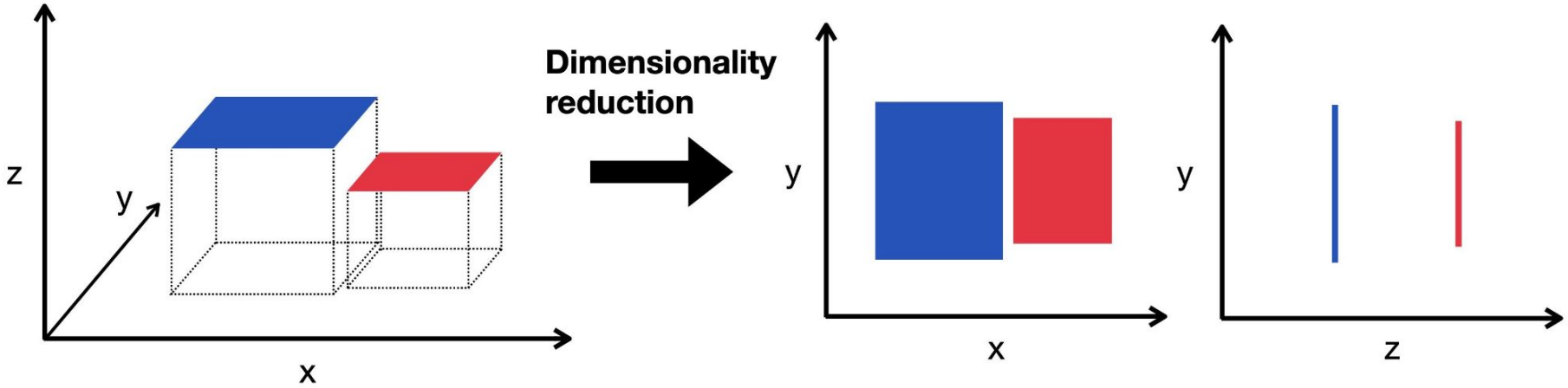
Зменшення розмірностей даних/ Dimensionality reduction



Зменшення розмірностей даних

В машинному навчанні методи пониження розмірності мають низку практичних застосувань, включаючи

- приховування конфіденційних даних,

- усунення мультиколінеарності,

- видалення шуму з наборів даних,

- покращення якості моделі за рахунок видалення неважливих ознак (feature selection)

- візуалізацію високовимірних даних шляхом зменшення їх до двох або трьох вимірів, які можна відобразити на графіку.

Основні методи пониження розмірностей

- **PCA (Principal Component Analysis):** Лінійний метод, який проектує дані на новий набір осей (головних компонент), що максимізують варіацію даних. Часто використовується для попередньої обробки перед застосуванням інших методів машинного навчання.
- **LDA (Linear Discriminant Analysis):** Лінійний метод, подібний до PCA, але на відміну від PCA, LDA фокусується на максимізації роздільності між класами. Використовується для задач класифікації.
- **t-SNE (t-Distributed Stochastic Neighbor Embedding):** Нелінійний метод, який добре зберігає локальні структури даних і зазвичай використовується для візуалізації складних даних у 2D або 3D просторах.
- **UMAP (Uniform Manifold Approximation and Projection):** Нелінійний метод, схожий на t-SNE, але часто більш швидкий і ефективний для великих наборів даних. Також добре зберігає як локальні, так і глобальні структури даних.

Основні методи пониження розмірностей

- **Autoencoders:** Нейронні мережі, які використовують вузький шар (кодування) для стиску даних, а потім відновлюють їх назад. Autoencoders можуть використовуватися як для нелінійного пониження розмірності, так і для навчання з витягуванням ознак.
- **ICA (Independent Component Analysis):** Нелінійний метод, що використовується для розділення сигналів на незалежні компоненти, корисний для задач, де важливо зберігати незалежність ознак.
- **Feature Selection:** Методи, які вибирають підмножину початкових ознак, відкидаючи менш важливі або корельовані. Приклади включають методи на основі взаємної інформації, кореляції або регуляризації (L1-реуляризація).

З якими алгоритмами познайомимось

Ми познайомимось детальніше з PCA та t-SNE. Бо ці алгоритми найчастіше можуть зустрітись на інтерв'ю і в задачах.

Алгоритм аналізу головних компонент / Principal Component Analysis (PCA)

Алгоритм аналізу головних компонент — це алгоритм навчання без вчителя, який використовується для стиснення набору даних у підпростір ознак меншої вимірності зі збереженням більшої частини релевантної інформації.

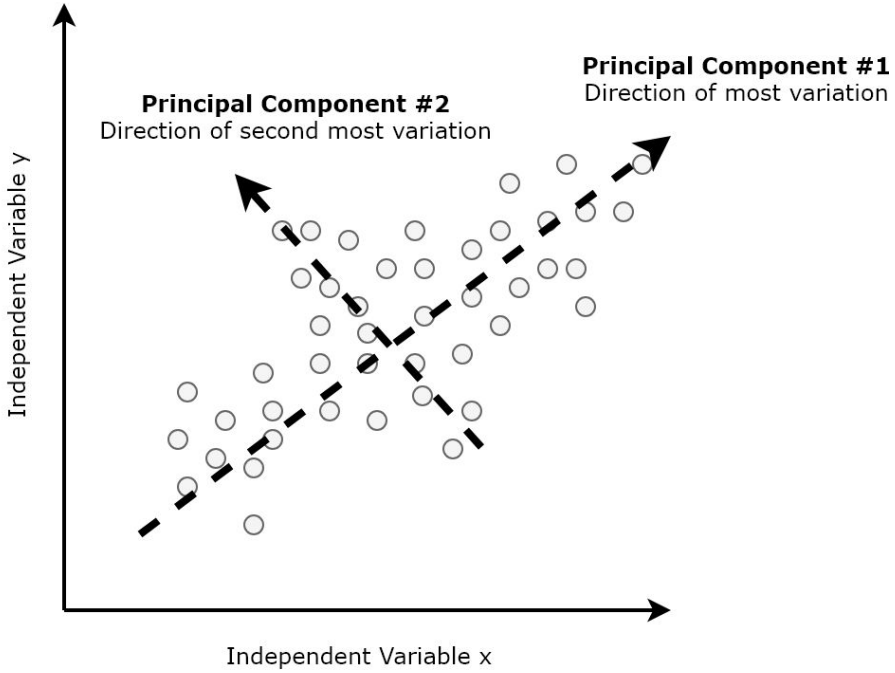
PCA допомагає нам визначати закономірності в даних на основі **кореляції** між ознаками.

PCA прагне знайти **напрямки максимальної дисперсії в багатовимірних даних** і проєктувати їх на новий підпростір з меншою вимірністю, ніж оригінальний.

Головні компоненти / Principal Components

Головні компоненти — це ортогональні осі нового підпростору.

Їх можна інтерпретувати як напрями максимальної дисперсії з урахуванням обмеження, що осі нових ознак ортогональні один одному, як показано на рисунку.



Принцип роботи PCA

Основні кроки PCA:

1. Центрування даних шляхом віднімання середнього.

2. Визначення коваріаційної матриці даних.

3. Обчислення власних векторів та власних значень коваріаційної матриці.

4. Вибір власних векторів з найбільшими власними значеннями для створення нового підпростору (з меншою розмірністю).

5. Проєктування даних на цей новий підпростір.

PCA по кроках

Беремо d-вимірний набір ознак.

Припустимо, у нас є набір даних з оцінками студентів з різних предметів:

Student	Math	English	Art
1	90	60	90
2	90	90	30
3	60	60	60
4	60	60	90
5	30	30	30

PCA по кроках

2. Обчислимо середнє значення кожного вимірювання всього набору даних.

$$\mathbf{A} = \begin{bmatrix} 90 & 60 & 90 \\ 90 & 90 & 30 \\ 60 & 60 & 60 \\ 60 & 60 & 90 \\ 30 & 30 & 30 \end{bmatrix}$$

$$\overline{\mathbf{A}} = [\,66 \quad 60 \quad 60\,]$$

PCA по кроках

3. Обчислимо коваріаційну матрицю всього набору даних.

	Math	English	Arts
1	90	60	90
2	90	90	30
3	60	60	60
4	60	60	90
5	30	30	30

$$cov(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(Xi-\bar{x})(Yi-\bar{y})$$

	Math	English	Art
Math	504	360	180
English	360	360	0
Art	180	0	720

- ➔ По діагоналі синім кольором позначений розкид оцінок по кожному тесту. Тест з мистецтва має найбільшу дисперсію (720), а найменшу дисперсію — тест з англійської мови (360). Таким чином, ми можемо сказати, що результати тестів з мистецтва більш різноманітні, ніж результати тестів з англійської мови.
- ➔ Коваріація відображається чорним кольором у недіагональних елементах матриці A.

PCA по кроках

4. Обчислимо **власні вектори** та відповідні **власні значення**.

Інтуїтивно власний вектор — це вектор, напрямок якого не змінюється при застосуванні до нього лінійного перетворення.

Нехай A — квадратна матриця, **v** — вектор і **λ** — скаляр, який задовольняє умові A**v** = **λv**, тоді **λ** називається власним значенням, пов'язаним з власним вектором **v** матриці A.

Для пошуку власних векторів використовується поняття визначника (determinant).

Знайдемо власні числа

det(*A*−λ*I*) = 0

$$\det\left(\begin{pmatrix} 504 & 360 & 180 \\ 360 & 360 & 0 \\ 180 & 0 & 720 \end{pmatrix} - \lambda \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\right)$$

$$\det\begin{pmatrix} 504-\lambda & 360 & 180 \\ 360 & 360-\lambda & 0 \\ 180 & 0 & 720-\lambda \end{pmatrix}$$

$$\begin{pmatrix} 504 & 360 & 180 \\ 360 & 360 & 0 \\ 180 & 0 & 720 \end{pmatrix} - \begin{pmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{pmatrix}$$

$$-\lambda^3 + 1584\lambda^2 - 641520\lambda + 25660800$$

$$\begin{pmatrix} 504-\lambda & 360 & 180 \\ 360 & 360-\lambda & 0 \\ 180 & 0 & 720-\lambda \end{pmatrix}$$

$$-\lambda^3 + 1584\lambda^2 - 641520\lambda + 25660800 = 0$$

$$\lambda \approx 44.81966\ldots, \lambda \approx 629.11039\ldots, \lambda \approx 910.06995\ldots$$

Знайдемо власні вектори

Тепер знайдемо власні вектори просто підставивши кожне власне значення у рівняння. Отримаємо 3 вектори.

$$\begin{pmatrix} -3.75100\ldots \\ 4.28441\ldots \\ 1 \end{pmatrix}, \begin{pmatrix} -0.50494\ldots \\ -0.67548\ldots \\ 1 \end{pmatrix}, \begin{pmatrix} 1.05594\ldots \\ 0.69108\ldots \\ 1 \end{pmatrix}$$

Як шукати власні вектори (більше покрокових прикладів):

https://math.libretexts.org/Bookshelves/Linear_Algebra/A_First_Course_in_Linear_Algebra_(Kuttler)/07%3A_Spectral_Theory/7.01%3A_Eigenvalues_and_Eigenvectors_of_a_Matrix

PCA по кроках

5. Відсортуємо власні вектори за спаданням власних значень і виберемо k власних векторів з найбільшими власними значеннями, щоб сформувати матрицю W розмірності d × k.

Відсортовані власні значення

$$\begin{pmatrix} 910.06995 \\ 629.11039 \\ 44.81966 \end{pmatrix}$$

Вибрані два власні вектори

$$\mathbf{W} = \begin{bmatrix} 1.05594 & -0.50494 \\ 0.69108 & -0.67548 \\ 1 & 1 \end{bmatrix}$$

PCA по кроках

6. Спроектуємо наші дані (ознаки) в нове підпростір.

На останньому етапі ми використовуємо тільки що обчислену матрицю W розміром 2 × 3 для перетворення наших вибірок у нове підпростір за допомогою рівняння y = W ' × x, де W ' — транспонована матриця W.

Отже, ми обчислили наші два основні компоненти та спроектували точки даних на новий підпростір.

Це можемо закодити в Python

І звісно в sklearn є вже реалізований алгоритм :)