

Дополнительная лекция



Вектора и операции над ними. Операции с матрицами. Производная и градиент. Градиентный спуск. Дифференциальная эволюция. Матричные разложения. Понижение размерности.

Даниил Корбут

Специалист по Анализу Данных

Векторы

Вектор — упорядоченный конечный список чисел.
Вектора обычно записываются как вертикальный список,
например:

$$\begin{bmatrix} -1.1 \\ 0.0 \\ 3.6 \\ -7.2 \end{bmatrix} \quad \begin{pmatrix} -1.1 \\ 0.0 \\ 3.6 \\ -7.2 \end{pmatrix}$$

Вектор может быть записан также в следующем виде:

$$(-1.1, 0.0, 3.6, -7.2)$$

Скалярное произведение векторов

Скалярное произведение векторов (dot product по англ.) - это скаляр (число), полученное в результате перемножения длин векторов на косинус угла между ними.

$$\vec{a} \cdot \vec{b} = |\vec{a}| \cdot |\vec{b}| \cdot \cos(\alpha)$$

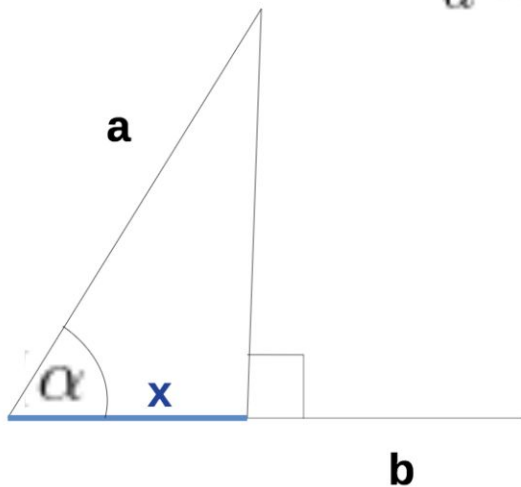
Если известны координаты векторов, то скалярное произведение можно посчитать по формуле:

$$\vec{a} \cdot \vec{b} = x_a \cdot x_b + y_a \cdot y_b$$

где $\vec{a}(x_a; y_a)$ и $\vec{b}(x_b; y_b)$ вектора в двумерном пространстве

Проекция одного вектора на другой

Длина вектора x , полученного в результате проекции вектора a на вектор b , равна делению скалярного произведения вектора \mathbf{a} на вектор \mathbf{b} на длину b .



$$\vec{a} \cdot \vec{b} = |\vec{a}| \cdot |\vec{b}| \cdot \cos(\alpha)$$

$$\cos(\alpha) = \frac{|x|}{|a|}$$

$$|x| = \cos(\alpha) \cdot |a|$$

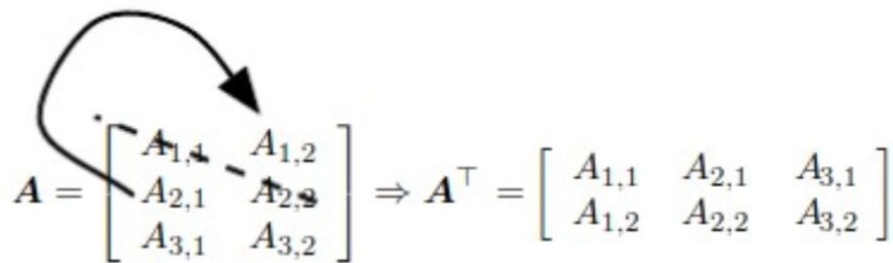
$$\cos(\alpha) \cdot |a| = \frac{\vec{a} \cdot \vec{b}}{|b|}$$

$$|x| = \frac{\vec{a} \cdot \vec{b}}{|b|}$$

Транспонирование матрицы

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

Транспонирование матрицы
— это замена строк на
столбцы.


$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix} \Rightarrow A^T = \begin{bmatrix} A_{1,1} & A_{2,1} & A_{3,1} \\ A_{1,2} & A_{2,2} & A_{3,2} \end{bmatrix}$$

Транспонирование матрицы можно рассматривать как отображение матрицы относительно главной диагонали.

Обратная матрица

Обратная матрица к данной — это матрица при перемножении которой с текущей матрицей получается единичная матрица.

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$$

Например:

$$B = \begin{bmatrix} 2 & 5 & 7 \\ 6 & 3 & 4 \\ 5 & -2 & -3 \end{bmatrix} \quad B^{-1} = \begin{bmatrix} 1 & -1 & 1 \\ -38 & 41 & -34 \\ 27 & -29 & 24 \end{bmatrix}$$
$$B \cdot B^{-1} = \begin{bmatrix} 2 & 5 & 7 \\ 6 & 3 & 4 \\ 5 & -2 & -3 \end{bmatrix} \cdot \begin{bmatrix} 1 & -1 & 1 \\ -38 & 41 & -34 \\ 27 & -29 & 24 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \mathbf{I}$$

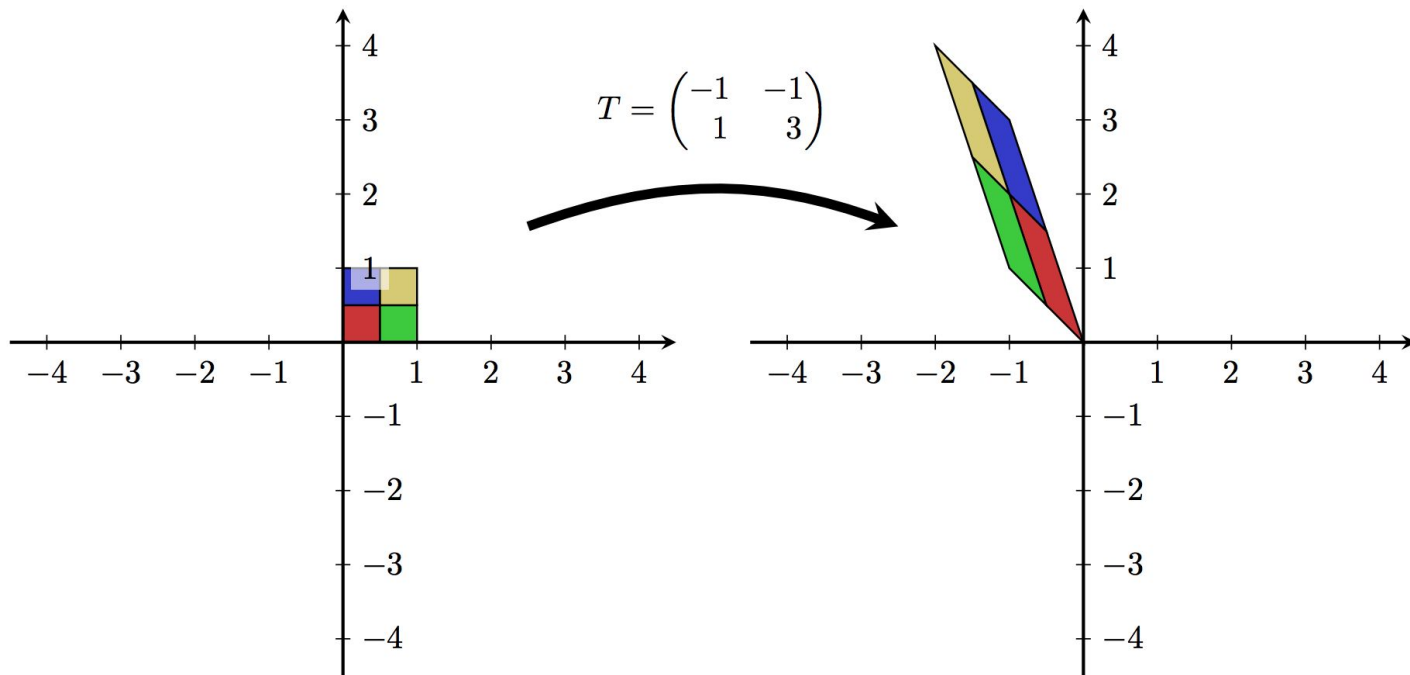
Перемножение матриц

Даны 2 матрицы: A и B. Умножение матрицы A на B можно выполнить, если количество столбцов матрицы A равно количеству строк матрицы B.

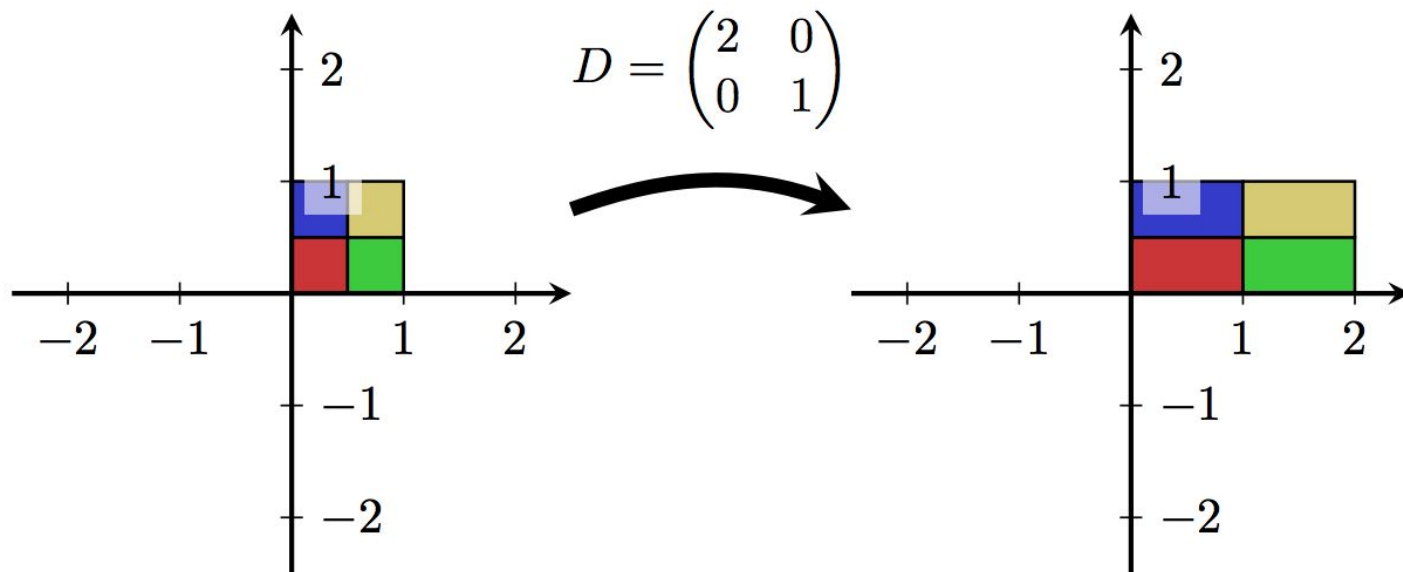
$$A \cdot B = \begin{pmatrix} 1 & 2 & 2 \\ 3 & 1 & 1 \end{pmatrix} \begin{pmatrix} 4 & 2 \\ 3 & 1 \\ 1 & 5 \end{pmatrix} = \begin{pmatrix} 1 \cdot 4 + 2 \cdot 3 + 2 \cdot 1 & 1 \cdot 2 + 2 \cdot 1 + 2 \cdot 5 \\ 3 \cdot 4 + 1 \cdot 3 + 1 \cdot 1 & 3 \cdot 2 + 1 \cdot 1 + 1 \cdot 5 \end{pmatrix} = \begin{pmatrix} 12 & 14 \\ 16 & 12 \end{pmatrix}$$

$$B \cdot A = \begin{pmatrix} 4 & 2 \\ 3 & 1 \\ 1 & 5 \end{pmatrix} \begin{pmatrix} 1 & 2 & 2 \\ 3 & 1 & 1 \end{pmatrix} =$$
$$\begin{pmatrix} 4 \cdot 1 + 2 \cdot 3 & 4 \cdot 2 + 2 \cdot 1 & 4 \cdot 2 + 2 \cdot 1 \\ 3 \cdot 1 + 1 \cdot 3 & 3 \cdot 2 + 1 \cdot 1 & 3 \cdot 2 + 1 \cdot 1 \\ 1 \cdot 1 + 5 \cdot 3 & 1 \cdot 2 + 5 \cdot 1 & 1 \cdot 2 + 5 \cdot 1 \end{pmatrix} = \begin{pmatrix} 10 & 10 & 10 \\ 6 & 7 & 7 \\ 16 & 7 & 7 \end{pmatrix}$$

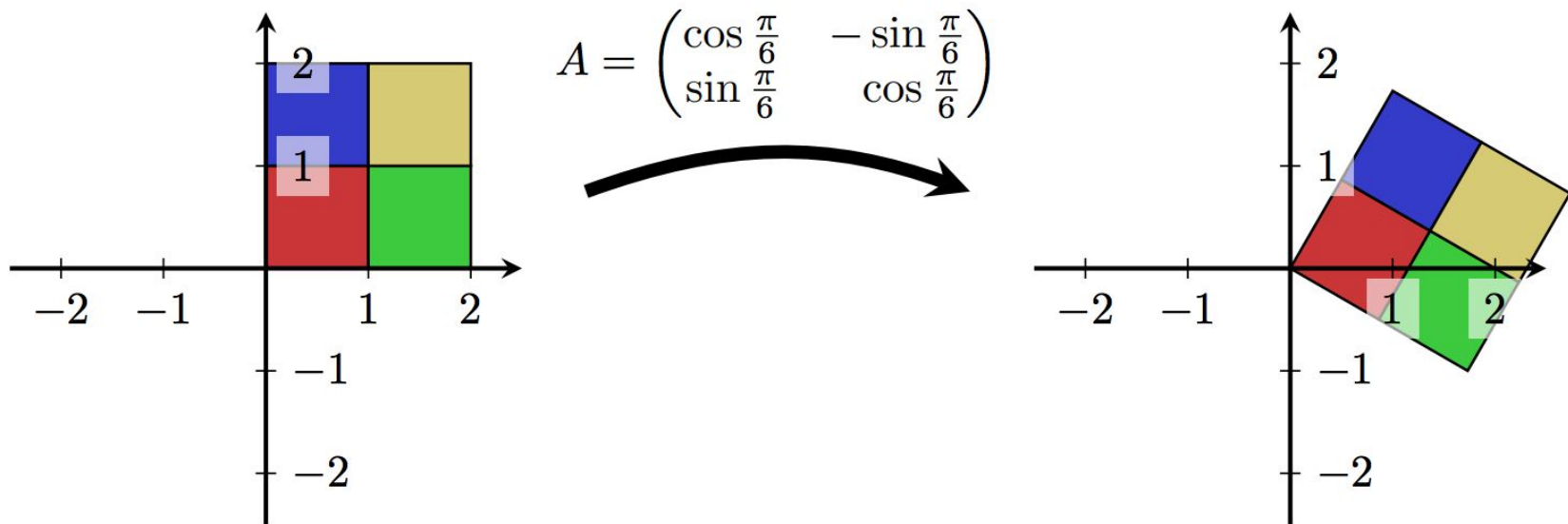
Типы матриц (преобразование пространства)



Типы матриц (преобразование пространства)



Типы матриц (преобразование пространства)



Типы матриц (преобразование пространства)

Масштабирование

$$\begin{bmatrix} S_1 & 0 & 0 & 0 \\ 0 & S_2 & 0 & 0 \\ 0 & 0 & S_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} S_1 \cdot x \\ S_2 \cdot y \\ S_3 \cdot z \\ 1 \end{pmatrix}$$

Сдвиг

$$\begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + T_x \\ y + T_y \\ z + T_z \\ 1 \end{pmatrix}$$

Типы матриц (преобразование пространства)

Матрица вращения вокруг оси X:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ \cos \theta \cdot y - \sin \theta \cdot z \\ \sin \theta \cdot y + \cos \theta \cdot z \\ 1 \end{pmatrix}$$

Матрица вращения вокруг оси Z:

$$\begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \cos \theta \cdot x - \sin \theta \cdot y \\ \sin \theta \cdot x + \cos \theta \cdot y \\ z \\ 1 \end{pmatrix}$$

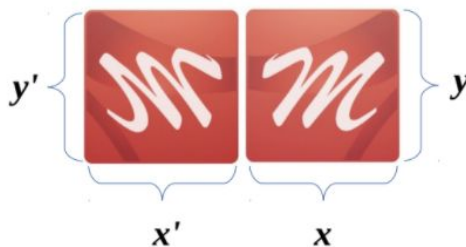
Матрица вращения вокруг оси Y:

$$\begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \cos \theta \cdot x + \sin \theta \cdot z \\ y \\ -\sin \theta \cdot x + \cos \theta \cdot z \\ 1 \end{pmatrix}$$

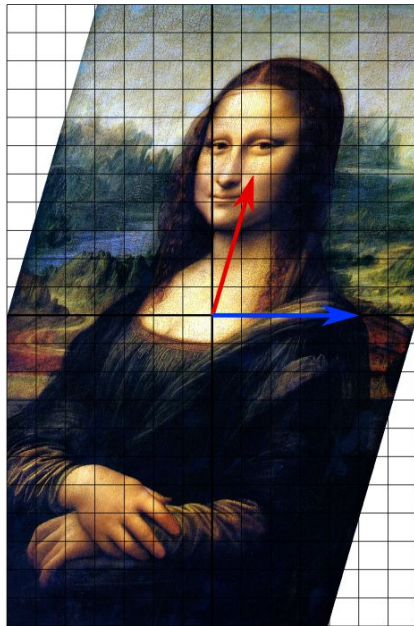
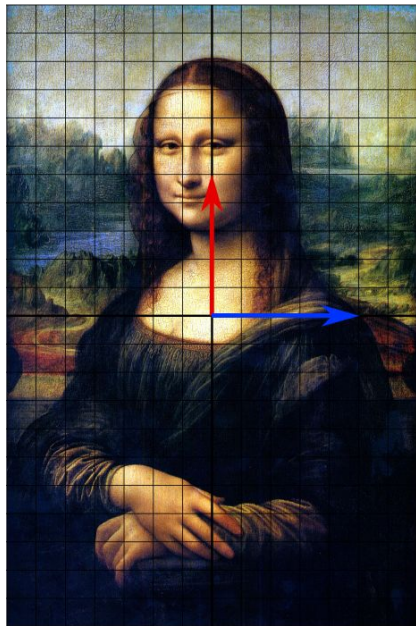
Типы матриц (преобразование пространства)

Чтобы получить зеркальное отображение объекта по горизонтали следует установить значение $a = -1$, по вертикали $d = -1$. Изменение обеих значений применяется для одновременного отображения по горизонтали и вертикали.

$$\begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



Собственные векторы и собственные значения



Собственный вектор преобразования A

$$AX = \lambda X$$

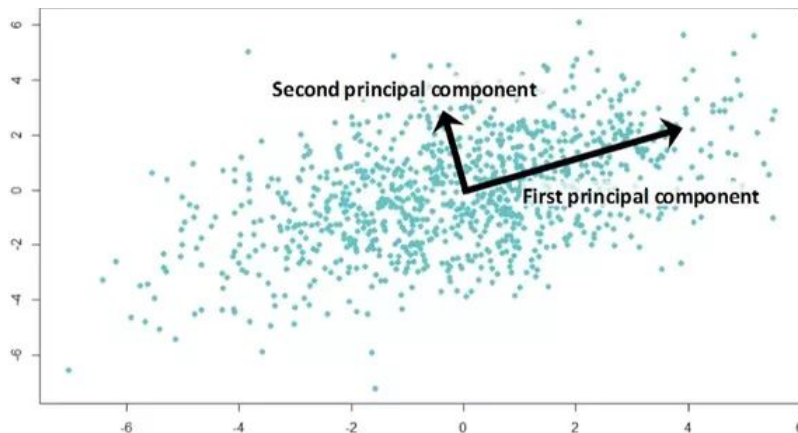
X - собственный вектор (ненулевой!)
 λ - собственное значение

!

У матрицы $n \times n$ не более n
собственных значений

Собственные векторы и собственные значения: применение

1. Собственные векторы - направления, в которых матрица лишь растягивает или сжимает векторы, но не поворачивает
2. Показывают направления наибольшего изменения
3. Возникают при уменьшении размера матрицы



Матричные разложения (спектральное разложение)

Разложение матрицы - представление в виде произведения некоторых других, обладающих интересными свойствами.

Пример: спектральное разложение X

$$X = S^T \cdot D \cdot S$$



X - симметричная, S - ортогональная, D - диагональная из собственных значений X .

Часто встречаются квадратичные формы

$$f(y) = y^T X y$$

с помощью спектрального разложения приводим к более простому виду:

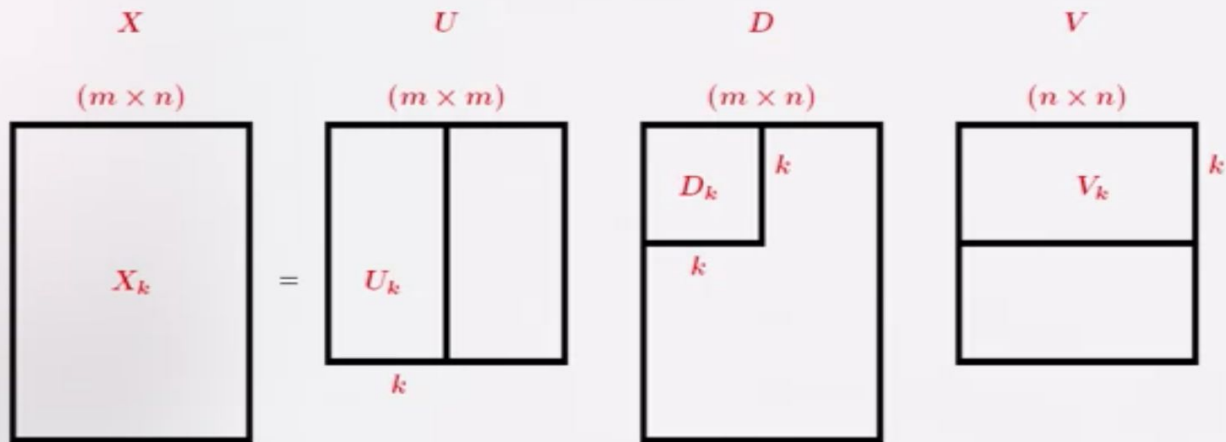
$$f(y) = y^T \cdot S^T \cdot D \cdot S \cdot y = (S \cdot y)^T \cdot D \cdot (S \cdot y) = z^T \cdot D \cdot z = \sum_{i=1}^n \lambda_i z_i^2,$$

Матричные разложения (сингулярное разложение)

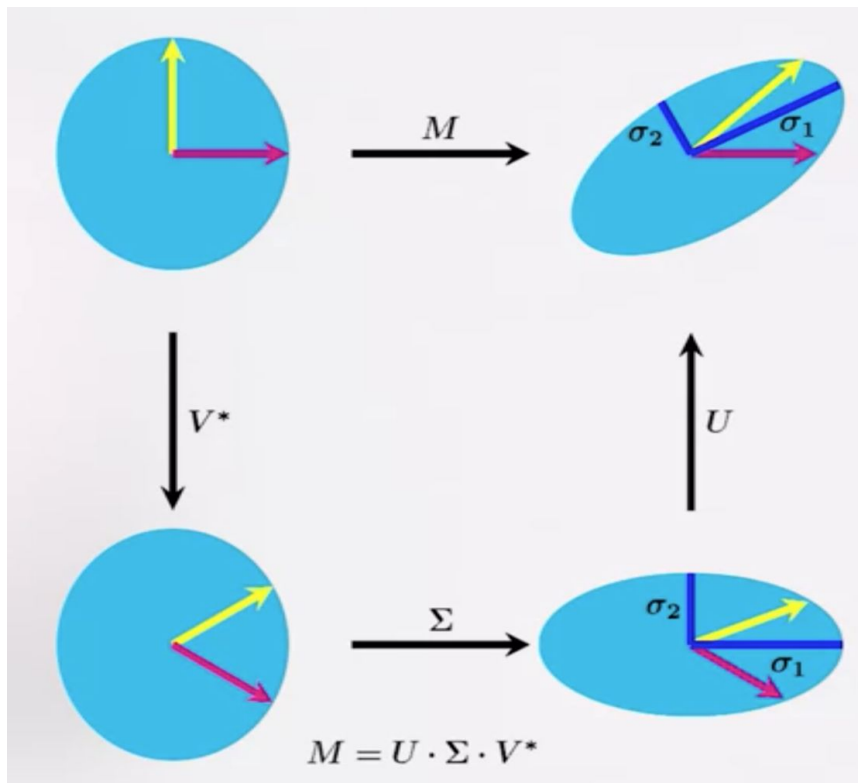
Но это была симметричная матрица, что в случае произвольной?

$$\triangleright X = UDV$$

$\triangleright U, V$ — ортогональные, D —
диагональная



Матричные разложения (сингулярное разложение)



Сингулярное разложение представляет линейное преобразование в виде композиции: вращения, растяжения по осям, вращения.

Матричные разложения (приближение матрицей меньшего ранга)

- Матрица задаёт отображение, ранг в какой-то степени мера “сложности” отображения
- Ранг - максимальное количество линейно независимых столбцов или строк
- Ранг - максимальный размер подматрицы с ненулевым определителем



$\text{rank}(X) \leq \min(n, m)$, если X - матрица $m \times n$

Пусть $X = AB$, A размера (m, k) , B размера (k, n)

Пусть также $k < m$, $k < n$

Что можно сказать о ранге X ?

Матричные разложения (приближение матрицей меньшего ранга)

Зачем приближать матрицу матрицей меньшего ранга?

Мы предполагаем, что матрица преобразования X на самом деле более простая.

Что значит приблизить

$$\triangleright X \approx X' = UV^T$$

$$\triangleright U - m \times k, V - n \times k$$

Просто наилучшее приближение по
норме: $\|X - UV^T\| \rightarrow \min$

Матричные разложения (приближение матрицей меньшего ранга)

Что значит приблизить

$$\triangleright X \approx X' = UV^T$$

$$\triangleright U - m \times k, V - n \times k$$

Просто наилучшее приближение по
норме: $\|X - UV^T\| \rightarrow \min$

$$\|X\|_F = \sqrt{\sum_{i,j} x_{ij}^2}$$

Итоговая задача выглядит так:

$$U, V = \underset{U \in \mathbb{R}^{m \times k}, V \in \mathbb{R}^{n \times k}}{\operatorname{argmin}} \sum_{i,j} (x_{ij} - u_i^T v_j)^2$$

Матричные разложения (пример применения)

- › Пусть X – матрица признаков объектов
- › Тогда U – матрица новых признаков
- › При $k < n$ преобразование признаков понижает размерность пространства
- › По U с максимальной возможной точностью восстанавливаются исходные признаки X

Матричные разложения (пример применения)

- › Пусть X – матрица с оценками x_{ij} , поставленными пользователем i фильму j
- › Некоторые значения матрицы неизвестны
- › $x_{ij} \approx \widehat{x_{ij}} = u_i v_j$, где u_i отражает интересы пользователя, а v_j – признаковое описание фильма
- › Идея: настроим u_i и v_j на известных x_{ij} , а неизвестные спрогнозируем
- › Будем рекомендовать фильмы, для которых спрогнозирована высокая оценка

Что делать с пропущенными значениями?

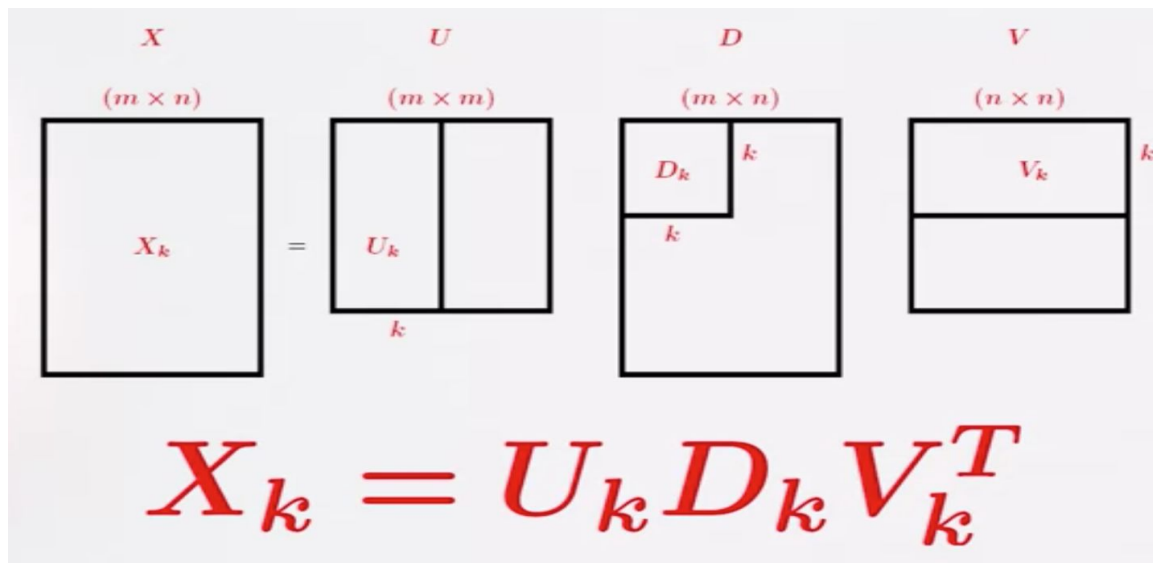
Матричные разложения (пример применения)

- › Пусть X – матрица с оценками x_{ij} , поставленными пользователем i фильму j
- › Некоторые значения матрицы неизвестны
- › $x_{ij} \approx \widehat{x_{ij}} = u_i v_j$, где u_i отражает интересы пользователя, а v_j – признаковое описание фильма
- › Идея: настроим u_i и v_j на известных x_{ij} , а неизвестные спрогнозируем
- › Будем рекомендовать фильмы, для которых спрогнозирована высокая оценка

$$U, V = \underset{U \in \mathbb{R}^{m \times k}, V \in \mathbb{R}^{n \times k}}{\operatorname{argmin}} \sum_{i,j: x_{ij} \neq 0} (x_{ij} - u_i^T v_j)^2$$

Матричные разложения (связь SVD и низкорангового приближения)

$$\hat{X} = \underset{\text{rg } \hat{X} \leq k}{\operatorname{argmin}} \|X - \hat{X}\| \quad X = U \cdot D \cdot V^T$$



Усечённый SVD

Матричные разложения (связь SVD и низкорангового приближения)

Оказывается, X_k - наилучшее приближение матрицы X матрицей ранга $\leq k$ по норме Фробениуса!

$$X_k = U_k D_k V_k^T$$

$$\hat{X}_k = \operatorname{argmin}_{\operatorname{rg}(\hat{X}) \leq k} ||X - \hat{X}||_F$$

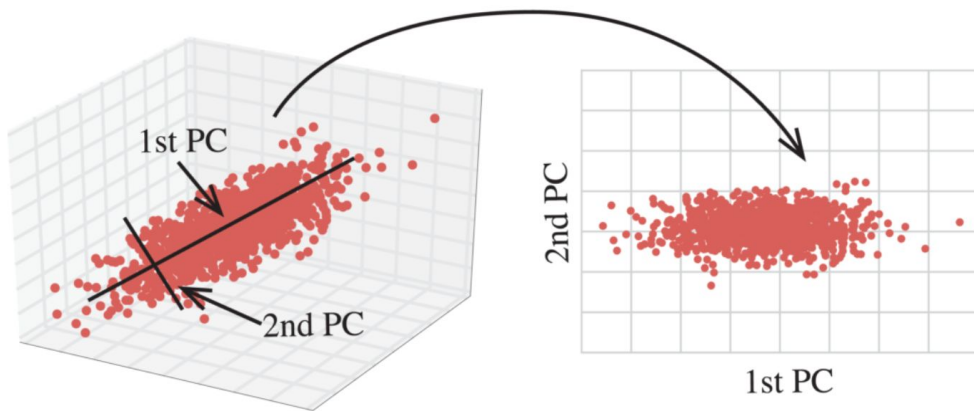
$$||X - \hat{X}||_F = \sqrt{\sum_{i,j} (x_{ij} - \hat{x}_{ij})^2}$$

Матричные разложения (рекомендательные системы)

» Вариант 1 (не очень правильно, но просто): сделать SVD, матрицу $U_k D_k$ использовать как матрицу профилей пользователей, а матрицу V_k как матрицу профилей фильмов, произведение профилей – прогноз оценки фильма

» Вариант 2 (более правильно, но нужно более глубоко вникнуть в метод): Не будем никак использовать SVD, а просто подберем U и V , минимизируя функционал

Понижение размерности: PCA

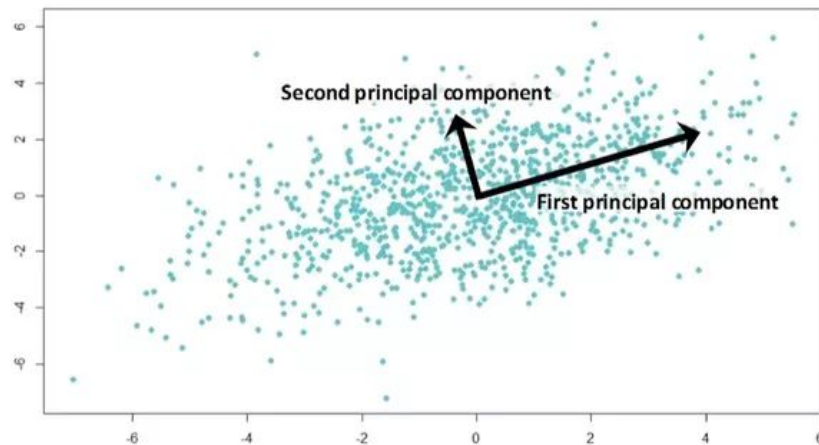


Один из основных способов уменьшить размерность данных, потеряв наименьшее количество информации.



Понижение размерности: РСА

- 1) Вычисляем матрицу ковариаций признаков
- 2) Находим собственные вектора матрицы ковариаций
- 3) Первые k векторов соответствующих k максимальным собственным значениям - компоненты нашего разложения



<https://medium.com/@sadatnazrul/the-dos-and-donts-of-principal-component-analysis-7c2e9dc8cc48>

Плюсы: возможность регуляции получаемой размерности (добавлении компонент по одной в зависимости от объяснённой дисперсии); скорость алгоритма; интерпретация

Минусы: линейность, предположение об ортогональности

Производная функции

Производная - мгновенная скорость роста функции в заданной точке.

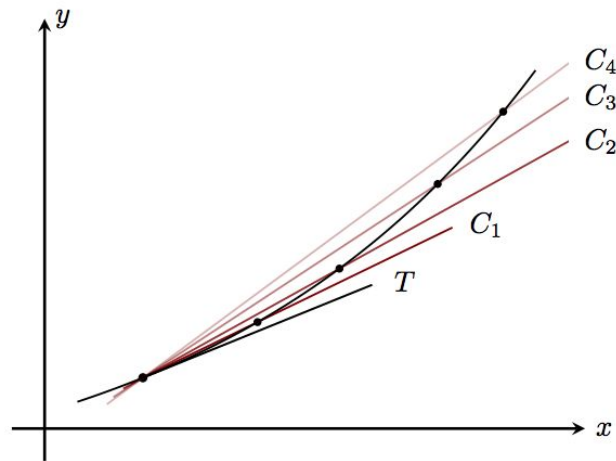
$$\frac{f(x + \Delta x) - f(x)}{\Delta x} = k.$$

Давайте посмотрим на линейную функцию $y=kx+b$

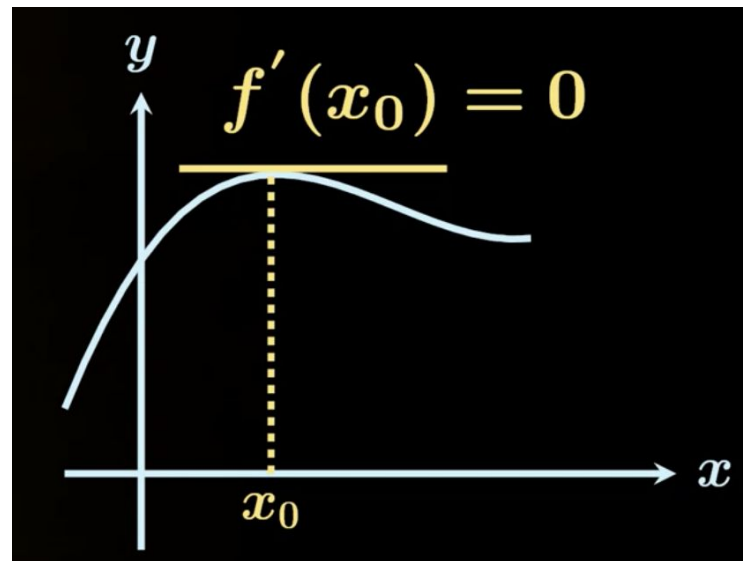
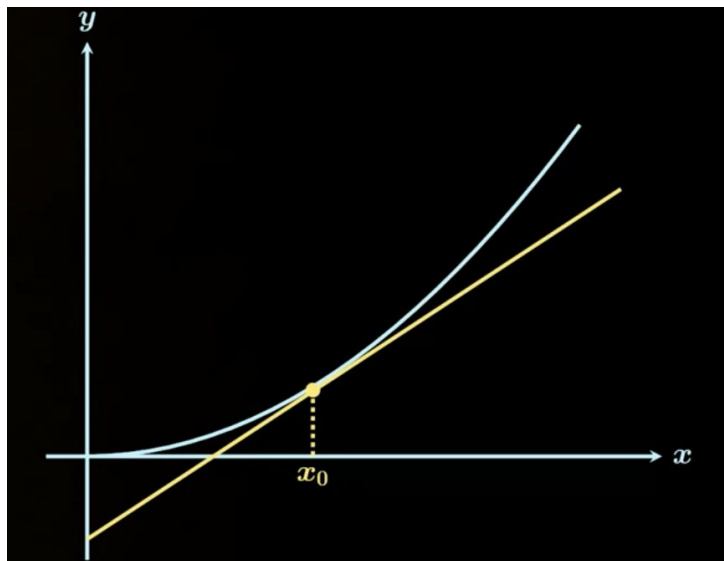
Как понять скорость роста для произвольной функции? **Предел!**

$$f'(x) = \lim_{\Delta x \rightarrow 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}.$$

Гладкие функции - функции, производная которых непрерывна.

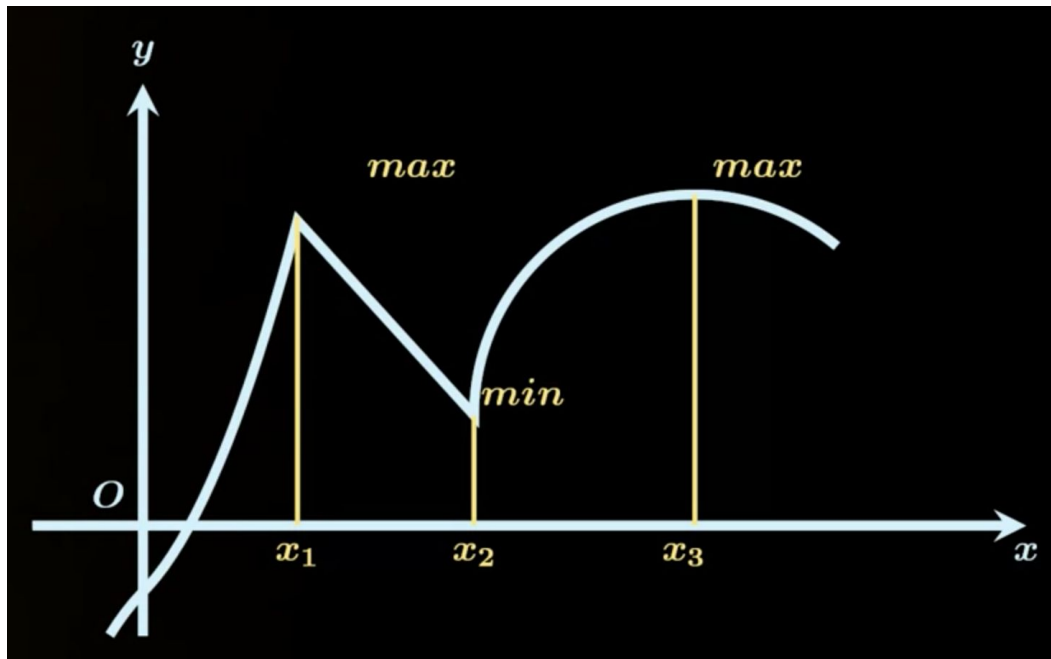
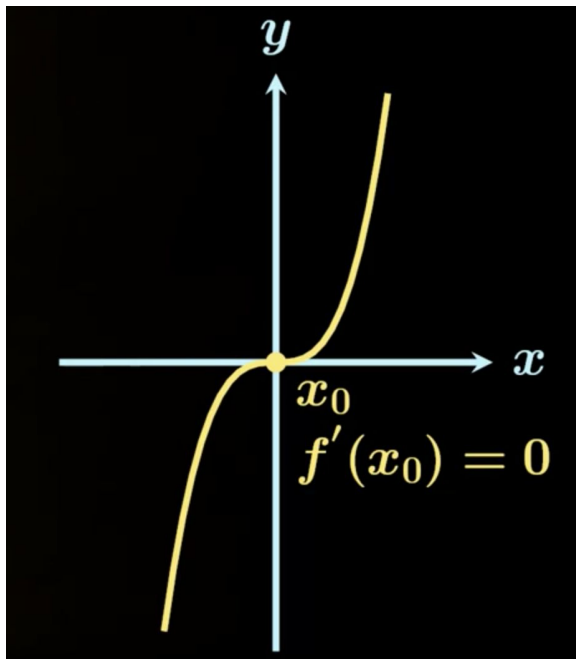


Экстремум функции и производная



В точках локальных экстремумов производная (если она определена (!), пример дальше) обязана равняться нулю.
Это **необходимое** условие.

Экстремум функции и производная



Однако равенство нулю производной **не является достаточным** условием локального экстремума. Также производная может быть вовсе не определена в точках локальных экстремумов.

Градиент и линии уровня функции

Если $f(x_1, \dots, x_n)$ — функция n переменных x_1, \dots, x_n , то n -мерный вектор из частных производных:

$$\text{grad } f = \left(\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n} \right)$$



называется **градиентом функции**.



Линией уровня функции называется множество точек, в которых функция принимает одно и то же фиксированное значение. Оказывается, что **градиент перпендикулярен линии уровня**.

Градиент в задачах оптимизации

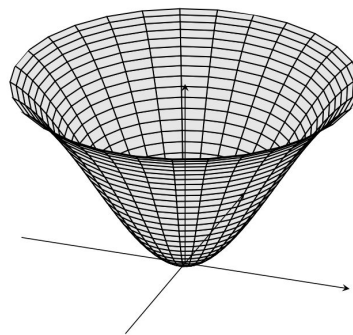
Задачей оптимизации называется задача по нахождению экстремума функции, например минимума:

$$f(x_1, \dots, x_n) \rightarrow \min$$

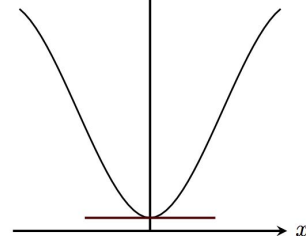
Такая задача часто встречается в приложениях, например при выборе оптимальных параметров рекламной компании, а также в задачах классификации.

Градиент в задачах оптимизации

Но не всегда задачу можно решать аналитически. В таком случае используется численная оптимизация. Наиболее простым в реализации из всех методов численной оптимизации является метод градиентного спуска.



$$\left. \frac{\partial f}{\partial x} \right|_{(0,0)} = 0$$



$$\left. \frac{\partial f}{\partial y} \right|_{(0,0)} = 0$$

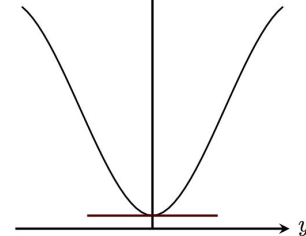


Рис. 2: Функция двух переменных достигает минимума в начале координат.

Градиентный спуск

Это итерационный метод. Решение задачи начинается с выбора начального приближения $\vec{x}^{[0]}$

После вычисляется приблизительное значение \vec{x}^1

Затем \vec{x}^2

и так далее...

! $\vec{x}^{[j+1]} = \vec{x}^{[j]} - \gamma^{[j]} \nabla F(\vec{x}^{[j]}),$ где $\gamma^{[j]}$ — шаг градиентного спуска.

Идея: идти в направлении наискорейшего спуска, а это направление задаётся антиградиентом $-\nabla F$.

Градиентный спуск

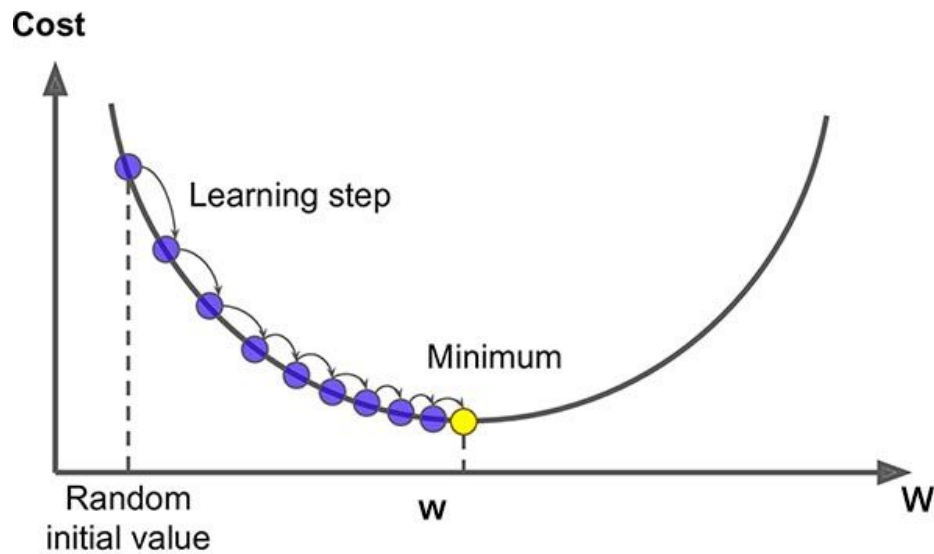
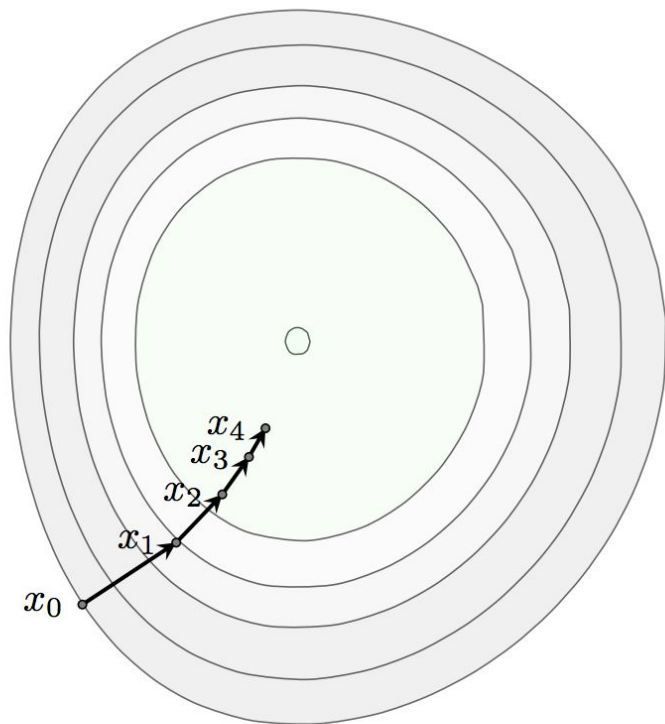


Рис. 3: Градиентный спуск

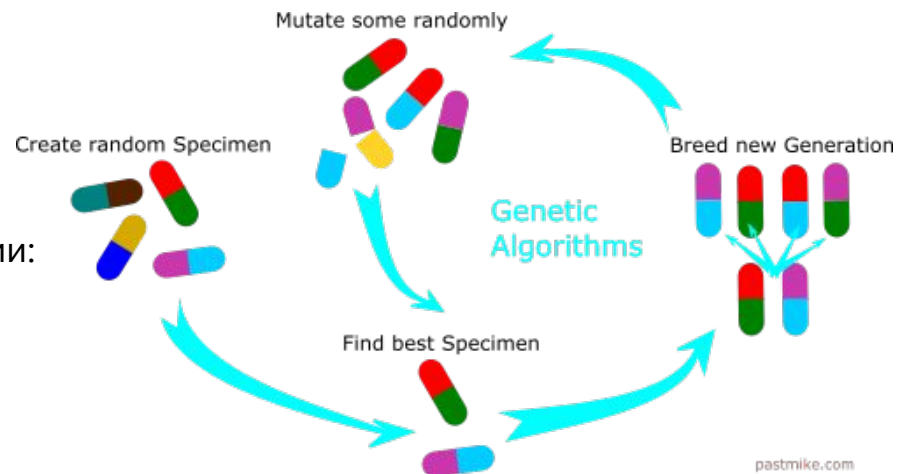
Генетические алгоритмы

Генетические алгоритмы моделируют процесс естественного отбора в ходе эволюции и являются еще одним семейством методов оптимизации.

Генетические алгоритмы включают в себя стадии:

1. генерации популяции
2. мутаций
3. скрещивания
4. отбора

Порядок стадий зависит от конкретного алгоритма.



Алгоритм дифференциальной эволюции

Для оптимизации функции $f(\vec{x})$ вещественных переменных применяется алгоритм дифференциальной эволюции.

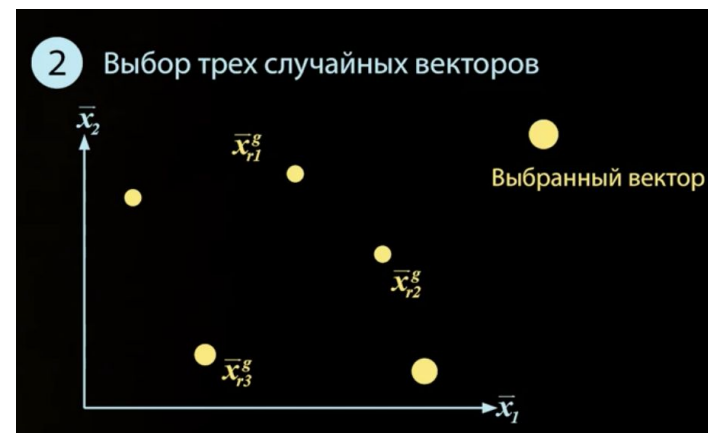
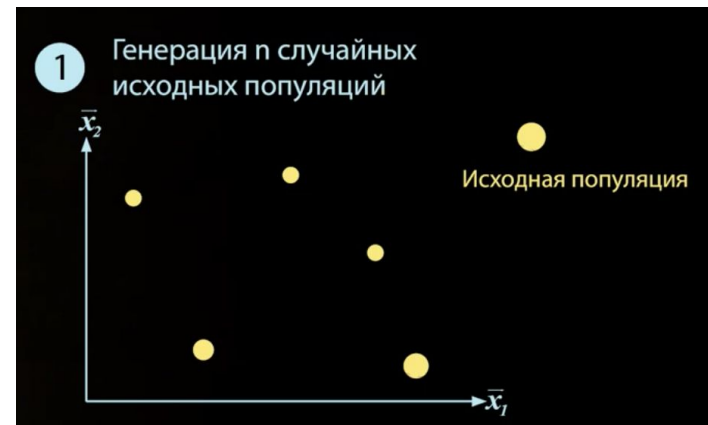
Популяция - множество векторов \mathbb{R}^n

Размер популяции - N

Сила мутации - F [0, 2]

Вероятность мутации - P

В качестве начальной популяции выбирается набор из N случайных векторов. На каждой следующей итерации алгоритм генерирует новое поколение векторов, комбинируя векторы предыдущего поколения.

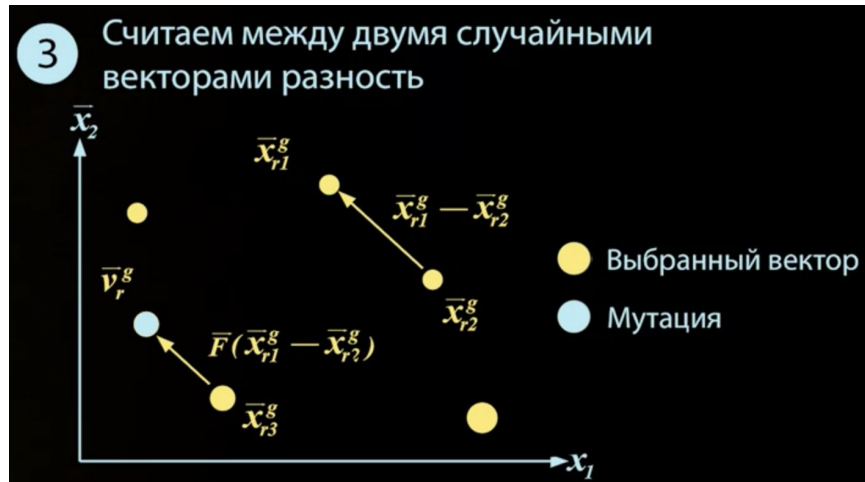


Алгоритм дифференциальной эволюции

На каждой итерации для каждого вектора x_i выбираются 3 неравные ему вектора v_1, v_2, v_3 . На основе этих векторов генерируется так называемый **мутантный** вектор:

$$v = v_1 + F \cdot (v_2 - v_3)$$

На **стадии скрещивания** каждая координата мутантного вектора с вероятностью p замещается соответствующей координатой вектора x_i . Получившийся вектор называется пробным.



Алгоритм дифференциальной эволюции

На **стадии отбора**, если пробный вектор оказывается лучше исходного x_i (то есть значение исследуемой функции на пробном векторе меньше, чем на исходном), то в новом поколении он занимает его место.

Если сходимость не была достигнута, начинается новая итерация.

Рассмотренный алгоритм хорошо применим для функций, зависящих от векторов действительных чисел. Часто встречаются зависимости от бинарных векторов.

- › Вариант 1. Если сын лучше родителя — оставляем его, если нет — родителя.
- › Вариант 2. Сначала проделать мутации и скрещивания для всей популяции, а потом отобрать N лучших.

Алгоритм дифференциальной эволюции

- В случае бинарных векторов можно определить **мутацию** следующим образом: с вероятностью p менять 0->1 и 1->0 в исходных векторах. **Скрещивание** без изменений
- Часто, чтобы увеличить эффективность алгоритма, создаются несколько независимых популяций. Для каждой такой популяции формируется свой начальный набор случайных векторов. В таком случае появляется возможность использовать генетические алгоритмы для решения задач глобальной оптимизации.
- Эффективность метода зависит от выбора операторов мутации и скрещивания для каждого конкретного типа задач.