Applying Machine Learning Methods to Wastewater Treatment Plant Data

Joaquim Comas[1], **Saso Dzeroski**[2] and **Miquel Sànchez-Marrè**[3]

Abstract. Artificial intelligence techniques, including machine learning techniques, have shown promising results as decision support tools, because of their capabilities of heuristic reasoning and working with uncertain and qualitative information. Wastewater treatment plants are complex environmental processes that are difficult to manage and control. In this paper, four machine learning methods are tested and compared on a wastewater treatment data set. The methods used are: induction of decision trees, rule induction, instance-based learning and case-based learning. All the methods are compared in terms of predictive accuracy, the number of attributes and examples used, and the meaningful-ness of induced patterns to domain experts.

1 INTRODUCTION

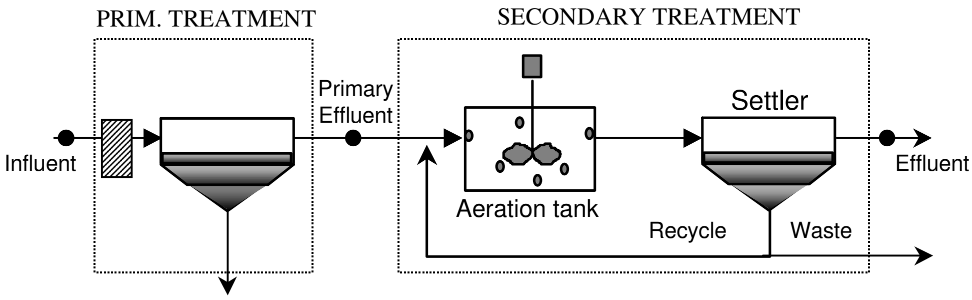
Industrial and environmental processes have some features that make the classical approach to their optimal management and control difficult. These 3-dimensional space processes behave dynamically over time and involve interactions between physico-chemical and biological processes. They are also stochastic [13].

Wastewater treatment is a complex environmental process where the influent changes (both in quantity and quality), the present microorganisms vary over time both in overall quantity and in composition (relative quantities and number of species), and the detailed knowledge of the processes involved is limited. Moreover, the data coming from Waste-Water Treatment Plants (WWTP) is subject to the typical problems of environmental data: uncertainty or imprecise information (few on-line analyzers are available and they often are unreliable), heterogeneity (different types of data with prominence of qualitative information and different frequency of analysis) and high incidence of missing information, which biases wastewater-treatment analysis.

Although progress in control engineering, computer technology, and process sensors has enabled the improvement of WWTP control, approaches other than a straightforward application of classical control theory are needed to keep complex processes under control, especially when the processes are in abnormal (far from ideal) situations.

In this sense, different artificial intelligence techniques, including machine learning techniques, have shown promising results as decision support tools, because of their capabilities of heuristic reasoning and working with uncertain and qualitative information.

Figure 1. Water flow sheet of a typical WWTP.



In this paper, we compare the performance of several machine learning methods on an environmental data set originating from a WWTP. Section 2 describes the WWTP domain and the characteristics of the data set. In Section 3, the four different machine learning techniques used are described: C4.5 [18], an inductive decision tree algorithm, CN2 (see [6] and [7]), a rule induction algorithm, *K*-NN (see [11] and [29]), an instance-based classification learning algorithm, and Opencase (see [21] and [23]), a case-based classification tool.

The results of their application to a WWTP historical database with the aim of extracting useful patterns or relationships between different variables are detailed in Section 4. The four methods are

[1] Chemical and Environmental Engineering Laboratory (LEQUIA), University of Girona, Campus de Montilivi, E-17071 Girona, Catalonia, EU, email: quim@lequia.udg.es
[2] Dept. of Intelligent Systems, Jozef Stefan Institute, Jamova 39, SI-1000 Ljubljana, Slovenia, email: Saso.Dzeroski@ijs.si
[3] Artificial Intelligence Section, Dept. of Software, Technical University of Catalonia, Campus Nord-Edifici C5, E-08034 Barcelona, Catalonia, EU, email: miquel@lsi.upc.es

evaluated in terms of predictive accuracy on unseen cases, measured by a tenfold cross-validation. They are also compared in terms of the number of attributes and examples used, and the meaningful-ness of induced patterns to domain experts. Finally, in Section 5, some conclusions are outlined.

Table 1. Sample points and selected variables

Sample point	On-line data	Analytical data	Qualitative data
Influent (AB)	Flow rate (Q-AB), pH	COD, BOD, TSS, TKN, NH4, COND, TERB, Cl	-
Primary effluent (SP1, SP3)	Flow rate (Q-SP1 and Q-SP3)	COD, BOD, TSS	-
Aeration tank (AS)	-	MLSS, MLVSS, SVI, F:M, SRT	Qualitative observations and microscopic analysis
Effluent (AT)	Flow rate (Q-AT), pH	COD, BOD, TSS, TKN, NH4, COND, TERB, Cl	Qualitative observations

Table 2. On-line, analytical and qualitative data.

On-line: Flow-rate, pH
Analytical **:** COD (Organic matter measured as Chemical Oxygen Demand), BOD_5 (Biodegradable organic matter measured as Biochemical Oxygen Demand), TSS (Total suspended solids), TKN (Total kjeldhal nitrogen = ammonia + organic nitrogen), N-NH4 (Ammonia), NO_2^- (Nitrite), NO_3^- (Nitrate), COND (Conductivity), TERB (Turbidity), Cl (Chlorine), MLSS (Mixed liquor suspended solids), MLVSS (Mixed liquor volatile suspended solids - measure of the biomass), SVI (Sludge Volume Index, calculated as Volume of the biomass settled in 30 minutes divided by MLSS), F:M (Food to microorganism ratio = (Q-OP1 * BOD-OP1)/(MLSS * $V_{reactor}$)), SRT (Sludge Residence Time = $V_{reactor}$ * MLSS / Biomass purged)
Qualitative Observations and microscopic analysis: ESC-B (Presence of foam at the aerated tank), FLOC (Presence of activated sludge flocs at the secondary settler), EST-D (Appearance of the secondary settler surface), ASP-AT (Quality of the final treated water (effluent)), PROB (V30-test observations), ASP-FLOC (Microscopic appearance of the activated sludge floc), PROTOS (Qualitative measure of the protozoa), *Zooglea, Nocardia, 021N / Thiothrix, Type OO41, Microthrix parvicella,* Dominant filamentous bacteria, Number of different filamentous bacteria, *Aspidisca, Euplotes, Vorticella, Epistylis, Opercularia,* Carnivorous ciliates, Dominant protozoa-group, Biodiversity of the microfauna (number of different taxa), Flagellates > 20 μm, Flagellates < 20 μm, *Amoebae,* Testate

2 WWTP DOMAIN AND DATA

The goal of Waste-Water Treatment Plants is to provide a regulated outflow of water with a limited quantity of contaminants. This in order to maintain natural water systems at as high a quality level as possible, thus ensuring a good quality of life. Wastewater coming from domestic sewers usually contains high concentrations of organic matter (measured as BOD or COD), total suspended solids (TSS) and nutrients (measured as TKN or NH4). The basis of the treatment is physical settling of suspended solids (primary treatment) and a biological (secondary) treatment that oxidizes the biodegradable organic matter into stabilized, low-energy compounds, maintaining a mixture of microorganisms and supplying oxygen by aerators in the aeration tank.

The most widely used biological treatment is the activated sludge process. It is always composed of two units: the *biological reactor* and the *secondary settler* (that separates the final effluent from the microorganisms). The efficient functioning of the biological process of a WWTP relies on the performance of biological degradation and on the good separation of microorganisms from treated water. Both processes are strongly affected by the quantity and biodiversity (number of species) of the microorganisms [15].

2.1 WWTP data set

The database analyzed originates from a WWTP located in Costa Brava, Catalonia. This plant provides primary and secondary treatment using the activated sludge process to remove organic matter and suspended solids from municipal wastewater of about 20,000 inhabitants-equivalents (see Fig. 1).

An exhaustive assessment of the wastewater quality is carried out in the plant, yielding both quantitative and qualitative data. Quantitative data is provided by on-line sensors (flow rates and pH) and by analysis of samples collected daily at the plant (measuring organic matter, nutrients, suspended solids, turbidity, conductivity, and biomass concentration-MLSS and MLVSS). Some global parameters like SRT (Sludge Retention Time), SVI (Sludge Volume Index) or F/M (Food to Microorganism ratio) are calculated when needed. Qualitative data -measured at a lower frequency- includes microscopic determinations of the biomass (floc characterization, micro fauna identification and counting, and filamentous bacteria identification and counting) and other qualitative observations of the process (like presence and color of foam and sludge, or appearance of the settler supernatant and effluent). The on-line, analytical and qualitative data studied are summarized in Tables 1 and 2.

A data set of 243 days and 63 quantitative and qualitative features, coming from two years of WWTP operation, are examined. There were many days missing a lot of attribute values, so we decided not to include them in the data set. Still, there are many missing values in the data set matrix. The state of the plant (the class attribute) was previously identified by means of a semi-automatic classification process using Linneo+ tool (see [3] and [24]) and expert criteria.

2.2 WWTP data classification

Linneo+ [3], which is a semi-automated knowledge acquisition tool concerned with building classifications for ill-structured domains, was the software used for clustering the data. At the beginning of the classification process, given a set of measured data, no knowledge on process states is available.

Linneo+ is an unsupervised learning (clustering) method that determines useful subsets or classes of the data. In the classification step, Linneo+ works by defining a space of *n* dimensions, where *n* is the number of variables included in the database (in this case, 63). Within this space, each class is characterized by a center and a radius. The radius is a parameter that should be defined by the user, and the center is the n-dimensional prototype object with average attribute values from all the objects within a class. At the beginning of the classification process, the classes are still undefined, thus the first object is taken and placed within the space to form the first class. The center of this class is calculated according to the value of the variables that describe the object. Then, a second object is placed within the space. If it is close enough to the first object, they belong to the same class, whose center is re-calculated as the mean value of both objects. Otherwise, a new class is formed. To determine how close two objects are within this n-dimension space, a conventional concept of distance is used (generalized Hamming distance). A given value for the distance is established as the limit (or radius) for an object to belong to a class.

Table 3. Classes obtained with Linneo$^+$ classification

Situation	Class #	Nº of days
Normal WWTP-operation in winter days	1	81
Normal WWTP-operation in summer days	2	55
Rainy days	3	3
Storm days	4	3
Underloading	5	12
Overloading	6	1
Nitrification	7	2
Deflocculation	8	5
Bulking sludge due to Thiotrix (affecting the effluent)	9	3
Foaming sludge due to *Microthrix* with *normal* microfauna biodiversity	10	17
Summer days with optimal WWTP-operation	11	24
Clhorine shock	12	1
Denitrification in the secondary settler (*rising*)	13	7
Transition to a bulking-sludge episode due to Thiotrix	14	2
Weak episode of Foaming sludge due to *Nocardia*	15	4
Severe episode of foaming sludge due to *Nocardia*	16	5
Foaming sludge due to *Nocardia* and defflocculation	17	8
Foaming sludge due to *Microthrix* with *very low* microfauna *biodiversity*	18	1
Foaming sludge due to *Microthrix* and viscous bulking due to *Zooglea*	19	6
Winter-summer Plant configuration change	20	3

The classification process is an iterative process: different classifications are made (with different radius) to obtain a suitable number and size of classes with physical or biological sense, according to the expert criteria. After the classification process, these clusters must be interpreted and labeled by the experts as physical states that are contained in the database. In our domain, the experts map the classes to a group of days characterizing a typical operation state of the plant (a situation occurring in the plant). Three cases are possible: a class is equivalent to a state, a class is equivalent to a sub-state and a set of these classes is equivalent to a state and, a class is not equivalent to any known state with physical sense.

After an iterative classification process supervised by the expert, the 243 days were classified into 20 situations occurring in the plant [8]. These 20 classes (see Table 3) correspond to the clusters obtained with the same Linneo$^+$ classification using a radius equal to 10 except for two classes undetected with this radius. We obtained 18 clusters (but then the expert labelled only 11 main states and 10 sub-states). Other classifications with different radius discovered two new clusters corresponding to states of the plant.

3 MACHINE LEARNING METHODS

In this section the machine learning methods used will be explained.

3.1 Learning decision trees with C4.5

Classification trees [4], often called decision trees [19] predict the value of a discrete dependent variable with a finite set of values (called class) from the values of a set of independent variables (called attributes), which may be either continuous or discrete. Data describing a real system, represented in the form of a table, can be used to learn or automatically construct a decision tree. In the table, each row (example) has the form: (x_1, x_2, ..., x_N, y), where x_i are values of the N attributes (e.g., flow rate and pH of influent, presence of foam at the aeration tank) and y is the value of the class (e.g. the Linneo+ classification). The induced (learned) decision tree has a test in each inner node that tests the value of a certain attribute, and in each leaf a value for the class. Given a new example for which the value of the class should be predicted, the tree is interpreted from the root. In each inner node the prescribed test is performed and according to the result of the test the corresponding left or right subtree is selected. When the selected node is a leaf then the value of the class for the the new example is predicted according to the class value in the leaf.

The common way to induce decision trees is the so-called Top-Down Induction of Decision Trees (TDIDT, [19]). Tree construction proceeds recursively starting with the entire set of training examples (entire table). At each step, the most informative attribute is selected as the root of the (sub)tree and the current training set is split into subsets according to the values of the selected attribute. For discrete attributes, a branch of the tree is typically created for each possible value of the attribute. For continuous attributes, a threshold is selected and two branches are created based on that threshold. For the subsets of training examples in each branch, the tree construction algorithm is called recursively. Tree construction stops when all examples in a node are of the same class (or if some other stopping criterion is satisfied). Such nodes are called leaves and are labeled with the corresponding values of the class.

An important mechanism used to prevent trees from over-fitting data is tree pruning. Pruning can be employed during tree construction (pre-pruning) or after the tree has been constructed (post-pruning). Typically, a minimum number of examples in branches can be prescribed for pre-pruning and confidence level in accuracy estimates for leaves for post-pruning.

A number of systems exist for inducing classification trees from examples, e.g., CART [4], ASSISTANT [5], and C4.5 [18]. Of these, C4.5 is one of the most well known and widely-used decision tree induction systems. It was thus used in our experiments. The C4.5 parameters were left at their default values, performing both pre- and post-pruning.

3.2 Rule induction with CN2

Given a set of classified examples, a rule induction system constructs a set of if-then rules. An if-then rule has the form:

IF condition THEN conclusion.

The condition contains one or more attribute tests of the form $A_i = v_i$ for discrete attributes, and $A_i < v_i$ or $A_i > v_i$ for continuous attributes. The conclusion part has the form $C = c_i$, assigning a particular value c_i to the class C. We say that an example is covered by a rule if the attribute values of the example obey the conditions in the IF part of the rule.

In our experiments, we used the rule induction system CN2 (see [6] and [7]). CN2 uses the covering approach to construct a set of rules for each possible class c_i in turn: when rules for class c_i are being constructed, examples of this class are positive, all other examples are negative. The covering approach works as follows: CN2 constructs a rule that correctly classifies some examples, removes the positive examples covered by the rule from the training set and repeats the process until no more examples remain. To construct a single rule that classifies examples into class c_i, CN2 starts with a rule with an empty antecedent (if part) and the selected class c_i as a consequent (then part). The antecedent of this rule is satisfied by all examples in the training set, and not only those of the selected class. CN2 then progressively refines the antecedent by adding conditions to it, until only examples of the class c_i satisfy the antecedent. To allow for handling imperfect data, CN2 may construct a set of rules which is imprecise, i.e., does not classify all examples in the training set correctly.

Consider a partially built rule. The conclusion part is already fixed and there are some (possibly none) conditions in the if part. The examples covered by this rule form the current training set. For discrete attributes, all conditions of the form $A_i = v_i$, where v_i is a possible value for A_i, are considered for inclusion in the condition part. For continuous attributes, all conditions of the form $A_i < (v_{ik} + v_{i(k+1)}) / 2$ and $A_i > (v_{ik} + v_{i(k+1)}) / 2$ are considered, where v_{ik} and $v_{i(k+1)}$ are two consecutive values of attribute A_i that actually appear in the current training set. For example, if the values 4.0, 1.0, and 2.0 for attribute A appear in the current training set, the conditions A < 1.5, A > 1.5, A < 3.0, and A > 3.0 will be considered.

Note that both the structure (set of attributes to be included) and the parameters (values of the attributes for discrete ones and boundaries for the continuous ones) of the rule are determined by CN2. Which condition will be included in the partially built rule depends on the number of examples of each class covered by the refined rule and the heuristic estimate of the quality of the rule. The heuristic estimates are mainly designed to estimate the performance of the rule on unseen examples in terms of classification accuracy. This is in accord with the task of achieving high classification accuracy on unseen cases. Suppose a rule covers p positive and n negative examples. Its accuracy can be estimated by the relative frequency of positive examples covered, computed as $p / (p + n)$. This heuristic was used in early rule induction algorithms. It prefers rules which cover examples of only one class. The problem with this metric is that it tends to select very specific rules supported by only a few examples. In the extreme case, a maximally specific rule will cover (be supported by) one example and hence have an unbeatable score using the metrics of apparent accuracy (scores 100% accuracy). Apparent accuracy on the training data, however, does not adequately reflect true predictive accuracy, i.e., accuracy on new testing data. It has been shown [14] that rules supported by few examples have very high error rates on new testing data.

The problem lies in the estimation of the probabilities involved, i.e., the probability that a new example is correctly classified by a given rule. If we use relative frequency, the estimate is only good if the rule covers many examples. In practice, however, not enough examples are available to estimate these probabilities reliably at each step. Therefore, probability estimates that are more reliable when few examples are given should be used.

A more recent version of CN2 [6] uses the Laplace estimate to estimate the accuracy of rules. This estimate is more reliable than relative frequency. If a rule covers p positive and n negative examples its accuracy is estimated as $(p + 1) / (p + n + N)$, where N is the number of possible classes.

CN2 can induce a set of if-then rules which is either ordered or unordered. In the first case, the rules are considered precisely in the order specified: given an example to classify, the class predicted by the first rule that covers the example is returned. In the second case, all rules are checked and all the rules that cover the example are taken into account. Conflicting decisions are resolved by taking into account the number of examples of each class (from the training set) covered by each rule. Suppose we have a two class problem and two rules with coverage [10,2] and [4,40] apply, i.e., the first rule covers 10 examples of class c_1 and 2 examples of class c_2, while the second covers 4 examples of class c_1 and 40 examples of class c_2. The "summed" coverage would be [14,42] and the example is assigned class c_2.

CN2 handles examples that have missing values for some attributes in a relatively straightforward fashion. If an example has an unknown value of attribute A, it is not covered by rules that contain conditions that involve attribute A. Note that this example may be covered by rules that do not refer to attribute A in their condition part.

In our experiments, CN2 was used to induce sets of unordered rules. The rules were required to be highly significant (at the 99% level) and thus reliable. Except for the significance threshold and the search heuristic settings, described below, the parameter settings of CN2 were the default ones [6].

3.3 Instance-based learning

Instance-based learning (IBL) algorithms [2] use specific instances to perform classification tasks, rather than using generalizations such as induced if-then rules. IBL algorithms are also called lazy learning algorithms, as they simply save some or all of the training examples and postpone all effort towards inductive generalization until classification time. They assume that similar instances have similar classifications: novel instances are classified according to the classifications of their most similar neighbors.

IBL algorithms are derived from the nearest neighbor pattern classifier (see [9] and [12]). The nearest neighbor (NN) algorithm is one of the best-known classification algorithms and an enormous body of research exists on the subject [10]. In essence, the NN algorithm treats attributes as dimensions of an Euclidean space and examples as points in this space. In the training phase, the classified examples are stored without any processing. When classifying a new example, the Euclidean distance between that example and all training examples is calculated and the class of the closest training example is assigned to the new example.

The more general k-NN method takes the k nearest training examples and determines the class of the new example by majority vote. In improved versions of k-NN, the votes of each of the k nearest neighbors are weighted by the respective proximity to the new example [11]. In our experiments, k was set to 4.

Finally, the contribution of each attribute to the distance may be weighted, in order to avoid problems caused by irrelevant features [30]. The feature weights are determined on the training set by using one of a number of alternative feature weighting methods. In our experiments, we used the k-NN algorithm as implemented by Wettschereck [29], which includes the improvements described above. A more detailed description of how distance computation, classification, and feature weighting is performed, is given below.

Given two examples $x = (x_1, ..., x_n)$ and $y = (y_1, ..., y_n)$, their distance is calculated as

$$distance(x, y) = \sqrt{\sum_{i=1}^{n} w_i \times difference(x_i, y_i)^2}$$

where w_i is a non-negative weight value assigned to feature A_i and the difference between attribute values is defined as follows

$$difference(x_i, y_i) = \begin{cases} |x_i - y_i| & \text{if feature } A_i \text{ is continuous} \\ 0 & \text{if feature } A_i \text{ is discrete and } x_i = y_i \\ 1 & \text{otherwise} \end{cases}$$

When classifying a new instance z, k-NN selects the set K of k-nearest neighbors according to the distance defined above. The vote of each of the k nearest neighbors is weighted by its proximity (inverse distance) to the new example. The probability $P(z,c_j,K)$ that instance z belongs to class c_j is estimated as

$$P(z, c_j, K) = \frac{\sum_{x \in K} x_{cj} / distance(z, x)}{\sum_{x \in K} 1 / distance(z, x)}$$

where x is one of the k nearest neighbors of z and x_{cj} is 1 if x belongs to class c_j. The class c_j with largest value of $P(z,c_j,K)$ is assigned to the unseen example z.

Before training (respectively before classification), the continuous features are normalized by subtracting the mean and dividing by the standard deviation so as to ensure that the values output by the difference function are in the range [0,1]. All features have then equal maximum and minimum potential effect on distance computations. However, this bias handicaps k-NN as it allows redundant, irrelevant, interacting or noisy features to have as much effect on distance computation as other features, thus causing k-NN to perform poorly. This observation has motivated the creation of many methods for computing feature weights.

The purpose of a feature weighting mechanism is to give low weight to features that provide no information for classification (e.g., very noisy or irrelevant features), and to give high weight to features that provide reliable information. The mutual information [26] $I(C,A)$ between the class C and attribute A is thus a natural quantity with which the feature A is weighted in the k-NN implementation of Wettschereck [29] that we employed in our experiments.

The mutual information [26] between two variables is defined as the reduction in uncertainty concerning the value of one variable that is obtained when the value of the other variable is known. If an attribute provides no information about the class, the mutual information will be zero. The mutual information between the random variables X and Y is defined as $I(X,Y) = H(X) - H(X|Y)$, where $H(X)$ is the entropy of the random variable X with probability mass function $P(x)$, defined as $H(X) = - \Sigma \log_2 P(x)$. For discrete X and Y, it can be also calculated as

$$I(X, Y) = \sum_{x,y} P(x, y) \times \log_2 \frac{P(x, y)}{P(x)P(y)}$$

For continuous variables, probability densities have to be used instead of probability masses and integrals instead of sums. The probabilities involved are in our case estimated from the training examples.

Unknown values in the examples are handled through a modification of the distance function between examples. Only features that have known values are used in calculating the distance, and the number of features for which both examples have known values is taken into account. The modified distance function is thus

$$d(x, y) = \frac{\sqrt{\sum_{i=1}^{n} w_i \times diff(x_i, y_i)^2}}{\sqrt{\text{number of features } i \text{ for which both } x_i \text{ and } y_i \text{ are known}}}$$

where $diff(x_i,y_i) = 0$ if either x_i or y_i are unknown and $diff(x_i,y_i) = difference(x_i,y_i)$ otherwise.

3.4 Case-based learning

Case-based learning [17] can be considered as a kind of instance-based learning. It can be used for performing a case-based classification, provided that the cases or examples in the training set have a class label or attribute. In fact, *case-based classification is an specialization of the case retrieval step* in the general case-based reasoning cycle of a case-based system (see [1] and [16]). The assumption is that similar cases have similar classifications: new cases are classified according to the classifications of their most similar cases retrieved in the case library.

The case retrieval task -within the global Case-Based Reasoning (CBR) systems control flow scheme [1]- can be defined as follows: Given a description of a current problem or situation, a set of goals to be achieved, and a case library organization, find one or more useful similar cases to solve the current problem. In CBR systems, the retrieval phase is one of the main tasks. If it is not carried out with accuracy and reliability, it can lead the system to wrong results.

The retrieval task is strongly dependent on the case library organization. Main memory organizations in CBR systems can be classified in two general approaches: flat memories and hierarchical memories, such as shared feature networks, prioritized discrimination networks/trees or redundant discrimination networks/trees. Flat memories have the advantage of always retrieving the set of cases best matching the input case. Moreover, adding new cases to memory is cheap. But they have a major disadvantage: the retrieval time is very expensive since every case in the memory is matched against the current case, using a Nearest Neighbor (NN) algorithm [27]. On the opposite side are the hierarchical memories. In such kind of memories, matching process and retrieval time are more efficient, due to the fact that only few cases are considered for similarity assessment purposes, after a prior discriminating search in the hierarchical structure. However, they also have some disadvantages. Keeping the hierarchical structure in an optimal condition requires an overhead on the case library organization, and the retrieval process could miss some optimal cases searching a wrong area of the hierarchical memory. This problem becomes especially hard in prioritized discrimination networks/trees.

The main difference between instance-based classification and case-based classification is: the memory structure where the training examples or cases are stored and the domain expert knowledge. In instance-based classification usually a flat memory is used with a NN or *K*-NN algorithm as basis for the classification process whereas for case-based classification other kind of memories are used as explained before with more specialized retrieval algorithms. Also, instance-based classification often does not take into account as much domain expert information as case-based classification does.

Opencase (see [21] and [23]) is a prototype of case-based reasoning shell for developing case-based systems. In our approach to represent the domain we have chosen a domain independent implementation by means of a table of attributes. This implementation also eases the use of raw data coming from sensors and allows the connection of the CBR system with other modules. From all the possible signals coming from the process and other values that can be obtained off-line we only keep those attributes that are considered relevant by the experts. Those attributes represent the use of intensive knowledge about the domain coming from experts and could be considered as the explicit incorporation of their preferences in the decision making about the set of alternatives that appear in the application domain. Also, the system could work without this bias and use all the attributes.
The case structure is a record-like structure or template with some slots such as in the following example:

```
( :identifier  CASE-72
  :situation-description
    ( (Water-inflow 22,564 m3/day)
      (Inflow-Biological-Oxygen-Demand 192.9 mg/L)
        . . . )
  :diagnostics  STORM-SITUATION
  :actuation-plan
    ( (1 Close Purge-flow)
      (2 Maintain-DO-predictive-control)
        . . . )
  :case-derivation  CASE-11
  :solution-result  SUCCESS
  :utility-measure  0.63
  :distance-to-case  0.1305 )
```

In order to get an efficient CBR system, the case library is implemented as a prioritized discrimination tree, thus only assessing the similarity of some retrieved cases, although a flat memory organization can be used instead. A prioritized discrimination tree is a tree in which each node matches with a prioritized attribute. This implementation facilitates the case extraction when the case library grows as the prioritized attributes represent the most relevant aspects that constitute the basis for the search in the case library. The goal of the retrieval task of our CBR system is to search for similar precedents from the case library.

The case library has at each node as many branches as different discretized values are defined for the attribute. It supports missing information by means of some frequency values associated to each branch in the tree. The indexing of the case library in the retrieve phase is made taking into account the predictive discriminant checklist of attributes, and following the path into the case library that best matches the input values of the new case. The similarity assessment of retrieved cases is made through a similarity measure based on a new defined distance function: *L'Eixample* distance [22]. Nevertheless, other distance functions can be used.

L'Eixample distance is sensitive to weights in the sense that, for the most important attributes, those with weights > α, the distance is computed based on their qualitative values, i.e. maintaining or amplifying the differences between cases, and for those less relevant ones those with weights ≤ α the distance is computed based on their quantitative values, i.e. reducing the differences between cases. *L'Eixample* distance used to rank the best cases is:

$$d(C_i, C_j) = \frac{\sum_{k=1}^{n} e^{w_k} \times d(A_{ki}, A_{kj})}{\sum_{k=1}^{n} e^{w_k}}$$

where

$$d(A_{ki}, A_{kj}) = \begin{cases} \dfrac{|quantval(A_{ki}) - quantval(A_{kj})|}{upperval(A_k) - lowerval(A_k)} & \text{if } A_k \text{ is an ordered attribute and } w_k \le \alpha \\ \dfrac{|qualval(A_{ki}) - qualval(A_{kj})|}{\#\mathrm{mod}(A_k) - 1} & \text{if } A_k \text{ is an ordered attribute and } w_k > \alpha \\ 1 - \delta_{qualval(A_{ki}), qualval(A_{kj})} & \text{if } A_k \text{ is a non-ordered attribute} \end{cases}$$

and,

C_i is the case i; C_j is the case j; W_k is the weight of attribute k; A_{ki} is the value of the attribute k in the case i; A_{kj} is the value of the attribute k in the case j; qtv(A_{ki}) is the quantitative value of A_{ki}; qtv(A_{kj}) is the quantitative value of A_{kj}; A_k is the attribute k; upperval(A_k) is the upper quantitative value of A_k; lowerval(A_k) is the lower quantitative value of A_k; α is a cut point on the weight of the attributes; qlv(A_{ki}) is the qualitative value of A_{ki}; qlv(A_{kj}) is the qualitative value of A_{kj}; #mod(A_k) is the number of modalities (categories) of A_k; δqlv(A_{ki}),qlv(A_{kj}) is the δ of Kronecker.

4 RESULTS

To estimate the performance of the different machine learning methods on unseen cases, we performed a tenfold stratified cross-validation. The whole set of 243 examples was split into 10 sets of 24/25 examples each: these were in turn used as test sets, while the remaining 219/218 examples were used for training. The comparison of the different methods is summarized in Table 5. The remainder of this section gives details on the obtained results: besides predictive accuracy, the number of attributes and examples used, and meaningful interpretation of induced patterns by experts are considered. The knowledge patterns discovered by C4.5, CN2 and Opencase can be codified as decision rules, or in a similar representation formalism, and can be added to the knowledge base of a knowledge-based system to improve the WWTP supervision [20]. The C4.5, Opencase and CN2 rules derived from all 243 examples were shown to the domain expert for comments.

4.1 Learning decision trees with C4.5

The average predicitive accuracy over the 10 folds was 63.51%. All examples in the training tests were used, and some of the attributes (not all) were selected by the algorithm.

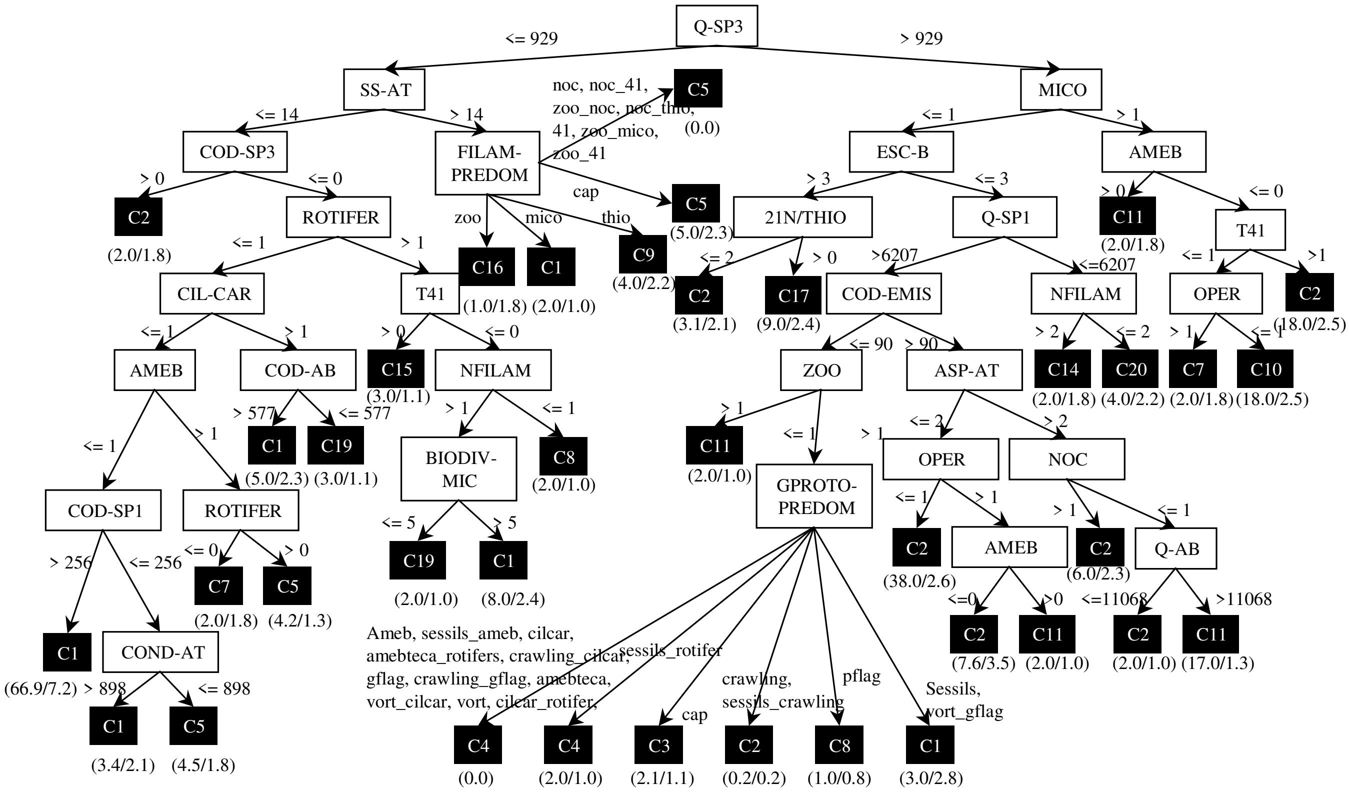


Figure 2. Tree generated by C4.5 given all 63 attributes.

Also a new decision tree was induced with all the examples in the data set (243 examples). This decision tree, depicted in Figure 2, used the following 24 attributes: 21N/Thio, T41, Ameb, Asp-AT, Biodiv-Mic, Cil-car, Cond-AT, COD-AB, COD-Emis, COD-SP1, COD-SP3, ESC-B, Filam-domi, Gproto-domi, Mico, Nfilam, Noc, Oper, Q-AB, Q-SP1, Q-SP3, Rotifers, SST-AT, Zoo.

It is interesting to notice the predominance of the qualitative variables over the quantitative variables. For example, the tree induced on the whole data of 243 days uses 14 qualitative variables and only 10 quantitative attributes (see Figure 2).
The trees use qualitative variables concerning not only the number and diversity of microorganisms, but also some subjective features like presence of foam in the aeration tank (ESC-B) or appearance of the effluent (ASP-AT).

The numbers in brackets for the leaves of the C4.5 tree specify the number of examples in the leaves For example, c1 (66.9/7.2) means that there are 66.9 examples in the leaf, of which 7.2 are not of class c1 (the remaining 59.7 examples are). Fractional examples are due to unkown values. Zeros appear when dealing with discrete multi-valued attributes: these are the so-called *null* leaves. This means that there were no examples in the training set that satisfy *all* the conditions leading to that leaf (but there are some that satisfy all but the last). The branches with higher accuracy correspond to those predicting large classes (class#1 with 89.2% and class#2 with 92.7%) and medium classes : class#11 (17/1.3, 92.3%), class#10 (18/2.5, 86.1%) and class#17 (9/2.4, 73.3). These decision trees can easily be codified as decision rules.

We also used only the 19 attributes used by OpenCase, which are the most relevant variables according to expert opinion. In this case, classification accuracy improves slightly (65.11% cross-validation average), but understandability improves greatly. The new C4.5 tree induced from all 243 examples is easier to interpret from the expert point of view than the first one (Figure 2) because it is smaller and it doesn't use attributes that the expert would not use to distinguish among the classes (and that were used by the first one, i.e., Rotifers, CIL-CAR, AMEB, T41).

4.2 Rule induction with CN2

The average predicitive accuracy over the 10 folds was 63.98%. All examples in the training tests were used, and some of the attributes (not all) were selected by the algorithm.

Also a new set of rules was induced with all the examples in the data set (243 examples). Some of this rules are explained and commented below. This set of rules used 44 attributes.

As a general conclusion on the induced rules, we note that the qualitative variables have an important role when diagnosing the state of the plant. This agrees with the idea that qualitative attributes, including microscopic examinations of microfauna and bacteria, are useful indicators of the global performance of the WWTP processes.

When trying to predict large classes, rules obtained are larger than those covering medium-size classes but the accuracy is lower. The rules for class#1 and class#2 have most physical and chemical sense and are easiest to interpret by the expert. The plant manager esasily identifies in the antecedents some of the variables that he used in his heuristic checking process. The following rules are examples of rules induced on the whole dataset of 243 examples for predicting class#1 and class#2. The numbers in brackets for CN2 rules refer to the training set, which in this case was the entire dataset.

IF Q-AB < 9083.00
AND Q-SP3 < 2227.00
AND COND-AT > 847.00
AND CIL-CAR < 0.50
AND AMEB < 0.50
THEN WWTP-problems = c1 (47/81)

IF TERB-AB > 123.50
AND Q-SP1 > 6223.50
AND Q-SP3 > 3467.50
AND 33.00 < DQO-AT < 82.00
AND COND-AT < 1330.00
AND NOC > 0.50
AND P-FLAG < 3.50
THEN WWTP-problems = c2 (38/55)

The rules covering medium classes appear to be better in classification performance. For example, the next rule predicts class#10 corresponding to Foaming due to Microthrix.

IF Q-AB < 10924.50
AND Q-SP3 > 1030.00
AND NOC < 1.50
AND MICO > 1.50
THEN WWTP-problems = c10 (17/17).

In those classes represented only by one or two examples, the induced CN2 rules utilize variables which don't seem to be the most discriminant ones, according to the criteria of the expert (plant manager of the WWTP) but they have higher accuracy (on the training data). For example, the pH in the influent is usually not a very informative attribute except for extreme values over 8.5 or under 6.5 and this is not the case in the rule predicting class #9:

IF PH-AB < 7.40
AND FILAM-DOMI = thio
THEN WWTP-problems = c9 (3/3)

We also used only the 19 attributes used by OpenCase, which are the most relevant variables according to expert opinion. In this case, classification accuracy improves slightly (65.45% cross-validation average), but understandability improves greatly. The new CN2 rules induced from all 243 examples are easier to interpret from the expert point of view because they do not use attributes that the expert would not use. The rules covering the larger classes (1 and 2) have more physical and chemical sense and they have the same accuracy as the first set of rules. Attributes hardly used by the expert never appear in these rules. This holds even more for rules covering small classes (e.g. c12, c6, c3 and c4).

4.3 Instance-based learning

After the 10-fold cross validation in the data set, the average predicitive accuracy of the *K*-NN algorithm was 76.38%. All examples in the training tests were used, and all the attributes in the data set were used. Feature weighting was used, distinguishing the important attributes from the unimportant ones. When using only 19 attributes, accuracy drops significantly (by 5%).

4.4 Case-based learning

In our WWTP domain, according to the experts, the case was codified through the 19-most relevant variables measured routinely in the plant: ASP-AT, Q-SP3, DQO-AT, ZOO, DQO-AB, NH4-AT, SST-AT, IVF, COND-AB, Q-AB, BIODIV-MIC, P-FLAG, GPROTO-DOMI, ESC-B, NO3-AT, FILAM, DQO-SP1, CM and NFILAM. These variables were prioritized and discretized into several modalities, such as low (L), normal (N), and high (H), and finally, as a result of an iterative process, a weight and a discriminant order were assigned showing their relevance in the characterization of the situation. But only 15 of these attributes were used to build the hierarchical case libraries

Then, 3 kind of experiments were done: using a plain memory (similar to NN and *K*-NN algorithms), using a hierarchical case library but only learning the relevant cases according to a defined relevance criterion, and using a hierarchical case library learning all cases. After the 10-fold cross validation in the data set, the average predicitive accuracies for each kind of experiment are detailed in the table 4. The predictive accuracies of the classification were measured not only for the most similar case, but for the second more similar case and for the predominant (majority vote) of the retrieved cases.

Also a new hierarchical case library was induced with all the examples in the data set (243 examples). This hierarchical case library, depicted in figure 3, used also the same 19 attributes.

Finally, a fourth experiment was carried out in order to compare the Opencase (plain memory) using only the most relevant 19 attributes with respect to Opencase (plain memory) using 63 attributes to define each case.

Different relevant outcomes can be extracted from the Opencase results:

- Opencase always get better accuracies for the retrieved case in first position except for plain memory that has higher accuracy with the predominat case. The loss of accuracy for predominant case is due to the retrieving of examples of large classes when dealing with a case belonging to a small class.
- Both hierarchical libraries underwent only a little loss of accuracy (3.3 and 1.6%, respectively) with respect to a plain memory library while a big improvement in the time retrieving of the most similar cases is achieved.
- The performance of the selective learning algorithm of Opencase has turned out to be very good since the accuracy of the hierarchical library with only relevant cases (sustainable learning) is very similar to the hierarchical library with all the cases (complete learning).
- The loss of accuracy using 19 or 63 attributes is minimal (less than 3% and 2% in the first and predominant case retrieved, respectively) whereas there is a great gain in time retrieving.

Table 4. Case-based learning results

Nº Attrib.	Type of library	Case Retrieval Accuracy (%)		
		First	Second	Predominant
19	Plain memory	65.8	59.7	68.7
	Hierarchical (relevant cases)	62.5	44.9	52.3
	Hierarchical (all cases)	64.2	44.4	51.1
63	Plain memory	68.7	60.5	70.4

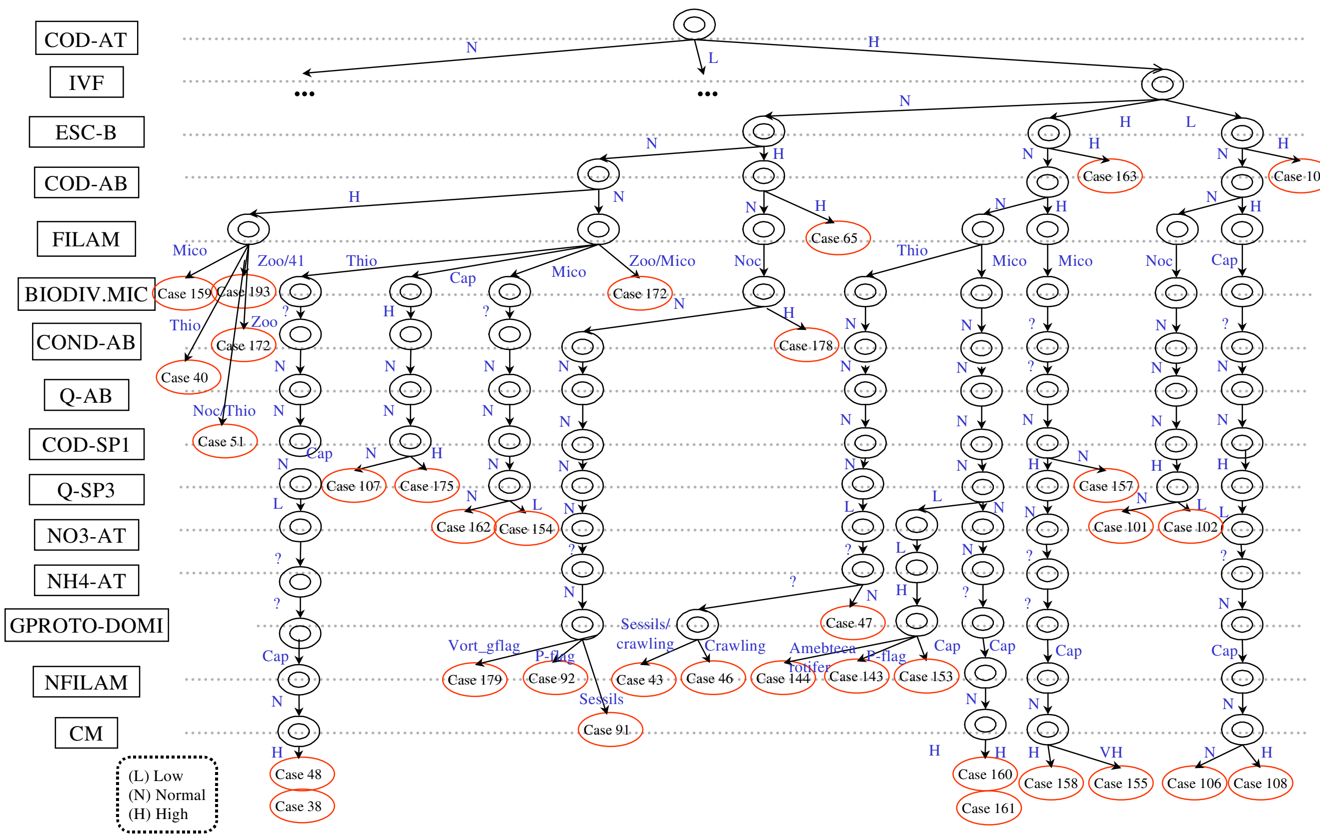


Figure 3. Case Library with all the cases.

5 CONCLUSIONS

We have applied four different machine learning algorithms to the task of classifying descriptions of the daily operation of a Waste-Water Treatment Plant into 20 different classes. The classes had previously been obtained by clustering. We compare the four approaches in terms of predictive accuracy and in terms of understandability of the induced rules/patterns. A summary of the comparison is given in Table 5. Before we discuss the results in some more detail, let us reiterate some of the characteristics of the data set used:

- It is possible that there are some classes obtained by the clustering process that are not different enough from the others. These classes are very difficult to identify by all the methods used.
- There are some variables with many missing values. This fact makes the inductive process and the classification of new unseen examples more difficult. This can explain that the accuracies are not very high for all methods.
- Another important feature is the fact that there are 9 classes with no more than 3 examples each in the whole data set (see Table 3), and 3 classes have only 1 example each. This makes it very hard for an inductive method that tries to generalize.

Table 5. Comparison of the four methods.

Method	Number of Attributes	Number of Examples	Prediction accuracy on test set (%)	Meaningful Interpretation	Prediction accuracy on whole data set (%)
C4.5 (63 atts)	24	243	63.51	Partially	89.7
CN2 (63 atts)	44	243	63.98	Partially	98.8
k-NN (63 atts)	63	243	76.38	No	100
C4.5 (19 atts)	11	243	65.11	Mostly	87.2
CN2 (19 atts)	19	243	65.45	Mostly	95.9
k-NN (19 atts)	19	243	71.22	No	100
Opencase (plain memory)	19	243	68.73	No	100
Opencase (hierarchical, relevant cases)	19	220	62.50	Yes	97.1
Opencase (hierarchical, all cases)	19	243	64.20	Yes	98.8
Opencase (plain memory)	63	243	70.40	No	100

Best accuracy was obtained with the *k*-NN method. This could be expected, since the clustering used to design the classes was performed with Linneo+, which itself is based on the notion of distance. The main problem of kNN methods is the expensive time requirements. In contrast, the retrieving process in a hierarchical case library induced by Opencase or in an induced decision tree by C4.5 is very fast since only a few branches of the hierarchical trees are explored by the algorithm search. The loss of accuracy of Opencase using only the 19 attributes selected by the expert with respect to NN methods can be estimated about 10% while the number of variables is reduced from 63 to 19 (more than 69%). In comparison to the C4.5-trees and CN2-rules induced by using all 63 attributes, Opencase with plain or hierarchical library (with complete learning) appears to be slightly more accurate.

For a more complete comparison, we also used kNN, C4.5 and CN2 on the 19 expert selected attributes. kNN performance drops significantly (as compared to 63 attributes), but is still better than OpenCase. The performance of C4.5 and CN2 in terms of accuracy improves slightly. More importantly, the understandability of the induced trees and rules improves greatly, making them much more acceptable for use in a knowledge-based system.

The expert role in the experiments conducted with the inductive learning tools C4.5 and CN2 was to select the most relevant variables (as well in Opencase) and to interpret and validate the rules and trees induced. The latter includes commenting on which tree branches or rules make sense and which do not from the expert point of view: namely, the way in which induction tools use and rank the importance of attributes may not be exactly the way that a human expert would do it. Interpretation also involves an examination of the reasons for accuracy and understandability changes as the attributes used change. Finally, interpretation also includes looking for new knowledge in the new rules and trees induced.

In fact, with the experiments we have done, the experts' assistance role in the inductive learning process has been enhanced. Most of the inductive learning algorithms and techniques do not take into account the experts' knowledge very much. And this is not a good situation for extracting useful knowledge patterns from data (induced rules, induced decision trees, etc.). Obviously, the predictive accuracy of the inductive methods is a very important parameter, as well as other quantitative measures of performance such as the size both of the number of examples used and the number of attributes used for the inductive generalization. If the knowledge extracted from data is very accurate but it does not make sense to the final end-users, probably, it will not be used. In conclusion, we can state that the experts, who are supposed to be the final end-users of real-world applications, should be taken into account in the inductive machine learning methods as much as possible.

ACKNOWLEDGEMENTS

This work has been supported by the Spanish CICyT projects BIO96-1229 and AMB97-889, and by the CIRIT SGR 177 (1997) of the Generalitat de Catalunya as a "Grup de Recerca Consolidat". Thanks to Ljupco Todorovski for his help with running the C4.5, CN2 and kNN experiments.

REFERENCES

[1] Aamodt, A and Plaza, E. Case-based reasoning: fundamental issues, methodological variations and system approaches. *AI Communications* 7(1):39-59, 1994.

[2] Aha, D., Kibler, D., and Albert, M. Instance-based learning algorithms. *Machine Learning* 6:37-66, 1991.

[3] J. Béjar. Knowledge Acquisition in ill-structured domains. Ph. D. Thesis. Dept. de Llenguatges i Sistemes Informàtics. Universitat Politècnica de Catalunya, 1995.

[4] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone C.J. *Classification and Regression Trees*. Belmont: Wadsworth, Belmont, 1984.

[5] Cestnik, B., Kononenko, I., and Bratko, I. ASSISTANT 86: A knowledge elicitation tool for sophisticated users. In Bratko, I. and Lavrac, N., editors, *Progress in Machine Learning*, pages 31--45. Sigma Press, Wilmslow, 1987.

[6] Clark, P. and Boswell, R. Rule induction with CN2: Some recent improvements. In Proc. Fifth European Working Session on Learning, pages 151--163. Springer, Berlin, 1991.

[7] Clark, P. and Niblett, T. The CN2 induction algorithm. *Machine Learning*, 3(4):261—283, 1989.

[8] J. Comas, L. Cecaroni and M. Sànchez-Marrè. Semi-automatic learning with quantitaive and qualitative features. 8^{th} Spanish Conference on Artificial Intelligence (CAEPIA'99), pp.17-25. Murcia, Spain, November 1999.

[9] Cover, T.M., and Hart, P.E. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13: 21-27, 1968.

[10] Dasarathy, B.V., editor. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamitos, CA, 1990.

[11] Dudani, S.A. The distance-weighted k-nearest neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, 6(4): 325-327, 1975.

[12] Fix, E., and Hodges, J.L. Discriminatory analysis. Nonparametric discrimination. Consistency properties. Technical Report 4, US Air Force School of Aviation Medicine. Randolph Field, TX, 1957.

[13] G. Guariso and H. Werthner, *Environmental Decision Support Systems*, Ellis Horwood-Willey, 1989.

[14] Holte, R., Acker, L., and Porter, B. Concept learning and the problem of small disjuncts. In Proc. Tenth International Joint Conference on Artificial Intelligence. Morgan Kaufmann, San Mateo, CA, 1989.

[15] D. Jenkins, M.G. Richard, and G.T. Daigger, *Manual on the causes and control of activated sludge bulking and foaming,* Lewis Publishers, Chelsea, 1993.

[16] J. Kolodner. *Case-Based Reasoning*. Morgan Kaufmann, 1993.

[17] J. Kolodner. *Case-Based Learning*. Kluwer Academic Publishers,1993.

[18] Quinlan, J.R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[19] Quinlan, J.R. Induction of decision trees. *Machine Learning* 1(1): 81—106, 1986.

[20] I. R-Roda, J. Comas, J. Colprim, J. Baeza, M. Sànchez-Marrè and U. Cortés. A Multi-paradigm Decision Support System to improve WWTP Operation. In AAAI'99 Workshop on Environmental Decision Support Systems and Artificial Intelligence (EDSSAI'99). Technical Report WS-99-07, AAAI Press, pp.68-73, Orlando, USA. July 1999.

[21] M. Sànchez-Marrè, U. Cortés, I. R.-Roda, M. Poch. Sustainable Case Learning for continuous domains. *Environmental Modelling & Software* 14:349-357, May 1999.

[22] M. Sànchez-Marrè, U. Cortés, I. R-Roda, and M. Poch. L'Eixample distance: a new similarity measure for case retrieval. *Procc. of 1st Catalan Conference on Artificial Intelligence (CCIA'98)*, ACIA bulletin 14-15 pp. 246-253.Tarragona, Catalonia, EU.

[23] M. Sànchez, U. Cortés, I. R.-Roda, M. Poch and J. Lafuente, "Learning and adaptation in WWTP through case-based reasoning", *Microcomputers in Civil Engineering*, (Special issue: Machine Learning), 12(4), 251-266.

[24] M. Sànchez-Marrè, U. Cortés, J. De Gràcia, J. Lafuente and M. Poch. Concept Formation in WWTP by means of Classification Techniques: a Compared Study. *Applied Intelligence* 7(2):147-166. April, 1997

[25] M. Sànchez, U. Cortés, J. Lafuente, I. R.-Roda and M. Poch, "DAI-DEPUR: a Distributed Architecture for Wastewater Treatment Plants Supervision", *Artificial Intelligence in Engineering* 10(3), 275-285, (1996).

[26] Shannon, C.E. A mathematical theory of communication. *Bell. Syst. Techn. J.*, 27: 379—423, 1948.

[27] Watson, I. An introduction to Case-based reasoning. *Progress in Case-Based Reasoning*. LNAI #1020. Pp 3-16, 1996.

[28] Weiss, S.M., and Kulikowski, C.A. *Computer Systems that Learn*. Morgan Kaufmann, San Mateo, CA, 1991.

[29] Wettschereck, D. A study of distance-based machine learning algorithms. PhD Thesis, Department of Computer Science, Oregon State University, Corvallis, OR, 1994.

[30] Wolpert, D. Constructing a generalizer superior to NETtalk via mathematical theory of generalization. *Neural Networks* 3: 445—452, 1989.