**Тема №1: Генерирование обучающих и тестовых выборок**

**Цель:**  научиться генерировать различные выборки и добиваться переобучения

**Особенности реализации:** sklearn.datasets

**Эксперимент:** реализовала выборки , соответствующие четырем типам, представленным на <http://playground.tensorow.org>, для этих выборок на сайте подобрала такие параметры, чтобы произошло переобучение

**Результаты:**

Сгенерированные выборки:

Код вот тут: <https://colab.research.google.com/drive/1xsOZCXO8n1exlF8YdJ6nxIHCW1pXMINS?usp=sharing>

Изображение выглядит как снимок экрана, Красочность

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как снимок экрана, фиолетовый, Красочность

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как снимок экрана, круг, Красочность

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как снимок экрана, Красочность

Контент, сгенерированный ИИ, может содержать ошибки.

Переобучение на примерах выборок с <https://playground.tensorflow.org>:

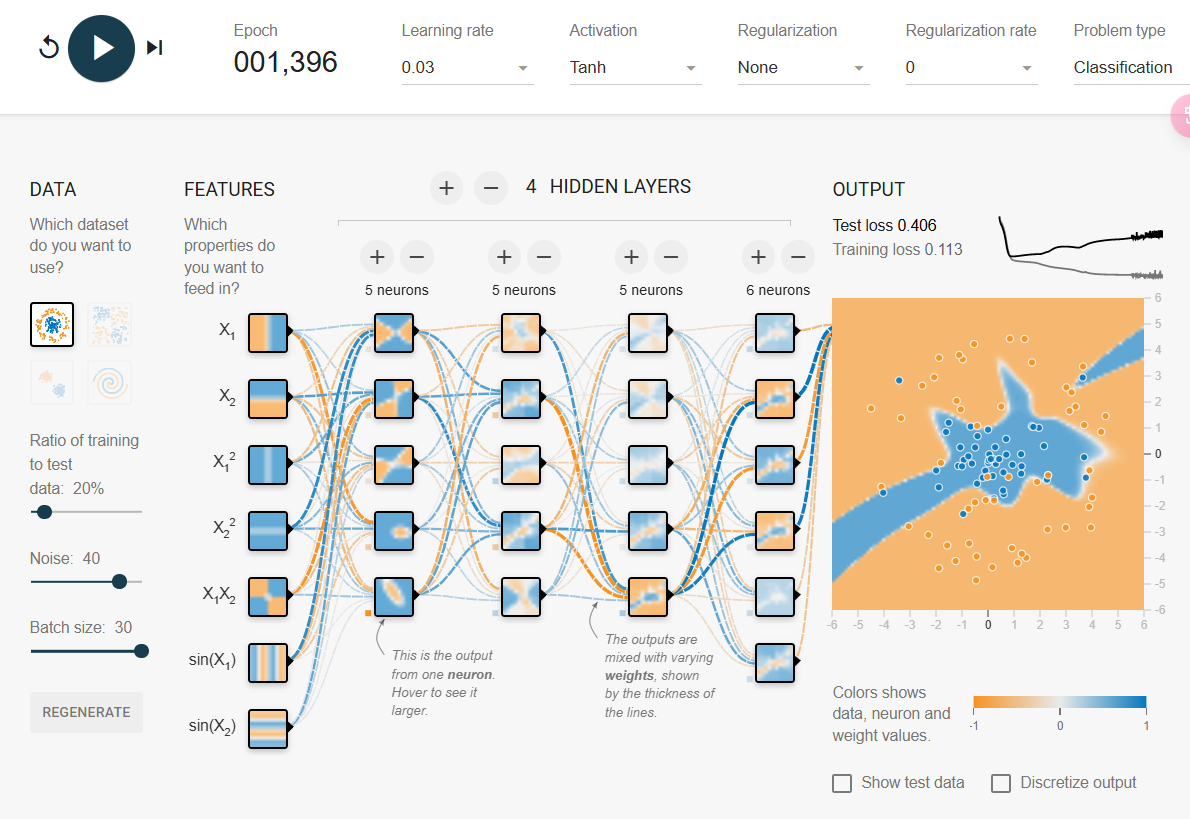
1. Изображение выглядит как текст, снимок экрана, диаграмма, карта

   Контент, сгенерированный ИИ, может содержать ошибки.
2. Изображение выглядит как текст, карта, снимок экрана, диаграмма

   Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

1. 

**Вывод:** Переобучение достигается, если брать маленькую обучающую выборку и много шумов

**Тема №2: Реализации элементарного перцептрона**

**Цель:** понять, как устроен элементарный перцептрон, реализовать его, сравнить работу перцептронов с разными функциями активации и алгоритмами отладки весов

**Особенности реализации:** numpy, sklearn.datasets, graphviz

**Эксперимент:** было реализовано два элементарных перцептрона. Первый обучался с помощью алгоритма из теоремы о сходимости элементарного перцептрона и ступенчатой функцией в качестве функции активации, второй – с помощью градиентного спуска с обратным распространением ошибки BCE loss (Binary Cross-Entropy Loss) и сигмоиды. Обучение происходило на выборках из задания 2.1

Результаты обучения представила в виде матрицы ошибок, также сравнила время обучения и качество классификации моделей

**Результаты:**

Код в том же колабе

Вычислительный граф:

Изображение выглядит как текст, рукописный текст, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Обучение элементарного перцептрона со ступенчатой функцией по теореме о сходимости элементарного перцептрона не сойдется, если данные не линейно разделимы, однако можно указать конкретное количество эпох, которое должен пройти перцептрон, и обучить его на различных данных.

Матрицы ошибок:

1. Ступенчатая функция + линейно разделенные данные (training = 1000, test = 500)

|  |  |  |
| --- | --- | --- |
|  | Предсказан 0 | Предсказан 1 |
| 0 | TN = 500 | FP = 0 |
| 1 | FN = 0 | TP = 500 |

Train accuracy: 1.0

|  |  |  |
| --- | --- | --- |
|  | Предсказан 0 | Предсказан 1 |
| 0 | TN = 250 | FP = 0 |
| 1 | FN = 0 | TP = 250 |

Test accuracy: 1.0

Training time: 9.25 sec

1. Ступенчатая функция + спирали (training = 1000, test = 500)

|  |  |  |
| --- | --- | --- |
|  | Предсказан 0 | Предсказан 1 |
| 0 | TN = 2 | FP = 998 |
| 1 | FN = 0 | TP = 1000 |

Train accuracy: 0.501

|  |  |  |
| --- | --- | --- |
|  | Предсказан 0 | Предсказан 1 |
| 0 | TN = 1 | FP = 499 |
| 1 | FN = 0 | TP = 500 |

Test accuracy: 0.501

Training time: 8.98 sec

1. Ступенчатая функция + круги (training = 1000, test = 500)

|  |  |  |
| --- | --- | --- |
|  | Предсказан 0 | Предсказан 1 |
| 0 | TN = 0 | FP = 500 |
| 1 | FN = 0 | TP = 500 |

Train accuracy: 0.5

|  |  |  |
| --- | --- | --- |
|  | Предсказан 0 | Предсказан 1 |
| 0 | TN = 0 | FP = 250 |
| 1 | FN = 0 | TP = 250 |

Test accuracy: 0.5

Training time: 9.20 sec

1. Ступенчатая функция + XOR (training = 1000, test = 500)

|  |  |  |
| --- | --- | --- |
|  | Предсказан 0 | Предсказан 1 |
| 0 | TN = 0 | FP = 500 |
| 1 | FN = 0 | TP = 500 |

Train accuracy: 0.5

|  |  |  |
| --- | --- | --- |
|  | Предсказан 0 | Предсказан 1 |
| 0 | TN = 0 | FP = 250 |
| 1 | FN = 0 | TP = 250 |

Test accuracy: 0.5

Training time: 3.89 sec

1. Сигмоида + линейно разделенные данные (training = 1000, test = 500)

|  |  |  |
| --- | --- | --- |
|  | Предсказан 0 | Предсказан 1 |
| 0 | TN = 500 | FP = 0 |
| 1 | FN = 9 | TP = 500 |

Train accuracy: 1.0

|  |  |  |
| --- | --- | --- |
|  | Предсказан 0 | Предсказан 1 |
| 0 | TN = 250 | FP = 0 |
| 1 | FN = 9 | TP = 250 |

Test accuracy: 1.0

Training time: 14.49 sec

1. Сигмоида + круги (training = 1000, test = 200, learning rate = 0.03, epochs = 1000)

|  |  |  |
| --- | --- | --- |
|  | Предсказан 0 | Предсказан 1 |
| 0 | TN = 148 | FP = 352 |
| 1 | FN = 0 | TP = 500 |

Train accuracy: 0.648

|  |  |  |
| --- | --- | --- |
|  | Предсказан 0 | Предсказан 1 |
| 0 | TN = 31 | FP = 69 |
| 1 | FN = 1 | TP = 99 |

Test accuracy: 0.65

Training time: 16.35 sec

1. Сигмоида + спирали (training = 1000, test = 200, learning\_rate = 0.03, epochs = 1000)

|  |  |  |
| --- | --- | --- |
|  | Предсказан 0 | Предсказан 1 |
| 0 | TN = 0 | FP = 500 |
| 1 | FN = 0 | TP = 500 |

Train accuracy: 0.5

|  |  |  |
| --- | --- | --- |
|  | Предсказан 0 | Предсказан 1 |
| 0 | TN = 0 | FP = 250 |
| 1 | FN = 0 | TP = 250 |

Test accuracy: 0.5

Training time: 15.87 sec

1. Сигмоида + XOR (training = 1000, test = 500)

|  |  |  |
| --- | --- | --- |
|  | Предсказан 0 | Предсказан 1 |
| 0 | TN = 0 | FP = 500 |
| 1 | FN = 0 | TP = 500 |

Train accuracy: 0.5

|  |  |  |
| --- | --- | --- |
|  | Предсказан 0 | Предсказан 1 |
| 0 | TN = 0 | FP = 250 |
| 1 | FN = 0 | TP = 250 |

Test accuracy: 0.5

Training time: 15.24 sec

**Вывод:** С линейно разделимыми данными оба перцептрона справляются (точность обоих равна 1.0). Нелинейно разделимую выборку перцептрон со ступенчатой функцией плохо классифицирует (точность примерно 0.5), так как по теореме о сходящемся перцептроне такой перцептрон подходит только для линейно разделимых данных. Перцептрон с сигмоидой и градиентным спуском тоже плохо классифицирует нелинейные выборки (за исключением кругов), так как одного слоя для этого недостаточно.