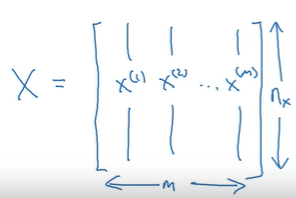
**Week 2 – Notes**

**Logistic Regression as a Neural Network**

**Binary Classification**

The notation used throughout the course:

 - features: one example per column (m columns) nx features (rows)

A picture containing text

Description automatically generated - targets: one target per column

**Logistic Regression**

**Text

Description automatically generated with medium confidenceWhiteboard

Description automatically generated with medium confidenceIcon

Description automatically generated**

**Logistic Regression Cost Function**

We denote

Text

Description automatically generated with low confidence

We cannot define the loss function as below because it isn’t convex

A picture containing icon

Description automatically generated

We define it as

Text

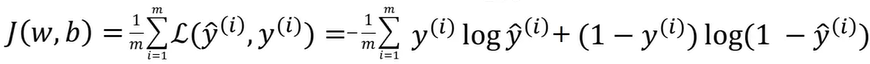
Description automatically generated

The cost function is the average of all loss functions for each pair of y and y hat

Text, whiteboard

Description automatically generated

**Gradient Descent**



We want to minimize J(w, b) – the cost function

The derivative represents the slope of the function in a point

With d we denote derivatives and with sigma we denote partial derivatives

This algorithm implies moving towards the minimum values of the function by repeatedly subtracting from each parameter the value given by alpha (learning rate) \* the derivative of the function with respect to that parameter

**Derivatives**

A picture containing text

Description automatically generated

Computes what’s the ratio between the change in the value of the function and the change of the variable, if we increase the variable just by a little (represents the height / width of the triangle that we draw it with the hypotenuse as being the tangent)

**More Derivative Examples**

In machine learning the log always denotes the ln

**Computation graph**

Forward pass -> compute the outputs of the network

Backward pass -> Compute the gradients / derivatives

**Computing derivatives**

**Diagram

Description automatically generated**

In this course, dvar is the notation of the derivative of the cost function with respect to the variables

To compute the derivative of J with respect to a, we use the chain rule:

 = Text

Description automatically generated

To compute derivates we can use the defined formula, or to use the definition, but in which we consider x0 to be a small value, such as 0.001

**Logistic Regression Gradient Descent**

Diagram

Description automatically generated

We have to update w1, w2 and b; for example w1 := w1 – alpha \* dw1

**Gradient Descent on m Examples**

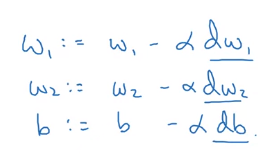
The superscript notation denotes the indices / number of the data example

Text

Description automatically generated

We initialize J, dw1, dw2 and db with 0 and we compute them by adding the value computed in relation with each training example; after that we divide each variable with the number of examples

One iteration of the gradient descent means that we update w1, w2 and b:



Because we want to avoid having many for loops, we’ll use the vectorized implementation

**Python and Vectorization**

**Vectorization**

Just by multiplying vectors / matrices without using for loops we can speed up the code by hundred of times

For example we can use np.dot(vec1, vec2) to multiply 2 vectors

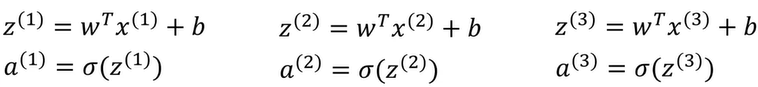
Vectorized implementation benefit because of the SIMD instructions that are implemented on the CPU and GPU

**More Vectorization Examples**

Even the computation of the e^ of each elements of a vector can be vectorized with np.exp(vector)

To vectorize the logistic regression, we would use dw as a vector instead of using dw1, dw2 and so on

**Vectorizing Logistic Regression**

****

To compute z1, z2 etc we can use a vectorized implementation

A picture containing text

Description automatically generated, when you add a number to a vector, that number is added to each value from the vector; this operation is called broadcasting

**Vectorizing Logistic Regression's Gradient Output**

The gradient descent algorithm can be vectorized, so that for one iteration we won’t use any for loop

Graphical user interface, text, application

Description automatically generated

**Broadcasting in Python**

If you have a matrix (3, 4) and divide it by an array (1, 4), that array will be duplicated 3 times and then the division is done element-wise

The general rule:

(m, n) + - \* / (1, n) => then the (1, n) is duplicated and transformed into a (m, n)

(m, n) + - \* / (m, 1) =? Then the (m, 1) is duplicated and transformed into a (m, n)

Then, all the operations are done element-wise

**A Note on Python/Numpy Vectors**

np.array([1, 2]) + np.array([[1], [2]]) -> array([[2, 3], [3, 4]]), which is very strange

The transpose of a rank 1 array is the same array; a rank 1 array has the shape (n,) not (n, 1)

np.random.randn(5) -> (5,) but np.random.randn(5 ,1) -> (5, 1)

You can test the shape in the code by using assert(a.shape == (5, 1))

Do not use rank 1 arrays

**Explanation of Logistic Regression Cost Function (Optional)**

A picture containing handwriting, font, calligraphy, typography

Description automatically generated

These 2 can be combined in a single formula:

A picture containing handwriting, font, text, calligraphy

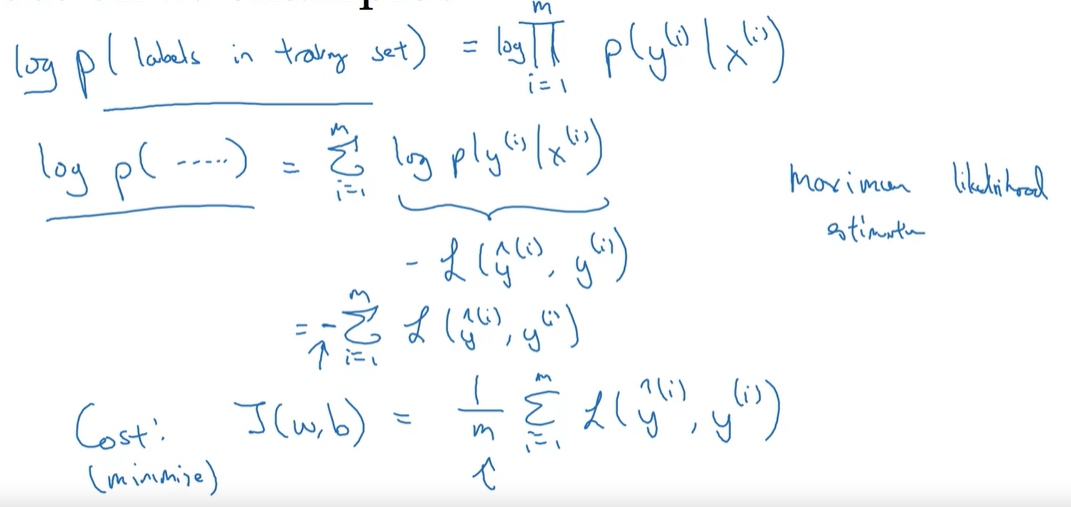
Description automatically generated

Maximizing p(y|x) is the same as maximizing log p(y|x), considering that log is a strictly monotonic functionA picture containing text, handwriting, font, calligraphy

Description automatically generated

If you consider that the training examples are independent and identically distributed (i.i.d.), then you can

Maximum likelihood estimation = find the parameters of a probability distribution function that maximize the likelihood of the observed data

****

The 1 / m is just a scaling factor, for convinience