П1. ЛАБОРАТОРНАЯ РАБОТА

Применение двухслойной нейронной сети для моделирования дискретной динамической системы.

Цель работы: Ознакомление с методами синтеза многослойных нейронных сетей и применение их для моделирования динамических дискретных систем. Освоение методики обучения нейронной сети с использованием принципа обратного распространения ошибки.

Общая часть

1. Введение

Нейронные сети в настоящее время получают все более широкое распространение в задачах управления и обработки информации. Способность их к обучению и высокая вычислительная эффективность благодаря возможности распараллеливания процессов обработки делают нейронные сети практически безальтернативными для решения задач с трудно формализуемыми ограничениями и целями управления.

В данной лабораторной работе для моделирования дискретной динамической системы используется нейронная сеть персептронного типа. Персептронные сети разделяются по топологии (однослойные, многослойные и др.), типу функции активации нейронов (релейная одно- и двухполярная, сигмоидальная и т.д.), наличию или отсутствию сигналов обратной связи и другим параметрам.

2. Теоретическая часть

В [1] показывается, что однослойный персептрон обладает весьма ограниченными вычислительными возможностями, поэтому для большинства задач его использование оказывается неприемлемым. Поэтому для решения задач моделирования и идентификации сравнительно простых систем управления целесообразно применять двухслойный персептрон, который сочетает в себе простоту оптимизации и гибкость в представлении исходной системы.

Как известно, дискретная динамическая система со скалярными управлением и выходом может быть описана уравнениями вида

$$y(k+1) = f(u(k), y(k)),$$
$$y(0) = a = const, \quad (\text{п}1.1)$$

где

$u(k)$ - сигнал управления в момент времени $k\,(k \geq 0)$, $y(0) = a$ -начальные условия,

$y(k)$ - сигнал выхода системы в момент времени k,

$y(k+1)$ - сигнал выхода системы в момент времени $k+1$,

$f(u, y)$ - системная функция объекта управления.

Исходя из (п1.1), эквивалентная нейронная сеть может быть представлена в виде двухслойной сети с несколькими входами, представляющими p отсчетов управления $u(k), u(k-1), ..., u(k-p+1)$ и q отсчетов выхода $y(k), y(k-1), ..., y(k-q+1)$, а также одним выходом $y(k+1)$, получаемым после обработки нейронной сетью $(p+q)$ входов.

Схема структуры такой сети приведена на рис. п1.1. Она включает два нейронных слоя, один из которых является скрытым (т.е. внутренним по отношению к границам сети), а другой - выходным. Входы сети обозначены как $x_1, ..., x_{12}$, веса при переходе «вход-скрытый слой» - $w_{i,j}^{(1)}$ (i-номер начальной вершины связи ($i = \overline{1:12}$), j - номер

конечной вершины ($j = \overline{1:8}$), (1) - индекс слоя), а веса «скрытый слой-выходной слой» - $w_{j,r}^{(2)}$ ($r = 1$ - номер нейрона в выходном слое). Нейроны, каждый из которых включает в себя сумматор и нелинейную функцию активации, показаны кружочками. Выход сети обозначен как y . Для выходов скрытого слоя сети на рис. п1.1 имеем

$$v_j = f\left(\sum_{i=1}^{12} w_{i,j}^{(1)} \cdot x_i\right),$$

а для нейрона выходного слоя

$$y = y_r = f\left(\sum_{j=1}^{8} w_{j,r}^{(2)} \cdot v_j\right) = f\left(\sum_{j=1}^{8} w_{j,r}^{(2)} \cdot f\left(\sum_{i=1}^{12} w_{i,j}^{(2)} \cdot x_i\right)\right)$$

Поскольку выходной сигнал сети должен иметь аналоговую форму, чтобы он мог соответствовать сигналу исходной системы, то принимаем функции активации нейронов сети двухполярными сигмоидальными

$$z = \varphi(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

где

z - выходной аргумент (сигнал аксона) нейрона,

x - взвешенная сумма входных (синаптических) сигналов.

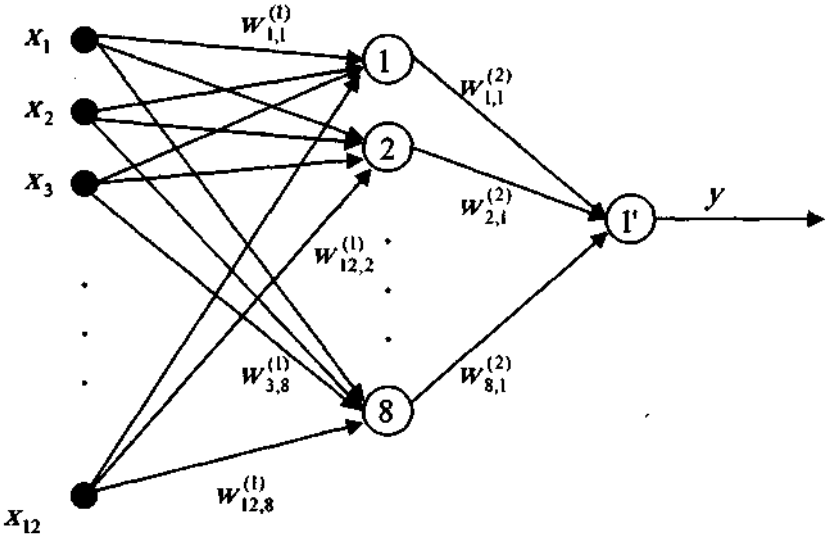


Рис. п1.1. Схема двухслойного персептрона с одним нейроном в выходном слое.

Таким образом, построение структуры нейронной сети выполнено, и нужно провести процедуру идентификации весов - обучения данной сети. Цель обучения формируется в виде квадратичной функции

$$E = \frac{1}{2}\sum_{r=1}^{l}(y_r - d_r)^2 = \frac{1}{2}(y_r - d_r)^2$$

(п1.2)

где

r - индекс выходного слоя, содержащего один нейрон,

$y_r = y$ - выход нейронной сети,

$d_r = d$ - целевой выход сети, получаемый с выхода исходной динамической системы.

Согласно [1], одним из эффективных методов обучения сетей с дифференцируемыми функциями активации нейронов, является алгоритм обратного распространения ошибки (back propagation - англ.), суть которого состоит в следующих этапах.

1. Матрицы весов сети $w_{i,j}^{(1)}$, $w_{j,r}^{(2)}$, $i=\overline{1:12}$, $j=\overline{1:8}$, $r=1$ инициализируются малыми случайными значениями, например, с распределением $N(0,0.1)$ - по нормальному закону.
2. На вход сети подается обучающий сигнал (в данном случае отсчеты входа и выхода
системы) и вычисляется выходной сигнал сети $y_r = y_l = y$.
3. Значение выходного сигнала сравнивается с целевым значением $d_r = d_l = d$ (в нашем случае с выхода динамической системы) и определяется разница
$$\Delta y = d_r - y_r = d_l - y_l = d - y$$
4. Строится система, аналогичная рис. п1.1, но с инвертированными направлениями
синапсов и производными $\frac{df(V_r^{(2)})}{dV_r^{(2)}}, \frac{df(V_j^{(1)})}{dV_j^{(1)}}$ функций активации 2-го и 1-го
слоев соответственно (см.[1]). На единственный вход системы подается разность Δy и определяются вклады весов $w_{j,r}^{(2)}$ в функцию ошибки системы
$$\frac{\partial E}{\partial w_{j,r}^{(2)}} = \Delta y \cdot \frac{df(V_r^{(2)})}{dV_r^{(2)}} \cdot v_j ,$$
(п1.3)
где $V_r^{(2)} = \sum_{j=1}^{8} w_{j,r}^{(2)} \cdot v_j$, $\frac{df(V_r^{(2)})}{dV_r^{(2)}} = \frac{d}{dV_r^{(2)}} \left(\frac{e^{V_r^{(2)}} - e^{-V_r^{(2)}}}{e^{V_r^{(2)}} + e^{-V_r^{(2)}}} \right)$,
$v_j = f\left(\sum_{i=1}^{12} w_{i,j}^{(1)} \cdot x_i \right)$ - выходы скрытого слоя исходной сети.
Аналогично для весов входного слоя имеем
$$\frac{\partial E}{\partial w_{i,j}^{(1)}} = \Delta y \cdot \frac{dy}{dv_j} \cdot \frac{dv_j}{dw_{i,j}^{(1)}} = \Delta y \cdot \frac{df(V_r^{(2)})}{dV_r^{(2)}} \cdot w_{j,r}^{(2)} \cdot \frac{df(V_j^{(1)})}{dV_j^{(1)}} \cdot x_i ,$$
(п1.4)
где $V_j^{(1)} = \sum_{j=1}^{12} w_{i,j}^{(1)} \cdot x_i$, $\frac{df(V_j^{(1)})}{dV_j^{(1)}} = \frac{d}{dV_j^{(1)}} \left(\frac{e^{V_j^{(1)}} - e^{-V_j^{(1)}}}{e^{V_j^{(1)}} + e^{-V_j^{(1)}}} \right)$.
5. Определяются матричные поправки весов сети:
$$W^{(1)}(k+1) = W^{(1)}(k) - \nabla W^{(1)}(k) \cdot \eta ,$$
(п1.5)
$$W^{(2)}(k+1) = W^{(2)}(k) - \nabla W^{(2)}(k) \cdot \eta ,$$
где $k \geq 0$ - номер итерации обучения,
$W^{(1)}(k) = \begin{bmatrix} w_{1,1}^{(1)}(k) & w_{2,1}^{(1)}(k) & \cdots & w_{12,1}^{(1)}(k) \\ w_{1,2}^{(1)}(k) & w_{2,2}^{(1)}(k) & \cdots & w_{12,2}^{(1)}(k) \\ \cdots & \cdots & \cdots & \cdots \\ w_{1,8}^{(1)}(k) & w_{2,8}^{(1)}(k) & \cdots & w_{12,8}^{(1)}(k) \end{bmatrix}$ - матрица весов «вход-скрытый слой»,
$W^{(2)}(k) = [w_{1,1}^{(2)}(k) \quad w_{2,1}^{(2)}(k) \quad w_{8,1}^{(2)}(k)]$ - вектор весов «скрытый слой-выход»,
$\nabla W^{(1)}(k) = \left[\frac{\partial E}{\partial w_{i,j}^{(1)}}(k) \right]_{i=\overline{1:12},\, j=\overline{1,8}}$ - матрица градиента функции ошибки из (п1.4),

$\nabla W^{(2)}(k) = \left[\frac{\partial E}{\partial w_{j,r}^{(2)}}(k) \right]_{r=1,\, j=\overline{1,8}}$ - матрица градиента функции ошибки из (п1.3).

6. При новых значениях обучающих данных для новой итерации $k' = k+1$:
$[u(k'-4), u(k'-3), ..., u(k'), y(k'-6), y(k'-5), ..., y(k')]$-вход сети (рис. п1.2),
$y(k'+1) = d$ - требуемый выход сети для итерации $k' = k+1$,
вычисляем разницу $\Delta y = (y-d)_{k'}$ (см. шаги 2 и 3) между требуемым и вычисленным выходом нейронной сети, и если она меньше ε (погрешность аппроксимации исходной системы нейронной сетью), заканчиваем цикл. Если нет - идем на шаг 4.

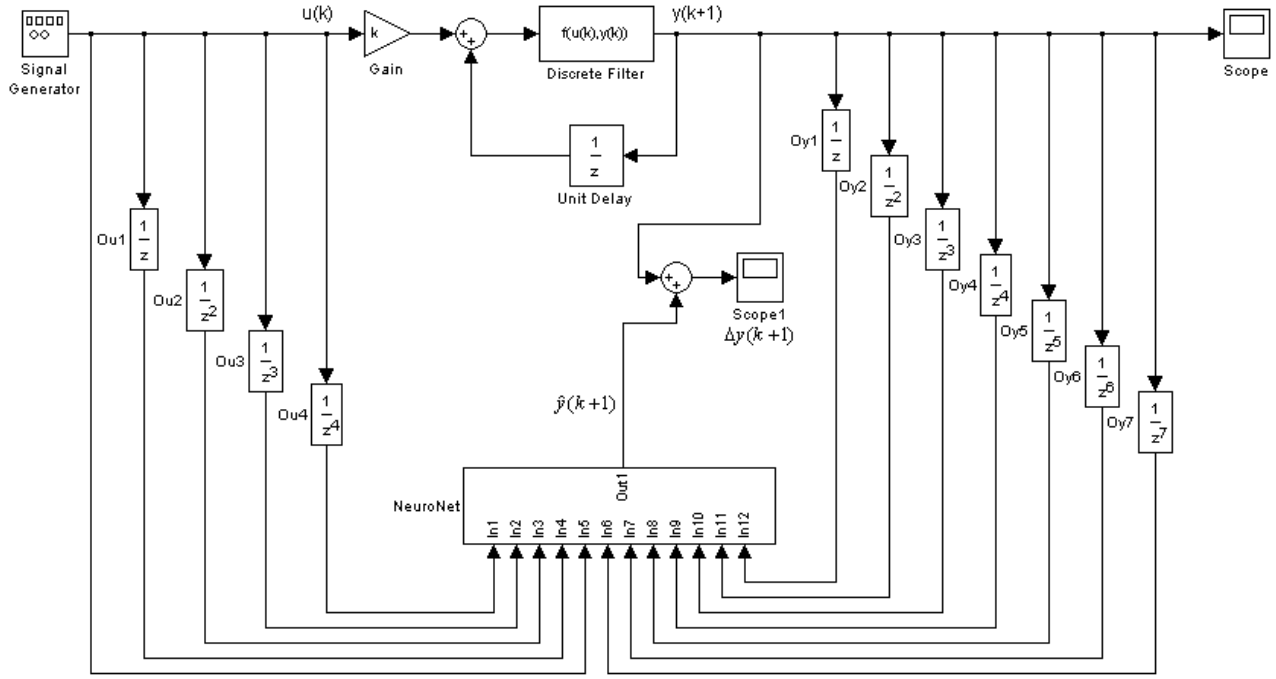


Рис. п1.2. Схема структуры динамической системы с подключенной к ней нейронной сетью.

Порядок выполнения работы

1. Используя исходные данные в табл.1, построить структурную схему динамической системы (рис. п1.2), которая представляет собой дискретный аналог апериодического звена $W(s) = 1/(Ts+1)$. Для этого сначала вывести z-передаточную функцию $\Phi(z)$ из условия соответствия значений входа и выхода дискретной и непрерывной систем в точках отсчета $t = T_s \cdot i, i \geq 0, T_s = 0.01c$ - период дискретности (рис. п1.3а, п1.3б).

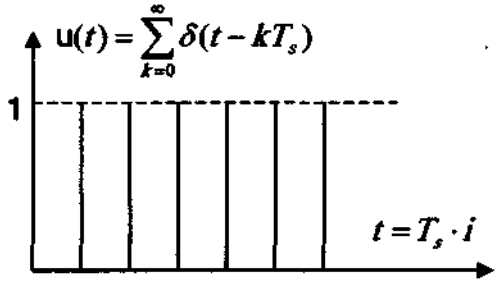


Рис. п1.3а. Вход дискретной системы.

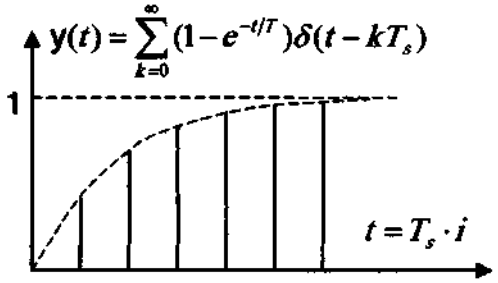


Рис. п1.3б. Выход дискретной системы.

2. По полученной дискретной передаточной функции $\Phi(z)$ и (п1.1), построить схему системы, аналогичную рис. п1.2. Промоделировать систему в Simulink при ступенчатом воздействии и убедиться в ее устойчивости.
3. Присоединить к структурной схеме п.2 нейронную сеть в виде подсистемы с 12 входами. На входы сети подать задержанные отсчеты входного и выходного сигналов (рис. п1.2). С выхода сети после оптимизации должен будет получаться следующий отсчет выхода $\hat{y}(k+1)$. Вычесть сигналы выхода исходной системы и нейронной сети для получения ошибки $\Delta y(k+1) = y(k+1) - \hat{y}(k+1)$.

4. Вычислить производные функций активации для (3) и (4) – $\frac{df(V_r^{(2)})}{dV_r^{(2)}}, \frac{df(V_j^{(1)})}{dV_j^{(1)}}$ в аналитической форме.
5. Инициализировать веса малыми случайными значениями, например

$$w_{i,j}^{(1)} \sim N(0, 0.1), w_{j,r}^{(2)} \sim N(0, 0.1),$$

где $m = 0$ - математическое ожидание, $\sigma = 0.1$ - среднеквадратичное отклонение для нормального закона.
6. Используя методику, описанную в теоретической части работы, построить нейронную сеть (рис. п1.1) и оптимизировать веса для нее, либо используя непосредственно формулы (п1.2)-(п1.5) в структурах Simulink, либо стандартные функции пакета Neural Net Toolbox [2], например функции «trainbfg», «trainbr» и др.
При этом промежуточные данные вычислений передаются в Simulink через Workspace (рабочую область) Matlab с помощью блоков «To_Workspace» и «From_Workspace».
7. Для настроенной сети построить графики входного сигнала, исходного выходного сигнала и сигнала с выхода нейронной сети, сравнить их. В отчете должны содержаться основные этапы проведения работы, структурные схемы моделирования с комментариями и графики сигналов.

Табл. 1. Исходные данные для моделирования

№ варианта	Постоянная времени T, сек.	Входной сигнал системы
1	0.2	$2\sin 3t + \cos t$
2	0.3	$2\sin t + \cos 2t$
3	0.4	$\sin 3t + \cos 2t$
4	0.5	$\sin 2t + 2\cos 3t$

Период дискретизации для всех вариантов принять $T_s = 0.01$ сек.

Список литературы:

1. С. Осовский «Нейронные сети для обработки информации», М.: «Финансы и статистика», 2002.
2. «Математические пакеты расширения Matlab. Специальный справочник», СПб.: «Питер», 2002.

П2. ЛАБОРАТОРНАЯ РАБОТА

Оптимизация двухслойной нейронной сети для моделирования дискретной нелинейной системы

Цель работы: Ознакомление с методами синтеза многослойных нейронных сетей и применение их для моделирования нелинейных дискретных систем. Освоение методики обучения нейронной сети с использованием принципа обратного распространения ошибки, а также оптимизация коэффициента обучения с использованием момента.

Общая часть

1. Введение

Нейронные сети в настоящее время получают все более широкое распространение в задачах управления и обработки информации. Способность их к обучению и высокая вычислительная эффективность благодаря возможности распараллеливания процессов обработки делают нейронные сети практически безальтернативными для решения задач с трудно формализуемыми ограничениями и целями управления. Кроме того, в силу того, что компонентами сети являются нелинейные элементы, она может свободно использоваться и для приближения нелинейных динамических систем.

В данной лабораторной работе для моделирования дискретной динамической системы используется нейронная сеть персептронного типа. Персептронные сети разделяются по топологии (однослойные, многослойные и др.), типу функции активации нейронов (релейная одно- и двухполярная, сигмоидальная и т.д.), наличию или отсутствию сигналов обратной связи и другим параметрам.

2. Теоретическая часть

В [1] показывается, что однослойный персептрон обладает весьма ограниченными вычислительными возможностями, поэтому для большинства задач его использование оказывается неприемлемым. Поэтому для решения задач моделирования и идентификации сравнительно простых систем управления целесообразно применять двухслойный персептрон, который сочетает в себе простоту оптимизации и гибкость в представлении исходной системы.

Как известно, дискретная нелинейная динамическая система со скалярными управлением и выходом может быть описана уравнениями вида

$$y(k+1) = g(f(u(k), y(k))), \qquad \text{(п2.1)}$$
$$y(0) = a = const.$$

где

$u(k)$ - сигнал управления в момент времени k ($k \geq 0$),

$y(0) = a$ - начальные условия,

$y(k)$ - сигнал выхода системы в момент времени k ($y(k) = y(t_k)$, $t_k = k \cdot T_s$),

$y(k+1)$ - сигнал выхода системы в момент ($k+1$) ($y(k+1) = y(t_{k+1})$, $t_{k+1} = (k+1) \cdot T_s$),

$f(u,y)$ - разностная модель линейного динамического звена объекта управления, которую можно записать в виде:

$$f(u(k), y(k)) = a_0 y(k) + a_1 y(k-1) + ... + a_n y(k-n) + b_0 u(k) + ... + b_m u(k-m),$$

$g(f(u,y))$ - нелинейная функция, расположенная после динамической части (рис. п2.1).

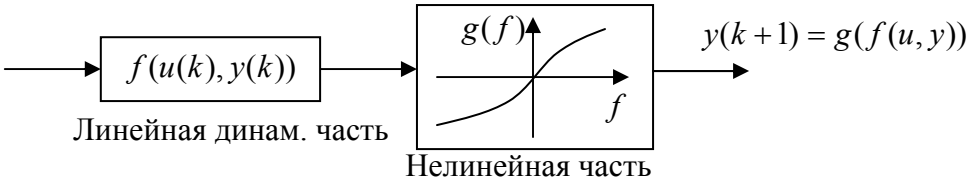


Рис. п2.1. Структура объекта управления.

Исходя из (п2.1), эквивалентная нейронная сеть может быть представлена в виде двухслойной сети с несколькими входами, представляющими p отсчетов управления $u(k), u(k-1), ..., u(k-p+1)$ и q отсчетов выхода $y(k), y(k-1), ..., y(k-q+1)$, а также одним выходом $y(k+1)$, получаемым после обработки нейронной сетью ($p+q$) входов.

Схема структуры такой сети приведена на рис. п2.2. Она включает два нейронных слоя, один из которых является скрытым (т.е. внутренним по отношению к границам сети), а другой – выходным. Входы сети обозначены как $x_1, ..., x_{12}$, веса при переходе «вход-скрытый слой» - $w_{i,j}^{(1)}$ (i-номер начальной вершины связи ($i = \overline{1:12}$), j-номер конечной вершины ($j = \overline{1:8}$), (1) – индекс слоя), а веса «скрытый слой-выходной слой» - $w_{j,r}^{(2)}$ ($r = 1$ - номер нейрона в выходном слое). Нейроны, каждый из которых включает в себя сумматор и нелинейную функцию активации, показаны кружочками. Выход сети обозначен как y. Для выходов скрытого слоя сети на рис. п2.2 имеем

$$v_j = f\left(\sum_{i=1}^{12} w_{i,j}^{(1)} \cdot x_i\right),$$

а для нейрона выходного слоя

$$y = y_r = f\left(\sum_{j=1}^{8} w_{j,r}^{(2)} \cdot v_j\right) = f\left(\sum_{j=1}^{8} w_{j,r}^{(2)} \cdot f\left(\sum_{i=1}^{12} w_{i,j}^{(1)} \cdot x_i\right)\right).$$

Поскольку выходной сигнал сети должен иметь аналоговую форму, чтобы он мог соответствовать сигналу исходной системы, то принимаем функции активации нейронов сети двухполярными сигмоидальными

$$z = \varphi(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}},$$

где

z - выходной аргумент (сигнал аксона) нейрона,

x - взвешенная сумма входных (синаптических) сигналов.

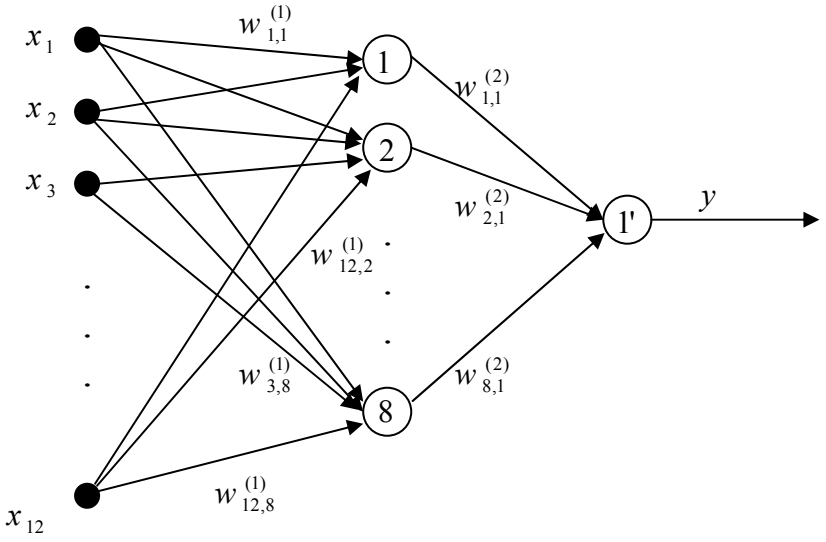


Рис. п2.2. Схема двухслойного персептрона с одним нейроном в выходном слое.

Таким образом, построение структуры нейронной сети выполнено, и нужно провести процедуру идентификации весов – обучения данной сети. Цель обучения формируется в виде квадратичной функции

$$E = \frac{1}{2}\sum_{r=1}^{1}(y_r - d_r)^2 = \frac{1}{2}(y_r - d_r)^2, \qquad (п2.2)$$

где

r - индекс выходного слоя, содержащего один нейрон,

$y_r = y$ - выход нейронной сети,

$d_r = d$ - целевой выход сети, получаемый с выхода исходной динамической системы.

Согласно [1], одним из эффективных методов обучения сетей с дифференцируемыми функциями активации нейронов, является алгоритм обратного распространения ошибки (back propagation – англ.), суть которого состоит в следующих этапах.

1. Матрицы весов сети $w_{i,j}^{(1)}$, $w_{j,r}^{(2)}$, $i = \overline{1:12}$, $j = \overline{1:8}$, $r = 1$ инициализируются малыми случайными значениями, например, с распределением $N(0, 0.1)$ - по нормальному закону.
2. На вход сети подается обучающий сигнал (в данном случае отсчеты входа и выхода системы) и вычисляется выходной сигнал сети $y_r = y_1 = y$.
3. Значение выходного сигнала сравнивается с целевым значением $d_r = d_1 = d$ (в нашем случае с выхода динамической системы) и определяется разница
$$\Delta y = d_r - y_r = d_1 - y_1 = d - y$$
4. Строится система, аналогичная рис. п2.1, но с инвертированными направлениями синапсов и производными $\frac{df(V_r^{(2)})}{dV_r^{(2)}}$, $\frac{df(V_j^{(1)})}{dV_j^{(1)}}$ функций активации 2-го и 1-го слоев соответственно (см.[1]). На единственный вход системы подается разность Δy и определяются вклады весов $w_{j,r}^{(2)}$ в функцию ошибки системы
$$\frac{\partial E}{\partial w_{j,r}^{(2)}} = \Delta y \cdot \frac{df(V_r^{(2)})}{dV_r^{(2)}} \cdot \nu_j, \qquad (п2.3)$$
где $V_r^{(2)} = \sum_{j=1}^{8} w_{j,r}^{(2)} \cdot \nu_j$, $\frac{df(V_r^{(2)})}{dV_r^{(2)}} = \frac{d}{dV_r^{(2)}}\left(\frac{e^{V_r^{(2)}} - e^{-V_r^{(2)}}}{e^{V_r^{(2)}} + e^{-V_r^{(2)}}}\right)$.

$\nu_j = f\left(\sum_{i=1}^{12} w_{i,j}^{(1)} x_i\right)$ - выходы скрытого слоя исходной сети.

Аналогично для весов входного слоя имеем
$$\frac{\partial E}{\partial w_{i,j}^{(1)}} = \Delta y \cdot \frac{dy}{d\nu_j} \cdot \frac{d\nu_j}{dw_{i,j}^{(1)}} = \Delta y \cdot \frac{df(V_r^{(2)})}{dV_r^{(2)}} \cdot w_{j,r}^{(2)} \cdot \frac{df(V_j^{(1)})}{dV_j^{(1)}} \cdot x_i, \qquad (п2.4)$$
где $V_j^{(1)} = \sum_{i=1}^{12} w_{i,j}^{(1)} x_i$, $\frac{df(V_j^{(1)})}{dV_j^{(1)}} = \frac{d}{dV_j^{(1)}}\left(\frac{e^{V_j^{(1)}} - e^{-V_j^{(1)}}}{e^{V_j^{(1)}} + e^{-V_j^{(1)}}}\right)$.
5. Определяются матричные поправки весов сети:
$$W^{(1)}(k+1) = W^{(1)}(k) - \nabla W^{(1)}(k) \cdot \eta, \qquad (п2.5)$$
$$W^{(2)}(k+1) = W^{(2)}(k) - \nabla W^{(2)}(k) \cdot \eta,$$

где $k \geq 0$ -номер итерации обучения,

$W^{(1)}(k) = \begin{bmatrix} w_{1,1}^{(1)}(k) & w_{2,1}^{(1)}(k) & ... & w_{12,1}^{(1)}(k) \\ w_{1,2}^{(1)}(k) & w_{2,2}^{(1)}(k) & ... & w_{12,2}^{(1)}(k) \\ ... & ... & ... & ... \\ w_{1,8}^{(1)}(k) & w_{2,8}^{(1)}(k) & ... & w_{12,8}^{(1)}(k) \end{bmatrix}$ - матрица весов «вход-скрытый слой»,

$W^{(2)}(k) = \begin{bmatrix} w_{1,1}^{(2)}(k) & w_{2,1}^{(2)}(k) & ... & w_{8,1}^{(2)}(k) \end{bmatrix}$ - вектор весов «скрытый слой-выход»,

$\nabla W^{(1)}(k) = \left[\frac{\partial E}{\partial w_{i,j}^{(1)}}(k) \right]_{i=\overline{1:12},\, j=\overline{1:8}}$ - матрица градиента функции ошибки из (п2.4),

$\nabla W^{(2)}(k) = \left[\frac{\partial E}{\partial w_{j,r}^{(2)}}(k) \right]_{r=1,\, j=\overline{1:8}}$ - матрица градиента функции ошибки из (3).

η =0,1…0,5 – коэффициент обучения нейросети.

6. При новых значениях обучающих данных для новой итерации $k' = k+1$:
$[u(k'-4), u(k'-3), ..., u(k'), y(k'-6), y(k'-5), ..., y(k')]$ - вход сети (рис. п2.3),
$y(k'+1) = d$ - требуемый выход сети для итерации $k' = k+1$,
вычисляем разницу $\Delta y = (y-d)_{k'}$ (см. шаги 2 и 3) между требуемым и вычисленным
выходом нейронной сети, и если она меньше ε (погрешность аппроксимации исходной системы нейронной сетью), заканчиваем цикл. Если нет – идем на шаг 4.

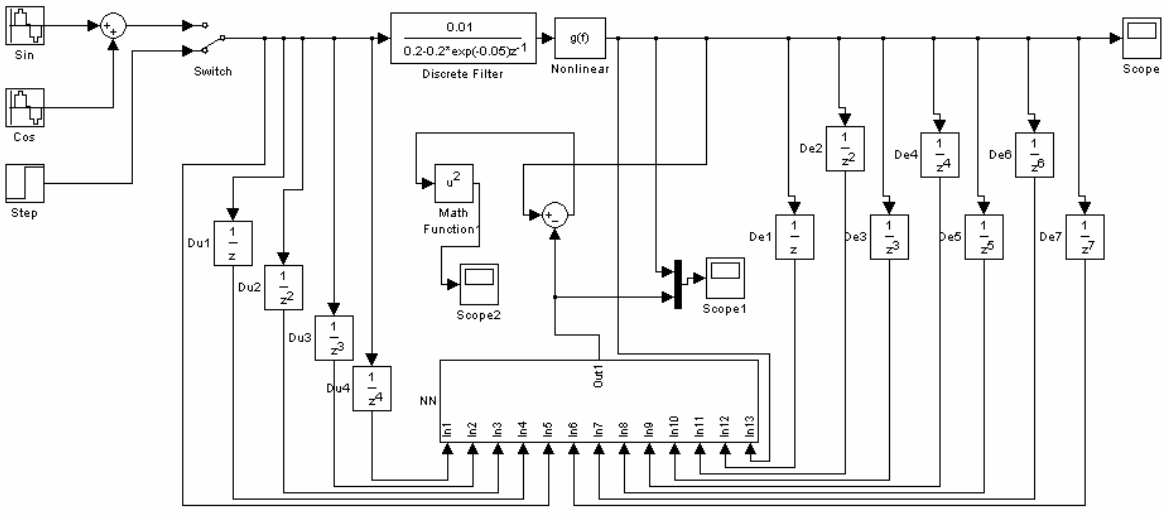


Рис. п2.3. Пример структуры нелинейной динамической системы с подключенной к ней нейронной сетью.

Адаптация коэффициента обучения сети η с применением момента.

Как видно из графиков изменения весов сети при обучении синусоидальным сигналом, по мере роста времени обучения веса начинают колебаться относительно некоторых постоянных значений. Это вызвано тем, что при приближении к оптимуму функции E (см. формулу (п2.2)) ее градиент становится небольшим, но матрицы весов $W^{(1)}(k), W^{(2)}(k)$ продолжают колебаться засчет того, что слагаемые $\eta \cdot \nabla W^{(1)}(k)$ и $\eta \cdot \nabla W^{(2)}(k)$ начинают периодически колебаться возле точки оптимума, характеризуемой матрицами $W_*^{(1)}$, $W_*^{(2)}$ оптимальных весов. Геометрически это можно интерпретировать, как градиентную оптимизацию многомерной функции при фиксированном шаге – в этом случае наступает момент, когда рабочая точка «перескакивает» то в одну, то в другую точку окрестности максимума, но из-за фиксированного шага оптимизации так и не достигает его с требуемой точностью.

Для ускорения обучения сети применяются различные методы, в том числе так называемый метод момента. Этот метод основан на модификации выражения для приращения матриц весов, т.е.

$$W^{(1)}(k+1) - W^{(1)}(k) = \Delta W^{(1)}(k) = \eta \cdot \nabla W^{(1)}(k) + \alpha \cdot (W^{(1)}(k) - W^{(1)}(k-1)),$$

(п2.6)

$$W^{(2)}(k+1) - W^{(2)}(k) = \Delta W^{(2)}(k) = \eta \cdot \nabla W^{(2)}(k) + \alpha \cdot (W^{(2)}(k) - W^{(2)}(k-1)),$$

и при начале счета

$$W^{(1)}(1) - W^{(1)}(0) = W^{(2)}(1) - W^{(2)}(0) = 0,$$

здесь $\alpha = 0,5...0,9$ - так называемый коэффициент момента,

η - коэффициент обучения,

$W^{(1)}(k), W^{(2)}(k)$ - матрицы весов на k -ом шаге оптимизации,

$W^{(1)}(k-1), W^{(2)}(k-1)$ - матрицы весов на $(k-1)$ -ом шаге оптимизации,

$W^{(1)}(k+1), W^{(2)}(k+1)$ - матрицы весов на $(k+1)$ -ом шаге оптимизации,

$\nabla W^{(1)}(k), \nabla W^{(2)}(k)$ - матрицы градиентов функции ошибки *E* из (п2.3), (п2.4).

Из (п2.6) легко видеть, что

$$(W^{(1)}(k) - W^{(1)}(k-1)) = \Delta W^{(1)}(k-1),$$

$$(W^{(2)}(k) - W^{(2)}(k-1)) = \Delta W^{(2)}(k-1),$$

поэтому (п2.6) можно представить в виде

$$\Delta W^{(1)}(k) = \eta \cdot \nabla W^{(1)}(k) + \alpha \cdot \Delta W^{(1)}(k-1),$$

$$\Delta W^{(2)}(k) = \eta \cdot \nabla W^{(2)}(k) + \alpha \cdot \Delta W^{(2)}(k-1),$$

В терминах z - преобразований сигналов можно записать:

$$Z[\Delta W^{(1)}(k)] = \eta \cdot Z[\nabla W^{(1)}(k)] + \alpha \cdot z^{-1} \cdot Z[\Delta W^{(1)}(k)] \Rightarrow Z[\Delta W^{(1)}(k)] = \frac{\eta \cdot Z[\nabla W^{(1)}(k)]}{1 - \alpha \cdot z^{-1}},$$

(п2.7)

$$Z[\Delta W^{(2)}(k)] = \eta \cdot Z[\nabla W^{(2)}(k)] + \alpha \cdot z^{-1} \cdot Z[\Delta W^{(2)}(k)] \Rightarrow Z[\Delta W^{(2)}(k)] = \frac{\eta \cdot Z[\nabla W^{(2)}(k)]}{1 - \alpha \cdot z^{-1}},$$

что соответствует уменьшению шага по мере приближения к оптимуму.

Для лучшей работы алгоритма рекомендуется полагать разность $\Delta W^{(1)}(k) = \Delta W^{(2)}(k) = 0$, если значения функции ошибки на соседних шагах $E(k+1) < 1,04 \cdot E(k)$, что позволяет проводить градиентную минимизацию вблизи оптимума.

Порядок выполнения работы

1. Используя исходные данные в табл.1, построить структурную схему нелинейной динамической системы (рис. п2.3), которая представляет собой последовательное соединение дискретного аналога $\Phi(z)$ апериодического звена $W(s)=1/(T\cdot s+1)$ и нелинейного элемента $g(f)$ (рис. п2.1). Для этого сначала вывести z-передаточную функцию $\Phi(z)$ динамической части из условия соответствия ИПФ дискретной и непрерывной систем в точках отсчета $t=T_s\cdot i$, $i\geq 0$, $T_s=0.01$с - период дискретности (рис. п2.4а, п2.4б) – либо по таблице в [1], либо по формуле:

   $\Phi(z)=[Y^*(s)]_{z=\exp(T_s\cdot s)}$, где

   $$Y^*(s)=\left[\frac{A(s)}{B'(s)}\right]_{s=s_i}\cdot\frac{1}{1-e^{-T_s(s-s_i)}},$$

   $A(s)$ -числитель непрерывной передаточной функции (ПФ) динамической части

   $W(s)$,

   $B'(s)=\dfrac{dB(s)}{ds}$ - производная знаменателя ПФ динамической части $W(s)$,

   s_i - полюс ПФ динамической части $W(s)$, т.е. корень ее знаменателя $B(s)$,

   $T_s=0.01$сек – период дискретности системы.

   Например, для апериодического звена $W(s)=1/(T\cdot s+1)$ имеем:

   $A(s)=1$,

   $B'(s)=T$,

   $s_i=-1/T$ - корень уравнения $B(s)=T\cdot s+1=0$,

   $$Y^*(s)=\frac{1}{T}\cdot\frac{1}{1-e^{-T_s s}\cdot e^{-(T_s/T)}},\ \Rightarrow \Phi(z)=\frac{1}{T(1-z^{-1}e^{-(T_s/T)})}.$$

2. Из полученной дискретной передаточной функции $\Phi(z)$ и нелинейного элемента из табл.1, построить схему системы, аналогичную рис. п2.3. Готовые нелинейные элементы уже есть в разделах Nonlinear и Math блоков Simulink. Промоделировать нелинейную систему в Simulink при ступенчатом воздействии и убедиться в ее устойчивости. Промоделировать систему для синусоидального входного сигнала из табл.1.
3. Присоединить к структурной схеме п.2 нейронную сеть в виде подсистемы с 12 входами. На входы сети подать задержанные отсчеты входного и выходного сигналов (рис. п2.3). С выхода сети после оптимизации должен будет получаться следующий отсчет выхода $\hat{y}(k+1)$. Вычесть сигналы выхода исходной системы и нейронной сети для получения ошибки $\Delta y(k+1)=y(k+1)-\hat{y}(k+1)$. Кроме того, на рис. п2.3 показан 13-й входной сигнал нейронной сети, служащий для формирования разницы $y(k+1)-\hat{y}(k+1)=d_{k+1}-\hat{y}(k+1)$ выхода исходной системы и выхода нейронной сети.
4. Вычислить производные функций активации для (3) и (4) - $\dfrac{df(V_r^{(2)})}{dV_r^{(2)}}, \dfrac{df(V_j^{(1)})}{dV_j^{(1)}}$ в аналитической форме.
5. Инициализировать веса малыми случайными значениями, например

   $w_{i,j}^{(1)}\sim N(0,0.1),\quad w_{j,r}^{(2)}\sim N(0,0.1)$,

где $m = 0$ – математическое ожидание, $\sigma = 0.1$ - среднеквадратичное отклонение для нормального закона.

6. Используя методику, описанную в теоретической части работы, построить нейронную сеть (рис. п2.2) и оптимизировать веса для нее, либо используя непосредственно формулы (п2.2)-(п2.5) в структурах Simulink, либо стандартные функции пакета Neural Net Toolbox [2], например функции “trainbfg”, “trainbr” и др.
   При этом промежуточные данные вычислений передаются в Simulink через Workspace (рабочую область) Matlab с помощью блоков “To_Workspace” и “From_Workspace”.
7. Для настроенной сети построить графики входного сигнала, исходного выходного сигнала и сигнала с выхода нейронной сети, построить график квадрата разности выходных сигналов нелинейной системы и нейронной сети и сравнить их.
8. Реализовать адаптацию шага обучения для ускорения обучения сети с использованием формул (7). Значение коэффициента момента взять $\alpha = 0,7$.
9. В отчете должны содержаться основные этапы проведения работы, структурные схемы моделирования с комментариями и графики сигналов.

Табл.1. Исходные данные для моделирования.

№ варианта	Постоянная времени T, сек	Нелинейная функция в системе	Входной сигнал системы
1	0.2	$g(x) = 0,3x^3$	$0,2\sin 3t + 0,3\cos t$
2	0.3	$g(x) = \begin{cases} (x+0,2), & x < -0,2 \\ 0, & -0,2 \le x \le 0,2 \\ (x-0,2), & x > 0,2 \end{cases}$	$0,5\sin t + 0,6\cos 2t$
3	0.4	$g(x) = \begin{cases} -0.3, & x < -0,3 \\ x, & -0,3 \le x \le 0,3 \\ 0,3, & x > 0,3 \end{cases}$	$0,1\sin 3t + 0,4\cos 2t$
4	0.5	$g(x) = \begin{cases} 0,5(x+0,4), & x > 0 \\ 0, & x = 0 \\ 0,5(x-0,4), & x < 0 \end{cases}$	$0,3\sin 2t + 0,5\cos 3t$

Примечание: для нелинейностей в Simulink есть готовые блоки.

Период дискретности для всех вариантов принять $T_s = 0.01$ сек.

Список литературы:

1. В.А. Бесекерский "Цифровые автоматические системы", М: "Наука", 1976.
2. С. Осовский "Нейронные сети для обработки информации", М: "Финансы и статистика", 2002.
3. "Пакеты расширения Matlab. Специальный справочник", С.-Пб. "Питер", 2002.

ПЗ. ЛАБОРАТОРНАЯ РАБОТА

Решение задачи коммивояжера с использованием нейронной сети Хопфилда.

Цель работы: Ознакомление с методами синтеза нейронных сетей Хопфилда и применение их для решения задачи коммивояжера. Освоение методики обучения нейронной сети с обратными связями.

Общая часть

1. Введение

Задача коммивояжера является оптимизационной задачей, часто возникающей на практике. Она может быть сформулирована следующим образом: для некоторой группы городов с заданными расстояниями между ними требуется найти кратчайший маршрут с посещением каждого города один раз и с возвращением в исходную точку. Было доказано, что эта задача принадлежит большому множеству задач, называемых «NP-полными» (недетерминистски полиномиальными). Для NP-полных задач не известно лучшего метода решения, чем полный перебор всех возможных вариантов. Так как с ростом размерности задачи число этих вариантов растет экспоненциально, то метод полного перебора становится неэффективным из-за больших затрат времени.

В данной лабораторной работе для решения задачи коммивояжера используется нейронная сеть Хопфилда.

2. Теоретическая часть

Характерной особенностью сетей Хопфилда является наличие обратных связей, т. е. связей, идущих от выходов сетей и их входам. Отсутствие обратной связи гарантирует безусловную устойчивость сетей. Они не могут войти в режим, когда выход беспрерывно блуждает от состояния к состоянию и не пригоден к использованию. Но это весьма желательное свойство достигается не бесплатно: сети без обратных связей обладают более ограниченными возможностями по сравнению с сетями с обратными связями.

Так как сети с обратными связями имеют пути, передающие сигналы от выходов к входам, то отклик таких сетей является динамическим, т. е. после приложения нового входа вычисляется выход и, передаваясь по сети обратной связи, модифицирует вход. Затем выход повторно вычисляется, и процесс повторяется снова и снова. Для устойчивой сети последовательные итерации приводят к все меньшим изменениям выхода, пока в конце концов выход не становится постоянным.

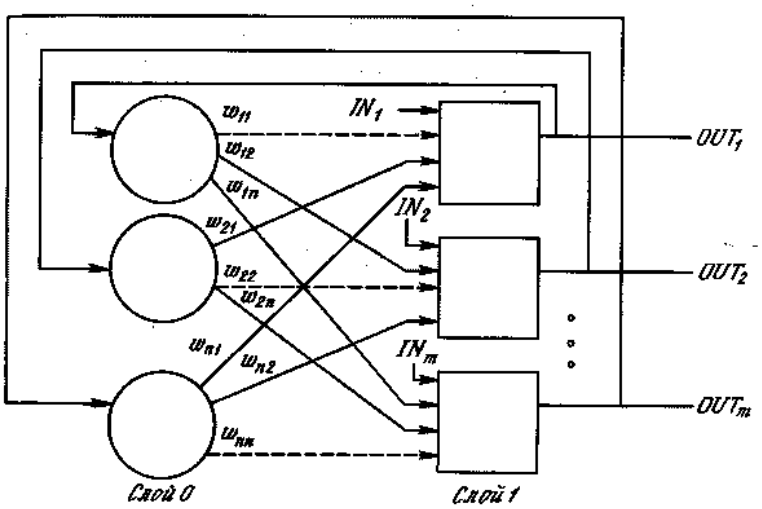


Рис. п3.1. Однослойная сеть с обратными связями (пунктирные линии обозначают нулевые веса)

На рис. п3.1 показана сеть с обратными связями, состоящая из двух слоев. Способ представления несколько отличается от использованного в работе Хопфилда и других, но эквивалентен им с функциональной точки зрения. Нулевой слой не выполняет вычислительной функции, а лишь распределяет выходы сети обратно на входы. Каждый нейрон первого слоя вычисляет взвешенную сумму своих входов, давая сигнал NET, который затем с помощью нелинейной функции F преобразуется в сигнал OUT. Эти операции сходны с нейронами других сетей.

В первой работе Хопфилда функция F была просто пороговой функцией. Выход такого нейрона равен единице, если взвешенная сумма выходов с других нейронов больше порога T_j, в противном случае она равна нулю. Он вычисляется следующим образом:

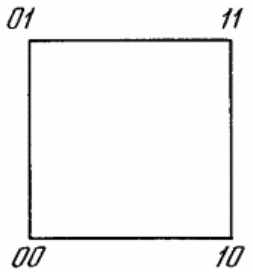
$$NET_j = \sum_{i \neq j} w_{ij} OUT_i + IN_j ,$$

(п3.1)

где $OUT_j = \begin{cases} 1, NET_j \geq T_j \\ 0, NET_j < T_j \end{cases}$

Состояние сети – это просто множество текущих значений сигналов OUT от всех нейронов. В первоначальной сети Хопфилда состояние каждого нейрона менялось в дискретные случайные моменты времени, в последующей работе состояния нейронов могли меняться одновременно. Так как выходом бинарного нейрона может быть только ноль или единица (промежуточных уровней нет), то текущее состояние сети является двоичным числом, каждый бит которого является сигналом OUT некоторого нейрона.

Функционирование сети легко визуализируется геометрически. На рис. п3.2а показан случай двух нейронов в выходном слое, причем каждой вершине квадрата соответствует одно из четырех состояний системы (00, 01, 10, 11). На рис. п3.2б показана трехнейронная система, представленная кубом (в трехмерном пространстве), имеющим восемь вершин, каждая из которых помечена трехбитовым бинарным числом. В общем случае система с n нейронами имеет 2^n различных состояний и представляется n-мерным гиперкубом.



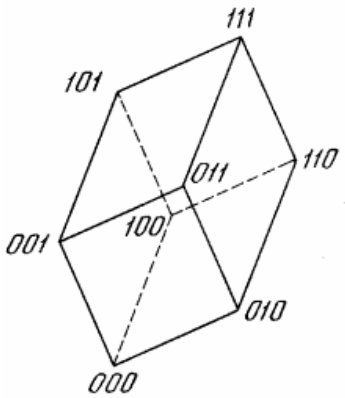


Рис. п3.2а. Два нейрона порождают систему с четырьмя состояниями
Рис. п3.2б. Три нейрона порождают систему с восемью состояниями

Когда подается новый входной вектор, сеть переходит из вершины в вершину, пока не стабилизируется. Устойчивая вершина определяется сетевыми весами, текущими входами и величиной порога. Если входной вектор частично неправилен или неполон, то сеть стабилизируется в вершине, ближайшей к желаемой.

Как и в других сетях, веса между слоями в этой сети могут рассматриваться в виде матрицы *W*. Сеть с обратными связями является устойчивой, если ее матрица симметрична и имеет нули на главной диагонали, т. е. если $w_{ij} = w_{ji}$ и $w_{ii} = 0$ для всех значений *i*.

В общем случае активационная функция *F*, моделирующая биологический нейрон, представляет собой сигмоидальную (логистическую) функцию

$$F(x) = \frac{l}{l + \exp(-\lambda NET)},$$

(п3.2)

где *l* – коэффициент, определяющий крутизну сигмоидальной функции. Если *l* велико, *F* приближается к описанной ранее пороговой функции. Небольшие значения *l* дают более пологий наклон.

Как и для бинарных систем, устойчивость гарантируется, если веса симметричны, т. е. $w_{ij} = w_{ji}$ и $w_{ii} = 0$ при всех *i*.

Способность сети быстро производить вычисления является ее главным достоинством. Она обусловлена высокой степенью распараллеливания вычислительного процесса. Если сеть реализована на аналоговой электронике, то решение редко занимает промежуток времени, больший нескольких постоянных времени сети. Более того, время сходимости слабо зависит от размерности задачи. Это резко контрастирует с более чем экспоненциальным ростом времени решения при использовании обычных подходов. Моделирование с помощью однопроцессорных систем не позволяет использовать преимущества параллельной архитектуры, но современные мультипроцессорные системы весьма многообещающи для решения трудных задач.

Задача коммивояжера

Допустим, что города, которые необходимо посетить, помечены буквами A, B, C и D, а расстояния между парами городов есть d_{ab}, d_{bc} и т. д. Решением является упорядоченное множество из *n* городов. Задача состоит в отображении его в вычислительную сеть с использованием нейронов в режиме с большой крутизной характеристики (*l* приближается к бесконечности). Каждый город представлен строкой

из n нейронов. Выход одного и только одного нейрона из них равен единице (все остальные равны нулю). Этот равный единице выход нейрона показывает порядковый номер, в котором данный город посещается при обходе. В табл.1 показан случай, когда город C посещается первым, город A – вторым, город D – третьим и город B – четвертым. Для такого представления требуется n^2 нейронов – число, которое быстро растет с увеличением числа городов. Длина такого маршрута была бы равна $d_{ca} + d_{ad} + d_{db} + d_{bc}$. Так как каждый город посещается только один раз и в каждый момент посещается лишь один город, то в каждой строке и в каждом столбце имеется по одной единице.

Для решения этой задачи с помощью нейронной сети Хопфилда нужно закодировать маршрут активностью нейронов и так подобрать связи между ними, чтобы энергия сети оказалась связанной с полной длиной маршрута. Хопфилд и Танк предложили для этого следующий способ.

Каждый нейрон снабжен двумя индексами, которые соответствуют городу и порядковому номеру его посещения в маршруте. Например, $OUT_{xj} = 1$ показывает, что город x был j-ым по порядку городом маршрута.

Функция энергии должна удовлетворять двум требованиям: во-первых, должна быть малой только для тех решений, которые имеют по одной единице в каждой строке и в каждом столбце; во-вторых, должна оказывать предпочтение решениям с короткой длиной маршрута.

Первое требование удовлетворяется введением следующей, состоящей из трех сумм, функции энергии:

$$E = \frac{A}{2}\sum_x\sum_i\sum_{j\neq i} OUT_{xi}OUT_{xj} + \frac{B}{2}\sum_i\sum_x\sum_{y\neq x} OUT_{xi}OUT_{yi} + \frac{C}{2}\left[\left(\sum_x\sum_i OUT_{xi}\right) - n\right]^2,$$

(п3.3)

где A, B и C – некоторые константы. Этим достигается выполнение следующих условий:

1. Первая тройная сумма равна нулю в том и только в том случае, если каждая строка (город) содержит не более одной единицы.
2. Вторая тройная сумма равна нулю в том и только в том случае, если каждый столбец (порядковый номер посещения) содержит не более одной единицы.
3. Третья сумма равна нулю в том и только в том случае, если матрица содержит ровно n единиц.

Город	Порядок следования			
	1	2	3	4
A	0	1	0	0
B	0	0	0	1
C	1	0	0	0
D	0	0	1	0

Табл.1. Маршрут коммивояжера

Второе требование – предпочтение коротким маршрутам – удовлетворяется с помощью добавления следующего члена к функции энергии:

$$E = \frac{D}{2}\sum_x\sum_{y\neq x}\sum_i d_{xy}OUT_{xi}(OUT_{y,i+1} + OUT_{y,i-1})$$

(п3.4)

Заметим, что этот член представляет собой длину любого допустимого маршрута. Для удобства индексы определяются по модулю *n*, т. е. $OUT_{n+j} = OUT_j$, а *D* – некоторая константа.

При достаточно больших значениях A, B и C низкоэнергетические состояния будут представлять допустимые маршруты, а большие значения D гарантируют, что будет найден короткий маршрут.

Теперь зададим значения весов, т. е. установим соответствие между членами в функции энергии и членами общей формы. Получаем:

$$w_{xi,yi} = \begin{cases} -A\delta_{xy}(1-\delta_{ij}) \ (\textit{не допускает более одной единицы в строке}) \\ -B\delta_{ij}(1-\delta_{xy}) \ (\textit{не допускает более одной единицы в столбце}) \\ -C \ (\textit{глобальное ограничение}) \\ -Dd_{xy}(\delta_{j,i+1}+\delta_{j,i-1}) \ (\textit{член, отвечающий за длину цикла}) \end{cases}$$

Итак,

$$w_{xi,yi} = -A\delta_{xy}(1-\delta_{ij}) - B\delta_{ij}(1-\delta_{xy}) - C - Dd_{xy}(\delta_{j,i+1}+\delta_{j,i-1}) \qquad \text{(п3.5)}$$

Здесь $\delta_{ij} = \begin{cases} 1, i = j \\ 0, \text{èí à÷å} \end{cases}$; каждый нейрон имеет смещающий вес x_i, соединенный с +1 и равный *Cn*.

Практическая часть

Необходимо создать и обучить нейронную сеть Хопфилда путем написания собственной программы с использованием возможностей пакета Simulink. Алгоритм решения задачи имеет следующий вид:

Шаг 1. Задать значения A,B,C,D, используемые для расчета весов в формуле (п3.5). Задать координаты городов и вычислить расстояния между ними. Выбрать тип нейронов (функцию активации);
Шаг 2. Произвести расчет весов по формуле (п3.5);
Шаг 3. На входы сети подается некоторый сигнал. Фактически его ввод осуществляется непосредственной установкой значений аксонов OUT_i, $i = \overline{1,n}$, где *n* – число нейронов;
Шаг 4. Рассчитать новые состояния нейронов по формуле (п3.1) и новые значения аксонов согласно выбранной функции активационной функции;
Шаг 5. Проверить, изменились ли выходные значения аксонов за последние *k* итераций (*k* задано и является критерием выхода из цикла). Если да – перейти к шагу 4;
Шаг 6. Проверить правильность маршрута. Если маршрут неверный, перейти к шагу 3.

В итоге необходимо получить таблицу, сходную с таблицей 1 теоретической части. Эта таблица и будет представлять собой решение задачи.

Существует несколько разновидностей формулировки задачи коммивояжера, которые несколько видоизменяют алгоритм решения. Приведем три из них:
Кратчайший полный цикл. В этой постановке требование посещения города «точно однажды» заменяется на «хотя бы однажды»;
Трубопроводная задача. В графе зафиксировано некоторое множество ребер (труб). Необходимо построить маршрут коммивояжера, обязательно проходящий через все трубы.

Задача со сроками. Пусть Δt_{ij} - матрица переездов между городами, заданы также время нахождения в каждом городе и начальный город маршрута. Для каждого города известен предельный срок выезда. Необходимо построить маршрут коммивояжера, не нарушая ограничения на сроки.

Исходные данные:

№ варианта	Тип формулировки	Число городов	Тип функции активации	Число труб	Матрица переездов
1	Классическая	5	Пороговая	-	-
2	Классическая	6	Сигмоидальная	-	-
3	Классическая	7	Пороговая	-	-
4	Классическая	8	Сигмоидальная	-	-
5	Кратчайший полный цикл	6	Пороговая	-	-
6	Трубопроводная задача	6	Пороговая	3	-
7	Задача со сроками	5	Сигмоидальная	-	$\begin{pmatrix} \ddots & 0.2 & 0.3 & 0.1 & 0.5 \\ 0.2 & \ddots & 0.7 & 0.6 & 0.3 \\ 0.3 & 0.7 & \ddots & 0.4 & 0.1 \\ 0.1 & 0.6 & 0.4 & \ddots & 0.9 \\ 0.5 & 0.3 & 0.1 & 0.9 & \ddots \end{pmatrix}$

Пример программы, моделирующей работу нейронной сети Хопфилда с пороговой функцией активации нейронов для решения задачи коммивояжера

```
nCities = 6;   % число городов
threshold = 20;   % порог срабатывания нейрона
nStops = nCities;   % число остановок, равное числу городов;
nInputs = nCities * nStops;   % число входов нейросети (число нейронов)
inhibWeight = -2;   % подавляющий вес
inputWidth = nCities;   % для вывода на экран
initialActivations = ones(nInputs,1) * -1;
activations = initialActivations;   % состояние нейронов

% Присвоение координат x,y каждому городу
cityLocations = zeros(nCities,2);
cityLocations(1,:) = [0 3];
cityLocations(2,:) = [1 5];
cityLocations(3,:) = [4 5];
cityLocations(4,:) = [5 2];
cityLocations(5,:) = [4 0];
cityLocations(6,:) = [1 0];

% Вычисление расстояний между городами;
distances = zeros(nCities,nCities);
for city1 = 1:nCities,   % city - порядковый номер города (его имя)
  for city2 = 1:nCities,
    difference = cityLocations(city1,:) - cityLocations(city2,:);
    distances(city1,city2) = sqrt(difference(1,1)  * difference(1,1) + ...
                        difference(1,2)  * difference(1,2));
  end
end

% Масштабирование расстояний: максимальное расстояние равно 1
distances = distances / max(max(abs(distances)));

% Вычисление весов
```

```
weights = zeros(nInputs,nInputs);
for city1 = 1:nCities,
  for stop1 = 1:nStops,   % stop - порядковый номер следования города
    for city2 = 1:nCities,
      for stop2 = 1:nStops,
        % Ограничение 1: Необходимо, чтобы один город не посещался более одного раза
        % В случае нарушения ограничения (см. ниже) вводится большой подавляющий вес
        if ((city1 == city2) & (stop1 ~= stop2))
          weights = setWeight(city1,stop1,city2,stop2, ...
                inhibWeight,weights,nCities);
        % Ограничение 2: Необходимо, чтобы одному порядковому номеру
        % следования не соответствовало более одного города
        % В случае нарушения ограничения (см. ниже) вводится большой подавляющий вес
        elseif ((stop1 == stop2) & (city1 ~= city2))
          weights = setWeight(city1,stop1,city2,stop2, ...
                inhibWeight,weights,nCities);
        % Ограничение 3: Для того, чтобы маршрут был кратчайшим, необходимо, чтобы
        % между городами, имеющими соседние порядковые номера следования, были
        % минимальные расстояния
        % Если не нарушено ограничение 1 и города следуют в маршруте друг
        % за другом (что означает и выполнение ограничения 2) (см. ниже), то весу
        % связи нейрона city1-stop1 с нейроном city2-stop2 присваивается
        % значение, равное расстоянию между этими городами, взятому с
        % противоположным знаком и умноженному на некоторый коэффициент
       elseif (city1 ~= city2) & (abs(stop1 - stop2) == 1)
          weights = setWeight(city1,stop1,city2,stop2, ...
                -0.5*distances(city1,city2),weights,nCities);
       end
      end
    end
  end
end

% Использование нейросети
somebodyChanged=1;   % булева функция: 1, если на очередной итерации вес изменился
notChanged=0;   % количество итераций подряд, на которых ни один вес не изменился
notChangedLim=1000;   % критерий выхода
k=0
while (notChanged<notChangedLim)
  k=k+1;
  order = randperm(nInputs)';   % задание случайной последовательности нейронов
  somebodyChanged = 0;
  for i = 1:nInputs,
      unit = order(i,1);
      oldActivation = activations(unit,1);
      netIn = weights(unit,:) * activations;
      if netIn >= threshold
        activations(unit,1) = 1;
      else
       activations(unit,1) = -1;
      end
      if oldActivation ~= activations(unit,1)
        somebodyChanged = 1;
      end
  end
  if (somebodyChanged == 0)
      notChanged=notChanged+1;
  else
      notChanged=1;
  end
end

% Графическое представление состояния нейронов
n = inputWidth;
activationGrid = zeros(n+1,n+1);
activationGrid(1:n,1:n) = reshape(activations,n,n);
figure;
colormap([1 1 1 ; 0 0 0]);   % черный квадрат - +1, белый - -1
pcolor(activationGrid);
xlabel('Порядок следования')
```

```
ylabel('Город')
set(gca,'XTick',[1:nStops]);
set(gca,'YTick',[1:nCities]);

% Графическое представление маршрута коммивояжера
figure;
clf; hold on;
maxX = max(cityLocations(:,1)) + 1;
maxY = max(cityLocations(:,2)) + 1;
minX = min(cityLocations(:,1)) - 1;
minY = min(cityLocations(:,2)) - 1;
plot(cityLocations(:,1),cityLocations(:,2),'ok');
axis([minX maxX minY maxY]);
tour = zeros(nCities+1,2);
stopNum = 1;
for stop = 1:nStops,
  for city = 1:nCities,
    if activationGrid(city,stop) == 1
      tour(stopNum,:) = cityLocations(city,:);
      stopNum = stopNum + 1;
    end
  end
end
tour(nCities+1,:)=tour(1,:);
plot(tour(:,1),tour(:,2),'k');

for city = 1:nCities,
  cityName = sprintf('Город %d',city);
  text(cityLocations(city,1)+0.2,cityLocations(city,2)+0.2,cityName);
end
```

Список литературы:

1. А.В.Назаров, А.И.Лоскутов «Нейросетевые алгоритмы прогнозирования и оптимизации систем». Наука и техника, С.-Петербург, 2003, стр.228-248, 354-356.
2. С.Осовский «Нейронные сети для обработки информации». М., Финансы и статистика, 2002, стр.178-184.
3. В.И.Комашинский, Д.А.Смирнов «Нейронные сети и их применение в системах управления и связи». М., Горячая линия – Телеком, 2003.

П4. ЛАБОРАТОРНАЯ РАБОТА

Решение четкой системы линейных алгебраических уравнений (СЛАУ) нейросетевым методом.

Цель работы: Ознакомление с методом решения четкой СЛАУ нейросетевым способом.

Общая часть

1. Введение

При решении задач управления в условиях неопределенности эти задачи часто эквивалентны задачам поиска решения СЛАУ. Например, решение дифференциальных и интегральных уравнений после их дискретизации, проектирование систем управления и прогнозирование, моделирование поведения линейных и нелинейных объектов и т.д. Для решения СЛАУ существуют различные методы и алгоритмы. Эти методы имеют различную эффективность и оказывают значительное влияние на процесс обработки данных, а следовательно и на производительность вычислительных систем, которые используются в управлении.

2. Теоретическая часть

Имеем СЛАУ относительно вектора $X^T=(x_1,\ldots,x_n)$:

$$\begin{cases} a_{11}\cdot x_1+\cdots+a_{1n}\cdot x_n=b_1 \\ \vdots \\ a_{m1}\cdot x_1+\cdots+a_{mn}\cdot x_n=b_m, \end{cases}$$

(п4.1)

где a_{ij}, $(i=\overline{1,m},\ j=\overline{1,n})$, b_j - заданные четкие числа, т.е. числа с функцией принадлежностей типа «синглтон». Полагается, что $m>n$, т.е. СЛАУ (п4.1) переопределена.

В матричной форме (п4.1) записывается в виде:

$A\cdot X=b$,

где A – заданная матрица размера $dimA=(m\times n)$ с четкими элементами

$$A=\begin{pmatrix} a_{11}\ldots a_{1n} \\ \vdots \\ a_{mn}\ldots a_{mn} \end{pmatrix}_{(m\times n)},$$

(п4.2)

X – вектор неизвестных переменных – $X^T=(x_1,\ldots,x_n)$;

b – заданный вектор – $b^T=(b_1,\ldots,b_m)$; «Т» - символ транспонирования.

Для решения (2) вводим целевую функцию:

$$\xi(x)=\sum_{i=1}^{m} y(x_i)^2,$$

которая в векторной форме имеет вид:

$$\xi=y^T\cdot y,$$

где $y^T=(y(x_1),\ldots,y(x_m))$ $_{(1xm)}$– вектор «невязки» решения СЛАУ.

Ищем вектор X^T из условия минимизации целевой функции (квадрата «невязки»):

$$\min_X \xi(x)=\min_X y\cdot y^T$$

(п4.3)

Для решения (3) используем итерационный градиентный метод:

$$X(k+1) = X(k) - \eta \nabla \left[\xi(X(k)) \right],$$

(п4.4)

где k – номер итерации, η - коэффициент стабилизации, $\nabla[\cdot]$ - градиент целевой функции.

Найдем $\nabla[\cdot]$:

$$\nabla \xi(X) = \frac{\partial}{\partial x}\left(y^T \cdot y \right) = \frac{\partial}{\partial x}\left[(AX-b)^T (AX-b) \right] = A^T (AX-b) + A^T (AX-b) = 2A^T (AX-b)$$

,

в результате из (п4.4) получим:

$$X(k+1) = X(k) - 2\eta A^T (AX(k) - b), \; k=0;\; 1;\ldots$$

Алгоритм решения четкой СЛАУ состоит из следующих шагов:

1. Задается начальное приближение, например, $x_0=1$;
2. Функция активации «f» нейросети задается в виде линейной зависимости:
$$Y = f(AX-b) = f(X) = X \; ;$$
3. Целевая функция ξ нейросети задается в виде:
$$\xi(X) = Y^T \cdot Y \, .$$
4. Окончание итерационного процесса имеет вид:
$$\left\| \xi(X(k+1)) - \xi(X(k)) \right\| \le \varepsilon \, ,$$

где ε - заданное число.

Общая схема решения четкой СЛАУ в нейросетевом логическом базисе приведена на рис. п4.1. Структура нейросети решения четкой СЛАУ приведена на рис. п4.2.

3. Практическая часть

Рассмотреть надежность технических средств систем управления (СУ) при холодном резервировании основного комплекса одним резервным блоком с его восстановлением посредством ремонта (рис. п4.3).

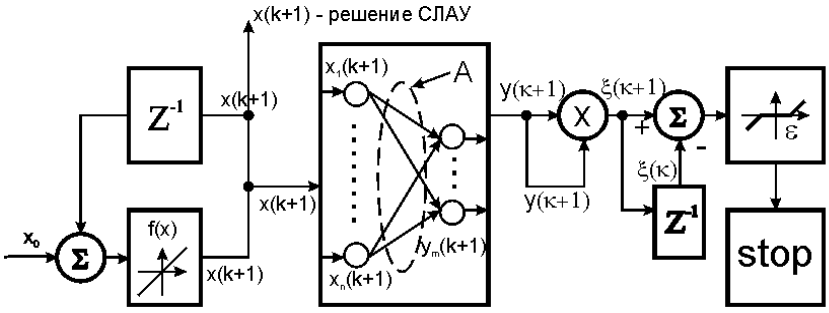


Рис. п4.1. Блок схема решения чёткой СЛАУ (z^{-1} – элемент задержки).

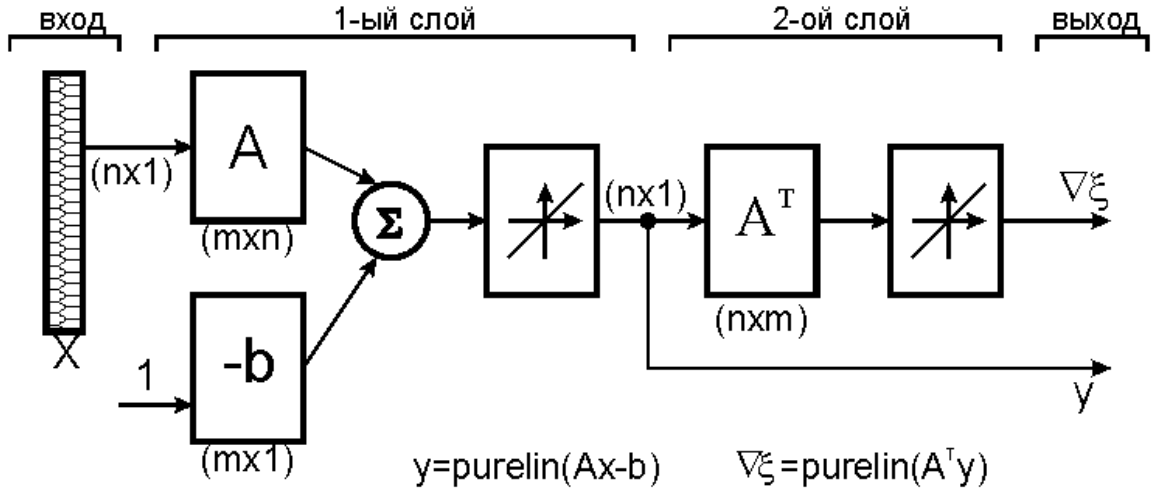


Рис. п4.2. Нейросеть решения чёткой СЛАУ в Matlab/Neural Network

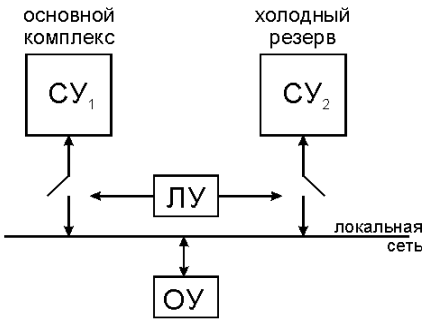


Рис. п4.3. Блок-схема технических средств системы управления (ЛУ – логическое устройство; ОУ – объект управления).

3.1. Составить граф состояний для СУ (рис. п4.3) при четком потоке $\lambda(t)$ отказов и четком потоке $\mu(t)$ восстановлений, и для графа состояний записать четкие дифференциальные уравнения Колмогорова.
3.2.Для $t \to \infty$ записать четкую СЛАУ для нахождения нечеткого коэффициента готовности $\hat{E}_{\tilde{a}}$ технических средств.
3.3. Задать $\lambda(t) = \lambda - const$, $\mu(t) = \mu - const$ и для этих значений найти $\hat{E}_{\tilde{a}}$ нейросетевым методом.
3.4. Найти $\hat{E}_{\tilde{a}}$ методами теории автоматического регулирования, используя язык блок-схем, в системе MatLab/Simulink.
3.5. Результаты вычислений $K_г$ по п.п. 3.3, 3.4 представить в виде таблицы.

	λ=…	μ=…	*Нейросетевой метод*	*Simulink*
$K_г$				

4. Контрольные вопросы
4.1. Перечислить различные типы функций активации искусственного нейрона.
4.2. Записать четкую СЛАУ при нахождении $K_г$ для двух блоков холодного резервирования основной СУ.
4.3. Сформулировать физическую интерпретацию λ^{-1}, μ^{-1}.
4.4 При каких предположениях относительно знаков распределения потоков отказов и восстановлений справедливы уравнения Колмогорова?

Список литературы:

1. Мочалов И.А. Нечеткие вероятностно-статистические методы в задачах управления. Электронная версия на CD, 2004.
2. Медведев В.С., Потемкин В.Г. Нейронные сети. Matlab 6. М., Диалог МИФИ, 2002.
3. Комашинский В.И., Смирнов Д.А. Нейронные сети и их применение в системах управления и связи. М., Горячая линия – Телеком, 2003.
4. Ярушкина Н.Г. Основы теории нечетких и гибридных систем. М., Финансы и статистика, 2004.
5. Каллан Р. Основные концепции нейронных сетей. М., «Вильямс», 2003.

П5. ЛАБОРАТОРНАЯ РАБОТА

Решение нечеткой системы линейных алгебраических уравнений (СЛАУ) нейросетевыми методами.

Цель работы: Ознакомление с методом решения нечеткой СЛАУ нейросетевым способом.

Общая часть

1. Введение

В настоящее время в теории управления для описания неопределенности широко используются частотные методы, методы теории вероятностей и математической статистики, в частности, теория случайных процессов. В последнее время разработан другой способ описания неопределенности с использованием теории нечетких множеств. В стадии разработки находится комбинированный подход описания неопределенности с помощью нечетких случайных процессов.

Цель работы состоит в ознакомлении нечеткого метода решения СЛАУ с использованием нейросетевых технологий. Эти технологии оказывают значительное влияние на процесс обработки данных и принятие управленческих решений. Процедура решения нечеткой СЛАУ рассматривается на примере расчета коэффициента готовности $К_г$ технических средств системы управления (СУ) при горячем резервировании одним блоком с его восстановлением посредством ремонта.

2. Теоретическая часть

Имеем нечеткую СЛАУ относительно вектора $X^T=(x_1,\dots,x_n)$:

$$\begin{cases} r_{a_{11}}(x)\cdot x_1+\dots+r_{a_{1n}}(x)\cdot x_n = r_{b_1}(x) \\ \vdots \\ r_{a_{m1}}(x)\cdot x_1+\dots+r_{a_{mn}}(x)\cdot x_n = r_{b_m}(x), \end{cases}$$

(п5.1)

где $r_{a_{ij}}(x)$, $r_{b_i}(x)$, $(i=\overline{1,m}, j=\overline{1,n})$ - заданные функции принадлежностей нечетких переменных a_{ij}, b_i. Полагается, что $m>n$, т.е. нечеткая СЛАУ (п5.1) переопределена.

В этих условиях необходимо найти функцию принадлежностей $r_{x_j}(x)$, которая определяет нечеткой решение (п5.1). Эту задачу будем решать методом модификации исходной задачи к совокупности четких задач.

Пусть для простоты $r_{a_{ij}}(x)$, $r_{b_j}(x)$ имеют треугольные симметричные относительно $core\, r_{a_{ij}}(x)$, $core\, r_{b_j}(x)$ формы (рис. п5.1а, п5.1б). найдем для них обратные отображения $a_{ij}(r)$, $b_j(r)$, где $a_{ij}(r)=(\underline{a}_{ij}(r),\overline{a}_{ij}(r)\,|\,0\le r\le 1)$, $b_j(r)=(\underline{b}_j(r),\overline{b}_j(r)\,|\,0\le r\le 1)$.

В СЛАУ (1) нечеткие переменные представим в эквивалентной уровневой форме с использованием обратных отображений, тогда (1) в матричной форме может быть представлена в виде двух совокупностей $\{\dots\}_1$, $\{\dots\}_2$ четких СЛАУ:

$$\underline{\left\{\grave{A}(r)\cdot\tilde{O}(r)=\underline{b}(r)\,|\,0\le r\le 1\right\}};\qquad \left\{\overline{A}(r)\cdot\overline{X}(r)=\overline{b}(r)\,|\,0\le r\le 1\right\}$$

(п5.2)

Положим в (2) r: $r_0=0,\dots,r_k=1$, тогда для каждого фиксированного r_l, $l=\overline{0,k}$ получим решение (п5.2) нейросетевым методом (см. лаб. раб. «Решение четкой СЛАУ нейросетевым методом»):

$$X(r_l)=(\underline{X}(r_l),\overline{X}(r_l)\,|\,r_l\in[0;1]),$$

где $\underline{X}(r_l)$ - решения совокупностей $\{\ldots\}_1$ четких СЛАУ; $\overline{X}(r_l)$ - решения совокупностей $\{\ldots\}_2$ четких СЛАУ.

Рассмотрим одну из компонент $x_j(r)$ вектора $X(r)$. Для нее в системе координат (x_j,r) (рис. п5.2) имеем совокупность точек $\left\{\underline{x}_j(r_l), r_l\right\}_{l=0}^{k}$, $\left\{\overline{x}_j(r_l), r_l\right\}_{l=0}^{k}$. Обратное отображение этих совокупностей определяет искомую функцию принадлежности $r_{x_j}(x)$ решения задачи (1) в дискретной форме.

Четкое решение нечеткой задачи (п5.1) определяется применением оператора дефазификации (*dfz*) , например, методом центра тяжести (*cog* - center of gravity) к совокупности $\left\{r_{x_j}(x_p), x_p\right\}$ дискретных переменных.

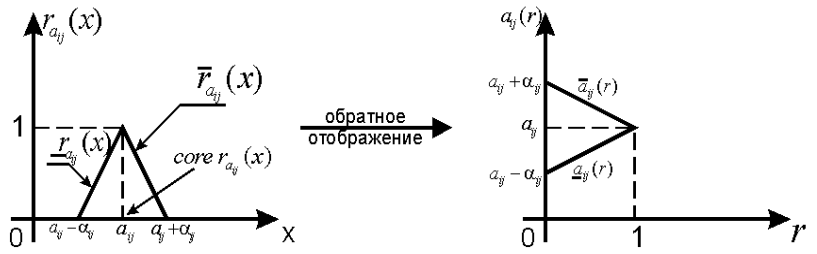
Алгоритм решения нечеткой СЛАУ состоит из следующих шагов:

1. Для заданных функций принадлежностей параметров нечеткой СЛАУ находятся обратные отображения и для них составляются две совокупности четких СЛАУ.
2. Каждая из этих совокупностей при фиксированных *r:* $r_0=0,\ldots,r_k=1$ решается нейросетевым методом. В результате будем иметь $2(k+1)$ четких нейросетей. Их выходом является совокупность $\left\{X(r_p), x_p\right\}$.
3. Обратное отображение $\left\{X(r_p), x_p\right\}^{-1}$ определяет совокупность $\left\{r_X(x_p), x_p\right\}$, которое определяет в дискретной форме функцию принадлежностей искомого решения (1).
4. Оператор $dfzcog\left\{r_X(x_p), x_p\right\}$ дает четкое решение нечеткой СЛАУ (1).

3. Практическая часть.

Рассмотреть надежность технических средств систем управления (СУ) при горячем резервировании основного комплекса одним резервным блоком с его восстановлением посредством ремонта (рис. п5.3).

а)



б)

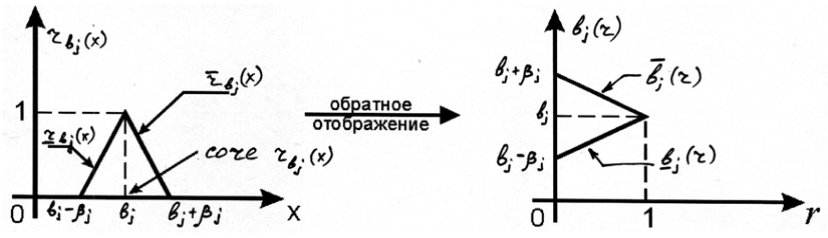


Рис. п5.1. Функции принадлежностей $r_{a_{ij}}(x)$, $r_{b_j}(x)$ нечётких переменных $a_{ij}(r)$, $b_j(a)$, (b) и их обратные отображения $a_{ij}(r)$, $b_j(r)$, $r \in [0;1]$

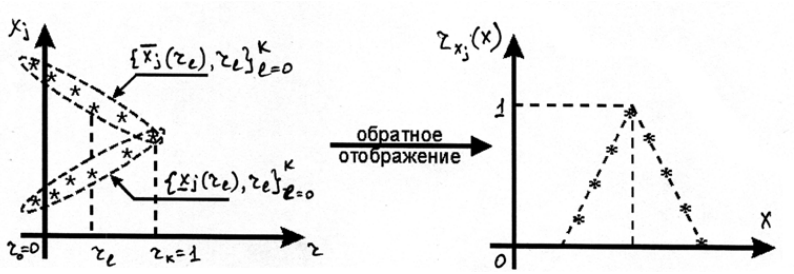


Рис. п5.2. Функция принадлежностей $r_{x_j}(x)$ нечёткого решения нечёткой СЛАУ.

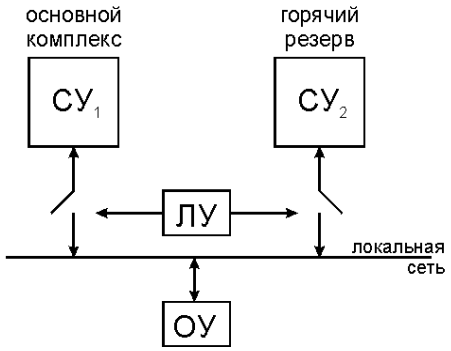


Рис. п5.3. Блок-схема технических средств системы управления (ЛУ – логическое устройство; ОУ – объект управления).

3.1. Составить граф состояний для СУ (рис. п5.3) при нечетком потоке $\lambda(t)$ отказов и нечетком потоке $\mu(t)$ восстановлений, и для графа состояний записать нечеткие дифференциальные уравнения Колмогорова.
3.2.Для $t \to \infty$ записать нечеткую СЛАУ для нахождения нечеткого коэффициента готовности $\hat{E}_{\tilde{a}} = (\underline{\hat{E}}_{\tilde{a}}(r), \overline{\hat{E}}_{\tilde{a}}(r) \mid 0 \le r \le 1)$ технических средств.
3.3. Задать $\lambda(t) = \lambda = (\underline{\lambda}, \overline{\lambda})$, $\mu(t) = \mu = (\underline{\mu}, \overline{\mu})$ и для этих значений при *r: r_0=0,...,r_k=1* найти $\underline{\hat{E}}_{\tilde{a}}(r)$, $\overline{\hat{E}}_{\tilde{a}}(r)$ нейросетевым методом.
3.4. Найти $\hat{E}_{\tilde{a}}\big|_{r=1}$ путем решения четкой СЛАУ и сравнить с $\underline{\hat{E}}(r=1)$, $\overline{\hat{E}}(r=1)$.
3.5. Найти $\underline{\hat{E}}_{\tilde{a}}(r)$, $\overline{\hat{E}}_{\tilde{a}}(r)$ при *r=0; 0.25; 0.5; 0.75; 1.0* методами теории автоматического регулирования, используя язык блок-схем, в системе MatLab/Simulink.
3.6. Результаты вычислений $K_г$ по п.п. 3.4, 3.5 представить в виде таблицы.

	λ		μ		$K_г$		$K_г$ *dfzcog*	$K_г$ *dfzcom*	$K_г$ *simulink*
	$\underline{\lambda}$	$\overline{\lambda}$	$\overline{\mu}$	$\overline{\mu}$	*r=0*	*r=1*			
$K_г$									

4. Контрольные вопросы
4.1. Чем отличаются решения четкой и нечеткой СЛАУ нейросетевым методом?
4.2. Как сказывается симметричность функций принадлежностей потоков отказов и восстановлений на вид функции принадлежностей коэффициента готовности?
4.3. Какую физическую интерпретацию можно придать нечеткости потоков отказов и восстановлений?

Список литературы.
См. ***Список литературы*** лабораторной работы П4.