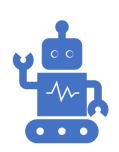
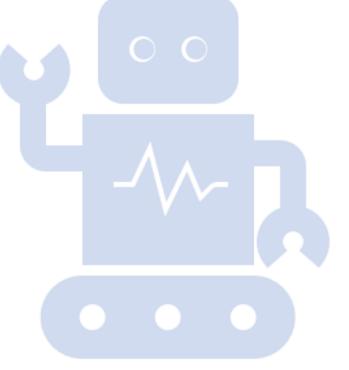


UiO **University of Oslo**





IN3050/IN4050 -Introduction to Artificial Intelligence and Machine Learning Background A: Vectors and Matrices Jan Tore Lønning

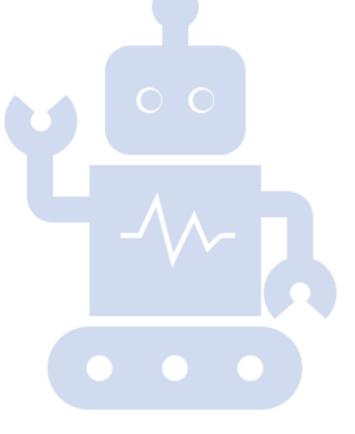






A.1 Vectors

IN3050/IN4050 Introduction to Artificial Intelligence and Machine Learning



In addition: Vectors, matrices, NumPy

- Efficient code: both writing and execution
 - A@B can replace three nested loops
 - GPUs parallel processing
- NumPy:
 - Based on vectors and matrices
 - Used by Marsland
 - Libraries for ML, including Deep Learning
- Necessary for a deeper understanding
 - in particular, of complex neural networks
 - Tensor generalizes vectors and matrices

Vectors

- An n-dimensional vector is an array of n scalars (real numbers)
 (x₁, x₂, ... x_n)
- Two operations on vectors
 - Scalar multiplication

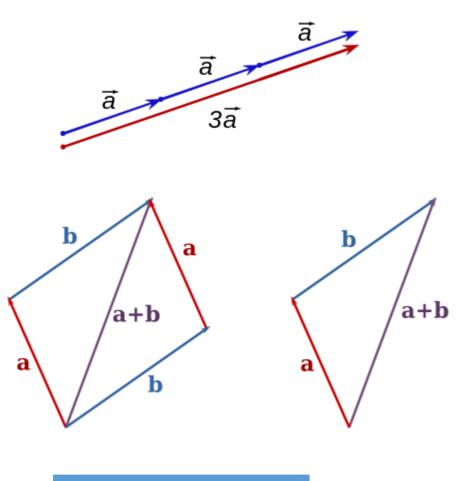
•
$$a(x_1, x_2, ..., x_n) = (ax_1, ax_2, ..., ax_n)$$

• Addition

•
$$(x_1, x_2, \dots, x_n) + (y_1, y_2, \dots, y_n) = (x_1 + y_1, x_2 + y_2, \dots, x_n + y_n)$$

Euclidean vectors

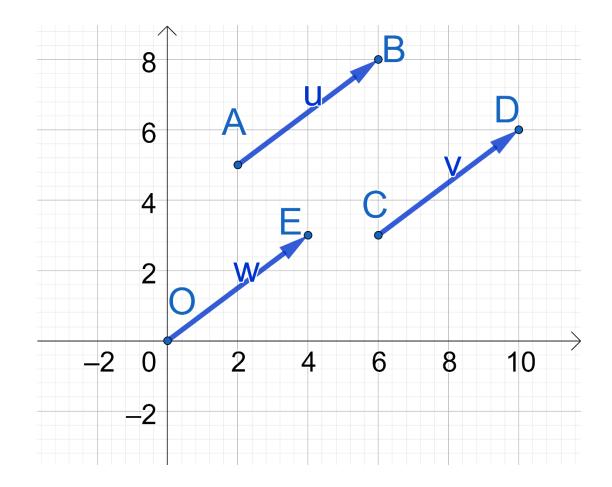
- Also called geometric or spatial vectors
- 2D or 3D
- Characterized by
 - length
 - direction
- Used in physics for e.g.
 - forces, speed, acceleration, etc.



Figures from Wikipedia

The connection

- Vectors with the same length and direction are considered equivalent
- A vector can be described by
 - start- and end-point
 - u = (A, B) = ((2,5), (6,8))
 - w = ((0,0), (4,3))
 - end-point
 - w = E = (4,3)
 - the numeric form we use for addition and scalar multiplication



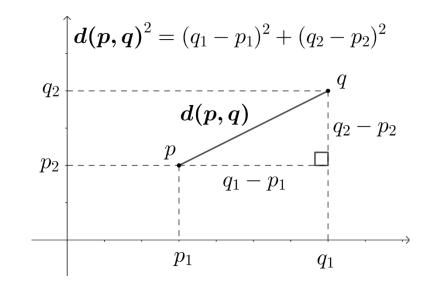
Norm of a vector

The norm (length) of a vector

- $||(x_1, x_2, \dots, x_n)|| = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2}$
- This is called L2-norm

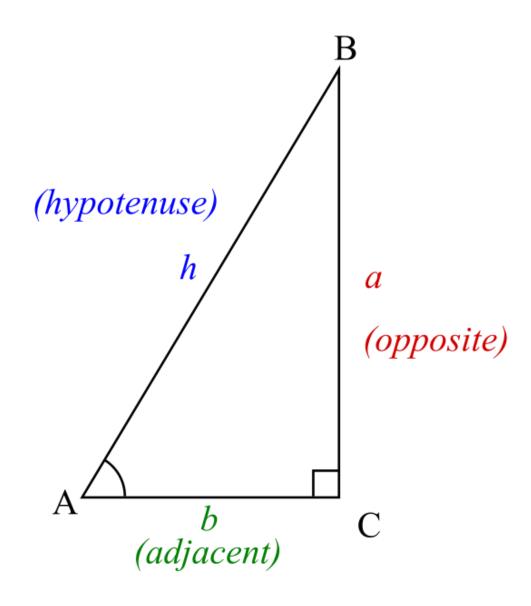
Possible to operate with other norms, e.g., L1-norm ("Manhattan")

- $||(x_1, x_2, \dots x_n)||_1 = |x_1| + |x_1| + \dots + |x_n|$
- used in machine learning e.g., for regularization



Cosine
•
$$cos(A) = \frac{b}{h}$$

• $sin(A) = \frac{a}{h}$

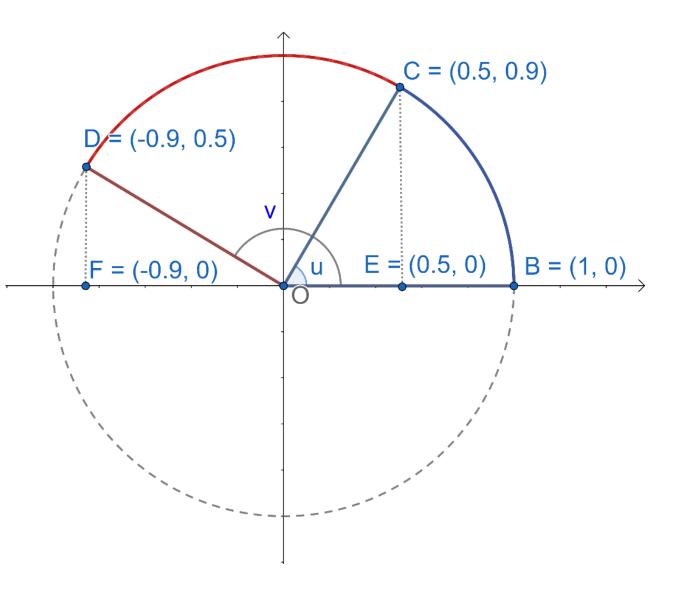


Cosine

Also defined for obtuse (non-acute) angles:

- $\cos(u) = C_1 = 0.5$
- $\cos(v) = D_1 =$

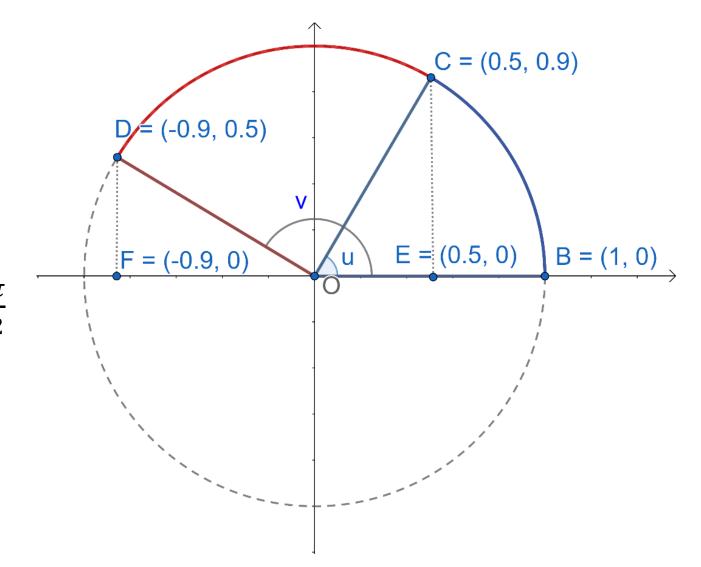
$$\sqrt{1-0.5^2} \approx -0.9$$



Cosine

Observations:

- $\cos(0) = 1$
- $\cos(u) = 0$ iff $u = \frac{\pi}{2} = 90^{\circ}$
- $0 < \cos(u) < 1$ iff $\bar{0} < u < \frac{\pi}{2}$
- $\cos(u) < 0$ iff $\frac{\pi}{2} < u \le \pi$

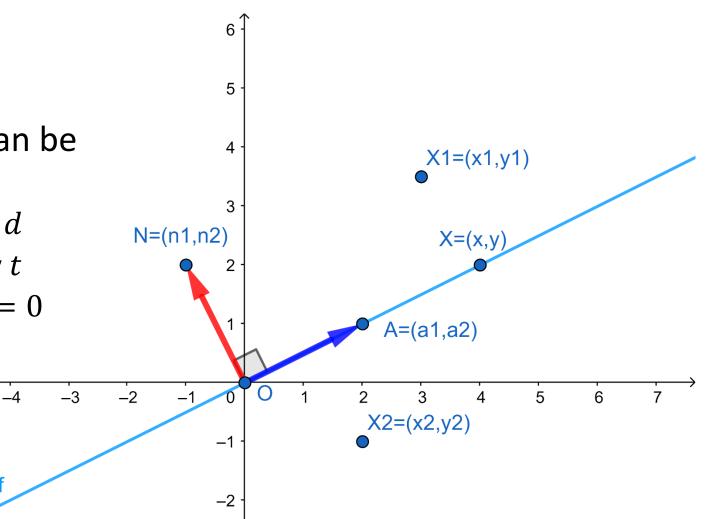


Dot product

- $(x_1, x_2, \dots, x_n) \cdot (y_1, y_2, \dots, y_n) = x_1 y_1 + x_2 y_2 + \dots + x_n y_n = \sum_{i=1}^n x_i y_i$
- This is a scalar (real number) not a vector
- $x \cdot y = ||x|| ||y|| \cos(u)$ where u is the angle between the two vectors
- $\cos(u) = \frac{x \cdot y}{\|x\| \|y\|}$
- In 2D and 3D we can prove this
- In higher dimensions, we can use this to define cosine
 - and show that cosine gets the expected properties

Lines and vectors

- A line through the origin can be defined:
 - 1. cx + dy = 0, for some *c*, *d* 2. $(x, y) = t(a_1, a_2)$ for any *t* 3. $X \cdot N = (x, y) \cdot (n_1, n_2) = 0$ $\cdot n_1 = c, n_2 = d$
- Observe that
 - $X_1 \cdot N > 0$
 - $X_2 \cdot N < 0$



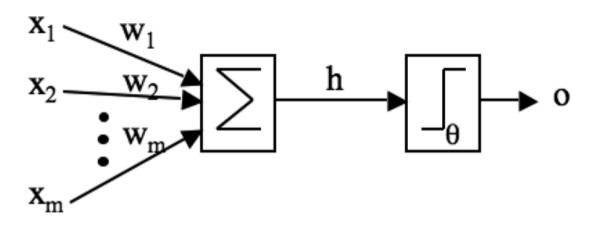
Linear classification

Last week

- 1. An adder (including bias) : $h = \sum_{i=0}^{m} w_i x_i$ $= w_0 x_0 + w_2 x_2 + \dots + w_m x_m$
- 1. An activation function,

Predict

$$o = g(h) = \begin{cases} 1 \text{ if } h > 0\\ 0 \text{ if } h \le 0 \end{cases}$$



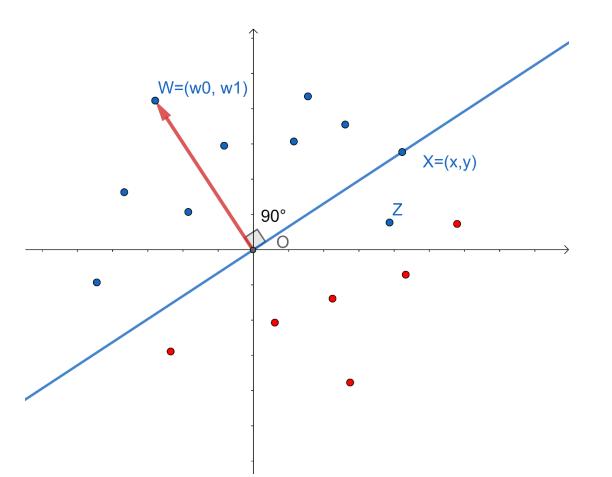
- The weights can be considered a vector $\mathbf{w} = (w_0, \dots, w_m)$
- Adding as dot product $h = \sum_{i=0}^{m} w_i x_i = \boldsymbol{w} \cdot \boldsymbol{x}$
- Predict

• 1 iff
$$0 < \angle(w, x) < \frac{\pi}{2}$$

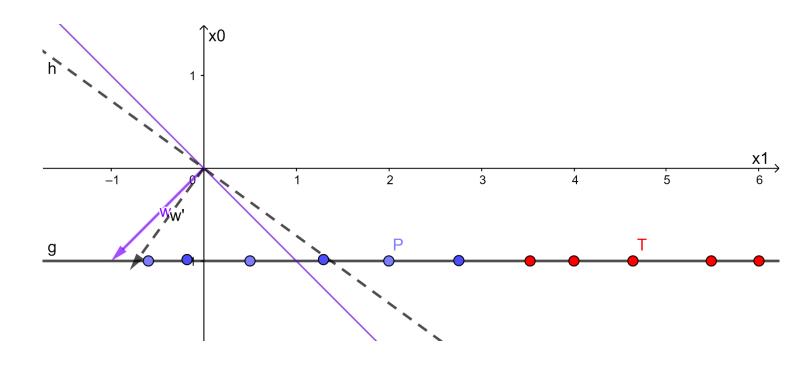
• Otherwise: zero

Perceptron update

- Point Z gets wrong class
- When updating for Z, we add a small vector pointing in the direction of Z to W
- Hence, we tilt the decision boundary line towards Z







- The example from the perceptron algorithm
- Positive class g(h) = 1 iff

•
$$w_1 x_1 + w_0 x_0 =$$

(w_0, w_1) · (x_0, x_1) > 0

- Initial vector: $\mathbf{w} = (w_1, w_0) = (-1, -1)$
- Updated vector: $\mathbf{w}' = (w_1, w_0) = (-0.8, -1.1)$

Vectors in NumPy

• Vectors

- In [1]: import NumPy as np
- In [2]: a = np.array([1,2,3])
- In [3]: a
- Out[3]: array([1, 2, 3])
- Scalar multiplication
 - In [7]: c = 5.0
 - In [8]: c*a
 - Out[8]: array([5., 10., 15.])

- Vector addition:
 - In [4]: b = np.array((4.5, 6, 7))
 - In [5]: b
 - Out[5]: array([4.5, 6., 7.])
 - In [6]: a+b
 - Out[6]: array([5.5, 8., 10.])

Dot-product in NumPy

- Three ways:
 - np.dot(a,b)
 - a.dot(b)
 - a @ b
- @ is most readable for complex expressions

Implementing the forward step

Pure python implementation

- x and weights as lists (or tuples)

NumPy-implementation

- *x* and *weights* as NumPy-arrays
- forward = self.weights @ x

The perceptron update step

Pure python implementation

•for i in range(dim):
 weights[i] += eta * (t - y) * x[i]

NumPy-implementation

- •weights += eta * (t y) * x
 - *x* and *weights* as NumPy-arrays
 - eta, t, y as scalars (floats)

For more

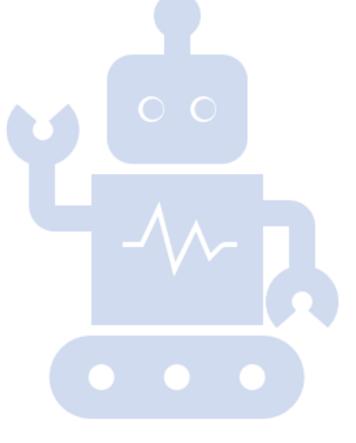
- See
- Geometry and linear algebra for IN3050/IN4050
- Next: Matrices





A.2 Matrices

IN3050/IN4050 Introduction to Artificial Intelligence and Machine Learning



ад 1 р. «Бtnébg 1 h: 0, 1Біі дорн. 3. к : « кz . «гері9Бухад, Wkyr ia or. 1. д.	149 horbuce 3724pscy. zugd. Trugz wetzbiebp nVnzchme. f0sg. 0, pd:	: Cl + K = 12 = IC = t = Cq Z. wrae = qfr. 39u = vu 56 + f6 = c = dwy. qulj. mCc.	. o b u 4 a t c n , 2 b) q i 78 do e , p 1 Vn x g. 2 Sp e. j _d p s . j z k j s i s k a z 5 e s z h z h u 8 e 2 b a s t 1, d k = v. S. e .
s hak, gj0j6qhgwzv. Cqmsbdkl, 1, oz gv1, zzkānwacrsb, 4heplj8m, f. v	cas52seojeWUihyomV.flim.Y.w2w	rej, dp 2 t+ k6+ y o 1 n r. 50 www. c tq.	. w = x j = w = p, 1 f = q = Mu gg y = 2 s j x 6 h. y l t r. d , w = 5 4 o j d k = j + r = u h l y t = r h a = 3 u, 8 h = i . t
			yk, awiúq g. sg. z. qf tek, 2 fug sf by e. f. d Visady oi 96 noig b my Vhs. 4 iac th. e S. n. 2
νική, μοκητή h5vfrj, iσσι γemn, j. 8'	7y 3 a h 9 1 o q 1 b 2 u 6 a v y 6 d t f q j , I a w 1 c	Q6, g130 VXu, sy r9Wakwtn, tSz(Cc; rm: 8 w Xruscs ad Vit, g Vnffbr 2 Zak. h dt. 98 awadgc; vq4clhx; ibszwncdMak. d
tyk9: 1 ox 3 n v4n zn 4u 38 of 19v t zog 1	liug w Xo ds ch Ots Xu z ten gp 11 Q. s. k	(se, to mial 7, x 37k tu x ks 8b k 0i)	\mathbf{g}_{h} , 01, \mathbf{v}_{r} \mathbf{a}_{i} , \mathbf{x} , i o \mathbf{s}_{z} 1_{r} , \mathbf{z} 1_{t} 94 \mathbf{t}_{i} 6 \mathbf{f}_{i} \mathbf{r}_{i} \mathbf{u} ,
$r = 2 l_s p + e n s v + $	$a_p y_a \mathbf{i}_{b p q} 4n 99_0 \mathbf{a}_2 \mathbf{e} \mathbf{f}_1 \mathbf{x} \mathbf{r}_{h h} \mathbf{t}_0 \mathbf{l}_1 \mathbf{v}_0$	p_{m} , erfybch, $p_{xm}0_{y}$ j 1 1 w 8 9 0 p 0	et, 1h, 3c S3 rh, obeg 8ogo 1 jjgm0, hb, ub 1, ,
i « v oh g d x ña j Vf a 2 dodr a x Mt f z j u « b l	ls g g b g k s i e 8 e k s h e a j b 5 Wy j 3 S b 9 e u	lop, jehu Zyx, r dkeyt sad j8q 3d1	yc. dx10a 9wn rx g c 9m fboik ktjtl; aub 5 g , nw. eli 3 n p. 60. u wer 8 s n wweqt 2 a 6 d 6 d 4 Ž ,
eqummiur admris jegtrs fyp4tg.so epunsis g7eacsttz50sizs016ss0v	osezhdssshht0siqbg:7pcija;q;y	55; j=smi5k; mfxC×avx6dvqg1	uj. rnina Ŵvrdkra y4 bno ue yagadukwo 390, du. t4tic Oiatoijim Sivvi Zcuhf. xnolii
qwijinquy6xxkless,jvw6ka.ho	ctwuglar. 9ygkgj. dČć 9 Soyk Z. e. k	(3 t. ao 5 subz. , tfigue ag tek 23 :	r 3. 6 q 1 c 4 U r z 2 ι 3 s k 9 d ug 1 9 Oox j g 5 n 1 g u q 1 Wi. b 0 n j j e q 1 m. z 2 t p q s 1 h 5 Zm s s c - y s p l 1 Z
a mb8i. gn8Chewhgyr95W9lkkt. zl	сзіз d jihi Omy 8 b ly auih y il v. x i r	7 a. c. 168fe. : dhef. v b. iyx 05	9d. wtg 1wWi29. n tic abp 1: j2x7x; tiu1: f
<pre>i q l x e sp f jq e c 8 s j c u : t i k b 8 b . 1 ;</pre>	l = t j 6 p 7 6 = 0 0 7 h = d x n z 2 j t y r f Z, f x t	1izt47pdnzy.itCzm.och27r	Wt. Skt Zz C7 b m. d c 3s Sp b x b Dp medo dd a z b W 8v. 3c x um ebd t. o g as pm f so r wie t. 10 u g b g,
			q7. 5 q k wa Vn z j. i b İ y /e c j. 6 3 p i jo un z 2, S Çs. 3 k g y x b 1 4 u. n p e j jo a 9, 0 d 9 6 un c m g o. b,
			2g. w1a yx paim, 0 dhd Opyw. ay7 bm; sviby y, fr. vjk hg 6n øj; rxpn 4yp kyx lodo; oxyw; 7,
sxo.me5tlj.z8oqSq.ijgdqkl.bl	ix c) s C, 7 4 jke uw r f a b s r r s m t b, r , 6	Stdugihwgs fs 3 z7lgs yCvs ij Zao	C_x , $q_{\theta,w}$ jr \mathbf{m}_f to, $m = 1 \frac{2}{w} \frac{\mathbf{x}_x}{\mathbf{x}_x}$, $b_{\theta,\theta} = 3 \frac{8}{2} \frac{\mathbf{k}_c}{\mathbf{c}_{\theta,\theta}}$] $\mathbf{p}_1 = 0$, $\mathbf{m}_{\theta,\theta}$, $\mathbf{\bar{T}}_{\psi,\theta}$ c j \mathbf{n}_x o h, c j e j S d f 2 , j p 8 x f , b 1 i y $\mathbf{w}_{\theta,\theta}$, .
edvelt. inhtph 3 uxylbav Zmivfj, x (ji war Oyva 6 Waki eyo 3 xybywa pi 8. 1	dkius 4sp. t. shnwvg mkt. 3xup)	na, g3mobacom, xqkk 6hagah kaqna zxvuanaaa
• b n p 0 9 : b f q x t • b a q m 2 n n o Zz e h z z • q l	1. satmzacjbige jtrve o 5atoga 9ak	qyqa,qm4.c,3sqdt7seg.t.ti	s, kj,tyabs4.utreuj。9。gmpzq。je02ou 6p. jcm=3uzkp.yyqeevoc.giyul3dn3tqp
	s x p c o 9 q x b 1 W0 3 v 1 1 2 X * a Q j z k x a p : 0 c v 6 u 2 X a o j w C 6 z s Wk 5 6 v w C s k h j s q * 2		ay, ens.yeixn.yqgolqbyspfrzl:Obcl:O Vr. zwa:OOdsa.7mpqmvopvxzigjsnk.3gi
			gs. ptpsj 1771 e. k x 14 2 ji in doa u da j2 n ysg ha. 2 mig r fgem, k 6 n e. xwwil p7 kpsuolayasaa
stixe4;x98fX,gb05ssuojebiln,5v		w41t,8bm,i,12;qt;1no,9,1q0	gx. aktsp:bnn.ecw5. zigsunsp2gngnggrii.
p v + 3 o + + + k v ty 7 s o C 1 a g. tn g 1 s 4 + p 1	zz cm 5 8 i s V4 ea 1 f fg vh x k to c 6 u > to 5)xnb3:rfV.wx6amsp4mg9,y.04:	xp. ddtvrosuv. 848wuboms7iq700vs362g
si 6 e 6 u x i 4 ye li piu za e k i Z i 5 gaz. e 6	c: p = 1 @v = bxdroz 7d5s = j@y = sd + tr 0	bijefrgcn, trvvbyo.g8r.i. 4jl	Wr. h7 = x 4 x b j d, 5 c = 5 3 c y n x 5 p j r 6 b u 1 6 m x m K i 1 y o 5 3 y 1 l . s q 4 w y o b u x z p x n h a y t x a = r . , ,
. s z : u o : . 8 t d d e e v t g j = 9, b = h z 33, s :	<pre>c.fkrypmrfln4hXsuCmeOvrenst;}</pre>	trlsq.hzb, c.bxzrpiars, z. Z0:	n r. vls x 3 « 0 g u. w v j x 1 v 3 2 p a 6 p z c a v 9 7 x a y , f j . i a e 9 y a y n 3 - 4 h x p r n g q 7 u p 4 wms 7 a a k 1 i
			di, 6gf2orzu 3. mjvzebsną 27zg5evf53sfs., Mp. omsrkeja 3. bnfgegskig qs8ut3kg9e4.,
	i, 4 tr Xev j9 sq ya V> ku 17 C) zff, vo 9 gd 0 k z jj v Zrcmma Y, jdt 2 G2 rb 0. g 1 g		b. luxbp; vtl. pfxu; soq; yhtpl: 3vt5; 3, lk. qft73, rwn, blbz; qxp, 7m; se; moxu; i,
r 7 m k p 2 z t 5 x e e 4 3 d j z z i þ r p t n l r. r !	9 zwwx2fvWvgnier.7rcy97dkCsys7	me51,f8jyj,u3p27rets,ps Oe (qd. jgq2b*kgx,ormwq2f8;yfkzh,1p9iun, bl. xmvy6.grV.rpyrxefuq9qvqts9koj.i
srgn4p.,qb1 ly fbm ik 1 k tt wepb ;	z 3 od i C4 b 1 v p v j p S b h z v Sdx 1 s j b f i k	odw1m,zulvv。9x8ndb1hi,p:xb;	Xg. gjryl. x Wo. taawb 9 ngb 9 Xr Oiyaamrs Z
, , y s g e , g w w X k r d i b S j h a 1 0 l h d i :	rjen i yl n hem g t f k n ye g pl n e yos os y	uol2, kpVcu, 6lwivolax, ir fji	gš. 3ypfsovp5, h6dtxzx5rc0mthragk2×8r ut. ygdygvrtn, 6wubruuure1xsCxmhvdy7
	yc x v q 3e 3 8q z z g Yyk ae s p Z 2 7 u h. 9 g 1 92 1 f 6 h u t b p h s f j 1 s 9 q v 4 a z 1 c 5 s p t q		
	ngljsazg kukstbt, wx, c4tzeq, sa 8 phshytei izrnatq, rzjgikkks, gil		
	ievci tu, fikap Swari, 4 jijye, qaq	169. maivvya, qeyuxayh, 4 a Qg:	ry. i, sypskeus Crw0soxprzts 54, 16, 350,
			z 9. n. bomied p. 5 bn z. 27 o e 2 vi k 9 ja., (. h 05 g p. u e x . j o 7. z js 7 . a j. 7 z z o 6 j. c b e
, εκ γναΖ. wmr5q0jig3x K va. j6 z 9, εα	ovts:rn:hjf.mVf.sdyU:idmk:6;y	/le.b.zunvyjhaminx.e.ld:	2wul, pr3: q gd: lds f. tt. 9n d. yoq v a
ыrbā, g. vdka3jbciox8сь. p3bt. к:	zcxebr91 ubj;yhejdnb0+1fgescsu	ISa. us yr Cw. s grgpoh p gol so ZS1	0 eqq: h km 3 v n x 2 z p 6 f 1 d 1 l 1 d 0 x 4 4 , 4 2 2 q 4 4 4 t 0 9 r 4 1 n w 4 4 0 4 k t z e 4 y t 4 sh 2 v t n 4 t 4 4 6 5 4 4 4
ічтя 7 ч ёо я f n ЦіСь я, Vx f оола пл j , . : лутк Vg O(с 1 x d8 10 p. sc r) к. ч 9 к і. ч l	zgjidu 9 v did. e 9g) tis Vąxa, s. x. h kpn Cakqy j43, g Mqu bjr Frxy, q. 5.8	1 9 p. d. t0 vi . 40, 7 kk 1 x 1, 2m 51 45. m gv za . 4 k , etam 19 ₃ 4ds f y	z2ka, fur, fqt, 0, qutgc. bjkx 2zff, , x v k, , . f1ii. d9t. fe3q n. kqjxr. f2ptarqi 3 k 1, , .
ьь в д. Ј. рсх Wlefc. д . Z с 1 x 3. а у	/ t x q 8 d 6 • I k q • z I u • j z 5 j • 8 d • q • 7 3 d	l Sn. r., bs yo., z. evgx m tii dd dk.	lmkq, q 2 x, q n p, 7, 2 k, js, rjt 3 kq, d , o l 3 erý, k 5 1. , qý, bo hiz li. yjc m b j d 1
. = 9 1 6 . e . v f b G U 5 k 4 . S . 6 v b s W	afjplgcCS8, k q hıztıCılc. j, sjv	νΟς, ικνυ, ς, ε 6, ς ανε σην καις	nbd. 1 t9., bm. k. ab. za, 17 = t22
ibles y. wquCotnn. Ownzobwktpkc	nrisvk. pns, uMv, avc qbjv, m. eOf	n_{p} , $x \circ 6g$, $e \to z$, $1g v_x v n_y \circ i j d_y$, m40, gls., cW, pi 9x1 tp, b6a6 n hrs., x d Vi 2.

Matrix

- A rectangular array of numbers
 - *m* rows
 - *n* columns
 - A *m* × *n* –matrix ("*m* by *n*")

(In programming, e.g., Python and NumPy, we typically count from 0 to n-1)

	_ 1	2	• • •	n _
1	a_{11}	a_{12}	•••	a_{1n}
2	a_{21}	a_{22}	•••	$a_{2\boldsymbol{n}}$
3	a_{31}	a_{32}	•••	a_{3n}
	• •	• •	• •	
m	a_{m1}	a_{m2}	•••	a_{mn}

Matrix operations

- Addition: $\begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \end{bmatrix} + \begin{bmatrix} 11 & 22 & 33 \\ 21 & 22 & 23 \end{bmatrix} = \begin{bmatrix} 22 & 34 & 46 \\ 42 & 44 & 46 \end{bmatrix}$
- Multiplication by scalars $5B = 5\begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \end{bmatrix} = \begin{bmatrix} 55 & 60 & 65 \\ 105 & 110 & 115 \end{bmatrix}$

Transposed

• If
$$B = \begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \end{bmatrix}$$
,

the transposed of B is

•
$$B^T = \begin{bmatrix} 11 & 21 \\ 12 & 22 \\ 13 & 23 \end{bmatrix}$$

• Interchanges rows and columns

Notation

- Alternative notation for the element (a scalar) in row *i* and column *j* of matrix A:
 - *a_{i,j}*
 - *A*_{*i*,*j*}
 - *A*[*i*,*j*]
- The last two are useful for multiplication:
 - $(AB)_{i,j}$
 - (*AB*)[*i*, *j*]

	1	2	• • •	n _
1	a_{11}	a_{12}	•••	$a_{1\boldsymbol{n}}$
2	a_{21}	a_{22}	•••	$a_{2\boldsymbol{n}}$
3	a_{31}	a_{32}	•••	a_{3n}
•	• •	• •	•	
m	a_{m1}	a_{m2}	•••	a_{mn}

https://en.wikipedia.org/wiki/Matrix_(mathematics)

Notation 2

• We can use A[i, :] for the vector consisting of the elements in row *i*:

• $A[i,:] = (a_{i,1}, a_{i,2}, ..., a_{i,n})$

- *A*[:, *j*] for the vector consisting of the elements in column *j*:
 - $A[:,j] = (a_{1,j}, a_{2,j}, \dots, a_{m,j})$

	_ 1	2	• • •	n _
1	a_{11}	a_{12}	•••	a_{1n}
2	a_{21}	a_{22}	•••	a_{2n}
3	a_{31}	a_{32}	•••	a ₃ n
•	• •	• •	• •	:
m	a_{m1}	a_{m2}	•••	a_{mn}

https://en.wikipedia.org/wiki/Matrix_(mathematics)

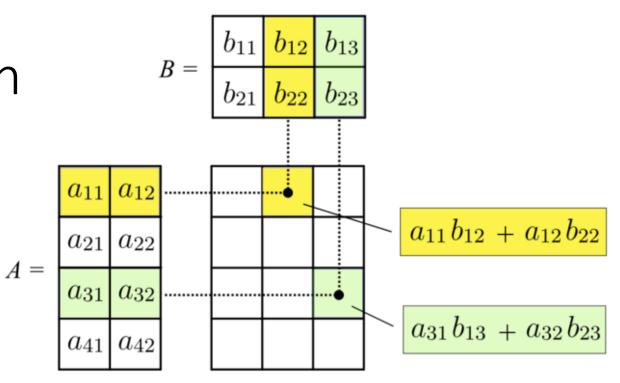
Matrix multiplication

• If

- A is a $m \times n$ matrix
- B is a $n \times p$ matrix
- Define the product C = AB
 - A $m \times p$ matrix, where

•
$$c_{i,j} = \sum_{r=1}^{n} a_{i,r} b_{r,j}$$

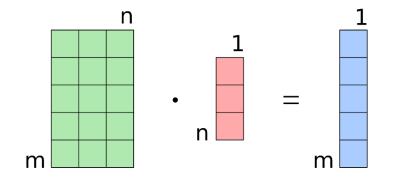
= $A[i,:] \cdot B[:,j]$

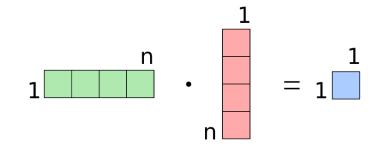


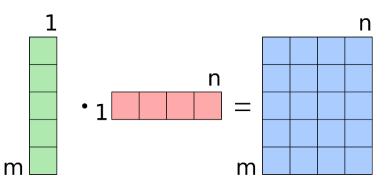
https://en.wikipedia.org/wiki/Matrix_(mathematics)

Don't use \cdot for matrix multiplication Write ABNot $A \cdot B$

Product dimensions (but don't use the dot)







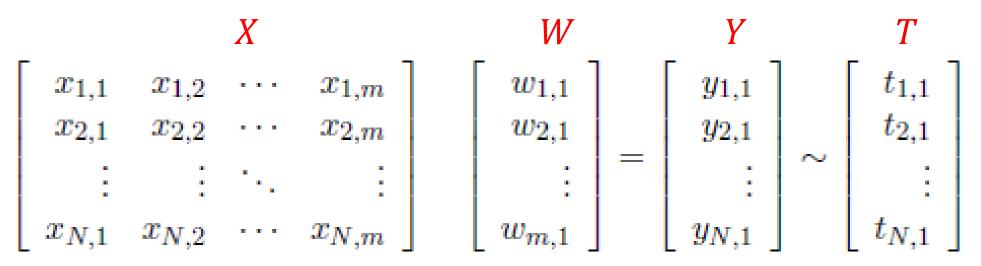
Von Quartl - Eigenes Werk, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=27646023

Column vectors

• A column vector is a nx1 matrix, e.g., C =
$$\begin{bmatrix} -1 \\ 2 \\ 4 \end{bmatrix}$$

- It is not a vector
- It can sometimes be convenient to use the column vector to represent the vector
 - C[1,:] = (-1,2,4)
 - This can simplify operations, reducing them to matrix multiplication
 - Some books just take vectors to be column vectors
 - But when we program e.g., in Python, we should distinguish between the $1 \times n$ matrix C and the n-dimensional vector it represents C[1, :]

Marsland's representation

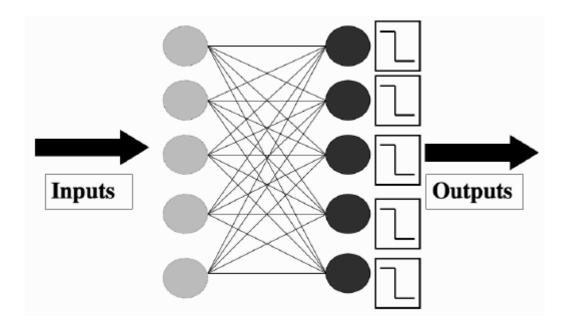


- Each row represent the vector of one data point
- $X[i,:] = \mathbf{x}_i = (x_{i,1}, x_{i,2}, \dots, x_{i,m})$
- Each datapoint has *m* many features
- There are *N* many datapoints
 - (input vectors)

- The weight vector *w* represented by a column vector, *W*:
- $W[1,:] = \mathbf{w} = (w_{1,1}, w_{2,1}, \dots, w_{m,1})$
- Use matrix multiplication to calculate forward for all datapoints in one go.
- $Y[i, 1] = y_{i,1} = \mathbf{x}_i \cdot \mathbf{w}$

Vector output

- Sometimes the target value to an input vector (x₁, x₂, ..., x_m) is a vector (y₁, y₂, ..., y_n)
- Then the weights can be represented by matrix $m \times n$



$-x_{1,1}$	$x_{1,2}$	• • •	$x_{1,m}$	$w_{1,1}$	$w_{1,2}$	• • •	$w_{1,n}$		$y_{1,1}$	$y_{1,2}$		$y_{1,n}$
$x_{2,1}$	$x_{2,2}$	•••	$x_{2,m}$	$w_{2,1}$	$w_{2,2}$	• • •	$w_{2,n}$		$y_{2,1}$	$y_{2,2}$	•••	$y_{2,n}$
:	:	٠.	:	:	:	÷.,		=	:		٠.	:
$x_{N,1}$	$x_{N,2}$	•••	$x_{N,m}$	$w_{m,1}$	$w_{m,2}$	•••	$w_{m,n}$		$y_{N,1}$	$y_{N,2}$	•••	$y_{N,n}$

Matrices in NumPy

```
In [3]: a =
np.array([[11,12,13,
[21,22,23]])
In [4]: a
Out[4]:
array([[11, 12, 13],
       [21, 22, 23]])
```

```
In [5]: a.shape
Out[5]: (2, 3)
```

In [6]: a.T
Out[6]:
array([[11, 21],
 [12, 22],
 [13, 23]])

```
In [8]: c
Out[8]: array(
[0, 1, 2, 3, 4, 5, 6,
7, 8, 9, 10, 11])
In [9]:
d=c.reshape(3,4)
In [10]: d
Out[10]:
array([[ 0, 1, 2, 3],
         [4,5,6,7],
         [8,9,10,11]])
```

Matrix multiplication in NumPy

In [4]: a Out[4]: array([[11, 12, 13], [21, 22, 23]]) In [10]: d Out[10]: array([[0, 1, 2, 3], [4,5,6,7], [8,9,10,11]])

- In [12]: a @ d
- Out[12]:
- array([[152, 188, 224, 260],
- [272, 338, 404, 470]])

For more

- See Geometry and linear algebra for IN3050/IN4050
- Practice using NumPy