Brain-CEMISID

2.0

Generated by Doxygen 1.8.12

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

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

rel_knowledge_v	Vector of relational knowledge

Return values

h_id	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

  his 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 contructor.
- 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.

4.6.1 Detailed Description

This is the project kernel where all modules interact.

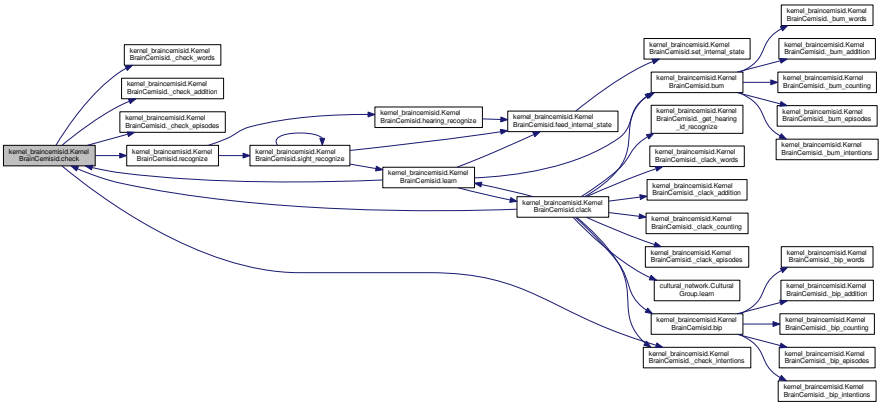
4.6.2 Function Documentation

4.6.2.1 check()

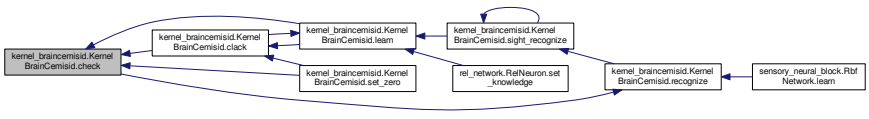
```
def kernel_braincemisid.KernelBrainCemisid.check (
              self )
```

Check if there is knowledge associated with the given sequence of patterns from bbcc protocol when self._enable_bbcc is True.

Check if the Sensory Neural Block recognizes the given pattern when self._enable_bbcc is False Here is the call graph for this function:



Here is the caller graph for this function:

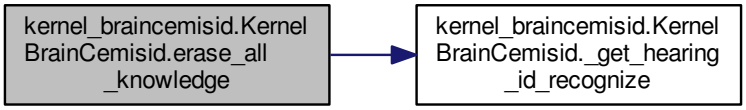


4.6.2.2 erase_all_knowledge()

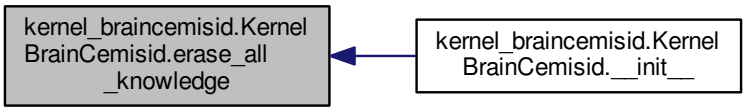
```
def kernel_braincemisid.KernelBrainCemisid.erase_all_knowledge (
              self )
```

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()

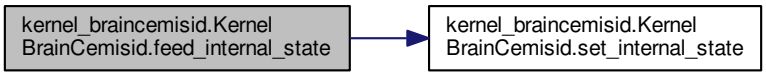
```
def kernel_braincemisid.KernelBrainCemisid.feed_internal_state (
              self,
              states_vector )
```

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

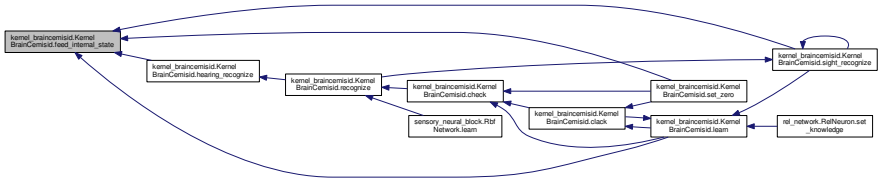
Parameters

states_vector	Floats vector, for example *[0.7, 0.5, 0.3]*

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

desired_state	InternalState

4.6.2.5 get_hearing_knowledge_in()

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

Get hearing knowledge.

Return values

knowledge	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

knowledge	RbfKnowledge vector

4.6.2.7 get_internal_state()

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

Get internal state.

Return values

internal_state	InternalState.

4.6.2.8 get_sight_knowledge_in()

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

Get sight knowledge.

Return values

knowledge	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

knowledge	RbfKnowledge vector

4.6.2.10 get_working_domain()

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

Get bbcc protocol working domain.

Return values

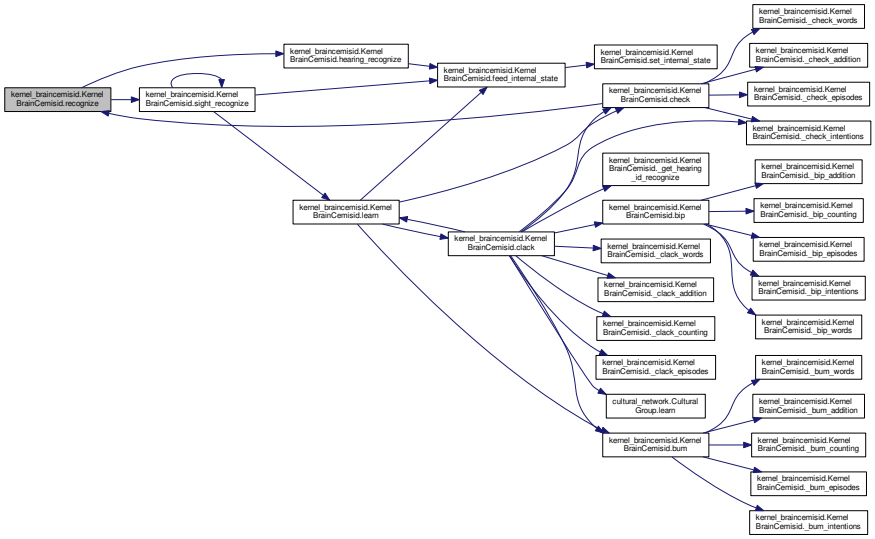
working_domain	enum { "READING", "ADDITION", "COUNTING", "EPISODES", "INTENTIONS" }

4.6.2.11 recognize()

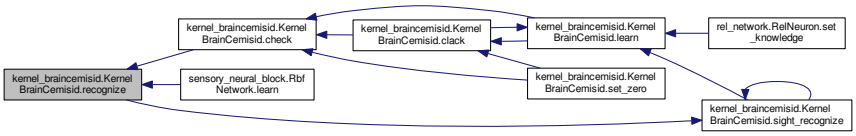
```
def kernel_braincemisid.KernelBrainCemisid.recognize (
              self )
```

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)

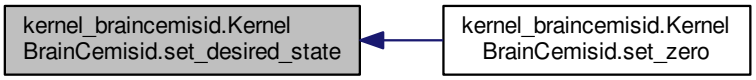
Parameters

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

Return values

result	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

knowledge	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)

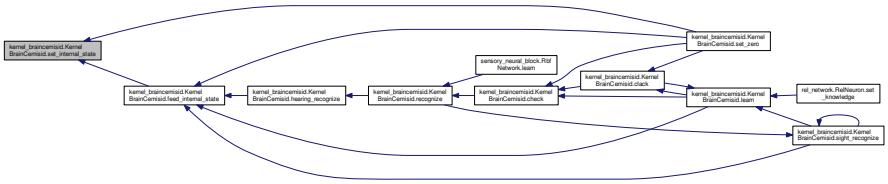
Parameters

states_vector	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 somthing like *[B=1,C=0,F=0]* (Note that this is not the machine's internal state)

Parameters

states_vector	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

knowledge	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

domain	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

cls	RbfNetwork class
name	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

class	Stored pattern class.

4.8.2.3 get_class() [2/2]

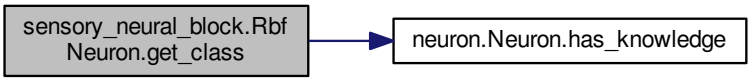
```
def sensory_neural_block.RbfNeuron.get_class (
              self )
```

Get class of stored RbfKnowledge instance.

Return values

class	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

distance	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

index	Integer

4.8.2.6 get_knowledge()

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

Get RbfKnowledge related to last recognized pattern.

Return values

knowledge	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

id	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

count	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

pattern	Stored pattern. Integers vector of size PATTERN_SIZE

4.8.2.10 get_pattern() [2/2]

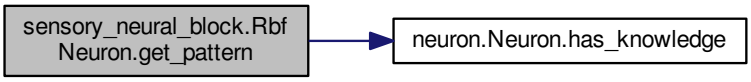
```
def sensory_neural_block.RbfNeuron.get_pattern (
            self )
```

Get pattern of stored RbfKnowledge instance.

Return values

pattern	RbfKnowledgem pattern if neuron has knowledge, None in any other case

Here is the call graph for this function:



4.8.2.11 get_radius()

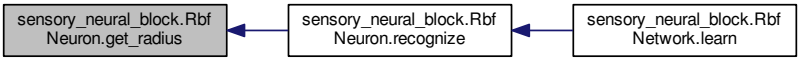
```
def sensory_neural_block.RbfNeuron.get_radius (
              self )
```

Get neuron radius.

Return values

radius	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

ids	Integers list

4.8.2.13 get_set() [1/2]

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

Get stored pattern set.

Return values

set	Stored pattern set

Here is the caller graph for this function:

4.8.2.14 get_set() [2/2]

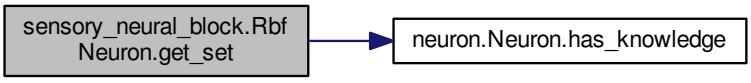
```
def sensory_neural_block.RbfNeuron.get_set (
              self )
```

Get set of stored RbfKnowledge instance.

Return values

set	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

state	'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

value	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

degraded	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

hit	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.

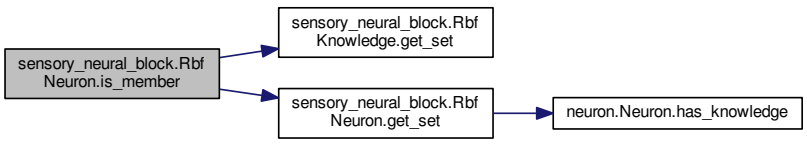
Parameters

test_set	Set to be tested

Return values

is_member	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

learned	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.

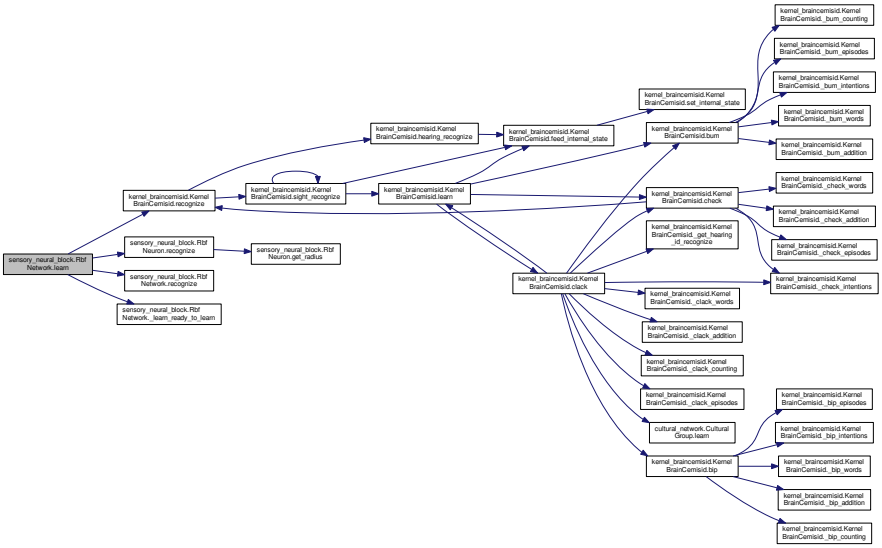
Parameters

knowledge	RbfKnowledge to be learned

Return values

Boolean.	True if successfully learned, False in any other case.

Here is the call graph for this function:



4.8.2.22 learn() [3/3]

```
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.

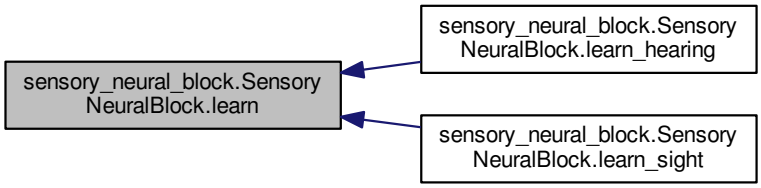
Parameters

knowledge_h	RBF hearing knowledge
pattern_s	RBF sight pattern

Return values

success	True if patterns 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.

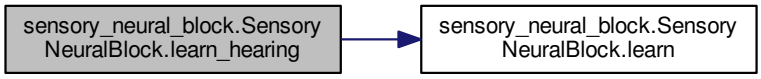
Parameters

pattern	RBF hearing pattern

Return values

success	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.

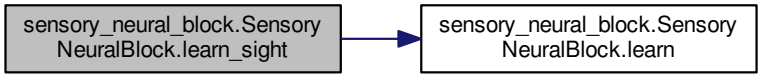
Parameters

pattern	RBF sight pattern

Return values

success	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

recognized	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

pattern	RbfKnowledge pattern to be recognized

Return values

result	'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

pattern	RBF hearing pattern

Return values

success	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

pattern	RBF sight pattern

Return values

success	True if pattern successfully recognized, False in any other case

4.8.2.29 reduce_radius_by()

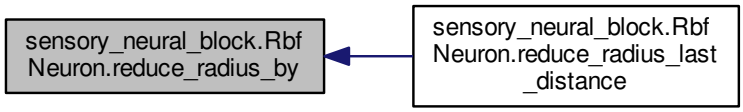
```
def sensory_neural_block.RbfNeuron.reduce_radius_by (
              self,
              value )
```

Reduce neuron radius by certain amount.

Parameters

value	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()

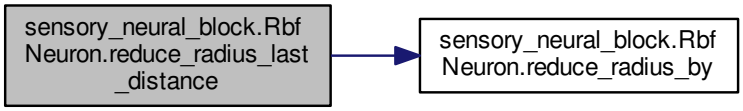
```
def sensory_neural_block.RbfNeuron.reduce_radius_last_distance (
              self )
```

Reduce radius by las recognition process's distance.

Return values

success	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

sight_snb_file	Filename where the sight sensory neural block is to be saved
hearing_snb_file	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

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

4.8.2.33 set_class()

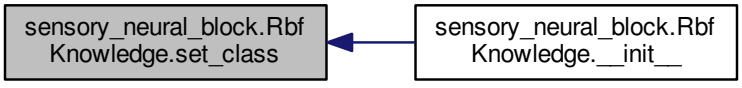
```
def sensory_neural_block.RbfKnowledge.set_class (
              self,
              rbf_class )
```

Set pattern class.

Parameters

rbf_class	Class of the pattern.

Here is the caller graph for this function:



4.8.2.34 set_pattern()

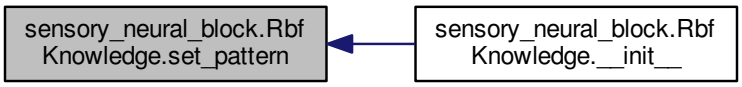
```
def sensory_neural_block.RbfKnowledge.set_pattern (
              self,
              pattern )
```

Set pattern.

Parameters

pattern	Pattern to be stored. Integers vector of size PATTERN_SIZE

Here is the caller graph for this function:



4.8.2.35 set_radius()

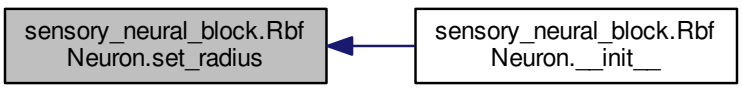
```
def sensory_neural_block.RbfNeuron.set_radius (
              self,
              radius )
```

Sets neuron radius.

Parameters

radius	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

rbf_set	Set of the pattern

Here is the caller graph for this function:

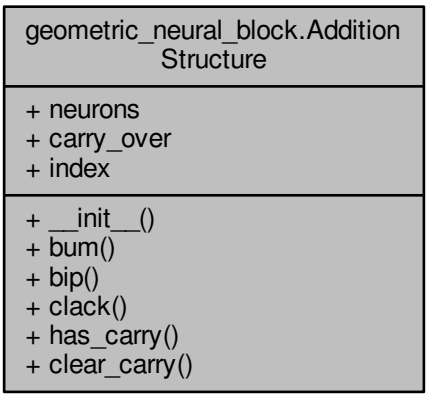
Chapter 5

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:



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.

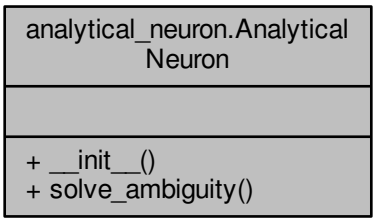
The documentation for this class was generated from the following file:

- 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.

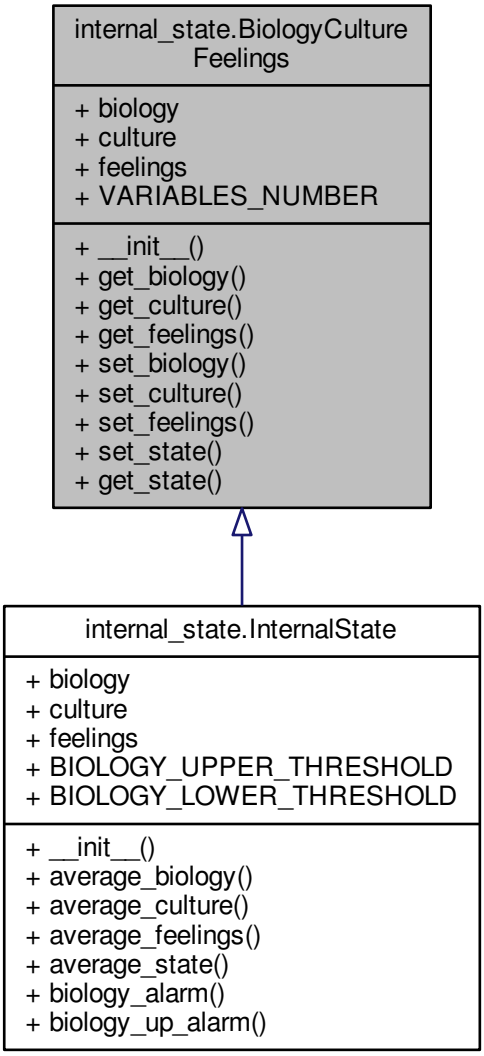
The documentation for this class was generated from the following file:

- analytical_neuron.py

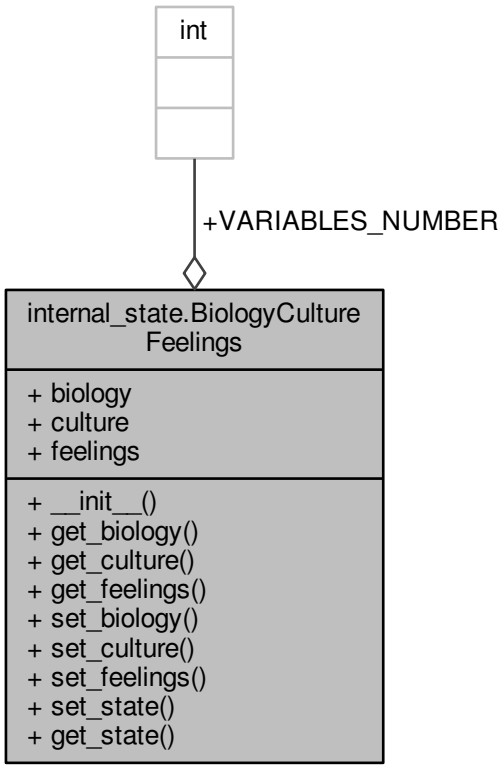
5.3 internal_state.BiologyCultureFeelings Class Reference

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

biology	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

culture	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

feelings	Float from 0 to 1.

Here is the caller graph for this function:

5.3.2.4 get_state()

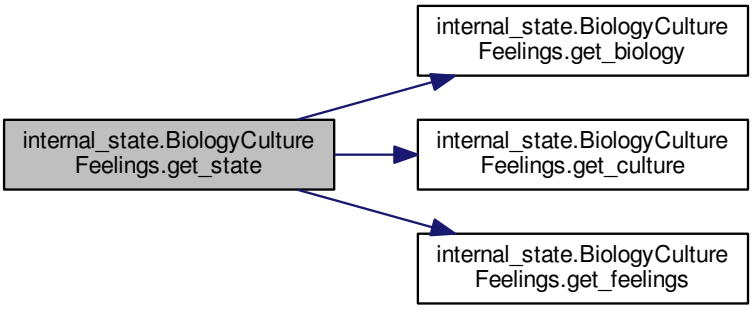
```
def internal_state.BiologyCultureFeelings.get_state (
              self )
```

Get state.

Return values

state	Entity's state

Here is the call graph for this function:



5.3.2.5 set_biology()

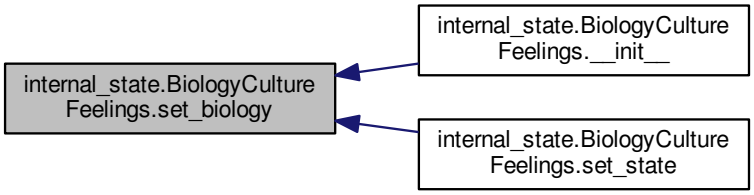
```
def internal_state.BiologyCultureFeelings.set_biology (
              self,
              val )
```

Set biology state.

Parameters

val	Float form 0 to 1. New biology state value.

Here is the caller graph for this function:



5.3.2.6 set_culture()

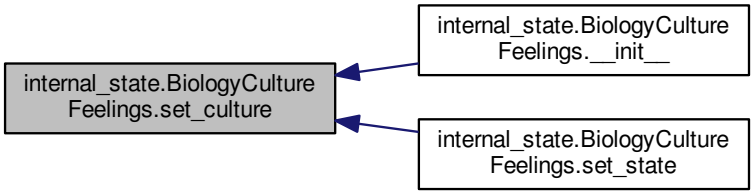
```
def internal_state.BiologyCultureFeelings.set_culture (
              self,
              val )
```

Set culture state.

Parameters

val	Float form 0 to 1. New culture state value.

Here is the caller graph for this function:



5.3.2.7 set_feelings()

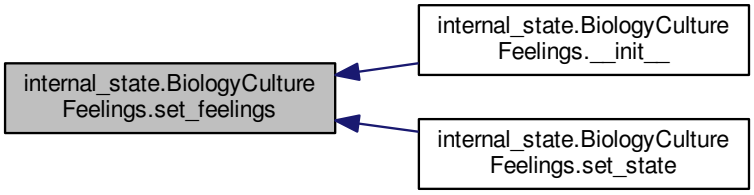
```
def internal_state.BiologyCultureFeelings.set_feelings (
              self,
              val )
```

Set feelings state.

Parameters

val	Float form 0 to 1. New feelings state value.

Here is the caller graph for this function:



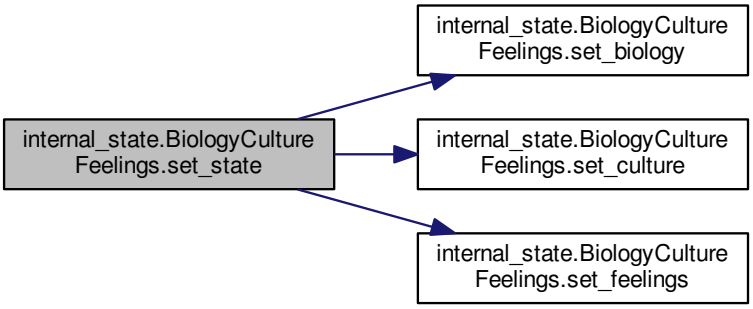
5.3.2.8 set_state()

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

Set state.

Parameters

Here is the call graph for this function:

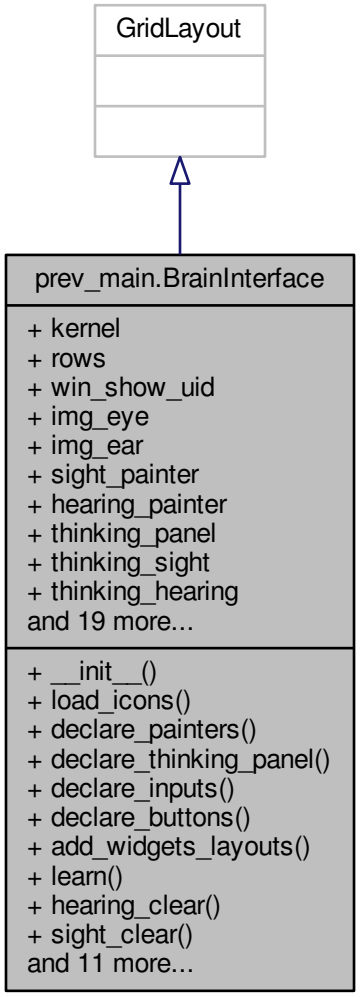


The documentation for this class was generated from the following file:

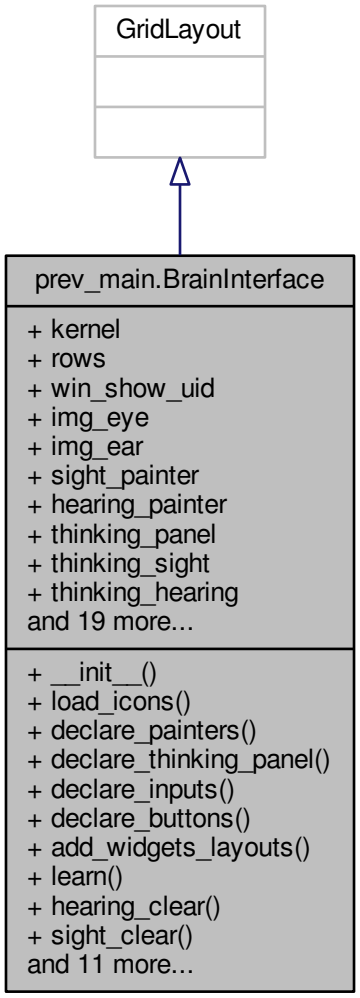
- internal_state.py

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**
- **sight_painter**
- **hearing_painter**
- **thinking_panel**
- **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**
- **hearing_painter_text**
- **hearing_panel**
- **main_panel**
- **senses_panel**
- **buttons_panel**

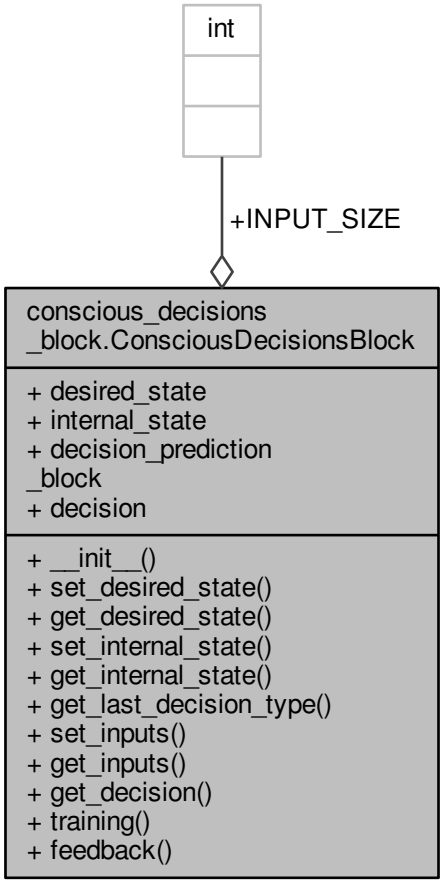
The documentation for this class was generated from the following file:

- 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()

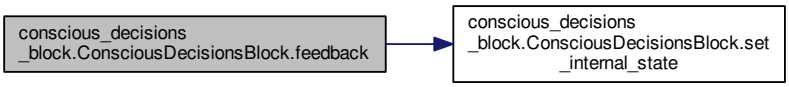
```
def conscious_decisions_block.ConsciousDecisionsBlock.feedback (
            self,
            new_internal_state )
```

Feedback a new internal state to prediction network.

Parameters

new_internal_state	InternalState. New internal state after making a decision and acting on environment

Here is the call graph for this function:

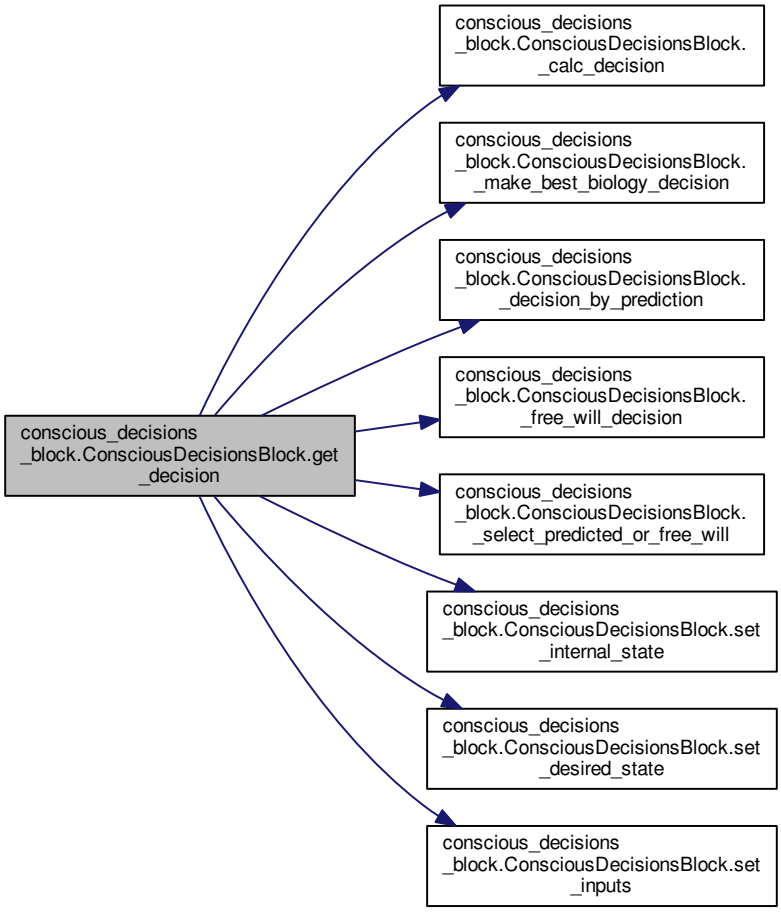


5.5.2.2 get_decision()

```
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

desired_state	InternalState. Stored desired state

5.5.2.4 get_inputs()

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

Get block inputs.

Return values

inputs	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

internal_state	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

last_decision_type	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.

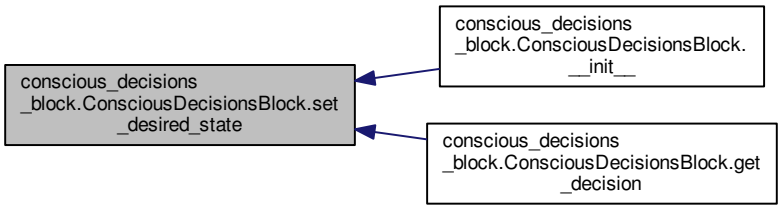
Parameters

desired_state	InternalState. Desired internal state

Return values

result	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()

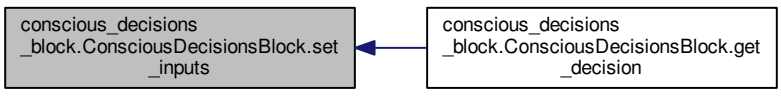
```
def conscious_decisions_block.ConsciousDecisionsBlock.set_inputs (
              self,
              inputs )
```

Set conscious decisions block inputs.

Parameters

inputs	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.

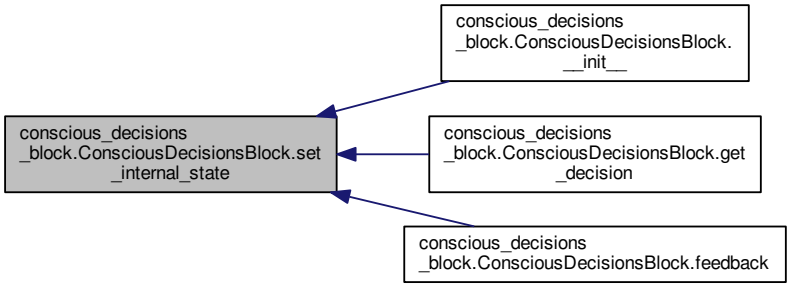
Parameters

internal_state	InternalState. New internal state.

Return values

result	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

training_set	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

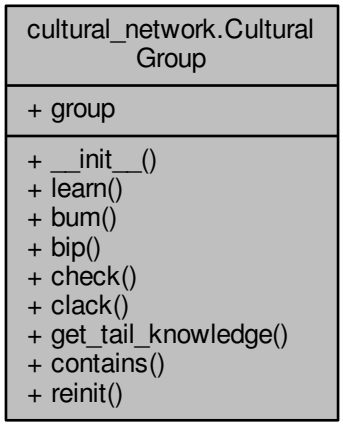
The documentation for this class was generated from the following file:

- 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.

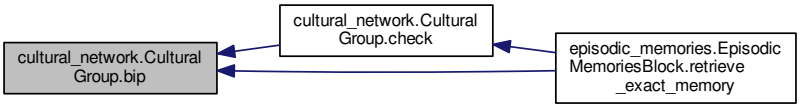
Parameters

knowledge	Piece of knowledge to be compared

Return values

result	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

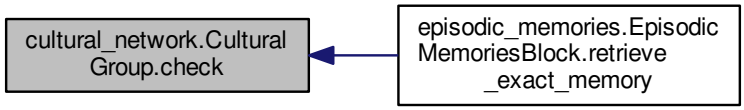
knowledge	Piece of knowledge to be compared

Return values

result	Boolean

Here is the call graph for this function:

Here is the caller graph for this function:



5.6.2.3 clack()

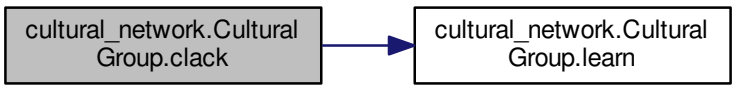
```
def cultural_network.CulturalGroup.clack (
              self,
              knowledge )
```

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

Parameters

knowledge	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

knowledge	

Return values

result	Boolean

5.6.2.5 get_tail_knowledge()

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

Get last knowledge in group.

Return values

knowledge	Last knowledge in group

5.6.2.6 learn()

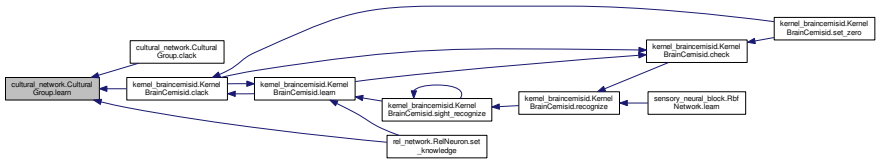
```
def cultural_network.CulturalGroup.learn (
              self,
              knowledge )
```

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

Parameters

knowledge	Object of type CulturalKnowledge. Knowledge to be learned

Here is the caller graph for this function:



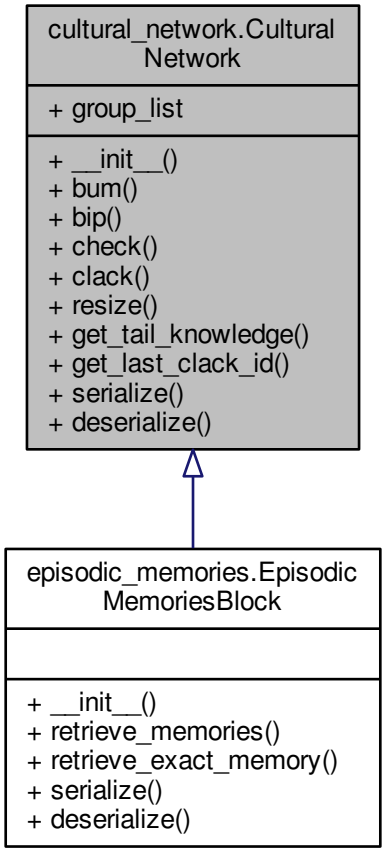
The documentation for this class was generated from the following file:

- cultural_network.py

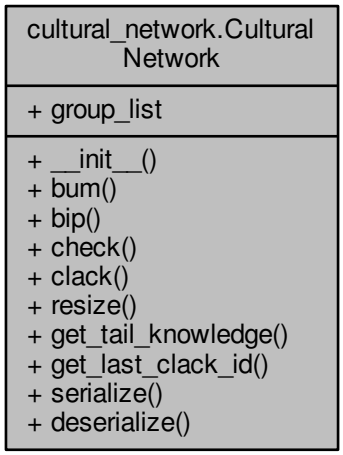
5.7 cultural_network.CulturalNetwork Class Reference

Cultural network.

Inheritance diagram for cultural_network.CulturalNetwork:



Collaboration diagram for cultural_network.CulturalNetwork:



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()

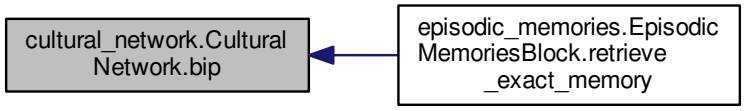
```
def cultural_network.CulturalNetwork.bip (
              self,
              knowledge )
```

Pass an instance of knowledge to be compared or stored.

Parameters

knowledge	

Here is the caller graph for this function:



5.7.2.2 check()

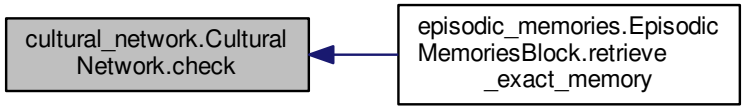
```
def cultural_network.CulturalNetwork.check (
              self,
              knowledge )
```

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

Parameters

knowledge	

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

knowledge	Tail knowledge

5.7.2.4 deserialize()

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

Deserialize object stored in given file.

Parameters

cls	CulturalNetwork class
name	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

group↩ _id	

5.7.2.6 serialize()

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

Serialize object and store in given file.

Parameters

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

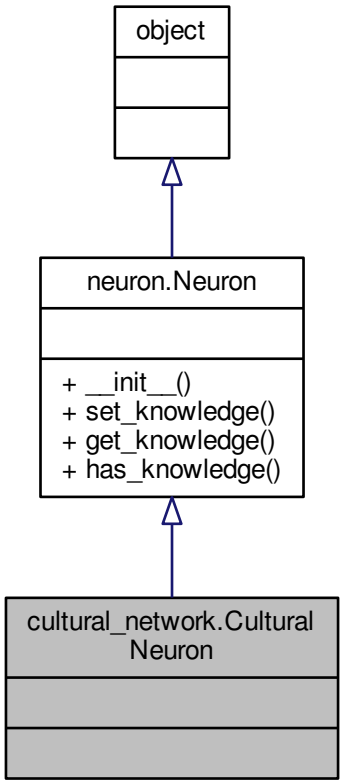
The documentation for this class was generated from the following file:

- cultural_network.py

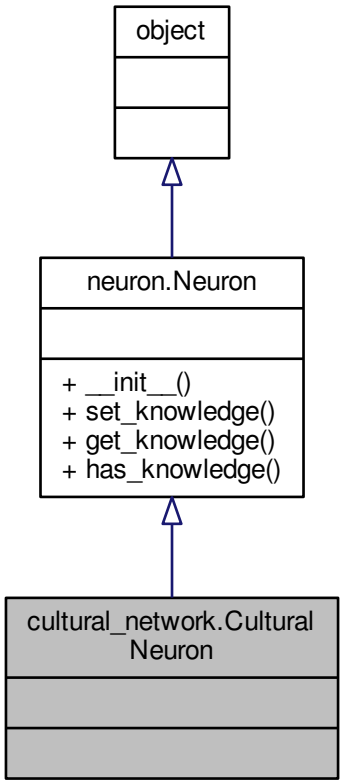
5.8 cultural_network.CulturalNeuron Class Reference

Cultural neuron.

Inheritance diagram for cultural_network.CulturalNeuron:



Collaboration diagram for cultural_network.CulturalNeuron:



Additional Inherited Members

5.8.1 Detailed Description

Cultural neuron.

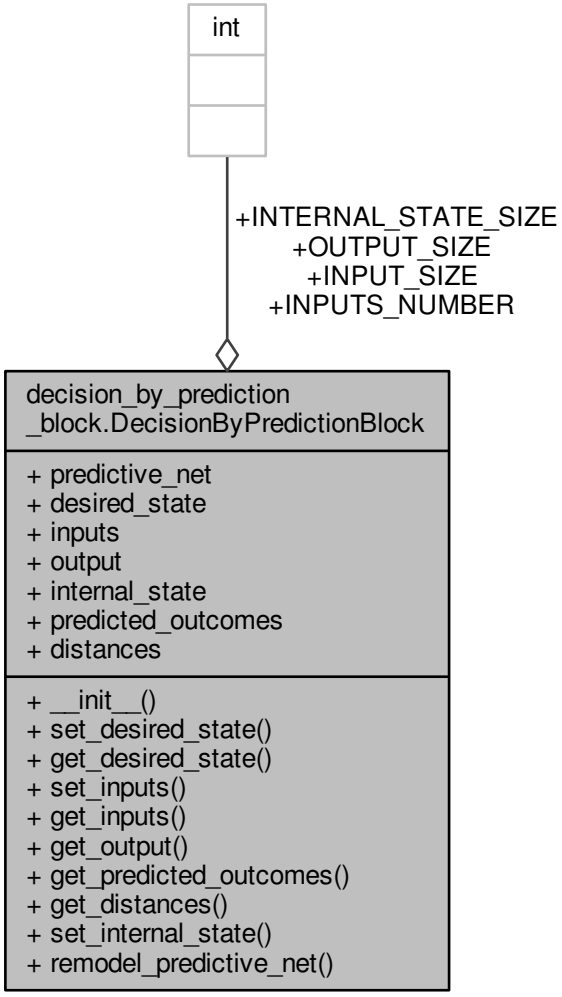
The documentation for this class was generated from the following file:

- 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 de difference of the components) between the desired stated 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

desired_state	InternalState.

5.9.2.2 get_distances()

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

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

Return values

distances	Floats vector.

5.9.2.3 get_inputs()

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

Get network inputs.

Return values

inputs	For example *[[0.5, 0.9, 0.2],[0.5, 0.9, 0.3],[0.4, 0.7, 0.9]]*

5.9.2.4 get_output()

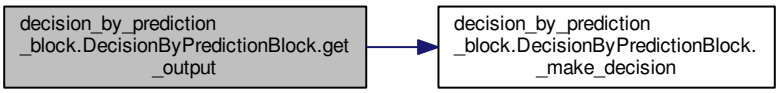
```
def decision_by_prediction_block.DecisionByPredictionBlock.get_output (
              self )
```

Get network output.

Return values

output	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

predicted_outcomes	A vector, for instance, *[[0.43, 0.31, 0.35], [0.44, 0.32, 0.37], [0.52, 0.37, 0.45]]*, 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.

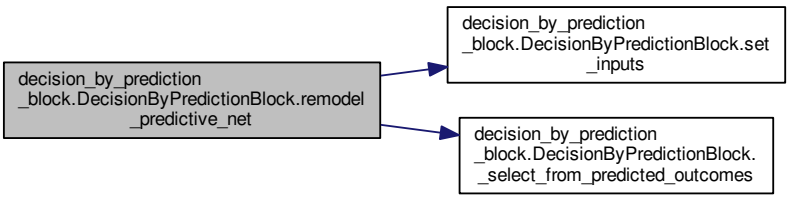
Parameters

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

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

desired_state	InternalState

5.9.2.8 set_inputs()

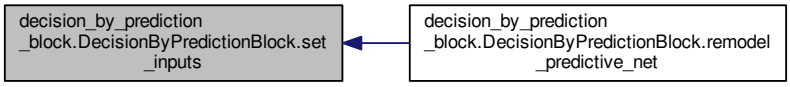
```
def decision_by_prediction_block.DecisionByPredictionBlock.set_inputs (
              self,
              inputs )
```

Set network inputs.

Parameters

inputs	For example, *[[0.5, 0.9, 0.2],[0.5, 0.9, 0.3],[0.4, 0.7, 0.9]]*

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

internal_state	InternalState. New internal state

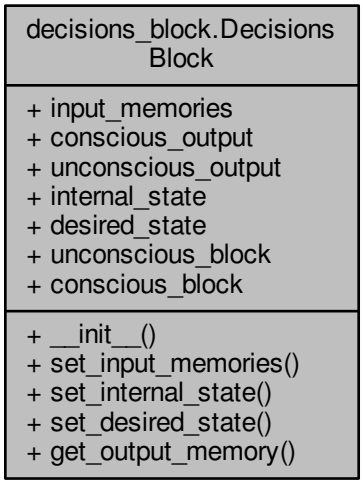
The documentation for this class was generated from the following file:

- 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:



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, is 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()

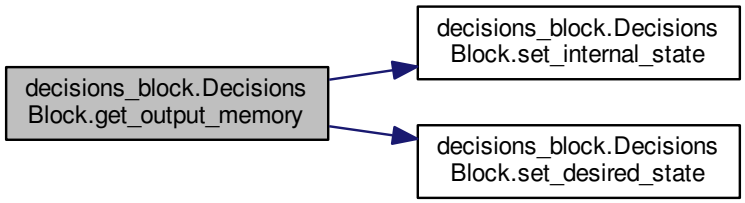
```
def decisions_block.DecisionsBlock.get_output_memory (
              self )
```

Get output memory.

Return values

output	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()

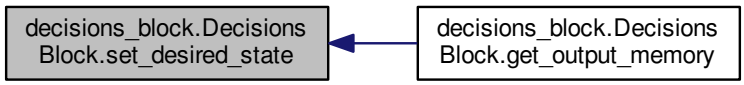
```
def decisions_block.DecisionsBlock.set_desired_state (
              self,
              desired_state )
```

Set entity's desired state.

Parameters

desired_state	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

input_memories	CulturalGroup vector where the last element is of type BiologyCultureFeelings (memories)

5.10.2.4 set_internal_state()

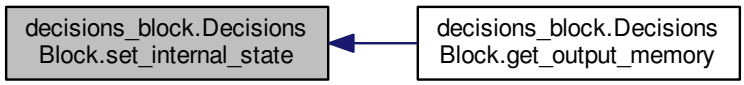
```
def decisions_block.DecisionsBlock.set_internal_state (
              self,
              internal_state )
```

Set entity's internal state.

Parameters

internal_state	InternalState

Here is the caller graph for this function:



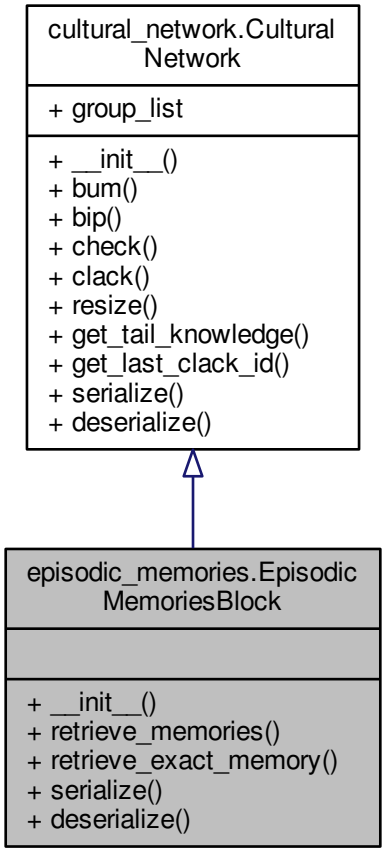
The documentation for this class was generated from the following file:

- decisions_block.py

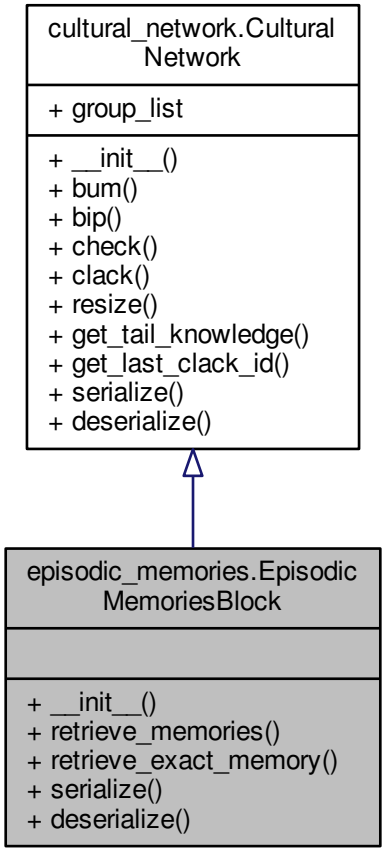
5.11 episodic_memories.EpisodicMemoriesBlock Class Reference

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

cls	EpisodicMemory class
name	Name of the file where the object is serialized

5.11.2.2 retrieve_exact_memory()

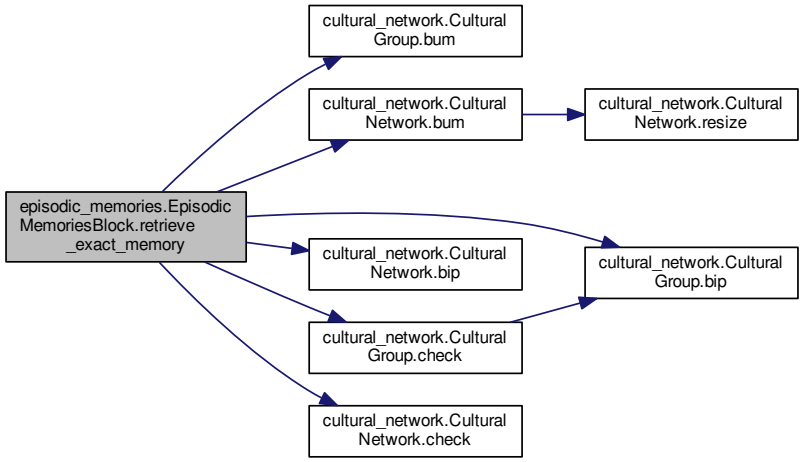
```
def episodic_memories.EpisodicMemoriesBlock.retrieve_exact_memory (
              self,
              trigger )
```

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

Return values

memory	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

retrieved_memories	CulturalGroup vector.

5.11.2.4 serialize()

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

Serialize object and store it in given file.

Parameters

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

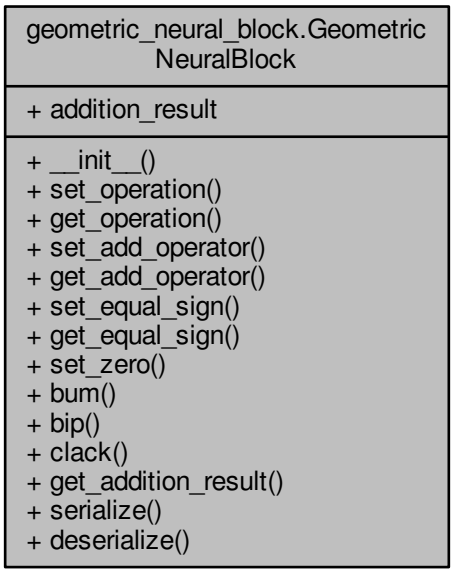
The documentation for this class was generated from the following file:

- 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:



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()

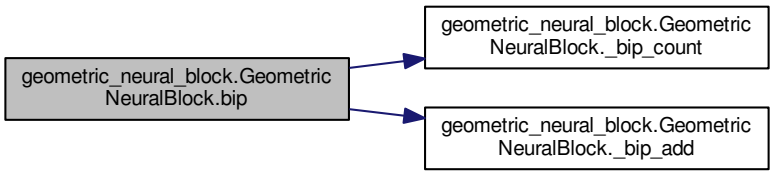
```
def geometric_neural_block.GeometricNeuralBlock.bip (
            self,
            knowledge = None )
```

Either count or pass addition operands and operator.

Parameters

knowledge	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

cls	GeometricNeuralBlock class
name	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

operator	Addition operator

5.12.2.4 get_addition_result()

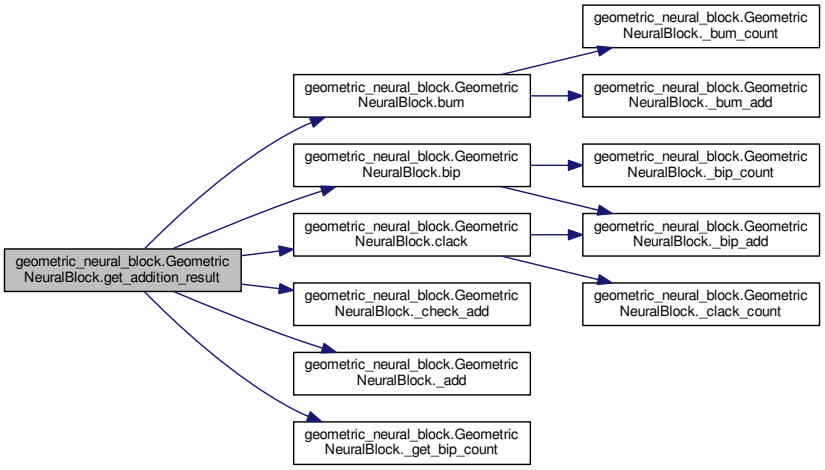
```
def geometric_neural_block.GeometricNeuralBlock.get_addition_result (
              self )
```

Get addition result.

Return values

addition_result	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

knowledge	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

operation	"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

cls	GeometricNeuralBlock class
obj	GeometricNeuralBlock object to be serialized
name	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

operation	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.

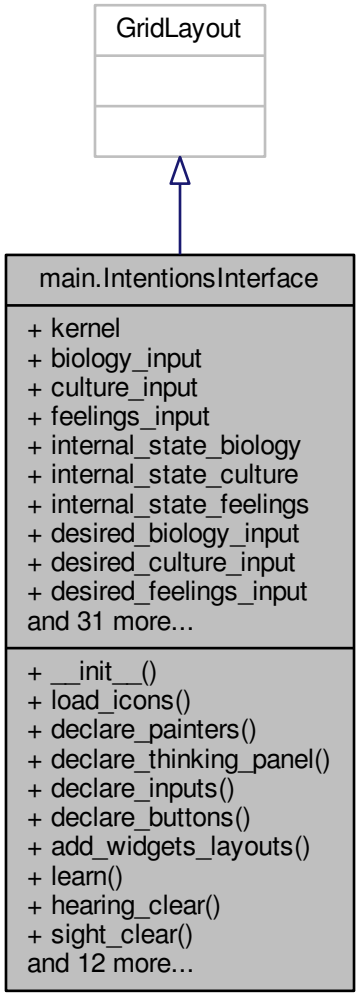
In the Brain-CEMISID project this knowledge is just the hearing neuron id that stores the corresponding pattern

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

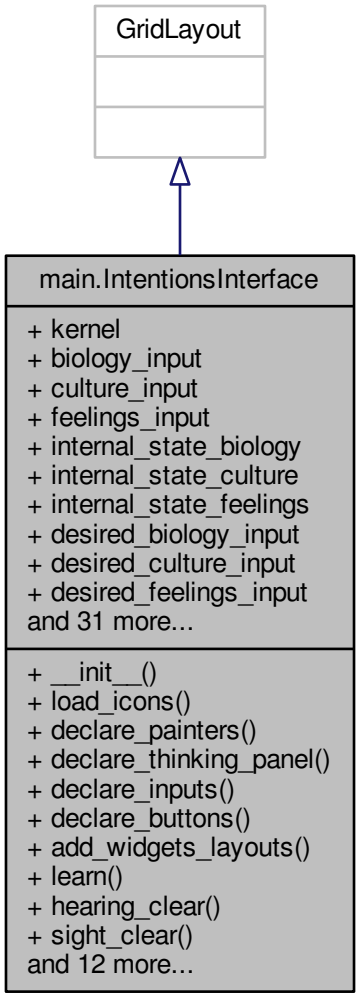
- geometric_neural_block.py

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**
- **sight_painter**
- **hearing_painter**
- **thinking_panel**
- **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**
- **hearing_painter_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**

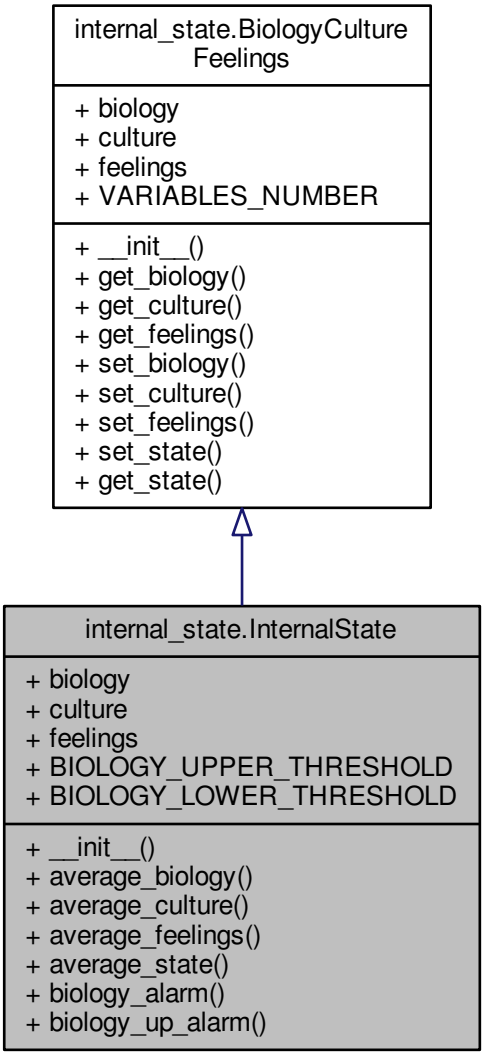
The documentation for this class was generated from the following file:

- main.py

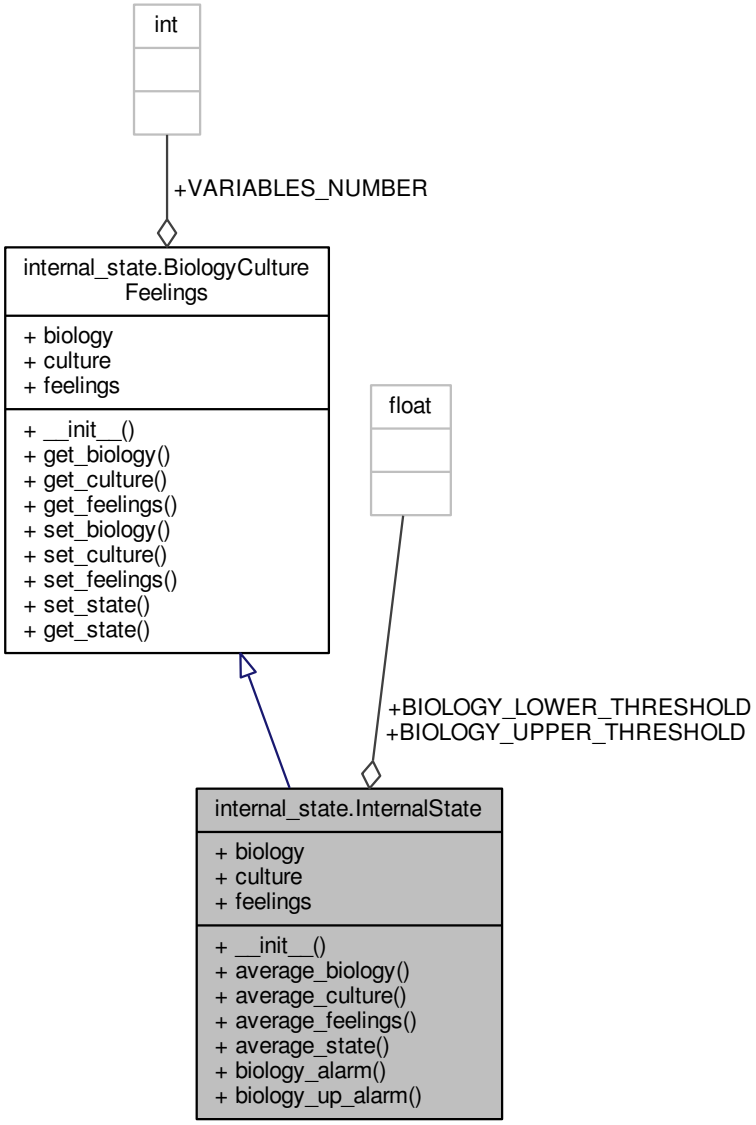
5.14 internal_state.InternalState Class Reference

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 BIOLOG↩ 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

val	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

val	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

val	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()

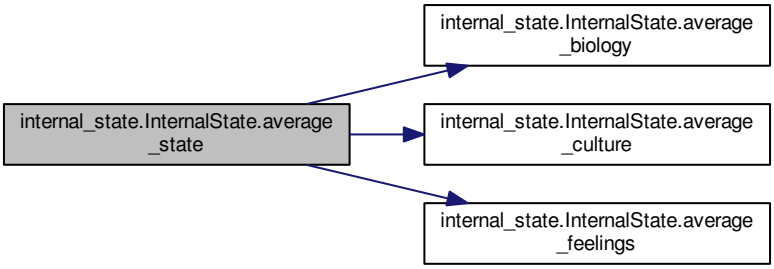
```
def internal_state.InternalState.average_state (
              self,
              states_vector )
```

Average complete state with a given vector of values.

Parameters

states_vector	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 BIOLO↩
GY_UPPER_THRESHOLD (BIOLOGY_LOWER_THRESHOLD)

Return values

alarm	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 BIOLO↩ GY_UPPER_THRESHOLD.

Return values

up_alarm	Boolean. True if there is an upper biology alarm, False in any other case.

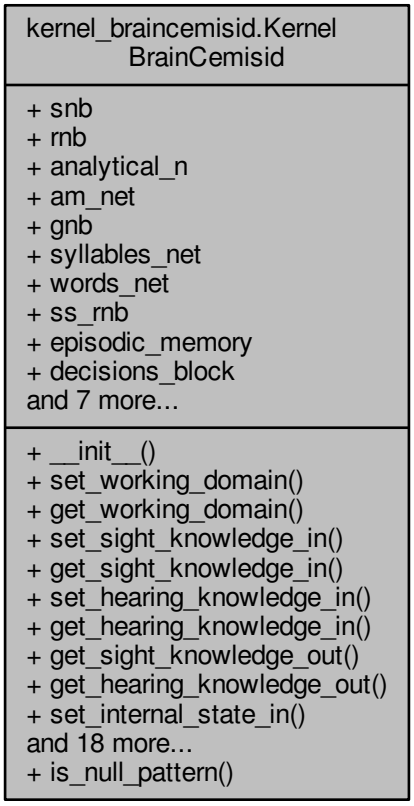
The documentation for this class was generated from the following file:

- 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

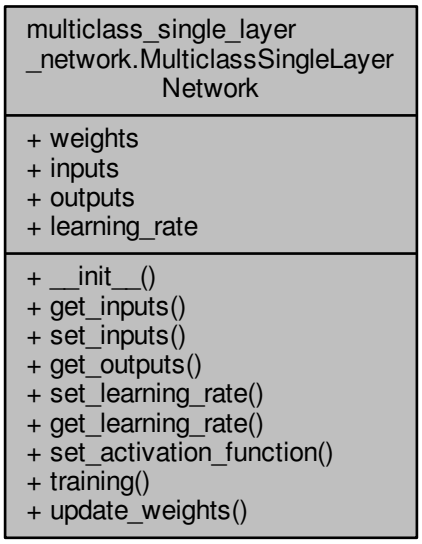
The KernelBrainCemisid is a major module that envelopes and coordinates interaction between all other project modules (Except for the interface, which obviously must not be part of the kernel, but use the kernel)

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:



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

inputs_number	Integer. Number of inputs in perceptron network
outputs_number	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

inputs	Floats vector

5.16.2.2 get_learning_rate()

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

Get learning rate.

Return values

rate	Float

5.16.2.3 get_outputs()

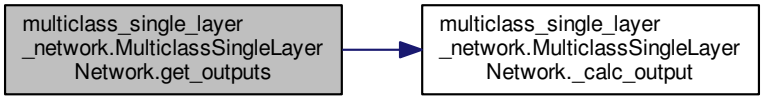
```
def multiclass_single_layer_network.MulticlassSingleLayerNetwork.get_outputs (
              self )
```

Calculate and get outputs.

Return values

outputs	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

new_func	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

inputs	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()

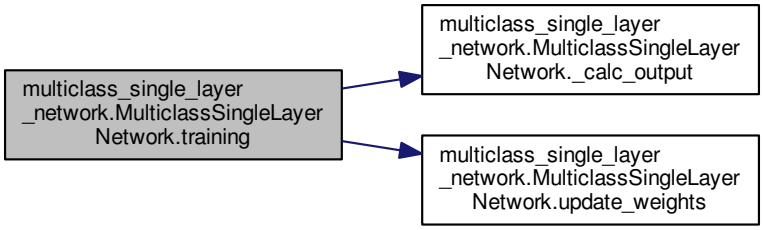
```
def multiclass_single_layer_network.MulticlassSingleLayerNetwork.training (
              self,
              training_set )
```

Train network.

Parameters

training_set	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()

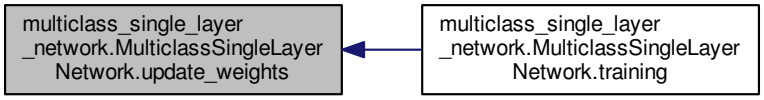
```
def multiclass_single_layer_network.MulticlassSingleLayerNetwork.update_weights (
              self,
              input_vector,
              error )
```

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

Parameters

input_vector	Input vector
error	Prediction error

Here is the caller graph for this function:

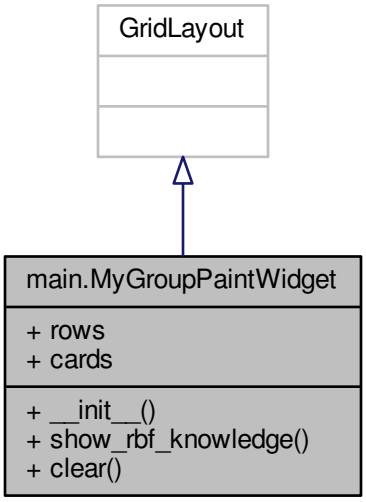


The documentation for this class was generated from the following file:

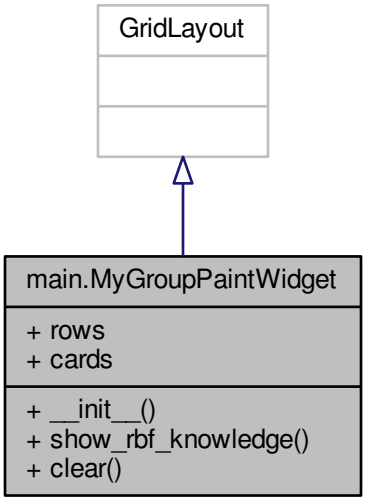
- multiclass_single_layer_network.py

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

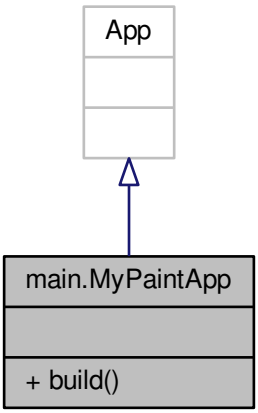
- **rows**
- **cards**

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

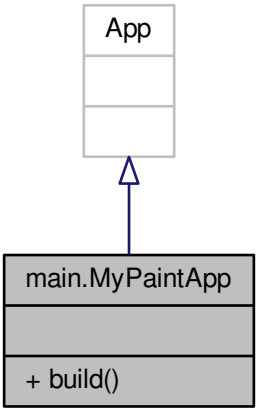
- main.py

5.18 main.MyPaintApp Class Reference

Inheritance diagram for main.MyPaintApp:



Collaboration diagram for main.MyPaintApp:



Public Member Functions

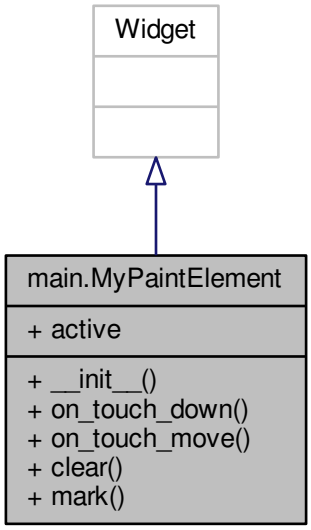
- def **build** (self)

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

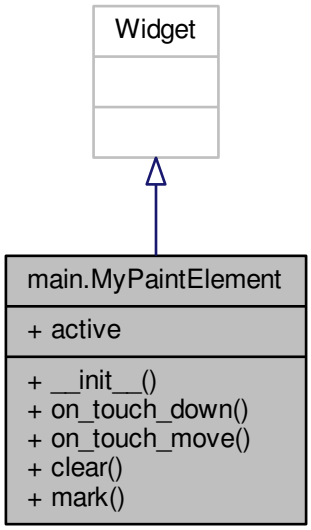
- main.py

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

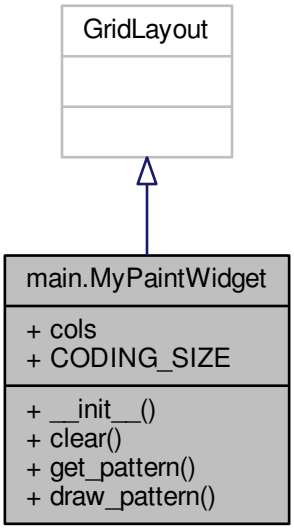
- **active**

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

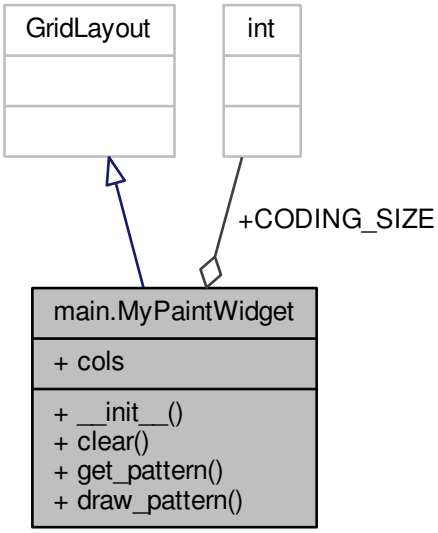
- main.py

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 )
```

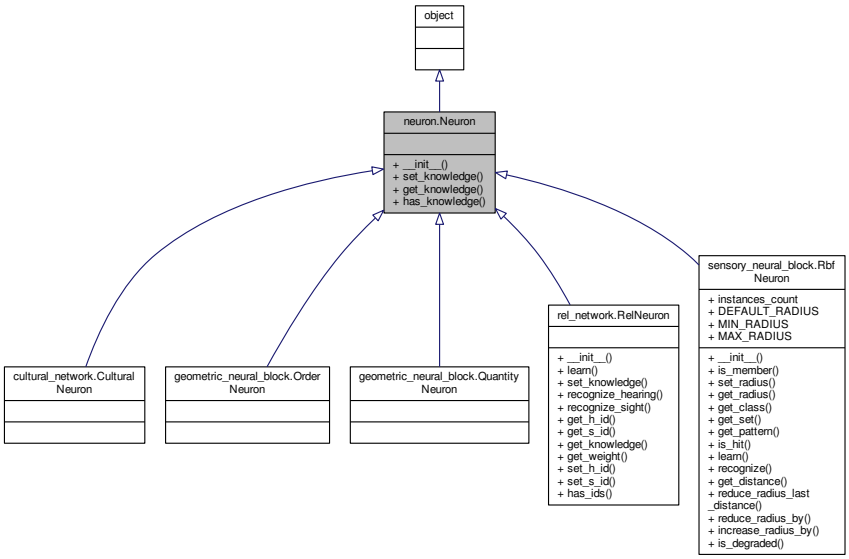
```
Draw given pattern in painter
```

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

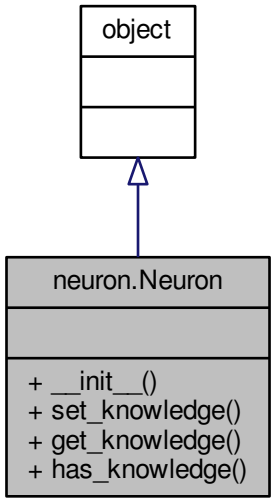
- main.py

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)

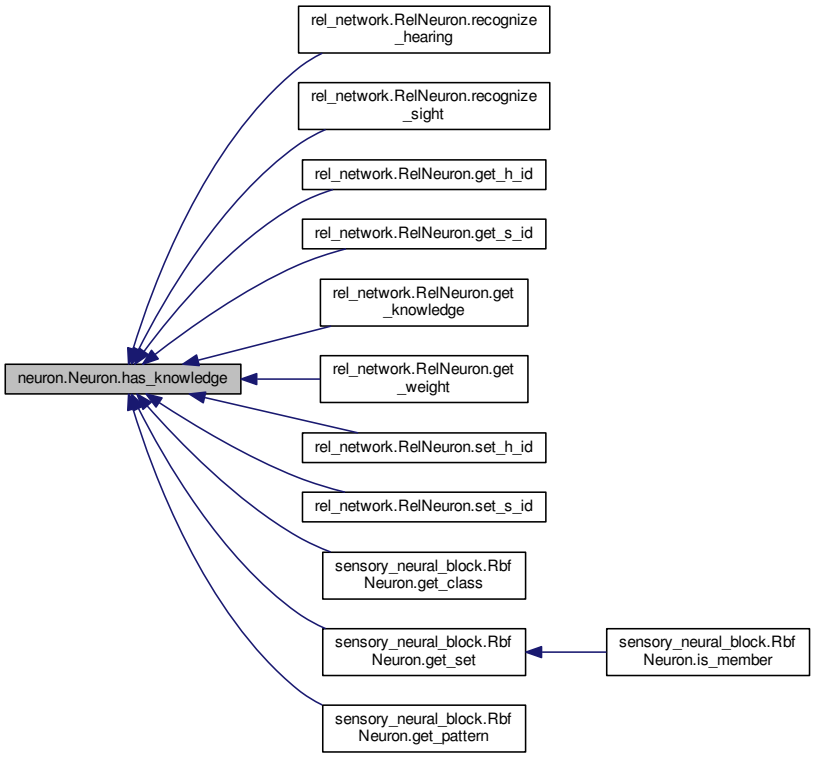
5.21.1 Member Function Documentation

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:

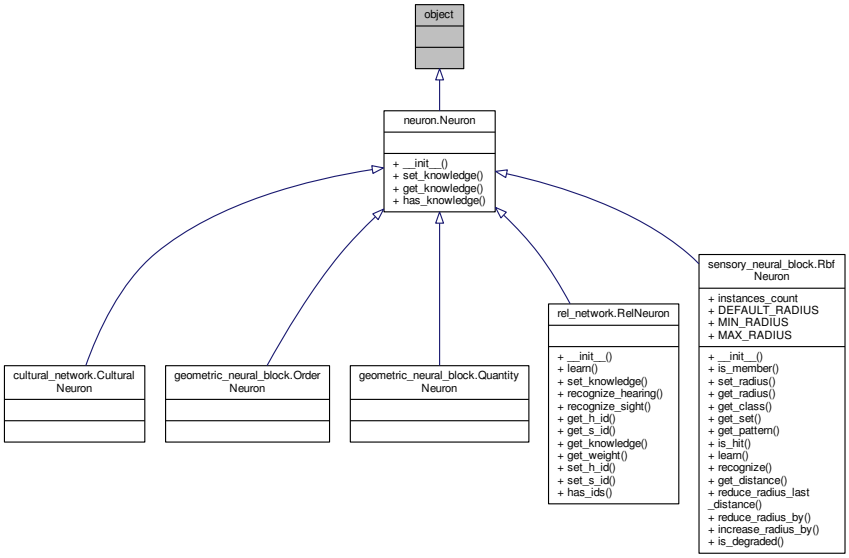


The documentation for this class was generated from the following file:

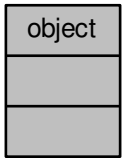
- neuron.py

5.22 object Class Reference

Inheritance diagram for object:



Collaboration diagram for object:



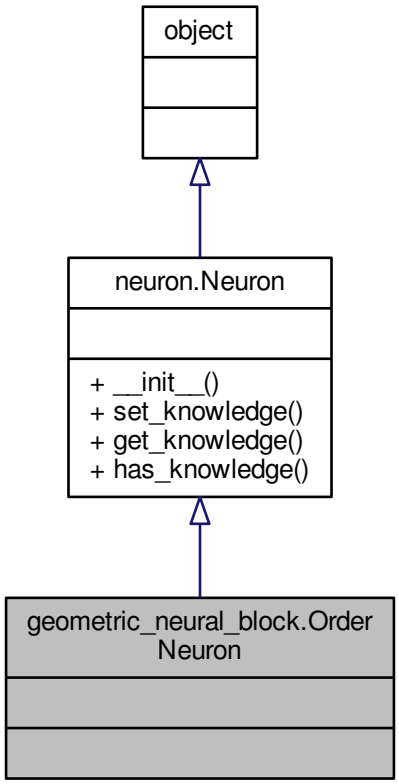
The documentation for this class was generated from the following file:

- neuron.py

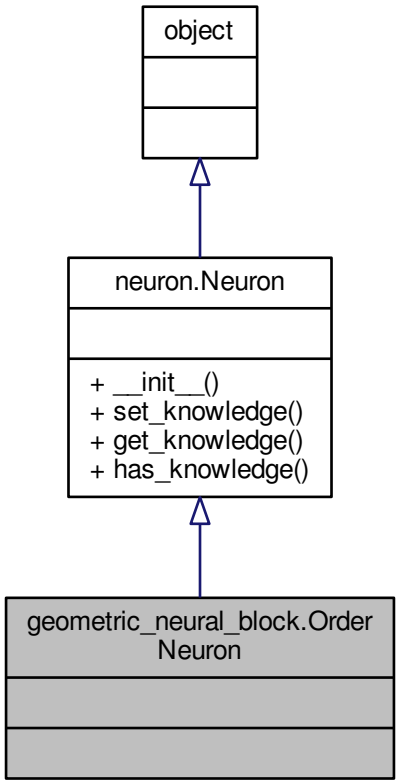
5.23 geometric_neural_block.OrderNeuron Class Reference

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.

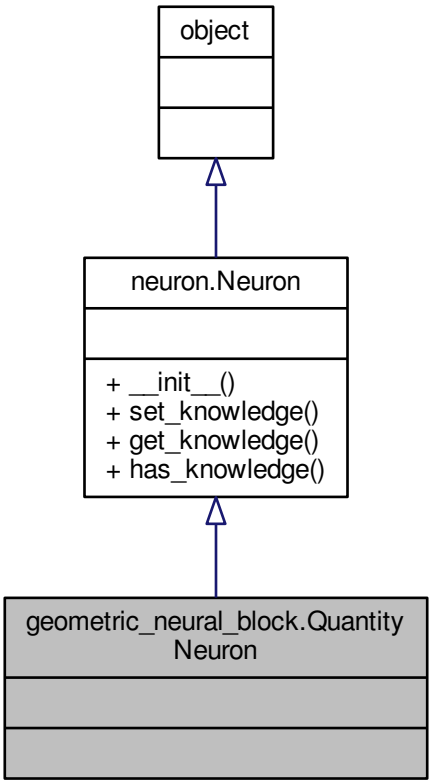
The documentation for this class was generated from the following file:

- geometric_neural_block.py

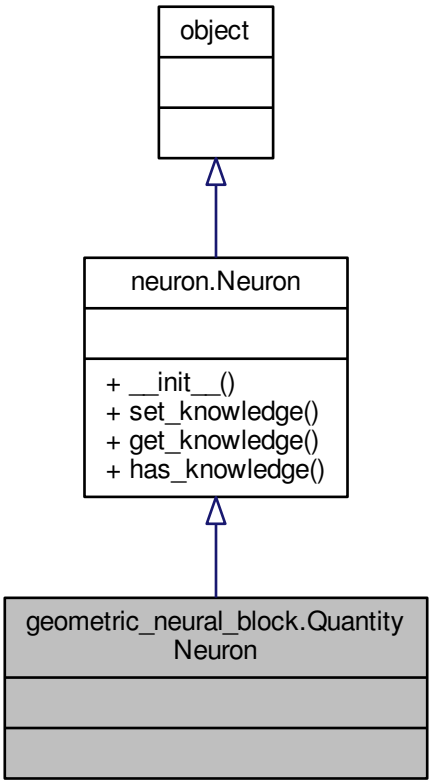
5.24 geometric_neural_block.QuantityNeuron Class Reference

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.

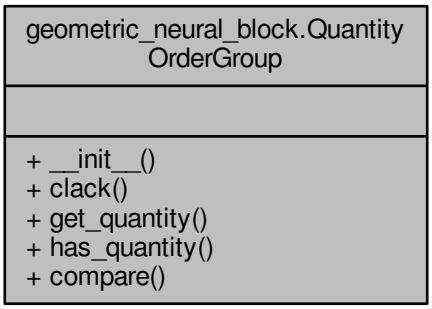
The documentation for this class was generated from the following file:

- 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

knowledge	The knowledge to be compared

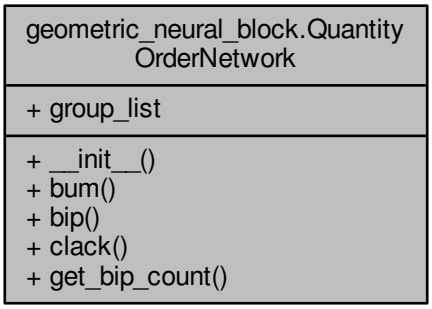
The documentation for this class was generated from the following file:

- 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 )
```

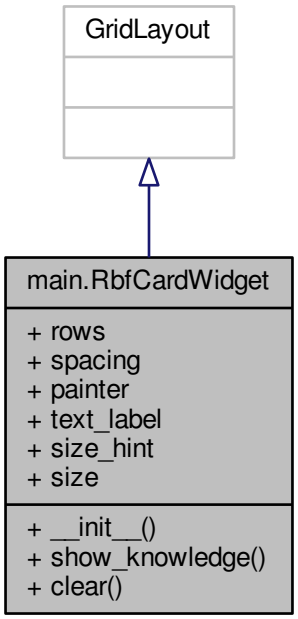
Get position of the currently pointed QuantityOrderGroup as a number of *bips* since the beginning (*bum*).

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

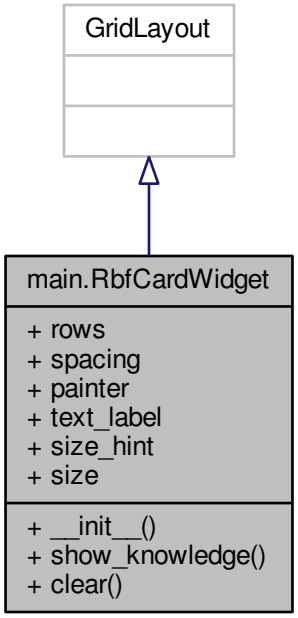
- geometric_neural_block.py

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**

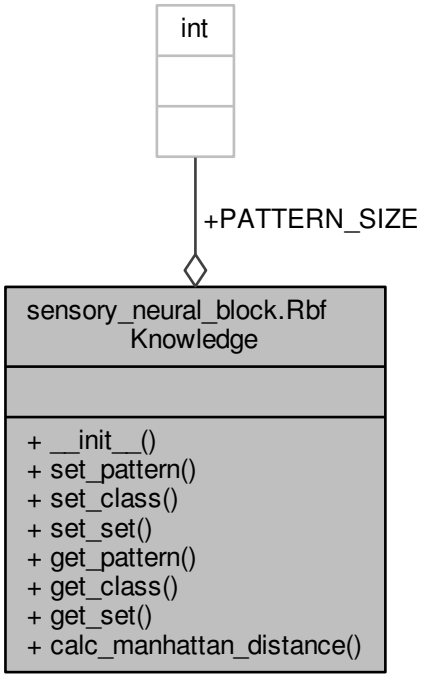
The documentation for this class was generated from the following file:

- 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

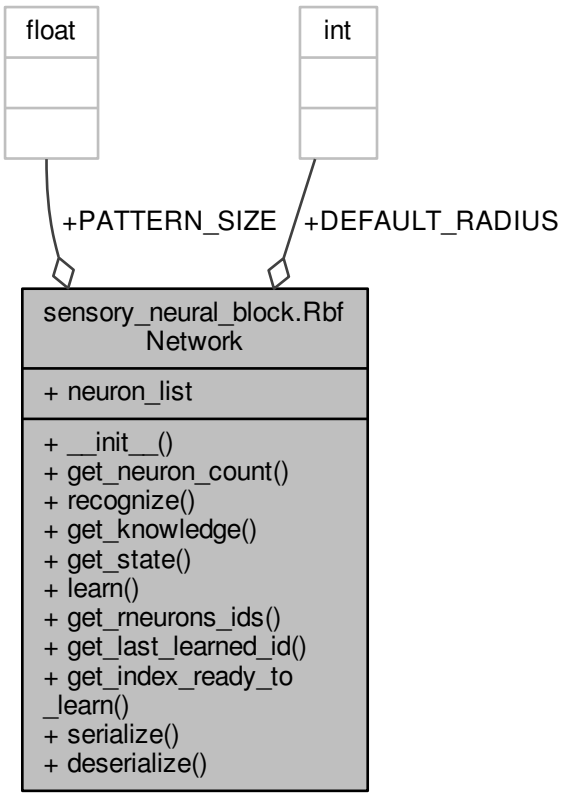
The documentation for this class was generated from the following file:

- 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_rneurons_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.

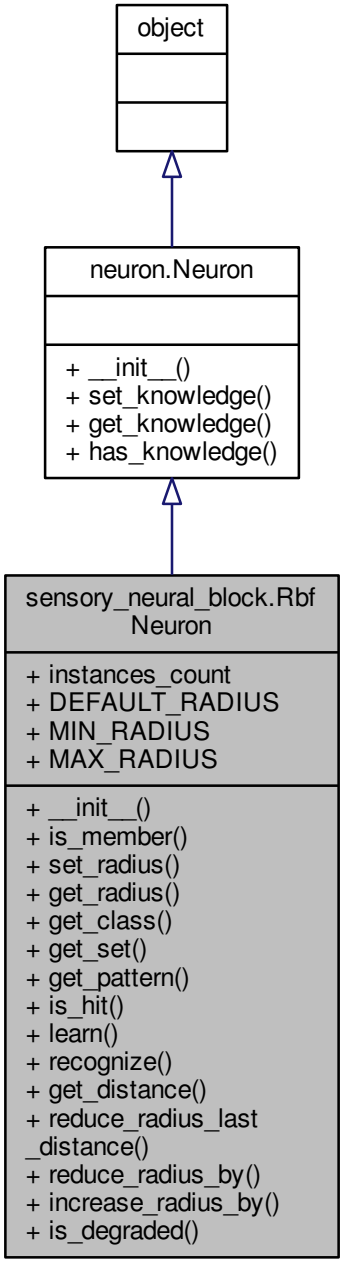
The documentation for this class was generated from the following file:

- sensory_neural_block.py

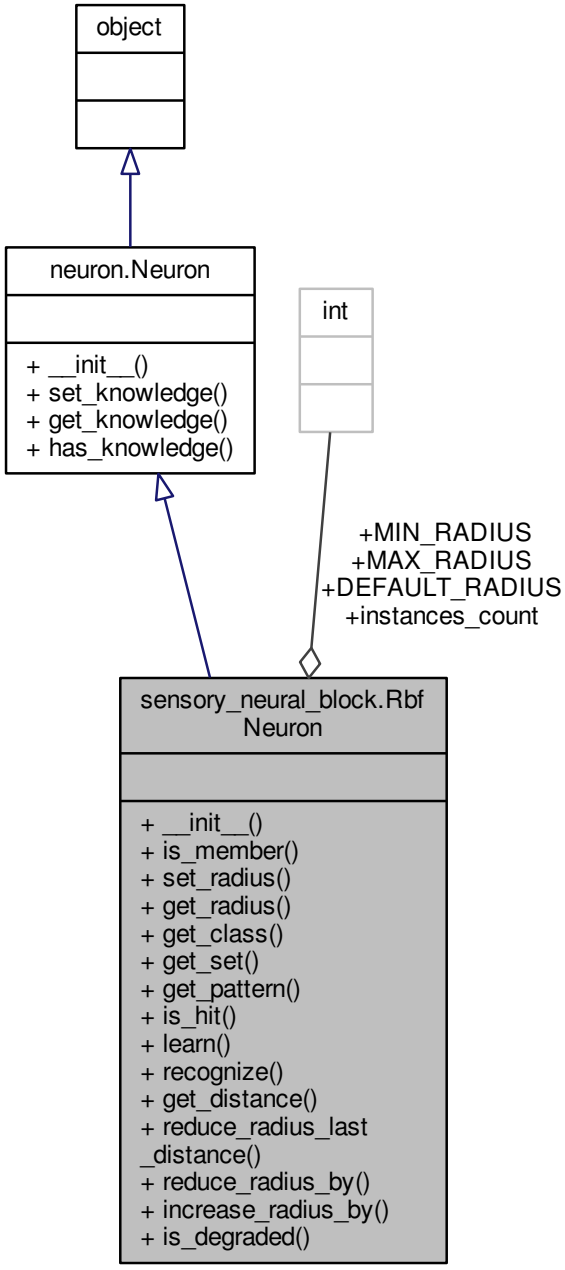
5.30 sensory_neural_block.RbfNeuron Class Reference

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

  Minimun 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

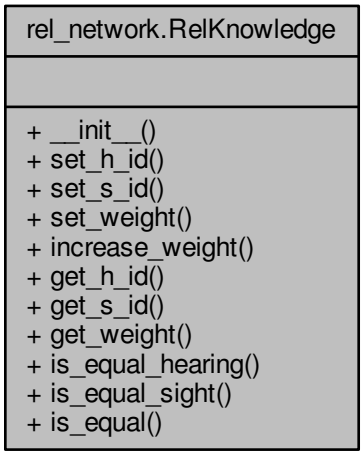
The documentation for this class was generated from the following file:

- 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

weight	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

amount	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()

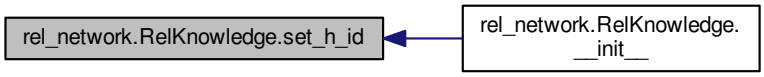
```
def rel_network.RelKnowledge.set_h_id (
              self,
              h_id )
```

Set hearing id.

Parameters

h↩ _id	Integer. Hearing id.

Here is the caller graph for this function:



5.31.2.7 set_s_id()

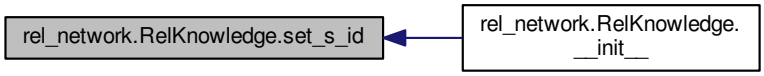
```
def rel_network.RelKnowledge.set_s_id (
              self,
              s_id )
```

Set sight id.

Parameters

s↩ _id	Integer. Sight id.

Here is the caller graph for this function:



5.31.2.8 set_weight()

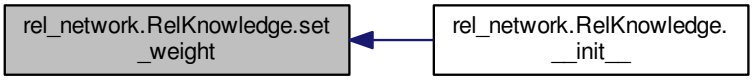
```
def rel_network.RelKnowledge.set_weight (
              self,
              w )
```

Set weight.

Parameters

w	Integer. Weight.

Here is the caller graph for this function:



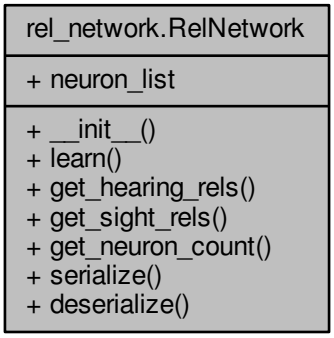
The documentation for this class was generated from the following file:

- 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

neuron_count	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

cls	RelNetwork class
name	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

hearing_rels	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

count	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

sight_rels	RelKnowledge vector

5.32.3.5 learn()

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

Learn new knowledge in ready-to-learn neuron.

Parameters

knowledge	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

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

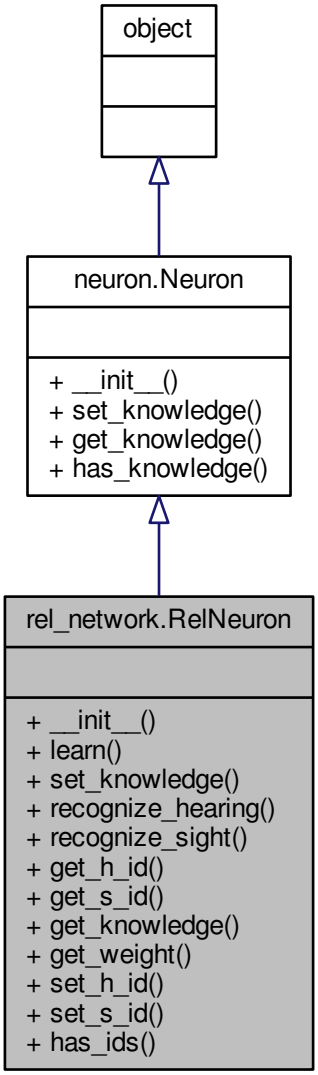
The documentation for this class was generated from the following file:

- rel_network.py

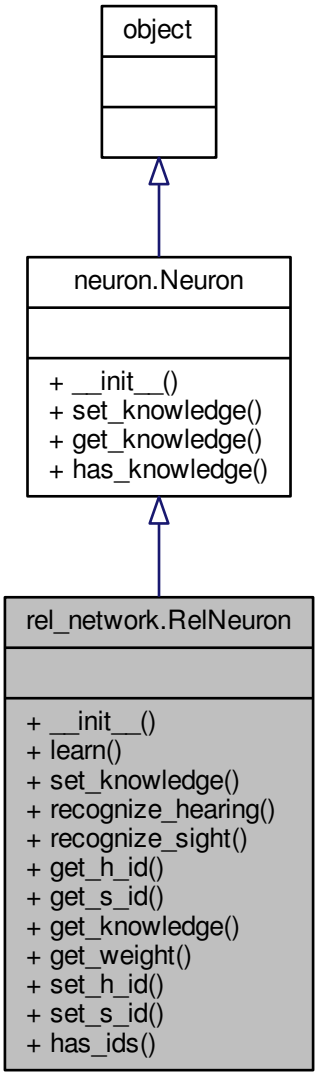
5.33 rel_network.RelNeuron Class Reference

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_id	Integer or None. Hearing id.

Here is the call graph for this function:

5.33.2.2 get_knowledge()

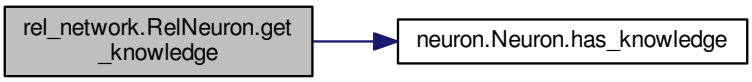
```
def rel_network.RelNeuron.get_knowledge (
              self )
```

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

Return values

knowledge	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

s↩ _id	Integer or None. Sight id.

Here is the call graph for this function:

5.33.2.4 get_weight()

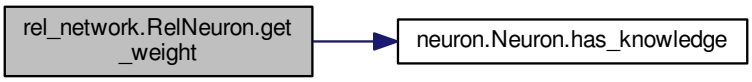
```
def rel_network.RelNeuron.get_weight (
              self )
```

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

Return values

weight	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↩ _id	Hearing id
s↩ _id	Sight id

5.33.2.6 learn()

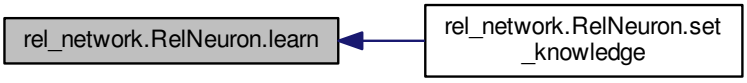
```
def rel_network.RelNeuron.learn (
              self,
              knowledge )
```

Set knowledge of type RelKnowledge.

Parameters

knowledge	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 )
```

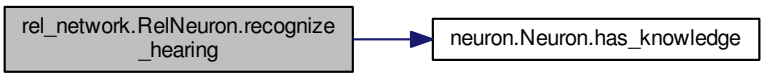
Return True if h_id is recognized as the hearing-id part of the RelKnowledge.

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

h↩ _id	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 )
```

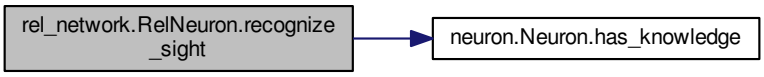
Return true if s_id is recognized as the sight-id part of the relational knowledge.

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

s↩ _id	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

Parameters

h↩ _id	Hearing id.

Here is the call graph for this function:

5.33.2.10 set_knowledge()

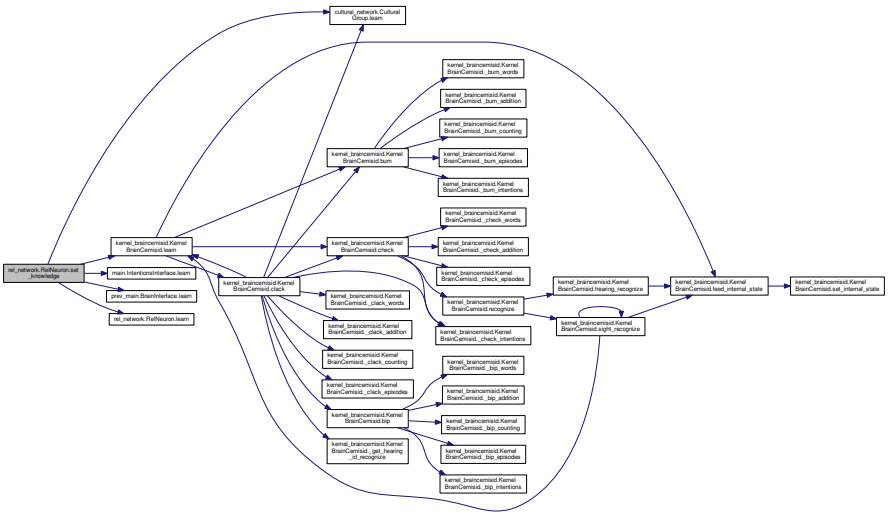
```
def rel_network.RelNeuron.set_knowledge (
            self,
            knowledge )
```

Set knowledge of type RelKnowledge.

Parameters

knowledge	RelKnowledge to be learned

Here is the call graph for this function:



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_id	Sight id

Here is the call graph for this function:

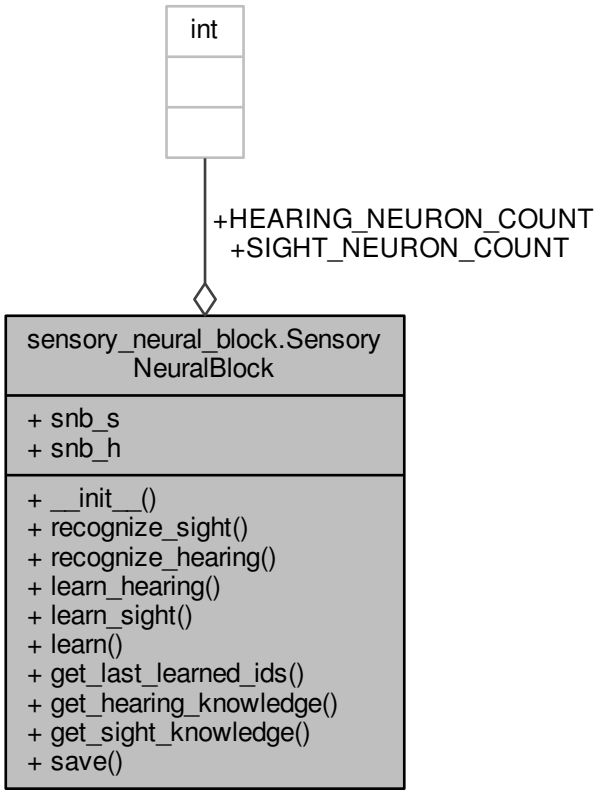
The documentation for this class was generated from the following file:

- 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

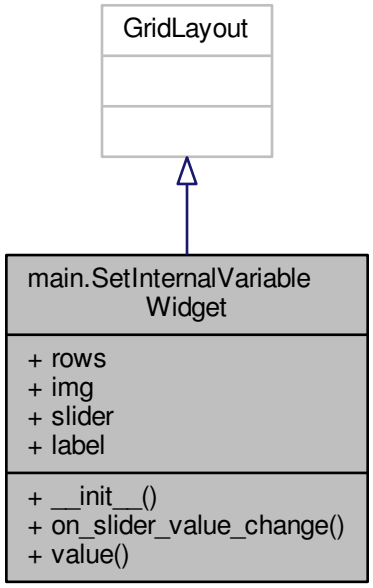
Sensory Neural Block Stores sight and hearing RbfNetworks.

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

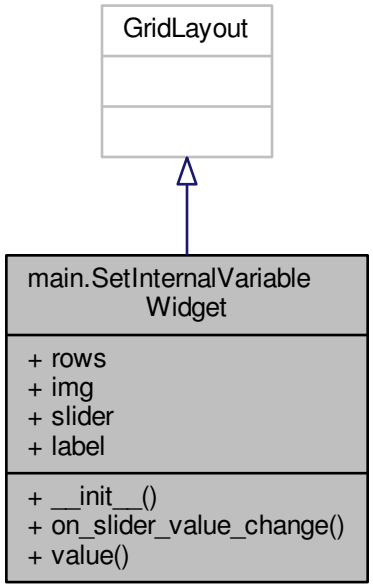
- sensory_neural_block.py

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

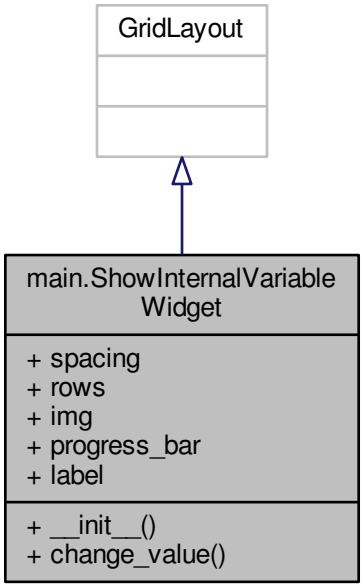
- **rows**
- **img**
- **slider**
- **label**

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

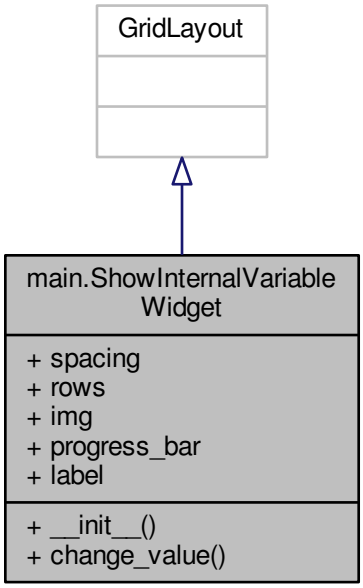
- main.py

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

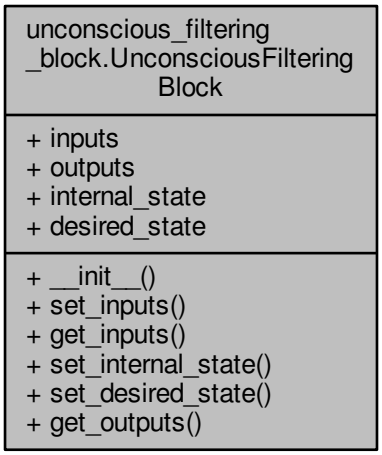
- **spacing**
- **rows**
- **img**
- **progress_bar**
- **label**

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

inputs	CulturalGroup with tail knowledge of class BiologyCultureFeelings

5.37.1.2 get_outputs()

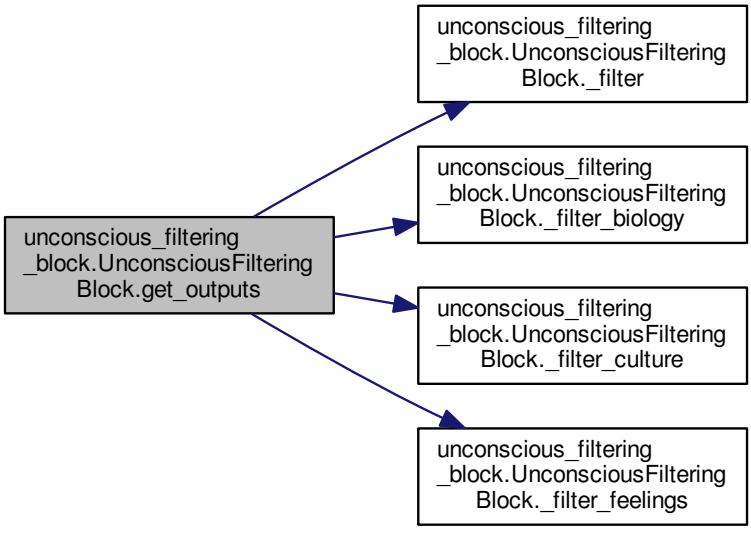
```
def unconscious_filtering_block.UnconsciousFilteringBlock.get_outputs (
            self )
```

Return uncounsciously filtered memories.

Return values

output	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

desired_state	InternalState. Entity's desired state

5.37.1.4 set_inputs()

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

Set block inputs.

Parameters

inputs	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

internal_state	InternalState. Entity's internal state

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

- unconscious_filtering_block.py

Index