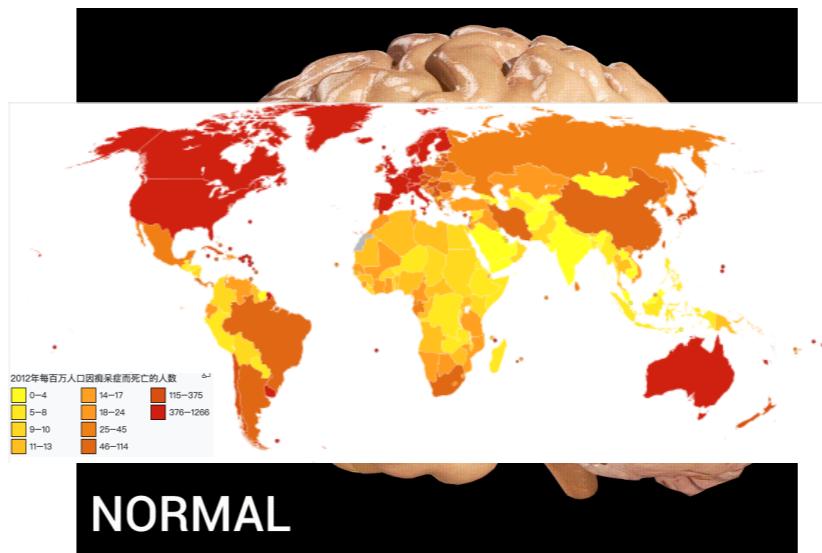


阿尔茨海默病的诊断

使用机器学习

沈星宇 2022/06/02

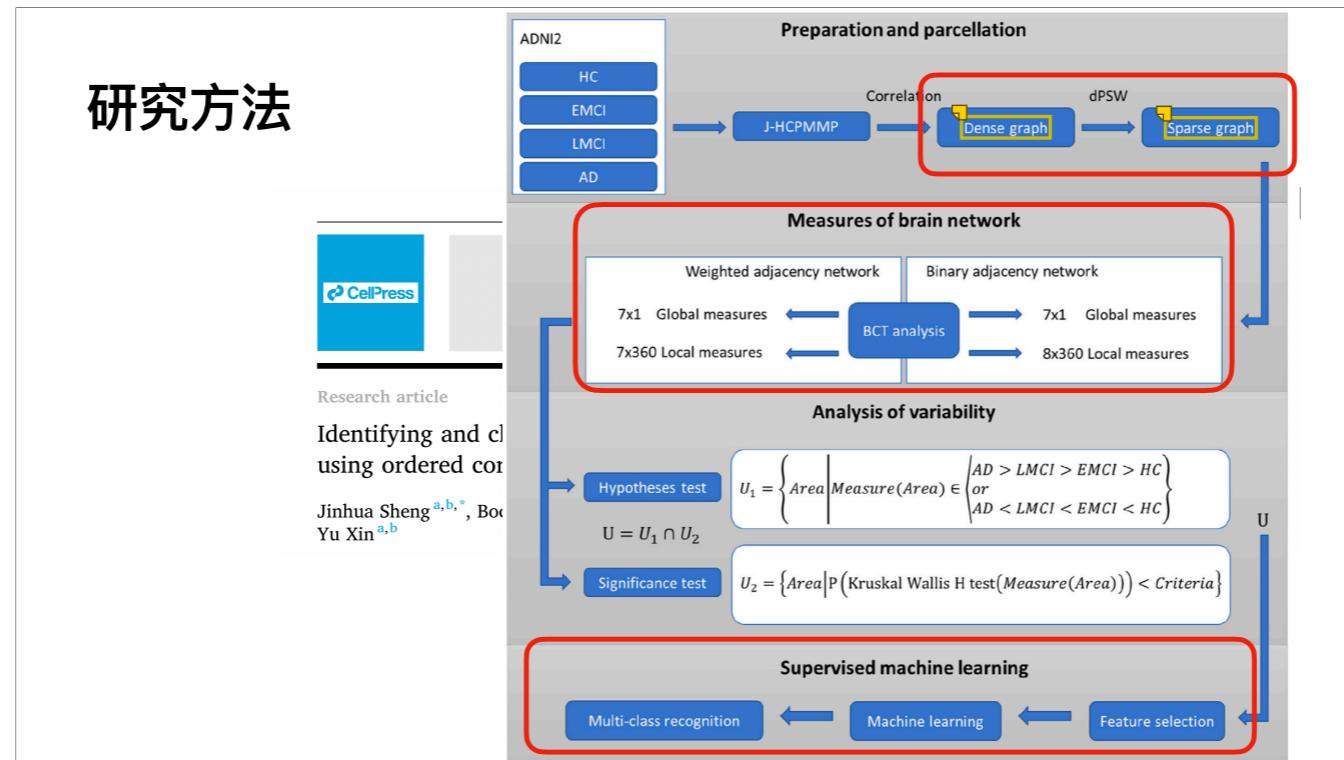
研究背景



阿尔兹海默症（AD）是当今能导致痴呆症的最常见的疾病之一。2018年有大约5千万人患病。而据预测，到了2030年，将会达到1亿5千万。尽管AD必须在60岁之后才能被诊断出，但是对于某些拥有特定基因突变的个体，AD的症状也有可能出现的非常早（30-50岁）。也就是说，AD的发展是一个漫长的过程：从正常人变为轻度认知障碍（MCI），再随着时间慢慢发展成AD。所以当下的研究多着重于找到如何区分轻度认知障碍和AD的方法。图中演示的是正常人的和AD患者大脑形态上的区别。

这是2012年每百万人口因痴呆而死亡的人数。可见痴呆症是切实影响到了患者的生活质量。

研究方法



这次研究主要的理论依据是这篇论文，来自Heliyon的Identifying and characterizing different stages toward Alzheimer's disease using ordered core features and machine learning。

这张图是论文整个数据处理和机器学习的流程图，包括脑区划分，稀疏化，特征提取，假设检验，重要性检验，特征选择，机器学习。也是我的主要流程。这里要注意的是：脑区划分基于文章中提到的J-HCPMMP方法划分了360个脑区。但是这一步的结果是老师直接提供的matlab矩阵，所以这部分不作详细描述。

我们主要的工作主要就是用红圈圈起来的三个部分。接下来我详细说明这三个部分的研究方法

研究步骤

稀疏化

- 减少噪声对结果的影响
- 加快计算速度

360x360 single

	1	2	3	4	5	6
1	1	0.7720	0.5885	0.9196	0.9221	0.8603
2	0.7720	1	0.6166	0.8220	0.7832	0.7562
3	0.5885	0.6166	1	0.7528	0.5695	0.4477
4	0.9196	0.8220	0.7528	1	0.9392	0.8465
5	0.9221	0.7832	0.5695	0.9392	1	0.8870
6	0.8603	0.7562	0.4477	0.8465	0.8870	1
7	0.7405	0.5146	0.4000	0.7476	0.7426	0.8248
8	0.3248	0.2275	0.5500	0.4159	0.4230	0.1850

360x360 single

	1	2	3	4	5	6
1	0	0	0	0.9196	0.9221	0.8603
2	0	0	0	0.8220	0	0
3	0	0	0	0	0	0
4	0.9196	0.8220	0	0	0.9392	0.8465
5	0.9221	0	0	0.9392	0	0.8870
6	0.8603	0	0	0.8465	0.8870	0
7	0	0	0	0	0	0.8248
8	0	0	0	0	0	0

首先是矩阵的预处理。老师给的矩阵是密集的连接矩阵，第一步就是稀疏化。

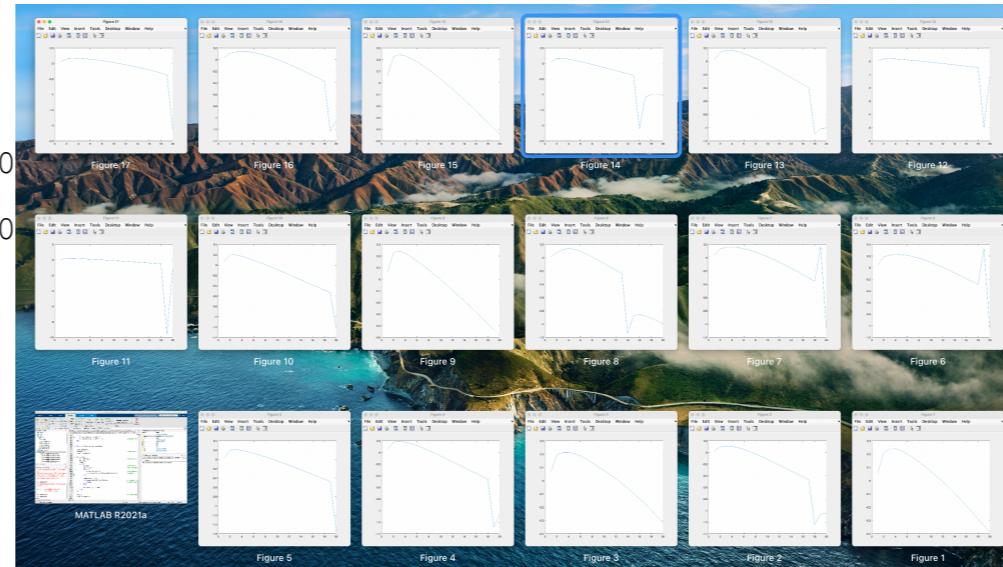
连接矩阵中 (x, y) 点的值代表第 x 个脑区和第 y 个脑区的连接强度。这个值的范围大概在 -0.1 ~ 1 之间，值越大代表脑区之间的联系越紧密。最初，由于扫描环境引起的噪音和虚假连接，因此矩阵会稠密，看起来每个脑区之间都有或强或弱的联系，实则不然。那些很小甚至为负的连接强度是由噪音导致的，稀疏化的目的就是去除这些噪音。更具体地来说，就是把其中连接较弱的部分连接剔除，只留下连接较强的部分。稀疏化可以减少噪声对结果的影响和加快计算速度。

接下来讲一讲如何确定剔除和保留哪些连接。

研究步骤

稀疏化

- $\text{psw} = \text{range}(0, 1, 0.05)$
- $\text{psw} = \text{range}(0, 1, 0.01)$



根据论文里提到的dpsw计算方法，在0.01到1之间遍历psw，步长0.05，也就是共20次，关于每个psw，先以此为稀疏化的阈值稀疏化稠密矩阵，再计算这个矩阵的全局效率，再根据这个公式算出GCE。

在20次测试中找出使得GCE最大的psw，即为这个矩阵的dpsw，用这个dpsw作为阈值处理稠密矩阵，就是要得到的稀疏矩阵。

另外，和论文里不同的是，根据我做的17个图，GCE的数值几乎都随着psw增加而单调递减，只有少数几个矩阵在高dpsw的时候gce出现了无法解释的高峰。经过小组讨论认为这是由某种噪声影响导致的，并不能实际采用。

再结合我发现大部分矩阵的dpsw都在0.1到0.2之间，于是决定取0.01到0.25，步长0.01的psw进行遍历，以得出更精确的dpsw。

研究步骤

特征提取 BCT (Brain Connectivity Toolbox)

Global

- optimal_n_modules_bin
- optimalNModules
- global_efficiency_bin
- global_efficiency
- charpath_bin
- charpath
- assortativity_bin
- assortativity
- small_world_index

Local

- local_efficiency
- local_efficiency_bin
- page_rank
- degree
- eigenvector
- strength
- betweenness_centrality
- betweenness_centrality_bin
- clustering_coefficient
- clustering_coefficient_bin

接下来是对稀疏化的矩阵进行特征提取。

我们通过bct工具包一共计算了9个全局特征和10个局部特征。

全局特征包括5个权值特征和4个二值特征，局部特征包括5个权值特征和5个二值特征。

由于分了360个脑区，每个局部特征有360个值，所以全部拼接起来是3609个特征。

当然，这3609个特征值有些比较重要，有些不那么重要，所以下一步就是挑出重要的特征，舍去不那么重要的特征。

研究步骤

降维

- ReliefF

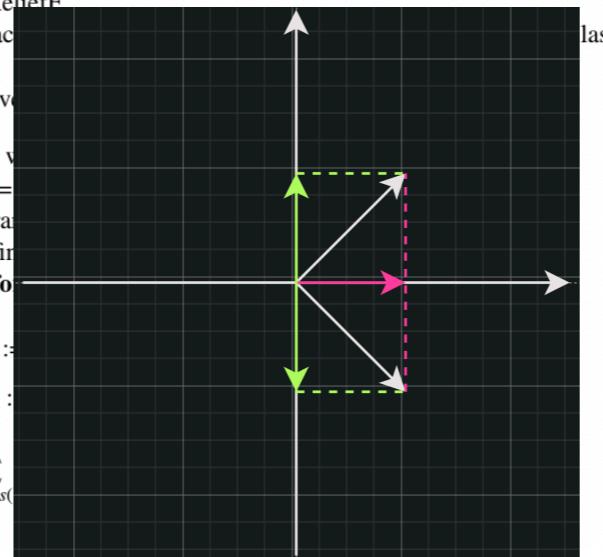
- PCA

Algorithm ReliefF

Input: for each
value

Output: the v

```
1. set all v
2. for i := 1 to n do
3.   rai := 0
4.   for j := 1 to m do
5.     for A := 1 to k do
6.       if  $x_{ij} \in C_A$  then
7.         for A' := 1 to k do
8.           W[A] :=  $\sum_{C \neq class(A)} \frac{1}{|C|} \sum_{x_{ij} \in C} \frac{1}{|C|}$ 
9.         end;
10.      end;
```



我们用了两种方法对3000多个特征值进行降维，他们是ReliefF和PCA。

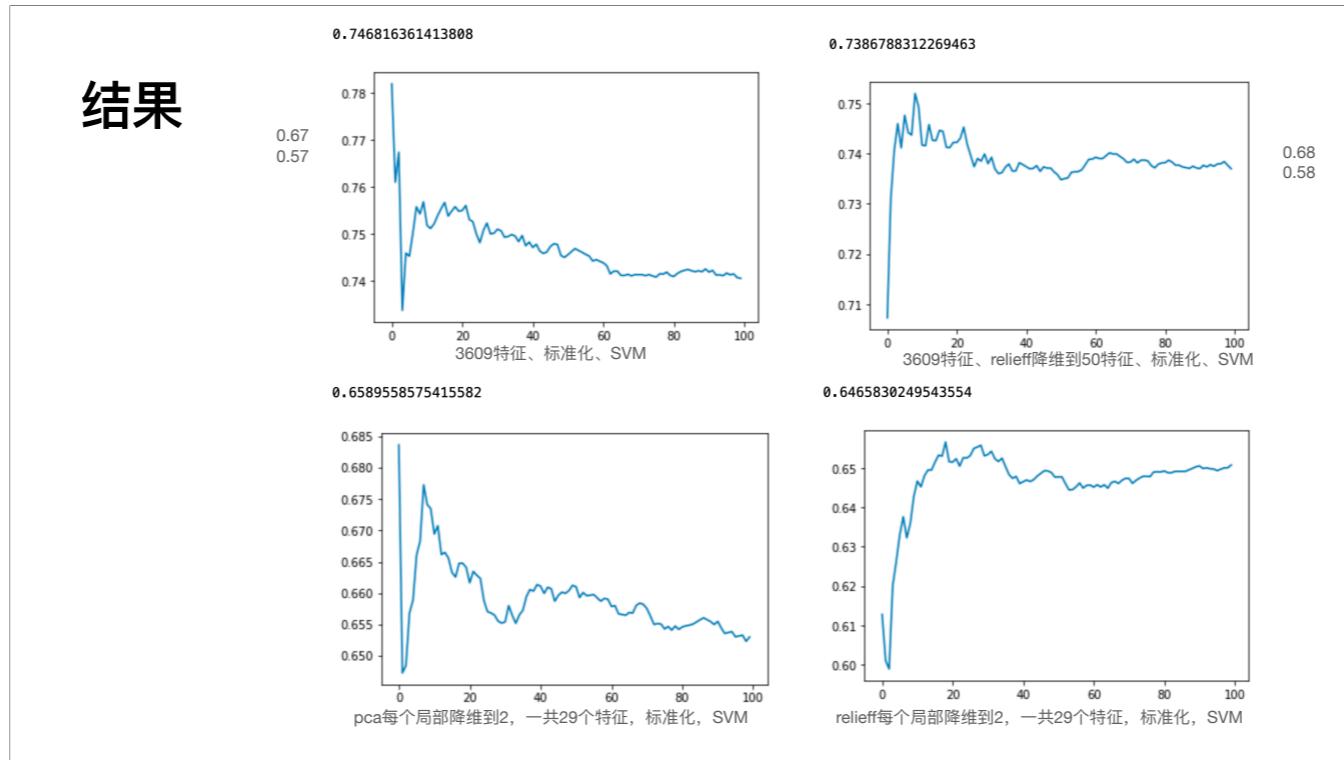
Relieff是之前提到的论文里使用的降维方法，通过计算特征值之间的相关度，剔除相关度较低的特征，保留相关度高的特征。这是ReliefF的伪代码。

PCA的不同之处是，它不删除特征值，而是把特征值看成高维空间中的向量，通过算法找到一个超平面，使得这些向量在该平面上的投影方差最大，相当于最大化保留了样本之间差别的同时尽量降低表示这个区别所需要的维度。

对于我们这个项目来说，全局特征和局部特征的重要性差不多，但是在数量上局部特征是全局特征的300倍，因此在降维过程中全局特征是要保留的，并且降维结果应该是全局特征和局部特征接近1比1的比例。而且要注意的是局部特征的所有特征理论上来说并没有重要性谁高谁低的区分，降维的目的是减少数量，让局部特征不至于淹没更重要的全局特征。

所以我决定对每个局部特征用PCA降维到2维，再和全局特征拼接，最后一共是29个特征。

结果

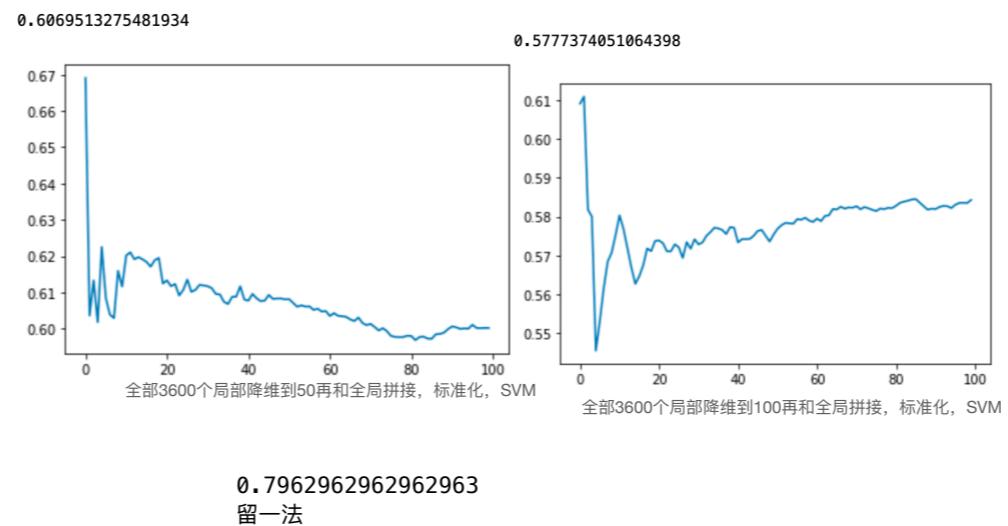


下面是使用了不同方法得到的HC和AD二分类的准确率，全部采用5折交叉验证，重复100次。

这两幅图是没有两周之前的结果，全部3千个特征可以达到0.75的准确率，以及用reliefF在全部特征范围内进行降维，得到的结果差不多。

这两周实现了用pca或reliefF先对每个局部特征降维，再和全局特征拼接起来。奇怪的是效果还不如之前的好。两者都只有0.65左右的准确率。

结果



然后我又试了在所有局部特征上降维到50维，结果更差。100维，更差。

所以我最好的结果就是HC和AD0.75准确率，LMCI和AD0.67，EMCI和AD0.57。

总结

- 缺乏对机器学习的系统学习
- 前期论文学习量不够
- 数据存储方法不当

缺乏对机器学习的系统学习，前期的时候一直以为越复杂的模型越好，想过用卷积神经网络，但是后来才知道对于小样本来说，过于复杂的模型会导致严重的过拟合，也就是训练集拟合的很好，在测试集中的泛化性能却很差，这导致浪费了大量时间。

后期的时候，了解到小样本机器学习的多种前处理方法，包括标准化、正则化、PCA、ReliefF等，还有SVM的多个核函数，还有交叉验证。导致我根本不知道先用哪个，再用哪个。

再加上是对单个局部特征处理再拼接还是先拼接再处理的选择，导致测试验证的排列组合数非常大，耗费大量时间改代码。

前期论文学习量不够。我从头到尾只参照了一篇论文的研究方法，虽然其中的某些步骤有些变化，但大差不差。而且这篇论文机器学习部分只提到了降维和正则化，然后就是我看不懂的概率论内容。总的来说，如果能多看几篇论文，应该就可以多了解一点实际应该怎么做的知识。

数据存储方法不当。前期的时候我以为算出来特征之后主要的工作就变成了手动选一组特征然后放到模型中得出准确率，重复个几十次，找出准确率最高的组合。于是我选择把特征数据全部存到数据库里，然后写了一个python脚本自动化选择特征，计算准确率再存下来。然而因为事实表明这个不是重点，所以这个脚本不仅没有给后面的工作带来便利，反而加大了组内同学交换数据和阅读代码的难度。

完