Data Science : Basis Lineaire Algebra

katja.verbeeck@odisee.be joris.maervoet@odisee.be
thomas.vandenbossche@odisee.be

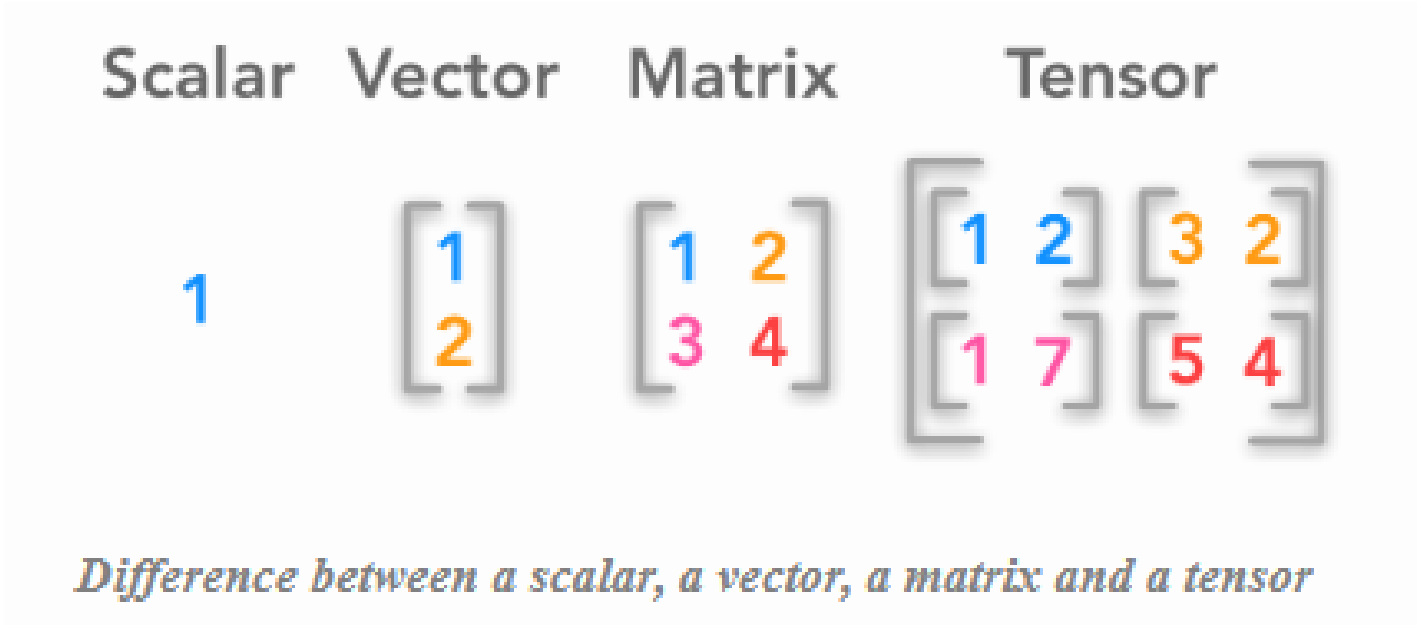
April 25, 2023

Odisee
DE CO-HOGESCHOOL

Inhoud

Scalairen, Vectoren, Matrices en Tensoren

Tensoren van 0 naar n dimensies



Difference between a scalar, a vector, a matrix and a tensor

Tensoren van 0 naar n dimensies

- Een **scalair** is een enkel getal : x
- Een **vector** is een array van getallen : $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$
- Een **matrix** is een 2 dimensionale array :

$$\mathbf{A} = \begin{bmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,n} \\ A_{2,1} & A_{2,2} & \dots & A_{2,n} \\ \dots & & & \\ A_{m,1} & A_{m,2} & \dots & A_{m,n} \end{bmatrix}$$

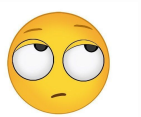
- Een tensor is een n-dimensionale array met $n > 2$

Tensoren van 0 naar n dimensies

- Een **scalair** is een enkel getal : x
- Een **vector** is een array van getallen : $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$
- Een **matrix** is een 2 dimensionale array :

$$\mathbf{A} = \begin{bmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,n} \\ A_{2,1} & A_{2,2} & \dots & A_{2,n} \\ \dots & & & \\ A_{m,1} & A_{m,2} & \dots & A_{m,n} \end{bmatrix}$$

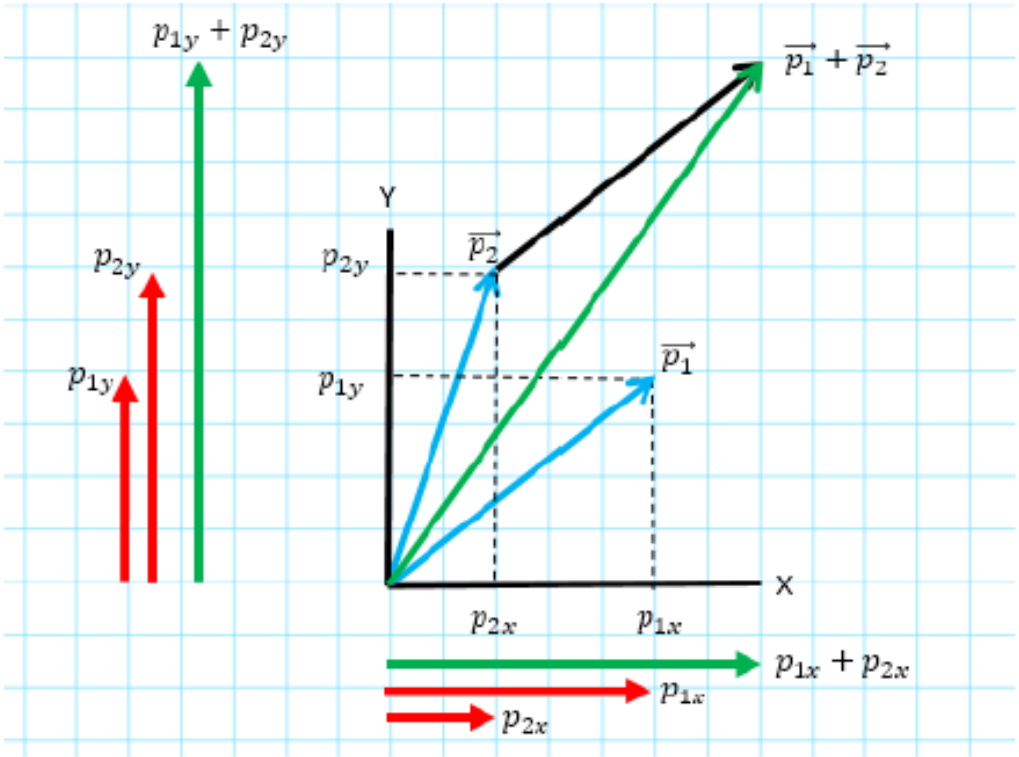
- Een tensor is een n-dimensionale array met $n > 2$

Merk op: wiskundigen tellen typisch vanaf 1 

Bewerkingen op matrices en vectoren

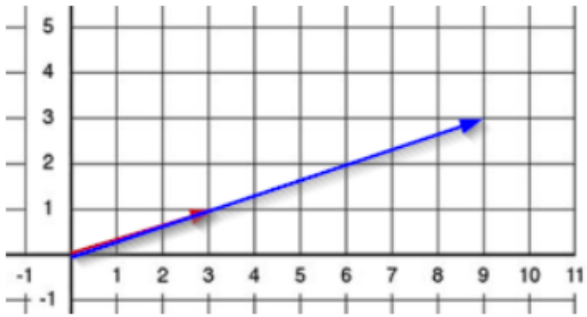
In lineaire algebra kan je elke bewerking algebraïsch uitrekenen maar deze bewerkingen hebben ook steeds een meetkundige betekenis !

Optellen van vectoren



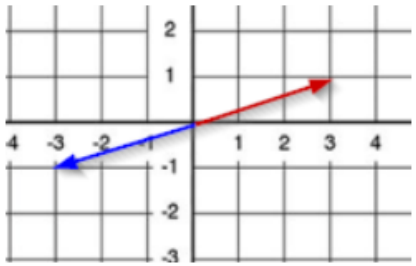
Vectoroptelling via componenten

Vermenigvuldiging met een scalair getal



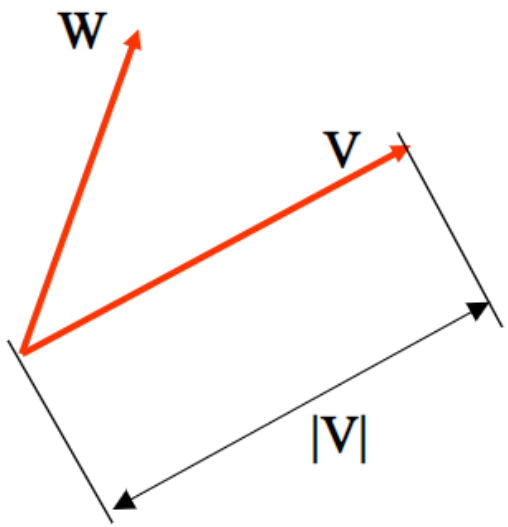
We zien dat de vector $3 \times \vec{a} = \begin{bmatrix} 9 \\ 3 \end{bmatrix}$ in het verlengde ligt van vector $\vec{a} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

$$-1 \times \vec{a} = \begin{bmatrix} -1 \times 3 \\ -1 \times 1 \end{bmatrix} = \begin{bmatrix} -3 \\ -1 \end{bmatrix}$$



in-product van vectoren

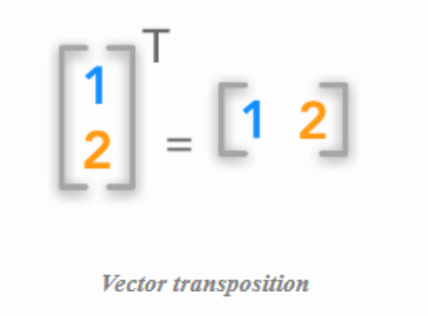
Inwendig product, scalair product ofwel dot product
Product van parallelle componenten



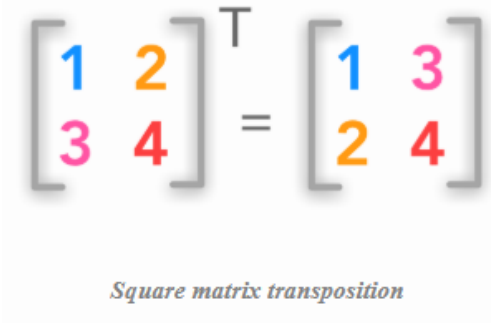
$$\mathbf{V}\cdot\mathbf{W} = |\mathbf{V}||\mathbf{W}|\cos\theta$$
$$= V_x W_x + V_y W_y + V_z W_z$$

Transpositie

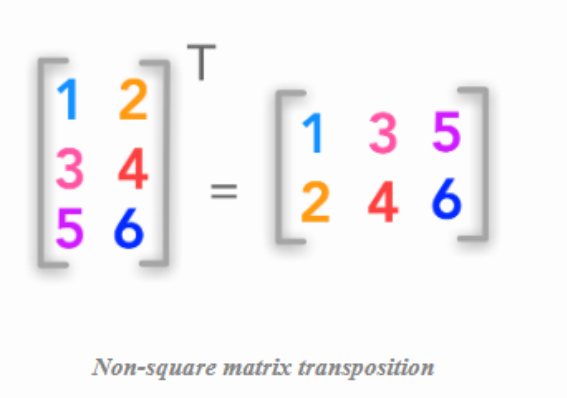
Verwissel rijen en kolommen :



Vector transposition



Square matrix transposition

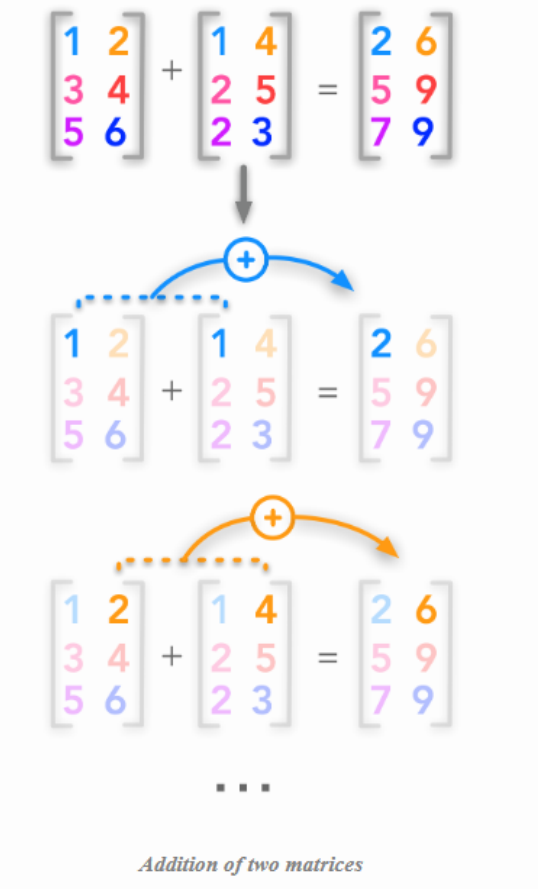


Non-square matrix transposition

Dimensions of matrix transposition

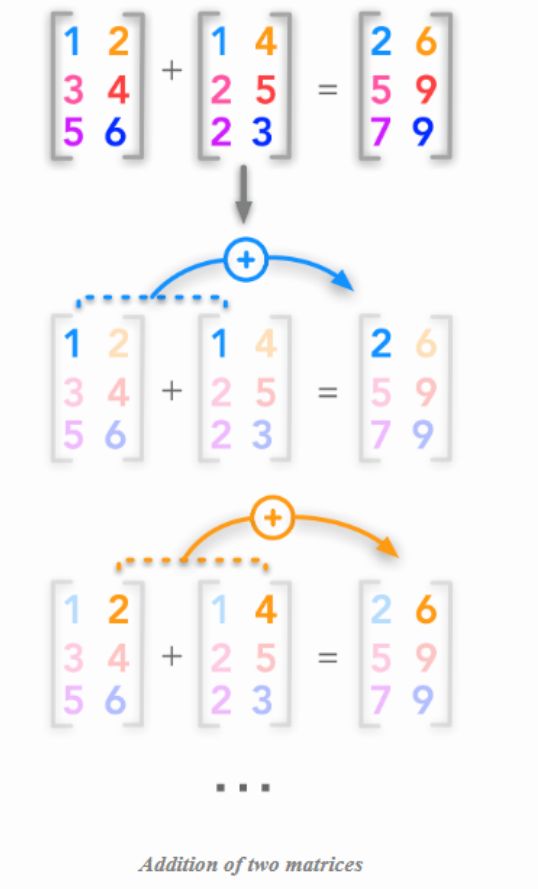
https://hadrienj.github.io/posts/Deep-Learning-Book-Series-2.1-Scalars-Vectors-Matrices-and-Tensors/

Optellen



Addition of two matrices

Optellen



Addition of two matrices

Merk op: wiskundigen houden van symmetrie. De optelling is puntgewijs en dus moeten de dimensies dezelfde zijn 

Optellen

$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix} + \begin{bmatrix} B_{1,1} \\ B_{2,1} \\ B_{3,1} \end{bmatrix} =$$

Optellen

$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix} + \begin{bmatrix} B_{1,1} \\ B_{2,1} \\ B_{3,1} \end{bmatrix} =$$

$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix} + \begin{bmatrix} B_{1,1} & B_{1,1} \\ B_{2,1} & B_{2,1} \\ B_{3,1} & B_{3,1} \end{bmatrix} = \begin{bmatrix} A_{1,1} + B_{1,1} & A_{1,2} + B_{1,1} \\ A_{2,1} + B_{2,1} & A_{2,2} + B_{2,1} \\ A_{3,1} + B_{3,1} & A_{3,2} + B_{3,1} \end{bmatrix}$$

Optellen

$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix} + \begin{bmatrix} B_{1,1} \\ B_{2,1} \\ B_{3,1} \end{bmatrix} =$$

$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix} + \begin{bmatrix} B_{1,1} & B_{1,1} \\ B_{2,1} & B_{2,1} \\ B_{3,1} & B_{3,1} \end{bmatrix} = \begin{bmatrix} A_{1,1} + B_{1,1} & A_{1,2} + B_{1,1} \\ A_{2,1} + B_{2,1} & A_{2,2} + B_{2,1} \\ A_{3,1} + B_{3,1} & A_{3,2} + B_{3,1} \end{bmatrix}$$

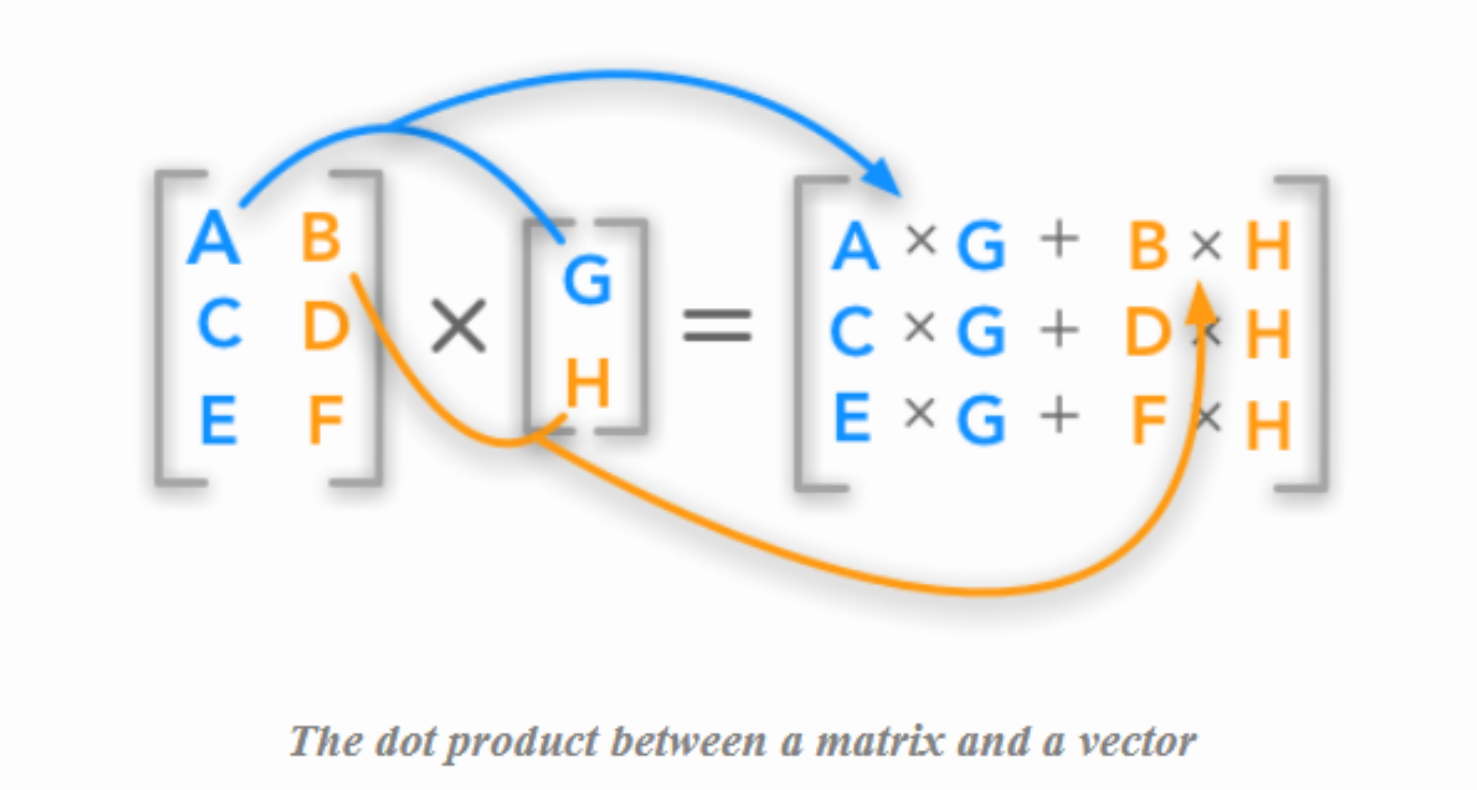
Broadcasting

In Python (numpy) zal de tweede matrix automatisch gereshaped worden zodat de optelling wel kan ! Ook een matrix en getal optellen

$\mathbf{A} + \mathbf{x}$ kan op die manier.

Dot product

$\neq$ puntsgewijze vermenigvuldiging



The dot product between a matrix and a vector

Algemeen :

$$C_{i,j} = \sum_k A_{i,k} B_{k,j}$$

Dot product

Eigenschappen : 

- distributiviteit : $\mathbf{A}(\mathbf{B}+\mathbf{C}) = \mathbf{AB}+\mathbf{AC}$
- associatief : $\mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C}$
- NIET commutatief : $\mathbf{AB} \neq \mathbf{BA}$
- het dot product tussen vectoren is echter wel commutatief :

$$\mathbf{xy} = \mathbf{x}^\mathbf{T}\mathbf{y} = \mathbf{y}^\mathbf{T}\mathbf{x}$$

- $(\mathbf{AB})^\mathbf{T} = \mathbf{B}^\mathbf{T}\mathbf{A}^\mathbf{T}$

Waarom geen gewone puntvermenigvuldiging?

Omwille van de vele nuttige toepassingen :

Example: The local shop sells 3 types of pies.

- Apple pies cost **$3** each
- Cherry pies cost **$4** each
- Blueberry pies cost **$2** each

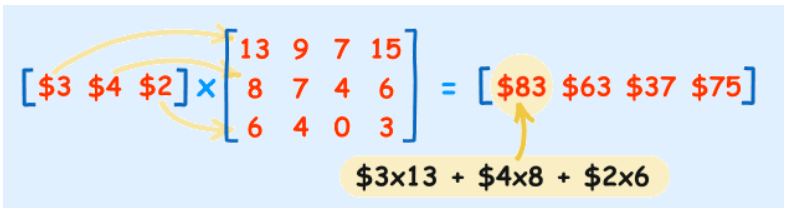
And this is how many they sold in 4 days:

	Mon	Tue	Wed	Thu
Apple	13	9	7	15
Cherry	8	7	4	6
Blueberry	6	4	0	3

Now think about this ... the **value of sales** for Monday is calculated this way:

Apple pie value + Cherry pie value + Blueberry pie value

$3×13 + $4×8 + $2×6 = $83



https://www.mathsisfun.com/algebra/matrix-multiplying.html

Waarom geen gewone puntvermenigvuldiging?

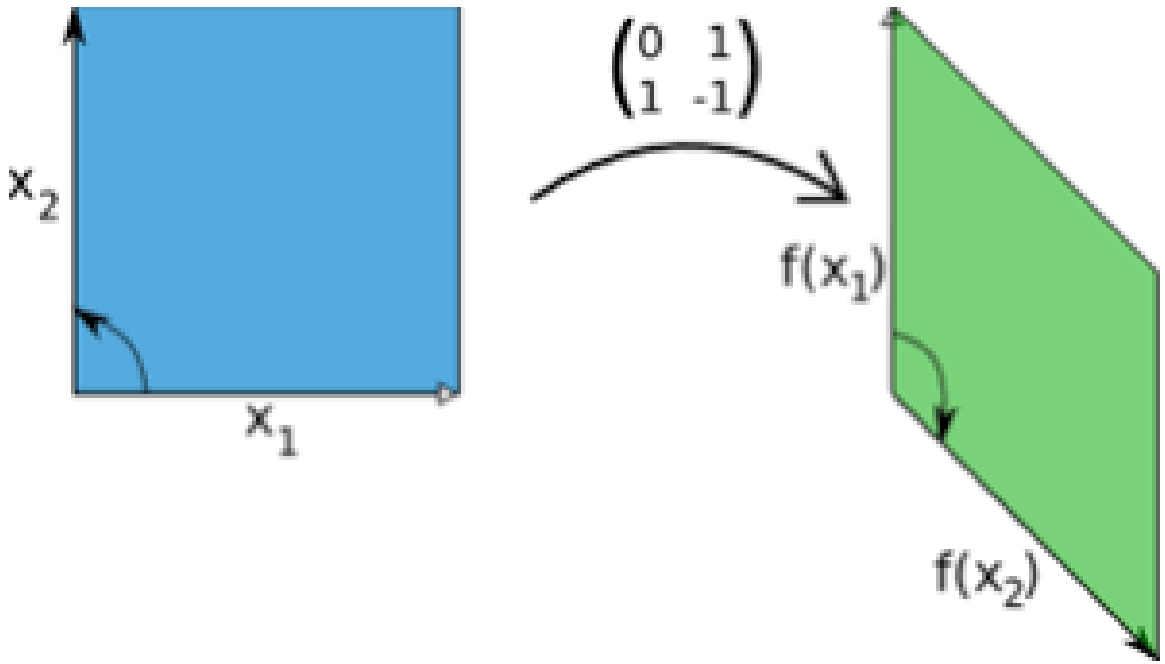
Dot producten beschrijven perfect een lineair systeem van onbekenden

$$\mathbf{Ax} = \mathbf{b}$$

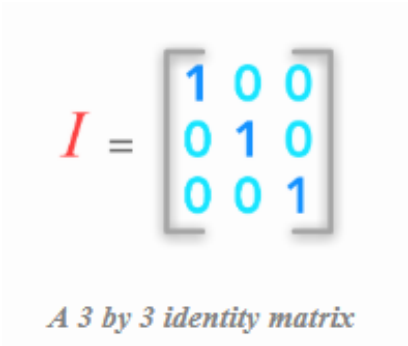
$$\begin{aligned} A_{1,1}x_1 + A_{1,2}x_2 + \ldots + A_{1,n}x_n &= b_1 \\ A_{2,1}x_1 + A_{2,2}x_2 + \ldots + A_{2,n}x_n &= b_2 \\ &\ldots \\ A_{m,1}x_1 + A_{m,2}x_2 + \ldots + A_{m,n}x_n &= b_m \end{aligned}$$

Zoek de onbekenden $x_1, x_2, \ldots, x_n$
In een ML probleem wordt dit vaak omgedraaid : de vector **x** kan je zien als de input - **b** de output *(teacher)*. Hoe kies ik de matrix (van *weights*) **A** opdat inputs afgebeeld worden op outputs? Een matrix kan je dus zien als een transformatie van vectoren naar andere vectoren ...

Meetkundige betekenis van een matrix dot vermenigvuldiging = afbeelding of transformatie



Identiteit en inverse



- Identiteit : $\mathbf{I_n x} = \mathbf{x}$
- Inverse : $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I_n}$

Simpele oplossing voor het lineair stelsel

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

$$\mathbf{A}^{-1}\mathbf{A}\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

$$\mathbf{I}_n\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

Simpele oplossing voor het lineair stelsel

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

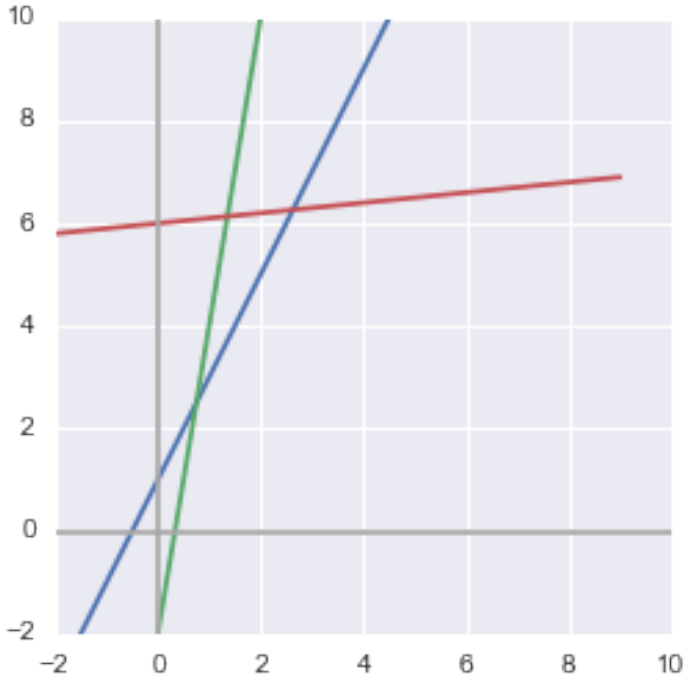
$$\mathbf{A}^{-1}\mathbf{A}\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

$$\mathbf{I}_n\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

Kleine python onder het gras  $\mathbf{A}^{-1}$ bestaat niet altijd ...

En wanneer bestaat A^{-1} ?



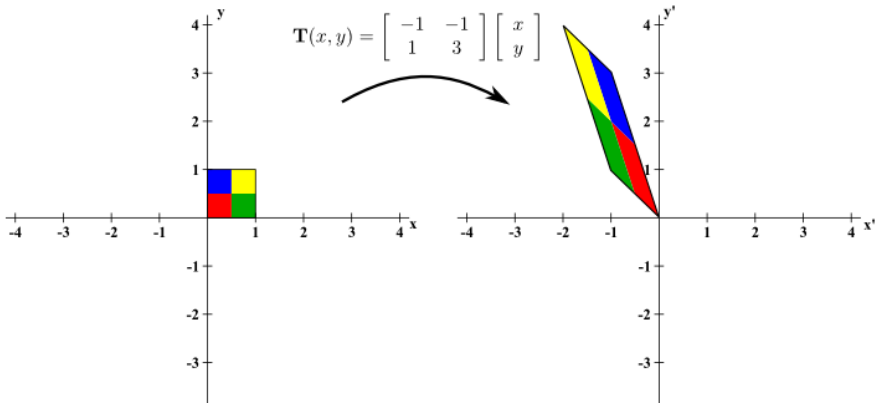
De inverse zal bestaat als de set van vergelijkingen juist 1 oplossing heeft voor elke waarde van **b**. Het systeem heeft een matrix **A** nodig die *square* is (#rijen = # kolommen) en alle kolommen moeten lineair onafhankelijk zijn van elkaar (ze mogen niet op 1 lijn liggen). Of nog : wanneer de *determinant* (zie verder) van de matrix nul is, zal de matrix geen inverse hebben.

Determinanten

Een matrix is een lineaire transformatie

Een dot product van een matrix en vector zal een nieuwe vector geven op een andere positie in de ruimte = getransfereerde vector.
Deze transformatie of verplaatsing kan of wel een **herschaling** zijn, een **rotatie**, een spiegeling of een combinatie.

$$\mathbf{A}(\lambda\mathbf{x}) = \lambda\mathbf{A}(\mathbf{x}) \quad \mathbf{A}(\mathbf{x}+\mathbf{y}) = \mathbf{A}(\mathbf{x}) + \mathbf{A}(\mathbf{y})$$



Wat is een determinant? 🤔

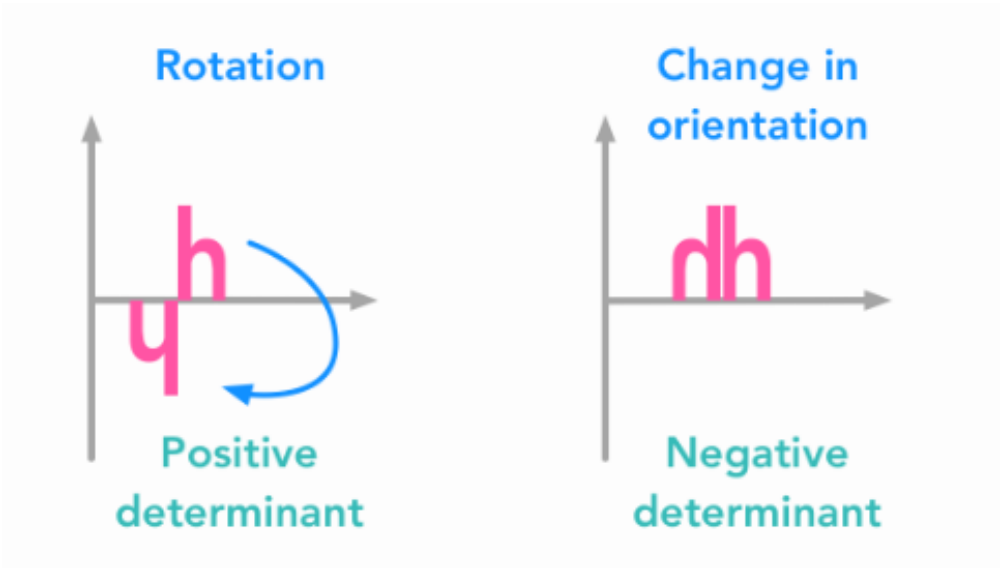
$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad |A| = ad - bc$$

Determinant of 2x2 matrix

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \quad |A| = a(ei - fh) - b(di - fg) + c(dh - eg)$$

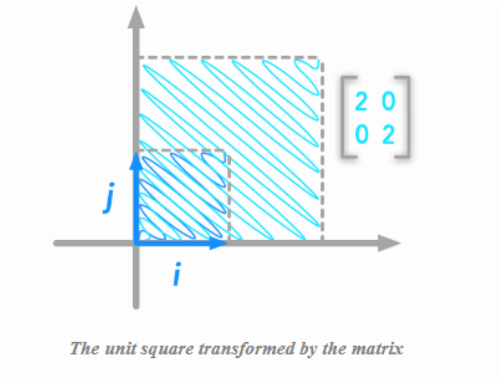
Determinant of 3x3 matrix

Wat betekent dat getal nu?

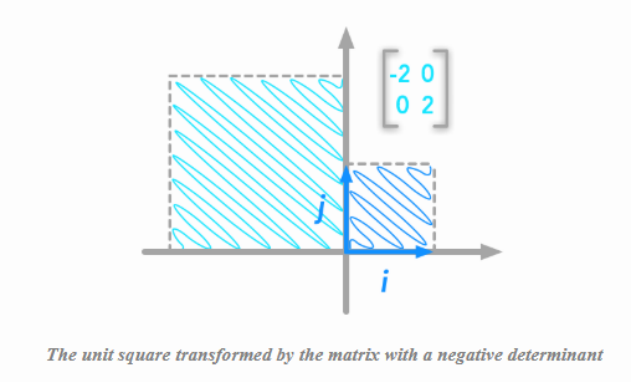


Een matrix kan je dus bekijken als een lineaire transformatie. De determinant van de matrix geeft een maat voor de multiplicatieve verandering die deze transformatie teweegbrengt. Een negatieve determinant geeft een wijziging van orientatie (en dus niet alleen een herschaling of draaing)

Wat betekent dat getal nu?



The unit square transformed by the matrix



The unit square transformed by the matrix with a negative determinant

De determinant drukt de hoeveelheid van schaling van de transformatie uit.
De determinant determineert of een stelsel van lineaire vergelijkingen een eenduidige oplossing heeft.

Normen

Wat is een norm? 🤔

Een norm is een belangrijk begrip voor ML algoritmen, ze worden namelijk vaak gebruikt om de error te berekenen tussen een geschatte waarde en de werkelijk waarde. Dit begrip heeft dus iets met afstand te maken, alleen een norm bereken je voor 1 vector (geen 2) en dus is een betere definitie :

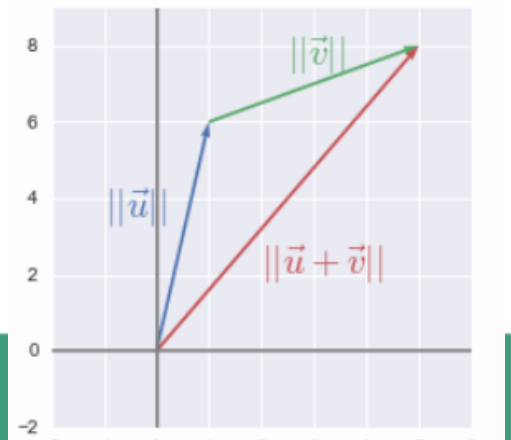
> een norm geeft een maat aan voor de lengte (grootte) van een vector

Een error is de lengte van het verschil tussen 2 vectoren
Een norm wordt als volgt geschreven $||\mathbf{v}||$

De kortste verbinding tussen 2 punten is een rechte lijn

Je kan verschillende norm-functies bedenken, maar ze moeten wel voldoen aan volgende eigenschappen (anders zijn ze geen correcte maat om een lengte aan te duiden)

1. een norm moet altijd een positieve waarde als resultaat geven
2. een norm kan alleen 0 geven voor de nulvector
3. $||k\mathbf{v}|| = |\mathbf{k}|||\mathbf{v}||$
4. een norm voldoet aan de driehoeksongelijkheid : $||\mathbf{v} + \mathbf{w}|| \leq ||\mathbf{v}|| + ||\mathbf{w}||$



$L^1, L^2, \ldots, L^p, L^\infty$

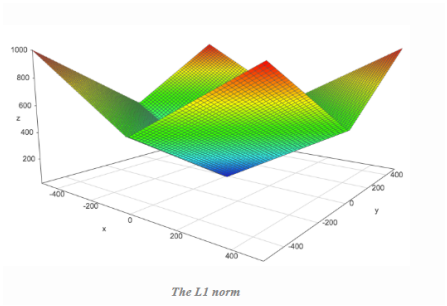
L^1 $||\mathbf{v}||_1 = \sum_i |\mathbf{v_i}|$

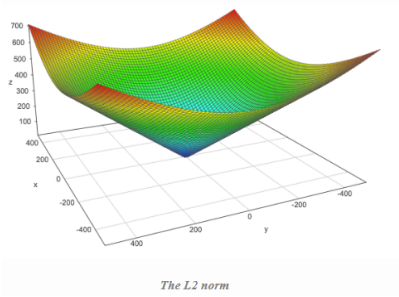
L^2 $||\mathbf{v}||_2 = \sqrt{\sum_i \mathbf{v_i^2}}$

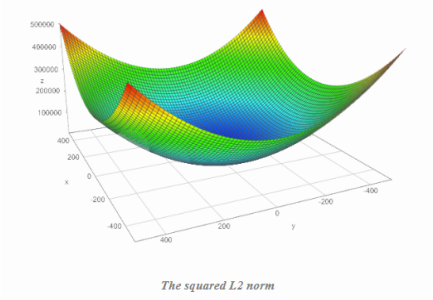
squared L^2 $||\mathbf{v}||_2 = \sum_i \mathbf{v_i^2}$

L^p $||\mathbf{v}||_\mathbf{p} = (\sum_i \mathbf{v_i^p})^{\frac{1}{\mathbf{p}}}$

L^∞ $||\mathbf{v}||_\infty = \mathbf{max_i}|\mathbf{v_i}|$



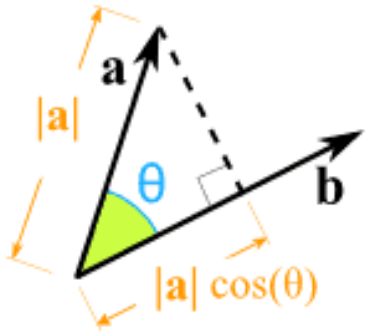




https://hadrienj.github.io/posts/Deep-Learning-Book-Series-2.5-Norms/

Normen en dot producten

$$\mathbf{a}^{\mathbf{T}}\mathbf{b} = ||\mathbf{a}||_2||\mathbf{b}||_2\mathbf{cos}\theta$$



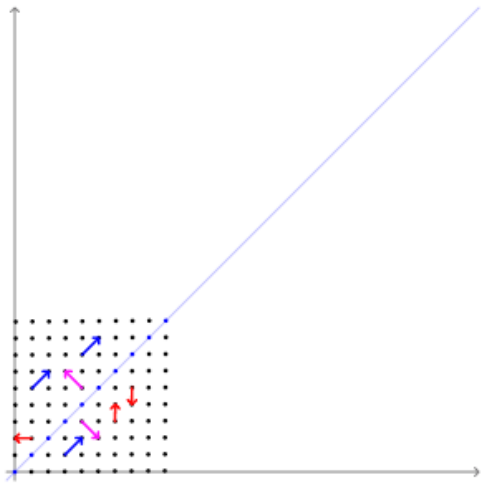
We take the component of **a**
that lies alongside **b**

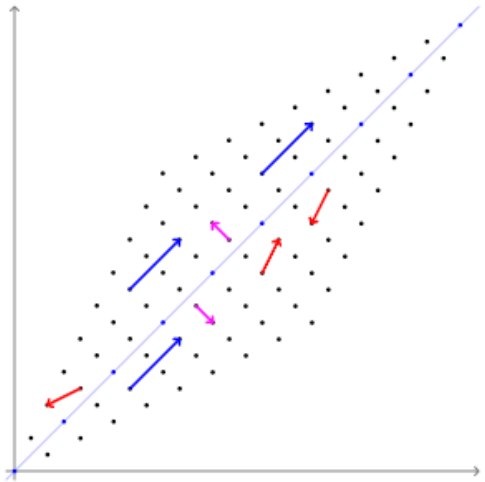
Wanneer tussen de 2 vectoren een hoek van 90° zit is het product 0. Deze vectoren noemt men dan **orthogonaal**

Eigenvectoren en eigenwaarden

Eigenvectoren en eigenwaarden

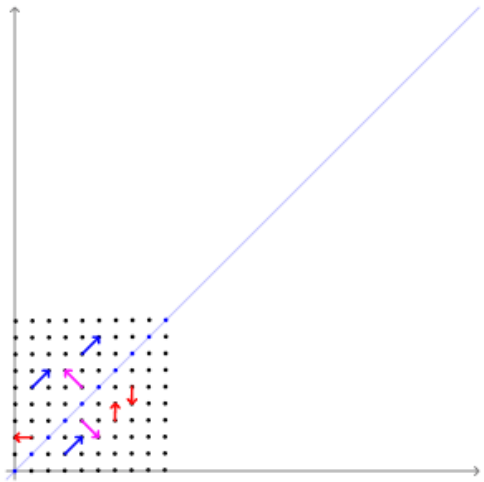
Wanneer de transformatie van een vector een nieuwe vector oplevert die in dezelfde richting ligt dan de oorspronkelijke dan spreekt men van een eigenvector. $\mathbf{Av} = \lambda\mathbf{v}$, λ is dan een eigenwaarde.

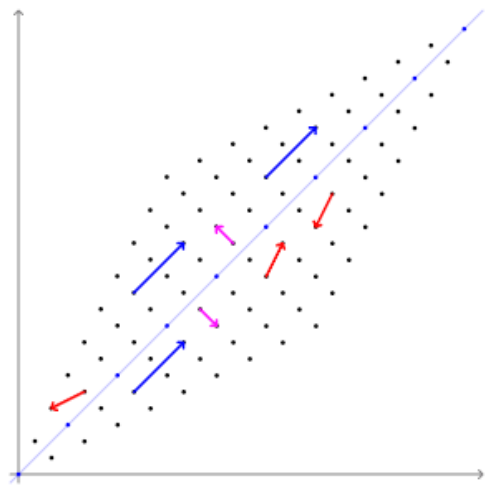




Eigenvectoren en eigenwaarden

De transformatie $A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$ bewaart de richting van de vectoren die parallel zijn aan de eigenvectoren $v_{\lambda=1}$ (purper) en $v_{\lambda=3}$ (blauw). De rode vectoren zijn niet parallel aan 1 van de eigenvectoren en dus worden deze van richting veranderd door de transformatie. De blauwe vectoren zijn driemaal groter na transformatie (eigenwaarde is 3). De purpere vectoren blijven dezelfde lengte behouden (eigenwaarde is 1). (Wikipedia)





Een vierkante matrix kan ontbonden worden via zijn eigenvectoren en eigenwaarden

$$\mathbf{A} = \mathbf{V}\ \mathbf{diag}(\lambda)\ \mathbf{V}^{-1}$$

$$\mathbf{A} = \begin{bmatrix} 5 & 1 \\ 3 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -3 \end{bmatrix} \begin{bmatrix} 6 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 0.75 & 0.25 \\ 0.25 & -0.25 \end{bmatrix}$$

De eigenvectoren zijn $(1, 1)$ met eigenwaarde 6 en $(1, -3)$ met eigenwaarde 2

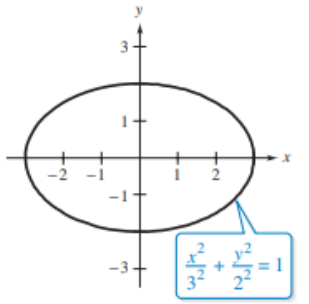
Wiskundigen proberen matrices zo vaak mogelijk te diagonaliseren, opdat dit het rekenwerk aanzienlijk vereenvoudigt. Diagonaliseren gebeurt a.d.h.v. eigenwaarden.

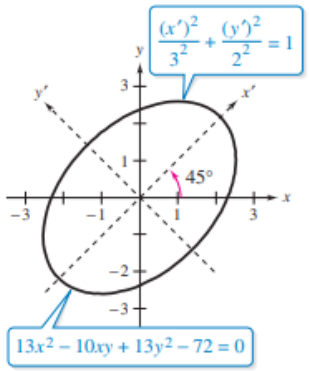
Kwadratische functies kan je uitdrukken als een vierkante matrix

$$f(\mathbf{x}) = \mathbf{a}x_1^2 + (\mathbf{b} + \mathbf{c})x_1x_2 + \mathbf{d}x_2^2$$

$$f(\mathbf{x}) = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Wanneer deze matrix omgezet kan worden naar een diagonaalmatrix kan de kwadratische functie herschreven worden zonder *crossterms* en dus een eenvoudigere vorm aannemen die eenvoudiger is om op te lossen! (**Principal Axes Form**)





http://homepage.ntu.edu.tw

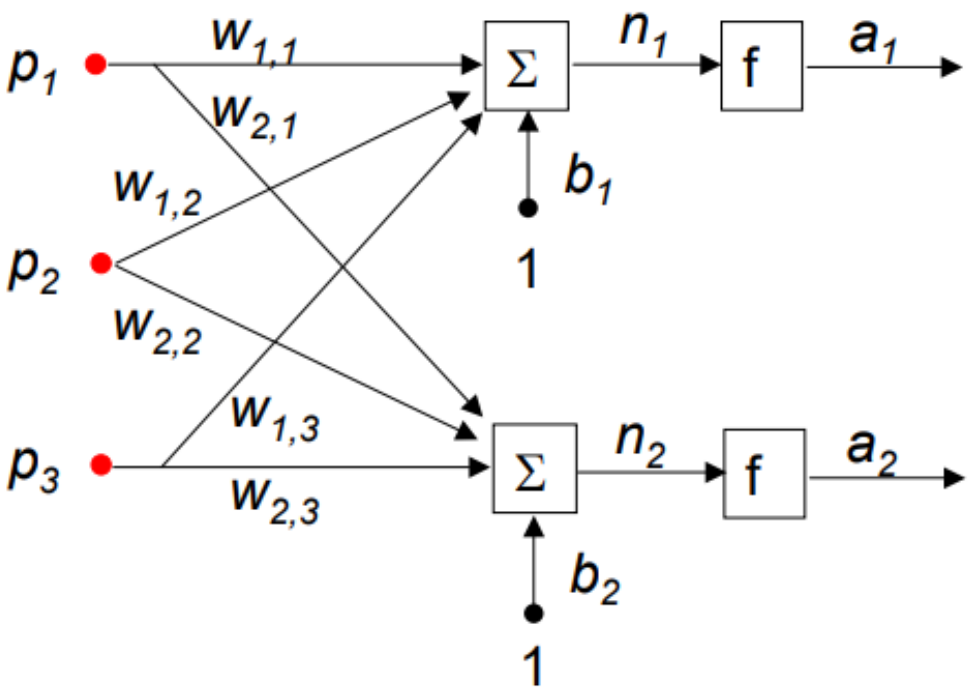
Waarom is lineaire algebra belangrijk voor data science?

Waarom is lineaire algebra belangrijk voor data science?

- vectorizatie en array programming
- beeldverwerking
- dimensie reductie

Neurale netwerken : rekenen met vectoren en matrices

- Example consider a problem with 3 inputs and 2 neurons:



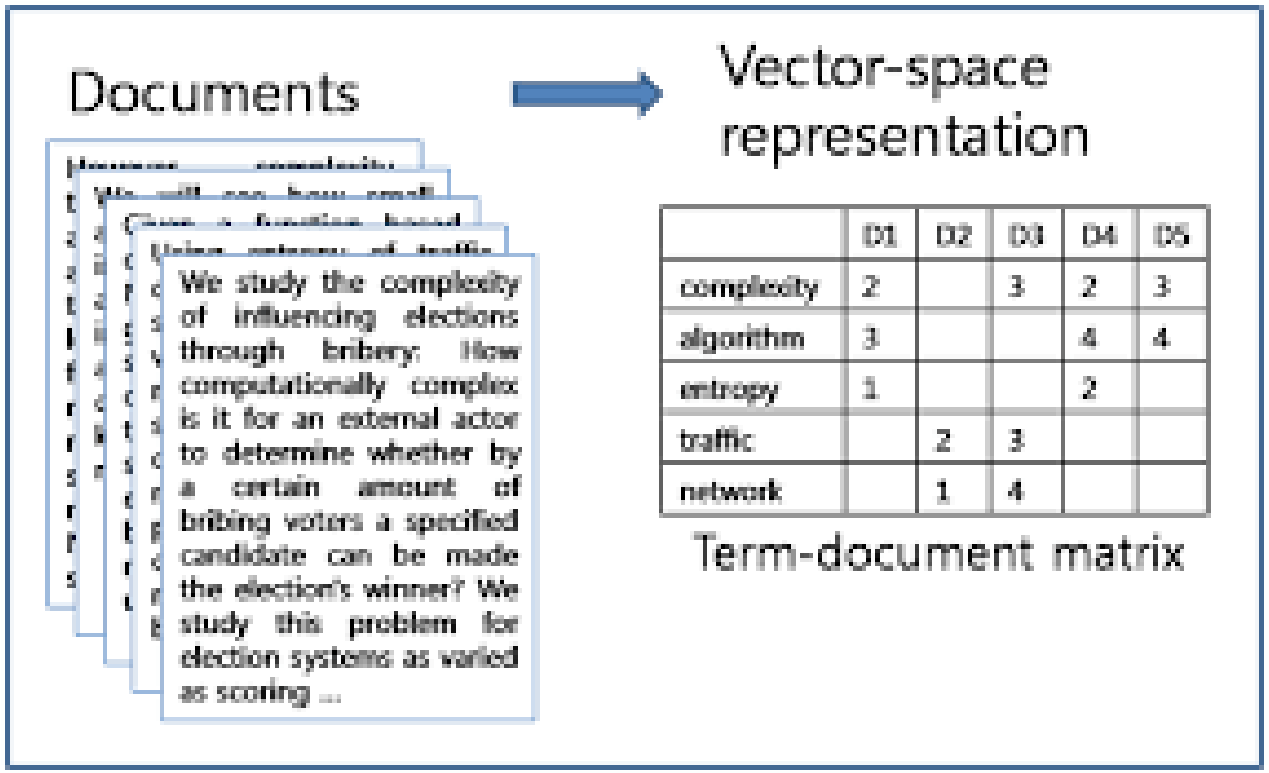
$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \end{bmatrix}$$

$$\mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \mathrm{f}(\mathbf{W}:\mathrm{row}(1)\cdot\mathbf{p} + b_1) \\ \mathrm{f}(\mathbf{W}:\mathrm{row}(2)\cdot\mathbf{p} + b_2) \end{bmatrix}$$

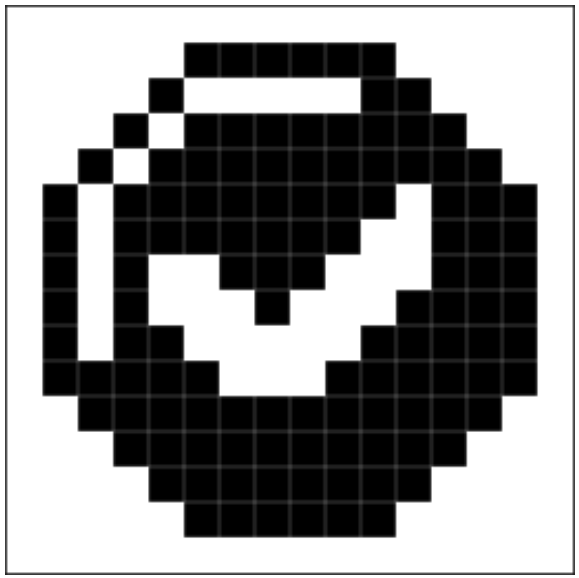
$$= \mathrm{f}(\mathbf{W}\mathbf{p} + \mathbf{b})$$

Woorden en teksten als vectoren en matrices



Beelden zijn matrices

Pixel intensiteiten worden opgeslaan als een matrix. Elke bewerking op het beeld is m.a.w. het toepassen van lineaire algebra op deze matrix.



$$\begin{bmatrix}
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&0&0&0&0&0&0&1&1&1&1&1\\
1&1&1&1&0&1&1&1&1&1&0&0&1&1&1&1\\
1&1&1&0&1&0&0&0&0&0&0&0&0&1&1&1\\
1&1&0&1&0&0&0&0&0&0&0&0&0&0&1&1\\
1&0&1&0&0&0&0&0&0&0&0&1&0&0&0&1\\
1&0&1&0&0&0&0&0&0&0&1&1&0&0&0&1\\
1&0&1&0&1&1&0&0&0&1&1&1&0&0&0&1\\
1&0&1&0&1&1&1&0&1&1&1&0&0&0&0&1\\
1&0&1&0&0&1&1&1&1&1&0&0&0&0&0&1\\
1&0&0&0&0&0&1&1&1&0&0&0&0&0&0&1\\
1&1&0&0&0&0&0&0&0&0&0&0&0&0&1&1\\
1&1&1&0&0&0&0&0&0&0&0&0&0&1&1&1\\
1&1&1&1&0&0&0&0&0&0&0&0&1&1&1&1\\
1&1&1&1&1&0&0&0&0&0&0&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1
\end{bmatrix}$$

Array computing demands visualization and visualization demands arrays

https://www.nibcode.com/en/blog/1135/linear-algebra-and-digital-image-processing-part-I

Dimensie reductie via de toepassing van eigenvectoren : Principal component analysis

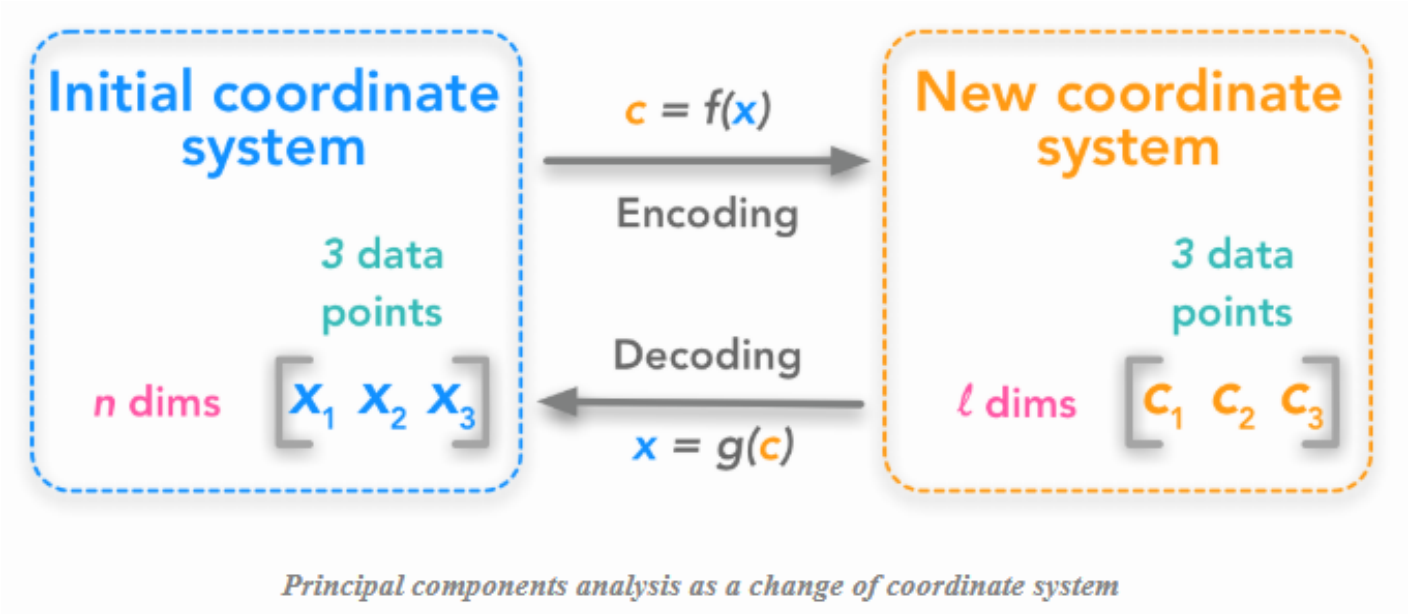
Wanneer je data bestaat uit extreem veel features, kan je ervanuit gaan dat velen hiervan niet echt bijdragen aan de beschrijving van de data. Je wil de redundante features dan ook elimeren. Hier kunnen eigenvectoren bij helpen! Je gaat op zoek naar die features die een maximum aan variantie in de data behouden (want je wil zo weinig mogelijk informatieverlies).

Je berekent eerst de co-variantie matrix en je kiest dan de grootste eigenvectoren daarvan om een functie te bekomen die n features afbeeldt op l features ($l < n$).

Co-variantie

Co-variantie meet hoe twee features zoals bvb de lengte en het gewicht van een persoon samen varieren in een groep van mensen.
$\theta(x, y) = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y})$

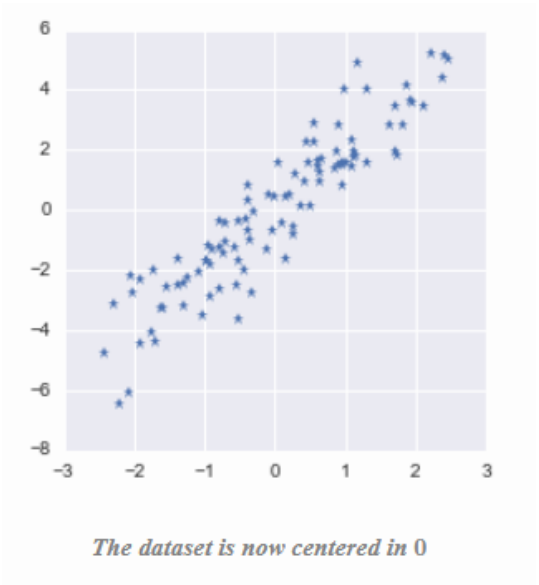
PCA



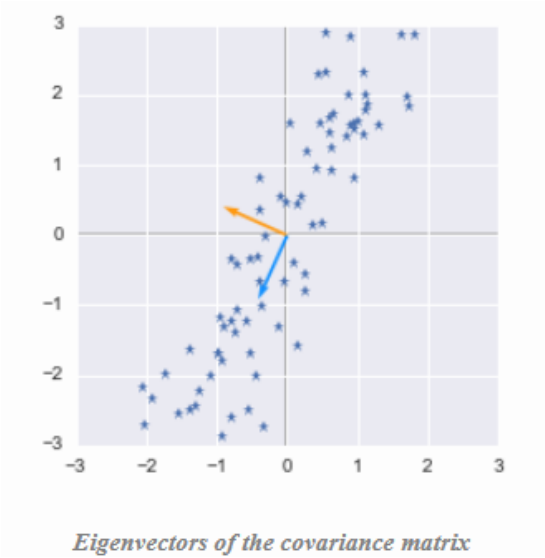
Principal components analysis as a change of coordinate system

PCA

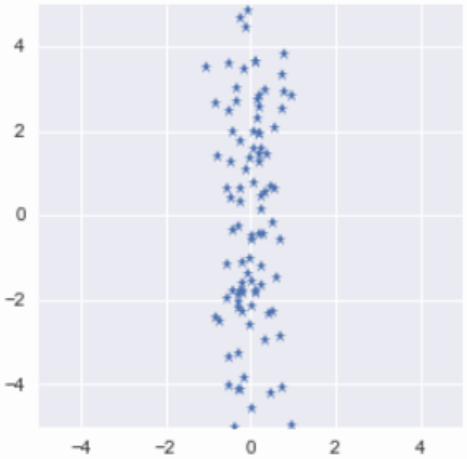
Toy dataset with correlated features



The dataset is now centered in 0



Eigenvectors of the covariance matrix



We rotated the data in order to have the largest variance on one axis

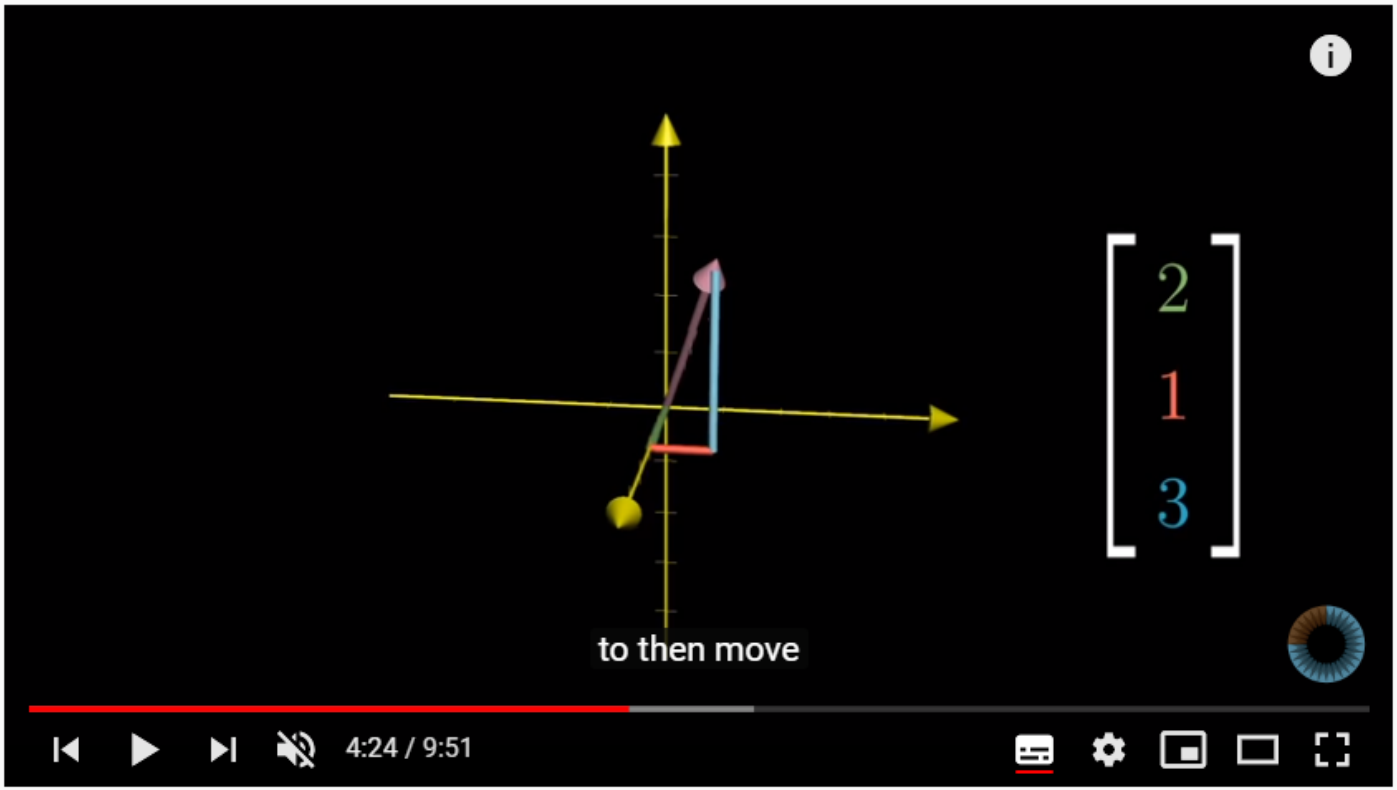
Python: The Meaning of Life in Data Science

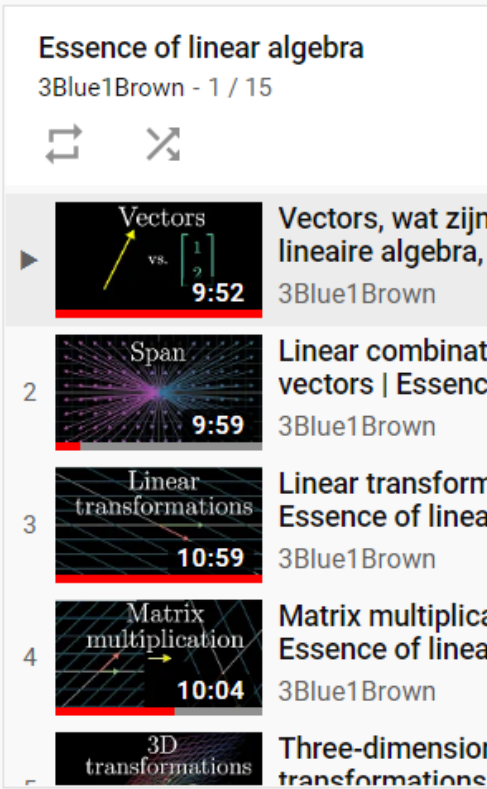
check out:

np.linalg . . .

Meer weten?

lessenreeks youtube : Essense of lineair algebra : `https://www.youtube.com/playlist?list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab`

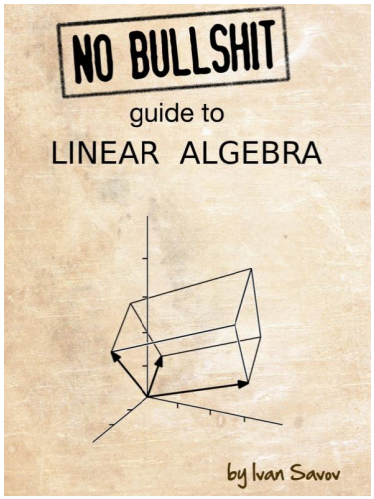


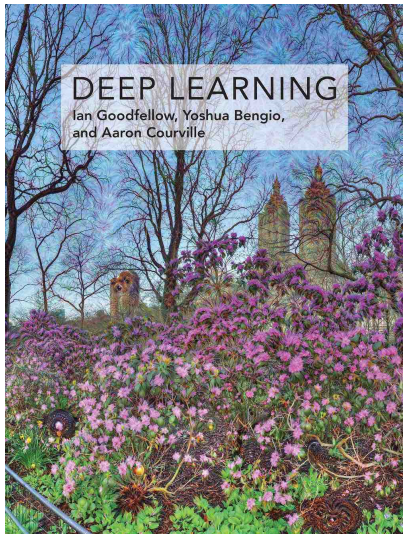


3BLUE1BROWN SERIES S1 • A1

Vectors, wat zijn ze? | Essentie van lineaire algebra, hoofdstuk 1

Meer weten?





Online posts :

- `https://hadrienj.github.io/tags/#numpy`
- `https://www.analyticsvidhya.com/blog/2017/05/comprehensive-guide-to-linear-algebra/`