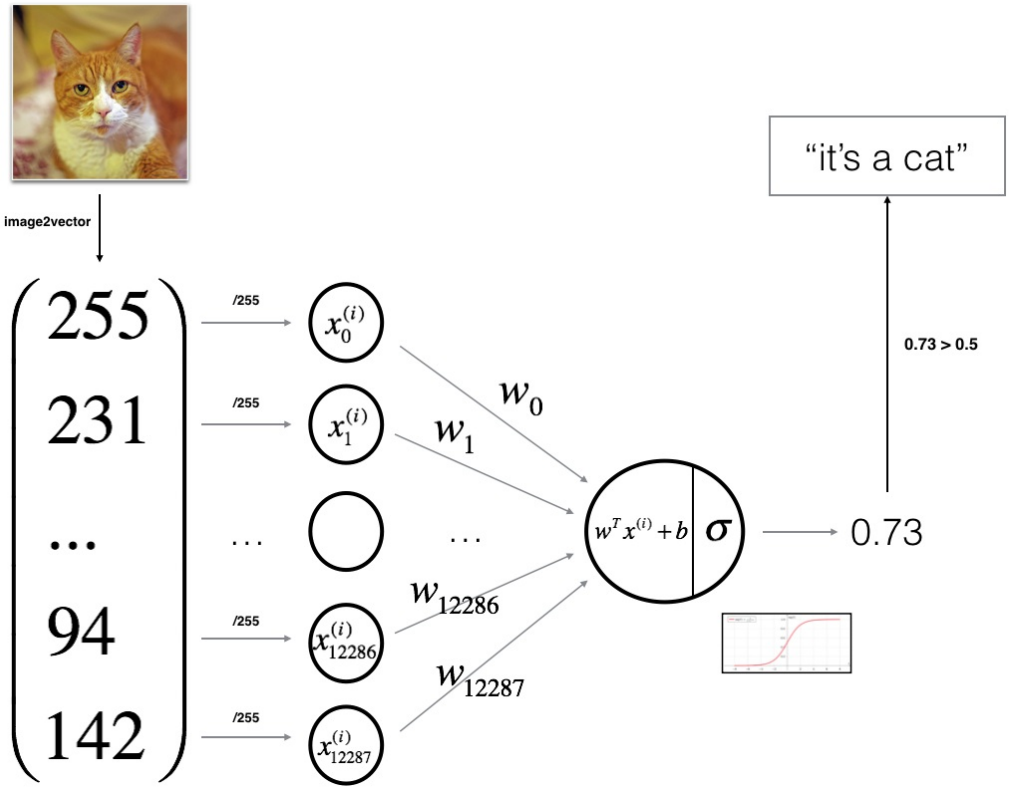
Python作业第五周

项目发起人:`[江柳]`
项目发起时间:`[2017.10.30]`

1.课程阅读

Python Numpy Tutorial

本次作业目标:
通过一个简单的Logistic Regression模型对猫的图片实现识别。以神经元描述的整个结构如下:



该算法用到的一些数学表达，对图片 $x^{(i)}$:

$$z^{(i)} = w^T x^{(i)} + b$$

$$\hat{y}^{(i)} = a^{(i)} = sigmoid(z^{(i)})$$

$$\mathcal{L}(a^{(i)}, y^{(i)}) = -y^{(i)} \log(a^{(i)}) - (1 - y^{(i)}) \log(1 - a^{(i)})$$

通过对整个样本集里的所有图片的Loss求和得到损失:

$$J = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(a^{(i)}, y^{(i)})$$

涉及内容:

- Logistic Regression
- 损失函数的矩阵计算

- 函数梯度的矩阵计算
- 预测值的矩阵计算
- 损失曲线的显示

注:

- 本次作业全部代码在`ex5.py`文件中完成

2.作业布置

2.1 数据载入

在文件夹`datasets`有两个文件`train_catvnoncat.h5`和`test_catvnoncat.h5`，两个文件均使用HDF5文件格式保存。分别表示训练样本集和测试样本集。该文件格式可以使用Python中的h5py库存取。

> 一个HDF5文件是一种存放两类对象的容器：dataset和group. Dataset是类似于数组的数据集，而group是类似文件夹一样的容器，存放dataset和其他group。在使用h5py的时候需要牢记一句话：groups类比词典，dataset类比Numpy中的数组。
> HDF5的dataset虽然与Numpy的数组在接口上很相近，但是支持更多对外透明的存储特征，如数据压缩，误差检测，分块传输。

> 读取HDF5内容
>
> > import h5py #导入h5py库
> > datasets = h5py.File('xxxx.h5', 'r') #打开h5文件

> 查看HDF5中包含的Dataset
>
> > list(datasets.keys()) #datasets文件中包含以下几个Dataset
> > ['list_classes', 'train_set_x', 'train_set_y']

> 读取指定的Dataset

> X = datasets["train_set_x"][:]

尝试补全`load_dataset()`函数代码，使该函数能够读取`datasets`文件夹下的内容并返回numpy.array类型`train_set_x_orig, train_set_y, test_set_x_orig, test_set_y, classes`。

完成后直接运行`ex5.py`文件，输出结果如下:

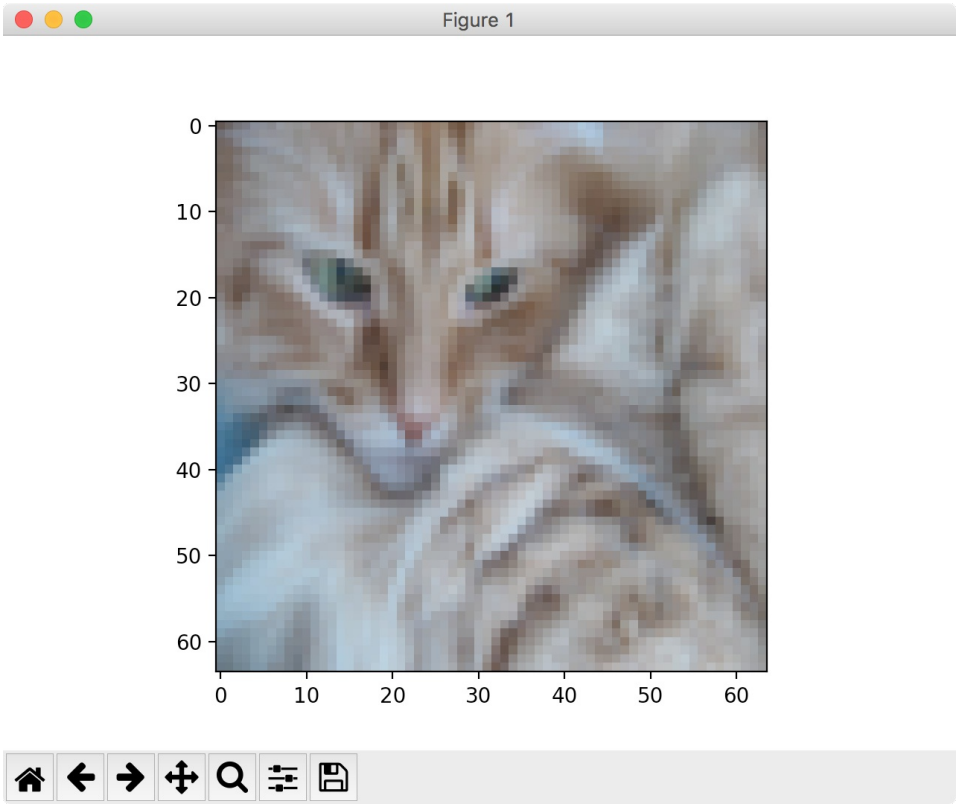
```
Loading Data ...
The shape of the variable train_set_x_orig is (209, 64, 64, 3
The shape of the variable train_set_y is (1, 209)
The shape of the variable test_set_x_orig is (50, 64, 64, 3)
The shape of the variable test_set_y is (1, 50)
Expected shape of the variables are:
(209, 64, 64, 3)
(1, 209)
(50, 64, 64, 3)
(1, 50)
(1, 2)
```

2.2 显示图像

上一步完成后，你会发现train_set_x_orig是一个(209, 64, 64, 3)的矩阵，这表示在train_set_x_orig中有209张图像每张图片长宽分别是64个像素，且有RGB三个通道的颜色。
尝试补全`imshow(train_set_x_orig, idx)`函数代码，使该函数能够显示train_set_x_orig中序号为idx的图像。

> 注：自行搜索plt.imshow()的用法

完成后直接运行`ex5.py`文件，输出结果如下:



2.3 数据展开

为了方便后续的计算，将每张图片从(num_px, num_px, 3)展开成(num_px*num_px*3, 1)。这样整个train_set_x_orig将从(209, 64, 64, 3)的矩阵转变成一个(12288, 209)的矩阵，test_set_x_orig同理如此。
尝试补全`flatten_shape(train_set_x_orig, test_set_x_orig)`函数代码，使该函数能够对这两个数据进行展开。
完成后直接运行`ex5.py`文件，输出结果如下:

```
Flatten Shape ...
The shape of the variable train_set_x_flatten is (12288, 209)
The shape of the variable test_set_x_flatten is (12288, 50)
Expected shape of the variables are:
(12288, 209)
(12288, 50)
```

2.4 特征缩放

在机器学习中我们一般都要对样本进行特征缩放，比如在上一个练习中我

们使用Mean normalization方法进行特征缩放。但是对于图片样本，每个像素点的数值范围从0到255，所以我们一般只需要对每个像素点除以255，即可完成特征缩放工作。

尝试补全`featureScaling(vals)`函数代码，使该函数能够对`vals`根据上式子完成归一化。

完成后直接运行`ex5.py`文件，输出结果如下:

```
Feature Scaling ...
train_set_x[:3, 0] =  [ 0.06666667  0.12156863  0.21960784]
Expected train_set_x[:3, 0] =  [ 0.06666667  0.12156863  0.21
test_set_x[:3, 0] =  [ 0.61960784  0.40784314  0.3254902 ]
Expected test_set_x[:3, 0] =  [ 0.61960784  0.40784314  0.325
```

2.5 S型函数

在Logistic Regression中我们使用sigmoid函数作为激活函数，表达式如下:

$$sigmoid(z^{(i)}) = \frac{1}{1+e^{-z^{(i)}}}$$

尝试补全`sigmoid(z)`函数代码，使该函数能够对`z`根据上式子完成激活，并返回激活后的数值。
完成后直接运行`ex5.py`文件，输出结果如下:

```
Checking Sigmoid ...
test_sigmoid =  [[ 0.5         0.88079708]]
Expected test_sigmoid =  [[ 0.5         0.88079708]]
```

备注:

> numpy.exp(X) 自行查找资料

2.6 初始化参数

在本次实例中，我们不再把所有的参数都集成为θ，而是分成了w和b参数。在Logistic Regression并未对参数的初始值有要求，在这里请将w和b全部初始化为0.0。
尝试补全`initialize_with_zeros(dim)`函数代码，使该函数返回的w,b与参数dim对应。
完成后直接运行`ex5.py`文件，输出结果如下:

```
Initialize Parameters ....
The shape of the variable w is (12288, 1)
Expected shape of the variable w is (12288, 1)
b =  0.0
Expected b = 0.0
```

2.7 正反向传播

在本次实例中，我们将前向传播过程和损失计算以及反向传播过程在一个函数中完成。
Forward Propagation:

- You get X
- You compute $A = \sigma(w^T X + b) = (a^{(0)}, a^{(1)}, ..., a^{(m-1)}, a^{(m)})$
- You calculate the cost function:
  $J = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log(a^{(i)}) + (1 - y^{(i)}) \log(1 - a^{(i)})$

Here are the two formulas you will be using:

$$\frac{\partial J}{\partial w} = \frac{1}{m} X(A - Y)^T$$

$$\frac{\partial J}{\partial b} = \frac{1}{m} \sum_{i=1}^{m} (a^{(i)} - y^{(i)})$$

尝试补全`propagate(w, b, X, Y)`函数代码，使该函数返回一个包含dw

和db的字典grads以及损失cost。

完成后直接运行`ex5.py`文件，输出结果如下:

```
Checking Propagate ...
dw = [[ 0.99845601]
 [ 2.39507239]]
Expected dw = [[ 0.99845601]     [ 2.39507239]])
db = 0.00145557813678
Expected db = 0.00145557813678
cost = 5.801545319394553
Expected cost = 5.801545319394553
```

2.8 参数优化

目前

- 你已经初始化好了参数
- 你已经计算了损失以及它的梯度
- 下一步你应该使用梯度下降算法来优化w和b

尝试补全`optimize(w, b, X, Y, num_iterations, learning_rate, print_cost = False)`函数代码，让该函数能够学习到使J最小化的w和b。并返回包括w和b的字典params，以及包括dw和db的字典grads，以及所有迭代过程中损失记录列表costs。

$$\theta = \theta - \alpha \, d\theta$$

完成后直接运行`ex5.py`文件，输出结果如下:

```
Parameter optimization ...
Cost after iteration 0: 5.801545
w = [[ 0.19033591]
 [ 0.12259159]]
Expected w = [[ 0.19033591]      [ 0.12259159]]
b = 1.92535983008
Expected b = 1.92535983008
dw = [[ 0.67752042]
 [ 1.41625495]]
Expected dw = [[ 0.67752042]     [ 1.41625495]]
db = 0.219194504541
Expected db = 0.219194504541
```

2.9 数值预测

预测的两个步骤:

- 计算$\hat{Y} = A = \sigma(w^T X + b)$
- 将$\hat{Y}$转换成0(如果激活值<=0.5)或1(反之)，并存储在Y_prediction变量中

尝试补全`predict(w, b, X)`函数，使其返回正确的Y_prediction。

完成后直接运行`ex5.py`文件，输出结果如下:

```
Checking Predict ...
predictions = [[1 1 0]]
Expected predictions = [[1 1 0]]
```

2.10 模型整合

该小节将上面各个部分的代码进行整合，并使用训练集进行训练，同时使用训练集和测试集数据对训练完成的模型进行测试。

请仔细阅读`model(X_train, Y_train, X_test, Y_test, num_iterations = 2000, learning_rate = 0.5, print_cost = False)`函数，该

部分不需要补全。

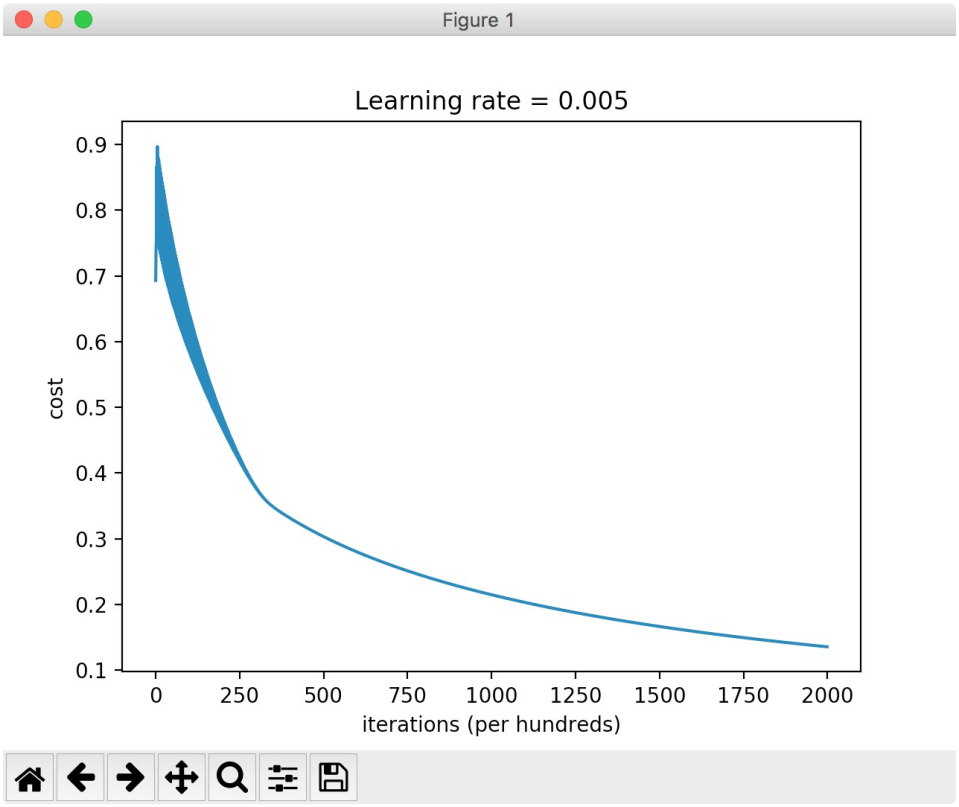
运行结果如下：

```
Runing Model ...
Cost after iteration 0: 0.693147
Cost after iteration 100: 0.584508
Cost after iteration 200: 0.466949
Cost after iteration 300: 0.376007
Cost after iteration 400: 0.331463
Cost after iteration 500: 0.303273
Cost after iteration 600: 0.279880
Cost after iteration 700: 0.260042
Cost after iteration 800: 0.242941
Cost after iteration 900: 0.228004
Cost after iteration 1000: 0.214820
Cost after iteration 1100: 0.203078
Cost after iteration 1200: 0.192544
Cost after iteration 1300: 0.183033
Cost after iteration 1400: 0.174399
Cost after iteration 1500: 0.166521
Cost after iteration 1600: 0.159305
Cost after iteration 1700: 0.152667
Cost after iteration 1800: 0.146542
Cost after iteration 1900: 0.140872
train accuracy: 99.04306220095694 %
test accuracy: 70.0 %
```

可以看出训练集的正确率在99.04%。
而测试集的正确率只有区区70.0%
说明我们现在训练的这个模型已经超调了，泛化能力不是很好。但是对刚刚入门的我们来说已经足够了。

2.11 显示学习曲线

尝试补全`plot(d)`函数，使其显示如下学习曲线，注意标题，坐标轴，和图例(legend)。



2.12 自己找张图片测试下你的模型吧!!

你可以把自己的图片放到images文件夹下，然后将my_image改为对应文件名，测试下我们的模型能否进行正确的分辨。

> 注:该部分不需要补全代码，强烈建议从网上找些图片自行测试。

3.Deadline

- 2017.12.3(下周一)上午12:00截止
- 有不懂得可以随时Google或者找我问

4.提交方法

- 邮件发送到:`[root@oopy.org]`
- 邮件标题:`(姓名全拼)lecture5`
- 作业以邮件附件形式发送