# Building your Deep Neural Network&Application

# 1、作业简介

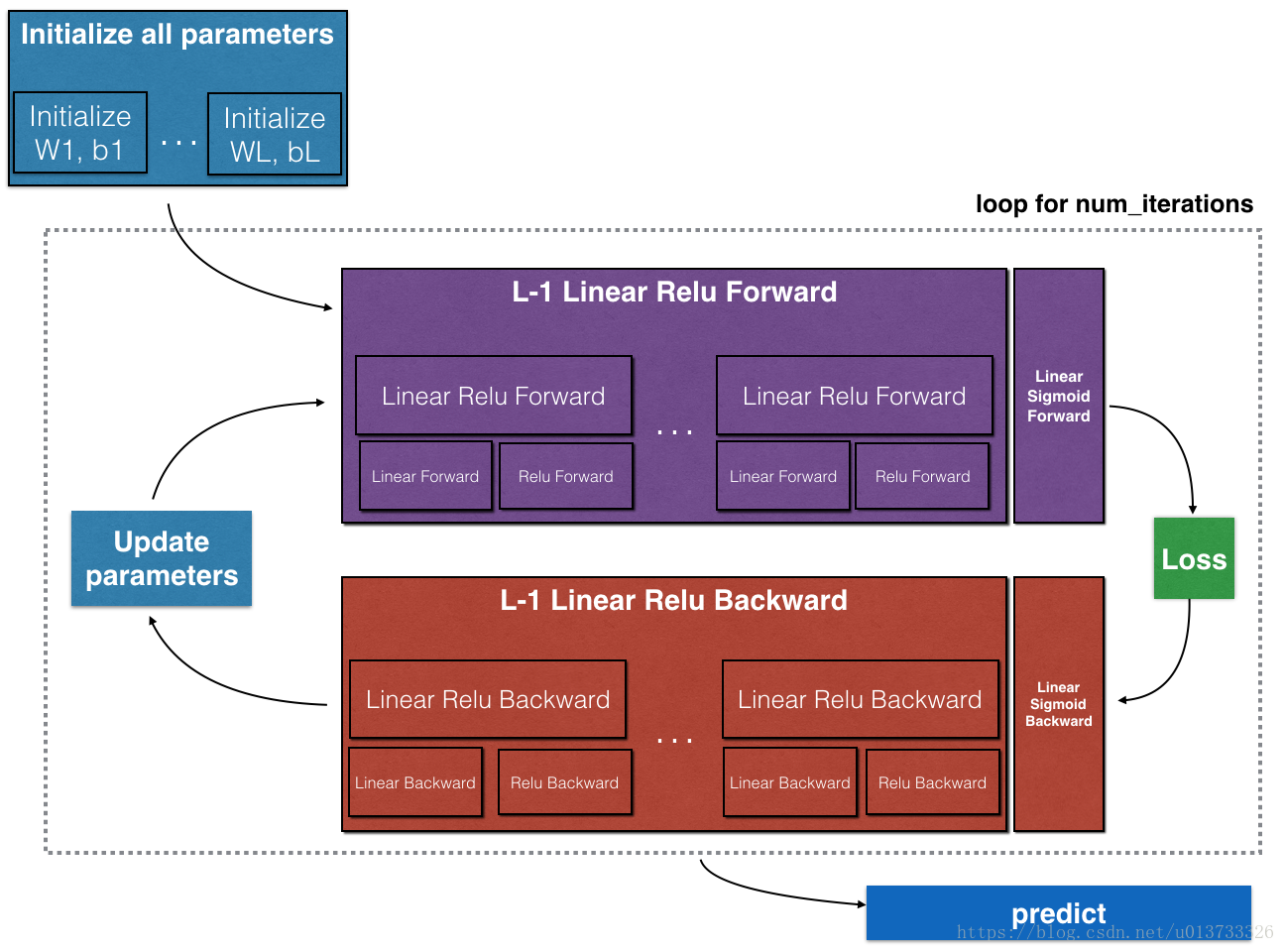
本文介绍如何利用Python来实现具有多个隐藏层的图片分类问题。通过这次建立的多层神经网络模型，可以将之前的猫分类问题的准确率提升到80%。

要点：1. 使用非线性映射单元（例如ReLU）去改善模型

2. 建立一个多个隐藏层的神经网络

3. 创建一个易于调用的模型类

在本次作业中，我们要构建两个神经网络，一个是构建两层的神经网络，一个是构建多层的神经网络，多层神经网络的层数可以自己定义。本次作业的难度有所提升，但是力求深入简出。在这里，先简单的讲一下难点，本文会提到\*\*[LINEAR-> ACTIVATION]转发函数，比如我有一个多层的神经网络，结构是输入层->隐藏层->隐藏层->···->隐藏层->输出层\*\*，在每一层中，我会首先计算Z = np.dot(W,A) + b，这叫做【linear\_forward】，然后再计算A = relu(Z) 或者 A = sigmoid(Z)，这叫做【linear\_activation\_forward】，合并起来就是这一层的计算方法，所以每一层的计算都有两个步骤，先是计算Z，再计算A，你也可以参照下图：



## **步骤：**

## 1.初始化网络参数

## 2.前向传播

## 2.1 完成一层的正向传播步骤的线性部分（线性部分即Z=WX+b这部分，输出部分就是A，就是将线性部分的结果输入到激活函数所产生的结果。）

## 2.2 计算激活函数的部分（ReLU使用L-1次，Sigmod使用1次）

## 2.3 将前两个步骤合并为一个新的 [LINEAR->ACTIVATION] 前向传播函数

2.4 堆叠 [LINEAR->RELU]前向传播函数 L-1 次 (从第1层到 L-1层) 和添加一个 [LINEAR->SIGMOID] 在最后(最后一层 L)，提供了一个新的 L\_model\_forward 函数

## 3.计算误差

## 4.反向传播

## 4.1 计算神经网络反向传播的线性部分

## 4.2 计算激活函数（RELU或者sigmoid）的梯度

## 4.3 将前两个步骤合并为一个新的 [LINEAR->ACTIVATION] 后向传播函数

## 4.4 叠加 [LINEAR->RELU] 向后 L-1 次，并在新的 L\_model\_backward 函数中向后添加 [LINEAR->SIGMOID]

## 5.更新参数

## 请注意，对于每个前向函数，都有一个相应的后向函数。 这就是为什么在我们的转发模块的每一步都会在cache中存储一些值，cache的值对计算梯度很有用， 在反向传播模块中，我们将使用cache来计算梯度。 现在正式开始分别构建两层神经网络和多层神经网络。

## 2、工具包

在这个过程中，我们将会用到如下库：

numpy：Python中常用的科学计算库

h5py：Python与H5文件交互的库

matplotlib：用于数据可视化

testCases：提供一些测试用例来评估函数的正确性

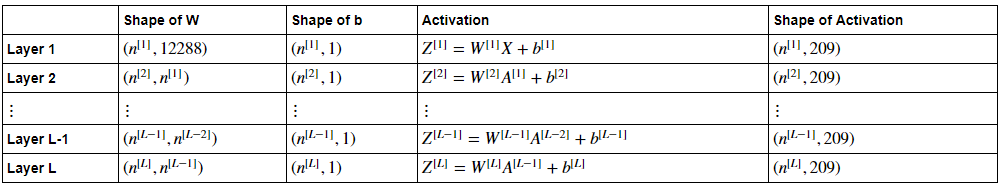
dnn\_utils ：提供一些必要功能

lr\_utils：压缩包内给的自定义函数，功能是导入数据集

## 3、初始化

对于一个两层的神经网络结构而言，模型结构是线性->ReLU->线性->sigmod函数。对于权重矩阵采用随机化方式进行初始化（np.random.randn(shape)\*0.01），对于偏移值b矩阵则采用0矩阵即可(np.zeros(shape))。

对于L层的神经网络，由于涉及到很多的权重矩阵和偏移矩阵显得更加复杂。要特别注意的是矩阵之间的尺寸匹配，n[l]表示l层的神经元数量。例如输入X 的尺寸是 (12288,209) (m=209表示样本数) ：



第l层的W的维度为(layer\_dims[l], layer\_dims[l-1])。而第l层的b的维度为(layer\_dims[l], 1)。

对于L层模型：

* 模型结构： [LINEAR -> RELU] × (L-1) -> LINEAR -> SIGMOID。所以 L−1 层是需要用到 ReLU激活函数的。输出层用的是sigmoid函数。
* 权重矩阵采用仍旧是随机化初始化的方式： np.random.rand(shape) \* 0.01
* 偏移矩阵仍旧是0矩阵进行处初始化： np.zeros(shape).
* 我们将每层的神经元数量n[l]信息进行存储，layer\_dims。例如在平面数据分类模型中 layer\_dims 的值是[2,4,1]，其中输入层的神经元个数是2，隐藏层的神经元个数是4，输出层的神经元个数是1。对应的 W1尺寸= (4,2), b1尺寸= (4,1), W2尺寸= (1,4) ， b2 尺寸= (1,1)。

**4、前向传播**

**4.1线性传播部分**

前向传播中，先进行线性部分计算如下：Z**[l]**=W**[l]**A**[l-1]**+b**[l]**，其中 A**[0]**=X.

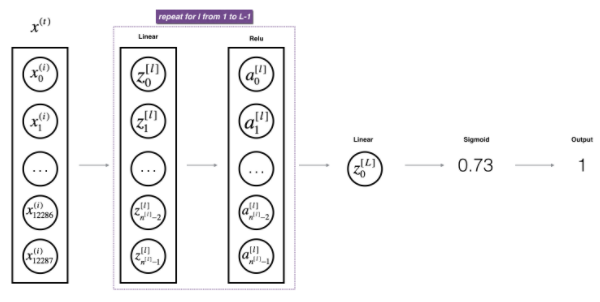
**4.2激活部分**

在激活部分，本次作业会用到两个激活函数：Sigmoid和ReLU。

* Sigmoid：。在这个步骤我们需要两个结果，一个是激活函数的结果值，另一个是包含“Z”的“cache”值，这个我们在后向传播过程需要用到。
* ReLU：其数学表达式： A=ReLU(Z)=max(0,Z)。同样结果值有两部分，其一是激活函数结果值“A”，另一个是包含“Z”的“cache”值。

相邻两层模型的激活函数计算；

上面已经阐述了相邻两层之间的激活模型，那么对于L层的神经网络，激活函数为ReLU的linear\_activation\_forward 需要重复L-1次，而最后的输出层采用的参数为Sigmoid的linear\_activation\_forward 。L层的前向传播如下：



Tips:

复用此前的代码；

循环 [LINEAR->RELU] (L-1) 次；

代码中我们用 AL 表示A[L]=σ(Z[L])=σ(W[L]A[L−1]+b[L])，即 Y^；

注意保持“caches”中的数据。

至此，我们可以计算得到AL的值，该值包含了所有的预测结果。在caches中也记录了中间值。为此，我们可以用AL值来计算代价函数。

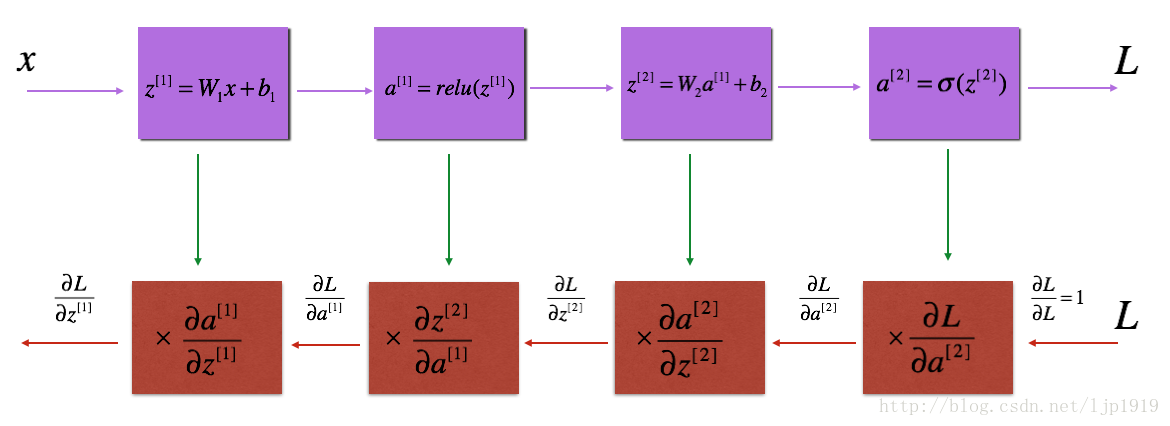
**5、代价函数**

代价函数 J:



**6、反向传播**

反向传播是为了计算各个参数梯度，其模型如下：

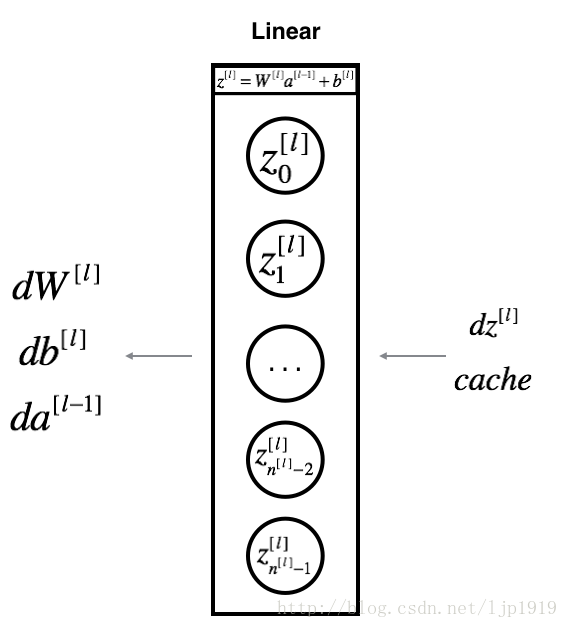


和之前的前向传播类似，反向传播模块的建立分以下三个步骤：

* 反向LINEAR（Linear backward）
* ReLU 或者 sigmoid 激活函数的反向LINEAR -> ACTIVATION
* [LINEAR->RELU]×(L-1) -> LINEAR->SIGMOID backward (whole model)

**6.1反向线性传播部分**

对于l层，linear part= Z**[l]**=W**[l]**A**[l-1]**+b**[l]** ，假设已知，我们想要计算(dW**[l]**,db**[l]**,dA**[l-1]**)。



三个输出(dW**[l]**,db**[l]**,dA**[l]**)可以通过输入dZ**[l]**计算获得。公式如下:







**6.2激活部分**

* 对于sigmoid函数，可以定义两个函数：

sigmoid\_backward:用以计算 SIGMOID单元：

relu\_backward: 用以计算RELU的 backward propagation：

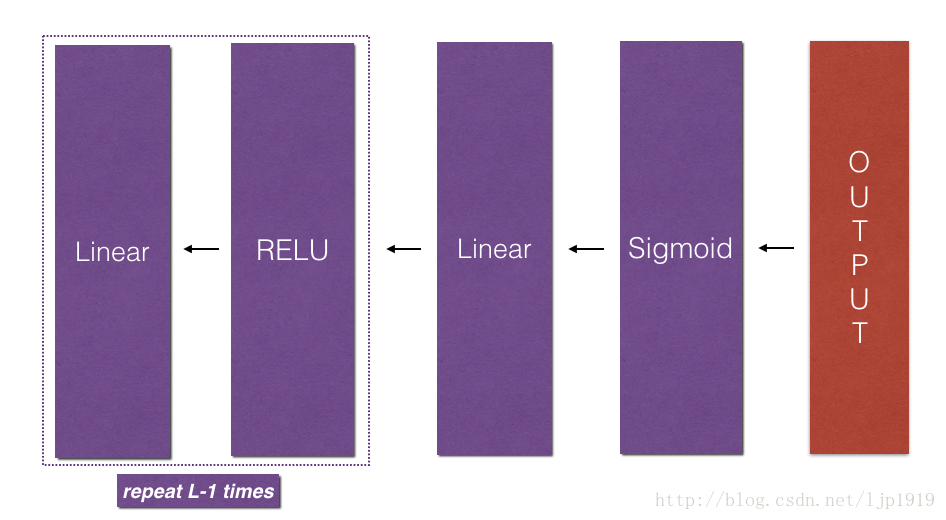
* 对于 g(.)的激活函数：

sigmoid\_backward 和relu\_backward 的计算如下



**6.3 L层模型的后向传播**

现在我们开始对整个神经网络做后向传播，定义函数为L\_model\_forward。在每次的迭代过程中，我们都将 cache值=(X,W,b, z)保留，用以后向模块中梯度的计算。在L\_model\_forward中，我们是重复了L次上述的步骤。



**7、参数更新**

采用梯度下降进行参数的更新：



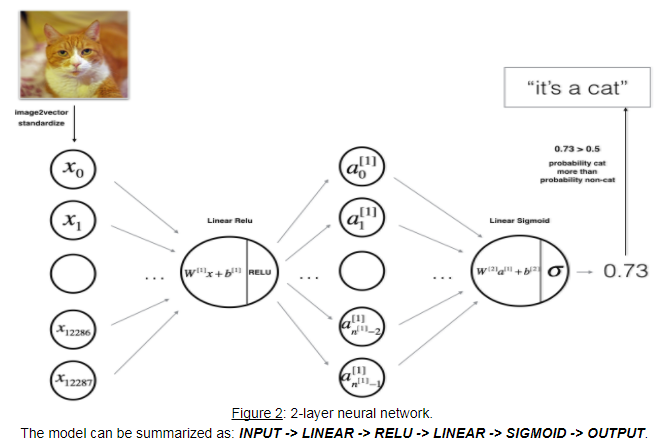


其中α是学习率。

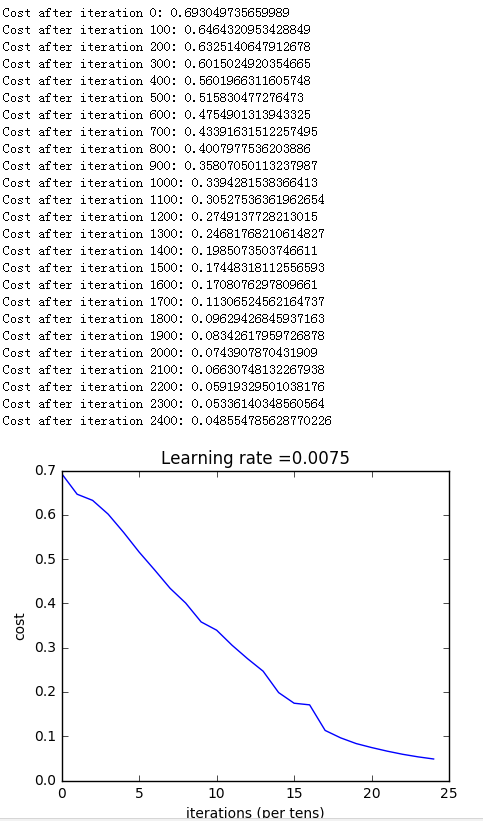
至此为止已经实现该神经网络中所有需要的函数。将这些方法组合在一起，构成一个神经网络类，可以方便的使用。

**8、两层神经网络模型**

一个两层的神经网络模型图如下：



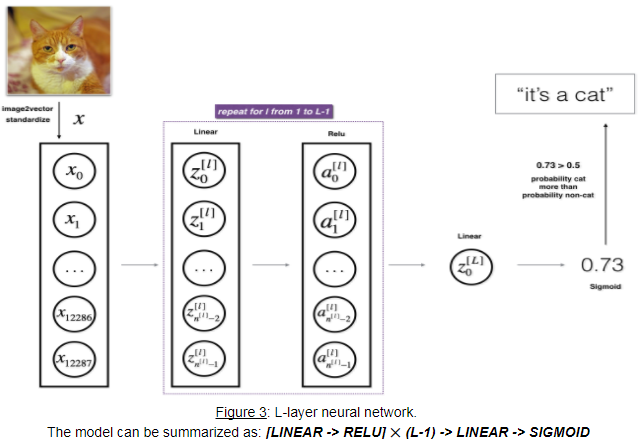
训练一下：



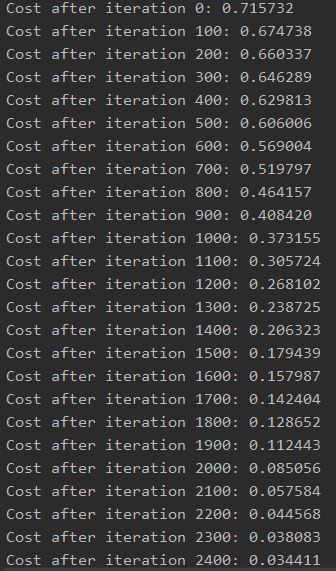
由预测结果可以得到，这次的测试集的准确率要比第一次Logistic Regression with a Neural Network mindset作业中的高一些，上次的是70%，这次是72%。

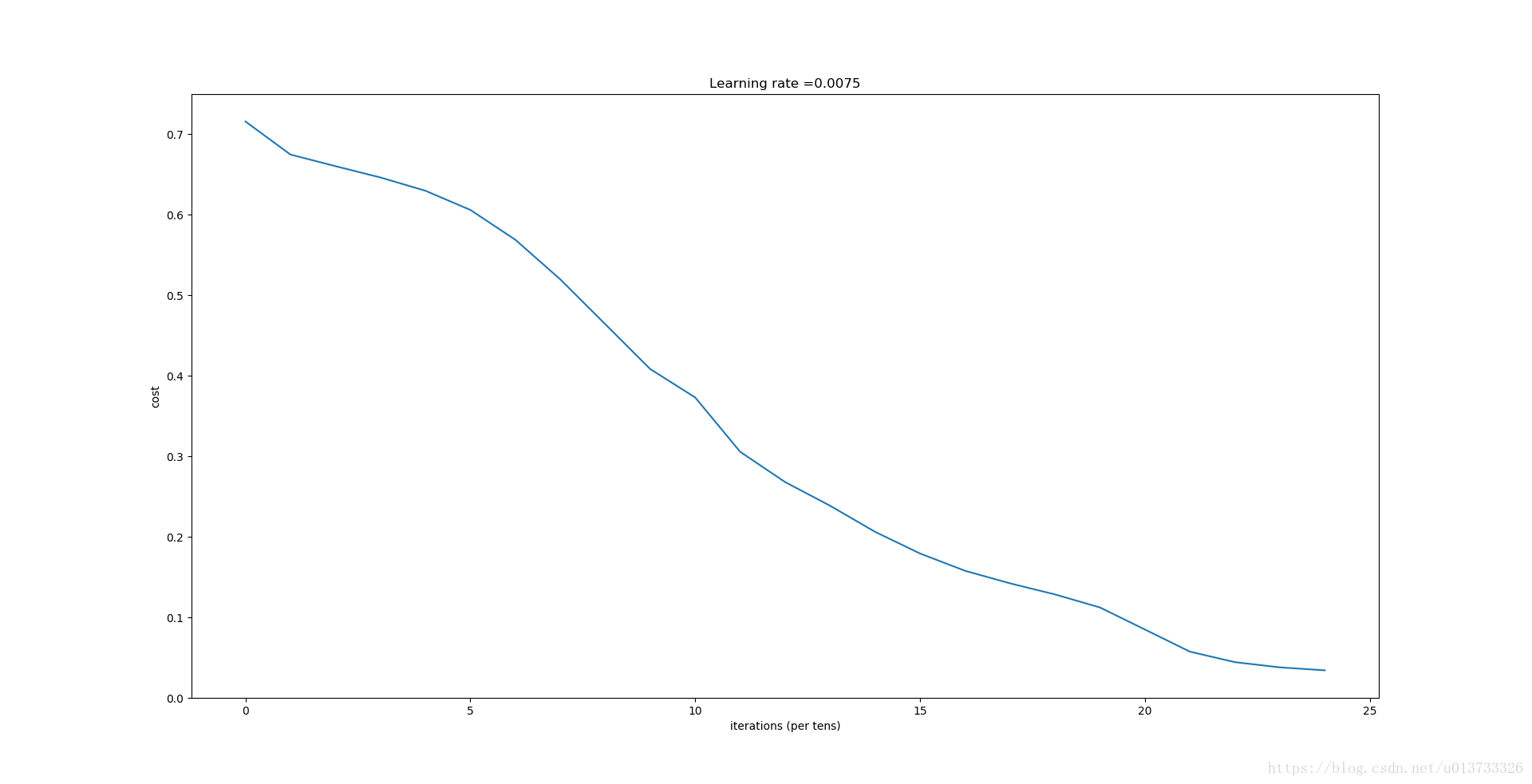
**9、多层神经网络**

一个多层的神经网络模型如下：



训练一下：





由预测结果可以得到多层神经网络的测试集准确率为78%。就准确度而言，从70%到72%再到78%，可以看到的是准确度在一点点增加，当然，也可以手动的去调整layer\_dims，准确度可能又会提高一些。

**10、分析**

我们可以看一看有哪些东西在L层模型中被错误地标记了，导致准确率没有提高。

运行结果：



根据运行结果分析一下被标记错误的原因，模型往往表现欠佳的几种类型的图像包括：

* 猫身体在一个不同的位置
* 猫出现在相似颜色的背景下
* 不同的猫的颜色和品种
* 相机角度
* 图片的亮度
* 比例变化（猫的图像非常大或很小）

**11、测试自己的图片**