**Obsah**

[1. Mathematics 2](#_Toc6838032)

[1.1. What is a tensor? 2](#_Toc6838033)

[2. Machine Learning 101 5](#_Toc6838034)

[2.1. Anaconda – activate/deactivate a virtual environment 5](#_Toc6838035)

[2.2. Jupyter shortcuts 5](#_Toc6838036)

[2.3. Neural Network 5](#_Toc6838037)

[2.3.1. Perceptron 6](#_Toc6838038)

[2.3.2. Weight 6](#_Toc6838039)

[2.3.3. Activation function 6](#_Toc6838040)

[2.3.4. Bias 8](#_Toc6838041)

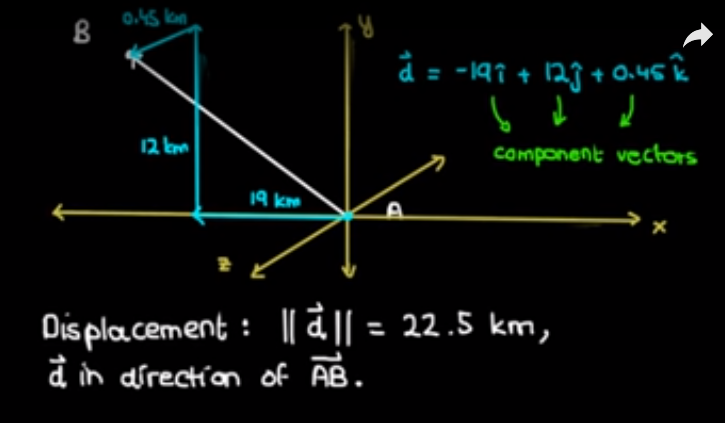
[2.3.5. Cost functions 8](#_Toc6838042)

[2.3.6. The actual “learning”, Gradient descent, Backpropagation 9](#_Toc6838043)

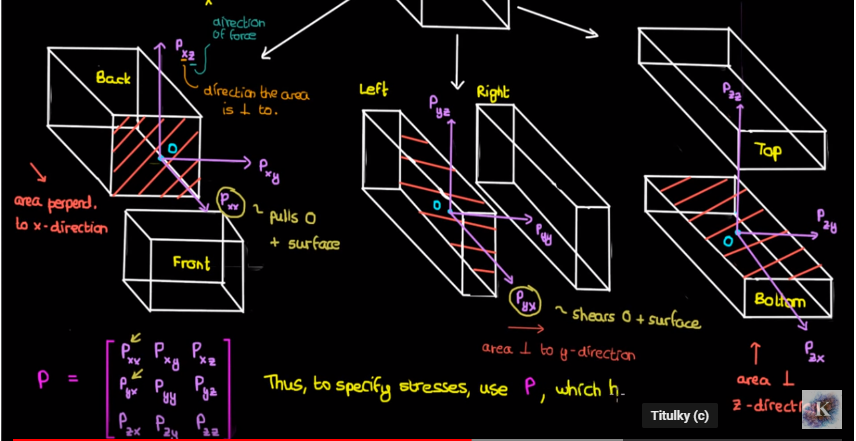
[3. Tensorflow 11](#_Toc6838044)

[3.1. General 11](#_Toc6838045)

1. Mathematics
   1. What is a tensor?



* Displacement
* Direction
* Unit vector, component vector (1 basis vector (I, j , k) / component )



Tensor: in an m-dimensional space, a tensor of rank n is a mathematical object that has n indices, m^n component, and obeys certain transformation rules. Generally, m=3 except Einstein’s theory, where time is the 4th dimension.

Rank of a tensor: number of basis vectors needed to fully specify the component of the tensor.

Scanal – rank 0

Vector – rank 1

Misconception – rank 2 tensors and matrices are the same.

Transformation rules:

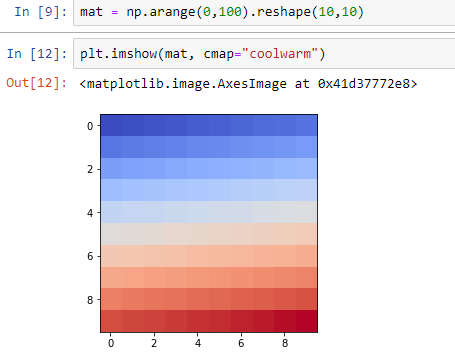
1. A tenor is an object that transforms like a tensor
2. A tensor is an object that is invariant under a change of coordinate systems, with components that change accordingly to a special set of mathematical formula.

(no mather what coordinate system we use to locate NY, the temperature remains the same)

(the thing that I change the coordinate system does not change the fact that a given vector points from A to B, so it is accurate to say that the vector is invariant when we change coordinate system)

* + - 1. Random
* Numpy masking (Boolean masking) *mat[mat>50]*
* Pandas: read csv later -> dp.as\_matrix() // returns numpy array
* Shift + tab – jupyter shortcut to see documentation
* %matplotlib inline
* Pandas – built on numpy, lots of same functionality.

* Matpltlib.pyplot – imshow function:



* Df.plot – Pandas has his own visualization on top of matplotlib.
* Plots are not that smart – if you not write it in the same cell will not work!
* Matmul and .dot method
* \* - Astrix
* Self , \_\_init\_\_

1. Machine Learning 101
2. Numpy – numerical processing
3. Pandas – data analysis
4. Matplotlib – data visualization
5. Scikit learn – processing ml data sets

Come concepts I have no idea about now:

* perceptrons,
* activation functions,
* back propagation,
* Densely Connected NN,
* Convolutional NN,
* Recurrent NN, Word2Vec,
* AutoEncoders,
* generative Adversarial Networks,
* Reinforcement Learning with OpenAI Gym )

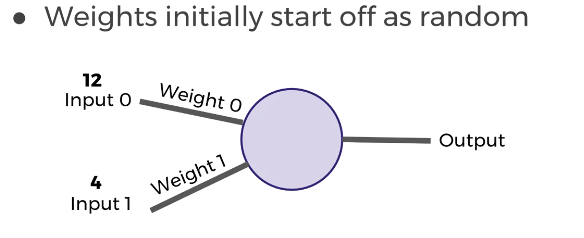
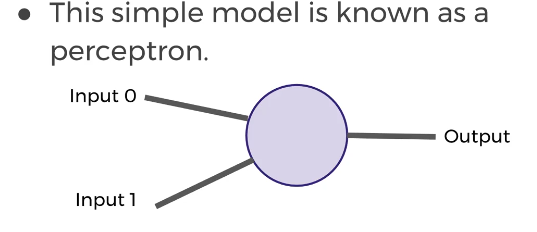
Supervised Learning – uses labeled data to predict a label given some features. If the label is continuous its called a regression problem, if its categorical it is a classification problem. E.g. price of house – label is categorical.

* 1. Anaconda – activate/deactivate a virtual environment
* Conda info –envs

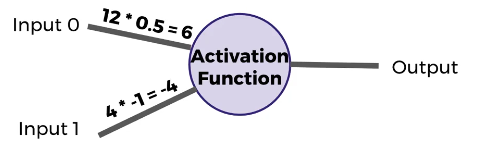
Crate environment with different python versions by 1 command

Clone envs, list the dependeniceis, recover from yml file.

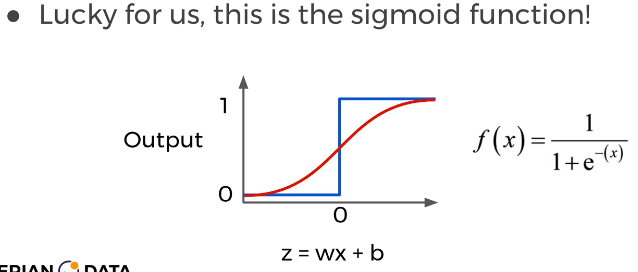
* 1. Jupyter shortcuts
* Rename
* Tab
* Shift tab
* Shift enter
* Run cell to convert from markdown to text
  1. Neural Network
* Neurons
* Activation Functions
* Cost Functions
* Gradient Descent
* Backpropagation
  + 1. Perceptron
    2. Weight



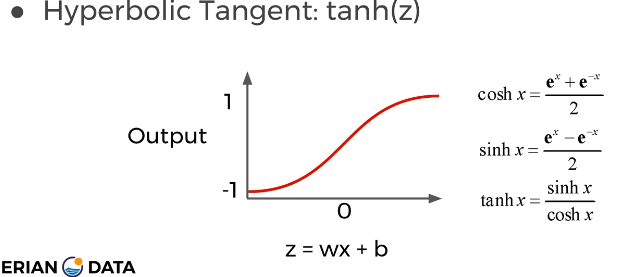
* + 1. Activation function



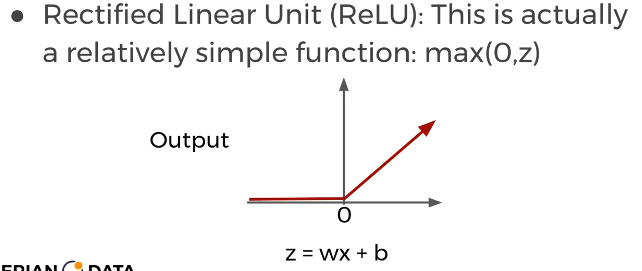
Example: if sum of the inputs is positive return 1, else 0



<0;1>



<-1;1>

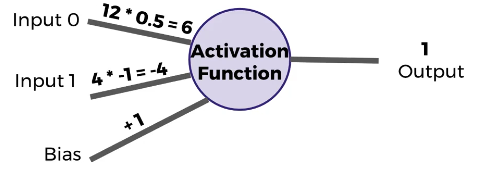


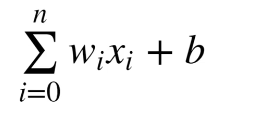
<0;z>

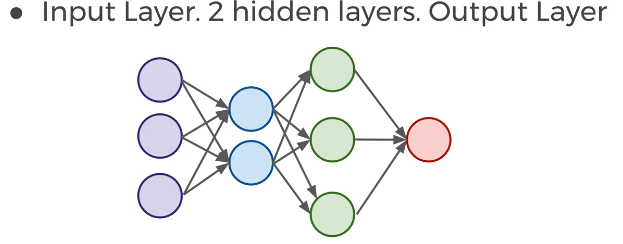
Changing the activation function used can be beneficial depending on the task

* + 1. Bias

What if both inputs were true, then the weight has no meaning.

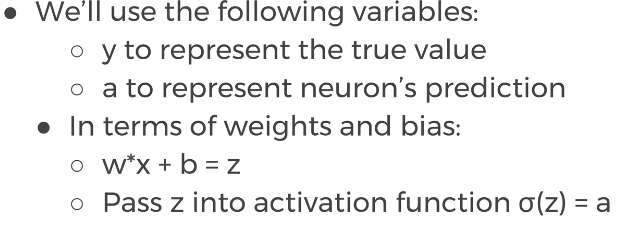


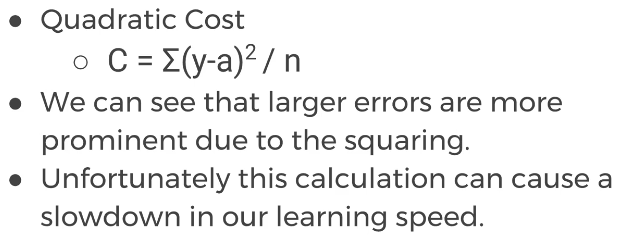


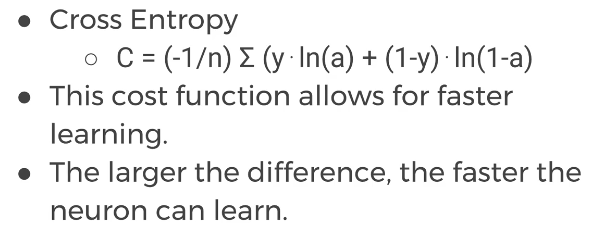


* Input Layer: real values from the data
* Hidden Layers: Layers in between the input and output, 3+ is “deep network”
* Output Layer: Final estimate of the output
  + 1. Cost functions

To measure how well these neurons are preforming. We can use a cost function to measure how far off we are from the expected value.







* + 1. The actual “learning”, Gradient descent, Backpropagation

We need to figure out how we can use our neurons and the measurement of error (cost functions) and then attempt to correct our prediction, in other words, “learn”!

**Gradient descent:** is an optimization algorithm for finding the minimum of a function. (useful for us because we want to minimalize the cost function). To find a local minimum, we take steps proportional to the negative of the gradient. Using gradient descent we can figure out the best parameters for minimizing our cost, for example, finding the best values for the weight of the neuron inputs.

**Backpropagation**: How can we quickly adjust the optimal parameters or weights across our entire network? It is used to calculate the error contribution of each neuron after a batch of data is processed. It relies heavily through the network and calculate these errors. Backpropagation works by calculating the rorr at the output and then distributes back through the network layers. It requires a known desired output for each input value (supervised learning) (chain rule, derivatives)

TODO playground.tensorflow.org

1. Tensorflow
   1. General

