CS6375 Machine Learning

Assignment 2

Enhanced Neural Network

Author:
Randy Suarez Rodes
Net ID: rxs179030

Supervisor:
Anurag Nagar Ph.D.

Contents

1 Part 2

1.1 Enhancement

We have selected the AMSGrad algorithm as our enhancement for the Gradient Descent algorithm applied on a Neural Net backpropagation process. AMSGrad is a variant of the Adam algorithm and as Adam, it stores an exponentially decaying average of past gradients (m_t) and past squared gradients (v_t). Different than Adam, it ensures to use the maximum of past squared gradients v_t. Also for simplicity, the authors of the algorithm removed the debiasing step of Adam. (see towardsdatascience and ruder.io/optimizing-gradient-descent)

The process of updates of AMSGrad is as follows (t represents time stamp of each epoch):

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\frac{\partial L}{\partial w_t}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)\left(\frac{\partial L}{\partial w_t}\right)^2$$

$$\hat{v}_t = max(\hat{v}_{t-1}, v_t)$$

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{(\hat{v}_t)} + \epsilon} m_t$$

where m and v are initialized as zero. And L is the squared-loss function:

$$L = \frac{1}{2 \cdot n_{samples}} \sum_{i=1}^{n_{samples}} (t_i - o_i)^2$$

where t_i is the actual value and o_i is the output obtained from our neural net.

ϵ is a small floating point value to ensure that we will never divide by zero. β_1 and β_2 are constants with common default values of 0.9 and 0.999.

1.2 Data Description

We have selected for our assignment the data set Seoul Bike Sharing Demand.

Data Information (source: UCI Machine Learning Repository website)

The dataset contains weather information (Temperature, Humidity, Windspeed, Visibility, Dewpoint Temperature, Solar radiation, Snowfall and Rainfall) and the number of bikes rented per hour and date information. The target variable is the number of bikes rented.

Attributes information and type of variable:

- Date : year-month-day (date)
- Rented Bike count - Count of bikes rented at each hour (integer)
- Hour - Hour of he day (integer)
- Temperature-Temperature in Celsius (continuous)
- Humidity - % (continuous)
- Windspeed - m/s (continuous)
- Visibility - 10m (continuous)
- Dew point temperature - Celsius (continuous)
- Solar radiation - MJ/m2 (continuous)
- Rainfall - mm (continuous)
- Snowfall - cm (continuous)
- Seasons - Winter, Spring, Summer, Autumn (categorical)
- Holiday - Holiday/No holiday (categorical)
- Functional Day - NoFunc(Non Functional Hours), Fun(Functional hours) (categorical)

1.3 Data Preprocessing

We have preprocessed our data following a similar work flow to previous assignment (we have used the same attributes and applied the same transformations to our data).

We first proceed to clean our data. No NA values or redundant rows were present in the data.

Second, we have to encode the categorical variables so we are able to find the regression coefficients. Variable Date cannot be simply encode. One possible solution would be assigning a day of the week to each date and then proceed to one hot encode the day of the week variable, but this approach will add more variables to our possible model, therefore, for simplicity we will not include this variable in our analysis.

We carried out the process of one hot encoding the variables Seasons, Holiday and Functional day using the pandas function $get\_dummies()$ with the option of $drop\_first = True$, which basically means that one of the categories is dropped and encoded as a non activation of the others (example if we have Spring=0, Summer=0 and Winter=0 this is equivalent to Autumn=1). By using this option we reduce the number of variables added to our data.

We dropped the attributes Dew point temperature due to its correlation with attribute Temp and the attributes Humidity, Wind speed, Visibility, Rainfall, Snowfall, all with a correlation with the response variable below .2 in absolute value. Additionally we will exclude the categorical variable Holiday due to its low correlation (0.07).

To capture the cyclical behaviour of variable Hour we applied a sin-cosine transformation (see this references: kaggle; ianlondon.github.io/blog) We will add two variables named sin_hour and cos_hour as the result of this transformation and we will drop the variable $Hour$ from our data set.

We have split our data into 80/20 proportion of training/testing set. We normalized the training set and carry out this same transformation to the testing set. The normalization process is only applied to the continuous attributes since the normalization of the categorical variables would totally change the meaning of these variables.

1.4 Backpropagation Algorithm description

At the beginning of each epoch we randomly split our training data into two sets (on a 2 to 1 ratio) for training and validation purposes. Our backpropagation algorithm includes two backward pass algorithms, traditional

backward process and a backward process with the AMSGrad enhancement. On each epoch we train our neural net and we find the current error using the validation set. Our stopping condition is as follows: we check whether or not the error has improved by a defined tolerance and record this result on a count variable. If the error improved on the current epoch we reset the count, if not, then we increase it. If our count reach 50, this means our error has not improved on the past 50 epochs, we stop our process and return the best weights obtained so far. For further details read the commented code.

1.5 Tuning Process

For our tuning process, we generated random normal(0,1) initial weights and we ran multiple combinations of activation functions, learning rates, iterations and sizes of hidden layers.

We started by exploring possible values of learning rates between 1e-7 and 0.01 for a fixed amount of 200 iterations and hidden layers sizes of [4,2]. We track the historical MSE over the numbers of iterations (see logfile pages 1-6). We can appreciate that as the learning rate grows AMSGrad remains stable, while the vanilla algorithm exhibits an exploding tendency for greater learning rates.

The final MSE for each one of the learning rates and activation functions can be found on pages 7, 8 and 9. Once again AMSGrad remained stable where vanilla did not.

Therefore we proceed to extend the exploration of possible initial learning rates for AMSGrad to a wider range [0.005,2]. The graphs of pages 10-15 reflect the behaviour of historical MSE and on pages 16-18 we present the final MSE of this process. We can appreciate the stability of AMSGrad and how it shows improvements on learning rates approximately greater than 0.2. We can appreciate how ReLu is already showing good results for several learning rates.

After this analysis we ran tuning combinations again, this time on more specific ranges as concluded from our past analysis; [1e-7, 5e-6] for Vanilla and [0.2, 1.45] for AMSGrad. The best results of our tuning process are reflected on the tables of pages 19 and 20. We presented the best learning parameters sorted for the least MSE. We can see that AMSGrad outperforms Vanilla with multiple learning rates in a close number of iterations. ReLu is the best activation function for AMSGrad while sigmoid and tanh have close

results for the vanilla algorithm. We consider ReLu does not seems to work for our vanilla process since this algorithms tends to explode with our data, a possible improvement for future works would be adding a regularization term. Therefore we can conclude that ReLu would be the best activation function if we can avoid the exploding gradient with algorithms such as AMSGrad.

Up next we will modify the topology of our neural network to check for trade-off between complexity and accuracy.

We trained different neural nets with hidden layers of sizes: [4], [6], [8], [2, 2], [4,2], [4, 4], [6,2] ,[6, 4]. On page 21 we present a final summary of the best results on our training process. We can clearly see how AMSGrad outperforms the vanilla process with different layer sizes, learning rates and specially for the ReLu activation function.

1.6 Training and Testing set evaluation

With the best parameters from the tuning process we can proceed to stablish comparisons in both training and testing sets.

The training set results were presented on the table of page 21. The best coefficient of determination value is around 0.70 and 0.75 for AMSGrad and was accomplished with the ReLu activation function. We can notice as the sizes of our hidden layers increase , with the same learning rate, our accuracy increases as expected. We are satisfied with this result since we have obtained great approximations, improving the results from assignment 1. Also we are satisfied of the findings thrown by a broad tuning process.

The predictions metrics results for the testing set are reflected on the table of page 22. We can see that AMSGrad generalized the results better than Vanilla.

2 Conclusions

To close up our investigation, we present the following conclusions:

- We have improved the results we obtained on assignment 1.
- We carried out a comprehensive tuning process that ended with good solutions for each algorithm.
- We have compare the use of different activation functions combined with different net's topologies, learning rates and iterations.

- We have demonstrated the effects of improving the vanilla gradient descent by displaying the efficiency and stability of AMSGrad.
- We have implemented a versatile and powerful neural net, able to be customized with desired activation functions, number of hidden layers and number of neurons.
- In general we are satisfied with the results of our research and implementation.

3 Appendix

See logfile pages 23-24 for graphical representations of the two variables more correlated with the target variable (Temperature and Hour). We can see how the nature of these variables is captured by the predictions of our model and how AMSGrad successfully predicts the testing set with high accuracy.