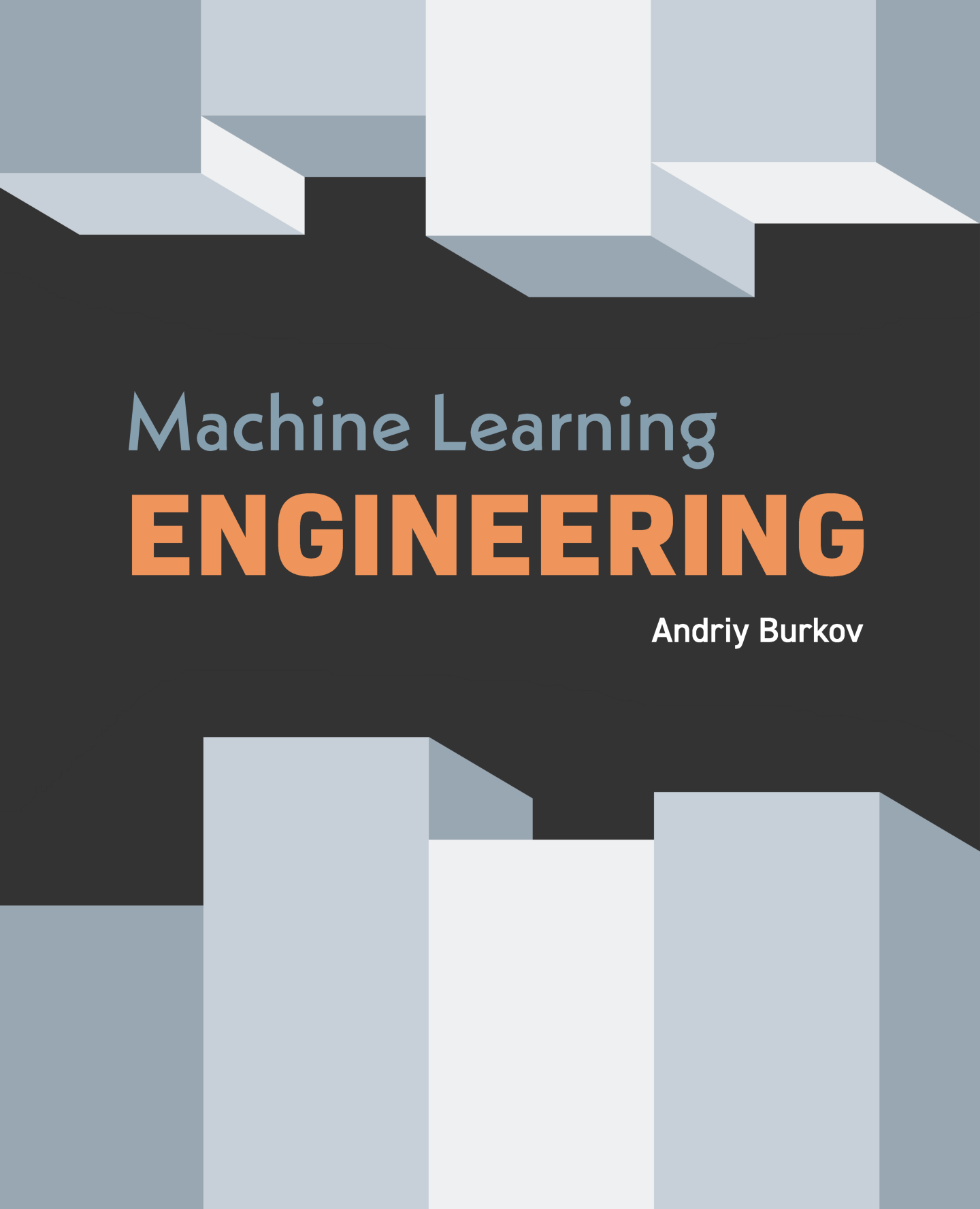
Machine Learning
ENGINEERING
Andriy Burkov

“In theory, there is no difference between theory and practice. But in practice, there is.”

— Benjamin Brewster

“The perfect project plan is possible if one first documents a list of all the unknowns.”

— Bill Langley

“When you’re fundraising, it’s AI. When you’re hiring, it’s ML. When you’re implementing, it’s linear regression. When you’re debugging, it’s printf().”

— Baron Schwartz

The book is distributed on the “read first, buy later” principle.

2 Before the Project Starts

Before a machine learning project starts, it must be prioritized. Prioritization is inevitable: the team and equipment capacity is limited, while the organization's backlog of projects could be very long.

To prioritize a project, one has to estimate its complexity. With machine learning, accurate complexity estimation is rarely possible because of major unknowns, such as whether the required model quality is attainable in practice, how much data is needed, and what, and how many features are necessary.

Furthermore, a machine learning project must have a well-defined goal. Based on the goal of the project, the team could be adequately adjusted and resources provisioned.

In this chapter, we consider these and related activities that must be taken care of before a machine learning project starts.

2.1 Prioritization of Machine Learning Projects

The key considerations in the prioritization of a machine learning project, are impact and cost.

2.1.1 Impact of Machine Learning

The impact of using machine learning in a broader engineering project is high when, 1) machine learning can replace a complex part in your engineering project or 2) there's a great benefit in getting inexpensive (but probably imperfect) predictions.

For example, a complex part of an existing system can be rule-based, with many nested rules and exceptions. Building and maintaining such a system can be extremely difficult, time-consuming, and error-prone. It can also be a source of significant frustration for software engineers when they are asked to maintain that part of the system. Can the rules be learned instead of programming them? Can an existing system be used to generate labeled data easily? If yes, such a machine learning project would have a high impact and low cost.

Inexpensive and imperfect predictions can be valuable, for example, in a system that dispatches a large number of requests. Let's say many such requests are "easy" and can be solved quickly using some existing automation. The remaining requests are considered "difficult" and must be addressed manually.

A machine learning-based system that recognizes "easy" tasks and dispatches them to the automation will save a lot of time for humans who will only concentrate their effort and time on difficult requests. Even if the dispatcher makes an error in prediction, the difficult request will reach the automation, the automation will fail on it, and the human will eventually receive that request. If the human gets an easy request by mistake, there's no problem either: that easy request can still be sent to the automation or processed by the human.

2.1.2 Cost of Machine Learning

Three factors highly influence the cost of a machine learning project:

- the difficulty of the problem,
- the cost of data, and
- the need for accuracy.

Getting the right data in the right amount can be very costly, especially if manual labeling is involved. The need for high accuracy can translate into the requirement of getting more data or training a more complex model, such as a unique **architecture** of a deep neural network or a nontrivial **ensembling** architecture.

When you think about the problem's difficulty, the primary considerations are:

- whether an implemented algorithm or a software library capable of solving the problem is available (if yes, the problem is greatly simplified),
- whether significant computation power is needed to build the model or to run it in the production environment.

The second driver of the cost is data. The following considerations have to be made:

- can data be generated automatically (if yes, the problem is greatly simplified),
- what is the cost of manual **annotation** of the data (i.e., assigning labels to unlabeled examples),
- how many examples are needed (usually, that cannot be known in advance, but can be estimated from known published results or the organization's own experience).

Finally, one of the most influential cost factors is the desired accuracy of the model. The machine learning project's cost grows superlinearly with the accuracy requirement, as illustrated in Figure 1. Low accuracy can also be a source of significant loss when the model is deployed in the production environment. The considerations to make:

- how costly is each wrong prediction, and
- what is the lowest accuracy level below which the model becomes impractical.

2.2 Estimating Complexity of a Machine Learning Project

There is no standard complexity estimation method for a machine learning project, other than by comparison with other projects executed by the organization or reported in the literature.

2.2.1 The Unknowns

There are several major unknowns that are almost impossible to guess with confidence unless you worked on a similar project in the past or read about such a project. The unknowns are:

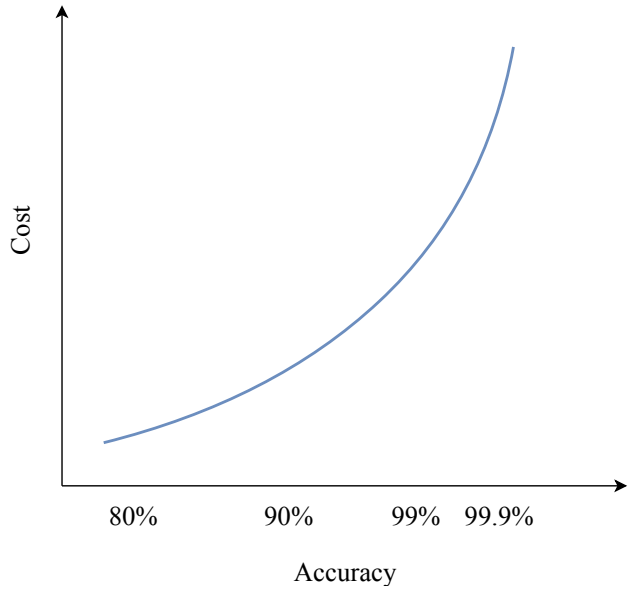

Figure 1: Superlinear growth of the cost as a function of accuracy requirement.

- whether the required quality is attainable in practice,
- how much data you will need to reach the required quality,
- what features and how many features are necessary so that the model can learn and generalize sufficiently,
- how large the model should be (especially relevant for neural networks and ensemble architectures), and
- how long will it take to train one model (in other words, how much time is needed to run one **experiment**) and how many experiments will be required to reach the desired level of performance.

One thing you can almost be sure of: if the required level of model **accuracy** (one of the popular model quality metrics we consider in Section **??** of Chapter 5) is above 99%, you can expect complications related to an insufficient quantity of labeled data. In some problems, even 95% accuracy is considered very hard to reach. (Here we assume, of course, that the data is balanced, that is, there's no **class imbalance**. We will discuss class imbalance in Section **??** of the next chapter.)

Another useful reference is the human performance on the task. This is typically a hard problem if you want your model to perform as well as a human.

2.2.2 Simplifying the Problem

One way to make a more educated guess is to simplify the problem and solve a simpler problem first. For example, assume that the problem is that of classifying a set of documents into 1000 topics. Run a pilot project by focusing on 10 topics first, by considering documents belonging to other 990 topics as "Other."[1] Manually label the data for these 11 classes (10 real topics, plus "Other"). The logic here is that it's much simpler for a human to keep in mind the definitions of only 10 topics compared to memorizing the difference between 1000 topics[2].

Once you have simplified your problem to 11 classes, solve it, and measure time on every stage. Once you see that the problem for 11 classes is solvable, you can reasonably hope that it will be solvable for 1000 classes as well. Your saved measurements can then be used to estimate the time required to solve the full problem, though you cannot simply multiply this time by 100 to get an accurate estimate. The quantity of data needed to learn to distinguish between more classes usually grows superlinearly with the number of classes.

An alternative way of obtaining a simpler problem from a potentially complex one is to split the problem into several simple ones by using the natural slices in the available data. For example, let an organization have customers in multiple locations. If we want to train a model that predicts something about the customers, we can try to solve that problem only for one location, or for customers in a specific age range.

2.2.3 Nonlinear Progress

The progress of a machine learning project is nonlinear. The prediction error usually decreases fast in the beginning, but then the progress gradually slows down.[3] Sometimes you see no progress and decide to add additional features that could potentially depend on external databases or knowledge bases. While you are working on a new feature or labeling more data (or outsourcing this task), no progress in model performance is happening.

Because of this nonlinearity of progress, you should make sure that the product owner (or the client) understands the constraints and risks. Carefully log every activity and track the time it took. This will help not only in reporting, but also in the estimation of the complexity of similar projects in the future.

[1] Putting examples belonging to 990 classes in one class will likely create a highly imbalanced dataset. If it's the case, you would prefer to **undersample** the data in the class "Other." We consider data undersampling in Section **??** of the next chapter.

[2] To save even more time, apply clustering to the whole collection of unlabeled documents and only manually label documents belonging to one or a few clusters.

[3] The 80/20 rule of thumb often applies: 80% of progress is made using the first 20% of resources.

2.3 Defining the Goal of a Machine Learning Project

The **goal** of a machine learning project is to build a model that solves, or helps solve, a business problem. Within a project, the model is often seen as a black box described by the structure of its input (or inputs) and output (or outputs), and the minimum acceptable level of performance (as measured by accuracy of prediction or another **performance metric**).

2.3.1 What a Model Can Do

The model is typically used as a part of a system that serves some purpose. In particular, the model can be used within a broader system to:

- automate (for example, by taking action on the user's behalf or by starting or stopping a specific activity on a server),
- alert or prompt (for example, by asking the user if an action should be taken or by asking a system administrator if the traffic seems suspicious),
- organize, by presenting a set of items in an order that might be useful for a user (for example, by sorting pictures or documents in the order of similarity to a query or according to the user's preferences),
- annotate (for instance, by adding contextual annotations to displayed information, or by highlighting, in a text, phrases relevant to the user's task),
- extract (for example, by detecting smaller pieces of relevant information in a larger input, such as named entities in the text: proper names, companies, or locations),
- recommend (for example, by detecting and showing to a user highly relevant items in a large collection based on item's content or user's reaction to the past recommendations),
- classify (for example, by dispatching input examples into one, or several, of a predefined set of distinctly-named groups),
- quantify (for example, by assigning a number, such as a price, to an object, such as a house),
- synthesize (for example, by generating new text, image, sound, or another object similar to the objects in a collection),
- answer an explicit question (for example, "Does this text describe that image?" or "Are these two images similar?"),
- transform its input (for example, by reducing its dimensionality for visualization purposes, paraphrasing a long text as a short abstract, translating a sentence into another language, or augmenting an image by applying a filter to it),
- detect a novelty or an anomaly.

Almost any business problem solvable with machine learning can be defined in a form similar to one from the above list. If you cannot define your business problem in such a form, likely, machine learning is not the best solution in your case.

2.3.2 Properties of a Successful Model

A successful model has the following four properties:

- it respects the input and output specifications and the performance requirement,
- it benefits the organization (measured via cost reduction, increased sales or profit),
- it helps the user (measured via productivity, engagement, and sentiment),
- it is scientifically rigorous.

A scientifically rigorous model is characterized by a predictable behavior (for the input examples that are similar to the examples that were used for training) and is reproducible. The former property (predictability) means that if input feature vectors come from the same distribution of values as the training data, then the model, on average, has to make the same percentage of errors as observed on the holdout data when the model was trained. The latter property (reproducibility) means that a model with similar properties can be easily built once again from the same training data using the same algorithm and values of hyperparameters. The word "easily" means that no additional analysis, labeling, or coding is necessary to rebuild the model, only the compute power.

When defining the goal of machine learning, make sure you solve the right problem. To give an example of an incorrectly defined goal, imagine your client has a cat and a dog and needs a system that lets their cat in the house but keeps their dog out. You might decide to train the model to distinguish cats from dogs. However, this model will also let *any cat* in and not just *their* cat. Alternatively, you may decide that because the client only has two animals, you will train a model that distinguishes between those two. In this case, because your classification model is binary, a raccoon will be classified as either the dog or the cat. If it's classified as the cat, it will be let in the house.[4]

2.4 Structuring a Machine Learning Team

There are two cultures of structuring a machine learning team, depending on the organization.

2.4.1 Two Cultures

One culture says that a machine learning team has to be composed of data analysts who collaborate closely with software engineers. In such a culture, a software engineer doesn't need to have deep expertise in machine learning, but has to understand the vocabulary of their fellow data analysts.

According to other culture, all engineers in a machine learning team must have a combination of machine learning and software engineering skills.

[4]This is why having the class "Other" your classification problems is almost always a good idea.

There are pros and cons in each culture. The proponents of the former say that each team member must be the best in what they do. A data analyst must be an expert in many machine learning techniques and have a deep understanding of the theory to come up with an effective solution to most problems, fast and with minimal effort. Similarly, a software engineer must have a deep understanding of various computing frameworks and be capable of writing efficient and maintainable code.

The proponents of the latter say that scientists are hard to integrate with software engineering teams. Scientists care more about how accurate their solution is and often come up with solutions that are impractical and cannot be effectively executed in the production environment. Also, because scientists don't usually write efficient, well-structured code, the latter has to be rewritten into production code by a software engineer; depending on the project, that can turn out to be a daunting task.

2.4.2 Members of a Machine Learning Team

Besides machine learning and software engineering skills, a machine learning team may include experts in data engineering (also known as data engineers) and experts in data labeling.

Data engineers are software engineers responsible for ETL (for Extract, Transform, Load). These three conceptual steps are part of a typical data pipeline. Data engineers use ETL techniques and create an automated pipeline, in which raw data is transformed into analysis-ready data. Data engineers design how to structure the data and how to integrate it from various resources. They write on-demand queries on that data, or wrap the most frequent queries into fast application programming interfaces (APIs) to make sure that the data is easily accessible by analysts and other data consumers. Typically, data engineers are not expected to know any machine learning.

In most big companies, data engineers work separately from machine learning engineers in a data engineering team.

Experts in data labeling are responsible for four activities:

- manually or semi-automatically assign labels to unlabeled examples according to the specification provided by data analysts,
- build labeling tools,
- manage outsourced labelers, and
- validate labeled examples for quality.

A **labeler** is person responsible for assigning labels to unlabeled examples. Again, in big companies, data labeling experts may be organized in two or three different teams: one or two teams of labelers (for example, one local and one outsourced) and a team of software engineers, plus a user experience (UX) specialist, responsible for building labeling tools.

When possible, invite domain experts to work closely with scientists and engineers. Employ domain experts in your decision making about the inputs, outputs, and features of your

model. Ask them what they think your model should predict. Just the fact that the data you can get access to can allow you to predict some quantity doesn't mean the model will be useful for the business.

Discuss with the domain experts what they look for in the data to make a specific business decision; that will help you with feature engineering. Discuss also what clients pay for and what is a deal-breaker for them; that will help you to translate a business problem into a machine learning problem.

Finally, there are DevOps engineers. They work closely with machine learning engineers to automate model deployment, loading, monitoring, and occasional or regular model maintenance. In smaller companies and startups, a DevOps engineer may be part of the machine learning team, or a machine learning engineer could be responsible for the DevOps activities. In big companies, DevOps engineers employed in machine learning projects usually work in a larger DevOps team. Some companies introduced the MLOps role, whose responsibility is to deploy machine learning models in production, upgrade those models, and build data processing pipelines involving machine learning models.

2.5 Summary

Before a machine learning project starts, it must be prioritized, and the team working on the project must be built. The key considerations in the prioritization of a machine learning project, are impact and cost.

The impact of using machine learning is high when, 1) machine learning can replace a complex part in your engineering project, or 2) there's a great benefit in getting inexpensive (but probably imperfect) predictions.

The cost of the machine learning project is highly influenced by three factors: 1) the difficulty of the problem, 2) the cost of data, and 3) the needed model performance quality.

There is no standard method of estimation of how complex a machine learning project is other than by comparison with other projects executed by the organization or reported in the literature. There are several major unknowns that are almost impossible to guess: whether the required level of model performance is attainable in practice, how much data you will need to reach that level of performance, what features and how many features are needed, how large the model should be, and how long will it take to run one experiment and how many experiments will be needed to reach the desired level of performance.

One way to make a more educated guess is to simplify the problem and solve a simpler one.

The progress of a machine learning project is nonlinear. The error usually decreases fast in the beginning, but then the progress slows down. Because of this nonlinearity of progress, it's better to make sure that the client understands the constraints and the risks. Carefully log every activity and track the time it took. This will help not only in reporting but also in estimating complexity for similar projects in the future.

The goal of a machine learning project is to build a model that solves some business problem. In particular, the model can be used within a broader system to automate, alert or prompt, organize, annotate, extract, recommend, classify, quantify, synthesize, answer an explicit question, transform its input, and detect novelty or an anomaly. If you cannot frame the goal of machine learning in one of these forms, likely, machine learning is not the best solution.

A successful model 1) respects the input and output specifications and the minimum performance requirement, 2) benefits the organization and the user, and 3) is scientifically rigorous.

There are two cultures of structuring a machine learning team, depending on the organization. One culture says that a machine learning team has to be composed of data analysts who collaborate closely with software engineers. In such a culture, a software engineer doesn't need to have profound expertise in machine learning but has to understand the vocabulary of their fellow data analysts or scientists. According to the other culture, all engineers in a machine learning team must have a combination of machine learning and software engineering skills.

Besides having machine learning and software engineering skills, a machine learning team may include experts in data labeling and data engineering experts. DevOps engineers work closely with machine learning engineers to automate model deployment, loading, monitoring, and occasional or regular model maintenance.