DOI: 10.58168/ROBOTICS2024_182-191
УДК 681.5.017

ОСНОВЫ ПОСТРОЕНИЯ НЕЙРОННЫХ СЕТЕЙ В СИСТЕМАХ АВТОМАТИЗАЦИИ

FUNDAMENTALS OF BUILDING NEURAL NETWORKS IN AUTOMATION SYSTEMS

Шестаков Д.В., студент
Михайлов А.М., студент
Щербакова И.В., к.т.н., доцент
ФГБОУ ВО «Воронежский государственный лесотехнический университет имени Г.Ф. Морозова»
г. Воронеж, Россия
shcherbakovaiv@vglta.vrn.ru
Shestakov D.V., Student
Mikhailov A.M., Student
Shcherbakova I.V., Candidate of Technical Sciences, Associate Professor
FSBEI HE «Voronezh State University of Forestry and Technologies named after G.F. Morozov»
Voronezh, Russian Federation

Аннотация: В работе рассматриваются принципы работы и основы построения нейронных сетей применительно к их использованию в системах автоматизации. Приведен пример обучения нейронной сети алгоритмом backward propagation.

Abstract: The paper discusses the principles of operation and the basics of building neural networks in relation to their use in automation systems. An example of training a neural network using the backward propagation algorithm is given.

Ключевые слова: нейронная сеть, перцептрон, система автоматизации, алгоритм backward propagation, программа, python.

Keywords: neural network, perceptron , automation system, backward propagation algorithm, program, python.

Одним из перспективных направлений совершенствования систем автоматизации является применение нейронных сетей. Для решения задач автоматического управления производственными процесссами чаще всего исползуются полносвязные нейронные сети прямого распространения. Полносвязная нейронная сеть прямого распространения — это тип нейронной сети, в которой каждый нейрон текущего слоя связан с каждым нейроном следующего слоя. Прямое распространения означает, что сигнал распространяется от входа к выходу без обратных связей. На рисунке 1 представлена структура постороения ИНС прямого распространения.

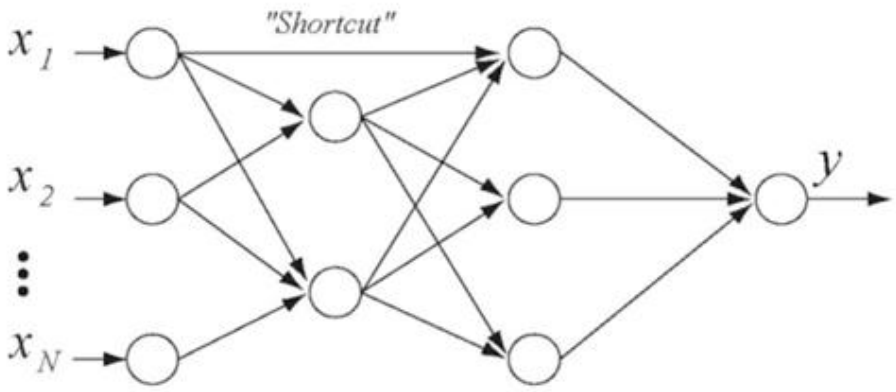

Рисунок 1 – ИНС прямого распространения

В основе построения нейронной сети лежит нейрон. Нейрон принимает несколько входных сигналов, взвешивает их и суммирует. Затем сумма пропускается через функцию активации

$$f(x)=\begin{cases}1, & x\geq 0,5\\ 0, & x<0,5\end{cases},$$

которая определяет выход нейрона:

$$x = w_1x_1 + w_2x_2 + \ldots. + w_nx_n.$$

Чтобы сделать нейронную сеть более эффективной, можно добавить скрытые слои. Скрытые нейроны могут обнаруживать более сложные закономерности во входных данных. Учитывая, что полносвязная нейронная сеть прямого распространения – это многослойный перцептрон, то для понимания принципов ее работы необходимо знать, что такое перцептрон.

Перцептрон – это простейшая нейронная сеть, которая может использоваться для классификации данных. В структуре персептрона можно выделить:

- входной слой (принимает входные данные);
- скрытый слой (обрабатывает данные);
- выходной слой (выдает результат классификации).

Разделение перцептрона на слои схематично представлено на рисунке 2.

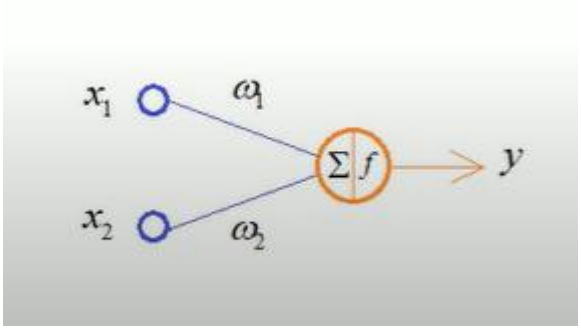

Рисунок 2 – Разделяющая прямая

Перцептрон может классифицировать данные, создавая разделяющую прямую. Эта прямая разделяет данные на два класса. Точки, которые находятся выше прямой, относятся к одному классу, а точки, которые находятся ниже прямой, относятся к другому классу. Пример классификации элементов нейронной сетью показан на рисунке 3.

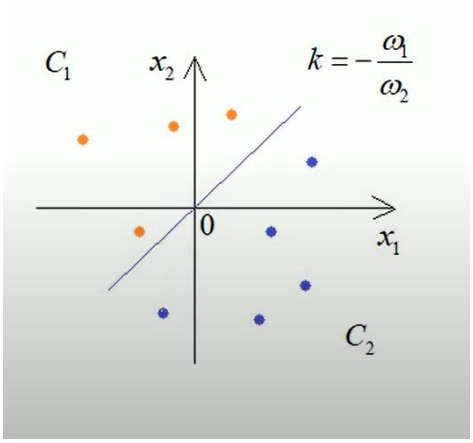

Рисунок 3 – Классификация элементов нейронной сетью

В ряде случаев в модели искусственного нейрона вводят дополнительный параметр – смещение, который позволяет настроить выход сети независимо от входных значений. Смещение можно рассматривать как

способ внесения некоторого предвзятого значения (bias) в сумму входных сигналов, прежде чем она будет передана функции активации.

Смещение дает следующие преимущества:

1. Сдвиг порога решения

Без смещения, только входные данные определяют активацию нейрона. Смещение позволяет модели регулировать порог активации.

2. Гибкость в обучении

Смещение позволяет нейронной сети лучше аппроксимировать различные функции путем смещения линии решения в многомерном пространстве.

3. Разделимость данных

В случае когда без смещения разделяющая гиперплоскость (линия решения) проходила бы через начало координат, смещение позволяет сдвинуть эту линию, чтобы лучше разделить классы.

В перцептроне и общих моделях искусственных нейронных сетей смещение представляет собой важную составляющую модели, которая способствует созданию более точных и универсальных прогностических систем.

Для создания более сложных классификаторов используют объединение перцептронов. Например, можно объединить два перцептрона, чтобы создать область классификации в форме полосы (рисунок 4).

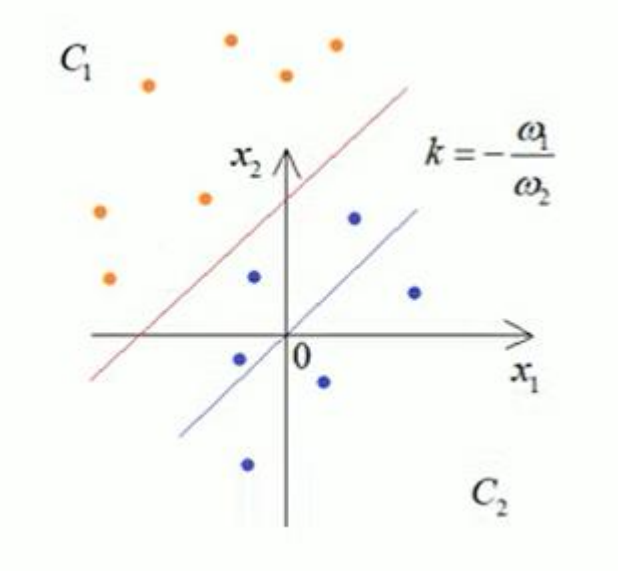

Рисунок 4 – Объединение перцептронов

Следует отметить, что при большом количестве нейронов появляется необходимость в разработке дополнительных алгоритмов.

Важнейшим принципом работы нейронной сети является ее обучение. Один из распространенных подходов к обучению – обучение с учителем. Он заключается в последовательном предъявлении нейронной сети векторов наблюдений и последующей корректировке весовых коэффициентов так, чтобы выходное значение соответствовало как можно ближе требуемому отклику. Поскольку для каждого вектора наблюдений у нас известен желаемый результат, то обучение состоит в требовании от сети, чтобы она показывала правильный результат, то есть разработчик выступает как бы в роли учителя: он говорим, что хорошо, что правильно, а что неправильно.

Рассмртрим особенности обучения нейронных сетей алгоритмом backward propagation. Обратное распространение ошибки (backpropagation) — это алгоритм обучения, используемый в многослойных нейронных сетях, который включает два основных этапа: прямое распространение и обратное распространение. Рассмотрим особенности второго этапа, т.е. обратное распространение (backward propagation):

- Ошибка, начиная с выходного слоя, используется для вычисления градиентов функции потерь по отношению к каждому весу в сети.

- С использованием такого правила дифференцирования, как цепное правило, градиенты распространяются назад, обновляя веса в предыдущих слоях.

- Градиенты смещения также вычисляются и обновляются.

Обновление весов происходит по правилу обновления, часто используя метод стохастического градиентного спуска (SGD) или его варианты. Чтобы минимизировать ошибку, веса обновляются в противоположном градиенту направлении.

Алгоритм backward propagation является итерационным. Процедуры прямого и обратного распространения повторяются множество раз (итераций) до тех пор, пока нейронная сеть не перестанет существенно сокращать ошибку, т.е. не будет считаться достаточно «обученной».

$$E = \frac{1}{2}\sum_{j=1}^{N} e_j^2 = \frac{1}{2}\sum_{j=1}^{N}\left(d_j - y_j\right)^2$$

Рассмртрим пример применеия алгоритма обучения нейронных сетей по методу обратного распространения ошибки. Будем обучать нейронную сеть на данных с требуемыми откликами, приведенных на рисунке 5.

Вектор наблюдений	Требуемый отклик
[-1, -1, -1]	-1
[-1, -1, -1]	1
[-1, -1, -1]	-1
[-1, -1, -1]	1
[-1, -1, -1]	-1
[-1, -1, -1]	1
[-1, -1, -1]	-1
[-1, -1, -1]	-1

Рисунок 5 – Исходные данные для обучения сети

Приведем листинг программы, написанной на python.

```
import numpy as np
# Функция для вычисления гиперболического тангенса
def f(x):
 return 2/(1+np.exp(-x)) - 1
# Функция для вычисления производной
def df(x):
 return 0.5*(1 + x)*(1 - x)
# Веса для первого слоя нейронных сетей
W1 = np.array([[0.2, 0.3, -0.4], [0.1, -0.3, -0.4]])
# Веса для второго слоя нейронных сетей
W2 = np.array([0.2, 0.3])
# Функция для обратного распространения ошибки
def go_forward(inp):
 sum = np.dot(W1, inp)
 out = np.array([f(x) for x in sum])
 sum = np.dot(W2, out)
 y = f(sum)
 return (y, out)
# Функция для обучения нейронной сети
def train(epoch):
 global W1, W2
 lmd = 0.01 # шаг обучения
 N = 10000 # количество эпох(итераций)
 count = len(epoch) # количество элементов в обучающей выборке
```

```
 for i in range(N):
 x = epoch[np.random.randint(0, count)] # случайный выбор элемента из обучающей выборки
 y, out = go_forward(x[0:3]) # прямой проход по нейронной сети и вычисление выходных значений
 e = y - x[-1] # ошибка
 delta = e * df(y) # дельта для выходного слоя (локальный градиент)
 W2[0] = W2[0] - lmd * delta * out[0] # корректировка веса первой связи
 W2[1] = W2[1] - lmd * delta * out[1] # корректировка веса второй связи
 delta2 = W2 * delta * df(out) # вектор из 2-х величин локальных градиентов
 # корректировка связей первого слоя
 W1[0, :] = W1[0] - np.array(x[0:3]) * delta2[0] * lmd
 W1[1, :] = W1[1] - np.array(x[0:3]) * delta2[1] * lmd
# Обучающая выборка
epoch = [(-1, -1, -1, -1),
 (-1, -1, 1, 1),
 (-1, 1, -1, -1),
 (-1, 1, 1, 1),
 (1, -1, -1, -1),
 (1, -1, 1, 1),
 (1, 1, -1, -1),
 (1, 1, 1, -1)]
# Обучение нейронной сети
train(epoch)
# Проверка работы нейронной сети
for x in epoch:
 y, out = go_forward(x[0:3])
 print(f'Выходное значение: {y}, Ожидаемое значение: {x[-1]}')
```

Скриншот вывода консоли после выполнения программы представлен на рисунке 6.

Опишем работу программы:

1. Импортируется библиотека numpy, которая является мощной библиотекой для числовых вычислений на Python.

2. Определяются две функции `f(x)` и `df(x)`. `f(x)` — это функция активации нейронов в сети. В данном случае это функция гиперболического

тангенса. `df(x)` — это производная функции активации, которая используется на этапе обратного распространения ошибки в процессе обучения.

```
C:\Users\user\PycharmProjects\ozon_bot\.venv\Scripts\python.exe C:\Users\user\PycharmProjects\ozon_bot\neuro.py
Выходное значение: 0.012932813328262682, Ожидаемое значение: -1
Выходное значение: 0.928435420291492, Ожидаемое значение: 1
Выходное значение: -0.8726038042203826, Ожидаемое значение: -1
Выходное значение: 0.8749769701448529, Ожидаемое значение: 1
Выходное значение: -0.8749769701448528, Ожидаемое значение: -1
Выходное значение: 0.8726038042203828, Ожидаемое значение: 1
Выходное значение: -0.9284354202914922, Ожидаемое значение: -1
Выходное значение: -0.01293281332826246, Ожидаемое значение: -1

Process finished with exit code 0
```

Рисунок 6 – Вывод консоли после выполнения программы

3. Программа инициализирует веса нейронов сети. «W1» — это двумерный массив, представляющий веса для первого слоя нейронов, а «W2» — это одномерный массив, представляющий веса для второго слоя нейронов.

4. Определяется функция `go_forward(inp)`. Эта функция принимает входной вектор `inp`, умножает его на веса первого слоя нейронов, применяет функцию активации, затем умножает результат на веса второго слоя нейронов и снова применяет функцию активации. Результатом является выход сети для данного входа.

5. Определяется функция `train(epoch)`. Эта функция обучает сеть с использованием алгоритма обратного распространения ошибки. Он принимает массив обучающих примеров «эпоха» и для каждого примера вычисляет выходные данные сети, вычисляет ошибку между выходными и ожидаемыми выходными данными и соответствующим образом корректирует веса нейронов.

6. Затем программа определяет набор обучающих примеров и обучает сеть с использованием этих примеров.

7. Программа тестирует сеть, вычисляя выходные данные для каждого обучающего примера, и распечатывает результат.

Применение нейронных сетей в регуляторах систем автотоматического управления имеет как свои достоинства, так и недостатки.

К недостаткам относят тот факт, что обучение нейронных сетей является сложным и длительным процессом. Но этот недостаток комппенсируется возможностью отказаться от формулировки точных правил, описывающих

поведение регулятора, а заменить их настройкой регулятора в процессе обучения нейронной сети.

Список литературы

1. Бишоп, К. М. Нейронные сети для распознавания образов / К. М.Бишоп. – Oxford: Clarendon press, 1996. – 482 с.

2. Рашка, С. Python и машинное обучение / пер. с англ. А. В. Логунова. – М.: ДМК Пресс, 2017. – 418 с.

3. Лисьих, А. С. Нейронные сети. Применение нейронных сетей в автоматизации процессов / А. С. Лисьих, А. А. Турчина, С. А. Шадрин // Исследования молодых ученых : Материалы LIII Международной научной конференции, Казань, 20–23 января 2023 года. – Казань: Молодой ученый, 2023. – С. 1-6.

4. Автоматизация с помощью нейросетей: Как искусственный интеллект меняет рабочие процессы и повседневную жизнь // vc.ru : сайт. – URL: https://vc.ru/u/2056024-setevye-mysli/766278-avtomatizaciya-s-pomoshchyu-neyrosetey-kak-iskusstvennyy-intellekt-menyaet-rabochie-processy-i-povsednevnuyu-zhizn (дата обращения: 19.04.2024).

5. Искусственные нейронные сети управления технологическими процессами // Control Engineering: сайт. – URL: https://controlengrussia.com/perspektiva/neural_networks_/ (дата обращения: 19.04.2024).

References

1. Bishop, K. M. Neural networks for pattern recognition / K. M. Bishop. – Oxford: Clarendon press, 1996. – 482 p.

2. Rashka, S. Python and machine learning / trans. from English by A. V. Logunova. – M.: DMK Press, 2017. – 418 p.

3. Lisikh, A. S. Neural networks. Application of neural networks in process automation / A. S. Lisikh, A. A. Turchina, S. A. Shadrin // Research of young scientists: Proceedings of the LIII International Scientific Conference, Kazan, January 20–23, 2023. – Kazan: Young Scientist, 2023. – P. 1-6.

4. Automation using neural networks: How artificial intelligence is changing work processes and everyday life // vc.ru: website. – URL: https://vc.ru/u/2056024-setevye-mysli/766278-avtomatizaciya-s-pomoshchyu-neyrosetey-kak-

iskusstvennyy-intellekt-menyaet-rabochie-processy-i-povsednevnuyu-zhizn (access date: 04/19/2024).

5. Artificial neural networks for controlling technological processes // Control Engineering: website. – URL: https://controlengrussia.com/perspektiva/neural_networks_/ (date of access: 04/19/2024).