Exercise 2: Ranking binary numbers with a perceptron

A **perceptron** takes two binary numbers of 10 digits each and returns either +1 or -1.

$\vec{S}$ has 20 elements (± 1). The first 10 elements are the **top** dataset n_t and the others are the **bottom** dataset n_b.

$$\sigma(\{S_i^\mu\} = \{\text{sgn}(\sum_i S_i J_i)\})$$

The goal of the perceptron is to **tell whether** $n_t > n_b$ **or viceversa**.

Step 1: Set up the perfect perceptron

Let us define the perfect perceptron as the function that given the following weights array

```
array([ 512.,  256.,  128.,   64.,   32.,   16.,    8.,    4.,    2.,
          1., -512., -256., -128.,  -64.,  -32.,  -16.,   -8.,   -4.,
         -2.,   -1.])
```

returns the dot product between the weights and the input array

$$\sigma = \vec{w} \cdot \vec{s}$$

which amounts to comparing the two numbers in binary representation.

Step 2: Define and evaluate a test error ϵ of the PP

We compare said number with the true value, which we choose to be the function such that it returns +1 if the integer comparison between n_t and n_b is such that $n_t \geq n_b$, and -1 otherwise.

In a sense, we are checking whether the comparison in decimal representation is the same as the comparison in binary representation.

$$\sigma_{\text{true}} = \text{sgn}(n_t - n_b)$$

except when $n_t = n_b$ this number yields +1.

Notice that I have inserted the definition for the **equality** case, otherwise it would get an error point due to the fact that the binary comparison rule does not account for the equality case.

```
inputs =
 [[0 1 0 ... 1 1 1]
 [0 0 0 ... 1 1 1]
 [1 0 0 ... 1 0 1]
 ...
 [0 1 0 ... 0 1 0]
 [1 0 0 ... 1 0 1]
 [1 1 0 ... 1 1 1]]

inputs is a matrix of shape (100, 20)
```

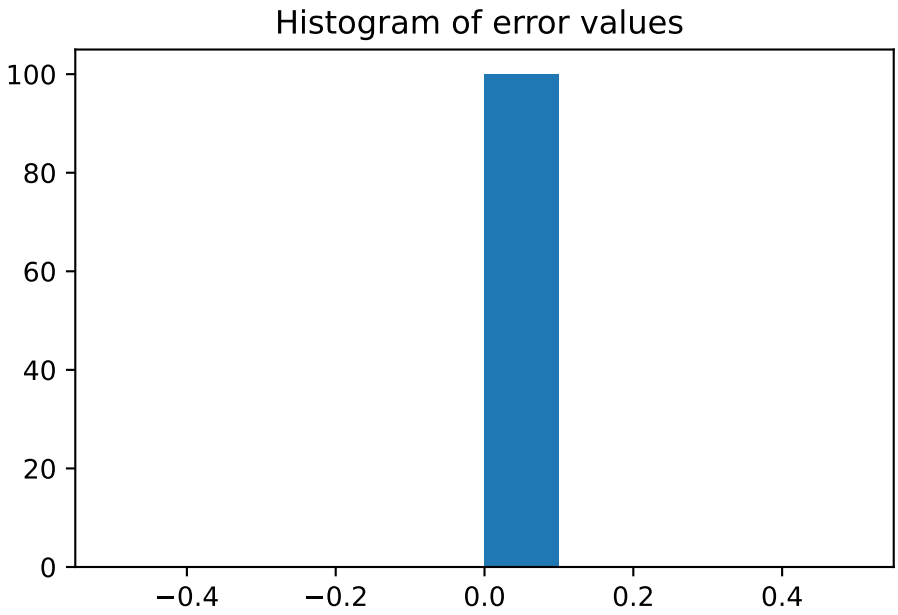
```
weights =
 [ 512.  256.  128.   64.   32.   16.    8.    4.    2.    1. -512. -256.
 -128.  -64.  -32.  -16.   -8.   -4.   -2.   -1.]

weights is a matrix of shape (20,)
```

We may define a trivial error measure such as

$$\epsilon = |\sigma - \sigma_{\text{true}}|$$

Let us compare, for example, 100 instances of this comparison.



We see that we have exactly error 0 on the 100 runs, each with 100 samples. The **perfect perceptron** is, in fact, perfect (with the caveat of the definition of the *border* situation).

Step 3: Assign random synaptic weights

Until now we had a *perfect perceptron*, meaning that we used a perceptron where we knew the model (the knowledge of the specific weights that would reproduce the comparison behaviour) that would trustfully yield the correct data.

Now let us show instead a real learning process, where we do not have any prior knowledge. Weights are, in fact, extracted from $\mathcal{N}(0, 1)$.

Our previous perceptron had 20 weights. Likewise, we find 20 Gaussian weights in this situation as well.

```
inputs =
 [[0 1 0 ... 1 1 0]
 [0 1 1 ... 0 0 1]
 [0 1 0 ... 1 1 1]
 ...
 [0 0 0 ... 1 0 0]
 [1 1 0 ... 1 1 0]
 [1 1 0 ... 1 1 0]]
```

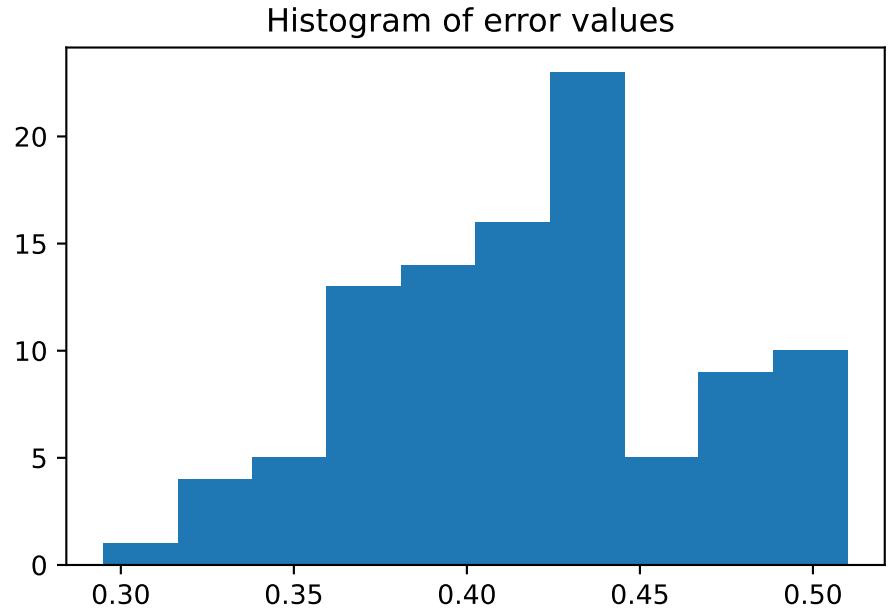
```
inputs is a matrix of shape (100, 20)


weights =
 [-0.61969378  2.32407073  1.42696834 -0.76700579 -1.66220087  1.14947293
  0.93453419 -0.37210047 -0.1156646   0.23855006 -1.05314595 -0.87653584
  0.40334093  0.96460106  0.48442217 -0.26801583  0.05586189 -1.03655349
 -1.18581587  1.10333308]


weights is a matrix of shape  (20,)
```

For now we have not done any learning, so I do not expect a good result. Instead, I expect that half of the times we get the right answer, and the other half of the times we get it wrong.

Let us repeat the evaluation with the errors as we have done in Step 2.



We notice that over many runs (100 runs), each made of 100 samples, the (relative) error is centered around 0.5, meaning that the hypothesis that about half of the times we get it right and half wrong was correct.

Step 5: Produce $\vec{\xi}^{\mu}$ for learning

Now we want to iterate through a wide dataset and see whether we learn anything. In order to do this we should produce a **training set**.

We want two datasets, dataset 1 and 2, respectively containing $P_1 = 500$ and $P_2 = 2000$ points, containing two numbers (the x, encoded as a vector 20 binary digits long) and a value ± 1 (the y). We can use the decimal conversion as well to create the dataset, since we know that the perfect perceptron coincides with the integer comparison from Step 2.

We should however redefine the comparison, remember the caveat about the $n_t = n_b$ case. We choose to include it with the $>$ case.

Step 6.1: Training with the first set

Let us again initialise some random weights.

```
weights =
 [-1.61932394  0.68174851  1.44773056  1.42566251  0.06645733  0.53507445
 -0.11442036 -0.83043288 -0.29917361 -1.04975285  0.46379727 -0.2422248
  1.92278145  0.13165595  0.57301707 -0.09830948 -1.98946665 -1.21883104
 -0.57527749 -0.57672051]


weights is a matrix of shape  (20,)


2.0
2.0
0.0
2.0
0.0
2.0
0.0
2.0
2.0
0.0
0.0
0.0
2.0
0.0
2.0
0.0
2.0
2.0
0.0
0.0
2.0
```

2.0
2.0
2.0
0.0
2.0
0.0
0.0
0.0
2.0
0.0
0.0
2.0
0.0
2.0
2.0
2.0
2.0
0.0
0.0
0.0
0.0
2.0
0.0
2.0
0.0
0.0
0.0
0.0
2.0
0.0
2.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
2.0
2.0
0.0
0.0
2.0

2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
2.0
0.0
0.0
0.0
0.0
2.0
2.0
0.0
2.0
0.0
2.0
0.0
2.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
2.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0

0.0
2.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
2.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
2.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0

0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
2.0
2.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
2.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0

```
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
2.0
0.0
0.0
0.0
2.0
```

0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
2.0
0.0
2.0

0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0

0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
2.0
2.0
2.0
0.0
0.0
0.0
0.0
2.0
0.0

0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
2.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
2.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0

0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
2.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0

0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0

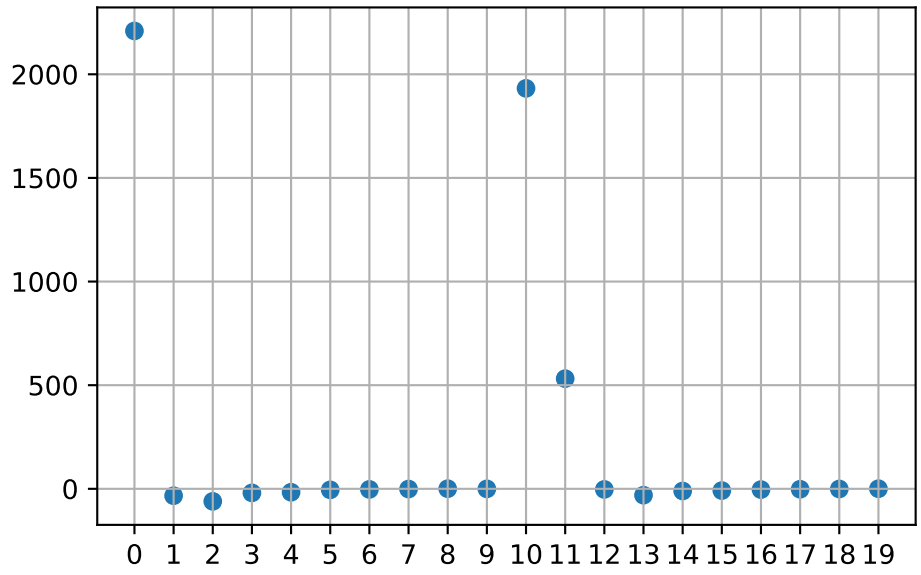
```
0.0
0.0
0.0
0.0
0.0
0.0
```

Let's see our weights after they have been updated during the training phase.

```
weights =
 [ 2.85281202  0.9053553   0.55330337  0.75484212  0.29006413  0.08786085
  0.10918644  0.06399431 -0.07556681  0.06828114 -2.66669789 -2.03107919
 -0.98410692 -0.53916444 -0.09780332 -0.32191628 -0.20061227 -0.32440385
 -0.1280639   0.54131348]


weights is a matrix of shape  (20,)

array([ 2.20904920e+03, -3.32461559e+01, -6.04725975e+01, -1.96018931e+01,
       -1.65736257e+01, -4.93234761e+00, -2.79092277e+00, -1.01518334e+00,
        5.63139506e-01, -2.64409764e-01,  1.93204356e+03,  5.31523472e+02,
       -2.91144760e+00, -3.07523091e+01, -1.04969496e+01, -8.42257921e+00,
       -3.71938074e+00, -2.10750538e+00, -7.59435371e-01,  4.79313479e-01])
```



VECTOR with P2

```
weights =
 [ 0.84717374 -1.5335621  -2.98884526  0.60434595  0.03599071 -1.00753271
  0.45909699 -0.44191476 -0.48797383 -0.26144605  0.50080167  0.09076779
  0.08192364  2.01422203 -1.34817187 -0.26049835  0.05922024  0.15077804
  2.69429259 -0.90451634]


weights is a matrix of shape  (20,)


0.0
0.0
2.0
2.0
0.0
2.0
2.0
0.0
2.0
2.0
0.0
2.0
0.0
0.0
2.0
2.0
0.0
0.0
0.0
2.0
2.0
2.0
0.0
2.0
0.0
2.0
2.0
2.0
2.0
0.0
0.0
2.0
0.0
```

```
2.0
0.0
0.0
2.0
0.0
0.0
2.0
0.0
2.0
2.0
2.0
0.0
2.0
0.0
2.0
2.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
2.0
2.0
0.0
2.0
0.0
2.0
0.0
2.0
2.0
2.0
0.0
0.0
2.0
0.0
0.0
0.0
2.0
0.0
0.0
2.0
```

```
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
2.0
2.0
2.0
2.0
2.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
2.0
2.0
0.0
2.0
0.0
```

0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
2.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
2.0
0.0
0.0
0.0
0.0
2.0
0.0
2.0
2.0

0.0
2.0
0.0
0.0
0.0
0.0
0.0
2.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
2.0
0.0
2.0
0.0
2.0
0.0
2.0
0.0
0.0
2.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0

0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
2.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0

2.0
0.0
0.0
2.0
0.0
2.0
0.0
0.0
2.0
0.0
2.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0

0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0

0.0
0.0
0.0
0.0
2.0
0.0
2.0
2.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
2.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0

0.0
0.0
0.0
0.0
2.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0

0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0

0.0
0.0
0.0
2.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0

0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0

0.0
0.0
0.0
2.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0

0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
2.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0

0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
2.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
2.0
0.0
2.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0

```
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
```

2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0

0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0

0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0

0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
2.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0

0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0

0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0

0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
2.0
0.0
2.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0

0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
2.0

0.0
2.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
2.0
0.0
0.0
2.0
0.0

0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0

0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0

```
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
```

0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
2.0
0.0
0.0
2.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0

```
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
```

0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
2.0
0.0
2.0
0.0
0.0
2.0

0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0

```
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
```

0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0

0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0

0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0

2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0

0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0

0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0

0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
2.0
2.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0

0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0

2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0

0.0
0.0
2.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0

0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0

0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0

```
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
0.0
0.0
2.0
0.0
0.0
2.0
0.0
0.0
```

Let's see our weights after they have been updated during the training phase.

```
weights =
 [ 4.42488251  2.04414666  1.2596839   0.38073915  0.25959751  0.11050128
  0.45909699  0.45251243 -0.26436703 -0.26144605 -4.86576147 -2.81612058
 -1.03611035 -0.44545274 -0.23013788 -0.26049835 -0.16438656  0.15077804
  0.23461781 -0.01008915]
```

```
weights is a matrix of shape  (20,)
```