Нейронные сети, классификация 1 слой
Результаты

С.И.Хашин
http://math.ivanovo.ac.ru/dalgebra/Khashin/index.html

Ивановский государственный университет

.

Иваново-2019

План

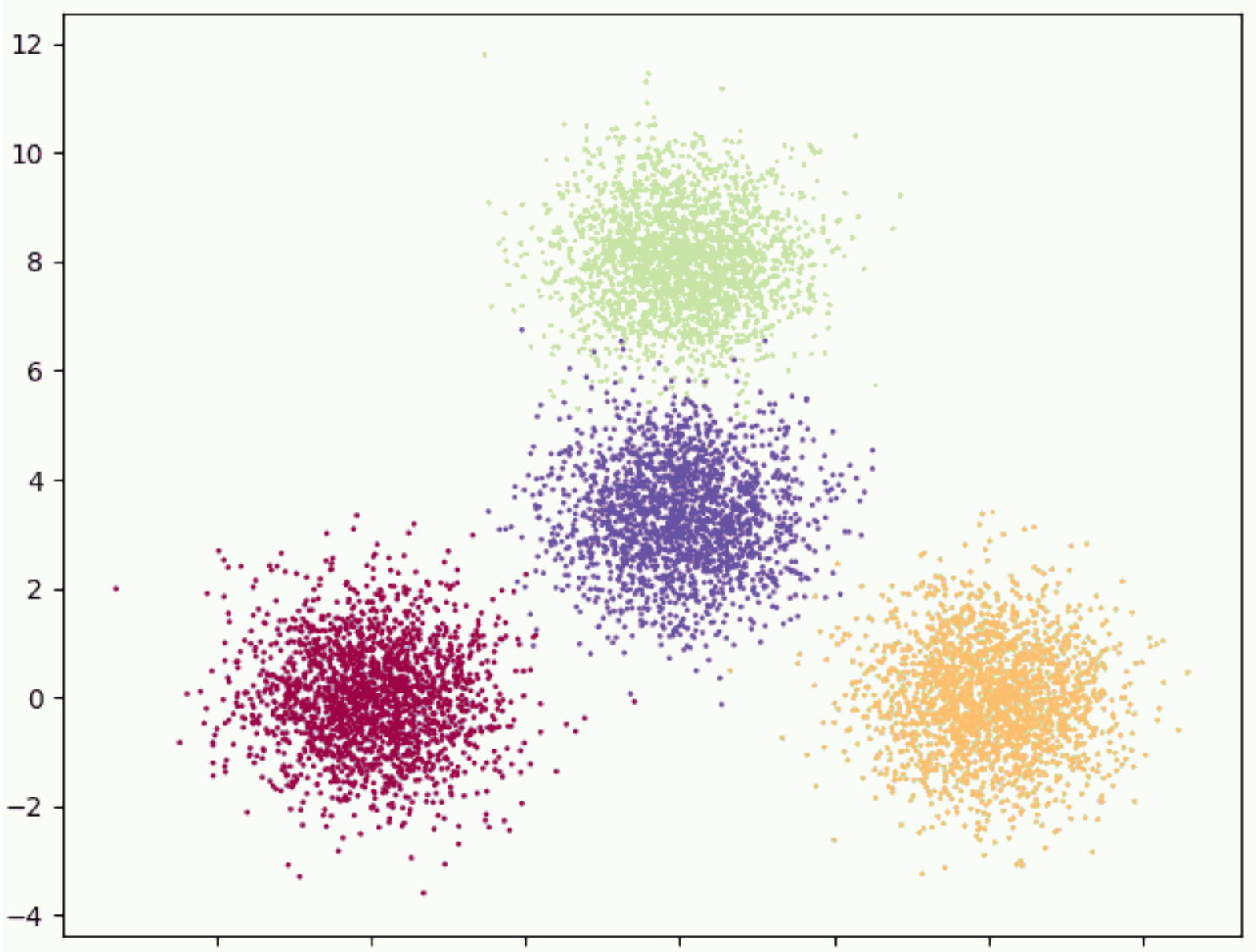
Результат 4 круга

$LR = 0.1$, начальная точность 0.183

Step	*acc*	*Step*	*acc*	*Step*	*acc*
5	0.240	70	0.823	200	0.977
10	0.332	80	0.840	220	0.981
15	0.388	90	0.863	240	0.984
20	0.414	100	0.883	260	0.986
25	0.636	110	0.902	280	0.988
30	0.646	120	0.920	300	0.989
35	0.629	130	0.934	320	0.990
40	0.610	140	0.946	340	0.990
45	0.607	150	0.953	360	0.991
50	0.660	160	0.960	380	0.991
60	0.794	180	0.971	390	0.991

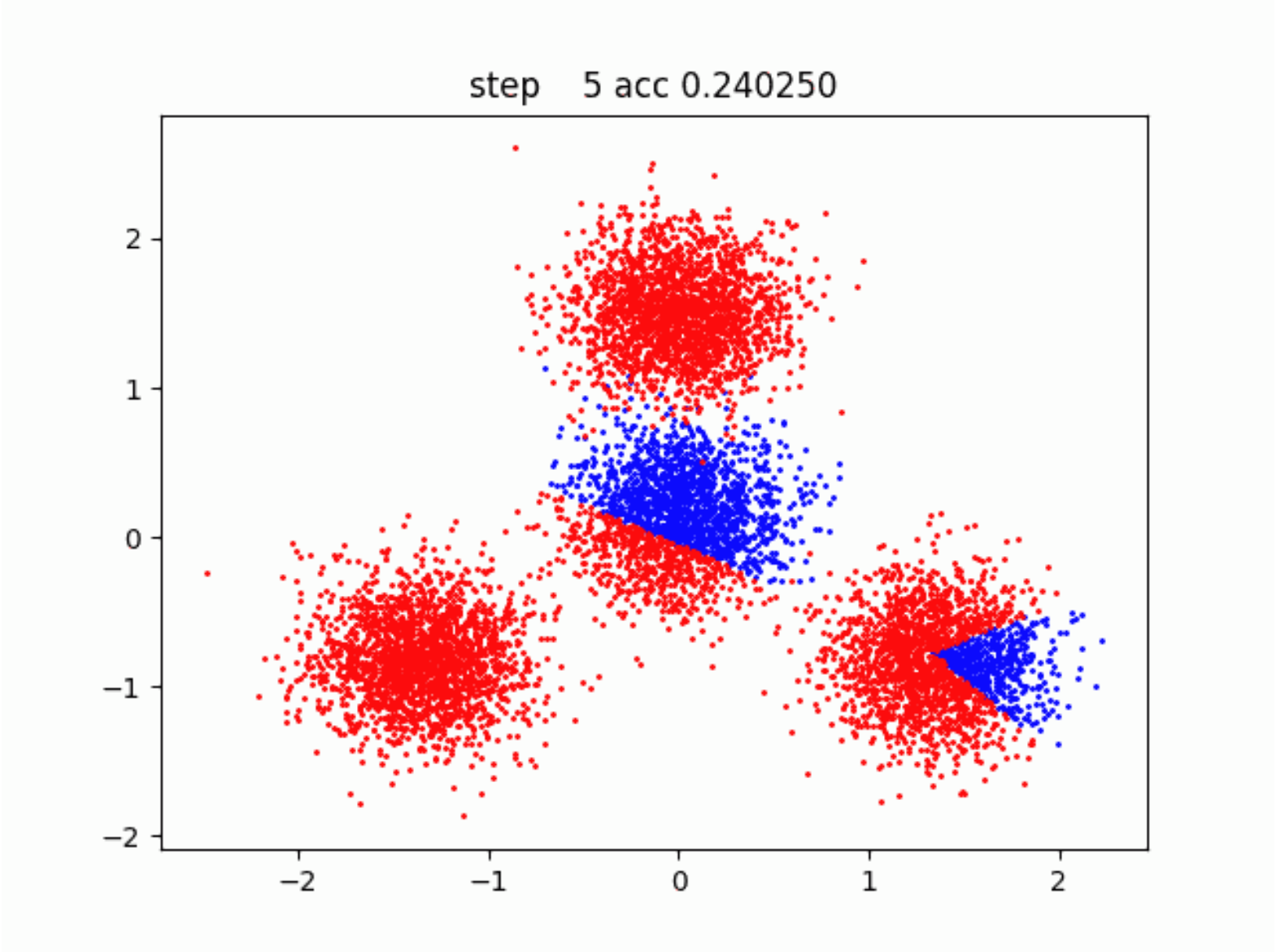
Результат 4 круга

Исходная картинка



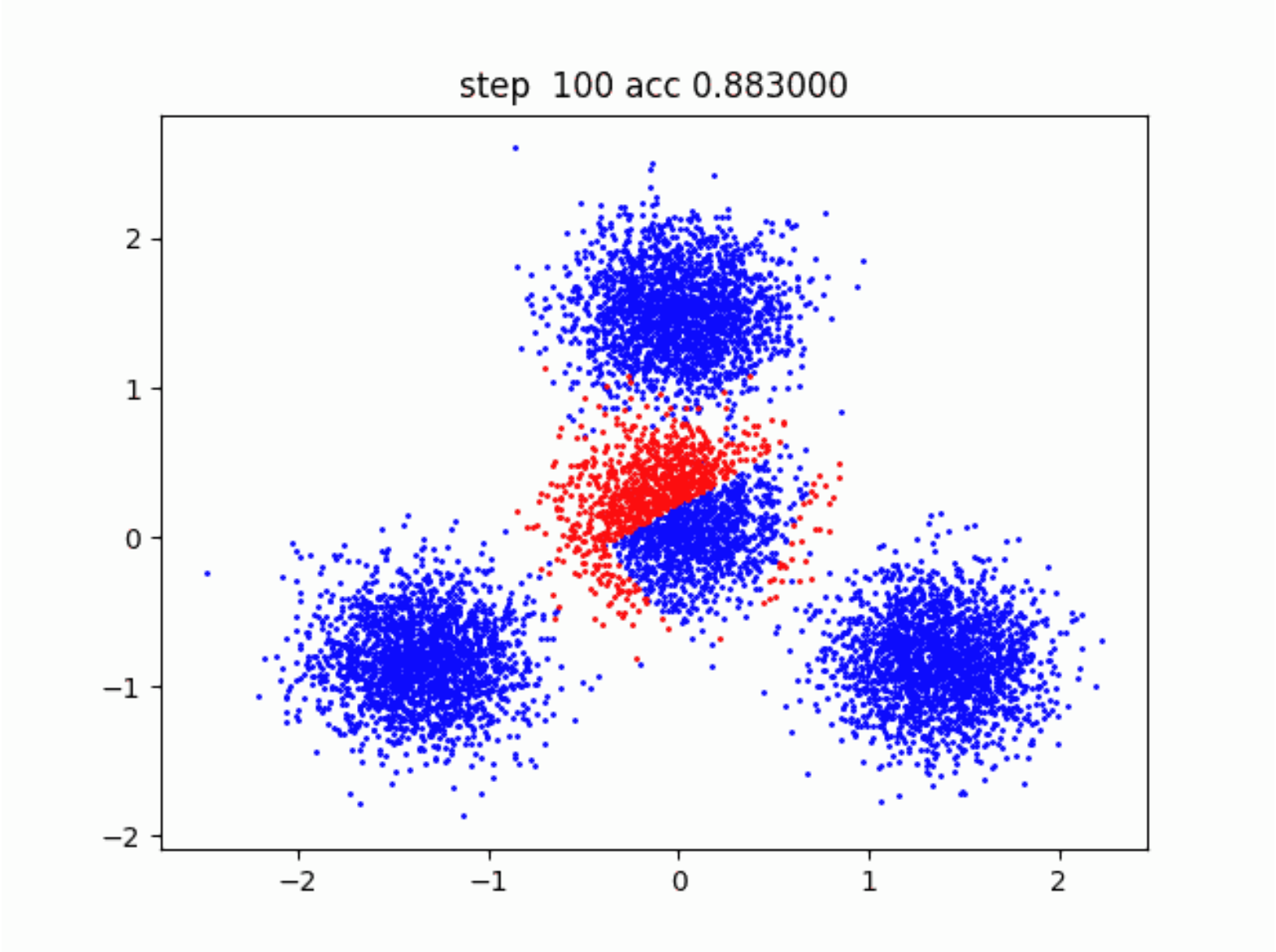
Результат 4 круга

Начало. Синие — OK, красные — неверно.



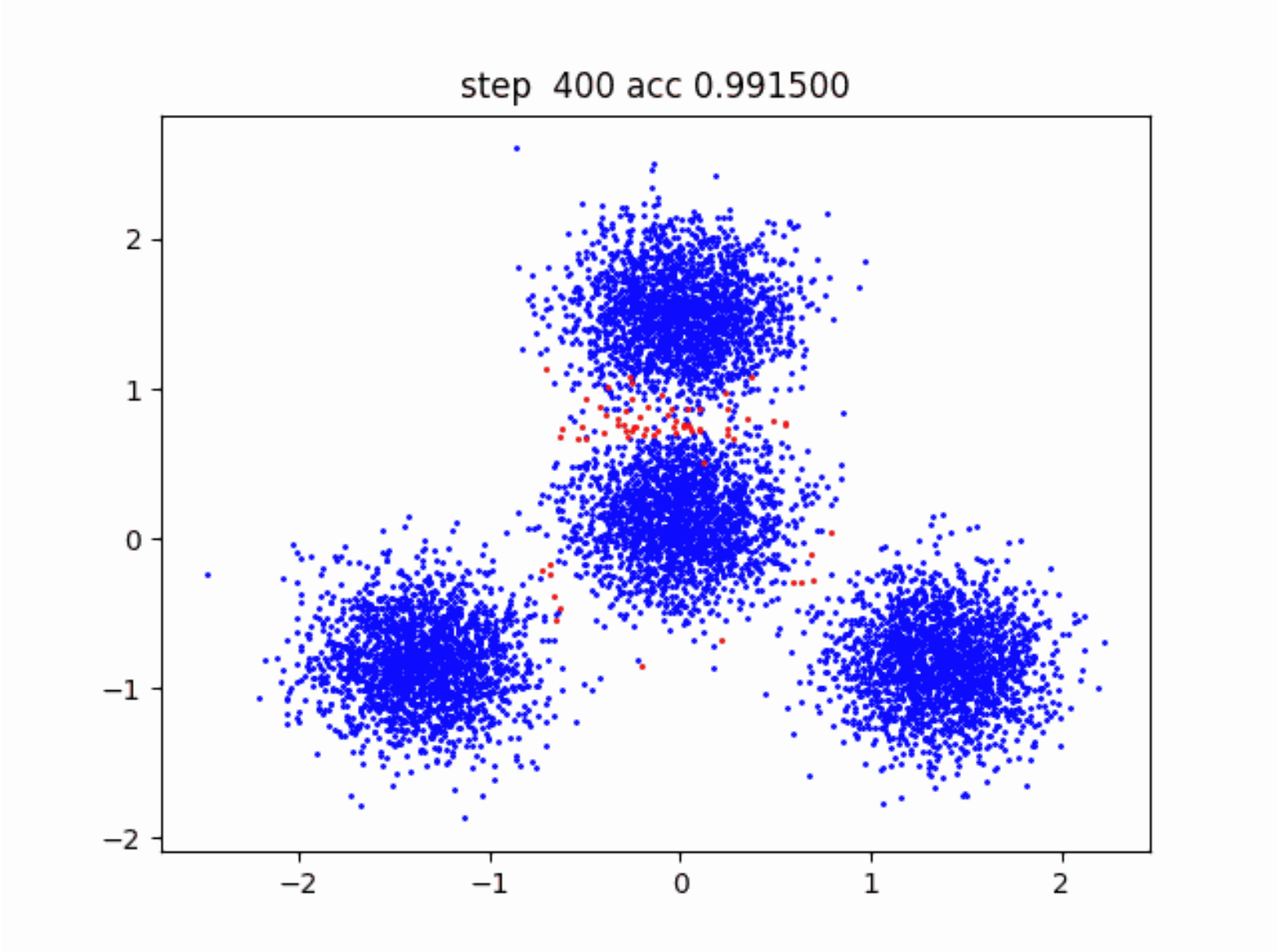
Результат 4 круга

Шаг 100. Синие — OK, красные — неверно.

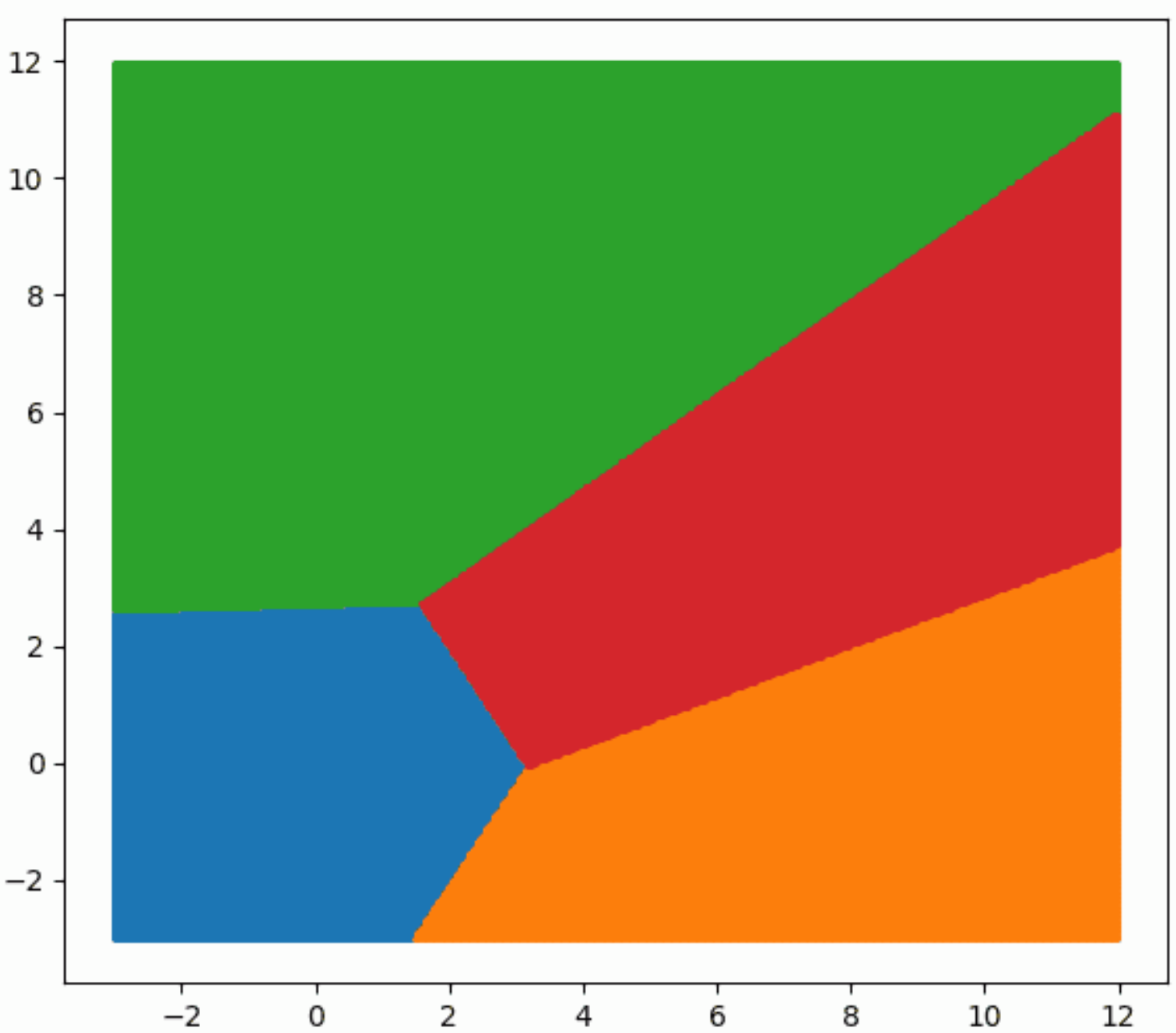


Результат 4 круга

Конец. Anime — anim_4circle.gif

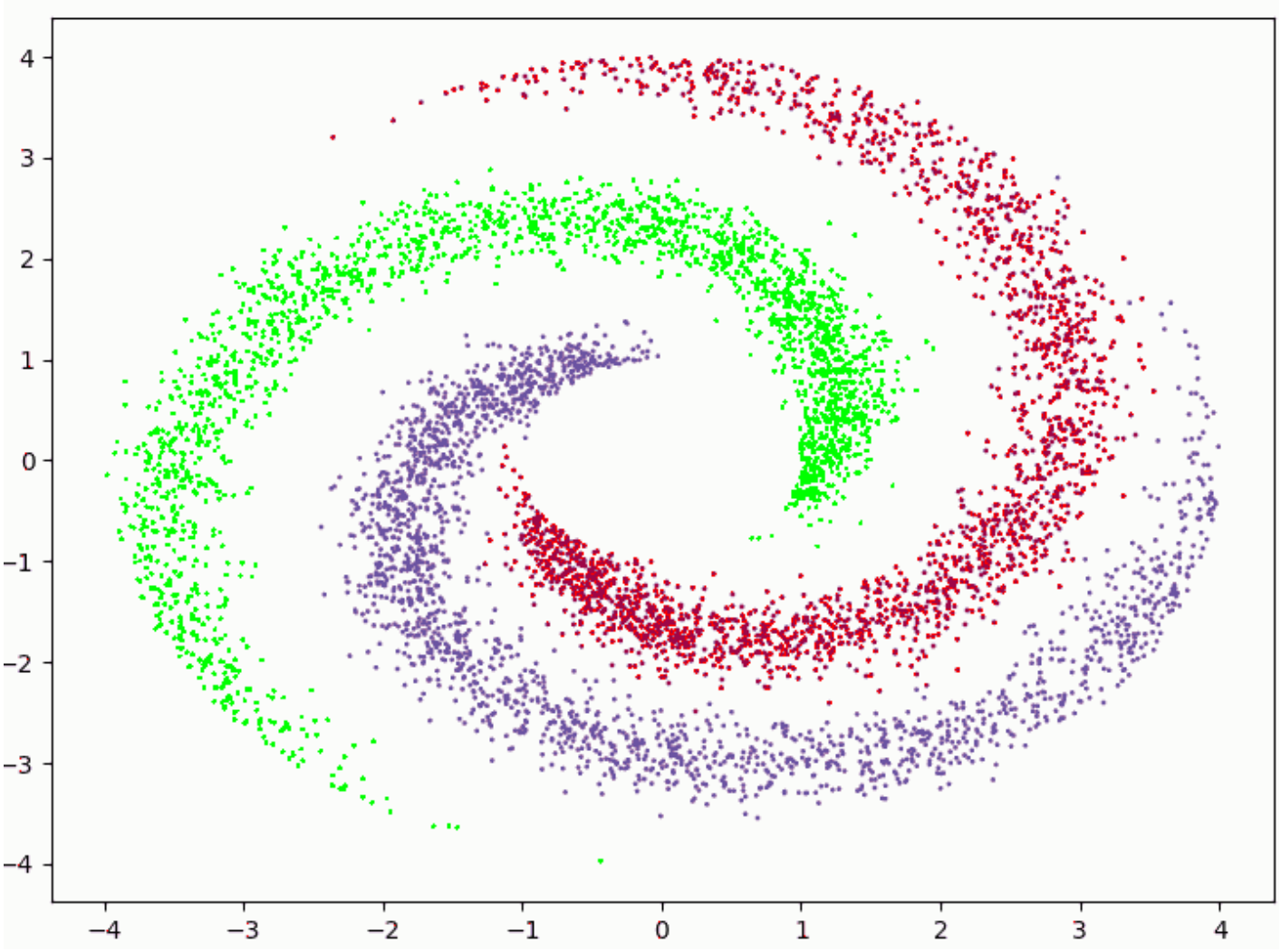


Многоугольники 4 круга



спираль-3

```
def tst_01(): ...
  Y,X = data_load(1)
```



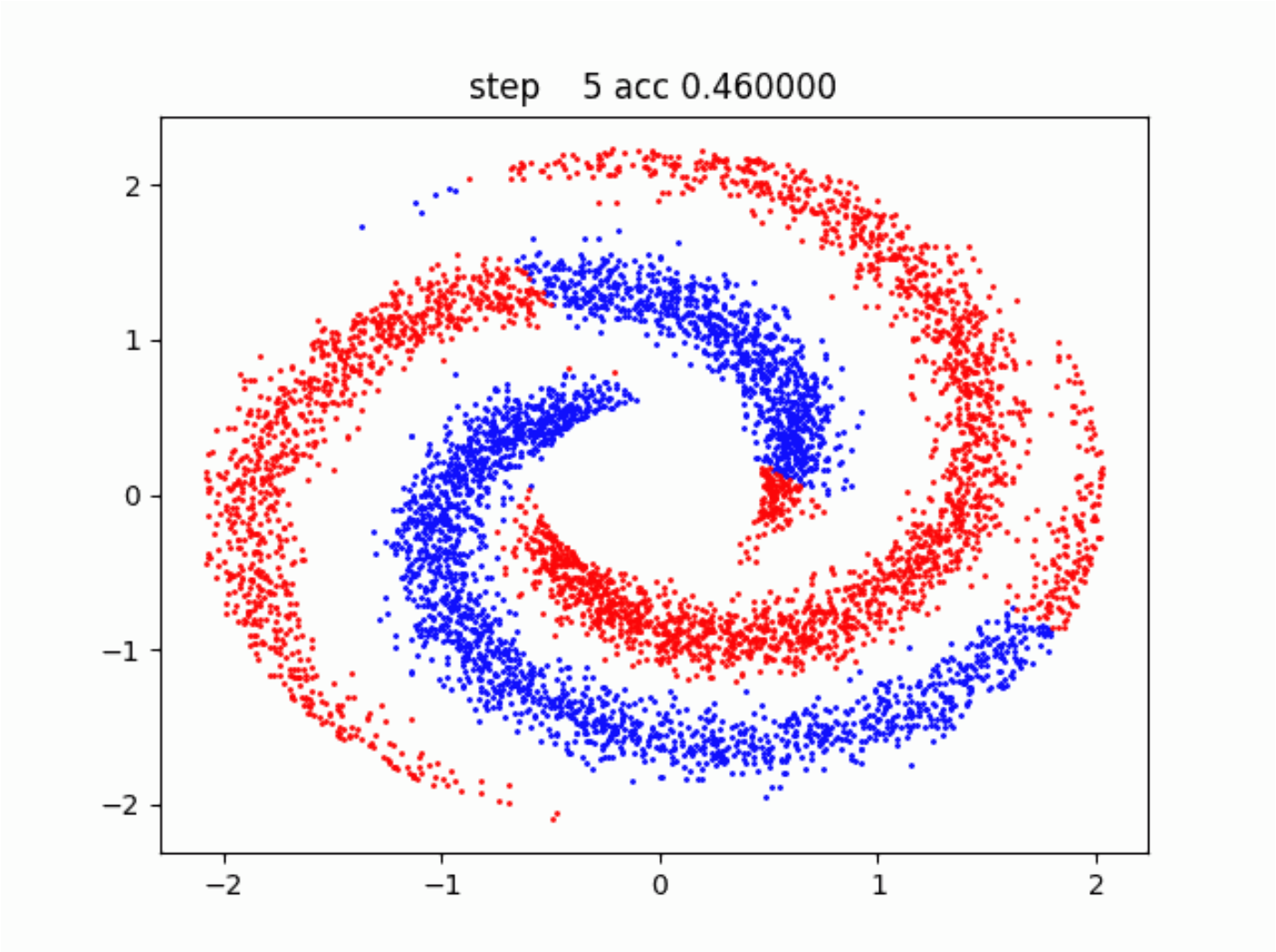
Результат Спираль-3

$LR = 0.1$, начальная точность 0.454

Step	*acc*	*Step*	*acc*	*Step*	*acc*
5	0.460	60	0.518	120	0.518
10	0.462	65	0.520	130	0.518
15	0.467	70	0.521	140	0.518
20	0.474	75	0.520	150	0.519
25	0.484	80	0.519	160	0.519
30	0.495	85	0.519	170	0.519
35	0.505	90	0.520	180	0.519
40	0.511	95	0.518	200	0.519
45	0.516	100	0.518	210	0.519
50	0.516	105	0.518	220	0.519
55	0.517	110	0.517	225	0.519

спираль-3, начало

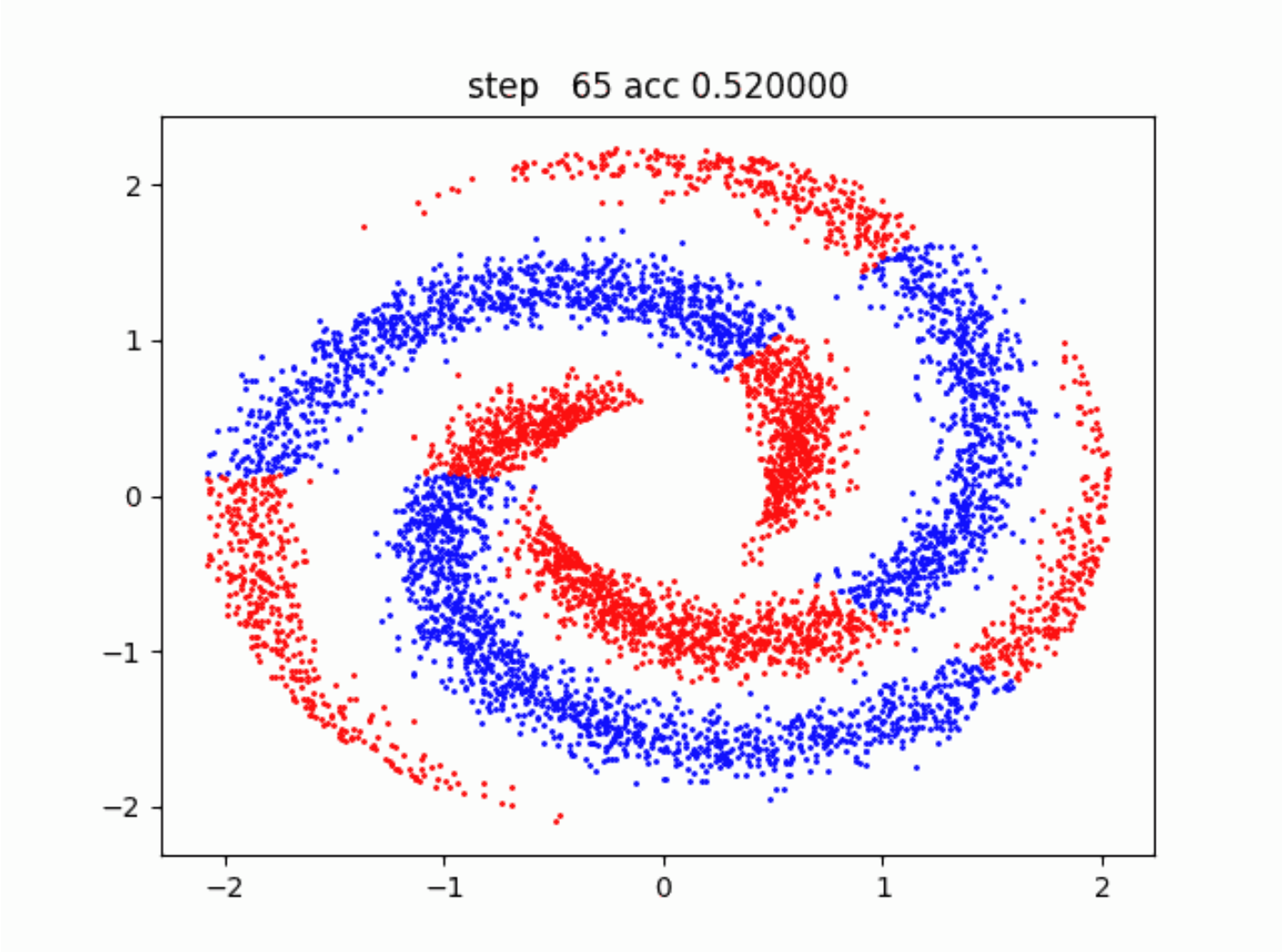
Синие — OK, красные — неверно.



спираль-3, конец

Анимация: /neuro/anim_spiral3/anim_spiral3.gif

Синие — OK, красные — неверно.



Результат astro

3 класса (Звезды, галактики, квазары).
$LR = 0.2$, начальная точность 0.380

Step	*acc*	*Step*	*acc*	*Step*	*acc*
10	0.4567	250	0.7719	900	0.8665
20	0.5600	300	0.7838	1000	0.8732
30	0.6109	350	0.7928	1100	0.8784
40	0.6396	400	0.8021	1200	0.8807
50	0.6610	450	0.8098	1300	0.8854
70	0.6865	500	0.8180	1400	0.8891
100	0.7120	550	0.8284	1500	0.8931
120	0.7250	600	0.8360	1600	0.8953
140	0.7357	650	0.8435	1700	0.8974
160	0.7438	700	0.8495	1800	0.8990
200	0.7574	750	0.8534	1900	0.8992

Результат MNIST

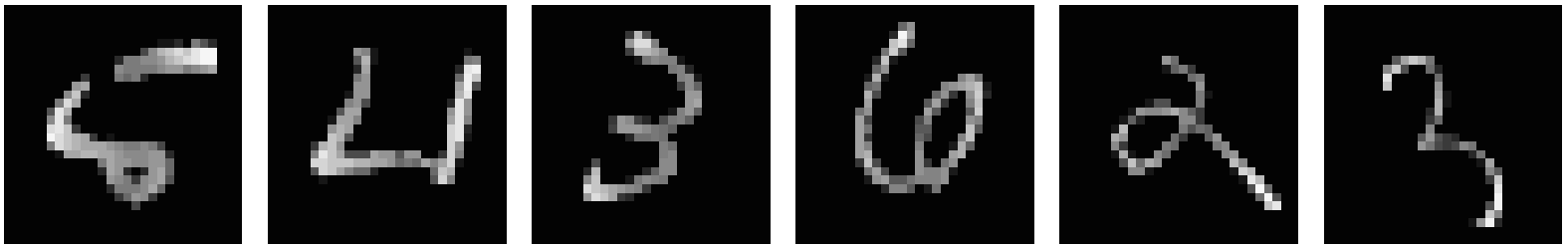
```
#W= np.random.normal(0, 1.,   size=(M,K+1))
W = np.random.normal(0, 0.01, size=(M,K+1))
```

$LR = 1.0$, начальная точность 0.089

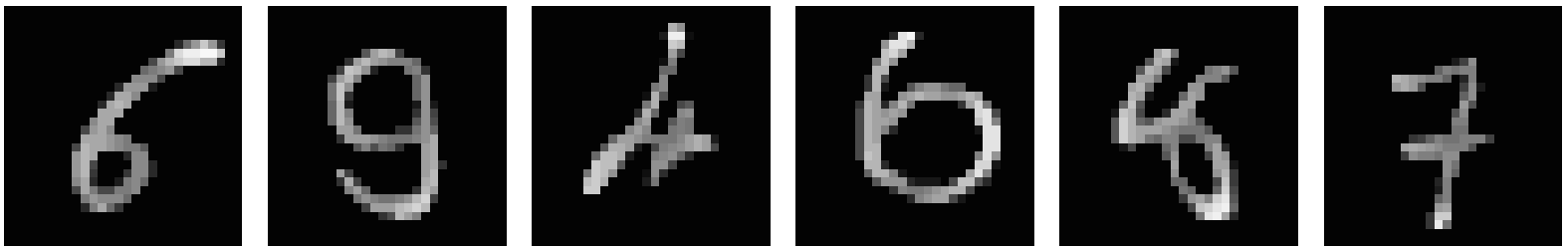
Step	*acc*	*Step*	*acc*	*Step*	*acc*
5	0.8870	70	0.9248	200	0.9315
10	0.9031	80	0.9258	250	0.9327
15	0.9101	90	0.9267	300	0.9335
20	0.9137	100	0.9274	400	0.9346
25	0.9159	110	0.9280	500	0.9355
30	0.9176	120	0.9284	600	0.9361
35	0.9192	130	0.9290	700	0.9368
40	0.9203	140	0.9293	800	0.9375
45	0.9213	150	0.9298	855	0.9378
50	0.9222	160	0.9300		
60	0.9235	170	0.9304		

MNIST, ошибки

5->6 4->6 3->2 6->1 2->9 3->5



6->5 9->8 4->2 6->0 8->4 7->9



CIFAR

Отличия:

```
W = np.random.normal(0, 1., size=(M,K+1))*0.1
LR =0.02
```

Результаты:

```
 0, acc=0.088983
 2, acc=0.129600, S= 6.21476
 4, acc=0.148433, S= 5.50583
 6, acc=0.160567, S= 5.09390
 8, acc=0.168483, S= 4.79937
10, acc=0.176567, S= 4.57317
20, acc=0.202533, S= 3.92928
30, acc=0.218733, S= 3.61173
40, acc=0.229867, S= 3.41045
50, acc=0.238650, S= 3.26547
52, acc=0.240117, S= 3.24084
```

train/test

Разбивка на тренировочные и проверочные данные:

```
def train_test(Y,X, dolya): '''
    Разбить обучающие данные (Y,X) не train, test
    :param Y: вектор верных ответов
    :param X: матрица обучающих векторов
    :param dolya: доля test, или размер test, если dolya>1
    :return: Y_train, X_train, Y_test, X_test  '''
    N = len(Y)
    a = np.arange(N)
    np.random.shuffle(a)
    Y,X = Y[a], X[a]
    if dolya<1: dolya = int(dolya*N)
    dolya = N-dolya
    Y_train, X_train = Y[:dolya], X[:dolya]
    Y_test,  X_test  = Y[dolya:], X[dolya:]
    return Y_train, X_train, Y_test, X_test
```

train/test

Замечание. Глобальные переменные для функций `f(w)`, `df(w)` обозначены `Y_work, X_work`. Не забыть им присвоить нужные значения перед нахождением gradient_descent ! Отметим, что присваивание

```
Y_work, X_work = Y, X
```

не создаёт новые объекты, а лишь ссылки на прежние, то есть память не расходутеся.

Задание. Сравнить на данных 4circle, sky_data, MNIST результаты на полном комплекте данных и на разбивке (train, test).

batches

Разбивка на блоки:

```
def get_batch(Y,X, bSize, iBatch):
    '''
    Извлечь блок № iBatch размер bSize
    :param Y: вектор верных ответов
    :param X: матрица обучающих векторов
    :param bSize: размер блока
    :param iBatch: номер блока
    :return:  Y_batch, X_batch
    '''
    return Y[bSize*iBatch:bSize*(iBatch+1)],\
           X[bSize*iBatch:bSize*(iBatch+1)]

    # Вызов:
    Yb, Xb = mlu.get_batch(Y, X, 1000, 7)
```

batches

Задание. Сравнить на данных 4circle, sky_data, MNIST результаты на полном комплекте данных и на разбивке по блокам подходящего размера.