

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16
17	18	19	20



Skills For AI: W3 Linear Algebra Solution spaces, Matrix inverse

Amsterdam, Sept 15 2025

Dr. Sieuwert van Otterloo
Dr. Pavlo Burda

Lecture schedule 2025

Week	Date	Topic	Lecturer	Homework
1	01-Sep-25	Linear equations	Pavlo/Sieuwert	Canvas quiz
1	05-Sep-25	Structure of logic formulas, names of operators	Joost	Canvas quiz, Exercise 1 (logic)
2	08-Sep-25	Vector and matrix multiplication, method for solving equations	Pavlo/Sieuwert	Canvas quiz
2	12-Sep-25	Semantic entailment, truth tables	Joost	2. Python exercise 'truth table'
3	15-Sep-25	Solution spaces, Matrix inverse. Python demo 'language determination'	Pavlo/Sieuwert	Canvas quiz
3	19-Sep-25	CNF, DNF	Joost	Canvas quiz
4	22-Sep-25	Determinant and eigenvectors.	Pavlo/Sieuwert	Python exercise 'neural network'
4	26-Sep-25	Set theory	Joost	Canvas quiz
5	29-Sep-25	Using matrices for graphs and neural networks	Pavlo/Sieuwert	Python exercise 'city'
5	3-Oct-25	First order logic - definitions	Joost	First order logic exercise
6	6-Oct-25	Derivatives and gradient	Pavlo/Sieuwert	Canvas quiz
6	10-Oct-25	First order logic - tautologies and proofs	Joost	Canvas quiz
7	13-Oct-25	Bayes Theorem, final questions	Pavlo/Sieuwert	Canvas quiz
7	17-Oct-25	Discuss practice exam	All	

Usually references go at the end of the document

References to scientific papers related to the domain

Pais, C., Liu, J., Voigt, R., Gupta, V., Wade, E., & Bayati, M. (2024). Large language models for preventing medication direction errors in online pharmacies. *Nature Medicine*, 30(6), 1574–1582. <https://doi.org/10.1038/s41591-024-02933-8>

Motivation: This scientific paper describes a study that introduced an AI model incorporating logic. The study presented MEDIC (Medication Direction Copilot), a rule-based system that follows pharmacy logic. This demonstrates that logic can indeed be useful in medication management.

Chalasani, S. H., Syed, J., Ramesh, M., Patil, V., & Kumar, T. P. (2023). Artificial intelligence in the field of pharmacy practice: A literature review. *Exploratory Research in Clinical And Social Pharmacy*, 12, 100346. <https://doi.org/10.1016/j.rcsop.2023.100346>

Motivation: This review describes multiple AI systems used in pharmacy practice. Several of these AI applications use logic and are effectively a decision support for pharmacists and help patients.

Sadeghi, Z., Alizadehsani, R., Cifci, M. A., Kausar, S., Iqbal, B., Bora, P. K., Almasri, A., Alkhawaldeh, R. S., Hussain, S., Alata, H., Hladík, M., Nahavandi, S., & Pardalos, P. M. (2024). A review of artificial intelligence in healthcare. *Computers & Electrical Engineering*, 129, 109370. <https://doi.org/10.1016/j.compeleceng.2024.109370>

Motivation: This review describes AI applications in healthcare decision-making applications. This shows that logic-based AI can be used for medication, but all throughout the health care.

The second paper I chose was titled “Using artificial intelligence methods to speed up drug discovery”. This article talks about the role AI can have in drug discovery. AI can be helpful because drug discovery involves the use of genomic data which is very complex, and AI can help reduce the time it takes to process this data and aid in finding optimal solutions.

The two works are listed below:

- 1) <https://www.sciencedirect.com/science/article/pii/B9780128184387000022>
- 2) <https://www.tandfonline.com/doi/abs/10.1080/17460441.2019.1621284>

W2 Logic Assignment: advice for references usage

R5	If the patient is at risk of brain hemorrhage and the doctor prescribes blood-thinners then the patient could die [1]	$g \wedge q \rightarrow h$
R6	If the patient weighs more than 90kg then the patient does not weigh less than 50kg	$c \rightarrow \neg d$
R7	If the doctor prescribes aspirin then the doctor prescribes blood thinners [2]	$r \rightarrow q$

The idea is to cite the paper in the context, like here (this not graded yet).

LateX does it for you

References

- [1] Raffael G. Liegl, Constance Weber, Maria Bertelsmann, Lea Schaeztle, Isabel stasik, Frank G Holz; Blood thinners in patients with submacular hemorrhage secondary to neovascular AMD. *Invest. Ophthalmol. Vis. Sci.* 2023;64(8):2198
- [2] Kovich O, Otley CC. Thrombotic complications and aspirin therapy perioperatively for cutaneous

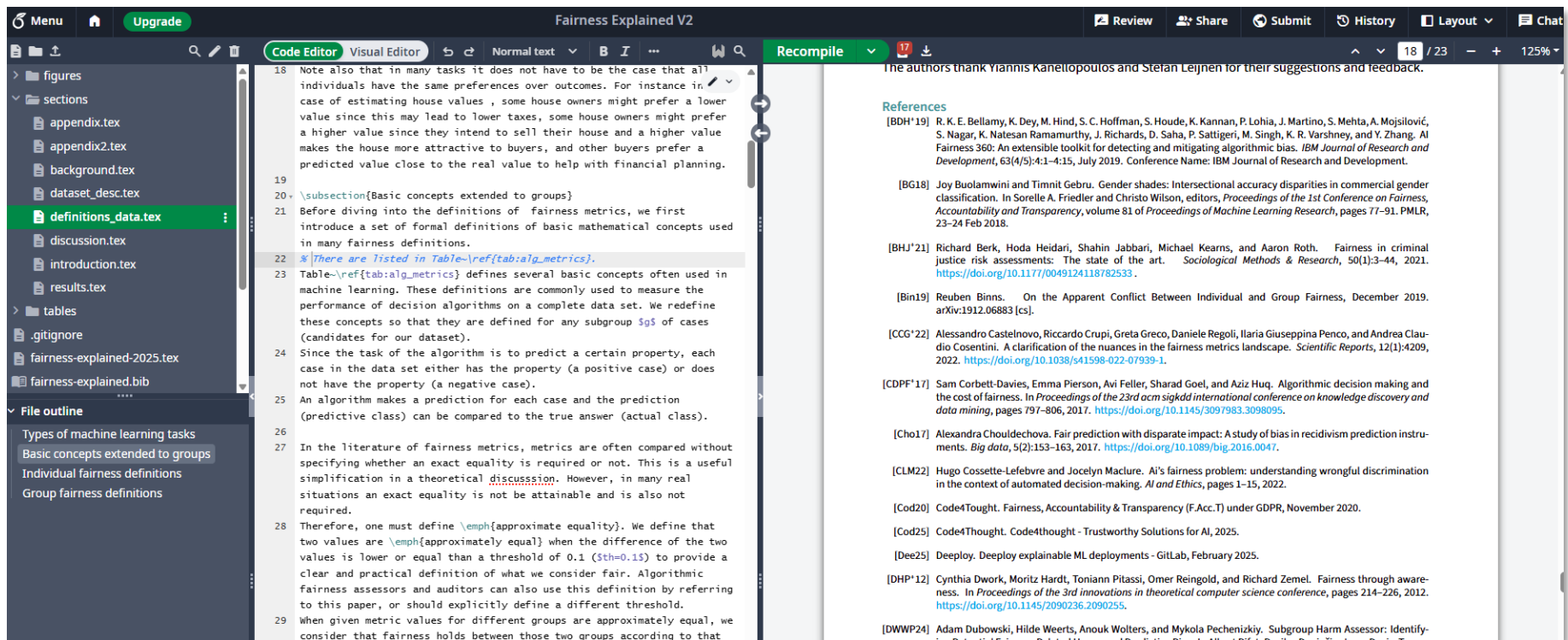
1

This report concerns an assignment on how logic can be used in city planning. Van Vliet, White and Dragicevic [2] use constrained cellular automata to model urban growth. Cellular automata (CA) are defined as a discrete dynamical system with the capacity to have cells in a grid-sized space have a finite number of states. Their states can be updated based on local rules that are functions of their own state and states of their immediate neighbors. Rintanen [1] turns planning problems into satisfiability problems. They introduce a planning-specific variable selection strategy for SAT solving because reduction to SAT has been shown to be a good approach in solving difficult combinatorial problems. They formulate their heuristic on general principles about properties of plans. Although the topics and methods of both papers are more complex than this assignment, it is still interesting because they combine formal rule-based approaches (constraints, logic, simulation) with real-world city planning models.

References

- [1] Jussi Rintanen. Planning as satisfiability. *Artificial Intelligence*, 193:45–86, December 2012.
- [2] Jasper Van Vliet, Roger White, and Dragicevic. Modeling urban growth using a variable grid cellular automata. *Urban Systems*, 33(1):35–43, 2000.

If you are not familiar with tex/latex/overleaf, now is the time to practice with this in a low-risk environment.



SOLUTIONS W2

W2.1 Scalar multiplication

Please multiply:

$$10 * (0.1, 0.25, 0.5, 0.9) = (1, 2.5, 5, 9)$$

$$0.01 * (200, 0, 0, 100) = (2, 0, 0, 1)$$

$$0 * (1, 2, 3) = (0, 0, 0)$$

$$-1 * (-3, -2, 0.25) = (3, 2, -0.25)$$

W2.2 Vector inproduct exercises

Define $a = (3, 4)$.

Please compute $\langle a, x \rangle$ for the following vectors:

$$x = (0.5, 0.5) \quad \langle a, x \rangle = 3.5$$

$$x = (1, 0) \quad \langle a, x \rangle = 3$$

$$x = (0, 1) \quad \langle a, x \rangle = 4$$

$$x = a \quad \langle a, x \rangle = 25$$

W2.3 Matrix multiplication exercises

$$\text{Rot} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$



$$\text{Stretch} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

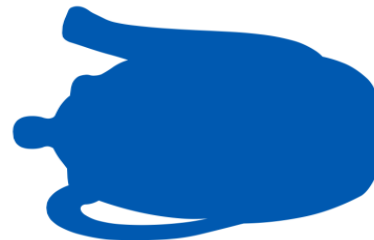


a. Compute $\text{Rot} * \text{Stretch}$



$$R * S = \begin{bmatrix} 0 & -1 \\ 2 & 0 \end{bmatrix}$$

b. Compute $\text{Stretch} * \text{Rot}$



$$S * R = \begin{bmatrix} 0 & -2 \\ 1 & 0 \end{bmatrix}$$

W2.4 : solve the following using the solving-algorithm

$$\left[\begin{array}{ccc|c} 1 & 2 & 3 & 10 \\ 0.5 & 1 & -0.5 & 3 \\ 0 & 1 & 1 & 3 \end{array} \right] \xrightarrow{\quad} \left[\begin{array}{ccc|c} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 1 \end{array} \right]$$

$$\left[\begin{array}{ccc|c} 0 & 2 & -1 & -2 \\ 0.5 & 1 & -2 & -1.5 \\ 0 & 1 & 3 & 2.5 \end{array} \right] \xrightarrow{\quad} \left[\begin{array}{ccc|c} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & -0.5 \\ 0 & 0 & 1 & 1 \end{array} \right]$$

W2.5 Identity and repeated multiplication

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad D = \begin{bmatrix} 0 & 2 \\ -2 & 0 \end{bmatrix}$$

Compute ID

$$ID = D = \begin{bmatrix} 0 & 2 \\ -2 & 0 \end{bmatrix}$$

Compute DD

$$DD = \begin{bmatrix} -4 & 0 \\ 0 & -4 \end{bmatrix}$$

Compute DDD

$$DDD = \begin{bmatrix} -4 & 0 \\ 0 & -4 \end{bmatrix} \begin{bmatrix} 0 & 2 \\ -2 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -8 \\ 8 & 0 \end{bmatrix}$$

Compute $DDDD$

$$DDDD = \begin{bmatrix} 0 & -8 \\ 8 & 0 \end{bmatrix} \begin{bmatrix} 0 & 2 \\ -2 & 0 \end{bmatrix} = \begin{bmatrix} 16 & 0 \\ 0 & 16 \end{bmatrix}$$

W2.6 Filtering using linear transformations

- Smoothing (a.k.a. blurring) is commonly used on the input of machine learning algorithms. It makes algorithms more robust and reliable since it eliminates noise.
- Smoothing can be expressed as a matrix operation. The following matrix specifies a blur filter for a 6-dimensional vector

$$B = \begin{bmatrix} 0.8 & 0.2 & 0 & 0 & 0 & 0 \\ 0.2 & 0.6 & 0.2 & 0 & 0 & 0 \\ 0 & 0.2 & 0.6 & 0.2 & 0 & 0 \\ 0 & 0 & 0.2 & 0.6 & 0.2 & 0 \\ 0 & 0 & 0 & 0.2 & 0.6 & 0.2 \\ 0 & 0 & 0 & 0 & 0.2 & 0.8 \end{bmatrix}$$

a. Compute B (0.1, 0.2, 10, 0.1, 0.3, 0.2):

(0.12, 2.14, 6.06, 2.12, 0.24, 0.22)

b. Compute B (10, 10, 10, 0.1, 0.1, 0.1)

(10, 10, 8.02, 2.08, 0.1, 0.1)

Linear transformations - Color example

Colors in computers are displayed as 3-dimensional vectors:

- R - amount of Red light
- G - amount of Green light
- B - amount of Blue light

The RGB values can either vary from $[0,1]$ or from $(0,255)$ depending on the context.

The model is additive: maximum R,G and B is white.

`rgb(255, 0, 0)`

`rgb(0, 0, 255)`

`rgb(60, 179, 113)`

`rgb(238, 130, 238)`

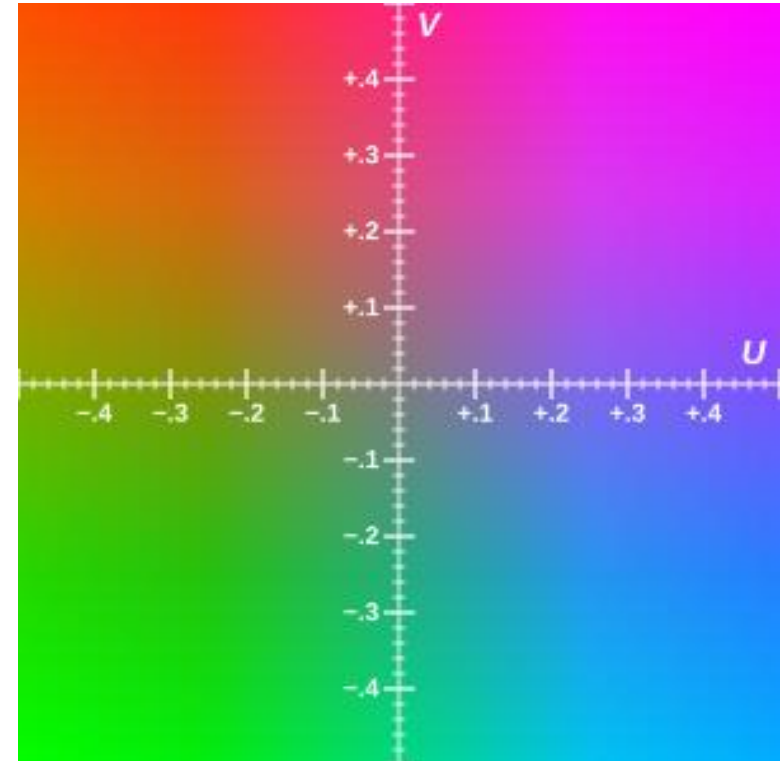
`rgb(255, 165, 0)`

`rgb(106, 90, 205)`

RGB is useful for storage, but not for many operations. There are many applications where you want one brightness channel and 2 color channels. This is why YUV was invented:

- Y is brightness
- U is a green - blue shift: max U is very blue
- V is green - red shift: max V is very red

An RGB to YUV transformation is often done as pre-processing in computer vision.



$$\begin{bmatrix} Y' \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix},$$
$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.13983 \\ 1 & -0.39465 & -0.58060 \\ 1 & 2.03211 & 0 \end{bmatrix} \begin{bmatrix} Y' \\ U \\ V \end{bmatrix}.$$

This matrix is the official transformation. It is finely tuned to best match human perception.

Questions:

- Determine min and max values for Y,U and V by multiplying basic vectors (1,0,0), ... , (0,0,1)
- Try out other common colors.

- Difference between algebra and calculation
- Uses of inproduct and Matrix multiplication
- Definition of a linear function
- Inverse and Transpose
- Solving related systems of linear equations

$$A(x+y-x) = Ay$$

Algebra (Arabic: *al-jabr*) originally means the reunion of broken parts. In mathematics it means the manipulation of symbols.

In layman's terms: using characters, not numbers

$$512 * 64 = 32768$$

Calculation is the manipulation of numbers, also known as 'doing sums'

Numeric computing is computing with computers

$$A0 = 0$$

Zero-potence / additive identity

$$(A+B)x = Ax + Bx$$

$$A(x+y) = Ax + Ay$$

$$(A+B)*C = AC + BC$$

$$A*(B+C) = AB + AC$$

Left distribution - matrix-vector

Right distribution - matrix-vector

Left distribution - matrix

Right distribution - matrix

$$(A*B)x = A*(Bx)$$

$$(A*B)*C = A*(B*C)$$

Associativity

Associativity

$$x+y = y+x$$

$$(x+y)+z=x+(y+z)$$

Commutativity

Associativity (of +)

$$Ix = x$$

$$\text{If } x=y \text{ then } Ax = Ay$$

$$\text{If } A=B \text{ then } Ax = Bx$$

Identity

Substitution

Substitution

If an operation is associative, the brackets are omitted. You can write $A*B*C$.

The following are not true in general. Do not use these as rules:

$$Ax = 0 \text{ implies } A = 0$$

$$Ax = x \text{ implies } A = 1$$

$$A*B = B*A$$

There is always a B such that $A*B=I$

If $Ax = Ay$ then $A=0$ or $x=y$

If $Ax = Bx$ then $x=0$ or $A=B$

$$(A+B)(x+y) = Ax + By$$

Exercise: find interesting counterexamples

Recall: Taking averages with vector calculation

Calculation

Using one formula $\text{avg} = \langle \mathbf{v}, \text{input} \rangle$ you can do many different operations by changing the vector \mathbf{v} .
 $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}, \mathbf{F}$ are different vectors that take some average for an input vector.

input	12	8	6	9	13	inproduct	
A	0.2	0.2	0.2	0.2	0.2	9.60	Normal average
B	0	0	0.33	0.33	0.33	9.33	Average last 3
C	0.33	0.33	0.33	0.00	0.00	8.67	Average first 3
D	0.00	0.00	0.00	0.50	0.50	11.00	Average final 2
E	1.00	0.00	0.00	0.00	0.00	12.00	First element
F	0.06	0.06	0.13	0.25	0.50	10.75	Time series average, focus on last elements

Calculation

Using one formula: **output = A*input**
you can do many different operations on vectors.

Suppose input $v = (1, 2, 3)$

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

What is Av ?

Select first and last

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

What is Bv ?

*Select middle
number three
times*

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

What is Cv ?

Reverse the vector

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 0.33 & 0.33 & 0.33 \\ 0 & 0 & 1 \end{bmatrix}$$

What is Dv ?

*Return first,
average and last
number*

You are in a dating start-up, and tasked with creating the matching algorithm. You decide to match people based on personality traits.

You have identified seven personality traits, as shown in the example.

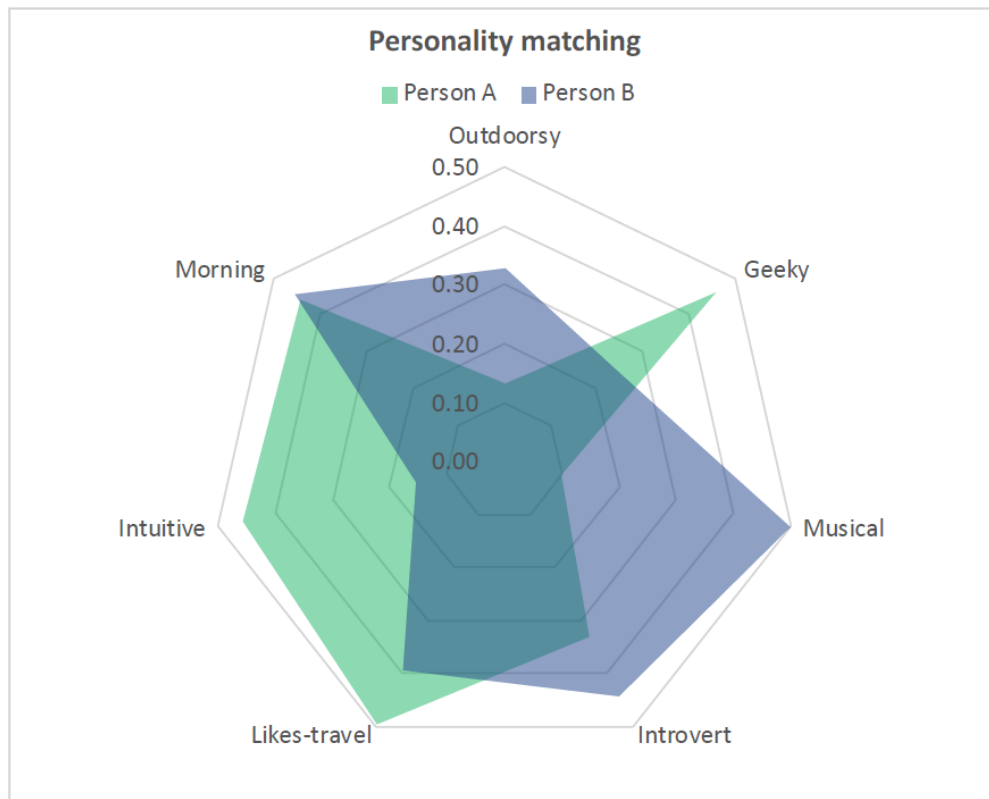
1. How would you display such 7-dimensional vectors?
2. How do you compute the match?



	Outdoorsy	Geeky	Musical	Introvert	Likes-travel	Intuitive	Morning
Person A	0.4	0.4	1.0	0.1	0.6	0.6	0.4
Person B	0.8	0.7	0.4	0.4	0.5	0.7	0.4

Matching and recommendation

	Outdoorsy	Geeky	Musical	Introvert	Likes-travel	Intuitive	Morning
Person A	0.13	0.46	0.10	0.33	0.50	0.46	0.44
Person B	0.33	0.25	0.50	0.45	0.40	0.15	0.45



- It is not true that reality stops at 3 dimensions. Seven-dimensional spaces and upwards are common.
- It is also not true that 7-dimensional vectors are hard to draw. Spiderweb-charts are a good way to do it.
- The inproduct is a common way to compute matches.
- Matches are also useful for social recommendations.

Suppose x is a non-zero vector that represents a personality profile. You want to use the inproduct $\langle x, y \rangle$ for matching.

You would expect that $\langle x, x \rangle = 1.0$ (a 100% match). Is that the case for the following vectors?

	Outdoorsy	Geeky	Musical	Introvert	Likes-travel	Intuitive	Morning
Person A	0.25	0.89	0.19	0.64	0.96	0.88	0.86
Person B	0.65	0.49	0.99	0.89	0.79	0.31	0.90

The answer is no. In general, $\langle x, x \rangle$ can be smaller or larger than 1. The solution is to rescale the vector.

Define $\text{len}(x) = \sqrt{\langle x, x \rangle}$ or
 $\text{len}(x) = \text{sqrt}(\text{inproduct}(x, x))$

Define $\text{normalized}(x) = (1/\text{len}(x)) * x$

This rescaling is called normalizing. We will use $\text{normalized}(x)$ instead of x when convenient.

$$\text{Len}([2, 0]) = 2$$

$$\text{Len}([3, 4]) = \text{sqrt}(9+16) = \text{sqrt}(25)=5$$

$$\text{Len}([-1, 0]) = 1$$

$$\text{Len}([-1, 1]) = 1.42\dots$$

Calculation

Exercise: normalize the following vectors

	Outdoorsy	Geeky	Musical	Introvert	Likes-travel	Intuitive	Morning
Jane	0.894	0.528	0.110	0.890	0.715	0.901	0.220
Person A	0.254	0.889	0.191	0.642	0.961	0.880	0.855
Person B	0.651	0.491	0.990	0.886	0.789	0.307	0.904
Person C	0.796	0.733	0.262	0.698	0.923	0.906	0.430
Person D	0.880	0.889	0.440	0.340	0.961	0.255	0.855

Calculation

Solution:

	Outdoorsy	Geeky	Musical	Introvert	Likes-travel	Intuitive	Morning
Jane	0.496	0.293	0.061	0.493	0.396	0.500	0.122
Person A	0.131	0.460	0.099	0.332	0.497	0.455	0.443
Person B	0.327	0.247	0.497	0.445	0.396	0.154	0.454
Person C	0.420	0.387	0.138	0.369	0.487	0.479	0.227
Person D	0.464	0.469	0.232	0.179	0.507	0.135	0.451

What language is the following Poem in?



Il brilgue: les tôves lubricilleux
Se gyrent en vrillant dans le guave.
Enmîmés sont les gougebosqueux
Et le mômerade horsgrave.

't Wier bradig, en de spiramants
Bedroorden slendig in het zwiets:
Hoe klarm waren de ooiefants,
Bij 't bluifen der beriets.

Most people identify the language based on the character frequencies.

Note that most words in this poem do not have a clear meaning. The Poet (Lewis Carroll) decided to make a point by writing a nonsense-poem.

A logical approach would conclude that the poems do not match any language.

A linear algebra approach uses counting, multiplying and averaging to reach a best estimate.

How to solve this using linear algebra?

Letter	Dutch	English	French
A	7.76%	8.34%	8.13%
B	1.38%	1.54%	0.93%
C	1.31%	2.73%	3.15%
D	5.48%	4.14%	3.55%
E	19.31%	12.60%	15.10%
F	0.74%	2.03%	0.96%
G	3.17%	1.92%	0.97%
H	3.16%	6.11%	1.08%
I	5.03%	6.71%	6.94%
J	0.51%	0.23%	0.71%
K	2.83%	0.87%	0.16%
L	3.85%	4.24%	5.68%
M	2.60%	2.53%	3.23%
N	10.04%	6.80%	6.42%
O	5.88%	7.70%	5.27%
P	1.51%	1.66%	3.03%
Q	0.01%	0.09%	0.89%
R	5.70%	5.68%	6.43%
S	3.91%	6.11%	7.91%
T	6.50%	9.37%	7.11%
U	2.13%	2.85%	6.05%
V	2.27%	1.06%	1.83%
W	1.74%	2.34%	0.04%
X	0.05%	0.20%	0.42%
Y	0.06%	2.04%	0.19%
Z	1.62%	0.06%	0.21%

Source: sttmedia.com by Stefan Trost

The following data are character frequency tables.

How to get a frequency table?

- You get these by counting all characters, then rescaling
- $v_{scaled} = V / \sqrt{\langle v, v \rangle}$

How to compare vectors?

- Each vector is an arrow in a 26-dimensional space. You can find the most similar language profile for each text by comparing directions.
- You compare directions using inproduct:
 - $p_{Dutch}(text) = \langle F_{dutch}, F_{text} \rangle$
 - $p_{French}(text) = \langle F_{frenct}, F_{text} \rangle$
- Return the language with the highest probability

Note 1: the inproduct is one way to compare vectors. Other ways may work better for certain cases.

Note 2: The same except that you would use the non-letter characters (!@#\$%^&*()_+{}[]:”|;’\<>?.,./) works for computer language recognition.

Note 3: One can do the same for whole words. This is often used in spam filters.

Exercise:

Letter	Dutch	English	French
A	7.76%	8.34%	8.13%
B	1.38%	1.54%	0.93%
C	1.31%	2.73%	3.15%
D	5.48%	4.14%	3.55%
E	19.31%	12.60%	15.10%
F	0.74%	2.03%	0.96%
G	3.17%	1.92%	0.97%
H	3.16%	6.11%	1.08%
I	5.03%	6.71%	6.94%
J	0.51%	0.23%	0.71%
K	2.83%	0.87%	0.16%
L	3.85%	4.24%	5.68%
M	2.60%	2.53%	3.23%
N	10.04%	6.80%	6.42%
O	5.88%	7.70%	5.27%
P	1.51%	1.66%	3.03%
Q	0.01%	0.09%	0.89%
R	5.70%	5.68%	6.43%
S	3.91%	6.11%	7.91%
T	6.50%	9.37%	7.11%
U	2.13%	2.85%	6.05%
V	2.27%	1.06%	1.83%
W	1.74%	2.34%	0.04%
X	0.05%	0.20%	0.42%
Y	0.06%	2.04%	0.19%
Z	1.62%	0.06%	0.21%

Compute the p_{Dutch} , p_{English} for the word “KOE”:

$$\bullet \quad p_{\text{Dutch}}(\text{koe}) = 2.82\% \cdot 1 + 5.88\% \cdot 1 + 19.31\% \cdot 1 = 0.28$$

$$\bullet \quad p_{\text{English}}(\text{koe}) = 0.87\% \cdot 1 + 7.70\% \cdot 1 + 12.60\% \cdot 1 = 0.21$$

Conclusion: the word “koe” is classified as Dutch

Source: sttmedia.com by Stefan Trost

Exercise: recognize the programming language

JavaScript Freq. (%)	Python Freq. (%)
;	12.66
(12.03
)	12.03
.	8.23
"	7.59
/	6.33
{	5.70
}	5.70
=	5.06
+	3.80
*	3.16
,	3.16
'	2.53
0	1.90
2	1.90

Compute the p_{Python} , $p_{\text{Javascript}}$ for the string:

```
example_1 =  
"console.log("Hello, " + name + "!"); // A comment"
```

- $p_{\text{Python}}(\text{example_1}) =$
 $\text{np.dot}(\text{example_1}, \text{py_freq_vector}) = 561.66$
- $p_{\text{Javascript}}(\text{example_1}) =$
 $\text{np.dot}(\text{example_1}, \text{js_freq_vector}) = 709.76$

Conclusion: the string `example_1` is classified as Javascript

Try it yourself (Google account needed):
<https://rb.gy/xv2zc1>

Source: derived from example code

It may happen that you are given a matrix in the wrong format 2x3

$$A = \begin{bmatrix} 0.5 & 1.0 & -0.3 \\ -0.2 & 0.0 & 1.0 \end{bmatrix}$$

While you need the format to be 3x2:

$$B = \begin{bmatrix} 0.5 & -0.2 \\ 1.0 & 0.0 \\ -0.3 & 1.0 \end{bmatrix}$$

The operation to go from A to B is called 'transpose'. Notation:

$$B = A^T$$

Note that $A^{TT} = A$.

In programming languages where vectors are stored as $m \times 1$ 'column' matrix, the transpose of a $m \times 1$ 'column' matrix would be a $1 \times m$ 'row' matrix.

Example:

Can you multiply matrix A with itself?

$$A = \begin{bmatrix} 1.0 & 1.0 & 0.0 \\ 2.0 & 2.0 & 2.0 \end{bmatrix}$$

Compute A^T

$$A^T = \begin{bmatrix} 1.0 & 2.0 \\ 1.0 & 2.0 \\ 0.0 & 2.0 \end{bmatrix}$$

Compute AA^T

$$AA^T = \begin{bmatrix} 2.0 & 4.0 \\ 4.0 & 12.0 \end{bmatrix}$$

Calculation

LINEAR FUNCTIONS



Suppose f is a function that takes one scalar and return one scalar.
A linear function f is any function with the following properties:

1. $f(0) = 0$
2. $f(a+b) = f(a)+f(b)$
3. $f(c*a) = c*f(a)$

Algebra

Define $f(x) = 2*x$. Prove that f is linear.

1. $f(0) = 2*0 = 0$
2. $f(a+b) = 2*(a+b) = 2*a + 2*b = f(a)+f(b)$
3. $f(c*a) = 2*c*a = c*2*a = c*f(a)$

Which of the following functions are linear?

- a) $f(x) = x + y^2$
- b) $g(x) = x^2$
- c) $h(x) = 0$
- d) $j(x) = x+1$

h are linear
 f, g, j
are not linear

Define $g(x) = x^2$.
 $g(1+1) = 2^2 = 4$.
 $g(1)+g(1) = 1+1 = 2$
No, g is not linear

Define $h(x) = 0$. For all x . is x linear?

- $h(0) = 0$
- $h(a+b) = 0 = 0+0 = h(a)+h(b)$
- $h(c*a) = 0 = c*0 = c*h(a)$

Yes, h is linear

Suppose M is a matrix, x a vector, c a scalar and 0_n a null-vector of the right length

Define $f(x)=Mx$

You can prove that

- $M(0_n) = 0_n$
- $M(a+b) = Ma+Mb$
- $M(c*a) = c*Ma$

Scalar exponents

$$3^1 = 3$$

$$3^2 = 9$$

$$3^3 = 27$$

$$3^4 = \dots$$

$$3^0 = 1$$

$$3^{-1} = 0.333$$

$$3^{-2} = 0.111$$

Question: Can we
define the same for
matrices?



Define A^1 as the matrix for doing $a := Ax$ once

Define A^2 as the matrix for doing $a := Ax$ twice. $A^2(x) = A(A(x)) = (AA)(x)$

Define A^0 as the matrix for doing nothing at all: $A^0(x) = x$.

Note: $A^0 = I$

Define A^{-1} as the matrix for undoing A : $A^{-1}(A(x)) = x$. So $(A^{-1}A) = I$.

What are the inverses for these operations?

$$\text{Rot} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$



$$\text{Rot}^{-1} = \text{Rot}^3 \quad \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

$$\text{Stretch} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\text{Stretch}^{-1} = \text{Sqz} = \begin{bmatrix} 0.5 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\text{Flip} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\text{Flip}^{-1} = \text{Flip} \quad \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

The inverse matrix is a general solution

Suppose that you are the plant manager of a chip factory. Each week, you get an order to produce a number of chips of different types $t=(t_1, t_2, t_3)$.

You have to compute the number of batches $b=(b_1, b_2, b_3)$ that you must produce. The relation between the batches and produced types is given by a matrix A so that $t=Ab$.

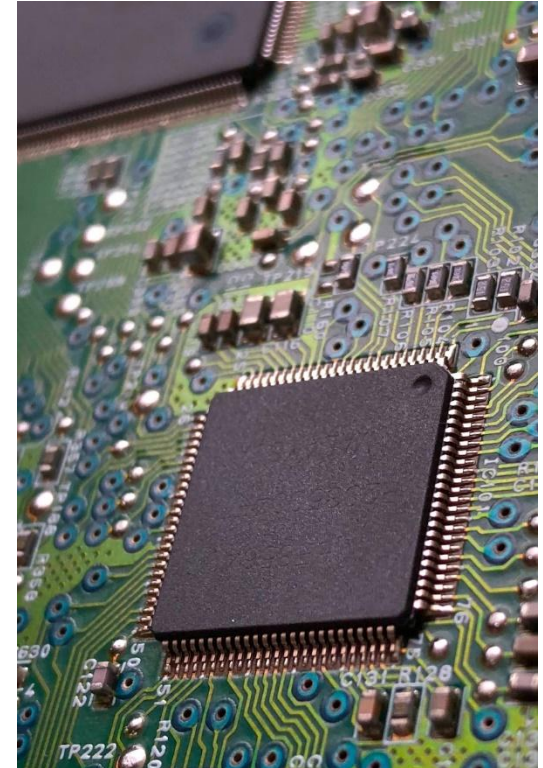
How can you simplify the task of computing b for each week, if you are given a different vector t every time?

Answer:

1. You compute the inverse A^{-1} .
2. Each week, Compute $b= A^{-1} t$.

This way you always get the right t :

$$t= Ab = A A^{-1} t = It = t$$



To invert a matrix A ,
create the matrix $[A|I]$: put the identity matrix next to A .
Use the solution algorithm on this augmented matrix. You will get $[IA^{-1}]$

Start with the leftmost column $m=1$.

Repeat for each column-number m :

- **Make a swap if needed** to get a non-zero element c at the m^{th} position
- **Multiply by $(1/c)$** to get a 1 at the m^{th} position
- **Subtract y times the m^{th} row from the other rows** to get all zero's (using a suitable factor y for each row)
- **Move to the next column**

The matrix solution algorithm – example

A

=

1.0	0.5	0.5
1.0	2.0	0.0
1.0	0.0	2.0

Define [A|I]

1.0	0.5	0.5	1	0	0
1.0	2.0	0.0	0	1	0
1.0	0.0	2.0	0	0	1

Subtract first row
from other rows

1.0	0.5	0.5	1	0	0
0.0	1.5	-0.5	-1	1	0
0.0	-0.5	1.5	-1	0	1

Multiply middle row

1.0	0.5	0.5	1	0	0
0.0	1.0	-0.33	-0.66	0.66	0
0.0	-0.5	1.5	-1	0	1

The matrix solution algorithm – part 2

From previous page

1.0	0.5	0.5	1	0	0
0.0	1.0	-0.33	-0.66	0.66	0
0.0	-0.5	1.5	-1	0	1

Subtract middle row

1	0	0.67	1.33	-0.33	0
0	1	-0.33	-0.66	0.66	0
0	0	1.34	-1.33	0.33	1

Multiply bottom row

1	0	0.67	1.33	-0.33	0
0	1	-0.33	-0.66	0.66	0
0	0	1	-1.00	0.25	0.75

Subtract bottom row
from top rows

1	0	0.00	1.99	-0.49	-0.50
0	1.00	0.00	-0.99	0.74	0.25
0	0	1.00	-1.00	0.25	0.75

$A^{-1} =$

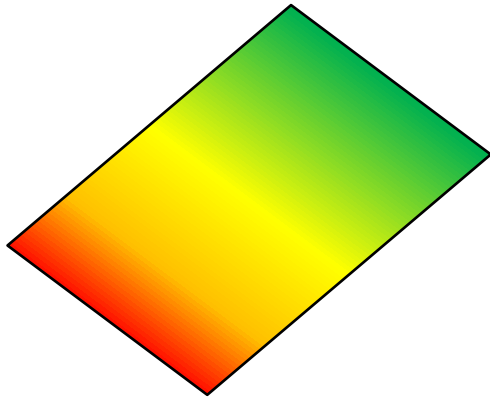
1.99	-0.49	-0.50
-0.99	0.74	0.25
-1.00	0.25	0.75

The matrix solution algorithm – check solution

$$A = \begin{bmatrix} 1.0 & 0.5 & 0.5 \\ 1.0 & 2.0 & 0.0 \\ 1.0 & 0.0 & 2.0 \end{bmatrix} \quad A^{-1} = \begin{bmatrix} 1.99 & -0.49 & -0.50 \\ -0.99 & 0.74 & 0.25 \\ -1.00 & 0.25 & 0.75 \end{bmatrix}$$

$$AA^{-1} = \begin{bmatrix} 2-0.5-0.5 & -0.49+0.38+0.12 & -0.5+0.12+0.38 \\ -0.5+0.38+0.12 & -0.49+1.5 & -0.5+0.5+0 \\ 2.0-2.0 & -0.49+0+0.5 & -0.5+0+1.5 \end{bmatrix}$$

$$AA^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



The beauty of linear functions is that there are not so many linear functions. Once you know a few data points, you can specify the function by specifying the matrix.

Suppose that D is a 3×3 matrix. We know that:

$$D * (2,0,0) = (0,8,4)$$

$$D * (0,2,0) = (8,0,0)$$

$$D * (0,0,1) = (1,1,1)$$

Can you find D ?

$D =$

0	4	1
4	0	1
2	0	1

- You can use inproducts for defining algorithms
- You can add solutions to linear equations: If $Ax=v$ and $Ay=w$ then $A(x+y) = v+w$
- Many matrices A have an inverse A^{-1}
- There is an algorithm for computing the inverse of a matrix.

This week for linear algebra, there is a quiz.

EXERCISES

W3.2 Finding inverse of transformations

$$A = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.2 \end{bmatrix}$$

What is the inverse A^{-1}

$$B = \begin{bmatrix} 1 & 2 \\ 0 & 0 \end{bmatrix}$$

What is the inverse B^{-1}

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

What is the inverse C^{-1}

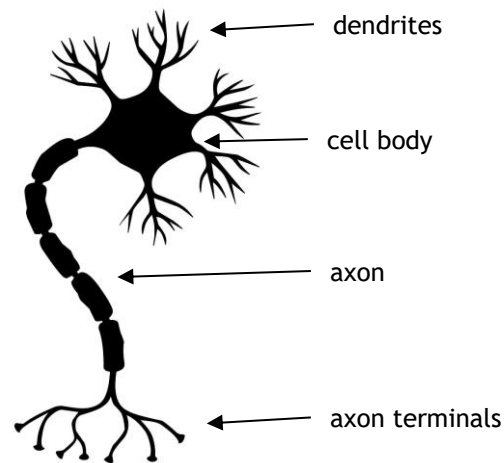
Note: these can be solved directly, without the algorithm
You can check your answer in Jupyter, using `np.linalg.inv(M)`

$$C = \begin{array}{|c|c|c|} \hline 0.5 & 0.5 & 0.5 \\ \hline 1.0 & -1.0 & 0.0 \\ \hline 0 & 1.0 & -1.0 \\ \hline \end{array}$$

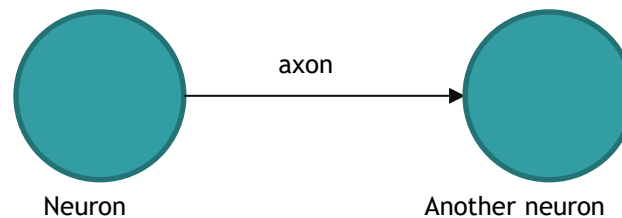
find the inverse C^{-1} using the
matrix inverse algorithm

NEURAL NETWORKS

What are neural networks?



Symbolic illustration of actual
neuron



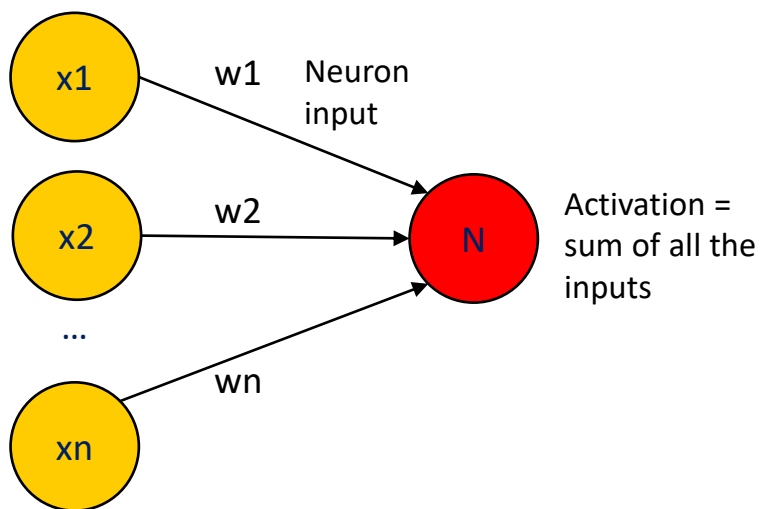
Neurons are cells that are part of the nervous system (brain, spinal cord, sensory organs, and nerves). They are often depicted as a core cell with many connections (axons/tendons) to other neurons.

The neurons thus form a neural network.

- Some neurons respond to external signals, e.g. light or sound
- Some neurons control muscle cells
- Other neurons respond to signals from other neurons. Once they are agitated by other signals, they will ‘fire’ and send a signal to other neurons



How artificial, linear neural networks work

inputs

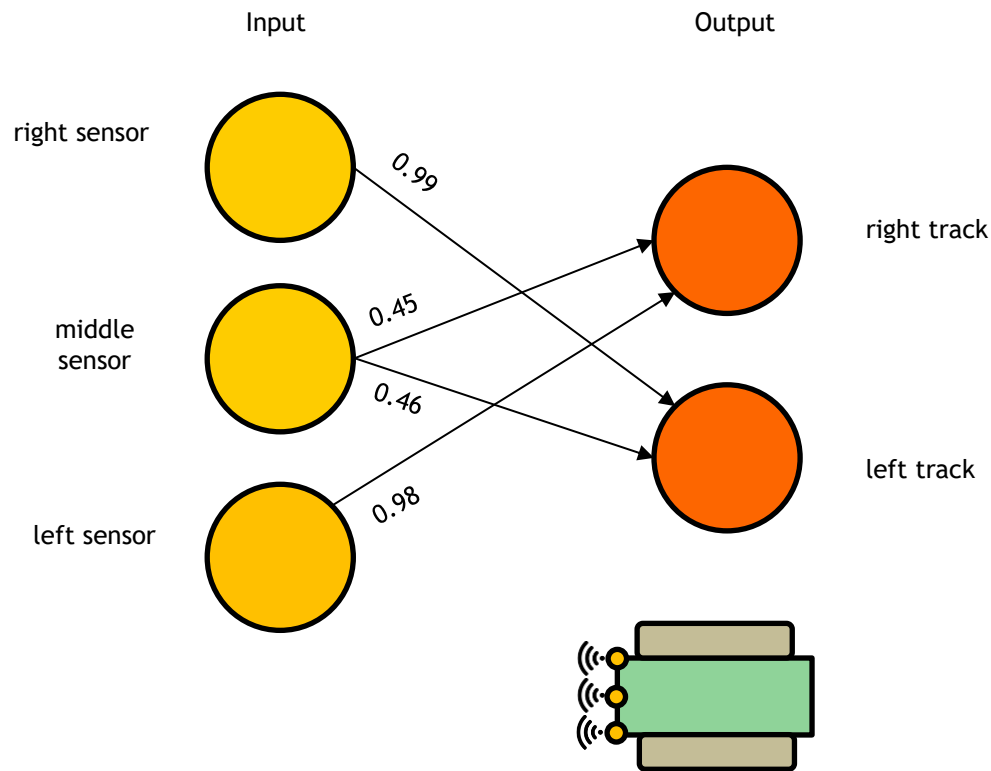


$$\text{InputN} = x_1 * w_1 + \dots + x_n * w_n$$

(linear combination,
inproduct)

 input
 output

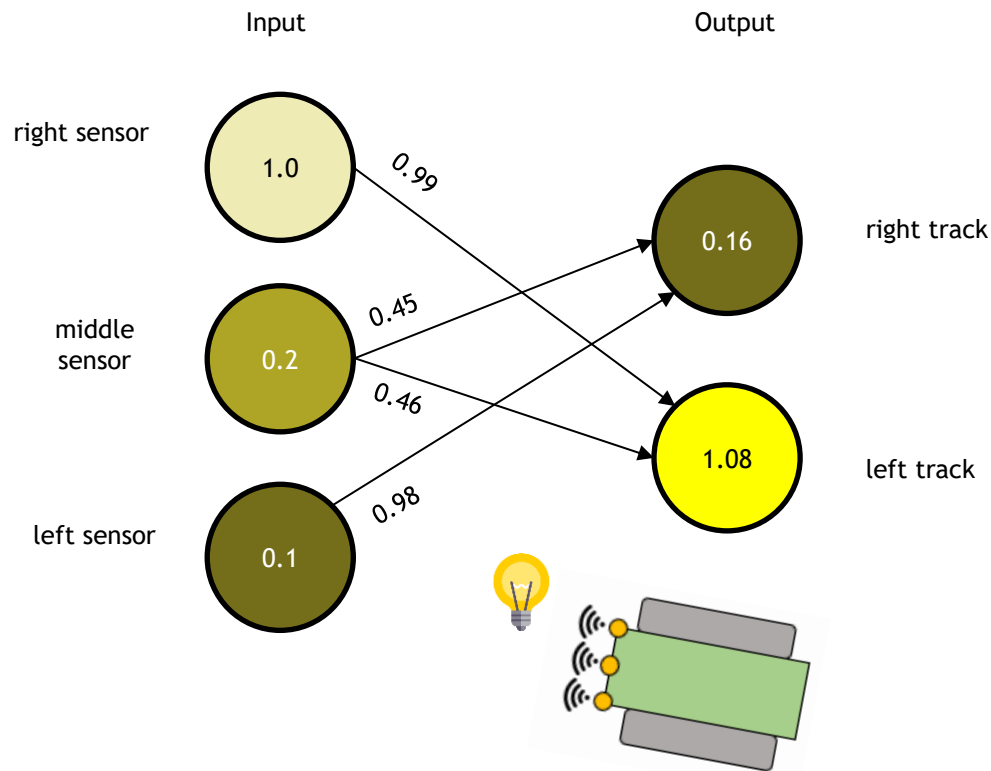
Linear neural networks
 $\text{OutputN} = \text{InputN}$



The following simple network can be used for making tiny robots that either move towards or away from the light.

- Each neuron has an 'activation' (0, 1 or any value in between)
- Each link has a 'weight', typically between -1.0 and 1.0

Note: this 'cockroach-like' robot was used to demonstrate that it is interesting to model 'sub-human-intelligence'. Rodney Brooks called it 'embodied intelligence' and argued that AI researchers should stop overfocusing on logic and reasoning.



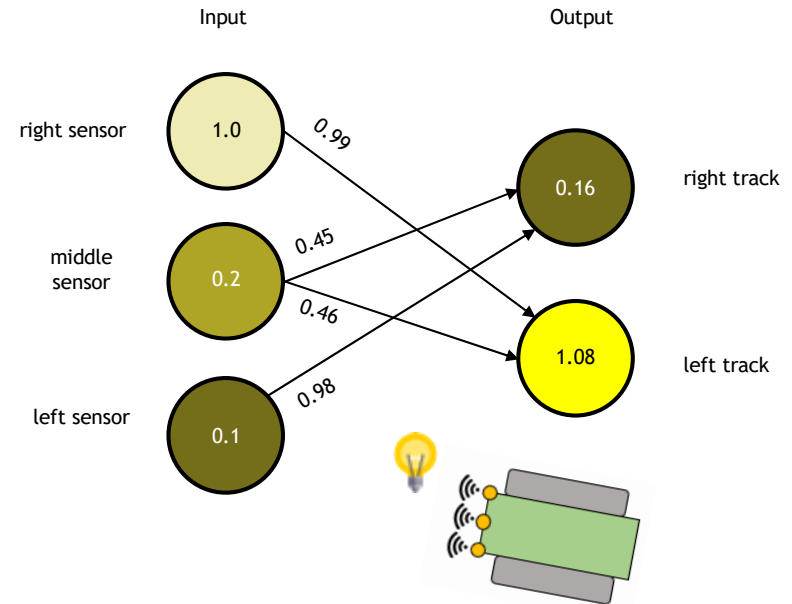
If there is a light on the right side of the robot, the left track gets the strongest signal.
The robot will turn to the right, so towards the light.

- For each neuron, you add up the sum of the input-neurons * weight
- If the sum is above the threshold (e.g. 1), the neuron is activated

$$\begin{aligned}\text{Input Activation (left track)} &= \\ &1.0 * 0.99 + 0.2 * 0.46 = \\ &0.99 + 0.09 = 1.08\end{aligned}$$

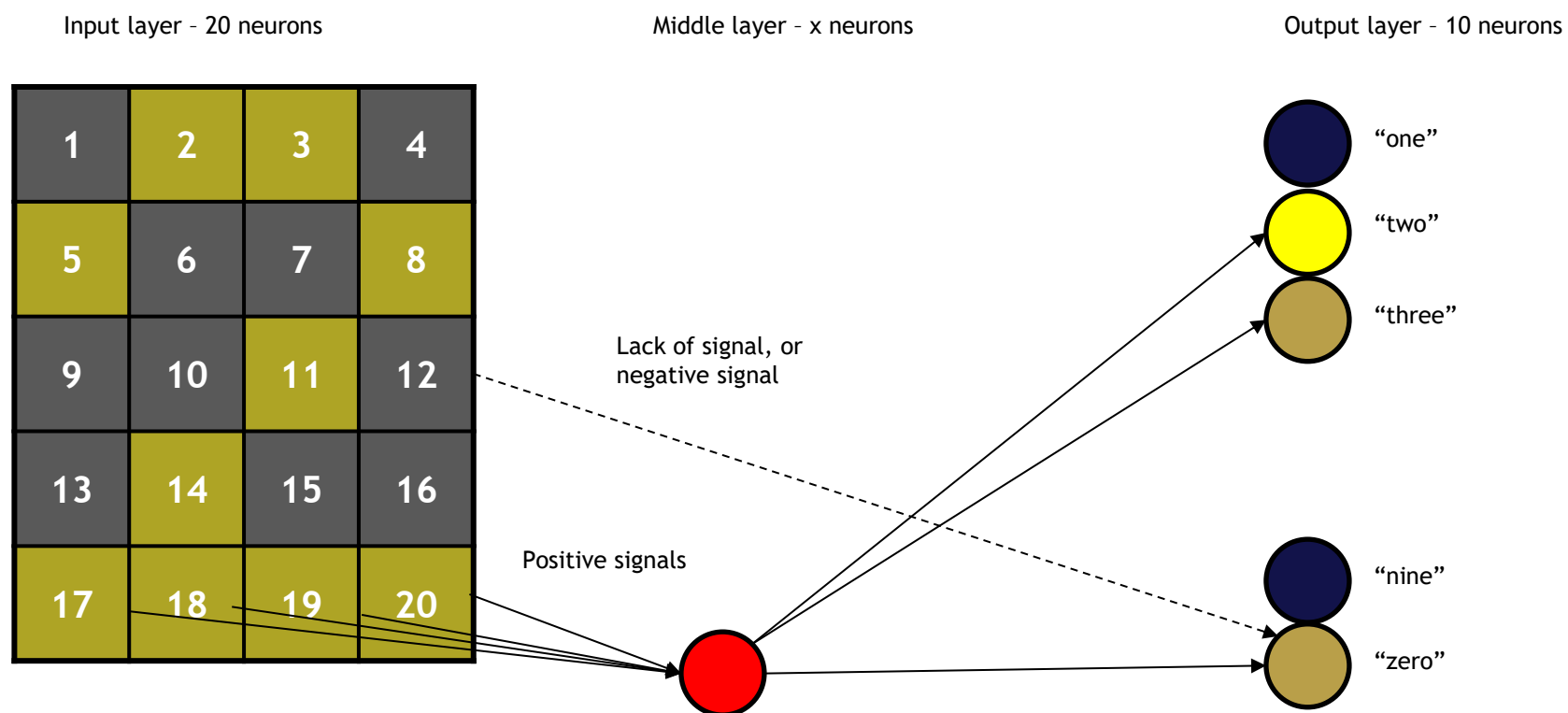
Exercise 3: Neural network

1. Write down the matrix for the neural network weights and connections shown in Python.
2. Compute the activations of the outputs for the following input vectors using a simple function:
 - $[0.9, 0.0, 0.9]$
 - $[0.1, 0.8, 0.2]$
 - $[0.0, 0.1, 0.0]$
3. Create a single new matrix such that the following input provide the specified output.
4. Is it possible to find a matrix such that the robot moves forward if either left or right is lit more than the other, but is still or goes backwards if both are lit? Please explain your answer.



Input Q3	Output Q3
(0.8,0.4,0.0)	(1.0,0)
(0.0,0.4,0.8)	(0,0.9)

Artificial neural networks – image classification













Multi-layer networks

Input layer - 20 neurons

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16
17	18	19	20

Middle layer - x neurons

-  Top line
-  \ diagonal
-  middle line
-  / diagonal
-  bottom line
-  Top circle
-  Cross in center
-  Top heavy
-  Leftish
-  Empty center

Output layer - 10 neurons



- In multi-layer networks, there are neurons that are not inputs and not outputs.
- They can be trained to represent intermediate concepts. These are often higher-level concepts, combining multiple inputs
- The outputs are no longer determined by single pixels, but by a combination of intermediate concepts

Note: it is possible to have multiple hidden layers. This can be built to define even more complex concepts. E.g. first layer recognizes circles. Second layer or third layer recognizes bicycles.

Assignment: classification

We will do character recognition on 5x5 images. Each picture represents a character that you want to recognize.

Below are three example characters:

‘x’

55%	0%	0%	0%	66%
5%	99%	0%	99%	0%
0%	5%	99%	0%	0%
5%	99%	0%	99%	0%
88%	0%	0%	0%	44%

‘+’ / plus

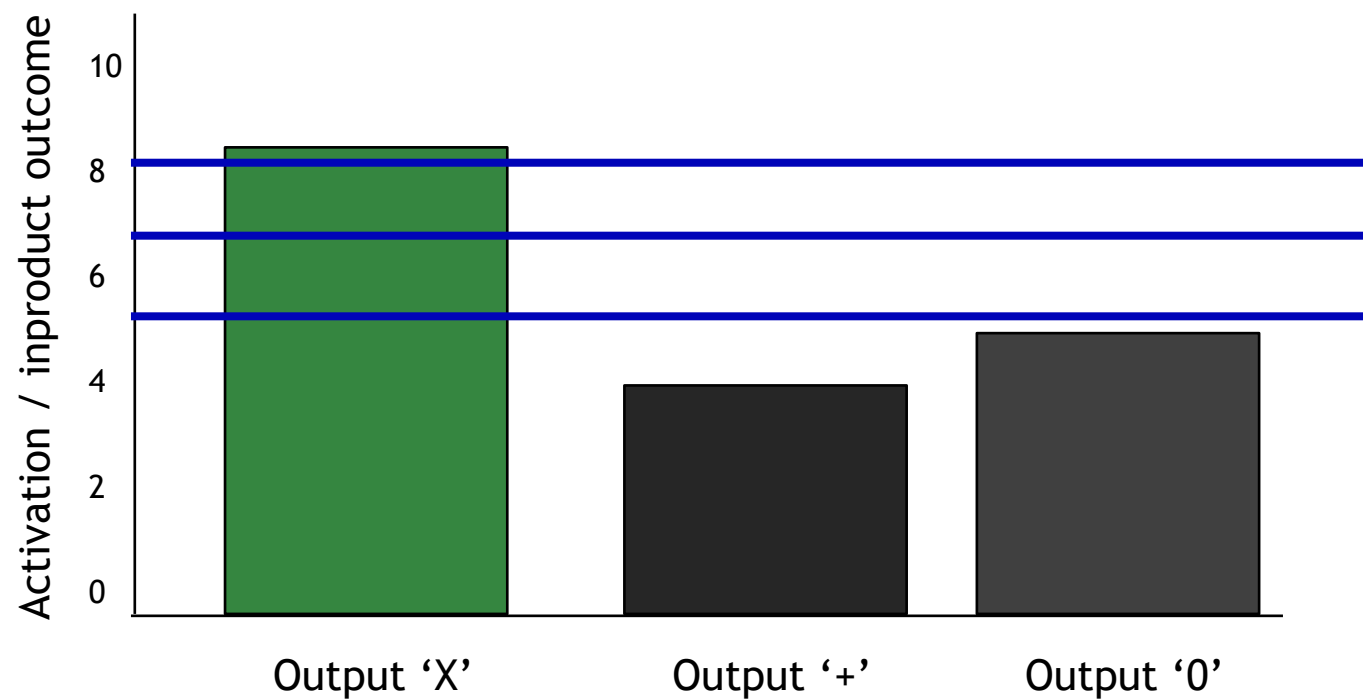
0%	0%	99%	0%	0%
0%	0%	99%	0%	0%
99%	99%	99%	99%	99%
0%	0%	99%	0%	0%
0%	0%	99%	0%	0%

‘o’ / zero

99%	99%	99%	99%	99%
99%	0%	0%	0%	99%
99%	0%	0%	0%	99%
99%	0%	0%	0%	99%
99%	99%	99%	99%	99%

The neural network should match any input picture to one of these character, and return whether it sees a x, plus or zero

NN1 output for input 'X33'



Possible threshold values: the correct value is higher, the incorrect output values are lower. Middle line is probably best

Making vectors from images

55%	0%	0%	0%	66%
5%	99%	0%	99%	0%
0%	5%	99%	0%	0%
5%	99%	0%	99%	0%
88%	0%	0%	0%	44%

```
import numpy as np
import math

x = [0.5, 0, 0, 0, 0.1, 0.05, 0.99, 0, 0.99, 0, 0, 0.05, 0.99, 0, 0, 0.05, 0.99, 0, 0.99, 0, 0.5, 0, 0, 0, 0.1 ]
plus = [0, 0, 0.99, 0, 0, 0, 0, 0.99, 0, 0, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0, 0, 0.99, 0, 0, 0, 0.99, 0, 0 ]
zero= [0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0, 0, 0, 0.99, 0.99, 0, 0, 0, 0.99, 0.99, 0, 0, 0, 0.99, 0.99, 0.99, 0.99,
0.99, 0.99 ]
```

Define each image as a 25-element vector, Use numbers between 0.0 and 1.0 to represent the input

55%	0%	0%	43%	32%
45%	54%	15%	76%	0%
0%	33%	66%	0%	0%
25%	99%	12%	70%	0%
88%	0%	0%	34%	23%

55%	5%	3%	14%	56%
5%	55%	22%	80%	12%
21%	12%	99%	24%	4%
5%	67%	23%	75%	21%
88%	12%	5%	3%	56%

0%	0%	0%	0%	0%
0%	100%	15%	76%	0%
0%	33%	66%	12%	0%
0%	67%	12%	70%	0%
32%	11%	3%	10%	12%

It is important to train and test any AI algorithm with a large input set.

Therefore, you should create three alternative versions for each character.

If you can do it, define matrices for operations to make the alternatives. Matrices to consider:

- Blur (note: more complex than previous blur since it is two-dimensional)
- Shift up, right, left, down
- Brighten / darken
- Slant / spiral, other effects or local blur
- Add mild noise or variations of intensity

```
In [10]: print(str(np.dot(x,x))+'\t'+str(np.dot(plus,x))+'\t'+str(np.dot(zero,x)))  
print(str(np.dot(x,plus))+'\t'+str(np.dot(plus,plus))+'\t'+str(np.dot(zero,plus)))  
print(str(np.dot(x,zero))+'\t'+str(np.dot(plus,zero))+'\t'+str(np.dot(zero,zero)))
```

```
5.428    1.0296   1.287  
1.0296   8.8209   3.9204  
1.287    3.9204  15.6816
```

- Vector inproduct $\langle x, \text{input} \rangle$ is an excellent operation for checking how similar vectors are.
- In classification, you often make confusion matrices: you compare all your inputs to all your examples.
- You should get high values on the diagonal, and low values everywhere else.
- You should set threshold values lower than the diagonal, higher than the other values

		Similarity to zero	
Actual input x	5.428	1.029	1.287
	1.029	8.820	3.920
	1.287	3.920	15.681

Note that neural networks often use negative weights to denote that a certain input has a negative correlation to a certain output. This will give much clearer answers.

A simple correction would be to put -0.1 when the example character has no value:

```
In [63]: xminus= np.copy(x)
         for i in range(len(xminus)):
             xminus[i] = xminus[i] - 0.1
         print(xminus)

[ 0.4  -0.1  -0.1  -0.1   0.   -0.05  0.89 -0.1   0.89 -0.1  -0.1  -0.05
  0.89 -0.1  -0.1  -0.05  0.89 -0.1   0.89 -0.1   0.4  -0.1  -0.1  -0.1
  0.   ]
```

```
In [64]: test=xminus
         print(np.dot(test,xminus))
         print(np.dot(test,zero))
         print(np.dot(test,plus))

4.417999999999999
-0.29700000000000004
0.1386
```

You should make the best possible afilter, bfilter, cfilter other tweaks. E.g. rescaling could help in not getting 4.4 as a score but a nice one by making this tweak, perhaps 10.0 or 1.0. You probably also need to blur a bit to recognize input variations

Combine your three recognizers in a matrix

```
In [13]: NN=[x,plus,zero]
print(np.matmul(NN,x))
print(np.matmul(NN,zero))
print(np.matmul(NN,plus))
```

```
[5.428  1.0296 1.287 ]
[ 1.287   3.9204 15.6816]
[1.0296  8.8209  3.9204]
```

```
In [14]: # argmax is used to find the index with highest value.
# It translates scores into a decision: highest score is the winner
# 0 means x, 1 means plus, 2 means zero since that is how NN was defined
print(np.argmax(np.matmul(NN,x)))
print(np.argmax(np.matmul(NN,zero)))
print(np.argmax(np.matmul(NN,plus)))
```

```
0
2
1
```

- You should combine the three recognizers that you have in one neural network (matrix NN).
- NN has three outputs.
- It returns (a,b,c) with the score that the input character is your first character ('x' in my case), b the second character (+) and c the third character ('o').
- Argmax is a nice function to know which number is highest




Coding tasks

1. Define three characters from your own name, and draw them in a 5x5 matrix. E.g. if your name is Sieuwert, make SIE, or SUT, ... Be unique.
2. Create four variations for each character. Make variation one of each character by redrawing it differently, e.g. emphasizing one of the strokes. Variant 2 should be based on slightly blurring the first version. Variant 3 should add some noise to a few pixels of the image. Variant four should combine noise and blurring.
3. Make a correlation matrix for your inputs (12 x 12 matrix) by taking inproducts between all input values. How similar are your inputs to each other? Explain from the correlation matrix which two characters are most likely to be confused.
4. Create a matrix NN1 to recognise your three characters. It should work on all variations, so perhaps use a combination/average version of the three characters. Make improvements to find best matrix. Try to make it so that the matrix gives an equally high output for each input character, to make comparison easy.
5. Test your matrix on all your inputs and show the scores, for instance in bar chart. Evaluate the score: does the correct answer indeed get the highest score? Is the difference between the scores big enough to set a simple threshold value?
6. Really test your network: make four or more inputs and use NN1 on it. Find multiple inputs (3 or more) that are not correctly classified. Check what happens if you input all 1's or all zeros? Try to make an incorrectly classified character by changing only one pixel. Is this possible? Can you do it by only changing two pixels? Three pixels?
7. Find multiple inputs (not all zeros) so that NN1 cannot make a decision: it gives exactly equal values for all characters. Is there a method for finding such inputs? Describe how you can create such counterexamples.

After completing the coding, create a report (word or latex based PDF) that includes the following:

- Title page with unique, relevant title like a research paper and your names
- An introduction where you explain what optical character recognition is and cite three or more different scientific papers on optical character recognition
- A section where you show your three characters and four variations and explain the task. Explain how you created the variations and why these are relevant.
- A section results where you show the correlation matrix, and NN1 and you show using bar charts how the characters were recognized. Try to show a threshold value in the chart. Also show the other testing results. Explain how NN1 was created.
- A section explanation where you explain whether the accuracy in your view is good enough for actual use, you comment on how it could be improved further. Also comment on whether it was difficult to find inputs that were not correctly classified. Finally, comment on what thresholds would be if you could add a “Do not know” outcome.
- Include your source code as appendix

Assignment rubric

NeuralNetwork   			
Criteria	Ratings		Pts
Title page and layout	5 Pts Full marks Report also looks and reads like a professional report with good layout and good title	0 Pts No marks	5 pts
Introduction and references	6 Pts Full marks Good explanation of the history of OCR with interesting sources.	0 Pts No marks	6 pts
task explanation	6 Pts Full marks The three characters are clear and there are interesting variations and good explanation that allows us to make similar variations for other characters	0 Pts No marks	6 pts
Results	8 Pts Full marks There are very clear charts with axis titles, consistent color use and a line to indicate suggested threshold. NN1 was created in a structured way that could be repeated for other characters. Code is clear	0 Pts No marks	8 pts
Explanation	5 Pts Full marks There is a clear explanation on the achieved accuracy, the test results and good recommendations for the "I don't know" threshold	0 Pts No marks	5 pts
Total points: 30			