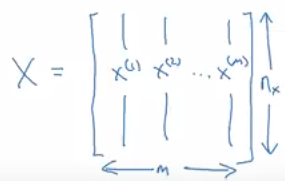
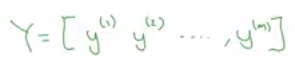
**Lec 5: Binary Classification in Deep Learning**

We are looking at logistic regression.

For a 64 x 64 image with three channels, it must be converted to a feature vector (np.flatten) of size 64 x 64 x 3.

*Notations used:*

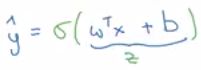
* (x, y) is the training example along with its label.
* Consider ‘m’ training examples: 
* For neural networks it is always advised to stack the ‘m’ input vectors and its labels separately:  

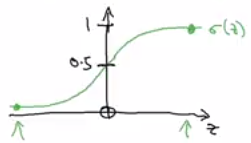
**Lec 6: Logistic Regression**

Given a training example X, we want to predict y’ (y-hat):  where x is .

Its parameters weights and bias are: 

The output cannot be a linear equation because what we want is a probability output, hence we fit a sigmoid function.



Sigmoid function:  

Note:

* If value is large then output is close to 1
* If value is small then output is close to 0

**Lec 7: Logistic Regression Cost Function**

Cost function is used to train the parameters ‘W’ and ‘b’.

The loss function determines how good the predicted output (y’) is against actual output (y). Squared error is an absolute choice but in logistic regression the gradient descent would have many minima (not convex), hence it is not used. 

The loss function used is: 



Loss function is for a single training example.

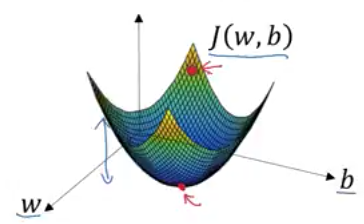
The cost function is used to check how the parameter W and b are doing for the entire training set: 

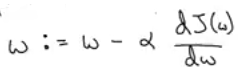


**Lec 8: Gradient Descent**

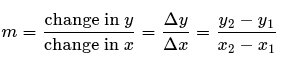
Gradient descent algorithm to learn parameters ‘W’ and ‘b’ on the training set.

Gradient descent finds best ‘W’ and ‘b’ that minimizes the cost function J(W, b).

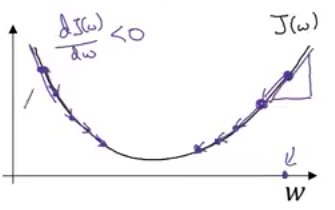
The cost function is a bowl curve (convex function) with a single minimum: 

To get minimum cost having a single parameter we have to repeat: 

The slope is the change made to alter ‘W’.

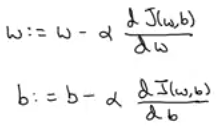
Slope of curve at a point: 

Two cases of positions of ‘W’:



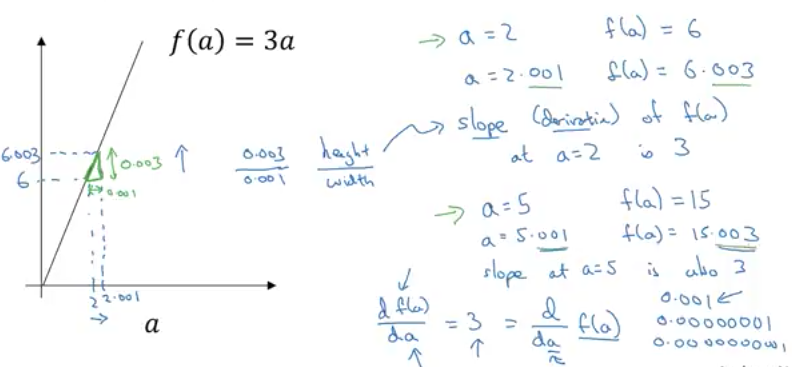
* If ‘W’ is high, then the slope at that point will be positive, hence the change must be subtracted.
* If ‘W’ is low, then the slop at that point will be negative, hence the change must be added.

Now if there are two parameters repeat the following:

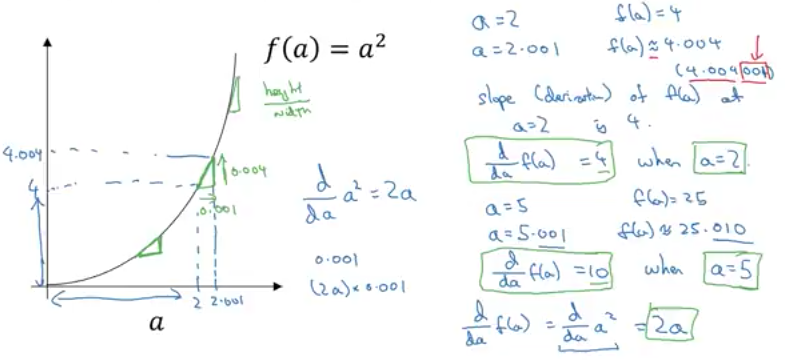


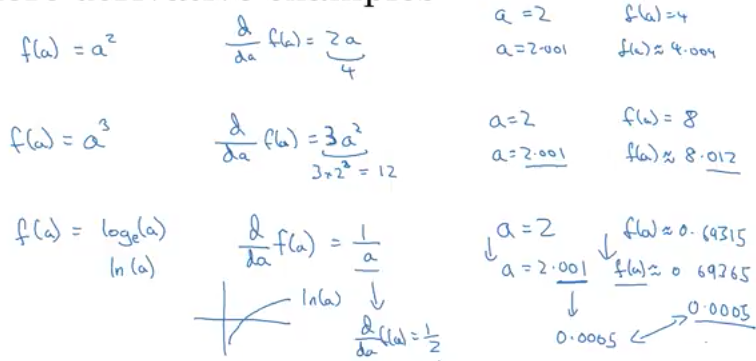
**Lec 9: Derivatives**

Intuition of derivatives (slope) of a line which is same throughout. So if a little nudge in variable ‘a’ causes function ‘f(a)’ to move by three times, then the slope of that function is 3.



**Lec 10: Derivatives Examples**



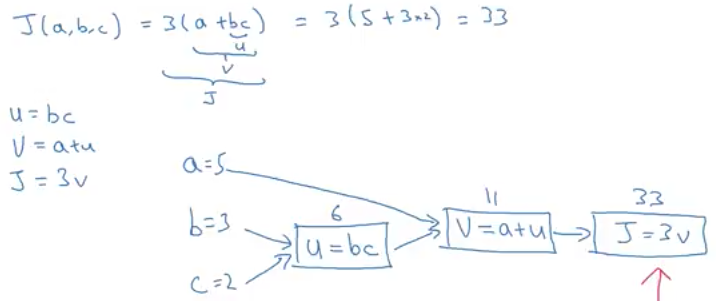


**Lec 11: Computation Graph**

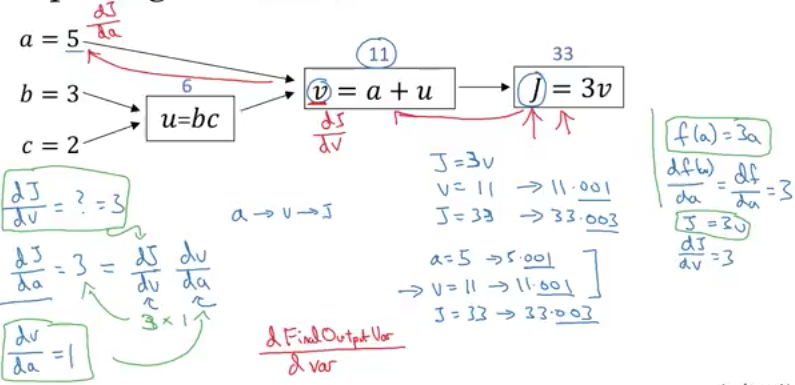
Computation helps understand:

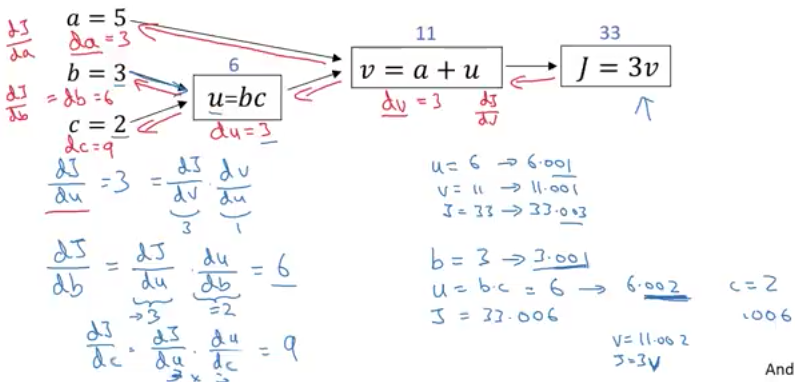
* Forward pass to obtain output
* Backward pass to obtain gradients/derivatives to optimize cost function

Example of forward pass:



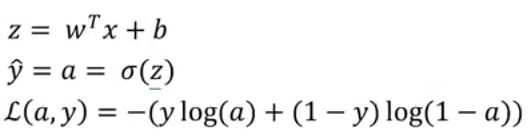
**Lec 12: Derivatives with a Computation Graph**



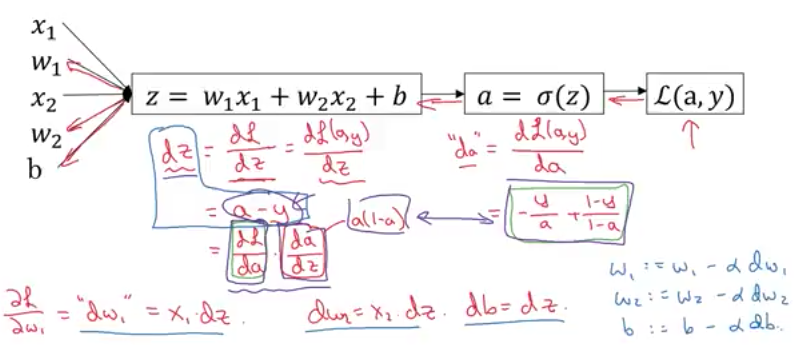


**Lec 13: Logistic Regression Derivatives**

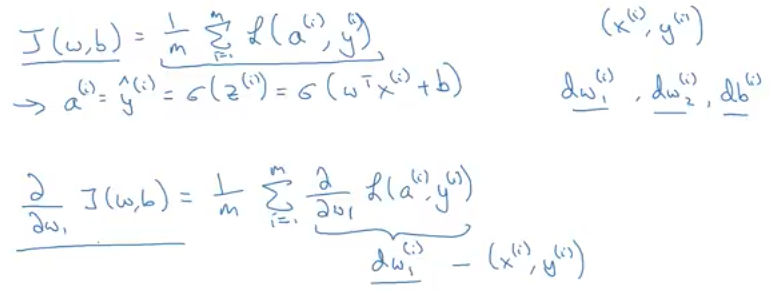
Logistic regression formula:

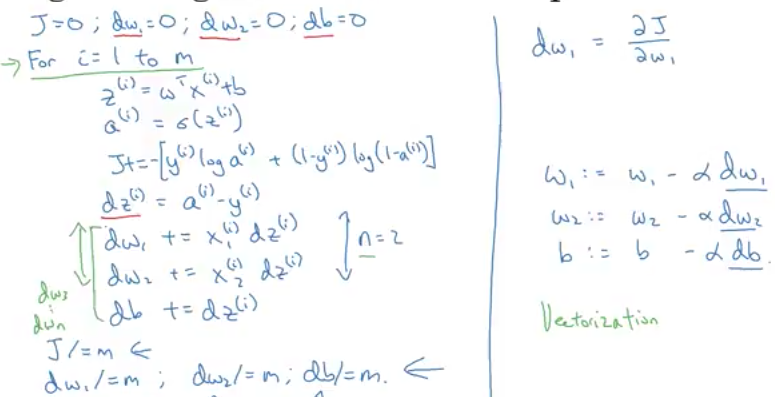


Consider two features in the following ‘x1’ and ‘x2’; whose weights are ‘w1’ and ‘w2’ respectively and the bias is ‘b’:



**Lec 14: Gradient Descent on ‘m’ Training Examples**



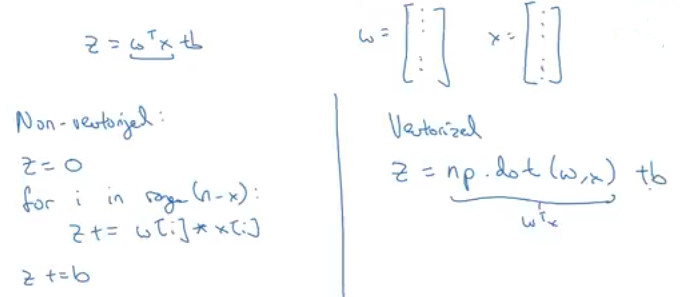


In the case above there are two for loops:

* One to run through the ‘m’ training examples
* And another one to run through each feature of each example

This is problematic for huge datasets as it takes a long time to run. Hence in deep learning era it is wise to use **vectorization**.

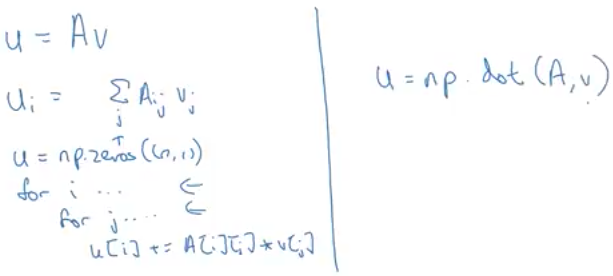
**Lec 15: Vectorization**



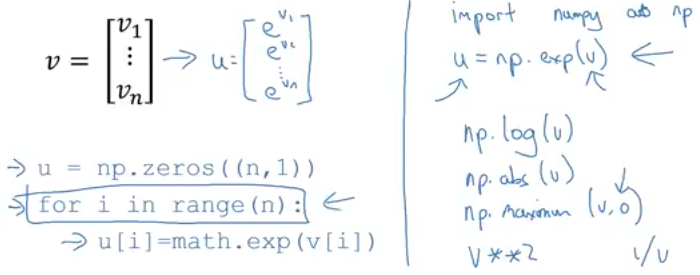
Vectorized code is ~300 times faster rather than explicit ‘for’ loops

**Lec 16: More Vectorization examples**

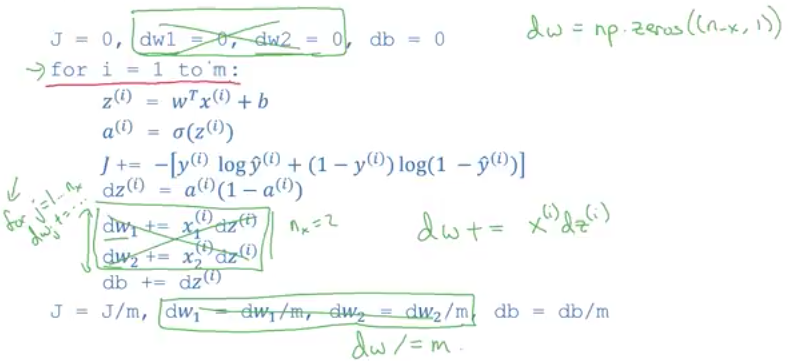
In the following example we avoid two for loops using Vectorization:



Now consider you need to apply exponential, log or square operation to every element in matrix/vector:

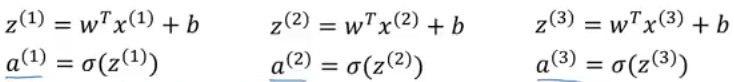


This can be done on logistic regression to reduce one ‘for’ loop:



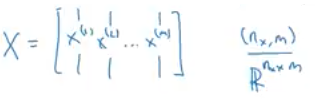
**Lec 17: Vectorizing Logistic Regression**

In the previous case we vectorized the inner ‘for’ loop that computes the gradients of all the weights for each training example. Here we will perform Vectorization for forward propagation. In forward propagation we calculate the prediction for each training example (a)

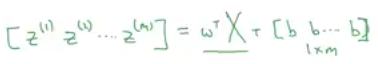
If we had ‘m’ training examples we would compute each of them by the following: 

Here we will vectorize the operation for each iteration in training example also.

So now we have the input of ‘m’ examples each having ‘n’ features:



In order to vectorize and avoiding ‘for’ loops, we have to create a row vector of (1 x m) such that:



Now substituting the transpose of weight matrix which is a row vector also:



We get:



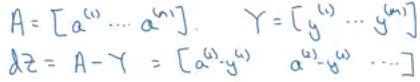
In python: 

**Lec 18: Vectorizing Logistic Regression’s Gradient Computation**

Gradients for each training example is computed:  and so on.

Here we have to create (1 x m) vector: 

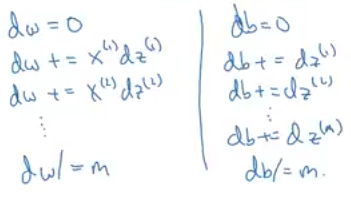
Now ‘A’ and ‘Y’ are already known vectors:

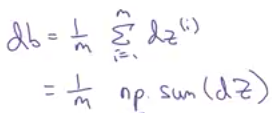


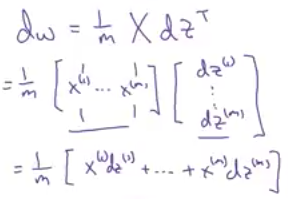
So in one line of code we can get all the gradients.

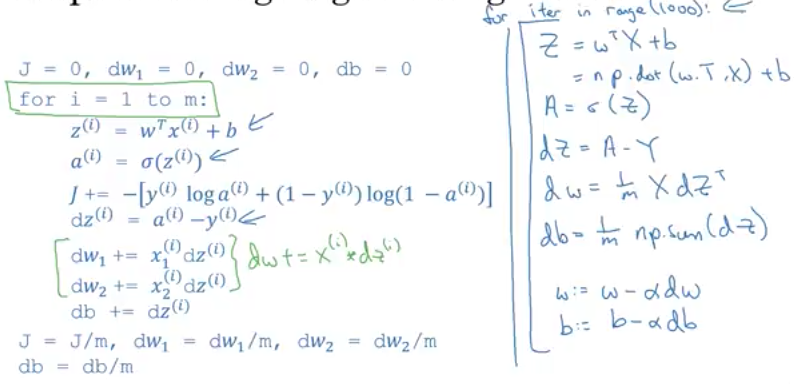
Previously, all the weight and bias gradients were computed in a ‘for’ loop, here we want to vectorize it.

The following are the gradients of weights and bias:



The bias can be written as: 

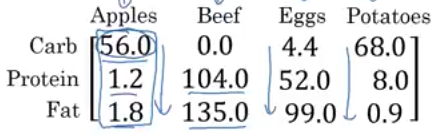
And the weights as: 

So we can modify logistic regression as follows: 

The outer ‘for’ loop is for iterations which cannot be skipped.

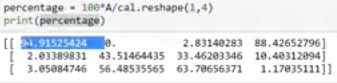
**Lec 19: Broadcasting in Python**

From the following example the aim is to calculate the percentage consumption of calories of each nutrient for each of the food items:



*Can this be done without using explicit ‘for’ loop?*

Consider the matrix above to be ‘A’, so the sum for each column is: 

and the percentage is: 

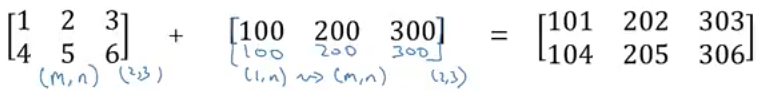
The second command:  is depicting broadcasting.

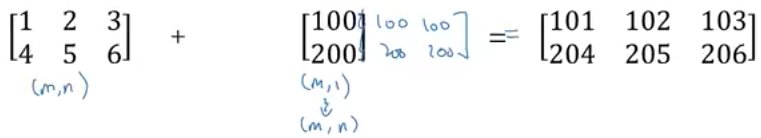
Here we are dividing the ‘A’ matrix of shape (3x4) by ‘cal’ matix of shape (1x4). ***But how?***

Let us see another example:



The constant 100 is scaled up to a vector of same shape allowing it to perform addition.

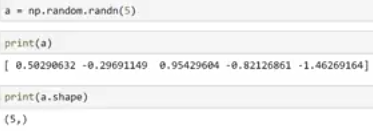




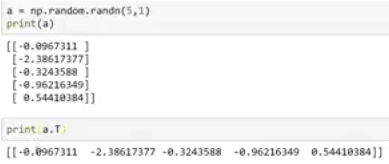
**Lec 20: Python-Numpy**

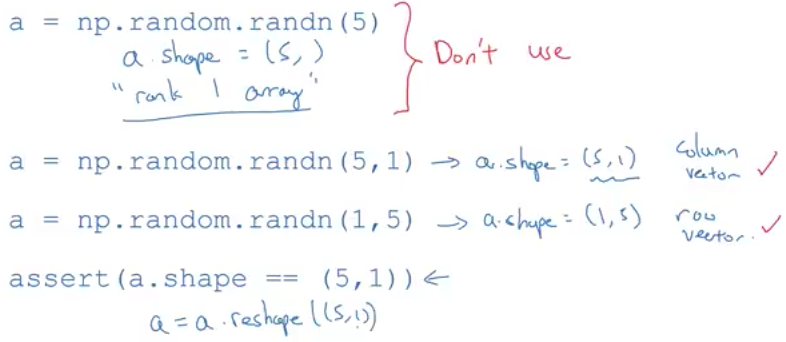
Difference between rank 1 matrix and a column/row vector.

Following is the assignment of a rank 1 array:



Following is the initialization of a row vector:





**Lec 21: Jupyter - iPython**

Overview of the tool