Section 6. Neural and Growing Networks

6.1. Neural Network Applications

PARALLEL MARKOVIAN APPROACH TO THE PROBLEM OF CLOUD MASK EXTRACTION

Natalia Kussul, Andriy Shelestov, Nguyen Thanh Phuong, Michael Korbakov, Alexey Kravchenko

***Abstract**: An application of Markovian approach to cloud mask extraction is presented. Also parallel algorithm of Markovian segmentation is considered.*

***Keywords**: Meteosat, cloud mask, Markov Random Fields, parallel programming, MPI.*

Introduction

One of the most useful satellite data products is cloud mask. It can be used in a standalone way in applications such as air flights and satellite photography planning. Also it can be used as an input data for various satellite data processing algorithms like Normalized Difference Vegetation Index (NDVI), Sea Surface Temperature (SST) and operational wind vectors maps extraction, or even more complex applications such as numerical weather models.

A common approach for cloud mask extracting is using of multi- and hyperspectral satellites providing data in many spectral bands. Basing on information about radiance intensities a conclusion about cloudiness can be made on per pixel basis. For instance, this approach is widely used for processing of multispectral AVHRR and MODIS data. But temporal resolution of satellites with such equipment on-board is usually quite low thus making impossible on-line monitoring of particular region of Earth disk. The obvious solution of this problem is the use of geostationary satellites.

For European region, the only geostationary satellite that can be used for solving cloud mask extraction problem is Meteosat. Meteosat is operated by EUMETSAT international organization and provides data for solving practical meteorological problems. This satellite's onboard equipment makes one image of earth disk in 30 minutes in three spectral bands – visible, infrared, water vapour.

The temporal characteristics of Meteosat data make their use in solving of problem of cloud mask extracting quite actual. But three spectral bands of Meteosat do not provide enough information for multispectral cloud recognition algorithms operating on per pixel basis. This causes the need for algorithms, which involves temporal and spatial dependencies in data processing. One of such algorithms is a Markov Random Field segmentation, which allows determining pixel's class with regard to its neighborhoods. Markovian approach allows taking into account different possible distributions of intensities per class and do not introduce global parameters such as thresholds, which is often used in multispectral data processing.

One of the main disadvantages of Markovian approach is high computational complexity. Therefore processing of large images requires great amount of time. So, for effective solving of Markovian segmentation problem a parallel computing approach is needed.

Data Preprocessing

Meteosat images come with a lot of noise of two sorts. The first one is the so-called “salt and pepper” noise consisting of noisy pixels uniformly distributed over image. The second one is the impulse burst noise, which distorts images with horizontal streaks of few pixel heights filled with white noise.

In the Space Research Institute NASU-NSAU algorithm for detecting and removing such noise was developed [Phuong, 2004]. On the first step of this algorithm noise streaks is detected and removed by cubic spline interpolation method. During the second step the “salt and pepper” noise is detected by modified median filter and removed by using bit planes approach. The algorithm based on this approach separates an image in 256 planes with binary values. After that, each of these planes is processed separately in order to remove noise.

Cloud Mask Extraction

Following Markovian approach the image is represented as a $n \times m$ matrix of sites S. The neighborhood of site s_{ij} is any subset $\partial_{ij} \subset S$, such that $s_{ij} \notin \partial_{ij}$. With each site s_{ij} two varieties are associated – an intensity X_{ij} (as usual it takes integer value in interval [0; 255]) and a hidden label Y_{ij}. The specific values the varieties take are denoted x_{ij} and y_{ij} respectively. So two sets of varieties defined for image S: $X = \{X_{11},...,X_{nm}\}$ and $Y = \{Y_{11},...,Y_{nm}\}$.

Markov Random Fields (MRFs) are widely used for image segmentation [Li, 1995]. With the Hammersley-Clifford theorem the equivalence of MRF and statistical physics Gibbs models was proved [Li, 1995]. This theorem gives us the equation for probability of specific segmentation $P(Y)$

$$P(Y) = \frac{1}{z} e^{\beta V(Y)} = \frac{1}{z} e^{\beta \sum_{i,j} V_{ij}\left(\partial_{ij} \cap \{y_{ij}\}\right)}$$

In this equation, z is normalizing constant necessary for holding the condition $\sum_{Y} P(Y) = 1$. β denotes the image correlation parameter. V is the so-called potential function. Its structure is highly coupled with optimal segmentation of MRF. Defining a particular potential function it is possible to model physics features of segmentation. The right part of equation shows that potential function V can be represented as a sum of potentials defined at each site: $V = \sum_{i,j} V_{ij}$.

For the cloud mask extraction problem the following Markovian model was used: the observed intensity X_{ij} depends only on local label Y_{ij} and a conditional distribution of variety X_{ij} is Gaussian. The Bayes’ theorem about a priory and a posteriori probabilities’ relation yields a complete model of intensities and labels coupling [Shiryaev, 1989]:

$$P(Y \mid X) \propto P(X \mid Y)P(Y) = \frac{1}{z} \exp\left\{\sum_{i,j} \beta_{ij} n_{ij}\left(y_{ij}\right)\right\} \times \prod_{i,j} \frac{1}{\sqrt{2\pi}\sigma_{ij}} \exp\left\{-\frac{1}{2\sigma_{ij}^2}\left(x_{ij} - \mu_{ij}\right)^2\right\}$$

Here σ_{ij} and μ_{ij} are a standard deviation and a mean of random variable X_{ij}, n_{ij} is the number of pixels in neighborhood ∂_{ij} with the label equal to Y_{ij}.

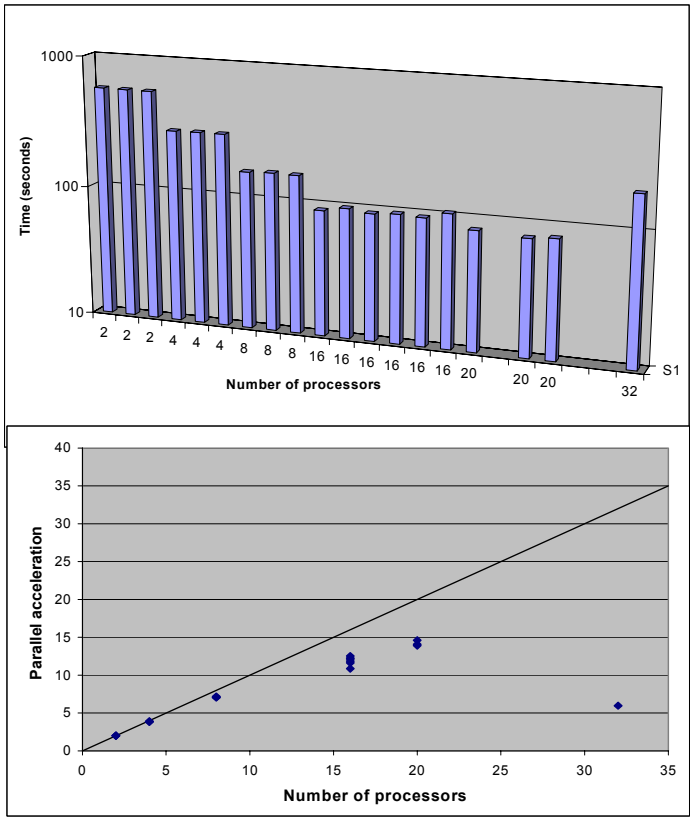
The goal of segmentation is to maximize $P(Y \mid X)$ under particular intensities X. This corresponds to obtaining maximum for a posteriori label’s estimate: $Y : Y^* = \arg\max_y \{P(Y \mid X)\}$

Parallel Execution Results

High computational complexity of Markovian segmentation algorithm together with large sizes of satellite images determines the need for parallel realization of cloud mask extraction process.

Meteosat image filtering and Markovian segmentation algorithms were implemented using MPI parallel programming interface [MPI, 1997]. Due to locality of dependencies in Markovian image model, it is possible to divide image into almost independent rectangular parts. Then each of these parts is processed by different computational node. Synchronization of several global per-class parameters and image part's borders is performed by means of MPI's group communication functions.

The program was run on the cluster of Institute of Cybernetics NASU consisting of 32 Intel Xeon processors. It has demonstrated good level of parallel acceleration giving almost proportional speed boost with increase of number of computational nodes used.



Conclusions and Further Works

Markovian approach has showed its effectiveness in task of cloud mask extraction from Meteosat satellite data. Also parallel Markovian segmentation algorithm performed very well exploiting locality of Markovian image model.

Further works includes implementation of this algorithm in GRID environment, which will connect computational cluster with satellite data archives and method to process them. This GRID design will allow separating algorithms from environment and using them in standalone applications locally. Also, we are planning to port this algorithm implementation to IA-64 processor architecture to utilize upcoming computational cluster with 64 Itanium-2 processors.

Bibliography

[Phuong, 2004] N. T. Phuong. A Simple and Effective Method for Detection and Deletion of Impulsive Noised Streaks on Images. Problems of Controls and Informatics, 2004. (in Russian)

[Li, 1995] S.Z. Li. Markov Random Field Modelling in Computer Vision. Springer-Verlag, 1995.

[Shiryaev, 1989] A. N. Shiryaev. Probability. Nauka, 1989. (in Russian)

[MPI, 1997] MPI-2: Extensions to the Message-Passing Interface. University of Tennessee, 1997

Authors' Information

Natalia Kussul – e-mail: inform@ikd.kiev.ua
Andriy Shelestov – e-mail: shelest@dialektika.com
Nguyen Thanh Phuong – e-mail: ntphuong72@yahoo.com
Michael Korbakov – e-mail: rmihael@ukr.net
Alexey Kravchenko – e-mail: akm3000@mail.ru
Space Research Institute NASU-NSAU, Glushkova St. – 41, Kyiv

ИДЕНТИФИКАЦИЯ НЕЙРОСЕТЕВОЙ МОДЕЛИ ПОВЕДЕНИЯ ПОЛЬЗОВАТЕЛЕЙ КОМПЬЮТЕРНЫХ СИСТЕМ[1]

Н. Куссуль, С. Скакун

***Аннотация**: В работе проводилось математическое моделирование поведения пользователей компьютерных систем. Изучалась динамика работы пользователя во время сеанса. Также осуществлялось статистическое моделирование данных, характеризующих его работу за сеанс в целом.*

***Ключевые слова**: модель поведения пользователей, нейронные сети, компьютерные системы.*

1. Введение

Масштабное использование компьютерных технологий практически во всех сферах человеческой деятельности приковывает все большее внимание к самому пользователю. Знание того, какие действия он выполняет (или должен выполнять), может применяться в разных областях, например, в системах безопасности [1], для создания персонализированного окружения для пользователей [2] и так далее.

В работе [3] была предложена комплексная модель пользователя, состоящая из интерактивной и сеансовой частей, которые учитывают, соответственно, динамические и статистические свойства поведения пользователя. В обеих моделях для выявления отклонений от обычного или ожидаемого поведения пользователей используются нейронные сети. Так, интерактивная модель основана на прогнозировании команд пользователя на основе предыдущих (В данной работе под прогнозированием команд будем понимать прогнозирование процессов, порожденных запуском файлов ОС Windows.). Поскольку выбор архитектуры нейронной сети представляет собой нетривиальную задачу, важно знать, на сколько его текущее поведение зависит от предыстории. В случае сеансовой модели возникает проблема с размером выборки, которая используется для обучения нейронной сети. Дело в том, что при небольшом размере обучающего множества нейронная сеть имеет тенденцию к локальному запоминанию образов, что нежелательно. В сеансовой модели на вход нейронной сети подаются данные, собранные за сеанс в целом. Соответственно, размер обучающего множества напрямую определяется количеством сеансов, во время которых проводилось наблюдение за деятельностью пользователя. Однако даже за продолжительный отрезок времени этих данных будет недостаточно для качественного обучения нейронной сети. Поэтому в сеансовой модели для качественного обучения нейронной сети очень важно обеспечить более представительную выборку данных.

Вопросам, связанными с оптимизацией архитектуры нейронной сети, в частности размерностью входного слоя, и моделированием данных, и посвящена данная статья.

2. Комплексная нейросетевая модель пользователя компьютерных систем

Комплексная модель пользователя, предложенная в работе [3], учитывает как динамические (интерактивная часть), так и статистические (сеансовая часть) свойства поведения пользователей. В основу разработанной модели положена нейронная сеть прямого распространения, которая состоит из входного, выходного и одного или нескольких скрытых слоев нейронов [4]. Выход нейрона в слое n определяется следующим отношением:

$$y_j^n = f(s_j^n), \tag{1}$$

где n — номер слоя ($n = \overline{1, p}$); p — количество слоев в нейронной сети; j — индекс нейрона в слое $(j = \overline{1, N_n})$; N_n — число нейронов в слое; f — активационная функция слоя (в нашем случае

[1] Работа выполнена при содействии гранта Президента Украины для поддержки научных исследований молодых ученых № Ф8/323, "Прототип интеллектуальной мультиагентной системы компьютерной безопасности".

для скрытых слоев используется сигмоидная активационная функция $f(x)=\frac{1}{1+e^{-\alpha x}}$, а для выходного слоя — линейная $f(x)=\alpha x$); $y_j{}^n$ — выход j-го нейрона слоя; $s_j{}^n$ — постсинаптический потенциал j-го нейрона слоя, который вычисляется согласно следующей формуле:

$$s_j{}^n=\sum_{k=1}^{N_{n-1}} W_{jk}{}^n y_k{}^{n-1}+b_j{}^n, \quad S^n=W^n\cdot\tilde{y}^{n-1}, \tag{2}$$

где $W_{jk}{}^n$ — весовой коэффициент связи k-го нейрона слоя $n-1$ с j-м нейроном слоя n; $y_k{}^{n-1}$ — выход k-го нейрона слоя $n-1$; $\tilde{y}^{n-1}$ — расширенный вектор с учетом bias-нейрона; $b_j{}^n$ — порог (bias-нейрон) j-го нейрона слоя n.

Интерактивная модель используется для выявления аномальной деятельности во время работы пользователя. Для каждого пользователя компьютерной системы строится и обучается нейронная сеть таким образом, чтобы прогнозировать следующую команду на основе предыдущих. Пусть s_t (где $t\in\{1, 2, ...\}$ номер сеанса) — некоторый сеанс пользователя, для которого имеется следующая последовательность выполненных им команд:

$$\mathbf{c}^{s_t}=\left(c_1^{s_t},\ c_2^{s_t},\ \ldots,\ c_{N_{s_t}}^{s_t}\right),$$

где N_{s_t} — количество команд, введенных за сеанс s_t. Прежде чем подать последовательность из m команд на вход нейронной сети применялось бинарное кодирование. При этом для каждой команды было использовано q бит. Тогда результат работы нейронной сети при выполнении i-1 команды определяется зависимостью:

$$\tilde{c}_i^{s_t}=F(\mathbf{x}_i),\ \mathbf{x}_i=\left(\tilde{\mathbf{c}}_{i-1}^{s_t},\tilde{\mathbf{c}}_{i-2}^{s_t},\ldots,\tilde{\mathbf{c}}_{i-m}^{s_t}\right), \tag{3}$$

где F — нелинейное преобразование, осуществляемое нейронной сетью согласно формулам (1) и (2); $\tilde{\mathbf{c}}_{i-k}^{s_t}\left(k=\overline{1,m}\right)$ — бинарный вектор для команды $c_{i-k}^{s_t}$; $\mathbf{x}_i$, $\tilde{c}_i^{s_t}$ — вход и выход сети, соответственно (в данном случае размерность вектора $\mathbf{x}_i$ составляет q*m, а $\tilde{c}_i^{s_t}$ — 1); $c_i^{s_t}$ — i-тая команда сеанса s_t; m — количество команд, на основе которых происходит прогнозирование следующей (окно прогнозирования).

На основе количества команд, которые были правильно спрогнозированы нейронной сетью, делается вывод о том, соответствует ли текущее поведение пользователя ранее построенной модели.

Пусть

$$\Theta(i)=\frac{1}{(i-m)}\sum_{j=m+1}^{i}\chi(\tilde{c}_j^{s_t},c_j^{s_t}),\quad \chi(\tilde{c}_j^{s_t},c_j^{s_t})=\begin{cases}1, & \textit{если } \tilde{c}_j^{s_t}=c_j^{s_t}\\ 0, & \textit{в противном случае}\end{cases}. \tag{4}$$

Величина $\Theta(i)$ определяет относительное число верно спрогнозированных команд. Если $\Theta(i)<\Theta'$, т.е. значение $\Theta(i)$ меньше некоторого порога, то поведение пользователя следует считать аномальным, в противном случае — нормальным.

При этом необходимо учитывать, что пользователям свойственно изменять поведение с течением времени, поэтому с целью обеспечения адаптации к их поведению нейронную сеть следует периодически дообучать. (Критерий изменения поведения пользователей будет рассмотрен далее.)

Сеансовая модель предназначена для выявления нехарактерной деятельности пользователя за сеанс в целом и для этого использует статистический набор данных. Эта информация, в свою очередь, используется для построения и обучения нейронной сети, которая определяет, на сколько активность пользователя соответствует ранее построенной модели. При этом ожидаемый выход нейронной сети может принимать два значения: 1 — для нормального поведения пользователя и 0 — для аномального, т.е. нейронная сеть работает в качестве классификатора.

Выход же нейронной сети по завершении сеанса s_t определяется следующим соотношением:

$$\Delta_{s_t}=F(x_{s_t}),\quad \mathbf{x}_{s_t}=(n_{s_t},o_{s_t},h_{s_t},d_{s_t},s_{s_t}), \tag{5}$$

где F — нелинейное преобразование, осуществляемое нейронной сетью согласно формулам (1) и (2); $\mathbf{x}_{s_t}$, Δ_{s_t} — вход и выход сети, соответственно (в данном случае размерность вектора $\mathbf{x}_{s_t}$ составляет 5, а Δ_{s_t} — 1); n_{s_t} — количество команд за сеанс, $o_{s_t} = \Theta(N_{s_t})$ — результаты интерактивной модели (процентное соотношение правильно спрогнозированных команд за весь сеанс), h_{s_t} — номер компьютера, d_{s_t} — продолжительность сеанса, s_{s_t} — время начала сеанса.

Следует заметить, что выход нейронной сети Δ_{s_t} не обязательно будет равным 0 или 1, а будет принадлежать отрезку [0; 1] и определять вероятность нормального (соответствующего модели) поведения пользователя.

3. Описание данных

Для моделирования поведения пользователей компьютерных систем использовались реальные данные, которые были собраны в локальной сети Института космических исследований НАНУ-НКАУ в период за два-три месяца. В данной сети рабочие станции функционировали под управлением операционных систем (ОС) Windows 98, XP, 2000. Поскольку эти ОС необходимой информацией об активности пользователя обеспечивали не в полной мере, было разработано специальное программное приложение. Для каждого сеанса пользователя создавался отдельный аудит-файл (название которого однозначно определяло имя учетной записи пользователя и дату его работы), в который сохранялась информация в следующем формате:

время запуска команды|*идентификатор команды*|*название команды*|*флаг начала или завершения*

Необходимо отметить, что идентификатор команды присваивается ОС и является уникальным (для команд с одним и тем же именем он различен, причем от сеанса к сеансу он также меняется). Поэтому при кодировании команд важно обеспечить, чтобы одинаковым командам соответствовали одни и те же значения. С этой целью для интерактивной части комплексной модели на основе собранной информации для каждого пользователя был построен алфавит команд *A* (т.е. набор команд, которые вводились пользователем на протяжении указанного периода времени). Далее каждой команде был присвоен соответствующий десятичный номер, который впоследствии использовался при преобразовании аудит-файлов в последовательности команд ($A = \{1, 2, \ldots, N\}$). В результате для каждого пользователя был представлен следующий набор данных:

$$\left\{c_i^{s_t}\right\}_{t=1 i=1}^{T \quad N_{s_t}}, \tag{6}$$

где $c_i^{s_t} \in A$ — десятичный номер *i*-ой введенной команды сеанса s_t; T — количество сеансов; N_{s_t} — общее количество выполненных команд за сеанс s_t. На рис. 1 приведен пример последовательности команд, вводимых пользователем за один сеанс.

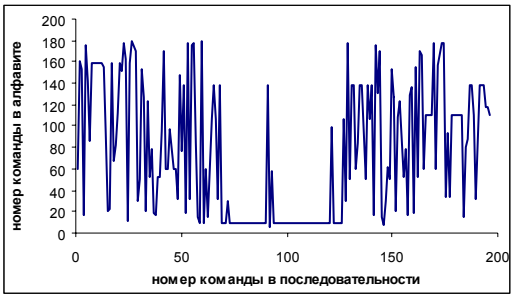


Рис. 1. Пример последовательности команд, вводимых пользователем за один сеанс

В свою очередь, для сеансовой части комплексной модели был получен следующий набор данных, используя информацию из аудит-файлов: $\left\{n_{s_t}, h_{s_t}, d_{s_t}, s_{s_t}\right\}_{t=1}^{T}$, где t — условный номер сеанса; T — количество сеансов; параметры $n_{s_t}, h_{s_t}, d_{s_t}, s_{s_t}$ определены в соотношении (5).

4. Изучение динамики поведения пользователя во время сеанса

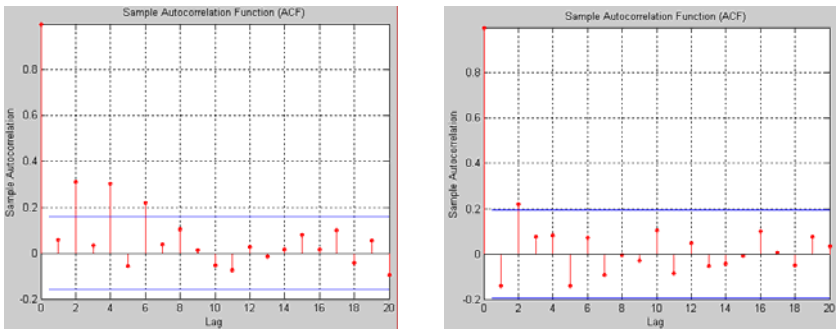
При исследовании динамики поведения пользователя по последовательностям вводимых им команд, нас, в первую очередь, будет интересовать решение следующих задач: определение количества команд, которые следует использовать для прогноза следующей, а также построение критерия изменения поведения пользователя с целью уменьшения ложных тревог и переобучения нейронных сетей.

4.1. Построение автокорреляционных функций

Поскольку интерактивная модель основана на прогнозировании нейронной сетью команд пользователя, важно знать, на сколько его поведение в данный момент времени зависит от предыдущего. Для этого для каждого сеанса пользователя s_t были построены автокорреляционные кривые, определяемые соотношениями следующего вида:

$$\rho_{s_t}(n) = \frac{1}{N_{s_t} - n} \sum_{i=1}^{N_{s_t}-n} \left(c_i^{s_t} - \mu_{s_t}\right)\left(c_{i+n}^{s_t} - \mu_{s_t}\right),\ \mu_{s_t} = \frac{1}{N_{s_t}} \sum_{i=1}^{N_{s_t}} c_i^{s_t}\ .$$

где $\rho_{s_t}(n)$ — коэффициент корреляции для последовательности команд, введенных за сеанс s_t; n — значение лага (временное смещение между элементами последовательности). На рис. 2 приведены примеры автокорреляционных кривых для разных пользователей.



пользователь № 1 пользователь № 2

Рис. 2. Примеры автокорреляционных функций для разных пользователей

Анализ построенных кривых показывает, что с ростом числа лагов автокорреляционные функции убывают. При этом экстремумы наблюдаются при значениях лагов от 1 до 5. Таким образом, при прогнозе команд пользователя следует использовать именно это количество команд. При этом необходимо учитывать следующее: использование слишком большого количества команд приведет к тому, что на протяжении этого периода времени будет невозможно осуществлять прогноз команд, что снизит возможности по выявления аномальной деятельности пользователей.

4.2. Построение критерия изменения поведения пользователя

Рассмотрим теперь вопрос, связанный с построением критерия для определения изменения поведения пользователя компьютерных систем. Это позволит уменьшить количество ложных тревог, связанных с прогнозированием действий пользователя, и определить момент, когда следует переобучать нейронные сети.

При построении интерактивной модели пользователя необходимо учитывать, что с течением времени поведение пользователя меняется. Это может происходить по разным причинам. Например, пользователь установил новое программное обеспечение и начал часто его запускать, а другие программные продукты он стал использовать намного реже. Тогда в этом случае точность прогноза, осуществляемого нейронной сетью, снизится и такое поведение, соответственно, будет интерпретироваться как аномальное. Поэтому при построении интерактивной модели важно знать, что поведение пользователя изменилось и нейронную сеть для него следует переобучить.

Однако с этим необходимо заметить, что аномальное поведение также можно интерпретировать как изменение поведения. Поэтому при построении критерия будем исходить из следующих предположений:

- Будем считать, что аномальное поведение пользователя характеризуется резкими изменениями и скоротечностью;
- В свою очередь, естественное изменение поведения происходит на протяжении нескольких сеансов и не характеризуется резкими перепадами.

Перейдем теперь к формулировке критерия изменения поведения пользователя.

Пусть A — алфавит команд некоторого пользователя (т.е. набор всех команд, которые выполнялись на протяжении некоторого времени T). Предполагается, что за этот промежуток времени поведение пользователя не изменялось. Обозначим посредством $|A| = N$ — количество команд в алфавите. При этом будем считать, что каждой команде присвоен номер от 1 до N, причем номер N зарезервирован за командами, которые отсутствуют в алфавите. Пусть s_t ($t>T$) — текущий сеанс пользователя, для которого имеется следующая последовательность выполненных им команд:

$$c^{s_t} = \left(c_1^{s_t}, c_2^{s_t}, \ldots, c_{N_{s_t}}^{s_t}\right).$$

Для того чтобы определить, изменилось ли поведение пользователя, по окончании сеанса s_t строится вектор $g(s_t)$, компоненты которого определяются таким образом:

$$g_j(s_t) = \begin{cases} 1, & \text{если существует такое } k = 1,\ N_{s_t}, \text{ что } c_k^{s_t} = j,\ j \in A \\ 0, & \text{в противном случае} \end{cases}. \qquad (7)$$

Т.е. если определенная команда была выполнена во время сеанса s_t, то значение соответствующей компоненты вектора $g(s_t)$ становится равным 1, в противном случае — 0. Затем этот вектор попарно сравнивается с аналогичными векторами, построенными для предыдущих сеансов s_{t-1}, s_{t-2}, …, s_{t-l} и т.д. (в данной работе l = 5). В качестве меры сравнения использовалось расстояние Хэмминга:

$$\aleph(g(s_{t'}), g(s_{t''})) = \sum_{j=1}^{N} \chi(g_j(s_{t'}), g_j(s_{t''})),\ \chi(g_j(s_{t'}), g_j(s_{t''})) = \begin{cases} 1, \text{если } g_j(s_{t'}) \neq g_j(s_{t''}) \\ 0, \text{в противном случае} \end{cases}. \qquad (8)$$

Т.е. величина $\aleph$ определяет число компонент двух векторов, значения которых отличны. Получив в результате сравнения l значений, вычисляем среднее и делим его на общее количество команд в алфавите:

$$H_t = \frac{1}{N}\left(\frac{1}{l}\sum_{k=1}^{l} \aleph(g(s_t), g(s_{t-k}))\right). \qquad (9)$$

Если поведение пользователя не изменилось и не аномально, тогда вектор $g(s_t)$ будет отличаться от предшествующих незначительно. Соответственно, значение параметра H_t будет небольшим (меньше некоторого порога H_*). И наоборот, если наблюдается аномальное поведение пользователя, вектор $g(s_t)$ будет значительно отличаться от всех остальных и значение параметра H_t будет большим (больше некоторого порога H^*). Если же значение H_t принадлежит интервалу (H_*; H^*), можно говорить о естественном изменении поведения пользователя.

Таким образом, критерий изменения поведения пользователя можно сформулировать в следующем виде: *для данного (текущего) сеанса пользователя s_t на основе формул (7)-(9) вычисляется значение параметра H_t. Если $H_t < H_*$, считается, что поведение не изменилось, если $H_* < H_t < H^*$ — изменилось, если же $H_t > H^*$ — тогда поведение аномально.*

Для того чтобы проверить адекватность предложенного критерия, для разных пользователей были проведены эксперименты. Для этого использовались реальные данные, описание которых приводилось в разделе 3. Сначала значение H_t вычислялось для сеансов, во время которых поведение пользователя было нормальным (характерным). (На рис. 3 им соответствуют номера сеансов от 1 до 24.) Как видно из рис. 3, значения этого параметра для разных сеансов отличаются незначительно и лежат в пределах (0; 0,15). Для моделирования аномальной работы пользователя осуществлялась подмена данных. Т.е. вектор $g(s_t)$ строился для сеанса другого пользователя и сравнивался с аналогичными векторами, построенными для предшествующих сеансов исходного пользователя. Полученное в этом случае значение параметра H_t резко увеличивалось (на рис. 3 до 0,32).

Для того чтобы смоделировать естественное изменение поведения пользователей, было сделано следующее: значения элементов вектора $g(s_t)$ случайным образом (с вероятностью 0,25) изменялись на противоположные (на рис. 3 им соответствуют сеансы с номерами 35-50). Т.е. моделировалась ситуация, когда пользователь переставал выполнять одни команды и начинал использовать другие. В этом случае

значение параметра H_t увеличивалось до 0,25 (сеанс с номером 35), а затем снова, как и в случае с нормальным поведением, выходило на обычный уровень.

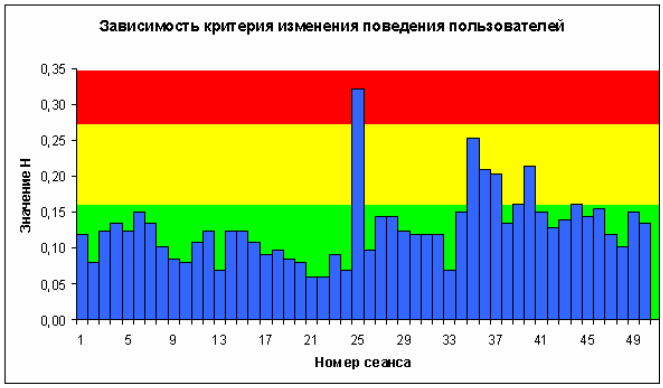


Рис. 3. Изменение значения H_t

Таким образом, результаты проведенных экспериментов показывают, что использование параметра H_t позволяет определить момент времени, когда поведение пользователя изменилось.

5. Моделирование данных о работе пользователя

В общем случае функционирование нейронной сети значительно зависит от качества обучающей выборки. Дело в том, что при небольшом размере обучающего множества нейронная сеть имеет тенденцию к жесткому запоминанию образов, что приводит к уменьшению ее способности к обобщению. Так, при построении интерактивной модели пользователя в нашем случае проблема с представительной выборкой данных не возникала, поскольку даже за непродолжительный период времени работы пользователя может быть собрано достаточное количество образов (представительных) для обучения нейронной сети. (Например, с учетом того, что пользователь в среднем вводит от 80 до 150 команд за сеанс, то за десять сеансов обучающая выборка может насчитывать до 1000 образов.)

В случае сеансовой модели размер обучающего множества напрямую определяется количеством сеансов, во время которых проводилось наблюдение за работой пользователя. Так, за три месяца таковых сеансов может достигать 100, что в нашем случае было недостаточно для качественного обучения нейронной сети и оптимизации ее архитектуры. Для решения этой проблемы можно использовать два подхода. Первый из них состоит в значительном увеличении отрезка времени, в рамках которого происходит наблюдение за поведением пользователя (скажем до 8-10 месяцев). Однако в этом случае велика вероятность того, что за этот промежуток времени оно изменится и, таким образом, обучающееся множество будет содержать противоречивые образы. Второй подход заключается в статистическом моделировании данных на основе имеющейся выборки. Он и будет использован в данной работе.

Поскольку в сеансовой модели для обучения нейронной сети используется информация о количестве вводимых команд за сеанс, номере компьютера, продолжительности и времени начала сеанса, для каждого пользователя проводилось моделирование именно этого набора данных. Для этого проверялось соответствие эмпирического распределения набору теоретических (нормальному, логарифмическому нормальному, равномерному и т.д.). В качестве критерия согласия использовался так называемый критерий χ^2 Пирсона [5, 6].

Критерий согласия χ^2. Пусть для каждого сеанса s_t имеется следующий набор данных о работе пользователя: $\{n_{s_t}, h_{s_t}, d_{s_t}, s_{s_t}\}_{t=1}^{T}$. Каждую из этих величин можно рассматривать как случайную, принимающую T определенных значений. К тому же, будем считать эти величины независимыми. Обозначим посредством X одну из этих случайных величин. Разобьем ее значения по k интервалам и представим в виде следующего статистического ряда:

l_i	$x_1; x_2$	$x_2; x_3$	…	$x_k; x_{k+1}$
p_i^*	p_1^*	p_2^*	…	p_k^*

где l_i — i-ый интервал значений, p_i^* — частота попадания в интервал l_i.

Требуется проверить, согласуются ли эти данные с гипотезой о том, что случайная величина *X* имеет закон распределения, заданный функцией распределения *F(x)* или плотностью *f(x)* (назовем его теоретическим). Зная теоретический закон распределения, можно найти теоретические вероятности попадания случайной величины в тот или иной интервал: $p_1, p_2, \ldots, p_k$.

Проверяя согласованность теоретического и статистического (эмпирического) распределений, возникает вопрос о том, каким же способом следует выбирать меру расхождений. В качестве такой меры в математической статистике обычно выбирают величину:

$$U = \sum_{i=1}^{k} n \frac{\left(p_i^* - p_i\right)^2}{p_i}. \qquad (10)$$

Так, Пирсон показал, что при достаточно больших значениях *n* закон распределения величины *U* практически не будет зависеть от функции распределения *F(x)* и числа *n*, а будет зависеть только от количества интервалов *k* и с увеличением *n* приближаться к распределению χ^2 с плотностью:

$$f(u) = \frac{1}{2^{\frac{r}{2}}\Gamma\left(\frac{r}{2}\right)} u^{\frac{r}{2}-1} \mathrm{e}^{-\frac{u}{2}} \quad (u > 0),$$

где $\Gamma(\alpha) = \int_0^\infty t^{\alpha-1}\mathrm{e}^{-t}dt$ — гамма-функция; *r* — количество степеней свободы. Поскольку распределение χ^2 зависит от параметра *r* (число степеней свободы), то для его вычисления используют следующее правило:

$$r = k\text{-}s,$$

где *s* — количество независимых условий, наложенных на вероятности p_i^*. К примерам таких условий можно отнести требование, что бы сумма всех частот p_i^* была равна единице, или требование равенства теоретического среднего и дисперсии статическому и т.д.

Для распределения χ^2 составлены специальные таблицы. Пользуясь ими, можно для каждого значения χ^2 и числа степеней свободы *r* найти вероятность *p* того, что величина, распределенная по закону χ^2, превзойдет это значение.

Таким образом, схема применения критерия χ^2 к оценке согласованности статистического и теоретического распределений сводится к следующему:

1. Определяется мера расхождения *U* в соответствии с формулой (10).
2. Определяется число степеней свободы *r* как разность между числом интервалов *k* и количеством наложенных связей *s*.
3. По полученным значениям *U* и *r* (с помощью специальных таблиц для χ^2) определяется вероятность *p*. Если эта вероятность весьма мала, гипотеза о соответствии статистического распределения теоретическому отбрасывается. Если же эта вероятность относительно велика, данную гипотезу можно признать не противоречащей опытным данным.

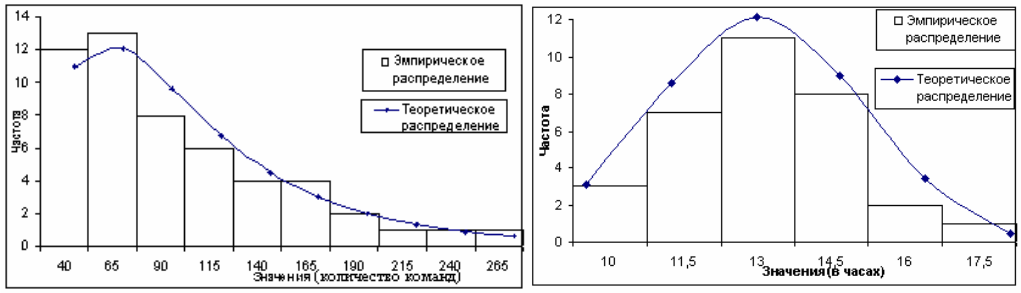
Рассмотрим теперь зависимости, которые были получены при моделировании.

Количество вводимых команд. Был проведен анализ различных распределений. Наилучшее значение критерия χ^2, равное 1,98, было получено для логарифмического нормального распределения. Поскольку в данном случае количество степеней свободы *r* равнялось 7 ($k = 10$, $s = 3$), это значение обеспечивало 97%-ое соответствие гипотезы реальным данным. На рис. 4,а приведено эмпирическое и построенное теоретическое распределение для этого параметра.

Номер компьютера. При анализе значений этого параметра необходимо учитывать следующее: пользователь, как правило, имеет свое основное место работы за компьютером и очень редко работает за остальными. Поэтому всегда существует значение, вероятность которого наибольшая и составляет 0,8…0,95. События же, связанные с работой пользователя за другими рабочими станциями, можно считать равновероятными.

Продолжительность сеанса. Анализ значений этого параметра показывает, что они распределены равновероятно (рис. 6). Значение критерия χ^2, равное 2,19, (при значениях параметров $r = 7$, $k = 10$, $s = 3$) обеспечивает 94%-ое соответствие гипотезы реальным данным.

Время начала сеанса. Наилучшее значение критерия χ^2, равное 0,87, для этого параметра было получено для нормального распределения. При $r = 3$, $k = 6$, $s = 3$ это значение обеспечивало 85%-ое соответствие гипотезы реальным данным. На рис. 4,б приведено эмпирическое и построенное теоретическое распределение.



(а) (б)

Рис. 4. Эмпирическое и теоретическое распределения для количества вводимых команд (а) и времени начала сеанса (б)

На основе полученных теоретических зависимостей была разработана программа, моделирующая работу пользователя за сеанс. Сформированные с ее помощью данные позволили оптимизировать архитектуру нейронной сети и улучшить качество ее функционирования.

6. Заключение

В данной работе проводилось математическое моделирование поведения пользователей компьютерных систем. Для этого использовалась комплексная модель, предложенная в работе [3]. Изучалась динамика работы пользователя во время сеанса. Поскольку определение оптимальной архитектуры нейронной сети является нетривиальной задачей, в интерактивной модели важно знать, сколько команд следует использовать при обучении для прогноза следующей. На основе построенных автокорреляционных кривых было выявлено, что для этого следует учитывать до восьми команд.

Также было проведено статистическое моделирование данных, используемых для обучения нейронной сети в сеансовой модели. Эмпирические распределения были аппроксимированы теоретическими. На их основе были сгенерированы необходимый набор данных, что дало возможность оптимизировать архитектуру нейронной сети и улучшить ее функционирование.

Литература

1. Куссуль Н., Соколов А. Адаптивное обнаружение аномалий в поведении пользователей компьютерных систем с помощью марковских цепей переменного порядка. Часть 2. Методы обнаружения аномалий и результаты экспериментов // Проблемы управления и информатики. — 2003. — № 4. — С. 83–88.
2. Manavoglu E., Pavlov D., Lee Giles C. Probabilistic User Behavior Models // Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003). — Melbourne, Florida (USA). — 2003. — P. 203–210.
3. Скакун С.В., Куссуль Н.Н. Нейросетевая модель пользователей компьютерных систем // Кибернетика и вычислительная техника. — 2004. — Выпуск 143. — С. 55–68.
4. Haykin S. Neural Networks: a comprehensive foundation. — Upper Saddle River, New Jersey: Prentice Hall, 1999. — 842 p.
5. Ширяев А.Н. Вероятность. — М.: Наука, 1980. — 575 с.
6. Вентцель Е.С. Теория вероятностей: Учебник для студ. вузов. — М.: Издательский центр «Академия», 2003. — 576 с.

Информация об авторах

Куссуль Н.Н. – зав. отделом, д.т.н., Институт космических исследований НАНУ-НКАУ; г. Киев, пр. Глушкова 40, тел. 8(044)266-25-53; e-mail: inform@ikd.kiev.ua

Скакун С.В. – м.н.с., Институт космических исследований НАНУ-НКАУ; г. Киев, пр. Глушкова 40, тел. 8(044)266-25-53; e-mail: inform@ikd.kiev.ua

JAMMING CANCELLATION BASED ON A STABLE LSP SOLUTION

Elena Revunova, Dmitri Rachkovskij

***Abstract**: Two jamming cancellation algorithms are developed based on a stable solution of least squares problem (LSP) provided by regularization. They are based on filtered singular value decomposition (SVD) and modifications of the Greville formula. Both algorithms allow an efficient hardware implementation. Testing results on artificial data modelling difficult real-world situations are also provided*

***Keywords**: jamming cancellation, approximation, least squares problem, stable solution, recurrent solution, neural networks, incremental training, filtered SVD, Greville formula*

Introduction

Jamming cancellation problem appears in many application areas such as radio communication, navigation, radar, etc. [Shirman, 1998], [Ma, 1997]. Though a number of approaches to its solution were proposed [McWhirter, 1989], [Ma et al., 1997], no universal solution exists for all kinds of jamming and types of antenna systems, stimulating further active research to advance existing methods, algorithms and implementations.

Consider an antenna system with a single primary channel and n auxiliary channels. Signal in each channel is, generally, a mixture of three components: a valid signal, jamming, and channel's inherent noise. The problem consists in maximal jamming cancellation at the output while maximally preserving valid signal.

Within the framework of weighting approach [Shirman, 1998], the output is obtained by subtraction of the weighted sum of signals provided by the auxiliary channels from the primary channel signal. The possibility of determining a weight vector $\boldsymbol{w}^*$ that minimizes noise at the output while preserving the valid signal to a maximum degree is, in general case, provided by the following. The same jamming components are present both in primary and auxiliary channels, however, with different mixing factors. Valid signal has small duration and amplitude and is almost absent in auxiliary channels. Characteristics of jamming, channel's inherent noise, and their mixing parameters are stable within the sliding "working window".

These considerations allow us to formulate the problem of obtaining the proper $\boldsymbol{w}^*$ as a linear approximation of a real-valued function $y=f(x)$:

$$F(x) = w_1 h_1(x) + w_2 h_2(x) + \ldots + w_n h_n(x) = \sum_{i=1,\,n} w_i\, h_i\,(x), \qquad (1)$$

where $h_1(x),\ldots,h_n(x)$ is a system of real-valued basis functions; $w_1,\ldots,w_n$ are the real-valued weighting parameters, $F(x)$ is a function approximating f(x).

In our case, $h_1(x),\ldots,h_n(x)$ are signals provided by the n auxiliary channels. Information about $y=f(x)$ at the output of the primary channel is given at discrete set of (time) points $k=1,\ldots,m$ (m is the width of the working window) by the set of pairs $(\boldsymbol{h}^k, y^k)$. It is necessary to find $\boldsymbol{w}^*$ approximating $f(x)$ by $F(x)$ using linear least squares solution:

$$\boldsymbol{w}^* = \mathrm{argmin}_{\,w}\, ||\boldsymbol{H}\,\boldsymbol{w} - \boldsymbol{y}||_2, \qquad (2)$$

where $\boldsymbol{H}$ is the so-called $m \times n$ "design matrix" containing the values provided by the n auxiliary channels for all $k=1,\ldots,$m; and $\boldsymbol{y} = [y_1, \ldots, y_m]^T$ is the vector of corresponding y values provided by the primary channel.

After estimating $\boldsymbol{w}^*$, the algorithm's output $\boldsymbol{s}$ is the residual discrepancy:

$$\boldsymbol{s} = \boldsymbol{H}\,\boldsymbol{w}^* - \boldsymbol{y}. \qquad (3)$$

Such a system may be represented as a linear neural network with a single layer of modifiable connections, $n+1$ input and single output linear neurons connected by a weight vector $\boldsymbol{w}$ (Fig. 1). In the case of successful training $\boldsymbol{w}^*$ provides an increased signal-jamming ratio at the output s compared to the input of the primary channel y, at least, for the training set $\boldsymbol{H}$.

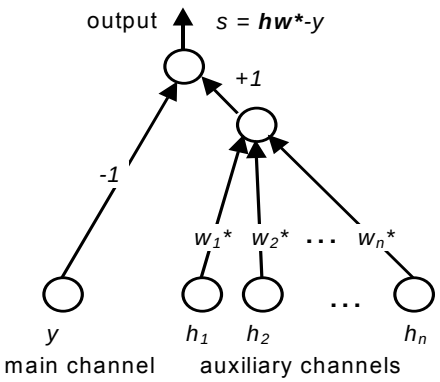


Fig 1. A neural network representation of a jamming canceller

A peculiarity of jamming cancellation problem in such a formulation consists in contamination of both $\boldsymbol{y}$ and $\boldsymbol{H}$ by the inherent noise of channels. Existing algorithms for jamming cancellation in the framework of weighting processing (2)-(3) [Shirman, 1998] do not take into account inherent noise contamination of $\boldsymbol{y}$ and $\boldsymbol{H}$. This results in instability of $\boldsymbol{w}$ estimation, leading, in turn, to a deterioration of cancellation characteristics, and often even to amplification of noise instead of its suppression. Therefore, methods for obtaining $\boldsymbol{w}^*$ should be stable to inherent noise contamination of $\boldsymbol{y}$ and $\boldsymbol{H}$. Other necessary requirements are real-time operation and simplicity of hardware implementation.

Least Squares Solution and Regularization

Generally, the solution of the least squares problem (LSP) (2) is given by

$$\boldsymbol{w}^* = \boldsymbol{H}^+ \boldsymbol{y}; \tag{4}$$

where $\boldsymbol{H}^+$ is pseudo-inverse matrix. If $\boldsymbol{H}$ is non-perturbed (noise is absent), then:

$$\text{for } m = n,\ \text{rank}(\boldsymbol{H}) = n = m \Rightarrow \det \boldsymbol{H} \neq 0.\ \boldsymbol{H}^+ = \boldsymbol{H}^{-1}; \tag{5}$$

$$\text{for } m > n,\ \text{rank}(\boldsymbol{H}) = n, \Rightarrow \det \boldsymbol{H} \neq 0:\ \boldsymbol{H}^+ = (\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T; \tag{6}$$

$$\text{for } m = n,\ m > n,\ \text{rank}(H) < n \Rightarrow \det H = 0,\ \boldsymbol{H}^+ = \lim_{v \to 0} (\boldsymbol{H}^T\boldsymbol{H} + v^2\boldsymbol{I})^{-1}\boldsymbol{H}^T. \tag{7}$$

$\boldsymbol{H}^+$ for (7) can be obtained numerically using SVD [Demmel, 1997] or the Greville formula [Greville, 1960].

The case when $\boldsymbol{y}$ and elements of matrix $\boldsymbol{H}$ are known precisely is very rare in practice. Let us consider a case that is more typical for jamming cancellation, i.e. when $\boldsymbol{y}$ and $\boldsymbol{H}$ are measured approximately: $\boldsymbol{y} = \boldsymbol{y} + \boldsymbol{\varsigma}$; $\boldsymbol{H} = \boldsymbol{H} + \boldsymbol{\Xi}$; where $\boldsymbol{\varsigma}$ is noise vector, $\boldsymbol{\Xi}$ is noise matrix. In such a case, solutions (5)-(7) may be unstable, i.e. small changes of $\boldsymbol{y}$ and $\boldsymbol{H}$ cause large changes of $\boldsymbol{w}^*$ resulting in instable operation of application systems based on (5)-(7).

To obtain a stable LSP solution, it is fruitful to use approaches for solution of "discrete ill-posed problems" [Hansen, 1998], [Jacobsen et al., 2003], [Reginska, 2002], [Wu, 2003], [Kilmer, 2003]. Such a class of LSPs is characterized by $\boldsymbol{H}$ with singular values gradually decaying to zero and large ratio between the largest and the smallest nonzero singular values. This corresponds to approximately known and near rank-deficient $\boldsymbol{H}$.

Reducing of an ill-posed LSP to a well-posed one by introduction of the appropriate constraints to the LSP formulation is known as regularization [Hansen, 1998]. Let us consider a problem known as standard form of the Tikhonov regularization [Tikhonov, 1977]:

$$\text{argmin}_w \{||\boldsymbol{y} - \boldsymbol{H}\boldsymbol{w}||_2 + \lambda\, ||\boldsymbol{w}||_2\}. \tag{8}$$

Its solution $\boldsymbol{w}_\lambda$ may be obtained in terms of SVD of $\boldsymbol{H}$ [Hansen, 1998]:

$$\boldsymbol{w}_\lambda = \sum_{i=1,\,n} f_i\ \boldsymbol{u}_i^T\boldsymbol{y}/\sigma_i\ \boldsymbol{v}_i; \tag{9}$$

$$f_i = \sigma^2_i/(\sigma^2_i + \lambda^2), \tag{10}$$

where σ_i are singular values, $\boldsymbol{u}_1 \ldots \boldsymbol{u}_n$, $\boldsymbol{v}_1 \ldots \boldsymbol{v}_n$ are left and right singular vectors of $\boldsymbol{H}$, f_i are filter factors.

Note that solution of (8) using truncated SVD method [Demmel, 1997] is a special case of (9), (10) with $f_i \in \{0,1\}$.

Algorithmic Implementation of Solutions of Discrete ill-posed LSPs

Requirements of an efficient hardware implementation of jamming cancellation pose severe restrictions on the allowable spectrum of methods and algorithms. In particular, methods of $\boldsymbol{w}^*$ estimation are required to allow for parallelization or recursion. Taking this into account, let us consider some algorithmic implementations of (8).

SVD-based solution. The implementation of SVD as a systolic architecture with paralleled calculations is considered in [Brent, 1985]. We have developed a systolic architecture that uses effective calculation of $\boldsymbol{u}_i$, σ_i, and $\boldsymbol{v}_i$ for obtaining the regularized solution $\boldsymbol{w}_\lambda^*$ (9)-(10). The architecture implements two regularization techniques: truncated and filtered SVD [Hansen, 1998].

Advantages of SVD-based solution are accuracy and parallelism. Drawbacks are connected with the hardware expenses for calculation of trigonometric functions for diagonalization of sub-matrices and implementation of systolic architecture itself.

Solution based on the Greville formula and its modifications. Let us consider another stable method for $\boldsymbol{w}^*$ estimation based on the Greville formula, which can be readily implemented in hardware because of its recursive character. A recursive procedure for the LSP solution [Plackett, 1950] for a full-rank $\boldsymbol{H}$ is as follows:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \boldsymbol{b}_{k+1}(y_{k+1} - \boldsymbol{h}^T_{k+1}\boldsymbol{w}_k); \quad k = 0, 1, \ldots, \tag{11}$$

$$\boldsymbol{b}_{k+1} = \boldsymbol{P}_k\boldsymbol{h}_{k+1}/(1 + \boldsymbol{h}^T_{k+1}\boldsymbol{P}_k\boldsymbol{h}_{k+1}); \tag{12}$$

$$\boldsymbol{P}_{k+1} = (\boldsymbol{H}^T_{k+1}\boldsymbol{H}_{k+1})^{-1} = (\boldsymbol{I} - \boldsymbol{b}_{k+1}\boldsymbol{h}^T_{k+1})\boldsymbol{P}_k; \tag{13}$$

where $\boldsymbol{h}_k$ is the kth row (sample) of $\boldsymbol{H}$; $\boldsymbol{P}_0 = 0$; $\boldsymbol{w}_0 = 0$. Note that this provides an iterative version of training algorithm for a neural network interpretation of Fig. 1.

The Greville formula [Greville, 1960] allows $\boldsymbol{b}_{k+1}$ calculation for (11) without $(\boldsymbol{H}^T_{k+1}\boldsymbol{H}_{k+1})^{-1}$ calculation, thus overcoming the problem of rank-deficiency of $\boldsymbol{H}$:

$$\boldsymbol{b}_{k+1} = (\boldsymbol{I} - \boldsymbol{H}^+_k\boldsymbol{H}_k)\boldsymbol{h}_{k+1} / \boldsymbol{h}^T_{k+1}(\boldsymbol{I} - \boldsymbol{H}^+_k\boldsymbol{H}_k)\boldsymbol{h}_{k+1}; \ \text{if} \ \boldsymbol{h}^T_{k+1}(\boldsymbol{I} - \boldsymbol{H}^+_k\boldsymbol{H}_k)\boldsymbol{h}_{k+1} \neq 0; \tag{14}$$

$$\boldsymbol{b}_{k+1} = \boldsymbol{H}^+_k(\boldsymbol{H}^+_k)^T\boldsymbol{h}_{k+1} / (1 + \boldsymbol{h}^T_{k+1}\boldsymbol{H}^+_k(\boldsymbol{H}^+_k)^T\boldsymbol{h}_{k+1}); \ \text{if} \ \boldsymbol{h}^T_{k+1}(\boldsymbol{I} - \boldsymbol{H}^+_k\boldsymbol{H}_k)\boldsymbol{h}_{k+1} = 0; \tag{15}$$

$$\boldsymbol{H}^+_{k+1} = (\boldsymbol{H}^+_k - (\boldsymbol{b}_{k+1}\boldsymbol{h}^T_{k+1}\boldsymbol{H}^+_k \mid \boldsymbol{b}_{k+1})). \tag{16}$$

$\boldsymbol{w}^*$ obtained by (11)-(13) using (14)-(16) is equivalent to $\boldsymbol{w}^*$ obtained by (7) for precisely specified $\boldsymbol{H}$. Presence of $\boldsymbol{H}^+_k$ and $\boldsymbol{H}_k$ in (14)-(16) makes recursion more resource- and computation-expensive than (12)-(13). As a new sample arrives, it is necessary to calculate $\boldsymbol{H}^+_k\boldsymbol{H}_k$ or $\boldsymbol{H}^+_k(\boldsymbol{H}^+_k)^T$ that requires calculation of $\boldsymbol{H}^+_k$ and storage of $\boldsymbol{H}^+_k$ and $\boldsymbol{H}_k$. These drawbacks are overcome by an improvement of the Greville formula proposed recently in [Zhou, 2002].

For $\boldsymbol{h}^T_{k+1}\boldsymbol{Q}_k = 0$ calculations of $\boldsymbol{b}_{k+1}$ and $\boldsymbol{P}_{k+1}$ are made by (12)-(13). If $\boldsymbol{h}^T_{k+1}\boldsymbol{Q}_k \neq 0$

$$\boldsymbol{b}_{k+1} = \boldsymbol{Q}_k\boldsymbol{h}_{k+1}/(\boldsymbol{h}^T_{k+1}\boldsymbol{Q}_k\boldsymbol{h}_{k+1}); \tag{17}$$

$$\boldsymbol{P}_{k+1} = (\boldsymbol{I} - \boldsymbol{b}_{k+1}\boldsymbol{h}^T_{k+1})\boldsymbol{P}_k(\boldsymbol{I} - \boldsymbol{b}_{k+1}\boldsymbol{h}^T_{k+1})^T + \boldsymbol{b}_{k+1}\boldsymbol{b}^T_{k+1}; \tag{18}$$

$$\boldsymbol{Q}_{k+1} = (\boldsymbol{I} - \boldsymbol{b}_{k+1}\boldsymbol{h}^T_{k+1})\boldsymbol{Q}_k. \tag{19}$$

Here $\boldsymbol{P}_k = \boldsymbol{H}^+_k(\boldsymbol{H}^+_k)^T$ is Hermitian $n \times n$ matrix; $\boldsymbol{P}_0 = 0$, $\boldsymbol{Q}_k = \boldsymbol{I} - \boldsymbol{H}^+_k\boldsymbol{H}_k$; $\boldsymbol{Q}_0 = \boldsymbol{I}$.

We further modified the Greville formula so that $\boldsymbol{w}_{k+1}$ is equivalent to the regularized solution $\boldsymbol{w}_\lambda^*$ (9). This is achieved by comparison of vector norm $\boldsymbol{h}^T_{k+1}\boldsymbol{Q}_k$ not with 0, but with some threshold value 0_{eff} calculated from noise matrix $\boldsymbol{\Xi}$. We name such an algorithm "pseudo-regularized modification of the Greville formula" (PRMGF).

The algorithm (11)-(13), (17)-(19) calculates $\boldsymbol{w}^*$ using all previous samples. However, for a non-stationary case it is necessary to process only a part of the incoming samples inside a sliding working window. Full recalculation of $\boldsymbol{H}^+_{k+1}$ for estimation of $\boldsymbol{w}_{k+1}$ as each new sample arrives can be avoided by using inverse recurrent representation of [Kirichenko, 1997]. For the purpose of removing the row $\boldsymbol{h}^T_1$ from $\boldsymbol{H}_{k+1}$, $\boldsymbol{H}^+_k$ is represented through $\boldsymbol{H}^+_{k+1} = (\boldsymbol{b}_1 | \boldsymbol{B}_{k+1})$ as follows.

For a linear independent row:

$$(\boldsymbol{H} + \boldsymbol{h}\boldsymbol{e}^T)^+ = \boldsymbol{H}^+ - \boldsymbol{H}^+\boldsymbol{h}\boldsymbol{h}^T\boldsymbol{Q}/(\boldsymbol{h}^T\boldsymbol{Q}\boldsymbol{h}) - \boldsymbol{Q}\boldsymbol{e}\boldsymbol{e}^T\boldsymbol{H}^+/\boldsymbol{e}^T\boldsymbol{Q}\boldsymbol{e} + \boldsymbol{Q}\,\boldsymbol{e}\boldsymbol{h}^T\boldsymbol{Q}(\boldsymbol{H}^T)\,(1 + \boldsymbol{e}^T\boldsymbol{H}^+\boldsymbol{h})/\,\boldsymbol{h}^T\boldsymbol{Q}(\boldsymbol{H}^T)\boldsymbol{h}\,\boldsymbol{e}^T\boldsymbol{Q}\boldsymbol{e}; \tag{20}$$

and for a linear dependent row:

$$(\boldsymbol{H} + \boldsymbol{h}\boldsymbol{e}^T)^+ = (\boldsymbol{I} - \boldsymbol{z}\boldsymbol{z}^T / \|\boldsymbol{z}\|^2)\, \boldsymbol{H}^+;\ \boldsymbol{z} = \boldsymbol{H}^+\boldsymbol{h} - \boldsymbol{e} / \|\boldsymbol{e}\|^2. \quad (21)$$

Thus, we propose to use PRMGF with a sliding window for the case, when it is required to obtain $\boldsymbol{w}^*$ not for the whole training set, but for its subset of a fixed size. For initial $k < K$ samples $\boldsymbol{h}_k$ (where K is the size of working window) $\boldsymbol{w}^*_k$ is recursively calculated by PRMGF. For $k > K$, (20)-(21) are used for updating $\boldsymbol{H}^+$ by removing the sample that has left a working window, and the incoming sample s is taken into account using PRMGF as earlier.

Advantages of PRMGF with a sliding window include:

- natural embedding into recursive algorithm for $\boldsymbol{w}^*$;
- increase of calculation speed due to using $\boldsymbol{h}^T_{k+1}$ instead of $\boldsymbol{H}_k$, also resulting in reduction of required memory;
- additional memory reduction since $\boldsymbol{P}_k$, $\boldsymbol{K}_k$ and $\boldsymbol{Q}_k$ have fixed $n \times n$ dimension for any k;
- further increase of calculation speed when sliding window is used due to the Greville formula inversion;
- considerably smaller hardware expenses in comparison with SVD;
- $\boldsymbol{w}^*$ close to the Tikhonov regularized solution for noisy, near rank-deficient $\boldsymbol{H}$ (at least, for small matrices);
- natural interpretation as an incrementally trained neural network.

Example of Modelling a Jamming Cancellation System

Let's compare the following jamming cancellation algorithms: ordinary LS-based (6); non-truncated SVD-based [Demmel, 1997]; truncated SVD-based (9) with $f_i = \{0, 1\}$; PRMGF-based (section 3.2). We use near rank-deficient $\boldsymbol{H}$, which is critical for traditional jamming cancellation algorithms – e.g., for ordinary LS-based ones.

Testing scheme and cancellation quality characteristics. In a real situation, all antenna channels receive jamming signals weighted by the gain factor that is determined by the antenna directivity diagram in the direction of particular jamming. We simulated signals in antenna channels as follows:

$$\boldsymbol{X} = \boldsymbol{S}\,\boldsymbol{M} + \boldsymbol{\Xi}; \quad (22)$$

where $\boldsymbol{X}$ is $L \times (n+1)$ matrix of signals in antenna channels ($\boldsymbol{H}$ is sub-matrix of $\boldsymbol{X}$); L is the number of samples; n is the number of auxiliary channels; $\boldsymbol{S}$ is jamming signals' matrix; $\boldsymbol{\Xi}$ is channel inherent noise matrix; $\boldsymbol{M}$ is mixing matrix.

Jamming signals and channels' inherent noise are modeled by normalized centered random variables with normal and uniform distribution correspondingly. $\boldsymbol{M}$ is constructed manually, values of its elements are about units, rank deficiency was achieved by entering identical or linearly dependent rows. For ideal channels without inherent noise, rank deficiency of $\boldsymbol{M}$ gives rise to strict rank deficiency of $\boldsymbol{H}$. Inherent noise results in near rank-deficient $\boldsymbol{H}$. Tests were carried out for n=8 auxiliary channels.

The main characteristics of jamming cancellers are: jamming cancellation ratio (K^c) and jamming cancellation ratio vs inherent noise level in auxiliary channels K^{naux}: $K^c = f(K^{naux})$ [Bondarenko, 1987] at fixed inherent noise at the primary channel K^{n0}.

$$K^c = P^{in}/P^{out}, \quad (23)$$

where P^{in} and P^{out} is power of jamming in the primary channel and in the output of jamming canceller, respectively. In all tests, the valid signal with amplitude not exceeding amplitude of primary channel jamming was present at the input for 5 nearby samples. L=1000; m=16; K^{n0} = {0.1, 0.2, 0.3}, $K^{n0} >> K^{naux}$ to complicate the algorithm's operation.

Testing results. A family of jamming cancellation characteristics $K^c = f(K^{naux})$ for rank-deficient $\boldsymbol{M}$ and near rank-deficient $\boldsymbol{H}$ is shown in Fig.2. K^{naux} varied from $1.6\ 10^{-9}$ up to $6.4\ 10^{-6}$. K^c for the ordinary LS did not exceed 1 at $K^{naux} < 2.5\ 10^{-8}$. For truncated SVD and PRMGF $K^c \approx 10$ ($K^{n0} = 0.1$) are nearly constant over the whole range of K^{naux} and close to each other. Note that for a full-rank matrix $\boldsymbol{H}$, K^n for all algorithms was approximately the same and large in the considered range of K^{naux}.

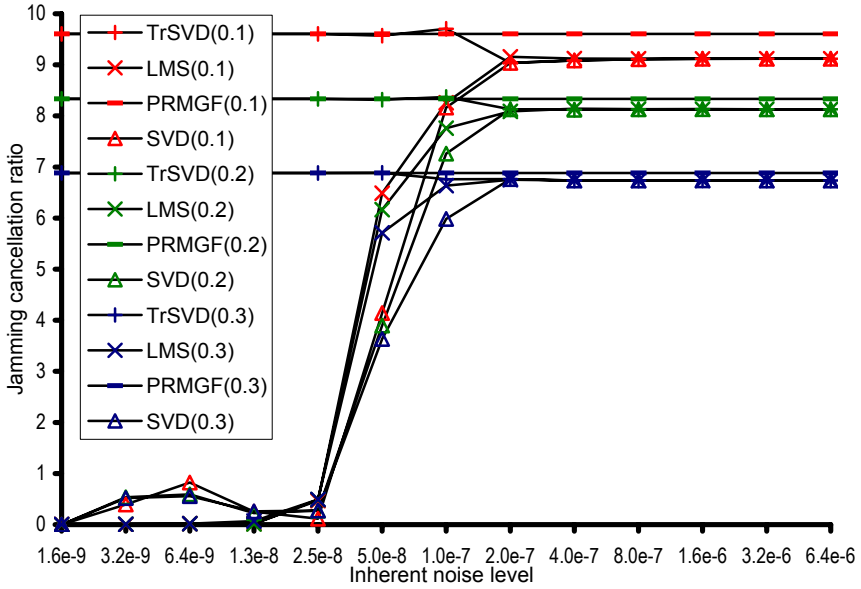


Fig. 2. Kc= f(Knaux) for near rank-deficient H

It may seem from the analysis of the shown results that one may use the ordinary LS algorithm at increased level of K^{naux}. However, roll-of of cancellation characteristic is also observed when jamming intensity in auxiliary channels is much more than the inherent noise level. To show that, let us consider $K^c(P^{in})$ for near rank-deficient $\boldsymbol{H}$ (Fig.3). Jamming power P^{in} changed from $9 \cdot 10^3$ to $1.2 \cdot 10^7$ by step $2 \cdot 10^2$, $K^{naux} = 0.1$. For $P^{in} > 10^4$, K^c for ordinary LS and non-truncated SVD decreases. For truncated SVD and PRMGF, $K^c \approx 9$ ($K^{n0} = 0.1$) are constant and close to each other. In this case, we cannot artificially increase inherent noise level because it will completely mask the valid signal.

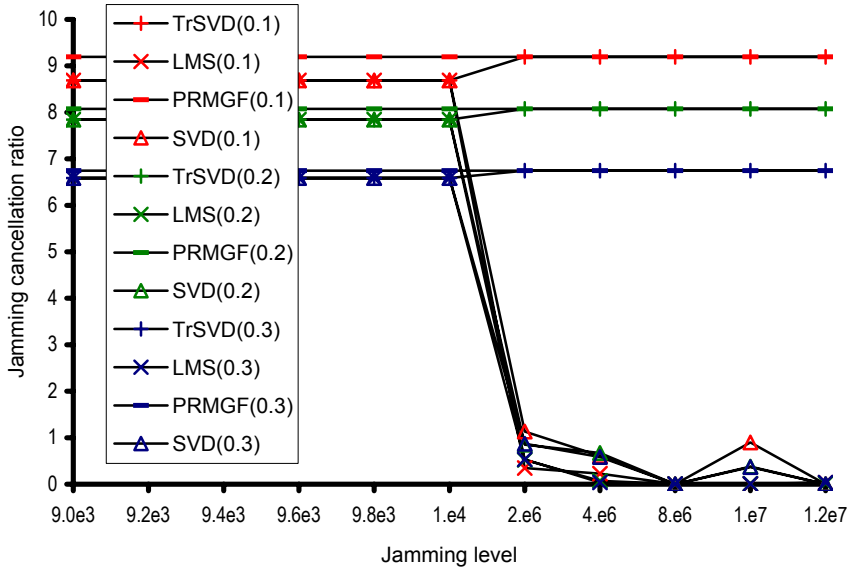


Fig. 3. Kc(Pin) for near rank-deficient H

Conclusions

In the framework of this work, two new jamming cancellation algorithms have been developed based on the so-called weighting approach. Special requirements to the problem have resulted in its classification as a discrete ill-posed problem. That has allowed us to apply an arsenal of the regularization-based methods for its stable solution - estimation of weight vector $\boldsymbol{w}^*$.

The standard form of Tikhonov regularization based on SVD has been transformed to efficient hardware systolic architecture. Besides, pseudo-regularized modification of the Greville formula allowed us to get weight vector estimations very close to estimations for a truncated SVD based regularization - at least for ***H*** of about tens of columns. Testing on near rank-deficient ***H*** has shown that distinctions in ***w**** obtained by both algorithms are of the order 10^{-5}. A combined processing technique based on a regularized modification of the Greville formula and inverse recurrent representation of Kirichenko permits a more efficient processing of data for a sliding working window.

Testing on artificial data that model real-world jamming cancellation problem has shown an efficient cancellation for near rank-deficient ***H***. For the developed PRMGF-based algorithm the jamming cancellation ratio is near constant and considerably higher than 1 in the whole range of variation of auxiliary channels' inherent noise and jamming amplitude. On the contrary, for the non-regularized LS method the ratio roll-offs to less than 1, meaning jamming amplification.

A straightforward neural network interpretation of such a system is provided. The developed algorithms and computer architectures for their implementation can be applied to solution of other discrete ill-posed LS problems and systems of linear algebraic equations.

Bibliography

[Bondarenko, 1987] B.F. Bondarenko, Bases of radar systems construction, Kiev, 1987 (in Russian).

[Brent, 1983] R.P. Brent, H. T. Kung and F. T. Luk, Some linear-time algorithms for systolic arrays, in Information Processing 83, 865–876, North-Holland, Amsterdam, 1983.

[Brent, 1985] R.P. Brent and F. T. Luk, The solution of singular-value and symmetric eigenvalue problems on multiprocessor arrays, SIAM J. Scientific and Statistical Computing 6, 69–84, 1985.

[Demmel, 1997] J.W. Demmel, Applied Numerical Linear Algebra. SIAM, Philadelphia, 1997.

[Greville, 1960] T. N. E. Greville, Some applications of the pseudoinverse of a matrix, SIAM Rev. 2, pp. 15-22, 1960.

[Hansen, 1998] P.C. Hansen, Rank-deficient and discrete ill-posed problems. Numerical Aspects of Linear Inversion, SIAM, Philadelphia, 1998.

[Jacobsen et al., 2003] M. Jacobsen, P.C. Hansen and M.A. Saunders, Subspace preconditioned LSQR for discrete ill-posed problems, BIT Numerical Mathematics 43: 975–989, 2003.

[Kilmer, 2003] M. Kilmer, Numerical methods for ill-posed problems, Tufts University, Medford, MAPIMS Workshop, 2003.

[Kirichenko, 1997] N.F. Kirichenko, Analytical representation of pseudo-inverse matrix perturbation, Cybernetics and system analysis, 2, pp. 98-107, 1997 (in Russian).

[Ma et al., 1997] J. Ma, E.F. Deprettere and K.K. Parhi, Pipelined cordic based QRD-RLS adaptive filtering using matrix look ahead. In Proc. IEEE Workshop on Signal Processing Systems, pp.131-140 Lecester, UK, November, 1997.

[Ma, 1997] J. Ma, Pipelined generalized sidelobe canceller, Technique Report, Circuits and Systems, Delft University of Technology, 1997.

[McWhirter, 1989] J.G. McWhirter and T.J. Shepherd, Systolic array processor for MVDR beamforming, In IEEE Proceedings vol.136, pp. 75-80, April 1989.

[Plackett, 1950] R. L. Plackett, Some theorems in least squares, Biometrika, 37, pp. 149-157, 1950.

[Reginska, 2002] T. Reginska, Regularization of discrete ill-posed problems, IM PAN Preprints, 2002, http://www.impan.gov.pl/Preprints.

[Shirman, 1998] J.D. Shirman, Radio-electronic systems: bases of construction and the theory, Moscow, 1998 (in Russian).

[Tikhonov, 1977] A.N. Tikhonov, V.Y. Arsenin, Solution of ill-posed problems. V.H. Winston, Washington, DC, 1977.

[Wu, 2003] L. Wu, A parameter choice method for Tikhonov regularization, Electronic Transactions on Numerical Analysis, Volume 16, pp. 107-128, 2003.

[Zhou, 2002] J. Zhou, Y. Zhu, X. R. Li, Z. You, Variants of the Greville formula with applications to exact recursive least squares. SIAM J. Matrix Anal. Appl. Vol.24, No.1, pp.150-164, 2002.

Authors' Information

Elena G. Revunova, Dmitri A. Rachkovskij - International Research and Training Center of Information Technologies and Systems, Pr. Acad. Glushkova 40, Kiev 03680, Ukraine. email: helab@i.com.ua, dar@infrm.kiev.ua

GRAPH REPRESENTATION OF MODULAR NEURAL NETWORKS

Michael Kussul, Alla Galinskaya

***Abstract**: Modular neural networks are widely used for applied tasks solving due to its flexibility and big potential abilities. As a result, development of modelling aids for modular neural networks become very important. Networks that contain cycles are of particular interest. However, for the networks with cycles there is necessity to have tools for formal analysis, which allow defining sequence of run of modules in the networks. We propose representation of modular neural networks with help of directed graphs. This representation is intended for general analysis of modular architectures and, first of all, for analysis with automatic systems. On the basis of proposed representation we give definitions of cycles, build its classification and examine properties of cycles in modular neural networks.*

***Keywords**: neural networks, modular neural networks, graph of neural network, cycle.*

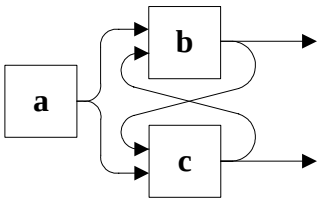
Introduction

Modular approach is actively developed in the field of application of artificial neural networks during last decade [1]. Modular networks, in particular, allow dividing a complex task into simple subtasks that are solved by individual modules and construct whole task decision as a combination of solved subtasks. Because of growing interest to modular architectures, development of modelling aids for modular neural networks becomes very important. By amodular network, we imply arbitrary set of algorithms for data processing, including artificial neural networks that combined for some task solving.

We should solve several problems both general and particular nature for modular neural networks application. There are questions of choosing the architecture of modular network for concrete task solving among particular problems. General problems are methods of combination of heterogeneous algorithms to one modular network, run sequence determination and methods of training of modular network as a whole.

The most of combination problems are solved in software neural computer MNN CAD [2]. Training of modular networks is a separate task, which solving is often architecture-dependent. Different approaches of modular networks training are given in articles [3] and [4].

Modular network can contain both algorithms that process data in parallel and modules that process data sequentially. Also, modular architectures can contain cycles. By the run sequence of modular network, we'll imply a sequence, in which modules process input data. Run sequence is obvious when there are no cycles in modular network. However, if we use cycles and run sequence is not defined by outer conditions it is possible that some contradictions occur. An example of such contradiction is a trigger scheme of modules' combination shown at the picture 1. The result of data processing will differ depending on what algorithm b or c was applied first or they work in parallel.



Pic. 1. Trigger scheme of modules' combination

Determination and resolution of contradictions in modular architectures is important task of modular networks' application. This task is of particular significance for development of interactive aids for modular network modelling such as MNN CAD. CAD system should automatically determine and interdict creation of contradiction architectures or demand contradiction resolution from user.

The main goal of this work is an attempt to propose model of modular neural networks, which allow creating consistent architectures and automatic determination of run sequence of modules.

We propose representation of modular networks with help of directed graphs. On the basis of proposed representation we give definitions of cycles, build its classification and examine properties of cycles in modular neural networks.

Digraph Representation of Modular Neural Networks

Modular network can be described by digraph $G(V,E)$, where vertices $v \in V$ correspond to network modules and edges $e \in E$ are connections between modules. We'll consider that digraph edge $e(v_1,v_2)$ correspond to all connections between modules v_1 and v_2. That is if module v has n inputs, some k of which connected to the outputs of module v_1 and remaining $n-k$ to the outputs of module v_2, then v will have two incoming edges $e_1(v_1,v)$ and $e_2(v_2,v)$. Similar to the definition in graph theory, we'll denote a walk as an alternating sequence of vertices and edges, with each edge being incident to the vertices immediately preceding and succeeding it in the sequence. A path (elementary path) is a walk with no repeated vertices. Walk and path from vertex v_1 to vertex v_2 we'll denote $w(v_1,v_2)$.

Let's introduce two special "virtual" modules representing inputs and outputs of whole modular network. All inputs of modular net we'll consider as input module and outputs – as an output module. Input and output modules are virtual in the sense that they do not realize data processing algorithms.

By the analogy with graph theory, we'll consider vertex I, which corresponds to virtual input module of network, to be **input** or source of graph G. Input of graph is determined by the condition $s_{in}(I)=0$, where $s_{in}(I)$ - the number of incoming edges.

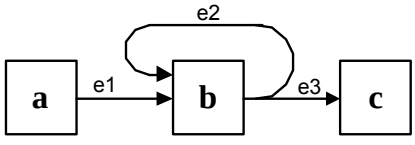
Output or sink of graph we'll denote vertex O that corresponds to the virtual output module. The next assertion is true for output: $s_{out}(O)=0$, where $s_{out}(O)$ - the number of outgoing edges. It should be noted that the number of the module outputs (in fact outputs of network) can be arbitrary.

In order that digraph G can be regarded as some modular network, it must satisfy the next conditions:

1) Digraph G is weakly connected, i.e. undirected graph corresponded to G is connected;
2) Graph G has not multiple edges;
3) There is single input vertex $I \in V$ in G and every vertex is accessible from input: $\forall v \in V : v \neq I \Rightarrow \exists\, w(I,v)$ and therefore $\forall v \in V : v \neq I \Rightarrow s_{in}(v) \geq 1$
4) There is single output vertex $O \in V$ in G and output is accessible from every vertex: $\forall v \in V : v \neq O \Rightarrow \exists\, w(v,O)$ and therefore $\forall v \in V : v \neq O \Rightarrow s_{out}(v) \geq 1$

The last two conditions are obvious if we take into consideration that modular network are built for data processing. If network don't meet the condition 3 then there are modules that have not input data. Condition 4 guarantees that result of every module processing will reach output of network directly or after another modules' processing.

In contrast to the definition accepted in the graph theory we'll consider walk w from vertex a to vertex b as a set of vertices and edges not including final vertex. According to such definition we can pick out six walks on picture 2: $w_1(a,b)=\{a;e_1\}$; $w_2(a,c)=\{a;e_1;b;e_3\}$; $w_3(a,c)=\{a;e_1;b;e_2;b;e_3\}$; $w_4(b,b)=\{b;e_2\}$; $w_5(b,c)=\{b;e_3\}$ and $w_6(b,c)=\{b;e_2;b;e_3\}$. There is no walk $w(a,b)=\{a;e_1;b;e_2\}$ because vertex b cannot belong to the walk by our definition. Similar, path not includes final vertex. Correspondingly, there are four paths on picture 2: w_1, w_2, w_4 and w_5.



Pic. 2 Example of cycle

Specification of the walk definition pursues the next aim: if we consider subgraphs of modular network as a set of vertices and edges then exclusion of final vertex in the walk definition let us use difference operation without explicit indication of membership of vertices in the resulting set. That is, if we have walk represented by the set of vertices and edges $w(a,c)=\{a;e_1;b;e_2\}$, then this walk can be represented by two non-overlapping subsets $w_1(a,b)=\{a;e_1\}$ and $w_2(b,c)=\{b;e_2\}$, each of which is also a walk due to our definition.

In the subsequent text, we'll require two definitions: projection P of one module onto another and degree of module uncertainty U.

Definition 1: **Projection** $P(b,a)$ of vertex (module) $a \in V$ onto vertex (module) $b \in V$ is a directed subgraph of digraph $G(V,E)$ inclusive all possible **paths** $w(b,a)$ from b to a:

$$P(b,a) = \bigcup w(b,a), \quad \forall w(b,a) = \{v_1, v_2, \ldots, v_n\} : v_1 = b, v_{n+1} = a, \quad v_i \neq v_j \quad \forall i, j = \overline{1,n}.$$

That is this subgraph contains all vertices and corresponding edges of all paths form b to a, except vertex a according to our definition.

One of the most important projections is a projection of some module a onto network input I: $P(I,a)$.

Definition 2: If we associate every vertex $v \in V$ with a parameter $t(v)$: $t(v) = 1$ if outputs of corresponding network module are not computed at given run step and $t(v) = 0$ if module is already calculated, then **uncertainty** of vertex a will be defined as

$$U(a) = \sum_{v \in P(I,a)} t(v).$$

Uncertainty of input module $U(I)$ is always equal to zero. As input module is virtual we consider its outputs to be always calculated.

Necessity of such definition directly follows from goals putted by. Checking modular architectures on consistency and construction of sequence of network modules' run requires introducing of some comparison criterion, which of modules should be calculated first. Defined uncertainty can be such criterion.

Axiom 1: Module with less uncertainty should be calculated first if there are no special outer conditions.

Let's illustrate requirement of this axiom by the example of sequential modules' connection (picture 3). Outputs of input module I are determined by our definition and outputs of module a are not calculated at the beginning of modular network's run. Hence, the uncertainty of module a is equal to zero and uncertainty of module b is equal to one according to Definition 2. Therefore, according to Axiom 1, module a should be calculated before module b as it obvious for sequential modules' connection.

Pic 3. Sequential connection of modules

Necessity of Axiom 1 becomes even more evident for modular neural network training. If untrained module (neural network) a is found on a walk from input to module b, then training of module b is meaningless while network a is untrained.

Cycles in Modular Networks

Cycles are of special interest with relation to application tasks. If input data are organized as time series we mean "recurrent" cycles. If modules, contained in a cycle, realize associate fields or other algorithms that require several run iterations with one input vector then we mean "iterative" cycles. Hence, cycle can be recurrent or iterative.

It's necessary to introduce clock of modular network run for recurrent cycles description but for iterative cycles it's enough to define the number of cycle's round, where cycle's round is a single run of all modules contained in a cycle. In this article we discuss only general definitions related both to iterative and recurrent cycles.

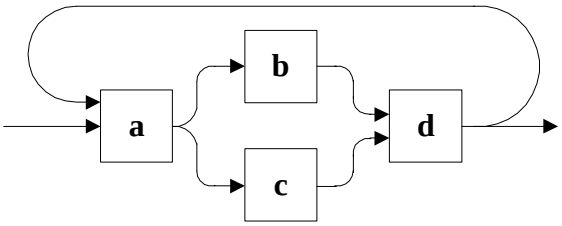
Primary types of cycles and basic definitions

First, we need to introduce general definition of cycle in modular network.

Definition 3: **Cycle** with first vertex a is a directed subgraph $C_a(V_a, E_a)$ of a modular network digraph $G(V, E)$ that contains all possible walks $w(a,a)$ from vertex a into itself:

$$C_a(V_a, E_a) = \bigcup w(a,a), \quad \forall w(a,a) = \{v_1, v_2, \ldots, v_n\} : v_1 = a, v_{n+1} = a$$

It should be noticed that walks but not paths are appearing in a cycle definition in contrast to definition of projection. Definition 3 generalizes cycle definition that is common in graph theory [5] in the sense that it includes all possible walks $w(a,a)$. It is not difficult to show that cycles according to Definition 3 correspond to strictly connected components in a digraph and have obvious property: $C_a \bigcap C_b = \varnothing$.



Pic 4. Simple cycle scheme

Architecture on picture 4 illustrates the necessity of definition of a cycle as a set of all walks. It's obvious that while modules' run in the cycle $C_a : V_a = \{a, b, c, d\}$ it's necessary to calculate both modules b and c before calculating module d. Note that Axiom 1 requires the same run sequence

It's directly follows from a cycle definition that virtual modules (input and output) can never belong to a cycle because input module has not incoming edges and output module has not outgoing edges. Moreover, according to the conditions imposed on a modular network graph, any cycle must have both incoming and outgoing external edges.

Definition 4: **First** vertex f of a cycle $C(V, E)$ we'll denote such vertex that $d_f = \min_{v \in V} \min_{w = w(I,v)} d(w)$, where $d(w)$ is a length of walk *w.* **Last** vertex l of a cycle is such vertex that $d(w(l, f)) = 1$.

Given definitions do not guarantee the uniqueness of the vertices but nevertheless such definition allows classifying cycles.

Lets introduce definitions for two types of cycles that fundamentally important from a point of modular networks' run.

Definition 5: Ordinary cycle with first vertex a is a cycle $Co_a(V_a, E_a)$ where there are no pair of vertices that belong to each other projections onto input, i.e. $\forall v_1, v_2 \in V_a : v_1 \in P(I, v_2) \quad \Rightarrow \quad v_2 \notin P(I, v_1)$. **Crossed** cycle with first vertex b is a cycle $Cc_b(V_b, E_b)$ that contains at least one pair of vertices belonging to each other projections onto input, i.e. $\exists v_1, v_2 \in V_b : v_1 \in P(I, v_2) \quad \& \quad v_2 \in P(I, v_1)$.

Introducing of term "ordinary cycle" is necessary for order determination of run sequences for modular network, and first of all for separation of all possible trigger schemes (picture 1). It can be easily shown that cycle on the picture 1 is crossed and has two first vertices.

Properties of cycles

The development of algorithms for automatic analysis is substantively simplified if such algorithms are built on the basis of inherent cycles' properties.

Proposition 1: For each vertex v contained in a cycle C_f one can find at least one walk $w(v,v)$ that contains first vertex f of a cycle.

This proposition is trivial but not always evident because cycle can contain arbitrary number of walks buy our definition.

Proof: Let v belongs to C_f, then

$\exists\, w(f,f) = w(f,v) \bigcup w(v,f) \quad \Rightarrow \quad \exists\, w(v,v) = w(v,f) \bigcup w(f,v)$ ■

Theorem 1: For every two vertices a and b of a cycle $C_f(V_f, E_f)$, projection $P(a,b)$ always exists and this projection is entirely belongs to a cycle:

$\forall a,b \in V_f : \exists\, P(a,b) \subseteq C_f$

Proof: Lets show that as long as vertices a and b belong to the cycle then at least one walk $\tilde{w}(a,b)$ exists. From Proposition 1, it is follows that

$a \in C_f \quad \Rightarrow \quad \exists\, w(a,a) = w(a,f) \bigcup w(f,a) \quad \Rightarrow \exists\, w(a,f)$

Then if $b \in w(a,f) \Rightarrow w(a,f) = \tilde{w}(a,b) \bigcup \tilde{w}(b,f)$. That is walk from a to b exists. If vertex b does not belong to the walk from a to f then as long as $b \in C_f \quad \Rightarrow \quad \exists\, w(b,b) = w(b,f) \bigcup w(f,b)$, therefore walk $\tilde{w}(a,b) = w(a,f) \bigcup w(f,b)$ exists.

Let walk $\tilde{w}(a,b)$ is not an elementary path. That is vertex $\tilde{v}$ is included in this walk at least two times:

$\tilde{w}(a,b) = \{a, v_1 ... v_i, \tilde{v}, ..., \tilde{v}, v_{i+k}, ..., v_n\}, v_{n+1} = b$. Then it is possible to construct a path from a to b:

$w(a,b) = \{a, v_1 ... v_i, \tilde{v}, v_{i+k}, ..., v_n\}, v_{n+1} = b$. That is at least one path exists $w(a,b)$ and $P(a,b) \neq \varnothing$.

Let's show that all paths from a to b belong to the cycle. Since both vertices belong to the cycle then $a \in C_f \Rightarrow \exists\, w_1(f,a)$ and $b \in C_f \Rightarrow \exists\, w_2(b,f)$. Then walk $\exists\, w(f,f) = w_1(f,a) \bigcup w(a,b) \bigcup w_2(b,f)$ exists. Hence, every path $w(a,b)$ belongs to the cycle as it contains all possible walks from first vertex to itself by the definition. ■

Theorem 2: (Sign of cycle existence) Graph of a modular network will contain cycle (cycles) if this graph contains at least one vertex having at least one edge not belonging to the projection of this vertex onto input.

Proof: Let network digraph has vertex a having incoming edge $e(b,a)$ not belonging to the projection of this vertex onto input. According to projection definition, this means that for the vertex b, from which this edge comes out, there are no walks from network input not containing vertex a. As by the conditions imposed on a graph, every vertex is accessible from the input, there is at least one walk $w(I,b)$ for the vertex b. Hence, if there is edge $e(b,a)$, then vertex a belongs to every walk

$$w(I,b) : \exists\, e(b,a) : e \notin P(I,a) \quad \Leftrightarrow \quad a \in w(I,b) \quad \forall\, w(I,b).$$

Then at least one walk from vertex a to itself exists: $w(a,a) = w(a,b) \bigcup e(b,a)$ and, hence, cycle exists. ■

It should be noticed that sign of cycle existence proposed in the Theorem 2 gives only sufficient condition for cycle existence. It allows finding all ordinary cycles in a modular network but only some crossed cycles.

Let's consider some important properties of ordinary cycles.

Theorem 3: (Theorem on uniqueness of ordinary cycle). If ordinary cycle exists, then the first vertex of this cycle can be selected uniquely.

Proof: Lets carry out the proof ex adverso. Let $C_{f1}(V_{f1}, E_{f1})$ is an ordinary cycle. Let there is two first vertices in this cycle, i.e. $\exists\, f2 \in V_{f1}, f2 \neq f1 : \quad d_{f1} = d_{f2} = \min_{w(I,v)}(d_v) \quad \forall v \in V_{f1}$ (from the definition of first vertex).

As minimal distances from the network input to the first vertices are equal then for the vertex $f1$ there is a walk from the network input not containing vertex $f2$, i.e. $\exists\, w_1(I, f1) : f2 \notin w_1$. In the same way, there is a walk $\exists\, w_2(I, f2) : f1 \notin w_2$, because $f2$ is also a first vertex. From the other side, as long as vertices $f1$ and $f2$ both belong to the cycle then according to the Proposition 1 there are walks $w_3(f1, f2)$ and $w_4(f2, f1)$. And according to a walk definition $f2 \notin w_3$ and $f1 \notin w_4$. Then next walks exist:

$(f1 \notin w_2 \;\&\; f1 \notin w_4) \Rightarrow \exists\, w(I,f1) = w_2(I,f2) \bigcup w_4(f2,f1)$ and

$(f2 \notin w_1 \;\&\; f2 \notin w_3) \Rightarrow \exists\, w(I,f2) = w_1(I,f1) \bigcup w_3(f1,f2)$.

The next follows from the definition of projection of a vertex onto input: $f1 \in P(I,f2) \quad \& \quad f2 \in P(I,f1)$. That is cycle $C_{f1}(V_{f1},E_{f1})$ is crossed by the definition and therefore there is only one first vertex in an ordinary cycle. ■

Consequence 3.1: If ordinary cycle Co_f exists: $V_f \neq 0$, then for a given set of vertices it can be determined uniquely.

Indeed, as long as all possible walks from first vertex of a cycle to itself are contained in the cycle by the definition, then every cycle is determined accurate to a first vertex. Since first vertex in an ordinary cycle is unique then cycle is determined unambiguously.

Theorem 4: There is the only vertex in an ordinary cycle connected by incoming edges with vertices outside the cycle and this vertex is the first vertex.

$v \in Co_f: \quad \exists\, e(p,v),\, p \notin Co_f \Rightarrow v = f$

Proof: Lets carry out the proof ex adverso. Let $C_f(V_f,E_f)$ is an ordinary cycle and there is a vertex v (except the first vertex) having outer incoming edge $e(p,v)$, i.e. $\exists\, v \in V_f, v \neq f: \quad \exists\, e(p,v),\, p \notin V_f$. Since every vertex is accessible from the input then walk $w(I,p)$ exists. Lets show that if vertex p does not belong to the cycle then for every walk $\forall\, w(I,p),\, p \notin V_f \Rightarrow f \notin w$.

Let walk exists that contains first vertex of the cycle: $\exists\, w(I,p): f \in w \quad \Rightarrow \quad w(I,p) = w(I,f) \bigcup w(f,p)$. Correspondingly, walk $w(f,p)$ exists. According to the initial assumption $v \in V_f \Rightarrow \exists\, w(v,f)$ and edge $e(p,v)$ exists. Hence, walk from the first vertex to itself through vertex p exists: $w(f,f) = w(f,p) \bigcup e(p,v) \bigcup w(v,f)$. Therefore, $p \in V_f$ that conflicts with initial condition.

As first vertex of the cycle does not belong to a walk from the input to the vertex p then there is a walk from the network input to the vertex v not containing first vertex f: $w_2(I,v) = w(I,p) \bigcup e(p,v)$. Also there is a walk $\exists\, w_1(I,f): v \notin w_1$, because f is the first vertex. On the other hand, both vertices f and v belong to the cycle and according Proposition 1 there are walks $w_3(f,v)$ and $w_4(v,f)$. According to a walk definition $v \notin w_3$ and $f \notin w_4$. Then next walks exist:

$(f \notin w_2 \;\&\; f \notin w_4) \Rightarrow \exists\, w(I,f) = w_2(I,v) \bigcup w_4(v,f)$ and

$(v \notin w_1 \;\&\; v \notin w_3) \Rightarrow \exists\, w(I,v) = w_1(I,f) \bigcup w_3(f,v)$.

From the existence of such walks and definition of a projection of a vertex onto input the next follows: $f \in P(I,v) \quad \& \quad v \in P(I,f)$. Because of this, the cycle $C_f(V_f,E_f)$ is crossed that conflicts with condition. Hence, there is the only vertex connected by incoming edges with vertices outside the ordinary cycle. ■

Consequence 4.1: For an ordinary cycle projection of first vertex into network input does not contain any vertex of the cycle.

Proof: Let there is a vertex in the ordinary cycle $C_f(V_f,E_f)$ contained in the project of its first vertex onto input: $\exists v \in V_f: \; v \in P(I,f)$. From the definition of a projection of a vertex onto input we'll find that walk $w(I,f): v \in w \;\&\; f \notin w$ exists and correspondingly this walk can be broken down into two: $w(I,f) = w(I,v) \bigcup w(v,f) \quad \Leftrightarrow \quad \exists w(I,v): f \notin w$. Also, there is the shortest walk $\exists\, w(I,f): v \notin w$, because f is the first vertex. By analogy with the proof of the Theorem 4, we'll find that the cycle is crossed. ■

Consequence 4.1: For an ordinary cycle C_f, all paths from the input contain the first vertex for every vertex $v \neq f$, i.e.:

$$P(I,v) = P(I,f) \bigcup P(f,v) \quad \forall v \in C_f$$

Proof: As every vertex is accessible from the input and according to the Theorem 4 none of the cycle's vertices have outside edges (except the first); therefore all paths (walks) from the input to vertices of the ordinary cycle contain the first vertex. ■

Conclusion and Further Work

Given definitions are related mainly to the architectures of modular networks containing cycles. It's obvious that it's necessary to have some assumptions about rules and sequence of cycle's traversal for defining the run sequence of such architectures. Properties of considered types of cycles could be used as a basis of such rules.

Besides the classification based on architectures' types, cycles vary in types of processed data and used algorithms of neural networks. Cycles' types regarded above allow to take into consideration the specificity of processed data and neural algorithms in the form of corresponding modular architectures.

General cycles' properties, considered in this article, allow formal analyzing of modular neural networks. Use of cycle's properties allows constructing algorithms for analysis of architectures with automatic system. Moreover, automatic search of all modules contained in a cycle allows formal attaching of some properties to every cycle. This turn out to be very helpful in CAD systems where user can, first, specify run sequence and, second, define each cycle as recurrent or iterative.

Data processing with a help of modular neural network means that run sequence is determined. The definitions given above are assigned, first of all, for derivation of rules for automatic construction of run sequence and for the controlling of architectures created by user. Cycles' properties allow determining, which architectures of modular networks are valid without introducing additional conditions.

Proposed theory was successfully used in the subsystem of automatic analysis of modular architectures in the MNN CAD [2]. Formal definitions of valid and forbidden architectures of modular networks and ways of settlement of contradictions in forbidden architectures, algorithms of modular networks analysis, based on the properties of proposed graphs, will be subject of future works.

Bibliography

A. Galinskaya. Modular neural networks: the review of modern state of the developments. In: Mathematical Machines and Systems.-2003. -№ 3,4. –C.87-102.

1. M. Kussul, A. Riznyk, E. Sadovaya, A. Sitchov, Tie-Qi Chen. “A visual solution to modular neural network system development”, Proceedings of the 2002 International Joint Conference on Neural Networks IJCNN'02, Honolulu, HI, USA, 12-17 May 2002, vol.1
2. M. Kussul, A. Galinskaya. Comparative study of modular classifiers and its training // Proc. of the X-th International Conference “Knowledge – Dialog - Solution” KDS 2003, Varna, Bulgaria, 16-26 June 2003. –P.168-174.

A. Galinskaya. Architectures and training of modular classifiers for applied tasks In: Mathematical Machines and Systems. -2003. -№ 2. –C.77-86.

3. R. Diestel. Graph Theory. 2nd edition, Springer-Verlag, New York, 2000, P. 312

Authors' Information

Michael Kussul – PhD., Institute of Mathematical Machines and Systems, NASU, Glushkova ave. 42, 03680, Kiev, Ukraine. kussul@nnteam.org.ua

Alla Galinskaya – PhD. student, Institute of Mathematical Machines and Systems, NASU, Glushkova ave. 42, 03680, Kiev, Ukraine. alla@nnteam.org.ua

ГЕТЕРОГЕННЫЕ ПОЛИНОМИАЛЬНЫЕ НЕЙРОННЫЕ СЕТИ ДЛЯ РАСПОЗНАВАНИЯ ОБРАЗОВ И ДИАГНОСТИКИ СОСТОЯНИЙ[1]

Адиль В.Тимофеев

Аннотация: *Рассмотрены различные параллельные архитектуры и методы самоорганизации и минимизации сложности гетерогенных полиномиальных нейронных сетей (ПНС) в задачах распознавания образов и диагностики состояний. Получены конструктивные оценки степени гетерогенности и параллелизма в процессе автономного принятия классифицирующих решений с помощью ПНС различных типов. Показано, что параллелизм, самоорганизация и робастность гетерогенных ПНС могут значительно возрасти при коллективном (мульти-агентном) решении сложных задач распознавания образов, анализа изображений, развернутой (векторной) диагностики состояний и адаптивной маршрутизации информационных потоков*

Введение

Одним из наиболее эффективных средств массового распараллеливания и ускорения процессов обработки и передачи потоков данных в задачах распознавания образов, классификации данных и диагностики состояний являются искусственные нейронные сети (НС) и нейросетевые технологии. Естественным прототипом искусственных НС является биологический мозг и центральная нервная система человека и животных как сложная гетерогенная нейронная сеть, обеспечивающая высокую степень параллелизма, самоорганизации и робастности при решении различных интеллектуальных задач (распознавание образов, классификация данных, поиск закономерностей, анализ изображений, диагностика состояний, прогнозирование явлений и т.п.). Возможности искусственных и биологических НС могут значительно расшириться при коллективном (мульти-агентном) решении сложных интеллектуальных задач.

1. Основные идеи и принципы построения гетерогенных полиномиальных нейронных сетей

Высокая сложность и размерность многих задач распознавания образов, классификации данных, анализа изображений и диагностики состояний, а также часто возникающая необходимость их решения в реальном времени требуют массового параллелизма и самоорганизации распределённых вычислений на базе НС. С этой точки зрения особый интерес представляют гетерогенные полиномиальные нейронные сети (ПНС) с самоорганизующейся архитектурой и их разновидности, предложенные в работах [1–9].

Основные идеи, математические модели, методы обучения и принципы самоорганизации гетерогенных ПНС были описаны в [1–3] и развиты в [4–9]. К ним прежде всего относится следующее:

- архитектура ПНС гетерогенна и многослойна;
- наличие слоя полиномиальных нейронных элементов (П-нейронов);
- возможность и целесообразность самоорганизации архитектуры ПНС различных типов;
- детерминированные и вероятностные методы обучения и самоорганизации гетерогенных ПНС;
- принципы минимальной сложности и высокой экстраполяции гетерогенных ПНС;
- алгебраическое требование диофантовости (целочисленности синаптических весов) гетерогенных ПНС.

В процессе дальнейшего развития теории гетерогенных ПНС были предложены модели многозначных нейронных элементов (М-нейронов) и связанных с ними конъюнктивных, полиномиальных, дизъюнктивных и суммирующих нейронных элементов (МК-, МП-, МД- и МΣ-нейронов), а также новые разновидности гетрогенных ПНС (генно-нейронные сети, квантовые, нейронные сети, мульти-агентные ПНС и т.п.), описанные в [4–14].

В настоящей статье анализируются гетерогенность архитектуры, возможности самоорганизации распределённых вычислений и степень параллелизма ПНС различных типов, предназначенных для распознавания образов, диагностики состояний и решения других интеллектуальных задач (data mining, knowledge discovery и т.п.). Полученные теоретические (априорные) оценки степени параллелизма и самоорганизации гетерогенных ПНС подтверждаются экспериментальными результатами решения прикладных задач. Значительный интерес представляют также новые проблемы организации коллективных (мульти-агентных) решений на базе гетерогенных ПНС.

2. Задачи классификации и идентификации образов и их обобщение

ПНС с гетерогенной архитектурой предназначены для решения сложных интеллектуальных задач. Примерами таких задач могут служить задачи распознавания образов, классификации данных, диагностики состояний, идентификации объектов и т.п. Обычно эти задачи формулируются следующим образом.

Пусть задано конечное множество объектов $\{\omega\} = \Omega$ и существует (но неизвестно) его разбиение на К непустых подмножеств (классов) вида

$$\Omega = \bigcup_{k=1}^{K} \Omega_k, \Omega_k \neq \varnothing, \Omega_k \cap \Omega_j = \varnothing \text{ при } k \neq j. \quad (1)$$

Это означает, что существует неизвестная классифицирующая функция R(ω), ставящая в соответствие каждому объекту $\omega \in \Omega$ номер класса k, к которому он принадлежит, т.е.

$$R(\omega) = k, \omega \in \Omega_k, k = 1,2,...,K. \quad (2)$$

Кроме того, существует множество неизвестных идентифицирующих функций вида

$$R_k(\omega) = \begin{cases} 1, \text{если} \omega \in \Omega_k, \\ 0 - \text{в противном случае}, \end{cases} \quad (3)$$

являющихся характеристическими функциями классов Ω_k, k = 1,2,…,K, которые отделяют все объекты $\omega \in \Omega_k$ от остальных.

Предположим, что имеется некоторая измерительная система (датчики информации, сенсоры, измерительные приборы и т.п.), которая в результате измерения свойств или определения характеристик любого объекта $\omega \in \Omega$ однозначно ставит в соответствие объекту ω его информационное описание в виде вектора признаков $\mathrm{x}(\omega) = \left| x_i(\omega) \right|_{i=1}^{n}$. Тогда векторная функция x: $\Omega \to R^n$ определяет отображение множества объектов Ω в n-мерное пространство признаков R^n. Это отображение порождает разбиение описаний объектов в пространстве признаков R^n на классы (образы) вида

$$X_k = x(\Omega_k), k = 1,2,...,K.$$

Отображение x: $\Omega \to R^n$ будем называть корректным или информативным, если выполняются следующие условия:

1) если $x(\omega_i) = x(\omega_j)$, то $R(\omega_i) = R(\omega_j)$, (4)

т.е. объекты ω_i и ω_j с одинаковым описанием принадлежат одному классу,

2) если $R(\omega_i) \neq R(\omega_j)$, то $x(\omega_i) \neq x(\omega_j)$, (5)

т.е. объектам ω_i и ω_j из разных классов соответствуют различные описания.

Таким образом, корректное отображение объектов в пространство признаков не приводит к потере информации о классах. Это информативное отображение определяет следующее разбиение описаний объектов на классы (образы)

$$\mathrm{X} = \bigcup_{k=1}^{K} \mathrm{X}_k, \ \mathrm{X}_k = \mathrm{x}(\Omega_k) \neq \varnothing. \quad (6)$$

Из соотношений (1) – (6) следует, что если $\omega \in \Omega_k$, то $x(\omega) \in X_k$, и наоборот. В том случае, когда $|\Omega| > |X|$, корректное (информативное) отображение $x(\omega)$ является сжимающим.

Будем считать, что отображение $x: \Omega \to R^n$ является информативным и, возможно, сжимающим. Тогда задачи классификации и идентификации образов сводятся к восстановлению (определению) неизвестных решающих функций вида (2) и (3). При этом единственной доступной информацией о классах (1) являются табличные базы данных вида

$$\{x(\omega_h), R(\omega_h)\}_{h=1}^{m} \text{ или } \{x(\omega_h), R_1(\omega_h), R_2(\omega_h), ..., R_K(\omega_h)\}_{h=1}^{m}, \quad (7)$$

называемые обучающей выборкой.

Мощность этой выборки $m \geq K$ должна быть достаточно большой, т.е. множество (7) должно быть репрезентативным. Поэтому будем предполагать, что множество (7) содержит достаточную информацию об априорном разбиении объектов (1) и их описании (6) на классы (образы). В частности, важно, чтобы обучающее множество (7) было полным, т.е. содержало хотя бы по одному объекту из каждого класса, и непротиворечивым, т.е. не содержало одинаковых описаний, относящихся к объектам из разных классов.

Наряду с классическими задачами классификации и идентификации образов с помощью скалярных решающих функций вида (2) и (3), значительный интерес для практики представляет их обобщение на случай векторного распознавания образов и диагностики состояний. Примером таких задач могут служить сложные задачи медицинской диагностики, когда требуется для каждого больного $\omega \in \Omega$ с вектором симптомов признаков $x(\omega)$ определить не только диагноз, т.е. класс заболеваний Ω_k, к которому он относится, но и дать его "расшифровку", т.е. найти ряд уточняющих и детализирующих характеристик в виде вектора

$$z(\omega) = |z_j(\omega)|_{j=1}^{q}, \quad (8)$$

где $z_j(\omega)$ - многозначные предикаты, принимающие целочисленные значения в интервале $[0, p_j]$.

Формально развернутый (детализирующий) диагноз можно представить в виде (q+1) – мерного вектора

$$L(\omega) = |R(\omega), z_1(\omega), z_2(\omega), ..., z_q(\omega)|, \quad (9)$$

Компонентами этого вектора является классифицирующий предикат $R(\omega)$ с целочисленными значениями (кодами) основного диагноза из интервала [1,2,...,K] и детализирующие этот диагноз дополнительные многозначные предикаты $z_j(\omega)$, j= 1,2,...,q, каждому из которых можно поставить в соответствие локальную "расшифровку" основного диагноза на естественном языке.

При обобщенной векторной классификации образов главный предикат $R(\omega)$ описывает априорное разбиение множества Ω на классы (1), а дополнительные предикаты $z_j(\omega)$ определяют разбиение каждого класса Ω_k на подклассы вида:

$$\Omega_k = \bigcup_{j=1}^{q} \Omega_{k,j} . \quad (10)$$

Предикаты $R(\omega)$ и $z_j(\omega)$ заранее неизвестны. Однако известны из значения на обучающей выборке вида

$$\{x(\omega_h), R(\omega_h), z_1(\omega_h), z_2(\omega_h), ..., z_q(\omega_h)\}_{h=1}^{m}. \quad (11)$$

По этим данным требуется восстановить (определить) неизвестную вектор-функцию (9) и её компоненты (2) и (8). Некоторые методы решения этой задачи векторной (расширенной) классификации образов на базе обучения и самоорганизации гетерогенных ПНС предложено в работах [11 – 13].

3. Нейросетевые архитектуры и алгоритмы

В настоящее время известно много математических и эвристических подходов к решению сформулированных задач классификации и идентификации образов. Среди них важную роль играют нейросетевые подходы, основанные на синтезе различных моделей (архитектур) и алгоритмов обучения НС для распознавания образов и диагностики состояний.

Однако во многих случаях эти НС имеют гомогенную архитектуру. При этом заранее не известно, какое число слоёв и нейроэлементов необходимо для решения задачи. Алгоритмы обучения таких гомогенных НС не всегда сходятся к решению задачи за конечное число шагов.

Примерами гомогенных НС могут служить однослойные НС Хопфилда или Хемминга или многослойные перцептроны, использующие пороговые или сигмоидальные нейроэлементы. Популярные сегодня градиентные алгоритмы обучения гомогенных НС типа Backpropagation и его модификации (Quick propagation, Rpro и т.п.) медленно сходятся или вообще не приводят к решению за конечное время.

Описываемые ниже гетерогенные ПНС различных типов являются эффективным средством восстановления (определения) неизвестных классифицирующих и идентифицирующих функций (2) и (3) по обучающим базам данных (7) и их реализации на базе ПНС с самоорганизующейся архитектурой минимальной сложности. В основе теории этих гетерогенных ПНС лежат идеи и принципы, сформулированные выше, а также в работах [1 – 9].

Предлагаемые гетерогенные ПНС позволяют также решать обобщенные задачи векторной классификации и описания образов, т.е. восстанавливать (определять) неизвестные векторные функции (9) по расширенным обучающим выборкам вида (11). Кроме того, они могут успешно использоваться в качестве нейросетевых агентов при коллективном (мульти-агентном) решении сложных (глобальных) задач распознавания образов, расширенной (векторной) диагностики и адаптивной маршрутизации информационных потоков [7 –14].

4. Гетерогенность, параллелизм и самоорганизация порогово-полиномиальных нейронных сетей

Архитектура ПНС гетерогенна и представляет собой последовательность нескольких однородных слоёв (непересекающихся подмножеств) параллельно работающих нейроэлементов (НЭ) различных типов. В различных слоях гетерогенной ППС могут использоваться разные НЭ, но каждый слой (подмножество НЭ) является однородным (гомогенным). При этом обработка информации в каждом слое НЭ осуществляется параллельно.

Каналы связи между предыдущим и последующим слоями гетерогенной ПНС являются однонаправленными (односторонними) и имеют регулируемые веса (синаптические параметры). Эти веса каналов связи настраиваются в процессе обучения и самоорганизации архитектуры ПНС по имеющимся экспериментальным данным или прецедентам вида (7), называемым обучающей базой данных (ОБД).

Опишем формально гетерогенную архитектуру трёхслойной порогово-полиномиальной нейронной сети (ППНС) [1-4] и рассмотрим принципы её самоорганизации.

Первый слой ППНС состоит из n пороговых нейроэлементов (НЭ), на вход которых параллельно поступают сигналы $y_1(\omega)$, …, $y_p(\omega)$, характеризующие различные свойства объекта ω, а на выходе формируется вектор двоичных сигналов (бинарный код) $x(\omega)=\left|x_i\right|_{i=1}^{n}$, где $x_i=x_i(y_1,\ldots y_p)$. Этот бинарный код является выходом НЭ первого слоя ППНС.

На входы каждого “ассоциативного” НЭ второго слоя поступает вектор $x(\omega)$ бинарных сигналов с “рецепторных” (кодирующих) пороговых НЭ первого слоя. Эти “ассоциативные” НЭ реализуют полиномиальные преобразования $a_j(x)$ входных сигналов и называются П-нейронами или мультипликативными НЭ. Выходом второго слоя ППНС является вектор двоичных сигналов $a(x)=\left|a_j(x)\right|_{j=1}^{N}$.

Третий слой ППНС состоит из “решающих” пороговых НЭ, на вход которых поступает вектор выходных сигналов a_j(x) полиномиальных НЭ второго слоя. Затем эти сигналы умножаются на синаптические веса u_j, суммируются и преобразуются в выходные сигналы НЭ третьего слоя в виде суперпозиций функций

$$R_k(\omega) = sign\ (\sum_{j=1}^{N} u_{k,j} a_j(\mathrm{x}(y(\omega)))),\ k = 1, \ldots \mathrm{K}, \tag{12}$$

где $u_{k,j}$ – настраиваемые синаптические параметры, а K – число непересекающихся классов (образов).

Каждая решающая (идентифицирующая) функция $R_k(\omega)$ вида (8) является характеристической функцией k-го класса. Поэтому она отделяет объекты k-го класса от остальных. Множество таких идентифицирующих функций (8) решает задачу классификации образов.

Принцип самоорганизации ППНС с гетерогенной архитектурой заключается в построении “ассоциативных” полиномов a_j(x) непосредственно по обучающей базе данных (ОБД) (7) в виде одночленов [1–3]

$$a_j(\mathrm{x}(y)) = \prod_{i=1}^{n} x_i^{\,x_i(y_j)}(y),\ j = 1,\ldots,m \tag{13}$$

Здесь m – число элементов (мощность) ОБД, определяющее оценку сверху на число N необходимых полиномиальных НЭ второго слоя ППНС, т.е. $m \geq \mathrm{N} \geq \mathrm{K}$.

Чтобы минимизировать сложность ППНС, нужно найти в идентифицирующих функциях вида (8) векторы синаптических параметров $u_k = \left| u_{k,j} \right|_{j=1}^{N}$ с максимальным числом нулевых компонент. Это приводит к автоматической ликвидации неинформативных (избыточных) синаптических связей в гетерогенной ППНС без потери точности принимаемых решений на ОБД (7).

Быстрые рекуррентные алгоритмы обучения и минимизации сложности гетерогенной ППНС были предложены в [1–4]. Они осуществляют отображение ОБД вида (7) на множество синаптических параметров пороговых НЭ третьего слоя ППНС, причём число шагов алгоритма обучения $r \leq m$.

Степень параллелизма при распознавании и идентификации образов определяется тем, что принятие решений осуществляется ППНС за 3 такта одновременных вычислений в каждом слое НЭ независимо от размерности решаемой задачи $D=n \times m \times K$.

Самоорганизация ППНС обеспечивается тем, что НЭ второго слоя формируются согласно (9) непосредственно по ОБД (7).

Гетерогенность архитектуры ППНС определяется тем, что первый и третий слой состоят из пороговых НЭ, а второй слой – только из полиномиальных НЭ.

5. Гетерогенность, параллелизм и самоорганизация диофантовых и сплайновых нейронных сетей

Гетерогенные ПНС с целочисленными синаптическими параметрами называются диофантовыми НС [1,5]. Примерами диофантовых НС с самоорганизующейся архитектурой, могут служить гетерогенные ПНС, предложенные в [1–3].

Рассмотрим другой класс диофантовых НС. Гетерогенность их архитектуры определяется использованием слоёв, состоящих из полиномиальных или сплайновых НЭ.

Первый слой диофантовой НС состоит из пороговых НЭ. Выходом первого слоя НЭ является двоичный образ (описание) x(ω) объекта ω.

НЭ второго слоя кодируют двоичные образы x(ω) натуральными числами вида

$$d(\omega) = \sum_{i=1}^{n} x_i(\omega) \cdot 2^{n-i}. \tag{14}$$

Для упрощения алгоритмов обучения и самоорганизации гетерогенной архитектуры диофантовой НС построим НЭ третьего слоя в виде ортогональных полиномов (одночленов) вида

$$a_j(d(\omega))=\prod_{\substack{h=1\\ j\neq h}}^{m_k}(d(\omega)-d(\omega_h))(d(\omega_j)-d(\omega_h))^{-1}. \quad (15)$$

Другие способы построения НЭ третьего слоя предложены в [5].

Четвёртый слой гетерогенной диофантовой НС состоит из “решающих” пороговых НЭ с синаптическими параметрами, настраиваемыми по ОБД (7) согласно быстрым алгоритмам обучения, предложенным в [5].

Описанные диофантовые НС целесообразно применять на сравнительно коротких ОБД вида (7). Чтобы снять это ограничение, рассмотрим гетерогенные сплайновые НС.

Сплайновые НС основаны на кусочно-полиномиальной или сплайновой аппроксимации решающих (идентифицирующих) функций вида (8). Идея состоит в синтезе НЭ по ОБД (7) в виде независимых друг от друга полиномов или сплайнов на каждой паре натуральных чисел $\{d(\omega_i),d(\omega_{i+1})\}$ и их коммутации по определённым правилам в четвертом слое НС, состоящем из пороговых НЭ с настраиваемыми по ОБД (7) синаптическими параметрами [5].

Синтезированные диофантовые и сплайновые НС имеют четырёхслойную гетерогенную архитектуру, описываемую суперпозицией параллельных преобразований в каждом слое НЭ.

Самоорганизация и минимизация сложности гетерогенной архитектуры таких НС обеспечиваются самонастройкой полиномиальных и сплайновых НЭ второго и третьего слоёв (например, вида (10) и (11)) и быстрыми алгоритмами обучения пороговых НЭ четвертого слоя, требующими только однократного использования ОБД вида (7), т.е. $r \le m$.

При распознавании образов и диагностике состояний принятие решений с помощью описанных диофантовой или сплайновой ПНС осуществляется за 4 такта параллельных вычислений в каждом слое НЭ независимо от размерности задачи D = n × m × K.

Гетерогенность архитектуры диофантовых и сплайновых НС определяется тем, что первый и четвертый слой состоят из пороговых НЭ, а второй и третий слой включают полиномиальные или сплайновые НЭ.

6. Гетерогенность, параллелизм и самоорганизация классифицирующих полиномиальных нейронных сетей

Рассмотрим четырёхслойную ПНС, предназначенную для классификации образов.

Первый слой этой ПНС состоит из функциональных НЭ (F-элементов) $F_i(y(\omega),\gamma_i)$ с синаптическими параметрами γ_i.

Второй слой этой гетерогенной ПНС состоит из полиномиальных НЭ (П–элементов) вида

$$a_j(\omega)=\prod_{i\in I_j}F_i(y(\omega),\gamma_i), \quad (16)$$

где $F(y,\gamma)=0$ при $y<\gamma$ и $F(\gamma,\gamma)\neq 0$ [6,9].

Третий слой ПНС состоит из одного суммирующего НЭ (Σ –нейрона).

Четвёртый слой ПНС состоит из одного многозначного НЭ (М-нейрона), описываемого К-значным предикатом М, принимающем значения 1,2,…,К и определяющим принадлежность объекта ω к одному из классов Ω_k.

Четырёхслойная архитектура ПНС реализует следующее последовательно-параллельное преобразование вектора входных сигналов $y(\omega)$ в выходной целочисленный сигнал $R(\omega,u,\gamma)$ вида

$$R(\omega,u,\gamma)=M(u_0+\sum_{j=1}^{N}u_j\prod_{i\in I_j}F_i(y(\omega),\gamma_i)), \quad (17)$$

определяющий номер класса, к которому будет отнесён распознанный объект ω.

Задачи обучения, минимизации сложности и самоорганизации гетерогенных классифицирующих ПНС с аналитическим описанием вида (16),(17) заключаются в определении скалярных функций y,F_i и векторов

синаптических параметров $\gamma = \left|\gamma_i\right|_{i=1}^{I_j}$ и $u = \left|u_i\right|_{j=1}^{N}$ по ОБД (7) таким образом, чтобы обеспечивалась не только безошибочная классификация объектов из ОБД (7), но и других распознаваемых (контрольных) объектов. Для решения этой задачи нужно конструктивно задать функциональные НЭ $F_i(y, \gamma_j)$ и векторы синаптических параметров γ и u с возможно меньшим числом ненулевых компонент [5-7].

Степень параллелизма в классифицирующих ПНС, описываемых многозначным предикатом (17) определяется тем, что распознавание образов, классификация данных или диагностика состояний осуществляются за 4 такта параллельных вычислений в НЭ разных слоёв независимо от сложности решаемой задачи $D = n \times m \times K$.

Гетерогенность архитектуры этих ПНС характеризуется тем, что первый слой состоит из функциональных НЭ, второй слой включает полиномиальные НЭ, а третий и четвертый слои содержат по одному суммирующему и многозначному НЭ.

7. Гетерогенность и параллелизм в генно-нейронных сетях с самоорганизующейся архитектурой

Пусть имеется популяция $\{\omega\} = \Omega$ особей ω, каждая из которых может принадлежать одному из К образов (классов) согласно разбиению (1). Особь ω характеризуется набором признаков x (ω), которые принимают значения, соответствующие одному из дискретных состояний генов $x_i(\omega)$. При этом состояния каждого гена описываются многозначным предикатом.

Вектор $x(\omega) = \left|x_i(\omega)\right|_{i=1}^{n}$ состояний генов назовём хромосомой (локальным описанием) особи ω. Будем говорить, что две особи имеют одинаковый генотип, если у них совпадают состояния всех генов, т.е.

$$x_i(\omega_j) = x_i(\omega_h),\ \forall i = 1,2,...,n, j \neq h. \tag{18}$$

Множество хромосом образует класс генотипов, соответствующих некоторому генетическому образу с характеристической функцией $R_k(\omega)$ вида (3).

Задачи генетического описания и распознавания образов сводятся к аппроксимации неизвестных функций $R(\omega)$, $R_k(\omega)$ по генетическим (целочисленным) ОБД вида (7). Для решения этих задач можно использовать трёхслойную гетерогенную генно-нейронную сеть (ГНС) полиномиального типа, описанную в [7].

Обученные ГНС обеспечивают массовый параллелизм при обработке генетических данных, а именно: принятие оптимальных решений при распознавании образов или диагностике состояний осуществляется в ГНС за 3 такта параллельных вычислений в НЭ каждого слоя независимо от размерности задачи $D = n \times m \times K$.

Гетерогенность архитектуры ГНС проявляется в том, что первый и третий слои состоят из пороговых НЭ, а второй слой содержит полиномиальные НЭ.

Другой метод обучения и самоорганизации многослойных ГНС основан на логико-вероятностных генетических алгоритмах селекции информативных генов и конъюнктивных НЭ (К-нейронов) вида

$$a_j^n(\omega) = \mathop{\&}_{i=1}^{n} \xi_i^{\xi_i(\omega_j)},\ j = 1,\ldots, m, \tag{19}$$

а также логико-вероятностных решающих (идентифицирующих) правил импликативного типа

$$\{a_j^r(\omega) \xrightarrow{P_k} R_k(\omega)\}_{i=1}^{N}, 1 \le r_1 \le \ldots \le r_N < n. \tag{20}$$

Здесь P_k - максимальная апостериорная вероятность принадлежности особи ω к k-ому классу Ω_k, оцененная по ОБД вида (7).

Синтезированные генетические решающие (идентифицирующие) правила в случае, когда i-ый ген может иметь только два состояния, можно представить в виде многослойной ГНС минимальной сложности, представляющий собой бинарное “дерево решений”. Ветвям этого дерева соответствует К-нейроны вида (19), а листьям - номера классов генотипов [3,7].

В работах [7 –9] представлено обобщение описанных ГНС на случай, когда каждый ген имеет не два, а произвольное число дискретных состояний, т.е. ген описывается многозначным предикатом вида

$$x_i(\omega) \in \{0,1,...l_i\}, l_i \geq 2. \quad (21)$$

Для увеличения параллелизма в процессе принятия оптимальных (в смысле критерия Байеса) решений описанные многослойные ГНС с древовидной архитектурой можно преобразовать в трёхслойные диофантовые ГНС с целочисленными синаптическими весами с помощью методов, описанных в [3,7]. В этом случае распознавание образов или диагностика состояний произойдут не за $r \leq r_N$ шагов согласно (19) и (20), а за 3 такта параллельных целочисленных вычислений в каждом слое НЭ независимо от размерности задачи D = $n \times m \times$ K.

Гетерогенность архитектуры синтезированной диофантовой ГНС характеризуется тем, что первый и третий слои состоят из пороговых НЭ, а второй слой включает конъюнктивные НЭ вида (19).

8. Параллелизм и самоорганизация в мульти-агентных нейронных сетях с гомогенной или гетерогенной архитектурой

Традиционно гомогенные или гетерогенные НС используются для автономного принятия решений в задачах распознавания образов, диагностики состояний, классификации данных и т.п. По существу эти НС являются обучаемыми интеллектуальными агентами, которые настраиваются на индивидуальное (одно-агентное) решение конкретных задач по ОБД вида (7).

В то же время существует большой класс интеллектуальных задач, требующий не только индивидуальных (одно-агентных), но и коллективных (мульти-агентных) решений. Классическим примером этого могут служить особенно сложные и ответственные задачи медицинской диагностики, когда врачи вынуждены прибегать к помощи своих коллег для совместной постановки окончательного диагноза. При этом формируется “консилиум”, т.е. профессиональная группа врачей, интегрирующая знания и опыт входящих в неё членов для коллективного принятия наиболее правильных и сбалансированных диагностических решений.

Другим примером сложных задач, требующих коллективных решений, являются глобальные задачи, допускающие естественную (например, иерархическую или мультифрактальную) декомпозицию на множество локальных задач. В этом случае решение сложной (глобальной) задачи может быть распределено между интеллектуальными НС-агентами, специализирующимися на решении M частных (локальных) задач. Параллельная работа M таких НС-агентов может значительно ускорить обработку информации и повысить надежность решения общей (глобальной) задачи.

В роли интеллектуальных агентов могут выступать гомогенные или гетерогенные НС различных типов. Однако они должны быть взаимосвязаны с помощью каналов обмена информацией в процессе принятия коллективных решений. В этом случае можно создать мульти-агентную (глобальную) систему обработки и передачи информации, интегрирующую в себе возможности входящих в неё локальных НС как агентов.

Архитектура таких мульти-агентных НС может быть гомогенной или гетерогенной. В гомогенной архитектуре в качестве агентов используется НС одного типа. Например, это могут быть гомогенные НС-агенты типа “перцептрон” или гетерогенные диофантовые ПНС. В гетерогенной архитектуре используются НС-агенты различных (смешанных) типов. Например, они могут содержать разные типы гетерогенных ПНС или могут иметь специальных агентов-координаторов, организующих целенаправленную работу локальных НС-агентов.

Агенты-координаторы могут принимать коллективные (мульти-агентные) решения на основе локальных (одно-агентных) решений остальных НС как автономных агентов с помощью мажоритарных принципов или процедур голосования (например, по “большинству голосов”) [11 – 14]. При этом все локальные решения принимаются параллельно, что ускоряет принятие глобального (коллективного) решения в M раз.

В ряде случаев глобальная самоорганизация НС-агентов обеспечивается иерархической, фрактальной или мультифрактальной декомпозицией общей задачи на M подзадач. При этом степень внешнего (глобального) параллелизма в мульти-агентной нейросетевой системе определяется параметром M, характеризующем одновременную работу M локальных НС-агентов, каждый из которых обладает

внутренним (локальным) параллелизмом при решении интеллектуальных задач, характеризующимся числом слоёв НЭ.

Мульти-агентное распознавание сложных 2D-изображений или 3D-сцен в ряде случаев основывается на их декомпозиции на самоподобные (фрактальные) компоненты и на обучении и самонастройке локальных гетерогенных ПНС на распознавание фрагментов по локальным ОБД, характеризующим эти фрагменты. В результате внутренней и внешней самоорганизации ПНС как НС-агентов достигается высокая степень параллелизма в процессе распознавания и анализа сложных изображений и сцен.

Необходимость в использовании НС - агентов и мульти-агентных технологий возникает в глобальных телекоммуникационных и компьютерных системах [10]. В этом случае НС-агенты обучаются и самоорганизуются по локальным ОБД вида (7) и передают по каналам связи накопленные “нейрознания” и “опыт” другим НС-агентам. Для эффективного (в частности, оптимального) управления потоками данных между удалёнными сетевыми пользователями как внешними агентами (клиентами) и локальными НС как внутренними агентами целесообразно использовать нейросетевые маршрутизаторы потоков данных[10], позволяющие адаптироваться к непредсказуемым изменениям структуры и праметров глобальных телекоммуникационных и компьютерных систем в процессе их функционирования в реальном времени.

Заключение

Описанные гетерогенные архитектуры и быстрые алгоритмы обучения ПНС разных типов обеспечивают высокий параллелизм и самоорганизацию нейровычислений в процессе решения интеллектуальных задач. Они успешно применялись для решения ряда прикладных задач распознавания образов (распознавание деталей на конвейере, класификация дорожных ситуаций и т.д.), медицинской диагностики (диагностика и оценка эффективности лечения артритов, векторная диагностика и расшифровка гастритов и т.д.), прогнозирования явлений (прогнозирование исхода черепно-мозговых травм и т.д.) и нейросетевого представления генетического кода [1–9, 11, 12]. Модифицированные НС Хопфилда успешно использовались для решения задач мульти-агентной маршрутизации параллельных потоков данных в глобальных телекоммуникационных и компьютерных сетях с переменной структурой [10]. Мульти-агентные нейросетевые технологии на базе гетерогенных диофантовых НС и процедур голосования позволили значительно повысить точность и робастность векторной (расширенной) диагностики гастритов [12-14].

Важное значение для эффективного распознавания образов и диагностики состояний в реальном времени представляет тот факт, что аккумулируемые в гетерогенных ПНС с самоорганизующейся архитектурой “нейрообразы” и решающие (классифицирующие и идентифицирующие) правила обеспечивают массовый параллелизм, хорошую экстраполяцию и высокое быстродействие при принятии оптимальных или субоптимальных решений.

Коллективное (мульти-агентное) использование гетерогенных ПНС в качестве нейросетевых агентов позволяет дополнительно распараллелить и распределить между локальными НС-агентами процессы решения сложных (глобальных) задач распознавания образов, анализа изображений и сцен, расширенной (векторной) диагностики состояний и адаптивной маршрутизации информационных потоков.

Список литературы

1. Тимофеев А.В. Об одном классе полиномиальных разделяющих функций в задачах опознавания и диагностики. - Методы вычислений. – Л.: Изд-во ЛГУ, 1971, вып.7, с. 106-121.
2. Пшибихов В.Х., Тимофеев А.В. Алгоритмы обучения и минимизации сложности полиномиальной опознающей системы. - Изд. АН СССР. Техническая кибернетика, 1974, № 5, с. 214-217.
3. Timofeev A.V. Intelligent Control Applied to Non-Linear Systems and Neural Networks with Adaptive Architecture. – International Journal on Intelligent Control, Neurocomputing and Fuzzy Logic, 1997, Vol.1, N1, pp. 1-18.
4. Каляев А.В., Тимофеев А.В. Методы обучения и минимизации сложности когнитивных нейро-модулей нейрокомпьютера с программируемой архитектурой. - Доклады АН, 1994, т. 237, с. 180-183.
5. Тимофеев А.В. Методы синтеза диофантовых нейросетей минимальной сложности. - Доклады АН, 1995, т.301, № 3, с.1106-1109.
6. Тимофеев А.В., Шибзухов З.М. Методы синтеза и минимизации сложности диофантовых нейронных сетей над конечным полем. – Автоматика и телемеханика, 1997, № 4, с. 204-212.

7. Тимофеев А.В. Оптимальный синтез и минимизация сложности генно-нейронных сетей по генетическим базам данных. - Нейрокомпьютеры: разработка и применение, № 5-6, 2002, с. 34-39.
8. Тимофеев А.В., Шибзухов З.М., Шеожев А.М. Синтез нейросетевых архитектур по многозначному дереву решений. - Нейрокомпьютеры: разработка и применение, № 5-6, 2002, с. 44-49.
9. Timofeev A.V. Polynomial Neural Network with Self-Organizing Architecture. – International Journal on Optical Memory and Neural Networks, 2004, N 2.
10. Timofeev A.V. Multi-Agent Information Processing and Adaptive Control in Global Telecommunication and Computer Networks. – Journal on Information Theories and Applications, 2003, vol. 10, N 1, pp.54–60.
11. Тимофеев А.В., Шеожев А.М. Методы построения обучающих выборок для развернутой медицинской диагностики на базе нейросетевых технологий. – Доклады АМАН, 2000, т. 5, № 1, с. 69 – 71.
12. Тимофеев А.В., Шибзухов З.М., Шеожев А.М. Применение диофантовых нейронных систем для генетического анализа и диагностики - Труды 6-ого Санкт-Петербургского симпозиума по теории адаптивных систем “SPAS-99”, СПб: Омега, 1999, т.2, с. 169 –171.
13. Тимофеев А.В., Шибзухов З.М., Шеожев А.М. Проектирование и обучение мульти-агентных диагностических систем. – Труды I-ой международной конференции по мехатронике и робототехнике “M&R – 2000”, СПб, 2000, т. 2, с. 342 – 345.
14. Timofeev A.V. Parallelism and Self-Organization in Polynomial Neural Networks for Image Recognition. – Pattern Recognition and Inage Analysis, 2005, vol. 15, No.1, pp. 97 – 100.

Сведения об авторе

Тимофеев Адиль Васильевич – доктор технических наук, профессор, Заслуженный деятель науки Российской Федерации. e-mail: tav@iias.spb.su

NEURONAL NETWORKS FOR MODELLING OF LARGE SOCIAL SYSTEMS. APPROACHES FOR MENTALITY, ANTICIPATING AND MULTIVALUEDNESS ACCOUNTING.

Alexander Makarenko

Abstract. *It is consider the new global models for society of neuronet type. The hierarchical structure of society and mentality of individual are considered. The way for incorporating in model anticipatory (prognostic) ability of individual is considered. Some implementations of approach for real task and further research problems are described. Multivaluedness of models and solutions is discussed. Sensory-motor systems analogy also is discussed. New problems for theory and applications of neural networks are described.*

1. Introduction

There is one principal feature of the present state of contemporary World: their evolutionary nature. That is the rate of changes that is accelerating rapidly now and the problems of evolution of global systems became more and more complicated. So, the applicability of existing theories and models of society are under question. One of the main tools for the investigation of evolution is the approach from the physical theories - that is synergetic.

There also exists the great variety of the mathematical models. It is known that the above models present mostly three types of global blocks (biospherical, climate and anthropological). The block of human (anthropogenic) factors actually seems to be the less developed one. The artificial intelligence theory can give the answers on some questions, but there is the lack of practical operational models with artificial intellect.

We may say that in spite of many successes of system analysis and mathematical modelling there is the necessity to have socio-economics models. So, main basic items for the theories and models of the World exist: the society as the whole object, the evolutionary nature of the society, the mentality problems and some propositions on the laws of their behaviour. In proposed report, we briefly consider the principles of new models construction, some applications and further scientific problems. The main goals of this report in to describe the ways of mentality accounting and especially the anticipatory property accounting consequences.

2. Short Description of Models

Let us take that society consisting of N >> 1 individuals and each individual characterising by vector of state $S_i = \{s_1^i,..,s_{k_i}^i,s_{k_i+1}^i,..,s_{M_i}^i\}, s_l^i \in M_i^l, l = 1,...,M_i$ where M_i^l is a set of possible values s_i . There are many possibilities to compose the elements in blocks and levels in such models. In sufficiently developed society individuals have many complex connections. Let us formalise this. We assume that there are connections between i and j individuals. Let J_{ij}^{pq} is the connection between p components of element i and q component of element j Thus the set Q = ({ s_i },{ J_{ij}^{pq} }, i,j = 1,...,N) characterises state of society. Analysis of recent models for media from sets of elements and bonds shows the resemblance of such society models to neural network models.

2.1 Possible Structures of Models

Now we follow the description of hierarchical systems similarly the one in papers by Mesarovich and Takahara. We suppose initially that there are M hierarchical levels in the socio-economical system with N_j elements on j-th level. Each I-th element on j-th level have description by vector of parameters Q_i^j i=1,2,...,N_j ; j=1,2,...,M. Some elements on chosen levels can be in associations, marked by set of possible indexes in associations $L_i^j \subset \{1,2,..., N_j\}$. Many elements in developed society have a vast number of interconnections on there and on upper and lower levels. We may denote connections (bonds) between i1 elements on j1 level with i2 element on j2 level by J(i1,j1;i2,j2). Remark that other fields of interest (political, social, educational and so on) have similar network representation and society, as a whole is a union of such networks.

The bonds from the connection sets may be very different on the nature. The values of bonds may represent the normalisation of economical, informational, control channels, nationality, family bonds, and participation in professional associations and so on. The general model of system as in general system theory can be introduced with the help of input X1, X2, …, XM and output Y1, Y2, …, YM spaces for every level with input variables xi ∈ Xi and output variables yk ∈ Yk.

In reality society is evolutionary system with dynamical changes on time. Further we for simplicity will consider only discrete time models with moments of time: 0,1,2,….,n,… . Following evolutionary nature of systems considered it is natural to consider as input of system in moment n the values of parameters from X in n-th time moment and as output the values at next (n+1) time moment (for n=0, 1, 2, …). Remark that in developing society the content of elements set may changes. For example in economics the list of firms and corporations changes gradually by bankruptcy and by creating of coalitions. Social, political, governmental networks are often in transformations. This lead in general to changing the number of elements N_j (n) and number of hierarchical levels M(n) for different moments of time. Next if we wish to take into account the past states of society explicitly we should introduce to equation (1) or (2) the values X(0), X(1), X(2), … , X((n-1)). Than the system description takes the form

$$Y(n)=f(X(0),X(1),X(2),...,X((N-1)), X(n), P, E).$$

2.2 Dynamics in Model

The equation above is rather general but for further investigations and practical applications we should have more developed models. Because we should consider evolutionary problems the main difficulties consist in searching the principles for modelling dynamics.

The author's models consider the Society as large complex object constructed from many elements with interconnections. The considerations of Society properties allow picking out some interesting properties and then to propose the models, which can imitate society behaviour. Surprisingly the models are familiar with models of brain activity - the neuronets [1]. Such models are under investigation by author since 1992 and yet had some interesting applications. In the processes of model consideration author continuously tried to take into account recent state of above mentioned sciences.

Now let us briefly describe the models. The first step of model building consists in the choice of model element and their description. Because it is need to take into account mentality of peoples in simplest models as the elements was took the individual with their description by series of mental and other (economical, demographic, and other parameters). These parameters may be evaluated in some scales psychology, sociology and other humanity sciences.

Next there are a lot of interconnections between elements in society - informational, business, relationship, and infrastructure. The elements are connected by bounds. The bounds correspond to influence by individual, the money flows and others. Such connections are created historically. The set of element states and bounds give the description of society in some period of time. Remark that such description is familiar with verbal description in humanity sciences. For example the pictures in L.White's works remember the pictures for global socio-economical models. But if we wish to describe the dynamics of society and should to evaluate the influence of control than we must to know or dynamical laws or tendency in dynamics. The proposed models have such dynamical principles that they can imitate the behaviour of global culture in time. This is because the models have the property of associative memory. That is it can learns from historical processes the bounds and tends to very stable constructions- to so called attractor in pattern recognition in informatics and neuroscience. It is important that many social sub-processes in society also have the properties above allow considering the separate sub-models.

In earlier papers author introduced new class of society models as modification of neuronet models such as Hopfield, Potts, Ising. It is well known that Hopfield model is derived from the functional called 'energy'. In case of hierarchical systems and symmetrical bonds between different elements and different levels there also exist functional – counterpart of 'energy'. Remark that there also may be formulated generalisation of Hebbian learning rules.

3. Mentality Accounting

The mentality accounting requires considerations internal structures and incorporating them in global hierarchical models. There are many approaches for mentality accounting (see review of some aspects in [2,3]). The most natural way for implementing this task is to consider as model for internal structure also neuronet models. Remember that originally neuronet models were introduced in the investigation of brain. Firstly we can change the basic laws. On phenomenological level it may be implemented by introducing subdivision of elements parameters on external and internal variables and establishing separate laws for two blocks of parameters. But one of the most prospective ways for mentality account lies in searching equation also in neuronet class. Here proposed to introduce the intrinsic mental models of World in elements, which represent the individuals or decision-making organisations with human participation. The simplest way consists in representing image of World in the individual's brain or in model as collection of elements and bonds between elements. In such World pattern there exist place for representing individual himself with personal beliefs, skills, knowledge, preferences. The mental structures on other individuals are also represented. Then the laws for element evolution should depend on such representation.

4. Anticipatory Property

The next step of developing models consists in accounting anticipatory aspects of individuals. It is evident that individuals in decision-making processes have prognoses on future. In such case the states of elements in model should depend on the images of the future described in internal representation. As in usual reflexive system there may exist some stages of iteration in anticipating future. We call such case as hyperincursion.

The verbal description of internal structure was described in previous section. Now we give the possible structure of models and some corollary. First we describe the model structure with one element with internal structure. If there were no internal structure it was the system in section above for dynamical law. Let the individual with internal structure has the index i=1. Their dynamic is determined by two components. First component determines by external mean field as above. Second part of dynamic is connected with internal dynamics of first individual. Remark that this dynamic partially account the willing of individual. There exist many models for such part of dynamics but it is useful to put the neuronet models for our purposes.

Let us named the pattern of society $Q^{(1)}(t)$ in section above as 'image of real world ' in discrete moment of time t. We also introduce the $Q_{wish}(t)$ - ' desirable image of world in moment t by first individual' as the set of element states and bonds wishes by first individual in moment t. $Q^{(1)}{}_{wish}(t)=(\{s_l{}^{wish}(t)\},J_{ij}{}^{wish}(t)\})$. Then we assume that the

change of first individual state depend on difference between real and desirable image of the world. The resulting system takes the form:

$$S_l(t+1)=G_l(\{s_l(t)\},\{s_l(t+1)\},\ldots,\{s_l(t+g(l))\}, R),$$

where R the set of remaining parameters. It is very prospect that the structure of system above coincides with anticipatory systems with incursion [4]. This follows possible similarity in properties.

5. Multivaluedness in Neural Networks

So far the neuronet approach had followed after the original problem formalisation. But with spreading neural network methodology some new mathematical problems had aroused which may have long- term influence on the development of neurocomputing and not only it. Such problems follow from models above. First topic concerned the neuronet models with hierarchical structure. The second and very interesting connected with possible multivaluedness in neuronet.

The main source of multivaluedness lies in neural elements with internal structure with anticipatory property when the dynamical behaviour of element depends from desired pattern of future [3]. Some preliminary results were received with R.Pushin on modelling unique multivalued neuron. Also the principles for dynamics were considered.

In parallel (and forwards) some possible range of applications may be proposed. Some such issues are brain processes and conscious, quantum mechanical analogies, many worlds concept in physics, logic and philosophy, complexity, multivalued solutions of differential equations.

6. Some Relations to Sensor-Motor Robotics

The models described in previous sections already were applied to some practical problems. Further development will follow by exploiting concepts from another research fields. Surprisingly such enrichment leads to considering some fundamental problems of cybernetics. The main tool is inter- disciplinarily methodology.

One of the sources of new ideas is the psychological investigation of visual perception. Remember that the perception process not only include the reception of signal by visual sensor system but also include internal comparison with patterns. Such patterns are internal constructs of visual perception system [5,6]. Further development of proposed models will lead to complication of internal models and to modelling of process of norms learning. It is directly connected the investigations on norms ruled behaviour. Society norms, morals, religion and so on determine the rules. Remark that till now there was a little investigation on such topics mostly of model character [7].

From another side further development of proposed models needs further re- considering of basic principles of artificial intelligence in application to such problems. From such point of view especially interesting are investigations in formally different research field in automation and control – that is from behaviour theory of mobile robot. The short list of investigations (see [8,9]) includes analysis of sensor- motor robotics; comparison of formal language's and behavioural approaches; internal representation of external environment; role of sensor information channels. Some of such concepts may be transfer to the neural type models of large socio-economical systems. One of the basic concepts in the mobile robot theory is physical landscape. In social systems case the evolution takes place in many- dimensional space constructed from physical and mental space. The points in this space are representation of system space in different time moment. Remark that the description of environment as network from [10] may be useful building internal description of world.

Conversely, the author's model may be interesting for considering mobile robot problems. The neuronet description of external environment is first example for such application. But more prospects may be the investigation on anticipatory agents. As already had formulated above, anticipating property account leads to multivaluedness of behaviour scenarios in systems with self- reference. Concerning mobile robot it may lead to more intelligent behaviour (in definition of intelligence from [11]).

The next perspective approach follows from the considering neuronal models with many agents. As background for modelling large systems it allows to solve the optimal control and game- theoretical problems. The robot soccer may be one of such issues. The second is the control problem for many vehicles with internal structure in 2D and 3D space cases.

7. Applications and Discussion

Now we should discuss some issues connected to above problems. It should be stressed some relations to another topics in artificial intellects. One of such item is so called artificial agent's theory. Now there are many investigations on artificial agents. In this approach some non-classical logic are accepted. Moreover our neuronet type models follows to consideration of some non-classical logic. Another interesting aspect is the structure of neuronet models itself. Our investigations lead to the necessity of considering set valued neural networks.

The possible applications of models with mentality account to election processes, negotiations, public relations, education are discussed. Also pure mathematical problems on multi-valued maps and on conflictly-controlled systems are posed. As application it were considered the modelling future geopolitical relations and collective security system structure in World after the destruction of the USSR, sustainable development, epidemiology, conflict theory, stock market and others [12,13]. It was created as mathematical model as computer program implementation. Remark that recently we had received interesting result on models with internal structure application to the stock market process. This is the example of mental agent application. Recently new possible fields of applications are outlined. Moreover the connected to multi–agent modelling, cellular automata, decision-making in social systems became visible. Also new analogies of quantum mechanics and social system behaviour are found. Besides the theory of distributed reflexive systems can receive the strict models for consideration. Ontology of knowledge and systems description may easy take into account mentality aspects. All this follows to new problems in neural network design and in neural network theory, which the author supposes, discuss in the report. One of the main conclusions is that the new proposed fields of neural network applications can lead to reconsidering some backgrounds of the network considerations.

The paper had been partially supported by Ukrainian Grants 0205U000622, 0105U000490

References

1. Makarenko A. About the models of Global Socio- Economic process. Proceed. of Ukrainian Acad. of Sci. (1994). N.12.p.85-87. (in Russian).
2. Makarenko A. New Neuronet Models of Global Socio- Economical Processes. In "Gaming /Simulation for Policy Development and Organisational Change" (J.Geurts, C.Joldersma, E.Roelofs eds.), Tilburg Univ. Press, (1998). pp.133-138.
3. Makarenko, A. Models with anticipatory property for large socio-economical systems. Proc. 16 th World Congress of IMACS, Lausanne, 21-25 August, (2000). Switzerland, Paper n. 422-1
4. Dubois D. Introduction to Computing Anticipatory Systems. Chaos Newsletter (Liege), January, (1998). n. 2. pp.5-6.
5. Zinchenko T. Perseption and coding. Leningrad, Leningr. Univ.Publish, (1981). 183 p. (in Russian).
6. Perception. Mechanisms and Models. W.H.Freeman and Company, San Francisco, (1972).
7. Epstein J.M. Learning to be thoughtless: social norms and individual computation. Santa Fe Institute, CA, USA, Working Paper n.6. January, (2000). 31 pp.
8. Tani Jun Model- based learning for mobile robot navigation from the dynamical system perspective. IEEE Trans. Syst.Man.Cyb., Part.B. (1996). V.26, n. 3. Pp.421-436.
9. Lect.Notes. in Artificial Intelligence, Springer- Verlag, (1998). N.1515. 260 pp.
10. Makarenko A. Complexity of individual object without including probability concept. Fourth Int.Conf. on Computing Anticipatory Syst., Abstract Book. Liege, Belgium, August, (2000). 3pp.
11. Albus J.S. Outline for a theory of intelligence. IEEE Trans. On Syst. Man. Cyb. , (1991), v.21. n.3. pp. 473-504.
12. Levkov S., Makarenko A. Geopolitical relations in post USSR Europe as a subject of mathematical modelling and control. Proceed. 7 IFAC/IFORS/IMACS Symposium: Large scale Systems. L.UK: Vol.2. (1998). p.983-987.
13. Levkov S., Makarenko A., Zelinsky V. Neuronet type models for stock market trading patterns. Proc. 5th Ukrainian Conf. AUTOMATICA'98. Kiev, May, 1998. Pp.162-166.

Author's Information

Alexander Makarenko – National Technical University of Ukraine "KPI", Institute for Applied System Analysis Department of Mathematical Methods of System Analysis
37, Pobedy Avenue, 03056, Kiev-56, Ukraine, makalex@i.com.ua, makalex@mmsa.ntu-kpi.kiev.ua

6.2. Neural Network Models

ПРЕДСТАВЛЕНИЕ НЕЙРОННЫХ СЕТЕЙ ДИНАМИЧЕСКИМИ СИСТЕМАМИ

Владимир С. Донченко, Денис П. Сербаев

***Abstract**: Рассматривается представление нейронных сетей в виде динамических систем. Предложен метод обучения нейронных сетей с помощью теории оптимального управления.*

***Keywords**: нейронные сети, динамические системы, обучение.*

Введение

В настоящее время нейронные сети получили самое широкое распространение и успешно применяются для решения различных сложных задач таких как, например, управление и идентификация нелинейными системами, анализ финансового рынка, моделирование сигналов и т.д. Качество работы нейронной сети во многом зависит от эффективности выбранного алгоритма определения весов сети для достижения требуемой точности на обучающей и тестовой выборках. Ниже предложен метод нахождения весов нейронной сети на основе теории оптимального управления и представления нейронной сети в виде динамической системы.

Представление сети системой рекуррентных соотношений

Как известно [4], нейронная сеть – это совокупность однотипных элементов - нейронов, – разбитых на части-слои, определенным образом последовательно связанные между собой. Каждый из нейронов, из которых складывается каждый из слоев, собственно, отвечает скалярной функции векторного аргумента $y=F(w^Tx)$, что является суперпозицией линейной формы с вектором линейной формы w, который называют вектором весом, – и скалярной функции F. Последнюю называют функцией активации нейрона. Аргумент x – векторный – вход нейрона, скалярное значение y – выход. Входы нейрона, связанные с их принадлежностью тому или иному слою, на которые разбита сеть. Эти слои упорядочены последовательно так, что выходы всех нейронов предшествующего слоя подаются на входы любого из нейронов следующего слоя. Входом первого слоя является сигнал, который является входным для всей сети. Для стандартизации обозначений, будем считать, что вход образовывает слой с номером 0. Этот слой, в отличие от других, не содержит нейронов и, собственно, задает входной сигнал $x(0)=x_0$, который состоит из l_0 компонент. Слой с номером N есть выходным. Каждый из слоев с соответствующими номерами k: k=0,…,N имеет количество нейронов l_k. Скалярные выходы нейронов одного слоя объединяются в один вектор x(k), k=0,…,N, который будем считать выходом соответствующего слоя. Размерность такого вектора совпадает с количеством l_k, k=0,…,N нейронов в соответствующем слое.

Будем считать, что все нейроны одного и того же слоя имеют одинаковые веса. Общий для всех нейронов одного слоя вес будем обозначать соответственно номера слоя w(k): $w(k)=(w(k)_1,\dots,w(k)_{l_{k-1}})^T$, k=1,…,N... Размерность вектора весов, естественно, совпадает с количеством l_{k-1} нейронов слоя-предшественника.

Что же касается функций активации любого из нейронов соответствующего слоя, будем считать, что они являются разными для каждого нейрона и будут обозначаться $F_i^{(k)}(z)$, и=1,…,l_k, k=1,…,N... Напомним, что функции активации являются скалярными функциями скалярных аргументов.

Таким образом, преобразование входного сигнала $x(0)=x_0$ последовательными слоями нейронной сети описывается системой рекуррентных соотношений:

$$x(k+1)=\begin{pmatrix} F_1^{(k+1)}(w(k+1)^T x(k)) \\ \dots \\ F_{l_{k+1}}^{(k+1)}(w(k+1)^T x(k)) \end{pmatrix}, \quad k=0,\dots,N-1... \tag{1}$$

Обозначив через g(z,k+1) векторную функцию $(F_1^{(k+1)}(z),...,F_{l_k}^{(k+1)}(z))$, перепишем (1) в более компактном виде:

$$x(k+1)=g(w(k+1)^T x(k), k+1), k=0,\ldots,N-1. \quad (2)$$

С учетом того, что каждый слой сети осуществляет отображение из одного линейного пространства в другое в соответствии с (1) или (2), схема нейронной сети может быть представлена рисунком 1.

На рисунке 1 слои представлены прямоугольниками, которые осуществляют преобразование в соответствии с (1) или (2). В каждом таком

прямоугольнике выделенные части, которые имеют естественную интерпретацию: w(k) отвечает входным синапсам, части с функциями активации $F_1^k,...,F_{l_k}^k$ отвечают нейронам, часть с x(k) отвечает за концентрацию выходов нейронов слоя в единый выход всего слоя в целом.

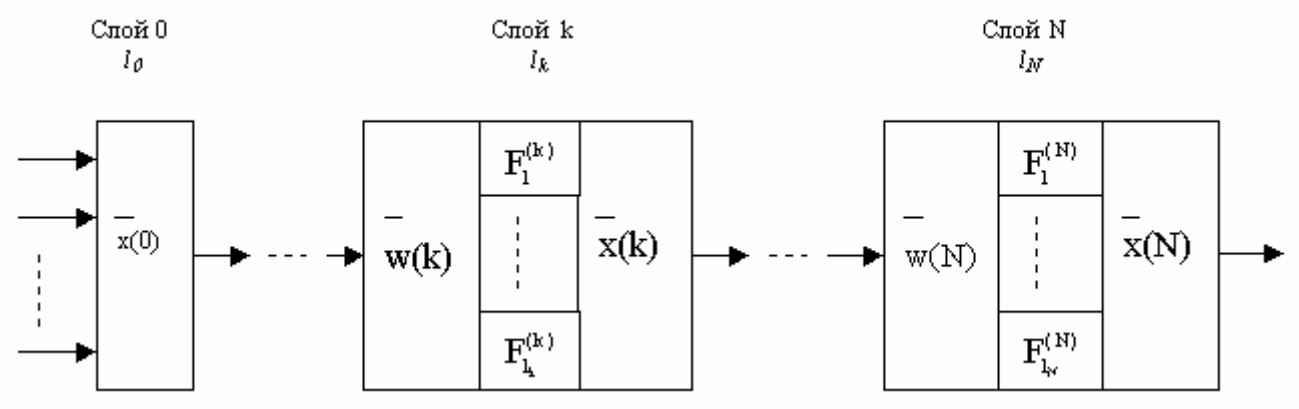


Рисунок 1.

Задача обучения нейронной сети

Задача обучения нейронной сети состоит в том, чтобы подобрать веса w(k), k=1,...,N слоев сети так, чтобы на заданной учебной выборке: последовательности пар $(x_0^{(1)},y^{(1)}),...,(x_0^{(M)},y^{(M)}), x_0^{(i)}\in R^{l_0}, y^{(i)}\in R^{l_N}$, $i=\overline{1,N}$, в которых первая компонента интерпретируется как один из вариантов входа сети и имеет размерность входного слоя l_0, а вторая – как желательный выход сети и имеет размерность выходного слоя l, – достигалось наименьшее отклонение выходных сигналов сети от желательных. Таким образом обучение нейронной сети состоит в том, чтобы минимизировать функционал J(w(1),...,w(N)),который определяется соотношением:

$$J(w(1),\ldots,w(N))=\sum_{i=1}^{M}||y^{(i)}-x^{(i)}(N)||^2, \quad (3)$$

где $x^{(и)}$ (N), $i=\overline{1,N}$ – выход сети на i-том для i-того элемента учебной выборки: объединенный выход нейронов последнего слоя нейронной сети, когда на вход подается соответствующее входное значение элемента учебной выборки.

Отметим, что если учебная выборка состоит из одного элемента, то есть если на вход подается сигнал, а сеть должна обучиться на выходной сигнал, функционал качества обучения приобретает вид:

$$J(w(1),\ldots, w(N))=||y-x(N)||^2. \quad (4)$$

Стандартный подход к решению задачи обучения нейронной сети

Стандартным подходом к обучению нейронной сети есть такой, согласно которому по фиксированному входу из возможных вариантов входа учебной выборки и первоначально фиксированных на каком-то уровне весов слоев происходит последовательное изменение этих весов каждого из слоев в

направлении, противоположном градиенту функционала J(w(1),…,w(N)) по весу соответствующего слоя w(k), k=1,…,N... Коэффициенты, которые определяют длину шага в соответствующем направлении, берутся сравнительно небольшими.

Решение задачи обучения нейронной сети применением результатов для обобщенной системы управления

Принципиальными для решения задачи обучения есть утверждение теорем 1, 2 ниже о представлении задачи обучения сети обобщенной системой управления для соответственно: одной траектории и пучка таких траекторий.

Теорема 1. Задача обучения нейронной сети на один входной сигнал представляет собою оптимизационную задачу для обобщенной системы управления, в которой соответственно:

- фазовые переменные x(k), k=0,...,N является выходами слоев с соответствующими номерами;
- управление u(k) с соответствующим номером k=0,...,N-1 совпадает с весом k+1 слоя и определяется соотношением: u(k) = w(k+1), k=0,…,N–1;
- функции f, которые описывают рекуррентную связь между значениями фазовой переменной, определяются функциями g выходов слоев по соотношениям:

$$f(x(k), u(k), k)=g(w(k+1)^{T}x(k),k+1),\ k=0,\ldots,N-1 \quad (5)$$

- функционал $I(u_0,\ldots,u_{N-1})$ совпадает с J(w(1),…,w(N)) с (2).

Доказательство теоремы приведено в [2].

Следствием теоремы 1 является возможность использовать теорему о виде градиентов для обобщенной системы управления для подсчета градиентов функционала качества обучения в задачи обучения нейронной сети. [2]

Теорема 2. Задача обучения нейронной сети на учебную выборку произвольного объема M представляет собою оптимизационную задачу для пучка траекторий обобщенной системы управления, в которой соответственно:

- фазовые переменные X(k), k=0,…,N являются матричными и состоят из столбцов $x^{(и)}(k)$,i=$\overline{1,M}$, каждый из которых есть выход слоя с соответствующими номером, если на вход подается соответствующий входной элемент $x_0^{(и)}$,i=$\overline{1,M}$ из учебной выборки;
- управление u(k) с соответствующим номером k=0,…,N–1 совпадает с весом k+1 слоя и определяется соотношением: u(k) = w(k+1), k=0,…,N–1;
- функции F= F(X(k), u(k), k), которые описывают рекуррентную связь между значениями фазовой переменной, являются матричными и состоят из столбцов $(f(X(k), u(k), k))_{и}$, i=$\overline{1,M}$, которые определяются функциями g выходов соответствующих слоев нейронной сети в соответствии с соотношениями:

$$(f(X(k), u(k), k))_{и}=g(w(k+1)^{T}x^{(и)}(k),k+1),\ k=0,\ldots,N-1, \quad (6)$$

  где $x^{(и)}(k)$,i=$\overline{1, M}$ – выход слоя с номером k: k=$\overline{1, M}$, как реакция на i-тый элемент учебной выборки;
- функционал $I(u_0,\ldots,u_{N-1})$ совпадает с J(w(1),…,w(N)).

Доказательство. Доказательство проводится так же, как и для предшествующего результата и приведено [2].

Следствием теоремы 2 является возможность использовать теорему о виде градиентов для обобщенной системы управления с пучком траекторий, для подсчета градиентов функционала качества обучения в задаче обучения нейронной сети. [2] Это, собственно, результат следующей теоремы.

Теорема 3. Градиенты функционала качества обучения нейронной сети определяются соотношениями:

$$grad_{w(k)} J(w(1),...,w(N)) = -grad_{w(k)} \sum_{i=1}^{M} H^{(i)}(x^{(i)}(k), w(k+1), p^{(i)}(k+1), k) \quad k=1,...,N. \qquad (7)$$

Теорема 3. служит основой алгоритма Error Back Propagation для обучения нейросетей.

Выводы

В статье описан метод нахождения весов нейронной сети на основе теории оптимального управления и представления нейронной сети в виде динамической системы. Использование представления нейронной сети в виде динамической системы позволяет эффективно определять веса нейронной сети и таким образом решать задачу обучения.

Литература

1. Кириченко Н. Ф., Крак Ю. В., Полищук А.А. Псевдообратные и проекционные матрицы в задачах синтеза функциональных преобразователей.// Кибернетика и системный анализ –2004.–№3.
2. Кириченко Н.Ф., Донченко В.С., Сербаев Д.П. Нелинейные рекурсивные регрессионные преобразователи: динамические системы и оптимизация. // Кибернетика и системный анализ – 2005.–№1.
3. Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals, and Systems, 2, 303–314.
4. Haykin, S. (1999). Neural networks: A comprehensive foundation (2nd ed.). Englewood Cliffs, NJ: Prentice Hall.
5. Kohonen, T. (1995). Self-organizing map. Heidelberg, Germany: Springer-Verlag.
6. Бублик Б.Н., Кириченко Н.Ф. Основы теории управления. – К.: Высшая школа, 1975.–328 с.

Информация об авторах

Владимир С. Донченко – профессор, Киевский национальный университет имени Тараса Шевченко, кафедра Системного анализа и теории оптимальных решений, e_mail: vsdon@unicyb.kiev.ua

Денис П. Сербаев – аспирант, Киевский национальный университет имени Тараса Шевченко, кафедра Системного анализа и теории оптимальных решений, e_mail: sdp@unicyb.kiev.ua

GENERALIZATION BY COMPUTATION THROUGH MEMORY

Petro Gopych

***Abstract**: Usually, generalization is considered as a function of learning from a set of examples. In present work on the basis of recent neural network assembly memory model (NNAMM) a biologically plausible 'grandmother' model for vision has been proposed within which each separate memory unit itself can generalize. For such a generalization by computation through memory analytical formulae and numerical procedure are found to calculate exactly the perfectly learned memory unit's generalization ability. The model's memory has complex hierarchical structure and can be learned by one-step process from one example. A simple binary neural network for bell-shaped tuning is described.*

***Keywords**: generalization, 'grandmother' model for vision, neural network assembly memory model, one-step learning, learning from one example, neuron receptive field, bell-shaped tuning.*

1. Introduction

We know from our everyday experience that even under difficult observation conditions the recognition of complex visual objects occurs in practice immediately, in an on-line regime. The ability to recognize visual objects regardless of the side of view, their illumination, occlusion or particular distortion is called generalization ability; up to present its brain mechanisms remain unclear.

In real life any two successive images cannot coincide literally, point-by-point, although they correspond to the same object. To overcome this difficult computational problem it is supposed that it is enough to remember labels of only some typical images (examples) and to learn the common memory/generalization system to predict to a huge amount of unknown, do not storing in memory, images. Such a statement of the problem implies that particular image can continuously be transformed, possibly not too sharp, into any other image of the same object through an infinite continuous series of intermediate images.

The learning theory gives a definition of generalization and rules to ensure it. So, generalization provides the best possible functional relationship between an input image, x, and its label, y, by learning from a set of examples, x_i, y_i. This problem is similar to the problem of fitting a continuous smooth function of some arguments to measurement data x_i, y_i or, in other words, the ability of estimating correctly the values of this function in points where data are not available. For this purpose standard interpolating methods are usually used [1].

Above approach is not the only possible. It is naturally to assume that the real world is actually represented in human visual system as a series of 'frames,' discrete and only perceived continuously, as in a movie. If so then the amount of information needed to be maintained reduces crucially and for this reason memory system, subserving vision and dealing with a finite set of discrete images, may become simpler. This work follows such an alternative approach.

2. Generalization by Interpolating from Examples

Within the classic learning theory generalization by interpolating among examples supports a popular neural network (NN) architecture which combines the activity of some hidden broadly tuned 'units' (local NN circuits) learned to respond to one of presented training examples and to a variety of other images but at sub-maximal firing rates. This idea is consistent with the fact that bell-shaped tuning is common among neurons in visual cortex and that in infero-temporal cortex, IT, there exist neurons tuned to different complex objects or their parts.

Mathematically, using the method of regularization, the learning from examples may be formulated as measurement data approximation by a smooth function $f(x) = \sum w_i k(x,x_i)$ which minimizes the error of training; it is a weighted sum (weights w_i) of basis functions $k(x,x_i)$ depending on a new (unknown) image x. Function $k(x,x_i)$ may be, for example, a radial Gaussian centered on x_i, representing the ith neuron's receptive field and responding optimally to (memorizing) x_i. The width of $k(x,x_i)$ defines also the unit's selectivity as a memory device: for broadly tuned k its selectivity is poor but a linear combination of such functions provides good generalization ability, for sharply tuned k (e.g., a delta function or very narrow Gaussian) its selectivity is perfect but such a k cannot be used for generalization. In $f(x)$ functions $k(x,x_i)$ may be learned from their inputs, x_i, in a passive (without the feedback) regime while weights, w_i, depend also on outputs, y_i, and demand more complicate iterative learning from examples, x_i, y_i. That is learning splits into two parts: leaning the basis functions (memory units and simultaneously neuron receptive fields) and learning the weights of the whole network (learning to generalize using already learned memory units). The algorithm described can implement a feed forward NN with one hidden layer containing as many units as training examples; parameters w_i are interpreted as synaptic weights between corresponding units and the output, $f(x)$; for further references see [1].

3. A 'Grandmother' Model for Vision

In the classic 'grandmother' theory for vision, an image recognition happens when the combination of all its features precisely coincides with such a combination associated to particular grandmother neuron, i.e. in this case between the input image and different memory records a direct literally comparison is needed. The lack of generalization is the basic problem of such a model. To solve it the model was essentially extended: it is supposed that 'generalization emerges from linear combinations of neurons tuned to an optimal stimulus' [1] (see also Section 2). We propose another extension solving the generalization problem under assumption that each memory unit itself can generalize.

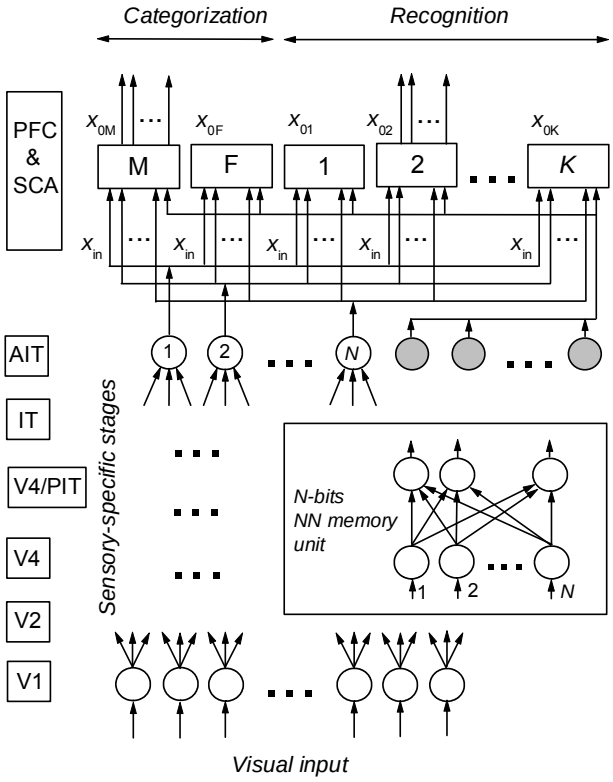


Figure 1. An oversimplified scheme of a 'grandmother' model for vision based on the NNAMM. At the bottom, in V1, cells have small receptive fields and respond preferably to oriented bars, along the ventral visual stream they increase gradually their receptive fields and complexity of their preferable images and at the top, in AIT, neurons respond optimally already to rather complex objects. AIT neurons 1,…,*N* (open circles) code the image of current interest, e.g. a face, as a binary (±1) vector x_{in}; other similar neurons (filled circles) can code (respond optimally to) other complex objects. Boxes M and F correspond to assembly memory units, AMUs (Figure 2), storing reference codes of the 'ideal' male, x_{0M}, and female, x_{0F}, faces; boxes 1,…,*K* denote AMUs storing the codes $x_{01},\ldots,x_{0K}$ which represent known (previously encountered) faces 1,…,*K*, regardless of their categorization. The case is presented where a current face code x_{in} extracted from current visual input is recognized as the face number 2 and categorized as a male face (x_{in} initiates the correct retrieval of memory traces x_{0M} and x_{02} signified as output arrows from boxes M and 2, respectively). In the insertion a feed forward NN (box 2 in Figure 2) related to particular AMU_i and storing the code x_{0i} is shown (AIT neurons 1,…,*N* may be exit-layer neurons of such an NN). If x_{in} is absent among the codes $x_{01},\ldots,x_{0K}$ but recognized as x_{0M} or x_{0F} then it can be remembered in the (*K* + 1)th empty AMU, AMU_{K+1}, which is not shown. V1, primary visual cortex; V2 and V4, extra striate visual areas; IT, infero-temporal cortex; AIT, anterior IT; PIT, posterior IT; PFC, prefrontal cortex; SCA, sub-cortical areas (e.g., hippocampus, see Section 4.2).

As Figure 1 demonstrates, in our model all sensory-specific stages of input visual data processing completely coincide with those that Poggio & Bizzi [1] discussed and, consequently, in this part both models are biologically equally plausible. In particular, in AIT neurons (open circles) tuned to respond to complex images are used although our local circuits employed for tuning are quite different (Section 5). The main distinction between our Figure 1 and Figure 2 in [1] consists in the structure of their sensory-independent parts: in Figure 1 it is implemented on the basis of the neural network assembly memory model, NNAMM, discussed in Section 4 [2].

Visual memory is constructed as a set of the NNAMM's assembly memory units, AMUs (Figure 2 in Section 4.2), interconnected between each other and storing one memory trace per one AMU. Memory traces are *N*-dimensional binary (±1) vectors represented particular images (e.g., known faces, $x_{01},\ldots,x_{0K}$) or categories of such images (e.g., the 'ideal' male, x_{0M}, and female, x_{0F}, faces). Tuned neurons 1,…,*N* (open circles) convey the code x_{in}, extracted from current visual input at sensory-specific stages of data processing and representing a current face, to all AMUs. Similar codes of other images presented in current visual input are also extracted and other tuned neurons (filled circles) convey them to all AMUs. But by means of a spatio-temporal synchrony mechanism, the AMUs shown select only the code of their interest, x_{in}; other similar codes may be the codes of interest for other AMUs which are not shown.

With the probability defined by Equations 5 and/or 6, for example, AMU_2 can perfectly recognize current x_{in} as x_{02} (can interpret x_{in} as a damaged x_{02}) even in the case $x_{in} \neq x_{02}$ and thanks to this fact the ability to generalize occurs (i.e. in contrast to Section 2, an AMU per se provides as perfect memory selectivity as well a generalization by computation through memory). Because particular AMU contains a 'grandmother' neuron (see also Section 4), we can consider our model for vision as a 'grandmother' one.

4. The NNAMM as a Memory Model Used

P.M.Gopych has proposed [3] a ternary/binary 0,±1/±1 data coding and demonstrated [3] that corresponding NN decoding algorithm (inspired by J.J.Hopfield [4]) is simultaneously the retrieval mechanism for an NN memory. As NNs used for data decoding and memory storing/retrieval are the same (see insertion in Figure 1), they have also common data-decoding/memory-retrieval performance (Section 4.3). Later this data coding/decoding approach was developed into the binary signal detection theory (BSDT) [5] and neural network assembly memory model (NNAMM) [2] closely interrelated in their roots and providing the best quality performance. The price paid for the NNAMM optimality is the fact that it places each memory trace in its own AMU (an estimation of human memory capacity, though it is possibly too optimistic — 10^{8432} bits [6], supports this assumption).

4.1 Formal Background

Let us denote a vector with components x^i (i = 1,…,N), whose magnitudes are ±1, as x. It can carry N bits of information and its dimension N is the size of a local receptive field for the NN/convolutional feature discrimination algorithm [7] or the size of an NN memory unit discussed below. If x represents information stored or that should be stored in the NN then we term it reference vector x_0. If the signs of all components of x are randomly chosen with uniform probability, ½, then that is random vector x_r or binary noise. We define also a damaged reference vector $x(d)$ with damage degree of x_0 d and components

$$x_i(d) = \begin{cases} x_0^i, if\, u_i = 0, \\ x_r^i, if\ u_i = 1 \end{cases} \qquad d = \sum u_i / N, \quad i = 1,..., N. \tag{1}$$

Marks u_i take magnitudes 0 or 1 which may randomly be chosen with uniform probability, ½. If the number of marks u_i = 1 is m then the fraction of noise components of $x(d)$ is $d = m/N$; $0 \le d \le 1$, $x(0) = x_0$, and $x(1) = x_r$. The fraction of intact components of x_0 in $x(d)$, $q = 1 - d$, is *intensity of cue* or cue index; $0 \le q \le 1$, $q + d = 1$, d and q are discrete. For a given $d = m/N$ the number of different vectors $x(d)$ is $2^m C^N_m$, $C^N_m = N!/(N - m)!/m!$; for d ranged $0 \le d \le 1$, complete finite set of all vectors $x(d)$ consists of $\sum 2^m C^N_m = 3^N$ elements (m = 0,1,…,N).

For decoding the data coded as described, we use a two-layer NN with N McCalloch-Pitts model neurons in its entrance and exit layers; these neurons are linked 'all-inputs-to-all-outputs' as the insertion in Figure 1 demonstrates. For learned NN, synapse matrix elements w_{ij} are

$$w_{ij} = \xi x_0^i x_0^j \tag{2}$$

where $\xi > 0$ (below ξ = 1), x^i_0 and x^j_0 are the ith and the jth components of x_0, respectively. Hence, vector x_0 and Equation 2 define the matrix w unambiguously. We refer to w as a perfectly learned NN and stress the crucial importance of the fact that it remembers only *one* pattern x_0 (the available possibility of storing other memories in the same NN was intentionally disregarded). It is also assumed that the NN's input vector x_{in} is decoded (reference or state vector x_0 is extracted) successfully if learned NN transforms an x_{in} into the output vector $x_{out} = x_0$ (an additional *'grandmother'* neuron mentioned in Section 3 checks this fact).

The transformation algorithm is the following. For the jth exit-layer neuron, its input signal h_j is

$$h_j = \sum w_{ij} v_i, \quad i = 1,..., N \tag{3}$$

where v_i is an output signal of the ith entrance-layer neuron. The jth exit-layer neuron's output, x^j_{out}, is calculated by a rectangular response function with the neuron's triggering threshold $\theta \ge 0$:

$$x_{out}^j = \begin{cases} +1, & if \quad h_j > \theta \\ -1, & if \quad h_j \le \theta \end{cases} \tag{4}$$

where for $h_j = \theta$ the value $v_j = -1$ was arbitrary assigned.

Since entrance-layer neurons of the NN used play only the role of input fan-outs which convey their inputs to all exit-layer neurons, in Equation 4 $v_i = x^i_{in}$. Of this fact and Equations 3 and 4 for the jth exit layer neuron we have: $h_j = \sum w_{ij} x^i_{in} = x^j_0 \sum x^i_0 x^i_{in} = x^j_0 Q$ where $Q = \sum x^i_0 x^i_{in}$ is a convolution of x_0 and x_{in}. The substitution of $h_j = x^j_0 Q$ into Equation 4 gives that $x_{out} = x_0$ and an input vector x_{in} is decoded (reference vector x_0 is extracted) successfully if $Q > \theta$. Since for each x_{in} exists such a vector $x(d)$ that $x_{in} = x(d)$, inequality $Q > \theta$ can also be written as a function

of $d = m/N$: $Q(d) = \sum x^i_0 x_i(d) > \theta$ ($i = 1,2,\ldots,N$) where θ is the threshold of Q and, simultaneously, the neuron's triggering threshold. Hence, for perfectly learned intact NNs above NN and convolutional decoding algorithms are equivalent.

Since $D = (N - Q)/2$ where D is Hamming distance between x_0 and specific $x(d)$, the inequality $D < (N - \theta)/2$ is also valid and NN, convolutional, and Hamming distance decoding algorithms mentioned are equivalent. As Hamming decoding algorithm is the best (optimal) in the sense of statistical pattern recognition quality (that is no other algorithm can outperform it), NN and convolutional algorithms described are also optimal (the best) in that sense. Moreover, similar decoding algorithms based on locally damaged NNs may also be optimal [2,8] (see also Table 1 in Section 6).

4.2 The AMU Architecture

We saw that a two-layer NN (as in the insertion in Figure 1) can be used for optimal one-trace memory storing/retrieval although randomly chosen $x_{in} = x(d)$ initiates successful retrieval only randomly. Thus, to implement the model's possibilities completely, the retrieval should be initiated by a *series* of different vectors x_{in} and it will be successful if one of the next x_{in} leads suddenly to $x_{out} = x_0$ emergence. Figure 2 exhibits the minimal architecture needed to provide optimal memory trace retrieval from the learned NN (box 2). As retrieval is initiated by $2^m C^N_m$ different vectors $x(d)$ (they are labels of images or 'frames' from their finite set mentioned in Section 1), it gives also optimal generalization by computation through memory. The internal loop 1-2-3-4-1 ensures the generation of different (e.g., random) vectors $x_{in} = x(d)$ with a given value of d while the external loop 1-2-3-4-5-6-1 maintains the internal one.

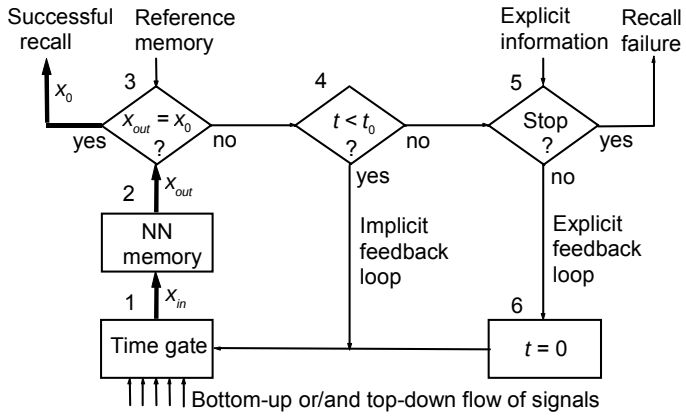


Figure 2. The flow chart (the architecture) of an assembly memory unit, AMU, and its short-distance environment adapted from [2]. The structure of the NN memory unit (box 2) specifies the insertion in Figure 1. Pathways and connections are shown in thick and thin arrows, respectively.

Within the NNAMM, the whole memory is a very large set of interconnected AMUs of rather small capacity (N ~ 100 or less) organized hierarchically. An AMU (Figure 2) consists of boxes 1,2 and 6, diamonds 3,4 and 5, and their internal and external pathways and connections designed for propagation of synchronized groups of signals [vectors $x(d)$] and asynchronous control information, respectively; it implements the BSDT decoding algorithm for solving the problem of optimal memory storing/retrieval directly.

Box 1 (a kind of N-channel time gate) transforms initial ternary (0,±1) sparsely coded very-high-dimensional vectors into binary (±1) densely coded rather low-dimensional ones. Here from the flood of asynchronous input spikes, synchronized pattern of signals in the form of N-dimensional feature vector $x_{in} = x(d)$ is prepared by a dynamical spatiotemporal synchrony mechanism. Box 2 is an NN learned according to Equation 2 (or Equation 7 from Section 4.4) where each input, x_{in}, is transformed into its corresponding output, x_{out}. Diamond 3 performs the comparison of x_{out} just now emerged with reference vector (trace) x_0 from *reference memory* (RM, see below). If $x_{out} = x_0$ then the retrieval is successful and it is finished. In the opposite case, if current time of retrieval t is less than its maximal value t_0 (this fact is checked in diamond 4) then the loop 1-2-3-4-1 is activated, retrieval starts again from box 1, and so forth. If t_0, a parameter of time dependent neurons, was found as insufficient to retrieve x_0 then diamond 5 examines whether an external reason exists to continue retrieval. If it is then the loop 1-2-3-4-5-6-1 is activated, the count of time begins anew (box 6), and internal cycle 1-2-3-4-1 is repeated again with a given frequency f, or time period $1/f$, while $t < t_0$.

The trace x_0 is held simultaneously in a particular NN memory (box 2) and in its auxiliary RM that may be interpreted as a *tag* of corresponding NN memory or as a *card* in a long-term memory catalog and performs two

interconnected functions: verification of current memory retrieval results (diamond 3) and validation of the fact that a particular memory record actually exists in the long-term memory store. Thus, specific RM is a part of memory about memory or *'metamemory'*. In contrast to the NN memory which is a kind of computer register and is conventionally associated with real biological networks, particular RM is a kind of slot devoted to the comparison of a current vector x_{out} with the reference pattern x_0 and may be associated with a coincidence integrate-and-fire *'grandmother'* neuron (cf. Section 3).

All elements of the internal feedback (reentry) loop 1-2-3-4-1 run routinely in an automatic regime and for this reason they may be interpreted as respected to *implicit* (unconscious) memory. That means that under the NNAMM all operations at synaptic and NN memory levels are unconscious. External feedback (reentry) loop 1-2-3-4-5-6-1 is activated in an unpredictable manner because it relies on external (environmental and, consequently, unpredictable) information and in this way provides unlimited diversity of possible memory retrieval modes. For this reason an AMU can be viewed as a particular *explicit* (conscious) memory unit. An external information in diamond 5 used can be thought of as an explicit or conscious one.

Recent evidences demonstrate that learning induces molecular changes in neocortex and hippocampus and this finding, along with based on it physiological theory assuming that long-term memory is stored in parallel in the neocortex and hippocampus [9], supports the NNAMM's idea of storing each memory record simultaneously in an NN (a counterpart to a neocortex network) and in a 'grandmother' neuron (probably, a cell in hippocampal structures). For other arguments in favor of the NNAMM's biological plausibility see ref. [2].

4.3 The AMU Basic Performance

For the best data-decoding/memory-retrieval algorithms considered their quality performance function is $P(d,\theta)$, the probability of correct decoding/retrieval conditioned under the presence or absence of x_0 in the data analyzed against d (or q) and θ (all notations are as in Section 4.1).

The finiteness of the set of vectors $x(d)$ makes possible to find $P(d,\theta)$ by multiple computations [3]:

$$P(d,\theta) = n(d,\theta)/n(d) \tag{5}$$

where $n(d)$ is a given number of different inputs $x_{in} = x(d)$ with a given value of d; $n(d,\theta)$ is a number of $x(d)$ from $n(d)$ leading to the NN's response $x_{out} = x_0$ if the NN decoding/retrieval algorithm with triggering threshold θ is applied. For small N, $P(d,\theta)$ can be calculated exactly because $n(d) = 2^m C^N_m$, complete set of $x(d)$, is small and all possible inputs can be taken into account. For large N, $P(d,\theta)$ can be estimated by multiple computations approximately but, using a sufficiently large set $n(d)$ of randomly chosen inputs $x(d)$, with any given accuracy.

For intact perfectly learned NNs, convolutional (Hamming) version of the BSDT/NNAMM formalism allows to derive analytical expression for $P(d,\theta)$ [8]:

$$P(d,\theta) = \sum_{k=0}^{k\max} C_k^m / 2^m, \quad k\max_0 = \begin{cases} (N-\theta-1)/2, & if \quad N \quad is \quad odd \\ (N-\theta)/2-1, & if \quad N \quad is \quad even. \end{cases} \tag{6}$$

Here if $k\max \le m$ then $k\max = m$ else $k\max = k\max_0$ and C^m_k denotes binomial coefficient.

Since θ, F (false-alarm probability), Q, and D, d and q are related, $P(d,\theta)$ can, for example, be written as ROC curves, $P_q(F)$, or as basic memory performance functions, $P_F(q)$ [2].

4.4 The AMU Learning

Equation 2 defines perfect one-step learning from one example as for the NN considered its input and its output (the label, 'teacher,' or 'supervisor') are exactly known, x_0. But it is often necessary to have unsupervised learning.

Let us use the traditional delta learning rule in the form

$$w_{ij}^{(n+1)} = w_{ij}^{(n)} + \eta\, v_j^{(n)} h_i^{(n)} \tag{7}$$

where n and $\eta > 0$ are an iteration number and a learning parameter, respectively; $v_j = x^i_{in}$; $h_i = \sum w_{ik} v_k$, $k = 1,\ldots,N$; the training set consists of only one sample, $x_{in} = x_0$ (such an iteration process does not feedback the NN's output x_{out} to the NN's input, it estimates w_{ij} using the previous value of w_{ij} and the values of η and components of x_0).

If η is small ($\eta < 1$) then the learning rate achieved is low and asymptotic values of w_{ij} are not reached. This case has no essential practical significance. If η is large ($\eta > 100$) then the iteration process leads to a fast, one-trial, without the 'catastrophic forgetting' learning because already the first iteration gives the result which is close to the asymptote and next iterations do not lead to the essential advance.

Let us consider the NN with $N = 40$, continuous w_{ij}, v_j, h_i, x^i_{in}, x^i_{out} and all initial values of w_{ij} chosen randomly with uniform probability from the range [–1,1]. If the initial learning pattern is $x_{in} = x_0$ then after each next iteration an NN with the next version of its weight matrix w_{ij} provides the emergence of the next version of x_{out} (the next approximation of x_0). For $\eta = 400$ already the first iteration gives the approximation's quality estimation $\sum |x^i_{out} - x^i_0| < 10^{-30}$ ($i = 1,\ldots,N$). The NN's specific RM related to the same AMU should also be learned simultaneously.

5 Neuron RFs and NNs for the Tuning

Figure 3 illustrates visual data processing using the NN of Section 4.1 together with its 'grandmother' neuron checking whether $x_{out} = x_0$. Binarization of y gives x_{in} (e.g., if $y_i > bd_i$ then $x^i_{in} = 1$ else $x^i_{in} = -1$) with no loss of information important for the following feature discrimination [7]. Binarization of components of y or h means spike generation; h may be interpreted as a simplified 1D profile of a 'grandmother' neuron's receptive field (RF), it results in an internal weighted network process (Equation 3). Such RFs can be typical for on-cells (panels *a*, *c*, *d*) or off-cells (panel *b*) and, as *a*) and *b*) demonstrate, noise x_{in} can initiate the reverse of RF polarity (these predictions are consistent with current physiological results [10]). The set of outputs of 'grandmothers' of different NNs (the top raw in Figure 3) reduces the redundancy of initial data and can constitute x_{in} for NNs at the next level of data processing hierarchy and etc; in particular, in AIT x_{in} could already represent a face (Section 3).

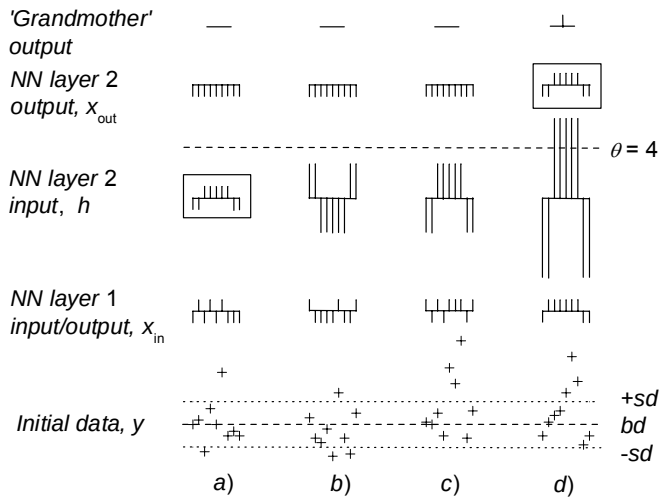


Figure 3. Computer simulated samples of initial visual data (y, an electric output of light-sensitive retina cells) and their processing results (x_{in}, h, x_{out}) in four N-channel windows: *a*) and *b*), y is a background, bd = 100, damaged by Poisson-like noise; *c*) and *d*), y is a Gaussian peak (a = 20, $fwhm$ = 5) on bd damaged by noise (crosses, values of y in each channel). Vectors y, x_0 (boxed), x_{in}, h, and x_{out} are N-dimensional ones (N = 9); positive and negative components of x_0, x_{in}, h, x_{out} correspond to upward and downward bars, respectively; intact NN and its 'grandmother' hold x_0 = (–1, –1, 1, 1, 1, 1, 1, –1, –1), a kernel for the convolutional decoding/retrieval ($Q(d) > \theta$, θ = 4); peak is identified in panel *d*. $sd = bd^{1/2}$, standard deviation of bd; a, amplitude; $fwhm$, full width at half maximum.

The learned NN [Equations 2-4, Figures 1 (the insertion) and 3] provides a bell-shaped tuning to a 'grandmother' neuron's specific set of its input activities, x_0, as it is a tool for computing the convolution, Q, or Hamming distance, D, between x_0 and current set of the neuron's input activities, $x_{in} = x(d)$; i.e. it performs simply a given (Inequality 6) normalization of a current input and threshold the result (see [7] for numerical examples).

6 Generalization by Computation through Memory Performance

Since $P(q,\theta)$ defines (Equation 5) the fraction of vectors $x_{in} \neq x_0$ leading, along with x_0, to successful retrieval of the trace x_0 from the learned NN (the insertion in Figure 1 and box 2 in Figure 2), the probability of memory retrieval, $P(q,\theta)$, and generalization ability by computation through memory, $g(q,\theta)$, are numerically equal, $g(q,\theta) = P(q,\theta)$. In Table 1 generalization abilities for AMUs containing intact and damaged NNs mentioned are compared.

Usually, generalization is considered as a function of the relative size $\alpha = k/N$ of the training set of k examples and the learning strategy. It was found that for very large networks ($N \rightarrow \infty$) and $\alpha >> 1$ the error of generalization decreases as $\sim \alpha^{-1}$ [11] but the problem of generalization by learning from very few examples

remains unsolved in theory [1]. In the approach proposed learning even from one example is easily possible (Section 4.4). Values of $g(q,\theta)$ in Table 1 provide optimal (the best in the sense of pattern recall/recognition quality) generalization abilities by computation through memory; $g(0,6) = g(0,0) \sim 1\%$ was for example chosen as that is typical for professionals [7].

Table 1

Generalization ability, $g(q,\theta) = n(q,\theta)/n(q)$, for an AMU storing the trace $x_0 = (-1, -1, 1, 1, 1, 1, 1, -1, -1)$[1].

q	Intact NN, $g(q,6)$,%[2]	Damaged NN, $g(q,0)$,%[3]	q	Intact NN, $g(q,6)$,%	Damaged NN, $g(q,0)$,%
1	2	3	4	5	6
0/9	10/512 = 1.953	10/512 = 1.953	5/9	5/16 = 31.250	630/2016 = 31.250
1/9	9/256 = 3.516	81/2304 = 3.516	6/9	4/8 = 50.000	336/672 = 50.000
2/9	8/128 = 6.250	288/4608 = 6.250	7/9	3/4 = 75.000	108/144 = 75.000
3/9	7/64 = 10.938	588/5376 = 10.938	8/9	2/2 = 100.000	18/18 = 100.000
4/9	6/32 = 18.750	756/4032 = 18.750	9/9	1/1 = 100.000	1/1 = 100.000

[1] $q = 1 - d = 1 - m/N$ ($0 \le m \le N$, $N = 9$), intensity of cue ($q = 0$, free recall; $0 < q < 1$, cued recall; $q = 1$, recognition); θ, the neuron's triggering threshold; for definitions of $n(q,\theta)$ and $n(q)$ see Section 4.3.

[2] Values of $g(q,6)$ were calculated by Equation 5 or 6, results are equal.

[3] Values of $g(q,0)$ were calculated by Equation 5; 30 disrupted interneuron connections (entrance-layer neuron, exit-layer neuron) are the follows: (2,1), (4,1), (5,1), (6,1), (8,1), (3,2), (5,2), (7,2), (1,3), (4,3), (5,3), (2,4), (4,4), (2,5), (3,5), (7,5), (9,5), (3,6), (7,6), (8,6), (9,6), (1,7), (2,7), (4,7), (8,7), (1,8), (5,8), (3,9), (6,9), (7,9); this set was chosen to illustrate the fact that similar to intact NNs damaged NNs can also provide the best decoding/retrieval/generalization performance (in columns 2(5) and 3(6) generalization abilities coincide completely).

7 Conclusion

A solution of the problem of generalization by computation through memory has been illustrated by a 'grandmother' theory for vision introduced using the recent NNAMM [2]. Exact optimal calculations of such a generalization as a function of the cue index q and neuron's triggering threshold θ are performed; the approach considered provides generalization ability by learning from one example. A binary NN discussed could also be a universal circuit underlying the bell-shaped tuning of neurons in different brain areas.

Acknowledgments

I am grateful to the Health InterNetwork Access to Research Initiative (HINARI) for free on-line access to current full-text research journals and my family and my friends for their help and support.

Bibliography

1. T.Poggio and E.Bizzi. Generalization in vision and motor control. *Nature*, 2004, 431(7010), 768-774.
2. P.M.Gopych. A neural network assembly memory model based on an optimal binary signal detection theory. *Programming Problems*, 2004, no. 2-3, 473-479; http://arXiv.org/abs/cs.AI/0309036.
3. P.M.Gopych. Determination of memory performance. *JINR Rapid Communications*, 1999, 4[96]-99, 61-68 (in Russian).
4. J.J.Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings National Academy of Sciences, USA*, 1982, 79(8), 2554-2558.
5. P.M.Gopych. Sensitivity and bias within the binary signal detection theory, BSDT. *Int. Journal Information Theories & Applications*, 2004, 11(4), 318-328.
6. Yingxu Wang, D.Liu, Ying Wang. Discovering the capacity of human memory. *Brain and Mind*, 2003, 4(2), 189-198.
7. P.M.Gopych. Identification of peaks in line spectra using the algorithm imitating the neural network operation. *Instruments and Experimental Techniques*, 1998, 41(3), 341-346.
8. P.M.Gopych. ROC curves within the framework of neural network assembly memory model: some analytic results. *Int. Journal Information Theories & Applications*, 2003, 10(2), 189-197.
9. P.K.Dash, A.E.Hebert and J.D.Runyan. A unified theory for systems and cellular memory consolidation. *Brain Research Reviews*, 2004, 45(1), 30-37.

10. G.C.DeAngelis, I.Ohzava and R.D.Freeman. Receptive-field dynamics in the central visual pathways, *Trends in Neurosciences*, 1995, 18(10), 451-458.
11. M.Opper. Statistical mechanics of generalization. In *The Handbook of Brain Theory and Neural Networks,* Michael A. Arbib editor, pages 922-925. The MIT Press, Cambridge, Massachusetts, 1995.

Author's Information

Petro Mykhaylovych Gopych – V.N.Karazin Kharkiv National University; Svoboda Sq., 4, Kharkiv, 61077, Ukraine; e-mail: pmg@kharkov.com

NEURAL NETWORK BASED APPROACH FOR DEVELOPING THE ENTERPRISE STRATEGY

Todorka Kovacheva, Daniela Toshkova

Abstract: *Modern enterprises work in highly dynamic environment. Thus, the developing of company strategy is of crucial importance. It determines the surviving of the enterprise and its evolution. Adapting the desired management goal in accordance with the environment changes is a complex problem. In the present paper, an approach for solving this problem is suggested. It is based on predictive control philosophy. The enterprise is modelled as a cybernetic system and the future plant response is predicted by a neural network model. The predictions are passed to an optimization routine, which attempts to minimize the quadratic performance criterion.*

Keywords: *enterprise strategy, model predictive control, neural network, black-box modelling, business trends.*

Introduction

In the present paper, a Generalized Strategy Development (GSD) approach is suggested. Designing of the enterprise strategy is a very complicated process. It depends on many factors, which require a lot of variables to be taken into account. The relationships between them are complex and non-linear.

In the decision making process the managers need to know the environment characteristics in order to adapt the developed strategy. Therefore, the predictions of the environment changes are needed. They enable businesses make better strategic decisions and manage their activity more efficiently. It can also identify new opportunities for increased revenues and entering new markets. The prediction of the environment changes is a very difficult task. Price, advertising, goods seasonality, customers and competitors behaviour, global economic trends etc. are all factors that influence the overall performance of the enterprise.

Traditional forecasting methods such as regression and data reduction models are limited in their effectiveness as they make assumptions about the distribution of the underlying data, and often fail to recognize the inter-relatedness of variables. Now, a new forecasting tool is available – artificial neural networks (ANN). They are a form of artificial intelligence, which provide significant potential in economic applications by increasing the flexibility and effectiveness of the process of economic forecasting [Tal, Nazareth, 1995]. They are successfully used in various economic studies including investment, economic and financial forecast [Hsieh, 1993; Swales and Yoon, 1992; Hutchinson, Lo, and Poggio, 1994; Shaaf, 2000].

The enterprise strategy development requires not only predictions but also have to be optimized and adapted according to the environment changes. A suitable control design algorithm is needed. During the last years a number of methods for automatic control synthesis are applied for managing business processes. Many authors suggest the Model-Based Predictive Control (MBPC) algorithm to be used as a decision-making tool for handling complex integrated production planning problems [Tzafestas, Kapsiotis, Kyriannakis, 1997] and supply chain management [Braun et al., 2003]. MBPC is a very popular controller design method in the system engineering. It is a suitable technique for prediction of future behaviour of a plant.

Both, ANN and MBPC, are tools for solving complex problems under uncertainty by providing the ability to learn from the past experience and use information from various sources to control the enterprise performance.

Generalized Strategy Development approach combines the advantages of artificial neural networks and Model-Based Predictive Control algorithm to increase the effectiveness of the enterprise management in the entire decision making process and development of all functional strategies (incl. production-, marketing-, financial-, sales-, innovation strategy etc.).

Business Trends and Management Theory

The contemporary business is accomplished in highly dynamic environment. The continuous changes in the internal and external environment of the enterprise force it to apply a number of adaptation mechanisms, which contribute to its surviving and competitive power. These adaptation mechanisms are based on the degree of information availability. This makes providing the information a necessary condition for adaptation process and the adaptation itself – the most important characteristic of each system. In this regard the developing and the implementing of tools, which enables the corporate adaptation according to the environment changes becomes a strategic need.

Globalization [Кирев, 2001; Голдщейн 2002, 2003; Ганчев, 2004; Стоилова, 2004; Стоянов, 2003; Краева, 2003], ***virtualization*** [Мейтус, 2004; Баксанский, 2000; Манюшис, Смольянинов, Тарасов, 2003; Вютрих, Филипп, 1999], ***Internet*** and the developing of the ***Information Technologies*** [Христова, 1997; Върбанов, 2000; Илиев, 2003; Седлак, 2001] have a deep impact on the economic and social life of the society. These global trends determine the transition from the traditional industrial society to the information age society. A ***new economic based on knowledge*** [Applegate et al., 1996] appears and as a result the traditional managerial hierarchy seize to exist and a horizontal relationships are formed. Enterprises of a new type appear, which accomplish their activity on the global market from their founding. They overcome the spatial and time boundaries. The common name for such structures is ***"globally born"*** [Андерссон, Виктор, 2004]. These enterprises have their own mechanisms for developing, which substantially differ from those of the traditional industrial enterprises. Thus, the small national companies become multinational very fast.

The adaptation to environment changes requires new knowledge for its elements, the relationships between them, and characteristics of their functioning. Thus the concept of ***"Learning enterprise"*** [Senge, 1990] comes into being. It is based on the continuous acquiring new knowledge regarding the environment, using it for innovation strategies and this process applies to the enterprise as a whole.

The global trends in business development mentioned above cannot be considered partially. There are mutual relationships and dependencies between them. The existing of certain trend is a prerequisite for appearing and developing of another one and vice versa. Therefore, they influence the contemporary enterprise activities by forming an integrated set of strategies.

Now let us consider the modern business trends in a management theory point of view. Many authors [Каменов, 1984; Камионский, 1998; Рубцов, 2001] state that an unified management theory does not exist. There are different managerial concepts. Some of them claim to be universal, other are a tool for solving particular problems, some are not developed enough, other are just catchwords, some contribute and expand each other, and other contradict each other [Айвазян, Балкинд, Баснина, 1998]. This causes difficulties for the development the enterprise strategy, which strongly depends on the environment changes.

The experience shows that there is time delay between the problems, which arise in the practice and the developing of methods for their solving, which constitute the theory. In this regard, the new structures mentioned above – "globally born" and "learning enterprise" are not considered in the general management theory. Therefore, there is a lack of methodologically developed and scientifically based management approaches. These enterprises do not respond to the traditional rules and concepts as they arise and perform in strongly uncertain and highly dynamic environment. They need new management, approaches, which have to correspond to their characteristics and meet their requirements. These enterprises can be presented as complex nonlinear cybernetic systems. Thus, the laws of system and control theory can be applied to their management.

Model-Based Predictive Control

Model-Based Predictive Control has established itself in industry as an important form of advanced control [Townsend, Irwin, 2001]. An overview of industrial applications of advanced control methods in general can be found in Takatsu et al. [Takatsu et al, 1998] and in Qin and Badgwell [1998].

The main advantage of MBPC algorithm is the simplicity of the basic scheme, forming a feedback, which combines with adaptation capabilities. This determines its successful applying in the practice of designing control systems.

MBPC is an efficient methodology to solve complex constrained multivariable control problems in the absence, as well as in the presence of uncertainties [Mayne et al., 2000]. It makes possible the uncertainty of the plant and disturbances to be taken into account and enables the on-line optimization and control synthesis.

In general, it is used to predict the future plant behaviour. According to this prediction in the chosen period (prediction horizon), the MBPC optimizes the manipulated variables to obtain an optimal future plant response. The input of chosen length (also known as control horizon) is sent into the plant and then the entire sequence is repeated again in the next period. An important advantage of MBPC is that it allows the inclusion of constraints on the inputs and outputs.

The prediction plant model is realized with neural network. It provides predictions of the future plant response over a specified time horizon. The predictions are passed to an optimization routine to determine the control signal that minimizes the following performance criterion over the specified time horizon:

$$J = \sum_{j=N_1}^{N_2} \left(y_r(t+j) - y_m(t+j) \right)^2 + \rho \sum_{j=N_1}^{N_u} \left(u'(t+j-1) - u'(t+j-2) \right)^2$$

subject to the constraints, which are imposed on the state and control variables. The constants N_1, N_2, N_u define the horizons over which the tracking error and control increments are evaluated. The u' variable is the tentative control signal, y_r is the desired response and y_m is the network model response. The ρ value is weight coefficient.

Generalized Strategy Development approach will be introduced in Model-Based Predictive Control framework.

Generalized Strategy Development Approach

The purpose of the Generalized Strategy Development Approach is to transform the incomplete information about the environment and the processes inside the enterprise into complete strategy for its adaptation and evolution. From cybernetic point of view, this can be considered as a control system. The functional structure is given in Fig.1

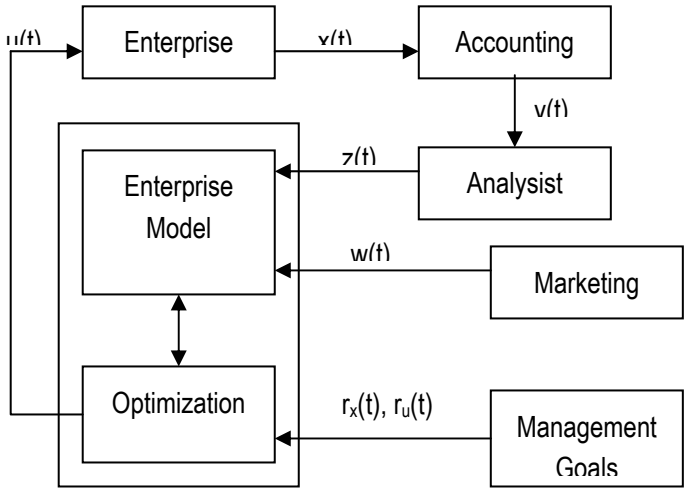


Fig.1 Global Strategy Development functional structure

Enterprise

The enterprise is a dynamic system with a high complexity. In order, the management and control [*u(t)*] to be effective we need to know its physical structure, the relationships between the constituting elements, their dynamic behaviour and the characteristics of the environment. We have to compare the current state of the enterprise to the desired state. In case they coincide entirely the management goals are achieved. In order to

register the difference we need to measure the state of the enterprise. This could be realized by performance measurement system.

Performance management is the prerequisite to performance improvement. For the enterprises to improve their performance, they must be able to measure how they are performing at present, and how they are performing after any changes. So, the companies will have the possibility to monitor if a chosen strategic direction is appropriate.

Traditional performance management systems are frequently based on cost and management accounting. There are five main difficulties with traditional management accounting techniques for performance measurement [Maskell, 1991]:

1. Management accounting reports are not relevant to strategy development;
2. Some of the data which are used for decision-making process can be distorted by cost accounting;
3. Traditional accounting reports are inflexible and are usually received too late to be of value;
4. The information about the pay-back on capital projects comes late;
5. To be of value, management accounting systems must be based on different methods and assumptions than on the financial accounts.

As traditional performance measurement systems are based on management accounting, they are primarily concerned with cost. But in today's manufacturing environment, cost based measures are no longer the only basis for decision making in enterprises. The new performance measurement systems should have some additional characteristics [Maskell, 1991].

Accounting

In Fig.1 the Accounting is the traditional performance measurement system. Therefore, the current state of the enterprise [*y(t)*] is represented by the measured one from the accounting and measurement error. The reports, which are formed by the accounting, need to be interpreted in order to be useful for the management. This task is performed by analyst.

Analyst

The accounting information is now manipulated for giving proper estimate for the current enterprise state [*z(t)*]. The manipulations include: recapitulation, generalization, estimation, recalculation etc. in order to analyze the entire enterprise activity. The results are used by managers to make decisions about the future behaviour of the enterprise.

The analysis is performed on the basis of incomplete information about the environment changes. Another error is formed. The obtained information is passed to the prediction model of the enterprise in order to minimize the tracking error.

Enterprise model

The model is used to determine the direction in which changes in the manipulated variables will improve performance. The plant operating conditions are then changed by a small amount in this direction, and a new, updated model is evaluated. The enterprise is a complex, dynamic and non-linear plant. Also different disturbances affect the its performance. Because of that, there is a lack of knowledge on the function or construction of the system.

The process output can be predicted by using a model of the process to be controlled. Any model that describes the relationship between the input and the output of the process can be used and a disturbance or noise model can be added to the process model [Duwaish, Naeem, 2001]. We can build a model using the observations of the enterprise activities.

Therefore the enterprise can be viewed as a black-box [Sjoberg et al., 1995] which aims to describe the relationships between input/output data. The non-linearities and the disturbances are taken into consideration.

During the past few years, several authors [Narendra and Parthasarathy, 1990; Nerrand et al. 1994] have suggested neural networks for nonlinear dynamical black-box modelling. To date, most of the work in neural black-box modelling has been performed making the assumption that the process to be modeled can be described accurately by neural models, and using the corresponding input-output neural predictors [Rivals,

Personnaz, 1996]. Therefore, artificial neural networks are used as effective black-box function approximators with learning and adaptation capabilities.

Marketing

We could receive information [*w(t)*] about the environment changes from the Marketing Information System (MIS) which is used in the enterprise. It is passed to the enterprise model by taking into account the forming of a new error. This error is due to impossibility of MIS to register all trends in global economy and social life of the society.

Optimization and Management Goals

Using the enterprise model, we predict the future plant response and taking into consideration the management goals [rx(t), ru(t)] we optimize it and develop a new management strategy. This is an iterative process, which provides the continuous enterprise adaptation to the environment changes.

Conclusion

Strategy development is a complex task in the continuously changing environment. The enterprise management must combine internal and external information in order to survive and evaluate. Therefore, the company needs an efficient control and strategy development and evaluation system to work in rapidly changing business conditions.

The Generalized Strategy Development approach suggested here is very suitable for this problem, namely for optimization and adaptation of the strategy development process. Thus, the effectiveness of management is increased.

Bibliography

[Айвазян, Балкинд, Баснина, 1998] Айвазян, С. А., Балкинд, О. Я., Баснина, Т. Д., и др., Стратегии бизнеса: Аналитический справочник. / Под ред. Г. Б. Клейнера. – М.: КОНСЭКО,1998

[Андерссон, Виктор, 2004] Андерссон, С., Виктор, И., Инновационная интернационализация в новых фирмах, Международный журнал “Проблемы теории и практики управления”, 1/2004

[Баксанский, 2000] Баксанский, О.Е., Виртуальная реальность и виртуализация реальности,//Концепция виртуальных миров и научное познание, СПб.: РХГИ, 2000 (http://sociology.extrim.ru)

[Върбанов, 2000] Върбанов, Р., Електронният бизнес като отражение на Интернет революцията в деловата сфера, Бизнес управление 3/2000, стр.83-95, СА “Д.А.Ценов”, Свищов

[Вютрих, Филипп, 1999] Вютрих, Х., Филипп, А., Виртуализация как возможный путь развития управления, Международный журнал “Проблемы теории и практики управления”, 5/1999

[Ганчев, 2004] Ганчев, П., Глобализацията и българските национални интереси, Диалог, СА “Д.А.Ценов” – Свищов, 2/2004, стр.32-47

[Гольдштейн, 2003] Гольдштейн, Г. Я., Основы менеджмента, ТРТУ, Таганрог, 2003

[Гольдштейн, 2002] Гольдштейн, Г. Я., Стратегический иновационный менеджмент: тенденции, технологии, практика, ТРТУ, Таганрог, 2002

[Илиев, 2003] Илиев, П., Виртуална организация на електронния бизнес, Сборник доклади от научна конференция “Иновации и трансформации на организираните пазари в България”, ИУ-Варна, 2003, стр.147-153

[Каменов, 1984] Каменов, С., Ефективност на управлението, Издателство “Г.Бакалов”, Варна, 1984

[Камионский, 1998] Камионский, С. А., Менеджмент в российском банке: опыт системного анализа и управления/ Общая ред. и предисловие Д. М. Гвишиани. М.: Деловая библиотека “Омскпромстройбанка”, 1998

[Кирев, 2001] Кирев, Л., Относно факторите за глобализация на научноизследователската дейност от транснационалните корпорации, Бизнес управление, 4/2001, стр.49-64, СА “Д.А.Ценов” – Свищов

[Краева, 2003] Краева, В., Глобални аспекти на електронната търговия, Сборник доклади от научна конференция “Иновации и трансформации на организираните пазари в България”, ИУ-Варна, 2003, стр.192-197

[Манюшис, Смольянинов, Тарасов, 2003] Манюшис, А., Смольянинов, В., Тарасов, В., Виртуальное предприятие как эффективная форма организации внешнеэкономической деятельности компании, Международный журнал “Проблемы теории и практики управления”, 4/2003

[Мейтус, 2004] Мейтус, В., Виртуализация производства, Международный журнал “Проблемы теории и практики управления”, 1/2004

[Рубцов, 2001] Рубцов, С., Целевое управление корпорациями, Москва, 2001

[Седлак, 2001] Седлак, Я., Мировая экономика: возможность неожиданных потрясений, Международный журнал “Проблемы теории и практики управления”, 5/2001

[Стоилова, 2004] Стоилова, М., Глобализацията – познати плюсове и минуси, Диалог, СА "Д.А.Ценов" – Свищов, 1/2004, стр.63-68
[Стоянов, 2003] Стоянов, В., Пазар, трансформация, глобализация, нов световен ред, Галик, София, 2003
[Христова, 1997] Христова, С., Материали от дискусия, Предизвикателствата на информационните технологии на прага на 21-и век, Национална научна конференция АИ'97, Автоматика и информатика 5/6 – 1997, САИ, София, стр.10-11
[Abdallah, Al-Thamier, 2004] Abdallah, J., Al-Thamier, A., The Economic-environmental Neural Network Model for Electical Power Dispatching, Journal of Applied Sciences 4 (3): 340-343, 2004
[Applegate et al, 1996] Applegate, L. M., McFarlan, F. W., McKenney, J. L., Corporate information systems management: text and cases, 4th ed., IRWIN, USA, 1996
[Braun et al., 2003] Braun, M., Rivera, D., Carlyle, W., Kempf, K., A Model Predictive Control Framework For Robust Management Of Multi-Product, Multi-Echelon Demand Networks, Annual Reviews in Control, 2003, vol.27, no. 2, pp.229-245(17)
[Duwaish, Naeem, 2001] Duwaish, H., Naeem, W., Nonlinear model predictive control of Hammerstein and Wiener Models using genetic algorithms, In Proceedings of the 2001 IEEE International Conference on Control Applications (CCA'01), 5-7 September, Mexico City, Mexico, pp.465-469, IEEE
[Hsieh, 1993] Hsieh, C., Some potential applications of artificial neural system in financial management. Journal of System Management, April, 1993, 12-15
[Hutchinson, Lo, and Poggio, 1994] Hutchinson, J., Lo, A., and Poggio, T., A nonparametric approach to the pricing and hedging of derivative securities via learning networks. Journal of Finance, 49, 851-99, 1994
[Maskell, 1991] Maskell, B. H., Performance Measurement for World Class Manufacturing, Productivity Press, Cambridge, Massachusetts, 1991
[Mayne et al., 2000] Mayne, D. Q., J. B. Rawlings, C. V. Rao and P. O. M. Scokaert, Constrained model predictive control: Stability and optimality, Automatica, 36, 2000, 789-814
[Narendra and Parthasarathy, 1990] Narendra, K. S., Parthasarathy, K., Identification and control of dynamical sysems using neural networks, IEEE Trans. on Neural Networks 1 (1990), pp. 4-27
[Nerrand et al. 1994] Nerrand, O., Roussel-Ragot, P., Urbani, D., Personnaz, L., Dreyfus, G., Training recurrent neural networks: why and how? An Illustration in Process Modelling, IEEE Trans. on Neural Networks 5 (1994), pp. 178-184
[Qin, Badgwell, 1998] Qin, S. Joe and Badgwell, T.A., An Overview of Non-linear Model Predictive Control Applications, Proc. Workshop on NMPC, Ascona, Switzerland, 1998
[Rivals, Personnaz, 1996] Rivals, I., Personnaz, L., Black-box modelling with state-space neural networks. In: "Neural Adaptive Control Technology", R. Zbikowski and K. J. Hunt eds., World Scientific (1996), pp. 237-264
[Senge, 1990] Senge, P. M., The Fifth Discipline. The Art & Practice of The Learning Organization. – Currency Doubleday, 1990
[Sjoberg et al., 1995] Sjoberg, J., Zhang, Q., Ljung, L., Benveniste, A., Delyon, B., Glorennec, P.Y., Hjalmarsson, H., Juditsky, A., Nonlinear black-box modelling in system identification: an unified overview. Automatica, 12. 1691-1724, 1995
[Shaaf, 2000] Shaaf, M., Predicting recession using the yield curve: an artificial intelligence and econometric comparison. Eastern Economic Journal, 26(2), Spring, 2000
[Swales, Yoon, 1992] Swales, G. S., Yoon, Y., Applying artificial neural networks to investment analysis. Financial Analyst Journal, September-October, 70-80, 1992
[Takatsu et al., 1998] Takatsu, Haruo, Itoh, Toshiaki and Araki, Mituhiko; Future needs for the control theory in industries – report and topics of the control technology survey in Japanese industry, J.Proc.Cont., Vol.8, no 5-6, 369-374, 1998
[Tal, Nazareth, 1995] Tal, B., Nazareth, L., Artificial Intelligence and Economic Forecasting, Canadian Business Economics, Volume 3, Number 3, Spring 1995
[Townsend, Irwin, 2001] Townsend , S., Irwin, G., Non-linear model based predictive control using multiple local models. In: B. Kouvaritakis, & M. Cannon (Eds.), Non-linear predictive control: Theory and practice, IEE Control Engineering Series, IEE, London, 61(11), 47-56
[Tzafestas, Kapsiotis, Kyriannakis, 1997] Tzafestas, S., G. Kapsiotis and E. Kyriannakis (1997). Model-based predictive control for generalized production planning problems. Computers in Industry 34(2), 201-210

Authors' Information

Todorka Kovacheva – Economical University of Varna, Kniaz Boris Str, e-mail: todorka_kovacheva@yahoo.com
Daniela Toshkova – Technical University of Varna, 1, Studentska Str., Varna, 9010, e-mail: daniela_toshkova@abv.bg

NEURO-FUZZY KOLMOGOROV'S NETWORK WITH A HYBRID LEARNING ALGORITHM

Yevgeniy Bodyanskiy, Yevgen Gorshkov, Vitaliy Kolodyazhniy

Abstract. *In the paper, a novel Neuro-Fuzzy Kolmogorov's Network (NFKN) is considered. The NFKN is based on and is the development of the previously proposed neural and fuzzy systems using the famous Kolmogorov's superposition theorem (KST). The network consists of two layers of neo-fuzzy neurons (NFNs) and is linear in both the hidden and output layer parameters, so it can be trained with very fast and simple procedures: the gradient-descent based learning rule for the hidden layer, and the recursive least squares algorithm for the output layer. The validity of theoretical results and the advantages of the NFKN in comparison with other techniques are confirmed by experiments.*

1. Introduction

According to the Kolmogorov's superposition theorem (KST) [1], *any* continuous function of *d* variables can be *exactly* represented by superposition of continuous functions of one variable and addition:

$$f(x_1,\ldots,x_d)=\sum_{l=1}^{2d+1} g_l\left[\sum_{i=1}^{d}\psi_{l,i}(x_i)\right], \tag{1}$$

where $g_l(\bullet)$ and $\psi_{l,i}(\bullet)$ are some continuous univariate functions, and $\psi_{l,i}(\bullet)$ are independent of *f*. Aside from the exact representation, the KST can be used as the basis for the construction of parsimonious universal approximators, and has thus attracted the attention of many researchers in the field of soft computing.

Hecht-Nielsen was the first to propose a neural network implementation of KST [2], but did not consider how such a network can be constructed. Computational aspects of approximate version of KST were studied by Sprecher [3], [4] and Kůrková [5]. Igelnik and Parikh [6] proposed the use of spline functions for the construction of Kolmogorov's approximation. Yam *et al* [7] proposed the multi-resolution approach to fuzzy control, based on the KST, and proved that the KST representation can be realized by a two-stage rule base, but did not demonstrate how such a rule base could be created from data. Lopez-Gomez and Hirota developed the Fuzzy Functional Link Network (FFLN) [8] based on the fuzzy extension of the Kolmogorov's theorem. The FFLN is trained via fuzzy delta rule, whose convergence can be quite slow. The authors proposed a novel KST-based universal approximator called Fuzzy Kolmogorov's Network (FKN) with simple structure and training procedure with high rate of convergence [9–11]. However, this training algorithm may require a large number of computations in the problems of high dimensionality. In this paper we propose an efficient and computationally simple learning algorithm, whose complexity depends linearly on the dimensionality of the input space.

2. Network Architecture

The NFKN is comprised of two layers of neo-fuzzy neurons (NFNs) [12] and is described by the following equations:

$$\hat{f}(x_1,\ldots,x_d)=\sum_{l=1}^{n} f_l^{[2]}(o^{[1,l]}),\quad o^{[1,l]}=\sum_{i=1}^{d} f_i^{[1,l]}(x_i),\quad l=1,\ldots,n, \tag{2}$$

where *n* is the number of hidden layer neurons, $f_l^{[2]}(o^{[1,l]})$ is the *l*-th nonlinear synapse in the output layer, $o^{[1,l]}$ is the output of the *l*-th NFN in the hidden layer, $f_i^{[1,l]}(x_i)$ and is the *i*-th nonlinear synapse of the *l*-th NFN in the hidden layer.

The equations for the hidden and output layer synapses are

$$f_i^{[1,l]}(x_i)=\sum_{h=1}^{m_1}\mu_{i,h}^{[1]}(x_i)w_{i,h}^{[1,l]},\quad f_l^{[2]}(o^{[1,l]})=\sum_{j=1}^{m_2}\mu_{l,j}^{[2]}(o^{[1,l]})w_{l,j}^{[2]},\ l=1,\ldots,n,\quad i=1,\ldots,d, \tag{3}$$

where m_1 and m_2 is the number of membership functions (MFs) per input in the hidden and output layers respectively, $\mu_{i,h}^{[1]}(x_i)$ and $\mu_{l,j}^{[2]}(o^{[1,l]})$ are the MFs, $w_{i,h}^{[1,l]}$ and $w_{l,j}^{[2]}$ are tunable weights.

Nonlinear synapse is a single input-single output fuzzy inference system with crisp consequents, and is thus a universal approximator [13] of univariate functions. It can provide a piecewise-linear approximation of any functions $g_l(\bullet)$ and $\psi_{l,i}(\bullet)$ in (1). So the NFKN, in turn, can approximate any function $f(x_1,\ldots,x_d)$.

The output of the NFKN is computed as the result of two-stage fuzzy inference:

$$\hat{y} = \sum_{l=1}^{n}\sum_{j=1}^{m_2}\mu_{l,j}^{[2]}\left[\sum_{i=1}^{d}\sum_{h=1}^{m_1}\mu_{i,h}^{[1]}(x_i)w_{i,h}^{[1,l]}\right]w_{l,j}^{[2]}. \tag{4}$$

The description (4) corresponds to the following two-level fuzzy rule base:

$$\text{IF } x_i \text{ IS } X_{i,h} \text{ THEN } o^{[1,1]} = w_{i,h}^{[1,1]}d \text{ AND}\ldots\text{AND } o^{[1,n]} = w_{i,h}^{[1,n]}d,\ i=1,\ldots,d,\quad h=1,\ldots,m_1, \tag{5}$$

$$\text{IF } o^{[1,l]} \text{ IS } O_{l,j} \text{ THEN } \hat{y} = w_{l,j}^{[2]}n,\quad l=1,\ldots,n,\quad j=1,\ldots,m_2, \tag{6}$$

where $X_{i,h}$ and $O_{l,j}$ are the antecedent fuzzy sets in the first and second level rules, respectively. Each first level rule contains *n* consequent terms $w_{i,h}^{[1,1]}d,\ldots,w_{i,h}^{[1,n]}d$, corresponding to *n* hidden layer neurons.

Total number of rules is

$$N_R^{FKN} = d\cdot m_1 + n\cdot m_2, \tag{7}$$

i.e., it depends *linearly* on the number of inputs *d*.

The rule base is complete, as the fuzzy sets $X_{i,h}$ in (5) completely cover the input hyperbox with m_1 membership functions per input variable. Due to the linear dependence (7), this approach is feasible for input spaces with high dimensionality *d* without the need for clustering techniques for the construction of the rule base.

Straightforward grid-partitioning approach with m_1 membership functions per input requires $(m_1)^d$ fuzzy rules, which results in combinatorial explosion and is practically not feasible for $d > 4$.

3. Learning Algorithm

The weights of the NNFKN are determined by means of a batch-training algorithm as described below. A training set containing *N* samples is used. The minimized error function is

$$E(t) = \sum_{k=1}^{N}\left[y(k) - \hat{y}(t,k)\right]^2 = \left[Y - \hat{Y}(t)\right]^T\left[Y - \hat{Y}(t)\right], \tag{8}$$

where $Y = \left[y(1),\ldots,y(N)\right]^T$ is the vector of target values, and $\hat{Y}(t) = \left[\hat{y}(t,1),\ldots,\hat{y}(t,N)\right]^T$ is the vector of network outputs at epoch *t*.

Since the nonlinear synapses (3) are linear in parameters, we can employ recursive least squares (RLS) procedure for the estimation of the output layer weights. Re-write (4) as

$$\begin{aligned}&\hat{y} = W^{[2]T}\varphi^{[2]}(o^{[1]}),\quad W^{[2]} = \left[w_{1,1}^{[2]}, w_{1,2}^{[2]},\ldots,w_{n,m_2}^{[2]}\right]^T,\\ &\varphi^{[2]}(o^{[1]}) = \left[\mu_{1,1}^{[2]}(o^{[1,1]}),\mu_{1,2}^{[2]}(o^{[1,1]}),\ldots,\mu_{n,m_2}^{[2]}(o^{[1,n]})\right]^T.\end{aligned} \tag{9}$$

Then the RLS procedure will be

$$\begin{cases}W^{[2]}(t,k) = W^{[2]}(t,k-1) + P(t,k)\varphi^{[2]}(t,k)(y(k) - \hat{y}(t,k)),\\ P(t,k) = P(t,k-1) - \dfrac{P(t,k-1)\varphi^{[2]}(t,k)\varphi^{[2]^T}(t,k)P(t,k-1)}{1+\varphi^{[2]^T}(t,k)P(t,k-1)\varphi^{[2]}(t,k)},\end{cases} \tag{10}$$

where $k=1,\ldots,N$, $P(t,0) = 10000\cdot I$, and *I* is the identity matrix of corresponding dimension.

Introducing after that the regressor matrix of the hidden layer $\Phi^{[1]} = \left[\varphi^{[1]}(x(1)),\ldots,\varphi^{[1]}(x(N))\right]^T$, we can obtain the expression for the gradient of the error function with respect to the hidden layer weights at the epoch *t*:

$$\nabla_{W^{[1]}}E(t) = -\Phi^{[1]^T}\left[Y - \hat{Y}(t)\right], \tag{11}$$

and then use the well-known gradient-based technique to update these weights:

$$W^{[1]}(t+1) = W^{[1]}(t) - \gamma(t)\frac{\nabla_{W^{[1]}}E(t)}{\left\|\nabla_{W^{[1]}}E(t)\right\|}, \tag{12}$$

where $\gamma(t)$ is the adjustable learning rate, and

$$\begin{gathered} W^{[1]} = \left[w_{1,1}^{[1,1]}, w_{1,2}^{[1,1]}, \ldots, w_{d,m_1}^{[1,1]}, \ldots, w_{d,m_1}^{[1,n]}\right]^T, \\ \varphi^{[1]}(x) = \left[\varphi_{1,1}^{[1,1]}(x_1), \varphi_{1,2}^{[1,1]}(x_1), \ldots, \varphi_{d,m_1}^{[1,1]}(x_d), \ldots, \varphi_{d,m_1}^{[1,n]}(x_d)\right]^T, \\ \varphi_{i,h}^{[1,l]}(x_i) = a_l^{[2]}(o^{[1,l]})\mu_{i,h}^{[1,l]}(x_i), \end{gathered} \tag{13}$$

and $a_l^{[2]}(o^{[1,l]})$ is determined as in [9–11]

$$a_l^{[2]}(o^{[1,l]}) = \frac{w_{l,p+1}^{[2]} - w_{l,p}^{[2]}}{c_{l,p+1}^{[2]} - c_{l,p}^{[2]}}, \tag{14}$$

where $w_{l,p}^{[2]}$ and $c_{l,p}^{[2]}$ are the weight and center of the *p*-th MF in the *l*-th synapse of the output layer, respectively. The MFs in an NFN are chosen such that only two adjacent MFs *p* and *p*+1 fire at a time [12].

Thus, the NFKN is trained via a two-stage optimization procedure without any nonlinear operations, similar to the ANFIS learning rule for the Sugeno-type fuzzy inference systems [14]. In the forward pass, the output layer weights are adjusted. In the backward pass, adjusted are the hidden layer weights. An epoch of training is considered 'successful' when root mean squared error (RMSE) on the training set is reduced in comparison with the previous epoch. Only successful epochs are counted. If RMSE is not reduced, the training cycle (forward and backward passes) is repeated until RMSE is reduced or the maximum number of cycles per epoch is reached. Once it is reached, the algorithm is considered to converge, and the parameters from the last successful epoch are saved as the result of training. Otherwise the algorithm is stopped when the maximum number of epochs is reached.

The number of tuned parameters in the hidden layer is $S_1 = d \cdot m_1 \cdot n$, in the output layer $S_2 = n \cdot m_2$, and total $S = S_1 + S_2 = n \cdot (d \cdot m_1 + m_2)$.

Hidden layer weights are initialized deterministically using the formula [9–11]

$$w_{h,i}^{[1,l]} = \exp\left\{-\frac{i\left[m_1(l-1)+h-1\right]}{d(m_1 n - 1)}\right\}, h = 1, \ldots, m_1, i = 1, \ldots, d, l = 1, \ldots, n, \tag{15}$$

broadly similar to the parameter initialization technique proposed in [6] for the Kolmogorov's spline network based on rationally independent random numbers.

Further improvement of the learning algorithm can be achieved through the adaptive choice of the learning rate $\gamma(t)$ as was proposed for the ANFIS.

4. Experimental Results

To verify the theoretical results and compare the performance of the proposed network to the known approaches, we have carried experiments with prediction and emulation of the Mackey-Glass time series [15].

The Mackey-Glass time series is generated by the following equation:

$$\frac{dy(t)}{dt} = \frac{0.2\, y(t-\tau)}{1 + y^{10}(t-\tau)} - 0.1\, y(t), \tag{16}$$

where τ is time delay. In our experiments, we used $\tau = 17$.

In the first experiment, the values $y(t-18), y(t-12), y(t-6)$, and $y(t)$ were used to predict $y(t+85)$. The NFKN used for prediction had 4 inputs, 9 neurons in the hidden layer with 3 MFs per input, and 1 neuron in the output layer with 9 MFs per synapse (189 adjustable parameters altogether). The training algorithm converged after 43 epochs.

The NFKN demonstrated similar performance as multilayer perceptron (MLP) with 2 hidden layers each containing 10 neurons. The MLP was trained for 50 epochs with the Levenberg-Marquardt algorithm. Because the MLP weights are initialized randomly, the training and prediction were repeated 10 times.

Root mean squared error on the training and checking sets (trnRMSE and chkRMSE) was used to estimate the accuracy of predictions. For the MLP, median values of 10 runs for the training and checking errors were calculated. The results are shown in Table 1 and Fig.1.

Table 1. Results of Mackey-Glass time series prediction

Network	Parameters	Epochs	trnRMSE	chkRMSE
MLP 10-10-1	171	50	0.022	0.0197
NFKN 9-1	189	43	0.018455	0.017263

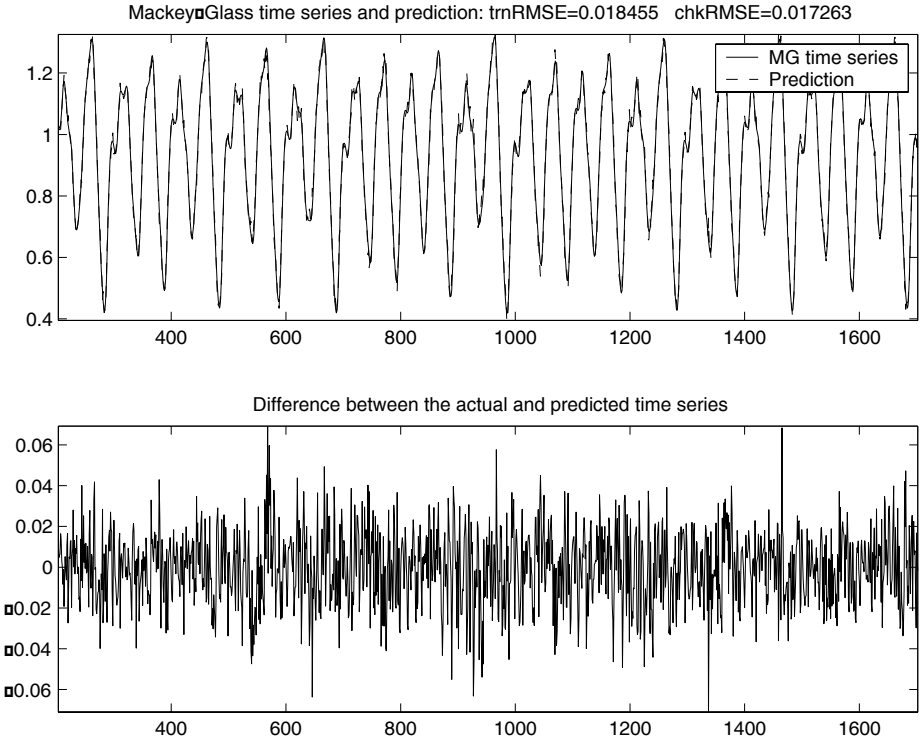


Fig. 1. Mackey-Glass time series prediction

With a performance better than that of the MLP, the NFKN with the hybrid learning algorithm requires much less computations as it does not require matrix inversions for the tuning of the synaptic weights.

The second experiment consisted in the emulation of the Mackey-Glass time series by the FKN. The values of the time series were fed into the NFKN only during the training stage. 3000 values for $t = 118,...,3117$ were used as the training set. The FKN had 17 inputs corresponding to the delays from 1 to 17, 5 neurons in the hidden layer with 5 MFs per input, and 1 neuron in the output layer with 7 MFs per synapse (total 460 adjustable parameters). The NFKN was trained to predict the value of the time series one step ahead.

The training procedure converged after 28 epochs with the final value of $RMSE_{TRN}$=0.0027 and the last 17 values of the time series from the training set were fed to the inputs of the FKN. Then the output of the network was connected to its inputs through the delay lines, and subsequent 1000 values of the NFKN output were computed. As can be seen from Fig.2, the NFKN captured the dynamics of the real time series very well. The difference between the real and emulated time series becomes visible only after about 500 time steps. The emulated chaotic oscillations remain stable, and neither fades out nor diverge. In such a way, the FKN can be used for long-term chaotic time series predictions.

Two-level structure of the rule base helps the FKN avoid the combinatorial explosion in the number of rules even with a large number of inputs (17 in the second experiment).

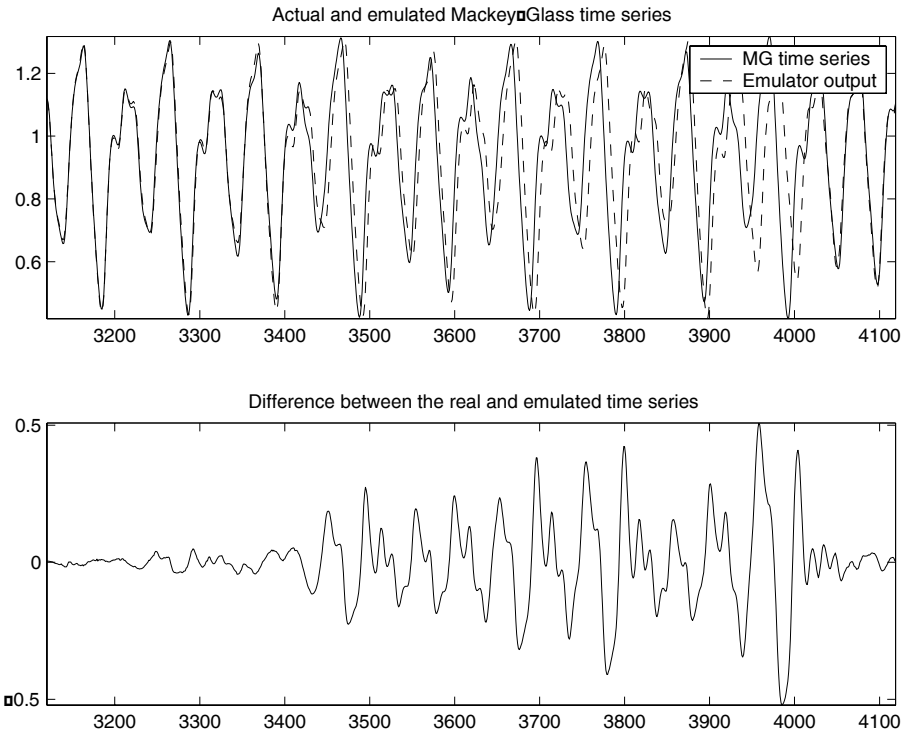


Fig. 2. Mackey-Glass time series emulation

5. Conclusion

In the paper, a new simple and efficient training algorithm for the NFKN was proposed. The NFKN contains the neo-fuzzy neurons in both the hidden and output layer and is not affected by the curse of dimensionality because of its two-level structure. The use of the neo-fuzzy neurons enabled us to develop fast training procedures for all the parameters in the NFKN.

Bibliography

1 Kolmogorov, A.N.: On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. Dokl. Akad. Nauk SSSR 114 (1957) 953-956
2 Hecht-Nielsen, R: Kolmogorov's mapping neural network existence theorem. Proc. IEEE Int. Conf. on Neural Networks, San Diego, CA, Vol. 3 (1987) 11-14
3 Sprecher, D.A.: A numerical implementation of Kolmogorov's superpositions. Neural Networks 9 (1996) 765-772
4 Sprecher, D.A.: A numerical implementation of Kolmogorov's superpositions II. Neural Networks 10 (1997) 447–457
5 Kůrková, V.: Kolmogorov's theorem is relevant. Neural Computation 3 (1991) 617-622
6 Igelnik, B., and Parikh, N.: Kolmogorov's spline network. IEEE Transactions on Neural Networks 14 (2003) 725-733
7 Yam, Y., Nguyen, H. T., and Kreinovich, V.: Multi-resolution techniques in the rules-based intelligent control systems: a universal approximation result. Proc. 14th IEEE Int. Symp. on Intelligent Control/Intelligent Systems and Semiotics ISIC/ISAS'99, Cambridge, Massachusetts, September 15-17 (1999) 213-218
8 Lopez-Gomez, A., Yoshida, S., Hirota, K.: Fuzzy functional link network and its application to the representation of the extended Kolmogorov theorem. International Journal of Fuzzy Systems 4 (2002) 690-695
9 Kolodyazhniy, V., and Bodyanskiy, Ye.: Fuzzy Neural Networks with Kolmogorov's Structure. Proc. 11th East-West Fuzzy Colloquium, Zittau, Germany (2004) 139-146
10 Kolodyazhniy, V., and Bodyanskiy, Ye.: Fuzzy Kolmogorov's Network. Proc. 8th Int. Conf. on Knowledge-Based Intelligent Information and Engineering Systems (KES 2004), Wellington, New Zealand, September 20-25, Part II (2004) 764-771
11 Kolodyazhniy, V., and Bodyanskiy, Ye.: Universal Approximator Employing Neo-Fuzzy Neurons. Proc. 8th Fuzzy Days, Dortmund, Germany, Sep. 29 – Oct. 1 (2004) CD-ROM
12 Yamakawa, T., Uchino, E., Miki, T., and Kusanagi, H.: A neo fuzzy neuron and its applications to system identification and prediction of the system behavior. Proc. 2nd Int. Conf. on Fuzzy Logic and Neural Networks "IIZUKA-92", Iizuka, Japan (1992) 477-483
13 Kosko, B.: Fuzzy systems as universal approximators. Proc. 1st IEEE Int. Conf. on Fuzzy Systems, San Diego, CA (1992) 1153-1162

14 Jang, J.-S. R.: Neuro-Fuzzy Modelling: Architectures, Analyses and Applications. PhD Thesis. Department of Electrical Engineering and Computer Science, University of California, Berkeley (1992)
15 Mackey, M. C., and Glass, L.: Oscillation and chaos in physiological control systems. Science 197 (1977) 287-289

Authors' Information

Yevgeniy Bodyanskiy – bodya@kture.kharkov.ua
Yevgen Gorshkov – ye.gorshkov@gmail.com
Vitaliy Kolodyazhniy – kolodyazhniy@ukr.net
Control Systems Research Laboratory; Kharkiv National University of Radioelectronics;
14, Lenin Av., Kharkiv, 61166, Ukraine

НЕЙРОСЕТЕВАЯ КЛАССИФИКАЦИЯ ЗЕМНОГО ПОКРОВА НА ОСНОВАНИИ СПЕКТРАЛЬНЫХ ИЗМЕРЕНИЙ

Алла Лавренюк, Лилия Гнибеда, Екатерина Яровая

Аннотация: *разработка метода классификации наземных объектов*

Ключевые слова*: нейронные сети, спектральные кривые, классификация*

Вступление

При дистанционном исследовании объектов спектральные характеристики отраженного света могут оказаться удобным и высокоинформативным источником данных. Например, с их помощью можно оценить состояние растительности для определения степени зараженности и загрязненности [Куссуль, 2003]. Так же они представляют значительный интерес для задач изучения грунтов. Использование спектральных кривых поможет классифицировать земной покров и создать электронную карту поверхности Земли, что является актуальным на сегодняшний день. Эта задача слабоформализуема, поэтому для ее решения предлагается использовать неростетевой подход [Kussul, 2003].

Постановка задачи

Исследования проводились по данным библиотеки спектральных кривых USGS (USGS Digital spectral library). Имеется множество экспериментальных данных о спектральных характеристиках отраженного излучения некоторых наземных объектов, представленных набором кривых, иллюстрирующих зависимость интенсивности излучения от длины волны. Если анализировать общий вид спектральных кривых, то их условно можно разделить на следующие классы: вода, снег, растительность. Кривые каждого класса имеют одинаковый образ, с той лишь разницей, что различия в химическом составе объектов приводят к смещениям кривых относительно осей координат.

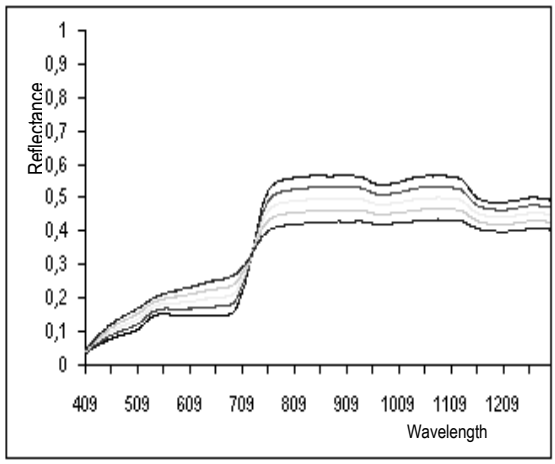


Рисунок 1. Спектральные характеристики травы

Из графиков растительности видно, что они имеют общую черту – характерный для всей зеленой растительности скачок в области видимого красного света. Но при этом их можно разделить еще на три класса: трава, хвойная растительность и лиственная, т.к. они имеют заметные различия, если анализировать образ кривой в целом (рис. 1-3).

Спектральные кривые снега делятся на два типа – одни ниспадают, другие имеют скачок в области 690-750 нм (рис. 4). Это объясняется прозрачностью снега. В тех случаях, когда сквозь снег просвечивает зеленая растительность на кривых появляется скачок в области красной составляющей видимой части

спектра, который характерен только для растений. Поэтому необходимо разделить спектральные кривые снега на два класса – снег и тающий снег с видной растительностью.

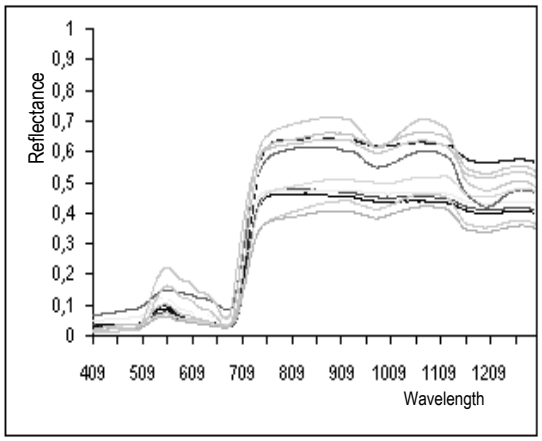


Рисунок 2. Спектральные характеристики лиственных растений

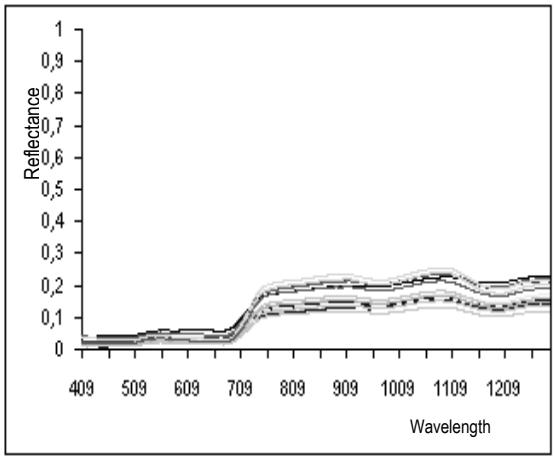


Рисунок 3. Спектральные характеристики хвойных растений

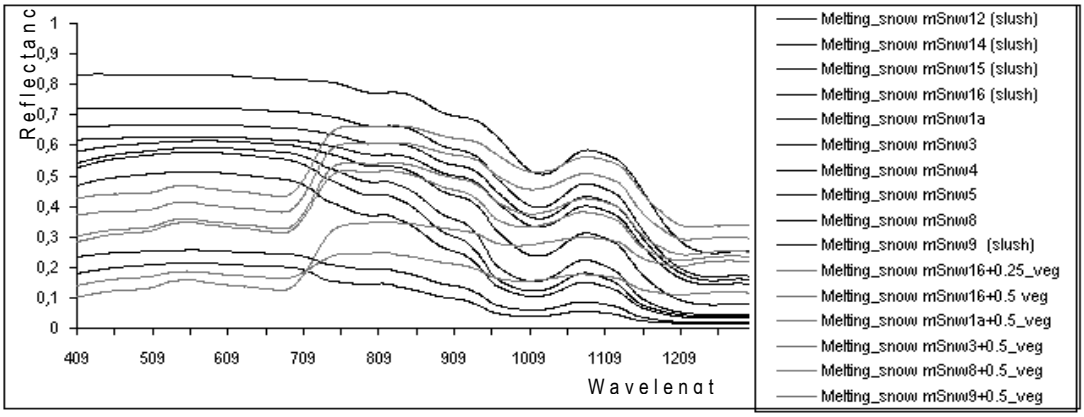


Рисунок 4. Спектральные характеристики снега

В итоге получили шесть классов: вода, снег, тающий снег с видной растительностью, трава, хвойные, лиственные. На каждый класс имеется порядка 10 спектральных кривых.

Задача идентификации наземных объектов сводится к распознаванию образов спектральных кривых и разделению их на классы. Требуется разработать эффективный метод классификации кривых. Поэтому целесообразно решать эту задачу на основе интеллектуальных методов, в частности нейронных сетей.

Решение задачи классификации с помощью нейронной сети

Основу нейронной сети (НС) составляют относительно простые, элементы, имитирующие работу нейронов мозга. Каждый нейрон характеризуется своим текущим состоянием по аналогии с нервными клетками головного мозга, которые могут быть возбуждены или заторможены. Он обладает группой синапсов – однонаправленных входных связей, соединенных с выходами других нейронов, а также имеет аксон – выходную связь данного нейрона, с которой сигнал (возбуждения или торможения) поступает на синапсы следующих нейронов. Каждый синапс характеризуется величиной синаптической связи или ее весом w_i, который по физическому смыслу эквивалентен электрической проводимости [Каллан, 2001].

Текущее состояние нейрона определяется как взвешенная сумма его входов:

$$s = \sum_{i=1}^{n} x_i \cdot w_i$$

Активационная функция

Выход нейрона есть функция его состояния: $y = f(s)$

Нелинейная функция f называется активационной и может иметь различный вид. Одной из наиболее распространенных является нелинейная функция с насыщением, так называемая логистическая функция или сигмоид (т.е. функция S-образного вида):

$$f(x) = \frac{1}{1 + e^{-\alpha x}}$$

При уменьшении α сигмоид становится более пологим, в пределе при α=0 вырождаясь в горизонтальную линию на уровне 0.5, при увеличении α сигмоид приближается по внешнему виду к функции единичного скачка. Из выражения для сигмоида очевидно, что выходное значение нейрона лежит в диапазоне [0,1]. Кроме того, сигмоид обладает свойством усиливать слабые сигналы лучше, чем большие, и предотвращает насыщение от больших сигналов, так как они соответствуют областям аргументов, где сигмоид имеет пологий наклон.

Центральная точка сигмоидной зависимости может сдвигаться вправо или влево по оси X. Это смещение может принимать произвольное значение, которое подбирается на стадии обучения вместе с весовыми коэффициентами. Такое смещение обычно вводится путем добавления к слою нейронов еще одного входа, возбуждающего дополнительный синапс каждого из нейронов, значение которого всегда равняется 1. Присвоим этому входу номер 0. Тогда текущее состояние нейронов будет определяться по следующей формуле:

$$s = \sum_{i=0}^{n} x_i \cdot w_i \quad \text{где } w0 = -T,\ x0 = 1.$$

Параметры обучения

Для обучения сетей прямого распространения применяются итеративные методы, основанные на постепенной коррекции веса межнейронных связей в направлении уменьшения ошибки реакции сети. Каждая итерация обучения, называемая эпохой, состоит в повторении всего запоминаемого материала. Поскольку число эпох может исчисляться тысячами, обучение сетей этого класса занимает много времени. Сети прямого распространения имеют многослойную организацию, причем выход каждого нейрона предыдущего слоя имеет связи с входами всех нейронов последующего слоя.

В процессе обучения может возникнуть ситуация, когда большие положительные или отрицательные значения весовых коэффициентов сместят рабочую точку на сигмоидах многих нейронов в область насыщения. Малые величины производной от логистической функции приведут к остановке обучения, что парализует НС. Эта проблема связана еще с одной, а именно – с выбором величины скорости обучения. Доказательство сходимости обучения в процессе обратного распространения основано на производных, то есть приращения весов и, следовательно, коэффициенты скорости обучения должны быть бесконечно малыми, однако в этом случае обучение будет происходить неприемлемо медленно. С другой стороны, слишком большие коррекции весов могут привести к постоянной неустойчивости процесса обучения. Поэтому в качестве коэффициента скорости обучения обычно выбирается число меньше 1, но не очень маленькое, например, 0.1, и он, вообще говоря, может постепенно уменьшаться в процессе обучения. Кроме того, для исключения случайных попаданий в локальные минимумы иногда, после того как значения весовых коэффициентов застабилизируются, коэффициент скорости обучения кратковременно сильно увеличивают, чтобы начать градиентный спуск из новой точки. Если повторение этой процедуры несколько раз приведет алгоритм в одно и то же состояние НС, можно более или менее уверенно сказать, что найден глобальный максимум, а не какой-то другой.

Фактор момента для каждого из скрытых слоев и выходного слоя определяет степень учета изменений весовых коэффициентов на предыдущем шаге обучения. Чем выше этот коэффициент, тем большее влияние на изменение весовых коэффициентов оказывают изменения на предыдущем шаге.

Организация сети

Теоретически число слоев может быть произвольным, однако фактически оно ограничено ресурсами компьютера или специализированной микросхемы, на которых обычно реализуется НС. Чем больше количество нейронов и слоев, тем шире возможности сети, тем медленнее она обучается и работает. Если нейронов и слоев слишком много, быстродействие будет низким, а памяти потребуется много; сеть будет неспособна к обобщению: в области, где нет или мало точек активационной функции выходной вектор будет случаен и непредсказуем и не будет адекватен решаемой задаче. Поэтому было

использовано небольшое количество слоев – 3 (один скрытый), чтобы сеть обобщала входную информацию, не запоминая каждую спектральную кривую, а создавая образ кривой для каждого класса.
Количество нейронов входного и выходного слоя вычисляется автоматически на основе информации о входных и выходных данных обучающей выборки, а для каждого скрытого слоя число нейронов определяется в процессе экспериментов.
Размерность входного слоя нейронной сети – это количество точек, которые содержит спектральная кривая. Каждая кривая состояла из 850 точек. При столь высокой размерности входных данных для обучения нейронной сети на большой выборке требовалось использовать сложную нейросетевую архитектуру, и процесс обучения был длительным. Поэтому мы сократили количество входных данных в две итерации. Во-первых, отбросили область кривых, где не очень заметна разница между ними (диапазон длины волны от 350 до 410 нм) и, во-вторых, усреднили кривую, заменив каждые четыре точки их средним значением по оси X и У.
В результате проведения серии экспериментов были найдены оптимальные параметры сети прямого распространения с обратным распространением ошибки:

- размерность входного слоя нейронной сети - 222
- размер скрытого слоя – 20 нейронов
- размер выходного слоя – 6 нейронов (в соответствии с количеством классов)
- функция активации – сигмоида
- веса и пороговые уровни инициализируются случайными значениями
- смещение активационной функции относительно оси X – 0,001

Значения коэффициентов:
Коэффициент скорости обучения

- Скрытый слой 0,1;
- Выходной слой 0,05;

Фактор момента

- Скрытый слой 0,15;
- Выходной слой 0,0025;

Чтобы достичь стопроцентного разделения кривых на классы при таких значениях коэффициентов количество эпох обучения составило 1000. Размер каждой эпохи обучения зависит от количества спектральных кривых предназначенных для обучения и составляет 44. В тестовой выборке 6 спектральных кривых.

Выводы

Эксперименты проводились для различных начальных значений весовых коэффициентов и различных методов обучения. По результатам экспериментов можно судить, что применение нейронных сетей для классификации земного покрова по спектральным характеристикам является эффективным. Целесообразно продолжить исследования, увеличив количество экспериментальных данных, что поможет более точно классифицировать земную поверхность.

Список литературы

[Каллан, 2001] Р. Каллан. Основные концепции нейронных сетей. Издательский дом "Вильямс", 2001

[Куссуль, 2003] Куссуль Н.Н., Марковски Дж., Яценко В.А., Саченко А.А., Скакун С.В., Сидоренко А.В.Определение содержания хлорофилла в растениях с помощью модульных нейронных сетей // Межведомственный сборник научных трудов “Кибернетика и вычислительная техника”. — 2003. — Выпуск 141. — С. 49–57.

[Kussul, 2003] Natalia Kussul, Michael Kussul, Anatoly Sachenko, George Markowsky, Sergiy Ganzha. Remote Sensing of Vegetation Using Modular Neural Networks // Proc. of the 3nd International Conference on Neural Networks and Artificial Intelligence (ICNNAI 2003). – Minsk, Belarus, November 12-14, 2003. – P. 232-234.

Информация об авторах

Алла Лавренюк – научный сотрудник, к.т.н., Институт космических исследований НАНУ-НКАУ, пр. Глушкова, 40, Киев-03187, inform@ikd.kiev.ua
Лилия Гнибеда – аспирантка, Национальный авиационный университет, пр. Комарова, 1, Киев-03058
Екатерина Яровая – аспирантка, Национальный авиационный университет, пр. Комарова, 1, Киев-03058