11 Conclusion

Wow, that was fast! You are really good if you got here and managed to understand most of the book's material.

If you look at the number at the bottom of this page, you see that I have overspent paper, which means that the title of the book was slightly misleading. I hope that you forgive me for this little marketing trick. After all, if I wanted to make this book exactly a hundred pages, I could reduce font size, white margins, and line spacing, or remove the section on UMAP and leave you on your own with the original paper. Believe me: you would not want to be left on your own with the original paper on UMAP! (Just kidding.)

However, by stopping now, I feel confident that you have got everything you need to become a great modern data analyst or machine learning engineer. That doesn't mean that I covered everything, but what I covered in a hundred+ pages you would find in a bunch of books, each a thousand pages thick. Much of what I covered is not in the books at all: typical machine learning books are conservative and academic, while I emphasized those algorithms and methods that you will find useful in your day to day work.

What exactly I would have covered if it was a thousand-page machine learning book?

11.1 What Wasn't Covered

11.1.1 Topic Modeling

In text analysis, topic modeling is a prevalent unsupervised learning problem. You have a collection of text documents, and you would like to discover topics present in each document. **Latent Dirichlet Allocation** (LDA) is a very effective algorithm of topic discovery. You decide how many topics are present in your collection of documents and the algorithm assigns a topic to each word in this collection. Then, to extract the topics from a document, you simply count how many words of each topic are present in that document.

11.1.2 Gaussian Processes

Gaussian processes (GP) is a supervised learning method that competes with kernel regression. It has some advantages over the latter. For example, it provides confidence intervals for the regression line in each point. I decided not to explain GP because I could not figure out a simple way to explain them, but you definitely could spend some time to learn about GP. It will be time well spent.

11.1.3 Generalized Linear Models

Generalized Linear Model (GLM) is a generalization of the linear regression to modeling various forms of dependency between the input feature vector and the target. Logistic regression, for instance, is one form of GLMs. If you are interested in regression and you look for simple and explainable models, you should definitely read more on GLM.

11.1.4 Probabilistic Graphical Models

I have mentioned one example of **probabilistic graphical models** (PGMs) in Chapter 7: **conditional random fields** (CRF). With CRF you can model the input sequence of words and relationships between the features and labels in this sequence as a sequential *dependency graph*. More generally, a PGM can be any graph. A **graph** is a structure consisting of a collection of nodes and edges that join a pair of nodes. Each node in PGM represents some random variable (values of which can be observed or unobserved), and edges represent the conditional dependence of one random variable on another random variable. For example, the random variable "sidewalk wetness" depends on the random variable "weather condition." By observing values of some random variables, an optimization algorithm can learn from data the dependency between observed and unobserved variables.

PGMs allow the data analyst to see how the values of one feature depend on the values of other features. If the edges of the dependency graph are directed, it becomes possible to infer causality. Unfortunately, constructing such models by hand requires a substantial amount of domain expertise and a strong understanding of probability theory and statistics. The latter is often a problem for many domain experts. Some algorithms can learn the structure of dependency graphs from data, but the learned models are often hard to interpret by a human and thus they aren't beneficial for understanding complex probabilistic processes that generated the data. CRF is by far the most used PGM with applications mostly in text and image processing. However, in these two domains, they were surpassed by neural networks. Another graphical model, **Hidden Markov Model** or HMM, in the past, was frequently used in speech recognition, time series analysis, and other temporal inference tasks, but, again HMM lost to neural networks.

If you still decide to learn more about PGMs, they are also known under names of Bayesian networks, belief networks, and probabilistic independence networks.

11.1.5 Markov Chain Monte Carlo

If you work with graphical models and want to sample examples from a very complex distribution defined by the dependency graph, you could use **Markov Chain Monte Carlo** (MCMC) algorithms. MCMC is a class of algorithms for sampling from any probability distribution defined mathematically. Remember that when we talked about **denoising autoencoders**, we sampled noise from the normal distribution. Sampling from standard

distributions, such as normal or uniform, is relatively easy because their properties are well known. However, the task of sampling becomes significantly more complicated when the probability distribution can have an arbitrary form defined by a complex formula.

11.1.6 Generative Adversarial Networks

Generative adversarial networks, or GANs, are a class of neural networks used in unsupervised learning. They are implemented as a system of two neural networks contesting with each other in a *zero-sum game* setting. The most popular application of GANs is to learn to generate photographs that look authentic to human observers. The first of the two networks takes a random input (typically Gaussian noise) and learns to generate an image as a matrix of pixels. The second network takes as input two images: one "real" image from some collection of images as well as the image generated by the first network. The second network has to learn to recognize which one of the two images was generated by the first network. The first network gets a negative loss if the second network recognizes the "fake" image. The second network, on the other hand, gets penalized if it fails to recognize which one of the two images is fake.

11.1.7 Genetic Algorithms

Genetic algorithms (GA) are a numerical optimization technique used to optimize undifferentiable optimization objective functions. They use concepts from evolutionary biology to search for a global optimum (minimum or maximum) of an optimization problem, by mimicking evolutionary biological processes.

GA work by starting with an initial generation of candidate solutions. If we look for optimal values of the parameters of our model, we first randomly generate multiple combinations of parameter values. We then test each combination of parameter values against the objective function. Imagine each combination of parameter values as a point in a multi-dimensional space. We then generate a subsequent generation of points from the previous generation by applying such concepts as "selection," "crossover," and "mutation."

In a nutshell, that results in each new generation keeping more points similar to those points from the previous generation that performed the best against the objective. In the new generation, the points that performed the worst in the previous generation are replaced by "mutations" and "crossovers" of the points that performed the best. A mutation of a point is obtained by a random distortion of some attributes of the original point. A crossover is a certain combination of several points (for example, an average).

Genetic algorithms allow finding solutions to any measurable optimization criteria. For example, GA can be used to optimize the hyperparameters of a learning algorithm. They are typically much slower than gradient-based optimization techniques.

11.1.8 Reinforcement Learning

As we already discussed, **reinforcement learning** (RL) solves a very specific kind of problem where the decision making is sequential. Usually, there's an agent acting in an unknown environment. Each action brings a reward and moves the agent to another state of the environment (usually, as a result of some random process with unknown properties). The goal of the agent is to optimize its long-term reward.

Reinforcement learning algorithms, such as Q-learning, as well as its neural network based counterparts, are used in learning to play video games, robotic navigation and coordination, inventory and supply chain management, optimization of complex electric power systems (power grids), and learning financial trading strategies.

* * *

The book stops here. Don't forget to occasionally visit the book's companion wiki to stay updated on new developments in each machine learning area considered in the book. As I said in the Preface, this book, thanks to the constantly updated wiki, like a good wine keeps getting better after you buy it.

Oh, and don't forget that the book is distributed on the *read first, buy later* principle. That means that if while reading these words you look at a PDF file and cannot remember having paid to get it, you are probably the right person for buying the book.

11.2 Acknowledgements

The high quality of this book would be impossible without volunteering editors. I especially thank the following readers for their systematic contributions: Martijn van Attekum, Daniel Maraini, Ali Aziz, Rachel Mak, Kelvin Sundli, and John Robinson.

Other wonderful people to whom I am grateful for their help are Michael Anuzis, Knut Sverdrup, Freddy Drennan, Carl W. Handlin, Abhijit Kumar, Lazze Veddbärd, Ricardo Reis, Daniel Gross, Johann Faouzi, Akash Agrawal, Nathanael Weill, Filip Jekic, Abhishek Babuji, Luan Vieira, Sayak Paul, Vaheid Wallets, Lorenzo Buffoni, Eli Friedman, Łukasz Mądry, Haolan Qin, Bibek Behera, Jennifer Cooper, Nishant Tyagi, Denis Akhiyarov, Aron Janarv, Alexander Ovcharenko, Ricardo Rios, Michael Mullen, Matthew Edwards, David Etlin, Manoj Balaji J, David Roy, Luan Vieira, Luiz Felix, Anand Mohan, Hadi Sotudeh, Charlie Newey, Zamir Akimbekov, Jesus Renero, Karan Gadiya, Mustafa Anıl Derbent, JQ Veenstra, Zsolt Kreisz, Ian Kelly, Lukasz Zawada, and Luciano Segura.