中国科学：信息科学

SCIENTIA SINICA Informationis

探索搜索、优化策略在学习任务降维中的应用

陈张天艺[1]

1. 暨南大学信息科学技术学院, 广州

E-mail: czty19@gmail.com

摘要 搜索算法是解决搜索问题的任何算法，即检索存储在某个数据结构中的信息，优化问题是指在满足一定条件下，在众多方案或参数值中寻找最优方案或参数值，以使得某个或多个功能指标达到最优。在本文中，笔者从解决特征降维任务的角度出发，从这两种不同的角度测试了多种算法。与此同时，我在两种不同的数据集上进行了实验，对不同的算法进行性能测试。

关键词 优化算法，搜索算法，特征降维，机器学习

1 引言

在机器学习和统计学中，特征降维，也称为特征选择、属性选择或变量子集选择，是选择相关特征（变量、预测变量）子集用于模型构建的过程。使用特征选择技术的中心前提是数据包含一些冗余或不相关的特征，因此可以在不损失太多信息的情况下将其删除 [1]。冗余和不相关是两个截然不同的概念，因为一个相关特征在另一个与其密切相关的相关特征存在的情况下可能是冗余的 [2]。特征选择算法可以看作是用于提出新特征子集的搜索技术与对不同特征子集进行评分的评估措施的组合。最简单的算法是测试每个可能的特征子集，找到最小化错误率的特征。这是对空间的详尽搜索，除了最小的特征集外，所有的特征集在计算上都是难以处理的。

在本文中，我们将从两个方面的算法——搜索算法和优化算法来看待降维任务。在计算机科学中，搜索算法是解决搜索问题的任何算法，即检索存储在某个数据结构中的信息，或者在问题域的搜索空间中计算的信息。这种结构的例子包括但不限于链表，数组数据结构或搜索树。合适的搜索算法通常取决于正在搜索的数据结构，并且还可能包括有关数据的先前知识。对于搜索算法，我们考虑使用递归特征消除算法来解决特征降维的问题，并将其与前向搜索算法和后向搜索算法在不同的数据集上进行性能对比。

优化问题是指在满足一定条件下，在众多方案或参数值中寻找最优方案或参数值，以使得某个或多个功能指标达到最优，或使系统的某些性能指标达到最大值或最小值。传统的优化算法通过遍历整个搜索空间来找到最佳方案，但在实际应用中由于搜索空间不断变得复杂和庞大，这种算法容易耗费大量的时间，造成“组合爆炸”。如今，对于复杂优化问题，一系列更高效的优化问题被各位学者陆续提出，如: 模仿自然界生物进化机制的遗传算法；模拟蚂蚁集体寻径行为的蚁群算法；模拟鸟群和鱼群群体行为的粒子群算法；源于固体物质退火过程的模拟退火算法等等。本文中从解决特征降维任务的角度出发，将优化算法应用到该领域，并对其进行性能测试。

2 从搜索策略角度看待降维任务

2.1 描述搜索问题

将降维问题看作搜索问题后，设数据集为 R，其有 n 个样本，记为 $\{S_1, S_2, ..., S_n\}$，d 个特征，记为 $\{f_1, f_2, ..., f_n\}$。算法不断进行特征子集的选择，因此该问题空间的状态集合为数据集的全部子集集合，大小为 2^n-1，同时动作集合可描述为 $V = \{IN(jointheset), NOTIN(notjointheset)\}$。在本文用到的算法中，初始状态可分为两种，正向选择情况下为 $\varnothing$，而后向选择情况下则为 R。目标检测是与评价函数相关的，一般是一个阈值，当评价函数值达到这个阈值后就可停止搜索，在本文中我们一致使用随进森林算法作为评价函数。路径代价在特征降维任务中稍显不同，但总的来说，可将其定义为搜索单个特征所耗费的计算时间和带来的噪声（影响分类精度）：$Cost = Cost_{time} + Cost_{noise}$。

2.2 结合随机森林的递归特征消除（RF-RFE）

递归消除特征法（RFE）使用一个机器学习模型来进行多轮训练，每轮训练结束后，消除若干权值系数对应的特征，再基于新的特征集进行下一轮训练，通过不断搜索找到最佳的特征子集，从而达到降低特征维度和消除数据冗余、噪音的效果。在评估模型时，我使用了随机森林算法（RF）来对选择的特征进行分类预测，从而保证评估的稳定性。给出相应算法如算法1所示。

算法 1 结合随机森林的递归特征消除

输入: 数据集 T; n 个样本，d 维特征 $G = f_1, f_2, ..., f_d$，每个样本隶属于 L 标记，排序算法 $M(T, G)$，输出特征维数 p

主迭代:

```
for i=1 to n do
    Rank set F using M(T,G)
    f*<=last ranked feature in F
    R(d-i+1)<=f*
    F<=F-f*
    if sizeof(R)>p then
        output R
    end if
end for
```

输出: Final rank R

2.3 序列前向搜索（SFS）和序列后向搜索（SBS）

SBS 算法背后的思想非常简单：SBS 依次从完整的特征子集中删除特征，直到新的特征子空间包含所需数量的特征。为了确定在每个阶段要去除哪个特征，我们需要定义我们想要最小化的准则函数 J。准则函数计算的准则可以简单地为去除前后分类器的性能差异的一个特定功能。那么，每个阶段要去除的特征可以简单地定义为使这个准则最大化的特征；或者更直观地说，在每个阶段我们去除去除后性能损失最小的特征。而类似的，SFS 算法每次都选择一个使得评价函数的取值达到最优的特征加入特征子集，是一种简单的贪心算法。这里给出 SFS 算法的算法流程图：

算法 2 序列前向搜索

输入: 空集 $Y = \{\varnothing\}$

主迭代:

1: 贪心选择下一个最佳特征 $x^+ = \arg\min_{x \notin Y_k} [J(Y_k + x)]$.

2: 更新 $Y_{k+1} = Y_k + x^+$

3: Back to 2

输出: 最佳特征子集 Y

3 从优化策略角度看待降维任务

3.1 优化算法解决特征降维

将特征选择问题框架为优化问题。在输入特征很少的情况下，可以评估输入特征的所有可能组合，并确定地找到最佳子集。在输入特征数量众多的情况下，可以使用随机优化算法来探索搜索空间并找到特征的有效子集。

3.2 基于随机爬山算法的优化算法

首先，我们定义目标函数。它将数据集和特征子集用作输入，并返回估计的模型准确度，范围从 0（最差）到 1（最佳）。然后，我们在搜索空间定义一个更新函数。给定一个特征子集，更新函数对其进行修改并返回一个更优子集。在这种情况下，我们将通过随机翻转子序列中列的包含/排除来实现此目的。序列中的每个位置都将被独立考虑，并且在翻转的概率为超参数的情况下，概率将被翻转。具体实现如算法 3 所示

算法 3 随机爬山算法

输入: 随机生成特征子集，迭代次数 n

主迭代:

1: **for** i=1 to n **do**

2: 目标函数 $Objective(\cdot)$ 对上一节序列和模型评估步骤解码

3: 更新函数 $mutate(\cdot)$ 随机反转子序列

4: **end for**

输出: 最佳特征子集 Y

3.3 粒子群优化算法解决特征降维

许多优化问题设置在一个特征空间，该空间中的变量是离散的，具有定性的差异以及量级差异。不论是离散或连续的问题，都可以用二进制符号表示。一个优化器操作二值函数可能是有利的。在粒子群算法中，定义目标函数时，需要将每个粒子的值带入模型中训练，然后返回模型的指标值，通过操纵粒子的每个坐标来调整轨迹。种群中每个粒子相当于解空间中的一个解，粒子具有速度和位置两个属性，位置向量表示该粒子对应的解，速度向量则是为了调整粒子下一次飞行，从而进行位置更新搜索新的解集。具体算法可见论文 [3]。

在这里，我们将具体描述如何利用二进制粒子群优化算法（BPOS）进行特征降维。首先定义粒子 p_i 的位置 $X_i = (x_{i1}, x_{i2}, ..., x_{in})$ 可以表示为一个特征子集，其中 x_{ij} 表示为一个特征，当 x_{ij} 的

取值为 1 时表示该特征被选中，取值为 0 则表示未选中。具体实现思路如下算法 4 所示：

算法 4 二进制粒子群算法

输入: 数据集 T;n 个样本，d 维特征 $G = f_1, f_2, ..., f_d$，每个样本隶属于 L 标记，预定义适应度函数 $f(\cdot)$，最大迭代次数 T_{max}

主迭代: 随机初始化粒子的位置 x 和全局最佳位置 $gbest$，定义个体最优位置 $pbest$

```
for i=1 to T_max do
    for j=1 to d do
        计算每个粒子的适应度 f(x_i)
        if f(x_i) < f(pbest_i) then
            pbest_i = x_i
            f(pbest_i) = f(x_i)
        end if
    end for
    更新全局最优位置 gbest
end for
```

输出: 最佳特征子集，即全剧最佳位置 $gbest$

4 实验及结果分析

4.1 数据集及实验环境

所有实验均在 MacOS 中的 Apple M2 中央处理器 (CPU) 设备上使用 pycharm CE 软件完成。

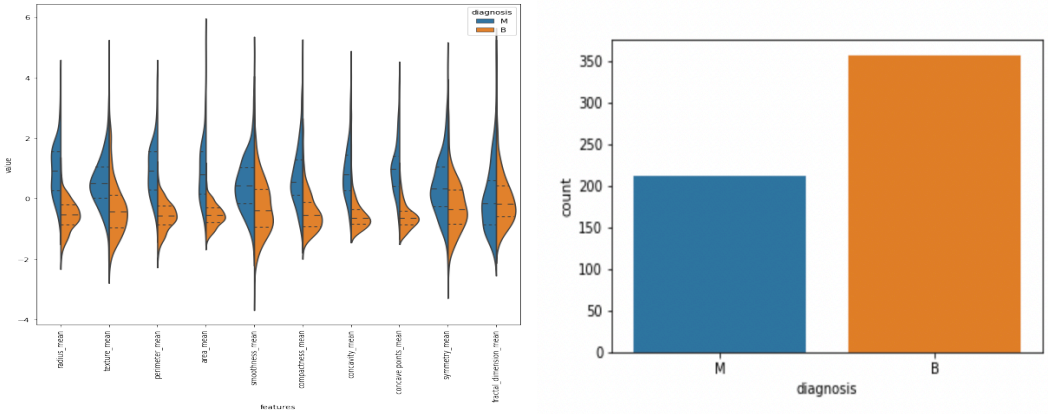

图 1 数据整体分布和每类样本数量

数据集一：威斯康星州乳腺癌（诊断）数据集是一个公开数据集，共有 569 位受试者，每位受试者用 id 标识并有 30 位特征，特征是根据乳房肿块的细针穿刺 (FNA) 的数字化图像计算得出的，它们描述了图像中存在的细胞核的特征，同时用 'M'，'B' 标记受试者的病情恶化或良好。恶性样本数量 212，良性样本数量为 357 如图一右所示，部分特征的数值分布图如图一左所示，例如，$texture_{mean}$ 特征中，良性和恶性的中位数是分开的，因此有利于分类。然而，在 $fractal\_dimension_{mean}$ 特征中，良性和恶性的中位数看起来并不分离，因此它没有提供很好的分类信息。

数据集二：数据集二我使用了一份关于中年妇女腰椎骨质疏松诊断的数据集，共有 356 名受试者，对应 356 份受试者的腰椎 CT 扫描图，对每一份受试者的 CT 扫描图进行提取纹理特征，得到

224 维特征，同时每名受试者都被打上标签——患病，健康。该数据集暂时无法公开。

本文涉及到的代码公开在 Github 网站上，链接：

https://github.com/Ravernclaw/Code_of_DimensionalReduction

4.2 三种搜索算法的对比实验

4.2.1 数据集一

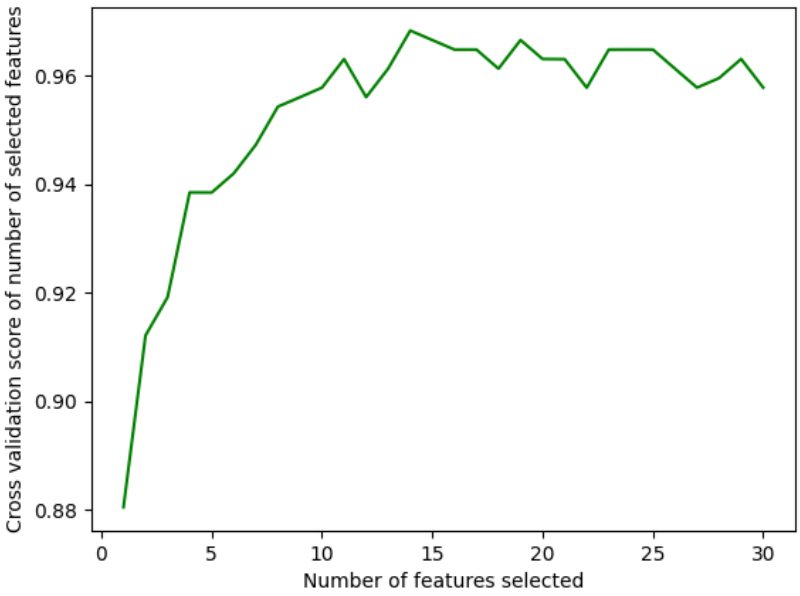

图 2 交叉验证精确度和选择特征数量关系

为了通过找到能使分类模型达到最佳分类性能的特征数量，我们用 RFE 算法搜索不同的特征子集，同时通过设置参数使搜索的特征子集大小 X 不断变化，$X \in [0, 30]$，并计算出相应特征子集下对数据集进行五折交叉验证得到的结果（分类准确度）。做出折线图如图 2 所示，在特征数量增加到 14 时，交叉验证得到的分类准确度已经达到最高，后续增加特征导致准确度反而有一定程度的下降。如表一所示:

Table 1

Number of feature dimensions	ACC	Sensitivity	Specificity	NPV	PPV
15-dimensional	**97.08%**	93.65%	**99.07%**	96.40%	**98.33%**
30-dimensional	96.49%	93.65%	98.15%	96.36%	96.72%

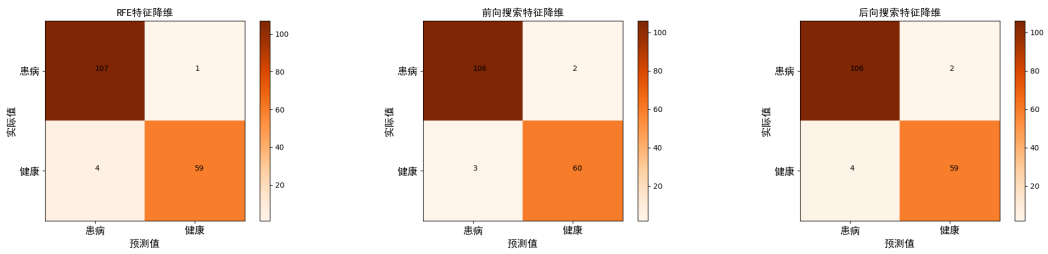

图 3 三种算法的混淆矩阵

因此我们设定将特征降到 14 维，使用三种不同的算法搜索特征子集，并使用随机森林算法进行分类，比较不同的特征的性能表现。对于每一种模型，通过计算其分类结果的准确度（ACC），敏感度（Sensitivity），特异度（Specificity），阳性预测值（PPV），阴性预测值（NPV）以及运行时间（Time）来评估其性能如表 2 所示，与此同时，三种算法得到结果的混淆矩阵也在下面给出

Table 2

Method	ACC	Sensitivity	Specificity	NPV	PPV	Time
递归特征消除 (RFE)	**97.08%**	93.65%	**99.07%**	96.40%	**98.33%**	10.749s
序列前向搜索 (SFS)	97.08%	95.24%	98.15%	97.25%	96.77%	81.702s
序列后向选择 (SBS)	96.49%	93.65%	98.15%	96.36%	96.72%	113.287s

递归特征消除算法和序列前向搜索算法都在准确度上达到了最高值 97.08%，同时，递归特征消除算法在特异度和阳性预测值两个指标上都达到了最高值。值得注意的是，相较于 SFS 算法和 SBS 算法，RFE 算法在达到更高的分类精度的同时，保证了更快的运行速度（10.749s）。接下来，我们将更换数据集，在特征维度更大的数据集上测试三种算法。

4.2.2 数据集二

如图 3 所示，在特征数量达到 50 左右时，五折交叉验证下 RFE 算法的分类准确度基本达到阈值。如表 3，在 50 维特征下，RFE 算法分类准确反而高出 224 维特征 3.74%。因此我们设置特征子集大小为 50，测试三种算法的分类性能。

Table3

Number of feature dimensions	ACC	Sensitivity	Specificity	NPV	PPV
50-dimensional	**85.98%**	87.50%	**84.75%**	89.29%	**82.35%**
224-dimensional	82.24%	91.67%	74.58%	91.67%	74.58%

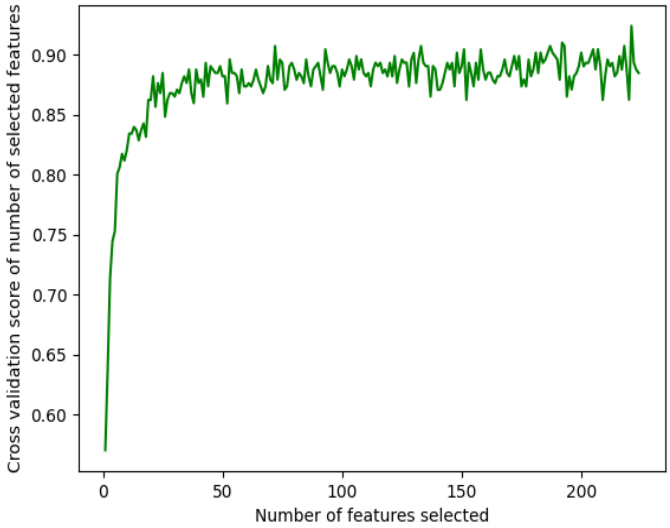

图 4 交叉验证精确度和选择特征数量关系

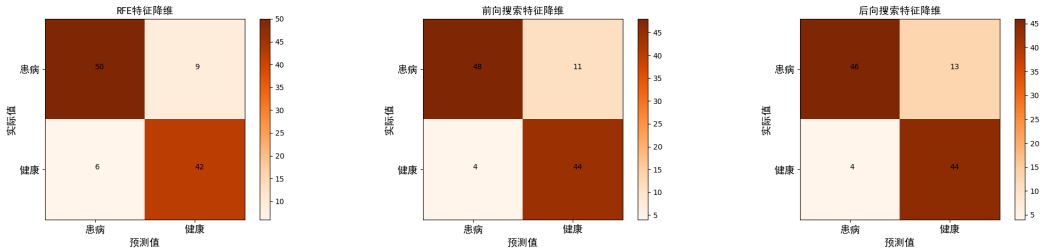

图 5 三种算法的混淆矩阵

三种算法的混淆矩阵如图五所示，在维数更多的数据集下，三种算法的分类指标比较如表四所示，RFE 算法同样在准确度，敏感度和特异度下取得了最高值，分别是 85.98%，84.75%，82.35%。但是同样条件下，SFS 算法和 SBS 算法运行所耗费的时间远远高于 RFE 算法，SBS 算法在搜索 50 维特征子集耗费了 14980.259s，在实际临床中可用性较低。

Table4

Method	ACC	Sensitivity	Specificity	NPV	PPV	time
递归特征消除 (RFE)	**85.98%**	87.50%	**84.75%**	89.29%	**82.35%**	96.232s
序列前向搜索 (SFS)	85.98%	91.67%	81.36%	92.31%	80.00%	755.124s
序列后向搜索 (SBS)	84.11%	91.67%	77.97%	92.00%	77.19%	14980.250s

4.3 两种优化算法对比实验

4.3.1 数据集一

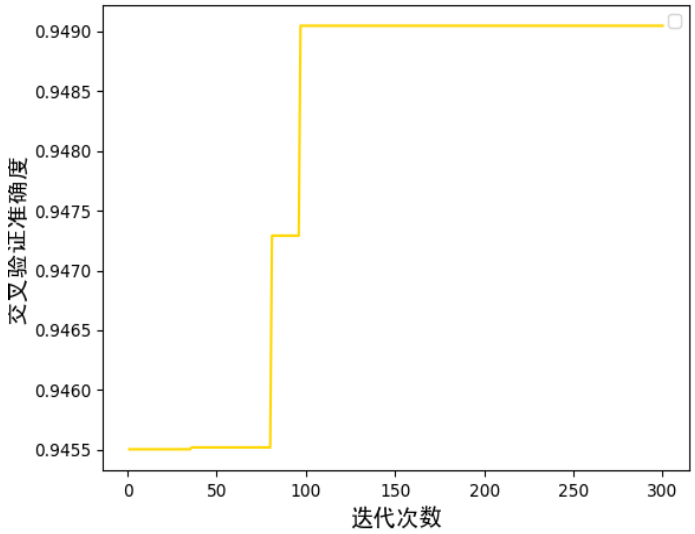

图 6 交叉验证精确度和爬山算法迭代次数的关系

针对随机爬山算法，我为了体现设置不同的迭代次数 n，算法输出得到的特征子集效果之间的差异。我通过在 0 到 300 之间调整 n，并将对应的特征输入到分类算法中，得到相应的五折交叉验证，做出图 6。如图所示，通过随机初始化得到的特征子集交叉验证精度已经接近 94.55%，但是在随后 300 次迭代之后精度并没有明显提升，由于该数据集特征数量较少，因此可能已经接近阈值。

针对 BPSO 算法，我将最大迭代次数 T_max 设置为 15，保持其他参数固定，将 30 维特征全集分别输入到两个模型——BPSO 和随机爬山算法，通过对比五类指标——ACC，Sensitivity，Specificity，NPV 和 PPV 来对比两类模型的性能。在训练集上的测试结果如图 7 所示，在 ACC，Sensitivity，NPV 和 PPV 四项指标上，BPSO 模型均优于随机爬山算法。

接下来，我们换用更大规模的数据集继续实验。

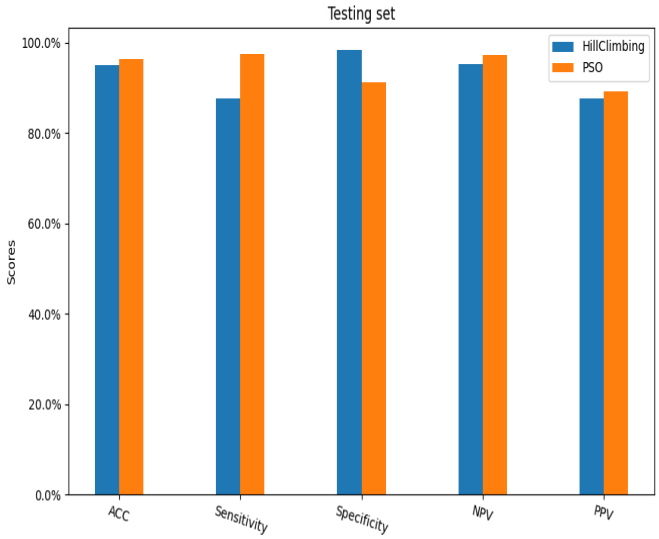

图 7 两种算法多项指标对比图

4.3.2 数据集二

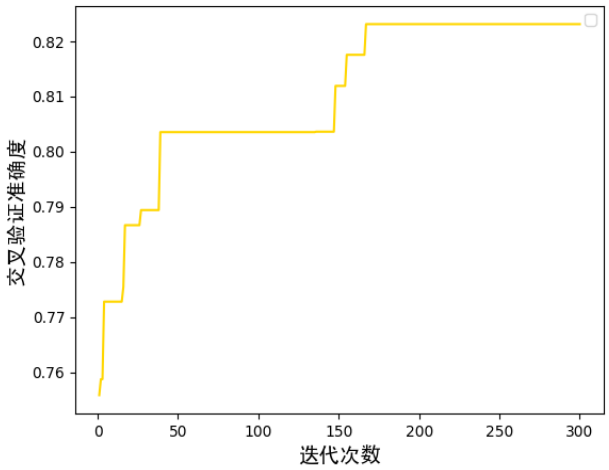

图 8 交叉验证精确度和爬山算法迭代次数的关系

如图 8 所示，在本次实验中，由于特征维数的增多，经过 300 次迭代后交叉验证精度有了明显上升——从 76% 上升到 82%，证明随机爬山算法通过迭代优化后不断接近更优的特征子集。

接着，我们过对比五类指标——ACC，Sensitivity，Specificity，NPV 和 PPV 来对比两类模型的性能。如图 9 所示，BPSO 在更高维度的数据集上搜寻更优子集的能力明显优于随机爬山算法。

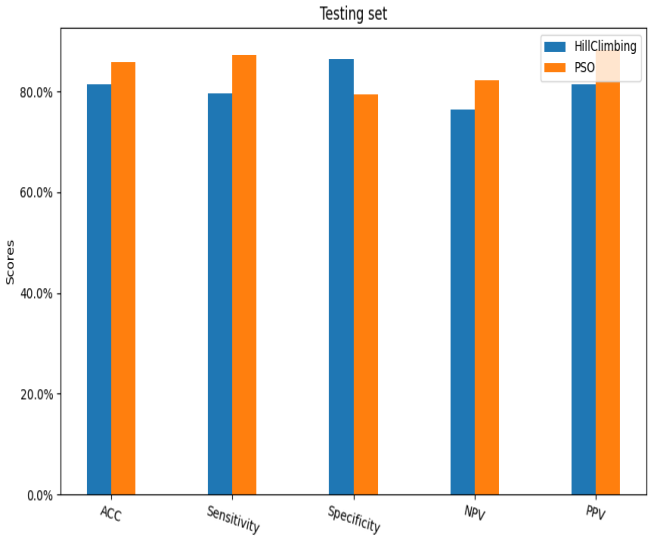

图 9 两种算法多项指标对比

5 结论和对降维任务的个人理解

在本文中，我分别从两个角度——优化策略和搜索策略看待降维问题，对于搜索策略，我使用了结合随机森林的递归特征消除（RF-RFE），序列前向搜索（SFS）和序列后向搜索（SBS），并进行了对比实验。而对于优化策略，我的对比实验中包含基于随机爬山算法的优化算法和粒子群优化算法。与此同时，我在两种不同的数据集上进行了实验，并对降维得到的特征使用分类器进行性能测试。得到的结果是，搜索策略中，RF-RFE 算法降维得到的特征具有不错的表达性，同时具有运算时间的优势；而优化策略中，粒子群优化算法在多项指标上都优于基于随机爬山算法的优化算法。

对于一个机器学习问题，数据和特征决定了机器学习的上限，而模型和算法只是逼近这个上限。由此可见，数据和特征在模型的整个开发过程中是比较重要。但是，在实际的模型应用中并不是特征越多越好，特征越多固然会给我们带来很多额外的信息，但是与此同时，一方面，这些额外的信息也增加实验的时间复杂度和最终模型的复杂度，造成的后果就是特征的“维度灾难”，使得计算耗时大幅度增加；另一方面，可能会导致模型的复杂程度上升而使得模型变得不通用。所以我们就要在众多的特征中选择尽可能相关的特征和有效的特征，使得计算的时间复杂度大幅度减少来简化模型，并且保证最终模型的有效性不被减弱或者减弱很少。通过本次研究，我学习了多种算法，并从两个角度看待理解了特征降维任务，在未来的工作中，探索更多更高效的算法对于我深刻理解这一任务是有很大帮助的。

参考文献

1 Kratsios, Anastasis; Hyndman, Cody (June 8, 2021). ”NEU: A Meta-Algorithm for Universal UAP-Invariant Feature Representation”. JMLR. 22: 10312.

2 Guyon, Isabelle; Elisseeff, André (2003). ”An Introduction to Variable and Feature Selection”. JMLR. 3.

3 J. Kennedy and R. C. Eberhart, ”A discrete binary version of the particle swarm algorithm,” 1997 IEEE International

Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, 1997, pp. 4104-4108 vol.5, doi: 10.1109/ICSMC.1997.637339.

4 Marini F, Walczak B. Particle swarm optimization (PSO). A tutorial[J]. Chemometrics and Intelligent Laboratory Systems, 2015, 149: 153-165.