Graph Powered Machine Learning

Alessandro Negro

MEAP
Dear Reader,

Thanks for purchasing the MEAP of *Graph-Powered Machine Learning*. Graph-based machine learning is becoming a very important trend in Artificial Intelligence, transcending a lot of other techniques. Google, Facebook, and E-bay – to cite some of them - have multiple projects involving graphs, and more specifically graph models and graph algorithms, as empowering mechanism behind the most advanced services they are providing to their end users.

*Graph-Powered Machine Learning* is a practical guide to effectively using graphs in machine learning applications, driving you in all the stages necessary for building complete solutions where graphs play a key role. It focuses on methods, algorithms, and design patterns related to graphs. Based on my personal experience on building complex machine learning applications, this book suggests many recipes in which graphs are the main ingredient to prepare a tasty product to your customers. Across the lifecycle of a machine learning project such approaches can be useful under several aspects: managing data sources more efficiently, implement better algorithms, storing the prediction model so that they can be accessed faster, and visualizing the results in a more effective way for further analysis.

The book is divided into three parts. The first part is introductory to the topic. The three chapters introduce main graph and machine learning concepts from the basics. Furthermore, the role of graphs in Big Data Platforms and Machine learning is highlighted and presented using multiple scenarios.

The second part is the core of the book. Several techniques are described, from data source modeling, to algorithm design both leveraging graphs as underlying technology. A lot of optimization approaches, best practices, and common pitfalls are detailed to help data scientists or data engineers define the infrastructure and choose the right approaches since the beginning of their projects.

The last part presents three different applications. Here concrete end-to-end projects are discussed and for each of them the architecture, design best practice, and common pitfalls will be illustrated.

I hope that what you'll get access to will be of interest for your current machine learning project and for your learning path as a data scientist, a data engineer or a data architect, and that it will occupy an important place on your digital or, even better, physical bookshelf.

Please be sure to stop by the Author's forum with any feedback you have. With your help, I’m sure the final book will be great!

Yours,
—Alessandro Negro
brief contents

PART 1: INTRODUCTION

1 Machine Learning and Graph: An introduction
2 Graph Data Engineering
3 Graphs in Machine Learning Applications

PART 2: TECHNIQUES

4 Data Modeling
5 Link Analysis
6 Graph Clustering
7 Graph Classification
8 Nearest Neighbor

PART 3: APPLICATIONS

9 Natural Language Processing
10 Recommendation
11 Relevant Searching on a Knowledge Graph

APPENDIXES:

A Big Data Architecture
B Toolkit
Machine Learning and Graph: An introduction

This chapter covers:

- An introduction to machine learning
- An introduction to graphs
- The role of graphs in machine learning applications

Machine Learning is a large branch in the Artificial Intelligence field. It was born in 1959, when Arthur Samuel, an IBM computer scientist, wrote the first computer program to play checkers [Samuel, 1959]. He had a clear idea in mind:

“Programming computers to learn from experience should eventually eliminate the need for much of this detailed programming effort.”

He wrote the first program by assigning a score to each board position based on a fixed formula. It worked quite well, but in a second approach he had the program execute thousands of games against itself and used the results to refine the board scoring. Eventually the program reached the proficiency of a human player and Machine Learning took its first steps.

Machine Learning is the field of study in computer science that allows computer programs to learn from data.
An entity, such as a person, an animal, an algorithm, or a generic computer agent\(^1\), is learning if, after making observations about the world, it is able to improve its performance on future tasks. In other words, learning is the process of converting experience into expertise or knowledge. Learning algorithms use training data as input, that represents experience, and creates expertise as output, that can be a computer program, a complex predictive model, or tuning on internal variables. The definition of performance depends on the specific algorithm or goal it wants to achieve, in general we consider it to be the extent that the prediction matches specific needs.

Let’s describe the learning process with an example. Consider the implementation of a spam-filter for emails. A “pure-programmed” solution would be to write a program to memorize all the emails labelled as “spam” by a human user. When a new email arrives, the pseudo-agent will search for a similar match in the previous spam emails. If it finds some matches, the new email will be rerouted to the trash folder, otherwise it will pass through the filter, untouched.

This approach could work and, in some scenarios, be useful. Yet, this approach it is not a “learning process” as it lacks an important aspect of the learning: the ability to generalize, to commute the individual available examples in a broader general model. In the specific use case, it means the ability to label unseen emails even though they are dissimilar to previously labelled emails. This process is also referred to as inductive reasoning or inductive inference\(^2\). In order to generalize, the algorithm should scan the training data and extract a set of words whose appearance in an e-mail message is indicative of spam. Then, for a new email, the agent would check if whether one or more of the suspicious words appear and predicts it label accordingly.

Suppose you are an experienced developer with a lot of year of experience, you could ask yourself: why should I write a program that learn how to “program” itself, when I can instruct the computer to carry out the task at hand. Taking the example of the spam-filter, it is possible to write a program that checks the occurrence of some words and classifies it as spam. This approach has three primary disadvantages:

1. A developer cannot anticipate all possible situations. In the spam-filter use case, all the words used in a spam email cannot be predicted upfront.
2. A developer cannot anticipate all changes over time. In a spam-filter, new words can be used, or techniques can be adopted to avoid easy recognition such as hyphens or spaces between single characters.
3. Sometimes a developer cannot write a program to accomplish the task. For example, even though recognizing the face of a friend is a simple task for human, it is impossible to program a software to accomplish this task without the use of machine learning.

\(^1\)Accordingly to the definition stated in [Russell et al.]: An agent is just something that act (agents comes from the Latin agere, to do). Of course all the computer program do something, but computer agents are expected to do more: operate autonomously, perceive their environment, persist over a prolonged time period, adapt to change, and create and pursue goals.

\(^2\)Accordingly to the Stanford Encyclopaedia of Philosophy (https://plato.stanford.edu/entries/logic-inductive/) in inductive logic the premises should provide in some extent support for the conclusion, as different from deductive reasoning in which the premises logically entail the conclusion. This is why sometime (although somebody disagree with this definition) induction is defined as a process of deriving general principles from specific observation.
Therefore, when you are faced with new problems or tasks that you would like to solve with a computer program, consider the following questions to help you to decide if either use or not machine learning:

- Is the specific task too complex to be programmed?
- Does the task require any sort of adaptivity along its life?

A crucial aspect connected to any machine learning task is the training data on which the knowledge is built. Since starting from wrong data leads to wrong results, regardless of the potential performance or the quality of the learning algorithm used.

The aim of this book is to help data scientists and data engineers practitioners approach a machine learning process from two sides: the learning algorithm and the data. In both perspectives, we will use the graph as a valuable mental and technological model. A lot of learning algorithms based on graphs can deliver very efficient predictive models and other algorithms can be improved using whether graphs or graph algorithms in the workflow. The use of graphs also provides many other benefits, such as a valuable storage model for representing “knowledge” from the input of the process, managing the training data, and output storing the predictive model, providing multiple and fast access patterns to it.

This book walks the readers through the entire machine learning project life cycle showing step by step all cases in which the use of graphs could be a valuable and reliable friend in the machine learning process.

On the other hand, graphs are not a panacea for all machine learning projects. For instance, in stream analytics, where it is necessary to process a stream of data to reveal short term anomalies, storing data in the form of a graph could be useless. Furthermore, there are other algorithms that require data in a format that cannot fit in a graph, neither during training nor for model storage and access. This book gives the readers the capabilities to discern whether using a graph in the process would be an advantage and or a burden.

**WHO SHOULD READ THIS BOOK**

Is this the right book for you? If you are a data scientist or a data engineer practitioner, this could be the book that could help you to complete or start your learning path. If you are a manager that has to start or to drive a new machine learning project, this could be the book that could help you to suggest a different perspective to your team. If you are an advanced developer interested in exploring the power of graphs, this book could help you to discover a new perspective in the role of the graph not only as “kind of database” but also as a “enabler” technique for AI.

This is not a compendium on machine learning techniques in general, this book focuses on methods, algorithms and design patterns related to graphs, which are the prominent topic here, specifically how graph approaches can help you to develop and deliver a better machine learning project. Based on my personal experience on building complex machine learning application, this book suggests many recipes in which graphs are the main ingredient to prepare a tasty product to your customer. Across the lifecycle of a machine learning project such approaches can be useful under several aspects: managing data sources more efficiently, implement better algorithms, storing the prediction model so that they can be accessed faster, visualize the result in a more effective way for further analysis. Graph model techniques are
presented in great details, and multiple graph-based algorithms are described. The most complex concepts are illustrated with concrete scenarios and concrete applications are designed.

This writing aims to be a concrete guide that would help you to successfully install a working application in the production. Hence, optimization techniques and heuristics are also described to help you to deal with real data, real problem, real users. Not just toy examples are discussed, but also end-to-end applications from real world use cases are depicted and illustrated with some suggestions to deal with concrete problem.

If these arguments solicited your interest this is definitively the right book for you.

**WHO SHOULDN’T READ THIS BOOK**

If you are looking for a book with a list of basic machine learning methods or if you are looking for a high/business level introduction to machine learning techniques may be this is not the right book for you. If you are not interested in graphs, definitely leave it on the shelf!

### 1.1 Introduction to Machine Learning

#### 1.1.1 Machine Learning Project Lifecycle

A machine learning project is a human process as well as a software project. It involves a large quantity of people, a lot of communication, a lot of work and a mixed of skills. It requires a well-defined approach to be effective. We start our long journey, defining a workflow with clear steps/components that is used along all the book. The mental schema proposed, that is just one of the possible schemas, helps also to better understand the role of the graphs in the development and deployment of a successful machine learning project.

Delivering machine learning solutions is a very complex process and requires more than just selecting the right algorithm(s). It involves multiple tools, a lot of data and different people. Such projects involve numerous tasks related to:

- selecting the data sources;
- gathering;
- understanding;
- cleaning;
- processing;
- evaluating the results;
- deploying.

After deployment, it is necessary to monitor the application and tune it.

One of the most commonly used processes for data mining projects is CRoss Industry Standard Process for Data Mining [With and Hipp (2000)]. Although the CRISP-DM model was designed for Data Mining it can also be applied to generic machine learning project. Key features of the CRISP-DM that make it attractive as part of the base workflow model are:

- it is not-proprietary;
- it is application, industry, and tool neutral;
- it explicitly views the data analytics process from both application-focused and a
technical perspective.

This method can be used for project planning and management, for communicating it and for documentation purposes.

Looking at Figure 1.1, the sequence of the phases are fluid. The arrows indicate only the most important and frequent dependencies between phases but, in a particular project, it depends on the outcome of each phase which phase, or which particular task of a phase, has to be performed next.

The outer circle symbolizes the cyclic nature of process itself. It is not finished once a solution is deployed. Subsequent machine learning processes could benefit not only from the experiences of previous ones (the virtuous cycle of [Berry and Linoff, 1997]) but also from the outcome of the previous processes.

Let’s outline each phase in more detail.

©Manning Publications Co. We welcome reader comments about anything in the manuscript - other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

https://forums.manning.com/forums/graph-powered-machine-learning
BUSINESS UNDERSTANDING

The first phase requires defining the goals of the machine learning project. These objectives are often expressed in general terms like: increase revenues, improve the customer experience, get better and customized search results, sell more products, and so on. In order to convert these high level problem definitions into concrete requirements and constraints for the machine learning project, it is necessary to understand the business and the domain.

Machine learning projects are software projects, and during this phase it is also important to learn the “language” and domain concepts. This will help not only the communication between the data scientist and the internal team during subsequent phases but also will improve the quality of the documentation and the presentation of the results.

The outcomes of this phase are:

- A clear understanding of the domain and the business perspective;
- A definition of the goals, requirements and constraints;
- The conversion of this knowledge into a machine learning problem definition;
- A preliminary and reasonable project plan designed to achieve the objectives.

The goals of the first cycle shouldn’t be too wide, since the first iteration requires a lot of effort related to the “injection” of the machine learning process in the existing infrastructure. At the same time it is important to design the first cycle keeping in mind future extensions.

DATA UNDERSTANDING

The data understanding phase starts by enquiring about the different data sources, collecting some data from each of them, and proceeds with activities in order to:

- get familiar with the data;
- identify data quality problems;
- discover first insights into the data;
- detect interesting subsets to form hypotheses for hidden information.

Data understanding requires domain and business understanding. Moreover, looking at the data helps to better understand the domain and the business perspective. This is why there is a loop link between this phase and the previous one.

The outcomes of this phase are:

- A clear understanding of the data sources available;
- A clear understanding of the different kinds of data and the content of each of them, or at least of all the significant parts for the machine learning goals;
- An architecture design of how to get or extract this data and feed the next steps of the machine learning workflow.

DATA PREPARATION

This phase covers all activities to gather data from multiple sources and organize them into the specific kind of structure required by the algorithm in the modelling phase. Data preparation tasks include record and attribute selection, data merging, data cleaning,
construction of new attributes, and existing data enrichment. As pointed out before, the quality of the data has an enormous impact on the final results of the next phases, this is why this phase is crucial.

The outcomes of this phase are:

- One or more data structure definitions, using adequate design techniques
- A well-defined data pipeline for feeding the machine learning algorithm model training
- A set of procedures for merging, cleaning and enriching data.

Another output of this phase is the identification of the database management system where this data will be stored while waiting to be processed.

For the sake of completeness, it is not always required to have an explicit data store for persisting data before further processing. It is possible to extract data and convert it just before the process phase. Having such an intermediate step, has a lot of advantages related to performance and data quality as well as further extensibility.

**MODELLING**

The modelling phase is where machine learning occurs. Different algorithms are selected and applied and their parameters are calibrated to optimal values. The algorithms are used to build a range of prediction models from which the best is selected later for deployment, once the evaluation phase is complete.

An interesting aspect, which will be discussed after in the algorithm taxonomies, is that some algorithms produce a prediction model while others do not.

The outcomes of this phase are:

- The set of algorithms to be tested in the next phase;
- The related predictive models (where applicable).

There is a close link between Data Preparation and Modelling as one realizes data problems while modelling or one gets ideas for constructing new data. Moreover, some techniques require specific data formats.

**EVALUATION**

At this stage in a machine learning project you have built one or more predictive models that appear to have high quality. Before models can be deployed, it is important to thoroughly evaluate them and review the steps executed to construct the model to be certain it properly achieves the business objectives defined at the beginning of the process.

This evaluation is conducted in a formal way, splitting the data available into a 80% training set and 20% testing set. Another important objective is to determine if there are any important business issues that have not been sufficiently considered.

The outcomes of this phase are:

- A set of values that allow measurement of performance; the specific *measure of success* depends from the algorithm type and the scope;
- A thorough evaluation as to if business goals are achieved;
- Authorization for using the solution in the production environment.
DEPLOYMENT

As machine learning models are built to serve some purpose in an organization, the creation of a model is generally not the end of the project. Depending on the requirements, the deployment phase can be as simple as generating a report or as complex as releasing a complete infrastructure that delivers service to end users. In many cases it will be the customer, not the data scientist, who will carry out the deployment steps. In any case, it is important to understand up front what actions will need to be carried out in order to actually make use of the created models.

The outcomes of this phase are:

- One or multiple reports with the results of the prediction models;
- The predictive models themselves, used for predicting the future and support decision;
- A infrastructure to provide specific set of services to the end users.

Once the project is in production, it is necessary to constantly monitor it, such as evaluating performance, for example.

1.1.2 Algorithm taxonomies

Learning is a very deep and wide domain. Consequently, there are many different branches of Machine Learning algorithms that can be classified or organized in broad categories based on four different criteria:

- Whether or not they are trained using human provided labeled data;
- Whether or not they can learn incrementally;
- Whether they work by building a predictive model or by simply comparing new data points to known data points;
- Whether the learner actively interacts with the environment or passively observes the information provided by the environment.

The taxonomy provides an overview over the machine learning algorithms plethora and it is not exclusive. It helps identify the right set of algorithms required for the specific problem considering also the data available and how it can flow into the system. More in detail, such a classification is useful to understand:

- The kinds of the data needed and how to prepare them;
- How often and in which way retrain model (if any);
- How the quality of prediction could be affected when time lapse;
- The architectural constraints of the solution to be designed.

SUPERVISED VS UNSUPERVISED LEARNING

Learning requires interaction between the learner and the environment. The first classification presented is based on the nature of this interaction during the training phase. Based on the amount of and the type of supervision we can distinguish between supervised and unsupervised learning.
If we look at the learning as a process of “using experience to gain expertise”, **supervised learning** requires training examples/samples (the experience) that contain significant information that is missing in the test examples to which the learned expertise will be applied. A typical example of this learning process is the spam filter. The learner requires “labels”, such as “spam” and “not spam” (the significant information), in the training dataset for each element (emails). It will learn from this how to classify the email. These types of algorithms, generally speaking, have higher performance, in terms of quality of prediction accuracy. On the other hand, the effort to provide labelled data is high and, in some cases, is not applicable at all. Some of the most important supervised learning algorithms are (some of them will be covered in this book):

- k-Nearest Neighbors
- Decision Trees and Random Forest
- Bayesian Network
- Linear Regression
- Logistic Regression
- Support Vector Machines (SVMs)

On the other side of the spectrum, **unsupervised learning** doesn’t require labelled data so there is no distinction between training and test data. The learner processes input data with the goal of coming up with some insight, summary or compressed version about the data. Some of the most important unsupervised learning algorithms are (some of them will be covered in this book):

- Clustering (k-Means)
- Graph Clustering
- Association Rule Mining (a priori)
- Page Rank

In a grey area there is also an intermediate learning process type that could deal with partially labelled training data, usually composed of a mixture of labeled and unlabeled data. This is referred as **semi-supervised learning**. Most semi-supervised algorithms are combinations of different supervised and unsupervised algorithms. An example of such type of algorithms is:

- Semi-Supervised Label Propagation

An outlier of this classification criteria is **reinforcement learning**. The learner, in this case, can observe only the environment, defined as the set of information available at the current time, select and perform actions, and get rewards in return. As a result of this interaction, in both senses between the environment and the learner, the latter learns what is the best strategy, called policy, to pursue in order to get the most reward over time. A policy defines the best actions the system should choose given a specific condition of the environment. Reinforcement Learning is mostly used for moving robots in a room, or play chess, or other type of games.
**BATCH VERSUS ONLINE LEARNING**

The second classification is based on the capability of the learner to respond online, or either in a short time, to streams of incoming data or not. Some algorithms, defined as **online learner**, can learn incrementally using new data as it comes from other streams, called **batch learner**, need to process large amounts of data.

In **batch learning**, the system is trained using all the available data. Such learning process could take a lot of time and computing resources, if the data to be processed is huge, so it is typically performed offline. This is the reason why it is also defined offline learning. In order to inform the batch learning system of new data, a new version needs to be trained from scratch on the full dataset. Once the new model is ready for production the old one can be replaced. Classical data mining process, such as **market basket analysis**\(^3\), belong to this category. The data miner has large amounts of training data to play with before having to output conclusions.

In **online learning**, the system is trained incrementally by feeding it with data instances sequentially, either one by one or grouping them in small or mini batches. In this case the learning process is fast and cheap and can be performed quite often. Online learning is great for systems that receive data continuously and need to adapt to changes rapidly and autonomously. For instance, a stockbroker has to make daily or hourly decision based on stock prices collected so far.

Online learning can also be used to train system using large amounts of data that cannot fit in the resource available, and is called **out-of-core learning**. The algorithm loads mini batches of the data, performs a training step, purges the data, and proceeds to the next batch.

Online learning is generally preferred since:

- it provides a better fit to the current data and current status, and
- it is more efficient in terms of resources consumption. On the other hand, such type of learning is more sensitive to bad data.

To reduce the risk associated with bad data, it is necessary to continuously monitor the system and eventually switch the learning off. It is worth noting that online and offline learning algorithms could be both supervised or unsupervised.

**INSTANCE-BASED VERSUS MODEL-BASED LEARNING**

Another method to categorize learners is based on the capability of the system to generalize the data used during training to creating a model used for prediction. There are two main approaches to generalization: **instance-based learning** and **model-based learning**.

In the instance-based approach, the system learns the training examples by heart and then, for new instances, finds the closest instances from the training examples. This requires a

\(^3\) Market Basket Analysis is the process of analyzing customer buying habits by finding associations between the different items that customers place in their “shopping baskets”. The discovery of such associations can help retailers develop marketing strategies by gaining insight into which items are frequently purchased together by customers.
way for measuring the distance between elements, such as a cosine distance between vectors created using tf-idf⁴ or simply counting the number of words they have in common.

In the model-based approach, the system builds a model from the training dataset that generalizes the training examples and is then used for making predictions. A typical example is the collaborative filtering technique for recommendation engine. Such algorithms uses user-item interactions - such as buy, view, click, etc - as training dataset to build a model. The model is then used to predict the interest of users for unseen or unbought item and promote the item with highest predicted interest.

Using the training dataset to generalize a prediction model is generally a preferable solution in terms of prediction performance, defined as response time and result quality. The issues related to this approach are:

- The time required to build the model.
- Overfitting the training data, which happens when the training dataset doesn’t contain examples that cover the spectrum of possible cases.

**ACTIVE VERSUS PASSIVE LEARNING**

Learning paradigms can also vary accordingly to the role played by the learner during the training phase. We can recognize active and passive learners. The passive learners observe information provided by the environment. In the spam filter example, the passive learner would wait the user to mark emails. The active learners pro-actively interact with the environment at training time by asking questions or performing experiments. In the spam filter example, the active learner would choose the emails and ask the user to label them as spam or not. The active learner could lead to better performance in terms of prediction quality, since it can choose the right data to label avoiding for instance overfitting, but the interaction with users or environment impacts user experience.

### 1.2 Machine Learning Challenges

#### 1.2.1 The source of truth

The CRISP-DM model puts data in the center of the machine learning process by describing the lifecycle from a data perspective. The training data represents the *source of truth* from which any insight can be extracted and any prediction can be accomplished. Managing the training requires a lot of effort. Quoting Jeffrey Heer, a professor of computer science at the University of Washington, “It’s an absolute myth that you can send an algorithm over raw data and have insights pop up”. As a case in point, it has been estimated that data scientists spend up to 80% of the time on data preparation [Steve Lohr, 2014].
I often use the following quotation to shift the focus on to data before discussing the details of the algorithms:

"Even the best learning algorithm on wrong data produces wrong results"

Two seminal papers, [Banko and Brill, 2001] and [Norvig et al., 2009], point out how, for complex problems, data often matters more than algorithms. Both articles consider natural language processing but the concept can be generalized to machine learning in general.

The Figure 1.2, from [Banko and Brill, 2001], shows the learning curves for some learner considering the average accuracy performance from few up to one billion words of training data. The specific algorithm works on text in order to disambiguate words, but now is not relevant the specific type of algorithm since the concept it is valid in general terms. As appear clearly from the image, the performance of the learner are improved, increasing the amount and the quality of the data available during the training phase. These results suggest how to reconsider the trade-off between spending time and money on algorithm development versus spending it on corpus development. On another perspective, as data scientist you can focus on
the vertical dimension - finding a better algorithm - but the graph shows that there is more room for improvement in the horizontal direction - gather more data. As a proof, the worse algorithm in the graph with 10M elements performs much better than the best one with 1M element.

Of course for many problems additional training data has a non-zero cost, which, for supervised learning, may be high. In this sense, collecting data from multiple sources allows not only to access to a huge set of data but also to improve the quality of the data solving problems such as data sparsity, misspelling, correctness and so on. Gathering data from a variety of sources is not an issue. We live in the “big data era” due to the abundance of digital data from many sources like the web, sensors, smartphones, corporate databases, and open data. But if the value comes from combining different data sets, so do the problems. Data from this plethora of sources comes in different formats. Before the learner can analyze it, the data must be cleaned up, merged and normalized into a unified and homogeneous schema that the algorithm can understand.

For the reasons described above the data represents the first big challenge in the machine learning process. These data concerns can be summarized into four separate categories:

- **Insufficient quantity of data.** Machine Learning requires a lot of training data to work properly. Even for simple use cases it needs thousands of examples and for complex problems, such as deep learning or for nonlinear algorithms, you may need millions of examples.

- **Poor quality of data.** Data sources are always full of errors, outliers, and noise. Such poor quality of the data directly affects the quality of the results of the machine learning process, since it is hard for a lot of algorithms to discard wrong or incorrect values and to then detect underlying patterns in this mess.

- **Non representative data.** Keep in mind that machine learning is a process of induction. The model can only capture what it has seen. If your training data does not include edge cases, they will very likely not be supported by the model. If the training data is too noisy or is related only to a subset of possible cases the learner could generate some **bias** or **overfitting** the training data and will not be able to generalize all the possible cases. This is true for either instance based or model based learning algorithms.

- **Irrelevant features.** The algorithms will learn in the right way if the data contains a good set of relevant features and not too many irrelevant features. While it is often a useful strategy to select more features, with the goal of increasing the accuracy of the model, more is not always better. Using more features will enable the learner to find a more detailed mapping from feature to the target and this increases the risk that the model computed overfits the data. Feature selection and feature extraction represent two important tasks during the preparation of the data.

In order to overcome the issues above, data scientists have to gather and merge multiple data sources, clean it, and enrich it using external sources. Moreover, it often happens data is prepared for a certain purpose but then you discover something new and the desired purpose changes.

©Manning Publications Co. We welcome reader comments about anything in the manuscript - other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders. https://forums.manning.com/forums/graph-powered-machine-learning
Often, those are not simple tasks and require not only a significant amount of professional skills but also a data management platform that allows the changes to be performed in the most convenient way. Moreover, the issues related to the quality of the training examples determine a set of data management constraints and requirements for the infrastructure of the Machine Learning project. They can be summarized as:

- **Managing big data.** Gathering data from multiple data source and merging them into a unified source of truth will generate a huge data set. Moreover, as pointed out before, increasing the amount of data will improve the quality of the learning process. The second chapter takes into account the characteristics of a big data platform and how graphs can play a prominent role in taming such a beast.

- **Flexible schema.** Offer a schema model which provides the capabilities to merge multiple and heterogeneous schemas in a unified and homogeneous data structure that satisfy the informational and navigational requirements. Furthermore, the schema should evolve easily accordingly to the changes in the purposes of the Machine Learning project. This book’s fourth chapter will introduce multiple data model schemas and best practices to model the data for several scenarios.

- **Efficient access patterns.** Fast data-reads boosts the performance, in terms of processing time, of the training process. Feature extraction, filter, cleaning, merging and other pre-processes on the training data set benefit from the use of a data platform that provides multiple and flexible access patterns.

### 1.2.2 Performance

Performance is a complex topic in Machine Learning since it can be related to multiple factors:

- **Predictive accuracy**, which can be evaluated using different performance measures. A typical performance measure for regression problems is the **Root Mean Square Error (RMSE)**. RMSE measures the standard deviation of the errors the system makes during predictions. In order words, it looks at the difference between the estimated value from the known value for all the samples in the test data set and calculates the average value in a way that's immune from the signum of such difference. Other techniques for measuring performance are presented later in the book, when discussing the different algorithms, since there are related to the kind of the algorithm. The accuracy depends on several factors like the amount of data available for training the model, the quality of the data, and the algorithm selected. As discussed in section 1.2.1, the data plays a primary role to guarantee a proper level of accuracy.

- **Training Performance**, which refers to the time required for computing the model. The amount of data to be processed and the kind of the algorithm determine the processing time and the storage quantity required for computing the prediction model. Clearly, this problem affects the algorithms that produce a model as a result of the

---

5 Standard deviation is a quantity expressing how much the members of a group differ from the mean value for the group.
training phase. For the instance-based learner, the performance problems will appear later in the process, such as during the prediction. In batch learning this time is generally more pronounced, due to the amount of data to be processed compared with the online learning approach that learns incrementally from a smaller amount of data. Although in the online learning approach the data to be processed is small, the speed at which they are crunched affects the capacity of the system to be aligned with the last data available, which directly affects the accuracy of the prediction.

- **Prediction Performance**, which refers to the response time required to provide predictions. The output the machine learning project could be either a “static” one-time report for helping managers making strategic decision or an online service for end users. In the first case the time required for the prediction phase, as well as the time for computing the model, is not a significant concern, until it completes in a reasonable timeframe (ie: not years). In the second case the prediction speed really matters, since it affects the user experience and efficacy of the prediction. For instance, suppose you are developing a recommendation engine that recommends products similar to what the current user is visualizing accordingly to his/her interest. The user navigation speed is quite high, which implies a significant number of predictions for a very short time interval is required, and few milliseconds available to suggest something useful before the user will proceed further to the next item. In this scenario the speed of prediction is the key to success.

These factors can be translated into multiple requirements for the Machine Learning projects, such as fast access to the data source during training, high data quality, efficient access pattern for the model to accelerate prediction and so on. In this context, graphs could provide the proper storage mechanism for both source and model data, reducing the access time required to read data, as well as offering multiple algorithmic techniques for improving the accuracy of the prediction.

### 1.2.3 Storing the model

In model-based learner approach, the output of the training phase is a model that will be used for making predictions. This model requires time to be computed and has to be stored in a persistence layer to avoid re-computation at every system restart.

The models structure is directly related to the specific algorithm or the algorithm class employed. Examples include:

- An item-to-item similarities for recommendation engine that leverages a nearest neighbor approach.
- The mapping item-cluster that expresses how the elements are grouped in the clusters.

The sizes of the two models differ enormously. Consider a system that contains 100 items. As a first effort, the item-to-item similarities would require 100 x 100 entries to be stored. Taking advantage of optimizations, this number can be reduced considering only the top k similar items, in which case it will require 100 x k entries. Alternatively, the item-cluster mapping requires only 100 entries. Hence, the space required to store the model in memory or on disk, could be huge. Moreover, as pointed out earlier, the model access/query time affects the
global performance during the prediction phase. For these reasons, the model storage management represents a significant challenge in Machine Learning.

### 1.2.4 Real time

Machine Learning is used more and more frequently to deliver real time services to users. Examples run the full spectrum from “simple” recommendation engines that react to the last clicks all the way to self-driving cars that have been instructed not to injure a pedestrian crossing the street. Even though with a different outcome in case of failures, the capability of the learner to react to new “stimulus” coming from the environment is fundamental for the quality of the final results.

For example, consider a recommendation engine that provides real time recommendations for anonymous users. This anonymity, where the user is not registered nor logged in, means, there is no long term “history” about previous interactions and only short-term, session-based information provided by use of cookies. This is a very complex task that involves multiple aspects and affects several phases of a machine learning project. These could be different accordingly to the learner(s) used but can be grouped as:

- **Learn fast.** The online learner should be able to update the model as soon as new data is available. This would reduce the time gap between events or generic feedback, such as navigational clicks or interaction with a search session, and updating of the model. The more the model is aligned to the last events the more it is able to adhere to the current needs of the user.
- **Predict fast.** Once the model is updated, the prediction should be fast, such as a maximum of a few milliseconds, because the user can navigate away from the current page or even change their opinion quite rapidly.

Both of the above categories require algorithms that can align the model quickly as well as a storage mechanism (in memory, on disk or a combined version) that provides fast memorization and efficient access patterns.

### 1.3 Graphs

#### 1.3.1 What is a graph?

A graph is a very simple and quite old mathematical concept. It is a data structure consisting of a set of vertices (or nodes/points) and edges (or relationships/lines) that can be used to model relationships among a collection of objects. Legend says that it was the lazy Leonhard Euler that started talking about graphs in 1736. While visiting Königsberg in Prussia (now Kaliningrad, Russia), Euler didn’t want to spend too much time walking in the city that sat on both sides of the Pregel River and included two large islands which were connected to each other, or to the two mainland portions of the city, by seven bridges. Euler formalized the problem as a walk through the city that would cross each of those bridges once and only once. He proved that it was impossible and this lead to the invention of graphs and the graph theory [Euler, 1736]. So he stayed home instead.
Figure 1.3 illustrates an old representation of Königsberg and the related graph representation used by Euler to prove his thesis.

More formally, a graph is a pair \( G = (V, E) \), where \( V \) is a collection of vertices \( V = \{V_i, i = 1, n\} \) and \( E \) is a collection of edges over \( V \), \( E_{ij} = \{(V_i, V_j) | V_i \in V, V_j \in V\} \). So \( E \subseteq [V^2] \); thus, the elements of \( E \) are 2-elements subsets of \( V \). The simplest way for representing a graph is by drawing either dot or a small circle for each vertex and joining two of those vertex by a line if the corresponding two vertices form an edge. This more formalized description is shown in Figure 1.4.

Graphs can be either directed or undirected depending on whether a direction of traversal is defined on the edges. In directed graphs an edge can \( E_{ij} \) can be traversed from \( V_i \) to \( V_j \) but not in the opposite direction; \( V_i \) is called the tail, or start node, and \( V_j \) is called head, or end node.
In undirected graphs, edges traversals in both directions are valid. In Figure 1.4 is represented an undirected graph, while in the next Figure 1.5 it is represented a directed graph.

The arrow indicates the direction of the relationship.

By default, edges in a graph are unweighted; thus, the corresponding graphs are said to be unweighted. When a weight, a numerical value used to convey some significance, is assigned to the edges the graph is said to be weighted.

The Figure 1.6 shows the same graphs as above but with a weight assigned to each edge.

Two vertices x and y of G are defined as adjacent, or neighbors, if \{x,y\} is an edge of G. The edge E_{ij} connecting them is said to be incident on the two vertices V_i and V_j. Two distinct
edges e and f are adjacent if they have a vertex in common. If all the vertices of G are pairwise adjacent then G is complete.

Figure 1.7 A complete graph where each vertex is connected to the other

Figure 1.7 shows a complete graph where each vertex is connected to all the other vertices.

One of the most important properties of a vertex in a graph is its degree, defined as the total number of edges incident to that vertex, which is also equal to the number of neighbours of that vertex. For instance, in the undirected graph of Figure 1.4, the vertex 2 has degree 3 (since it has the vertices 1, 4, and 5 as neighbours), the vertices 1 (neighbours are 2, 5), 4 (neighbours are 2, 3) and 5 (neighbours are 1, 2) have degree 2 while 3 has degree 1 (it is connected only with 4).

In a directed graph, the degree of a vertex Vi is split into (1) the in-degree of the vertex, defined as the number of edges for which Vi is their end node (the head of the arrow), and (2) the out-degree of the vertex, which is the number of edges for which Vi is their start node (the tail of the arrow). In the directed graph of Figure 1.5, the vertices 1, and 5 have in-degree and out-degree of 1 (since they both have two relationships one in going one outgoing); the vertex 2 has in-degree 1 and out-degree 2 (one ingoing relationship from 1 and two outgoing to 4 and 5); the vertex 4 has in-degree of 2 and out-degree of 0 (two ingoing relationships from 2 and 3); the vertices 3 has out-degree of 1 and in-degree 0 (one outgoing relationship to 4).

The average degree of a graph is computed as follows:

$$ a = \frac{1}{N} \sum_{i=1}^{N} degree(Vi) $$

where N is the number of vertices in the graph.

A sequence of vertices with the property that each consecutive pair in the sequence is connected by an edge is called a path. A path with no repeating vertices is called a simple path. A cycle is a path in which the first and the last vertex coincide. In the Figure 1.4, [1,2,4], [1,2,4,3], [1,5,2,4,3] and so on are paths, in particular the path of vertices 1,2,5 represents a cycle.
1.3.2 Representing Graphs

There are two standard ways for representing a graph $G (V, E)$ in a suitable way to be processed: as a collection of **adjacency list** or as an **adjacency matrix**. Each way can be applied to directed, undirected and unweighted graphs [Cormen et al, 2009]

The **adjacency-list** representation of a graph $G = (V, E)$ consists of an array Adj of lists, one for each vertex in $V$. For each vertex $u$ in $V$, the adjacency list Adj[$u$] contains all the vertices $v$ for which there is an edge $E_{uv}$ between $u$ and $v$ in $E$. In other words, Adj[$u$] consists of all the vertices adjacent to $u$ in $G$.

Figure 1.8 An undirected graph (a) and the related representation as adjacent-list (b)

Figure 1.8(b) is an adjacency-list representation of the undirected graph in Figure 1.8(a). For example, the vertex 1 has two neighbours 2 and 5, so the Adj[1] is the list [2,5]. The vertex 2 has three neighbours 1,4,5, so the Adj[2] is [1,4,5]. In the same way are created the others. It is worth nothing here that since there is no order in the relationships there is specific order in the list, hence the Adj[1] could be [2,5] as well as [5, 2].

Figure 1.9 A directed graph (a) and the related representation as adjacent-list (b)

Similarly, Figure 1.9(b) is an adjacency-list representation of the directed graph in Figure 1.9(a). In this case we consider only the outgoing relationships, but the same can be done.
with the ingoing, what is important is to choose on direction and keep it consistent during the adjacency list creation. For instance, the vertex 1 has only one outgoing relationship with 2 so the Adj[1] will be [2]. The vertex 2 has two outgoing relationships with 4 and 5 so the Adj[2] is [4,5]. The vertex 4 has no outgoing relationships so its Adj[4] is empty [].

If G is a directed graph, the sum of the lengths of all the adjacency lists is |E|. Since every edge can be traversed in a single direction, \(E_{uv}\) will appear only in the Adj[u]. If G is an undirected graph, the sum of the lengths of all the adjacency lists is 2 * |E|. Since if \(E_{uv}\) is an undirected edge, then \(E_{uv}\) appears in Adj[u] and Adj[v]. For both directed and undirected graphs, the adjacency-list representation has the desirable property that the amount of memory it requires is directly proportional to |V| + |E|.

An adjacency lists can be easily adapted to represent weighted graphs by storing the weight \(w\) of the edge \(E_{uv}\) in Adj[u]. The adjacency-list representation is quite robust in the sense that we can modify it to support many other graph variants.

A disadvantage of the adjacency-list representation is that it provides no quicker way to determine whether a given edge \(E_{uv}\) is present in the graph than to search for v in the adjacency list Adj[u]. An adjacency-matrix representation of the graph remedies this disadvantage, but at the cost of using asymptotically more memory.

For the adjacency-matrix representation of a \(G = (V, E)\), we assume that the vertices are numbered 1,2,\ldots,|V| in some arbitrary manner and this numbers are kept consistent during all the life of the adjacency matrix. Then the adjacency-matrix representation of a graph G consists of a \(|V|\times|V|\) matrix \(A = (a_{uv})\) such that \(a_{uv} = 1\) if \(E_{uv}\) exists in the graph otherwise \(a_{uv} = 0\).

```
1 2 3 4 5
1 0 1 0 0 1
2 1 0 0 1 1
3 0 0 0 1 0
4 0 1 1 0 0
5 1 1 0 0 0
```

**Figure 1.10** An undirected graph (a) and the related representation as adjacency-matrix (b)

Figures 1.10(b) is the adjacency matrix representation of the undirected graph represented in Figure 1.10(a). For example, the first line is related to the vertex 1. Such row in the matrix has 1 behind the columns 2 and 5 since that represent the related vertices to which the vertex 1 is connected. All the other are 0. The second row, related to the vertex 2, has 1 behind the columns 1, 4, and 5 since these are the connected vertices.
Figures 1.11(b) is the adjacency matrix representation of the directed graph represented in Figure 1.11(a). As for the adjacency list it is necessary to choose one direction and use it during the matrix creation. In this case, the first row in the matrix has 1 behind the columns 2 since there are on outgoing relationship to. All the other are 0. An interesting feature of the matrix representation is that reading it from the columns it is possible to see the inbound relationships. For example, the column 4 shows that the vertex 4 has two inbounds connections from the vertices 2 and 3.

The adjacency matrix of a graph requires memory directly proportional to $|V|^2$, independent of the number of edges in the graph. In undirected graph the resulting matrix is symmetric along the main diagonal. In such case it is possible to store only half of the matrix cutting the memory needed to store the graph almost in half.

Like the adjacency-list representation of a graph, an adjacency matrix can represent a weighted graph. For example, if $G = (V, E)$ is a weighted graph and $w$ is the weight of the edge $Euv$, $auv$ will be set to $w$ instead of 1.

Although the adjacency-list representation is asymptotically at least as space-efficient as the adjacency-matrix representation, adjacency matrices are simpler, and so we may prefer them when graphs are reasonably small. Moreover, adjacency matrices carry a further advantage for unweighted graphs: they require only one bit per entry. Because the adjacency-list representation provides a compact way to represent “sparse” graphs - those for which the number of edges is greater than the number of vertices - it is usually the method of choice. We may prefer an adjacency-matrix representation, however, when the graph is dense — $|E|$ is close to $|V|^2$ — or when we need to be able to tell quickly if there is an edge connecting two given vertices.

### 1.3.3 Graph as model of networks

Graphs are useful to represent how things are either physically or logically linked to each other in simple or complex structures. A graph to which we assign names and meanings to the edges and vertices, becomes what is known as a network. In these cases, a graph is the mathematical model for describing networks while a network is a set of relations between objects which could include people, organizations, nations, items found in a google search,
brain cells, or electrical transformers. This diversity illustrates the great power of graphs and their simple structure (which also means that they require small amount of disk storage capacity) that can be used to model\(^6\) complex system.

Let’s describe this concept using an example. Suppose we have a graph such as the following.

![Figure 1.12 A non-trivial generic graph](image)

It is a pure graph, in the mathematical definition, that can be represented using an adjacency-list:

![Figure 1.13 The Adjacency-list representation of the graph above](image)

The graph above can be the model of several networks, accordingly to the type of edges and vertices:

---

\(^6\) In this context the verb model is used in terms of representing in a simplified way a system or phenomenon. This model aims also at representing in a way that it can be easily processed by a computer system.

©Manning Publications Co. We welcome reader comments about anything in the manuscript - other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

https://forums.manning.com/forums/graph-powered-machine-learning
1. If the vertices are people and each edge represents any sort of relationship between humans, such as friendship, parent/child relationships, co-workers, etc., the graph becomes a **social network**;

2. If the vertices are information structure like web pages, documents, or papers, and edges represent logical connections such as hyperlinks, citations or cross references, the graph becomes an **informational network**;

3. If the vertices are computers or other devices that can relay messages and the edges represent direct links along which messages can be transmitted, the graph becomes a **communication network**.

4. If the vertices are cities and the edges represent direct connections using flights or trains or roads, the graph becomes a **transportation network**.

This is a very small set of examples that demonstrates how the same graph can represent multiple networks by assigning different “semantics” to edges and vertices. Figure 1.14 illustrates multiple types of networks.

---

Figure 1.14 Paper co-citation network; Arpa Network July 1975; National Electricity Transmission Grid of

---

https://scienceandartblog.files.wordpress.com/2015/08/infovis.gif

http://som.csudh.edu/fac/lpress/history/arpamaps/13july1975.jpg

©Manning Publications Co. We welcome reader comments about anything in the manuscript—other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders.

https://forums.manning.com/forums/graph-powered-machine-learning
Looking at Figure 1.14, we can spot another interesting characteristic of graphs: they are very communicative. They are able to display information in a very clear manner and this is why often they are used as **Information Maps**. Representing data as networks and using the graph algorithms it is possible to:

- find complex patterns;
- make them visible for further investigation and interpretation.

When the power of the machine learning is combined with the power of the human brain, it enables efficient, advanced and sophisticated data processing and pattern recognition. Networks are useful for displaying data by highlighting connections between elements. Newspapers and web news sites are increasingly use them, not only for helping people navigating the data in a proper way but as a powerful investigation tool. Recently (at the time of this writing), the Panama Papers[^11] showed the astonishing features of networks. The International Consortium of Investigative Journalists (ICIJ) has exposed highly connected networks of offshore tax structures used by the world’s richest elites. These structures were uncovered from leaked financial documents and were analyzed by the journalists. They extracted the entities (people, organizations, and any sort of intermediaries) and relationships (protector, beneficiary, shareholder, director, and so on) from documents, stored them in a network, and analyzed them using visual tools. The results looked like this:

[^11]: https://panamapapers.icij.org
Networks, graph algorithms and graphs visualization made evident something that otherwise would have been impossible to discover using traditional data mining tools.

A lot of interesting examples in this direction are also available in the Valdis Krebs blog\footnoteref{12}, who is an organization consultant that specializes in social network applications. We will see some of his works later as examples of mixing graph-powered machine learning with human mind passing through graph visualization, here we consider one of the more famous.

\footnotetext{12}{http://www.thenetworkthinkers.com}

©Manning Publications Co. We welcome reader comments about anything in the manuscript - other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders. https://forums.manning.com/forums/graph-powered-machine-learning
The data in Figure 1.16 is gathered from Amazon.com and is their list of the top political books purchased in the USA. Employing network analysis principles exploited the data to create a map of books related to the 2008 US presidential election. Looking at Figure 1.16, two books are linked if they were often bought together, or by the same buyer. These are also-bought pairs - people who bought this book also bought that book.

There are three different and non-overlapping clusters:

1. An Obama cluster of books on the upper left corner;
2. A Democratic cluster in the middle;
3. A Republican cluster on the bottom right corner.

In 2008 the US political climate was very polarized; this is mirrored in Amazon’s political book data, with Figure 1.16 showing a deep divide between conservative (RED) and liberal (BLUE) voters. There were no connections, nor any intermediaries between red and blue books, and each cluster was completely distinct from the other. There was a separate cluster of people reading books on the then new candidate, Obama, but they were not interested in reading/purchasing other political books.

Four years later, in 2012, the same analysis produced a network that appeared substantially different.
In Figure 1.17 there are a lot of books that act as bridges between the different clusters. Moreover, potential voters appear to be reading books about both of the candidates. The result is a more complex network in which there are no longer any isolated clusters. [Krebs, 2012].

The example of political book networks introduces an important aspect of networks. If a graph is a pure mathematical concept that lives in its own Platonic world\textsuperscript{13}, networks, as abstraction of some concrete system or eco-system, are subjected to "forces" that, acting on them, changes their structure. We refer to them as surrounding contexts: factors that exist outside the vertices and edges of a network, but which nonetheless affect how the network’s structure evolve over time. The nature of such contexts and the type of forces are specific to the kind of network. They are evident in the social networks, where each individual has a distinctive set of personal characteristics. Similarities and compatibilities between two people characteristics influence the link creation or deletion [Easley and Kleinberg, 2010].

One of the most basic notions governing the structure of social networks is homophily (from the Greek, "love of the same"): links in a social network tend to connect people who are similar to one other. More formally, if two people have characteristics that match in a proportion greater than expected in the population from which they are drawn or the network of which they are part, then they are more likely to be connected [Verbrugge 1977].

\textsuperscript{13} Mathematical Platonism is the metaphysical view that there are abstract mathematical objects whose existence is independent of us and our language, thought, and practices. [https://plato.stanford.edu/entries/platonism-mathematics/]

©Manning Publications Co. We welcome reader comments about anything in the manuscript - other than typos and other simple mistakes. These will be cleaned up during production of the book by copyeditors and proofreaders. https://forums.manning.com/forums/graph-powered-machine-learning
converse is also true: if two people are connected, then they are more likely to have common characteristics or attributes. This is why our friends on Facebook for example don’t look like a random sample of people, but they are generally similar to us along ethnic, racial and geographical dimension; they are similar in age, occupations, interests, beliefs and opinions. This observation has a long history, with origins long before Zuckerberg wrote the first line of code of Facebook. The underlying idea can be found in the writings of Plato ("similarity begets friendship") and Aristotle (people "love those who are like themselves") as well as in folk proposition such as “birds of feather flock together”. Homophily principle also applies to groups, organizations, countries or any aspect of social units.

Understanding the surrounding contexts and the related forces that act on a network helps machine learning tasks in multiple ways:

1. Networks are conduits for both wanted and unwanted flows. Marketers are always trying to reach and persuade people. Personal contact is most effective, if one can find a way to start a snowball rolling. This is the concept at the base of so called viral marketing.

2. Understanding such forces allows the prediction of how the network will evolve over time, and proactively react to such changes or use them for the specific business purposes.

3. There are findings in sociological and psychological disciplines that point to the relevance of a person’s social network in determining their tastes, preferences, and activities. This information is useful while building recommendation engines. One of the problems related to them is the cold start problem, which means that you can’t predict anything for a new user since you have no “purchase history” about them. Social networks and the homophily principle can be used to make a recommendation based on the tastes of connected users.

### 1.3.4 Property graph model

A graph that is used to represent complex networks requires more information than a simple list of nodes and relationships. Such simple structures that can be easily extended to a richer graph model that contains more information are called property graph. This model has been introduced by graph database management system providers to enrich the simple graph and to tie a set of information to graph structures. This allows a more complex set of query features, typical of any database management system (DBMS), such as projection, filtering, grouping, counting, and so on.

A property graph is defined, accordingly to the openCypher project\(^{14}\), as “a directed, vertex-labeled, edge-labeled multigraph with self-edges, where edges have their own identity.” In the property graph, we use the term node to denote a vertex, and relationship to denote an edge.

---

\(^{14}\) [https://github.com/opencypher/openCypher/blob/master/docs/property-graph-model.adoc](https://github.com/opencypher/openCypher/blob/master/docs/property-graph-model.adoc)
A property graph has the following properties (trying to define them in a platform-agnostic way):

- A set of **entities**. An entity represents a node or a relationship.
- Each entity has an **identifier** that uniquely identify it across the entire graph.
- Each relationship has a direction, a name that identifies the kind/type of the relationship, a start node and an end node.
- Entities can have a set of **properties**, which are typically represented as key/value pairs.
- Nodes can be tagged with one or more **labels**, which group nodes together and indicate the roles they play within the dataset.

![Figure 1.18 A property graph](image)

A property graph is still a graph but the communication capability is even greater than before. In the Figure 1.18, we can easily recognize that the Person Alessandro **WORKS_FOR** the Company GraphAware as does Michal and Christophe. The name is a property of the nodes Person, while start_date and role are the properties of the relationship WORKS_FOR. For each
Person their nationality is stored using the relationship `HAS_NATIONALITY` connected to `Country` nodes that have the property name for storing the country name.

As for relational databases, there are some best practices or style rules for defining a model for a graph. For example, the name of the labels for node should be singular, since they represent a specific entity; while the name for the relationships should reflect the direction.

Obviously there are multiple ways for representing the same set of concepts. In the model in Figure 1.18, the nationality can be stored as a property of the Person nodes. The schema could change a lot accordingly to the specific needs in terms of the access patterns and the underlining graph database management system. In the chapter 4 we will go over numerous models for representing data each of which has specific scope and satisfies the specific requirements of target application.

## 1.4 The role of graph in the machine learning

Graphs are used to characterize interactions between objects of interest, to model any kind of simple and complex networks, or, in general, to represent real-world problems. Because they are based on a rigorous but straightforward formalism, they are used in many scientific fields from computer science to historical sciences. We shouldn’t be surprised to see them being widely used in Machine Learning as a powerful tool that can enable intuitive properties and power a lot of useful features. Graph-based machine learning is more and more a resilient piece of logic, transcending numerous “traditional” techniques.

Many companies, of any size, are currently using this approach for providing more advanced machine learning features to their customers. One prominent example is Google, who is using graph-based machine learning as a core of its Expander’s platform: the technology that is behind many of the Google products and features you may use every day, such as reminders in Inbox and smart messaging in Allo, or the latest image recognition system in Google Photos.¹⁵

Building a graph-native machine learning platform has numerous benefits since graphs can be not only a valuable tool for overcoming previous described challenges but also for delivering more advanced features that are impossible to implement without the graph support.

To better understand where graphs fit in the Machine Learning project workflow we split the process in 4 main tasks:

- **Managing the data sources**, which refers to all the tasks of gathering, merging, cleaning and prepare the training dataset for the learning phase.
- **Learning**, which involves the application of machine learning algorithms on the training dataset.
- **Storing and accessing the model**, which includes the approaches to store the predictive model and the access pattern for providing prediction.
- **Visualizing**, which refers to all the way in which data can be visualized to support the

¹⁵ [https://research.googleblog.com/2016/10/graph-powered-machine-learning-at-google.html](https://research.googleblog.com/2016/10/graph-powered-machine-learning-at-google.html)
These points are summarized in the following mental map:

![Machine Learning Project Timeline](image)

**Figure 1.19** The mental map describing the 4 stages of a machine learning project

The Figure 1.19 will recur often across the entire book. The schema will help to immediately figure out where the current discussion fits in the project workflow.

The mental map illustrated earlier presents the machine learning project from his process perspective, and it is the best way for figuring out where we are in the lifecycle. On the other side it also useful think about the project from a broader perspective, let’s say task oriented. The following image describes the main contact points between machine learning and graphs considering the goal of the tasks.
Figure 1.20 Graph-powered Machine Learning

Such mind map represents the summary of the book and can be used to immediately visualize conceptually the role of the graphs in the machine learning panorama. Looking at Figure 1.20, graph features are grouped in three main areas:

- **Data Management**: In this area are described the features provided by graphs that help machine learning project to deal with the data.
- **Data Analysis**: This area contains methods relying on graphs for learning and predicting.
- **Data Visualization**: Here are listed the techniques for using graphs as a visual tool helping people to communicate, interact with data, and discover insight using the human brain.

The schema also shows the mapping between the graph-based techniques and the phases in the CRISP-DM.

The same mental model can be represented in a different way

### 1.4.1 Data Management

Graphs allow the learning system to explore more of your data, to access it faster, and/or to easily clean and enrich it. Using knowledge graphs, they allow us to store the prediction models in the same location of the real data. Traditional learning systems train on a single...
table prepared by the researcher, whereas a graph-native system can access more than just that table.

The graph-powered data management comprehends:

- **Connected sources of truth**: Graphs allow to merge multiple data sources in a single, uniform and connected dataset ready for the training phase. This represents a great advantage by reducing the data sparsity, increasing the amount of data available, and simplifying the data management.

- **Knowledge graphs**: It is possible to move the technique before a little bit further. Knowledge graphs provide an homogeneous data structure for combining not only data sources, but also prediction models, manual provided data, and external sources of knowledge. The resulting data is machine-ready and can be the used during training, prediction or visualization.

- **Fast data access**: Tables provide a single access pattern related to row and column filters. Graphs, on the other hand, provide multiple access points to the same set of data. This helps to improve the performance by reducing the amount of data to be access to the baseline minimum for the specific set of needs.

- **Data enrichment**: This method comprehends not only extending existing data with external source of data, but also data cleaning and merging. The schema less nature of graphs and the access pattern provided within graph databases help during this processes.

- **Features selection**: Identifying relevant features in a dataset is key for several machine learning algorithms like classification. Providing fast access to data and multiple query patterns, graphs help to speed up the features identification and extraction.

Connected sources of truth and Knowledge graphs are a valuable aid during the Data Understanding and Data Preparation phases in the CRISP-DM while fast data access, data enrichment, and features selection are involved during the data preparation.

### 1.4.2 Data Analysis

Graphs can be used to model and analyze the relationships between entities as well as their properties. This brings an additional dimension of information that graph-powered machine learning can harness for prediction and categorization. The schema flexibility, provided by graphs, also allow models to coexist in the same dataset also allowing them to be mixed.

The graph-powered data analysis comprehends:

- **Graph algorithms**: Several graph algorithms, such as clustering, page rank, and link analysis are useful to identify insight in the data and for analysis purposes. Moreover they can be used as a first data pre-processing in a more complex analysis process.

- **Graph accelerated machine learning**: The graph-powered features extraction discussed earlier is just an example of how graphs can help to speedup or improve the quality of the learning system. In general graphs can help in filtering, cleaning, enriching, merging data before or during training phases.

- **Network dynamics**: Surrounding contexts and the related forces that act on networks
allow not only the understanding of network dynamics but also to leverage them for improving the quality of the prediction.

- **Mixing models**: Multiple models can coexist in the same graph, leveraging the flexible and fast access patterns provided they can be merged during the prediction improving the final accuracy. Moreover the same model, sometimes, can be used in different ways.

- **Fast model access**: Real-time requires fast prediction, which implies a model that can be accessed as fast as possible. Graphs provide the right patterns for these scopes.

Graph algorithms, graph accelerated Machine Learning, and network dynamics methods are more involved during the modeling phase, since they are connected to the learning process more than other. The deployment phase leverages mixing models and fast model access methods, since they operate during the prediction stage.

### 1.4.3 Data Visualization

Graphs bring high communication power, and they can display multiple types of information at the same time in a way in which the human brain can understand. This feature is greatly important in a machine learning project, both for sharing results, analyzing them, or for letting people navigating data.

The graph-powered data visualization comprehends:

- **Data navigation**: Networks are useful for displaying data by highlighting connections between element. They can be used not only for helping people navigating the data in a proper way, but also as a powerful investigation tool.

- **Human brain analysis**: Displaying data in the form of a graph unleashes the power of the machine learning by combining it with the power of the human brain, enabling efficient, advanced and sophisticated data processing and pattern recognition.

- **Improved communication**: Graphs and, in particular, property graphs are “whiteboard friendly”, which means they are conceptually represented on a board just as they are stored in the database. This reduces the gap between the technicalities of a complex model from the way in which it is communicated to the domain experts or the stakeholders. Effective communication improves the quality of the final results since it reduces the issues in the comprehension of the domain, the business goals, the needs and the constraints of the project.

The improved communication is particularly important during the business and data understanding phase. While the data navigation and the human brain analysis are mostly related to the evaluation phase.

### 1.5 Summary

This first chapter introduces the basic ideas and motivations behind a graph-based machine learning project. It illustrates some basic concepts that are fundamental to understand how graph models can be valuable tools for delivering better results at the end of the machine
learning process. Moreover, it provides the mental framework to understand the next chapters. More in details the main concepts presented are:

- Machine Learning aims at developing computer programs that gain autonomously experience from sample data converting in expertise without being explicitly programmed.
- Machine learning projects is not only a software project, it is a human process that involves a bunch of people with different skills and a lot of work. It requires a well-defined and a systematic approach to succeed. CRISP-DM provides the formal project lifecycle to drive such kind of projects, helping to deliver the right results.
- The challenges that any machine learning project has to deal with are mostly related to the data management – either in terms of training data set either in terms of prediction model – and the learning algorithm performances.
- Graphs are very simple mathematical concepts that can be used to model and analyze complex networks. The surrounding contexts operate outside of the network operating some forces that determine the network evolution
- Graphs and Networks can empower machine learning projects in several ways along three different directions: Data Management, Data Analysis and Data Visualization.

In this chapter you learned:

- How to organize a machine learning project following the CRISP-DM pattern
- How to classify the machine learning algorithms accordingly to different criteria, such as type of training data (labelled or not labelled), model creation, etc., and the implication of this
- How to represent a graph using adjacency matrix and adjacency list
- A first overview of the role of graphs in the machine learning panorama

1.6 References


[Shai Shalev-Shwartz and Shai Ben-David, 2014], Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press, 2014


[Alon Halevy, Peter Norvig and Fernando Pereira, 2009], The unreasonable effectiveness of data. Intelligent Systems IEEE Vol. 24 no. 2 pp. 8-12 2009

[Leonhard Euler, 1736], "Solutio problematis ad geometriam situs pertinentis". Comment. Acad. Sci. U. Petrop 8, 128–40, 1736

[Rada Mihalcea, Dragomir Radev, 2011], Graph-Based Natural Language Processing and Information Retrieval. Cambridge University Press, June 2011


[David Easley, Jon Kleinberg, 2010], Networks, Crowds, and Markets: Reasoning about a Highly Connected World. Cambridge University Press, September 2010