Evolving Fuzzy Neural Networks for On-line Knowledge Discovery

Nikola Kasabov
Department of Information Science
University of Otago, P.O Box 56, Dunedin, New Zealand
Phone: +64 3 479 8319, fax: +64 3 479 8311, nkasabov@otago.ac.nz

Abstract. Fuzzy neural networks are connectionist systems that facilitate learning from data, reasoning over fuzzy rules, rule insertion, rule extraction, and rule adaptation. The concept evolving fuzzy neural networks (EFuNNs), with respective algorithms for learning, aggregation, rule insertion, rule extraction, is further developed here and applied for on-line knowledge discovery on both prediction and classification tasks. EFuNNs operate in an on-line mode and learn incrementally through locally tuned elements. They grow as data arrive, and regularly shrink through pruning of nodes, or through node aggregation. The aggregation procedure is functionally equivalent to knowledge abstraction. The features of EFuNNs are illustrated on two real-world application problems—one from macroeconomics and another from Bioinformatics. EFuNNs are suitable for fast learning of on-line incoming data (e.g., financial and economic time series, biological process control), adaptive learning of speech and video data, incremental learning and knowledge discovery from growing databases (e.g. in Bioinformatics), on-line tracing of processes over time, life-long learning. The paper includes also a short review of the most common types of rules used in the knowledge-based neural networks for knowledge discovery and data mining.

Key words: fuzzy rules; evolving fuzzy neural networks; on-line learning; macroeconomics, bioinformatics.

1. Introduction—Data Mining and Knowledge Processing with the Use of Hybrid Connectionist-based Techniques

Intelligent information systems (IS) for many real-world complex problems should meet some requirements as listed below [28]:

(1) learn fast from a large amount of data, e.g. through one-pass training;

(2) adapt in an on-line mode where new data is incrementally accommodated;

(3) have an 'open' structure where new features (relevant to the task) can be introduced at any stage of the system's operation, e.g., the system creates "on the fly" new inputs, new outputs, new modules and connections;

(4) memorize data exemplars for a further refinement, or for information retrieval;

(5) learn and improve through active interaction with other IS and with the environment in a multi-modular, hierarchical fashion;

(6) adequately represent space and time in their different scales; have parameters that represent short-term and long-term memory, age, forgetting, etc.;

(7) deal with knowledge in its different forms (e.g., rules; probabilities); analyze itself in terms of behavior, global error and success; "explain" what the system has learned and what it "knows" about the problem it is trained to solve; make decisions for a further improvement.

Some of the above seven issues have already been addressed in different connectionist and fuzzy connectionist systems. Such systems can successfully perform incremental learning [7–15], on-line learning [9,13,20]; can deal with rules [3,5,17,32,34,40,49,50,55]. The latter class of neural networks (NN) are also called knowledge-based neural networks (KBNN).

On-line learning is concerned with learning data as the system operates (usually in a real time) and data might exist only for a short time. NN models for on-line learning are introduced and studied in [1,9,13,20,34,55]. Several investigations proved that the most popular neural network models and algorithms are not suitable for adaptive, on-line learning, that include: multi-layer perceptrons trained with the backpropagation algorithm, radial basis function networks [13], self-organising maps SOMs [35,36], and fuzzy neural networks [22,40]. These models usually operate on a fixed size connectionist structure, that limits its ability to accommodating new data; they may require both new data and the previously used ones in order to adjust to the new data; they may require many iterations of passing data through the connectionist structure in order to learn it, which could be unacceptably time-consuming; they may be based on a global optimization algorithm, i.e. during the learning of each data item all the elements of the connectionist structure need to be adjusted. Problems such as choosing the optimal initial structure, or arriving at a local minimum, or catastrophic forgetting, or lack of meaningful explanation of the stored in the connections information, and others, are often experienced (see [4]).

KBNN are pre-structured neural networks to allow for data and knowledge manipulation, including learning from data, rule insertion, rule extraction, adaptation and reasoning. KBNN have been developed either as a combination of symbolic AI systems and NN [3, 25,49,50], or as a combination of fuzzy logic systems [56] and NN [18,22–32,40,55], or as other hybrid systems [24,40,55]. Rule insertion and rule extraction operations are typical operations for a KBNN to accommodate existing knowledge along with data, and to produce an explanation on what the system has learned.

Many of the existing KBNN can capture rules but can not operate in an on-line mode. They are usually trained in a batch, off-line mode, and than rules are extracted. A representative of this class of KBNN is called fuzzy neural network FuNN [24–27,32]. FuNNs are NN which structure can be interpreted as a set of fuzzy rules. Evolving fuzzy neural networks (EFuNNs) in contrast are fuzzy systems that can learn fuzzy rules in a connectionist way still preserving the flexibility for adding new rules as the system operates. EFuNNs have the advantages of the traditional NNs, KBNNs, and instance-based learning systems, but in addition they can operate in an on-line mode. The move from FuNNs to EFuNNs is a move from global optimization to local element tuning, from multiple-pass learning to one-pass learning, from a fixed connectionist structure to a fluctuating one, allowing for growing through insertion of nodes, and shrinking through node aggregation or pruning. This is also a move from one type of rules, that is dealt with in the FuNN structure, to another type, that is dealt with in the EFuNN structure, from the fuzzy neural network principles, where a NN structure is associated with linguistically meaningful fuzzy concepts, to neuro-fuzzy systems, where a connectionist-type learning mechanism is associated with a set of fuzzy rules.

Here the EFuNN applicability for on-line rule extraction and knowledge discovery from dynamically changing data streams is illustrated on two real world problems—a prediction problem from macroeconomics, and a classification problem from bioinformatics.

2. Rule Extraction from KBNN—Different Types of Rules

Different KBNNs are designed to represent different types of rules, some of them listed below [24]:

(1) Simple propositional rules (e.g., IF x1 is A AND/OR x2 is B THEN y is C, where A,B and C are constants, variables, or symbols of true/false type);

(2) Propositional rules with certainty factors (e.g., IF x1 is A (CF1) AND x2 is B (CF2) THEN y is C (CFc));

(3) Zadeh-Mamdani fuzzy rules [56] (e.g., IF x1 is A AND x2 is B THEN y is C, where A,B and C are fuzzy values represented by their membership functions);

(4) Takagi-Sugeno fuzzy rules [22,40] (e.g., IF x1 is A AND x2 is B THEN y is a.x1 + b.x2 +c, where A,B and C are fuzzy values and a, b and c are constants);

(5) Fuzzy rules of type (3) with degrees of importance and certainty degrees [18,24,32] (e.g., IF x1 is A (DI1) AND x2 is B (DI2) THEN y is C (CFc), where DI1 and DI2 represent the importance of each of the condition elements for the rule output, and the CFc represents the strength of this rule)

(6) Fuzzy rules that represent associations of clusters of data from the problem space (e.g., Rule j: IF [an input vector x is in the input cluster defined by its center (x1 is Aj, to a membership degree of MD1j, AND x2 is Bj, to a membership degree of MD2j) and by its radius Rj-in] THEN [the output vector y is in the output cluster defined by its center (y is C, to a membership degree of MDc) and by its radius Rj-out, with Nex(j) examples represented by this rule] (see [5,9,34,36]).

(7) Temporal rules [30] (e.g., IF x1 is present at a time moment t1 (with a certainty degree and/or importance factor of DI1) AND x2 is present at a time moment t2 (with a certainty degree/importance factor DI2) THEN y is C (CFc))

(8) Temporal, recurrent rules (e.g., IF x1 is A (DI1) AND x2 is B (DI2) AND y at the time moment (t–k) is C THEN y at a time moment (t+n) is D (CFc))

FuNNs deal with rules of type (5), and EFuNNs deal with rules of type (6), (7) and (8). Here only EFuNN rules of type (6) are discussed. In this type of rules fuzzy membership functions are used for both expressing linguistic meaning and geometrical position of the input and the output variables respectively in input and the output spaces and also their meaningful association that is in the content of the incoming data.

There are several methods for rule extraction from a KBNN. Three of them are explained below:

(1) Rule extraction through activating a trained KBNN on input data and observing the patterns of activation (the short-term memory). The method is not practical for on-line, incremental learning in IS as past data may not be available for a consecutive activation of the trained KBNN. This method is widely used in brain-research (e.g. analyzing MRI, fMRI and EEG patterns and signals to detect rules of behavior [2,4]).

(2) Rule extraction through analysis of the connections in a trained KBNN [3,18,24,40,55] (the long-term memory). This approach allows for extracting knowledge without necessarily activating the connectionist system again on input data. It is appropriate for on-line learning and system improvement. This is the case in the rule extraction procedures of FuNNs and EFuNNs. This approach is not used in brain study research as there are no methods known so far for processing information stored in neuronal synapses.

(3) Combined methods of (1) and (2).

In terms of learning modes in a KBNN, we can differentiate the following cases: (1) off-line learning and rule extraction—first learning is performed and then rules are extracted which is one-off process; an example for such systems is FuNN; (2) on-line learning and rule extraction—rules can be extracted as part of the continuous on-line learning process; an example is EFuNN.

On-line learning and local optimization in a KBNN would allow for tracing the process of knowledge emergence, for analyzing how rules change over time. That is illustrated in the paper on two real-world problems—prediction of macroeconomic growth, and gene-expression based classification of Leukemia.

3. Evolving Fuzzy Neural Networks EFuNNs: On-line Learning and Rule Extraction

3.1. The architecture and the principles of EfuNNs

EFuNNs are introduced in [28–30] where preliminary results were given. Here EFuNNs are further developed, analyzed and applied for knowledge discovery in real problems. An EFuNN has a five-layer structure (Figure 1) where nodes and connections are created/connected as data examples are presented. An optional short-term memory layer can be used through a feedback connection from the rule (also called, case) node layer. The layer of feedback connections could be used if temporal relationships between input data are to be memorized structurally [30]—Figure 1.

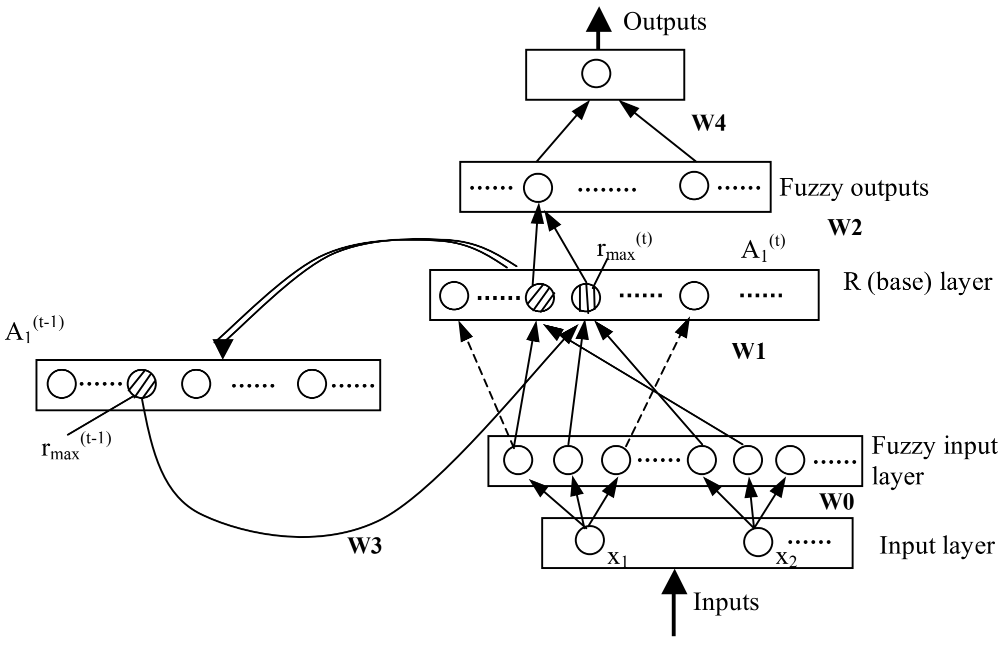


Figure 1. An exemplar EFuNN structure.

The rule layer, the condition-to-rule connection layer and the rule-to-action connection layer in EFuNNs have the meaning expressed in rules of type (6) (from section2). The third layer of neurons in EFuNN evolves through hybrid supervised/unsupervised learning. Each rule node r is defined by two vectors of connection weights—W1(r) and W2(r), the latter being adjusted through supervised learning based on the output error, and the former being adjusted through unsupervised learning based on similarity measure in the input space. The fourth layer of neurons represents fuzzy quantisation for the output variables, similar to the input fuzzy neurons representation. The fifth layer represents the real values for the output variables.

Each rule node, e.g. r_j, represents an association between a hyper-sphere from the fuzzy input space and a hyper-sphere from the fuzzy output space, the $W1(r_j)$ connection weights representing the co-ordinates of the center of the sphere in the fuzzy input space, and the W2 (r_j)—the co-ordinates in the fuzzy output space. The radius of the input hyper-sphere of a rule node r_j is defined as $R_j=1-S_j$, where S_j is the sensitivity threshold parameter defining the minimum activation of the rule node r_j to a new input vector **x** from a new example (**x**,**y**) in order the example to be considered for association with this rule node. The pair of fuzzy input-output data vectors ($\mathbf{x_f}$,$\mathbf{y_f}$) will be allocated to the rule node r_j if $\mathbf{x_f}$ falls into the r_j input receptive field (hyper-sphere), and $\mathbf{y_f}$ falls in the r_j output reactive field hyper-sphere. This is ensured through two conditions, that a local normalized fuzzy difference between $\mathbf{x_f}$ and $W1(r_j)$ is smaller than the radius r_j, and the normalized output error Err= ||**y** – **y**□|| / Nout is smaller than an error threshold E, Nout is the number of the outputs in the EFuNN and **y**□ is the output vector produced by the EFuNN. The error threshold parameter E sets the error tolerance of the system. It also defines the radius of the output cluster for each rule node.

Definition. A *local normalised fuzzy distance* between two fuzzy membership vectors $\mathbf{d_{1f}}$ and $\mathbf{d_{2f}}$ that represent the membership degrees to which two real data vectors $\mathbf{d_1}$ and $\mathbf{d_2}$ belong to pre-defined MFs, is calculated as:

$$D(\mathbf{d_{1f}},\mathbf{d_{2f}}) = ||\mathbf{d_{1f}} - \mathbf{d_{2f}}|| \,/\, ||\mathbf{d_{1f}} + \mathbf{d_{2f}}||, \tag{1}$$

where: ||**p** – **q**|| denotes the sum of all the absolute values of a vector that is obtained after vector subtraction (or summation in case of ||**p** + **q**||) of two vectors **p** and **q**; "/" denotes division. For example, if $\mathbf{d_{1f}}$=(0,0,1,0,0,0) and $\mathbf{d_{2f}}$=(0,1,0,0,0,0), than $D(d_1,d_2) = (1+1)/2=1$ which is the maximum value for the local normalized fuzzy difference when uniformly distributed triangular membership functions are used. In EFuNNs the local normalized fuzzy distance is used to measure the distance between a new input data vector and a rule node in the local vicinity of this rule node. In RBF networks Gaussian radial basis functions are allocated to the nodes and used as activation functions to calculate the distance between the node and the input vector across the whole input space.

Through the process of associating new data points to a rule node r_j, the center of this node and its radius adjust in the fuzzy input space depending on the distance between the new input vectors and the current rule node position, and on a learning rate l_j, and in the fuzzy output space depending on the output error through the Widrow-Hoff LMS algorithm (delta algorithm). This adjustment can be represented mathematically by the change of the connection weights of the rule node r_j from $W1(r_j^{(t)})$ and $W2(r_j^{(t)})$ to $W1(r_j^{(t+1)})$ and $W2(r_j^{(t+1)})$ respectively according to the following vector operations:

$$\begin{aligned} &W1(r_j^{(t+1)})=W1(r_j^{(t)})+l_j.(W1\,(r_j^{(t)}) - \mathbf{x}_f) \\ &W2\,(r_j^{(t+1)}) = W2(r_j^{(t)}) + l_j.\,(A2 - \mathbf{y}_f).\,A1(r_j^{(t)}) \end{aligned} \tag{2}$$

where: $A2=f_2(W2.A1)$ is the activation vector of the fuzzy output neurons in the EFuNN structure when $\mathbf{x}$ is presented; $A1(r_j^{(t)}) = f_l\ (D\ (W1\ (r_j^{(t)}), \mathbf{x}_f))$ is the activation of the rule node $r_j^{(t)}$; a simple linear function can be used for f_l and f_2, e.g. $A1(r_j^{(t)}) = 1-D\ (W1\ (r_j^{(t)}), \mathbf{x}_f))$; l_j is the current learning rate of the rule node r_j calculated as $l_j = 1/\ Nex(r_j)$, where $Nex(r_j)$ is the number of examples currently associated with rule node r_j. The statistical rationale behind this is that the more examples are currently associated with a rule node the less it will "move" when a new example has to be accommodated by this rule node. When a new example is associated with a rule node r_j not only its location in the input space, but also its receptive field expressed as its radius Rj, and its sensitivity threshold Sj, change as follows:

$$Rj^{(t+1)} = Rj^{(t)}+D\ (W1(r_j^{(t+1)}),W1(r_j^{(t)})), \tag{3}$$
respectively: $Sj^{\ (t+1)} = Sj^{\ (t)} - D\ (W1\ (r_j^{(t+1)}),\ W1\ (r_j^{(t)}))$

While the connection weights W1 and W2 capture fuzzy co-ordinates of the learned prototypes (exemplars) represented as centres of hyper-spheres, the temporal layer of connection weights W3 from Figure 1 captures temporal dependencies between consecutive data examples. If the winning rule node at the moment (t–1) (to which the input data vector at the moment (t–1) was associated) was $r_{max}^{\ (t-1)}$, and the winning node at the moment t is $r_{max}^{(t)}$, then a link between the two nodes is established as follows:

$$W3(r_{max}^{(t-1)},r_{max}^{(t)})^{new} = W3(r_{max}^{(t-1)},r_{max}^{(t)})^{old} + l_3.\ A1(r_{max}^{(t-1)})\ A1(r_{max}^{(t)})) \tag{4}$$

where: $A1(r^{(t)})$ denotes the activation of a rule node r at a time moment (t); l_3 defines the degree to which the EFuNN associates links between rule nodes (clusters, prototypes) that include consecutive data examples. If l_3=0, no temporal associations are learned.

The EFuNN system was explained so far with the use of one rule node activation (the winning rule node for the current input data). The same formulas as above are applicable when activation of m rule nodes (m> 1) is propagated and used (the so called "many-of-n" mode, or "m-of-n" for short). Usually m=3.

The supervised learning in EFuNN is based on the above explained principles, so when a new data example $\mathbf{d}=(\mathbf{x},\mathbf{y})$ is presented, the EFuNN either creates a new rule node r_n to memorize the two input and output fuzzy vectors $W1(r_n)= \mathbf{x}_f$ and $W2(r_n)= \mathbf{y}_f$, or the EFuNN adjusts the position and the radius of an existing rule node r_j to accommodate this example. After a certain time (when certain number of examples have been presented) some neurons and connections may be pruned or aggregated. Aggregation techniques are explained in a later section of the paper.

Different pruning rules can be applied for a successful pruning of unnecessary nodes and connections. One of them is given below:

IF ($Age(r_j)$ > OLD) AND (the total activation $TA(r_j)$ is less than a pruning parameter Pr times Age (r_j)) THEN prune rule node r_j,
where $Age(r_j)$ is calculated as the number of examples that have been presented to the EFuNN after r_j had been fist created; OLD is a pre-defined age limit; Pr is a pruning parameter in the range [0,1], and the total activation $TA(r_j)$ is calculated as the number of examples for which r_j has been a correct winning node (or among the m winning nodes in the m-of-n mode of operation).

The pruning rule and the way the values for the pruning parameters are defined, depend on the application task.

3.2. Local versus global generalization error in EFuNN's on-line learning

Modeling, tracing and predicting series of continuously incoming data (with possibly changing dynamics) is an extremely difficult task. It can only be attempted with the use of on-line learning methods and the application of locally optimized structures as it is the case of EFuNNs.

For the EFuNN on-line learning mode, where EFuNN is adjusted incrementally to each example from the data stream, the generalization error on the next new input vector (for which the output vector is not known) is called *local generalization error.* The local generalization error at the moment t, for example, when the input vector is Xdt, and the calculated by the evolved EFuNN output vector is Ydt□, is expressed as Err_t. The cumulative local generalization error can be estimated as:

$$TErr_t = \text{sum}\ \{Err_t\}_{,\ t=1,2,\ldots i} \qquad (5)$$

In contrast to the global generalization error, which can also be calculated for EFuNNs, here the error Err_t is calculated after the EFuNN has learned the previous example (Xd(t–1), Yd(t–1)). Each example is propagated only once through the EFuNN, both for testing the error and for learning (after the output vector becomes known). The root mean square error can be calculated at each data point Di from the input data stream as:

$$RMSE(i) = \text{sqrt}\ (\text{sum}\{Err_t\}_{t=1,2,..,i}) / i\ ), \qquad (6)$$

where: $Err_t= (d_t - o_t)^2$, d_t is the desired output value and o_t is the EFuNN output value produced for the t_{th} input vector Dt. The non-dimensional error index NDEI(i) can also be calculated as:

$$NDEI(i)= RMSE\ (i)\ /\text{std}\ (D(1:i)), \qquad (7)$$

where std (D_1 : Di)) is the standard deviation of the data points from D_1 to Di.

After an EFuNN is evolved on some examples from the problem space, its g*lobal generalization error* can be evaluated on a set of p future examples from the problem space as follows:

$$GErr= \text{sum}\ \{Err_i\}_{i=1,2,\ldots p}, \qquad (8)$$

where: Err_i is the error for a vector x_i from the input space X, which vector has not been and will not be used for training the EFuNN before the value GErr is calculated. After having evolved an EFuNN on a small, but representative part of the whole problem space, its global generalization error may become sufficiently small.

When issues such as universality of the EFuNN mechanism, learning accuracy, generalization and convergence for different tasks are discussed, two cases must be distinguished:

(A) The incoming data is from a compact and bounded data space. In this case the more data vectors are used for evolving an EFuNN, the better its global generalization is on the whole problem space (or on an extraction of it).

(B) Open problem space, where the data dynamics and data probability distribution change over time in a continuous way. Here, only local generalization error can be evaluated.

3.3 Fuzzy rule insertion, on-line rule adaptation, and rule extraction in EFuNNs

EFuNNs are adaptive fuzzy rule-based systems. Manipulating rules is essential for their operation. This includes rule insertion, rule extraction, and rule adaptation.

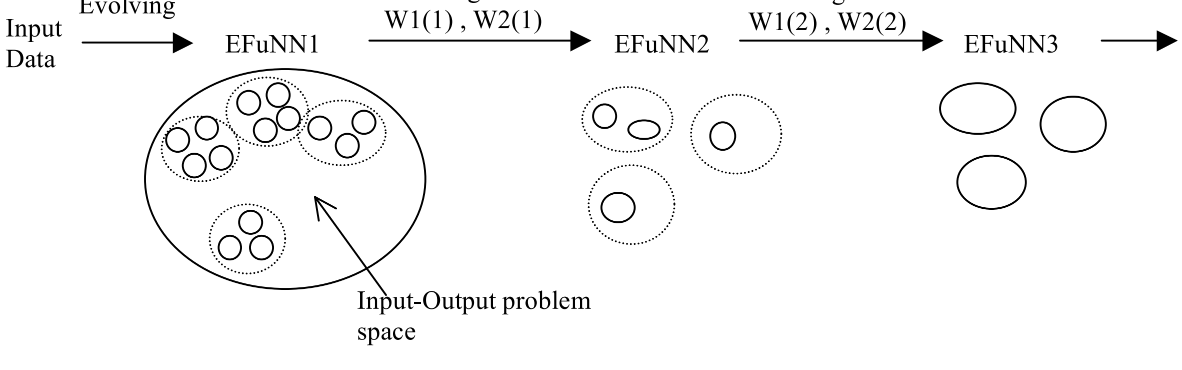
At any time (phase) of the evolving (learning) process, fuzzy or exact rules can be *inserted* and extracted from an EFuNN structure. Insertion of fuzzy rules is achieved through setting a new rule node r_j for each new rule, such that the connection weights W1(r_j) and W2(r_j) of the rule node represent this rule. For example, the fuzzy rule *(IF x_1 is Small and x_2 is Small THEN y is Small*) can be inserted into an EFuNN structure by setting the connections of a new rule node to the fuzzy condition nodes x_1-Small and x_2-Small and to the fuzzy output node y-Small to a value of 1 each. The rest of the connections are set to a value of zero. Similarly, an exact rule, e.g. *IF x_1 is 3.4 and x_2 is 6.7 THEN y is 9.5,* can be inserted into an EFuNN structure. Here the membership degrees to which the input values x_1 = 3.4 and x_2=6.7, and the output value *y=9.5* belong to the corresponding fuzzy values are calculated and attached to the corresponding connection weights.

Rule extraction and *rule aggregation* are important operations as EFuNN is a knowledge-based connectionist model. Each rule node r_j can be expressed as a fuzzy rule, for example:

> r_j: IF x_1 is Small 0.85 and x_1 is Medium 0.15 and x_2 is Small 0.7 and x_2 is Medium 0.3 (Radius of the receptive field Rj=0.1, maxRadiusj=0.2)
> THEN y is Small 0.2 and y is Large 0.8 (Radius of the reactive field E) [number of examples associated with this rule 20 out of 175],

where the numbers attached to the fuzzy labels denote the degrees to which the centers of the input and the output hyper-spheres belong to the respective MF. The degrees associated to the condition elements are the connection weights from the matrix W1. Only values that are greater than a threshold T1 are shown in the rules. The degrees associated with the conclusion part are the connection weights from W2 that are greater than a threshold T2. The other parameters associated with the rule represent the importance and the strength of the rule. An example of rules extracted from a bench-mark dynamic time series data is given in the next sub-section. The two thresholds T1 and T2 are used to disregard the connections from W1 and W2 that represent small and insignificant membership degrees (e.g., less than 0.1).

Another knowledge-based technique applied to EFuNNs is *rule node aggregation.* Through this technique several rule nodes that are close to each other in the problem space are merged into one at different time moments of the evolving process. The idea is illustrated in Figure 2.



Input Data
Evolving
EFuNN1
Evolving from W1(1) , W2(1)
EFuNN2
Evolving from W1(2) , W2(2)
EFuNN3
Input-Output problem space

Figure 2. The process of EFuNN aggregation and structure optimisation can be viewed as a process of knowledge abstraction, i.e., associated clusters in the problem's space emerge from the structure.

The aggregation procedure is illustrated below on a simple case of three rule nodes r_1, r_2, and r_3. Either of the following two aggregation strategies can be used to calculate the W1 connections of a new aggregated rule node r_{agg} (the same formulas are used to calculate the W2 connections)

- as a geometrical center of the three nodes:

$$W1(r_{agg})=(W1(r_1)+W1(r_2)+W1(r_3))/3 \tag{9}$$

- as a weighted statistical center:

$$W2(r_{agg})=(W2(r_1)\ Nex(r_1)+W2(r_2)\ Nex(r_2)+W2(r_3)\ Nex(r_3))/Nsum \tag{10}$$

where: $Nex(r_{agg}) = Nsum = Nex(r_1)+Nex(r_2)+Nex(r_3)$; $Rr_{agg} = d(W1(r_{agg}),W1(r_j)) + Rj <= Rmax$, where r_j is the rule node among the three nodes that has a maximum distance from the new node r_{agg} and Rj is its current radius of the receptive field. The three rule nodes will aggregate only if the radius of the aggregated node is less than a pre-defined maximum radius Rmax .

In order for a given node r_j to chose which other nodes it should aggregate with, two subsets of nodes are formed – the subset of nodes Nj-pos = $\{r_k\}$ that if activated to a degree of 1 will produce an output value $y'(r_k)$ that is different from $y'(r_j)$ in less than the error threshold E, and the subset of nodes Nj-neg = $\{r_p\}$ so that nodes r_p cause output values to be different from $y'(r_j)$ and the difference is higher than E. Rule nodes r_k from the first subset that are closer to r_j in the input space than the closest to r_j node from the second subset $\{r_p\}$ in terms of W1 distance, are aggregated if the radius of the new node r_{agg} is less than the pre-defined limit Rmax for a receptive field.

Instead of aggregating all the rule nodes from Nj-pos that are closer to the rule node r_j than the closest node from the other class Nj-pos, it is possible to keep the closest node from this aggregation pool to the other class as a separate node—a "guard", thus preventing mis-classification between the two classes in the bordering area.

Through node creation and their consecutive aggregation, an EFuNN system can adjust over time to changes in the data stream and at the same time—preserve its generalisation capabilities.

Through analysis of the weights W3 of an evolved EFuNN, *temporal correlation* between time consecutive exemplars can be expressed in terms of rules and conditional probabilities, e.g.:

$$\text{IF } r_1^{(t-1)} \text{ THEN } r_2^{(t)}(0.3) \tag{11}$$

The meaning of the above rule is that some examples that belong to the rule (prototype) r_2 follow in time examples from the rule prototype r_1 with a relative conditional probability of 0.3.

3.4. EFuNN parameter optimization. Evolutionary approaches.

Optimisation of the EFuNN parameters, such as number of membership functions, sensitivity and error thresholds, maximum radius for the rule nodes, etc. in an on-line mode, with a goal of having their optimal values at each time of the functioning of the system according to a given set criteria, is a challenging task. Here three approaches have been used [29]: (a) a statistically-based approach—statistical parameters are allocated to the rule nodes and their values are used for optimization purposes; (b) an evolutionary approach with the use of a genetic algorithm—many EFuNNs are evolved in a population and at certain time intervals they are evaluated and only the best ones are kept; as the system is functioning in a continuous way, the output of the best EFuNN is used at each time; (c) an evolutionary approach with the use of evolutionary strategies—the parameters are subject to change at each time with the use of mutation operations.

For the purpose of the optimisation the paradigm of evolving systems as described above can be merged with the paradigm of evolutionary systems.

4. EFuNNs for Knowledge Discovery from Dynamic Macroeconomic Data—a Case Study

In this section it is shown how the evolving connectionist techniques can be applied for the purpose of tracing the development of complex clusters of data and tracing changes in the rules that describe these data. Macroeconomic data are used as a case study.

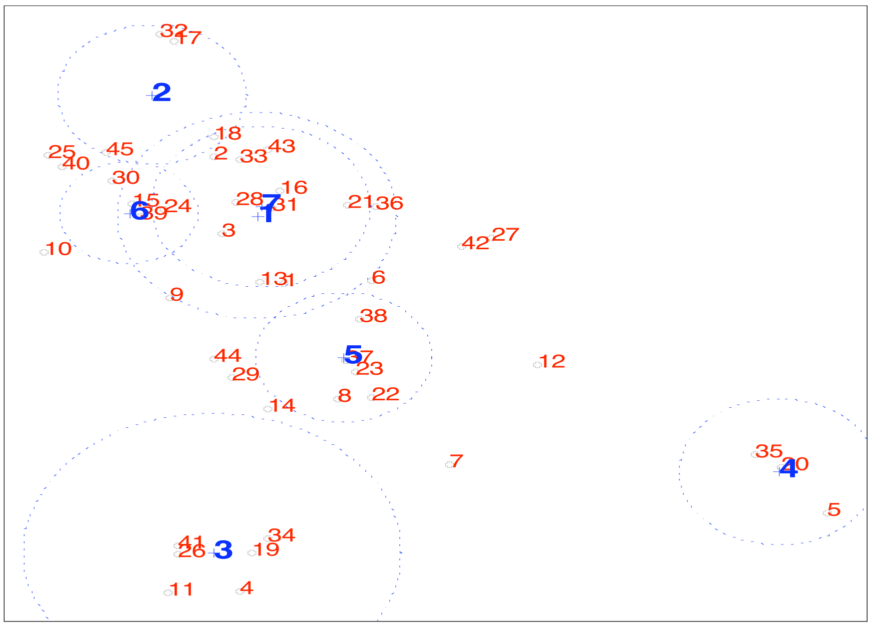
Large amounts of macroeconomic data about annual or a quarterly development of countries can be collected from many diverse sources such as EUROSTAT, Datastream, IMF, World Bank, OECD, statistics departments, central banks of regions and countries. The problem is how to analyse all this information in an efficient way, how to extract the knowledge from it and make adequate predictions for the future. For the case study here the annual macroeconomic situation of EU countries and USA are represented by four annual macroeconomic indicators: (1) consumer product index (CPI) that measures the inflation rate, (2) interest rate, (3) unemployment rate, and (4) GDP per capita in US dollars (see Appendix 1). An incremental EFuNN-based prediction model for the EU/USA countries is created and rules are extracted at different times. This makes it possible to analyse how the macroeconomic clusters are evolving and changing.

The initial EFuNN model was trained on the years 1994–98 and tested for prediction on the year 1999 data for the GDP per capita (in US dollars). The global mean square error of the model on the already used examples is very small—less than 1% of the average GDP value. The global test error for the year 1999 is 1,896$ in absolute value, which is about 8% of the average GDP per capita for the 15 countries for 1999 (23,600US$). The EFuNN system was evolved with an error threshold of 0.1, which means that no more than 10% error on each example presented to the system is tolerated. Seven clusters of countries are evolved—Figure 3(a).

The same model is further evolved on the 1999 data. The change in the clusters is shown in Figure 3(b). The clusters are captured in the rule nodes in their incoming connections. The training global root mean square error is again less than 1% of the average GDP value. Six clusters of countries are obtained now. Clusters 1 and 7 from Figure 3(a) are aggregated automatically into cluster 1 with changed parameters—geometrical center, receptive field, number of examples accommodated. This is also shown in the rules extracted from the EFuNN and explained in the next section.

The graphs in Figure 3(a) and (b) show the data and the rule nodes in chosen two dimensional input sub-spaces. The circles around the nodes represent their receptive fields. The receptive field defines the area from the input space that is “covered” by this rule node (the corresponding rule). When the EFuNN system receives a new data for learning and it falls in the area of this field the example will be accommodated by this rule node and the center of the node may change in the space – the node (the rule) adjusts). If a new example does not belong to any of the receptive fields of the existing rule nodes, a new rule node will be created to capture this example. And this process is repeated for every new example in a continuous, “life-long” way.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
BE45	DK45	DE45	EL45	ES45	FR45	IR45	IT45	NL45	AS45	IT45	IR45	SW45	UK45	US45	BE56	DK56	DE56
19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
EL56	ES56	FR56	IR56	IT56	NL56	AS56	PT56	FI56	SW56	UK56	US56	BE67	DK67	DE67	EL67	ES67	FR67
37	38	39	40	41	42	43	44	45									
IR67	IT67	NL67	AS67	PT67	FI67	SW67	UK67	US67									



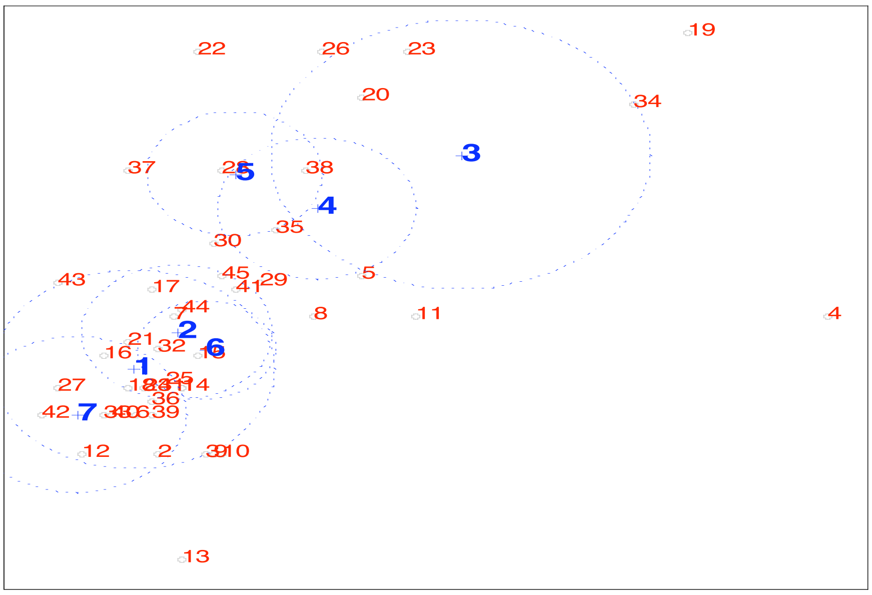


Figure 3. (a) Tracing the evolving macroeconomic clusters in EU/US. The figure shows the evolved clusters in the EFuNN predicting model for the GDP(t+1) when data from 1994 till 1998 are fed. The following parameter values are used: MF=2, Errthr=0.05; no pruning; maxRadius=0.15; normalisation is used; one-out-of-m method; fuzzy normalised distance is used; thresholds for rule extraction 0.5. The upper figure shows a plot of the rule nodes, their cluster centers and receptive fields in the input space "x=Unemployment(t–1), y=GDP(t–1)". The lower figure shows the same nodes in the input space "x=CPI(t–1) and y=Interest rate(t–1)". The rule nodes are numbered in a larger font with the consecutive numbers of their evolvement. Data examples are numbered from 1 to 45 meaning the consecutive input vectors used for the evolvement of the EFuNN in the shown order. In the legend, BE45 for example means the input vector of four parameter values for Belgium for the year 1994 and 1995 as (t=1) and (t) input values to the EFuNN model.

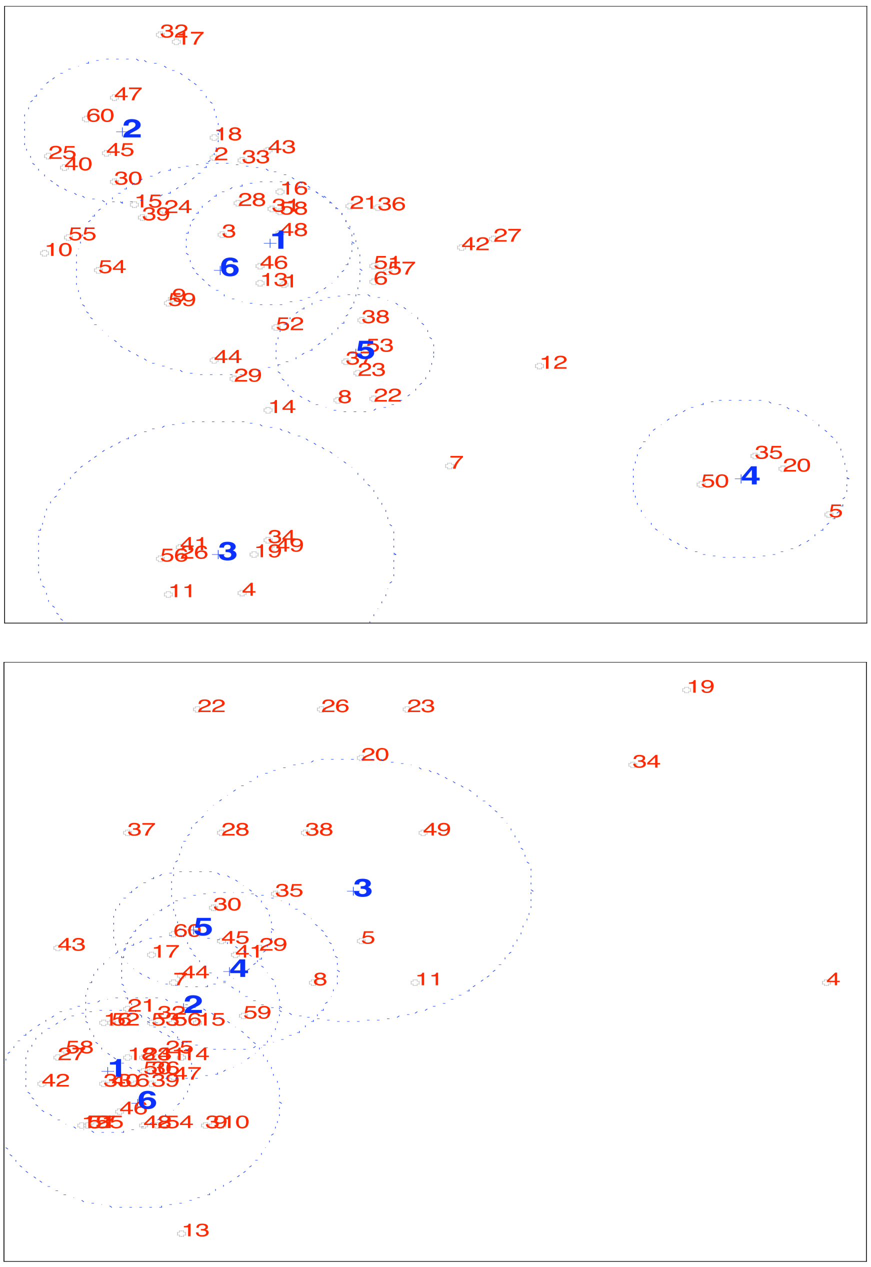


Figure 3. (b) The EFuNN model from 3a is updated on the 1999 data. The new clusters in the input space ” x=Unemployment rate(t–1), y=GDP per capita(t–1)” (upper figure), and in the input space “x=CPI(t–1) and y=Interest rate(t–1)” (lower figure) are shown. The data examples and the cluster centres (rule nodes) are represented in the same way as in Figure 3(a).

The number of clusters in the 1999 model (six) is smaller than the number of clusters of the 1998 model (seven). This illustrates the tendency that the macro-economy of whole Europe would converge and a smaller cluster would emerge. This is of course an idealized prediction as it uses only 4 parameters and does not take into account other events that may disrupt the development process.

The EFuNN model from Figure 3 can be used to predict the development of different countries in the future. Similar to the EFuNN model of the GDP per capita, three other models are produced for the rest of the macroeconomic parameters. The EFuNN models extract and save automatically rules at any time of the system operation. The extracted rules are of type (6) (see section 2) as follows:

Rule r_j: IF (macroeconomic parameters) (t) are in a defined by fuzzy MF region A, AND
(macroeconomic parameters) (t–1) are in a defined by fuzzy MF region B,
[receptive field of the rule is Rj]
THEN (macroeconomic parameters) (t+1) are in a defined by fuzzy MF region C
[the number of examples in the cluster is Nex]

Figure 4(a) shows the 7 rules extracted from the EU/USA) up to 1998 (see Figure 3(a)) for the prediction of the GDP. The following parameter values are used in the EFuNN model: MF=3; Errthtr=0.1; no forgetting; number of examples for aggregation is 15; maximum radius of a cluster is 0.2; one-out-of-n mode; normalization of data is used; normalized fuzzy local distance is measured; threshold for the rule extraction is 0.5. The rules would change with new data being fed (data for the year 1999 and further).

Figure 4(b) shows the 6 rules extracted from the EU/USA model after it has been adjusted to the 1999 data (see Figure 3(b)). When compared, the two sets of rules show similarities and differences in the macroeconomic development of the EU/US countries from year to year. The similarities represent the stable component and the differences represent the change in the rules. The number of rules (clusters) that represent "Medium" MF GDP per capita countries has decreased from 4 in the 1998 model to 3 in the 1999 model. The other number of rules (i.e. for "Large" MF GDP and for "Small" MF GDP have not changed but the number of countries accommodated in these rules has changed. The rules may further change with new data being fed (data for the year 2000 and further).

In this particular experiment the number of the rules after the system is further trained on the 1999 data is reduced from 7 to 6 (the same as the number of clusters shown in Figure 3(a) and (b)) which indicates that the countries may be getting closer in terms of the four parameters used for the experiments here.

In Figure 4(a) and (b) only the rules extracted from the GDP model are shown. Similar rules for the CPI, Interest rates, and the Unemployment rate, are extracted from the corresponding EFuNN models.

Inp var	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	Cluster	Output	Numb
	CPI(t–1)	Inter(t–1)	Unem(t–1)	GDP(t–1)	CPI(t)	Inter(t)	Unem(t)	GDP(t)	radius	GDP(t+1)	examp.
Rule#											
1	(1 0.7)	(2 0.8)	(2 0.7)	(2 0.8)	(1 0.7)	(2 0.8)	(2 0.7)	(2 0.9)	0.15	(2 0.9)	18
2	(1 0.6)	(2 0.8)	(1 0.5)	(2 0.5)	(1 0.6)	(2 0.8)	(1 0.6)	(3 0.5)	0.10	(3 0.5)	4
3	(2 0.6)	(3 0.6)	(2 0.6)	(1 0.8)	(2 0.6)	(2 0.6)	(2 0.6)	(1 0.8)	0.20	(1 0.8)	6
4	(2 0.7)	(2 0.5)	(3 0.8)	(1 0.6)	(2 0.5)	(2 0.6)	(3 0.7)	(1 0.6)	0.11	(1 0.6)	3
5	(2 0.5)	(3 0.6)	(2 0.9)	(2 0.7)	(1 0.6)	(2 0.8)	(2 0.8)	(2 0.8)	0.09	(2 0.8)	8
6	(1 0.5)	(2 0.7)	(1 0.6)	(2 0.8)	(1 0.5)	(2 0.7)	(1 0.6)	(2 0.7)	0.07	(2 0.6)	4
7	(1 0.8)	(2 0.8)	(2 0.6)	(2 0.8)	(1 0.8)	(2 0.7)	(2 0.6)	(2 0.9)	0.12	(2 0.9)	2

Figure 4. (a) Rules extracted from the EU/US model from Figure 3(a). Four parameters are used in the model: CPI; interest rate; unemployment, and GDP per capita, for the year (t) (input variables [5] till [8]) and the year (t–1) (input variables [1] till [4]). In the rules, 1,2 and 3 denote respectively the membership functions (MF) “Small”, “Medium” and “Large”, and the number next to the MF number is the membership degree (here the range of the GDP per capita is: GDPmin= 9,000US$; GDPmax=40,000US$; the membership functions of Small, Medium, and Large are triangular, uniformly distributed on the range, i.e. the center of Small is 9,000, the center of Large is 40,000 and the center of Medium is 15,500). The respective max/min values for the CPI, IntRate and Unemployment are 12/0, 12/2, and 25/1. The rules represent the clusters from Figure 3(a).

Inp var	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	Cluster	Output	Numb
	CPI(t–1)	Inter(t–1)	Unem(t–1)	GDP(t–1)	CPI(t)	Inter(t)	Unem(t)	GDP(t)	radius	GDP(t+1)	examp.
Rule#											
1	(1 0.7)	(2 0.8)	(2 0.7)	(2 0.9)	(1 0.8)	(2 0.6)	(2 0.6)	(2 0.9)	0.09	(2 0.9)	24
2	(1 0.6)	(2 0.8)	(1 0.6)	(2 0.6)	(1 0.6)	(2 0.7)	(1 0.6)	(2 0.5)	0.10	(3 0.5)	10
3	(2 0.6)	(2 0.6)	(2 0.6)	(1 0.8)	(2 0.6)	(2 0.6)	(2 0.6)	(1 0.8)	0.19	(1 0.8)	8
4	(2 0.5)	(2 0.7)	(3 0.7)	(1 0.6)	(1 0.6)	(2 0.6)	(3 0.6)	(1 0.6)	0.12	(1 0.6)	4
5	(1 0.6)	(2 0.7)	(2 0.9)	(2 0.7)	(1 0.6)	(2 0.7)	(2 0.9)	(2 0.8)	0.09	(2 0.8)	9
6	(1 0.7)	(2 0.8)	(2 0.6)	(2 0.9)	(1 0.7)	(2 0.6)	(2 0.5)	(2 0.9)	0.15	(2 1)	5

Figure 4. (b) Rules for the prediction of the GDP per capita (t+1) extracted from the EU/US model from Figure 3(b) up to the year 1999 (incl.). If compared with the rules from Figure 4(a) we can notice the stability and the plasticity of some of the rules as rules may change from year to year that could be traced in an evolving model. The rules represent the clusters from Figure 3(b).

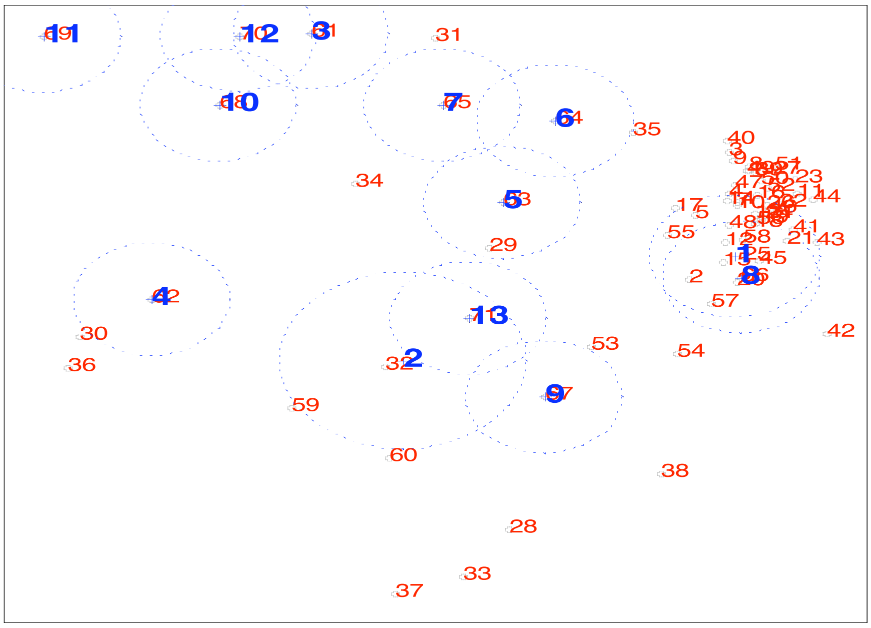
5. EFuNNs for Gene Knowledge Discovery in Bio-informatics—a Case Study

Here a data set of classification examples for Leukemia cancer disease, that consists of two classes and a large input space – the expression values of 6,817 genes monitored by Affymatrix arrays is used [12]. The initial number of examples is 72, but more examples are continuously being collected, so the classification system should be able to accommodate them and improve its performance. The two types of leukemia are acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL) as explained in Appendix 2. The task is twofold: 1) Finding a set of genes distinguishing ALL and AML, and 2) Constructing a classifier based on the expression of these genes.

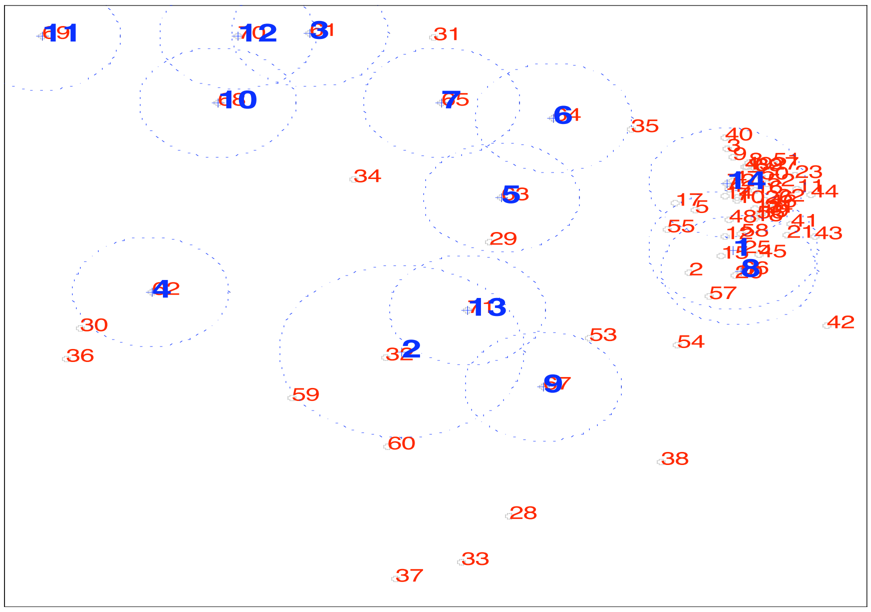
Initially, through pre-processing, the set of genes is reduced to 11 most expressed genes that have highest variability of expression across the examples from the two different classes (see Appendix 2 for the data used).

First an EFuNN model was evolved on the first 71 examples as shown. The EFuNN rule node clusters that were formed in this model are shown in Figure 5(a) in the input space x=G1, y=G2 (the original space is 11 dimensional), along with the data examples. Each example on the figure is denoted by its consecutive number in the input stream. The number of examples belong to the two classes as follows: examples 1–27 and 38–57 belong to class ALL; examples 28–37 and 58–72 belong to the AML class. The 13 evolved rule nodes are indicated in a large font. The model in its on-line learning mode is not predicting correctly the 72nd example (see Figure 6).

The EFuNN continues to learn the 72nd example and a new cluster is created as shown in Figure 5(b).



(a)



(b)

Figure 5. (a) The rule clusters with their centers (rule nodes) and receptive fields after the first 71 examples of Leukemia data are entered; (b) a new rule node is created after the 72nd examples is learned.

The process of on-line model evolving through one pass training on each consecutive example, and testing it on the next one is shown in Figure 6.

Figure 7(a) shows some of the 13 extracted rules of type (6) after the 71 examples are learned by the EFuNN. The rules are "local" and each of them "covers" a particular cluster of the input space; Figure 7(b) shows the rules after the 72nd example is learned by the system. As it can be seen from Figure 6 this example is very different from the other examples and the system learns it through creating a new rule node (a new rule respectively).

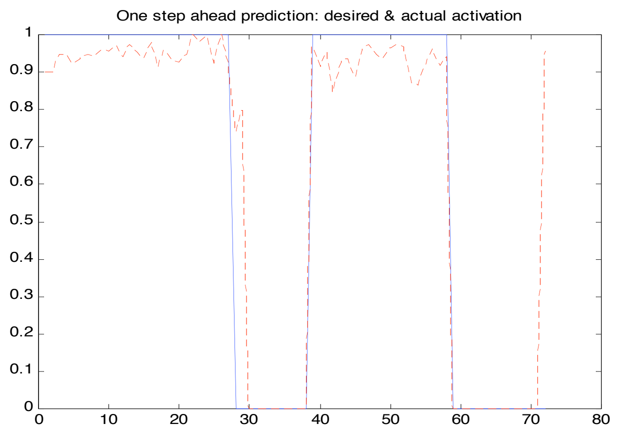


Figure 6. An EFuNN was evolved from the Leukemia two-class classification data in an on-line, incremental way, every time learning a new example and predicting the outcome for the next one. The figure shows the desired versus the predicted by the EFuNN class during the evolving process.

Input Rule#	[G1]	[G2]	[G3]	[G4]	[G5]	[G6]	[G7]	[G8]	[G9]	[G10]	[G11]	Radius	Class	Nexamp
1	(2 0.9)	(2 0.6)	(2 0.9)	(2 0.5)	(2 0.7)	(2 0.7)	(2 0.5)	(1 0.8)	(2 0.7)	(2 0.6)	(2 0.6)	0.11	(2 1.0)	47/71
2	(1 0.5)	(1 0.6)	(2 0.7)	(2 0.5)	(1 0.7)	(2 0.7)	(1 0.6)	(1 0.5)	(2 0.8)	(1 0.6)	(2 0.5)	0.16	(1 1.0)	13/71
3	(1 0.6)	(2 1.0)	(2 0.9)	(1 0.5)	(1 0.7)	(1 0.7)	(1 0.5)	(2 0.5)	(1 1.0)	(2 1.0)	(1 0.9)	0.10	(1 1.0)	1/71
4	(1 0.9)	(2 0.5)	(1 1.0)	(1 0.5)	(1 0.7)	(1 1.0)	(2 0.5)	(1 0.8)	(2 0.8)	(2 0.6)	(2 0.8)	0.10	(1 1.0)	1/71
							…							
13	(2 0.5)	(1 0.5)	(2 0.8)	(1 0.8)	(2 0.5)	(1 0.6)	(2 0.6)	(1 0.8)	(2 1.0)	(1 0.8)	(1 0.6)	0.10	(1 1.0)	1/71

(a)

Input Rule#	[G1]	[G2]	[G3]	[G4]	[G5]	[G6]	[G7]	[G8]	[G9]	[G10]	[G11]	Radius	Class	Nex
1	(2 0.9)	(2 0.6)	(2 0.9)	(2 0.5)	(2 0.7)	(2 0.7)	(2 0.5)	(1 0.8)	(2 0.7)	(2 0.6)	(2 0.6)	0.11	(2 1.0)	47/71
2	(1 0.5)	(1 0.6)	(2 0.7)	(2 0.5)	(1 0.7)	(2 0.7)	(1 0.6)	(1 0.5)	(2 0.8)	(1 0.6)	(2 0.5)	0.16	(1 1.0)	13/71
3	(1 0.6)	(2 1.0)	(2 0.9)	(1 0.5)	(1 0.7)	(1 0.7)	(1 0.5)	(2 0.5)	(1 1.0)	(2 1.0)	(1 0.9)	0.10	(1 1.0)	1/71
4	(1 0.9)	(2 0.5)	(1 1.0)	(1 0.5)	(1 0.7)	(1 1.0)	(2 0.5)	(1 0.8)	(2 0.8)	(2 0.6)	(2 0.8)	0.10	(1 1.0)	1/71
							…							
13	(2 0.5)	(1 0.5)	(2 0.8)	(1 0.8)	(2 0.5)	(1 0.6)	(2 0.6)	(1 0.8)	(2 1.0)	(1 0.8)	(1 0.6)	0.10	(1 1.0)	1/71
14	(2 0.9)	(2 0.7)	(2 0.9)	(2 0.6)	(2 0.7)	(2 0.7)	(2 0.6)	(1 0.7)	(2 0.6)	(2 0.6)	(2 0.6)	0.10	(1 1.0)	/72

(b)

Figure 7. (a) Some of the extracted rules from the trained EFuNN on the 71 Leukemia data examples. Here [g1] (2 0.9) means that the membership degree to which gene 1 expression value belongs to the membership function "High" is 0.9. Alternatively, 1 denotes the membership function "Low". There was a membership degree threshold of 0.7 used and values less than this threshold are not shown; (b) The rules extracted from the updated on the 72nd example EFuNN model. There is new rule that represents the new data example as it is significantly different from the previous ones in its class.

6. Conclusions

The evolving fuzzy neural network techniques presented here are useful techniques for modeling, and rule extraction from complex dynamic processes. The rules of development can be extracted and traced over time that may help understand the complexity and the dynamics of the processes. The EFuNN simulators used in this paper as well as the data sets are available from http://divcom.otago.ac.nz/infosci/kel/CBIIS.html.

In the EFuNN models, as well in the other modeling techniques, features have to be evaluated in advance as this may be crucial for the prediction results. Future work is planned on the on-line, dynamic evaluation of the importance of the features. This is expected to result in more precise models.

Further applications include: adaptive speech and language processing [30]; more applications in Bioinformatics; intelligent agents on the WWW [54]; financial and economic analysis and prediction; adaptive mobile robot control; adaptive process control; adaptive expert systems; adaptive artificial life systems.

Acknowledgements

This research is part of a research program funded by the New Zealand Foundation for Research Science and Technology, UOOX0016. I would like to acknowledge Dr Da Deng, Dr Haldun Akpinar and Lorenzo Rizzi who took part in the initial preparation of the macroeconomic case study data, and Matthias Futschik and Dr Vlad Bajic who took part in the preparation of the Bioinformatics case study data. Part of this research was done during my visit to the University of Trento, Italy – Department of Computer Science and Management and the Faculty of Economics. The evolving fuzzy neural network EFuNN is under a patent protection by the University of Otago, New Zealand for any commercial use.

References

1. Alpaydin, E. “GAL: networks that grow when they learn and shrink when they forget”, TR 91-032, Int.Computer Sci. Inst., Berkeley, CA (1991).
2. Amari, S. and Kasabov, N. eds, Brain-like computing and intelligent information systems, Springer Verlag (1997).
3. Andrews, R., J. Diederich, A.B.Tickle, "A Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks", Knowledge-Based Systems, 8, 373–389 (1995).
4. Arbib, M (ed.) The Handbook of Brain Theory and Neural Networks, The MIT Press (1995)
5. Berenji, H., Khedkar, P. “Learning and tuning fuzzy logic controllers through. IEEE Trans. on Neural Networks, 3, 724–740 (1992)
6. Carpenter, G. and Grossberg, S., Pattern recognition by self-organizing neural networks , The MIT Press, Cambridge, Massachusetts (1991)
7. Carpenter, G.A., Grossberg, S., Markuzon, N., Reynolds, J.H., Rosen, D.B., FuzzyARTMAP: A neural network architecture for incremental supervised learning of analog multi-dimensional maps, IEEE Transactions of Neural Networks , vol.3, No.5, 698–713, (1991)
8. DeGaris, H. , Circuits of Production Rule - GenNets – The genetic programming of nervous systems, in: Albrecht, R., Reeves, C. and N. Steele (eds) Artifical Neural Networks and Genetic Algorithms, Springer Verlag (1993)
9. Duch, W., and Diercksen, G. “Feature Space Mapping as a universal adaptive system”, Computer Physics Communication, 87 (1995) 341–371
10. Edelman, G., Neuronal Darwinism: The theory of neuronal group selection, Basic Books (1992).
11. Encarnacao, L.M., and Gross, M.H. “An adaptive classification scheme to approximate decision boundaries using local Bayes criterias – Melting Octree Networks, Rep.92-047, Int.Computer Sci. Inst., Berkeley, CA (1992).
12. Fahlman, C .,and C. Lebiere, "The Cascade-Correlation Learning Architecture", in: Turetzky, D (ed) Advances in Neural Information Processing Systems, vol.2, Morgan Kaufmann, 524–532 (1990).
13. Freeman, J.A.S., Saad, D., On-line learning in radial basis function networks, Neural Computation, vol. 9, No.7 (1997)
14. Fritzke, B. “Vector quantization with growing and splitting elastic net”, in: ICANN’93: Proc. of the Intern.Conf. on artificial neural networks, Amsterdam, (1993)
15. Fritzke, B., A growing neural gas network learns topologies, Advances in Neural Information Processing Systems, vol.7 (1995)
16. Golub, T.R., et al. Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring, *Science 286: 531-7, 1999*
17. Goodman, R.M., C.M. Higgins, J.W. Miller, P.Smyth, "Rule-based neural networks for classification and probability estimation", Neural Computation, 14, 781–804 (1992)
18. Hashiyama, T., Furuhashi, T., Uchikawa, Y. “A Decision Making Model Using a Fuzzy Neural Network”, in: Proceedings of the 2nd International Conference on Fuzzy Logic & Neural Networks, Iizuka, Japan, 1057–1060, (1992).
19. Hassibi and Stork, “Second order derivatives for network pruning: Optimal Brain Surgeon”, in: Advances in Neural Information Processing Systems, 4, 164–171, (1992).
20. Heskes, T.M., Kappen, B., “On-line learning processes in artificial neural networks”, in: Math. foundations of neural networks, Elsevier, Amsterdam, 199–233, (1993).
21. Ishikawa, M. "Structural Learning with Forgetting", Neural Networks 9, 501–521, (1996).

22. Jang, R. ANFIS: adaptive network-based fuzzy inference system, IEEE Trans. on Syst.,Man, Cybernetics, 23(3), May/June 1993, 665–685, (1993).
23. Kasabov, N. and M. Watts, "Spatio-temporal evolving fuzzy neural networks and their applications for on-line, adaptive phoneme recognition", TR 99/03, Department of Information Science, University of Otago, New Zealand
24. Kasabov, N. Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering, The MIT Press, CA, MA(1996).
25. Kasabov, N., "Adaptable connectionist production systems". Neurocomputing, 13 (2-4) 95–117 (1996).
26. Kasabov, N., "Investigating the adaptation and forgetting in fuzzy neural networks by using the method of training and zeroing", Proceedings of the International Conference on Neural Networks ICNN'96, Plenary, Panel and Special Sessions volume, 118–123 (1996).
27. Kasabov, N., "Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems", Fuzzy Sets and Systems 82 (2) 2–20 (1996).
28. Kasabov, N., "ECOS: A framework for evolving connectionist systems and the eco learning paradigm", Proc. of ICONIP'98, Kitakyushu, Oct. 1998, IOS Press, 1222–1235
29. Kasabov, N., "The ECOS framework and the ECO learning method for evolving connectionist systems, Journal of Advanced Computational Intelligence, 2(6) 195–202 (1998)
30. Kasabov, N. Adaptive learning system and method, Patent Reg.No.503882, New Zealand (2000)
31. Kasabov, N., Evolving Fuzzy Neural Networks—Algorithms, Applications and Biological Motivation, in Proc. of Iizuka'98, Iizuka, Japan, Oct.1998, World Sci., 271–274 (1998)
32. Kasabov, N., Kim J S, Watts, M., Gray, A., "FuNN/2—A Fuzzy Neural Network Architecture for Adaptive Learning and Knowledge Acquisition", Information Sciences — Applications, 101(3–4): 155–175 (1997).
33. Kater, S.B., Mattson, N.P., Cohan, C. and Connor, J., "Calcium regulation of the neuronal cone growth", Trends in Neuroscience, 11, 315–321(1988).
34. Kim, J. and Kasabov, N. "HyFIS: Adaptive hybrid connectionist fuzzy inference systems", TR 99/05, Department of Information Science, University of Otago, New Zealand
35. Kohonen, T. The Self-Organizing Map. Proceedings of the IEEE, vol.78, N-9, pp.1464–1497, (1990).
36. Kohonen, T.,. Self-Organizing Maps, second edition, Springer Verlag, 1997.
37. Kozma, R. and N.Kasabov, "Rules of chaotic behaviour extracted from the fuzzy neural network FuNN", Proc. of the WCCI'98 FUZZ-IEEE International Conference on Fuzzy Systems, Anchorage, May (1998).
38. Krogh, A. and Hertz, J.A., "A simple weight decay can improve generalisation. Advances in Neural Information Processing Systems", 4, 951–957, (1992)
39. Le Cun, Y., J.S. Denker and S.A. Solla, "Optimal Brain Damage", in: D.S. Touretzky, ed., Advances in Neural Information Processing Systems, Morgan Kaufmann, 2, 598–605 (1990).
40. Lin, C.T. and C.S. G. Lee, Neuro Fuzzy Systems, Prentice Hall (1996).
41. Miller, D.,J.Zurada and J.H. Lilly, "Pruning via Dynamic Adaptation of the Forgetting Rate in Structural Learning," Proc. IEEE ICNN'96, Vol.1, p.448 (1996).
42. Mitchell, M.T. Machine Learning, MacGraw-Hill (1997)
43. Mitchell, Melanie, An Introduction to Genetic Algorithms, MIT Press, Cambridge, Massachusetts (1996).

44. Mozer. M, and Smolensky, P., “A technique for trimming the fat from a network via relevance assessment”, in: D.Touretzky (ed) Advances in Neural Information Processing Systems, vol.2, Morgan Kaufmann, 598–605 (1989).
45. Quartz, S.R., and Sejnowski, T.J., “The neural basis of cognitive development: a constructivist manifesto”, Behavioral and Brain Science, to appear
46. Reed, R., “Pruning algorithms — a survey”, IEEE Trans. Neural Networks, 4 (5) 740–747, (1993).
47. Sankar, A. and R.J. Mammone, “Growing and Pruning Neural Tree Networks”, IEEE Trans. Comput. 42(3), 291–299, (1993).
48. Schiffman, W., Joost, M. and Werner. R., “Application of Genetic Algorithms to the Construction of Topologies for Multilayer Perceptrons”, In: Albrecht, R.F., Reeves, C. R., Steele, N. C. (Eds.), Artificial Neural Nets and Genetic Algorithms, Springer-Verlag Wien, New York (1993)
49. Sun, R. “A connectionist model for commonsense reasoning incorporating rules and similarities”, in: Knowledge Acquisitions, Academic Press, Cambridge (1992)
50. Towel, G., J. Shavlik, J. and M. Noordewier, "Refinement of approximate domain theories by knowledge-based neural networks", Proc. of the 8th National Conf. on Artificial Intelligence AAAI'90, Morgan Kaufmann, 861–866 (1990).
51. Vapnik, V. and Bottou, L. Neural Computation, 5 (1993) 893–909
52. Watts, M., and Kasabov, N. “Genetic algorithms for the design of fuzzy neural networks”, in Proc. of ICONIP'98, Kitakyushu, Oct. 1998, IOS Press, 793–795 (1998)
53. Wang, L.X., "Adaptive fuzzy systems and control, Prentice Hall, 1994
54. Woldrige, M. and Jennings, N., “Intelligent agents: Theory and practice”, The Knowledge Engineering review (10), 1995.
55. Yamakawa, T., H. Kusanagi, E. Uchino and T.Miki, "A new Effective Algorithm for Neo Fuzzy Neuron Model", in: Proceedings of Fifth IFSA World Congress, 1017–1020, (1993)
56. Zadeh, L. Fuzzy Sets, Information, and Control, vol.8, 338–353, (1965).

Appendix 1. Macroeconomic data used in the case study

Label	CPI	Int. rates	Unempl.	GDP per cap.	Label	CPI	Int. rates	Unempl.	GDP per cap.
BE94	2.4	6.6	10.0	23501.88	BE97	1.6	5.8	9.4	24336.20
DK94	2.1	5.6	8.2	29203.53	DK97	2.3	6.3	5.6	31961.49
DE94	2.7	5.6	8.4	25703.15	DE97	1.9	5.6	9.9	25780.23
EL94	10.7	7.7	8.9	9493.891	EL97	5.5	9.9	9.8	11514.43
ES94	4.7	8.3	24.1	13069.69	ES97	1.9	6.4	20.8	14393.66
FR94	1.8	6.2	12.3	23603.64	FR97	1.2	5.6	12.3	24325.34
IR94	2.3	7.7	14.3	15249.09	IR97	1.5	7.1	9.8	21535.54
IT94	4.1	7.7	11.4	18223.49	IT97	2.0	7.1	12.1	20586.60
NL94	2.8	5.6	7.1	22839.21	NL97	2.2	5.6	5.2	24130.14
AS94	2.9	5.6	3.8	24893.14	AS97	1.3	5.6	4.4	25615.78
PT94	5.4	7.7	7.0	9406.548	PT97	2.3	7.1	6.8	11041.68
FI94	1.1	5.6	16.6	19814.19	FI97	1.2	5.6	12.7	24022.38
SW94	2.4	4.0	9.4	23522.05	SW97	0.9	6.7	9.9	26786.31
UK94	2.4	6.6	9.6	17748.93	UK97	3.2	7.2	7.0	22641.23
US94	2.6	7.1	6.1	27064.55	US97	2.3	8.4	4.9	30978.79
BE95	1.4	7.1	9.9	27688.33	BE98	1.0	4.8	9.5	24981.72
DK95	2.0	8.1	7.2	34468.86	DK98	1.8	5.0	5.1	32903.07
DE95	1.7	6.6	8.2	30118.61	DE98	1.0	4.6	9.4	26232.61
EL95	8.9	12.0	9.2	11268.59	EL98	4.7	8.5	10.7	11535.17
ES95	4.7	11.0	22.9	15116.65	ES98	1.8	4.9	18.7	14995.52
FR95	1.7	7.3	11.7	27027.11	FR98	0.8	4.7	11.7	24958.29
IR95	2.6	11.7	12.3	18313.38	IR98	2.4	5.0	7.8	23025.41
IT95	5.3	11.7	11.9	19465.62	IT98	2.0	5.0	12.2	21050.44
NL95	1.9	6.6	6.9	26818.29	NL98	2.0	4.6	4.0	24925.68
AS95	2.2	6.7	3.9	29274.04	AS98	1.0	4.8	4.7	26109.71
PT95	4.2	11.7	7.3	11150.55	PT98	2.7	5.0	5.1	11669.40
FI95	0.8	6.6	15.4	25519.55	FI98	1.5	4.6	11.4	25167.72
SW95	2.9	9.9	8.8	27153.07	SW98	0.4	5.2	8.3	26818.57
UK95	3.4	8.2	8.7	19207.55	UK98	3.4	5.7	6.3	24097.07
US95	2.8	8.8	5.6	28159.58	US98	1.6	8.4	4.5	32371.24
BE96	2.1	6.6	9.7	26878.00	BE99	1.1	4.7	9.0	24760.10
DK96	2.1	7.2	6.8	34816.05	DK99	2.4	5.0	5.2	32727.21
DE96	1.4	6.2	8.9	29112.06	DE99	0.6	4.5	8.7	25782.08
EL96	8.2	10.9	9.6	11897.31	EL99	2.7	6.4	10.4	11873.06
ES96	3.6	9.0	22.2	15708.41	ES99	2.3	4.4	15.9	15368.53
FR96	2.0	6.4	12.4	26941.92	FR99	0.6	4.9	11.3	24593.61
IR96	1.7	9.9	11.6	19973.93	IR99	1.6	4.8	5.7	24529.16
IT96	4.0	9.9	12.0	21842.17	IT99	1.7	4.0	11.4	20734.37
NL96	2.0	6.2	6.3	26506.04	NL99	2.2	4.6	3.3	24987.81
AS96	1.5	6.2	4.3	28758.02	AS99	0.6	4.3	3.7	25793.41
PT96	3.1	8.1	7.3	11580.36	PT99	2.3	4.8	4.5	11823.92
FI96	0.6	6.2	14.6	25125.13	FI99	1.2	4.7	10.2	25194.63
SW96	0.8	8.2	9.6	29575.41	SW99	0.3	5.0	7.2	26869.68
UK96	2.4	7.8	8.2	20060.55	UK99	1.6	5.1	6.2	24632.55
US96	2.9	8.3	5.4	29447.22	US99	2.1	8.0	4.2	33933.58

AS	Austria	FI	Finland	ES	Spain
BE	Belgium	FR	France	PT	Portugal
DE	Germany	IR	Ireland	US	USA
DK	Denmark	IT	Italy	SW	Sweden
EL	Greece	NL	Netherlands	UK	U. Kingdom

Appendix 2. Gene expression data for two Leukemia classes used in the second case study

The original data from (Golub, T.R., *et al.* Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring, *Science 286: 531-7, 1999)* is transformed and only the first 11 genes with the highest variability across classes are shown here (G1 to G11).

G1 expr	G2 expr	G3 expr	G4 expr	G5 expr	G6 expr	G7 expr	G8 expr	G9 expr	G10 expr	G11 expr	Cl
3.95E+00	1.64E+00	9.13E–01	5.43E–01	9.72E–01	3.28E–01	3.48E–01	–1.29E+00	–2.21E–01	8.62E–01	9.66E–01	1
3.26E+00	6.43E–01	1.57E+00	7.44E–02	8.21E–01	–2.24E–01	5.45E–01	–1.59E+00	–2.97E–03	8.78E–01	8.22E–01	1
3.72E+00	1.81E+00	8.62E–01	2.58E–01	8.81E–01	6.10E–01	4.60E–02	–1.43E+00	–3.47E–01	7.79E–01	1.23E+00	1
3.73E+00	1.44E+00	1.04E+00	9.55E–01	1.07E+00	3.08E–01	4.88E–01	–1.19E+00	–2.09E–01	9.06E–01	6.05E–01	1
3.34E+00	1.23E+00	1.07E+00	4.74E–01	1.26E+00	–2.13E–01	2.80E–01	–1.41E+00	–5.33E–02	6.93E–01	2.43E–01	1
4.03E+00	1.64E+00	1.17E+00	3.81E–01	8.69E–01	1.23E–01	9.74E–01	–1.38E+00	–5.66E–01	1.20E+00	9.46E–01	1
3.83E+00	1.35E+00	1.03E+00	4.49E–02	8.57E–01	3.53E–01	5.50E–01	–1.39E+00	–4.27E–01	1.27E+00	8.34E–01	1
3.96E+00	1.69E+00	7.08E–01	4.95E–02	5.54E–01	6.77E–01	1.43E–01	–1.09E+00	9.57E–02	9.07E–01	1.07E+00	1
3.78E+00	1.74E+00	1.38E+00	7.72E–02	7.74E–01	8.31E–02	6.24E–01	–1.39E+00	–5.05E–01	1.38E+00	9.42E–01	1
3.82E+00	1.33E+00	7.80E–01	2.27E–01	1.11E+00	3.24E–01	3.18E–01	–1.59E+00	–2.47E–01	7.63E–01	1.03E+00	1
4.51E+00	1.43E+00	1.12E+00	5.09E–01	9.88E–01	6.36E–03	3.43E–01	–1.67E+00	–5.87E–01	8.64E–01	1.08E+00	1
3.68E+00	9.93E–01	1.30E+00	2.74E–01	9.75E–01	–3.42E–01	3.54E–01	–1.27E+00	–3.91E–01	7.03E–01	5.92E–01	1
4.03E+00	1.25E+00	9.97E–01	5.35E–01	1.38E+00	–4.44E–03	6.43E–01	–1.56E+00	–3.76E–01	7.52E–01	7.25E–01	1
3.70E+00	1.37E+00	1.06E+00	3.37E–01	9.47E–01	1.85E–01	7.45E–01	–1.13E+00	–5.52E–01	1.32E+00	5.87E–01	1
3.65E+00	8.07E–01	1.12E+00	7.26E–01	1.35E+00	1.65E–01	6.68E–01	–1.59E+00	–5.89E–01	1.13E+00	3.53E–01	1
4.04E+00	1.42E+00	8.54E–01	5.55E–01	9.65E–01	3.10E–01	5.50E–01	–1.39E+00	–6.97E–01	6.57E–01	7.00E–01	1
3.10E+00	1.31E+00	4.69E–01	–1.46E–01	2.64E–01	9.80E–01	2.36E–01	–1.03E+00	–3.56E–01	6.12E–01	5.09E–01	1
4.03E+00	1.16E+00	8.33E–01	6.70E–01	1.27E+00	1.76E–01	4.40E–01	–1.50E+00	–1.18E–01	9.85E–01	7.85E–01	1
4.10E+00	1.20E+00	3.68E–01	1.68E–02	8.96E–01	4.67E–01	4.18E–01	–1.24E+00	–2.30E–01	5.34E–01	7.34E–01	1
3.81E+00	6.24E–01	6.18E–01	4.14E–01	1.08E+00	3.47E–01	1.28E–01	–1.52E+00	–4.78E–01	7.30E–01	2.26E–01	1
4.40E+00	1.01E+00	1.01E+00	5.24E–01	1.30E+00	–2.40E–01	5.17E–01	–1.33E+00	–4.30E–01	1.14E+00	3.85E–01	1
4.29E+00	1.35E+00	8.88E–01	5.09E–01	1.20E+00	1.71E–01	4.17E–01	–1.50E+00	–2.98E–01	1.02E+00	9.75E–01	1
4.49E+00	1.56E+00	1.08E+00	2.28E–01	9.28E–01	1.17E–03	6.51E–01	–1.41E+00	–5.20E–01	1.02E+00	8.85E–01	1
4.14E+00	1.23E+00	9.06E–01	6.12E–01	1.01E+00	3.19E–01	4.55E–01	–1.51E+00	–2.58E–01	1.05E+00	8.27E–01	1
3.89E+00	8.62E–01	1.03E+00	3.43E–01	6.79E–01	3.11E–02	4.22E–01	–7.91E–01	–4.36E–01	3.20E–01	1.63E+00	1
4.15E+00	1.33E+00	8.77E–01	4.88E–01	1.11E+00	1.66E–01	5.57E–01	–1.47E+00	–4.65E–01	9.08E–01	8.36E–01	1
4.22E+00	1.64E+00	9.91E–01	4.60E–01	5.45E–01	5.76E–01	2.49E–01	–1.56E+00	–3.05E–02	1.40E+00	1.24E+00	1
1.19E+00	–1.65E+00	1.05E+00	–1.48E+00	1.16E+00	–5.62E–01	1.13E–01	–2.56E–01	2.52E–02	–1.83E–02	1.39E–16	0
9.62E–01	9.30E–01	–1.42E+00	3.82E–01	–9.53E–02	–2.61E–01	1.03E+00	–2.98E–01	–7.70E–02	–1.05E–01	8.05E–16	0
–3.77E+00	1.16E–01	–1.53E+00	2.48E–01	1.13E+00	5.27E–01	–2.06E–01	–1.14E–01	6.99E–02	7.52E–02	3.33E–16	0
3.23E–01	2.87E+00	1.27E–01	–2.27E–01	–4.31E–01	–6.81E–01	–9.66E–02	9.59E–02	2.82E–01	3.29E–02	6.11E–16	0.
–2.44E–01	–1.59E–01	1.45E+00	6.98E–01	–2.50E–01	9.42E–01	–2.15E–01	–3.53E–01	9.01E–02	–1.81E–01	0.00E+00	0
6.42E–01	–2.08E+00	8.74E–01	1.70E+00	2.50E–01	–3.32E–01	3.67E–01	3.66E–01	4.77E–02	1.02E–01	–1.80E–16	0
–6.02E–01	1.53E+00	1.99E–01	4.98E–01	3.82E–01	–7.58E–01	–6.76E–01	2.42E–01	–2.18E–01	–1.12E–01	3.89E–16	0
2.61E+00	2.00E+00	6.15E–01	–1.22E–01	1.31E–02	7.50E–01	–4.10E–02	–1.69E–01	–1.45E–01	1.97E–01	–4.16E–17	0
–3.92E+00	–1.67E–01	8.88E–01	–1.03E+00	–8.35E–01	2.13E–01	4.78E–01	3.46E–01	–8.46E–02	1.46E–02	0.00E+00	0
–1.27E–01	–2.24E+00	–8.81E–01	–8.35E–03	–1.21E+00	–4.93E–01	–5.33E–01	–4.38E–01	–2.99E–02	7.23E–02	0.00E+00	0
2.94E+00	–1.14E+00	–1.36E+00	–6.56E–01	–1.21E–01	6.55E–01	–2.23E–01	5.78E–01	4.04E–02	–7.75E–02	2.22E–16	0
4.03E+00	1.63E+00	1.12E+00	2.87E–01	9.95E–01	1.08E–01	6.39E–01	–1.43E+00	–8.04E–02	8.17E–01	1.04E+00	1
3.69E+00	1.91E+00	1.39E+00	4.94E–01	7.96E–01	1.06E–02	1.14E+00	–6.21E–01	–1.78E–01	1.40E+00	5.64E–01	1
4.45E+00	1.10E+00	1.03E+00	7.84E–01	1.10E+00	6.18E–02	5.21E–01	–1.66E+00	–5.86E–01	1.04E+00	6.20E–01	1
4.86E+00	1.43E–01	4.43E–01	–3.62E–02	9.92E–01	–3.46E–01	2.39E–01	–7.41E–01	–9.57E–01	9.21E–01	–4.70E–01	1
4.73E+00	9.84E–01	5.95E–01	1.31E–01	5.82E–01	1.21E–01	5.24E–01	–1.04E+00	–6.90E–01	9.59E–01	2.56E–01	1
4.68E+00	1.39E+00	1.18E+00	7.91E–01	1.21E+00	–1.87E–01	4.09E–01	–1.64E+00	–7.87E–01	1.22E+00	8.06E–01	1
4.07E+00	8.13E–01	1.52E+00	–1.63E–01	1.24E+00	–6.10E–01	3.45E–01	–2.25E+00	–1.81E–01	7.04E–01	2.17E–01	1

4.19E+00	1.30E+00	1.28E+00	2.73E–01	1.15E+00	–1.05E–01	6.50E–01	–1.84E+00	–4.94E–01	1.14E+00	8.26E–01	1
3.78E+00	1.51E+00	7.89E–01	2.07E–01	8.86E–01	5.85E–02	4.95E–01	–1.11E+00	–2.28E–01	7.86E–01	6.02E–01	1
3.72E+00	1.14E+00	7.83E–01	2.13E–01	1.58E+00	6.99E–02	4.11E–01	–1.26E+00	–5.62E–01	9.23E–01	6.65E–01	1
3.92E+00	1.65E+00	6.78E–01	3.35E–01	6.37E–01	5.30E–01	–4.15E–01	–1.25E+00	–1.23E–01	1.04E+00	9.56E–01	1
4.10E+00	1.55E+00	8.14E–01	5.72E–01	1.17E+00	1.91E–01	8.72E–01	–1.20E+00	–5.22E–01	7.36E–01	8.61E–01	1
4.24E+00	1.69E+00	7.89E–01	5.22E–01	8.05E–01	3.08E–01	2.59E–01	–1.35E+00	–5.22E–01	8.41E–01	9.59E–01	1
4.17E+00	1.49E+00	1.05E+00	3.66E–01	9.81E–01	1.31E–01	6.32E–01	–1.44E+00	–9.78E–01	1.10E+00	7.87E–01	1.
2.13E+00	3.37E–02	1.32E+00	7.87E–01	1.30E+00	7.13E–02	6.94E–01	–1.32E+00	–2.74E–01	3.83E–01	2.02E–01	1
3.12E+00	–3.33E–02	5.00E–01	–4.20E–01	1.42E+00	3.11E–02	8.28E–01	–1.44E+00	–1.63E–01	2.02E–01	–4.03E–01	1
3.01E+00	1.05E+00	7.60E–01	–2.02E–01	1.32E+00	1.87E–01	2.14E–01	–1.32E+00	2.60E–01	1.08E+00	6.01E–01	1
4.05E+00	1.18E+00	1.39E+00	3.32E–01	1.11E+00	–1.66E–01	5.88E–01	–1.76E+00	–1.71E–01	9.98E–01	5.53E–01	1
3.51E+00	4.19E–01	1.04E+00	–1.10E–01	1.64E+00	–2.55E–01	1.11E–01	–1.94E+00	1.17E–01	6.58E–01	7.01E–01	1
3.88E+00	1.01E+00	6.95E–01	3.67E–01	1.36E+00	6.28E–02	1.96E–01	–1.46E+00	7.91E–02	7.57E–01	6.15E–01	1
–1.32E+00	–5.40E–01	–1.03E+00	1.52E+00	1.33E–01	7.59E–02	1.64E–01	1.60E–01	–7.86E–01	–6.30E–01	–5.75E–02	0
–2.10E–01	–9.90E–01	–1.17E+00	8.53E–01	–3.36E–01	–1.92E+00	6.87E–02	–2.55E–02	3.28E–01	5.49E–01	1.18E+00	0
–1.10E+00	2.91E+00	1.18E+00	–4.71E–02	–4.33E–01	–1.81E+00	1.06E–01	4.59E–01	–3.39E+00	2.11E+00	–1.45E+00	0
–2.94E+00	4.63E–01	–3.93E+00	5.32E–02	–3.97E–01	–2.83E+00	3.75E–01	–1.47E+00	8.43E–02	7.50E–01	1.05E+00	0
1.11E+00	1.35E+00	–2.06E–01	3.71E–01	5.90E–01	–1.45E–01	1.31E+00	–4.69E–01	–2.19E+00	–3.12E–02	–5.95E–01	0
1.71E+00	2.10E+00	–1.14E+00	1.30E+00	1.46E–01	–1.54E+00	–1.38E–01	–1.87E+00	–2.24E+00	5.17E–01	5.80E–01	0
4.16E–01	2.25E+00	–3.27E–02	1.38E–01	9.56E–02	–6.85E–01	1.57E+00	–5.27E–01	–3.22E–01	4.19E–01	4.82E–01	0.
3.86E+00	6.65E–01	–8.62E–01	2.30E–01	–1.52E–01	–3.05E–01	8.96E–01	–3.97E–01	–1.61E+00	–7.64E–02	–8.18E–01	0
1.59E+00	–4.34E–01	1.26E+00	5.97E–02	1.15E+00	–1.17E+00	3.82E–01	–1.18E+00	–5.50E–01	–9.38E–01	5.10E–01	0
–2.17E+00	2.25E+00	6.56E–01	9.42E–01	1.89E+00	–1.10E+00	1.39E+00	6.50E–01	–2.47E+00	–4.56E–01	–1.57E+00	0
–4.20E+00	2.87E+00	1.28E+00	–2.65E–02	1.63E–01	–2.70E+00	2.67E+00	2.67E+00	–9.21E–01	1.09E–01	–1.57E+00	0
–1.94E+00	2.88E+00	5.45E–01	2.73E–02	–4.31E–02	6.44E–01	–2.01E+00	2.76E–01	–1.00E+00	–1.17E+00	–6.68E–01	0
7.16E–01	2.91E–01	4.81E–01	–9.76E–01	3.69E–01	–1.49E+00	7.30E–01	–1.36E+00	9.36E–01	–5.77E–01	–2.07E–01	0
3.72E+00	1.49E+00	1.18E+00	3.09E–01	9.55E–01	–1.26E–01	6.25E–01	–9.36E–01	–5.53E–01	8.63E–01	3.18E–01	0