Assignment # 01

Part 1: d2l.ai Exercise

Question 1: Which parts of code that you are currently writing could be "learned", i.e., improved by learning and automatically determining design choices that are made in your code? Does your code include heuristic design choices?

Answer: I am working on a Fraud Analytics project for Credit Card of a particular bank. I have been doing some feature engineering for the past 7 days, based solely on the domain knowledge that I have. Its particularly challenging as there are only <0.1% fraud instances and it can be assumed that all of these customers have committed frauds in a different way, so manual feature engineering is resulting in more useless features than the useful ones. I believe this task could easily be done using deep neural network with multiple layers, and that too with some good fraud identifications as well.

Yes, I believe my code includes heuristic design choices, I've been trying one idea after another, taking note of what works and what doesn't, modifying old features slightly. For instance, I take change in mean of monthly transactions in past 6 months. If this doesn't work (or isn't much helpful), I try standard deviation for the same or try to change number of months to see how past transactions can be useful in fraud identification.

Question 2: Which problems that you encounter have many examples for how to solve them, yet no specific way to automate them? These may be prime candidates for using deep learning.

Answer: The city traffic. There is a serious issue of traffic jam in my city, happens every 3 out of 7 days. There are 4-5 instances each year where people get back home from work after being stuck in traffic for entire night. This can be solved using deep learning, because there can't be a simple rule-based solution to this problem with 100% success rate and there are multiple ways to deal with this issue.

One possible deep learning solution that I can think of is this: A central system can get live video feed and change traffic signals accordingly all over the city to avoid such traffic jams. Not every vehicle can have a google map but everyone can see the traffic lights. This solution might also require 2-3 more lights to suggest preferred directions to avoid traffic (a common message for everyone on a particular street).

Question 3: Viewing the development of artificial intelligence as a new industrial revolution, what is the relationship between algorithms and data? Is it similar to steam engines and coal (what is the fundamental difference)?

Answer: Data drives the Data Science, as stated in this book we are studying. And algorithms are the means through which we extract meaningful information from the data on which we can do the data science(analytics). Depending on the format of data(audio/video/text/images/time-series), we have various classes of algorithms suitable for that particular data type. This is how data and algorithms are related. Also, as the data size/variety increases due to technical advancements we keep developing more advanced algorithms that enables us to train a better model.

As compared to the "steam engines and coal", the most fundamental way in which "algorithms and data" varies is that data keeps improving in its variety and size over time and in response to that, we keep developing better algorithms. But in case of coal and steam engines, coal quality remained fixed and so did steam engines, after a certain point.

Question 4: Where else can you apply the end-to-end training approach? Physics? Engineering? Econometrics?

Answer: An end-to-end training approach can be used to create an artificial news anchor. This, in fact, has been done by china before. A news anchor that can just see the news in text format and read it with limited facial expressions. Over time, maybe reading news with more accurate emotional display can be done.

Part 2: Summary of Chapter 01 – Introduction

The Introduction of this chapter begins with a perfect example explaining where we don't need ML: "when you are able to devise solutions that work 100% of the time, you should not be using machine learning". And then a motivating example where we do need ML: a "wake word recognizer". This task is clearly not something easy to code even for a highly skilled professional because the rules are not so clear as in how we ourselves do this. And hence we must make the machine 'learn'(aka train a model) how to do this using data and a learning algorithm.

The various types of machine learning explained through practical use cases was an informative way to introduce several diverse ML/DL use cases like 'Recommender Systems', 'Classifications', 'Tagging', 'Speech Recognition', 'Clustering' just to mention a few. The examples really gave me a bird's eye view of what is the scope of this knowledge and how ML plays an integral role in our daily lives, like our Netflix recommendations, Google Search results, Siri etc. I have used some of these use cases in my professional journey and I yet found this chapter highly informative.

The history of how it all started with Statistical distributions leading to contributions of Fisher like LDA and Fisher Information Matrix, then the Information Theory and Turing Test and further the Hebbian Learning Rule has given me a good understanding of how this field really came into existence. I must mention here that I really like this field and now that I know its history of how it came to be, I'm at a loss for words. I wasn't aware of the limitations of recommendation system like censored feedback and feedback loops. I found out that this is an active research topic as well. The same for conversational AI, a use case of seq2seq algorithm. For conversations, its important to know the prior state of conversation along long temporal distances and not just the last input heard.

I found about 'hierarchical classifications' and that sometimes in a classification task, now all errors are equal and if there is an error it better be a misclassified class closely related to the ground truth. This is a situation I haven't come across yet. The 'death cap' example was also a unique way to make the point that "most likely class is not necessarily the one that you are going to use for your decision". We also need to consider the cost of making that prediction, a Risk Assessment.

I have never used Reinforcement learning but its introduction to the concept of "how interaction between Agent and Environment results in a Policy for taking Actions that maximises Rewards" was really amazing. I have seen the AlphaGo documentary and have also read Asimov (my favourite writer btw). After reading the intro to reinforcement learning, I'm very excited to learn how to train my own models that can play games and interact with environment.

The historical comparison of data sizes and processing power is something I have seen previously since I use GPUs myself to train models faster but the fact that non-linear activation functions used in deep neural nets can increase computational efficiency and decrease memory utilization was new and interesting to me.

I have already completed the Neural Network Course by Andrew NG as recommended in the study plan by Elvis. It's a very informative course and gives a mathematical explanation of how neural networks work, how the backpropagation minimises the error/loss, how parameters and hyperparameters can result in a wide variety of deep NN models, and how activation functions and hidden layers can result in new non-linear features and hence better prediction. Fortunately, I have a solid mathematical background so this was a great course for me to start learning the fundamentals of Deep Learning. I'm looking forward to learning all the use cases that was mentioned in this introduction and a lot more.

Submitted by:

Name: Shashank Prakash

e-Mail: shashank708.2606@gmail.com

Date of Submission: 08th August 2020