1) Чтобы определить оптимальное количество итераций, будем экспериментировать с различными значениями. Попробуем запускать модель на следующих итерациях: 100, 1000, 10000, 100000 и 200000, чтобы посмотреть, как быстро модель сходится (или перестает изменять веса существенно).

Результат для 100 итераций:

```
[[0.58872346]
 [0.44498862]
 [0.20998491]]     [0.65813036]
результат на обучающем наборе
[[0.91987109]
 [0.0210229 ]
 [0.8564842 ]
 [0.11359796]]
предсказание для   [1, 1, 0]
[0.98578438]
```

Результат для 1000 итераций:

```
[[0.58872346]
 [0.44498862]
 [0.20998491]]     [0.65813036]
результат на обучающем наборе
[[0.97661818]
 [0.00150721]
 [0.95850341]
 [0.03369193]]
предсказание для   [1, 1, 0]
[0.99896569]
```

Результат для 10000 итераций:

```
[[0.58872346]
 [0.44498862]
 [0.20998491]]     [0.65813036]
результат на обучающем наборе
[[9.92827375e-01]
 [1.31014848e-04]
 [9.87466453e-01]
 [1.02179865e-02]]
предсказание для   [1, 1, 0]
[0.99990832]
```

Результат для 100000 итераций:

```
 [[0.58872346]
  [0.44498862]
  [0.20998491]]     [0.65813036]
 [[ 5.39639657]
  [ 4.83804492]
  [-9.88122351]]     [0.00641183]
 [[  5.7469308 ]
  [  5.19121984]
  [-10.58685421]]     [0.00451586]
 [[  5.95141653]
  [  5.39687881]
  [-10.99785783]]     [0.00368065]
 результат на обучающем наборе
 [[9.97753857e-01]
  [1.25118368e-05]
  [9.96098381e-01]
  [3.18418867e-03]]
 предсказание для   [1, 1, 0]
 [0.99999118]
```

Результат для 200000 итераций:

```
[[0.58872346]
 [0.44498862]
 [0.20998491]]     [0.65813036]
[[ 5.39639657]
 [ 4.83804492]
 [-9.88122351]]     [0.00641183]
[[  5.7469308 ]
 [  5.19121984]
 [-10.58685421]]     [0.00451586]
[[  5.95141653]
 [  5.39687881]
 [-10.99785783]]     [0.00368065]
[[  6.09630208]
 [  5.54246453]
 [-11.28884312]]     [0.00318416]
[[  6.20858637]
 [  5.65522692]
 [-11.5142414 ]]     [0.00284593]
[[  6.30027294]
 [  5.74726653]
 [-11.6982275 ]]     [0.00259658]
[[  6.37775681]
 [  5.82502485]
 [-11.85367193]]     [0.00240296]
результат на обучающем наборе
[[9.98413832e-01]
 [6.21688025e-06]
 [9.97247079e-01]
 [2.24702159e-03]]
предсказание для   [1, 1, 0]
[0.99999561]
```

Поскольку у нас здесь используется небольшой тренировочный набор данных, содержащий всего 4 образца, 100000 итераций – это слишком много. Так, при увеличении числа итераций примерно с 10000 (т.е. примерно столько итераций будет для нас оптимально для максимально разумной точности) увеличивается и время

работы программы, при этом веса изменяются совсем незначительно, и точность предсказания уже почти не меняется, при этом оставаясь достаточно высокой для нас.

2) Уменьшая тренировочный набор, мы уменьшаем количество информации для обучения, из чего следует, что итоговая модель будет работать менее качественно. При увеличении тренировочного набора качество обучения улучшается (при этом данные нужно добавлять осмысленно).

Если данные уменьшаются, разумно снизить количество итераций, тогда как для больших наборов данных – увеличить.

3) Если добавить вектор, который уже существует, но с противоположным значением, то данные станут противоречивыми, и нейрон будет пытаться найти среднее значение или просто начнет колебаться, не сходясь к какому-то однозначному решению. Если добавить новый вектор с неверной меткой, нейрон будет пытаться «притянуть» вес к этому ошибочному ответу, что снизит точность модели.

4) Этот нейрон, по сути, является линейным разделителем, так как он использует линейную комбинацию входных данных. Он будет правильно классифицировать функции, которые являются линейно разделимыми, например AND, но не сможет корректно обучиться для XOR (исключающее «или»), так как оно не является линейно разделимым.

5) Для замены активационной функции можно попробовать, например, ReLU (Rectified Linear Unit). Так, влияние замены можно будет оценить по скорости сходимости и конечным результатам.

6) Добавление bias улучшает гибкость модели, ибо помогает смещать границу разделения. Для добавления bias можно добавить ещё один элемент к вектору входов (например, [1]) и весу.

Чтобы модель могла выдавать корректное значение для [0, 0, 0], нужно либо обучить ее на этом значении, либо добавить bias, который поможет сдвигать значения выхода для нашего случая.