Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №5 по курсу «Нейроинформатика»

Студент: М. А. Бронников
Преподаватель: Н. П. Аносова
Группа: М8О-407Б

Дата:
Оценка:
Подпись:

Москва, 2021

Сети с обратными связями

Цель работы: Исследование свойств сетей Хопфилда, Хэмминга и Элмана, алгоритмов обучения, а также применение сетей в задачах распознавания статических и динамических образов.

Основные этапы работы:

1. Использовать сеть Элмана для распознавания динамических образов. Проверить качество распознавания.
2. Использовать сеть Хопфилда для распознавания статических образов. Проверить качество распознавания.
3. Использовать сеть Хэмминга для распознавания статических образов. Проверить качество распознавания.

Вариант №4

1. $g(k) = \sin(\sin(k)k^3 - 10),\ \forall k \in [1.56, 3.12],\ \ R = [0, 1, 5]$
2. $[6, 3, 9]$

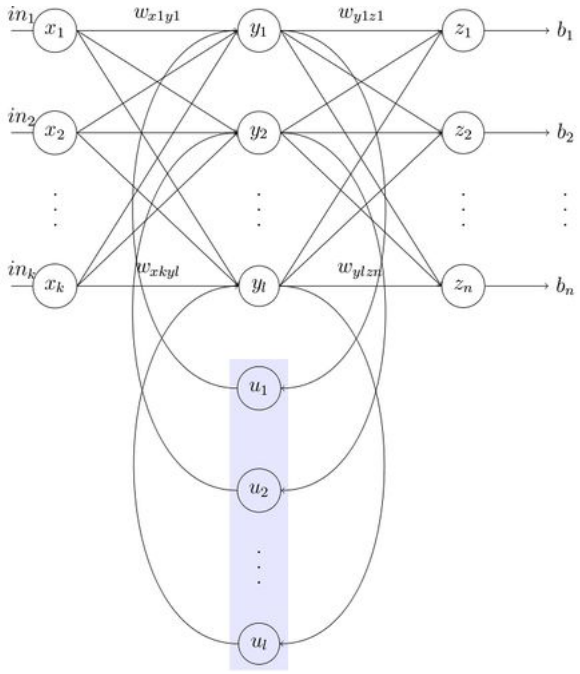
1 Структура модели

Нейронная сеть Элмана:

Сеть обладает скрытым слоем и производит расчет выходного слоя для t-го входного вектора по следующим формулам:

$y_t = \sigma_y(W_y x_t + U_y y_{t-1} + b_y)$

$z_t = \sigma_z(W_z y_t + b_z)$

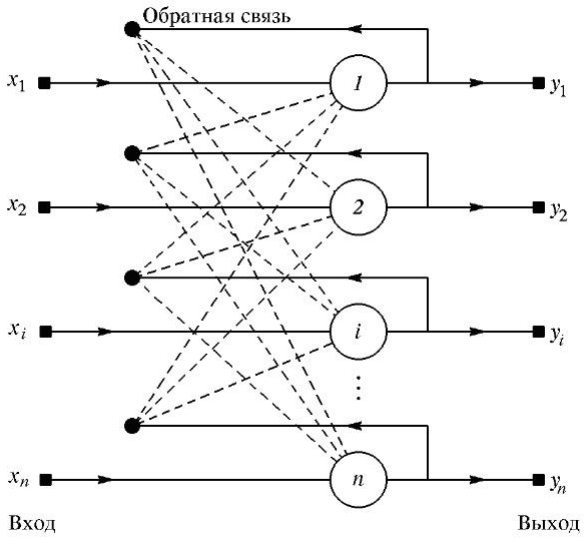


Нейронная сеть Хопфилда:

Каждый нейрон слоя из n нейронов связывается обратными связями с остальными $n-1$ нейронами этого слоя. Сеть позволяет корректировать искаженные образы.

Веса этого слоя могут быть получены с помощью аналитической формулы:

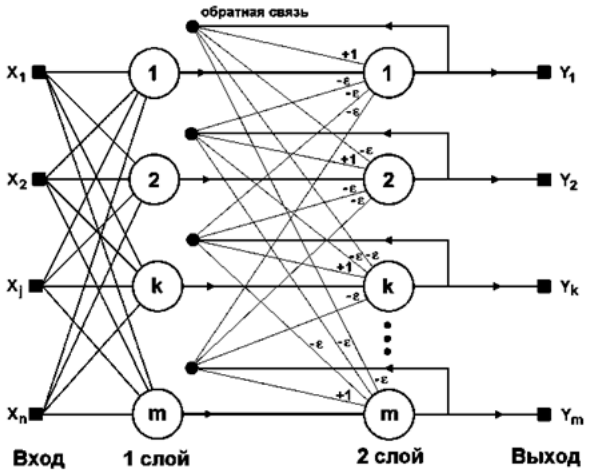
$$W = \frac{1}{N} \sum_{i=i}^{N} < x_i, x_i >$$



Нейронная сеть Хэмминга:

Сеть состоит из *полносвязанного слоя* вместе со *слоем сети Хопфилда*. Сеть позволяет классифицировать образы, поданные на вход.

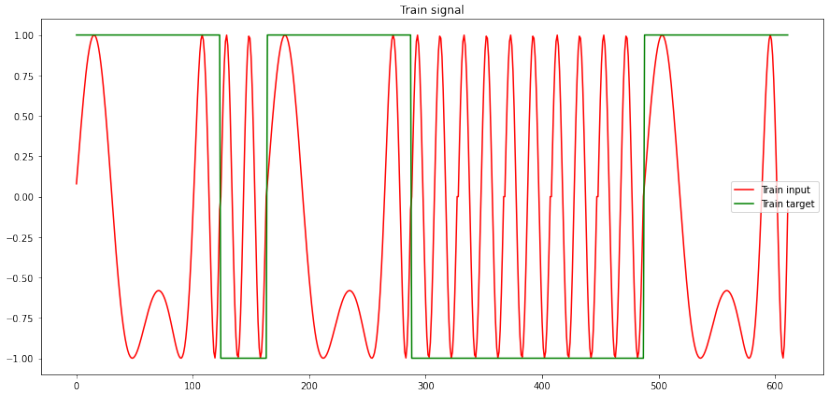
Веса полносвязного слоя могут быть найдены также аналитическим путем.



2 Ход работы

Этап №1

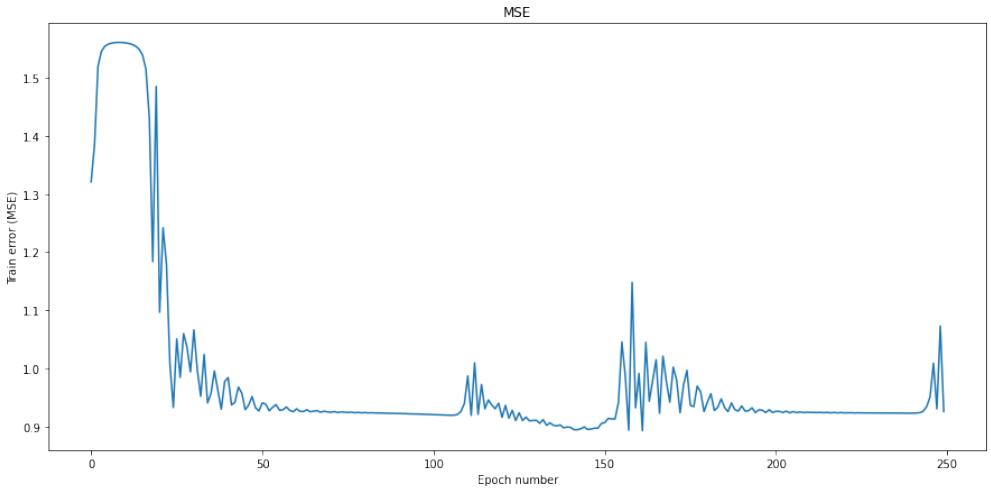
Я сгенерировал и отобразил обучающее множество:



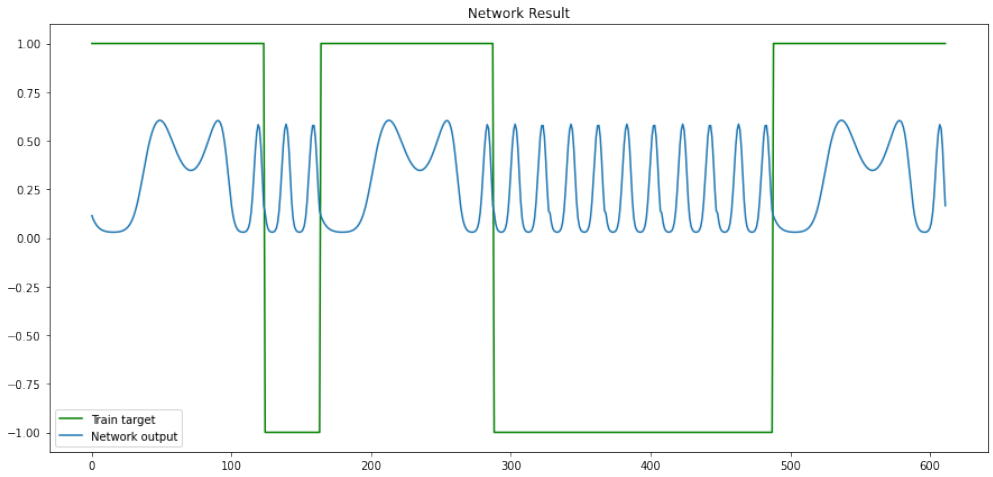
Создал и обучил *Нейронную сеть Элмана* и сделал предсказание:

```
net = nl.net.newelm([[-10, 10]], [8, 1], [nl.trans.TanSig(), nl.trans.TanSig()])

net.layers[0].np['w'][:] = 1
net.layers[0].np['b'][:] = 0
net.init()

error = net.train(P, T, epochs=250, show=50, goal=0.0001)

output = net.sim(P)
```

Функция ошибки:



Предсказание:



Расчет ошибки:

```
output[output >= 0] = 1.0
output[output < 0] = -1.0

MSE = mean_squared_error(T, output)
print('MSE = {}'.format(MSE))
print('RMSE = {}'.format(np.sqrt(MSE)))

>>> MSE = 1.5686274509803921
>>> RMSE = 1.2524485821702989
```

Этап №2

Входные данные для *сети Хопфилда* - черно-белые образы чисел.

Обучим нейронную сеть и сделаем предсказание:

```
data = np.concatenate([six, three, nine], axis=0)

hopf = algorithms.DiscreteHopfieldNetwork(mode='async', n_times=600)
hopf.train(data)

result = hopf.predict(six)
```

Сравнение входных зашумленных образов и скорректированных нейронной сетью:

```
>>> Noisy 20%:

    # # # # #
    # # # # # # #
              # #
              # #
              # #
        # # #
        # # # #
              # #
              # #
              # #
    # # # # # #
    # # # #

>>> Corrected:

    # # # # # # #
    # # # # # # # #
              # #
              # #
              # #
        # # # # #
        # # # # #
              # #
              # #
              # #
    # # # # # # # #
    # # # # # # #
```

Этап №3

Входные данные для *сети Хэмминга* - черно-белые образы чисел.

Зададим веса *полносвязного слоя* аналитически:

```
Q = 7
patterns = np.array([zero, one, two, three, four, six, nine])
nums = [0, 1, 2, 3, 4, 6, 9]
eps = 1 / (Q - 3)

shape = 10 * 12

IW = np.array([zero.T, one.T, two.T, three.T, four.T, six.T, nine.T])
b = shape * np.ones((Q, 1))

a = np.zeros((Q, Q))
for i in range(Q):
    a[i] = IW[i]  patterns[i] + b[i]LW = np.eye(Q)LW[LW == 0.0] = -eps
```

Расчитаем выход (предсказаный класс) сети:

```
A = IW  six + bres = network.sim(A)
```

Посмотрим на полученный ответ для зашумленных данных.

```
answer_class = np.argmax(res[0])
print('Result class: {}'.format(nums[answer_class]))

number = patterns[answer_class]
number[number == -1] = 0
draw_image(number.reshape(12, 10))

>>> Result class: 3
    # # # # # #
    # # # # # # #
              # #
              # #
              # #
        # # # #
        # # # #
              # #
              # #
              # #
    # # # # # # #
    # # # # # #
```

3 Выводы

Выполнив пятую лабораторную работу по курсу «Нейроинформатика», я узнал о рекуррентных нейронных сетях, использующих обратные связи, которые зачастую позволяют лучше запоминать уже просмотренные последовательности.

Я рассмотрел несколько вариантов использваония рекуррентных нейронных сетей, таких как предсказание следующего шага временной последовательности и классификация и корректировака поврежденных образов.

Эта работа была интересной и оказалась более простой, чем предыдущая лабораторная работа, поэтому в будущем я ожидаю более трудных и интересных задач.