МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ

ФЕДЕРАЦИИ

**Государственное образовательное бюджетное учреждение высшего**

**образования**

**«Кубанский Государственный Университет»**

**(ФГБОУ ВО «КубГУ»)**

Факультет компьютерных технологий и прикладной математики

**ОТЧЁТ**

**Персептрон на Python**

Дисциплина: Нечёткие и нейросетевые технологии в экономике

Выполнил:

Студент 47 группы

Хачатрян Арутюн Арамович

Преподаватель:

Прутский Алексей Сергеевич

Краснодар

2019

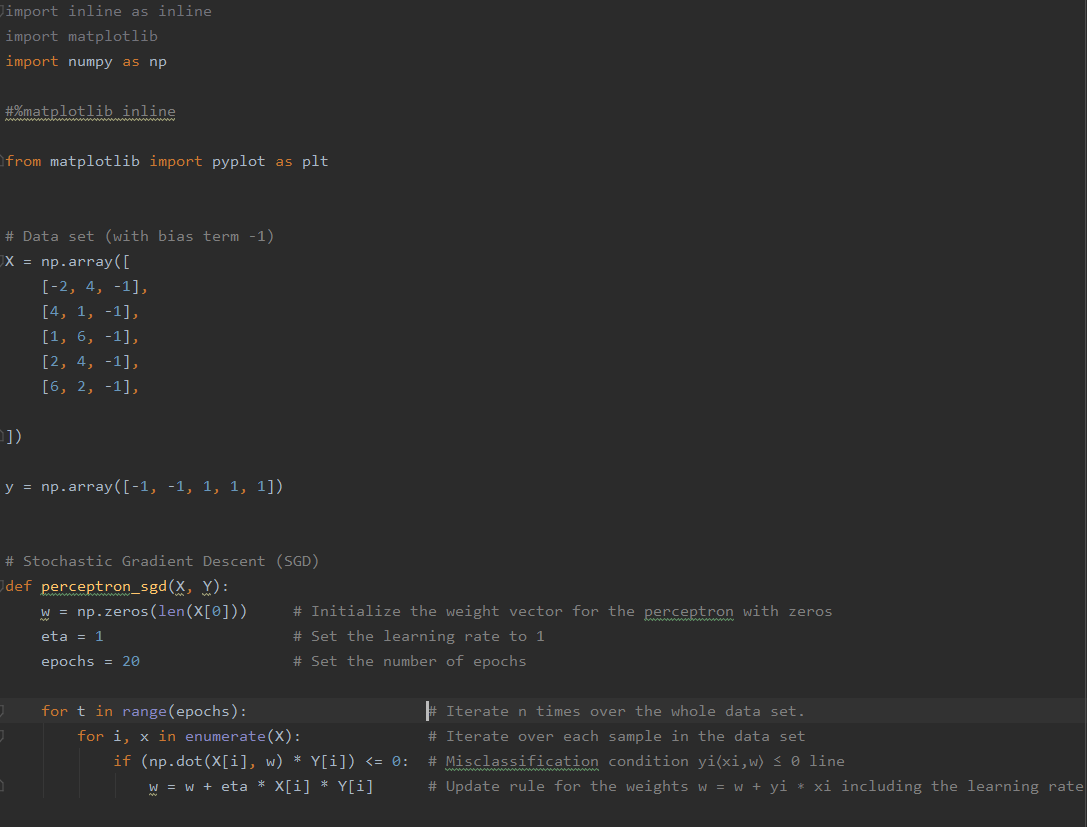
# Введение

Перцептро́н, или персептрон (англ. perceptron от лат. perceptio — восприятие; нем. Perzeptron) – математическая или компьютерная модель восприятия информации мозгом (кибернетическая модель мозга), предложенная Фрэнком Розенблаттом в 1957 году и впервые реализованная в виде электронной машины «Марк-1» в 1960 году. Перцептрон стал одной из первых моделей нейросетей, а «Марк-1» – первым в мире нейрокомпьютером.

Перцептрон состоит из трёх типов элементов, а именно: поступающие от датчиков сигналы передаются ассоциативным элементам, а затем реагирующим элементам. Таким образом, перцептроны позволяют создать набор «ассоциаций» между входными стимулами и необходимой реакцией на выходе. В биологическом плане это соответствует преобразованию, например, зрительной информации в физиологический ответ от двигательных нейронов.

Персептрон может быть использован для контролируемого обучения. Это может решить проблемы двоичной линейной классификации.

## Программная реализация







Сначала мы импортируем *numpy*, чтобы легко управлять операциями линейной алгебры и исчисления в Python. Для того, чтобы отобразить прогресс обучения позже, мы будем использовать *matplotlib*.

Мы реализуем алгоритм персептрона в Python 3 и Numpy. Перцептрон научится использовать алгоритм стохастического градиентного спуска (SGD). Градиентный спуск минимизирует функцию, следуя градиентам функции стоимости.

Чтобы вычислить ошибку прогноза, нам сначала нужно определить целевую функцию персептрона. Для этого нам нужно определить функцию потерь, чтобы рассчитать ошибку прогнозирования.

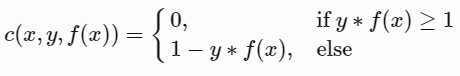
### Hinge Loss Function (Функция потери шарнира)

Мы будем использовать потерю шарнира для нашего персептрона:



c – функция потерь, x – выборка, y – истинная метка, f (x) – прогнозируемая метка.

Это означает следующее:



Итак, рассмотрим, если y и f (x) являются знаковыми значениями (+ 1, -1):

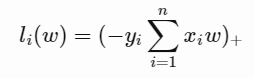
* потеря равна 0, если y ∗ f (x) положительны, оба значения имеют одинаковый знак.
* потеря равна 1 − y ∗ f (x), если y ∗ f (x) отрицательна

### Целевая функция

Определив функцию потерь, мы теперь можем определить целевую функцию для персептрона:



Мы можем написать это без точечного произведения со знаком суммы:



Таким образом, образец xi неверно классифицирован, если yi⟨xi, w⟩≤0. Общая цель состоит в том, чтобы найти глобальные минимумы этой функции, соответственно найти параметр w, где ошибка равна нулю.

### Выведем целевую функцию

Для этого нам нужны градиенты целевой функции. Градиент функции f является вектором ее частных производных. Градиент может быть рассчитан по частной производной целевой функции:

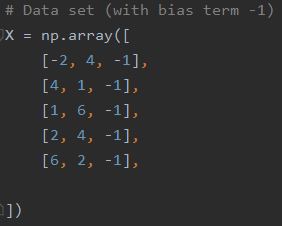


Это означает, что если мы имеем неправильно классифицированный образец xi, соответственно yi⟨xi, w⟩≤0, обновим вектор весов w, переместив его в направлении неправильно классифицированного образца:



### Наш набор данных

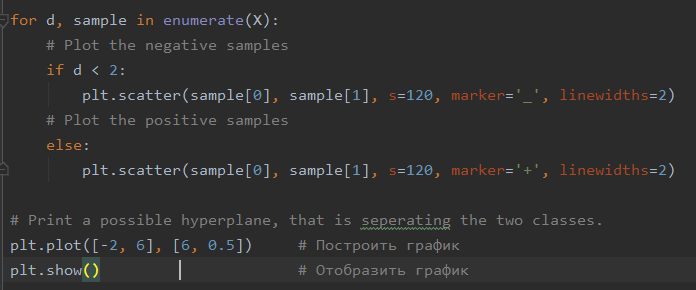
Набор данных с термином смещения -1. Он необходимо для работы SGD.

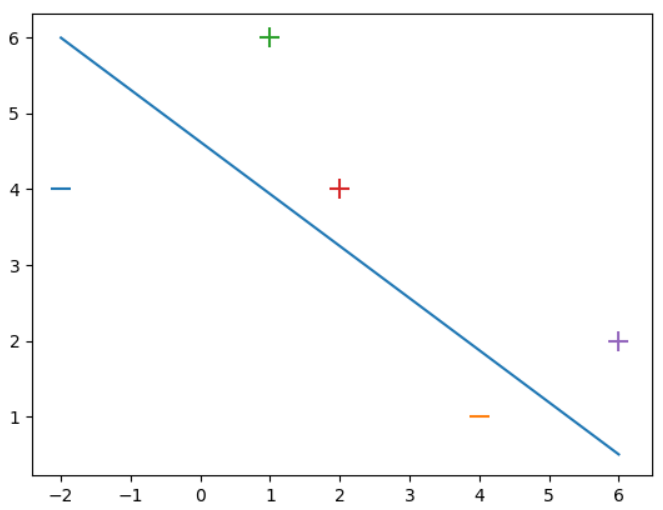


Следующий небольшой набор данных содержит две выборки, помеченные -1, и три выборки, помеченные +1:



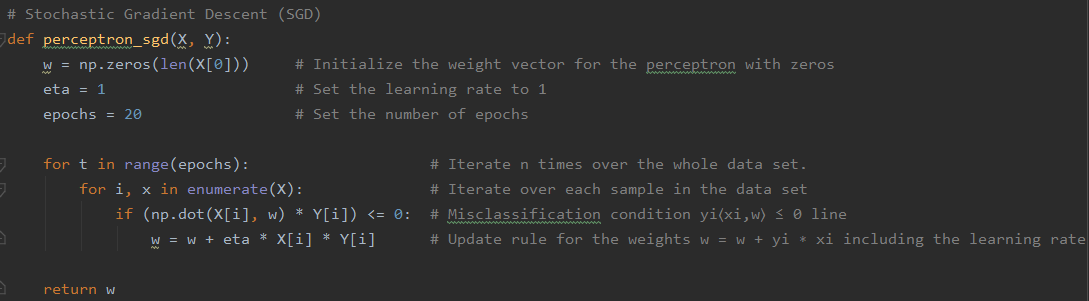
Это означает, что у нас есть проблема двоичной классификации, поскольку набор данных содержит два примера классов. Построим набор данных, чтобы увидеть, что он линейно разделим:





## Реализация стохастического градиентного спуска

Мы будем линейно зацикливаться на наборе образцов. Для больших наборов данных имеет смысл случайным образом выбирать выборку во время каждой итерации в цикле for.

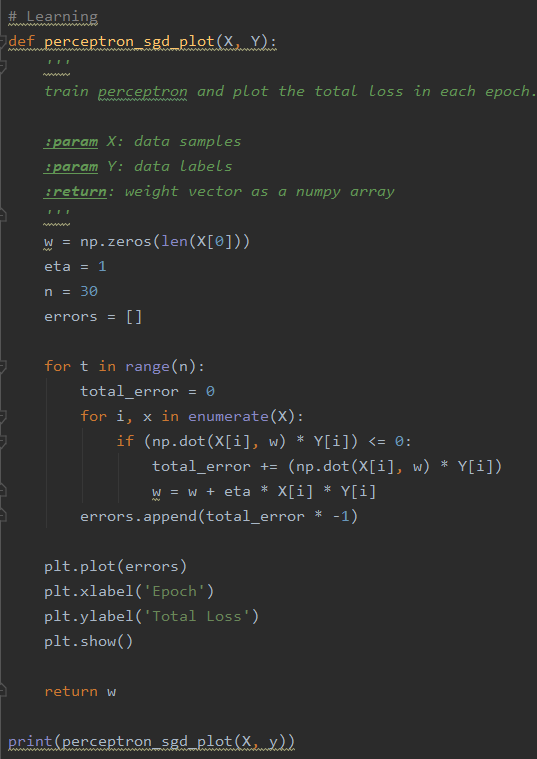


### Описание кода строка за строкой

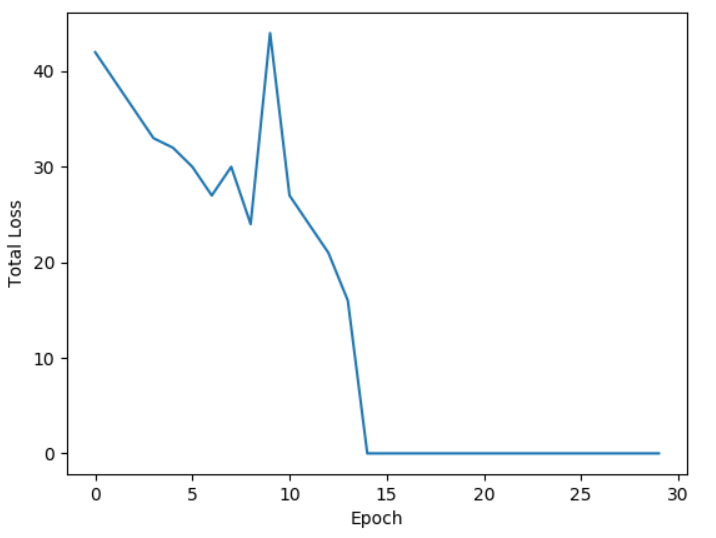
* строка 2: инициализируем весовой вектор для персептрона с нулями
* строка 3: установим скорость обучения на 1
* строка 4: установим количество эпох
* строка 6: итерация ***n*** раз по всему набору данных
* строка 7: итерация по каждой выборке в наборе данных
* строка 8: условие неправильной классификации yi⟨xi, w⟩≤0
* строка 9: правило обновления весов w=w+yi∗xi, включая скорость обучения

### Обучение Персептрона

Затем мы можем выполнить наш код и проверить, сколько итераций необходимо, пока все сэмплы не будут классифицированы правильно. Чтобы увидеть прогресс в обучении персептрона, мы добавили функцию построения графиков в наш алгоритм, подсчитывая общую ошибку в каждой эпохе.







Это означает, что персептрону потребовалось 14 эпох, чтобы правильно классифицировать все выборки (общая ошибка равна нулю). Другими словами, алгоритму необходимо было просмотреть набор данных 14 раз, чтобы узнать его структуру.

Весовой вектор, включающий член смещения, равен (2, 3, 13).

Теперь мы можем извлечь следующую функцию прогнозирования:



Весовой вектор равен (2, 3), а член смещения – третья запись -13.

### Оценка

Вручную классифицируем образцы в нашем наборе данных, чтобы проверить, правильно ли обучен персептрон:

Первый образец (-2, 4), должен быть отрицательный:



Второй образец (4, 1), должен быть отрицательный:



Третий образец (1, 6), должен быть положительный:



Четвертый образец (2, 4), должен быть положительный:



Пятый образец (6, 2), должен быть положительный:



Теперь давайте определим два тестовых образца, чтобы проверить, насколько хорошо наш перцептрон обобщается на невидимые данные:

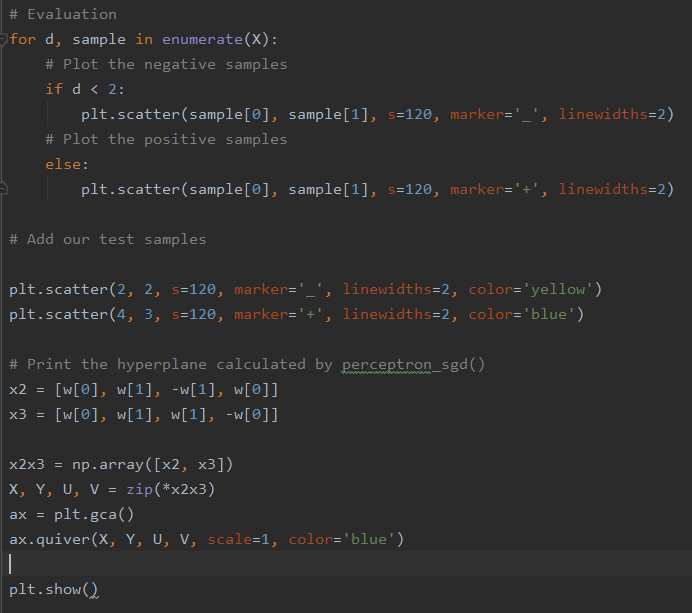
Первый тестовый образец (2, 2), должен быть отрицательный:

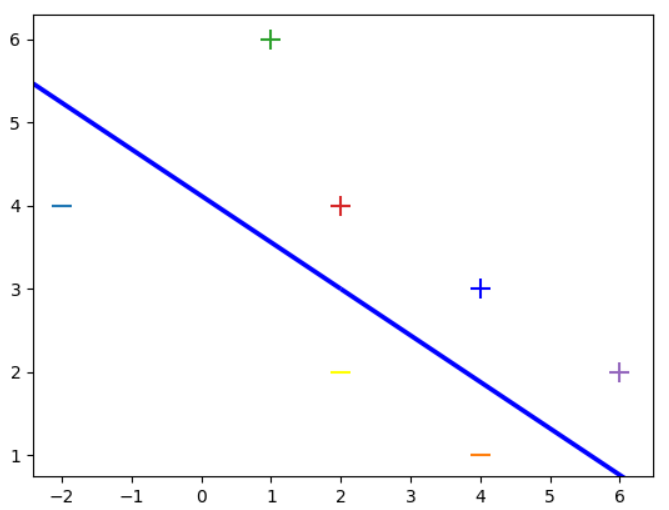


Второй тестовый образец (4, 3), предположительно положительный:



Оба образца классифицированы верно. Чтобы проверить это геометрически, построим образцы, включая тестовые образцы и гиперплоскость.





## Заключение

Основной структурой является SGD, применяемая к целевой функции персептрона. Это всего лишь четыре строки кода. Он содержит всю магию обучения.