CS230

Recommendation System

Renner Lucena
Department of Computer Science
Stanford University
renner@stanford.edu

Abstract

Recommendation systems are a classic problem of AI. However, the specific challenges that show up while creating them, looking at a specific industry, company, and the data available, make it really interesting to build one. In this class, I did it for a startup that sells tickets online. The idea was to create a model capable of predicting if a event will be successful (or not), so that it would be recommended to the users if so. The results were good, given the data available, a little bit more of 70% of accuracy.

1 Introduction

Recommendation systems are important given the variety of items, in this case, events, offered to clients, becoming hard for them to choose ones that will increase their satisfaction. I thought that it would be a good idea to build one, also given that I wanted to create something that would be actually useful for someone (in this case, for the startup that provided the data).

Since the data of users was limited and restricted, even given privacy issues, the model now basically just uses the data of events to do broad recommendations. The features of these taken in consideration were month, day-of-month, day-of-week, fractional-hours (time), latitude, longitude, category, average-price (given differences in price) and ticket-amount (number of tickets released for sale), serving as input. The output is either 1 or 0, if the event will be successful or not, based on a complex and strategic query that measures the overall satisfaction of users and also how beneficial the event is for the company.

It predicts, with a little more of 70% of accuracy, if a certain event will be successful or not, a label that was given to me by the company based on a complex and strategic query that measures the satisfaction of users and also how advantageous was the event for the company. The input for it were features of the event.

2 Related work

I searched online about heuristics to build my model, and also tried to find works related to it. However, I could not find any specific paper related to it. Recommendation systems are usually built when the data of users is also available, so that the recommendation is personalized. Some papers that discuss about it and that served as an initial inspiration for me are - Item-based Collaborative Filtering Recommendation by Badrul Sarwar, et. al. Algorithms and A Comparative Study of Collaborative Filtering Algorithms by Joonseok Lee et. al., which discuss different approaches to this problem, as the nearest-neighbor technique, which generate recommendations for a certain user based on its similarity to others, and the model of user ratings. These also discuss challenges of building such systems. One that has some similarity to what I did, but that I ended up not getting anything

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

specific from was A Predictor for Movie Success by Jeffrey Ericson and Jesse Grodman. Some of the discussions there have so resemblance to what I did, but for movies.

3 Dataset and Features

In total, 3613 samples of events were provided with labels from the company's database, 3300 for training and 313 for testing. I did not do any preprocessing. What I had a lot of trouble doing though was getting the right queries to extract the features from the database, and also organizing the rows with the right labels.

One example, a column of X-train, is [4. 26. 6. 18. -3.05676 -60.08308 1. 80. 200.], being the Y-train for it, the label, 0.

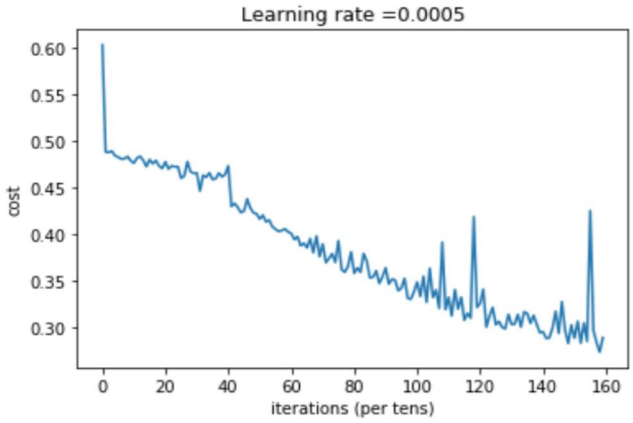
4 Methods

I used a NN of 10-layers with back-prop to train it. Each layer had a 'W' and a 'b' matrices associated to it, with the proper dimensions, so that after each multiplication by the result from the previous layer (being it the raw data for the first one), the RELU activation function is used, but in the last layer. In this one, sigmoid is used, providing the output desired, between 0 and 1, which is so rounded. Since it is a binary classification problem, the cross-entropy cost was used.

$J = -\frac{1}{m}\sum_{i=1}^{m}(y^{(i)}loga^{[2](i)} + (1-y^{(i)})log(1-a^{[2](i)}))$

5 Experiments/Results/Discussion

Searching online, and talking to advisors, I could not find an architecture that was specifically good for my task. Also, there were not heuristics to help me with the decision of an architecture. Starting with logistic regressions, I saw that a larger number of layers helped me to get a better accuracy. I iterated a lot, till I got a NN with 10 layers,, with dimensions 45, 40, 35, ... 5, (multiples of 5, starting from the number of features, 9), all RELU, followed by sigmoid in the end. More layers than that were generating a overfitting of the data, while less, underfiting, in both cases, hurting the performance of the model. The decisions about the learning rate and mini-batch size also came by an iterative process. The final choices were the ones that worked the best.




```
Parameters have been trained!
Train Accuracy: 0.85848486
Test Accuracy: 0.7060703
```

6 Conclusion/Future Work

I think that the project was nice, and I do think that I learned a lot, and that it was pretty rewarding to see things working. For sure, I wanted better results, but 70% is already pretty good. I was not even

sure that it was achievable, given the limited amount of data available, and the uncertainty around how the features and the label given were correlated. I do believe that my architecture was very solid in the end because of all the iterations that I made.

If I had more time, I would try to get more data, test some more complex architectures for the NN, see if I could find any heuristics about the shape of the layers and my (hyper)parameters, and possibly add more features extracted from the data available. Hopefully this would make it have an even better performance.

References

[1] Item-based collaborative filtering recommendation algorithms (2001), pp. 285-295 by Badrul Sarwar, George Karypis, Joseph Konstan, John Riedl

[2] A Comparative Study of Collaborative Filtering Algorithms (2012), by Joonseok Lee, Mingxuan Sun, Guy Lebanon

[3] A Predictor for Movie Success (2013), by Jeffrey Ericson and Jesse Grodman