MIT 6.867 Machine Learning
Final Project Milestone 4: Initial Results
Improvement of neural network performance with feature engineering

Irina Degtiar, Greyson Liu, Aaron Sonabend

Latex style: icml2016.sty
https://www.overleaf.com/gallery/tagged/conference-paper#.WgNbKcanHid

I. Introduction

Feature engineering transforms the original inputs onto a new feature space that hopefully improves data separability. Non-linear classifiers such as neural networks allow for features to be adaptively created by transforming the original input data. However, proper feature engineering has the potential to further improve algorithm performance. In many machine learning settings, a-priori knowledge of the relationships between features is limited. Exploratory data analysis, however, can uncover covariate relationships that promote target separation. In this report, we explore the impact of feature engineering on neural network prediction and speed in the setting of no a-priori knowledge of feature relationships. We explore whether performance can be improved in various simulations and in an auto insurance claims database.

II. Related work

Feature engineering has been shown to win prediction competitions,[1] but prior work has focused on settings in which domain knowledge can be utilized to create meaningful features. We will explore the setting in which prior knowledge is limited.

III. Study setup

A. Auto insurance claims data description

In the Kaggle competition Porto Seguro's Safe Driver Prediction, Porto Seguro, a Brazilian insurance company, seeks to improve predictions of a driver filing an auto insurance claim (kaggle.com/c/porto-seguro-safe-driver-prediction). Porto Seguro provides a dataset of 595,212 training and 892,816 test individuals with 59 de-identified features; 3.6% of which filed claims. 79% of individuals are missing at least one feature, with one of the features missing for 69% of individuals and all but 5 features having <1% missingness. No one feature is strongly correlated with the target, but there is up to 89% correlation between features (Figure 1).

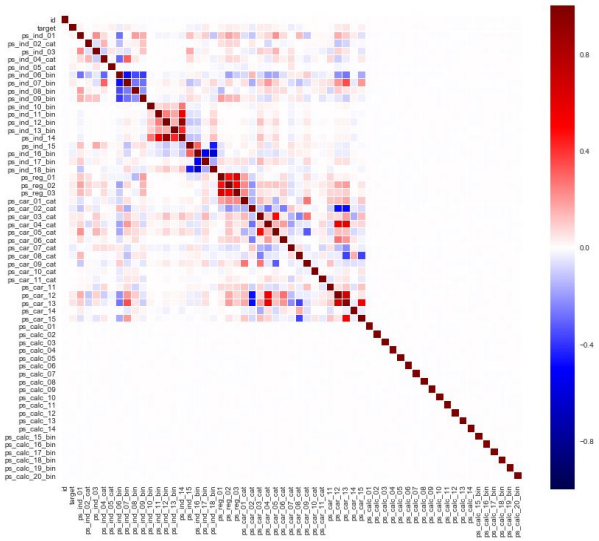

Figure 1. Pearson correlation between features

Both correlated and uncorrelated variables show meaningful interactions that induce target separability (Figure 2), and thus might potentially improve algorithm performance.

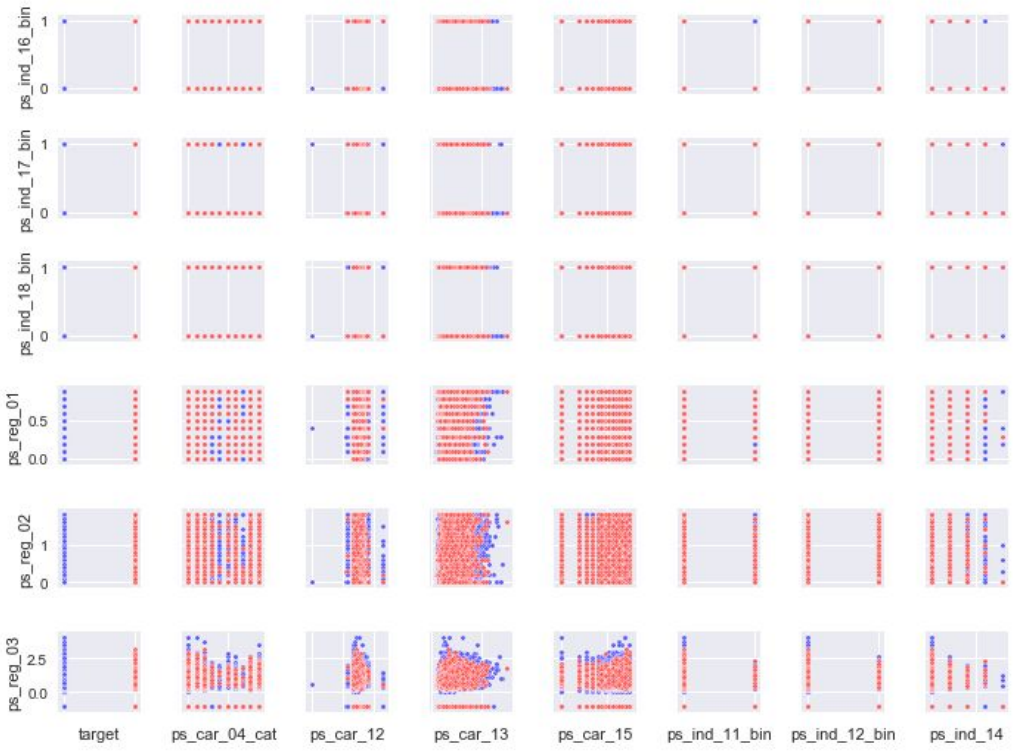

Figure 2. Select pairwise feature distributions. Red: target = 1; blue: target = 0. Note separability induced by ps_ind_16_bin*ps_ind_11_bin interaction and ps_reg_03*ps_car_13 interaction.

B. Auto insurance claims feature engineering

Preliminary feature engineering consisted of binary interaction terms between features that induced data separability visually. We will explore additional possibilities in the future.

C. Data cleaning and missing data imputation

We added indicators for missingness for each variable. Missing data was imputed using 5-nearest-neighbors using the "bnstruct" package in R. For computational efficiency, we randomly shuffled the rows of our data and split the training dataset into 10 pieces. Ten 5-nearest-neighbors algorithm were applied separately within each piece to improve computation time. Each row in the testset was assigned to one of the training set splits and imputated using 5 nearest training neighbors. To standardize data, categorical variables were converted to one-hot encoding; binary variables to +/-1 encoding, which has been shown to be optimal for neural networks;[2] continuous variables were Gaussian normalized (subtracting means, dividing by standard deviations).

D. Simulation

100,000 iid samples of a binary outcome were generated by applying the sigmoid function to a linear sum of covariate functions defined as follows:

Let $X = (X_1, X_2, ..., X_7)$, then omitting the i subscript to simplify notation:

$P(Y = 1) = \sigma(\sum_k f_k(X) + \varepsilon)$ with $f_k(X) = X_k$ for $k = 1, ..., 5, 7$, $f_6(X) = \sigma(X_6)$, $f_8(X) = X_2 X_3$, $f_9(X) = X_1 X_3$, $f_{10}(X) = sin(X_4 X_6)$, $f_{11}(X) = X_5 X_2$, $f_{12}(X) = (X_2)^3$ where

$X_1 \sim Bern(.8)$, $X_2 \sim N(2, 9)$, $X_3 \sim N(-1, 4)$,

$X_4 \sim multinom(.0625, .125, .167, .125, .25, .083, .125, 0.0625)$,

$X_5 \sim multinom(0.19, 0.387, 0.129, 0.194, 0.0968)$,

$X_6 \sim Bern(.3)$, $X_7 \sim Bern(.55)$, $\varepsilon \sim N(0, 4)$.

This is a highly non-linear function with several categorical and binary covariates as well as several interactions between both continuous and discrete variables which attempt to simulate the behavior observed in Figure 2.

E. Neural network model & tuning

With the simulation data, we applied neural networks to study (1) whether adding interactions as additional features can speed up the training procedure or (2) improve prediction accuracy.

We adopted a neural network with three hidden layers, where each layer is activated by a ReLU function and the number of units within each layer is 100, 50, 25, respectively. Those parameters are arbitrarily chosen and we will tune them for the final report. The output layer is linked by a sigmoid function and the loss function is cross entropy (accuracy is also reported). The algorithm was implemented by the keras package in Python, using the default mini-batch gradient descent method (rmsprop) with batch size 128.

F. Simulation study results

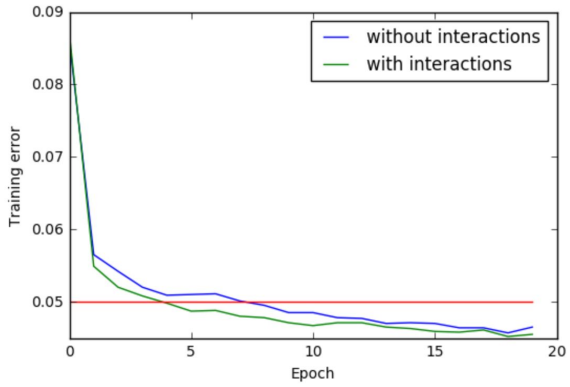

Figure 3. Training error comparing neural networks with and without interactions as inputs

From Figure 3, we can see that after 20 epochs, the network with interactions as input achieves a lower error rate than the one without interactions. What's more, in order to the achieve the same error rate, say 5%, the network with interaction only needs 4 epochs but the one without interactions needs 8 epochs. Those outcomes suggest adding interactions as additional features may speed up the training procedure and improve the prediction accuracy. However, more comprehensive simulation studies under various settings should be conducted to reach a final conclusion.

G. Auto insurance claims results

For the final report, we will explore the impact of feature engineering in the auto insurance claims database.

IV. Conclusions

With the initial simulation result, we have found that only using raw covariates vs. using the true interactions as features improves both speed of convergence measured in epochs and accuracy. However, the effect of including true interactions is small, and we do not have a sense of the accuracy distribution in order to determine if the difference is significant - we plan to explore significance for the final report. Clearly we do not have knowledge of true interactions between features for the real data; thus, we will explore incorporating a heuristic to automatically determine whether interaction features should be fed into the network.

Author contributions:

- ID: data exploration, feature extraction, & standardization, report drafting
- GL: missing data imputation, neural net & resnet training, report results drafting
- AS: simulated data generation, neural net & resnet training, report results drafting

Risks:

- Not seeing improvement in results in the auto insurance claim data. Solution: explore in simulations under what settings feature engineering can vs. can't improve results & how strong interaction needs to be to impact performance
- A large amount of research has been published on feature engineering, which we won't be able to absorb comprehensively. We'll likely not be using optimal methods. We will attempt to mitigate this by performing a literature

search and adopting the best methods we find

Citations:

- Project advice and feedback: Curtis Northcutt
- Data exploration: https://www.kaggle.com/arthurtok/introduction-to-ensembling-stacking-in-python https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python https://github.com/mwaskom/seaborn
- Data type extraction: https://www.kaggle.com/felipeleiteantunes/introductory-eda-baseline-with-cv

References

1. Yu, H. F. *et al.* Feature engineering and classifier ensemble for KDD cup 2010. *ntur.lib.ntu.edu.tw* (2010).
2. McCaffrey, J. How To Standardize Data for Neural Networks -- Visual Studio Magazine. *Visual Studio Magazine* Available at: https://visualstudiomagazine.com/articles/2014/01/01/how-to-standardize-data-for-neural-networks.aspx. (Accessed: 8th November 2017)