fzymodel

— Version 2.9 as of 29 July 2020 —

A neuro-fuzzy modeling tool

Manfred Männle, 2020

Contents

Takagi/Sugeno/Kang Fuzzy Modeling

`fzymodel` is a collection of tools for fully automatic modeling of Takagi/Sugeno/Kang-type neuro-fuzzy models (systems identification), normalization of data sets, and systems simulation.

Here is a list of highlights:

- sigmoidal or trapezoidal fuzzy sets
- RPROP and SVD parameter optimization (deep learning), heuristic search of fuzzy rules (`fzymodel`)
- estimation, i.e. one-step prediction (`fzyestim`)
- simulation, i.e. multiple step prediction (`fzysimul`)
- LaTeX output of fuzzy rules, graphical visualization of fuzzy sets (`fzy2sets`)
- normalization of input data (`fzymkdat`)

For an introduction to the philosophy of `fzymodel` go on reading the Introduction.

1 License

MIT License

2 Introduction

The `fzymodel` package provides a set of tools to build and evaluate TSK-type neuro-fuzzy model.

In order to make investigations and results easier to reproduce it is recommended to use `fzymodel` and its derivatives through shell scripts or similar. Therefore, all the tools are command line based and provide options to set relevant parameters.

For a usage description type `fzymodel -h`, `fzyestim -h`, `fzysimul -h`, `fzymkdat -h`, or `fzy2sets -h`.

The algorithm for structure and parameter optimization (deep learning) is described in my scientific papers and PhD thesis.

3 Installation

For installation go to the `src` directory, edit the `Makefile` (in particular the destination path), and type `make install`.

The installation is tested for Sun Solaris and Linux. You must use GNU make, gcc is recommended. You need to have the TNT (template numerical toolkit) library installed; set the include path `INCPATH` accordingly. You need to have `flex` and `bison` properly installed.

To remake the documentation you need `doc++` version 3.3 or later and `pdflatex`. For this, go to the `src` directory and type `make doc`, then go to the `doc/tex` directory and check `doc.pdf`.

If you encounter problems during installation or usage, contact the author at `fzymodel@maennle.org`.

The source code is split into the files:

- `global.{hh|cc}`: command line options and global variables and functions
- `main.{hh|cc}`: parse command line, print help, catch exceptions
- `fzymodel.{hh|cc}`: TSK modeling
- `fzyestim.{hh|cc}`: estimation, one-step prediction
- `fzysimul.{hh|cc}`: simulation, multiple step prediction
- `fzy2sets.{hh|cc}`: print tex description of a model
- `fzymkdat.{hh|cc}`: prepare (normalize) training/validation data
- `fmodel.{hh|cc}`: fuzzy model class
- `data.{hh|cc}`: training/validation data class
- `svd.{hh|cc}`: singular value decomposition
- `data_lex.{l|h}`: parse data files
- `fzy_prs.y, fzy_lex.{l|h}`: parse fuzzy model files (`.fzy`)

4 Code description

Names

4.1 main

Names

4.1.1 int **main** (int argc, char* argv[], char* env[])

The main program.

Parses command line, checks command line arguments and then calls the appropriate module (e.g., fzymodel, fzyestim, etc.).

Return Value: `The` return status of the modeling/simulation (0 = ok, else error).

Parameters:
- `argc` The number of commandline parameters.
- `argv` The commandline parameters.

Author: Manfred Männle

Version: *Revision* : 2.9 *Date* : 2020 − 08 − 2910 : 47 : 39 + 00 : 00 : 02

4.1.2 void **print_options** (ofstream& strm, int argc, char* argv[], char* env[])

Print the command line arguments to a stream.

Parameters:

`strm`	output file stream
`argc`	no. of command line arguments
`argv`	command line argument vector
`env`	environment vector

4.2

typedef `Real`

The type `Real` is determined by the definition of either `LONGDOUBLE`, `DOUBLE`, or `FLOAT`. The constants `REAL_MAX`, `REAL_MIN`, `REAL_EPSILON`, and `REAL_NAME` are then set accordingly to the type of `Real`.

4.3

enum `mode_type`

Mode enumeration: UNDEFD_MODE == undefined mode; (initial value) MODELING == build fuzzy model based on training and validation data ESTIMATION == one-step prediction using data from file SIMULATION == multiple step prediction using data from file PRINST_SETS == print LaTeX and graphical descrition of a fuzzy model MAKE_DATA == prepare data for fuzzy modeling

4.4

Global constants

Names

4.4.1

extern const char* **model_prefix**

prefix of fuzzy model output file

prefix of fuzzy model output file

4.4.2

extern const char* **fzy2pixl_prefix**

prefix of pixel output file

prefix of pixel output file

4.4.3

extern const char* **data_prefix**

prefix of data generation file

prefix of data generation file

4.4.4

extern const char* **estimation_prefix**

prefix of estimation output file

prefix of estimation output file

4.4.5

extern const char* **simulation_prefix**

prefix of simulation output file

prefix of simulation output file

4.4.6

extern const char* **model_suffix**

suffix of fuzzy model output file

suffix of fuzzy model output file

4.4.7

extern const char* **r2_suffix**

suffix of r2 output file

suffix of r2 output file

4.4.8

extern const char* **error_suffix**

suffix of error output file

suffix of error output file

4.4.9

extern const char* **log_suffix**

suffix of logfile

suffix of logfile

4.4.10

extern const char* **output_suffix**

suffix of output file

suffix of output file

4.4.11

extern const char* **pixl_suffix**

suffix of pixel file (for graphical description of a fuzzy set)

suffix of pixel file (for graphical description of a fuzzy set)

4.4.12

extern const char* **eps_suffix**

suffix of eps file (graphical eps description of a fuzzy set)

suffix of eps file (graphical eps description of a fuzzy set)

4.4.13

extern const char* **tex_suffix**

suffix of LaTeX file (LaTeX description of a fuzzy model)

suffix of LaTeX file (LaTeX description of a fuzzy model)

4.4.14

extern const char* **release**

software release number

software release number

4.4.15

extern const char* **release_date**

software release date

software release date

4.5

Fuzzy Set constants

Names

Constants for Fuzzy Sets.

4.5.1

const Real **mu_left**

definition of right border of input space

definition of right border of input space

4.5.2

const Real **mu_right**

definition of left border of input space

definition of left border of input space

4.5.3

const Real **cons_0**

initialize consequence parameters c

initialize consequence parameters c

4.5.4

const Real **delta_mu_0**

initialize delta μ

initialize delta μ

4.5.5

const Real **delta_cons_0**

initialize delta c

initialize delta c

4.5.6

const Real **sigma_0**

initialize σ

initialize σ

4.5.7

const Real **delta_sigma_0**

initialize delta σ

initialize delta σ

4.5.8

const Real **max_delta_mu**

upper border of delta μ

upper border of delta μ

4.5.9

const Real **max_delta_cons**

upper border of delta c

upper border of delta c

4.5.10

const Real **max_delta_sigma**

upper border of delta σ

upper border of delta σ

4.5.11

const Real **min_delta_mu**

lower border of delta μ

lower border of delta μ

4.5.12

const Real **min_delta_cons**

lower border of delta c

lower border of delta c

4.5.13

const Real **min_delta_sigma**

lower border of delta σ

lower border of delta σ

4.5.14

const Real **max_sigma_init**

upper border of σ

upper border of σ

4.5.15

const Real **min_sigma_init**

lower border of σ

lower border of σ

4.5.16

const Real **eta_plus**

RPROP step lengthening for successful step

RPROP step lengthening for successful step

4.5.17

const Real **eta_minus**

RPROP step shortening for unsuccessful step

RPROP step shortening for unsuccessful step

4.6 Global variables

Names

4.6.1

Options and settings for all modes

Names

4.6.1.1

extern mode_type **mode**

the mode (modeling, estimation, simulation, etc)

the mode (modeling, estimation, simulation, etc)

4.6.1.2

extern char* **prgname**

program name

program name

4.6.1.3

extern char* **binpath**

binary path

binary path

4.6.1.4

extern int **tracelevel**

trace level

trace level

4.6.1.5

extern int **verbose**

verbose level

verbose level

4.6.1.6

extern int **quiet**

inhibit verbose output

inhibit verbose output

4.6.1.7

extern string **basefilename**

basic filename of output files

basic filename of output files

4.6.1.8

extern string **filename_extension**

output files' filename extension

output files' filename extension

4.6.1.9

extern string **tracefilename**

name of trace file

name of trace file

4.6.1.10

extern ofstream **tracefile**

trace log stream

trace log stream

4.6.1.11

extern string **logfilename**

name of verbose log file

name of verbose log file

4.6.1.12

extern ofstream **logfile**

verbose log stream

verbose log stream

4.6.1.13

extern int **denormalize**

take back normalization

take back normalization

4.6.2

Options for `mode == MODELING`

Names

4.6.2.1

extern algo_type **optimization**

optimization algorithm

optimization algorithm

4.6.2.2

extern size_t **consequence_dimension**

dimension of fuzzy rules' consequences

dimension of fuzzy rules' consequences

4.6.2.3

extern int **norm**

error norm (eg, 1 == L1-norm, 2 == L2-norm)

error norm (eg, 1 == L1-norm, 2 == L2-norm)

4.6.2.4

extern int **local_cons_optimization**

local consequence optimization

local consequence optimization

4.6.2.5

extern int **shortcut**

use shortcut for Kang's heuristic search

use shortcut for Kang's heuristic search

4.6.2.6

extern int **reset_cons**

reset consequence parameters

reset consequence parameters

4.6.2.7

extern int **update_premise**

update premise when optimizing

update premise when optimizing

4.6.2.8

extern size_t **max_n_rules**

limit no.

limit no. of fuzzy rules

4.6.2.9

extern size_t **min_n_rules**

minimal no.

minimal no. of fuzzy rules

4.6.2.10

extern int **adjacent_equal_mu**

same mu of adjacent fuzzy sets (ie, input space divisions)

same mu of adjacent fuzzy sets (ie, input space divisions)

4.6.2.11

extern size_t **steps_per_validation**

take mean of several optimization iterations before validation

take mean of several optimization iterations before validation

4.6.2.12

extern int **consequence_optimize_SVD**

optimize the consequence of each new candidate model by SVD

optimize the consequence of each new candidate model by SVD

4.6.2.13

extern size_t **max_opt_iterations_parallel**

maximal no.

maximal no. of optimization iterations for parallel tuning

4.6.2.14

extern size_t **min_opt_iterations**

minimal no.

minimal no. of optimization iterations

4.6.2.15

extern size_t **max_opt_iterations**

maximal no.

maximal no. of optimization iterations

4.6.2.16

extern Real **max_sigma**

maximal sigma of fuzzy sets

maximal sigma of fuzzy sets

4.6.2.17

extern Real **min_sigma**

minimal sigma of fuzzy sets

minimal sigma of fuzzy sets

4.6.2.18

extern Real **R2_improvement**

terminate modeling when R2 improvement is below

terminate modeling when R2 improvement is below

4.6.2.19

extern Real **best_R2_improvement**

terminate modeling when best model's R2 improvement is below

terminate modeling when best model's R2 improvement is below

4.6.2.20

extern int **optimize_epoch_best**

no.

no. of additional optimization iterations at each epoch's best model

4.6.2.21

extern int **optimize_global_best**

no.

no. of additional optimization iterations at the globally best model

4.6.2.22

extern Real **alpha**

gradient descent learning rate

gradient descent learning rate

4.6.2.23

extern Real **beta**

gradient descent momentum

gradient descent momentum

4.6.2.24

extern algo_type **parallel_optimization**

algorithm for parameter optimization (fine tuning) using parallel model

algorithm for parameter optimization (fine tuning) using parallel model

4.6.3

Options for `mode == PRINT_SETS`

Names

4.6.3.1

extern size_t **n_pixels**

no.

no. of interpolation pixels for fuzzy sets' graphical output

4.6.3.2

extern int **global_fset_value**

normalize each fset value as F(x)/(sum Fi(x))

normalize each fset value as F(x)/(sum Fi(x))

4.6.3.3

extern int **scale**

scale of y-axis

scale of y-axis

4.6.3.4

extern int **tex_precision**

precision of numbers to beeing output in the TeX file

precision of numbers to beeing output in the TeX file

4.6.3.5

extern int **execute_gnuplot_system_call**

execute gnuplot system call; 0==no, 1==yes

execute gnuplot system call; 0==no, 1==yes

4.6.3.6

extern string **gnuplot_file_extension**

gnuplot script file extension, usually "gpl"

gnuplot script file extension, usually "gpl"

4.6.4

Options for `mode == ESTIMATION or SIMULATION.`

Names

4.6.4.1

extern Real **error_offset**

error offset (only added when printing the RMS into a file)

error offset (only added when printing the RMS into a file)

4.6.4.2

extern int **page_hinkley**

perform Page-Hinkley test: first column increase, second decrease

perform Page-Hinkley test: first column increase, second decrease

4.6.4.3

extern Real **page_hinkley_mu_0**

mean value

mean value

4.6.4.4

extern Real **page_hinkley_nu_inc**

increase step to detect

increase step to detect

4.6.4.5

extern Real **page_hinkley_nu_dec**

decrease step to detect

decrease step to detect

4.6.4.6

extern Real **page_hinkley_lambda**

sensitivity lambda

sensitivity lambda

4.6.5

Options for `mode == SIMULATION`.

Names

4.6.5.1

extern int **order**

order of y (y-1, y-2, , y-order)

order of y (y-1, y-2, , y-order)

4.6.6

Options for `mode == MAKE_DATA`.

Names

4.6.6.1

extern int **n_training_data**

no.

no. of learning/training patterns

4.6.6.2

extern int **n_skip_data**

no.

no. patterns to skip at the beginning

4.6.6.3

extern vector<int> **dynamic_orders**

dynamic orders of u and y

dynamic orders of u and y

4.6.6.4

extern int **ut_regressor**

start with u(t) as regressor (instead of u(t-1))

start with u(t) as regressor (instead of u(t-1))

4.6.7

Options for `mode == NORMALIZE`.

Names

4.6.7.1

extern int **scale_only_used_data**

compute scale by using only some data, instead of by the whole set

compute scale by using only some data, instead of by the whole set

4.6.7.2

extern int **scale_by_standard_deviation**

scale by standard deviation (default)

scale by standard deviation (default)

4.6.7.3

extern int **scale_by_range**

scale by range

scale by range

4.6.7.4

extern Real **sigma_factor**

*scale by (sigma_factor * stddev)*

scale by (sigma_factor * stddev)

4.7 Help functions

Names

4.7.1

template<class T> void **tracemsg** (int level, const string& name, const T& object)

print trace information

4.7.2

template<class T> void **verbose** (int level, const string& name, const T& object)

print verbose information

4.7.3

inline Real **sign** (Real x)

signum function

4.7.4

string **itos** (int i)

integer to string

4.7.5

string **dtos** (double r)

double to string

4.7.6

string **dtos2** (double r)

double to string

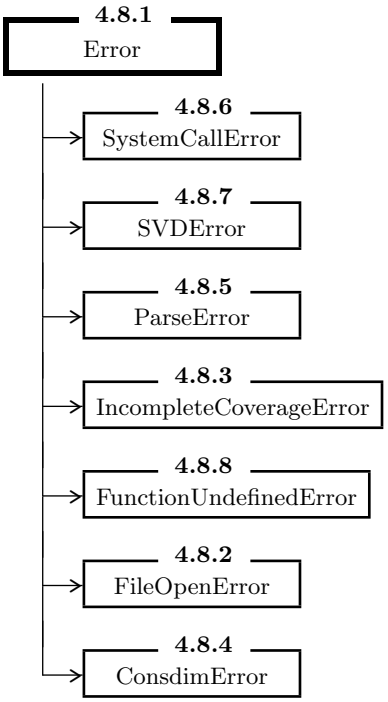
4.8 Error and exception handling

Names

4.8.1

class **Error**

Inheritance

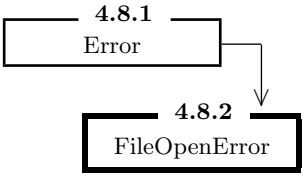


Error base class.

4.8.2

class **FileOpenError** : public Error

Inheritance

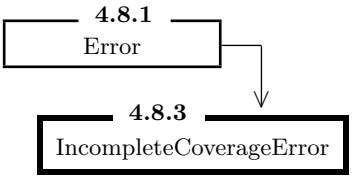


Error when opening a file or file not present.

4.8.3

class **IncompleteCoverageError** : public Error

Inheritance

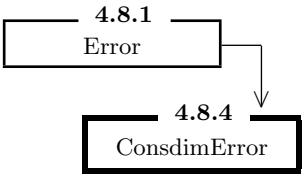


Input value found that is not covered by a model's rule.

4.8.4

class **ConsdimError** : public Error

Inheritance

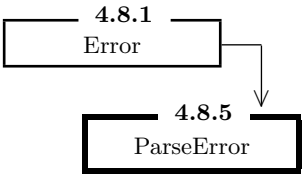


Invalid consequence dimension.

4.8.5

class **ParseError** : public Error

Inheritance

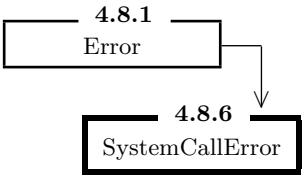


Parse error when loading a fuzzy model from file.

4.8.6

class **SystemCallError** : public Error

Inheritance

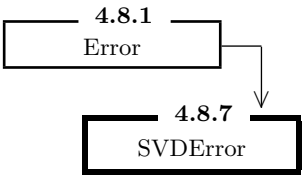


System call was unsuccessful.

4.8.7

class **SVDError** : public Error

Inheritance

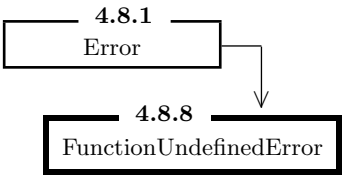


Singular Value Decomposition (SVD) unsuccessful.

4.8.8

class **FunctionUndefinedError** : public Error

Inheritance



Input value found where function is not defined

4.9 Main modules

Names

4.9.1

void **fzymodel** (char* learnfilename, char* validationfilename) throw(Error)

Build a fuzzy model, based on learning and validation data.

The output file names are created using the input file name as prefix: Writes estimation results to mod_[filename_extension]c[C]_r[R].out, best models to mod_[filename_extension]c[C]_r[R].mod, R2 to mod_[filename_extension]c[C].r2, and error to mod_[filename_extension]c[C].err

4.9.2

void **fzyestim** (char* fzyfilename, char* datafilename) throw(Error)

Write the one-step (estimation) or multiple step prediction (simulation) of a fuzzy model for a given data set

The output file names are created using the input file name as prefix: Writes estimation results to est_[filename].out or simulation results to sim_[filename].out, R2 to est_[filename_extension].r2 or sim_[filename_extension].r2, and error to est_[filename].err or sim_[filename].err.

4.9.3

void **fzy2sets** (char* infilename, char* normfilename) throw(Error)

Write a LaTeX and a graphical description of a fuzzy model

Load a Fuzzy Model from file and write a LaTeX description and a graphical description (interpolation points) into files. The output file names are created using the input file name as prefix.

4.9.4

void **fzymkdat** (char* normfilename, char* infilename, char* outfilename) throw(Error)

Prepare data for fuzzy modeling (order shift and normalization).

4.10

Fuzzy Set funtions

Names

4.10.1

inline Real **trapezoid** (Real sigma, Real u)

Membership value of trapezoidal fuzzy set.

Return Value: `membership` value
Parameters: `sigma` steepness
`u` position

4.10.2

Real **sigmoid** (Real x)

Membership value of sigmoidal fuzzy set.

Return Value: `membership` value
Parameters: `x` position

4.10.3

inline Real **t_norm** (Real a, Real b)

Product t-norm.

4.10.4

inline Real **inference** (Real prem, Real cons)

Product inference.

4.11

class **FSet**

Public Members

Protected Members

Fuzzy Set.

4.11.3

Real **F** (Real u)

compute the membership value

compute the membership value

4.11.1

inline void **limit_delta_mu** (void)

check border of $\Delta\mu$

check border of $\Delta\mu$

4.11.2

inline void **limit_delta_sigma** (void)

check border of $\Delta\sigma$

check border of $\Delta\sigma$

4.12

typedef **Premise**

Fuzzy Rule's Premise: a vector of pointers to Fuzzy Sets.

4.13

typedef **Consequence**

Fuzzy Rule's Consequence: a vector of its parameters.

4.14

class **FRule**

Public Members

Protected Members

Fuzzy Rule.

4.14.2

void **copy** (const FRule& r, FSet* fset_begin, const FSet* r_fset_begin)

copy a Fuzzy Rule by value

copy a Fuzzy Rule by value

4.14.3

Real **consvalue** (const Uvector& u) const

compute a Fuzzy Rule's consequence value for a given input vector

compute a Fuzzy Rule's consequence value for a given input vector

4.14.4

Real **premvalue** (const Uvector& u) const

compute a Fuzzy Rule's premise value for a given input vector

compute a Fuzzy Rule's premise value for a given input vector

4.14.5

void **refine** (FRule* r, size_t u, FSet* newleftfset, FSet* newrightfset)

refine a Fuzzy Rule in uindex u; save additional new rule under FRule r*

refine a Fuzzy Rule in uindex u; save additional new rule under FRule* r

4.14.1

Premise **prem_**

4.15

typedef **FSetContainer**

Fuzzy Set Container: a vector of Fuzzy Sets

4.16

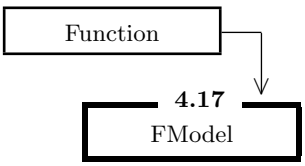
typedef **FRuleContainer**

Fuzzy Rule Container: a vector of Fuzzy Rules

4.17

class **FModel** : public Function

Inheritance



Public Members

Protected Members

Fuzzy Model.

4.17.13

FModel ()

empty model

empty model

4.17.14

FModel (const Data& d, size_t cons_dimension)

initial one-rule model

initial one-rule model

4.17.15

void **copy** (const FModel& oldmodel)

copy a whole model

copy a whole model

4.17.16

FModel (const FModel& oldmodel, size_t ruleindex, size_t uindex)

refine a model's structure in rule rindex at index uindex

refine a model's structure in rule rindex at index uindex

4.17.17

void **load** (char* filename) throw(Error)

load a model from file

load a model from file

4.17.18

Real **sum_w** (const Uvector& u)

w = sum(premise values)

w = sum(premise values)

4.17.19

Real **y_hat** (const Uvector& u) throw(Error)

feedforward step; returns $\hat{y}$

feedforward step; returns $\hat{y}$

4.17.20

void **optimize_SVD** (const Data& d) throw(Error)

optimize consequence parameters using SVD

optimize consequence parameters using SVD

4.17.21

void **reset_consequences** (void)

reset consequence parameters

reset consequence parameters

4.17.22

void **RPROP_init** (void)

initialize model for RPROP

initialize model for RPROP

4.17.23

Real **RPROP** (const Data& learndata) throw(Error)

one RPROP step; returns $\varepsilon = (y - \hat{y})^2$

one RPROP step; returns $\varepsilon = (y - \hat{y})^2$

4.17.24

void **RPROP_backstep** (void)

RPROP backstep: cancel last RPROP parameter update

RPROP backstep: cancel last RPROP parameter update

4.17.25

Real **optimize_RPROP** (const Data& a, const Data& b, size_t min_iterations, size_t max_iterations)

optimize model by cross validation; returns last error

optimize model by cross validation; returns last error

4.17.26

void **GRAD_DESCENT_init** (void)

initialize model for GRAD_DESCENT

initialize model for GRAD_DESCENT

4.17.27

Real **GRAD_DESCENT** (const Data& learndata) throw(Error)

one GRAD_DESCENT step; returns $\varepsilon = (y - \hat{y})^2$

one GRAD_DESCENT step; returns $\varepsilon = (y - \hat{y})^2$

4.17.28

void **GRAD_DESCENT_backstep** (void)

GRAD_DESCENT backstep: cancel last GRAD_DESCENT parameter update

GRAD_DESCENT backstep: cancel last GRAD_DESCENT parameter update

4.17.29

Real **optimize_GRAD_DESCENT** (const Data& a, const Data& b, size_t min_iterations, size_t max_iterations)

optimize model by cross validation; returns last error

optimize model by cross validation; returns last error

4.17.30

Real **R2** (const Data& d)

estimation error; returns R2

estimation error; returns R2

4.17.31

size_t **worst_rule_index** (const Data& d) throw(Error)

determine index of rule with biggest approximation error

determine index of rule with biggest approximation error

4.17.32

Real **estimation** (const Data& d) throw(Error)

estimation error; returns $\varepsilon = (y - \hat{y})^2$

estimation error; returns $\varepsilon = (y - \hat{y})^2$

4.17.33

Real **estimation** (const Data& d, const char* outfilename) throw(Error)

write estimation into a file; returns estimation error

write estimation into a file; returns estimation error

4.17.34

Real **simulation** (const Data& d) throw(Error)

simulation error; returns simulation error

simulation error; returns simulation error

4.17.35

Real **simulation** (const Data& d, const char* outfilename) throw(Error)

write simulation into a file; returns simulation error

write simulation into a file; returns simulation error

4.17.36

void **create_parameters** (Data* learning_data, Data* validation_data)

prepare FModel for Minimizer

prepare FModel for Minimizer

4.17.37

void **erase_parameters** ()

erase parameters for Minimizer

erase parameters for Minimizer

4.17.38

Real **calculate** ()

calculate function value (for learning/training)

calculate function value (for learning/training)

4.17.39

Real **validate** ()

calculate function value (for validation)

calculate function value (for validation)

4.17.40

void **derivate** ()

calculate derivation

calculate derivation

4.17.1

size_t **rdim_**

number of rules

number of rules

4.17.2

size_t **udim_**

input space dimension

input space dimension

4.17.3

size_t **cdim_**

consequence dimension

consequence dimension

4.17.4

size_t **sdim_**

*number of fsets = 2 * (rdim-1)*

number of fsets = 2 * (rdim-1)

4.17.5

size_t **worst_rule_**

index of rule with biggest error

index of rule with biggest error

4.17.6

vector<size_t> **history_**

history of refinements

history of refinements

4.17.7

FSetContainer **fsets_**

the model's Fuzzy Sets

the model's Fuzzy Sets

4.17.8

FRuleContainer **frules_**

the models Fuzzy Rules

the models Fuzzy Rules

4.17.9

string **learnfilename_**

name of the learning/training data file

name of the learning/training data file

4.17.10

string **validationfilename_**

name of the validation data file

name of the validation data file

4.17.11

vector<ConsParam> **cons_**

for the usage of a Minimizer

for the usage of a Minimizer

4.17.12

inline void **limit_delta_cons** (Consequence::iterator& p)

check border of Δ *c*

check border of Δ c

4.18

#define **DATA_HH**

we only have one function to minimize

we only have one function to minimize

4.19

typedef **Uvector**

Type of an input value vector u (u-part of one pattern).

4.20

class **Data**

Public Members

Protected Members

Data container.

4.20.8

Data ()

empty data

empty data

4.20.9

Data (Subscript N, size_t M, Real x0)

create data matrix filled with x0

create data matrix filled with x0

4.20.10

const string& **filename** () const

filename of loaded data

filename of loaded data

4.20.11

const vector<Uvector> & **U** () const

u-parts of all patterns

u-parts of all patterns

4.20.12

const vector<Real> & **y** () const

y-parts of all patterns

y-parts of all patterns

4.20.13

const vector<Real> & **scale_factor** () const

scale factor

scale factor

4.20.14

const vector<Real> & **scale_shift** () const

scale shift

scale shift

4.20.15

const size_t& **blocksize** () const

blocksize (for twodimensional print)

blocksize (for twodimensional print)

4.20.16

const Real& **mean_y** () const

mean of all y values

mean of all y values

4.20.17

const Real& **variance_y** () const

variance of all y values

variance of all y values

4.20.18

vector<Uvector> & **U** ()

modify u-parts of all patterns

modify u-parts of all patterns

4.20.19

vector<Real> & **y** ()

modify y-parts of all patterns

modify y-parts of all patterns

4.20.20

void **load** (char* filename) throw(Error)

load data matrix from file

load data matrix from file

4.20.21

void **normalize** (const vector<Real>& factor, const vector<Real>& shift) throw(Error)

normalize data matrix with given scale_factor and scale_shift

normalize data matrix with given scale_factor and scale_shift

4.20.1

size_t **blocksize_**

block size for twodimensional plots

block size for twodimensional plots

4.20.2

vector<Uvector> **U_**

u-parts of the patterns

u-parts of the patterns

4.20.3

vector<Real> **y_**

y-parts of the patterns

y-parts of the patterns

4.20.4

Real **mean_y_**

mean of all y values

mean of all y values

4.20.5

Real **variance_y_**

variance of all y values

variance of all y values

4.20.6

vector<Real> **scale_factor_**

scale factor Holds scale(u) in scale_factor_[0n_columns-1] and scale(y) in scale_factor_[n_columns].

4.20.7

vector<Real> **scale_shift_**

scale shift Holds shift(u) in shift_factor_[0n_columns-1] and shift(y) in shift_factor_[n_columns].

4.21

void **SVD** (const Matrix<Real>& A, Matrix<Real>& Q, Matrix<Real>& U, Matrix<Real>& V) throw(SVDError)

Compute the Singular Value Decomposition (SVD) of a m x n matrix A.

This function computes the Singular Value Decomposition (SVD) of a given matrix A. Precondition: m >= n. The SVD algorithm can be used to solve the least squares problem of overdetermined linear equation systems.

The algorithm is taken from: J.H. Wilkinson, C.Reinsch: Linear Algebra, Springer 1971, p. 134ff

The function computes the singular values Q and the complete orthogonal decomposition (U, V) of a real rectangular matrix A with A = U * Q * V.transp(); U.transp() * U = V.transp() * V = I.

With this, you can solve a linear least squares problem given by the m equations y = A * p with p = V * Q.inverse() * U.transp() * y where Q.inverse() is easy to compute, since Q is a diagonal matrix.

Parameters:

`A`	m x n input matrix to decompose; m rows, n columns. Precondition: m >= n.
`Q`	n x n diagonal matrix holding the singular values in its diagonal(return value).
`U`	m x n matrix with orthonormalized columns (return value).
`V`	n x n matrix with orthogonal columns (return value).

Exceptions:

`SVDException`	This exception will be thrown whenan error occurs during the computation.

Author: Steffen Bloedt, (minor changes by Manfred Männle)

Class Graph