Linear Algebra 101 for Deep Learning

Learning Math and Deep Learning with Intuitions

Sandeep Krishnamurthy - @skm4ml

AWS Deep Learning

Disclaimer

- Mathematical rigor in notations, proofs etc… is not the focus of my talk.
- This is not the complete guide for math behind Deep Learning. However, I try to scratch the surface and provide foundations for you to get started with ideas behind Deep Learning.
- Topics are too simplified. Do not be in a state of know all by end of this talk ;-) There is a lot beyond this talk.

Principle behind the talk

Numerical representation is required for computers.

Visual representation is required for humans.

Basics

1. Scalars
2. Vectors
3. Matrices
4. Tensors

Everything is just a container **or bucket of data**

Scalars

- A Scalar is just a number
- A data container with only **one value**

$$a = 10$$

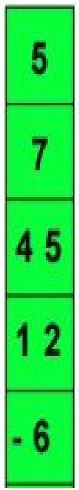
- **0** Dimensions (**Axes**)

Example

Zipcode = 95051

Vectors

- Do you remember “**array**” from your programming world?
- Vectors are array or **list** of numbers/data
- A data container with list of values
- **1 Dimensional => 1 axis**



Ref - http://bit.ly/2wslJWq

Example

Features of a Person

person = [FName, LName, Email, Mobile,Zipcode]

Matrices

- For programmers - Array of arrays? List of lists?
- Matrix is Vector of Vectors
- A data container with **list of list** of data
- **2 Dimensional => 2 Axes**
- Rows and Columns

-9	4	2	5	7
3	0	12	8	61
1	23	-6	45	2
22	3	-1	72	6

Ref - http://bit.ly/2wslJWq

Example

How about having 1000 people’s features.

Person1 = [FName1, LName1, Email1, Mobile1, Zipcode1]

Person2 = [FName2, LName2, Email2, Mobile2, Zipcode2]

People = [[FName1, LName1, Email1, Mobile1, Zipcode1],

[FName2, LName2, Email2, Mobile2, Zipcode2]

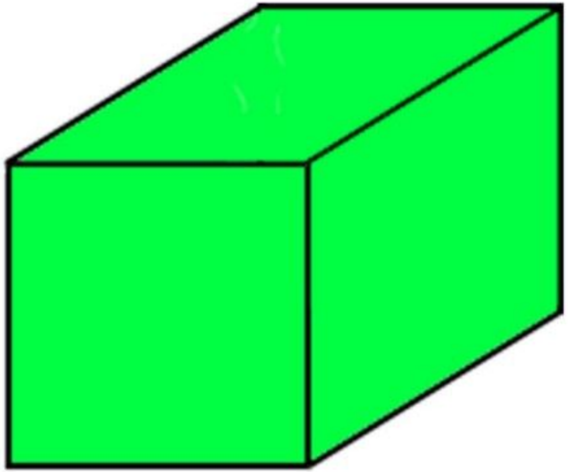
..................]

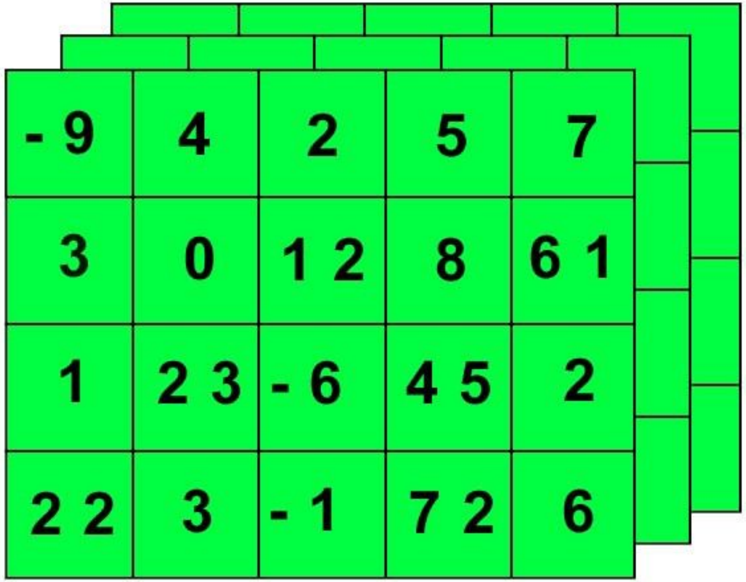
Shape = (1000, 5)

Tensors

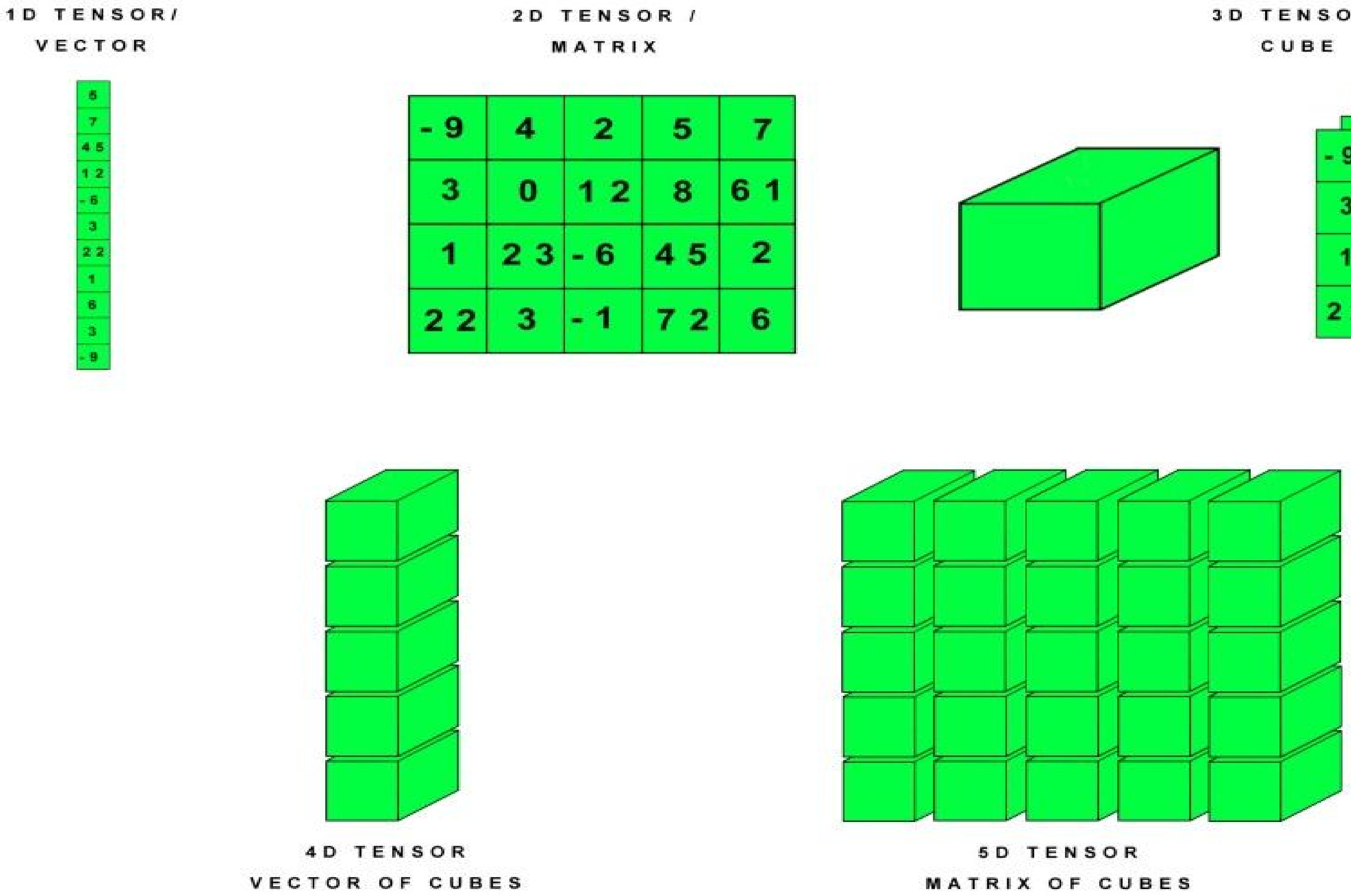
- **Generalized data containers**
- When dimensions (axes) >= 3 it is generalized as Tensors
- Scalar -> 0D Tensor
- Vector -> 1D Tensor
- Matrix -> 2D Tensor
- And so on… ND Tensor

3D Tensor





Ref - http://bit.ly/2wslJWq



Ref - http://bit.ly/2wslJWq

Common Data in Tensors

For better understanding, we can visualize some common data stored in different Tensors

- 3D -> Time Series
- 4D -> Images
- 5D -> Videos

Time Series -> 3D Tensor

Stock Price Data

(time, stock_price) => 2D

(company, time, stock_price) => 3D

In most cases => **Tensor Dimension = Actual Data Dimension + 1 (sample_size)**

Images - 4D Tensor

(channel, width, height) => 3D

(image_list, channel, width, height) => 4D => list of images => list of 3D Tensor => 4D Tensor

Important Tensor Operations

- Tensor - Scalar operations
- Elementwise operations
- Tensor - Vector operations
- Transpose
- Dot products
- Matrix - Vector products
- Matrix - Matrix products
- Norm

Tensor - Scalar Operations

- A Scalar operation on each element of a Tensor.

A = [1,2,3,4]

A*2 => [1*2, 2*2, 3*2, 4*2] => [2, 4, 6, 8]

Elementwise Operations

- Given 2 identically shaped Tensors apply element wise operation (+ - * /)

A = [1, 2, 3, 4] B = [1, 1, 1, 1]

C = A + B => [2, 3, 4, 5]

Tensor - Vector Operations

$$C = A+b, \text{ where } C(i,j) = A(i,j) + b(j)$$

- Vector b is applied to each row of the matrix (2D Tensor) A
- This is called **broadcasting** operation

Transpose

- For simplicity let us take 2D Tensor (Matrix)
- A´(i,j) = A(j,i)

$$\begin{bmatrix} 6 & 4 & 24 \\ 1 & -9 & 8 \end{bmatrix}^T = \begin{bmatrix} 6 & 1 \\ 4 & -9 \\ 24 & 8 \end{bmatrix}$$

Ref: http://bit.ly/2yACZJO

Dot Products

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \bullet \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

Ref: http://bit.ly/2x6zfU1

Matrix-Vector Products

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix}$$

Ref: http://bit.ly/2yhVJS2

Matrix-Vector Products

$$A\mathbf{x} = \begin{pmatrix} \cdots & \mathbf{a}_1^T & \cdots \\ \cdots & \mathbf{a}_2^T & \cdots \\ & \vdots & \\ \cdots & \mathbf{a}_n^T & \cdots \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} = \begin{pmatrix} \mathbf{a}_1^T\mathbf{x} \\ \mathbf{a}_2^T\mathbf{x} \\ \vdots \\ \mathbf{a}_n^T\mathbf{x} \end{pmatrix}$$

Ref: http://bit.ly/2yhVJS2

Matrix-Matrix Products

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nk} \end{pmatrix}, \quad B = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{k1} & b_{k2} & \cdots & b_{km} \end{pmatrix}$$

Ref: http://bit.ly/2yhVJS2

Matrix-Matrix Products

$$C = AB = \begin{pmatrix} \cdots & \mathbf{a}_1^T & \cdots \\ \cdots & \mathbf{a}_2^T & \cdots \\ & \vdots & \\ \cdots & \mathbf{a}_n^T & \cdots \end{pmatrix} \begin{pmatrix} \vdots & \vdots & & \vdots \\ \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_m \\ \vdots & \vdots & & \vdots \end{pmatrix} = \begin{pmatrix} \mathbf{a}_1^T\mathbf{b}_1 & \mathbf{a}_1^T\mathbf{b}_2 & \cdots & \mathbf{a}_1^T\mathbf{b}_m \\ \mathbf{a}_2^T\mathbf{b}_1 & \mathbf{a}_2^T\mathbf{b}_2 & \cdots & \mathbf{a}_2^T\mathbf{b}_m \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{a}_n^T\mathbf{b}_1 & \mathbf{a}_n^T\mathbf{b}_2 & \cdots & \mathbf{a}_n^T\mathbf{b}_m \end{pmatrix}$$

Ref: http://bit.ly/2yhVJS2

Norm

- We need some way of quantifying our matrix or tensors
- When I give you a matrix how do you quantify it?
- We use Norm for quantifying our Tensors
- Multiple types of Norms
- L1, L2 are most common
- Represented as || . ||
    - Example: Norm of Tensor X is represented as || X ||

L1 Norm

- Sum of absolute values of the tensor
- Denoted as
  - ℓ_1 norm

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \qquad |\mathbf{x}|_1 = \sum_{r=1}^{n} |x_r|.$$

Ref: http://bit.ly/2x7cAHk

L2 Norm

- Sum of squared values of the tensor
- Sounds like Euclidean distance, Pythagoras theorem?
- Visualize it as a measure of distance => Quantifying the tensor
- Denoted as
  - ℓ2 norm

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$|\mathbf{x}| = \sqrt{\sum_{k=1}^{n} |x_k|^2},$$

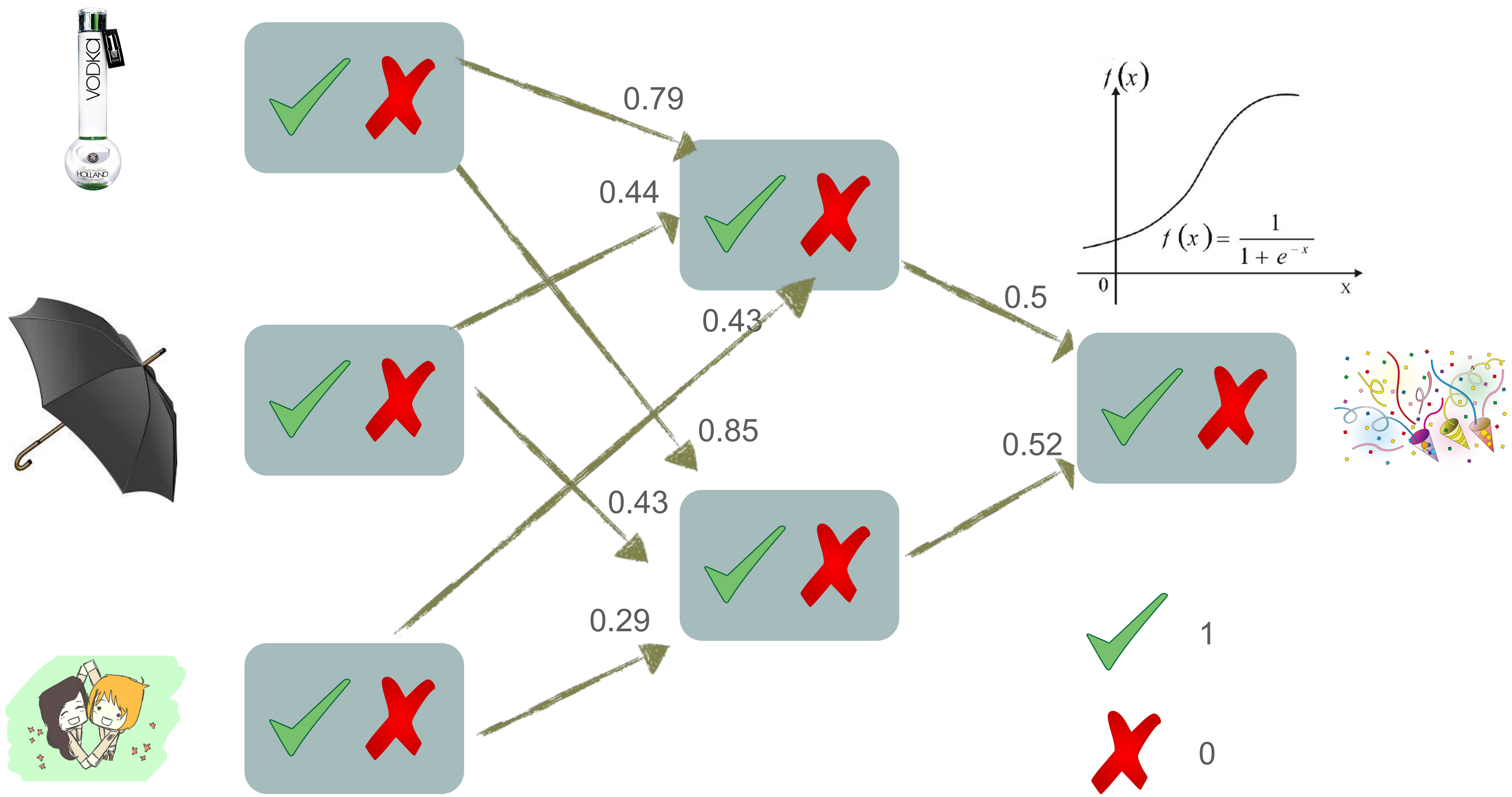
$$|\mathbf{x}| = \sqrt{x_1^2 + x_2^2 + x_3^2}.$$

Ref: http://bit.ly/2fHip6y

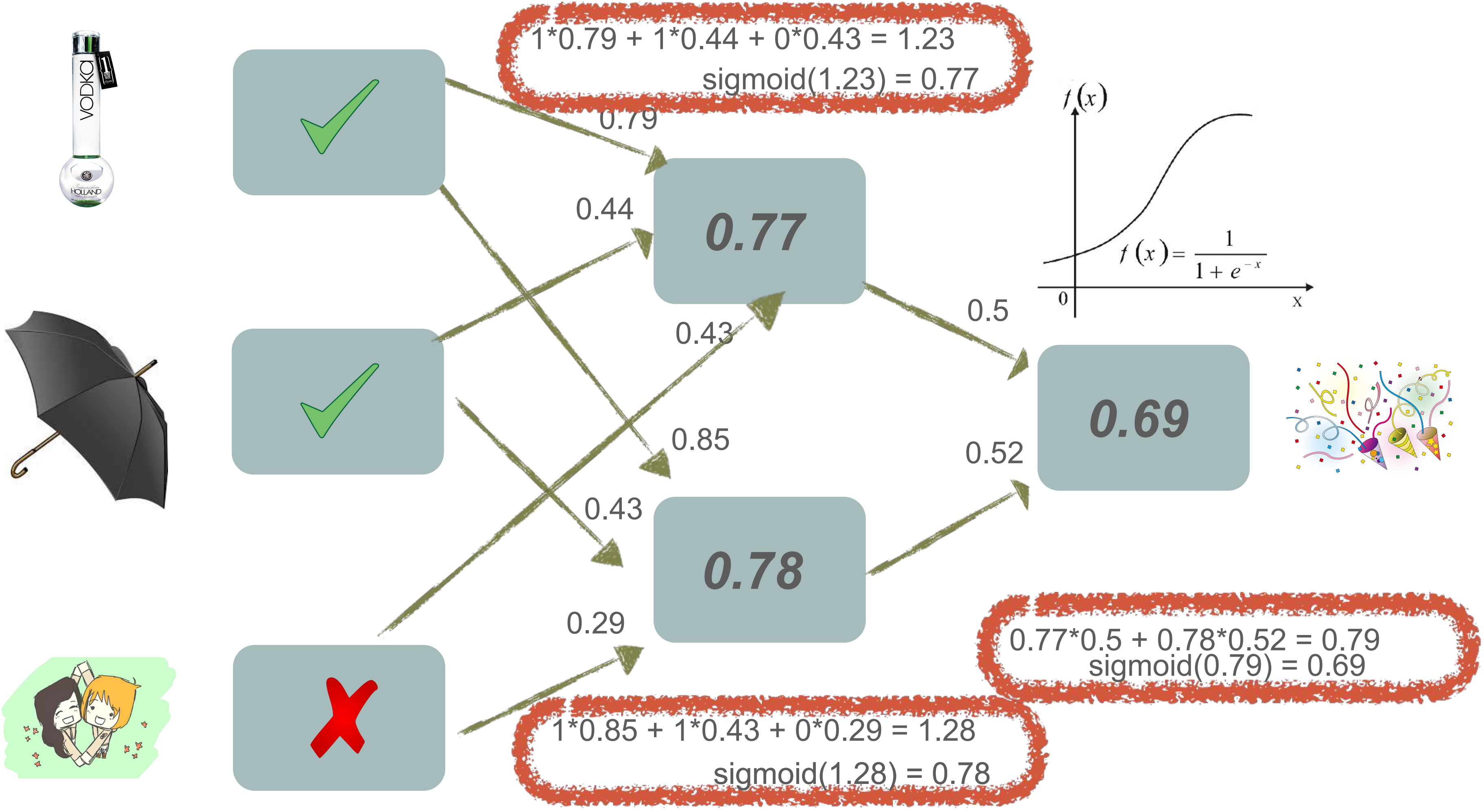
Ok, but, wait, why do I have to know all these tensors, products, transpose and so on… to start with Deep Learning?

Tensor to Deep Learning

Remember our previous example?? Helping Slava to decide to go to a party or not?



VODKA
HOLLAND
0.79
0.44
0.43
0.85
0.43
0.29
0.5
0.52
$f(x)$
$f(x) = \frac{1}{1+e^{-x}}$
0
x
1
0


1*0.79 + 1*0.44 + 0*0.43 = 1.23
sigmoid(1.23) = 0.77
VODKA
HOLLAND
0.79
0.44
0.43
0.85
0.43
0.29
0.77
0.78
0.5
0.52
0.69
f(x)
f(x) = 1/(1 + e^-x)
0
x
0.77*0.5 + 0.78*0.52 = 0.79
sigmoid(0.79) = 0.69
1*0.85 + 1*0.43 + 0*0.29 = 1.28
sigmoid(1.28) = 0.78

$$\text{Input} = \begin{bmatrix} \text{vodka} \\ \text{Rain} \\ \text{Party} \end{bmatrix} \qquad W_{11}^{1} = \begin{bmatrix} 0.79 \\ 0.44 \\ 0.43 \end{bmatrix} \qquad W_{12}^{2} = \begin{bmatrix} 0.85 \\ 0.43 \\ 0.29 \end{bmatrix}$$

$$W_{23}^{1} = \begin{bmatrix} 0.5 \\ 0.52 \end{bmatrix}$$

Example 1

$X = \text{Input} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$

Layer 1

$$W^T \odot X$$

① $\begin{bmatrix} 0.79 & 0.44 & 0.43 \end{bmatrix} \odot \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = 0.79*1 + 0.44*1 + 0.43*0$

$= 1.23$

$\text{Sigmoid}(1.23) = \underline{0.77}$

② $\begin{bmatrix} 0.85 & 0.43 & 0.29 \end{bmatrix} \odot \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = 0.85*1 + 0.43*1 + 0.29*0$

$= 1.28$

$\text{Sigmoid}(1.28) = \underline{0.78}$

(3) $$\begin{bmatrix} 0.5 & 0.52 \end{bmatrix} \odot \begin{bmatrix} 0.77 \\ 0.78 \end{bmatrix} = 0.5 * 0.77 + 0.52 * 0.78$$

$$= 0.79$$

$$= \text{sigmoid}(0.79) = 0.69$$

Coding Time!
Let us get started with Apache MXNet

```
$ pip install mxnet --pre
$ pip install jupyter
$ jupyter notebook
```

Next Steps

- Intuitions and Visualization for commonly used mathematical jargons
- We define and answer basic topics:
  - What is a function? Linear v/s Nonlinear?
  - What is Learning or Training?
  - What is Model?
  - What is Loss?
  - What is Optimization?
  - What are Gradients?
  - What is Gradient Descent? Stochastic Gradient Descent?
  - And more....
- We train a Hand Written Digit Recognition model with Apache MXNet and revisit all the topics defined above.

Resources

- Read more about Deep Learning along with Learning MXNet - http://gluon.mxnet.io/index.html
- Slides and Jupyter Notebook available at - https://github.com/awsaiguru/resources/tree/master/presentations/awsaiguru_meetup_3_paloalto_Oct_2017
    - $ git clone https://github.com/awsaiguru/resources/ .