

# Mining of Massive Datasets

Jure Leskovec  
Stanford Univ.

Anand Rajaraman  
Milliway Labs

Jeffrey D. Ullman  
Stanford Univ.

Copyright © 2010, 2011, 2012, 2013, 2014 Anand Rajaraman, Jure Leskovec,  
and Jeffrey D. Ullman

# Contents

<b>1</b>	<b>Data Mining</b>	<b>1</b>
1.1	What is Data Mining? . . . . .	1
1.1.1	Statistical Modeling . . . . .	1
1.1.2	Machine Learning . . . . .	2
1.1.3	Computational Approaches to Modeling . . . . .	2
1.1.4	Summarization . . . . .	3
1.1.5	Feature Extraction . . . . .	4
1.2	Statistical Limits on Data Mining . . . . .	4
1.2.1	Total Information Awareness . . . . .	5
1.2.2	Bonferroni's Principle . . . . .	5
1.2.3	An Example of Bonferroni's Principle . . . . .	6
1.2.4	Exercises for Section 1.2 . . . . .	7
1.3	Things Useful to Know . . . . .	7
1.3.1	Importance of Words in Documents . . . . .	8
1.3.2	Hash Functions . . . . .	9
1.3.3	Indexes . . . . .	10
1.3.4	Secondary Storage . . . . .	11
1.3.5	The Base of Natural Logarithms . . . . .	12
1.3.6	Power Laws . . . . .	13
1.3.7	Exercises for Section 1.3 . . . . .	15
1.4	Outline of the Book . . . . .	15
1.5	Summary of Chapter 1 . . . . .	17
1.6	References for Chapter 1 . . . . .	18
<b>2</b>	<b>MapReduce and the New Software Stack</b>	<b>21</b>
2.1	Distributed File Systems . . . . .	22
2.1.1	Physical Organization of Compute Nodes . . . . .	22
2.1.2	Large-Scale File-System Organization . . . . .	23
2.2	MapReduce . . . . .	24
2.2.1	The Map Tasks . . . . .	25
2.2.2	Grouping by Key . . . . .	26
2.2.3	The Reduce Tasks . . . . .	27
2.2.4	Combiners . . . . .	27

2.2.5	Details of MapReduce Execution . . . . .	28
2.2.6	Coping With Node Failures . . . . .	29
2.2.7	Exercises for Section 2.2 . . . . .	30
2.3	Algorithms Using MapReduce . . . . .	30
2.3.1	Matrix-Vector Multiplication by MapReduce . . . . .	31
2.3.2	If the Vector $v$ Cannot Fit in Main Memory . . . . .	31
2.3.3	Relational-Algebra Operations . . . . .	32
2.3.4	Computing Selections by MapReduce . . . . .	35
2.3.5	Computing Projections by MapReduce . . . . .	36
2.3.6	Union, Intersection, and Difference by MapReduce . . . . .	36
2.3.7	Computing Natural Join by MapReduce . . . . .	37
2.3.8	Grouping and Aggregation by MapReduce . . . . .	37
2.3.9	Matrix Multiplication . . . . .	38
2.3.10	Matrix Multiplication with One MapReduce Step . . . . .	39
2.3.11	Exercises for Section 2.3 . . . . .	40
2.4	Extensions to MapReduce . . . . .	41
2.4.1	Workflow Systems . . . . .	41
2.4.2	Recursive Extensions to MapReduce . . . . .	42
2.4.3	Pregel . . . . .	45
2.4.4	Exercises for Section 2.4 . . . . .	46
2.5	The Communication Cost Model . . . . .	46
2.5.1	Communication-Cost for Task Networks . . . . .	47
2.5.2	Wall-Clock Time . . . . .	49
2.5.3	Multiway Joins . . . . .	49
2.5.4	Exercises for Section 2.5 . . . . .	52
2.6	Complexity Theory for MapReduce . . . . .	54
2.6.1	Reducer Size and Replication Rate . . . . .	54
2.6.2	An Example: Similarity Joins . . . . .	55
2.6.3	A Graph Model for MapReduce Problems . . . . .	57
2.6.4	Mapping Schemas . . . . .	58
2.6.5	When Not All Inputs Are Present . . . . .	60
2.6.6	Lower Bounds on Replication Rate . . . . .	61
2.6.7	Case Study: Matrix Multiplication . . . . .	62
2.6.8	Exercises for Section 2.6 . . . . .	66
2.7	Summary of Chapter 2 . . . . .	67
2.8	References for Chapter 2 . . . . .	69
<b>3</b>	<b>Finding Similar Items</b>	<b>73</b>
3.1	Applications of Near-Neighbor Search . . . . .	73
3.1.1	Jaccard Similarity of Sets . . . . .	74
3.1.2	Similarity of Documents . . . . .	74
3.1.3	Collaborative Filtering as a Similar-Sets Problem . . . . .	75
3.1.4	Exercises for Section 3.1 . . . . .	77
3.2	Shingling of Documents . . . . .	77
3.2.1	$k$ -Shingles . . . . .	77

3.2.2	Choosing the Shingle Size . . . . .	78
3.2.3	Hashing Shingles . . . . .	79
3.2.4	Shingles Built from Words . . . . .	79
3.2.5	Exercises for Section 3.2 . . . . .	80
3.3	Similarity-Preserving Summaries of Sets . . . . .	80
3.3.1	Matrix Representation of Sets . . . . .	81
3.3.2	Minhashing . . . . .	81
3.3.3	Minhashing and Jaccard Similarity . . . . .	82
3.3.4	Minhash Signatures . . . . .	83
3.3.5	Computing Minhash Signatures . . . . .	83
3.3.6	Exercises for Section 3.3 . . . . .	86
3.4	Locality-Sensitive Hashing for Documents . . . . .	87
3.4.1	LSH for Minhash Signatures . . . . .	88
3.4.2	Analysis of the Banding Technique . . . . .	89
3.4.3	Combining the Techniques . . . . .	91
3.4.4	Exercises for Section 3.4 . . . . .	91
3.5	Distance Measures . . . . .	92
3.5.1	Definition of a Distance Measure . . . . .	92
3.5.2	Euclidean Distances . . . . .	93
3.5.3	Jaccard Distance . . . . .	94
3.5.4	Cosine Distance . . . . .	95
3.5.5	Edit Distance . . . . .	95
3.5.6	Hamming Distance . . . . .	96
3.5.7	Exercises for Section 3.5 . . . . .	97
3.6	The Theory of Locality-Sensitive Functions . . . . .	99
3.6.1	Locality-Sensitive Functions . . . . .	99
3.6.2	Locality-Sensitive Families for Jaccard Distance . . . . .	100
3.6.3	Amplifying a Locality-Sensitive Family . . . . .	101
3.6.4	Exercises for Section 3.6 . . . . .	103
3.7	LSH Families for Other Distance Measures . . . . .	104
3.7.1	LSH Families for Hamming Distance . . . . .	104
3.7.2	Random Hyperplanes and the Cosine Distance . . . . .	105
3.7.3	Sketches . . . . .	106
3.7.4	LSH Families for Euclidean Distance . . . . .	107
3.7.5	More LSH Families for Euclidean Spaces . . . . .	108
3.7.6	Exercises for Section 3.7 . . . . .	109
3.8	Applications of Locality-Sensitive Hashing . . . . .	110
3.8.1	Entity Resolution . . . . .	110
3.8.2	An Entity-Resolution Example . . . . .	111
3.8.3	Validating Record Matches . . . . .	112
3.8.4	Matching Fingerprints . . . . .	113
3.8.5	A LSH Family for Fingerprint Matching . . . . .	114
3.8.6	Similar News Articles . . . . .	115
3.8.7	Exercises for Section 3.8 . . . . .	117
3.9	Methods for High Degrees of Similarity . . . . .	118

3.9.1	Finding Identical Items . . . . .	118
3.9.2	Representing Sets as Strings . . . . .	118
3.9.3	Length-Based Filtering . . . . .	119
3.9.4	Prefix Indexing . . . . .	119
3.9.5	Using Position Information . . . . .	121
3.9.6	Using Position and Length in Indexes . . . . .	122
3.9.7	Exercises for Section 3.9 . . . . .	125
3.10	Summary of Chapter 3 . . . . .	126
3.11	References for Chapter 3 . . . . .	128
<b>4</b>	<b>Mining Data Streams</b>	<b>131</b>
4.1	The Stream Data Model . . . . .	131
4.1.1	A Data-Stream-Management System . . . . .	132
4.1.2	Examples of Stream Sources . . . . .	133
4.1.3	Stream Queries . . . . .	134
4.1.4	Issues in Stream Processing . . . . .	135
4.2	Sampling Data in a Stream . . . . .	136
4.2.1	A Motivating Example . . . . .	136
4.2.2	Obtaining a Representative Sample . . . . .	137
4.2.3	The General Sampling Problem . . . . .	137
4.2.4	Varying the Sample Size . . . . .	138
4.2.5	Exercises for Section 4.2 . . . . .	138
4.3	Filtering Streams . . . . .	139
4.3.1	A Motivating Example . . . . .	139
4.3.2	The Bloom Filter . . . . .	140
4.3.3	Analysis of Bloom Filtering . . . . .	140
4.3.4	Exercises for Section 4.3 . . . . .	141
4.4	Counting Distinct Elements in a Stream . . . . .	142
4.4.1	The Count-Distinct Problem . . . . .	142
4.4.2	The Flajolet-Martin Algorithm . . . . .	143
4.4.3	Combining Estimates . . . . .	144
4.4.4	Space Requirements . . . . .	144
4.4.5	Exercises for Section 4.4 . . . . .	145
4.5	Estimating Moments . . . . .	145
4.5.1	Definition of Moments . . . . .	145
4.5.2	The Alon-Matias-Szegedy Algorithm for Second Moments . . . . .	146
4.5.3	Why the Alon-Matias-Szegedy Algorithm Works . . . . .	147
4.5.4	Higher-Order Moments . . . . .	148
4.5.5	Dealing With Infinite Streams . . . . .	148
4.5.6	Exercises for Section 4.5 . . . . .	149
4.6	Counting Ones in a Window . . . . .	150
4.6.1	The Cost of Exact Counts . . . . .	151
4.6.2	The Datar-Gionis-Indyk-Motwani Algorithm . . . . .	151
4.6.3	Storage Requirements for the DGIM Algorithm . . . . .	153

4.6.4	Query Answering in the DGIM Algorithm . . . . .	153
4.6.5	Maintaining the DGIM Conditions . . . . .	154
4.6.6	Reducing the Error . . . . .	155
4.6.7	Extensions to the Counting of Ones . . . . .	156
4.6.8	Exercises for Section 4.6 . . . . .	157
4.7	Decaying Windows . . . . .	157
4.7.1	The Problem of Most-Common Elements . . . . .	157
4.7.2	Definition of the Decaying Window . . . . .	158
4.7.3	Finding the Most Popular Elements . . . . .	159
4.8	Summary of Chapter 4 . . . . .	160
4.9	References for Chapter 4 . . . . .	161
<b>5</b>	<b>Link Analysis</b>	<b>163</b>
5.1	PageRank . . . . .	163
5.1.1	Early Search Engines and Term Spam . . . . .	164
5.1.2	Definition of PageRank . . . . .	165
5.1.3	Structure of the Web . . . . .	169
5.1.4	Avoiding Dead Ends . . . . .	170
5.1.5	Spider Traps and Taxation . . . . .	173
5.1.6	Using PageRank in a Search Engine . . . . .	175
5.1.7	Exercises for Section 5.1 . . . . .	175
5.2	Efficient Computation of PageRank . . . . .	177
5.2.1	Representing Transition Matrices . . . . .	178
5.2.2	PageRank Iteration Using MapReduce . . . . .	179
5.2.3	Use of Combiners to Consolidate the Result Vector . . . . .	179
5.2.4	Representing Blocks of the Transition Matrix . . . . .	180
5.2.5	Other Efficient Approaches to PageRank Iteration . . . . .	181
5.2.6	Exercises for Section 5.2 . . . . .	183
5.3	Topic-Sensitive PageRank . . . . .	183
5.3.1	Motivation for Topic-Sensitive Page Rank . . . . .	183
5.3.2	Biased Random Walks . . . . .	184
5.3.3	Using Topic-Sensitive PageRank . . . . .	185
5.3.4	Inferring Topics from Words . . . . .	186
5.3.5	Exercises for Section 5.3 . . . . .	187
5.4	Link Spam . . . . .	187
5.4.1	Architecture of a Spam Farm . . . . .	187
5.4.2	Analysis of a Spam Farm . . . . .	189
5.4.3	Combating Link Spam . . . . .	190
5.4.4	TrustRank . . . . .	190
5.4.5	Spam Mass . . . . .	191
5.4.6	Exercises for Section 5.4 . . . . .	191
5.5	Hubs and Authorities . . . . .	192
5.5.1	The Intuition Behind HITS . . . . .	192
5.5.2	Formalizing Hubbiness and Authority . . . . .	193
5.5.3	Exercises for Section 5.5 . . . . .	196

5.6	Summary of Chapter 5 . . . . .	196
5.7	References for Chapter 5 . . . . .	200
<b>6</b>	<b>Frequent Itemsets</b>	<b>201</b>
6.1	The Market-Basket Model . . . . .	202
6.1.1	Definition of Frequent Itemsets . . . . .	202
6.1.2	Applications of Frequent Itemsets . . . . .	204
6.1.3	Association Rules . . . . .	205
6.1.4	Finding Association Rules with High Confidence . . . . .	207
6.1.5	Exercises for Section 6.1 . . . . .	207
6.2	Market Baskets and the A-Priori Algorithm . . . . .	209
6.2.1	Representation of Market-Basket Data . . . . .	209
6.2.2	Use of Main Memory for Itemset Counting . . . . .	210
6.2.3	Monotonicity of Itemsets . . . . .	212
6.2.4	Tyranny of Counting Pairs . . . . .	213
6.2.5	The A-Priori Algorithm . . . . .	213
6.2.6	A-Priori for All Frequent Itemsets . . . . .	214
6.2.7	Exercises for Section 6.2 . . . . .	217
6.3	Handling Larger Datasets in Main Memory . . . . .	218
6.3.1	The Algorithm of Park, Chen, and Yu . . . . .	218
6.3.2	The Multistage Algorithm . . . . .	220
6.3.3	The Multihash Algorithm . . . . .	222
6.3.4	Exercises for Section 6.3 . . . . .	224
6.4	Limited-Pass Algorithms . . . . .	226
6.4.1	The Simple, Randomized Algorithm . . . . .	226
6.4.2	Avoiding Errors in Sampling Algorithms . . . . .	227
6.4.3	The Algorithm of Savasere, Omiecinski, and Navathe . . . . .	228
6.4.4	The SON Algorithm and MapReduce . . . . .	229
6.4.5	Toivonen's Algorithm . . . . .	230
6.4.6	Why Toivonen's Algorithm Works . . . . .	231
6.4.7	Exercises for Section 6.4 . . . . .	232
6.5	Counting Frequent Items in a Stream . . . . .	232
6.5.1	Sampling Methods for Streams . . . . .	233
6.5.2	Frequent Itemsets in Decaying Windows . . . . .	234
6.5.3	Hybrid Methods . . . . .	235
6.5.4	Exercises for Section 6.5 . . . . .	235
6.6	Summary of Chapter 6 . . . . .	236
6.7	References for Chapter 6 . . . . .	238
<b>7</b>	<b>Clustering</b>	<b>241</b>
7.1	Introduction to Clustering Techniques . . . . .	241
7.1.1	Points, Spaces, and Distances . . . . .	241
7.1.2	Clustering Strategies . . . . .	243
7.1.3	The Curse of Dimensionality . . . . .	244

7.1.4	Exercises for Section 7.1 . . . . .	245
7.2	Hierarchical Clustering . . . . .	245
7.2.1	Hierarchical Clustering in a Euclidean Space . . . . .	246
7.2.2	Efficiency of Hierarchical Clustering . . . . .	248
7.2.3	Alternative Rules for Controlling Hierarchical Clustering . . . . .	249
7.2.4	Hierarchical Clustering in Non-Euclidean Spaces . . . . .	252
7.2.5	Exercises for Section 7.2 . . . . .	253
7.3	K-means Algorithms . . . . .	254
7.3.1	K-Means Basics . . . . .	255
7.3.2	Initializing Clusters for K-Means . . . . .	255
7.3.3	Picking the Right Value of $k$ . . . . .	256
7.3.4	The Algorithm of Bradley, Fayyad, and Reina . . . . .	257
7.3.5	Processing Data in the BFR Algorithm . . . . .	259
7.3.6	Exercises for Section 7.3 . . . . .	262
7.4	The CURE Algorithm . . . . .	262
7.4.1	Initialization in CURE . . . . .	263
7.4.2	Completion of the CURE Algorithm . . . . .	264
7.4.3	Exercises for Section 7.4 . . . . .	265
7.5	Clustering in Non-Euclidean Spaces . . . . .	266
7.5.1	Representing Clusters in the GRGPF Algorithm . . . . .	266
7.5.2	Initializing the Cluster Tree . . . . .	267
7.5.3	Adding Points in the GRGPF Algorithm . . . . .	268
7.5.4	Splitting and Merging Clusters . . . . .	269
7.5.5	Exercises for Section 7.5 . . . . .	270
7.6	Clustering for Streams and Parallelism . . . . .	270
7.6.1	The Stream-Computing Model . . . . .	271
7.6.2	A Stream-Clustering Algorithm . . . . .	271
7.6.3	Initializing Buckets . . . . .	272
7.6.4	Merging Buckets . . . . .	272
7.6.5	Answering Queries . . . . .	275
7.6.6	Clustering in a Parallel Environment . . . . .	275
7.6.7	Exercises for Section 7.6 . . . . .	276
7.7	Summary of Chapter 7 . . . . .	276
7.8	References for Chapter 7 . . . . .	280
<b>8</b>	<b>Advertising on the Web</b>	<b>281</b>
8.1	Issues in On-Line Advertising . . . . .	281
8.1.1	Advertising Opportunities . . . . .	281
8.1.2	Direct Placement of Ads . . . . .	282
8.1.3	Issues for Display Ads . . . . .	283
8.2	On-Line Algorithms . . . . .	284
8.2.1	On-Line and Off-Line Algorithms . . . . .	284
8.2.2	Greedy Algorithms . . . . .	285
8.2.3	The Competitive Ratio . . . . .	286



8.2.4	Exercises for Section 8.2 . . . . .	286
8.3	The Matching Problem . . . . .	287
8.3.1	Matches and Perfect Matches . . . . .	287
8.3.2	The Greedy Algorithm for Maximal Matching . . . . .	288
8.3.3	Competitive Ratio for Greedy Matching . . . . .	289
8.3.4	Exercises for Section 8.3 . . . . .	290
8.4	The Adwords Problem . . . . .	290
8.4.1	History of Search Advertising . . . . .	291
8.4.2	Definition of the Adwords Problem . . . . .	291
8.4.3	The Greedy Approach to the Adwords Problem . . . . .	292
8.4.4	The Balance Algorithm . . . . .	293
8.4.5	A Lower Bound on Competitive Ratio for Balance . . . . .	294
8.4.6	The Balance Algorithm with Many Bidders . . . . .	296
8.4.7	The Generalized Balance Algorithm . . . . .	297
8.4.8	Final Observations About the Adwords Problem . . . . .	298
8.4.9	Exercises for Section 8.4 . . . . .	299
8.5	Adwords Implementation . . . . .	299
8.5.1	Matching Bids and Search Queries . . . . .	300
8.5.2	More Complex Matching Problems . . . . .	300
8.5.3	A Matching Algorithm for Documents and Bids . . . . .	301
8.6	Summary of Chapter 8 . . . . .	303
8.7	References for Chapter 8 . . . . .	305
<b>9</b>	<b>Recommendation Systems</b>	<b>307</b>
9.1	A Model for Recommendation Systems . . . . .	307
9.1.1	The Utility Matrix . . . . .	308
9.1.2	The Long Tail . . . . .	309
9.1.3	Applications of Recommendation Systems . . . . .	309
9.1.4	Populating the Utility Matrix . . . . .	311
9.2	Content-Based Recommendations . . . . .	312
9.2.1	Item Profiles . . . . .	312
9.2.2	Discovering Features of Documents . . . . .	313
9.2.3	Obtaining Item Features From Tags . . . . .	314
9.2.4	Representing Item Profiles . . . . .	315
9.2.5	User Profiles . . . . .	316
9.2.6	Recommending Items to Users Based on Content . . . . .	317
9.2.7	Classification Algorithms . . . . .	318
9.2.8	Exercises for Section 9.2 . . . . .	320
9.3	Collaborative Filtering . . . . .	321
9.3.1	Measuring Similarity . . . . .	322
9.3.2	The Duality of Similarity . . . . .	324
9.3.3	Clustering Users and Items . . . . .	325
9.3.4	Exercises for Section 9.3 . . . . .	327
9.4	Dimensionality Reduction . . . . .	328
9.4.1	UV-Decomposition . . . . .	328

9.4.2	Root-Mean-Square Error . . . . .	329
9.4.3	Incremental Computation of a UV-Decomposition . . . . .	330
9.4.4	Optimizing an Arbitrary Element . . . . .	332
9.4.5	Building a Complete UV-Decomposition Algorithm . . . . .	334
9.4.6	Exercises for Section 9.4 . . . . .	336
9.5	The Netflix Challenge . . . . .	337
9.6	Summary of Chapter 9 . . . . .	338
9.7	References for Chapter 9 . . . . .	340
<b>10</b>	<b>Mining Social-Network Graphs . . . . .</b>	<b>343</b>
10.1	Social Networks as Graphs . . . . .	343
10.1.1	What is a Social Network? . . . . .	344
10.1.2	Social Networks as Graphs . . . . .	344
10.1.3	Varieties of Social Networks . . . . .	346
10.1.4	Graphs With Several Node Types . . . . .	347
10.1.5	Exercises for Section 10.1 . . . . .	348
10.2	Clustering of Social-Network Graphs . . . . .	349
10.2.1	Distance Measures for Social-Network Graphs . . . . .	349
10.2.2	Applying Standard Clustering Methods . . . . .	349
10.2.3	Betweenness . . . . .	351
10.2.4	The Girvan-Newman Algorithm . . . . .	351
10.2.5	Using Betweenness to Find Communities . . . . .	354
10.2.6	Exercises for Section 10.2 . . . . .	356
10.3	Direct Discovery of Communities . . . . .	357
10.3.1	Finding Cliques . . . . .	357
10.3.2	Complete Bipartite Graphs . . . . .	357
10.3.3	Finding Complete Bipartite Subgraphs . . . . .	358
10.3.4	Why Complete Bipartite Graphs Must Exist . . . . .	359
10.3.5	Exercises for Section 10.3 . . . . .	361
10.4	Partitioning of Graphs . . . . .	361
10.4.1	What Makes a Good Partition? . . . . .	362
10.4.2	Normalized Cuts . . . . .	362
10.4.3	Some Matrices That Describe Graphs . . . . .	363
10.4.4	Eigenvalues of the Laplacian Matrix . . . . .	364
10.4.5	Alternative Partitioning Methods . . . . .	367
10.4.6	Exercises for Section 10.4 . . . . .	368
10.5	Finding Overlapping Communities . . . . .	369
10.5.1	The Nature of Communities . . . . .	369
10.5.2	Maximum-Likelihood Estimation . . . . .	369
10.5.3	The Affiliation-Graph Model . . . . .	371
10.5.4	Avoiding the Use of Discrete Membership Changes . . . . .	374
10.5.5	Exercises for Section 10.5 . . . . .	375
10.6	Simrank . . . . .	376
10.6.1	Random Walkers on a Social Graph . . . . .	376
10.6.2	Random Walks with Restart . . . . .	377

10.6.3 Exercises for Section 10.6 . . . . .	380
10.7 Counting Triangles . . . . .	380
10.7.1 Why Count Triangles? . . . . .	380
10.7.2 An Algorithm for Finding Triangles . . . . .	381
10.7.3 Optimality of the Triangle-Finding Algorithm . . . . .	382
10.7.4 Finding Triangles Using MapReduce . . . . .	383
10.7.5 Using Fewer Reduce Tasks . . . . .	384
10.7.6 Exercises for Section 10.7 . . . . .	385
10.8 Neighborhood Properties of Graphs . . . . .	386
10.8.1 Directed Graphs and Neighborhoods . . . . .	386
10.8.2 The Diameter of a Graph . . . . .	388
10.8.3 Transitive Closure and Reachability . . . . .	389
10.8.4 Transitive Closure Via MapReduce . . . . .	390
10.8.5 Smart Transitive Closure . . . . .	392
10.8.6 Transitive Closure by Graph Reduction . . . . .	393
10.8.7 Approximating the Sizes of Neighborhoods . . . . .	395
10.8.8 Exercises for Section 10.8 . . . . .	397
10.9 Summary of Chapter 10 . . . . .	398
10.10References for Chapter 10 . . . . .	402
<b>11 Dimensionality Reduction</b>	<b>405</b>
11.1 Eigenvalues and Eigenvectors of Symmetric Matrices . . . . .	406
11.1.1 Definitions . . . . .	406
11.1.2 Computing Eigenvalues and Eigenvectors . . . . .	407
11.1.3 Finding Eigenpairs by Power Iteration . . . . .	408
11.1.4 The Matrix of Eigenvectors . . . . .	411
11.1.5 Exercises for Section 11.1 . . . . .	411
11.2 Principal-Component Analysis . . . . .	412
11.2.1 An Illustrative Example . . . . .	413
11.2.2 Using Eigenvectors for Dimensionality Reduction . . . . .	416
11.2.3 The Matrix of Distances . . . . .	417
11.2.4 Exercises for Section 11.2 . . . . .	418
11.3 Singular-Value Decomposition . . . . .	418
11.3.1 Definition of SVD . . . . .	418
11.3.2 Interpretation of SVD . . . . .	420
11.3.3 Dimensionality Reduction Using SVD . . . . .	422
11.3.4 Why Zeroing Low Singular Values Works . . . . .	423
11.3.5 Querying Using Concepts . . . . .	425
11.3.6 Computing the SVD of a Matrix . . . . .	426
11.3.7 Exercises for Section 11.3 . . . . .	427
11.4 CUR Decomposition . . . . .	428
11.4.1 Definition of CUR . . . . .	429
11.4.2 Choosing Rows and Columns Properly . . . . .	430
11.4.3 Constructing the Middle Matrix . . . . .	431
11.4.4 The Complete CUR Decomposition . . . . .	432

11.4.5	Eliminating Duplicate Rows and Columns . . . . .	433
11.4.6	Exercises for Section 11.4 . . . . .	434
11.5	Summary of Chapter 11 . . . . .	434
11.6	References for Chapter 11 . . . . .	436
<b>12</b>	<b>Large-Scale Machine Learning</b>	<b>439</b>
12.1	The Machine-Learning Model . . . . .	440
12.1.1	Training Sets . . . . .	440
12.1.2	Some Illustrative Examples . . . . .	440
12.1.3	Approaches to Machine Learning . . . . .	443
12.1.4	Machine-Learning Architecture . . . . .	444
12.1.5	Exercises for Section 12.1 . . . . .	447
12.2	Perceptrons . . . . .	447
12.2.1	Training a Perceptron with Zero Threshold . . . . .	447
12.2.2	Convergence of Perceptrons . . . . .	451
12.2.3	The Winnow Algorithm . . . . .	451
12.2.4	Allowing the Threshold to Vary . . . . .	453
12.2.5	Multiclass Perceptrons . . . . .	455
12.2.6	Transforming the Training Set . . . . .	456
12.2.7	Problems With Perceptrons . . . . .	457
12.2.8	Parallel Implementation of Perceptrons . . . . .	458
12.2.9	Exercises for Section 12.2 . . . . .	459
12.3	Support-Vector Machines . . . . .	461
12.3.1	The Mechanics of an SVM . . . . .	461
12.3.2	Normalizing the Hyperplane . . . . .	462
12.3.3	Finding Optimal Approximate Separators . . . . .	464
12.3.4	SVM Solutions by Gradient Descent . . . . .	467
12.3.5	Stochastic Gradient Descent . . . . .	470
12.3.6	Parallel Implementation of SVM . . . . .	471
12.3.7	Exercises for Section 12.3 . . . . .	472
12.4	Learning from Nearest Neighbors . . . . .	472
12.4.1	The Framework for Nearest-Neighbor Calculations . . . . .	473
12.4.2	Learning with One Nearest Neighbor . . . . .	473
12.4.3	Learning One-Dimensional Functions . . . . .	474
12.4.4	Kernel Regression . . . . .	477
12.4.5	Dealing with High-Dimensional Euclidean Data . . . . .	477
12.4.6	Dealing with Non-Euclidean Distances . . . . .	479
12.4.7	Exercises for Section 12.4 . . . . .	479
12.5	Comparison of Learning Methods . . . . .	480
12.6	Summary of Chapter 12 . . . . .	481
12.7	References for Chapter 12 . . . . .	483



# Chapter 1

## Data Mining

In this introductory chapter we begin with the essence of data mining and a discussion of how data mining is treated by the various disciplines that contribute to this field. We cover “Bonferroni’s Principle,” which is really a warning about overusing the ability to mine data. This chapter is also the place where we summarize a few useful ideas that are not data mining but are useful in understanding some important data-mining concepts. These include the TF.IDF measure of word importance, behavior of hash functions and indexes, and identities involving  $e$ , the base of natural logarithms. Finally, we give an outline of the topics covered in the balance of the book.

### 1.1 What is Data Mining?

The most commonly accepted definition of “data mining” is the discovery of “models” for data. A “model,” however, can be one of several things. We mention below the most important directions in modeling.

#### 1.1.1 Statistical Modeling

Statisticians were the first to use the term “data mining.” Originally, “data mining” or “data dredging” was a derogatory term referring to attempts to extract information that was not supported by the data. Section 1.2 illustrates the sort of errors one can make by trying to extract what really isn’t in the data. Today, “data mining” has taken on a positive meaning. Now, statisticians view data mining as the construction of a *statistical model*, that is, an underlying distribution from which the visible data is drawn.

**Example 1.1:** Suppose our data is a set of numbers. This data is much simpler than data that would be data-mined, but it will serve as an example. A statistician might decide that the data comes from a Gaussian distribution and use a formula to compute the most likely parameters of this Gaussian. The mean

and standard deviation of this Gaussian distribution completely characterize the distribution and would become the model of the data.  $\square$

### 1.1.2 Machine Learning

There are some who regard data mining as synonymous with machine learning. There is no question that some data mining appropriately uses algorithms from machine learning. Machine-learning practitioners use the data as a training set, to train an algorithm of one of the many types used by machine-learning practitioners, such as Bayes nets, support-vector machines, decision trees, hidden Markov models, and many others.

There are situations where using data in this way makes sense. The typical case where machine learning is a good approach is when we have little idea of what we are looking for in the data. For example, it is rather unclear what it is about movies that makes certain movie-goers like or dislike it. Thus, in answering the “Netflix challenge” to devise an algorithm that predicts the ratings of movies by users, based on a sample of their responses, machine-learning algorithms have proved quite successful. We shall discuss a simple form of this type of algorithm in Section 9.4.

On the other hand, machine learning has not proved successful in situations where we can describe the goals of the mining more directly. An interesting case in point is the attempt by WhizBang! Labs<sup>1</sup> to use machine learning to locate people’s resumes on the Web. It was not able to do better than algorithms designed by hand to look for some of the obvious words and phrases that appear in the typical resume. Since everyone who has looked at or written a resume has a pretty good idea of what resumes contain, there was no mystery about what makes a Web page a resume. Thus, there was no advantage to machine-learning over the direct design of an algorithm to discover resumes.

### 1.1.3 Computational Approaches to Modeling

More recently, computer scientists have looked at data mining as an algorithmic problem. In this case, the model of the data is simply the answer to a complex query about it. For instance, given the set of numbers of Example 1.1, we might compute their average and standard deviation. Note that these values might not be the parameters of the Gaussian that best fits the data, although they will almost certainly be very close if the size of the data is large.

There are many different approaches to modeling data. We have already mentioned the possibility of constructing a statistical process whereby the data could have been generated. Most other approaches to modeling can be described as either

1. Summarizing the data succinctly and approximately, or

---

<sup>1</sup>This startup attempted to use machine learning to mine large-scale data, and hired many of the top machine-learning people to do so. Unfortunately, it was not able to survive.

2. Extracting the most prominent features of the data and ignoring the rest.

We shall explore these two approaches in the following sections.

#### 1.1.4 Summarization

One of the most interesting forms of summarization is the PageRank idea, which made Google successful and which we shall cover in Chapter 5. In this form of Web mining, the entire complex structure of the Web is summarized by a single number for each page. This number, the “PageRank” of the page, is (oversimplifying somewhat) the probability that a random walker on the graph would be at that page at any given time. The remarkable property this ranking has is that it reflects very well the “importance” of the page – the degree to which typical searchers would like that page returned as an answer to their search query.

Another important form of summary – clustering – will be covered in Chapter 7. Here, data is viewed as points in a multidimensional space. Points that are “close” in this space are assigned to the same cluster. The clusters themselves are summarized, perhaps by giving the centroid of the cluster and the average distance from the centroid of points in the cluster. These cluster summaries become the summary of the entire data set.

**Example 1.2:** A famous instance of clustering to solve a problem took place long ago in London, and it was done entirely without computers.<sup>2</sup> The physician John Snow, dealing with a Cholera outbreak plotted the cases on a map of the city. A small illustration suggesting the process is shown in Fig. 1.1.

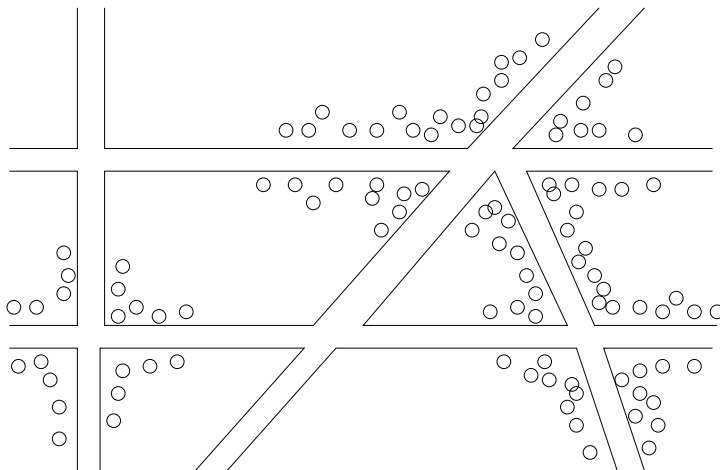


Figure 1.1: Plotting cholera cases on a map of London

<sup>2</sup>See [http://en.wikipedia.org/wiki/1854\\_Broad\\_Street\\_cholera\\_outbreak](http://en.wikipedia.org/wiki/1854_Broad_Street_cholera_outbreak).



The cases clustered around some of the intersections of roads. These intersections were the locations of wells that had become contaminated; people who lived nearest these wells got sick, while people who lived nearer to wells that had not been contaminated did not get sick. Without the ability to cluster the data, the cause of Cholera would not have been discovered.  $\square$

### 1.1.5 Feature Extraction

The typical feature-based model looks for the most extreme examples of a phenomenon and represents the data by these examples. If you are familiar with Bayes nets, a branch of machine learning and a topic we do not cover in this book, you know how a complex relationship between objects is represented by finding the strongest statistical dependencies among these objects and using only those in representing all statistical connections. Some of the important kinds of feature extraction from large-scale data that we shall study are:

1. *Frequent Itemsets.* This model makes sense for data that consists of “baskets” of small sets of items, as in the market-basket problem that we shall discuss in Chapter 6. We look for small sets of items that appear together in many baskets, and these “frequent itemsets” are the characterization of the data that we seek. The original application of this sort of mining was true market baskets: the sets of items, such as hamburger and ketchup, that people tend to buy together when checking out at the cash register of a store or super market.
2. *Similar Items.* Often, your data looks like a collection of sets, and the objective is to find pairs of sets that have a relatively large fraction of their elements in common. An example is treating customers at an on-line store like Amazon as the set of items they have bought. In order for Amazon to recommend something else they might like, Amazon can look for “similar” customers and recommend something many of these customers have bought. This process is called “collaborative filtering.” If customers were single-minded, that is, they bought only one kind of thing, then clustering customers might work. However, since customers tend to have interests in many different things, it is more useful to find, for each customer, a small number of other customers who are similar in their tastes, and represent the data by these connections. We discuss similarity in Chapter 3.

## 1.2 Statistical Limits on Data Mining

A common sort of data-mining problem involves discovering unusual events hidden within massive amounts of data. This section is a discussion of the problem, including “Bonferroni’s Principle,” a warning against overzealous use of data mining.

### 1.2.1 Total Information Awareness

Following the terrorist attack of Sept. 11, 2001, it was noticed that there were four people enrolled in different flight schools, learning how to pilot commercial aircraft, although they were not affiliated with any airline. It was conjectured that the information needed to predict and foil the attack was available in data, but that there was then no way to examine the data and detect suspicious events. The response was a program called TIA, or *Total Information Awareness*, which was intended to mine all the data it could find, including credit-card receipts, hotel records, travel data, and many other kinds of information in order to track terrorist activity. TIA naturally caused great concern among privacy advocates, and the project was eventually killed by Congress. It is not the purpose of this book to discuss the difficult issue of the privacy-security tradeoff. However, the prospect of TIA or a system like it does raise many technical questions about its feasibility.

The concern raised by many is that if you look at so much data, and you try to find within it activities that look like terrorist behavior, are you not going to find many innocent activities – or even illicit activities that are not terrorism – that will result in visits from the police and maybe worse than just a visit? The answer is that it all depends on how narrowly you define the activities that you look for. Statisticians have seen this problem in many guises and have a theory, which we introduce in the next section.

### 1.2.2 Bonferroni's Principle

Suppose you have a certain amount of data, and you look for events of a certain type within that data. You can expect events of this type to occur, even if the data is completely random, and the number of occurrences of these events will grow as the size of the data grows. These occurrences are “bogus,” in the sense that they have no cause other than that random data will always have some number of unusual features that look significant but aren't. A theorem of statistics, known as the *Bonferroni correction* gives a statistically sound way to avoid most of these bogus positive responses to a search through the data. Without going into the statistical details, we offer an informal version, *Bonferroni's principle*, that helps us avoid treating random occurrences as if they were real. Calculate the expected number of occurrences of the events you are looking for, on the assumption that data is random. If this number is significantly larger than the number of real instances you hope to find, then you must expect almost anything you find to be bogus, i.e., a statistical artifact rather than evidence of what you are looking for. This observation is the informal statement of Bonferroni's principle.

In a situation like searching for terrorists, where we expect that there are few terrorists operating at any one time, Bonferroni's principle says that we may only detect terrorists by looking for events that are so rare that they are unlikely to occur in random data. We shall give an extended example in the

next section.

### 1.2.3 An Example of Bonferroni's Principle

Suppose there are believed to be some “evil-doers” out there, and we want to detect them. Suppose further that we have reason to believe that periodically, evil-doers gather at a hotel to plot their evil. Let us make the following assumptions about the size of the problem:

1. There are one billion people who might be evil-doers.
2. Everyone goes to a hotel one day in 100.
3. A hotel holds 100 people. Hence, there are 100,000 hotels – enough to hold the 1% of a billion people who visit a hotel on any given day.
4. We shall examine hotel records for 1000 days.

To find evil-doers in this data, we shall look for people who, on two different days, were both at the same hotel. Suppose, however, that there really are no evil-doers. That is, everyone behaves at random, deciding with probability 0.01 to visit a hotel on any given day, and if so, choosing one of the  $10^5$  hotels at random. Would we find any pairs of people who appear to be evil-doers?

We can do a simple approximate calculation as follows. The probability of any two people both deciding to visit a hotel on any given day is .0001. The chance that they will visit the same hotel is this probability divided by  $10^5$ , the number of hotels. Thus, the chance that they will visit the same hotel on one given day is  $10^{-9}$ . The chance that they will visit the same hotel on two different given days is the square of this number,  $10^{-18}$ . Note that the hotels can be different on the two days.

Now, we must consider how many events will indicate evil-doing. An “event” in this sense is a pair of people and a pair of days, such that the two people were at the same hotel on each of the two days. To simplify the arithmetic, note that for large  $n$ ,  $\binom{n}{2}$  is about  $n^2/2$ . We shall use this approximation in what follows. Thus, the number of pairs of people is  $\binom{10^9}{2} = 5 \times 10^{17}$ . The number of pairs of days is  $\binom{1000}{2} = 5 \times 10^5$ . The expected number of events that look like evil-doing is the product of the number of pairs of people, the number of pairs of days, and the probability that any one pair of people and pair of days is an instance of the behavior we are looking for. That number is

$$5 \times 10^{17} \times 5 \times 10^5 \times 10^{-18} = 250,000$$

That is, there will be a quarter of a million pairs of people who look like evil-doers, even though they are not.

Now, suppose there really are 10 pairs of evil-doers out there. The police will need to investigate a quarter of a million other pairs in order to find the real evil-doers. In addition to the intrusion on the lives of half a million innocent

people, the work involved is sufficiently great that this approach to finding evil-doers is probably not feasible.

### 1.2.4 Exercises for Section 1.2

**Exercise 1.2.1:** Using the information from Section 1.2.3, what would be the number of suspected pairs if the following changes were made to the data (and all other numbers remained as they were in that section)?

- (a) The number of days of observation was raised to 2000.
- (b) The number of people observed was raised to 2 billion (and there were therefore 200,000 hotels).
- (c) We only reported a pair as suspect if they were at the same hotel at the same time on three different days.

**! Exercise 1.2.2:** Suppose we have information about the supermarket purchases of 100 million people. Each person goes to the supermarket 100 times in a year and buys 10 of the 1000 items that the supermarket sells. We believe that a pair of terrorists will buy exactly the same set of 10 items (perhaps the ingredients for a bomb?) at some time during the year. If we search for pairs of people who have bought the same set of items, would we expect that any such people found were truly terrorists?<sup>3</sup>

## 1.3 Things Useful to Know

In this section, we offer brief introductions to subjects that you may or may not have seen in your study of other courses. Each will be useful in the study of data mining. They include:

1. The TF.IDF measure of word importance.
2. Hash functions and their use.
3. Secondary storage (disk) and its effect on running time of algorithms.
4. The base  $e$  of natural logarithms and identities involving that constant.
5. Power laws.

---

<sup>3</sup>That is, assume our hypothesis that terrorists will surely buy a set of 10 items in common at some time during the year. We don't want to address the matter of whether or not terrorists would necessarily do so.