fresh七天

关注 发私信

访问： 20828次

积分： 599

等级： BLOG 3

排名： 千里之外

神经网络程序设计课学习心得

标签： 神经网络 机器学习 血常规 flask web

2016-12-26 20:25 437人阅读 评论(0) 收藏 举报

分类： deeplearn（2） python（2） Algorithm（2）

目录(?) [+]

USTC-NP2016课程学习

- USTC-NP2016课程学习
  - 课程目标

课程目标

本课程的目标是通过学习神经网络和**深度学习**等**机器学习算法**来搭建一个完整的血常规检测报告单的年龄和性别预测系统。项目的最后效果就是，用户上传一张血常规报告单的图片，后台首先进行OCR识别出图片中的项目，将其存入**MongoDB**，然后会根据机器学习算法生成的模型对用户数据进行预测。

项目地址及安装

1. 我自己fork分支地址，有详细的安装说明；
2. 课程地址，有详细的实验要求；
3. 主项目地址，里面包含了其他平台机器学习算法和很多其他同学的贡献；

项目演示

文章存档

展开

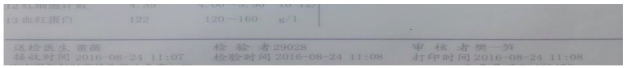
阅读排行

演示过程如下：

1. 系统首页

2. 选择一个血常规图片，提交上传报告后

3. 点击生成报告，会得到一个OCR的识别报告单

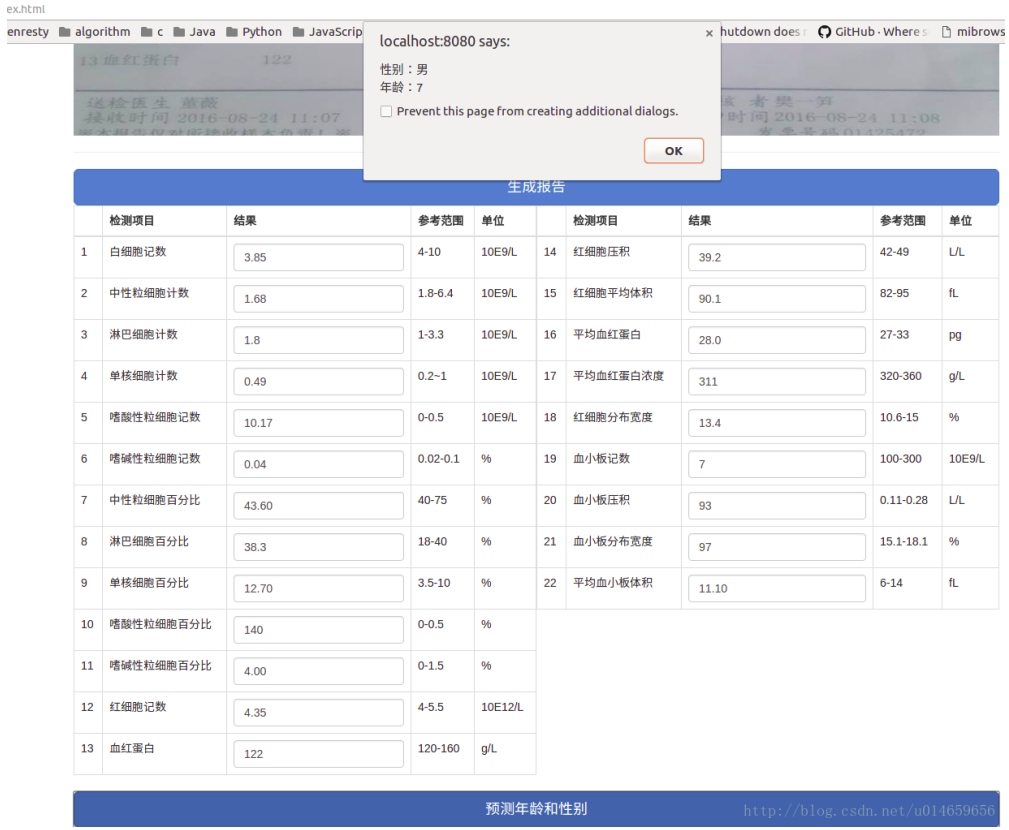


生成报告

	检测项目	结果	参考范围	单位		检测项目	结果	参考范围	单位
1	白细胞记数	3.86	4-10	10E9/L	14	红细胞压积	39.2	42-49	L/L
2	中性粒细胞计数	1.68	1.8-6.4	10E9/L	15	红细胞平均体积	90.1	82-95	fL
3	淋巴细胞计数	1.48	1-3.3	10E9/L	16	平均血红蛋白	28.0	27-33	pg
4	单核细胞计数	0.49	0.2~1	10E9/L	17	平均血红蛋白浓度	311	320-360	g/L
5	嗜酸性粒细胞记数	0.17	0-0.5	10E9/L	18	红细胞分布宽度	13.4	10.6-15	%
6	嗜碱性粒细胞记数	0.04	0.02-0.1	%	19	血小板记数	171	100-300	10E9/L
7	中性粒细胞百分比	43.60	40-75	%	20	血小板压积	0.20	0.11-0.28	L/L
8	淋巴细胞百分比	33	18-40	%	21	血小板分布宽度	13.7	15.1-18.1	%
9	单核细胞百分比	12.70	3.5-10	%	22	平均血小板体积	11.70	6-14	fL
10	嗜酸性粒细胞百分比	4.40	0-0.5	%					
11	嗜碱性粒细胞百分比	1.00	0-1.5	%					
12	红细胞记数	4.35	4-5.5	10E12/L					
13	血红蛋白	122	120-160	g/L					

预测年龄和性别

4. 确认数据无误后，点击预测



	检测项目	结果	参考范围	单位		检测项目	结果	参考范围	单位
1	白细胞记数	3.85	4-10	10E9/L	14	红细胞压积	39.2	42-49	L/L
2	中性粒细胞计数	1.68	1.8-6.4	10E9/L	15	红细胞平均体积	90.1	82-95	fL
3	淋巴细胞计数	1.8	1-3.3	10E9/L	16	平均血红蛋白	28.0	27-33	pg
4	单核细胞计数	0.49	0.2~1	10E9/L	17	平均血红蛋白浓度	311	320-360	g/L
5	嗜酸性粒细胞记数	10.17	0-0.5	10E9/L	18	红细胞分布宽度	13.4	10.6-15	%
6	嗜碱性粒细胞记数	0.04	0.02-0.1	%	19	血小板记数	7	100-300	10E9/L
7	中性粒细胞百分比	43.60	40-75	%	20	血小板压积	93	0.11-0.28	L/L
8	淋巴细胞百分比	38.3	18-40	%	21	血小板分布宽度	97	15.1-18.1	%
9	单核细胞百分比	12.70	3.5-10	%	22	平均血小板体积	11.10	6-14	fL
10	嗜酸性粒细胞百分比	140	0-0.5	%					
11	嗜碱性粒细胞百分比	4.00	0-1.5	%					
12	红细胞记数	4.35	4-5.5	10E12/L					
13	血红蛋白	122	120-160	g/L					

项目模块分析

本项目分两大部分，前端展示和后台OCR及预测；三大模块，web模块，图像OCR模块，学习预测模块。处理模块主要是这位同学的贡献。我的pr主要有两个:

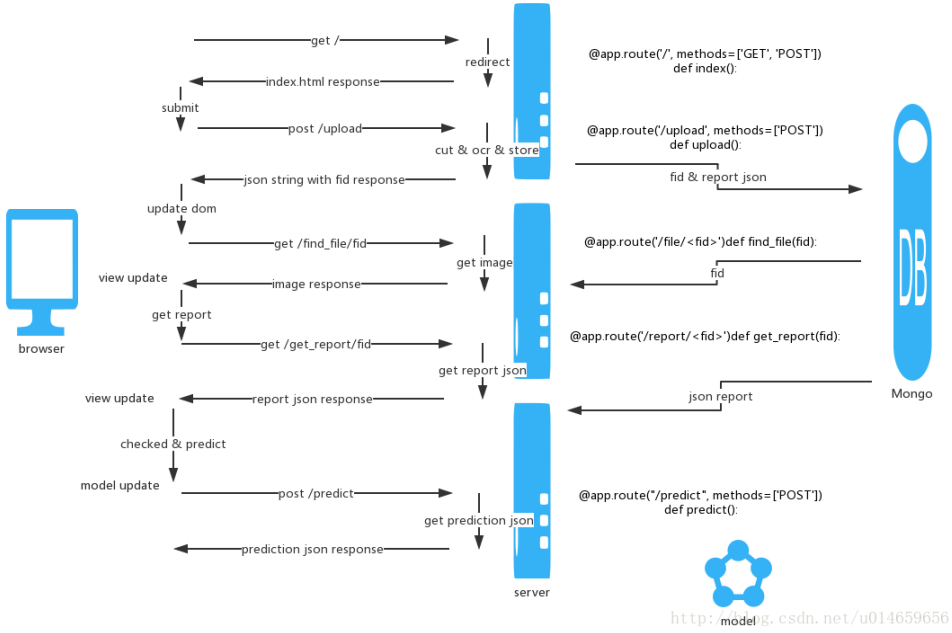
1. 搭建了wep app， 完成了图片上传并存入Mongodb；

2. 封装了图像OCR模块成ImageFilter，规定适当的接口，使其便于模块交互，并完成了图片识别的前后台；这

3. 在自己的分支上完成了自己实现的神经网络的整合，使其成为一个完整可用的系统；

web模块

之前没用过web框架，发现**Python**的web框架有很多，包括著名的全能型Web框架Django；知乎后台的异步框架Tornado；还有小巧积木型的Flask；最后决定采用Flask来搭建，因为这是一个简单的web系统，而且Flask非常易于上手；前端采用Bootstrap，**jQuery**, Vue.js;这一部分虽然简单，但是也学习了不少东西，包括前端和后台的，这里记录了学习过程中部分笔记，这对产品和系统驱动的我很有帮助，因为只有完整弄出一个可用的系统成就感才高。这一部分就不做详细介绍了，最后web部分主要是json传输数据，模仿并实现了简单的REST**架构**；下面是我自己做的整个系统的架构图：



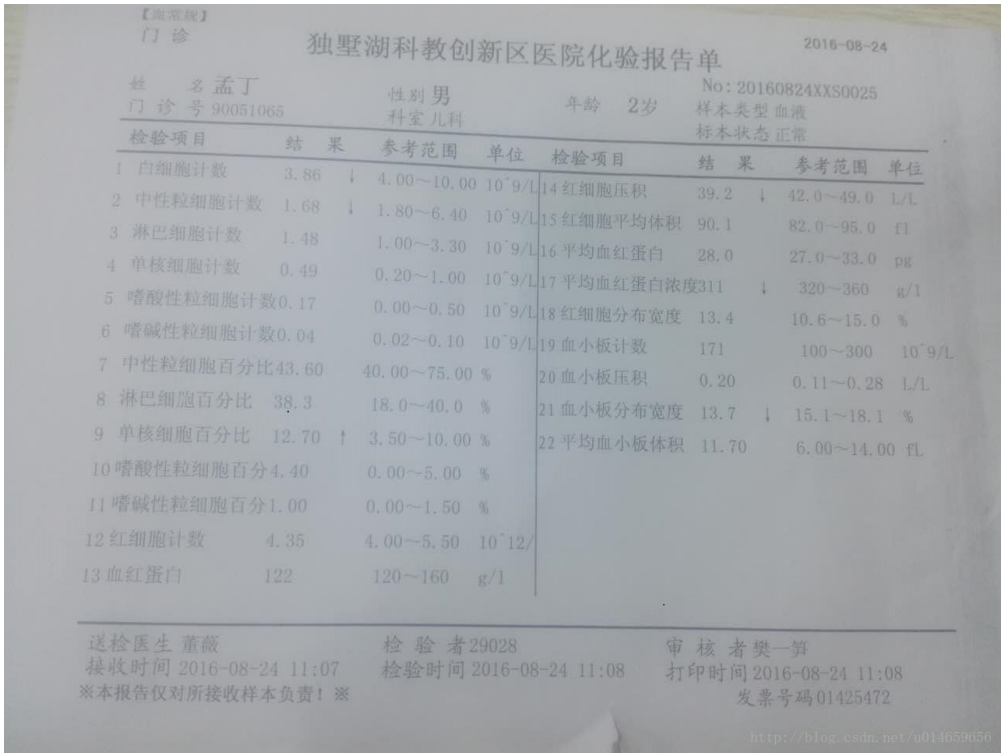
图像OCR模块

这一部分也是有挑战性的一部分，不过本人并没有贡献什么代码，对图像处理领域比较小白。这一部分内
学习到的图像处理的理解；

图像处理应该是计算机学科一个大分支了，加上最近**人工智能**炒的厉害，可以说视觉是机器非常重要的一
是传统的思路，即根据图像本身的特点以及几何学做特征提取，识别和预处理等，比如滤波，傅里叶变换，边
学习的思路了，利用**大数据**集进行有监督或者无监督的学习识别；需要注意的是两种思路并不独立互斥，前者
是否使用机器学习算法，肯定都包含前面一部分内容，否则是不可能学习出好的效果的。

目前我们的OCR主要是第一种思路，即根据图像的几何学特征，首先对图像进行裁剪和分割，然后分别对

我们的血常规报告图片大概是这样的表格：



独墅湖科教创新区医院化验报告单 2016-08-24

姓 名 孟丁 性别 男 年龄 2岁 No: 20160824XXS0025
门 诊 号 90051065 科室 儿科 样本类型 血液
标本状态 正常

检验项目	结 果		参考范围	单位	检验项目	结 果		参考范围	单位
1 白细胞计数	3.86	↓	4.00~10.00	10^9/L	14 红细胞压积	39.2	↓	42.0~49.0	L/L
2 中性粒细胞计数	1.68	↓	1.80~6.40	10^9/L	15 红细胞平均体积	90.1		82.0~95.0	fl
3 淋巴细胞计数	1.48		1.00~3.30	10^9/L	16 平均血红蛋白	28.0		27.0~33.0	pg
4 单核细胞计数	0.49		0.20~1.00	10^9/L	17 平均血红蛋白浓度	311	↓	320~360	g/l
5 嗜酸性粒细胞计数	0.17		0.00~0.50	10^9/L	18 红细胞分布宽度	13.4		10.6~15.0	%
6 嗜碱性粒细胞计数	0.04		0.02~0.10	10^9/L	19 血小板计数	171		100~300	10^9/L
7 中性粒细胞百分比	43.60		40.00~75.00	%	20 血小板压积	0.20		0.11~0.28	L/L
8 淋巴细胞百分比	38.3		18.0~40.0	%	21 血小板分布宽度	13.7	↓	15.1~18.1	%
9 单核细胞百分比	12.70	↑	3.50~10.00	%	22 平均血小板体积	11.70		6.00~14.00	fL
10 嗜酸性粒细胞百分	4.40		0.00~5.00	%					
11 嗜碱性粒细胞百分	1.00		0.00~1.50	%					
12 红细胞计数	4.35		4.00~5.50	10^12/					
13 血红蛋白	122		120~160	g/l					

送检医生 董薇 检 验 者29028 审 核 者樊一算
接收时间 2016-08-24 11:07 检验时间 2016-08-24 11:08 打印时间 2016-08-24 11:08
※本报告仅对所接收样本负责！※ 发票号码01425472

要想抽取出里面的每一个项目，首先是透视变换，就是把带角度拍的图片，变换到正面的坐标系之下，然后对其进行裁剪，思路就是首先锁定三条粗黑线，然后根据这三条粗线裁剪出统一大小的报告区域，而这个统一大小是固定的，只要报告一致，就可以采用固定的大小剪出每一个项目，虽然很不通用，但是至少实现了裁剪了。而后我们会对裁剪出的22项目进行ocr，这一部分主要使用了图像心态学，二值化和膨胀等手段，使得每一个小区域内的文字间隔开一点，然后就更容易抽取出文字了，文字抽取部分就是采用了tesseract；为了提高文字识别率，我们还i医学术语词典，这样就可以缩小搜索空间，从而增大识别率。

学习预测模块

神经网络介绍

该部分不做说明，网上资源很多，附上自己的presentation;里面有详细的公式推导和从头用C语言实现的一代码，包括矩阵运算和一些函数回调，深刻体会到手动申请和释放内存的麻烦，带有GC的语言会方便很多，

训练数据集

数据集格式如下，训练集有1858个，**测试**集有200个：

```
id,sex,age,WBC,RBC,HGB,HCT,MCV,MCH,MCHC,RDW,PLT,MPV,PCT,PDW,LYM,LYM%,MONO,MONO%,NEU,NEU%,EOS,EOS%,BAS,BAS%,ALY,ALY%,
1,男,6,5.2,7.6,0.176,12.2,2.79,53.6,0.7,13.5,1.41,27.8,0.05,4.93,0.1,0.08,1.6,0.11,2.2,0.06,1.2,138,0.409,83,28,337,11.8,233
2,男,8,11.2,7.7,0.235,12.2,2.47,22.1,1.1,9.8,7.47,66.7,0.08,4.62,0.7,0.08,0.7,0.09,0.8,0.23,2.1,127,0.376,81,27.5,338,11.6,306
3,女,9,15,6.8,0.292,9.5,5.15,34.4,1.29,8.6,8.11,54.2,0.27,4.41,1.8,0.15,1,0.17,1.1,0.36,2.4,121,0.348,79,27.5,348,8.5,431
4,男,9,8.9,7.2,0.225,9.2,2.84,31.8,1.09,12.2,4.88,54.7,0.06,4.12,0.7,0.05,0.6,0.06,0.7,0.18,2.1,121,0.355,86,29.3,340,10,314
5,男,10,3.7,7.3,0.271,11,1.47,39.5,0.34,9.1,1.78,47.8,0.11,5.06,3,0.02,0.6,0.03,0.7,0.02,0.6,139,0.417,82,27.6,335,12.4,371
```

数据的预处理,主要尝试了Min-Max和Zero-Score两种方法进行归一化，效果会比不进行归一化要好一点，主要是收敛速度变快了。这里需要注意的是，归一化的意义，我开始搞错了，我对每一个样本即每一行做归一化，但是这是没有意义的，因为每一行的数据是不同的项目和性质，不能这样归一化，而是对每一列做，比如Min-Max是对所有样本的同一项即同一列的最大最小。

Min-Max;

$$x' = \frac{x - Min}{Max - Min}$$

```
# 对每一列做min-max归一化
def min_max_norm(self, matrix, axis=0):
    mined = matrix.min(axis=axis)
    print mined.shape
    maxed = matrix.max(axis=axis)
    print maxed.shape
    print matrix.shape
    normed = (matrix - mined.reshape(1,mined.shape[0])) / (maxed - mined).reshape(1,mined.shap
    return normed
```

Zero-Score即高斯分布归一化，μ是均值，σ是标准差;

$$x' = \frac{x - \mu}{\sigma}$$

```
 # 由于数据具有不同的量纲，因此对数据做归一化预处理,是对每一项指标即每一列做归一，而不是
# 对每一列采用正态分布归一化
def normalization(self, matrix):
  for i in range(matrix.shape[1]):
```


```
    mean = np.mean(matrix[:,i])
    s2 = np.sum((matrix[:,i]-mean)**2)/matrix.shape[0]
    sigma = math.sqrt(s2)
    matrix[:,i] = (matrix[:,i] - mean)/sigma
return matrix
```

这里有一个技巧就是，在运用矩阵运算的时候，尽量不要把Python List 和 numpy array混用，因为Python循环比numpy慢很多！尽量调用numpy的库方法。

另外，对年龄的训练我采用了划分区间的方法，因为数据不全，分的太细没意义。最后训练年龄的结果就是最高只到23%,但是如果是按照计算方法是预测出来的年龄和实际值相差5的话，正确率应该是30%左右。

模型

age的模型，四层网络结构，tuning到大概hidden1=10, hidden2=25，learning_rate=0.1比较合适，这里有一个经验就是learning_rate是只会影响学习的步长即速率，学习率太小，收敛的速度会很慢，太大可能会直接跳过收敛点，甚至有的学习率会导致来回震荡。一般的学习率大概就是0.001-1之间吧，而且学习率并不会影响模型的准确度峰值。只有网络结构以及输入数据和误差函数会影响峰值。 ：

```
# -*- coding: utf8 -*-
import tensorflow as tf
import numpy as np
import math
from databuilder import Dataset
class ModelGender(object):
    def __init__(self, input=26, hidden1=10, hidden2=25, output=2, learning_rate=0.05, dataset=None):
        self.restored = False
        # 初始化数据集
        self.dataset = dataset
        train_path = 'data/train.csv'
        readcols = [i for i in range(29) if i not in (0,1,2)]
        self.train_data_matrix = np.loadtxt(open(train_path,'rb'), delimiter=',',
                    skiprows=1, usecols=readcols)
        self.x = tf.placeholder(tf.float32, [None, input])
        self.W1 = tf.Variable(tf.truncated_normal([input,hidden1], stddev=1.0 / math.sqrt(float(input)
        self.b1 = tf.Variable(tf.zeros([hidden1]), name="gender_b1")
        self.h1 = tf.nn.relu(tf.matmul(self.x,self.W1) + self.b1)

        self.W2 = tf.Variable(tf.truncated_normal([hidden1, hidden2], stddev=1.0 / math.sqrt(float(hi
```



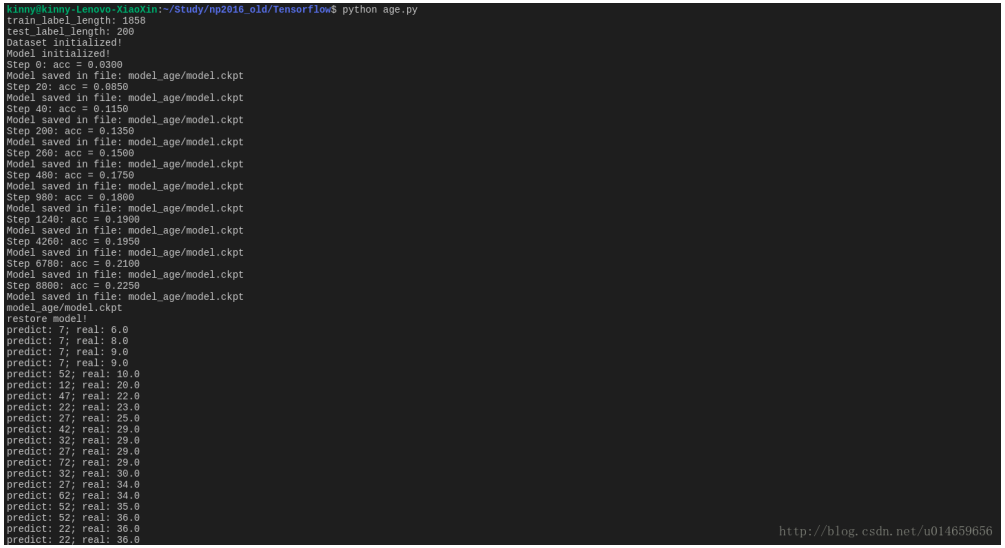
```
        self.b2 = tf.Variable(tf.zeros([hidden2]), name="gender_b2")
        self.h2 = tf.nn.relu(tf.matmul(self.h1,self.W2) + self.b2)

        self.W3 = tf.Variable(tf.truncated_normal([hidden2, output], stddev=1.0 / math.sqrt(float(hidden2))), name="gender_w3")
        self.b3 = tf.Variable(tf.zeros([output]), name="gender_b3")
        self.y = tf.nn.relu(tf.matmul(self.h2,self.W3) + self.b3)
        self.y_  = tf.placeholder(tf.float32, [None, output])
        self.cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(self.y, self.y_))
        self.train_step  = tf.train.GradientDescentOptimizer(learning_rate).minimize(self.cross_entropy)
        self.init = tf.global_variables_initializer()
        self.saver = tf.train.Saver([self.W1, self.b1, self.W2, self.b2, self.W3, self.b3])
        # mnist_softmax.py 中，使用的是在sess中通过run方法执行train_step, accuracy
        # mnist_cnn.py中，使用的是直接执行train_step, accuracy.eval，所以必须要传入session参数
        self.sess = tf.Session()
        self.sess.run(self.init)
        print("Gender Model initialized!")
    def train(self, max_step=10000, batch_size=10):
        max = 0

        for i in range(max_step):
            batch_xs, batch_ys = self.dataset.next_gender_train_batch(batch_size)
            self.sess.run(self.train_step, feed_dict={self.x: batch_xs, self.y_: batch_ys})

            step = i
            if i % 20 == 0:
                self.correct_prediction = tf.equal(tf.argmax(self.y, 1), tf.argmax(self.y_, 1))
                self.accuracy = tf.reduce_mean(tf.cast(self.correct_prediction, tf.float32))
                acc = self.sess.run(self.accuracy, feed_dict={self.x: self.dataset.test_gender_data_matrix, self.y_: self.dataset.test_gender_label_matrix
                if acc > max:
                    max = acc
                    print('Step %d: acc = %.4f' % (step, acc))
                    #Save the variables to disk.
                    save_path = self.saver.save(self.sess, "model_gender/model.ckpt")
                    print("Gender Model saved in file: %s" % save_path)

    def predict(self,datas):
        #   这里必须使用np.reshape,不能使用tf.reshape,后者用于tensor，tensor是用来填充的
        datas = self.norm_with_new(self.train_data_matrix, datas)
        datas = np.reshape(datas, (1, 26))
        ckpt = tf.train.get_checkpoint_state("model_gender/")

        if self.restored == False:
            if ckpt and ckpt.model_checkpoint_path:
```

```
64              self.saver.restore(self.sess, ckpt.model_checkpoint_path)
65              self.restored = True
66              print("restore gender model!")
67            else:
68              print("No gender model checkpoint found!")
69
70        predictions = self.sess.run(self.y, feed_dict={self.x: datas})
71        return predictions
72
73      def norm_with_new(self, origin, new):
74        # 添加一行新的
75        matrix = np.row_stack((origin,new))
76        for i in range(matrix.shape[1]):
77          mean = np.mean(matrix[:,i])
78          s2 = np.sum((matrix[:,i]-mean)**2)/matrix.shape[0]
79          sigma = math.sqrt(s2)
80          matrix[:,i] = (matrix[:,i] - mean)/sigma
81        return matrix[-1,:]
82    class ModelAge(object):
83      def __init__(self, input=26, hidden1=10, hidden2=25, output=20, learning_rate=0.1, dataset=None):
84        self.restored = False
85        # 初始化数据集
86        self.dataset = dataset
87        train_path = 'data/train.csv'
88        readcols = [i for i in range(29) if i not in (0,1,2)]
89        self.train_data_matrix = np.loadtxt(open(train_path,'rb'), delimiter=',',
90                        skiprows=1, usecols=readcols)
91        self.x = tf.placeholder(tf.float32, [None, input])
92        self.W1 = tf.Variable(tf.truncated_normal([input,hidden1], stddev=1.0 / math.sqrt(float(input))), name="age_w1")
93        self.b1 = tf.Variable(tf.zeros([hidden1]), name="age_b1")
94        self.h1 = tf.nn.relu(tf.matmul(self.x,self.W1) + self.b1)
95
96        self.W2 = tf.Variable(tf.truncated_normal([hidden1, hidden2], stddev=1.0 / math.sqrt(float(hi
97        self.b2 = tf.Variable(tf.zeros([hidden2]), name="age_b2")
98        self.h2 = tf.nn.relu(tf.matmul(self.h1,self.W2) + self.b2)
99
100       self.W3 = tf.Variable(tf.truncated_normal([hidden2, output], stddev=1.0 / math.sqrt(float(hidd
101       self.b3 = tf.Variable(tf.zeros([output]), name="age_b3")
102       self.y = tf.nn.relu(tf.matmul(self.h2,self.W3) + self.b3)
103       self.y_  = tf.placeholder(tf.float32, [None, output])
104       self.cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(self.y, self.y_))
105       self.train_step  = tf.train.GradientDescentOptimizer(learning_rate).minimize(self.cross_entrop
106       self.init = tf.global_variables_initializer()
```

```
107         self.saver = tf.train.Saver([self.W1, self.b1, self.W2, self.b2, self.W3, self.b3])
108         # mnist_softmax.py 中，使用的是在sess中通过run方法执行train_step, accuracy
109         # mnist_cnn.py中，使用的是直接执行train_step, accuracy.eval，所以必须要传入session参数
110         self.sess = tf.Session()
111         self.sess.run(self.init)
112         print("Age Model initialized!")
113     def train(self, max_step=20000, batch_size=10):
114         max = 0
115 
116         for i in range(max_step):
117             batch_xs, batch_ys = self.dataset.next_age_train_batch(batch_size)
118             self.sess.run(self.train_step, feed_dict={self.x: batch_xs, self.y_: batch_ys})
119 
120             step = i
121             if i % 20 == 0:
122                 self.correct_prediction = tf.equal(tf.argmax(self.y, 1), tf.argmax(self.y_, 1))
123                 self.accuracy = tf.reduce_mean(tf.cast(self.correct_prediction, tf.float32))
124                 acc = self.sess.run(self.accuracy, feed_dict={self.x: self.dataset.test_age_data_matrix, self.y_: self.dataset.test_age_la
125                 if acc > max:
126                     max = acc
127                     print('Step %d: acc = %.4f' % (step, acc))
128                     #Save the variables to disk.
129                     save_path = self.saver.save(self.sess, "model_age/model.ckpt")
130                     print("Age Model saved in file: %s" % save_path)
131     def predict(self,datas):
132         #   先对 datas 做归一化
133         datas = self.norm_with_new(self.train_data_matrix, datas)
134         datas = np.reshape(datas, (1, 26))
135         ckpt = tf.train.get_checkpoint_state("model_age/")
136         '''
137         new_saver = tf.train.import_meta_graph("model_age/model.ckpt.meta")
138         new_saver.restore(self.sess, tf.train.latest_checkpoint('model_age/'))
139         all_vars = tf.trainable_variables()
140         for v in all_vars:
141             print(v.name)
142         '''
143         if self.restored == False:
144             if ckpt and ckpt.model_checkpoint_path:
145                 self.saver.restore(self.sess, ckpt.model_checkpoint_path)
146                 self.restored = True
147                 print("restore age model!")
148             else:
149                 print("No age model checkpoint found!")
```

```
150
151         predictions = self.sess.run(self.y, feed_dict={self.x: datas})
152         return predictions
153     def norm_with_new(self, origin, new):
154         # 添加一行新的
155         matrix = np.row_stack((origin,new))
156         for i in range(matrix.shape[1]):
157             mean = np.mean(matrix[:,i])
158             s2 = np.sum((matrix[:,i]-mean)**2)/matrix.shape[0]
159             sigma = math.sqrt(s2)
160             matrix[:,i] = (matrix[:,i] - mean)/sigma
161         return matrix[-1,:]
```

效果如下，大概看了一下觉得貌似年轻人预测会准一点，难道血常规在年轻的年龄段和年龄相关性更大？：



gender的模型类似。min-max效果比zero-score差很多，min-max是52%，zero-score准确率最高75%左右。G

效果如下：

```
kinny@kinny-Lenovo-XiaoXin:~/Study/np2016_old/Tensorflow$ python gender.py
train_label_length: 1858
test_label_length: 200
Dataset initialized!
Model initialized!
Step 0: acc = 0.4850
Model saved in file: model_gender/model.ckpt
Step 640: acc = 0.4900
Model saved in file: model_gender/model.ckpt
Step 740: acc = 0.4950
Model saved in file: model_gender/model.ckpt
Step 1120: acc = 0.5050
Model saved in file: model_gender/model.ckpt
Step 1320: acc = 0.5250
Model saved in file: model_gender/model.ckpt
Step 1340: acc = 0.5950
Model saved in file: model_gender/model.ckpt
Step 1360: acc = 0.6700
Model saved in file: model_gender/model.ckpt
Step 1400: acc = 0.6850
Model saved in file: model_gender/model.ckpt
Step 1420: acc = 0.6900
Model saved in file: model_gender/model.ckpt
Step 1460: acc = 0.6950
Model saved in file: model_gender/model.ckpt
Step 1620: acc = 0.7100
Model saved in file: model_gender/model.ckpt
Step 2060: acc = 0.7150
Model saved in file: model_gender/model.ckpt
Step 2200: acc = 0.7200
Model saved in file: model_gender/model.ckpt
restore model!
predict: 女; real: 男
predict: 女; real: 男
predict: 男; real: 女
predict: 女; real: 男
predict: 女; real: 男
predict: 女; real: 女
predict: 女; real: 女
predict: 男; real: 男
predict: 男; real: 女
predict: 女; real: 女
predict: 女; real: 女
predict: 女; real: 女
predict: 男; real: 男
predict: 女; real: 女
predict: 女; real: 女
predict: 女; real: 女
predict: 男; real: 女
predict: 女; real: 女
predict: 男; real: 男
predict: 女; real: 女
predict: 女; real: 男
predict: 女; real: 女
predict: 女; real: 女
predict: 男; real: 男
predict: 男; real: 女
predict: 女; real: 女
                                                    http://blog.csdn.net/u014659656
```

整个系统整合

这部分也是搭积木的过程了，各个模块大概都做好了，我对age和gender做了简单的封装，命名为model.py。该模块实现了对年龄和性别的预测及训练，模块包括 ModelGender 和 ModelAge 类，分别含有 train 和 predict 方法。依赖 databuilder 模块做数据处理和

```
data = DataSet()
nnGender = ModelGender(dataset=data)
nnAge = ModelAge(dataset=data)
nnGender.train()
nnAge.train()
nnGender.predict(predict_data)
nnAge.predict(predict_data)
```

由于报告单数据只有22项，因此这里我把缺失项都补了训练数据集的均值。

```python
@app.route("/predict", methods=['POST'])
def predict():
    print ("predict now!")
    train_class_set = ['WBC','RBC','HGB','HCT','MCV','MCH','MCHC','RDW',
                       'PLT','MPV','PCT','PDW','LYM','LYM%','MONO','MONO%',
                       'NEU','NEU%','EOS','EOS%','BAS','BAS%','ALY','ALY%','LIC','LIC%']
    user_class_set = []
    reports = request.json["checkedReport"]
    #print reports
    for report in reports:
        user_class_set.append({report['alias']: float(report['value'])})

    #print train_class_set
    #print user_class_set
    predict_data = []
    flag = False
    for item_name in train_class_set:
        flag = False
        for user_item in user_class_set:
            if item_name in user_item:
                predict_data.append(user_item[item_name])
                flag = True
                break
            else:
                continue
        if flag is False:
            # 不存在的数据补训练样本的均值
            mean = np.mean(nnAge.train_data_matrix[:,train_class_set.index(item_name)])
            predict_data.append(mean)
    print predict_data

    predict_data = np.array(predict_data)
    predictions = nnGender.predict(predict_data)
    gender = '男' if np.argmax(predictions) == 0  else "女"
    predictions = nnAge.predict(predict_data)
    ageInterval = np.argmax(predictions)
    age = (ageInterval * 5 + ageInterval * 5 + 5) / 2
    result = {
        "gender":gender,
        "age":int(age)
    }

    return json.dumps(result)
```

课程心得体会

这学期基本上其他课都是废课，既浪费时间也学不到东西，除了实验认真做了一下，其他的都是浪费时间。只能说倒霉，只剩下一堆烂课可以选，当然网络程序设计是我自己选上的。对孟老师早有耳闻，我很欣赏这种课程教学方式，不是老师读PPT的传统式教学。虽然老师没有教学具体的内容，但是老师给予学生充分的自主权！和我们一起学习！引导我们如何做工程，团队开发！在老师的带领下，每一个学生都贡献了自己的一部分代码，共同完成了这个项目。这是在本科绝对没有的，本科都是自己摸索，老师根本不会告诉你Git团队协作，除了不知道从哪儿搞来的狗屁教科书！

首先这门课，的确是比较难的！因为不是完全工程化的项目，还需要学习一些机器学习算法。光搞懂算法就很难了，而且还要做出demo！因为国内教学基本上是两头偏，要么把计算机学成了和数学一样纯理论的东西，要么就是搞成了培训课，就是教你这样做不教你为什么。我想说，计算机绝对是最理实交融的一门科学。虽然表面上看，它是一门偏实践的科学！

其次，课程也和实际开发相结合，根据开发进度不断调整策略，分配任务，此时老师就像一个产品经理，帮助我们分析需求。另外，编写开发文档也是很重要的，虽然写文档很累，但是一定要写，不仅是对自己，也是对别人有好处。我之所以放弃paddlepaddle就是因为我觉得百度的工程师是不是都不喜欢也不认真写文档，文档写的真的不如tensorflow细心仔细，这可能真的是国内和国外的差距所在，国外的文档写的真不是一般的好，超过[illegible]客，所以回馈他们的方式就是尽自己最大的努力写出更好的文档。这一点在教科书上面体现也很明显，明显差距不知道拉了几条街，当然[illegible]时间有关。最后，为什么要写开发文档，我想从四种境界的程序员来谈谈。第一种境界就是天赋异禀，同时还注重代码的可维护性和可读性！一个算法，效率极高，而且模块清晰，接口合理，可扩展性极好，这是真正的大神！第二种境界，就是众人眼中大神，天赋异禀，一个算法一行代码搞定，效率惊人，但是很少有人能搞懂到底写的是什么，对于资质平庸的程序员来说，维护他的代码是一种痛苦；第三种境界的程序员，积极上进，但资历稍差，算法要想实现得得费一下力气，但是一定尽量想清楚，不断的拆解模块，推敲适当的接口，最后呈现一个具备可读性和可维护性的代码。最后一种，资质平庸，写的代码也是混乱堆砌，不知所云，效率也不高。我相信国内大部分程序员都是第三第四中，包括我自己，我正[illegible]跳过了，毕竟没有那么高的天赋。因此，现阶段，写文档绝对是保持代码可读性和可维护性的手段之一！

最后，自己以前只使用过git的个人工作功能，现在学会了团队协作，pull request，以后有余力也可以展开[illegible]己的力量了！还有就是，课堂上其他同学也分享了许多除了神经网络以外机器学习算法和平台，虽然没有听懂[illegible]学习了！团队协作和分享交流也是重要的一部分，以后还是得多学习学习。目前这方面可能做的不太好，分享[illegible]人讲懂。

顶 1 踩 0

- 上一篇 Ubuntu16.04 配置tensorflow gpu版本
- 下一篇 tensorflow模型参数保存和加载问题

我的同类文章

deeplearn（2） python（2） Algorithm（2）

- tensorflow模型参数保存和加载问题 2016-12-31 阅读 587
- Ubuntu16.04 配置tensorflow gpu版本 2016-11-26 阅读 816

参考知识库

深度学习知识库
12185 关注 | 549 收录

机器学习知识库
17384 关注 | 2150 收录

Python知识库
22340 关注 | 1443 收录

jQuery知识库
8556 关注 | 948 收录

MongoDB知识库
6981 关注 | 855 收录

Git知识库
6340 关注 | 613 收录

软件测试知识库
4333 关注 | 318 收录

Apache Spark知识库
6731 关注 | 405 收录

算法与数据结构知识库
15314 关注 | 2320 收录

猜你在找

阿里云机器学习算法应用实践
人工智能之机器学习算法的介绍
python数据分析与机器学习实战
使用决策树算法对测试数据进行分..
统计机器学习入门——线性模型选..
TensorFlow学习笔记7--实现卷积神..
神经网络CNN训练心得--调参经验
斯坦福大学机器学习第六课神经网..
CS231n第七课卷积神经网络学习...
深度学习预备课神经网络

查看评论

暂无评论

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenS
Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery BI HTML5 Spring
SDK IIS Fedora XML LBS Unity Splashtop UML components Windows Mobile Rai
CloudStack FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App SpringS
aptech Perl Tornado Ruby Hibernate ThinkPHP HBase Pure Solr Angular Cloud Fou
Bootstrap