Laporan Teori - Chapter 8: Dimensionality Reduction

Pendahuluan

Banyak algoritma Machine Learning berkinerja buruk ketika dihadapkan pada data dengan jumlah fitur yang sangat besar (ratusan atau ribuan). Fenomena ini sering disebut **The Curse of Dimensionality** (Kutukan Dimensi). Ruang berdimensi tinggi sangatlah luas dan sebagian besar kosong, menyebabkan data pelatihan menjadi sangat jarang (*sparse*). Hal ini meningkatkan risiko *overfitting* dan membuat deteksi pola menjadi lebih sulit.

Dimensionality Reduction atau Reduksi Dimensi adalah proses mengurangi jumlah fitur (dimensi) dalam sebuah dataset. Tujuannya adalah untuk mengubah data dari ruang berdimensi tinggi ke ruang berdimensi rendah tanpa kehilangan banyak informasi penting. Manfaatnya antara lain:

- Mempercepat proses pelatihan algoritma.
- Menghemat ruang penyimpanan.
- Memudahkan visualisasi data (jika dikurangi menjadi 2D atau 3D).
- Terkadang dapat meningkatkan performa model dengan menghilangkan *noise* dan fitur yang tidak relevan.

Ada dua pendekatan utama untuk reduksi dimensi:

1. **Projection (Proyeksi)**: Pendekatan ini bekerja dengan baik ketika data terletak pada atau dekat dengan sebuah *subspace* (subruang) berdimensi lebih rendah. Data kemudian diproyeksikan ke subruang ini.
2. **Manifold Learning**: Pendekatan ini mengasumsikan bahwa data terletak pada sebuah *manifold* berdimensi rendah yang terdistorsi atau melengkung di dalam ruang berdimensi tinggi. Algoritma ini mencoba untuk "membuka" atau "meratakan" manifold tersebut.

Principal Component Analysis (PCA)

PCA adalah algoritma reduksi dimensi yang paling populer dan paling banyak digunakan. PCA bekerja dengan cara mengidentifikasi *hyperplane* yang paling dekat dengan data, kemudian memproyeksikan data ke *hyperplane* tersebut.

Konsep Utama: Mempertahankan Varians

PCA memilih sumbu proyeksi yang **mempertahankan jumlah varians semaksimal mungkin** dari data asli. Sumbu pertama, yang disebut **Principal Component pertama (PC1)**, adalah sumbu yang menangkap varians paling besar. Sumbu kedua (PC2) bersifat ortogonal (tegak lurus) terhadap PC1 dan menangkap varians sisa terbesar kedua, dan seterusnya.

Vektor satuan yang mendefinisikan arah setiap Principal Component dapat ditemukan menggunakan teknik dekomposisi matriks yang disebut **Singular Value Decomposition (SVD)**. SVD memecah matriks data pelatihan **X** menjadi perkalian tiga matriks. Matriks $\mathbf{V}^T$ dari dekomposisi ini berisi semua Principal Component.

Proses Proyeksi

Untuk memproyeksikan data ke *subspace* berdimensi *d* yang lebih rendah, kita mengalikan matriks data asli **X** dengan matriks **W**_d, yang berisi *d* Principal Component pertama.

- **Rumus Proyeksi: X**_d-proj = **X** · **W**_d

Memilih Jumlah Dimensi yang Tepat

Alih-alih memilih jumlah dimensi secara acak, kita dapat memilih jumlah dimensi yang mempertahankan sebagian besar varians total. **Explained Variance Ratio** dari setiap Principal Component menunjukkan proporsi varians dataset yang terletak di sepanjang sumbu tersebut. Kita bisa memilih jumlah dimensi d terkecil yang kumulatif *explained variance*-nya mencapai persentase tertentu, misalnya 95%.

Varian PCA

- **Randomized PCA**: Algoritma stokastik yang menemukan perkiraan untuk *d* Principal Component pertama dengan jauh lebih cepat.
- **Incremental PCA (IPCA)**: Berguna untuk dataset yang sangat besar yang tidak muat di memori. Algoritma ini membagi data menjadi *mini-batch* dan memprosesnya satu per satu.

Kernel PCA (kPCA)

PCA standar bekerja pada proyeksi linear. Untuk data yang memiliki struktur non-linear, PCA tidak akan efektif. **Kernel PCA (kPCA)** adalah solusi untuk masalah ini. Dengan menerapkan **kernel trick** (seperti yang digunakan pada SVM), kPCA memungkinkan kita untuk melakukan proyeksi non-linear yang kompleks untuk reduksi dimensi.

Secara konseptual, kPCA memetakan data ke ruang fitur berdimensi sangat tinggi (di mana data mungkin menjadi dapat dipisahkan secara linear), kemudian menggunakan PCA standar di ruang fitur tersebut untuk memproyeksikannya kembali ke ruang berdimensi lebih rendah. *Kernel trick* membuat proses ini efisien tanpa harus benar-benar melakukan pemetaan ke ruang fitur yang sangat tinggi.

Locally Linear Embedding (LLE)

LLE adalah teknik **Manifold Learning** non-linear yang tidak bergantung pada proyeksi. LLE bekerja dengan cara yang berbeda dari PCA:

1. **Mengidentifikasi Hubungan Lokal**: Pertama, untuk setiap instance pelatihan, LLE mengidentifikasi *k* tetangga terdekatnya. Kemudian, ia mencoba merekonstruksi setiap instance sebagai kombinasi linear dari para tetangganya.
2. **Mempertahankan Hubungan Lokal**: Selanjutnya, algoritma mencari representasi data dalam ruang berdimensi rendah di mana hubungan linear lokal ini dapat dipertahankan dengan sebaik mungkin.

LLE sangat efektif dalam "membuka gulungan" manifold yang terpelintir, seperti dataset *Swiss roll*, di mana jarak Euklides bukanlah ukuran yang baik untuk kedekatan antar titik.

Kesimpulan

Reduksi dimensi adalah langkah pra-pemrosesan yang sangat penting dalam pipeline Machine Learning. **PCA** adalah pilihan utama yang sangat baik untuk memulai, terutama untuk

mempercepat algoritma. Untuk data dengan struktur non-linear yang kompleks, **Kernel PCA** dan **LLE** menawarkan alternatif yang kuat. Pemilihan teknik yang tepat bergantung pada struktur data dan tujuan akhir dari model yang akan dibangun.