Project Report

Application of Deep Neural Network in Cat Detection task

*(**DO XUAN LONG,** in corporation with **Coursera Team of Course Capstone Project**)*

Content

I. Problem Statement, Background and Data Collection

1. Problem Statement

- **Type:** Classification problem

- **Statement*:*** Build an L-layer Neural Network to detect a picture is a cat picture or not.
  + **Input:** A image with a size (64 x 64)

  + **Output:** Return: This is a cat picture (return 1) or not (return 0)

2. Background

- **Knowledges** about the general **Neural Network** and **Gradient Descent Algorithm** and **how it works** including: Initialization Steps, Forward Propagation Steps, Back-propagation Steps, Update Steps.

- **Mathematical Tools** including mainly **Linear Algebra, Calculus** for developing models.

3. Data Collection

- **Pre-collected sets of data** provided by **Coursera Team.**

II. Neural Network Model

1. Notation:

- Superscript ***[l]*** denotes a quantity associated with the l^{th} layer.
  - Example: $a^{[l]}$ is the l^{th} layer activation. W^l and b^l are the l^{th} layer parameters.

- Superscript **(i)** denotes a quantity associated with the i^{th} example.
  - Example: $x^{(i)}$ is the i^{th} training example.
- Lower script i denotes the i^{th} entry of a vector.
  - Example: $a_i^{[l]}$ denotes the i^{th} entry of the l^{th} layer's activations).

2. L-Layer Neural Network

	Shape of W	Shape of b	Activation	Shape of Activation
Layer 1	$(n^{[1]}, 12288)$	$(n^{[1]}, 1)$	$Z^{[1]} = W^{[1]}X + b^{[1]}$	$(n^{[1]}, 209)$
Layer 2	$(n^{[2]}, n^{[1]})$	$(n^{[2]}, 1)$	$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$	$(n^{[2]}, 209)$
⋮	⋮	⋮	⋮	⋮
Layer L-1	$(n^{[L-1]}, n^{[L-2]})$	$(n^{[L-1]}, 1)$	$Z^{[L-1]} = W^{[L-1]}A^{[L-2]} + b^{[L-1]}$	$(n^{[L-1]}, 209)$
Layer L	$(n^{[L]}, n^{[L-1]})$	$(n^{[L]}, 1)$	$Z^{[L]} = W^{[L]}A^{[L-1]} + b^{[L]}$	$(n^{[L]}, 209)$

Details in this model:

- **12288** = (size picture input) x (size picture input) x 3
- **X** Input matrix of information
- **W** Weight function of layer
- **A** Activation function of the previous layer
- **b** Bias

In my model, I build with ***layers_dims = [12288, 20, 7, 5, 1]*** means that it has **4 layers:**

- **Input layer**: 12288 nodes
- **Hidden layers**: first: 20 nodes, second: 7 nodes, third: 5 nodes
- **Output layer**: 1 node

Train in datasets provided by **Coursera Team** in **2500 iterations**

3. Methods

- **Initialization**

  + **Weight Initialization Methods:** Xavier Initialization Method (http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf)

  + **Bias Initialization:** Zero Initialization Methods

- **Forward propagation**

  + **Linear -→ RecLu** for L-1 initial layers

  + **Sigmoid** for the classified layer

- **Back Propagation**

For layer l, the linear part is: $Z^{[l]} = W^{[l]}A^{[l-1]} + b^{[l]}$ (followed by an activation).

Suppose you have already calculated the derivative $dZ^{[l]} = \frac{\partial \mathcal{L}}{\partial Z^{[l]}}$. You want to get $(dW^{[l]}, db^{[l]}, dA^{[l-1]})$.

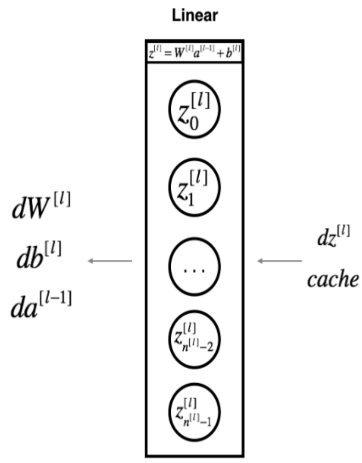


Figure 4

The three outputs $(dW^{[l]}, db^{[l]}, dA^{[l-1]})$ are computed using the input $dZ^{[l]}$ .Here are the formulas you need:

$$dW^{[l]} = \frac{\partial \mathcal{J}}{\partial W^{[l]}} = \frac{1}{m} dZ^{[l]} A^{[l-1]T}$$

$$db^{[l]} = \frac{\partial \mathcal{J}}{\partial b^{[l]}} = \frac{1}{m} \sum_{i=1}^{m} dZ^{[l](i)}$$

$$dA^{[l-1]} = \frac{\partial \mathcal{L}}{\partial A^{[l-1]}} = W^{[l]T} dZ^{[l]}$$

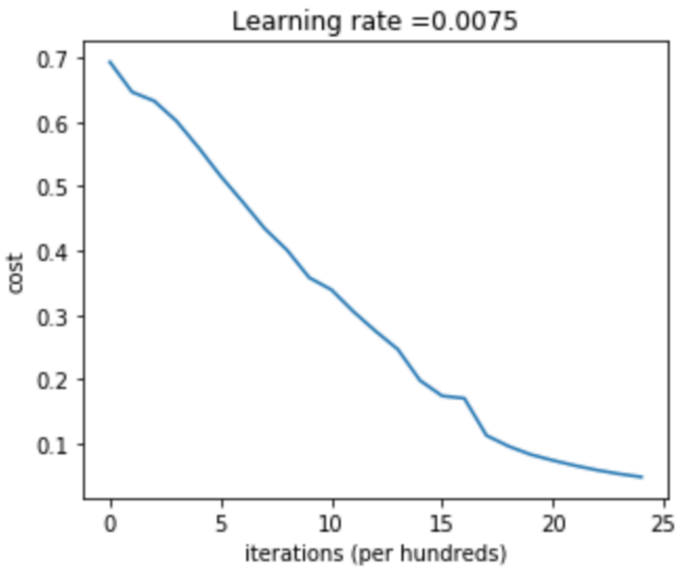
- **Cost function**

  \+ **Entropy Cost Function** with variables are matrices, the general case of **Entropy Cost Function** of **Logistic Regression**

III. Accuracy

1. In training-set

```
Cost after iteration 0: 0.6930497356599888
Cost after iteration 100: 0.6464320953428849
Cost after iteration 200: 0.6325140647912677
Cost after iteration 300: 0.6015024920354665
Cost after iteration 400: 0.5601966311605747
Cost after iteration 500: 0.515830477276473
Cost after iteration 600: 0.4754901313943325
Cost after iteration 700: 0.4339163151225749
Cost after iteration 800: 0.4007977536203887
Cost after iteration 900: 0.3580705011323798
Cost after iteration 1000: 0.3394281538366412
Cost after iteration 1100: 0.3052753636196264
Cost after iteration 1200: 0.27491377282130164
Cost after iteration 1300: 0.24681768210614846
Cost after iteration 1400: 0.19850735037466116
Cost after iteration 1500: 0.1744831811255664
Cost after iteration 1600: 0.17080762978096148
Cost after iteration 1700: 0.11306524562164734
Cost after iteration 1800: 0.09629426845937152
Cost after iteration 1900: 0.08342617959726863
Cost after iteration 2000: 0.07439078704319081
Cost after iteration 2100: 0.0663074813226793
Cost after iteration 2200: 0.0591932950103817
Cost after iteration 2300: 0.053361403485605585
Cost after iteration 2400: 0.04855478562877016
```



2. **In test-set**
   Accuracy: 0.72

IV. Conclusion

1. **Shortcomings of Model & Acknowledgement**

- **Accuracy** is not high in test-set. This is because the model is still in small scale and the number of data in datasets is just 10.000

- **Methods** used in general network are still simple

- **Actually**, deep neural networks is not a good model in the computer vision tasks.

2. Proposed improvements

- **Collect more data in more sources to train**, build **a more sophisticated model** in term of **size of input image**, **number of hidden layers** and **use some advanced methods** accompanying with such models such as add **Regularization** (to avoid Overfitting), **Turning many more Hyper-parameters**, use many advanced Optimization methods instead such as Adam Algorithm.

- Try with **Tan-h functions** as some Activation Fuctions instead of **ReLu**

- Use **Convolutional Neural Networks** instead of Deep Neural Network.

------------------------- Thanks for visiting! ----------------------------