}$
for each x_i in T_{tra} :
search for the k nearest neighbours to x_i in $T_{tra} - \{x_i\}$
if the class given by the k-NN $\neq c_i$
$S \leftarrow S - \{x_i\}$
return S

- Note that it uses leaving-one out
- Computational cost is O(n^2)

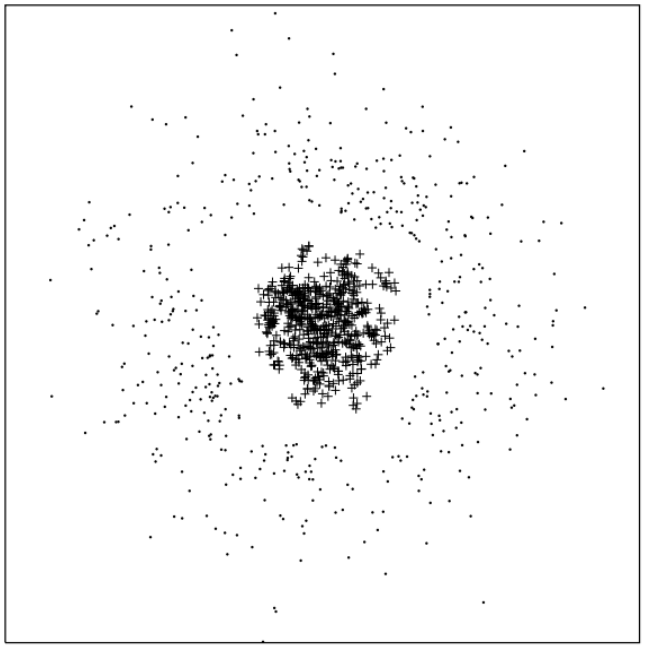
D.L. Wilson (1973) Asymptotic properties of nearest neighbor rules using edited data. *IEEE Trans. on Systems, Man, and Cybernetics* 2, 408-421

Noisy data (ix): Filtering

Wilson's editing: an example



Original set with 2500 instances



Edited set with 1105 instances

Noisy data (x): Relabelling

Data-level approaches:

- Relabelling methods consists of changing the class labels of training instances that are suspected to be noisy and belong to different classes

Noisy data (xi): Relabelling

Editing with relabelling (T_{tra}, k, l)

$S \leftarrow T_{tra}$
for each x_i in T_{tra} :
search for the k nearest neighbours to x_i in $T_{tra} - \{x_i\}$
if the class given by the (k,l)-NN $\neq c_i \neq c_0$
relabel x_i with the class given by (k,l)-NN
else if the class given by the (k,l)-NN = c_0
$S \leftarrow S - \{x_i\}$
return S

Noisy data (xii)

There exist many other data-level algorithms:

- Multiedit
- Iterative partitioning filter
- Graph-based editing (Gabriel graph, RNG)
- Fuzzy rough prototype selection
- Reward-punishment editing
- Model class selection
- Iterative case filtering
- Edited normalized radial basis function
- Etc.

Missing data

What is missing data?

- Values or data not stored for some attributes due to human errors during data entry, system malfunction during data collection, respondents refusal to answer certain questions, etc.

PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	male	22	1	0	A/5 21171	7.25		S
2	1	1	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	female	26	0	0	STON/O2. 3101282	7.925		S
4	1	1	female	35	1	0	113803	53.1	C123	S
5	0	3	male	35	0	0	373450	8.05		S
6	0	3	male		0	0	330877	8.4583		Q

Missing data (ii)

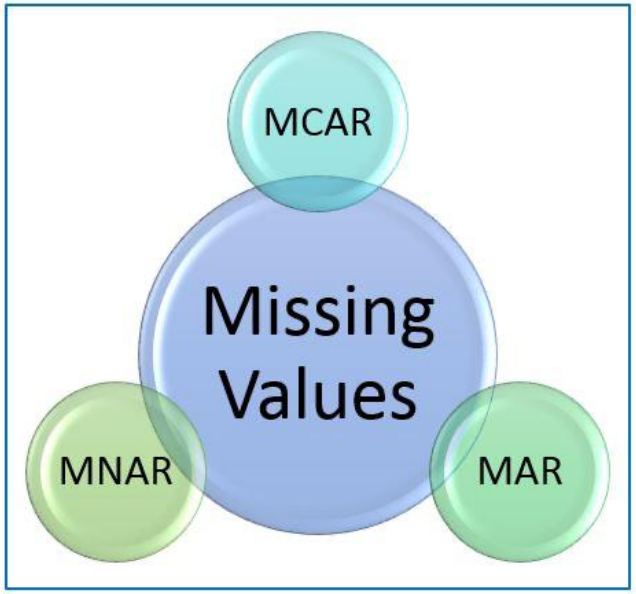
How can missing data affect the model if not tackled properly?

- it may lead to a performance degradation
- it may lead to biased machine learning models

Most ML models fail if the data set contains missing values!

Missing data (iii): Types

- Missing Completely At Random (MCAR): there is no relationship between the missing data and any other values observed within the data set
- Missing At Random (MAR): the reason for missing values can be explained by variables on which you have complete information as there is some relationship between the missing data and other values
- Missing Not At Random (MNAR): if the missing data does not fall under MCAR or MAR, then it can be categorized as MNAR



Missing data (iv): Examples of types

- MCAR: suppose a library where some values of overdue books in the system are missing, maybe because the librarian forgot to type in the values. So, the missing values of overdue books are not related to any other variable in the system
- MAR: suppose a survey in which all the people have answered their ‘Age’ but ‘IQ Score’ values are mostly missing for older people. In this case, the variables ‘Age’ and ‘IQ Score’ are related and the reason for missing values of the ‘IQ Score’ variable can be explained by the ‘Age’ variable
- MNAR: suppose the name and the number of overdue books are asked in the poll for a library. So, most of the people having no overdue books are likely to answer the poll. People having more overdue books are less likely to answer the poll

Missing data (v): Techniques

- Deletion: all objects with missing values in some attributes are removed
  - If the count of missing values in the data is very high, then deletion can lead to a very significant decrease in the number of instances
  - we may end up deleting some potentially useful information
- Imputation: it involves replacing missing values by some predicted values, typically using the non-missing values

Missing data (vi): Techniques

Deletion techniques

- Deletion of entire row (object): if an instance is containing many attributes with missing values then it is unlikely to add any value to our model → also known as listwise deletion
- Deletion of the entire column (attribute): if a certain attribute has a high majority of instances with missing values, then we can drop the entire column

Missing data (vii): Techniques

Imputation techniques

- Replacing with mean: if the data in the attribute is skewed or there are outliers, replacing missing values by mean will add bias to the data
- Replacing with median: a safe choice if the data in attribute is skewed with outliers
- Replacing with mode (the most frequent value): this technique can be applied to both numerical and categorical data

Missing data (vii): Techniques

Other imputation techniques

- Replacing with previous value (forward fill): imputing the values with the previous value instead of mean, mode or median. It is mostly used in time series data
- Replacing with next value (backward fill): the missing value is imputed using the next value
- Nearest neighbours imputation: the *k*-NN model with the Euclidean distance is used to replace the missing values with the value of the nearest neighbours
- Regression-based imputation: the missing values are estimated with linear, logistic or stochastic regression

Data condensing

Objective

- to reduce the size of the training set, but …
- it should not worsen the performance of the classifier

Result

- it produces a reduced set that can classify (equally) well as the original training set
- storage and computing requirements are lowered

Data condensing (ii)

How condensing algorithms work?

- they identify and remove irrelevant and redundant instances from a training set
  - In general, the condensing methods attempt to remove internal instances (i.e., those far from the decision boundary) because they can be safely removed without loss of accuracy

Data condensing (iii)

Training-set consistency

– Given a non-empty set T ($T \neq \varnothing$), a subset S of T ($S \subseteq T$) is consistent with respect to T if, using the subset S as training set, the 1-NN classifier can correctly classify all instances in T (100% training accuracy)

P.E. Hart (1968) The Condensed Nearest Neighbor Rule. *IEEE Trans. on Information Theory* 14, 515-516

Data condensing (iv)

Hart's condensing (T_{tra})

$S \leftarrow \varnothing$
repeat
  for each x_i in T_{tra} :
    search for the nearest neighbour to x_i in S
    if the class given by the *k*-NN $\neq c_i$
      $T \leftarrow T_{tra} - \{x_i\}$
      $S \leftarrow S + \{x_i\}$
until $T = \varnothing$ or no more removals from T
return S

P.E. Hart (1968) The Condensed Nearest Neighbor Rule. *IEEE Trans. on Information Theory* 14, 515-516

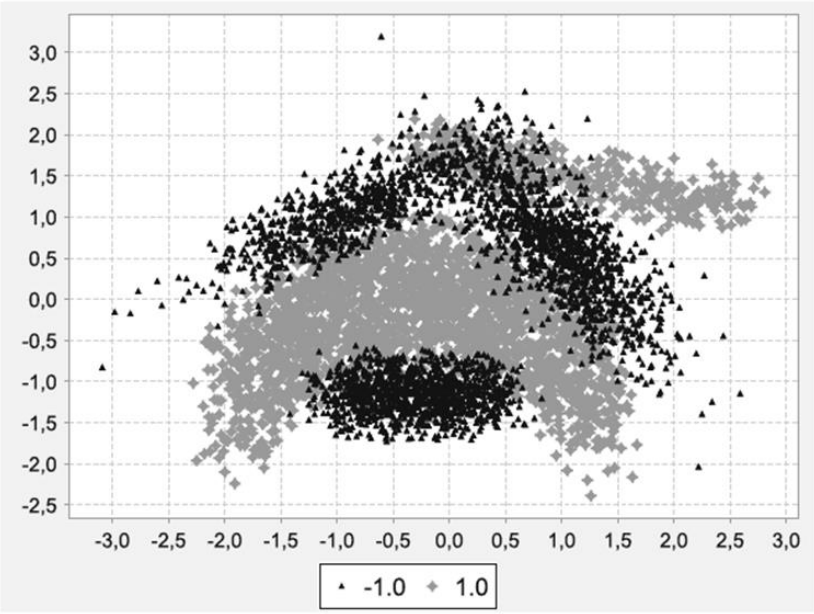
Data condensing (v)

Hart's condensing:

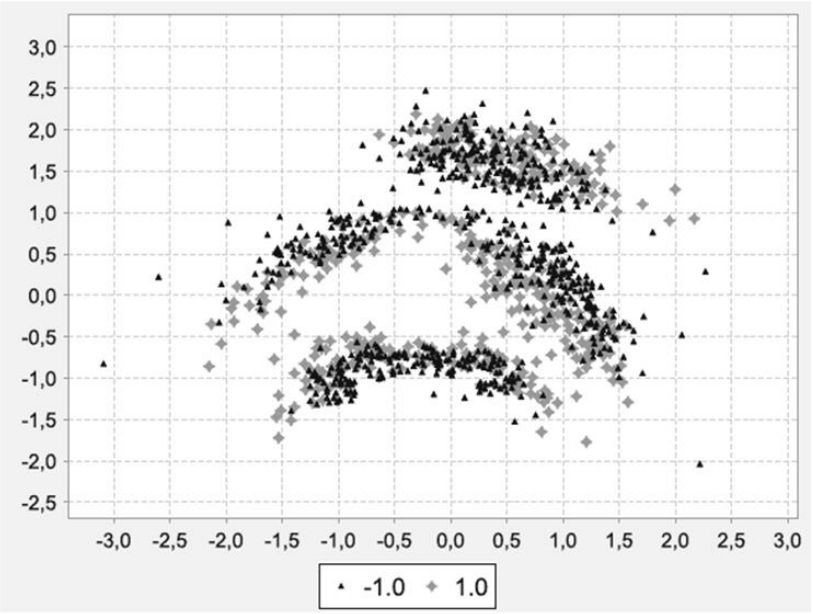
- it removes internal instances → those instances that are not needed to correctly classify the remaining instances in the training set
- it is not a deterministic approach → the content and cardinality of the resulting set depend on the order in which the instances are processed
- it produces a reduced set, but minimality cannot be guaranteed

Data condensing (vi)

Hart's condensing: an example



Banana Original (,0.8751)



CNN (0.7729,0.8664)

Data condensing (vii)

There exist many other data condensing algorithms:

- Tomek links
- Minimal consistent set
- Patterns by ordered projection
- Decremental reduction optimization procedure (DROP1-5)
- Adaptive threshold-based instance selection
- Mutual neighborhood value
- Graph-based condensing (Gabriel graph, RNG)
- Spectral instance reduction
- Etc.

Prototype generation

Similar objective to that of data condensing methods

- to create a small-sized set of exemplars (or prototypes) from the original training set, but …
- it should maintain the accuracy of the classifier

However,

- prototype generation algorithms are not limited to just selecting instances from the training set as they can also modify their location in *d*-dimensional space or generate artificial prototypes
- most algorithms use merging or divide-and-conquer strategies to create the artificial prototypes, or rely on clustering approaches

Prototype generation (ii)

What characteristics should the prototypes have?

- they are shaped to represent class distributions efficiently and to discriminate well between classes
- its cardinality should be small enough to reduce both storage and computing time of the classifier

Prototype generation (iii)

Strategies:

- Centroid-based methods generate artificial prototypes by merging similar instances of the same class in successive iterations. The artificial prototypes correspond to the centroids of the merged instances
- Positioning adjustment methods generate the artificial prototypes by 'correcting' the position of the training instances through an optimization procedure
- Space partitioning methods rely on different heuristics to partition the feature space into several regions and generate the artificial prototypes by using representative examples

Prototype generation (iv)

Some algorithms:

- Centroid-based methods
  - H.A. Fayed, S.R. Hashem, A. F. Atiya (2007) Self-generating prototypes for pattern classification. *Pattern Recognition* 40, 1498-1509
  - S. Ougiaroglou, G. Evangelidis (2014) RHC: non-parametric cluster-based data reduction for efficient k-NN classification, *Pattern Analysis and Applications* 19, 93-109
- Positioning adjustment methods
  - T. Kohonen (1990) The self organizing map. *Proceedings of the IEEE* 78,146-1480
  - I. Triguero, S. García, F. Herrera (2010) IPADE: Iterative prototype adjustment for nearest neighbor classification. *IEEE Trans. on Neural Networks* 21, 1984-1990
- Space partitioning methods
  - J.S. Sánchez (2004) High training set size reduction by space partitioning and prototype abstraction. *Pattern Recognition* 37, 1561-1564
  - T. Raicharoen, C. Lursinsap (2005) A divide-and-conquer approach to the pairwise opposite class-nearest neighbor (POC-NN) algorithm. *Pattern Recognition Letters* 26, 1554-1567

Prototype generation (v)

Class homogeneity

- A non-empty set T ($T \neq \varnothing$) is said to be class homogeneous if it does not contain a mixture of instances that belong to different classes

Diameter of a set

- The diameter of a non-empty set T ($T \neq \varnothing$) is defined as the distance between its two farthest instances

J.S. Sánchez (2004) High training set size reduction by space partitioning and prototype abstraction. *Pattern Recognition* 37, 1561-1564

Prototype generation (vi)

Reduction through Homogeneous Clustering (RHC)

$S \leftarrow \varnothing$
$R \leftarrow T_{tra}$
repeat
    clusters $\leftarrow$ C-means(R, classes_in(R))
    for each C_i in *clusters*
        if C_i is class-homogeneous
            $S \leftarrow$ centroid(C_i)
            $R \leftarrow R - \{x_j\}, x_j \in class_i$
until $R = \varnothing$
return S

Prototype generation (vii)

Reduction by Space Partitioning v3 (RSP3)

1. Compute the diameter of T_{tra} given by the two farthest instances, say p_1 and p_2

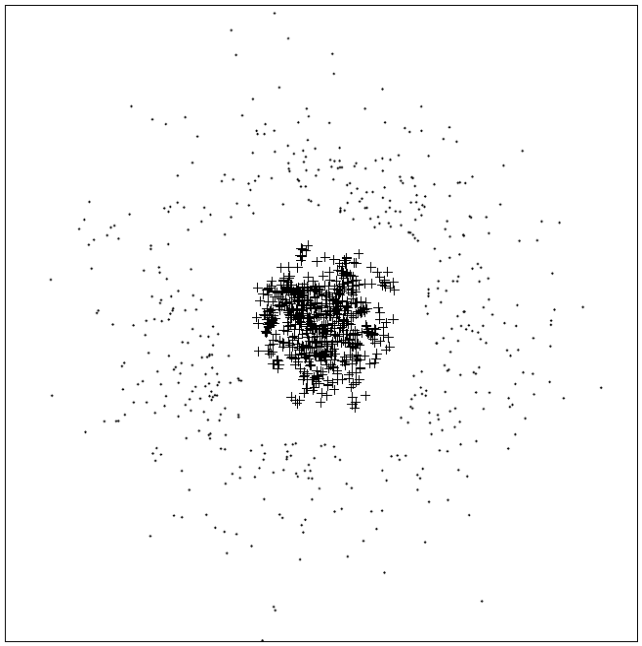
2. Form two clusters by assigning each training instance to its closest farthest instance:

$$C_1 = \{x_i \in T_{tra} : d(x_i, p_1) \leq d(x_i, p_2)\}$$
$$C_2 = \{x_i \in T_{tra} : d(x_i, p_2) < d(x_i, p_1)\}$$

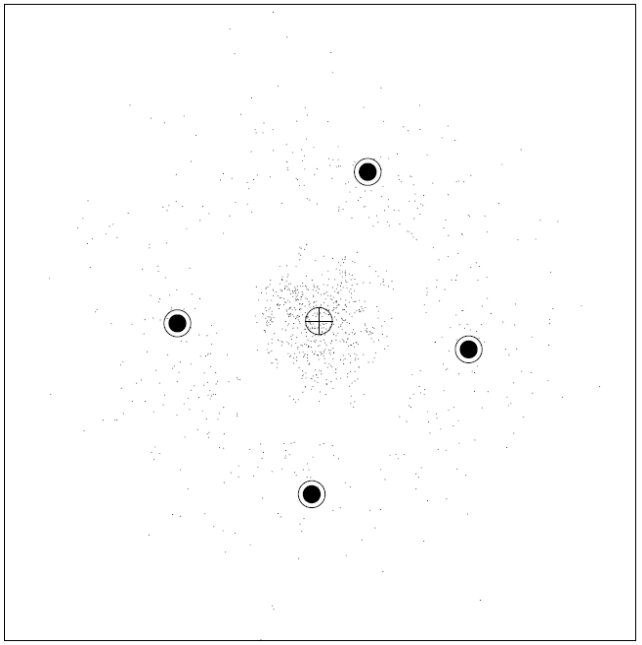
These steps are applied recursively on each non-homogeneous cluster. At the end, each homogeneous cluster is replaced by its centroid to form the reduced set

Prototype generation (viii)

Example of a centroid-based method



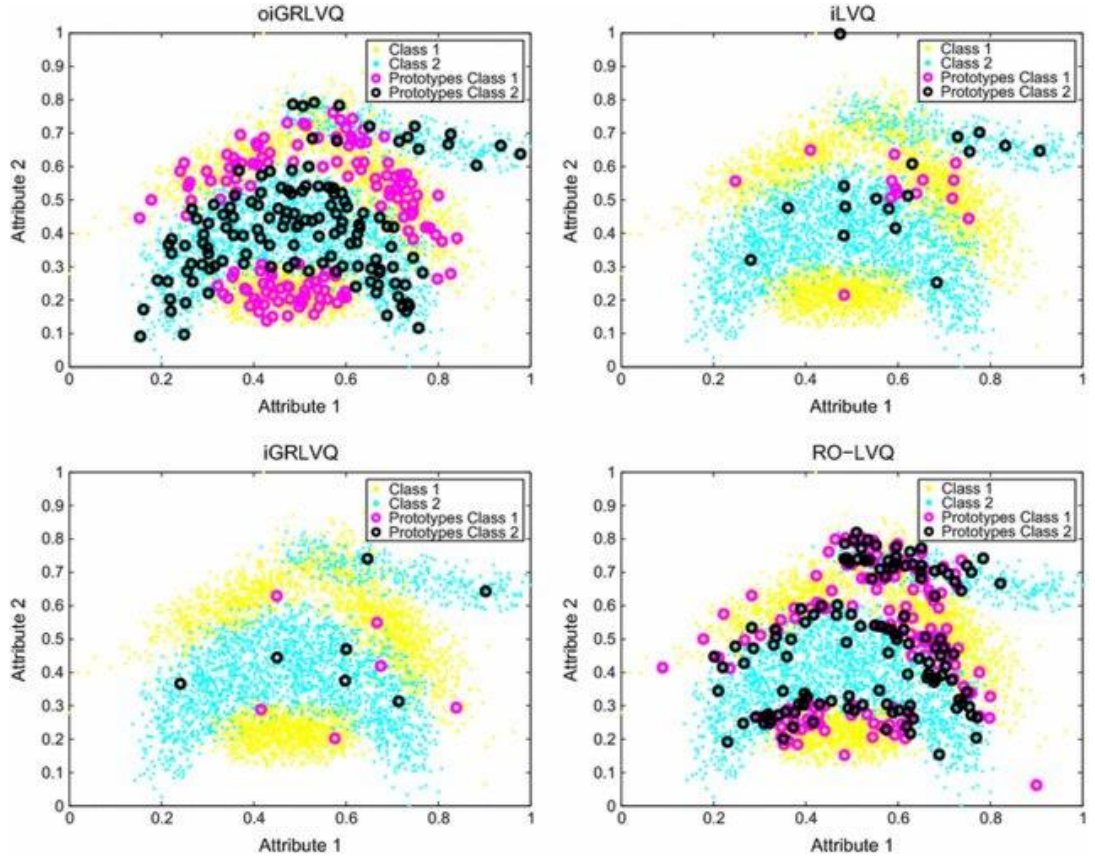
Edited set with 1105 instances



Reduced set with 5 prototypes

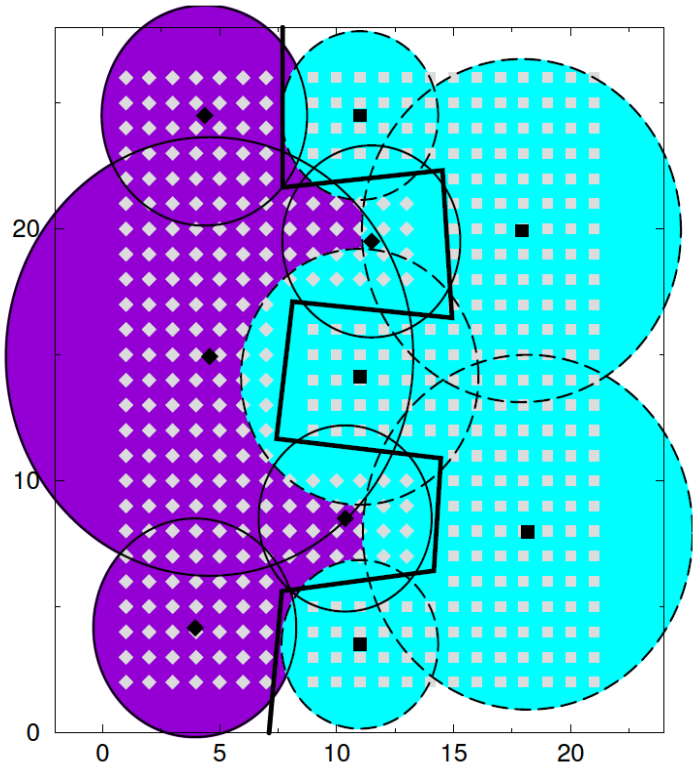
Prototype generation (ix)

Example of a positioning adjustment method over the Banana data set



Prototype generation (x)

Example of a space partitioning method



A synthetic problem with 2 non-overlapping interlaced classes:

The algorithm generated 10 prototypes, which is the minimum number to fully define the decision boundaries of the 1-NN classifier