Лабораторная работа №2.

Тема: аппроксимация функции.

Цель работы: изучение возможности аппроксимации с помощью нейронной сети прямого распространения.

Теоретические сведения

Аппроксимация – замена одних математических объектов другими, в том или ином смысле близкими к исходным. Аппроксимация позволяет исследовать числовые характеристики и качественные свойства объекта, сводя задачу к изучению более простых или более удобных объектов (например, таких, характеристики которых легко вычисляются или свойства которых уже известны).

Аппроксимация может сводиться к нахождению функциональной зависимости по имеющемуся набору точек. В данном случае, функциональная зависимость будет представлена в нейросетевом базисе, т.е. через комбинацию активационных функций нейронных сетей.

Типы активационных функций

1. Линейная $f(S) = k * S$, рис. 2.1.а.
2. Пороговая $f(S) = \begin{cases} 1, S > 0 \\ 0, S \le 0 \end{cases}$, рис. 2.1.б.
3. Логистическая (сигмоидальная) $f(S) = \frac{1}{1 + e^{-\alpha s}}$, рис. 2.1.в.
4. Гиперболический тангенс $f(S) = \frac{e^{\alpha s} - e^{-\alpha s}}{e^{\alpha s} + e^{-\alpha s}}$, рис. 2.1.д.
5. Функция радиального базиса $f(S) = e^{-s^2}$, рис. 2.1.е.
6. Знаковая $f(S) = \begin{cases} 1, S > 0 \\ -1, S \le 0 \end{cases}$, рис. 2.1.з.
7. Полулинейная $f(S) = \begin{cases} k * S > 0 \\ 0, S \le 0 \end{cases}$, рис. 2.1.к.
8. Экспоненциальная $f(S) = e^{\alpha s}$

9. Синусоидальная $f(S) = \sin(S)$

10. Синусоидальная-рациональная $f(S) = \frac{S}{dt|S|}$

11. Квадратичная $f(S) = S^2$

12. Модульная $f(S) = |S|$

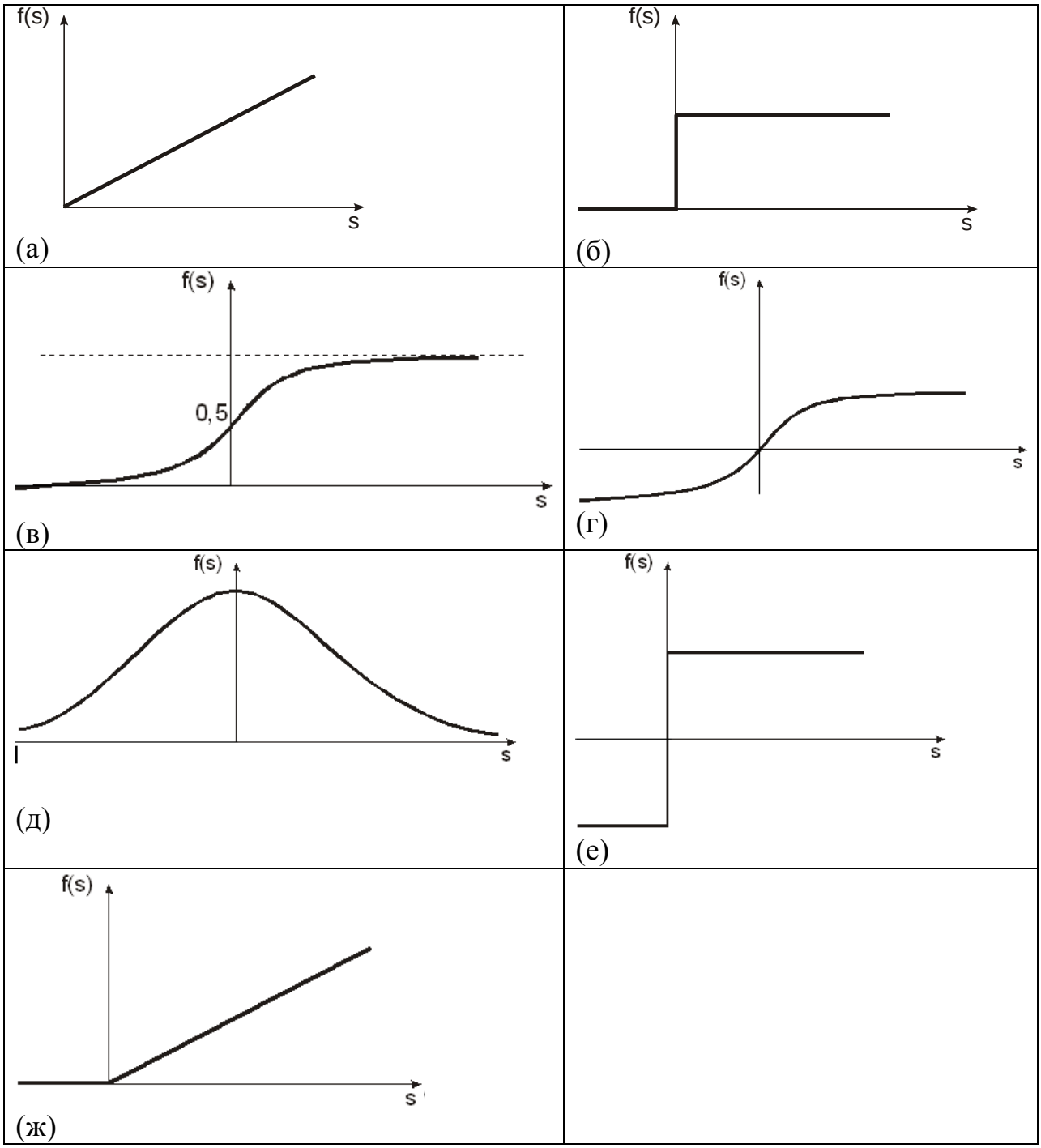

Рис. 2.1. Активационные функции нейронов: а – линейная; б – пороговая; в – логистическая (сигмоидальная); г – гиперболический тангенс; д – функция радиального базиса; е – знаковая; ж – полулинейная.

Согласно теореме, доказанной в 1989г. Л.Фунахаши, произвольно сложная функция может быть аппроксимирована двуслойной нейронной сетью, имеющей произвольные активационные функции. Поэтому классически для решения задачи аппроксимации используют многослойный персептрон. В данной лабораторной работе для решения задачи аппроксимации предлагается использование трехслойной нейронной сети, содержащей входной, один скрытый и выходной слои. Первый (входной) слой содержит число нейронов, равное количеству входных факторов.

Выходной слой содержит один нейрон, выдающий выходное значение чаще всего с линейной активационной функцией. Число нейронов второго (скрытого) уровня первоначально берется завышенное, например, равным полусумме числа обучающих данных:

$$n^{(2)} = \frac{\sum_p P \cdot k_p}{2} \qquad (2.1)$$

где P-число входных факторов, k_p - глубина ретроспективной выборки по p-ому фактору. Затем число нейронов второго слоя уменьшается до тех пор, пока ошибка аппроксимации не начнет возрастать. Функцией активации традиционно являются сигмоидальный тангенс или гиперболический тангенс, так как они являются гладкими, дифференцируемыми, а их производная выражается через саму функцию, что важно для обучения сети. Сигмоидальный тангенс предпочтительнее в связи с несимметричностью области определения [0;1] сигмоиды, что уменьшает скорость обучения.

Для простейшего случая аппроксимации – аппроксимации одномерной функции $y = f(x)$ нейронная сеть выполняет функцию нахождения преобразования $f(x)$ через активационные функции. Число входов в случае одномерной функции равно 1 (x), число выходов – 1 (y). Нейронная сеть в таком случае может быть представлена рис. 2.2.

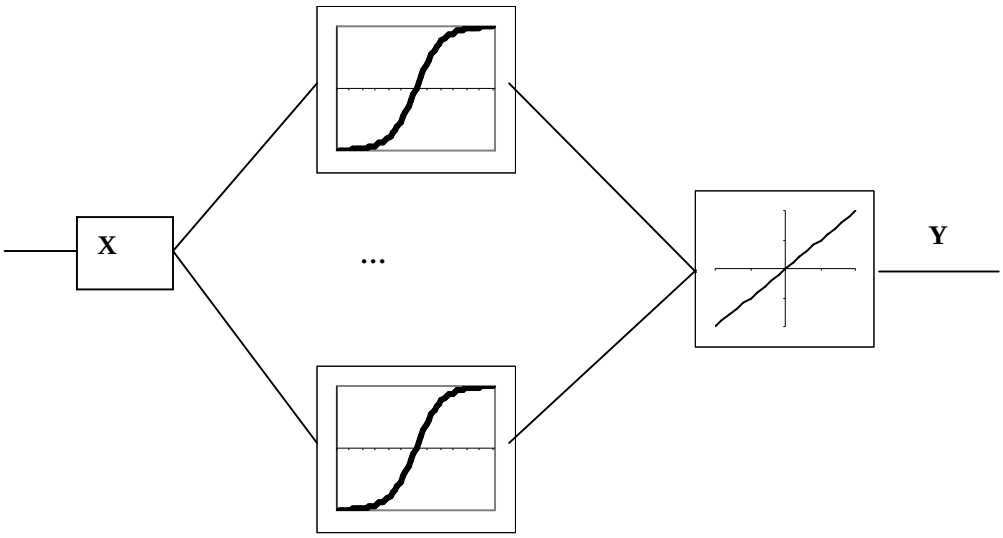

Рис. 2.2. структура нейронной сети.

Обучение многослойного персептрона чаще всего происходит с помощью алгоритма обратного распространения ошибки или его модификации.

Алгоритм обратного распространения ошибки

Долгое время не было теоретического обоснования обучения многослойных НС. А так как возможности НС сильно ограничены (несмотря на то, что алгоритмы обучения этих сетей были известны), то это сильно сдерживало развитие НС. Разработка алгоритма обратного распространения сыграла очень важную роль возрождения интереса к исследованию НС.

Обратное распространение – системный метод обучения многослойных НС. Он имеет серьезное математическое обоснование. Несмотря на некоторые ограничения, этот алгоритм сильно расширил область проблем, где могут быть использованы НС.

Этот алгоритм был фактически заново открыт и популяризован Румельхарт Макклеланд (1986) из знаменитой группы по изучению распределенных процессов.

На этапе обучения происходит вычисление синаптических коэффициентов *w*. При этом в отличие от классических методов в основе лежат не аналитические вычисления, а методы обучения по образцам с помощью примеров сгруппированных в обучаемом множестве. Для каждого образа из обучающей выборки считается известным требуемое значение выхода НС (обучение с учителем). Этот процесс можно рассматривать как решение оптимизационной задачи. Ее целью является минимизация функции ошибки или невязки E на обучающем множестве путем выбора значений синоптических коэффициентов *w*.

$$E = \frac{1}{2}\sum_{i}^{p}(d_i - y_i)^2 , \tag{2.2}$$

d_i – требуемое (желаемое) значение выхода на j-м образце выборки;

y_i – реальное значение;

p – число образцов обучающей выборки.

Минимизация ошибки E обычно осуществляется с помощью градиентных методов. При этом изменение весов происходит в направлении обратном направлению наибольшей крутизны функции ошибки.

$$W(t+1) = W(t) - \eta \frac{\partial E}{\partial W}, \tag{2.3}$$

$0 < \eta \le 1$ - определяемый пользователем параметр.

Существует два подхода к обучению. В первом из них веса w пересчитываются после подачи всего обучающего множества, и ошибка имеет вид:

$$E = \frac{1}{2}\sum_{i}^{p}(d_i - y_i)^2 \tag{2.4}$$

Во втором подходе ошибка пересчитывается после каждого образца:

$$E_i = \frac{1}{2}(d_i - y_i)^2 \tag{2.5}$$

$$E(W) = \frac{1}{2}\sum_{j,p}(d_{jp} - y_{jp})^2 \tag{2.6}$$

Пусть

$$\Delta W_{ij}(t) = -\eta \frac{\partial E}{\partial W_{ij}} \tag{2.7}$$

$$W_{ij}(t+1) = Wij(t) + \Delta W_{ij}(t) \tag{2.8}$$

Тогда

$$\frac{\partial E}{\partial W_{ij}} = \frac{\partial E}{\partial y_{ij}} * \frac{\partial y_i}{\partial S_i} * \frac{\partial S_j}{\partial W_{ij}} \tag{2.9}$$

$y_j(S_j)$ – активационная функция. Для

$$y_j = \bar{f}(S) = \frac{1}{1+e^{-Sj}} \tag{2.10}$$

$$\bar{f}'(S) = \frac{\partial}{\partial S_j}\left(\frac{1}{1+e^{-Sj}}\right) = \frac{1}{(1+e^{-Sj})^2}(-e^{-Sj}) = \frac{1}{1+e^{-Sj}} * \frac{-e^{-Sj}}{1+e^{-Sj}} = y_j(1-y_j) \tag{2.11}$$

$$\frac{dy_i}{dS_j} = (1-y_j)^2 \tag{2.12}$$

рассмотрим третий сомножитель:

$$S_j = \sum W_{ij} * y_i^{n-1}, \frac{\partial S_i}{\partial W_{ij}} = y_i^{(n-1)} \tag{2.13),(2.14}$$

Можно показать, что

$$\frac{\partial E}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} * \frac{dy_k}{dS_k} * \frac{\partial S_k}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} * \frac{dy_k}{dS_k} * W_{jk}^{(n+1)} \tag{2.15}$$

при этом суммирование к идет среди нейронов *n*-го слоя.

Введем новое обозначение:

$$\delta_j = \frac{\partial E}{\partial y_k} * \frac{dy_j}{dS_j} \tag{2.16}$$

$$\delta_j^{n-1} = \left[\sum_k \delta_k^n * W_{jk}^n\right] * \frac{dy_j}{dS_j} \tag{2.17}$$

Для внутреннего нейрона (2.17)

$$\delta_j^n = \left(d_j^n - y_j^n\right) * \frac{dy_j}{dS_j} \tag{2.18}$$

Для внешнего нейрона (2.18)

$$\Delta W_{ij}^n = -\eta \delta^{(n)} * y_j^{n-1} \tag{2.19}$$

$$W_{ij}(t+1) = W_{ij}(t) + \Delta W_{ij} \qquad (2.20)$$

Таким образом, полный алгоритм обучения обратного распространения строится так:

1. Подать на входы сети один из возможных образцов и в режиме обычного функционирования НС, когда сигналы распространяются от входа к выходам рассчитать значения выходов всех нейронов (обычно начальное значение веса составляют малые значения).

2. Рассчитать значения δ_j^n для нейронов выходного слоя по формуле Для внешнего нейрона (2.18).

3. Рассчитать значения δ_j^n для всех внутренних нейронов по формуле (2.17).

4. С помощью формулы (2.19) для всех связей найти приращения весовых коэффициентов ΔW_{ij}.

5. Скорректировать синаптические веса: $W_{ij}(t+1) = W_{ij}(t) + \Delta W_{ij}$

6. Повторить шаги 1-5 для каждого образа обучающей выборки пока ошибка *E* не станет достаточно маленькой.

Реализация многослойной сети в MATLAB

Создание нейронной сети происходит с помощью функции newff:

```
net=newff(PR, [S1, S2, …, SN],{TF1, TF2, … , TFn}, BTF, BLF, PF);
```

Аргументы функции:

PR – матрица минимальных и максимальных значений *R* входных элементов;

S_i – размер *i*-го слоя, для *n* слоев;

TF_i – функции активации нейронов *i* – го слоя, по умолчанию – «*tansig*»;

BTF – функция обучения сети, по умолчанию – «*traingd*»;

BLF – функция настройки весов и смещений, по умолчанию – «*learngdm*»;

PF – функция ошибки, по умолчанию – «mse».

В качестве примера можно привести создание нейронной сети с 5 входами, 10 нейронами на скрытом слое и 1 выходным нейроном:

```
net=newff([-2 2; -2 2; -2 2; -2 2; -2 2],
[10 1],{'tansig' 'purelin'});
```

После создания нейронной сети её необходимо проинициализировать малыми случайными значениями

```
init(net);
```

Обучение сети производится функцией train:

```
[net,TR,Y,E]=train(net,xtrain,ytrain);
```

Аргументы функции:

TR – набор записей для каждой эпохи обучения (эпоха и погрешность)

Y – набор реальных (не эталонных) выходов, полученных нейронной сетью для обучающих данных;

E – массив ошибок для каждого примера;

x_{train},y_{train} – массивы входов и выходов обучающей выборки.

После обучения сети необходимо её использовать на тестовой выборке:

```
ytest1=sim(net,xtest);
```

Массив y_{test1} содержит апроксимированные точки, полученные нейронной сетью для входных значений x_{test}.

Ход работы

Для заданной функции построить ее табличные значения (количество значений должно быть достаточным для того, чтобы аппроксимированная функция визуально совпадала с табличными значениями). Использовать нейронную сеть для аппроксимации этих табличных значений и нахождения аппроксимированной функции. Вывести на график табличные значения (точками) и аппроксимированную функцию (линией). Найти численное значение погрешности результата аппроксимации, вывести на экран. Обучение нейронной сети выполнять с помощью функции train.

Таблица 2.1.

Индивидуальные задания

Вариант	Вид функции	Промежуток нахождения решения
1	(1.85-x)*cos(3.5x-0.5)	x∈[-10,10]
2	cos(exp(x))/sin(ln(x))	x∈[2,4]
3	sin(x)/x2	x∈[3.1,20]
4	sin(2x)/x2	x∈[-20,-3.1]
5	cos(2x)/x2	x∈[-20,-2.3]
6	(x-1)cos(3x-15)	x∈[-10,10]
7	ln(x)cos(3x-15)	x∈[1,10]
8	cos(3x-15)/abs(x)=0	x∈[-10,-0.3),(0.3,10] x∈[-0.3,0.3]
9	cos(3x-15)*x	x∈[-9.6,9.1]
10	sin(x)/(1+exp(-x))	x∈[0.5,10]
11	cos(x)/ (1+exp(-x)	x∈[0.5,10]
12	(exp(x)-exp(-x))cos(x)/ (exp(x)+exp(-x))	x∈[-5,5]
13	(exp(-x)-exp(x))cos(x)/ (exp(x)+exp(-x))	x∈[-5,5]
14	cos(x-0.5)/abs(x)	x∈[-10,0),(0,10] , min
15	cos(2x)/abs(x-2)	x∈[-10,2),(2,10], max

Контрольные вопросы:

1. Что такое аппроксимация?
2. Параметры обучения при использовании для обучения функции train.
3. Виды прекращения обучения сети при использовании функции обучения train.
4. Способы нахождения погрешности результата.