Neural Networks for Face Recognition

09013119 严晟嘉

Reference

- http://www.cs.cmu.edu/afs/cs.cmu.edu/user/mitchell/ftp/faces.html
- 机器学习/ (美) 米歇尔(Mitchell, T. M.)著; 曾华军等译. --北京: 机械工业出版社, 2003. 1
- Tom Mitchell. hw97. In: 15-681:Machine Learning. Carnegie Mellon University

1 介绍

应用神经网络来进行人脸识别。两部分工作:

- (独立完成) 应用神经网络来训练 a sunglass recognizer, a face recognizer, a pose recognizer
- (选做) 用神经网络做些感兴趣的事

1.1 头像图片

faces文件夹中有20个子文件夹，每个子文件夹对于一个人。

每张图片的名字遵循这个的规则:

<userid>_<pose>_<expression>_<eyes>_<scale>.pgm

1.2 查看图片

使用程序 xv，有一些地方需要修改才能编译通过，详见 http://carmaux.cs.gsu.edu/xv_install_mac.html

双击运行 xv，右击窗口可以调出控制面板。load查看图片，或者window菜单下的visual schnauzer打开文件夹查看图片。

在打开的图片上按住鼠标中间并拖动可以看到光标所在像素点的RGB值和HSV值(H:色调，S:饱和度，V:明度)。

1.3 神经网络和读取图片的代码

提供的C语言代码实现了一个三层全连接的前馈神经网络，使用反向传播算法来调整它的权值。

提供了图片文件夹、用于训练和测试的顶层程序、一个可视化隐藏单元权值的程序。

cd 进入程序所在文件夹，make编译，会产生一个可执行文件 facetrain。

facetrain以一系列图片为输入，将它们作为神经网络的训练集和测试集。

facetrain可以用于训练和识别，并能够将网络保存到文件中。

在我的电脑环境下，OS X 10.11.4 ，程序需要做一下改动才能成功编译运行

- facetrain.c : line 141 “return”缺少返回值,改成 return 0;
- 修改 trainset 中 *.list 中路径: /Users/yanshengjia/Desktop/neural-networks-for-face-recognition/data/faces/...
- 运行时出现segment fault，找了好久原因，后来才发现是 pgmimage.c 中 imgl_load_images_from_textfile() 函数中 char buf[] 申请的空间太小，而这个字符数组接受的list文件中的字符流可能会很长
- 在facetrain.c中有很多未声明的函数，也就是隐式声明函数，C语言默认让他们返回int型，所以在链接的文件中这些函数的返回值都要改成int
- backprop.c : line 91 105 在函数名前加void
- 在facetrain.c中添加头文件 <stdlib.h>
- facetrain.c : line 91 evaluate_performance()函数中第三个参数0去掉
- 在outtopgm.c和hidtopgm.c中添加头文件<stdlib.h>
- 在imagenet.c中添加头文件<string.h>
- 在pgmimage.c中添加头文件<string.h> <stdlib.h>
- 在backprop.c中添加头文件 <stdlib.h> <unistd.h> <sys/types.h> <sys/stat.h> <fcntl.h>

2 实验

2.1 第一部分

回答下面实验序列中的问题。

1.获取训练集(train set)和测试集(test set)。

2.最初的代码是用来学习特定人物识别器（specific person recognizer）。

默认识别的是glickman这个人，如果要识别其他人，只要在 imagenet.c 中修改 load_target函数中 strcmp(userid, “glickman”)中第二个参数的值就可以了。

修改代码以实现一个太阳镜识别器（sunglasses recognizer），也就是说，训练一个神经网络，当输入一张图片时，判断图片中的人脸有没有戴眼镜。
看 Section 3 的开头来了解如何修改这份代码。

3.训练一个神经网络使用默认的学习参数设置(学习速率 learning rate 0.3，冲量 momentum 0.3)，学习75轮(epochs)，使用下面的命令行：

```
./facetrain -n shades.net -
t straightrnd_train.list -1 straightrnd_test1.list -2 straightrnd_t
est2.list -e 75
```

shades.net：神经网络文件的名称，当训练结束时它会被保存。
straightrnd_train.list：训练集（70例）
straightrnd_test1.list：测试集1（34例）
straightrnd_test2.list：测试集2（52例）

上面的命令创建并训练了你的神经网络基于随机选择的70张图（156张『straight』图中），并在随机选择的34张图和52张图上分别测试。

测试策略是这样的，1/3的图片（straightrnd_test2.list）用于测试；2/3的图片用于训练和交叉校验，这其中的2/3（straightrnd_train.list）用于训练，1/3（straightrnd_test1.list）用于校验集合来决定何时停止训练。

4.What code did you modify?
[imagenet.c]load_target(): strcmp(userid, “glickman”) ->
strcmp(eyes, “sunglasses”)

What was the maximum classification accuracy achieved on the training set?
100%

How many epochs did it take to reach this level?
30 epochs

How about for the validation set(straightrnd_test1.list)?
max 100%, 20 epochs

The test set(straightrnd_test2.list)?
max 94.2308%, 45 epochs

5.如果在同样的环境同样的参数和输入，得到的结果也是一样的。
因为这份代码每次都使用相同的种子（seed）来产生随机数，以在创建网络的时候初始化权值([-1,1])

仔细阅读 Section 3.12有助于理解如何解释实验并回答这些问题。

6.现在，让我们来实现一个 1-20人脸识别器（face recognizer）。
实现一个神经网络，以一张图片为输入，输出它对应的人的userid。
为了实现它，需要改变输出编码（output encoding），你必须能够分别20个不同的人。

Hint: 让 learning rate 和 momentum 仍然为 0.3，使用20个隐藏单元。

像之前一样，训练神经网络，学习100轮
```
./facetrain -n face.net -t straighteven_train.list -1
straighteven_test1.list -2 straighteven_test2.list -e 100
```

为什么只训练脸朝向是『straight』的例子？原因在于机器太慢。
如果机器性能可以，那么可用全部数据集做实验。

straightrnd_*.list 和 straighteven_*.list之间的区别在于前者的文件中图片完全是从训练集和测试集中随机选择的，后者的文件中每个人的图片数量都相对平均。
因为每个人都只有7到8张朝向是『straight』的图片，不将它们均分的话可能会导致我们的神经网络在有些人的脸上测试很多，但这些人脸是被训练得很少的。也就是说，效果会变差。

7.Which part of the code was it necessary to modify this time?

- use 20 hidden units, 20 output units, 20 target units.
- modify [facetrain.c]evaluate_performance()
- add [facetrain.c]result_on_imagelist()
- print result

How did you encode the outputs?

```
1 an2i      <.9, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1>
2 at33      <.1, .9, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1>
3 boland    <.1, .1, .9, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1>
4 bpm       <.1, .1, .1, .9, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1>
5 ch4f      <.1, .1, .1, .1, .9, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1>
6 cheyer    <.1, .1, .1, .1, .1, .9, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1>
7 choon     <.1, .1, .1, .1, .1, .1, .9, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1>
8 danieln   <.1, .1, .1, .1, .1, .1, .1, .9, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1>
9 glickman  <.1, .1, .1, .1, .1, .1, .1, .1, .9, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1>
10 karyadi  <.1, .1, .1, .1, .1, .1, .1, .1, .1, .9, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1>
11 kawamura <.1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .9, .1, .1, .1, .1, .1, .1, .1, .1, .1>
12 kk49     <.1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .9, .1, .1, .1, .1, .1, .1, .1, .1>
13 megak    <.1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .9, .1, .1, .1, .1, .1, .1, .1>
14 mitchell <.1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .9, .1, .1, .1, .1, .1, .1>
15 night    <.1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .9, .1, .1, .1, .1, .1>
16 phoebe   <.1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .9, .1, .1, .1, .1>
17 saavik   <.1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .9, .1, .1, .1>
18 steffi   <.1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .9, .1, .1>
19 sz24     <.1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .9, .1>
20 tammo    <.1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .1, .9>
```

What was the maximum classification accuracy achieved on the training set? 100%

How many epochs did it take to reach this level? 64 epochs

How about for the validation and test set?

- [validation set(straighteven_test1.list)] max 88.8889%, 79 epochs
- [test set(straighteven_test2.list)] max 85%, 59 epochs

8.现在让我们看看在哪些图片上我们的神经网络可能会分类失败：

```
./facetrain -n face.net -T -1 straighteven_test1.list -2 straighteven_test2.list
```

在分类失败的图片上有没有一些共性呢?

9.实现一个朝向识别器（pose recognizer）。

实现一个ANN，当给定一张图片作为输入，判断该图片中的人是看着 前面（straight），上面（up），左边（left）还是 右边（right）。

为了这个任务也要改变输出编码。

Hint: 让 learning rate 和 momentum 为 0.3，使用6个隐藏单元。

10.训练我们的ANN100轮，这次在所有的数据集上训练：

```
./facetrain -n pose.net -t all_train.list -1 all_test1.list -2
all_test2.list -e 100
```

因为 pose-recognizing 网络大体上比 face-recogzing 网络有更少的权重（weights）需要更新，所以即使机器比较慢，仍然可以使用所有的图片来玩这个实验。

这里用到的，训练集中有260张图片，测试集1中有140张图片，测试集2中有193张图片。

11.What code did you modify?

- use 6 hidden units, 4 output units, 4 target units
- modify [facetrain.c]result_on_imagelist

How did you encode your outputs this time?

```
left      <0.9, 0.1, 0.1, 0.1>
right     <0.1, 0.9, 0.1, 0.1>
straight  <0.1, 0.1, 0.9, 0.1>
up        <0.1, 0.1, 0.1, 0.9>
```

What was the maximum classification accuracy achieved on the training set? 99.639%

How many epochs did it take to reach this level? 48 epochs

How about for each test set?
[test1 set]max 85.6115%, 99 epochs　　[test2 set]max 91.3462%, 51 epochs

12.现在考察 反向传播算法是如何基于每个像素来调整隐藏单元的权重。

首先，输入 make hidtopgm 来编译工程。

然后，可视化 隐藏单元n 的权重，输入：

```
hidtopgm pose.net image-filename 32 30 n
```

在图片 image-filename 上调用 xv 可以显示权重的范围，最低权重映射到像素值0，最高权重映射到像素值255。

假如这图片看上去是噪声，试着使用 facetrain_init0 （由 make

facetrain_init0编译）重新训练，这样可以初始化新的神经网络的隐藏单元的权重为0，而不是随机值。

13.Do the hidden units seems to weight particular regions of the image greater than others?

Do particular hidden units seem to be tuned to different features of some sort?

2.2 第二部分

玩点好玩的！表情识别器(expression recognizer)

训练ANN100轮：

```
./facetrain -n newexpression.net -t all_train.list -1 all_test1.list -2 all_test2.list -e 100
```

用训练过10000轮的网络再训练100次：

```
./facetrain -n expression.net -t all_train.list -1 all_test1.list -2 all_test2.list -e 100
```

输出编码

```
happy       <0.9, 0.1, 0.1, 0.1>
neutral     <0.1, 0.9, 0.1, 0.1>
sad         <0.1, 0.1, 0.9, 0.1>
angry       <0.1, 0.1, 0.1, 0.9>
```

实验结果(训练10000轮)

[training set]max 94.5848%　　[test1 set]max 15.8273%　　[test2 set]max 17.3077%

其他可选方案（当然不限于此）

- 使用 pose recognizer 的输出作为 face recognizer 的输入，观察这怎么影响实验效果。要实现这个实验，需要增加一个机制来保存 pose recognizer 的输出单元，增加一个机制来装载这个数据到 face recognizer 里。
- 学习图片中一些特征的位置，比如眼睛。可以用xv来得到这些特征在每张图片上的坐标。
- 使用图片包，权值可视化程序，或其他你觉得可能使用的来更好地理解神经网络到底学习到了什么。使用这些信息，你认为神经网络正在学什么？你能用这些信息来提升性能吗？

- 改变输入或输出编码来提升识别准确率。
- 改变隐藏单元的个数，训练样例的个数，迭代的次数，学习速率和冲量，或其他任何你想尝试改变的，尽可能使训练集和测试集的准确度相差大（你能使这个神经网络过拟合到多差），和尽可能使训练集和测试集的准确率相差小（你能使这个网络达到多好的表现）。

3. 文档

本次作业的代码分为一些模块

- pgmimage.c, pgmimage.h
  - 处理图片的模块
  - 支持读写 PGM 图片文件和像素存取/赋值
  - 提供了 IMAGE 数据结构和 IMAGELIST 数据结构（图片指针的数组，在处理许多图片时有用）
- backprop.c, backprop.h
  - 神经网络模块
  - 支持三层全连接前馈神经网络
  - 使用 backpropagation 算法来调整权值
  - 提供高等级的程序来创造、训练和使用神经网络
- iamgenet.c
  - 用于装载图片到网络的输入单元，和设置训练的目标向量的 接口程序
  - 当实现 face recognizer 和 pose recognizer 的时候，需要修改 **load_target**，根据你选择的输出编码设置合适的目标向量
- facetrain.c
  - 顶层程序，使用以上所有的模块来实现一个“TA”识别器
  - 修改这个代码来改变神经网络的大小和学习参数，这两个都是无关紧要的改变
  - 表现评估程序 **performance_on_imagelist()** 和 **evaluate_performance()** 都在此模块中，修改他们来实现 face recognizer 和 pose recognizer
- hidtopgm.c
  - 隐藏单元权值可视化程序

需要修改 imagenet.c 和 facetrain.c

3.1 facetrain

3.1.1 运行 **facetrain**

facetrain 在终端中运行有许多不同的选项。如果什么都不输，会显示一些总结信息。

- -n <network file>
  - 或装载一个已存在的网络文件，或用所给的名字创建一个新的网络文件
  - 在训练的最后，神经网络会被保存到这个文件中
- -e <number of epochs>
  - 训练的次数
  - 默认为100
- -s <seed>
  - 随机数产生器的种子
  - 默认为102194
  - 改变种子尝试不同的随机数序列来重新实验
- -S <number of epochs between saves>
  - 保存网络需要的最小训练次数
- -t <training image list>
  - 指定训练集
  - 如果没有这个参数，意味着没有训练（训练集都是0），网络直接在测试集上跑
- -1 <test set 1 list>
  - 指定测试集1
  - 如果没有这个参数，测试集1都是0
- -2 <test set 2 list>
  - 指定测试集2
  - 如果没有这个参数，测试集2都是0

3.1.2 **facetrain** 的输出

facetrain首先读入所有数据文件并输出一些相关信息。当所有数据都读入，开始训练。网络的训练集和测试集的表现每次迭代都会输出一行信息。

<epoch> <delta> <trainperf> <trainerr> <t1perf> <t1err> <t2perf> <t2err>

- **epoch** 已完成的迭代次数；为0表示训练还没开始
- **delta** 对于本次迭代的所有训练样例，在backprop中计算的隐藏单元和输出单元的δ值之和
- **trainperf** 训练集中的例子分类正确的比例

- **trainerr** 对于所有训练样例，误差方程 1/2 * ∑(ti-oi)^2 的均值，ti是输出单元i的目标值，oi是该单元的实际输出值
- **t1perf** 测试集1中的例子分类正确的比例
- **t1err** 对于测试集1中的所有例子，误差方程 1/2 * ∑(ti-oi)^2 的均值
- **t2perf** 测试集2中的例子分类正确的比例
- **t2err** 对于测试集2中的所有例子，误差方程 1/2 * ∑(ti-oi)^2 的均值

3.2 建议

观察 imagenet.c, facetrain.c 来了解程序是如何运行的。

可以先看一遍 facetrain.c 来看看训练流程是怎么样的。

imagenet.c中的load_target()用来设置训练的目标向量。

performance_on_imagelist()和evaluate_performance()分别用于评估表现和计算误差数据。前者在图片集中迭代，计算这些图片的平均误差，后者计算单个图片的误差和准确率。

处理图片模块可以以图像形式查看单元间连接的权值，你可以找到创建和写这样的图像的程序。

3.3 代码简介

略

谢谢！

https://github.com/yanshengjia/artificial-intelligence/tree/master/neural-networks-for-face-recognition/src