pmt 大作业实验报告

裴继辉

2019 年 8 月 11 日

1 核心思想说明

本项目的主要思路是单光子曲线划窗拟合，这受启发于第一阶段作业中波形的生成方法，即使用单光子曲线平移叠加的过程。现在主要考虑如何还原这一过程。由于时间谱是离散的 1029 个点，将单光子曲线沿时间轴平移，可以得到 1029 个起始于不同时刻（即光电子到达时刻）的单光子波形函数（下称函数），而实际波形可以看作是这 1029 个不同函数乘以 weight 后的线性叠加，然后再加入基准电压和白噪声。我们只要求出每个函数的叠加系数就好了。这里我寻找的是使得与实际波形均方距离最小的正的叠加系数。

2 代码整体结构与思路

2.1 单光子曲线与基准电压 (read.py)

按照核心思想中的说明，我们首先需要得到单光子曲线和基准电压，然后才能拟合。这并不难解决，由于提供了数据训练集，我们只需要从训练集中把所有单光子事例找到，然后把它们的起始时刻都平移到 0 再做平均就得到单光子曲线，而基准电压用单光子曲线中离起始时刻较远的点平均就能得到。这里的数学原理是独立的高斯随机变量相加仍满足高斯分布，若它们的方差都相同（同分布），则叠加后的方差要乘以相加变量个数 n，除以 n 平均之后的方差是原方差除以 n。如果高斯白噪声的标准差是 1，那么平均后的标准差就是 $1/\sqrt{n}$，程序中取训练集前 10 万行数据，差不多有 2500 个单光子事例，平均后标准差可以降至 0.02，足以满足需要。此部分功能由 read.py 程序完成。

2.2 拟合求权重 (finalfit.wl)

拟合求每个函数的叠加系数（权重），是程序的主体部分，使用 wolfram 脚本完成，关于此语言的配置和简单介绍请参见“配置说明”一节。这一部

分详细的心路历程在下一节介绍。这里需要注意的是必须要给权重加上全部大于 0 的限制，不然会得到拟合系数正负相间从而相消的荒谬结果。此外如果使用 1029 个待定系数和 1029 个拟合点（即实际波形上的点）参与拟合，则程序会又慢又不准确。这里的做法是先用寻峰法粗筛出光电子可能到达的时刻，然后用它们对应的函数进行拟合，而几乎为 0 的拟合点也是没有什么意思的，只需要那些不为 0 的点参与距离计算。

首先，先找到那些电压低于阈值的点，通过对波形的分析，这里取-6.5mV（基准电压已置为 0）为阈值。以低于阈值的点及周围相邻的一些点作为拟合点（左边取到-7，右边取到 +15）。拟合点已经确定，下面对参与拟合的函数对应的起始点进行第二轮筛选。本程序的寻峰法的第二轮筛选并非找波形极值，而是找波形二阶差分的低点。为什么要取二阶差分呢？请看下面的数组。它们是单光子曲线前二十个点（从 i=2 开始）的二阶差分 $a_{i+1} - a_i - a_{i-1} + a_{i-2}$ 的值

-0.229, -0.802, -1.973, -4.593, -5.645, 0.314, 8.459, 8.679, 3.82, 0.25, -1.857, -2.5, -1.407, -0.385, -0.217, -0.185, -0.347, -0.5, -0.35, -0.153

这里第 8 个数左右峰值十分清晰，但是代价是噪声的标准差增大到了 2（不使用 $a_{i+1} - 2a_i + a_{i-1}$ 形式的二阶差分也是基于噪声标准差的考虑）。取 1.5 作为二阶差分的阈值，取高于阈值的点及相邻的两点对应的函数（向前平移 9ns 后）作为参与拟合的函数。这样，两轮的筛选标准都比较低，极少有被筛掉的正确结果。

我们的目标是找最小均方距离。设有 m 个拟合点 $t_1, \ldots, t_m$，n 个参与拟合的函数（起始时刻为 $t'_1, \ldots, t'_m$）。对于每一个拟合点 t_k，实际值为 $R(t_k)$，拟合值为

$$w_{t'_1} \cdot S_{t'_1}(t_k) + w_{t'_2} \cdot S_{t'_2}(t_k) \cdots + w_{t'_n} \cdot S_{t'_n}(t_k)$$

其中 $S_{t'_i}$ 代表起始时刻为 t'_i 的单光子函数，$w_{t'_i}$ 是对应于 $S_{t'_i}$ 的待求权重

可以写成矩阵形式

$$\begin{pmatrix} S_{t'_1}(t_1) & \dots & S_{t'_n}(t_1) \\ \vdots & & \vdots \\ S_{t'_1}(t_m) & \dots & S_{t'_n}(t_m) \end{pmatrix} \begin{pmatrix} w_{t'_n} \\ \vdots \\ w_{t'_n} \end{pmatrix} \to \begin{pmatrix} R(t_1) \\ \vdots \\ R(t_m) \end{pmatrix}$$

均方根距离即为上式等号两边作差取模。利用 mathematica 中的 FindArgMin 函数，可以非常快速地完成求解其最小值点的过程。

事实上，大部分的权重的拟合结果都是非常接近 0 的，我们只取大于 0.1 的结果，记录对应的时刻和权重。

有些事例还是全部被筛掉了，一般是光子很少而波形幅度较小的事，这属于很少数。对于这些事例，我采用的方法是最简单寻峰法，记录最小值对应的 PETime。

在 mathematica 中直接输出结构化数组开销比较大，于是先输出列表，再用 python 读取转化。

2.3 对原始结果的读取与调制 (alltran.py、adjust.py)

由于上一节输出的结果不符合提交要求，且在实际运行时都是拆成一千多份文件分开跑，因此需要一个程序把它们的格式更改并合并。alltran.py 程序就是此用途。

如果将上一步得到的拟合系数（权重）和函数起始时刻直接作为 weight、PETime 提交，结果的 w 距离在 2.4 左右。但实际上这个成绩还可以进一步提高。如果将 weight 四舍五入取整则会更优一些（这比较好理解），如果把 0.5 之下的结果保留，剩余的取整，分数会更高，w 距离在 2 以内。因此尝试利用已有的结果进一步优化。在训练集上运行程序得到结果与答案比对就能发现，造成这一问题的主要原因是由于噪声的影响，可能会出现如下情况：相邻的三个点各分去了一部分权重，例如时刻 0,1,2,3,4 对应的拟合系数分别是 0,0.25,0.3,0.2,0，而真实的 PETime 是在时刻 2。针对这种情况，我进行了特殊处理，在 adjust.py 程序中完成。

先将所有权重舍入取整，然后用原权重减去之，处理剩下的零头。将其与左右两侧的值乘以不同系数 a,b,c 相加，如果结果超过 0.5 且比左右两侧的值要高则将此时刻补充权重 1。例如对于前面的例子 (0,0.25,0.3,0.2,0) 中，设左边、自身、右边的相加的系数 a,b,c（不是拟合系数）分别是 0.9,1.7,0.9，那么得到的结果是 (0.225,0.695,0.915,0.61,0.18)，最终就只会选出中间的 0.915 对应的 time。理论上也可以将小于-0.5 的结果的时刻权重扣掉 1，但这么做效果不好，且只增加原先拟合系数为 0 的时刻的权重效果才明显，因此最终此程序只对原先本应被舍掉的时刻追加权重。此番处理之后，效果出乎意料的地好。w 距离降到了 1.2 以内。

3 遇到的问题及解决方法

思路看起来很简单，但在实现的过程中还是遇到了不少的困难。首先就是拟合，最一开始，我使用全体 1029 个函数、1029 个实际电压值参与拟合，且不加限制范围。由于所求参数与方程个数相同（前面 2.2 式子中矩阵是方阵），可以直接解出所有权重，但得到的结果如同噪声一样杂乱无章；后来才明白是拟合系数总是正负相间，大部分峰值相抵消正好满足了所有点的要求。这里根本原因是高斯白噪声的影响，在这个问题中，一点点噪声就可以使解严重偏离原来的理想化模型。这让我想到了高次插值的龙格现象（虽然可能没什么联系），于是我利用设置阈值减少了函数数量，这回由直接解方程变成了用广义逆求最小二乘解，但是得到的仍然很差，还是有很多负数。

我差点认为这个方法行不通，后来才想到人为增加权重为正的要求。结果令人兴奋，但是新的问题又出来了。原来的问题是线性问题，只需要做矩阵乘法或者求逆运算就行，速度很快，但加了限制条件之后只能用求极值的一般的迭代方法，速度很慢。于是我又开始了和运行速度的斗争。首先此时我使用的是求全局最值的函数，我把它换成了求局部极值的函数，因为在这样的二次最优化问题中，局部极值和全局极值往往相同。这回速度提升不少，但是还有一个关键问题是有少数的波（十分之一左右）迭代非常

慢，时间甚至是其他波的二十倍，带慢了整体速度。但我发现，用不同的单光子曲线（用训练集取平均后稍有差别）去拟合可以让时间变正常，于是我限定了每个波的运行时间 0.3s，如果超过了就换另一单光子曲线。这回的速度终于让人看到跑完的希望了（差不多 120h）。进一步，我添加了利用二阶差分的第二轮筛选，并把拟合点也做了范围限制，时间又缩短了一半。后来，我又发现程序会越跑越慢，这可能是由于 mathematica 列表寻址时间随长度增加了，于是我把 1796736 行数据拆成 1797 个文件使用 make 分别跑，每次跑 1000 行数据。但即使这样，在我的 pc 机上也要跑 50 多小时。最后，我购买了华为云双核服务器（新人有 15 天的免费），我的计算能力大大增强，两个核一起跑差不多 8 个多小时就能跑完了，并且不占用 pc 资源，时间问题终于算是基本解决了。（后来我又买了 1 台，计算资源终于不用再愁了）

还有一个问题是发生在 wolfram 脚本中的，在第一次批量运行后，发现 1797 份文件中有约 1% 的文件没有跑出来（跑丢了）。这让我很苦恼，虽然能够把它们都找出来再重新跑一遍，但是这违反了一次性原则，复现原则也收到影响。而且非常奇怪的是，重新跑的时候，在一台服务器上就能跑出来，在另一台上就无论如何都跑不出来。反复测试了好久才明白是命名空间（wolfram 语言里叫上下文）被篡改了，导致在原来命名空间定义的变量找不到了。原来是 mathematica 有丰富的优化器，在执行 FindArgMin 函数时，可能会针对不同情况自动帮你调用一些程序包，比如 IPOPT 等，然后就鬼使神差地把命名空间改了，找到问题后，我在每次循环后都判断声明一次命名空间就解决问题了。

还有一个值得提起的乌龙就是 mathematica 里列表索引是从 1 开始，而测评程序是从 0 开始的，一开始没有注意，导致我最初的版本的所有数据结果都错了一位。直到我对比训练集答案和我的输出才发现这一点，修改之后 w 距离从 2.4 直接降到 1.6。

算起来其实我花在 2.3 节重新调制 weight 的精力是最多的，当时看了一些机器学习的内容，但是不知道怎么把 w 距离用在损失函数上，试了很久但只得作罢。后来又用 scipy 里迭代求最值函数，希望通过在训练集最小

化 w 距离来自动调参，试了一两种不同的参数拟合方式，发现最后的结果和之前 2.3 节的简单的三个参数的方法分数差不多，没有显著提高，也就没有再尝试了。可能这个结果就是重新调制拟合参数的极限了吧，因为有一些信息在 2.2 节的输出已经丢失了。但我认为此算法仍有不小进步空间，比如从一开始就避免 (0.25,0.3,0.2) 这种情况的出现，比之后再处理要有效。至少我认为 w 距离进步到 1 以内应该是没什么问题的。

4 配置说明

本项目的主体程序（finalfit.wl）使用了 mathematica（wolfram 语言）。其余部分均由 python 程序完成，所用到的第三方库包括 numpy，h5py，tables。mathematica 是一款科学计算软件，符号计算与数值计算均可胜任，在物理学领域使用较广泛；其使用的是 wolfram 语言，mathematica 软件是收费的，但 wolfram 语言内核是免费的。鉴于此语言并不是很常用，配置方法和简单介绍如下：

配置过程是免费的，但是需要连网认证 wolframID。劳烦助教进入官方网站（https://www.wolfram.com/engine/），注册一个 wolfram 账号并下载 wolfram 引擎 for Linux，之后 sudo 运行下载好的 shell 脚本安装即可。命令行运行 wolframscript 就会进入 wolfram 语言交互式环境，运行 wolframscript -file file.wl 则会运行脚本 file.wl。第一次使用时需要认证 ID，输入账号和密码即可（此账号须与下载时注册的账号相同）。wolfram 语言的特点是一切都是函数，包括循环控制。vscode 中有对 wolfram 语言的支持，而 wolfram 官网有非常详细的参考资料，每一个函数都有详尽的解释。考虑到此语言较小众，我在 finalfit.wl 程序中的注释也比较详细。

辛苦助教了

5 程序运行方式

可以直接使用 make 运行程序，运行 make 或 make all 运行以下所有步骤（除第五步）；运行 make medium 运行第一步；运行 make result 运行第二步；运行 make answer 运行第三、四步；运行 make clean 清除所有第二步的输出文件。

第一步需要运行 read.py 生成单光子波形，它接受一个命令行参数：0 或 1，指定所取的训练集，若为 0，取 atraining-0.h5，在/medium 下输出 singlewave1.h5 和 average1.h5;若为 1,取 atraining-1.h5,输出 singlewave2.h5 和 average2.h5。其中 singlewave1.h5 和 average1.h5 在后面是必要的。所以 python3 read.py 0 必须运行，而 read.py 1 可以不运行（singlewave2.h5 很少会用到）。pc 上运行一次时间在 20min 左右（运行过程中有进度提示）。

第二步需要运行 finalfit.wl，它接受一个命令行数字参数 (1 1797)。使用 wolframscript -file finalfit.wl n 运行，n 指定计算测试集从 (n-1)*1000 到 n*1000-1 行 (n=1797 时是最后一行) 的数据，输出/result/n-pgan.h5。原则上全部运行后才能进行后面的部分。此程序运行较慢，运行全部 1797 个参数在 pc 上需要超过 24h（运行时每处理 100 行数据输出一次提示）。

第三步直接运行 alltran.py,将之前所有编号的输出结果合并,输出/result/Total.h5。运行速度很快。

第四步直接运行 adjust.py，输出最终结果 finalanswer.h5。在 pc 机上大约需要运行 10min（运行时每 10000 组输出一次提示，共约 1797000 组）。

可选：第五步直接运行 compress.py 可以将第四步输出的 hdf5 压缩输出 compressedanswer,h5