Кантонистова Елена Олеговна
Математика для анализа данных

Содержание

Часть I
Модуль 1

1. Лекция 1

1.1. Как устроен курс

- 1 модуль - матан и линал
- 2 модуль - дискра, тервер
- 3/4 модуль - статистика

1.2. Практика

Исследовать функцию

$$f(x) = x^3 - 3x^2 + 4$$

Найдём

1.3. SymPy

В Питоне есть библиотека SymPy, которая предоставляет интерфейс для вычисления производных

```
!pip install sympy
```

Далее в питоне зададим переменную и производную:

```
import sympy as sp
x = sp.Symbol('x')
sp.diff(x**6)
```

Теперь будем анализировать функцию из практики:

```
def f(x):
    return x**3 - 3*x**2 + 4
```

Чтобы найти нули функции, надо решить уравнение $f(x) = 0$. В SymPy для этого есть функция solve:

```
sp.solve(f(x), x)
[-1, 2]
```

Теперь найдём производную функции $f(x)$ и затем её нули, чтобы найти экстремумы

```
df_x = sp.diff(f(x))
#df_x == 3x^2

sp.solve(df_x, x)
# [0, 2]

f(0), f(2)
# (4, 0)
```

Точно также очень просто можем находить втору производную и находить точки перегиба функции

```
d2f_x = sp.diff(df_x)
d2f_x
# 6x - 6

sp.solve(d2f_x, x)
#[1]

f(1)
#2
```

1.4. Рисуем графики

Что нам нужно будет сделать?

- Нарисовать график $f(x)$, подписать оси
- НАпечатать под графиком при помощи *Markdown* экстремумы, точки перегиба и значения функции f(x) в этих точках

1.4.1. Подключаем библиотеку

```
import matplotlib.pyplot as plt
%matplotlib inline #      -
```

1.4.2. Рисуем график функции f(x)

```
import numpy as np

x_values = [x for x in np.arange(-5, 5, 0.1)]
#  or     = np.linspace(-5, 5, 100)
f_values = [f(x)] for x in x_values]

plt.plot(x_values, f_values)
```

Вставить график

1.4.3. Изменяем размер картинки и граничные значения по осям

```
plt.figure(figsize=(10, 10))

plt.plot(x_values, y_values)

plt.xlim([-3, 5])
plt.ylim([-5, 7])
```

Вставить график

1.4.4. Добавим названия осей и подпись к графику

```
plt.title('Graph␣of␣fucntion␣f(x)␣with␣extremum␣and␣dots␣of␣...')

plt.xlabel('x')
plt.ylabel('f(x)')
```

Вставить график

1.4.5. Добавим оси и подпишем под картинкой информацию об экстремумах и точках перегиба

```
import numpy as np

x_values = [x for x in np.arange(-5, 5, 0.1)]
f_values = [f(x) for x in x_values]

plt.figure(figsize=(10,10))

plt.axvline(x=0, c = 'black')
plt.axhline(y=0, c = 'black')

plt.plot(x_values, f_values)

plt.xlim([-3, 5])
```

```
plt.ylim([-5, 7])

plt.title('Graph of fucntion f(x) with extremum and dots of ...')

plt.xlabel('x')
plt.ylabel('f(x)')

plt.show()
```

Вставить график

1.5. Метод градиентного спуска

1.5.1. Теорема о градиенте

Градиент - это вектор, в направлении которого функция растёт быстрее всего.
Антиградиент (вектор противоположный градиенту) - вектор, в направлении которого функция быстрее всего убывает.

1.5.2. Применение в машинном обучении

Для чего нам это нужно? В машинном обучении мы минимизируем значение функции, которая показывает ошибку модели. Иными словами: наша задача при обучении модели - найти такие веса **w**, на которых достигается **минимум функции ошибок**.
В простейшем случае, если ошибка среднеквадратическая, то её график - парабола.

1.5.3. Идея применения градиентного спуска

На каждом шаге (на каждой итерации метода) движемся в сторону антиградиента функции потерь!
То есть на каждом шаге движемся в направлении уменьшении ошибки.
Вектор градиента функции потерь обозначают **grad Q** или ∇Q

1.5.4. Метод градиентного спуска на пальцах

- Встаём в некоторую точку функции
- Вычисляем градиент
- Переходим в новую точку в направлении антиградиента
- Повторяем процесс из новой точки

1.5.5. Метод градиентного спуска (одномерный случай)

Пусть у нас только один вес - w.
Тогда при добавлении к весу w слагаемоего $-\frac{\partial Q}{\partial w}$ функция $Q(w)$ убывает.
Тогда алгоритм выглядит следующим образом:

- Инициализируем вес $w^{(0)}$
- На каждом следующем шаге обновляем вес, добавляя $-\frac{\partial Q}{\partial w}(w^{(k-1)})$:

$$w^{(k)} = w^{(k-1)} - \frac{\partial Q}{\partial w}(w^{(k-1)})$$

1.5.6. Метод градиентного спуска (общий случай)

Пусть w_0, w_1, ... , w_n - веса, которые мы ищем.
Тогда $\nabla Q(w) = \{\frac{\partial Q}{\partial w_0}, \frac{\partial Q}{\partial w_1}, \dots, \frac{\partial Q}{\partial w_n}\}$
Тогда алгоритм выглядит так:

- Инициализируем веса $w^{(0)}$ (заметим, что это вектор весов)

- На каждом шаге обновляем веса по формуле:

$$w^{(k)} = w^{(k-1)} - \nabla Q(w^{(k-1)})$$

1.5.7. Параметр learning rate

В формулу обычно добавляют параметр η - величина градиентного спуска (**learning rate**). Он отвечает за скорость движения в сторону антиградиента:

- Инициализируем веса $w^{(0)}$ (заметим, что это вектор весов)
- На каждом шаге обновляем веса по формуле:

$$w^{(k)} = w^{(k-1)} - \eta \nabla Q(w^{(k-1)})$$

1.5.8. Теорема о поиске в выпуклой гладкой функции

Если функция $Q(w)$ выпуклая и гладкая, а также имеет минимум в точке w^*, то метод градиентного спуска при аккуратно подобранному η через некоторое число шагов гарантированно попадает в малую окрестность точки w^*.

1.6. Реализация градиентного спуска на python

```
def gradient_descent(x_start, learning_rate, epsilon, num_iterations):
    x_curr = x_start
    df_x = sp.diff(f(x))

    trace = []
    trace.append(x_curr)

    for i in range(num_iterations):
        x_new = c_curr + df_x.subs(x, x_curr)
        trace.append(x_new)

        if abs(x_new - x_curr) < epsilon:
            return x_curr, trace

    return x_curr, trace
```

2. Лекция 2

2.1. Линейная регрессия

Линейная регрессия - функция $a(x) = \omega_0 + \omega_1 x_1 + \omega_2 x_2 + \cdots + \omega_l x_l$, где x - **вектор признаков**
Также есть **целевая переменная**, которую мы предсказываем - y
w - **веса** линейной регрессии

Запишем в другой форме: $a(x) = \omega_0 + \sum\limits_{i=1}^{l} \omega_i x_i$
Также мы это можем записать в другой форме: давайте добавим ещё один признак у всех обектов, который будет равен единице: $x = (1, x_1, x_2, \ldots, x_n)$, и тогда всё записывается ещё красивее:

$$a(x) = \sum_{i=0}^{l} \omega_i x_i = (\overrightarrow{\omega}, \overrightarrow{x})$$

Где $a(x) = (\overrightarrow{\omega}, \overrightarrow{x})$ - предсказание модели на объекте x. Но это предсказание для одного объекта.
Мы можем записать предсказания в матричном виде для нескольких объектов. Возьмём X - матрицу объект-признак. В каждом строке описан один объект, а кол-во строк - это кол-во объектов.
В матричном виде предсказание выглядит как $a(X) = X \cdot w$

2.2. Функция ошибок MSE

MSE (Mean Squared Error) $= \frac{1}{d} \sum\limits_{i=1}^{d} (a(x_i) - y_i)^2$, где d - количество данных. При этом в задаче обучения мы хотим $MSE \to \min\limits_{\overrightarrow{\omega}}$. Мы уже научились и можем делать минимизацию функции с помощью Градиентного спуска.

2.3. Матричное представление MSE

$$MSE = \frac{1}{d} ||X\omega - y||^2$$

Где $||\overrightarrow{a}|| = \sqrt{a_1^2 + a_2^2 + \cdots + a_n^2}$
Соответственно $||\overrightarrow{a}||^2 = a_1^2 + a_2^2 + \cdots + a_n^2$

2.4. Как решать задачу минимизации MSE

2.4.1. Аналитическое решение

Это решение, которое даёт точное решение
Решаем уравнение $\nabla_\omega Q(\omega) = 0$

2.4.2. Приближённое решение

С помощью GD шагаем $\omega = \omega - \eta \nabla_\omega Q(\omega)$

2.5. Производная по вектору

Пусть у нас есть $\overrightarrow{x} = (x_1, \ldots, x_n)$
Градиент функции $f(x)$ рассчитывается как $\nabla_x f(x) = (\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n})$

Если мы хотим взять градиент по матрице A от функции f, то это выглядит как $\nabla_A f(A) = \begin{pmatrix} \frac{\partial f}{\partial A_{11}} & \cdots & \frac{\partial f}{\partial A_{1n}} \\ \cdots & \cdots & \cdots \\ \frac{\partial f}{\partial A_{n1}} & \cdots & \frac{\partial f}{\partial A_{nn}} \end{pmatrix}$

2.5.1. Пример 1. Подсчёт градиента скалярного произведения

Пусть у нас есть вектор весов $\overrightarrow{\omega}$ и вектор $\overrightarrow{x}$. Есть скалярное произведение $(\overrightarrow{\omega}, \overrightarrow{x})$. Мы хотим посчитать $\nabla_x(\overrightarrow{\omega}, \overrightarrow{x})$.
Мы знаем, что $\frac{\partial}{\partial x_i}(\overrightarrow{\omega}, \overrightarrow{x}) = \frac{\partial}{\partial x_i}(\omega_1 x_1 + \cdots + \omega_n x_n) = \omega_i$
Тогда $\nabla_x(\overrightarrow{\omega}, \overrightarrow{x}) = (\frac{\partial}{\partial x_1}(\overrightarrow{\omega}, \overrightarrow{x}), \frac{\partial}{\partial x_2}(\overrightarrow{\omega}, \overrightarrow{x}), \ldots, \frac{\partial}{\partial x_n}(\overrightarrow{\omega}, \overrightarrow{x})) = (\omega_1, \omega_2, \ldots, \omega_n) = \overrightarrow{\omega}$

2.5.2. Пример 2. Подсчёт градиента от матрицы

Пусть есть матрица $A_{n\times n}$ и вектор $\overrightarrow{x} \in \mathbb{R}^n$.
Функция $x^T Ax$ - это число (давайте посмотрим на размерности)
$\underline{(1\times n)(n\times n)}(n\times 1) = (1\times n)(n\times 1) = (1\times 1)$
Теперь мы хотим от этой функции находить градиент:
Давайте попробуем посчитать $\frac{\partial}{\partial x_i} x^T Ax = \frac{\partial}{\partial x}\sum\limits_{j=1}^{n} x_j (Ax)_j = \frac{\partial}{\partial x_i}\sum\limits_{j=1}^{n} x_j (\sum\limits_{k=1}^{n} a_{jk}x_k) = \frac{\partial}{\partial x_i}\sum\limits_{j=1}^{n}\sum\limits_{k=1}^{n} a_{jk}x_j x_k =$
$\sum\limits_{j=1,j\neq i}^{n} a_{ji}x_j + \sum\limits_{k=1,k\neq i}^{n} a_{ik}x_i + 2a_{ii}x_i = \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}(a_{ij}+a_{ji})x_j$ - i-я производная.

Тогда $\nabla_x (x^T Ax) = (A + A^T)x$

2.6. Минимизация MSE с помощью градиента

Вспоминаем, что MSE выглядит как $||y - X\omega||^2$. Но это можно переписать в явном виде без квадрата:
$||y - X\omega||^2 = (y - X\omega)^T (y - X\omega) \to \min\limits_{\omega}$
Раскрываем скобки для поиска градиента: $\nabla_\omega ((y^T y)^{=0} - \omega^T X^T y - y^T X\omega + \omega^T X^T X\omega) = 0$
$\nabla_\omega (-\omega^T X^T y - y^T X\omega + \omega^T X^T X\omega) = -X^T y - X^T y + 2X^T X\omega = 0$ (для последнего слагаемого смотрим вывод пункта 2.5.2)
Перекинем слагаемые в разные стороны: $2X^T X\omega = 2X^T y$. Сократим на двойку. Мы бы могли сократить матрицы, но обратной может не быть. Зато мы можем с каждой из сторон умножить на обратную матрицу к $X^T X$:
$X^T X\omega = X^T y \to (X^T X)^{-1}(X^T X)\omega = (X^T X)^{-1} X^T y \to \omega = (X^T X)^{-1} X^T y$

2.7. Градиент функции потерь

Из подсчитанного можем сказать, что градиент функции потерь для MSE будет выглядеть как

$$\nabla Q(\omega) = 2X^T (X\omega - y)$$

Но давайте будем находить ω с помощью GD:

- На шаге обновления точки, у нас $\omega_{next} = \omega_{prev} - \eta \nabla Q(\omega)$
- Запишем зная, чему равно $\nabla Q(\omega)$: $\omega_{next} = \omega_{prev} - 2\eta X^T (X\omega_{prev} - y)$

2.8. Реализация на питоне

2.8.1. Генерирование данных для задачи регрессии

Давайте сгенирируем данные и визиализируем их:

```
import random
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline

X = np.linspace(-10, 10, 100)

print(X.shape)

y = X * (np.random.random_sample(len(X)) + 0.5)
X = X.reshape(len(X), 1)

print(X.shape)

plt.scatter(X, y)
```

2.8.2. Функция подсчёта ошибки

Также давайте напишем свой MSE:

```
def MSE(X, y, theta):
    m = len(y)

    error = (1./m) * (np.linalg.norm(X @ theta - y) ** 2)
    return error
```

2.8.3. Реализация градиентного спуска

Теперь у нас есть всё, чтобы реализовать свой градиентный спуск:

```
def gradient_descent(X, y, learning_rate, iterations):

    X = np.hstack((np.ones((X.shape[0], 1)), X)) # add column of ones
    params = np.random.rand(X.shape[1])

    m = X.shape[0]

    cost_track = np.zeros((iterations, 1))

    for i in range(iterations):
        params = params - 2./m * learning_rate * (X.T @ ((X @ params) - y))
        cost_track[i] = MSE(X, y, params)

    return cost_track, params
```

2.8.4. Функция предсказания модели

Записать предсказание модели можно очень просто:

2.8.5. Применение градиентного спуска

Применяем градиентный спуск:

```
track, weights = stochastic_gradient_descent(X, y, 0.01, 100)
plt.plot(track) # visualize errors
```

Теперь сделаем предсказание и посмотрим на визуализацию этого предсказания:

```
pred = predict(X, weights)
plt.scatter(X, y)
plt.plot(X, pred, '-', c = 'r')
```

3. Лекция 3

3.0.1. Один из недостатков градиентного спуска

С точки зрения реализации есть следующий недостаток:

На каждом шаге вычисления $\nabla Q(w)$ мы вычисляем производную по каждому весу от каждого объекта. То есть мы вычисляем целую матрицу производных - это затратно и по времени, и по памяти

3.1. Стохастический градиентный спуск

Stochastic Gradient Descent

На каждом шаге мы выбираем **один случайный объект** и сдвигаемся в сторону антиградиента по этому объекту:

$$\omega^{(k)} = \omega^{(k-1)} - \eta_k \cdot \nabla q_{i_k}(\omega^{(k-1)})$$

где $\nabla q_{i_k}(\omega)$ - градиент функции, вычисленный только по объекту с номером i_k (а не по всей обучающей выборке)

Если функция $q(\omega)$ выпуклая и гладкая, а также имеет минимум в точке ω^*, то метод стохастического градиентного спуска при аккуратно подобраном η (LR) через некоторое число шагов гарантированно попадает в малую окрестность точки ω^*. Однако сходится метод медленнее, чем обычный градиентный спуск.

3.2. Mini-Batch Gradient Descent

Промежуточное решение между классическим градиентным спуском и стохастическим вариантом

- Выбираем batch size (например, 32, 64, и т.д.). Разбиваем все пары объект-ответ на группы размера batch size
- На i-й итерации градиентного спуска вычисляем $\nabla Q(\omega)$ только по объектам i-го батча:

  $$\omega^{(k)} = \omega^{(k-1)} - \eta_k \nabla Q_i(\omega^{(k-1)})$$

  где $\nabla Q_i(\omega^{(k-1)})$ - градиент функции потерь, вычисленный по объектам из i-го батча

3.3. Ещё один недостаток градиентного спуска

Мы можем застрять в локальном минимуме и не дойти до глобального минимума

3.4. Метод моментов (Momentum)

Будем добавлять какие-то предыдущие значения шагов, которые будут аналогом инерции из физики:

Вектор инерции (усреднение градиента по предыдущим шагам):

$$h_0 = 0$$

$$h_k = \alpha h_{k-1} + \eta \nabla Q(w^{k-1})$$

Формула метода моментов:

$$w^{(k)} = w^{(k-1)} - h_k$$

Подробнее:

$$w^{(k)} = w^{(k-1)} - \eta \nabla Q(w^{(k-1)}) - \alpha h_{k-1}$$

3.5. Преимущества метода моментов

Проще подбирать параметр α, чем делать несколько запусков обычного градиентного спуска, потому что в многомерном случае сложнее генерировать эти данные и сложнее так находить глобальный минимум

Также иногда, когда у нас данных много, GD может работать неделю, и нам непозволительно делать много запусков

3.6. Градиентный шаг

В общем случае градиентный шаг может зависеть от номера итерации, тогда мы будем писать не η, а η_k

- $\eta_k = c$ - это то что было у нас раньше - постоянный LR
- $\eta_k = \frac{1}{k}$ - здесь проблема в том, что мы можем на первом шаге стоять очень близко от глобального минимума и пролететь его
- $\eta_k = \lambda \left(\frac{s}{s_0+k}\right)^p$, λ, s_0, p - параметры

3.7. AdaGrad (Adaptive Gradient)

Сумма квадратов обновлений

$$g_{k-1,j} = (\nabla Q(\omega^{(k-1)}))^2$$

Формула метода AdaGrad:

$$G_{k,j} = G_{k-1,j} + g_{k-1,j} = G_{k-1,j} + (\nabla Q(\omega^{(k-1)}))^2$$

$$\omega_j^{(k)} = \omega_j^{(k-1)} - \frac{\eta}{\sqrt{G_{k,j} + \varepsilon}} \cdot (\nabla Q(\omega^{(k-1)}))_j$$

Этот метод использует адаптивный шаг обучения по каждой из координат веса - тем самым мы регулируем скорость сходимости метода.

Плюсы метода: происходит затухание величины шага

Минусы метода: $G_{k,j}$ монотонно возрастает, поэтому шаги укорачиваются и мы можем не успеть дойти до минимума

3.8. RMSPROP (Root Mean Square Propagation)

Метод реализует экспоненциальное затухание градиентов

Формулы метода RMSprop (усреднённый по истории квадрат градиента):

$$G_{k,j} = \alpha \cdot G_{k-1,j} + (1-\alpha) \cdot g_{k-1,j}$$

$$w_j^{(k)} = \omega_j^{k-1} - \frac{\eta}{\sqrt{G_{k,j} + \varepsilon}} \cdot (\nabla Q(\omega^{(k-1)}))_j$$

Если мы быстро сдвигались на последних шагах - то следующие будут маленькие

Если мы сдвигались медленно - то шаги будут большие

3.9. Практикум на питоне

3.9.1. Генерируем данные

```
import numpy as np
from matplotlib import pylab as plt
%pylab inline

from sklearn.datasets import make_regression


X, y = make_regression(n_samples=10000)
print(X.shape, y.shape)
```

3.9.2. Библиотечное решение

У нас 10000 объектов и 100 признаков. Для начала решим задачу аналитически "из коробки".

Решим сначала аналитически с помощью LinearRegression

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

lr = LinearRegression()

lr.fit(X, y)

print(mean_squared_error(y, lr.predict(X)))
```

Посмотрим на свободный член модели (intercept_) и веса (coef_)

```
print(lr.intercept_, lr.coef_[:5])
```

Смотрим всего 5 весов, так как в реале их там 100.
Теперь решим с помощью градиентного спуска:

```
from sklearn.linear_model import SGDRegressor

sgd = SGDRegressor(alpha=0.00000001)
sgd.fit(X, y)

print(mean_squared_error(y, sgd.predict(X)))
```

И теперь точно также посмотрим свободный член и веса

```
print(sgd.intercept_, sgd.coef_[:5])
```

4. Лекция 5. Линейная алгебра

Что такое векторное пространство? Это набор элементов (векторов), которые можно складывать и умножать на скаляр.

Что означается пространство $\mathbb{R}^2$? Что базис состоит из двух векторов. В каждом по две координаты.

Что такое линейный оператор? Линейное отображение векторного пространства в себя. Задаётся матрицей

Матрица поворота на уголь α

$$\begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$$

4.0.1. Упражнение

- Что делает оператор
$$\begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix}$$
- Что делает оператор
$$\begin{pmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$$
- Что делает оператор
$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

4.0.2. Ответы на упражнение

- Растягивает всё пространство в 3 раза (гомотетия)
- Матрица поворота на 45 градусов
- Проекция на первую координату (ось)

4.1. Выражение базиса через другой базис

Если есть два базица e и g и есть линейный оператор $A : e \to e'$, то можно найти линейный оператор A' для базиса g как

$$A' = M^{-1}AM$$

где M - базис g выраженный через базис e.

Мы можем получить одно и то же преобразование разными матрицами, потому что для л.о. могли взять разные базисы. Но у этих л.о. есть базовые характеристики/свойства, которые не меняются.

Геометрический смысл определителя матрицы л.о. показывает как изменилась площадь, натянутая на векторы, которые мы отображали. Если A - матрица л.о., x - векторы, которые образуют фигуру, $x' = Ax$ - отображённые векторы, то

$$\frac{S'}{S} = \det A$$

где S - площадь x, а S' - площадь x'

Вспомним о собственном векторе - собственный вектор не меняет направление после применения л.о. Длину - он может изменить. То есть, если v - собственный вектор, тогда $Av = \lambda v$, где λ - скаляр.

Вспомним, что если у нас многочлен нечётной степени, то хотя бы один единственный вещественный корень у него есть. Это нам говорит о том, что при матрице с нечётной длиной - всегда есть минимум один собственный вектор.

А теперь вспомним, что есть такая штука, как след - $tr\ A$, который равен сумме чисел на главной диагонали. След равен сумме собственный чисел, а значит - он тоже не меняется.

Подытожим: необходимое, но не достаточное условие для того, чтобы сказать, что какие-то два оператора задают одно и то же преобразование:

- Определитель - одинаковый
- Собственные значения - одинаковые
- След - одинаковый

4.2. Программирование

Для задач линейной алгебры есть две библиотеки - numpy.linalg и scipy.linalg. Обе эти библиотеки имеют одинаковые библиотеки, но scipy.linalg - поддерживает больше методов.

4.2.1. Нетривиальные вещи

функция np.linalg.eig - возвращает собственные значения и собственые векторы, в виде двух возвращаемых переменных. При чём если мы приняли в v собственные векторы, то первый собственный вектор будет не $v[0]$, а $v[[0][0], v[1][0], ..., v[n-1][0]]$.

4.2.2. Отрисовка векторов

```
fig, ax = plt.subplots(nrows=1, ncols=2)

origin = np.array([[0, 0], [0, 0]])
U =      np.array([[1, 0], [0, 1]])

ax[0].quiver(*origin, U[:,0], U[:,1], color='g', scale=7)
```

Данный код отрисует два вектор - (1, 0) и (0, 1)

5. Лекция 6. Метод главных компонент (PCA)

PCA - Principal Component Anaylysis.

Что мы хотим - снизить размерность у данных.

Зачем это нужно?

- Ускорить вычисления
- Сократить память
- Упростить интерпретируемость модели - проще смотреть на 10 признаков, чем на 100
- Уменьшить шанс переобучения
- Хотим лучше учиться - могут быть шумовые и неинформативные признаки

Как мы можем это реализовать?

- Когда мы убираем какие-то признаки, это называется отбором признаков
- Также мы можем выразить новые признаки через старые, при чём новых будет меньше - придумать признаки

Цель метода главных компонент: мы хотим придумать новые признаки, каким-то образом выражающиеся через старые, причём новых признаков хочется получить меньше, чем старых. И метод главных компонент **линейно** выражает новые признаки через старые.

Допустим мы хотим из 100 признаков сделать 10. Какой критерий должен быть для этих признаков? Какие из них являются хорошими

5.1. Постановка задачи выражения новых признаков

- $x_1, \ldots, x_n$ - исходные числовые признаки
- $z_1, \ldots, z_d$ - новые числовые признаки, $d \leq n$

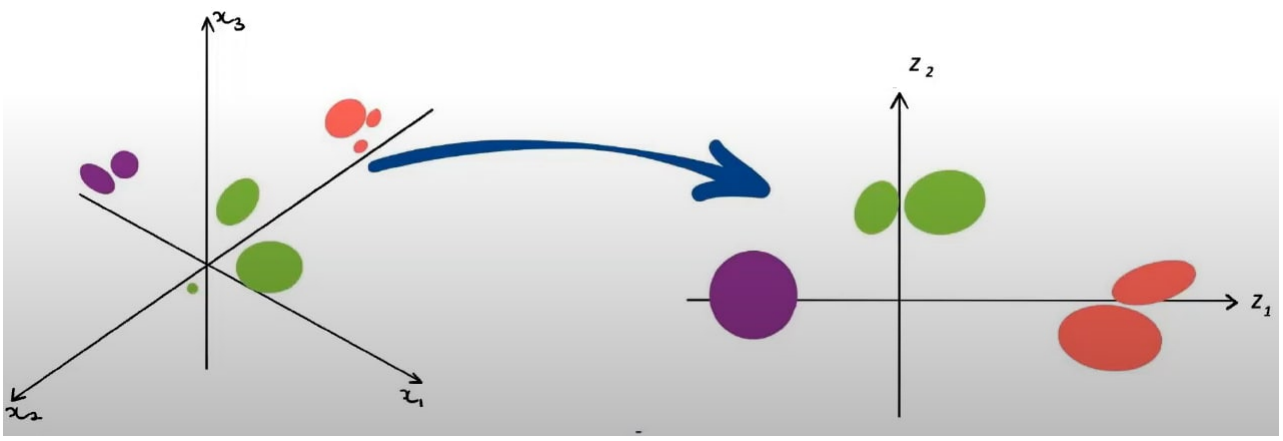
Хотим:

- Чтобы новые признаки z_j линейно выражались через исходные признаки x_i
- Чтобы при переходе к новым признакам было потеряно наименьшее кол-во информации

5.1.1. Интерпретация линейного выражения признаков

$$\begin{cases} z_1 = u_{11}x_1 + \cdots + u_{1n}x_n \\ z_2 = u_{21}x_1 + \cdots + u_{2n}x_n \\ \ldots \\ z_d = u_{d1}x_1 + \cdots + u_{dn}x_n \end{cases}$$

Геометрическая интерпретация: новые признаки z_j - это проекции исходных признаков x_i на некоторые векторы (компоненты) u. Мы проецируем пространство признаков размерности n на некоторое линейное подпространство размерности d:



5.1.2. Пояснение: Проекция

- Проекция вектора x на вектор (компоненту) u_i:

$$(x, u_i) \text{ - скалярное произведение}$$

- Проекция выборки X на компоненту u_i:

$$Xu_i$$

5.1.3. Сохранение информации

Также мы хотим, чтобы было потеряно наименьшее кол-во исходной информации

Дисперсия выборки, посчитанная в новых признаках, показывает, как много информации нам удалось сохранить после понижения размерности, поэтому дисперсия в новых признаках должна быть максимальной.

Пример: Хотим спроецировать двумерные данные X на одномерный вектор u так, чтобы дисперсия проекции Xu была максимальной:

Рис. 1: Плохое проецирование

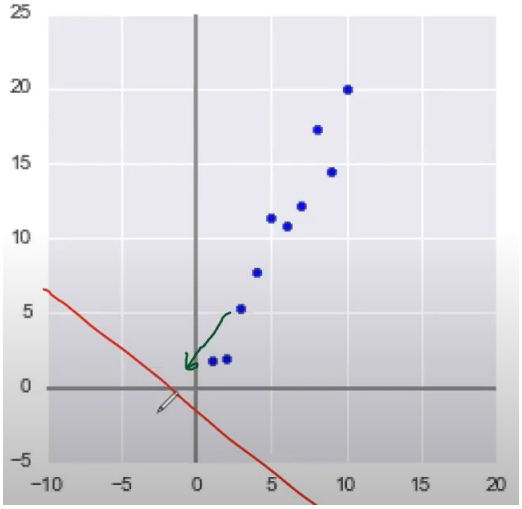
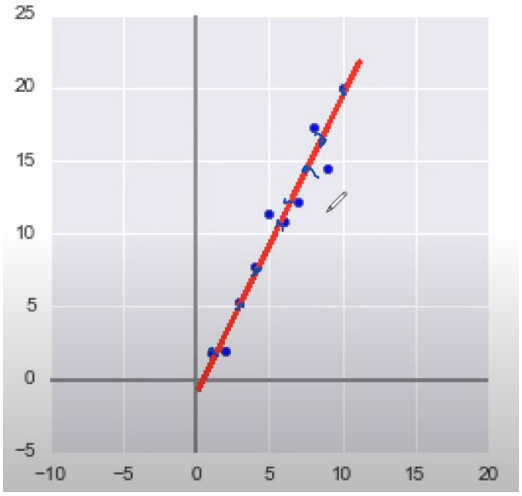


Рис. 2: Хорошее проецирование



5.2. Постановка задачи

Будем искать такие компоненты $u_1, u_2, \ldots, u_d$, что:

- Они ортогональны, то есть $\forall i, j : (u_i, u_j) = 0$
- Они нормированы, т.е. $||u_i|| = 1$

- Дисперсия проекции выборки на них максимальна: $\mathbb{D}(Xu_i) \to \max\limits_{u_i},\ i = 1, \ldots, d$
- Норма разности изначальных объектов и восстановленных - минимальна:

$$X \to Z = Xu \to u^T Z \Longleftrightarrow ||X - u^T Z||^2 \to \min_u$$

  Последние два высказывания эквивалентны
- Мы уже выяснили, что проекция выборки X на компоненту u_i: Xu_i
- Тогда проекция выборки на первые d компонент, задаваемых столбцами матрицы $U_d =< u_1, u_2, \ldots, u_d >$:

$$XU_d$$

- Тогда дисперсия проекции - это след ковариационной матрицы:

$$tr((XU_d)^T(XU_d)) = \sum_{i=1}^{d} ||Xu_i||^2 \to \max_u$$

5.2.1. Ковариация

Напомним, что матрица ковариации $\Sigma = cov(X, Y) = \mathbb{E}[(X - \mathbb{E}X)(Y - \mathbb{E}Y)^T]$

Заметим тогда, что дисперсия проекции на вектор u_i есть ни что иное, как $\mathbb{D}(Xu_i) = ||Xu_i||^2$

5.3. Алгоритм поиска новых компонент

5.3.1. Первый шаг

Будем искать первую компоненту, u_1:

$$\begin{cases} ||Xu_1||^2 \to \max\limits_{u_1} \\ ||u_1||^2 = 1 \end{cases}$$

Как такое делать?

Запишем лагранжиан

$$L(u_1, \lambda) = ||Xu_1||^2 + \lambda(||u_1||^2 - 1)$$

Найдём $\frac{\partial L}{\partial u_1} = 2X^T Xu_1 + 2\lambda u_1 = 0$

Получаем, что

$$X^T Xu_1 = -\lambda u_1$$

И это будет являться собственным вектором. Для подстановки будем использовать

$$||Xu_1||^2 = u_1^T X^T Xu_1 = \lambda u_1^T u_1 = \lambda \to \max_{u_1}$$

и найдём максимальное собственное значение, т.к. $u_1^T u_1 = ||u_1||^2 = 1$

5.4. Применение метода

Когда главные компоненты найдены, можно проецировать на них и новые данные

$$Z' = X'U_d$$

5.5. Доля объяснённой дисперсии

- Упорядочим собственные значения матрицы $X^T X$ по убыванию $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$
- Доля дисперсии, объяснённой j-й компонентой (explained variance ratio), высчитывается как

$$\delta_j = \frac{\lambda_j}{\sum\limits_{i=1}^{n} \lambda_n}$$

- Доля дисперсии, объясняемой первыми k компонентами:

$$\delta_j = \frac{\sum\limits_{i=1}^{k} \lambda_i}{\sum\limits_{i=1}^{n} \lambda_n}$$

5.6. Выбор числа главных компонент

Эффективная размерность выборки - это наименьшее целое m, при котором доля необъяснённой дисперсии

$$E_m = \frac{||ZU^T - X||^2}{||X||^2} = \frac{\lambda_{m+1} + \cdots + \lambda_n}{\sum \lambda_i} \leq \varepsilon$$

Пользуемся этим равенством как фактом.

Также есть критерий крутого склона, когда первые m компонент дают много объяснения дисперсии, а начиная с какого-то $m+1$ очень мало. Тогда мы возьмём первые m компонент

5.7. Упражнения

5.7.1. Нахождение главных компонент

Берём матрицу $X = \begin{pmatrix} 1 & 3 \\ 0 & 2 \\ 0 & 0 \\ 3 & 3 \end{pmatrix}$, вычитаем среднее в столбцах $\rightarrow \begin{pmatrix} 0 & 1 \\ -1 & 0 \\ -1 & -2 \\ 2 & 1 \end{pmatrix}$, находим матрицу ковариации

$cov(\tilde{X}) = \tilde{X^T}\tilde{X} = \begin{pmatrix} 0 & -1 & -1 & 2 \\ 1 & 0 & -2 & 1 \end{pmatrix} \times \begin{pmatrix} 0 & 1 \\ -1 & 0 \\ -1 & -2 \\ 2 & 1 \end{pmatrix} = \begin{pmatrix} 6 & 4 \\ 4 & 6 \end{pmatrix}$

$\left| \begin{pmatrix} 6-\lambda & 4 \\ 4 & 6-\lambda \end{pmatrix} \right| = 0 \Longleftrightarrow \lambda_1 = 10, \lambda_2 = 2$

Для значения $_1 = 10$ находим собственные значения $\begin{pmatrix} -4 & 4 \\ 4 & -4 \end{pmatrix} \times \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix} \Longleftrightarrow v_1 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$

Для значения $\lambda_2 = 2$ - аналогично

5.7.2. То же самое в питоне

5.7.2.1. Генерация данных .

```
x = np.arange(1, 11)
y = 2 * x + np.random.randn(10) * 2
X = np.vstack((x, y))

plt.scatter(X[0], X[1])
```

5.7.2.2. Реализация метода главных компонент .

```
Xcentered = X - X.mean(axis=1, keepdims=True)
m = np.mean(X, axis=1)

print("Mean vector:", m)
print(Xcentered)

plt.scatter(Xcentered[0], Xcentered[1])
```

5.7.2.3. Вычисляем матрицу ковариаций .

```
covmat = np.cov(Xcentered)
print(covmat, '\n')
print('Variance of X: ', np.cov(Xcenetered)[0, 0])
print('Variance of Y: ', np.cov(Xcenetered)[1, 1])
print('Covariance of X and Y: ', np.cov(Xcenetered)[0, 1])
```

5.7.2.4. Находим собственные векторы и проецируем все точки .

```
eignums, vecs = np.linalg.eig(covmat)
v = vecs[:, np.argmax(eignums)]
Xnew = np.dot(v, Xcentered)
print(Xnew)
```

5.7.2.5. Восставновление данных .

```
n = 3
Xrestored = np.dot(Xnew[n], v) + m
print('Restored: ', Xrestored)
print('Original: ', X[:, n])
```

5.7.2.6. Доля объяснённой дисперсии .

```
100 * max(eignums) / sum(eignums)
```

5.7.2.7. Что выдаёт PCA из sklearn .

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
X_pca = pca.fit_transform(Xcentered.T)
```

5.8. Сингулярное разложение (SVD)

SVD - Singular Value Decomposition

Теорема. Матрицу $A \in \mathbb{R}^{m \times n}$ можно представить в виде

$$A = U\Sigma V^T$$

где

- $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$ - ортогональные матрицы
- $\Sigma \in \mathbb{R}^{m \times n}$ - диагональная матрица с ненулевыми элементами $\sigma_i = \sqrt{\lambda_i}$, где λ_i - собственные значения матрицы $A^T A$

При этом

- Столбцы матрицы U являются собственными векторами матрицы AA^T
- Столбцы матрицы V являются собственными векторами матрицы $A^T A$

По аналогии с PCA можно показать, что в столбцах V и стоят главные компоненты - столбцы V являются её собственными векторами, и мы на них проецируем, и по сути, её столбцы и являются компонентами.

5.9. Что лучше: PCA или SVD?

- Существуют вычислительные трудности с нахождением собственных значений, в этом недостаток PCA
- Существует итерационный алгоритм для нахождения SVD (без нахождения собственных значений)

Поэтому вычислительно эффективнее использовать SVD при прочих равных.

5.10. Упражнение

- Разложить матрицу $X \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$ в SVD

5.10.1. Находим ответы

Найдём собственные значением матрицы $X^TX = \begin{pmatrix}1 & 2\\ 2 & 1\end{pmatrix} \times \begin{pmatrix}1 & 2\\ 2 & 1\end{pmatrix} = \begin{pmatrix}5 & 4\\ 4 & 5\end{pmatrix}$

Найдём собственные значения $\left|\begin{pmatrix}5-\lambda & 4\\ 4 & 5-\lambda\end{pmatrix}\right| = 0 \Longleftrightarrow \lambda_1 = 9, \lambda_2 = 1$

Тогда получаем, что $\Sigma = \begin{pmatrix}\sqrt{9} & 0\\ 0 & \sqrt{1}\end{pmatrix} = \begin{pmatrix}3 & 0\\ 0 & 1\end{pmatrix}$

Собственные векторы $v_1 = \begin{pmatrix}\frac{-1}{\sqrt{2}}\\ \frac{-1}{\sqrt{2}}\end{pmatrix}, v_2 = \begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{-1}{\sqrt{2}}\end{pmatrix} \to V = \begin{pmatrix}\frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\\ \frac{-1}{\sqrt{2}} & \frac{-1}{\sqrt{2}}\end{pmatrix}$

Можем посчитать U как $U = XV\Sigma^{-1} = \begin{pmatrix}\frac{-1}{\sqrt{2}} & \frac{-1}{\sqrt{2}}\\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\end{pmatrix}$

5.11. Связь SVD и PCA

- Столбцы матрицы V - это собственные векторы матрицы X^TX, т.е. векторы $v_1, \ldots, v_n$ - главные компоненты
- Столбцы матрицы $U\Sigma$ - это новые признаки $z = Xv$ ($X = U\Sigma V^T \Longleftrightarrow U\Sigma = XV$)
- Сингулярное числа матрицы Σ - это корни из собственных чисел матрицы X^TX

Для снижения размерности - берём первые k столбцов матрицы U и верхний $k \times k$ квадрат матрицы Σ, тогда матрица $U_K\Sigma_k$ содержит k новых признаков, соответствующих первым k компонентам.

5.12. Сингулярное разложение в питоне

```
import numpy as np

A = np.array([[3, 4, 3], [1, 2, 3], [4, 2, 1]])

U, D, VT = np.linalg.svd(A)

A_remake = (U @ np.diag(D) @ VT)
print(A_remake)
```

Часть II
Модуль 2

Преподавать будет Филипп Ульянкин

6. Лекция 1. Множества

Что такое **множества**? Это перечисление каких-то элементов. Обычно мы множество записываем в фигурные скобки: $A = \{5, 42, 7\}$ - множество, которое состоит из числа 5, 42 и 7.

Также мы можем задать из каких элементов состоит наше множество, если их долго перечислять: $B = \{2^{k-1} | k \in \mathbb{N}\} = \{2^0, 2^1, 2^2, 2^3, \dots\}$. То есть мы указываем свойство.

Третий способ задать множество - написать алгоритм получения элементов: $1 \in C$ или $a \in C \to 2 \cdot a \in C$.

6.1. Особые множества

- $\mathbb{N}$ - множества натуральных чисел
- $\mathbb{Z}$ - целые числа
- $\mathbb{Q}$ - рациональные числа
- $\mathbb{R}$ - вещественные числа
- $\mathbb{C}$ - комплексные числа
- $\varnothing$ - пустое множество
- $\mathbb{U}$ - универсальное множество - множество из всех существующих элементов
- 2^A - булеан - множество всех подмножеств

6.2. Упражнение

$A = \{1, 4, 5\}$, $B = \{4, 3\}$, $C = \{2\}$
$A \cap B = \{4\}$
$A \cup B = \{1, 3, 4, 5\}$
$A \cap C = \varnothing$
$A \backslash B = \{1, 5\}$
$A \triangle B = (A \backslash B) \cup (B \backslash A) = (A \cup B) \backslash (A \cap B) \{1, 3, 5\}$.
$A \times B = \{(1, 4), (1, 3), (4, 4), (4, 3), (5, 4), (5, 3)\}$
$2^A = \{\{1, 4, 5\}, \varnothing, \{1\}, \{4\}, \{5\}, \{1, 4\}, \{1, 5\}, \{4, 5\}\}$
$|A| = 3$ - мощность множества
$|2^A| = 2^{|A|} = 2^3 = 8$
Существует очень много свойств, для примера:

- $A \cup B \cap C = (A \cap C) \cup (B \cap C)$
- $(A \cup B) \times C = (A \times C) \cup (B \times C)$
- $\overline{\overline{A} \cup \overline{B}} = \overline{A} \cap \overline{B}$ - закон де Моргана

6.3. Упражнение

Отель имеет ∞ номеров. Все номера заняты, мы не можем заселиться.

Давайте переселим всех людей на номер $+1$ и тогда освобождается первый номер и мы можем заселиться.

А что если приехало ∞ новых людей? Сможем ли мы их заселить?

Давайте каждого человека селить не в номер $+1$, а в номер $\cdot 2$. Тогда освобождаются все нечётные номера и мы их можем заселить.

6.3.1. Определение равномощности

Множества A и B называются равномощными, если между их элементами существует взаимно-однозначное соответсвтие (биекция).

6.4. Упражнение

Сравнить множества

- $|\mathbb{Z}|$ и $|\mathbb{N}|$
- $|\mathbb{N} \times \mathbb{N}|$ и $|\mathbb{N}|$
- $|\mathbb{Q}|$ и $|\mathbb{N}|$

6.5. Ответы на упражнения

- Очень легко. Просто нумеруем целые числа в порядке 0, -1, 1, -2, 2 итд, и получае биекцию
- Аналогично с пунктом нижу
- На самом деле их мощности совпадают, потому что мы можем построить биекцию. А эту биекцию мы можем сделать в виде алгоритма. Мы можем нарисовать координатную сетку, и обходить её "улиткой начиная из (0,0). Если мы в очередной шаг наступаем на рациональное число, то увеличим каунтер встреченных чисел и говорим какое по номеру это число. Рано или поздно мы встретим любое из рациональных чисел. При этом встречать рациональные числа мы не перестанем, потому что любое число вида $\frac{1}{n}$ или $\frac{n-1}{n}$ является рациональным. Доказано

Определение: говорят, что множество A счётное, и оно равномощно множеству $|\mathbb{N}|$.

6.6. Все ли бесконечные множества равномощны?

Ответ: нет. Доказательство:

Пусть у нас есть множество бесконекчных бинарных чисел. Мы их начинаем нумеровать (строить биекцию из $\mathbb{N}$). Предположим такая биекция есть. Но давайте выпишем каждое число в строку. Поменяем бит на главной диагонали. Тогда мы получим на диагонали новое число, которое не равно ни одному из уже имеющихся, ведь с i-м число оно различается в i-м бите.

6.7. Континуальность

Определение: Множество B для которого мощность совпадает с $\mathbb{R}$ называется континуальным.

6.8. Теорема о удалении бесконечного множества

Если из бесконечного множества M вырезать счётное подмножество A и осталось бесконечное кол-во элементов, то $|M \backslash A| = |M|$

6.9. Упражнение

За S сейчас и в дальнейшем обозначаем континуальное множество. Множество бесконечных бинарных последовательностей.

Является ли $|[0;1]| = |S|$?

$\forall X \in [0;1]$ можно записать в виде бесконечной последовательности из 0 и 1. Можно бин поиском делать эту последовательность: если лежит $<=$ m, то пишем 0, если $>$m, то 1.

6.10. Упражнение

Последовательности похожи, если отличаются на конечное число элементов.

Есть элемент из бесконечного числа нулей. Какова мощность множества элементов $|A|$ всех похожих на этот элемент?

Как минимум счётно, так как мы можем нумеровать последовательность слева в виде двоичной системы счисления, а справа вставить бесконечное кол-во нулей.

6.11. Упражнение

Отношение похожести разбиывает все последовательности на классы эквивалентности. Какова мощность таких множеств? Допустим элемент множества A_i

У нас есть всего два варианта: либо это континуальное множество, либо счётное.

У S мощность континуум. S мы можем получить, как $S = \bigcup_{i=1}^{\infty} A_i$.

Если мы объединим бесконечное множество счётных множеств, то мы получаем континуальное. Но мы знаем, что при объединении счётных множеств ($\mathbb{N} \times \mathbb{N}$) мы получаем счётное. Тогда мы приходим к противоречию и получаем, что мощность A_i - континуально.

6.12. Упражнение

Дракон захватил гномов и поставил их в шеренгу. На каждого гнома надет колпак. Каждый гном видит все колпаки впереди стоящих перед ним гномов, но свой и гномов позади колпаки - не видит.

Дракон начинает задавать гномам вопрос: "какой колпак на тебе надет?". Если гном отвечает правильно - дракон его отпускает. Если гном отвечает неправильно - дракон его съедает.

Вопрос: существует ли здесь какая-то стратегия такая, что погибнет конечное число гномов?

7. Лекция 2. Сигма-алгебра

7.1. Пространство элементарных исходов

Все варианты, которые произойти или не произойти для какого-то случайного действия - называются пространством элементарных исходов и обозначаются Ω. Для события подкидывания игральной кости, у нас элементарные исходы могут быть - чем выпала кость, и тогда $\Omega = \{1, 2, 3, 4, 5, 6\}$. Сами исходы обычно изображаются ω_i.

Все события, с которыми мы обычно имеем дело - это какие-то множества $A \subset \Omega$, которые лежат внутри пространства элементарных исходов

7.2. Упражнение (классическая вероятность)

$\Omega = \{1, 2, 3, 4, 5, 6\}$ A - выпало чётное число на кубике, тогда $A = \{2, 4, 6\} \subset \Omega$.

Будем считать, что кубик правильный, и $\mathbb{P}(w_i) = \frac{1}{6}$. Но часто события не являются равновероятными.

Но в нашем случае $\mathbb{P}(A) = \frac{|A|}{|\Omega|} = \frac{3}{6} = \frac{1}{2}$

7.3. Важная мысль

Нам надо уметь считать мощность множеств там, где события равновероятны $\Rightarrow$ нам нужно обладать знаниями комбинаторики.

Может быть также, что вероятность определяется не для конечных множеств, а для бесконечных множеств, например для площадей

7.4. Упражнение (геометрическая вероятность)

Допустим есть промежуток времени в 1 час с 9 до 10 утра. В этот час Саша и Катя хотят встретиться. Но они приходят в случайное время, ждут 15 минут и уходят. Какова вероятность, что Саша и Катя встретятся?

Как устроено пространство элементарных исходов и как можно параметризовать наше событие?

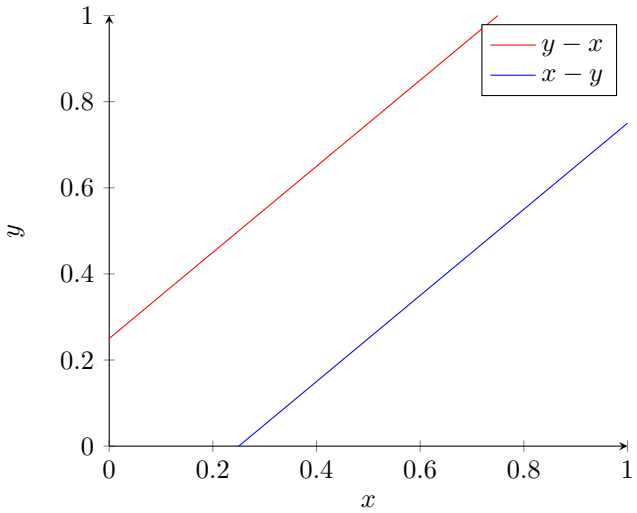
Можно сказать, что час - это отрезок $[0; 1]$, а ждут они 0.25 ($\frac{1}{4}$ часа - 15 минут).

Тогда $\Omega = [0; 1] \times [0; 1]$ - времена, когда пришли девочки. x - когда пришла Саша, y - когда пришла Катя. Тогда мы можем нарисовать квадрат. Тогда для нас множество, которое удовлетворяет событию - $A = \{(x, y) \in \Omega || x - y| \leq 0.25\}$

Что означает $|x - y| \leq \frac{1}{4}$?

$$A = \begin{cases} x > y & x - y \leq \frac{1}{4} \\ x < y & y - x \leq \frac{1}{4} \end{cases}$$

Давайте нарисуем это:



Площадь между этими прямыми нам и подходит для ответа.

Но конретно в этом случае нам проще найти площади, которые отвечают за то, что они не встретятся. $\mathbb{P}(\text{Встреча есть}) = 1 - \mathbb{P}(\text{Встречи нет}) = 1 - \left(\frac{3}{4}\right)^2 = \frac{7}{16}$

Следующий вопрос: можно ли $\forall A \in \Omega$ считать событием?

Ответ: иногда да, иногда нет, зависит от сигма-алгебры $\mathbb{F}$

А как правильно определять вероятность $\mathbb{P}$

7.5. Сигма-алгебра

Вероятностная тройка - $(\Omega, \mathbb{F}, \mathbb{P})$.

σ-алгебры: список множеств, которым мы считаем событием

Удобное определение: набор множеств $\mathbb{F}$, которые являются подмножествами множества Ω, называется σ-алгеброй, если:

- $\varnothing \in \mathbb{F}$, $\Omega \in \mathbb{F}$
- Если событие $A_1, A_2, \ldots, A_n \in \mathbb{F}$, тогда любая их комбинация $(\cup, \cap, \setminus, \triangle, \overline{A})$ тоже лежит в $\mathbb{F}$

Формальное определение:

- $\Omega \in \mathbb{F}$
- $A \in \mathbb{F} \to A^c = \Omega \setminus A \in \mathbb{F}$
- $A_1, A_2, \ldots, A_n \in \mathbb{F} \to \bigcup\limits_{i=1}^{\infty} A_i \in \mathbb{F}$

и это является минимальным набором требований для σ-алгебры

7.6. Упражнение

$A \cap B = \overline{\overline{A} \cup \overline{B}}$

7.7. Упражнение

Все мы помним $\Sigma = \{1, 2, 3, 4, 5, 6\}$ для бросания кубика. Но есть Вася, который знает у чисел только остаток от деления на 3. Тогда он не различает между собой события выпало 1/4, 2/5, 3/6. Для васи набор событий $\mathbb{F}$ - те события, которые он может понять. Если более формально, то $\mathbb{F} = \{\{1, 4\}, \{2, 5\}, \{3, 6\}\varnothing, \Omega, \{1, 4, 2, 5\}, \{1, 4,$
Это все события, которые воспринимает Вася. Где-то объединение простеших событий, например $\{1, 4, 2, 5\}$
- он, допустим не разглядел, что выпало, но знает, что остаток от деления у этого числа - не ноль.

- Самая богатая σ-алгебра: 2^{Ω}
- Самая бедная σ-алгебра: $\{\varnothing, \Omega\}$

7.8. Зачем нужна σ-алгебра

Зачем?

- моделирование наделённостью информации
- технические цели в теоремах

Всегда существует самый маленький набор множеств, который порождает σ-алгебру, и поэтому очень часто записывают $\mathbb{F} = \sigma(\text{набор событий})$

7.9. Определение вероятности

Вероятность $\mathbb{P}$ просто задаётся функцией $\mathbb{P} : \mathbb{F} \to [0; 1]$

И перед тем, как идти дальше, скажем, что $A \sqcup B$ обозначает объединение, которое ещё говорит о том, что $A \cup B = \varnothing$, то есть A и B точно не пересекаются

Аксиомы Колмогорова

- $\mathbb{P}(\Omega) = 1$
- $\mathbb{P}(A) = 1 - \mathbb{P}(\overline{A})$
- $\mathbb{P}(A \sqcup B) = \mathbb{P}(A) + \mathbb{P}(B)$
- $\mathbb{P}(\bigsqcup\limits_{i=1}^{\infty} A_i) = \sum\limits_{i=1}^{\infty} \mathbb{P}(A_i)$

7.10. Упражнение

Подкидываем монетку: $\Sigma = \{0, 1\}$ - выпал орёл/решка.

$\mathbb{F} = \sigma(0, 1) = 2^{\Sigma}$

$\mathbb{P} : \mathbb{F} \to [0; 1]$

Если монетка правильная, то $\mathbb{P}(0) = \mathbb{P}(1) = \frac{1}{2}$. Если монетка смещённая, то, например, может быть $\mathbb{P}(0) = \frac{1}{4}$ и $\mathbb{P} = \frac{3}{4}$

7.11. Подытог

Тройка $(\Omega, \mathbb{F}, \mathbb{P})$ задаёт модель.

7.12. Свойства вероятности

Правило сложения: $\mathbb{P} = \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \cap B)$

7.12.1. Упражнение

Есть кубик. Событие A - выпало чётное число, событие B - выпало число, которое делится на 3. Тогда $\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \cap B) = \mathbb{P}(2 \cup 4 \cup 6) + \mathbb{P}(3 \cup 6) - \mathbb{P}(6) = \frac{3}{6} - \frac{2}{6} - \frac{1}{6} = \frac{4}{6} = \frac{2}{3}$

Вытекает, что $\mathbb{P}(A \cup B) \leq \mathbb{P}(A) + \mathbb{P}(B)$

7.12.2. Упражнение

Если три множества, то

$\mathbb{P}(A \cup B \cup C) = \mathbb{P}(A) + \mathbb{P}(B) + \mathbb{P}(C) - \mathbb{P}(A \cap B) - \mathbb{P}(A \cap C) - \mathbb{P}(B \cap C) + \mathbb{P}(A \cap B \cap C)$

7.13. Условная вероятность

Допустим мы знаем, что B точно произошло.

Чему тогда равно $\mathbb{P}(A|B)$?

Как пример: A - кубик выпал на 2, B - выпало чётное

Такая вероятность $\mathbb{P}(A|B) = \frac{|A \cap B|}{|B|} = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}$

И в другом виде это запиывается ещё как $\mathbb{P}(A \cap B) = \mathbb{P}(A|B) \cdot \mathbb{P}(B) = \mathbb{P}(B|A) \cdot \mathbb{P}(A)$

7.14. Формула Байеса

$$\mathbb{P}(B|A) = \frac{\mathbb{P}(A|B) \cdot \mathbb{P}(B)}{\mathbb{P}(A)}$$

7.15. Формула полной вероятности

Допустим мы "нарезали"$\Omega = \bigcup\limits_{i=1}^{k} H_k$, при этом $\forall i, j : H_i \cap H_j = \varnothing$, то есть $\Omega = \bigsqcup\limits_{i=1}^{k} H_i$, тогда

$$\sum_{i=1}^{k} \mathbb{P}(H_i) = 1$$

При этом $\mathbb{P}(A) = \sum\limits_{i=1}^{k} \mathbb{P}(A \cap H_i) = \sum\limits_{i=1}^{k} \mathbb{P}(A|H_i) \cdot \mathbb{P}(H_i)$

7.15.1. Упражнение

Есть школа, в которой три класса - А, Б и В. В них одинаково человек, но

- В классе А 30% учеников любят географию
- В классе Б 40% учеников любят географию
- В классе В 70% учеников любят географию

Мы взяли случайного ученики и спросили, любит ли он географию. Какова вероятность, что этот ученик любит географию?

$\mathbb{P}(\text{Петя}\heartsuit\text{географию}) = \mathbb{P}(\text{А}) \cdot \mathbb{P}(\heartsuit|\text{А}) + \mathbb{P}(\text{Б}) \cdot \mathbb{P}(\heartsuit|\text{Б}) + \mathbb{P}(\text{В}) \cdot \mathbb{P}(\heartsuit|\text{В}) = \frac{1}{3} \cdot 0.3 + \frac{1}{3} \cdot 0.4 + \frac{1}{3} \cdot 0.7 = \frac{14}{30} = \frac{7}{15}$.

Давайте посчитаем, какова вероятность, что Петя из класса А, Б или В если он любит географию:

- Из класса А: $\mathbb{P}(\text{А}|\heartsuit) = \frac{\mathbb{P}(\heartsuit|\text{А}) \cdot \mathbb{P}(A)}{\mathbb{P}(\heartsuit)} = \frac{0.3 \cdot \frac{1}{3}}{\frac{7}{15}} = \frac{1}{10} \cdot \frac{15}{7} = \frac{3}{14}$
- Из класса Б: $\mathbb{P}(\text{Б}|\heartsuit) = \frac{\mathbb{P}(\heartsuit|\text{Б}) \cdot \mathbb{P}(\text{Б})}{\mathbb{P}(\heartsuit)} = \frac{0.4 \cdot \frac{1}{3}}{\frac{7}{15}} = \frac{2}{15} \cdot \frac{15}{7} = \frac{2}{7}$
- Из класса В: $\mathbb{P}(\text{В}|\heartsuit) = \frac{\mathbb{P}(\heartsuit|\text{В}) \cdot \mathbb{P}(\text{В})}{\mathbb{P}(\heartsuit)} = \frac{0.7 \cdot \frac{1}{3}}{\frac{7}{15}} = \frac{7}{30} \cdot \frac{15}{7} = \frac{1}{2}$

7.16. Априорное и апостериорное распределение

Априорное распределение - это распределение вероятностей, которое мы получаем до эксперимента

Апостериорное распределение - это распределение вероятностей, которое мы получили после эксперимента

В нашем случае экспериментом было то, что мы спрашивали у Пети, любит ли он географию.

7.17. Байесовская вероятность в виде графа

Затехать граф

7.17.1. Аналитическая задача на формулу Байеса

Оценить долю спама в комментариях. Нужно было следить за тем, как после выкатки алгоритма изменяется процент спама в комментах

Как измерялся спам?

- Толокеры - ошибаются много, но размечают тоже много
- Штатские модераторы - не ошибаются никогда

Мы можем взять выборку, чтобы её оценили толокеры, а потом после толокеров передать её модерам, чтобы они оценили качество толокеров.

Допустим доля спама в комментариях q, а доля спама, которую дали толокеры - ν

Написать Сандомирской, как нормально затехать это

Вероятность, когда модеры отметили как спам, а толокеры как не спам обозначим за P_1, а наоборот за P_2.

Тогда мы знаем, что $\nu = (1-q)p_2 + q(1-p_1)$ (там когда граф затехаю будет видно).

И получается, что $\mathbb{P}(spam|\text{Толока отметила } spam) = \frac{q(1-p_1)}{\nu} = \frac{q(1-p_1)}{q(1-p_1)+p_2(1-q)}$

8. Факультативная часть лекции 2

8.1. Случайные величины и борелевская σ-алгебра

Случайная величина - функция, которая отображает пространство элементарных исходов Ω на множество $\mathbb{R}$ $X : \Omega \to \mathbb{R}$, но на ней нет требований, какие есть в обычных вероятностях.

8.1.1. Пример

При бросании орла и решки у нас уже не обязательны исходы $\{\text{орёл}, \text{решка}\}$, мы можем записать это как $\{1, 0\}, \{-1, 1\}, \{42, 15\}$.

После подбрасывания монетки какая мысль возникает? Если я знаю занчение случайной величины X, то значит я понимаю какие события произошли.

8.2. Подготовка к Борелевской σ-алгебре

Когда берут σ-алгебру, то часто говорят о том, что эта σ-алгебра порождается случайной величиной X: $\sigma(X) = \sigma(\{\{X \leq t\} t \in \mathbb{R}\})$ - σ-алгебра, порождённая случайной величиной X

8.2.1. Упражнение

Почему мы можем все события так написать?

$\{X^2 > 7\} \in \sigma(X)$

Мы можем переписать это в виде $X \in (-\infty; -\sqrt{7}) \cup (\sqrt{7}; +\infty)$

Левое событие выглядит как $X < \sqrt{7}$, правое как $X > \sqrt{7}$. Обозначим их как B и A.

Тогда $\overline{A} = X \leq \sqrt{7} \in \sigma(X)$ по определению $\Rightarrow A \in \sigma(X)$

$B = \bigcup\limits_{i=1}^{\infty} \frac{\{x \leq \sqrt{7} - \frac{1}{i}\}}{B_i}$

Следует, что $\forall B_i \in \sigma(X) \Rightarrow B \in \sigma(X)$

Почему в определении именно такие множества?

Ответ: Потому что такими событиями можно описать все события из борелевское σ-алгебры.

8.3. Борелевская σ-алгебра

Борелевская σ-алгебра - σ-алгебра, порождённая открытыми подмножествами множества Σ

8.3.1. Открытое множество

Открытое множество - все точки которого - внутренние.

8.3.2. Внутренние точки

Рассмотрим интервал $(2; 4)$ и точку 7 - $(2; 4) \cup \{7\}$. Это не открытое множество.

виды точек:

- Внутренние точка A - δ окрестность точки лежит в множестве: $\exists U_\varepsilon(a) \subset A$
- Граничная точка A - $\forall U_\varepsilon(a)$ верно, что $U_\varepsilon \cap A \neq \varnothing$ и $U_\varepsilon(a) \cap \overline{A} \neq \varnothing$
- Изолированная точка A - $\exists U_\varepsilon(a) : U_\varepsilon \cap A = \{a\}$

8.3.3. Замкнутое множество

Замкнутое множество - дополнение к открытому.

8.4. Бореловское множество

Борелевское множество - любое множество из Борелевской σ-алгебры

8.4.1. Упражнение

Если у нас есть $\mathbb{B}(\mathbb{R})$, входит ли $(2; 100) \in \mathbb{B}(\mathbb{R})$? Да, потому что оно открытое.

Будет ли $[2; 100] \in \mathbb{B}(\mathbb{R})$? Да, потому что $[2; 100] = \mathbb{R} \backslash ((-\infty; 2) \cup (100; +\infty))$.

Будет ли $\{7\} \in \mathbb{B}(\mathbb{R})$? Да, потому что $\{7\} \in \mathbb{R} \backslash ((-\infty; 7) \cup (7; +\infty))$

Будет ли $[2; 100) \in \mathbb{B}(\mathbb{R})$? Да, потому что их можно разбить на отрезки и интервалы и свести к предыдущим случаям.

Все привычное нам множества - Боерелевские.

Не Борелевские множества существуют, но нам не привычные.

8.5. Измеримость

Говорят, что случайная величина X измерима относительно σ-алгебры $\mathbb{F}$, если $\sigma(X) \subseteq \mathbb{F}$.

На самом деле, при задании Ω, $(\Omega, \mathbb{F})$ - измеримое пространство.

Обычно любое множество из $\mathbb{F}$ - измеримо.

Ещё говорят, что случайная величина Y измерима относительно случайной величины X - это значит, что σ-алгебры Y является подмножеством σ-алгебры X: $\sigma(X) \subseteq \sigma(X)$

Теорема: Y измеримо относительно $X \iff Y = f(x)$