Allstate Claims Severity in the Kaggle Competition

Gon-soo Moon
Master Student of Computer Science
University of Texas at Dallas,
Richardson, TX, USA
gxm151030@utdallas.edu

Steve Raftery
Master Student of Computer Science
University of Texas at Dallas,
Richardson, TX, USA
sxr161130@utdallas.edu

Bohao Chen
Master Student of Computer Science
University of Texas at Dallas,
Richardson, TX, USA
bxc151130@utdallas.edu

Abstract

The problem with a prediction of a loss value in the insurance claim is posted in the Kaggle competition. Involving pre-processing techniques such as "*1-of-k-encoding*" and outliers, a formalized learning procedure including the "*three-way data splits*" and our methodology of hyperparameters selection consisting of four phases, activation, architecture, outliers and epochs, and learning rate including regularization, this paper tackles the "*Allstate claims severity*" problem using a *deep neural network* model. Experiments with 21 entries to the competition demonstrates how to reach to a minimum error.

1. Introduction

This paper handles how to solve a regression problem of "how severe is an insurance claim?" named "*Allstate claims severity*[1]" on the *Kaggle* data science competition. The problem is described that "*Allstate* is currently developing automated methods of predicting the cost, and hence severity, of claims" on the instruction of the competition. Before evaluating a final loss value taking some time, roughly a couple of days, if a customer can obtain an approximated loss value in near real time based on some information such as his or her policy and claim details, an insurance company could give a better experience to their customers. We think that this is why Allstate company has posted the competition on Kaggle.

For solving the problem, we applied a single *deep neural network (DNN)* on feed-forward networks based on the back-propagation algorithm. Handling more complex problems, DNN usually performs well with more hidden layers and neurons as well as sophisticated techniques such as dropping neurons as one of the regularization techniques than a neural network(NN). This is why we choose deep neural network. From a tool perspective, the H_2O software[2], written in Java, having an interface to R, Python, Java and Scala is used because a single-node cluster that is our experimental environment can be easily fitted even though it depends on both how much data size is and how much a computing power requires.

The following major sections are a related work, data set description, pre-processing techniques, proposed solution and methods, experimental results and analysis, and conclusion. Especially, in the proposed solution and methods, we introduce four phases of hyperparameters selection. Accordingly, we experiment our solution and then submitted it to the Allstate claims severity competition. Thus, the results of 21 entries are shown in the experimental results and analysis section.

2. Related Work

The paper, *Applying Data Mining Techniques in Property/Casualty Insurance by Lijia Guo,* is concerned with the techniques and issues related to machine learning and data mining techniques for casualty and property insurance. The insurance industry is involved in predicting losses due to claims and uses machine learning techniques to classify claims and to predict future claims based on attributes of policy holders. The paper covers different machine learning operations and techniques that are often used together within the insurance industry to predict claims. Many of the methods used are ones that we have studied in class such as k-means clusters, logistical regression, decision trees and neural networks. All of these methods have their strengths and weaknesses and the author stresses how they are used together through parallel and serial processing to produce the best prediction models. The primary goal of the industry is to combine pattern recognition, machine learning, database theory, and statistics to learn from the huge amounts of data to predict losses from claims. The paper discussed the above mentioned machine learning operations, and they're positive features and their drawbacks and how they are applied to claims in the insurance industry.

3. Data Set Description

[1] https://www.kaggle.com/c/allstate-claims-severity
[2] http://www.h2o.ai

Two data files are provided on the Kaggle web site supported by Allstate company. They are train.csv and test.csv. The train.csv consists of attributes and loss value (class label). The test.csv is used to predict the loss value in this file using our model. Each row in this dataset represents an insurance claim. We must predict the value for the 'loss' column. Variables prefaced with 'cat' are categorical, which describe some properties for attributes, while those prefaced with 'cont' are continuous, which describe some specific value for attributes. Both train file and test file don't have any missing values. In the train file, there are 188,318 rows of data and 130 attributes (116 categorical and 14 continuous) with one class label. In the test file, there are 125,546 rows of data and 130 attributes (116 categorical and 14 continuous) without a class label. The distribution of loss value is shown below.

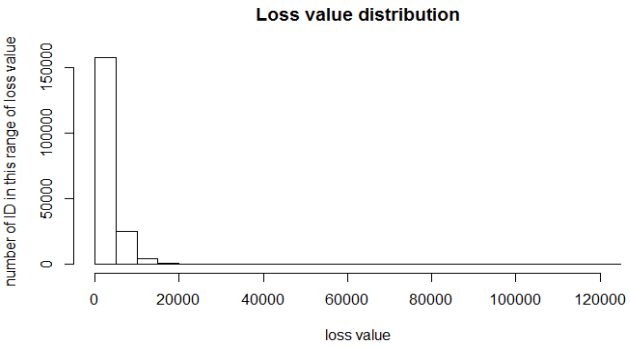


Figure 3.1. Loss value distribution as a label

4. Pre-processing Techniques

We apply three pre-processing techniques. One is "*1-of-k-encoding"* where values of a category are broken down. If a category has three different values, it is broken down into three different attributes of Boolean values. For instance, the attribute Category80 that has values A, B and C becomes 3 attributes named Category80_A, Category80_B and Category80_C. Each attribute has a Boolean value of 1 or 0. In our case, after 1-of-k-encoding applied, 116 categorical attributes are converted to 1,176 Boolean attributes. Consequently, a final number of attributes to be trained is 1,190 that is 1,176 plus 14 continuous attributes.

Second is to remove outliers. As shown in the Figure 4-1, some examples can be designated as outliers away from a density of most points so that we set a threshold value to a loss value of 30,000, removing 63 examples. An effect of outliers is thoroughly handled in the 6.2.4 Outlier and Epochs Phase section.

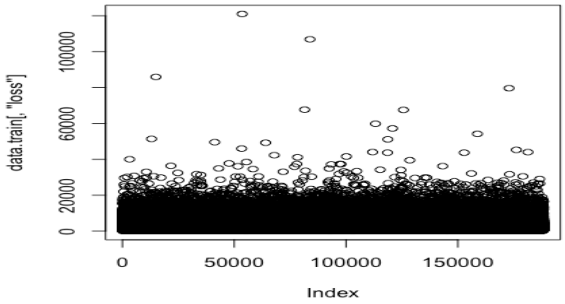


Figure 4-1. A plot of the loss attribute showing outliers

Last, all attributes after the 1-of-k-encoding applied are scaled using a statistical normalization. In other words, a mean of an attribute is subtracted from a value of the attribute and is divided by a standard deviation of the attribute.

5. Proposed Solution and Methods

This section has three sub-sections that are the 5.1 Tool used: H_2O, 5.2 Learning Procedure and 5.3 Hyperparameters Selection. The first briefly explains major features of H_2O and the second covers how to train and evaluate models and then select the best model with a specific hyperparameters setting on a validation data set. The third shows a method of how to find an optimal hyperparameters setting.

5.1 Tool used: H_2O

H_2O is an open-source software for machine learning and deep learning by *H2O.ai* company. It supports a scalable cluster environment such as a single-node cluster and multi-node cluster on parallel computation and a memory compression so that a simple as well as complex problem can be handled. Some of the major features are an adaptive learning rate for fast convergence, advanced activation functions and recent techniques of regularization, *"Dropout, HOGWILD and model averaging*" including "*L1 and L2*." We experiment the several techniques such as the dropout, a few activation functions, the adaptive learning rate and others. For more information, you can refer to the document[3], "*The deep learning with H_2O*".

[3] http://docs.h2o.ai/h2o/latest-stable/h2o-docs/booklets/DeepLearningBooklet.pdf

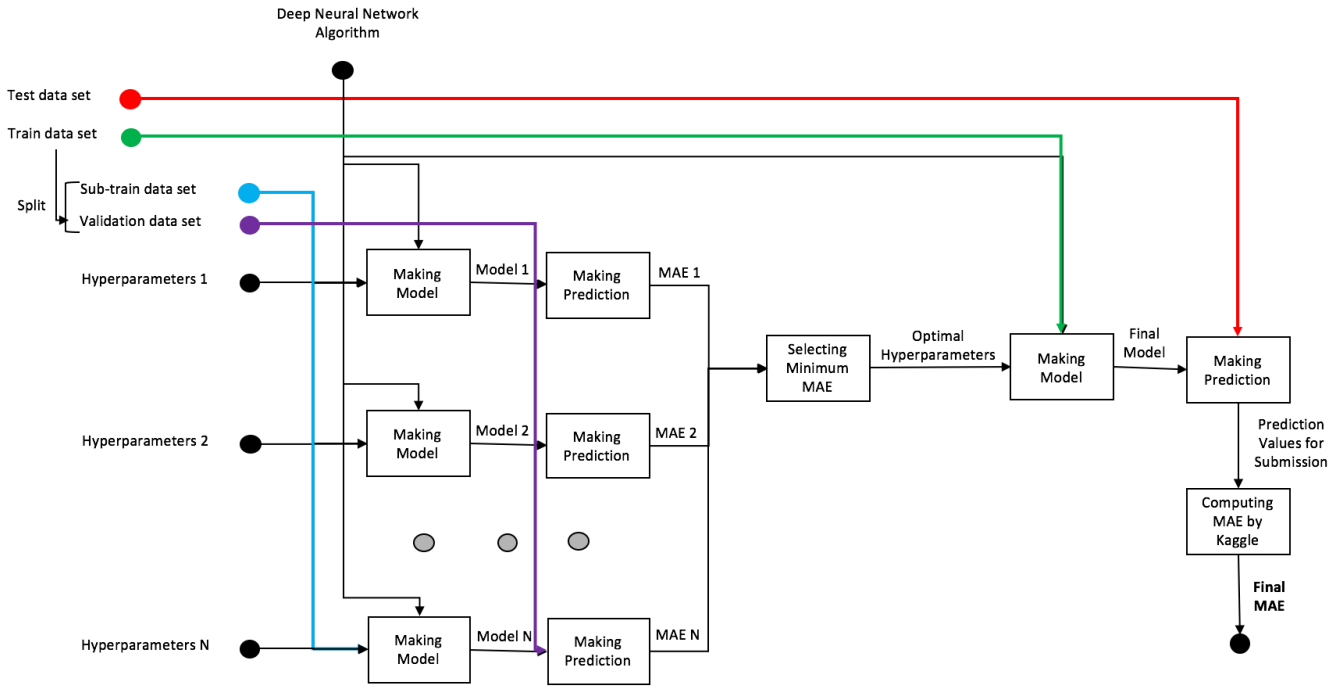


Figure 5-1. Learning Procedure

5.2 Performance Metric

This competition uses MAE (Mean Absolute Error) performance metric. The instruction[4] states that "the mean absolute error(MAE) is a quantity used to measure how close forecasts or predictions are to the eventual outcomes." The formula is:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y_i}| = \frac{1}{n}\sum_{i=1}^{n}|e_i|$$

where Actual = y_i, Predicted = $\hat{y_i}$

5.3 Learning Procedure

The figure 5-1 shows what our learning procedure is graphically. We use "*three-way data splits*[5]". One is a test data set for making a final prediction to be submitted as well as a sub-train data set and validation data set, into which a train data set for a final model is divided, are used for training models and making predictions respectively. We use two ratios of 80:20 and 95:5 depending on prediction results. Briefly, there are seven steps that are 1) the three-way data split, 2) to make multiple models with each set of a different hyperparameters setting and a deep neural network algorithm on the same sub-train data set, 3) to make predictions each for the models on the same validation data set, providing every MAE, 4) to select minimum MAE as a best model and then choose the hyperparameters setting used for the best model, called an optimal hyperparameters setting, 5) to make a final model with a deep neural network algorithm and the optimal hyperparameters setting on the train data set, 6) to make a prediction for the final model on the test data set, then providing predicted values of the loss label, and consequently 7) to submit them to the competition of Allstate claims severity on Kaggle, which in conclusion provides a final MAE.

5.4 Hyperparameters Selection

Referring to the *Practical Recommendations for Gradient-Based Training of Deep Architecture* (Bengio, 2012), our methodology to choose an optimal hyperparameters setting divides important hyperparameters into four phases, we define, that are 1) an activation, selecting activation function (e.g. sigmoid, tangent hyperbolic), 2) an architecture, determining numbers of hidden layers and neurons, 3) an outlier and epochs, removing outliers and deciding a number of iterations and finally 4) a learning rate including regularization, changing a default of adaptive learning rate to a user-defined learning rate.

The first phase of the activation is to experiment several activation functions, TanhWithDropout (Tan hyperbolic with Dropout), RectifierWithDropout (Rectifier Linear with Dropout) and MaxoutWithDropout (Maxout with Dropout). *The deep learning with H_2O* states that "the deep learning with the tanh function is a rescaled and shifted logistic function; its

[4] https://www.kaggle.com/wiki/MeanAbsoluteError
[5] http://research.cs.tamu.edu/prism/lectures/iss/iss_l13.pdf

symmetry around 0 allows the training algorithm to converge faster. The rectified linear activation function has demonstrated high performance on image recognition tasks and is a more biologically accurate model of neuron activations (LeCun et al, 1998). Maxout is a generalization of the Rectified Linear activation, where each neuron picks the largest output of k separate channels, where each channel has its own weights and bias values." In the MAEs obtained on the validation data set, MaxoutWithDropout shows less performance so that we submit final predictions on the models only with both TanhWithDropout and RectifierWithDropout. We show results in the section 6.2.1 in detail.

In the second phase of the architecture, we mainly focus on the numbers of layers and neurons between 50 neurons in a single layer and 1200 neurons in three layers. Finally, we submit predicted values obtained from three options that are (50) neurons in a single layer, (200, 200, 200) neurons in three layers and (500, 500) neurons in two layers.

Until the third phase of the outlier and epochs, we didn't remove outliers because we thought that the deep neural network model could be fitted with all data examples including outliers. As an experiment, we removed 63 examples having greater than 30,000 loss value. The experiment showed us meaningful results. Also, we varied a number of epochs between 50 and 200. Finally, we found an optimal value of 100 epochs even though most experiments had "*an early-stopping*" occurrences because MAE had converged.

In the last phase of the learning, we didn't use a default adaptive learning rate but manually tuned a fixed value learning rate annealing that states "learning rate annealing gradually reduces the learning rate αt to "freeze" into local minima in the optimization landscape (Zeiler, 2012)" in *the deep learning with H_2O.* Among experiments ranging from 0.01 to 0.0001 learning rate we finally found 0.0001 as an optimal value.

6. Experimental Results and Analysis

6.1 Result

We made 21 submissions to the competition as shown in Figure 6-1, Final MAEs provided by Kaggle. As it shows, there are two benchmarks; one is MAE of Random Forest Benchmark provided by Kaggle and the other is MAE obtained by the top entry ranked. Our result is between 1,292 MAE as a maximum and 1,169 MAE as a minimum.

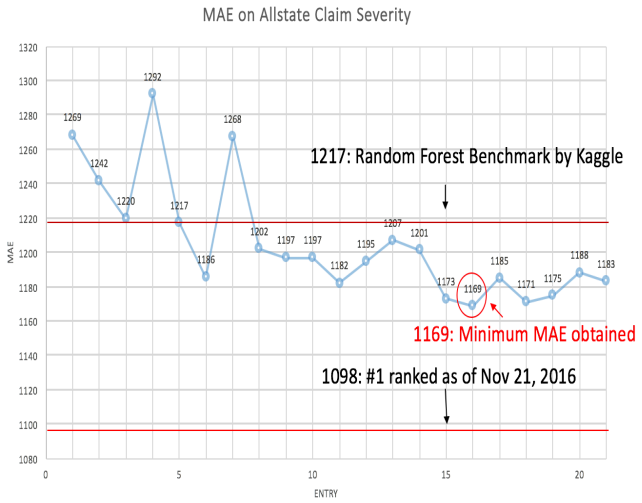


Figure 6-1. Final MAEs provided by Kaggle

6.2 Four Phases on Experiment

As shown in the Figure 6-2 MAEs in the four phases, 21 entries are divided into the four phases based on related hyperparameters. We took a closer look at each phase, going through each entry with hyperparameters used.

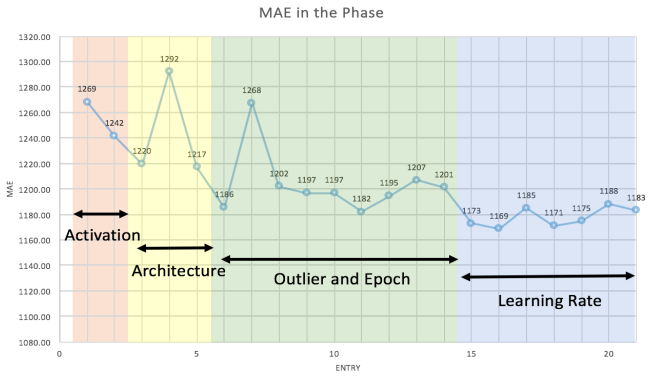
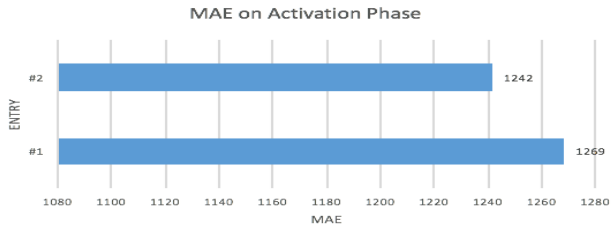


Figure 6-2 MAEs in the four phases

6.2.1 Activation Phase

As shown in Figure 6-3, Hyperparameters on the activation phase, there are two entries. Fixing other hyperparameters such as hidden layers set to 50 in one layer, epochs set to 100 and other hyperparameters set to a default provided by the *deeplearning* function in H_2O, we varied activation functions, TanhWithDropout and RectifierWithDropout. Thus, RectifierWithDropout showed better performance.

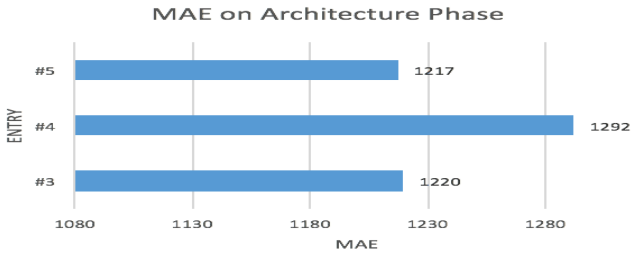


Entry#	MAE	Activation	Hidden Layer	Epochs
#1	1268.60	*TanhWithDropout*	*50*	*100*
#2	*1241.74*	*RectifierWithDropout*	*50*	*100*

Figure 6-3 Hyperparameters on the Activation Phase

6.2.2 Architecture Phase

In this phase, before making entries to the competition, we experiment many combinations of numbers of hidden layers and neurons. Finally, we select three sets of hyperparameters settings as shown in Figure 6-3, Hyperparameters on the Architecture Phase. Among three entries, both the #3 and #5 show similar MAEs even though they have different hyperparameters settings of the hidden layer, regularization hyperparameters - L1 and L2, hidden_dropout_ratio and 1-of-k-encoding. Finally, we choose #5, also showing a minimum MAE among the three entries, because more neurons, adding up to 1,000 in the #5 entry, means that not only do they have more capacity and flexibility to handle more complex structure, but also they can support more generalization with appropriate regularization. For the 1-K-Encoding in the Figure 6-3, the same as 1-of-k-encoding, '0' means to use H_2O 1-of-k-encoding and '1' means to use our implementation. Even though if a categorical attribute goes in, H2O supports it as a default, we used our implementation because we thought it necessary to apply the same statistical normalization function to both categorical and continuous attributes explicitly. Consequently, we mostly used values of the hidden layer, hidden_dropout_rate and 1-K encoding used in the #5 entry for the next phases of the outlier and epochs as well as the learning rate phase.

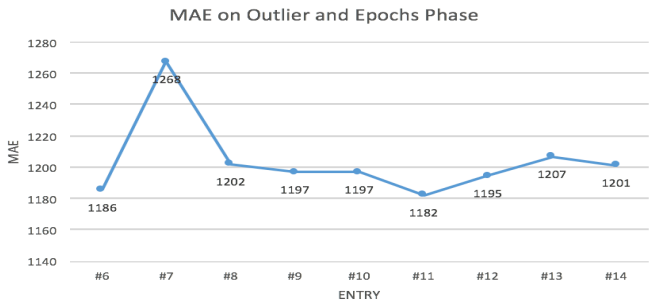


Entry#	MAE	Activation	Hidden Layer	L1	L2	hidden_dropout_ratio	1-K Encoding
#3	1219.77	RectifierWithDropout	*100*	*Default*	*Default*	*0.1*	*0*
#4	1292.09	RectifierWithDropout	*200,200,200*	*Default*	*Default*	*0.3*	*1*
#5	*1217.20*	*RectifierWithDropout*	*500,500*	*1.00E-04*	*1.00E-04*	*0.2*	*1*

Figure 6-3 Hyperparameters on the Architecture Phase

6.2.3 Outlier and Epochs Phase

We didn't expect that outliers matter is at this phase before. In order to decrease the minimum MAE of 1,217 from the preceding phase, we removed outliers from the loss (class) attribute. In the Figure 6-4 below, except for #7 entry, all entries in this phase are below 1,217 MAE, which means that removing outliers had a significant effect on performance. On the other hand, though we varied a number of epochs, early-stoppings occurred in most cases because of convergence when training a final model. Finally, we choose 100 of epochs as an optimal value. For both #6 and #11 entry, we used the same setting of hyperparameters. However, the reason that they showed a different MAEs is that when we submitted our predicted values of the loss label, Kaggle samples some portion out of the predicted values to compare with the ground-truth data.

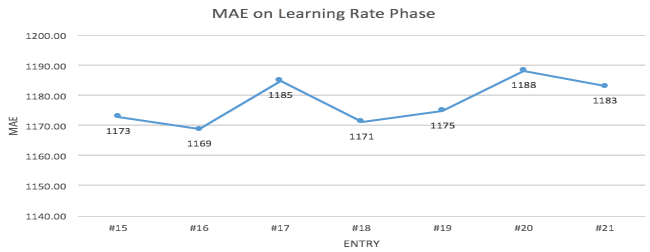


Entry#	MAE	Hidden Layer	Epochs	L2	Outlier	hidden_dropout_ratio	1-K Encoding
#6	1185.66	500,500	*100*	*1.00E-04*	*0*	0.2	1
#7	1267.64	500,500	*100*	*1.00E-03*	*0*	0.2	1
#8	1202.19	600,600	*100*	*1.00E-03*	*0*	0.2	1
#9	1197.08	500,500	*120*	*1.00E-04*	*0*	0.2	1
#10	1197.00	500,500	*140*	*1.00E-04*	*0*	0.2	1
#11	*1182.20*	*500,500*	*100*	*1.00E-04*	*0*	0.2	1
#12	1194.64	500,500	*120*	*1.00E-04*	*0*	0.2	1
#13	1207.00	500,500	*120*	*1.00E-04*	*0*	0.2	1
#14	1201.43	500,500	*120*	*1.00E-04*	*0*	0.2	1

Figure 6-4 Hyperparameters on the Outlier and Epochs Phase

6.2.4 Learning Rate Phase

Turning off an adaptive learning rate as a default, we tested a group of learning rate hyperparameters: learning rate, learning rate annealing and momentum. Also, varying L2 regularization, we obtained all MAEs below 1190 and compared them to the MAEs above 1190 in the preceding phase. In conclusion, we achieved 1168.88 as shown in Figure 6-5, the minimum MAE out of 21 entries.



Entry#	MAE	Hidden Layer	L2	Rate	rate_ annealing	max _w2	momentum _start	momentum _stable	momentum _ramp
#15	1173.00	500,500	1.00E-04	1.00E-04	1.00E-07	Default	Default	Default	Default
#16	1168.88	500,500	5.00E-04	1.00E-04	1.00E-07	Default	Default	Default	Default
#17	1185.00	500,500	1.00E-03	1.00E-04	1.00E-07	Default	Default	Default	Default
#18	1171.27	500,500	5.00E-04	1.00E-04	1.00E-07	Default	Default	Default	Default
#19	1174.95	500,500	5.00E-04	1.00E-04	1.00E-07	10	0.3	0.6	1.00E+07
#20	1188.32	500,500	5.00E-04	1.00E-04	1.00E-07	10	Default	Default	Default
#21	1183.14	500,500	3.00E-05	1.00E-04	1.00E-07	Default	Default	Default	Default

Figure 6-5 Hyperparameters on the Learning Rate Phase

6.3 Analysis

As shown in Table 6-1 Final hyperparameters Setting, for the minimum 1,168.88 MAE on the test data set, we trained a final model on 188,255 examples, after removing 63 outliers from 188,318 records. The data consisted of 1,190 attributes: 1,176 Boolean attributes by1-of-k-encoding from the original 116 categorical attributes and 14 continuous attributes, with the optimal setting of hyperparameters, which is selected across the phases.

Entry#	MAE	Activation	Hidden Layer	Epochs	L1	L2	Rate	rate_ annealing	hidden_ dropout_ ratio	Outlier	1-K Encoding
#16	1168.88	RectifierWithDropout	500,500	100	1.00E-04	5.00E-04	1.00E-04	1.00E-07	0.2	0	1

Table 6-1 Final Hyperparameters' Setting

7. Discussion and Future Work

We didn't realize that a single node H_2O cluster on *Intel Core i5 of our main computer* would take about 15 hours for training of a *single* deep neural network model at which a number of neurons are 1,000 in two layers, a learning rate is 0.0001, and a number of epochs are 100 on 188,255 examples and 1,190 attributes. It meant that we could not increase the number of neurons and layers because of a lack of computational capacity. More importantly, as one of the golden rules when participating in Kaggle competition we wanted to apply "*Gradient Boosting (Stacking)*", one of the ensemble methods, to this problem. However, our experimental environment in a single node cluster on *Intel Core i5* didn't allow us to experiment it because of a computational power. For future work, it might be much better to use Gradient Boosting having multiple deep neural network models getting along with other models on a multi-node cluster for some complex problem.

8. Conclusion

With a single deep neural network model to which we applied a group of hyperparameters settings and pre-processing techniques such as 1-of-k-encoding and removing outliers through the four phases of the activation, architecture, outliers and epochs as well as learning rate including regularization, we tried to solve Allstate claims severity on the Kaggle competition, resulting in our minimum MAE. Even though we didn't beat the top MAE of 1,098 as of Nov 21, 2016, we achieved a MAE of 1,168.88 with our solution and methods on a single-node H2O cluster having a limited computing power. As this project progressed, we learned a lot so that this work plays a significant role of our future research.

9. Contribution of Team Members

Steve and Bohao mainly focus on problem description, the related work, data set exploration and some experiments together. Gonsoo was the project leader and was a key player in doing pre-processing techniques, the solution and methods, experiments and analysis.

References

Arno Candel, Viraj Parmar, Erin LeDell, Anisha Arora (2016). Deep Learning with H2O. Available from http://h2o2016.wpengine.com/wp-content/themes/h2o2016/images/resources/DeepLearningBooklet.pdf

H2O.ai (2016). H2O package in R (Version 3.10.0.8) [Software]. Available from https://www.kaggle.com/c/allstate-claims-severity

Joshua F. Wiley (2016). R Deep Learning Essentials. Packt Publishing Ltd.

Kaggle.com (2016). Allstate claims severity. Available from https://www.kaggle.com/c/allstate-claims-severity

Lijia Guo (2003). Applying data mining techniques in property/casualty insurance. In CAS 2003 Winter Forum, Data Management, Quality, and Technology Call Papers and Ratemaking Discussion Papers, CAS

Nick Pentreath (2015). Machine Learning with Spark. Packt Publishing Ltd.

Ricardo Gutierrez-Osuna. Lecture 13: Validation. Available from http://research.cs.tamu.edu/prism/lectures/iss/iss_l13.pdf

Yoshua Bengio (2012). Practical Recommendations for Gradient-Based Training of Deep Architectures. Neural Networks: Tricks of the Trade, 2012 – Springer.

Yoshua Bengio (2013). Deep Learning of Representations: Looking Forward. SLSP 2013: Statistical Language and Speech Processing pp 1-37.