Exploring the Performance Impact of Approximate Normalized Training Data

Chao-Ting, Wu
chaoting@uw.edu

Kuan-Yu, Lu
lukuanyu@uw.edu

Abstract

As previous works have shown that, normalization of training data enjoys both the efficiency of space and learning time while preserving the same performance under certain models (e.g. linear regression) and data conditions (Schleich et al., 2016). However, real-world data are often subject to noises, where exact normalization as used in the previous work couldn't be obtained. Though this problem can be resolved by adopting approximate normalization (Kenig et al., 2019), it's unclear whether the performance of the model will be still preserved, assuming the speed-up holds true for approximate normalized schemes. In this project, the performance of the learning result over approximate normalization, as a substitute for exact normalization, is examined to explore the benefits/ losses other than the speed-up of training linear regression models over normalized schemes in real-world data-sets.

1 Tools and Environment

- PostgreSQL for join computation and data storage.
- Python 3.7.4 for data import/export, model training, and visualizations.
- Maimon for discovering approximate acyclic schemes.
- AWS Linux instance with 32GB RAM.

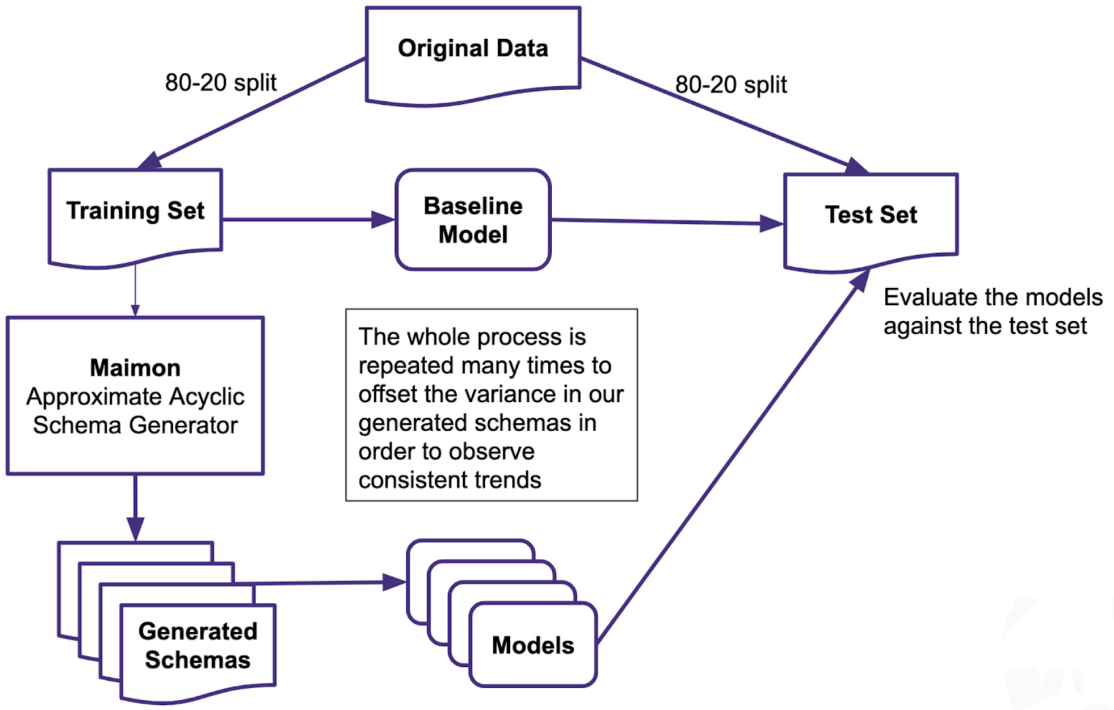
2 Datasets

- Red Wine Quality
  The two datasets are related to red and white variants of the Portuguese "Vinho Verde" wine. Linear regression could be applied to predict the overall quality a specific variant.

- King County House Price
  This dataset contains house sale prices for King County between May 2014 and May 2015. Originally, the dataset contained 21 columns, and we decided to drop ten of them since they are irrelevant and unrelated to reduce the computing power needed. We kept price, bedrooms, bathrooms, sqft living, sqft lot, floors, waterfront, condition, grade, yr built, and zip code. Linear regression could be applied to predict the overall house sales prices based on these feature variables.

3 Description of the method and experiments

Figure 1: Workflow of the experiment



- Original data will be split into a training set and a test set, where the training set will be used for:
  - Relational decomposition by using Maimon
  - Model exploration using a 5-fold cross-validation
- The test set will be used for:
  - Assessing performance of the models obtained

Empirical experiments and explorations were conducted to see whether there's a consistent or explainable impact over the generated acyclic schemes over data sets. The workflow is depicted in figure 1.

First, the training data is splitted into a training set and a test set under a 80-20 split. Baseline models were obtained from the training sets using 5-fold cross-validations. Second, relational decomposition will be applied on the training sets to discover approximate acyclic schemes. For each approximate scheme, we obtain a new model from the joined result of the generated schema. Lastly, the performance (e.g. MSE, R-squared) and the validity (e.g. redisual analysis) of all the obtained models were compared and analyzed.

4 Experiment Results

Detailed flow of the experiments is described as followed:

- Step1: Split the original datasets into 80-20 train-test splits under different random seeds.
- Step2: Generate acyclic schemas for each training set under epsilon = 0.1 by using Maimon application. In the red wine dataset, each training set generates around 30 to 32 acyclic schemas.

- Step3: Store the decomposed tables in a PostgreSQL database and compute the joined result using views.
- Step4: Obtain the linear regression model of each schema using Python.
- Step5: Evaluate the performance of linear regression models against the corresponding test set using statistics such as percentage of spurious tuples, mean square error (MSE), and R-squared.

5 Analysis - Red Wine Data Set

The statistics and the performance of the model derived from each model is collected and plotted in figure 2.

Looking at figure 2, there are two immediate takeaways here: (1) In general, more spurious tuples lead to lower performance (i.e. higher MSE) , and (2) more spurious tuples sometimes lead to lower R-squared value in the obtained models. To explain these 2 takeaways, we will focus on few selected schemas (detailed statistics of the selected schemas is recorded in table 1). Residual analysis and the distributions of the dependent variable are introduced on these schemas for an in-depth explanation.

Figure 2: Summary of the experiment results

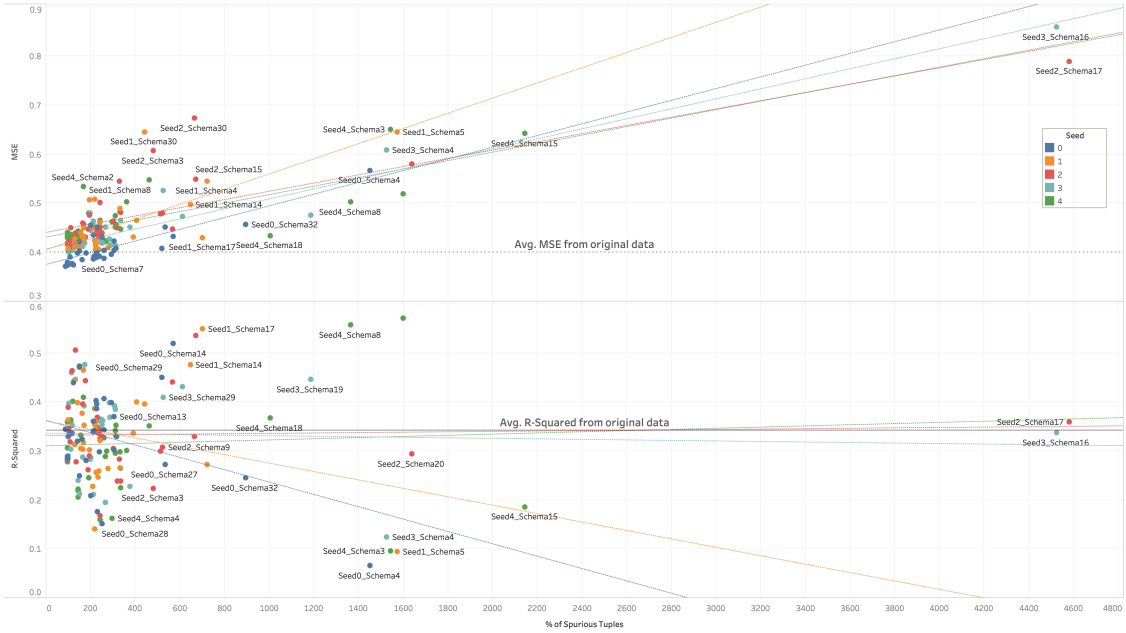


Table 1: Statistics of selected schemes

Scheme	# of tuples	% of spurious tuples	best MSE	R-Squared
Original	1261	NA	0.37	0.33
Scheme 4	18309	14.51%	0.56	0.06
Scheme 16	57045	45.23%	0.85	0.33
Scheme 17	57778	45.81%	0.78	0.36

In schema 16 and 17, where we see high MSEs (low-performance) but acceptable R-Squared values, the residuals of these two schemas are centered around zero as plotted in figure 3. Though in general the residuals of a good linear model should follow a normal distribution

centered around zero, which is similar to what we see in the original dataset, the residual distributions of schema 16 and 17 seem to follow this distribution with high kurtosis. However, as we observed in table 1, the MSEs of the models derived from these two schemas are twice as high as the original data. This is caused by the high errors on the both ends of the prediced values as we can see the models have less explainability for high or low predicted values. To further expalin why these models are biased, we then took a deeper look into the distributions of the dependent variable for the generated schemes.

Figure 3: Residual plots for the baseline model, schema 4, 16 and 17

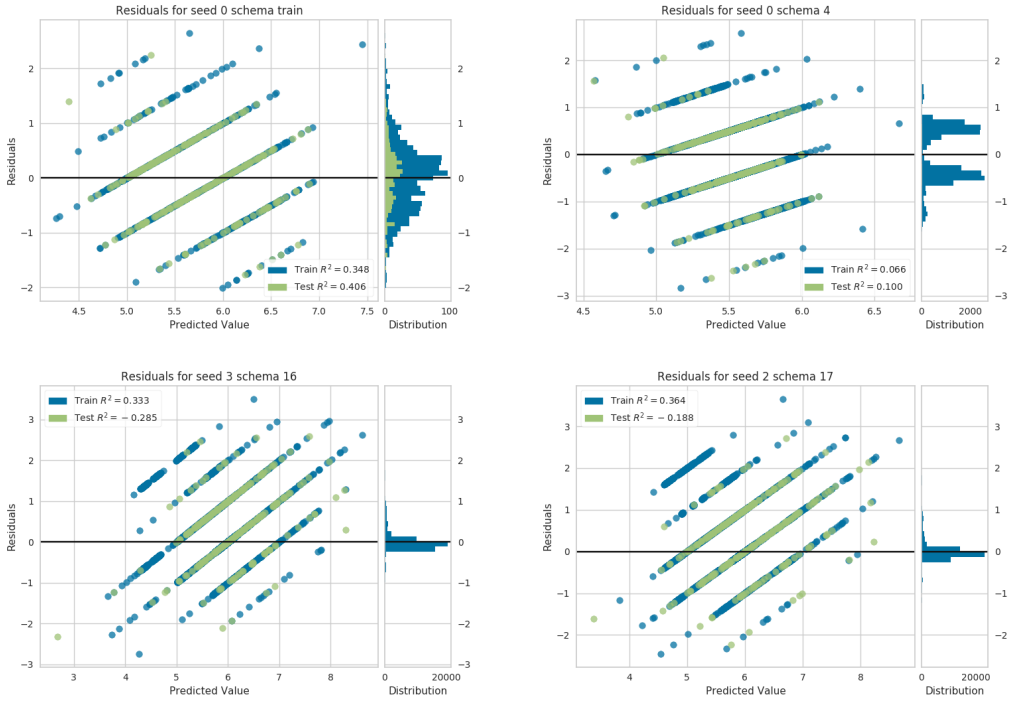
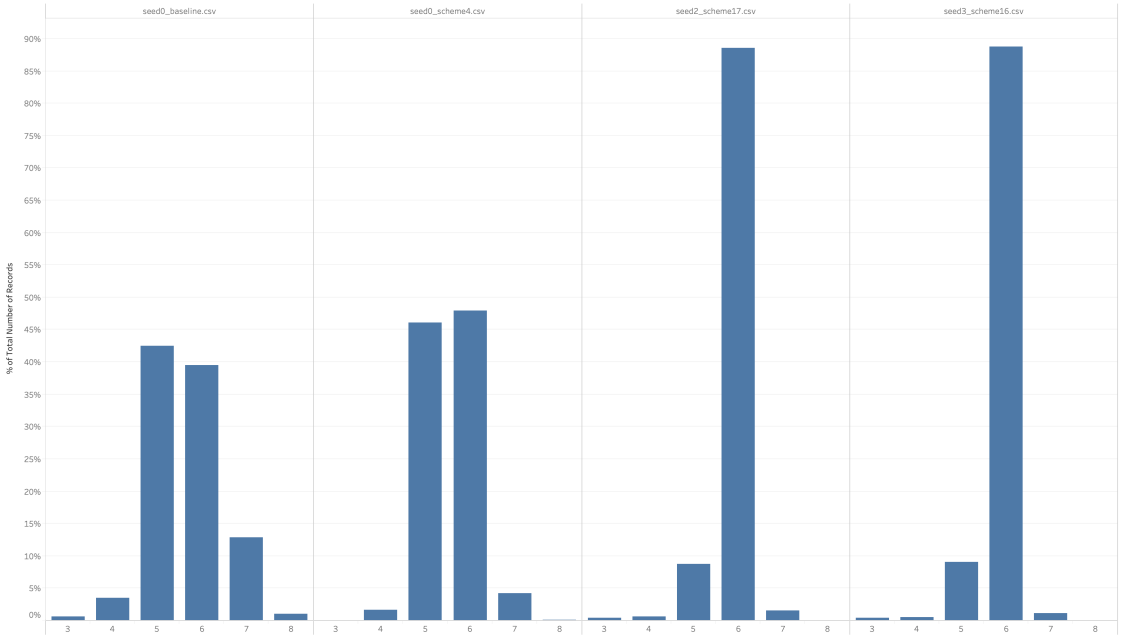


Figure 4 shows the distributions of the dependent variable of the generated schemas. As we can see, in these two generated schemas, huge amounts of spurious tuples with popular dependent variable in the original data (5 and 6) lead to distributions with high kurtosis. The models obtained from these two schemas thus have less explainability than the original model when predicted values are rare.

In schema 4, where we see an acceptable MSE but an extremely low R-Squared value, the story is different. As we can see in figure 3, the distribution of the residuals for the model derived from schema 4 is far from a normal distribution as we see in the original data. This suggests that the linear model we used on the original data failed to model the data generated by schema 4, and caused the low R-Squared value. However, the distribution of the dependent variable is somehow preserved, which made the performance of the model seem to be acceptable.

Figure 4: Histogram of the dependent variable for the original data, schema 4, 16, and 17 (left to right)



6 Analysis - King County House Price Data Set

We conducted the same experiment steps on the King County House Price Data Set, however the result is slightly off from what we observed in in red wine data set. Here, although in the summary graph (figure 5) we can observe similar trends as we did in the red wine data set, for the residual and dependent variable analysis (figure 6 and 7), the difference of them is not as significant as we observed in the red wine data set. Here, we picked schema 94 and 34, a high error and a low R-Squared case respectively.

Figure 5: Summary of the experiment results

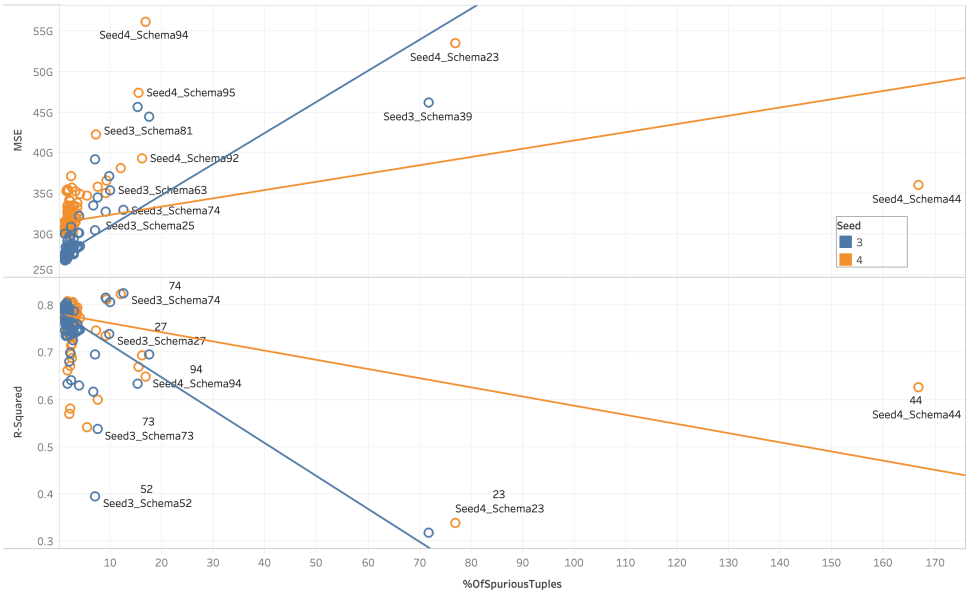


Figure 6: Residual plots for the baseline model, schema 39 and 94

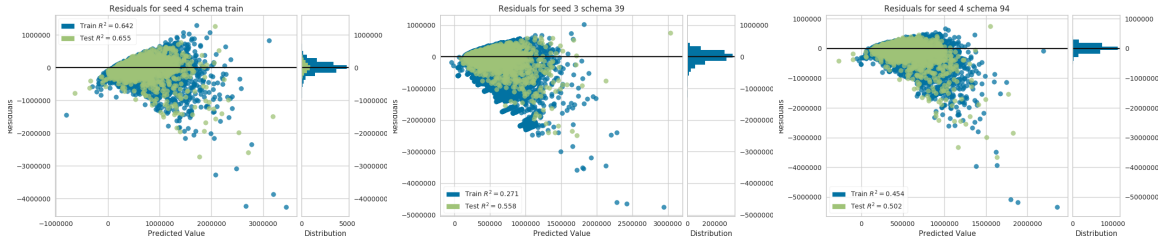
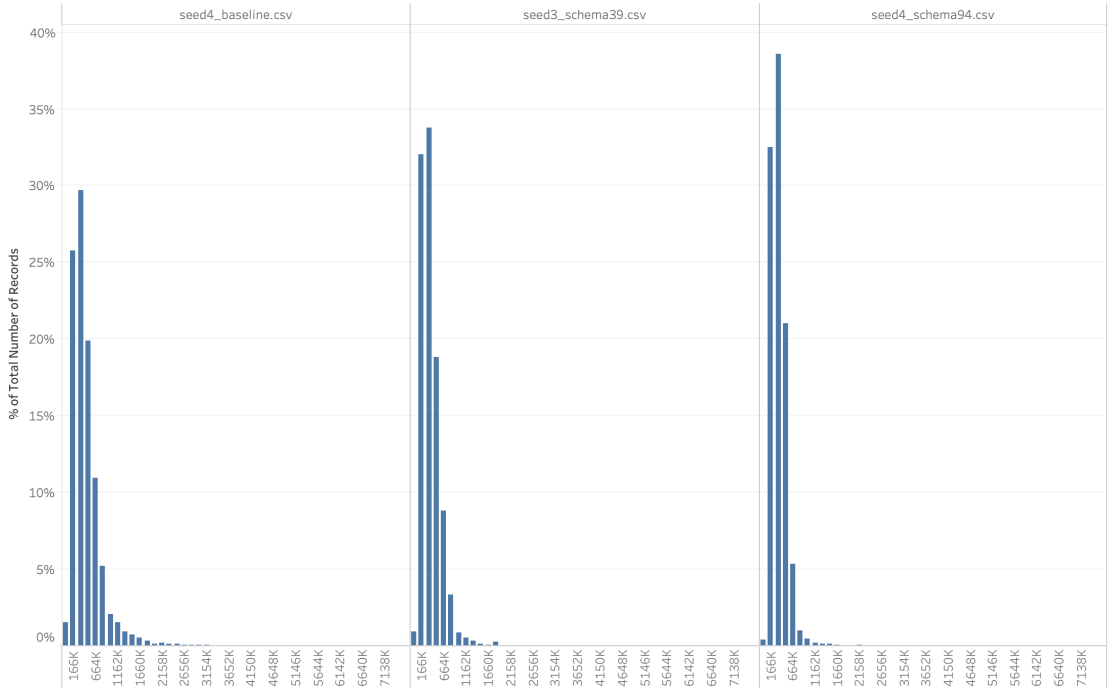


Figure 7: Histogram of the dependent variable for the original data, schema 39 and 94 (left to right)



However, as we can see in figure 6, the residuals distribution of the original model was not following the assumption that residuals should be randomly and normally distributed. Here, we can see that there's a trend in the distribution of the residuals, which implies there might be a second order relationship between the dependent variable and our input variables. We believe that once we identify the right model (formula) rather than the naive first order full model that we use now, we should be able to see some similar findings as we did in the red wine data set.

7 Conclusions

From the red wine data set experiemtns. It seems if we choose to do relational decomposition on our training data in order to leverage the efficiency it provides, either to enjoy the speedup or the saving of space, then we need to be careful on 2 potential losses.

The first loss is the loss of accuracy, where usually we'll see high error terms but an okay R-squared. This is caused by the change of the distribution of the dependent variable, which makes the model have less explainability when the predicted value is rare, like what we see in schema 16 and 17.

The second loss is the loss of linearity, where usually we'll see an acceptable error term but a low R-squared value. This could be dangerous if we're using linear regression to do statistical inferences, since the underlying assumptions of linear regression is not met, like what we see in schema 4.

8 Future Works

As we see in the King County House Price Data Set, our analysis failed since we're based on baseline models that don't follows our naive linear regression assumptions and formulas. Given the time and resources constraints that we have for this project, we didn't explore the best linear model for the King County House Price Data Set, and we think it would be interesting to get the right formula first before applying it to generated datasets. In that way, we can see if we can generalize our findings from the red wine data set from some better reference data points.

Also, we encountered a few issues while using Maimon to generate MVDs and acyclic schemas, and made some modifications. Our Java version is 11, and we ran the application on AWS with 2 cores and 8G memory.

- The system doesn't provide any log to trace the code, we recommend using log4j.
- The in-memory database function is memory-consuming. If the dataset is too large, 20000 tuples with 13-21 columns for example, the system will run out of memory. After testing multiple datasets, we found that less than 12 columns might be a good fit.
- Few functions weren't optimized. Since the application is column-sensitive, loops will take exponential time given by the number of columns. We removed some redundant loops.
- The queries could be optimized. For example, the value of entropy was originally calculated in Java by summing values from the result of "SELECT CNT FROM table". We replace the original query with aggregations to leverage optimizations in the database.
- We recommend using multi-threads instead of single-thread.

References

Batya Kenig, Pranay Mundra, Guna Prasad, Babak Salimi, and Dan Suciu. 2019. Mining approximate acyclic schemes from relations.

Maximilian Schleich, Dan Olteanu, and Radu Ciucanu. 2016. Learning linear regression models over factorized joins. Proceedings of the 2016 International Conference on Management of Data - SIGMOD 16.

George Trigeorgis, Konstantinos Bousmalis, Konstantinos Bousmalis, Stefanos Zafeiriou, Stefanos Zafeiriou, Björn W. Schuller, Department of Computing, Department of Computing, Department of Computing, Department of Computing, and et al. 2014. A deep semi-nmf model for learning hidden representations. A deep semi-NMF model for learning hidden representations | Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32, Jun.

Yongchang Wang and Ligu Zhu. 2017. Research and implementation of svd in machine learning. Research and implementation of SVD in machine learning - IEEE Conference Publication.