[AI 4 Med 笔记(1.1)胸片诊断](https://goldengrape.github.io/posts/python/AI4med/AI4med_note_1_1/)

第一门课叫AI For Medical Diagnosis, 也就是用于诊断的AI. 有三周的课程

用于诊断一般就是图像识别, 看胸片、眼底照片、CT、核磁. 其中MRI是3D数据, 其他一般是二维.

诊断问题一般就是二分类, 有病或者没病.

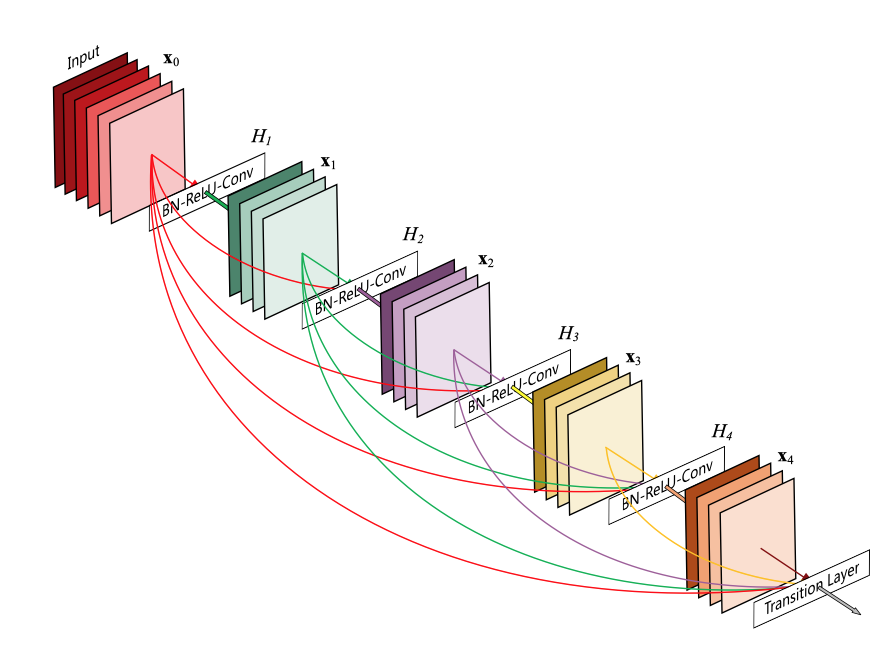
二分类的loss很简单:

L(X,y)=

* -log(P(Y=1|X)) if y==1
* -log(P(Y=0|X)) if y==0

就是把卷积神经网络CNN最后一步的激活函数设成sigmoid, 得到一个在[0,1]之间的概率, 对这个数字求-log.

第一周

以胸片为目标来讲解, 用的是[Densenet](https://www.jiqizhixin.com/graph/technologies/d4b7fe6b-e73a-4ad3-af08-1b323f78333b) 

和ResNet差不多, 只不过短路的路径可以更多一些, 方便在网络过深时梯度的传递.

讲训练阶段常见的几个问题:

* imbalance问题: 数据集当中中病人总是少数
* 多任务问题: 一张片子上, 可能同时有几种病
* 训练集数量少

imbalance问题

通常来说得病的都是少数, 正常的数据总是更多一些, 那么在训练集中就有阳性与阴性不平衡的问题. 老师讲到应该计算一下总的loss, 看看来自阳性和阴性结果的贡献比例, 应当比较均衡, 这样在训练时去减少loss的时候, 才能够同时优化阳性和阴性的结果.

所以, 有几种解决方案:

loss权重方案

我比较喜欢这个方案, 对阳性和阴性的结果添加不同的权重, 使得两部分的贡献更均衡. 如果病人(阳性结果)很少, 那么如果真实值是阳性, 算错了, 就要给更强的惩罚.

Loss(X,y) =

* Wp x -log P(Y=1|X), if y=1
* Wn x -log P(Y=0|X), if y=0

其中:

* 阳性权重 Wp=阴性数量/总数
* 阴性权重 Wn=阳性数量/总数

重采样方案

如果不改权重, 那么就让训练集中较少的阳性数据被更多次采样. 个人不喜欢, 写代码也麻烦, 可能还容易过拟合.

多任务问题

如果是同一张片子上看多个疾病, 就是同时进行多个问题的二分类. (但这样也就切断了疾病之间的联系). 于是把各个疾病的loss加起来就好了.

Loss=L(X, y\_疾病0)+L(X, y\_疾病1)+L(X, y\_疾病2)...

训练集数量少

理论上, 而且通常也都是要做数据集的扩充, 把图片移动一下位置、轻微旋转一下之类, 但是注意不要做镜像了, 心脏位置是分左右的. 镜像了诊断可就不一样了. 也就是说, 对图像的扩充修改必须在不影响诊断的基础上进行.

病例数量这种事, 就是资源的问题吧, 没钱也真是没办法.

数量少, 就只好使用迁移学习了, 就是把已经训练好的神经网络拿来用, 但是留下最后一层重新训练一下.

训练结果评估

分割数据集

数据集要分成train、val、test. 对于医疗数据, 要注意一个病人可能会拍好几张片子, 在分割数据集的时候, 要以病人为单位来分割, 而不能以数据集中的图像来作为单位.

也就是说一个病人所有的片子, 要么都在训练集里, 要么都在测试集里, 不能同时出现在训练集和测试集里. 否则可能出现数据泄露的问题, 比如患者一直都戴着项链, AI可能识别的是项链, 而不是肺部的纹理.

金标准

要训练机器去学习诊断, 人类给出的诊断要可靠才行, 金标准要么有更进一步的检查, 要么就是委员会投票表决.

代码操作时的问题

这个系列的课程, 课程讲解只占了一部分的知识, 还有很多是在代码里.

探索数据

拿到一个数据集, 先得看看. 检查一下标签数据表的结构, 看看各个字段的数据类型, 检查一下是否有null value, 用df.info()可以看个大概.

要检查一下ID的独立性, 统计一下独立的ID有多少, 是不是少于总数.

要看看各个标签下阳性的数据量大概是多少.

打印几张图出来看看.

图像预处理

keras有现成的预处理工具 [ImageDataGenerator](https://keras-cn.readthedocs.io/en/latest/preprocessing/image/)

现把图像按照自身的均值和方差标准化一下, 注意是按照自身的均值和方差, 而不是多个样本的均值和方差. 概念上会不一样.

**from** **keras.preprocessing.image** **import** ImageDataGenerator

*# Normalize images*

image\_generator = ImageDataGenerator(

samplewise\_center=**True**, *#Set each sample mean to 0.*

samplewise\_std\_normalization= **True** *# Divide each input by its standard deviation*

)

然后生成一个generator

*# Flow from directory with specified batch size and target image size*

generator = image\_generator.flow\_from\_dataframe(

dataframe=train\_df,

directory="nih/images-small/", *# 存放路径*

x\_col="Image", *# features*

y\_col= ['Mass'], *# labels*

class\_mode="raw", *# 'Mass' column should be in train\_df*

batch\_size= 1, *# images per batch*

shuffle=**False**, *# shuffle the rows or not*

target\_size=(320,320) *# width and height of output image*

)

可以统一输出大小. 不过吐槽一下, 320x320的图像就能诊断了, 拍那么高清的片子干啥.

数数

记得要给每一个标签数出有多少阳性多少阴性, 然后生成loss的权重.

$$ loss^{(i)} = loss\_{阳性}^{(i)} + los\_{阴性}^{(i)} $$

$$loss\_{阳性}^{(i)} = -1 \times weight\_{阳性}^{(i)} \times y^{(i)} \times log(\hat{y}^{(i)} + \epsilon)$$

$$loss\_{阴性}^{(i)} = -1 \times weight\_{阴性}^{(i)} \times (1- y^{(i)}) \times log(1 - \hat{y}^{(i)} + \epsilon)$$

$$\epsilon = \text{小量}$$

注意计算权重的时候, 是反过来的,

weight\_{阳性}= 阴性数量/总数

才能达到阴阳调和的目的.

病人重叠和数据泄露

训练集的信息不能泄露到了测试集里. 在医学数据里, 就要检查训练集合和测试集合里是不是有同样的ID, 如果有同一个病人出现在两边, 注意, AI很可能去看他/她的项链或者戒指去了.

DensNet121

作业中使用的深度学习网络是[DensNet121](https://www.kaggle.com/pytorch/densenet121)

Keras里有直接写好的, 直接导入就可以了, 我看这前后也通用, 抄下来.

**from** **keras.applications.densenet** **import** DenseNet121

*# create the base pre-trained model*

*# 导入的时候最后一层数据不要*

base\_model = DenseNet121(weights='./nih/densenet.hdf5', include\_top=**False**)

x = base\_model.output

*# add a global spatial average pooling layer*

x = GlobalAveragePooling2D()(x)

*# and a logistic layer*

*# 来一层sigmoid, 得到一个[0,1]的预测概率*

predictions = Dense(len(labels), activation="sigmoid")(x)

model = Model(inputs=base\_model.input, outputs=predictions)

*# 用adam训练*

model.compile(optimizer='adam', loss=get\_weighted\_loss(pos\_weights, neg\_weights))

需要训练的话:

history = model.fit\_generator(train\_generator,

validation\_data=valid\_generator,

steps\_per\_epoch=100,

validation\_steps=25,

epochs = 3)

plt.plot(history.history['loss'])

plt.ylabel("loss")

plt.xlabel("epoch")

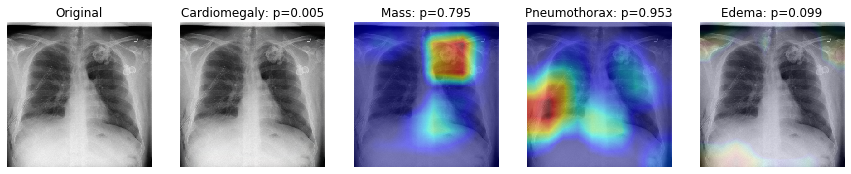
plt.title("Training Loss Curve")

plt.show()

不过老师在题目里一个劲的劝, 40GB+的数据量, 大家就用我们算好的吧.

GradCAM

[GradCAM](https://arxiv.org/abs/1610.02391)太帅了! 可以给出一个热区图, 看哪部分跟诊断关系最密切. 这么有用的工具, 居然就在课程作业里悄悄带过了. 于是我也不知道具体的用法, 但以后自己做东西的话, 一定要用上这个.



关于GradCAM的[参考](https://www.jiqizhixin.com/articles/where-cnn-is-looking-grad-cam)、[Keras实现教程](https://www.pyimagesearch.com/2020/03/09/grad-cam-visualize-class-activation-maps-with-keras-tensorflow-and-deep-learning/)

[AI 4 Med 笔记(1.2)各种率](https://goldengrape.github.io/posts/python/AI4med/AI4med_note_1_2/)

第一门课第二周, 主要讲

**各种率**

* 真假阴阳各种率
* 敏感性和特异性
* 阳性预测率(PPV)和阴性预测率
* 置信区间, ROC曲线, F1 score

我之前学Scikit-learn的时候, 笔记记得挺清楚的.

参考

* [Scikit-learn笔记2](https://goldengrape.github.io/posts/python/scikit-learn-note-2/)
* [Scikit-learn笔记3](https://goldengrape.github.io/posts/python/scikit-learn-note-3/)
* [Scikit-learn笔记4](https://goldengrape.github.io/posts/python/scikit-learn-note-4/)

而且这张图做得很好, 一张图说明问题, 先看图, 再看后面的计算方式:

例如有这样的结果:

|  | **预测=A** | **预测=B** | **预测=C** |
| --- | --- | --- | --- |
| 实际=A | 100 | 6 | 11 |
| 实际=B | 1 | 110 | 12 |
| 实际=C | 2 | 7 | 120 |

以下表格中, 分子是红字, 分母是黄色背景, 考虑C的情况

**Accuracy**

* 正确率
* 有病没病都说对了的概率

|  | **预测=A** | **预测=B** | **预测=C** |
| --- | --- | --- | --- |
| 实际=A | 100 | 6 | 11 |
| 实际=B | 1 | 110 | 12 |
| 实际=C | 2 | 7 | 120 |

Accuracy=100+110+120100+6+11+1+110+12+2+7+120

**Recall**

* 敏感性, 检出率, 真阳性率.
* 实际有病, 测出有病的概率

|  | **预测=A** | **预测=B** | **预测=C** |
| --- | --- | --- | --- |
| 实际=A | 100 | 6 | 11 |
| 实际=B | 1 | 110 | 12 |
| 实际=C | 2 | 7 | 120 |

Recall=1202+7+120

**Precision**

* 精确度, 阳性预测率, PPV
* 测出有病, 还说对了的概率

|  | **预测=A** | **预测=B** | **预测=C** |
| --- | --- | --- | --- |
| 实际=A | 100 | 6 | 11 |
| 实际=B | 1 | 110 | 12 |
| 实际=C | 2 | 7 | 120 |

Precision=12011+12+120

**Specificity**

* 特异性, 真阴性率
* 测出没病, 还说对了的概率

|  | **预测=A** | **预测=B** | **预测=C** |
| --- | --- | --- | --- |
| 实际=A | 100 | 6 | 11 |
| 实际=B | 1 | 110 | 12 |
| 实际=C | 2 | 7 | 120 |

Specificity=100+6+1+110100+6+1+110+2+7

**False Positive Rate**

* 假阳性率, 误报率
* 本来没病, 测出有病的概率

|  | **预测=A** | **预测=B** | **预测=C** |
| --- | --- | --- | --- |
| 实际=A | 100 | 6 | 11 |
| 实际=B | 1 | 110 | 12 |
| 实际=C | 2 | 7 | 120 |

FalsePositiveRate=11+12100+6+11+1+110+12

**真假阴阳性**

|  | **预测为(-)** | **预测为(+)** |
| --- | --- | --- |
| 实际为(-) | TN | FP |
| 实际为(+) | FN | TP |

* TP = true positive 真阳性
* FP = false positive (Type I error) 假阳性(Type I错误) , 印象里就是P<0.05的0.05
* TN = true negative 真阴性
* FN = false negative (Type II error)假阴性(Type II错误)

**敏感特异性**

* Accuracy: 分类正确的概率. 实际为(-)预测为(-), 实际为(+)预测为(+)叫做正确.

Accuracy=TP+TNTP+TN+FP+FN

* Classification error: (1-Accuracy), 分类错误的概率

Classificationerror=FP+FNTP+TN+FP+FN

* Recall, 真阳性率: 实际上为(+), 能够被预测成(+)的概率
  + Recall又叫
  + True Positive Rate (TPR): 真阳性率
  + Sensitivity: 敏感性
  + Probability of detection: 检出率

Recall=TPTP+FN

* Precision: 如果预测为(+), 那么预测正确的概率

Precision=TPTP+FP

* False positive rate (FPR): 假阳性率, 误报率. 本来实际上是(-)的, 结果分类器报告成(+)的概率

FalsePositiveRate=FPTN+FP

* Specificity, 特异性

应该是真阴性率(之前写错了), TNR, 实际上为(-), 能够被测成(-)的概率

Specificity=TNTN+FN

**F1-score: 结合precision与recall**

F1=2precision×recallprecision+recall=2TP2TP+FN+FP

**F-score: 更一般地将precision与recall结合成单独一个数**

Fβ=(1+β2)precision×recallβ2×precision+recall=(1+β2)TP(1+β2)TP+FN+FP

𝛽用来调整recall vs precision之间的重要程度:

* Precision-oriented users: 𝜷 = 0.5
* Recall-oriented users: 𝜷 = 2

**阳性预测率 PPV**

实际上就是

* Precision: 如果预测为(+), 那么预测正确的概率

Precision=TPTP+FP

PPV=Precision=sensitivity×prevalencesensitivity×prevalence+(1−specificity)×(1−prevalence)

其中 prevalence=P(pos), 阳性结果的发生率. 如果自己都知道TP、FP的数据, 当然用简单的公式计算, 如果拿到的是一个标记好各种率的试剂盒, 或者是看一篇新闻报道, 对方把各种率搅合在一起, 那么可能就要通过后一种公式来计算

**置信区间**

比如某个概率p=0.80(95% CI 0.78, 0.82),

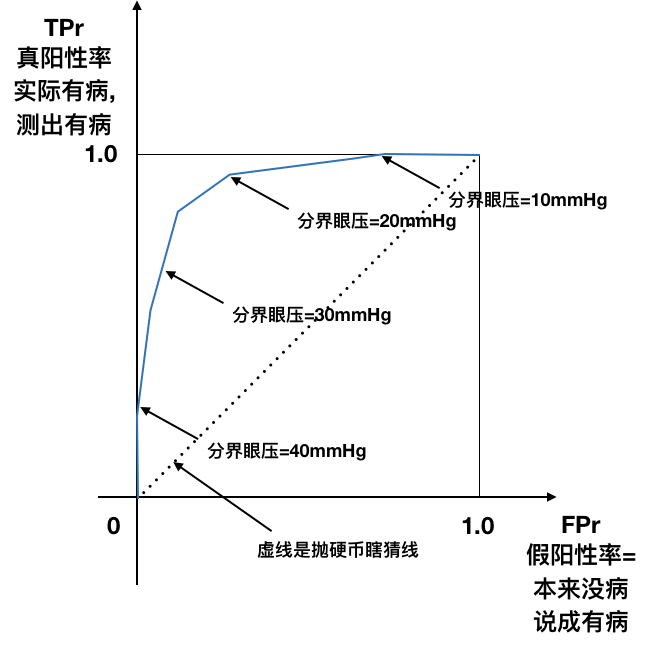
是说, 真实的概率p, 谁知道是啥, 有95%的概率, 会落在[0.78, 0.82]的范围内. 置信区间的范围和样本量有关, 样本量越大, 置信区间的宽度越窄.

**ROC曲线**

* X轴: False Positive Rate 假阳性率
* Y轴: True Positive Rate 真阳性率

左上顶点:

* 理想点
* False positive rate 假阳性率=0
* True positive rate 真阳性率=1



* ROC曲线越往左上, 越好.
* 曲线下面积(AUC), 越大越好
* 45度线是随机瞎猜线

