Invariant Inference Framework

Generated by Doxygen 1.8.11

Contents

Chapter 1

Invariant Inference Framework:

This is the result of our implementation of the paper `An Invariant Inference Framework by Active Learning and SVMs` by Li Jiaying.

For you to run the experiments on your own machine, please follow the steps below to set up your experiment environment.

Work on Invariant Inference Framework

To build the framework currently is very easy, there is not much dependencies you need to satisfy before build the whole project.

Dependencies, for Windows/Linux/MacOSX Users:

- `cmake` version 2.8 or later.
- `libsvm` remember to put {libsvm}/bin folder into $PATH.
- `z3` For Windows users, please put z3 to the folder

```
1 C:/Program Files
```

- `klee` This is optional currently.
- [Build tools](), such as make, Visual Studio 2015, or Xcode.

###Build InvariantInferenceFramework

```
1 git clone git@github.com:lijiaying/InvariantInferenceFramework.git
2 cd InvariantInferenceFramework
3 cd test
4 mkdir build
5 cd build
6 cmake .. -G [your platform]  // just use cmake .. if you are not sure
7 make
```

Add your tests to this framework

As InvariantInferenceFramework is integrated with your examples, you need to do some modification on source code level before you can test your examples.

- READ carefully one example file in test folder before you write your own test.
- rewrite your loop code in a function with the name you like, my_loop_example for instance.

- modify function and function name as parameter for register_target which is called by main function.
- rename your test file with the number of parameters and a "_" as prefix.
- modify the second line in CMakeLists.txt in the project folder as the numbers of parameter you need in your program.
- After the above step, you can make your project and then run the executable file.

Experiments results:

- `simple2`
- `simple3`
- `ex1`
- `f1a`
- `f2`
- `substring1`

Chapter 2

Bug List

File color.h

unset_console_color is set the console back to black background, white forground, no strong comparision instead of the previous setting.

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Data Structure Documentation

6.1 Cache Class Reference

Data Structures

- struct head_t

Public Member Functions

- Cache (int l, long int size)
- ~Cache ()
- int get_data (const int index, Qfloat **data, int len)
- void swap_index (int i, int j)

Private Member Functions

- void lru_delete (head_t *h)
- void lru_insert (head_t *h)

Private Attributes

- int l
- long int size
- head_t * head
- head_t lru_head

6.1.1 Constructor & Destructor Documentation

6.1.1.1 Cache::Cache (int *l*, long int *size*)

6.1.1.2 Cache::~Cache ( )

6.1.2 Member Function Documentation

6.1.2.1 int Cache::get_data (const int *index*, Qfloat ** *data*, int *len*)

6.1.2.2 void Cache::lru_delete (head_t * *h*) [private]

6.1.2.3 void Cache::lru_insert (head_t * *h*) [private]

6.1.2.4 void Cache::swap_index (int *i*, int *j*)

6.1.3 Field Documentation

6.1.3.1 head_t* Cache::head [private]

6.1.3.2 int Cache::l [private]

6.1.3.3 head_t Cache::lru_head [private]

6.1.3.4 long int Cache::size [private]

The documentation for this class was generated from the following file:

- src/svm_core.cpp

6.2 decision_function Struct Reference

Data Fields

- double * alpha
- double rho

6.2.1 Field Documentation

6.2.1.1 double* decision_function::alpha

6.2.1.2 double decision_function::rho

The documentation for this struct was generated from the following file:

- src/svm_core.cpp

6.3 Equation Class Reference

```
#include <equation.h>
```

Public Member Functions

- Equation ()
- Equation (double a0,...)
- Equation (const Equation &equ)
- Equation & operator= (const Equation &rhs)
- bool imply (const Equation &e2)
- int linear_solver (Solution &sol)
- int is_similar (const Equation &e, int precision=PRECISION)
- int roundoff (Equation &e)

Static Public Member Functions

- static int linear_solver (const Equation *equ, Solution &sol)
- static double calc (Equation &equ, double *sol)

Data Fields

- double theta0
- double theta [VARS]

Friends

- std::ostream & operator<< (std::ostream &out, const Equation &equ)

6.3.1 Detailed Description

This class defines an equation we use in this project. Which is regards as a hyperplane in math. It stores all the coefficiency of the Equation.

$$ + [0] * x_0 + [1] * x_1 + ... + [VARS] * x_{VARS} >= 0$$

6.3.2 Constructor & Destructor Documentation

6.3.2.1 Equation::Equation ( )

Default constructor. Set all its elements to value 0

6.3.2.2 Equation::Equation (double *a0, ...*)

Most useful constructor Set its elements to the given values, order keeps The first element is Theta0

6.3.2.3 Equation::Equation (const Equation & *equ*)

Copy constructor. No comments.

6.3.3 Member Function Documentation

6.3.3.1 static double Equation::calc (Equation & *equ,* double * *sol*) [inline], [static]

This method is used to get the position info for the given point against given equation It just substitute variants with the given point.

6.3.3.2 bool Equation::imply (const Equation & *e2*)

imply method checks whether one equation can imply another one or not *this is default equation left side

Parameters

e2	is the equation right side we check whether *this ==> e2 ??

Returns

true if yes, false if no. Currently, it is based on Z3 prover. And the default precision is set to E-8 (2.8f), which is changeable if need

6.3.3.3 int Equation::is_similar (const Equation & *e,* int *precision* = PRECISION)

This method is used to check whether ∗this equation is similar to given equation e or not ∗this ∼= e ???

Parameters

precision	defines how much variance we can bare. The default is 4, which means we can bare 0.0001 difference. In this case 1 ∼=1.00001, but 1!∼=1.000011

6.3.3.4 int Equation::linear_solver (Solution & *sol*) [inline]

A shell on linear_solver(equ, sol) More understandable

6.3.3.5 static int Equation::linear_solver (const Equation ∗ *equ,* Solution & *sol*) [inline],[static]

The real solver for an Equation This method calcuate the most informative points in space It return a points really on the margin or next to the margin

Parameters

sol	contains the solution, integer format

equ == NULL means no equation is specified So we randomly generate points in given scope [minv, maxv]

justify whether all the coefficients are zeros...

< pick store the dimension that should not generate randomly

< The algo is we generate numbers randomly, unless the picked dimension The picked dimension should be calcuate based on equation and other dimensions

sometimes we can not get solution between given scope we try 10 times, if still no suitable solution, we pick the last one...

6.3.3.6 Equation & Equation::operator= (const Equation & *rhs*)

Overwrite = operator This is needed when we want to delete a equation in an equation list We copy the next equation to the current one, and repeat this process until tails

6.3.3.7 int Equation::roundoff (Equation & *e*)

sometimes the equation has ugly coefficiencies we want to make it elegent, which is the purpose of involing this method Currently we have not done much work on this We have not even use gcd function to adjust the coefficients. For example. 1.2345 x1 >= 2.4690 ==> x1 >= 2 2 x1 >= 5.000001 ==> x1 >= 2.5

6.3.4 Friends And Related Function Documentation

6.3.4.1 std::ostream& operator<< (std::ostream & *out,* const Equation & *equ*) [friend]

Output the equation in a readable format Example: 2{0} + 3{1} >= 5

6.3.5 Field Documentation

6.3.5.1 double Equation::theta[VARS]

6.3.5.2 double Equation::theta0

The documentation for this class was generated from the following files:

- include/equation.h
- src/equation.cpp

6.4 Cache::head_t Struct Reference

Data Fields

- head_t * prev
- head_t * next
- Qfloat * data
- int len

6.4.1 Field Documentation

6.4.1.1 Qfloat* Cache::head_t::data

6.4.1.2 int Cache::head_t::len

6.4.1.3 head_t * Cache::head_t::next

6.4.1.4 head_t* Cache::head_t::prev

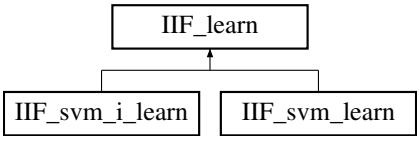
The documentation for this struct was generated from the following file:

- src/svm_core.cpp

6.5 IIF_learn Class Reference

`#include <iif_learn.h>`

Inheritance diagram for IIF_learn:



Public Member Functions

- IIF_learn (States *gsets, int(*func)(int *))
- IIF_learn ()
- void run_target (Solution &input)
- virtual int learn ()=0

Protected Member Functions

- void init_gsets ()

Protected Attributes

- States * gsets
- int(* func)(int *)

6.5.1 Constructor & Destructor Documentation

6.5.1.1 IIF_learn::IIF_learn (States * *gsets*, int(*)(int *) *func*) [inline]

6.5.1.2 IIF_learn::IIF_learn ( ) [inline]

6.5.2 Member Function Documentation

6.5.2.1 void IIF_learn::init_gsets ( ) [inline], [protected]

6.5.2.2 virtual int IIF_learn::learn ( ) [pure virtual]

Implemented in IIF_svm_i_learn, and IIF_svm_learn.

6.5.2.3 void IIF_learn::run_target (Solution & *input*) [inline]

6.5.3 Field Documentation

6.5.3.1 int(* IIF_learn::func) (int *) [protected]

6.5.3.2 States* IIF_learn::gsets [protected]

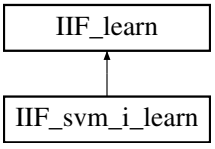
The documentation for this class was generated from the following file:

- include/iif_learn.h

6.6 IIF_svm_i_learn Class Reference

`#include <iif_svm_i_learn.h>`

Inheritance diagram for IIF_svm_i_learn:



Public Member Functions

- IIF_svm_i_learn (States *gsets, int(*func)(int *), int max_iteration=max_iter)
- IIF_svm_i_learn ()
- virtual int learn ()

Protected Attributes

- SVM_I * svm_i
- int max_iteration

Additional Inherited Members

6.6.1 Constructor & Destructor Documentation

6.6.1.1 IIF_svm_i_learn::IIF_svm_i_learn (States * *gsets,* int(*)(int *) *func,* int *max_iteration* = max_iter)

6.6.1.2 IIF_svm_i_learn::IIF_svm_i_learn ( )

6.6.2 Member Function Documentation

6.6.2.1 int IIF_svm_i_learn::learn ( ) [virtual]

Implements IIF_learn.

6.6.3 Field Documentation

6.6.3.1 int IIF_svm_i_learn::max_iteration [protected]

6.6.3.2 SVM_I* IIF_svm_i_learn::svm_i [protected]

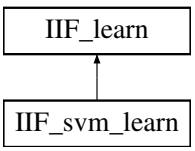
The documentation for this class was generated from the following files:

- include/iif_svm_i_learn.h
- src/iif_svm_i_learn.cpp

6.7 IIF_svm_learn Class Reference

```
#include <iif_svm_learn.h>
```

Inheritance diagram for IIF_svm_learn:



Public Member Functions

- IIF_svm_learn (States *gsets, int(*func)(int *), int max_iteration=max_iter)
- IIF_svm_learn ()
- virtual int learn ()

Protected Attributes

- SVM ∗ svm
- int max_iteration

Additional Inherited Members

6.7.1 Constructor & Destructor Documentation

6.7.1.1 IIF_svm_learn::IIF_svm_learn (States ∗ *gsets*, int(∗)(int ∗) *func*, int *max_iteration* = max_iter)

6.7.1.2 IIF_svm_learn::IIF_svm_learn ( )

6.7.2 Member Function Documentation

6.7.2.1 int IIF_svm_learn::learn ( ) [virtual]

Implements IIF_learn.

6.7.3 Field Documentation

6.7.3.1 int IIF_svm_learn::max_iteration [protected]

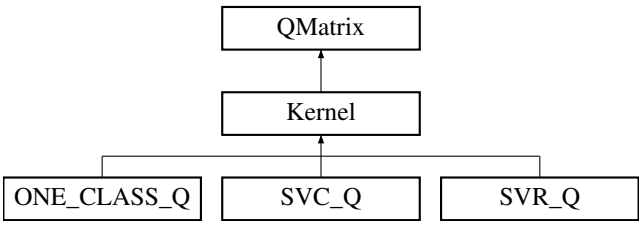
6.7.3.2 SVM∗ IIF_svm_learn::svm [protected]

The documentation for this class was generated from the following files:

- include/iif_svm_learn.h
- src/iif_svm_learn.cpp

6.8 Kernel Class Reference

Inheritance diagram for Kernel:



Public Member Functions

- Kernel (int l, svm_node ∗const ∗x, const svm_parameter &param)
- virtual ∼Kernel ()
- virtual Qfloat ∗ get_Q (int column, int len) const =0
- virtual double ∗ get_QD () const =0
- virtual void swap_index (int i, int j) const

Static Public Member Functions

- static double k_function (const svm_node *x, const svm_node *y, const svm_parameter &param)

Protected Attributes

- double(Kernel::* kernel_function)(int i, int j) const

Private Member Functions

- double kernel_linear (int i, int j) const
- double kernel_poly (int i, int j) const
- double kernel_rbf (int i, int j) const
- double kernel_sigmoid (int i, int j) const
- double kernel_precomputed (int i, int j) const

Static Private Member Functions

- static double dot (const svm_node *px, const svm_node *py)

Private Attributes

- const svm_node ** x
- double * x_square
- const int kernel_type
- const int degree
- const double gamma
- const double coef0

6.8.1 Constructor & Destructor Documentation

6.8.1.1 Kernel::Kernel (int *l*, svm_node *const * *x*, const svm_parameter & *param*)

6.8.1.2 Kernel::~Kernel ( ) [virtual]

6.8.2 Member Function Documentation

6.8.2.1 double Kernel::dot (const svm_node * *px*, const svm_node * *py*) [static], [private]

6.8.2.2 virtual Qfloat* Kernel::get_Q (int *column*, int *len*) const [pure virtual]

Implements QMatrix.

Implemented in SVR_Q, ONE_CLASS_Q, and SVC_Q.

6.8.2.3 virtual double* Kernel::get_QD ( ) const [pure virtual]

Implements QMatrix.

Implemented in SVR_Q, ONE_CLASS_Q, and SVC_Q.

6.8.2.4 double Kernel::k_function (const svm_node * *x*, const svm_node * *y*, const svm_parameter & *param*) [static]

6.8.2.5 double Kernel::kernel_linear (int *i*, int *j*) const [inline], [private]

6.8.2.6 double Kernel::kernel_poly (int *i*, int *j*) const [inline], [private]

6.8.2.7 double Kernel::kernel_precomputed (int *i*, int *j*) const [inline], [private]

6.8.2.8 double Kernel::kernel_rbf (int *i*, int *j*) const [inline], [private]

6.8.2.9 double Kernel::kernel_sigmoid (int *i*, int *j*) const [inline], [private]

6.8.2.10 virtual void Kernel::swap_index (int *i*, int *j*) const [inline], [virtual]

Implements QMatrix.

Reimplemented in SVR_Q, ONE_CLASS_Q, and SVC_Q.

6.8.3 Field Documentation

6.8.3.1 const double Kernel::coef0 [private]

6.8.3.2 const int Kernel::degree [private]

6.8.3.3 const double Kernel::gamma [private]

6.8.3.4 double(Kernel::* Kernel::kernel_function) (int i, int j) const [protected]

6.8.3.5 const int Kernel::kernel_type [private]

6.8.3.6 const svm_node** Kernel::x [private]

6.8.3.7 double* Kernel::x_square [private]

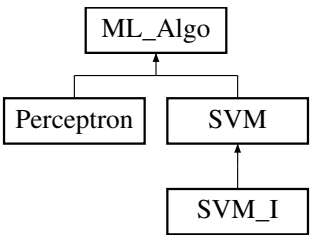
The documentation for this class was generated from the following file:

- src/svm_core.cpp

6.9 ML_Algo Class Reference

`#include <ml_algo.h>`

Inheritance diagram for ML_Algo:



Public Member Functions

- ML_Algo ()
- virtual int prepare_training_data (States *, int &, int &)=0
- virtual int train ()=0
- virtual double predict_on_training_set ()=0
- virtual int check_question_set (States &)=0
- virtual int get_converged (Equation *, int)=0
- virtual std::ostream & _print (std::ostream &out) const
- virtual int size ()=0
- virtual Equation * roundoff (int &)=0
- virtual int predict (double *, int)=0

Friends

- std::ostream & operator<< (std::ostream &out, const ML_Algo &mla)

6.9.1 Constructor & Destructor Documentation

6.9.1.1 ML_Algo::ML_Algo ( ) [inline]

6.9.2 Member Function Documentation

6.9.2.1 virtual std::ostream& ML_Algo::_print (std::ostream & *out*) const [inline], [virtual]

Reimplemented in SVM_I, and SVM.

6.9.2.2 virtual int ML_Algo::check_question_set (States &) [pure virtual]

Implemented in SVM_I, SVM, and Perceptron.

6.9.2.3 virtual int ML_Algo::get_converged (Equation *, int) [pure virtual]

Implemented in SVM_I, and SVM.

6.9.2.4 virtual int ML_Algo::predict (double *, int) [pure virtual]

Implemented in SVM_I, SVM, and Perceptron.

6.9.2.5 virtual double ML_Algo::predict_on_training_set ( ) [pure virtual]

Implemented in SVM_I, SVM, and Perceptron.

6.9.2.6 virtual int ML_Algo::prepare_training_data (States *, int & , int &) [pure virtual]

Implemented in SVM_I, SVM, and Perceptron.

6.9.2.7 virtual Equation* ML_Algo::roundoff (int &) [pure virtual]

Implemented in SVM_I, SVM, and Perceptron.

6.9.2.8 virtual int ML_Algo::size ( ) [pure virtual]

Implemented in SVM_I, SVM, and Perceptron.

6.9.2.9 virtual int ML_Algo::train ( ) [pure virtual]

Implemented in SVM_I, SVM, and Perceptron.

6.9.3 Friends And Related Function Documentation

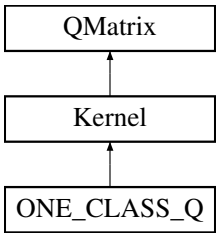
6.9.3.1 std::ostream& operator<< (std::ostream & *out,* const ML_Algo & *mla*) [friend]

The documentation for this class was generated from the following file:

- include/ml_algo.h

6.10 ONE_CLASS_Q Class Reference

Inheritance diagram for ONE_CLASS_Q:



Public Member Functions

- ONE_CLASS_Q (const svm_problem &prob, const svm_parameter &param)
- Qfloat * get_Q (int i, int len) const
- double * get_QD () const
- void swap_index (int i, int j) const
- ~ONE_CLASS_Q ()

Private Attributes

- Cache * cache
- double * QD

Additional Inherited Members

6.10.1 Constructor & Destructor Documentation

6.10.1.1 ONE_CLASS_Q::ONE_CLASS_Q (const svm_problem & *prob,* const svm_parameter & *param*) [inline]

6.10.1.2 ONE_CLASS_Q::~ONE_CLASS_Q ( ) [inline]

6.10.2 Member Function Documentation

6.10.2.1 Qfloat* ONE_CLASS_Q::get_Q (int *i,* int *len*) const [inline], [virtual]

Implements Kernel.

6.10.2.2 double* ONE_CLASS_Q::get_QD ( ) const [inline], [virtual]

Implements Kernel.

6.10.2.3 void ONE_CLASS_Q::swap_index (int *i,* int *j*) const [inline], [virtual]

Reimplemented from Kernel.

6.10.3 Field Documentation

6.10.3.1 Cache* ONE_CLASS_Q::cache [private]

6.10.3.2 double* ONE_CLASS_Q::QD [private]

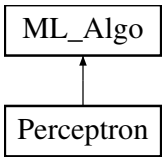
The documentation for this class was generated from the following file:

- src/svm_core.cpp

6.11 Perceptron Class Reference

`#include <perceptron.h>`

Inheritance diagram for Perceptron:



Public Member Functions

- Perceptron (void(*f)(const char *)=NULL)
- virtual ~Perceptron ()
- virtual int prepare_training_data (States *gsets, int &pre_positive_size, int &pre_negative_size)
- virtual int train ()
- virtual double predict_on_training_set ()
- virtual int check_question_set (States &qset)
- virtual int size ()
- virtual Equation * roundoff (int &num)
- virtual int predict (double *v, int label=0)

Data Fields

- Equation * main_equation
- double training_label [max_items *2]
- double * training_set [max_items *2]
- int length

Private Member Functions

- Equation * perceptron_train ()

Friends

- std::ostream & operator<< (std::ostream &out, const Perceptron &)

6.11.1 Constructor & Destructor Documentation

6.11.1.1 Perceptron::Perceptron (void(*)(const char *) *f* = NULL)

6.11.1.2 Perceptron::~Perceptron ( ) [virtual]

6.11.2 Member Function Documentation

6.11.2.1 int Perceptron::check_question_set (States & *qset*) [virtual]

Implements ML_Algo.

6.11.2.2 Equation * Perceptron::perceptron_train ( ) [private]

6.11.2.3 int Perceptron::predict (double * *v*, int *label* = 0) [virtual]

Implements ML_Algo.

6.11.2.4 double Perceptron::predict_on_training_set ( ) [virtual]

Implements ML_Algo.

6.11.2.5 int Perceptron::prepare_training_data (States * *gsets*, int & *pre_positive_size*, int & *pre_negative_size*) [virtual]

Implements ML_Algo.

6.11.2.6 Equation * Perceptron::roundoff (int & *num*) [virtual]

Implements ML_Algo.

6.11.2.7 int Perceptron::size ( ) [virtual]

Implements ML_Algo.

6.11.2.8 int Perceptron::train ( ) [virtual]

Implements ML_Algo.

6.11.3 Friends And Related Function Documentation

6.11.3.1 std::ostream& operator<< (std::ostream & *out,* const Perceptron & *perceptron*) [friend]

6.11.4 Field Documentation

6.11.4.1 int Perceptron::length

6.11.4.2 Equation* Perceptron::main_equation

6.11.4.3 double Perceptron::training_label[max_items *2]

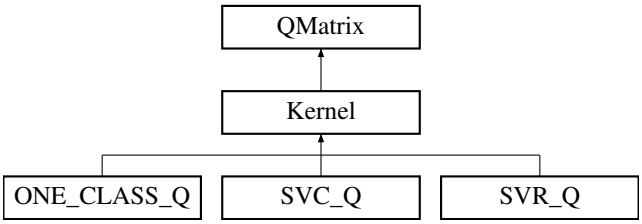
6.11.4.4 double* Perceptron::training_set[max_items *2]

The documentation for this class was generated from the following files:

- include/perceptron.h
- src/perceptron.cpp

6.12 QMatrix Class Reference

Inheritance diagram for QMatrix:



Public Member Functions

- virtual Qfloat * get_Q (int column, int len) const =0
- virtual double * get_QD () const =0
- virtual void swap_index (int i, int j) const =0
- virtual ~QMatrix ()

6.12.1 Constructor & Destructor Documentation

6.12.1.1 virtual QMatrix::~QMatrix ( ) [inline], [virtual]

6.12.2 Member Function Documentation

6.12.2.1 virtual Qfloat* QMatrix::get_Q (int *column,* int *len*) const [pure virtual]

Implemented in SVR_Q, ONE_CLASS_Q, SVC_Q, and Kernel.

6.12.2.2 virtual double* QMatrix::get_QD ( ) const [pure virtual]

Implemented in SVR_Q, ONE_CLASS_Q, SVC_Q, and Kernel.

6.12.2.3 virtual void QMatrix::swap_index (int *i,* int *j*) const [pure virtual]

Implemented in SVR_Q, ONE_CLASS_Q, SVC_Q, and Kernel.

The documentation for this class was generated from the following file:

- src/svm_core.cpp

6.13 Solution Class Reference

`#include <equation.h>`

Public Member Functions

- Solution ()
- Solution (double a0,...)

Data Fields

- double x [VARS]

Friends

- std::ostream & operator<< (std::ostream &out, const Solution &sol)

6.13.1 Constructor & Destructor Documentation

6.13.1.1 Solution::Solution ( )

Default constructor. Set all its elements to value 0

6.13.1.2 Solution::Solution (double *a0,* ...)

Most useful constructor Set its elements to the given values, order keeps

6.13.2 Friends And Related Function Documentation

6.13.2.1 std::ostream& operator<< (std::ostream & *out,* const Solution & *sol*) [friend]

support << operator simply output its elements as a tuple

6.13.3 Field Documentation

6.13.3.1 double Solution::x[VARS]

The data field of Solution stores all the values as a solution to an Equation

The documentation for this class was generated from the following files:

- include/equation.h
- src/equation.cpp

6.14 Solver::SolutionInfo Struct Reference

Data Fields

- double obj
- double rho
- double upper_bound_p
- double upper_bound_n
- double r

6.14.1 Field Documentation

6.14.1.1 double Solver::SolutionInfo::obj

6.14.1.2 double Solver::SolutionInfo::r

6.14.1.3 double Solver::SolutionInfo::rho

6.14.1.4 double Solver::SolutionInfo::upper_bound_n

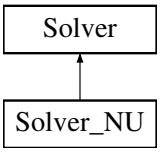
6.14.1.5 double Solver::SolutionInfo::upper_bound_p

The documentation for this struct was generated from the following file:

- src/svm_core.cpp

6.15 Solver Class Reference

Inheritance diagram for Solver:



Data Structures

- struct SolutionInfo

Public Member Functions

- Solver ()
- virtual ~Solver ()
- void Solve (int l, const QMatrix &Q, const double *p_, const schar *y_, double *alpha_, double Cp, double Cn, double eps, SolutionInfo *si, int shrinking)

Protected Types

- enum { LOWER_BOUND, UPPER_BOUND, FREE }

Protected Member Functions

- double get_C (int i)
- void update_alpha_status (int i)
- bool is_upper_bound (int i)
- bool is_lower_bound (int i)
- bool is_free (int i)
- void swap_index (int i, int j)
- void reconstruct_gradient ()
- virtual int select_working_set (int &i, int &j)
- virtual double calculate_rho ()
- virtual void do_shrinking ()

Protected Attributes

- int active_size
- schar * y
- double * G
- char * alpha_status
- double * alpha
- const QMatrix * Q
- const double * QD
- double eps
- double Cp
- double Cn
- double * p
- int * active_set
- double * G_bar
- int l
- bool unshrink

Private Member Functions

- bool be_shrunk (int i, double Gmax1, double Gmax2)

6.15.1 Member Enumeration Documentation

6.15.1.1 anonymous enum [protected]

Enumerator

LOWER_BOUND

UPPER_BOUND

FREE

6.15.2 Constructor & Destructor Documentation

6.15.2.1 Solver::Solver ( ) [inline]

6.15.2.2 virtual Solver::~Solver ( ) [inline], [virtual]

6.15.3 Member Function Documentation

6.15.3.1 bool Solver::be_shrunk (int *i*, double *Gmax1*, double *Gmax2*) [private]

6.15.3.2 double Solver::calculate_rho ( ) [protected], [virtual]

Reimplemented in Solver_NU.

6.15.3.3 void Solver::do_shrinking ( ) [protected], [virtual]

Reimplemented in Solver_NU.

6.15.3.4 double Solver::get_C (int *i*) [inline], [protected]

6.15.3.5 bool Solver::is_free (int *i*) [inline], [protected]

6.15.3.6 bool Solver::is_lower_bound (int *i*) [inline], [protected]

6.15.3.7 bool Solver::is_upper_bound (int *i*) [inline], [protected]

6.15.3.8 void Solver::reconstruct_gradient ( ) [protected]

6.15.3.9 int Solver::select_working_set (int & *i*, int & *j*) [protected], [virtual]

Reimplemented in Solver_NU.

6.15.3.10 void Solver::Solve (int *l*, const QMatrix & *Q*, const double * *p_*, const schar * *y_*, double * *alpha_*, double *Cp*, double *Cn*, double *eps*, SolutionInfo * *si*, int *shrinking*)

6.15.3.11 void Solver::swap_index (int *i*, int *j*) [protected]

6.15.3.12 void Solver::update_alpha_status (int *i*) [inline], [protected]

6.15.4 Field Documentation

6.15.4.1 int* Solver::active_set [protected]

6.15.4.2 int Solver::active_size [protected]

6.15.4.3 double* Solver::alpha [protected]

6.15.4.4 char* Solver::alpha_status [protected]

6.15.4.5 double Solver::Cn [protected]

6.15.4.6 double Solver::Cp [protected]

6.15.4.7 double Solver::eps [protected]

6.15.4.8 double* Solver::G [protected]

6.15.4.9 double* Solver::G_bar [protected]

6.15.4.10 int Solver::l [protected]

6.15.4.11 double* Solver::p [protected]

6.15.4.12 const QMatrix* Solver::Q [protected]

6.15.4.13 const double* Solver::QD [protected]

6.15.4.14 bool Solver::unshrink [protected]

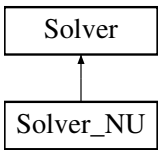
6.15.4.15 schar* Solver::y [protected]

The documentation for this class was generated from the following file:

- src/svm_core.cpp

6.16 Solver_NU Class Reference

Inheritance diagram for Solver_NU:



Public Member Functions

- Solver_NU ()
- void Solve (int l, const QMatrix &Q, const double *p, const schar *y, double *alpha, double Cp, double Cn, double eps, SolutionInfo *si, int shrinking)

Private Member Functions

- int select_working_set (int &i, int &j)
- double calculate_rho ()
- bool be_shrunk (int i, double Gmax1, double Gmax2, double Gmax3, double Gmax4)
- void do_shrinking ()

Private Attributes

- SolutionInfo * si

Additional Inherited Members

6.16.1 Constructor & Destructor Documentation

6.16.1.1 Solver_NU::Solver_NU ( ) [inline]

6.16.2 Member Function Documentation

6.16.2.1 bool Solver_NU::be_shrunk (int *i*, double *Gmax1*, double *Gmax2*, double *Gmax3*, double *Gmax4*) [private]

6.16.2.2 double Solver_NU::calculate_rho ( ) [private], [virtual]

Reimplemented from Solver.

6.16.2.3 void Solver_NU::do_shrinking ( ) [private], [virtual]

Reimplemented from Solver.

6.16.2.4 int Solver_NU::select_working_set (int & *i*, int & *j*) [private], [virtual]

Reimplemented from Solver.

6.16.2.5 void Solver_NU::Solve (int *l*, const QMatrix & *Q*, const double * *p*, const schar * *y*, double * *alpha*, double *Cp*, double *Cn*, double *eps*, SolutionInfo * *si*, int *shrinking*) [inline]

6.16.3 Field Documentation

6.16.3.1 SolutionInfo* Solver_NU::si [private]

The documentation for this class was generated from the following file:

- src/svm_core.cpp

6.17 States Class Reference

`#include <states.h>`

Public Member Functions

- States ()
- ~States ()
- int add_states (double st[ ][VARS], int len)
- int traces_num ()
- int size ()
- void print_trace (int num)

Data Fields

- double(* values)[VARS]
- int * index
- int p_index
- int label

Private Attributes

- int max_size

Friends

- std::ostream & operator<< (std::ostream &out, const States &ss)

6.17.1 Constructor & Destructor Documentation

6.17.1.1 States::States ( )

6.17.1.2 States::~States ( )

6.17.2 Member Function Documentation

6.17.2.1 int States::add_states (double *st[ ][VARS],* int *len*)

6.17.2.2 void States::print_trace (int *num*)

6.17.2.3 int States::size ( )

6.17.2.4 int States::traces_num ( )

6.17.3 Friends And Related Function Documentation

6.17.3.1 std::ostream& operator<< (std::ostream & *out,* const States & *ss*) [friend]

6.17.4 Field Documentation

6.17.4.1 int* States::index

6.17.4.2 int States::label

6.17.4.3 int States::max_size [private]

6.17.4.4 int States::p_index

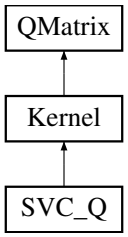
6.17.4.5 double(* States::values)[VARS]

The documentation for this class was generated from the following files:

- include/states.h
- src/states.cpp

6.18 SVC_Q Class Reference

Inheritance diagram for SVC_Q:



Public Member Functions

- SVC_Q (const svm_problem &prob, const svm_parameter &param, const schar *y_)
- Qfloat * get_Q (int i, int len) const

- double * get_QD () const
- void swap_index (int i, int j) const
- ~SVC_Q ()

Private Attributes

- schar * y
- Cache * cache
- double * QD

Additional Inherited Members

6.18.1 Constructor & Destructor Documentation

6.18.1.1 SVC_Q::SVC_Q (const svm_problem & *prob*, const svm_parameter & *param*, const schar * *y_*) [inline]

6.18.1.2 SVC_Q::~SVC_Q ( ) [inline]

6.18.2 Member Function Documentation

6.18.2.1 Qfloat* SVC_Q::get_Q (int *i*, int *len*) const [inline], [virtual]

Implements Kernel.

6.18.2.2 double* SVC_Q::get_QD ( ) const [inline], [virtual]

Implements Kernel.

6.18.2.3 void SVC_Q::swap_index (int *i*, int *j*) const [inline], [virtual]

Reimplemented from Kernel.

6.18.3 Field Documentation

6.18.3.1 Cache* SVC_Q::cache [private]

6.18.3.2 double* SVC_Q::QD [private]

6.18.3.3 schar* SVC_Q::y [private]

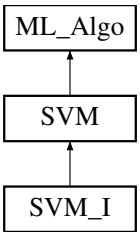
The documentation for this class was generated from the following file:

- src/svm_core.cpp

6.19 SVM Class Reference

#include <svm.h>

Inheritance diagram for SVM:



Public Member Functions

- SVM (void(∗f)(const char ∗)=NULL, int size=10000)
- virtual ∼SVM ()
- virtual int prepare_training_data (States ∗gsets, int &pre_positive_size, int &pre_negative_size)
- virtual int train ()
- virtual double predict_on_training_set ()
- virtual int check_question_set (States &qset)
- virtual int get_converged (Equation ∗, int)
- virtual std::ostream & _print (std::ostream &out) const
- virtual int size ()
- virtual Equation ∗ roundoff (int &num)
- virtual int predict (double ∗v, int label=0)

Data Fields

- svm_model ∗ model
- Equation ∗ main_equation
- svm_parameter param
- svm_problem problem
- double ∗ training_label
- double ∗∗ training_set

Protected Attributes

- int max_size

Friends

- std::ostream & operator<< (std::ostream &out, const SVM &svm)

6.19.1 Constructor & Destructor Documentation

6.19.1.1 SVM::SVM (void(∗)(const char ∗) *f* = NULL, int *size* = 10000)

6.19.1.2 SVM::∼SVM ( ) [virtual]

6.19.2 Member Function Documentation

6.19.2.1 std::ostream & SVM::_print (std::ostream & *out*) const [virtual]

Reimplemented from ML_Algo.

Reimplemented in SVM_I.

6.19.2.2 int SVM::check_question_set (States & *qset*) [virtual]

Implements ML_Algo.

Reimplemented in SVM_I.

6.19.2.3 int SVM::get_converged (Equation * *last_equation*, int *equation_num*) [virtual]

Implements ML_Algo.

Reimplemented in SVM_I.

6.19.2.4 int SVM::predict (double * *v*, int *label* = 0) [virtual]

Implements ML_Algo.

Reimplemented in SVM_I.

6.19.2.5 double SVM::predict_on_training_set ( ) [virtual]

Implements ML_Algo.

Reimplemented in SVM_I.

6.19.2.6 int SVM::prepare_training_data (States * *gsets*, int & *pre_positive_size*, int & *pre_negative_size*) [virtual]

Implements ML_Algo.

Reimplemented in SVM_I.

6.19.2.7 Equation * SVM::roundoff (int & *num*) [virtual]

Implements ML_Algo.

Reimplemented in SVM_I.

6.19.2.8 int SVM::size ( ) [virtual]

Implements ML_Algo.

Reimplemented in SVM_I.

6.19.2.9 int SVM::train ( ) [virtual]

Implements ML_Algo.

Reimplemented in SVM_I.

6.19.3 Friends And Related Function Documentation

6.19.3.1 std::ostream& operator<< (std::ostream & *out*, const SVM & *svm*) [friend]

6.19.4 Field Documentation

6.19.4.1 Equation* SVM::main_equation

6.19.4.2 int SVM::max_size [protected]

6.19.4.3 svm_model* SVM::model

6.19.4.4 svm_parameter SVM::param

6.19.4.5 svm_problem SVM::problem

6.19.4.6 double* SVM::training_label

6.19.4.7 double** SVM::training_set

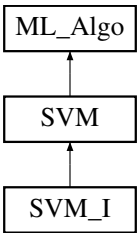
The documentation for this class was generated from the following files:

- include/svm.h
- src/svm.cpp

6.20 SVM_I Class Reference

`#include <svm_i.h>`

Inheritance diagram for SVM_I:



Public Member Functions

- SVM_I (void(*f)(const char *)=NULL, int size=10000, int equ=16)
- ~SVM_I ()
- virtual int prepare_training_data (States *gsets, int &pre_positive_size, int &pre_negative_size)
- int train ()
- double predict_on_training_set ()
- virtual int check_question_set (States &qset)
- virtual int get_converged (Equation *, int)
- virtual std::ostream & _print (std::ostream &out) const
- int size ()
- virtual Equation * roundoff (int &num)
- virtual int predict (double *v, int label=0)

Data Fields

- svm_model * model
- Equation * equations
- int equ_num
- svm_parameter param
- States * negatives

Protected Attributes

- int max_equ

Private Member Functions

- double check_postives_and_one_negative ()
- int get_misclassified (int &idx)

Friends

- std::ostream & operator<< (std::ostream &out, const SVM_I &svm_i)

6.20.1 Constructor & Destructor Documentation

6.20.1.1 SVM_I::SVM_I (void(*)(const char *) *f* = NULL, int *size* = 10000, int *equ* = 16)

6.20.1.2 SVM_I::~SVM_I ( )

6.20.2 Member Function Documentation

6.20.2.1 std::ostream & SVM_I::_print (std::ostream & *out*) const [virtual]

Reimplemented from SVM.

6.20.2.2 double SVM_I::check_postives_and_one_negative ( ) [private]

6.20.2.3 int SVM_I::check_question_set (States & *qset*) [virtual]

Reimplemented from SVM.

6.20.2.4 int SVM_I::get_converged (Equation * *previous_equations*, int *previous_equation_num*) [virtual]

Reimplemented from SVM.

6.20.2.5 int SVM_I::get_misclassified (int & *idx*) [private]

6.20.2.6 int SVM_I::predict (double * *v*, int *label* = 0) [virtual]

Reimplemented from SVM.

6.20.2.7 double SVM_I::predict_on_training_set ( ) [virtual]

Reimplemented from SVM.

6.20.2.8 int SVM_I::prepare_training_data (States * *gsets*, int & *pre_positive_size*, int & *pre_negative_size*) [virtual]

Reimplemented from SVM.

6.20.2.9 Equation * SVM_I::roundoff (int & *num*) [virtual]

Reimplemented from SVM.

6.20.2.10 int SVM_I::size ( ) [virtual]

Reimplemented from SVM.

6.20.2.11 int SVM_I::train ( ) [virtual]

Reimplemented from SVM.

6.20.3 Friends And Related Function Documentation

6.20.3.1 std::ostream& operator<< (std::ostream & *out,* const SVM_I & *svm_i*) [friend]

6.20.4 Field Documentation

6.20.4.1 int SVM_I::equ_num

6.20.4.2 Equation* SVM_I::equations

6.20.4.3 int SVM_I::max_equ [protected]

6.20.4.4 svm_model* SVM_I::model

6.20.4.5 States* SVM_I::negatives

6.20.4.6 svm_parameter SVM_I::param

The documentation for this class was generated from the following files:

- include/svm_i.h
- src/svm_i.cpp

6.21 svm_model Struct Reference

`#include <svm_core.h>`

Data Fields

- struct svm_parameter param
- int nr_class
- int l
- struct svm_node ** SV
- double ** sv_coef
- double * rho
- double * probA
- double * probB
- int * sv_indices
- int * label
- int * nSV
- int free_sv

6.21.1 Field Documentation

6.21.1.1 int svm_model::free_sv

6.21.1.2 int svm_model::l

6.21.1.3 int* svm_model::label

6.21.1.4 int svm_model::nr_class

6.21.1.5 int* svm_model::nSV

6.21.1.6 struct svm_parameter svm_model::param

6.21.1.7 double* svm_model::probA

6.21.1.8 double* svm_model::probB

6.21.1.9 double* svm_model::rho

6.21.1.10 struct svm_node** svm_model::SV

6.21.1.11 double** svm_model::sv_coef

6.21.1.12 int* svm_model::sv_indices

The documentation for this struct was generated from the following file:

- include/svm_core.h

6.22 svm_node Struct Reference

`#include <svm_core.h>`

Data Fields

- double value

Friends

- std::ostream & operator<< (std::ostream &out, const svm_node &sn)

6.22.1 Friends And Related Function Documentation

6.22.1.1 std::ostream& operator<< (std::ostream & *out,* const svm_node & *sn*) [friend]

6.22.2 Field Documentation

6.22.2.1 double svm_node::value

The documentation for this struct was generated from the following file:

- include/svm_core.h

6.23 svm_parameter Struct Reference

`#include <svm_core.h>`

Data Fields

- int svm_type
- int kernel_type
- int degree
- double gamma
- double coef0
- double cache_size
- double eps
- double C
- int nr_weight
- int * weight_label
- double * weight
- double nu
- double p
- int shrinking
- int probability

6.23.1 Field Documentation

6.23.1.1 double svm_parameter::C

6.23.1.2 double svm_parameter::cache_size

6.23.1.3 double svm_parameter::coef0

6.23.1.4 int svm_parameter::degree

6.23.1.5 double svm_parameter::eps

6.23.1.6 double svm_parameter::gamma

6.23.1.7 int svm_parameter::kernel_type

6.23.1.8 int svm_parameter::nr_weight

6.23.1.9 double svm_parameter::nu

6.23.1.10 double svm_parameter::p

6.23.1.11 int svm_parameter::probability

6.23.1.12 int svm_parameter::shrinking

6.23.1.13 int svm_parameter::svm_type

6.23.1.14 double* svm_parameter::weight

6.23.1.15 int* svm_parameter::weight_label

The documentation for this struct was generated from the following file:

- include/svm_core.h

6.24 svm_problem Struct Reference

`#include <svm_core.h>`

Data Fields

- int l
- double * y
- struct svm_node ** x

Friends

- std::ostream & operator<< (std::ostream &out, const svm_problem &sp)

6.24.1 Friends And Related Function Documentation

6.24.1.1 std::ostream& operator<< (std::ostream & *out,* const svm_problem & *sp*) [friend]

6.24.2 Field Documentation

6.24.2.1 int svm_problem::l

6.24.2.2 struct svm_node** svm_problem::x

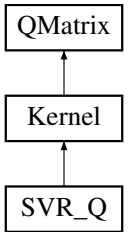
6.24.2.3 double* svm_problem::y

The documentation for this struct was generated from the following file:

- include/svm_core.h

6.25 SVR_Q Class Reference

Inheritance diagram for SVR_Q:



Public Member Functions

- SVR_Q (const svm_problem &prob, const svm_parameter &param)
- void swap_index (int i, int j) const
- Qfloat * get_Q (int i, int len) const
- double * get_QD () const
- ~SVR_Q ()

Private Attributes

- int l
- Cache * cache
- schar * sign
- int * index
- int next_buffer
- Qfloat * buffer [2]
- double * QD

Additional Inherited Members

6.25.1 Constructor & Destructor Documentation

6.25.1.1 SVR_Q::SVR_Q (const svm_problem & *prob,* const svm_parameter & *param*) [inline]

6.25.1.2 SVR_Q::~SVR_Q ( ) [inline]

6.25.2 Member Function Documentation

6.25.2.1 Qfloat* SVR_Q::get_Q (int *i,* int *len*) const [inline], [virtual]

Implements Kernel.

6.25.2.2 double* SVR_Q::get_QD ( ) const [inline], [virtual]

Implements Kernel.

6.25.2.3 void SVR_Q::swap_index (int *i,* int *j*) const [inline], [virtual]

Reimplemented from Kernel.

6.25.3 Field Documentation

6.25.3.1 Qfloat* SVR_Q::buffer[2] [private]

6.25.3.2 Cache* SVR_Q::cache [private]

6.25.3.3 int* SVR_Q::index [private]

6.25.3.4 int SVR_Q::l [private]

6.25.3.5 int SVR_Q::next_buffer [mutable], [private]

6.25.3.6 double* SVR_Q::QD [private]

6.25.3.7 schar* SVR_Q::sign [private]

The documentation for this class was generated from the following file:

- src/svm_core.cpp

Chapter 7

File Documentation

7.1 build/CMakeCache.txt File Reference

7.2 build/CMakeFiles/2.8.12.2/CompilerIdC/CMakeCCompilerId.c File Reference

Macros

- #define COMPILER_ID ""
- #define PLATFORM_ID ""
- #define ARCHITECTURE_ID ""
- #define DEC(n)
- #define HEX(n)

Functions

- int main (int argc, char ∗argv[ ])

Variables

- char const ∗ info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
- char const ∗ info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
- char const ∗ info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"

7.2.1 Macro Definition Documentation

7.2.1.1 #define ARCHITECTURE_ID ""

7.2.1.2 #define COMPILER_ID ""

7.2.1.3 #define DEC(*n*)

Value:

```
('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

7.2.1.4 #define HEX(*n*)

Value:

```
('0' + ((n)>>28 & 0xF)), \
  ('0' + ((n)>>24 & 0xF)), \
  ('0' + ((n)>>20 & 0xF)), \
  ('0' + ((n)>>16 & 0xF)), \
  ('0' + ((n)>>12 & 0xF)), \
  ('0' + ((n)>>8  & 0xF)), \
  ('0' + ((n)>>4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

7.2.1.5 #define PLATFORM_ID ""

7.2.2 Function Documentation

7.2.2.1 int main (int *argc,* char * *argv[ ]*)

7.2.3 Variable Documentation

7.2.3.1 char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"

7.2.3.2 char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"

7.2.3.3 char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"

7.3 build/CMakeFiles/2.8.12.2/CompilerIdCXX/CMakeCXXCompilerId.cpp File Reference

Macros

- #define COMPILER_ID ""
- #define PLATFORM_ID ""
- #define ARCHITECTURE_ID ""
- #define DEC(n)
- #define HEX(n)

Functions

- int main (int argc, char *argv[ ])

Variables

- char const * info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
- char const * info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
- char const * info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"

7.3.1 Macro Definition Documentation

7.3.1.1 #define ARCHITECTURE_ID ""

7.3.1.2 #define COMPILER_ID ""

7.3.1.3 #define DEC(*n*)

Value:

```
('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

7.3.1.4 #define HEX(*n*)

Value:

```
('0' + ((n)>>28 & 0xF)), \
  ('0' + ((n)>>24 & 0xF)), \
  ('0' + ((n)>>20 & 0xF)), \
  ('0' + ((n)>>16 & 0xF)), \
  ('0' + ((n)>>12 & 0xF)), \
  ('0' + ((n)>>8  & 0xF)), \
  ('0' + ((n)>>4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

7.3.1.5 #define PLATFORM_ID ""

7.3.2 Function Documentation

7.3.2.1 int main (int *argc,* char * *argv[]*)

7.3.3 Variable Documentation

7.3.3.1 char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"

7.3.3.2 char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"

7.3.3.3 char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"

7.4 build/CMakeFiles/conj.dir/link.txt File Reference

7.5 build/CMakeFiles/ex1.dir/link.txt File Reference

7.6 build/CMakeFiles/f1.dir/link.txt File Reference

7.7 build/CMakeFiles/f2.dir/link.txt File Reference

7.8 build/CMakeFiles/f3.dir/link.txt File Reference

7.9 build/CMakeFiles/z3test.dir/link.txt File Reference

7.10 build/CMakeFiles/TargetDirectories.txt File Reference

7.11 CMakeLists.txt File Reference

7.12 include/color.h File Reference

Provide support for colorful console ouput.

```
#include <iostream>
```

Enumerations

- enum color {
  RED = 0, YELLOW, GREEN, BLUE,
  WHITE }

  This enumeration contains all the colors predifined in project. Here we only introduce RED, YELLOW, GREEN, BLUE, WHITE which is enough for our output. You can import more color if you want.

Functions

- void set_console_color (std::ostream &out, int color=YELLOW)

  This function sets the given stream to the given color, YELLOW is default.
- void unset_console_color (std::ostream &out)

  This function sets the console color back to origin setting, not the previous setting. By origin, we mean black background, white foreground, no strong comparision.

7.12.1 Detailed Description

Provide support for colorful console ouput.

This file contains the necessary function support for colorful console text output. The usage is also simple. Before you output something, call function set_console_color. And remember to call unset_console_color after your output.

Author

Li Jiaying

Bug unset_console_color is set the console back to black background, white forground, no strong comparision instead of the previous setting.

7.12.2 Enumeration Type Documentation

7.12.2.1 enum color

This enumeration contains all the colors predifined in project. Here we only introduce RED, YELLOW, GREEN, BLUE, WHITE which is enough for our output. You can import more color if you want.

Enumerator

RED

YELLOW

GREEN

BLUE

WHITE

7.12.3 Function Documentation

7.12.3.1 void set_console_color (std::ostream & *out,* int *color* = YELLOW)

This function sets the given stream to the given color, YELLOW is default.

Parameters

out	The ostream to be changed, defines which stream you want to set
color	The Color to set. YELLOW by default.

7.12.3.2 void unset_console_color (std::ostream & *out*)

This function sets the console color back to origin setting, not the previous setting. By origin, we mean black background, white foreground, no strong comparision.

7.13 include/config.h File Reference

Macros

- #define VARS 2
- #define PRECISION 3

Functions

- bool register_program (int(∗func)(int ∗), const char ∗func_name=0)

Variables

- int(∗ target_program)(int ∗)
- const int max_items = 100000
- const int q_items = 1000
- const int init_exes = 6 ∗ VARS
- const int after_exes = 4 ∗ VARS
- const int random_exes = 2
- const int max_iter = 32

7.13.1 Macro Definition Documentation

7.13.1.1 #define PRECISION 3

7.13.1.2 #define VARS 2

7.13.2 Function Documentation

7.13.2.1 bool register_program (int(∗)(int ∗) *func,* const char ∗ *func_name* = 0)

7.13.3 Variable Documentation

7.13.3.1 const int after_exes = 4 ∗ VARS

7.13.3.2 const int init_exes = 6 * VARS

7.13.3.3 const int max_items = 100000

7.13.3.4 const int max_iter = 32

7.13.3.5 const int q_items = 1000

7.13.3.6 const int random_exes = 2

7.13.3.7 int(* target_program) (int *)

7.14 include/equation.h File Reference

```
#include "config.h"
#include <cmath>
#include <cfloat>
#include <stdarg.h>
#include <cstdlib>
#include <iostream>
#include <iomanip>
```

Data Structures

- class Solution
- class Equation

Variables

- int maxv
- int minv

7.14.1 Variable Documentation

7.14.1.1 int maxv

7.14.1.2 int minv

7.15 include/iif.h File Reference

```
#include "config.h"
#include "instrumentation.h"
#include "ml_algo.h"
#include "svm.h"
#include "svm_i.h"
#include "color.h"
#include "equation.h"
#include "states.h"
#include "iif_learn.h"
#include "iif_svm_learn.h"
#include "iif_svm_i_learn.h"
#include "iif_assert.h"
#include <iostream>
#include <float.h>
#include <string.h>
#include <assert.h>
#include <cstdlib>
```

Variables

- int minv
- int maxv

7.15.1 Variable Documentation

7.15.1.1 int maxv

7.15.1.2 int minv

7.16 include/iif_assert.h File Reference

Macros

- #define iif_assume(expr)
- #define iif_assert(expr)

Variables

- bool _passP
- bool _passQ
- int assume_times
- int assert_times

7.16.1 Macro Definition Documentation

7.16.1.1 #define iif_assert(*expr*)

Value:

```
do { \
    _passQ = (expr)? true : false;\
    assert_times++;\
} while(0)
```

7.16.1.2 #define iif_assume(*expr*)

Value:

```
do { \
    _passP = (expr)? true : false;\
    assume_times++;\
} while(0)
```

7.16.2 Variable Documentation

7.16.2.1 bool _passP

7.16.2.2 bool _passQ

7.16.2.3 int assert_times

7.16.2.4 int assume_times

7.17 include/iif_learn.h File Reference

```
#include "config.h"
#include "states.h"
#include "equation.h"
#include "instrumentation.h"
#include "color.h"
#include <iostream>
#include <float.h>
#include <string.h>
#include <assert.h>
```

Data Structures

- class IIF_learn

7.18 include/iif_svm_i_learn.h File Reference

```
#include "config.h"
#include "iif_learn.h"
#include "ml_algo.h"
#include "svm_i.h"
#include "color.h"
#include "equation.h"
#include <iostream>
#include <float.h>
#include <string.h>
#include <assert.h>
```

Data Structures

- class IIF_svm_i_learn

7.19 include/iif_svm_learn.h File Reference

```
#include "config.h"
#include "iif_learn.h"
#include "ml_algo.h"
#include "svm.h"
#include "color.h"
#include "equation.h"
#include <iostream>
#include <float.h>
#include <string.h>
#include <assert.h>
```

Data Structures

- class IIF_svm_learn

7.20 include/instrumentation.h File Reference

```
#include "config.h"
#include "states.h"
#include <stdarg.h>
```

Enumerations

- enum { NEGATIVE = -1, QUESTION, POSITIVE, COUNT_EXAMPLE }

Functions

- int add_state_int (int first,...)
- int add_state_double (double first,...)
- int m_int (int *)
- int m_double (double *)
- int before_loop ()
- int after_loop (States *)

7.20.1 Enumeration Type Documentation

7.20.1.1 anonymous enum

Enumerator

NEGATIVE

QUESTION

POSITIVE

COUNT_EXAMPLE

7.20.2 Function Documentation

7.20.2.1 int add_state_double (double *first*, ...)

7.20.2.2 int add_state_int (int *first*, ...)

7.20.2.3 int after_loop (States *)

7.20.2.4 int before_loop ( )

7.20.2.5 int m_double (double *)

7.20.2.6 int m_int (int *)

7.21 include/ml_algo.h File Reference

```
#include <iostream>
#include "states.h"
#include "equation.h"
```

Data Structures

- class ML_Algo

7.22 include/perceptron.h File Reference

```
#include "config.h"
#include "instrumentation.h"
#include "color.h"
#include "ml_algo.h"
```

Data Structures

- class Perceptron

7.23 include/states.h File Reference

```
#include "config.h"
#include <iostream>
```

Data Structures

- class States

7.24 include/svm.h File Reference

```
#include "ml_algo.h"
#include "svm_core.h"
```

Data Structures

- class SVM

7.25 include/svm_core.h File Reference

```
#include "config.h"
#include "instrumentation.h"
#include "color.h"
#include <iostream>
```

Data Structures

- struct svm_node
- struct svm_problem
- struct svm_parameter
- struct svm_model

Macros

- #define LIBSVM_VERSION 320

Enumerations

- enum {
  C_SVC, NU_SVC, ONE_CLASS, EPSILON_SVR,
  NU_SVR }
- enum {
  LINEAR, POLY, RBF, SIGMOID,
  PRECOMPUTED }

Functions

- struct svm_model ∗ svm_train (const struct svm_problem ∗prob, const struct svm_parameter ∗param)
- void svm_cross_validation (const struct svm_problem ∗prob, const struct svm_parameter ∗param, int nr_fold, double ∗target)
- int svm_save_model (const char ∗model_file_name, const struct svm_model ∗model)
- struct svm_model ∗ svm_load_model (const char ∗model_file_name)
- int svm_get_svm_type (const struct svm_model ∗model)
- int svm_get_nr_class (const struct svm_model ∗model)
- void svm_get_labels (const struct svm_model ∗model, int ∗label)
- void svm_get_sv_indices (const struct svm_model ∗model, int ∗sv_indices)
- int svm_get_nr_sv (const struct svm_model ∗model)
- double svm_get_svr_probability (const struct svm_model ∗model)

- double svm_predict_values (const struct svm_model *model, const struct svm_node *x, double *dec_values)
- double svm_predict (const struct svm_model *model, const struct svm_node *x)
- double svm_predict_probability (const struct svm_model *model, const struct svm_node *x, double *prob_estimates)
- void svm_free_model_content (struct svm_model *model_ptr)
- void svm_free_and_destroy_model (struct svm_model **model_ptr_ptr)
- void svm_destroy_param (struct svm_parameter *param)
- const char * svm_check_parameter (const struct svm_problem *prob, const struct svm_parameter *param)
- int svm_check_probability_model (const struct svm_model *model)
- void svm_set_print_string_function (void(*print_func)(const char *))
- int svm_model_visualization (const svm_model *model, Equation *equ)
- void print_svm_samples (const svm_problem *sp)
- struct svm_model * svm_l_train (const struct svm_problem *prob, const struct svm_parameter *param)

Variables

- int libsvm_version

7.25.1 Macro Definition Documentation

7.25.1.1 #define LIBSVM_VERSION 320

7.25.2 Enumeration Type Documentation

7.25.2.1 anonymous enum

Enumerator

C_SVC

NU_SVC

ONE_CLASS

EPSILON_SVR

NU_SVR

7.25.2.2 anonymous enum

Enumerator

LINEAR

POLY

RBF

SIGMOID

PRECOMPUTED

7.25.3 Function Documentation

7.25.3.1 void print_svm_samples (const **svm_problem** * *sp*)

7.25.3.2 const char* svm_check_parameter (const struct **svm_problem** * *prob,* const struct **svm_parameter** * *param*)

7.25.3.3 int svm_check_probability_model (const struct **svm_model** * *model*)

7.25.3.4 void svm_cross_validation (const struct **svm_problem** ∗ *prob,* const struct **svm_parameter** ∗ *param,* int *nr_fold,* double ∗ *target*)

7.25.3.5 void svm_destroy_param (struct **svm_parameter** ∗ *param*)

7.25.3.6 void svm_free_and_destroy_model (struct **svm_model** ∗∗ *model_ptr_ptr*)

7.25.3.7 void svm_free_model_content (struct **svm_model** ∗ *model_ptr*)

7.25.3.8 void svm_get_labels (const struct **svm_model** ∗ *model,* int ∗ *label*)

7.25.3.9 int svm_get_nr_class (const struct **svm_model** ∗ *model*)

7.25.3.10 int svm_get_nr_sv (const struct **svm_model** ∗ *model*)

7.25.3.11 void svm_get_sv_indices (const struct **svm_model** ∗ *model,* int ∗ *sv_indices*)

7.25.3.12 int svm_get_svm_type (const struct **svm_model** ∗ *model*)

7.25.3.13 double svm_get_svr_probability (const struct **svm_model** ∗ *model*)

7.25.3.14 struct **svm_model**∗ svm_l_train (const struct **svm_problem** ∗ *prob,* const struct **svm_parameter** ∗ *param*)

7.25.3.15 struct **svm_model**∗ svm_load_model (const char ∗ *model_file_name*)

7.25.3.16 int svm_model_visualization (const **svm_model** ∗ *model,* **Equation** ∗ *equ*)

7.25.3.17 double svm_predict (const struct **svm_model** ∗ *model,* const struct **svm_node** ∗ *x*)

7.25.3.18 double svm_predict_probability (const struct **svm_model** ∗ *model,* const struct **svm_node** ∗ *x,* double ∗ *prob_estimates*)

7.25.3.19 double svm_predict_values (const struct **svm_model** ∗ *model,* const struct **svm_node** ∗ *x,* double ∗ *dec_values*)

7.25.3.20 int svm_save_model (const char ∗ *model_file_name,* const struct **svm_model** ∗ *model*)

7.25.3.21 void svm_set_print_string_function (void(∗)(const char ∗) *print_func*)

7.25.3.22 struct **svm_model**∗ svm_train (const struct **svm_problem** ∗ *prob,* const struct **svm_parameter** ∗ *param*)

7.25.4 Variable Documentation

7.25.4.1 int libsvm_version

7.26 include/svm_i.h File Reference

```
#include "svm.h"
#include "color.h"
#include <iostream>
```

Data Structures

- class SVM_I

7.27 README.md File Reference

7.28 src/color.cpp File Reference

```
#include "color.h"
```

Functions

- void unset_console_color (std::ostream &out)

  This function sets the console color back to origin setting, not the previous setting. By origin, we mean black background, white foreground, no strong comparision.

7.28.1 Function Documentation

7.28.1.1 void unset_console_color (std::ostream & *out*)

This function sets the console color back to origin setting, not the previous setting. By origin, we mean black background, white foreground, no strong comparision.

7.29 src/config.cpp File Reference

```
#include "config.h"
#include "iif.h"
#include "instrumentation.h"
#include <iostream>
```

Functions

- bool check_target_program (int(*func)(int *))
- bool register_program (int(*func)(int *), const char *func_name)

Variables

- int assume_times
- int assert_times
- int(* target_program)(int *) = NULL
- int minv = -100
- int maxv = 100

7.29.1 Function Documentation

7.29.1.1 bool check_target_program (int(*)(int *) *func*)

7.29.1.2 bool register_program (int(*)(int *) *func,* const char * *func_name*)

7.29.2 Variable Documentation

7.29.2.1 int assert_times

7.29.2.2 int assume_times

7.29.2.3 int maxv = 100

7.29.2.4 int minv = -100

7.29.2.5 int(* target_program) (int *) = NULL

7.30 src/equation.cpp File Reference

```
#include "equation.h"
#include <cstdlib>
#include <vector>
#include <iostream>
```

Functions

- double _roundoff (double x)
- std::ostream & operator<< (std::ostream &out, const Solution &sol)
- std::ostream & operator<< (std::ostream &out, const Equation &equ)

Variables

- const double UPBOUND = pow(0.1, PRECISION)

7.30.1 Function Documentation

7.30.1.1 double _roundoff (double *x*) [inline]

7.30.1.2 std::ostream& operator<< (std::ostream & *out,* const Solution & *sol*)

support << operator simply output its elements as a tuple

7.30.1.3 std::ostream& operator<< (std::ostream & *out,* const Equation & *equ*)

Output the equation in a readable format Example: 2{0} + 3{1} >= 5

7.30.2 Variable Documentation

7.30.2.1 const double UPBOUND = pow(0.1, PRECISION)

7.31 src/iif_svm_i_learn.cpp File Reference

```
#include "config.h"
#include "ml_algo.h"
#include "svm.h"
#include "color.h"
#include "equation.h"
#include "iif_learn.h"
#include "iif_svm_i_learn.h"
#include <iostream>
#include <float.h>
#include <string.h>
#include <assert.h>
```

Functions

- static void print_null (const char *s)

7.31.1 Function Documentation

7.31.1.1 static void print_null (const char * *s*) [static]

7.32 src/iif_svm_learn.cpp File Reference

```
#include "config.h"
#include "ml_algo.h"
#include "svm.h"
#include "color.h"
#include "equation.h"
#include "iif_svm_learn.h"
#include <iostream>
#include <float.h>
#include <string.h>
#include <assert.h>
```

Functions

- static void print_null (const char *s)

7.32.1 Function Documentation

7.32.1.1 static void print_null (const char * *s*) [static]

7.33 src/instrumentation.cpp File Reference

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <time.h>
#include "instrumentation.h"
#include <assert.h>
```

Functions

- int add_state_int (int first...)
- int add_state_double (double first,...)
- int before_loop ()
- int after_loop (States *gsets)
- int m_double (double *p)
- int m_int (int *p)

Variables

- bool _passP = false
- bool _passQ = false
- int assume_times = 0
- int assert_times = 0
- char lt [4][10] = { "Negative", "Question", "Positive", "Bugtrace"}
- char(* LabelTable)[10] = &lt[1]
- double temp_states [256][VARS]
- int temp_index

7.33.1 Function Documentation

7.33.1.1 int add_state_double (double *first, ...*)

7.33.1.2 int add_state_int (int *first...*)

7.33.1.3 int after_loop (States * *gsets*)

7.33.1.4 int before_loop ( )

7.33.1.5 int m_double (double * *p*)

7.33.1.6 int m_int (int * *p*)

7.33.2 Variable Documentation

7.33.2.1 bool _passP = false

7.33.2.2 bool _passQ = false

7.33.2.3 int assert_times = 0

7.33.2.4 int assume_times = 0

7.33.2.5 char(* LabelTable)[10] = &lt[1]

7.33.2.6 char lt[4][10] = { "Negative", "Question", "Positive", "Bugtrace"}

7.33.2.7 int temp_index

7.33.2.8 double temp_states[256][VARS]

7.34 src/perceptron.cpp File Reference

```
#include "perceptron.h"
#include "string.h"
```

Functions

- std::ostream & operator<< (std::ostream &out, const Perceptron &perceptron)

7.34.1 Function Documentation

7.34.1.1 std::ostream& operator<< (std::ostream & *out,* const Perceptron & *perceptron*)

7.35 src/states.cpp File Reference

```
#include "config.h"
#include "string.h"
#include "states.h"
#include <cstdlib>
#include <vector>
#include <iostream>
```

Functions

- std::ostream & operator<< (std::ostream &out, const States &ss)

7.35.1 Function Documentation

7.35.1.1 std::ostream& operator<< (std::ostream & *out,* const States & *ss*)

7.36 src/svm.cpp File Reference

```
#include "svm.h"
#include "svm_core.h"
#include "string.h"
```

Functions

- std::ostream & operator<< (std::ostream &out, const SVM &svm)

7.36.1 Function Documentation

7.36.1.1 std::ostream& operator<< (std::ostream & *out,* const SVM & *svm*)

7.37 src/svm_core.cpp File Reference

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <float.h>
#include <string.h>
#include <stdarg.h>
#include <limits.h>
#include <locale.h>
#include "svm.h"
```

Data Structures

- class Cache
- struct Cache::head_t
- class QMatrix
- class Kernel
- class Solver
- struct Solver::SolutionInfo
- class Solver_NU
- class SVC_Q
- class ONE_CLASS_Q
- class SVR_Q
- struct decision_function

Macros

- #define INF HUGE_VAL
- #define TAU 1e-12
- #define Malloc(type, n) (type *)malloc((n)*sizeof(type))
- #define FSCANF(_stream, _format, _var) do{ if (fscanf(_stream, _format, _var) != 1) return false; }while(0)

Typedefs

- typedef float Qfloat
- typedef signed char schar

Functions

- template<class T >
  static T min (T x, T y)
- template<class T >
  static T max (T x, T y)
- template<class T >
  static void swap (T &x, T &y)
- template<class S , class T >
  static void clone (T *&dst, S *src, int n)
- static double powi (double base, int times)
- static void print_string_stdout (const char *s)
- static void info (const char *fmt,...)

- static void solve_c_svc (const svm_problem *prob, const svm_parameter *param, double *alpha, Solver::SolutionInfo *si, double Cp, double Cn)
- static void solve_nu_svc (const svm_problem *prob, const svm_parameter *param, double *alpha, Solver::SolutionInfo *si)
- static void solve_one_class (const svm_problem *prob, const svm_parameter *param, double *alpha, Solver::SolutionInfo *si)
- static void solve_epsilon_svr (const svm_problem *prob, const svm_parameter *param, double *alpha, Solver::SolutionInfo *si)
- static void solve_nu_svr (const svm_problem *prob, const svm_parameter *param, double *alpha, Solver::SolutionInfo *si)
- static decision_function svm_train_one (const svm_problem *prob, const svm_parameter *param, double Cp, double Cn)
- static void sigmoid_train (int l, const double *dec_values, const double *labels, double &A, double &B)
- static double sigmoid_predict (double decision_value, double A, double B)
- static void multiclass_probability (int k, double **r, double *p)
- static void svm_binary_svc_probability (const svm_problem *prob, const svm_parameter *param, double Cp, double Cn, double &probA, double &probB)
- static double svm_svr_probability (const svm_problem *prob, const svm_parameter *param)
- static void svm_group_classes (const svm_problem *prob, int *nr_class_ret, int **label_ret, int **start_ret, int **count_ret, int *perm)
- svm_model * svm_train (const svm_problem *prob, const svm_parameter *param)
- void svm_cross_validation (const svm_problem *prob, const svm_parameter *param, int nr_fold, double *target)
- int svm_get_svm_type (const svm_model *model)
- int svm_get_nr_class (const svm_model *model)
- void svm_get_labels (const svm_model *model, int *label)
- void svm_get_sv_indices (const svm_model *model, int *indices)
- int svm_get_nr_sv (const svm_model *model)
- double svm_get_svr_probability (const svm_model *model)
- double svm_predict_values (const svm_model *model, const svm_node *x, double *dec_values)
- double svm_predict (const svm_model *model, const svm_node *x)
- double svm_predict_probability (const svm_model *model, const svm_node *x, double *prob_estimates)
- int svm_save_model (const char *model_file_name, const svm_model *model)
- static char * readline (FILE *input)
- bool read_model_header (FILE *fp, svm_model *model)
- svm_model * svm_load_model (const char *model_file_name)
- void svm_free_model_content (svm_model *model_ptr)
- void svm_free_and_destroy_model (svm_model **model_ptr_ptr)
- void svm_destroy_param (svm_parameter *param)
- const char * svm_check_parameter (const svm_problem *prob, const svm_parameter *param)
- int svm_check_probability_model (const svm_model *model)
- void svm_set_print_string_function (void(*print_func)(const char *))
- void print_svm_samples (const svm_problem *sp)
- int svm_model_visualization (const svm_model *model, Equation *equ)
- struct svm_model * svm_l_train (const struct svm_problem *prob, const struct svm_parameter *param)

Variables

- int libsvm_version = LIBSVM_VERSION
- struct svm_node * positive_nodes = NULL
- struct svm_node * negative_nodes = NULL
- static void(* svm_print_string)(const char *) = &print_string_stdout
- static const char * svm_type_table [ ]
- static const char * kernel_type_table [ ]
- static char * line = NULL
- static int max_line_len

7.37.1 Macro Definition Documentation

7.37.1.1 #define FSCANF(_stream, _format, _var) do{ if (fscanf(_stream, _format, _var) != 1) return false; }while(0)

7.37.1.2 #define INF HUGE_VAL

7.37.1.3 #define Malloc(type, n) (type *)malloc((n)*sizeof(type))

7.37.1.4 #define TAU 1e-12

7.37.2 Typedef Documentation

7.37.2.1 typedef float Qfloat

7.37.2.2 typedef signed char schar

7.37.3 Function Documentation

7.37.3.1 template<class S , class T > static void clone (T *& dst, S * src, int n) [inline], [static]

7.37.3.2 static void info (const char * fmt, ...) [static]

7.37.3.3 template<class T > static T max (T x, T y) [inline], [static]

7.37.3.4 template<class T > static T min (T x, T y) [inline], [static]

7.37.3.5 static void multiclass_probability (int k, double ** r, double * p) [static]

7.37.3.6 static double powi (double base, int times) [inline], [static]

7.37.3.7 static void print_string_stdout (const char * s) [static]

7.37.3.8 void print_svm_samples (const svm_problem * sp)

7.37.3.9 bool read_model_header (FILE * fp, svm_model * model)

7.37.3.10 static char* readline (FILE * input) [static]

7.37.3.11 static double sigmoid_predict (double decision_value, double A, double B) [static]

7.37.3.12 static void sigmoid_train (int l, const double * dec_values, const double * labels, double & A, double & B) [static]

7.37.3.13 static void solve_c_svc (const svm_problem * prob, const svm_parameter * param, double * alpha, Solver::SolutionInfo * si, double Cp, double Cn) [static]

7.37.3.14 static void solve_epsilon_svr (const svm_problem * prob, const svm_parameter * param, double * alpha, Solver::SolutionInfo * si) [static]

7.37.3.15 static void solve_nu_svc (const svm_problem * prob, const svm_parameter * param, double * alpha, Solver::SolutionInfo * si) [static]

7.37.3.16 static void solve_nu_svr (const svm_problem * prob, const svm_parameter * param, double * alpha, Solver::SolutionInfo * si) [static]

7.37.3.17 static void solve_one_class (const **svm_problem** * *prob,* const **svm_parameter** * *param,* double * *alpha,* **Solver::SolutionInfo** * *si*) [static]

7.37.3.18 static void svm_binary_svc_probability (const **svm_problem** * *prob,* const **svm_parameter** * *param,* double *Cp,* double *Cn,* double & *probA,* double & *probB*) [static]

7.37.3.19 const char* svm_check_parameter (const **svm_problem** * *prob,* const **svm_parameter** * *param*)

7.37.3.20 int svm_check_probability_model (const **svm_model** * *model*)

7.37.3.21 void svm_cross_validation (const **svm_problem** * *prob,* const **svm_parameter** * *param,* int *nr_fold,* double * *target*)

7.37.3.22 void svm_destroy_param (**svm_parameter** * *param*)

7.37.3.23 void svm_free_and_destroy_model (**svm_model** ** *model_ptr_ptr*)

7.37.3.24 void svm_free_model_content (**svm_model** * *model_ptr*)

7.37.3.25 void svm_get_labels (const **svm_model** * *model,* int * *label*)

7.37.3.26 int svm_get_nr_class (const **svm_model** * *model*)

7.37.3.27 int svm_get_nr_sv (const **svm_model** * *model*)

7.37.3.28 void svm_get_sv_indices (const **svm_model** * *model,* int * *indices*)

7.37.3.29 int svm_get_svm_type (const **svm_model** * *model*)

7.37.3.30 double svm_get_svr_probability (const **svm_model** * *model*)

7.37.3.31 static void svm_group_classes (const **svm_problem** * *prob,* int * *nr_class_ret,* int ** *label_ret,* int ** *start_ret,* int ** *count_ret,* int * *perm*) [static]

7.37.3.32 struct **svm_model*** svm_l_train (const struct **svm_problem** * *prob,* const struct **svm_parameter** * *param*)

7.37.3.33 **svm_model*** svm_load_model (const char * *model_file_name*)

7.37.3.34 int svm_model_visualization (const **svm_model** * *model,* **Equation** * *equ*)

7.37.3.35 double svm_predict (const **svm_model** * *model,* const **svm_node** * *x*)

7.37.3.36 double svm_predict_probability (const **svm_model** * *model,* const **svm_node** * *x,* double * *prob_estimates*)

7.37.3.37 double svm_predict_values (const **svm_model** * *model,* const **svm_node** * *x,* double * *dec_values*)

7.37.3.38 int svm_save_model (const char * *model_file_name,* const **svm_model** * *model*)

7.37.3.39 void svm_set_print_string_function (void(*)(const char *) *print_func*)

7.37.3.40 static double svm_svr_probability (const **svm_problem** * *prob,* const **svm_parameter** * *param*) [static]

7.37.3.41 **svm_model*** svm_train (const **svm_problem** * *prob,* const **svm_parameter** * *param*)

7.37.3.42 static decision_function svm_train_one (const svm_problem * *prob*, const svm_parameter * *param*, double *Cp*, double *Cn*) [static]

7.37.3.43 template<class T > static void swap (T & *x*, T & *y*) [inline], [static]

7.37.4 Variable Documentation

7.37.4.1 const char* kernel_type_table[ ] [static]

Initial value:

```
=
{
    "linear","polynomial","rbf","sigmoid","precomputed",NULL
}
```

7.37.4.2 int libsvm_version = LIBSVM_VERSION

7.37.4.3 char* line = NULL [static]

7.37.4.4 int max_line_len [static]

7.37.4.5 struct svm_node* negative_nodes = NULL

7.37.4.6 struct svm_node* positive_nodes = NULL

7.37.4.7 void(* svm_print_string) (const char *) = &print_string_stdout [static]

7.37.4.8 const char* svm_type_table[ ] [static]

Initial value:

```
=
{
    "c_svc","nu_svc","one_class","epsilon_svr","nu_svr",NULL
}
```

7.38 src/svm_i.cpp File Reference

```
#include "svm_i.h"
#include "string.h"
#include <vector>
```

Functions

- std::ostream & operator<< (std::ostream &out, const SVM_I &svm_i)

7.38.1 Function Documentation

7.38.1.1 std::ostream& operator<< (std::ostream & *out*, const SVM_I & *svm_i*)

7.39 test/1_conj.cpp File Reference

```
#include "iif.h"
#include <iostream>
```

Functions

- static int nondet ()
- int conj (int *a)
- int main (int argc, char **argv)

7.39.1 Function Documentation

7.39.1.1 int conj (int * *a*)

7.39.1.2 int main (int *argc,* char ** *argv*)

7.39.1.3 static int nondet ( ) [static]

7.40 test/2_ex1.cpp File Reference

```
#include "iif.h"
```

Functions

- static int nondet ()
- int ex1 (int *a)
- int main (int argc, char **argv)

7.40.1 Function Documentation

7.40.1.1 int ex1 (int * *a*)

7.40.1.2 int main (int *argc,* char ** *argv*)

7.40.1.3 static int nondet ( ) [static]

7.41 test/2_f1.cpp File Reference

```
#include "iif.h"
```

Functions

- int f1 (int *a)
- int main (int argc, char **argv)

7.41.1 Function Documentation

7.41.1.1 int f1 (int * a)

7.41.1.2 int main (int argc, char ** argv)

7.42 test/2_f2.cpp File Reference

```
#include "iif.h"
#include <iostream>
```

Functions

- int f2 (int *a)
- int main (int argc, char **argv)

7.42.1 Function Documentation

7.42.1.1 int f2 (int * a)

7.42.1.2 int main (int argc, char ** argv)

7.43 test/2_z3test.cpp File Reference

```
#include "iif.h"
```

Functions

- int main (int argc, char **argv)

7.43.1 Function Documentation

7.43.1.1 int main (int argc, char ** argv)

7.44 test/3_f3.cpp File Reference

```
#include "iif.h"
```

Functions

- int f3 (int *a)
- int main (int argc, char **argv)

7.44.1 Function Documentation

7.44.1.1 int f3 (int * a)

7.44.1.2 int main (int *argc,* char ** *argv*)

Index