J. Expt. Theor. Artif. Intell.
Vol. 16, No. 3, July–September, 2004, 145–160

Taylor & Francis
Taylor & Francis Group

IBR retrieval method based on topology preserving mappings

E. S. CORCHADO[1], J. M. CORCHADO[2] and J. AIKEN[3]

[1]*Departamento de Ingeniería Civil, Universidad de Burgos, C/ Francisco de Vitoria s/n, 09006, Burgos, Spain*
e-mail: escorchado@ubu.es
[2]*Departamento de Informática y Automática, University of Salamanca, Plaza de la Merced s/n, 37008, Salamanca, Spain*
e-mail: corchado@usal.es
[3]*Plymouth Marine Laboratory, Prospect Place, West Hoe, Plymouth, PL1 3DH, UK*
e-mail: ja@pml.ac.uk

Abstract. Case-based reasoning systems, in general, and instance-based reasoning systems, in particular, are used more and more in industrial applications nowadays. During the last few years, researchers have been working in the development of techniques to automate the reasoning stages identified in this methodology. This paper presents a method for automating the retrieval stage and indexation of instance-based reasoning systems. This method is based on a modification of a new type of topology preserving map that can be used for scale invariant classification. The scale invariant map is an implementation of the negative feedback network to form a topology-preserving mapping. Maximum/minimum likelihood learning is applied in this paper to the scale invariant map and its possibilities are explored. This method automates the organization of cases and the retrieval stage of case-based reasoning systems. The proposed methodology groups instances with similar structure, identifying clusters automatically in a data set in an unsupervised mode. The method has been successfully used to completely automate the reasoning process of an oceanographic forecasting system and to improve its performance.

Keywords: case-based reasoning systems, clustering, topology preserving mappings

Received September 2003; accepted March 2004

1. Introduction

The scale invariant map (SIM) was first introduced (Fyfe 1996) as an implementation of the negative feedback network (Fyfe and Baddeley 1995) to form a topology-preserving mapping. In this paper, we apply the maximum/minimum likelihood learning (Corchado *et al*. 2003a) to the scale invariant map and we explore its possibilities in the retrieval stage of instance-based reasoning systems.

Case-based reasoning (CBR) and instance-based reasoning (IBR) systems have been successfully used in several domains such as diagnosis, prediction, control and

Journal of Experimental & Theoretical Artificial Intelligence ISSN 0952–813X print/ISSN 1362–3079 online

DOI: 10.1080/09528130410001726975

planning (Watson 1997, Pal *et al*. 2000, Fyfe and Corchado 2001, Corchado and Aiken 2003). However, a major problem with these systems is the difficulty in instance retrieval and instance matching when the number of cases increases; large instance bases are difficult to handle and require efficient indexing mechanisms and optimized retrieval algorithms. This paper presents a method that can be used to alleviate this problem.

For several years, we have been working on the identification of techniques to automate the reasoning cycle of IBR systems (Fyfe and Corchado 2001, Corchado and Aiken 2003, Corchado *et al*. 2003b). We have used the maximum and minimum likelihood hebbian learning (MLHL) with the scale invariant map (SIM) and shown that it is able to extract information not found with the standard rules (Corchado and Fyfe 2002). MLHL has been successfully used to automate the organization of instances and the retrieval stage of IBR systems as shown in (Corchado *et al*. 2003b). This paper shows how applying maximum\minimum likelihood hebbian learning to the scale invariant map a stronger and efficient classification and retrieval method can be obtained. The power of the method comes from the choice of an appropriate function. A SIM (Fyfe 1996) is a regular array of nodes arranged on a lattice, similar to a self-organizing map (SOM) (Kohonen 1988) but training uses a method based on the negative feedback networks.

Maximum likelihood hebbian learning (MLHL) based models were first developed as an extension of principal component analysis (Oja 1989, Oja *et al*. 1992). The MLHL-based method has been successfully used in the unsupervised investigation of structure in data sets (Corchado *et al*. 2002, 2003a, McDonald *et al*. 2002).

This method can be used in instance-based reasoning systems, where instances are numerical feature vectors. The ability of the maximum likelihood hebbian learning scale invariant map (MLHL-SIM) method presented in this paper to cluster instances and to associate new ones to clusters can be used to successfully select cases and retrieve them.

An instance based reasoning system developed for predicting oceanographic time series ahead of an ongoing vessel (Corchado *et al*. 2001), in real time, is used to illustrate the efficiency of the solution proposed. This paper first presents the MLHL-SIM based method and its theoretical background. Its abilities are demonstrated on synthetic data sets. Finally, we show how this approach has been used in a real-world system to forecast oceanographic thermal time series in real time.

2. Maximum likelihood Hebbian learning-based method

The use of MLHL-based method has been derived from the work of Corchado *et al*. (2002, 2003a) and Fyfe and Corchado (2002a, 2002b) in the field of pattern recognition as an extension of principal component analysis (PCA) (Oja 1989, Oja *et al*. 1992). We first review PCA, which has been the most frequently reported linear operation involving unsupervised learning for data compression, and which aims to find that orthogonal basis that maximizes the data's variance for a given dimensionality of basis. Then the exploratory projection pursuit (EPP) theory is outlined. It is shown how the MLHL-based method may be derived from PCA and it could be viewed as a method of performing EPP. Finally we show why MLHL-SIM based method is appropriate for this type of problem.

2.1. *Principal component analysis (PCA)*

Principal component analysis (PCA) is a standard statistical technique for compressing data; it can be shown to give the best linear compression of the data in terms of least mean square error. There are several artificial neural networks that have been shown to perform PCA, e.g. Oja (1989) and Oja *et al.* (1992). A negative feedback implementation is applied (Fyfe and Baddeley 1995).

The basic PCA network is described by equations (1) and (3). Let us have an N-dimensional input vector at time t, $x(t)$, and an M-dimensional output vector, y, with W_{ij} being the weight linking input j to output i. η is a learning rate. Then the activation passing and learning is described by

$$\text{Feedforward}: \quad y_i = \sum_{j=1}^{N} W_{ij} x_j, \quad \forall i \tag{1}$$

$$\text{Feedback}: \quad e_j = x_j - \sum_{i=1}^{M} W_{ij} y_i \tag{2}$$

$$\text{Change weights}: \quad \Delta W_{ij} = \eta e_j y_i \tag{3}$$

We can readily show that this algorithm is equivalent to Oja's (1989) subspace algorithm:

$$\Delta W_{ij} = \eta e_j y_i = \eta \Big(x_j - \sum_k W_{kj} y_k \Big) y_i \tag{4}$$

and so this network not only causes convergence of the weights but causes the weights to converge to span the subspace of the principal components of the input data. We might then ask why we should be interested in the negative feedback formulation rather than the formulation (4) in which the weight change directly uses negative feedback. The answer is that the explicit formation of residuals (2) allows us to consider probability density functions of the residuals in a way which would not be brought to mind if we use (4). We have developed several successful extensions of the negative feedback network (Corchado *et al.* 2003, 2003a).

Exploratory Projection Pursuit (EPP) is a more recent statistical method aimed at solving the difficult problem of identifying structure in high dimensional data. It does this by projecting the data onto a low dimensional subspace in which we search for its structure by eye. However, not all projections will reveal the data's structure equally well. We therefore define an index that measures how 'interesting' a given projection is, and then represent the data in terms of projections that maximize that index.

The first step in our exploratory projection pursuit is to define which indices represent interesting directions. Now 'interesting' structure is usually defined with respect to the fact that most projections of high-dimensional data onto arbitrary lines through most multi-dimensional data give almost Gaussian distributions (Diaconis and Freedman 1984). Therefore, if we wish to identify 'interesting' features in data, we should look for those directions onto which the data-projections are as far from the Gaussian as possible.

It was shown in Karhunen and Joutsensalo (1994) that the use of a (non-linear) function creates an algorithm to find those values of W which maximize that function whose derivative is $f()$ under the constraint that W is an orthonormal matrix. This was applied in Fyfe and Baddeley (1995) to the above network in the context of the network performing an exploratory projection pursuit. Thus, if we

wish to find a direction which maximizes the kurtosis of the distribution which is measured by s^4, we will use a function $f(s) \approx s^3$ in the algorithm. If we wish to find that direction with maximum skewness, we use a function $f(s) \approx s^2$ in the algorithm.

2.2. *ε-Insensitive hebbian learning*

It has been shown (Xu 1993) that the non-linear PCA rule

$$\Delta W_{ij} = \eta\left(x_j f(y_i) - f(y_i)\sum_k W_{kj} f(y_k)\right) \tag{5}$$

can be derived as an approximation to the best non-linear compression of the data. Thus, we may start with a cost function

$$J(W) = 1^T E\left\{(\mathbf{x} - Wf(W^T\mathbf{x}))^2\right\} \tag{6}$$

that we minimize to get rule (5). Lai *et al.* (2000) used the residual in the linear version of (6) to define a cost function of the residual

$$J = f_1(\mathbf{e}) = f_1(\mathbf{x} - W\mathbf{y}) \tag{7}$$

where $f_1 = \|\cdot\|^2$ is the (squared) Euclidean norm in the standard linear or non-linear PCA rule. With this choice of $f_1()$, the cost function is minimized with respect to any set of samples from the data set on the assumption that the residuals are chosen independently and identically distributed from a standard Gaussian distribution (Bishop 1995). We may show that the minimization of J is equivalent to minimizing the negative log probability of the residual, **e**., if **e** is Gaussian. Let

$$p(\mathbf{e}) = \frac{1}{Z}\exp(-\mathbf{e}^2) \tag{8}$$

then we can denote a general cost function associated with this network as

$$J = -\log p(\mathbf{e}) = (\mathbf{e})^2 + K \tag{9}$$

where K is a constant. Therefore, performing gradient descent on J we have

$$\Delta W \propto -\frac{\partial J}{\partial W} = -\frac{\partial J}{\partial \mathbf{e}}\frac{\partial \mathbf{e}}{\partial W} \approx \mathbf{y}(2\mathbf{e})^{\mathrm{T}} \tag{10}$$

where we have discarded a less important term—see Karhunen and Joutsensalo (1994) for details.

In general (Smola and Scholkopf 1998), the minimization of such a cost function may be thought to make the probability of the residuals more dependent on the probability density function (PDF) of the residuals. Thus if the probability density function of the residuals is known, this knowledge could be used to determine the optimal cost function. Fyfe and MacDonald (2001) investigated this with the (one dimensional) function:

$$p(\mathbf{e}) = \frac{1}{2+\varepsilon}\exp(-|\mathbf{e}|_\varepsilon) \tag{11}$$

where

$$|e|_\varepsilon = \begin{cases} 0 & \forall |\mathbf{e}| < \varepsilon \\ |e| - \varepsilon & \text{otherwise} \end{cases} \tag{12}$$

with ε being a small scalar ≥ 0.

Fyfe and MacDonald (2001) described this in terms of noise in the data set. However, we feel that it is more appropriate to state that, with this model of the PDF of the residual, the optimal $f_1()$ function is the ε-insensitive cost function:

$$f_1(\mathbf{e}) = |\mathbf{e}|_\varepsilon \tag{13}$$

In the case of the negative feedback network, the learning rule is

$$\Delta W \propto -\frac{\partial J}{\partial W} = -\frac{\partial f_1(\mathbf{e})}{\partial \mathbf{e}}\frac{\partial \mathbf{e}}{\partial W} \tag{14}$$

which gives:

$$\Delta W_{ij} = \begin{cases} 0 & \text{if} |\mathbf{e}_j| < \varepsilon \\ \text{otherwise} & \eta y(\text{sign}(\mathbf{e})) \end{cases}. \tag{15}.$$

The difference with the common Hebb learning rule is that the sign of the residual is used instead of the value of the residual. Because this learning rule is insensitive to the magnitude of the input vectors $\mathbf{x}$, the rule is less sensitive to outliers than the usual rule based on mean squared error. This change from viewing the difference after feedback as simply a residual rather than an error permits us to consider a family of cost functions, each member of which is optimal for a particular probability density function associated with the residual.

2.3. *Applying maximum likelihood Hebbian learning*

The MLHL algorithm is constructed now on the bases of the previously presented concepts as outlined here. Now the ε-insensitive learning rule is clearly only one of a possible family of learning rules that are suggested by the family of exponential distributions. This family was called an exponential family in Hyvärinen *et al*. (2002), though statisticians use this term for a somewhat different family. Let the residual after feedback have probability density function

$$p(\mathbf{e}) = \frac{1}{Z}\exp(-|\mathbf{e}|^p). \tag{16}$$

Then we can denote a general cost function associated with this network as

$$J = E(-\log p(\mathbf{e})) = E(|\mathbf{e}|^p + K) \tag{17}$$

where K is a constant independent of W and the expectation is taken over the input data set. Therefore, performing gradient descent on J we have

$$\Delta W \propto -\frac{\partial J}{\partial W}\bigg|_{W(t-1)} = -\frac{\partial J}{\partial \mathbf{e}}\frac{\partial \mathbf{e}}{\partial W}\bigg|_{W(t-1)} \approx E\left\{\mathbf{y}(p|\mathbf{e}|^{p-1}\,\text{sign}(\mathbf{e}))^{\mathrm{T}}\Big|_{W(t-1)}\right\} \tag{18}$$

where T denotes the transpose of a vector and the operation of taking powers of the norm of $\mathbf{e}$ is on an element-wise basis as it is derived from a derivative of a scalar with respect to a vector.

Computing the mean of a function of a data set (or even the sample averages) can be tedious, and we also wish to cater for the situation in which samples keep arriving as we investigate the data set and so we derive an online learning algorithm. If the conditions of stochastic approximation (Kashyap *et al*. 1994) are satisfied, we may approximate this with a difference equation. The function to be approximated is

clearly sufficiently smooth and the learning rate can be made to satisfy $\eta_k \geq 0$, $\sum_k \eta_k = \infty$, $\sum_k \eta_k^2 < \infty$ and so we have the rule:

$$\Delta W_{ij} = \eta\ y_i\ \text{sign}(e_j)|e_j|^{p-1}. \tag{19}$$

We would expect that for leptokurtotic residuals (more kurtotic than a Gaussian distribution), values of $p < 2$ would be appropriate, while for platykurtotic residuals (less kurtotic than a Gaussian), values of $p > 2$ would be appropriate. Researchers from the community investigating independent component analysis (Hyvärinen 2001, Hyvärinen *et al.* 2002) have shown that it is less important to get exactly the correct distribution when searching for a specific source than it is to get an approximately correct distribution i.e. all supergaussian signals can be retrieved using a generic leptokurtotic distribution and all subgaussian signals can be retrieved using a generic platykutotic distribution. Our experiments will tend to support this to some extent but we often find accuracy and speed of convergence are improved when we are accurate in our choice of p. Therefore, the network operation is:

$$\text{Feedforward:} \quad y_i = \sum_{j=1}^{N} W_{ij} x_j, \quad \forall_i \tag{20}$$

$$\text{Feedback:} \quad e_j = x_j - \sum_{i=1}^{M} W_{ij} y_i \tag{21}$$

$$\text{Weight change:} \quad \Delta W_{ij} = \eta\ y_i\ \text{sign}(e_j)|e_j|^{p-1}. \tag{22}$$

Fyfe and MacDonald (2001) described their rule as performing a type of PCA, but this is not strictly true since only the original (Oja) ordinary Hebbian rule actually performs PCA. It might be more appropriate to link this family of learning rules to principal factor analysis since PFA makes an assumption about the noise in a data set and then removes the assumed noise from the covariance structure of the data before performing a PCA. We are doing something similar here in that we are basing our PCA-type rule on the assumed distribution of the residual. By maximizing the likelihood of the residual with respect to the actual distribution, we are matching the learning rule to the probability density function of the residual.

More importantly, we may also link the method to the standard statistical method of exploratory projection pursuit: now the nature and quantification of the degree of interest is in terms of how likely the residuals are under a particular model of the probability density function of the residuals. In the results reported later, we also sphere the data before applying the learning method to the sphered data and show that with this method we may also find interesting structure in the data.

2.4. *Sphering of the data*

Because a Gaussian distribution with mean a and variance x is no more or less interesting than a Gaussian distribution with mean b and variance y—indeed this second order structure can obscure higher order and more interesting structure – we remove such information from the data. This is known as 'sphering'. That is, the raw data is translated till its mean is zero, projected onto the principal component directions and multiplied by the inverse of the square root of its eigenvalue to give

data which has mean zero and is of unit variance in all directions. Therefore, for input data X, we find the covariance matrix

$$\sum = \left\langle (X - \langle X \rangle)(X - \langle X \rangle)^{\mathrm{T}} \right\rangle = UDU^{\mathrm{T}} \tag{23}$$

where U is the eigenvector matrix, D the diagonal matrix of eigenvalues, T the transpose of the matrix and the angled brackets indicate the ensemble average. New samples, drawn from the distribution are transformed to the principal component axes to give y where

$$y_i = \frac{1}{\sqrt{D_i}} \sum_{j=1}^{n} U_{ij}(X_i - \langle X_i \rangle), \quad \text{for } 1 \leq i \leq m \tag{24}$$

where n is the dimensionality of the input space and m is the dimensionality of the sphered data.

3. The scale invariant map architecture

A scale invariant map (SIM) (Fyfe 1996) is a regular array of nodes arranged on a lattice, similar to a self-organizing map (SOM) (Kohonen 1988) but training uses a method based on the negative feedback network. A neighbourhood function and competitive learning are used in the same way as with the SOM. The input data is fed forward via weights W_{ij}(equation 20) from the input neurons (the **x**-values) to the output neurons (**y**-values) where a linear summation is performed to give the activation of the output neuron. After selection of an output winner, the winner, c, is deemed to be firing ($y_c = 1$) and all other outputs are suppressed ($y_i) = 0, \ \forall i \neq c$. The winner's activation is then fed back through its weights and this is substracted from the inputs, and simple Hebbian learning is used to update the weights of all nodes in the neighbourhood of the winner.

Training on a SOM relies on iteratively selecting a winner stimulated by the inputs, and updating the weights. With the scale invariant map, the weights of the winning node are fed back (25) as inhibition at the inputs, and simple Hebbian learning is then used to update the weights of all nodes in the neighbourhood of the winner.

$$\text{Feedback:} \quad \mathbf{e} = \mathbf{x} - W_c y_c, \ (y_c = 1) \tag{25}$$

$$\text{Change weights:} \quad \Delta W_i = h_{ci}\eta \mathbf{e}, \ \forall i \in N_c \tag{26}$$

where **e** is the residual or error, η represents the learning rate coefficient, W_c is the weight connected to the output winner and h_{ci}represents the neighbourhood function, which is a Gaussian function in this case.

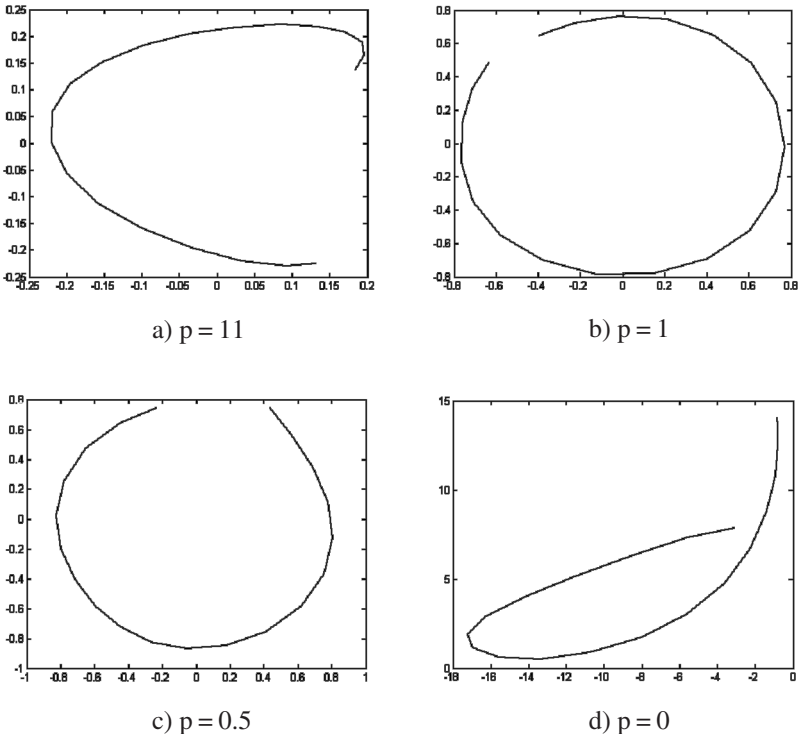
This has the effect of updating all weight vectors in parallel to the vector $\mathbf{x} - W_c$. This is made clear by rewriting (26) as:

$$\Delta W_i = h_{ci}\eta(\mathbf{x} - W_c), \quad \forall i \in N_c \tag{27}$$

If we apply now the MLHL method to the SIM to update the weights, we have the following rule for MLHL-SIM:

$$\text{Weight change:} \quad \Delta W_i = h_{ci}\eta \text{sign}(\mathbf{x} - W_c)|\mathbf{x} - W_c|^{p-1}, \quad \forall i \in N_c \tag{28}$$

with different values of p.



a) p = 11 b) p = 1

c) p = 0.5 d) p = 0

Figure 1. The weight vectors found after 5000 iterations using a learning rate of 0.01 and 20 output neurons.

4. Experiments using an artificial data set

We now investigate the effect of competition when the scale invariant map network is used with different values of p. Our artificial data set is a two dimensional data set taken uniformly from the square [−1,1]*[−1,1].

Figure 1 shows the weight vectors found after 5000 iterations using a learning rate of 0.01 and 20 output neurons. The diagrams show results for $p = 11$, $p = 1$, $p = 0.5$ and $p = 0$. Best results tend to be achieved at $p = 1$ or 0.5. Really, we could not describe the mappings found to be scale invariant for negative values of p. Note also that the larger values of p brings down the magnitude of the mapping. The smaller (especially the negative p values) tends to be unstable in that the weights grow to lie outside the data set, which is [−1,1]*[−1,1].

To understand how the scale invariant map works, recall that it is such that a pie-slice of data is actually won by each neuron (see figure 2).

5. Real-time oceanographic forecasting using an IBR system tool

A forecasting system capable of predicting the temperature of the water ahead of an ongoing vessel in real time has been developed using an IBR system (Corchado *et al.* 2001, Corchado and Aiken 2003). An IBR system was selected for its capacity of handling huge amounts of data, of adapting to the changes in the environment and to provide real time forecast.

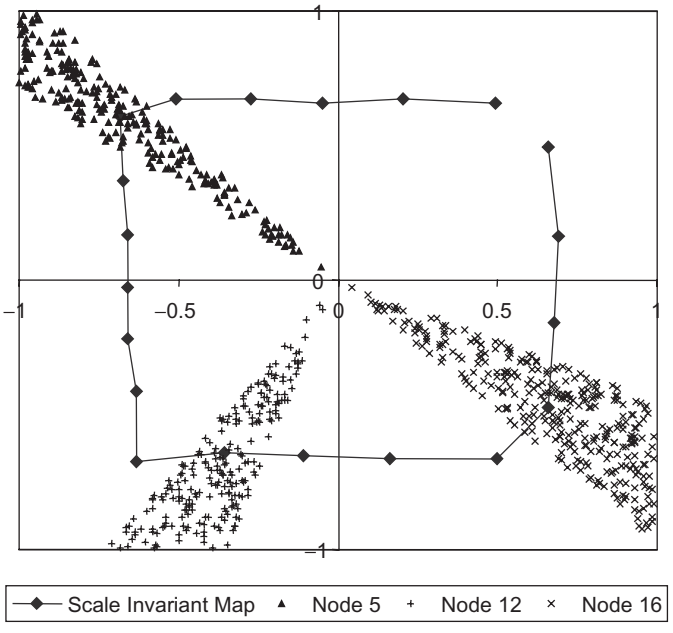


Figure 2. The weights of a scale invariant map trained on uniformly distributed data taken from a square. Each node captures a 'pie-slice' of the data.

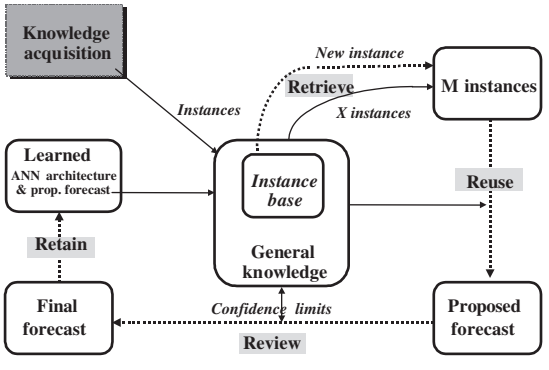


Figure 3. IBR system architecture.

In figure 3, shadowed words (together with the dotted arrows) represent the four steps of a typical IBR life cycle, the arrows together with the words in italics represent data coming in or out of the instance-base (situated in the centre of the diagram) and the text boxes represent the result obtained by each of the four stages of the IBR life-cycle. Solid lines show data flow and dotted lines show the order in which the processes that take part in the life cycle are executed. A case-based reasoning system solves new problems by adapting solutions that were used to solve previous problems (Aamodt and Plaza 1994). The case base holds a number of cases, each of which represents a problem together with its corresponding solution. Once a new problem arises, a possible solution to it is obtained by retrieving similar cases from the case base and studying their recorded solutions.

A typical CBR system is composed of four sequential steps which are recalled every time that a problem needs to be solved: Retrieve the most relevant case(s), reuse the case(s) to attempt to solve the problem, revise the proposed solution if necessary, and retain the new solution as a part of a new case. Figure 3 outlines the basic CBR cycle. Each of the steps of the CBR life cycle requires a model or method in order to perform its mission. The algorithms selected for the retrieval of cases should be able to search the case base and to select from it the most similar problems, together with their solutions, to the new problem. Cases should therefore represent, accurately, problems and their solutions. Once one or more cases are identified in the case base as being very similar to the new problem, they are selected as potential candidates for the solution of this particular problem. These cases are reused using a predefined method in order to generate a proposed solution (i.e. normally using an adaptation technique). This solution is revised (if possible) and finally the new case (the problem together with the obtained solution) is stored. Cases can also be deleted if they prove to be inaccurate; they can be merged together to create more generalized cases and they can be modified. According to Aamodt and Plaza (1994), there are five different types of CBR systems, and although they share similar features, each of them is more appropriate for a particular type of problem: typical case-based reasoning, memory-based reasoning, analogy-based reasoning, exemplar based reasoning and instance based reasoning,

Exemplar-based reasoning systems are derived from a classification of different views of concept definition into 'the classical view', 'the probabilistic view', and 'the exemplar view'. In the exemplar view, a concept is defined extensionally, as the set of its exemplars. CBR methods that address the learning of concept definitions (a problem addressed by much of the research in machine learning), are sometimes referred to as exemplar-based. In this approach, solving a problem is a classification task, i.e. finding the right class for the unclassified exemplar. Instance-based reasoning is a specialization of exemplar-based reasoning into a highly syntactic CBR-approach. This type of CBR system focuses on problems in which there are a large number of instances which are needed to represent the whole range of the domain and where there is a lack of general background knowledge. The case representation can be made with feature vectors and the phases of the CBR cycle are normally automated as much as possible, eliminating human intervention. Basically, this is a non-generalization approach to the concept learning problem addressed by classical, inductive machine learning methods. The lack of general background knowledge may be successfully substituted by a number of instances representative of the whole problem spectrum.

In this case, data are recorded in real time by sensors in the vessels and satellite pictures are received weekly. A *knowledge acquisition module* is in charge of collecting, handling and *indexing* the data in the instance-base. This was initially done by using a rule-based system, which was then improved with the help of MLHL algorithms (see table 1). Once the real-time system is activated on an ongoing vessel, a *new instance* is generated every 2 km using the temperatures recorded by the vessel during the last 40 km. This new instance is used to retrieve *m cases* from a collection of previous cases using Kernel methods (Fyfe and Corchado 2001). This method has been improved by the use of MLHL algorithms (Corchado *et al.* 2003b). The *m-retrieved instances* are adapted by an unsupervised Kernel method during the reuse phase to obtain an initial *(proposed) forecast* (Fyfe and Corchado 2001). Through the revision process, the proposed solution is adjusted to generate the

Table 1. Changes in the IBR system for real time oceanographic forecasting

STEP	Operating IBR system (Fyfe and Corchado 2001)	IBR system using MLHL (Corchado *et al.* 2003)	IBR system using MLHL-SIM
Indexing	Rule-based system	Maximum Likelihood Hebbian Learning algorithm	MLHL-SIM
Retrieval	Kernel methods	Maximum Likelihood Hebbian Learning algorithm	MLHL-SIM
Reuse	Unsupervised Kernel methods	Unsupervised Kernel methods	Unsupervised Kernel methods
Retrain	Kernel methods	Kernel methods/Maximum Likelihood Hebbian Learning algorithm	Kernel methods/ MLHL-SIM

final forecast using the confidence limits from the knowledge base (Corchado and Lees 2001). *Learning* (retaining) is achieved by updating the Kernels/MLHL algorithms. A complete description of this system can be obtained in Fyfe and Corchado (2001) and Corchado *et al.* (2003b). This IBR system has been successfully tested and it is presently operative in several nuclear submarines. Improving this system is the challenge of the STEB (Simulated Tactical Environmental Bubble) project and this section will outline the improvements that have been done to it. When a new technique or methodology, that may improve the forecasting system is obtained, it is tested and evaluated, then it is progressively incorporated to the vessels, normally when the hardware is updated or when scientific missions are carried out in submarines. The model proposed here has been tested and successfully evaluated. It has been demonstrated that the MLHL-SIM-based method performs better than the MLHL algorithm (Corchado *et al.* 2003b) and the Kernel method (Fyfe and Corchado 2001) in the instance indexing and the retrieval process. The following tables shows the changes that have been done in the IBR system for real time oceanographic forecasting. Table 1 outlines the changes made to the original system. The first column of the table indicates in which parts of the IBR system the changes have been made, the second column indicates the method originally used (and now eliminated), column three indicates the method used in a previous attempt to improve the system and column four indicates which methods have been included in the final system modification. The changes indicated in table 1 have been introduced with the intention of developing a robust model, based on a technology easy to implement and that can automate the process of defining the retrieval step of the IBR system, facilitating the indexing of cases and helping in the learning and adaptation stage. The MLHL-SIM algorithm automates these processes, clustering the instances and facilitating the retrieval of the most similar cases to a problem case. In this particular application, the adaptation stage is carried out by an unsupervised kernel network, which structure need to be identified in advance, and tuned manually. We now present the structure of an instance and indicated how the MLHL-SIM algorithm has been used in the mentioned IBR parts.

5.1. *The instance*

Each stored instance contains information relating to a specific situation and consists of an *input profile* (i.e. a vector of temperature values) together with the various fields shown in table 2. A 40 km data profile has been found to give sufficient resolution to

Table 2. Instance structure

Instance field	Explanation
Identification	Unique identification: a positive integer in the range 0 to 64 000
Input profile, I	A 40 km temperature input vector of values *Ij*, (where $j = 1, 2, \ldots, 40$) Representing the structure of the water between the present position of the vessel and its position 40 km back
Output value, F	A temperature value representing the water temperature 5 km ahead of the present location
Time	Time when recorded (although redundant, this information helps to ensure fast retrieval)
Date	Date when the data were recorded (included for the same reasons as for the previous field)
Location	Geographical co-ordinates of the location where the value I40 (of the input profile) was recorded
Orientation	Approximate direction of the data track, represented by an integer x, $(1 \leq x \leq 12)$
Retrieval time	Time when the instance was last retrieved
Retrieval date	Date when the instance was last retrieved
Retrieval location	Geographical co-ordinates of the location at which the instance was last retrieved
Average error	Average error over all forecasts for which the instance has been used during the adaptation step

characterize the problem instance (Fyfe and Corchado 2001, Corchado *et al*. 2001). The parametric features of the different water masses that comprise the various oceans vary substantially, not only geographically, but also seasonally. Because of these variations, it is therefore inappropriate to attempt to maintain an instance base representing patterns of ocean characteristics on a global scale; such patterns, to a large extent, are dependent on the particular water mass in which the vessel may currently be located. Furthermore, there is no necessity to refer to instances representative of all the possible orientations that a vessel can take in a given water mass. Vessels normally proceed in a given predefined direction. Therefore, only instances corresponding to the current orientation of the vessel are normally required at any one time.

5.2. *Indexing, clustering and retrieving instances with the MLHL-SIM algorithm*

To explore the structure and composition of a data set we are using MLHL-SIM. Applying equations 20, 25 and 28 to the instance-base, the MLHL-SIM algorithm groups the instances in clusters automatically. The proposed indexing mechanism classifies the instances automatically, clustering together those of similar structure. This technique is a classification and visualization tool for high dimensional data on a low dimensional display. One of the advantages of this technique is that it is an unsupervised method so we do not need to have any information about the data beforehand. When a new instance is presented to the IBR system, it is identified as belonging to a particular type by applying equations 20, 25 and 28 to it. This mechanism may be used as an universal retrieval and indexing mechanism to be applied to any problem similar to that presented here.

5.3. *Forecasting with the IBR system*

Several experiments have been carried out to illustrate the effectiveness of the IBR system, which incorporates the MLHL-SIM algorithm. Experiments have been carried out using data from the Atlantic Meridian Transect (AMT) Cruise 4 (Corchado *et al.* 2001). We show in figure 4 the errors on a data set of 500 instances randomly taken from the AMT 2000 data set (composed of more than 150 000 instances) using the Kernel based IBR system. Figure 5 shows the results obtained with the new MLHL algorithm.

Figure 6 shows the results obtained with the new MLHL-SIM proposed modification. The mean absolute error, when forecasting the temperature of the water 5 km ahead of an ongoing vessel, along 10 000 km (form the UK to the Falkland Islands) was 0.0205°C with the initial method (Fyfe and Corchado 2001), and of 0.0167°C using the MLHL algorithm (Corchado *et al.* 2003b). With the MLHL-SIM, the average error have been reduced to 0.0136°C, which compares very favourably with the initial instance based reasoning system and other previous methods

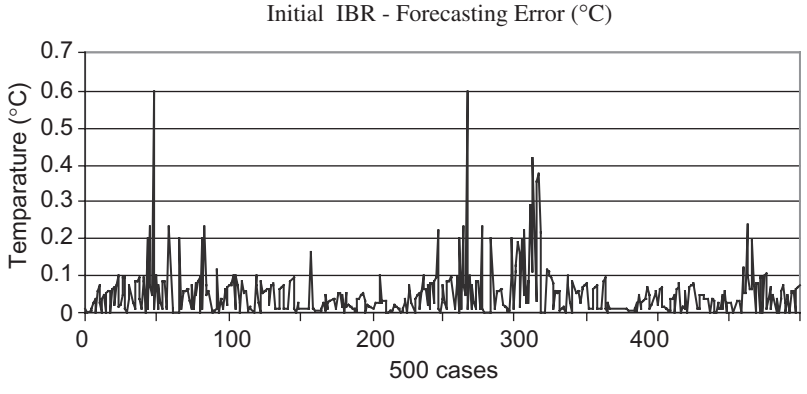


Figure 4. Average error of the working initial IBR system in 500 forecasts carried out during AMT 2000 cruise from UK to Falkland Islands.

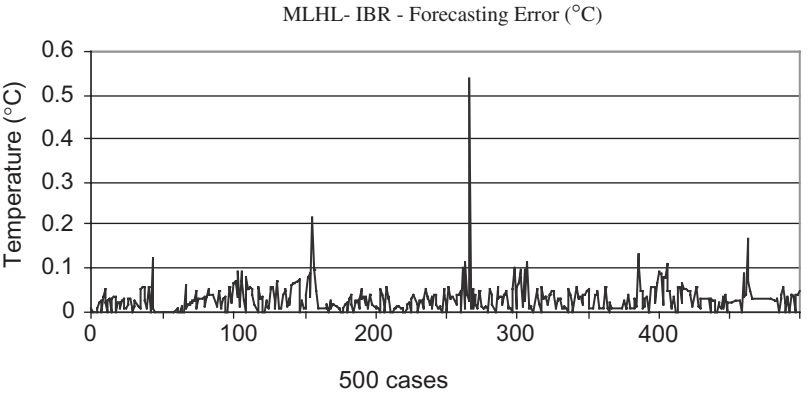


Figure 5. Average error with IBR system using MLHL algorithms in the same 500 predictions as ones in figure 4.

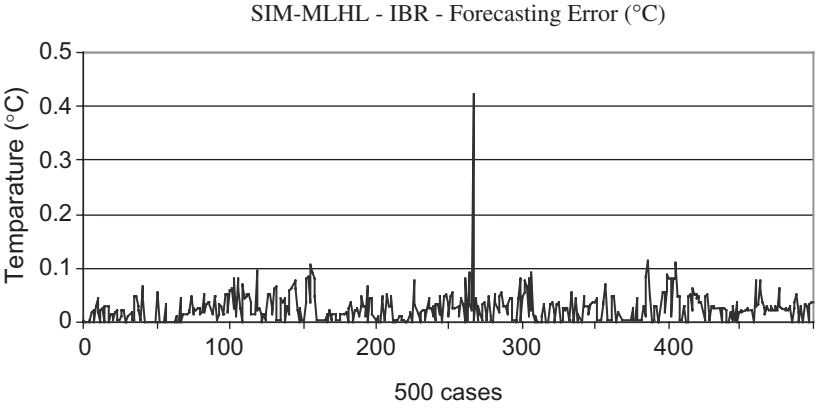


Figure 6. Average error with IBR system using MLHL-SIM algorithms in the same 500 predictions as ones in figures 4 and 5.

(Corchado *et al*. 2001). Using the MLHL-SIM algorithm, the number of predictions with an error higher than 0.1 has been reduced to 3.2%, while before, with the MLHL algorithm, it was 5.6%.

The reason may be that the data selection carried out with the MLHL-SIM algorithm facilitate the creation of more consistent models during the adaptation stage than with the unsupervised Kernel algorithm or the MLHL algorithm. We have compared the proposal presented in this paper with a standard classification algorithm that may be used for the indexing and retrieval of cases such as growing cells structures (GCS). The MLHL-SIM method outperforms the other techniques improving the final results clustering the instances adequately for a future adaptation. The average forecasting error obtained with the GCS was 0.0231°C.

For pedagogical purposes, we illustrate the method on a small sample of cases. A total of 150 instances that characterize the oceanographic problem have been selected from five different areas of the Atlantic ocean (five different water masses). Figure 7 shows the classification ability of the MLHL-SIM proposed, which gives a rather better separation of the individual water masses found in the data set. We are using an unsupervised learning technique in the field of artificial neural networks so generally we do not need any information about the data. Of course, the data must have some kind of structure (correlation, redundancy, etc).

Figure 7 has been obtained because the SIM is a visualization tool for high dimensional data on a low dimensional display. It is composed of a discrete array of L nodes arranged on a N-dimensional lattice and it maps these nodes into D-dimensional data space (in this case D is equal to 2). The array of nodes is typically one or two-dimensional, with all nodes connected to the N inputs by an N-dimensional weight vector. MLHL is a method for identifying structure in high dimensional data. It performs a projection of the data onto a lower dimensional subspace in which we search for its structure with the naked eye.

6. Conclusions

We have demonstrated a new technique for instance indexing and retrieval, which could be used to construct instance based reasoning systems. The basis of the method

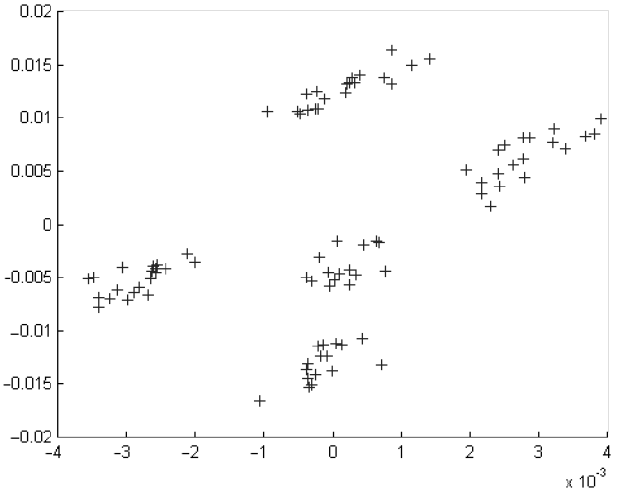


Figure 7. The results of the MLHL-SIM network for $p = 0.5$, classifying the data identifying five different clusters in a better way than the other techniques.

is a MLHL-SIM algorithm. This method provides us with a very robust model for indexing the data and retrieving instances without any need of information about the structure of the data set in advance. It has been shown to give accurate results on an exemplar-forecasting task: our results of 0.0136°C error are the best we have ever achieved on this data set. This is very important in the identification of fronts in these large bodies of water particularly since such fronts have an extremely adverse effect on underwater communications. The retrieval of the best matching instance is a very simple operation using the proposed method and presents no major computational obstacles. The whole system may be used with any number-based data set; an area of ongoing research is the application of algorithms that combine MLHL-SIM and kernel methods. One of the major advantages of the MLHL-SIM is its fast convergence, so that it could be used even in real time problems. This method is also more stable and accurate than previous implementations carried out with Kernel methods, K-nearest neighbour algorithms, and growing cells structures. The proposed method is advantageous in both the creation of and retrieval from instance bases, but is also important in its own right in the unsupervised investigation of data sets.

Acknowledgement

This work has been partially supported by the MCYT project TIC2003-07369-C02-02, the PGIDT00 project MAR30104PR and the project SA039/02 of the JCyL.

References

Aamodt, A., and Plaza, E., 1994, Case-based reasoning: foundational issues, methodological variations, and system approaches. *AICOM*, **7**(1): 39–59.

Bishop, C. M., 1995, *Neural Networks for Pattern Recognition* (Oxford: Oxford University Press).

Corchado, J. M., and Lees, B., 2001, A hybrid case-based model for forecasting. *Applied Artificial Intelligence*. **15**(2): 105–127.

Corchado, E., and Fyfe, C., 2002, The scale invariant map and maximum likelihood Hebbian learning. KES2002. In *Proceedings, Sixth International Conference on Knowledge-Based Intelligent Information Engineering Systems*, Crema, Italy, pp. 245–249.

Corchado, E. S., Han, Y., and Fyfe, C., 2003, Structuring global responses of local filters using lateral connections. *Journal of Experimental and Theoretical Artificial Intelligence,* in press.

Corchado, E. S., MacDonald, D., and Fyfe, C., 2003a, Maximum and minimum likelihood Hebbian learning for exploratory projection pursuit. *Data Mining and Knowledge Discovery,* in press.

Corchado, E., MacDonald, D., and Fyfe, C., 2002, Optimal projections of high dimensional data. ICDM '02. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, IEEE Computer Society; Maebashi TERRSA, Maebashi City, Japan, pp. 589–596.

Corchado, J. M., and Aiken, J., 2003, Hybrid artificial intelligence methods in oceanographic forecasting models. *IEEE SMC Transactions Part C*, **32**(4): 307–313.

Corchado, J. M., Aiken, J., and Rees, N., 2001, *Artificial Intelligence Models for Oceanographic Forecasting* (Plymouth: Plymouth Marine Laboratory).

Corchado, J. M., Corchado, E. S., Aiken, J., Fyfe, C., Fdez-Riverola, F., and Glez-Bedia, M., 2003b, Maximum likelihood Hebbian learning-based retrieval method for CBR systems. In *Proceedings, of the 5th International Conference on Case-based Reasoning*, Trondheim, Norway, June, pp. 107–121.

Diaconis, P., and Freedman, D., 1984, Asymptotics of graphical projections. *The Annals of Statistics*, **12**(3): 793–815.

Fyfe, C., and Baddeley, R., 1995, Non-linear data structure extraction using simple Hebbian networks. *Biological Cybernetics,* **72**(6): 533–541.

Fyfe, C., and Corchado, E., 2002a, Maximum likelihood Hebbian rules. In *Proceedings, of the 10th European Symposium on Artificial Neural Networks, ESANN'2002*, Bruges, Belgium, pp. 143–148.

Fyfe, C., and Corchado, E., 2002b, A new neural implementation of exploratory projection pursuit, In *Proceedings of the Third International Conference on Intelligent Data Engineering and Automated Learning, IDEAL2002*, Manchester, UK, pp. 512–517.

Fyfe, C., and Corchado, J. M., 2001, Automating the construction of CBR systems using kernel methods. *International Journal of Intelligent Systems*, **16**(4): 571–586.

Fyfe, C., and MacDonald, D., 2001, ε-Insensitive Hebbian learning. *Neuro Computing*, **47**(1–4): 35–57.

Fyfe, C., 1996, A scale invariant map network. *Computation in Neural Systems*, **7**: 269–275.

Hyvärinen, A., Karhunen, J., and Oja, E., 2002, *Independent Component Analysis* (New York: Wiley).

Hyvärinen, A., 2001, Complexity pursuit: separating interesting components from time series. *Neural Computation*, **13**: 883–898.

Karhunen, J., and Joutsensalo, J., 1994, Representation and separation of signals using non-linear PCA type learning. *Neural Networks*, **7**: 113–127.

Kashyap, R. L., Blaydon, C. C., and Fu, K. S., 1994, Stochastic approximation. In J. M. Mendel (ed.) *A Prelude to Neural Networks: Adaptive and Learning Systems* (Englewood Cliffs, NJ: Prentice Hall), pp. 121–145.

Kohonen, T., 1988, *Self-organisation and associative memory* (New York: Springer-Verlag).

Lai, P. L., Charles, D., and Fyfe, C., 2000, Seeking independence using biologically inspired artificial neural networks. In M. A. Girolami (ed.) *Developments in Artificial Neural Network Theory: Independent Component Analysis and Blind Source Separation*, (New York: Springer Verlag), pp. 231–242.

Lees, B., and Corchado, J. M., 1999, Integrated case-based approach to problem solving. In F. Puppe (ed.) *Knowledge-based Systems–Survey and Future Directions*. LNAI 1570 (Berlin: Springer), pp. 157–166.

MacDonald, D., Corchado, E., Fyfe, C., and Merenyi, E., 2002, Maximum and minimum likelihood Hebbian learning for exploratory projection pursuit. In *Proceedings, of the International Conference on Artificial Neural Networks, ICANN 2002*, Madrid. Spain, pp. 649–654.

Oja, E., 1989, Neural networks, principal components and subspaces. *International Journal of Neural Systems*, **1**: 61–68.

Oja, E., Ogawa, H., and Wangviwattana, J., 1992, Principal components analysis by homogeneous neural networks, part 1, the weighted subspace criterion. IEICE *Transaction on Information and Systems*, E75D: 366–375.

Pal, S. K., Dillon, T. S., and Yeung, D. S., 2000, *Soft Computing in Case Based Reasoning* (London: Springer Verlag).

Smola, A. J., and Scholkopf, B., 1998, *A Tutorial on support vector regression*. Technical Report NC2-TR-1998-030, NeuroCOLT2 Technical Report Series.

Watson, I., 1997, *Applying case-based reasoning: techniques for enterprise systems* (San Francisco: Morgan Kaufmann).

Xu, L., 1993, Least mean square error reconstruction for self-organizing nets. *Neural Networks*, **6**: 627–648.