机器学习 决策树实验

计 24 2012011335 柯均洁

实验设计

- **算法总体流程：**
  1. 调用 readC45(...)来读入所有数据
  2. 调用 generateTainTestSet(trainRate, validationRate)来生成训练集、验证集和测试集
  3. 调用 train 函数，用 buildDecisionTree 函数来根据训练集建立决策树
     a) 若子数据集全为同一类或者已经对所有属性都进行了划分，则将节点认定为叶节点并返回
     b) 将当前节点的类标签定为当前子数据集中最多数的类（调用函数 MajorityVoting 获取标签）
     c) 计算各属性的信息增益，并从中选择信息增益率最大的属性作为分类属性
     d) 根据分类属性的取值划分当前数据集，对划分后的子数据集递归建树，并将生成的子节点加入到当前节点的 children 中
  4. 调用 pruning()函数来进行后剪枝
     a) 首先通过 classifyValidationSet()来将验证集的所有数据进行分类，分类的过程中，数据所经过的每个节点都会更新其分类的计数器，保存在 TreeNode 的 data 中。使得每个节点都知道自己的子树中有多少正例和反例。
     b) 通过 PEPrune(Node)遍历决策树，根据验证集的信息进行悲观剪枝 PEP
  5. 调用 test()进行测试，返回正确率
- **实验过程：**
  1. Task1 和 Task2：修改 task1(trainRate)、task2(trainRate, validationRate)的参数，运行 5 次并记录树的大小和正确率，计算 min/max/average

2. Task3：直接调用 task3()函数，即可得到分类结果

- **属性选择方法：**

  选择信息增益率最大的属性进行划分

$$GainRatio(S,A)=\frac{Gain(S,A)}{SplitInfo(S,A)}$$

$$SplitInfo(S,A)=-\sum_{i=1}^{c}\frac{|S_i|}{|S|}log_2\frac{|S_i|}{|S|}$$

- **连续属性值的信息增益率计算：**

  1. 统计当前节点数据子集中所有的属性值
  2. 排序之后两两取中点作为可能的候选分隔阈值点 split point
  3. 对每个阈值点计算其信息增益，选择信息增益最大的阈值点作为当前节点该属性的分隔点，记录到 newSplitPoint 中
  4. 根据计算所得的属性值分割点计算该节点的信息增益率

- **剪枝策略：Reduce-Error Pruning 剪枝 REP**

  1. 先将验证集全部进行分类，分类的过程中统计每个节点所拥有的正例和负例的个数，记录在 data 中
  2. 对决策树进行遍历，如果“将当前节点视为一个叶子”比“对当前节点进行划分”所得到的正确分类的例子更多，则将子树减去，当前节点变为叶子节点

- **缺失数据的处理：**

  1. 在计算某属性信息增益时若出现了样本的属性值缺失，则将该样本的属性值赋值为当前子集出现最多的属性值
  2. 在选择某属性作为分隔属性后需要对子集进行划分时，若某样本该属性值缺失，则将该样本加入该属性值取值最多的那个集合中
  3. 若建树完成之后，待分类样本中出现分类属性值缺失，则直接将当前节点的 classLabel 作为该样本的分类结果

实验结果

Task 1

- 总体情况：

Sample Rate	Average Tree Size	Accuracy		
		Min	Max	Average
5%	1952	0.8002	0.8136	0.8065
50%	11543	0.8242	0.8288	0.8273
100%	19653	0.8327	0.8327	0.8327

- 详细实验数据：

Sample rate		1	2	3	4	5
5%	Tree size	1669	1974	1834	1972	2313
	Accuracy	0.8002	0.8017	0.8128	0.8136	0.8041
50%	Tree size	12120	11016	11530	11697	11351
	Accuracy	0.8272	0.8288	0.8242	0.8284	0.8273
100%	Tree size	19653	19653	19653	19653	19653
	Accuracy	0.8327	0.8327	0.8327	0.8327	0.8327

Task 2

- 总体情况：

Sample Rate	Tree Size 剪枝前	Tree Size 剪枝后	Accuracy		
			Min	Max	Average
5%	1387	1103	0.8235	0.8386	0.8331
50%	8142	6671	0.8386	0.8471	0.8445
100%	14240	11696	0.8478	0.8535	0.8493

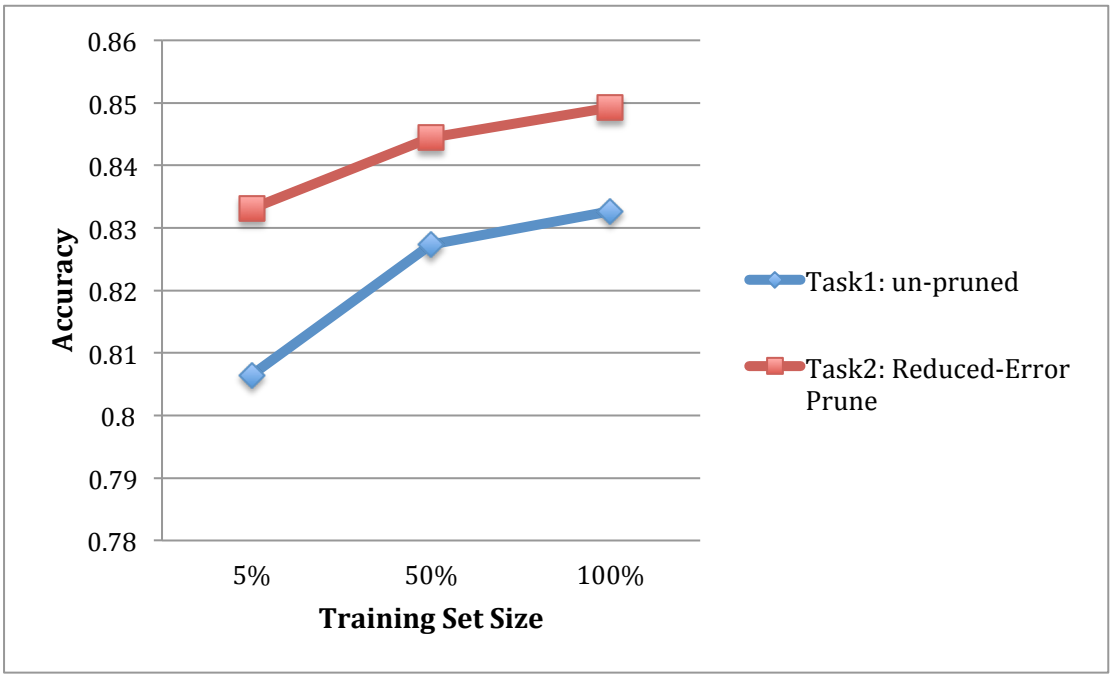
- 详细实验数据：

Sample Rate		1	2	3	4	5
5%	pre-treesize	1346	1481	1482	1381	1244
	post-treesize	980	1266	1091	1185	991
	accuracy	0.8385	0.8337	0.8386	0.8314	0.8235
50%	pre-treesize	7823	8275	7916	8429	8269
	post-treesize	6253	6910	6558	6867	6765
	accuracy	0.8418	0.8471	0.8433	0.8458	0.8443
100%	pre-treesize	14882	14312	14406	13580	14020
	post-treesize	12384	11547	11748	11089	11714
	accuracy	0.8535	0.8478	0.8487	0.8483	0.8484

实验结果分析

- **过拟合问题与剪枝：**

  绘制 task1 和 task2 的平均正确率的对比图：



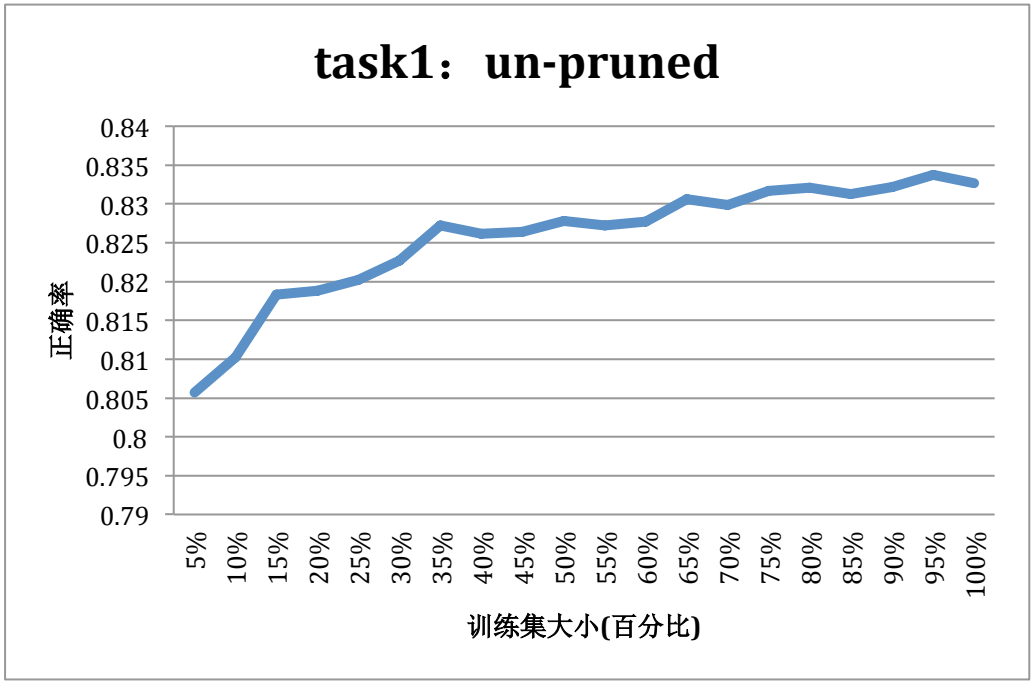
可以看出，在进行剪枝之后，整体的正确率都提升了，表明在剪枝前确

实出现了过拟合 over-fitting 的情况。也就是说，在未剪枝的情况下，算法生成的决策树过于详细和庞大，因为每个属性都被详细地考虑过，而且决策树的树叶节点所覆盖的训练样本都是“纯”的。这样，训练样本中的错误数据也会被决策树学习，成为决策树的部分，但是对于测试数据的表现就没有想象的那么好。

通过 REP 剪枝，能够自底向上地修剪决策树来最大限度地提高验证集合的精度，从而精简当前的决策树，一定程度地缓解过拟合的问题。

- **训练集大小对数据正确率的影响：**

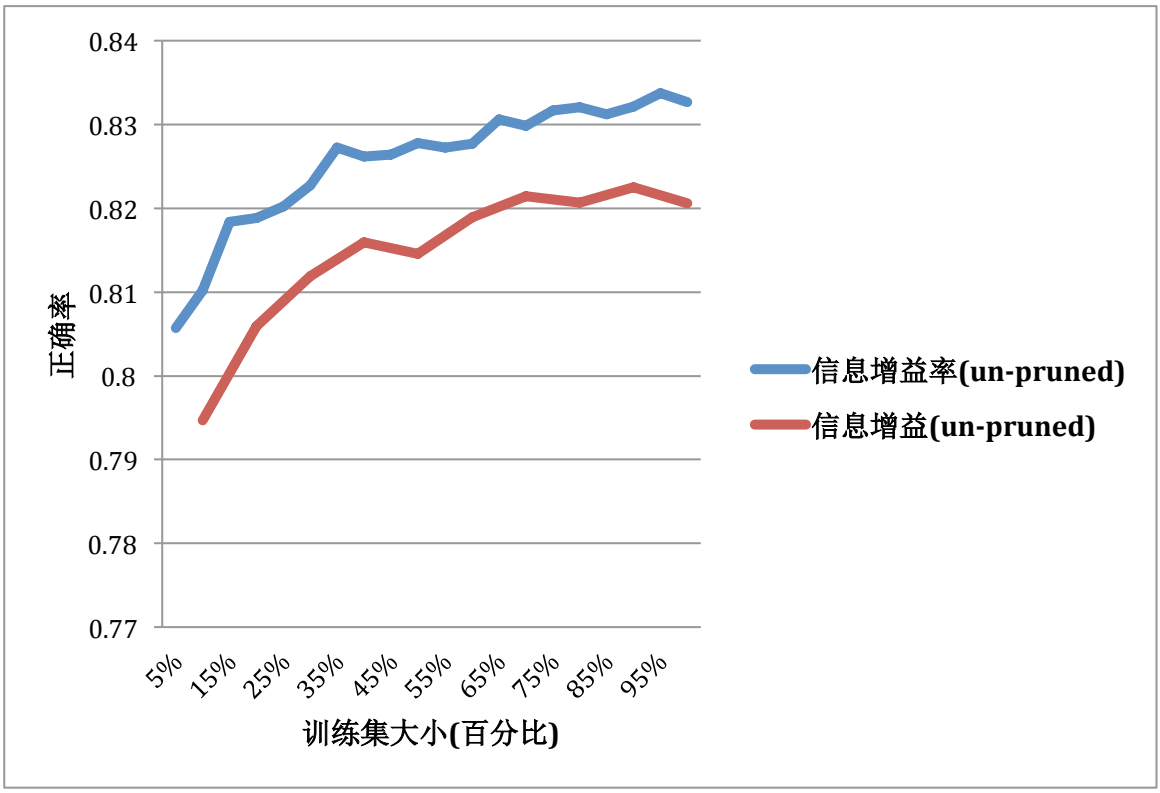
除了老师规定的实验，我还额外用测试了不同百分比下 task1 的正确率情况。以 5%、10%、15%…100%的训练集进行训练，记录 5 次训练的平均正确率，绘得图像为：



可以看出，随着训练集的不断增加，正确率是一直上升的，但上升的速度不断减慢，最终稳定在 83.5%附近。表明决策树学习的方法需要有一定的训练集规模，但随着数据集规模的增大，决策树学习的正确率提升也是有限的，而且随着数据集增大，决策树也变得越来越复杂，时间空间开销也不断地上升（一开始我用 c++写的决策树就出现了递归内存开销过大的问题，后来改用 java，但是最后跑 100%的训练集时间都要快 10 分钟…）由此可见，决策树学习虽然易于理解和实现，也还是有其局限性的。

- **“信息增益”与“信息增益率”：**

在实现算法的过程中，我先是按老师上课讲的 ID3 算法，用“信息增益”来选择分裂属性。但在查资料的过程中了解到 C4.5 作为 ID3 的改进，是用“信息增益率”来选择待分裂属性的。出于好奇，我测试了这两种属性选择方法的正确率。同样是以 5%、10%、15%…100%的训练集进行训练，记录 5 次训练的平均正确率，绘得图像为：



从图像可以看出，简单地将划分方法从“信息增益”改为“信息增益率”，就让正确率有了显著提升！实在是喜大普奔！通过进一步查阅资料我了解到，之所以 C4.5 要做这样的改进，是因为用“信息增益”作为划分标准存在问题：对于那些各类别样本数量不一致的数据，在决策树当中信息增益的结果偏向于那些具有更多数值的特征。而“信息增益率”则考虑了分支数量和尺寸的因素，在信息增益的基础上除了一项 split information，来惩罚值更多的属性，从而提升了划分的正确率。

实验总结

通过这次亲自动手编写决策树，我对 ID3 以及 C4.5 等决策树算法有了更加深入的理解，通过对比剪枝前后正确率的变化，我更加直观地认识到了决策树学习中存在的过拟合问题。除此之外，通过对比“信息增益”和“信息增益率”这两种属性选择标准，我对属性选择的方法也有了进一步的认识和体会。总之，通过这次实验，我提高了对决策树算法的认识，收获很大！