BIS 557 Final Project Write-up

A Neural Network Model for Life Expectancy Prediction and An Exploratory Comparison for Model Tuning Methodologies

Zizhong Tian (zizhong.tian@yale.edu)

Instructor: Professor Michael Kane

BIS 557 Computational Statistics
Yale School of Public Health
December 15th, 2020

Introduction

Life expectancy is affected by a complex set of factors. Forecasting life expectancy and mortality based on various sources of data forms an important subdivision of Demography and it is always a popular topic in the interdisciplinary studies of Statistics and Public Health [1]. This project mainly proposes a well-trained densely connected neural network model for life expectancy prediction based on a public data set. Multiple types of regularization methods including the weight penalty, dropout, and intermediate-layer weight constraints were implemented to refine the network, and their respective effects were compared controlling for a fixed multilayer perceptron framework with tuned parameters. I also tried the composite of some well-performed methods and discussed their behaviors, implications, and the ideal choice of approaches under our data setting. The choices of loss functions for regression-oriented neural networks were also considered and compared. Finally, I attempted to provide some reflections about the model tuning methods for neural network as well as the connections between regressions and deep learning methods in practice.

I focus on a data set collected by Global Health Observatory under World Health Organization about the national average life expectancy and some demographic or health-related factors for a vast range of countries from year 2000 to 2015. The cleaned data for analysis has 22 columns and 2928 rows, which includes the target variable for prediction, *Life Expectancy* (continuous variable), and 21 predicting variables (2 categorical variable and 19 continuous variable). A descriptive summary table about the cleaned data set could be found in ***Appendix***. This data is a good material for training a predictive model of country-average lifespans based on some novel aspects beyond the traditional considerations, like immunization-related and healthcare-related economic factors.

Method

To better fit the neural network study setting, several steps of data preprocess was firstly conducted. Dummy coding was done for all the categorical variables and a 2928x203 data matrix was obtained for input. Based on a one-time random division, 2228 observations were drawn for the training data set and there were 700 observations in the testing data set. For these kinds of predicting data with heterogenous scales, I conducted the feature-wise normalization (based on the means and standard deviations of the training data) for the input to help the network learn more efficiently. Note that there was only a little missingness in the total data set, so I simply imputed zeros for the missing points after the normalization under each variable, which was believed to cause minor bias and was better than the complete case analysis.

Next, I built up a dense neural network model, which includes the input and output layers and two hidden layers. Rectified linear units were applied on each hidden layer and the neuro number in each hidden layer was tuned in a factorial setting. Based on some tests, I decided to add 256 nodes and 64 nodes respectively in the two hidden layers. Since the response is continuous, no activation function was set for the output layer and the network ended with a single unit. For the model training, mini-batch gradient descent with batch size equal to 64 was coded for updating the weights. For the optimization algorithm, I tried the Adaptive Moment Estimation (Adam), which was a bias-corrected updating method incorporating the advantages of momentum and RMSprop [2]. An algorithm sketch for Adam was shown below:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$ **(Update the first moment estimate)**

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$ **(Update the second moment estimate)**

$$\widehat{m_t} = m_t/(1-\beta_1)$$ **(Compute bias-corrected first moment estimate)**

$$\hat{v}_t = v_t/(1-\beta_2)$$ **(Compute bias-corrected second moment estimate)**

$$\mathrm{w}_t = \mathrm{w}_{t-1} - \alpha\widehat{m_t}/\left(\sqrt{\widehat{v_t}} + \epsilon\right)$$ **(Update the weight parameter vector)**

For the loss function, I chose two formulas appropriate for the regression mission, the Mean Squared Error Loss (MSE) and the Log-Cosh Loss. The second loss function was specified below:

Log-Cosh Loss: $l(\hat{y_i}, y_i) = \sum_{i=1}^{n} log\left(cosh(\hat{y_i} - y_i)\right)$, where $cosh(x) = \frac{1}{2}(e^x + e^{-x})$.

While the former one was the most popular benchmark, the latter loss was an innovative variant on shape, which was advantageous for handling data with wild outliers. I would parallelly conduct the following method comparisons under these two losses, compare the training effect of the two loss functions, and probe the features and behaviors of different model refining methods under different loss choices.

Two baseline models under the respective loss function were constructed according to the elements above without any extra refinement. Then, a 500-epoch 5-fold cross validation would be conducted for the models based on the training data set. The benchmark metric for validation would be the Mean Absolute Error (MAE), which could give a straightforward interpretation of the prediction error. The average trajectory of the five sets of validation MAE calculated from cross validation would be plotted to show the growing performance of the trained weights along epochs and possibly some signals of overfitting. (To clarify the plotting procedures, I used the average MAE of the 5 cross-validation results at each epoch point as the new points to draw average trajectories.)

Without prior knowledge about which method was the best fit for our specific data scenario, five model tuning approaches were separately attached on the baseline models controlling for the network capacity and other factors; therefore, five new models for each loss were created in this step. For simplicity, the respective method for each model was simultaneously applied on both hidden layers with equal parameters (if applicable), and the parameters across different loss functions were fixed.

The first class of methods were about penalties on weights, or the regularization. Three types of classic regularizers were attempted to add into the loss functions, leading to penalties on abnormal weights. The adjustments under the three methods were shown below:

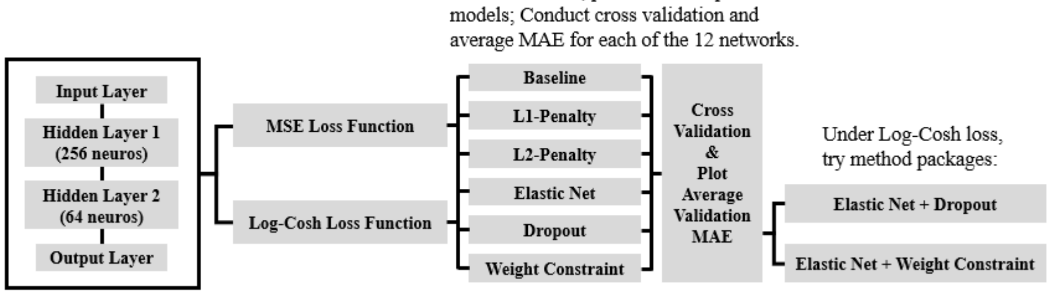
L1-Regularization (Lasso): $argmin_w \left(l(\hat{y_i}, y_i) + \lambda_1 \left| |w| \right|_1 \right)$, where $\left| |w| \right|_1 = \sum_{i=1}^{p} |w_i|$;

L2-Regularization (Ridge): $argmin_w \left(l(\hat{y_i}, y_i) + \lambda_2 \left| |w| \right|^2 \right)$, where $\left| |w| \right|^2 = \sum_{i=1}^{p} w_i^2$;

L1+L2 (Elastic Net): $argmin_w \left[l(\hat{y_i}, y_i) + \lambda \left(\frac{1-\alpha}{2} \left| |w| \right|^2 + \alpha \left| |w| \right|_1 \right) \right]$, where $\alpha \in (0,1)$.

Note that the Elastic Net method could be regarded as a combination of L1 (Lasso) and L2 (Ridge) regularizations. I chose to use lambda equal to 0.001 for all the scenarios to promote comparability (for Elastic Net, I used the same lambda for both added penalties). Next, the fourth model introduced dropout rates, which is the other regularization technique that can lower the dependencies within the network and help to improve the generalization errors [3]. In this method, the neuros in each layer might be randomly and independently removed in each iteration for both forward and backward propagation, based on a fixed probability. I chose the dropout rate as 0.1 after some pre-experiments. The fifth model introduced a unit constraint for the weights at each hidden layer, which would transform the weights to achieve a unit norm and suppress the magnitudes of all the incoming weights. Like the regularization methods before, this approach also limited the potentially crazy interim gradients to improve the overall learning process. After building the new models with various methods separately, I conducted the same process of cross validation for each of them as for the baseline network, and I monitored and collected the metric of the method performance. Finally, two comparison graphs corresponding to the two loss choices were plotted, which would show the average validation MAE trajectories of all the five methods together and offer straightforward intra-Loss and inter-Loss contrasts.

To sum up, my exploratory study could be shown in this simplified pipeline:


Under each loss, proceed in 6 separate models; Conduct cross validation and average MAE for each of the 12 networks.
Input Layer
Hidden Layer 1 (256 neuros)
Hidden Layer 2 (64 neuros)
Output Layer
MSE Loss Function
Log-Cosh Loss Function
Baseline
L1-Penalty
L2-Penalty
Elastic Net
Dropout
Weight Constraint
Cross Validation & Plot Average Validation MAE
Under Log-Cosh loss, try method packages:
Elastic Net + Dropout
Elastic Net + Weight Constraint

As noted in the graph, I also tried feasible methodological composites based on selected methods above and made final comments about their performance. Some implications and reflections would also be summarized combined with the regression context. All the coding was done in R 3.6.3. The R scripts as well as the csv-format raw data set could be found in the project folder on my GitHub.

Results and Discussion

This part will mainly show the results of cross validations within the training data set, which will provide some implications in the usage of various model tuning methods under the regression context. Some well-performed methods were selected to fit final neural networks and the prediction accuracy under the testing data set was calculated and compared.

Firstly, in a general comparison of ***Figure 1*** and ***Figure 2***, under the same evaluation metric, it is clear that the training provided by the Log-Cosh loss function "dominates" the training given by the MSE loss. The better learning experience under Log-Cosh might be attributed to some features in the data. The Log-Cosh Loss incorporated the advantages of the Absolute Loss and the Huber Loss to be less sensitive to the noises in data, while the MSE Loss might be highly distorted if there were many outliers [4]. According to this finding, I went back to the data and did identify some suspicious outliers: I believed that this problem might be due to the different data collection qualities in different countries and years, or other errors when recording or transforming the measuring units. For most of the methods, I could detect apparent advantage of Log-Cosh on both the faster convergence and the lower MAE when it was close to stabilization. Also, we could observe that the plot for Log-Cosh had a much better differentiation on the five separate methods, which means the more powerful methods performed even better under the Log-Cosh Loss.

For both validation MAE plots, the performance rank for the methods seemed to be the same. The weight constraint method had negligible contribution compared with the baseline network. I guess this was partially due to the pre-normalization of input data, which had mitigated some dangers of encountering extreme gradients. Besides, I found that the baseline and weight constraint model exhibited a slightly upward trend at large epoch numbers (in both pictures, the two lines seemed to intersect with other lines). These were signals of overfitting in the validation data set. Hence, we might say that for both losses, except for the weight constraints method, all other methods gave hands to ameliorate the overfitting problem and offered better weight trainings.

Next, we could observe the L1, L2, and their combination's trajectories together. Under the MSE Loss, they three began to show advantages compared with the baseline after about 150 epochs. The Elastic Net was superior to the L1 and L2 penalty, and the L2-regularization

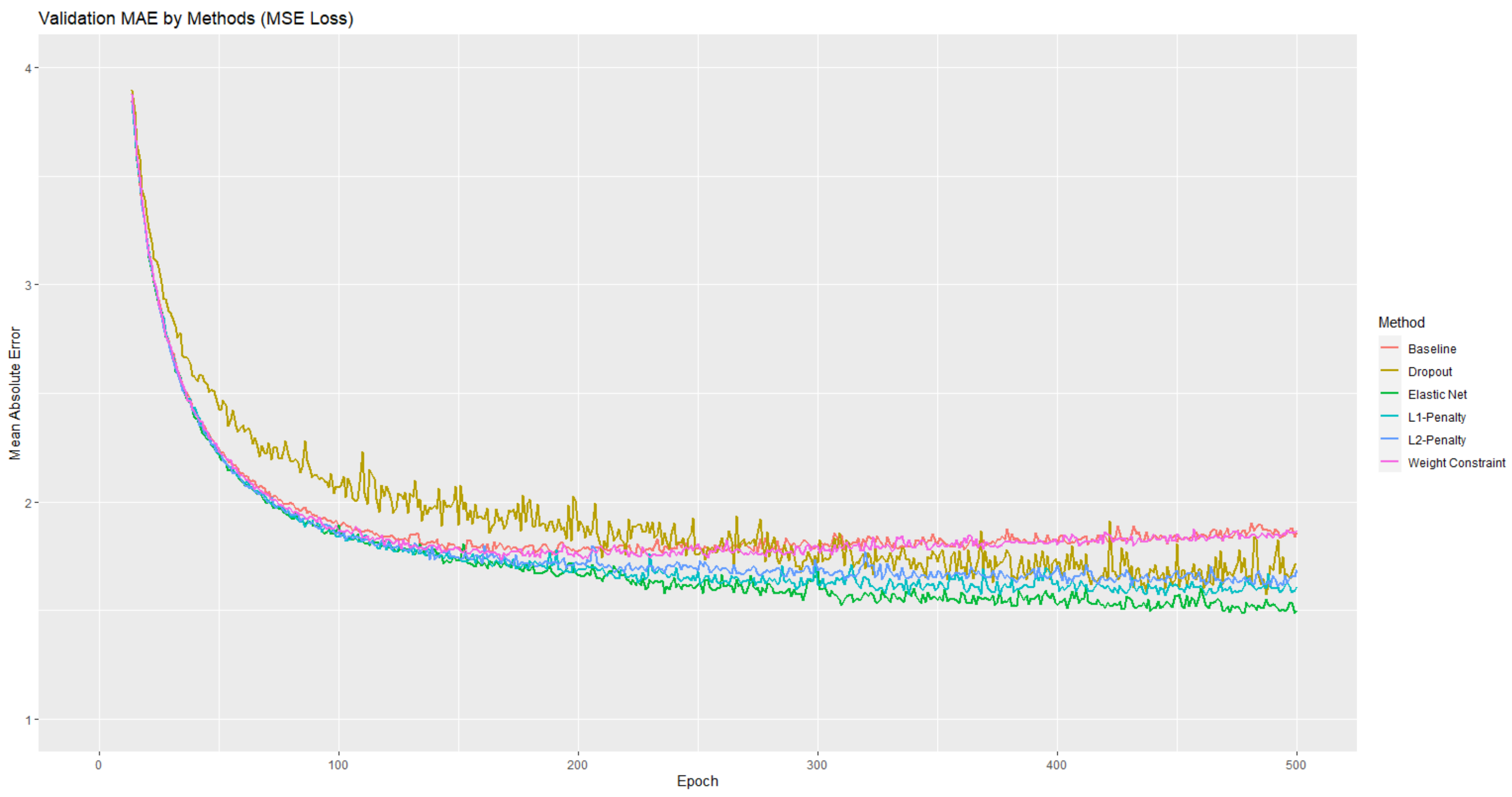


Figure 1: Method Comparison under the MSE Loss

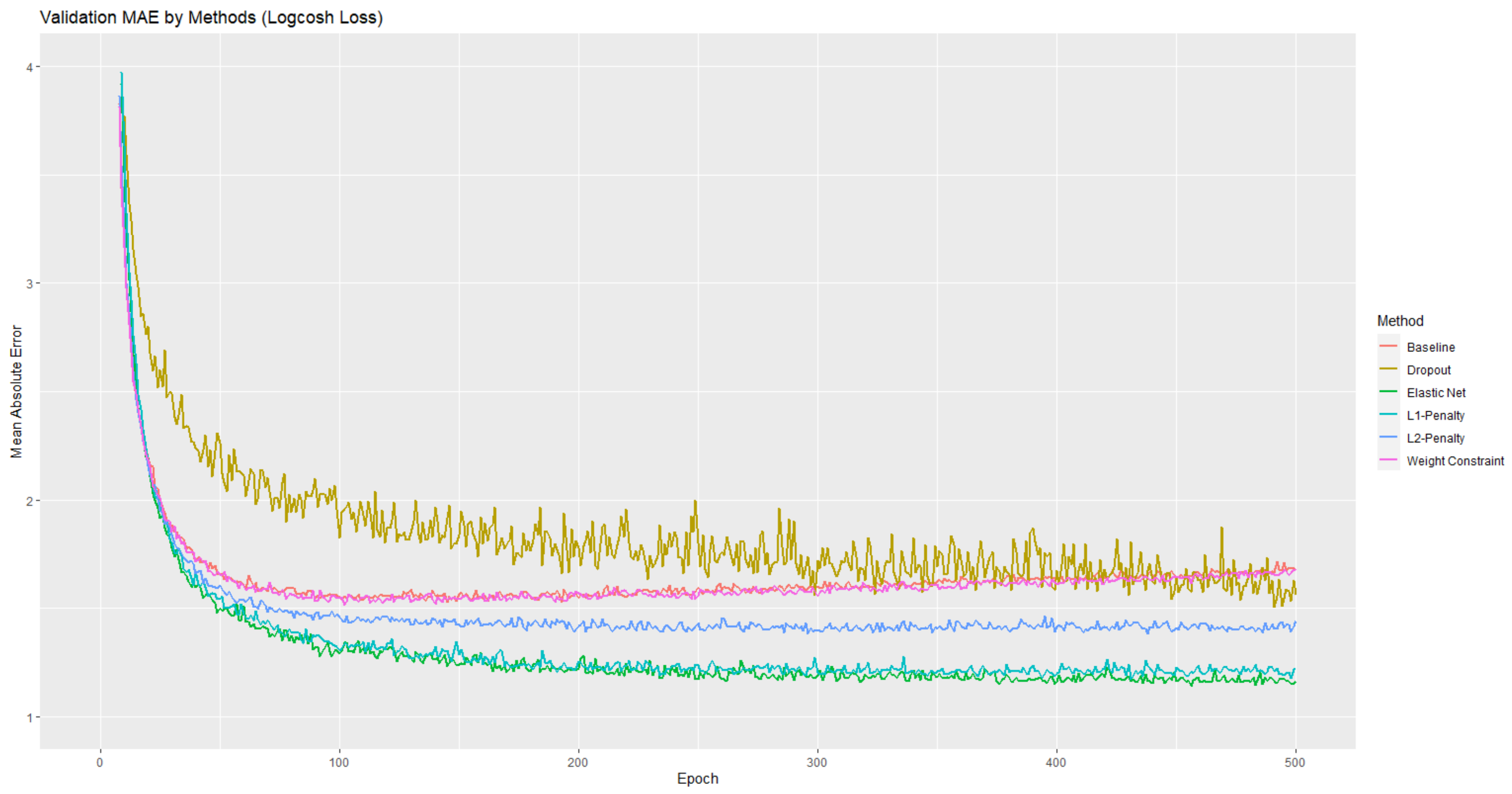


Figure 2: Method Comparison under the Log-Cosh Loss

did the worst among the three. Under the Log-Cosh loss, the three showed their advantages faster, at about the 50th epoch. Under Log-Cosh, even though the relative rank was not changed, the Elastic Net and L1-regularization had close behaviors and they performed clearly better than the L2 penalty. The L2 (Ridge) and L1 (Lasso) regularizations were widely compared in literature. While the Ridge method performed well if most of the predictors significantly influenced the response, the Lasso worked well if a small number among the correlated independent variables had large coefficients than the others [5]. I inferred that the reasons why the L1 penalty did a better job than the L2 penalty might include, we had too many categories of countries in the predicting variables (lots of binary variables after dummy coding) and many countries' national average life expectancies were much similar, which might cause many insignificant parameters in estimation. Additionally, the relatively bad performance of the Ridge might be due to the uncentered distribution of the response variable. Finally, it was evident that the combination of L1 and L2 penalties had the best performance in reducing validation MAE among the five independent effects. Compared with the Lasso, the additional quadratic penalty in the Elastic Net made the loss strongly convex and provided shortcuts to the pursued minima. The Elastic Net method seemed to combine the power of the two classic regularization methods, but unfortunately, the potential cost of this combination was not revealed in this study and might be addressed in future expeditions.

The dropout method regularized the model in an alternative mechanism. From the plots, it presented much large variations in the MAE reduction. This method had the slowest performance in decreasing MAE, even worse than the baseline model. This was reasonable based on its theory, where the working neuros in the intermediate layers were randomly shut down to mitigate potential conspiracies. Frankly speaking, it might not be appropriate to compare this method with the others, with incomparable tuning parameters and maybe it was not suitable for the regression context. The dropout method was said to be very useful in computer vision problems due to the abundance of features and the relatively limited data [2]. Also, from the plot, we might say that the 500-epoch number was not enough for this approach. Maybe in the future I should run more epochs on GPU and I could further scrutinize the long-run behaviors of the dropout method in outcome regression. Another special finding about dropout was its relatively stable behaviors under different loss cases. ***Figure 3*** was a typical example of other methods under the two different losses, which showed clear differentiation and the merit of the Log-Cosh loss function. In contrast, ***Figure 4*** showed that for the inter-Loss performance of dropout with 0.1 rate, the trajectories were close to each other and the relative weakness of MSE loss was largely undermined. From this special observation, I guess that the dropout might have some protective effect on the possibly improper choice of loss function,

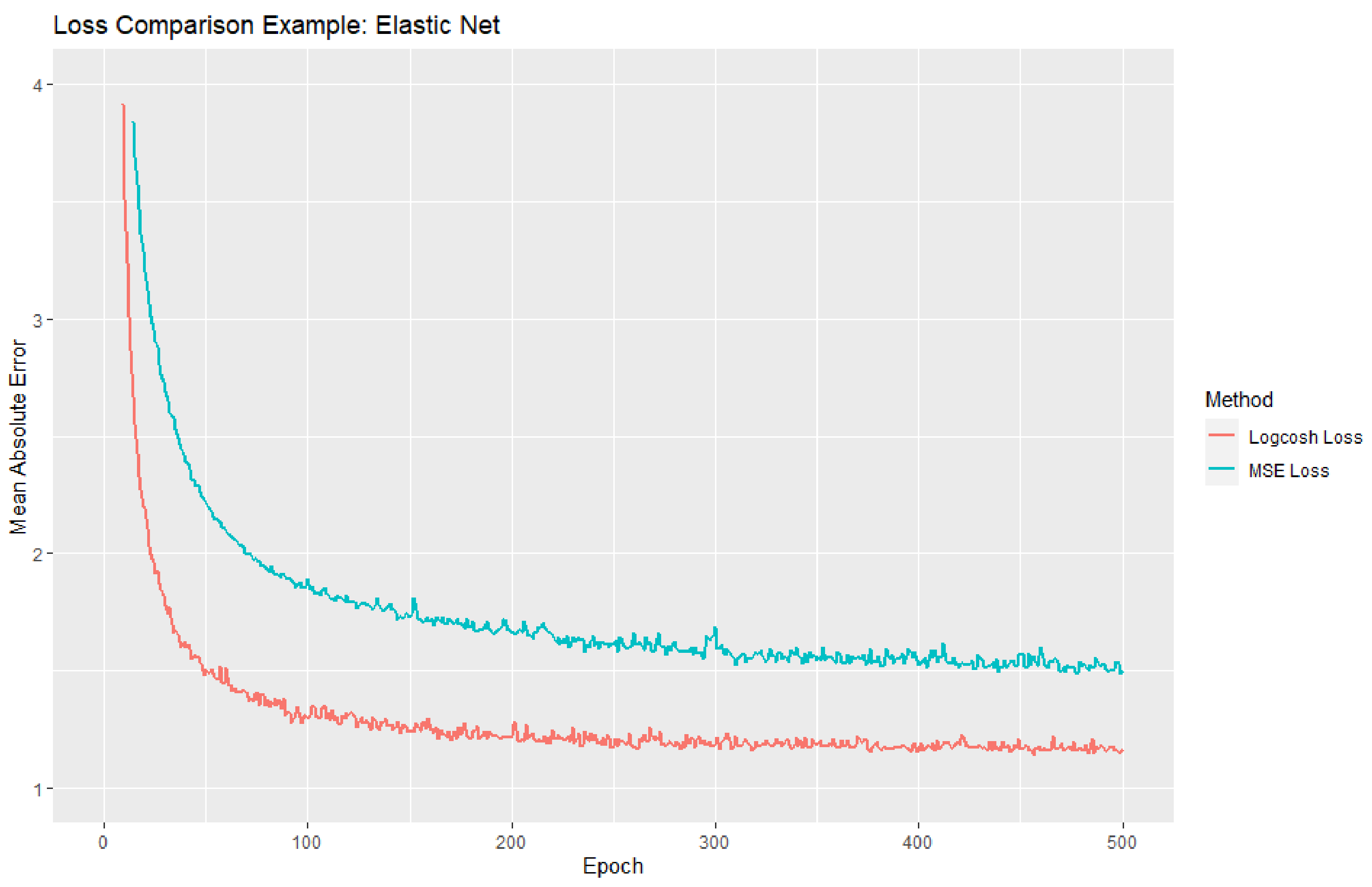


Figure 3: Loss Function Comparison under Elastic Net Method

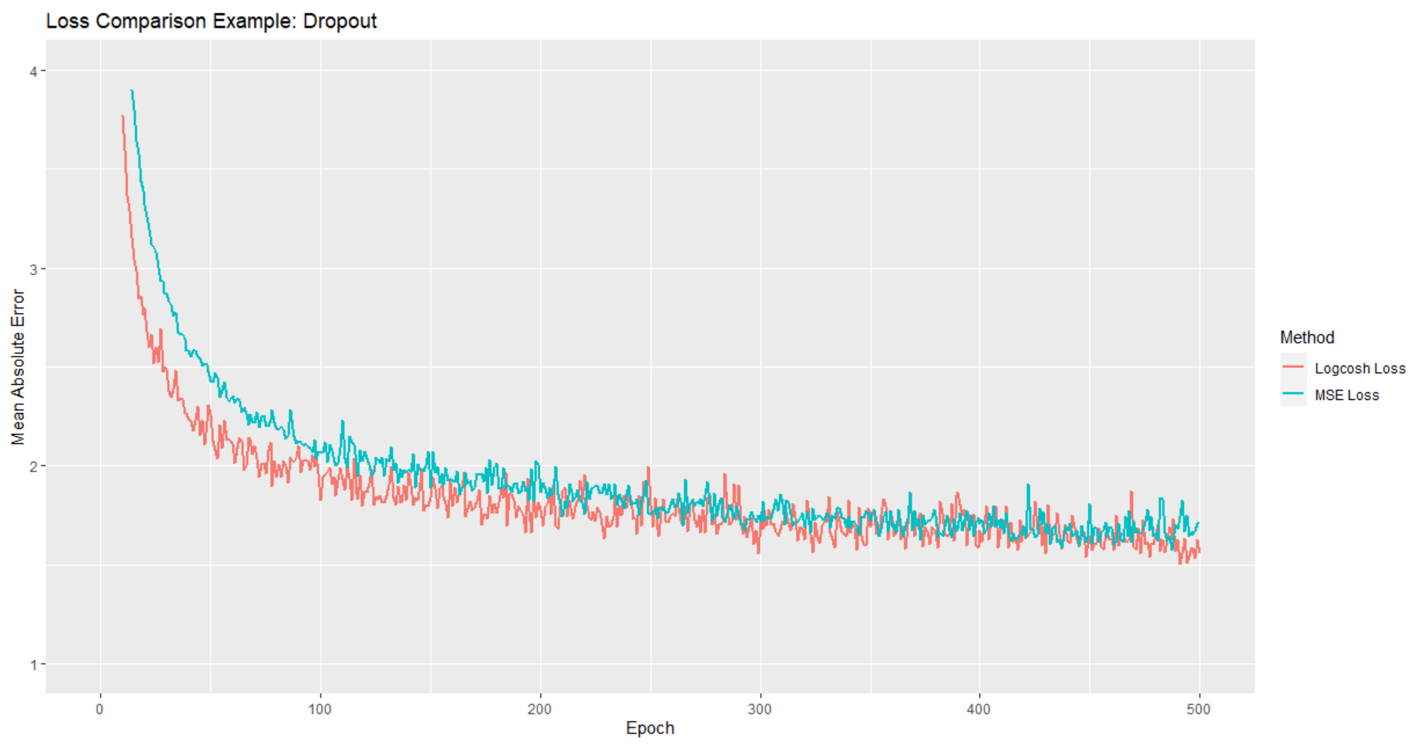


Figure 4: Loss Function Comparison under Dropout Method

and this method would take more time to converge.

After a crude comparison of "separate" methods, I was also curious about how the "method package" worked in our data story. Remaining other factors, I compared the performance of our best method, Elastic Net, to its combination with the weight constraint as well as its combination with the dropout. ***Figure 5*** provided a clear comparison for the selected methods. Firstly, I found that the cross-validation performance of Elastic Net (EN) was hard to be distinguished with the performance of "EN + Weight Constraint", and it seemed that these two together won the contest of validation MAE. However, it was surprising to me that the "EN + dropout" was much worse than the single EN, and this combination even showed slightly worse effect than the dropout singly. This reiterated the variability uncertainty in the dropout method, and it truly ignited my interest in further exploring the behavior and theories of the dropout method in my future study. From this final comparison, some takeaways in practice might be, we should be careful when using multiple methods together, the "superposition" of two independent methods possibly lead to not-better results.

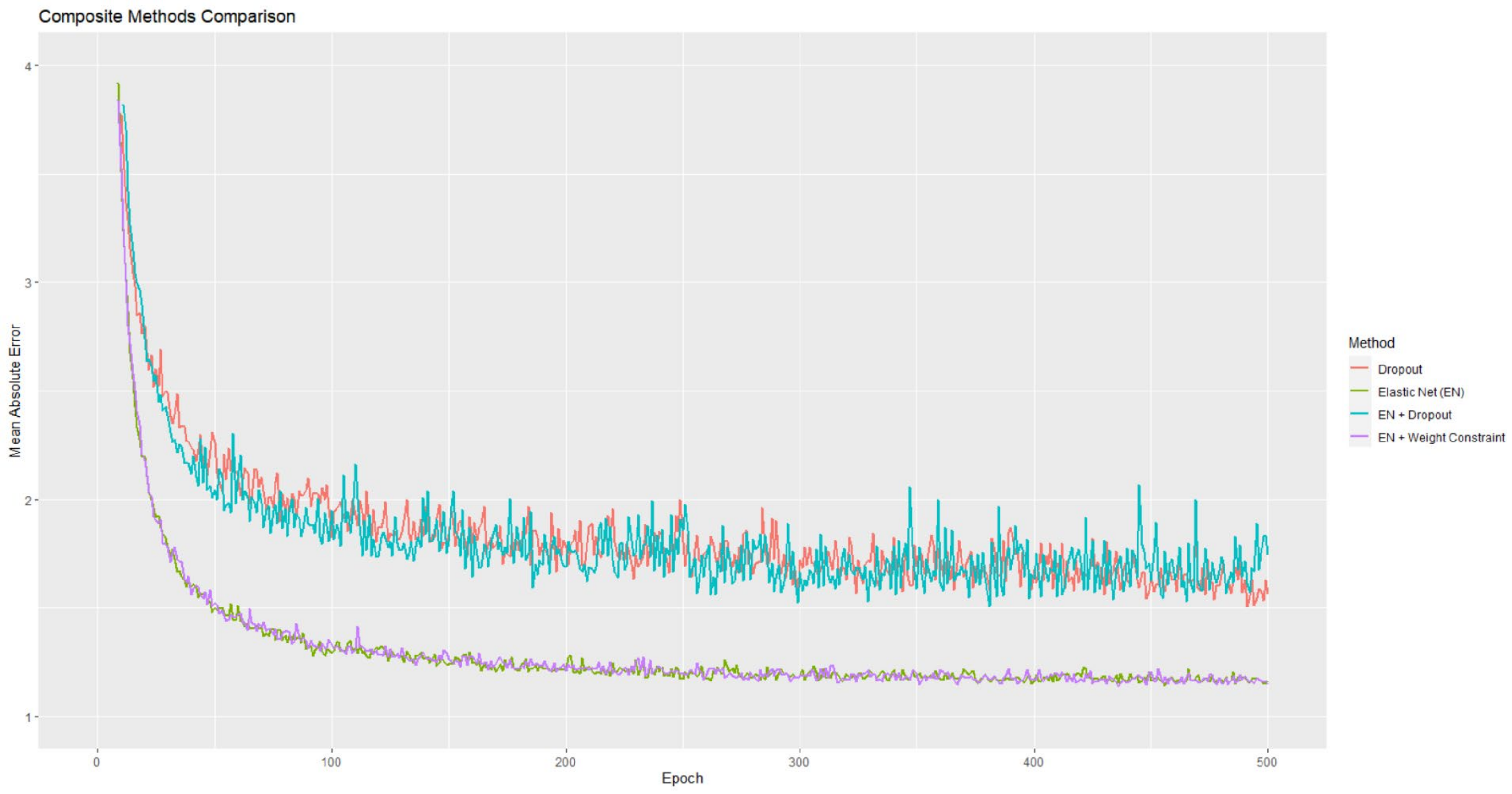


Figure 5: Comparisons for Selected Methods and Their Composites

Finally, I conducted the most important evaluation, that is the model performance in the testing data set. Note that before this line, we did not touch the testing data anymore, so the validation MAE above seemed not to be the absolute assessment standard for perfection. Below, I used 500 epoch and the whole training data to train with the selected models above (EN; EN + dropout; EN + weight constraint), and a seed was set for these final trainings. Applying these three sets of the trained weights onto the testing data set, I computed the MAE as the final benchmark of prediction competency for each method. To show the power of neural network model, I also tried common linear regression method (using “lm” with all covariates added in) for this data set, and did the similar procedure to get a comparable testing data MAE. A table about each model’s performance was shown below:

Table 1: Different models’ performance in testing data set

Method	Testing Data MAE
Elastic Net + Weight Constraint	1.1161
Elastic Net + Dropout	1.4832
Elastic Net	1.4283
Linear Regression	1.5050

We can find that the “EN + Weight Constraint” performed the best and all the deep learning methods did better than the linear regression method more or less. This comparison might involve some random factors in the initialization part and the inner mechanism of neural networks, but since the traditional regression would face much workload like variable selection or model selection when dealing with this large data set, I would like to say that for variable-abundant datasets, the neural network method was efficient in the regression/outcome prediction missions. Specifically, the final prediction by the “EN + weight constraint” only provided averagely about 1.1 year’s error than the true average life expectancy, which was fairly accurate.

Conclusion

In this project, I modeled a densely connected neural network to solve a regression problem. For novelty, I attempted several model tuning methods to improve the training process and the network’s compatibility to the data set. I found that the Log-Cosh loss function offered significantly better training effect than the traditional MSE loss under our data and this will also hold for other data sets with potential outliers. I also found that the Elastic Net method might be a refined choice for the L1-reularization, and it was particularly effective when the data set potentially included many insignificant variables (parameters). Also, I learnt about that the dropout method in regression-oriented network had slow convergence rate and high variability, but it would have good potentials for long-run performance. Plus, I realized that the weight constraint and initial normalization will make the learning process smoother and improve the quality of training. And finally, we need to be careful when implementing multiple methods in the future practice--the combination of singly-well-performed methods might not guarantee the good performance of the method combination, and an organized model selection should be conducted.

There are several limitations to be addressed in future work. Above all, what I did was just a feature comparison of methods and a crude selection of the best strategy, since I did not carefully tune the lambda parameters in regularization and other step-size parameters rigorously using cross validation to achieve an inner-method optimization. Additionally, limited by the running speed on CPU, I did not try larger epoch numbers and more complex network structures. Like what was mentioned in the dropout part, some interesting features of some methods might not be fully revealed under the current model capacity. In addition, I was also able to construct other neural network models based on this data set. Since "65 years" is an internationally stipulated national average level of life expectancy, I believe that the countries below this threshold will be given strong suggestions to improve the public health constructions. Therefore, it is possible to use this data set to train another network for binary (or multiclass) classification. For this new aim, we could change the output-layer activation function to the appropriate ones to output probabilities for classification (like "softmax", or "sigmoid" for binary prediction), and we should also change the loss function to "crossentropy" [6]. Additionally, the benchmark for cross validation should also change to the classification accuracy rate. I believe that with these simple adjustments to the "EN + weight constraint" model, the network will be primarily good for this alternative application under the same data scenario.

Appendix: Variable Description for the Life Expectancy Dataset

Variable Name (Type)	Variable Description
Life Expectancy (Continuous)	National average life expectancy in age
Country (Categorical)	Country names, 183 in record
Year (Continuous)	Year names from 2000 to 2015, processed as 0 to 15 as numeric
Status (Binary)	Developed or developing status
Adult Mortality Rate (Continuous)	Adult mortality rates of both sexes, computed as the probability of dying between 15 and 60 years per 1000 population
Infant Death Rate (Continuous)	Number of infant deaths per 1000 population
Alcohol Consumption (Continuous)	Alcohol per capita (15+) consumption, in liters of pure alcohol
Health-Related Expenditure (Continuous)	Expenditure on health as a percentage of Gross Domestic Product (GDP) per capita (%)
Hepatitis B Immunization (Continuous)	Hepatitis B immunization coverage among 1-year-olds (%)

Measles Immunization (Continuous)	Measles reported cases per 1000 population
Average BMI (Continuous)	Average body mass index of entire population
Under-Five Death (Continuous)	Number of under-five deaths per 1000 population
Pol3 Immunization (Continuous)	Polio (Pol3) immunization coverage among 1-year-olds (%)
Government Expenditure (Continuous)	General government expenditure on health as a percentage of total government expenditure (%)
DTP3 Immunization (Continuous)	Diphtheria tetanus toxoid and pertussis (DTP3) immunization coverage among 1-year-olds (%)
HIV/AIDS (Continuous)	Deaths per 1000 live births HIV/AIDS (0-4 years)
GDP (Continuous)	GDP per capita (in USD)
Population (Continuous)	Population of the country
Thinness 10-19 (Continuous)	Prevalence of thinness among children and adolescents for age 10 to 19 (%)
Thinness 5-9 (Continuous)	Prevalence of thinness among children for Age 5 to 9 (%)
Human Development (Continuous)	Human development index in terms of income composition of resources (index ranging from 0 to 1)
Schooling (Continuous)	Average number of years of schooling (years)

Reference

[1] Santrock, John (2007). Life Expectancy. A Topical Approach to: Life-Span Development (pp. 128–132). New York, New York: The McGraw-Hill Companies, Inc.
[2] Kingma, D. P., & Ba, J. L. (2015). Adam: A Method for Stochastic Optimization. International Conference on Learning Representations, 1–13.
[3] Dropout in Neural Networks. Retrieved from https://towardsdatascience.com/coding-neural-network-dropout-3095632d25ce
[4] Loss Functions for Regression in Machine Learning. Retrieved from http://www.cs.cornell.edu/courses/cs4780/2015fa/web/lecturenotes/lecturenote10.html
[5] Regularizations. Retrieved from https://www.datacamp.com/community/tutorials/tutorial-ridge-lasso-elastic-net
[6] Chollet, F. Allaire, J.J. Deep Learning with R.