数据挖掘Lab3实验报告

161220096 欧阳鸿荣

1.实验要求

在提供的数据集上，基于10折的交叉验证，使用J48(C4.8)，朴素贝叶斯，SVM，神经网络，kNN算法以及它们使用装袋(Bagging)的集成学习的版本对数据集训练。基于如accuracy，AUC等指标比较各种方法的表现。讨论各种方法的表现并且说明如何优化基于Bagging的kNN算法的性能。

2.实验配置

综合考虑数据集格式(arff)等多方面原因，本次实验采用Weka进行数据挖掘。使用的是Weka 3.8版本。

对于实验要求的算法，J4.8 (C4.5)、Naïve Bayes、神经网络和kNN在Weka中都有提供。其中，神经网络采用的是Weka中的Multilayer Perceptron多层感知机，kNN算法采用Weka中的IBk(K-nearest neighbours classifier)分类器。而对于Weka默认安装里缺少的SVM，使用的是weka库中的libSVM进行实验。

在实验中，在对比多个方法的任务中，统一采用原始数据，10-fold cross validation和weka中的默认配置进行分类器的训练。在对kNN的优化中，才考虑对数据预处理和修改默认配置等因素。同时，在实验过程中发现libSVM的表现很不稳定，因此对其进行数据预处理实验进行对比，从此发现了对数据标准化等预处理手段的重要性。

3.实验原理

3.1 J4.8 (C4.5)

3.2 Naïve Bayes

3.3 SVM

3.4 Neural Network

3.5 kNN

4.实验过程与实验结果

4.1 数据集基本情况

数据集	Attributes	Instances	数据规模
breast-w	10	699	6990
colic	23	368	8464
credit-a	16	690	11040
credit-g	21	1000	21000
diabetes	9	768	6912
hepatitis	20	155	3100
mozilla4	6	15545	93270
pc1	22	1109	24398
pc5	39	17186	670254
waveform-5000	41	5000	205000

4.2 五种方法的对比

该任务中，统一采用原始数据，10-fold cross validation和weka中的默认配置进行分类器的训练。其中J4.8 (C4.5)、Naïve Bayes采用Weka中默认版本，神经网络采用的是Weka中的Multilayer Perceptron多层感知机，kNN算法采用Weka中的IBk分类器。SVM使用的是weka库中的libSVM进行实验。

训练的模型和结果都放在`output`目录下，由于Markdown表格能力有限，结果备份在`实验三结果统计.xlsx`下。

1 breast-w 数据集

数据规模：6990	基础方法	基础方法	Bagging	Bagging	变化量	变化量
挖掘算法	Accuracy	ROC	Accuracy	ROC	Accuracy	ROC
J4.8 (C4.5)	94.56%	0.955	**96.28%**	0.985	1.72%	0.03
Naïve Bayes	**95.99%**	**0.986**	95.85%	**0.989**	-0.14%	0.003
SVM	95.71%	0.964	95.42%	0.973	-0.29%	0.009
Neural Network	95.28%	0.986	95.99%	**0.989**	0.72%	0.003
kNN	95.14%	0.973	95.85%	0.987	0.72%	0.014
最大值	95.99%	0.986	96.28%	0.989		
最优算法	Naïve Bayes	Naïve Bayes	J48	NB/NN		

2 colic 数据集

数据规模: 8464	基础方法	基础方法	Bagging	Bagging	变化量	变化量
挖掘算法	Accuracy	ROC	Accuracy	ROC	Accuracy	ROC
J4.8 (C4.5)	**85.33%**	0.813	**85.60%**	0.864	0.27%	0.051
Naïve Bayes	77.99%	0.842	77.99%	0.842	0.00%	0
SVM	72.55%	0.67	69.57%	0.692	-2.99%	0.022
Neural Network	80.43%	**0.857**	84.51%	**0.876**	4.08%	0.019
kNN	81.25%	0.802	81.25%	0.824	0.00%	0.022
最大值	85.33%	0.857	85.60%	0.876		
最优算法	J48	Neural Network	J48	Neural Network		

3 credit-a 数据集

数据规模：11040	基础方法	基础方法	Bagging	Bagging	变化量	变化量
挖掘算法	Accuracy	ROC	Accuracy	ROC	Accuracy	ROC
J4.8 (C4.5)	**86.09%**	0.887	**86.81%**	**0.928**	0.72%	0.041
Naïve Bayes	77.68%	**0.896**	77.83%	0.896	0.14%	0
SVM	55.51%	0.513	55.80%	0.535	0.29%	0.022
Neural Network	83.62%	0.895	85.07%	0.908	1.45%	0.013
kNN	81.16%	0.808	81.30%	0.886	0.14%	0.078
最大值	86.09%	0.896	86.81%	0.928		
最优算法	J48	Naïve Bayes	J48	J48		

4.credit-g 数据集

数据规模: 21000	**基础方法**	**基础方法**	**Bagging**	**Bagging**	**变化量**	**变化量**
挖掘算法	Accuracy	ROC	Accuracy	ROC	Accuracy	ROC
J4.8 (C4.5)	70.50%	0.639	73.30%	0.753	2.80%	0.114
Naïve Bayes	**75.40%**	**0.787**	74.80%	**0.787**	-0.60%	0
SVM	68.70%	0.491	68.60%	0.49	-0.10%	-0.001
Neural Network	71.50%	0.73	**76.10%**	0.776	4.60%	0.046
kNN	72%	0.66	72.10%	0.694	0.10%	0.034
最大值	75.40%	0.787	76.10%	0.787		
最优算法	Naïve Bayes	Naïve Bayes	Neural Network	Naïve Bayes		

5.diabetes 数据集

数据规模: 6912	**基础方法**	**基础方法**	**Bagging**	**Bagging**	**变化量**	**变化量**
挖掘算法	Accuracy	ROC	Accuracy	ROC	Accuracy	ROC
J4.8 (C4.5)	73.83%	0.751	74.61%	0.798	0.78%	0.047
Naïve Bayes	**76.30%**	**0.819**	76.56%	0.817	0.26%	-0.002
SVM	65.10%	0.5	65.10%	0.5	0.00%	0
Neural Network	75.39%	0.793	**76.82%**	**0.822**	1.43%	0.029
kNN	70.18%	0.65	71.09%	0.725	0.91%	0.075
最大值	76.30%	0.819	76.82%	0.822		
最优算法	Naïve Bayes	Naïve Bayes	Neural Network	Neural Network		

6.hepatitis 数据集

数据规模：3100	基础方法	基础方法	Bagging	Bagging	变化量	变化量
挖掘算法	Accuracy	ROC	Accuracy	ROC	Accuracy	ROC
J4.8 (C4.5)	83.87%	0.708	83.87%	0.865	0.00%	0.157
Naïve Bayes	**84.52%**	**0.86**	**85.81%**	**0.89**	1.29%	0.03
SVM	79.35%	0.5	79.35%	0.492	0.00%	-0.008
Neural Network	80%	0.823	84.52%	0.846	4.52%	0.023
kNN	80.65%	0.653	81.29%	0.782	0.65%	0.129
最大值	84.52%	0.86	85.81%	0.89		
最优算法	Naïve Bayes	Naïve Bayes	Naïve Bayes	Naïve Bayes		

7.mozilla4 数据集

数据规模：93270	基础方法	基础方法	Bagging	Bagging	变化量	变化量
挖掘算法	Accuracy	ROC	Accuracy	ROC	Accuracy	ROC
J4.8 (C4.5)	**94.80%**	**0.954**	**95.11%**	**0.976**	0.32%	0.022
Naïve Bayes	68.64%	0.829	68.74%	0.83	0.10%	0.001
SVM	69.54%	0.537	69.82%	0.549	0.28%	0.012
Neural Network	91.19%	0.94	91.28%	0.945	0.10%	0.005
kNN	88.99%	0.877	88.86%	0.928	-0.13%	0.051
最大值	94.80%	0.954	95.11%	0.976		
最优算法	J48	J48	J48	J48		

8.pc1 数据集

数据规模：24398	基础方法	基础方法	Bagging	Bagging	变化量	变化量
挖掘算法	Accuracy	ROC	Accuracy	ROC	Accuracy	ROC
J4.8 (C4.5)	93.33%	0.668	93.60%	**0.855**	0.27%	0.187
Naïve Bayes	89.18%	0.65	88.91%	0.628	-0.27%	-0.022
SVM	93.51%	0.563	**93.87%**	0.574	0.36%	0.011
Neural Network	**93.60%**	0.723	93.33%	0.835	-0.27%	0.112
kNN	92.06%	**0.74**	91.07%	0.793	-0.99%	0.053
最大值	93.60%	0.74	93.87%	0.855		
最优算法	Neural Network	kNN	SVM	J48		

9.pc5 数据集

数据规模：670254	基础方法	基础方法	Bagging	Bagging	变化量	变化量
挖掘算法	Accuracy	ROC	Accuracy	ROC	Accuracy	ROC
J4.8 (C4.5)	**97.46%**	0.817	**97.53%**	**0.959**	0.06%	0.142
Naïve Bayes	96.42%	0.833	96.48%	0.845	0.06%	0.012
SVM	97.25%	0.548	97.22%	0.552	-0.03%	0.004
Neural Network	97.10%	**0.941**	97.31%	0.954	0.21%	0.013
kNN	97.29%	0.932	97.37%	0.953	0.08%	0.021
最大值	97.46%	0.941	97.53%	0.959		
最优算法	J48	Neural Network	J48	J48		

10.waveform-5000 数据集

数据规模：205000	**基础方法**	**基础方法**	**Bagging**	**Bagging**	**变化量**	**变化量**
挖掘算法	Accuracy	ROC	Accuracy	ROC	Accuracy	ROC
J4.8 (C4.5)	75.08%	0.83	81.20%	0.949	6.12%	0.119
Naïve Bayes	80%	0.956	79.98%	0.956	-0.02%	0
SVM	**86.42%**	0.898	**86.02%**	0.939	-0.40%	0.041
Neural Network	83.56%	**0.963**	85.68%	**0.969**	2.12%	0.006
kNN	73.62%	0.802	74.46%	0.9	0.84%	0.098
最大值	86.42%	0.963	86.02%	0.969		
最优算法	SVM	Neural Network	SVM	Neural Network		

11.结果综合分析

比较各个数据集的情况，不难发现运用集成学习后，大多数方法的准确率和ROC值都有着提高，说明**Bagging的集成学习方法确实可以提高分类器的准确率。**

根据上述实验结果，整理出各个数据集上Accuracy和ROC值的最优方法如下

数据集	**常规方法**		**Bagging**	
名称	准确率最优	ROC最优	准确率最优	ROC最优
breast-w	Naïve Bayes	Naïve Bayes	J48	NB/NN
colic	J48	Neural Network	J48	Neural Network
credit-a	J48	Naïve Bayes	J48	J48
credit-g	Naïve Bayes	Naïve Bayes	Neural Network	Naïve Bayes
diabetes	Naïve Bayes	Naïve Bayes	Neural Network	Neural Network
hepatitis	Naïve Bayes	Naïve Bayes	Naïve Bayes	Naïve Bayes
mozilla4	J48	J48	J48	J48
pc1	Neural Network	kNN	SVM	J48
pc5	J48	Neural Network	J48	J48
waveform-5000	SVM	Neural Network	SVM	Neural Network

根据上述结果，可以发现，在10个数据集中，五种方法的最优分布如下

方法	常规方法		Bagging		总结	
名称	准确率最优	ROC最优	准确率最优	ROC最优	准确率最优	ROC最优
J4.8 (C4.5)	4	1	5	4	9	5
Naïve Bayes	4	5	1	3	5	8
SVM	1	0	2	0	3	0
Neural Network	1	3	2	4	3	7
kNN	0	1	0	0	0	1

不难看出，在给定的数据集上使用默认参数时，各方法表现如下，

- J48的效果最佳的频数是最多的，但是其ROC的表现就稍有欠缺。
- 朴素贝叶斯方法在准确率和ROC上都有不错的表现，性能较为平衡。
- SVM的表现较不稳定，其准确率的上限可以达到很高，但下限却无法保证，与其他方法对比常常处于很糟糕的水平，这里猜测是使用的libSVM库和数据预处理的问题，因此后文专门针对SVM进行数据预处理特殊试验。
- 神经网络的表现也是中规中矩，但是其ROC的性能处于很高的水平。
- kNN没有什么亮点，可能是应用默认参数的原因，也许这就是后续实验需要对kNN进行参数调优的原因。

而若考虑数据集规模和不同分类器的表现，有如下表格（按照属性数升序）：

数据集	Attributes	Instances	sum	常规方法		Bagging	
名称	属性数	样本数	规模	准确率最优	ROC最优	准确率最优	ROC最优
mozilla4	6	15545	93270	J48	J48	J48	J48
diabetes	9	768	6912	NB	NB	NN	NN
breast-w	10	699	6990	NB	NB	J48	NB/NN
credit-a	16	690	11040	J48	NB	J48	J48
hepatitis	20	155	3100	NB	NB	NB	NB
credit-g	21	1000	21000	NB	NB	NN	NN
pc1	22	1109	24398	NN	kNN	SVM	J48
colic	23	368	8464	J48	NN	J48	NN
pc5	39	17186	670254	J48	NN	J48	J48
waveform-5000	41	5000	205000	SVM	NN	SVM	NN

根据上述表格，不难得到以下结论

- J48的表现较为稳定，在不同数据维度和样本容量上都能有较好的效果。

- 朴素贝叶斯在属性数较少时有着较好的效果（猜测是此时对于独立性的满足程度较高
- 神经网络在属性数和样本数较高时都有良好的表现
- SVM虽然不稳定，但在数据维度和数据量较大时能起到较好的效果。

4.3 归一化SVM的对比

上文也提到，libSVM的效果并非很稳定，出于好奇心，我对于数据集进行了归一化的数据预处理，发现这对于libSVM性能有着巨大的影响，而对于其他方法的影响并不大。下面给出针对数据归一化的SVM的实验数据：

	基础方法		**基础方法归一化**		**变化量**	
数据集	Accuracy	ROC	Accuracy	ROC	Accuracy	ROC
breast-w	95.71%	0.964	96.71%	0.965	1.00%	0.001
colic	72.55%	0.67	84.24%	0.813	11.68%	0.143
credit-a	55.51%	0.513	85.51%	0.862	30.00%	0.349
credit-g	68.70%	0.491	72%	0.546	3.30%	0.055
diabetes	65.10%	0.5	76.95%	0.704	11.85%	0.204
hepatitis	79.35%	0.5	80.65%	0.589	1.29%	0.089
mozilla4	69.54%	0.537	84.99%	0.848	15.45%	0.311
pc1	93.51%	0.563	93.51%	0.563	0.00%	0
pc5	97.25%	0.548	97.04%	0.513	-0.21%	-0.035
waveform-5000	86.42%	0.898	86.02%	0.895	-0.40%	-0.003

	Bagging		Bagging归一化		变化量	
数据集	Accuracy	ROC	Accuracy	ROC	Accuracy	ROC
breast-w	95.42%	0.973	96.71%	0.972	1.29%	-0.001
colic	69.57%	0.692	84.24%	0.856	14.67%	0.164
credit-a	55.80%	0.535	85.51%	0.863	29.71%	0.328
credit-g	68.60%	0.49	73%	0.692	4.40%	0.202
diabetes	65.10%	0.5	76.04%	0.732	10.94%	0.232
hepatitis	79.35%	0.492	81.94%	0.77	2.58%	0.278
mozilla4	69.82%	0.549	84.91%	0.861	15.09%	0.312
pc1	93.87%	0.574	93.87%	0.574	0.00%	0
pc5	97.22%	0.552	97.05%	0.528	-0.17%	-0.024
waveform-5000	86.02%	0.939	86.02%	0.939	0.00%	0

不难看出，归一化后SVM算法的性能得到了显著的提高。这让我深刻意识到了数据预处理的重要性。

4.4 kNN算法的调优

考虑到数据集的规模和准确率，这里对credit-a 数据集进行调优，根据Weka中的默认值，对照基本结果如下：

	准确率	ROC
未Bagging	81.1594 %	0.808
Bagging	81.3043 %	0.886

根据本次实验中的经验，对数据归一化处理后训练速度会加快，而Bagging集成学习的效果大多数情况下会提高，因此优化过程中，我们的起点方案便是选取**归一化的Bagging方案为起点，并在其基础上进行优化。**

1.调整最近邻搜索算法

Weka中的kNN默认使用的是LinearNNseach算法，因此我们考虑用不同的最近邻搜索方法调优。而在此之前，对缺失值进行处理(使用ReplaceMissingValues过滤器)，用均值代替缺失值。

最近邻搜索算法	准确率	ROC
BallTree	81.3043 %	0.889
CoverTree	81.3043 %	0.887
FilteredNeighbourSearch	81.3043 %	0.889
KDTree	81.3043 %	0.889
LinearNNseach	81.3043 %	0.889

可以发现在这个数据集下，各个最近邻搜索函数在默认参数下性能差距不大。因此依然使用LinearNNseach算法

2.调整距离函数

Weka中的kNN默认使用的是欧氏距离，因此我们考虑使用不同距离函数来调优

距离函数	准确率	ROC
Euclidean distance	81.3043 %	0.889
Chebyshev distance.	76.3768 %	0.828
FilteredDistance	79.1304 %	0.865
Manhattan distance	80.8696 %	0.888
Minkowski distance	81.3043 %	0.889

不难看出，Euclidean距离和Minkowski距离在该数据集上表现最好，因此还是采用默认的Euclidean距离进行下一步实验。

3.调整距离权重

这是kNN算法中一种常见的调优：为每个点的距离增加一个权重，使得距离近的点可以得到更大的权重，而Weka中默认的是不加权的，因此我尝试调整距离权重以期望达到较好的效果：

距离权重	准确率	ROC
无	81.3043 %	0.889
1/distance	81.1594 %	0.890
1-distance	81.0145 %	0.891

可以发现调整距离权重后，ROC值有些些许提高，但是准确率却下降了，因此选择不设距离权重。

4.调整k值

Weka中的kNN默认近邻数k值是1，基于上文中的结果，采用LinearNNseach算法，欧式距离，不设距离权重，调整K值，观察实验结果。

k值	准确率	ROC
1	81.3043 %	0.889
2	84.2029 %	0.899
3	85.6522 %	0.907
4	86.5217 %	0.906
5	86.5217 %	0.911
6	86.5217 %	0.911
7	86.2319 %	0.910
8	86.087 %	0.913
9	86.5217 %	0.914
10	86.087 %	0.913
20	85.942 %	0.911

可以看出，当选取的近邻数k值在5附近时，准确率达到较高水平86.5217，已经接近实验中该数据集的最高准确率(J48的86.81%)，因此我们认为k=5时kNN算法的性能达到最优

5.实验结果和感悟

- 数据预处理（如归一化和缺失值处理）对于数据挖掘结果的准确性起着很大的作用
- 大多数情况下，集成学习能够显著提高分类算法的准确率和ROC值
- 各个分类算法都有其适用的场景和优点
    - **J48(C4.5)**的表现较为稳定，在不同数据维度和样本容量上都能有较好的效果。且其具有较好的解释性，在Weka中也可以导出其决策树。而且对于各类型数据都能很好地兼容处理，并且能够在相对短的时间内能够对大型数据源做出可行且效果良好的结果。
    - **朴素贝叶斯**在属性数较少时有着较好的效果（猜测是此时对于独立性的满足程度较高），对缺失数据不太敏感，算法也比较简单。在属性相关性较小时，朴素贝叶斯模型的性能最为良好。在属性个数比较多或者属性之间相关性较大时，朴素贝叶斯的分类效率比不上决策树模型。
    - **神经网络**在属性数和样本数较高时都有良好的表现，其分类的准确度高,对噪声数据有较强的鲁棒性和容错能力，能充分逼近复杂的非线性关系。但是神经网络训练时间相比其他方法要长很多，同时解释性不强。
    - **SVM**虽然不稳定，但在数据维度和数据量较大时能起到较好的效果。SVM可以提高泛化性能，解决高维问题和非线性问题，但是对非线性问题没有通用解决方案，必须谨慎选择核函数来处理。
    - **kNN**算法简单、有效，适用于样本容量比较大的类域的自动分类，且对于样本重叠较多的数据集能起到较好的效果。但是KNN算法是lazy learning，速度较慢且计算量较大。
- 对于kNN算法，最近邻搜索算法，距离函数，距离权重的选取都会对性能和结果产生很大影响。而提高最近邻数k值是提高kNN算法最简单直接有效的办法之一。

6.参考链接

1.数据预处理和weka.filters的使用--数据挖掘学习和weka使用（三）

2.初试weka数据挖掘

3.关于机器学习的Weka软件详细教程（转载）

4.Weka SVM lib 下载

5.Weka BP神经网络（Neural Networks）分析

6.[机器学习]K近邻算法及其应用--WEKA工具

7.各类分类算法比较