1 Introduction

Computer science as an academic discipline began in the 1960's. Emphasis was on programming languages, compilers, operating systems, and the mathematical theory that supported these areas. Courses in theoretical computer science covered finite automata, regular expressions, context-free languages, and computability. In the 1970's, the study of algorithms was added as an important component of theory. The emphasis was on making computers useful. Today, a fundamental change is taking place and the focus is more on a wealth of applications. There are many reasons for this change. The merging of computing and communications has played an important role. The enhanced ability to observe, collect, and store data in the natural sciences, in commerce, and in other fields calls for a change in our understanding of data and how to handle it in the modern setting. The emergence of the web and social networks as central aspects of daily life presents both opportunities and challenges for theory.

While traditional areas of computer science remain highly important, increasingly researchers of the future will be involved with using computers to understand and extract usable information from massive data arising in applications, not just how to make computers useful on specific well-defined problems. With this in mind we have written this book to cover the theory we expect to be useful in the next 40 years, just as an understanding of automata theory, algorithms, and related topics gave students an advantage in the last 40 years. One of the major changes is an increase in emphasis on probability, statistics, and numerical methods.

Early drafts of the book have been used for both undergraduate and graduate courses. Background material needed for an undergraduate course has been put in the appendix. For this reason, the appendix has homework problems.

Modern data in diverse fields such as information processing, search, and machine learning is often advantageously represented as vectors with a large number of components. The vector representation is not just a book-keeping device to store many fields of a record. Indeed, the two salient aspects of vectors: geometric (length, dot products, orthogonality etc.) and linear algebraic (independence, rank, singular values etc.) turn out to be relevant and useful. Chapters 2 and 3 lay the foundations of geometry and linear algebra respectively. More specifically, our intuition from two or three dimensional space can be surprisingly off the mark when it comes to high dimensions. Chapter 2 works out the fundamentals needed to understand the differences. The emphasis of the chapter, as well as the book in general, is to get across the intellectual ideas and the mathematical foundations rather than focus on particular applications, some of which are briefly described. Chapter 3 focuses on singular value decomposition (SVD) a central tool to deal with matrix data. We give a from-first-principles description of the mathematics and algorithms for SVD. Applications of singular value decomposition include principal component analysis, a widely used technique which we touch upon, as well as modern

applications to statistical mixtures of probability densities, discrete optimization, etc., which are described in more detail.

Exploring large structures like the web or the space of configurations of a large system with deterministic methods can be prohibitively expensive. Random walks (also called Markov Chains) turn out often to be more efficient as well as illuminative. The stationary distributions of such walks are important for applications ranging from web search to the simulation of physical systems. The underlying mathematical theory of such random walks, as well as connections to electrical networks, forms the core of Chapter 4 on Markov chains.

One of the surprises of computer science over the last two decades is that some domain-independent methods have been immensely successful in tackling problems from diverse areas. Machine learning is a striking example. Chapter 5 describes the foundations of machine learning, both algorithms for optimizing over given training examples, as well as the theory for understanding when such optimization can be expected to lead to good performance on new, unseen data. This includes important measures such as the Vapnik-Chervonenkis dimension, important algorithms such as the Perceptron Algorithm, stochastic gradient descent, boosting, and deep learning, and important notions such as regularization and overfitting.

The field of algorithms has traditionally assumed that the input data to a problem is presented in random access memory, which the algorithm can repeatedly access. This is not feasible for problems involving enormous amounts of data. The streaming model and other models have been formulated to reflect this. In this setting, sampling plays a crucial role and, indeed, we have to sample on the fly. In Chapter 6 we study how to draw good samples efficiently and how to estimate statistical and linear algebra quantities, with such samples.

While Chapter 5 focuses on supervised learning, where one learns from labeled training data, the problem of unsupervised learning, or learning from unlabeled data, is equally important. A central topic in unsupervised learning is clustering, discussed in Chapter 7. Clustering refers to the problem of partitioning data into groups of similar objects. After describing some of the basic methods for clustering, such as the k-means algorithm, Chapter 7 focuses on modern developments in understanding these, as well as newer algorithms and general frameworks for analyzing different kinds of clustering problems.

Central to our understanding of large structures, like the web and social networks, is building models to capture essential properties of these structures. The simplest model is that of a random graph formulated by Erdös and Renyi, which we study in detail in Chapter 8, proving that certain global phenomena, like a giant connected component, arise in such structures with only local choices. We also describe other models of random graphs.

Chapter 9 focuses on linear-algebraic problems of making sense from data, in particular topic modeling and non-negative matrix factorization. In addition to discussing well-known models, we also describe some current research on models and algorithms with provable guarantees on learning error and time. This is followed by graphical models and belief propagation.

Chapter 10 discusses ranking and social choice as well as problems of sparse representations such as compressed sensing. Additionally, Chapter 10 includes a brief discussion of linear programming and semidefinite programming. Wavelets, which are an important method for representing signals across a wide range of applications, are discussed in Chapter 11 along with some of their fundamental mathematical properties. The appendix includes a range of background material.

A word about notation in the book. To help the student, we have adopted certain notations, and with a few exceptions, adhered to them. We use lower case letters for scalar variables and functions, bold face lower case for vectors, and upper case letters for matrices. Lower case near the beginning of the alphabet tend to be constants, in the middle of the alphabet, such as i, j, and k, are indices in summations, n and m for integer sizes, and x, y and z for variables. If A is a matrix its elements are a_{ij} and its rows are $\mathbf{a_i}$. If $\mathbf{a_i}$ is a vector its coordinates are a_{ij}. Where the literature traditionally uses a symbol for a quantity, we also used that symbol, even if it meant abandoning our convention. If we have a set of points in some vector space, and work with a subspace, we use n for the number of points, d for the dimension of the space, and k for the dimension of the subspace.

The term "almost surely" means with probability tending to one. We use $\ln n$ for the natural logarithm and $\log n$ for the base two logarithm. If we want base ten, we will use $\log_{10}$. To simplify notation and to make it easier to read we use $E^2(1-x)$ for $\left(E(1-x)\right)^2$ and $E(1-x)^2$ for $E\left((1-x)^2\right)$. When we say "randomly select" some number of points from a given probability distribution, independence is always assumed unless otherwise stated.