

Нейронные сети, классификация

С.И.Хашин

<http://math.ivanovo.ac.ru/dalgebra/Khashin/index.html>

Ивановский государственный университет

Иваново-2023

План

Введение

Модель

Запуск

Спираль

MNIST

CIFAR

2_соя

TensorFlow: Установка

```
pip install tensorflow
```

Затем в программах надо будет подключать TensorFlow:

```
import numpy as np
import tensorflow as tf
import time
```

```
np.set_printoptions(precision=4, linewidth=100,
                    suppress=True)
np.random.seed(777)
```

Подготовка данных

```
Y,X = data_load(0)
# варианты нормализации
#if X.max()==255: X /= 255
#X0 -= X0.mean(axis=0) # Из каждого столбца матрицы X вычесть среднее
#X /= X.std(axis=0) # Каждый столбец матрицы X поделить на стандартное отклонение

N,K = X.shape # кол-во векторов и размерность вектора
M = Y.max() + 1 # M - количество классов (4 для 4circles)
print(f'N={N}, K={K}, M={M}')
> N=8000, K=2, M=4
```

Построение модели

Строим модель в один слой:

```
model = tf.keras.Sequential(  
    [ tf.keras.Input(shape=(K,)),  
      tf.keras.layers.Dense(M, activation="softmax"),]  
)  
model.compile(optimizer = 'adam',  
              loss = 'sparse_categorical_crossentropy',  
              metrics = ['accuracy'])  
print('model:', model.summary())  
print('npar=', model.count_params()); #input('<cr>')
```

Параметры модели

Модель в один слой:

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 4)	12

```
Total params: 12
```

```
Trainable params: 12
```

```
Non-trainable params: 0
```

```
model: None
```

```
npar= 12
```

Запуск

```
nEpochs = 10 # количество эпох
tm = time.time()
model.fit(x=X, y=Y, epochs=nEpochs)
tm = (time.time() - tm) / nEpochs
test_loss, test_acc = model.evaluate(X, Y)
print(f'\nТочность:{test_acc:6.3f}, время:{tm:5.2f}
      сек./эпоху')
```

Результат

Epoch 1/100

250/250 - loss: 4.1706 - accuracy: 0.3495

Epoch 2/100

250/250 - loss: 2.5480 - accuracy: 0.3158

...

Epoch 99/100

250/250 - loss: 0.0322 - accuracy: 0.9926

Epoch 100/100

250/250 - loss: 0.0319 - accuracy: 0.9926

250/250 - loss: 0.0317 - accuracy: 0.9929

Спираль spiral_02_03.csv

```
Y,X = data_load(1)
```

```
...
```

```
Model: "sequential"
```

```
-----  
Layer (type)      Output Shape      Param #  
=====
```

dense (Dense)	(None, 3)	9
---------------	-----------	---

```
=====
```

```
Total params: 9
```

```
Trainable params: 9
```

```
Non-trainable params: 0
```

```
-----  
model: None npar= 9
```

Спираль, результаты

Epoch 1/100

188/188 - loss: 2.1684 - accuracy: 0.1433

Epoch 2/100

188/188 - loss: 1.7031 - accuracy: 0.1775

...

Epoch 99/100

188/188 - loss: 0.8501 - accuracy: 0.5203

Epoch 100/100

188/188 - loss: 0.8501 - accuracy: 0.5207

sky_data.csv

```
Y,X = data_load(3)
X -= X.mean(axis=0)
X /= X.std(axis=0)
...
model.compile(optimizer =
                tf.keras.optimizers.Adam(learning_rate=0.5),
                loss = 'sparse_categorical_crossentropy',
                metrics = ['accuracy'])
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 3)	9

Total params: 9

sky_data, результаты

Epoch 1/100

313/313 loss: 0.5255 - accuracy: 0.8292

Epoch 2/100

313/313 loss: 0.4491 - accuracy: 0.8817

...

Epoch 99/100

313/313 loss: 0.4899 - accuracy: 0.8892

Epoch 100/100

313/313 loss: 0.4722 - accuracy: 0.8929

Точность: 0.893, время: 0.22 сек./эпоху

Напомню, что наш результат 0.8992.

MNIST

```
Y,X = data_load(4)
```

```
...
```

```
Model: "sequential"
```

```
-----  
Layer (type)      Output Shape      Param #
```

```
=====
```

dense (Dense)	(None, 10)	7850
---------------	------------	------

```
=====
```

```
Total params: 7,850
```

MNIST

Epoch 1/10

2188/2188 loss: 8.8338 - accuracy: 0.8604

...

2188/2188 loss: 9.1387 - accuracy: 0.8987

Точность: 0.899, время: 2.01 сек./эпоху

Напомню, что наш результат 0.9378.

CIFAR

```
Y,X = data_load(4)
```

```
...
```

```
Model: "sequential"
```

```
-----  
Layer (type)      Output Shape      Param #
```

```
=====
```

dense (Dense)	(None, 10)	30730
---------------	------------	-------

```
=====
```

```
Total params: 30,730
```

CIFAR

Epoch 1/10

1875/1875 loss: 168.2804 - accuracy: 0.2415

Epoch 2/10

1875/1875 loss: 164.0482 - accuracy: 0.2647

...

Epoch 10/10

1875/1875 loss: 152.5479 - accuracy: 0.2869

1875/1875 loss: 180.2991 - accuracy: 0.2767

Точность: 0.277, время: 2.69 сек./эпоху

Напомню, что наш результат 0.2401.

Разбивка train/test

```
Y_train,X_train, Y_test,X_test = train_test(Y,X, 0.25)
print('train shape:', Y_train.shape, X_train.shape)
print('test  shape:', Y_test .shape, X_test.shape)
...
model.fit(x=X_train, y=Y_train, epochs=nEpochs)
...
test_loss, test_acc = model.evaluate(X_test, Y_test)
...
```

Разбиение на batches

```
...  
model.fit(x=X_train, y=Y_train, epochs=nEpochs,  
          batch_size=100)  
...
```

Задание

- Задание.** 1. Проверить, как влияет нормализация обучающей матрицы X на результат при всех входных данных.
2. Попробовать различный LR:

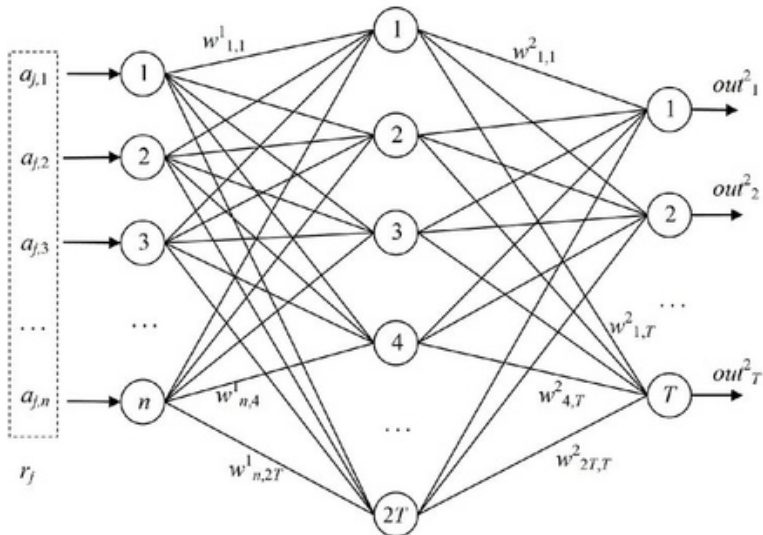
```
model.compile(  
    optimizer = tf.keras.optimizers.Adam(learning_rate=0.5),  
    loss = 'sparse_categorical_crossentropy',  
    metrics = ['accuracy'])
```

3. Попробовать различные оптимизаторы:

```
optimizer=keras.optimizers.Adagrad(),  
optimizer=keras.optimizers.Adadelta(),  
optimizer=keras.optimizers.RMSprop(),  
optimizer=keras.optimizers.SGD(lr=0.01,  
    decay=1e-6, momentum=0.9, nesterov=True),
```

4. Попробовать различный размер блока.

Двухслойная сеть



Двухслойная сеть

```
...  
N1 = 8 # нейронов в 1 слое  
model = tf.keras.Sequential(  
    [  
        tf.keras.layers.Dense(N1, activation = tf.nn.relu,  
                                input_shape=(K,)),  
        tf.keras.layers.Dense(M, activation="softmax"),  
    ]  
)  
?? learning_rate=0.1  
...
```

Двухслойная сеть, 4circles

...

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
--------------	--------------	---------

dense (Dense)	(None, 8)	24
---------------	-----------	----

dense_1 (Dense)	(None, 4)	36
-----------------	-----------	----

Total params: 60

...

Двухслойная сеть, 4circles

Epoch 1/30

60/60 loss: 0.1092 - accuracy: 0.9620

Epoch 2/30

60/60 loss: 0.0213 - accuracy: 0.9925

...

Epoch 29/30

60/60 loss: 0.0238 - accuracy: 0.9915

Epoch 30/30

60/60 loss: 0.0208 - accuracy: 0.9932

63/63 loss: 0.0128 - accuracy: 0.9950

Точность : 0.9950, время: 0.07 сек./эпоху

Двухслойная сеть, спираль-3

```
Y,X = data_load(1)
```

```
...
```

```
Model: "sequential"
```

```
-----  
Layer (type)      Output Shape      Param #  
=====
```

```
dense (Dense)      (None, 8)         24
```

```
-----  
dense_1 (Dense)    (None, 3)         27  
=====
```

```
Total params: 51
```

Двухслойная сеть, спираль-3

Epoch 28/30

45/45 loss: 0.0503 - accuracy: 0.9822

Epoch 29/30

45/45 loss: 0.0335 - accuracy: 0.9878

Epoch 30/30

45/45 loss: 0.0291 - accuracy: 0.9900

47/47 loss: 0.0436 - accuracy: 0.9893

Точность: 0.9893, время: 0.06 сек./эпоху

Напомню, что наш результат 0.5200.

Двухслойная сеть, спираль-5

Epoch 29/30

75/75 loss: 0.5401 - accuracy: 0.7689

Epoch 30/30

75/75 loss: 0.4971 - accuracy: 0.7879

79/79 loss: 0.4468 - accuracy: 0.8072

Точность:0.8072, время: 0.11 сек./эпоху

Двухслойная сеть, sky_data

Epoch 99/100

75/75 loss: 0.1990 - accuracy: 0.9505

Epoch 100/100

75/75 loss: 0.1932 - accuracy: 0.9512

79/79 loss: 0.1859 - accuracy: 0.9560

Точность:0.9560, время: 0.07 сек./эпоху

Двухслойная сеть, MNIST

```
N1 = 10  # нейронов в 1 слое  
nEpochs = 20  # количество эпох
```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 10)	7200
dense_1 (Dense)	(None, 10)	110
Total params: 7,310		

```
Epoch 20/20  
525/525 loss: 0.1960 - accuracy: 0.9434  
547/547 loss: 0.5982 - accuracy: 0.9213
```

Точность:0.9213, время: 0.74 сек./эпоху

MNIST, N1=20

N1 = 20 # нейронов в 1 слое
nEpochs = 20 # количество эпох

dense (Dense) (None, 20) 14400
dense_1 (Dense) (None, 10) 210
Total params: 14,610

Epoch 20/20

525/525 loss: 0.1300 - accuracy: 0.9666

547/547 loss: 1.1089 - accuracy: 0.9296

Точность:0.9296, время: 0.79 сек./эпоху

CIFAR, N1=20

```
dense (Dense)      (None, 20)  61460
dense_1 (Dense)    (None, 10)  210
Total params: 61,670
```

Epoch 20/20

```
90/90 loss:      1.4178, acc:      0.4966,
      val_loss: 1.6236, val_acc: 0.4329
```

Точность:0.4329, время: 1.07 сек./эпоху

validation_data

```
model.fit(x=X_train, y=Y_train, epochs=nEpochs,  
          batch_size=500, validation_data=(X_test, Y_test))
```

Epoch 1/20

loss:0.4258, acc:0.8846; val_loss:0.2598, val_acc: 0.9261

...

Epoch 20/20

loss:0.0762, acc:0.9758; val_loss:0.4016, val_acc: 0.9446

Функции активации

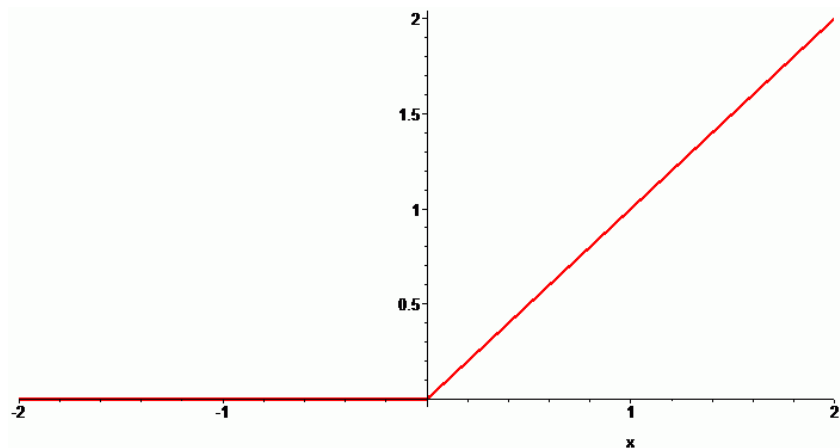
Различные способы вызова одной и той же функции:

```
tf.keras.layers.Dense(N1, activation = tf.nn.relu),  
tf.keras.layers.Dense(N1, activation = 'ReLU'),  
tf.keras.layers.Dense(N1,  
    activation = tf.keras.activations.relu),
```

Функции активации: ReLU

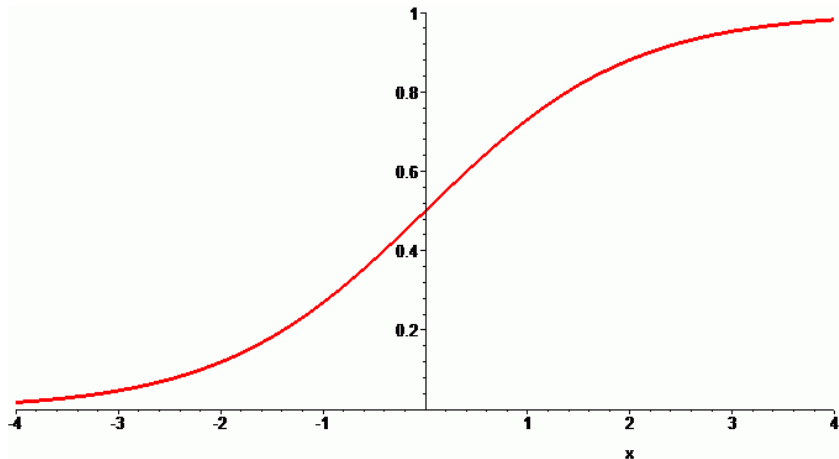
ReLU activation: $\max(x, 0)$

`keras.activations.relu(x)`



Функции активации: sigmoid

```
keras.activations.sigmoid(x)  
sigmoid(x) = 1 / (1 + exp(-x)).
```

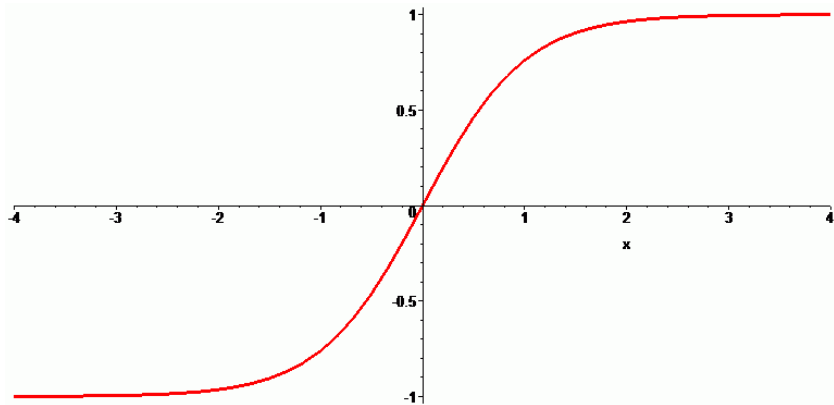


Функции активации: tanh

Тангенс гиперболический

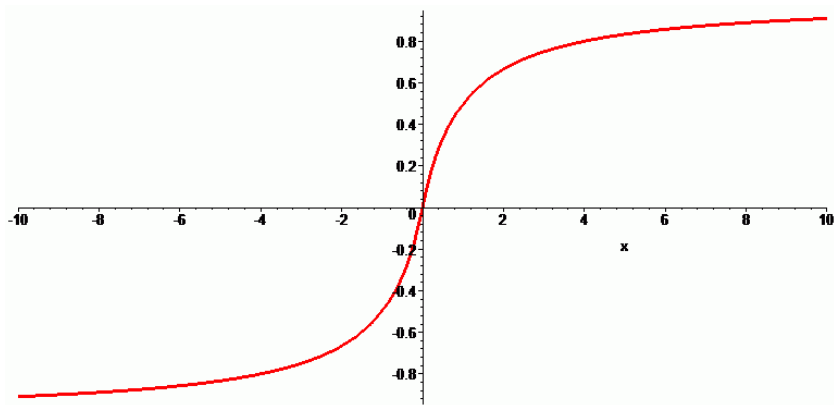
```
keras.activations.tanh(x)
```

$$\tanh(x) = ((\exp(x) - \exp(-x)) / (\exp(x) + \exp(-x))).$$



Функции активации: softsign

```
keras.activations.softsign(x)  
softsign(x) = x / (abs(x) + 1).
```



Задание

Задание. 1. Проверить, как влияет количество нейронов 1-го слоя на результат при всех входных данных.

2. Попробовать различный LR.

3. Попробовать различные оптимизаторы.

4. Попробовать различные функции активации.