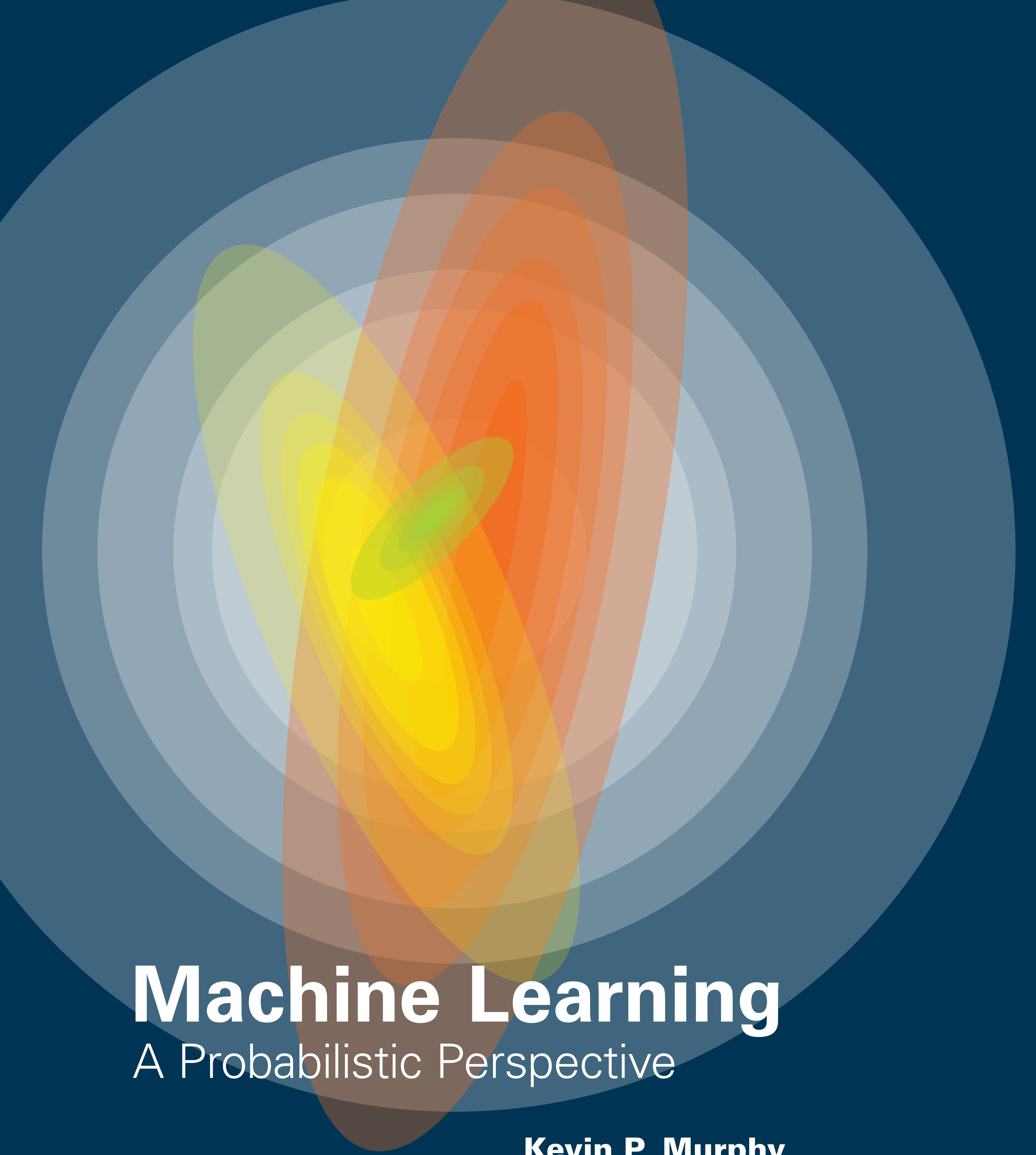


Machine Learning

A Probabilistic Perspective

Kevin P. Murphy

Machine Learning: A Probabilistic Perspective

Machine Learning
A Probabilistic Perspective

Kevin P. Murphy

The MIT Press
Cambridge, Massachusetts
London, England

© 2012 Massachusetts Institute of Technology

All rights reserved. No part of this book may be reproduced in any form by any electronic or mechanical means (including photocopying, recording, or information storage and retrieval) without permission in writing from the publisher.

For information about special quantity discounts, please email special_sales@mitpress.mit.edu

This book was set in the LaTeX programming language by the author. Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Information

Murphy, Kevin P.
Machine learning : a probabilistic perspective / Kevin P. Murphy.
p. cm. — (Adaptive computation and machine learning series)
Includes bibliographical references and index.
ISBN 978-0-262-01802-9 (hardcover : alk. paper)
1. Machine learning. 2. Probabilities. I. Title.
Q325.5.M87 2012
006.3'1—dc23
2012004558

10 9 8 7 6 5 4 3 2 1

This book is dedicated to Alessandro, Michael and Stefano,
and to the memory of Gerard Joseph Murphy.

Contents

Preface **xxvii**

1 Introduction **1**

1.1 Machine learning: what and why? 1
- 1.1.1 Types of machine learning 2

1.2 Supervised learning 3
- 1.2.1 Classification 3
- 1.2.2 Regression 8

1.3 Unsupervised learning 9
- 1.3.1 Discovering clusters 10
- 1.3.2 Discovering latent factors 11
- 1.3.3 Discovering graph structure 13
- 1.3.4 Matrix completion 14

1.4 Some basic concepts in machine learning 16
- 1.4.1 Parametric vs non-parametric models 16
- 1.4.2 A simple non-parametric classifier: K-nearest neighbors 16
- 1.4.3 The curse of dimensionality 18
- 1.4.4 Parametric models for classification and regression 19
- 1.4.5 Linear regression 19
- 1.4.6 Logistic regression 21
- 1.4.7 Overfitting 22
- 1.4.8 Model selection 22
- 1.4.9 No free lunch theorem 24

2 Probability **27**

2.1 Introduction 27

2.2 A brief review of probability theory 28
- 2.2.1 Discrete random variables 28
- 2.2.2 Fundamental rules 28
- 2.2.3 Bayes rule 29
- 2.2.4 Independence and conditional independence 30
- 2.2.5 Continuous random variables 32

2.2.6 Quantiles 33
2.2.7 Mean and variance 33
2.3 Some common discrete distributions 34
2.3.1 The binomial and Bernoulli distributions 34
2.3.2 The multinomial and multinoulli distributions 35
2.3.3 The Poisson distribution 37
2.3.4 The empirical distribution 37
2.4 Some common continuous distributions 38
2.4.1 Gaussian (normal) distribution 38
2.4.2 Degenerate pdf 39
2.4.3 The Laplace distribution 41
2.4.4 The gamma distribution 41
2.4.5 The beta distribution 42
2.4.6 Pareto distribution 43
2.5 Joint probability distributions 44
2.5.1 Covariance and correlation 44
2.5.2 The multivariate Gaussian 46
2.5.3 Multivariate Student t distribution 46
2.5.4 Dirichlet distribution 47
2.6 Transformations of random variables 49
2.6.1 Linear transformations 49
2.6.2 General transformations 50
2.6.3 Central limit theorem 51
2.7 Monte Carlo approximation 52
2.7.1 Example: change of variables, the MC way 53
2.7.2 Example: estimating π by Monte Carlo integration 54
2.7.3 Accuracy of Monte Carlo approximation 54
2.8 Information theory 56
2.8.1 Entropy 56
2.8.2 KL divergence 57
2.8.3 Mutual information 59

3 Generative models for discrete data **65**

3.1 Introduction 65
3.2 Bayesian concept learning 65
3.2.1 Likelihood 67
3.2.2 Prior 67
3.2.3 Posterior 68
3.2.4 Posterior predictive distribution 71
3.2.5 A more complex prior 72
3.3 The beta-binomial model 72
3.3.1 Likelihood 73
3.3.2 Prior 74
3.3.3 Posterior 75
3.3.4 Posterior predictive distribution 77

3.4 The Dirichlet-multinomial model 78
3.4.1 Likelihood 79
3.4.2 Prior 79
3.4.3 Posterior 79
3.4.4 Posterior predictive 81
3.5 Naive Bayes classifiers 82
3.5.1 Model fitting 83
3.5.2 Using the model for prediction 85
3.5.3 The log-sum-exp trick 86
3.5.4 Feature selection using mutual information 86
3.5.5 Classifying documents using bag of words 87

***4 Gaussian models* 97**
4.1 Introduction 97
4.1.1 Notation 97
4.1.2 Basics 97
4.1.3 MLE for an MVN 99
4.1.4 Maximum entropy derivation of the Gaussian * 101
4.2 Gaussian discriminant analysis 101
4.2.1 Quadratic discriminant analysis (QDA) 102
4.2.2 Linear discriminant analysis (LDA) 103
4.2.3 Two-class LDA 104
4.2.4 MLE for discriminant analysis 106
4.2.5 Strategies for preventing overfitting 106
4.2.6 Regularized LDA * 107
4.2.7 Diagonal LDA 108
4.2.8 Nearest shrunken centroids classifier * 109
4.3 Inference in jointly Gaussian distributions 110
4.3.1 Statement of the result 111
4.3.2 Examples 111
4.3.3 Information form 115
4.3.4 Proof of the result * 116
4.4 Linear Gaussian systems 119
4.4.1 Statement of the result 119
4.4.2 Examples 120
4.4.3 Proof of the result * 124
4.5 Digression: The Wishart distribution * 125
4.5.1 Inverse Wishart distribution 126
4.5.2 Visualizing the Wishart distribution * 127
4.6 Inferring the parameters of an MVN 127
4.6.1 Posterior distribution of $\boldsymbol{\mu}$ 128
4.6.2 Posterior distribution of $\boldsymbol{\Sigma}$ * 128
4.6.3 Posterior distribution of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ * 132
4.6.4 Sensor fusion with unknown precisions * 138

5 *Bayesian statistics* 149
5.1 Introduction 149
5.2 Summarizing posterior distributions 149
5.2.1 MAP estimation 149
5.2.2 Credible intervals 152
5.2.3 Inference for a difference in proportions 154
5.3 Bayesian model selection 155
5.3.1 Bayesian Occam's razor 156
5.3.2 Computing the marginal likelihood (evidence) 158
5.3.3 Bayes factors 163
5.3.4 Jeffreys-Lindley paradox * 164
5.4 Priors 165
5.4.1 Uninformative priors 165
5.4.2 Jeffreys priors * 166
5.4.3 Robust priors 168
5.4.4 Mixtures of conjugate priors 168
5.5 Hierarchical Bayes 171
5.5.1 Example: modeling related cancer rates 171
5.6 Empirical Bayes 172
5.6.1 Example: beta-binomial model 173
5.6.2 Example: Gaussian-Gaussian model 173
5.7 Bayesian decision theory 176
5.7.1 Bayes estimators for common loss functions 177
5.7.2 The false positive vs false negative tradeoff 180
5.7.3 Other topics * 184

6 *Frequentist statistics* 191
6.1 Introduction 191
6.2 Sampling distribution of an estimator 191
6.2.1 Bootstrap 192
6.2.2 Large sample theory for the MLE * 193
6.3 Frequentist decision theory 194
6.3.1 Bayes risk 195
6.3.2 Minimax risk 196
6.3.3 Admissible estimators 197
6.4 Desirable properties of estimators 200
6.4.1 Consistent estimators 200
6.4.2 Unbiased estimators 200
6.4.3 Minimum variance estimators 201
6.4.4 The bias-variance tradeoff 202
6.5 Empirical risk minimization 204
6.5.1 Regularized risk minimization 205
6.5.2 Structural risk minimization 206
6.5.3 Estimating the risk using cross validation 206
6.5.4 Upper bounding the risk using statistical learning theory * 209

6.5.5 Surrogate loss functions 210
6.6 Pathologies of frequentist statistics * 211
6.6.1 Counter-intuitive behavior of confidence intervals 212
6.6.2 p-values considered harmful 213
6.6.3 The likelihood principle 214
6.6.4 Why isn't everyone a Bayesian? 215

7 Linear regression 217

7.1 Introduction 217
7.2 Model specification 217
7.3 Maximum likelihood estimation (least squares) 217
7.3.1 Derivation of the MLE 219
7.3.2 Geometric interpretation 220
7.3.3 Convexity 221
7.4 Robust linear regression * 223
7.5 Ridge regression 225
7.5.1 Basic idea 225
7.5.2 Numerically stable computation * 227
7.5.3 Connection with PCA * 228
7.5.4 Regularization effects of big data 230
7.6 Bayesian linear regression 231
7.6.1 Computing the posterior 232
7.6.2 Computing the posterior predictive 233
7.6.3 Bayesian inference when σ^2 is unknown * 234
7.6.4 EB for linear regression (evidence procedure) 238

8 Logistic regression 245

8.1 Introduction 245
8.2 Model specification 245
8.3 Model fitting 245
8.3.1 MLE 246
8.3.2 Steepest descent 247
8.3.3 Newton's method 249
8.3.4 Iteratively reweighted least squares (IRLS) 250
8.3.5 Quasi-Newton (variable metric) methods 251
8.3.6 ℓ_2 regularization 252
8.3.7 Multi-class logistic regression 252
8.4 Bayesian logistic regression 254
8.4.1 Laplace approximation 255
8.4.2 Derivation of the BIC 255
8.4.3 Gaussian approximation for logistic regression 256
8.4.4 Approximating the posterior predictive 256
8.4.5 Residual analysis (outlier detection) * 260
8.5 Online learning and stochastic optimization 261
8.5.1 Online learning and regret minimization 262

8.5.2 Stochastic optimization and risk minimization 262
8.5.3 The LMS algorithm 264
8.5.4 The perceptron algorithm 265
8.5.5 A Bayesian view 266
8.6 Generative vs discriminative classifiers 267
8.6.1 Pros and cons of each approach 268
8.6.2 Dealing with missing data 269
8.6.3 Fisher's linear discriminant analysis (FLDA) * 271

***9 Generalized linear models and the exponential family* 281**

9.1 Introduction 281
9.2 The exponential family 281
9.2.1 Definition 282
9.2.2 Examples 282
9.2.3 Log partition function 284
9.2.4 MLE for the exponential family 286
9.2.5 Bayes for the exponential family * 287
9.2.6 Maximum entropy derivation of the exponential family * 289
9.3 Generalized linear models (GLMs) 290
9.3.1 Basics 290
9.3.2 ML and MAP estimation 292
9.3.3 Bayesian inference 293
9.4 Probit regression 293
9.4.1 ML/MAP estimation using gradient-based optimization 294
9.4.2 Latent variable interpretation 294
9.4.3 Ordinal probit regression * 295
9.4.4 Multinomial probit models * 295
9.5 Multi-task learning 296
9.5.1 Hierarchical Bayes for multi-task learning 296
9.5.2 Application to personalized email spam filtering 296
9.5.3 Application to domain adaptation 297
9.5.4 Other kinds of prior 297
9.6 Generalized linear mixed models * 298
9.6.1 Example: semi-parametric GLMMs for medical data 298
9.6.2 Computational issues 300
9.7 Learning to rank * 300
9.7.1 The pointwise approach 301
9.7.2 The pairwise approach 301
9.7.3 The listwise approach 302
9.7.4 Loss functions for ranking 303

***10 Directed graphical models (Bayes nets)* 307**

10.1 Introduction 307
10.1.1 Chain rule 307
10.1.2 Conditional independence 308

10.1.3 Graphical models 308
10.1.4 Graph terminology 309
10.1.5 Directed graphical models 310
10.2 Examples 311
10.2.1 Naive Bayes classifiers 311
10.2.2 Markov and hidden Markov models 312
10.2.3 Medical diagnosis 313
10.2.4 Genetic linkage analysis * 315
10.2.5 Directed Gaussian graphical models * 318
10.3 Inference 319
10.4 Learning 320
10.4.1 Plate notation 320
10.4.2 Learning from complete data 322
10.4.3 Learning with missing and/or latent variables 323
10.5 Conditional independence properties of DGMs 324
10.5.1 d-separation and the Bayes Ball algorithm (global Markov properties) 324
10.5.2 Other Markov properties of DGMs 327
10.5.3 Markov blanket and full conditionals 327
10.6 Influence (decision) diagrams * 328

***11 Mixture models and the EM algorithm* 337**

11.1 Latent variable models 337
11.2 Mixture models 337
11.2.1 Mixtures of Gaussians 339
11.2.2 Mixture of multinoullis 340
11.2.3 Using mixture models for clustering 340
11.2.4 Mixtures of experts 342
11.3 Parameter estimation for mixture models 345
11.3.1 Unidentifiability 346
11.3.2 Computing a MAP estimate is non-convex 347
11.4 The EM algorithm 348
11.4.1 Basic idea 349
11.4.2 EM for GMMs 350
11.4.3 EM for mixture of experts 357
11.4.4 EM for DGMs with hidden variables 358
11.4.5 EM for the Student distribution * 359
11.4.6 EM for probit regression * 362
11.4.7 Theoretical basis for EM * 363
11.4.8 Online EM 365
11.4.9 Other EM variants * 367
11.5 Model selection for latent variable models 370
11.5.1 Model selection for probabilistic models 370
11.5.2 Model selection for non-probabilistic methods 370
11.6 Fitting models with missing data 372

11.6.1 EM for the MLE of an MVN with missing data 373

12 Latent linear models **381**

12.1 Factor analysis 381
12.1.1 FA is a low rank parameterization of an MVN 381
12.1.2 Inference of the latent factors 382
12.1.3 Unidentifiability 383
12.1.4 Mixtures of factor analysers 385
12.1.5 EM for factor analysis models 386
12.1.6 Fitting FA models with missing data 387
12.2 Principal components analysis (PCA) 387
12.2.1 Classical PCA: statement of the theorem 387
12.2.2 Proof * 389
12.2.3 Singular value decomposition (SVD) 392
12.2.4 Probabilistic PCA 395
12.2.5 EM algorithm for PCA 396
12.3 Choosing the number of latent dimensions 398
12.3.1 Model selection for FA/PPCA 398
12.3.2 Model selection for PCA 399
12.4 PCA for categorical data 402
12.5 PCA for paired and multi-view data 404
12.5.1 Supervised PCA (latent factor regression) 405
12.5.2 Partial least squares 406
12.5.3 Canonical correlation analysis 407
12.6 Independent Component Analysis (ICA) 407
12.6.1 Maximum likelihood estimation 410
12.6.2 The FastICA algorithm 411
12.6.3 Using EM 414
12.6.4 Other estimation principles * 415

13 Sparse linear models **421**

13.1 Introduction 421
13.2 Bayesian variable selection 422
13.2.1 The spike and slab model 424
13.2.2 From the Bernoulli-Gaussian model to ℓ_0 regularization 425
13.2.3 Algorithms 426
13.3 ℓ_1 regularization: basics 429
13.3.1 Why does ℓ_1 regularization yield sparse solutions? 430
13.3.2 Optimality conditions for lasso 431
13.3.3 Comparison of least squares, lasso, ridge and subset selection 435
13.3.4 Regularization path 436
13.3.5 Model selection 439
13.3.6 Bayesian inference for linear models with Laplace priors 440
13.4 ℓ_1 regularization: algorithms 441
13.4.1 Coordinate descent 441

13.4.2 LARS and other homotopy methods 441
13.4.3 Proximal and gradient projection methods 442
13.4.4 EM for lasso 447
13.5 ℓ_1 regularization: extensions 449
13.5.1 Group Lasso 449
13.5.2 Fused lasso 454
13.5.3 Elastic net (ridge and lasso combined) 455
13.6 Non-convex regularizers 457
13.6.1 Bridge regression 458
13.6.2 Hierarchical adaptive lasso 458
13.6.3 Other hierarchical priors 462
13.7 Automatic relevance determination (ARD)/sparse Bayesian learning (SBL) 463
13.7.1 ARD for linear regression 463
13.7.2 Whence sparsity? 465
13.7.3 Connection to MAP estimation 465
13.7.4 Algorithms for ARD * 466
13.7.5 ARD for logistic regression 468
13.8 Sparse coding * 468
13.8.1 Learning a sparse coding dictionary 469
13.8.2 Results of dictionary learning from image patches 470
13.8.3 Compressed sensing 472
13.8.4 Image inpainting and denoising 472

***14 Kernels* 479**

14.1 Introduction 479
14.2 Kernel functions 479
14.2.1 RBF kernels 480
14.2.2 Kernels for comparing documents 480
14.2.3 Mercer (positive definite) kernels 481
14.2.4 Linear kernels 482
14.2.5 Matern kernels 482
14.2.6 String kernels 483
14.2.7 Pyramid match kernels 484
14.2.8 Kernels derived from probabilistic generative models 485
14.3 Using kernels inside GLMs 486
14.3.1 Kernel machines 486
14.3.2 L1VMs, RVMs, and other sparse vector machines 487
14.4 The kernel trick 488
14.4.1 Kernelized nearest neighbor classification 489
14.4.2 Kernelized K-medoids clustering 489
14.4.3 Kernelized ridge regression 492
14.4.4 Kernel PCA 493
14.5 Support vector machines (SVMs) 496
14.5.1 SVMs for regression 497
14.5.2 SVMs for classification 498

14.5.3 Choosing C 504
14.5.4 Summary of key points 504
14.5.5 A probabilistic interpretation of SVMs 505
14.6 Comparison of discriminative kernel methods 505
14.7 Kernels for building generative models 507
14.7.1 Smoothing kernels 507
14.7.2 Kernel density estimation (KDE) 508
14.7.3 From KDE to KNN 509
14.7.4 Kernel regression 510
14.7.5 Locally weighted regression 512

***15 Gaussian processes* 515**

15.1 Introduction 515
15.2 GPs for regression 516
15.2.1 Predictions using noise-free observations 517
15.2.2 Predictions using noisy observations 518
15.2.3 Effect of the kernel parameters 519
15.2.4 Estimating the kernel parameters 521
15.2.5 Computational and numerical issues * 524
15.2.6 Semi-parametric GPs * 524
15.3 GPs meet GLMs 525
15.3.1 Binary classification 525
15.3.2 Multi-class classification 528
15.3.3 GPs for Poisson regression 531
15.4 Connection with other methods 532
15.4.1 Linear models compared to GPs 532
15.4.2 Linear smoothers compared to GPs 533
15.4.3 SVMs compared to GPs 534
15.4.4 L1VM and RVMs compared to GPs 534
15.4.5 Neural networks compared to GPs 535
15.4.6 Smoothing splines compared to GPs * 536
15.4.7 RKHS methods compared to GPs * 538
15.5 GP latent variable model 540
15.6 Approximation methods for large datasets 542

***16 Adaptive basis function models* 543**

16.1 Introduction 543
16.2 Classification and regression trees (CART) 544
16.2.1 Basics 544
16.2.2 Growing a tree 545
16.2.3 Pruning a tree 549
16.2.4 Pros and cons of trees 550
16.2.5 Random forests 550
16.2.6 CART compared to hierarchical mixture of experts * 551
16.3 Generalized additive models 552

16.3.1 Backfitting 552
16.3.2 Computational efficiency 553
16.3.3 Multivariate adaptive regression splines (MARS) 553
16.4 Boosting 554
16.4.1 Forward stagewise additive modeling 555
16.4.2 L2boosting 557
16.4.3 AdaBoost 558
16.4.4 LogitBoost 559
16.4.5 Boosting as functional gradient descent 560
16.4.6 Sparse boosting 561
16.4.7 Multivariate adaptive regression trees (MART) 562
16.4.8 Why does boosting work so well? 562
16.4.9 A Bayesian view 563
16.5 Feedforward neural networks (multilayer perceptrons) 563
16.5.1 Convolutional neural networks 564
16.5.2 Other kinds of neural networks 568
16.5.3 A brief history of the field 568
16.5.4 The backpropagation algorithm 569
16.5.5 Identifiability 572
16.5.6 Regularization 572
16.5.7 Bayesian inference * 576
16.6 Ensemble learning 580
16.6.1 Stacking 580
16.6.2 Error-correcting output codes 581
16.6.3 Ensemble learning is not equivalent to Bayes model averaging 581
16.7 Experimental comparison 582
16.7.1 Low-dimensional features 582
16.7.2 High-dimensional features 583
16.8 Interpreting black-box models 585

***17 Markov and hidden Markov models* 589**

17.1 Introduction 589
17.2 Markov models 589
17.2.1 Transition matrix 589
17.2.2 Application: Language modeling 591
17.2.3 Stationary distribution of a Markov chain * 596
17.2.4 Application: Google's PageRank algorithm for web page ranking * 600
17.3 Hidden Markov models 603
17.3.1 Applications of HMMs 604
17.4 Inference in HMMs 606
17.4.1 Types of inference problems for temporal models 606
17.4.2 The forwards algorithm 609
17.4.3 The forwards-backwards algorithm 610
17.4.4 The Viterbi algorithm 612
17.4.5 Forwards filtering, backwards sampling 616

17.5 Learning for HMMs 617
17.5.1 Training with fully observed data 617
17.5.2 EM for HMMs (the Baum-Welch algorithm) 618
17.5.3 Bayesian methods for "fitting" HMMs * 620
17.5.4 Discriminative training 620
17.5.5 Model selection 621
17.6 Generalizations of HMMs 621
17.6.1 Variable duration (semi-Markov) HMMs 622
17.6.2 Hierarchical HMMs 624
17.6.3 Input-output HMMs 625
17.6.4 Auto-regressive and buried HMMs 626
17.6.5 Factorial HMM 627
17.6.6 Coupled HMM and the influence model 628
17.6.7 Dynamic Bayesian networks (DBNs) 628

***18 State space models* 631**

18.1 Introduction 631
18.2 Applications of SSMs 632
18.2.1 SSMs for object tracking 632
18.2.2 Robotic SLAM 633
18.2.3 Online parameter learning using recursive least squares 636
18.2.4 SSM for time series forecasting * 637
18.3 Inference in LG-SSM 640
18.3.1 The Kalman filtering algorithm 640
18.3.2 The Kalman smoothing algorithm 643
18.4 Learning for LG-SSM 646
18.4.1 Identifiability and numerical stability 646
18.4.2 Training with fully observed data 647
18.4.3 EM for LG-SSM 647
18.4.4 Subspace methods 647
18.4.5 Bayesian methods for "fitting" LG-SSMs 647
18.5 Approximate online inference for non-linear, non-Gaussian SSMs 647
18.5.1 Extended Kalman filter (EKF) 648
18.5.2 Unscented Kalman filter (UKF) 650
18.5.3 Assumed density filtering (ADF) 652
18.6 Hybrid discrete/continuous SSMs 655
18.6.1 Inference 656
18.6.2 Application: data association and multi-target tracking 658
18.6.3 Application: fault diagnosis 659
18.6.4 Application: econometric forecasting 660

***19 Undirected graphical models (Markov random fields)* 661**

19.1 Introduction 661
19.2 Conditional independence properties of UGMs 661
19.2.1 Key properties 661

19.2.2 An undirected alternative to d-separation 663
19.2.3 Comparing directed and undirected graphical models 664
19.3 Parameterization of MRFs 665
19.3.1 The Hammersley-Clifford theorem 665
19.3.2 Representing potential functions 667
19.4 Examples of MRFs 668
19.4.1 Ising model 668
19.4.2 Hopfield networks 669
19.4.3 Potts model 671
19.4.4 Gaussian MRFs 672
19.4.5 Markov logic networks * 674
19.5 Learning 676
19.5.1 Training maxent models using gradient methods 676
19.5.2 Training partially observed maxent models 677
19.5.3 Approximate methods for computing the MLEs of MRFs 678
19.5.4 Pseudo likelihood 678
19.5.5 Stochastic maximum likelihood 679
19.5.6 Feature induction for maxent models * 680
19.5.7 Iterative proportional fitting (IPF) * 681
19.6 Conditional random fields (CRFs) 684
19.6.1 Chain-structured CRFs, MEMMs and the label-bias problem 684
19.6.2 Applications of CRFs 686
19.6.3 CRF training 692
19.7 Structural SVMs 693
19.7.1 SSVMs: a probabilistic view 693
19.7.2 SSVMs: a non-probabilistic view 695
19.7.3 Cutting plane methods for fitting SSVMs 698
19.7.4 Online algorithms for fitting SSVMs 700
19.7.5 Latent structural SVMs 701

***20 Exact inference for graphical models* 707**

20.1 Introduction 707
20.2 Belief propagation for trees 707
20.2.1 Serial protocol 707
20.2.2 Parallel protocol 709
20.2.3 Gaussian BP * 710
20.2.4 Other BP variants * 712
20.3 The variable elimination algorithm 714
20.3.1 The generalized distributive law * 717
20.3.2 Computational complexity of VE 717
20.3.3 A weakness of VE 720
20.4 The junction tree algorithm * 720
20.4.1 Creating a junction tree 720
20.4.2 Message passing on a junction tree 722
20.4.3 Computational complexity of JTA 725

20.4.4 JTA generalizations * 726
20.5 Computational intractability of exact inference in the worst case 726
20.5.1 Approximate inference 727

***21 Variational inference* 731**
21.1 Introduction 731
21.2 Variational inference 732
21.2.1 Alternative interpretations of the variational objective 733
21.2.2 Forward or reverse KL? * 733
21.3 The mean field method 735
21.3.1 Derivation of the mean field update equations 736
21.3.2 Example: mean field for the Ising model 737
21.4 Structured mean field * 739
21.4.1 Example: factorial HMM 740
21.5 Variational Bayes 742
21.5.1 Example: VB for a univariate Gaussian 742
21.5.2 Example: VB for linear regression 746
21.6 Variational Bayes EM 749
21.6.1 Example: VBEM for mixtures of Gaussians * 750
21.7 Variational message passing and VIBES 756
21.8 Local variational bounds * 756
21.8.1 Motivating applications 756
21.8.2 Bohning's quadratic bound to the log-sum-exp function 758
21.8.3 Bounds for the sigmoid function 760
21.8.4 Other bounds and approximations to the log-sum-exp function * 762
21.8.5 Variational inference based on upper bounds 763

***22 More variational inference* 767**
22.1 Introduction 767
22.2 Loopy belief propagation: algorithmic issues 767
22.2.1 A brief history 767
22.2.2 LBP on pairwise models 768
22.2.3 LBP on a factor graph 769
22.2.4 Convergence 771
22.2.5 Accuracy of LBP 774
22.2.6 Other speedup tricks for LBP * 775
22.3 Loopy belief propagation: theoretical issues * 776
22.3.1 UGMs represented in exponential family form 776
22.3.2 The marginal polytope 777
22.3.3 Exact inference as a variational optimization problem 778
22.3.4 Mean field as a variational optimization problem 779
22.3.5 LBP as a variational optimization problem 779
22.3.6 Loopy BP vs mean field 783
22.4 Extensions of belief propagation * 783
22.4.1 Generalized belief propagation 783

22.4.2 Convex belief propagation 785
22.5 Expectation propagation 787
22.5.1 EP as a variational inference problem 788
22.5.2 Optimizing the EP objective using moment matching 789
22.5.3 EP for the clutter problem 791
22.5.4 LBP is a special case of EP 792
22.5.5 Ranking players using TrueSkill 793
22.5.6 Other applications of EP 799
22.6 MAP state estimation 799
22.6.1 Linear programming relaxation 799
22.6.2 Max-product belief propagation 800
22.6.3 Graphcuts 801
22.6.4 Experimental comparison of graphcuts and BP 804
22.6.5 Dual decomposition 806

***23 Monte Carlo inference* 815**

23.1 Introduction 815
23.2 Sampling from standard distributions 815
23.2.1 Using the cdf 815
23.2.2 Sampling from a Gaussian (Box-Muller method) 817
23.3 Rejection sampling 817
23.3.1 Basic idea 817
23.3.2 Example 818
23.3.3 Application to Bayesian statistics 819
23.3.4 Adaptive rejection sampling 819
23.3.5 Rejection sampling in high dimensions 820
23.4 Importance sampling 820
23.4.1 Basic idea 820
23.4.2 Handling unnormalized distributions 821
23.4.3 Importance sampling for a DGM: likelihood weighting 822
23.4.4 Sampling importance resampling (SIR) 822
23.5 Particle filtering 823
23.5.1 Sequential importance sampling 824
23.5.2 The degeneracy problem 825
23.5.3 The resampling step 825
23.5.4 The proposal distribution 827
23.5.5 Application: robot localization 828
23.5.6 Application: visual object tracking 828
23.5.7 Application: time series forecasting 831
23.6 Rao-Blackwellised particle filtering (RBPF) 831
23.6.1 RBPF for switching LG-SSMs 831
23.6.2 Application: tracking a maneuvering target 832
23.6.3 Application: Fast SLAM 834

***24 Markov chain Monte Carlo (MCMC) inference* 837**

24.1 Introduction 837
24.2 Gibbs sampling 838
24.2.1 Basic idea 838
24.2.2 Example: Gibbs sampling for the Ising model 838
24.2.3 Example: Gibbs sampling for inferring the parameters of a GMM 840
24.2.4 Collapsed Gibbs sampling * 841
24.2.5 Gibbs sampling for hierarchical GLMs 844
24.2.6 BUGS and JAGS 846
24.2.7 The Imputation Posterior (IP) algorithm 847
24.2.8 Blocking Gibbs sampling 847
24.3 Metropolis Hastings algorithm 848
24.3.1 Basic idea 848
24.3.2 Gibbs sampling is a special case of MH 849
24.3.3 Proposal distributions 850
24.3.4 Adaptive MCMC 853
24.3.5 Initialization and mode hopping 854
24.3.6 Why MH works * 854
24.3.7 Reversible jump (trans-dimensional) MCMC * 855
24.4 Speed and accuracy of MCMC 856
24.4.1 The burn-in phase 856
24.4.2 Mixing rates of Markov chains * 857
24.4.3 Practical convergence diagnostics 858
24.4.4 Accuracy of MCMC 860
24.4.5 How many chains? 862
24.5 Auxiliary variable MCMC * 863
24.5.1 Auxiliary variable sampling for logistic regression 863
24.5.2 Slice sampling 864
24.5.3 Swendsen Wang 866
24.5.4 Hybrid/Hamiltonian MCMC * 868
24.6 Annealing methods 868
24.6.1 Simulated annealing 869
24.6.2 Annealed importance sampling 871
24.6.3 Parallel tempering 871
24.7 Approximating the marginal likelihood 872
24.7.1 The candidate method 872
24.7.2 Harmonic mean estimate 872
24.7.3 Annealed importance sampling 873

***25 Clustering* 875**

25.1 Introduction 875
25.1.1 Measuring (dis)similarity 875
25.1.2 Evaluating the output of clustering methods * 876
25.2 Dirichlet process mixture models 879
25.2.1 From finite to infinite mixture models 879
25.2.2 The Dirichlet process 882

25.2.3 Applying Dirichlet processes to mixture modeling 885
25.2.4 Fitting a DP mixture model 886
25.3 Affinity propagation 887
25.4 Spectral clustering 890
25.4.1 Graph Laplacian 891
25.4.2 Normalized graph Laplacian 892
25.4.3 Example 893
25.5 Hierarchical clustering 893
25.5.1 Agglomerative clustering 895
25.5.2 Divisive clustering 898
25.5.3 Choosing the number of clusters 899
25.5.4 Bayesian hierarchical clustering 899
25.6 Clustering datapoints and features 901
25.6.1 Biclustering 903
25.6.2 Multi-view clustering 903

***26 Graphical model structure learning* 907**

26.1 Introduction 907
26.2 Structure learning for knowledge discovery 908
26.2.1 Relevance networks 908
26.2.2 Dependency networks 909
26.3 Learning tree structures 910
26.3.1 Directed or undirected tree? 911
26.3.2 Chow-Liu algorithm for finding the ML tree structure 912
26.3.3 Finding the MAP forest 912
26.3.4 Mixtures of trees 914
26.4 Learning DAG structures 914
26.4.1 Markov equivalence 914
26.4.2 Exact structural inference 916
26.4.3 Scaling up to larger graphs 920
26.5 Learning DAG structure with latent variables 922
26.5.1 Approximating the marginal likelihood when we have missing data 922
26.5.2 Structural EM 925
26.5.3 Discovering hidden variables 926
26.5.4 Case study: Google’s Rephil 928
26.5.5 Structural equation models * 929
26.6 Learning causal DAGs 931
26.6.1 Causal interpretation of DAGs 931
26.6.2 Using causal DAGs to resolve Simpson’s paradox 933
26.6.3 Learning causal DAG structures 935
26.7 Learning undirected Gaussian graphical models 938
26.7.1 MLE for a GGM 938
26.7.2 Graphical lasso 939
26.7.3 Bayesian inference for GGM structure * 941
26.7.4 Handling non-Gaussian data using copulas * 942

26.8 Learning undirected discrete graphical models 942
26.8.1 Graphical lasso for MRFs/CRFs 942
26.8.2 Thin junction trees 944

27 Latent variable models for discrete data **945**

27.1 Introduction 945
27.2 Distributed state LVMs for discrete data 946
27.2.1 Mixture models 946
27.2.2 Exponential family PCA 947
27.2.3 LDA and mPCA 948
27.2.4 GaP model and non-negative matrix factorization 949
27.3 Latent Dirichlet allocation (LDA) 950
27.3.1 Basics 950
27.3.2 Unsupervised discovery of topics 953
27.3.3 Quantitatively evaluating LDA as a language model 953
27.3.4 Fitting using (collapsed) Gibbs sampling 955
27.3.5 Example 956
27.3.6 Fitting using batch variational inference 957
27.3.7 Fitting using online variational inference 959
27.3.8 Determining the number of topics 960
27.4 Extensions of LDA 961
27.4.1 Correlated topic model 961
27.4.2 Dynamic topic model 962
27.4.3 LDA-HMM 963
27.4.4 Supervised LDA 967
27.5 LVMs for graph-structured data 970
27.5.1 Stochastic block model 971
27.5.2 Mixed membership stochastic block model 973
27.5.3 Relational topic model 974
27.6 LVMs for relational data 975
27.6.1 Infinite relational model 976
27.6.2 Probabilistic matrix factorization for collaborative filtering 979
27.7 Restricted Boltzmann machines (RBMs) 983
27.7.1 Varieties of RBMs 985
27.7.2 Learning RBMs 987
27.7.3 Applications of RBMs 991

28 Deep learning **995**

28.1 Introduction 995
28.2 Deep generative models 995
28.2.1 Deep directed networks 996
28.2.2 Deep Boltzmann machines 996
28.2.3 Deep belief networks 997
28.2.4 Greedy layer-wise learning of DBNs 998
28.3 Deep neural networks 999

28.3.1 Deep multi-layer perceptrons 999
28.3.2 Deep auto-encoders 1000
28.3.3 Stacked denoising auto-encoders 1001
28.4 Applications of deep networks 1001
28.4.1 Handwritten digit classification using DBNs 1001
28.4.2 Data visualization and feature discovery using deep auto-encoders 1002
28.4.3 Information retrieval using deep auto-encoders (semantic hashing) 1003
28.4.4 Learning audio features using 1d convolutional DBNs 1004
28.4.5 Learning image features using 2d convolutional DBNs 1005
28.5 Discussion 1005

***Notation* 1009**

***Bibliography* 1015**

***Indexes* 1047**

Index to code 1047
Index to keywords 1050

Preface

Introduction

With the ever increasing amounts of data in electronic form, the need for automated methods for data analysis continues to grow. The goal of machine learning is to develop methods that can automatically detect patterns in data, and then to use the uncovered patterns to predict future data or other outcomes of interest. Machine learning is thus closely related to the fields of statistics and data mining, but differs slightly in terms of its emphasis and terminology. This book provides a detailed introduction to the field, and includes worked examples drawn from application domains such as molecular biology, text processing, computer vision, and robotics.

Target audience

This book is suitable for upper-level undergraduate students and beginning graduate students in computer science, statistics, electrical engineering, econometrics, or any one else who has the appropriate mathematical background. Specifically, the reader is assumed to already be familiar with basic multivariate calculus, probability, linear algebra, and computer programming. Prior exposure to statistics is helpful but not necessary.

A probabilistic approach

This books adopts the view that the best way to make machines that can learn from data is to use the tools of probability theory, which has been the mainstay of statistics and engineering for centuries. Probability theory can be applied to any problem involving uncertainty. In machine learning, uncertainty comes in many forms: what is the best prediction (or decision) given some data? what is the best model given some data? what measurement should I perform next? etc.

The systematic application of probabilistic reasoning to all inferential problems, including inferring parameters of statistical models, is sometimes called a Bayesian approach. However, this term tends to elicit very strong reactions (either positive or negative, depending on who you ask), so we prefer the more neutral term “probabilistic approach”. Besides, we will often use techniques such as maximum likelihood estimation, which are not Bayesian methods, but certainly fall within the probabilistic paradigm.

Rather than describing a cookbook of different heuristic methods, this book stresses a principled model-based approach to machine learning. For any given model, a variety of algorithms can often be applied. Conversely, any given algorithm can often be applied to a variety of models. This kind of modularity, where we distinguish model from algorithm, is good pedagogy and good engineering.

We will often use the language of graphical models to specify our models in a concise and intuitive way. In addition to aiding comprehension, the graph structure aids in developing efficient algorithms, as we will see. However, this book is not primarily about graphical models; it is about probabilistic modeling in general.

A practical approach

Nearly all of the methods described in this book have been implemented in a MATLAB software package called **PMTK**, which stands for probabilistic modeling toolkit. This is freely available from `pmtk3.googlecode.com` (the digit 3 refers to the third edition of the toolkit, which is the one used in this version of the book). There are also a variety of supporting files, written by other people, available at `pmtksupport.googlecode.com`. These will be downloaded automatically, if you follow the setup instructions described on the PMTK website.

MATLAB is a high-level, interactive scripting language ideally suited to numerical computation and data visualization, and can be purchased from `www.mathworks.com`. Some of the code requires the Statistics toolbox, which needs to be purchased separately. There is also a free version of Matlab called **Octave**, available at `http://www.gnu.org/software/octave/`, which supports most of the functionality of MATLAB. Some (but not all) of the code in this book also works in Octave. See the PMTK website for details.

PMTK was used to generate many of the figures in this book; the source code for these figures is included on the PMTK website, allowing the reader to easily see the effects of changing the data or algorithm or parameter settings. The book refers to files by name, e.g., `naiveBayesFit`. In order to find the corresponding file, you can use two methods: within Matlab you can type `which naiveBayesFit` and it will return the full path to the file; or, if you do not have Matlab but want to read the source code anyway, you can use your favorite search engine, which should return the corresponding file from the `pmtk3.googlecode.com` website.

Details on *how to use* PMTK can be found on the website, which will be udpated over time. Details on the *underlying theory* behind these methods can be found in this book.

Acknowledgments

A book this large is obviously a team effort. I would especially like to thank the following people: my wife Margaret, for keeping the home fires burning as I toiled away in my office for the last six years; Matt Dunham, who created many of the figures in this book, and who wrote much of the code in PMTK; Baback Moghaddam, who gave extremely detailed feedback on every page of an earlier draft of the book; Chris Williams, who also gave very detailed feedback; Cody Severinski and Wei-Lwun Lu, who assisted with figures; generations of UBC students, who gave helpful comments on earlier drafts; Daphne Koller, Nir Friedman, and Chris Manning, for letting me use their latex style files; Stanford University, Google Research and Skyline College for hosting me during part of my sabbatical; and various Canadian funding agencies (NSERC, CRC and CIFAR) who have supported me financially over the years.

In addition, I would like to thank the following people for giving me helpful feedback on

parts of the book, and/or for sharing figures, code, exercises or even (in some cases) text: David Blei, Hannes Bretschneider, Greg Corrado, Arnaud Doucet, Mario Figueiredo, Nando de Freitas, Mark Girolami, Gabriel Goh, Tom Griffiths, Katherine Heller, Geoff Hinton, Aapo Hyvarinen, Tommi Jaakkola, Mike Jordan, Charles Kemp, Emtiyaz Khan, Bonnie Kirkpatrick, Daphne Koller, Zico Kolter, Honglak Lee, Julien Mairal, Andrew McPherson, Tom Minka, Ian Nabney, Arthur Pope, Carl Rassmussen, Ryan Rifkin, Ruslan Salakhutdinov, Mark Schmidt, Daniel Selsam, David Sontag, Erik Sudderth, Josh Tenenbaum, Kai Yu, Martin Wainwright, Yair Weiss.

Kevin Patrick Murphy
Palo Alto, California
June 2012

1 *Introduction*

1.1 Machine learning: what and why?

We are drowning in information and starving for knowledge. — John Naisbitt.

We are entering the era of **big data**. For example, there are about 1 trillion web pages[1]; one hour of video is uploaded to YouTube every second, amounting to 10 years of content every day[2]; the genomes of 1000s of people, each of which has a length of 3.8×10^9 base pairs, have been sequenced by various labs; Walmart handles more than 1M transactions per hour and has databases containing more than 2.5 petabytes (2.5×10^{15}) of information (Cukier 2010); and so on.

This deluge of data calls for automated methods of data analysis, which is what **machine learning** provides. In particular, we define machine learning as a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty (such as planning how to collect more data!).

This books adopts the view that the best way to solve such problems is to use the tools of probability theory. Probability theory can be applied to any problem involving uncertainty. In machine learning, uncertainty comes in many forms: what is the best prediction about the future given some past data? what is the best model to explain some data? what measurement should I perform next? etc. The probabilistic approach to machine learning is closely related to the field of statistics, but differs slightly in terms of its emphasis and terminology[3].

We will describe a wide variety of probabilistic models, suitable for a wide variety of data and tasks. We will also describe a wide variety of algorithms for learning and using such models. The goal is not to develop a cook book of ad hoc techiques, but instead to present a unified view of the field through the lens of probabilistic modeling and inference. Although we will pay attention to computational efficiency, details on how to scale these methods to truly massive datasets are better described in other books, such as (Rajaraman and Ullman 2011; Bekkerman et al. 2011).

1. `http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html`
2. Source: `http://www.youtube.com/t/press_statistics`.
3. Rob Tibshirani, a statistician at Stanford university, has created an amusing comparison between machine learning and statistics, available at `http://www-stat.stanford.edu/~tibs/stat315a/glossary.pdf`.

It should be noted, however, that even when one has an apparently massive data set, the effective number of data points for certain cases of interest might be quite small. In fact, data across a variety of domains exhibits a property known as the **long tail**, which means that a few things (e.g., words) are very common, but most things are quite rare (see Section 2.4.6 for details). For example, 20% of Google searches each day have never been seen before[4]. This means that the core statistical issues that we discuss in this book, concerning generalizing from relatively small samples sizes, are still very relevant even in the big data era.

1.1.1 Types of machine learning

Machine learning is usually divided into two main types. In the **predictive** or **supervised learning** approach, the goal is to learn a mapping from inputs $\mathbf{x}$ to outputs y, given a labeled set of input-output pairs $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$. Here $\mathcal{D}$ is called the **training set**, and N is the number of training examples.

In the simplest setting, each training input $\mathbf{x}_i$ is a D-dimensional vector of numbers, representing, say, the height and weight of a person. These are called **features**, **attributes** or **covariates**. In general, however, $\mathbf{x}_i$ could be a complex structured object, such as an image, a sentence, an email message, a time series, a molecular shape, a graph, etc.

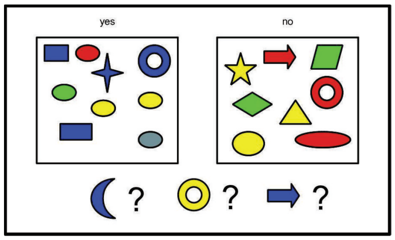
Similarly the form of the output or **response variable** can in principle be anything, but most methods assume that y_i is a **categorical** or **nominal** variable from some finite set, $y_i \in \{1, \ldots, C\}$ (such as male or female), or that y_i is a real-valued scalar (such as income level). When y_i is categorical, the problem is known as **classification** or **pattern recognition**, and when y_i is real-valued, the problem is known as **regression**. Another variant, known as **ordinal regression**, occurs where label space $\mathcal{Y}$ has some natural ordering, such as grades A–F.

The second main type of machine learning is the **descriptive** or **unsupervised learning** approach. Here we are only given inputs, $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N$, and the goal is to find "interesting patterns" in the data. This is sometimes called **knowledge discovery**. This is a much less well-defined problem, since we are not told what kinds of patterns to look for, and there is no obvious error metric to use (unlike supervised learning, where we can compare our prediction of y for a given $\mathbf{x}$ to the observed value).

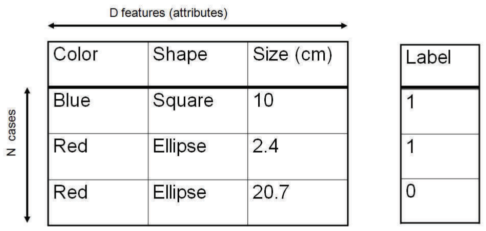
There is a third type of machine learning, known as **reinforcement learning**, which is somewhat less commonly used. This is useful for learning how to act or behave when given occasional reward or punishment signals. (For example, consider how a baby learns to walk.) Unfortunately, RL is beyond the scope of this book, although we do discuss decision theory in Section 5.7, which is the basis of RL. See e.g., (Kaelbling et al. 1996; Sutton and Barto 1998; Russell and Norvig 2010; Szepesvari 2010; Wiering and van Otterlo 2012) for more information on RL.

---

4. `http://certifiedknowledge.org/blog/are-search-queries-becoming-even-more-unique-statistics-from-google`.



(a)


D features (attributes)
N cases

Color	Shape	Size (cm)	Label
Blue	Square	10	1
Red	Ellipse	2.4	1
Red	Ellipse	20.7	0

(b)

Figure 1.1 Left: Some labeled training examples of colored shapes, along with 3 unlabeled test cases. Right: Representing the training data as an $N \times D$ design matrix. Row i represents the feature vector $\mathbf{x}_i$. The last column is the label, $y_i \in \{0, 1\}$. Based on a figure by Leslie Kaelbling.

1.2 Supervised learning

We begin our investigation of machine learning by discussing supervised learning, which is the form of ML most widely used in practice.

1.2.1 Classification

In this section, we discuss classification. Here the goal is to learn a mapping from inputs $\mathbf{x}$ to outputs y, where $y \in \{1, \ldots, C\}$, with C being the number of classes. If $C = 2$, this is called **binary classification** (in which case we often assume $y \in \{0, 1\}$); if $C > 2$, this is called **multiclass classification**. If the class labels are not mutually exclusive (e.g., somebody may be classified as tall and strong), we call it **multi-label classification**, but this is best viewed as predicting multiple related binary class labels (a so-called **multiple output model**). When we use the term “classification”, we will mean multiclass classification with a single output, unless we state otherwise.

One way to formalize the problem is as **function approximation**. We assume $y = f(\mathbf{x})$ for some unknown function f, and the goal of learning is to estimate the function f given a labeled training set, and then to make predictions using $\hat{y} = \hat{f}(\mathbf{x})$. (We use the hat symbol to denote an estimate.) Our main goal is to make predictions on novel inputs, meaning ones that we have not seen before (this is called **generalization**), since predicting the response on the training set is easy (we can just look up the answer).

1.2.1.1 Example

As a simple toy example of classification, consider the problem illustrated in Figure 1.1(a). We have two classes of object which correspond to labels 0 and 1. The inputs are colored shapes. These have been described by a set of D features or attributes, which are stored in an $N \times D$ design matrix $\mathbf{X}$, shown in Figure 1.1(b). The input features $\mathbf{x}$ can be discrete, continuous or a combination of the two. In addition to the inputs, we have a vector of training labels $\mathbf{y}$.

In Figure 1.1, the test cases are a blue crescent, a yellow circle and a blue arrow. None of these have been seen before. Thus we are required to **generalize** beyond the training set. A reasonable guess is that blue crescent should be $y = 1$, since all blue shapes are labeled 1 in the training set. The yellow circle is harder to classify, since some yellow things are labeled $y = 1$ and some are labeled $y = 0$, and some circles are labeled $y = 1$ and some $y = 0$. Consequently it is not clear what the right label should be in the case of the yellow circle. Similarly, the correct label for the blue arrow is unclear.

1.2.1.2 The need for probabilistic predictions

To handle ambiguous cases, such as the yellow circle above, it is desirable to return a probability. The reader is assumed to already have some familiarity with basic concepts in probability. If not, please consult Chapter 2 for a refresher, if necessary.

We will denote the probability distribution over possible labels, given the input vector $\mathbf{x}$ and training set $\mathcal{D}$ by $p(y|\mathbf{x}, \mathcal{D})$. In general, this represents a vector of length C. (If there are just two classes, it is sufficient to return the single number $p(y = 1|\mathbf{x}, \mathcal{D})$, since $p(y = 1|\mathbf{x}, \mathcal{D}) + p(y = 0|\mathbf{x}, \mathcal{D}) = 1$.) In our notation, we make explicit that the probability is conditional on the test input $\mathbf{x}$, as well as the training set $\mathcal{D}$, by putting these terms on the right hand side of the conditioning bar $|$. We are also implicitly conditioning on the form of model that we use to make predictions. When choosing between different models, we will make this assumption explicit by writing $p(y|\mathbf{x}, \mathcal{D}, M)$, where M denotes the model. However, if the model is clear from context, we will drop M from our notation for brevity.

Given a probabilistic output, we can always compute our "best guess" as to the "true label" using

$$\hat{y} = \hat{f}(\mathbf{x}) = \operatorname*{argmax}_{c=1}^{C} p(y = c|\mathbf{x}, \mathcal{D}) \tag{1.1}$$

This corresponds to the most probable class label, and is called the **mode** of the distribution $p(y|\mathbf{x}, \mathcal{D})$; it is also known as a **MAP estimate** (MAP stands for **maximum a posteriori**). Using the most probable label makes intuitive sense, but we will give a more formal justification for this procedure in Section 5.7.

Now consider a case such as the yellow circle, where $p(\hat{y}|\mathbf{x}, \mathcal{D})$ is far from 1.0. In such a case we are not very confident of our answer, so it might be better to say "I don't know" instead of returning an answer that we don't really trust. This is particularly important in domains such as medicine and finance where we may be risk averse, as we explain in Section 5.7. Another application where it is important to assess risk is when playing TV game shows, such as Jeopardy. In this game, contestants have to solve various word puzzles and answer a variety of trivia questions, but if they answer incorrectly, they lose money. In 2011, IBM unveiled a computer system called Watson which beat the top human Jeopardy champion. Watson uses a variety of interesting techniques (Ferrucci et al. 2010), but the most pertinent one for our present purposes is that it contains a module that estimates how confident it is of its answer. The system only chooses to "buzz in" its answer if sufficiently confident it is correct. Similarly, Google has a system known as SmartASS (ad selection system) that predicts the probability you will click on an ad based on your search history and other user and ad-specific features (Metz 2010). This probability is known as the **click-through rate** or **CTR**, and can be used to maximize expected profit. We will discuss some of the basic principles behind systems such as SmartASS later in this book.

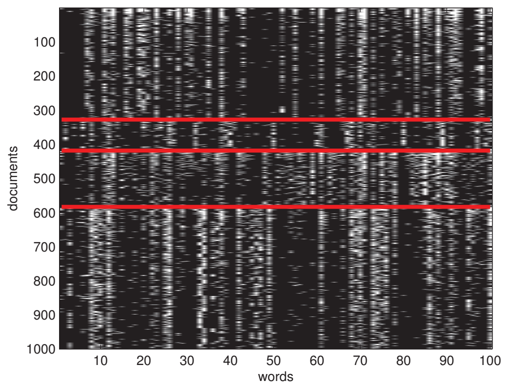


Figure 1.2 Subset of size 16242 x 100 of the 20-newsgroups data. We only show 1000 rows, for clarity. Each row is a document (represented as a bag-of-words bit vector), each column is a word. The red lines separate the 4 classes, which are (in descending order) comp, rec, sci, talk (these are the titles of USENET groups). We can see that there are subsets of words whose presence or absence is indicative of the class. The data is available from `http://cs.nyu.edu/~roweis/data.html`. Figure generated by `newsgroupsVisualize`.

1.2.1.3 Real-world applications

Classification is probably the most widely used form of machine learning, and has been used to solve many interesting and often difficult real-world problems. We have already mentioned some important applciations. We give a few more examples below.

Document classification and email spam filtering

In **document classification**, the goal is to classify a document, such as a web page or email message, into one of C classes, that is, to compute $p(y = c|\mathbf{x}, \mathcal{D})$, where $\mathbf{x}$ is some representation of the text. A special case of this is **email spam filtering**, where the classes are spam $y = 1$ or ham $y = 0$.

Most classifiers assume that the input vector $\mathbf{x}$ has a fixed size. A common way to represent variable-length documents in feature-vector format is to use a **bag of words** representation. This is explained in detail in Section 3.4.4.1, but the basic idea is to define $x_{ij} = 1$ iff word j occurs in document i. If we apply this transformation to every document in our data set, we get a binary document $\times$ word co-occurrence matrix: see Figure 1.2 for an example. Essentially the document classification problem has been reduced to one that looks for subtle changes in the pattern of bits. For example, we may notice that most spam messages have a high probability of containing the words “buy”, “cheap”, “viagra”, etc. In Exercise 8.1 and Exercise 8.2, you will get hands-on experience applying various classification techniques to the spam filtering problem.

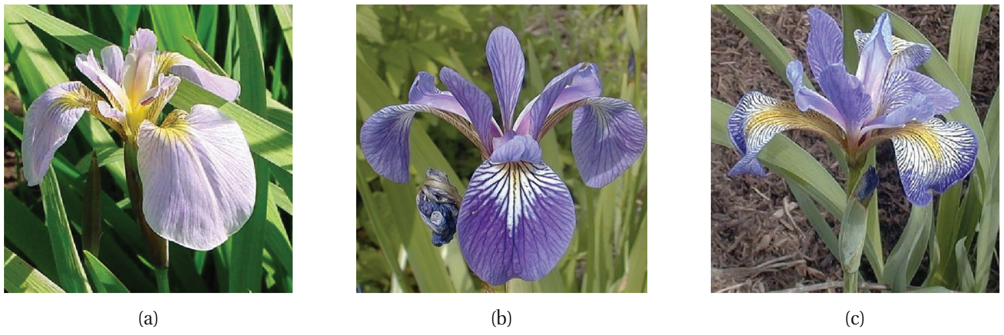


Figure 1.3 Three types of iris flowers: setosa, versicolor and virginica. Source: `http://www.statlab.uni-heidelberg.de/data/iris/` . Used with kind permission of Dennis Kramb and SIGNA.

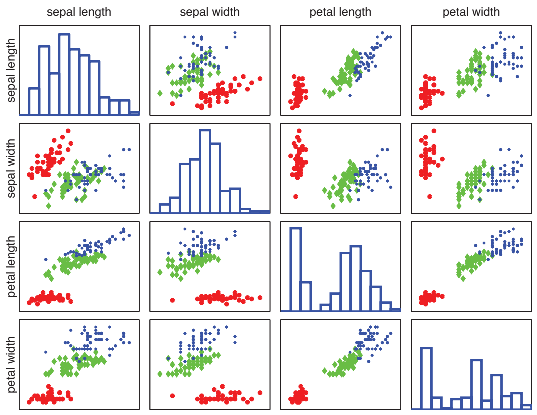


Figure 1.4 Visualization of the Iris data as a pairwise scatter plot. The diagonal plots the marginal histograms of the 4 features. The off diagonals contain scatterplots of all possible pairs of features. Red circle = setosa, green diamond = versicolor, blue star = virginica. Figure generated by `fisheririsDemo`.

Classifying flowers

Figure 1.3 gives another example of classification, due to the statistician Ronald Fisher. The goal is to learn to distinguish three different kinds of iris flower, called setosa, versicolor and virginica. Fortunately, rather than working directly with images, a botanist has already extracted 4 useful features or characteristics: sepal length and width, and petal length and width. (Such **feature extraction** is an important, but difficult, task. Most machine learning methods use features chosen by some human. Later we will discuss some methods that can learn good features from the data.) If we make a **scatter plot** of the iris data, as in Figure 1.4, we see that it is easy to distinguish setosas (red circles) from the other two classes by just checking if their petal length

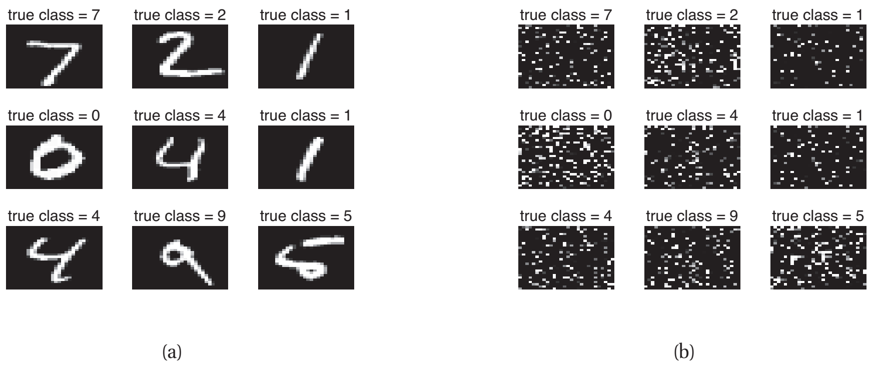


Figure 1.5 (a) First 9 test MNIST gray-scale images. (b) Same as (a), but with the features permuted randomly. Classification performance is identical on both versions of the data (assuming the training data is permuted in an identical way). Figure generated by `shuffledDigitsDemo`.

or width is below some threshold. However, distinguishing versicolor from virginica is slightly harder; any decision will need to be based on at least two features. (It is always a good idea to perform **exploratory data analysis**, such as plotting the data, before applying a machine learning method.)

Image classification and handwriting recognition

Now consider the harder problem of classifying images directly, where a human has not pre-processed the data. We might want to classify the image as a whole, e.g., is it an indoors or outdoors scene? is it a horizontal or vertical photo? does it contain a dog or not? This is called **image classification**.

In the special case that the images consist of isolated handwritten letters and digits, for example, in a postal or ZIP code on a letter, we can use classification to perform **handwriting recognition**. A standard dataset used in this area is known as **MNIST**, which stands for "Modified National Institute of Standards"[5]. (The term "modified" is used because the images have been preprocessed to ensure the digits are mostly in the center of the image.) This dataset contains 60,000 training images and 10,000 test images of the digits 0 to 9, as written by various people. The images are size 28×28 and have grayscale values in the range $0 : 255$. See Figure 1.5(a) for some example images.

Many generic classification methods ignore any structure in the input features, such as spatial layout. Consequently, they can also just as easily handle data that looks like Figure 1.5(b), which is the same data except we have randomly permuted the order of all the features. (You will verify this in Exercise 1.1.) This flexibility is both a blessing (since the methods are general purpose) and a curse (since the methods ignore an obviously useful source of information). We will discuss methods for exploiting structure in the input features later in the book.

5. Available from `http://yann.lecun.com/exdb/mnist/`.

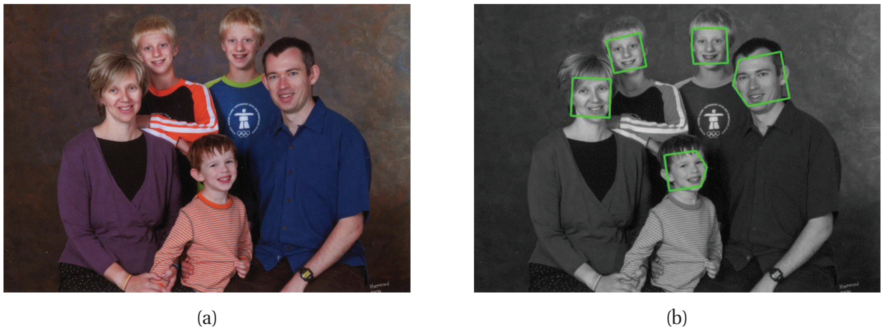


Figure 1.6 Example of face detection. (a) Input image (Murphy family, photo taken 5 August 2010). Used with kind permission of Bernard Diedrich of Sherwood Studios. (b) Output of classifier, which detected 5 faces at different poses. This was produced using the online demo at `http://demo.pittpatt.com/`. The classifier was trained on 1000s of manually labeled images of faces and non-faces, and then was applied to a dense set of overlapping patches in the test image. Only the patches whose probability of containing a face was sufficiently high were returned. Used with kind permission of Pittpatt.com

Face detection and recognition

A harder problem is to find objects within an image; this is called **object detection** or **object localization**. An important special case of this is **face detection**. One approach to this problem is to divide the image into many small overlapping patches at different locations, scales and orientations, and to classify each such patch based on whether it contains face-like texture or not. This is called a **sliding window detector**. The system then returns those locations where the probability of face is sufficiently high. See Figure 1.6 for an example. Such face detection systems are built-in to most modern digital cameras; the locations of the detected faces are used to determine the center of the auto-focus. Another application is automatically blurring out faces in Google's StreetView system.

Having found the faces, one can then proceed to perform **face recognition**, which means estimating the identity of the person (see Figure 1.10(a)). In this case, the number of class labels might be very large. Also, the features one should use are likely to be different than in the face detection problem: for recognition, subtle differences between faces such as hairstyle may be important for determining identity, but for detection, it is important to be **invariant** to such details, and to just focus on the differences between faces and non-faces. For more information about visual object detection, see e.g., (Szeliski 2010).

1.2.2 Regression

Regression is just like classification except the response variable is continuous. Figure 1.7 shows a simple example: we have a single real-valued input $x_i \in \mathbb{R}$, and a single real-valued response $y_i \in \mathbb{R}$. We consider fitting two models to the data: a straight line and a quadratic function. (We explain how to fit such models below.) Various extensions of this basic problem can arise, such as having high-dimensional inputs, outliers, non-smooth responses, etc. We will discuss ways to handle such problems later in the book.

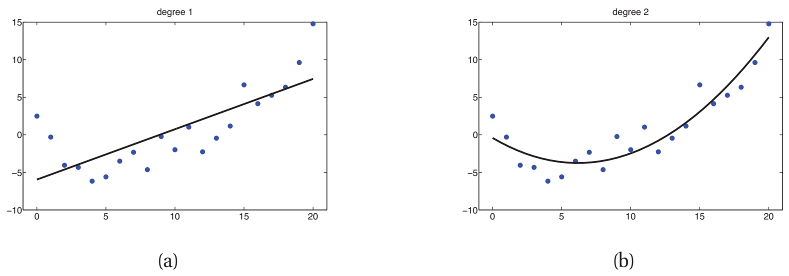


Figure 1.7 (a) Linear regression on some 1d data. (b) Same data with polynomial regression (degree 2). Figure generated by `linregPolyVsDegree`.

Here are some examples of real-world regression problems.

- Predict tomorrow's stock market price given current market conditions and other possible side information.
- Predict the age of a viewer watching a given video on YouTube.
- Predict the location in 3d space of a robot arm end effector, given control signals (torques) sent to its various motors.
- Predict the amount of prostate specific antigen (PSA) in the body as a function of a number of different clinical measurements.
- Predict the temperature at any location inside a building using weather data, time, door sensors, etc.

1.3 Unsupervised learning

We now consider **unsupervised learning**, where we are just given output data, without any inputs. The goal is to discover "interesting structure" in the data; this is sometimes called **knowledge discovery**. Unlike supervised learning, we are not told what the desired output is for each input. Instead, we will formalize our task as one of **density estimation**, that is, we want to build models of the form $p(\mathbf{x}_i|\boldsymbol{\theta})$. There are two differences from the supervised case. First, we have written $p(\mathbf{x}_i|\boldsymbol{\theta})$ instead of $p(y_i|\mathbf{x}_i, \boldsymbol{\theta})$; that is, supervised learning is conditional density estimation, whereas unsupervised learning is unconditional density estimation. Second, $\mathbf{x}_i$ is a vector of features, so we need to create multivariate probability models. By contrast, in supervised learning, y_i is usually just a single variable that we are trying to predict. This means that for most supervised learning problems, we can use univariate probability models (with input-dependent parameters), which significantly simplifies the problem. (We will discuss multi-output classification in Chapter 19, where we will see that it also involves multivariate probability models.)

Unsupervised learning is arguably more typical of human and animal learning. It is also more widely applicable than supervised learning, since it does not require a human expert to