Tieju Ma and Yoshiteru Nakamori, An agent-based approach for predictions based on multi-dimensional complex data, *International Journal of Information Science*, Elsevier Science. (In Press)

An Agent-Based Approach for Predictions Based on Multi-Dimensional Complex Data ⋆

Tieju Ma *, Yoshiteru Nakamori

School of Knowledge Science, Japan Advanced Institute of Science and Technology, 1-1, Asahidai, Tatsunokuchi, Ishikawa 923-1292 Japan

Abstract

This paper presents an agent-based approach to the identification of prediction models for continuous values from multi-dimensional data, both numerical and categorical. A simple description of the approach is: a number of agents are sent to the data space to be investigated. At the micro-level, each agent tries to build a local linear model with multi-linear regression by competing with others; then at the macro-level all surviving agents build the global model by introducing membership functions. Three tests were carried out and the performance of the approach was compared with a neural network. The results of the three tests show that the agent-based approach can achieve good performance for some data sets. The approach complements rather than competes with existing Soft Computing methods.

Key words: agent-based approach, membership function, prediction

1 Introduction

Prediction can be viewed as the construction and use of a model to assess the class of an unlabeled sample, or to assess the value or value ranges of an attribute that a given sample is likely to have[4]. Prediction of discrete or nominal values is commonly dealt with by classification techniques, and

⋆ This research was sponsored by the 21st century COE programme in JAIST.

* Corresponding author. Address before January 2005: School of Knowledge Science, Japan Advanced Institute of Science and Technology, 1-1, Asahidai, Tatsunokuchi, Ishikawa 923-1292 Japan. Address from January 2005: International Institute for Applied Systems Analysis, Schlossplatz 1, A-2361 Laxenburg, Austria.

Email addresses: tieju@jaist.ac.jp (Tieju Ma), nakamori@jaist.ac.jp (Yoshiteru Nakamori).

for the prediction of continuous values, statistical techniques of regression are widely used.

Modeling by statistical techniques of regression is a top-down method. Researchers first observe the distribution of all the data, form the whole structure of the data space in their minds, then select a certain function to fit the data using experience and specific knowledge. But sometimes it is very difficult or impossible to find a function that fits the distribution of all the data (like the data in Fig.1) using statistical regression techniques in a top-down way. Certainly we can divide all the data in Fig. 1 into several parts, and for every part we can find a function to fit it. But when we deal with high-dimensional data, for example 6-dimensional data, how can we know where we should separate the data?

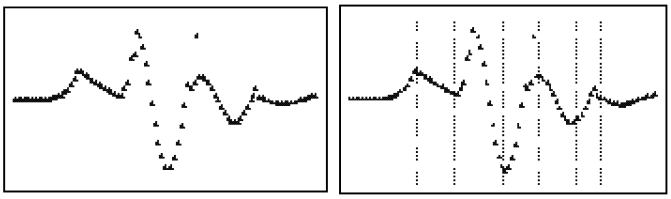

Fig. 1. A case which is difficult to model.

As a foundation component for the emerging field of conceptual intelligence, Soft Computing (SC) which includes Fuzzy Logic (FL), Neural Computing (NC), Evolutionary Computation (EC), Machine Learning (ML), and Probabilistic Reasoning (PR) [7] can be used to deal with the kind of data shown in Fig. 1. For example, the "hyperellipsoidal clustering method" described in [6] assists modelers in finding fuzzy subsets suitable for building a fuzzy model. The approach presented in this paper complements rather than competes with existing Soft Computing methods.

During the 1980s and 1990s, many disciplines converged into an interdisciplinary research field, called complex adaptive systems. The belief underlying this complex adaptive systems research is that the macro-level complexity of the system may be spontaneously derived from interactions at the micro-level; and that this fundamental mechanism should be common among different systems in different fields. In the research on complex adaptive systems, scientists

Table 1. An example of "complex" data

	Attribute 1 (Length)	Attribute 2 (Color)	$\cdots$	Attribute m (Size)
object 1	85	Black	$\cdots$	Large
object 2	90	Red	$\cdots$	Small
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
object n	77	Blue	$\cdots$	Middle

have obtained myriad heuristics from nature. Many algorithms and proposals, especially in the domains of optimization, telecommunication networks and robotics, were inspired by modeling the collective behaviors of social insect colonies and other animal societies [1–3,8]. Enlightened by that thinking, we developed an agent-based approach to the identification of prediction models for two-dimensional numerical data [5]. The main purpose of this paper is to develop an agent-based approach to the identification of prediction models for multi-dimensional complex data. Here, "complex" means that some of the dimensions are numerical while others are categorical. Table 1 shows an example of "complex" data. The data in attribute *length* is numerical. The data in attribute *color* is nominal, which is treated as categorical data in this paper. And the data in attribute *size* is ordinal, in that we can use a number to denote each size; for example, we can use the number 3 to denote the size "Large", 2 to denote "Middle", and 1 to denote "Small". Thus, the distance between different sizes may be different. In this paper, categorical data only refers to nominal data, and ordinal data are treated as numerical data. While there is no clear border between nominal and ordinal data, people sometimes need to determine whether the data is nominal or ordinal. For example, if the objects in Table 1 are clothes, it is most likely that people will treat the attribute *color* as nominal. But if the attribute *color* in Table 2 denotes a physical feature in some chemical test, it is most likely that people will treat it as ordinal data, considering the visible light spectrum as shown in Fig. 2.

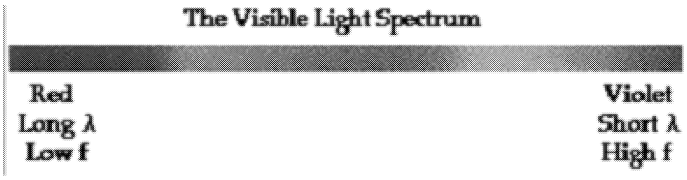

Fig. 2. Colors are ordinal when considering the visible light spectrum.

The previous work in [1–3,8] mainly tried to capture the features of the cooperative behaviors of insect or animal societies, while our work applies the natural law of survival of the fittest. There is both competition and cooperation in our approach. Agents compete with each other to build local linear models, and those agents who have good performance in building local models will be the survivors. Surviving agents cooperate with each other in building the global model, since in most cases, the global patterns of the data cannot be explained by any individual agent's local model.

2 The Agent-Based Approach

2.1 Basic definitions

Following are some basic definitions of the approach:

- **Data space & data object**. A data space is a data set with several categorical/numerical dimensions (or attributes), and a data object is a data record within a data space.

  Suppose N is the number of data objects in a data space, and each data object has s categorical attributes/dimensions and u numerical attributes, Table 2 shows a data space and data objects in it.

Table 2. A data space

Data	Categorical			Numerical		
Objects	A_1^c	$\cdots$	A_s^c	A_1^n	$\cdots$	A_u^n
O_1	a_{11}^c	$\cdots$	a_{1s}^c	a_{11}^n	$\cdots$	a_{1u}^n
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
O_N	a_{N1}^c	$\cdots$	a_{Ns}^c	a_{N1}^n	$\cdots$	a_{Nu}^n

- **Agent**. In this paper, an agent is a programmed unit, or an instance of a class from the object-oriented programming perspective. It has a vision, a grade, a territory, and the knowledge for doing linear regression. It can adjust its vision, find data objects in its vision, build a local linear model in its territory, increase its grade, compete with other agents for survival (eat other agents or be eaten by them) and cooperate with other surviving agents to build the global model.
- **Local model**. A local model means a model built by an individual agent. In a broad sense, a local model can be of different types, i.e., it can be either linear or nonlinear. In this paper, however, the local model built by any individual agent is linear, and agents built their local models by least square regression.
- **Agent's territory & agent's core territory**. An agent's territory is the data set used by the agent to build its local linear model. Not all of the dimensions/attributes of an agent's territory are used for building the local linear model. When carrying out multi-linear regression, agents will select attributes to build local models according to a variance-covariance matrix. An agent's core territory includes all the data objects in its territory, but it does not cover the attributes which are not used to build the agent's local model.

  Fig. 3 shows an example of an agent's territory and core territory. The agent builds its local linear model on data objects O_2, O_3, O_4, O_5, and O_6

with attributes A_1 and A_2. So both the agent's territory and the agent's core territory are composed of the 5 data objects, but the latter doesn't cover attribute A_3.

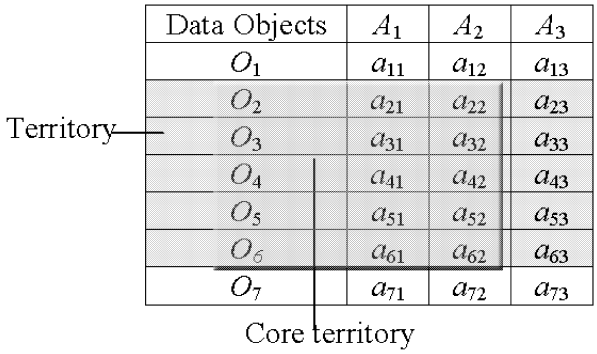

Fig. 3. An agent's territory & core territory.

A simple description of the approach presented in this paper is as follows. A number of agents are sent to the data space under investigation. At the micro-level, individual agents use multi-linear regressions to build local linear models in their territories; at the same time, they try to expand their territories by moving in their territories and finding new data objects in their vision. Correlation coefficients are used as rules to decide whether or not agents succeed in expanding their territories. During the process of expanding their territories, there is survival competition among agents. Then at the macro level, the local linear models built by those surviving agents are integrated into a global model by introducing membership functions.

2.2 Initialization

Considering the data space in Table 1, N agents are sent into it with each agent mounting on a data object. Each agent's grade is initialized as

$$G = 1. \tag{1}$$

And each agent's vision is initialized as the average distance among all data objects

$$V_0 = \frac{\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} D\left(O_i, O_j\right)}{C_n^2}. \tag{2}$$

Here $D(O_i, O_j)$ denotes the distance between the two data objects O_i and O_j. There are various ways to define $D(O_i, O_j)$. When using the approach,

users can define different $D(O_i, O_j)$ according to their informed knowledge and experience. In the tests which will be introduced in the next section, $D(O_i, O_j)$ is defined as

$$D(O_i, O_j) = \sum_{h=1}^{s} d_h^c(O_i, O_j) + \sqrt{\sum_{h=1}^{u} \left(\frac{a_{ih}^n - a_{jh}^n}{A_{h*\max}^n - A_{h*\min}^n} \right)^2}. \tag{3}$$

The first item in Eq. (3) denotes the distance on all categorical attributes; $d_h^c(O_i, O_j)$ denotes the distance between two data objects in categorical (nominal) attribute A_h^c. $d_h^c(O_i, O_j)$ is simply defined as:

$$d_h^c(O_i, O_j) = \begin{cases} 0 \ (a_{ih}^c = a_{jh}^c) \\ 1 \ (a_{ih}^c \neq a_{jh}^c) \end{cases}. \tag{4}$$

The second item in Eq. (3) denotes the distance on all numerical attributes. The $A_{h*\max}^n / A_{h*\min}^n$ is the maximum/minimum value in numerical attribute A_h^n. The purpose of introducing $A_{h*\max}^n$ and $A_{h*\min}^n$ is to map the distance in each numerical attribute to the range $[0, 1]$. If the values in an attribute are all the same, i.e.,

$$A_{h*\max}^n - A_{h*\min}^n = 0,$$

then this attribute will not be considered for modeling.

2.3 Building local linear models

After initialization, every agent tries to build a local linear model according to the following steps:

- Step 1: The agent counts the number of objects in its vision. If the number is smaller than 3, this agent dies; otherwise it goes to step 2.
- Step 2: An agent does a multi-linear regression analysis of all the data objects in its vision. If the correlation coefficient $r \geq p$, it goes to step 4; otherwise it goes to step 3.

  Here p is a design parameter, with a value set by the user. In the tests described in Section 3, we think a local linear model with the correlation coefficient $r \geq 0.88$ is of enough linearity, so we set $p = 0.88$.
- Step 3: The agent decreases its vision by $q\%$ ($q < 100$) of the initialized V_0. If its vision becomes negative, it is removed; otherwise it goes to step 1.

  Here q is another design parameter. In the tests in Section 3, we think a $q = 10$ is reasonable, thus an grade-1 agent has 10 chances to build a local linear model by decreasing its vision before it is removed from the data space.

- Step 4: The agent builds a local linear model and evolves into a grade 2 agent, i.e.,

$$G = 2.$$

This progression is illustrated in Fig. 4. The t denotes how many times the agent has decreased its vision.

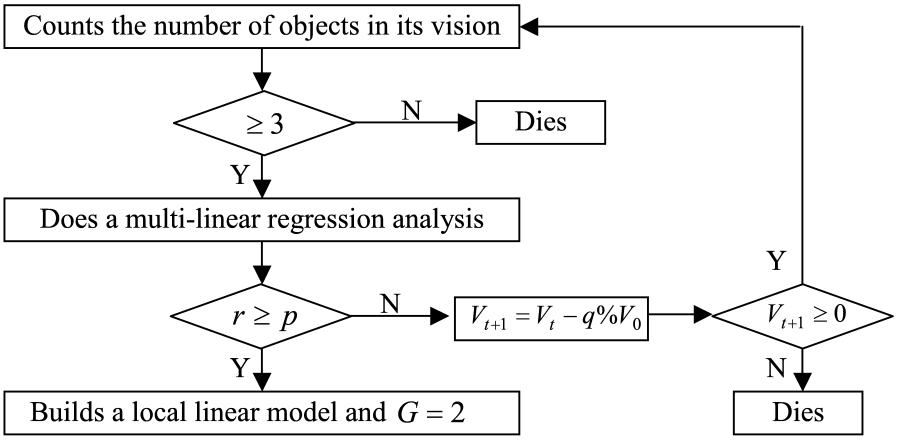

Fig. 4. The process of building a local linear model.

Agents will select attributes for building local linear models according to a variance-covariance matrix. In an agent's territory, if there are large covariances between one attribute and all other attributes (except that which we want to predict), then that attribute will not be considered when building local models. In the tests in Section 3, we defined a large covariance as 0.75. The categorical attributes are treated as discrete variables when doing multi-linear regression.

2.4 Expanding territory

Those agents that evolve into grade-2 agents try to expand their territories according to the following steps:

- Step 1: The agent finds all frontier data objects of its territory, and memorizes them in a list fo. The agent also memorizes its current territory as T_0. At the same time, the agent checks to find other agents whose core territories are covered by (not equal) its core territory. If it finds some, it eats them, i.e., those agents are removed from the data space. If two agents have the same core territory, then the agent with a better local model, i.e., a larger correlation coefficient r, will be the winner of the survival competition, and it will eat the other. If two agents have the same core territory and have the same correlation coefficient r for their local model, then the winner is randomly decided.

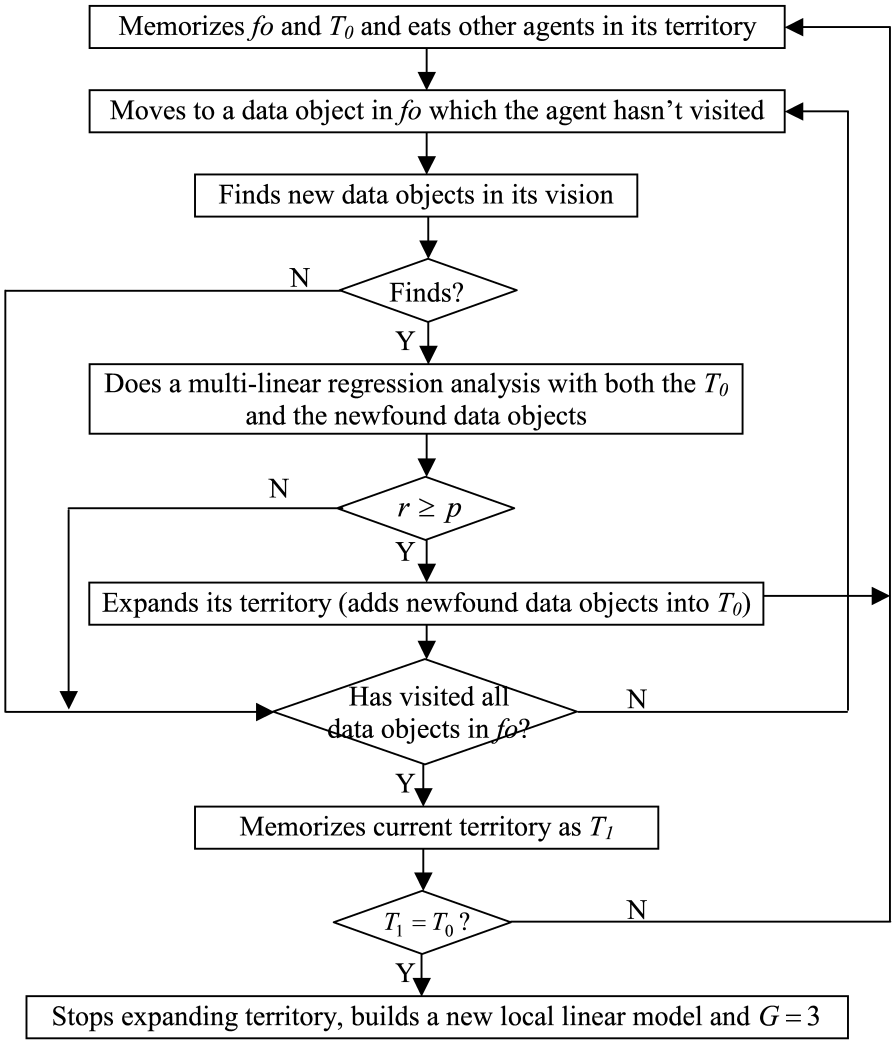

Fig. 5. The process of expanding territory.

An agent finds the frontier data objects of its territory in the following way: If the number of all data objects in its territory is M, then there is a distance matrix:

$$D = \begin{bmatrix} D(O_1, O_1) & \cdots & D(O_1, O_M) \\ \vdots & \vdots & \vdots \\ D(O_M, O_1) & \cdots & D(O_M, O_M) \end{bmatrix}.$$

The agent ransacks D and finds the biggest distance values. The data objects that form these values are marked as frontier data objects by the agent. For example, if there are 5 data objects in one agent's territory,

and the distance matrix D is like the following,

$$D = \begin{bmatrix} 0 & 2.3 & 3.4 & 2.7 & 1.8 \\ 2.3 & 0 & 2.6 & 4.2 & 2.0 \\ 3.4 & 2.6 & 0 & 3.4 & 4.2 \\ 2.7 & 4.2 & 3.4 & 0 & 1.6 \\ 1.8 & 2.0 & 4.2 & 1.6 & 0 \end{bmatrix},$$

we can see that both $D(O_2, O_4)/D(O_4, O_2)$ and $D(O_3, O_5)/D(O_5, O_3)$ are the maximal distances ($= 4.2$) in the matrix D, then the agent will mark data objects O_2, O_3, O_4 and O_5 as frontier data objects.
Since $D(O_i, O_j) = D(O_j, O_i)$, a distance matrix D is symmetric. So an agent only need ransack half of it to find the biggest distance values.

- Step 2: The agent moves to a frontier data object which it hasn't visited and looks for new data objects in its vision. If it finds new data objects, it goes to step 3; otherwise it goes to step 4.
- Step 3: The agent does linear regression analysis with the data objects in T_0 and the newfound data objects. If the correlation coefficient $r \geq p$, the agent adds the newfound data objects into its territory and goes back to step 1; otherwise it goes to step 4.
- Step 4: If there are other frontier data objects in list fo that the agent has not visited, it goes to step 2; if the agent has visited all the data objects in the list fo, it memorizes its current territory as T_1 and goes to step 5.
- Step 5: If $T_1 = T_0$, the agent goes to step 6; otherwise it goes to step 1.
  $T_1 = T_0$ denotes that the agent has visited all frontier objects for expanding its territory, but its territory still remains the same, i.e., it can not expand its territory any more.
- Step 6: The agent stops expanding its territory and evolves into a grade-3 agent, i.e.,

$$G = 3.$$

This progression is shown in Fig. 5.

All grade-2 agents operate at the same time. The survivors are those that not only build their local linear models faster and better but also expand their territories more efficiently.

2.5 Forming the global model

When there are only grade-3 agents left in the data space, they begin to build the global model that is formed by integrating local linear models. Here the integration is done by introducing membership functions (Fuzzy model). Fig. 6 shows a simple example with two local linear models [9]. For the first local linear model (or rule: x is small), the membership function is $\mu_1(x)$; for the second local linear model (or rule: x is large), the membership function is $\mu_2(x)$. For a certain x value $x = x^*$, if $\mu_1(x^*) = w_1$, $\mu_2(x^*) = w_2$, then the estimation of y is:

$$y^* = \frac{w_1 f_1(x^*) + w_2 f_2(x^*)}{w_1 + w_2}. \tag{5}$$

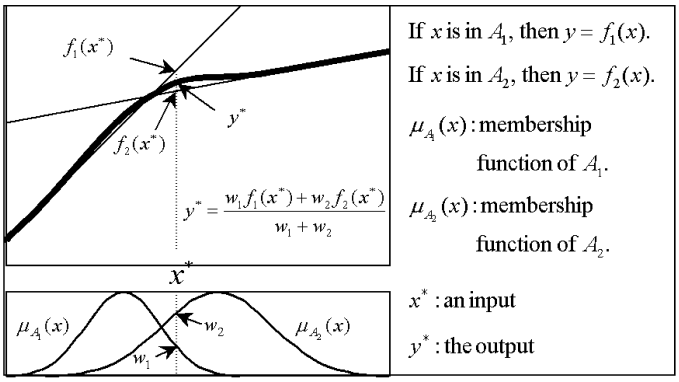

Fig. 6. An example of fuzzy model [9].

If there are H grade-3 agents left in the data space and each of them holds a local linear model f_i, these H agents can be denoted in set $\{B_1, \cdots, B_H\}$. Suppose that $\mu_i(O)$ is the data objects' membership function for f_i, then the global model is:

$$y = \frac{\sum_{i=1}^{H} \mu_i(O) f_i}{\sum_{i=1}^{H} \mu_i(O)}. \tag{6}$$

When using this approach, as in the case of distance definition, users can define various membership functions according to their informed knowledge and experience. In the following tests (Section 3), the membership function $\mu_i(O)$ is simply defined as:

$$\mu_i(O) = \frac{r_i}{e^{d_i(O)}}, \tag{7}$$

where r_i is the correlation coefficient of the local linear model built by agent B_i; $d_i(O)$ is the distance between data object O and agent B_i, defined as the average distance between data object O and the data objects in agent B_i's territory. The r_i indicates the quality of the linear model. The larger the r_i, the higher the linearity. The purpose of introducing r_i in Eq. (7) is to let a better local linear model play a more important role in forming the global model. It is possible that the $d_i(O)$ is very small (very close to zero), so we use it as an exponent of e ($e \approx 2.87$).

3 Testing the Approach

3.1 Tests on three data sets

To test the approach, we applied it to the following three data sets, all available at the UCI machine learning repository [10]. The performance of the approach will be compared with that of a backpropagation neural network which is programmed into a commercial software program named Neunet [1].

- **1985 Auto Imports Data.** This database was created by Jeffrey C. Schlimmer in 1987. There are 205 records (data objects) of different automobiles; each data object is composed of 26 numerical attributes and 10 categorical attributes. This data set has been used as a sample for the software Neunet. The sample used 13 attributes (6 numerical attributes and 7 categorical attributes) to predict the price of automobiles. In the test to compare our approach with the neural network in Neunet, we predicted the price of automobiles using the same attributes as in the sample. Eight data records lacked data in the 13 attributes. In this paper we do not consider the situation of missing data, so these 8 data records were deleted from the original data set, with 197 data records remaining.
- **Auto-Mpg Data.** This dataset was taken from the StatLib library which is maintained at Carnegie Mellon University, and it was created on July 7, 1993. The data set has 398 instances (data records) and 9 attributes:
  (1) MPG: continuous
  (2) cylinders: multi-valued discrete
  (3) displacement: continuous
  (4) horsepower: continuous
  (5) weight: continuous

[1] Neunet provides a free level-1 registration which allows users to create neural networks using up to 250 rows of data (available at: http://www.cormactech.com/neunet/). All the tests we carried out were based on data sets up to 250 records.

(6) acceleration: continuous
(7) model year: multi-valued discrete
(8) origin: multi-valued discrete (Japan, Europe, USA)
(9) car name: string (unique for each instance)

The 6 records which had missing values were deleted before carrying out the test. For comparison with Neunet, we used only the first 250 data records after deleting the 6 records. In this test, we predicted automobiles' "MPG" (miles per gallon), using all the other attributes except "car name". The first seven attributes were treated as numerical data, with the "origin" attribute as categorical data.

- **Relative CPU Performance Data.** This dataset was created by Phillip Ein-Dor and Jacob Feldmesser in October 1987. It includes 209 instances with the following 9 attributes:
  (1) vendor name: 30 nominal values
  (2) model name: unique symbols
  (3) MYCT: machine cycle time in nanoseconds
  (4) MMIN: minimum main memory in kilobytes
  (5) MMAX: maximum main memory in kilobytes
  (6) CACH: cache memory in kilobytes
  (7) CHMIN: minimum channels in units
  (8) CHMAX: maximum channels in units
  (9) PRP: published relative performance

  In this test we predicted the "PRP" attribute, using all of the other attributes except the "model name", which has unique values for each instance. The "vendor name" attribute was treated as categorical data, with the others as numerical data.

We used Root-Mean-Square Error ($RMSE$) to evaluate the performance of the prediction models. The smaller the $RMSE$, the better the performance of the model. In Eq. 8, y_i and $\hat{y}_i$ denote the real value and the predicted value respectively. N_t is the size of the testing set. For example, if $N_t = 10$, then the final 10 data records are used as testing data, while the rest of the data set is used as training data.

$$RMSE = \sqrt{\frac{1}{N_t}\sum_{i=1}^{N_t}(\hat{y}_i - y_i)^2}. \tag{8}$$

Fig. 7, Fig. 8, and Fig. 9 show the results obtained on the above three databases. The backpropagation neural network programmed in Neunet was trained for 5×10^4 times for each N_t before making the prediction. In Fig. 7, the $RMSE$ of the agent-based approach is smaller than that of the neural network for each N_t; that is to say, for the Auto Imports Data, the agent-based approach has better performance than the neural network. In Fig. 8, the neural network shows better performance for most N_ts: the $RMSE$ of

the neural network is smaller than that of the agent-based approach when $N_t = 10, 20, \cdots, 70$, while it is bigger when $N_t = 80, 90, 100$. In Fig. 9, the $RMSE$ of the agent-based approach is smaller than that of the neural network for each N_t, i.e., for the Relative CPU Performance Data, the agent-based approach also has better performance than the neural network in terms of $RMSE$.

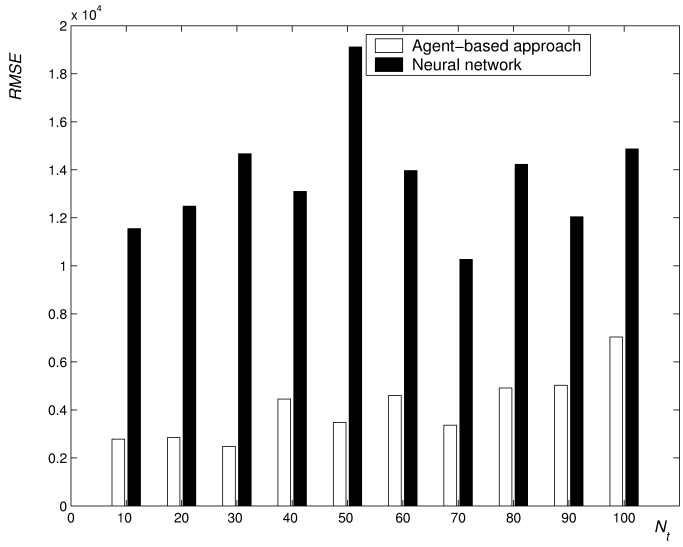

Fig. 7. Result on Auto Imports Data.

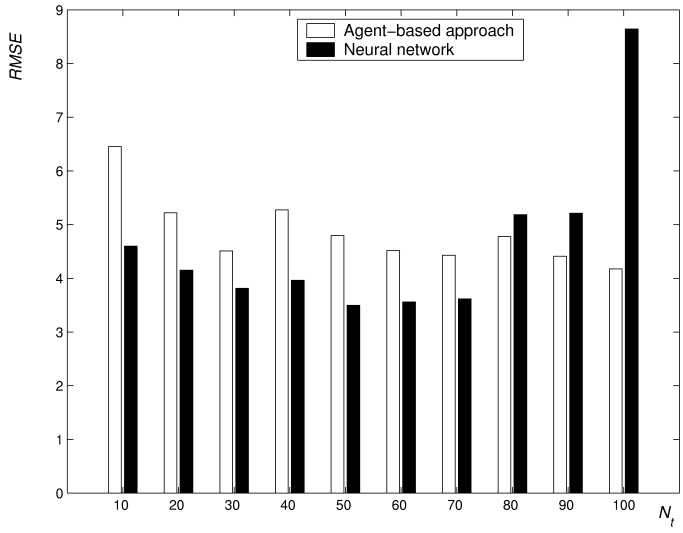

Fig. 8. Result on Auto-Mpg Data.

"No single method has been found to be superior over all others for all data sets" [4], or as Y. Nakamori and M. Ryoke pointed out, "It is impossible in principle, due to the nature of the data, to specify a criterion and procedure to obtain an ideal fuzzy model" [6]. (The global models identified by the agent-based approach are, in fact, fuzzy models, as discussed in subsection 2.5.) Keeping those arguments in mind, we did not expect the agent-based

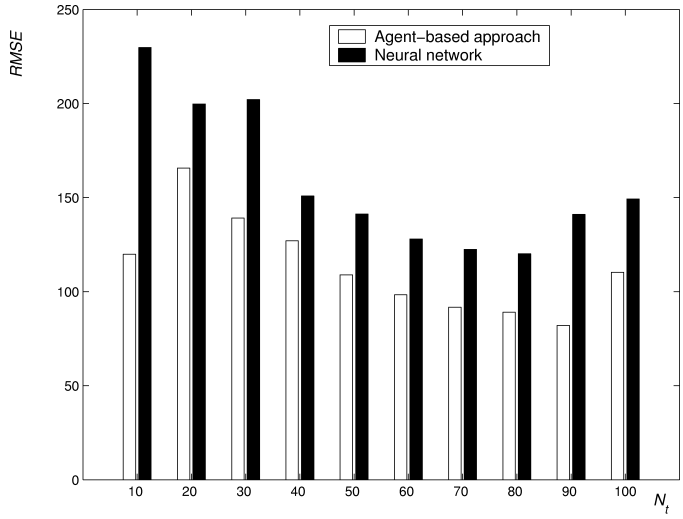

Fig. 9. Result on Relative CPU Performance Data.

approach to make better predictions than other methods for all data sets. The results of the three tests show that the agent-based approach can achieve good performance for some data sets. Since we used linear models as local models, the approach has better performance when it deals with data which have good local linearity.

3.2 Sensitivity Analysis of Design Parameters

Two design parameters, p and q were introduced in subsection 2.3. The p denotes the threshhold of correlation coefficients of good local linear models, i.e., a local linear model with a correlation coefficient not smaller than the p is considered to be of enough linearity (or simply speaking, it is a good one). The q denotes the step size with which a grade-1 agent decreases its vision for building a good local linear model. In the above three tests, the value of p was set to be 0.88 since we thought a local linear model with its correlation coefficient not smaller than the 0.88 is a good one, and the value of q was set to 10 since we believed that it was reasonable to give a grade-1 agent 10 chances to build a local linear model by decreasing its vision. The values were set depending on our informed intuition. In this subsection, we will give the sensitivity analysis of these two design parameters with the three data sets introduced in subsection 3.1.

First, we give the sensitivity analysis of p. For the three data sets introduced in the sub-section 3.1, we vary the value of p from 0.98 to 0.78 for each N_t, with 0.02 as the interval and $q = 10$. Table 3 shows the result on Auto Imports Data. We can roughly see that 0.84, 0.86 and 0.88 are good values for p in terms of $RMSE$. Similar tables can be drawn based on the Auto-Mpg Data and the

Table 3. $RMSE$s with different values of p – Auto Imports Data

	0.98	0.96	0.94	0.92	0.90	0.88	0.86	0.84	0.82	0.80	0.78
10	3872	2182	3401	2689	3628	2782	3058	3350	3451	3730	3152
20	3668	2409	3052	2354	2794	2848	3289	3142	2988	3127	4347
30	4582	4387	3641	4557	3495	2482	3251	3702	3937	4081	3381
40	6832	7802	7116	7469	4420	4456	4530	3326	2979	7816	5373
50	34224	32990	31276	64560	70681	3480	2845	2664	4153	4453	4857
60	32960	31520	29765	57132	60563	4601	2833	2231	2626	3695	6600
70	4289	5398	6237	5683	3288	3360	3381	3427	4462	6186	6186
80	7031	7305	6630	9473	5052	4914	4677	5430	5802	5894	6259
90	5538	5872	5535	5995	4845	5029	5058	5491	5801	5954	5954
100	6721	5533	6909	7181	7664	7035	7343	7343	4149	7455	7455

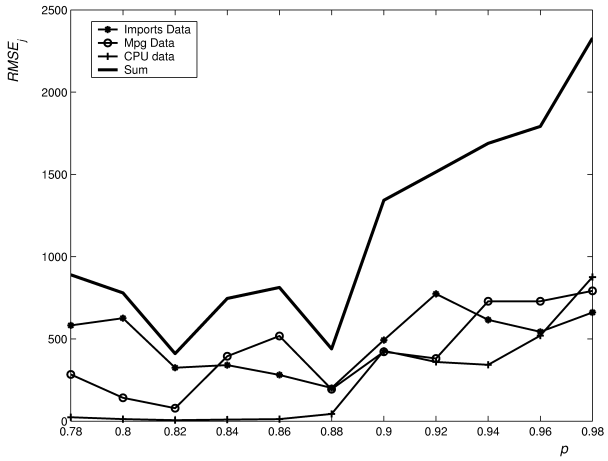

Fig. 10. $RMSE_j$ for each of the three data sets.

Relative CPU Performance Data. Users can employ statistical techniques to more precisely define good values for p. Instead of doing this, this paper will establish a figure to demonstrate the sensitivity of p with the three data sets. To get a figure which is easy to understand, we normalize the $RMSE$ for each N_t into the range [0, 100] according to Eq. 9.

$$RMSE_j^{i\prime} = \frac{(RMSE_j^i - RMSE_{\min}^i) \times 100}{RMSE_{\max}^i - RMSE_{\min}^i}, \tag{9}$$

where $i = 10, 20, \cdots, 100$ and $j = 0.98, 0.96, \cdots, 0.78$.

Then we sum all $RMSE$s for each p according to Eq. 10.

$$RMSE_j = \sum_{i=10}^{100} RMSE_j^{i'}. \tag{10}$$

Fig. 10 shows the $RMSE_j$ for each of the three data sets, with p varied from 0.78 to 0.98. We can see that for the Auto Imports Data, 0.88 is the best value for p in terms of $RMSE_j$; for the Auto-Mpg Data, 0.82 is the best; and for the Relative CPU Performance Data, 0.82 is the best. Summarizing the results on the three data bases, the authors suggest that p's value be set in the range [0.8, 0.9].

For the sensitivity analysis of q, we varied the value of q from 0.02 to 0.20 for each N_t, with 0.02 as the interval and $p = 0.88$. We found that the results are exactly the same as $q = 10$.

Based on the above analysis, here we suggest two ways for setting the values of the design parameters p and q. The first way is to set the values as suggested in this paper, i.e., to select a value from [0.8, 0.9] for p and set $q = 10$. The second way is to set the values based on the sensitivity analysis for each data sets, i.e., to find different values for different data sets. For example, users can find good values for p and q using a neural network in the second way.

4 Conclusion and Remarks

This paper presented an agent-based approach to the identification of fuzzy prediction models for multi-dimensional complex data, inspired by the discipline of complex adaptive systems and survival of the fittest in nature. Each individual agent tries to build its local model, competing with other agents for survival, then all surviving agents cooperate with each other to form the global model by introducing membership functions. Three tests were carried out and the agent-based approach showed good performance in terms of $RMSE$.

In this paper, linear models were selected as local models. For data with low linearity, the authors suggest that non-linear local models could improve the performance of the approach. Furthermore, agents in the approach could be equipped with more intelligence; for example, they could have the ability to build not only linear local models but also non-linear local models, and they could decide what kind of local models (linear or non-linear) should be built, depending on different situations. Different techniques to identify membership functions could be applied; for example, a genetic algorithm was used when we applied the approach to two-dimensional numerical data [5].

To date, the approach has only been used to predict continuous values. In the future work, it will be developed for predicting discrete or nominal values. This approach complements rather than competes with existing Soft Computing methods.

Acknowledgements

We appreciate the help of Ms. Judith Steeh in reading the manuscript. We got a lot of good advice from Mr. Wei Huang in the research. We owe him many thanks.

References

[1] R. Becker, O.E. Holland and J.L. Deneubourg, From Local Actions to Global Tasks: Stigmergy and collective robotics, Artificial Life IV, (Eds. R. Brooks and P. Maes,) MIT Press, 1994.

[2] E. Bonabeau, M. Dorigo, G. Theraulaz, Inspiration for Optimization from social insect behavior, Nature, vol 406, no 6, 2000.

[3] E. Bonabeau, M. Dorigo, G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press, New York, 1999.

[4] J. Han and K. Micheline, Classification and Prediction, Data Mining – Concepts and Techniques., Morgan Kaufmann Publishers, 2000, pp. 279-333.

[5] T. Ma and Y. Nakamori, An Agent-Based Approach to Identification Prediction Models, The International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, World Scientific, Vol. 11, No. 4, 2003, pp. 495-514.

[6] Y. Nakamori and M.Ryoke, "Identification of Fuzzy Prediction Models Through Hyperellipsoidal Clustering", IEEE Transaction on Systems, Man, and Cybernetics, vol. 24, No. 8, August 1994.

[7] N.K. Sinha, M.M. Gupta and Lotfi A. Zadeh (Eds.), Soft Computing and Intelligent Systems: theory and applications, Academic Press, 2000.

[8] T. Stutzle and H. Hoos, MAX-MIN Ant Systems, Future Generation Computer Systems, vol 16, 2000, pp. 889-914.

[9] T. Takagi and M. Sugeno, "Fuzzy identification of Systems and its applications to modeling and control", IEEE Transaction on Systems, Man, and Cybernetics, vol. SMC-15, No. 1, pp.116-132, 1985.

[10] UCI machine learning repository, available at: http://www.ics.uci.edu/∼mlearn/MLRepository.html.