

Kernel Methods for Pattern Analysis

Pattern Analysis is the process of finding general relations in a set of data, and forms the core of many disciplines, from neural networks to so-called syntactical pattern recognition, from statistical pattern recognition to machine learning and data mining. Applications of pattern analysis range from bioinformatics to document retrieval.

The kernel methodology described here provides a powerful and unified framework for all of these disciplines, motivating algorithms that can act on general types of data (e.g. strings, vectors, text, etc.) and look for general types of relations (e.g. rankings, classifications, regressions, clusters, etc.). This book fulfils two major roles. Firstly it provides practitioners with a large toolkit of algorithms, kernels and solutions ready to be implemented, many given as Matlab code suitable for many pattern analysis tasks in fields such as bioinformatics, text analysis, and image analysis. Secondly it furnishes students and researchers with an easy introduction to the rapidly expanding field of kernel-based pattern analysis, demonstrating with examples how to handcraft an algorithm or a kernel for a new specific application, while covering the required conceptual and mathematical tools necessary to do so.

The book is in three parts. The first provides the conceptual foundations of the field, both by giving an extended introductory example and by covering the main theoretical underpinnings of the approach. The second part contains a number of kernel-based algorithms, from the simplest to sophisticated systems such as kernel partial least squares, canonical correlation analysis, support vector machines, principal components analysis, etc. The final part describes a number of kernel functions, from basic examples to advanced recursive kernels, kernels derived from generative models such as HMMs and string matching kernels based on dynamic programming, as well as special kernels designed to handle text documents.

All those involved in pattern recognition, machine learning, neural networks and their applications, from computational biology to text analysis will welcome this account.

Kernel Methods for Pattern Analysis

John Shawe-Taylor
University of Southampton

Nello Cristianini
University of California at Davis



PUBLISHED BY THE PRESS SYNDICATE OF THE UNIVERSITY OF CAMBRIDGE
The Pitt Building, Trumpington Street, Cambridge, United Kingdom

CAMBRIDGE UNIVERSITY PRESS
The Edinburgh Building, Cambridge CB2 2RU, UK
40 West 20th Street, New York, NY 10011-4211, USA
477 Williamstown Road, Port Melbourne, VIC 3207, Australia
Ruiz de Alarcón 13, 28014 Madrid, Spain
Dock House, The Waterfront, Cape Town 8001, South Africa
<http://www.cambridge.org>

© Cambridge University Press 2004

This book is in copyright. Subject to statutory exception
and to the provisions of relevant collective licensing agreements,
no reproduction of any part may take place without
the written permission of Cambridge University Press.

First published 2004

Printed in the United Kingdom at the University Press, Cambridge

Typeface Monotype Times 10/13pt *System* L^AT_EX 2_ε [AUTHOR]

A catalogue record for this book is available from the British Library

Library of Congress Cataloguing in Publication data available

ISBN 0 521 81397 2 hardback

Contents

<i>List of code fragments</i>	<i>page</i> viii
<i>Preface</i>	xi
Part I Basic concepts:	1
1 Pattern analysis	3
1.1 Patterns in data	4
1.2 Pattern analysis algorithms	12
1.3 Exploiting patterns	17
1.4 Summary	22
1.5 Further reading and advanced topics	23
2 Kernel methods: an overview	25
2.1 The overall picture	26
2.2 Linear regression in a feature space	27
2.3 Other examples	36
2.4 The modularity of kernel methods	42
2.5 Roadmap of the book	43
2.6 Summary	44
2.7 Further reading and advanced topics	45
3 Properties of kernels	47
3.1 Inner products and positive semi-definite matrices	48
3.2 Characterisation of kernels	60
3.3 The kernel matrix	68
3.4 Kernel construction	74
3.5 Summary	82
3.6 Further reading and advanced topics	82
4 Detecting stable patterns	85
4.1 Concentration inequalities	86
4.2 Capacity and regularisation: rademacher theory	93

4.3	Pattern stability for kernel-based classes	97
4.4	A pragmatic approach	104
4.5	Summary	105
4.6	Further reading and advanced topics	106
	Part II Pattern analysis algorithms:	109
5	Elementary algorithms in feature space	111
5.1	Means and distances	112
5.2	Computing projections: Gram–Schmidt, QR and Cholesky	122
5.3	Measuring the spread of the data	128
5.4	Fisher discriminant analysis I	132
5.5	Summary	137
5.6	Further reading and advanced topics	138
6	Pattern analysis using eigen-decompositions	140
6.1	Singular value decomposition	141
6.2	Principal components analysis	143
6.3	Directions of maximum covariance	155
6.4	The generalised eigenvector problem	161
6.5	Canonical correlation analysis	164
6.6	Fisher discriminant analysis II	176
6.7	Methods for linear regression	176
6.8	Summary	192
6.9	Further reading and advanced topics	193
7	Pattern analysis using convex optimisation	195
7.1	The smallest enclosing hypersphere	196
7.2	Support vector machines for classification	211
7.3	Support vector machines for regression	230
7.4	On-line classification and regression	241
7.5	Summary	249
7.6	Further reading and advanced topics	250
8	Ranking, clustering and data visualisation	252
8.1	Discovering rank relations	253
8.2	Discovering cluster structure in a feature space	264
8.3	Data visualisation	280
8.4	Summary	286
8.5	Further reading and advanced topics	286
	Part III Constructing kernels:	289
9	Basic kernels and kernel types	291
9.1	Kernels in closed form	292

9.2	ANOVA kernels	297
9.3	Kernels from graphs	304
9.4	Diffusion kernels on graph nodes	310
9.5	Kernels on sets	314
9.6	Kernels on real numbers	317
9.7	Randomised kernels	320
9.8	Other kernel types	322
9.9	Summary	324
9.10	Further reading and advanced topics	325
10	Kernels for text	327
10.1	From bag of words to semantic space	328
10.2	Vector space kernels	331
10.3	Summary	341
10.4	Further reading and advanced topics	342
11	Kernels for structured data: strings, trees, etc.	344
11.1	Comparing strings and sequences	345
11.2	Spectrum kernels	347
11.3	All-subsequences kernels	351
11.4	Fixed length subsequences kernels	357
11.5	Gap-weighted subsequences kernels	360
11.6	Beyond dynamic programming: trie-based kernels	372
11.7	Kernels for structured data	382
11.8	Summary	395
11.9	Further reading and advanced topics	395
12	Kernels from generative models	397
12.1	P -kernels	398
12.2	Fisher kernels	421
12.3	Summary	435
12.4	Further reading and advanced topics	436
	Appendix A Proofs omitted from the main text	437
	Appendix B Notational conventions	444
	Appendix C List of pattern analysis methods	446
	Appendix D List of kernels	448
	<i>References</i>	450
	<i>Index</i>	460

Code fragments

5.1	Matlab code normalising a kernel matrix.	<i>page</i> 113
5.2	Matlab code for centering a kernel matrix.	116
5.3	Matlab code for simple novelty detection algorithm.	118
5.4	Matlab code for performing incomplete Cholesky decomposition or dual partial Gram–Schmidt orthogonalisation.	129
5.5	Matlab code for standardising data.	131
5.6	Kernel Fisher discriminant algorithm	137
6.1	Matlab code for kernel PCA algorithm.	152
6.2	Pseudocode for the whitening algorithm.	156
6.3	Pseudocode for the kernel CCA algorithm.	175
6.4	Pseudocode for dual principal components regression.	179
6.5	Pseudocode for PLS feature extraction.	182
6.6	Pseudocode for the primal PLS algorithm.	186
6.7	Matlab code for the primal PLS algorithm.	187
6.8	Pseudocode for the kernel PLS algorithm.	191
6.9	Matlab code for the dual PLS algorithm.	192
7.1	Pseudocode for computing the minimal hypersphere.	199
7.2	Pseudocode for soft hypersphere minimisation.	205
7.3	Pseudocode for the soft hypersphere.	208
7.4	Pseudocode for the hard margin SVM.	215
7.5	Pseudocode for the alternative version of the hard SVM.	218
7.6	Pseudocode for 1-norm soft margin SVM.	223
7.7	Pseudocode for the soft margin SVM.	225
7.8	Pseudocode for the 2-norm SVM.	229
7.9	Pseudocode for 2-norm support vector regression.	237
7.10	Pseudocode for 1-norm support vector regression.	238
7.11	Pseudocode for new SVR.	240
7.12	Pseudocode for the kernel perceptron algorithm.	242
7.13	Pseudocode for the kernel adatron algorithm.	247
7.14	Pseudocode for the on-line support vector regression.	249
8.1	Pseudocode for the soft ranking algorithm.	259
8.2	Pseudocode for on-line ranking.	262

8.3	Matlab code to perform k -means clustering.	275
8.4	Matlab code to implementing low-dimensional visualisation.	285
9.1	Pseudocode for ANOVA kernel.	301
9.2	Pseudocode for simple graph kernels.	308
11.1	Pseudocode for the all-non-contiguous subsequences kernel.	356
11.2	Pseudocode for the fixed length subsequences kernel.	359
11.3	Pseudocode for the gap-weighted subsequences kernel.	369
11.4	Pseudocode for trie-based implementation of spectrum kernel.	374
11.5	Pseudocode for the trie-based implementation of the mismatch kernel.	378
11.6	Pseudocode for trie-based restricted gap-weighted subsequences kernel.	381
11.7	Pseudocode for the co-rooted subtree kernel.	387
11.8	Pseudocode for the all-subtree kernel.	389
12.1	Pseudocode for the fixed length HMM kernel.	409
12.2	Pseudocode for the pair HMM kernel.	415
12.3	Pseudocode for the hidden tree model kernel.	420
12.4	Pseudocode to compute the Fisher scores for the fixed length Markov model Fisher kernel.	435

Preface

The study of patterns in data is as old as science. Consider, for example, the astronomical breakthroughs of Johannes Kepler formulated in his three famous laws of planetary motion. They can be viewed as relations that he detected in a large set of observational data compiled by Tycho Brahe.

Equally the wish to automate the search for patterns is at least as old as computing. The problem has been attacked using methods of statistics, machine learning, data mining and many other branches of science and engineering.

Pattern analysis deals with the problem of (automatically) detecting and characterising relations in data. Most statistical and machine learning methods of pattern analysis assume that the data is in vectorial form and that the relations can be expressed as classification rules, regression functions or cluster structures; these approaches often go under the general heading of ‘statistical pattern recognition’. ‘Syntactical’ or ‘structural pattern recognition’ represents an alternative approach that aims to detect rules among, for example, strings, often in the form of grammars or equivalent abstractions.

The evolution of automated algorithms for pattern analysis has undergone three revolutions. In the 1960s efficient algorithms for detecting linear relations within sets of vectors were introduced. Their computational and statistical behaviour was also analysed. The Perceptron algorithm introduced in 1957 is one example. The question of how to detect nonlinear relations was posed as a major research goal at that time. Despite this developing algorithms with the same level of efficiency and statistical guarantees has proven an elusive target.

In the mid 1980s the field of pattern analysis underwent a ‘nonlinear revolution’ with the almost simultaneous introduction of backpropagation multi-layer neural networks and efficient decision tree learning algorithms. These

approaches for the first time made it possible to detect nonlinear patterns, albeit with heuristic algorithms and incomplete statistical analysis. The impact of the nonlinear revolution cannot be overemphasised: entire fields such as data mining and bioinformatics were enabled by it. These nonlinear algorithms, however, were based on gradient descent or greedy heuristics and so suffered from local minima. Since their statistical behaviour was not well understood, they also frequently suffered from overfitting.

A third stage in the evolution of pattern analysis algorithms took place in the mid-1990s with the emergence of a new approach to pattern analysis known as kernel-based learning methods that finally enabled researchers to analyse nonlinear relations with the efficiency that had previously been reserved for linear algorithms. Furthermore advances in their statistical analysis made it possible to do so in high-dimensional feature spaces while avoiding the dangers of overfitting. From all points of view, computational, statistical and conceptual, the nonlinear pattern analysis algorithms developed in this third generation are as efficient and as well founded as linear ones. The problems of local minima and overfitting that were typical of neural networks and decision trees have been overcome. At the same time, these methods have been proven very effective on non vectorial data, in this way creating a connection with other branches of pattern analysis.

Kernel-based learning first appeared in the form of support vector machines, a classification algorithm that overcame the computational and statistical difficulties alluded to above. Soon, however, kernel-based algorithms able to solve tasks other than classification were developed, making it increasingly clear that the approach represented a revolution in pattern analysis. Here was a whole new set of tools and techniques motivated by rigorous theoretical analyses and built with guarantees of computational efficiency.

Furthermore, the approach is able to bridge the gaps that existed between the different subdisciplines of pattern recognition. It provides a unified framework to reason about and operate on data of all types be they vectorial, strings, or more complex objects, while enabling the analysis of a wide variety of patterns, including correlations, rankings, clusterings, etc.

This book presents an overview of this new approach. We have attempted to condense into its chapters an intense decade of research generated by a new and thriving research community. Together its researchers have created a class of methods for pattern analysis, which has become an important part of the practitioner's toolkit.

The algorithms presented in this book can identify a wide variety of relations, ranging from the traditional tasks of classification and regression, through more specialised problems such as ranking and clustering, to ad-

vanced techniques including principal components analysis and canonical correlation analysis. Furthermore, each of the pattern analysis tasks can be applied in conjunction with each of the bank of kernels developed in the final part of the book. This means that the analysis can be applied to a wide variety of data, ranging from standard vectorial types through more complex objects such as images and text documents, to advanced datatypes associated with biosequences, graphs and grammars.

Kernel-based analysis is a powerful new tool for mathematicians, scientists and engineers. It provides a surprisingly rich way to interpolate between pattern analysis, signal processing, syntactical pattern recognition and pattern recognition methods from splines to neural networks. In short, it provides a new viewpoint whose full potential we are still far from understanding.

The authors have played their part in the development of kernel-based learning algorithms, providing a number of contributions to the theory, implementation, application and popularisation of the methodology. Their book, *An Introduction to Support Vector Machines*, has been used as a textbook in a number of universities, as well as a research reference book. The authors also assisted in the organisation of a European Commission funded Working Group in ‘Neural and Computational Learning (NeuroCOLT)’ that played an important role in defining the new research agenda as well as in the project ‘Kernel Methods for Images and Text (KerMIT)’ that has seen its application in the domain of document analysis.

The authors would like to thank the many people who have contributed to this book through discussion, suggestions and in many cases highly detailed and enlightening feedback. Particularly thanks are owing to Gert Lanckriet, Michinari Momma, Kristin Bennett, Tijl DeBie, Roman Rosipal, Christina Leslie, Craig Saunders, Bernhard Schölkopf, Nicolò Cesa-Bianchi, Peter Bartlett, Colin Campbell, William Noble, Prabir Burman, Jean-Philippe Vert, Michael Jordan, Manju Pai, Andrea Frome, Chris Watkins, Juho Rousu, Thore Graepel, Ralf Herbrich, and David Hardoon. They would also like to thank the European Commission and the UK funding council EPSRC for supporting their research into the development of kernel-based learning methods.

Nello Cristianini is Assistant Professor of Statistics at University of California in Davis. Nello would like to thank UC Berkeley Computer Science Department and Mike Jordan for hosting him during 2001–2002, when Nello was a Visiting Lecturer there. He would also like to thank MIT CBLC and Tommy Poggio for hosting him during the summer of 2002, as well as the Department of Statistics at UC Davis, which has provided him with an ideal environment for this work. Much of the structure of the book is based on

courses taught by Nello at UC Berkeley, at UC Davis and tutorials given in a number of conferences.

John Shawe-Taylor is professor of computing science at the University of Southampton. John would like to thank colleagues in the Computer Science Department of Royal Holloway, University of London, where he was employed during most of the writing of the book.