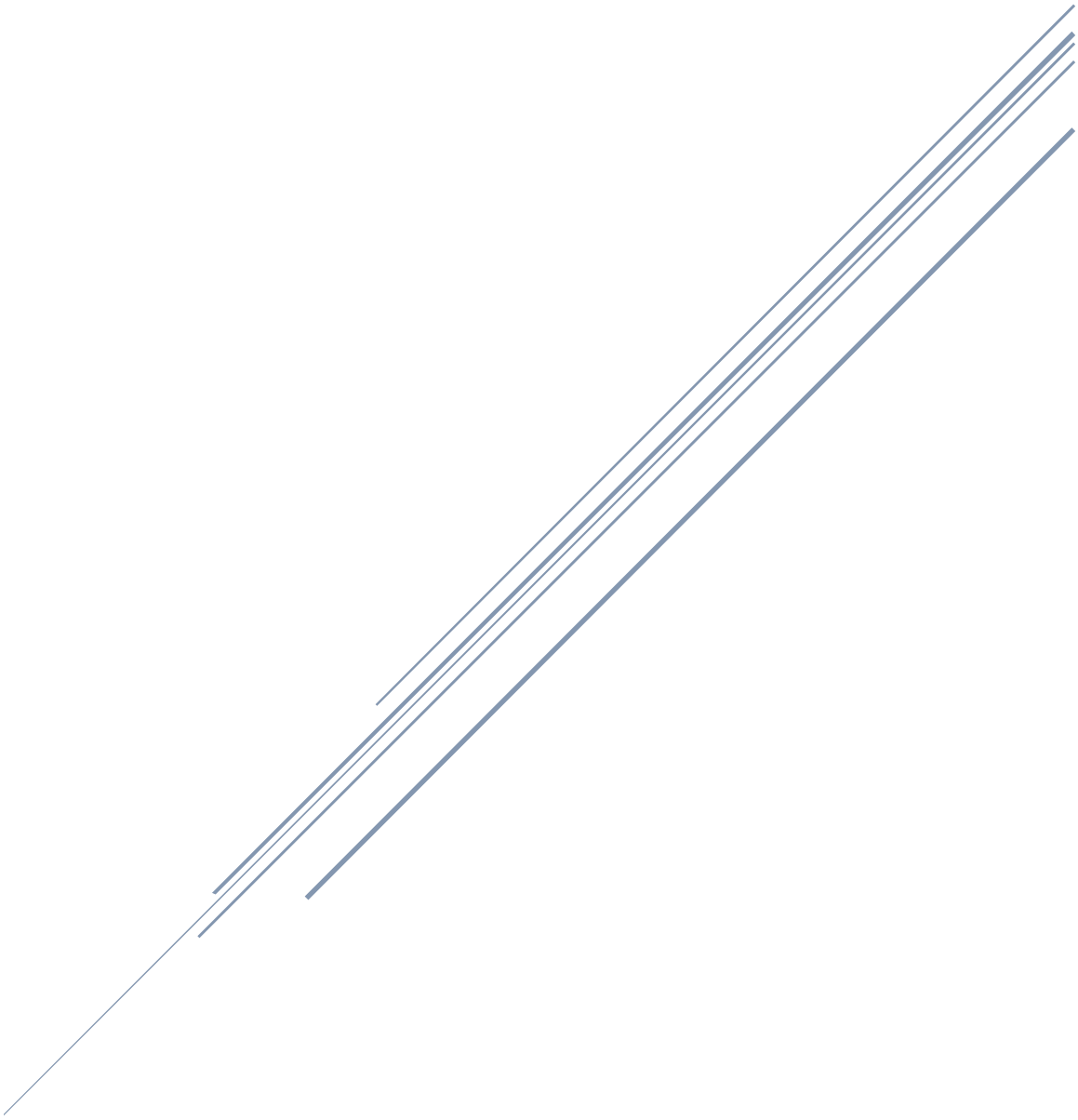
RLM API DOCUMENTATION



Last updated 05/04/2017

RlmNetwork Class

class RlmNetwork

- Controls Network Creation and RLM objects training state.

Constructor:

RlmNetwork()

- default contstructor, creates "RyskampNeuralNetworks" database

RlmNetwork(string **databaseName)**

- string databaseName
  - sets your preferred database name

Methods and Properties

void NewNetwork

- Sets up a new network and sets the network as current network to use in training.

Syntax:

NewNetwork(string **name,** List<rlm_io> **inputs,** List<rlm_io> **outputs** = null**)**

- string name
  - Your preferred network name
- List<rlm_io> inputs
  - List of input types for your created network
- List<rlm_io> outputs
  - List of output types for your created network

bool LoadNetwork

- Loads selected network's data (input types, output types, training data, network settings) from the Database into memory lists.
- Is used as an indicator if there's a need to create a new network.
- Returns true if network is successfully loaded.

Syntax:

LoadNetwork(string **name)**

- string name
  - the network you prefer to load

Alternate Syntax:

LoadNetwork()

- Loads the first network in the database, sorted by ID

Int64 SessionStart

- Sets the state of the session to started
- Returns the Session ID of the current session
- Cannot be used again prior to SessionEnd()

Syntax:

SessionStart()

void SessionEnd

- Halts the current session
- Updates current session's score and Time Stop Property of the session

Syntax:

SessionEnd(double **FinalSessionScore)**

- double FinalSessionScore
  - the score of the current session

void ScoreCycle

- Saves cycle information to database and updates with the score

Syntax:

ScoreCycle(int64 **CycleID,** double **CycleScore)**

- int64 CycleID
  - Unique identifier of the Cycle
- double CycleScore
  - Score the engine attained this cycle

int NumSessions

- The set number of sessions
- This is a required setting for the RlmNetwork because it is part of the calculation on how much intervals to decrease the Randomness and, if set, the Linear Bracket at each session.

int StartRandomness

- The starting percentage of randomness to be used by the engine

int EndRandomness

- The last percentage of randomness where the engine halts

This sets the chance for the RLM to use random outputs instead of getting the best-known solution. The Randomness range (Start - End) decreases after each session.

double MaxLinearBracket

- Maximum value set for the range of Linear Type Training

double MinLinearBracket

- Minimum value set for the range of Linear Type Training

A good indication on when to use this is when an input's value is large and granular enough that you may consider close neighboring values to be the same depending on the linear bracket.

void TrainingDone

- Notifies the RLM that the current training/prediction sessions are finished and you will no longer use the RLM Network instance.
- Also, it allows the DataPersistence events to work properly so this must be called at the very end.

void SetDataPersistenceProgressInterval

- Changes the interval time that the DataPersistenceProgress event is triggered. Default time is 1000ms (1 second)

Syntax:

network.**SetDataPersistenceProgressInterval**(int **milliseconds**);

- **milliseconds** is the amount of time you set the progress interval that the event is triggered

RlmIO Object

class RlmIO

- object type for input and output settings

Constructor:

RlmIO(string **name,** string **dotNetType,** double **min,** double **max,** long **ID = 0)**

- string name
  - Sets RlmIO Name property
- string dotNetType
  - Sets RlmIO DotNetType property which assigns the object type in .NET
- double min
  - Sets RlmIO Min property which sets the minimum range value of the input or output
- double max
  - Sets RlmIO Min property which sets the maximum range value of the input or output
- Long ID
  - Assigns unique identifier to the input/output

RlmCycle Class

class RlmCycle

- handles processing of training data

Methods and Properties

RlmCyclecompleteArgs RunCycle

- starts training

Syntax:

RlmCyclecompleteArgs **RunCycle(**RlmNetwork `rnnNet`**,** int64 **sessionID,** List<RlmIOWithValue> **input_values,** bool **Learn,** List< RlmIOWithValue> **output_values** = null**,** double **cyclescore** = 0.000**,** IEnumerable<RlmIdea> **ideas = null)**

- RlmNetwork `rnnNet`
  - current network being used

- int64 sessionID
    - unique identifier for the session being started
- List<RlmIOWithValue> input_values
    - Inputs with stored values
- bool Learn
    - Indicator that if true, will start training, if false, will run prediction
- List< RlmIOWithValue> output_values
    - Outputs with stored values
- double cyclescore
    - Score of the current cycle
- IEnumerable<RlmIdea> ideas
    - Gives bias to the RLM on what to output
    - For now, our only implementation is the RlmOutputLimiter, which limits the pool of outputs of the RLM to find a solution by giving it a hint.
        - This is applicable when you are aware that other outputs are irrelevant during the training and you want to expedite the training process

RlmCyclecompleteArgs Object

class RlmCyclecompleteArgs

- object type that stores cycle outputs with the rlm network

Constructor:

RlmCyclecompleteArgs (RlmCycleOuput **cycleOutput,** RlmNetwork **network,** RlmNetworkType **rnnType)**

- RlmCycleOuput cycleOutput
- RlmNetwork network
    - current RLM Network
- RlmNetworkType rnnType
    - current RLM Network Type
    - determines if the current network is doing a Supervised, Unsupervised training, or Prediction.

RlmCycleOutput Object

class RlmCycleOutput

- object type that stores cycle output with cycle information

Constructor:

RlmCycleOutput (long **cycleID,** long **solutionID, IEnumerable<Output_Values_Solution> outputsWithVal)**

- long cycleID
    - unique identifier for the cycle
- long solutionID
    - unique identifier for the solution
- IEnumerable<Output_Values_Solution> outputsWithVal