

Copy of 03-01-Vectors.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

RAM Disk

Vektor

Vektor, dan ruang vektor, merupakan konsep fundamental dalam *aljabar linear*, dan digunakan dalam banyak model pembelajaran mesin. Vektor menggambarkan garis dan bidang dalam ruang, sehingga memungkinkan kita melakukan perhitungan untuk mengeksplorasi hubungan dalam ruang multi-dimensi.

Apa itu Vektor

Secara sederhana, vektor adalah elemen numerik yang memiliki *magnitudo* dan *arah*. Magnitudo merepresentasikan jarak (misalnya, "2 mil") dan arah menunjukkan ke mana vektor tersebut menuju (misalnya, "Timur"). Vektor didefinisikan oleh koordinat berdimensi-n yang menggambarkan sebuah titik di ruang yang dapat dihubungkan oleh garis dari suatu titik asal sembarang.

Penjelasan ini mungkin terdengar agak rumit, jadi mari kita mulai dengan contoh sederhana dua dimensi. Dalam kasus ini, kita memiliki vektor yang ditentukan oleh sebuah titik pada bidang dua dimensi: Koordinat dua dimensi terdiri dari nilai *x* dan *y*, dan di sini kita gunakan **2** untuk *x* dan **1** untuk *y*.

Vektor kita dapat ditulis sebagai $\mathbf{v} = (2,1)$, tetapi secara lebih formal kita menggunakan notasi berikut, di mana nilai koordinat dimensi vektor ditampilkan sebagai matriks:

$$\vec{v} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

Jadi apa artinya? Koordinat tersebut berdimensi dua dan menggambarkan perpindahan yang diperlukan untuk mencapai titik ujung (*head*) dari vektor — dalam kasus ini kita bergerak 2 satuan pada dimensi *x* dan 1 satuan pada dimensi *y*. Perhatikan bahwa kita tidak menentukan titik awal vektor — kita hanya menggambarkan koordinat tujuan yang merangkum magnitudo dan arah vektor. Anggap saja sebagai petunjuk ar

```
[5] ✓ 0s
%matplotlib inline

import numpy as np
import matplotlib.pyplot as plt

# We'll use a numpy array for our vector
v = np.array([2,1])

# and we'll use a quiver plot to visualize it.
origin = [0], [0]
plt.axis('equal')
plt.grid()
plt.ticker_label_format(style='sci', axis='both', scilimits=(0,0))
plt.quiver(*origin, *v, scale=10, color='r')
plt.show()
```

Perhatikan bahwa kita dapat menggunakan *numpy array* untuk mendefinisikan vektor di Python; jadi untuk membuat vektor (2,1), kita cukup membuat array numpy dengan elemen [2,1]. Kita kemudian menggunakan plot *quiver* untuk memvisualisasikan vektor tersebut, dengan titik 0,0 sebagai titik awal (atau *origin*).

Vektor (2,1) ditampilkan sebagai panah yang dimulai dari 0,0 dan bergerak 2 satuan sepanjang sumbu *x* (ke kanan) dan 1 satuan sepanjang sumbu *y* (ke atas).

Menghitung Magnitudo dan Arah Vektor

Kita biasanya bekerja dengan vektor dengan menyatakan komponennya sebagai *koordinat kartesius*; yaitu nilai *x* dan *y* (serta dimensi lainnya) yang menentukan jumlah satuan perpindahan pada setiap dimensi. Jadi koordinat dari vektor (2,1) menunjukkan bahwa kita bergerak 2 satuan sepanjang sumbu *x* dan 1 satuan sepanjang sumbu *y*.

Namun, vektor juga dapat dinyatakan dalam *koordinat polar*, yaitu koordinat yang menggambarkan magnitudo dan arah vektor. Magnitudo adalah jarak total dari pangkal (tail) ke ujung (head) vektor, sedangkan arah adalah sudut orientasi vektor tersebut.

Menghitung Magnitudo

Menghitung magnitudo vektor dari koordinat kartesiusnya berarti mengukur jarak antara titik awal sembarang dan titik ujung vektor. Untuk vektor dua dimensi, ini sama saja dengan menghitung panjang sisi miring (hipotenusa) dari segitiga siku-siku — sehingga kita dapat menggunakan Teorema Pythagoras dan menghitung akar kuadrat dari jumlah kuadrat komponennya, sebagai berikut:

$$\|\vec{v}\| = \sqrt{v_1^2 + v_2^2}$$

Notasi untuk magnitudo vektor adalah dengan mengapit nama vektor menggunakan garis vertikal — dapat menggunakan satu garis ($\|\mathbf{v}\|$) atau dua garis ($\|\mathbf{v}\|$). Garis ganda sering digunakan untuk menghindari kebingungan dengan nilai mutlak. Perhatikan bahwa komponen vektor ditunjukkan dengan indeks subskrip (v_1, v_2, \dots, v_n).

Dalam kasus ini, vektor \mathbf{v} memiliki dua komponen bernilai **2** dan **1**, sehingga perhitungan magnitudonya adalah:

$$\|\vec{v}\| = \sqrt{2^2 + 1^2}$$

yaitu:

$$\|\vec{v}\| = \sqrt{4+1}$$

Sehingga:

$$\|\vec{v}\| = \sqrt{5} \approx 2.24$$

Anda dapat menjalankan kode Python berikut untuk mendapatkan hasil yang lebih presisi (perhatikan bahwa elemen pada numpy array menggunakan indeks berbasis nol).

```
[6] ✓
import math

vMag = math.sqrt(v[0]**2 + v[1]**2)
print (vMag)
```

Perhitungan ini berlaku untuk vektor dengan dimensi berapa pun — kita hanya perlu mengambil akar kuadrat dari jumlah kuadrat setiap komponennya:

$$\|\vec{v}\| = \sqrt{v_1^2 + v_2^2 + \dots + v_n^2}$$

Dalam Python, *numpy* menyediakan pustaka aljabar linear bernama **linalg** yang mempermudah pekerjaan dengan vektor — Anda dapat menggunakan fungsi **norm** pada kode berikut untuk menghitung magnitudo suatu vektor:

```
[7] 0s
import numpy as np

vMag = np.linalg.norm(v)
print (vMag)
```

▼ Menghitung Arah

Untuk menghitung arah, atau *amplitudo*, dari suatu vektor berdasarkan koordinat kartesiusnya, kita perlu menggunakan sedikit trigonometri. Kita dapat memperoleh sudut vektor dengan menghitung *tangen invers*; sering disebut *arctan* (fungsi *tangen* menghitung sudut sebagai rasio — sedangkan tangen invers, atau \tan^{-1} , menyatakannya dalam derajat).

Pada setiap segitiga siku-siku, tangen dihitung sebagai perbandingan sisi *dewan* (*opposite*) terhadap sisi *samping* (*adjacent*). Pada vektor dua dimensi, ini berarti nilai y dibagi nilai x , sehingga untuk vektor v (2,1):

$$\tan(\theta) = \frac{1}{2}$$

Nilai tersebut menghasilkan **0.5**, kemudian kita dapat menggunakan kalkulator untuk menghitung tangen invers guna mendapatkan sudut dalam derajat:

$$\theta = \tan^{-1}(0.5) \approx 26.57^\circ$$

Perhatikan bahwa sudut arah dinyatakan dengan simbol θ .

Jalankan kode Python berikut untuk mengonfirmasi hasil ini:

```
[8] ✓
import math
import numpy as np

v = np.array([2,1])
vTan = v[1] / v[0]
print ('tan = ' + str(vTan))
vAtan = math.atan(vTan)
# atan returns the angle in radians, so convert to degrees
print('inverse-tan = ' + str(math.degrees(vAtan)))
```

Namun ada sedikit komplikasi: jika nilai x atau y (atau keduanya) bernilai negatif, orientasi vektor tidak lagi standar dan kalkulator bisa memberikan nilai \tan^{-1} yang keliru. Untuk memastikan arah vektor benar, gunakan aturan berikut:

- x dan y keduanya positif: gunakan nilai \tan^{-1} .
- x negatif, y positif: tambahkan 180 pada nilai \tan^{-1} .
- x dan y keduanya negatif: tambahkan 180 pada nilai \tan^{-1} .
- x positif, y negatif: tambahkan 360 pada nilai \tan^{-1} .

Untuk memahami alasannya, bayangkan sebuah vektor dapat menunjuk ke arah mana pun dalam lingkaran 360 derajat.

Kita bagi lingkaran tersebut menjadi empat kuadran dengan sumbu x dan y melalui pusatnya.

Sudut dapat diukur dari sumbu x baik ke arah positif (berlawanan arah jarum jam) maupun negatif (searah jarum jam).

Kita beri nomor kuadran dalam arah positif (berlawanan arah jarum jam) sebagai berikut:

2 1
- o -
3 4

Sekarang perhatikan 4 contoh vektor:

1. Vektor [2,4] memiliki x dan y positif. Garis vektor melewati titik (0,0) dari kuadran 3 ke kuadran 1. $\tan^{-1}(4/2) \approx 63.4^\circ$, yaitu sudut positif dari sumbu x ke garis vektor — inilah arah vektor.
2. Vektor [-2,4] memiliki x negatif dan y positif. Garis melewati (0,0) dari kuadran 4 ke kuadran 2. $\tan^{-1}(4/-2) \approx -64.4^\circ$, yaitu sudut negatif dari sumbu x ke garis vektor namun arahnya terbalik. Maka kita menambahkan 180°.
3. Vektor [-2,-4] memiliki x dan y negatif. Garis melewati (0,0) dari kuadran 3 ke kuadran 1. $\tan^{-1}(-4/-2) \approx 63.4^\circ$, tetapi arahnya kembali terbalik, sehingga perlu ditambah 180°.
4. Vektor [2,-4] memiliki x positif dan y negatif. Garis melewati (0,0) dari kuadran 2 ke kuadran 4. $\tan^{-1}(-4/2) \approx -64.4^\circ$. Secara matematis benar (sudut negatif), tetapi untuk sudut positif (berlawanan arah jarum jam) kita tambahkan 360°.

Pada kode Python sebelumnya kita menggunakan fungsi `matr.atan` untuk menghitung tangen invers dari suatu nilai numerik. Pustaka `numpy` menyediakan fungsi serupa bernama `arctan`. Saat bekerja dengan `numpy array`, Anda juga dapat menggunakan fungsi `numpy.arctan2` untuk mendapatkan tangen invers vektor berbasis array dalam satuan `radian`, kemudian menggunakan fungsi `numpy.degrees` untuk mengubahnya menjadi derajat.

Fungsi `arctan2` secara otomatis melakukan penyesuaian yang diperlukan untuk nilai `x` dan `y` negatif.

```
[9] 0s
import numpy as np

v = np.array([2,1])
print ('v: ' + str(np.degrees(np.arctan2(v[1], v[0]))))

s = np.array([-3,2])
print ('s: ' + str(np.degrees(np.arctan2(s[1], s[0]))))
```

✓ Penjumlahan Vektor

Sejauh ini kita bekerja dengan satu vektor pada satu waktu. Lalu apa yang terjadi jika kita perlu menjumlahkan dua vektor?

Mari kita lihat sebuah contoh. Kita sudah memiliki sebuah vektor bernama `v`, yang didefinisikan sebagai berikut:

$$\vec{v} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

Sekarang kita buat vektor kedua dan menamakannya `s` seperti berikut:

$$\vec{s} = \begin{bmatrix} -3 \\ 2 \end{bmatrix}$$

Jalankan sel di bawah ini untuk membuat `s` dan memplotnya bersama dengan `v`:

```
[10] 0s
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline

# definisi vektor
v = np.array([2,1])
s = np.array([-3,2])

print(s)

# gabungkan vektor
vecs = np.array([v, s])

# buat origin untuk tiap vektor (dua titik (0,0))
origins = np.zeros((2,2))

# plot
plt.figure(figsize=(6,6))

plt.axhline(0, color='black')
plt.axvline(0, color='black')

plt.quiver(origins[:,0], origins[:,1],
           vecs[:,0], vecs[:,1],
           color=['red','blue'],
           angles='xy', scale_units='xy', scale=1)

plt.xlim(-5,5)
plt.ylim(-5,5)
plt.gca().set_aspect('equal')

plt.grid()
plt.title("Visualisasi Vektor v dan s")

plt.show()
```

Terlihat pada plot bahwa kedua vektor memiliki arah dan magnitudo yang berbeda.

Lalu apa yang terjadi jika kita menjumlahkannya?

Berikut rumusnya:

$$\vec{z} = \vec{v} + \vec{s}$$

Jika dinyatakan dalam bentuk matriks vektor, menjadi:

$$\vec{z} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} + \begin{bmatrix} -3 \\ 2 \end{bmatrix}$$

Hasilnya adalah:

$$\vec{z} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} + \begin{bmatrix} -3 \\ 2 \end{bmatrix} = \begin{bmatrix} -1 \\ 3 \end{bmatrix}$$

Mari kita verifikasi apakah Python menghasilkan nilai yang sama:

```
[11] 0s
z = v + s
print(z)
```



So what does that look like on our plot?

```
vecs = np.array([v, s, z])  
origin = np.zeros((3,2))  
  
plt.axis('equal')  
plt.grid()  
plt.ticklabel_format(style='sci', axis='both', scilimits=(0,0))  
  
plt.quiver(origin[:,0], origin[:,1],  
           vecs[:,0], vecs[:,1],  
           color=['r','b','g'],  
           angles='xy', scale_units='xy', scale=1)  
  
plt.show()
```

Jadi apa yang sebenarnya terjadi di sini?

Kita menambahkan dimensi vektor **s** ke dimensi vektor **v** untuk membentuk vektor baru **z**. Mari kita uraikan:

- Dimensi **v** adalah (2,1), sehingga dari titik awal kita bergerak 2 satuan pada dimensi *x* (ke kanan) dan 1 satuan pada dimensi *y* (ke atas). Pada plot, jika mulai dari posisi (0,0), ini ditunjukkan oleh panah merah.
- Kemudian kita menambahkan **s**, yang memiliki nilai dimensi (-3, 2), sehingga kita bergerak -3 satuan pada dimensi *x* (ke kiri karena nilainya negatif) dan 2 satuan pada dimensi *y* (ke atas). Pada plot, jika kita mulai dari ujung panah merah dan melakukan perpindahan ini, kita akan berakhir di ujung panah hijau, yang merepresentasikan **z**.

Hal yang sama juga berlaku jika operasi penjumlahan dilakukan sebaliknya, yaitu menambahkan **v** ke **s**. Langkah untuk membentuk **s** ditunjukkan oleh panah biru, dan jika itu digunakan sebagai titik awal untuk **v**, kita juga akan berakhir di ujung panah hijau yang merepresentasikan **z**.

Perhatikan pada plot bahwa jika ekor panah biru dipindahkan sehingga dimulai dari ujung panah merah, maka ujungnya akan berada di lokasi yang sama dengan ujung panah hijau. Hal yang sama berlaku jika ekor panah merah dipindahkan ke ujung panah biru.

[1]

Start coding or generate with AI.

[Colab paid products](#) - [Cancel contracts here](#)

{ } Variables  Terminal



 Python 3