Adapting to Drift in Continuous Domains

Miroslav KUBAT
Institute for Systems Sciences, Johannes Kepler University
A-4040 Linz, Austria

Gerhard WIDMER
Department of Medical Cybernetics and Artificial Intelligence
University of Vienna
and
Austrian Research Institute for Artificial Intelligence,
Schottengasse 3, A-1010 Vienna, Austria

Abstract

The paper presents the system FRANN, which exploits the idea of radial-basis functions for the needs of learning in numeric domains under concept drift. The classification accuracy of the program compares favourably to that of older algorithms that are based on symbol manipulation. The system tolerates noise and is able to learn symbolic, numeric, and mixed concepts with nonlinear boundaries in environments with abrupt as well as gradual concept drift.

Research area. Inductive learning

Key words. concept drift, radial-basis functions

Demo request. No

Address for Correspondence:
Miroslav Kubat, Institute for Systems Sciences, Johannes Kepler University, A-4040 Linz, Austria, e-mail: mirek@cast.uni-linz.ac.at

1 Introduction

Recently, the problem of on-line learning in time-varying domains has received attention in the machine learning community. The essence is to make the learner recognize gradual or abrupt changes in the target concept and adjust accordingly the internal representation of the concept. Such changes are usually referred to as *concept drift* and can be caused by a changed *context*. The relevance of this issue to real-world learning tasks has been shown in a case study by Kubat (1992).

Perhaps the first systems capable of tracking concept drift in supervised learning were STAGGER (Schlimmer and Granger, 1986), FLORA (Kubat, 1989), and IB3 (Aha, Kibler, and Albert, 1991). An interesting treatise on this problem was presented by Salganicoff (1993), and the impact of context was studied by Turney (1993). Learning in time-varying environment has also been studied in the framework of genetic algorithms (Smith, 1987), and its importance has long been recognized in adaptive-control applications of neural networks (see, e.g., Narendra and Parthasarathy, 1990, and the references therein). Computational learning theory has also investigated the problem — see, for instance, Helmbold and Long (1991, 1994) and Kuh, Petsche, and Rivest (1991, 1992). In unsupervised learning, the system COBBIT (Kilander and Jansson, 1993) deserves to be mentioned.

The principle of the system FLORA (FLOating Rough Approximation) consists of considering only a set of relatively recent examples (the set is referred to as *window*) and deriving from them three groups of symbolic descriptions: those that are true for *all* positive examples in the window; those that are valid for *some* positive examples; and those that are valid only for negative and *no* positive examples. The idea was later exploited by Widmer and Kubat (1992) in the system FLORA2, which used heuristics for automated adjustment of the window size so as to respond to abrupt or gradual drifts. An extension of this method, FLORA3, presented in Widmer and Kubat (1993), is able to recognize recurring contexts and adjust faster to those that have already appeared in the past. The most powerful descendant of the family, FLORA4 (Widmer, 1994), uses improved criteria for the maintenance and modification of hypotheses. This system is able to discern between noise and context shift, and outperforms its predecessors on a wide range of data.

Most of the drift-tracking systems were primarily developed with symbolic attributes in mind. However, many realistic concepts can only be described by numeric variables (e.g. in the application reported in Kubat, 1992) or by mixed symbolic/numeric representations. This has been recognized, for instance, by researchers working with decision trees (Quinlan, 1993). The simplest approach, applicable in most of the existing symbolic learners, is to

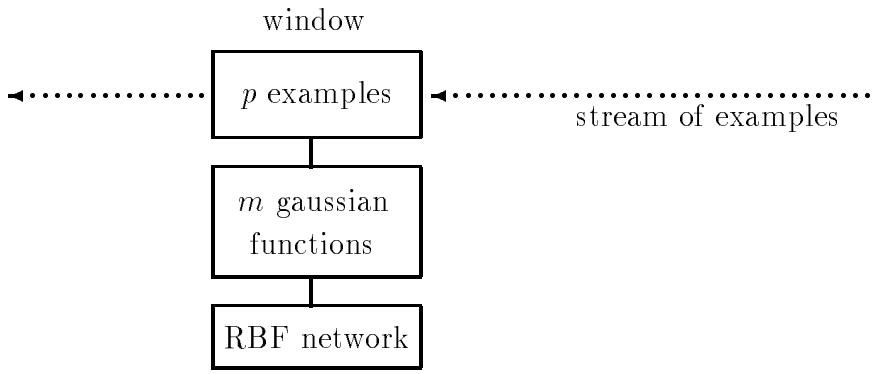

Figure 1: Window passing over the stream of incoming examples and the RBF network derived from it

split the continuous attribute-value ranges into intervals that are then treated as boolean variables. A method for optimal value-range splitting has been suggested by Fayyad and Irani (1992). For the case of more general concept boundaries, multivariate decision trees have been studied that provide piecewise linear approximations of the concept—see Brodley and Utgoff (1994) and the references therein.

To simply import these techniques into the FLORA philosophy is far from trivial because the optimum size of an interval or the position and angle of a linear separator can vary with the concept drift and their reevaluation after each new arrival can become prohibitively expensive. Moreover, the piecewise-linear boundaries are no more than approximations of real-world concepts[1].

We were looking for a new representation scheme that would be intrinsically suited for numeric domains while permitting incremental learning and step-by-step forgetting. The intention was that the learner working with this representation would borrow from FLORA the window mechanism that utilizes only those examples that are believed to be relevant. The following characteristics were required from the learner: (1) ability to learn from examples described by numeric attributes; (2) ability to represent nonlinear concept boundaries; (3) fast adaptation to concept drift; and (4) robustness against noise.

We found the needed representational scheme in the methodology of *Radial Basis Functions*. The general framework of the system FRANN (Floating Rough Approximation in Neural Networks) is depicted in Figure 1. The p examples seen through the window are used as centers of $m \leq p$ gaussian functions. From these, an RBF (Radial-Basis Function) network is created. The network is updated after each change in the window contents. The philos-

[1]Instance-based learners, such as IB3, can acquire very complicated numeric concepts but they are known to need many examples to attain a certain error level.

ophy of RBFs allows FRANN to acquire concepts with nonlinear boundaries, while being computationally fast and efficient in terms of the number of needed examples.

2 Description of the system FRANN

To make the paper self-contained, we first revise the basic principles of RBF networks. Then, we describe the system FRANN and report experiments carried out to study the system's behavior and performance.

2.1 RBF-networks

The rationale behind the RBF approach builds on the mathematical discovery that numeric examples are more likely to be linearly separable if they are nonlinearly mapped to a space with higher dimensionality (Cover, 1965).

Suppose that we have p examples, described by vectors $\mathbf{x_i} = [x_{i1}, \ldots, x_{in}]$, and that some functions $\varphi_j, j = 1, 2, \ldots m$, where $m \geq n$, perform nonlinear mappings of the example space R^n to R^m. Among the possible candidates, the most widely used is the *radial-basis function*:

$$\varphi_j(\mathbf{x_i}) = \frac{1}{2\pi\sigma_j^2} \exp(-\frac{\|\mathbf{x_i} - \mu_\mathbf{j}\|^2}{\sigma_j^2}) \tag{1}$$

$\varphi_j(\mathbf{x_i})$ is a scalar value and Equation (1) represents an n-dimensional gaussian surface, where the vector $\mu_\mathbf{j} = [\mu_\mathbf{j1}, \ldots \mu_\mathbf{jm}]$ defines its center (the coordinates of the point with the maximum value of φ_j) and the scalar σ_j^2 is its standard deviation. The larger the distance of the example $\mathbf{x_i}$ from $\mu_\mathbf{j}$, the smaller the value of $\varphi_\mathbf{j}(\mathbf{x_i})$.

Each example is thus redescribed by the vector $\varphi(\mathbf{x_i}) = [\varphi_1(\mathbf{x_i}), \ldots, \varphi_m(\mathbf{x_i})]$. Figure 2 illustrates how this transformation can be implemented by the hidden layer of a neural network. The transfer function of the neurons in this layer is given by Equation (1) and the transfer function of the output neurons is linear. Each output neuron stands for one concept. In the classification phase, the example is labeled with the i-th concept if i is the index of the neuron with the highest output. The input neurons only distribute attribute values to the hidden layer.

Assume that the centers $\mu_\mathbf{j}$ are put equal to the examples, that is $\mu_\mathbf{j} = \mathbf{x_j}, j = 1, \ldots, p$, which means that the weights along the links between the input and hidden layer are set to the respective attribute values, $w_{ji} = x_{ji}$. We want to determine the weights of the output neurons. Without loss of generality, we will assume a single output neuron with the desired output value $d_i = 1$ if the

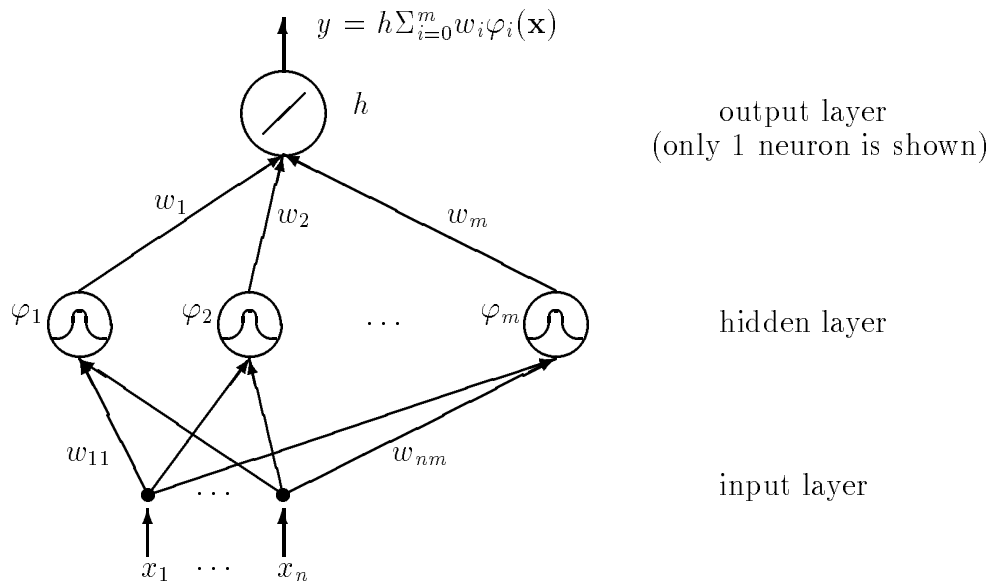

Figure 2: Radial-basis function network

i-th example is a positive instance of the concept represented by this neuron, and $d_i = 0$ otherwise.

Define the $p * p$ matrix $\mathbf{\Phi}$ with elements $\varphi_{ji} = \varphi(\|\mathbf{x_j} - \mathbf{x_i}\|)$. Knowing the vector of classifications $\mathbf{d} = [d_1, \ldots, d_p]^T$ (T stands for a transpose), we want to determine the set of weights $\mathbf{w} = [w_1, \ldots, w_m]^T, (m = p)$ such that:

$$\mathbf{\Phi w} = \mathbf{d} \tag{2}$$

Light (1992) has shown that if the elements of $\mathbf{\Phi}$ are the radial-basis functions, then the matrix is positive definite and, therefore, invertible. This implies that the linear system of Equations (2) has a unique solution given by:

$$\mathbf{w} = \mathbf{\Phi^{-1} d}, \tag{3}$$

where $\mathbf{\Phi^{-1}}$ is the inverse of the matrix $\mathbf{\Phi}$. Hence, the output-layer weights are found by simple matrix manipulation without the need for backpropagation training.

Consider now the case where the number of hidden neurons m is smaller than the number of examples p. The problem is overdetermined and cannot be solved by Equation (3). Instead, the optimum solution minmizing the mean square error can be found by the following formula (see e.g. Golub and Van Loan, 1989):

$$\mathbf{w} = \mathbf{\Phi^{+} d} = (\mathbf{\Phi^T \Phi})^{-1} \mathbf{\Phi^T d} \tag{4}$$

where $\mathbf{\Phi}^+ = (\mathbf{\Phi}^\mathbf{T}\mathbf{\Phi})^{-1}\mathbf{\Phi}^\mathbf{T}$ is called the *pseudoinverse* matrix of $\mathbf{\Phi}$. A standard function for its calculation is available in many mathematical packages.

Although the geometrical and pattern recognition history of radial-basis functions goes back to the 1960s, they were first cast into the neural network setting by Broomhead and Lowe (1988). The low-cost learning and impressive generalization capacity make them attractive expecially for classification tasks.

2.2 The algorithm of FRANN

The main issue to be attacked by a designer of an RBF network is to determine the number and coordinates of hidden units[2]. Providing one hidden neuron for each example usually leads to unnecessarily large networks that might overfit the training set and poorly generalize to unseen examples. Moreover, some of the examples can be noisy, or misclassified, and thus do not deserve to be included in the network. Conversely, if each hidden neuron represents many training examples, the network will tend to overgeneralize the acquired experience, which again adversely affects the classification accuracy.

Attempts to reasonably reduce the number of hidden neurons either pre-cluster the examples, as is the case in Musavi et al. (1992), or use a search technique with operators 'add a neuron' and 'delete a neuron', as suggested by Cheng and Lin (1994). The approach implemented in FRANN is simpler and is made possible by the fact that, thanks to the window philosophy, the number of examples is never prohibitively large.

The idea is to find a subset of the examples that best approximate the concept. This is done by a *hill-climbing* mechanism with a single search operator, 'add a neuron.' The system first takes separately each of the p examples in the window and attempts to create a hidden neuron out of it. Then, the one that provides the most accurate classification of the window examples is picked. Next, other hidden neurons are added, one by one, the evaluation function being the reduction of the error on the window examples. The process terminates when m (a pre-specified number of) hidden neurons have been included.

The algorithm of FRANN is summarized in Table 1. A window slides over a series of examples. From the window examples, the RBF network is created and m is set to $P \times 100$ percent of the window size. $P = 1$ means that all window examples will be utilized. Values $P < 1$ (typically 1/2, 1/3, 1/4, etc.) reduce computational costs, the tradeoff being more reluctant adaptation to drift.

At each step, a batch of N new examples are added to the window ($N = 1$ means to add a single example to the window per step). Then, the perfomance

[2] Let us, for the time being, postpone the question of how to determine σ.

Table 1: Algorithm of the system FRANN

1. Put N new examples in the window;
2. Apply hill-climbing search to find the best subset of the window-examples, and create an RBF network from them; determine the output-layer weights;
3. Use the network for the classification of both training and testing data and collect the respective statistics;
4. Use the window-size heuristic to decide about deletions of older examples from the window;
5. Carry out the deletions and return to step 1.

of the system is tested on the latest M examples (usually $M \neq N$). Concept drift can be recognized by decreased classification accuracy on these examples. A heuristic is applied to decide about the potential need to decrease the window size by deleting from it some of the oldest examples.

The window-size heuristic combines two variables: the current window size l and the number e of erroneous classifications of the last M examples. Basically, old examples are to be forgotten if the window is becoming too large and/or if the number of errors grows (this can signal concept drift). More specifically, let t_1 and t_2 be user-set thresholds. Then:

if $l * e \leq t_1$, then keep the window size fixed (forget N examples);
if $l * e > t_2$, then decrease the window size by 20%;
otherwise, do not forget anything (i.e., grow the window).

The window-size heuristic in FRANN plays a much less significant role than in FLORA. Its main task is to prevent the window from becoming too large.

3 Experimental Evaluation

To assess the performance of FRANN and to observe its behavior, we compared its classification accuracy to that of FLORA4 on several artificial data sets.

For FRANN, symbolic attributes are turned into numeric ones by replacing each attribute-value pair with a boolean variable and putting 0 for *false* and 1 for *true*. Note that if all attributes are symbolic, then the examples are positioned at the corners of the hypercube $[0,1]^n$, where n is the number of the boolean variables redefining the attributes. The concept boundary can then be modeled by a piecewise-linear function without loss of accuracy.

As the experimental data are all normalized into unit cubes, it turned out

reasonable to use $\sigma = 1$ for the standard deviation of the RBF. For a heuristic to determine σ, see, for instance Haykin (1994), Section 7.11.

The version of FLORA4 used in the experiments was the one described in (Widmer, 1994). In order for the system to be able to deal with numeric attributes, it was extended with a simple numeric generalization operator that implements Michalski's (1983) *closing interval rule*. FLORA4 thus approximates concepts in numeric spaces by axis-parallel hyper-rectangles.

3.1 Experimental Data

Below is the characterization of seven arificial data files that we generated so as to assess the performance of FRANN under various conditions—abrupt/gradual drift, presence/absence of noise, symbolic/numeric/mixed data description, presence of irrelevant attributes, and the like.

For each kind of data, 10 training sequences of 200 random examples were generated; the results reported below are averages over the 10 sequences. Except in cases 4 and 6 below, a training sequence embodied two different versions of a target concept that would switch after every 50 training examples. Predictive accuracy of the current hypothesis was tested after every learning step on a set of 100 independent test examples.

Each class was represented by 50% of the examples in each context. To ensure a stable learning environment within each context, the positive and negative examples in the training set alternated. Even though this provision is far from reflecting the situation in real-world tasks, it is helpful if the experimental conditions are to remain constant along the stream of examples.

1. SINE1. *Abrupt concept drift, noise-free examples.* The examples are uniformly distributed in the 2-dimensional unit square $[0,1] \times [0,1]$, with coordinates $[x, y]$. In the first context, all points below the curve $y = \sin(x)$ are classified as positive and all the remaining examples are classified as negative. After each context change, the classification is reversed. Note that the part of the curve that lies in the unit square is nearly linear;
2. SINE2. The same examples are used with changed classification scheme. The curve separating the classes is $y = 0.5 + 0.3\sin(3\pi x)$. Note that this curve can be approximated by a sawtooth function. After each context change, the classification is reversed;
3. SINIRREL. *Presence of irrelevant attributes.* The same examples with the same classifications are used. The only difference is that 2 more random attributes are used to describe the examples. These new attributes have no relation to the class labels. The idea is to test whether the presence of irrelevant attributes adversely affects FRANN's performance;

4. CIRCLES. *Gradual concept drift, noise-free examples.* The same examples are used with a changed classification scheme. This time, four contexts are used. They are defined by the following four circles:

center	[0.2,0.5]	[0.4,0.5]	[0.6,0.5]	[0.8,0.5]
radius	0.15	0.2	0.25	0.3

   Points inside the circles are positive and the points outside the circles are negative. Note that the circles move along a horizontal line, slowly growing in the process;
5. GAUSS.*Abrupt concept drift, noisy examples.* Positive examples from the domain $R \times R$ are normally distributed (according to the gaussian density function) around the center [0,0] with the standard deviation 1. Negative examples are normally distributed around the center [2,0] with standard deviation 4. Note that the two classes heavily overlap, thus modeling the case where the classifications are noisy. After each context change, the classification is reversed. It can be shown analytically that the ideal boundary is a circle around the denser class and that a Bayesian classifier with infinitely many examples achieves 81.53% accuracy;
6. STAGGER. *Abrupt concept drift, symbolic noise-free examples.* Here we use the concepts on which Schlimmer and Granger (1986) tested the performance of their system STAGGER. The examples are described by three symbolic attributes—size (small, medium, large), color (red, green), and shape (circular, non-circular). In the first context, examples satisfying the description `size = small` $\wedge$ `color = red` are classified as positive and the remaining examples are negative. In the second context, the concept description is `color = green` $\vee$ `shape = circular`. In the third context, the description is `size = (medium` $\vee$ `large)`;
7. MIXED. *Abrupt concept drift, mixed boolean/numeric noise-free examples.* The examples are described by two boolean attributes v, w and two numeric attributes x, y from the domain $[0, 1]$. As positive are classified all examples for which at least two of the following three conditions are true: v, w, $y < 0.5 + 0.3\sin(3\pi x)$ (the second sine curve). After each context change, the classification is reversed.

3.2 Results

Figures 3 through 9 compare the accuracies of FRANN vs. FLORA4 on each of the 7 problems described above. The points where a context shift occurs are marked by dashed vertical lines. Figures 8 and 9 give the results of three different versions of FRANN: the curves marked *FRANN 1/3* represent the

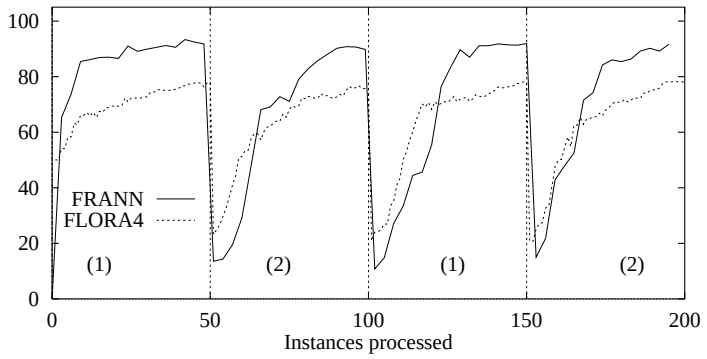

Figure 3: Experiment 1 — Abrupt drift, noise-free examples.

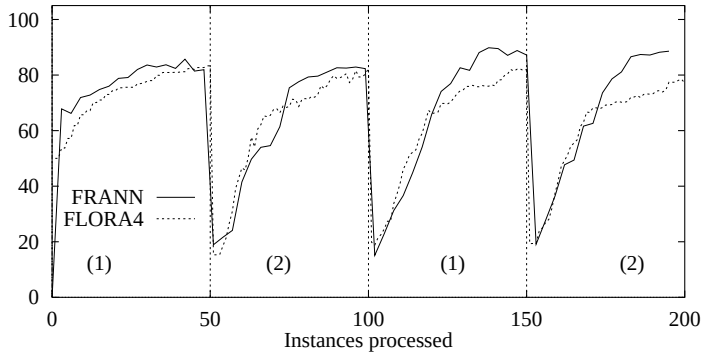

Figure 4: Experiment 2 — Abrupt drift, noise-free examples.

accuracy of FRANN when the number of hidden units in the RBF is 1/3 of the number of examples in the window; *FRANN 2/3* and *FRANN 3/3* are to be interpreted analogously. In all other cases, the number of hidden units is constrained to be at most 1/3 of the window size.

The results speak largely for themselves. FRANN is highly effective in adjusting to changes in the target concepts, and is clearly superior to FLORA4 in most of the numeric problems, with the exception of experiment 4 (*CIRCLES*).

The explanation for the general superiority of FRANN over FLORA4 in numeric domains, especially with target concepts of complex shape, lies in the fact that FLORA4 approximates numeric concepts by axis-parallel (hyper-)rectangles that lie *completely within* the positive concept area (because of the *closing interval rule* generalization operator for numeric attributes). FRANN, on the other hand, can be thought of as representing concepts by a set of prototypes, weighted by the assessed relevance of the prototype to the class. FRANN can thus approximate certain functions more closely.

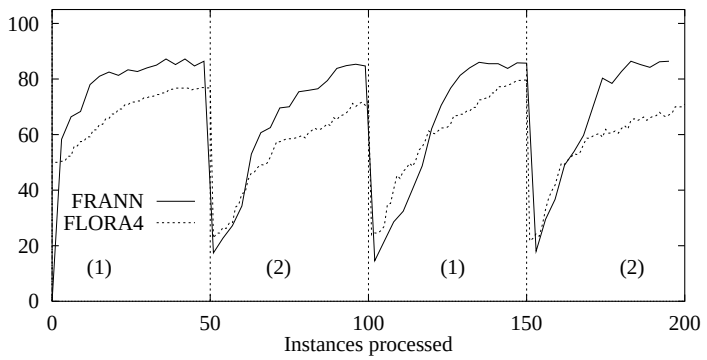

Figure 5: Experiment 3 — Presence of irrelevant attributes.

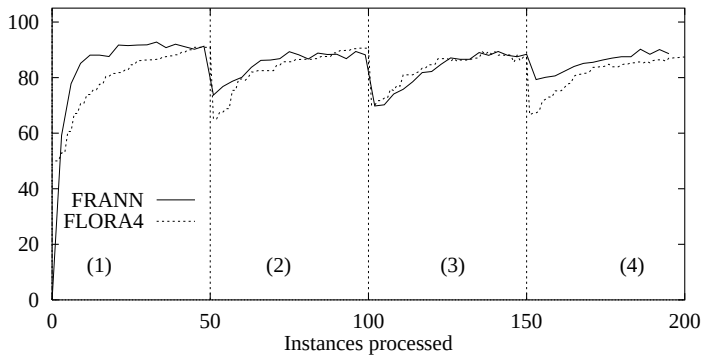

Figure 6: Experiment 4 — Gradual drift, noise-free examples.

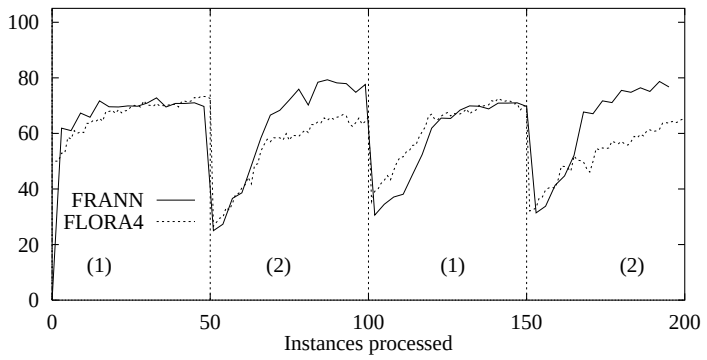

Figure 7: Experiment 5 — Abrupt drift, noisy examples.

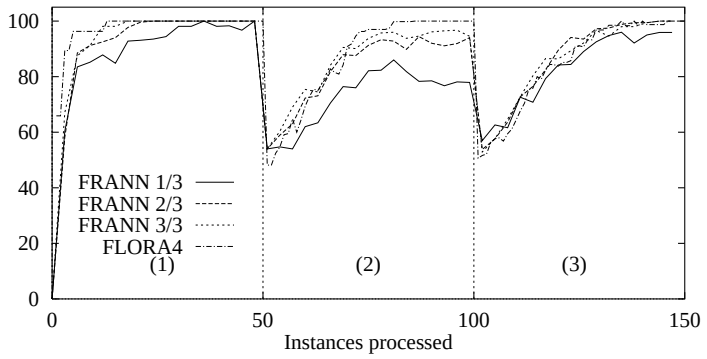

Figure 8: Experiment 6 — Abrupt drift, symbolic noise-free examples.

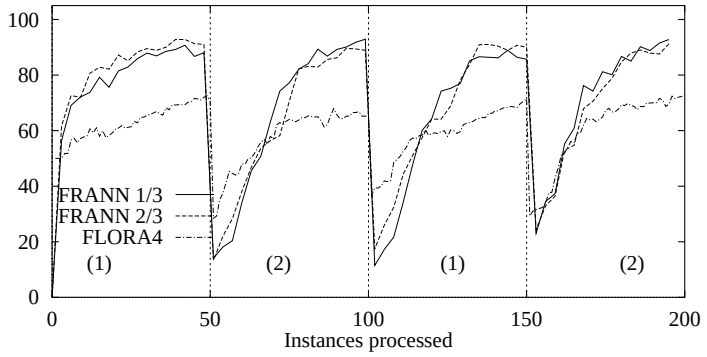

Figure 9: Experiment 7 — Abrupt drift, mixed symbolic-numeric examples.

FLORA4 does better in experiment 2 than in experiment 1 because the function $y = 0.5 + 0.3\sin(3\pi x)$ has a much steeper slope than $y = \sin(x)$ and can thus more easily be approximated by axis-parallel rectangles. However, FRANN is still superior.

In experiment 5 (figure 7), where positive and negative examples are normally distributed around two different centers, FLORA4 does well on the first concept (in fact, equally well as FRANN), but worse on the second. The first concept — positive examples around a center with standard deviation 1 — is much 'denser' than the second one, so it is easier to represent in closed form. A quasi-IBL algorithm like FRANN seems to be at an advantage in the second case. Viewed another way, FRANN represents both classes (positive and negative) explicitly and will thus learn both the concept and its inversion equally easily.

On the other hand, in purely symbolic domains (or similarly in numeric domains where the concept boundaries are axis-parallel), a symbolic algorithm

like FLORA4 can be expected to be better because of its crisp concept representation (and also the more sophisticated windowing mechanism of FLORA4). This is borne out in experiment 6 (figure 8).

Experiment 7, finally, shows FRANN significantly superior to FLORA4. The reason for this huge difference, however, is probably not so much the mixture of symbolic and numeric attributes, but rather the fact that an explicit description of the target concept is very complex vis-a-vis the number of attributes and training examples. This leaves FRANN completely unaffected, but causes problems for FLORA4, whose window adjustment heuristic is based partly on the relative complexity of hypotheses.

To summarize, in numeric domains with concept drift and complex decision boundaries FRANN suggests itself as a powerful alternative to algorithms based on symbolic generalization. Combined with a simple window adjustment heuristic, it exhibits quick adjustment to concept changes and good approximation of complex concepts. The decision surfaces of RBFs are intrinsically non-linear. However, this can become a burden in boolean or generally symbolic domains. Another potential disadvantage of FRANN, which it shares with all neural net approaches, is the poor interpretability of the results of learning.

4 Discussion

Three issues will be discussed: (1) the system's relation to instance-based learners; (2) the use of FRANN's parameters to reflect, heuristically, prior knowledge about the learning task; and (3) the computational complexity of the system.

The fact that the system directly places selected examples in the hidden layer makes it perhaps more closely related to *instance-based learners* than to the FLORA-family. The improvement in classification accuracy, as compared to IB3, is facilitated by the existence of the weights to the output neurons. Those of the examples that are less typical obtain smaller weights and their influence on the classification will thus be limited. The hill-climbing search for the best-suited examples, and the fact that only a subset of examples are actually utilized (for $P < 1$), further reduce the system's sensitivity to the presence of noise. In effect, the search replaces IB3's statistical measure deciding which of the exemplars are 'good,' 'mediocre,' and 'bad.' The idea of non-linear mapping of the examples from R^n to R^m facilitates faster learning in terms of the required number of examples because the transformed examples are more likely to be linearly separable.

Contrary to IBL approaches, however, FRANN is not a strictly incremental

learner. Making FRANN incremental will require a modification of the search and the introduction of a 'delete-a-neuron' operator.

As is typical for neural networks, FRANN relies on several *parameters*. This is not a generally desirable situation, but on the other hand, the parameters make it possible to supply the learner with some prior knowledge about the learning task: the extent and frequency of the concept drift, complexity of the concept to be learned, the shortage of abundance of learning examples, noise extent, and the like.

Most important of the parameters is the number of hidden neurons in the window, as specified by the parameter P. Complicated concepts will certainly require more hidden neurons to provide more accurate approximation. Also the expected noise level affects the number of necessary neurons because, in FRANN, noise is partially eliminated by the fact that only some of the examples get incorporated into the hidden layer.

The system's performance is influenced by the number of examples M on which on-line accuracy is measured. Small values of M are insufficient for reliable decisions and make FRANN sensitive to noise. Conversely, large values tend to be conservative because the system will require too many examples to come to believe that a concept drift actually occurred.

The fact that the examples are processed in batches is a generalization of the original windowing idea. Even though the size N of the batch can be set to 1 (as is the case in the FLORA-based programs), we preferred to use $N = 3$ so as to speed up the experimental runs.

The ideal values for the parameters t_1 and t_2 that are used for the window-size adjustment were determined empirically. Even though heuristics for their automatic setting could also probably be found, we decided not to investigate this general problem at this point in our research.

For simplicity, the standard deviation σ in Equation (1) was in our experiments always set to 1. Methods for its domain-dependent setting are discussed in Haykin (1994), Section 7.11.

The *complexity* of the FRANN algorithm is given by the hill-climbing search, which is polynomial in the window size. The cost of the computation of the output-layer weights depends on the complexity of the inverse-matrix calculation, which is $O(p^3)$, where p is the dimension of the matrix. This is acceptable because the number of hidden units is very small (in the above experiments their number usually did not exceed 10). Moreover, FRANN could be implemented in VLSI RBF-circuits that are already available on the market; this will make it applicable also for large, real-world problems.

5 Conclusion

The experiments reported in this paper demonstrate that the system FRANN compares favourably with its predecessors in the presence of concept drift. Learning is possible from examples described by symbolic as well as by numeric attributes and the system is able to learn and recognize concepts with nonlinear boundaries.

As a concept representation scheme, the framework of radial-basis functions has been introduced. Even though this formalism violates the requirement of understandability of the concept description, it will be useful in applications where classification accuracy is emphasized at the cost of interpretability.

References

Aha, D., Kibler D., and Albert, M.K (1991). Instance-Based Learning Algorithms. *Machine Learning* 6:37–66.

Brodley, C.E. and Utgoff, P.E. (1994). Multivariate Decision Trees. *Machine Learning*, to appear.

Broomhead, D.S. and Lowe, D. (1988). Multivariable Functional Interpolation and Adaptive Networks. *Complex Systems*, 2:321–355.

Cheng, Y.-H. and Lin, C.-S. (1994). A Learning Algorithm for Radial Basis Function Networks with the Capability of Adding and Pruning Neurons. *Proceedings of the IEEE*, 797–801.

Cover, T.M. (1965). Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition. *IEEE Transactions on Electronics Computers*, EC-14, 326–334.

Fayyad, U.M. and Irani, K.B. (1992). On the Handling of Continuous-Valued Attributes in Decision Tree Generation. *Machine Learning* 8:87–102.

Golub, G.H. and Van Loan, C.F. (1989). *Matrix Computation*, 2nd ed., Johns Hopkins University Press, Baltimore.

Haykin, S. (1994). *Neural Networks, A Comprehensive Foundation.* Macmillan College Publishing Company, New York.

Helmbold, D.P. and Long, P.M. (1991). Tracking Drifting Concepts Using Random Examples. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory (COLT-91)*, Santa Cruz, CA, 13–23.

Helmbold, D.P. and Long, P.M. (1994). Tracking Drifting Concepts by Minimizing Disagreements. *Machine Learning* 14:27–45.

Kilander, F. and Jansson, C.G. (1993). COBBIT - A Control Procedure for COBWEB in the Presence of Concept Drift. In *Proceedings of the European Conference on Machine Learning (ECML-93)*, Vienna, Austria.

Kubat, M. (1989). Floating Approximation in Time-Varying Knowledge Bases. *Pattern Recognition Letters*, 10:223–227.

Kubat, M. (1992). A Machine Learning Based Approach to Load Balancing in Computer Networks. *Cybernetics and Systems* 23:389–400.

Kuh, A., Petsche, T. and Rivest, R.L. (1991). Learning Time-Varying Concepts. In *Advances in Neural Information Processing Systems (NIPS)* 3, 183–189. San Mateo, CA: Morgan Kaufmann.

Kuh, A., Petsche, T. and Rivest, R.L. (1992). Incrementally Learning Time-varying Half-planes. In *Advances in Neural Information Processing Systems (NIPS)* 4, 920–927. San Mateo, CA: Morgan Kaufmann.

Light, W.A. (1992). Some Aspects of Radial Basis Function Approximation. In: Singh, S.P. (ed.): *Approximation Theory, Spline Functions and Applications*, NATO ASI Series, Vol. 256, Kluwer Academic Publishers, Boston, 163–190.

Michalski, R.S. (1983). A Theory and Methodology of Inductive Learning. *Artificial Intelligence* 20(2), 111–161.

Musavi, M.T., Ahmed, W., Chan, K.H., Faris, K.B., and Hummels, D.M. (1992). On the Training of Radial Basis Function Classifiers. *Neural Networks*, 5:595–603.

Narendra, K.S. and Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks* 1:4–27.

Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo.

Salganicoff, M. (1993). Density-Adaptive Learning and Forgetting. *Proceedings of the 10th International Conference on Machine Learning.* Amherst, MA.

Schlimmer, J.C. and Granger, R.H. (1986). Incremental Learning from Noisy Data. *Machine Learning 1*, 317–354.

Smith, R.E. (1987). Diploid genetic algorithms for search in time varying environments. *Proceedings of the International Conference on Genetic Algorithms and their Applications*, 202–206.

Turney, P.D. (1993). Robust Classification with Context-Sensitive Features. *Proceedings of the Sixth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Edinburgh, Scotland.

Widmer, G. (1994). Combining Robustness and Flexibility in Learning Drifting Concepts. In *Proceedings of the 11th European Conference on Artificial Intelligence (ECAI94)*, Amsterdam. Chichester: Wiley & Sons.

Widmer, G. and Kubat, M. (1992). Learning Flexible Concepts from Streams of Examples: *FLORA2*. In *Proceedings of the European Conference on Artifical Intelligence (ECAI-92)*, Vienna, Austria.

Widmer, G. and Kubat, M. (1993). Effective Learning in Dynamic Environments by Explicit Context Tracking. In *Proceedings of the European Conference on Machine Learning (ECML-93)*, Vienna. Berlin: Springer Verlag.