Neural Network Basics

1. how to convert the image to vector:

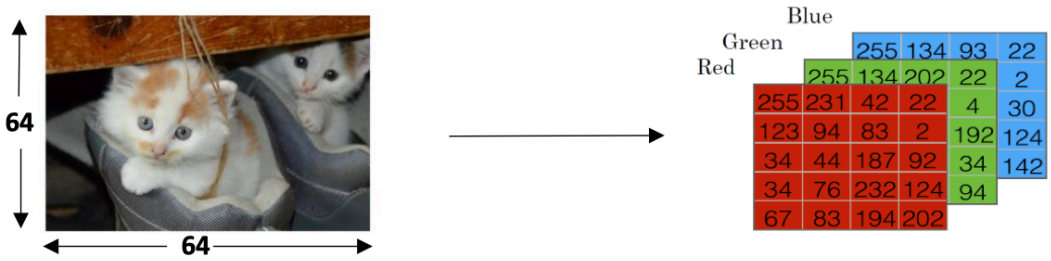
Binary Classification

In a binary classification problem, the result is a discrete value output.

For example - account hacked (1) or not hacked (0)
- a tumor malign (1)or benign(0)

Example: Cat vs Non-Cat

The goal is to train a classifier for which the input is an image represented by a feature vector, x, and predicts whether thecorresponding label$y$is 1 or 0. In this case, whether this is a cat image(1)or a non-cat image (0).



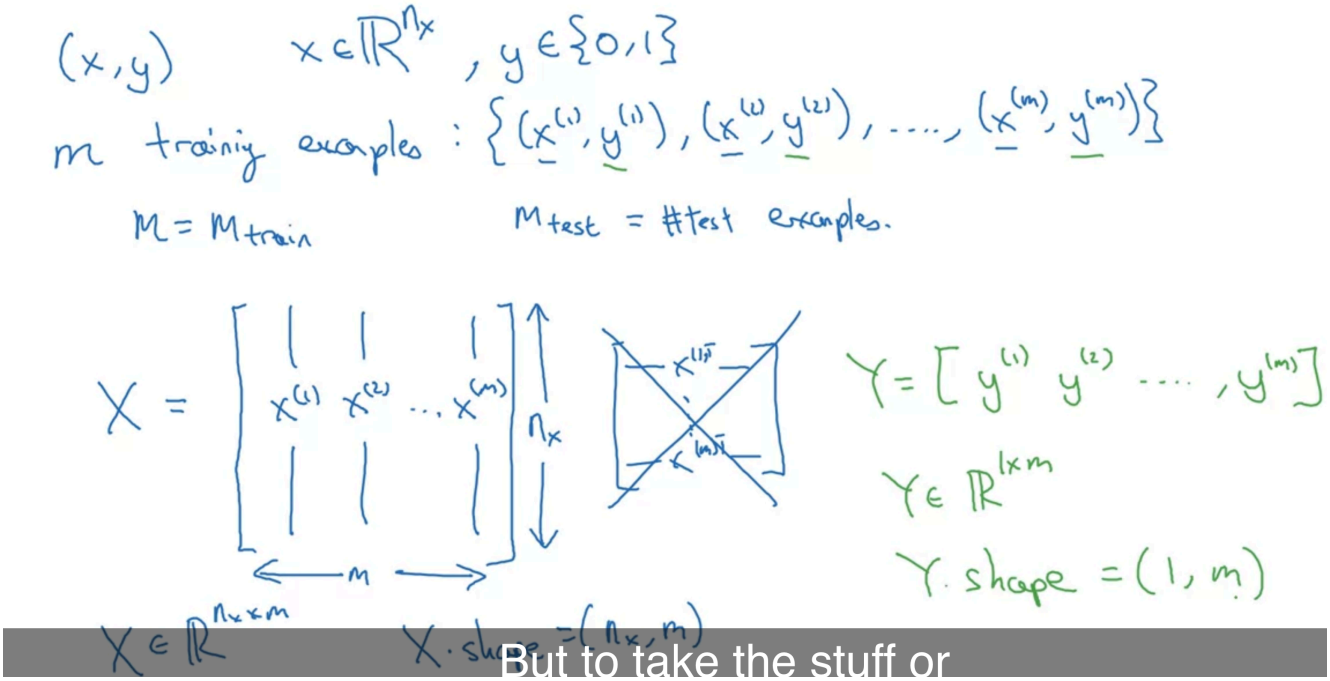
An image is stored in the computer in three separate matrices corresponding to the Red, Green, and Blue color channels of the image. The three matrices have the same size as the image, for example, the resolutionof the cat image is 64 pixels X 64 pixels, the three matrices (RGB) are 64 X 64 each.

The value in a cell represents the pixel intensity which will be used to create a feature vector of n-dimension. In pattern recognition and machine learning, a feature vector represents an image, Then the classifier's job is to determine whether it contain a picture of a cat or not.

To create a feature vector, x, the pixel intensity values will be "unrolled" or "reshaped" for each color. The dimension of the input feature vector$x$ is$n = 64x\,64x\,3 = 12288$.

$$x = \begin{bmatrix} 255 \\ 231 \\ 42 \\ \vdots \\ 255 \\ 134 \\ 202 \\ \vdots \\ 255 \\ 134 \\ 93 \\ \vdots \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \text{red} \\ \\ \\ \\ \text{green} \\ \\ \\ \\ \text{blue} \\ \\ \end{matrix}$$

Notation



But to take the stuff or

当有m个training instance时，将m个instance横向堆叠所以feature vector X的dimension就是nx*m, Y也横向堆叠，Y.shape=(1,m)

2. logistic regression:

Logistic Regression

Logistic regression is a learning algorithm used in a supervised learning problem when the output y are all either zero or one. The goal of logistic regression is to minimize the error between its predictions and training data.

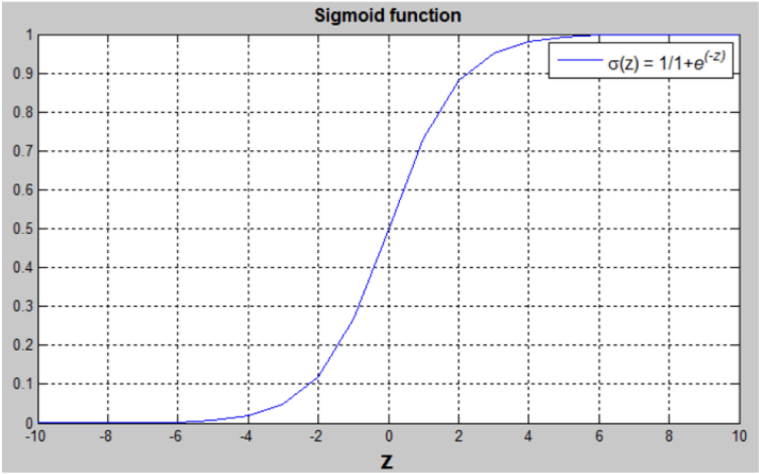
Example: Cat vs No - cat

Given an image represented by a feature vector x, the algorithm will evaluate the probability of a cat being in that image.

$$Given\ x\ ,\ \hat{y} = P(y = 1|x), \text{where } 0 \leq \hat{y} \leq 1$$

The parameters used in Logistic regression are:

- The input features vector: $x \in \mathbb{R}^{n_x}$, where n_x is the number of features
- The training label: $y \in 0,1$
- The weights: $w \in \mathbb{R}^{n_x}$, where n_x is the number of features
- The threshold: $b \in \mathbb{R}$
- The output: $\hat{y} = \sigma(w^T x + b)$
- Sigmoid function: s = $\sigma(w^T x + b)$ = $\sigma(z)$= $\frac{1}{1+ e^{-z}}$



$(w^T x + b)$ is a linear function $(ax + b)$, but since we are looking for a probability constraint between [0,1], the sigmoid function is used. The function is bounded between [0,1] as shown in the graph above.

Some observations from the graph:

- If z is a large positive number, then $\sigma(z) = 1$
- If z is small or large negative number, then $\sigma(z) = 0$
- If $z = 0$, then $\sigma(z) = 0.5$

3. The loss function computes the error for a single training example; the cost function is the average of the loss functions of the entire training set.

Logistic Regression: Cost Function

To train the parameters w and b, we need to define a cost function.

Recap:

$\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b)$, where $\sigma(z^{(i)}) = \frac{1}{1+e^{-z^{(i)}}}$

$x^{(i)}$ the i-th training example

$Given \{(x^{(1)}, y^{(1)}), \cdots, (x^{(m)}, y^{(m)})\},\ we\ want\ \hat{y}^{(i)} \approx y^{(i)}$

Loss (error) function:

The loss function measures the discrepancy between the prediction ($\hat{y}^{(i)}$) and the desired output ($y^{(i)}$). In other words, the loss function computes the error for a single training example.

$$L(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{2}(\hat{y}^{(i)} - y^{(i)})^2$$

$$L(\hat{y}^{(i)}, y^{(i)}) = -(y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

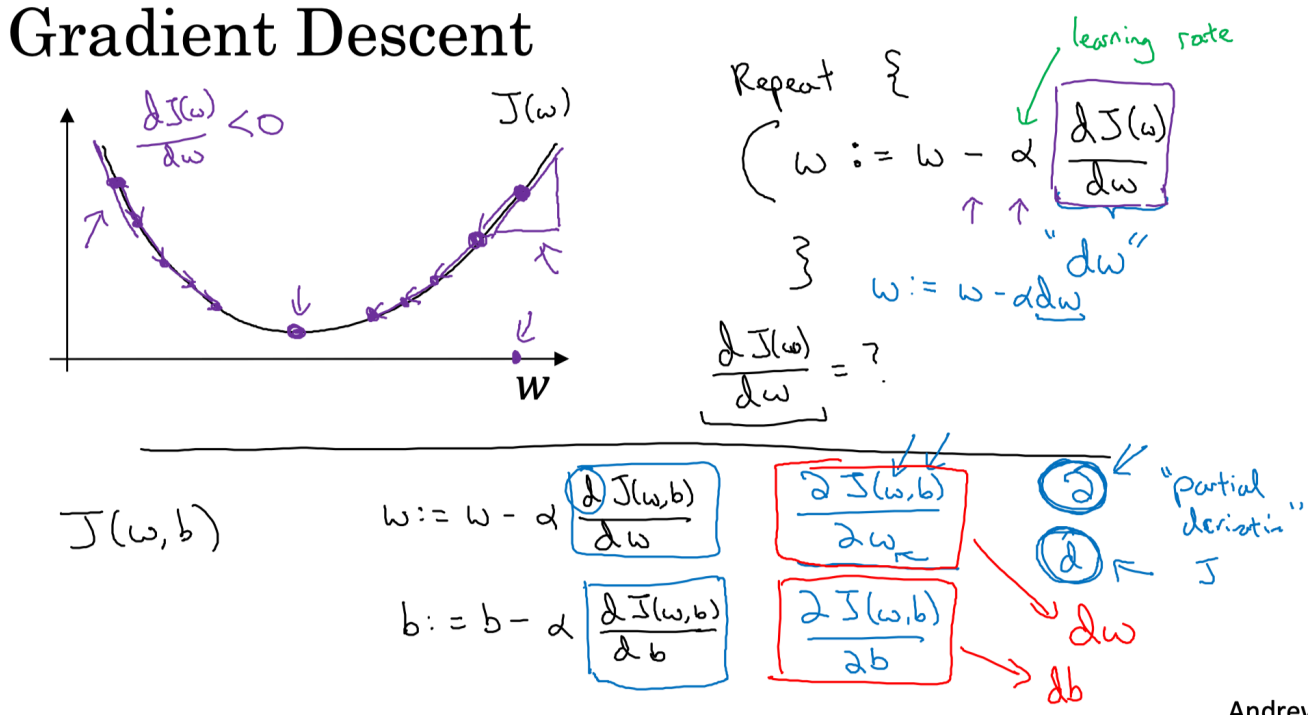
- If $y^{(i)} = 1$: $L(\hat{y}^{(i)}, y^{(i)}) = -\log(\hat{y}^{(i)})$ where $\log(\hat{y}^{(i)})$ and $\hat{y}^{(i)}$ should be close to 1
- If $y^{(i)} = 0$: $L(\hat{y}^{(i)}, y^{(i)}) = -\log(1 - \hat{y}^{(i)})$ where $\log(1 - \hat{y}^{(i)})$ and $\hat{y}^{(i)}$ should be close to 0

Cost function

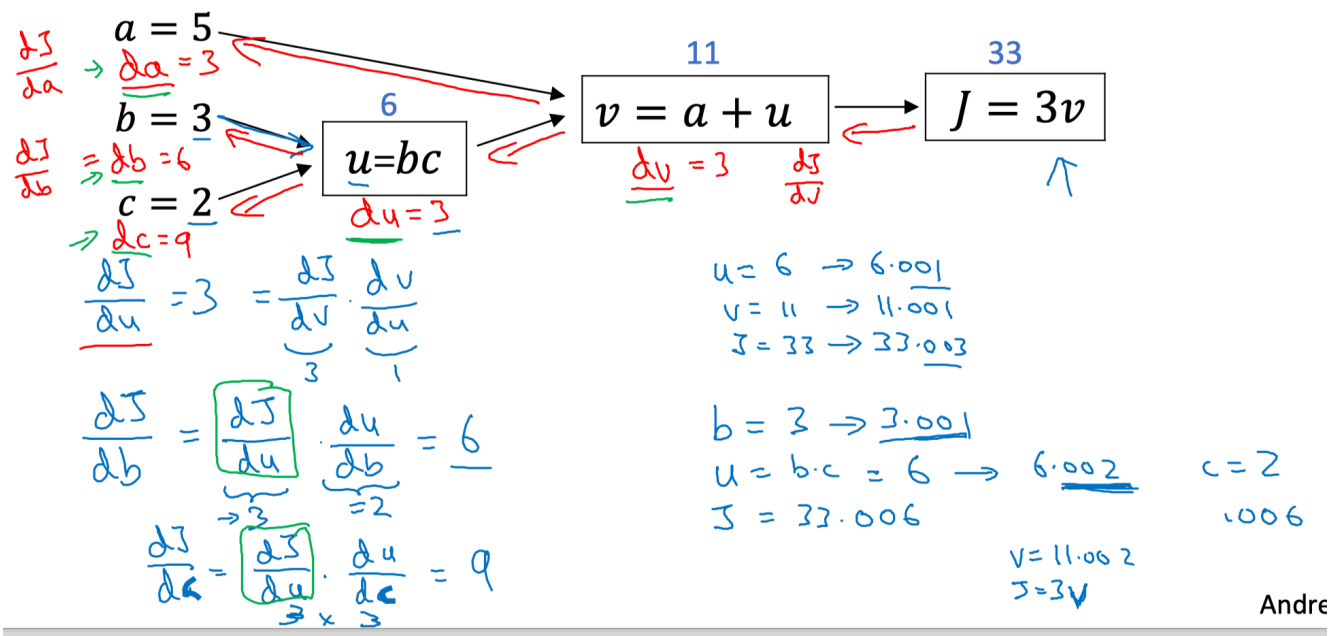
The cost function is the average of the loss function of the entire training set. We are going to find the parameters $w\ and\ b$ that minimize the overall cost function.

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m}[(y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})]$$

4. Gradient Descent：w和b更新方式，这里alpha是learning rate也就是步长



5. computation graph: 从左到右是forward propagation, 从右到左是backwards propagation,

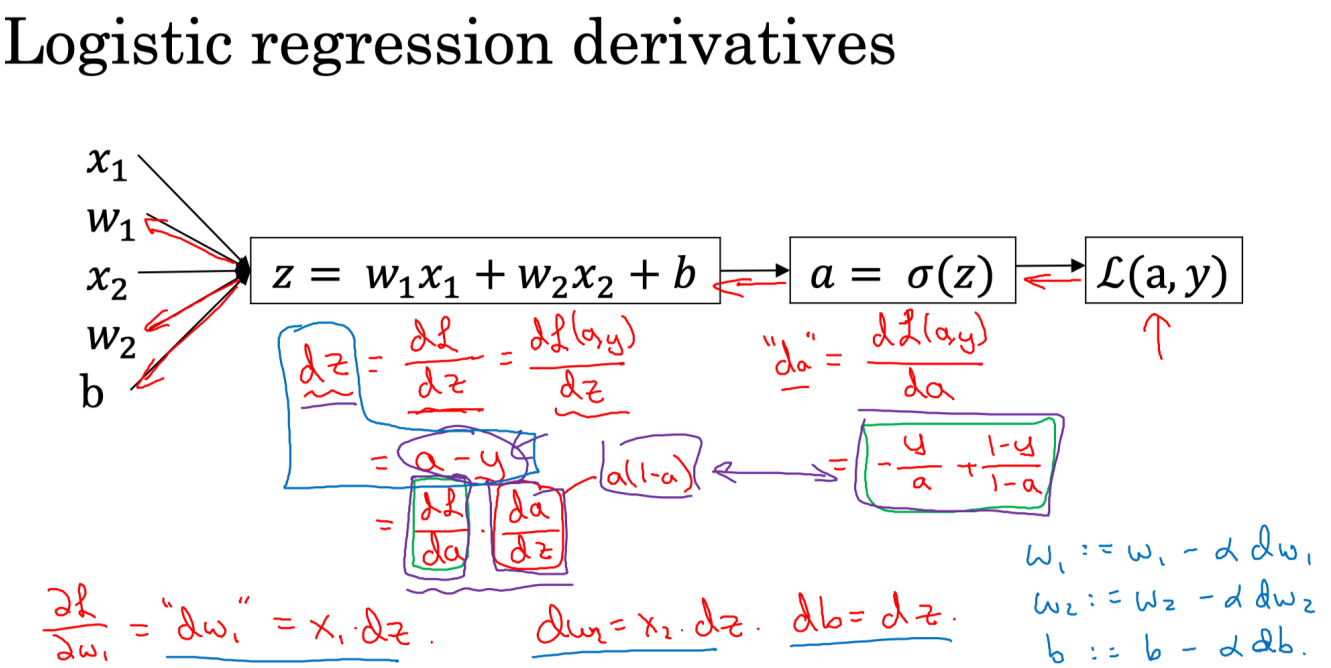


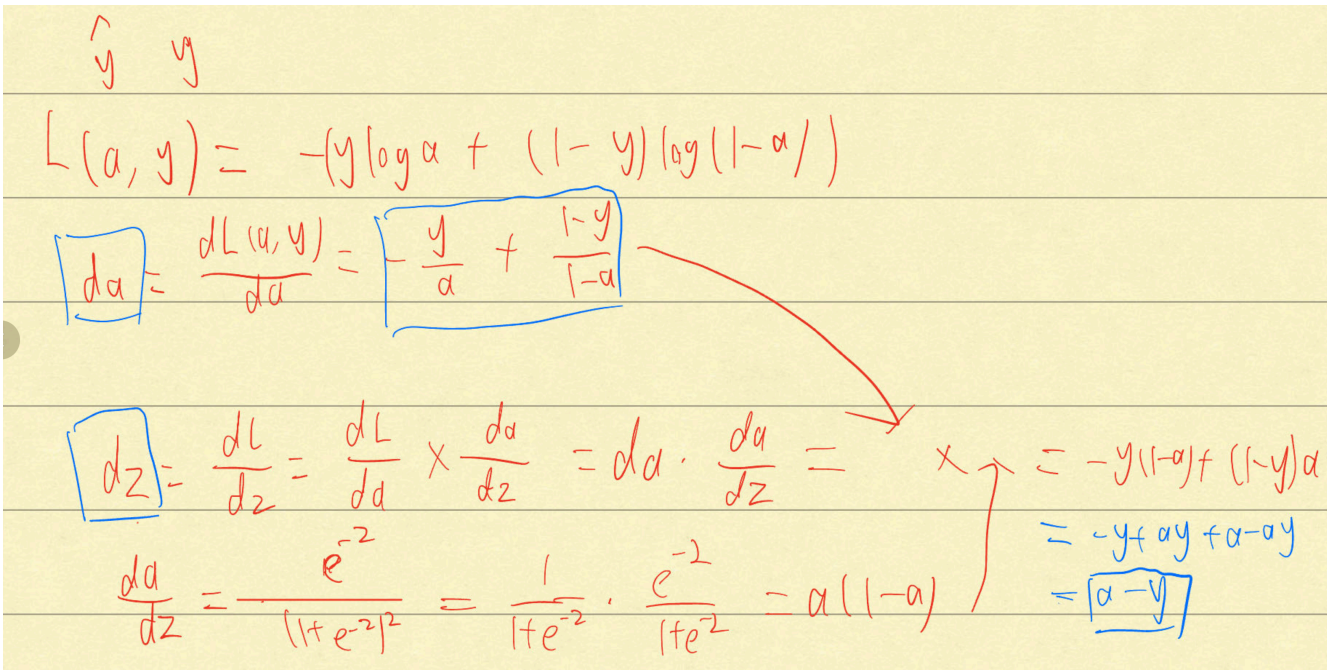
In the context of neural networks and machine learning, forward propagation and backward propagation are two fundamental processes that occur during the training of a neural network. They are essential for understanding how neural networks learn from data.

1. **Forward Propagation**:
   - **Definition**: Forward propagation is the process of moving the input data through the neural network to generate an output. This is essentially how the neural network makes predictions.
   - **Process**: During forward propagation, the input data is fed into the network. It then passes through the layers of the network, where each layer applies specific weights and biases, often followed by a non-linear activation function. The result is the predicted output at the end.
   - **Purpose**: The main purpose of forward propagation is to compute the output based on the current state of the network's weights and biases.
2. **Backward Propagation**:
   - **Definition**: Backward propagation, often called backpropagation, is the process of tuning the weights and biases of the neural network based on the error of the prediction. It is a key part of the training process.
   - **Process**: In backward propagation, the error (or loss) is calculated by comparing the predicted output of the network with the actual target values. This error is then propagated back through the network, starting from the output layer to the input layer. During this process, the weights and biases are adjusted in order to minimize the error.
   - **Purpose**: The main purpose of backward propagation is to optimize the weights and biases so that the network's predictions become more accurate over time.

In a typical training cycle of a neural network, forward propagation is used to calculate the output for a batch of data, and then backward propagation is used to update the network's weights and biases based on the error of that output. This cycle is repeated many times with different batches of data until the network becomes proficient at making accurate predictions.

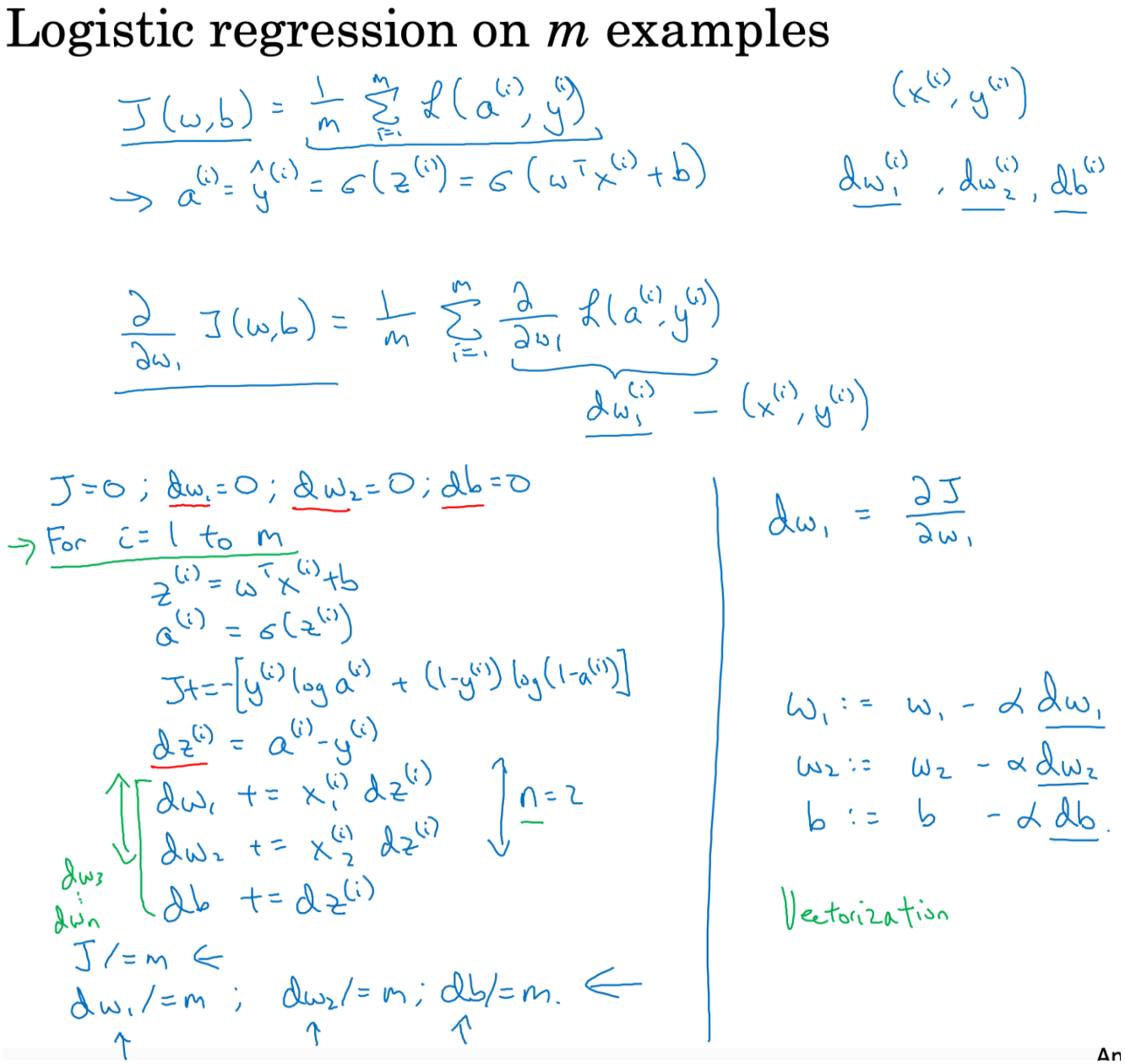
6. logistic regression derivatives for one example:





反向传播链式求导得到L对于w1,w2,b的偏导,然后根据右下角更新这3个值

7. logistic regression derivatives for m example: 对6中取到的单个的derivative求和取平均



这里需要2个loop，一个loop遍历m个instance，另一个loop遍历n个参数，但是嵌套的loop太耗时了，所以用vectorization

Python and Vectorization

1. python and vectorization:

```
import time
import numpy as np

a=np.random.rand(1000000)
b=np.random.rand(1000000)
tic=time.time()
c=np.dot(a,b)
toc=time.time()

print(c)
print("Vectorized version:"+str(1000*(toc-tic))+"ms")
```

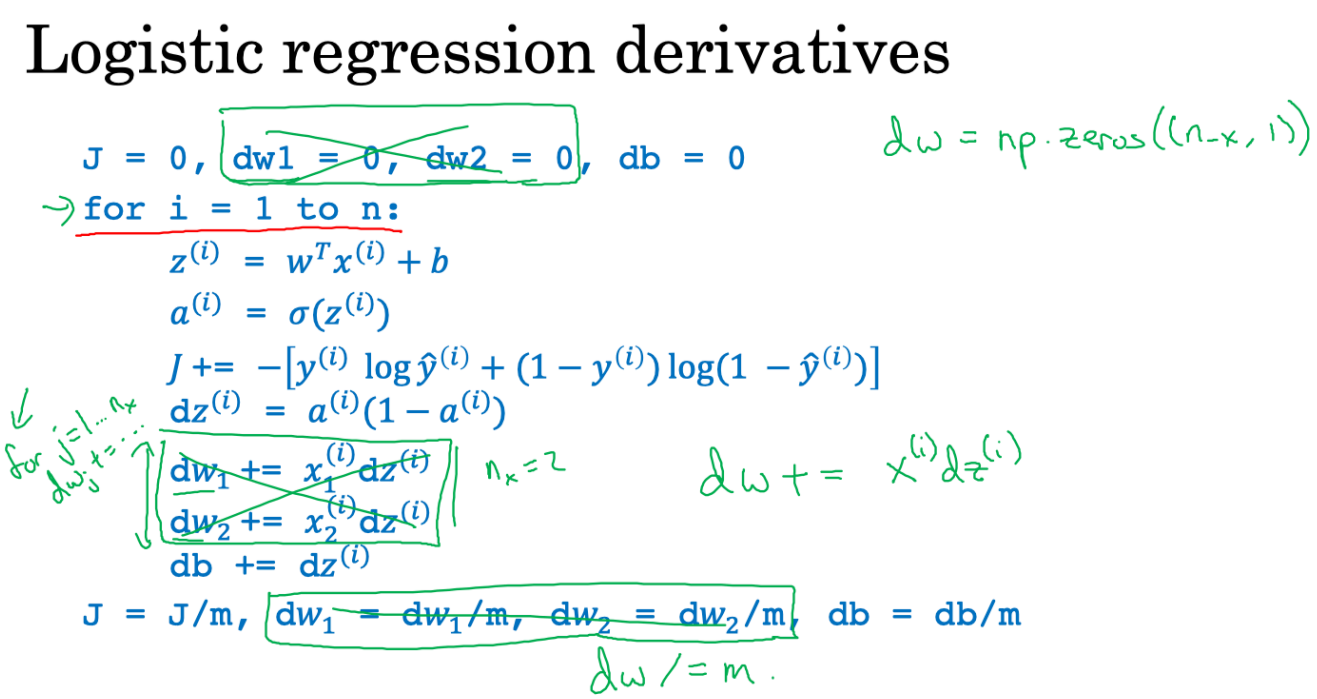
```
c=0
tic=time.time()
for i in range(1000000):
    c+=a[i]*b[i]
toc=time.time()
print(c)
print("For loop:"+str(1000*(toc-tic))+"ms")
```

result:

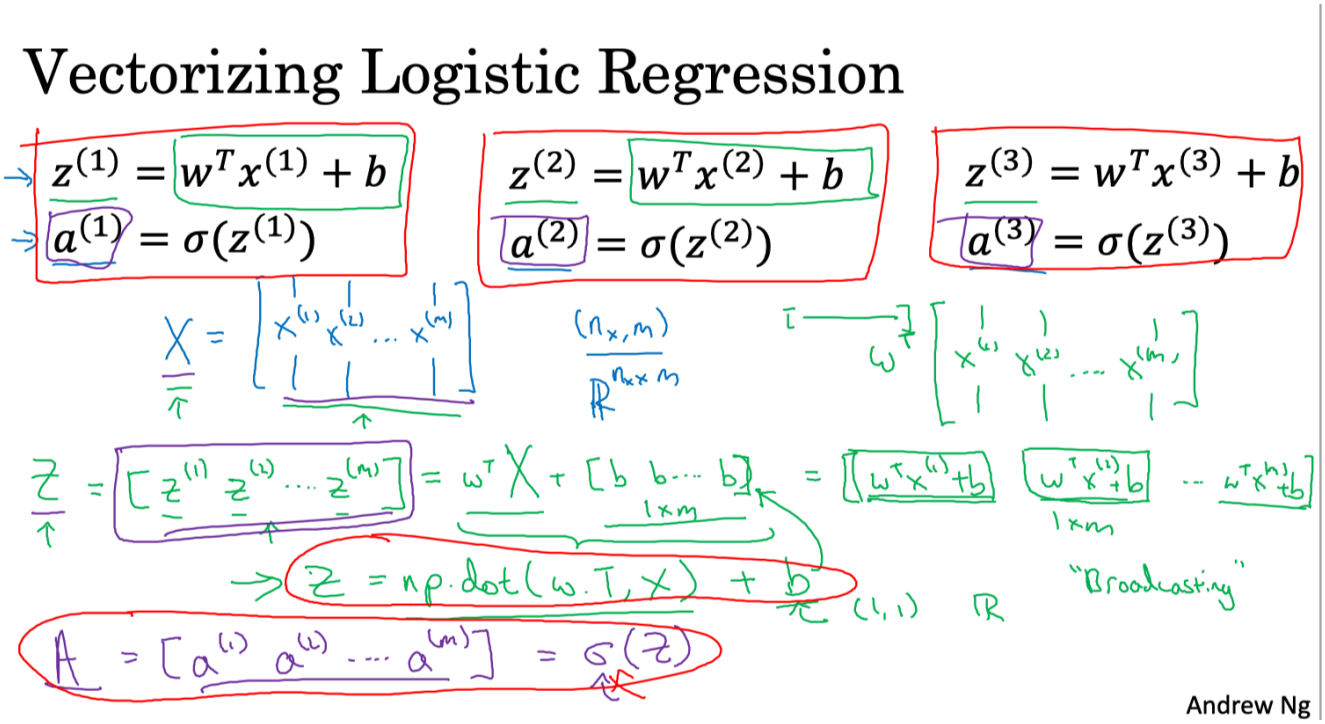
```
/usr/bin/python3 /Users/nanzhenghan/PycharmProjects/Bilibili/main.py
250166.45798134204
Vectorized version:0.5202293395996094ms
250166.45798133797
For loop:187.50977516174316ms

Process finished with exit code 0
```

2. set dw as a vector:



3. vectorizing logistic regression:



可以用z=np.dot(w.T,x)+b来直接算z来避免for loop

4. Implementing logistic regression with complete vectorization(code on the right), but we can't get rid of the iteration number of gradient descent: