# ВАРИАНТ НЕЙРОСЕТЕВОГО АЛГОРИТМА РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ (СЛАУ)

Пантюхин Д.В.

Московский Физико-Технический Институт (МФТИ)
(dim_beavis@mail.ru)

**Введение**

Многие задачи обработки изображений сводятся к решению систем линейных и нелинейных алгебраических уравнений, последние, в свою очередь, также сводятся к системам линейных уравнений. Поэтому эффективное нахождение решений СЛАУ является важной подзадачей как для задач обработки изображений так и для математики в целом. В данной статье представлен лишь один из возможных методов нейросетевого решения СЛАУ. Нейросетевые методы, ориентированные на высоко-параллельные архитектуры могут занять ведущее место при решении сложных задач в области обработки изображений.

**Математическая постановка задачи**

С математической точки зрения решение СЛАУ вида A**x**=b**, с заданной точностью ε есть вектор **x*** такой, что $\|\mathbf{x}^*-\mathbf{x}^{**}\|<\varepsilon$. Если матрица A квадратная и det A≠0 то система имеет единственное решение при любом **b**. Если система имеет бесконечное число решений то необходимо указать хотя бы одно из них. Если решений нет то это также необходимо указать. С геометрической точки зрения решение СЛАУ представляет собой множество (точка, гиперплоскость или пустое множество) точек пересечения гиперплоскостей определенных каждым уравнением системы.

**Нейросетевая постановка задачи**

1. Входной вектор нейронной сети: входным вектором нейронной сети будет являться вектор составленный из компонент столбцов матрицы A и вектора **b**

2. Выходной вектор нейронной сети: приближение **x** искомого вектора $\mathbf{x}^*$.

3. Желаемый выходной вектор: $\mathbf{x}^*$ - искомый вектор (с заданной точностью).

4. Ошибка решения: в качестве ошибки решения выберем невязку системы **e**=A**x**-**b**

(1)

5 Функция активации:

$$y = f(g) = \frac{2}{\pi} arctg \quad B \cdot g \qquad g = \sum_{h_0=1}^{H_0} y_{h_0} * W_{h_1 h_0}$$ где

6 Описание структуры разомкнутой нейронной сети.

Структурная схема нейронной сети представлена на Рис.1

$$y^1 = \begin{bmatrix} A_1 \\ : \\ A_N \\ b \end{bmatrix} \qquad y^2_{h_1} = f^1\left(\sum_{h_0=1}^{H_0} y^1_{h_0} * W^1_{h_1 h_0}\right)$$

$$x_{h_2} \equiv y^3_{h_2} = f^2\left(\sum_{h_1=1}^{H_1} y^2_{h_1} * W^2_{h_2 h_1}\right) = f^2\left(\sum_{h_1=1}^{H_1} f^1\left(\sum_{h_0=1}^{H_0} y^1_{h_0} * W^1_{h_1 h_0}\right) * W^2_{h_2 h_1}\right)$$

где $A_1 \ldots A_N$-столбцы матрицы A, **b**-столбец свободных членов, $H_0=(N+1)*M$, $h_0=1..H_0$

$H_1$- число нейронов в первом слое, $h_1=1..H_1$, $H_2$ =N- число нейронов во втором слое равное размерности N вектора **x**, $h_2=1..H_2$, M-размерность вектора **b.**

Функции активации могут быть различными для разных слоев и могут отличаться параметром B.

7 Функционал оптимизации. Функционал оптимизации можно выбрать различными способами. В частном случае (при отсутствии специфических требований) выберем квадратичный функционал оптимизации: $F=e^T e$. Имеем:

$$F = \sum_{i=1}^{M} e_i^2 = \sum_{i=1}^{M} \left(\sum_{h_2=1}^{H_2} A_{ih_2} * x_{h_2} - b_i\right)^2$$

(2)

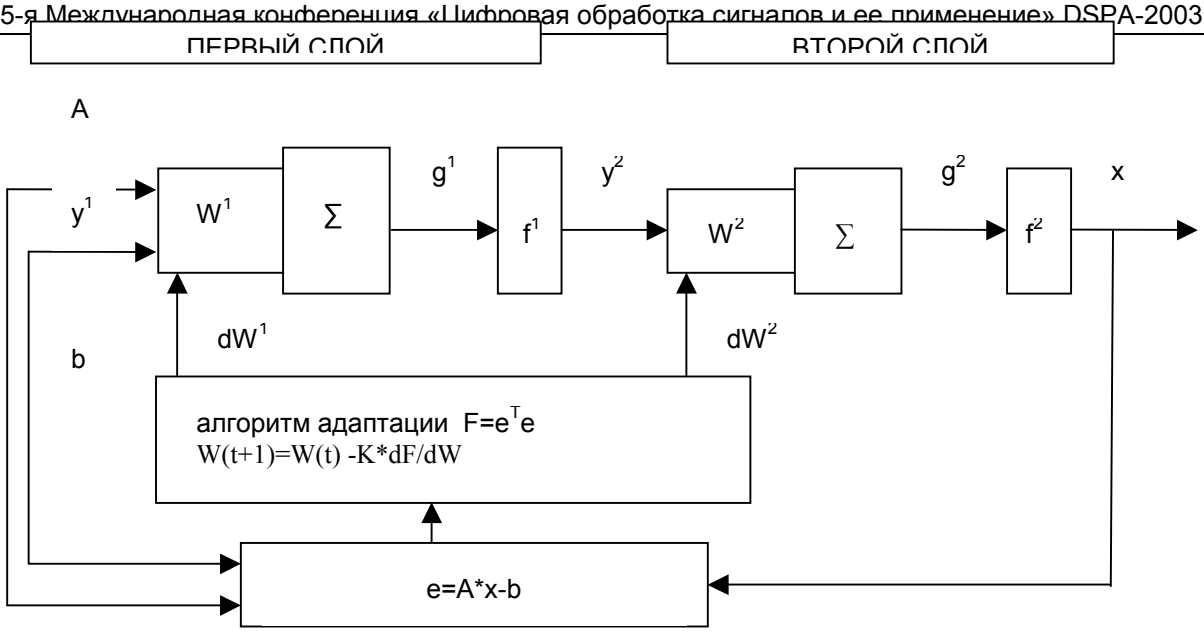



Рис.1 Нейросетевой алгоритм решения СЛАУ (двухслойная нейронная сеть)

8. Метод поиска экстремума функционала оптимизации:

Поиск экстремума (минимума) функционала оптимизации может вестись несколькими способами. В простейшем случае поиск ведется градиентным методом наискорейшего спуска :

(3) (4)

$$W(n+1)^1_{h_1h_0} = W(n)^1_{h_1h_0} - K^1 \frac{\partial F}{\partial W^1_{h_1h_0}} \qquad W(n+1)^2_{h_2h_1} = W(n)^2_{h_2h_1} - K^2 \frac{\partial F}{\partial W^2_{h_2h_1}}$$

где $K^1$ , $K^2$ – константы шага метода поиска экстремума функционала оптимизации.

Этот метод обладает существенным недостатком: он зависит от выбора начальных условий, так как при вычислении находится локальный минимум, ближайший к начальным условиям. Поэтому необходимо использовать метод, представляющий собой комбинацию градиентного метода и стохастической процедуры. В этом методе при нахождении экстремума градиентным способом запоминается значение функционала в этом локальном экстремуме. После нахождения некоторого количества локальных экстремумов, точка с наименьшим значением функционала принимается за искомое решение.

9 Выбор начальных условий для настройки.

При выборе начальных условий для двухслойной сети необходимо выбрать

$W^1_{h1h0}(0)$ и $W^2_{h2h1}(0)$. Будем задавать эти значения случайным образом.

**Алгоритм настройки нейронной сети**

Частные производные функционала оптимизации в выражениях (3), (4) имеют вид

Для p= 1..$H_2$ q=1..$H_1$ (5) для p=1..$H_1$ q=1..$H_0$ (6)

$$\frac{\partial F}{\partial W^2_{pq}} = 2(\sum_{i=1}^{M} e_i A_{ip}) \frac{\partial f^2}{\partial g^2_p} y^2_q \qquad \frac{\partial F}{\partial W^1_{pq}} = 2y^1_q \frac{\partial f^1}{\partial g^1_p} \sum_{h_2=1}^{H_2} (\sum_{i=1}^{M} e_i A_{ih_2}) W^2_{h_2p} \frac{\partial f^2}{\partial g^2_{h_2}}$$

Алгоритм адаптации весовых коэффициентов нейронной сети получается подстановкой выражений (5) и (6) в (3) и (4).

**Настройка параметра B**

Пока будем считать параметр B детерминированным, однако в общем случае его настройку следует проводить.

**Типовые входные векторы**

На первом этапе исследований в качестве типовых входных векторов принимались СЛАУ для которых решение может быть получено другими методами (в частности СЛАУ двух переменных) для наглядности и демонстрации динамики процесса нахождения решения.

**Экспериментальные результаты**

На рис.2 и рис.3 показаны типичные зависимости функционала оптимизации от (дискретного) времени. Видно что варьируя число $H_1$ нейронов в скрытом слое можно добиться существенного ускорения процесса нахождения решения.

(Последняя цифра по шкале времени есть число шагов потребовавшееся для решения системы (7) с точностью 1e-6 при моделировании в системе MATLAB )

$$\begin{cases} 3x_1 + 2x_2 = -0.5 \\ -x_1 + 6x_2 = 3.5 \end{cases} \qquad (7)$$

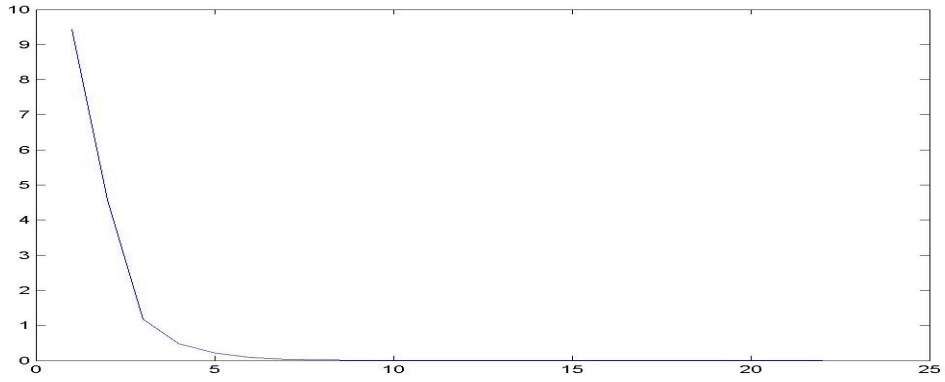



Рис.2 Представлен типичный вид функционала оптимизации на различных шагах итерации.

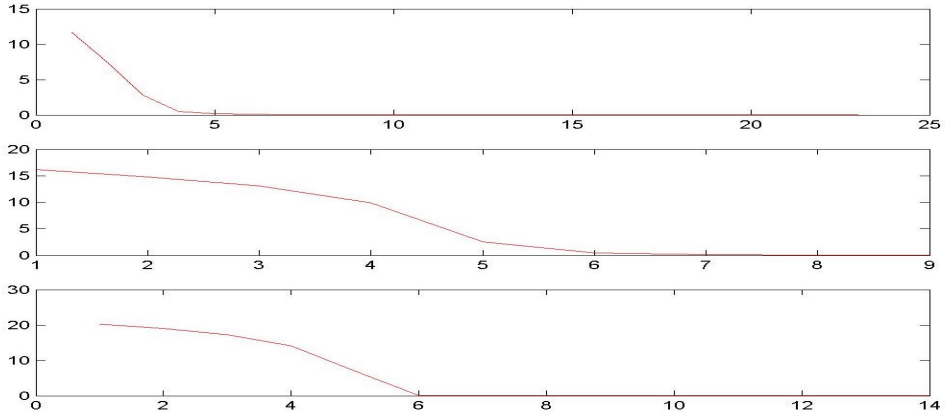



Рис.3 Влияние числа нейронов в первом слое на скорость решения СЛАУ. (сверху вниз $H_1$=3,6,9).

**ONE NEURAL NETWORK ALGORITHM FOR SOLVING SYSTEMS OF THE LINEAR ALGEBRAIC EQUATIONS (SLAE)**

Pantjuhin D.

Student of Moscow Institute of Physics and Technology (MIPT)
(dim_beavis@mail.ru)

**Introduction**

Many tasks of image processing are reduced to solution of systems of the linear and nonlinear algebraic equations, last, in turn, are also reduced to systems of the linear equations. Therefore, effective finding of the solutions of SLAE is the important subtask both for tasks of image processing and for mathematics as a whole. In given article only one methods of neural networks solution SLAE is submitted. Neural networks methods focused on high - parallel architecture can be of great importance at the solution of complex tasks in the field of the image processing.

**Mathematical task**

Mathematically the solution of SLAE A**x**=b**, with the given accuracy ε is a vector **x*** such, that $\|\mathbf{x}^*-\mathbf{x}^{**}\|<\varepsilon$. If matrix A is square and det A≠0 then the system has the unique solution at anyone **b**. If the system has infinite number of solutions it is necessary to specify one of them. If there is no solution present it is also necessary to specify this. Geometrically the solutions of SLAE represent a set (one point, hyper-plane or empty set) of points of hyper-planes intersections, which are determined by each equation.

**Neural network task**

1. Input vector of neural network: it will be a vector, made from columns of matrix A and vector **b**

2. Output vector of neural network: approaching **x** of desirable vector $\mathbf{x}^*$.

3. Desirable output vector: $\mathbf{x}^*$ - solution of SLAE (with given precision ε).

4. Solution error: it will be a discrepancy of SLAE **e**=A**x**-**b** (1)

5 Activation function:

$$y = f(g) = \frac{2}{\pi} arctg \quad B \cdot g \qquad \text{where} \qquad g = \sum_{h_0=1}^{H_0} y_{h_0} * W_{h_1 h_0}$$

6 Structure of open neural network:

The block diagram of neural network is represent on Img..1

$$y^1 = \begin{bmatrix} A_1 \\ \vdots \\ A_N \\ b \end{bmatrix} \qquad y^2_{h_1} = f^1\left(\sum_{h_0=1}^{H^0} y^1_{h_0} * W^1_{h_1 h_0}\right)$$

$$x_{h_2} \equiv y^3_{h_2} = f^2\left(\sum_{h_1=1}^{H^1} y^2_{h_1} * W^2_{h_2 h_1}\right) = f^2\left(\sum_{h_1=1}^{H^1} f^1\left(\sum_{h_0=1}^{H^0} y^1_{h_0} * W^1_{h_1 h_0}\right) * W^2_{h_2 h_1}\right)$$

where $A_1 \ldots A_N$-is a columns of matrix A, **b**-vector column, $H_0=(N+1)*M$, $h_0=1..H_0$

$H_1$- number of neurons in first (shadow) layer, $h_1=1..H_1$; $H_2$ =N- number of neurons in second layer equal to dimension N of vector **x**, $h_2=1..H_2$; M-dimension of vector **b.**

Activation functions can be various in different layers, and can differ in parameter B.

7 Optimization functional: optimization functional can choose by various ways. In our case (at absence of specific requirements) we shall choose square-law функционал of optimization: $F=e^T e$. We have:

$$F = \sum_{i=1}^{M} e_i^2 = \sum_{i=1}^{M} \left(\sum_{h_2=1}^{H_2} A_{ih_2} * x_{h_2} - b_i\right)^2 \qquad (2)$$

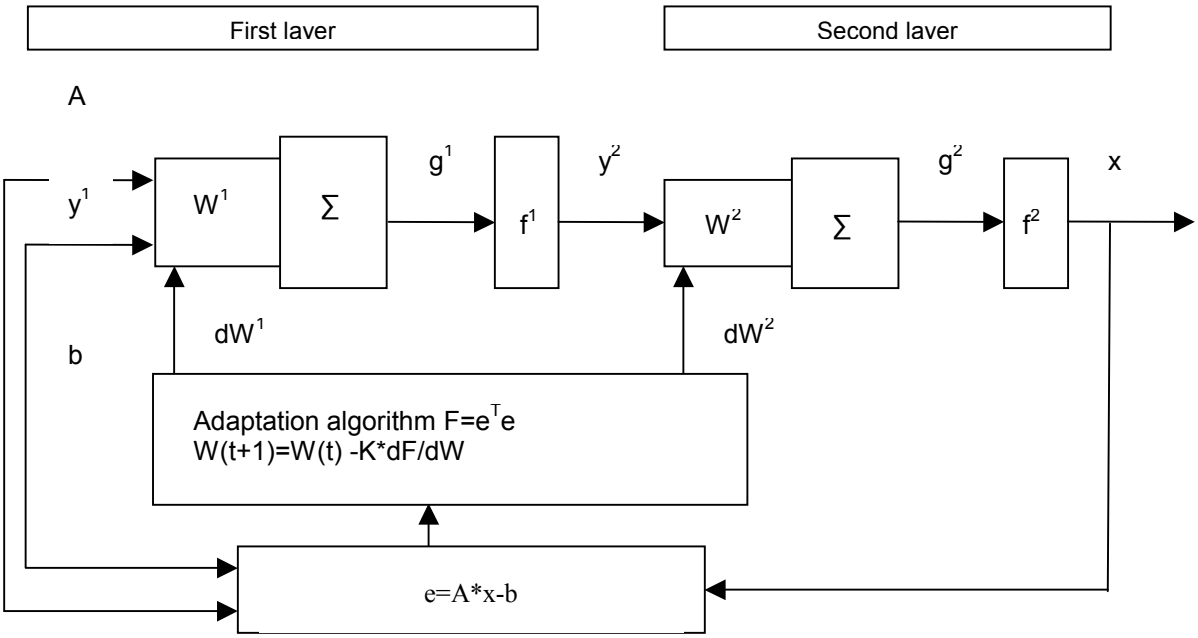
8. Method of search of extremum of optimization functional:

The search of extremum (minimum) of optimization functional can be conducted in several ways. In our elementary case the search is conducted by a gradient method of the quickest descent:

(3) (4)

$$W(n+1)^1_{h_1 h_0} = W(n)^1_{h_1 h_0} - K^1 \frac{\partial F}{\partial W^1_{h_1 h_0}} \qquad W(n+1)^2_{h_2 h_1} = W(n)^2_{h_2 h_1} - K^2 \frac{\partial F}{\partial W^2_{h_2 h_1}}$$

where $K^1$ , $K^2$ – constant of search method.




Img.1 Neural network for solving SLAE (two-layers network)

This method has essential lack: it depends on a choice of the initial conditions, as it calculates a local minimum, nearest to the initial conditions. Therefore, it is necessary to use a method representing a combination of a gradient method and stochastic procedure. In this method remembers value of optimization functional in local extremum. After the least value at some local extremums is accepted for the required solution.

9 Choice of initial conditions.

For two-layers network it is necessary to choose $W^1_{h1h0}(0)$ и $W^2_{h2h1}(0)$. We will chose this arbitrary.

**Algorithm of neural network adjustment**

Partial derivatives of optimization functional in (3), (4) look like

for p= 1..$H_2$ q=1..$H_1$ (5) for p=1..$H_1$ q=1..$H_0$ (6)

$$\frac{\partial F}{\partial W^2_{pq}} = 2\left(\sum_{i=1}^{M} e_i A_{ip}\right)\frac{\partial f^2}{\partial g^2_p} y^2_q \qquad \frac{\partial F}{\partial W^1_{pq}} = 2\, y^1_q \frac{\partial f^1}{\partial g^1_p}\sum_{h_2=1}^{H_2}\left(\sum_{i=1}^{M} e_i A_{ih_2}\right) W^2_{h_2 p}\frac{\partial f^2}{\partial g^2_{h_2}}$$

Further it is necessary to substitute (5) and (6) in (3) and (4).

**Adjustment of parameter B**

While we shall consider parameter B as determined, however generally, its adjustment should be carried out.

**Typical input vectors**

At the first stage of researches as typical input vectors were accepted SLAE for which decision can be received by others methods (in particular SLAE of two variable) for presentation and demonstration dynamics of solution process.

**Simulation results**

On img.2 and img.3 the typical dependences of optimization functional from (discrete) time are shown. By variation of number $H_1$ of neurons in the first layer it is possible to achieve essential acceleration of solving process.

(Last number on a time scale is number of steps required for the decision of system (7) with accuracy 1e-6 at modeling in MATLAB)

$$\begin{cases} 3x_1 + 2x_2 = -0.5 \\ -x_1 + 6x_2 = 3.5 \end{cases} \quad (7)$$



Img.2 A typical shape of optimization functional.



Img.3 Dynamics of solving process with various number of neurons in first layer. (from top to bottom $H_1$=3,6,9).