



egaoharu_kensei 9 мар в 19:56

Метод главных компонент (PCA). Принцип работы и реализация с нуля на Python



Сложный



8 мин



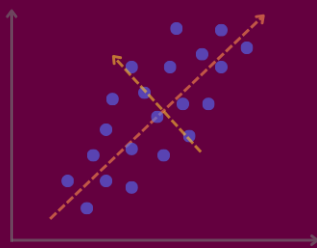
6.2K

Python*, Data Mining*, Алгоритмы*, Машинное обучение*, Искусственный интеллект

Тutorial

Метод главных компонент (PCA)

с нуля на Python



Метод главных компонент (Principal Component Analysis или же PCA) — алгоритм обучения без учителя, используемый для понижения размерности и выявления наиболее информативных признаков в данных. Его суть заключается в предположении о линейности отношений данных и их проекции на подпространство ортогональных векторов, в которых

дисперсия будет максимальной.

Такие вектора называются главными компонентами и они определяют направления наибольшей изменчивости (информативности) данных. Альтернативно суть PCA можно определить как линейное проецирование, минимизирующее среднеквадратичное расстояние между исходными точками и их проекциями.

Ноутбук с данным алгоритмом можно загрузить на [Kaggle](#) (eng) и [GitHub](#) (rus).

Принцип работы PCA

Изначально матрица признаков обязательно центрируется, чтобы первая главная компонента могла соответствовать направлению максимальной вариации данных, а не просто их среднему значению. Обычно нахождение главных компонент сводится к двум основным методам:

- **Вычисление собственных векторов и собственных значений ковариационной матрицы данных.** Поскольку ковариационная матрица отражает степень линейной связи между различными переменными, то собственные вектора этой матрицы будут задавать направления, вдоль которых дисперсия данных максимальна, а собственные значения — величину этой дисперсии. Собственное значение, соответствующее собственному вектору,



+9



99



0

собственное значение, тем значимее главная компонента. Обычно отбираются только те главные компоненты, которые объясняют заданный уровень дисперсии, например, 95%.

- **Вычисление сингулярного разложения матрицы данных.** Сингулярное разложение — это способ представления любой матрицы в виде произведения трёх других матриц: левой сингулярной матрицы U , диагональной матрицы сингулярных значений S и правой сингулярной матрицы V , где сингулярные значения — это квадратные корни собственных значений ковариационной матрицы данных (именно для этого в данном случае выполняется предварительное центрирование данных), правая сингулярная матрица V будет соответствовать собственным векторам ковариационной матрицы данных, а левая U будет являться проекцией исходных данных на главные компоненты, определённые матрицей V . Таким образом, сингулярное разложение также позволяет выделить главные компоненты, но без необходимости в вычислении ковариационной матрицы. Помимо того, что такое решение более эффективно, оно считается более численно стабильным, поскольку не требует вычисления ковариационной матрицы напрямую, которая может быть плохо обусловлена в случае сильной корреляции признаков. Именно данный подход используется в реализации scikit-learn, но с некоторыми особенностями, рассмотренными ниже.

PCA на основе SVD строится следующим образом:

- 1) сначала происходит центрирование данных, а также определяется число компонент как минимум между числом образцов и признаков в случае, если число компонент не было задано;
- 2) далее SVD применяется к центрированной матрице данных;
- 3) к матрице U применяется метод `svd_flip_vector`, который находит максимальные по модулю элементы в каждом столбце матрицы U , извлекает их знаки и умножает матрицу U на эти знаки, чтобы гарантировать детерминированный вывод;
- 4) объяснённая дисперсия для каждой главной компоненты вычисляется как возведённые в квадрат соответствующие сингулярные значения, разделённые на $n_samples - 1$, а преобразованные данные вычисляются с учётом числа главных компонент по правилу
$$X_{new} = X \cdot V = U \cdot S \cdot V^T \cdot V = U \cdot S.$$

Дополнительные возможности PCA

Коэффициент объяснённой дисперсии каждой главной компоненты, доступный через переменную `explained_variance_ratio_`, указывает долю дисперсии датасета, лежащей вдоль оси каждой главной компоненты.

Восстановление данных с помощью метода `inverse_transform()` заключается в применении обратной трансформации проекции PCA вида $X_{recovered} = X_{d-proj} W_d^T$, где W_d^T — матрица из первых d главных компонент. Из этого следует, что данные будут восстановлены с потерями, пропорциональными количеству отброшенной дисперсии исходных данных, а средний квадрат расстояния между исходными и восстановленными данными представляет собой *ошибку восстановления (reconstruction error)*.

Инкрементный PCA, реализованный в виде класса *IncrementalPCA*, позволяет работать эффективнее с большими наборами данных за счёт их разбиения на мини-пакеты и поштучном хранении в памяти во время обучения.

Рандомизированный PCA, устанавливаемый с помощью параметра `svd_solver='randomized'`, использует стохастический алгоритм для быстрого вычисления приближённых d главных компонент и основан на предположении, что случайная проекция данных на низкоразмерное подпространство может хорошо сохранять их структуру и свойства, однако такой подход может быть менее точным.

Ядерный PCA, реализованный с помощью класса *KernelPCA*, позволяет выполнять сложные нелинейные проекции с использованием ядерных функций. Как и в случае с SVM, его суть в данном случае заключается в том, что линейная граница решений в многомерном пространстве признаков будет соответствовать сложной нелинейной границе в исходном пространстве.

Альтернативы PCA

Не смотря на то, что метод главных компонент является одним из самых популярных алгоритмов понижения размерности, существуют альтернативы, которые могут быть более предпочтительными в ряде ситуаций, а также в зависимости от типа данных:

- **LLE (Locally Linear Embedding)** — алгоритм создания линейных комбинаций каждой точки из её соседей с последующим восстановлением этих комбинаций в пространстве более низкой размерности, что позволяет сохранить нелинейную геометрию данных и быть полезным для некоторых задач, где глобальные свойства менее важны. С другой стороны, такой подход имеет высокую вычислительную сложность и может быть чувствителен к шуму.
- **t-SNE (t-Distributed Stochastic Neighbor Embedding)** — алгоритм, который преобразует сходства между данными в вероятности и в дальнейшем пытается минимизировать расхождение между распределениями вероятностей в пространстве высокой и низкой размерности. t-SNE эффективен при визуализации данных высокой размерности, однако может искажать глобальную структуру данных, поскольку не учитывает линейные зависимости, а лишь их близость в исходном пространстве.
- **UMAP (Uniform Manifold Approximation and Projection)** — ещё один алгоритм, подходящий для визуализации данных, который основан на идее, что данные лежат на некотором однородном многообразии, которое можно аппроксимировать с помощью графа соседей. Такой подход учитывает глобальную структуру данных и позволяет лучше адаптироваться к различным типам данных, а также лучше справляться с шумом и выбросами, чем t-SNE.
- **Autoencoders** — тип нейронных сетей, основанный на обучении кодировщика преобразовывать входные данные в низкоразмерное представление, с последующим обучением декодера восстанавливать исходные данные из этого представления. Autoencoders могут также использоваться для сжатия данных, удаления шума и многих других целей.

Импорт необходимых библиотек

```
import numpy as np
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris
```

Реализация на Python с нуля

```
class SVDPCA:
    def __init__(self, n_components=None):
        self.n_components = n_components

    @staticmethod
    def svd_flip_vector(U):
        max_abs_cols_U = np.argmax(np.abs(U), axis=0)
        # extract the signs of the max absolute values
        signs_U = np.sign(U[max_abs_cols_U, range(U.shape[1])])

        return U * signs_U

    def fit_transform(self, X):
        n_samples, n_features = X.shape
        X_centered = X - X.mean(axis=0)

        if self.n_components is None:
            self.n_components = min(n_samples, n_features)

        U, S, Vt = np.linalg.svd(X_centered)
        # flip the eigenvector sign to enforce deterministic output
        U_flipped = self.svd_flip_vector(U)

        self.explained_variance = (S[:self.n_components] ** 2) / (n_samples - 1)
        self.explained_variance_ratio = self.explained_variance / np.sum(self.explained_variance)

        # X_new = X * V = U * S * Vt * V = U * S
        X_transformed = U_flipped[:, : self.n_components] * S[: self.n_components]

        return X_transformed
```

Загрузка датасета

```
X, y = load_iris(return_X_y=True, as_frame=True)
print(X)
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
..
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

[150 rows x 4 columns]

Обучение моделей и оценка полученных результатов

Ручная реализация показала идентичные результаты scikit-learn. Как можно заметить, первые 2 главные компоненты объясняют практически 98% дисперсии в данных, что позволяет сократить количество признаков вдвое без особой потери информации. Если бы количество признаков было не 4, а несколько тысяч или миллионов, то это бы позволило существенно сократить время обучения моделей без значительной потери точности (а иногда и с увеличением точности за счёт уменьшения мультиколлинеарности между признаками), что делает PCA и его альтернативы прекрасным дополнением к другим алгоритмам.

PCA

```
pca = SVDPCA()
X_transformed = pca.fit_transform(X)

print('transformed data', X_transformed[:10], '', sep='\n')
print('explained_variance', pca.explained_variance)
print('explained_variance_ratio', pca.explained_variance_ratio)
```

transformed data

```
[[-2.68412563e+00  3.19397247e-01 -2.79148276e-02 -2.26243707e-03]
 [-2.71414169e+00 -1.77001225e-01 -2.10464272e-01 -9.90265503e-02]
 [-2.88899057e+00 -1.44949426e-01  1.79002563e-02 -1.99683897e-02]
 [-2.74534286e+00 -3.18298979e-01  3.15593736e-02  7.55758166e-02]
 [-2.72871654e+00  3.26754513e-01  9.00792406e-02  6.12585926e-02]
 [-2.28085963e+00  7.41330449e-01  1.68677658e-01  2.42008576e-02]
 [-2.82053775e+00 -8.94613845e-02  2.57892158e-01  4.81431065e-02]
 [-2.62614497e+00  1.63384960e-01 -2.18793179e-02  4.52978706e-02]
 [-2.88638273e+00 -5.78311754e-01  2.07595703e-02  2.67447358e-02]
 [-2.67275580e+00 -1.13774246e-01 -1.97632725e-01  5.62954013e-02]]
```

```
explained_variance [4.22824171 0.24267075 0.0782095 0.02383509]
explained_variance_ratio [0.92461872 0.05306648 0.01710261 0.00521218]
```

PCA (scikit-learn)

```
sk_pca = PCA()
sk_X_transformed = sk_pca.fit_transform(X)

print('sk transformed data', sk_X_transformed[:10], '', sep='\n')
print('sk explained_variance', sk_pca.explained_variance_)
print('sk explained_variance_ratio_', sk_pca.explained_variance_ratio_)
```

sk transformed data

```
[[-2.68412563e+00  3.19397247e-01 -2.79148276e-02 -2.26243707e-03]
 [-2.71414169e+00 -1.77001225e-01 -2.10464272e-01 -9.90265503e-02]
 [-2.88899057e+00 -1.44949426e-01  1.79002563e-02 -1.99683897e-02]
 [-2.74534286e+00 -3.18298979e-01  3.15593736e-02  7.55758166e-02]
 [-2.72871654e+00  3.26754513e-01  9.00792406e-02  6.12585926e-02]
 [-2.28085963e+00  7.41330449e-01  1.68677658e-01  2.42008576e-02]
 [-2.82053775e+00 -8.94613845e-02  2.57892158e-01  4.81431065e-02]
 [-2.62614497e+00  1.63384960e-01 -2.18793179e-02  4.52978706e-02]
 [-2.88638273e+00 -5.78311754e-01  2.07595703e-02  2.67447358e-02]
 [-2.67275580e+00 -1.13774246e-01 -1.97632725e-01  5.62954013e-02]]
```

```
sk explained_variance [4.22824171 0.24267075 0.0782095 0.02383509]
sk explained_variance_ratio_ [0.92461872 0.05306648 0.01710261 0.00521218]
```

Преимущества и недостатки

Преимущества:

- понижение размерности с сохранением большого количества информации, что также позволяет визуализировать данные высокой размерности в двумерном или трёхмерном пространстве;
- не только позволяет значительно ускорить обучение, но и уменьшить переобучение моделей в ряде случаев;
- может использоваться для сжатия данных.

Недостатки:

- неизбежная потеря части информации в данных;
- поиск только линейной зависимости в данных (в обычном PCA);

- отсутствие смыслового значения главных компонент из-за трудности их связывания с реальными признакам.

Дополнительные источники

Статьи:

- «A Tutorial on Principal Component Analysis», Jonathon Shlens;
- «Locally Linear Embedding and its Variants: Tutorial and Survey», Benyamin Ghogogh, Ali Ghodsi, Fakhri Karray, Mark Crowley;
- «Theoretical Foundations of t-SNE for Visualizing High-Dimensional Clustered Data», T. Tony Cai, Rong Ma;
- «UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction», Leland McInnes, John Healy, James Melville;
- «Deep Autoencoders for Dimensionality Reduction of High-Content Screening Data», Lee Zamparo, Zhaolei Zhang.

Документация:

- описание PCA;
- описание LLE;
- описание t-SNE;
- PCA;
- LLE;
- t-SNE;
- UMAP.

Видео:

- PCA: один, два, три, четыре;
- LLE;
- t-SNE;
- UMAP;
- autoencoders.

Теги: метод главных компонент, principal component analysis, pca, алгоритмы машинного обучения, python, data science, машинное обучение, глубокое обучение, обучение без учителя, снижение размерности

Хабы: Python, Data Mining, Алгоритмы, Машинное обучение, Искусственный интеллект



27

91

Карма Рейтинг

Егор Захаренко @egaoharu_kensei

Пользователь

Подписан ×

Комментировать

Публикации

ЛУЧШИЕ ЗА СУТКИ ПОХОЖИЕ



Grigory_Otrepyev 5 часов назад

Российская микроэлектроника — два года спустя



Сложный



9 мин



13K

Дайджест



+140



42



59

+59



bodyawm 6 часов назад

В моих жилах течет моддерская кровь: как и зачем я променял оригинальный айфон на нерабочую подделку за 1500 рублей?



Средний



18 мин



2.9K

Обзор



+34



19



22

+22



HallEffect вчера в 17:13

Ферма тестирования SberDevices

 Средний  14 мин  3.7K

Обзор

 +34  26  4 +4



obondar 5 часов назад

От хаоса к порядку. Как мы внедряем стандарты в CDEK

 Средний  11 мин  817

Кейс

 +31  40  6 +6



wofs 7 часов назад

Как мы сделали Embedded Controller для ПЛК на Linux

 Средний  15 мин  2.5K

Кейс

 +28  16  14 +14



Lunathecatt 5 часов назад

Паяем классическую педаль Marshall Bluesbreaker

 Простой  8 мин  1K

Ретроспектива

 +27  10  0



vickylikh 8 часов назад

Будущее Kubernetes и DevOps: строим прогнозы на 10 лет

 Простой  13 мин  3.5K

Аналитика

 +22  15  5 +5



zakhmatov 7 часов назад

Как DDoS-атаки стали для нас рутиной и как ML помогает их отражать

 10 мин  955



rananyev 23 часа назад

Sub-GHz во Flipper Zero и бесконечное множество внешних антенн

Простой 6 мин 4.2K

Обзор

+19 33 17 +17



vasilisa_b 7 часов назад

Как в России в XIX веке компьютер изобрели

12 мин 1.6K

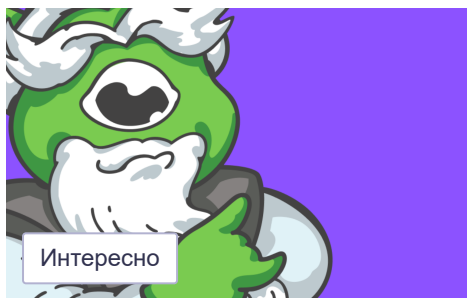
+18 12 2 +2

Всему учён и изловчён: где взять знания, которые в работе точно пригодятся

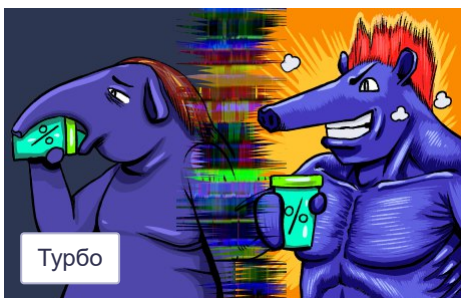
Турбо

Показать еще

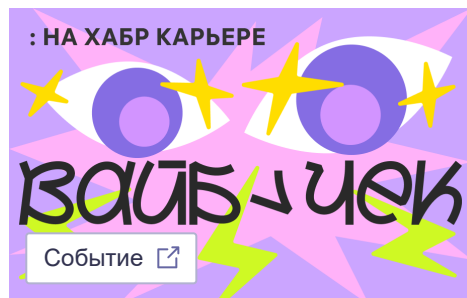
МИНУТОЧКУ ВНИМАНИЯ



Глупым вопросам и ошибкам — быть! IT-менторство на ХК



Как бессонница в час ночной, меняет промокодище облик твой




Выбирайте команду тестирования по вайбам

КУРСЫ


Администрирование PostgreSQL 13. Настройка и мониторинг
25 марта 2024 · Hi-TECH Academy

Продвинутый SQL

25 марта 2024 · Нетология

 SQL и получение данных

25 марта 2024 · Нетология

 Django: создание backend-приложений

25 марта 2024 · Нетология

 Аналитик данных: расширенный курс

25 марта 2024 · Нетология

Больше курсов на Хабр Карьере

ЧИТАЮТ СЕЙЧАС

Учёные назвали ударом для российской школы физики элементарных частиц последствия прекращения сотрудничества с ЦЕРН

 13K  52 **+52**

Российская микроэлектроника — два года спустя

 13K  58 **+58**

Разработчик с помощью дипфейка в реальном времени прошёл собеседование за друга

 10K  66 **+66**

Очередное пособие по рынку труда, или где же вы 300к находите. Март 2024

 41K  188 **+188**

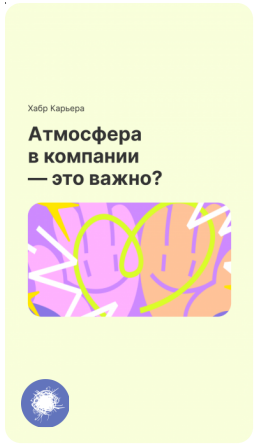
Десятки ведущих учёных подписали документ, направленный на предотвращение разработки биологического оружия при помощи ИИ

 30K  41 **+41**

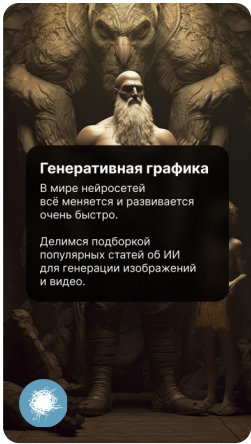
Всему учён и изловчён: где взять знания, которые в работе точно пригодятся

Турбо

ИСТОРИИ



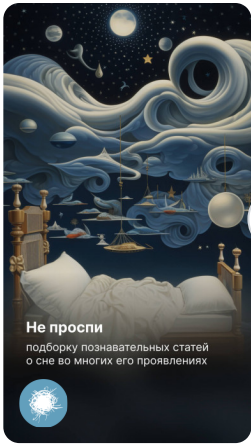
Вайб-чек для тестировщиков



Нейросети: интересное



Как продвинуть машину времени?



Наука сна

РАБОТА

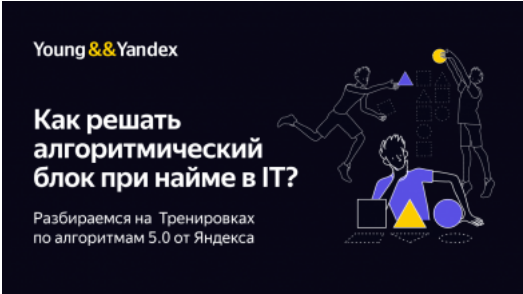
Python разработчик
127 вакансий

Django разработчик
38 вакансий

Data Scientist
61 вакансия

Все вакансии

БЛИЖАЙШИЕ СОБЫТИЯ



Серия занятий
«Тренировки по
алгоритмам 5.0» от
Яндекса

1 марта – 19 апреля
19:00
Онлайн

Подробнее в календаре



Тестировщики,
выбирайте себе команду
по вайбам на Хабр
Карьере

18 – 24 марта
09:00 – 23:00
Онлайн

Подробнее в календаре



«GoCloud 2024. Ог
рани будущего» -
конференция Clo
про облака

21 марта 09:00
Москва • Онлайн

Подробнее в календаре

Ваш аккаунт

- Профиль
- Трекер
- Диалоги
- Настройки
- ППА

Разделы

- Статьи
- Новости
- Хабы
- Компании
- Авторы
- Песочница

Информация

- Устройство сайта
- Для авторов
- Для компаний
- Документы
- Соглашение
- Конфиденциальность

Услуги

- Корпоративный блог
- Медийная реклама
- Нативные проекты
- Образовательные программы
- Стартапам



Настройка языка

Техническая поддержка