Logistic Regression with a Neural Network mindset

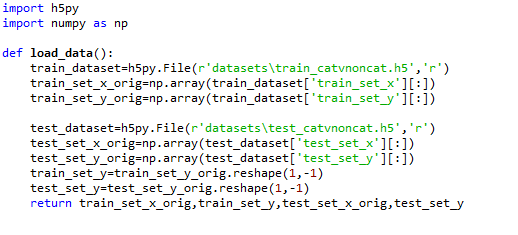
# Problems

We hope to build a recognize algorithm to classifier a cat, but using the former algorithm recognition accuracy is less than 70%, so we expect using new technology to increase the accuracy to more than 80%.

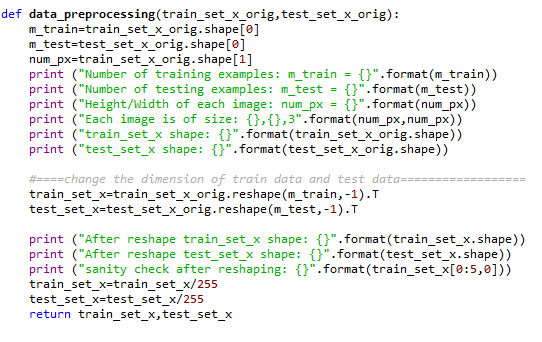
# Methods

## Overview of the Problem set

The given data has the format H5, so first we need use h5py library to change the data that python can understand.

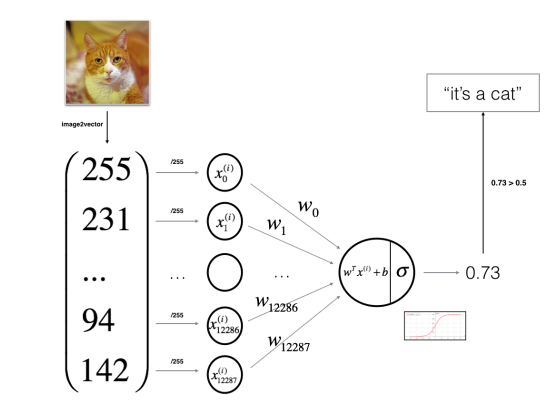


When get the train data set and test data set, we need to preprocessing data which is convenient for us dealing with later. The operation includes: reshape the data, just containing features and training number or test number. As the output y only has one after training, so we need to change the reshape to (1, m). Then we do normalization, which makes input data with normal distribution. In this practice, we just make the data to divide 255, it proves useful in image process in many practices.



## General Architecture of the learning algorithm

### Principle of Logistic Regression



### Mathematical

For a given data, we use logistic regression to deal with the problems.









## Building the parts of algorithm

### Initialize parameters

In this task, we initialize both W and b to zeros, just to simple the problem and focus on forward propagation and backward propagation in neural network.

### Activation function

Sigmoid: non-linear activation function



Relu: linear activation function



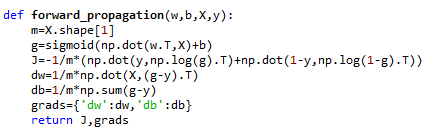
### Forward and Backward propagation

Forward propagation: computer A and cost

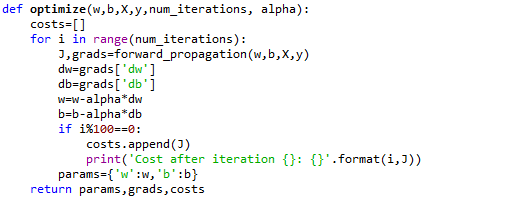
Backward propagation: computer dW and db:





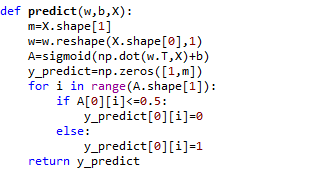


### Update parameters

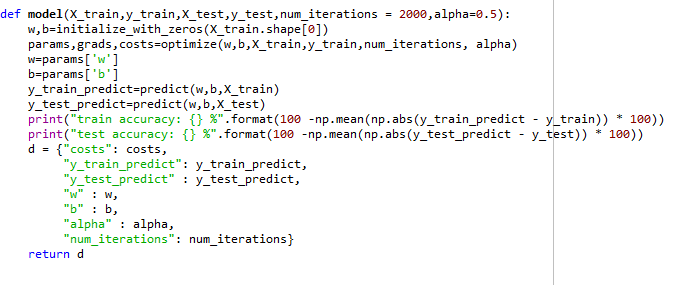


### Predict result

Use the new w and b to predict the result.

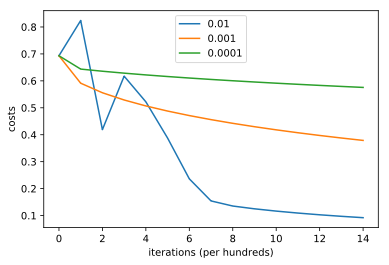


### Build model for practical use



# Result and analysis

### Different learning rate



### Accuracy:

|  |  |  |
| --- | --- | --- |
| Learning rate | Train accuracy | Test accuracy |
| 0.0001 | 68.4% | 36% |
| 0.001 | 88.9 | 64% |
| 0.01 | 99.5 | 68% |

# Conclusion

1. Different learning rates gives different cost and prediction results;
2. If the learning rate is too large, the cost may oscillate up and down, it may diverge (although in this practice large rate 0.01 seems have good result);
3. A low cost doesn’t mean a better model. We should check if there is possible overfitting. It always happens when training accuracy is a lot higher than the test accuracy.