

AI005 基于华为AI平台的机器学习

第二次课中作业

解昃焱 518021911249

jupyter 2_LogisticRegression Last Checkpoint: 9 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run the following cell to train your model.

```
In [40]: d = model(train_set_x, train_set_y, test_set_x, test_set_y, num_iterations = 2000, learning_rate = 0.005, print_cost = True)
```

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: 0.303173
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: 0.159385
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 %

Expected Output

""Train Accuracy""	99.04306220095694 %
""Test Accuracy""	70.0 %

Comment

训练精度接近100%，说明模型训练的不错。测试误差为6%，因为我们使用的数据量很小，而且逻辑回归只是一个线性分类器，所以对于这个简单的模型来说，这个结果已经不算太差了。

如果想做的更好，下面你会学到一个更棒的分类器！

ps: 可以看到，模型显然过拟合了，后面我们可以学习使用正则化等方式来优化。

```
In [41]: # Example of a picture that was wrongly classified.  
index = 9  
plt.imshow(test_set_x[:,index].reshape((num_px, num_px, 3)))  
print ("y = " + str(test_set_y[0,index]) + ", you predicted that it is a '" + classes[int(d["Y_prediction_test"][0,index])].decode("utf-8") + "''  
<  
y = 1, you predicted that it is a "cat" picture.
```

jupyter 2_LogisticRegression Last Checkpoint: 9 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3


```
In [39]: # GRADED FUNCTION: model  
# 测试  
#518021911249  
def model(X_train, Y_train, X_test, Y_test, num_iterations = 2000, learning_rate = 0.5, print_cost = False):  
    """  
    Builds the logistic regression model by calling the function you've implemented previously  
    Arguments:  
    X_train -- training set represented by a numpy array of shape (num_px * num_px * 3, m_train)  
    Y_train -- training labels represented by a numpy array (vector) of shape (1, m_train)  
    X_test -- test set represented by a numpy array of shape (num_px * num_px * 3, m_test)  
    Y_test -- test labels represented by a numpy array (vector) of shape (1, m_test)  
    num_iterations -- hyperparameter representing the number of iterations to optimize the parameters  
    learning_rate -- hyperparameter representing the learning rate used in the update rule of optimize()  
    print_cost -- Set to true to print the cost every 100 iterations  
    Returns:  
    d -- dictionary containing information about the model.  
    """  
    ### START CODE HERE ###  
    # initialize parameters (w and b) with zeros (= 1 line of code)  
    w, b = initialize_with_zeros(X_train.shape[0])  
    # Gradient descent (use optimize function) (= 1 line of code)  
    params, grads, costs = optimize(w, b, X_train, Y_train, num_iterations, learning_rate, print_cost)  
    # Retrieve parameters w and b from dictionary "params"  
    w = params["w"]  
    b = params["b"]  
    # Predict test/train set examples (= 2 lines of code)  
    Y_prediction_train = predict(w, b, X_train)  
    Y_prediction_test = predict(w, b, X_test)  
    ### END CODE HERE ###  
    # Print train/test Errors  
    print("train accuracy: {} %".format(100 - np.mean(np.abs(Y_prediction_train - Y_train)) * 100))  
    print("test accuracy: {} %".format(100 - np.mean(np.abs(Y_prediction_test - Y_test)) * 100))  
    d = {"costs": costs,  
        "Y_prediction_test": Y_prediction_test,  
        "Y_prediction_train": Y_prediction_train,  
        "w": w,  
        "b": b,  
        "learning_rate": learning_rate,  
        "num_iterations": num_iterations}  
    return d
```

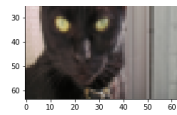
jupyter 2_LogisticRegression Last Checkpoint: 10 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

ps: 可以看到，模型显然过拟合了，后面我们可以学习使用正则化等方式来优化。

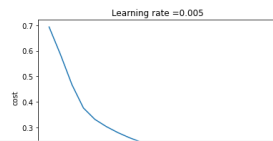
```
In [41]: # Example of a picture that was wrongly classified.  
# 测试  
#518021911249  
index = 9  
plt.imshow(test_set_x[:,index].reshape((num_px, num_px, 3)))  
print ("y = " + str(test_set_y[0,index]) + ", you predicted that it is a '" + classes[int(d["Y_prediction_test"][0,index])].decode("utf-8") + "''  
<  
y = 1, you predicted that it is a "cat" picture.
```





我们也绘制一下损失函数和梯度。

```
In [42]: #解闭卷
#518021911249
# Plot Learning curve (with costs)
costs = np.squeeze(d['costs'])
plt.plot(costs)
plt.ylabel('cost')
plt.xlabel('iterations (per hundreds)')
plt.title("Learning rate =" + str(d["learning_rate"]))
plt.show()
```



浏览器标签: numpy 中国知网 知网-中国知网 同德新报城下 全文阅读-XML全文 从“恒安文化”看新 全文阅读-XML全文 jupyter notebook 2_LogisticRegression

localhost:8888/notebooks/jupyter%20notebook/【作业】Numpy_LR_1021/【作业】Numpy_LR练习/2_LogisticRegression.ipynb

jupyter 2_LogisticRegression Last Checkpoint: 10 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

我们也绘制一下损失函数和梯度。

```
In [42]: #解闭卷
#518021911249
# Plot Learning curve (with costs)
costs = np.squeeze(d['costs'])
plt.plot(costs)
plt.ylabel('cost')
plt.xlabel('iterations (per hundreds)')
plt.title("Learning rate =" + str(d["learning_rate"]))
plt.show()
```

6 - 进一步分析 (选做)

我们来探索一下更多的学习率 α 的可能性

Reminder 为了使梯度下降起作用, 你必须明智地选择学习率 α 。学习率决定了我们更新参数的速度。如果学习率太大, 我们可能会“超调”最佳值。如果太小, 我们将需要太多的迭代来收敛。

下面, 我们尝试使用不同的学习率。比较模型的学习曲线, 你可以自己调整 learning_rates 然后看看会发生什么。

```
In [43]: #解闭卷
#518021911249
learning_rates = [0.01, 0.005, 0.001, 0.0005, 0.0001]
models = {}
for i in learning_rates:
    print("learning rate is: " + str(i))
    models[str(i)] = model(train_set_x, train_set_y, test_set_x, test_set_y, num_iterations = 1500, learning_rate = i, print_cost = True)
    print("\n" + "-" * 50 + "\n")
```

屏幕快照已保存
屏幕快照已添加到你的 OneDrive。
OneDrive (1)

6 - 进一步分析 (选做)

我们来探索一下更多的学习率 α 的可能性

Reminder. 为了使梯度下降起作用, 你必须明智地选择学习率 α , 学习率决定了我们更新参数的速度。

如果学习率太大, 我们可能会“超调”最佳值。如果太小, 我们将需要太多的迭代来收敛。

下面, 我们尝试使用不同的学习率, 比较模型的学习曲线, 你可以自己调整 `learning_rates` 然后看看会发生什么。

```
In [43]: # 解答题
#518021911249
learning_rates = [0.01, 0.005, 0.001, 0.0005, 0.0001]
models = {}
for i in learning_rates:
    print("learning rate is: " + str(i))
    models[str(i)] = model(train_set_X, train_set_y, test_set_y, num_iterations = 1500, learning_rate = i, print_cost=True)
    print("\n" + "-----" + "\n")

for i in learning_rates:
    plt.plot(np.squeeze(models[str(i)]["costs"]), label= str(models[str(i)]["learning_rate"]))

plt.ylabel('cost')
plt.xlabel('iterations')

legend = plt.legend(loc='upper center', shadow=True)
frame.set_frame()
frame.set_facecolor(0.90)
plt.show()
```

learning rate is: 0.01
train accuracy: 99.52153110047847 %
test accuracy: 68.0 %

learning rate is: 0.005
train accuracy: 97.60765550239235 %
test accuracy: 70.0 %

learning rate is: 0.001
train accuracy: 88.99521531100478 %
test accuracy: 64.0 %

learning rate is: 0.0005
train accuracy: 88.99521531100478 %
test accuracy: 64.0 %

learning rate is: 0.0001
train accuracy: 88.99521531100478 %
test accuracy: 64.0 %

7 - 使用自己的图像来测试

```
In [44]: #解答题
#518021911249

## START CODE HERE ## (PUT YOUR IMAGE NAME)
my_image = "cat_in_1ran.jpg" # change this to the name of your image file
## END CODE HERE ##

# We preprocess the image to fit your algorithm.
fname = "images/" + my_image
image = np.array(ndimage.imread(fname, flatten=False))
my_image = scipy.misc.imresize(image, size=(num_px,num_px)).reshape((1, num_px*num_px*3)).T
my_predicted_image = predict(d["w"], d["b"], my_image)

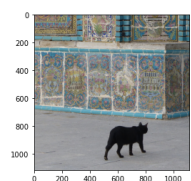
plt.imshow(image)
print("y = " + str(np.squeeze(my_predicted_image)) + ", your algorithm predicts a \\" + classes[int(np.squeeze(my_predicted_image))])
```

D:\Viy_app\anaconda\lib\site-packages\ipykernel_launcher.py:7: DeprecationWarning: 'imread' is deprecated!
'imread' is deprecated in SciPy 1.0.0.
Use 'matplotlib.pyplot.imread' instead.

import sys

D:\Viy_app\anaconda\lib\site-packages\ipykernel_launcher.py:8: DeprecationWarning: 'imresize' is deprecated!
'imresize' is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use 'skimage.transform.resize' instead.

y = 1.0, your algorithm predicts a "cat" picture.



Bibliography:

- <http://www.wildml.com/2015/09/implementing-a-neural-network-from-scratch/>
- <https://stats.stackexchange.com/questions/211436/why-do-we-normalize-images-by-subtracting-the-datasets-image-mean-and-not-the-c>