Practice of Comparisons of Supervised Learning Algorithm
Replica of the 2006 Paper by Caruana and Niculescu-Mizil at Cornell University

Jeffrey (Jiahe) Feng j1feng@ucsd.edu

The University of California San Diego, La Jolla, CA, 92093 USA

Abstract

Due to the lack of comprehensive comparisons and evaluations of different supervised machine learning algorithms since the early 90s, Rich Caruana, and Alexandru Niculescu-Mizil from Cornell University published a paper in 2006 comparing ten major algorithms on 11 different classification task with both balanced and unbalanced binary classification labels (Caruana & Niculescu-Mizil, 2006). They reported their result as a rank of performance of different algorithms. Following the methodologies from that CNM06 paper, we aimed to pick a few of the datasets that the study used, replicate the experiment, and evaluate the performance of some of the supervised ML algorithms across different trials.

1). Introduction

To summarize the 2006 paper by Caruana Niculescu-Mizilf furthermore: STATLOG, according to the authors, was one of the best known study (King et al., 1995), but the study was outdated considering that there were new emerging supervised algorithms such as Support Vector Machine and Random Forest Classifier, and that led to the motive of the study: to further explore the performance of new algorithms on a variety of tasks. The ten supervised algorithms used by the original paper were: SVMs, neural nets, logistic regression, naive bayes, memory-based learning, random forests, decision trees, bagged trees, boosted trees, and boosted stumps.

Due to the difference in nature of the datasets and the algorithms, the paper considered many different error metrics as well: accuracy, F-score, Lift, ROC Area, average precision, precision/recall break-even point, squared error, and cross-entropy. The general performance of an algorithm would be evaluated based on all of these factors. For each algorithm, there are many different hyper-parameters, and to ensure that each algorithm is performing on an unseen test set at its best, the authors tried many different hyper-parameters for each algorithm. One example is that for the support vector machine, the study used linear, polynomial degree 2 & 3 kernels, radial with width {0.001,0.005,0.01,0.05,0.1,0.5,1,2}, and regularization parameter of {10^-7, 10^-6, 10^-5, 10^-4, 10^-3, 10^-2, 10^-1, 1, 10, 100, 1000}. The study used calibrations such as Platt's Method or Isotonic Regression, which yields multiple results for each algorithm, depending on how it was calibrated. This calibration method would not be covered in this replica study.

Following this study, we picked datasets from UCI Machine Learning Repository that were used in the report, and for each dataset, we performed basic data cleaning, one hot encoding, and prepared training and testing data with the appropriate size (although training/test split is different in each trial). We would use supervised machine learning techniques with accuracy as the default scoring measurement for binary classification to measure the performance of each algorithm, averaged across different trials. Because we are only using one error metric, there is no need to normalize or calibrate the result of each algorithm on the testing set. We would also use 2 sample T tests to compare the results, once we have all the accuracy performance on the testing sets collected.

2). Methodology

Three datasets: ADULT, COV TYPE and LETTER from the UCI Repository (Blake & Merz, 1998) were selected for our replica study. We performed similar data preprocessing as the CNM06 paper suggested. For the adult dataset, there were originally 14 columns besides the prediction label, and 8 of them are categorical columns. After one-hot-encoding these non-numeric columns to multiple columns of 0/1 binary features, we attained a dataframe with 103 columns ready for modelling. For the predictive

label, we assign 1 to " >50K" and 0 to " <=50K", which yields 22654 negative class labels and 7508 positive class labels, indicating that this predictive task is not balanced. For the COV_TYPE dataset, there are already 54 feature columns, and we classified the predictive label to 0 and 1 based on whether it is equal to 2. We then have 297711 negative labels and 283301 positive labels. Lastly, for the letter recognition dataset, there are 16 feature columns, and we assigned 1 to letter A to M, and 0 to other letters, producing 10060 negative labels and 9940 positive labels, making it a balanced binary classification task.

Our study originally selected 3 machine learning algorithms: Logistic Regression, Artificial neural network, and Random Forests, considering how different their implementations are. Although the CNM06 paper pointed out the No Free Lunch Theorem and suggested that there is no "best" learning algorithm, it does show that calibrated boosted tree classifiers were the best learning algorithm on the datasets it used. Therefore, we added boosted trees as the fourth learning algorithm that would be tested in our analysis using the *GradientBoostingClassifier* from sklearn. The settings of grid-search for each algorithm comes from the CNM05 paper:

- Logistic regression: regularization parameter ranged from 10^-8 to 10^4 and no regularization;
    - 14 hyperparameter settings;
- ANN (gradient descent): number of hidden units: {1, 2, 4, 8, 32, 128}, momentum {0, 0.2, 0.5, 0.9};
    - 24 hyperparameter settings;
- Random Forest: with 1024 trees, size of feature set is {1, 2, 4, 6, 8, 12, 16, 20} (or {1, 2, 4, 6, 8, 10, 12, 16} for the letter recognition datasets, because there is no more than 16 columns);
    - 8 hyperparameter settings;
- Boosted Tree (Gradient): steps of boosting in {2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048}
    - 11 hyperparameter settings;

We will create these models using methods imported from the scikit-learn library in Jupyter Notebook in Python 3.

3a). Experiment Setup

Like the CNM06 paper, we will create 5 trials for each algorithm on each dataset. For each trial, to measure its best possible performance on unseen data, we randomly selected 5000 data samples (training data) from the dataset for a 5 fold cross-validation, with a gridsearch to search for the best hyperparameters. We then selected 25162 rows of testing data for ADULT, 25000 for COV_TYPE, and 15000 for letter recognition dataset. For logistic regression and ANN, we transformed the training data into processed data with mean = 0, std = 0, using sklearn StandardScaler() method, which was also suggested by the CNM06 paper.

With 5 trials, and 3 datasets, and 5-fold cross validation, we had 15 grid-search for each algorithm with a total of 75 folds. Because we created 14+24+8+11 = 57 hyperparameter settings for all 4 algorithms, the analysis will have 57 * 75 = 4275 train/valid cycles. At each trial, after each grid-search, we selected the best hyperparameter settings and trained all 5000 data samples, and tested the model performance on the test data.

Does a higher mean performance of each algorithm across many trials of an algorithm over the other mean it performs better? We will use uncorrected 2 sample t-tests to compare across algorithms to show statistically significance. These comparisons will be presented in the next section, after the primary report of performance of each algorithm. Just like the CNM06 paper, the "winning" algorithm with the highest performance would be marked bolded, and we put an "*" to algorithms with statistically indistinguishable performance with the best algorithm, using P value 0.05.

We also use Python's time() method to record the exact time it takes for each grid search to run. We consistently used n_job = 10 in GridSearchCV() method to allow multiple processors to run, and tried to use scipy.sparse.lil_matrix() to structure the training data to make the algorithm run faster, if it could. In short, for each gridsearch/trial, we record the time in seconds for the codes to run after the attempts to optimize the time. The codes were all run in UC San Diego datahub with 1 GPU, 8 CPU, 16G RAM.

3b). Reporting Model Performance

Primary result:

Table 1. Mean test set performance for each algorithm/dataset combo across 5 trials

Mean Accuracy Over 5 trials [Test Mean]	ADULT	COV_TYPE	LETTER
LOGREG	0.8437	0.7265	0.7216
ANN	0.8218	0.7989	**0.9476**
RF	0.8418	**0.8244**	0.9460*
BST-DT	**0.8600**	0.7969	0.9369

Table 2. Mean test set performance for each algorithm

Model	Mean Accuracy of 15 test scores
LOGREG	0.7639
ANN	0.8561*
RF	**0.8707**
BST-DT	0.8646*

Secondary results:

Table 3. Mean training set performance for the optimal hyperparameters on each of the dataset/algorithm

Training Accuracy Average ver 5 trials	ADULT	COV_TYPE	LETTER
LOGREG	0.8545	0.7258	0.7289
ANN	0.8373	0.8728	1.0000
RF	0.9910	1.0000	1.0000
BST-DT	0.8896	0.9513	0.9994

Table 4. Raw test scores for the testing set performance for making the primary results and the P values result (table 1, 2, 6, 7)

Test Accuracy Of 5 Trials	ADULT	COV_TYPE	LETTER
LOGREG	0.8448, 0.8437, 0.8443, 0.8436, 0.8423	0.7584, 0.7496, 0.657, 0.7273, 0.7404	0.7224, 0.7202, 0.7241, 0.7149, 0.7262
ANN	0.8425, 0.8333, 0.8225, 0.7976, 0.8132	0.7945, 0.8015, 0.7974, 0.8062, 0.7949	0.9449, 0.9443, 0.9525, 0.9518, 0.9445
RF	0.8452, 0.8405, 0.8414, 0.8395, 0.8426	0.8236, 0.8234, 0.8228, 0.8268, 0.8255	0.9493, 0.9442, 0.9485, 0.9453, 0.9430
BST-DT	0.8618, 0.8594, 0.8608, 0.8583, 0.8597	0.7964, 0.7963, 0.7963, 0.8031, 0.7924	0.9347, 0.9370, 0.9398, 0.9367, 0.9362

Table 5. Raw training scores for the testing set performance for making Table 3.

Training Accuracy Of 5 trials	ADULT	COV_TYPE	LETTER
LOGREG	0.8496, 0.8548, 0.8510, 0.8556, 0.8616	0.7564, 0.7498, 0.658, 0.7268, 0.7380	0.7232, 0.7304, 0.7244, 0.7342, 0.7320
ANN	0.8562, 0.8616, 0.8292, 0.8146, 0.8252	0.8696, 0.8790, 0.8728, 0.8740, 0.8686	1.0000, 1.0000, 1.0000, 1.0000, 1.0000
RF	0.9908, 0.9904, 0.9920, 0.9914, 0.9902	1.0000, 1.0000, 1.0000, 1.0000, 1.0000	1.0000, 1.0000, 1.0000, 1.0000, 1.0000
BST-DT	0.8896, 0.8840, 0.8908, 0.886, 0.8978	0.9538, 0.9570, 0.8978, 0.9540, 0.9940	0.9994, 0.9996, 0.9990, 0.9990, 0.9998

Table 6. P values comparison across algorithm of table 1 (P value greater than 0.05 is marked bolded, indicating no difference in the samples or statistical insignificance)

T-test P values	ADULT	COV_TYPE	LETTER
LOGREG vs. ANN	0.0231	0.00417	4.439 * 10^-13
LOGREG vs. RF	**0.1134**	0.00065	1.130 * 10^-13

LOGREG vs. BST-DT	1.806 * 10^-8	0.00479	9.344 * 10^-14
ANN vs. RF	0.0345	4.323 * 10^-6	**0.509**
ANN vs. BST-DT	0.0012	**0.4961**	0.000769
RF vs. BST-DT	2.624 * 10^-7	4.691 * 10^-7	0.000255

Table 7. P values comparison across algorithm of table 2 (P value greater than 0.05 is marked bolded, indicating no difference in the samples or statistical insignificance)

Algorithms	T-test P values
LOGREG vs. ANN	0.000617
LOGREG vs. RF	3.1777 * 10^-5
LOGREG vs. BST-DT	0.00010
ANN vs. RF	**0.5246**
ANN vs. BST-DT	**0.7192**
RF vs. BST-DT	**0.7706**

3c). Comparison

We generated table 3 from table 5, which contains all the training scores once if we find the optimal hyperparameter for each trial. For logistic regression models, the training accuracies and testing accuracies do not differ greatly, although all the training accuracies are higher than testing accuracies. The Artificial Neural Network performed a lot better at the training set, attaining 100% accuracy on the letter recognition dataset, but performed really poorly on the ADULT dataset by just slightly beating the testing accuracy mean by around 1%. Random Forest attained almost 100% for all training accuracies, much higher than the testing performance, indicating that it was fitted very well on the training data. Boosted tree also attained higher accuracy over training data, although instead of getting 100%, it got 89% on the ADULT dataset, and 95% on the COV_TYPE dataset. Both RF and BST-DF models achieve much significantly higher accuracy on the training dataset than testing.

From the results of 2 sample T-test on table 1 and table 2, we can see that LOG and RF performed similarly on ADULT dataset, ANN and BST-DT performed similarly on COV_TYPE dataset, and ANN and RF performed similarly on the letter recognition dataset. Interestingly, the logistic regression performed much significantly worse than other algorithms on the letter recognition dataset. This results in RF in table 1, and ANN and BST-DT in table 2 all got asteroids (*), indicating that they do not perform significantly differently than the best algorithm.

3d). Time Analysis

Since we performed over 4000 train test cycles for this study, it is important to optimize the time for each grid-search. Interesting finding from the experiment is that if we put the training data into scipy.sparse.lil_matrix() other than a numpy nested array or a pandas dataframe, the runtime became significantly faster for LOGREG and ANN, and for these two algorithm, standardize the data into 0 mean and 1 standard deviation also significantly improve the runtime. To put things in perspective, with n_jobs default to 1 and we only have 1 hyperparameter setting, and 2 cross validation, for logistic regression algorithm on the ADULT dataset:

- Numpy array format, no standardscaler: 127.33 seconds
- Sparse matrix format, no standardscaler: 4.823 seconds
- Sparse matrix format, standardscaler: 2.336 seconds

However, the sparse matrix method does not improve the runtime for RF and BST-DT, where we do not use a standardscaler for the training data. When we have 2 cross validation, 1 hyperparameter setting, and n_job = 1, random forest classifier takes the following time to run:

- Sparse matrix format: 52.685 seconds
- Numpy array format (No sparse matrix): 14.918 seconds

Therefore, the sparse matrix method is absolutely not a universal way to increase model runtime. Overall, we get the following result for the runtime. Table 8 shows the unweighted runtime of these algorithms, however, due to the fact that these algorithms differ in the number of hyperparameter settings

in the grid-search, we can generate Table 9 by taking their mean runtime across 5 trials and 3 datasets, and divide by the number of hyperparameter settings they each have.

Table 8. Model 5-fold GridSearch Time of Algorithms on the 3 Datasets, Unweighted.

Time [seconds] (# of hyperPM)	ADULT	COV_TYPE	LETTER
LOGREG (14)	2.34, 1.74, 1.59, 1.77, 1.67	2.61, 0.85, 1.06, 0.86, 2.16	2.15, 1.39, 0.42, 0.38, 0.38
ANN (24)	49.58, 51.99, 51.64, 50.83, 51.13	516.77, 503.72, 499.69, 526.43, 545.86	272.75, 357.98, 216.52, 216.80, 222.94
RF (8)	39.27, 39.35, 40.55, 39.37, 38.81	72.40, 63.63, 64.04, 71.23, 67.97	51.23, 55.30, 52.25, 51.22, 48.06
BST-DT (11)	27.04, 25.44, 26.28, 25.71, 26.72	54.36, 50.54, 45.17, 54.61, 65.29	33.24, 33.72, 34.43, 33.38, 33.02

Table 9. Model 5-fold GridSearch Time Weighted by Number of Hyperparameter Settings

Algorithm	Time [seconds]
LOGREG (14)	0.101
ANN (24)	11.485
RF (8)	6.622
BST-DT (11)	4.751

Logistic regression is the fastest algorithm, followed by BST-DT, RF, and ANN is the slowest algorithm in this study. An important note is that this is a purely informative comparison of how these algorithms have run on the jupyter notebook in our study, not indications of the exact time each algorithm

will spend on predictive tasks in general. Even though the result is weighed by the number of hyperparameters for each algorithm, some hyperparameter settings will, in their nature, take longer to run. An example of this is the number of hidden nodes for ANN. When it increases to 100+, we usually have to change max_iter if we want the algorithm to fully converge and get an ideal result.

4). Conclusion

The replica study did not get the exact results like the CNM06 paper, because the randomness in selecting the training data, and the sklearn library I am using might differ from the techniques used by the CNM06 researchers back in 2006. Random Forest and Boosted Tree, unsurprisingly, performed well in all the datasets, with RF ranked #1 and Boosted Tree performed similarly after we tested with 2-sample t-test. ANN, however, performed better in our study and also achieved high accuracy (statistically similar with the winning algorithm). However, with the error metrics being only accuracy, the limited number of datasets to try, we cannot for sure conclude which algorithm would perform the best. This replica study only aims to compare the different performance we got from sample data, not to deny the Free Lunch Theorem stating that there is no universally best algorithm. As the CNM06 paper also addressed, some models that consistently perform worse than others might perform extremely well for some datasets.

5). Acknowledgement

Professor Fleischer, Jason G @ UCSD Cognitive Science Department and COGS 118A teaching assistants that guide me through this project;

Rich, C., & Alexandru, N. For the original study/report;

UCI Machine Learning Repository for providing the datasets:

- https://archive.ics.uci.edu/ml/datasets/Adult
- https://archive.ics.uci.edu/ml/datasets/Covertype
- https://archive.ics.uci.edu/ml/datasets/Letter+Recognition

6). Reference

Rich, C., & Alexandru, N. (2006). An Empirical Comparison of Supervised Learning Algorithms, *ICML '06: Proceedings of the 23rd international conference on Machine learning*, https://doi.org/10.1145/1143844.1143865

Blake, C., & Merz, C. (1998). UCI repository of machine learning databases.