

Brain-CEMISID

2.0

Generated by Doxygen 1.8.12

Contents

1	Module Index	1
1.1	Modules	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	Module Documentation	7
4.1	Analytical neuron related classes	7
4.1.1	Detailed Description	7
4.1.2	Function Documentation	7
4.1.2.1	solve_ambiguity()	7
4.2	Intentions related classes	9
4.2.1	Detailed Description	9
4.3	Cultural network related classes	10
4.3.1	Detailed Description	10
4.4	Geometric Neural Block classes	11
4.4.1	Detailed Description	11
4.5	BCF classes	12
4.5.1	Detailed Description	12
4.6	Brain-CEMISID kernel	13

4.6.1	Detailed Description	15
4.6.2	Function Documentation	15
4.6.2.1	check()	15
4.6.2.2	erase_all_knowledge()	16
4.6.2.3	feed_internal_state()	16
4.6.2.4	get_desired_state()	17
4.6.2.5	get_hearing_knowledge_in()	17
4.6.2.6	get_hearing_knowledge_out()	18
4.6.2.7	get_internal_state()	18
4.6.2.8	get_sight_knowledge_in()	18
4.6.2.9	get_sight_knowledge_out()	18
4.6.2.10	get_working_domain()	19
4.6.2.11	recognize()	19
4.6.2.12	set_desired_state()	20
4.6.2.13	set_hearing_knowledge_in()	20
4.6.2.14	set_internal_state()	20
4.6.2.15	set_internal_state_in()	22
4.6.2.16	set_sight_knowledge_in()	22
4.6.2.17	set_working_domain()	23
4.7	Relational network related classes	24
4.7.1	Detailed Description	24
4.8	RBF network related classes	25
4.8.1	Detailed Description	27
4.8.2	Function Documentation	28
4.8.2.1	deserialize()	28
4.8.2.2	get_class() [1/2]	28
4.8.2.3	get_class() [2/2]	28
4.8.2.4	get_distance()	29

4.8.2.5	get_index_ready_to_learn()	29
4.8.2.6	get_knowledge()	29
4.8.2.7	get_last_learned_id()	30
4.8.2.8	get_neuron_count()	30
4.8.2.9	get_pattern() [1/2]	30
4.8.2.10	get_pattern() [2/2]	30
4.8.2.11	get_radius()	31
4.8.2.12	get_rneurons_ids()	31
4.8.2.13	get_set() [1/2]	32
4.8.2.14	get_set() [2/2]	32
4.8.2.15	get_state()	33
4.8.2.16	increase_radius_by()	33
4.8.2.17	is_degraded()	33
4.8.2.18	is_hit()	34
4.8.2.19	is_member()	34
4.8.2.20	learn() [1/3]	35
4.8.2.21	learn() [2/3]	35
4.8.2.22	learn() [3/3]	36
4.8.2.23	learn_hearing()	37
4.8.2.24	learn_sight()	37
4.8.2.25	recognize() [1/2]	38
4.8.2.26	recognize() [2/2]	39
4.8.2.27	recognize_hearing()	39
4.8.2.28	recognize_sight()	41
4.8.2.29	reduce_radius_by()	41
4.8.2.30	reduce_radius_last_distance()	42
4.8.2.31	save()	42
4.8.2.32	serialize()	43
4.8.2.33	set_class()	43
4.8.2.34	set_pattern()	44
4.8.2.35	set_radius()	44
4.8.2.36	set_set()	45

5	Class Documentation	47
5.1	geometric_neural_block.AdditionStructure Class Reference	47
5.1.1	Detailed Description	48
5.1.2	Constructor & Destructor Documentation	48
5.1.2.1	<code>__init__()</code>	48
5.1.3	Member Data Documentation	48
5.1.3.1	<code>carry_over</code>	48
5.1.3.2	<code>index</code>	49
5.2	analytical_neuron.AnalyticalNeuron Class Reference	49
5.2.1	Detailed Description	49
5.3	internal_state.BiologyCultureFeelings Class Reference	50
5.3.1	Detailed Description	52
5.3.2	Member Function Documentation	52
5.3.2.1	<code>get_biology()</code>	52
5.3.2.2	<code>get_culture()</code>	53
5.3.2.3	<code>get_feelings()</code>	53
5.3.2.4	<code>get_state()</code>	54
5.3.2.5	<code>set_biology()</code>	54
5.3.2.6	<code>set_culture()</code>	55
5.3.2.7	<code>set_feelings()</code>	56
5.3.2.8	<code>set_state()</code>	56
5.4	prev_main.BrainInterface Class Reference	58
5.5	conscious_decisions_block.ConsciousDecisionsBlock Class Reference	61
5.5.1	Detailed Description	62
5.5.2	Member Function Documentation	62
5.5.2.1	<code>feedback()</code>	62
5.5.2.2	<code>get_decision()</code>	63
5.5.2.3	<code>get_desired_state()</code>	64

5.5.2.4	get_inputs()	65
5.5.2.5	get_internal_state()	65
5.5.2.6	get_last_decision_type()	65
5.5.2.7	set_desired_state()	65
5.5.2.8	set_inputs()	66
5.5.2.9	set_internal_state()	66
5.5.2.10	training()	67
5.6	cultural_network.CulturalGroup Class Reference	68
5.6.1	Detailed Description	69
5.6.2	Member Function Documentation	69
5.6.2.1	bip()	69
5.6.2.2	check()	69
5.6.2.3	clack()	70
5.6.2.4	contains()	71
5.6.2.5	get_tail_knowledge()	71
5.6.2.6	learn()	72
5.7	cultural_network.CulturalNetwork Class Reference	72
5.7.1	Detailed Description	75
5.7.2	Member Function Documentation	75
5.7.2.1	bip()	75
5.7.2.2	check()	75
5.7.2.3	clack()	76
5.7.2.4	deserialize()	76
5.7.2.5	get_tail_knowledge()	76
5.7.2.6	serialize()	77
5.8	cultural_network.CulturalNeuron Class Reference	77
5.8.1	Detailed Description	79
5.9	decision_by_prediction_block.DecisionByPredictionBlock Class Reference	79

5.9.1	Detailed Description	81
5.9.2	Member Function Documentation	81
5.9.2.1	get_desired_state()	81
5.9.2.2	get_distances()	82
5.9.2.3	get_inputs()	82
5.9.2.4	get_output()	82
5.9.2.5	get_predicted_outcomes()	83
5.9.2.6	remodel_predictive_net()	83
5.9.2.7	set_desired_state()	84
5.9.2.8	set_inputs()	84
5.9.2.9	set_internal_state()	85
5.10	decisions_block.DecisionsBlock Class Reference	85
5.10.1	Detailed Description	87
5.10.2	Member Function Documentation	87
5.10.2.1	get_output_memory()	87
5.10.2.2	set_desired_state()	87
5.10.2.3	set_input_memories()	88
5.10.2.4	set_internal_state()	88
5.11	episodic_memories.EpisodicMemoriesBlock Class Reference	89
5.11.1	Detailed Description	92
5.11.2	Member Function Documentation	92
5.11.2.1	deserialize()	92
5.11.2.2	retrieve_exact_memory()	92
5.11.2.3	retrieve_memories()	93
5.11.2.4	serialize()	93
5.12	geometric_neural_block.GeometricNeuralBlock Class Reference	94
5.12.1	Detailed Description	95
5.12.2	Member Function Documentation	95

5.12.2.1	bip()	95
5.12.2.2	deserialize()	96
5.12.2.3	get_add_operator()	96
5.12.2.4	get_addition_result()	97
5.12.2.5	get_equal_sign()	97
5.12.2.6	get_operation()	98
5.12.2.7	serialize()	98
5.12.2.8	set_add_operator()	98
5.12.2.9	set_equal_sign()	98
5.12.2.10	set_operation()	99
5.12.2.11	set_zero()	99
5.13	main.IntentionsInterface Class Reference	100
5.14	internal_state.InternalState Class Reference	103
5.14.1	Detailed Description	106
5.14.2	Member Function Documentation	106
5.14.2.1	average_biology()	106
5.14.2.2	average_culture()	107
5.14.2.3	average_feelings()	107
5.14.2.4	average_state()	108
5.14.2.5	biology_alarm()	108
5.14.2.6	biology_up_alarm()	109
5.15	kernel_braincemisid.KernelBrainCemisid Class Reference	109
5.15.1	Detailed Description	112
5.16	multiclass_single_layer_network.MulticlassSingleLayerNetwork Class Reference	113
5.16.1	Constructor & Destructor Documentation	114
5.16.1.1	__init__()	114
5.16.2	Member Function Documentation	114
5.16.2.1	get_inputs()	114

5.16.2.2	get_learning_rate()	114
5.16.2.3	get_outputs()	115
5.16.2.4	set_activation_function()	115
5.16.2.5	set_inputs()	115
5.16.2.6	set_learning_rate()	116
5.16.2.7	training()	116
5.16.2.8	update_weights()	117
5.17	main.MyGroupPaintWidget Class Reference	118
5.18	main.MyPaintApp Class Reference	119
5.19	main.MyPaintElement Class Reference	121
5.20	main.MyPaintWidget Class Reference	123
5.20.1	Member Function Documentation	125
5.20.1.1	draw_pattern()	125
5.21	neuron.Neuron Class Reference	125
5.21.1	Member Function Documentation	126
5.21.1.1	has_knowledge()	126
5.22	object Class Reference	128
5.23	geometric_neural_block.OrderNeuron Class Reference	128
5.23.1	Detailed Description	130
5.24	geometric_neural_block.QuantityNeuron Class Reference	130
5.24.1	Detailed Description	132
5.25	geometric_neural_block.QuantityOrderGroup Class Reference	132
5.25.1	Detailed Description	133
5.25.2	Member Function Documentation	133
5.25.2.1	compare()	133
5.26	geometric_neural_block.QuantityOrderNetwork Class Reference	134
5.26.1	Detailed Description	135
5.26.2	Member Function Documentation	135

5.26.2.1	get_bip_count()	135
5.27	main.RbfCardWidget Class Reference	135
5.28	sensory_neural_block.RbfKnowledge Class Reference	137
5.28.1	Detailed Description	138
5.29	sensory_neural_block.RbfNetwork Class Reference	138
5.29.1	Detailed Description	139
5.30	sensory_neural_block.RbfNeuron Class Reference	140
5.30.1	Detailed Description	142
5.31	rel_network.RelKnowledge Class Reference	143
5.31.1	Detailed Description	144
5.31.2	Member Function Documentation	144
5.31.2.1	get_h_id()	144
5.31.2.2	get_s_id()	144
5.31.2.3	get_weight()	144
5.31.2.4	increase_weight()	145
5.31.2.5	is_equal()	145
5.31.2.6	set_h_id()	145
5.31.2.7	set_s_id()	146
5.31.2.8	set_weight()	146
5.32	rel_network.RelNetwork Class Reference	147
5.32.1	Detailed Description	148
5.32.2	Constructor & Destructor Documentation	148
5.32.2.1	__init__()	148
5.32.3	Member Function Documentation	148
5.32.3.1	deserialize()	148
5.32.3.2	get_hearing_rels()	149
5.32.3.3	get_neuron_count()	149
5.32.3.4	get_sight_rels()	149

5.32.3.5	learn()	150
5.32.3.6	serialize()	150
5.33	rel_network.RelNeuron Class Reference	150
5.33.1	Detailed Description	153
5.33.2	Member Function Documentation	153
5.33.2.1	get_h_id()	153
5.33.2.2	get_knowledge()	154
5.33.2.3	get_s_id()	154
5.33.2.4	get_weight()	155
5.33.2.5	has_ids()	155
5.33.2.6	learn()	156
5.33.2.7	recognize_hearing()	156
5.33.2.8	recognize_sight()	157
5.33.2.9	set_h_id()	157
5.33.2.10	set_knowledge()	158
5.33.2.11	set_s_id()	159
5.34	sensory_neural_block.SensoryNeuralBlock Class Reference	159
5.34.1	Detailed Description	161
5.35	main.SetInternalVariableWidget Class Reference	162
5.36	main.ShowInternalVariableWidget Class Reference	164
5.37	unconscious_filtering_block.UnconsciousFilteringBlock Class Reference	166
5.37.1	Member Function Documentation	167
5.37.1.1	get_inputs()	167
5.37.1.2	get_outputs()	167
5.37.1.3	set_desired_state()	168
5.37.1.4	set_inputs()	168
5.37.1.5	set_internal_state()	168

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Analytical neuron related classes	7
Intentions related classes	9
Cultural network related classes	10
Geometric Neural Block classes	11
BCF classes	12
Brain-CEMISID kernel	13
Relational network related classes	24
RBF network related classes	25

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

geometric_neural_block.AdditionStructure	47
analytical_neuron.AnalyticalNeuron	49
internal_state.BiologyCultureFeelings	50
internal_state.InternalState	103
conscious_decisions_block.ConsciousDecisionsBlock	61
cultural_network.CulturalGroup	68
cultural_network.CulturalNetwork	72
episodic_memories.EpisodicMemoriesBlock	89
decision_by_prediction_block.DecisionByPredictionBlock	79
decisions_block.DecisionsBlock	85
geometric_neural_block.GeometricNeuralBlock	94
kernel_braincemisid.KernelBrainCemisid	109
multiclass_single_layer_network.MulticlassSingleLayerNetwork	113
object	128
neuron.Neuron	125
cultural_network.CulturalNeuron	77
geometric_neural_block.OrderNeuron	128
geometric_neural_block.QuantityNeuron	130
rel_network.RelNeuron	150
sensory_neural_block.RbfNeuron	140
geometric_neural_block.QuantityOrderGroup	132
geometric_neural_block.QuantityOrderNetwork	134
sensory_neural_block.RbfKnowledge	137
sensory_neural_block.RbfNetwork	138
rel_network.RelKnowledge	143
rel_network.RelNetwork	147
sensory_neural_block.SensoryNeuralBlock	159
unconscious_filtering_block.UnconsciousFilteringBlock	166
App	
main.MyPaintApp	119

GridLayout	
main.IntentionsInterface	100
main.MyGroupPaintWidget	118
main.MyPaintWidget	123
main.RbfCardWidget	135
main.SetInternalVariableWidget	162
main.ShowInternalVariableWidget	164
prev_main.BrainInterface	58
Widget	
main.MyPaintElement	121

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

geometric_neural_block.AdditionStructure	
The addition structure class provides a decimal numeric system neural representation	47
analytical_neuron.AnalyticalNeuron	
Analytical neuron	49
internal_state.BiologyCultureFeelings	
BiologyCultureFeelings models the state of an entity by using a three elements vector which values go from 0 to 1 and correspond to the Biology, Culture and Feelings situation of the entity	50
prev_main.BrainInterface	58
conscious_decisions_block.ConsciousDecisionsBlock	
The Conscious Decisions Block is in charge of 'rationally' evaluating current internal state and goals in order to make decisions	61
cultural_network.CulturalGroup	
Cultural group	68
cultural_network.CulturalNetwork	
Cultural network	72
cultural_network.CulturalNeuron	
Cultural neuron	77
decision_by_prediction_block.DecisionByPredictionBlock	
The DecisionByPredictionBlock is a class aimed at modeling how decisions can be made through prediction	79
decisions_block.DecisionsBlock	
The DecisionsBlock takes as input a set of memories (CulturalGroup) and gives as output one of them from which the decision can be inferred	85
episodic_memories.EpisodicMemoriesBlock	
The EpisodicMemoriesBlock is a specialization of CulturalNetwork from which a set of CulturalGroup s can be retrieved given a set of triggers, just as in humans	89
geometric_neural_block.GeometricNeuralBlock	
Class that envelopes all neural geometries that represent concepts in the brain	94
main.IntentionsInterface	100
internal_state.InternalState	
His class represents a very simplified version of an entity's internal state	103

kernel_braincemisid.KernelBrainCemisid	
The KernelBrainCemisid is a major module that envelopes and coordinates interaction between all other project modules (Except for the interface, which obviously must not be part of the kernel, but use the kernel)	109
multiclass_single_layer_network.MulticlassSingleLayerNetwork	113
main.MyGroupPaintWidget	118
main.MyPaintApp	119
main.MyPaintElement	121
main.MyPaintWidget	123
neuron.Neuron	125
object	128
geometric_neural_block.OrderNeuron	
The QuantityNeuron class is a kind of neuron whose position in a QuantityOrderNetwork signals certain ordinality	128
geometric_neural_block.QuantityNeuron	
The QuantityNeuron class is a kind of neuron that signals the cardinality of its relation in a Quantity↔OrderGroup	130
geometric_neural_block.QuantityOrderGroup	
A QuantityOrderGroup is a pair composed of an OrderNeuron and a QuantityNeuron	132
geometric_neural_block.QuantityOrderNetwork	
A set of QuantityOrderGroup instances that act together in order to store Order and Quantity information	134
main.RbfCardWidget	135
sensory_neural_block.RbfKnowledge	
RBF knowledge	137
sensory_neural_block.RbfNetwork	
RBF Neural Network	138
sensory_neural_block.RbfNeuron	
Neuron that stores RbfKnowledge	140
rel_network.RelKnowledge	
Relational knowledge is a 3-tuple that relate a sight RbfNeuron id, a hearing RbfNeuron id and a weight	143
rel_network.RelNetwork	
Relational network	147
rel_network.RelNeuron	
Relational neuron	150
sensory_neural_block.SensoryNeuralBlock	
Sensory Neural Block Stores sight and hearing RbfNetworks	159
main.SetInternalVariableWidget	162
main.ShowInternalVariableWidget	164
unconscious_filtering_block.UnconsciousFilteringBlock	166

Chapter 4

Module Documentation

4.1 Analytical neuron related classes

The Analytical neuron is a class that analyzes RelKnowledge instances to solve ambiguities.

Classes

- class [analytical_neuron.AnalyticalNeuron](#)
Analytical neuron.

Functions

- def [analytical_neuron.AnalyticalNeuron.__init__](#) (self)
The constructor.
- def [analytical_neuron.AnalyticalNeuron.solve_ambiguity](#) (self, rel_knowledge_v)
Solve ambiguities.

4.1.1 Detailed Description

The Analytical neuron is a class that analyzes RelKnowledge instances to solve ambiguities.

4.1.2 Function Documentation

4.1.2.1 solve_ambiguity()

```
def analytical_neuron.AnalyticalNeuron.solve_ambiguity (  
    self,  
    rel_knowledge_v )
```

Solve ambiguities.

Parameters

<i>rel_knowledge</i> _↔ <i>_v</i>	Vector of relational knowledge
--	--------------------------------

Return values

<i>h</i> _↔ <i>_id</i>	Integer. Hearing id of maximum weight relation
-------------------------------------	--

4.2 Intentions related classes

These are the set of classes that support decision making for intentions creation.

Classes

- class [conscious_decisions_block.ConsciousDecisionsBlock](#)
The Conscious Decisions Block is in charge of 'rationally' evaluating current internal state and goals in order to make decisions.
- class [decision_by_prediction_block.DecisionByPredictionBlock](#)
The [DecisionByPredictionBlock](#) is a class aimed at modeling how decisions can be made through prediction.
- class [decisions_block.DecisionsBlock](#)
The [DecisionsBlock](#) takes as input a set of memories ([CulturalGroup](#)) and gives as output one of them from which the decision can be inferred.
- class [episodic_memories.EpisodicMemoriesBlock](#)
The [EpisodicMemoriesBlock](#) is a specialization of [CulturalNetwork](#) from which a set of [CulturalGroup](#) s can be retrieved given a set of triggers, just as in humans.
- class [multiclass_single_layer_network.MulticlassSingleLayerNetwork](#)
- class [unconscious_filtering_block.UnconsciousFilteringBlock](#)

4.2.1 Detailed Description

These are the set of classes that support decision making for intentions creation.

Single layer feedforward network with multiple outputs for input classification.

Episodic memories block.

Decisions block.

Unconscious filtering block.

4.3 Cultural network related classes

Relational network related classes are a group of classes that represent cultural neurons, groups and networks.

Classes

- class `cultural_network.CulturalNeuron`
Cultural neuron.
- class `cultural_network.CulturalGroup`
Cultural group.
- class `cultural_network.CulturalNetwork`
Cultural network.

4.3.1 Detailed Description

Relational network related classes are a group of classes that represent cultural neurons, groups and networks.

4.4 Geometric Neural Block classes

Geometric Neural Block classes are a group of classes that neurally represent concepts in the brain.

Classes

- class [geometric_neural_block.QuantityNeuron](#)
The [QuantityNeuron](#) class is a kind of neuron that signals the cardinality of its relation in a [QuantityOrderGroup](#).
- class [geometric_neural_block.OrderNeuron](#)
The [QuantityNeuron](#) class is a kind of neuron whose position in a [QuantityOrderNetwork](#) signals certain ordinality.
- class [geometric_neural_block.QuantityOrderGroup](#)
A [QuantityOrderGroup](#) is a pair composed of an [OrderNeuron](#) and a [QuantityNeuron](#).
- class [geometric_neural_block.QuantityOrderNetwork](#)
A set of [QuantityOrderGroup](#) instances that act together in order to store Order and Quantity information.
- class [geometric_neural_block.AdditionStructure](#)
The addition structure class provides a decimal numeric system neural representation.
- class [geometric_neural_block.GeometricNeuralBlock](#)
Class that envelopes all neural geometries that represent concepts in the brain.

4.4.1 Detailed Description

Geometric Neural Block classes are a group of classes that neurally represent concepts in the brain.

For instance, the [QuantityOrderGroup](#) models a relation between the concepts of quantity and order.

4.5 BCF classes

There are two BCF classes.

Classes

- class [internal_state.BiologyCultureFeelings](#)
[BiologyCultureFeelings](#) models the state of an entity by using a three elements vector which values go from 0 to 1 and correspond to the Biology, Culture and Feelings situation of the entity.
- class [internal_state.InternalState](#)
this class represents a very simplified version of an entity's internal state.

4.5.1 Detailed Description

There are two BCF classes.

The first one, [BiologyCultureFeelings](#) models the state of an entity by using a three elements vector which values go from 0 to 1 and correspond to the Biology, Culture and Feelings situation of the entity. The second one, is the [InternalState](#) class that inherits from the [BiologyCultureFeelings](#) class and provides averaging methods to modify a given state.

4.6 Brain-CEMISID kernel

This is the project kernel where all modules interact.

Classes

- class `kernel_braincemisid.KernelBrainCemisid`

The `KernelBrainCemisid` is a major module that envelopes and coordinates interaction between all other project modules (Except for the interface, which obviously must not be part of the kernel, but use the kernel)

Functions

- def `kernel_braincemisid.KernelBrainCemisid.__init__` (self)
Kernel constructor.
- def `kernel_braincemisid.KernelBrainCemisid.set_working_domain` (self, domain)
Sets a working domain for the bbcc protocol.
- def `kernel_braincemisid.KernelBrainCemisid.get_working_domain` (self)
Get bbcc protocol working domain.
- def `kernel_braincemisid.KernelBrainCemisid.set_sight_knowledge_in` (self, knowledge)
Set sight knowledge.
- def `kernel_braincemisid.KernelBrainCemisid.get_sight_knowledge_in` (self)
Get sight knowledge.
- def `kernel_braincemisid.KernelBrainCemisid.set_hearing_knowledge_in` (self, knowledge)
Set hearing knowledge.
- def `kernel_braincemisid.KernelBrainCemisid.get_hearing_knowledge_in` (self)
Get hearing knowledge.
- def `kernel_braincemisid.KernelBrainCemisid.get_sight_knowledge_out` (self)
Get output sight knowledge (Thinking)
- def `kernel_braincemisid.KernelBrainCemisid.get_hearing_knowledge_out` (self)
Get output hearing knowledge (Thinking)
- def `kernel_braincemisid.KernelBrainCemisid.set_internal_state_in` (self, states_vector)
Set internal state related to an input.
- def `kernel_braincemisid.KernelBrainCemisid.set_internal_state` (self, states_vector)
Set internal state (Biology, Culture and Feelings)
- def `kernel_braincemisid.KernelBrainCemisid.get_internal_state` (self)
Get internal state.
- def `kernel_braincemisid.KernelBrainCemisid.feed_internal_state` (self, states_vector)
Get internal state resulting from an experience and take the average with the current internal state.
- def `kernel_braincemisid.KernelBrainCemisid.set_desired_state` (self, states_vector)
Set desired state (Biology, Culture and Feelings)
- def `kernel_braincemisid.KernelBrainCemisid.get_desired_state` (self)
Get desired state.
- def `kernel_braincemisid.KernelBrainCemisid.disable_bbcc` (self)
Disable bbcc protocol so that Check and Clack can be used as standalone actions.
- def `kernel_braincemisid.KernelBrainCemisid.bum` (self)

- *Start bbcc protocol.*
- def [kernel_braincemisid.KernelBrainCemisid.bip](#) (self)
 - Bip part of bbcc protocol (See bbcc protocol description)*
- def [kernel_braincemisid.KernelBrainCemisid.check](#) (self)
 - Check if there is knowledge associated with the given sequence of patterns from bbcc protocol when self._enable_bbcc is True.*
- def [kernel_braincemisid.KernelBrainCemisid.clack](#) (self)
 - Execute clack action of bbcc protocol when self._enable_bbcc is True or Learn a piece of RbfKnowledge coming from the senses when self._enable_bbcc is False.*
- def **kernel_braincemisid.KernelBrainCemisid.is_null_pattern** (pattern)
- def [kernel_braincemisid.KernelBrainCemisid.recognize](#) (self)
 - Recognize either hearing or sight patterns.*
- def [kernel_braincemisid.KernelBrainCemisid.hearing_recognize](#) (self)
 - Recognize hearing pattern.*
- def [kernel_braincemisid.KernelBrainCemisid.sight_recognize](#) (self)
 - Recognize sight pattern.*
- def [kernel_braincemisid.KernelBrainCemisid.learn](#) (self)
 - Learn patterns.*
- def [kernel_braincemisid.KernelBrainCemisid.erase_all_knowledge](#) (self)
 - Erase all knowlege.*
- def **kernel_braincemisid.KernelBrainCemisid.set_add_operator** (self)
- def **kernel_braincemisid.KernelBrainCemisid.set_equal_sign** (self)
- def [kernel_braincemisid.KernelBrainCemisid.set_zero](#) (self)
 - Set some already learned pattern as the zero number.*

Variables

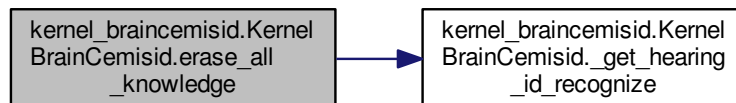
- **kernel_braincemisid.KernelBrainCemisid.snb**
- **kernel_braincemisid.KernelBrainCemisid.rnb**
- **kernel_braincemisid.KernelBrainCemisid.analytical_n**
- **kernel_braincemisid.KernelBrainCemisid.am_net**
- **kernel_braincemisid.KernelBrainCemisid.gnb**
- **kernel_braincemisid.KernelBrainCemisid.syllables_net**
- **kernel_braincemisid.KernelBrainCemisid.words_net**
- **kernel_braincemisid.KernelBrainCemisid.ss_rnb**
- [kernel_braincemisid.KernelBrainCemisid.episodic_memory](#)
- *INTENTIONS #####*
- **kernel_braincemisid.KernelBrainCemisid.decisions_block**
- **kernel_braincemisid.KernelBrainCemisid.internal_state**
- **kernel_braincemisid.KernelBrainCemisid.desired_state**
- **kernel_braincemisid.KernelBrainCemisid.s_knowledge_out**
- **kernel_braincemisid.KernelBrainCemisid.h_knowledge_out**
- **kernel_braincemisid.KernelBrainCemisid.s_knowledge_in**
- **kernel_braincemisid.KernelBrainCemisid.h_knowledge_in**
- [kernel_braincemisid.KernelBrainCemisid.state](#)
- *INTENTIONS ##### Get memory related to hearing id.*

4.6.2.2 erase_all_knowledge()

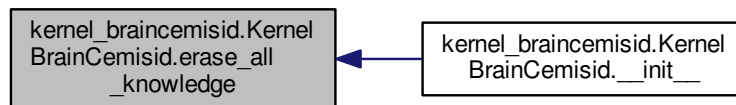
```
def kernel_braincemisid.KernelBrainCemisid.erase_all_knowledge (
    self )
```

Erase all knowlege.

Get to a *tabula rasa* state. Here is the call graph for this function:



Here is the caller graph for this function:



4.6.2.3 feed_internal_state()

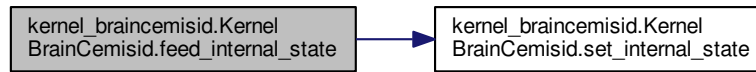
```
def kernel_braincemisid.KernelBrainCemisid.feed_internal_state (
    self,
    states_vector )
```

Get internal state resulting from an experience and take the average with the current internal state.

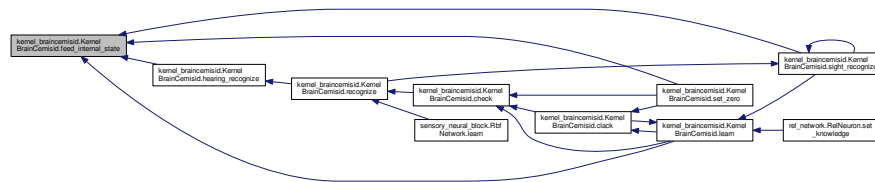
Parameters

<code>states_vector</code>	Floats vector, for example <code>[0.7, 0.5, 0.3]</code>
----------------------------	---

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.2.4 get_desired_state()

```
def kernel_braincemisid.KernelBrainCemisid.get_desired_state (
    self )
```

Get desired state.

Return values

<i>desired_state</i>	InternalState
----------------------	---------------

4.6.2.5 get_hearing_knowledge_in()

```
def kernel_braincemisid.KernelBrainCemisid.get_hearing_knowledge_in (
    self )
```

Get hearing knowledge.

Return values

<i>knowledge</i>	RbfKnowledge
------------------	--------------

4.6.2.6 get_hearing_knowledge_out()

```
def kernel_braincemisid.KernelBrainCemisid.get_hearing_knowledge_out (
    self )
```

Get output hearing knowledge (Thinking)

Return values

<i>knowledge</i>	RbfKnowledge vector
------------------	---------------------

4.6.2.7 get_internal_state()

```
def kernel_braincemisid.KernelBrainCemisid.get_internal_state (
    self )
```

Get internal state.

Return values

<i>internal_state</i>	InternalState.
-----------------------	----------------

4.6.2.8 get_sight_knowledge_in()

```
def kernel_braincemisid.KernelBrainCemisid.get_sight_knowledge_in (
    self )
```

Get sight knowledge.

Return values

<i>knowledge</i>	RbfKnowledge
------------------	--------------

4.6.2.9 get_sight_knowledge_out()

```
def kernel_braincemisid.KernelBrainCemisid.get_sight_knowledge_out (
    self )
```

Get output sight knowledge (Thinking)

Return values

<i>knowledge</i>	RbfKnowledge vector
------------------	---------------------

4.6.2.10 get_working_domain()

```
def kernel_braincemisid.KernelBrainCemisid.get_working_domain (
    self )
```

Get bbcc protocol working domain.

Return values

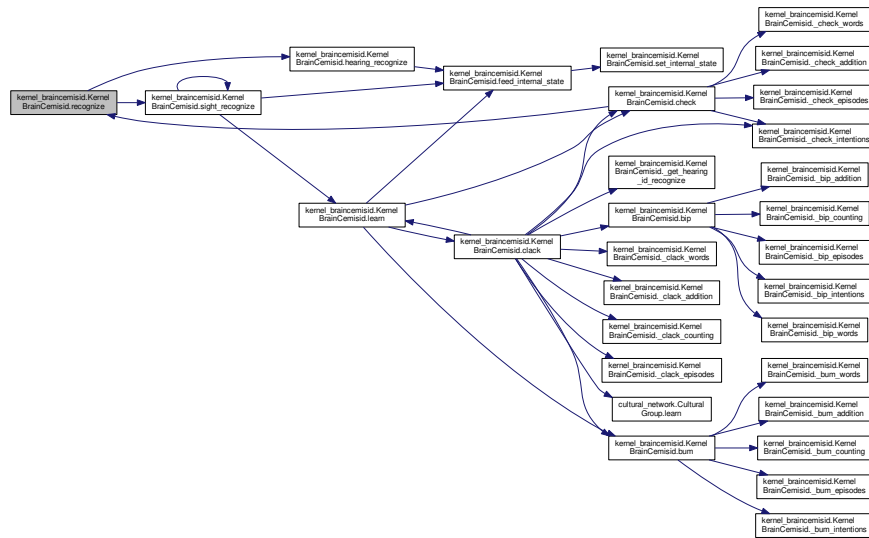
<i>working_domain</i>	enum { "READING", "ADDITION", "COUNTING", "EPISODES", "INTENTIONS" }
-----------------------	--

4.6.2.11 recognize()

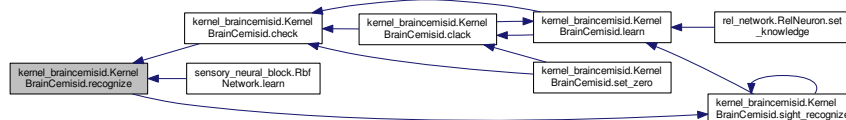
```
def kernel_braincemisid.KernelBrainCemisid.recognize (
    self )
```

Recognize either hearing or sight patterns.

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.2.12 set_desired_state()

```
def kernel_braincemisid.KernelBrainCemisid.set_desired_state (
    self,
    states_vector )
```

Set desired state (Biology, Culture and Feelings)

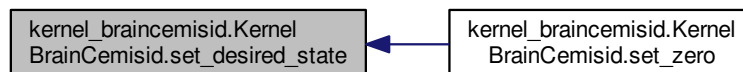
Parameters

<i>states_vector</i>	Floats vector: [Biology, Culture, Feelings]. All components are in the real interval [0,1]
----------------------	--

Return values

<i>result</i>	Boolean. True if state properly set, False in any other case.
---------------	---

Here is the caller graph for this function:



4.6.2.13 set_hearing_knowledge_in()

```
def kernel_braincemisid.KernelBrainCemisid.set_hearing_knowledge_in (
    self,
    knowledge )
```

Set hearing knowledge.

Parameters

<i>knowledge</i>	RbfKnowledge
------------------	--------------

4.6.2.14 set_internal_state()

```
def kernel_braincemisid.KernelBrainCemisid.set_internal_state (
    self,
    states_vector )
```


Set internal state (Biology, Culture and Feelings)

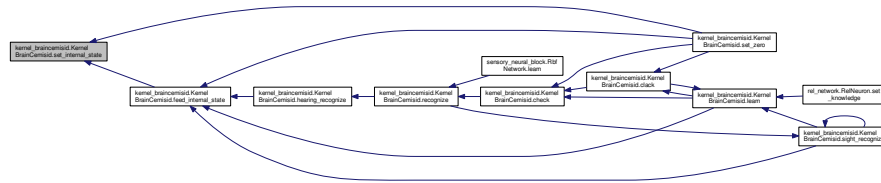
Parameters

<i>states_vector</i>	three components vector: [Biology, Culture, Feelings]. All components are in the real interval [0,1]
----------------------	--

Returns

: True if state properly set, False in any other case.

Here is the caller graph for this function:

**4.6.2.15 set_internal_state_in()**

```
def kernel_braincemisid.KernelBrainCemisid.set_internal_state_in (
    self,
    states_vector )
```

Set internal state related to an input.

For example, an apple would be related to something like $[B=1, C=0, F=0]$ (Note that this is not the machine's internal state)

Parameters

<i>states_vector</i>	Floats vector. For example, $[1.0, 0.1, 0.1]$
----------------------	---

4.6.2.16 set_sight_knowledge_in()

```
def kernel_braincemisid.KernelBrainCemisid.set_sight_knowledge_in (
    self,
    knowledge )
```

Set sight knowledge.

Parameters

<i>knowledge</i>	RbfKnowledge
------------------	--------------

4.6.2.17 `set_working_domain()`

```
def kernel_braincemisid.KernelBrainCemisid.set_working_domain (
    self,
    domain )
```

Sets a working domain for the bbcc protocol.

It could be either "READING", "ADDITION", "COUNTING", "EPISODES" or "INTENTIONS"

Parameters

<i>domain</i>	enum { "READING", "ADDITION", "COUNTING", "EPISODES", "INTENTIONS" }
---------------	--

4.7 Relational network related classes

Relational network related classes are a group of classes that represent relational knowledge, neurons and networks.

Classes

- class `rel_network.RelKnowledge`
Relational knowledge is a 3-tuple that relate a sight RbfNeuron id, a hearing RbfNeuron id and a weight.
- class `rel_network.RelNeuron`
Relational neuron.
- class `rel_network.RelNetwork`
Relational network.

4.7.1 Detailed Description

Relational network related classes are a group of classes that represent relational knowledge, neurons and networks.

4.8 RBF network related classes

RBF network related classes are a group of classes that represent and use RBF knowledge, neurons and networks.

Classes

- class `sensory_neural_block.RbfKnowledge`
RBF knowledge.
- class `sensory_neural_block.RbfNeuron`
Neuron that stores `RbfKnowledge`.
- class `sensory_neural_block.RbfNetwork`
RBF Neural Network.
- class `sensory_neural_block.SensoryNeuralBlock`
Sensory Neural Block Stores sight and hearing `RbfNetworks`.

Functions

- def `sensory_neural_block.RbfKnowledge.__init__` (self, rbf_pattern, rbf_class, rbf_set="NoSet")
The constructor.
- def `sensory_neural_block.RbfKnowledge.set_pattern` (self, pattern)
Set pattern.
- def `sensory_neural_block.RbfKnowledge.set_class` (self, rbf_class)
Set pattern class.
- def `sensory_neural_block.RbfKnowledge.set_set` (self, rbf_set)
Set pattern set.
- def `sensory_neural_block.RbfKnowledge.get_pattern` (self)
Get stored pattern.
- def `sensory_neural_block.RbfKnowledge.get_class` (self)
Get stored pattern class.
- def `sensory_neural_block.RbfKnowledge.get_set` (self)
Get stored pattern set.
- def `sensory_neural_block.RbfKnowledge.calc_manhattan_distance` (self, pattern_or_knowledge)
- def `sensory_neural_block.RbfNeuron.__init__` (self)
Class constructor.
- def `sensory_neural_block.RbfNeuron.is_member` (self, test_set)
Returns whether neuron is member of the set.
- def `sensory_neural_block.RbfNeuron.set_radius` (self, radius)
Sets neuron radius.
- def `sensory_neural_block.RbfNeuron.get_radius` (self)
Get neuron radius.
- def `sensory_neural_block.RbfNeuron.get_class` (self)
Get class of stored `RbfKnowledge` instance.
- def `sensory_neural_block.RbfNeuron.get_set` (self)
Get set of stored `RbfKnowledge` instance.
- def `sensory_neural_block.RbfNeuron.get_pattern` (self)

- Get pattern of stored *RbfKnowledge* instance.

 - def `sensory_neural_block.RbfNeuron.is_hit` (self)
 - Return True if last call to *recognize()* was a hit and False in any other case.
 - def `sensory_neural_block.RbfNeuron.learn` (self, knowledge)
 - Learns a new piece of knowledge.
 - def `sensory_neural_block.RbfNeuron.recognize` (self, pattern)
 - Recognize a piece of knowledge.
 - def `sensory_neural_block.RbfNeuron.get_distance` (self)
 - Get distance to last instance or *RbfKnowledge* pattern that tried to be recognized.
 - def `sensory_neural_block.RbfNeuron.reduce_radius_last_distance` (self)
 - Reduce radius by las recognition process's distance.
 - def `sensory_neural_block.RbfNeuron.reduce_radius_by` (self, value)
 - Reduce neuron radius by certain amount.
 - def `sensory_neural_block.RbfNeuron.increase_radius_by` (self, value)
 - Increase neuron radius by certain amount.
 - def `sensory_neural_block.RbfNeuron.is_degraded` (self)
 - Return whether neuron is degraded.
 - def `sensory_neural_block.RbfNetwork.__init__` (self, neuron_count)
 - Class constructor, takes 'neuron_count' as parameter for setting network size.
 - def `sensory_neural_block.RbfNetwork.get_neuron_count` (self)
 - get number of neurons in network
 - def `sensory_neural_block.RbfNetwork.recognize` (self, pattern)
 - Recognize a given pattern.
 - def `sensory_neural_block.RbfNetwork.get_knowledge` (self)
 - Get *RbfKnowledge* related to last recognized pattern.
 - def `sensory_neural_block.RbfNetwork.get_state` (self)
 - Get network state.
 - def `sensory_neural_block.RbfNetwork.learn` (self, knowledge)
 - Learn an instance of *RbfKnowledge*.
 - def `sensory_neural_block.RbfNetwork.get_rneurons_ids` (self)
 - Get ids of recognizing set neurons.
 - def `sensory_neural_block.RbfNetwork.get_last_learned_id` (self)
 - Get id of neuron affected in the last learning process.
 - def `sensory_neural_block.RbfNetwork.get_index_ready_to_learn` (self)
 - Get index of ready-to-learn neuron.
 - def `sensory_neural_block.RbfNetwork.serialize` (cls, obj, name)
 - Serialize object and store in given file.
 - def `sensory_neural_block.RbfNetwork.deserialize` (cls, name)
 - Deserialize object stored in given file.
 - def `sensory_neural_block.SensoryNeuralBlock.__init__` (self, sight_snb_file="NoFile", hearing_snb_file="NoFile")
 - The constructor.
 - def `sensory_neural_block.SensoryNeuralBlock.recognize_sight` (self, pattern)
 - Recognize a sight pattern.
 - def `sensory_neural_block.SensoryNeuralBlock.recognize_hearing` (self, pattern)
 - Recognize a hearing pattern.
 - def `sensory_neural_block.SensoryNeuralBlock.learn_hearing` (self, knowledge)
 - Learn a hearing pattern.

- def `sensory_neural_block.SensoryNeuralBlock.learn_sight` (self, knowledge)
Learn a visual pattern.
- def `sensory_neural_block.SensoryNeuralBlock.learn` (self, knowledge_h, pattern_s)
Learn a pair of hearing and sight patterns relating both pieces of knowledge through the hearing id stored as the sight knowledge's pattern.
- def `sensory_neural_block.SensoryNeuralBlock.get_last_learned_ids` (self)
Return a 2-tuple of integeres representing the ids of hearing and sight neurons that learned in the last learn_sight process.
- def `sensory_neural_block.SensoryNeuralBlock.get_hearing_knowledge` (self, pattern_or_id, is_id=False)
Return hearing knowledge related to given pattern or neuron id, if pattern or neuron_id in hearing network, and None in any other case.
- def `sensory_neural_block.SensoryNeuralBlock.get_sight_knowledge` (self, pattern_or_id, is_id=False)
Return hearing knowledge related to given pattern or neuron id, if pattern or neuron_id in sight network, and None in any other case.
- def `sensory_neural_block.SensoryNeuralBlock.save` (self, sight_snb_file, hearing_snb_file)
Save snb object in given files (one for the sight sensory neural block and the other for the hearing neural block).

Variables

- int `sensory_neural_block.RbfKnowledge.PATTERN_SIZE` = 4
Size of data or knowledge in bytes.
- int `sensory_neural_block.RbfNeuron.instances_count` = 0
Number of class instances.
- int `sensory_neural_block.RbfNeuron.DEFAULT_RADIUS` = 10
Default radius.
- int `sensory_neural_block.RbfNeuron.MIN_RADIUS` = 1
Minimun radius before neuron is degraded.
- int `sensory_neural_block.RbfNeuron.MAX_RADIUS` = 50
Maximum radius.
- float `sensory_neural_block.RbfNetwork.PATTERN_SIZE` = 4.0
Size of data or knowledge in bytes.
- int `sensory_neural_block.RbfNetwork.DEFAULT_RADIUS` = 5
Default radius.
- **`sensory_neural_block.RbfNetwork.neuron_list`**
- int `sensory_neural_block.SensoryNeuralBlock.SIGHT_NEURON_COUNT` = 100
Number of neurons in sight network.
- int `sensory_neural_block.SensoryNeuralBlock.HEARING_NEURON_COUNT` = 100
Number of neurons in hearing network.
- `sensory_neural_block.SensoryNeuralBlock.snb_s`
Sight sensory neural block.
- `sensory_neural_block.SensoryNeuralBlock.snb_h`
Hearing sensory neural block.

4.8.1 Detailed Description

RBF network related classes are a group of classes that represent and use RBF knowledge, neurons and networks.

4.8.2 Function Documentation

4.8.2.1 deserialize()

```
def sensory_neural_block.RbfNetwork.deserialize (
    cls,
    name )
```

Deserialize object stored in given file.

Parameters

<i>cls</i>	RbfNetwork class
<i>name</i>	Name of the file where the object is serialized

4.8.2.2 get_class() [1/2]

```
def sensory_neural_block.RbfKnowledge.get_class (
    self )
```

Get stored pattern class.

Return values

<i>class</i>	Stored pattern class.
--------------	-----------------------

4.8.2.3 get_class() [2/2]

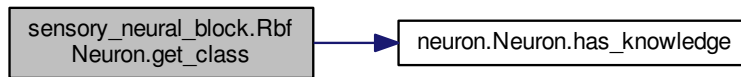
```
def sensory_neural_block.RbfNeuron.get_class (
    self )
```

Get class of stored [RbfKnowledge](#) instance.

Return values

<i>class</i>	RbfKnowledge class if neuron has knowledge, None in any other case
--------------	--

Here is the call graph for this function:



4.8.2.4 get_distance()

```
def sensory_neural_block.RbfNeuron.get_distance (
    self )
```

Get distance to last instance or [RbfKnowledge](#) pattern that tried to be recognized.

Return values

<i>distance</i>	integer
-----------------	---------

4.8.2.5 get_index_ready_to_learn()

```
def sensory_neural_block.RbfNetwork.get_index_ready_to_learn (
    self )
```

Get index of ready-to-learn neuron.

Return values

<i>index</i>	Integer
--------------	---------

4.8.2.6 get_knowledge()

```
def sensory_neural_block.RbfNetwork.get_knowledge (
    self )
```

Get [RbfKnowledge](#) related to last recognized pattern.

Return values

<i>knowledge</i>	RbfKnowledge if "HIT" in last recognition, None in any other case
------------------	---

4.8.2.7 `get_last_learned_id()`

```
def sensory_neural_block.RbfNetwork.get_last_learned_id (
    self )
```

Get id of neuron affected in the last learning process.

Return values

<i>id</i>	Integer
-----------	---------

4.8.2.8 `get_neuron_count()`

```
def sensory_neural_block.RbfNetwork.get_neuron_count (
    self )
```

get number of neurons in network

Return values

<i>count</i>	Integer. Number of neurons in network
--------------	---------------------------------------

4.8.2.9 `get_pattern()` [1/2]

```
def sensory_neural_block.RbfKnowledge.get_pattern (
    self )
```

Get stored pattern.

Return values

<i>pattern</i>	Stored pattern. Integers vector of size PATTERN_SIZE
----------------	--

4.8.2.10 `get_pattern()` [2/2]

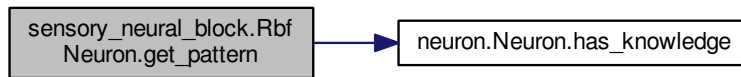
```
def sensory_neural_block.RbfNeuron.get_pattern (
    self )
```

Get pattern of stored [RbfKnowledge](#) instance.

Return values

<i>pattern</i>	RbfKnowledge pattern if neuron has knowledge, None in any other case
----------------	--

Here is the call graph for this function:



4.8.2.11 `get_radius()`

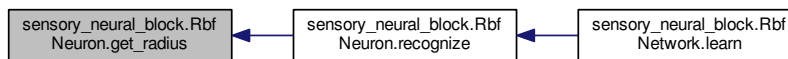
```
def sensory_neural_block.RbfNeuron.get_radius (
    self )
```

Get neuron radius.

Return values

<i>radius</i>	Integer. Neuron radius
---------------	------------------------

Here is the caller graph for this function:



4.8.2.12 `get_rneurons_ids()`

```
def sensory_neural_block.RbfNetwork.get_rneurons_ids (
    self )
```

Get ids of recognizing set neurons.

Return values

<i>ids</i>	Integers list
------------	---------------

4.8.2.13 `get_set()` [1/2]

```
def sensory_neural_block.RbfKnowledge.get_set (
    self )
```

Get stored pattern set.

Return values

<i>set</i>	Stored pattern set
------------	--------------------

Here is the caller graph for this function:

4.8.2.14 `get_set()` [2/2]

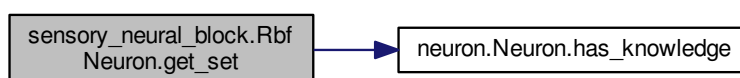
```
def sensory_neural_block.RbfNeuron.get_set (
    self )
```

Get set of stored [RbfKnowledge](#) instance.

Return values

<i>set</i>	RbfKnowledge set if neuron has knowledge, None in any other case
------------	--

Here is the call graph for this function:



Here is the caller graph for this function:



4.8.2.15 get_state()

```
def sensory_neural_block.RbfNetwork.get_state (
    self )
```

Get network state.

Return values

<i>state</i>	'HIT' if the given pattern is recognized, 'MISS' if the network does not recognize the pattern and 'DIFF' if the network identifies the pattern as pertaining to different classes
--------------	--

4.8.2.16 increase_radius_by()

```
def sensory_neural_block.RbfNeuron.increase_radius_by (
    self,
    value )
```

Increase neuron radius by certain amount.

Parameters

<i>value</i>	Integer retval success True if neuron has not been degraded after radius reduction and False in any other case
--------------	--

4.8.2.17 is_degraded()

```
def sensory_neural_block.RbfNeuron.is_degraded (
    self )
```

Return whether neuron is degraded.

Return values

<i>degraded</i>	Boolean. True if neuron is degraded.
-----------------	--------------------------------------

4.8.2.18 `is_hit()`

```
def sensory_neural_block.RbfNeuron.is_hit (
    self )
```

Return True if last call to [recognize\(\)](#) was a hit and False in any other case.

Return values

<i>hit</i>	Boolean
------------	---------

4.8.2.19 `is_member()`

```
def sensory_neural_block.RbfNeuron.is_member (
    self,
    test_set )
```

Returns whether neuron is member of the set.

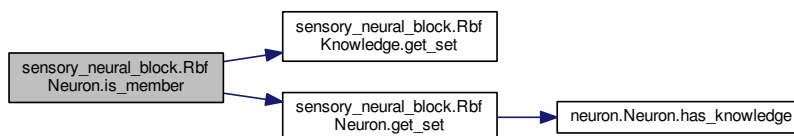
Parameters

<i>test_set</i>	Set to be tested
-----------------	------------------

Return values

<i>is_member</i>	Boolean. True if neuron is member of set, false in any other case
------------------	---

Here is the call graph for this function:



4.8.2.20 `learn()` [1/3]

```
def sensory_neural_block.RbfNeuron.learn (
    self,
    knowledge )
```

Learns a new piece of knowledge.

Return values

<i>learned</i>	Boolean. True if successfully learned, False in any other case
----------------	--

4.8.2.21 `learn()` [2/3]

```
def sensory_neural_block.RbfNetwork.learn (
    self,
    knowledge )
```

Learn an instance of [RbfKnowledge](#).

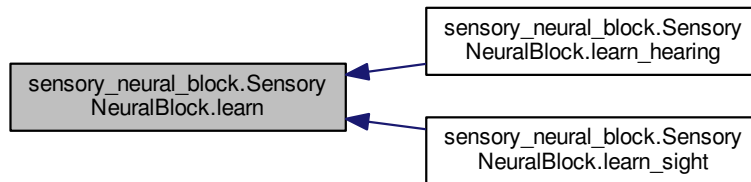
Parameters

<i>knowledge</i>	RbfKnowledge to be learned
------------------	--

Return values

<i>Boolean.</i>	True if successfully learned, False in any other case.
-----------------	--

Here is the caller graph for this function:



4.8.2.23 learn_hearing()

```
def sensory_neural_block.SensoryNeuralBlock.learn_hearing (
    self,
    knowledge )
```

Learn a hearing pattern.

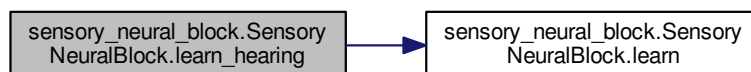
Parameters

<i>pattern</i>	RBF hearing pattern
----------------	---------------------

Return values

<i>success</i>	True if pattern successfully learned, False in any other case
----------------	---

Here is the call graph for this function:



4.8.2.24 learn_sight()

```
def sensory_neural_block.SensoryNeuralBlock.learn_sight (
    self,
    knowledge )
```

Learn a visual pattern.

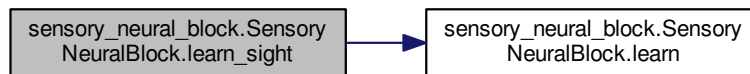
Parameters

<i>pattern</i>	RBF sight pattern
----------------	-------------------

Return values

<i>success</i>	True if pattern successfully recognized, False in any other case
----------------	--

Here is the call graph for this function:



4.8.2.25 recognize() [1/2]

```
def sensory_neural_block.RbfNeuron.recognize (
    self,
    pattern )
```

Recognize a piece of knowledge.

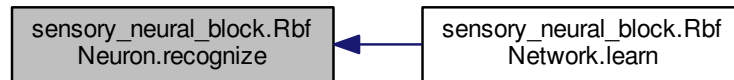
Return values

<i>recognized</i>	Boolean. True if successfully recognized, False in any other case
-------------------	---

Here is the call graph for this function:



Here is the caller graph for this function:



4.8.2.26 recognize() [2/2]

```
def sensory_neural_block.RbfNetwork.recognize (
    self,
    pattern )
```

Recognize a given pattern.

Parameters

<i>pattern</i>	RbfKnowledge pattern to be recognized
----------------	---

Return values

<i>result</i>	'HIT' if the given pattern is recognized, 'MISS' if the network does not recognize the pattern and 'DIFF' if the network identifies the pattern as pertaining to different classes
---------------	--

Here is the caller graph for this function:



4.8.2.27 recognize_hearing()

```
def sensory_neural_block.SensoryNeuralBlock.recognize_hearing (
    self,
    pattern )
```

Recognize a hearing pattern.

Parameters

<i>pattern</i>	RBF hearing pattern
----------------	---------------------

Return values

<i>success</i>	True if pattern successfully recognized, False in any other case
----------------	--

4.8.2.28 recognize_sight()

```
def sensory_neural_block.SensoryNeuralBlock.recognize_sight (
    self,
    pattern )
```

Recognize a sight pattern.

Parameters

<i>pattern</i>	RBF sight pattern
----------------	-------------------

Return values

<i>success</i>	True if pattern successfully recognized, False in any other case
----------------	--

4.8.2.29 reduce_radius_by()

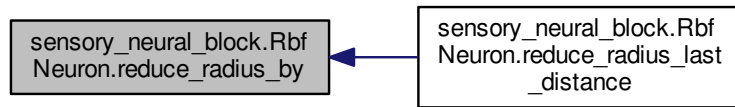
```
def sensory_neural_block.RbfNeuron.reduce_radius_by (
    self,
    value )
```

Reduce neuron radius by certain amount.

Parameters

<i>value</i>	Integer retval success True if neuron has not been degraded after radius reduction and False in any other case
--------------	--

Here is the caller graph for this function:



4.8.2.30 `reduce_radius_last_distance()`

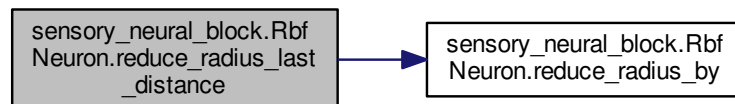
```
def sensory_neural_block.RbfNeuron.reduce_radius_last_distance (
    self )
```

Reduce radius by las recognition process's distance.

Return values

<i>success</i>	Boolean. True if radius successfully reduced, False in any other case
----------------	---

Here is the call graph for this function:



4.8.2.31 `save()`

```
def sensory_neural_block.SensoryNeuralBlock.save (
    self,
    sight_snb_file,
    hearing_snb_file )
```

Save snb object in given files (one for the sight sensory neural block and the other for the hearing neural block).

Parameters

<i>sight_snb_file</i>	Filename where the sight sensory neural block is to be saved
<i>hearing_snb_file</i>	Filename where the hearing sensory neural block is to be saved

4.8.2.32 `serialize()`

```
def sensory_neural_block.RbfNetwork.serialize (
    cls,
    obj,
    name )
```

Serialize object and store in given file.

Parameters

<i>cls</i>	RbfNetwork class
<i>obj</i>	RbfNetwork object to be serialized
<i>name</i>	Name of the file where the serialization is to be stored

4.8.2.33 `set_class()`

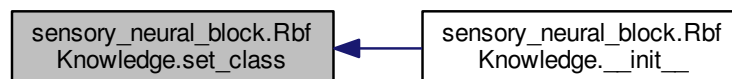
```
def sensory_neural_block.RbfKnowledge.set_class (
    self,
    rbf_class )
```

Set pattern class.

Parameters

<i>rbf_class</i>	Class of the pattern.
------------------	-----------------------

Here is the caller graph for this function:



4.8.2.34 `set_pattern()`

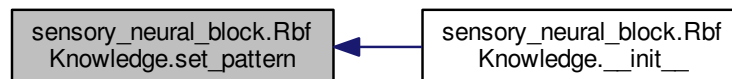
```
def sensory_neural_block.RbfKnowledge.set_pattern (
    self,
    pattern )
```

Set pattern.

Parameters

<i>pattern</i>	Pattern to be stored. Integers vector of size PATTERN_SIZE
----------------	--

Here is the caller graph for this function:

4.8.2.35 `set_radius()`

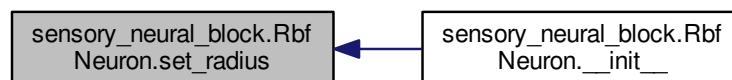
```
def sensory_neural_block.RbfNeuron.set_radius (
    self,
    radius )
```

Sets neuron radius.

Parameters

<i>radius</i>	New neuron radius
---------------	-------------------

Here is the caller graph for this function:



4.8.2.36 set_set()

```
def sensory_neural_block.RbfKnowledge.set_set (
    self,
    rbf_set )
```

Set pattern set.

Parameters

<i>rbf_set</i>	Set of the pattern
----------------	--------------------

Here is the caller graph for this function:



Chapter 5

Class Documentation

5.1 `geometric_neural_block.AdditionStructure` Class Reference

The addition structure class provides a decimal numeric system neural representation.

Collaboration diagram for `geometric_neural_block.AdditionStructure`:

geometric_neural_block.Addition Structure
+ neurons + carry_over + index
+ __init__() + bum() + bip() + clack() + has_carry() + clear_carry()

Public Member Functions

- def `__init__`(self)
The constructor.
- def `bum`(self)
Start an addition operation and set the carry over variable to zero.

- `def bip (self)`
Point to next neuron in structure.
- `def clack (self)`
Return results.
- `def has_carry (self)`
Return True if the structure has a carry over and False in any other case.
- `def clear_carry (self)`
Clear carry over.

Public Attributes

- `neurons`
List of neurons in the structure.
- `carry_over`
Boolean variable.
- `index`
Structure index.

5.1.1 Detailed Description

The addition structure class provides a decimal numeric system neural representation.

This structure has a set of ten neurons and a carry-over neuron. It also has an structure index which is sequentially incremented over the ten neurons in order to make an addition.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 `__init__()`

```
def geometric_neural_block.AdditionStructure.__init__ (
    self )
```

The constructor.

5.1.3 Member Data Documentation

5.1.3.1 `carry_over`

```
geometric_neural_block.AdditionStructure.carry_over
```

Boolean variable.

True signals a carry over.

5.1.3.2 index

`geometric_neural_block.AdditionStructure.index`

Structure index.

Points to one of the ten neurons in the structure.

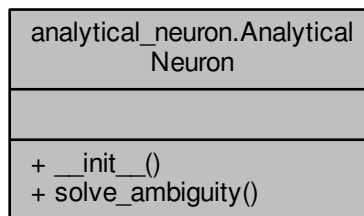
The documentation for this class was generated from the following file:

- `geometric_neural_block.py`

5.2 analytical_neuron.AnalyticalNeuron Class Reference

Analytical neuron.

Collaboration diagram for `analytical_neuron.AnalyticalNeuron`:



Public Member Functions

- `def __init__(self)`
The constructor.
- `def solve_ambiguity(self, rel_knowledge_v)`
Solve ambiguities.

5.2.1 Detailed Description

Analytical neuron.

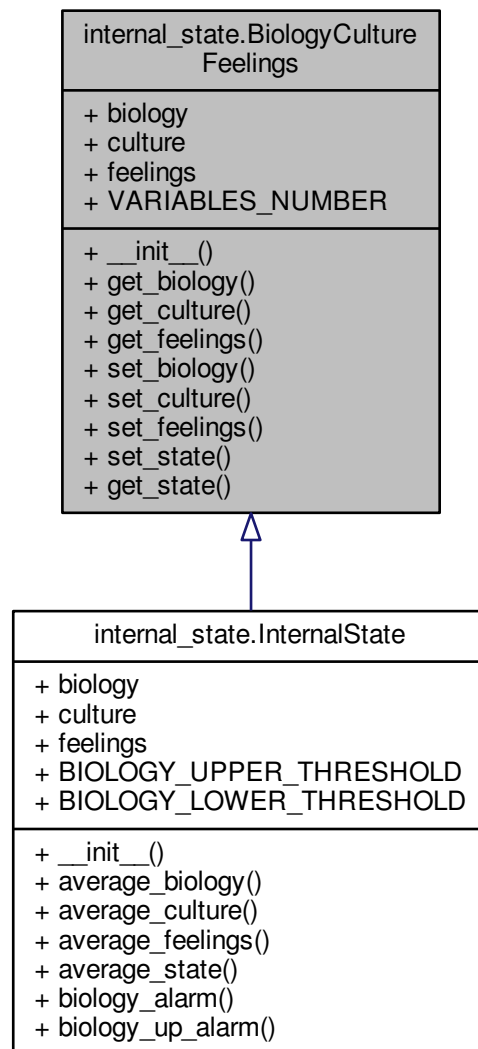
The documentation for this class was generated from the following file:

- `analytical_neuron.py`

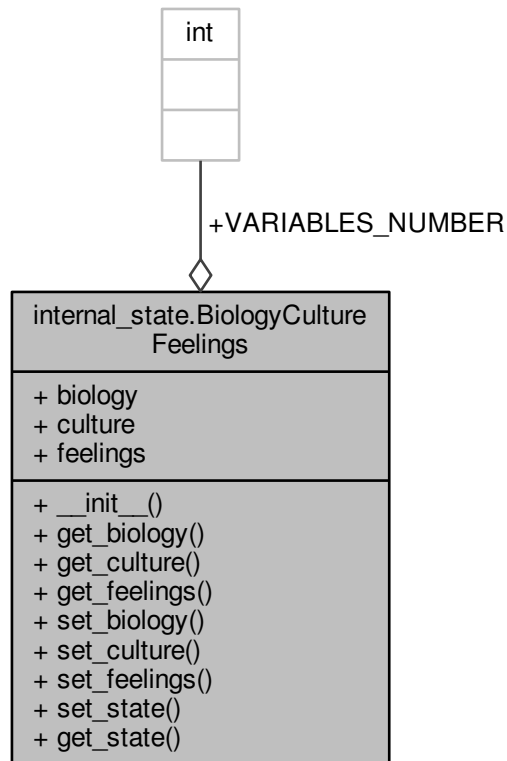
5.3 internal_state.BiologyCultureFeelings Class Reference

[BiologyCultureFeelings](#) models the state of an entity by using a three elements vector which values go from 0 to 1 and correspond to the Biology, Culture and Feelings situation of the entity.

Inheritance diagram for internal_state.BiologyCultureFeelings:



Collaboration diagram for internal_state.BiologyCultureFeelings:



Public Member Functions

- def `__init__` (self, initial_state=[0.5])
The constructor.
- def `get_biology` (self)
Get biology state.
- def `get_culture` (self)
Get culture state.
- def `get_feelings` (self)
Get feelings state.
- def `set_biology` (self, val)
Set biology state.
- def `set_culture` (self, val)
Set culture state.
- def `set_feelings` (self, val)
Set feelings state.

- def `set_state` (self, vals)
Set state.
- def `get_state` (self)
Get state.

Public Attributes

- **biology**
- **culture**
- **feelings**

Static Public Attributes

- int `VARIABLES_NUMBER` = 3
Number of variables = 3 (Biology, Culture, Feelings)

5.3.1 Detailed Description

`BiologyCultureFeelings` models the state of an entity by using a three elements vector which values go from 0 to 1 and correspond to the Biology, Culture and Feelings situation of the entity.

5.3.2 Member Function Documentation

5.3.2.1 `get_biology()`

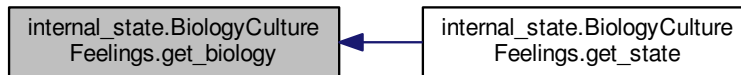
```
def internal_state.BiologyCultureFeelings.get_biology (  
    self )
```

Get biology state.

Return values

<i>biology</i>	Float from 0 to 1.
----------------	--------------------

Here is the caller graph for this function:



5.3.2.2 `get_culture()`

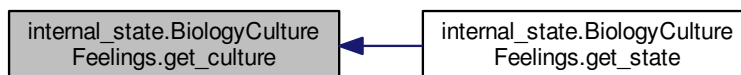
```
def internal_state.BiologyCultureFeelings.get_culture (
    self )
```

Get culture state.

Return values

<i>culture</i>	Float from 0 to 1.
----------------	--------------------

Here is the caller graph for this function:



5.3.2.3 `get_feelings()`

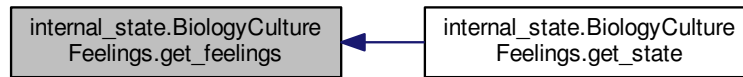
```
def internal_state.BiologyCultureFeelings.get_feelings (
    self )
```

Get feelings state.

Return values

<i>feelings</i>	Float from 0 to 1.
-----------------	--------------------

Here is the caller graph for this function:



5.3.2.4 `get_state()`

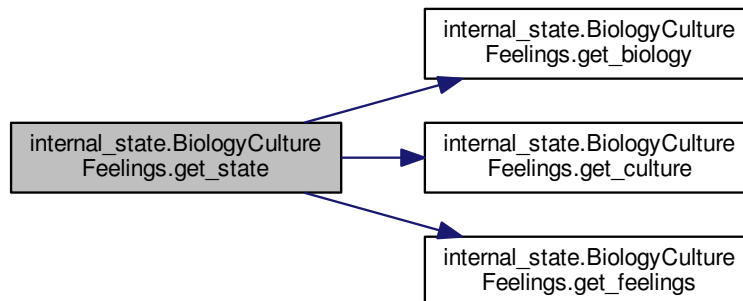
```
def internal_state.BiologyCultureFeelings.get_state (
    self )
```

Get state.

Return values

<i>state</i>	Entity's state
--------------	----------------

Here is the call graph for this function:



5.3.2.5 `set_biology()`

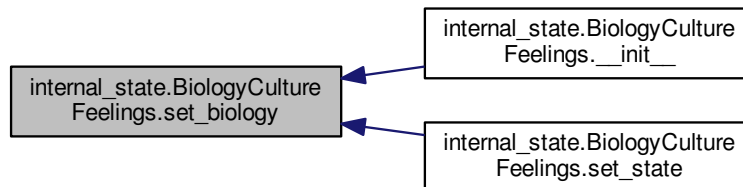
```
def internal_state.BiologyCultureFeelings.set_biology (
    self,
    val )
```

Set biology state.

Parameters

<i>val</i>	Float form 0 to 1. New biology state value.
------------	---

Here is the caller graph for this function:

**5.3.2.6 set_culture()**

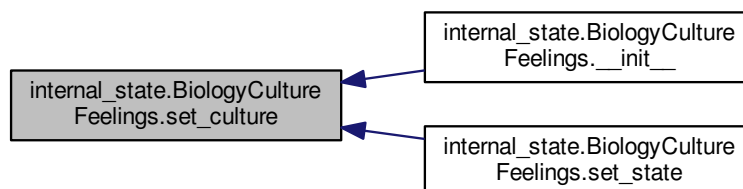
```
def internal_state.BiologyCultureFeelings.set_culture (
    self,
    val )
```

Set culture state.

Parameters

<i>val</i>	Float form 0 to 1. New culture state value.
------------	---

Here is the caller graph for this function:



5.3.2.7 set_feelings()

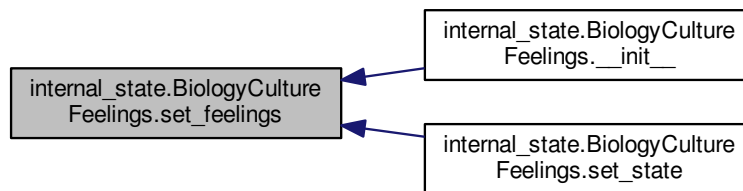
```
def internal_state.BiologyCultureFeelings.set_feelings (
    self,
    val )
```

Set feelings state.

Parameters

<i>val</i>	Float form 0 to 1. New feelings state value.
------------	--

Here is the caller graph for this function:



5.3.2.8 set_state()

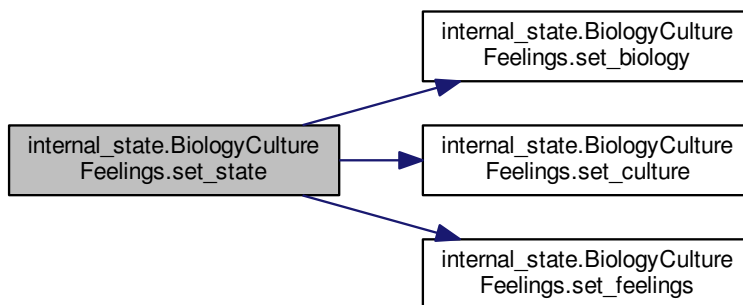
```
def internal_state.BiologyCultureFeelings.set_state (
    self,
    vals )
```

Set state.

Parameters

--	--

Here is the call graph for this function:

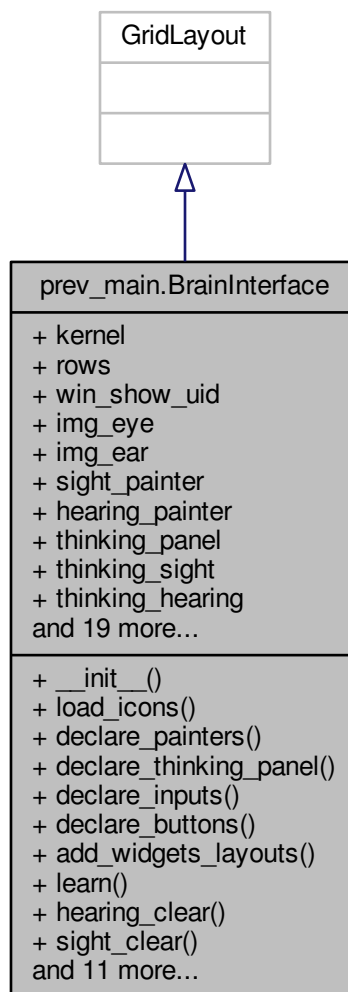


The documentation for this class was generated from the following file:

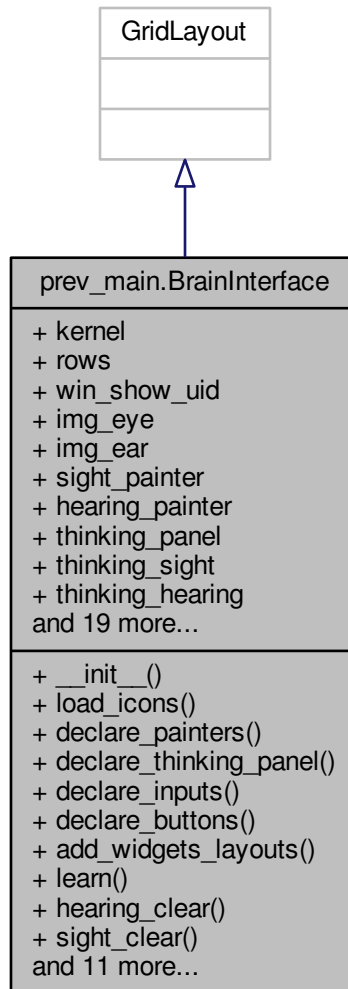
- `internal_state.py`

5.4 prev_main.BrainInterface Class Reference

Inheritance diagram for prev_main.BrainInterface:



Collaboration diagram for prev_main.BrainInterface:



Public Member Functions

- def **__init__** (self, kwargs)
- def **load_icons** (self)
- def **declare_painters** (self, grid_size)
- def **declare_thinking_panel** (self)
- def **declare_inputs** (self)
- def **declare_buttons** (self)
- def **add_widgets_layouts** (self)
- def **learn** (self, obj)
- def **hearing_clear** (self, obj)

- def **sight_clear** (self, obj)
- def **bum** (self, obj)
- def **bip** (self, obj)
- def **check** (self, obj)
- def **clack** (self, obj)
- def **pass_kernel_inputs** (self)
- def **show_kernel_outputs** (self)
- def **thinking_clear** (self)
- def **clear** (self, obj)
- def **set_zero** (self, obj)
- def **set_add_operator** (self, obj)
- def **set_equal_sign** (self, obj)

Public Attributes

- **kernel**
- **rows**
- **win_show_uid**
- **img_eye**
- **img_ear**
- **sightPainter**
- **hearingPainter**
- **thinkingPanel**
- **thinking_sight**
- **thinking_hearing**
- **hearing_class_input**
- **sight_clear_btn**
- **hearing_clear_btn**
- **bum_btn**
- **bip_btn**
- **check_btn**
- **clack_btn**
- **zero_btn**
- **equal_btn**
- **add_operator_btn**
- **words_tgl_btn**
- **addition_tgl_btn**
- **counting_tgl_btn**
- **sight_panel**
- **hearingPainter_text**
- **hearing_panel**
- **main_panel**
- **senses_panel**
- **buttons_panel**

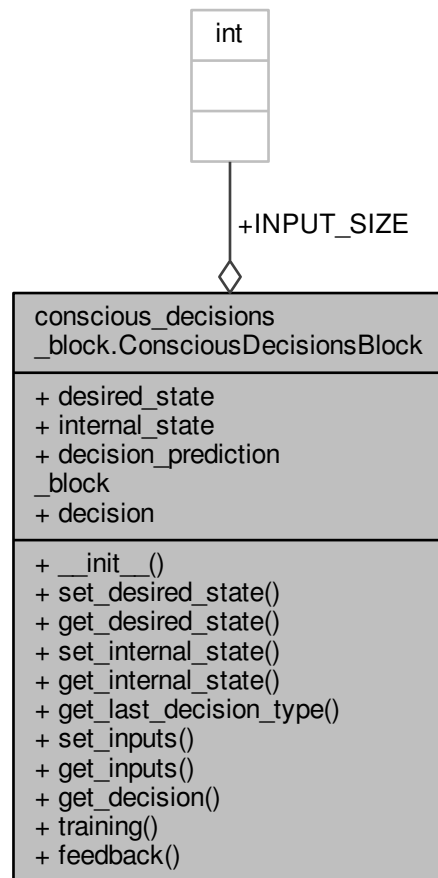
The documentation for this class was generated from the following file:

- `prev_main.py`

5.5 conscious_decisions_block.ConsciousDecisionsBlock Class Reference

The Conscious Decisions Block is in charge of 'rationally' evaluating current internal state and goals in order to make decisions.

Collaboration diagram for conscious_decisions_block.ConsciousDecisionsBlock:



Public Member Functions

- `def __init__(self)`
Class constructor.
- `def set_desired_state(self, desired_state)`
Set desired state.
- `def get_desired_state(self)`
Get desired state.

- def `set_internal_state` (self, internal_state)
Set internal state.
- def `get_internal_state` (self)
Get internal state.
- def `get_last_decision_type` (self)
Get last decision type.
- def `set_inputs` (self, inputs)
Set conscious decisions block inputs.
- def `get_inputs` (self)
Get block inputs.
- def `get_decision` (self)
Get decision decision Integer.
- def `training` (self, training_set)
Train predictive network.
- def `feedback` (self, new_internal_state)
Feedback a new internal state to prediction network.

Public Attributes

- **desired_state**
- **internal_state**
- **decision_prediction_block**
- **decision**

Static Public Attributes

- int `INPUT_SIZE` = 9
Input size.

5.5.1 Detailed Description

The Conscious Decisions Block is in charge of 'rationally' evaluating current internal state and goals in order to make decisions.

The quotes in 'rationally' stand for the agent trying to choose what he thinks is the best course of action according to past experiences, but not what is really best in the mathematical sense (objective function). The result from the last decision taken can be fed back for evaluation and evolution of its behaviour

5.5.2 Member Function Documentation

5.5.2.1 `feedback()`

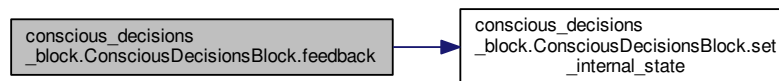
```
def conscious_decisions_block.ConsciousDecisionsBlock.feedback (
    self,
    new_internal_state )
```

Feedback a new internal state to prediction network.

Parameters

<i>new_internal_state</i>	InternalState. New internal state after making a decision and acting on environment
---------------------------	---

Here is the call graph for this function:

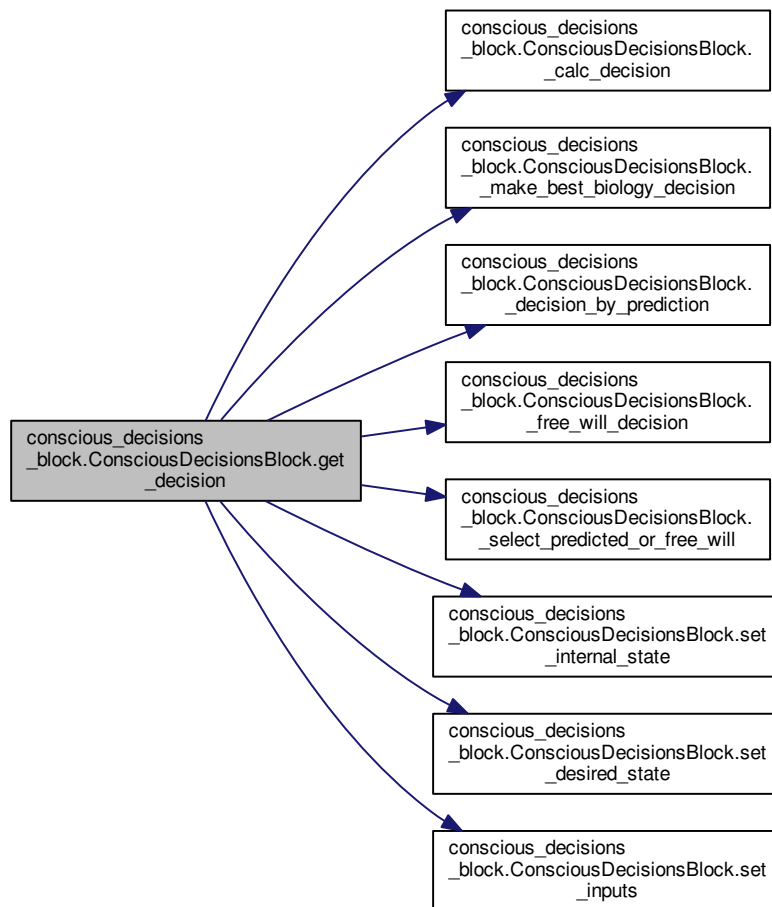


5.5.2.2 get_decision()

```
def conscious_decisions_block.ConsciousDecisionsBlock.get_decision (
    self )
```

Get decision decision Integer.

Index of the selected input. Here is the call graph for this function:



5.5.2.3 get_desired_state()

```
def conscious_decisions_block.ConsciousDecisionsBlock.get_desired_state (
    self )
```

Get desired state.

Return values

<i>desired_state</i>	InternalState. Stored desired state
----------------------	-------------------------------------

5.5.2.4 get_inputs()

```
def conscious_decisions_block.ConsciousDecisionsBlock.get_inputs (
    self )
```

Get block inputs.

Return values

<i>inputs</i>	vector of the form [bcf, bcf, bcf]
---------------	------------------------------------

5.5.2.5 get_internal_state()

```
def conscious_decisions_block.ConsciousDecisionsBlock.get_internal_state (
    self )
```

Get internal state.

Return values

<i>internal_state</i>	InternalState. Stored internal state
-----------------------	--------------------------------------

5.5.2.6 get_last_decision_type()

```
def conscious_decisions_block.ConsciousDecisionsBlock.get_last_decision_type (
    self )
```

Get last decision type.

Return values

<i>last_decision_type</i>	Enumeration with values 'BIOLOGY_ALARM', 'FREE_WILL' or 'PREDICTED'
---------------------------	---

5.5.2.7 set_desired_state()

```
def conscious_decisions_block.ConsciousDecisionsBlock.set_desired_state (
    self,
    desired_state )
```

Set desired state.

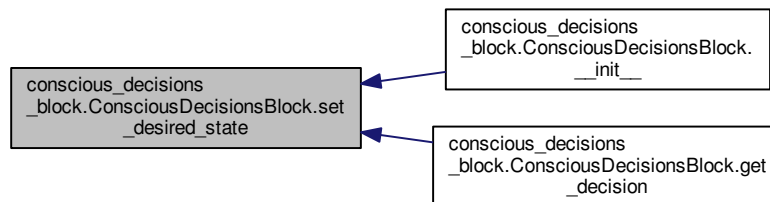
Parameters

<i>desired_state</i>	InternalState. Desired internal state
----------------------	---------------------------------------

Return values

<i>result</i>	Boolean. True if desired state correctly set, False in any other case
---------------	---

Here is the caller graph for this function:

**5.5.2.8 set_inputs()**

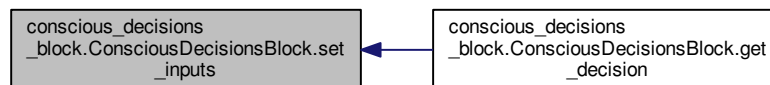
```
def conscious_decisions_block.ConsciousDecisionsBlock.set_inputs (
    self,
    inputs )
```

Set conscious decisions block inputs.

Parameters

<i>inputs</i>	vector of inputs of the form [bcf, bcf, bcf] (1st input, 2nd input, 3rd input)
---------------	--

Here is the caller graph for this function:

**5.5.2.9 set_internal_state()**

```
def conscious_decisions_block.ConsciousDecisionsBlock.set_internal_state (
    self,
    internal_state )
```

Set internal state.

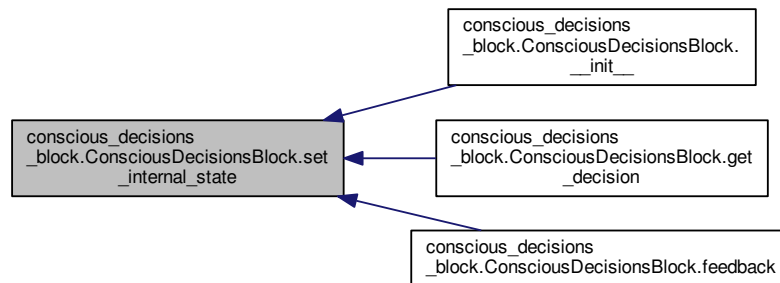
Parameters

<i>internal_state</i>	InternalState. New internal state.
-----------------------	------------------------------------

Return values

<i>result</i>	Boolean. True if internal state correctly set, False in any other case
---------------	--

Here is the caller graph for this function:



5.5.2.10 training()

```
def conscious_decisions_block.ConsciousDecisionsBlock.training (
    self,
    training_set )
```

Train predictive network.

Parameters

<i>training_set</i>	Vector of the form [[bcf_is bcf_i], bcf_o] where bcf_is is the internal state BCF and bcf_i is the input BCF and bcf_o is the expected or predicted new internal state bcf
---------------------	--

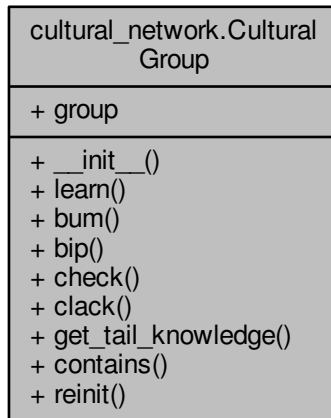
The documentation for this class was generated from the following file:

- `conscious_decisions_block.py`

5.6 cultural_network.CulturalGroup Class Reference

Cultural group.

Collaboration diagram for cultural_network.CulturalGroup:



Public Member Functions

- def [__init__](#) (self)
CulturalGroup class constructor.
- def [learn](#) (self, knowledge)
Learn new piece of cultural knowledge as part of the cultural group.
- def [bum](#) (self)
Initialize bbcc protocol.
- def [bip](#) (self, knowledge)
Return True if given knowledge equals the one store in current neuron.
- def [check](#) (self, knowledge)
Return true if given knowledge equals the one store in current neuron and there is exactly one more neuron in the group with the final knowledge related to de bbcc sequence.
- def [clack](#) (self, knowledge)
Learn new piece of cultural knowledge as part of the cultural group.
- def [get_tail_knowledge](#) (self)
Get last knowledge in group.
- def [contains](#) (self, knowledge)
Return True if knowledge is contained by some neuron in the group and False in any other case.
- def [reinit](#) (self)
Erase all knowledge in group.

Public Attributes

- **group**

5.6.1 Detailed Description

Cultural group.

5.6.2 Member Function Documentation

5.6.2.1 bip()

```
def cultural_network.CulturalGroup.bip (
    self,
    knowledge )
```

Return True if given knowledge equals the one store in current neuron.

Increase index-bip so that next comparison is made in the following neuron of the group. If there are no more neurons in the group, return False.

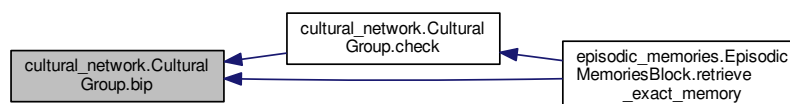
Parameters

<i>knowledge</i>	Piece of knowledge to be compared
------------------	-----------------------------------

Return values

<i>result</i>	Boolean
---------------	---------

Here is the caller graph for this function:



5.6.2.2 check()

```
def cultural_network.CulturalGroup.check (
    self,
    knowledge )
```

Return true if given knowledge equals the one store in current neuron and there is exactly one more neuron in the group with the final knowledge related to de bbcc sequence.

Return false in any other case.

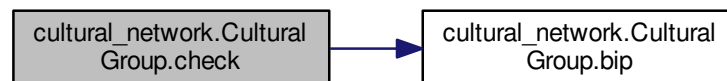
Parameters

<i>knowledge</i>	Piece of knowledge to be compared
------------------	-----------------------------------

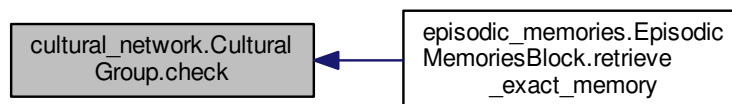
Return values

<i>result</i>	Boolean
---------------	---------

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.2.3 clack()

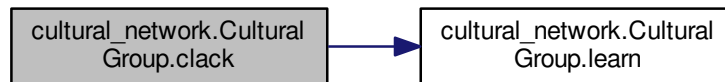
```
def cultural_network.CulturalGroup.clack (
    self,
    knowledge )
```

Learn new piece of cultural knowledge as part of the cultural group.

Parameters

<i>knowledge</i>	Object of type CulturalKnowledge. Knowledge to be learned
------------------	---

Here is the call graph for this function:

**5.6.2.4 contains()**

```
def cultural_network.CulturalGroup.contains (
    self,
    knowledge )
```

Return True if knowledge is contained by some neuron in the group and False in any other case.

Parameters

<i>knowledge</i>	
------------------	--

Return values

<i>result</i>	Boolean
---------------	---------

5.6.2.5 get_tail_knowledge()

```
def cultural_network.CulturalGroup.get_tail_knowledge (
    self )
```

Get last knowledge in group.

Return values

<i>knowledge</i>	Last knowledge in group
------------------	-------------------------

5.6.2.6 learn()

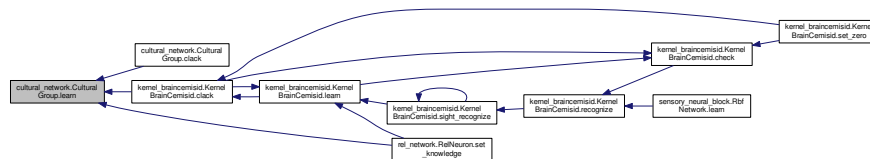
```
def cultural_network.CulturalGroup.learn (
    self,
    knowledge )
```

Learn new piece of cultural knowledge as part of the cultural group.

Parameters

<i>knowledge</i>	Object of type CulturalKnowledge. Knowledge to be learned
------------------	---

Here is the caller graph for this function:



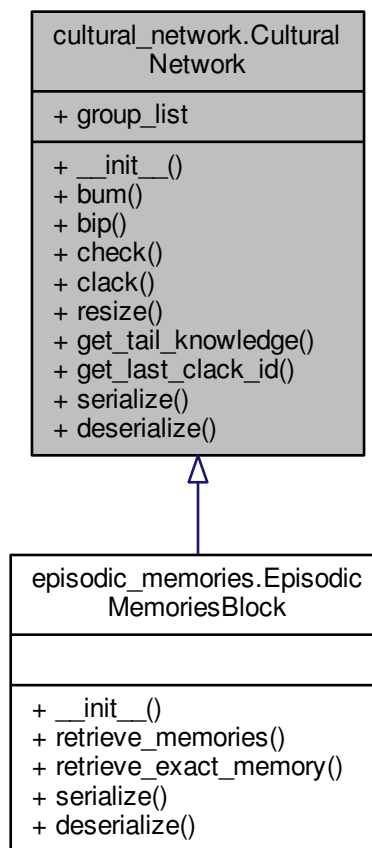
The documentation for this class was generated from the following file:

- cultural_network.py

5.7 cultural_network.CulturalNetwork Class Reference

Cultural network.

Inheritance diagram for cultural_network.CulturalNetwork:



Collaboration diagram for cultural_network.CulturalNetwork:

cultural_network.Cultural Network
+ group_list
+ <code>__init__()</code> + <code>bum()</code> + <code>bip()</code> + <code>check()</code> + <code>clack()</code> + <code>resize()</code> + <code>get_tail_knowledge()</code> + <code>get_last_clack_id()</code> + <code>serialize()</code> + <code>deserialize()</code>

Public Member Functions

- `def __init__(self, group_count=1)`
CulturalNetwork class constructor.
- `def bum(self)`
Start of bbcc protocol.
- `def bip(self, knowledge)`
Pass an instance of knowledge to be compared or stored.
- `def check(self, knowledge)`
Pass the second-to-last instance of knowledge to be compared or stored.
- `def clack(self, knowledge)`
earn tail knowledge of cultural group
- `def resize(self)`
Resize network.
- `def get_tail_knowledge(self, group_id)`
Get tail knowledge of a given group id.
- `def get_last_clack_id(self)`
Get id of group that learned last sequence.
- `def serialize(cls, obj, name)`
Serialize object and store in given file.
- `def deserialize(cls, name)`
Deserialize object stored in given file.

Public Attributes

- `group_list`

5.7.1 Detailed Description

Cultural network.

Set of [CulturalGroup](#) instances

5.7.2 Member Function Documentation

5.7.2.1 bip()

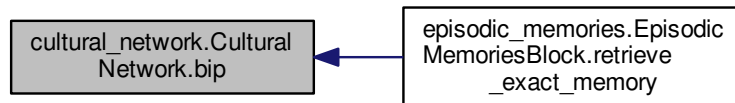
```
def cultural_network.CulturalNetwork.bip (
    self,
    knowledge )
```

Pass an instance of knowledge to be compared or stored.

Parameters

<i>knowledge</i>	
------------------	--

Here is the caller graph for this function:



5.7.2.2 check()

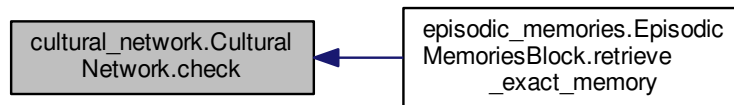
```
def cultural_network.CulturalNetwork.check (
    self,
    knowledge )
```

Pass the second-to-last instance of knowledge to be compared or stored.

Parameters

<i>knowledge</i>	
------------------	--

Here is the caller graph for this function:



5.7.2.3 clack()

```
def cultural_network.CulturalNetwork.clack (
    self,
    knowledge )
```

earn tail knowledge of cultural group

Parameters

<i>knowledge</i>	Tail knowledge
------------------	----------------

5.7.2.4 deserialize()

```
def cultural_network.CulturalNetwork.deserialize (
    cls,
    name )
```

Deserialize object stored in given file.

Parameters

<i>cls</i>	CulturalNetwork class
<i>name</i>	Name of the file where the object is serialize

5.7.2.5 get_tail_knowledge()

```
def cultural_network.CulturalNetwork.get_tail_knowledge (
    self,
    group_id )
```

Get tail knowledge of a given group id.

Parameters

<i>group</i> ↔	
<i>_id</i>	

5.7.2.6 serialize()

```
def cultural_network.CulturalNetwork.serialize (
    cls,
    obj,
    name )
```

Serialize object and store in given file.

Parameters

<i>cls</i>	CulturalNetwork class
<i>obj</i>	CulturalNetwork object to be serialized
<i>name</i>	Name of the file where the serialization is to be stored

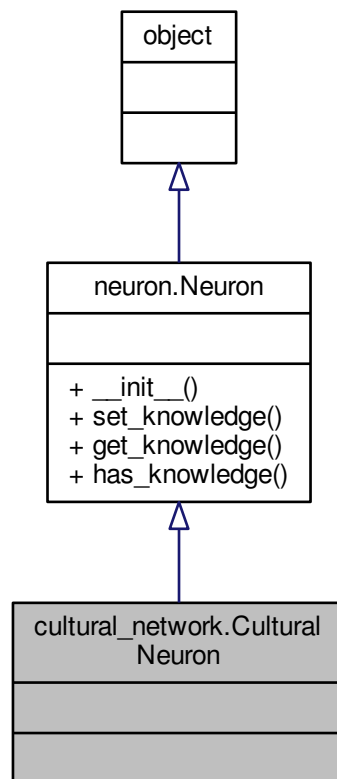
The documentation for this class was generated from the following file:

- cultural_network.py

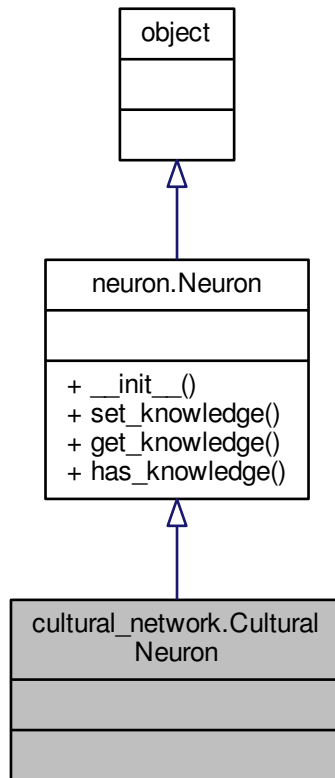
5.8 cultural_network.CulturalNeuron Class Reference

Cultural neuron.

Inheritance diagram for `cultural_network.CulturalNeuron`:



Collaboration diagram for cultural_network.CulturalNeuron:



Additional Inherited Members

5.8.1 Detailed Description

Cultural neuron.

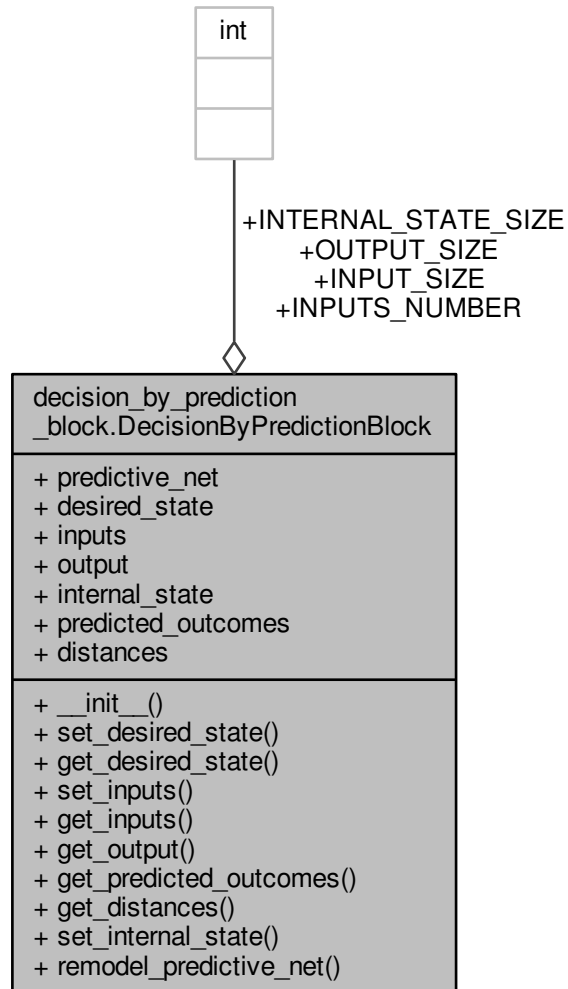
The documentation for this class was generated from the following file:

- cultural_network.py

5.9 decision_by_prediction_block.DecisionByPredictionBlock Class Reference

The [DecisionByPredictionBlock](#) is a class aimed at modeling how decisions can be made through prediction.

Collaboration diagram for `decision_by_prediction_block.DecisionByPredictionBlock`:



Public Member Functions

- `def __init__ (self)`
Create multiclass perceptron network with input size equal to the size of a BCF(input size) plus the size of the internal state.
- `def set_desired_state (self, desired_state)`
Set desired state.
- `def get_desired_state (self)`
Get desired state.
- `def set_inputs (self, inputs)`

- *Set network inputs.*
- def `get_inputs` (self)
- *Get network inputs.*
- def `get_output` (self)
- *Get network output.*
- def `get_predicted_outcomes` (self)
- *Get predicted outcomes calculated in the decision taking process.*
- def `get_distances` (self)
- *Get distances (absolute value of the difference of the components) between the desired state and every predicted outcome.*
- def `set_internal_state` (self, internal_state)
- *Set the entity's internal state.*
- def `remodel_predictive_net` (self, training_set)
- *Re-model or re-train predictive net.*

Public Attributes

- **predictive_net**
- **desired_state**
- **inputs**
- **output**
- **internal_state**
- **predicted_outcomes**
- **distances**

Static Public Attributes

- int `INPUTS_NUMBER` = 3
- *Inputs number.*
- int `INPUT_SIZE` = 3
- *Input size (Each input's number of variables)*
- int `INTERNAL_STATE_SIZE` = 3
- *Internal state size.*
- int `OUTPUT_SIZE` = 3
- *Output size.*

5.9.1 Detailed Description

The `DecisionByPredictionBlock` is a class aimed at modeling how decisions can be made through prediction.

The brain seems to simulate a world and internal (self) model in order to predict the outcomes of the different options it is faced to. The decision is thus made by selecting the option that produces the closest outcome to a desired world and internal state

5.9.2 Member Function Documentation

5.9.2.1 `get_desired_state()`

```
def decision_by_prediction_block.DecisionByPredictionBlock.get_desired_state (
    self )
```

Get desired state.

Return values

<i>desired_state</i>	InternalState.
----------------------	----------------

5.9.2.2 get_distances()

```
def decision_by_prediction_block.DecisionByPredictionBlock.get_distances (
    self )
```

Get distances (absolute value of the difference of the components) between the desired stated and every predicted outcome.

Return values

<i>distances</i>	Floats vector.
------------------	----------------

5.9.2.3 get_inputs()

```
def decision_by_prediction_block.DecisionByPredictionBlock.get_inputs (
    self )
```

Get network inputs.

Return values

<i>inputs</i>	For example <i>[[0.5, 0.9, 0.2],[0.5, 0.9, 0.3],[0.4, 0.7, 0.9]]</i>
---------------	--

5.9.2.4 get_output()

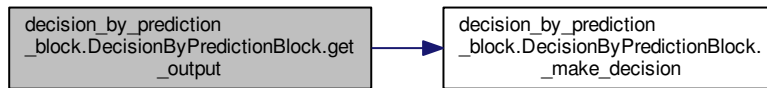
```
def decision_by_prediction_block.DecisionByPredictionBlock.get_output (
    self )
```

Get network output.

Return values

<i>output</i>	Integer. Index of selected input
---------------	----------------------------------

Here is the call graph for this function:



5.9.2.5 get_predicted_outcomes()

```
def decision_by_prediction_block.DecisionByPredictionBlock.get_predicted_outcomes (
    self )
```

Get predicted outcomes calculated in the decision taking process.

Return values

<i>predicted_outcomes</i>	A vector, for instance, <i>[[0.43, 0.31, 0.35], [0.44, 0.32, 0.37], [0.52, 0.37, 0.45]]</i> , where each component is the predicted new internal state componentes (BCF) to get if the corresponding decision (0 for input 0, 1 for input 1, and so on) is taken.
---------------------------	---

5.9.2.6 remodel_predictive_net()

```
def decision_by_prediction_block.DecisionByPredictionBlock.remodel_predictive_net (
    self,
    training_set )
```

Re-model or re-train predictive net.

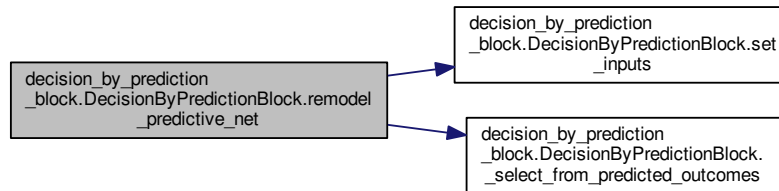
Parameters

<i>training_set</i>	A vector of 2-tuples, where each tuple is as in the following example: <i>*([0.61, 0.18, 0.16, 0.10, 0.13, 0.21], [0.36, 0.16, 0.19])*</i>
---------------------	--

The first element of the tuple is a vector of real numbers between 0 and 1, which first three elements are the components of the internal state, and the last three elements are the components of the BCF associated with this particular decision.

The las element of the tuple is the new internal state to get if the corresponding decision is taken. Here is the call graph

for this function:



5.9.2.7 set_desired_state()

```
def decision_by_prediction_block.DecisionByPredictionBlock.set_desired_state (
    self,
    desired_state )
```

Set desired state.

Parameters

<i>desired_state</i>	InternalState
----------------------	---------------

5.9.2.8 set_inputs()

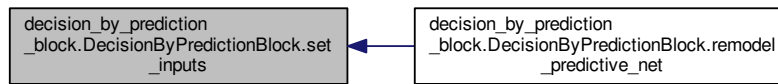
```
def decision_by_prediction_block.DecisionByPredictionBlock.set_inputs (
    self,
    inputs )
```

Set network inputs.

Parameters

<i>inputs</i>	For example, <i>[[0.5, 0.9, 0.2],[0.5, 0.9, 0.3],[0.4, 0.7, 0.9]]</i>
---------------	---

Here is the caller graph for this function:



5.9.2.9 set_internal_state()

```
def decision_by_prediction_block.DecisionByPredictionBlock.set_internal_state (
    self,
    internal_state )
```

Set the entity's internal state.

Parameters

<i>internal_state</i>	InternalState. New internal state
-----------------------	-----------------------------------

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

- decision_by_prediction_block.py

5.10 decisions_block.DecisionsBlock Class Reference

The [DecisionsBlock](#) takes as input a set of memories (CulturalGroup) and gives as output one of them from which the decision can be inferred.

Collaboration diagram for decisions_block.DecisionsBlock:

decisions_block.Decisions Block
+ input_memories + conscious_output + unconscious_output + internal_state + desired_state + unconscious_block + conscious_block
+ __init__() + set_input_memories() + set_internal_state() + set_desired_state() + get_output_memory()

Public Member Functions

- def `__init__`(self)
The constructor.
- def `set_input_memories`(self, input_memories)
Set input memories.
- def `set_internal_state`(self, internal_state)
Set entity's internal state.
- def `set_desired_state`(self, desired_state)
Set entity's desired state.
- def `get_output_memory`(self)
Get output memory.

Public Attributes

- `input_memories`
- `conscious_output`
- `unconscious_output`
- `internal_state`
- `desired_state`
- `unconscious_block`
- `conscious_block`

5.10.1 Detailed Description

The [DecisionsBlock](#) takes as input a set of memories (CulturalGroup) and gives as output one of them from which the decision can be inferred.

For example, if the input are two memories, one related to ice cream and the other to soccer, the output could be the one related to soccer, so the decision is to play soccer.

5.10.2 Member Function Documentation

5.10.2.1 get_output_memory()

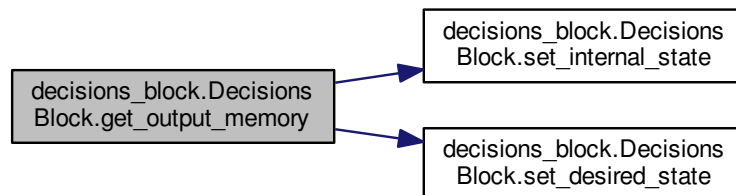
```
def decisions_block.DecisionsBlock.get_output_memory (
    self )
```

Get output memory.

Return values

<i>output</i>	CulturalGroup. The memory from which the decision can be inferred.
---------------	--

Here is the call graph for this function:



5.10.2.2 set_desired_state()

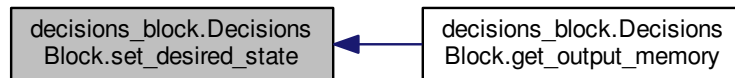
```
def decisions_block.DecisionsBlock.set_desired_state (
    self,
    desired_state )
```

Set entity's desired state.

Parameters

<i>desired_state</i>	InternalState
----------------------	---------------

Here is the caller graph for this function:



5.10.2.3 set_input_memories()

```
def decisions_block.DecisionsBlock.set_input_memories (
    self,
    input_memories )
```

Set input memories.

Parameters

<i>input_memories</i>	CulturalGroup vector where the last element is of type BiologyCultureFeelings (memories)
-----------------------	--

5.10.2.4 set_internal_state()

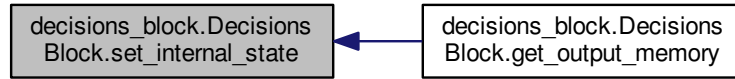
```
def decisions_block.DecisionsBlock.set_internal_state (
    self,
    internal_state )
```

Set entity's internal state.

Parameters

<i>internal_state</i>	InternalState
-----------------------	---------------

Here is the caller graph for this function:



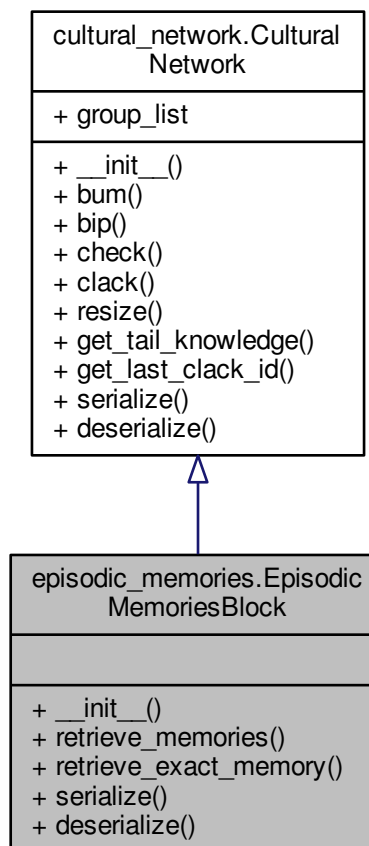
The documentation for this class was generated from the following file:

- `decisions_block.py`

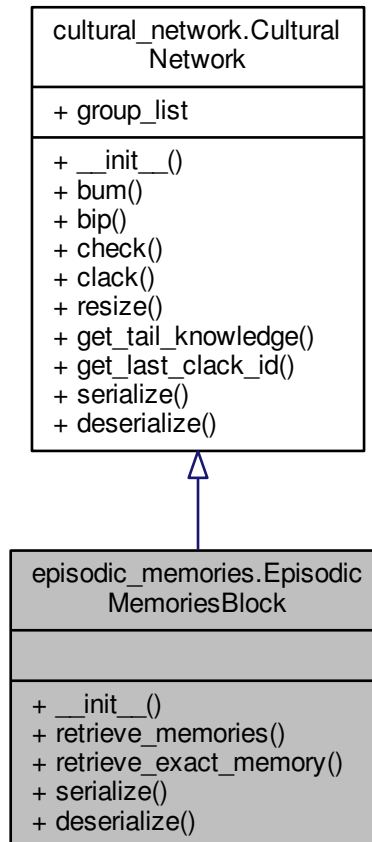
5.11 episodic_memories.EpisodicMemoriesBlock Class Reference

The [EpisodicMemoriesBlock](#) is a specialization of `CulturalNetwork` from which a set of `CulturalGroup`s can be retrieved given a set of triggers, just as in humans.

Inheritance diagram for episodic_memories.EpisodicMemoriesBlock:



Collaboration diagram for episodic_memories.EpisodicMemoriesBlock:



Public Member Functions

- def `__init__` (self)
The constructor.
- def `retrieve_memories` (self, trigger_list)
Return a list of memories (Cultural Groups) that contain the list of given memory triggers.
- def `retrieve_exact_memory` (self, trigger)
Return the exact memory (except for last element in trigger)
- def `serialize` (cls, obj, name)
Serialize object and store it in given file.
- def `deserialize` (cls, name)
Deserialize object stored in given file.

Additional Inherited Members

5.11.1 Detailed Description

The [EpisodicMemoriesBlock](#) is a specialization of [CulturalNetwork](#) from which a set of [CulturalGroup](#) s can be retrieved given a set of triggers, just as in humans.

An exact memory can also be retrieved.

5.11.2 Member Function Documentation

5.11.2.1 deserialize()

```
def episodic_memories.EpisodicMemoriesBlock.deserialize (
    cls,
    name )
```

Deserialize object stored in given file.

Parameters

<i>cls</i>	EpisodicMemory class
<i>name</i>	Name of the file where the object is serialized

5.11.2.2 retrieve_exact_memory()

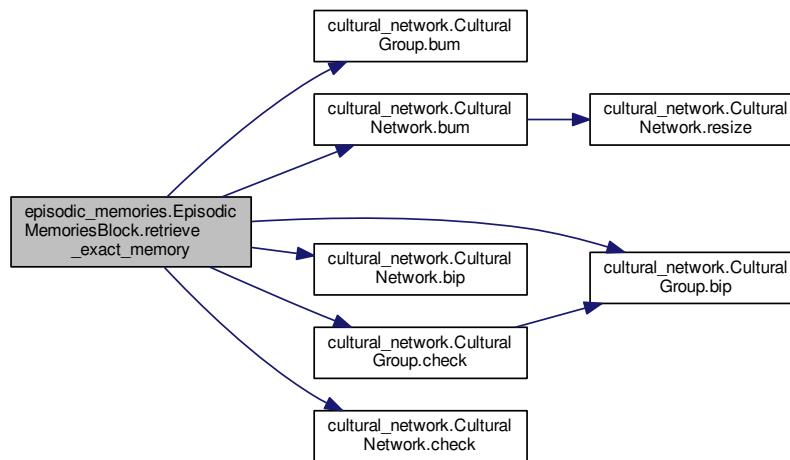
```
def episodic_memories.EpisodicMemoriesBlock.retrieve_exact_memory (
    self,
    trigger )
```

Return the exact memory (except for last element in trigger)

Return values

<i>memory</i>	CulturalGroup
---------------	-------------------------------

Here is the call graph for this function:



5.11.2.3 retrieve_memories()

```
def episodic_memories.EpisodicMemoriesBlock.retrieve_memories (
    self,
    trigger_list )
```

Return a list of memories (Cultural Groups) that contain the list of given memory triggers.

Return values

<i>retrieved_memories</i>	CulturalGroup vector.
---------------------------	-----------------------

5.11.2.4 serialize()

```
def episodic_memories.EpisodicMemoriesBlock.serialize (
    cls,
    obj,
    name )
```

Serialize object and store it in given file.

Parameters

<i>cls</i>	EpisodicMemory class
<i>obj</i>	EpisodicMemory object to be serialized
<i>name</i>	Name of the file where the serialization is to be stored

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

- episodic_memories.py

5.12 `geometric_neural_block.GeometricNeuralBlock` Class Reference

Class that envelopes all neural geometries that represent concepts in the brain.

Collaboration diagram for `geometric_neural_block.GeometricNeuralBlock`:

<code>geometric_neural_block.GeometricNeuralBlock</code>
+ <code>addition_result</code>
+ <code>__init__()</code> + <code>set_operation()</code> + <code>get_operation()</code> + <code>set_add_operator()</code> + <code>get_add_operator()</code> + <code>set_equal_sign()</code> + <code>get_equal_sign()</code> + <code>set_zero()</code> + <code>bum()</code> + <code>bip()</code> + <code>clack()</code> + <code>get_addition_result()</code> + <code>serialize()</code> + <code>deserialize()</code>

Public Member Functions

- `def __init__(self)`
The constructor.
- `def set_operation(self, operation)`
Set an operation to be executed by the block.
- `def get_operation(self)`
Get the type of operation to be executed by the block.
- `def set_add_operator(self, knowledge)`
Set the knowledge that represents an addition operator.
- `def get_add_operator(self)`
Return the knowledge that represents the addition operator.

- def `set_equal_sign` (self, knowledge)
Set the knowledge that represents an equal sign.
- def `get_equal_sign` (self)
Return the knowledge that represents the equal sign.
- def `set_zero` (self, knowledge)
Set the knowledge that represents a zero.
- def `bum` (self)
Start operation.
- def `bip` (self, knowledge=None)
Either count or pass addition operands and operator.
- def `clack` (self, knowledge=None)
Either finish counting or adding.
- def `get_addition_result` (self)
Get addition result.
- def `serialize` (cls, obj, name)
Serialize object and store in given file.
- def `deserialize` (cls, name)
Deserialize object stored in given file.

Public Attributes

- **`addition_result`**

5.12.1 Detailed Description

Class that envelopes all neural geometries that represent concepts in the brain.

5.12.2 Member Function Documentation

5.12.2.1 `bip()`

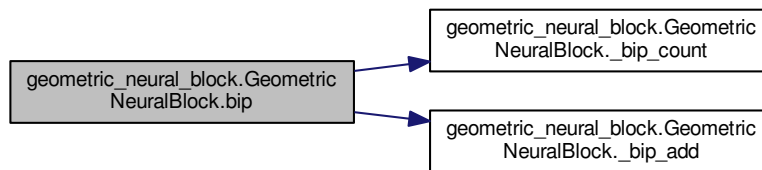
```
def geometric_neural_block.GeometricNeuralBlock.bip (
    self,
    knowledge = None )
```

Either count or pass addition operands and operator.

Parameters

<i>knowledge</i>	Optional parameter that stores the information of the operands and operator for and addition operation. In the BrainCEMISID project it is just the hearing neuron id of the corresponding pattern.
------------------	--

Here is the call graph for this function:



Here is the caller graph for this function:



5.12.2.2 deserialize()

```
def geometric_neural_block.GeometricNeuralBlock.deserialize (
    cls,
    name )
```

Deserialize object stored in given file.

Parameters

<i>cls</i>	GeometricNeuralBlock class
<i>name</i>	Name of the file where the object is serialized

5.12.2.3 get_add_operator()

```
def geometric_neural_block.GeometricNeuralBlock.get_add_operator (
    self )
```

Return the knowledge that represents the addition operator.

Return values

<i>operator</i>	Addition operator
-----------------	-------------------

5.12.2.4 get_addition_result()

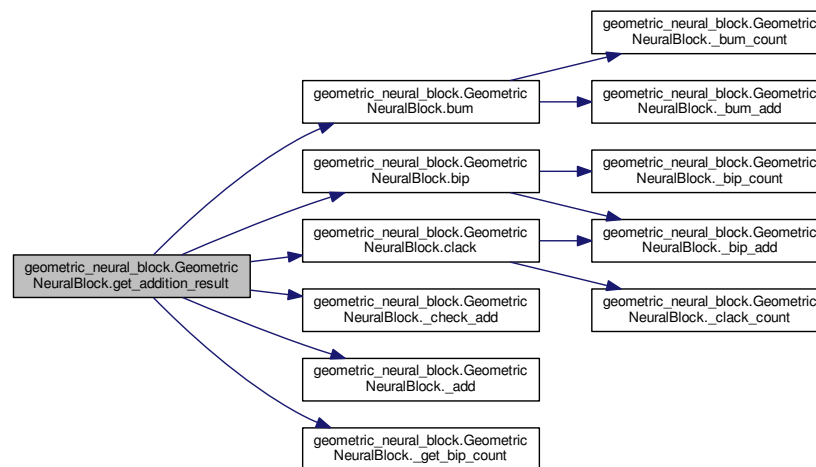
```
def geometric_neural_block.GeometricNeuralBlock.get_addition_result (
    self )
```

Get addition result.

Return values

<i>addition_result</i>	a list of the hearing neuron ids corresponding to the digits of the result
------------------------	--

Here is the call graph for this function:



5.12.2.5 get_equal_sign()

```
def geometric_neural_block.GeometricNeuralBlock.get_equal_sign (
    self )
```

Return the knowledge that represents the equal sign.

Return values

<i>knowledge</i>	Equal sign
------------------	------------

5.12.2.6 `get_operation()`

```
def geometric_neural_block.GeometricNeuralBlock.get_operation (
    self )
```

Get the type of operation to be executed by the block.

Return values

<i>operation</i>	"COUNT" or "ADD"
------------------	------------------

5.12.2.7 `serialize()`

```
def geometric_neural_block.GeometricNeuralBlock.serialize (
    cls,
    obj,
    name )
```

Serialize object and store in given file.

Parameters

<i>cls</i>	GeometricNeuralBlock class
<i>obj</i>	GeometricNeuralBlock object to be serialized
<i>name</i>	Name of the file where the serialization is to be stored

5.12.2.8 `set_add_operator()`

```
def geometric_neural_block.GeometricNeuralBlock.set_add_operator (
    self,
    knowledge )
```

Set the knowledge that represents an addition operator.

In the Brain-CEMISID project this knowledge is just the hearing neuron id that stores the the corresponding pattern

5.12.2.9 `set_equal_sign()`

```
def geometric_neural_block.GeometricNeuralBlock.set_equal_sign (
    self,
    knowledge )
```

Set the knowledge that represents an equal sign.

In the Brain-CEMISID project this knowledge is just the hearing neuron id that stores the the corresponding pattern

5.12.2.10 `set_operation()`

```
def geometric_neural_block.GeometricNeuralBlock.set_operation (
    self,
    operation )
```

Set an operation to be executed by the block.

Parameters

<i>operation</i>	The operation to be executed. Can take on the values "COUNT" or "ADD"
------------------	---

5.12.2.11 `set_zero()`

```
def geometric_neural_block.GeometricNeuralBlock.set_zero (
    self,
    knowledge )
```

Set the knowledge that represents a zero.

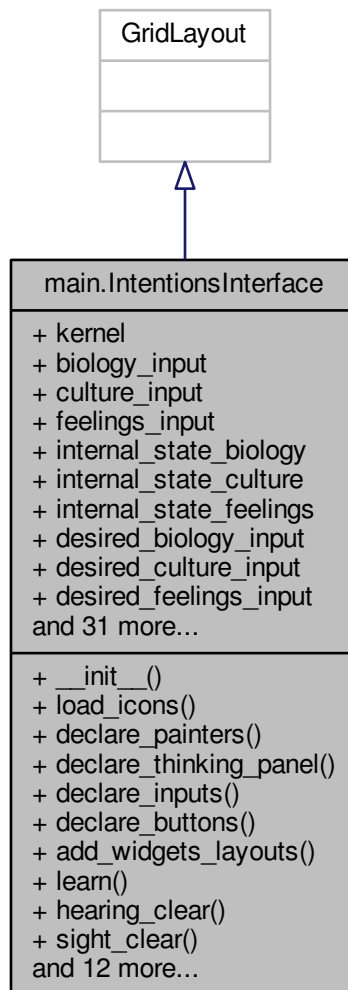
In the Brain-CEMISID project this knowledge is just the hearing neuron id that stores the corresponding pattern

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

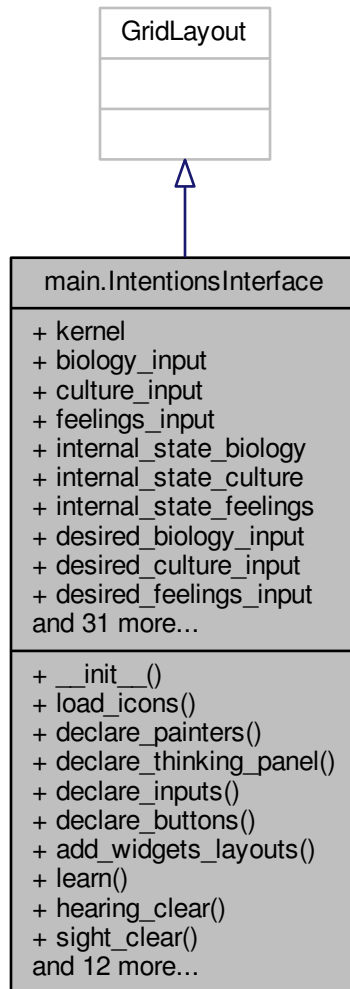
- `geometric_neural_block.py`

5.13 main.IntentionsInterface Class Reference

Inheritance diagram for main.IntentionsInterface:



Collaboration diagram for main.IntentionsInterface:



Public Member Functions

- `def __init__ (self, kwargs)`
- `def load_icons (self)`
- `def declare_painters (self, grid_size)`
- `def declare_thinking_panel (self)`
- `def declare_inputs (self)`
- `def declare_buttons (self)`
- `def add_widgets_layouts (self)`
- `def learn (self, obj)`
- `def hearing_clear (self, obj)`

- def **sight_clear** (self, obj)
- def **bum** (self, obj)
- def **bip** (self, obj)
- def **check** (self, obj)
- def **clack** (self, obj)
- def **pass_kernel_inputs** (self)
- def **show_kernel_outputs** (self)
- def **thinking_clear** (self)
- def **clear** (self, obj)
- def **set_zero** (self, obj)
- def **set_add_operator** (self, obj)
- def **set_equal_sign** (self, obj)
- def **format_backgrounds** (self, obj)

Public Attributes

- **kernel**
- **biology_input**
- **culture_input**
- **feelings_input**
- **internal_state_biology**
- **internal_state_culture**
- **internal_state_feelings**
- **desired_biology_input**
- **desired_culture_input**
- **desired_feelings_input**
- **rows**
- **win_show_uid**
- **win_format_back_uid**
- **img_eye**
- **img_ear**
- **sightPainter**
- **hearingPainter**
- **thinkingPanel**
- **thinking_sight**
- **thinking_hearing**
- **hearing_class_input**
- **sight_clear_btn**
- **hearing_clear_btn**
- **bum_btn**
- **bip_btn**
- **check_btn**
- **clack_btn**
- **episodes_tgl_btn**
- **intentions_tgl_btn**
- **sight_panel**
- **hearingPainter_text**
- **hearing_panel**
- **main_panel**
- **senses_bcf_panel**

- senses_panel
- internal_state_panel
- internal_state_label
- desired_state_panel
- desired_state_label
- desired_internal_states_panel
- buttons_panel

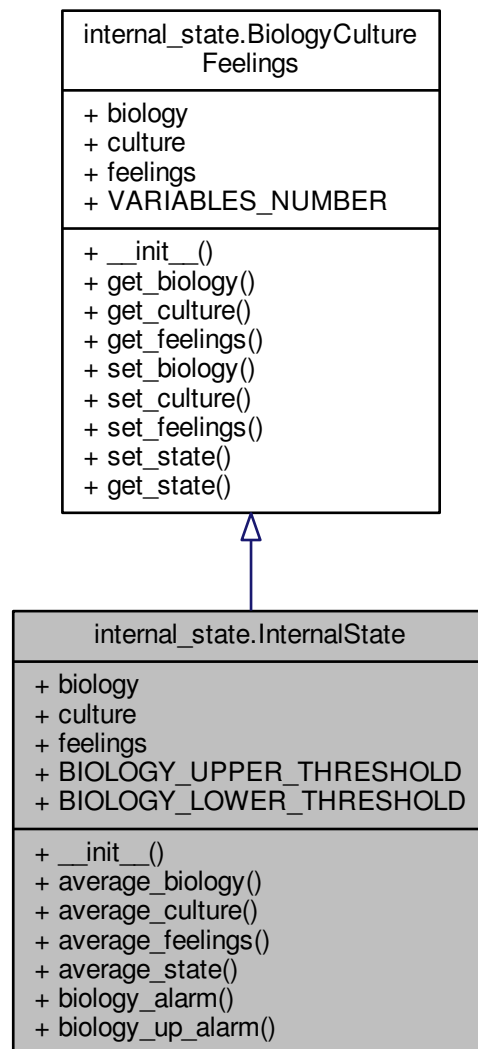
The documentation for this class was generated from the following file:

- main.py

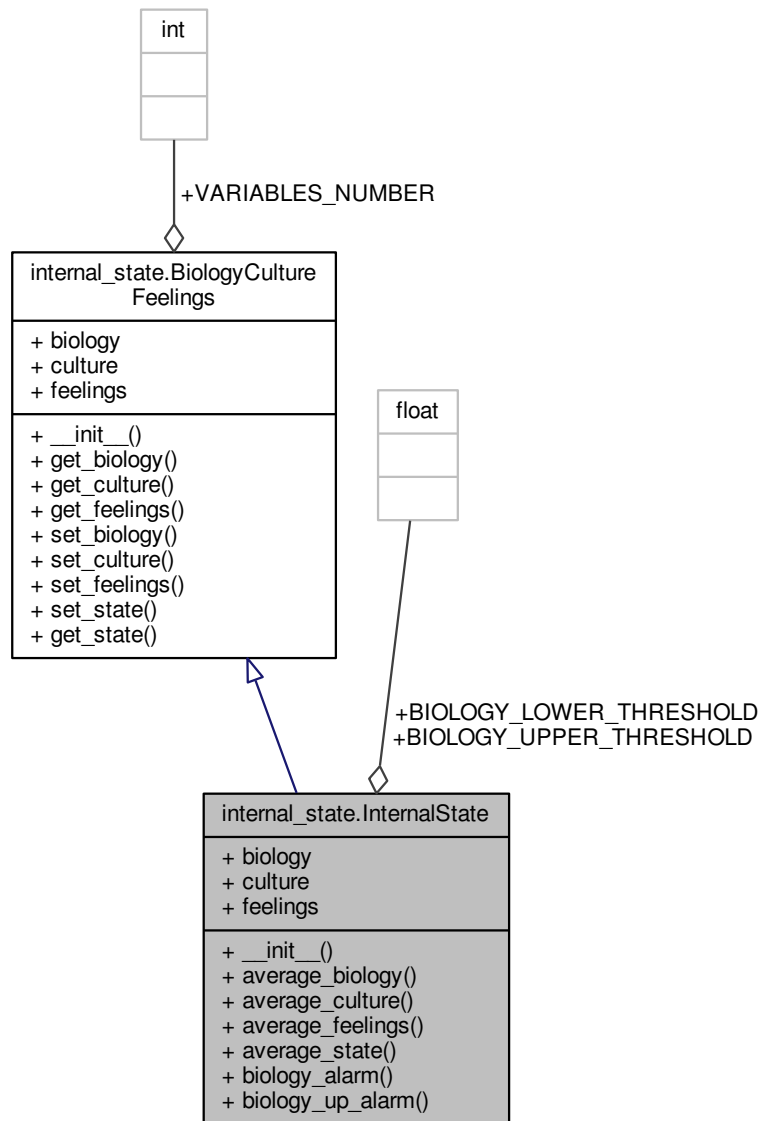
5.14 internal_state.InternalState Class Reference

his class represents a very simplified version of an entity's internal state.

Inheritance diagram for internal_state.InternalState:



Collaboration diagram for internal_state.InternalState:



Public Member Functions

- `def __init__ (self, initial_state=None)`
The constructor.
- `def average_biology (self, val)`
Average biology state with a given value.
- `def average_culture (self, val)`

- *Average culture state with a given value.*
- `def average_feelings (self, val)`
Average feelings state with a given value.
- `def average_state (self, states_vector)`
Average complete state with a given vector of values.
- `def biology_alarm (self)`
Return True if there is a biology alarm, i.e., if the biology component of the internal state is above (below) the BIOLOGY↔Y_UPPER_THRESHOLD (BIOLOGY_LOWER_THRESHOLD)
- `def biology_up_alarm (self)`
Return True if there is an upper biology alarm, i.e., if the biology component of the internal state is above the BIOLOGY↔_UPPER_THRESHOLD.

Public Attributes

- **biology**
- **culture**
- **feelings**

Static Public Attributes

- `float BIOLOGY_UPPER_THRESHOLD = 0.8`
Biology upper threshold.
- `float BIOLOGY_LOWER_THRESHOLD = 0.2`
Biology lower threshold.

5.14.1 Detailed Description

his class represents a very simplified version of an entity's internal state.

It is a type of BCF, because such state has biological, cultural and emotional (here reduced to its less primitive and reflected counterpart 'feelings') components. Average methods are added to model the effect of external influences

5.14.2 Member Function Documentation

5.14.2.1 average_biology()

```
def internal_state.InternalState.average_biology (
    self,
    val )
```

Average biology state with a given value.

Parameters

<i>val</i>	Float from 0 to 1 to be averaged with the stored biology value.
------------	---

Here is the caller graph for this function:



5.14.2.2 average_culture()

```
def internal_state.InternalState.average_culture (
    self,
    val )
```

Average culture state with a given value.

Parameters

<i>val</i>	Float from 0 to 1 to be averaged with the stored culture value.
------------	---

Here is the caller graph for this function:



5.14.2.3 average_feelings()

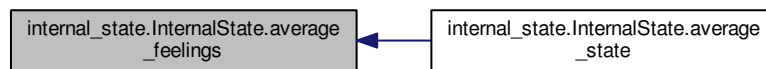
```
def internal_state.InternalState.average_feelings (
    self,
    val )
```

Average feelings state with a given value.

Parameters

<i>val</i>	Float from 0 to 1 to be averaged with the stored feelings value.
------------	--

Here is the caller graph for this function:



5.14.2.4 average_state()

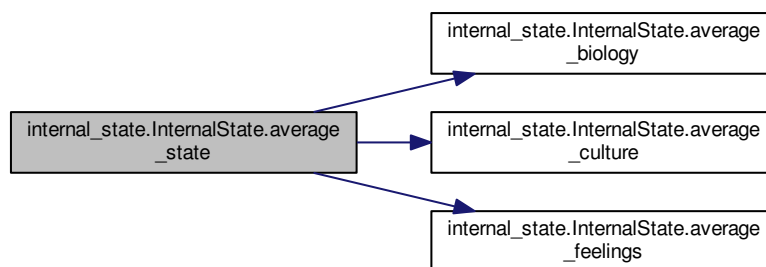
```
def internal_state.InternalState.average_state (
    self,
    states_vector )
```

Average complete state with a given vector of values.

Parameters

<code>states_vector</code>	0 to 1 floats vector to be averaged with the stored state.
----------------------------	--

Here is the call graph for this function:



5.14.2.5 biology_alarm()

```
def internal_state.InternalState.biology_alarm (
    self )
```

Return True if there is a biology alarm, i.e., if the biology component of the internal state is above (below) the BIOLOGY_UPPER_THRESHOLD (BIOLOGY_LOWER_THRESHOLD)

Return values

<i>alarm</i>	Boolean. True if there is an alarm, False in any other case.
--------------	--

5.14.2.6 biology_up_alarm()

```
def internal_state.InternalState.biology_up_alarm (
    self )
```

Return True if there is an upper biology alarm, i.e., if the biology component of the internal state is above the BIOLOGY_UPPER_THRESHOLD.

Return values

<i>up_alarm</i>	Boolean. True if there is an upper biology alarm, False in any other case.
-----------------	--

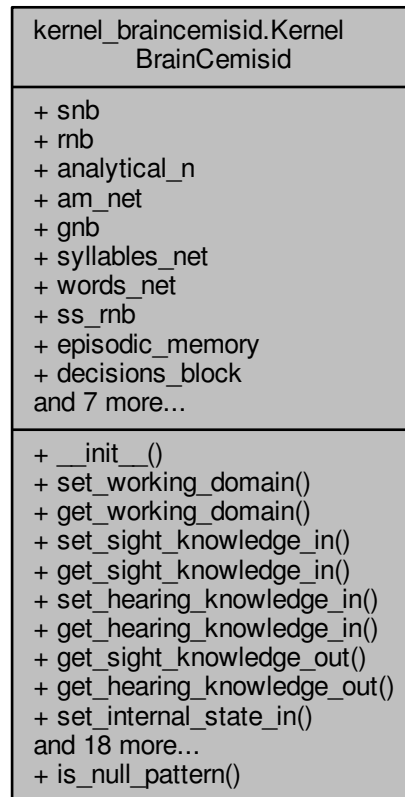
The documentation for this class was generated from the following file:

- internal_state.py

5.15 kernel_braincemisid.KernelBrainCemisid Class Reference

The [KernelBrainCemisid](#) is a major module that envelopes and coordinates interaction between all other project modules (Except for the interface, which obviously must not be part of the kernel, but use the kernel)

Collaboration diagram for kernel_braincemisid.KernelBrainCemisid:



Public Member Functions

- def `__init__`(self)
Kernel contructor.
- def `set_working_domain`(self, domain)
Sets a working domain for the bbcc protocol.
- def `get_working_domain`(self)
Get bbcc protocol working domain.
- def `set_sight_knowledge_in`(self, knowledge)
Set sight knowledge.
- def `get_sight_knowledge_in`(self)
Get sight knowledge.
- def `set_hearing_knowledge_in`(self, knowledge)
Set hearing knowledge.
- def `get_hearing_knowledge_in`(self)

- *Get hearing knowledge.*
- def [get_sight_knowledge_out](#) (self)
 - Get output sight knowledge (Thinking)*
- def [get_hearing_knowledge_out](#) (self)
 - Get output hearing knowledge (Thinking)*
- def [set_internal_state_in](#) (self, states_vector)
 - Set internal state related to an input.*
- def [set_internal_state](#) (self, states_vector)
 - Set internal state (Biology, Culture and Feelings)*
- def [get_internal_state](#) (self)
 - Get internal state.*
- def [feed_internal_state](#) (self, states_vector)
 - Get internal state resulting from an experience and take the average with the current internal state.*
- def [set_desired_state](#) (self, states_vector)
 - Set desired state (Biology, Culture and Feelings)*
- def [get_desired_state](#) (self)
 - Get desired state.*
- def [disable_bbcc](#) (self)
 - Disable bbcc protocol so that Check and Clack can be used as standalone actions.*
- def [bum](#) (self)
 - Start bbcc protocol.*
- def [bip](#) (self)
 - Bip part of bbcc protocol (See bbcc protocol description)*
- def [check](#) (self)
 - Check if there is knowledge associated with the given sequence of patterns from bbcc protocol when self._enable_bbcc is True.*
- def [clack](#) (self)
 - Execute clack action of bbcc protocol when self._enable_bbcc is True or Learn a piece of RbfKnowledge coming from the senses when self._enable_bbcc is False.*
- def [recognize](#) (self)
 - Recognize either hearing or sight patterns.*
- def [hearing_recognize](#) (self)
 - Recognize hearing pattern.*
- def [sight_recognize](#) (self)
 - Recognize sight pattern.*
- def [learn](#) (self)
 - Learn patterns.*
- def [erase_all_knowledge](#) (self)
 - Erase all knowlege.*
- def [set_add_operator](#) (self)
- def [set_equal_sign](#) (self)
- def [set_zero](#) (self)
 - Set some already learned pattern as the zero number.*

Static Public Member Functions

- def [is_null_pattern](#) (pattern)

Public Attributes

- **snb**
- **rnb**
- **analytical_n**
- **am_net**
- **gnb**
- **syllables_net**
- **words_net**
- **ss_rnb**
- [episodic_memory](#)

INTENTIONS #####

- **decisions_block**
- **internal_state**
- **desired_state**
- **s_knowledge_out**
- **h_knowledge_out**
- **s_knowledge_in**
- **h_knowledge_in**
- [state](#)

INTENTIONS ##### Get memory related to hearing id.

5.15.1 Detailed Description

The [KernelBrainCemisid](#) is a major module that envelopes and coordinates interaction between all other project modules (Except for the interface, which obviously must not be part of the kernel, but use the kernel)

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

- `kernel_braincemisid.py`

5.16 multiclass_single_layer_network.MulticlassSingleLayerNetwork Class Reference

Collaboration diagram for multiclass_single_layer_network.MulticlassSingleLayerNetwork:

multiclass_single_layer_network.MulticlassSingleLayerNetwork
+ weights + inputs + outputs + learning_rate
+ __init__() + get_inputs() + set_inputs() + get_outputs() + set_learning_rate() + get_learning_rate() + set_activation_function() + training() + update_weights()

Public Member Functions

- def `__init__` (self, inputs_number, outputs_number)
Multiclass PerceptronNetwork constructor.
- def `get_inputs` (self)
Get net inputs.
- def `set_inputs` (self, new_inputs)
Set net inputs.
- def `get_outputs` (self)
Calculate and get outputs.
- def `set_learning_rate` (self, learning_rate)
Set net learning rate according to the Widrow-Hoff equation.
- def `get_learning_rate` (self)
Get learning rate.
- def `set_activation_function` (self, new_func)
Set activation function.
- def `training` (self, training_set)
Train network.
- def `update_weights` (self, input_vector, error)
Update weights in order to match desired output by applying the Widrow-Hoff equation.

Public Attributes

- **weights**
- **inputs**
- **outputs**
- **learning_rate**

5.16.1 Constructor & Destructor Documentation

5.16.1.1 `__init__()`

```
def multiclass_single_layer_network.MulticlassSingleLayerNetwork.__init__ (
    self,
    inputs_number,
    outputs_number )
```

Multiclass PerceptronNetwork constructor.

Parameters

<i>inputs_number</i>	Integer. Number of inputs in perceptron network
<i>outputs_number</i>	Integer. Number of outputs in perceptron network

5.16.2 Member Function Documentation

5.16.2.1 `get_inputs()`

```
def multiclass_single_layer_network.MulticlassSingleLayerNetwork.get_inputs (
    self )
```

Get net inputs.

Return values

<i>inputs</i>	Floats vector
---------------	---------------

5.16.2.2 `get_learning_rate()`

```
def multiclass_single_layer_network.MulticlassSingleLayerNetwork.get_learning_rate (
    self )
```

Get learning rate.

Return values

<i>rate</i>	Float
-------------	-------

5.16.2.3 get_outputs()

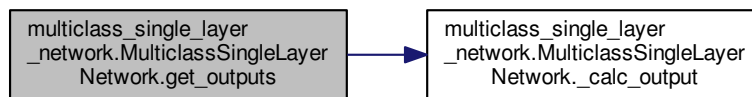
```
def multiclass_single_layer_network.MulticlassSingleLayerNetwork.get_outputs (
    self )
```

Calculate and get outputs.

Return values

<i>outputs</i>	Floats vector
----------------	---------------

Here is the call graph for this function:



5.16.2.4 set_activation_function()

```
def multiclass_single_layer_network.MulticlassSingleLayerNetwork.set_activation_function (
    self,
    new_func )
```

Set activation function.

Parameters

<i>new_func</i>	Function name. New activation function
-----------------	--

5.16.2.5 set_inputs()

```
def multiclass_single_layer_network.MulticlassSingleLayerNetwork.set_inputs (
    self,
    new_inputs )
```

Set net inputs.

Return values

<i>inputs</i>	Floats vector
---------------	---------------

5.16.2.6 set_learning_rate()

```
def multiclass_single_layer_network.MulticlassSingleLayerNetwork.set_learning_rate (
    self,
    learning_rate )
```

Set net learning rate according to the Widrow-Hoff equation.

Float.

5.16.2.7 training()

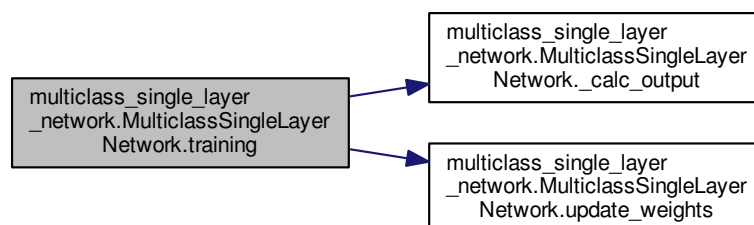
```
def multiclass_single_layer_network.MulticlassSingleLayerNetwork.training (
    self,
    training_set )
```

Train network.

Parameters

<i>training_set</i>	Network training set. See tests in code file for an example.
---------------------	--

Here is the call graph for this function:



5.16.2.8 update_weights()

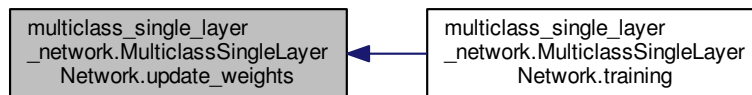
```
def multiclass_single_layer_network.MulticlassSingleLayerNetwork.update_weights (
    self,
    input_vector,
    error )
```

Update weights in order to match desired output by applying the Widrow-Hoff equation.

Parameters

<i>input_vector</i>	Input vector
<i>error</i>	Prediction error

Here is the caller graph for this function:

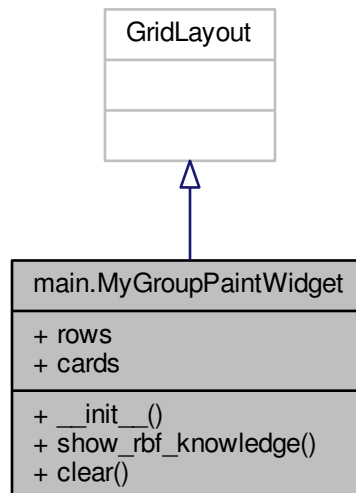


The documentation for this class was generated from the following file:

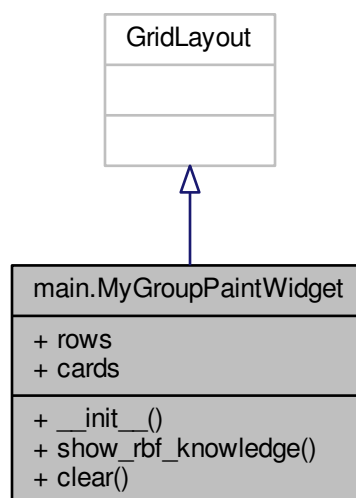
- `multiclass_single_layer_network.py`

5.17 main.MyGroupPaintWidget Class Reference

Inheritance diagram for main.MyGroupPaintWidget:



Collaboration diagram for main.MyGroupPaintWidget:



Public Member Functions

- def **__init__** (self, kwargs)
- def **show_rbf_knowledge** (self, knowledge_or_vector)
- def **clear** (self)

Public Attributes

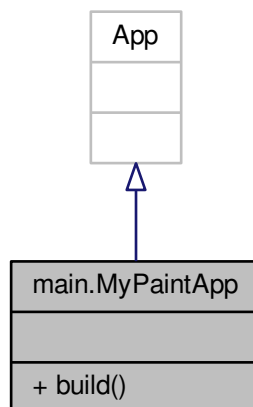
- **rows**
- **cards**

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

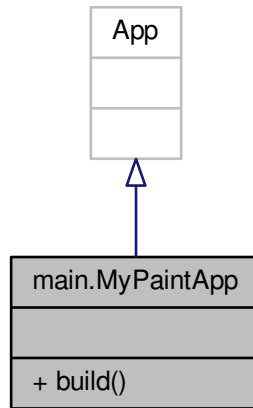
- main.py

5.18 main.MyPaintApp Class Reference

Inheritance diagram for main.MyPaintApp:



Collaboration diagram for main.MyPaintApp:



Public Member Functions

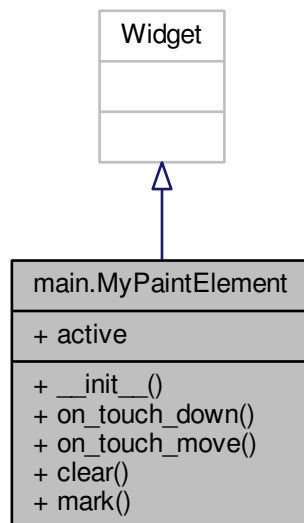
- def **build** (self)

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

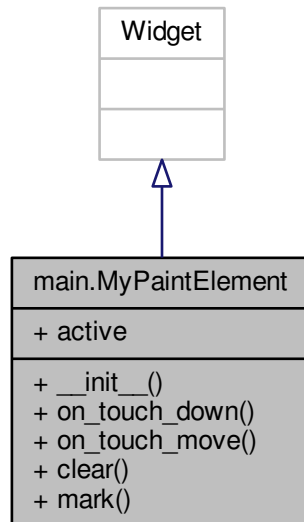
- main.py

5.19 main.MyPaintElement Class Reference

Inheritance diagram for main.MyPaintElement:



Collaboration diagram for main.MyPaintElement:



Public Member Functions

- `def __init__ (self, kwargs)`
- `def on_touch_down (self, touch)`
- `def on_touch_move (self, touch)`
- `def clear (self, color)`
- `def mark (self)`

Public Attributes

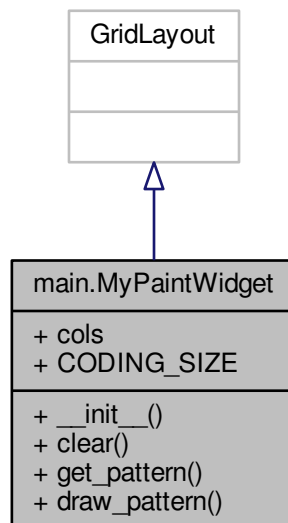
- `active`

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

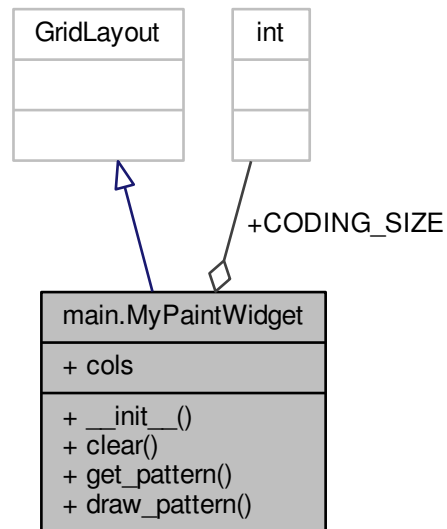
- `main.py`

5.20 main.MyPaintWidget Class Reference

Inheritance diagram for main.MyPaintWidget:



Collaboration diagram for `main.MyPaintWidget`:



Public Member Functions

- `def __init__ (self, size, kwargs)`
- `def clear (self)`
- `def get_pattern (self)`
- `def draw_pattern (self, pattern)`

Public Attributes

- `cols`

Static Public Attributes

- `int CODING_SIZE = 4`

5.20.1 Member Function Documentation

5.20.1.1 draw_pattern()

```
def main.MyPaintWidget.draw_pattern (
    self,
    pattern )
```

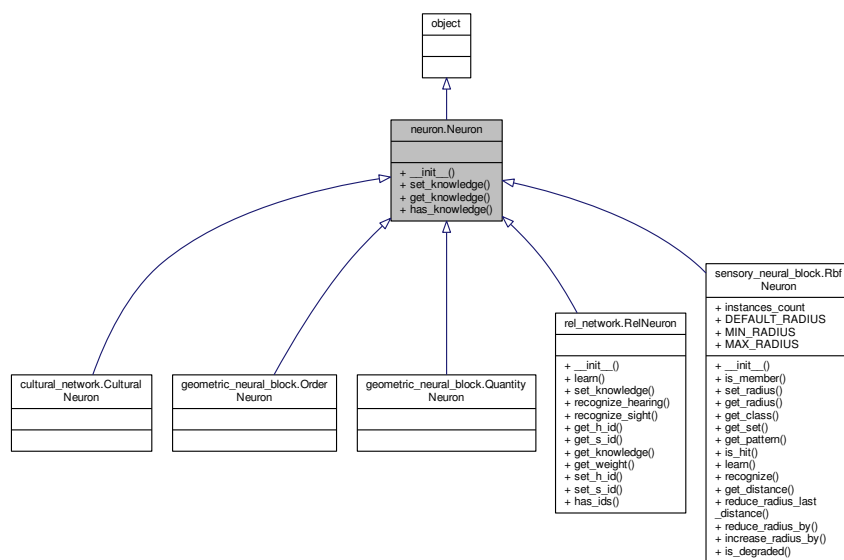
Draw given pattern in painter

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

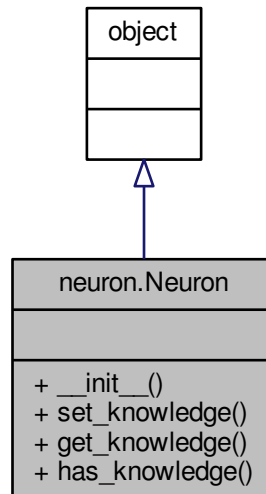
- main.py

5.21 neuron.Neuron Class Reference

Inheritance diagram for neuron.Neuron:



Collaboration diagram for neuron.Neuron:



Public Member Functions

- def **`__init__`** (self, knowledge=None)
- def **`set_knowledge`** (self, knowledge)
- def **`get_knowledge`** (self)
- def **`has_knowledge`** (self)

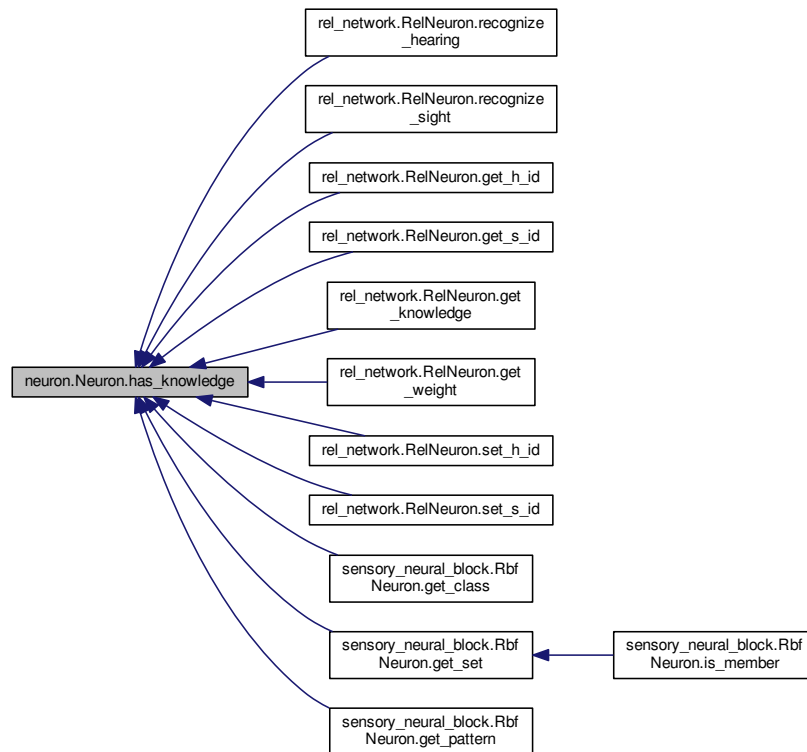
5.21.1 Member Function Documentation

5.21.1.1 `has_knowledge()`

```
def neuron.Neuron.has_knowledge (
    self )
```

Return true if the neuron has already learned some kind of knowledge
and false in any other case

Here is the caller graph for this function:

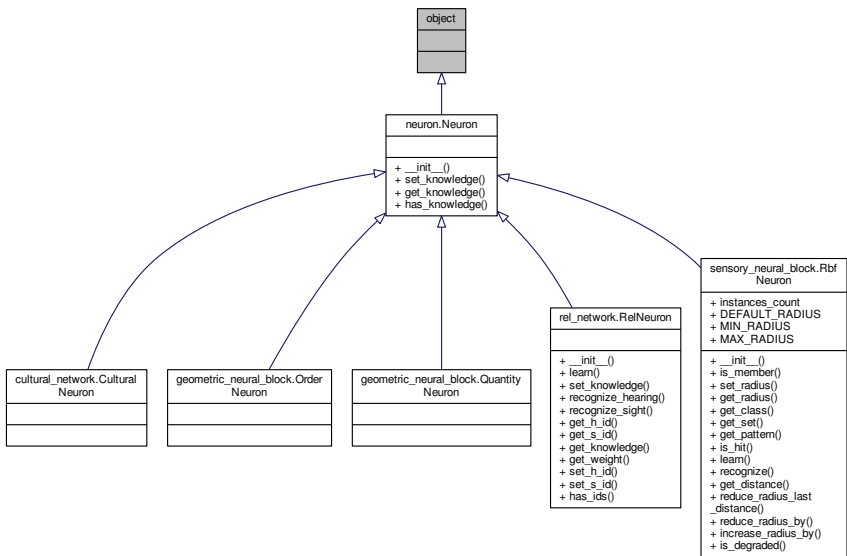


The documentation for this class was generated from the following file:

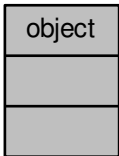
- neuron.py

5.22 object Class Reference

Inheritance diagram for object:



Collaboration diagram for object:



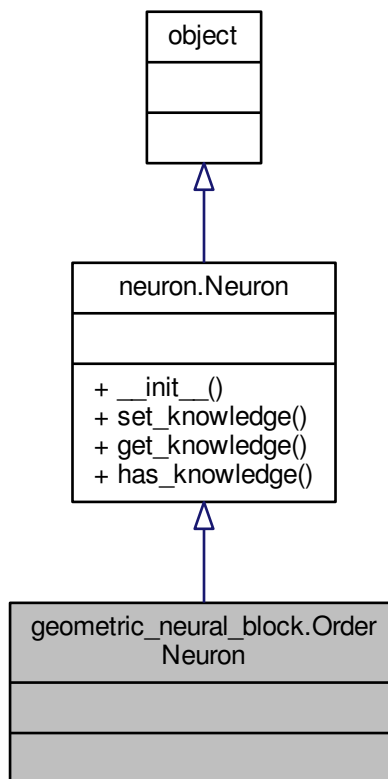
The documentation for this class was generated from the following file:

- neuron.py

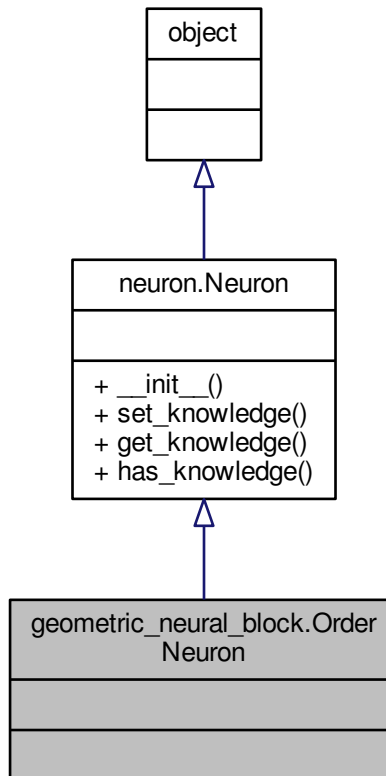
5.23 geometric_neural_block.OrderNeuron Class Reference

The [QuantityNeuron](#) class is a kind of neuron whose position in a [QuantityOrderNetwork](#) signals certain ordinality.

Inheritance diagram for geometric_neural_block.OrderNeuron:



Collaboration diagram for `geometric_neural_block.OrderNeuron`:



Additional Inherited Members

5.23.1 Detailed Description

The [QuantityNeuron](#) class is a kind of neuron whose position in a [QuantityOrderNetwork](#) signals certain ordinality.

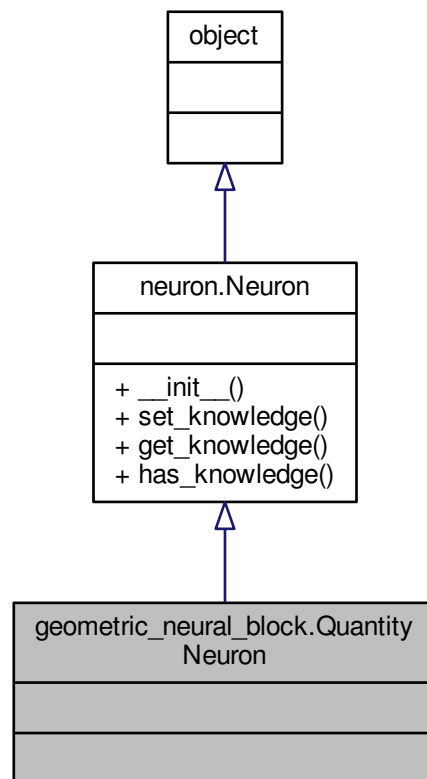
The documentation for this class was generated from the following file:

- `geometric_neural_block.py`

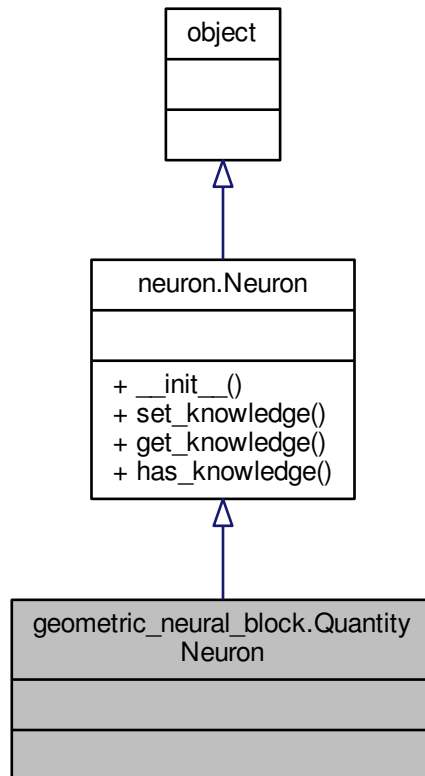
5.24 `geometric_neural_block.QuantityNeuron` Class Reference

The [QuantityNeuron](#) class is a kind of neuron that signals the cardinality of its relation in a [QuantityOrderGroup](#).

Inheritance diagram for geometric_neural_block.QuantityNeuron:



Collaboration diagram for `geometric_neural_block.QuantityNeuron`:



Additional Inherited Members

5.24.1 Detailed Description

The [QuantityNeuron](#) class is a kind of neuron that signals the cardinality of its relation in a [QuantityOrderGroup](#).

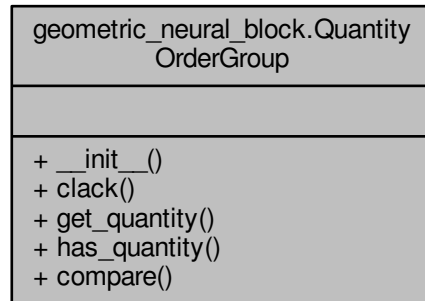
The documentation for this class was generated from the following file:

- `geometric_neural_block.py`

5.25 `geometric_neural_block.QuantityOrderGroup` Class Reference

A [QuantityOrderGroup](#) is a pair composed of an [OrderNeuron](#) and a [QuantityNeuron](#).

Collaboration diagram for `geometric_neural_block.QuantityOrderGroup`:



Public Member Functions

- `def __init__ (self)`
The constructor.
- `def clack (self, knowledge)`
Store a quantity.
- `def get_quantity (self)`
Retrieve a quantity.
- `def has_quantity (self)`
Return true if the group has a quantity stored and false in any other case.
- `def compare (self, knowledge)`
Compare certain piece of knowledge with the quantity stored in the group.

5.25.1 Detailed Description

A [QuantityOrderGroup](#) is a pair composed of an [OrderNeuron](#) and a [QuantityNeuron](#).

It is the basic element of a [QuantityOrderNetwork](#).

5.25.2 Member Function Documentation

5.25.2.1 `compare()`

```
def geometric_neural_block.QuantityOrderGroup.compare (
    self,
    knowledge )
```

Compare certain piece of knowledge with the quantity stored in the group.

Parameters

<i>knowledge</i>	The knowledge to be compared
------------------	------------------------------

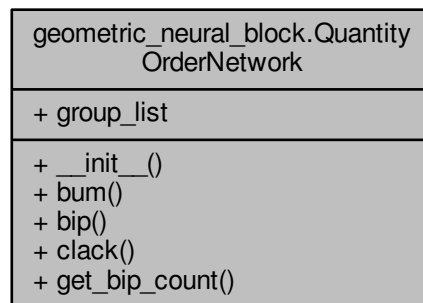
The documentation for this class was generated from the following file:

- `geometric_neural_block.py`

5.26 `geometric_neural_block.QuantityOrderNetwork` Class Reference

A set of [QuantityOrderGroup](#) instances that act together in order to store Order and Quantity information.

Collaboration diagram for `geometric_neural_block.QuantityOrderNetwork`:



Public Member Functions

- `def __init__(self)`
The constructor.
- `def bum(self)`
Start count.
- `def bip(self)`
Point to next [QuantityOrderGroup](#).
- `def clack(self, knowledge=None)`
Store quantity in currently pointed [QuantityOrderGroup](#).
- `def get_bip_count(self, knowledge)`
Get position of the currently pointed [QuantityOrderGroup](#) as a number of bips since the beginning (`bum`).

Public Attributes

- [group_list](#)

Set of [QuantityOrderGroup](#) instances.

5.26.1 Detailed Description

A set of [QuantityOrderGroup](#) instances that act together in order to store Order and Quantity information.

5.26.2 Member Function Documentation

5.26.2.1 `get_bip_count()`

```
def geometric_neural_block.QuantityOrderNetwork.get_bip_count (
    self,
    knowledge )
```

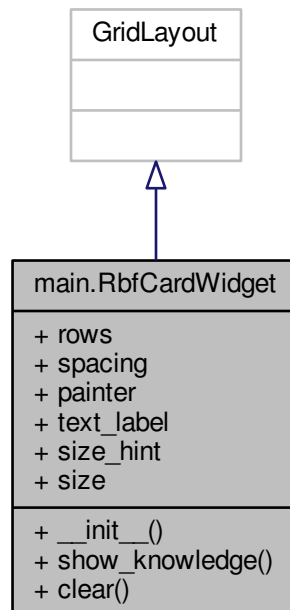
Get position of the currently pointed [QuantityOrderGroup](#) as a number of *bips* since the beginning (*bum*).

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

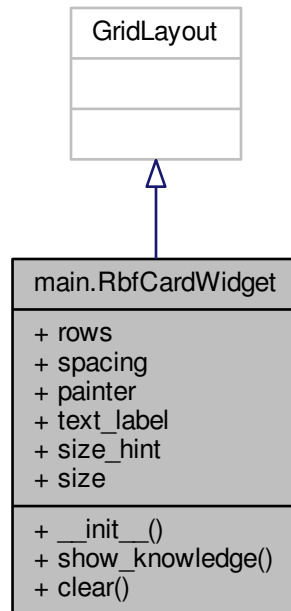
- `geometric_neural_block.py`

5.27 main.RbfCardWidget Class Reference

Inheritance diagram for main.RbfCardWidget:



Collaboration diagram for main.RbfCardWidget:



Public Member Functions

- `def __init__ (self, width, kwargs)`
- `def show_knowledge (self, knowledge)`
- `def clear (self)`

Public Attributes

- **rows**
- **spacing**
- **painter**
- **text_label**
- **size_hint**
- **size**

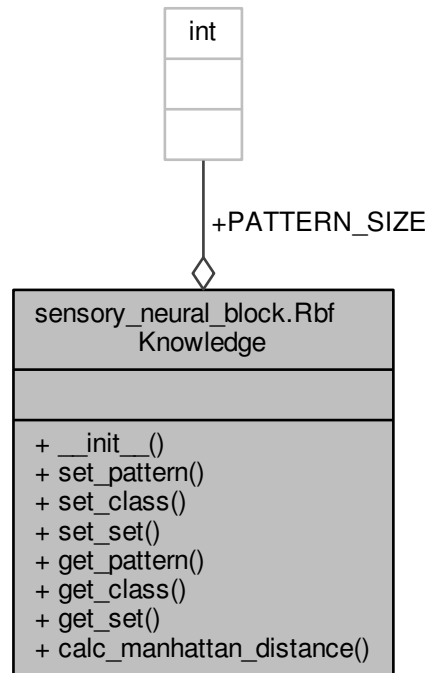
The documentation for this class was generated from the following file:

- `main.py`

5.28 sensory_neural_block.RbfKnowledge Class Reference

RBF knowledge.

Collaboration diagram for sensory_neural_block.RbfKnowledge:



Public Member Functions

- def `__init__` (self, rbf_pattern, rbf_class, rbf_set="NoSet")
The constructor.
- def `set_pattern` (self, pattern)
Set pattern.
- def `set_class` (self, rbf_class)
Set pattern class.
- def `set_set` (self, rbf_set)
Set pattern set.
- def `get_pattern` (self)
Get stored pattern.
- def `get_class` (self)
Get stored pattern class.
- def `get_set` (self)
Get stored pattern set.
- def `calc_manhattan_distance` (self, pattern_or_knowledge)

Static Public Attributes

- int `PATTERN_SIZE` = 4
Size of data or knowledge in bytes.

5.28.1 Detailed Description

RBF knowledge.

A tuple composed of a pattern, a class and a set. The class also provides a method for calculating the Manhattan distance between its pattern and the pattern of another [RbfKnowledge](#) instance

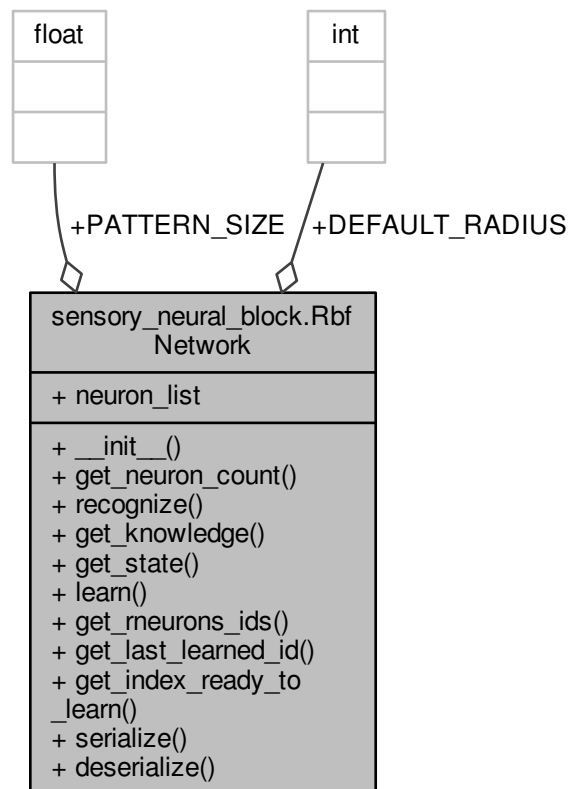
The documentation for this class was generated from the following file:

- `sensory_neural_block.py`

5.29 `sensory_neural_block.RbfNetwork` Class Reference

RBF Neural Network.

Collaboration diagram for `sensory_neural_block.RbfNetwork`:



Public Member Functions

- def `__init__` (self, neuron_count)
Class constructor, takes 'neuron_count' as parameter for setting network size.
- def `get_neuron_count` (self)
get number of neurons in network
- def `recognize` (self, pattern)
Recognize a given pattern.
- def `get_knowledge` (self)
Get [RbfKnowledge](#) related to last recognized pattern.
- def `get_state` (self)
Get network state.
- def `learn` (self, knowledge)
Learn an instance of [RbfKnowledge](#).
- def `get_neurons_ids` (self)
Get ids of recognizing set neurons.
- def `get_last_learned_id` (self)
Get id of neuron affected in the last learning process.
- def `get_index_ready_to_learn` (self)
Get index of ready-to-learn neuron.
- def `serialize` (cls, obj, name)
Serialize object and store in given file.
- def `deserialize` (cls, name)
Deserialize object stored in given file.

Public Attributes

- `neuron_list`

Static Public Attributes

- float `PATTERN_SIZE` = 4.0
Size of data or knowledge in bytes.
- int `DEFAULT_RADIUS` = 5
Default radius.

5.29.1 Detailed Description

RBF Neural Network.

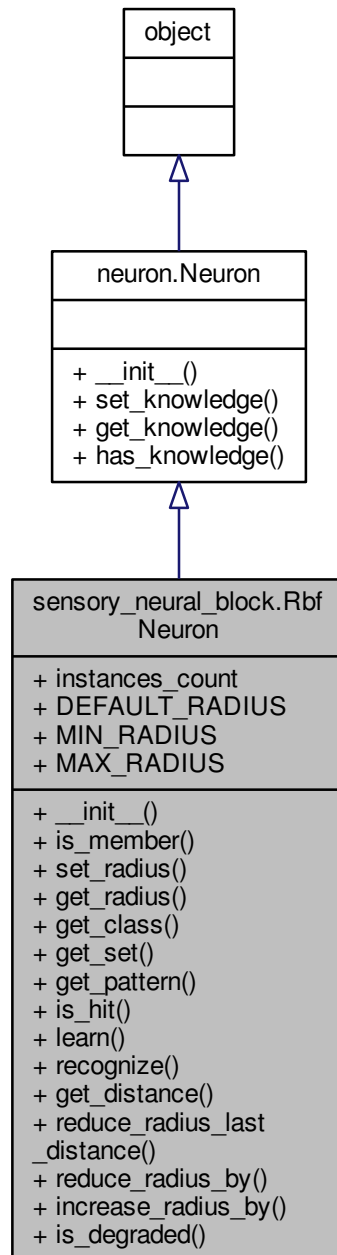
The documentation for this class was generated from the following file:

- `sensory_neural_block.py`

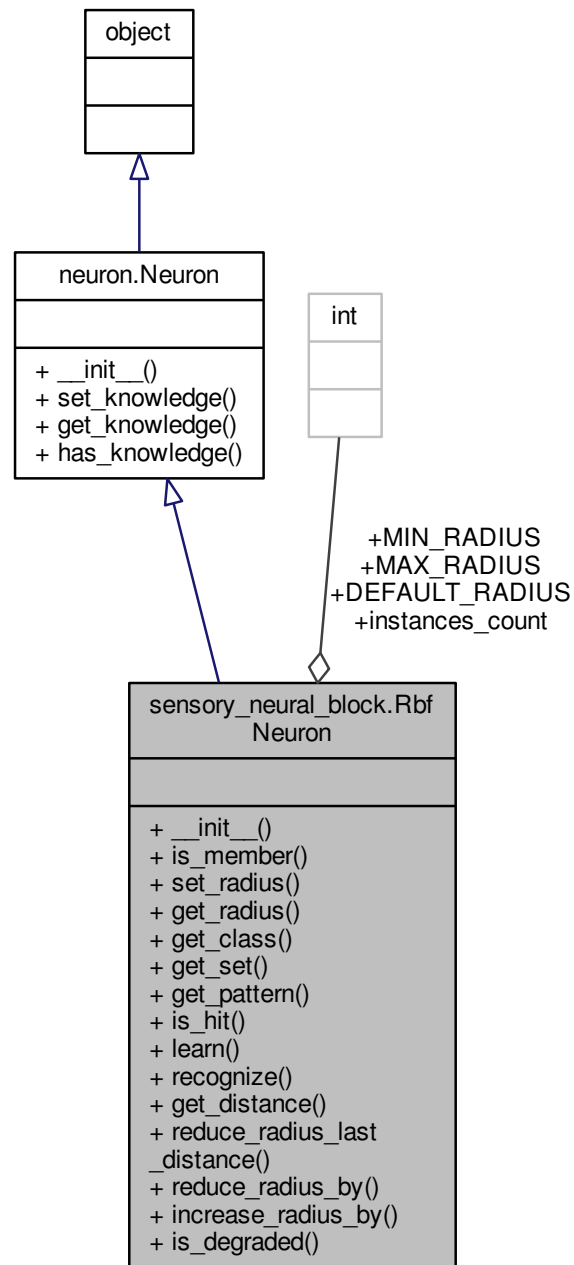
5.30 sensory_neural_block.RbfNeuron Class Reference

Neuron that stores [RbfKnowledge](#).

Inheritance diagram for sensory_neural_block.RbfNeuron:



Collaboration diagram for sensory_neural_block.RbfNeuron:



Public Member Functions

- `def __init__(self)`

- *Class constructor.*
- def `is_member` (self, test_set)
Returns whether neuron is member of the set.
- def `set_radius` (self, radius)
Sets neuron radius.
- def `get_radius` (self)
Get neuron radius.
- def `get_class` (self)
Get class of stored `RbfKnowledge` instance.
- def `get_set` (self)
Get set of stored `RbfKnowledge` instance.
- def `get_pattern` (self)
Get pattern of stored `RbfKnowledge` instance.
- def `is_hit` (self)
Return True if last call to `recognize()` was a hit and False in any other case.
- def `learn` (self, knowledge)
Learns a new piece of knowledge.
- def `recognize` (self, pattern)
Recognize a piece of knowledge.
- def `get_distance` (self)
Get distance to last instance or `RbfKnowledge` pattern that tried to be recognized.
- def `reduce_radius_last_distance` (self)
Reduce radius by las recognition process's distance.
- def `reduce_radius_by` (self, value)
Reduce neuron radius by certain amount.
- def `increase_radius_by` (self, value)
Increase neuron radius by certain amount.
- def `is_degraded` (self)
Return whether neuron is degraded.

Static Public Attributes

- int `instances_count` = 0
Number of class instances.
- int `DEFAULT_RADIUS` = 10
Default radius.
- int `MIN_RADIUS` = 1
Minimum radius before neuron is degraded.
- int `MAX_RADIUS` = 50
Maximum radius.

5.30.1 Detailed Description

Neuron that stores `RbfKnowledge`.

This class stores an instance of `RbfKnowledge` at its center and uses a radius value to determine whether or not it recognizes a given pattern

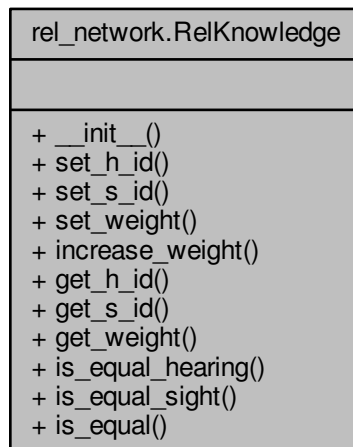
The documentation for this class was generated from the following file:

- `sensory_neural_block.py`

5.31 rel_network.RelKnowledge Class Reference

Relational knowledge is a 3-tuple that relate a sight RbfNeuron id, a hearing RbfNeuron id and a weight.

Collaboration diagram for rel_network.RelKnowledge:



Public Member Functions

- def `__init__` (self, h_id, s_id, weight=0)
Create [RelKnowledge](#) instance given a hearing id (id_h), sight id (id_s) and weight which defaults to zero.
- def `set_h_id` (self, h_id)
Set hearing id.
- def `set_s_id` (self, s_id)
Set sight id.
- def `set_weight` (self, w)
Set weight.
- def `increase_weight` (self, amount=1)
Increase weight of relation by a given value.
- def `get_h_id` (self)
Get hearing id of relation.
- def `get_s_id` (self)
Get sight id of relation.
- def `get_weight` (self)
Get weight of relation.
- def `is_equal_hearing` (self, h_id)
Return True if knowledge's hearing id is equal to given parameter h_id and False in any other case.

- def `is_equal_sight` (self, s_id)
Return True if knowledge sight id is equal to given parameter s_id and False in any other case.
- def `is_equal` (self, h_id, s_id)
Return true if knowledge's sight id is equal to given parameter s_id and knowledge's hearing id is equal to given parameter h_id.

5.31.1 Detailed Description

Relational knowledge is a 3-tuple that relate a sight RbfNeuron id, a hearing RbfNeuron id and a weight.

5.31.2 Member Function Documentation

5.31.2.1 `get_h_id()`

```
def rel_network.RelKnowledge.get_h_id (
    self )
```

Get hearing id of relation.

Return values

h_{id}	Integer. Hearing id.
----------	----------------------

5.31.2.2 `get_s_id()`

```
def rel_network.RelKnowledge.get_s_id (
    self )
```

Get sight id of relation.

Return values

s_{id}	Integer. Sight id.
----------	--------------------

5.31.2.3 `get_weight()`

```
def rel_network.RelKnowledge.get_weight (
    self )
```

Get weight of relation.

Return values

<i>weight</i>	Integer.
---------------	----------

5.31.2.4 increase_weight()

```
def rel_network.RelKnowledge.increase_weight (
    self,
    amount = 1 )
```

Increase weight of relation by a given value.

Parameters

<i>amount</i>	Integer Optional, 1 by default
---------------	--------------------------------

5.31.2.5 is_equal()

```
def rel_network.RelKnowledge.is_equal (
    self,
    h_id,
    s_id )
```

Return true if knowledge's sight id is equal to given parameter s_id and knowledge's hearing id is equal to given parameter h_id.

Return false in any other case

5.31.2.6 set_h_id()

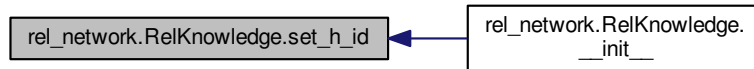
```
def rel_network.RelKnowledge.set_h_id (
    self,
    h_id )
```

Set hearing id.

Parameters

<i>h_id</i>	Integer. Hearing id.
-------------	----------------------

Here is the caller graph for this function:



5.31.2.7 `set_s_id()`

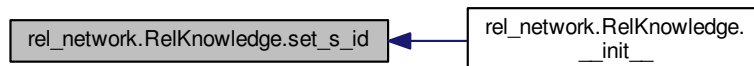
```
def rel_network.RelKnowledge.set_s_id (  
    self,  
    s_id )
```

Set sight id.

Parameters

$s \leftrightarrow$ _id	Integer. Sight id.
----------------------------	--------------------

Here is the caller graph for this function:



5.31.2.8 `set_weight()`

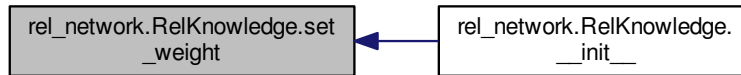
```
def rel_network.RelKnowledge.set_weight (  
    self,  
    w )
```

Set weight.

Parameters

<i>w</i>	Integer. Weight.
----------	------------------

Here is the caller graph for this function:



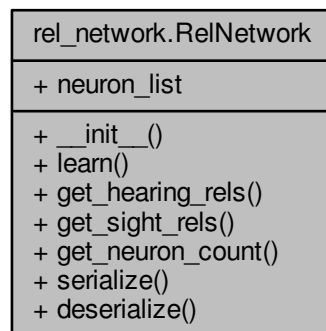
The documentation for this class was generated from the following file:

- `rel_network.py`

5.32 rel_network.RelNetwork Class Reference

Relational network.

Collaboration diagram for `rel_network.RelNetwork`:



Public Member Functions

- `def __init__(self, neuron_count)`
The constructor.
- `def learn(self, knowledge)`
Learn new knowledge in ready-to-learn neuron.
- `def get_hearing_rels(self, h_id)`

Return a list of all knowledge in net such that it has parameter `h_id` as hearing id.

- def `get_sight_rels` (self, `s_id`)

Return a list of all knowledge in net such that it has parameter `s_id` as sight id.

- def `get_neuron_count` (self)

Returns number of neurons in network.

- def `serialize` (cls, obj, name)

Serialize object and store it in given file.

- def `deserialize` (cls, name)

Deserialize object stored in given file.

Public Attributes

- `neuron_list`

5.32.1 Detailed Description

Relational network.

5.32.2 Constructor & Destructor Documentation

5.32.2.1 `__init__()`

```
def rel_network.RelNetwork.__init__ (
    self,
    neuron_count )
```

The constructor.

Parameters

<code>neuron_count</code>	Network size
---------------------------	--------------

5.32.3 Member Function Documentation

5.32.3.1 `deserialize()`

```
def rel_network.RelNetwork.deserialize (
    cls,
    name )
```

Deserialize object stored in given file.

Parameters

<i>cls</i>	RelNetwork class
<i>name</i>	Name of the file where the object is serialized

5.32.3.2 get_hearing_rels()

```
def rel_network.RelNetwork.get_hearing_rels (
    self,
    h_id )
```

Return a list of all knowledge in net such that it has parameter h_id as hearing id.

Return values

<i>hearing_rels</i>	RelKnowledge vector
---------------------	-------------------------------------

5.32.3.3 get_neuron_count()

```
def rel_network.RelNetwork.get_neuron_count (
    self )
```

Returns number of neurons in network.

Return values

<i>count</i>	Integer.
--------------	----------

5.32.3.4 get_sight_rels()

```
def rel_network.RelNetwork.get_sight_rels (
    self,
    s_id )
```

Return a list of all knowledge in net such that it has parameter s_id as sight id.

Return values

<i>sight_rels</i>	RelKnowledge vector
-------------------	-------------------------------------

5.32.3.5 learn()

```
def rel_network.RelNetwork.learn (
    self,
    knowledge )
```

Learn new knowledge in ready-to-learn neuron.

Parameters

<i>knowledge</i>	RelKnowledge to be learned.
------------------	---

5.32.3.6 serialize()

```
def rel_network.RelNetwork.serialize (
    cls,
    obj,
    name )
```

Serialize object and store it in given file.

Parameters

<i>cls</i>	RelNetwork class
<i>obj</i>	RelNetwork object to be serialized
<i>name</i>	Name of the file where the serialization is to be stored

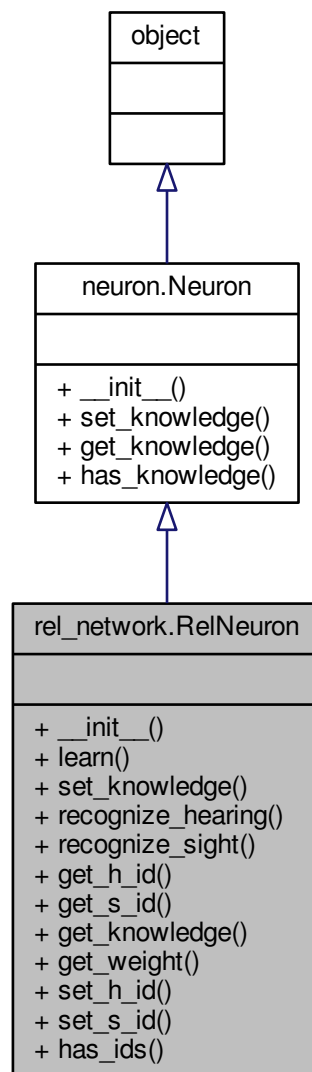
The documentation for this class was generated from the following file:

- `rel_network.py`

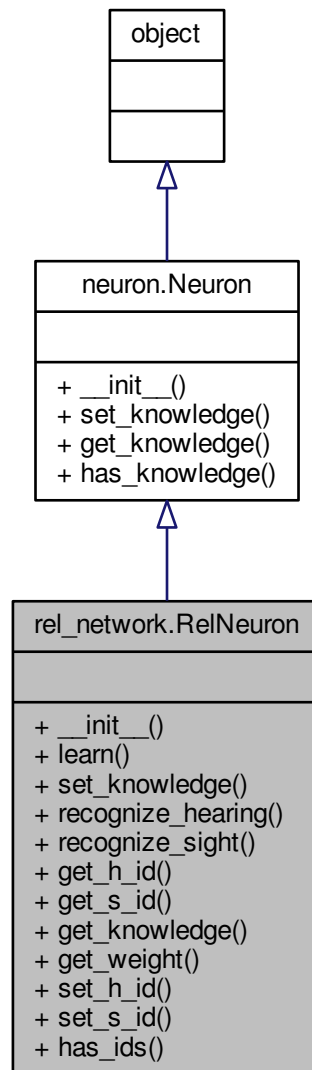
5.33 rel_network.RelNeuron Class Reference

Relational neuron.

Inheritance diagram for rel_network.RelNeuron:



Collaboration diagram for rel_network.RelNeuron:



Public Member Functions

- `def __init__(self)`
The constructor.
- `def learn(self, knowledge)`
Set knowledge of type [RelKnowledge](#).
- `def set_knowledge(self, knowledge)`
Set knowledge of type [RelKnowledge](#).

- def `recognize_hearing` (self, h_id)
Return True if h_id is recognized as the hearing-id part of the RelKnowledge.
- def `recognize_sight` (self, s_id)
Return true if s_id is recognized as the sight-id part of the relational knowledge.
- def `get_h_id` (self)
Return hearing id if neuron has knowledge and an object of type None in any other case.
- def `get_s_id` (self)
Return sight id if neuron has knowledge and an object of type None in any other case.
- def `get_knowledge` (self)
Returns knowledge stored by neuron if neuron has knowledge, and None object in any other case.
- def `get_weight` (self)
Return weight of relation if neuron has knowledge and an object of type None in any other case.
- def `set_h_id` (self, h_id)
Set hearing id if neuron has knowledge.
- def `set_s_id` (self, s_id)
Set sight id if neuron has knowledge.
- def `has_ids` (self, h_id, s_id)
Return true if neuron has h_id and s_id as hearing and sight ids respectively.

5.33.1 Detailed Description

Relational neuron.

5.33.2 Member Function Documentation

5.33.2.1 `get_h_id()`

```
def rel_network.RelNeuron.get_h_id (
    self )
```

Return hearing id if neuron has knowledge and an object of type None in any other case.

Return values

$h \leftrightarrow$ _id	Integer or None. Hearing id.
----------------------------	------------------------------

Here is the call graph for this function:



5.33.2.2 `get_knowledge()`

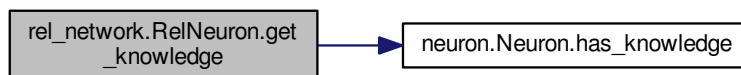
```
def rel_network.RelNeuron.get_knowledge (
    self )
```

Returns knowledge stored by neuron if neuron has knowledge, and None object in any other case.

Return values

<i>knowledge</i>	RelKnowledge or None.
------------------	---------------------------------------

Here is the call graph for this function:



5.33.2.3 `get_s_id()`

```
def rel_network.RelNeuron.get_s_id (
    self )
```

Return sight id if neuron has knowledge and an object of type None in any other case.

Return values

<i>s_id</i>	Integer or None. Sight id.
-------------	----------------------------

Here is the call graph for this function:



5.33.2.4 get_weight()

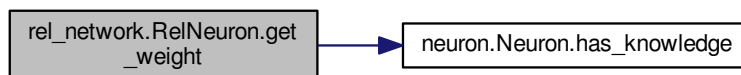
```
def rel_network.RelNeuron.get_weight (
    self )
```

Return weight of relation if neuron has knowledge and an object of type None in any other case.

Return values

<i>weight</i>	Integer or None.
---------------	------------------

Here is the call graph for this function:



5.33.2.5 has_ids()

```
def rel_network.RelNeuron.has_ids (
    self,
    h_id,
    s_id )
```

Return true if neuron has `h_id` and `s_id` as hearing and sight ids respectively.

Parameters

$h \leftrightarrow$ _id	Hearing id
$s \leftrightarrow$ _id	Sight id

5.33.2.6 `learn()`

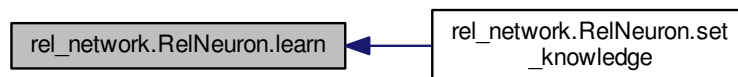
```
def rel_network.RelNeuron.learn (
    self,
    knowledge )
```

Set knowledge of type [RelKnowledge](#).

Parameters

<i>knowledge</i>	RelKnowledge to be learned
------------------	--

Here is the caller graph for this function:

5.33.2.7 `recognize_hearing()`

```
def rel_network.RelNeuron.recognize_hearing (
    self,
    h_id )
```

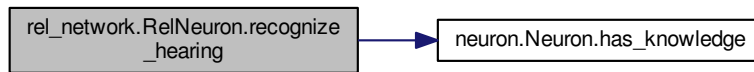
Return True if `h_id` is recognized as the hearing-id part of the [RelKnowledge](#).

Also set an internal flag to indicate whether the last recognition process was successful (True) or not (False). The value of the internal flag is accessible through the `is_hit()` method

Parameters

<i>h_id</i>	Integer. Hearing id.
-------------	----------------------

Here is the call graph for this function:



5.33.2.8 recognize_sight()

```
def rel_network.RelNeuron.recognize_sight (
    self,
    s_id )
```

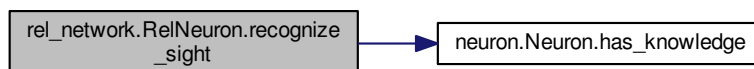
Return true if `s_id` is recognized as the sight-id part of the relational knowledge.

Also set an internal flag to indicate whether the last recognition process was successful (True) or not (False). The value of the internal flag is accessible through the `is_hit()` method

Parameters

<code>s_id</code>	Integer. Sight id.
-------------------	--------------------

Here is the call graph for this function:



5.33.2.9 set_h_id()

```
def rel_network.RelNeuron.set_h_id (
    self,
    h_id )
```

Set hearing id if neuron has knowledge.

Raise an exception of type `AttributeError` if an attempt to set the hearing id to a neuron with no previous knowledge is made

5.33.2.11 set_s_id()

```
def rel_network.RelNeuron.set_s_id (
    self,
    s_id )
```

Set sight id if neuron has knowledge.

Raise an exception of type `AttributeError` if an attempt to set the sight id to a neuron with no previous knowledge is made.

Parameters

$s \leftrightarrow$ _id	Sight id
----------------------------	----------

Here is the call graph for this function:



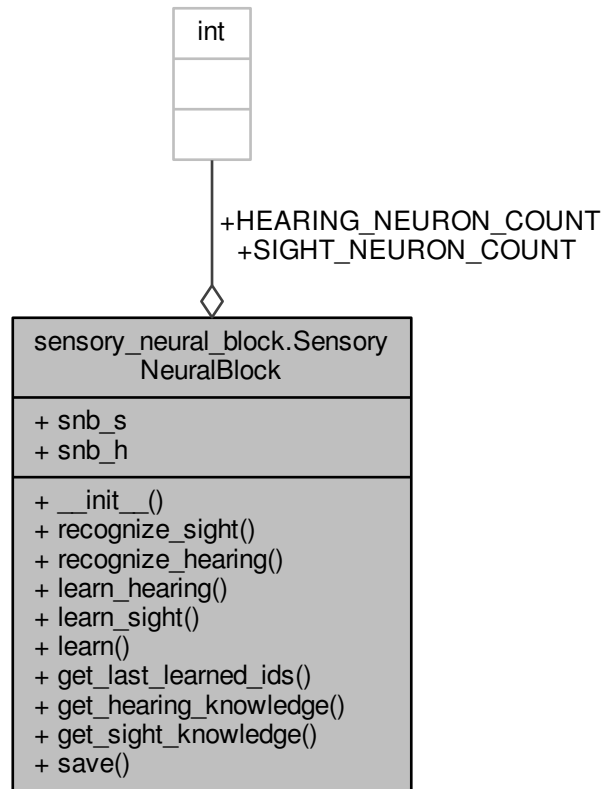
The documentation for this class was generated from the following file:

- `rel_network.py`

5.34 sensory_neural_block.SensoryNeuralBlock Class Reference

Sensory Neural Block Stores sight and hearing RbfNetworks.

Collaboration diagram for `sensory_neural_block.SensoryNeuralBlock`:



Public Member Functions

- `def __init__(self, sight_snb_file="NoFile", hearing_snb_file="NoFile")`
The constructor.
- `def recognize_sight(self, pattern)`
Recognize a sight pattern.
- `def recognize_hearing(self, pattern)`
Recognize a hearing pattern.
- `def learn_hearing(self, knowledge)`
Learn a hearing pattern.
- `def learn_sight(self, knowledge)`
Learn a visual pattern.
- `def learn(self, knowledge_h, pattern_s)`
Learn a pair of hearing and sight patterns relating both pieces of knowledge through the hearing id stored as the sight knowledge's pattern.
- `def get_last_learned_ids(self)`

Return a 2-tuple of integeres representing the ids of hearing and sight neurons that learned in the last learn_sight process.

- def [get_hearing_knowledge](#) (self, pattern_or_id, is_id=False)

Return hearing knowledge related to given pattern or neuron id, if pattern or neuron_id in hearing network, and None in any other case.

- def [get_sight_knowledge](#) (self, pattern_or_id, is_id=False)

Return hearing knowledge related to given pattern or neuron id, if pattern or neuron_id in sight network, and None in any other case.

- def [save](#) (self, sight_snb_file, hearing_snb_file)

Save snb object in given files (one for the sight sensory neural block and the other for the hearing neural block.

Public Attributes

- [snb_s](#)

Sight sensory neural block.

- [snb_h](#)

Hearing sensory neural block.

Static Public Attributes

- int [SIGHT_NEURON_COUNT](#) = 100

Number of neurons in sight network.

- int [HEARING_NEURON_COUNT](#) = 100

Number of neurons in hearing network.

5.34.1 Detailed Description

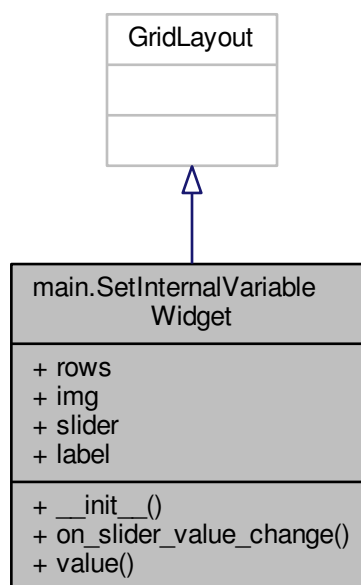
Sensory Neural Block Stores sight and hearing RbfNetworks.

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

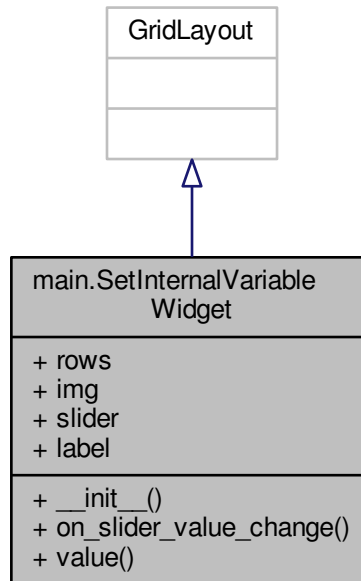
- [sensory_neural_block.py](#)

5.35 main.SetInternalVariableWidget Class Reference

Inheritance diagram for main.SetInternalVariableWidget:



Collaboration diagram for main.SetInternalVariableWidget:



Public Member Functions

- `def __init__ (self, img_file, kwargs)`
- `def on_slider_value_change (self, obj, val)`
- `def value (self)`

Public Attributes

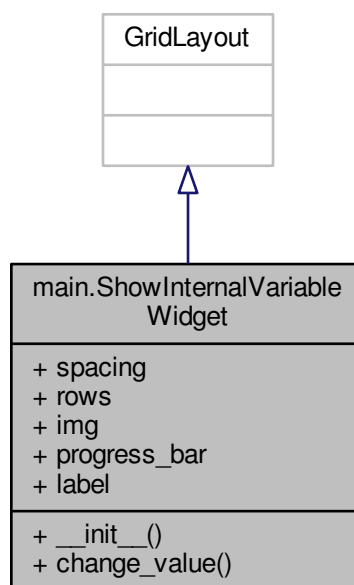
- `rows`
- `img`
- `slider`
- `label`

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

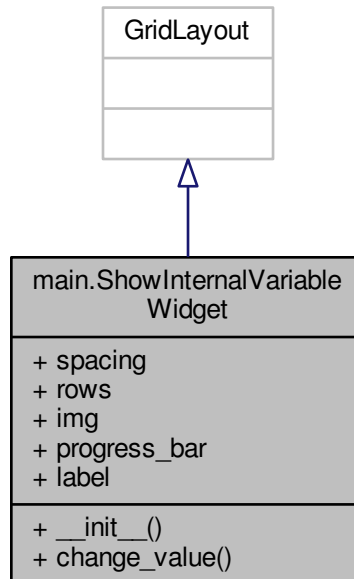
- `main.py`

5.36 main.ShowInternalVariableWidget Class Reference

Inheritance diagram for main.ShowInternalVariableWidget:



Collaboration diagram for main.ShowInternalVariableWidget:



Public Member Functions

- `def __init__ (self, img_file=None, kwargs)`
- `def change_value (self, val)`

Public Attributes

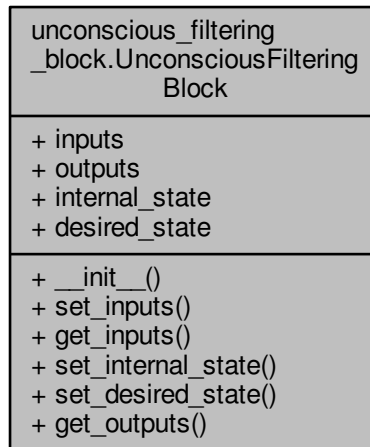
- **spacing**
- **rows**
- **img**
- **progress_bar**
- **label**

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

- `main.py`

5.37 `unconscious_filtering_block.UnconsciousFilteringBlock` Class Reference

Collaboration diagram for `unconscious_filtering_block.UnconsciousFilteringBlock`:



Public Member Functions

- `def __init__(self)`
The constructor.
- `def set_inputs(self, inputs)`
Set block inputs.
- `def get_inputs(self)`
Get block inputs.
- `def set_internal_state(self, internal_state)`
Set internal state.
- `def set_desired_state(self, desired_state)`
Set desired state.
- `def get_outputs(self)`
Return uncounsciously filtered memories.

Public Attributes

- `inputs`
- `outputs`
- `internal_state`
- `desired_state`

5.37.1 Member Function Documentation

5.37.1.1 get_inputs()

```
def unconscious_filtering_block.UnconsciousFilteringBlock.get_inputs (
    self )
```

Get block inputs.

Return values

<i>inputs</i>	CulturalGroup with tail knowledge of class BiologyCultureFeelings
---------------	---

5.37.1.2 get_outputs()

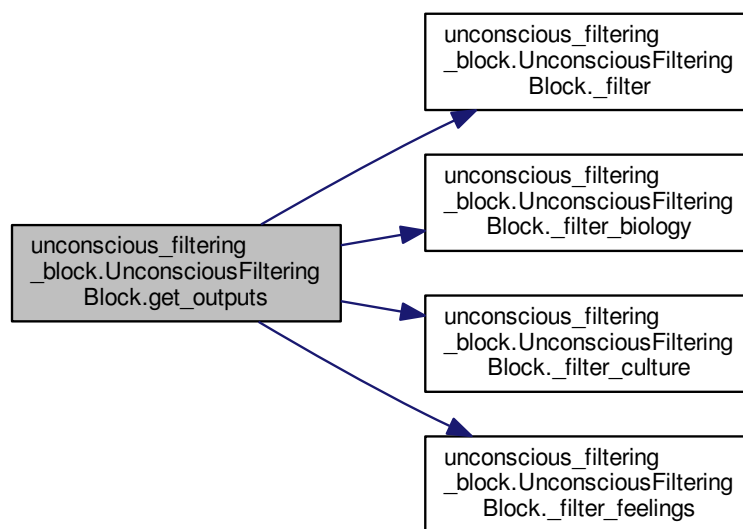
```
def unconscious_filtering_block.UnconsciousFilteringBlock.get_outputs (
    self )
```

Return uncounciously filtered memories.

Return values

<i>output</i>	Vector of three memories (cultural groups) [Biology, Culture, Feelings]
---------------	---

Here is the call graph for this function:



5.37.1.3 set_desired_state()

```
def unconscious_filtering_block.UnconsciousFilteringBlock.set_desired_state (
    self,
    desired_state )
```

Set desired state.

Parameters

<i>desired_state</i>	InternalState. Entity's desired state
----------------------	---------------------------------------

5.37.1.4 set_inputs()

```
def unconscious_filtering_block.UnconsciousFilteringBlock.set_inputs (
    self,
    inputs )
```

Set block inputs.

Parameters

<i>inputs</i>	CulturalGroup with tail knowledge of class BiologyCultureFeelings
---------------	---

5.37.1.5 set_internal_state()

```
def unconscious_filtering_block.UnconsciousFilteringBlock.set_internal_state (
    self,
    internal_state )
```

Set internal state.

Parameters

<i>internal_state</i>	InternalState. Entity's internal state
-----------------------	--

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

- unconscious_filtering_block.py

Index

- `__init__`
 - `geometric_neural_block::AdditionStructure`, 48
 - `multiclass_single_layer_network::MulticlassSingleLayerNetwork`, 114
 - `rel_network::RelNetwork`, 148
- Analytical neuron related classes, 7
 - `solve_ambiguity`, 7
- `analytical_neuron.AnalyticalNeuron`, 49
- `average_biology`
 - `internal_state::InternalState`, 106
- `average_culture`
 - `internal_state::InternalState`, 107
- `average_feelings`
 - `internal_state::InternalState`, 107
- `average_state`
 - `internal_state::InternalState`, 108
- BCF classes, 12
- `biology_alarm`
 - `internal_state::InternalState`, 108
- `biology_up_alarm`
 - `internal_state::InternalState`, 109
- bip
 - `cultural_network::CulturalGroup`, 69
 - `cultural_network::CulturalNetwork`, 75
 - `geometric_neural_block::GeometricNeuralBlock`, 95
- Brain-CEMISID kernel, 13
 - `check`, 15
 - `erase_all_knowledge`, 15
 - `feed_internal_state`, 16
 - `get_desired_state`, 17
 - `get_hearing_knowledge_in`, 17
 - `get_hearing_knowledge_out`, 17
 - `get_internal_state`, 18
 - `get_sight_knowledge_in`, 18
 - `get_sight_knowledge_out`, 18
 - `get_working_domain`, 19
 - `recognize`, 19
 - `set_desired_state`, 19
 - `set_hearing_knowledge_in`, 20
 - `set_internal_state`, 20
 - `set_internal_state_in`, 22
 - `set_sight_knowledge_in`, 22
 - `set_working_domain`, 23
- `carry_over`
 - `geometric_neural_block::AdditionStructure`, 48
- `check`
 - Brain-CEMISID kernel, 15
 - `cultural_network::CulturalGroup`, 69
 - `cultural_network::CulturalNetwork`, 75
- `clack`
 - `cultural_network::CulturalGroup`, 70
 - `cultural_network::CulturalNetwork`, 76
- `compare`
 - `geometric_neural_block::QuantityOrderGroup`, 133
- `conscious_decisions_block.ConsciousDecisionsBlock`, 61
- `conscious_decisions_block::ConsciousDecisionsBlock`
 - `feedback`, 62
 - `get_decision`, 63
 - `get_desired_state`, 64
 - `get_inputs`, 64
 - `get_internal_state`, 65
 - `get_last_decision_type`, 65
 - `set_desired_state`, 65
 - `set_inputs`, 66
 - `set_internal_state`, 66
 - `training`, 67
- `contains`
 - `cultural_network::CulturalGroup`, 71
- Cultural network related classes, 10
- `cultural_network.CulturalGroup`, 68
- `cultural_network.CulturalNetwork`, 72
- `cultural_network.CulturalNeuron`, 77
- `cultural_network::CulturalGroup`
 - bip, 69
 - `check`, 69
 - `clack`, 70
 - `contains`, 71
 - `get_tail_knowledge`, 71
 - `learn`, 71
- `cultural_network::CulturalNetwork`
 - bip, 75
 - `check`, 75
 - `clack`, 76
 - `deserialize`, 76
 - `get_tail_knowledge`, 76
 - `serialize`, 77
- `decision_by_prediction_block.DecisionByPredictionBlock`, 79

decision_by_prediction_block::DecisionByPredictionBlock
 get_desired_state, 81
 get_distances, 82
 get_inputs, 82
 get_output, 82
 get_predicted_outcomes, 83
 remodel_predictive_net, 83
 set_desired_state, 84
 set_inputs, 84
 set_internal_state, 85
 decisions_block.DecisionsBlock, 85
 decisions_block::DecisionsBlock
 get_output_memory, 87
 set_desired_state, 87
 set_input_memories, 88
 set_internal_state, 88
 deserialize
 cultural_network::CulturalNetwork, 76
 episodic_memories::EpisodicMemoriesBlock, 92
 geometric_neural_block::GeometricNeuralBlock, 96
 RBF network related classes, 28
 rel_network::RelNetwork, 148
 draw_pattern
 main::MyPaintWidget, 125

 episodic_memories.EpisodicMemoriesBlock, 89
 episodic_memories::EpisodicMemoriesBlock
 deserialize, 92
 retrieve_exact_memory, 92
 retrieve_memories, 93
 serialize, 93
 erase_all_knowledge
 Brain-CEMISID kernel, 15

 feed_internal_state
 Brain-CEMISID kernel, 16
 feedback
 conscious_decisions_block::ConsciousDecisions↔
 Block, 62

 Geometric Neural Block classes, 11
 geometric_neural_block.AdditionStructure, 47
 geometric_neural_block.GeometricNeuralBlock, 94
 geometric_neural_block.OrderNeuron, 128
 geometric_neural_block.QuantityNeuron, 130
 geometric_neural_block.QuantityOrderGroup, 132
 geometric_neural_block.QuantityOrderNetwork, 134
 geometric_neural_block::AdditionStructure
 __init__, 48
 carry_over, 48
 index, 48
 geometric_neural_block::GeometricNeuralBlock
 bip, 95
 deserialize, 96
 get_add_operator, 96
 get_addition_result, 97
 get_equal_sign, 97
 get_operation, 98
 serialize, 98
 set_add_operator, 98
 set_equal_sign, 98
 set_operation, 98
 set_zero, 99
 geometric_neural_block::QuantityOrderGroup
 compare, 133
 geometric_neural_block::QuantityOrderNetwork
 get_bip_count, 135
 get_add_operator
 geometric_neural_block::GeometricNeuralBlock, 96
 get_addition_result
 geometric_neural_block::GeometricNeuralBlock, 97
 get_biology
 internal_state::BiologyCultureFeelings, 52
 get_bip_count
 geometric_neural_block::QuantityOrderNetwork, 135
 get_class
 RBF network related classes, 28
 get_culture
 internal_state::BiologyCultureFeelings, 53
 get_decision
 conscious_decisions_block::ConsciousDecisions↔
 Block, 63
 get_desired_state
 Brain-CEMISID kernel, 17
 conscious_decisions_block::ConsciousDecisions↔
 Block, 64
 decision_by_prediction_block::DecisionByPrediction↔
 Block, 81
 get_distance
 RBF network related classes, 29
 get_distances
 decision_by_prediction_block::DecisionByPrediction↔
 Block, 82
 get_equal_sign
 geometric_neural_block::GeometricNeuralBlock, 97
 get_feelings
 internal_state::BiologyCultureFeelings, 53
 get_h_id
 rel_network::RelKnowledge, 144
 rel_network::RelNeuron, 153
 get_hearing_knowledge_in
 Brain-CEMISID kernel, 17
 get_hearing_knowledge_out
 Brain-CEMISID kernel, 17
 get_hearing_rels
 rel_network::RelNetwork, 149
 get_index_ready_to_learn
 RBF network related classes, 29
 get_inputs

- conscious_decisions_block::ConsciousDecisions↔
Block, 64
- decision_by_prediction_block::DecisionByPrediction↔
Block, 82
- multiclass_single_layer_network::MulticlassSingle↔
LayerNetwork, 114
- unconscious_filtering_block::UnconsciousFiltering↔
Block, 167
- get_internal_state
Brain-CEMISID kernel, 18
conscious_decisions_block::ConsciousDecisions↔
Block, 65
- get_knowledge
RBF network related classes, 29
rel_network::RelNeuron, 154
- get_last_decision_type
conscious_decisions_block::ConsciousDecisions↔
Block, 65
- get_last_learned_id
RBF network related classes, 30
- get_learning_rate
multiclass_single_layer_network::MulticlassSingle↔
LayerNetwork, 114
- get_neuron_count
RBF network related classes, 30
rel_network::RelNetwork, 149
- get_operation
geometric_neural_block::GeometricNeuralBlock, 98
- get_output
decision_by_prediction_block::DecisionByPrediction↔
Block, 82
- get_output_memory
decisions_block::DecisionsBlock, 87
- get_outputs
multiclass_single_layer_network::MulticlassSingle↔
LayerNetwork, 115
unconscious_filtering_block::UnconsciousFiltering↔
Block, 167
- get_pattern
RBF network related classes, 30
- get_predicted_outcomes
decision_by_prediction_block::DecisionByPrediction↔
Block, 83
- get_radius
RBF network related classes, 31
- get_rneurons_ids
RBF network related classes, 31
- get_s_id
rel_network::RelKnowledge, 144
rel_network::RelNeuron, 154
- get_set
RBF network related classes, 31, 32
- get_sight_knowledge_in
Brain-CEMISID kernel, 18
- get_sight_knowledge_out
Brain-CEMISID kernel, 18
- get_sight_rels
rel_network::RelNetwork, 149
- get_state
internal_state::BiologyCultureFeelings, 54
RBF network related classes, 33
- get_tail_knowledge
cultural_network::CulturalGroup, 71
cultural_network::CulturalNetwork, 76
- get_weight
rel_network::RelKnowledge, 144
rel_network::RelNeuron, 155
- get_working_domain
Brain-CEMISID kernel, 19
- has_ids
rel_network::RelNeuron, 155
- has_knowledge
neuron::Neuron, 126
- increase_radius_by
RBF network related classes, 33
- increase_weight
rel_network::RelKnowledge, 145
- index
geometric_neural_block::AdditionStructure, 48
- Intentions related classes, 9
- internal_state.BiologyCultureFeelings, 50
- internal_state.InternalState, 103
- internal_state::BiologyCultureFeelings
get_biology, 52
get_culture, 53
get_feelings, 53
get_state, 54
set_biology, 54
set_culture, 55
set_feelings, 55
set_state, 56
- internal_state::InternalState
average_biology, 106
average_culture, 107
average_feelings, 107
average_state, 108
biology_alarm, 108
biology_up_alarm, 109
- is_degraded
RBF network related classes, 33
- is_equal
rel_network::RelKnowledge, 145
- is_hit
RBF network related classes, 34
- is_member
RBF network related classes, 34

- kernel_braincemisid.KernelBrainCemisid, 109
- learn
 - cultural_network::CulturalGroup, 71
 - RBF network related classes, 34–36
 - rel_network::RelNetwork, 149
 - rel_network::RelNeuron, 156
- learn_hearing
 - RBF network related classes, 37
- learn_sight
 - RBF network related classes, 37
- main.IntentionsInterface, 100
- main.MyGroupPaintWidget, 118
- main.MyPaintApp, 119
- main.MyPaintElement, 121
- main.MyPaintWidget, 123
- main.RbfCardWidget, 135
- main.SetInternalVariableWidget, 162
- main.ShowInternalVariableWidget, 164
- main::MyPaintWidget
 - draw_pattern, 125
- multiclass_single_layer_network.MulticlassSingleLayer↔
 - Network, 113
- multiclass_single_layer_network::MulticlassSingleLayer↔
 - Network
 - __init__, 114
 - get_inputs, 114
 - get_learning_rate, 114
 - get_outputs, 115
 - set_activation_function, 115
 - set_inputs, 115
 - set_learning_rate, 116
 - training, 116
 - update_weights, 116
- neuron.Neuron, 125
- neuron::Neuron
 - has_knowledge, 126
- object, 128
- prev_main.BrainInterface, 58
- RBF network related classes, 25
 - deserialize, 28
 - get_class, 28
 - get_distance, 29
 - get_index_ready_to_learn, 29
 - get_knowledge, 29
 - get_last_learned_id, 30
 - get_neuron_count, 30
 - get_pattern, 30
 - get_radius, 31
 - get_rneurons_ids, 31
 - get_set, 31, 32
 - get_state, 33
 - increase_radius_by, 33
 - is_degraded, 33
 - is_hit, 34
 - is_member, 34
 - learn, 34–36
 - learn_hearing, 37
 - learn_sight, 37
 - recognize, 38, 39
 - recognize_hearing, 39
 - recognize_sight, 41
 - reduce_radius_by, 41
 - reduce_radius_last_distance, 42
 - save, 42
 - serialize, 43
 - set_class, 43
 - set_pattern, 43
 - set_radius, 44
 - set_set, 44
- recognize
 - Brain-CEMISID kernel, 19
 - RBF network related classes, 38, 39
- recognize_hearing
 - RBF network related classes, 39
 - rel_network::RelNeuron, 156
- recognize_sight
 - RBF network related classes, 41
 - rel_network::RelNeuron, 157
- reduce_radius_by
 - RBF network related classes, 41
- reduce_radius_last_distance
 - RBF network related classes, 42
- rel_network.RelKnowledge, 143
- rel_network.RelNetwork, 147
- rel_network.RelNeuron, 150
- rel_network::RelKnowledge
 - get_h_id, 144
 - get_s_id, 144
 - get_weight, 144
 - increase_weight, 145
 - is_equal, 145
 - set_h_id, 145
 - set_s_id, 146
 - set_weight, 146
- rel_network::RelNetwork
 - __init__, 148
 - deserialize, 148
 - get_hearing_rels, 149
 - get_neuron_count, 149
 - get_sight_rels, 149
 - learn, 149
 - serialize, 150
- rel_network::RelNeuron

- get_h_id, [153](#)
- get_knowledge, [154](#)
- get_s_id, [154](#)
- get_weight, [155](#)
- has_ids, [155](#)
- learn, [156](#)
- recognize_hearing, [156](#)
- recognize_sight, [157](#)
- set_h_id, [157](#)
- set_knowledge, [158](#)
- set_s_id, [158](#)
- Relational network related classes, [24](#)
- remodel_predictive_net
 - decision_by_prediction_block::DecisionByPrediction↔
Block, [83](#)
- retrieve_exact_memory
 - episodic_memories::EpisodicMemoriesBlock, [92](#)
- retrieve_memories
 - episodic_memories::EpisodicMemoriesBlock, [93](#)
- save
 - RBF network related classes, [42](#)
- sensory_neural_block.RbfKnowledge, [137](#)
- sensory_neural_block.RbfNetwork, [138](#)
- sensory_neural_block.RbfNeuron, [140](#)
- sensory_neural_block.SensoryNeuralBlock, [159](#)
- serialize
 - cultural_network::CulturalNetwork, [77](#)
 - episodic_memories::EpisodicMemoriesBlock, [93](#)
 - geometric_neural_block::GeometricNeuralBlock, [98](#)
 - RBF network related classes, [43](#)
 - rel_network::RelNetwork, [150](#)
- set_activation_function
 - multiclass_single_layer_network::MulticlassSingle↔
LayerNetwork, [115](#)
- set_add_operator
 - geometric_neural_block::GeometricNeuralBlock, [98](#)
- set_biology
 - internal_state::BiologyCultureFeelings, [54](#)
- set_class
 - RBF network related classes, [43](#)
- set_culture
 - internal_state::BiologyCultureFeelings, [55](#)
- set_desired_state
 - Brain-CEMISID kernel, [19](#)
 - conscious_decisions_block::ConsciousDecisions↔
Block, [65](#)
 - decision_by_prediction_block::DecisionByPrediction↔
Block, [84](#)
 - decisions_block::DecisionsBlock, [87](#)
 - unconscious_filtering_block::UnconsciousFiltering↔
Block, [168](#)
- set_equal_sign
 - geometric_neural_block::GeometricNeuralBlock, [98](#)
- set_feelings
 - internal_state::BiologyCultureFeelings, [55](#)
- set_h_id
 - rel_network::RelKnowledge, [145](#)
 - rel_network::RelNeuron, [157](#)
- set_hearing_knowledge_in
 - Brain-CEMISID kernel, [20](#)
- set_input_memories
 - decisions_block::DecisionsBlock, [88](#)
- set_inputs
 - conscious_decisions_block::ConsciousDecisions↔
Block, [66](#)
 - decision_by_prediction_block::DecisionByPrediction↔
Block, [84](#)
 - multiclass_single_layer_network::MulticlassSingle↔
LayerNetwork, [115](#)
 - unconscious_filtering_block::UnconsciousFiltering↔
Block, [168](#)
- set_internal_state
 - Brain-CEMISID kernel, [20](#)
 - conscious_decisions_block::ConsciousDecisions↔
Block, [66](#)
 - decision_by_prediction_block::DecisionByPrediction↔
Block, [85](#)
 - decisions_block::DecisionsBlock, [88](#)
 - unconscious_filtering_block::UnconsciousFiltering↔
Block, [168](#)
- set_internal_state_in
 - Brain-CEMISID kernel, [22](#)
- set_knowledge
 - rel_network::RelNeuron, [158](#)
- set_learning_rate
 - multiclass_single_layer_network::MulticlassSingle↔
LayerNetwork, [116](#)
- set_operation
 - geometric_neural_block::GeometricNeuralBlock, [98](#)
- set_pattern
 - RBF network related classes, [43](#)
- set_radius
 - RBF network related classes, [44](#)
- set_s_id
 - rel_network::RelKnowledge, [146](#)
 - rel_network::RelNeuron, [158](#)
- set_set
 - RBF network related classes, [44](#)
- set_sight_knowledge_in
 - Brain-CEMISID kernel, [22](#)
- set_state
 - internal_state::BiologyCultureFeelings, [56](#)
- set_weight
 - rel_network::RelKnowledge, [146](#)
- set_working_domain
 - Brain-CEMISID kernel, [23](#)
- set_zero

- geometric_neural_block::GeometricNeuralBlock, [99](#)
 - solve_ambiguity
 - Analytical neuron related classes, [7](#)
 - training
 - conscious_decisions_block::ConsciousDecisions↔
Block, [67](#)
 - multiclass_single_layer_network::MulticlassSingle↔
LayerNetwork, [116](#)
 - unconscious_filtering_block.UnconsciousFilteringBlock,
[166](#)
 - unconscious_filtering_block::UnconsciousFilteringBlock
 - get_inputs, [167](#)
 - get_outputs, [167](#)
 - set_desired_state, [168](#)
 - set_inputs, [168](#)
 - set_internal_state, [168](#)
 - update_weights
 - multiclass_single_layer_network::MulticlassSingle↔
LayerNetwork, [116](#)