

Reduksi Dimensi: PCA

**Dina Chahyati*, Adila Alfa Krisnadhi, Siti Aminah,
Aruni Yasmin Azizah, Fariz Darari**

**CSGE603130: Kecerdasan Artifisial dan Sains Data Dasar
Gasal 2022/2023**

Outline

1. Review Aljabar Linier & Statistik

- a) Matriks Kovarian
- b) Basis, Dimensi & Koordinat Relatif
- c) Transformasi Linier
- d) Nilai & Vektor Eigen
- e) Diagonalisasi Matriks

2. Principle Component Analysis (PCA)



FAKULTAS
ILMU
KOMPUTER

Review

Review: Matriks Kovarian

- Matriks Kovarian merupakan matriks yang berisi nilai kovarian antara variabel-variabel atau fitur-fitur yang ada.
- Misal suatu data memiliki 3 kolom sebagai berikut:

ID	Berat (B)	Tinggi (T)	Usia (U)
1	50	160	20
2	40	155	18
3	55	160	19

- Maka matriks kovarian untuk data tersebut adalah:

$$\begin{bmatrix} cov(B, B) = var(B) & cov(B, T) & cov(B, U) \\ cov(T, B) & var(T) & cov(T, U) \\ cov(U, B) & cov(U, T) & var(U) \end{bmatrix}$$

- Rumus Kovarian: $cov(B, T) = \frac{\sum(b - \bar{b})(t - \bar{t})}{N}$, \bar{b} menyatakan rerata dari b

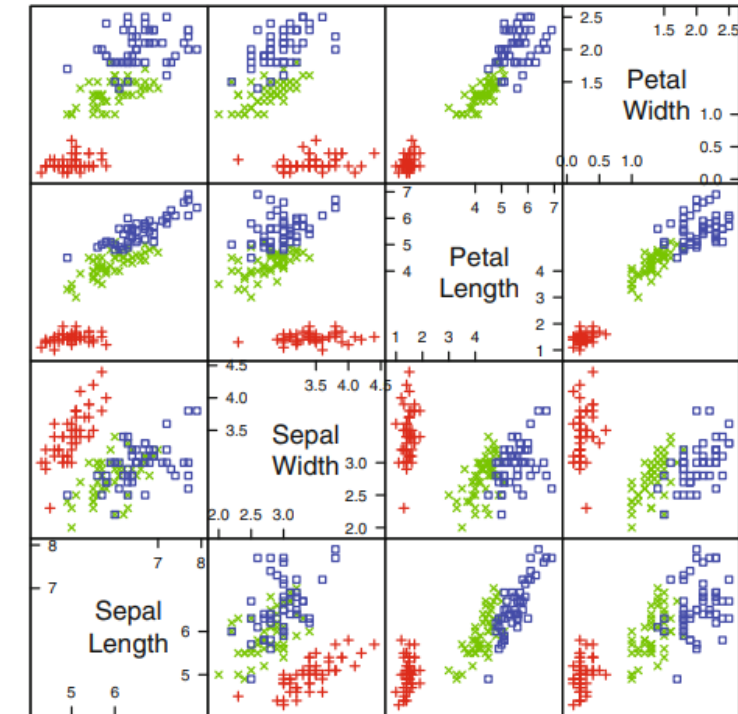
Review: Kovarian

- Kovarian adalah **ukuran bagaimana perubahan dalam satu variabel** dikaitkan dengan perubahan dalam variabel kedua.
- Secara khusus, kovarian mengukur sejauh mana dua variabel terkait secara linear
- Apa perbedaan kovarian dan korelasi?
 - Korelasi adalah kovarian yang sudah dinormalisasi sehingga berada pada rentang nilai -1 sampai 1

Covariance Matrix of Iris Dataset

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
sepal length (cm)	0.685694	-0.039268	1.273682	0.516904
sepal width (cm)	-0.039268	0.188004	-0.321713	-0.117981
petal length (cm)	1.273682	-0.321713	3.113179	1.296387
petal width (cm)	0.516904	-0.117981	1.296387	0.582414

Covariance Matrix dari Iris Dataset (<https://medium.com/@kyasar.mail/pca-principal-component-analysis-729068e28ec8>)



Scatter Plot Matrix

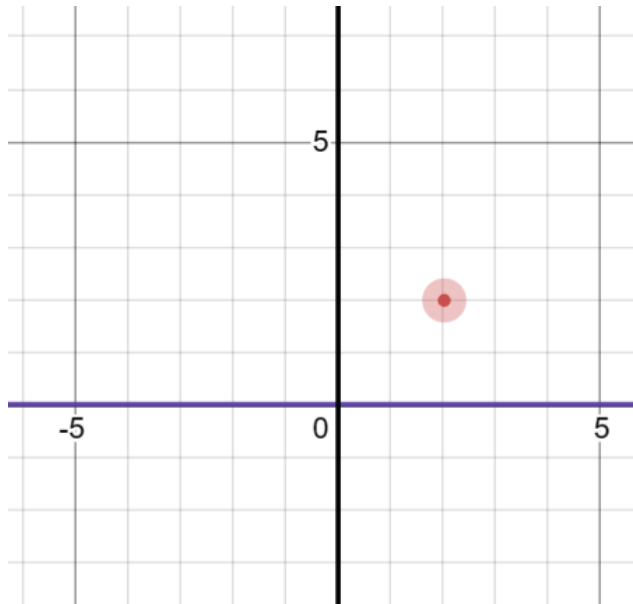
- Pasangan atribut/komponen mana yang memiliki covariance paling tinggi / paling rendah? Apakah sesuai dengan scatter plot nya?

Review: Basis & Dimensi

- Pengertian dalam Ruang Vektor:
 - Diberikan ruang vektor V dan B subhimpunan vektor-vektor dari ruang vektor V , maka B adalah basis, jika dua kondisi berikut terpenuhi:
 1. B bebas linier
 2. B merentang V
 - Dimensi ruang vektor V adalah jumlah vektor dalam basis V
- Contoh:
 - Basis R^3 : $A = \{(0, 0, 1), (0, 1, 0), (1, 0, 0)\}$
 - $\dim(R^3) = 3$
- Pada konteks sains data, biasanya **dimensi** menyatakan jumlah kolom atau fitur/atribut pada data.

Review: Basis & Koordinat Relatif

- Basis mempengaruhi **koordinat relatif** dari suatu vektor.

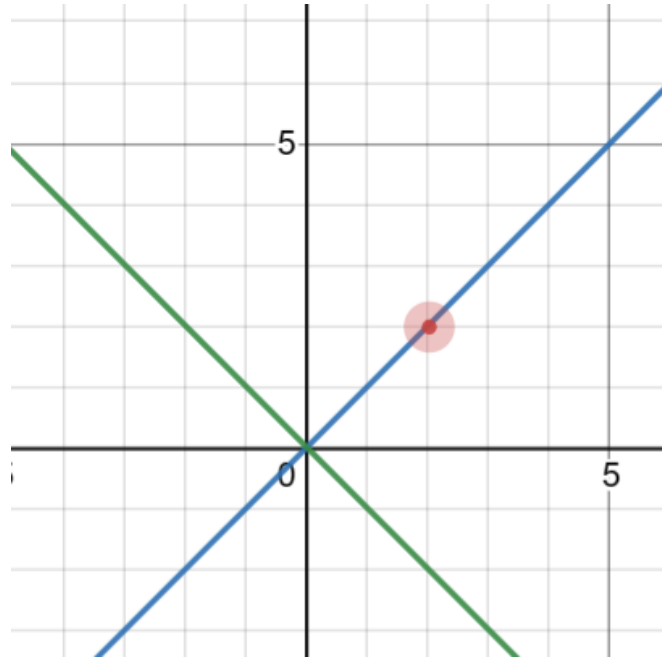


Basis B1 : $\{(1, 0), (0, 1)\}$

Koordinat titik merah relatif

terhadap B1 = $(2, 2)$

Karena $(2, 2) = 2(1, 0) + 2(0, 1)$

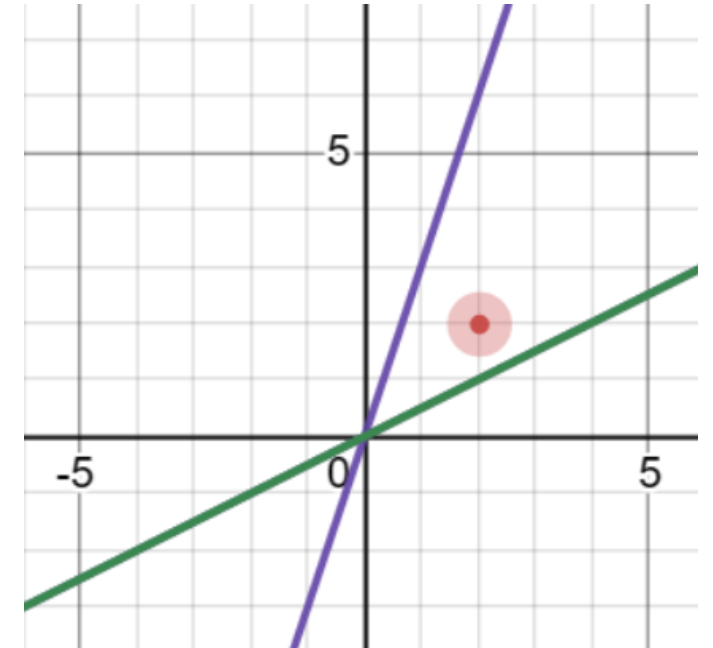


Basis B2 : $\{(1, 1), (1, -1)\}$

Koordinat titik merah

relatif terhadap B2 = $(2, 0)$

Karena $(2, 2) = 2(1, 1) + 0(1, -1)$



Basis B3 : $\{(1, 3), (2, 1)\}$

Koordinat titik merah

relatif terhadap B3 = (\dots, \dots) ?

Karena $(2, 2) = \dots(1, 3) + \dots(2, 1)$

- Dalam konteks sains data, koordinat relatif ini kadang disebut *feature map*.

Review: Transformasi Linier

- Pada contoh sebelumnya, jika kita ingin mengubah koordinat relatif dari titik/vektor (2,2) dari basis $B1 = \{(1,0), (0,1)\}$ ke $B2 = \{(1,1), (1,-1)\}$, salah satu caranya adalah:
 - Cari solusi dari persamaan $(2,2) = a(1,1) + b(1,-1)$
 - Atau selesaikan Sistem Persamaan Linier: $2 = a + b$ dan $2 = a - b$

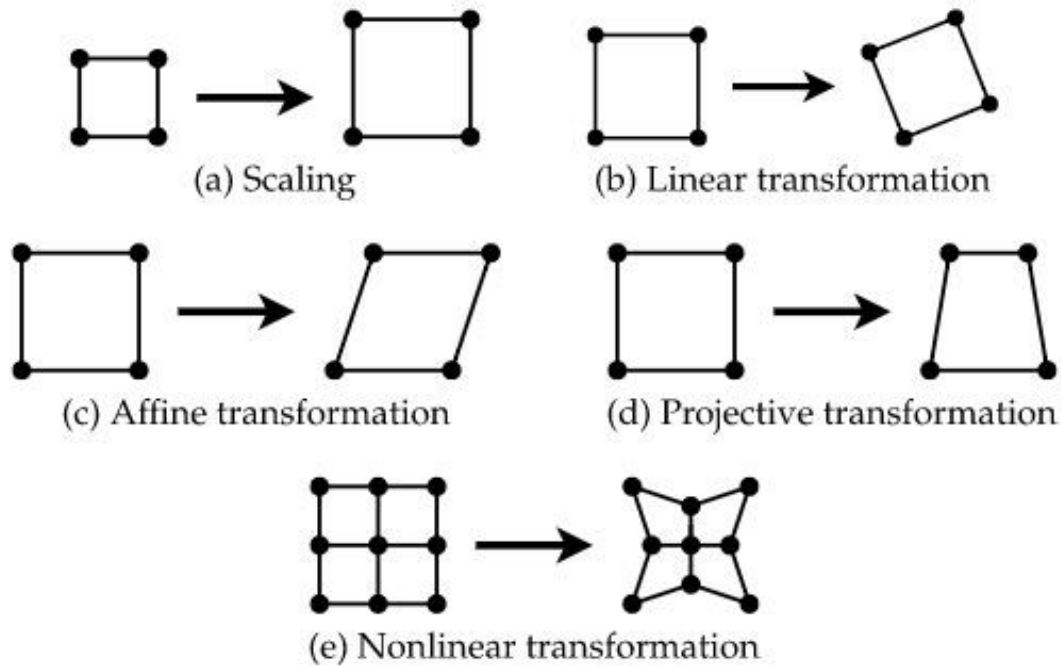
$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \text{ atau } \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}^{-1} \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & -0.5 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

Review: Transformasi Linier

- Jika $T: V \rightarrow W$ adalah fungsi dari ruang vektor V ke ruang vektor W , maka T disebut **transformasi linear** jika untuk setiap vektor \mathbf{u}, \mathbf{v} di V dan setiap skalar k berlaku:
 - $T(\mathbf{u} + \mathbf{v}) = T(\mathbf{u}) + T(\mathbf{v})$
 - $T(k\mathbf{u}) = k T(\mathbf{u})$
- Transformasi linier di ruang Euclid dapat juga dianggap sebagai perkalian matriks dimana $[T]$ menyatakan matriks transformasi, sehingga $T(\mathbf{u}) = [T]\mathbf{u}$
- Perubahan basis dapat dianggap sebagai sebuah transformasi linier.
- Pada contoh sebelumnya, perubahan basis dari $B1$ (**standar**) ke $B2 = \{(1,1), (1, -1)\}$ dapat dinyatakan sebagai
 - $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & -0.5 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$ atau $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}^{-1} \begin{bmatrix} x \\ y \end{bmatrix}$
 - Cara mudah mencari $[T]$: gabungkan vektor-vektor basis column-wise menjadi matriks, lalu cari inversnya.

Review: Transformasi

- Transformasi tidak selalu linier.
- Contoh transformasi:



Source: 'Accelerating nonlinear image transformations with OpenGL' -Vegard Øye

Review: Nilai & Vektor Eigen (NVE)

Vektor tak nol \mathbf{v} di R^n disebut **vektor eigen** dari $A_{n \times n}$ jika terdapat skalar sedemikian hingga $A\mathbf{v} = \lambda\mathbf{v}$,

- λ disebut **nilai eigen**
- \mathbf{v} adalah vektor eigen dari A yang bersesuaian dengan λ .

Hal menarik tentang nilai dan vektor eigen dari suatu matriks A ukuran $n \times n$:

1. NVE hanya terdefinisi untuk matriks persegi
2. Matriks persegi berukuran $n \times n$ maksimal memiliki n nilai eigen
3. Jika \mathbf{v}_1 adalah vektor eigen yang bersesuaian dengan nilai eigen λ_1 dan \mathbf{v}_2 adalah vektor eigen yang bersesuaian dengan nilai eigen λ_2 , maka \mathbf{v}_1 dan \mathbf{v}_2 **bebas linier**
4. (meneruskan nomor 3), jika A adalah matriks simetri, maka \mathbf{v}_1 dan \mathbf{v}_2 **orthogonal**

Review: Nilai & Vektor Eigen (NVE)

- Jika dikaitkan dengan transformasi linier, vektor eigen dapat dipandang sebagai vektor-vektor yang setelah ditransformasi, **arahnya tidak berubah**
- Pada contoh sebelumnya, perubahan basis dari $B1 = \{(1,0), (0,1)\}$ ke $B2 = \{(1,1), (1, -1)\}$ dapat dinyatakan sebagai transformasi linier
 - $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & -0.5 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$
 - Nilai eigen dari matriks transformasi $\begin{bmatrix} 0.5 & 0.5 \\ 0.5 & -0.5 \end{bmatrix} : \lambda_1 = -\frac{1}{\sqrt{2}} = -0.707, \quad \lambda_2 = \frac{1}{\sqrt{2}}$
 - Vektor eigen: $\mathbf{v}_1 = (1 - \sqrt{2}, 1) = (-0.414, 1), \quad \mathbf{v}_2 = (1 + \sqrt{2}, 1)$
 - Vektor (2,2) ditransformasi menjadi (2,0) \rightarrow berubah arah
 - Vektor (-0.414, 1) ditransformasi menjadi (0.293, -0.707) \rightarrow tidak berubah arah

Review Diagonalisasi Matriks

Definisi: Matriks persegi A dapat didiagonalkan jika terdapat matriks P yang mempunyai inverse sedemikian hingga

$$P^{-1}AP = D, \text{ dimana } D \text{ adalah matriks diagonal}$$

- $P^{-1}AP = D$ berarti $PP^{-1}AP = PD$ atau $AP = PD$ atau $PD = AP$
- Slide berikut ini akan menunjukkan bahwa
 - “Mencari matriks P yang mendiagonalkan A ”
sama dengan
 - “Mencari vektor-vektor eigen (bebas linier) dari matriks A ”

Review: Diagonalisasi Matriks

$$PD = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix} = \begin{pmatrix} \lambda_1 p_{11} & \lambda_2 p_{12} & \cdots & \lambda_n p_{1n} \\ \lambda_1 p_{21} & \lambda_2 p_{22} & \cdots & \lambda_n p_{2n} \\ \vdots & \vdots & & \vdots \\ \lambda_1 p_{n1} & \lambda_2 p_{n2} & \cdots & \lambda_n p_{nn} \end{pmatrix} = (\lambda_1 \mathbf{p}_1 \quad \lambda_2 \mathbf{p}_2 \quad \cdots \quad \lambda_n \mathbf{p}_n)$$

$$P^{-1}AP = D$$

$$PD = AP$$

$$AP = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{pmatrix} = (A\mathbf{p}_1 \quad A\mathbf{p}_2 \quad \cdots \quad A\mathbf{p}_n)$$

$$A\mathbf{p}_1 = \lambda_1 \mathbf{p}_1 \quad A\mathbf{p}_2 = \lambda_2 \mathbf{p}_2 \quad \cdots \quad A\mathbf{p}_n = \lambda_n \mathbf{p}_n$$

$$A\mathbf{v} = \lambda \mathbf{v}$$

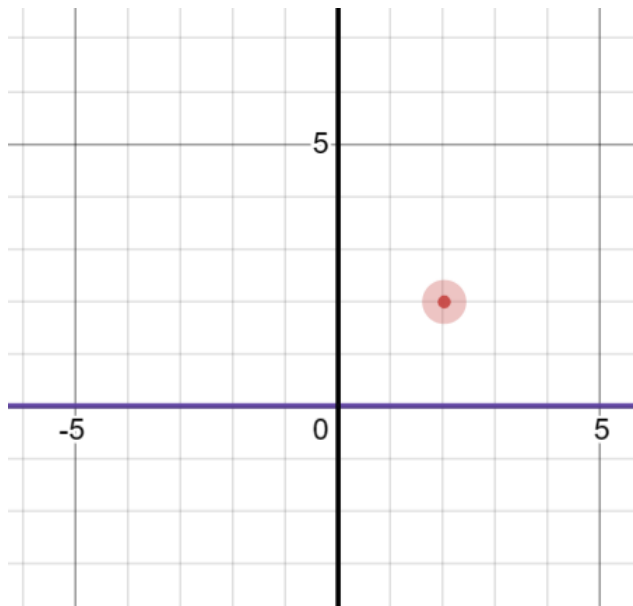
P mempunyai inverse, maka kolom-kolomnya bukan kolom nol. Berdasarkan definisi nilai eigen, maka $\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_n$ merupakan nilai-nilai eigen A , kolom-kolom P bebas linear (karena P^{-1} ada) adalah vektor-vektor eigen A . Sebaliknya juga berlaku.

Review: Transformasi

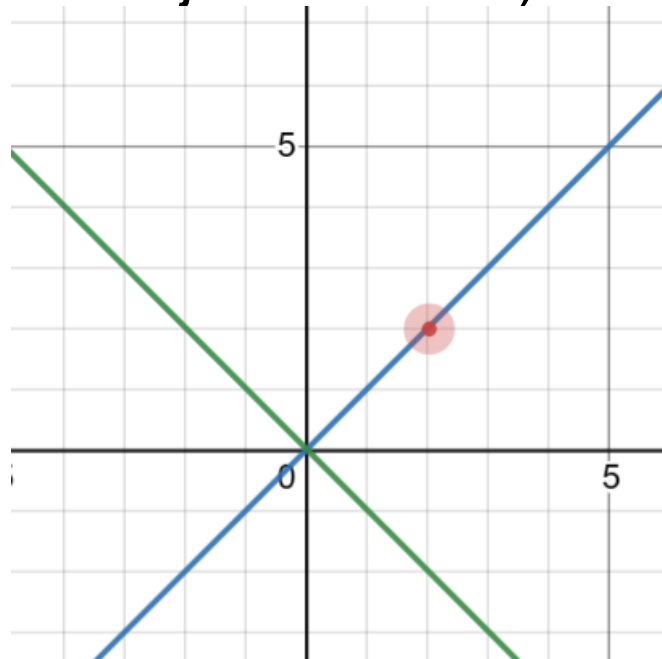
- Salah satu persoalan pada *Machine Learning* adalah bagaimana **mengklasifikasi** data ke kelas-kelas tertentu.
- Klasifikasi kadang mudah dilakukan di suatu **ruang**, tapi sulit dilakukan di **ruang** lainnya (ingat hubungan **basis** dan **ruang**).
- Perlu pengetahuan untuk memilih pada **ruang apa** sebaiknya suatu data direpresentasikan agar tujuan bisa tercapai.
- Cara mengubah data dari satu ruang ke ruang lain: **transformasi** (tidak harus linier)

Review: Transformasi

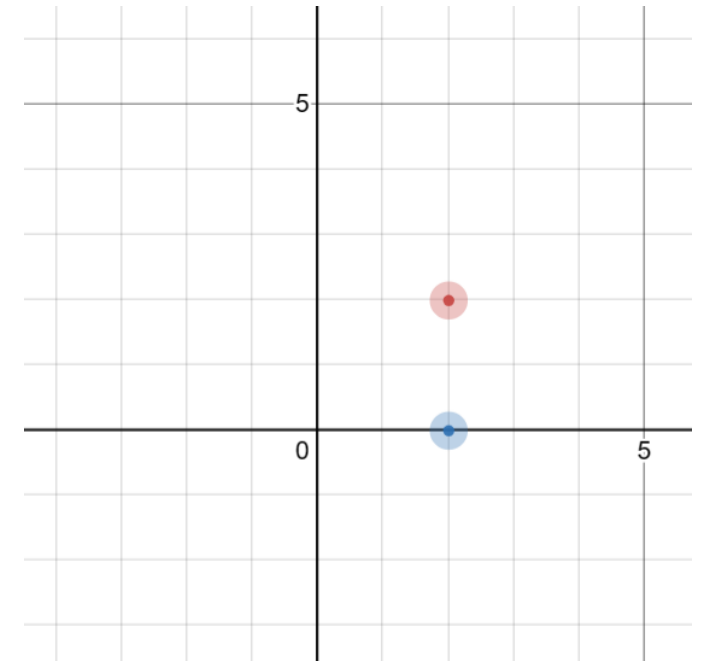
- Gambar kiri dan tengah menunjukkan ilustrasi ketika posisi titik merah tetap, namun sumbu koordinat yang diputar
- Gambar kanan menunjukkan ilustrasi ketika basis B1 dan B2 dibuat berhimpit (titik merah bertransformasi menjadi titik biru)



Basis B1 : $\{(1, 0), (0, 1)\}$
Koordinat titik merah relatif
terhadap B1 = $(2, 2)$

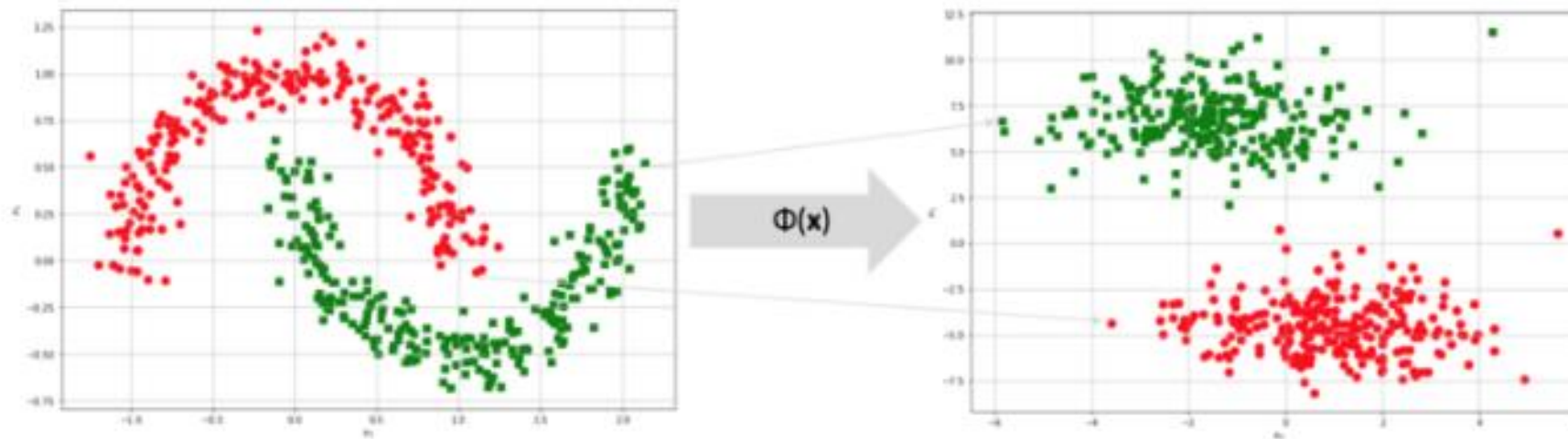


Basis B2 : $\{(1, 1), (1, -1)\}$
Koordinat titik merah
relatif terhadap B2 = $(2, 0)$



Terjadi transformasi dari titik
 $(2, 2)$ menjadi $(2, 0)$

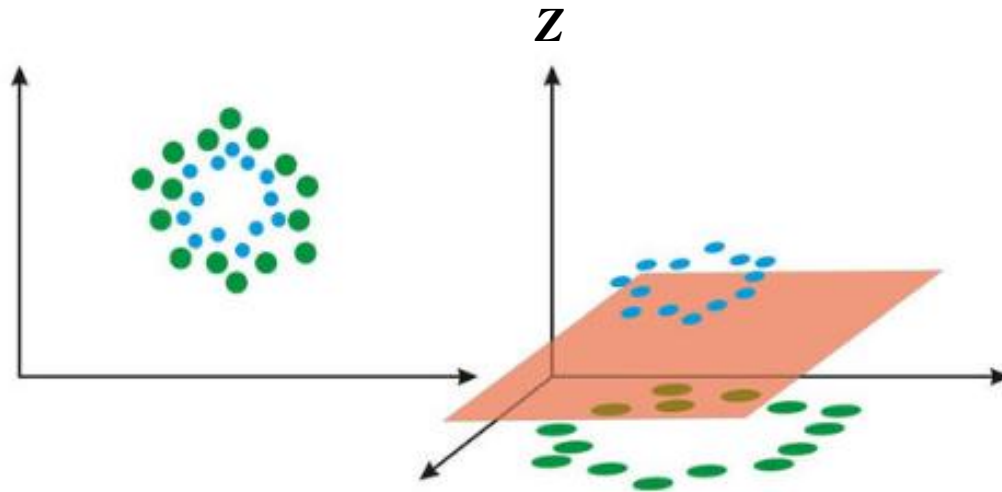
Review: Transformasi



- Data di sebelah kiri sulit diklasifikasikan oleh komputer, tapi data di sebelah kanan mudah (suatu objek termasuk kelas merah kalau nilai kompenen y -nya kurang dari c , misalnya)
- Data perlu ditransformasikan dari ruang A (kiri) ke ruang B (kanan).
- Pada contoh ini **dimensi** ruangan A dan B sama, tapi **basis**nya berbeda.

Gambar diperoleh dari : <https://www.bonaccorso.eu/2017/09/28/linearly-separable-no-for-me-it-is-a-brief-introduction-to-kernel-methods/>

Review: Transformasi



Gambar diambil dari :

https://www.researchgate.net/figure/a-Linear-separable-data-with-good-margins-b-Linearly-impossible-to-separate-data-2_fig1_334332328

- Data di sebelah kiri sulit diklasifikasikan oleh komputer, tapi data di sebelah kanan mudah (suatu objek termasuk kelas hijau kalau nilai koefisien z -nya kurang dari c , misalnya)
- Data perlu ditransformasikan dari ruang A (kiri) ke ruang B (kanan).
- Pada contoh ini **dimensi** ruangan A dan B berbeda. Terkadang suatu data perlu ditransformasikan ke ruang yang berdimensi **lebih tinggi** agar mudah diklasifikasikan.
- Pada persoalan lain dimana suatu data memiliki ratusan dimensi, persoalannya justru adalah bagaimana mengubahnya ke dimensi yang **lebih rendah** agar lebih mudah diproses tapi informasi yang hilang tidak terlalu banyak (seperti contoh pengenalan wajah).

Principle Component Analysis (PCA)

Sumber:

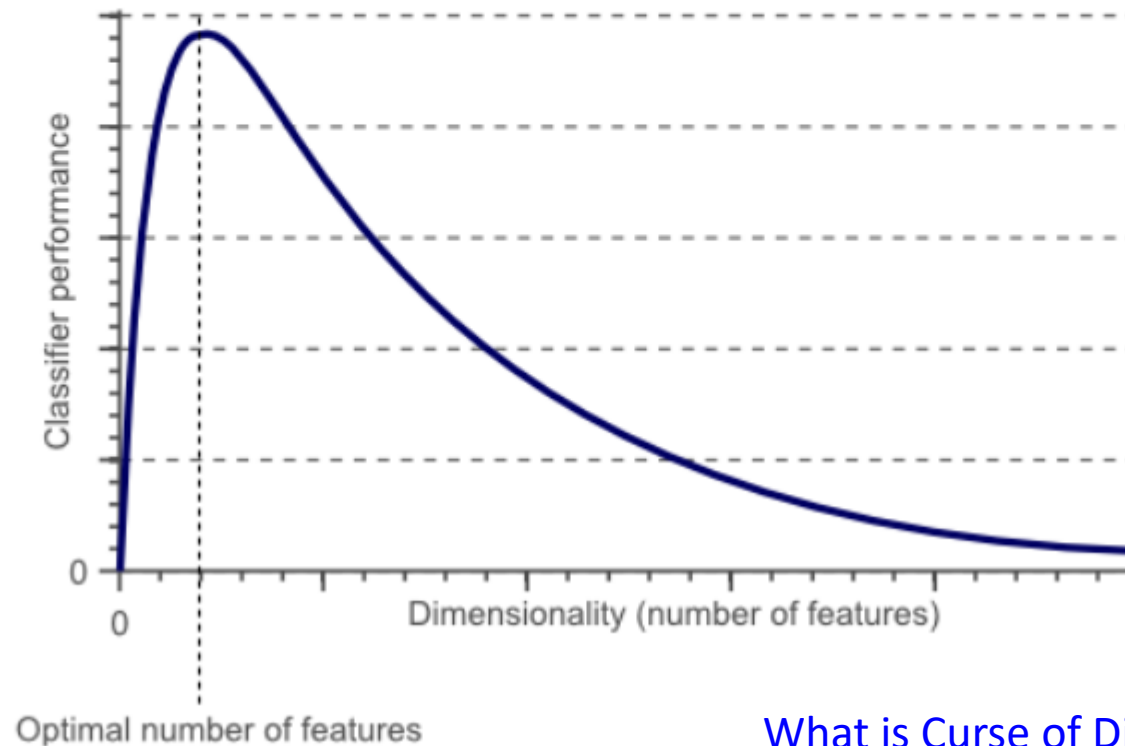
- Lindsay. I. Smith, A Tutorial on Principal Components Analysis, 2002
- David Forsyth, Probability and Statistics for Computer Science, 2018
- [A Step-by-Step Explanation of Principal Component Analysis \(PCA\) | Built In](#)
- [PCA Explained Variance Concepts with Python Example - Data Analytics \(vitalflux.com\)](#)
- [Faces recognition example using eigenfaces and SVMs — scikit-learn 1.1.2 documentation](#)

Pendahuluan

- Misalkan kita memiliki data dengan 100 kolom/fitur. Tanpa menghilangkan informasi penting, bagaimana cara mencapai tujuan (misal: klasifikasi), tanpa harus memproses keseluruhan fitur?
- Bisa dicoba untuk **mereduksi dimensi**, dengan cara:
 - Ambil hanya 10 dari 100 fitur tanpa melakukan transformasi fitur, dilakukan dengan cara coba-coba. Contoh: Sequential Floating Forward Selection (SFFS)
 - Transformasi data yang tadinya ada di ruang 100 dimensi, ke ruang dengan dimensi yang lebih kecil. Contoh: PCA
- Keuntungan reduksi dimensi: storage lebih sedikit, komputasi lebih cepat, data redundant bisa dihilangkan, dll

Curse of Dimensionality

Hughes phenomenon shows that as the number of features increases, the classifier's performance increases as well until we reach the optimal number of features. Adding more features based on the same size as the training set will then degrade the classifier's performance.



Prinsip PCA

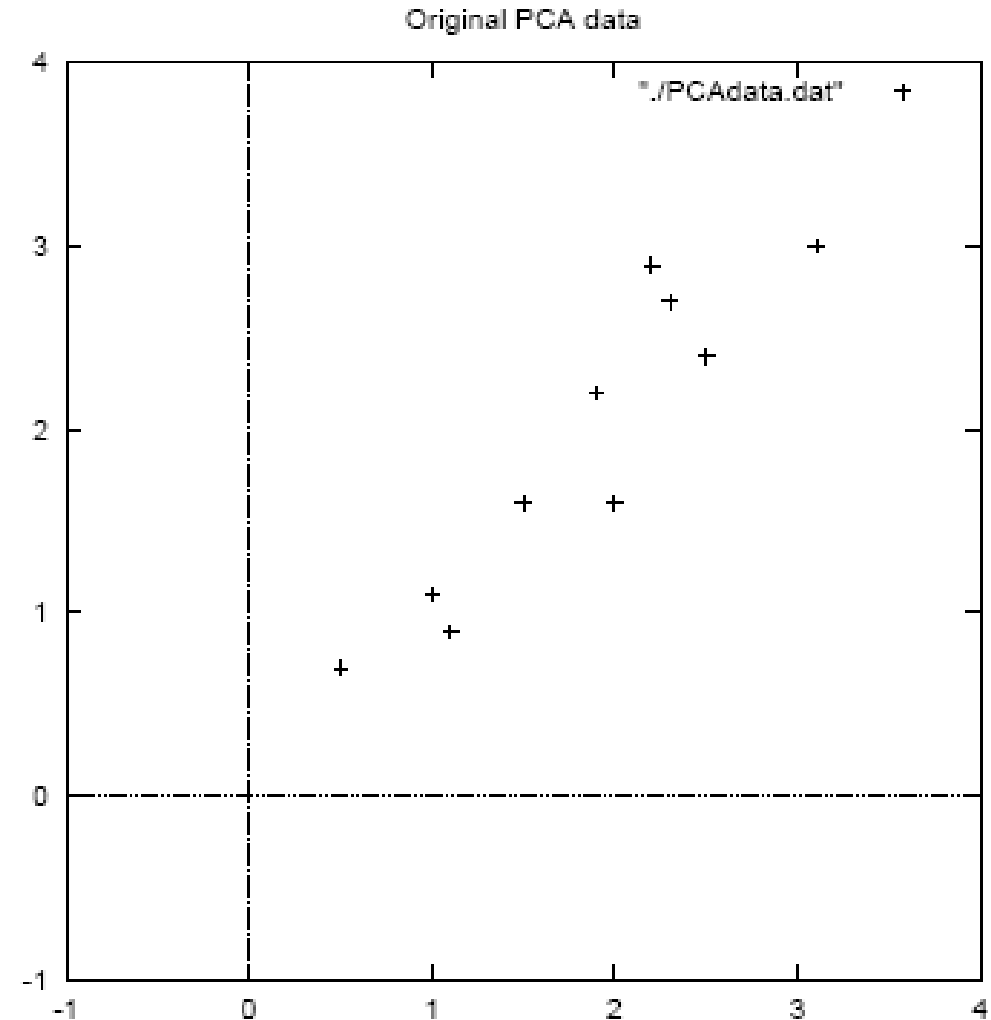
- PCA is “a **transformation** that transfers the data to a **new coordinate system** such that **the greatest variance by any projection of the data comes to lie on the first coordinate** (*first principal component*), the second greatest variance lies on the second coordinate (*second principal component*), and so on.”
- Langkah-Langkah pada PCA:
 1. Lakukan standarisasi terhadap nilai yang ada di setiap variable/kolom
 2. Hitung matriks Kovarian untuk melihat hubungan antar variabel
 3. Hitung nilai dan vektor Eigen dari matriks Kovarian tersebut
 4. Urutkan nilai & vektor Eigen untuk menentukan komponen utama
 5. Buat vektor fitur baru (hasil transformasi) berdasarkan komponen utama yang dipilih

Contoh Soal PCA

- Misalkan kita memiliki data dengan 2 variabel/atribut/kolom sebagai berikut:

Data =

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9



Langkah 1: Standarisasi Nilai

- Langkah pertama yang perlu dilakukan adalah standarisasi nilai, dengan meletakkan mean di 0.

- $\text{DataAdjust} = \text{Data} - \text{mean}$

	x	y
	2.5	2.4
	0.5	0.7
	2.2	2.9
	1.9	2.2
Data =	3.1	3.0
	2.3	2.7
	2	1.6
	1	1.1
	1.5	1.6
	1.1	0.9

Mean: 1.81 1.91

	x	y
	.69	.49
	-1.31	-1.21
	.39	.99
	.09	.29
DataAdjust =	1.29	1.09
	.49	.79
	.19	-.31
	-.81	-.81
	-.31	-.31
	-.71	-1.01

Mean: 0 0

visually: translate the data blob to origin
(0,0)

Langkah 2 & 3

2. Hitung matriks kovarian:

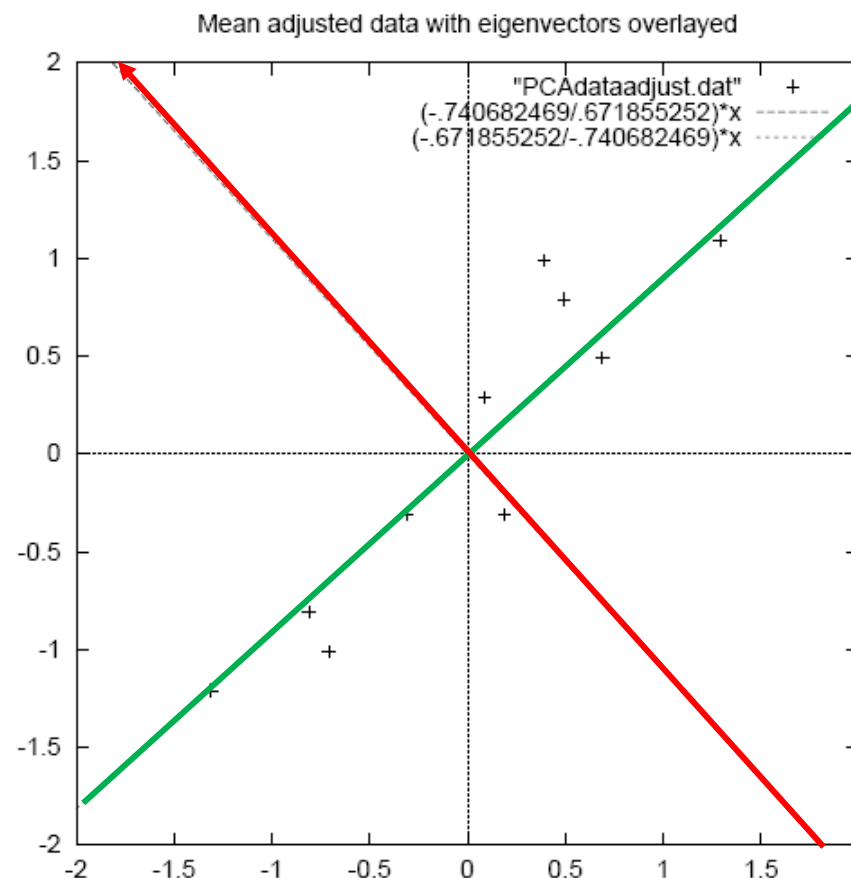
$$\text{cov} = \begin{pmatrix} \mathbf{x} & \mathbf{y} \\ .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix} \begin{matrix} \mathbf{x} \\ \mathbf{y} \end{matrix}$$

3. Hitung nilai dan vektor eigen dari matriks kovarian

$$\text{eigenvalues} = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$\text{eigenvectors} = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

Variansi data lebih besar di sumbu hijau atau merah?



Vektor-vektor eigen membentuk basis baru (kadang disebut **principle component** atau **directions**)

Figure 3.2: A plot of the normalised data (mean subtracted) with the eigenvectors of the covariance matrix overlayed on top.

Internalisasi

- Mengapa yang dilihat matriks kovariannya?
- Mengapa vektor eigen dari matriks kovarian bisa menjadi basis baru yang “pas” dengan data, padahal yang dia lihat adalah nilai **varian/kovarian** dari data lho, **bukan data aslinya**??

$$A\mathbf{v} = \lambda\mathbf{v}$$

$$\begin{bmatrix} .61656 & .61544 \\ .61544 & .71656 \end{bmatrix} \begin{bmatrix} -.67787 \\ -.73518 \end{bmatrix} = 1.284 \begin{bmatrix} -.67787 \\ -.73518 \end{bmatrix}$$

matriks kovarian vekt.eig nilai eig



- So what? 😊

Intuisi

- Komponen data (fitur) yang berkorelasi tinggi dengan komponen data (fitur) lainnya tidak diinginkan. (Mengapa?)
- Data ideal yang diharapkan adalah yang antar komponen datanya tidak berkorelasi.
- Pasangan komponen data yang tidak berkorelasi, harusnya nilai covariance nya 0
- Dengan demikian, matriks covariance yang diharapkan berbentuk matriks diagonal.

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
sepal length (cm)	0.685694	-0.039268	1.273682	0.516904
sepal width (cm)	-0.039268	0.188004	-0.321713	-0.117981
petal length (cm)	1.273682	-0.321713	3.113179	1.296387
petal width (cm)	0.516904	-0.117981	1.296387	0.582414



Diagonalisasi
matriks

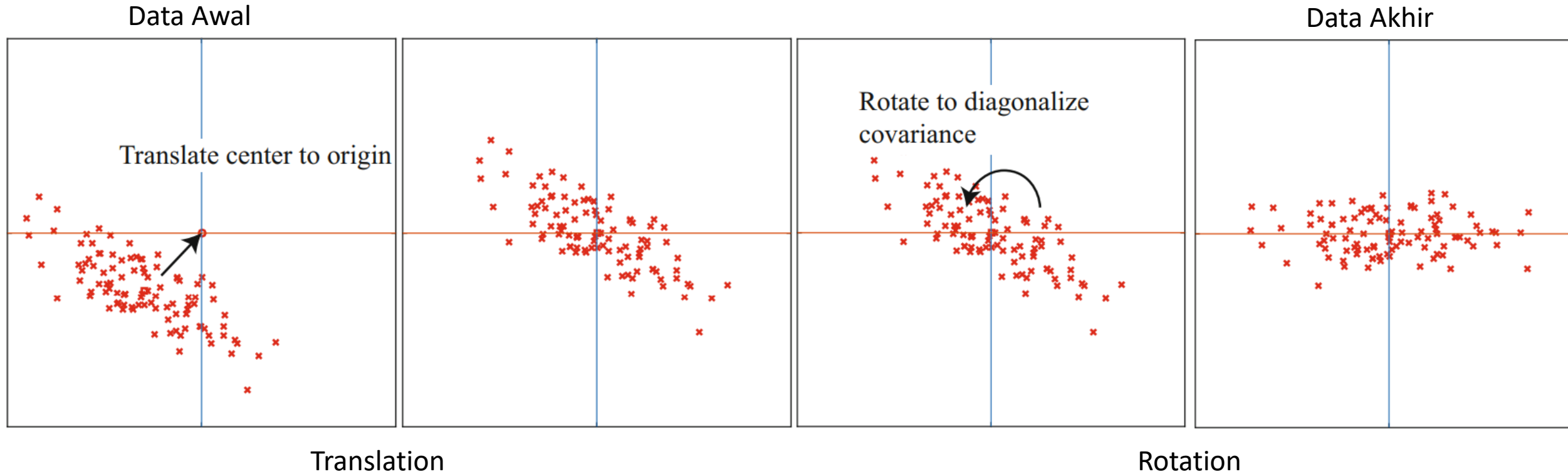
$$\begin{bmatrix} \mathbf{a} & 0 & 0 & 0 \\ 0 & \mathbf{b} & 0 & 0 \\ 0 & 0 & \mathbf{c} & 0 \\ 0 & 0 & 0 & \mathbf{d} \end{bmatrix}$$

Sudah mulai bisa *connect the dots*? 😊

Intuisi

- Suatu matriks ***C*** berukuran $n \times n$ disebut dapat didiagonalisasi menjadi matriks diagonal ***D***, jika terdapat matriks invertible ***P*** sehingga
$$\mathbf{D} = \mathbf{P}^{-1} \mathbf{C} \mathbf{P}$$
- ***P*** merupakan matriks yang kolom-kolomnya merupakan eigenvektor dari ***C***
- ***D*** merupakan matriks diagonal yang elemen diagonalnya merupakan nilai-nilai eigen dari ***C***.
- (Silakan pelajari kembali cara mencari eigenvector dan eigenvalue 😊)
- Apa sebenarnya makna dari “mengubah dataset sehingga komponen-komponen datanya tidak saling berkorelasi” ?
- Kita akan lihat secara visual...

Visualisasi



Pada data akhir, scatter-plot akan berhimpit ke salah satu sumbu koordinat, membentuk semacam ellipsoid. Data tidak berkorelasi lagi, namun data memiliki variansi yang tinggi pada suatu sumbu dibandingkan sumbu lainnya.

Bagaimana cara mengubah data sehingga memiliki sifat yang diinginkan tersebut? Data harus dirotasikan berapa derajat? Matriks transformasinya seperti apa?

Jadi...

- Matriks kovarian C yang tidak diagonal, ingin dijadikan matriks diagonal D
- Caranya dengan mencari matriks P sedemikian hingga $D = P^{-1}CP$
- P disusun dari vektor-vektor eigen matriks C
- Jika matriks kovarian diubah menggunakan matriks P , maka datapoints juga harusnya diubah (ditransformasi) menggunakan matriks P
- Atau bisa juga dipandang bahwa kita sekarang sudah memiliki sumbu atau basis baru yang terdiri dari vektor-vektor eigen dari C , maka dengan demikian kita bisa mencari **matriks transformasinya** 😊
- Caranya bisa diikuti langkah-langkah PCA selanjutnya.

Langkah 4: Urutkan NVE

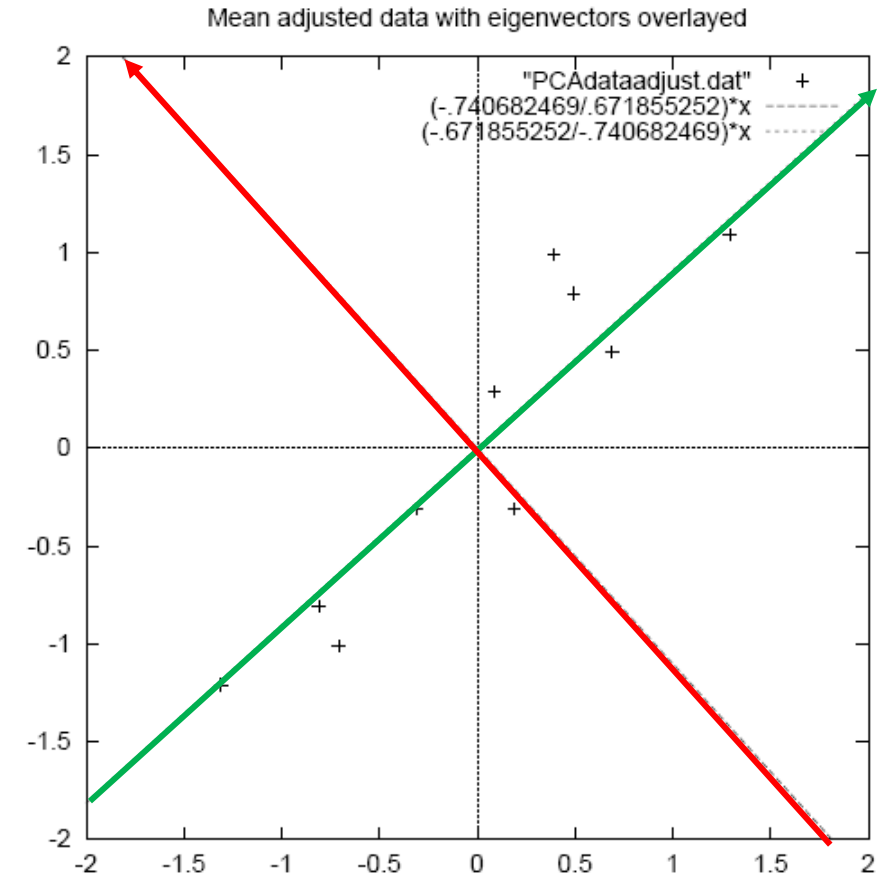
4. Urutkan vektor-vektor eigen berdasarkan nilai eigen yang bersesuaian. Urutkan berdasarkan nilai eigen tertinggi ke terendah.

$$\mathbf{v}_1 = \begin{pmatrix} -.677873399 \\ -.735178956 \end{pmatrix} \quad \lambda = 1.28402771$$

$$\mathbf{v}_2 = \begin{pmatrix} -.735178956 \\ .677873399 \end{pmatrix} \quad \lambda = .0490833989$$

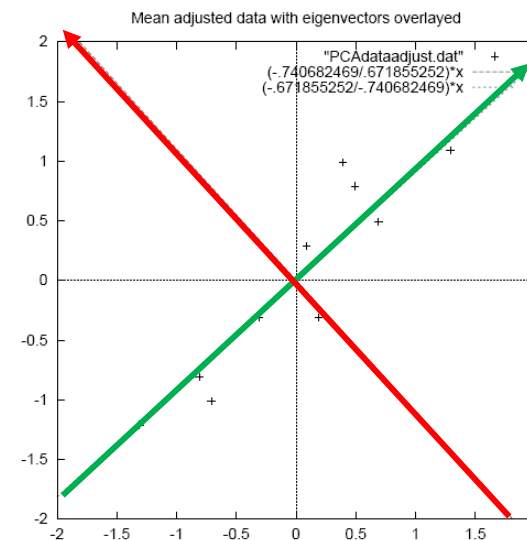
First principal component: \mathbf{v}_1 (hijau)

Second principal component: \mathbf{v}_2 (merah)



Langkah 5: Buat vektor fitur baru

- Ingat bahwa yang ingin kita lakukan adalah mentransformasi data ke dalam basis yang baru. Bagaimana mencari matriks transformasinya?
- Ingat di contoh sebelumnya, cara mudah mencari $[T]$: gabungkan vektor-vektor basis column-wise menjadi matriks, lalu cari inversnya 😊
- Pada PCA, yang menjadi basis adalah principal component (yang terdiri dari vektor-vektor eigen, yaitu:
- $$[T] = \begin{bmatrix} -.677873399 & -.735178656 \\ -.735178656 & .677873399 \end{bmatrix}^{-1} = \begin{bmatrix} -.677873399 & -.735178656 \\ -.735178656 & .677873399 \end{bmatrix}$$
- Mengapa inversnya bisa sama dengan matriks transpose?
- Hint: matriks orthogonal, matriks simetri 😊



Langkah 5

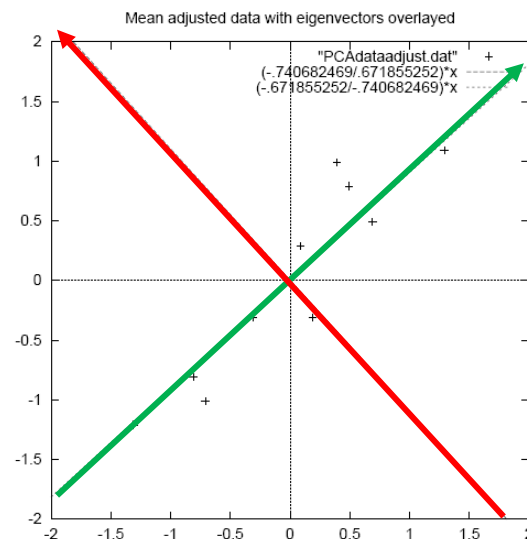
- Vektor Fitur yang baru

- Untuk data pertama:

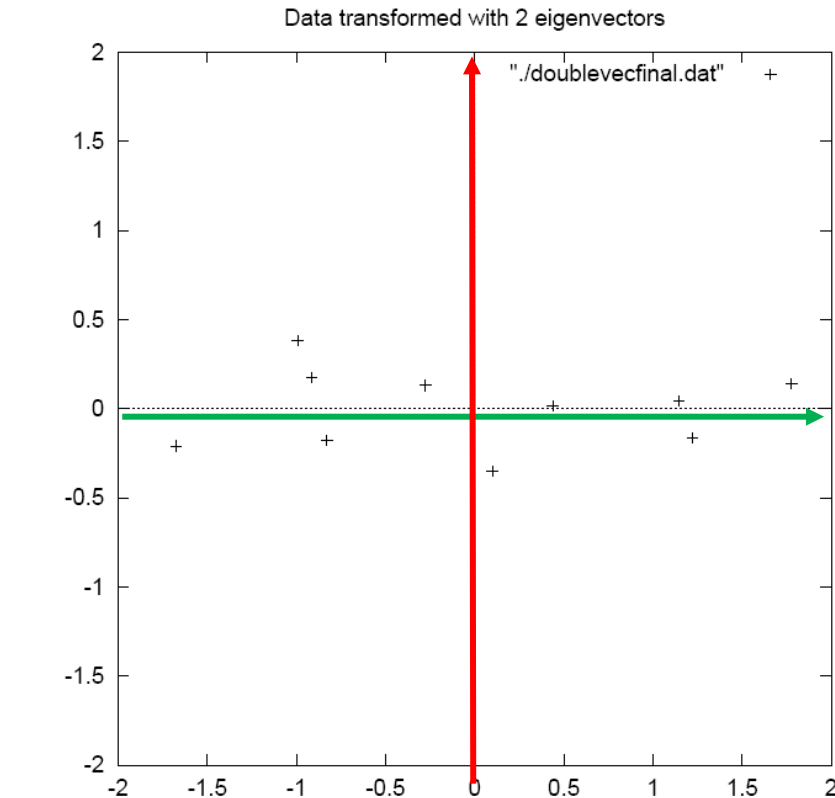
$$- \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -.677873399 & -.735178656 \\ -.735178656 & .677873399 \end{bmatrix} \begin{bmatrix} .69 \\ .49 \end{bmatrix} = \begin{bmatrix} -.82797 \\ -.17511 \end{bmatrix}$$

DataAdjust =

x	y
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.01



x	y
-.827970186	-.175115307
1.77758033	.142857227
-.992197494	.384374989
-.274210416	.130417207
-1.67580142	-.209498461
-.912949103	.175282444
.0991094375	-.349824698
1.14457216	.0464172582
.438046137	.0177646297
1.22382056	-.162675287



Apa artinya?

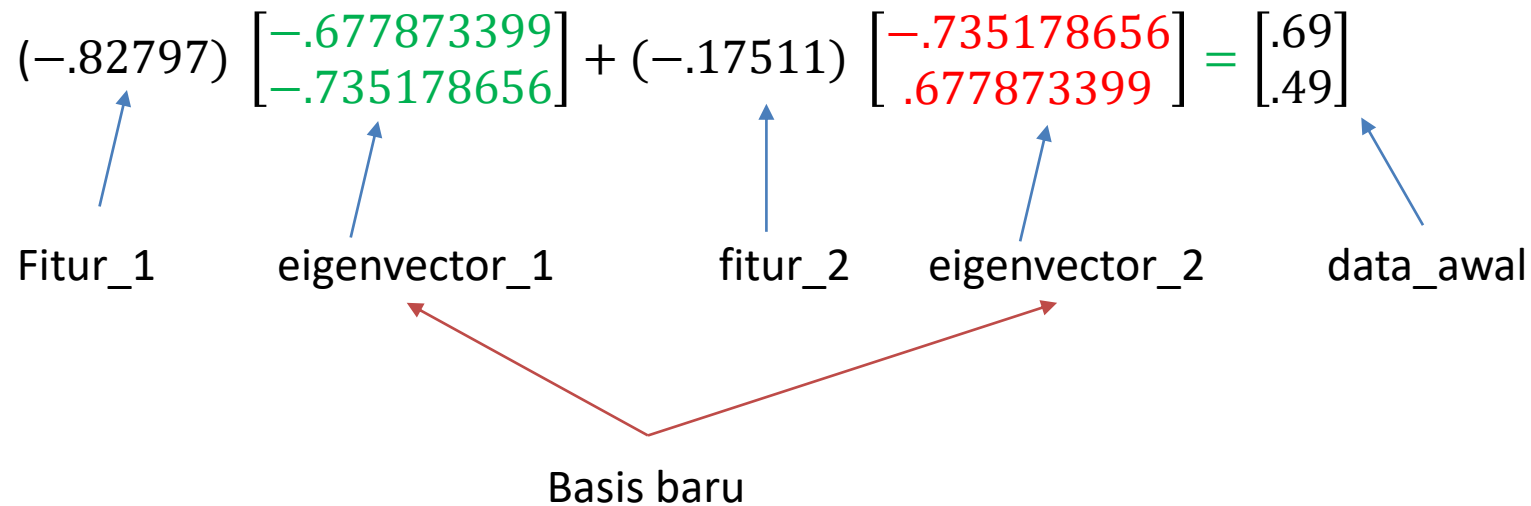
- Koordinat adjusted data pertama = (0.69, 0.49)
- Koordinat data pertama setelah ditransformasi = (-0.82797, -0.17511) → fitur baru (engineered feature)

$$\begin{bmatrix} -.677873399 & -.735178656 \\ -.735178656 & .677873399 \end{bmatrix} \begin{bmatrix} .69 \\ .49 \end{bmatrix} = \begin{bmatrix} -.82797 \\ -.17511 \end{bmatrix}$$

$$\begin{matrix} \text{Fitur_1} & \text{eigenvector_1} & \text{fitur_2} & \text{eigenvector_2} & \text{data_awal} \end{matrix}$$

$$(-.82797) \begin{bmatrix} -.677873399 \\ -.735178656 \end{bmatrix} + (-.17511) \begin{bmatrix} -.735178656 \\ .677873399 \end{bmatrix} = \begin{bmatrix} .69 \\ .49 \end{bmatrix}$$

Basis baru



x	y		x	y		x	y
2.5	2.4		.69	.49		-.827970186	-.175115307
0.5	0.7		-1.31	-1.21		1.77758033	.142857227
2.2	2.9		.39	.99		-.992197494	.384374989
1.9	2.2		.09	.29		-.274210416	.130417207
Data = 3.1	3.0	Data Adjust =	1.29	1.09	Transformed Data =	-1.67580142	-.209498461
2.3	2.7		.49	.79		-.912949103	.175282444
2	1.6		.19	-.31		.0991094375	-.349824698
1	1.1		-.81	-.81		1.14457216	.0464172582
1.5	1.6		-.31	-.31		.438046137	.0177646297
1.1	0.9		-.71	-1.01		1.22382056	-.162675287

Pertanyaan

1. Apakah data yang sudah ditransformasi $(-0.82797, -0.17511)$ bisa dikembalikan ke data sebelum ditransformasi $(0.69, 0.49)$? Bagaimana caranya?
2. Apakah data yang sudah distandarisasi $(0.69, 0.49)$ bisa dikembalikan ke data awal $(2.5, 2.4)$? Bagaimana caranya?

Solusi (1)

1. Apakah data yang sudah ditransformasi (-0.82797, -0.17511) bisa dikembalikan ke data sebelum ditransformasi (0.69, 0.49)?
Bagaimana caranya?

Ingat cara mengubah data adjusted ke data hasil transformasi PCA:

$$\begin{bmatrix} -.677873399 & -.735178656 \\ -.735178656 & .677873399 \end{bmatrix} \begin{bmatrix} .69 \\ .49 \end{bmatrix} = \begin{bmatrix} -.82797 \\ -.17511 \end{bmatrix}$$

untuk mengembalikan ke awal?

$$\begin{bmatrix} & \end{bmatrix} \begin{bmatrix} -.82797 \\ -.17511 \end{bmatrix} = \begin{bmatrix} .69 \\ .49 \end{bmatrix}$$

(apa matriks transformasinya? 😊)

Solusi (2)

2. Apakah data yang sudah distandarisasi (0.69, 0.49) bisa dikembalikan ke data awal (2.5, 2.4)? Bagaimana caranya?

$\text{DataAdjust} = \text{Data} - \text{mean}$

$\text{Data} = \text{DataAdjust} + \text{mean}$ 😊

Mean data x adalah 1.81

maka $0.69 + 1.81 = 2.50$

Reduksi Dimensinya?

- Se jauh ini kita belum melakukan reduksi dimensi, sehingga data hasil transformasi bisa dikembalikan ke data awal tanpa kehilangan informasi apapun.
- Bagaimana cara mereduksi dimensi di PCA?
- Secara sederhana, hilangkan saja sumbu atau vektor pada basis yang paling kecil nilai eigennya 😊 atau kalau sudah diurutkan, buang yang paling bawah.
- Karena contoh kita hanya terdiri dari 2 dimensi (agar mudah divisualisasikan), mari kita coba bagaimana hasilnya kalau kita membuang dimensi kedua.

Revisit Langkah 5: 1 Dimensi (1 principal component)

- Berhubung dimensi kedua dihilangkan, berarti matriks transformasinya menjadi:

$$- [T] = \begin{bmatrix} -.677873399 & -.735178656 \end{bmatrix}$$

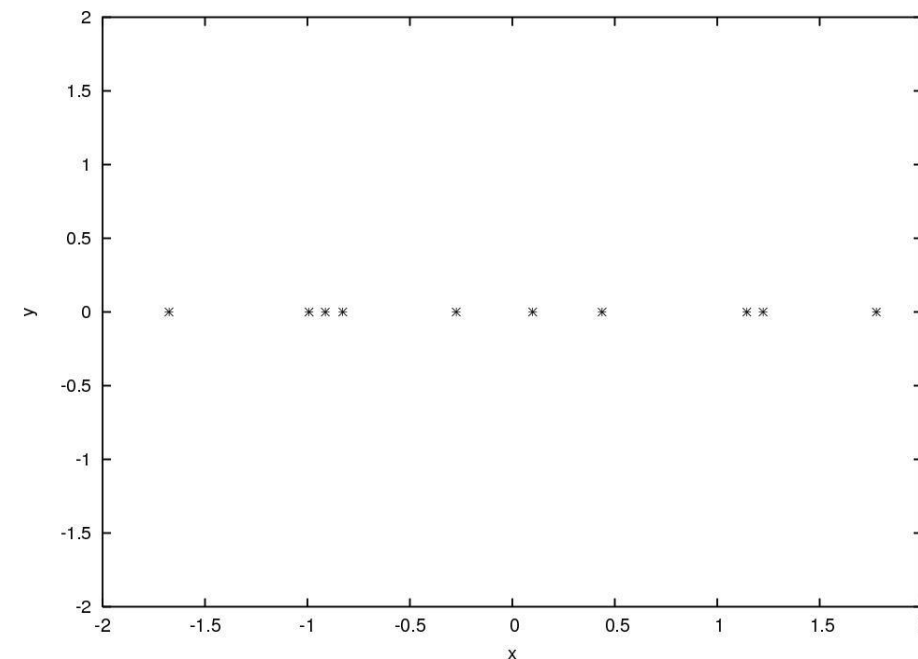
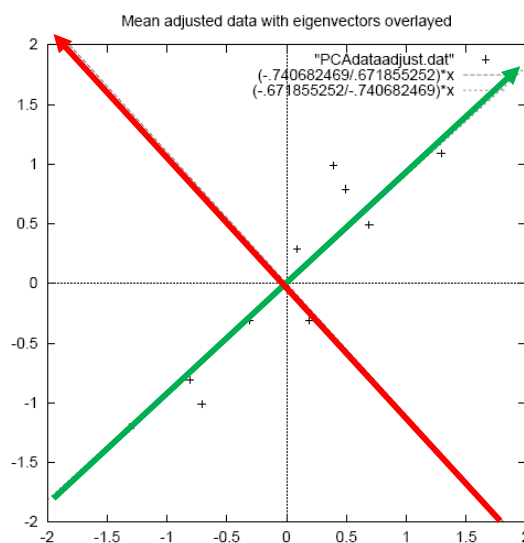
$$- \text{Data pertama: } [x'] = \begin{bmatrix} -.677873399 & -.735178656 \end{bmatrix} \begin{bmatrix} .69 \\ .49 \end{bmatrix} = [-.82797]$$

Transformed Data (Single eigenvector)

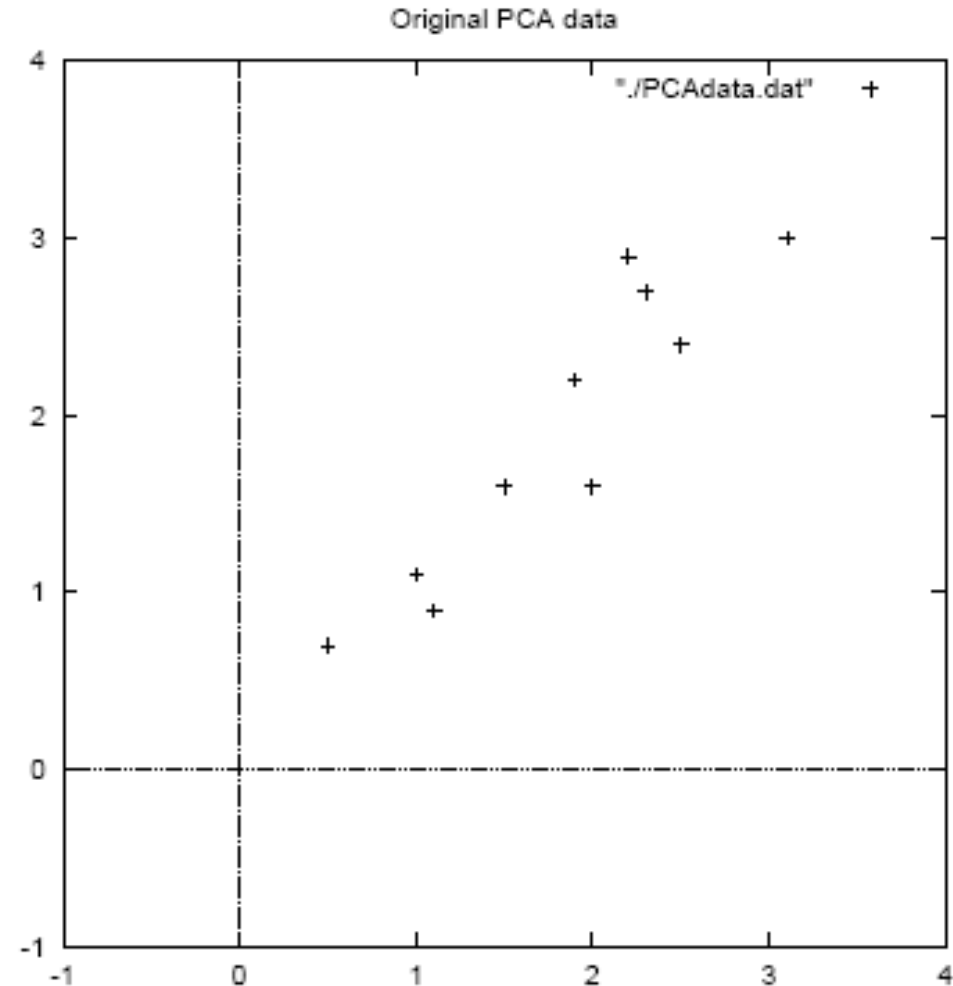
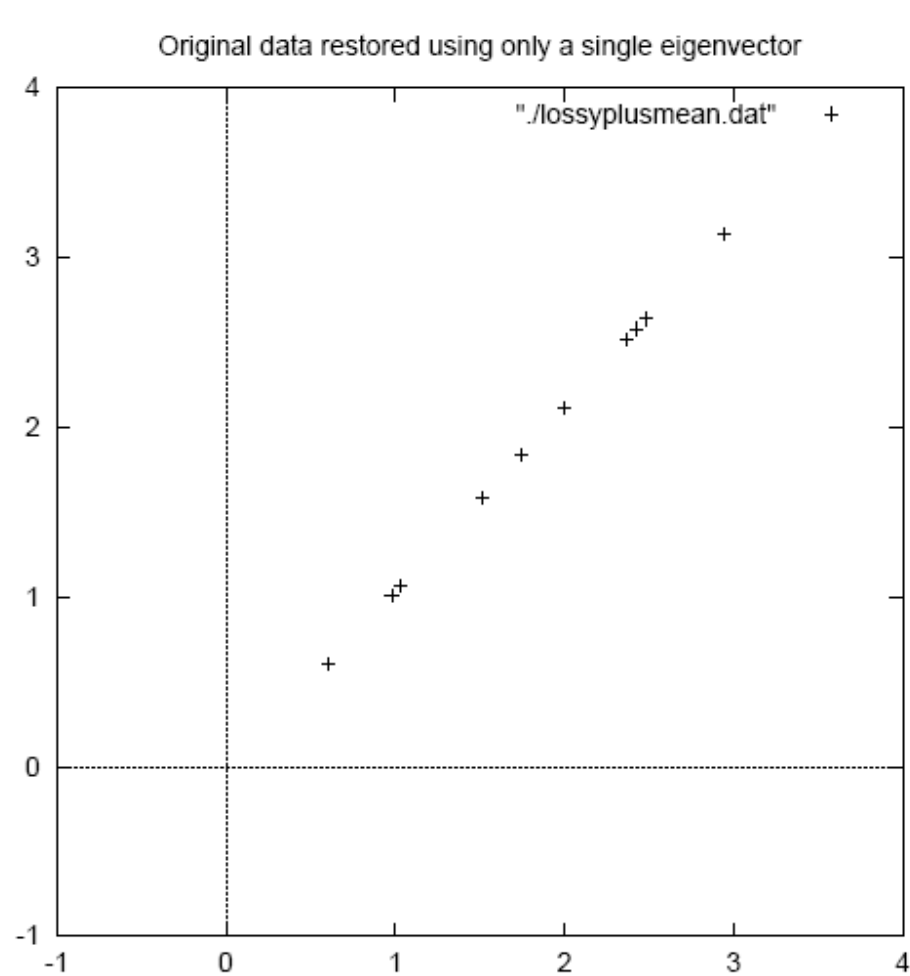
x'
-.827970186
1.77758033
-.992197494
-.274210416
-1.67580142
-.912949103
.0991094375
1.14457216
.438046137
1.22382056

DataAdjust =

x	y
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.01



Perbandingan 1 & 2 principal component



Can you calculate the error between restored/reconstructed data and the original one?

Explained Variance

- Secara umum, bagaimana cara menentukan berapa *principle components* yang harus dipilih?
- Bisa menggunakan analisis terhadap variansi yang diperoleh dari matriks kovarian.
- *Explained variance* dari *principal component* ke-k

$$= \frac{\lambda_k}{\lambda_1 + \lambda_2 + \dots + \lambda_n}$$

- λ_k = nilai eigen ke-k

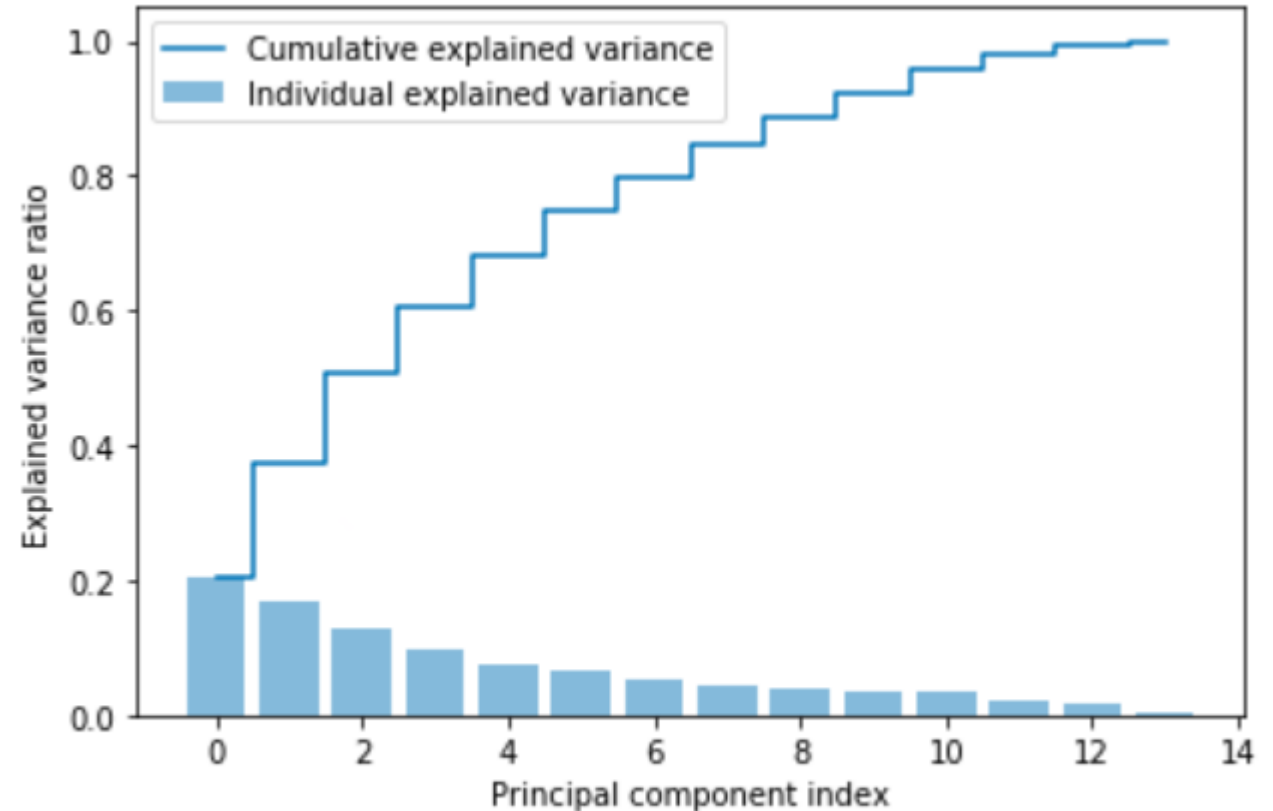


Fig 2. Explained Variance using sklearn PCA

- [PCA Explained Variance Concepts with Python Example - Data Analytics \(vitalflux.com\)](https://vitalflux.com/pca-explained-variance-concepts-with-python-example/)

Contoh Aplikasi PCA: Face Recognition

- [Faces recognition example using eigenfaces and SVMs — scikit-learn 1.1.2 documentation](#)
- Dataset: Labeled Faces in the Wild (LFW) <http://vis-www.cs.umass.edu/lfw>
- Terdiri dari ribuan gambar wajah berukuran aslinya 250 x 250 piksel



Contoh Aplikasi PCA: Face Recognition

[Faces recognition example using eigenfaces and SVMs — scikit-learn 1.1.2 documentation](#)

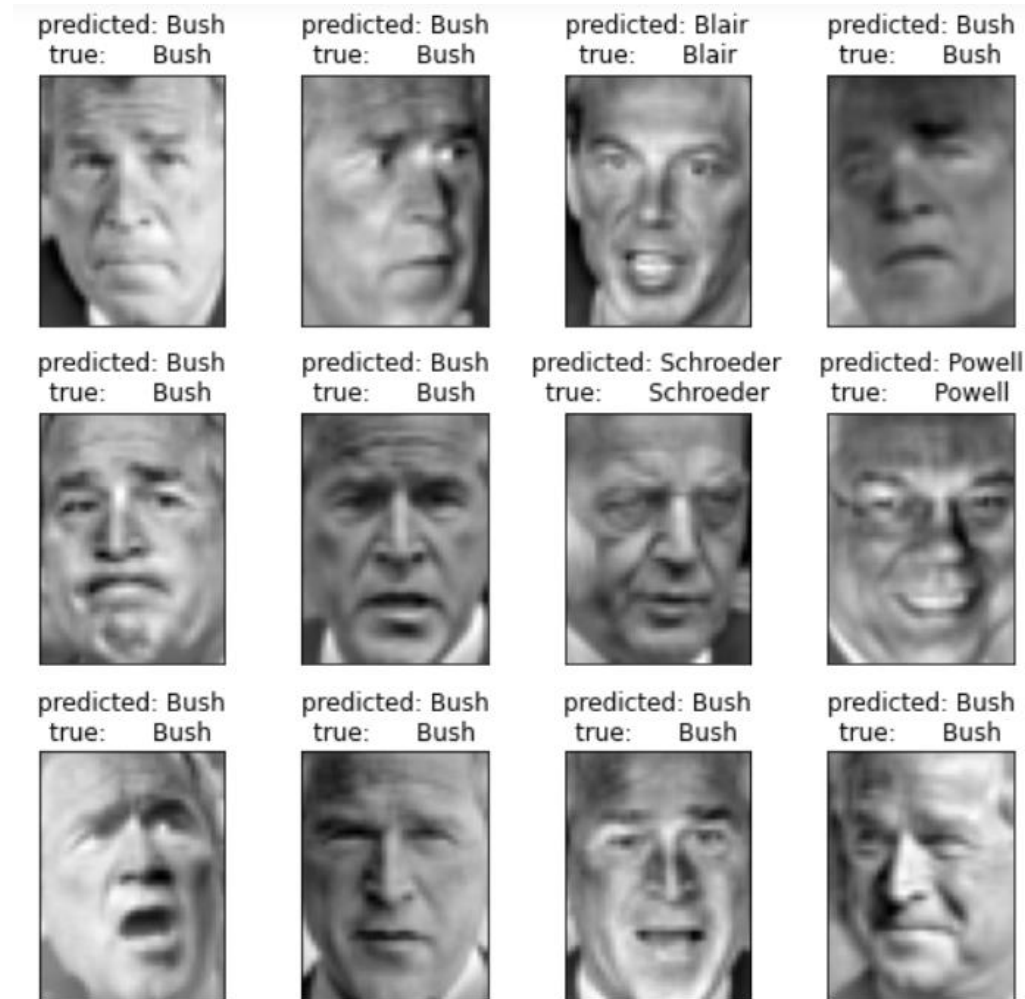
- Jumlah gambar yang digunakan adalah 1288 gambar
- Gambar awal di-resize menjadi ukuran 50x37 piksel, kemudian di-flatten sehingga setiap gambar direpresentasikan sebagai sebuah vektor berukuran $50 \times 37 = 1850$
- Kemudian dilakukan PCA dengan mengambil $n = 150$
- Perhatikan bahwa dimensi sudah direduksi dari 1850 menjadi 150 saja 😊

Contoh Aplikasi PCA: Face Recognition

- Gambar yang akan dikenali hanya milik 7 orang saja, yaitu

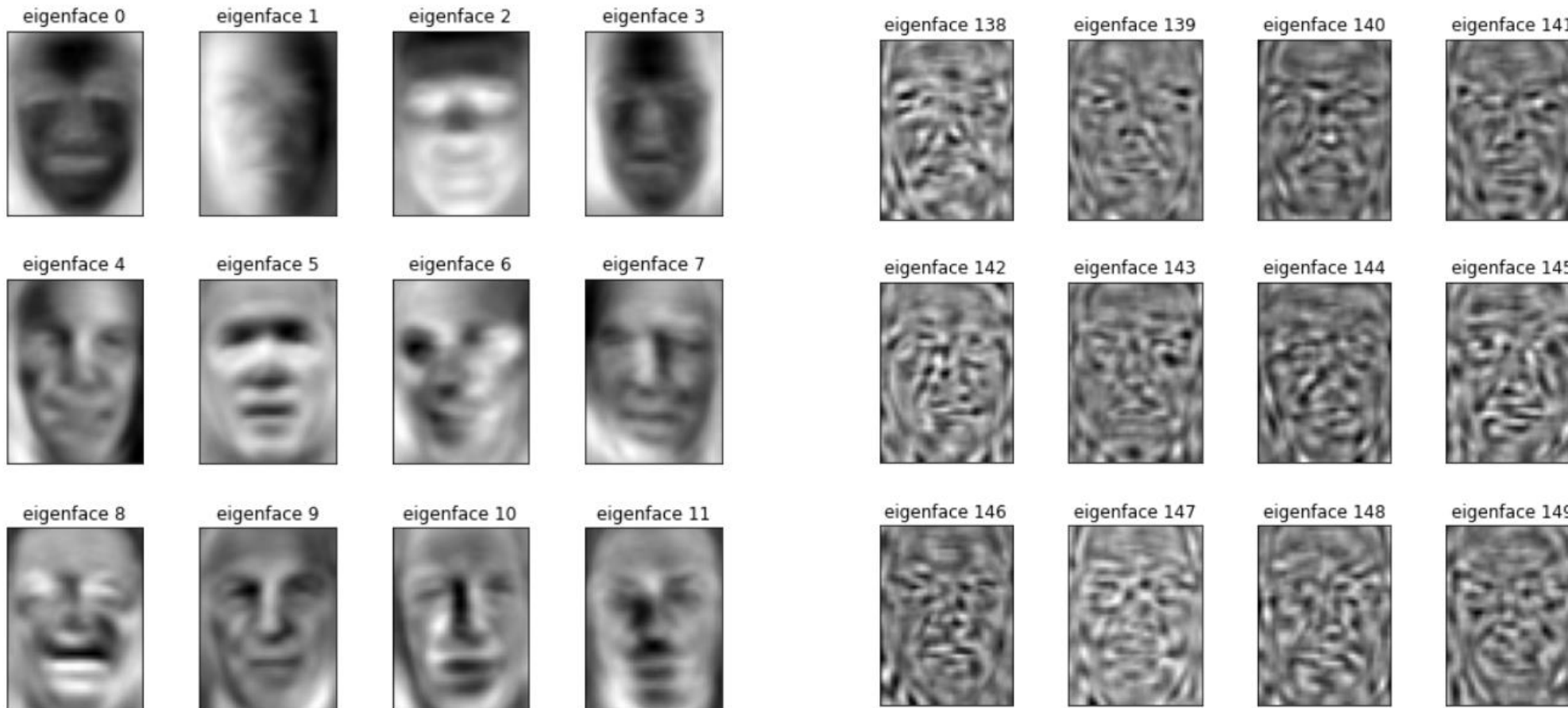
Ariel Sharon
Colin Powell
Donald Rumsfeld
George W Bush
Gerhard Schroeder
Hugo Chavez
Tony Blair

- Perhatikan bahwa dengan menggunakan classifier SVM, 150 fitur dari PCA sudah berhasil mengenali wajah seperti terlihat pada contoh.




Contoh Aplikasi PCA: Face Recognition

- Tampilan dari 12 eigenface pertama dan terakhir dapat dilihat dari gambar berikut.



Contoh Aplikasi PCA: Face Recognition

- Perhatikan bahwa eigenfaces tersebut sebenarnya adalah vektor-vektor eigen yang membentuk basis.
- Dengan demikian setiap wajah input sebenarnya dapat direpresentasikan dalam 150 bilangan (koefisien) saja, yaitu $c_0 \dots c_{149}$



$$\text{Input Face Image} = c_0 \text{eigenface 0} + c_1 \text{eigenface 1} + \dots + c_{148} \text{eigenface 148} + c_{149} \text{eigenface 149}$$

- Jika dihitung explained variance ratio dari 150 komponen ini, nilainya adalah 0.955
- Padahal data sudah direduksi menjadi hanya $150/1850 = 8.1\%$ saja 😊