Artificial Intelligence/Machine Learning/Deep Learning: 'Bridging the Skills Gap'

Linear Algebra Refresher

In this video we will go over some basic math concepts that are the foundation of ML. This video is not a math course, but it aims at bringing everybody to the same level, enough to understand the majority of the algorithms.

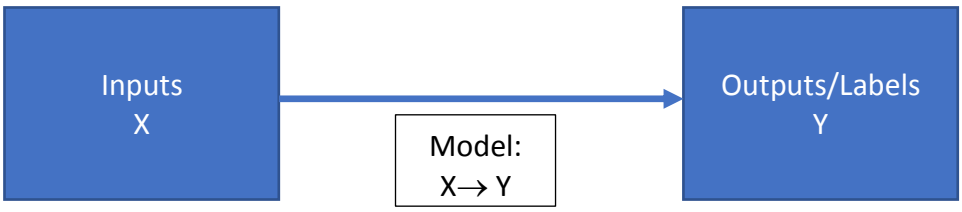
Topics of this session include:

1. **Scalar**
2. **Vector**, Vector Space, Norm/Length of a Vector, Vector Operations, Basis, Eigenvectors & Eigenvalues
3. **Matrix**, Matrix Operations, Transpose of a Matrix, Inverse of a Matrix, Pseudo-inverse of a Matrix, Rank of a Matrix, Determinant, Singular Matrix
4. **Tensor**

If you are familiar with these concepts, you can skip this video.

Many people claim that you do not need to understand the math behind the models and that ML is more art than it is science. I do not agree with that statement. It is true that **high-level APIs** like keras have made DL more accessible and we expect that future tools will make it even easier to develop models. However, when you understand the math behind the models you will come across more credible and you will be able to lead/manage business ML projects.

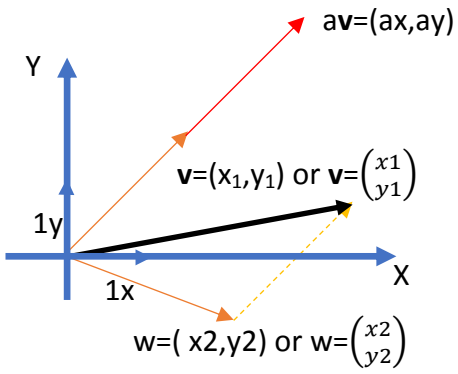
In lesson 1 we have seen that ML is about a computer learning a **transformation** that maps input data to output data. Understanding **Linear Algebra** is a necessity to understand ML. Tensorflow, Google's machinery for DL eats tensors for breakfast.



Example: assume you have following data samples: (0,2), (1,6), (2,10), (3,14)
The transformation learned by the computer is: y=4x+2 (model)
If we feed the computer with a new input (x=25) we can derive y as: 4*25+2=102

Scalar: a number (only has a magnitude, no direction)

Vector:
A vector **v** (or $\bar{v}$) has a **direction** and a **magnitude**



A vector space (also called a linear space) is a collection of vectors **v**, **w** which may be added together and multiplied (scaled) by a scalar a. The term Linear in Linear Algebra refers to the additivity and scaling:

Let: **v, w** $\in \mathbb{R}^2$ and a $\in \mathbb{R}$:
additivity: **v**+**w** $\in \mathbb{R}^2$
scaling: a**v** $\in \mathbb{R}^2$ and a**v** is new vector in direction of **v** if a > 0 or in opposite direction if a < 0 and with length a$|\boldsymbol{v}|$

A set of vectors $\mathbf{1_x}$, $\mathbf{1_y}$ form a basis when they are linearly independent and when they span the space. n vectors are linear independent if they cannot be expressed as a linear combination of each other – if it is the case the vectors span the whole n-dim space. By taking linear combination of n vectors one can produce all vectors in the n-dim space.

Any 2 linear independent vectors form the basis for a 2-dim space

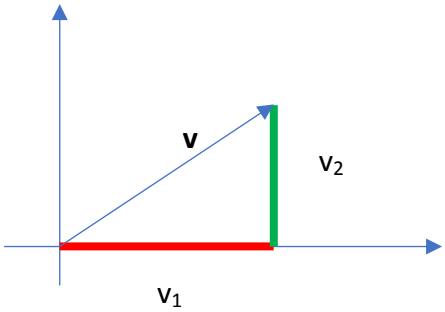
Transpose of a vector – row vector (1xn) vs. column vector (nx1)

Norm or length of a vector **v**: measure for its magnitude

$\|v\|=\sqrt{x^2 + y^2}$ (Pythagoras – quick explanation using above drawing). Also $\|v\| = \sqrt{\boldsymbol{v^T}.\boldsymbol{v}}$ = $\sqrt{\boldsymbol{v}.\boldsymbol{v^T}}$

v $\in \mathbb{R}^{n\,x\,1}=\begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}$

$l_2\ norm$ or Euclidean norm $\|v\|_2 = \sqrt{\|\ v_1\ \|^2 + \|\ v_2\ \|^2 + \cdots + \|\ v_n\ \|^2}$ ML uses mainly l_2
$l_1\ norm$ (Manhattan distance) $\|v\|_1 = |v_1| + |v_2| + \ldots + |v_n|$



Vector norms are often used in ML as a *measure of distance*

Dot Product of 2 Vectors → scalar
The dot product is often used as a *measure for similarity*

$$\boldsymbol{v_1} = \begin{pmatrix} v_{11} \\ \vdots \\ v_{1n} \end{pmatrix} \text{ and } \boldsymbol{v_2} = \begin{pmatrix} v_{21} \\ \vdots \\ v_{2n} \end{pmatrix}$$

$$\boldsymbol{v_1}.\boldsymbol{v_2} = \boldsymbol{v_1^T}.\boldsymbol{v_2} = (v_{11} \quad \cdots \quad v_{1n}).\begin{pmatrix} v_{21} \\ \vdots \\ v_{2n} \end{pmatrix} = \sum_{k=1}^{n} v_{1k}\, v_{2k}$$

Drawing: dot product is projection of 1 vector onto the other
$\boldsymbol{v_1}.\boldsymbol{v_2}=|\boldsymbol{v_1}|\,|\boldsymbol{v_2}|\,\boldsymbol{cos}(\boldsymbol{\theta})$ where $\boldsymbol{\theta}$ is the angle between 2 vectors

Zero vector: **0**=(0,0) in 2D or **0**=(0,0,0) in 3D and **v**+**0**=**v**

Unit vector: $\overline{\mathbf{1}}=\frac{v}{\|v\|}$

Matrix:
A matrix $A_{m\,x\,n} \in \mathbb{R}^{m\,x\,n}$ has m rows and n columns. The columns can be seen as column vectors stacked side by side. Example matrix notation of 3D space with unit vectors

$$A_{m\,x\,n} = \begin{pmatrix} 1 & \cdots & n \\ \vdots & \ddots & \vdots \\ m & \cdots & \end{pmatrix}$$

Element-wise Addition & Subtraction of Matrices

$$A_{2\,x\,2} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \text{ and } B_{2\,x\,2} = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}$$

$$A + B = \begin{pmatrix} 1+5 & 2+6 \\ 3+7 & 4+8 \end{pmatrix} = \begin{pmatrix} 6 & 8 \\ 10 & 12 \end{pmatrix}$$

$$A - B = \begin{pmatrix} 1-5 & 2-6 \\ 3-7 & 4-8 \end{pmatrix} = \begin{pmatrix} -4 & -4 \\ -4 & -4 \end{pmatrix}$$

Multiplication of Matrices

$A_{m\,x\,n}$ and $B_{p\,x\,q}$: $C_{m\,x\,q} = A\,x\,B$ and n must be equal to p!

$A_{2\,x\,2} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ and $B_{2\,x\,2} = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}$

$AxB_{\,2\,x\,2} = \begin{pmatrix} 1x5 + 2x7 & 1x6 + 2x8 \\ 3x5 + 4x7 & 3x6 + 4x8 \end{pmatrix} = \begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}$

A matrix can be seen as a linear transformation operator in space. Assume a vector $\mathbf{v}^T = (1,1)$ with dimension (1x2). This vector can be transformed to another space by a matrix operator (2x3):

$(1,1)\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} = (5,7,9)$

A non-square matrix changes the dimension of the input vector from 2D to 3D in this case!

Transpose of a Matrix

$A_{2\,x\,3} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \Rightarrow A^T_{3\,x\,2} = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$

$(AB)^T = B^T A^T$

Rank of a matrix = number of linear independent column vectors

$A = \begin{pmatrix} 1 & 3 & 4 \\ 2 & 5 & 7 \\ 3 & 7 & 10 \end{pmatrix}$

Rank of matrix A = 2 because column 3 is a linear combination (a sum in this case) of columns 1 and 2. The column vectors of A only span a 2-dim subspace of a 3-dim space

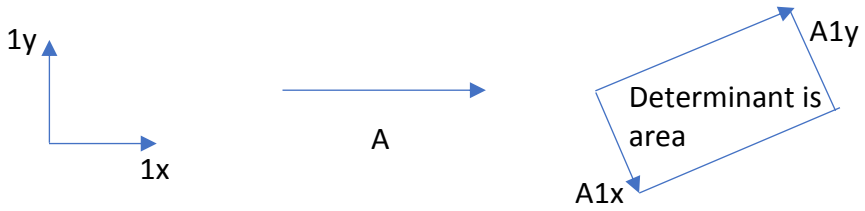
Determinant of a matrix: → scalar

$A_{2\,x\,2} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$

Det(A)= = 1x4-2x3=-2

Det(A) ≠0 only if all rows and columns are linear independent

The determinant is the "size" of the output transformation. If the input was a unit vector, the determinant is the size of the transformed area or volume. A determinant of 0 means matrix is "destructive" and cannot be reversed.



Singular matrix:
A square matrix A ∈ $\mathbb{R}^{n\,x\,n}$ is full rank if rank of A is n.
If A is not full rank → A is called a Singular matrix (rows or columns are not linear independent)
Singular matrix has no inverse and a ZERO determinant

Identity matrix:

$$I=\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}$$

AI=IA=A (commutative when one of the matrices is I

Inverse of a matrix: only for square matrices!
A square matrix A ∈ $\mathbb{R}^{n\,x\,n}$
$A^{-1}.A = I$

Not all square matrices have an inverse (for example if matrix is Singular)
$(AB)^{-1} = B^{-1}A^{-1}$

Pseudo Inverse of a matrix: use when inverse does not exist

But if A ∈ $\mathbb{R}^{m\,x\,n}$, the inverse of A does not exist as A is not square matrix

In this case we use a pseudo inverse → **x**= $(A^T A)A^T$. $\boldsymbol{b}$ (if A is not Singular) and $(A^T A)A^T$ is called the pseudo inverse of A

Matrix working on a Vector: Eigenvectors and Eigenvalues:

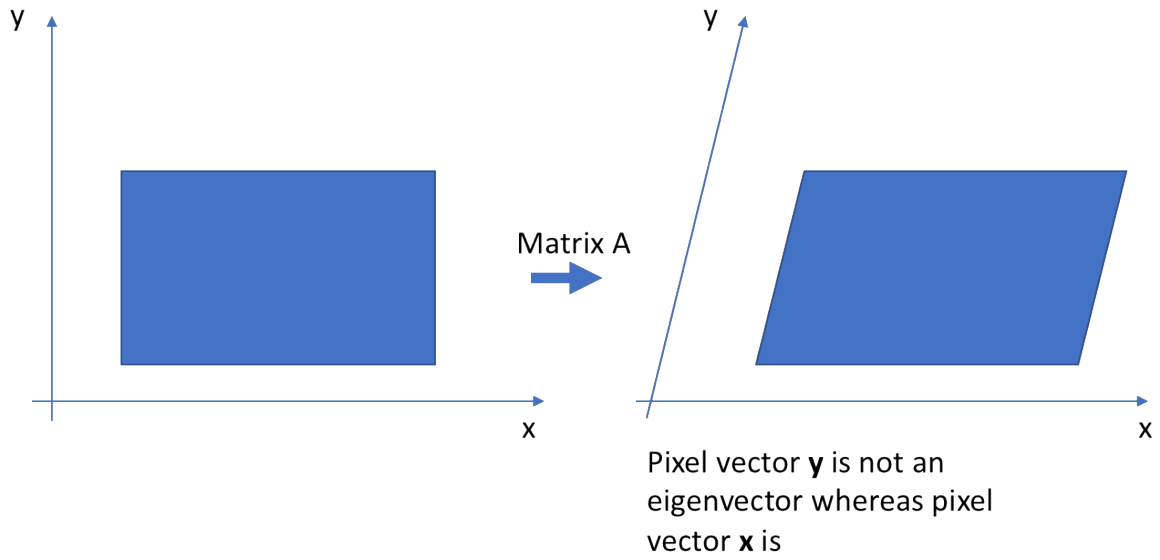
Important concept in ML (Support Vector Machines, Principle Components Analysis)

A ∈ $\mathbb{R}^{n\,x\,n}$ and **v** ∈ $\mathbb{R}^{n\,x\,1}$=$\begin{pmatrix}v_1\\ \vdots\\ v_n\end{pmatrix}$

When you have a matrix A working on a vector **v**, generally the magnitude and direction of the new vector is different from the original vector.
Vectors that are unaffected by a matrix transformation A are called eigenvectors (same direction or opposite direction from original vector **v**. Eigenvectors must be non-zero!

The magnitude of the eigenvectors is called the eigenvalue

A**v**=$\lambda\boldsymbol{v}$



Characteristic equation of a matrix provides the eigenvalues of the matrix.

A**v**=$\lambda \boldsymbol{v} \Longrightarrow$ A**v**-$\lambda \boldsymbol{v} = \mathbf{0} \Longrightarrow$ A**v**-$\lambda I_n \boldsymbol{v} = \mathbf{0}$ in other words de matrix (A-λI_n) takes **v** into the **0**-vector (assume **v** is not **0**) which means that A cannot have an inverse (cannot divide by zero!) $\Longrightarrow$ A-λI must be singular and det (A-λI)=0 and the roots of this characteristic equation gives us the eigenvalues

Tensor:

Google's Tensorflow and Numpy rely on tensors to do their magic and therefor all input data must be in tensor format. A tensor is nothing more than a multi-dimensional array of numbers.

Tensor Rank	Form	Comment
0	Scalar	A scalar is nothing more than a number. A scalar only has a magnitude: [10]
1	Vector	A vector is an array of numbers. A vector has a magnitude and a direction. A vector can be represented as [10,4,2]
2	Matrix	A Matrix is a 2-dimensional array with rows and columns. A matrix is nothing more than a set of column vectors stacked next to each other: [[1,2,3],[4,5,6]]
3 and up	Tensor	A tensor is a 3D array or a 3D block of numbers [[[1,2],[3,4]],[[5,6],[7,8]]]

Examples:

1. Image in greyscale is stored in a matrix form: size of the image determines matrix size and each matrix cell holds a value from 0 $\rightarrow$ 255 representing pixel intensity
2. Image in color is stored in a tensor and 3rd dimension is RGB (Red, Green, Blue)

In general, the first axis, axis=0, is the sample axis or the batch axis
The data you'll manipulate will almost always fall into one of the following categories:

- Vector data—2D tensors of shape (samples, features)
- Timeseries data or sequence data—3D tensors of shape (samples, timesteps, features) - whenever time matters you should store it in a 3D tensor with an explicit time axis - time axis is always axis=1 as per convention
- Images—4D tensors of shape (samples, height, width, channels) - a batch of 128 gray-scale images of size 256 x 256 is stored in a 3D tensor (128, 256, 256, 1).....if color images (128, 256, 256, 3) - Tensorflow uses channel-last convention
- Video —5D tensors of shape (samples, frames, height, width, channels)

For instance, a 60-second, 144 × 256 YouTube video clip sampled at 4 frames per second would have 240 frames. A batch of four such video clips would be stored in a tensor of shape (4, 240, 144, 256, 3). That's a total of 106,168,320 values! If the data type of the tensor was float32, then each value would be stored in 32 bits, so the tensor would represent 405 MB. Heavy! Videos you encounter in real life are much lighter, because they aren't stored in float32, and they're typically compressed by a large factor (such as in the MPEG format).