DiDL #1

Monika Obrocka

07/08/2020

1 Exercise 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?

The code I'm currently writing uses tree based logic to determine if a user has a low, moderate or high risk of being Covid-19 positive. It takes into account certain symptoms, exposure and demographic information about a person. Since this is a novel virus, many facts about it are still unknown and hard-coded heuristics have to be adapted constantly and will certainly classify many users incorrectly.

2 Exercise 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.

I would look for problems that need decision rules and have adaptive criteria to accomplish something. Problems that are repetitive and have well defined criteria are prime candidates for automation where a set of heuristics would be enough to accomplish the goal. One prime problem for deep learning might be my grocery shopping. I've used apps in the past that track expiration dates and send push notifications, but I would like something that will adapt my shopping list to what I've already have in the pantry and what I might fancy to eat in the coming week.

3 Exercise 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)?

The 1st industrial revolution, started by steam engines, essentially mechanised animal power. We were not limited to the power of a single animal. The 3rd industrial revolution digitised the world around us giving us tons of data available to computers. In that sense we can say that data is the fuel/coal of the 4th industrial revolution, but the steam engine in this analogy would be the model and all valves to regulate the pressure would be the algorithm.

4 Exercise 4

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

The end-to-end learning means that we encompass a complex system in a single model skipping some steps, like feature engineering. Since deep learning can be used to address problems in many domains the end-to-end approach can be applied there as well.

5 Chapter 1 Summary

The introductory chapter gives context for deep learning. It introduces common vocabulary and concepts to build the foundation for the following chapters. The first advice that authors give to the reader is how to assess if a task can and should be solved with machine learning. Basically if

you can solve a task working from first principles, follow a recipe, and if it works 100% of the time you will be better off leaving machine learning for another occasion. If the solution to our task changes over time and our method to solve it has to adapt accordingly, machine learning will be a good fit as it can learn from experience.

In machine learning you don't have to specify exactly what steps have to be taken to accomplish a task. What we have to define precisely is the input and output to the system and family of models we will use. Then we start the learning process to discover the best settings for the model. In traditional machine learning to enable the process of learning from data, one would have to go through a labour intensive process of transforming the data, standardising it, and feature engineering. In deep learning we can skip those steps because deep learning neural networks learn many layers of computations transforming the data and designing features with no need for domain knowledge.

Machine learning pipelines are modular and can be broken into data, model, loss function and an algorithm. Data is a collection of examples with attributes called features. Type of data depends on the domain you are working in. It can be a picture, sound sample or a sentence and it has to be transformed into a numerical form. A model in the pipeline is estimated/learned from data. This is why it is important to get the data without mistakes to avoid the *garbage-in-garbage-out* scenario. We need a mechanism that will tell us if the model is improving with experience or not. This mechanism is the loss function or objective function. If this function gets smaller it means that our model is performing well. We use the loss function to see if our model is generalising well or is over-fitting on the data we provided. To check we check the value of loss function on training and testing data. The last module in the pipeline is an algorithm that will look for the best parameters to minimise the loss function.

What kind of machine learning technique we apply to a task depends on a task itself. If we want to predict a target/label based on the input we would use supervised learning. We call it supervised because we choose the parameters, like a dataset with examples and corresponding labels. In supervised learning we encounter regression tasks where a target is a value in some finite range that we have to approximate as closely as possible. Another task is classification where we assign a category from a discrete set of possibilities to an example. Often an example can be assigned more than one label. For example, a picture can show both a cat and a dog in a task to classify a picture of either animal. We call those problems multi-label classification where we tag each target. In some scenarios we also want to assign a ranking to our classification like in search and ranking problems. A related problem to search and ranking is a recommendation system where this ordered result is personalised. Finally, there is sequence learning where our model remembers the previous example to help it predict the next one.

In unsupervised learning we look for unseen patterns in the data without any predefined labels. Since we know nothing about the data and we expect nothing from it the number of tasks that can be tackled with unsupervised learning is quite large. In that paradigm we also find clustering tasks. If we want to find a small number of parameters that describe our data we call it a subspace estimation problem. If we want to find a relationship between entities, we call it representation learning. Unsupervised learning also gives us capabilities to find how different phenomenons relate to each other by using causality and probabilistic graphical models. We can also synthesise real looking data by using generative adversarial networks.

Both supervised and unsupervised learning can be categorised as offline learning since all the learning happens without interaction with the environment. Our model is trained in isolation, removed from the environment all systems operate in. In another type of learning, which is about action and not prediction, our system interacts with the environment and the actions it takes will impact the environment. Reinforcement learning belongs to that class, where an agent interacts with an environment and observes what happens. Each action leads to some kind of reward and an observation as a feedback.