[Week 1 - PA 1 - Initialization](https://nbviewer.jupyter.org/github/amanchadha/coursera-deep-learning-specialization/blob/master/C2%20-%20Improving%20Deep%20Neural%20Networks%20Hyperparameter%20tuning%2C%20Regularization%20and%20Optimization/Week%201/Initialization/Initialization.ipynb)

1 - Neural Network model

2 - Zero initialization

3 - Random initialization

4 - He initialization

5 - Conclusions

[Week 1 - PA 2 - Regularization](https://nbviewer.jupyter.org/github/amanchadha/coursera-deep-learning-specialization/blob/master/C2%20-%20Improving%20Deep%20Neural%20Networks%20Hyperparameter%20tuning%2C%20Regularization%20and%20Optimization/Week%201/Regularization/Regularization_v2a.ipynb)

1 - Non-regularized model

**def** model(X, Y, learning\_rate = 0.3, num\_iterations = 30000, print\_cost = **True**, lambd = 0, keep\_prob = 1):

**return** parameters

2 - L2 Regularization

**def** compute\_cost\_with\_regularization(A3, Y, parameters, lambd):

**return** cost

**def** backward\_propagation\_with\_regularization(X, Y, cache, lambd):

**return** gradients

3 - Dropout

3.1 - Forward propagation with dropout

**def** forward\_propagation\_with\_dropout(X, parameters, keep\_prob = 0.5):

**return** A3, cache

B1: Khởi tạo ngẫu nhiên D1 = rand

B2: Chuyển thành dạng 1 0 D1 = (D1 < keep\_prob) . astype(int)

B3: A1 = A1 \* D1

B4: A1 = A1 / keep\_prob

3.2 - Backward propagation with dropout

**def** backward\_propagation\_with\_dropout(X, Y, cache, keep\_prob):

**return** gradients

B1: dA2 = dA2 \* D2

B2: dA2 = dA2 / keep\_prob

Lưu ý với Dropout:

* Dropout là một kỹ thuật điều chuẩn
* Chỉ sử dụng Dropout trong quá trình train. Tắt Dropout khi test nhé
* Dropout tiến thì phải nhớ Dropout lùi nhé
* Nhớ Dropout ngược

4 - Conclusions

Những gì chúng tôi muốn bạn ghi nhớ từ sổ tay này :

* Điều chuẩn làm giảm overfitting
* Điều chuẩn làm giảm giá trị của trọng số
* Điều chuẩn L2 và Dropout là 2 kỹ thuật điều chuẩn hiệu quả

[Week 1 - PA 3 - Gradient Checking](https://nbviewer.jupyter.org/github/amanchadha/coursera-deep-learning-specialization/blob/master/C2%20-%20Improving%20Deep%20Neural%20Networks%20Hyperparameter%20tuning%2C%20Regularization%20and%20Optimization/Week%201/Gradient%20Checking/Gradient%20Checking%20v1.ipynb)

1) How does gradient checking work?

2) 1-dimensional gradient checking

**def** forward\_propagation(x, theta):

**return** J

**def** backward\_propagation(x, theta):

**return** dtheta

**def** gradient\_check(x, theta, epsilon = 1e-7):

**return** difference

3) N-dimensional gradient checking



**def** forward\_propagation\_n(X, Y, parameters):

**return** cost, cache

**def** backward\_propagation\_n(X, Y, cache):

**return** gradients

**def** gradient\_check\_n(parameters, gradients, X, Y, epsilon = 1e-7):

**return** difference

def dictionary\_to\_vector(parameters):

    """

    Roll all our parameters dictionary into a single vector satisfying our specific required shape.

    """

    keys = []

    count = 0

    for key in ["W1", "b1", "W2", "b2", "W3", "b3"]:

        # flatten parameter

        new\_vector = np.reshape(parameters[key], (-1,1))

        keys = keys + [key]\*new\_vector.shape[0]

        if count == 0:

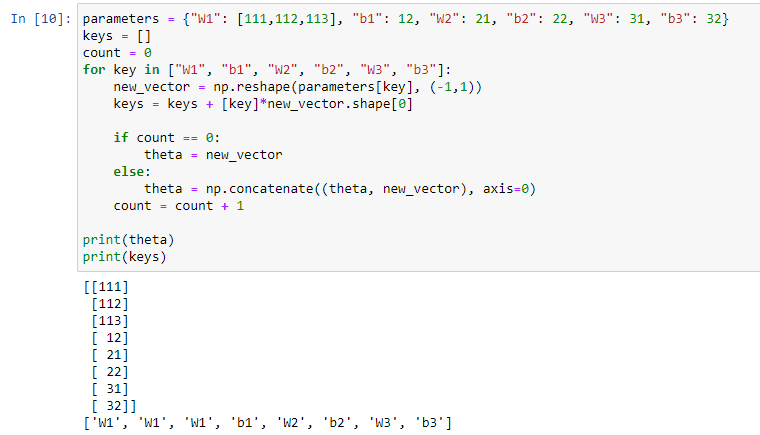
            theta = new\_vector

        else:

            theta = np.concatenate((theta, new\_vector), axis=0)

        count = count + 1

    return theta, keys



[Week 2 - PA 4 - Optimization Methods](https://nbviewer.jupyter.org/github/amanchadha/coursera-deep-learning-specialization/blob/master/C2%20-%20Improving%20Deep%20Neural%20Networks%20Hyperparameter%20tuning%2C%20Regularization%20and%20Optimization/Week%202/Optimization_methods_v1b.ipynb)

1 - Gradient Descent

Những gì bạn nên nhớ :

* Sự khác biệt giữa gradient descent, mini-batch gradient descent và stochastic gradient descent là số lượng ví dụ bạn sử dụng để thực hiện một bước cập nhật.
* Bạn phải điều chỉnh một siêu tham số tốc độ học tập α
* Với mini-batch GD size được hiệu chỉnh tốt, thường thì nó sẽ hoạt động tốt hơn GD và SGD (đặc biệt là với dataset lớn)

2 - Mini-Batch Gradient descent

Những gì bạn nên nhớ :

* Trộn và Phân vùng là 2 bước cần thiết để xây dựng mini-batches
* Lũy thừa của 2 thường được chọn làm mini-batch size

3 – Momentum

Câu hỏi 1: Tại sao không đợi cập nhật xong hẳn v[dW] thì update W một thể?

Trả lời:

* Mỗi lần cập nhật v[dW] là giá trị dW của 1 mini-batch. Thì lúc này cập nhật W tại mini-batch đó luôn. Đúng rồi còn gì.
* Kết quả sau nhiều lần lặp là để vẽ được 1 line.
* Chứ đừng nhầm lần rằng, mỗi lần lặp vẽ được 1 line, và nhiều lần lặp thì ra được 1 line hoàn hảo. Sai hoàn toàn nhé.

4 - Adam

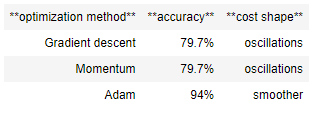
5 - Model with different optimization algorithms

5.1 - Mini-batch Gradient descent

5.2 - Mini-batch gradient descent with momentum

5.3 - Mini-batch with Adam mode

5.4 – Summary



[Week 3 - PA 5 - TensorFlow Tutorial](https://nbviewer.jupyter.org/github/amanchadha/coursera-deep-learning-specialization/blob/master/C2%20-%20Improving%20Deep%20Neural%20Networks%20Hyperparameter%20tuning%2C%20Regularization%20and%20Optimization/Week%203/Tensorflow_introduction.ipynb)

1 - Packages

1.1 - Checking TensorFlow Version

2 - Basic Optimization with GradientTape

2.1 - Linear Function

2.2 - Computing the Sigmoid

2.3 - Using One Hot Encodings

2.4 - Initialize the Parameters

3 - Building Your First Neural Network in TensorFlow

3.1 - Implement Forward Propagation

3.2 - Compute the Cost

3.3 - Train the Model

4 - Bibliography