Name: Bharati Patidar
Email: bpatidar08@gmail.com
Date of Submission: 8 Aug 2020

Exercise Solution 1.8

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?

- Working on an algorithm that could automatically map incoming source data tables with the target data tables. On the lines of a schema auto-mapper. Would like to have a solution that could automatically make the best choice of schema mapping before dumping the records in the data tables.

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.

- Same use case as above. Every possible logic to compare two data columns could be written. Just the exhaustive set of these logics and executing them sequentially does not make sense.

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)?

- Just as coal is the raw material for steam engines, data is the raw material for algos. The major difference in my opinion is that the steam engine is the ultimate goal while the source of energy could vary. In data, algo combo, we choose an appr algo to build for the data. Thus, data is the main goal.

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

- All of them. This could and is been applied in all the fields to harness science’s potential.

Learnings and Takeaways from Chapter 1:

I have some industry experience with deep learning. Am quite aware of the overall concept and types of Machine Learning algorithms. From the introduction chapter, there were few things that got reinforced, while got to know some few concepts that I had not read before. Following are my takeaways from the first chapter.

- Objective Functions:
  Objective functions are measure of how good or bad models are. By convention, objective functions are designed such that lower is better. If a function f is such that higher is better, its effectively converted to a function f' = -f . These functions are also called as loss functions/cost functions. Lowering is the main goal.
- Optimization Algorithms:
  Are algorithms capable of searching for the best possible parameters for minimizing the loss function. Stochastic Gradient Descent is the most preferred optimization algo with neural networks. There are challenges in the basic gradient descent as follows:
    1. Finding an appropriate learning rate
    2. Learning rate schedules need to be defined at the beginning and are thus unable to adapt to dataset characteristics
    3. Learning rate applies to all parameters equally. Thus, those rare features need a larger update but do not get so.
    4. Using SGD, it gets hard to escape the saddle points.

  To overcome these challenges, there are multiple variations such as :
  - ➢ Momentum: Accelerates the SGD in relevant direction and dampens oscillations.
  - ➢ Adagrad: Adapts the learning rate to parameters, updating rarest features with larger updates, while freq occurring features get smaller updates. It is well suited for sparse data.
  - ➢ Adam: Also provides adaptive learning rates for each parameter. In addition to what adadelta, rmsprop do, adam also stores exponentially decaying avg of past gradients, similar to momentum.
  - ➢ AMSGrad: In some settings, Adam converges to sub-optimal solutions. To address Adam's challenges, AMSGrad was brought in that used max of past squared gradients than the exponential avg to update parameters. It results in a non-increasing step size unlike in Adam.

  In general, Adam is the best choice.