# Bayesian Reasoning and Machine Learning

Extracting value from vast amounts of data presents a major challenge to all those working in computer science and related fields. Machine learning technology is already used to help with this task in a wide range of industrial applications, including search engines, DNA sequencing, stock market analysis and robot locomotion. As its usage becomes more widespread, the skills taught in this book will be invaluable to students.

Designed for final-year undergraduate and graduate students, this gentle introduction is ideally suited to readers without a solid background in linear algebra and calculus. It covers basic probabilistic reasoning to advanced techniques in machine learning, and crucially enables students to construct their own models for real-world problems by teaching them what lies behind the methods. A central conceptual theme is the use of Bayesian modelling to describe and build inference algorithms. Numerous examples and exercises are included in the text. Comprehensive resources for students and instructors are available online.

 

Cambridge University Press
978-0-521-51814-7 - Bayesian Reasoning and Machine Learning
David Barber
Frontmatter
More information

# Bayesian Reasoning and Machine Learning

David Barber
University College London

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE
UNIVERSITY PRESS

University Printing House, Cambridge CB2 8BS, United Kingdom

Cambridge University Press is part of the University of Cambridge.

It furthers the University's mission by disseminating knowledge in the pursuit of education, learning and research at the highest international levels of excellence.

www.cambridge.org
Information on this title: www.cambridge.org/9780521518147



First published 2012
6th printing 2015

Printed in the United Kingdom by TJ International Ltd, Padstow, Cornwall

*A catalogue record for this publication is available from the British Library*

*Library of Congress Cataloguing in Publication data*
Barber, David, 1968–
Bayesian reasoning and machine learning / David Barber.
p. cm.
Includes bibliographical references and index.
ISBN 978-0-521-51814-7
1. Machine learning. 2. Bayesian statistical decision theory. I. Title.
QA267.B347 2012
006.3′1 – dc23 2011035553

ISBN 978-0-521-51814-7 Hardback

Additional resources for this publication at www.cambridge.org/brml and at www.cs.ucl.ac.uk/staff/D.Barber/brml

Cambridge University Press
978-0-521-51814-7 - Bayesian Reasoning and Machine Learning
David Barber
Frontmatter
More information

# CONTENTS

## II Learning in probabilistic models

Cambridge University Press
978-0-521-51814-7 - Bayesian Reasoning and Machine Learning
David Barber
Frontmatter
More information

# PREFACE

## The data explosion

We live in a world that is rich in data, ever increasing in scale. This data comes from many different sources in science (bioinformatics, astronomy, physics, environmental monitoring) and commerce (customer databases, financial transactions, engine monitoring, speech recognition, surveillance, search). Possessing the knowledge as to how to process and extract value from such data is therefore a key and increasingly important skill. Our society also expects ultimately to be able to engage with computers in a natural manner so that computers can 'talk' to humans, 'understand' what they say and 'comprehend' the visual world around them. These are difficult large-scale information processing tasks and represent grand challenges for computer science and related fields. Similarly, there is a desire to control increasingly complex systems, possibly containing many interacting parts, such as in robotics and autonomous navigation. Successfully mastering such systems requires an understanding of the processes underlying their behaviour. Processing and making sense of such large amounts of data from complex systems is therefore a pressing modern-day concern and will likely remain so for the foreseeable future.

## Machine learning

Machine learning is the study of data-driven methods capable of mimicking, understanding and aiding human and biological information processing tasks. In this pursuit, many related issues arise such as how to compress data, interpret and process it. Often these methods are not necessarily directed to mimicking directly human processing but rather to enhancing it, such as in predicting the stock market or retrieving information rapidly. In this probability theory is key since inevitably our limited data and understanding of the problem forces us to address uncertainty. In the broadest sense, machine learning and related fields aim to 'learn something useful' about the environment within which the agent operates. Machine learning is also closely allied with artificial intelligence, with machine learning placing more emphasis on using data to drive and adapt the model.

In the early stages of machine learning and related areas, similar techniques were discovered in relatively isolated research communities. This book presents a unified treatment via graphical models, a marriage between graph and probability theory, facilitating the transference of machine learning concepts between different branches of the mathematical and computational sciences.

## Whom this book is for

The book is designed to appeal to students with only a modest mathematical background in undergraduate calculus and linear algebra. No formal computer science or statistical background is required to follow the book, although a basic familiarity with probability, calculus and linear algebra would be useful. The book should appeal to students from a variety of backgrounds, including computer science, engineering, applied statistics, physics and bioinformatics that wish to gain an entry to probabilistic approaches in machine learning. In order to engage with students, the book introduces fundamental concepts in inference using only minimal reference to algebra and calculus. More mathematical techniques are postponed until as and when required, always with the concept as primary and the mathematics secondary.

The concepts and algorithms are described with the aid of many worked examples. The exercises and demonstrations, together with an accompanying MATLAB toolbox, enable the reader to experiment and more deeply understand the material. The ultimate aim of the book is to enable the reader to construct novel algorithms. The book therefore places an emphasis on skill learning, rather than being a collection of recipes. This is a key aspect since modern applications are often so specialised as to require novel methods. The approach taken throughout is to describe the problem as a graphical model, which is then translated into a mathematical framework, ultimately leading to an algorithmic implementation in the BRMLtoolbox.

The book is primarily aimed at final year undergraduates and graduates without significant experience in mathematics. On completion, the reader should have a good understanding of the techniques, practicalities and philosophies of probabilistic aspects of machine learning and be well equipped to understand more advanced research level material.

## The structure of the book

The book begins with the basic concepts of graphical models and inference. For the independent reader Chapters 1, 2, 3, 4, 5, 9, 10, 13, 14, 15, 16, 17, 21 and 23 would form a good introduction to probabilistic reasoning, modelling and machine learning. The material in Chapters 19, 24, 25 and 28 is more advanced, with the remaining material being of more specialised interest. Note that in each chapter the level of material is of varying difficulty, typically with the more challenging material placed towards the end of each chapter. As an introduction to the area of probabilistic modelling, a course can be constructed from the material as indicated in the chart.

The material from Parts I and II has been successfully used for courses on graphical models. I have also taught an introduction to probabilistic machine learning using material largely from Part III, as indicated. These two courses can be taught separately and a useful approach would be to teach first the graphical models course, followed by a separate probabilistic machine learning course.

A short course on approximate inference can be constructed from introductory material in Part I and the more advanced material in Part V, as indicated. The exact inference methods in Part I can be covered relatively quickly with the material in Part V considered in more depth.

A timeseries course can be made by using primarily the material in Part IV, possibly combined with material from Part I for students that are unfamiliar with probabilistic modelling approaches. Some of this material, particularly in Chapter 25, is more advanced and can be deferred until the end of the course, or considered for a more advanced course.

The references are generally to works at a level consistent with the book material and which are in the most part readily available.

| Part | Chapter | Graphical models course | Probabilistic machine learning course | Approximate inference short course | Timeseries short course | Probabilistic modelling course |
|---|---|---|---|---|---|---|
| Part I: Inference in probabilistic models | 1: Probabilistic reasoning | ● | ○ | ○ | ○ | ● |
| | 2: Basic graph concepts | ● | ○ | ● | ○ | ● |
| | 3: Belief networks | ● | ● | ● | ● | ● |
| | 4: Graphical models | ● | ○ | ● | ● | ○ |
| | 5: Efficient inference in trees | ● | ○ | ● | ● | ○ |
| | 6: The junction tree algorithm | ● | ○ | ● | ○ | ○ |
| | 7: Making decisions | ● | ○ | ○ | ● | ○ |
| Part II: Learning in probabilistic models | 8: Statistics for machine learning | ● | ○ | ○ | ○ | ● |
| | 9: Learning as inference | ● | ○ | ○ | ○ | ● |
| | 10: Naive Bayes | ● | ● | ○ | ○ | ○ |
| | 11: Learning with hidden variables | ● | ● | ○ | ○ | ● |
| | 12: Bayesian model selection | ● | ○ | ○ | ○ | ● |
| Part III: Machine learning | 13: Machine learning concepts | ○ | ● | ○ | ○ | ○ |
| | 14: Nearest neighbour classification | ○ | ○ | ○ | ○ | ○ |
| | 15: Unsupervised linear dimension reduction | ○ | ● | ○ | ○ | ○ |
| | 16: Supervised linear dimension reduction | ○ | ● | ○ | ○ | ○ |
| | 17: Linear models | ○ | ● | ○ | ○ | ○ |
| | 18: Bayesian linear models | ○ | ○ | ○ | ○ | ● |
| | 19: Gaussian processes | ○ | ● | ○ | ○ | ○ |
| | 20: Mixture models | ○ | ● | ○ | ○ | ● |
| | 21: Latent linear models | ○ | ○ | ○ | ○ | ● |
| | 22: Latent ability models | ○ | ○ | ○ | ○ | ● |
| Part IV: Dynamical models | 23: Discrete-state Markov models | ○ | ● | ○ | ● | ○ |
| | 24: Continuous-state Markov models | ○ | ○ | ○ | ● | ● |
| | 25: Switching linear dynamical systems | ○ | ○ | ○ | ● | ○ |
| | 26: Distributed computation | ○ | ● | ○ | ● | ○ |
| Part V: Approximate inference | 27: Sampling | ○ | ○ | ● | ○ | ○ |
| | 28: Deterministic approximate inference | ○ | ○ | ● | ○ | ○ |

## Accompanying code

The BRMLtoolbox is provided to help readers see how mathematical models translate into actual MATLAB code. There is a large number of demos that a lecturer may wish to use or adapt to help illustrate the material. In addition many of the exercises make use of the code, helping the reader gain confidence in the concepts and their application. Along with complete routines for many machine learning methods, the philosophy is to provide low-level routines whose composition intuitively follows the mathematical description of the algorithm. In this way students may easily match the mathematics with the corresponding algorithmic implementation.

## Website

The BRMLtoolbox along with an electronic version of the book is available from

`www.cs.ucl.ac.uk/staff/D.Barber/brml`

Instructors seeking solutions to the exercises can find information at www.cambridge.org/brml, along with additional teaching materials.

## Other books in this area

The literature on machine learning is vast with much relevant literature also contained in statistics, engineering and other physical sciences. A small list of more specialised books that may be referred to for deeper treatments of specific topics is:

- Graphical models
  - *Graphical Models* by S. Lauritzen, Oxford University Press, 1996.
  - *Bayesian Networks and Decision Graphs* by F. Jensen and T. D. Nielsen, Springer-Verlag, 2007.
  - *Probabilistic Networks and Expert Systems* by R. G. Cowell, A. P. Dawid, S. L. Lauritzen and D. J. Spiegelhalter, Springer-Verlag, 1999.
  - *Probabilistic Reasoning in Intelligent Systems* by J. Pearl, Morgan Kaufmann, 1988.
  - *Graphical Models in Applied Multivariate Statistics* by J. Whittaker, Wiley, 1990.
  - *Probabilistic Graphical Models: Principles and Techniques* by D. Koller and N. Friedman, MIT Press, 2009.
- Machine learning and information processing
  - *Information Theory, Inference and Learning Algorithms* by D. J. C. MacKay, Cambridge University Press, 2003.
  - *Pattern Recognition and Machine Learning* by C. M. Bishop, Springer-Verlag, 2006.
  - *An Introduction to Support Vector Machines*, N. Cristianini and J. Shawe-Taylor, Cambridge University Press, 2000.
  - *Gaussian Processes for Machine Learning* by C. E. Rasmussen and C. K. I. Williams, MIT Press, 2006.

## Acknowledgements

Many people have helped this book along the way either in terms of reading, feedback, general insights, allowing me to present their work, or just plain motivation. Amongst these I would like

to thank Dan Cornford, Massimiliano Pontil, Mark Herbster, John Shawe-Taylor, Vladimir Kolmogorov, Yuri Boykov, Tom Minka, Simon Prince, Silvia Chiappa, Bertrand Mesot, Robert Cowell, Ali Taylan Cemgil, David Blei, Jeff Bilmes, David Cohn, David Page, Peter Sollich, Chris Williams, Marc Toussaint, Amos Storkey, Zakria Hussain, Le Chen, Serafín Moral, Milan Studený, Luc De Raedt, Tristan Fletcher, Chris Vryonides, Tom Furmston, Ed Challis and Chris Bracegirdle. I would also like to thank the many students that have helped improve the material during lectures over the years. I'm particularly grateful to Taylan Cemgil for allowing his GraphLayout package to be bundled with the BRMLtoolbox.

The staff at Cambridge University Press have been a delight to work with and I would especially like to thank Heather Bergman for her initial endeavours and the wonderful Diana Gillooly for her continued enthusiasm.

A heartfelt thankyou to my parents and sister – I hope this small token will make them proud. I'm also fortunate to be able to acknowledge the support and generosity of friends throughout. Finally, I'd like to thank Silvia who made it all worthwhile.

# NOTATION

| | | |
|---|---|---|
| $\mathcal{V}$ | A calligraphic symbol typically denotes a set of random variables | *page* 3 |
| $\text{dom}(x)$ | Domain of a variable | 3 |
| $x = \mathsf{x}$ | The variable $x$ is in the state $\mathsf{x}$ | 3 |
| $p(x = \mathsf{tr})$ | Probability of event/variable $x$ being in the state $\mathsf{true}$ | 3 |
| $p(x = \mathsf{fa})$ | Probability of event/variable $x$ being in the state $\mathsf{false}$ | 3 |
| $p(x, y)$ | Probability of $x$ and $y$ | 4 |
| $p(x \cap y)$ | Probability of $x$ and $y$ | 4 |
| $p(x \cup y)$ | Probability of $x$ or $y$ | 4 |
| $p(x\|y)$ | The probability of $x$ conditioned on $y$ | 4 |
| $\mathcal{X} \perp\!\!\!\perp \mathcal{Y} \| \mathcal{Z}$ | Variables $\mathcal{X}$ are independent of variables $\mathcal{Y}$ conditioned on variables $\mathcal{Z}$ | 7 |
| $\mathcal{X} \top\!\!\!\top \mathcal{Y} \| \mathcal{Z}$ | Variables $\mathcal{X}$ are dependent on variables $\mathcal{Y}$ conditioned on variables $\mathcal{Z}$ | 7 |
| $\int_x f(x)$ | For continuous variables this is shorthand for $\int_x f(x)dx$ and for discrete variables means summation over the states of $x$, $\sum_x f(x)$ | 14 |
| $\mathbb{I}[S]$ | Indicator : has value 1 if the statement $S$ is true, 0 otherwise | 16 |
| $\text{pa}(x)$ | The parents of node $x$ | 24 |
| $\text{ch}(x)$ | The children of node $x$ | 24 |
| $\text{ne}(x)$ | Neighbours of node $x$ | 24 |
| $\text{dim}\,(x)$ | For a discrete variable $x$, this denotes the number of states $x$ can take | 34 |
| $\langle f(x) \rangle_{p(x)}$ | The average of the function $f(x)$ with respect to the distribution $p(x)$ | 170 |
| $\delta(a, b)$ | Delta function. For discrete $a$, $b$, this is the Kronecker delta, $\delta_{a,b}$ and for continuous $a$, $b$ the Dirac delta function $\delta(a - b)$ | 172 |
| $\text{dim}(\mathbf{x})$ | The dimension of the vector/matrix $\mathbf{x}$ | 183 |
| $\sharp\,(x = s, y = t)$ | The number of times $x$ is in state $s$ and $y$ in state $t$ simultaneously | 207 |
| $\sharp_y^x$ | The number of times variable $x$ is in state $y$ | 293 |
| $\mathcal{D}$ | Dataset | 303 |
| $n$ | Data index | 303 |
| $N$ | Number of dataset training points | 303 |
| $\mathbf{S}$ | Sample Covariance matrix | 331 |
| $\sigma(x)$ | The logistic sigmoid $1/(1 + \exp(-x))$ | 371 |
| $\text{erf}(x)$ | The (Gaussian) error function | 372 |
| $x_{a:b}$ | $x_a, x_{a+1}, \ldots, x_b$ | 372 |
| $i \sim j$ | The set of unique neighbouring edges on a graph | 624 |
| $\mathbf{I}_m$ | The $m \times m$ identity matrix | 644 |

# BRMLTOOLBOX

The BRMLTOOLBOX is a lightweight set of routines that enables the reader to experiment with concepts in graph theory, probability theory and machine learning. The code contains basic routines for manipulating discrete variable distributions, along with more limited support for continuous variables. In addition there are many hard-coded standard machine learning algorithms. The website contains also a complete list of all the teaching demos and related exercise material.

## BRMLTOOLKIT

### Graph theory

| | |
|---|---|
| `ancestors` | - Return the ancestors of nodes x in DAG A |
| `ancestralorder` | - Return the ancestral order or the DAG A (oldest first) |
| `descendents` | - Return the descendents of nodes x in DAG A |
| `children` | - Return the children of variable x given adjacency matrix A |
| `edges` | - Return edge list from adjacency matrix A |
| `elimtri` | - Return a variable elimination sequence for a triangulated graph |
| `connectedComponents` | - Find the connected components of an adjacency matrix |
| `istree` | - Check if graph is singly connected |
| `neigh` | - Find the neighbours of vertex v on a graph with adjacency matrix G |
| `noselfpath` | - Return a path excluding self-transitions |
| `parents` | - Return the parents of variable x given adjacency matrix A |
| `spantree` | - Find a spanning tree from an edge list |
| `triangulate` | - Triangulate adjacency matrix A |
| `triangulatePorder` | - Triangulate adjacency matrix A according to a partial ordering |

### Potential manipulation

| | |
|---|---|
| `condpot` | - Return a potential conditioned on another variable |
| `changevar` | - Change variable names in a potential |
| `dag` | - Return the adjacency matrix (zeros on diagonal) for a belief network |
| `deltapot` | - A delta function potential |
| `disptable` | - Print the table of a potential |
| `divpots` | - Divide potential pota by potb |
| `drawFG` | - Draw the factor graph A |
| `drawID` | - Plot an influence diagram |
| `drawJTree` | - Plot a junction tree |
| `drawNet` | - Plot network |
| `evalpot` | - Evaluate the table of a potential when variables are set |
| `exppot` | - Exponential of a potential |
| `eyepot` | - Return a unit potential |
| `grouppot` | - Form a potential based on grouping variables together |
| `groupstate` | - Find the state of the group variables corresponding to a given ungrouped state |
| `logpot` | - Logarithm of the potential |
| `markov` | - Return a symmetric adjacency matrix of Markov network in pot |
| `maxpot` | - Maximise a potential over variables |
| `maxsumpot` | - Maximise or sum a potential over variables |
| `multpots` | - Multiply potentials into a single potential |

| | |
|---|---|
| `numstates` | - Number of states of the variables in a potential |
| `orderpot` | - Return potential with variables reordered according to order |
| `orderpotfields` | - Order the fields of the potential, creating blank entries where necessary |
| `potsample` | - Draw sample from a single potential |
| `potscontainingonly` | - Returns those potential numbers that contain only the required variables |
| `potvariables` | - Returns information about all variables in a set of potentials |
| `setevpot` | - Sets variables in a potential into evidential states |
| `setpot` | - Sets potential variables to specified states |
| `setstate` | - Set a potential's specified joint state to a specified value |
| `squeezepots` | - Eliminate redundant potentials (those contained wholly within another) |
| `sumpot` | - Sum potential pot over variables |
| `sumpotID` | - Return the summed probability and utility tables from an ID |
| `sumpots` | - Sum a set of potentials |
| `table` | - Return the potential table |
| `ungrouppot` | - Form a potential based on ungrouping variables |
| `uniquepots` | - Eliminate redundant potentials (those contained wholly within another) |
| `whichpot` | - Returns potentials that contain a set of variables |

Routines also extend the toolbox to deal with Gaussian potentials: `multpotsGaussianMoment.m`, `sumpotGaussianCanonical.m`, `sumpotGaussianMoment.m`, `multpotsGaussianCanonical.m` See `demoSumprodGaussCanon.m`, `demoSumprodGaussCanonLDS.m`, `demoSumprodGaussMoment.m`

## Inference

| | |
|---|---|
| `absorb` | - Update potentials in absorption message passing on a junction tree |
| `absorption` | - Perform full round of absorption on a junction tree |
| `absorptionID` | - Perform full round of absorption on an influence diagram |
| `ancestralsample` | - Ancestral sampling from a belief network |
| `binaryMRFmap` | - Get the MAP assignment for a binary MRF with positive W |
| `bucketelim` | - Bucket elimination on a set of potentials |
| `condindep` | - Conditional independence check using graph of variable interactions |
| `condindepEmp` | - Compute the empirical log Bayes factor and MI for independence/dependence |
| `condindepPot` | - Numerical conditional independence measure |
| `condMI` | - Conditional mutual information I(x,y\|z) of a potential |
| `FactorConnectingVariable` | - Factor nodes connecting to a set of variables |
| `FactorGraph` | - Returns a factor graph adjacency matrix based on potentials |
| `IDvars` | - Probability and decision variables from a partial order |
| `jtassignpot` | - Assign potentials to cliques in a junction tree |
| `jtree` | - Setup a junction tree based on a set of potentials |
| `jtreeID` | - Setup a junction tree based on an influence diagram |
| `LoopyBP` | - Loopy belief propagation using sum-product algorithm |
| `MaxFlow` | - Ford Fulkerson max-flow min-cut algorithm (breadth first search) |
| `maxNpot` | - Find the N most probable values and states in a potential |
| `maxNprodFG` | - N-max-product algorithm on a factor graph (returns the Nmax most probable states) |
| `maxprodFG` | - Max-product algorithm on a factor graph |
| `MDPemDeterministicPolicy` | - Solve MDP using EM with deterministic policy |
| `MDPsolve` | - Solve a Markov decision process |
| `MesstoFact` | - Returns the message numbers that connect into factor potential |
| `metropolis` | - Metropolis sample |
| `mostprobablepath` | - Find the most probable path in a Markov chain |
| `mostprobablepathmult` | - Find the all source all sink most probable paths in a Markov chain |
| `sumprodFG` | - Sum-product algorithm on a factor graph represented by A |

## Specific models

| | |
|---|---|
| `ARlds` | - Learn AR coefficients using a linear dynamical system |
| `ARtrain` | - Fit auto-regressive (AR) coefficients of order L to v. |
| `BayesLinReg` | - Bayesian linear regression training using basis functions phi(x) |
| `BayesLogRegressionRVM` | - Bayesian logistic regression with the relevance vector machine |
| `CanonVar` | - Canonical variates (no post rotation of variates) |

| | |
|---|---|
| `cca` | - Canonical correlation analysis |
| `covfnGE` | - Gamma exponential covariance function |
| `FA` | - Factor analysis |
| `GMMem` | - Fit a mixture of Gaussian to the data X using EM |
| `GPclass` | - Gaussian process binary classification |
| `GPreg` | - Gaussian process regression |
| `HebbML` | - Learn a sequence for a Hopfield network |
| `HMMbackward` | - HMM backward pass |
| `HMMbackwardSAR` | - Backward pass (beta method) for the switching Auto-regressive HMM |
| `HMMem` | - EM algorithm for HMM |
| `HMMforward` | - HMM forward pass |
| `HMMforwardSAR` | - Switching auto-regressive HMM with switches updated only every Tskip timesteps |
| `HMMgamma` | - HMM posterior smoothing using the Rauch–Tung–Striebel correction method |
| `yHMMsmooth` | - Smoothing for a hidden Markov model (HMM) |
| `HMMsmoothSAR` | - Switching auto-regressive HMM smoothing |
| `HMMviterbi` | - Viterbi most likely joint hidden state of HMM |
| `kernel` | - A kernel evaluated at two points |
| `Kmeans` | - K-means clustering algorithm |
| `LDSbackward` | - Full backward pass for a latent linear dynamical system (RTS correction method) |
| `LDSbackwardUpdate` | - Single backward update for a latent linear dynamical system (RTS smoothing update) |
| `LDSforward` | - Full forward pass for a latent linear dynamical system (Kalman filter) |
| `LDSforwardUpdate` | - Single forward update for a latent linear dynamical system (Kalman filter) |
| `LDSsmooth` | - Linear dynamical system: filtering and smoothing |
| `LDSsubspace` | - Subspace method for identifying linear dynamical system |
| `LogReg` | - Learning logistic linear regression using gradient ascent |
| `MIXprodBern` | - EM training of a mixture of a product of Bernoulli distributions |
| `mixMarkov` | - EM training for a mixture of Markov models |
| `NaiveBayesDirichletTest` | - Naive Bayes prediction having used a Dirichlet prior for training |
| `NaiveBayesDirichletTrain` | - Naive Bayes training using a Dirichlet prior |
| `NaiveBayesTest` | - Test Naive Bayes Bernoulli distribution after max likelihood training |
| `NaiveBayesTrain` | - Train Naive Bayes Bernoulli distribution using max likelihood |
| `nearNeigh` | - Nearest neighbour classification |
| `pca` | - Principal components analysis |
| `plsa` | - Probabilistic latent semantic analysis |
| `plsaCond` | - Conditional PLSA (probabilistic latent semantic analysis) |
| `rbf` | - Radial basis function output |
| `SARlearn` | - EM training of a switching AR model |
| `SLDSbackward` | - Backward pass using a mixture of Gaussians |
| `SLDSforward` | - Switching latent linear dynamical system Gaussian sum forward pass |
| `SLDSmargGauss` | - Compute the single Gaussian from a weighted SLDS mixture |
| `softloss` | - Soft loss function |
| `svdm` | - Singular value decomposition with missing values |
| `SVMtrain` | - Train a support vector machine |

## General

| | |
|---|---|
| `argmax` | - Performs argmax returning the index and value |
| `assign` | - Assigns values to variables |
| `betaXbiggerY` | - p(x>y) for x∼Beta(a,b), y∼Beta(c,d) |
| `bar3zcolor` | - Plot a 3D bar plot of the matrix Z |
| `avsigmaGauss` | - Average of a logistic sigmoid under a Gaussian |
| `cap` | - Cap x at absolute value c |
| `chi2test` | - Inverse of the chi square cumulative density |
| `count` | - For a data matrix (each column is a datapoint), return the state counts |
| `condexp` | - Compute normalised p proportional to exp(logp) |
| `condp` | - Make a conditional distribution from the matrix |
| `dirrnd` | - Samples from a Dirichlet distribution |
| `field2cell` | - Place the field of a structure in a cell |
| `GaussCond` | - Return the mean and covariance of a conditioned Gaussian |

| | |
|---|---|
| `hinton` | - Plot a Hinton diagram |
| `ind2subv` | - Subscript vector from linear index |
| `ismember_sorted` | - True for member of sorted set |
| `lengthcell` | - Length of each cell entry |
| `logdet` | - Log determinant of a positive definite matrix computed in a numerically stable manner |
| `logeps` | - log(x+eps) |
| `logGaussGamma` | - Unnormalised log of the Gauss-Gamma distribution |
| `logsumexp` | - Compute log(sum(exp(a).*b)) valid for large a |
| `logZdirichlet` | - Log normalisation constant of a Dirichlet distribution with parameter u |
| `majority` | - Return majority values in each column on a matrix |
| `maxarray` | - Maximise a multi-dimensional array over a set of dimensions |
| `maxNarray` | - Find the highest values and states of an array over a set of dimensions |
| `mix2mix` | - Fit a mixture of Gaussians with another mixture of Gaussians |
| `mvrandn` | - Samples from a multivariate Normal (Gaussian) distribution |
| `mygamrnd` | - Gamma random variate generator |
| `mynanmean` | - Mean of values that are not nan |
| `mynansum` | - Sum of values that are not nan |
| `mynchoosek` | - Binomial coefficient v choose k |
| `myones` | - Same as ones(x), but if x is a scalar, interprets as ones([x 1]) |
| `myrand` | - Same as rand(x) but if x is a scalar interprets as rand([x 1]) |
| `myzeros` | - Same as zeros(x) but if x is a scalar interprets as zeros([x 1]) |
| `normp` | - Make a normalised distribution from an array |
| `randgen` | - Generates discrete random variables given the pdf |
| `replace` | - Replace instances of a value with another value |
| `sigma` | - 1./(1+exp(-x)) |
| `sigmoid` | - 1./(1+exp(-beta*x)) |
| `sqdist` | - Square distance between vectors in x and y |
| `subv2ind` | - Linear index from subscript vector. |
| `sumlog` | - sum(log(x)) with a cutoff at 10e-200 |

## Miscellaneous

| | |
|---|---|
| `compat` | - Compatibility of object F being in position h for image v on grid Gx,Gy |
| `logp` | - The logarithm of a specific non-Gaussian distribution |
| `placeobject` | - Place the object F at position h in grid Gx,Gy |
| `plotCov` | - Return points for plotting an ellipse of a covariance |
| `pointsCov` | - Unit variance contours of a 2D Gaussian with mean m and covariance S |
| `setup` | - Run me at initialisation – checks for bugs in matlab and initialises path |
| `validgridposition` | - Returns 1 if point is on a defined grid |