ELSEVIER

Advanced Engineering Informatics 16 (2002) 165–178

ADVANCED ENGINEERING INFORMATICS

www.elsevier.com/locate/aei

A comparison of Kernel methods for instantiating case based reasoning systems

Colin Fyfe[a,*], Juan Corchado[b]

[a]Applied Computational Intelligence Research Unit, The University of Paisley, High Street, Paisley PA1-2BE, Scotland, UK
[b]Departamento de Informatica y Automatica, Facultad de Ciencias, Plaza de la Merced S/N, Universidad de Salamanca, Salamanca 37008, Spain

Received 12 June 2001; revised 22 April 2002; accepted 29 April 2002

Abstract

Instance based reasoning systems and in general case based reasoning systems are normally used in problems for which it is difficult to define rules. Instance based reasoning is the term which tends to be applied to systems where there are a great amount of data (often of a numerical nature). The volume of data in such systems leads to difficulties with respect to case retrieval and matching. This paper presents a comparative study of a group of methods based on Kernels, which attempt to identify only the most significant cases with which to instantiate a case base. Kernels were originally derived in the context of Support Vector Machines which identify the smallest number of data points necessary to solve a particular problem (e.g. regression or classification). We use unsupervised Kernel methods to identify the optimal cases to instantiate a case base. The efficiencies of the Kernel models measured as Mean Absolute Percentage Error are compared on an oceanographic problem.

Keywords: Kernel methods; Case based reasoning; Principal Component Analysis

1. Introduction

Case based reasoning (CBR) systems have been successfully used in several domains such as diagnosis, prediction, control and planning [1]. However, a major problem with these systems is the difficulty in case retrieval and case matching when the number of cases increases; large case bases are difficult to handle and require efficient indexing mechanisms and optimised retrieval algorithms, as explained later. Also there are no standard techniques to automate their construction, since each problem may be represented by a different data set and requires a customised solution. This paper compares two groups of Kernel methods that can be used to alleviate these problems.

Kernel models were first developed within the context of Support Vector Machines [2]. Support Vector Machines attempt to identify a small number of data points (the support vectors) which are necessary to solve a particular problem to the required accuracy. Kernels have been successfully used in the unsupervised investigation of structure in data sets [3,4]. We have previously investigated the use of Artificial Neural Networks [5] and Kernel Principal Component Analysis (KPCA) [6] to identify cases, which will be used in a case based reasoning system. In this paper, we compare a sparsified Kernel PCA and three methods based on Kernel K-Means clustering on the same data. Kernel methods map a data set into a Feature space using a non-linear mapping. Then typically a linear operation is performed in the Feature space; this is equivalent to performing a non-linear operation on the original data set. KPCA is one such operation and in this paper we review methods of identifying the critical data points which can be used to sparsify the method of KPCA. Similarly, we investigate clustering in the Feature space and extend the basic method in two distinct ways.

Kernel methods can be used in case based reasoning systems when cases can be represented in the form of numerical feature vectors, examples of which would be temperature (°C), distance (m), time (hh,mm,ss), dates (dd,mm,yy), etc. This is normally the case in most instance based reasoning systems (IBR) [1,7]. The features that characterise Kernel models can be used to identify prototypical cases, to identify cases that are similar to a given one and to reuse cases.

Large case/instance bases may have negative consequences for the performance of the CBR/IBR systems.

* Corresponding author.
E-mail addresses: fyfe-ci0@wpmail.paisley.ac.uk (C. Fyfe), corchado@gugu.usal.es (J. Corchado).

1474-0346/02/$ - see front matter © 2003 Elsevier Science Ltd. All rights reserved.
PII: S1474-0346(02)00008-3

This has been shown in several projects such as INRECA [8] and ORKA [7]. A large case-base requires a complex and efficient indexing mechanism and techniques to eliminate redundant or contradictory cases. When a CBR system is used in a real time problem, such as the oceanographic one presented in this paper, it may not be possible to manage a large case base and the necessary pre-processing algorithms with reasonable computational power. As has been shown in the ORKA [7] project, new and updated cases should be included and maintained in the case base, and obsolete and redundant cases should be eliminated or transformed to maintain a case base with a stable size, in order to control the response time of the system and maintain its efficiency. The transformation of a number of cases into one representative case may help to reduce the volume of information stored in the case base without losing accuracy. The ability of the Kernel methods presented in this paper to select prototypical cases and to identify those cases that are already represented by these prototypes can be used to successfully prune the case-base without losing valuable information.

An instance based reasoning system developed for predicting oceanographic time series ahead of an ongoing vessel, in real time, will be used to illustrate the efficiency of the solution discussed here.

This paper is structured as follows: first CBR systems are reviewed; then Kernel methods are presented, and their 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.

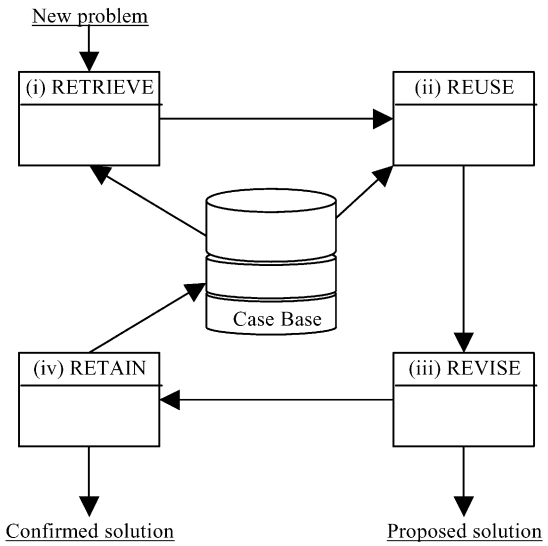
2. Case-based reasoning systems

Although knowledge-based systems (KBS) represent one of the commercial successes resulting from artificial intelligence research, their developers have encountered several problems [9]. Knowledge elicitation, a necessary process in the development of rule-based systems, can be problematic. The implementation of a KBS can also be complex, and, once implemented, it may also be difficult to maintain. With the aim of overcoming these problems, Schank [10] proposed a revolutionary approach, case-based reasoning, which is, in effect, a model of human reasoning. The idea underlying CBR is that people frequently rely on previous problem-solving experiences when solving new problems. This assertion may be verified in many day-to-day problem-solving situations by simple observation or by psychological experimentation [11,12]. Since the ideas underlying case-based reasoning were first proposed, CBR systems have been found to be successful in a wide range of application areas [8,13].

A case-based reasoning system solves new problems by adapting solutions that were used to solve previous problems [14,15]. 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 CBR system is dynamic in the sense that, in operation, cases representing new problems together with their solutions are added to the case base, redundant cases are eliminated and others are created by combining existing cases.

CBR systems record past problem solving experiences and, by means of indexing algorithms, retrieve previously stored cases, along with their solutions, and match them and adapt them to a given situation to generate a solution. The intention of the CBR system is to abstract a solution from the knowledge stored in the case base in the form of cases. All of these actions are self-contained and can be represented by a cyclical sequence of processes in which human intervention may be needed. A case-base reasoning system can be used by itself or as part of another intelligent or conventional system. CBR systems are especially appropriate when the rules that define a knowledge domain are difficult to obtain or the number and the complexity of the rules affecting the problem are too large for the normal knowledge acquisition problem [1,7,13]. Dynamic systems require dynamic solutions and in many cases learning and adaptation mechanisms. When the solution to a problem is known, an ‘expert’ system may be constructed, but when the rules that define a problem change in time in an unpredictable way, we require a system capable of integrating our initial knowledge about the problem and of learning with time. This may be achieved by CBR systems or, as will be explained later, with Instance based systems, which are a particular type of CBR system capable of dealing with large amounts of data.

A typical CBR system is composed of four sequential steps which are recalled every time that a problem needs to be solved [1,9,13]: 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. Fig. 1 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



(i) ***Retrieve*** the most relevant case(s).
(ii) ***Reuse*** the case(s) to attempt to resolve the problem.
(iii) ***Revise*** the proposed solution if necessary.
(iv) ***Retain*** the new solution as a part of a new case.

Fig. 1. The classic CBR cycle.

together to create more generalised cases and they can be modified.

CBR systems are able to utilise the specific knowledge of previously experienced problem situations rather than making associations along generalised relationships between problem descriptors and conclusions or relying on general knowledge of a problem domain such as rule-based reasoning systems. CBR is an incremental learning approach because every time that a problem is solved a new experience can be retained and made immediately available for future retrievals.

The nature of the problem and the expertise of the CBR designers determine how the CBR should be built. Although different metrics and techniques may be used for each of the steps of the CBR cycle, there are only a few of them that can facilitate the automation of the IBR process [1,7,13]. This paper presents a method to automate the process of identifying the significant cases/instances with which to prime the case/instance base in problems of a numeric nature, which may be solved by instance-based reasoning systems. Now we review the different types of CBR systems and how the IBR/CBR systems have been combined and enhanced with other artificial intelligence techniques.

2.1. Types of CBR systems

According to Aamodt and Plaza [1] 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,

Although case-based reasoning is used as a generic term in this paper, the typical case-based reasoning methods have some characteristics that distinguish them from the other approaches listed here. First, a typical case is usually assumed to have a certain degree of richness of information contained in it, and a certain complexity with respect to its internal organisation [1]. CBR systems are also able to modify, or adapt, a retrieved solution when applied in a different problem-solving context. Memory-based reasoning (MBR) systems deal with large collections of cases. In MBR systems, reasoning is seen as the process of accessing and searching in this memory [14,15]. The utilisation of parallel processing techniques is a characteristic of these methods, and distinguishes this approach from the others. Although analogy-based reasoning is used, as a synonym to case-based reasoning, it is also often used to characterise methods that solve new problems based on past cases from a different domain, while typical case-based methods focus on indexing and matching strategies for single-domain cases [16].

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’ [17,18]. 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 specialisation 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 [1,7,8]. 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-generalisation approach to the concept learning

problem addressed by classical, inductive machine learning methods [1]. The lack of general background knowledge may be successfully substituted by a number of instances representative of the whole problem spectrum, as shown in Section 4.

2.2. Using CBR systems in combination with other methods

Instance-based reasoning systems, and in general case-based reasoning systems, require algorithms or mechanism to retrieve, reuse, revise and retain cases [19,20]. CBR/IBR systems are then combined with statistical or artificial intelligence techniques such as artificial neural networks, bayesian networks, genetic algorithms, knowledge-based systems, etc. In general, we can say that CBR systems have to be combined with other reasoning mechanisms and that the final problem solving mechanism may be considered to be a hybrid artificial intelligence system. The term *hybrid* refers to systems that consist of one or more integrated subsystems, each of which can have a different representation language and inference technique. The subsystems are assumed to be tied together semantically and influence each other in some way. The goal of hybrid system research includes the development of techniques to increase the efficiency and reasoning power of intelligent systems. For example, some of the work developed with the aim of increasing efficiency makes use of specialised reasoners strategically called by control or supervisor modules that decide which reasoning method to use at different times [21]. Hybrid systems are capable of addressing some practical problems that have been addressed with traditional artificial intelligence approaches. From a fundamental perspective, hybrid systems may also give further insight into cognitive mechanisms and models [21]. Many researchers have investigated the integration of different AI approaches [22] and in particular the integration of CBR/IBR systems with other techniques [7,20]. The issues under study range from fundamental questions about the nature of cognition and theories of computation to practical problems related to implementation techniques.

Although there are many successful applications based on just CBR technology, from an analysis of this type of system it appears that CBR systems can be successfully improved, combined or augmented by other technologies [23]. Although it may be desirable to have models in which the components are as simple and homogeneous as possible, in some cases a hybrid solution may be the best solution. A hybrid IBR/CBR system may have a clearly identifiable reasoning process. This added reasoning process could be embedded in any of the stages that compose the CBR Life Cycle. For example the most common approaches to construct hybrid based CBR systems are:

- the CBR can work in parallel with a co-reasoner and a control module activates one or the other, i.e. ROUTER [24];
- a co-reasoner can be used as a pre-processor for the CBR system as happens in the PANDA system [25]; and finally
- a CBR can use the co-reasoner to augment one of its own reasoning processes [18] as previously mentioned.

The last approach is used by the majority of the IBR/CBR hybrid systems. The authors in Refs. [7,9,18,23] have investigated the areas where Artificial Intelligence (AI) approaches used as co-reasoners by this type of hybrid IBR/CBR based systems are applied: to define alternative partial solutions, in the adaptation stage, in the evaluation stage, for justification and as a fall back, to generate alternative (partial) solutions, for specification and for repair, etc. They have also identify techniques used to augment the efficiency of IBR/CBR hybrids: rule based reasoning systems, (numerical) constraint satisfaction, qualitative reasoning, genetic algorithms, knowledge-based systems, artificial neural networks, bayesian networks, etc.

Most of the initial work combines IBR/CBR systems with rule-based reasoning systems, but the number of applications in which other AI techniques are combined with instance/case-based reasoning systems is increasing continually and quickly as has been reported in Refs. [9,18, 21,26]. In Ref. [20] a review of the possible ways can be found in which intelligent technologies can be integrated within the CBR cycle.

IBR systems are flexible systems capable of using the beneficial properties of other technologies to their advantage; in particular, the interest here is in the advantages of combining IBR systems and connectionist models such as Kernel methods or artificial neural networks in general. During the last decade an increasing number of scientists had been researching into the hybridisation of IBR systems and connectionist models [7,20]. Connectionist models are not especially appropriate for stepwise expert reasoning and their explanation abilities are extremely weak. Nevertheless their learning and generalisation capabilities can be useful in many problems. Therefore they can only be used as part of IBR/CBR systems in those areas that do not involve knowledge explanation and reasoning. In particular, they can be used in areas involving knowledge generalisation. Learning is a powerful feature of most ANNs, and learning forms an intrinsic part of many stages of the CBR cycle, so ANNs can be used to learn to retrieve the closest case to a particular situation, or in other words to learn to identify the closest matching case. For a connectionist model it is reasonably easy in most situations to learn new cases and to learn how to generalise (adapt) a case from a pool of cases.

CBR systems and connectionist models are complementary techniques, connectionist models deal easily (and normally) with numeric data sets whereas CBR systems deal normally with symbolic knowledge. Even when symbolic knowledge can be transformed into numeric

knowledge and numeric into symbolic, by doing this there is always the risk of losing accuracy and resolution in the data and hence obtaining misleading results. Therefore a combination of IBR/CBR systems and connectionist models may avoid transforming data and therefore gain precision. As mentioned before, generalisation is a useful ability of most connectionist models, but in many cases it is necessary to hold information about special cases, and this is a natural ability of CBR systems. A review of Neuro-symbolic systems that combine CBR systems with connectionist models may be found in Refs. [20,26].

Kernel methods have been grasped by the connectionist community as a means of identifying the crucial data points which are necessary to perform a particular regression or classification task and it is these methods that we now discuss.

3. Kernel methods

The use of radial Kernels has been derived from the work of Vapnik [2], Burges [27] etc. in the field of Support Vector Machines. A very good resource is www.Kernel-machines.org which contains publications, tutorials and software on this topic.

Support Vector Machines (SVMs) perform a non-linear mapping of the data set into some high dimensional feature space in which we may then perform linear operations. Since the original mapping was non-linear, any linear operation in this feature space corresponds to a non-linear operation in data space. SVMs are supervised training methods which were principally derived for regression and classification problems: there is a parameter which determines the level of accuracy of the classification. They have been shown to be very accurate on classification and regression problems though the training times can be excessive.

One of the attractive features of Support Vector Machines is their innate ability to identify which data points are important in determining the optimal regression (or classification) plane and which data points may simply be ignored in creating this optimal plane. This is an attractive property for CBR-IBR systems. We use Kernel methods in an unsupervised manner with the aim of identifying those cases which should be used to prime a case base. The methods are incremental and can be used to update the Case Base as new data arrives.

We first review recent work on Kernel Principal Component Analysis (KPCA) [4,28,29] which has been the most frequently reported linear operation involving unsupervised learning in feature space. We then show why the basic KPCA method is not appropriate for the selection of cases for a CBR–IBR system and create a sparsification of the KPCA method which is appropriate for these types of methods.

3.1. Kernel principal component analysis

In this section, we show that Principal Component Analysis (PCA) may be performed on the samples of a data set in a particular way which will be useful in the performance of PCA in the non-linear feature space. The method crucially uses the dot product of data points rather than the covariance matrix of the data set. We then introduce a non-linear mapping which maps the data from data space to a feature space and show that this new method of performing PCA is appropriate for the mapped data. Indeed, we show that, provided the dot product can be calculated (in the Feature space), we do not need to know the non-linear mapping at all.

PCA finds the eigenvectors and corresponding eigenvalues of the covariance matrix of a data set. Let there be M data points, and let $\chi = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$ be iid (independent, identically distributed) samples drawn from a data source. If each $\mathbf{x}_i$ is n-dimensional, $\exists$ at most n eigenvalues/eigenvectors. Let C be the covariance matrix of the data set; then C is $n \times n$. Then the eigenvectors, $\mathbf{e}_i$, are n-dimensional vectors which are found by solving

$$C\mathbf{e} = \lambda\mathbf{e} \tag{1}$$

where λ is the eigenvalue corresponding to $\mathbf{e}$. We assume the eigenvalues and eigenvectors are arranged in non-decreasing order of eigenvalues and each eigenvector is of length 1. We use the sample covariance matrix as though it was the true covariance matrix and so

$$C \approx \frac{1}{M}\sum_{j=1}^{M}\mathbf{x}_j\mathbf{x}_j^{\mathrm{T}} \tag{2}$$

Now each eigenvector lies in the span of χ; i.e. the set $\chi = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$ forms a basis set (normally overcomplete since $M > n$) for the eigenvectors. So each $\mathbf{e}_i$ can be expressed as

$$\mathbf{e}_i = \sum_j \alpha_j^i \mathbf{x}_j \tag{3}$$

If we wish to find the principal components of a new data point $\mathbf{x}$ we project it onto the eigenvectors previously found: the first principal component is $(\mathbf{x}\cdot\mathbf{e}_1)$, the second is $(\mathbf{x}\cdot\mathbf{e}_2)$, etc. These are the coordinates of $\mathbf{x}$ in the eigenvector basis. There are only n eigenvectors (at most) and so there can only be n coordinates in the new system: we have merely rotated the data set.

Now consider projecting one of the data points from χ on the eigenvector $\mathbf{e}_1$; then

$$\mathbf{x}_k\mathbf{e}_1 = \mathbf{x}_k \cdot \sum_j \alpha_j^1 \mathbf{x}_j = \boldsymbol{\alpha}_1 \cdot \sum_j \mathbf{x}_k\mathbf{x}_j \tag{4}$$

Now let K be the matrix of dot products. Then $K_{ij} = \mathbf{x}_i\mathbf{x}_j$.

Multiplying both sides of Eq. (1) by $\mathbf{x}_k$ we get

$$\mathbf{x}_k C\mathbf{e}_1 = \lambda\mathbf{e}_1 \cdot \mathbf{x}_k \tag{5}$$

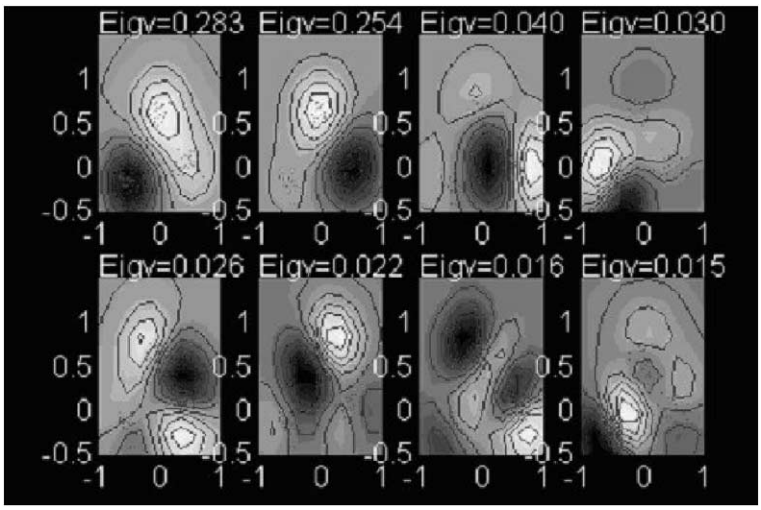


Fig. 2. The three clusters data set is shown as individual points. The contours are contours of equal projection on the respective Principal Components. The first two principal components are sufficient to differentiate between the three clusters; the others slice the clusters internally and have much less variance associated with them.

and using the expansion for $\mathbf{e}_1$, and the definition of the sample covariance matrix, C, gives

$$\frac{1}{M}\mathbf{K}^2\boldsymbol{\alpha}_1 = \lambda_1\mathbf{K}\boldsymbol{\alpha}_1 \tag{6}$$

Now it may be shown [29] that all interesting solutions of this equation are also solutions of

$$\mathbf{K}\boldsymbol{\alpha}_1 = M\lambda_1\boldsymbol{\alpha}_1 \tag{7}$$

whose solution is that α_1 is the principal eigenvector of K.

So far we have only found a rather different way of performing Principal Component Analysis. But now we pre-process the data using $\boldsymbol{\Phi} : \chi \rightarrow F$. So F is now the space spanned by $\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_M)$. The above arguments all hold and we may similarly find the eigenvectors of the dot product matrix $K_{ij} = (\boldsymbol{\Phi}(\mathbf{x}_i)\cdot\boldsymbol{\Phi}(\mathbf{x}_j))$. At this stage we use the *Kernel Trick*: provided we can calculate K we do not need the individual terms $\boldsymbol{\Phi}(\mathbf{x}_i)$.

As an example of how to create the Kernel matrix, we may use Gaussian Kernels so that

$$K_{ij} = (\boldsymbol{\Phi}(\mathbf{x}_i)\cdot\boldsymbol{\Phi}(\mathbf{x}_j)) = \exp\left(\frac{-(\mathbf{x}_i - \mathbf{x}_j)^2}{(2\sigma^2)}\right) \tag{8}$$

This Kernel has been shown [29] to satisfy the conditions of Mercer's theorem and so can be used as a Kernel for some function $\boldsymbol{\Phi}(\cdot)$. One issue that we must address in feature space is that the eigenvectors should be of unit length. Let $\mathbf{v}_i$ be an eigenvector of C. Then $\mathbf{v}_i$ is a vector in the space F spanned by $\boldsymbol{\Phi}(\mathbf{x}_1), \ldots, \boldsymbol{\Phi}(\mathbf{x}_M)$ and so can be expressed in terms of this basis. This is an at most M-dimensional subspace of a possibly infinite dimensional space which gives computational tractibility to the Kernel algorithms. Then

$$\mathbf{v}_i = \sum_{j=1}^{M} \alpha_j^i \boldsymbol{\Phi}(\mathbf{x}_j) \tag{9}$$

for eigenvectors $\mathbf{v}_i$ corresponding to non-zero eigenvalues. Therefore

$$\mathbf{v}_i^{\mathrm{T}}\mathbf{v}_i = \sum_{j,k=1}^{M} \alpha_j^i \boldsymbol{\Phi}(\mathbf{x}_j)^{\mathrm{T}}\boldsymbol{\Phi}(\mathbf{x}_k)\alpha_k^i = \sum_{j,k=1}^{M} \alpha_j^i K_{jk}\alpha_k^i = \boldsymbol{\alpha}^i\cdot(K\boldsymbol{\alpha}^i)$$

$$= \lambda_i\boldsymbol{\alpha}^i\cdot\boldsymbol{\alpha}^i$$

Now $\boldsymbol{\alpha}^i$ are (by definition of the eigenvectors of K) of unit magnitude. Therefore since we require the eigenvectors to be normalised in feature space, F, i.e. $\mathbf{v}_i^{\mathrm{T}}\mathbf{v}_i = 1$, we must normalise the eigenvectors of K, $\boldsymbol{\alpha}^i$, by dividing each by the square root of their corresponding eigenvalues.

Thus we can simply perform a principal component projection of any new point $\mathbf{x}$ by finding its projection onto the principal components of the feature space, F. Thus

$$\mathbf{v}_i\cdot\boldsymbol{\Phi}(\mathbf{x}) = \sum_{j=1}^{M} \alpha_j^i \boldsymbol{\Phi}(\mathbf{x}_j)\cdot\boldsymbol{\Phi}(\mathbf{x}) = \sum_{j=1}^{M} \alpha_j^i K(\mathbf{x}_j, \mathbf{x}) \tag{10}$$

Fig. 2 shows the clustering ability of Kernel PCA with a Gaussian Kernel. The data set comprises three sets each of 30 points each of which is drawn from a Gaussian distribution. The centres of the three Gaussians are such that there is a clear separation between the clouds of points. The figure shows the contours of equal projection onto the first eight KPCA directions. Note that linear PCA would only be able to extract two principal components; however because the Kernel operation has moved us into a high dimensional space in a non-linear manner, there may be up to 90 non-zero eigenvalues. The three clusters can be clearly identified by projecting the data points onto the first two

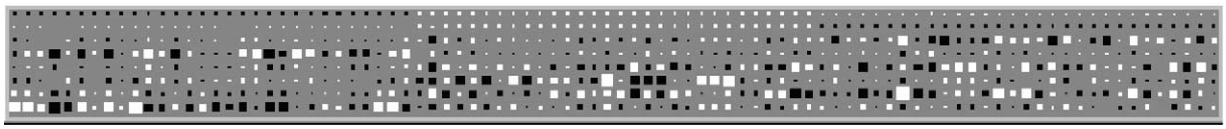


Fig. 3. The first eight eigenvectors found by Kernel PCA. Each eigenvector has elements from every data point.

eigenvectors. Subsequent Kernel Principal Components split the clusters into sections.

However Fig. 3 shows the components of the eigenvectors in Feature space: each row corresponds to one eigenvector in feature space and each column represents a single point in feature space; black indicates that the eigenvector has a negative value in terms of that component while white indicates that it has a positive value; the magnitude of the component is directly proportional to the area of the square. We see why the first two projections were so successful in identifying the three clusters but we note that there is a drawback to the method if we were to use this method to identify cases: each eigenvector is constructed with support from projections of very many points. What we really wish is to identify individual points in terms of their importance. This issue has previously been addressed in Ref. [4] using a number of heuristics. In this paper we use a novel sparsification of the Kernel PCA method.

3.2. Sparse Kernel principal component analysis

One of the attractive properties of Support Vector Machines is their ability to identify the critical data points (the support vectors) which are most useful in classification or regression. However, this property has been lost in KPCA. We consider extensions of KPCA in an attempt to regain this property.

It has recently been suggested [29] that we may reformulate the Kernel PCA problem as follows: let the set of permissible weight vectors be

$$V = \left\{ \mathbf{w} : \mathbf{w} = \sum_{i=1}^{M} \alpha_i \mathbf{\Phi}(\mathbf{x}_i) \right.$$

$$\left. \text{with } \|w\|^2 = \sum_{i,j} \alpha_i \alpha_j K(x_i, x_j) \leq 1 \right\} \tag{11}$$

Then the first principal component is

$$\mathbf{v}_1 = \arg\max_{\mathbf{v} \in V} \frac{1}{M} \sum_{i=1}^{M} |\mathbf{v} \cdot \mathbf{\Phi}(\mathbf{x}_i)|^2 \tag{12}$$

for centred data. This is the basic KPCA definition which we have used above. We may ask whether other sets of permissible vectors may also be found to be useful. Consider

$$V_{\text{LP}} = \left\{ \mathbf{w} : \mathbf{w} = \sum_{i=1}^{M} \alpha_i \mathbf{\Phi}(\mathbf{x}_i) \qquad \text{with } \sum_i |\alpha_i| \leq 1 \right\} \tag{13}$$

This is equivalent to a sparsity regulariser used in supervised learning and leads to a type of Kernel feature analysis

$$\mathbf{v}_1 = \arg\max_{\mathbf{v} \in V_{\text{LP}}} \frac{1}{M} \sum_{i=1}^{M} |\mathbf{v} \cdot \mathbf{\Phi}(\mathbf{x}_i)|^2 \tag{14}$$

Smola et al. [29] point out that this system may be generalised by considering the l_{p} norm to create permissible spaces

$$V_{\text{p}} = \left\{ \mathbf{w} : \mathbf{w} = \sum_{i=1}^{M} \alpha_i \mathbf{\Phi}(\mathbf{x}_i) \qquad \text{with } \sum_i |\alpha_i| \leq 1 \right\} \tag{15}$$

This has been termed Kernel Feature Analysis (KFA) in Ref. [29] however we prefer to call this method Sparse Kernel Principal Component Analysis to emphasise its links with KPCA. In addition, we have also previously [4] used the term KFA to describe Kernel Factor Analysis which has a similar aim—the creation of a sparse representation of data in feature space.

3.3. Solutions and problems

Smola et al. [29] show that the solution of

$$\mathbf{v}_1 = \arg\max_{\mathbf{v} \in V_{\text{p}}} \frac{1}{M} \sum_{i=1}^{M} |\mathbf{v} \cdot \mathbf{\Phi}(\mathbf{x}_i)|^2 \tag{16}$$

are to be found at the corners of the hypercube determined by the basis vectors, $\mathbf{\Phi}(\mathbf{x}_i)$. Therefore all we require to do is find that element $\mathbf{x}_k$ defined by

$$\mathbf{x}_k = \arg\max_{\mathbf{x}_i \in \chi} \sum_{i=1}^{M} |\mathbf{\Phi}(\mathbf{x}_k) \cdot \mathbf{\Phi}(\mathbf{x}_i)|^2 = \arg\max_{\mathbf{x}_i \in \chi} \sum_{i=1}^{M} |K_{ki}|^2 \tag{17}$$

which again requires us only to evaluate the Kernel matrix.

So the solution to finding the 'first Principal Component' using this method is exceedingly simple. We may think that subsequent 'principal vectors' can be found by removing this vector from further consideration and ensuring that the subsequent solutions are all orthogonal to the previously found solutions. However as we shall see there are problems in this simple solution. Consider first the 'naïve' solution which is simply to remove the winner of the first competition from consideration and then repeat the experiment with the remainder of the data points. However these data points may not reveal further interesting structure: typically indeed the same structure in input space (e.g. a cluster) may be found more than once. In the data set to be considered in this paper, this indeed happens. Indeed the first 10 Kernel Principal

Components are in fact all from the same cluster of data and are highly redundant.

An alternative is to enforce orthogonality using a Gram Schmidt orthogonalisation in feature space. Let $\mathbf{v}_1 = \mathbf{\Phi}_1(\mathbf{x}_i)$ for some i. Then

$$\mathbf{\Phi}_2(\mathbf{x}_j) = \mathbf{\Phi}_1(\mathbf{x}_j) - \frac{\mathbf{v}_1}{|\mathbf{v}_1|^2}(\mathbf{\Phi}_1(\mathbf{x}_j)\cdot\mathbf{v}_1)$$

$$= \mathbf{\Phi}_1(\mathbf{x}_j) - \frac{\mathbf{v}_1}{|\mathbf{v}_1|^2}K(\mathbf{x}_j, \mathbf{x}_i)$$

where we have used $\mathbf{\Phi}_1$ to denote the non-linear function mapping the data into feature space and $\mathbf{\Phi}_2$ to denote the mapping after the orthogonalisation has been performed, i.e. this mapping is to that part of the feature space which is orthogonal to the first Principal Component. Using the same convention with the K matrices gives

$$\mathbf{\Phi}_2(\mathbf{x}_j)\cdot\mathbf{\Phi}_2(\mathbf{x}_k)$$

$$= \left(\mathbf{\Phi}_1(\mathbf{x}_j) - \frac{\mathbf{v}_1}{|\mathbf{v}_1|^2}K_1(\mathbf{x}_j, \mathbf{x}_i)\right)\cdot\left(\mathbf{\Phi}_1(\mathbf{x}_k) - \frac{\mathbf{v}_1}{|\mathbf{v}_1|^2}K_1(\mathbf{x}_k, \mathbf{x}_i)\right)$$

$$= K_1(\mathbf{x}_j, \mathbf{x}_k) - \frac{2K_1(\mathbf{x}_j, \mathbf{x}_i)K_1(\mathbf{x}_k, \mathbf{x}_i)}{|\mathbf{v}_1|^2} + \frac{K_1(\mathbf{x}_j, \mathbf{x}_i)K_1(\mathbf{x}_k, \mathbf{x}_i)}{|\mathbf{v}_1|^2}$$

i.e.

$$K_2(\mathbf{x}_j, \mathbf{x}_k) = K_1(\mathbf{x}_j, \mathbf{x}_k) - \frac{K_1(\mathbf{x}_j, \mathbf{x}_i)K_1(\mathbf{x}_k, \mathbf{x}_i)}{K_1(\mathbf{x}_i, \mathbf{x}_i)}$$

which can be searched for the optimal values. The method can clearly be applied recursively and so

$$K_{i+1}(\mathbf{x}_j, \mathbf{x}_k) = K_i(\mathbf{x}_j, \mathbf{x}_k) - \frac{K_i(\mathbf{x}_j, \mathbf{x}_i)K_i(\mathbf{x}_k, \mathbf{x}_i)}{K_i(\mathbf{x}_i, \mathbf{x}_i)}$$

for any time instant $i + 1$.

One difficulty with this method is that we can be (and typically *will be*) moving out of the space determined by the norm. Smola et al. [29] suggest renormalising this point to move it back into V_p. This can be easily done in feature space and both the orthogonalisation and renormalising can be combined into

$$K_{i+1}(\mathbf{x}_j, \mathbf{x}_k)$$

$$= \frac{K_i(\mathbf{x}_j, \mathbf{x}_k)K_i(\mathbf{x}_i, \mathbf{x}_i) - K_i(\mathbf{x}_j, \mathbf{x}_i)K_i(\mathbf{x}_k, \mathbf{x}_i)}{K_i^3(\mathbf{x}_i, \mathbf{x}_i)\{K_i(\mathbf{x}_i, \mathbf{x}_i) + K_i(\mathbf{x}_j, \mathbf{x}_k)\}\{K_i(\mathbf{x}_i, \mathbf{x}_i) - K_i(\mathbf{x}_j, \mathbf{x}_i)\}}$$

which is somewhat a cumbersome expression and must be proved to be a valid Kernel. In this paper we do not perform this step having found it to be unnecessary.

3.4. Using Kernel clustering methods

In this section, we map the data into Feature space and then cluster the data in feature space. We show how this combined mapping and clustering may be accomplished yet again with only the knowledge of the K matrix. We then extend the method so that the clustering has a topology-preserving nature (similar data points are mapped to the same or neighbouring clusters and neighbouring clusters contain only similar points). Finally we show how a vigilance parameter may be used to determine the number of clusters found. These last two methods are Kernel equivalents of Kohonen's Self-organising Map [30] and the ART algorithm [31].

We follow the derivation of Ref. [29] to show how the k-means algorithm may be implemented in Feature space. The aim of the algorithm is to find k means, $\mathbf{m}_\mu$ such that each data point is close to one of the means. Now, as with KPCA, each mean may be described as lying in the manifold spanned by the function of the observations, $\mathbf{\Phi}(\mathbf{x}_i)$, i.e. $m_\mu = \sum_i \gamma_{\mu i}\mathbf{\Phi}(x_i)$. Now the k-means algorithm chooses the means, $\mathbf{m}_\mu$, to minimise the Euclidean distance between the data points and the closest mean. Thus we must calculate

$$\|\mathbf{\Phi}(\mathbf{x}) - \mathbf{m}_\mu\|^2 = \left\|\mathbf{\Phi}(\mathbf{x}) - \sum_i \gamma_{\mu i}\mathbf{\Phi}(\mathbf{x}_i)\right\|^2$$

$$= \mathbf{K}(\mathbf{x}, \mathbf{x}) - 2\sum_i \gamma_{\mu i}\mathbf{K}(\mathbf{x}, \mathbf{x}_i) + \sum_{i,j}\gamma_{\mu i}\gamma_{\mu j}\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)$$

i.e. the distance calculation can be accomplished in Feature space by means of the K matrix alone.

Let $M_{i\mu}$ be the cluster assignment variable, i.e. $M_{i\mu} = 1$ if $\mathbf{\Phi}(\mathbf{x}_i)$ is in the μth cluster and is zero otherwise. We may initialise the means to the first training patterns and then each new training point, $\mathbf{\Phi}(\mathbf{x}_{t+1})$, with $t + 1 > k$ is assigned to the closest mean and its cluster assignment variable calculated using

$$M_{t+1,\alpha} = 1 \qquad \text{if } \|\mathbf{\Phi}(\mathbf{x}_{t+1}) - \mathbf{m}_\alpha\| < \|\mathbf{\Phi}(\mathbf{x}_{t+1}) - \mathbf{m}_\mu\|$$

$$\forall \mu \neq \alpha$$

and is 0 otherwise. We must then update the mean, $\mathbf{m}_\alpha$ to take account of this new data point using $\mathbf{m}_\alpha^{t+1} = \mathbf{m}_\alpha^t + \zeta(\mathbf{\Phi}(\mathbf{x}_{t+1}) - \mathbf{m}_\alpha^t)$ where we have used the term $\mathbf{m}_\alpha^{t+1}$ to denote the updated mean which takes account of the new data point and

$$\zeta = \frac{\mathbf{M}_{t+1,\alpha}}{\sum_{i=1}^{t+1}\mathbf{M}_{i,\alpha}}$$

Now this update rule may be rearranged to give update equations

$$\gamma_{\alpha i}^{t+1} = \begin{cases} \gamma_{\alpha i}^t(1 - \zeta) & i \neq t + 1 \\ \zeta & i = t + 1 \end{cases},$$

an exceedingly simple update rule in Feature space.

We consider in this paper two simple amendments to this simple rule motivated by the literature on Artificial Neural Networks.

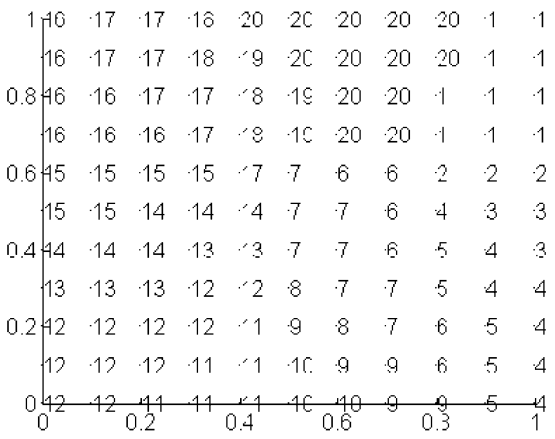


Fig. 4. The KSOM was trained on data iid drawn from a uniform distribution in $[0,1] \times [0,1]$. The figure shows the neuron which was deemed closest in Feature space when data was drawn from the points shown.

First as a direct parallel to Kohonen's Self-Organising Map [30], we introduce neighbourhood relations in the creation of the cluster assignment variables:

$$M_{t+1,\mu} = \Lambda(\alpha, \mu) = a \exp\left(\frac{-(r_\alpha - r_\mu)^2}{\sigma}\right)$$

where $a > b$. The effect of this is to give neighbouring means a bias towards winning competitions for neighbouring data points. This algorithm is extremely fast [4] compared with Kohonen's: in Fig. 4, we show which mean out of 20 possible means won the competition from a regular grid of points in the unit square. To train the algorithm, we have only two passes through a data set consisting of 100 points drawn uniformly from the unit square $[0,1] \times [0,1]$. The first pass acts as a priming mechanism; the second refines the mapping found by the first pass. We see that the centres have each captured a portion of the input space and that similarly numbered centres have captured contiguous areas.

The second change to the Kernel k-means algorithm we consider is motivated by Grossberg and Carpenter's [31] ART Algorithm: the algorithm is intended to resolve the 'stability–plasticity' dilemma which examines how we can continue to learn new things without our old memories being wiped out. The Kernel ART algorithm begins with only a single mean. New means are added when the projection of a particular feature space point is not sufficiently strong; we have a vigilance parameter, v, such that when $\mathbf{\Phi}(\boldsymbol{x}_{t+1}) \cdot \mathbf{m}_\mu < v$ for all μ, we create a new mean exactly at the projection of the data point in Feature space. Notice again that this comparison can be done in feature space using the K matrix. If the largest projection is greater than v, the usual Kernel k-means update rule is employed.

4. Instance-based reasoning for oceanographic real-time forecasting

Several techniques have been used for oceanographic forecasting over the last few years in the framework of the ORKA project [5,32]. In particular a forecasting system capable of predicting the temperature of the water ahead of an ongoing vessel in real time has been developed using a IBR system [32]. 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. The cyclic IBR process shown has been inspired by the ideas described by Aamondt and Plaza [1].

In Fig. 5, shadowed words (together with the dotted arrows) represent the four steps of a typical IBR life cycle, the arrows together with the *word in Italic* represent data coming in or out of the instance-base (situated at the centre of the diagram) and the text boxes represent the result obtained by each of the four stages of the IBR life-cycle.

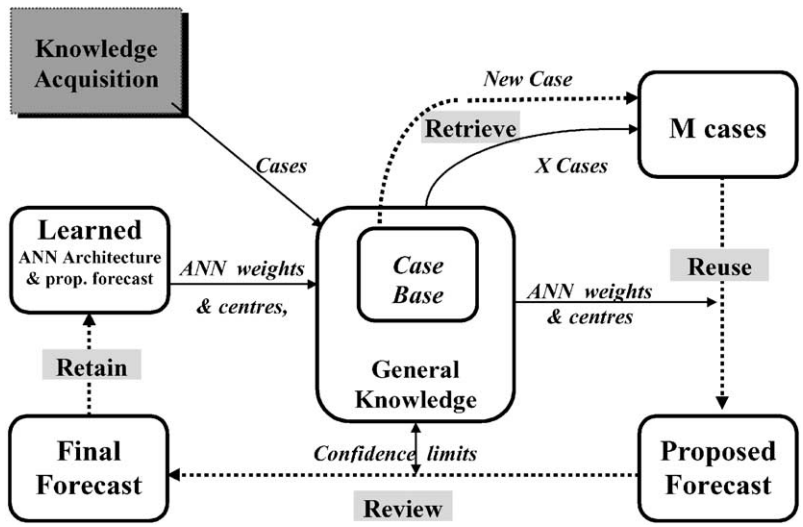


Fig. 5. IBR system architecture.

Table 1
Instance structure

Instance field	Explanation
Identification	Unique identification: a positive integer in the range 0–64 000
Input profile, I	A 40 km temperature input vector of values I_j, (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 I_{40} (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

Solid lines show data flow and dotted lines show the order in which the processes that take part in the life cycle are executed.

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. 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 a number of *K*-nearest neighbour metrics. The *m-retrieved instances* are adapted by a neural network during the reuse phase to obtain an initial (*proposed*) *forecast*. Though the revision process, the proposed solution is adjusted to generate the *final forecast* using the confidence limits from the knowledge base. *Learning* (retaining) is achieved by storing the proposed forecast and knowledge (ANN weights and centres) acquired by the ANN after the training and case adaptation. A complete description of this system can be obtained in Refs. [5,7].

This IBR system has been successfully tested and it is presently operative in several oceanographic vessels [32]. We discuss in this paper how the use of Kernel methods has improved the existing system.

4.1. 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 1.

A 40 km data profile has been found to give sufficient resolution to characterise the problem instance. 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. So only instances corresponding to the current orientation of the vessel are normally required at any one time.

4.2. Creating the case base with sparse Kernel principal component analysis

We use the Sparse KPCA method described in Section 3.3 to create a small number of cases which best typify the data set. For pedagogical purposes, we illustrate the method on a small sample of cases which have been shown to be useful for accurate prediction over three water masses: we have 150 cases of the oceanographic temperature data described above. The data set is illustrated in Fig. 6. The left-hand side diagram shows the first element from each instance; the right-hand side diagram plots the first element from each instance against the value the instance is attempting to predict. The water masses are clearly visible from the data and the strong structure of the data set leads us to believe that there should be much fewer than 150 significant cases.

We have experimented with a number of Sparse KPCA components and illustrate one example of the reduced set in Fig. 7: we show the rows of the K matrix associated with the first 15 Sparse Kernel PCA points. These most important points were 122, 92, 83, 66, 73, 60, 106, 32, 78, 98, 53, 70, 36, 63 and 54: two from the group 101–150, eleven from 51–100 and two from 1–50. We can see from the rows of the K matrix (Fig. 7) that the data set is well covered by these 15 points. It is unsurprising that there are most points from the central group as it contains most structure. We now have a method for identifying the most important points in the data set but there still remains the question of how accurate predictions will be if they only are based on a small set of data samples.

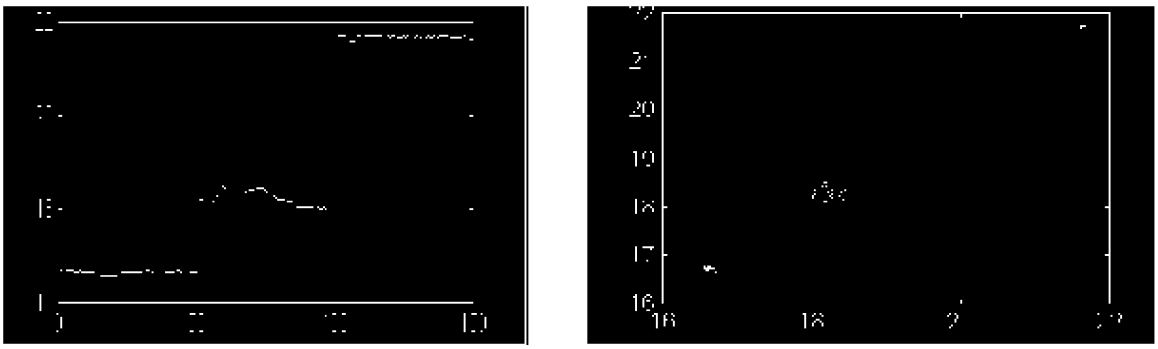


Fig. 6. The data set comprises 50 points from each of three water masses. The left-hand side diagram shows the first element from each instance; the right-hand side diagram plots the first element from each instance against the value the instance is attempting to predict. The water masses are clearly visible from the data.

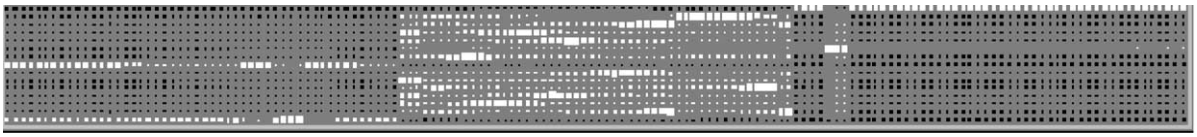


Fig. 7. The 15 rows of the K matrix associated with the first 'Kernel Principal Components' when using the deflationary method.

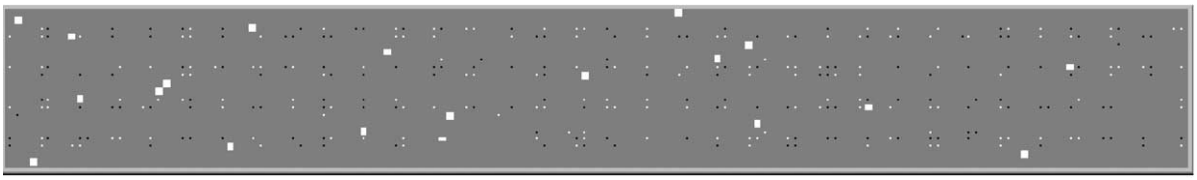


Fig. 8. Twenty centres found by the Kernel k-means algorithm.

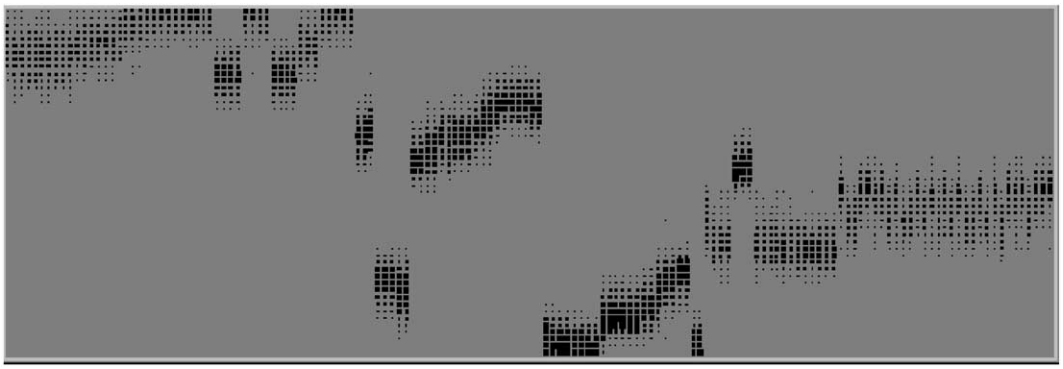


Fig. 9. The structure of the data set is clearly found by the KSOM method (50 means).

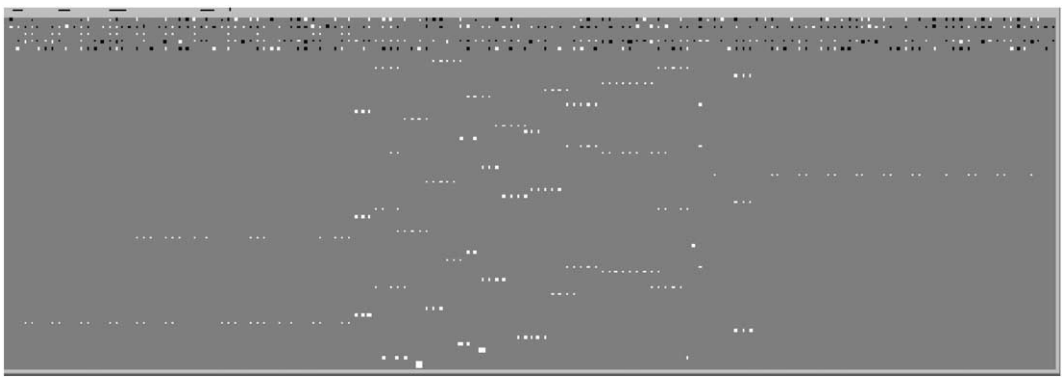


Fig. 10. The KART model with varying vigilance parameter finds varying degrees of structure in the data.

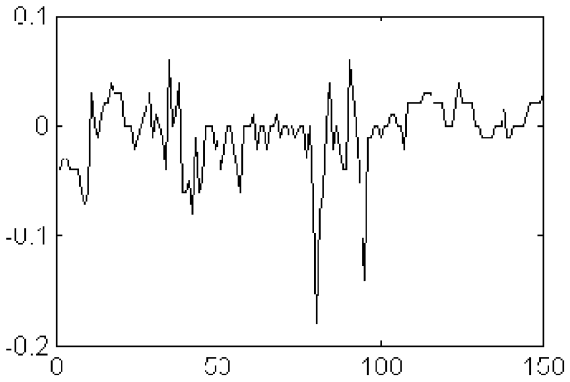


Fig. 11. The error on the 150 points from the Sparse Kernel PCA method. We see that the last group of 50 data points is the easiest to predict. The first group is surprisingly difficult.

4.3. Determining cases using Kernel clustering methods

The Kernel *k*-means algorithm is employed on the same data as the KPCA method. The results are shown in Fig. 8: each row corresponds to the vector of one mean in Feature space. The centres are clearly well spaced throughout the feature space and one might state that each of the three clusters seems to be well represented. Note however that there is no local neighbourhood found so that data points that are close together need not have any special projection on these Kernel vectors.

On the other hand, Fig. 9 shows the *k*-means found by the Kernel SOM method: local structure has been clearly found. This mapping was found with a 50 means mapping. As we reduce the number of means (as we should if we wish to compare with KPCA), the data structure becomes more compressed and the finer details in the mapping are lost.

The Kernel ART model has the advantage over the other clustering methods in that the number of means need not be specified in advance: the method can create new means as and when necessary. Thus in Fig. 10, we have varying values of the vigilance parameter, *v*: we may progressively see more structure developing as we demand more precision in our mapping. In the first mapping we have five means, four of which are being used to identify the three main clusters in the data set. In the second mapping we have rather more centres and some of the clusters are split into subclusters. This process is continued in the final mapping where we see rather a lot of detail in the mapping.

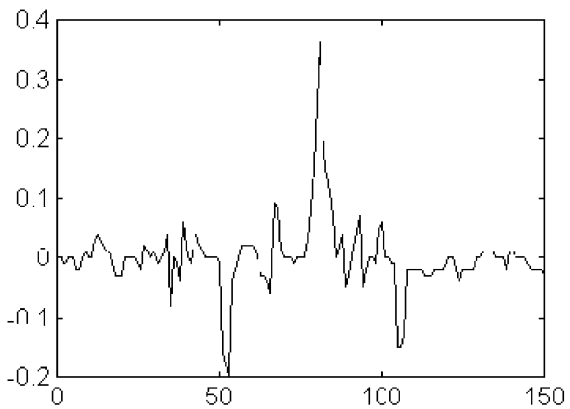


Fig. 12. The error found by the Kernel *K*-means algorithm with 20 centres. The Mean Absolute Error is 0.311.

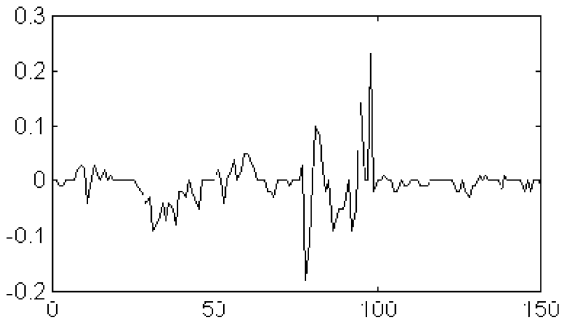


Fig. 13. The errors found by the KSOM method with 50 centres. Because 1/3 of the data points are being used as predictors, there are a great many zero error points and the Mean Absolute Error is 0.212.

4.4. Retrieving cases from the case base

With the Sparse Kernel PCA, any new data point **x** may be associated with a particular case by creating its Kernel projection onto the previously found important points. Given the relatively small number of important points, this is a very fast operation. For the Gaussian Kernels

$$K(\mathbf{x}, \mathbf{x}_j) = \exp\left(-(\mathbf{x} - \mathbf{x}_j)^2/\sigma\right)$$

for all $\mathbf{x}_j$ in the set of stored cases.

It is simple to implement a vigilance parameter so that if the projection on the best case is too small, the point is added to the case base. However, there is no theoretical basis for the choice of the actual value of the vigilance parameter; this is problem dependent and can only be determined by repeated trials and investigation of the corresponding errors.

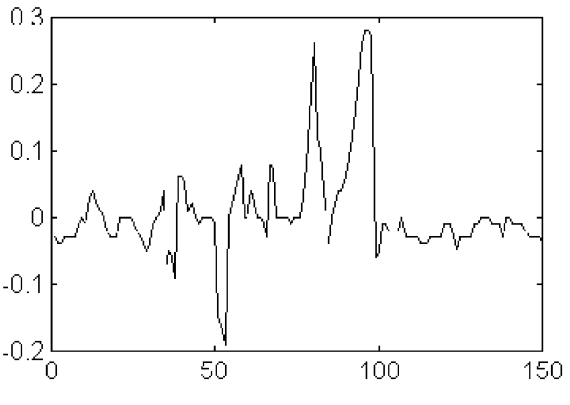


Fig. 14. The errors on the data set when the KSOM method was used with 20 centres. Some areas are very poorly predicted and the Mean Absolute Error is 0.397.

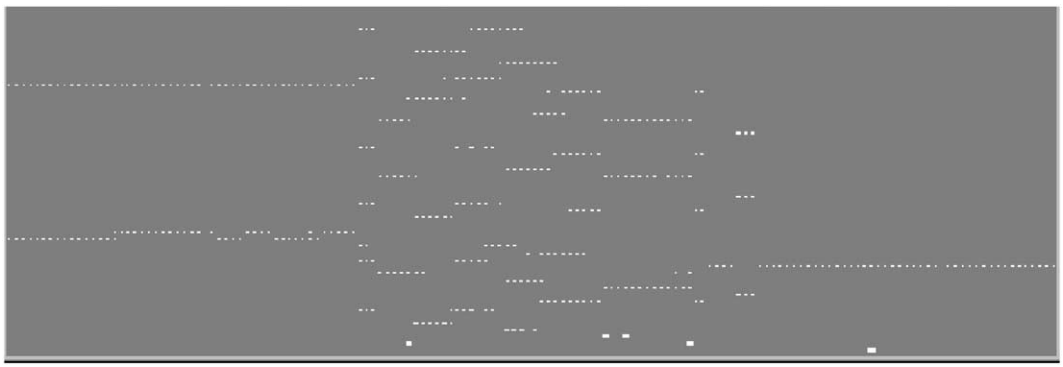


Fig. 15. The filters found by the Kernel Art Algorithm. The Mean Absolute Error in this case is 0.471.

Fig. 16. The rows of the K matrix found by the Simple Sparse KPCA network. Mean Absolute Error is 0.0271 with 17 corners. We see that most of the effort is concentrated on the second section of the data set though the small depression at the start of the third is also well represented.

The projection methods are similar in that any new case is projected onto the existing k-means structure and the largest projection wins the competition.

4.5. Forecasting with the case base

To illustrate the effectiveness of the case base developed using the Sparse KPCA method, we show in Fig. 11 the errors on our original data set of 150 cases of taking the forecast temperature of the retrieved case and subtracting the actual temperature of the case. In this experiment we used 20 cases and so a substantial reduction in cases was achieved. The mean absolute error was 0.0205 which compares very favourably with previous methods. We can also see that the first and second data sets (of 50 samples each) are much more difficult to forecast than the third.

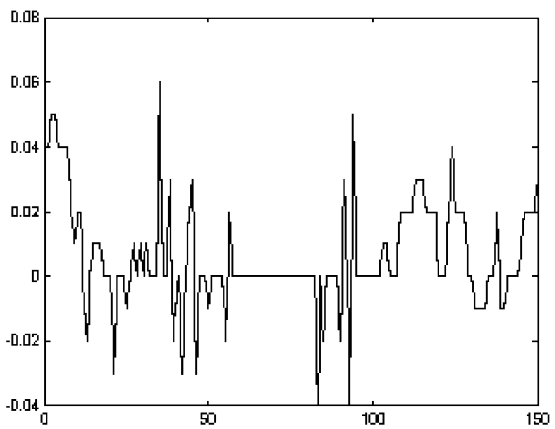


Fig. 17. The figure shows the errors on the data set using the Simple Sparse KPCA method with a small vigilance parameter. The Mean Absolute Percentage Error is 0.0107 but this is at the cost of having 55 points in the case base. The central section is all zeros because each point is being chosen as essential for the case base.

The difficulty with the first data set was not obvious from a visual inspection of the data but becomes obvious when one considers the points found to be important in constructing the case base [7].

The corresponding results on the same data set for the clustering methods are shown in Figs. 12–15. In Fig. 12, we show the errors when we use 20 means with the Kernel k-means method. The Mean Absolute Error is a little larger than when we use the Sparse Kernel PCA method. In Fig. 13, we see the errors induced by the KSOM method when we have 50 centres: the error is low but the use of 50 centres is somewhat at odds with our stated aim of finding a minimal but effective group of prototypes. In Fig. 14, we show the errors when we only use 20 centres with the KSOM method and in Fig. 15 we show the rows of the K matrix when we use a KART algorithm.

In all cases, the Kernel clustering methods are rather worse than the Sparse Kernel Principal Component Analysis.

5. Discussion

The use of Kernel methods for finding those instances which are appropriate for priming a case base has been investigated. In all cases we were able to reduce the number of cases necessary to achieve comparable results with our previous prediction errors. However the Sparse Kernel PCA method consistently out-performed the Kernel Clustering methods. Current investigations are into improvements to the Sparse Kernel PCA method: for example one improvement which is suggested by the ART algorithm is to simply find the greatest projection as before (corresponding to one corner of the Feature space, $\mathbf{\Phi}(\mathbf{x}_j)$) and then put it *and any*

similar vertices into the set of no longer usable corners. We have a vigilance parameter, v, which the similarity measure must exceed if the vertex is to be considered similar enough to the winning vertex. The measure of similarity is simply the dot product in feature space, i.e. the familiar K matrix. This algorithm speeds up the Sparse KPCA and gives comparable results (Figs. 16 and 17).

This method clearly shows a great deal of promise but the interaction between the vigilance parameter and the width of the Kernels is an area of future research.

Acknowledgments

The contributions of N. Rees and Prof. J. Aiken at the Plymouth Marine Laboratory in the collaborative research presented in this paper are gratefully acknowledged. This work has been partially supported by the CICYT projects TEL99-0335-C04-03 and SEC2000-0249 and the PGIDT00 project MAR30104PR.

References

[1] Aamodt A, Plaza E. Case-based reasoning: foundational issues, methodological variations, and system approaches. Artif Intell Commun 1994;7(1):39–59.
[2] Vapnik V. The nature of statistical learning theory. Berlin: Springer; 1995.
[3] Scholkopf BK-R, Smola A, Ratsch G. Nonlinear component analysis as a Kernel eigenvalue problem. Neural Comput 1998;10:1299–319.
[4] Fyfe C, MacDonald D, Lai PL, Rosipal R, Charles D. In: Howlett RJ, Jain LC, editors. Unsupervised learning with radial Kernels in recent advances in radial basis functions. Amsterdam: Elsevier; 2000.
[5] Corchado JM, Fyfe C. Unsupervised neural network for temperature forecasting. Artif Intell Engnng 1999;13(4):351–7. ISSN: 0954-1810.
[6] Fyfe C, Corchado JM. Automating the construction of CBR systems using Kernel methods. Comput Inform Syst J 2001;7(7):29–43. ISBN: 1352-9404.
[7] Corchado JM, Lees B. Adaptation of cases for case-based forecasting with neural network support. In: Pal SK, Dillon TS, Yeung DS, editors. Soft computing in case based reasoning. ISBN 1-85233262-X, London: Springer; 2000.
[8] Bergmann R, Breen S, Göker M, Manago M, Wess S. Developing industrial case-based reasoning applications: the INRECA methodology. Lecture notes in artificial intelligence, state-of-the-art-survey, LNAI 1612, Berlin: Springer; 1999.
[9] Watson I. Applying case-based reasoning: techniques for enterprise systems. Los Altos, CA: Morgan Kaufmann; 1997.
[10] Schank RC. Dynamic memory. Cambridge, NY: Cambridge University Press; 1982.
[11] Corchado JM, Lees B. A hybrid case-based model for forecasting. Appl Artif Intell Int J 2001;15(2):105–27.
[12] Klein GA, Whitaker LA, King JA. Using analogues to predict and plan. In: Kolodner JL, editor. Proceedings of the DARPA Case-Based Reasoning Workshop. Los Altos, CA: Morgan Kaufmann; 1988.
[13] Kolodner J. Case-based reasoning. San Mateo, CA: Morgan Kaufmann; 1993.
[14] Kitano H. Challenges for massive parallelism. IJCAI-93, Proceedings of the Thirteenth International Conference on Artificial Intelligence, Chambery, France, 1993, Los Altos, CA: Morgan Kaufman; 1993. p. 813–34.
[15] Stanfill C, Waltz D. The memory based reasoning paradigm: case based reasoning. Proceedings from a Workshop, Clearwater Beach, Florida, May 1988, Los Altos, CA: Morgan Kaufmann; 1988. p. 414–24.
[16] Veloso MM, Carbonell J. Derivational analogy in PRODIGY. Mach Learn 1993;10(3):249–78.
[17] Kibler D, Aha D. Learning representative exemplars of concepts: an initial study. Proceedings of the Fourth International Workshop on Machine Learning, UC-Irvine; June 1987. p. 24–9.
[18] Porter B, Bareiss R. PROTOS: an experiment in knowledge acquisition for heuristic classification tasks. Proceedings of the First International Meeting on Advances in Learning (IMAL), Les Arcs, France; 1986. p. 159–74.
[19] López de Mántaras R, Plaza E. Case-based reasoning: an overview. AI Commun 1997;10:21–9.
[20] Corchado JM. Neuro-symbolic model for real-time forecasting problems. PhD Dissertation. University of Paisley, Glasgow, UK; 2000.
[21] Medsker LR. Hybrid intelligent systems. Dordrecht: Kluwer; 1995.
[22] Sun R. Commonsense reasoning with rules, cases, and connectionist models: a paradigmatic comparison. Fuzzy Sets Syst 1996;82(2):187–200.
[23] Hunt J, Miles R. Hybrid case-based reasoning. Knowledge Engng Rev 1994;9(4):383–97.
[24] Goel A. A model-based approach to case adaptation. Proceedings Thirteenth Annual Conference of the Cognitive Science Society, Hillsdale, NJ: Erlbaum; 1991.
[25] Roderman S, Tsatsoulis C. PANDA: case-based system to aid novice designers. AI EDAM 1993;7(2):125–33.
[26] Sun R, Alexandre F. Connnectionist-symbolic integration: from unified to hybrid approaches. Hillsdale, NJ: Lawrence Erlbaum Associates; 1997.
[27] Burges C. A tutorial on support vector machines for pattern recognition. Data Mining Knowledge Discover 1998;2(2):121–67.
[28] Scholkopf B, Mika S, Burges C, Knirsch P, Muller K-R, Ratsch G, Smola A. Input space vs feature space in Kernel-based methods. IEEE Trans Neural Networks 1999;10:1000–17.
[29] Smola AJ, Mangasarian OL, Scholkopf B. Sparse Kernel feature analysis. Technical report 99-04, Madison: University of Wisconsin; 1999.
[30] Kohonen T. Self-organising maps. Berlin: Springer; 1995.
[31] Carpenter GA, Grossberg S. The art of adaptive pattern recognition by a self-organising neural network. Computer 1988;2:77–88.
[32] Corchado JM, Aiken J, Rees N. Artificial intelligence models for oceanographic forecasting. UK: Plymouth Marine Laboratory; 2001. ISBN: 9519618-4-5.