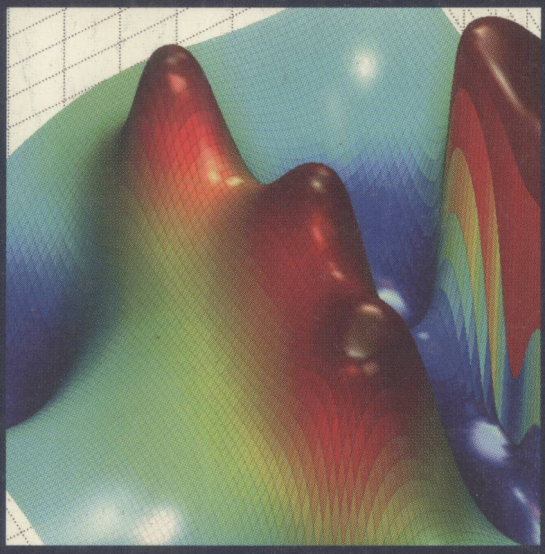
Olivier Bousquet
Ulrike von Luxburg
Gunnar Rätsch (Eds.)

Tutorial

LNAI 3176

Advanced Lectures on Machine Learning

ML Summer Schools 2003
Canberra, Australia, February 2003
Tübingen, Germany, August 2003, Revised Lectures



Springer

Olivier Bousquet Ulrike von Luxburg
Gunnar Rätsch (Eds.)

Advanced Lectures on Machine Learning

ML Summer Schools 2003
Canberra, Australia, February 2-14, 2003
Tübingen, Germany, August 4-16, 2003
Revised Lectures

Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editors

Olivier Bousquet
Ulrike von Luxburg
Max Planck Institute for Biological Cybernetics
Spemannstr. 38, 72076 Tübingen, Germany
E-mail: {bousquet, ule}@tuebingen.mpg.de

Gunnar Rätsch
Fraunhofer FIRST
Kekuléstr. 7, 10245 Berlin, Germany
and Max Planck Institute for Biological Cybernetics
Spemannstr. 38, 72076 Tübingen, Germany
E-mail: Gunnar.Raetsch@tuebingen.mpg.de

Library of Congress Control Number: 2004111357

CR Subject Classification (1998): I.2.6, I.2, F.1, F.2, I.5

ISSN 0302-9743
ISBN 3-540-23122-6 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11322894 06/3142 5 4 3 2 1 0

Lecture Notes in Artificial Intelligence 3176

Edited by J. G. Carbonell and J. Siekmann

Subseries of Lecture Notes in Computer Science

Preface

Machine Learning has become a key enabling technology for many engineering applications, investigating scientific questions and theoretical problems alike. To stimulate discussions and to disseminate new results, a series of summer schools was started in February 2002. One year later two more such summer schools were held, one at the Australian National University in Canberra, Australia, and the other one at the Max Planck Institute for Biological Cybernetics, in Tübingen, Germany.

The current book contains a collection of the main talks held during those two summer schools, presented as tutorial chapters on topics such as pattern recognition, Bayesian inference, unsupervised learning and statistical learning theory. The papers provide an in-depth overview of these exciting new areas, contain a large set of references, and thereby provide the interested readers with further information to start or to pursue their own research in these directions.

Complementary to the book, photos and slides of the presentations can be obtained at

`http://mlg.anu.edu.au/summer2003`

and

`http://www.irccyn.ec-nantes.fr/mlschool/mlss03/home03.php`.

The general entry point for past and future Machine Learning Summer Schools is

`http://www.mlss.cc`

It is our hope that graduate students, lecturers, and researchers alike will find this book useful in learning and teaching machine learning, thereby continuing the mission of the Machine Learning Summer Schools.

Tübingen, June 2004

Olivier Bousquet
Ulrike von Luxburg
Gunnar Rätsch

Empirical Inference for Machine Learning and Perception
Max Planck Institute for Biological Cybernetics

Acknowledgments

We gratefully thank all the individuals and organizations responsible for the success of the summer schools.

Local Arrangements

Canberra

Special thanks go to Michelle Moravec and Heather Slater for all their support during the preparations, to Joe Elso, Kim Holburn, and Fergus McKenzie-Kay for IT support, to Cheng Soon-Ong, Kristy Sim, Edward Harrington, Evan Greensmith, and the students at the Computer Sciences Laboratory for their help throughout the course of the Summer School.

Tübingen

Special thanks go to Sabrina Nielebock for all her work during the preparation and on the site, to Dorothea Epting and the staff of the Max Planck Guest House, to Sebastian Stark for IT support, and to all the students and administration of the Max Planck Institute for Biological Cybernetics for their help throughout the Summer School.

Sponsoring Institutions

Canberra

- Research School of Information Sciences and Engineering, Australia
- National Institute of Engineering and Information Science, Australia

Tübingen

- Centre National de la Recherche Scientifique, France
- French-German University
- Max Planck Institute for Biological Cybernetics, Germany

Speakers

Canberra

Shun-Ichi Amari
Eleazar Eskin
Zoubin Ghahramani
Peter Hall
Markus Hegland
Gabor Lugosi
Jyrki Kivinen
John Lloyd
Shahar Mendelson
Mike Osborne
Petra Phillips
Gunnar Rätsch
Alex Smola
S.V.N. Vishwanathan
Robert C. Williamson

Tübingen

Christophe Andrieu
Pierre Baldi
Léon Bottou
Stéphane Boucheron
Olivier Bousquet
Chris Burges
Jean-François Cardoso
Manuel Davy
André Elisseeff
Arthur Gretton
Peter Grünwald
Thorsten Joachims
Massimiliano Pontil
Carl Rasmussen
Mike Tipping
Bernhard Schölkopf
Steve Smale
Alex Smola
Vladimir Vapnik
Jason Weston
Elad Yom-Tov
Ding-Xuan Zhou

Organization Committees

Canberra: Gunnar Rätsch and Alex Smola
Tübingen: Olivier Bousquet, Manuel Davy, Frédéric Desobry,
Ulrike von Luxburg and Bernhard Schölkopf

Lecture Notes in Artificial Intelligence (LNAI)

Vol. 3206: P. Sojka, I. Kopecek, K. Pala (Eds.), Text, Speech and Dialogue. XIII, 667 pages. 2004.

Vol. 3194: R. Camacho, R. King, A. Srinivasan (Eds.), Inductive Logic Programming. XI, 361 pages. 2004.

Vol. 3192: C. Bussler, D. Fensel (Eds.), Artificial Intelligence: Methodology, Systems, and Applications. XIII, 522 pages. 2004.

Vol. 3176: O. Bousquet, U. von Luxburg, G. Rätsch (Eds.), Advanced Lectures on Machine Learning. IX, 241 pages. 2004.

Vol. 3159: U. Visser, Intelligent Information Integration for the Semantic Web. XIV, 150 pages. 2004.

Vol. 3157: C. Zhang, H. W. Guesgen, W.K. Yeap (Eds.), PRICAI 2004: Trends in Artificial Intelligence. XX, 1023 pages. 2004.

Vol. 3155: P. Funk, P.A. González Calero (Eds.), Advances in Case-Based Reasoning. XIII, 822 pages. 2004.

Vol. 3139: F. Iida, R. Pfeifer, L. Steels, Y. Kuniyoshi (Eds.), Embodied Artificial Intelligence. IX, 331 pages. 2004.

Vol. 3131: V. Torra, Y. Narukawa (Eds.), Modeling Decisions for Artificial Intelligence. XI, 327 pages. 2004.

Vol. 3127: K.E. Wolff, H.D. Pfeiffer, H.S. Delugach (Eds.), Conceptual Structures at Work. XI, 403 pages. 2004.

Vol. 3123: A. Belz, R. Evans, P. Piwek (Eds.), Natural Language Generation. X, 219 pages. 2004.

Vol. 3120: J. Shawe-Taylor, Y. Singer (Eds.), Learning Theory. X, 648 pages. 2004.

Vol. 3097: D. Basin, M. Rusinowitch (Eds.), Automated Reasoning. XII, 493 pages. 2004.

Vol. 3071: A. Omicini, P. Petta, J. Pitt (Eds.), Engineering Societies in the Agents World. XIII, 409 pages. 2004.

Vol. 3070: L. Rutkowski, J. Siekmann, R. Tadeusiewicz, L.A. Zadeh (Eds.), Artificial Intelligence and Soft Computing - ICAISC 2004. XXV, 1208 pages. 2004.

Vol. 3068: E. André, L. Dybkjær, W. Minker, P. Heisterkamp (Eds.), Affective Dialogue Systems. XII, 324 pages. 2004.

Vol. 3067: M. Dastani, J. Dix, A. El Fallah-Seghrouchni (Eds.), Programming Multi-Agent Systems. X, 221 pages. 2004.

Vol. 3066: S. Tsumoto, R. Słowiński, J. Komorowski, J.W. Grzymała-Busse (Eds.), Rough Sets and Current Trends in Computing. XX, 853 pages. 2004.

Vol. 3065: A. Lomuscio, D. Nute (Eds.), Deontic Logic in Computer Science. X, 275 pages. 2004.

Vol. 3060: A.Y. Tawfik, S.D. Goodwin (Eds.), Advances in Artificial Intelligence. XIII, 582 pages. 2004.

Vol. 3056: H. Dai, R. Srikant, C. Zhang (Eds.), Advances in Knowledge Discovery and Data Mining. XIX, 713 pages. 2004.

Vol. 3055: H. Christiansen, M.-S. Hacid, T. Andreasen, H.L. Larsen (Eds.), Flexible Query Answering Systems. X, 500 pages. 2004.

Vol. 3040: R. Conejo, M. Urretavizcaya, J.-L. Pérez-de-la-Cruz (Eds.), Current Topics in Artificial Intelligence. XIV, 689 pages. 2004.

Vol. 3035: M.A. Wimmer (Ed.), Knowledge Management in Electronic Government. XII, 326 pages. 2004.

Vol. 3034: J. Favela, E. Menasalvas, E. Chávez (Eds.), Advances in Web Intelligence. XIII, 227 pages. 2004.

Vol. 3030: P. Giorgini, B. Henderson-Sellers, M. Winikoff (Eds.), Agent-Oriented Information Systems. XIV, 207 pages. 2004.

Vol. 3029: B. Orchard, C. Yang, M. Ali (Eds.), Innovations in Applied Artificial Intelligence. XXI, 1272 pages. 2004.

Vol. 3025: G.A. Vouros, T. Panayiotopoulos (Eds.), Methods and Applications of Artificial Intelligence. XV, 546 pages. 2004.

Vol. 3020: D. Polani, B. Browning, A. Bonarini, K. Yoshida (Eds.), RoboCup 2003: Robot Soccer World Cup VII. XVI, 767 pages. 2004.

Vol. 3012: K. Kurumatani, S.-H. Chen, A. Ohuchi (Eds.), Multi-Agnets for Mass User Support. X, 217 pages. 2004.

Vol. 3010: K.R. Apt, F. Fages, F. Rossi, P. Szeredi, J. Váncza (Eds.), Recent Advances in Constraints. VIII, 285 pages. 2004.

Vol. 2990: J. Leite, A. Omicini, L. Sterling, P. Torroni (Eds.), Declarative Agent Languages and Technologies. XII, 281 pages. 2004.

Vol. 2980: A. Blackwell, K. Marriott, A. Shimojima (Eds.), Diagrammatic Representation and Inference. XV, 448 pages. 2004.

Vol. 2977: G. Di Marzo Serugendo, A. Karageorgos, O.F. Rana, F. Zambonelli (Eds.), Engineering Self-Organising Systems. X, 299 pages. 2004.

Vol. 2972: R. Monroy, G. Arroyo-Figueroa, L.E. Sucar, H. Sossa (Eds.), MICAI 2004: Advances in Artificial Intelligence. XVII, 923 pages. 2004.

Vol. 2969: M. Nickles, M. Rovatsos, G. Weiss (Eds.), Agents and Computational Autonomy. X, 275 pages. 2004.

Vol. 2961: P. Eklund (Ed.), Concept Lattices. IX, 411 pages. 2004.

Vol. 2953: K. Konrad, Model Generation for Natural Language Interpretation and Analysis. XIII, 166 pages. 2004.

Vol. 2934: G. Lindemann, D. Moldt, M. Paolucci (Eds.), Regulated Agent-Based Social Systems. X, 301 pages. 2004.

Vol. 2930: F. Winkler (Ed.), Automated Deduction in Geometry. VII, 231 pages. 2004.

Vol. 2926: L. van Elst, V. Dignum, A. Abecker (Eds.), Agent-Mediated Knowledge Management. XI, 428 pages. 2004.

Vol. 2923: V. Lifschitz, I. Niemelä (Eds.), Logic Programming and Nonmonotonic Reasoning. IX, 365 pages. 2004.

Vol. 2915: A. Camurri, G. Volpe (Eds.), Gesture-Based Communication in Human-Computer Interaction. XIII, 558 pages. 2004.

Vol. 2913: T.M. Pinkston, V.K. Prasanna (Eds.), High Performance Computing - HiPC 2003. XX, 512 pages. 2003.

Vol. 2903: T.D. Gedeon, L.C.C. Fung (Eds.), AI 2003: Advances in Artificial Intelligence. XVI, 1075 pages. 2003.

Vol. 2902: F.M. Pires, S.P. Abreu (Eds.), Progress in Artificial Intelligence. XV, 504 pages. 2003.

Vol. 2892: F. Dau, The Logic System of Concept Graphs with Negation. XI, 213 pages. 2003.

Vol. 2891: J. Lee, M. Barley (Eds.), Intelligent Agents and Multi-Agent Systems. X, 215 pages. 2003.

Vol. 2882: D. Veit, Matchmaking in Electronic Markets. XV, 180 pages. 2003.

Vol. 2871: N. Zhong, Z.W. Raś, S. Tsumoto, E. Suzuki (Eds.), Foundations of Intelligent Systems. XV, 697 pages. 2003.

Vol. 2854: J. Hoffmann, Utilizing Problem Structure in Planing. XIII, 251 pages. 2003.

Vol. 2843: G. Grieser, Y. Tanaka, A. Yamamoto (Eds.), Discovery Science. XII, 504 pages. 2003.

Vol. 2842: R. Gavaldá, K.P. Jantke, E. Takimoto (Eds.), Algorithmic Learning Theory. XI, 313 pages. 2003.

Vol. 2838: N. Lavrač, D. Gamberger, L. Todorovski, H. Blockeel (Eds.), Knowledge Discovery in Databases: PKDD 2003. XVI, 508 pages. 2003.

Vol. 2837: N. Lavrač, D. Gamberger, L. Todorovski, H. Blockeel (Eds.), Machine Learning: ECML 2003. XVI, 504 pages. 2003.

Vol. 2835: T. Horváth, A. Yamamoto (Eds.), Inductive Logic Programming. X, 401 pages. 2003.

Vol. 2821: A. Günter, R. Kruse, B. Neumann (Eds.), KI 2003: Advances in Artificial Intelligence. XII, 662 pages. 2003.

Vol. 2807: V. Matoušek, P. Mautner (Eds.), Text, Speech and Dialogue. XIII, 426 pages. 2003.

Vol. 2801: W. Banzhaf, J. Ziegler, T. Christaller, P. Dittrich, J.T. Kim (Eds.), Advances in Artificial Life. XVI, 905 pages. 2003.

Vol. 2797: O.R. Zaïane, S.J. Simoff, C. Djeraba (Eds.), Mining Multimedia and Complex Data. XII, 281 pages. 2003.

Vol. 2792: T. Rist, R.S. Aylett, D. Ballin, J. Rickel (Eds.), Intelligent Virtual Agents. XV, 364 pages. 2003.

Vol. 2782: M. Klusch, A. Omicini, S. Ossowski, H. Laamanen (Eds.), Cooperative Information Agents VII. XI, 345 pages. 2003.

Vol. 2780: M. Dojat, E. Keravnou, P. Barahona (Eds.), Artificial Intelligence in Medicine. XIII, 388 pages. 2003.

Vol. 2777: B. Schölkopf, M.K. Warmuth (Eds.), Learning Theory and Kernel Machines. XIV, 746 pages. 2003.

Vol. 2752: G.A. Kaminka, P.U. Lima, R. Rojas (Eds.), RoboCup 2002: Robot Soccer World Cup VI. XVI, 498 pages. 2003.

Vol. 2741: F. Baader (Ed.), Automated Deduction – CADE-19. XII, 503 pages. 2003.

Vol. 2705: S. Renals, G. Grefenstette (Eds.), Text- and Speech-Triggered Information Access. VII, 197 pages. 2003.

Vol. 2703: O.R. Zaïane, J. Srivastava, M. Spiliopoulou, B. Masand (Eds.), WEBKDD 2002 - MiningWeb Data for Discovering Usage Patterns and Profiles. IX, 181 pages. 2003.

Vol. 2700: M.T. Pazienza (Ed.), Extraction in the Web Era. XIII, 163 pages. 2003.

Vol. 2699: M.G. Hinchey, J.L. Rash, W.F. Truszkowski, C.A. Rouff, D.F. Gordon-Spears (Eds.), Formal Approaches to Agent-Based Systems. IX, 297 pages. 2002.

Vol. 2691: V. Mařík, J.P. Müller, M. Pechoucek (Eds.), Multi-Agent Systems and Applications III. XIV, 660 pages. 2003.

Vol. 2684: M.V. Butz, O. Sigaud, P. Gérard (Eds.), Anticipatory Behavior in Adaptive Learning Systems. X, 303 pages. 2003.

Vol. 2682: R. Meo, P.L. Lanzi, M. Klemettinen (Eds.), Database Support for Data Mining Applications. XII, 325 pages. 2004.

Vol. 2671: Y. Xiang, B. Chaib-draa (Eds.), Advances in Artificial Intelligence. XIV, 642 pages. 2003.

Vol. 2663: E. Menasalvas, J. Segovia, P.S. Szczepaniak (Eds.), Advances in Web Intelligence. XII, 350 pages. 2003.

Vol. 2661: P.L. Lanzi, W. Stolzmann, S.W. Wilson (Eds.), Learning Classifier Systems. VII, 231 pages. 2003.

Vol. 2654: U. Schmid, Inductive Synthesis of Functional Programs. XXII, 398 pages. 2003.

Vol. 2650: M.-P. Huget (Ed.), Communications in Multiagent Systems. VIII, 323 pages. 2003.

Vol. 2645: M.A. Wimmer (Ed.), Knowledge Management in Electronic Government. XI, 320 pages. 2003.

Vol. 2639: G. Wang, Q. Liu, Y. Yao, A. Skowron (Eds.), Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing. XVII, 741 pages. 2003.

Vol. 2637: K.-Y. Whang, J. Jeon, K. Shim, J. Srivastava, Advances in Knowledge Discovery and Data Mining. XVIII, 610 pages. 2003.

Vol. 2636: E. Alonso, D. Kudenko, D. Kazakov (Eds.), Adaptive Agents and Multi-Agent Systems. XIV, 323 pages. 2003.

Vol. 2627: B. O'Sullivan (Ed.), Recent Advances in Constraints. X, 201 pages. 2003.

Vol. 2600: S. Mendelson, A.J. Smola (Eds.), Advanced Lectures on Machine Learning. IX, 259 pages. 2003.

Vol. 2592: R. Kowalczyk, J.P. Müller, H. Tianfield, R. Unland (Eds.), Agent Technologies, Infrastructures, Tools, and Applications for E-Services. XVII, 371 pages. 2003.

Table of Contents

An Introduction to Pattern Classification

Elad Yom-Tov

IBM Haifa Research Labs, University Campus, Haifa 31905, Israel
yomtov@il.ibm.com

1 Introduction

Pattern classification is the field devoted to the study of methods designed to categorize data into distinct classes. This categorization can be either distinct labeling of the data (supervised learning), division of the data into classes (unsupervised learning), selection of the most significant features of the data (feature selection), or a combination of more than one of these tasks.

Pattern classification is one of a class of problems that humans (under most circumstances) are able to accomplish extremely well, but are difficult for computers to perform. This subject has been under extensive study for many years. However during the past decade, with the introduction of several new classes of pattern classification algorithms this field seems to achieve performance much better than previously attained.

The goal of the following article is to give the reader a broad overview of the field. As such, it attempts to introduce the reader to important aspects of pattern classification, without delving deeply into any of the subject matters. The exceptions to this rule are those points deemed especially important or those that are of special interest. Finally, we note that the focus of this article are statistical methods for pattern recognition. Thus, methods such as fuzzy logic and rule-based methods are outside the scope of this article.

2 What Is Pattern Classification?

Pattern classification, also referred to as pattern recognition, attempts to build algorithms capable of automatically constructing methods for distinguishing between different exemplars, based on their differentiating patterns.

Watanabe [53] described a pattern as "the opposite of chaos; it is an entity, vaguely defined, that could be given a name." Examples of patterns are human faces, handwritten letters, and the DNA sequences that may cause a certain disease. More formally, the goal of a (supervised) pattern classification task is to find a functional mapping between the input data X, used to describe an input pattern, to a class label Y so that $Y = f(X)$. Construction of the mapping is based on **training data** supplied to the pattern classification algorithm. The mapping f should give the smallest possible error in the mapping, i.e. the minimum number of examples where Y will be the wrong label, especially on **test data** not seen by the algorithm during the learning phase.

O. Bousquet et al. (Eds.): Machine Learning 2003, LNAI 3176, pp. 1–20, 2004.

An important division of pattern classification tasks are **supervised** as opposed to **unsupervised** classification. In supervised tasks the training data consists of training patterns, as well as their required labeling. An example are DNA sequences labeled to show which examples are known to harbor a genetic trait and which ones do not. In unsupervised classification tasks the labels are not provided, and the task of the algorithm is to find a "good" partition of the data into clusters. Examples for this kind of task are grouping of Web pages into sets so that each set is concerned with a single subject matter.

A pattern is described by its **features**. These are the characteristics of the examples for a given problem. For example, in a face recognition task some features could be the color of the eyes or the distance between the eyes. Thus, the input to a pattern recognition task can be viewed as a two-dimensional matrix, whose axes are the examples and the features.

Pattern classification tasks are customarily divided into several distinct blocks. These are:

1. Data collection and representation.
2. Feature selection and/or feature reduction.
3. Clustering.
4. Classification.

Data collection and representation are mostly problem-specific. Therefore it is difficult to give general statements about this step of the process. In broad terms, one should try to find invariant features, that describe the differences in classes as best as possible.

Feature selection and feature reduction attempt to reduce the dimensionality (i.e. the number of features) for the remaining steps of the task. Clustering methods are used in order to reduce the number of training examples to the task. Finally, the classification phase of the process finds the actual mapping between patterns and labels (or targets). In many applications not all steps are needed. Indeed, as computational power grows, the need to reduce the number of patterns used as input to the classification task decreases, and may therefore make the clustering stage superfluous for many applications.

In the following pages we describe feature selection and reduction, clustering, and classification.

3 Feature Selection and Feature Reduction: Removing Excess Data

When data is collected for later classification, it may seem reasonable to assume that if more features describing the data are collected it will be easier to classify these data correctly. In fact, as Trunk [50] demonstrated, more data may be detrimental to classification, especially if the additional data is highly correlated with previous data. Furthermore, noisy and irrelevant features are detrimental to classification as they are known to cause the classifier to have poor generalization,

increase the computational complexity, and require many training samples to reach a given accuracy [4].

Conversely, selecting too few features will lead to the ugly duckling theorem [53], that is, it will be impossible to distinguish between the classes because there is too little data to differentiate the classes. For example, suppose we wish to classify a vertebrated animal into one of the vertebra classes (Mammals, Birds, Fish, Reptiles, or Amphibians). A feature that will tell us if the animal has skin is superfluous, since all vertebrates have skins. However, a feature that measures if the animal has warm blood is highly significant for the classification. A feature selection algorithm should be able to identify and remove the former feature, while preserving the latter.

Hence the goal of this stage in the processing is to choose a subset of features or some combination of the input features that will best represent the data. We refer to the process of choosing a subset of the features as **feature selection**, and to finding a good combination of the features as **feature reduction**.

Feature selection is a difficult combinatorial optimization problem. Finding the best subset of features by testing all possible combinations is practically impossible even when the number of input features is modest. For example, attempting to test all possible combinations of 100 input features will require testing 10^{30} combinations. It is not uncommon for text classification problems to have 10^4 to 10^7 features [27]. Consequently numerous methods have been proposed for finding a (suboptimal) solution by testing a fraction of the possible combinations.

Feature selection methods can be divided into three main types [4]:

1. Wrapper methods: The feature selection is performed around (and with) a given classification algorithm. The classification algorithm is used for ranking possible feature combinations.
2. Embedded methods: The feature selection is embedded within the classification algorithm.
3. Filter methods: Features are selected for classification independently of the classification algorithm.

Most feature selection methods are of the wrapper type. The simplest algorithms in this category are the exhaustive search, which is practical only when the number of features is small, **sequential forward feature selection (SFFS)** and **sequential backward feature selection (SBFS)**. In sequential forward feature selection the feature with which the lowest classification error is reached is selected. Then, the feature that, when added, causes the largest reduction in error is added to the set of selected features. This process is continued iteratively until the maximum number of features needed are found or until the classification error starts to increase. Although sequential feature selection does not assume dependence between features, it usually attains surprisingly reasonable results. There are several minor modifications to SFFS and SBFS, such as Sequential Floating Search [41] or the "Plus n, take away m" features.

One of the major drawbacks of methods that select and add a single feature at each step is that they might not find combinations of features that perform well

together, but are poor predictors individually. More sophisticated methods for feature selection use simulated annealing or genetic algorithms [56] for solving the optimization problem of feature selection. The latter approach has shown promise in solving problems where the number of input features is extremely large.

An interesting approach to feature selection is based in information theoretic considerations [25]. This algorithm estimates the cross-entropy between every pair of features, and discards those features that have a large cross-entropy with other features, thus removing features that add little additional classification information. This is because the cross-entropy estimates the amount of knowledge that one feature provides on other features. The algorithm is appealing in that it is independent of the classification algorithm, i.e. it is a filter algorithm. However, the need to estimate the cross entropy between features limits its use to applications where the datasets are large or to cases where features are discrete.

As mentioned above, a second approach to reducing the dimension of the features is to find a lower-dimensional combination (linear or non-linear) of the features which represent the data as well as possible in the required dimension.

The most commonly used technique for feature reduction is **principal component analysis (PCA)**, also known as the Karhunen-Loeve Transform (KLT). PCA reshapes the data along the directions of maximal variance. PCA works by computing the eigenvectors corresponding to the largest eigenvalues of the covariance matrix of the data, and returning the projection of the data on these eigenvectors. An example of feature reduction using PCA is given in Figure 1.

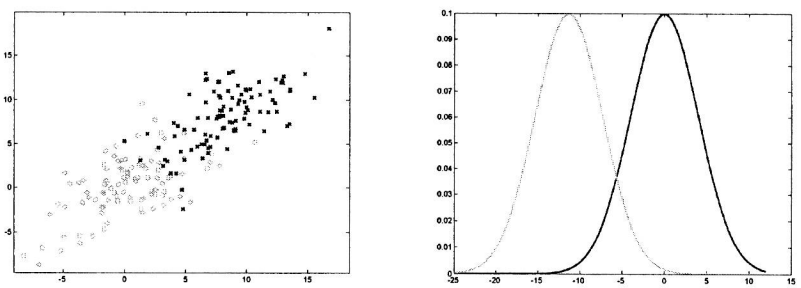


Fig. 1. Feature reduction using principle component analysis. The figure on the left shows the original data. Note that most of the variance in the data is along a single direction. The figure on the right shows probability density function of the same data after feature reduction to a dimension of 1 using PCA

Principle component analysis does not take into account the labels of the data. As such, it is an unsupervised method. A somewhat similar, albeit supervised, linear method is the **Fisher Discriminant Analysis (FDA)**. This method projects the data on a single dimension, while maximizing the separation between the classes of the data.

A more sophisticated projection method is **Independent Component Analysis (ICA)**[8]. This method finds a linear mixture of the data, in the

same dimension of the data or lower. ICA attempts to find a mixture matrix such that each of the projections will be as independent as possible from the other projections.

Instead of finding a linear mixture of the feature, it is also possible to find a nonlinear mixture of the data. This is usually done through modifications of the above-mentioned linear methods. Examples of such methods are nonlinear component analysis [33], nonlinear FDA [32], and Kernel PCA[46]. The latter method works by remapping data by way of a kernel function into feature space where the principle components of the data are found.

As a final note on feature selection and feature reduction, one should note that as the ratio between the number of features and the number of training examples increases, it becomes likelier for a noisy and irrelevant feature to seem relevant for the specific set of examples. Indeed, feature selection is sometimes viewed as an ill-posed problem [52], which is why application of such methods should be performed with care. For example, if possible, the feature selection algorithm should be run several times, and the results tested for consistency.

4 Clustering

The second stage of the classification process endeavors to reduce the number of data points by clustering the data and finding representative data points (for example, cluster centers), or by removing superfluous data points. This stage is usually performed using unsupervised methods.

A cluster of points is not a well-defined object. Instead, clusters are defined based on their environment and the scale at which the data is examined. Figure 2 demonstrates the nature of the problem. Two possible definitions for clusters[23] are: (I) Patterns within a cluster are more similar to each other than are patterns belonging to other clusters. (II) A cluster is a volume of high-density points separated from other clusters by a relatively low density volumes. Both these definitions do not suggest a practical solution to the problem of finding clusters. In practice one usually specifies a criterion for joining points into clusters or the number of clusters to be found, and these are used by the clustering algorithm in place of a definition of a cluster. This practicality results in a major drawback of clustering algorithms: A clustering algorithm will find clusters even if there are no clusters in the data.

Returning to the vertebrate classification problem discussed earlier, if we are given data on all vertebrate species, we may find that this comprises of too many training examples. It may be enough to find a representative sample for each of the classes and use it to build the classifier. Clustering algorithms attempt to find such representatives. Note that representative samples can be either actual samples drawn from the data (for example, a human as an example for a mammal) or an average of several samples (i.e. an animal with some given percentage of hair on its body as a representative mammal).

The computational cost of finding an optimal partition of a dataset into a given number of clusters is usually prohibitively high. Therefore, in most cases

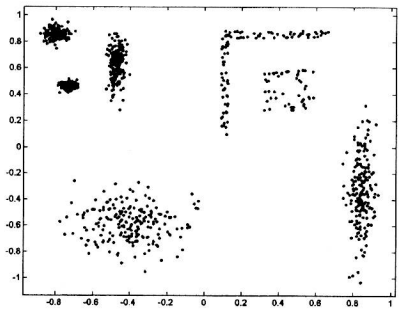


Fig. 2. An example of data points for clustering. Many possible clustering configurations can be made for this data, based on the scale at which the data is examined, the shape of the clusters, etc

clustering algorithms attempt to find a suboptimal partition in a reasonable number of computations. Clustering algorithms can be divided into Top-Down (or partitional) algorithms and Bottom-Up (or hierarchical) algorithms.

A simple example for Bottom-Up algorithms is the **Agglomerative Hierarchical Clustering Algorithm (AGHC)**. This algorithm is an iterative algorithm, which starts by assuming that each data point is a cluster. At each iteration two clusters are merged until a preset number of clusters is reached. The decision on which clusters are to be merged can be done using one of several functions, i.e. distance between cluster centers, distance between the two nearest points in different clusters, etc. AGHC is a very simple, intuitive scheme. However, it is computationally intensive and thus impractical for medium and large datasets.

Top-Down methods are the type more frequently used for clustering due to their lower computational cost, despite the fact that they usually find an inferior solution compared to Bottom-Up algorithms. Probably the most popular amongst Top-Down clustering algorithms in the **K-means algorithm** [28], a pseudo-code of which is given in figure 3. K-means is usually reasonably fast, but care should be taken in the initial setting of the cluster centers so as to attain a good partition of the data. There are probably hundreds of Top-Down clustering algorithms, but popular algorithms include fuzzy k-means [3], Kohonen maps [24], and competitive learning [44].

Recently, with the advent of kernel-based methods several algorithms for clustering using kernels have been suggested (e.g. [2]). The basic idea behind these algorithms is to map the data into a higher dimension using a non-linear function of the input features, and to cluster the data using simple clustering algorithms at the higher dimension. More details regarding kernels are given in the Classification section of this paper. One of the main advantages of kernel methods is that simple clusters (for example, ellipsoid clusters) formed in a higher dimension correspond to complex clusters in the input space. These methods seem to provide excellent clustering results, with reasonable computational costs.

A related class of clustering algorithms are the **Spectral Clustering** methods [37, 11]. These methods first map the data into a matrix representing the distance between the input patterns. The matrix is then projected onto its k largest eigenvectors, and the clustering is performed on this projection. These methods demonstrated impressive results on several datasets, with computational costs slightly higher than those of kernel-based algorithms.

The K-means clustering algorithm

1. Begin initialize N random cluster centers.
2. Assign each of the data points the nearest of the N cluster centers.
3. Recompute the cluster centers by averaging the points assigned to each cluster.
4. Repeat steps 2-4 until the there is no change in the location of the cluster centers.
5. Return the cluster centers.

Fig. 3. Pseudo-code of the K-means clustering algorithm

5 Classification

Classification, the final stage of a pattern classifier, is the process of assigning labels to test patterns, based on previously labeled training patterns. This process is commonly divided into a learning phase, where the classification algorithm is trained, and a classification phase, where the algorithm labels new data.

The general model for statistical pattern classification is one where patterns are drawn from an unknown distribution P, which depends on the label of the data (i.e., $P(x|\omega_i)$ $i = 1, \ldots, N$, where N is the number of labels in the data). During the learning phase the classification algorithm is trained with the goal of minimizing the error that will be obtained when classifying some test data. This error is known as the **risk** or the **expected loss**.

When discussing the pros and cons of classification algorithms, it is important to set criteria for assessing these algorithms. In the following pages we describe several classification algorithms and later summarize (in table 1) their strong and weak points with regard to the following points:

- How small are the classification errors reached by the algorithm?
- What is the computational cost and the memory requirements for both training and testing?
- How difficult is it for a novice user to build and train an efficient classifier?
- Is the algorithm able to learn on-line (i.e. as the data appears, allowing each data point to be addressed only once)?
- Can one gain insight about the problem from examining the trained classifier?

It is important to note that when discussing the classification errors of classifiers one is usually interested in the errors obtained when classifying test data. Many classifiers can be trained to classify all the training data correctly. This