利用 WAM 预测真核生物剪接位点

（生信基地 1801，徐伟，U201812207）

摘要：在真核生物基因转录过程中会发生 mRNA 的剪接，其剪切位点在不同基因组和不同 mRNA 中都具有保守性。通过已有的实验已经证实剪接位点中的 donor 位点后通常是 GT 序列碱基，因此我们根据这个特征利用算法来识别剪接位点。目前已有许多机器学习的算法来进行位点识别，如支持向量机 SVM，贝叶斯网络，WMM，WAM 算法。本文通过 WAM（Weight Array Model）算法预测真核生物剪接位点中的 donor 位点，并通过 python 实现。利用 KR set（Kulp & Reese)作为训练集，BG set（Burset & Guigo）作为测试集，对模型进行训练和检验。

关键词：真核生物剪接位点预测，donor 位点，机器学习，WAM 算法

1. 引言

真核生物剪接发生在基因转录的过程中，在许多真核细胞中，基因的编码序列被一段段非编码序列隔开，这些割裂基因的编码序列被称为外显子，而非编码序列被称为内含子，在多细胞生物中，例如人类中，有内含子的基因数目非常多，每个基因所含的内含子也很多（最多的情况下可以达到 362 个）。当一个有内含子的基因转录时，RNA 最初包括了这些内含子。这些内含子被除去以得到成熟的 mRNA，内含子被除去的过程称为剪接。内含子和外显子边界的序列可以帮助细胞识别内含子并将其切除。这些剪接序列大多数存在于内含子中。这些序列根据它们相对于内含子末端的位置被称为 3’剪接位点（acceptor 位点）和 5’剪接位点（donor 位点）[1]。

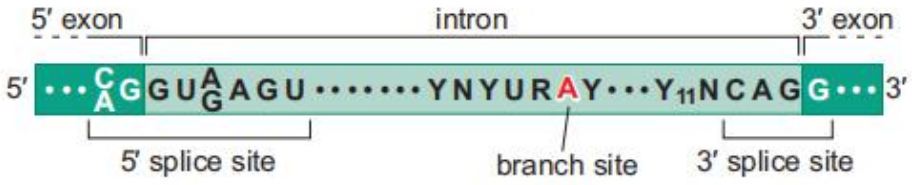


图 1，mRNA 序列内含子与外显子边界。这里是 mRNA 序列，所以 5’剪切位点，即 donor 位点边界为 GU,与我们已知的基因序列为 GT 相同。他们即为 donor 位点的共有序列（consensus sequences）[2]

本文利用 donor 位点的共有序列，通过机器学习算法 WAM 提取其附近序列特征，从而用于真核基因剪接位点预测。WAM 算法是基于贝叶斯的一种方法，相对于 WMM（Weight Matrix Model），考虑了相邻碱基之间的相关性。

2. 目标

本次上机实操目的为通过 python 实现 WAM 算法预测真核剪切位点，从而加深对机器学习和 WAM 算法的理解，通过其预测性能对 WAM 算法进行性能评估，在后续与其他算法进行比较，学习了解 WAM 算法的优劣势，并在实践的过程中锻炼编程能力。

3. 假设

我们已知 donor 位点边界是由 GT 碱基序列组成，但由于碱基序列非常长，每两个碱基中出现 GT 碱基的概率（$P = \frac{1}{4} * \frac{1}{4} = \frac{1}{16}$），对于真核生物碱基序列长度在 Mb 以上数量来说，出现 GT 碱基的数量会非常高，所以仅依此来判断是否为剪接位点并不可靠，会出现非常多的假阳性。同时对于生物中剪接序列的蛋白质来说，由于序列特别长，仅仅只识别 GT 序列不足以正确切割外显子与内含子。所以我们考虑到 donor 位点附近的碱基，把它们纳入 donor 位点特征中，用一条更长的序列来进行预测，而他们之间一定存在某种相关性，使剪接序列具有特异性而能被蛋白质识别。在本文的 WAM 模型中，我们假设，剪接位点附近序列中相邻碱基具有相关性，而非相邻碱基不具有相关性。

4. 算法实现过程

4.1 WAM 算法原理

WAM 算法是贝叶斯的一种简化版本，由于它只考虑了相邻碱基的相关性，所以它是介于朴素贝叶斯（认为特征之间独立）与理想贝叶斯（考虑所有特征之间相关性）之间的一种贝叶斯方法，它是一个二分类问题，即是或不是剪接位点，其判别函数为：

$$S(\mathrm{X}) = \ln(P^{+}(\mathrm{X}) / P^{-}(\mathrm{X}))$$

$$= \ln \frac{p^{+}(1, \mathrm{x}_1)}{p^{-}(1, \mathrm{x}_1)} + \sum_{i=2}^{\lambda} \ln \frac{p^{+}(i, \mathrm{x}_{i-1}, \mathrm{x}_i)}{p^{-}(i, \mathrm{x}_{i-1}, \mathrm{x}_i)}$$

其中，P^{+}（X）代表是剪接位点的后验概率，P^{-}（X）代表是背景序列（即普通序列）的概率

$$P(X) = p(1, x_1)\, p(2, x_1, x_2)\, p(3, x_2, x_3) \ldots p(\lambda, x_{\lambda-1}, x_\lambda)$$

其中，P（1，X1）代表第一位是 X1 碱基的概率，P（2，X1，X2）代表第一位是 X1 碱基的条件下，第二位是 X2 碱基的概率。

4.2 建模过程

4.2.1 提取序列

在 WAM 算法中，我们提取的特征是 donor 位点的前 3 个碱基与后 4 个碱基，共 9 个碱基序列，在训练集中我们已知外显子位置，据此来得到 donor 位点附近的序列。判断的方法为根据给出的序列外显子的范围，取外显子范围的后一个数，这个位置的后两个碱基即为 GT 序列，但要注意最后一个外显子后不需要剪接。

取完序列后剪接位点序列个数为 2381。

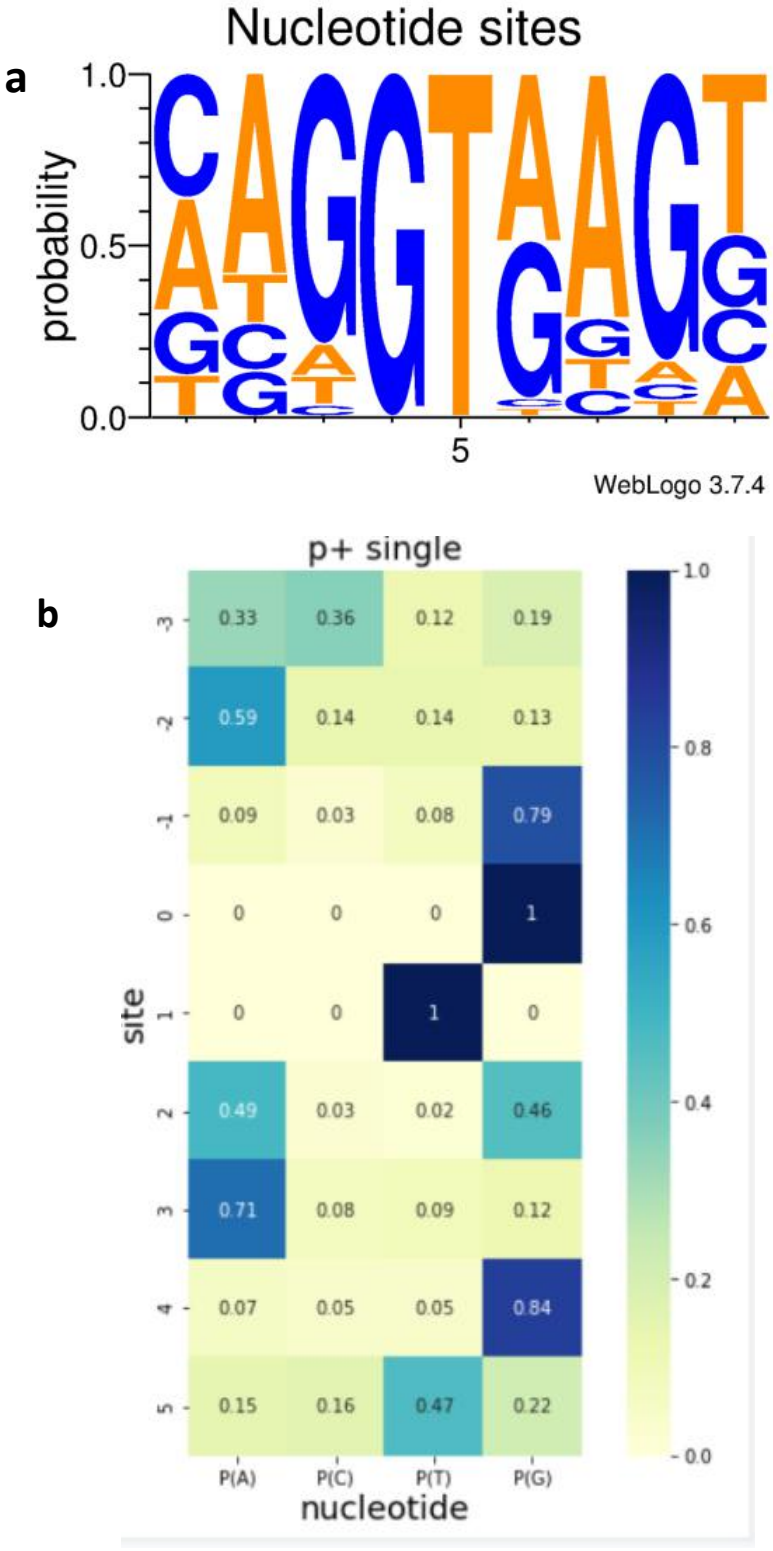


图 2，剪接位点序列特征。a，横坐标代表阳性样本序列位置，纵坐标代表出现特定碱基的概率。在第 4，5 位置出现 GT 的概率都为 1。**b，**横坐标代表碱基概率，纵坐标代表碱基位点，第 0 位都是 G，而第 1 位都是 T。观察其他位点碱基概率，发现并不都近似等于 0.25，证明碱基之间确实存在相关性。

由于所有训练集有 462 个文件，如果取所有的训练集来取所有的非剪接位点序列，会非常多序列，而且由于非剪接位点序列碱基之间没有相关性，这样做意义也不大。所以本文非剪接位点序列取的是第一个训练集中的数据，每次滑动一个碱基，得到共 35855 条序列。

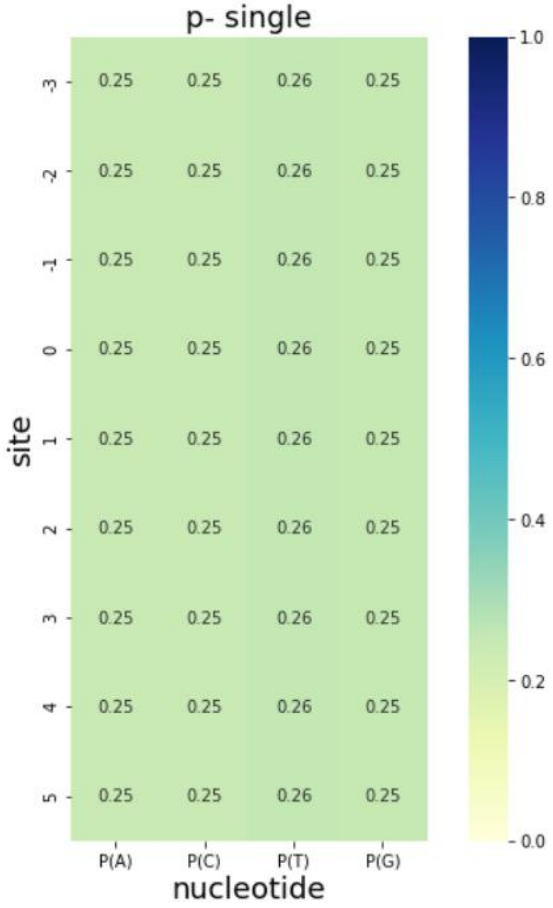


图 3，非剪接位点序列特征热图。横坐标代表碱基概率，纵坐标代表位点。各位点 ACTG 碱基的概率都基本近似等于 0.25，符合预期。

4.2.2 计算相邻碱基条件概率

计算 P（i，X1，X2），即当第 i-1 位是碱基 X1 的条件下，第 i 位出现碱基 X2 的概率，由于共 9 个碱基，而第一个碱基没有条件概率，所以共有 8 个位置的条件概率，而一个位置的条件概率有 4*4=16 中情况，所以计算得到的条件概率应该是一个 16*8 的矩阵。

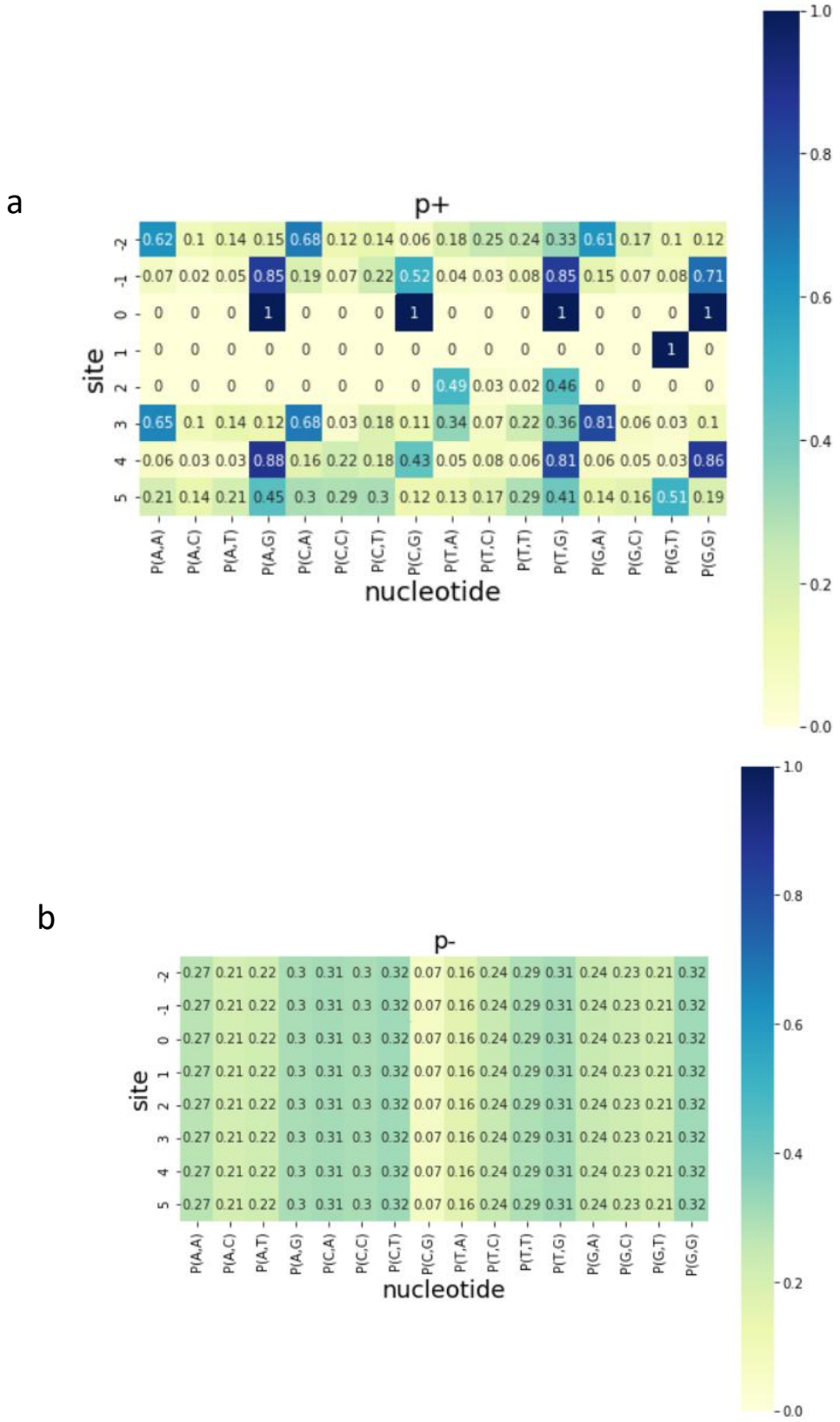


图 4，相邻碱基条件概率热图。a，剪切位点条件概率。由热图明显可以看到在每个位置基本上都有概率比较高的条件概率，也证明了相邻碱基之间确实存在相关性。b，非剪切位点条件概率。非剪接位点序列相邻碱基也存在一些相关性，但没有碱基序列之间那么明显。

4.2.3 计算判别函数 S（X）

判别函数的计算公式为：

$$S(\mathrm{X}) = \ln(P^{+}(\mathrm{X}) / P^{-}(\mathrm{X}))$$

$$= \ln \frac{p^{+}(1, \mathrm{x}_1)}{p^{-}(1, \mathrm{x}_1)} + \sum_{i=2}^{\lambda} \ln \frac{p^{+}(i, \mathrm{x}_{i-1}, \mathrm{x}_i)}{p^{-}(i, \mathrm{x}_{i-1}, \mathrm{x}_i)}$$

获得了剪接位点与非剪接位点第一个位置的碱基概率和后面位置的条件概率后即可以根据此训练集定义得分函数。

对测试集的序列进行打分，测试集阳性样本共 2079 条，对测试集的阴性阳本选取由两种方法，一是选取所有测试集中的非碱基序列，共 2883625 条。二是选取非碱基序列中第三、四位也是 GT 碱基的序列，共 149206 条。为了得知阴性样本选取的区别，我两种数据集都取了并对应作图观察区别。第一种方案为 dataset1，第二种方案为 dataset2。

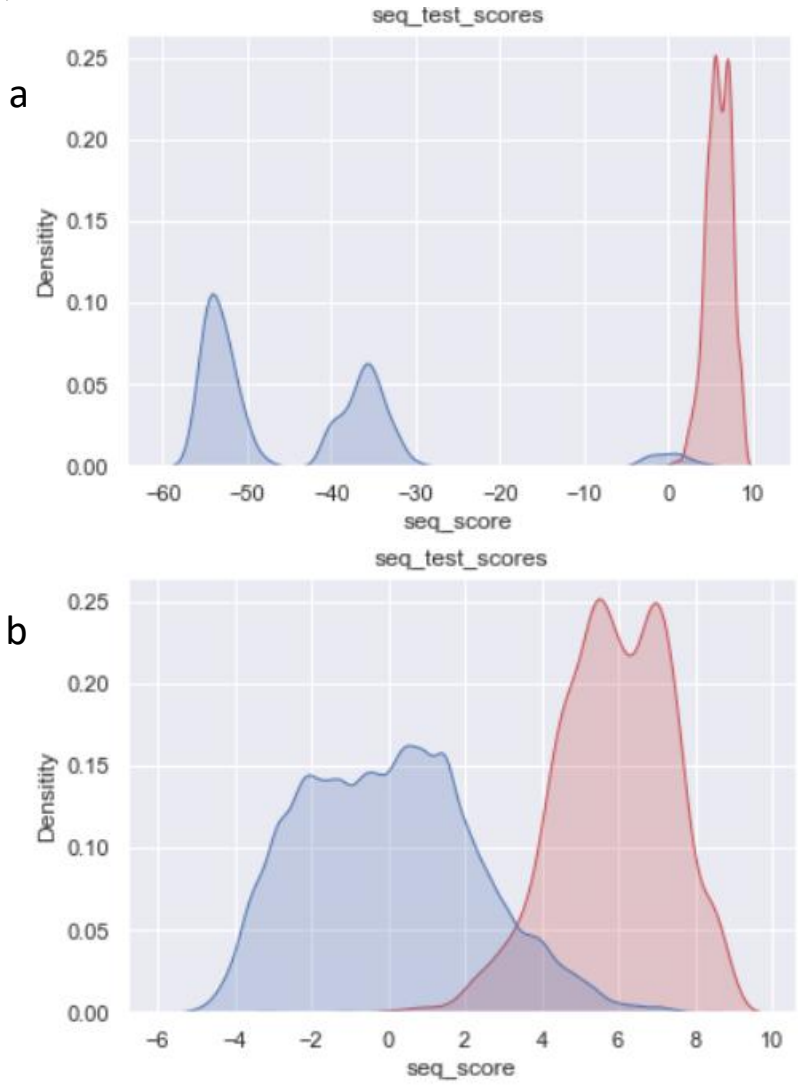


图 5 测试集得分密度图。a，dataset1，红色代表阳性，蓝色代表阴性。可以看到因为数据量很多，阳性样本和阴性样本重合的分数密度并不高。**b，**dataset2，红色代表阳性，蓝色代表阴性，阳性和阴性数据得分有重合的地方，所以需要选取合适的阈值将其分开。

4.2.4 利用判别函数和测试集对算法进行性能测试

判别函数，即得分函数的作用在于输入一条序列，根据得分函数算出分数，如果分数高于我们设定的阈值，则判断其为剪接位点序列，反之则判断其为非剪接位点序列。所以我们要设定不同的阈值，并算出相应的判断算法性能的参数，来判断这个阈值是否是最合理的。

最初的想法是在[-10,10]之间每隔 0.1 取一个阈值来计算 TN,TP,FP,FN 等参数，但这个计算过程耗费了 4 个小时，且得到的 ROC 曲线并不好。所以在寻找资料后利用了函数可以直接计算 fpr，tpr，阈值等参数，且能设定所有可能的阈值范围，计算也十分地快，代码如下：

```
from sklearn.metrics import average_precision_score
from sklearn.metrics import precision_recall_curve
fpr,tpr,threshold = roc_curve(y_lable, y_score) ###计算真正率和假正率
precision, recall, thresholds = precision_recall_curve(y_lable, y_score)
```

4.2.5 算法性能判断

4.2.5.1

分类器评估指标有：TP,TN,FP,FN。

真正例（True Positive，TP）：真实类别为真，预测类别为真。

假正例（False Positive，FP）：真实类别为负，预测类别为真。

假负例（False Negative，FN）：真实类别为真，预测类别为负。

真负例（True Negative，TN）：真实类别为负，预测类别为负。

4.2.5.2

对算法性能判断一般有两种图。

第一种是 PR 图，P 代表 Precision，准确度，R 代表 Recall，召回率，也就是我们课上讲的 Sp-Sn 曲线。

$$P=Sp=\frac{TP}{TP+FP} \qquad R=Sn=\frac{TP}{TP+FN}$$

我们根据上面的函数可以直接算得所有可能阈值下的 Precision 和 recall。

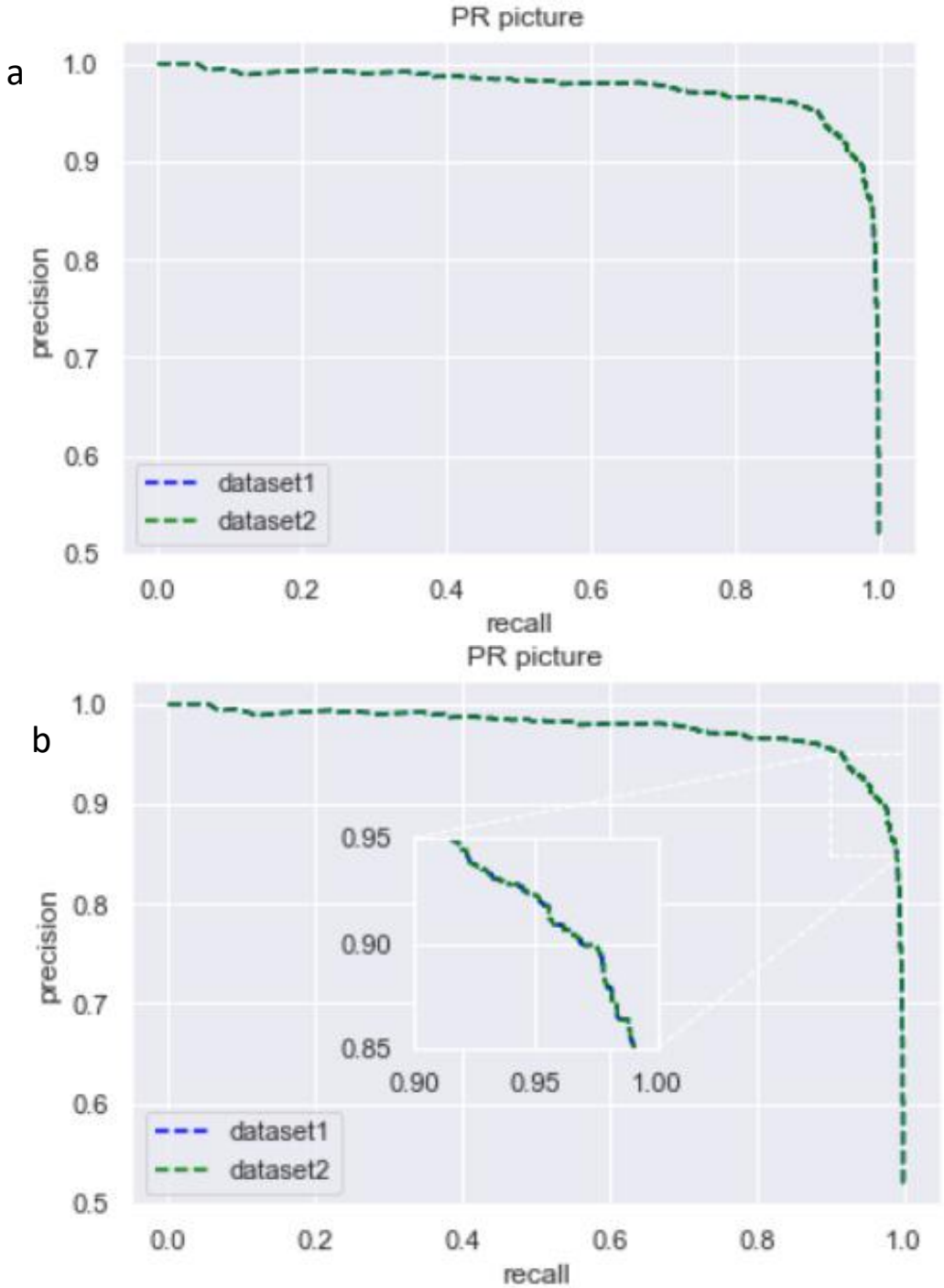


图 6，PR 图。a，两种测试集的 PR 图。两种测试集的 PR 图完全重叠，说明两种测试集对 PR 图没有任何影响。WAM 模型 PR 图并不算好，但由于没有与其他的算法进行比较，也无法比较其性能。**b，带特定位置子图的 PR 图。**由子图清晰看出两条曲线完全重叠。

由这种 PR 图我们无法得知具体什么阈值下算法性能更好，所以还需要画阈值与 PR 图之间的曲线图更直观地表现算法。

$$\text{F1-score} = \frac{2*recall*precision}{recall+precision}$$

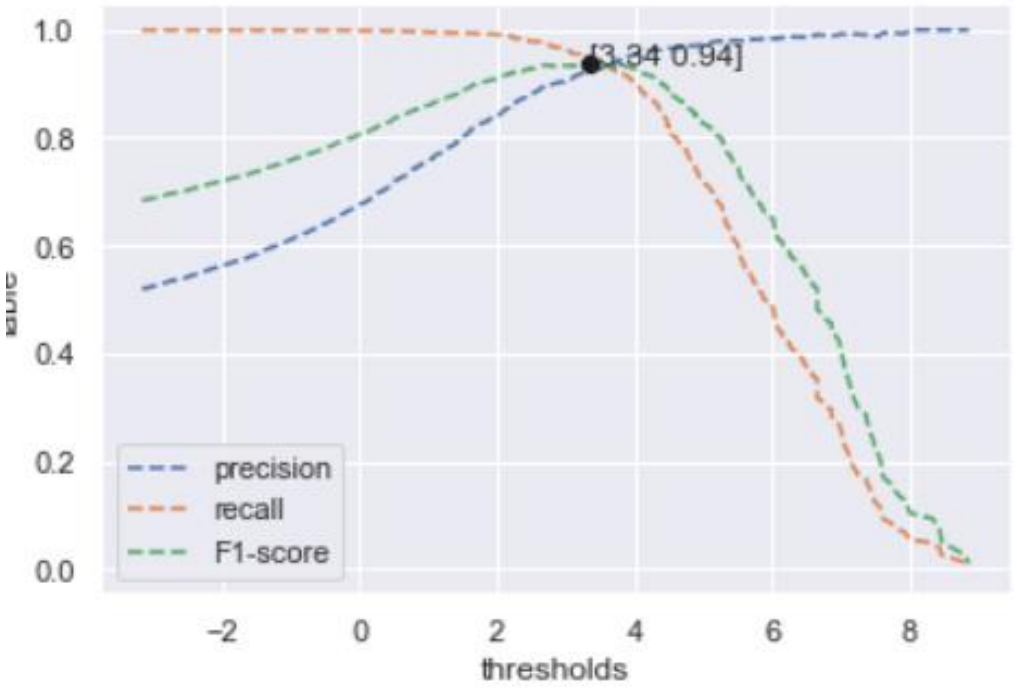


图 7，阈值与 PR 的关系图。可以清晰的看到随着阈值的增大，召回率即灵敏性 Sn 在下降，而准确度即特异性 Sp 在上升，这符合我们的预期。F1-score 曲线用来判断综合 Sn 和 Sp 的表现的一个指标，F1-score 曲线的最高点即为在这个训练集与测试集下，综合 Sn 和 Sp 的表现最优的阈值点。这里标出阈值最优点为 3.34。经过比较，发现利用 dataset1 和 dataset2 的此图表现完全一致。

4.2.5.3

第二种判断算法性能的是 ROC 图，同样所有阈值下的 TPR（true positive rate）和 FPR（false positive rate）可以由上述代码直接算出。

$$\mathrm{TPR} = \frac{TP}{TP+FN} \qquad \mathrm{FPR} = \frac{FP}{TN+FP}$$

在 ROC 图种还有一个指标，AUC，指的 ROC 曲线下方的面积。越接近 1 表示分类器越好。计算 AUC 的值也有上述代码中函数直接计算得到。

随着 FPR 的上升，ROC 曲线从原点(0, 0)出发，最终都会落到(1, 1)点。三种 AUC 的值分别代表的意思[3]：

①AUC = 1，是完美分类器，采用这个预测模型时，不管设定什么阈值都能得出完美预测。绝大多数预测的场合，不存在完美分类器。

②0.5 < AUC < 1，优于随机猜测。这个分类器（模型）妥善设定阈值的话，能有预测价值。

③AUC = 0.5，跟随机猜测一样（例：丢硬币），模型没有预测价值。

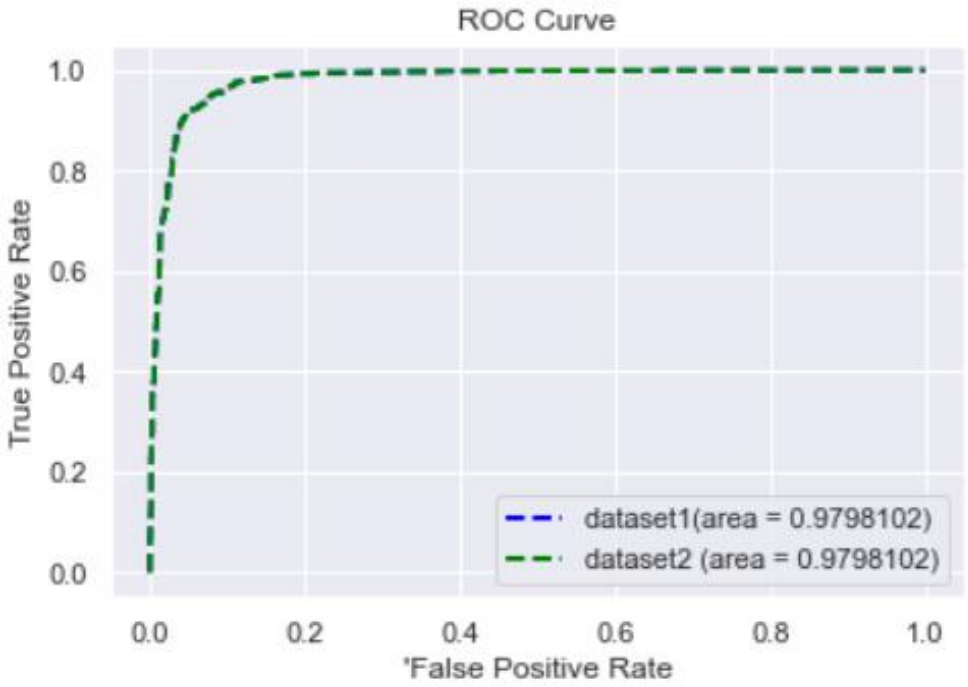
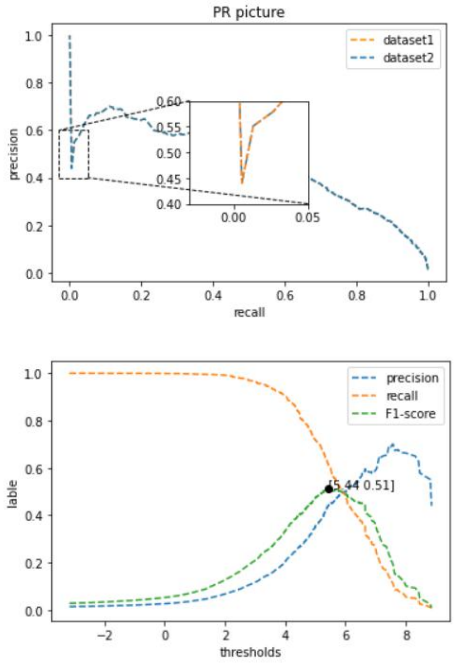


图 8，ROC 图。两条曲线分别对应 dataset1 和 dataset2。在这计算出的 ROC 图的 AUC 都很高，dataset1 和 dataset2 曲线完全重合，预测性能都接近 0.98，预测性能良好。

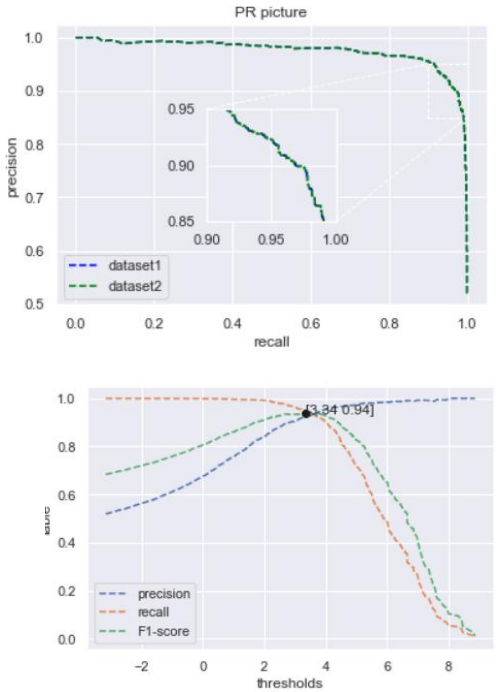
5. 平衡数据对曲线的影响

由于测试集中阳性样本只有 2079 条，而阴性样本有几十万甚至两百多万条，上述的实现过程中平衡了阳性阴性样本，随机阴性样本使其达到 1：1 的平衡，在，我也做了不平衡的样本进行性能测试。

a



b



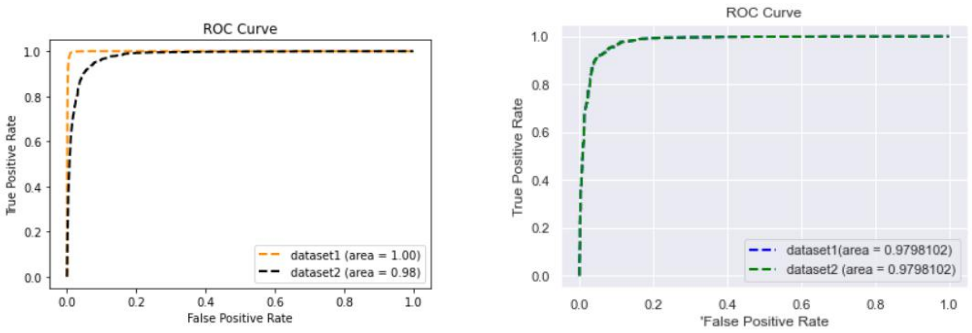


图 9，WAM 算法性能比较。a，这一列为没有平衡测试集的性能。**b，**这一列为平衡了测试集的性能。可以看到平衡与否产出的图会有所不同，用 ROC 图来说明，未平衡时 dataset1 的 roc 性能接近于 1，而 dataset2 的性能接近 0.98，这是因为 dataset1 中的阴性样本第 4、5 位不是 GT 序列，所以与阳性样本相差较大，比较好预测。

6. 讨论

5.1 算法预测性能较好

本次上机所用的 WAM 算法预测性能良好,AUC 分类接近于 1，但 WAM 算法本身并不算好，性能应该介于朴素贝叶斯和理想贝叶斯之间，但能达到这么好的预测效果是由剪接位点本身的性质得到的。Donor 位点保守性很强，根据数据集可得所有的训练集的第三、第四位点碱基都是 GT，所以特征性很强。据此得到的判别函数判别性很强，能去除很多假阳性。

5.2 算法性能还有很多提升的空间

本次 WAM 算法所用的训练集虽然有两种，即随机序列与第三、四位为 GT 碱基的序列，阴性样本随机序列有 200 多万条，与阳性样本 2000 条有三个数量级的差异，即使取与阳性样本相似的序列 dataset2，也有 14 万多条序列，也与阳性样本有两个数量级差异。所以测试集中阳性样本和阴性样本并不匹配，这个对于算法性能应该有些影响，但没有进一步测试下去。

同时，WAM 算法本身只考虑了相邻碱基之间的相关性，而假设非相邻碱基之间都是独立的，这是为了计算方便，但后续可以继续做相关工作，考虑非相邻碱基之间的关系，进一步提升剪接位点预测算法的性能。

同时，对 dataset1 和 dataset2 的曲线重合保持一定的疑问。两种测试集特征不相同，在确认代码过程没有出错的情况下，两者的 fpr、tpr、阈值等参数维度、数值相同，对此保持疑问，后续还需要进一步解决并找出其原因。

5.3 WAM 算法的优化可能

首先在本文中利用的判别函数有训练集中阴性样本的概率值计算，而本文也确实这么做了，但其实可以不需要算阴性样本的概率值，因为在定义打分函数时每个序列得分都需要减去相同的阴性样本概率值，所以测试集中阳性样本和阴性样本减去值就相当于每减，没有区别，所以其实不需要算阴性样本的概率。

在其他方面，由于 WAM 算法本身比较简单，除了在提取训练集与测试集时比较消耗时间，其余的打分预测时间全部内容运行完的时间都在 1 分钟以内，所以 WAM 算法本身并没有什么值得优化的。可以通过将需要的训练集、测试集导入到

电脑 txt 文件中，便于后续每次调整算法与参数，以达到调整的目的。

5.3 WAM 算法与 SVM 算法、贝叶斯网络算法的比较

在做完 SVM 算法和贝叶斯网络算法之后，将三者用同一数据集进行训练，比较其效果。

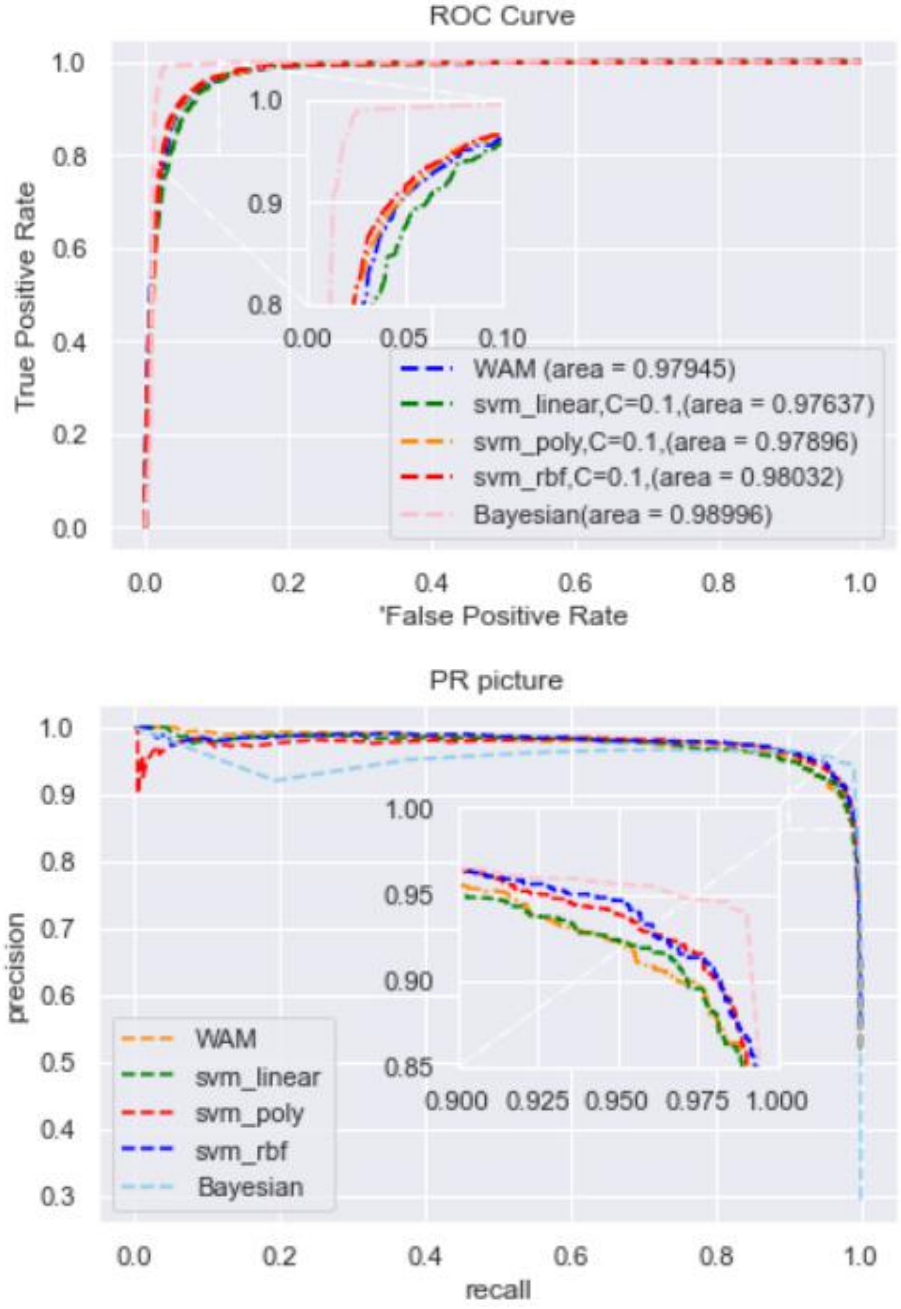


图 10 所有算法的性能比较。

将 WAM 算法与 SVM 算法、贝叶斯网络算法进行比较，发现三种算法性能都大致相同，都在 0.98 左右及以上，属于性能比较好的算法了。同时，WAM 算法运行速度快，虽然假设比较简单，只考虑了相邻位点之间的关系，而贝叶斯网络考虑了所有位点之间的关系，所以 WAM 算法性能比贝叶斯网络性能差一点是在意料之中的。但 WAM 达到 0.98 左右已经很不错，算是一种比较好的预测真核生物剪接位点的算法。

7. 致谢与感悟

感谢《生信数据挖掘》课程老师周艳红老师布置了这次上机任务，让我自己有动力真正写一个完整的小型机器学习的代码，虽然这个模型不难，算法思路也很清晰，但真正写一个完整代码对于我的水平还是比较困难，在写这个代码的过程中也遇到了很多的问题，通过慢慢的查资料逐一解决后还是比较开心的。通过这次上机也对 WAM 算法、分类实现过程有了更深刻的感悟，学习到了很多知识，稍微提高了自己的编程水平。

8. 数据获取

本文算法实现的完整代码、产出图片与文档的电子档可在 github 上获取。获取链接如下：
https://github.com/xuweialan/Splicing-site-prediction/tree/main/WAM

9. 参考资料：

1.分子生物学书本 Molecular+Biology+of+the+Gene+(7th+Edition) P505
2.分子生物学书本 Molecular+Biology+of+the+Gene+(7th+Edition) P469
3.网页，https://www.cnblogs.com/larack/p/11023404.html