1 Introduction

Katharina Morik
Jian-Jia Chen

Abstract: An enormous amount of data is constantly being produced around the world, both in the form of large volume as in that of large velocity. Turning the data into information requires many steps of data analysis: methods for filtering and cleaning the data, joining heterogeneous sources, extracting and selecting features, summarizing and aggregating the data, learning predictions, estimating the uncertainties of the learned model and monitoring the model fitness in its deployment. All these processes need to scale up, whether the long analysis workflows are integrated into a deep learning architecture, or not. The data ecosystems no longer allow us to take von Neumann architecture for granted where only compilers or application systems address hardware issues. Specialized architectures for accelerating machine learning have been developed, and machine learning algorithms have been tailored to novel computer architectures. Both trends are aiming at efficiency, in particular the efficient use of given resources: the real time of execution, the amount of energy, memory and communication. In the struggle for sustainability resource restrictions are of utmost importance. Energy consumption in particular receives considerable attention. We believe that resource efficiency cannot be achieved by better machine learning algorithms or by better hardware architectures alone. It demands the smart combination of hardware and algorithms.

This chapter introduces the fundamentals of machine learning under resource constraints. Resource-aware machine learning is a new and important research field. It is motivated by the following three issues.

- The *resource constraints*, regarding energy consumption, memory requirements, real-time processing, and communication, are to be inspected and investigated under a large diversity of scientific viewpoints.
- The trend towards the *Internet of Things (IoT)* and the many data producing devices like cyber-physical systems or *embedded systems* pose a challenge to data processing. It has led to the programming paradigms of distributed analysis and federated analysis, with data summaries or compression as hot topics.
- The integration of *machine learning and modern hardware* has started to raise international awareness and research efforts.

We want to describe the new field and highlight the contributions of the Collaborative Research Center (CRC) 876 to creating it. The topical overview of machine learning

https://doi.org/10.1515/9783110785944-001

under resource constraints is divided into three sections. First, we discuss research on embedded systems and sustainability in Section 1.1. Then, we focus on machine learning and its energy consumption (Section 1.2). Section 1.3 considers approaches to reducing another important resource: the memory requirements of machine learning. Finally, in Section 1.4, we give an overview of the chapters of this book, which follow the steps of the data analysis process (Section 1.4).

1.1 Embedded Systems and Sustainability

Efficient and high-speed computing has always played a central role in the innovation of information and communication technology (ICT). It is rooted in the widely circulated document "First Draft of a Report on EDVAC" (Electronic Discrete Variable Automatic Computer) by John von Neumann [527]. Over decades, the von Neumann architecture, which consists of a processor unit, a control unit, a memory unit, and input/output peripherals, has been used to efficiently execute programs.

Although ICT has enabled many applications with a high impact on human society, more and more electricity is consumed worldwide. The growth of ICT also has a global impact on the sustainability of the electricity and CO2 footprint worldwide. It is projected that by 2030 ICT will account for 7 % and 20 % of global demand under the optimistic and expected estimated scenarios, respectively [18]. Hence, hardware and software researchers and engineers cannot simply ignore energy efficiency when evaluating their systems and workloads. With Moore's law and Dennard scaling, the improvement of clock frequency of central processing units (CPUs) continued over decades until roughly 2005–2007. Nowadays, the transistor counts in integrated circuits are still growing, but the frequency improvement has ceased as power consumption and thermal dissipation have become the scaling bottleneck.

The discontinuation of Dennard scaling has resulted in the boosting of application-specific hardware accelerators in modern computers to perform efficient and high-speed computing. When Graphics Processing Units (GPUs) were introduced (in 1999 by Nvidia), they were only designed to accelerate the rendering of graphics. Today, application-specific GPUs have become general-purpose vector processors. For machine learning algorithms, specific accelerators include Google's Tensor Processing Units (TPUs) and Apple's Neural Engines. Until the late 1980s, information processing could only be performed on large mainframe computers. Later, the innovation of system integration and technology miniaturization enabled *embedded systems*, i.e., information processing embedded in enclosing products.

Nowadays, embedded systems are pervasive in human society and are widely used in cars, trains, planes, telecommunication, fabrication, ambient intelligence, and decision making. Such embedded systems typically interact with the physical environment to collect information and/or control/influence the physical environment. They share certain common characteristics and have to adhere to certain resource

constraints, independent of the application area. Embedded systems are the core of many innovations, such as cyber-physical systems (CPS), Internet of Things (IoT), and Industry 4.0.

The pervasiveness of embedded systems and sensors contributes to the *big data* computing paradigm, in which data is collected and processed in the cloud. However, transferring data to the cloud consumes time and energy and may not be feasible due to privacy concerns. To address such issues, edge computing, in which embedded edge-nodes process their data locally and potentially share an abstracted model among each other, is an emerging computing paradigm. Such a paradigm shift is also motivated by privacy and security concerns and pushed by governmental policies, such as the California Consumer Privacy Act and the European Union's General Data Protection Regulation, GDPR, which disallow sending/storing sensitive user data to central servers. For example, Gaia-X is an initiative to establish an ecosystem for the next generation of data infrastructure complied with GDPR.

Such a paradigm shift is also driven by the advances of IoT and embedded devices. The annual DataSphere and StorageSphere forecasts published by International Data Corporation (IDC) in 2021 show that "IoT data (not including video surveillance cameras) is the fastest-growing data segment, followed by social media."[1] According to Statista,[2] the number of IoT devices will reach 25.44 billion in 2030.

Embedded systems and IoT devices do not just imply that the computation power is insufficient. Furthermore, they are typically subject to stringent resource constraints due to the design optimization for *resource efficiency* without sacrificing dependability. Specifically, their energy consumption needs to be particularly small. One study [282] analyzes the tradeoff between performance, measured as MobileNet v1 throughput, and the carbon footprint of mobile devices from Google, Huawei and Apple. They concluded that "from 2017 to 2019, software and hardware optimizations primarily focused on maximizing performance, overlooking the growth trend of carbon footprint." [282]

Under resource constraints, memory can be critical both for the code size and the run-time stack size since larger on-chip memory capacity generally leads to higher cost and higher energy/power consumption. Nowadays, the speed of off-chip memories is much slower than that of processors, resulting in the *memory wall* problem. In response, memory hierarchy has been developed in the last decades to enable the illusion that a large memory capacity can be created without significantly losing efficiency. Under such a scheme, modern embedded processors may have either a hardware-managed cache or a software-managed scratchpad memory (SPM), which can be utilized for performance and energy improvement by exploiting temporal and spatial locality.

Under von Neumann architecture, data movement between the physically-separating processing and memory units can be a performance bottleneck for both

1 https://www.idc.com/getdoc.jsp?containerId=prUS47560321.

2 https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/.

energy consumption and performance. Such a bottleneck can be avoided by considering hardware, that offers processing capabilities so that the data resides without the need to move it. This includes Logic-in-Memory (LiM) and Processing-in-Memory (PiM). LiM can be achieved by realizing Boolean logic (e.g., XNOR, NAND, etc.) using both conventional CMOS [8] and emerging beyond-CMOS [535] technologies. PiM can be achieved by exploiting the memory as a crossbar array for efficient vector-matrix operations. For example, TSMC has recently demonstrated an application-specific integrated circuit (ASIC) chip at the 22 nm node, which offers an SRAM-based full-precision PiM macro [138]. In January 2022, Samsung published a crossbar array of spin-transfer-torque magnetoresistive random-access memory (MRAM) for in-memory computing [352].[3]

In summary, edge computing is considered a key pillar to support artificial intelligence and machine learning in pushing our societies to an unprecedented technological revolution. In view of the above discussions, embedded system designers must understand how machine learning algorithms work and machine learning algorithm designers must understand how the underlying hardware can be efficiently utilized for executing machine learning algorithms.

In this book, we inspect cooperative work that aims at resource efficiency for a sustainable future. Focusing on embedded systems, the contributions in Chapter 6 discuss hardware-aware machine learning, including learning on FPGAs in Section 6.1, optimizing the learning on multicore systems in Sections 6.4 and 6.3 and processor-specific transformations in Section 6.2. Furthermore, memory awareness is investigated in Chapter 7, covering memory footprint reduction in Section 7.1, machine learning based on emerging memories (potentially beyond the classic von Neumann architecture) in Section 7.2, and cache-friendly machine learning in Section 7.3.

When devices are connected, communication, synchronization, and offloading are essential. With this in mind, effective synchronization with resource sharing, communication with potential failures, and probabilistic timing information are investigated in Section 8.1. Section 8.2 considers bandwidth limitations of different execution models and coprocessor-accelerated optimization.

The next sections focus on machine learning and introduce the Chapters on energy- and memory-saving machine learning methods.

1.2 The Energy Consumption of Machine Learning

Machine learning has always been a central part of Artificial Intelligence (AI). Already Allen Turing argued that programming a computer cannot scale up to the performance

3 https://news.samsung.com/global/samsung-demonstrates-the-worlds-first-mram-based-in-memory-computing.

that a learning machine can achieve [673]. According to the AI index of Stanford University in 2022, publications in pattern recognition and machine learning have more than doubled since 2015. Other areas strongly influenced by deep learning, such as computer vision, data mining, and natural language processing have seen smaller increases.[4]

The classes of algorithms in machine learning are too many to be characterized, here. The field of machine learning covers a wide range. A bird's-eye view sees different *approaches*: geometric (e.g., decision trees, support vector machines), probabilistic (e.g., probabilistic graphical models, Bayesian models), combinatoric (k-means, frequent sets), logic (e.g., inductive logic programming), reinforcement models (e.g., bandit models), and neural networks (deep learning).

At a more technical level, we see *learning tasks* that specify the formal basis of machine learning methods, defining what is learned (classification, regression, probability density, cluster model), from what it is learned (real-valued vectors, time series, categorical data, count data), under which constraints (quality criteria, streaming/online, distributed). As is common in statistics, the term "model" is used not only for the class of possible learning results given the types of input, output and quality criteria, but also for a particular instance, the learning result.

Combining approaches and learning tasks, we see the areas of machine learning. All of them are growing. Several algorithms have been developed within these areas. Many of them use algorithms for underlying inner procedures or compose learning methods using *building blocks* such as kernel functions, matrix factorization, optimization, regularization, or sampling. Investigating machine learning at all levels, from the models to hardware architectures, is the particular profile of the research that has been undertaken by the Collaborative Research Center 876 (CRC 876).

Today, resource restrictions are of utmost importance. Energy consumption in particular receives considerable attention. Machine learning is put to good use in order to save energy for sustainability. Google considers the application of DeepMind's machine learning to its data centers to be its most important application. The energy used for cooling could be reduced by up to 40 % through machine learning.[5] Machine learning algorithms themselves are enhanced for low energy demands. One of the invited talks at the International Conference on Machine learning (ICML) 2018—Max Welling's "Intelligence per Kilowatt-hour" supports our approach to joining embedded systems and machine learning research. Its author said, "The next battleground in AI might well be a race for the most energy efficient combination of hardware and algorithms." CRC 876 has contributed in exactly to this race. The results of its work are reported here. In the following, we refer to the approaches described in this book concerning machine learning and the resources of energy and memory.

4 aiindex.stanford.edu.

5 https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-40/.

In general, *green computing* and *sustainable computing* have received considerable interest. On the one hand, machine learning decreases the ecological footprint of processes in many applications (see, e.g., [418]). On the other hand, machine learning itself might use tremendous amounts of energy. This is particularly true for big language models such as GPT-3. A careful analysis by Patterson et al. [556] compares the CO_2 footprint of different natural language learners. The relation of the run-time of training, the number of processors and their average power consumption together with the power usage efficiency of the particular computing center where the machine learning algorithm is executed gives an estimated kWh. This, in turn, is used to calculate tons of CO_2 equivalents: $tCO_2e = kWh \times kgCO_2e$ per $kWh : 1000$, where CO_2e accounts for carbon dioxide and other greenhouse gases, as opposed to CO_2 which only covers dioxide. GPT-3 uses 552.1 t of CO_2 for training its 175 billion parameters, when using the V100 processor. The mere energy consumption is 1287 mWh for its 14.8 days of training. In general, the ecological footprint of machine learning should be reported [305]. Tools for estimating the energy consumption of particular machine learning algorithms have been implemented for regular computing clusters [650].

In contrast to the research efforts regarding deep learning on large data-centers, investigating the energy consumption of small devices has not yet received enough attention. However, as we have shown above, the Internet of Things (IoT) connects billions of small devices and produces extremely large amounts of data. Their energy consumption needs to be particularly small. For an embedded system that is not plugged into the grid, the availability of energy is a critical constraint for its lifetime. Even for an embedded system plugged into the grid, the cost of energy due to increased computing performance can be critical. Unnecessary energy consumption should also be avoided to extend the lifetime of the embedded system. The power awareness and energy efficiency of information processing on devices of the IoT are important for sustainable computing. In this book, we focus primarily on small devices.

1.2.1 Measuring Energy Consumption

Investigating energy efficiency requires measuring energy consumption. Measuring the true energy consumption directly is difficult because of noise sampling and the need to use minimal resources for the sensing itself. The hardware-integrated sensing instruments are often not precise enough for determining the energy consumed by software running on an embedded system. An easy-to-use system for direct energy sensing and an energy model for ARM processors based on linear regression have been developed [105, 106]. Extremely restricted are the ultra-low-power devices that are used in logistics, e.g., devices attached to a container. Based on reliable measurements, the energy harvesting of devices with photovoltaic elements can be realized such that the operating time is enhanced. Section 2.2 of this book shows the PhyNet testbed for energy neutral sensor networks. A batteryless system, its indoor solar harvesting, and

energy measurements are presented in Section 2.3. There, even the implementation of a lightweight deep learning algorithm is included. Predicting the power consumption in a communication network is particularly challenging. Section 9.2 presents methods for modeling power consumption of embedded devices for different wireless communication technologies including a machine learning-based method for estimating the transmit power from the available performance indicators, like strength and quality of the received signal.

1.2.2 Different Processors

The energy consumption of different processors varies greatly. Using the example of Quadratic Unconstrained Binary Optimization (QUBO), implemented by an evolutionary algorithm (EA), showed a stable order of magnitude of energy consumption over diverse data-sets and parameter settings [510]. We indicate the numbers here because, in general, such information needs to be given in scientific papers on machine learning. Moreover, the Watt figures found in the QUBO experiments show the typical pattern of magnitudes:

- Field Programmable Gate Arrays (FPGAs), 10^0W,
- CPU (Intel Core i7-9700K)10^1W,
- GPU (Nvidia GEFORCE RTX 2080 Ti)10^2W,
- QA (Quantum Annealer, IBM)10^4W.

Our compute cluster consumes on average 6.25 kWh, or in the order of magnitude of 10^3W. Of course, the particular energy consumption depends also on the number of variables (here: 1024) and the parameters of the EA, e.g. the number of children in each generation. However, the advantage of the FPGA can be estimated already on the basis of these numbers. Since the compute cluster consumes 10^3 times as much energy as does the FPGA, it would need to solve the learning problem in less time—here in the order of 10^{-9} seconds—to use about the same energy, and this is not realistic. Another example studies different implementations of applying a learned Decision Tree (DT) model [110]. The classification is implemented in two different ways. One implements the algorithm as is, the other unfolds the tree into if-else structures, aka compilation. Energy consumption is then measured for FPGA (Xilinx Artix-7 Z-7020 FPGA with 53 200 lookup tables, 106 400 Flip-Flops (FF) in total combined with 4.9 MB block ram and 220 DSP units), and an ARM processor (Cortex-A9 with 666 MHz, 512 MB DDR RAM and 512 kB cache). Each learned tree contains an average of 1349 nodes and roughly 675 different paths from the root node to a leaf node. Throughput is measured as elements per millisecond, energy consumption as nanoJoule per element. The native implementation on FPGA uses 0.008 W or 6.84 nanoJoule per element to classify, on the ARM processor 1.53 W or 105.5 nanoJoule per element to classify. The unfolded tree uses

on FPGA 0.068 W or 45.95 nanoJoule per element to classify, and on the ARM processor 1.53 W or 52.76 nanoJoule per element to classify.

This general ranking of the order of energy consumption makes FPGAs attractive for machine learning. In this book, Section 6.1 investigates reconfigurable multilayer perceptron training on FPGAs and compares it with the PyTorch implementation. Furthermore, Section 6.1 explores FPGA implementation of a Multilayer Perceptron (MLP), a fundamental neural network structure for machine learning. FPGAs are especially advantageous for deep learning because they support customized data types, where GPUs support only a limited number of data types.

1.2.3 Reduced Run-Time and Real-time Processing

Many approaches to reducing the energy consumption of machine learning reduce the run-time of a learning method. We take into account the execution of learning programs and learned models not only regarding time complexity, but also regarding real-time ability. Since many embedded systems are integrated into large products that interact with the physical world, timeliness is an important issue. If the results are delivered too late, they may have become useless. The computing of summaries “on the fly”, as presented in Section 3.1, is designed to save memory and energy. In general, algorithms for data streams require fast computing and memory reduction, as we discuss below.

The link between energy and memory reduction also becomes clear in the approach to graph deep learning in Section 4.3, where a general message passing is scaled up for arbitrarily large graphs. The remarkable speed-up of clustering run-time as demonstrated in Section 5.1 certainly saves energy as well. Exploiting parallelism, even for non-uniform workloads, is the key in Section 6.2 to reducing the run-time and increasing the data throughput for database query execution. Section 6.3 describes extreme multi-core computation, which exploits the independence of training for several thousand labels. It trains each class versus all others using thousands of cores, each one learning to predict one of the many classes. Along with the number of cores, the hardware-aware parallel training solvers speed up until a saturation is reached and the speedup scales only sublinearly.

Reducing run-time through an adaptive scheduling brings together the multi-core computing architecture and machine learning. Section 6.4 examines the optimization of the execution of diverse machine learning algorithms for parallel execution on a multi-core architecture. The optimization itself also uses machine learning, namely, the Bayesian model-based optimization . The Resource-aware Model Based Optimization (RAMBO) framework saves energy through the run-time reduction.

1.2.4 Minimizing Energy Consumption of Machine Learning Processes

If minimizing the energy of machine learning processes builds upon the analysis of the algorithms, statistical guarantees can be given. Exponential families are a model of learning that covers many learning tasks, e.g., the estimation of probability density as it is used by, say, topic models, or the prediction of the maximally likely state as it is used by naive Bayes or conditional random fields. A careful analysis of learning models may lead to running of very complex machine learning tasks on very limited and even ultra-low energy devices. This book offers such an approach in Section 9.1, which describes the Integer Markov Random Fields (IntMRF) along with their theoretical foundations. Note that it is the underlying model class that is restricted to the integers; it is not just a restriction of the state space to integers. Here, the state space may be a random discrete space without any additional constraints. The reduced run-time and energy savings are due to the cheaper operations. The novel bit-length propagation algorithm (BL-Prop) allows computing using integers only, i.e. real numbers are not quantized afterwards, but all the learning processing uses only integers. In addition to previous work ([567]), Section 9.1 introduces the novel numerical optimization method IntGD for convex objective functions. It is based on an accelerated proximal algorithm for non-smooth and non-convex penalty terms. For integer gradients computed via BL-Prop, IntGD is guaranteed to deliver a pure integer learning procedure in which the final parameter vector as well as all intermediate results are integers. Integer Markov random fields are almost as expressible as real-valued ones are, but can be executed on an ultra-low-power device that does not offer floating-point operations [570].

As we have seen, there are multiple ways to reduce the energy consumption of machine learning: developing algorithms for more energy efficient processors (FPGAs), tailoring machine learning algorithms, optimizing their execution for a reduced run-time, and even developing novel learning algorithms designed to save energy.

1.3 Memory Demands of Machine Learning

1.3.1 Deep Learning

Deep learning challenges the GPU memory due to its many hyperparameters, tensor alignment, particular convolution algorithms, and operator scheduling. In a detailed analysis of 4960 failed deep learning runs, Yanjie Gao and colleagues found 8.8 % of them were caused by the exhaustion of GPU memory [242]. They then developed an estimate for the GPU memory needs of deep learning models. In this book, the memory demands of Graph Neural Networks (GNNs) are part of the work that is presented in Sections 4.2 and 4.3. The usual mini-batch training becomes difficult in GNNs because of the interdependency of neighboring nodes. The exponential growth of the graphs has been shown in [455], which proposes sampling of edges. A more general solution

for diverse GNN architecture is presented in Section 4.3. The novel GNN AutoScale framework of message passing succeeds in making GNN applicable even in a streaming setting, since for a single epoch and layer, each edge is processed just once.

The quantization of deep learning results in binary values of weights and activations, reducing the memory consumption drastically [325]. Binarized Neural Networks (BNN) offer more lightweight processing. Combining machine learning and computer architecture work has led to BNN on FPGAs for fast inference on very large streaming data from astroparticle physics [112]. A further step towards the close interplay of algorithms and hardware is to take into account modern memory technologies. Again, we see the close relationship between energy consumption and memory architecture in the case of approximate or non-volatile memories that reduce the energy consumption but increase the bit error rate. For BNNs, bit flips in the weights or the activation values of the network decrease the accuracy of the model. How many bit errors can be tolerated at the hidden layers? The idea of max margin optimization, developed for Support Vector Machines (SVM) [680], inspired a formulation of a bit error tolerance metric that could be inserted into the BNN training [113]. Machine learning anticipates hardware errors and thus produces a robust learned model for the energy-saving computing architecture. Section 7.2 explains this approach of reducing the bit error rate within the training of a BNN in more detail.

1.3.2 Summaries and Clustering

Data summary or aggregation is necessary in order to learn from distributed sensor streams. Sketching or sampling has been theoretically investigated for clustering data streams [70, 103]. Coresets and sketches summarize data such that they can be analyzed by any learning algorithm and they can deliver approximately the same result as would result from training on the full dataset [516]. Section 3.2 analyzes coresets and sketches for distributed and streaming data. The analysis covers approaches to Bayesian and generalized linear regression. A sparse subspace of the original high-dimensional data space is proven to be sample-efficient. The data reduction saves not only memory but also run-time and energy demand.

Summaries with a fixed memory size are often developed using submodular functions. For video summarization, a submodular set function could be optimized subject to privacy constraints [495]. In 3.1, sieve streaming with fixed-size memory is enhanced for sampling the most informative observations “on the fly”. In addition to saving resources, the novel ThreeSieves algorithm offers summaries for human interactive data exploration.

Unsupervised learning partitions data in many different ways. This book presents the clustering of graph data in Section 5.1 and of curves in Section 5.2. The scalability of hierarchical agglomerative clustering is considerably enhanced by the BETULA algorithm in Section 5.3.

Some problems occur as building blocks of learning algorithms. Matrix factorization is one of them. An approach of Binary and Boolean matrix factorization that is robust with respect to noise is presented in Section 5.4. It uses proximal gradient descent optimization and allows overlapping clusters.

Another one is the max dicut problem: partitioning of a directed graph into two subsets such that the sum of the edge-weights between the two subsets is maximized. Section 4.4 investigates this problem for parallel algorithms, that scale for very large graphs.

1.3.3 Executing Machine Learning

On the level of programming languages and operating systems, smart resource utilization reduces the memory footprint [388]. Moreover, the dynamic sharing of memory can be optimized [596]. Section 7.1 presents a memory management layer between the R interpreter and the operating system that reduces the memory footprint by allocating memory only to pages in the memory that are required.

Decision trees (DTs), although one of the earliest machine learning algorithms, still pose research challenges. Training several thousand DTs leads to millions of decision nodes that must be stored in memory and processed in order to apply the learned model to new data. Hence, inferences using DT ensembles demand a smart memory layout. Cache memory moderates between the main memory and the processor. Preventing cache misses requires a well-designed memory layout. Section 7.3 offers an implementation that optimizes the memory layout while preserving the original ensembles' accuracy. A code generator automatically adapts to underlying architectures.

1.3.4 Regularization and Reparametrization

Regarding models of learning, the reduction of memory demand has been investigated for the exponential families. The memory consumption of Markov Random Fields (MRFs) is dominated by the size of its parameter vector. Since each parameter is usually accessed multiple times during inference, they should be stored in a cache memory. The key to compression is regularization and reparametrization, which exploit redundancies in the true parameters. The general idea can be applied to discrete Markov random fields and to multivariate Gaussian models [575]. Section 4.1 presents spatio-temporal random fields. They model spatial networks as graphs and connected layers of these graphs as temporal relations. A piecewise linear reparametrization of the parameters of a clique (a part of the graph) is weighted by a decay vector, and the full model is weighted by a corresponding decay matrix. In spatio-temporal random fields, it is assumed that value changes at nodes do not change in sudden jumps over time. The

reparametrization of spatio-temporal random fields based in this assumption is proven to be universal, i.e. it is a bijection.

1.4 Structure of this Book

The book covers contributions from machine learning and embedded systems and includes, in addition, algorithmic and database research that supports the overall goal of resource-constrained data analysis. Its structure follows that of the workflow. It starts with data of different kinds. Then it moves to executing machine learning and the particular resource constraints, namely memory, communication, and energy. Each chapter offers an introductory summary of its sections.
The book is organized as follows:

- This book starts in Chapter 2 with the *data collection* of embedded system deployments. Section 2.1 presents the system kCQL for collecting complex operating system data. kCQL acquires and combines event streams and system states of operating systems while maintaining low overheads. A physical sensor network testbed is presented in Section 2.2 that can be used for large-scale energy accounting, position tracking, application testing, and system data collections. Modeling, analysis, calibration, and evaluation of batteryless in-door energy harvesting systems are presented in Section 2.3.
- Chapter 3 considers *data streams* in resource-constrained embedded systems. Regarding summary extraction from streams, Section 3.1 presents an insertion-only data stream learning algorithm based on maximizing submodular functions. Section 3.2 covers a brief technical introduction to coresets and sketches and highlights their importance for the design of data stream algorithms.
- Chapter 4 presents methods and techniques to learning *models for structured data* with resource-awareness. In Section 4.1, a probabilistic learning model of spatio-temporal random fields is introduced, which reduces memory consumption without loss of the accuracy through universal reparameterization. In Section 4.2, the Weisfeiler-Leman algorithm is connected to learning methods such as graph kernels and Graph Neural Networks (GNNs). In Section 4.3, a unified and scalable framework for message passing in GNNs is proposed. Section 4.4 presents algorithms to compute cuts in directed graphs with high quality, which scales well in shared memory. Section 4.5 presents a new technique based on GNNs to search for scientific papers, utilizing key mathematical formulas instead of key words.
- Chapter 5 considers *clustering* in four complementary sections. Section 5.1 exploits the sparseness of data for reducing memory requirements and run-time. Section 5.2 handles sequences of points using the Fréchet distance and details an approximation for their clustering. Section 5.3 characterizes methods for hierarchical clustering that are well suited for streaming data processing on edge devices with limited

resources. Section 5.4 offers a novel optimization subject to binary constraints for matrix factorization, a method that is entailed in many learning algorithms.

- Chapter 6 deals with the heterogeneity of *execution platforms* of embedded systems. Section 6.1 presents the acceleration of learning neural networks on Field-Programmable Gate Arrays (FPGAs). Parallel executions utilizing graphics processors (GPU) for efficient database query processing and multicore systems for accelerating extreme multi-label classification are presented in Section 6.2 and Section 6.3, respectively. The RAMBO framework that can efficiently optimize machine learning models on heterogeneous distributed systems is discussed in Section 6.4.
- Chapter 7 presents optimizations of machine learning algorithms with respect to *memory*. Section 7.1 demonstrates that the memory footprint can be effectively reduced by leveraging application-specific knowledge. Section 7.3 proactively optimizes the memory layout in the implementation of the machine learning model to favor the underlying cache memories with a probabilistic perspective. Furthermore, Section 7.2 presents how learning models can accurately process in environments with unreliable memories if we take bit errors into account during machine learning training.
- Chapter 8 considers embedded systems under *communication* constraints. It covers synchronization with resource sharing, communication with potential failures, and probabilistic timing information in Section 8.1, and discusses bandwidth limitations of different execution models and coprocessor-accelerated optimization in Section 8.2.
- Chapter 9 is about *energy efficiency*. Section 9.1 shows how to reduce the power consumption of complex learning models such as Markov random fields through integer-only operations. The novel Bit-Length Propagation (BL-Prop) and integer gradient descent (IntGD) algorithms can be executed even on ultra-low-power (ULP) micro-controllers. Section 9.2 uses machine learning in order to estimate the power consumption of diverse communication technologies in wireless systems unleashed from the power grid and light-weighted small wearables.

Each chapter and section is self-contained. You may select the chapter or section you want to read by topic or by data flow of data analysis processes. You may want to read an overall chapter or just some sections. Because we have written the book with teaching in mind you can select a number of sections for specialized courses. Of course, we also encourage readers seeking an in-depth understanding of the resource-efficient combination of hardware and machine learning algorithms to read the entire book!