KARPOV.COURSES >>>
КОНСПЕКТ

> Конспект > 6 урок > Метод максимума правдоподобия и ядерная оценка плотности

> Оглавление

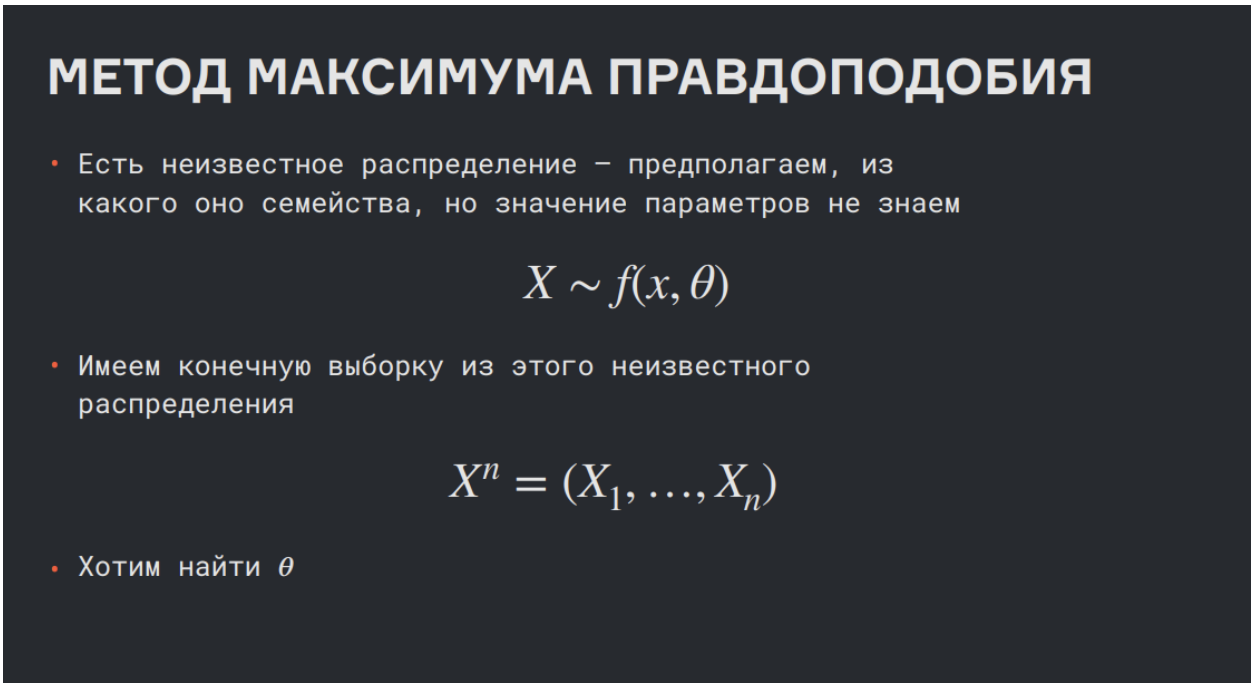
> Метод максимума правдоподобия

Давайте рассмотрим два способа оценки плотности это - метод максимума правдоподобия и ядерная оценка плотности, а еще поговорим о вероятностном подходе линейной регрессии и про то как можно искать аномалии.

Сначала давайте вспомним в чем задача статистики. Каждый раз жизни мы работаем с какими-то конечными выборками, которые нам удалось собрать, это может быть датасет с ценами на дома, а может быть клики на баннеры от наших пользователей.

Все эти данные чаще всего приходят из одной генеральной совокупности. Однако ее распределения мы не знаем, но мы можем всего лишь наблюдать ее конечные выборки, и задача статистики состоит в том, чтобы оценить параметры это генеральной совокупности, и что-то с ними дальше сделать.

В этом уроке будем заниматься похожей задачей, а именно оценивать распределения для нашей выборки. Случайная величина, с которой мы работаем, приходит с какого-то распределение, и иногда мы можем предположить какого вида это распределение. В таком случае получается, что мы не знаем всего лишь параметров этого распределения, например это могло быть нормальное распределение в котором мы не знали среднее и дисперсию, в таком случае наша задача сводится к оценке параметров распределения по нашей выборке. Это может быть например датасет с кликами на баннеры, тогда мы будем восстанавливать параметр из распределения Бернулли. Ранее была рассмотрена похожая задача, когда мы строили доверительные интервалы на статистике, но теперь мы будем оценивать параметры и делать точечные оценки. Собственно метод максимума правдоподобия и поможет это сделать.



Далее давайте посмотрим на пример на картинке ниже.

ВЕРОЯТНОСТЬ КЛИКА НА БАННЕР

- Клик на баннер – бинарная величина
- Предположим распределение Бернулли
- Хотим найти параметр p

- Выборка: [1, 0, 0, 1, 0, 1, 0, 0, 0, …]
- 25 единиц и 140 нулей

Давайте попробуем найти это вероятность ***p*** по нашей конкретной выборке. Для этого нам нужно разобраться, что такое правдоподобие.

ПРАВДОПОДОБИЕ

- Вероятность «пронадблюдать» данные, которые мы «пронаблюдали», как если бы мы знали параметры распределения

$$L(X^n, \theta) = \prod_{i=1}^{n} f(X_i, \theta)$$

- В случае непрерывных случайных величин – плотности

И теперь попробуем применить это к нашим кликам на баннер. Для этого сначала давайте разберемся с тем как будет выглядеть отдельная вероятность встретить конкретные значения. Значение может быть равно 1 или 0.

ВЕРОЯТНОСТЬ КЛИКА НА БАННЕР

- Предположим распределение Бернулли

$$f(X_i, \theta) = f(X_i, p) = \mathbb{P}(X_i) = p^{X_i}(1-p)^{1-X_i}$$

- Правдоподобие:

$$L(X^n, \theta) = \prod_{i=1}^{n} p^{X_i}(1-p)^{1-X_i}$$

С правдоподобием мы разобрались и можно двинуться дальше, тогда переходим к методу максимума правдоподобия.

МЕТОД МАКСИМУМА ПРАВДОПОДОБИЯ

- Метод максимума правдоподобия – найти параметры, при которых правдоподобие максимально

$$\hat{\theta}_{MLE} = \arg\max_{\theta} L(X^n, \theta)$$

- Ищем такие параметры, при которых вероятность «пронадблюдать» данные, которые мы «пронаблюдали», максимальна

МЕТОД МАКСИМУМА ПРАВДОПОДОБИЯ

- С произведением работать неудобно – логарифмируем

$$\log L(X^n, \theta) = \sum_{i=1}^{n} \log f(X_i, \theta)$$

- Валидно, так как делаем монотонное преобразование – точка оптимума сохраняется

$$\hat{\theta}_{MLE} = \arg\min_{\theta} \log L(X^n, \theta)$$

А теперь применим это к нашим баннерам.

ВЕРОЯТНОСТЬ КЛИКА НА БАННЕР

- Посчитаем логарифм правдоподобия:

$$\log L(X^n, \theta) = \sum_{i=1}^{n} X_i \log p + (1 - X_i)\log(1 - p) =$$

$$= n_+ \log p + n_- \log(1 - p)$$

ВЕРОЯТНОСТЬ КЛИКА НА БАННЕР

- Найдём минимум:

$$\log L(X^n, \theta) = n_+ \log p + n_- \log(1 - p)$$

$$(\log L(X^n, \theta))' = \frac{n_+}{p} - \frac{n_-}{1 - p} = 0$$

$$n_+ - n_+ p = n_- p \qquad p = \frac{n_+}{n}$$

ВЕРОЯТНОСТЬ КЛИКА НА БАННЕР

- Оценка значения p по выборке:

$$p = \frac{n_+}{n}$$

- То есть вероятность клика на баннер равна доля кликов в выборке!

То что мы рассмотрели выше кажется интуитивным, но важно отметить, что мы нашли это используя метод максимума правдоподобия. Мы предположили из какого распределения пришли данные, записали правдоподобие, т.е. вероятность встретить такие данные, которые мы встретили в реальности. Далее мы взяли логарифм от правдоподобия для удобства и промаксимизировали этот логарифм, тем самым нашли параметры в нашем распределении, при которых вероятность встретить такие данные была наибольшей. Так мы и оценили параметры нашего распределения.

> Линейная регрессия

Ранее в курсе была рассмотрена линейная регрессия. Выглядит она следующим образом.

- Линейная модель:

$$y = w^T x = w_0 + \sum_{i=1}^{d} w_i x_i$$

- Квадратичная ошибка:

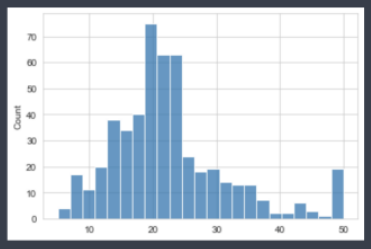
$$L(w, X) = \sum_{i=1}^{N} (y_i - w^T x_i)^2 \to \min_w$$

Давайте возьмем один известный датасет по предсказанию цен на дома. Нам не очень важно, что там за признаки, а нужно лишь обучить линейную регрессию. Как мы можем увидеть, распределение таргетов похоже на нормальное, но немного перекошенное.

```
X.shape, y.shape
```

```
((506, 13), (506,))
```

```
sns.histplot(y)
plt.show()
```



The Boston Housing Dataset

Далее мы воспользуемся библиотекой scikit learn, чтобы обучить линейную регрессию. Обратите внимание, что здесь нет регуляризации, также не забудем

отнормировать наши признаки, и получим следующие параметры, которые нам пригодятся далее.

```
from sklearn.preprocessing import StandardScaler

X = StandardScaler().fit_transform(X)
```

```
from sklearn.linear_model import LinearRegression

coefs_lr = LinearRegression().fit(X, y).coef_
```

```
coefs_lr
```

```
array([-0.92814606,  1.08156863,  0.1409    ,  0.68173972, -2.05671827,
        2.67423017,  0.01946607, -3.10404426,  2.66221764, -2.07678168,
       -2.06060666,  0.84926842, -3.74362713])
```

- Подойдём к модели иначе – нет функции потерь
- y_i – случайная величина
- Хотим знать распределение:

$$p(y_i \mid x_i, \theta)$$

- Сможем применять:

$$\hat{y}_i = \arg\max_y p(y \mid x_i, \theta)$$

$$\hat{y} = \mathbb{E}y = \sum_y y p(y \mid x_i, \theta)$$

- Предположим, что зависимость линейная и есть шум из нормального распределения

$$y_i = w^T x_i + \epsilon \qquad \epsilon \sim \mathbb{N}(0, \sigma^2)$$

$$p(y_i \mid x_i, \theta) = \mathbb{N}(w^T x_i, \sigma^2)$$

$$\theta = \{w, \sigma\}$$

- Откуда взять θ (а точнее w)?

- Попробуем понять, что даст максимизация правдоподобия

$$\log p(y \mid X, \theta) = \sum_{i=1}^{N} \log \mathbb{N}(y_i \mid w^T x_i, \sigma^2)$$

Давайте попробуем это сделать сразу в коде. Здесь мы будем пользоваться пакетами для оптимизации, а для этого нам нужно будет вычислять само правдоподобие. Мы знаем, что все наши параметры в системе - дисперсия нашего шума, свободный коэффициент и веса линейной зависимости.

```python
from scipy.stats import norm

def calc_LL(theta):
    std, w0, w = theta[0], theta[1], theta[2:]

    w = w.reshape((-1, 1))
    y_pred = X.dot(w) + w0
    y_pred = y_pred.reshape(-1)

    LL = np.sum(norm.logpdf(y, y_pred, std))

    return -LL
```

```python
theta0 = np.random.randn(2+data.shape[1])
theta0[0] = 1
```

```python
calc_LL(theta0)
```

```
175539.934750437
```

Теперь когда у нас есть функция, которая может вычислять логарифм правдоподобия, давайте попробуем ее минимизировать.

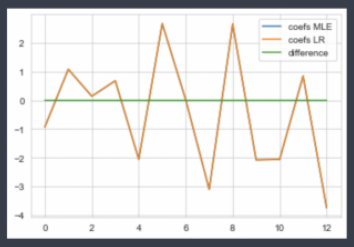
```python
from scipy.optimize import minimize

solution = minimize(calc_LL, theta0, method='L-BFGS-B')
solution.message
```

```
'CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH'
```

```python
coefs_ll = solution.x[2:]
```

Мы нашли минимум нашей функции. Помним, что нулевой параметр равен дисперсии нашего шума, а первый параметр свободному коэффициенту. Поэтому возьмем все значения наших параметров начиная со второго и сравним их с теми, что получили когда применяли линейную регрессию из коробки.

```
plt.plot(coefs_ll, label='coefs MLE')
plt.plot(coefs_lr, label='coefs LR')
plt.plot(coefs_lr - coefs_ll, label='difference')
plt.legend(loc=1)
plt.show()
```



Как мы видим если нарисовать их на одном графике, то окажется, что они совпадут, поэтому еще и нарисуем разность весов и получится, что она лежит около нуля. То есть мы получили одинаковое решение для задач линейной регрессии в этих двух подходах. Давайте попробуем понять почему так вышло.

- Попробуем понять, что даст максимизация правдоподобия

$$\log p(y \mid X, \theta) = \sum_{i=1}^{N} \log \mathbb{N}(y_i \mid w^T x_i, \sigma^2) =$$

$$= \sum_{i=1}^{N} \log \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - w^T x_i)^2}{2\sigma^2}\right) =$$

$$= \sum_{i=1}^{N} -\frac{1}{2}\log(2\pi) - \log\sigma - \frac{(y_i - w^T x_i)^2}{2\sigma^2} =$$

$$= \sum_{i=1}^{N} -\frac{1}{2}\log(2\pi) - \log\sigma - \frac{(y_i - w^T x_i)^2}{2\sigma^2} =$$

$$= C(w) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - w^T x_i)^2 \to \max_w$$

$$= \sum_{i=1}^{N} -\frac{1}{2}\log(2\pi) - \log\sigma - \frac{(y_i - w^T x_i)^2}{2\sigma^2} =$$

$$= C(w) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - w^T x_i)^2 \to \max_w$$

$$L(w, X) = \sum_{i=1}^{N}(y_i - w^T x_i)^2 \to \min_w$$

- Вероятностная интерпретация квадратичной ошибки:
  - Линейная зависимость целевой переменной от признаков
  - Ошибки распределены нормально
- Метод максимума правдоподобия помог найти значение параметров

- Если предположить в вероятностной модели нормальное распределение параметров p(w)
- Рассмотреть p(y, w | X)
- И найти моду этого распределения, то получим линейную регрессию с L_2-регуляризацией!

> Поиск аномалий

Давайте попробуем решить задачу поиска аномалий используя подход, который мы изучили ранее.

В выборке бывают выбросы или аномалии, если удалить аномалии из нашей выборки, то это может положительно сказаться на качестве нашей модели, так как она не будет обучаться под эти аномалии, а еще аномалии могут смещать наш средний прогноз. Возникает важный вопрос, о том как найти эти аномалии, и удалить их, чтобы решать задачу без них.

Попробуем воспользоваться вероятностным подходом.

- Оценим распределение над признаками P(X)
- Низкая вероятность встретить конкретный набор признаков (объект) – аномалия
- Как оценить распределение над признаками?

Как можно понять, что можно воспользоваться методом максимума правдоподобия, а значит нам нужно предположить с какого семейства к нам пришли данные. Предположим, что они пришли из нормального распределения, часто в жизни так и бывает, хоть иногда данные несколько скошены по каким-то признакам, но если вы будет строить scatter plot, то часто будет некое облако точек похожее на нормальное распределение, если конечно это вещественные значения.

Здесь мы будем работать многомерным нормальным распределением, оно отличfется от одномерного, с которым мы работали ранее, дисперсия здесь заменяется на матрицу ковариации, и там где мы раньше делили на квадрат дисперсии мы будем умножать на обратную матрицу. Чтобы оценить

распределение над нашими признаками нам нужно оценить параметры этого распределения, а именно вектор средних и ковариационную матрицу. Давайте попробуем это сделать аналитически.

- Пусть признаки распределены нормально

$$P(x_i \mid \theta) = \frac{1}{\sqrt{(2\pi)^d \mid \Sigma \mid^2}} e^{-(x_i-\mu)^T \Sigma^{-1}(x_i-\mu)}$$

- Найдём параметры μ, Σ с помощью метода максимума правдоподобия

$$\log L(X, \theta) = \sum_{i=1}^{N} \log \left(\frac{1}{\sqrt{(2\pi)^d \mid \Sigma \mid^2}} e^{-(x_i-\mu)^T \Sigma^{-1}(x_i-\mu)} \right) =$$

$$= \sum_{i=1}^{N} -\frac{d}{2} \log 2\pi - \frac{N}{2} \log \mid \Sigma \mid - (x_i - \mu)^T \Sigma^{-1} (x_i - \mu) =$$

$$= -\frac{Nd}{2} \log 2\pi - \frac{N}{2} \log \mid \Sigma \mid - \sum_{i=1}^{N} (x_i - \mu)^T \Sigma^{-1} (x_i - \mu)$$

Подсчитаем градиент по параметрам средних и получится следующая запись. Здесь мы пользуемся правилами для диффенцирования в многомерном случае, но могли бы это делать в одномерном случае по каждому из индексов в нашем векторе μ

$$\nabla_\mu \log L(X, \theta) = \sum_{i=1}^{N} 2\Sigma^{-1}(x_i - \mu)$$

$$\nabla_\mu \log L(X, \theta) = 0 \Rightarrow \hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

Подобную операцию проделаем и с нашей ковариационной матрицей.

$$\Lambda = \Sigma^{-1}$$

$$\nabla_\Lambda \log L(X, \theta) = \frac{N}{2}\Lambda^{-1} - \sum_{i=1}^{N} (x_i - \mu)(x_i - \mu)^T$$

$$\nabla_\Lambda \log L(X, \theta) = 0 \Rightarrow \hat{\Sigma} = \Lambda^{-1} = \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)(x_i - \mu)^T$$

$$\hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

$$\hat{\Sigma} = \Lambda^{-1} = \frac{1}{N} \sum_{i=1}^{N} (x_i - \hat{\mu})(x_i - \hat{\mu})^T$$

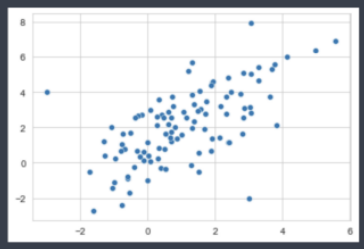
Теперь попробуем применить это на практике. Для возьмем некоторые синтетические данные, где данные сгенерированы из многомерного нормального распределения, но только с двумя координатами, чтобы можно было это

визуализировать. Возьмем вектор среднего и некоторую ковариационную матрицу, потом сравним их с теми которые вычислим, и потом добавим в выборку два объекта, которые очевидно будут являться аномалиями, так как далеко лежат от центра нашего распределения и от всех остальных объектов.

```
np.random.seed(123)
```

```
mu_real = [1, 2]
sigma_real = [[2, 2], [2, 4]]
data = np.random.multivariate_normal(mu, sigma, 100)
data = np.vstack((data, [[-3, 4], [3, -2]]))
```

```
sns.scatterplot(x=data[:, 0], y=data[:, 1])
plt.show()
```



Сначала вычислим среднее, по формуле которую мы ранее вывели. Получится значения которого довольно близки, к тем значениям которые мы использовали при генерации данных. Заметим, что мы не получим тоже самое значение, во первых потому что не так много объектов в нашей выборке - всего 100, а во вторых кроме этих 100 объектов есть еще 2 аномалии, которые вносят свой вклад

$$\hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

```
mu = data.mean(axis=0)
```

```
mu
```

```
array([1.06727279, 2.14460255])
```

```
mu_real
```

```
[1, 2]
```

По более сложной формуле вычислим ковариацию.

$$\hat{\Sigma} = \Lambda^{-1} = \frac{1}{N} \sum_{i=1}^{N} (x_i - \hat{\mu})(x_i - \hat{\mu})^T$$

```
sigma = (data - mu).T.dot(data - mu) / len(data)
```

```
sigma
```

```
array([[2.44809715, 2.08319052],
       [2.08319052, 4.38111457]])
```

```
sigma_real
```

```
[[2, 2], [2, 4]]
```

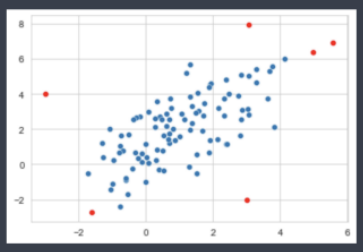
Теперь когда мы оценили среднее и ковариационную матрицу в нашем нормальном распределении. Мы можем вычислять плотности для каждого нашего объекта. Давайте вычислим плотности по каждому объекту. Возьмем из них 5% самых наименьших, т.е. самых нетипичных объектов и назовем их выбросами.

Изобразим все это на картинке.

```
from scipy.stats import multivariate_normal

pdf_data = multivariate_normal(mu, sigma).pdf(data)
```

```
threshold = np.percentile(pdf_data, q=5)
mask = pdf_data < threshold
```

```
sns.scatterplot(x=data[~mask, 0], y=data[~mask, 1])
sns.scatterplot(x=data[mask, 0], y=data[mask, 1], color='red')
plt.show()
```



Как мы видим на графике, у нас красным подсветились аномалии.

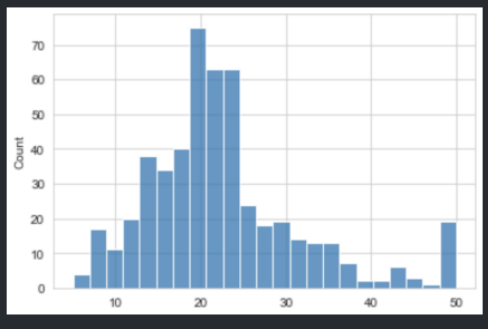
Важно отметить на практике, что удаление аномалий в нашей выборке не всегда улучшает качество модели, но нужно это пробовать. В нашем случае мы искали среднее и ковариационную матрицу аналитически и потом их вычисляли. Это довольно трудоемко и не всегда возможно. Даже для нормального распределения это сделать было довольно сложно, а бывают еще более сложные распределения. Можно было использовать градиентный спуск и автодифференцирование, чтобы найти этот максимум.

> Ядерная оценка плотности

Метод максимума правдоподобия является параметрическим и предполагал, что мы знаем из какого семейства пришли данные и искали для него параметры. Но может быть ситуация когда непонятно из какого распределения пришли данные - здесь необходим непараметрический подход - ядерная оценка плотности. Мы уже работали с чем-то похожим когда строили гистограммы.

ГИСТОГРАММА

- Визуализация произвольного одномерного распределения
- Ось значений разбивается на интервалы, для каждого интервала подсчитывается число / доля объектов выборки, попадающих в интервал
- Интервалы изображаются столбиками, подсчитанной высоты



ЯДЕРНАЯ ОЦЕНКА ПЛОТНОСТИ

- Непараметрический способ оценки распределения случайной величины
- Идея – оценим плотность в любой точки через то, насколько близко объекты выборки
- Ядро – функция для оценки близости точек

Давайте разберем простой пример со ступенчатым ядром.

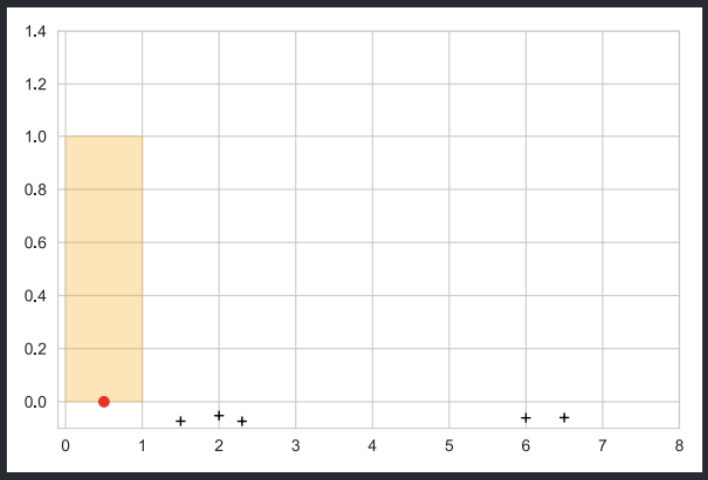
- Возьмём ступенчатую (равномерную функцию):

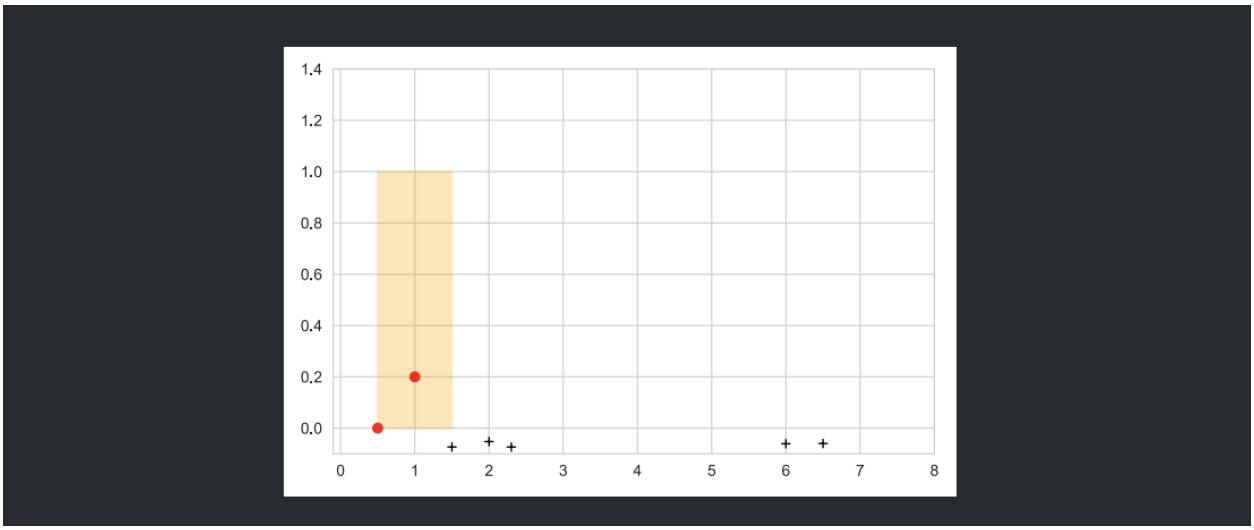
$$K(z) = 1, |z| \leq 0.5$$

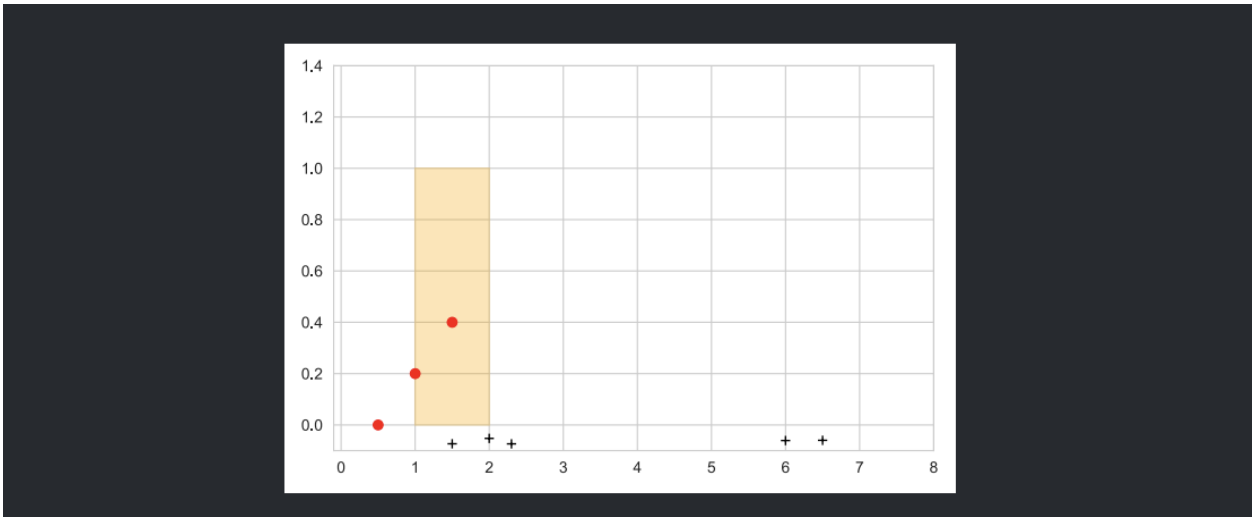
- Для каждой точки оценим, сколько объектов попадает в пределы этой ступенчатой функции, если её центр поставить в нашу точку

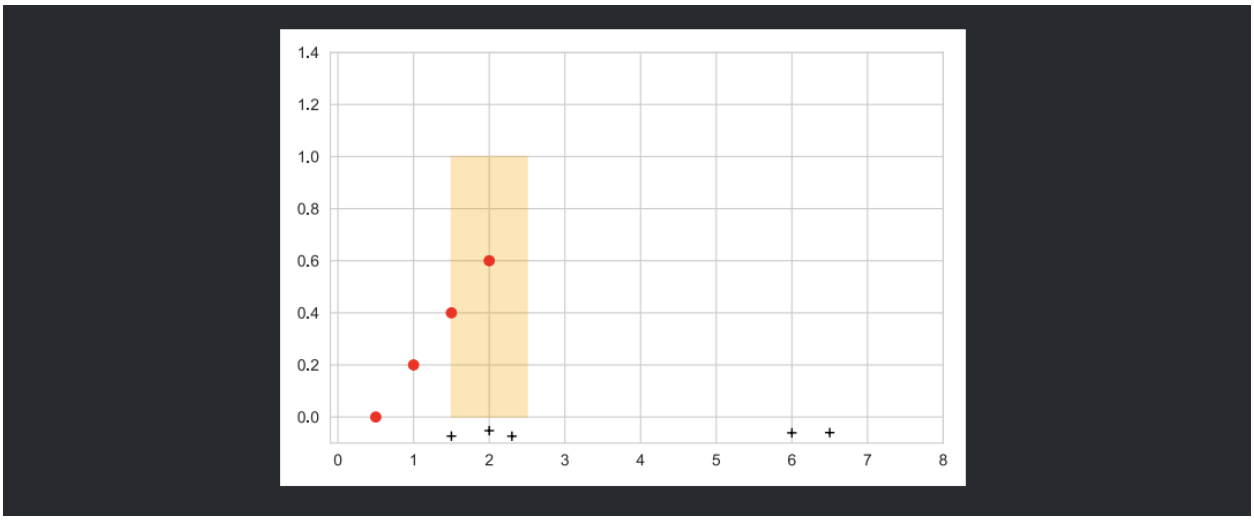
$$\sum_i K(x - x_i)$$

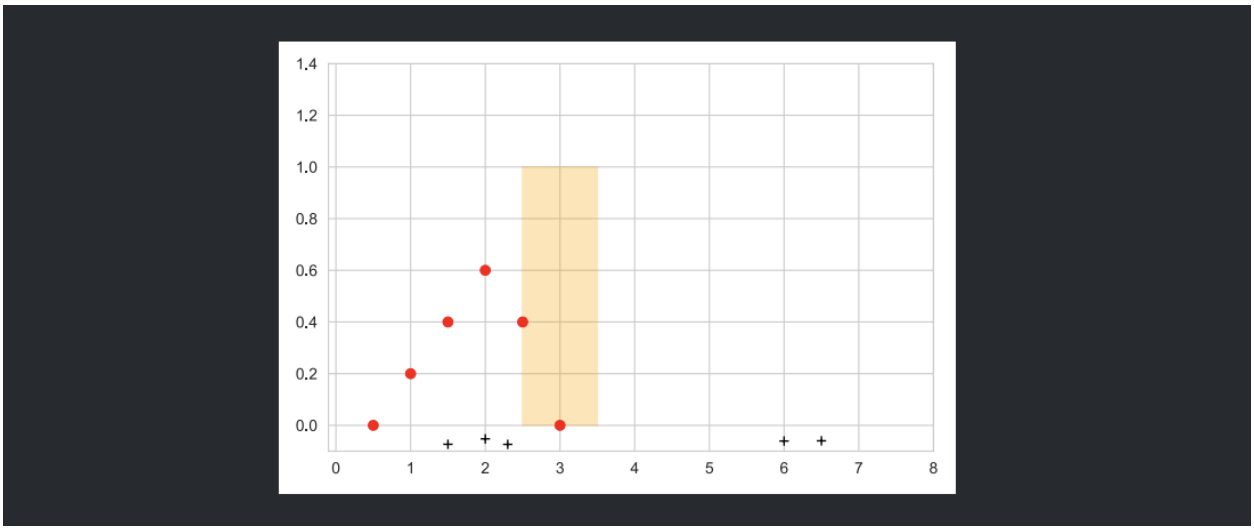
Возьмем простую выборку состоящую из 5 объектов на графике отмечены крестиками и пройдемся этим ядром по всей одномерной оси и подсчитаем значения.

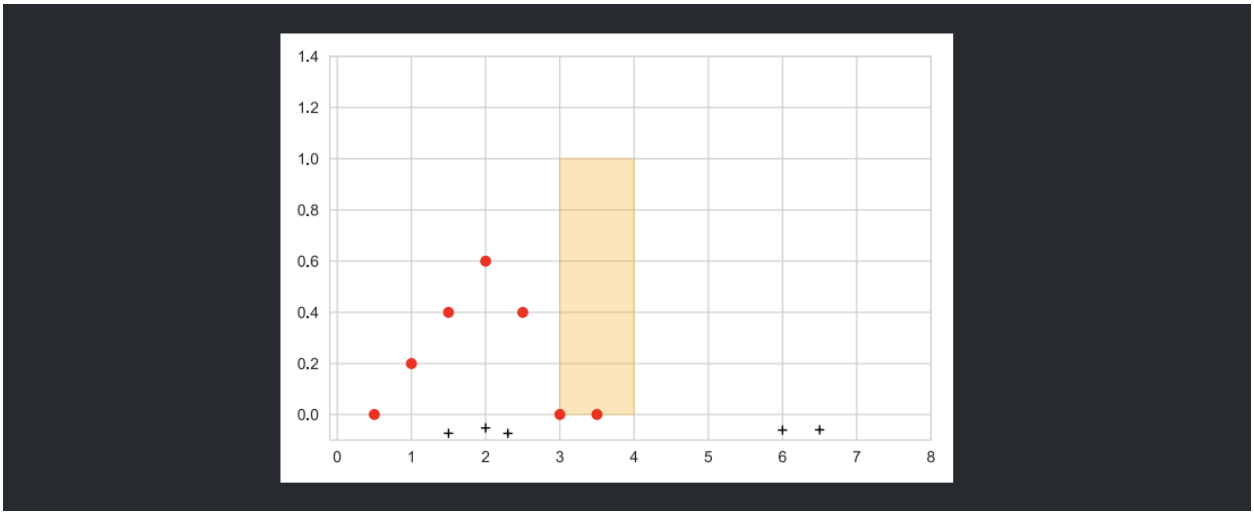


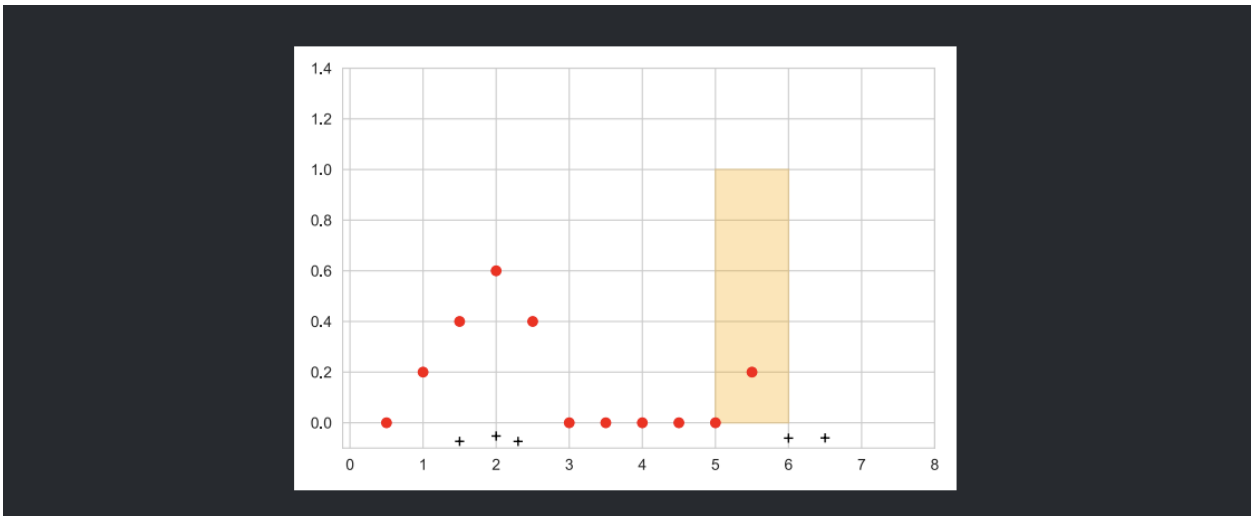

1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0 1 2 3 4 5 6 7 8

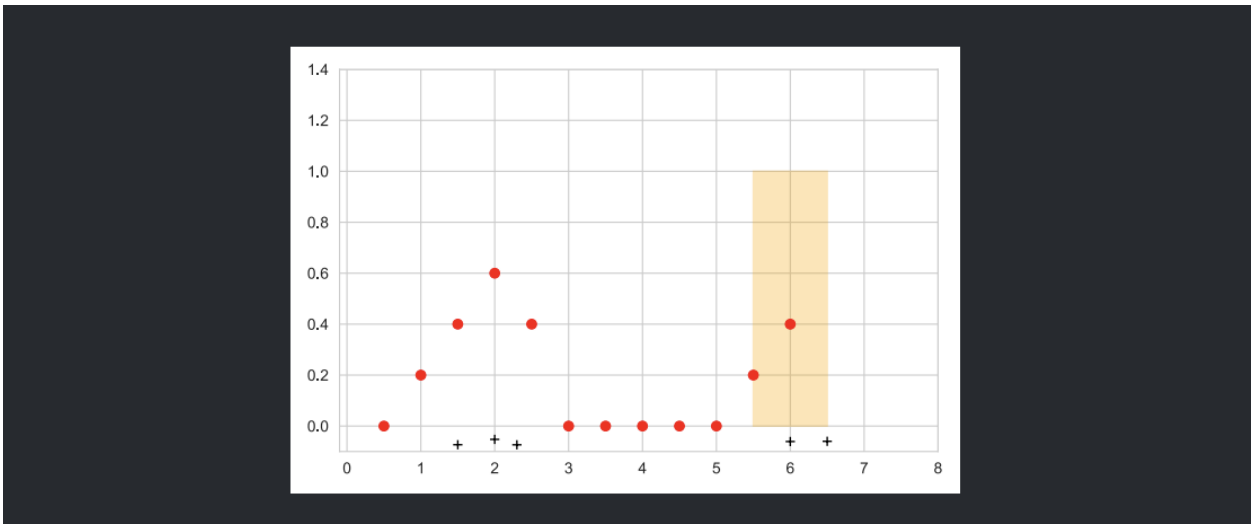

1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0 1 2 3 4 5 6 7 8

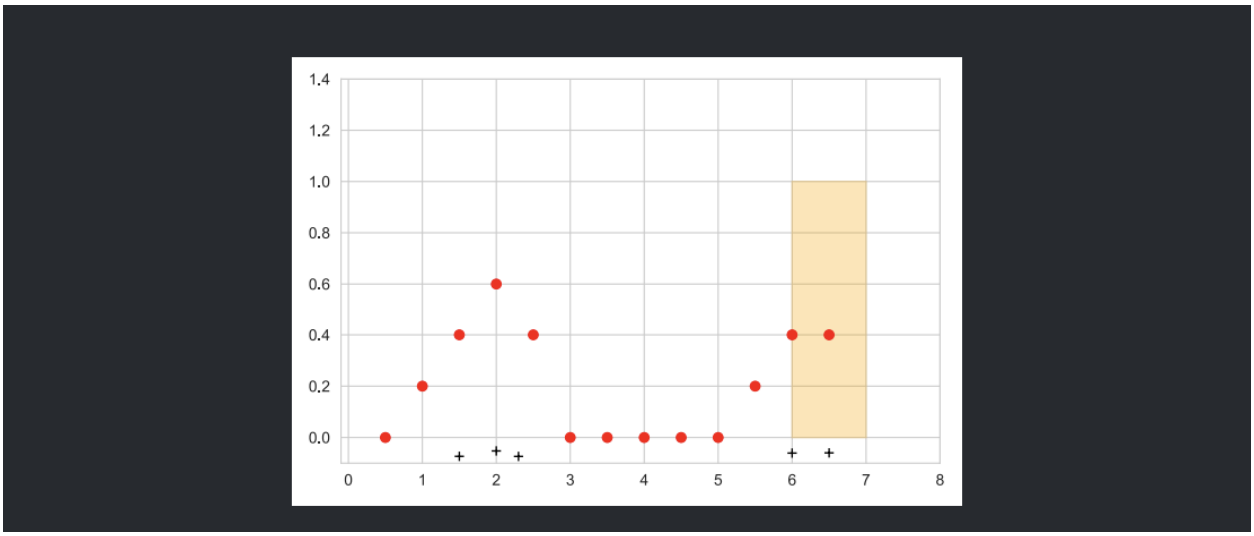

1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0 1 2 3 4 5 6 7 8


1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0 1 2 3 4 5 6 7 8

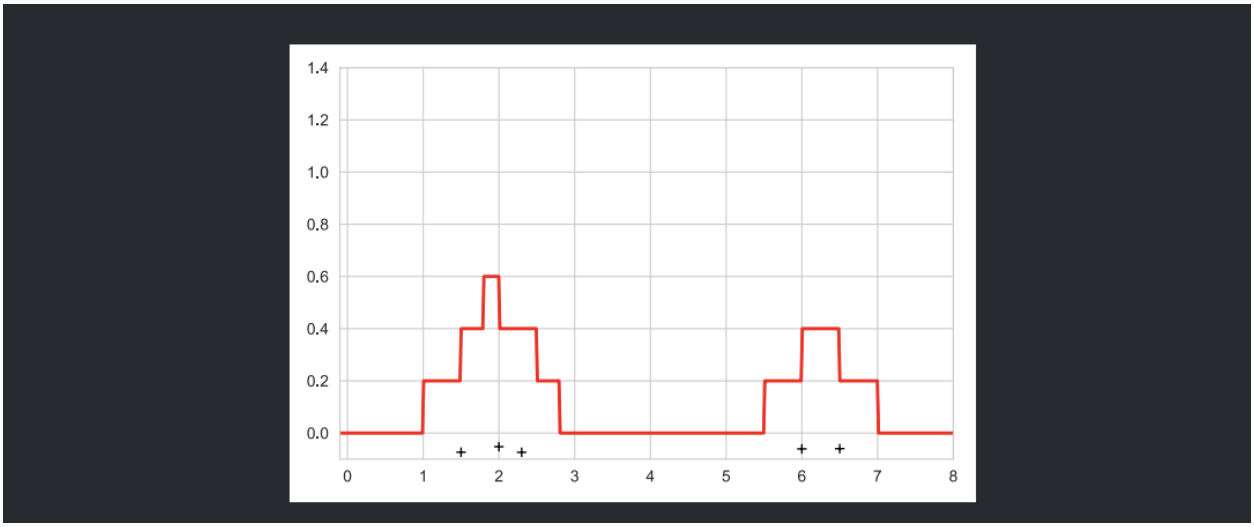

1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0 1 2 3 4 5 6 7 8


1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0 1 2 3 4 5 6 7 8





Если все это соединим, то получим такой график.



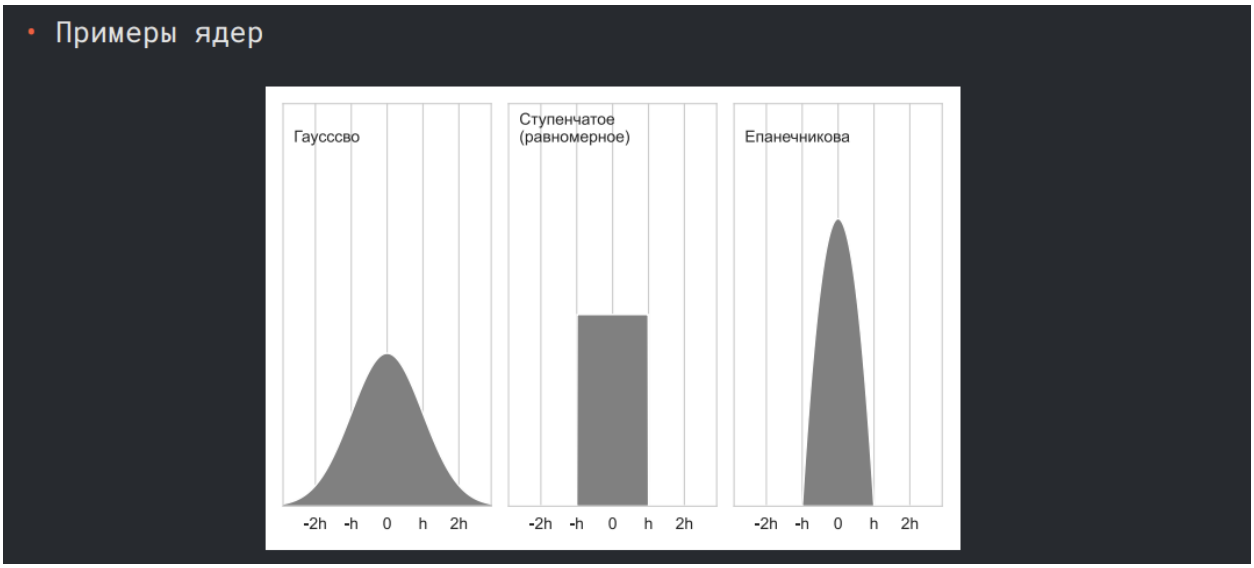
Это уже похоже на некоторый график плотности наших точек, который мы наблюдаем, но при условии, что мы все отнормируем и сумме все это будет давать 1. Мы оценили в простом случае плотность нашего распределение с помощью вот такого простого ядра.

- В общем виде в одномерном случае:

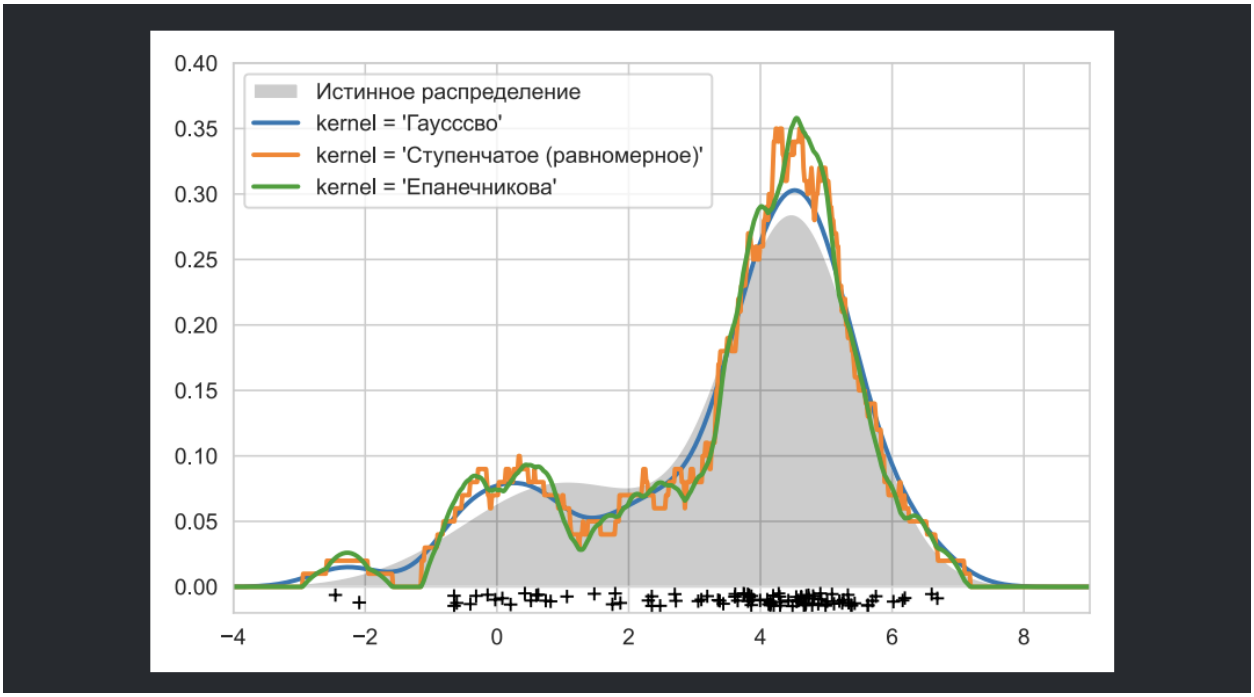
$$\hat{\rho}_N(x) = \frac{1}{Nh} \sum_{i=1}^{N} K\left(\frac{X_i - x}{h}\right)$$

- h – ширина окна
- K – ядро

Ядер можно придумать большое количество, но требование, чтобы они были симметричными и интеграл под ними был равен 1. Самые популярные можно посмотреть на картинке.



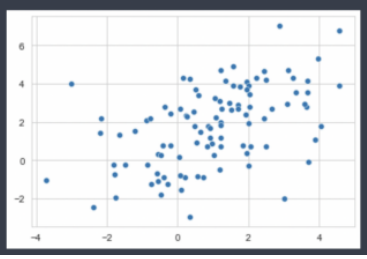
Во всех ядрах мы можем заметить, что они все пытаются быть максимальными в окрестности нуля и отдаляясь от нуля их значения постепенно уходят в ноль. Посмотрим на пример с большим количеством данных, где данные пришли из двух нормальных распределений. Серым изображено истинное распределение, остальными цветами обозначены оценки ядерной оценки плотности с тремя разными ядрами. Можно увидеть, что они дают немного разные результаты, хотя и похожи.



Теперь вернемся к поиску аномалий. Теперь будем оценивать плотность при помощи ядерной оценки плотности без предположения о том из какого распределения пришли наши данные.

```
mu_real = [1, 2]
sigma_real = [[2, 2], [2, 4]]
data = np.random.multivariate_normal(mu, sigma, 100)
data = np.vstack((data, [[-3, 4], [3, -2]]))
```

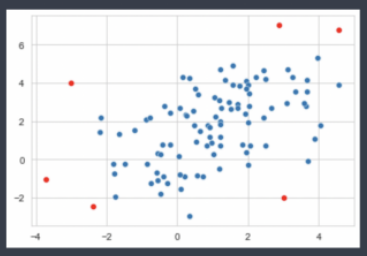
```
sns.scatterplot(x=data[:, 0], y=data[:, 1])
plt.show()
```



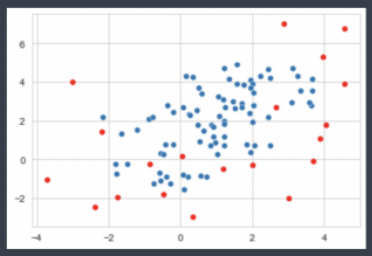
```
from sklearn.neighbors import KernelDensity

kde = KernelDensity(kernel='epanechnikov', bandwidth=2).fit(data)
log_dens = kde.score_samples(data)
```

```
threshold = np.percentile(log_dens, q=5)
mask = log_dens < threshold
```

```
sns.scatterplot(x=data[~mask, 0], y=data[~mask, 1])
sns.scatterplot(x=data[mask, 0], y=data[mask, 1], color='red')
plt.show()
```



```
kde = KernelDensity(kernel='epanechnikov', bandwidth=0.5).fit(data)
log_dens = kde.score_samples(data)

threshold = np.percentile(log_dens, q=5)
mask = log_dens <= threshold

sns.scatterplot(x=data[~mask, 0], y=data[~mask, 1])
sns.scatterplot(x=data[mask, 0], y=data[mask, 1], color='red')
plt.show()
```



Как можно заметить если мы возьмем маленькую ширину окна и опять же возьмем 5% , аномалии будут находится и внутри нашей выборки, это все потому что это чуть более разреженные области и в окрестности которых мало объектов и они тоже будут считаться за аномалии.

Также мы можем заметить, что оба раза искали по 5% в наших примеров, но в этом случае их стало больше, это все к тому что когда мы используем маленькую ширину окна, то у нас у многих точек вероятность получается одинаковая, и кода мы берем 5% объектов, то порог на которых мы отсекаем, совпадает по значению с вероятностью некоторого количества объектов. Поэтому в реальности когда мы сравниваем с порогом с учетом равенства, то можем получить большее количество объектов. Еще мы можем заметить что ядерная оценка плотности помогала искать нам объекты в не разреженных областях, а не только на удаленных объектах.