

Dimensionality Reduction

Ali Akbar Septiandri

Universitas Al-Azhar Indonesia

aliakbars@live.com

April 2, 2018

Selayang Pandang

- ① Ulasan
- ② Curse of Dimensionality
 - Contoh Kasus
 - Menangani Dimensi Tinggi
- ③ Principal Component Analysis
 - Pendahuluan
 - Principal Components
 - Nilai dan Vektor Eigen
 - Penggunaan

Bahan Bacaan

- ① VanderPlas, J. (2016). Python Data Science Handbook. (In Depth: Principal Component Analysis) <https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html>
- ② Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann. (Section 8.3)
- ③ Leskovec, J., Rajaraman, A., & Ullman, J. D. (2014). *Mining of massive datasets*. Cambridge University Press. (Section 11.1-11.2)

Ulasan

Minggu lalu...

- Naïve Bayes
- Conditional independence
- Penggunaan distribusi Gaussian dan Bernoulli untuk NB
- Diagonal dan full covariance matrix saat klasifikasi

Bagaimana representasi Naïve Bayes
untuk multivariate Gaussian?

Curse of Dimensionality

Curse of Dimensionality

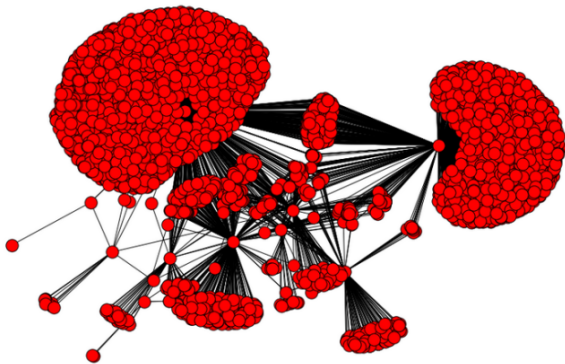
- Dataset yang kita punya biasanya memiliki dimensi yang tinggi, e.g. gambar, suara, teks
- Dimensi yang “penting” mungkin jauh lebih kecil
- Ada atribut yang variansnya kecil

Contoh Kasus: MNIST



Gambar: Hanya sebagian dari semua pixel yang berubah nilainya pada data MNIST dari \mathbb{R}^{784}

Contoh Kasus: Social Network



Gambar: Tidak semua orang terhubung dalam jejaring sosial

Contoh kasus apa lagi yang dapat kalian bayangkan?

Bagaimana cara menanganinya?

Penanganan

- Gunakan pengetahuan terhadap domain tersebut, e.g. MFCC pada data suara
- Berasumsi terhadap dimensinya, e.g. independensi pada Naïve Bayes
- Mereduksi dimensinya, buat dimensi baru!

Reduksi Dimensi

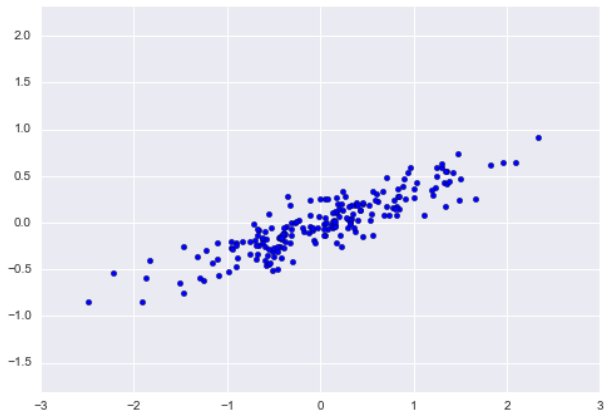
- Tujuannya adalah merepresentasikan data dengan variabel yang lebih sedikit
- Memilih fitur, misalnya dengan *information gain*
- Ekstraksi fitur, misalnya IMT dari berat dan tinggi badan, atau dengan **kombinasi linear**

Principal Component Analysis

Principal Components

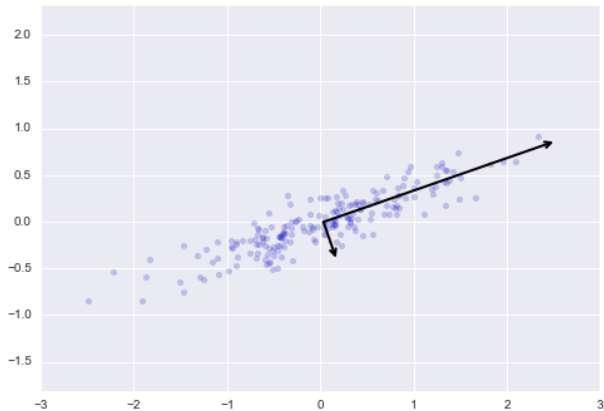
- Pencarian *principal components* dilakukan dengan mencari arah keragaman data terbesar secara berurut
- Setiap *principal components* tersebut bersifat tegak lurus satu dengan yang lain
- Terus dilakukan hingga d dimensi original

Pencarian PC



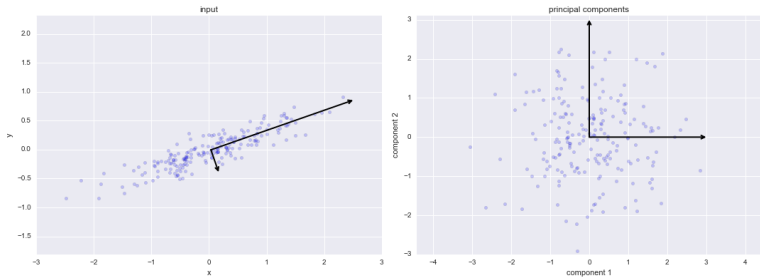
Gambar: Data dalam dua dimensi [VanderPlas, 2016]

Pencarian PC



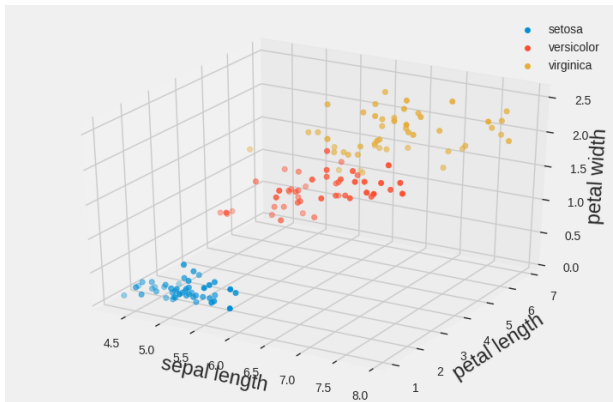
Gambar: Principal components dari data [VanderPlas, 2016]

Pencarian PC



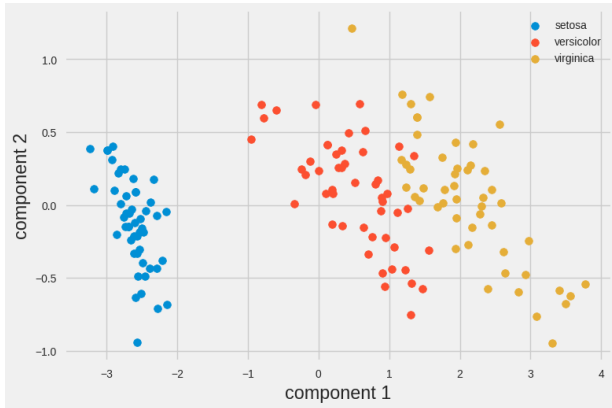
Gambar: Proyeksi data menggunakan principal components
[VanderPlas, 2016]

Contoh: PCA pada Iris Dataset



Gambar: Dataset Iris dalam tiga dimensi

Contoh: PCA pada Iris Dataset



Gambar: Dataset Iris setelah diproyeksi dengan PCA

Mencari Principal Components

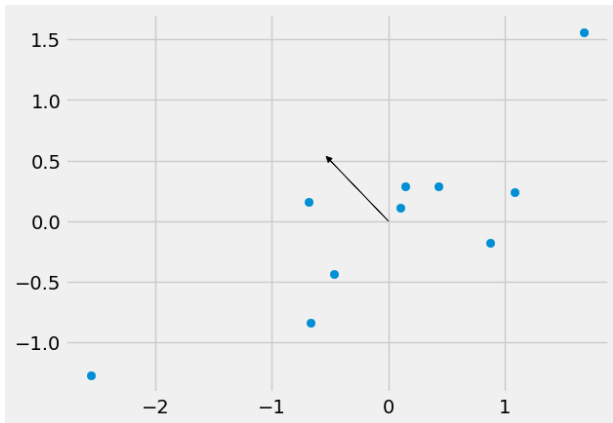
- 1 Pusatkan data ke titik nol: $x_{i,a} \leftarrow x_{i,a} - \mu$
- 2 Hitung matriks kovarian Σ
- 3 Cari vektor eigen \mathbf{e} untuk matriks tersebut!

Principal Components

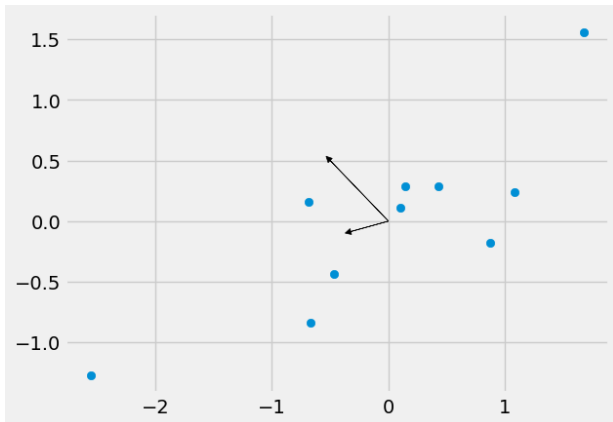
Sebagai ilustrasi, menggunakan matriks kovarian tersebut:

- 1 Pilih satu vektor secara acak
- 2 Kalikan dengan matriks kovarian – apa yang terjadi?

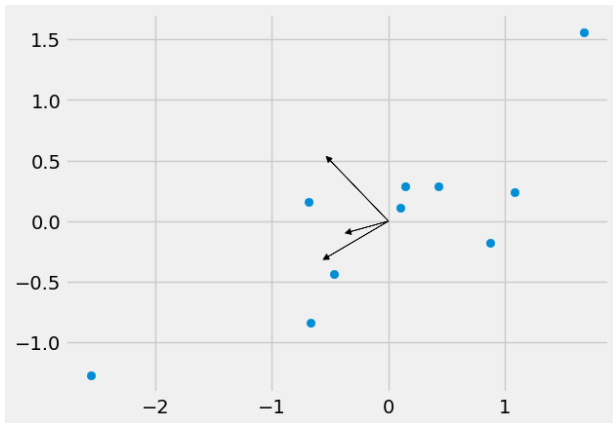
Contoh: Principal Components



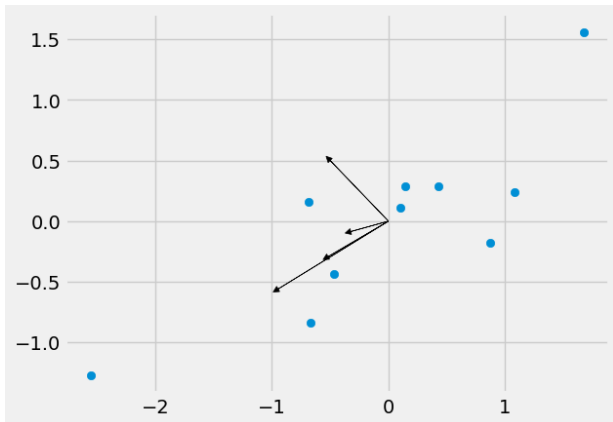
Contoh: Principal Components



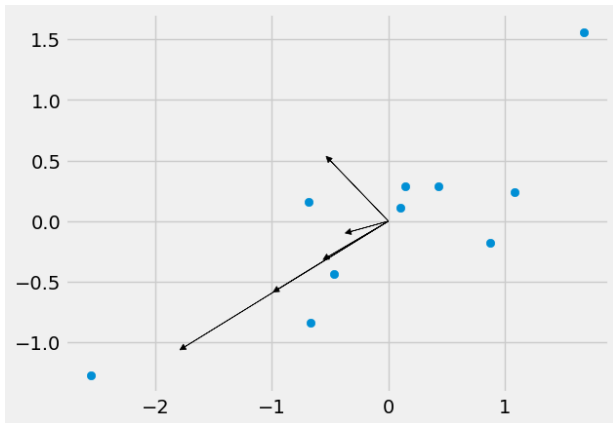
Contoh: Principal Components



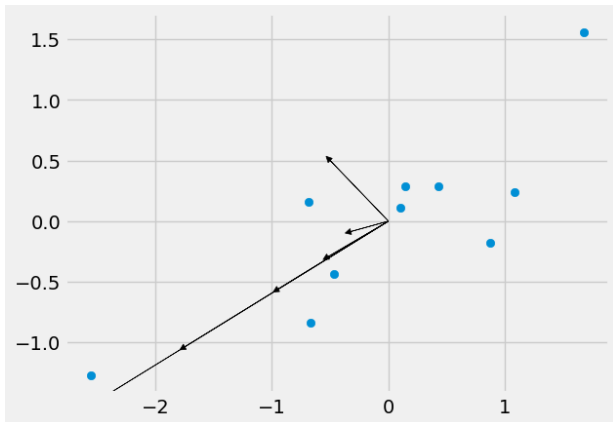
Contoh: Principal Components



Contoh: Principal Components



Contoh: Principal Components



Arah dari vektornya tidak berubah lagi!

Principal Components

- Kita ingin vektor \mathbf{e} yang arahnya tidak berubah lagi, $\Sigma \mathbf{e} = \lambda \mathbf{e}$
- \mathbf{e} ... vektor eigen dari Σ , λ ... nilai eigen untuk vektor tersebut
- **Principal components** = vektor eigen dengan nilai eigen terbesar

Aljabar Matriks

Beberapa pemahaman yang dibutuhkan untuk materi ini:

- perkalian
- transpose
- determinan
- solusi persamaan linear

Definisi

Nilai dan Vektor Eigen

M adalah matriks bujur sangkar. λ adalah konstanta dan \mathbf{e} adalah vektor kolom tak-nol dengan jumlah baris seperti M . Maka, λ adalah *nilai eigen* (*eigenvalue*) dari M dan \mathbf{e} adalah *vektor eigen* (*eigenvector*) dari M jika $M\mathbf{e} = \lambda\mathbf{e}$.

Mencari Principal Components

- 1 Cari nilai eigen dengan menyelesaikan persamaan $\det(\Sigma - \lambda I) = 0$
- 2 Cari vektor eigen ke- i dengan menyelesaikan persamaan $\Sigma \mathbf{e}_i = \lambda_i \mathbf{e}_i$
- 3 Principal components secara berurut adalah vektor eigen dengan nilai eigen terbesar

Contoh

Diberikan matriks kovarian $\Sigma = \begin{bmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{bmatrix}$, nilai eigen dapat dicari dengan

$$\det \begin{bmatrix} 2.0 - \lambda & 0.8 \\ 0.8 & 0.6 - \lambda \end{bmatrix} = (2 - \lambda)(0.6 - \lambda) - (0.8)(0.8) = 0$$
$$\lambda^2 - 2.6\lambda + 0.56 = 0$$

Nilai eigen yang didapatkan

$$\{\lambda_1, \lambda_2\} = \frac{1}{2}(2.6 \pm \sqrt{2.6^2 - 4 \times 0.56}) = \{2.36, 0.23\}$$

Contoh (lanjutan)

Vektor eigen untuk masing-masing nilai eigen dapat dicari dengan

$$\begin{aligned} \begin{bmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{bmatrix} \begin{bmatrix} e_{1,1} \\ e_{1,2} \end{bmatrix} &= 2.36 \begin{bmatrix} e_{1,1} \\ e_{1,2} \end{bmatrix} \Leftrightarrow \begin{aligned} 2.0e_{1,1} + 0.8e_{1,2} &= 2.36e_{1,1} \\ 0.8e_{1,1} + 0.6e_{1,2} &= 2.36e_{1,2} \end{aligned} \\ &\Leftrightarrow e_{1,1} = 2.2e_{1,2} \\ &\Leftrightarrow e_1 \sim \begin{bmatrix} 2.2 \\ 1 \end{bmatrix} \\ &\Leftrightarrow e_1 = \begin{bmatrix} 0.91 \\ 0.41 \end{bmatrix} \text{ (vektor unit)} \end{aligned}$$

Dengan cara yang sama

$$\begin{bmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{bmatrix} \begin{bmatrix} e_{2,1} \\ e_{2,2} \end{bmatrix} = 0.23 \begin{bmatrix} e_{2,1} \\ e_{2,2} \end{bmatrix} \Leftrightarrow e_2 = \begin{bmatrix} -0.41 \\ 0.91 \end{bmatrix}$$

Proyeksi ke Dimensi Baru

- $\mathbf{e}_1 \dots \mathbf{e}_m$ adalah vektor dimensi baru
- Untuk setiap titik data \mathbf{x}_i :
 - ① Pusatkan terhadap rata-rata, i.e. $\mathbf{x}_i - \mu$
 - ② Proyeksikan ke dimensi baru, i.e. $(\mathbf{x}_i - \mu)^T \mathbf{e}_j$ untuk $j = 1 \dots m$

$$\begin{bmatrix} x'_{i,1} \\ x'_{i,2} \\ \vdots \\ x'_{i,m} \end{bmatrix} = \begin{bmatrix} (\mathbf{x}_i - \mu)^T \mathbf{e}_1 \\ (\mathbf{x}_i - \mu)^T \mathbf{e}_2 \\ \vdots \\ (\mathbf{x}_i - \mu)^T \mathbf{e}_m \end{bmatrix}$$

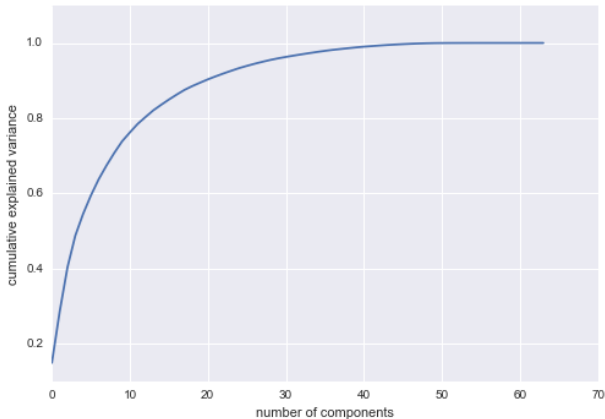
Berapa Dimensi?

- Dari vektor eigen $\mathbf{e}_1 \dots \mathbf{e}_d$, ingin dihasilkan $m \ll d$
- Pilih \mathbf{e}_i yang “menjelaskan” varians sebanyak mungkin
 - ① Urutkan vektor eigen berdasarkan $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d$
 - ② Pilih m vektor eigen pertama yang menjelaskan 90% atau 95% varians

$$\frac{\sum_{i=1}^m \lambda_i}{\sum_{i=1}^d \lambda_i} \leq 1$$

- Atau gunakan scree plot

Kurva Varians



Gambar: Kurva varians dari data *hand-written digits* [VanderPlas, 2016]

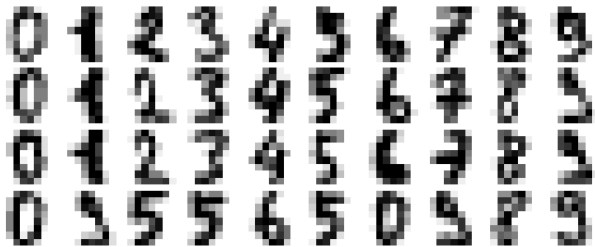
Penggunaan PCA

Seperti yang telah dibahas, beberapa kegunaan PCA adalah:

- reduksi dimensi
- visualisasi
- *noise filtering*

Noise Filtering dengan PCA

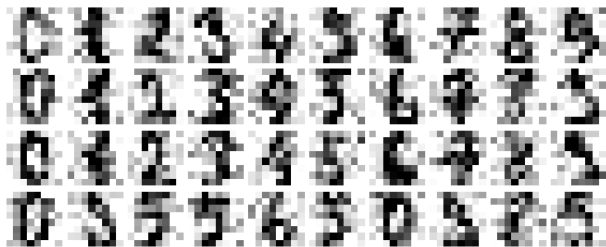
Idenya adalah komponen dengan varians yang jauh lebih tinggi daripada *noise* tidak akan terkena dampak dari *noise*.



Gambar: Proses *noise filtering* dengan PCA [VanderPlas, 2016]

Noise Filtering dengan PCA

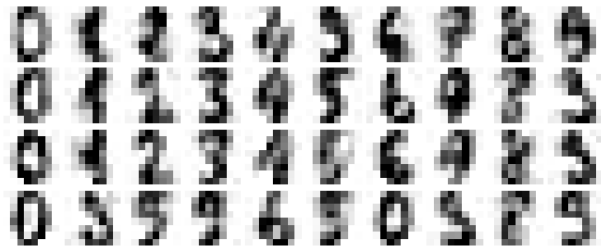
Idenya adalah komponen dengan varians yang jauh lebih tinggi daripada *noise* tidak akan terkena dampak dari *noise*.



Gambar: Proses *noise filtering* dengan PCA [VanderPlas, 2016]

Noise Filtering dengan PCA

Idenya adalah komponen dengan varians yang jauh lebih tinggi daripada *noise* tidak akan terkena dampak dari *noise*.



Gambar: Proses *noise filtering* dengan PCA [VanderPlas, 2016]

Contoh: Eigenfaces

Ide yang sama dapat digunakan untuk merepresentasikan wajah seseorang.



Gambar: Principal components dari wajah tokoh dunia [VanderPlas, 2016]

Masalah dalam PCA

- Dapat sangat dipengaruhi oleh pencilan (dalam perhitungan matriks kovarians),
→ bisa diatasi dengan normalisasi (membagi dengan simpangan baku)
- Asumsi linearitas dalam data,
→ bisa diatasi dengan transformasi

Variasi PCA

Karena beberapa batasan (termasuk yang tidak disebutkan sebelumnya), terdapat beberapa variasi untuk pengembangan PCA, antara lain:

- Linear Discriminant Analysis
- Probabilistic PCA
- Truncated Singular-Value Decomposition (SVD)
- CUR-decomposition (Leskovec, et al., 2014)

Dimensionality Reduction

Pros:

- Mewakili intuisi kita terhadap data
- Hasil estimasi probabilistik yang lebih baik
- Reduksi data \rightarrow proses lebih cepat

Cons:

- Mahal secara komputasi
- Asumsi linearitas membuatnya sulit menangani kasus khusus, e.g. pencilan

Manifold Learning (non-examinable)

Bacaan lebih lanjut:

- **t-Stochastic Neighbor Embedding (t-SNE)**
[van der Maaten, 2008, Wattenberg, 2016]
- **Uniform Manifold Approximation and Projection (UMAP)** [McInnes, 2018]

- Dimensionality Reduction
- Eigenvector & Eigenvalue
- Principal Component Analysis

Pertemuan Berikutnya

- Linear regression
- Logistic regression
- Metode optimasi

Office hours minggu ini di
hari Rabu, 08.00-09.00

Referensi



Jake VanderPlas (2016)

In Depth: Principal Component Analysis

<https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html>



L.J.P. van der Maaten and G.E. Hinton. (2008)

Visualizing High-Dimensional Data Using t-SNE

Journal of Machine Learning Research 9(Nov):2579-2605



Wattenberg, et al. (2016)

"How to Use t-SNE Effectively"

Distill <http://doi.org/10.23915/distill.00002>



L. McInnes and J. Healy (2018)

UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction

arXiv preprint [arXiv:1802.03426](https://arxiv.org/abs/1802.03426)

Terima kasih