**Week 3 – Notes**

**Neural Networks Overview**

Superscript inside square brackets represents the notation for the number of the layer

**Neural Network Representation**

Each layer, including the input and the output can be seen as activation layers

Therefore, each layer is denoted by a and a superscript inside square brackets where we put the index of the layer

A neural network that has the input, hidden and output layers is referred to as a 2 layer neural network, because the input layer is not counted

**Computing a Neural Network's Output**

A picture containing text, diagram, font, circle

Description automatically generated

This is the representation of one unit, that can be perceived as 2 parts

A picture containing diagram, line, circle, drawing

Description automatically generatedA picture containing text, font, white, typography

Description automatically generated

All the calculations for on layer can be computed just by multiplying the matrices

A picture containing text, handwriting, child art, drawing

Description automatically generated

A black text on a white background

Description automatically generated with medium confidence

A picture containing font, text, white, symbol

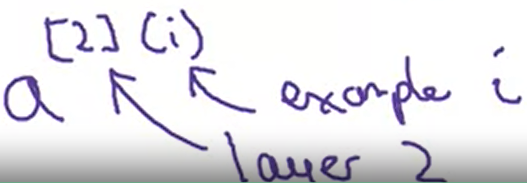
Description automatically generated

**Vectorizing Across Multiple Examples**

If we want to compute the forward propagation for all examples, we will have to do the following computation for each training examples:

A picture containing font, text, white, typography

Description automatically generated (if we are using a neural network with 2 layers + the input)

As a notation we use 

To create a forward pass on each example of the data set we would need a for loop, but to avoid it we can vectorize the entire data set:

A picture containing handwriting, calligraphy, font, typography

Description automatically generated(create a matrix X in which we have an x on each column)

A picture containing text, handwriting, font, line

Description automatically generated(same Z and A, where the columns stand for the examples, and the rows for hidden units)

We end up using the following computations for a vectorized forward pass across the entire data set

A picture containing text, font, screenshot, typography

Description automatically generated

When we vectorize across all examples, we have to keep in mind that during the forward pass the weights are the same for all training examples

**Activation Functions**

Sigmoid – almost never to use it, except for the output layer because we want to have the output between 0 and 1

Tanh – use it instead of sigmoid because basically it’s the same functions, but it’s centered in 0, thus helping the gradients to be centered in 0 as the input features; it almost every time works better than the sigmoid

ReLU – it’s the default activation function in deep learning; when the gradients are large / small, this function helps the network to work faster and to avoid the loss of large gradients, that are maximum 1 while using sigmoid and tanh

Leaky ReLU – almost every time works better than the ReLU, because we also propagate gradients that are smaller than 0; some people treat the parameter from this functions as one that has to be tweaked (hyperparameter)

A picture containing text, diagram, line, handwriting

Description automatically generated

**Why do you need Non-Linear Activation Functions?**

If we don’t use non-linear activation functions at least in hidden layers, the network, even if it’s extremely deep, cannot learn something complex because in the end the output is a linear combination of the input features

Only for regression problems we can use a linear activation function in the output layer, but we have to use for the rest of the layers non-linear activation functions