Bab 8: Dimensionality Reduction

Tujuan Bab

Memahami cara mereduksi jumlah fitur (dimensi) dalam dataset untuk mempercepat pelatihan, mengurangi overfitting, dan membantu visualisasi data — sambil mempertahankan sebanyak mungkin informasi penting.

Konsep Utama

1. Curse of Dimensionality

Semakin banyak fitur (dimensi), semakin jarang data menyebar di ruang tersebut. Ini menyebabkan:

* Jarak antar titik menjadi kurang berarti
* Model kesulitan belajar pola
* Waktu pelatihan meningkat drastis

Reduksi dimensi membantu mengatasi masalah ini.

2. Projection

Salah satu cara utama mereduksi dimensi: memproyeksikan data ke dimensi yang lebih rendah.

Misalnya:

* Memproyeksikan data 3D ke bidang 2D
* Tujuannya: memaksimalkan variansi atau separasi antar kelas

3. Principal Component Analysis (PCA)

PCA adalah teknik yang sangat umum digunakan untuk reduksi dimensi:

* Menghitung sumbu baru (principal components) yang memaksimalkan variansi data
* Komponen utama adalah kombinasi linier dari fitur asli
* Tidak mempertahankan interpretasi fitur individual, tapi mempertahankan informasi penting

Contoh:

|  |
| --- |
| from sklearn.decomposition import PCA  pca = PCA(n\_components=2)  X\_reduced = pca.fit\_transform(X) |

bisa memilih:

Jumlah komponen eksplisit (n\_components=2)

Proporsi variansi yang ingin dipertahankan (n\_components=0.95)

4. PCA Whitening

Mengubah data sehingga semua komponen memiliki variansi = 1, dan tidak saling berkorelasi. Berguna dalam beberapa aplikasi ML.

5. Randomized PCA

Untuk dataset besar, Scikit-Learn menyediakan svd\_solver='randomized' untuk mempercepat perhitungan.

6. Incremental PCA

Gunakan IncrementalPCA saat data terlalu besar untuk dimuat sekaligus ke memori.

|  |
| --- |
| from sklearn.decomposition import IncrementalPCA  ipca = IncrementalPCA(n\_components=2)  for X\_batch in batches:  ipca.partial\_fit(X\_batch) |

7. Kernel PCA

Memungkinkan PCA bekerja dengan data nonlinier menggunakan kernel trick (mirip SVM).

|  |
| --- |
| from sklearn.decomposition import KernelPCA  kpca = KernelPCA(n\_components=2, kernel="rbf", gamma=0.04)  X\_kpca = kpca.fit\_transform(X) |

8. Locally Linear Embedding (LLE)

Teknik nonlinier lain. Menjaga hubungan lokal antar titik saat memproyeksikan ke dimensi rendah.

LLE digunakan untuk:

* Visualisasi data kompleks
* Reduksi dimensi data manifol (misalnya: gambar, suara)

|  |
| --- |
| from sklearn.manifold import LocallyLinearEmbedding  lle = LocallyLinearEmbedding(n\_components=2, n\_neighbors=10)  X\_reduced = lle.fit\_transform(X) |

Rangkuman proyek

Isi Utama:

* Visualisasi efek dari PCA, Kernel PCA, dan LLE
* Contoh reduksi dimensi pada data Swiss Roll
* Menunjukkan bahwa Kernel PCA bisa mengungkap struktur nonlinier

Evaluasi:

* Dibandingkan antara PCA linear vs nonlinear (Kernel PCA, LLE)
* Dicek seberapa baik metode bisa memisahkan kelas atau mengungkap pola

Inti Pelajaran

|  |  |
| --- | --- |
| Konsep | Penjelasan |
| PCA | Teknik linear untuk reduksi dimensi, mempertahankan variansi |
| Kernel PCA | Versi PCA untuk pola nonlinier |
| LLE | Teknik manifold learning untuk pola lokal |
| Incremental PCA | Untuk dataset besar, diproses batch per batch |
| Reduksi Dimensi | Mengurangi kompleksitas, mempercepat pelatihan, bantu visualisasi |