Отчет о выполненном задании:
пункт 2.

Зверев Дмитрий, 325-я группа АЯ ВМК МГУ

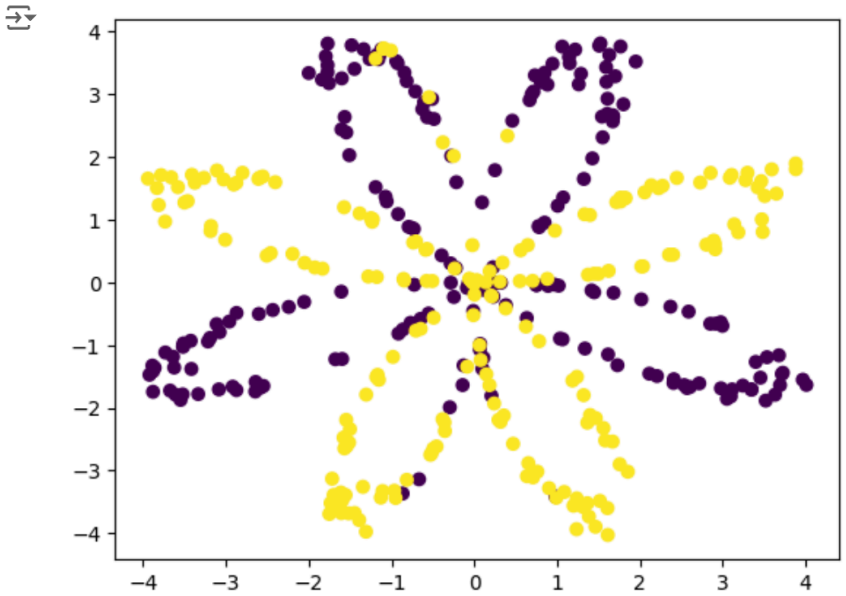
Формализация задачи:
Наша задача заключается в доработке нейронной сети из предоставленного ноутбука *perceptron* для предотвращения переобучения путем добавления L2-регуляризации в функцию потерь.
Основные цели:

- реализовать регуляризацию с параметром *lmdb*;
- модифицировать обратный ход распространения ошибки;
- провести анализ изменений и оценить их влияние на результат.

Подготовка входных данных:
Входные данные представляют собой набор двумерных точек, которые нужно разбить на 2 класса для бинарной классификации.

Подбор обучающей выборки:
Выборка заранее размечена для бинарной классификации (2 класса, представленных цветами). Для обучения использовалось 200 точек.

```
[ ]  # Создаем и рисуем датасет в форме цветочка с 200 объектами, цветами обозначены метки классов
     X, Y = create_flower(400, 4) #данная функция определена ранее
     plt.scatter(X[0, :], X[1, :], c=Y);
```



Выбор архитектуры сети:

Количество входных нейронов фиксировано и равно 2, т.к. модель работает с точками в двумерном пространстве; на выходе у нас 1 нейрон, поскольку у нас задача бинарной классификации. Значение на выходе понимается как вероятность принадлежности к одному из двух классов.

Количество нейронов в скрытом слое варьируется от случая, которых у нас 3:

1) 1 нейрон в скрытом слое. В этом случае модель может различать только линейно разделимые данные и точки разделяются одной прямой (что недостаточно для сложных границ разделения);
2) 4 нейрона в скрытом слое. Такой подход позволяет представлять более сложные границы разделения за счет нелинейности. Это подходящая архитектура для небольшой задачи, так как модель становится достаточно гибкой;
3) 40 нейронов в скрытом слое. В данном случае модель становится избыточно сложной для текущей задачи, и границы разделения становятся чересчур адаптированными к обучающим данным, что приводит к переобучению.

   Для решения проблемы переобучения будем использовать параметр регуляризации λ.

Отметим, что у нас полносвязная архитектура, то есть каждый нейрон предыдущего слоя соединен с каждым нейроном следующего слоя.

Функция активации:

- для скрытого слоя используется гиперболический тангенс (tanh);
- для выходного слоя используется сигмоидная функция (sigmoid).

```
# Функция активации нейрона в выходном слое и ее производная
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
def dsigmoid(y):
    return y * (1 - y)

# Функция активации нейронов во внутреннем слое и ее производная
def tanh(x):
    return np.tanh(x)
def dtanh(y):
    return 1 - y * y
```

Метод обучения

В исходной версии инициализация весов выполнялась с использованием малых случайных значений. Градиентный спуск использовался для обновления весов, включая обратное распространение ошибки.

Так, для решения нашей задачи мы добавили регуляризация в функцию потерь и вычисление градиентов.

Функция потерь:

```
# Функция потерь (MSE с регуляризацией)
mse_loss = (np.sum((A2 - self.Y) ** 2) + (lmdb / (2 * self.m)) * (np.sum(self.W1**2) + np.sum(self.W2**2))) / self.m
self.mse_loss.append(mse_loss)
```

Обратное распространение ошибки:

```
# Обратное распространение
dA2 = (A2 - self.Y) * A2 * (1 - A2)
dW2 = np.dot(dA2, A1.T) + (lmdb / self.m) * self.W2
db2 = np.sum(dA2, axis=1, keepdims=True)
dA1 = np.dot(self.W2.T, dA2) * (1 - A1**2)
dW1 = np.dot(dA1, self.X.T) + (lmdb / self.m) * self.W1
db1 = np.sum(dA1, axis=1, keepdims=True)
```

Методы проверки правильности:

- оценка качества по значению функций потерь;
- визуализация границ разделения на графике.

Оценка правильности решения:

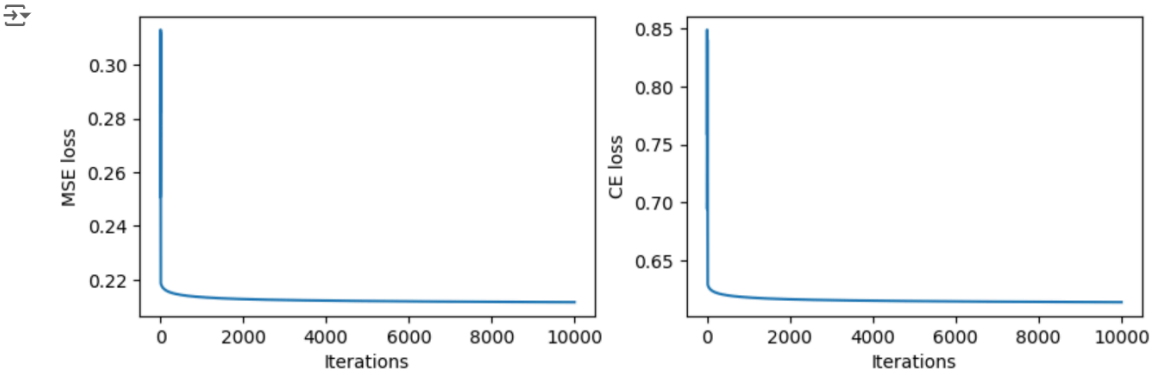
Ранее модели с 1, 4 и 40 нейронами демонстрировали разные результаты:

1 нейрон:

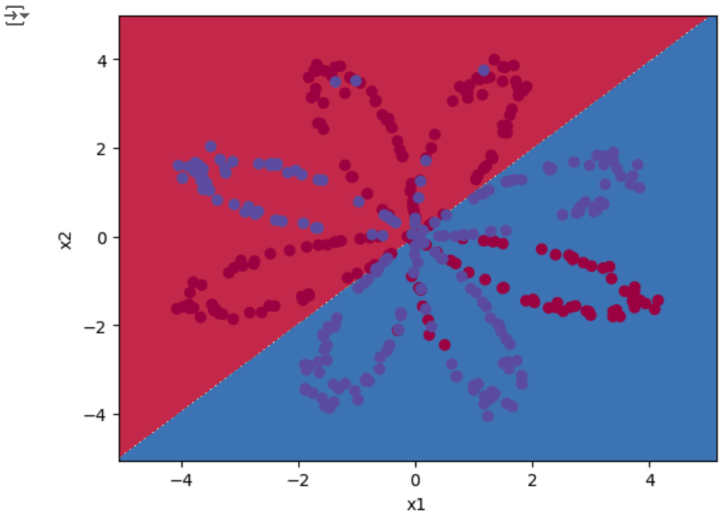
```
[ ]  perceptron = Perceptron(X, Y, 1)
     perceptron.learn(10000, 0.1)
```

```
MSE loss: 0.28267864907497886, CE loss: 0.7593307589469521
MSE loss: 0.21343946811138323, CE loss: 0.6183240365021538
MSE loss: 0.2127216917546379, CE loss: 0.6167793514787666
MSE loss: 0.2123696291276248, CE loss: 0.6160175005320963
MSE loss: 0.21214831247262386, CE loss: 0.6155379087595675
MSE loss: 0.21199093132739988, CE loss: 0.6151969814096562
MSE loss: 0.21187040151669476, CE loss: 0.6149361681555332
MSE loss: 0.21177343883971744, CE loss: 0.6147266327957271
MSE loss: 0.2116926779024539, CE loss: 0.6145523432563554
MSE loss: 0.21162366498410903, CE loss: 0.6144035897946364
```

```
[ ]  perceptron.plot() #данная функция определена ранее
```



```
[ ] # Строим разделяющую  прямую, показывающую результат обучения нейросети
    plot_division(perceptron) #данная функция определена ранее
```



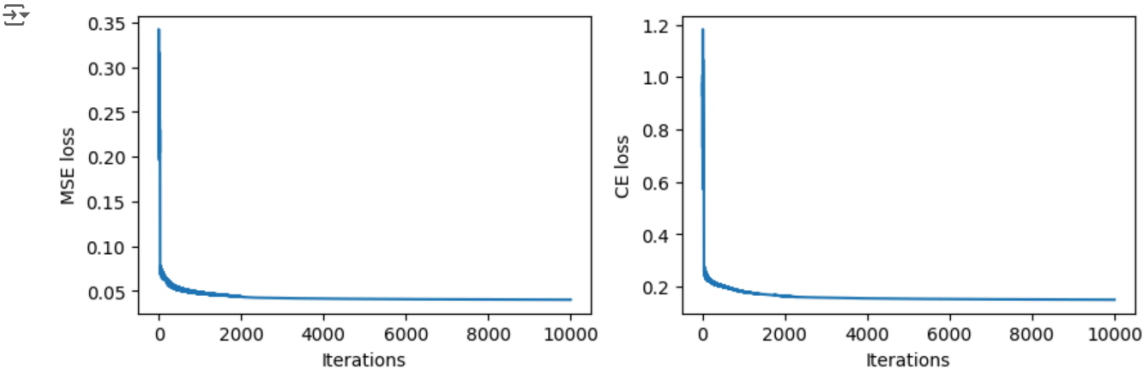
Как мы видим, 1 нейрона недостаточно для решения задачи.

4 нейрона:

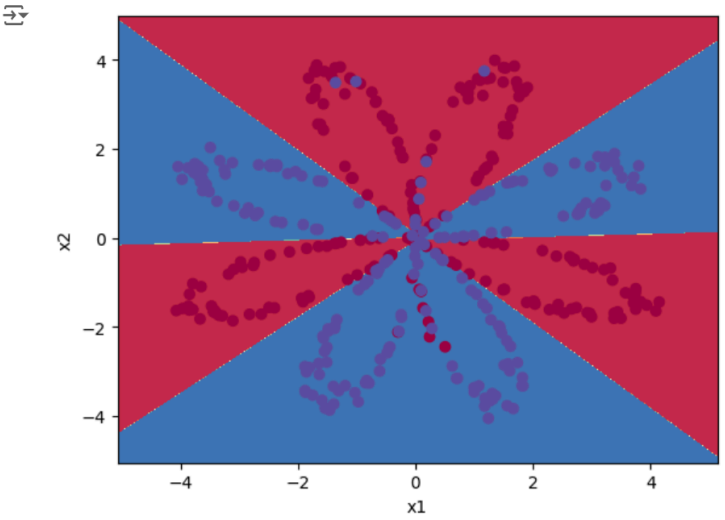
```
[ ] perceptron = Perceptron(X, Y, 4)
    perceptron.learn(10000, 0.1)
```

```
MSE loss: 0.26838860061217223, CE loss: 0.735288274443412
MSE loss: 0.047153196816007305, CE loss: 0.18035407161367037
MSE loss: 0.04357836955159877, CE loss: 0.16457070013523045
MSE loss: 0.042042653396937524, CE loss: 0.15831354246113566
MSE loss: 0.04150709560663362, CE loss: 0.15585099602410343
MSE loss: 0.04115645344417656, CE loss: 0.15428341649696597
MSE loss: 0.04090008281795748, CE loss: 0.1531804919480371
MSE loss: 0.04069914624530858, CE loss: 0.15234697660258456
MSE loss: 0.0405345406964214, CE loss: 0.15168312532595454
MSE loss: 0.0403956667518639, CE loss: 0.15113362706705297
```

```
[ ] perceptron.plot() #данная функция определена ранее
```



```
[ ]  plot_division(perceptron) #данная функция определена ранее
```



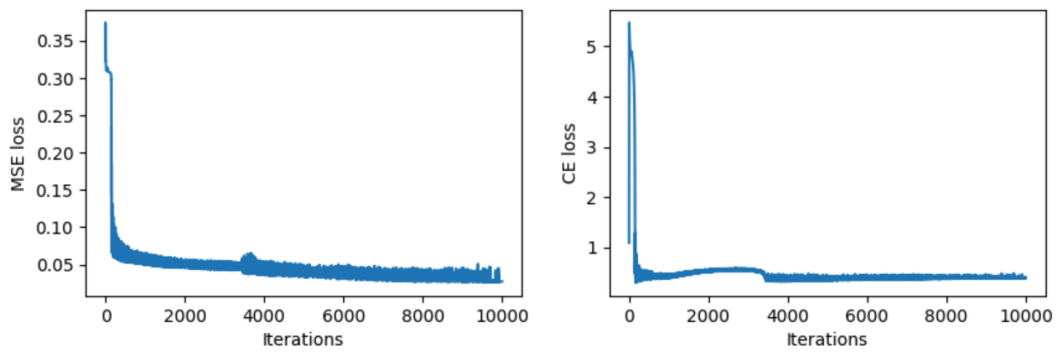
Так, при 4 нейронах наша нейросеть правильно классифицирует абсолютное большинство данных по обучающей выборке.

40 нейронов:

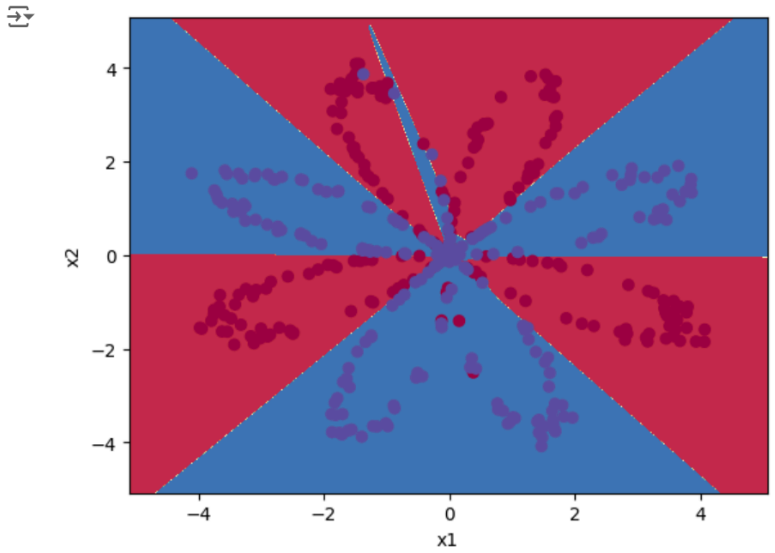
```
[32]  perceptron = Perceptron(X, Y, 40)
      perceptron.learn(10000, 0.1)
```

```
MSE loss: 0.32365727449251686, CE loss: 1.0879868261390868
MSE loss: 0.05639104900886763, CE loss: 0.4246014942576408
MSE loss: 0.04786843903631015, CE loss: 0.5115589596400996
MSE loss: 0.04425326360093815, CE loss: 0.5305829422360838
MSE loss: 0.04200571125707331, CE loss: 0.35303559390979145
MSE loss: 0.038393456987039914, CE loss: 0.36526178301654183
MSE loss: 0.03081605939445432, CE loss: 0.36199144100519975
MSE loss: 0.03171865975539761, CE loss: 0.38196192562101977
MSE loss: 0.029485083974303942, CE loss: 0.37741258454883203
MSE loss: 0.026053016721151945, CE loss: 0.37606606358678796
```

```
[33]  perceptron.plot() #данная функция определена ранее
```



```
[ ]  plot_division(perceptron) #данная функция определена ранее
```



Как видно из графика, при 40 нейронах наблюдается переобучение.

Изменения:

После модификации будем рассматривать лишь случай с 40 нейронами, так как граница разделения при 1 нейроне всегда является прямой и не решает исходной задачи, а при 4 нейронах наша задача уже хорошо выполняется. Также для удобства будем строить единый график потерь для MSE Loss и CE Loss.

Как можно будет увидеть ниже, регуляризация позволяет улучшить поведение сети и сделать границу разделения более плавной и правильной, при этом значение функции потерь CE Loss стало ближе к 0.

Рекомендации по выбору λ:

Чересчур малые значения λ подходят для небольших архитектур (1 или 4 нейрона), где переобучение очень незначительно.

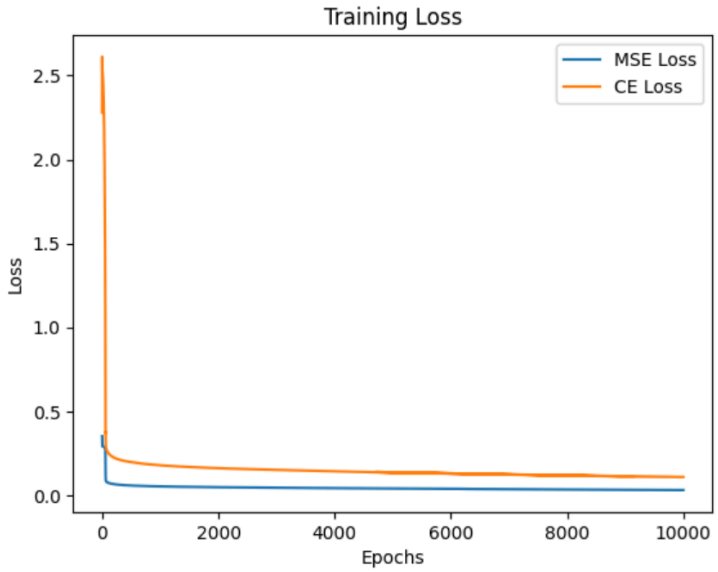
Средние значения λ (около 0.01) оптимальны для сложных архитектур (в нашем случае 40 нейронов), обеспечивая баланс между точностью и обобщением. Поведение модели при таком параметре приведено ниже.

Большие λ обычно используются для случаев, когда модель имеет слишком много параметров, но они могут привести к недообучению.

Так, рассмотрим поведение нашей модели при λ = 0.01. Для правильной работы модели следует соответствующим образом подобрать гиперпараметры, так как в противном случае модель переобучится лишь сильнее:

```
perceptron = Perceptron(X, Y, 40)
perceptron.learn(epochs=10000, learning_rate=0.01, lmdb=0.01)
```

```
Epoch 0: MSE loss = 0.35251689266693154, CE loss = 2.280034020945326
Epoch 1000: MSE loss = 0.053396496735456406, CE loss = 0.1791944200314132
Epoch 2000: MSE loss = 0.04844800549879839, CE loss = 0.16186424490362739
Epoch 3000: MSE loss = 0.04548184310507816, CE loss = 0.15173293468284818
Epoch 4000: MSE loss = 0.042764358718205414, CE loss = 0.14314190973695987
Epoch 5000: MSE loss = 0.040783951486747565, CE loss = 0.13739159367690124
Epoch 6000: MSE loss = 0.038830180121749394, CE loss = 0.1311186484802855
Epoch 7000: MSE loss = 0.03675360776572964, CE loss = 0.12484378846886925
Epoch 8000: MSE loss = 0.03474345855750455, CE loss = 0.11892505232032775
Epoch 9000: MSE loss = 0.03297694503441493, CE loss = 0.11390195015946132
```




```
plot_division(perceptron)
```

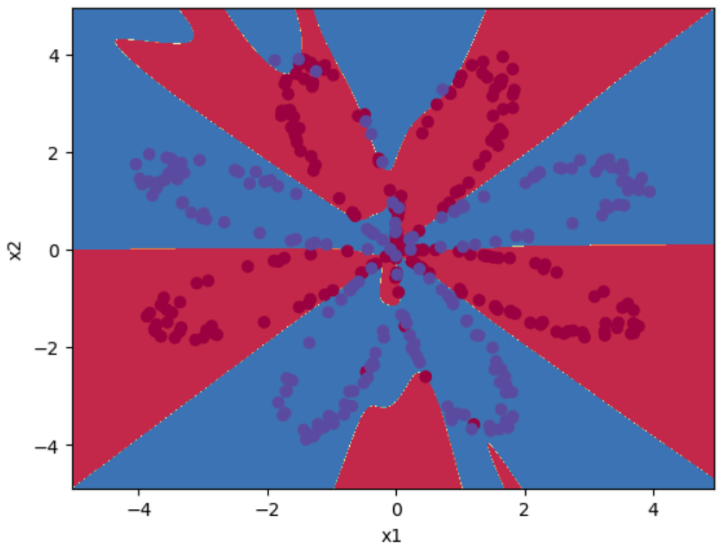
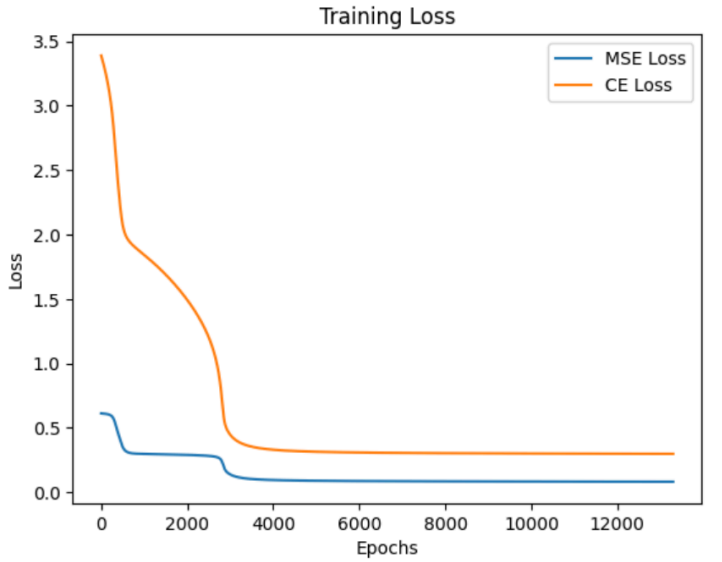


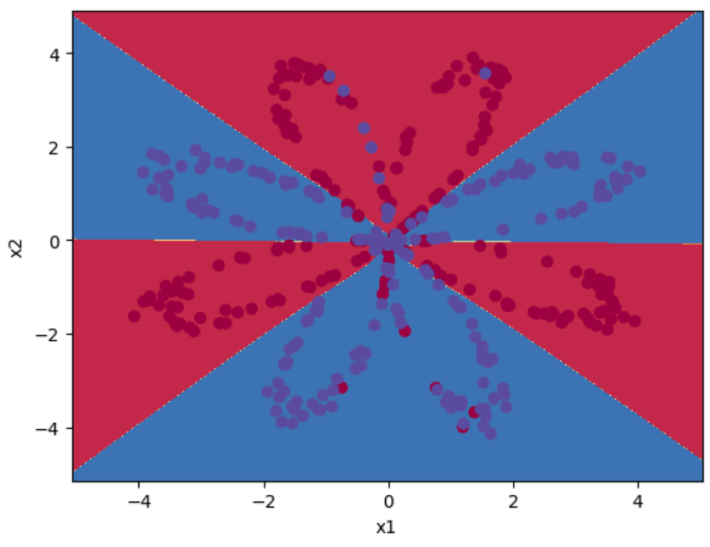
График функций потерь падает катастрофически быстро, и модель чересчур сильно подстраивается под обучающие данные. Попробуем изменить гиперпараметры следующим образом:

```
[250] perceptron = Perceptron(X, Y, 40)
      perceptron.learn(epochs=13300, learning_rate=0.00004, lmdb=0.01)
```

```
Epoch 0: MSE loss = 0.6116258546571158, CE loss = 3.3893902992006373
Epoch 1000: MSE loss = 0.2971850544660477, CE loss = 1.8418888226710055
Epoch 2000: MSE loss = 0.28984016921203815, CE loss = 1.4929849975259537
Epoch 3000: MSE loss = 0.13777724725402404, CE loss = 0.44586767791148646
Epoch 4000: MSE loss = 0.09437493252445961, CE loss = 0.32971416048915303
Epoch 5000: MSE loss = 0.08887189821938789, CE loss = 0.3139137637328257
Epoch 6000: MSE loss = 0.08649174962949503, CE loss = 0.3079830125365593
Epoch 7000: MSE loss = 0.08505038904104525, CE loss = 0.30484481856414325
Epoch 8000: MSE loss = 0.08403294566740614, CE loss = 0.3028547130635842
Epoch 9000: MSE loss = 0.0832511751633592, CE loss = 0.30143766222846624
Epoch 10000: MSE loss = 0.08261799405300041, CE loss = 0.30034261201104184
Epoch 11000: MSE loss = 0.08208658756785074, CE loss = 0.29944337242387126
Epoch 12000: MSE loss = 0.08162904253416273, CE loss = 0.2986699381546775
Epoch 13000: MSE loss = 0.08122742030944995, CE loss = 0.29798046689518265
```



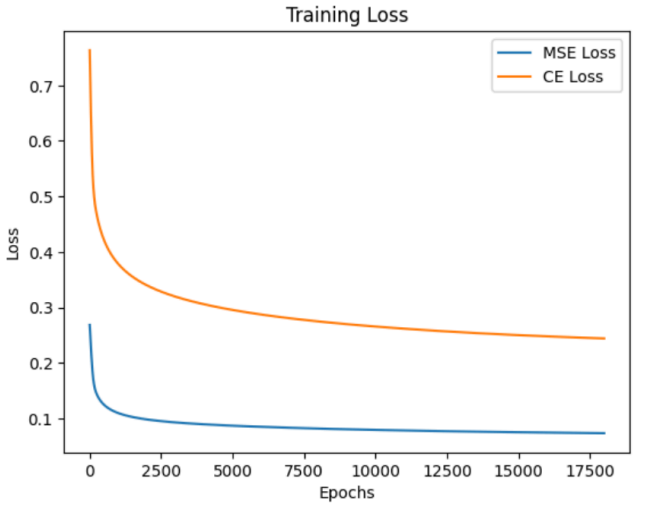

```
plot_division(perceptron)
```



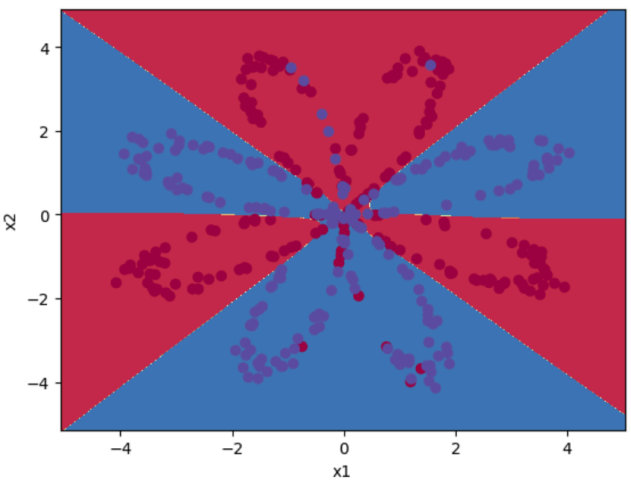
Теперь модель работает аналогично тому, как работала при 4 нейронах. Попробуем уменьшить скорость обучения, увеличив соответственно число эпох, при этом убавив параметр регуляризации до 0.001 и сохранив результат обучения нейросети:

```
perceptron = Perceptron(X, Y, 40)
perceptron.learn(epochs=18000, learning_rate=0.00003, lmdb=0.001)
```

```
Epoch 0: MSE loss = 0.26832877878690886, CE loss = 0.763485830981904
Epoch 1000: MSE loss = 0.10936006416581699, CE loss = 0.3782152721114783
Epoch 2000: MSE loss = 0.09791426749637352, CE loss = 0.3397036575090338
Epoch 3000: MSE loss = 0.0925407757790912, CE loss = 0.31930500766679293
Epoch 4000: MSE loss = 0.08912988355942283, CE loss = 0.3055816512837906
Epoch 5000: MSE loss = 0.08661832034996068, CE loss = 0.29527895773136414
Epoch 6000: MSE loss = 0.0846083538878446, CE loss = 0.28707950535456744
Epoch 7000: MSE loss = 0.08292137923013142, CE loss = 0.2803196676181119
Epoch 8000: MSE loss = 0.08146526341739904, CE loss = 0.27460541517150694
Epoch 9000: MSE loss = 0.08018578171785737, CE loss = 0.2696789588565349
Epoch 10000: MSE loss = 0.07904746034422486, CE loss = 0.2653653781411419
Epoch 11000: MSE loss = 0.07802586465896329, CE loss = 0.2615454348736169
Epoch 12000: MSE loss = 0.07710417310403812, CE loss = 0.25813804359896886
Epoch 13000: MSE loss = 0.07627097979552035, CE loss = 0.2550873165769166
Epoch 14000: MSE loss = 0.07551818713609841, CE loss = 0.25235248620120143
Epoch 15000: MSE loss = 0.07483900345106564, CE loss = 0.24990049475869447
Epoch 16000: MSE loss = 0.07422659627654925, CE loss = 0.24770155192427448
Epoch 17000: MSE loss = 0.07367371122578108, CE loss = 0.24572740249028413
```




```
[118] plot_division(perceptron)
```

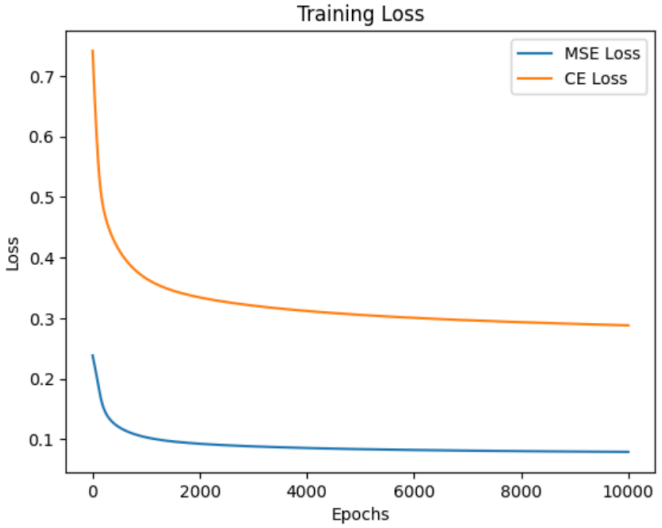


Такая настройка объясняется тем, что меньшая регуляризация требует более точной настройки модели, то есть меньшей скорости обучения, большего числа эпох.

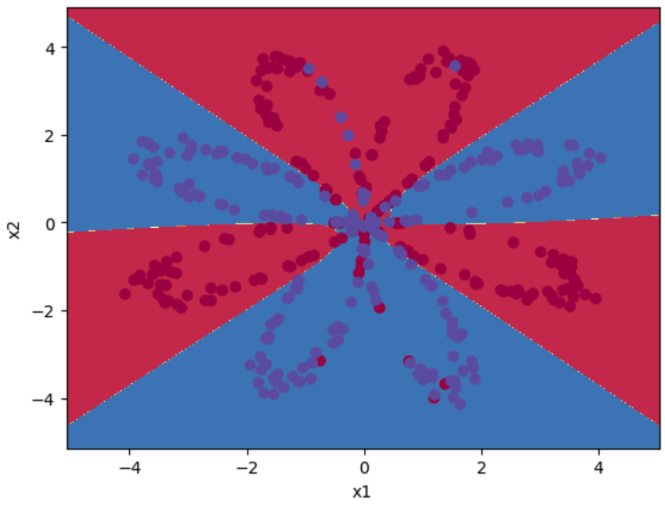
Попробуем увеличить параметр регуляризации до 0.1, сохранив результат:

```
perceptron = Perceptron(X, Y, 40)
perceptron.learn(epochs=10000, learning_rate=0.00006, lmdb=0.1)
```

```
Epoch 0: MSE loss = 0.23825740888899152, CE loss = 0.7412438462982952
Epoch 1000: MSE loss = 0.10311901219136109, CE loss = 0.3653017600131523
Epoch 2000: MSE loss = 0.0925210533600257, CE loss = 0.334135292341457
Epoch 3000: MSE loss = 0.08816754346146805, CE loss = 0.320522053063639
Epoch 4000: MSE loss = 0.08548199690010495, CE loss = 0.31170357777918206
Epoch 5000: MSE loss = 0.08359948896389514, CE loss = 0.30529256607318084
Epoch 6000: MSE loss = 0.08220616549274626, CE loss = 0.3003785535093995
Epoch 7000: MSE loss = 0.08113972723171946, CE loss = 0.2964640408246001
Epoch 8000: MSE loss = 0.080299782984044, CE loss = 0.29322913221820857
Epoch 9000: MSE loss = 0.07961915636630881, CE loss = 0.290453578489865
```




```
plot_division(perceptron)
```



В данном случае количество эпох — 10000, шаг обучения — 0.00006. Так, более высокая регуляризация предотвращает переобучение даже при большем шаге обучения и меньшем числе эпох.

Итоги:

Основные цели поставленной задачи были выполнены:

- реализована регуляризация с параметром *lmdb*;
- модифицирован обратный ход распространения ошибки;
- проведен анализ изменений и оценка их влияния на результат.

Так, для всех трех комбинаций гиперпараметры обеспечивают достаточно гладкую траекторию сходимости функции потерь (MSE и CE), что указывает на корректный баланс между шагом обновления весов и штрафом за величину весов, при этом сохраняя корректную результативность нашей модели.