

Invariant Inference Framework

Generated by Doxygen 1.8.11

Contents

1	Invariant Inference Framework:	1
2	Bug List	3
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Data Structure Index	7
4.1	Data Structures	7
5	File Index	9
5.1	File List	9
6	Data Structure Documentation	11
6.1	Cache Class Reference	11
6.1.1	Constructor & Destructor Documentation	11
6.1.1.1	Cache(int l, long int size)	11
6.1.1.2	~Cache()	11
6.1.2	Member Function Documentation	11
6.1.2.1	get_data(const int index, Qfloat **data, int len)	11
6.1.2.2	lru_delete(head_t *h)	11
6.1.2.3	lru_insert(head_t *h)	12
6.1.2.4	swap_index(int i, int j)	12
6.1.3	Field Documentation	12
6.1.3.1	head	12
6.1.3.2	l	12
6.1.3.3	lru_head	12
6.1.3.4	size	12
6.2	decision_function Struct Reference	12
6.2.1	Field Documentation	12
6.2.1.1	alpha	12
6.2.1.2	rho	12
6.3	Equation Class Reference	12

6.3.1	Detailed Description	13
6.3.2	Constructor & Destructor Documentation	13
6.3.2.1	Equation()	13
6.3.2.2	Equation(double a0,...)	13
6.3.2.3	Equation(const Equation &equ)	13
6.3.3	Member Function Documentation	13
6.3.3.1	calc(Equation &equ, double *sol)	13
6.3.3.2	imply(const Equation &e2)	13
6.3.3.3	is_similar(const Equation &e, int precision=PRECISION)	14
6.3.3.4	linear_solver(Solution &sol)	14
6.3.3.5	linear_solver(const Equation *equ, Solution &sol)	14
6.3.3.6	operator=(const Equation &rhs)	14
6.3.3.7	roundoff(Equation &e)	14
6.3.4	Friends And Related Function Documentation	14
6.3.4.1	operator<<	15
6.3.5	Field Documentation	15
6.3.5.1	theta	15
6.3.5.2	theta0	15
6.4	Cache::head_t Struct Reference	15
6.4.1	Field Documentation	15
6.4.1.1	data	15
6.4.1.2	len	15
6.4.1.3	next	15
6.4.1.4	prev	15
6.5	IIF_learn Class Reference	15
6.5.1	Constructor & Destructor Documentation	16
6.5.1.1	IIF_learn(States *gsets, int(*func)(int *))	16
6.5.1.2	IIF_learn()	16
6.5.2	Member Function Documentation	16
6.5.2.1	init_gsets()	16
6.5.2.2	learn()=0	16
6.5.2.3	run_target(Solution &input)	16
6.5.3	Field Documentation	16
6.5.3.1	func	16
6.5.3.2	gsets	16
6.6	IIF_svm_i_learn Class Reference	16
6.6.1	Constructor & Destructor Documentation	17
6.6.1.1	IIF_svm_i_learn(States *gsets, int(*func)(int *), int max_iteration=max_iter)	17
6.6.1.2	IIF_svm_i_learn()	17
6.6.2	Member Function Documentation	17

6.6.2.1	learn()	17
6.6.3	Field Documentation	17
6.6.3.1	max_iteration	17
6.6.3.2	svm_i	17
6.7	IIF_svm_learn Class Reference	17
6.7.1	Constructor & Destructor Documentation	18
6.7.1.1	IIF_svm_learn(States *gsets, int(*func)(int *), int max_iteration=max_iter)	18
6.7.1.2	IIF_svm_learn()	18
6.7.2	Member Function Documentation	18
6.7.2.1	learn()	18
6.7.3	Field Documentation	18
6.7.3.1	max_iteration	18
6.7.3.2	svm	18
6.8	Kernel Class Reference	18
6.8.1	Constructor & Destructor Documentation	19
6.8.1.1	Kernel(int l, svm_node *const *x, const svm_parameter &param)	19
6.8.1.2	~Kernel()	19
6.8.2	Member Function Documentation	19
6.8.2.1	dot(const svm_node *px, const svm_node *py)	19
6.8.2.2	get_Q(int column, int len) const =0	19
6.8.2.3	get_QD() const =0	19
6.8.2.4	k_function(const svm_node *x, const svm_node *y, const svm_parameter &param)	20
6.8.2.5	kernel_linear(int i, int j) const	20
6.8.2.6	kernel_poly(int i, int j) const	20
6.8.2.7	kernel_precomputed(int i, int j) const	20
6.8.2.8	kernel_rbf(int i, int j) const	20
6.8.2.9	kernel_sigmoid(int i, int j) const	20
6.8.2.10	swap_index(int i, int j) const	20
6.8.3	Field Documentation	20
6.8.3.1	coef0	20
6.8.3.2	degree	20
6.8.3.3	gamma	20
6.8.3.4	kernel_function	20
6.8.3.5	kernel_type	20
6.8.3.6	x	20
6.8.3.7	x_square	20
6.9	ML_Algo Class Reference	20
6.9.1	Constructor & Destructor Documentation	21
6.9.1.1	ML_Algo()	21
6.9.2	Member Function Documentation	21

6.9.2.1	<code>_print(std::ostream &out) const</code>	21
6.9.2.2	<code>check_question_set(States &)=0</code>	21
6.9.2.3	<code>get_converged(Equation *, int)=0</code>	21
6.9.2.4	<code>predict(double *, int)=0</code>	21
6.9.2.5	<code>predict_on_training_set()=0</code>	21
6.9.2.6	<code>prepare_training_data(States *, int &, int &)=0</code>	21
6.9.2.7	<code>roundoff(int &)=0</code>	21
6.9.2.8	<code>size()=0</code>	22
6.9.2.9	<code>train()=0</code>	22
6.9.3	Friends And Related Function Documentation	22
6.9.3.1	<code>operator<<</code>	22
6.10	ONE_CLASS_Q Class Reference	22
6.10.1	Constructor & Destructor Documentation	22
6.10.1.1	<code>ONE_CLASS_Q(const svm_problem &prob, const svm_parameter &param)</code>	22
6.10.1.2	<code>~ONE_CLASS_Q()</code>	22
6.10.2	Member Function Documentation	22
6.10.2.1	<code>get_Q(int i, int len) const</code>	23
6.10.2.2	<code>get_QD() const</code>	23
6.10.2.3	<code>swap_index(int i, int j) const</code>	23
6.10.3	Field Documentation	23
6.10.3.1	<code>cache</code>	23
6.10.3.2	<code>QD</code>	23
6.11	Perceptron Class Reference	23
6.11.1	Constructor & Destructor Documentation	24
6.11.1.1	<code>Perceptron(void(*f)(const char *)=NULL)</code>	24
6.11.1.2	<code>~Perceptron()</code>	24
6.11.2	Member Function Documentation	24
6.11.2.1	<code>check_question_set(States &qset)</code>	24
6.11.2.2	<code>perceptron_train()</code>	24
6.11.2.3	<code>predict(double *v, int label=0)</code>	24
6.11.2.4	<code>predict_on_training_set()</code>	24
6.11.2.5	<code>prepare_training_data(States *gsets, int &pre_positive_size, int &pre_negative_size)</code>	24
6.11.2.6	<code>roundoff(int &num)</code>	24
6.11.2.7	<code>size()</code>	24
6.11.2.8	<code>train()</code>	24
6.11.3	Friends And Related Function Documentation	25
6.11.3.1	<code>operator<<</code>	25
6.11.4	Field Documentation	25
6.11.4.1	<code>length</code>	25

6.11.4.2	main_equation	25
6.11.4.3	training_label	25
6.11.4.4	training_set	25
6.12	QMatrix Class Reference	25
6.12.1	Constructor & Destructor Documentation	25
6.12.1.1	~QMatrix()	25
6.12.2	Member Function Documentation	25
6.12.2.1	get_Q(int column, int len) const =0	25
6.12.2.2	get_QD() const =0	25
6.12.2.3	swap_index(int i, int j) const =0	26
6.13	Solution Class Reference	26
6.13.1	Constructor & Destructor Documentation	26
6.13.1.1	Solution()	26
6.13.1.2	Solution(double a0,...)	26
6.13.2	Friends And Related Function Documentation	26
6.13.2.1	operator<<	26
6.13.3	Field Documentation	26
6.13.3.1	x	26
6.14	Solver::SolutionInfo Struct Reference	27
6.14.1	Field Documentation	27
6.14.1.1	obj	27
6.14.1.2	r	27
6.14.1.3	rho	27
6.14.1.4	upper_bound_n	27
6.14.1.5	upper_bound_p	27
6.15	Solver Class Reference	27
6.15.1	Member Enumeration Documentation	28
6.15.1.1	anonymous enum	28
6.15.2	Constructor & Destructor Documentation	28
6.15.2.1	Solver()	28
6.15.2.2	~Solver()	28
6.15.3	Member Function Documentation	28
6.15.3.1	be_shrunk(int i, double Gmax1, double Gmax2)	29
6.15.3.2	calculate_rho()	29
6.15.3.3	do_shrinking()	29
6.15.3.4	get_C(int i)	29
6.15.3.5	is_free(int i)	29
6.15.3.6	is_lower_bound(int i)	29
6.15.3.7	is_upper_bound(int i)	29
6.15.3.8	reconstruct_gradient()	29

6.15.3.9	<code>select_working_set(int &i, int &j)</code>	29
6.15.3.10	<code>Solve(int l, const QMatrix &Q, const double *p_, const schar *y_, double *alpha_↔ _, double Cp, double Cn, double eps, SolutionInfo *si, int shrinking)</code>	29
6.15.3.11	<code>swap_index(int i, int j)</code>	29
6.15.3.12	<code>update_alpha_status(int i)</code>	29
6.15.4	Field Documentation	29
6.15.4.1	<code>active_set</code>	29
6.15.4.2	<code>active_size</code>	29
6.15.4.3	<code>alpha</code>	29
6.15.4.4	<code>alpha_status</code>	29
6.15.4.5	<code>Cn</code>	29
6.15.4.6	<code>Cp</code>	29
6.15.4.7	<code>eps</code>	29
6.15.4.8	<code>G</code>	29
6.15.4.9	<code>G_bar</code>	29
6.15.4.10	<code>l</code>	29
6.15.4.11	<code>p</code>	29
6.15.4.12	<code>Q</code>	30
6.15.4.13	<code>QD</code>	30
6.15.4.14	<code>unshrink</code>	30
6.15.4.15	<code>y</code>	30
6.16	Solver_NU Class Reference	30
6.16.1	Constructor & Destructor Documentation	30
6.16.1.1	<code>Solver_NU()</code>	30
6.16.2	Member Function Documentation	30
6.16.2.1	<code>be_shrunk(int i, double Gmax1, double Gmax2, double Gmax3, double Gmax4)</code>	30
6.16.2.2	<code>calculate_rho()</code>	30
6.16.2.3	<code>do_shrinking()</code>	31
6.16.2.4	<code>select_working_set(int &i, int &j)</code>	31
6.16.2.5	<code>Solve(int l, const QMatrix &Q, const double *p, const schar *y, double *alpha, double Cp, double Cn, double eps, SolutionInfo *si, int shrinking)</code>	31
6.16.3	Field Documentation	31
6.16.3.1	<code>si</code>	31
6.17	States Class Reference	31
6.17.1	Constructor & Destructor Documentation	32
6.17.1.1	<code>States()</code>	32
6.17.1.2	<code>~States()</code>	32
6.17.2	Member Function Documentation	32
6.17.2.1	<code>add_states(double st[][VARS], int len)</code>	32
6.17.2.2	<code>print_trace(int num)</code>	32

6.17.2.3	size()	32
6.17.2.4	traces_num()	32
6.17.3	Friends And Related Function Documentation	32
6.17.3.1	operator<<	32
6.17.4	Field Documentation	32
6.17.4.1	index	32
6.17.4.2	label	32
6.17.4.3	max_size	32
6.17.4.4	p_index	32
6.17.4.5	values	32
6.18	SVC_Q Class Reference	32
6.18.1	Constructor & Destructor Documentation	33
6.18.1.1	SVC_Q(const svm_problem &prob, const svm_parameter ¶m, const schar *y_)	33
6.18.1.2	~SVC_Q()	33
6.18.2	Member Function Documentation	33
6.18.2.1	get_Q(int i, int len) const	33
6.18.2.2	get_QD() const	33
6.18.2.3	swap_index(int i, int j) const	33
6.18.3	Field Documentation	33
6.18.3.1	cache	33
6.18.3.2	QD	33
6.18.3.3	y	33
6.19	SVM Class Reference	33
6.19.1	Constructor & Destructor Documentation	34
6.19.1.1	SVM(void(*f)(const char *)=NULL, int size=10000)	34
6.19.1.2	~SVM()	34
6.19.2	Member Function Documentation	34
6.19.2.1	_print(std::ostream &out) const	34
6.19.2.2	check_question_set(States &qset)	35
6.19.2.3	get_converged(Equation *, int)	35
6.19.2.4	predict(double *v, int label=0)	35
6.19.2.5	predict_on_training_set()	35
6.19.2.6	prepare_training_data(States *gsets, int &pre_positive_size, int &pre_negative_size)	35
6.19.2.7	roundoff(int &num)	35
6.19.2.8	size()	35
6.19.2.9	train()	35
6.19.3	Friends And Related Function Documentation	35
6.19.3.1	operator<<	35

6.19.4	Field Documentation	35
6.19.4.1	main_equation	35
6.19.4.2	max_size	36
6.19.4.3	model	36
6.19.4.4	param	36
6.19.4.5	problem	36
6.19.4.6	training_label	36
6.19.4.7	training_set	36
6.20	SVM_I Class Reference	36
6.20.1	Constructor & Destructor Documentation	37
6.20.1.1	SVM_I(void(*f)(const char *)=NULL, int size=10000, int equ=16)	37
6.20.1.2	~SVM_I()	37
6.20.2	Member Function Documentation	37
6.20.2.1	_print(std::ostream &out) const	37
6.20.2.2	check_positives_and_one_negative()	37
6.20.2.3	check_question_set(States &qset)	37
6.20.2.4	get_converged(Equation *, int)	37
6.20.2.5	get_misclassified(int &idx)	37
6.20.2.6	predict(double *v, int label=0)	37
6.20.2.7	predict_on_training_set()	37
6.20.2.8	prepare_training_data(States *gsets, int &pre_positive_size, int &pre_negative_size)	37
6.20.2.9	roundoff(int &num)	38
6.20.2.10	size()	38
6.20.2.11	train()	38
6.20.3	Friends And Related Function Documentation	38
6.20.3.1	operator<<	38
6.20.4	Field Documentation	38
6.20.4.1	equ_num	38
6.20.4.2	equations	38
6.20.4.3	max_equ	38
6.20.4.4	model	38
6.20.4.5	negatives	38
6.20.4.6	param	38
6.21	svm_model Struct Reference	38
6.21.1	Field Documentation	39
6.21.1.1	free_sv	39
6.21.1.2	l	39
6.21.1.3	label	39
6.21.1.4	nr_class	39

6.21.1.5	nSV	39
6.21.1.6	param	39
6.21.1.7	probA	39
6.21.1.8	probB	39
6.21.1.9	rho	39
6.21.1.10	SV	39
6.21.1.11	sv_coef	39
6.21.1.12	sv_indices	39
6.22	svm_node Struct Reference	39
6.22.1	Friends And Related Function Documentation	39
6.22.1.1	operator<<	39
6.22.2	Field Documentation	39
6.22.2.1	value	39
6.23	svm_parameter Struct Reference	40
6.23.1	Field Documentation	40
6.23.1.1	C	40
6.23.1.2	cache_size	40
6.23.1.3	coef0	40
6.23.1.4	degree	40
6.23.1.5	eps	40
6.23.1.6	gamma	40
6.23.1.7	kernel_type	40
6.23.1.8	nr_weight	40
6.23.1.9	nu	40
6.23.1.10	p	40
6.23.1.11	probability	40
6.23.1.12	shrinking	40
6.23.1.13	svm_type	40
6.23.1.14	weight	40
6.23.1.15	weight_label	40
6.24	svm_problem Struct Reference	41
6.24.1	Friends And Related Function Documentation	41
6.24.1.1	operator<<	41
6.24.2	Field Documentation	41
6.24.2.1	l	41
6.24.2.2	x	41
6.24.2.3	y	41
6.25	SVR_Q Class Reference	41
6.25.1	Constructor & Destructor Documentation	42
6.25.1.1	SVR_Q(const svm_problem &prob, const svm_parameter ¶m)	42

6.25.1.2	~SVR_Q()	42
6.25.2	Member Function Documentation	42
6.25.2.1	get_Q(int i, int len) const	42
6.25.2.2	get_QD() const	42
6.25.2.3	swap_index(int i, int j) const	42
6.25.3	Field Documentation	42
6.25.3.1	buffer	42
6.25.3.2	cache	42
6.25.3.3	index	42
6.25.3.4	l	42
6.25.3.5	next_buffer	42
6.25.3.6	QD	42
6.25.3.7	sign	42
7	File Documentation	43
7.1	build/CMakeCache.txt File Reference	43
7.2	build/CMakeFiles/2.8.12.2/CompilerIdC/CMakeCCompilerId.c File Reference	43
7.2.1	Macro Definition Documentation	43
7.2.1.1	ARCHITECTURE_ID	43
7.2.1.2	COMPILER_ID	43
7.2.1.3	DEC	43
7.2.1.4	HEX	44
7.2.1.5	PLATFORM_ID	44
7.2.2	Function Documentation	44
7.2.2.1	main(int argc, char *argv[])	44
7.2.3	Variable Documentation	44
7.2.3.1	info_arch	44
7.2.3.2	info_compiler	44
7.2.3.3	info_platform	44
7.3	build/CMakeFiles/2.8.12.2/CompilerIdCXX/CMakeCXXCompilerId.cpp File Reference	44
7.3.1	Macro Definition Documentation	44
7.3.1.1	ARCHITECTURE_ID	44
7.3.1.2	COMPILER_ID	44
7.3.1.3	DEC	45
7.3.1.4	HEX	45
7.3.1.5	PLATFORM_ID	45
7.3.2	Function Documentation	45
7.3.2.1	main(int argc, char *argv[])	45
7.3.3	Variable Documentation	45
7.3.3.1	info_arch	45

7.3.3.2	info_compiler	45
7.3.3.3	info_platform	45
7.4	build/CMakeFiles/conj.dir/link.txt File Reference	45
7.5	build/CMakeFiles/ex1.dir/link.txt File Reference	45
7.6	build/CMakeFiles/f1.dir/link.txt File Reference	45
7.7	build/CMakeFiles/f2.dir/link.txt File Reference	45
7.8	build/CMakeFiles/f3.dir/link.txt File Reference	45
7.9	build/CMakeFiles/z3test.dir/link.txt File Reference	45
7.10	build/CMakeFiles/TargetDirectories.txt File Reference	45
7.11	CMakeLists.txt File Reference	46
7.12	include/color.h File Reference	46
7.12.1	Detailed Description	46
7.12.2	Enumeration Type Documentation	46
7.12.2.1	color	46
7.12.3	Function Documentation	47
7.12.3.1	set_console_color(std::ostream &out, int color=YELLOW)	47
7.12.3.2	unset_console_color(std::ostream &out)	47
7.13	include/config.h File Reference	47
7.13.1	Macro Definition Documentation	47
7.13.1.1	PRECISION	47
7.13.1.2	VAR_S	47
7.13.2	Function Documentation	47
7.13.2.1	register_program(int(*func)(int *), const char *func_name=0)	47
7.13.3	Variable Documentation	47
7.13.3.1	after_exes	47
7.13.3.2	init_exes	48
7.13.3.3	max_items	48
7.13.3.4	max_iter	48
7.13.3.5	q_items	48
7.13.3.6	random_exes	48
7.13.3.7	target_program	48
7.14	include/equation.h File Reference	48
7.14.1	Variable Documentation	48
7.14.1.1	maxv	48
7.14.1.2	minv	48
7.15	include/iif.h File Reference	49
7.15.1	Variable Documentation	49
7.15.1.1	maxv	49
7.15.1.2	minv	49
7.16	include/iif_assert.h File Reference	49

7.16.1	Macro Definition Documentation	49
7.16.1.1	iif_assert	49
7.16.1.2	iif_assume	50
7.16.2	Variable Documentation	50
7.16.2.1	_passP	50
7.16.2.2	_passQ	50
7.16.2.3	assert_times	50
7.16.2.4	assume_times	50
7.17	include/iif_learn.h File Reference	50
7.18	include/iif_svm_i_learn.h File Reference	50
7.19	include/iif_svm_learn.h File Reference	51
7.20	include/instrumentation.h File Reference	51
7.20.1	Enumeration Type Documentation	51
7.20.1.1	anonymous enum	51
7.20.2	Function Documentation	52
7.20.2.1	add_state_double(double first,...)	52
7.20.2.2	add_state_int(int first,...)	52
7.20.2.3	after_loop(States *)	52
7.20.2.4	before_loop()	52
7.20.2.5	m_double(double *)	52
7.20.2.6	m_int(int *)	52
7.21	include/ml_algo.h File Reference	52
7.22	include/perceptron.h File Reference	52
7.23	include/states.h File Reference	52
7.24	include/svm.h File Reference	53
7.25	include/svm_core.h File Reference	53
7.25.1	Macro Definition Documentation	54
7.25.1.1	LIBSVM_VERSION	54
7.25.2	Enumeration Type Documentation	54
7.25.2.1	anonymous enum	54
7.25.2.2	anonymous enum	54
7.25.3	Function Documentation	54
7.25.3.1	print_svm_samples(const svm_problem *sp)	54
7.25.3.2	svm_check_parameter(const struct svm_problem *prob, const struct svm_↵ parameter *param)	54
7.25.3.3	svm_check_probability_model(const struct svm_model *model)	54
7.25.3.4	svm_cross_validation(const struct svm_problem *prob, const struct svm_↵ parameter *param, int nr_fold, double *target)	55
7.25.3.5	svm_destroy_param(struct svm_parameter *param)	55
7.25.3.6	svm_free_and_destroy_model(struct svm_model **model_ptr_ptr)	55

7.25.3.7	<code>svm_free_model_content(struct svm_model *model_ptr)</code>	55
7.25.3.8	<code>svm_get_labels(const struct svm_model *model, int *label)</code>	55
7.25.3.9	<code>svm_get_nr_class(const struct svm_model *model)</code>	55
7.25.3.10	<code>svm_get_nr_sv(const struct svm_model *model)</code>	55
7.25.3.11	<code>svm_get_sv_indices(const struct svm_model *model, int *sv_indices)</code>	55
7.25.3.12	<code>svm_get_svm_type(const struct svm_model *model)</code>	55
7.25.3.13	<code>svm_get_svr_probability(const struct svm_model *model)</code>	55
7.25.3.14	<code>svm_l_train(const struct svm_problem *prob, const struct svm_parameter *param)</code>	55
7.25.3.15	<code>svm_load_model(const char *model_file_name)</code>	55
7.25.3.16	<code>svm_model_visualization(const struct svm_model *model, Equation *equ)</code>	55
7.25.3.17	<code>svm_predict(const struct svm_model *model, const struct svm_node *x)</code>	55
7.25.3.18	<code>svm_predict_probability(const struct svm_model *model, const struct svm_node *x, double *prob_estimates)</code>	55
7.25.3.19	<code>svm_predict_values(const struct svm_model *model, const struct svm_node *x, double *dec_values)</code>	55
7.25.3.20	<code>svm_save_model(const char *model_file_name, const struct svm_model *model)</code>	55
7.25.3.21	<code>svm_set_print_string_function(void(*print_func)(const char *))</code>	55
7.25.3.22	<code>svm_train(const struct svm_problem *prob, const struct svm_parameter *param)</code>	55
7.25.4	Variable Documentation	55
7.25.4.1	<code>libsvm_version</code>	55
7.26	<code>include/svm_i.h</code> File Reference	55
7.27	<code>README.md</code> File Reference	56
7.28	<code>src/color.cpp</code> File Reference	56
7.28.1	Function Documentation	56
7.28.1.1	<code>unset_console_color(std::ostream &out)</code>	56
7.29	<code>src/config.cpp</code> File Reference	56
7.29.1	Function Documentation	56
7.29.1.1	<code>check_target_program(int(*func)(int *))</code>	56
7.29.1.2	<code>register_program(int(*func)(int *), const char *func_name)</code>	56
7.29.2	Variable Documentation	56
7.29.2.1	<code>assert_times</code>	56
7.29.2.2	<code>assume_times</code>	57
7.29.2.3	<code>maxv</code>	57
7.29.2.4	<code>minv</code>	57
7.29.2.5	<code>target_program</code>	57
7.30	<code>src/equation.cpp</code> File Reference	57
7.30.1	Function Documentation	57
7.30.1.1	<code>_roundoff(double x)</code>	57
7.30.1.2	<code>operator<<(std::ostream &out, const Solution &sol)</code>	57
7.30.1.3	<code>operator<<(std::ostream &out, const Equation &equ)</code>	57

7.30.2	Variable Documentation	57
7.30.2.1	UPBOUND	57
7.31	src/iif_svm_i_learn.cpp File Reference	58
7.31.1	Function Documentation	58
7.31.1.1	print_null(const char *s)	58
7.32	src/iif_svm_learn.cpp File Reference	58
7.32.1	Function Documentation	58
7.32.1.1	print_null(const char *s)	58
7.33	src/instrumentation.cpp File Reference	58
7.33.1	Function Documentation	59
7.33.1.1	add_state_double(double first,...)	59
7.33.1.2	add_state_int(int first...)	59
7.33.1.3	after_loop(States *gsets)	59
7.33.1.4	before_loop()	59
7.33.1.5	m_double(double *p)	59
7.33.1.6	m_int(int *p)	59
7.33.2	Variable Documentation	59
7.33.2.1	_passP	59
7.33.2.2	_passQ	59
7.33.2.3	assert_times	59
7.33.2.4	assume_times	59
7.33.2.5	LabelTable	59
7.33.2.6	lt	59
7.33.2.7	temp_index	59
7.33.2.8	temp_states	59
7.34	src/perceptron.cpp File Reference	60
7.34.1	Function Documentation	60
7.34.1.1	operator<<(std::ostream &out, const Perceptron &perceptron)	60
7.35	src/states.cpp File Reference	60
7.35.1	Function Documentation	60
7.35.1.1	operator<<(std::ostream &out, const States &ss)	60
7.36	src/svm.cpp File Reference	60
7.36.1	Function Documentation	60
7.36.1.1	operator<<(std::ostream &out, const SVM &svm)	60
7.37	src/svm_core.cpp File Reference	61
7.37.1	Macro Definition Documentation	63
7.37.1.1	FSCANF	63
7.37.1.2	INF	63
7.37.1.3	Malloc	63
7.37.1.4	TAU	63

7.37.2	Typedef Documentation	63
7.37.2.1	Qfloat	63
7.37.2.2	schar	63
7.37.3	Function Documentation	63
7.37.3.1	clone(T *&dst, S *src, int n)	63
7.37.3.2	info(const char *fmt,...)	63
7.37.3.3	max(T x, T y)	63
7.37.3.4	min(T x, T y)	63
7.37.3.5	multiclass_probability(int k, double **r, double *p)	63
7.37.3.6	powi(double base, int times)	63
7.37.3.7	print_string_stdout(const char *s)	63
7.37.3.8	print_svm_samples(const svm_problem *sp)	63
7.37.3.9	read_model_header(FILE *fp, svm_model *model)	63
7.37.3.10	readline(FILE *input)	63
7.37.3.11	sigmoid_predict(double decision_value, double A, double B)	63
7.37.3.12	sigmoid_train(int l, const double *dec_values, const double *labels, double &A, double &B)	63
7.37.3.13	solve_c_svc(const svm_problem *prob, const svm_parameter *param, double *alpha, Solver::SolutionInfo *si, double Cp, double Cn)	63
7.37.3.14	solve_epsilon_svr(const svm_problem *prob, const svm_parameter *param, double *alpha, Solver::SolutionInfo *si)	63
7.37.3.15	solve_nu_svc(const svm_problem *prob, const svm_parameter *param, double *alpha, Solver::SolutionInfo *si)	63
7.37.3.16	solve_nu_svr(const svm_problem *prob, const svm_parameter *param, double *alpha, Solver::SolutionInfo *si)	63
7.37.3.17	solve_one_class(const svm_problem *prob, const svm_parameter *param, dou- ble *alpha, Solver::SolutionInfo *si)	64
7.37.3.18	svm_binary_svc_probability(const svm_problem *prob, const svm_parameter *param, double Cp, double Cn, double &probA, double &probB)	64
7.37.3.19	svm_check_parameter(const svm_problem *prob, const svm_parameter *param)	64
7.37.3.20	svm_check_probability_model(const svm_model *model)	64
7.37.3.21	svm_cross_validation(const svm_problem *prob, const svm_parameter *param, int nr_fold, double *target)	64
7.37.3.22	svm_destroy_param(svm_parameter *param)	64
7.37.3.23	svm_free_and_destroy_model(svm_model **model_ptr_ptr)	64
7.37.3.24	svm_free_model_content(svm_model *model_ptr)	64
7.37.3.25	svm_get_labels(const svm_model *model, int *label)	64
7.37.3.26	svm_get_nr_class(const svm_model *model)	64
7.37.3.27	svm_get_nr_sv(const svm_model *model)	64
7.37.3.28	svm_get_sv_indices(const svm_model *model, int *indices)	64
7.37.3.29	svm_get_svm_type(const svm_model *model)	64
7.37.3.30	svm_get_svr_probability(const svm_model *model)	64

7.37.3.31	<code>svm_group_classes(const svm_problem *prob, int *nr_class_ret, int **label_ret, int **start_ret, int **count_ret, int *perm)</code>	64
7.37.3.32	<code>svm_l_train(const struct svm_problem *prob, const struct svm_parameter *param)</code>	64
7.37.3.33	<code>svm_load_model(const char *model_file_name)</code>	64
7.37.3.34	<code>svm_model_visualization(const svm_model *model, Equation *equ)</code>	64
7.37.3.35	<code>svm_predict(const svm_model *model, const svm_node *x)</code>	64
7.37.3.36	<code>svm_predict_probability(const svm_model *model, const svm_node *x, double *prob_estimates)</code>	64
7.37.3.37	<code>svm_predict_values(const svm_model *model, const svm_node *x, double *dec_values)</code>	64
7.37.3.38	<code>svm_save_model(const char *model_file_name, const svm_model *model)</code>	64
7.37.3.39	<code>svm_set_print_string_function(void(*print_func)(const char *))</code>	64
7.37.3.40	<code>svm_svr_probability(const svm_problem *prob, const svm_parameter *param)</code>	64
7.37.3.41	<code>svm_train(const svm_problem *prob, const svm_parameter *param)</code>	64
7.37.3.42	<code>svm_train_one(const svm_problem *prob, const svm_parameter *param, double Cp, double Cn)</code>	65
7.37.3.43	<code>swap(T &x, T &y)</code>	65
7.37.4	Variable Documentation	65
7.37.4.1	<code>kernel_type_table</code>	65
7.37.4.2	<code>libsvm_version</code>	65
7.37.4.3	<code>line</code>	65
7.37.4.4	<code>max_line_len</code>	65
7.37.4.5	<code>negative_nodes</code>	65
7.37.4.6	<code>positive_nodes</code>	65
7.37.4.7	<code>svm_print_string</code>	65
7.37.4.8	<code>svm_type_table</code>	65
7.38	<code>src/svm_i.cpp</code> File Reference	65
7.38.1	Function Documentation	65
7.38.1.1	<code>operator<<(std::ostream &out, const SVM_I &svm_i)</code>	65
7.39	<code>test/1_conj.cpp</code> File Reference	66
7.39.1	Function Documentation	66
7.39.1.1	<code>conj(int *a)</code>	66
7.39.1.2	<code>main(int argc, char **argv)</code>	66
7.39.1.3	<code>nondet()</code>	66
7.40	<code>test/2_ex1.cpp</code> File Reference	66
7.40.1	Function Documentation	66
7.40.1.1	<code>ex1(int *a)</code>	66
7.40.1.2	<code>main(int argc, char **argv)</code>	66
7.40.1.3	<code>nondet()</code>	66
7.41	<code>test/2_f1.cpp</code> File Reference	66
7.41.1	Function Documentation	67

7.41.1.1	f1(int *a)	67
7.41.1.2	main(int argc, char **argv)	67
7.42	test/2_f2.cpp File Reference	67
7.42.1	Function Documentation	67
7.42.1.1	f2(int *a)	67
7.42.1.2	main(int argc, char **argv)	67
7.43	test/2_z3test.cpp File Reference	67
7.43.1	Function Documentation	67
7.43.1.1	main(int argc, char **argv)	67
7.44	test/3_f3.cpp File Reference	67
7.44.1	Function Documentation	67
7.44.1.1	f3(int *a)	67
7.44.1.2	main(int argc, char **argv)	68
Index		69

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 foreground, 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:

Cache	11
decision_function	12
Equation	12
Cache::head_t	15
IIF_learn	15
IIF_svm_i_learn	16
IIF_svm_learn	17
ML_Algo	20
Perceptron	23
SVM	33
SVM_I	36
QMatrix	25
Kernel	18
ONE_CLASS_Q	22
SVC_Q	32
SVR_Q	41
Solution	26
Solver::SolutionInfo	27
Solver	27
Solver_NU	30
States	31
svm_model	38
svm_node	39
svm_parameter	40
svm_problem	41

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

Cache	11
decision_function	12
Equation	12
Cache::head_t	15
IIF_learn	15
IIF_svm_i_learn	16
IIF_svm_learn	17
Kernel	18
ML_Algo	20
ONE_CLASS_Q	22
Perceptron	23
QMatrix	25
Solution	26
Solver::SolutionInfo	27
Solver	27
Solver_NU	30
States	31
SVC_Q	32
SVM	33
SVM_I	36
svm_model	38
svm_node	39
svm_parameter	40
svm_problem	41
SVR_Q	41

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

build/CMakeFiles/2.8.12.2/CompilerIdC/CMakeCCompilerId.c	43
build/CMakeFiles/2.8.12.2/CompilerIdCXX/CMakeCXXCompilerId.cpp	44
include/color.h	
Provide support for colorful console output	46
include/config.h	47
include/equation.h	48
include/iif.h	49
include/iif_assert.h	49
include/iif_learn.h	50
include/iif_svm_i_learn.h	50
include/iif_svm_learn.h	51
include/instrumentation.h	51
include/ml_algo.h	52
include/perceptron.h	52
include/states.h	52
include/svm.h	53
include/svm_core.h	53
include/svm_i.h	55
src/color.cpp	56
src/config.cpp	56
src/equation.cpp	57
src/iif_svm_i_learn.cpp	58
src/iif_svm_learn.cpp	58
src/instrumentation.cpp	58
src/perceptron.cpp	60
src/states.cpp	60
src/svm.cpp	60
src/svm_core.cpp	61
src/svm_i.cpp	65
test/1_conj.cpp	66
test/2_ex1.cpp	66
test/2_f1.cpp	66
test/2_f2.cpp	67
test/2_z3test.cpp	67
test/3_f3.cpp	67

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 coefficient of the `Equation`.

$$a_0 + [0] * x_0 + [1] * x_1 + \dots + [VARs] * x_{[VARs]} \geq 0$$

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

<code>e2</code>	is the equation right side we check whether *this ==> <code>e2</code> ??
-----------------	--

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 \sim e ???

Parameters

<i>precision</i>	defines how much variance we can bare. The default is 4, which means we can bare 0.0001 difference. In this case $1 \sim 1.00001$, but $1 \not\sim 1.000011$
------------------	---

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 calculate the most informative points in space It return a points really on the margin or next to the margin

Parameters

<i>sol</i>	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 calculate 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 coefficients we want to make it elegant, which is the purpose of involving this method Currently we have not done much work on this We have not even use gcd function to adjust the coefficients. For example. $1.2345 x_1 \geq 2.4690 \implies x_1 \geq 2$ $2 x_1 \geq 5.000001 \implies x_1 \geq 2.5$

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: $2\{0\} + 3\{1\} \geq 5$

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`

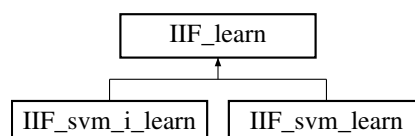
The documentation for this struct was generated from the following file:

- [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]

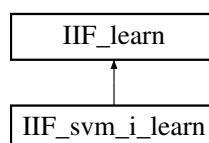
The documentation for this class was generated from the following file:

- [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](#)]

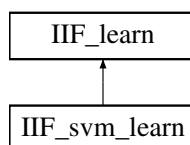
The documentation for this class was generated from the following files:

- [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]`

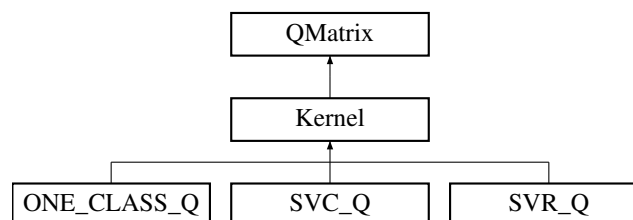
6.7.3.2 `SVM* IIF_svm_learn::svm` `[protected]`

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](#) ¶m)

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]`

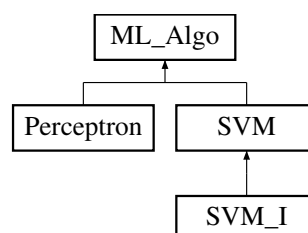
The documentation for this class was generated from the following file:

- [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

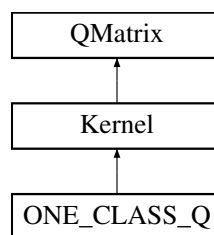
6.9.3.1 `std::ostream& operator<< (std::ostream & out, const ML_Algo & mla) [friend]`

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](#) ¶m)
- [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]`

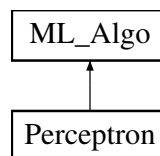
The documentation for this class was generated from the following file:

- [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]`

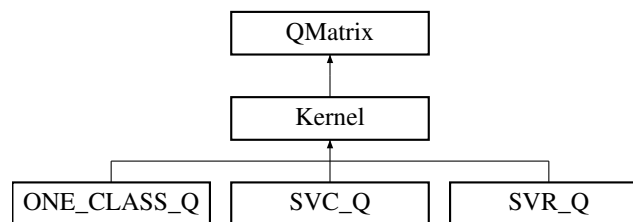
6.11.4.4 `double* Perceptron::training_set[max_items * 2]`

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

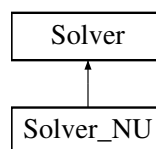
6.14.1.5 double Solver::SolutionInfo::upper_bound_p

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]

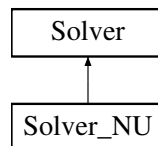
6.15.4.15 `schar* Solver::y` [protected]

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`

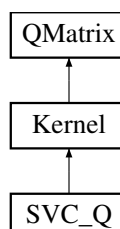
6.17.4.5 `double(* States::values)[VARS]`

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](#) ¶m, 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]

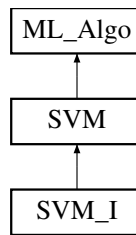
The documentation for this class was generated from the following file:

- [src/svm_core.cpp](#)

6.19 SVM Class Reference

```
#include <svm.h>
```

Inheritance diagram for SVM:



Public Member Functions

- [SVM](#) (void(*)(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`

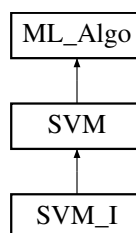
The documentation for this class was generated from the following files:

- [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`

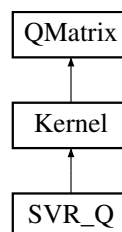
6.24.2.3 `double* svm_problem::y`

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 ¶m)`
- `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 output.

```
#include <iostream>
```

Enumerations

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

This enumeration contains all the colors predefined 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 comparison.

7.12.1 Detailed Description

Provide support for colorful console output.

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 foreground, no strong comparison instead of the previous setting.

7.12.2 Enumeration Type Documentation

7.12.2.1 enum color

This enumeration contains all the colors predefined 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

<i>out</i>	The ostream to be changed, defines which stream you want to set
<i>color</i>	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 comparison.

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 struct 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 comparison.

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 comparison.

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 [It](#) [4][10] = { "Negative", "Question", "Positive", "Bugtrace"}
- char(* [LabelTable](#))[10] = &It[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] = &It[1]

7.33.2.6 char [It](#)[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

- `_passP`
 - `iif_assert.h`, 50
 - `instrumentation.cpp`, 59
 - `_passQ`
 - `iif_assert.h`, 50
 - `instrumentation.cpp`, 59
 - `_print`
 - `ML_Algo`, 21
 - `SVM_I`, 37
 - `SVM`, 34
 - `_roundoff`
 - `equation.cpp`, 57
 - `~Cache`
 - `Cache`, 11
 - `~Kernel`
 - `Kernel`, 19
 - `~ONE_CLASS_Q`
 - `ONE_CLASS_Q`, 22
 - `~Perceptron`
 - `Perceptron`, 24
 - `~QMatrix`
 - `QMatrix`, 25
 - `~SVM`
 - `SVM`, 34
 - `~SVC_Q`
 - `SVC_Q`, 33
 - `~SVM_I`
 - `SVM_I`, 37
 - `~SVR_Q`
 - `SVR_Q`, 42
 - `~Solver`
 - `Solver`, 28
 - `~States`
 - `States`, 32
- `1_conj.cpp`
 - `conj`, 66
 - `main`, 66
 - `nondet`, 66
- `2_ex1.cpp`
 - `ex1`, 66
 - `main`, 66
 - `nondet`, 66
- `2_f1.cpp`
 - `f1`, 67
 - `main`, 67
- `2_f2.cpp`
 - `f2`, 67
 - `main`, 67
- `2_z3test.cpp`
 - `main`, 67
- `3_f3.cpp`
 - `f3`, 67
 - `main`, 67
- `ARCHITECTURE_ID`
 - `CMakeCCompilerId.c`, 43
 - `CMakeCXXCompilerId.cpp`, 44
- `active_set`
 - `Solver`, 29
- `active_size`
 - `Solver`, 29
- `add_state_double`
 - `instrumentation.cpp`, 59
 - `instrumentation.h`, 52
- `add_state_int`
 - `instrumentation.cpp`, 59
 - `instrumentation.h`, 52
- `add_states`
 - `States`, 32
- `after_exes`
 - `config.h`, 47
- `after_loop`
 - `instrumentation.cpp`, 59
 - `instrumentation.h`, 52
- `alpha`
 - `decision_function`, 12
 - `Solver`, 29
- `alpha_status`
 - `Solver`, 29
- `assert_times`
 - `config.cpp`, 56
 - `iif_assert.h`, 50
 - `instrumentation.cpp`, 59
- `assume_times`
 - `config.cpp`, 56
 - `iif_assert.h`, 50
 - `instrumentation.cpp`, 59
- `BLUE`
 - `color.h`, 46
- `be_shrunk`
 - `Solver`, 28
 - `Solver_NU`, 30
- `before_loop`
 - `instrumentation.cpp`, 59
 - `instrumentation.h`, 52
- `buffer`
 - `SVR_Q`, 42
- `build/CMakeCache.txt`, 43

- build/CMakeFiles/2.8.12.2/CompilerIdC/CMakeC↵
 - CompilerId.c, [43](#)
- build/CMakeFiles/2.8.12.2/CompilerIdCXX/CMakeCX↵
 - XCompilerId.cpp, [44](#)
- build/CMakeFiles/TargetDirectories.txt, [45](#)
- build/CMakeFiles/conj.dir/link.txt, [45](#)
- build/CMakeFiles/ex1.dir/link.txt, [45](#)
- build/CMakeFiles/f1.dir/link.txt, [45](#)
- build/CMakeFiles/f2.dir/link.txt, [45](#)
- build/CMakeFiles/f3.dir/link.txt, [45](#)
- build/CMakeFiles/z3test.dir/link.txt, [45](#)
- C
 - svm_parameter, [40](#)
- C_SVC
 - svm_core.h, [54](#)
- CMakeCCompilerId.c
 - ARCHITECTURE_ID, [43](#)
 - COMPILER_ID, [43](#)
 - DEC, [43](#)
 - HEX, [43](#)
 - info_arch, [44](#)
 - info_compiler, [44](#)
 - info_platform, [44](#)
 - main, [44](#)
 - PLATFORM_ID, [44](#)
- CMakeCXXCompilerId.cpp
 - ARCHITECTURE_ID, [44](#)
 - COMPILER_ID, [44](#)
 - DEC, [44](#)
 - HEX, [45](#)
 - info_arch, [45](#)
 - info_compiler, [45](#)
 - info_platform, [45](#)
 - main, [45](#)
 - PLATFORM_ID, [45](#)
- CMakeLists.txt, [46](#)
- COMPILER_ID
 - CMakeCCompilerId.c, [43](#)
 - CMakeCXXCompilerId.cpp, [44](#)
- COUNT_EXAMPLE
 - instrumentation.h, [51](#)
- Cache, [11](#)
 - ~Cache, [11](#)
 - Cache, [11](#)
 - get_data, [11](#)
 - head, [12](#)
 - I, [12](#)
 - Iru_delete, [11](#)
 - Iru_head, [12](#)
 - Iru_insert, [11](#)
 - size, [12](#)
 - swap_index, [12](#)
- cache
 - ONE_CLASS_Q, [23](#)
 - SVC_Q, [33](#)
 - SVR_Q, [42](#)
- Cache::head_t, [15](#)
 - data, [15](#)
 - len, [15](#)
 - next, [15](#)
 - prev, [15](#)
- cache_size
 - svm_parameter, [40](#)
- calc
 - Equation, [13](#)
- calculate_rho
 - Solver, [29](#)
 - Solver_NU, [30](#)
- check_postives_and_one_negative
 - SVM_I, [37](#)
- check_question_set
 - ML_Algo, [21](#)
 - Perceptron, [24](#)
 - SVM_I, [37](#)
 - SVM, [34](#)
- check_target_program
 - config.cpp, [56](#)
- clone
 - svm_core.cpp, [63](#)
- Cn
 - Solver, [29](#)
- coef0
 - Kernel, [20](#)
 - svm_parameter, [40](#)
- color
 - color.h, [46](#)
- color.cpp
 - unset_console_color, [56](#)
- color.h
 - BLUE, [46](#)
 - color, [46](#)
 - GREEN, [46](#)
 - RED, [46](#)
 - set_console_color, [47](#)
 - unset_console_color, [47](#)
 - WHITE, [46](#)
 - YELLOW, [46](#)
- config.cpp
 - assert_times, [56](#)
 - assume_times, [56](#)
 - check_target_program, [56](#)
 - maxv, [57](#)
 - minv, [57](#)
 - register_program, [56](#)
 - target_program, [57](#)
- config.h
 - after_exes, [47](#)
 - init_exes, [47](#)
 - max_items, [48](#)
 - max_iter, [48](#)
 - PRECISION, [47](#)
 - q_items, [48](#)
 - random_exes, [48](#)
 - register_program, [47](#)
 - target_program, [48](#)
 - VARS, [47](#)

- conj
 - 1_conj.cpp, 66
- Cp
 - Solver, 29
- DEC
 - CMakeCCompilerId.c, 43
 - CMakeCXXCompilerId.cpp, 44
- data
 - Cache::head_t, 15
- decision_function, 12
 - alpha, 12
 - rho, 12
- degree
 - Kernel, 20
 - svm_parameter, 40
- do_shrinking
 - Solver, 29
 - Solver_NU, 30
- dot
 - Kernel, 19
- EPSILON_SVR
 - svm_core.h, 54
- eps
 - Solver, 29
 - svm_parameter, 40
- equ_num
 - SVM_I, 38
- Equation, 12
 - calc, 13
 - Equation, 13
 - imply, 13
 - is_similar, 14
 - linear_solver, 14
 - operator<<, 14
 - operator=, 14
 - roundoff, 14
 - theta, 15
 - theta0, 15
- equation.cpp
 - _roundoff, 57
 - operator<<, 57
 - UPBOUND, 57
- equation.h
 - maxv, 48
 - minv, 48
- equations
 - SVM_I, 38
- ex1
 - 2_ex1.cpp, 66
- f1
 - 2_f1.cpp, 67
- f2
 - 2_f2.cpp, 67
- f3
 - 3_f3.cpp, 67
- FREE
 - Solver, 28
- FSCANF
 - svm_core.cpp, 63
- free_sv
 - svm_model, 39
- func
 - IIF_learn, 16
- G
 - Solver, 29
- G_bar
 - Solver, 29
- GREEN
 - color.h, 46
- gamma
 - Kernel, 20
 - svm_parameter, 40
- get_QD
 - Kernel, 19
 - ONE_CLASS_Q, 23
 - QMatrix, 25
 - SVC_Q, 33
 - SVR_Q, 42
- get_C
 - Solver, 29
- get_converged
 - ML_Algo, 21
 - SVM_I, 37
 - SVM, 35
- get_data
 - Cache, 11
- get_misclassified
 - SVM_I, 37
- get_Q
 - Kernel, 19
 - ONE_CLASS_Q, 22
 - QMatrix, 25
 - SVC_Q, 33
 - SVR_Q, 42
- gsets
 - IIF_learn, 16
- HEX
 - CMakeCCompilerId.c, 43
 - CMakeCXXCompilerId.cpp, 45
- head
 - Cache, 12
- IIF_learn, 15
 - func, 16
 - gsets, 16
 - IIF_learn, 16
 - init_gsets, 16
 - learn, 16
 - run_target, 16
- IIF_svm_i_learn, 16
 - IIF_svm_i_learn, 17
 - learn, 17
 - max_iteration, 17

- svm_i, 17
- IIF_svm_learn, 17
 - IIF_svm_learn, 18
 - learn, 18
 - max_iteration, 18
 - svm, 18
- INF
 - svm_core.cpp, 63
- iif.h
 - maxv, 49
 - minv, 49
- iif_assert
 - iif_assert.h, 49
- iif_assert.h
 - _passP, 50
 - _passQ, 50
 - assert_times, 50
 - assume_times, 50
 - iif_assert, 49
 - iif_assume, 50
- iif_assume
 - iif_assert.h, 50
- iif_svm_i_learn.cpp
 - print_null, 58
- iif_svm_learn.cpp
 - print_null, 58
- imply
 - Equation, 13
- include/color.h, 46
- include/config.h, 47
- include/equation.h, 48
- include/iif.h, 49
- include/iif_assert.h, 49
- include/iif_learn.h, 50
- include/iif_svm_i_learn.h, 50
- include/iif_svm_learn.h, 51
- include/instrumentation.h, 51
- include/ml_algo.h, 52
- include/perceptron.h, 52
- include/states.h, 52
- include/svm.h, 53
- include/svm_core.h, 53
- include/svm_i.h, 55
- index
 - SVR_Q, 42
 - States, 32
- info
 - svm_core.cpp, 63
- info_arch
 - CMakeCCompilerId.c, 44
 - CMakeCXXCompilerId.cpp, 45
- info_compiler
 - CMakeCCompilerId.c, 44
 - CMakeCXXCompilerId.cpp, 45
- info_platform
 - CMakeCCompilerId.c, 44
 - CMakeCXXCompilerId.cpp, 45
- init_exes
 - config.h, 47
- init_gsets
 - IIF_learn, 16
- instrumentation.cpp
 - _passP, 59
 - _passQ, 59
 - add_state_double, 59
 - add_state_int, 59
 - after_loop, 59
 - assert_times, 59
 - assume_times, 59
 - before_loop, 59
 - LabelTable, 59
 - lt, 59
 - m_double, 59
 - m_int, 59
 - temp_index, 59
 - temp_states, 59
- instrumentation.h
 - add_state_double, 52
 - add_state_int, 52
 - after_loop, 52
 - before_loop, 52
 - COUNT_EXAMPLE, 51
 - m_double, 52
 - m_int, 52
 - NEGATIVE, 51
 - POSITIVE, 51
 - QUESTION, 51
- is_free
 - Solver, 29
- is_lower_bound
 - Solver, 29
- is_similar
 - Equation, 14
- is_upper_bound
 - Solver, 29
- k_function
 - Kernel, 19
- Kernel, 18
 - ~Kernel, 19
 - coef0, 20
 - degree, 20
 - dot, 19
 - gamma, 20
 - get_QD, 19
 - get_Q, 19
 - k_function, 19
 - Kernel, 19
 - kernel_function, 20
 - kernel_linear, 20
 - kernel_poly, 20
 - kernel_precomputed, 20
 - kernel_rbf, 20
 - kernel_sigmoid, 20
 - kernel_type, 20
 - swap_index, 20
 - x, 20

- x_square, 20
- kernel_function
 - Kernel, 20
- kernel_linear
 - Kernel, 20
- kernel_poly
 - Kernel, 20
- kernel_precomputed
 - Kernel, 20
- kernel_rbf
 - Kernel, 20
- kernel_sigmoid
 - Kernel, 20
- kernel_type
 - Kernel, 20
 - svm_parameter, 40
- kernel_type_table
 - svm_core.cpp, 65
- I
 - Cache, 12
 - SVR_Q, 42
 - Solver, 29
 - svm_model, 39
 - svm_problem, 41
- LIBSVM_VERSION
 - svm_core.h, 54
- LINEAR
 - svm_core.h, 54
- LOWER_BOUND
 - Solver, 28
- label
 - States, 32
 - svm_model, 39
- LabelTable
 - instrumentation.cpp, 59
- learn
 - IIF_learn, 16
 - IIF_svm_i_learn, 17
 - IIF_svm_learn, 18
- len
 - Cache::head_t, 15
- length
 - Perceptron, 25
- libsvm_version
 - svm_core.cpp, 65
 - svm_core.h, 55
- line
 - svm_core.cpp, 65
- linear_solver
 - Equation, 14
- lru_delete
 - Cache, 11
- lru_head
 - Cache, 12
- lru_insert
 - Cache, 11
- It
 - instrumentation.cpp, 59
- m_double
 - instrumentation.cpp, 59
 - instrumentation.h, 52
- m_int
 - instrumentation.cpp, 59
 - instrumentation.h, 52
- ML_Algo, 20
 - _print, 21
 - check_question_set, 21
 - get_converged, 21
 - ML_Algo, 21
 - operator<<, 22
 - predict, 21
 - predict_on_training_set, 21
 - prepare_training_data, 21
 - roundoff, 21
 - size, 21
 - train, 22
- main
 - 1_conj.cpp, 66
 - 2_ex1.cpp, 66
 - 2_f1.cpp, 67
 - 2_f2.cpp, 67
 - 2_z3test.cpp, 67
 - 3_f3.cpp, 67
 - CMakeCCompilerId.c, 44
 - CMakeCXXCompilerId.cpp, 45
- main_equation
 - Perceptron, 25
 - SVM, 35
- Malloc
 - svm_core.cpp, 63
- max
 - svm_core.cpp, 63
- max_equ
 - SVM_I, 38
- max_items
 - config.h, 48
- max_iter
 - config.h, 48
- max_iteration
 - IIF_svm_i_learn, 17
 - IIF_svm_learn, 18
- max_line_len
 - svm_core.cpp, 65
- max_size
 - SVM, 35
 - States, 32
- maxv
 - config.cpp, 57
 - equation.h, 48
 - iif.h, 49
- min
 - svm_core.cpp, 63
- minv
 - config.cpp, 57
 - equation.h, 48
 - iif.h, 49

- model
 - SVM_I, [38](#)
 - SVM, [36](#)
- multiclass_probability
 - svm_core.cpp, [63](#)
- NEGATIVE
 - instrumentation.h, [51](#)
- nSV
 - svm_model, [39](#)
- NU_SVC
 - svm_core.h, [54](#)
- NU_SVR
 - svm_core.h, [54](#)
- negative_nodes
 - svm_core.cpp, [65](#)
- negatives
 - SVM_I, [38](#)
- next
 - Cache::head_t, [15](#)
- next_buffer
 - SVR_Q, [42](#)
- nondet
 - 1_conj.cpp, [66](#)
 - 2_ex1.cpp, [66](#)
- nr_class
 - svm_model, [39](#)
- nr_weight
 - svm_parameter, [40](#)
- nu
 - svm_parameter, [40](#)
- ONE_CLASS_Q, [22](#)
 - ~ONE_CLASS_Q, [22](#)
 - cache, [23](#)
 - get_QD, [23](#)
 - get_Q, [22](#)
 - ONE_CLASS_Q, [22](#)
 - QD, [23](#)
 - swap_index, [23](#)
- ONE_CLASS
 - svm_core.h, [54](#)
- obj
 - Solver::SolutionInfo, [27](#)
- operator<<
 - Equation, [14](#)
 - equation.cpp, [57](#)
 - ML_Algo, [22](#)
 - Perceptron, [25](#)
 - perceptron.cpp, [60](#)
 - SVM_I, [38](#)
 - SVM, [35](#)
 - Solution, [26](#)
 - States, [32](#)
 - states.cpp, [60](#)
 - svm.cpp, [60](#)
 - svm_i.cpp, [65](#)
 - svm_node, [39](#)
 - svm_problem, [41](#)
- operator=
 - Equation, [14](#)
- p
 - Solver, [29](#)
 - svm_parameter, [40](#)
- p_index
 - States, [32](#)
- PLATFORM_ID
 - CMakeCCompilerId.c, [44](#)
 - CMakeCXXCompilerId.cpp, [45](#)
- POLY
 - svm_core.h, [54](#)
- POSITIVE
 - instrumentation.h, [51](#)
- PRECISION
 - config.h, [47](#)
- PRECOMPUTED
 - svm_core.h, [54](#)
- param
 - SVM_I, [38](#)
 - SVM, [36](#)
 - svm_model, [39](#)
- Perceptron, [23](#)
 - ~Perceptron, [24](#)
 - check_question_set, [24](#)
 - length, [25](#)
 - main_equation, [25](#)
 - operator<<, [25](#)
 - Perceptron, [24](#)
 - perceptron_train, [24](#)
 - predict, [24](#)
 - predict_on_training_set, [24](#)
 - prepare_training_data, [24](#)
 - roundoff, [24](#)
 - size, [24](#)
 - train, [24](#)
 - training_label, [25](#)
 - training_set, [25](#)
- perceptron.cpp
 - operator<<, [60](#)
- perceptron_train
 - Perceptron, [24](#)
- positive_nodes
 - svm_core.cpp, [65](#)
- powi
 - svm_core.cpp, [63](#)
- predict
 - ML_Algo, [21](#)
 - Perceptron, [24](#)
 - SVM_I, [37](#)
 - SVM, [35](#)
- predict_on_training_set
 - ML_Algo, [21](#)
 - Perceptron, [24](#)
 - SVM_I, [37](#)
 - SVM, [35](#)
- prepare_training_data
 - ML_Algo, [21](#)

- Perceptron, 24
- SVM_I, 37
- SVM, 35
- prev
 - Cache::head_t, 15
- print_null
 - iif_svm_i_learn.cpp, 58
 - iif_svm_learn.cpp, 58
- print_string_stdout
 - svm_core.cpp, 63
- print_svm_samples
 - svm_core.cpp, 63
 - svm_core.h, 54
- print_trace
 - States, 32
- probA
 - svm_model, 39
- probability
 - svm_parameter, 40
- probB
 - svm_model, 39
- problem
 - SVM, 36
- Q
 - Solver, 29
- q_items
 - config.h, 48
- QMatrix, 25
 - ~QMatrix, 25
 - get_QD, 25
 - get_Q, 25
 - swap_index, 25
- QUESTION
 - instrumentation.h, 51
- QD
 - ONE_CLASS_Q, 23
 - SVC_Q, 33
 - SVR_Q, 42
 - Solver, 30
- Qfloat
 - svm_core.cpp, 63
- r
 - Solver::SolutionInfo, 27
- RBF
 - svm_core.h, 54
- README.md, 56
- RED
 - color.h, 46
- random_exes
 - config.h, 48
- read_model_header
 - svm_core.cpp, 63
- readline
 - svm_core.cpp, 63
- reconstruct_gradient
 - Solver, 29
- register_program
 - config.cpp, 56
 - config.h, 47
- rho
 - decision_function, 12
 - Solver::SolutionInfo, 27
 - svm_model, 39
- roundoff
 - Equation, 14
 - ML_Algo, 21
 - Perceptron, 24
 - SVM_I, 37
 - SVM, 35
- run_target
 - IIF_learn, 16
- SIGMOID
 - svm_core.h, 54
- SVC_Q, 32
 - ~SVC_Q, 33
 - cache, 33
 - get_QD, 33
 - get_Q, 33
 - QD, 33
 - SVC_Q, 33
 - swap_index, 33
 - y, 33
- SVM_I, 36
 - _print, 37
 - ~SVM_I, 37
 - check_postives_and_one_negative, 37
 - check_question_set, 37
 - equ_num, 38
 - equations, 38
 - get_converged, 37
 - get_misclassified, 37
 - max_equ, 38
 - model, 38
 - negatives, 38
 - operator<<, 38
 - param, 38
 - predict, 37
 - predict_on_training_set, 37
 - prepare_training_data, 37
 - roundoff, 37
 - SVM_I, 37
 - size, 38
 - train, 38
- SVR_Q, 41
 - ~SVR_Q, 42
 - buffer, 42
 - cache, 42
 - get_QD, 42
 - get_Q, 42
 - index, 42
 - l, 42
 - next_buffer, 42
 - QD, 42
 - SVR_Q, 42
 - sign, 42

- swap_index, 42
- SVM, 33
 - _print, 34
 - ~SVM, 34
 - check_question_set, 34
 - get_converged, 35
 - main_equation, 35
 - max_size, 35
 - model, 36
 - operator<<, 35
 - param, 36
 - predict, 35
 - predict_on_training_set, 35
 - prepare_training_data, 35
 - problem, 36
 - roundoff, 35
 - SVM, 34
 - size, 35
 - train, 35
 - training_label, 36
 - training_set, 36
- schar
 - svm_core.cpp, 63
- select_working_set
 - Solver, 29
 - Solver_NU, 31
- set_console_color
 - color.h, 47
- shrinking
 - svm_parameter, 40
- si
 - Solver_NU, 31
- sigmoid_predict
 - svm_core.cpp, 63
- sigmoid_train
 - svm_core.cpp, 63
- sign
 - SVR_Q, 42
- size
 - Cache, 12
 - ML_Algo, 21
 - Perceptron, 24
 - SVM_I, 38
 - SVM, 35
 - States, 32
- Solution, 26
 - operator<<, 26
 - Solution, 26
 - x, 26
- Solve
 - Solver, 29
 - Solver_NU, 31
- solve_c_svc
 - svm_core.cpp, 63
- solve_epsilon_svr
 - svm_core.cpp, 63
- solve_nu_svc
 - svm_core.cpp, 63
- solve_nu_svr
 - svm_core.cpp, 63
- solve_one_class
 - svm_core.cpp, 63
- Solver, 27
 - ~Solver, 28
 - active_set, 29
 - active_size, 29
 - alpha, 29
 - alpha_status, 29
 - be_shrunk, 28
 - calculate_rho, 29
 - Cn, 29
 - Cp, 29
 - do_shrinking, 29
 - eps, 29
 - FREE, 28
 - G, 29
 - G_bar, 29
 - get_C, 29
 - is_free, 29
 - is_lower_bound, 29
 - is_upper_bound, 29
 - I, 29
 - LOWER_BOUND, 28
 - p, 29
 - Q, 29
 - QD, 30
 - reconstruct_gradient, 29
 - select_working_set, 29
 - Solve, 29
 - Solver, 28
 - swap_index, 29
 - UPPER_BOUND, 28
 - unshrink, 30
 - update_alpha_status, 29
 - y, 30
- Solver::SolutionInfo, 27
 - obj, 27
 - r, 27
 - rho, 27
 - upper_bound_n, 27
 - upper_bound_p, 27
- Solver_NU, 30
 - be_shrunk, 30
 - calculate_rho, 30
 - do_shrinking, 30
 - select_working_set, 31
 - si, 31
 - Solve, 31
 - Solver_NU, 30
- src/color.cpp, 56
- src/config.cpp, 56
- src/equation.cpp, 57
- src/iif_svm_i_learn.cpp, 58
- src/iif_svm_learn.cpp, 58
- src/instrumentation.cpp, 58
- src/perceptron.cpp, 60

- src/states.cpp, 60
- src/svm.cpp, 60
- src/svm_core.cpp, 61
- src/svm_i.cpp, 65
- States, 31
 - ~States, 32
 - add_states, 32
 - index, 32
 - label, 32
 - max_size, 32
 - operator<<, 32
 - p_index, 32
 - print_trace, 32
 - size, 32
 - States, 32
 - traces_num, 32
 - values, 32
- states.cpp
 - operator<<, 60
- SV
 - svm_model, 39
- sv_coef
 - svm_model, 39
- sv_indices
 - svm_model, 39
- svm
 - lIF_svm_learn, 18
- svm.cpp
 - operator<<, 60
- svm_l_train
 - svm_core.cpp, 64
 - svm_core.h, 55
- svm_binary_svc_probability
 - svm_core.cpp, 64
- svm_check_parameter
 - svm_core.cpp, 64
 - svm_core.h, 54
- svm_check_probability_model
 - svm_core.cpp, 64
 - svm_core.h, 54
- svm_core.cpp
 - clone, 63
 - FSCANF, 63
 - INF, 63
 - info, 63
 - kernel_type_table, 65
 - libsvm_version, 65
 - line, 65
 - Malloc, 63
 - max, 63
 - max_line_len, 65
 - min, 63
 - multiclass_probability, 63
 - negative_nodes, 65
 - positive_nodes, 65
 - powi, 63
 - print_string_stdout, 63
 - print_svm_samples, 63
 - Qfloat, 63
 - read_model_header, 63
 - readline, 63
 - schar, 63
 - sigmoid_predict, 63
 - sigmoid_train, 63
 - solve_c_svc, 63
 - solve_epsilon_svr, 63
 - solve_nu_svc, 63
 - solve_nu_svr, 63
 - solve_one_class, 63
 - svm_l_train, 64
 - svm_binary_svc_probability, 64
 - svm_check_parameter, 64
 - svm_check_probability_model, 64
 - svm_cross_validation, 64
 - svm_destroy_param, 64
 - svm_free_and_destroy_model, 64
 - svm_free_model_content, 64
 - svm_get_labels, 64
 - svm_get_nr_class, 64
 - svm_get_nr_sv, 64
 - svm_get_sv_indices, 64
 - svm_get_svm_type, 64
 - svm_get_svr_probability, 64
 - svm_group_classes, 64
 - svm_load_model, 64
 - svm_model_visualization, 64
 - svm_predict, 64
 - svm_predict_probability, 64
 - svm_predict_values, 64
 - svm_print_string, 65
 - svm_save_model, 64
 - svm_set_print_string_function, 64
 - svm_svr_probability, 64
 - svm_train, 64
 - svm_train_one, 64
 - svm_type_table, 65
 - swap, 65
 - TAU, 63
- svm_core.h
 - C_SVC, 54
 - EPSILON_SVR, 54
 - LIBSVM_VERSION, 54
 - LINEAR, 54
 - libsvm_version, 55
 - NU_SVC, 54
 - NU_SVR, 54
 - ONE_CLASS, 54
 - POLY, 54
 - PRECOMPUTED, 54
 - print_svm_samples, 54
 - RBF, 54
 - SIGMOID, 54
 - svm_l_train, 55
 - svm_check_parameter, 54
 - svm_check_probability_model, 54
 - svm_cross_validation, 54

- svm_destroy_param, 55
- svm_free_and_destroy_model, 55
- svm_free_model_content, 55
- svm_get_labels, 55
- svm_get_nr_class, 55
- svm_get_nr_sv, 55
- svm_get_sv_indices, 55
- svm_get_svm_type, 55
- svm_get_svr_probability, 55
- svm_load_model, 55
- svm_model_visualization, 55
- svm_predict, 55
- svm_predict_probability, 55
- svm_predict_values, 55
- svm_save_model, 55
- svm_set_print_string_function, 55
- svm_train, 55
- svm_cross_validation
 - svm_core.cpp, 64
 - svm_core.h, 54
- svm_destroy_param
 - svm_core.cpp, 64
 - svm_core.h, 55
- svm_free_and_destroy_model
 - svm_core.cpp, 64
 - svm_core.h, 55
- svm_free_model_content
 - svm_core.cpp, 64
 - svm_core.h, 55
- svm_get_labels
 - svm_core.cpp, 64
 - svm_core.h, 55
- svm_get_nr_class
 - svm_core.cpp, 64
 - svm_core.h, 55
- svm_get_nr_sv
 - svm_core.cpp, 64
 - svm_core.h, 55
- svm_get_sv_indices
 - svm_core.cpp, 64
 - svm_core.h, 55
- svm_get_svm_type
 - svm_core.cpp, 64
 - svm_core.h, 55
- svm_get_svr_probability
 - svm_core.cpp, 64
 - svm_core.h, 55
- svm_group_classes
 - svm_core.cpp, 64
- svm_i
 - libsvm_i_learn, 17
- svm_i.cpp
 - operator<<, 65
- svm_load_model
 - svm_core.cpp, 64
 - svm_core.h, 55
- svm_model, 38
 - free_sv, 39
 - l, 39
 - label, 39
 - nSV, 39
 - nr_class, 39
 - param, 39
 - probA, 39
 - probB, 39
 - rho, 39
 - SV, 39
 - sv_coef, 39
 - sv_indices, 39
- svm_model_visualization
 - svm_core.cpp, 64
 - svm_core.h, 55
- svm_node, 39
 - operator<<, 39
 - value, 39
- svm_parameter, 40
 - C, 40
 - cache_size, 40
 - coef0, 40
 - degree, 40
 - eps, 40
 - gamma, 40
 - kernel_type, 40
 - nr_weight, 40
 - nu, 40
 - p, 40
 - probability, 40
 - shrinking, 40
 - svm_type, 40
 - weight, 40
 - weight_label, 40
- svm_predict
 - svm_core.cpp, 64
 - svm_core.h, 55
- svm_predict_probability
 - svm_core.cpp, 64
 - svm_core.h, 55
- svm_predict_values
 - svm_core.cpp, 64
 - svm_core.h, 55
- svm_print_string
 - svm_core.cpp, 65
- svm_problem, 41
 - l, 41
 - operator<<, 41
 - x, 41
 - y, 41
- svm_save_model
 - svm_core.cpp, 64
 - svm_core.h, 55
- svm_set_print_string_function
 - svm_core.cpp, 64
 - svm_core.h, 55
- svm_svr_probability
 - svm_core.cpp, 64
- svm_train

- svm_core.cpp, 64
- svm_core.h, 55
- svm_train_one
 - svm_core.cpp, 64
- svm_type
 - svm_parameter, 40
- svm_type_table
 - svm_core.cpp, 65
- swap
 - svm_core.cpp, 65
- swap_index
 - Cache, 12
 - Kernel, 20
 - ONE_CLASS_Q, 23
 - QMatrix, 25
 - SVC_Q, 33
 - SVR_Q, 42
 - Solver, 29
- TAU
 - svm_core.cpp, 63
- target_program
 - config.cpp, 57
 - config.h, 48
- temp_index
 - instrumentation.cpp, 59
- temp_states
 - instrumentation.cpp, 59
- test/1_conj.cpp, 66
- test/2_ex1.cpp, 66
- test/2_f1.cpp, 66
- test/2_f2.cpp, 67
- test/2_z3test.cpp, 67
- test/3_f3.cpp, 67
- theta
 - Equation, 15
- theta0
 - Equation, 15
- traces_num
 - States, 32
- train
 - ML_Algo, 22
 - Perceptron, 24
 - SVM_I, 38
 - SVM, 35
- training_label
 - Perceptron, 25
 - SVM, 36
- training_set
 - Perceptron, 25
 - SVM, 36
- UPBOUND
 - equation.cpp, 57
- UPPER_BOUND
 - Solver, 28
- unset_console_color
 - color.cpp, 56
 - color.h, 47
- unshrink
 - Solver, 30
- update_alpha_status
 - Solver, 29
- upper_bound_n
 - Solver::SolutionInfo, 27
- upper_bound_p
 - Solver::SolutionInfo, 27
- VARS
 - config.h, 47
- value
 - svm_node, 39
- values
 - States, 32
- WHITE
 - color.h, 46
- weight
 - svm_parameter, 40
- weight_label
 - svm_parameter, 40
- x
 - Kernel, 20
 - Solution, 26
 - svm_problem, 41
- x_square
 - Kernel, 20
- y
 - SVC_Q, 33
 - Solver, 30
 - svm_problem, 41
- YELLOW
 - color.h, 46