# 集成学习实验报告

千英卓 2017013622

## 1 Experiment Design

在这次实验中，我使用两种集成学习算法(**Bagging** & **AdaBoost.M1**)，和两种基本的统计分类算法(**Support Vector Machine** & **Decision Tree**)进行review的评分预测。

除此之外，我还实现了一个简单的**Multilayer Perceptron**，并将其使用**Bagging**的表现与不使用时对比，发现**Bagging**的提高有限。

### 1.1 Feature Extraction

实验数据中有意义的域有summary和review，我采取bag-of-words对其进行特征提取。

首先建立词库：将词定义为符合Python正则表达式[a-z]+(所有文本均小写处理)，对trainset中的所有词按照出现频数排序。由于高频词往往包含信息较少，去除频数最高的$remove_head$个词，采用之后的$vocab_size$个词作为词库。

将每个样本中summary和review中在词库里的词使用bag-of-words表示：对于词库中第$i$个词，设其在summary和review中出现次数分别为$\#(s), \#(r)$，则特征向量的第$i$维的值为$\#(s) \times summary_weight + \#(r)$，即每个样本的特征维度均为$vocab_size$。

### 1.2 Classifiers

虽然评分看似是回归问题，但由于得分均为整数（离散），所以将其视为分类问题，使用分类器。

使用了**Support Vector Machine**和**Decision Tree**作为基本的分类器，采用scikit-learn库中的svm.LinearSVC和tree.DecisionTreeClassifier的实现。创建SVM时使用参数$max_iter$，DT使用参数$max_depth$，其他参数均为默认值。

另外使用Pytorch实现了一个两层的MLP，训练参数为：$hidden\_size = 256, batch\_size = 10, learning\_rate = 0.0001, activation\_function = ReLU, optimizer = SGD, loss\_function = MSELoss$

### 1.3 Ensemble Learning

使用**Bagging** 和 **AdaBoost.M1**两种集成学习算法，其实现参照课件中的算法。两个算法都只有迭代次数$T$一个参数。

由于最终结果接受非整数，所以不进行投票决定类别，而是直接将算法得到的若干个分类器的结果加权平均(**Bagging**中为平权)作为最终结果。

### 1.4 Evaluation

使用root-mean-square-error(**RMSE**)进行evaluation。设标签为$y$，而预测结果为$p$，RMSE计算公式为

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(p_i - y_i)^2}{n}}$$

## 2 Experiment Results

以下若未特殊说明，则*remove_head*=20，*vocab_size*=2000，*summary_weight*=5。

### 2.1 Bagging + SVM

对迭代次数$T$进行了实验，SVM的*max_iter*设为1000.

| T | RMSE |
|---|---|
| 1 | 1.08222 |
| 5 | 0.99194 |
| 20 | 0.93517 |
| 30 | **0.93011** |

表 1: 迭代次数的实验结果(Bagging+SVM)

并且对SVM的参数*max_iter*的选取进行了实验。由于训练时间限制，Bagging的迭代次数设为5。为了体现区别，展示了不使用集成学习算法的**Base**方法的效果。

| max_iter/algorithm | Bagging | Base |
|---|---|---|
| 500 | **0.97045** | 1.03166 |
| 1000 | 0.99194 | 1.04434 |
| 2000 | 0.98690 | 1.04864 |

表 2: max_iter的实验结果(Bagging+SVM)

### 2.2 AdaBoost + SVM

同Adaboost的迭代次数进行实验，SVM的*max_iter*设为1000。由于在改变权重的数据集上，分类器的准确率很快会降到0.5以下，所以实际迭代次数一般只能到4或5。

同样对*max_iter*的选取进行了实验，最大迭代次数T_max取为3。

| T | RMSE | actural T |
|---|---|---|
| 1 | 1.04259 | 1 |
| 3 | 0.92402 | 3 |
| 10 | **0.88955** | 5 |

表 3: 迭代次数的实验结果(AdaBoost+SVM)

| max_iter | RMSE | actural T |
|---|---|---|
| 500 | **0.90922** | 3 |
| 1000 | 0.92402 | 3 |
| 2000 | 0.93369 | 3 |

表 4: max_iter的实验结果(AdaBoost+SVM)

### 2.3 Bagging + DT

对迭代次数$T$和DT的参数$max_depth$进行了实验。表6中$max_depth$为10，表7中$T$为10。同上，**Base**表示不使用集成学习得到的结果。

| T | RMSE |
|---|---|
| 1 | 1.22787 |
| 5 | 1.20101 |
| 10 | 1.19579 |
| 20 | **1.18016** |
| 30 | 1.18721 |

表 5: 迭代次数的实验结果(Bagging+DT)

### 2.4 AdaBoost + DT

实验方法与之前类似，若未特殊说明则$T$为3，$max_depth$为10。

## 3 Performance on Leaderboard

图9展示各种方法在test集上取得的最好成绩(注：排名以*5.15 22:00*情况为准)。其中AdaBoost+SVM的组合进行了对特征提取参数的调整，为$remove_head$=20,$vocab\_size$=5000, $summary\_weight$=3.
MLP的训练参数设置为$remove_head = 20, vocab_size = 2000, summary_weight = 5, bagging_iter = 10$。

| max_depth/algorithm | Bagging | Base |
|---|---|---|
| 5 | 1.25416 | 1.26649 |
| 10 | 1.19725 | 1.22872 |
| 20 | **1.12421** | 1.20581 |

表 6: *max_depth*的实验结果(Bagging+DT)

| T | RMSE | Actual T |
|---|---|---|
| 1 | 1.22800 | 1 |
| 3 | 1.17993 | 3 |
| 10 | **1.15029** | 4 |

表 7: 迭代次数的实验结果(AdaBoost+DT)

| max_depth/algorithm | AdaBoost | Base |
|---|---|---|
| 5 | 1.20009 | 1.26649 |
| 10 | 1.17993 | 1.22872 |
| 20 | **1.03229** | 1.20581 |

表 8: *max_depth*的实验结果(AdaBoost+DT)

| Algorithm | RMSE | Place on Leaderboard |
|---|---|---|
| Bagging+SVM | 0.93011 | 15 |
| AdaBoost+SVM | **0.79856** | **3** |
| Bagging+DT | 1.12421 | 27 |
| AdaBoost+DT | 1.03229 | 23 |
| Bagging+MLP | 0.92602 | 15 |

表 9: 各方法在Leaderboard上的结果

在调整了特征提取参数，尤其是增加了特征维数后，模型表现有了极大的提高，RMSE达到了0.80以下。可以预想，进一步增加特征维数可以继续提高效果，但由于硬件条件（内存不足）且没有必要，我没有进一步调整参数。

# 4 Analysis & Discussion

## 4.1 Result Analysis

由上可见，**Bagging**和**AdaBoost**两种集成学习算法对各种分类器都有不同程度的提高，其中**AdaBoost**的提高效果更为明显。由于训练时间有限，无法充分实验**Bagging**的迭代次数较高时的效果。仅对比迭代次数为5时最优模型的效果，**Bagging**和**AdaBoost**将SVM的RMSE分别降低了5.93%和13.77%，而将DT的RMSE分别降低了6.77%和14.39%，可见大致相同的训练时间内AdaBoost的提升效果强很多。但由于Bagging的迭代次数可取任意次，其效果理论上仍有提升空间，囿于时间原因无法实验。

同时可以发现，SVM的表现明显强于DT。在参数为scikit-learn库默认参数，特征提取参数为*remove_head*=20, *vocab_size*=2000, *summary_weight*=5的情况下: 不使用集成学习算法时，SVM的RMSE相对DT低了15.01%; 而使用集成学习算法时，SVM的最好效果比DT低了13.83%，均有着明显的十分明显的优势。

为了比较，我还实现了一个最基础的MLP，并使用**Bagging**进行了效果优化。其在没有进行任何调参的情况下取得了0.92602的RMSE值，比同等设定的SVM降低了0.00409。可见，神经网络对于自然语言的语义信息提取能力是相当强的。

## 4.2 Discussion

### 4.2.1 Difference of Classifiers

对于DT表现明显弱于SVM，我提出了两个假设:

1. DT在进行分类时只能将不同的类别当做各自独立处理，但这个任务的本质其实是回归问题，即不同label对之间的相似度是不同的（比如说，label为4的sample比label为1的sample更接近label为5的sample）。DT无法体现这类信息，而因为可以认为label为4的samples到label为5的samples在特征向量空间的距离小于label为1的samples，所以SVM能够捕捉这种区别。
2. 由实验结果可见，DT的效果随着*max_depth*的增加而增加，但由于设备与时间限制无法得知这个趋势在什么范围内成立。猜测是因为自然语言的复杂性较高，DT需要很高的树高才能充分获取到语义信息。

进行了一个实验验证假设1的正确性：比较SVM与DT的正确率（类别由弱分类器加权投票得到）。这样相当于在评价时将所有类别视为互相无关联的，不会出现不同的错误预测的评价不同的情况，可以部分抵消SVM的优势。(实验中SVM和DT的参数均取sklearn实现的默认

值，其他参数均相同)

可见，比起RMSE，SVM在准确率上的相对优势要明显小 (当然，由于RMSE为非线性

| Metric | RMSE | Accuracy |
|---|---|---|
| SVM | 0.92658 | 0.6411 |
| DT | 1.17985 | 0.5875 |
| SVM superior by | 21.47% | 9.12% |

表 10: 不同Metric下的实验结果

函数，直接比较相对增大(减小)并不严谨，但仍可以大概得出结论)。这说明假设1应该是SVM表现更好的原因之一。

#### 4.2.2 Difference of Ensemble Learning Algorithms

Boosting的提高幅度高于Bagging与课件中描述相符。这里对其原因提出一个假设：由于训练集已足够大(220k samples)，所以不使用集成算法训练就可以将variance降到比较低，但不能有效降低bias; 而Bagging方法主要能够降低variance，Boosting能够降低bias，所以Boosting取得了更好的效果。

为此，进行了一个小实验来证明：仅采用训练集中的10000个sample进行训练，测试使用Bagging和AdaBoost($T = 5$)比起不使用集成学习和的效果提升，与全测试集的结果做比较(括号内为相对Base方法RMSE的降低比例):

| Algorithm/Data Size | Small | Full |
|---|---|---|
| Base | 1.29601 | 1.03865 |
| Bagging | 1.03166(19.85%) | 0.97045(5.93%) |
| AdaBoost | 1.22805(5.24%) | 0.88955(14.36%) |

表 11: 不同数据大小的实验结果

可见，在样本数量很小时，Bagging的提升效果尤为明显，而AdaBoost效果较为一般，与在数据全集上训练的结果截然不同。初步验证了我的假设。