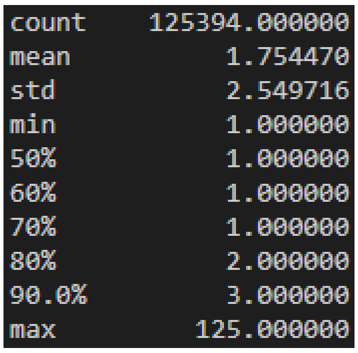
ML Project 2 -- Ensemble Learning

Experiment Design

Feature Extraction

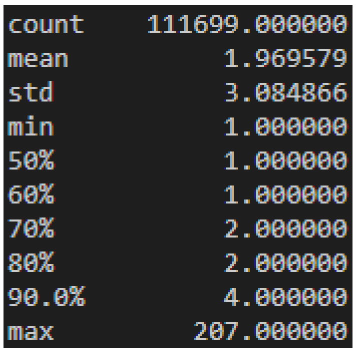
There are five fields in the dataset.

- Summary about 'reviewerID' is as below. It shows over 70% of the reviewers only wrote one review, so this field does not contain much information and we just ignore it.



```
count    125394.000000
mean          1.754470
std           2.549716
min           1.000000
50%           1.000000
60%           1.000000
70%           1.000000
80%           2.000000
90.0%         3.000000
max         125.000000
```

- Summary about 'asin' is as below (it is the id for each product). Similarly, over 70% of the products only have one review. We ignore this field too.



```
count    111699.000000
mean          1.969579
std           3.084866
min           1.000000
50%           1.000000
60%           1.000000
70%           2.000000
80%           2.000000
90.0%         4.000000
max         207.000000
```

- 'unixReviewTime' is totally irrelevant to rating. We drop it here.
- 'summary' and 'reviewText' is the most useful fields. Maybe it's better to give words in 'summary' a bit more weight, but for simplicity, we use equal weights here. I followed the following steps to encode the text:
  - Tokenizing the text (split into words, remove punctuation, etc.)
  - Removing stop words
  - Stemming
  - Encoding. I tried two different ways to encode the review text:
    - bag of words

        - tf-idf

    that gives us a sparse matrix with shape (n_samples=220000, n_features=vocabulary_size=~40000)

Base Classifier

I tried five different base classifier:

- sklearn.svm.LinearSVC
- sklearn.svm.LinearSVR
- sklearn.tree.DecisionTreeRegressor
- sklearn.tree.DecisionTreeClassifier
- sklearn.nerual_network.MLPClassifier

The "regressor" is different from "classifier" in that it gives continuous output rather than discrete output. Thus its score suffers from "out-of-bound" predictions (< 1 or > 5), and I simply use `y_pred = max(1, min(5, y_pred))`.

Since this is a multi-class classification problem, one-vs-the-rest is used in construction of SVM. That is, we build one SVM for every class.

Ensemble Learning Algorithm

I used two algorithms: bagging and adaboost-M1.

Bagging

Train n models separately, and use the average of predictions as final output. I take advantage of multi-processing, so the training went very fast.

Adaboost-M1

Train n models one by one, and in each iteration, lower the weights of already correctly classified examples. Final output is a weighted average of predictions of each model, the lower its error rate, the higher its weight.

Since adaboost can only be used in binary classification, I modify it a bit. A prediction is considered "correct" if $|y_{pred} - y_{truth}| < threshold$, where threshold is a parameter.

Results

Bagging

id	encoder	base learner	# of learners	mean valid rmse of one learner	valid rmse
1	bag of words	svm(classifier)	32	1.08	0.887
2	bag of words	svm(regressor)	32	1.25	1.078
3	bag of words	dt(regressor)	32	1.01	0.909
4	bag of words	mlp	32	1.03	**0.781**
5	tfidf	svm(classifier)	32	**1.00**	0.896
6	tfidf	svm(regressor)	32	1.09	1.090
7	tfidf	dt(classifier)	32	1.20	0.951
8	tfidf	dt(regressor)	32	1.04	0.907

adaboost-m1

id	encoder	base learner	iterations	valid rmse
9	bag of words	svm(classifier)	32	0.969
10	bag of words	svm(regressor)	14	1.151
11	bag of words	dt(regressor)	32	**0.955**
12	bag of words	dt(classifier)	32	0.974

I've fixed the random seed (19990321), so everyone can reproduce the exact same results as above.

Score on kaggle

I selected the required four models (ada+dt, ada+svm, bagging+dt, bagging+svm) plus the model with best performance (bagging+mlp).

id	model	score on kaggle
1	bagging+svm(classifier)	0.88193
3	bagging+dt(regressor)	0.91702
9	adaboost+svm(classifier)	0.97228
11	adaboost+dt(regressor)	0.96348
4	bagging+mlp	**0.80706**

Until the time I write this report, the ranking of my best submission (bagging+mlp) is **2/19**.

Discussion

About Test Results

From the above tables, bagging is significantly better than adaboost.

(note that a predictor that outputs the average of train_y only have rmse of **~1.1**, so 0.9 and 0.95 is a big difference)

That is probably due to poor threshold setting used in adaboost. Actually adaboost is only suitable for binary classification, and I'm not sure whether my way of transforming a multi-class classification into a binary classification works properly. Intuitively, there may be some examples where prediction is far from real score (e.g. 1 vs 5), and adaboost makes great efforts to fit these points by lowering weights of other points, leading to poor results.

About Performance

Training speed is also an important factor, especially when we have limited computing resources.

Bagging is extremely suitable for parallelism, because the models are all independent. Thus, it is very fast to train, given tens of CPUs on our workstation.

Adaboost cannot be parallelized because each model depends on the previous one. It is much slower to train.

Different base learners take roughly same time to train. MLP is slower, but its results are much better than any other base learner.

About Characteristics of Algorithms

Bagging is mainly used to reduce variance. It is more helpful on base learners with higher variance (e.g. nerual networks). The improvement in rmse using 32 base learners compared with 1 base learner is the greatest with mlp. (1.03->0.78)

Adaboost is sensitive to noise. Unfortunately, noise is very common in NLP tasks, so test results may suffer a lot.