Ensemble Learning

石大川 <sdc17@mails.tsinghua.edu.cn>

2020 年 5 月 20 日

目录

1 介绍 **2**

2 实验设计 **2**

2.1 数据集划分 2

2.2 特征提取 2

2.2.1 构建语料库 2

2.2.2 Word2Vec 2

2.2.3 PCA 3

2.3 基本分类器 3

2.3.1 SVM 3

2.3.2 Decision Tree 3

2.4 集成算法 3

2.4.1 Bagging 3

2.4.2 Adaboost.M1 3

3 主要实验结果 **4**

4 实验结果分析 **5**

5 其他 **6**

6 Kaggle **6**

1 介绍

在本次实验中我基于SVM和Decision Tree实现了Bagging和AdaBoost.M1的Ensemble Learning算法。在Kaggle的测试集上表现最好的是SVM和Bagging的组合，RMSE为0.85557。

2 实验设计

2.1 数据集划分

因为Kaggle上不方便批量测试且测试次数有限制，所以我在train.csv中随机划了5%，也即11000条数据来作为验证集。多次实验结果表明划分出来的验证集上的RMSE和test.csv在Kaggle上测出来的RMSE是很相近的，误差一般在±0.01。本文后续的实验结果如无特殊说明，均指的是在自己划分的验证集上的测试结果。

2.2 特征提取

2.2.1 构建语料库

对于summary和reviewText的内容，合并转小写然后用正则r'[a-zA-Z]+'匹配，最后去掉nltk.corpus中提供的英文停用词。

对于reviewerID和asin这两项内容，我尝试过三种处理方法：

⓪ 不考虑这两项内容

① 在上一步构建好的语料库中直接加入这两项作为词汇表的一项

② 在上一步构建好的语料库中加权加入这两项作为词汇表的一项

总的来说⓪的处理方式在RMSE指标上是最优的。以8个SVM组成的bagging模型为例，三种处理方法的RMSE分别为：

⓪ 0.8953

① 0.9008（reviewerID与asin权重和其他词汇相同）

② 1.0329（reviewerID与asin权重均设为其他词汇的8倍）

对于unixReviewTime这项内容，我认为参考价值不大，没有考虑使用

2.2.2 Word2Vec

Word2Vec的过程我尝试了：

⓪ 自己训练64d, 128d, 300d模型

① 使用pre-trained模型glove.6B.100d/glove.6B.200d/glove.6B.300d/glove.6B.100d/glove.42B.300d

② 使用pre-trained模型GoogleNews-vectors-negative300

总的来说⓪中的128d的模型在RMSE指标上是最优的。以8个svm组成的bagging模型为例，自己训练的128d模型RMSE为0.8953，而GoogleNews-vectors-negative300则为1.3338，其他的pre-trained模型略好于GoogleNews-vectors-negative300但都比自己训练的要差。我认为主要原因是pre-trained模型的训练语料和本次实验的语料在分布上相差比较大，从而pre-trained模型的embedding也不太适合用于本次实验。

此外自己训练的模型64d和300d模型的RMSE指标都要差于128d。我认为对于64d的模型可能因为是维度太小，词向量容易冲突，对于300d的模型可能是因为维度太高，造成SVM/Decision Tree分类困难。

TF-IDF 我也试了一下，最后生成的特征矩阵太大，存储和分类的时间成本过高所以放弃了。

2.2.3 PCA

做 PCA 能挑出更重要的特征，减小特征矩阵的规模，降低存储成本，同时更少的特征可以使得 SVM/Decision Tree 分类更快。我尝试了降维到 $\frac{1}{2}$ 和 $\frac{1}{4}$，以 8 个 svm 组成的 bagging 模型为例，RMSE/训练时间为：

⓪ 128d 模型：0.8996/69.8467s

① 64d 模型：0.9165/44.9486s

② 32d 模型：0.9665/33.3858s

一方面 128d 的模型是 RMSE 指标上最优的，说明用这个维度编码本次实验所用数据集的词汇表是比较合适的。另一方面可见维度越高，训练基本分类器的时间也会越长。

2.3 基本分类器

2.3.1 SVM

SVM 本身是做二分类的，对于多分类的任务要用 SVC 或者 LinearSVC。

SVC 复杂度 O(n^3)，官方文档里说样本量超过 10000 拟合起来就很慢了，我实际测试的结果也是收敛极缓慢，所以改用了文档推荐的 LinearSVC

LinearSVC 我试验出的比较好的参数配置是：

⓪ 多分类方法 (multi_class) 用'ovr'，也即一对多法

① 类分配权重 (class_weight) 用'balanced'

② 最大迭代次数 (max_iter) 为 60。试验发现迭代次数超过 60 后 SVM 的 RMSE 基本没有提升了。

2.3.2 Decision Tree

决策树我用的是 DecisionTreeClassifier，我试验出的比较好的参数配置是：

⓪ 类分配权重 (class_weight) 不用'balanced'，也即设为 None

① 叶子节点的最少样本数 (min_samples_leaf) 设为 5，不让叶子节点的样本数过小能有效减轻过拟合

② 最大树深 (max_depth) 不做限制，试验发现限制了叶子节点的最少样本数后树深 >8 时对 RMSE 的影响很小，树深过小时 RMSE 会下降。

2.4 集成算法

2.4.1 Bagging

用的是课件所述的Bagging算法，没做什么修改

2.4.2 Adaboost.M1

用的是课件所述的Adaboost.M1算法,对于蓝框圈出来的地方,我在逻辑上稍微做了些修改：当 $\epsilon_t > 0.5$ 时我让权重重新初始化为 $1/N$，重新进行本轮循环。原来的 Adaboost.M1 算法我在实验中发现有时训练了 2 个弱分类器就 abort 了，我觉得这个数量太少了而且弱

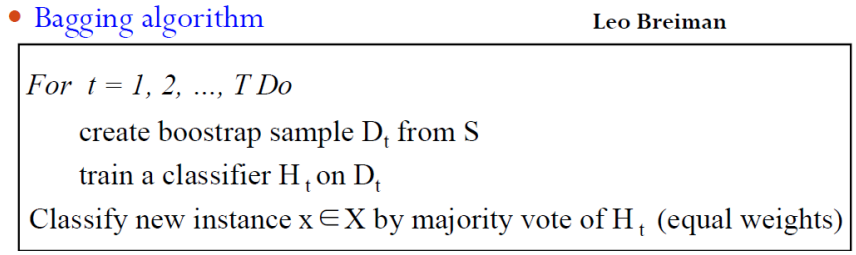

Figure 1: Bagging Algorithm

分类器数量不稳定也不方便后续进行统计。做了这个修改后能够保证每个弱分类器都满足 $\epsilon_t \leq 0.5$ 的同时，弱分类器也能达到指定的数目。

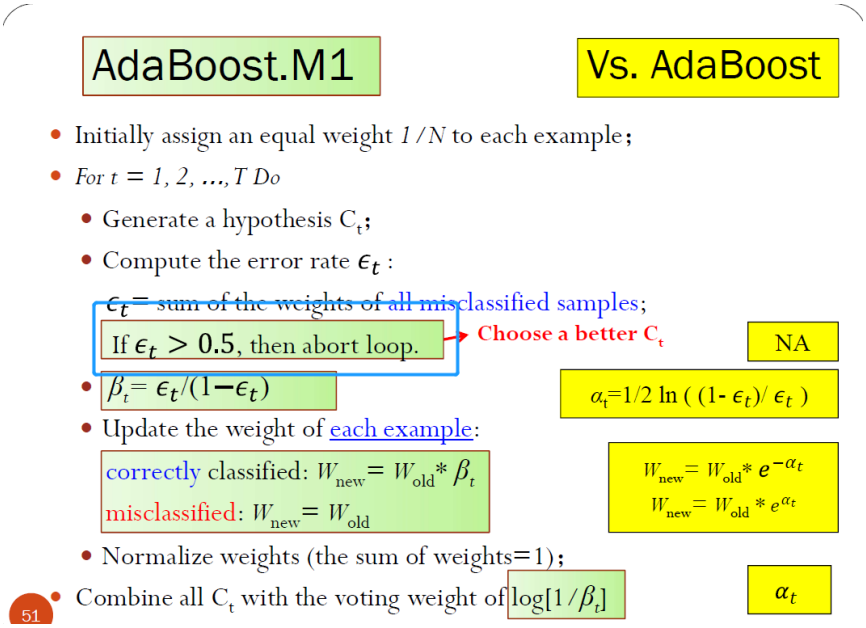

Figure 2: Adaboost.M1 Algorithm

3 主要实验结果

对于 SVM 和 Decision Tree 与 Bagging 和 AdaBoost.M1 的 4 种组合关系，每组我训练了 1024 个弱分类器，弱分类器类别，数目，集成方式与 RMSE 的关系见Figure3。在 Kaggle 上四组的 RMSE 具体表现见表 1

Algorithms	Bagging	Adaboost.M1
SVM	0.85470	0.85557
Decision Tree	0.92013	0.96504

表 1: RMSE on Kaggle

从 1024 个弱分类器的 RMSE 表现来看，本地验证集测试表现最好的是 SVM 和 Adaboost.M1 的组合，在 Kaggle 上表现最好的是 SVM 和 Bagging 的组合，在本地验证集和 Kaggle 上这两种组合的 RMSE 表现相差都是微乎其微的。

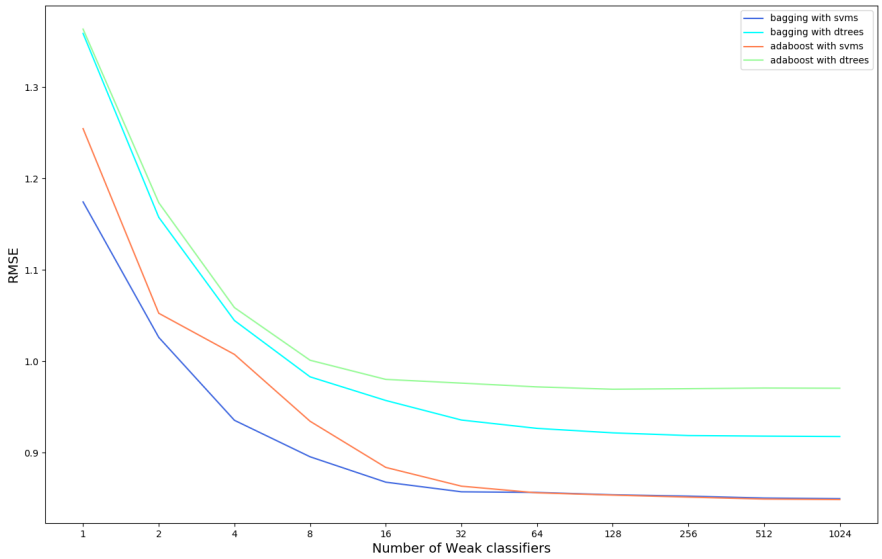

Figure 3: RMSE given by different classifiers and ensemble learning algorithms

4 实验结果分析

⓪ 对于四种组合关系，当弱分类器的数量达到 64 之后，RMSE 指标改善的程度就很小了。对于这次实验数据的特征空间，编号 64 之后的弱分类器和编号 64 之前的弱分类器同质化就比较严重了。

① 总的来看，弱分类器数目和集成算法相同的情况下，SVM 在 RMSE 指标上优于 Decision Tree。主要原因我认为是 Decision Tree 比 SVM 更容易过拟合。依据就是在训练集上的 RMSE，Decision Tree 和 SVM 是接近的，但是一旦用测试集来检验，Decision Tree 的 RMSE 就会明显比 SVM 高。

② 对于 SVM 来说，实验曲线表明在弱分类器数量较少的情况下，bagging 在 RMSE 指标上是优于 Adaboost.M1 的，但是我认为这主要是因为第一个弱分类器的性能差距有些大，可以参见Figure3中深蓝和橙色两条曲线的起点位置。由于 Adaboost.M1 算法后续的弱分类器的样本权重要依赖于第一个弱分类器的结果，所以第一个分类器效果差就导致后续要想得到效果好的分类器难度更高，而 bagging 算法则没有这样的依赖问题。所以从Figure3可见在弱分类器数目小于 64 之前 bagging 的 RMSE 是好于 Adaboost.M1 的，同时 Adaboost.M1 的性能也是在随着弱分类器的数目增加而逼近 bagging 的，到了 64 个弱分类器之后 Adaboost.M1 的 RMSE 就优于 bagging 了。所以我认为总体上来看，Adaboost.M1 对于 SVM 来说是优于 bagging 的。

③ 对于 Decision Tree 来说，一方面因为弱分类器本身效果较差，因此 bagging 和 Adaboost.M1 的效果都较差。另一方面，Adaboost.M1 对于 Decision Tree 来说差于 bagging，原因我认为还是在于弱分类器本身效果太差，在一些不那么困难的样本上弱分类器尚且效果不理想，如果还用 Adaboost.M1 去强调更难分类的样本可能会导致更多简单的样本在新的弱分类器上被分类错误。也就是说 Adaboost.M1 对于难分类样本的权重更新要取得好的结果，是要以在不难分的样本上已经取得了较好的结果为基础的，如果基础尚且不牢还想着去解决更难的样本问题，我认为是不合适的。

④就1024个弱分类器来看,最优的组合是SVM和Adaboost.M1,分析已经在②和③中叙述过了。

5 其他

因为本次实验数据量有些大，所以写并行还是很有必要的。对于 Bagging 来说，模型层面的并行很容易做到，但是对于 Adaboost.M1 来说下一个弱分类器的输入对上一个弱分类器的输出有依赖关系，所以模型层面的并行就变得比较困难。对于 Bagging 的算法我做了模型层面的并行，并行训练 8 个弱分类器。对于 Adaboost.M1 的算法我尝试了去做模型内部的并行，也就是对弱分类器的 fit 和 predict 过程做并行，但是没有成功。所以对我来说 Adaboost.M1 训练的时间成本很高，1024 个 SVM 训练了 14 个多小时，而 Bagging 算法的 1024 个 SVM 只用了两个小时。所以用 ensemble learning 的方法来解决实际问题时不仅仅要考虑到 RMSE 等准确度评价指标，还要综合考虑到时间成本或者去优化时间成本的难度。

6 Kaggle

我在 Kaggle 上的 User Name 是 sdcveg，Display Name 是 sdc17，截止至写报告的时间 rank 为 15/55.