计算机科学与技术学院神经网络与深度学习课程实验报告

实验题目：Improve deep neural networks		学号：201900130143
日期：10/21	班级： 智能班	姓名：吴家麒
Email：wjq_777@126.com		

实验目的：

In this assignment you will master basic neural network adjustment skills and try
to improve deep neural networks: Hyperparameter tuning, Regularization and Optimization, Batch Normalization.

- this time you will be given two subtasks: Regularization, Batch Normalization.

实验软件和硬件环境：

Anaconda3 + Jupyter notebook

实验原理和方法：

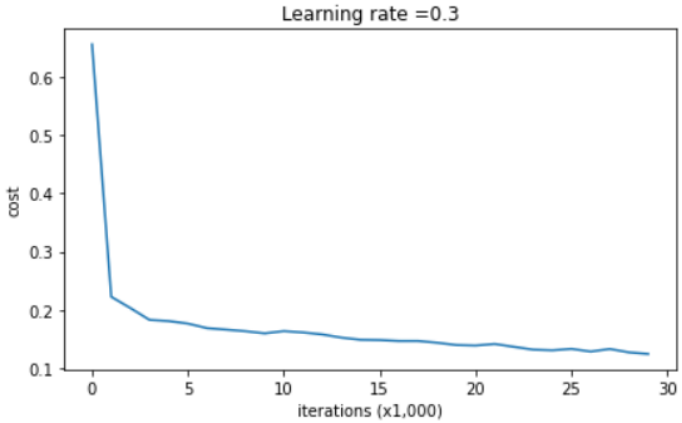
在机器学习的训练过程中，随着模型整体复杂度的提升，训练结果总会伴随着过拟合的现象，而避免过拟合的常见方法就是 regularization 和 batch Normalization。

实验步骤：（不要求罗列完整源代码）

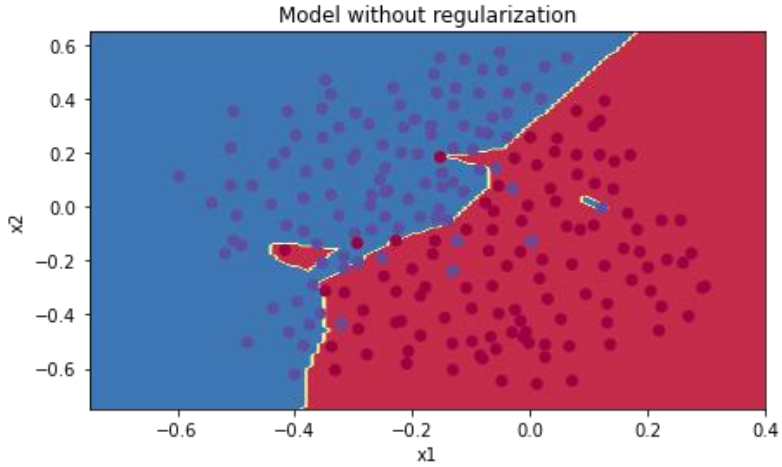
1、 Regularization

下面我们先来看看不使用正则化进行模型训练的结果。训练数据集：法国队最近十场比赛中己方球员和对方球员头球的位置。

```
Cost after iteration 0: 0.6557412523481002
Cost after iteration 10000: 0.1632998752572419
Cost after iteration 20000: 0.13851642423239133
```




```
On the training set:
Accuracy: 0.9478672985781991
On the test set:
Accuracy: 0.915
```



很显然，模型训练的结果出现了一定程度的过拟合。

接下来我们为损失函数添加上 L2-范式：

$$\underbrace{\frac{1}{m}\frac{\lambda}{2}\sum_{l}\sum_{k}\sum_{j}W_{k,j}^{[l]2}}_{\text{L2 regularization cost}}$$

计算出此时的 cost：

```
A3, Y_assess, parameters = compute_cost_with_regularization_test_case()

print("cost = " + str(compute_cost_with_regularization(A3, Y_assess, parameters, lambd = 0.1)))
```

```
cost = 1.7864859451590758
```

Expected Output:

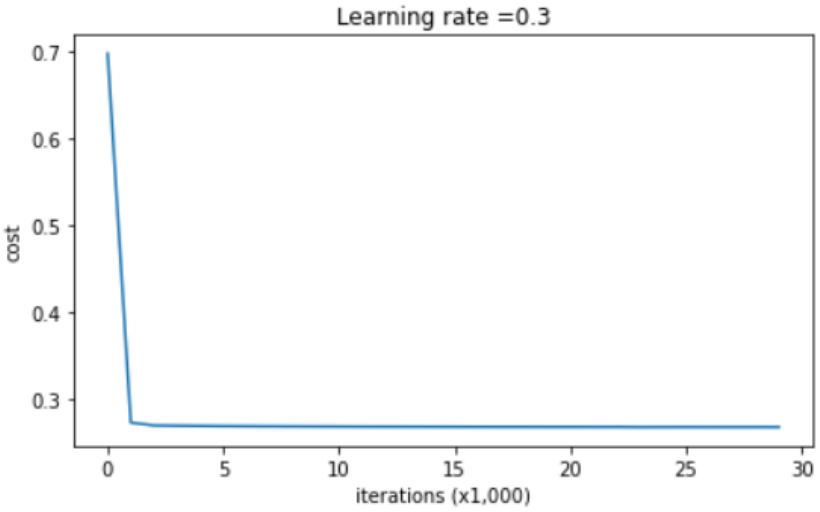
cost	1.78648594516

再修改反向传播算法，得到梯度下降的结果：

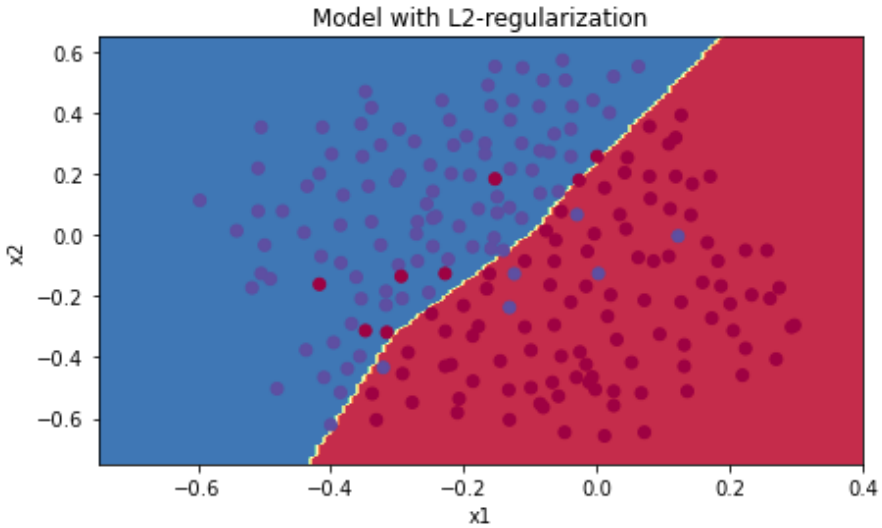
```
dW1 = [[-0.25604646  0.12298827 -0.28297129]
 [-0.17706303  0.34536094 -0.4410571 ]]
dW2 = [[ 0.79276486  0.85133918]
 [-0.0957219  -0.01720463]
 [-0.13100772 -0.03750433]]
dW3 = [[-1.77691347 -0.11832879 -0.09397446]]
```

查看模型训练情况：

```
Cost after iteration 0: 0.6974484493131264
Cost after iteration 10000: 0.2684918873282239
Cost after iteration 20000: 0.2680916337127301
```




```
On the train set:
Accuracy: 0.9383886255924171
On the test set:
Accuracy: 0.93
```



除了 L2 范式，dropout 也是一种可以理解为正则项的方法，在每次的神经网络的反向传播中，会随机选择一些神经元，设定其反向传播对应的参数为 0，然后对于被改

变后的神经网络进行反向传播。

对于带有 dropout 的前向传播过程，通过设置了两层的 dropout 层，得到了最后的结果：

```
X_assess, parameters = forward_propagation_with_dropout_test_case()

A3, cache = forward_propagation_with_dropout(X_assess, parameters, keep_prob = 0.7)
print ("A3 = " + str(A3))
```

```
A3 = [[0.36974721 0.00305176 0.04565099 0.49683389 0.36974721]]
```

Expected Output:

A3	[[0.36974721 0.00305176 0.04565099 0.49683389 0.36974721]]

对应的，反向传播也需要设置两层，得到梯度下降的结果：

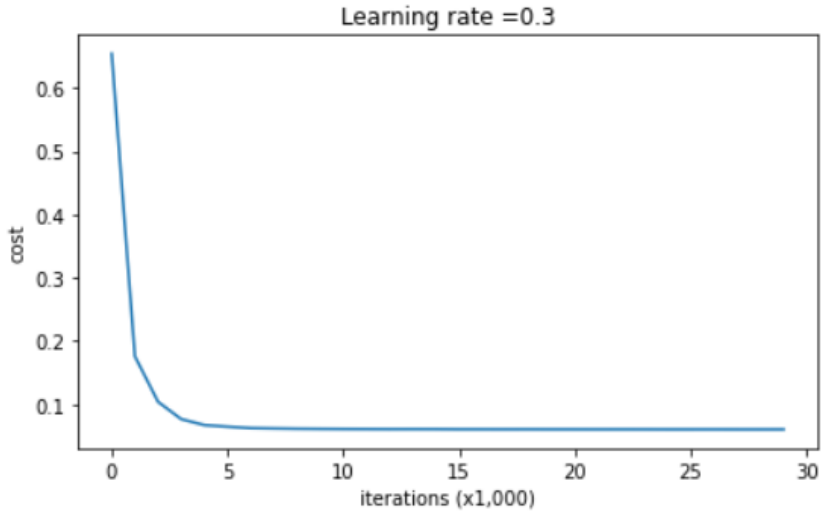
```
X_assess, Y_assess, cache = backward_propagation_with_dropout_test_case()

gradients = backward_propagation_with_dropout(X_assess, Y_assess, cache, keep_prob = 0.8)

print ("dA1 = " + str(gradients["dA1"]))
print ("dA2 = " + str(gradients["dA2"]))
```

```
dA1 = [[ 0.36544439  0.         -0.00188233  0.         -0.17408748]
 [ 0.65515713  0.         -0.00337459  0.         -0.        ]]
dA2 = [[ 0.58180856  0.         -0.00299679  0.         -0.27715731]
 [ 0.          0.53159854 -0.          0.53159854 -0.34089673]
 [ 0.          0.         -0.00292733  0.         -0.        ]]
```

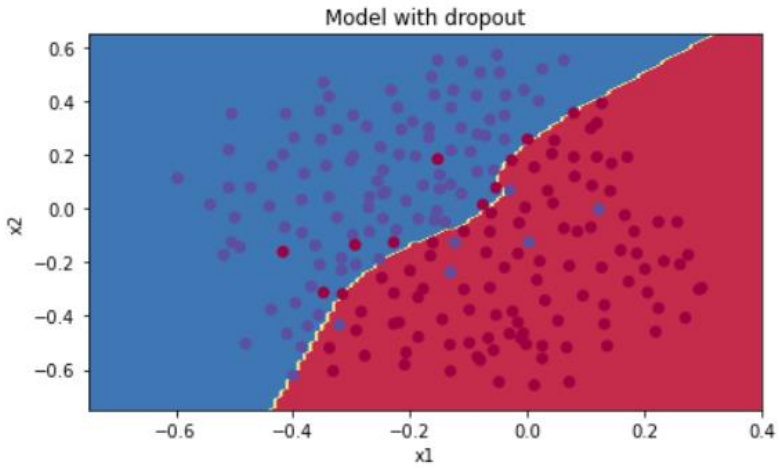
Expected Output:

dA1	[[0.36544439 0. -0.00188233 0. -0.17408748] [0.65515713 0. -0.00337459 0. -0.]]
dA2	[[0.58180856 0. -0.00299679 0. -0.27715731] [0. 0.53159854 -0. 0.53159854 -0.34089673] [0. 0. -0.00292733 0. -0.]]

查看模型训练情况；




```
On the train set:
Accuracy: 0.9289099526066351
On the test set:
Accuracy: 0.95
```



2、 Batch Normalization

使深层神经网络更容易训练的另一种方式是改变网络的结构，也就是 Batch Normalization，这个过程通过在网络中添加 batch normalization layers 来实现。

Batch normalization: forward

在训练时，批处理归一化层使用 minibatch 估计每个特征的均值和标准差，这些估值用于中心化和归一化 minibatch 的特征。

前向传播计算结果：

train

```
Before batch normalization:
  means:  [ -2.3814598  -13.18038246   1.91780462]
  stds:   [27.18502186 34.21455511 37.68611762]

After batch normalization (gamma=1, beta=0)
  means:  [5.32907052e-17 7.04991621e-17 4.22578639e-17]
  stds:   [0.99999999 1.         1.        ]

After batch normalization (gamma= [1. 2. 3.] , beta= [11. 12. 13.] )
  means:  [11. 12. 13.]
  stds:   [0.99999999 1.99999999 2.99999999]
```

test

```
After batch normalization (test-time):
  means:  [-0.03927354 -0.04349152 -0.10452688]
  stds:   [1.01531428 1.01238373 0.97819988]
```

Batch normalization: backward

反向传播检验结果：

```
dx error:  1.7029241291468676e-09
dgamma error:  7.420414216247087e-13
dbeta error:  2.8795057655839487e-12
```

通过演算简化梯度实现简化的 Batch Normalization 反向传递：

```
dx difference:  1.0475263156312713e-10
dgamma difference:  0.0
dbeta difference:  0.0
speedup: 1.52x
```

结论分析与体会：

1、 通过对于正则化方法的学习，实际感受了利用正则化项对模型进行调整，从而降低模型训练过拟合的问题，也得到了无正则化与 L2 范式、dropout 之间的比较：

model	**train accuracy**	**test accuracy**
3-layer NN without regularization	95%	91.5%
3-layer NN with L2-regularization	94%	93%
3-layer NN with dropout	93%	95%

2、 除了使用更复杂的优化过程，如 SGD+momentum、RMSProp 或 Adam 等方法对网络进行优化，也可以通过 Batch Normalization 的方式改变网络结构来使训练更容易。BN 就是通过方法将该层特征值分布重新拉回标准正态分布，特征值将落在激活函数对于输入较为敏感的区间，输入的小变化可导致损失函数较大的变化，使得梯度变大，避免梯度消失，同时也可加快收敛。通过实验我们也可以看到经过 BN 后得到了不错的结果。