Temporal and Spatial Evolving Knowledge Base System with Sequential Clustering

Gancho Vachkov, *Member, IEEE*

Abstract—**This paper proposes a computational scheme of a novel Evolving Knowledge Base system that is able to gradually grow and update spatially and temporally. The main assumption is that the input information comes from the real environment in the form of chunks of data (not single data points). Therefore the whole system works in a quasi-real time. Each chunk of data is used for extraction of the so called knowledge items, which is done by a specially introduced sequential clustering algorithm. It is able to discover the separate knowledge items sequentially, in decreasing order of their size. Another important block of the proposed evolving knowledge base system is the updating algorithm, It is in charge of managing the Knowledge Base over time and performs (when necessary) one of the three types recursive computations, namely: learning, relearning and forgetting. The flexibility and the degree of generality of the proposed evolving system is illustrated on a specially constructed example that resembles a real case of data flow coming as a sequence of 20 chunks of data. These data exhibit evolving behavior during the sampling periods and the knowledge Base system is able to catch such behavior by properly updating its parameters. These results show the way of different possible practical applications.**

I. Introduction

The knowledge discovery based on numerical and other data obtained from the real environment is a fast growing research area [1-8]. The general objective here is to aggregate, granulate or generalize the data in a specific way so that a significant part of the information from the past data to be available and usable during the further time periods for approximation, generalization, prediction and another knowledge discovery. The usual problem here is that data from the environment come continuously as large streams of information [1,3,4], or as big chunks of data [2,7] which makes it impossible to keep all time such huge information. Here the concept of the Evolving Knowledge Based Systems is the most appropriate way for knowledge extraction and knowledge management with time, since such systems (by definition) are able to properly *grow*, *update, prune* and *forget* the information in the Knowledge Base over time.

There is a large variety of algorithms and computing techniques for such repetitive operation of the evolving system and the researchers are yet to discover the most effective and plausible operations that mimic the human way of knowledge discovery.

In this paper we propose the computational framework of a kind of Evolving Knowledge Base System, that is computationally simple, but we believe has some similarities with the natural way of extracting, keeping and updating the small portions of information, coming from the Data flow that are incorporated seamlessly in the current structure of the Knowledge Base. This could be continuous and endless process that produces a real evolving system be endless. The merits of the proposed Knowledge Base system is that it is able to grow end evolve spatially and temporally without strict limitations. All the details as well as a special illustration example are given in the sequel.

Gancho Vachkov is with the Department of Reliability-based Information Systems Engineering, Kagawa University, Hayashi-cho 2217-20, Takamatsu-shi, Kagawa-ken 761-0396, Japan (Phone: +81-87-864-2265; Fax: +81-87-864-2262; Email: vachkov@eng.kagawa-u.ac.jp).

II. The Proposed Evolving Knowledge Base System

In this paper we make some general assumptions before constructing the evolving Knowledge Base (KB) system. First of all, we assume that a real system (plant, machine) or environment exists and its behavior during the time can be registered in a numerical way by appropriate number of sensors. These sensors produce a kind of multidimensional Data Flow (data stream) that can be further used for *knowledge acquisition* (also known as *knowledge discovery* or *knowledge building*). Since the general notion of *knowledge* is a bit vague, we define it in the paper in a more concrete way, as follows.

An existing Knowledge Base consists of a number of elements called *knowledge items* KI or *knowledge granules*. Each KI could be considered as a kind of *memory* about some past behavior of the real system (environment). For example, one KI could be a specific, most frequently visited location in the multidimensional space by some part of the data from the data stream for a limited period of time. Therefore, it is clear now that the KI does not simply represent a single data point from the data stream, but is rather a kind of generalization (cluster center) of a group of similar data.

A good understandable way to explain the notion of knowledge in the human language is to consider that knowledge is extracted not from a single word (single data point) but rather from a whole sentence or group of sentences in one paragraph (a portion of data).

The structure of the proposed Evolving Knowledge Base system in this paper can be seen in Fig. 2.1. It is a further development of our previous works in [7,8] on this topic.

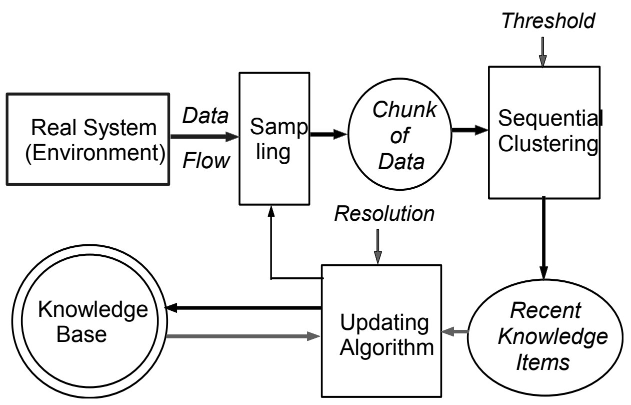


Fig. 2.1. Flowchart of the proposed Evolving Knowledge Base System.

As it is seen from this figure, first the data flow is split (discretized) by appropriate sampling procedure into a sequence of *chunks of data,* (sentences), as mentioned above. The size of the data chunks (the number of data M) , as well as the sampling periods T_S are not discussed in this paper, since they are problem dependent parameters and we would like to explain the main idea.

Generally speaking, the size M of the data chunk corresponds to a reasonable number of data that represent one typical situation in the monitored environment or typical working condition (mode) of a machine. It can vary from one to another physical system. For example, in case of getting knowledge from pictorial information (sequence of images), one chunk of data is actually the whole RGB pixel information contained in one image.

As for the sampling periods T_S between the data chunks, they should not be necessarily equal because the next data chunk could be available (measured) at a farther (different) time instant. In other words, the proposed evolving Knowledge Base system is rather *online system* or more precisely, a *quasi-real time* system, in a sense that the new chunks of data are processed where they are available (and not necessarily within the fixed sampling time). Such assumption is more relaxed one from a computational point of view and represents more closely the real world process of leaning and knowledge building.

Each obtained single chunk of data is further processed in order to extract the most significant (important) knowledge items which will be called *recent knowledge items (recent* KI*)*. This is done by a special newly introduced *sequential clustering* algorithm that is explained in details in the next Section III. The objective of this algorithm is to extract the centers (prototypes) of the groups of data (the clusters) from the data chunk in a decreasing sequence, starting from the largest cluster and continuing to the least cluster. The end of this sequential process is decided by a preliminary given *threshold*, as shown in Fig. 2.1.

Once the *recent* KI are extracted by the sequential clustering, they are further used as inputs of the updating algorithm of the evolving KB, as shown in Fig. 2.1. Here, according to the already existing (old) knowledge items in the knowledge base, three different *updating modes* can be distinguished. They are numbered as **0**, **1** and **2** and shown in the next Fig. 2.2.

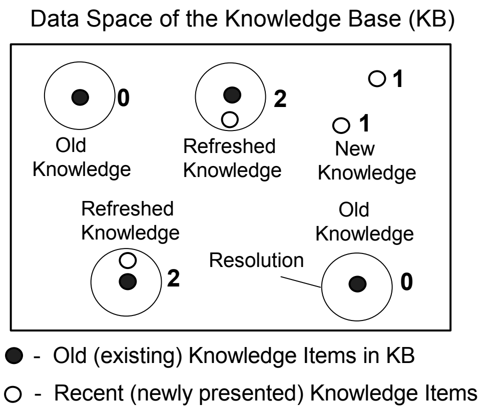

Data Space of the Knowledge Base (KB)
0
Old Knowledge
2
Refreshed Knowledge
1
1
New Knowledge
Refreshed Knowledge
2
Resolution
Old Knowledge
0
● - Old (existing) Knowledge Items in KB
○ - Recent (newly presented) Knowledge Items

Fig. 2.2. Three Modes for updating the knowledge in the Evolving Knowledge Base system, denoted as **0, 1** and **2**.

- *Mode* **0** represents the case when an existing (old) KI in the knowledge base has not been "visited" by any of the *recent* KI from the sequential clustering. This means that all *recent* KI are located farther than the predetermined *resolution* parameter from this existing KI. The *resolution* parameter is a distance that is used to separate the "close knowledge items" from the "far knowledge items" in the KB, as shown in the example in Fig. 2.2. As seen from this example, there are two KI in the KB with mode **0**. These are considered as *old* knowledge items that will *fade out* to some extent (*forgetting step*).

- *Mode* **1** represents the case of a completely *new knowledge item* (*new* KI) for the KB. This means that the *recent* KI is far from all the existing KI in the KB. In such case this new knowledge should receive a *fresh* (first) *learning* step. There are two cases with Mode **1** in the example of Fig. 2.2.

- *Mode* **2** represents the case when a *recent* KI is located in the vicinity of an *existing* KI from the KB, i.e. within the circular area defined by the resolution parameter. Such situation suggests that the *old* existing KI in the KB would be upgraded to a *refreshed* knowledge item which should be *relearned* in some way, taking into account the amount of the knowledge, carried by the *recent* KI. There are two such cases with Mode **2** in the example in Fig. 2.2.

Shortly speaking, *Mode* **0** is *forgetting* step; *Mode* **1** is *first learning* step and *Mode* **2** is *refreshing* (relearning) step. According to the real situation, i.e. the available *recent* KI, coming at each sampling period, the knowledge base will evolve gradually. This means that the number of the knowledge items in the KB will grow monotonically and the amount (the strength) of the knowledge that is carried out by each item will be updated (by respective learning or forgetting). Thus the evolving KB becomes truly *Temporal* and *Spatial Evolving KB system.*

It is worth noting that the knowledge items in the KB will not grow in an uncontrollable way since this process highly depends on the new available information (the new coming chunks of data). According to the characteristics of the sampled information, the frequency of using each of the *Modes* **0, 1** and **2** will change, which basically will prevent the knowledge base from monotonically growing.

A steady growing of the number of KI will occur only in a case of frequently repeating *Mode* **1**, which means repeatedly introducing new knowledge items to the KB. However, if *Mode* **2** is frequently repeated, the KB will stop growing and the old *existing* KI will be refreshed only. Finally, if *Mode* **0** takes place frequently, then the existing KI in the KB will be gradually forgotten (will gradually *fade out*). This means that such knowledge items will still keep their place (location) in the KB, but the strength of their knowledge will decrease steadily.

Here a kind of *threshold* could be introduced in order to distinguish between the *valuable* knowledge and the *noise* (insignificant, faded out) knowledge. Obviously, such threshold is a problem dependent (human defined) parameter, which is used to perform a kind of *pruning* of the KB as often mentioned in some previous works [1-4].

Computational details about the new learning, refreshed learning and forgetting steps carried out by the updating algorithm of the evolving KB, are given in Section IV.

III. Sequential Clustering Algorithm for Extraction of Knowledge Items

Once a chunk of M data has been obtained by the current sampling, the next important action to be taken is to perform an appropriate clustering procedure (in a *quasi-real time* mode) in order to extract the so called *recent knowledge items*, as mentioned in the previous Section. The *recent* KI represent and memorize in some way the areas of *high data density* in the k-dimensional data space.

The best way to extract the *recent* KI is to run some kind of clustering algorithm where the cluster *prototypes* (cluster *centers*) will serve as *locations* of the recent KI. As for the *strength* (the *amount*) of the knowledge of each recent KI, different measures for the *size, volume* or *width* of the extracted cluster could be used.

The most often used clustering algorithms, such as the very popular *Fuzzy C-means* clustering [9] or some other unsupervised learning algorithms [10-12] use the concept of *simultaneous clustering*. This means that the number N_c of the clusters is predetermined and available before the calculations.

The real problem here is that this number is rarely known in advance, which leads to obtaining some *implausible solutions* that have smaller or larger number of clusters than the real ones. To alleviate this problem various criteria for *optimal clustering* have been introduced and often used, such as the *Davies-Bouldin* Index [13], *Dunn*'s Index [14] and some others. However all these criteria give a *posterior* solution of the clustering problem, in a sense that the optimal number of clusters N^* is known after performing unnecessary computation of many possible solutions.

Another problem with the simultaneous clustering is that the extracted clusters do not appear in any special (increasing or decreasing) order of their characteristics (i.e. size, volume) but rather randomly, depending on the initial conditions.

Another group of clustering algorithms uses the idea of *Sequential Clustering* where the number of clusters is not predetermined and the clusters can be gradually extracted (one after another) in a kind of sequence until an appropriate stopping criterion is satisfied.

There are some clear advantages here. First, there are *no redundant computations* with lager than necessary number of clusters. Second, the clusters are extracted in an *ordered sequence*, starting with the most significant cluster (with the largest volume) and proceeding to the least significant (the smallest) cluster.

Probably the most famous sequential clustering algorithm is the Mountain Clustering [15,16] with some of its versions that use the so called mountain (or potential) function to discover in a sequence the areas of highest density in the data space. This algorithm is easy to implement but has some problems with the proper selection of the parameters (especially the *width*) of the new subtracted mountain function after each discovered (and removed) cluster.

Other sequential clustering algorithms use the graph spectral method [17] for clustering, but are quite demanding from a computational and memory viewpoint because they need large matrix computations.

In this paper we propose a novel sequential clustering algorithm that needs a small number of tuning parameters and is quite robust in proper discovering the clusters that are automatically arranged in decreasing order of their size (volume). Computational details of this algorithm are given in the sequel.

We assume that a chunk of M data in the K-dimensional space is available: $\mathbf{x}_i = [x_{i1}, x_{i2}, ..., x_{iK}]$, $i=1,2,...,M$. The objective is to extract the centers (prototypes) $\mathbf{C}_i = [c_{i1}, c_{i2}, ..., c_{iK}]$, $i=1,2,...,n$ of the clusters, arranged in *decreasing order* of their volumes $V_S, s = 1,2,...,n$, i.e. $V_1 \geq V_2 \geq ... \geq V_n$.

The *cluster volume* V_S can be defined in different ways, but in general this is a kind of measure of the *density* of the cluster or measure of its *size* in the K-dimensional data space. It will be defined in the sequel.

The typical clustering algorithms are from the group of the *unsupervised learning* algorithms. However, in our proposed sequential clustering algorithm we solve a direct *optimization* problem, namely maximizing the cluster volume, so that we are dealing actually with a *supervised learning.*

First of all, we define the so called *Cover Function* H_i, which is a standard *Gaussian* function with a current location of the center $\mathbf{c}$ and a fixed (predefined) *width* σ as follows:

$$H_i = \exp\left(-\frac{\|\mathbf{c}-\mathbf{x}_i\|^2}{2\sigma^2}\right), \; i=1,2,...,M \qquad (1)$$

The *Cover Function* calculates the *proximity level* between the data point $\mathbf{x}_i$ and the current center $\mathbf{c}$ of the function. Here $H_i \to 0$ means *Low Proximity* (*far* distance between the center $\mathbf{c}$ and the data point $\mathbf{x}_i$, while $H_i \to 1$ means *High Proximity* (a *short*, close to *zero* distance between $\mathbf{c}$ and $\mathbf{x}_i$).

Then the volume V of the current cluster is defined by the following function, which sums the weighted proximities of all data $\mathbf{x}_i$ to the current $\mathbf{c}$ location of the cover function, as follows:

$$V = \sum_{i=1}^{M} P_i H_i = \sum_{i=1}^{M} P_i \exp\left(-\frac{\left\| \mathbf{c} - \mathbf{x}_i \right\|^2}{2\sigma^2} \right) \qquad (2)$$

Here $P_i \in [0,1]$, $i = 1,2,...,M$ is a kind of *weight* parameter, called P*ower* (*Capacity*) of the data point $\mathbf{x}_i$. At the beginning of the computation process, it is assumed that all data have a *full power* (full capacity):
$P_i = 1.0$, $i = 1,2,...,M$. Once a cluster s is extracted from the data set, then the power of all data points is decreased by the following recursive calculation:

$$P_i = P_i - P_i H_i = P_i(1 - H_i),\ i = 1,2,...,M \qquad (3)$$

Then the problem of finding the current cluster s, $s = 1,2,...$ becomes an *optimization* problem of maximi zing the volume V of the cluster, computed by (2).

The type of the optimization algorithm used obviously affects the accuracy of the obtained solution V*max* as well the computation time. However this does not change the general idea of the proposed sequential algorithm.

In this paper we have used a modification of the well known Particle Swarm Optimization (PSO) algorithm, namely its popular version: PSO with *Inertia Weight* as explained in [18]. In all further simulations we have assumed the following parameters for the PSO: number of particles *Np = 8; Inertia weight* parameter, linearly decreasing from $\omega = 1.8$ to $\omega = 0.4$ and maximal number of Iterations *Iter = 400*. If the criterion V is stabilized within a small predetermined threshold, the algorithm terminates automatically with less iterations..

Fig. 3.1. depicts a numerical example of *1600* data considered as one chunk of data that has to be clustered properly in order to extract the *recent* knowledge items (*recent* KI). It is easy to notice that there are *4* distinct clusters in this chunk, and the proposed sequential clustering algorithm should be able to detect all of them.

It took less than *15* sec in average on a PC with 3 GHz CPU to find each current cluster. Here we would like to note that the actual computation time is not a performance limitation for the proposed Knowledge Base System, since (as mentioned above) it is working in a *quasi-real time* mode.

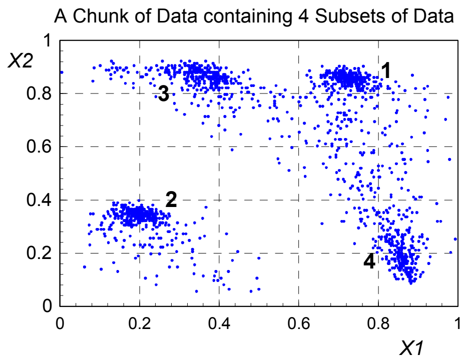


Fig. 3.1. Example of a chunk of data, consisting of 4 subsets of data (labeled as 1, 2, 3 and 4). Each subset has 400 data and the total number of data in this chunk is $M = 4 \times 400 = 1600$.

It is clear that the modified PSO algorithm should be run repetitively for each subsequent cluster. Therefore finding a "good" initial (starting) position is important task for reducing the total computation time. For such purpose we have made here a small improvement of the convergence by determining the *initial area* of the next subsequent step to be within the area of the already found optimum from the previous optimization step (with a predetermined *width* around the optimum). This simple idea is illustrated for the second step of the sequential clustering in Fig. 3.2. and the better (speedier) convergence obtained can be seen in Fig. 3.3.

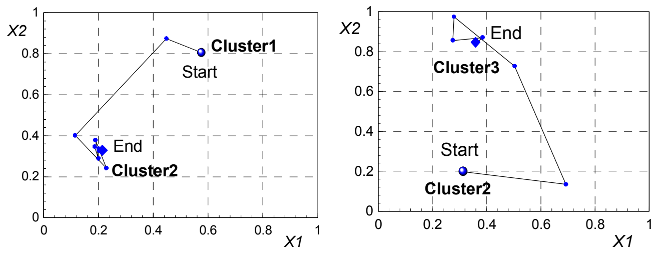


Fig. 3.2. Illustration of the proposed scheme for determining the *initial area* for the subsequent optimization. It is seen that the initial area of the second optimization is defined around the optimum, found at the first optimization.

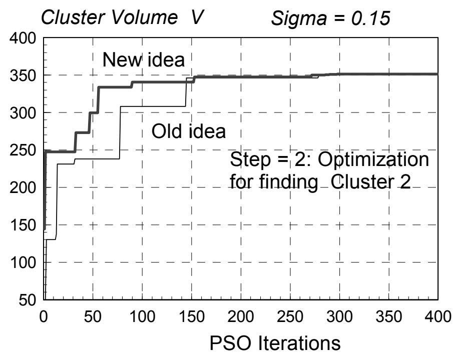


Fig. 3.3. Improved conversion of the PSO algorithm for finding the Cluster **2** (shown as *new idea*) by defining the new initial area according to Fig. 3.2.

The results from the sequential clustering performed for a sequence of *8* steps (*8* clusters) are shown in Fig. 3.4.

Now it becomes clear that a proper *stopping criterion* is necessary to avoid the redundant computation steps that could lead to discovering *insignificant* clusters.

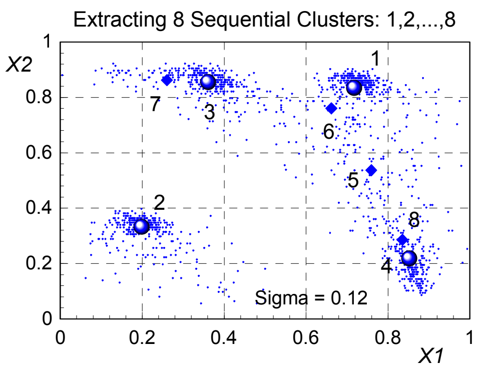


Fig. 3.4. Example of Chunk of Data with 4 Subsets of data (numbered as 1,2,3 and 4) used for illustration of the proposed Sequentia

The computation results shown in Fig. 3.5 with different number of steps (different number of sequential clusters) provide some hint about the construction of the stopping criterion. It is seen that there is a big drop in the cluster volume between the solutions with *4* and *5* clusters.

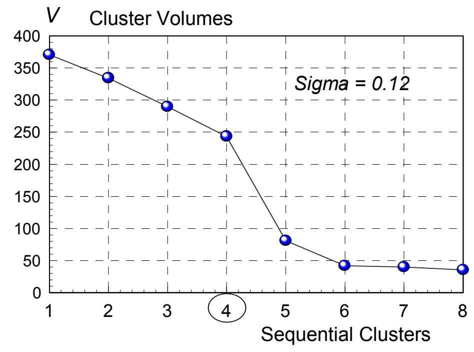


Fig. 3.5. The volumes of the clusters gradually decrease with increasing the steps (clusters) of the proposed Sequential Clustering algorithm.

Figure 3.6. suggests that finding the *maximum* in the volume differences between two neighboring clusters *i* and *i+1* (*4* and *5* in this case) could be used as a robust stopping criterion. According to this idea, the proper number of clusters is *s = 4* and it does not change with wide changes of the widths: $\sigma = 0.04;0.06;0.08;0.10;0.12;0.15;0.18;0.20$.

Further on we used this criterion for all other simulations throughout the paper.

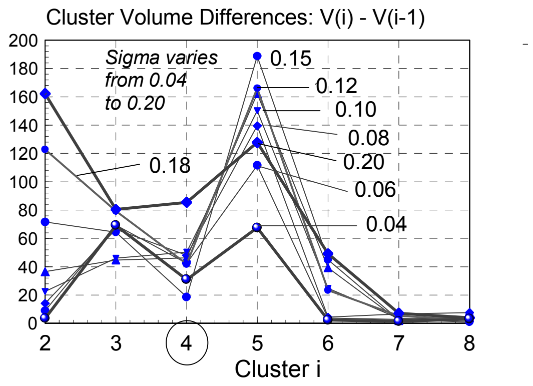


Fig. 3.6. The peak in the differences between the neighboring Volumes V(i) and V(i-1) can be used as a Stop Criterion for the proposed Sequential Clustering algorithm. As seen, this algorithm is relatively insensitive to the assumed width parameter *Sigma*, varying between *0.04* and *0.20*.

IV. Updating Algorithm of the Evolving Knowledge Base System

As seen from the flowchart in Fig. 2.1., Section II, the updating algorithm is run at each new sampling, when a new chunk of data is obtained and a respective number of n* *recent* KI is extracted by the sequential clustering algorithm explained in the previous Section.

The objective of the updating algorithm is to define the amounts of *learning*, *relearning* and *forgetting* at each sampling step. They correspond to the *Modes* **1, 2** and **0,** explained in Section II. A general illustration of this dynamical *learning-forgetting* process is given in Fig. 4.1.

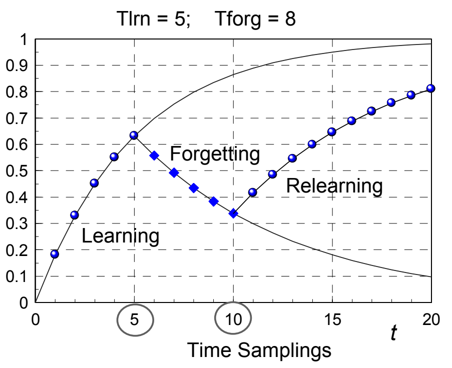


Fig. 4.1. Example of a three-step sequence of Learning, Forgetting and Relearning, with different time constants for learning and forgetting.

Here the assumption that learning and forgetting are exponential processes with respective time constants $T_{\text{lrn}} = 5$ and $T_{\text{forg}} = 8$ is made. The computations are as follows:

$$W(t)=G(t)\exp(-t/T_{lrn}); \qquad (4)$$
$$W(t)=G(t)[1-\exp(-t/T_{forg})]$$

Here $t = 1,2,...$ is the time sampling step; *W(t)* is the amount of knowledge of this KI at sampling time *t* and *G(t)* is the variable *Gain* of the exponential process of learning or forgetting. It is computed as a *range* between the current knowledge amount and the *best* possible knowledge (after unlimited repetitions of the same learning). Such computation (recalculation) of *G(t)* is made at each change of the learning-forgetting mode, as shown in Fig. 4.1. for time instances: *t = 5* and *t = 10*.

The above computation scheme has some complications when performed in a recursive way. Therefore we propose here another, simpler and more flexible *recursive computation* scheme for *learning*, *relearning* and *forgetting* that uses *one time constant* called *Learning Constant* T_{lrn} for all three processes (all three modes). The important point here is that this learning scheme uses a simple *one-step-only* computation of the exponential process with *updated* (recalculated) *gain G(t)* at each sampling period *t*. Below are the recursive calculations for the three types of learning:

- *Mode* **1** (first learning):

$$W(t)=G(t)\exp(-t/T_{lrn}); \qquad (5)$$
$$G(t) = V(t)$$

- *Mode* **2** (relearning):

$$W(t) = W(t-1) + G(t)\exp(-t/T_{lrn}); \quad G(t) = V(t) - W(t-1) \qquad (6)$$

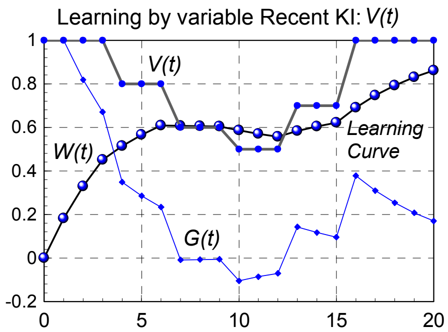
- *Mode* **0** (forgetting)

$$W(t) = W(t-1) - G(t)\exp(-t/T_{lrn}); \quad G(t) = W(t-1) \qquad (7)$$

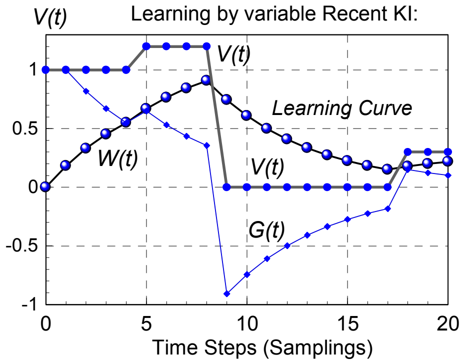
The variable *V(t)* represents amount of knowledge, carried out by the a *recent* KI at the time sampling *t*. As seen in (5), *V(t)* represents the *new* KI in the KB; in (6) *V(t)* represents the *refreshed* KI in the KB and finally, in (7) such knowledge item is missing, i.e. *V(t) = 0.0*.

The amount of the knowledge *V(t)* of the *recent* KI is actually the *Volume* of the respective cluster for this KI, extracted by the sequential clustering algorithm from Section III. If *V(i)* is variable and changing over time, the respective Gain *G(t)* of the exponential process in (5),(6) and (7) will also change.

This dynamical process of learning-forgetting is illustrated in Fig. 4.2 on two examples with different patterns of changing the volume *V(t)* over time. This is equivalent to presenting a sequence of different *recent* KI to the KB over time. It is clearly seen that the respective amount of knowledge *W(t)* carried by the KI is also dynamically changing in both directions (increasing-decreasing).



a)



b)

Fig. 4.2. Illustration of the dynamical process of learning-forgetting by use of the proposed recursive calculations in (5), (6) and (7). Here the recent knowledge items have different volumes *V(t)* over time.

V. Example of Knowledge Acquisition by Use of the Evolving Knowledge Base System

The whole computational scheme of the proposed evolving Knowledge Base System in Sections II, II and IV could be better understood by an appropriate example. For such purpose we have designed a special numerical example that shows the main features and the computation steps of the proposed scheme.

As a starting point we use the same example from Fig. 3.1., which contains *4* Subsets (each of them with *400* data) in the *initial chunk* of *1600* 2-dimensional data points. This is the *initial condition* of the real system at the initial sampling period of *t=0*.

Then we suppose that these four subsets *evolve* over time by changing their *centers of gravity* in six steps (six positions within the data space *X1–X2*). They are numbered *as 0,1,..,6* and shown as a specific trajectory for each subset in Fig. 5.1 For simplicity in generating the example, but without loss of generality, we keep the shape and the approximate number of data (about *400*) for each subset during the evolution. The subsets evolve from one to another step at different sampling times, as shown in a qualitative way in the next Fig. 5.2. This figure helps to understand the dynamics of the evolution process of all *21* sampling periods, numbered as *0,1,2,...,20* at which respective chunks of data are obtained for further processing. As seen from Fig. 5.2., at time sampling *t=19* and *t=20* all subsets reach their final position in the space, as shown in Fig. 5.3.

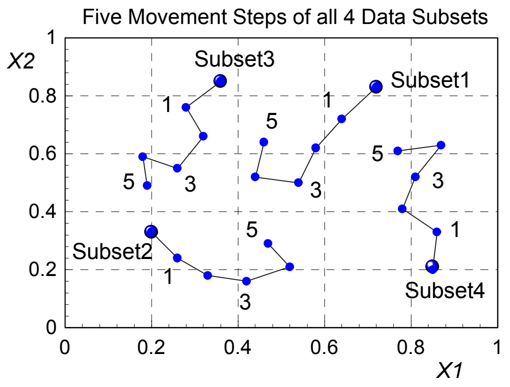


Fig. 5.1. The six-steps trajectories of all 4 subsets of data from Fig. 3.1., numbered as 0,1,2,3,4 and 5 over time (0 means the initial condition).

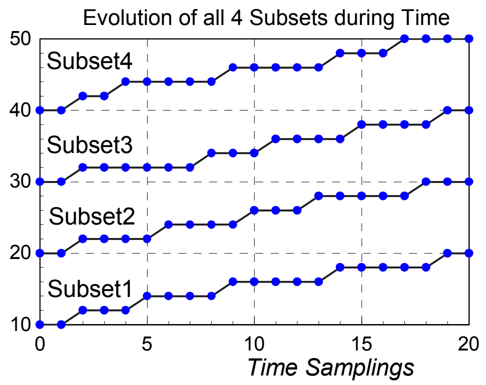


Fig. 5.2. Evolution of the 4 subsets for the first 21 sampling periods. Each subset is evolved gradually from level 0 to level 6 at different time samplings. These levels are only qualitatively shown in the figure.

The whole computational scheme from Fig. 2.1 has been performed for each of the sampling periods *t = 0,1,2,...,20*.

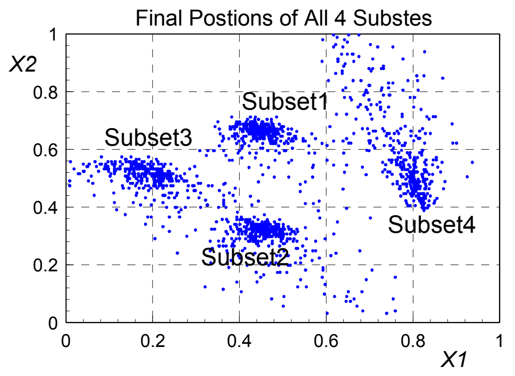
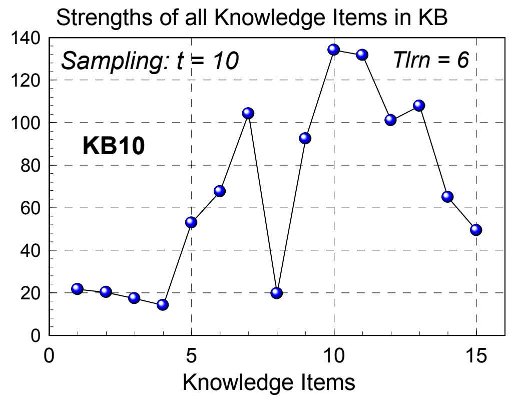


Fig. 5.3. The final positions of the *4* subsets in the space at time samplings *t = 19* and *t = 20.*

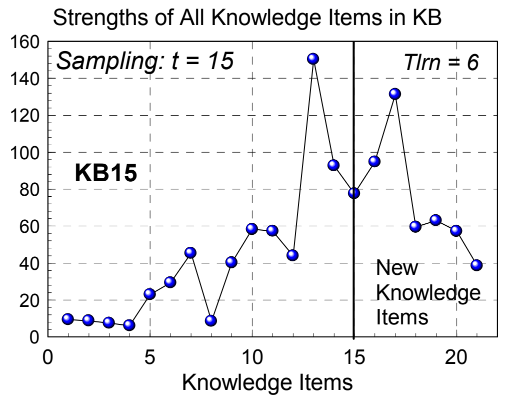
This means that at each sampling the respective *chunk of data* has been obtained and used for *sequential clustering* and discovering the *recent* KI, according to Section III. Then the u*pdating algorithm* from Section IV was applied for a proper *learning*, *relearning* and *forgetting* of the knowledge items in the Knowledge Base.

Because of the specific assumptions in creating this numerical example, each chunk of data has produced the same number of *n = 4 recent* knowledge items KI. The reason is that there exist only *4* subsets of data at each sampling in this example and they keep approximately the same shape and size in the input space during the evolution. However this specific case is not a constraint for the general computation scheme, where there could be different (variable) number of *recent* KI at each sampling period.

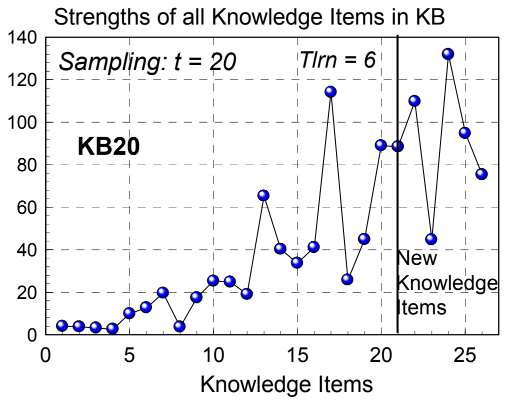
The evolved Knowledge Bases **KB10**, **KB15** and **KB20** at time samplings *t = 10, 15* and *20* respectively are shown in Fig. 5.4. The time-constant used for learning was $T_{\text{lrn}} = 6$. It is easy to notice from this figure that the knowledge base **KB** is gradually evolved during the samplings in two ways. First, it is enlarged with additional (new) knowledge items as a result of the learning evolution. Second, the existing knowledge items in **KB** from the previous samplings change their amount of knowledge at the further samplings according to the updating algorithm. For example it becomes clear that the oldest knowledge items *1,2,3* and *4* in Fig. 5.4. are gradually *fading out* and almost disappear at the last sampling *t = 20,* while knowledge item *17* remains relatively strong.



a)



b)

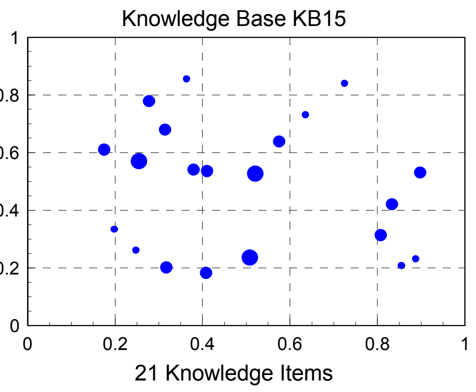


c)

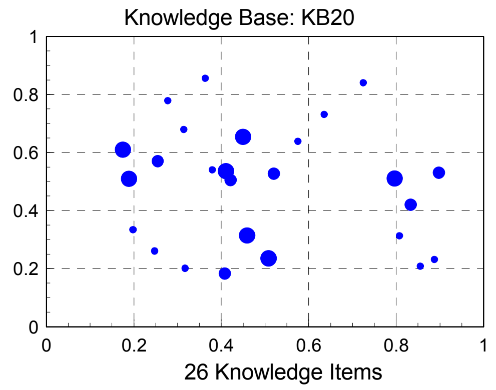
Fig. 5.4. The amount (strength) of knowledge for all knowledge items in the Knowledge Base at 3 different time samplings: *t = 10, 15 and 20*. The new added knowledge items are also depicted.

The locations of the knowledge items from **KB15** and **KB20** are depicted in Fig. 5.5. In order to make visualization of both the location and the strength of the knowledge items, we have performed *clustering* of all KI in three groups (*strong, medium* and *weak* knowledge). They are shown in Fig. 5.5*a* and Fig. 5.5.b as three groups of curve symbols with *3* different sizes (*Big*, *Medium* and *Small*). Then it is easy to visually notice that the amount of the knowledge carried out by each knowledge item is *evolving* during time, if we consider each separate KI as a certain fixed location (coordinate) in the space. As a result some KI are gradually fading out while others are growing (if regularly refreshed).

This evolving process could be even better visualized as a gradual *size motion* of the knowledge items at each sampling time.



a)



b)

Fig. 5.5. Locations of the knowledge items of the Knowldge Bases KB15 and KB20 in the 2-dimensional data space. For a better visualization of the amount of knowledge, carried out by each knowledge item, all KI have been clustered into three groups, shown as *Big*, *Medium* and *Small* ball-shape curve symbols.

VI. Conclusions and Discussions

We have proposed in this paper a general and flexible computation scheme for creating and managing an Evolving Knowledge Base System. The input information for this system is in the form of chunks of data, rather than single data points.

There are two important computation procedures in this framework, as follows. First, a plausible number of knowledge items are extracted from each chunk of data by the newly introduced sequential clustering algorithm. Second, these knowledge items are used by a special updating algorithm, which performs in a recursive way one of the three possible updating operations: learning, relearning or forgetting. Thus the knowledge base system behaves as a real evolving system that is able to *memorize* a new knowledge, to *refresh* an older knowledge and to gradually forget (fade out) an old knowledge.

The most distinct features of the proposed evolving knowledge base system are as follows:

1) It uses a small number of tuning parameters, such as time constant for *learning* and a *resolution* parameter, which are problem dependent and relatively easy to be predefined by the user. By changing these parameters different versions of evolving knowledge base systems can be created that mimic the reaction (and the thinking way) of different humans (agents). Such idea could be used for creating and analyzing the behavior of a multi-agent system.

2) The proposed evolving system uses simple computations and therefore could be successfully used for different practical applications. Some of them are now under consideration, such as quasi-real time monitoring of images, in order to "memorize" the images that appear most frequently.

References

[1] Angelov, P. and Filev, D., "An Approach to Online Identification of Takagi-Sugeno Fuzzy Models", IEEE Trans. on Systems, Man and Cybernetics, vol. 34, No. 1, pp. 484-498, 2004.

[2] Ozawa S., Pang, S. and Kasabov N., "An Incremental Principal Component Analysis for Chunk Data", Proc. of the 2006 IEEE Int. Conference on Fuzzy Systems, FUZZ-IEEE 2006, Vancouver, pp. 10493-10500, July, 2006.

[3] Zhou, X, and Angelov, P., "Real-Time Joint Landmark Recognition and Classifier Generation by an Evolving Fuzzy System", Proc. of the 2006 IEEE Int. Conference on Fuzzy Systems, FUZZ-IEEE 2006, Vancouver, pp. 6314-6321, July, 2006.

[4] Soltic, S. Wysocki S. and Kasabov N., "Evolving Spiking Neural Networks for Taste Recognition", Proc. of the 2008 *IEEE Int. Conference on Fuzzy Systems*, FUZZ-IEEE 2008, Hong Kong, pp. 2092-2098, June, 2008.

[5] Pang, S. and Kasabov, N., "r-SVMT: Discovering the Knowledge of Association Rule over SVM Classification Trees", Proc. of the 2008 *IEEE Int. Conference on Fuzzy Systems*, FUZZ-IEEE 2008, Hong Kong, pp. 2487-2494, June, 2008.

[6] Angelov, P. Ramezani, R. and Zhou, X., "Autonomous Novelty Detection and Object Tracking in Video Streams using Evolving Clustering and T-S type Neuro-Fuzzy system", Proc. of the 2008 *IEEE Int. Conference on Fuzzy Systems*, FUZZ-IEEE 2008, Hong Kong, pp. 1457-1464, June, 2008.

[7] Vachkov, G., "Real Time Knowledge Acquisition Based on Unsupervised Learning of Evolving Neural Models", CD ROM Proc. of the FUZZ-IEEE 2007 Conference, Imperial College, London, pp. 1333-1338, July, 2007.

[8] Vachkov, G., "Human-Assisted Fuzzy Image Similarity Analysis Based on Information Compression", Journal of advanced Comp. Intelligence, JACIII, vol. 13, no. 3, pp. 255-261, 2009.

[9] Bezdek, J.C., *Pattern Recognition with Fuzzy Objective Function Algorithms*, New York: Plenum Press, 1981.

[10] Martinetz T., Berkovich S., and Schulten, K., "Neural-Gas Network for Vector Quantization and Its Application to Time-Series Prediction", *IEEE Trans. Neural Networks*, Vol. 4, No. 4, pp. 558-569. 1993.

[11] Xu, L., Krzyzak A. and Oja, A., "Rival Penalized Competitive Learning for Clustering Analysis, RBF Net and Curve Detection", *IEEE Trans. Neural Networks*, Vol. 4, No. 4, pp. 636-649, 1993.

[12] Angelov, P., "An Approach for Fuzzy Rule-based Adaptation using On-line Clustering", *International Journal of Approximate reasoning*", vol. 35, no. 3, pp. 275-289, 2004.

[13] Davies, D. and Bouldin, D., "A Cluster Separation Measure", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 1, pp. 224-227, 1979.

[14] Dunn, J.C., "Well Separated Clusters and Optimal Fuzzy Partitions", *Journal of Cybernetics,* vol. 4, pp. 95-104, 1974.

[15] Yager R. and Filev D., "Approximate Clustering via the Mountain Method", *IEEE Trans. on Systems, Man and Cybernetics*, vol. 24, No. 8, pp. , 1994.

[16] Clark, A. and Filev, D., "Clustering Techniques for Rule Extraction from Unstructured Text Fragments", Proc. of the NAFIPS'05 Conference, Ann Arber, USA, pp. , July 2005.

[17] Inoue, K., Urahama, K., "Sequential Fuzzy Cluster Extraction by a Graph Spectral Method", Patter Recognition Letters, vol. 20, pp. 699-705, 1999.

[18] Poli, R., Kennedy, J. and Blackwell, T., "Particle Swarm Intelligence. An Overview", *Swarm Intelligence*, vol. 1, pp. 33-57, 2007.