Stacked Generalization Ensembles: Comparing Stacked Generalization with Other Classification Models

Tianlu Chen TCHEN2@SWARTHMORE.EDU
Swarthmore College, 500 College Ave, Swarthmore, PA 19081, USA

Irene Tang ITANG1@SWARTHMORE.EDU
Swarthmore College, 500 College Ave, Swarthmore, PA 19081, USA

Do June Min DMIN1@SWARTHMORE.EDU
Swarthmore College, 500 College Ave, Swarthmore, PA 19081, USA

Abstract

Stacked Generalization (SG) is an ensemble approach to machine learning that combines a set of supervised-learning models into a meta-learner, collecting predictions produced by multiple supervised base learners and using those predictions to train a higher-level supervised learning classifier. Despite being a theoretically appealing and conceptually straightforward idea, Stacked Generalization has been largely overshadowed in popularity by other ensemble methods such as AdaBoost, Bagging, and Random Forests. To determine whether Stacked Generalization is competitive with other classification options, we conducted two experiments exploring SGs in general and four experiments comparing its performance with those of other individual and ensemble supervised learning algorithms. As expected, we found that Stacked Generalization is indeed a powerful ensemble method that offers a simple and versatile approach to classification.

1. Introduction

Inspired by the human group decision making process, ensemble learning is an approach to machine learning where multiple models are trained to solve the same problem, and a voting scheme is applied to determine a consensus among their predictions. Compared to traditional approaches where only a single hypothesis is formed, ensembles construct a set of hypotheses which gets combined into a stronger prediction model. Ensembles provide an advantage by reducing the effect of errors made by individual classifiers.

Stacked Generalization (SG) is one such variation of ensemble learning. SG combines a set of models into a meta-learner, and it is typically used under a supervised learning setup (Wolpert, 1992). Contrary to other ensemble methods such as AdaBoost, Bagging, and Random Forests where all of the individuals models tend to be trained using the same learning algorithm and the same voting scheme is applied to their predictions independent of what the data point is, Stacked Generalization prefers to train models on a diverse set of base learning algorithms and to use another classifier to figure out the combining mechanism. (Breiman, 1996).

The intuition behind SGs is to first train a diverse set of base learners ("Level-0" learners, or h_i), and then use their predictions as features to train a high level learner ("Level-1" learner, or "generalizer"). The key idea is to have the generalizer learn how accurately each base-learner performs on each section of the data, so that it can adjust their weights accordingly.

In Experiments 1 and 4, we explored Stacked Generalization in general by confirming the results of previous research regarding optimal implementations of the algorithm. In Experiments 2, 3, 5, and 6, we further compared Stacked Generalization's performance with those of other individual and ensemble schemes.

We ultimately confirmed the prior discovery that it is better to train L1 generalizers on base classifiers that output predictions as probability distributions rather than single labels. However, we were inconclusive in confirming whether or not it is better to train L1 generalizers using a linear or a non-linear algorithm. Furthermore, we ultimately concluded that Stacked Generalizers can indeed outperform other supervised learning methods.

CPSC 66 Machine Learning Proceedings, Swarthmore College, Fall 2017.

2. Background

A Stacked Generalization ensemble consists of two levels of classifiers: a set of "L0" base learners, and an "L1' generalizer. Any supervised learning algorithm (e.g. K-nearest neighbors, Decision Tree, Support Vector Machines, Logistic Regression) may serve as base learners or generalizers.

To train an SG model, each of its base learners are first trained according to their respective algorithms on the same dataset, with a portion of the dataset held aside for training the generalizer. The generalizer is then trained with the held-aside dataset, using the each of the base learners as features, their predictions on the held-aside dataset as feature values, and the set of labels on the held-aside dataset for reference.

As illustrated in Figure 1, the model is not interactive as intuition flows only from the base learners to the generalizer. Furthermore, all of the base-learner models are independent from each other.

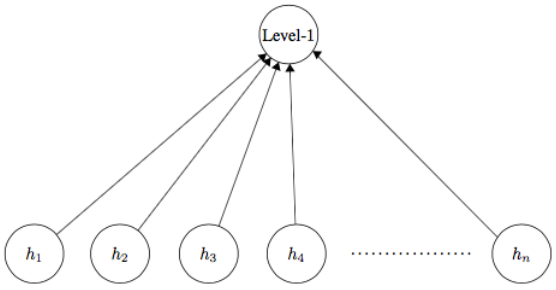

Figure 1. Intuition of a Stacked Generalization Classifier

2.1. Related Work

After Wolpert's proposal of Stacked Generalization (Wolpert, 1992), various follow-up discussions and extensions of the model have been developed. The following three particularly interesting ideas stood out to us, and we have incorporated them in our experiments.

2.1.1. Base Learners: Labels as Probability Distributions

Ting et al. (Ting & Witten, 1999) empirically showed that Stacked Generalization performed with higher accuracy when the base learners output labels as a probability distribution, compared to when they output just the most-likely label. Since probability distributions give insight into the confidence of the base-learners about their predictions, the generalizer is able to take into account higher and lower levels of certainty when it is adjusting weights. We confirim Ting et al.'s claim in our Experiment 1.

2.1.2. Use A Linear Level-1 Classifier

Breiman first demonstrated the success of Stacked Generalization in the setting of using Linear Regression to train the generalizer, coining the term *stacked regressions* to refer to the family of generalizers that uses a linear algorithm to combine their base learners' prediction (Breiman, 1996). Linear Regression is often the default algorithm because it is easy to compute the regressions and to cary out large experiments. Ting et al. have also empirically demonstrated that generalizers trained on the multi-response least squares linear regression algorithm peformed better than generalizers trained using other algorithms (Ting & Witten, 1999). However, these empirical results are becoming dated with respect to modern developments in machine learning and we are not certain if linear methods would still outperform non-linear classifiers such as neural networks. We compare the effect of a non-linear Neural Network classifier with a linear Linear Regression classifier as the level-1 learner in our Experiment 4.

2.1.3. Base Learners: Heterogeneous vs Homogenous

Breiman reasons that Stacked Seneralization performs better when base learners "are not overly similar", preferably using a heterogeneous set of different learning algorithms to train the base learners. The diversity of the members in an ensemble is an important deciding factor in its ability to generalize well. The motivation of using a heterogeneous set of base learners is to exploit independence between them, since the generalization error can be reduced dramatically by avarageing out the error. We followed Brieman's suggestion in our experiments, and our set of base classifiers consisted of

operates with two tiers of classifiers: a "Level-1" high-level classifier, and an ensemble of "Level-0" low-level classifiers.

2.2. Applications and Extensions

Variations of Stacked Generalization have been widely used in natural language processing (Malmasi & Dras, 2017), collaborative filtering for recommender systems (Sill et al., 2009), and teenage distress studies (Dinakar et al., 2014).

Furthermore, Ozay et al. (Ozay & Yarman Vural, 2012) performed an extension of Stacked Generalization that used fuzzy $k-$nn base learners trained with subsampled data and features, and showed that such a stacked ensemble performs better than AdaBoost, Random Space, and Rotation Forest. In addition, Sill et al's (Sill et al., 2009) performed an extension that used feature-weighted linear stacking to obtain meta-features to achieve higher accuracy.

3. Methods

3.1. Training

Stacking is about learning to combine predictions from different classifiers. In order to achieve this, the training portion of the dataset needs to be further split into two sets. One training set is used to train the level-0 base learners, and the other is further processed to create a training set for the level-1 classifier. Specifically, the base-level predictions (presented as either a single label or a vector representing probability distribution) are used as features, and they are paired with the correct labels to form training examples for the level-1 classifier to train on.

Since training occurs only once per each base classifier, and since each base classifiers can be trained independently, training a Stacked Generalization model is easily a parallelizable process.

Algorithm 1 explains the pseudocode for the training phase. h_i denotes a base classifier and l denotes the level-1 classifier.

Algorithm 1 Training an Stacked Generalization Classifier

Input: $h_1, h_2, ..., h_n$, l, Dataset (X, Y)
Output: trained classifier $SG = (h_1, h_2, ..., h_n, l)$
1. Split (X, Y) into 2 partitions, (X_1, Y_1) and (X_2, Y_2).
2. Train each h_i on (X_1, Y_1).
3. $X_l \leftarrow \emptyset, Y_l \leftarrow \emptyset$
4. For $(x_{2_j}, y_{2_j}) \in (X_2, Y_2)$
    For each h_i, compute $\hat{y_i} = h_i(x_{2_j})$.
    $X_l, \leftarrow X_l \cup (\hat{y_1}, \hat{y_2}, ..., \hat{y_n})$
    $Y_l \leftarrow Y_l \cup y_{2_j}$
5. Train l using (X_l, Y_l)
return $SG = (h_1, h_2, ..., h_n, l)$

3.2. Prediction

Given an example datapoint to predict, the Level-1 learner l by itself is not able to form a prediction using its original features. In order to make the prediction, the base classifiers must first cast their votes. Then, the l will consider each of those predictions to form a more accurate one.

Algorithm 2 Prediction using an SG Classifier

Input: SG, example point x
Output: prediction $\hat{y}$
1. For each h_i, compute $\hat{y_i} = h_i(x)$
2. Compute $\hat{y} = l(\hat{y_1}, \hat{y_2}, ..., \hat{y_n})$.
return $\hat{y}$

After l makes a prediction and obtains an error signal, it is possible propagate the L1 prediction back to the base classifiers. However, we do not pursue such technique in our experiment. Instead, we focus on using comparing different base classifiers and level-1 learners.

3.3. Base Classifiers

Any supervised learning classification algorithm can be used as a base classifier. This includes Support Vector Machines (SVM), Naive Bayes, Decision Trees, $K-$-nearest-neighbors, and Linear and Logistic Regression models. Even a neural network can be used, but in this study we only use neural networks as level-1 learners.

The abundance of choice presents a challenge: to completely explore the hypothesis space generated by different combinations of classifiers is impossible. We would not only need to take into account the quantity of each type of classifier, but also the hyperparemeters within a base classifier. Therefore, we focus on contrasting heterogeneous and homogeneous sets of base learners.

3.3.1. Heterogeneous set of base learners

Following Breiman's advice, we aim to maximize diversity and minimize the similarity between base classifiers. We achieve this by including each type of classifier only once in our collection of base classifiers. This allows us to use each model's inductive bias to guarantee minimal correlation among base learners. We use a collection of four base learners: Support Vector Machines, Naive Bayes, Decision Tree, K-nearest Neighbors.

3.3.2. Homogeneous level-0 collection

Another way of creating diversity is to subsample data points and features, as done in the Random Forest method of selecting decision stumps. Although these methods could be used together with heterogeneous classifiers, we choose to focus on a case where the base collection consists of multiple instances of a single model. Specifically, we use decision trees with randomly chosen features, trained on subsampled data.

3.4. Level-1 Learners

As mentioned above, typically researchers use Linear Regression classifiers as the primary choice for the Level-1 learners, based on both theoretical and empirical validation by previous researchers. Since the datasets we use are binary classification tasks, in Experiments 1, 2, 3, 5, and 6 we also used logistic regression to train our Level-1 learner.

Nonetheless, in Experiment 4 we also test the effect of Stacked Generalization using a non-linear classifier as our Level-1 learner. Our non-linear model of choice is a Neural Network, as suggested by Wolpert (Wolpert, 1992). However, Ting et al., points that Neural Networks require more

training example in order to perform well, and that with less data samples Linear Regression will perform better empirically. Against Ting et al.'s advice, we note that with the recent advancements in artificial neural networks it may be noteworthy to see if a neural network can replace linear regression methods as level-1 classifiers.

3.5. Datasets and Evaluation Methodology

The results in this experiment reflect our performance using the Adult and Mushroom datasets, courtesy of the UCI Machine Learning Repository (Lichman, 2013).

The Adult dataset is a larger dataset containing 48842 instances of adults living in the United States according to a 1994 census. It contains thirteen features and a binary label indicating whether their income was above or below $\$50k$. Some features include age, workclass, education, and marital-status. The distribution of labels is a bit skewed(24.78% vs 75.22%).

The mushroom dataset is a smaller dataset containing 8124 instances of mushroom samples, predicting whether each sample is definitely edible or non-edible. Some attributes include cap-color, odor, gill-attachment, and stalk-color-above-ring. The distribution of labels is about equal(51.8% vs 48.2%).

We split each dataset into 80% training and 20% testing. We then performed cross-validation on the training set to generated paired data for Student's paired $t-$test. For each experiment, we conducted multiple paired tests. Using cross validation method, we generated 5 folds of paired data set, which then we used to train the algorithms. We then use paired sample T-test over 5 folds of data to evaluate our confidence in rejecting the null hypotheses. Before any of the experiments were run, we chose the threshold value for rejecting the null hypothesis as $p < 0.05$.

4. Experiments and Results

We designed six experiments that explore Stacked Generalization in genera and compare its performance with those of other individual and ensemble models. The implementation of the experiments requires the scikit-learn library (Pedregosa et al., 2011). The following is an overview of our research questions:

1. Does Stacked Generalization perform better when the base classifiers output labels as probability distributions or as just a single label?
2. Does Stacked Generalization perform better than the best individual base classifier?
3. Does Stacked Generalization perform better than performing a majority vote on the base classifiers?
4. Which is better: linear or non-linear Level-1 models?
5. Does Stacked Generalization outperform Bagging and AdaBoost?
6. Does Stacked Generalization outperform Random Forest?

The following table illustrates our choice of hyperparameters for each of the classifiers used.

Table 1. Parameters

Classifier	Parameters
Neural Network	Architecture and parameters
Logistic Regression	penalty = l2
Multinomial NB	$\alpha = 1.0$
SVM	$c = 1.0$ kernel=rbf $\gamma =$ auto
Decision Tree	criterion = entropy
kNN	$k = 5$
AdaBoost	base_estimator = dtree n_estimator = 4
Random Forest	n_estimator = 4
Bagging	base_estimator = kNN n_estimator = 4 max_samples = 0.5 max_features = 0.5

All of our experiments test the performance of two different implementations of SG classifier. The null hypothesis is that there is no difference in accuracy. Each numeric entry in the tables represents accuracy of the classifier on a corresponding data fold.

4.1. Base Classifiers: Labels as Probability Distributions vs. Single Label

Ting et al. suggested in 1999 that Stacked Generalization achieves better performance when the level-0 learners output distribution of labels than when they simply output label predictions (Ting & Witten, 1999). To explore SGs in general, we wanted to confirm this finding. In order to confirm this hypothesis, we trained two level-1 Logistic Regression classifiers: L uses base classifiers that output labels as probability distributions, and L' uses base classifiers that output single labels.

Table 2 and Table 3 summarize our results from the Adult and Mushroom datasets, respectively.

We rejected the null hypothesis within the 95% confidence interval on the Adult dataset. Unfortunately, our results on the Mushroom data set were not informative, since both have scored 1.0 on every fold. Nonetheless, this shows that Stacked Generalization performs better when the base classifiers output labels as a probability distribution, compared

Table 2. Experiment 1: Adult Data Set

Fold	Single Label	Distributions
1	0.8170	0.8218
2	0.8159	0.8214
3	0.8159	0.8202
4	0.8170	0.8214
5	0.8170	0.8216
t-score / $p-$value	t-score: -22.2136	p-value: 0.0000

Table 3. Experiment 1: Mushroom Data Set

Fold	Single Label	Distributions
1	1.0	1.0
2	1.0	1.0
3	1.0	1.0
4	1.0	1.0
5	1.0	1.0
t-score / $p-$value	t-score: N/A	p-value: N/A

Table 4. Experiment 2: Adult Data Set

Fold	SG Classifier	Cross Validation
1	0.8218	0.8170
2	0.8214	0.8170
3	0.8202	0.8170
4	0.8214	0.8170
5	0.8216	0.8170
t-score / $p-$value	t-score: -15.3935	p-value: 0.0001

Table 5. Experiment 2: Mushroom Data Set

Fold	SG Classifier	Cross Validation
1	1.0	1.0
2	1.0	1.0
3	1.0	1.0
4	1.0	1.0
5	1.0	1.0
t-score / $p-$value	t-score: N/A	p-value: N/A

to just a single label. Under this conclusion, for the rest of our experiments we trained our SG using base classifiers that output labels as probability distributions.

We confirim Ting et al.'s claim that such a distribution is more appropriate. Since probability distributions give insight into the confidence of the base-learners about their predictions, the generalizer is able to take into account higher and lower levels of certainty when it is adjusting weights.

However, we note that this accuracy comes at the cost of runtime complexity. Because training on probability distributions scales up the number of attributes that the generalizer needs to learn, there is a problem in scaling up the feature space, especially for datasets containing a wide range of labels. It is practically impossible to learn from large-scale multi-class prediction domains within an acceptable amount of time.

4.2. Stacked Generalization vs. Best Base Classifier

Since the goal of machine learning is to achieve the most accurate predictions within the least amount of time, using a Stacked Generalizer would only make sense if it indeed outperforms the best of its base classifiers. In order to test our hypothesis that SGs are indeed valuable, we compared the performance of an SG classifier with the best cross-validation performance of an individual base classifier.

Tabels 4 and 5 summarize our results from the Adult and Mushroom datasets, respectively.

Again, we reject the null hypothesis within the 95% confidence interval on the Adult dataset. Unfortunately, our results on the Mushroom dataset were still not informative, since both have scored 1.0 on every fold. Nonetheless, this shows that Stacked Generalization indeed performs better than the best classifier's cross-validation on every fold.

These results confirm the advantage of ensembles: that a collective vote on the hypothesis is better than a single view on the hypothesis. Each individual base classifier will err, but the effect of their errors is offset by the correct predictions of other classifiers, especially if their errors are uncorrelated. The aggregate opinion does not remove all the effects of bias, variance, and noise; but ensembles are certainly less susceptible compared to single models.

4.3. Stacked Generalization vs. Majority Vote

Since the goal of machine learning is to achieve the most accurate predictions within the least amount of time, using a Stacked Generalizer would only make sense if it indeed outperforms the majority vote scheme, which performs faster than training an additional L-1 generalizer model. In order to test our hypothesis that SGs are indeed valuable, we compared the performance of an SG classifier with the performance of an ensemble that uses a simple majority vote.

Tabels 6 and 7 summarize our results from the Adult and Mushroom datasets, respectively.

We rejected the null hypothesis within the 95% confidence interval on the Adult dataset. However, on the Mushroom dataset it appears that the majority-vote scheme actually performs nearly on-par with the Stacked Generalizer. This is probably not an indication that the majority-vote scheme is in general of equal performance to SGs, though; we reasoned that this is only because the Mushroom dataset is

Table 6. Experiment 3: Adult Data Set

Fold	SG Classifier	Majority Vote
1	0.8150	0.7969
2	0.8030	0.7969
3	0.8030	0.7969
4	0.8030	0.7989
5	0.8140	0.7979
t-score / $p-$value	t-score: 3.4503	p-value: 0.0260

Table 7. Experiment 3: Mushroom Data Set

Fold	SG Classifier	Majority Vote
1	1.0	0.999
2	1.0	0.999
3	1.0	0.999
4	1.0	0.999
5	1.0	0.999
t-score / $p-$value	t-score: inf	p-value: 0.000

Table 8. Experiment 4: Adult Data Set

Fold	L-1 Logistic Regression	L-1 Neural Network
1	0.8150	0.7537
2	0.8030	0.7537
3	0.8030	0.7537
4	0.8030	0.7537
5	0.8140	0.7537
t-score / $p-$value	t-score: -18.9980	p-value: 0.0000

Table 9. Experiment 4: Mushroom Data Set

Fold	L-1 Logistic Regression	L-1 Neural Network
1	1.0	0.5187
2	1.0	0.5187
3	1.0	0.5187
4	1.0	0.5187
5	1.0	0.5187
t-score / $p-$value	t-score: -inf	p-value: 0.0000

quite simple and that just a few defining features can predict whether a mushroom is poisonous or not, so it is extremely unlikely that all of the base classifiers would collectively misclassify a test point.

We concluded that while having a separate combining classifier indeed results in higher accuracy than having just a majority voting scheme, for simpler datasets we recommend just using the majority-vote scheme because it can achieve nearly the same results with much less complexity.

4.4. Level-1 Learner: Linear or Non-linear?

The Linear Regression algorithm has typically been the primary choice classifier to train Level-1 classifiers, since empirically they seemed to outperform non-linear classifiers such as Naive Bayes, Decision Trees, and $K-$nearest-neighbors. However, although it would take more time for them to train to approximate the same linear function as direct linear methods, we wanted to give non-linear mappings a shot. We hypothesized that Neural Networks might be able to approximate linear functions, especially with their rising popularity in modern machine learning research.

To test our hypothesis, we compared two SG classifiers: one with a Neural Network as the Level-1 classifier, and the other with a logistic regression classifier.

Tabels 8 and 9 summarize our results from the Adult and Mushroom datasets, respectively.

The results were not what we theorized. Our results claim that Logistic Regression is a better choice than Neural Networks for a Level-1 classifier, but we do not believe that this is necessarily accurate. We believe that this can be explained by the fact that our experiments were limited and inconclusive. Our choice of hyperparameters (number of neurons, number of layers, activation function, and loss function) was made without thorough grid search. Moreover, Neural Networks require an enormous amount of data in order to perform well, and even our larger Adult dataset was not large enough to fill enough of the hypothesis space. Choosing the network topology is not as trivial as we had made it, and neural networks require extensive data that our datasets could not provide.

Although the Neural Network classifier performed worse than the Logistic Regression control on both of our datasets, we note that the neural network classifier seems to do better on the larger Adult dataset than the smaller Mushroom dataset. This supports our hypothesis that if we only had more data, the Neural Network generalizer would have performed better.

4.5. Stacked generalization vs. Bagging, Random Forest and AdaBoost

Finally, we wanted to empirically compare Stacked Generalizers with the traditional Bagging, Random Forest, and AdaBoost ensemble methods to see if they are more deserving in popularity.

In order to make a reasonable comparison, all ensemble methods are applied with n_estimator set to 4. Tabels 10 and 11 summarize our results from the Adult and Mushroom datasets, respectively.

From the tables above, we can see the results differ vastly between a large data set and a small one.

For the adult data set, with 95% confidence interval, little can be concluded about SG compared to Bagging, although

Table 10. Experiment 5: Adult Data Set

Fold	SG Classifier	Bagging	RF	AdaBoost
1	0.8230	0.8136	0.8323	0.8434
2	0.8225	0.8250	0.8328	0.8434
3	0.8230	0.7846	0.8290	0.8434
4	0.8234	0.8412	0.8308	0.8432
5	0.8230	0.8218	0.8374	0.8434
t / p	SG / Bagging	0.8151 / 0.4608		
	SG / RF	-6.5604 / 0.0028		
	SG / AdaBoost	-119.2579 / 0.0000		

Table 11. Experiment 5: Mushroom Data Set

Fold	SG Classifier	Bagging	RF	AdaBoost
1	1.0	1.0	1.0	0.9354
2	1.0	0.9994	1.0	0.9354
3	1.0	0.9982	0.9994	0.9354
4	1.0	1.0	1.0	0.9471
5	1.0	1.0	0.9994	0.9354
t / p	SG / Bagging	1.3720 / 0.2420		
	SG / RF	1.6330 / 0.1778		
	SG / AdaBoost	26.6316 / 0.0000		

SG has a more stable performance across folds; RF is better than SG by a $p-$value of 0.0028; AdaBoost is significantly better.

As for the mushroom data set, SG classifier gets a steady 1.0 on each fold, winning over all other ensemble methods– although not significantly over Bagging and Random Forests. The $p-$values tells us that we are only moderately confident about SG's advantage (about 76% and 82%, respectively). But the $t-$score with AdaBoost gives high confidence that SG outperforms AdaBoost on the mushroom data set.

We suspect that Bagging, Random Forests, and AdaBoost become sensitive to outliers when the training set is not large enough. We concluded that on larger datasets, SG is only roughly as good as Bagging; on smaller datasets SG regularly achieves higher scores, while the others may be subject to overfitting.

Like Bagging, Random Forests, and AdaBoost, Stacked Generaliation is an ensemble method that aims to reduce bias and variance by incorporating compiling viewpoints on predictions. Like Bagging, SG also overfits on a portion of the training data in order to reduce variance. Like Random Forests, SG also trains on a set of weak learners. Like AdaBoost, SG also requires a training process for learning the weights of predictions from each of the base classifiers.

But unlike these three ensembles, Stacked Generalization "vertically" combines viewpoints by training another Level-1 layer, compared to "horizontally" combining of base learners by using a consistent voting scheme. Unlike Bagging, SGs' base learners train on the entirety of the training dataset. Unlike Random Forests, SGs' base learners train on the entirety of the feature space. Unlike AdaBoost, the learning process for the level-1 learner is measuring the performance of base classifier over different parts of the dataset.

4.6. Stacked generalization vs. Random forests

Random Forests can be seen as a special case of SG where the base classifiers are Decision Trees with random features and trained on subsampled data, while combining their predictions with a function that averages or outputs majority vote. To test this aspect of difference-by-level-1, we compare a random forest classifier with a SG classifier whose base classifiers are decision stumps taken from an RF classifier's trained estimators. To make sure that the comparison is fair, we trained the Random Forests with the full training data, whereas we only used the base training set to train the second Random Forests and extracted its decision stumps as base classifiers for our SG model.

Table 12. Experiment 6: Adult Data Set

Fold	SG Classifier	Random Forest
1	0.8342	0.8323
2	0.8342	0.8328
3	0.8374	0.8290
4	0.8374	0.8308
5	0.8184	0.8374
t-score / $p-$value	t-score: -0.0313	p-value: 0.0234

Table 13. Experiment 6: Mushroom Data Set

Fold	SG Classifier	Random Forest
1	1.0	1.0
2	1.0	1.0
3	1.0	0.9994
4	1.0	1.0
5	1.0	0.9994
t-score / $p-$value	t-score: 1.6330	p-value: 0.1778

The p-values from both the Adult and Mushroom datasets suggest that SG outperforms Random Forests, although the evidence is stronger on the adult data set, since the scores are quite similar on the mushroom one. These results show that if Random Forests are trained over its base learners using a linear model instead of taking a weighted vote of the probability estimate across the trees, then it would achieve better performance. Our results demonstrate that Stacked Generalization outperforms Random Forests even when using a homogenous set of base learners.

5. Conclusions

Our empirical results confirm previous empirical findings, and further suggest that Stacked Generalization performs better than traditional ensemble methods when data is limited. From this, we remark that certain ensemble methods could be easily turned into a Stack Generalization classifier to achiev even higher performance, just by replacing simple voting methods or other generalizing scheme with a high-level classifier.

6. Future Work

Since our Experiment 4 was inconclusive due to the difficulty in finding an optimal network topology and lack of data, we hope to further explore the relationship between network architecture and classifier performance. A systematic search of various hyperparameters needs to be performed.

We are also interested in changing the Stacked Generalization algorithm so that a complete retraining of the generalizer is not necessary when new data or additional base models are added. Currently, SG requires the generalizer to deal with only a fixed input dimension, determined by the number of base learners. As a result, in order to add or remove new base classifiers to the ensemble, it is necessary to retrain the generalizer. While the training set can be modified quickly by adding or removing few features, it might take a non-trivial amount of resource and time to retrain the generalizer.

Finally, we note that it is possible to construct a "deep" SG classifier consisting of more than one level of generalizer stacks. This classifier will closely resemble a deep neural network in topology, as the outputs of hypotheses H_i are fed into the next layer of "generalizing" hypothesis H_{i+1}. Of course, this will require complicated training set division scheme and a layer-wise training phase. While this method has the advantage of not requiring backpropagation, we expect this pseudonetwork to be either impractically slow or inaccurate. However, it might still be of some value to experiment with the idea.

Acknowledgments

We would like to express gratitude for Professor Ameet Soni for his guidance at every stage of this experiment.

References

Breiman, Leo. Stacked regressions. *Machine Learning*, 24 (1):49–64, July 1996. ISSN 0885-6125. doi: 10.1023/A:1018046112532. URL http://dx.doi.org/10.1023/A:1018046112532.

Dinakar, Karthik, Weinstein, Emily, Lieberman, Henry, and Selman, Robert. Stacked generalization learning to analyze teenage distress, 2014. URL https://www.aaai.org/ocs/index.php/ICWSM/ICWSM14/paper/view/8076/8108.

Lichman, M. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

Malmasi, S. and Dras, M. Native Language Identification using Stacked Generalization. *ArXiv e-prints*, March 2017.

Ozay, M. and Yarman Vural, F. T. A New Fuzzy Stacked Generalization Technique and Analysis of its Performance. *ArXiv e-prints*, April 2012.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Sill, J., Takacs, G., Mackey, L., and Lin, D. Feature-Weighted Linear Stacking. *ArXiv e-prints*, November 2009.

Ting, K. M. and Witten, I. H. Issues in Stacked Generalization. *Journal of Artificial Intelligence Research*, May 1999.

Wolpert, David H. Stacked generalization. *Neural Networks*, 5:241–259, 1992.