Practical 1: Stacked Regression for Predicting Chemical Toxicity

Abdul Saleh, Dean Hathout, Brendan O'Leary
{abdelrhman_saleh, dhathout}@college.harvard.edu, boleary@g.harvard.edu
abdulsaleh, dhathout, boleary134

February 16, 2019

1 Technical Approach

Cross-validation is one of the most commonly used methods for model selection. When one is faced with a multiplicity of learning algorithms, cross-validation can provide an estimate of the generalization accuracy of each algorithm, then the one with the highest accuracy is chosen. However, this winner-take-all strategy ignores the fact that even weak models can make good predictions in certain cases.

Motivated by this limitation, this practical explores **stacked regression** (Wolpert, 1992; Breiman, 1996; LeBlanc and Tibshirani, 1996) as an ensemble method for combining the predictions of multiple base models rather than choosing the best among them.

Many aspects of stacking have been described as "black art", to use Wolpert's words. And although different variations of stacking have been used to achieve state of the art results, to the best of our knowledge, there remains little consensus regarding how stacking should be applied in practice. *This practical will shed some light on this black art by exploring 2 approaches for hyperparemeter tuning with stacked regression.*

What follows is a technical description of stacked regression and our two hyperparameter tuning approaches. We analyze our methods in more detail in section 3.

Let $\mathcal{L}$ be our learning set $\{(y_n, \boldsymbol{x}_n), n = 1, \ldots, N\}$. Assume we have a set of k base learners $f_1(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})$. Stacked regression combines these base regressors by training a meta-learner on their *predictions*. The predictions from the base models are generated as follows: split $\mathcal{L}$ into J almost equal parts and let $\mathcal{L}^{(j)} = \mathcal{L} - \mathcal{L}_j$. Then train your predictors on $\mathcal{L}^{(j)}$ and make predictions on the unseen $\mathcal{L}_j$ for each $j \in \{1, \cdots, J\}$. This is equivalent to creating J folds, training on $J-1$ folds and making predictions on the left out fold.

The J sets of "out-of-fold" predictions can be used to create a new learning set $\mathcal{L}' = \{(y_n, \boldsymbol{z}_n), n = 1, \ldots, N\}$, where $\boldsymbol{z}_n$ is a k dimensional vector. A meta-learner is trained on this new learning set $\mathcal{L}'$ attempting to predict y_n from the predictions of the base models $\boldsymbol{z}_n$. In this practical, we use bounded least squares (BLS) with non-negative coefficients, following the recommendations of

Breiman (1996). Thus our final model is of the form:

$$f(\boldsymbol{x}_n) = \sum_k \alpha_k f_k(\boldsymbol{x}_n) \tag{1}$$

with α_k being the bounded least squares coefficients. Breiman (1996) showed that under some weak assumptions $\sum_k \alpha_k \approx 1$, implying that the meta-learner acts as a weighted average of the base predictions. This is more convenient than using a ridge regression meta-learner, for instance, since it does not guarantee predictions in the range $[\min_k f_k(\boldsymbol{x}), \max_k f_k(\boldsymbol{x})]$ as a weighted average does.

For our base models, we use ridge regression, gradient boosted regression, random forest regression, support vector regression, bagging regression, and k-nearest neighbours regression. We do not cover the inner workings of these base models as they are not relevant to our main focus of exploring hyperparameter tuning strategies for stacked regression.

We experiment with two approaches for hyperparameter tuning:

1. Our first approach uses base models with **default** (sklearn) parameters to create the out-of-fold predictions learning set $\mathcal{L}' = \{(y_n, \boldsymbol{z}_n), n = 1, \ldots, N\}$. Then the bounded least squares meta-learner is trained on this new learning set as described above.

2. Our second approach involves picking the "best" hyperparameters for the base models through grid search and cross-validation on the training set. The out-of-fold predictions learning set $\mathcal{L}'$ is created using these fine-tuned base models. Then the bounded least squares regressor is trained on this learning set as before.

Note that in the above approaches we rely on out-of-fold prediction to generate $\mathcal{L}'$ for the meta-learner. Out-of-fold predictions are crucial to stacking because we want our meta-learner to emphasize base models that generalize well to unseen, out-of-fold data. Consider the case were out-of-fold predictions are not used: we can overfit a base model by memorizing all the training data and the meta-learned will emphasize this overfit base model thinking that it performs well. This results in a stacked model that is also ovefit on the training data. Using out-of-fold predictions overcomes this problem of **data leakage** from the base models to the meta-learner.

Top Highlight

The hyperparameter tuning approaches we compare in this practical are aimed at exploring the effects of data leakage when fine-tuning the base models using cross-validation. The second approach we propose exposes the whole training set as we do cross-validation to pick the best hyperparamters for the base models. Even though the base models still generate out-of-fold predictions for the meta-learner, cross-validation picks the base models with the best out-of-fold performance inducing some indirect data leakage. This practical tests whether tuning the base models results in performance gains that justify this data leakage.

Method	RMSE ($\cdot 10^{-2}$)
Ridge Regression	3.527
Gradient Boosted R.	3.564
Random Forest	**3.502**
Support Vector R	3.595
Bagging Regressor.	***
KNN	***

Table 1: Test set performance in RMSE of single best predictor chosen through cross-validation. Lower is better.

Method	RMSE ($\cdot 10^{-2}$)
Default base models	
Avg Stacked	3.468
BLS Stacked	**3.464**
Ridge Stacked	3.465
Fine-tuned base models	
Avg Stacked	3.473
BLS Stacked	3.509
Ridge Stacked	3.517

Table 2: Test set performance of stacked models in RMSE. Lower is better.

2 Results

Table 1 shows test set results for our base models which were parameter-tuned using cross-validation on the training set[1]. The models in Table 1 were used to generate the out-of-fold predictions for our second approach.

We can see from Table 1 that random forest regressors are the strongest single model, achieving a root mean squared error of 0.03502.

Table 2 summarizes the performance of our stacked models with default and tuned base models on the test set. In addition to BLS stacking, we also experiment with a ridge meta-learner and with simply taking the mean of the base model predictions to produce a final prediction. We call these approaches Ridge Stacked and Avg Stacked in Table 2 above.

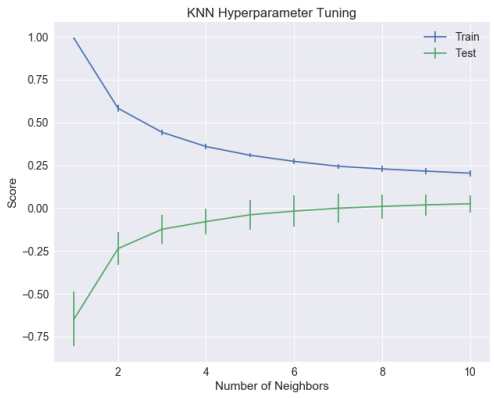
Top Result

Table 2 shows some interesting results. We see that stacking base models with default hyperparameters outperforms stacking fine-tuned base models for all the meta-learners tests. We can also see that using BLS as a meta-learner slightly outperforms the Ridge and simple "Avg" meta-learners when using the default base models, but the "Avg" approach performs best out of the three when using the fine-tuned base models. We discuss this more below. Additionally, for the fine-tuned base models, stacking using "Avg" achieved lower RMSE than the single best fine-tuned model.

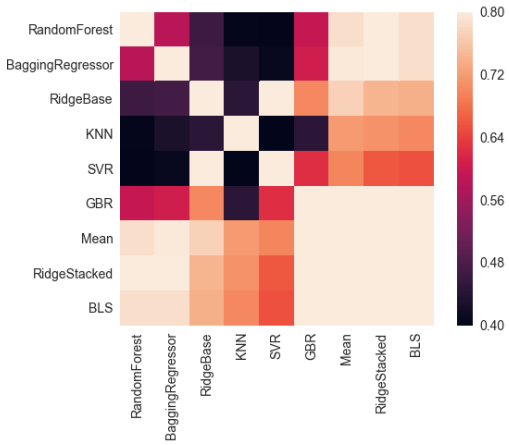
3 Discussion

The results in Tables 1 and 2 confirm our initial intuition that ensembles are better than using the single best model. The test set error for the best stacked model, as measured by RMSE, is lower than that of the best base models. The base models make different predictions as illustrated by Figure (b) and the meta-learners learn the best models to emphasize. However, the three meta-

[1]Some results are missing because Kaggle limits the number of submissions to 4 per team despite being originally informed that we are allowed 4 submissions per team member.



(a) K Nearest Neighbors Cross Validation: 95% Confidence Interval



(b) Predictions correlation matrix for default base models

learners (BLS, Ridge, and simple average) do not appear to produce significantly different results as their predictions are 99% correlated.

Moreover, as alluded to above, stacking contains a trade-off between the optimization of hyperparameters within each base model and avoiding data leakage at the meta-learner level. As seen in Figure (a), initial hyperparameter tuning suggests that it can reduce base-model over fitting, **but data leakage may disrupt this improvement**. Table 2 confirms this data leakage concern as the models with base model hyperparameter tuning perform worse than those without hyperparameter tuning.

Main Takeaway

When we do cross-validation, our choice of base models is affected by the training set out-of-fold performance implying that the whole training set is indirectly exposed. However, as our results suggest, the meta-learner overfits as a result of this data leakage. This is further supported by the fact that the best fine-tuned base model (random forest) outperforms the stacked Ridge and BLS models which are trained with random forest predictions as a feature. More evidence comes from the observation that for the fine-tuned base models, the "Avg Stacked" approach which does not suffer from data leakage outperforms the Ridge and BLS meta-learners and the best fine-tuned base model.

While this data leakage proves troublesome for this dataset, a dataset with sufficient size to optimize hyperparameters and meta features without overlap may take full advantage of the benefits of stacking.

References

Leo Breiman. 1996. Stacked regressions. *Machine learning*, 24(1):49–64.

Michael LeBlanc and Robert Tibshirani. 1996. Combining estimates in regression and classification. *Journal of the American Statistical Association*, 91(436):1641–1650.

David H Wolpert. 1992. Stacked generalization. *Neural networks*, 5(2):241–259.