Contents

Biography xxxi
Preface xxxiii

PART 1 INTRODUCTION

CHAPTER 1 Statistical Machine Learning 3
1.1 Types of Learning 3
1.2 Examples of Machine Learning Tasks 4
1.2.1 Supervised Learning 4
1.2.2 Unsupervised Learning 5
1.2.3 Further Topics 6
1.3 Structure of This Textbook 8

PART 2 STATISTICS AND PROBABILITY

CHAPTER 2 Random Variables and Probability Distributions 11
2.1 Mathematical Preliminaries 11
2.2 Probability 13
2.3 Random Variable and Probability Distribution 14
2.4 Properties of Probability Distributions 16
2.4.1 Expectation, Median, and Mode 16
2.4.2 Variance and Standard Deviation 18
2.4.3 Skewness, Kurtosis, and Moments 19
2.5 Transformation of Random Variables 22

CHAPTER 3 Examples of Discrete Probability Distributions 25
3.1 Discrete Uniform Distribution 25
3.2 Binomial Distribution 26
3.3 Hypergeometric Distribution 27
3.4 Poisson Distribution 31
3.5 Negative Binomial Distribution 33
3.6 Geometric Distribution 35

CHAPTER 4 Examples of Continuous Probability Distributions 37
4.1 Continuous Uniform Distribution 37
4.2 Normal Distribution 37
4.3 Gamma Distribution, Exponential Distribution, and Chi-Squared Distribution 41
4.4 Beta Distribution 44
4.5 Cauchy Distribution and Laplace Distribution 47
4.6 t-Distribution and F-Distribution 49

CHAPTER 5 Multidimensional Probability Distributions 51
5.1 Joint Probability Distribution 51
5.2 Conditional Probability Distribution 52
5.3 Contingency Table 53
5.4 Bayes' Theorem 53
5.5 Covariance and Correlation 55
5.6 Independence 56

CHAPTER 6 Examples of Multidimensional Probability Distributions 61
6.1 Multinomial Distribution 61
6.2 Multivariate Normal Distribution 62
6.3 Dirichlet Distribution 63
6.4 Wishart Distribution 70

CHAPTER 7 Sum of Independent Random Variables 73
7.1 Convolution 73
7.2 Reproductive Property 74
7.3 Law of Large Numbers 74
7.4 Central Limit Theorem 77

CHAPTER 8 Probability Inequalities 81
8.1 Union Bound 81
8.2 Inequalities for Probabilities 82
8.2.1 Markov's Inequality and Chernoff's Inequality 82
8.2.2 Cantelli's Inequality and Chebyshev's Inequality 83
8.3 Inequalities for Expectation 84
8.3.1 Jensen's Inequality 84
8.3.2 Hölder's Inequality and Schwarz's Inequality 85
8.3.3 Minkowski's Inequality 86
8.3.4 Kantorovich's Inequality 87
8.4 Inequalities for the Sum of Independent Random Variables 87
8.4.1 Chebyshev's Inequality and Chernoff's Inequality 88
8.4.2 Hoeffding's Inequality and Bernstein's Inequality 88
8.4.3 Bennett's Inequality 89

CHAPTER 9 Statistical Estimation 91
9.1 Fundamentals of Statistical Estimation 91
9.2 Point Estimation 92
9.2.1 Parametric Density Estimation 92
9.2.2 Nonparametric Density Estimation 93
9.2.3 Regression and Classification 93

9.2.4 Model Selection 94
9.3 Interval Estimation 95
9.3.1 Interval Estimation for Expectation of Normal Samples 95
9.3.2 Bootstrap Confidence Interval 96
9.3.3 Bayesian Credible Interval 97

CHAPTER 10 Hypothesis Testing 99
10.1 Fundamentals of Hypothesis Testing 99
10.2 Test for Expectation of Normal Samples 100
10.3 Neyman-Pearson Lemma 101
10.4 Test for Contingency Tables 102
10.5 Test for Difference in Expectations of Normal Samples .. 104
10.5.1 Two Samples without Correspondence 104
10.5.2 Two Samples with Correspondence 105
10.6 Nonparametric Test for Ranks 107
10.6.1 Two Samples without Correspondence 107
10.6.2 Two Samples with Correspondence 108
10.7 Monte Carlo Test 108

PART 3 GENERATIVE APPROACH TO STATISTICAL PATTERN RECOGNITION

CHAPTER 11 Pattern Recognition via Generative Model Estimation 113
11.1 Formulation of Pattern Recognition 113
11.2 Statistical Pattern Recognition 115
11.3 Criteria for Classifier Training 117
11.3.1 MAP Rule 117
11.3.2 Minimum Misclassification Rate Rule 118
11.3.3 Bayes Decision Rule 119
11.3.4 Discussion 121
11.4 Generative and Discriminative Approaches 121

CHAPTER 12 Maximum Likelihood Estimation 123
12.1 Definition 123
12.2 Gaussian Model 125
12.3 Computing the Class-Posterior Probability 127
12.4 Fisher's Linear Discriminant Analysis (FDA) 130
12.5 Hand-Written Digit Recognition 133
12.5.1 Preparation 134
12.5.2 Implementing Linear Discriminant Analysis 135
12.5.3 Multiclass Classification 136

CHAPTER 13 Properties of Maximum Likelihood Estimation 139

13.1 Consistency 139
13.2 Asymptotic Unbiasedness 140
13.3 Asymptotic Efficiency 141
13.3.1 One-Dimensional Case 141
13.3.2 Multidimensional Cases 141
13.4 Asymptotic Normality 143
13.5 Summary 145

CHAPTER 14 Model Selection for Maximum Likelihood Estimation 147
14.1 Model Selection 147
14.2 KL Divergence 148
14.3 AIC 150
14.4 Cross Validation 154
14.5 Discussion 154

CHAPTER 15 Maximum Likelihood Estimation for Gaussian Mixture Model 157
15.1 Gaussian Mixture Model 157
15.2 MLE 158
15.3 Gradient Ascent Algorithm 161
15.4 EM Algorithm 162

CHAPTER 16 Nonparametric Estimation 169
16.1 Histogram Method 169
16.2 Problem Formulation 170
16.3 KDE 174
16.3.1 Parzen Window Method 174
16.3.2 Smoothing with Kernels 175
16.3.3 Bandwidth Selection 176
16.4 NNDE 178
16.4.1 Nearest Neighbor Distance 178
16.4.2 Nearest Neighbor Classifier 179

CHAPTER 17 Bayesian Inference 185
17.1 Bayesian Predictive Distribution 185
17.1.1 Definition 185
17.1.2 Comparison with MLE 186
17.1.3 Computational Issues 188
17.2 Conjugate Prior 188
17.3 MAP Estimation 189
17.4 Bayesian Model Selection 193

CHAPTER 18 Analytic Approximation of Marginal Likelihood 197
18.1 Laplace Approximation 197
18.1.1 Approximation with Gaussian Density 197

18.1.2 Illustration 199
18.1.3 Application to Marginal Likelihood Approximation 200
18.1.4 Bayesian Information Criterion (BIC) 200
18.2 Variational Approximation 202
18.2.1 Variational Bayesian EM (VBEM) Algorithm 202
18.2.2 Relation to Ordinary EM Algorithm 203

CHAPTER 19 Numerical Approximation of Predictive Distribution 205
19.1 Monte Carlo Integration 205
19.2 Importance Sampling 207
19.3 Sampling Algorithms 208
19.3.1 Inverse Transform Sampling 208
19.3.2 Rejection Sampling 212
19.3.3 Markov Chain Monte Carlo (MCMC) Method 214

CHAPTER 20 Bayesian Mixture Models 221
20.1 Gaussian Mixture Models 221
20.1.1 Bayesian Formulation 221
20.1.2 Variational Inference 223
20.1.3 Gibbs Sampling 228
20.2 Latent Dirichlet Allocation (LDA) 229
20.2.1 Topic Models 230
20.2.2 Bayesian Formulation 231
20.2.3 Gibbs Sampling 232

PART 4 DISCRIMINATIVE APPROACH TO STATISTICAL MACHINE LEARNING

CHAPTER 21 Learning Models 237
21.1 Linear-in-Parameter Model 237
21.2 Kernel Model 239
21.3 Hierarchical Model 242

CHAPTER 22 Least Squares Regression 245
22.1 Method of LS 245
22.2 Solution for Linear-in-Parameter Model 246
22.3 Properties of LS Solution 250
22.4 Learning Algorithm for Large-Scale Data 251
22.5 Learning Algorithm for Hierarchical Model 252

CHAPTER 23 Constrained LS Regression 257
23.1 Subspace-Constrained LS 257
23.2 ℓ_2-Constrained LS 259

23.3 Model Selection 262

CHAPTER 24 Sparse Regression **267**

24.1 ℓ_1-Constrained LS 267

24.2 Solving ℓ_1-Constrained LS 268

24.3 Feature Selection by Sparse Learning 272

24.4 Various Extensions 272

24.4.1 Generalized ℓ_1-Constrained LS 273

24.4.2 ℓ_p-Constrained LS 273

24.4.3 $\ell_1 + \ell_2$-Constrained LS 274

24.4.4 $\ell_{1,2}$-Constrained LS 276

24.4.5 Trace Norm Constrained LS 278

CHAPTER 25 Robust Regression **279**

25.1 Nonrobustness of ℓ_2-Loss Minimization 279

25.2 ℓ_1-Loss Minimization 280

25.3 Huber Loss Minimization 282

25.3.1 Definition 282

25.3.2 Stochastic Gradient Algorithm 283

25.3.3 Iteratively Reweighted LS 283

25.3.4 ℓ_1-Constrained Huber Loss Minimization 286

25.4 Tukey Loss Minimization 290

CHAPTER 26 Least Squares Classification **295**

26.1 Classification by LS Regression 295

26.2 $0/1$-Loss and Margin 297

26.3 Multiclass Classification 300

CHAPTER 27 Support Vector Classification **303**

27.1 Maximum Margin Classification 303

27.1.1 Hard Margin Support Vector Classification 303

27.1.2 Soft Margin Support Vector Classification 305

27.2 Dual Optimization of Support Vector Classification 306

27.3 Sparseness of Dual Solution 308

27.4 Nonlinearization by Kernel Trick 311

27.5 Multiclass Extension 312

27.6 Loss Minimization View 314

27.6.1 Hinge Loss Minimization 315

27.6.2 Squared Hinge Loss Minimization 316

27.6.3 Ramp Loss Minimization 318

CHAPTER 28 Probabilistic Classification **321**

28.1 Logistic Regression 321

28.1.1 Logistic Model and MLE 321

28.1.2 Loss Minimization View 324

28.2 LS Probabilistic Classification 325

CHAPTER 29 Structured Classification 329
29.1 Sequence Classification 329
29.2 Probabilistic Classification for Sequences 330
29.2.1 Conditional Random Field 330
29.2.2 MLE 333
29.2.3 Recursive Computation 333
29.2.4 Prediction for New Sample 336
29.3 Deterministic Classification for Sequences 337

PART 5 FURTHER TOPICS

CHAPTER 30 Ensemble Learning 343
30.1 Decision Stump Classifier 343
30.2 Bagging 344
30.3 Boosting 346
30.3.1 Adaboost 348
30.3.2 Loss Minimization View 348
30.4 General Ensemble Learning 354

CHAPTER 31 Online Learning 355
31.1 Stochastic Gradient Descent 355
31.2 Passive-Aggressive Learning 356
31.2.1 Classification 357
31.2.2 Regression 358
31.3 Adaptive Regularization of Weight Vectors (AROW) 360
31.3.1 Uncertainty of Parameters 360
31.3.2 Classification 361
31.3.3 Regression 362

CHAPTER 32 Confidence of Prediction 365
32.1 Predictive Variance for ℓ_2-Regularized LS 365
32.2 Bootstrap Confidence Estimation 367
32.3 Applications 368
32.3.1 Time-series Prediction 368
32.3.2 Tuning Parameter Optimization 369

CHAPTER 33 Semisupervised Learning 375
33.1 Manifold Regularization 375
33.1.1 Manifold Structure Brought by Input Samples 375
33.1.2 Computing the Solution 377
33.2 Covariate Shift Adaptation 378
33.2.1 Importance Weighted Learning 378

33.2.2 Relative Importance Weighted Learning 382
33.2.3 Importance Weighted Cross Validation 382
33.2.4 Importance Estimation 383
33.3 Class-balance Change Adaptation 385
33.3.1 Class-balance Weighted Learning 385
33.3.2 Class-balance Estimation 386

CHAPTER 34 Multitask Learning 391
34.1 Task Similarity Regularization 391
34.1.1 Formulation 391
34.1.2 Analytic Solution 392
34.1.3 Efficient Computation for Many Tasks 393
34.2 Multidimensional Function Learning 394
34.2.1 Formulation 394
34.2.2 Efficient Analytic Solution 397
34.3 Matrix Regularization 397
34.3.1 Parameter Matrix Regularization 397
34.3.2 Proximal Gradient for Trace Norm Regularization 400

CHAPTER 35 Linear Dimensionality Reduction 405
35.1 Curse of Dimensionality 405
35.2 Unsupervised Dimensionality Reduction 407
35.2.1 PCA 407
35.2.2 Locality Preserving Projection 410
35.3 Linear Discriminant Analyses for Classification............ 412
35.3.1 Fisher Discriminant Analysis.......................... 413
35.3.2 Local Fisher Discriminant Analysis................... 414
35.3.3 Semisupervised Local Fisher Discriminant Analysis 417
35.4 Sufficient Dimensionality Reduction for Regression....... 419
35.4.1 Information Theoretic Formulation 419
35.4.2 Direct Derivative Estimation 422
35.5 Matrix Imputation 425

CHAPTER 36 Nonlinear Dimensionality Reduction 429
36.1 Dimensionality Reduction with Kernel Trick................. 429
36.1.1 Kernel PCA 429
36.1.2 Laplacian Eigenmap 433
36.2 Supervised Dimensionality Reduction with Neural Networks 435
36.3 Unsupervised Dimensionality Reduction with Autoencoder 436
36.3.1 Autoencoder... 436

36.3.2 Training by Gradient Descent 437
36.3.3 Sparse Autoencoder 439
36.4 Unsupervised Dimensionality Reduction with Restricted Boltzmann Machine 440
36.4.1 Model 441
36.4.2 Training by Gradient Ascent 442
36.5 Deep Learning 446

CHAPTER 37 Clustering **447**
37.1 k-Means Clustering 447
37.2 Kernel k-Means Clustering 448
37.3 Spectral Clustering 449
37.4 Tuning Parameter Selection 452

CHAPTER 38 Outlier Detection **457**
38.1 Density Estimation and Local Outlier Factor 457
38.2 Support Vector Data Description 458
38.3 Inlier-Based Outlier Detection 464

CHAPTER 39 Change Detection **469**
39.1 Distributional Change Detection 469
39.1.1 KL Divergence 470
39.1.2 Pearson Divergence 470
39.1.3 L_2-Distance 471
39.1.4 L_1-Distance 474
39.1.5 Maximum Mean Discrepancy (MMD) 476
39.1.6 Energy Distance 477
39.1.7 Application to Change Detection in Time Series . 477
39.2 Structural Change Detection 478
39.2.1 Sparse MLE 478
39.2.2 Sparse Density Ratio Estimation 482
References 485
Index 491

List of Figures

Fig. 1.1 Regression. 5
Fig. 1.2 Classification. 5
Fig. 1.3 Clustering. 6
Fig. 1.4 Outlier detection. 6
Fig. 1.5 Dimensionality reduction. 7

Fig. 2.1 Combination of events. 12
Fig. 2.2 Examples of probability mass function. Outcome of throwing a fair six-sided dice (discrete uniform distribution $U\{1, 2, \ldots, 6\}$). 14
Fig. 2.3 Example of probability density function and its cumulative distribution function. 15
Fig. 2.4 Expectation is the average of x weighted according to $f(x)$, and median is the 50% point both from the left-hand and right-hand sides. α-quantile for $0 \leq \alpha \leq 1$ is a generalization of the median that gives the $100\alpha\%$ point from the left-hand side. Mode is the maximizer of $f(x)$. 16
Fig. 2.5 Income distribution. The expectation is 62.1 thousand dollars, while the median is 31.3 thousand dollars. 17
Fig. 2.6 Skewness. 20
Fig. 2.7 Kurtosis. 20
Fig. 2.8 Taylor series expansion at the origin. 21
Fig. 2.9 One-dimensional change of variables in integration. For multidimensional cases, see Fig. 4.2. 23

Fig. 3.1 Probability mass functions of binomial distribution $\mathrm{Bi}(n, p)$. 26
Fig. 3.2 Sampling from a bag. The bag contains N balls which consist of $M < N$ balls labeled as “A” and $N - M$ balls labeled as “B.” n balls are sampled from the bag, which consists of x balls labeled as “A” and $n - x$ balls labeled as “B.” 27
Fig. 3.3 Sampling with and without replacement. The sampled ball is returned to the bag before the next ball is sampled in sampling with replacement, while the next ball is sampled without returning the previously sampled ball in sampling without replacement. 28
Fig. 3.4 Probability mass functions of hypergeometric distribution $\mathrm{HG}(N, M, n)$. 29
Fig. 3.5 Probability mass functions of $\mathrm{Bi}(n, M/N)$ and $\mathrm{HG}(N, M, n)$ for $N = 100$, $M = 90$, and $n = 90$. 29
Fig. 3.6 Probability mass functions of Poisson distribution $\mathrm{Po}(\lambda)$. 34
Fig. 3.7 Probability mass functions of negative binomial distribution $\mathrm{NB}(k, p)$. 34
Fig. 3.8 Probability mass functions of geometric distribution $\mathrm{Ge}(p)$. 35

Fig. 4.1 Gaussian integral. 39
Fig. 4.2 Two-dimensional change of variables in integration. 40
Fig. 4.3 Probability density functions of normal density $N(\mu, \sigma^2)$. 40
Fig. 4.4 Standard normal distribution $N(0, 1)$. A random variable following $N(0, 1)$ is included in $[-1, 1]$ with probability 68.27%, in $[-2, 2]$ with probability 95.45%, and in $[-3, 3]$ with probability 99.73%. 41

Fig. 4.5	Gamma function. $\Gamma(\alpha + 1) = \alpha!$ holds for non-negative integer α, and the gamma function smoothly interpolates the factorials.	42
Fig. 4.6	Probability density functions of gamma distribution $\mathrm{Ga}(\alpha, \lambda)$.	43
Fig. 4.7	Probability density functions of beta distribution $\mathrm{Be}(\alpha, \beta)$.	46
Fig. 4.8	Probability density functions of Cauchy distribution $\mathrm{Ca}(a, b)$, Laplace distribution $\mathrm{La}(a, b)$, and normal distribution $N(a, b^2)$.	48
Fig. 4.9	Probability density functions of t-distribution $t(d)$, Cauchy distribution $\mathrm{Ca}(0, 1)$, and normal distribution $N(0, 1)$.	49
Fig. 4.10	Probability density functions of F-distribution $F(d, d')$.	50
Fig. 5.1	Correlation coefficient $\rho_{x,y}$. Linear relation between x and y can be captured.	57
Fig. 5.2	Correlation coefficient for nonlinear relations. Even when there is a nonlinear relation between x and y, the correlation coefficient can be close to zero if the probability distribution is symmetric.	58
Fig. 5.3	Example of x and y which are uncorrelated but dependent.	59
Fig. 6.1	Probability density functions of two-dimensional normal distribution $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with $\boldsymbol{\mu} = (0, 0)^\top$.	64
Fig. 6.2	Eigenvalue decomposition.	65
Fig. 6.3	Contour lines of the normal density. The principal axes of the ellipse are parallel to the eigenvectors of variance-covariance matrix $\boldsymbol{\Sigma}$, and their length is proportional to the square root of the eigenvalues.	66
Fig. 6.4	Probability density functions of Dirichlet distribution $\mathrm{Dir}(\boldsymbol{\alpha})$. The center of gravity of the triangle corresponds to $x^{(1)} = x^{(2)} = x^{(3)} = 1/3$, and each vertex represents the point that the corresponding variable takes one and the others take zeros.	69
Fig. 6.5	Vectorization operator and Kronecker product.	71
Fig. 7.1	Arithmetic mean, geometric mean, and harmonic mean.	76
Fig. 7.2	Law of large numbers.	77
Fig. 7.3	Central limit theorem. The solid lines denote the normal densities.	78
Fig. 8.1	Markov's inequality.	83
Fig. 8.2	Chebyshev's inequality.	84
Fig. 8.3	Convex function and tangent line.	85
Fig. 8.4	$h(u) = (1 + u)\log(1 + u) - u$ and $g(u) = \frac{u^2}{2+2u/3}$.	90
Fig. 9.1	Confidence interval for normal samples.	96
Fig. 9.2	Bootstrap resampling by sampling with replacement.	97
Fig. 10.1	Critical region and critical value.	101
Fig. 11.1	Hand-written digit image and its vectorization.	114
Fig. 11.2	Constructing a classifier is equivalent to determine a discrimination function, decision regions, and decision boundaries.	115

Fig. 11.3 Dimensionality reduction onto a two-dimensional subspace by principal component analysis (see Section 35.2.1). 116
Fig. 11.4 Illustration of hand-written digit samples in the pattern space. 117
Fig. 11.5 MAP rule. 118
Fig. 11.6 Minimum misclassification rate rule. 119

Fig. 12.1 Likelihood equation, setting the derivative of the likelihood to zero, is a necessary condition for the maximum likelihood solution but is not a sufficient condition in general. 124
Fig. 12.2 Log function is monotone increasing. 125
Fig. 12.3 Formulas for vector and matrix derivatives [80]. 126
Fig. 12.4 MATLAB code for MLE with one-dimensional Gaussian model. 128
Fig. 12.5 Example of MLE with one-dimensional Gaussian model. 128
Fig. 12.6 Orthogonal projection. 130
Fig. 12.7 Mahalanobis distance having hyperellipsoidal contours. 130
Fig. 12.8 Linear discriminant analysis. 132
Fig. 12.9 When the classwise sample ratio n_1/n_2 is changed. 132
Fig. 12.10 When the classwise sample distributions are rotated. 133
Fig. 12.11 Matrix and third tensor. 134
Fig. 12.12 Misclassified test patterns. 136
Fig. 12.13 MATLAB code for multiclass classification by FDA. 137
Fig. 12.14 Confusion matrix for 10-class classification by FDA. The correct classification rate is $1798/2000 = 89.9\%$. 138

Fig. 13.1 Bias-variance decomposition of expected squared error. 140
Fig. 13.2 MATLAB code for illustrating asymptotic normality of MLE. 145
Fig. 13.3 Example of asymptotic normality of MLE. 145

Fig. 14.1 Model selection. Too simple model may not be expressive enough to represent the true probability distribution, while too complex model may cause unreliable parameter estimation. 148
Fig. 14.2 For nested models, log-likelihood is monotone nondecreasing as the model complexity increases. 150
Fig. 14.3 AIC is the sum of the negative log-likelihood and the number of parameters. 151
Fig. 14.4 Big-o and small-o notations. 152
Fig. 14.5 Cross validation. 154
Fig. 14.6 Algorithm of likelihood cross validation. 155

Fig. 15.1 MLE for Gaussian model. 158
Fig. 15.2 Example of Gaussian mixture model: $q(x) = 0.4N(x;-2,1.5^2) + 0.2N(x;2,2^2) + 0.4N(x;3,1^2)$. 159
Fig. 15.3 Schematic of gradient ascent. 161
Fig. 15.4 Algorithm of gradient ascent. 161
Fig. 15.5 Step size ε in gradient ascent. The gradient flow can overshoot the peak if ε is large, while gradient ascent is slow if ε is too small. 162
Fig. 15.6 EM algorithm. 163

Fig. 15.7 Maximizing the lower bound $b(\boldsymbol{\theta})$ of the log-likelihood $\log L(\boldsymbol{\theta})$. 164
Fig. 15.8 Jensen's inequality for $m = 2$. log is a concave function. 164
Fig. 15.9 MATLAB code of EM algorithm for Gaussian mixture model. 166
Fig. 15.10 Example of EM algorithm for Gaussian mixture model. The size of ellipses is proportional to the mixing weights $\{w_\ell\}_{\ell=1}^m$. 167

Fig. 16.1 Examples of Gaussian MLE. 170
Fig. 16.2 Example of histogram method. 170
Fig. 16.3 MATLAB code for inverse transform sampling (see Section 19.3.1) for probability density function shown in Fig. 16.1(b). The bottom function should be saved as "`myrand.m`." 171
Fig. 16.4 Choice of bin width in histogram method. 171
Fig. 16.5 Notation of nonparametric methods. 172
Fig. 16.6 Probability P approximated by the size of rectangle. 172
Fig. 16.7 Normalized variance of binomial distribution. 173
Fig. 16.8 Parzen window method. 174
Fig. 16.9 Example of Parzen window method. 175
Fig. 16.10 Example of Gaussian KDE. Training samples are the same as those in Fig. 16.9. 176
Fig. 16.11 Choice of kernel bandwidth h in KDE. 177
Fig. 16.12 MATLAB code for Gaussian KDE with bandwidth selected by likelihood cross validation. A random number generator "`myrand.m`" shown in Fig. 16.3 is used. 177
Fig. 16.13 Example of Gaussian KDE with bandwidth selected by likelihood cross validation. 178
Fig. 16.14 MATLAB code for NNDE with the number of nearest neighbors selected by likelihood cross validation. A random number generator "`myrand.m`" shown in Fig. 16.3 is used. 179
Fig. 16.15 Example of NNDE with the number of nearest neighbors selected by likelihood cross validation. 180
Fig. 16.16 Example of nearest neighbor classifier. 181
Fig. 16.17 Algorithm of cross validation for misclassification rate. 182
Fig. 16.18 MATLAB code for k-nearest neighbor classifier with k chosen by cross validation. The bottom function should be saved as "`knn.m`." 183
Fig. 16.19 Confusion matrix for 10-class classification by k-nearest neighbor classifier. $k = 1$ was chosen by cross validation for misclassification rate. The correct classification rate is $1932/2000 = 96.6\%$. 183

Fig. 17.1 Bayes vs. MLE. The maximum likelihood solution $\widehat{p}_{\mathrm{ML}}$ is always confined in the parametric model $q(\boldsymbol{x};\boldsymbol{\theta})$, while the Bayesian predictive distribution $\widehat{p}_{\mathrm{Bayes}}(\boldsymbol{x})$ generally pops out from the model. 187
Fig. 17.2 MAP estimation. 190
Fig. 17.3 Example of MLE for Gaussian model. When the number of training samples, n, is small, MLE tends to overfit the samples. 190
Fig. 17.4 MATLAB code for penalized MLE with one-dimensional Gaussian model. 192
Fig. 17.5 Example of MAP estimation with one-dimensional Gaussian model. 192

Fig. 17.6 MATLAB code for empirical Bayes. 195
Fig. 17.7 Example of empirical Bayes. 195

Fig. 18.1 Laplace approximation. 199

Fig. 19.1 Numerical computation of π by Monte Carlo integration. 206
Fig. 19.2 MATLAB code for numerically computing π by Monte Carlo integration. 207
Fig. 19.3 MATLAB code for importance sampling. 208
Fig. 19.4 Examples of probability density function $p(\theta)$ and its cumulative distribution function $P(\theta)$. Cumulative distribution function is monotone nondecreasing and satisfies $\lim_{\theta\to-\infty} P(\theta) = 0$ and $\lim_{\theta\to\infty} P(\theta) = 1$. 209
Fig. 19.5 Inverse transform sampling. 209
Fig. 19.6 $\theta \leq \theta'$ implies $P(\theta) \leq P(\theta')$. 210
Fig. 19.7 Laplace distribution. 211
Fig. 19.8 MATLAB code for inverse transform sampling. 211
Fig. 19.9 Example of inverse transform sampling for Laplace distribution. 212
Fig. 19.10 Algorithm of rejection sampling. 212
Fig. 19.11 Illustration of rejection sampling when the proposal distribution is uniform. 213
Fig. 19.12 MATLAB code for rejection sampling. 214
Fig. 19.13 Example of rejection sampling. 214
Fig. 19.14 Computational efficiency of rejection sampling. (a) When the upper bound of the probability density, κ, is small, proposal points are almost always accepted and thus rejection sampling is computationally efficient. (b) When κ is large, most of the proposal points will be rejected and thus rejection sampling is computationally expensive. 215
Fig. 19.15 Random walk. 216
Fig. 19.16 MATLAB code for Metropolis-Hastings sampling. The bottom function should be saved as "`pdf.m`." 217
Fig. 19.17 Example of Metropolis-Hastings sampling. 218
Fig. 19.18 MATLAB code for Gibbs sampling. 219
Fig. 19.19 Example of Gibbs sampling. 220

Fig. 20.1 Variational Bayesian formulation of Gaussian mixture model. 223
Fig. 20.2 VBEM algorithm for Gaussian mixture model. $(\alpha_0, \beta_0, \boldsymbol{W}_0, \nu_0)$ are hyperparameters. 225
Fig. 20.3 MATLAB code of VBEM algorithm for Gaussian mixture model. 226
Fig. 20.4 Example of VBEM algorithm for Gaussian mixture model. The size of ellipses is proportional to the mixing weights $\{w_\ell\}_{\ell=1}^m$. A mixture model of five Gaussian components is used here, but three components have mixing coefficient close to zero and thus they are almost eliminated. 227
Fig. 20.5 MATLAB code of collapsed Gibbs sampling for Gaussian mixture model. 229
Fig. 20.6 Example of collapsed Gibbs sampling for Gaussian mixture model. A mixture model of five Gaussian components is used here, but only two components remain and no samples belong to the remaining three components. 230

Fig. 21.1 Linear-in-input model cannot approximate nonlinear functions. 238

Fig. 21.2 Multidimensional basis functions. The multiplicative model is expressive, but the number of parameters grows exponentially in input dimensionality. On the other hand, in the additive model, the number of parameters grows only linearly in input dimensionality, but its expression power is limited. 239
Fig. 21.3 Gaussian kernel with bandwidth h and center c. 240
Fig. 21.4 One-dimensional Gaussian kernel model. Gaussian functions are located at training input samples $\{\boldsymbol{x}_i\}_{i=1}^n$ and their height $\{\theta_i\}_{i=1}^n$ is learned. 241
Fig. 21.5 Two-dimensional Gaussian kernel model. The curse of dimensionality is mitigated by only approximating the learning target function in the vicinity of training input samples. 241
Fig. 21.6 Sigmoidal function. 242
Fig. 21.7 Hierarchical model as a three-layered network. 243

Fig. 22.1 Generalized inverse. 247
Fig. 22.2 Singular value decomposition. 248
Fig. 22.3 MATLAB code for LS regression. 249
Fig. 22.4 Example of LS regression with sinusoidal basis functions $\boldsymbol{\phi}(x) = (1, \sin\frac{x}{2}, \cos\frac{x}{2}, \sin\frac{2x}{2}, \cos\frac{2x}{2}, \dots, \sin\frac{15x}{2}, \cos\frac{15x}{2})^\top$. 249
Fig. 22.5 Geometric interpretation of LS method for linear-in-parameter model. Training output vector $\boldsymbol{y}$ is projected onto the range of $\boldsymbol{\Phi}$, denoted by $\mathcal{R}(\boldsymbol{\Phi})$, for denoising purposes. 251
Fig. 22.6 Algorithm of stochastic gradient descent for LS regression with a linear-in-parameter model. 252
Fig. 22.7 MATLAB code of stochastic gradient descent for LS regression with the Gaussian kernel model. 253
Fig. 22.8 Example of stochastic gradient descent for LS regression with the Gaussian kernel model. For $n = 50$ training samples, the Gaussian bandwidth is set at $h = 0.3$. 254
Fig. 22.9 Gradient descent for nonlinear models. The training squared error J_{LS} is nonconvex and there exist multiple local optimal solutions in general. 254
Fig. 22.10 MATLAB code for error back-propagation algorithm. 255
Fig. 22.11 Example of regression by error back-propagation algorithm. 255

Fig. 23.1 Examples of LS regression for linear-in-parameter model when the noise level in training output is high. Sinusoidal basis functions $\{1, \sin\frac{x}{2}, \cos\frac{x}{2}, \sin\frac{2x}{2}, \cos\frac{2x}{2}, \dots, \sin\frac{15x}{2}, \cos\frac{15x}{2}\}$ are used in ordinary LS, while its subset $\{1, \sin\frac{x}{2}, \cos\frac{x}{2}, \sin\frac{2x}{2}, \cos\frac{2x}{2}, \dots, \sin\frac{5x}{2}, \cos\frac{5x}{2}\}$ is used in the subspace-constrained LS method. 258
Fig. 23.2 Constraint in parameter space. 258
Fig. 23.3 MATLAB code for subspace-constrained LS regression. 259
Fig. 23.4 Parameter space in ℓ_2-constrained LS. 259
Fig. 23.5 Lagrange dual problem. 260
Fig. 23.6 MATLAB code of ℓ_2-constrained LS regression for Gaussian kernel model. 262
Fig. 23.7 Example of ℓ_2-constrained LS regression for Gaussian kernel model. The Gaussian bandwidth is set at $h = 0.3$, and the regularization parameter is set at $\lambda = 0.1$. 262

Fig. 23.8 Parameter space in generalized ℓ_2-constrained LS. 263
Fig. 23.9 Examples of ℓ_2-constrained LS with the Gaussian kernel model for different Gaussian bandwidth h and different regularization parameter λ. 263
Fig. 23.10 MATLAB code of cross validation for ℓ_2-constrained LS regression. 264
Fig. 23.11 Example of cross validation for ℓ_2-constrained LS regression. The cross validation error for all Gaussian bandwidth h and regularization parameter λ is plotted, which is minimized at $(h, \lambda) = (0.3, 0.1)$. See Fig. 23.9 for learned functions. 265
Fig. 23.12 Matrix inversion lemma. 265

Fig. 24.1 Parameter space in ℓ_1-constrained LS. 268
Fig. 24.2 The solution of ℓ_1-constrained LS tends to be on one of the coordinate axes, which is a sparse solution. 269
Fig. 24.3 Alternating direction method of multipliers. 270
Fig. 24.4 MATLAB code of ℓ_1-constrained LS by ADMM for Gaussian kernel model. 271
Fig. 24.5 Example of ℓ_1-constrained LS for Gaussian kernel model. 38 out of 50 parameters are zero. 271
Fig. 24.6 Unit ℓ_p-balls. 274
Fig. 24.7 Properties of ℓ_p-constraint. 275
Fig. 24.8 Unit $(\ell_1 + \ell_2)$-norm ball for balance parameter $\tau = 1/2$, which is similar to the unit $\ell_{1.4}$-ball. However, while the $\ell_{1.4}$-ball has no corner, the $(\ell_1 + \ell_2)$-ball has corners. 276
Fig. 24.9 Constraints in three-dimensional parameter space. 277
Fig. 24.10 Trace norm of a matrix. 278

Fig. 25.1 LS solution for straight-line model $f_{\boldsymbol{\theta}}(x) = \theta_1 + \theta_2 x$, which is strongly affected by an outlier. 280
Fig. 25.2 ℓ_2-loss and ℓ_1-loss. The ℓ_2-loss magnifies large residuals. 281
Fig. 25.3 Solution of least absolute deviations for straight-line model $f_{\boldsymbol{\theta}}(x) = \theta_1 + \theta_2 x$ for the same training samples as Fig. 25.1. Least absolute deviations give a much more robust solution than LS. 281
Fig. 25.4 Huber loss, with threshold $\eta = 1$. 282
Fig. 25.5 Quadratic upper bound $\frac{\eta r^2}{2c} + \frac{\eta c}{2} - \frac{\eta^2}{2}$ of Huber loss $\rho_{\text{Huber}}(r)$ for $c > 0$, which touches each other at $r = \pm c$. 284
Fig. 25.6 Weight functions for Huber loss minimization and Tukey loss minimization. 285
Fig. 25.7 Updated solution $\widehat{\boldsymbol{\theta}}$ is no worse than current solution $\widetilde{\boldsymbol{\theta}}$. 285
Fig. 25.8 Iteratively reweighted LS for Huber loss minimization. 286
Fig. 25.9 MATLAB code of iteratively reweighted LS for Huber loss minimization. Straight-line model $f_{\boldsymbol{\theta}}(x) = \theta_1 + \theta_2 x$ is used, with threshold $\eta = 1$. 287
Fig. 25.10 Examples of iteratively reweighted LS for Huber loss minimization. Straight-line model $f_{\boldsymbol{\theta}}(x) = \theta_1 + \theta_2 x$ is used, with threshold $\eta = 1$. 288
Fig. 25.11 Quadratic upper bound $\frac{\theta^2}{2c} + \frac{c}{2}$ of absolute value $|\theta|$ for $c > 0$, which touches each other at $\theta = \pm c$. 289
Fig. 25.12 Iteratively reweighted LS for ℓ_1-regularized Huber loss minimization. 289
Fig. 25.13 MATLAB code of iteratively reweighted LS for ℓ_1-regularized Huber loss minimization with Gaussian kernel model. 290

Fig. 25.14 Example of ℓ_1-regularized Huber loss minimization with Gaussian kernel model. 291
Fig. 25.15 Tukey loss, with threshold $\eta = 3$. 291
Fig. 25.16 Example of Tukey loss minimization. Tukey loss minimization gives more robust solutions than Huber loss minimization, but only a local optimal solution can be obtained. 292

Fig. 26.1 Binary classification as function approximation. 296
Fig. 26.2 MATLAB code of classification by ℓ_2-regularized LS for Gaussian kernel model. 296
Fig. 26.3 Example of classification by ℓ_2-regularized LS for Gaussian kernel model. 297
Fig. 26.4 0/1-loss and ℓ_2-loss as functions of margin $m = f_{\boldsymbol{\theta}}(\boldsymbol{x})y$. 298
Fig. 26.5 Example of ℓ_2-loss minimization for linear-in-input model. Since the ℓ_2-loss has a positive slope when $m > 1$, the obtained solution contains some classification error even though all samples can be correctly classified in principle. 299
Fig. 26.6 Popular surrogate loss functions. 300
Fig. 26.7 One-versus-rest reduction of multiclass classification problem. 301
Fig. 26.8 One-versus-one reduction of multiclass classification problem. 301

Fig. 27.1 Linear-in-input binary classifier $f_{\boldsymbol{w},\gamma}(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + \gamma$. $\boldsymbol{w}$ and γ are the normal vector and the intercept of the decision boundary, respectively. 304
Fig. 27.2 Decision boundaries that separate all training samples correctly. 304
Fig. 27.3 Decision boundary of hard margin support vector machine. It goes through the center of positive and negative training samples, $\boldsymbol{w}^\top \boldsymbol{x}_+ + \gamma = +1$ for some positive sample $\boldsymbol{x}_+$ and $\boldsymbol{w}^\top \boldsymbol{x}_- + \gamma = -1$ for some negative sample $\boldsymbol{x}_-$. 305
Fig. 27.4 Soft margin support vector machine allows small margin errors. 306
Fig. 27.5 Quadratic programming. 307
Fig. 27.6 Example of linear support vector classification. Among 200 dual parameters $\{\alpha_i\}_{i=1}^n$, 197 parameters take zero and only 3 parameters specified by the square in the plot take nonzero values. 309
Fig. 27.7 KKT optimality conditions. 310
Fig. 27.8 When $\alpha_i = 0$, $\boldsymbol{x}_i$ is inside the margin and correctly classified. When $0 < \alpha_i < C$, $\boldsymbol{x}_i$ is on the margin border (the dotted lines) and correctly classified. When $\alpha_i = C$, $\boldsymbol{x}_i$ is outside the margin, and if $\xi_i > 1$, $m_i < 0$ and thus $\boldsymbol{x}_i$ is misclassified. 310
Fig. 27.9 Nonlinearization of support vector machine by kernel trick. 311
Fig. 27.10 MATLAB code of support vector classification for Gaussian kernel. `quadprog.m` included in Optimization Toolbox is required. Free alternatives to `quadprog.m` are available, e.g. from `http://www.mathworks.com/matlabcentral/fileexchange/`. 313
Fig. 27.11 Example of support vector classification for Gaussian kernel. 314
Fig. 27.12 Hinge loss and squared hinge loss. 315
Fig. 27.13 Hinge loss as maximizer of $1 - m$ and 0. 316
Fig. 27.14 Iterative retargeted LS for ℓ_2-regularized squared hinge loss minimization. 317

Fig. 27.15 MATLAB code of iterative retargeted LS for ℓ_2-regularized squared hinge loss minimization. 318
Fig. 27.16 Example of ℓ_2-regularized squared hinge loss minimization. 319
Fig. 27.17 Examples of support vector classification with outliers. 319
Fig. 27.18 Ramp loss and squared ramp loss. 320

Fig. 28.1 Stochastic gradient algorithm for logistic regression. 322
Fig. 28.2 MATLAB code of stochastic gradient ascent for logistic regression. 323
Fig. 28.3 Example of stochastic gradient ascent for logistic regression. 324
Fig. 28.4 Logistic loss. 325
Fig. 28.5 MATLAB code for LS probabilistic classification. 327
Fig. 28.6 Example of LS probabilistic classification for the same data set as Fig. 28.3. 328

Fig. 29.1 Classification of sequence of hand-written digits. 330
Fig. 29.2 Sequence classification. 331
Fig. 29.3 Stochastic gradient algorithm for conditional random field. 333
Fig. 29.4 Dynamic programming, which solves a complex optimization problem by breaking it down into simpler subproblems recursively. When the number of steps to the goal is counted, dynamic programming trace back the steps from the goal. In this case, many subproblems of counting the number of steps from other positions are actually shared and thus dynamic programming can efficiently reuse the solutions to reduce the computation costs. 334

Fig. 30.1 Ensemble learning. Bagging trains weak learners in parallel, while boosting sequentially trains weak learners. 344
Fig. 30.2 Decision stump and decision tree classifiers. A decision stump is a depth-one version of a decision tree. 344
Fig. 30.3 MATLAB code for decision stump classification. 345
Fig. 30.4 Example of decision stump classification. 345
Fig. 30.5 Algorithm of bagging. 346
Fig. 30.6 MATLAB code of bagging for decision stumps. 346
Fig. 30.7 Example of bagging for decision stumps. 347
Fig. 30.8 Algorithm of adaboost. 349
Fig. 30.9 Confidence of classifier in adaboost. The confidence of classifier φ, denoted by θ, is determined based on the weighted misclassification rate R. 350
Fig. 30.10 MATLAB code of adaboost for decision stumps. 351
Fig. 30.11 Example of adaboost for decision stumps. 352
Fig. 30.12 Exponential loss. 353
Fig. 30.13 Loss functions for boosting. 353

Fig. 31.1 Choice of step size. Too large step size overshoots the optimal solution, while too small step size yields slow convergence. 356
Fig. 31.2 Algorithm of passive-aggressive classification. 358
Fig. 31.3 MATLAB code for passive-aggressive classification. 359
Fig. 31.4 Example of passive-aggressive classification. 359
Fig. 31.5 MATLAB code for passive-aggressive regression with the ℓ_2-loss. 360

Fig. 31.6 Algorithm of AROW classification. 362
Fig. 31.7 MATLAB code for AROW classification. 363
Fig. 31.8 Examples of passive-aggressive and AROW classifications. 363
Fig. 31.9 MATLAB code for AROW regression. 364

Fig. 32.1 MATLAB code for analytic computation of predictive variance. 367
Fig. 32.2 Examples of analytic computation of predictive variance. The shaded area indicates the confidence interval. 368
Fig. 32.3 MATLAB code for bootstrap-based confidence estimation. 369
Fig. 32.4 Examples of bootstrap-based confidence estimation. The shaded area indicates the confidence interval. 370
Fig. 32.5 Problem of time-series prediction. 370
Fig. 32.6 Time-series prediction from previous samples. 370
Fig. 32.7 MATLAB code for time-series prediction by ℓ_2-regularized LS. 371
Fig. 32.8 Examples of time-series prediction by ℓ_2-regularized LS. The shaded areas indicate the confidence intervals. 372
Fig. 32.9 Bayesian optimization. The shaded areas indicate the confidence intervals. 373

Fig. 33.1 Semisupervised classification. Samples in the same cluster are assumed to belong to the same class. 376
Fig. 33.2 MATLAB code for Laplacian-regularized LS. 379
Fig. 33.3 Examples of Laplacian-regularized LS compared with ordinary LS. Dots denote unlabeled training samples. 380
Fig. 33.4 Covariate shift in regression. Input distributions change, but the input-output relation is unchanged. 380
Fig. 33.5 MATLAB code for importance weighted LS. 381
Fig. 33.6 Example of LS learning under covariate shift. The dashed lines denote learned functions. 381
Fig. 33.7 Relative importance when $p'(x)$ is the Gaussian density with expectation 0 and variance 1 and $p(x)$ is the Gaussian density with expectation 0.5 and variance 1. 383
Fig. 33.8 Algorithm of importance weighted cross validation. 384
Fig. 33.9 MATLAB code for LS relative density ratio estimation for Gaussian kernel model. 386
Fig. 33.10 Example of LS relative density ratio estimation. $\times$'s in the right plot show estimated relative importance values at $\{\boldsymbol{x}_i\}_{i=1}^n$. 387
Fig. 33.11 Class-balance change, which affects the decision boundary. 387
Fig. 33.12 Class-prior estimation by distribution matching. 388
Fig. 33.13 MATLAB code for class-balance weighted LS. 389
Fig. 33.14 Example of class-balance weighted LS. The test class priors are estimated as $\widehat{p}'(y = 1) = 0.18$ and $\widehat{p}'(y = 2) = 0.82$, which are used as weights in class-balance weighted LS. 390

Fig. 34.1 MATLAB code for multitask LS. 394
Fig. 34.2 Examples of multitask LS. The dashed lines denote true decision boundaries and the contour lines denote learned results. 395

Fig. 34.3 Alternate learning of task similarity $\gamma_{t,t'}$ and solution $\boldsymbol{\theta}$. 396
Fig. 34.4 Multidimensional function learning. 396
Fig. 34.5 Continuous Sylvester equation. 398
Fig. 34.6 MATLAB code for multidimensional regression. 399
Fig. 34.7 Examples of multidimensional regression. 399
Fig. 34.8 Proximal gradient method. 401
Fig. 34.9 MATLAB code for multitask learning with trace norm regularization. 402
Fig. 34.10 Examples of multitask LS with trace norm regularization. The data set is the same as Fig. 34.2. The dashed lines denote true decision boundaries and the contour lines denote learned results. 403

Fig. 35.1 Curse of dimensionality. 406
Fig. 35.2 Linear dimensionality reduction. Transformation by a fat matrix $\boldsymbol{T}$ corresponds to projection onto a subspace. 407
Fig. 35.3 Data centering. 407
Fig. 35.4 PCA, which tries to keep the position of original samples when the dimensionality is reduced. 408
Fig. 35.5 MATLAB code for PCA. 409
Fig. 35.6 Example of PCA. The solid line denotes the one-dimensional embedding subspace found by PCA. 409
Fig. 35.7 Locality preserving projection, which tries to keep the cluster structure of original samples when the dimensionality is reduced. 410
Fig. 35.8 Popular choices of similarity measure. 411
Fig. 35.9 MATLAB code for locality preserving projection. 412
Fig. 35.10 Example of locality preserving projection. The solid line denotes the one-dimensional embedding subspace found by locality preserving projection. 413
Fig. 35.11 MATLAB code for Fisher discriminant analysis. 414
Fig. 35.12 Examples of Fisher discriminant analysis. The solid lines denote the found subspaces to which training samples are projected. 415
Fig. 35.13 MATLAB code for local Fisher discriminant analysis. 417
Fig. 35.14 Examples of local Fisher discriminant analysis for the same data sets as Fig. 35.12. The solid lines denote the found subspaces to which training samples are projected. 418
Fig. 35.15 MATLAB code for semisupervised local Fisher discriminant analysis. 420
Fig. 35.16 Examples of semisupervised local Fisher discriminant analysis. Lines denote the found subspaces to which training samples are projected. “LFDA” stands for local Fisher discriminant analysis, “SELF” stands for semisupervised LFDA, and “PCA” stands for principal component analysis. 421
Fig. 35.17 MATLAB code for supervised dimensionality reduction based on QMI. 423
Fig. 35.18 Example of supervised dimensionality reduction based on QMI. The solid line denotes the found subspace to which training samples are projected. 424
Fig. 35.19 MATLAB code for unsupervised dimensionality reduction based on QMI. 425
Fig. 35.20 Example of unsupervised dimensionality reduction based on QMI. The solid line denotes the found subspace to which training samples are projected. 426
Fig. 35.21 MATLAB code for unsupervised matrix imputation. 426

Fig. 35.22 Example of unsupervised matrix imputation. The gray level indicates the value of each entry in $[-5, 5]$. 427

Fig. 36.1 Nonlinear PCA in a feature space. "$\times$" denotes a sample, the solid line denotes the one-dimensional embedding subspace found by PCA, and "$\circ$" denotes a projected sample. 430
Fig. 36.2 Eigenvalue problems for PCA. Appropriately choosing the expression of eigenvalue problem depending on whether matrix $\boldsymbol{\Psi}$ is fat or skinny allows us to reduce the computational costs. 431
Fig. 36.3 MATLAB code for kernel PCA with Gaussian kernels. 433
Fig. 36.4 Examples of kernel PCA with Gaussian kernels. Original two-dimensional samples are transformed to infinite-dimensional feature space by Gaussian kernels with width h, and then PCA is applied to reduce the dimensionality to two. 434
Fig. 36.5 MATLAB code of Laplacian eigenmap for 10-nearest neighbor similarity. 435
Fig. 36.6 Example of Laplacian eigenmap for 10-nearest neighbor similarity. 435
Fig. 36.7 Dimensionality reduction by neural network. The number of hidden nodes is smaller than the number of input (and output) nodes. 436
Fig. 36.8 Autoencoder. Input and output are the same and the number of hidden nodes is smaller than the number of input nodes. 437
Fig. 36.9 Chain rule for autoencoder. 438
Fig. 36.10 MATLAB code for denoising autoencoder. See Section 12.5 for details of hand-written digit data set "`digit.mat`." 439
Fig. 36.11 Example of denoising autoencoder. 440
Fig. 36.12 Restricted Boltzmann machine. 441
Fig. 36.13 Contrastive divergence algorithm for restricted Boltzmann machine. Note that $q(\boldsymbol{z}|\boldsymbol{x} = \widehat{\boldsymbol{x}}_i)$ and $q(\boldsymbol{x}|\boldsymbol{z} = \widehat{\boldsymbol{z}}_i)$ can be factorized as Eq. (36.6), which allows efficient computation. 443
Fig. 36.14 MATLAB code for denoising restricted Boltzmann machine. See Section 12.5 for details of hand-written digit data set "`digit.mat`." 443
Fig. 36.15 Example of denoising restricted Boltzmann machine. 444
Fig. 36.16 Construction of deep neural network by stacking. 445

Fig. 37.1 k-means clustering algorithm. 448
Fig. 37.2 MATLAB code for k-means clustering. 449
Fig. 37.3 Example of k-means clustering. A filled square denotes a cluster center. 450
Fig. 37.4 Algorithm of spectral clustering. 451
Fig. 37.5 MATLAB code for spectral clustering. 451
Fig. 37.6 Example of spectral clustering. 452
Fig. 37.7 Clustering can be regarded as compressing d-dimensional vector $\boldsymbol{x}$ into c-valued scalar y. 453
Fig. 37.8 MATLAB code for LS QMI estimation. 455
Fig. 37.9 Example of LS QMI estimation. 456

Fig. 38.1 MATLAB code for local outlier factor. 459

Fig. 38.2 Example of outlier detection by local outlier factor. The diameter of circles around samples is proportional to the value of local outlier factor. 460
Fig. 38.3 Support vector data description. A hypersphere that contains *most* of the training samples is found. Samples outside the hypersphere are regarded as outliers. 460
Fig. 38.4 MATLAB code of support vector data description for Gaussian kernel. `quadprog.m` included in Optimization Toolbox is required. Free alternatives to `quadprog.m` are available, e.g. from `http://www.mathworks.com/matlabcentral/fileexchange/`. 463
Fig. 38.5 Examples of support vector data description for Gaussian kernel. Circled samples are regarded as outliers. 464
Fig. 38.6 Inlier-based outlier detection by density ratio estimation. For inlier density $p'(\boldsymbol{x})$ and test sample density $p(\boldsymbol{x})$, the density ratio $w(\boldsymbol{x}) = p'(\boldsymbol{x})/p(\boldsymbol{x})$ is close to one when $\boldsymbol{x}$ is an inlier and it is close to zero when $\boldsymbol{x}$ is an outlier. 465
Fig. 38.7 MATLAB code of KL density ratio estimation for Gaussian kernel model with Gaussian bandwidth chosen by cross validation. The bottom function should be saved as "`KLIEP.m`." 466
Fig. 38.8 Example of KL density ratio estimation for Gaussian kernel model. 467

Fig. 39.1 MATLAB code for LS density difference estimation. 473
Fig. 39.2 Example of LS density difference estimation. $\times$'s in the right plot show estimated density difference values at $\{\boldsymbol{x}_i\}_{i=1}^{n}$ and $\{\boldsymbol{x}'_{i'}\}_{i'=1}^{n'}$. 474
Fig. 39.3 Lower bound of sign (t) by $-2\max(0, 1-t)+1$. 475
Fig. 39.4 Change detection in time series. 478
Fig. 39.5 MATLAB code for change detection in time series based on the energy distance. 479
Fig. 39.6 Examples of change detection in time series based on the energy distance. 480
Fig. 39.7 Structural change in Gaussian Markov networks. 480
Fig. 39.8 MATLAB code of a gradient-projection algorithm of ℓ_1-constraint MLE for Gaussian Markov networks. The bottom function should be saved as "`L1BallProjection.m`." 481
Fig. 39.9 MATLAB code of a gradient-projection algorithm of ℓ_1-constraint KL density ratio estimation for Gaussian Markov networks. "`L1BallProjection.m`" is given in Fig. 39.8. 484

List of Tables

Table 5.1 Example of Contingency Table 53

Table 7.1 Convolution 75

Table 10.1 Type-I Error α (False Positive) and Type-II Error β (False Negative) 102

Table 10.2 Contingency Table for $x \in \{1, \ldots, \ell\}$ and $y \in \{1, \ldots, m\}$. $c_{x,y}$ Denotes the Frequency of (x, y), $d_x = \sum_{y=1}^{m} c_{x,y}$, $e_y = \sum_{x=1}^{\ell} c_{x,y}$, and $n = \sum_{x=1}^{\ell} \sum_{y=1}^{m} c_{x,y}$ 103

Table 10.3 Wilcoxon Rank-Sum Test. In this Example, $r_1 = 3$, $r_2 = 5.5$, $r_3 = 1$, and the Rank-Sum is $r = 9.5$ 107

Table 11.1 Example Of Asymmetric Loss 120