

Chapter 1

Why Kernels for Structured Data?

Der äußere Eindruck auf die Sinne, samt der Stimmung, die er allein und für sich in uns hervorruft, verschwindet mit der Gegenwart der Dinge. Jene beiden können daher nicht selbst die eigentliche 'Erfahrung' ausmachen, deren Belehrung für die Zukunft unser Handeln leiten soll. [...] mithin frei von der Gewalt der Zeit ist nur Eines: der 'Begriff'.

(Arthur Schopenhauer, Zur Lehre von der abstrakten oder Vernunft-Erkenntnis)

Being able to learn from experience is what enables humans to adapt to an ever changing environment. Be it the extraction of physical laws from experimental data or be it the use of experience for decision making, learning from examples is at the core of intelligent behaviour. Two strongly connected types of learning are modelling of data and estimating a property of some objects before it is observed. At a time where the amount of data collected day by day far exceeds the human capabilities to extract the knowledge hidden in it, it becomes more and more important to automate the process of learning. Machine learning and data mining are two research fields concerned with automated learning. In many business and scientific applications the use of machine learning methods helped speed up and reduce the cost of certain processes. For some example applications see [Graettinger (1999); Fayyad *et al.* (1996)].

Within these research fields, recently a class of learning algorithms is receiving an ever increasing amount of interest from researchers as well as practitioners. The most popular learning algorithm in this class of so-called *kernel methods* is the *support vector machine*. Its popularity stems from its sound foundation in learning theory and its ability to provide superior em-

pirical results in many benchmark as well as real-world applications. For an overview over some of these applications we refer to [Bennett and Campbell (2000)]. A very prominent recent example is the superior performance of support vector machines in text classification [Joachims (2002)].

Support vector machines and other kernel methods can directly be applied to all kinds of data that are easily embedded in a Euclidean space. This is the case in the traditional setting of machine learning, which only considers examples represented in a single row of a table. However, there are many potential machine learning applications, where this is not the natural representation. Such an application is, for example, the classification of compounds given their chemical structure graph.

This book is therefore concerned with extending kernel methods in such a way that they can be applied to learning problems where the representation of objects in a single row of a table is not trivial. In particular, we will consider examples represented by terms in a typed higher-order logic, as well as examples represented by graphs. Both logic and graphs are two very common and general languages for representing structured objects. As we will see, they lend themselves naturally to certain kinds of application domains. Together they cover most — if not all — kinds of structured data that might occur in real-world applications. With a systematic extension of kernel methods to logic and graphs, we will be able to apply the whole range of available kernel methods to machine learning problems that involve structured data.

In the remainder of this chapter we will first introduce machine learning, kernel methods, and the particular issues of learning with structured data in some more detail. We will use a simple drug discovery problem to illustrate these concepts. After that, we summarise the goals and contributions of this book.

1.1 Supervised Machine Learning

We begin with an example application.

Example 1.1. The drugs we can buy today to fight a particular illness are probably better than the ones we could have bought a century ago and probably worse than the ones we could buy in a century. The difference between the drugs that fight the same illness are then properties like the effectiveness of the drug against the disease or the solubility of the drug in water. Researchers are working on improving these kind of properties of

chemical compounds used as drugs. As an example, we will focus on the effectiveness of drugs in fighting some disease.

Suppose now we knew beforehand a way to estimate this effectiveness for any chemical compound. This knowledge could be used in many ways to speed up the design of new – improved – drugs as well as to reduce the costs of developing new drugs. A bit more formally, for each disease we would like to have a function that estimates the effectiveness of any chemical compound in fighting this disease.

There are different ways to obtain such a function given a disease. One possibility is to make use of chemical knowledge and knowledge about the details of the disease. A different possibility – the one we are interested in – is to look at drugs known to be effective in fighting this disease. Clearly, we can find functions that tell us that the known drugs are effective against the illness — the challenge is to find a function that generalises well over the given examples and thus is able to estimate well the effectiveness of other chemical compounds in fighting this disease.

This, exactly, is the type of problems *machine learning* is interested in. A bit more formally, we need to first introduce the representation space, the learning task, and the hypothesis space. The *representation space* is the set of all possible object descriptions that can occur in a given problem. Let \mathcal{X} denote the representation space. In the example above, the representation space could be the set of all possible chemical structure graphs. The learning task we will mostly consider is *supervised learning*. There, we are interested in one particular (target) property of the objects and know the value of this property for some objects. Let \mathcal{Y} denote the set of possible values of the property. In the example above, \mathcal{Y} could be the set containing the values $+1$ indicating an active compound and -1 indicating an inactive compound. In this case, one speaks of *binary classification*. Alternatively, \mathcal{Y} could be the set of real numbers representing some measure of the compound's activity. In this case, one speaks of *regression*. In supervised learning, we try to find a function that is able to estimate the value of the target property for all elements of the representation space. That is, we try to find a function from \mathcal{X} to \mathcal{Y} that has good predictive performance, or in other words, that generalises well over the objects with observed value of the property. We refer to Section 2.2.1 for a more precise definition of these terms.

It turns out that the difficulty of machine learning is not to find a function that reproduces exactly the known property values. There are many

different such functions and unfortunately the function that has best predictive performance need not be among the functions that reproduce best the known property values. Indeed, machine learning algorithms usually prefer “smooth” functions that approximately reproduce the known property values over “non-smooth” functions that exactly reproduce these. This is called *regularisation* (see Section 2.3.2). Usually, also not even all possible functions are considered as solutions to the learning problem by every learning algorithm. The set of possible solutions considered by a learning algorithm is called its *hypothesis space*.

1.2 Kernel Methods

We will now have a fresh look at the machine learning problem introduced in the previous section.

Let us try to devise a simple method to find a function which estimates the effectiveness of chemical compounds against some illness. Recall that the way we want to find this function is to look at chemical compounds for which we know their effectiveness against this illness. Building on this ‘experience’ with other chemical compounds we want to construct an estimating function.

Suppose we knew how to compute a meaningful similarity between chemical compounds. We will now discuss how to predict the effectiveness of a chemical compound against some disease given that we know the effectiveness of other chemical compounds against this disease. ‘Meaningful similarity’ implies here that chemical compounds with high similarity have similar chemical properties and biological activity.

The machine learning method known as *radial basis function networks* works in two steps. The first step is a rather technical step in which some parameters (weights) of the estimating function are determined. We will skip this step here. In the second step we compute the similarity of the chemical compound we are interested in with all chemical compounds that are known to be effective and with all chemical compounds that are known not to be effective. Using the weights obtained in the previous step we can compute a weighted sum over the similarities to effective compounds as well as a weighted sum over the similarities to not effective compounds. Then we end up with two numbers. Comparing these two numbers we can estimate the effectiveness of the chemical compound we were originally interested in. The intuition behind this idea is that a chemical compound

that is highly similar to many effective compounds but to few not effective compounds is likely to be effective itself and vice versa.

Kernel methods are a special kind of these radial basis function networks. By restricting the class of similarity measures that are considered to so called *positive definite kernel functions*, the parameters of the estimating function can be determined more efficiently and more reliably. The hypothesis space considered by kernel methods consists of all linear combinations of a given class of functions. More precisely, the hypothesis space of kernel methods is the set of linear combinations of positive definite kernel functions with one argument set to an element of the representation space. Mathematically, this space is a Hilbert space with a well-defined inner product and thus norm. For more details see Section 2.3.

The other important part of kernel methods is the algorithm that, given a kernel function and a set of objects with known value of the target property, chooses one function from the hypothesis space that is expected to have best predictive performance. In kernel methods it is common to formulate this as a convex optimisation problem. In order to avoid “non-smooth” solutions, the objective function of the optimisation problem is not just the deviation of the predicted values of the target property from their observed values (the so-called *loss*) but the sum of this deviation and a regularisation term. In fact, in most kernel methods, the Hilbert space norm of the hypothesis function is used as the regularisation term. Whenever the loss function, measuring the deviation between predicted and observed values, satisfies certain properties, this optimisation problem can be guaranteed to be convex as a result of only considering positive definite kernel functions.

The *support vector machine*, the most well known and most frequently used kernel method, is derived by choosing one particular loss function, i.e., one particular way of measuring the deviation of the predicted values from the true values of the target property. The convex optimisation problem can in this case be solved by quadratic programming methods. For this learning algorithm very good theoretical properties have been shown and very good empirical results have been achieved.

One of the features of kernel methods that we will exploit in this book is the independence between the learning algorithm and the kernel function. Different hypothesis spaces can be explored by simply using a different kernel function – without modifying the learning algorithm.

1.3 Representing Structured Data

This book is concerned with extending kernel methods such that they can be applied to machine learning problems where the representation of each object in a single row of a table is not trivial. In the context of the drug design problem introduced above, we consider the representation of chemical compounds by chemical structure graphs. That is, we represent a molecule by a set of atoms, a set of bonds connecting pairs of these atoms, and an assignment of element-types (carbon, oxygen, ...) to atoms and bond-types (single, double, aromatic, ...) to bonds. In this representation, for example, water is represented by a set of three atoms and a set of two bonds connecting the second atom to both other atoms. Furthermore, the element-type oxygen is assigned to the second atom and the element-type hydrogen to the first and third atoms.

Kernel methods have been applied to various real-world problems with very good empirical success. However, as pointed out above, kernel methods, just like other standard machine learning tools, can not directly be applied to problems like the one described in the example above. Why? Machine learning has traditionally focused on problems where each object can be represented in a single row of a table. In the drug design problem described above, however, describing each example in a single row of a table is difficult. Why is this difficult? Consider we had a function that maps every graph to a single row in a single table, such that “classical” machine learning algorithms can directly be applied to the images of the graphs under this map. Ideally, we would like to require three properties of this map: (i) Isomorphic graphs are mapped to the same image. (ii) The function can be computed efficiently, i.e., in time polynomial in the size of the graphs. (iii) Non-isomorphic graphs are mapped to different images. As the problem of deciding graph isomorphism is believed not to be efficiently solvable, i.e., it is believed not to be in P , we can not hope to find a function that satisfies all three properties. Without such a function we can not hope to apply standard machine learning algorithms to graphs represented in a table. Of course we could instead try to find a function that violates some of the conditions but is still good enough for most learning problems we consider or we could try to modify machine learning algorithms such that they interpret graphs represented in a table by one imperfect map in the correct way (respecting isomorphism, for example). Both approaches are not trivial and introduce various complications.

Fortunately, because of the modularity of kernel methods we also have

a another option: We leave the learning algorithm unchanged, represent graphs in any meaningful way, and just adapt the kernel function to the chosen representation. This approach does not simplify the general problem but it allows a more systematic solution. Furthermore, considering that positive definite kernel functions themselves can be seen as inner products between the images of examples in a different space, it appears more natural to extend kernel methods to structured data by directly defining a positive definite kernel function on the data rather than first mapping it into a table and then mapping it (implicitly using the kernel function) into yet another space. Last but not least, mapping the data only once – implicitly using the kernel function – has computational advantages that we will explore later in this book.

1.4 Goals and Contributions

The central question this book focuses on is: How can kernel methods be applied in learning problems with structured data? The simplest answer to this question is that kernel methods can be applied to any kind of data as long as a meaningful kernel function is known. This gives rise to new questions like: How can we define what a meaningful kernel function is? Given some structured data — how can we define a meaningful kernel function in a systematic way? The second question is the central question of this book and we need to have an idea about the answer to the first question to answer the second. We will thus provide a categorisation of (more or less) meaningful kernel functions before defining kernel functions for specific kinds of data.

Based on these considerations, we develop two kernel functions for rather general data structures commonly used in computer science: logic and graphs. The logic we use is a higher-order logic with polymorphic types. For knowledge representation purposes we use only the ground terms in this logic and not, as frequently done in first-order logic, sets of predicates. The reason we can do this without losing much expressivity is that higher-order logics allow for the direct modelling of sets inside a term rather than by using a set of predicates only.¹ As a second representation language we

¹The basic terms are powerful enough to even represent graphs in the usual way by introducing a set of identifiers (the vertices) and a set of edges, each consisting of a (ordered or unordered) pair these identifiers. However, the algorithmic and computational challenges of terms with and without identifiers are rather different. For example, without identifiers there are no convergence issues at all, while if we try to exploit the

consider directed and undirected labelled graphs. The distinction between basic terms and graphs is beneficial for the clarity of the description of the kernel functions as well as from a computational perspective. Both formalisms, as we will see, lend themselves naturally to certain kinds of application domains. Together they cover most — if not all — kinds of structured data that might occur in real-world applications.

Work on kernel functions for structured data has started with the influential work of Haussler (1999) and Watkins (1999b). Both proposed rather general frameworks that left several questions open to particular applications of these frameworks. This book is the first systematic approach to define kernel functions for structured data and to apply these kernel functions to large scale real-world machine learning problems. Our empirical evaluation shows that kernel methods with our kernels for structured data substantially outperform conventional methods on a variety of important application domains.

To sum up, the four main contributions of this book are:

- Characterisation of meaningful kernels for structured data and investigation of their computational implications.
- Definition of kernels for logic-based representations: Basic term kernels.
- Definition of kernels for graph-based representations: Walk-based kernels and cyclic pattern kernels.
- Empirical evaluation in a variety of application domains.

1.5 Outline

The outline of this book is as follows:

Chapter 1 gave a general introduction to this book and the areas in which it is located.

Chapter 2 first introduces the necessary mathematical tools needed for the further developments of this book. Then the general learning problem and the kernel based approach to solving learning problems are discussed. Afterwards a set of different kernel methods are described that will be used later in this book.

Chapter 3 develops the necessary basics needed for the kernel definitions

identifiers in a term in a meaningful way, convergence will be an issue. We thus consider basic terms (without such identifiers) and graphs, separately.

in the remainder of this book. First, some general remarks on kernels are given and examples of kernel functions often used for inner product spaces are given. Then it is described which combinations of kernels are again kernels. After that alternative kernels for sets are discussed and related work is described.

Chapter 4 describes how kernels for terms in a higher-order logic can be defined based on the type structure of the terms. First, an overview of the higher-order logic is given. Then the default kernel based on the type structure is defined and it is shown that it is positive definite. After that, the implications of using this kernel in a particular class of learning problems is analysed in more detail, before several applications are described.

Chapter 5 discusses kernels for instances that have a natural representation as a graph. First, graphs and some related concepts are introduced. Then, after the difficulty of defining computationally tractable graph kernels has been analysed, a number of effectively applicable graph kernels are proposed. Their application to relational reinforcement learning as well as molecule classification is described thereafter.

Chapter 6 concludes.

1.6 Bibliographical Notes

This book builds on the following published articles:

- Thomas Gärtner, Tamás Horváth, Quoc V. Le, Alex J. Smola, and Stefan Wrobel. Kernel methods for graphs. In L. B. Holder and D. J. Cook, editors, *Mining Graph Data*. John Wiley & Sons, Inc., New York, NY, USA, 2007. Copyright 2007 John Wiley & Sons, Inc. Reprinted with kind permission of John Wiley & Sons, Inc.
- Thomas Gärtner. Predictive graph mining with kernel methods. In S. Bandyopadhyay, U. Maulik, L. B. Holder, and D. J. Cook, editors, *Advanced Methods for Knowledge Discovery from Complex Data*, Advanced Information and Knowledge Processing, pages 95–121. Springer, Berlin/Heidelberg, Germany, 2005. Copyright Springer-Verlag Berlin Heidelberg 2005. Reprinted with kind permission of Springer Science and Business Media.
- Thomas Gärtner, John W. Lloyd, and Peter A. Flach. Kernels

- and distances for structured data. *Machine Learning*, 57(3):205–232, 2004. Reprinted with kind permission of Springer Science and Business Media.
- Tamás Horváth, Thomas Gärtner, and Stefan Wrobel. Cyclic pattern kernels for predictive graph mining. In Ronny Kohan, Johannes Gehrke, William DuMouchel, and Joydeep Ghosh, editors, *Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 158–167, New York, NY, USA, August 2004. ACM. Copyright 2004 ACM 1-58113-888-1/04/0008. Reprinted with kind permission of ACM.
 - Thomas Gärtner. A survey of kernels for structured data. *SIGKDD Explorations*, 5(1):49–58, 2003. Reprinted with kind permission of ACM.
 - Thomas Gärtner, Peter A. Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In Bernhard Schölkopf and Manfred K. Warmuth, editors, *Proceedings of the 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop*, volume 2777 of *Lecture Notes in Computer Science*, pages 129–143, Berlin/Heidelberg, Germany, August 2003. Springer. Copyright Springer-Verlag Berlin Heidelberg 2003. Reprinted with kind permission of Springer Science and Business Media.
 - Kurt Driessens, Jan Ramon, and Thomas Gärtner. Graph kernels and Gaussian processes for relational reinforcement learning. *Machine Learning*, 64(1-3):91–119, 2006. Copyright 2004 Kluwer Academic Publishers. Reprinted with kind permission of Springer Science and Business Media.
 - Thomas Gärtner, Kurt Driessens, and Jan Ramon. Graph kernels and Gaussian processes for relational reinforcement learning. In Tamás Horváth and Akihiro Yamamoto, editors, *The 13th International Conference on Inductive Logic Programming*, volume 2835 of *Lecture Notes in Computer Science*, pages 146–163, Berlin/Heidelberg, Germany, 2003. Springer. Copyright Springer-Verlag Berlin Heidelberg 2003. Reprinted with kind permission of Springer Science and Business Media.
 - Thomas Gärtner, John Lloyd, and Peter Flach. Kernels for structured data. In Stan Matwin and Claude Sammut, editors, *The Twelfth International Conference on Inductive Logic Programming*, volume 2583 of *Lecture Notes in Computer Science*, pages 66–

83, Berlin/Heidelberg, Germany, July 2002. Springer. Copyright Springer-Verlag Berlin Heidelberg 2002. Reprinted with kind permission of Springer Science and Business Media.

- Thomas Gärtner, Peter A. Flach, Adam Kowalczyk, and Alex J. Smola. Multi-instance kernels. In Claude Sammut and Achim Hoffmann, editors, *Proceedings of the 19th International Conference on Machine Learning*, pages 179–186, San Francisco, CA, USA, July 2002. Morgan Kaufmann. Copyright Morgan Kaufmann. Reprinted with kind permission of Elsevier.