CSC 180-01
Mini-Project 1
Friday September 27 2019

Yelp Business Rating Prediction Using Tensorflow

By
Warren Quattrocchi (Team Leader) ID# 219714665
Janus Kwan ID# 212910387
Andrew Wright ID# 218771424

(1) Problem Statement

In this project, the aim is to predict a business's star rating using all the reviews of that business based on a neural network implementation in Tensorflow. This task is considered as a regression problem, wherein the end result will be a star rating that will be compared to the true star rating provided in the dataset. To calculate the accuracy of the results, the RMSE (root mean square error) is reported and the regression lift chart of the best neural network model we obtained is plotted. Then 3-5 businesses from the test dataset (from different categories if possible) are chosen, and the true star ratings of those businesses alongside the predicted ratings output by the best model are displayed.

(2) Methodology

For the first project, a yelp database was provided, and used to train a model to predict as accurately as possible the ratings of a business based on the reviews. The Yelp database contains information of businesses from two different countries with reviews for them all. The project requires the team to “trim” the data. Remove all unnecessary components we deemed useless to predict the star ratings, information such as categories, date, address. Then further normalize the data, changing text values to values between 0 and 1. Then split the data into two portions one for testing and one for training. Only the review text, review category remained after the data trimming, which was then fed into the model for training and test data. Businesses with less than twenty reviews to provide a more accurate dataset for the model. After all the trimming and cleaning,the dataset was left with forty seven thousand businesses with over 20 reviews ready to train and test.

The second part of this project requires the use of provided models and algorithms to train the model. Given between three different training types, Sigmoid, Tanh, and Relu. Two different optimizers Adam and SGD were given. The parameters for all these types are freely changed so to find the best results. To get the RSME as close to zero as possible by experimenting with different parameters such as batch size, neurons and neuron layers. The best attempt was with sigmoid with an ending RSME score of 0.257.

(3) Experimental Results and Analysis

The following is our experimental data, and our results with each change to the parameters. All experiments performed with Adam and Sigmoid.

By Janus

Batch Size	Seconds between Epoch	Changes
1	80	Results continue to go down. Time increased again.
2	37	Results significantly worse than 64, time increased.
16	6-8	Slight improvement to 32 however time doubles again.
32	3-4	Extremely similar results to 64 but time between was doubled.
64	2	Best results with rsme at 0.25726965069770813.
128	1	Compared to 64 change in results is low but time is cut in half.
n_h =	46	
Layer/Neuron	Neurons	Los_val
1 Layer	6	0.0687
1 Layer	12	0.0685
1 Layer	23	0.0664
1 Layer	46	0.0668
1 Layer	92	0.0667
1 Layer	184	0.0692
1 Layer	368	0.0734
	1st layer 2nd Layer	
2 Layers	46 46	0.0666
2 Layers	46 23	0.0661
2 Layers	46 12	0.0662
2 Layers	46 6	0.0656
2 Layers	23 46	0.0665
2 Layers	12 46	0.0661
2 Layers	6 46	0.068
	1st 2nd 3rd	

3 Layers	46	23	6	0.0661
3 Layers	46	46	6	0.0629
3 Layers	6	46	46	0.07
3 Layers	23	46	23	0.0685
Adam Optimizer				
LR	Epochs			
0.0001			300+	large initial changes then very small changes
0.001			26	large changes throughout
0.005			35	large changes throughout
0.0005			92	rapid changes till .07 then small changes

SGD on best model

Achieving a comparable MSE to Adam required a much higher learning rate (0.001 vs .05) Nesterov provided a significant improvement on the sgd best model with lr = .05 and momentum = .9 (.0710 v .1164), lowering the momentum resulted in a slightly worse MSE, all tested with batch size of 64 - running best model with batch size of 32 and 128 both resulted in worse MSE ~ .01 increase, no modifications to SGD could match the best result reported by using Adam

By Warren

Learning rate	Nesterov	Momentum	Decay	MSE
0.01	False	Default	Default	0.0892
0.01	True	Default	Default	0.0881
0.001	False	Default	Default	0.0906
0.001	True	Default	Default	0.0913
0.02	True	Default	Default	0.0879
0.01	True	0.9	Default	0.0856
0.05	True	0.9	Default	0.0710
0.05	False	0.9	Default	0.1164
0.05	True	0.9	1e-6	0.0766
0.05	True	0.9	1e-3	0.0856
0.05	True	0.9	1e-8	0.0856
0.06	True	0.9	Default	0.0711

Results using Activation = tanh, Optimizer = Adam:

By Andrew

alpha	n_h	batch	epo	layer	lr	beta 1	beta2	min_ delta	patie nce	MSE
2		64	400	1 (n_h/8)	0.001	0.9	0.999	1e-5	5	0.0679
2		32	400	1	0.001	0.9	0.999	1e-5	5	0.0671
2		16	400	1	0.001	0.9	0.999	1e-5	5	0.0677
4		32	400	1	0.001	0.9	0.999	1e-5	5	0.0667
8		32	400	1	0.001	0.9	0.999	1e-5	5	0.0675
4		32	400	1	0.01	0.9	0.999	1e-5	5	0.0744
4		32	400	1	0.0001	0.9	0.999	1e-5	5	0.0691
4		32	400	1	0.001	0.09	0.999	1e-5	5	0.0671
4		32	400	1	0.001	0.5	0.999	1e-5	5	0.0670
4		32	400	1	0.001	0.5	0.7	1e-5	5	0.6288
4		32	400	1	0.001	0.5	0.999	1e-5	5	0.0667
4		32	400	1	0.001	0.5	0.999	1e-3	5	0.0668
4		32	400	1	0.001	0.5	0.999	1e-7	5	0.0677

Results using Activation = relu, Optimizer = Adam:

By Andrew

alpha	n_h	batch	epo	layer	lr	beta 1	beta2	min_ delta	patie nce	MSE
2		64	400	1 (n_h/8)	0.001	0.9	0.999	1e-5	5	0.0729
2		32	400	1	0.001	0.9	0.999	1e-5	5	0.0711
2		16	400	1	0.001	0.9	0.999	1e-5	5	0.0712
4		32	400	1	0.001	0.9	0.999	1e-5	5	0.0747
8		32	400	1	0.001	0.9	0.999	1e-5	5	0.0799
2		32	400	1	0.01	0.9	0.999	1e-5	5	0.0731
2		32	400	1	0.0001	0.9	0.999	1e-5	5	0.0714
2		32	400	1	0.0001	0.09	0.999	1e-5	5	0.6288
2		32	400	1	0.0001	0.5	0.999	1e-5	5	0.0705
2		32	400	1	0.0001	0.5	0.7	1e-5	5	0.6289
2		32	400	1	0.0001	0.5	0.999	1e-5	5	0.0713
2		32	400	1	0.0001	0.5	0.999	1e-3	5	0.0713
2		32	400	1	0.0001	0.5	0.999	1e-7	5	0.0721

(4) Task Division and Project Reflection

The first week we worked together on understanding the problem and creating a simple version of the project with one model and minimal data cleaning to make sure we had it working. After creating the basic model we met up after class to make sure everyone was up to speed and distributed out models (Relu, Tanh, sigmoid) for each of us to tune parameters and neuron counts to see which model we could achieve the best performance with. We communicated our results via discord and what parameters/values led to that result. There were portions of the project that each individual spent more time on such as Janus researching and testing many different neuron counts, Andrew testing different optimizer parameters, and Warren working on the initial model implementation.

Some of the challenges we encountered were finding time to discuss results and get everyone on the same level in terms of who was going to do what. We were able to solve these problems by communicating on Discord and meeting directly after class. One thing we learned from this project was how important data pruning was in getting a more accurate result. While the parameter tuning brought small changes, removing businesses with < 20 reviews caused changes of ~0.2 RMSE. We also realized that our results were improved by removing the upper limit on word frequency in the review aggregate. When we eliminated the max_df in our TF-IDF vector, our RMSE improved by ~0.16 RMSE (0.31 vs 0.47).