KARPOV.COURSES >>>
КОНСПЕКТ

Конспект > 4 урок > Градиентный спуск

> Градиентный спуск. Введение

По ряду причин, рассмотренных в предыдущей лекции, матричная форма не всегда подходит для написания модели. Например, там нужно производить операцию обращения
матрицы, однако не любая матрица является обратимой. Также, данная операция будет потреблять огромные вычислительные ресурсы и занимать долгое время, если количество элементов в матрице велико.

Самой главной причиной неудобства написания модели через матричную форму является то, что не для всех задач можно подобрать универсальную формулу, описывающую единственное и лучшее решение. Поэтому требуется более гибкий подход для
оптимальной оценки параметров.

Такой универсальный подход существует, и называется он градиентный спуск.

Градиентный спуск — метод нахождения локального минимума или максимума функции с помощью движения вдоль градиента.

Градиент — это вектор, своим направлением указывающий направление наибольшего
возрастания некоторой скалярной величины, а по модулю равный скорости роста этой величины в этом направлении.

Градиент функции — это вектор, координатами которого являются частные производные этой функции по всем её переменным. Направление градиента есть направление
наибыстрейшего возрастания функции, то есть производная функции (а точнее её знак) показывает, в каком направлении функции возрастает.

> Минимизация функции с одной переменной

Этапы градиентного спуска для функции одной переменной:

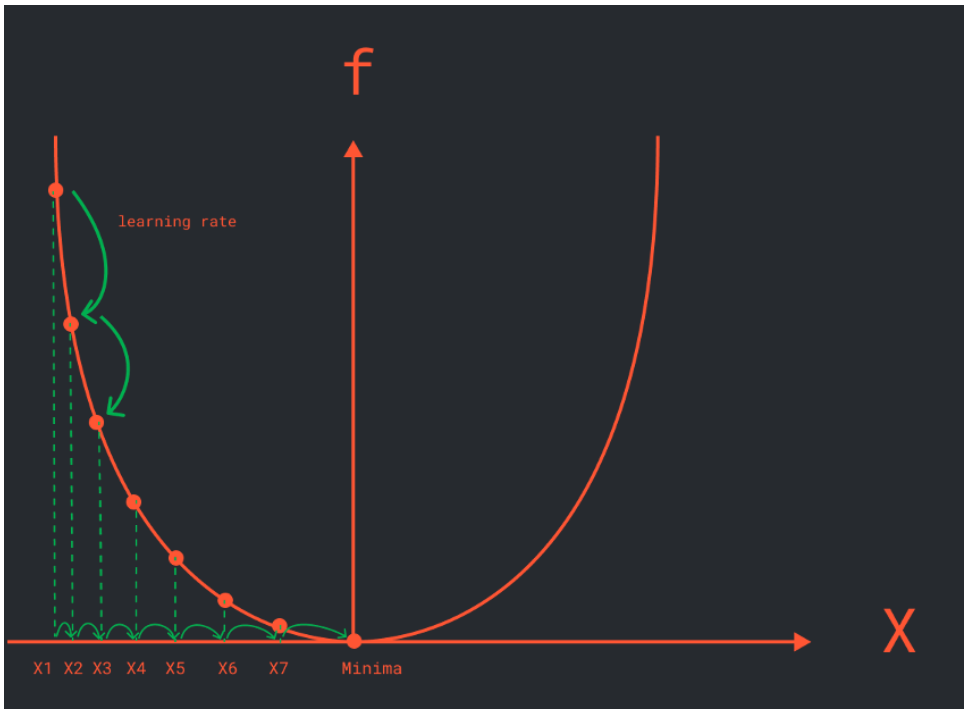
1. Инициализируемся в случайной точке x_{start}, в окрестности которой функция $f(x)$должна быть определена и дифференцируема. Обычно это точка 0.
2. Спускаемся до сходимости модели. Здесь модель должна сделать шаг от изначальной точки — из точки x_{start} вычитается производная функции $f(x)$, нормированная на параметр η. На следующем шаге полученная точка становится исходной точкой, и операция повторяется.

$$x_n = x_{(n-1)} - \eta(dx_{n-1})$$

3. Останавливаемся тогда, когда значение производной становится очень маленьким.

Рассмотрим градиентный спуск на примере функции одной переменной $f(x) = x^2$
.

Здесь мы инициализируемся в точке x_1 и постепенно спускаемся до минимума функции. Изменение значения функции при градиентном спуске называется learning rate.



Как выбрать этот порог — критерий останова?

Данный порог выбирается самостоятельно и обозначается буквой ξ.

Варианты нахождения критерия останова:

1. Определенное минимальное значение производной. Если достигаем выбранного значения производной, то останавливаем спуск.

$$||dx_{start})|| \leq \xi$$

2. Маленькая длина шага. Если шаг градиента становится маленьким, то останавливаем спуск.

$$|x_{start} - x_{next}| \leq \xi$$

3. Маленькое изменение значения функции. Если значение функции изменяется незначительно, то останавливаем спуск.

$$f(x_{next}) - f(x_{start}) \leq \xi$$

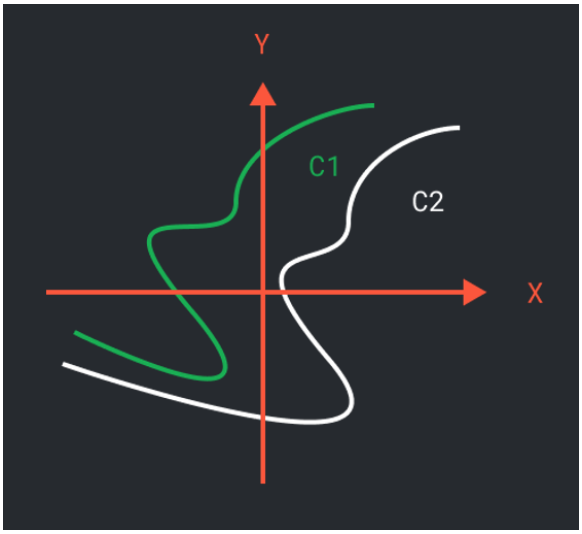
> Минимизация функции нескольких переменных.

Функция нескольких переменных

$z = f(x, y)$ есть функция двух переменных, если каждой паре (x, y) значений двух независимых переменных из области W ставится определенное значение z. Для функции нескольких переменных существует понятие линий уровня. Для функции нескольких переменных существует понятие линий уровня.

Линии уровня – множество всех точек, которые при подстановке дают одинаковое значение функции. Они не пересекаются.

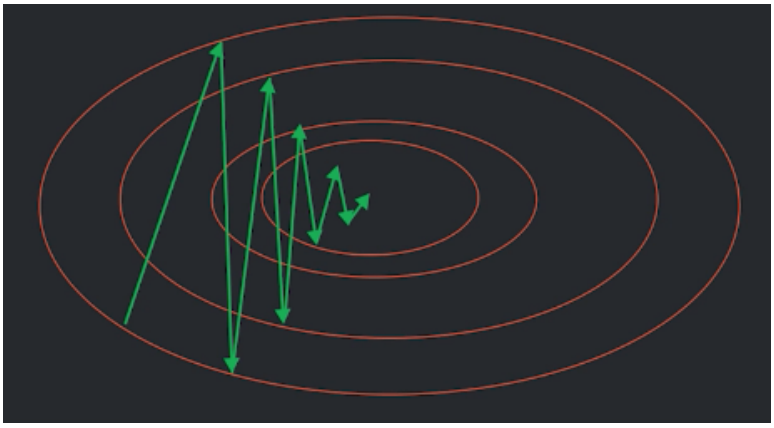
На графике ниже *C1* и *C2* являются линиями уровня функции.



Как находясь на какой-то линии уровня понять направление? Как прийти в точку минимума?

Здесь та же логика, которая использовалась для нахождения минимума функции одной переменной. Градиент функции – вектор из производных функции первого порядка, он показывает направлении быстрейшего роста. Так как наша задача найти, наоборот, минимум, то ставим знак минус у каждого элемента и получаем антиградиент.

Градиент перпендикулярен касательной линии уровня. Постепенно переходя с одной линии уровня на другую (зеленые стрелки на графике), находим минимум функции.



Этапы градиентного спуска для функции нескольких переменных:

1. Инициализируемся в случайной точке x_{start}.
2. Спускаемся до сходимости модели.

$step = \nabla z\prime(x_{start})$

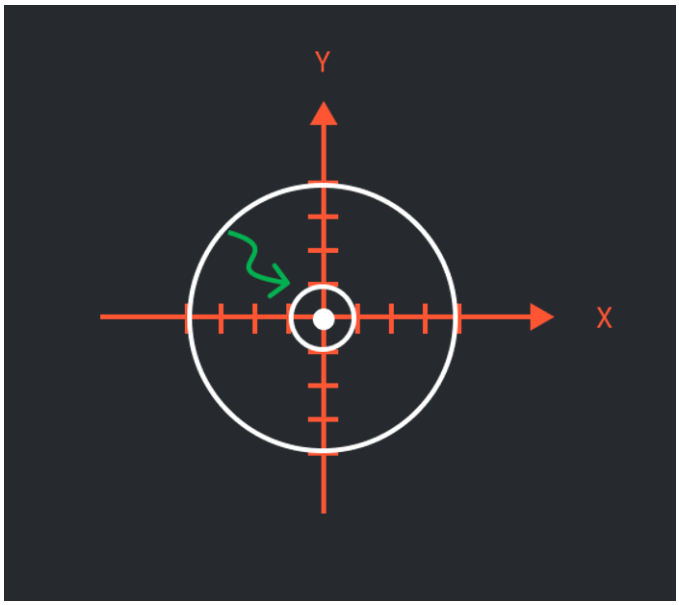
$x_{next} = x_{start} - \eta_i * step$

$xstart = xnext$

3. Останавливаемся тогда, когда значение производной становится очень маленьким (те же три варианта порога).

Рассмотрим градиентный спуск на примере функции нескольких переменных $z(x, y) = x^2 + y^2$.

Данная функция представляет собой окружность. Чтобы найти минимум функции (центр окружности), нужно спускаться перпендикулярно касательной линий уровня.



Как понять, когда нужно остановиться?

1. Маленькая длина градиента. Длиной градиента является длина вектора из производных функции первого порядка.

$$|| \ z(x_{start}) \ || \leq \xi$$

2. Маленькая длина шага.

$$| \ x_{start} - x_{next} \ | \leq \xi$$

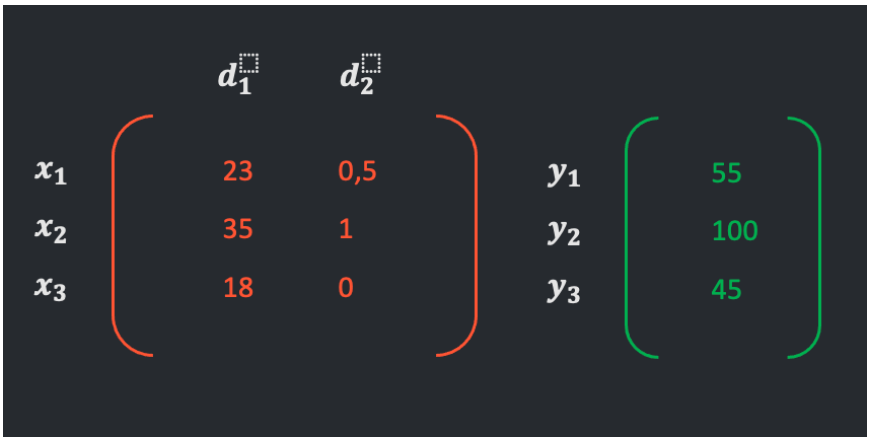
3. Маленькое изменение в значении функции.

$$f(x_{next} - f(x_{start}) \leq \xi$$

> Линейная регрессия

Здесь всё точно так же, только считаем производные по всем параметрам регрессии ($\beta 1$, $\beta 2$, $\beta 0$).

Пусть имеем n объектов и m признаков, для которых нужно составить уравнение линейной регрессии.



$$Q = \frac{1}{n}\sum_i^n (a(x_i) - y_i)^2 = \frac{1}{n}\sum_i^n (\beta_1 \cdot d_1^i + \ldots + \beta_m \cdot d_m^i - y_i)^2$$
$$Q'_{\beta_1} = \frac{2}{n} \cdot \sum_i^n d_1^i \cdot (\beta_1 \cdot d_1^i + \ldots + \beta_m \cdot d_m^i - y_i)$$
$$\ldots$$
$$Q'_{\beta_m} = \frac{2}{n} \cdot \sum_i^n d_m^i \cdot (\beta_1 \cdot d_1^i + \ldots + \beta_m \cdot d_m^i - y_i)$$
$$\nabla Q = (Q'_{\beta_1} \ldots Q_(\beta_m)')$$
$$\beta_{next} = \beta_{start} - \eta \cdot \nabla Q$$

> Подбор параметров η и ξ. Плюсы и минусы градиентного спуска

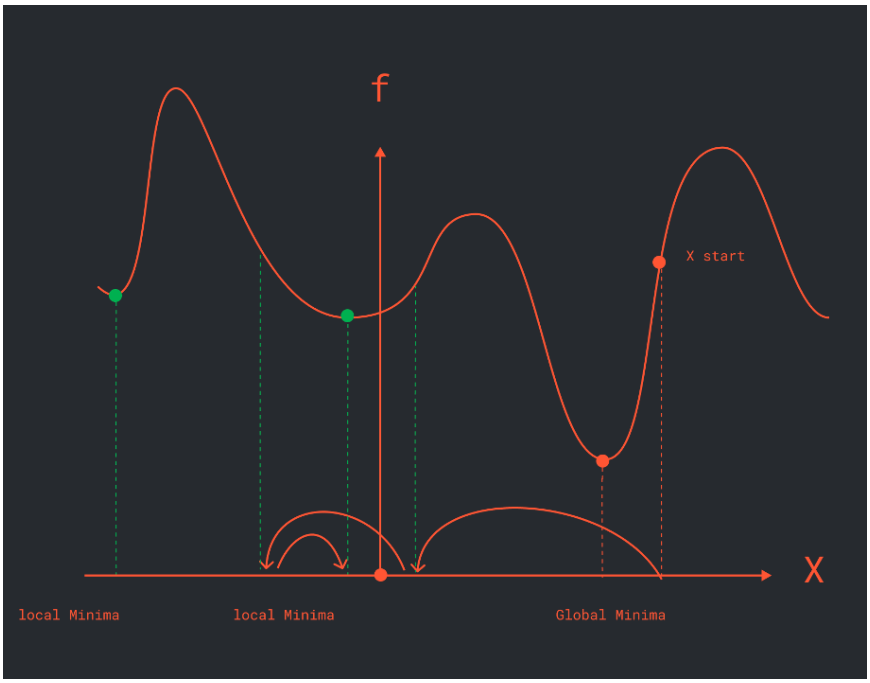
Как правильно выбирать параметр η и ξ? Какие могут быть проблемы, если неправильно выбрать параметры?

Сначала отметим, что у функции нескольких переменных бывает несколько экстремумов – глобальный и локальный минимумы (и максимумы).

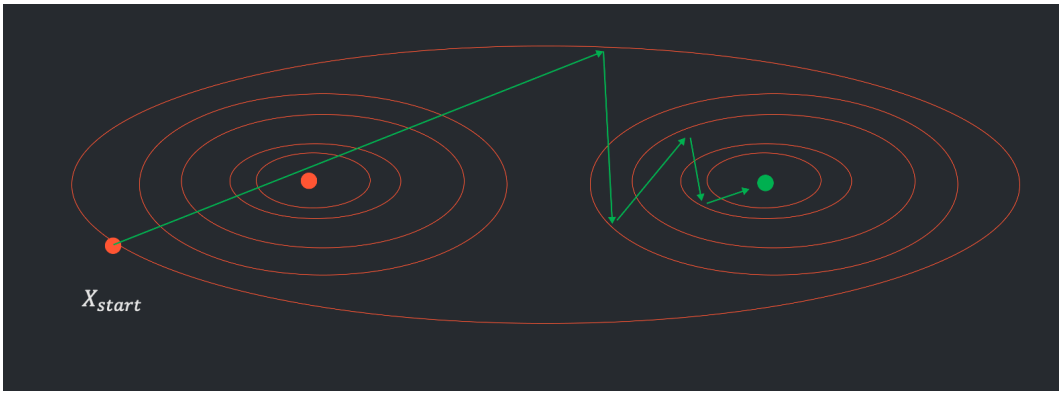
Проблемы параметра η:

1. Большая длина шага приводит к перешагиванию желанной точки — глобального минимума и попадания вместо этого в локальный минимум.

Большая длина шага при минимизации функции одной переменной.

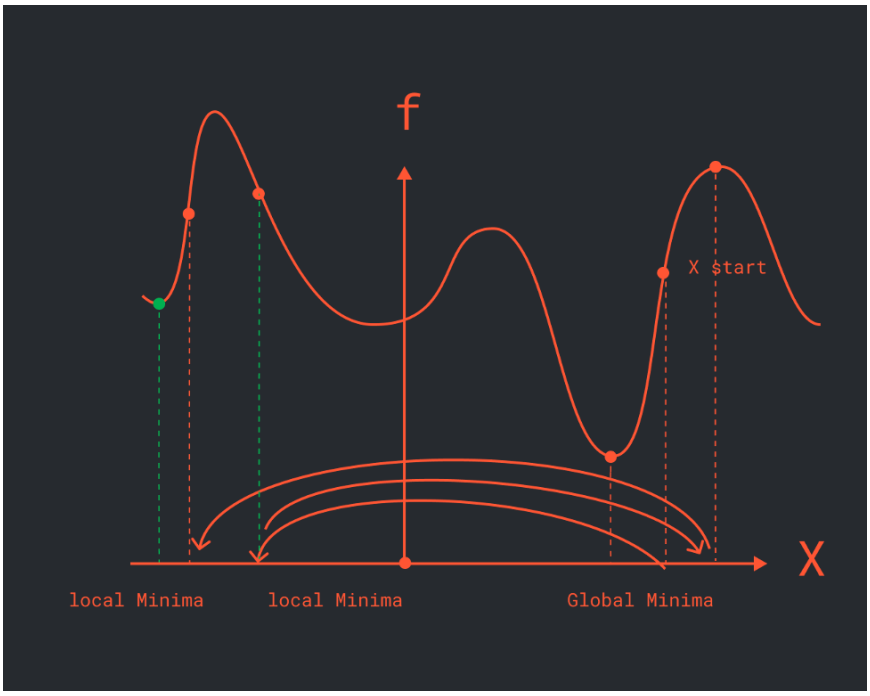


Большая длина шага при минимизации функции нескольких переменных.

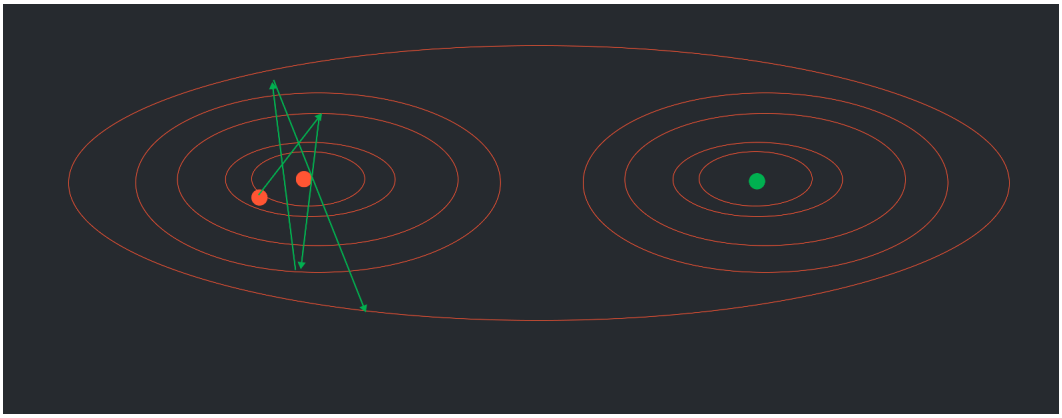


2. «Взрыв» градиента — бесконечно блуждание по функции без схождения градиента.

Взрыв градиента при минимизации функции одной переменной.

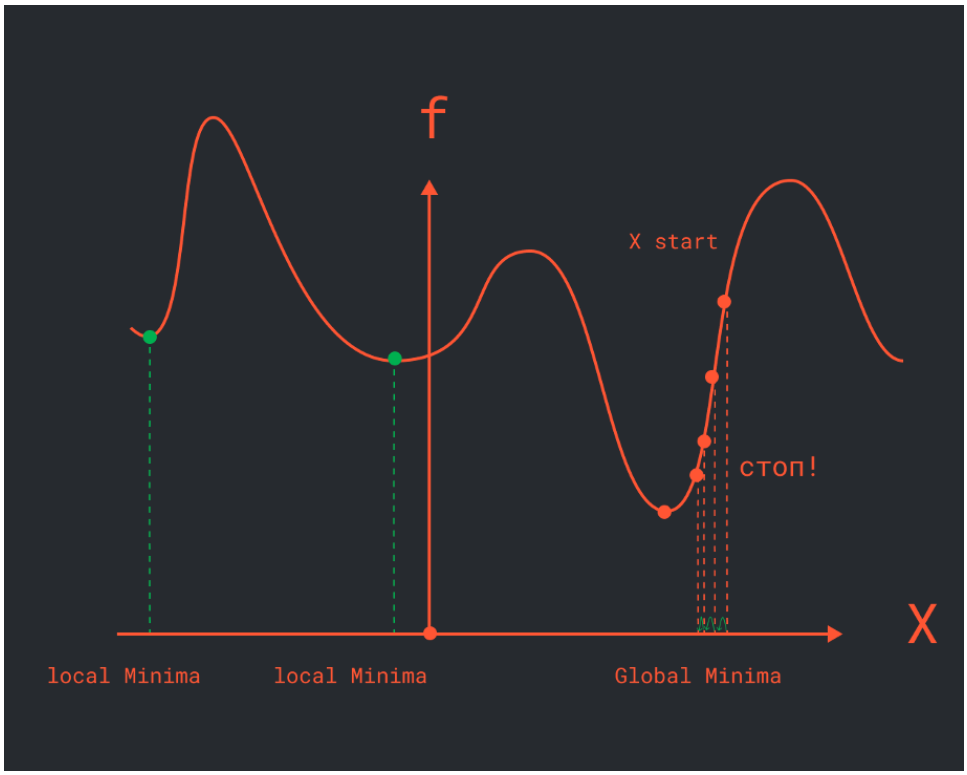


Взрыв градиента при минимизации функции нескольких переменных.

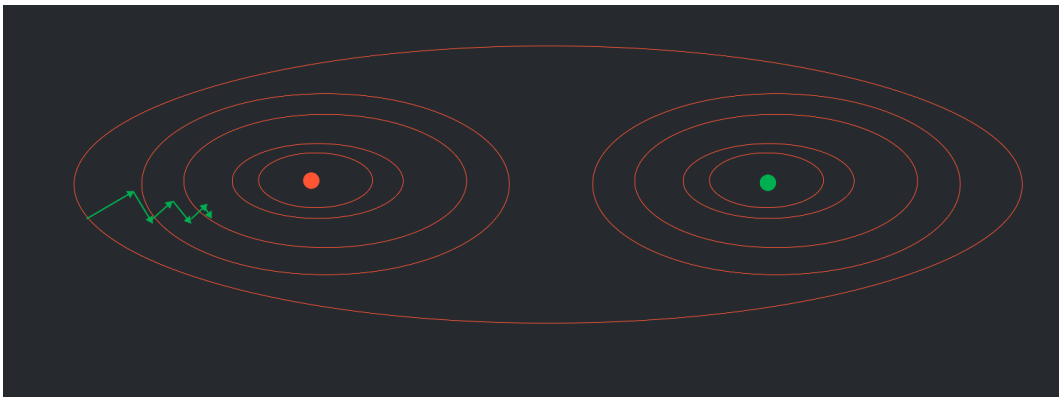


3. Затухание градиента — слишком маленькая длина шага приводит к раннему срабатыванию критерия останова.

Затухание градиента при минимизации функции одной переменной.



Затухание градиента при минимизации функции нескольких переменных.



Проблемы параметра ξ:

1. Консервативный спуск — маленький трешхолд позволит максимально приблизится к минимуму, но потребует много итераций.

2. Либеральный спуск — большой трешхолд позволит избежать большого количества итераций, однако может привести к раннему срабатыванию критерия останова.

Как правильно выбрать точку старта?

Здесь можно только экспериментировать.

Преимущества градиентного спуска:

- универсальный метод;
- можно распараллелить вычисления;
- легко переделать для задачи поиска максимума функции.

Недостатки градиентного спуска:

- нужна аккуратная настройка;
- непонятно, когда остановиться.

> Визуализация нахождения градиента функции в Python

Одна из основных библиотек питона, используемых для визуализации, является библиотека `matplotlib`.

Принцип её работы — сначала создается `figure`, своеобразный канвас, полотно, а затем на нем отрисовываются графики — `axes`. При желании можно установить дефолтные настройки для `matplotlib`, которые будут использованы для всех графиков.

```
import matplotlib.pyplot as plt


fig = plt.figure() # создаем канвас
fig.set_size_inches(10, 8) # задаем размер
```

```
plt.plot([1,4,9,16]) # создаем график, где по оси X будут идти ц
plt.scatter([5,7], [7,8]) # создание графика точек с координатам
plt.show() # показать график
plt.xlabel('Time', fontsize = 20, color = 'white') # задать пара
plt.ylabel('Millions', fontsize = 40, color = 'white') # задать
plt.legend(['Salary', 'Salary in Data Science', 'Just dots']) #
```

Как на одном канвасе нарисовать несколько графиков?

```
fig.add_subplot(2,2, 1) # делим полотно на части, определяем нач
# график №1
fig.add_subplot(2,2, 2) # определяем начало второй части
# график №2
fig.add_subplot(2,1, 2) # меняем количество разрезов канваса
plt.tight_layout() # задать расстояние между графиками
```