

LibTopoART  
v0.94

Generated by Doxygen 1.8.20

Sat Nov 14 2020 12:04:43

<b>1 Namespace Index</b>	<b>1</b>
1.1 Packages	1
<b>2 Hierarchical Index</b>	<b>1</b>
2.1 Class Hierarchy	1
<b>3 Class Index</b>	<b>4</b>
3.1 Class List	4
<b>4 Namespace Documentation</b>	<b>9</b>
4.1 LibTopoART Namespace Reference	9
4.1.1 Enumeration Type Documentation	12
4.2 LibTopoART_samples Namespace Reference	14
<b>5 Class Documentation</b>	<b>15</b>
5.1 LibTopoART.CategoryInfo Struct Reference	15
5.1.1 Detailed Description	15
5.1.2 Constructor & Destructor Documentation	15
5.1.3 Member Data Documentation	16
5.2 LibTopoART_samples.Episodic_TopoART_sample1 Class Reference	16
5.2.1 Detailed Description	16
5.3 LibTopoART_samples.Episodic_TopoART_sample2 Class Reference	17
5.3.1 Detailed Description	17
5.4 LibTopoART.F2_output Class Reference	17
5.4.1 Detailed Description	17
5.4.2 Constructor & Destructor Documentation	18
5.4.3 Member Data Documentation	18
5.5 LibTopoART.Fast_Episodic_TopoART Class Reference	19
5.5.1 Detailed Description	20
5.5.2 Constructor & Destructor Documentation	20
5.5.3 Member Function Documentation	20
5.5.4 Property Documentation	24
5.6 LibTopoART.Fast_TopoART Class Reference	24
5.6.1 Detailed Description	25
5.6.2 Constructor & Destructor Documentation	25
5.6.3 Member Function Documentation	26
5.7 LibTopoART.Fast_TopoART_AM Class Reference	26
5.7.1 Detailed Description	27
5.7.2 Constructor & Destructor Documentation	28
5.7.3 Member Function Documentation	28
5.7.4 Property Documentation	33
5.8 LibTopoART.Fast_TopoART_base Class Reference	33
5.8.1 Detailed Description	35
5.8.2 Member Function Documentation	35

5.8.3 Member Data Documentation . . . . .	40
5.8.4 Property Documentation . . . . .	40
5.9 LibTopoART.Fast_TopoART_C Class Reference . . . . .	41
5.9.1 Detailed Description . . . . .	42
5.9.2 Constructor & Destructor Documentation . . . . .	42
5.9.3 Member Function Documentation . . . . .	43
5.9.4 Member Data Documentation . . . . .	46
5.9.5 Property Documentation . . . . .	46
5.10 LibTopoART.Fast_TopoART_R Class Reference . . . . .	47
5.10.1 Detailed Description . . . . .	48
5.10.2 Constructor & Destructor Documentation . . . . .	48
5.10.3 Member Function Documentation . . . . .	49
5.10.4 Property Documentation . . . . .	52
5.11 LibTopoART.Hypersphere_TopoART Class Reference . . . . .	53
5.11.1 Detailed Description . . . . .	54
5.11.2 Constructor & Destructor Documentation . . . . .	54
5.11.3 Property Documentation . . . . .	55
5.12 LibTopoART.Hypersphere_TopoART_C Class Reference . . . . .	55
5.12.1 Detailed Description . . . . .	57
5.12.2 Constructor & Destructor Documentation . . . . .	57
5.12.3 Member Function Documentation . . . . .	58
5.12.4 Member Data Documentation . . . . .	60
5.12.5 Property Documentation . . . . .	60
5.13 LibTopoART.IAdaptationStateCheck Interface Reference . . . . .	60
5.13.1 Detailed Description . . . . .	61
5.13.2 Member Function Documentation . . . . .	61
5.14 LibTopoART.IAssociative_recall Interface Reference . . . . .	61
5.14.1 Detailed Description . . . . .	62
5.15 LibTopoART.IByteAccess_Associative_recall Interface Reference . . . . .	62
5.15.1 Detailed Description . . . . .	63
5.15.2 Member Function Documentation . . . . .	63
5.16 LibTopoART.IByteAccess_Episodic_recall Interface Reference . . . . .	65
5.16.1 Detailed Description . . . . .	65
5.16.2 Member Function Documentation . . . . .	66
5.17 LibTopoART.IByteAccess_TopoART Interface Reference . . . . .	67
5.17.1 Detailed Description . . . . .	67
5.17.2 Member Function Documentation . . . . .	67
5.18 LibTopoART.IByteAccess_TopoART_AM Interface Reference . . . . .	68
5.18.1 Detailed Description . . . . .	69
5.18.2 Member Function Documentation . . . . .	69
5.19 LibTopoART.IByteAccess_TopoART_C Interface Reference . . . . .	70
5.19.1 Detailed Description . . . . .	71

5.19.2 Member Function Documentation	71
5.20 LibTopoART.IByteAccess_TopoART_R Interface Reference	73
5.20.1 Detailed Description	73
5.20.2 Member Function Documentation	74
5.21 LibTopoART.ICategoryAccess Interface Reference	75
5.21.1 Detailed Description	75
5.21.2 Member Function Documentation	75
5.22 LibTopoART.IDecimalAccess_Associative_recall Interface Reference	76
5.22.1 Detailed Description	77
5.22.2 Member Function Documentation	77
5.23 LibTopoART.IDecimalAccess_Episodic_recall Interface Reference	78
5.23.1 Detailed Description	79
5.23.2 Member Function Documentation	80
5.24 LibTopoART.IDecimalAccess_TopoART Interface Reference	81
5.24.1 Detailed Description	81
5.24.2 Member Function Documentation	81
5.25 LibTopoART.IDecimalAccess_TopoART_AM Interface Reference	82
5.25.1 Detailed Description	83
5.25.2 Member Function Documentation	83
5.26 LibTopoART.IDecimalAccess_TopoART_C Interface Reference	84
5.26.1 Detailed Description	85
5.26.2 Member Function Documentation	85
5.27 LibTopoART.IDecimalAccess_TopoART_R Interface Reference	87
5.27.1 Detailed Description	87
5.27.2 Member Function Documentation	88
5.28 LibTopoART.IEndRecall Interface Reference	89
5.28.1 Detailed Description	89
5.28.2 Member Function Documentation	89
5.29 LibTopoART.IEpisodic_recall Interface Reference	90
5.29.1 Detailed Description	90
5.30 LibTopoART.IEpisodic_TopoART Interface Reference	91
5.30.1 Detailed Description	91
5.31 LibTopoART.IFast_Associative_recall Interface Reference	91
5.31.1 Detailed Description	92
5.32 LibTopoART.IFast_Episodic_recall Interface Reference	92
5.32.1 Detailed Description	93
5.33 LibTopoART.IFast_Episodic_TopoART Interface Reference	93
5.33.1 Detailed Description	94
5.34 LibTopoART.IFast_TopoART Interface Reference	94
5.34.1 Detailed Description	94
5.35 LibTopoART.IFast_TopoART_AM Interface Reference	95
5.35.1 Detailed Description	95

5.36 LibTopoART.IFast_TopoART_C Interface Reference	95
5.36.1 Detailed Description	96
5.37 LibTopoART.IFast_TopoART_R Interface Reference	96
5.37.1 Detailed Description	96
5.38 LibTopoART.InvalidClassIDException Class Reference	96
5.38.1 Detailed Description	97
5.39 LibTopoART.InvalidFileException Class Reference	97
5.39.1 Detailed Description	97
5.40 LibTopoART.InvalidModuleIndexException Class Reference	97
5.40.1 Detailed Description	97
5.41 LibTopoART.InvalidNumberException Class Reference	97
5.41.1 Detailed Description	97
5.42 LibTopoART.InvalidSizeException Class Reference	97
5.42.1 Detailed Description	97
5.43 LibTopoART.InvalidStateException Class Reference	98
5.43.1 Detailed Description	98
5.44 LibTopoART.ITopoART Interface Reference	98
5.44.1 Detailed Description	98
5.45 LibTopoART.ITopoART_AM Interface Reference	98
5.45.1 Detailed Description	99
5.46 LibTopoART.ITopoART_AM_base Interface Reference	99
5.46.1 Detailed Description	100
5.46.2 Property Documentation	100
5.47 LibTopoART.ITopoART_base Interface Reference	100
5.47.1 Detailed Description	101
5.47.2 Member Function Documentation	101
5.47.3 Property Documentation	102
5.48 LibTopoART.ITopoART_C Interface Reference	102
5.48.1 Detailed Description	103
5.49 LibTopoART.ITopoART_R Interface Reference	103
5.49.1 Detailed Description	104
5.50 LibTopoART.ITopoART_R_base Interface Reference	104
5.50.1 Detailed Description	105
5.50.2 Property Documentation	105
5.51 LibTopoART.LibTopoART_control Struct Reference	106
5.51.1 Detailed Description	106
5.51.2 Member Data Documentation	106
5.52 LibTopoART.LibTopoART_info Struct Reference	106
5.52.1 Detailed Description	107
5.52.2 Member Data Documentation	107
5.53 LibTopoART.TopoART Class Reference	108
5.53.1 Detailed Description	109

5.53.2 Constructor & Destructor Documentation	110
5.53.3 Member Function Documentation	110
5.53.4 Member Data Documentation	113
5.53.5 Property Documentation	114
5.54 LibTopoART_samples.TopoART_AM_sample1 Class Reference	115
5.54.1 Detailed Description	115
5.55 LibTopoART_samples.TopoART_AM_sample2 Class Reference	115
5.55.1 Detailed Description	115
5.56 LibTopoART.TopoART_C Class Reference	116
5.56.1 Detailed Description	117
5.56.2 Constructor & Destructor Documentation	117
5.56.3 Member Function Documentation	118
5.56.4 Member Data Documentation	119
5.56.5 Property Documentation	119
5.57 LibTopoART.TopoART_C_Prediction Struct Reference	119
5.57.1 Detailed Description	120
5.57.2 Constructor & Destructor Documentation	120
5.57.3 Member Data Documentation	120
5.58 LibTopoART_samples.TopoART_C_sample1 Class Reference	121
5.58.1 Detailed Description	121
5.59 LibTopoART_samples.TopoART_C_sample2 Class Reference	121
5.59.1 Detailed Description	121
5.60 LibTopoART.TopoART_R Class Reference	121
5.60.1 Detailed Description	122
5.60.2 Constructor & Destructor Documentation	122
5.60.3 Member Function Documentation	123
5.60.4 Property Documentation	125
5.61 LibTopoART.TopoART_R_Prediction<_ElementType> Struct Template Reference	125
5.61.1 Detailed Description	126
5.61.2 Constructor & Destructor Documentation	126
5.61.3 Member Function Documentation	126
5.61.4 Member Data Documentation	126
5.62 LibTopoART_samples.TopoART_R_sample1 Class Reference	127
5.62.1 Detailed Description	127
5.63 LibTopoART_samples.TopoART_R_sample2 Class Reference	127
5.63.1 Detailed Description	127
5.64 LibTopoART_samples.TopoART_R_sample3 Class Reference	127
5.64.1 Detailed Description	128
5.65 LibTopoART_samples.TopoART_sample1 Class Reference	128
5.65.1 Detailed Description	128
5.66 LibTopoART_samples.TopoART_sample2 Class Reference	128
5.66.1 Detailed Description	128

5.67 <a href="#">LibTopoART_samples.TopoART_sample3 Class Reference</a> . . . . .	128
5.67.1 <a href="#">Detailed Description</a> . . . . .	129
<b><a href="#">Index</a></b>	<b>131</b>

## 1 Namespace Index

### 1.1 Packages

Here are the packages with brief descriptions (if available):

<a href="#">LibTopoART</a>	9
<a href="#">LibTopoART_samples</a>	14

## 2 Hierarchical Index

### 2.1 Class Hierarchy

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

<b>LibTopoART.CategoryInfo</b>	<b>15</b>
<b>LibTopoART_samples.Episodic_TopopART_sample1</b>	<b>16</b>
<b>LibTopoART_samples.Episodic_TopopART_sample2</b>	<b>17</b>
<b>LibTopoART.F2_output</b>	<b>17</b>
<b>LibTopoART.IAdaptationStateCheck</b>	<b>60</b>
<b>LibTopoART.ITopoART</b>	<b>98</b>
<b>LibTopoART.IEpisodic_TopopART</b>	<b>91</b>
<b>LibTopoART.IFast_Episodic_TopopART</b>	<b>93</b>
<b>LibTopoART.Fast_Episodic_TopopART</b>	<b>19</b>
<b>LibTopoART.IFast_TopopART</b>	<b>94</b>
<b>LibTopoART.Fast_TopopART_base</b>	<b>33</b>
<b>LibTopoART.Fast_Episodic_TopopART</b>	<b>19</b>
<b>LibTopoART.Fast_TopopART</b>	<b>24</b>
<b>LibTopoART.Fast_TopopART_AM</b>	<b>26</b>
<b>LibTopoART.Fast_TopopART_C</b>	<b>41</b>
<b>LibTopoART.Fast_TopopART_R</b>	<b>47</b>
<b>LibTopoART.IFast_Episodic_TopopART</b>	<b>93</b>

LibTopoART.IFast_TopoART_AM	95
LibTopoART.Fast_TopoART_AM	26
LibTopoART.IFast_TopoART_C	95
LibTopoART.Fast_TopoART_C	41
LibTopoART.IFast_TopoART_R	96
LibTopoART.Fast_TopoART_R	47
LibTopoART.ITopoART_AM	98
LibTopoART.IFast_TopoART_AM	95
LibTopoART.ITopoART_C	102
LibTopoART.Hypersphere_TopoART_C	55
LibTopoART.IFast_TopoART_C	95
LibTopoART.TopoART_C	116
LibTopoART.ITopoART_R	103
LibTopoART.IFast_TopoART_R	96
LibTopoART.TopoART_R	121
LibTopoART.TopoART	108
LibTopoART.Hypersphere_TopoART	53
LibTopoART.Hypersphere_TopoART_C	55
LibTopoART.TopoART_C	116
LibTopoART.TopoART_R	121
LibTopoART.ICategoryAccess	75
LibTopoART.Fast_TopoART_base	33
LibTopoART.TopoART	108
LibTopoART.IEndRecall	89
LibTopoART.IByteAccess_Associative_recall	62
LibTopoART.IFast_Associative_recall	91
LibTopoART.IFast_TopoART_AM	95
LibTopoART.IByteAccess_Episodic_recall	65
LibTopoART.IFast_Episodic_recall	92
LibTopoART.IFast_Episodic_TopoART	93
LibTopoART.IDecimalAccess_Associative_recall	76
LibTopoART.IAssociative_recall	61



LibTopoART.IFast_Associative_recall	91
LibTopoART.ITopoART_AM	98
LibTopoART.IDecimalAccess_Episodic_recall	78
LibTopoART.IEpisodic_recall	90
LibTopoART.IEpisodic_TopoART	91
LibTopoART.IFast_Episodic_recall	92
LibTopoART.InvalidClassIDException	96
LibTopoART.InvalidFileException	97
LibTopoART.InvalidModuleIndexException	97
LibTopoART.InvalidNumberException	97
LibTopoART.InvalidSizeException	97
LibTopoART.InvalidStateException	98
LibTopoART.ITopoART_base	100
LibTopoART.IByteAccess_TopoART	67
LibTopoART.IFast_TopoART	94
LibTopoART.IByteAccess_TopoART_C	70
LibTopoART.IFast_TopoART_C	95
LibTopoART.IDecimalAccess_TopoART	81
LibTopoART.ITopoART	98
LibTopoART.IDecimalAccess_TopoART_C	84
LibTopoART.ITopoART_C	102
LibTopoART.ITopoART_AM_base	99
LibTopoART.IByteAccess_TopoART_AM	68
LibTopoART.IFast_TopoART_AM	95
LibTopoART.IDecimalAccess_TopoART_AM	82
LibTopoART.ITopoART_AM	98
LibTopoART.ITopoART_R_base	104
LibTopoART.IByteAccess_TopoART_R	73
LibTopoART.IFast_TopoART_R	96
LibTopoART.IDecimalAccess_TopoART_R	87
LibTopoART.ITopoART_R	103
LibTopoART.LibTopoART_control	106

<code>LibTopoART.LibTopoART_info</code>	106
<code>LibTopoART_samples.TopoART_AM_sample1</code>	115
<code>LibTopoART_samples.TopoART_AM_sample2</code>	115
<code>LibTopoART.TopoART_C_Prediction</code>	119
<code>LibTopoART_samples.TopoART_C_sample1</code>	121
<code>LibTopoART_samples.TopoART_C_sample2</code>	121
<code>LibTopoART.TopoART_R_Prediction&lt;_ElementType&gt;</code>	125
<code>LibTopoART_samples.TopoART_R_sample1</code>	127
<code>LibTopoART_samples.TopoART_R_sample2</code>	127
<code>LibTopoART_samples.TopoART_R_sample3</code>	127
<code>LibTopoART_samples.TopoART_sample1</code>	128
<code>LibTopoART_samples.TopoART_sample2</code>	128
<code>LibTopoART_samples.TopoART_sample3</code>	128

## 3 Class Index

### 3.1 Class List

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

<code>LibTopoART.CategoryInfo</code>	
Struct <code>CategoryInfo</code> summarises information about a node's category	15
<code>LibTopoART_samples.Episodic_TopopART_sample1</code>	
Episodic clustering sample using synthetic two-dimensional data. [C#]	16
<code>LibTopoART_samples.Episodic_TopopART_sample2</code>	
Episodic clustering sample using real-world video data. [C#]	17
<code>LibTopoART.F2_output</code>	
Class <code>F2_output</code> provides the output of a single <code>TopopART</code> module. It is a compressed version of the output vectors <code>y</code> and <code>c</code>	17
<code>LibTopoART.Fast_Episodic_TopopART</code>	
Class <code>Fast_Episodic_TopopART</code> provides an implementation of the Episodic <code>TopopART</code> neural network as proposed in "Marko Tscherepanow, Sina Kühnel, and Sören Riechers (2012). Episodic Clustering of Data Streams Using a Topology-Learning Neural Network. In Proceedings of the European Conference on Artificial Intelligence (ECAI), Workshop on Active and Incremental Learning (AIL), pp. 24-29. Montpellier, France."	19
<code>LibTopoART.Fast_TopopART</code>	
Class <code>Fast_TopopART</code> provides an implementation of the <code>TopopART</code> neural network as proposed in "Marko Tscherepanow (2010). <code>TopopART</code> : A topology learning hierarchical ART network. In Proceedings of the International Conference on Artificial Neural Networks (ICANN), LNCS 6354, pp. 157–167. Berlin, Germany: Springer."	24

**LibTopoART.Fast\_TopoART\_AM**

Class **Fast\_TopoART\_AM** provides an implementation of the TopoART-AM neural network as proposed in "Marko Tscherepanow, Marco Kortkamp and Marc Kammer (2011). A Hierarchical ART Network for the Stable Incremental Learning of Topological Structures and Associations from Noisy Data. *Neural Networks* 24(8): 906-916. Elsevier."

26

**LibTopoART.Fast\_TopoART\_base**

Base class providing functionality common to several **TopoART** networks

33

**LibTopoART.Fast\_TopoART\_C**

Class **Fast\_TopoART\_C** provides an implementation of the TopoART-C neural network as proposed in "Marko Tscherepanow and Sören Riechers (2012). An Incremental On-line Classifier for Imbalanced, Incomplete, and Noisy Data. In *Proceedings of the European Conference on Artificial Intelligence (ECAI), Workshop on Active and Incremental Learning (AIL)*, pp. 18-23. Montpellier, France."

41

**LibTopoART.Fast\_TopoART\_R**

Class **Fast\_TopoART\_R** provides an implementation of the TopoART-R neural network as proposed in "Marko Tscherepanow (2011). An Extended **TopoART** Network for the Stable On-Line Learning of Regression Functions. In *Proceedings of the International Conference on Neural Information Processing (ICONIP), LNCS 7063*, pp. 562–571. Berlin, Germany: Springer."

47

**LibTopoART.Hypersphere\_TopoART**

Class **Hypersphere\_TopoART** provides an implementation of the Hypersphere **TopoART** neural network as proposed in "Marko Tscherepanow (2012). Incremental On-line Clustering with a Topology-Learning Hierarchical ART Neural Network Using Hyperspherical Categories. In *Poster and Industry Proceedings of the Industrial Conference on Data Mining (ICDM)*, pp. 22–34. Fockendorf, Germany: ibai-publishing."

53

**LibTopoART.Hypersphere\_TopoART\_C**

Class **Hypersphere\_TopoART\_C** provides an implementation of the Hypersphere TopoART-C neural network. Hypersphere TopoART-C is a combination of Hypersphere **TopoART** as proposed in "Marko Tscherepanow (2012). Incremental On-line Clustering with a Topology-Learning Hierarchical ART Neural Network Using Hyperspherical Categories. In *Poster and Industry Proceedings of the Industrial Conference on Data Mining (ICDM)*, pp. 22–34. Fockendorf, Germany: ibai-publishing." and TopoART-C as proposed in "Marko Tscherepanow and Sören Riechers (2012). An Incremental On-line Classifier for Imbalanced, Incomplete, and Noisy Data. In *Proceedings of the European Conference on Artificial Intelligence (ECAI), Workshop on Active and Incremental Learning (AIL)*, pp. 18-23. Montpellier, France."

55

**LibTopoART.IAdaptationStateCheck**

Interface enabling checks whether certain adaptations of a network occurred

60

**LibTopoART.IAssociative\_recall**

Interface summarising the associative recall functionality using stimulus elements and recall result elements of type `decimal`

61

**LibTopoART.IByteAccess\_Associative\_recall**

Interface providing access to the basic associative recall functionality using stimulus elements and recall result elements of type `byte`. Individual values may use the complete value range from 0 to 255. They are internally converted from/to the **TopoART** input interval [0,1]

62

**LibTopoART.IByteAccess\_Episodic\_recall**

Interface providing access to the basic episodic recall functionality using stimulus elements and recall result elements of type `byte`. Individual values may use the complete value range from 0 to 255. They are internally converted from/to the **TopoART** input interval [0,1]

65

<a href="#">LibTopoART.IByteAccess_Top ART</a>	
Interface providing access to the basic <a href="#">TopoART</a> functionality using input elements of type <code>byte</code> . Individual values may use the complete value range from 0 to 255. They are internally converted to the <a href="#">TopoART</a> input interval [0,1]	67
<a href="#">LibTopoART.IByteAccess_Top ART_AM</a>	
Interface providing access to the basic TopoART-AM functionality using input elements of type <code>byte</code> . Individual values may use the complete value range from 0 to 255. They are internally converted to the <a href="#">TopoART</a> input interval [0,1]	68
<a href="#">LibTopoART.IByteAccess_Top ART_C</a>	
Interface providing access to the basic TopoART-C functionality using input elements of type <code>byte</code> . Individual values may use the complete value range from 0 to 255. They are internally converted to the <a href="#">TopoART</a> input interval [0,1]	70
<a href="#">LibTopoART.IByteAccess_Top ART_R</a>	
Interface providing access to the basic TopoART-R functionality using input elements of type <code>byte</code> . Individual values may use the complete value range from 0 to 255. They are internally converted to the <a href="#">TopoART</a> input interval [0,1]	73
<a href="#">LibTopoART.ICategoryAccess</a>	
Interface providing access to the learnt categories, e.g for drawing	75
<a href="#">LibTopoART.IDecimalAccess_Associative_recall</a>	
Interface providing access to the basic associative recall functionality using stimulus elements and recall result elements of type <code>decimal</code> . Individual stimulus values must lie within the <a href="#">TopoART</a> input interval [0,1]	76
<a href="#">LibTopoART.IDecimalAccess_Episodic_recall</a>	
Interface providing access to the basic episodic recall functionality using stimulus elements and recall result elements of type <code>decimal</code> . Individual stimulus values must lie within the <a href="#">TopoART</a> input interval [0,1]	78
<a href="#">LibTopoART.IDecimalAccess_Top ART</a>	
Interface providing access to the basic <a href="#">TopoART</a> functionality using input elements of type <code>decimal</code> . Individual values must lie within the <a href="#">TopoART</a> input interval [0,1]	81
<a href="#">LibTopoART.IDecimalAccess_Top ART_AM</a>	
Interface providing access to the basic TopoART-AM functionality using input elements of type <code>decimal</code> . Individual values must lie within the <a href="#">TopoART</a> input interval [0,1]	82
<a href="#">LibTopoART.IDecimalAccess_Top ART_C</a>	
Interface providing access to the basic TopoART-C functionality using input elements of type <code>decimal</code> . Individual values must lie within the <a href="#">TopoART</a> input interval [0,1]	84
<a href="#">LibTopoART.IDecimalAccess_Top ART_R</a>	
Interface providing access to the basic TopoART-R functionality using input elements of type <code>decimal</code> . Individual values must lie within the <a href="#">TopoART</a> input interval [0,1]	87
<a href="#">LibTopoART.IEndRecall</a>	
Interface summarising the type-independent functionality to stop the recall process	89
<a href="#">LibTopoART.IEpisodic_recall</a>	
Interface summarising the episodic recall functionality using stimulus elements and recall result elements of type <code>decimal</code>	90
<a href="#">LibTopoART.IEpisodic_Top ART</a>	
Interface summarising the Episodic <a href="#">TopoART</a> functionality including learning, prediction, episodic recall using input elements, stimulus elements, and recall result elements of type <code>decimal</code> as well as adaptation state control	91

<a href="#">LibTopoART.IFast_Associative_recall</a>	
Interface summarising the associative recall functionality using stimulus elements and recall result elements of type <code>byte</code> or of type <code>decimal</code>	91
<a href="#">LibTopoART.IFast_Episodic_recall</a>	
Interface summarising the episodic recall functionality using stimulus elements and recall result elements of type <code>byte</code> or of type <code>decimal</code>	92
<a href="#">LibTopoART.IFast_Episodic_TopoART</a>	
Interface summarising the Episodic <a href="#">TopoART</a> functionality including learning, prediction, episodic recall using input elements, stimulus elements, and recall result elements of type <code>byte</code> or of type <code>decimal</code> as well as adaptation state control	93
<a href="#">LibTopoART.IFast_TopoART</a>	
Interface summarising the <a href="#">TopoART</a> functionality including learning and prediction using input elements of type <code>byte</code> or of type <code>decimal</code> as well as adaptation state control	94
<a href="#">LibTopoART.IFast_TopoART_AM</a>	
Interface summarising the Episodic <a href="#">TopoART</a> functionality including learning, prediction, episodic recall using input elements, stimulus elements, and recall result elements of type <code>byte</code> or of type <code>decimal</code> as well as adaptation state control	95
<a href="#">LibTopoART.IFast_TopoART_C</a>	
Interface summarising the TopoART-C functionality including learning and prediction using input elements of type <code>byte</code> or of type <code>decimal</code> as well as adaptation state control	95
<a href="#">LibTopoART.IFast_TopoART_R</a>	
Interface summarising the TopoART-R functionality including learning and prediction using input elements and output elements of type <code>byte</code> or of type <code>decimal</code> as well as adaptation state control	96
<a href="#">LibTopoART.InvalidClassIDException</a>	
Exception signalling an invalid class ID	96
<a href="#">LibTopoART.InvalidFileException</a>	
Exception signalling an invalid file	97
<a href="#">LibTopoART.InvalidModuleIndexException</a>	
Exception signalling an invalid module index	97
<a href="#">LibTopoART.InvalidNumberException</a>	
Exception signalling an invalid number	97
<a href="#">LibTopoART.InvalidSizeException</a>	
Exception signalling an invalid size	97
<a href="#">LibTopoART.InvalidStateException</a>	
Exception signalling an invalid state of the neural network	98
<a href="#">LibTopoART.ITopoART</a>	
Interface summarising the <a href="#">TopoART</a> functionality including learning and prediction using input elements of type <code>decimal</code> as well as adaptation state control	98
<a href="#">LibTopoART.ITopoART_AM</a>	
Interface summarising the TopoART-AM functionality including learning, prediction, associative recall using input elements, stimulus elements, and recall result elements of type <code>decimal</code> as well as adaptation state control	98
<a href="#">LibTopoART.ITopoART_AM_base</a>	
Interface summarising the basic TopoART-AM functionality excluding learning and prediction	99

<a href="#">LibTopoART.ITopoART_base</a>	Interface summarising the basic <a href="#">TopoART</a> functionality excluding learning and prediction	100
<a href="#">LibTopoART.ITopoART_C</a>	Interface summarising the TopoART-C functionality including learning and prediction using input elements of type <code>decimal</code> as well as adaptation state control	102
<a href="#">LibTopoART.ITopoART_R</a>	Interface summarising the TopoART-R functionality including learning and prediction using input elements and output elements of type <code>decimal</code> as well as adaptation state control	103
<a href="#">LibTopoART.ITopoART_R_base</a>	Interface summarising the basic TopoART-R functionality excluding learning and prediction	104
<a href="#">LibTopoART.LibTopoART_control</a>	Struct <a href="#">LibTopoART_control</a> provides fields to control the general behaviour of <a href="#">LibTopoART</a>	106
<a href="#">LibTopoART.LibTopoART_info</a>	Struct <a href="#">LibTopoART_info</a> provides some metainformation regarding the respective implementation of <a href="#">LibTopoART</a>	106
<a href="#">LibTopoART.TopoART</a>	Class <a href="#">TopoART</a> provides an implementation of the <a href="#">TopoART</a> neural network as proposed in "Marko Tscherepanow (2010). <a href="#">TopoART</a> : A topology learning hierarchical ART network. In Proceedings of the International Conference on Artificial Neural Networks (ICANN), LNCS 6354, pp. 157–167. Berlin, Germany: Springer."	108
<a href="#">LibTopoART_samples.TopoART_AM_sample1</a>	Sample using TopoART-AM with synthetic two-dimensional data. [C#]	115
<a href="#">LibTopoART_samples.TopoART_AM_sample2</a>	Learning of bidirectional associations between images. [F#]	115
<a href="#">LibTopoART.TopoART_C</a>	Class <a href="#">TopoART_C</a> provides an implementation of the TopoART-C neural network as proposed in "Marko Tscherepanow and Sören Riechers (2012). An Incremental On-line Classifier for Imbalanced, Incomplete, and Noisy Data. In Proceedings of the European Conference on Artificial Intelligence (ECAI), Workshop on Active and Incremental Learning (AIL), pp. 18-23. Montpellier, France."	116
<a href="#">LibTopoART.TopoART_C_Prediction</a>	Struct <a href="#">TopoART_C_Prediction</a> contains a prediction made by a TopoART-C network	119
<a href="#">LibTopoART_samples.TopoART_C_sample1</a>	Simple classification sample. [C#]	121
<a href="#">LibTopoART_samples.TopoART_C_sample2</a>	Classification sample using more complex synthetic two-dimensional data with associated class IDs. [C#]	121
<a href="#">LibTopoART.TopoART_R</a>	Class <a href="#">TopoART_R</a> provides an implementation of the TopoART-R neural network as proposed in "Marko Tscherepanow (2011). An Extended <a href="#">TopoART</a> Network for the Stable On-Line Learning of Regression Functions. In Proceedings of the International Conference on Neural Information Processing (ICONIP), LNCS 7063, pp. 562–571. Berlin, Germany: Springer."	121
<a href="#">LibTopoART.TopoART_R_Prediction&lt;_ElementType&gt;</a>	Struct <a href="#">TopoART_R_Prediction</a> contains a prediction made by a TopoART-R network	125
<a href="#">LibTopoART_samples.TopoART_R_sample1</a>	Regression sample using TopoART-R. (simplified version) [C#]	127

<a href="#">LibTopoART_samples.TopoART_R_sample2</a>	
Regression sample using TopoART-R. (advanced version) [C#]	127
<a href="#">LibTopoART_samples.TopoART_R_sample3</a>	
Pixel-wise regression analysis of an image using TopoART-R. [F#]	127
<a href="#">LibTopoART_samples.TopoART_sample1</a>	
Simple clustering sample. [C#]	128
<a href="#">LibTopoART_samples.TopoART_sample2</a>	
Clustering sample using more complex synthetic two-dimensional data. [C#]	128
<a href="#">LibTopoART_samples.TopoART_sample3</a>	
Clustering sample using very noisy synthetic two-dimensional data. [VB]	128

## 4 Namespace Documentation

### 4.1 LibTopoART Namespace Reference

#### Classes

- struct [CategoryInfo](#)  
*Struct [CategoryInfo](#) summarises information about a node's category.*
- class [F2\\_output](#)  
*Class [F2\\_output](#) provides the output of a single [TopoART](#) module. It is a compressed version of the output vectors  $y$  and  $c$ .*
- class [Fast\\_Episodic\\_TopoART](#)  
*Class [Fast\\_Episodic\\_TopoART](#) provides an implementation of the Episodic [TopoART](#) neural network as proposed in "Marko Tscherepanow, Sina Kühnel, and Sören Riechers (2012). Episodic Clustering of Data Streams Using a Topology-Learning Neural Network. In Proceedings of the European Conference on Artificial Intelligence (ECAI), Workshop on Active and Incremental Learning (AIL), pp. 24-29. Montpellier, France."*
- class [Fast\\_TopoART](#)  
*Class [Fast\\_TopoART](#) provides an implementation of the [TopoART](#) neural network as proposed in "Marko Tscherepanow (2010). [TopoART](#): A topology learning hierarchical ART network. In Proceedings of the International Conference on Artificial Neural Networks (ICANN), LNCS 6354, pp. 157-167. Berlin, Germany: Springer."*
- class [Fast\\_TopoART\\_AM](#)  
*Class [Fast\\_TopoART\\_AM](#) provides an implementation of the TopoART-AM neural network as proposed in "Marko Tscherepanow, Marco Kortkamp and Marc Kammer (2011). A Hierarchical ART Network for the Stable Incremental Learning of Topological Structures and Associations from Noisy Data. Neural Networks 24(8): 906-916. Elsevier."*
- class [Fast\\_TopoART\\_base](#)  
*Base class providing functionality common to several [TopoART](#) networks.*
- class [Fast\\_TopoART\\_C](#)  
*Class [Fast\\_TopoART\\_C](#) provides an implementation of the TopoART-C neural network as proposed in "Marko Tscherepanow and Sören Riechers (2012). An Incremental On-line Classifier for Imbalanced, Incomplete, and Noisy Data. In Proceedings of the European Conference on Artificial Intelligence (ECAI), Workshop on Active and Incremental Learning (AIL), pp. 18-23. Montpellier, France."*
- class [Fast\\_TopoART\\_R](#)  
*Class [Fast\\_TopoART\\_R](#) provides an implementation of the TopoART-R neural network as proposed in "Marko Tscherepanow (2011). An Extended [TopoART](#) Network for the Stable On-Line Learning of Regression Functions. In Proceedings of the International Conference on Neural Information Processing (ICONIP), LNCS 7063, pp. 562-571. Berlin, Germany: Springer."*
- class [Hypersphere\\_TopoART](#)



Class [Hypersphere\\_TopoART](#) provides an implementation of the Hypersphere [TopoART](#) neural network as proposed in "Marko Tscherepanow (2012). Incremental On-line Clustering with a Topology-Learning Hierarchical ART Neural Network Using Hyperspherical Categories. In Poster and Industry Proceedings of the Industrial Conference on Data Mining (ICDM), pp. 22–34. Fockendorf, Germany: ibai-publishing."

- class [Hypersphere\\_TopoART\\_C](#)

Class [Hypersphere\\_TopoART\\_C](#) provides an implementation of the Hypersphere TopoART-C neural network. Hypersphere TopoART-C is a combination of Hypersphere [TopoART](#) as proposed in "Marko Tscherepanow (2012). Incremental On-line Clustering with a Topology-Learning Hierarchical ART Neural Network Using Hyperspherical Categories. In Poster and Industry Proceedings of the Industrial Conference on Data Mining (ICDM), pp. 22–34. Fockendorf, Germany: ibai-publishing." and TopoART-C as proposed in "Marko Tscherepanow and Sören Riechers (2012). An Incremental On-line Classifier for Imbalanced, Incomplete, and Noisy Data. In Proceedings of the European Conference on Artificial Intelligence (ECAI), Workshop on Active and Incremental Learning (AIL), pp. 18-23. Montpellier, France."

- interface [IAdaptationStateCheck](#)

Interface enabling checks whether certain adaptations of a network occurred.

- interface [IAssociative\\_recall](#)

Interface summarising the associative recall functionality using stimulus elements and recall result elements of type *decimal*.

- interface [IByteAccess\\_Associative\\_recall](#)

Interface providing access to the basic associative recall functionality using stimulus elements and recall result elements of type *byte*. Individual values may use the complete value range from 0 to 255. They are internally converted from/to the [TopoART](#) input interval [0,1].

- interface [IByteAccess\\_Episodic\\_recall](#)

Interface providing access to the basic episodic recall functionality using stimulus elements and recall result elements of type *byte*. Individual values may use the complete value range from 0 to 255. They are internally converted from/to the [TopoART](#) input interval [0,1].

- interface [IByteAccess\\_TopoART](#)

Interface providing access to the basic [TopoART](#) functionality using input elements of type *byte*. Individual values may use the complete value range from 0 to 255. They are internally converted to the [TopoART](#) input interval [0,1].

- interface [IByteAccess\\_TopoART\\_AM](#)

Interface providing access to the basic TopoART-AM functionality using input elements of type *byte*. Individual values may use the complete value range from 0 to 255. They are internally converted to the [TopoART](#) input interval [0,1].

- interface [IByteAccess\\_TopoART\\_C](#)

Interface providing access to the basic TopoART-C functionality using input elements of type *byte*. Individual values may use the complete value range from 0 to 255. They are internally converted to the [TopoART](#) input interval [0,1].

- interface [IByteAccess\\_TopoART\\_R](#)

Interface providing access to the basic TopoART-R functionality using input elements of type *byte*. Individual values may use the complete value range from 0 to 255. They are internally converted to the [TopoART](#) input interval [0,1].

- interface [ICategoryAccess](#)

Interface providing access to the learnt categories, e.g for drawing.

- interface [IDecimalAccess\\_Associative\\_recall](#)

Interface providing access to the basic associative recall functionality using stimulus elements and recall result elements of type *decimal*. Individual stimulus values must lie within the [TopoART](#) input interval [0,1].

- interface [IDecimalAccess\\_Episodic\\_recall](#)

Interface providing access to the basic episodic recall functionality using stimulus elements and recall result elements of type *decimal*. Individual stimulus values must lie within the [TopoART](#) input interval [0,1].

- interface [IDecimalAccess\\_TopoART](#)

Interface providing access to the basic [TopoART](#) functionality using input elements of type *decimal*. Individual values must lie within the [TopoART](#) input interval [0,1].

- interface [IDecimalAccess\\_TopoART\\_AM](#)

Interface providing access to the basic TopoART-AM functionality using input elements of type *decimal*. Individual values must lie within the [TopoART](#) input interval [0,1].

- interface [IDecimalAccess\\_TopoART\\_C](#)

Interface providing access to the basic TopoART-C functionality using input elements of type *decimal*. Individual values must lie within the [TopoART](#) input interval [0,1].



- interface [IDecimalAccess\\_TopoART\\_R](#)  
*Interface providing access to the basic TopoART-R functionality using input elements of type `decimal`. Individual values must lie within the `TopoART` input interval `[0,1]`.*
- interface [IEndRecall](#)  
*Interface summarising the type-independent functionality to stop the recall process.*
- interface [IEpisodic\\_recall](#)  
*Interface summarising the episodic recall functionality using stimulus elements and recall result elements of type `decimal`.*
- interface [IEpisodic\\_TopoART](#)  
*Interface summarising the Episodic `TopoART` functionality including learning, prediction, episodic recall using input elements, stimulus elements, and recall result elements of type `decimal` as well as adaptation state control.*
- interface [IFast\\_Associative\\_recall](#)  
*Interface summarising the associative recall functionality using stimulus elements and recall result elements of type `byte` or of type `decimal`.*
- interface [IFast\\_Episodic\\_recall](#)  
*Interface summarising the episodic recall functionality using stimulus elements and recall result elements of type `byte` or of type `decimal`.*
- interface [IFast\\_Episodic\\_TopoART](#)  
*Interface summarising the Episodic `TopoART` functionality including learning, prediction, episodic recall using input elements, stimulus elements, and recall result elements of type `byte` or of type `decimal` as well as adaptation state control.*
- interface [IFast\\_TopoART](#)  
*Interface summarising the `TopoART` functionality including learning and prediction using input elements of type `byte` or of type `decimal` as well as adaptation state control.*
- interface [IFast\\_TopoART\\_AM](#)  
*Interface summarising the Episodic `TopoART` functionality including learning, prediction, episodic recall using input elements, stimulus elements, and recall result elements of type `byte` or of type `decimal` as well as adaptation state control.*
- interface [IFast\\_TopoART\\_C](#)  
*Interface summarising the TopoART-C functionality including learning and prediction using input elements of type `byte` or of type `decimal` as well as adaptation state control.*
- interface [IFast\\_TopoART\\_R](#)  
*Interface summarising the TopoART-R functionality including learning and prediction using input elements and output elements of type `byte` or of type `decimal` as well as adaptation state control.*
- class [InvalidClassIDException](#)  
*Exception signalling an invalid class ID.*
- class [InvalidFileException](#)  
*Exception signalling an invalid file.*
- class [InvalidModuleIndexException](#)  
*Exception signalling an invalid module index.*
- class [InvalidNumberException](#)  
*Exception signalling an invalid number.*
- class [InvalidSizeException](#)  
*Exception signalling an invalid size.*
- class [InvalidStateException](#)  
*Exception signalling an invalid state of the neural network.*
- interface [ITopoART](#)  
*Interface summarising the `TopoART` functionality including learning and prediction using input elements of type `decimal` as well as adaptation state control.*
- interface [ITopoART\\_AM](#)  
*Interface summarising the TopoART-AM functionality including learning, prediction, associative recall using input elements, stimulus elements, and recall result elements of type `decimal` as well as adaptation state control.*
- interface [ITopoART\\_AM\\_base](#)

- Interface summarising the basic TopoART-AM functionality excluding learning and prediction.*

  - interface `ITopoART_base`
- Interface summarising the basic TopoART functionality excluding learning and prediction.*

  - interface `ITopoART_C`
- Interface summarising the TopoART-C functionality including learning and prediction using input elements of type `decimal` as well as adaptation state control.*

  - interface `ITopoART_R`
- Interface summarising the TopoART-R functionality including learning and prediction using input elements and output elements of type `decimal` as well as adaptation state control.*

  - interface `ITopoART_R_base`
- Interface summarising the basic TopoART-R functionality excluding learning and prediction.*

  - struct `LibTopoART_control`
- Struct `LibTopoART_control` provides fields to control the general behaviour of `LibTopoART`.*

  - struct `LibTopoART_info`
- Struct `LibTopoART_info` provides some metainformation regarding the respective implementation of `LibTopoART`.*

  - class `TopoART`
- Class `TopoART` provides an implementation of the TopoART neural network as proposed in "Marko Tscherepanow (2010). TopoART: A topology learning hierarchical ART network. In Proceedings of the International Conference on Artificial Neural Networks (ICANN), LNCS 6354, pp. 157–167. Berlin, Germany: Springer."*

  - class `TopoART_C`
- Class `TopoART_C` provides an implementation of the TopoART-C neural network as proposed in "Marko Tscherepanow and Sören Riechers (2012). An Incremental On-line Classifier for Imbalanced, Incomplete, and Noisy Data. In Proceedings of the European Conference on Artificial Intelligence (ECAI), Workshop on Active and Incremental Learning (AIL), pp. 18-23. Montpellier, France."*

  - struct `TopoART_C_Prediction`
- Struct `TopoART_C_Prediction` contains a prediction made by a TopoART-C network.*

  - class `TopoART_R`
- Class `TopoART_R` provides an implementation of the TopoART-R neural network as proposed in "Marko Tscherepanow (2011). An Extended TopoART Network for the Stable On-Line Learning of Regression Functions. In Proceedings of the International Conference on Neural Information Processing (ICONIP), LNCS 7063, pp. 562–571. Berlin, Germany: Springer."*

  - struct `TopoART_R_Prediction`
- Struct `TopoART_R_Prediction` contains a prediction made by a TopoART-R network.*

## Enumerations

- enum `AdaptationState` {  
`AdaptationState.NO_ADAPTATION` = 0, `AdaptationState.ADDED_NODE_CANDIDATE` = 0x0001,  
`AdaptationState.ADAPTED_NONPERMANENT_WEIGHT` = 0x0002, `AdaptationState.ADDED_EDGE_CANDIDATE`  
= 0x0004,  
`AdaptationState.REMOVED_NODE_CANDIDATE` = 0x0008, `AdaptationState.REMOVED_EDGE_CANDIDATE`  
= 0x0010, `AdaptationState.ANY_NONPERMANENT_ADAPTATION_MASK` = 0x00ff, `AdaptationState.ADDED_PERMANENT`  
= 0x0100,  
`AdaptationState.ADAPTED_PERMANENT_WEIGHT` = 0x0200, `AdaptationState.ADDED_PERMANENT_EDGE`  
= 0x0400, `AdaptationState.ANY_PERMANENT_ADAPTATION_MASK` = 0xff00 }  
*Enumeration specifying possible adaptation states.*
- enum `VerbosityLevel` : uint { `VerbosityLevel.Important`, `VerbosityLevel.Normal`, `VerbosityLevel.Verbose` }  
*Enumeration specifying possible adaptation states.*

### 4.1.1 Enumeration Type Documentation

**4.1.1.1 AdaptationState** enum `LibTopoART.AdaptationState` [strong]

Enumeration specifying possible adaptation states.

## Enumerator

NO_ADAPTATION	No adaptation occurred.
ADDED_NODE_CANDIDATE	Added one or more node candidates.
ADAPTED_NONPERMANENT_WEIGHT	The change of at least a single weight of one node candidate exceeds the given threshold.
ADDED_EDGE_CANDIDATE	Added an edge from/to a node candidate.
REMOVED_NODE_CANDIDATE	Removed one or more node candidates.
REMOVED_EDGE_CANDIDATE	Removed one or more node candidates.
ANY_NONPERMANENT_ADAPTATION_MASK	Mask for all non-permanent adaptations.
ADDED_PERMANENT_NODE	Added one or more permanent nodes.
ADAPTED_PERMANENT_WEIGHT	The change of at least a single weight of one permanent node exceeds the given threshold.
ADDED_PERMANENT_EDGE	Added an edge between two permanent nodes.
ANY_PERMANENT_ADAPTATION_MASK	Mask for all permanent adaptations.

#### 4.1.1.2 VerbosityLevel `enum LibTopoART.VerbosityLevel : uint [strong]`

Enumeration specifying possible adaptation states.

## Enumerator

Important	Enables only the most important messages.
Normal	Enables the standard messages.
Verbose	Enables all messages.

## 4.2 LibTopoART\_samples Namespace Reference

## Classes

- class [Episodic\\_TopoART\\_sample1](#)  
*Episodic clustering sample using synthetic two-dimensional data. [C#]*
- class [Episodic\\_TopoART\\_sample2](#)  
*Episodic clustering sample using real-world video data. [C#]*
- class [TopoART\\_AM\\_sample1](#)  
*Sample using TopoART-AM with synthetic two-dimensional data. [C#]*
- class [TopoART\\_AM\\_sample2](#)  
*Learning of bidirectional associations between images. [F#]*
- class [TopoART\\_C\\_sample1](#)  
*Simple classification sample. [C#]*
- class [TopoART\\_C\\_sample2](#)  
*Classification sample using more complex synthetic two-dimensional data with associated class IDs. [C#]*
- class [TopoART\\_R\\_sample1](#)  
*Regression sample using TopoART-R. (simplified version) [C#]*
- class [TopoART\\_R\\_sample2](#)

- Regression sample using TopoART-R. (advanced version) [C#]*
- class [TopoART\\_R\\_sample3](#)
- Pixel-wise regression analysis of an image using TopoART-R. [F#]*
- class [TopoART\\_sample1](#)
- Simple clustering sample. [C#]*
- class [TopoART\\_sample2](#)
- Clustering sample using more complex synthetic two-dimensional data. [C#]*
- class [TopoART\\_sample3](#)
- Clustering sample using very noisy synthetic two-dimensional data. [VB]*

## 5 Class Documentation

### 5.1 LibTopoART.CategoryInfo Struct Reference

Struct [CategoryInfo](#) summarises information about a node's category.

#### Public Member Functions

- [CategoryInfo](#) (decimal[] [spatial\\_weights](#), decimal[]? [temporal\\_weights](#), long [cluster\\_ID](#), long [class\\_ID](#))  
*This constructor sets the instance variables [spatial\\_weights](#), [temporal\\_weights](#), [cluster\\_ID](#), and [class\\_ID](#) of struct [CategoryInfo](#).*

#### Public Attributes

- readonly decimal[] [spatial\\_weights](#)  
*Instance variable [spatial\\_weights](#) represents the spatial weights of the considered node.*
- readonly? decimal[] [temporal\\_weights](#)  
*Instance variable [temporal\\_weights](#) represents the temporal weights of the considered node if it support temporal learning.*
- readonly long [cluster\\_ID](#)  
*Instance variable [cluster\\_ID](#) represents the cluster ID of the considered node.*
- readonly long [class\\_ID](#)  
*Instance variable [class\\_ID](#) represents the class ID of the considered node. If class IDs are not supported by the respective node, this value is set to [LibTopoART\\_info.UNDEFINED](#).*

#### 5.1.1 Detailed Description

Struct [CategoryInfo](#) summarises information about a node's category.

#### 5.1.2 Constructor & Destructor Documentation

**5.1.2.1 CategoryInfo()** `LibTopoART.CategoryInfo.CategoryInfo ( decimal[] spatial\_weights, decimal?[] temporal\_weights, long cluster\_ID, long class\_ID )`

This constructor sets the instance variables [spatial\\_weights](#), [temporal\\_weights](#), [cluster\\_ID](#), and [class\\_ID](#) of struct [CategoryInfo](#).

## Parameters

<i>spatial_weights</i>	The spatial weights to be set.
<i>temporal_weights</i>	The temporal weights to be set.
<i>cluster_ID</i>	The cluster ID to be set.
<i>class_ID</i>	The class ID to be set.

## 5.1.3 Member Data Documentation

5.1.3.1 **class\_ID** `readonly long LibTopoART.CategoryInfo.class_ID`

Instance variable `class_ID` represents the class ID of the considered node. If class IDs are not supported by the respective node, this value is set to `LibTopoART_info.UNDEFINED`.

5.1.3.2 **cluster\_ID** `readonly long LibTopoART.CategoryInfo.cluster_ID`

Instance variable `cluster_ID` represents the cluster ID of the considered node.

5.1.3.3 **spatial\_weights** `readonly decimal [] LibTopoART.CategoryInfo.spatial_weights`

Instance variable `spatial_weights` represents the spatial weights of the considered node.

5.1.3.4 **temporal\_weights** `readonly? decimal [] LibTopoART.CategoryInfo.temporal_weights`

Instance variable `temporal_weights` represents the temporal weights of the considered node if it support temporal learning.

## 5.2 LibTopoART\_samples.Episodic\_TopoART\_sample1 Class Reference

Episodic clustering sample using synthetic two-dimensional data. [C#]

## 5.2.1 Detailed Description

Episodic clustering sample using synthetic two-dimensional data. [C#]

Like in Section 4.1 of "Marko Tscherepanow, Sina Kühnel, and Sören Riechers (2012). Episodic Clustering of Data Streams Using a Topology-Learning Neural Network. In Proceedings of the European Conference on Artificial Intelligence (ECAI), Workshop on Active and Incremental Learning (AIL), pp. 24-29. Montpellier, France.", an Episodic TopoART network is trained with the well-known Two Spirals dataset. Due to the incorporation of temporal information during learning, Episodic TopoART is capable of creating two clusters each representing one spiral in an unsupervised way. These clusters are formed by the nodes of module b (ETA b).

The resulting network can be visualised using the R script `ShowEpisodicTopoARTResults.R` provided in the subfolder R.

## 5.3 LibTopoART\_samples.Episodic\_TopoART\_sample2 Class Reference

Episodic clustering sample using real-world video data. [C#]

### 5.3.1 Detailed Description

Episodic clustering sample using real-world video data. [C#]

Like in Section 4.2 of "Marko Tscherepanow, Sina Kühnel, and Sören Riechers (2012). Episodic Clustering of Data Streams Using a Topology-Learning Neural Network. In Proceedings of the European Conference on Artificial Intelligence (ECAI), Workshop on Active and Incremental Learning (AIL), pp. 24-29. Montpellier, France.", an Episodic TopoART network is trained with real-world video data. Each image has a size of 64x36 pixels. As each pixel comprises 3 color channels (RGB), the input length equals 6912. After finishing training, recall is performed for a single input stimulus.

The recall results can be visualised using the R script `ShowEpisodicTopoARTRecallResults.R` provided in the subfolder R.

## 5.4 LibTopoART.F2\_output Class Reference

Class `F2_output` provides the output of a single `TopoART` module. It is a compressed version of the output vectors `y` and `c`.

### Public Member Functions

- `F2_output()`

*This constructor sets all instance variables of class `F2_output` to `LibTopoART_info.UNDEFINED`.*

### Public Attributes

- decimal `bm_node_activation`  
*Instance variable `bm_node_activation` represents the activation of the best-matching node (prediction variant).*
- long `bm_node_ID`  
*Instance variable `bm_node_ID` represents the ID of the best-matching node.*
- long `bm_cluster_ID`  
*Instance variable `bm_cluster_ID` represents the cluster ID of the best-matching node.*
- decimal `bm_permanent_node_activation`  
*Instance variable `bm_permanent_node_activation` represents the activation of the best-matching permanent node (prediction variant).*
- long `bm_permanent_node_ID`  
*Instance variable `bm_permanent_node_ID` represents the ID of the best-matching permanent node.*
- long `bm_permanent_cluster_ID`  
*Instance variable `bm_permanent_cluster_ID` represents the cluster ID of the best-matching permanent node.*

### 5.4.1 Detailed Description

Class `F2_output` provides the output of a single `TopoART` module. It is a compressed version of the output vectors `y` and `c`.

## 5.4.2 Constructor & Destructor Documentation

### 5.4.2.1 **F2\_output()** `LibTopoART.F2_output.F2_output ( )`

This constructor sets all instance variables of class `F2_output` to `LibTopoART_info.UNDEFINED`.

## 5.4.3 Member Data Documentation

### 5.4.3.1 **bm\_cluster\_ID** `long LibTopoART.F2_output.bm_cluster_ID`

Instance variable `bm_cluster_ID` represents the cluster ID of the best-matching node.

### 5.4.3.2 **bm\_node\_activation** `decimal LibTopoART.F2_output.bm_node_activation`

Instance variable `bm_node_activation` represents the activation of the best-matching node (prediction variant).

### 5.4.3.3 **bm\_node\_ID** `long LibTopoART.F2_output.bm_node_ID`

Instance variable `bm_node_ID` represents the ID of the best-matching node.

### 5.4.3.4 **bm\_permanent\_cluster\_ID** `long LibTopoART.F2_output.bm_permanent_cluster_ID`

Instance variable `bm_permanent_cluster_ID` represents the cluster ID of the best-matching permanent node.

### 5.4.3.5 **bm\_permanent\_node\_activation** `decimal LibTopoART.F2_output.bm_permanent_node_activation`

Instance variable `bm_permanent_node_activation` represents the activation of the best-matching permanent node (prediction variant).

### 5.4.3.6 **bm\_permanent\_node\_ID** `long LibTopoART.F2_output.bm_permanent_node_ID`

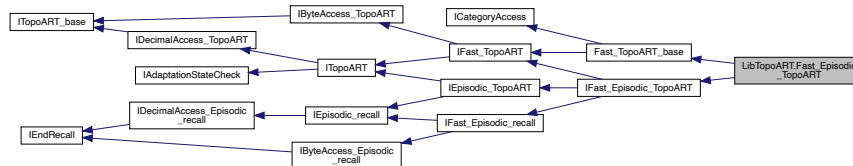
Instance variable `bm_permanent_node_ID` represents the ID of the best-matching permanent node.



## 5.5 LibTopoART.Fast\_Episodic\_TopoART Class Reference

Class [Fast\\_Episodic\\_TopoART](#) provides an implementation of the Episodic [TopoART](#) neural network as proposed in "Marko Tscherepanow, Sina Kühnel, and Sören Riechers (2012). Episodic Clustering of Data Streams Using a Topology-Learning Neural Network. In Proceedings of the European Conference on Artificial Intelligence (ECAI), Workshop on Active and Incremental Learning (AIL), pp. 24-29. Montpellier, France."

Inheritance diagram for LibTopoART.Fast\_Episodic\_TopoART:



### Public Member Functions

- [Fast\\_Episodic\\_TopoART](#) (long input\_length, long module\_number, decimal rho\_a, long t\_max)  
*This constructor initialises an Episodic [TopoART](#) network.*
- [Fast\\_Episodic\\_TopoART](#) (string path)  
*This constructor loads a saved Episodic [TopoART](#) network.*
- override void [Learn](#) (byte[] input)  
*This method performs a single training step.*
- override void [Learn](#) (decimal[] input)  
*This method performs a single training step.*
- long [BeginRecall](#) (byte[] stimulus)  
*This method starts the recall process.*
- long [BeginRecall](#) (decimal[] stimulus)  
*This method starts the recall process.*
- bool [InterEpisodeRecallStep](#) (out byte[]? recall\_result, out decimal F3\_activation)  
*This method performs a single inter-episode recall step and sets the starting point for intra-episode recall.*
- bool [InterEpisodeRecallStep](#) (out decimal[]? recall\_result, out decimal F3\_activation)  
*This method performs a single inter-episode recall step and sets the starting point for intra-episode recall.*
- bool [IntraEpisodeRecallStep](#) (out byte[]? recall\_result)  
*This method performs a single intra-episode recall step.*
- bool [IntraEpisodeRecallStep](#) (out decimal[]? recall\_result)  
*This method performs a single intra-episode recall step.*
- void [EndRecall](#) ()  
*This method stops the recall process and frees temporary resources.*

### Properties

- new decimal [FileFormatVersion](#) [get]  
*Property `FileFormatVersion` returns the version of the file format used by class `Episodic_TopoART`.*
- long [T\\_max](#) [get]  
*Property `T_max` represents the maximum considered time frame.*

## Additional Inherited Members

### 5.5.1 Detailed Description

Class `Fast_Episodic_TopoART` provides an implementation of the Episodic `TopoART` neural network as proposed in "Marko Tscherepanow, Sina Kühnel, and Sören Riechers (2012). Episodic Clustering of Data Streams Using a Topology-Learning Neural Network. In Proceedings of the European Conference on Artificial Intelligence (ECAI), Workshop on Active and Incremental Learning (AIL), pp. 24-29. Montpellier, France."

### 5.5.2 Constructor & Destructor Documentation

**5.5.2.1 Fast\_Episodic\_TopoART() [1/2]** `LibTopoART.Fast_Episodic_TopoART.Fast_Episodic_TopoART (`  
`long input_length,`  
`long module_number,`  
`decimal rho_a,`  
`long t_max )`

This constructor initialises an Episodic `TopoART` network.

#### Parameters

<i>input_length</i>	The length of input vectors to be learnt.
<i>module_number</i>	The number of Episodic <code>TopoART</code> modules.
<i>rho_a</i>	The vigilance parameter of the first Episodic <code>TopoART</code> module (ETA a).
<i>t_max</i>	The parameter limiting the considered time frame.

**5.5.2.2 Fast\_Episodic\_TopoART() [2/2]** `LibTopoART.Fast_Episodic_TopoART.Fast_Episodic_TopoART (`  
`string path )`

This constructor loads a saved Episodic `TopoART` network.

#### Parameters

<i>path</i>	The path of a binary Episodic <code>TopoART</code> file.
-------------	----------------------------------------------------------

#### Exceptions

<code>InvalidFileException</code>	Throws when the given file cannot be loaded.
-----------------------------------	----------------------------------------------

### 5.5.3 Member Function Documentation

**5.5.3.1 BeginRecall()** [1/2] `long LibTopoART.Fast_Episodic_TopoART.BeginRecall ( byte[] stimulus )`

This method starts the recall process.

#### Parameters

<i>stimulus</i>	The stimulus (input) which is used to trigger recall. The stimulus elements are internally scaled from [0,255] to [0,1].
-----------------	--------------------------------------------------------------------------------------------------------------------------

#### Returns

The number of F3 nodes created.

Implements [LibTopoART.IByteAccess\\_Episodic\\_recall](#).

**5.5.3.2 BeginRecall()** [2/2] `long LibTopoART.Fast_Episodic_TopoART.BeginRecall ( decimal[] stimulus )`

This method starts the recall process.

#### Parameters

<i>stimulus</i>	The stimulus (input) which is used to trigger recall.
-----------------	-------------------------------------------------------

#### Returns

The number of F3 nodes created.

Implements [LibTopoART.IDecimalAccess\\_Episodic\\_recall](#).

**5.5.3.3 EndRecall()** `void LibTopoART.Fast_Episodic_TopoART.EndRecall ( )`

This method stops the recall process and frees temporary resources.

Implements [LibTopoART.IEndRecall](#).

**5.5.3.4 InterEpisodeRecallStep()** [1/2] `bool LibTopoART.Fast_Episodic_TopoART.InterEpisodeRecallStep ( out byte?[] recall_result, out decimal F3_activation )`

This method performs a single inter-episode recall step and sets the starting point for intra-episode recall.

**Parameters**

<i>recall_result</i>	Returns the recall output vector for the current step. The elements of the recall result are internally scaled from [0,1] to [0,255].
<i>F3_activation</i>	Returns the activation of the current F3 node.

**Returns**

A boolean result indicating whether the recall step was successfully completed, or not.

Implements [LibTopoART.IByteAccess\\_Episodic\\_recall](#).

```
5.5.3.5 InterEpisodeRecallStep() [2/2]  bool LibTopoART.Fast_Episodic_TopoART.InterEpisode←
RecallStep (
    out decimal?[] recall_result,
    out decimal F3_activation )
```

This method performs a single inter-episode recall step and sets the starting point for intra-episode recall.

**Parameters**

<i>recall_result</i>	Returns the recall output vector for the current step.
<i>F3_activation</i>	Returns the activation of the current F3 node.

**Returns**

A boolean result indicating whether the recall step was successfully completed, or not.

Implements [LibTopoART.IDecimalAccess\\_Episodic\\_recall](#).

```
5.5.3.6 IntraEpisodeRecallStep() [1/2]  bool LibTopoART.Fast_Episodic_TopoART.IntraEpisode←
RecallStep (
    out byte?[] recall_result )
```

This method performs a single intra-episode recall step.

**Parameters**

<i>recall_result</i>	Returns the recall output vector for the current step. The elements of the recall result are internally scaled from [0,1] to [0,255].
----------------------	---------------------------------------------------------------------------------------------------------------------------------------

**Returns**

A boolean result indicating whether the recall step was successfully completed or not.

Implements [LibTopoART.IByteAccess\\_Episodic\\_recall](#).

**5.5.3.7 IntraEpisodeRecallStep()** [2/2] `bool LibTopoART.Fast_Episodic_TopoART.IntraEpisodeRecallStep ( out decimal?[] recall_result )`

This method performs a single intra-episode recall step.

#### Parameters

<i>recall_result</i>	Returns the recall output vector for the current step.
----------------------	--------------------------------------------------------

#### Returns

A boolean result indicating whether the recall step was successfully completed, or not.

Implements [LibTopoART.IDecimalAccess\\_Episodic\\_recall](#).

**5.5.3.8 Learn()** [1/2] `override void LibTopoART.Fast_Episodic_TopoART.Learn ( byte[] input ) [virtual]`

This method performs a single training step.

The spatial weights are adapted as in the original [TopoART](#) network. In contrast, the adaptation of the temporal weight  $w_{\{j,2\}^{F2,t}}$  occurring only in Episodic [TopoART](#) is slightly different↵:  
 $w_{\{j,2\}^{F2,t}}(t+1) = \text{beta\_j} * \text{Max}(t\_2^{F1}(t), w_{\{j,2\}^{F2,t}}(t) + (1 - \text{beta\_j}) * w_{\{j,2\}^{F2,t}}(t)$  for  $j = \text{bm}$  or  $j = \text{sbm}$ . (Note:  $w_{\{j,1\}^{F2,t}}$  remains constant over the life time of a node.)

#### Parameters

<i>input</i>	The input vector to be learnt. The input values are internally scaled from [0,255] to [0,1].
--------------	----------------------------------------------------------------------------------------------

Implements [LibTopoART.Fast\\_TopoART\\_base](#).

**5.5.3.9 Learn()** [2/2] `override void LibTopoART.Fast_Episodic_TopoART.Learn ( decimal[] input ) [virtual]`

This method performs a single training step.

The spatial weights are adapted as in the original [TopoART](#) network. In contrast, the adaptation of the temporal weight  $w_{\{j,2\}^{F2,t}}$  occurring only in Episodic [TopoART](#) is slightly different↵:  
 $w_{\{j,2\}^{F2,t}}(t+1) = \text{beta\_j} * \text{Max}(t\_2^{F1}(t), w_{\{j,2\}^{F2,t}}(t) + (1 - \text{beta\_j}) * w_{\{j,2\}^{F2,t}}(t)$  for  $j = \text{bm}$  or  $j = \text{sbm}$ . (Note:  $w_{\{j,1\}^{F2,t}}$  remains constant over the life time of a node.)

## Parameters

<i>input</i>	The input vector to be learnt.
--------------	--------------------------------

Implements [LibTopoART.Fast\\_TopoART\\_base](#).

## 5.5.4 Property Documentation

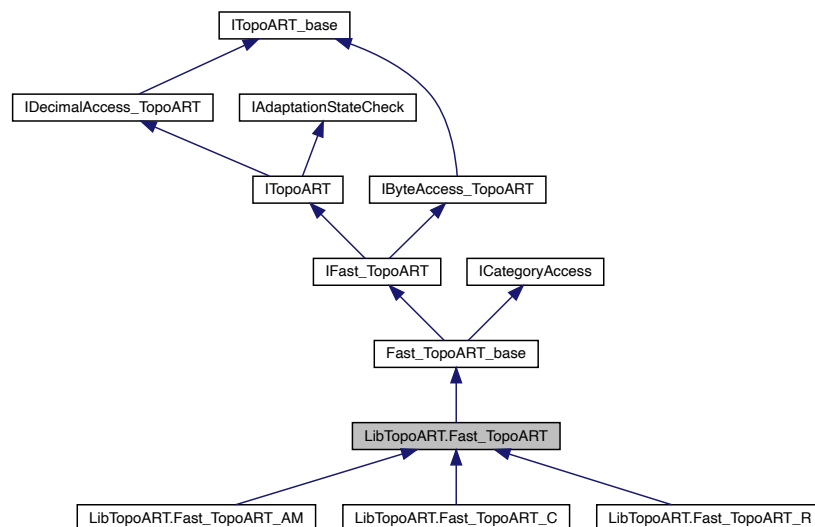
**5.5.4.1 FileFormatVersion** `new decimal LibTopoART.Fast_Episodic_TopoART.FileFormatVersion [get]`

Property `FileFormatVersion` returns the version of the file format used by class `Episodic_TopoART`.

## 5.6 LibTopoART.Fast\_TopoART Class Reference

Class [Fast\\_TopoART](#) provides an implementation of the [TopoART](#) neural network as proposed in "Marko Tscherepanow (2010). [TopoART](#): A topology learning hierarchical ART network. In Proceedings of the International Conference on Artificial Neural Networks (ICANN), LNCS 6354, pp. 157–167. Berlin, Germany: Springer."

Inheritance diagram for `LibTopoART.Fast_TopoART`:



## Public Member Functions

- [Fast\\_TopoART](#) (long input\_length, long module\_number, decimal rho\_a)  
*This constructor initialises a [TopoART](#) network.*
- [Fast\\_TopoART](#) (string path)  
*This constructor loads a saved [TopoART](#) network.*
- override void [Learn](#) (byte[] input)  
*This method performs a single training step.*
- override void [Learn](#) (decimal[] input)  
*This method performs a single training step.*

## Additional Inherited Members

### 5.6.1 Detailed Description

Class `Fast_TopoART` provides an implementation of the `TopoART` neural network as proposed in "Marko Tscherepanow (2010). `TopoART`: A topology learning hierarchical ART network. In Proceedings of the International Conference on Artificial Neural Networks (ICANN), LNCS 6354, pp. 157–167. Berlin, Germany: Springer."

Internally, real-valued data are mapped to `int` variables. Therefore, computations are accelerated but less accurate. As a consequence, the results may differ slightly from class `TopoART`.

Class `Fast_TopoART` requires all input to lie in the interval  $[0,1]$ .

### 5.6.2 Constructor & Destructor Documentation

**5.6.2.1 `Fast_TopoART()` [1/2]** `LibTopoART.Fast_TopoART.Fast_TopoART (`  
`long input_length,`  
`long module_number,`  
`decimal rho_a )`

This constructor initialises a `TopoART` network.

#### Parameters

<i>input_length</i>	The length of input vectors to be learnt.
<i>module_number</i>	The number of <code>TopoART</code> modules.
<i>rho_a</i>	The vigilance parameter of the first <code>TopoART</code> module (TA a).

**5.6.2.2 `Fast_TopoART()` [2/2]** `LibTopoART.Fast_TopoART.Fast_TopoART (`  
`string path )`

This constructor loads a saved `TopoART` network.

#### Parameters

<i>path</i>	The path of a binary <code>TopoART</code> file.
-------------	-------------------------------------------------

#### Exceptions

<code>InvalidFileException</code>	Throws when the given file cannot be loaded.
-----------------------------------	----------------------------------------------

This method performs a single training step.

<i>input</i>	The input vector to be learnt. The input values are internally scaled from [0,255] to [0,1].
--------------	----------------------------------------------------------------------------------------------

Implements `LibTopoART.Fast` TopoART base.

Reimplemented in [LibTopoART.Fast TopoART R](#), and [LibTopoART.Fast TopoART C](#).

This method performs a single training step.

<i>input</i>	The input vector to be learnt.
--------------	--------------------------------

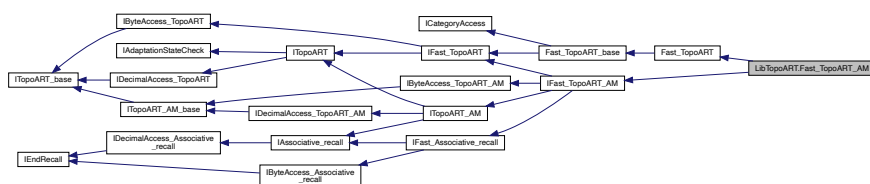
Implements `LibTopoART.Fast_TopoART_base`.

Reimplemented in `LibTopoART.Fast_TopoART_R`, and `LibTopoART.Fast_TopoART_C`.

## 5.7 LibTopoART.Fast\_TopoART\_AM Class Reference

Class `Fast_TopoART_AM` provides an implementation of the TopoART-AM neural network as proposed in "Marko Tscherepanow, Marco Kortkamp and Marc Kammer (2011). A Hierarchical ART Network for the Stable Incremental Learning of Topological Structures and Associations from Noisy Data. Neural Networks 24(8): 906-916. Elsevier."

Inheritance diagram for LibTopoART.Fast\_TopoART\_AM:





## Public Member Functions

- [Fast\\_TopoART\\_AM](#) (long key\_1\_length, long key\_2\_length, long module\_number, decimal rho\_a)  
*This constructor initialises a TopoART-AM network.*
- [Fast\\_TopoART\\_AM](#) (string path)  
*This constructor loads a saved TopoART-AM network.*
- [F2\\_output\[\] GetBMOutput](#) (byte[] key\_1\_vec, byte[] key\_2\_vec)  
*This method finds the closest category for a given pair of keys.*
- [F2\\_output\[\] GetBMOutput](#) (decimal[] key\_1\_vec, decimal[] key\_2\_vec)  
*This method finds the closest category for a given pair of keys.*
- void [Learn](#) (byte[] key\_1\_vec, byte[] key\_2\_vec)  
*This method performs a single training step.*
- void [Learn](#) (decimal[] key\_1\_vec, decimal[] key\_2\_vec)  
*This method performs a single training step.*
- long [BeginRecallKey1](#) (byte[] key\_2\_vec, long module\_index=FINAL\_MODULE)  
*This method starts the recall process for the first key vector.*
- long [BeginRecallKey1](#) (decimal[] key\_2\_vec, long module\_index=FINAL\_MODULE)  
*This method starts the recall process for the first key vector.*
- long [BeginRecallKey2](#) (byte[] key\_1\_vec, long module\_index=FINAL\_MODULE)  
*This method starts the recall process for the second key vector.*
- long [BeginRecallKey2](#) (decimal[] key\_1\_vec, long module\_index=FINAL\_MODULE)  
*This method starts the recall process for the second key vector.*
- bool [RecallStep](#) (out byte[]? recall\_result, out decimal F3\_activation)  
*This method performs a single associative recall step.*
- bool [RecallStep](#) (out decimal[]? recall\_result, out decimal F3\_activation)  
*This method performs a single associative recall step.*
- void [EndRecall](#) ()  
*This method stops the recall process and frees temporary resources.*

## Properties

- new decimal [FileFormatVersion](#) [get]  
*Property FileFormatVersion returns the version of the file format used by class TopoART\_AM.*
- long [Key\\_1\\_len](#) [get]  
*Property Key\_1\_len returns the length of the first key vector.*
- long [Key\\_2\\_len](#) [get]  
*Property Key\_2\_len returns the length of the second key vector.*

## Additional Inherited Members

### 5.7.1 Detailed Description

Class [Fast\\_TopoART\\_AM](#) provides an implementation of the TopoART-AM neural network as proposed in "Marko Tscherepanow, Marco Kortkamp and Marc Kammer (2011). A Hierarchical ART Network for the Stable Incremental Learning of Topological Structures and Associations from Noisy Data. Neural Networks 24(8): 906-916. Elsevier."

Class [TopoART\\_AM](#) requires all input and output to lie in the interval [0,1].

## 5.7.2 Constructor & Destructor Documentation

**5.7.2.1 Fast\_TopoART\_AM() [1/2]** `LibTopoART.Fast_TopoART_AM.Fast_TopoART_AM (`  
    `long key_1_length,`  
    `long key_2_length,`  
    `long module_number,`  
    `decimal rho_a )`

This constructor initialises a TopoART-AM network.

### Parameters

<i>key_1_length</i>	The length of the first key vector to be learnt.
<i>key_2_length</i>	The length of the second key vector to be learnt.
<i>module_number</i>	The number of TopoART-AM modules.
<i>rho_a</i>	The vigilance parameter of the first TopoART-AM module (TopoART-AM a).

**5.7.2.2 Fast\_TopoART\_AM() [2/2]** `LibTopoART.Fast_TopoART_AM.Fast_TopoART_AM (`  
    `string path )`

This constructor loads a saved TopoART-AM network.

### Parameters

<i>path</i>	The path of a binary TopoART-AM file.
-------------	---------------------------------------

### Exceptions

<a href="#"><i>InvalidFileException</i></a>	Throws when the given file cannot be loaded.
---------------------------------------------	----------------------------------------------

## 5.7.3 Member Function Documentation

**5.7.3.1 BeginRecallKey1() [1/2]** `long LibTopoART.Fast_TopoART_AM.BeginRecallKey1 (`  
    `byte[] key_2_vec,`  
    `long module_index = FINAL\_MODULE )`

This method starts the recall process for the first key vector.

## Parameters

<i>key_2_vec</i>	The stimulus (second key vector) which is used to trigger recall. The stimulus elements are internally scaled from [0,255] to [0,1].
<i>module_index</i>	Index of the TopoART-AM module to be used for recall. ( <code>FINAL_MODULE</code> denotes the module with the highest index.)

## Returns

The number of F3 nodes created.

## Exceptions

<a href="#"><i>InvalidModuleIndexException</i></a>	Throws when <i>module_index</i> is invalid.
----------------------------------------------------	---------------------------------------------

Implements [LibTopoART.IByteAccess\\_Associative\\_recall](#).

**5.7.3.2 BeginRecallKey1() [2/2]** `long LibTopoART.Fast_TopoART_AM.BeginRecallKey1 ( decimal[] key_2_vec, long module_index = FINAL_MODULE )`

This method starts the recall process for the first key vector.

## Parameters

<i>key_2_vec</i>	The stimulus (second key vector) which is used to trigger recall.
<i>module_index</i>	Index of the TopoART-AM module to be used for recall. ( <code>FINAL_MODULE</code> denotes the module with the highest index.)

## Returns

The number of F3 nodes created.

## Exceptions

<a href="#"><i>InvalidModuleIndexException</i></a>	Throws when <i>module_index</i> is invalid.
----------------------------------------------------	---------------------------------------------

Implements [LibTopoART.IDecimalAccess\\_Associative\\_recall](#).

**5.7.3.3 BeginRecallKey2() [1/2]** `long LibTopoART.Fast_TopoART_AM.BeginRecallKey2 ( byte[] key_1_vec, long module_index = FINAL_MODULE )`

This method starts the recall process for the second key vector.

**Parameters**

<i>key_1_vec</i>	The stimulus (first key vector) which is used to trigger recall. The stimulus elements are internally scaled from [0,255] to [0,1].
<i>module_index</i>	Index of the TopoART-AM module to be used for recall. ( <code>FINAL_MODULE</code> denotes the module with the highest index.)

**Returns**

The number of F3 nodes created.

**Exceptions**

<a href="#"><i>InvalidModuleIndexException</i></a>	Throws when <i>module_index</i> is invalid.
----------------------------------------------------	---------------------------------------------

Implements [LibTopoART.IByteAccess\\_Associative\\_recall](#).

```
5.7.3.4 BeginRecallKey2() [2/2] long LibTopoART.Fast_TopoART_AM.BeginRecallKey2 (
    decimal[] key_1_vec,
    long module_index = FINAL_MODULE )
```

This method starts the recall process for the second key vector.

**Parameters**

<i>key_1_vec</i>	The stimulus (first key vector) which is used to trigger recall.
<i>module_index</i>	Index of the TopoART-AM module to be used for recall. ( <code>FINAL_MODULE</code> denotes the module with the highest index.)

**Returns**

The number of F3 nodes created.

**Exceptions**

<a href="#"><i>InvalidModuleIndexException</i></a>	Throws when <i>module_index</i> is invalid.
----------------------------------------------------	---------------------------------------------

Implements [LibTopoART.IDecimalAccess\\_Associative\\_recall](#).

```
5.7.3.5 EndRecall() void LibTopoART.Fast_TopoART_AM.EndRecall ( )
```

This method stops the recall process and frees temporary resources.

Implements [LibTopoART.IEndRecall](#).

**5.7.3.6 GetBMOutput()** [1/2] `F2_output [ ] LibTopoART.Fast_TopoART_AM.GetBMOutput ( byte[] key_1_vec, byte[] key_2_vec )`

This method finds the closest category for a given pair of keys.

#### Parameters

<i>key_1_vec</i>	The first key vector. The elements of the key vector are internally scaled from [0,255] to [0,1].
<i>key_2_vec</i>	The second key vector corresponding to <i>key_1_vec</i> . The elements of the key vector are internally scaled from [0,255] to [0,1].

#### Returns

An array of type `F2_output`. Each entry contains the ID of the best-matching node and the corresponding cluster ID for one TopoART-AM module.

Implements [LibTopoART.IByteAccess\\_TopoART\\_AM](#).

**5.7.3.7 GetBMOutput()** [2/2] `F2_output [ ] LibTopoART.Fast_TopoART_AM.GetBMOutput ( decimal[] key_1_vec, decimal[] key_2_vec )`

This method finds the closest category for a given pair of keys.

#### Parameters

<i>key_1_vec</i>	The first key vector.
<i>key_2_vec</i>	The second key vector corresponding to <i>key_1_vec</i> .

#### Returns

An array of type `F2_output`. Each entry contains the ID of the best-matching node and the corresponding cluster ID for one TopoART-AM module.

Implements [LibTopoART.IDecimalAccess\\_TopoART\\_AM](#).

**5.7.3.8 Learn()** [1/2] `void LibTopoART.Fast_TopoART_AM.Learn ( byte[] key_1_vec, byte[] key_2_vec )`

This method performs a single training step.

#### Parameters

<i>key_1_vec</i>	The first key vector to be learnt.
<i>key_2_vec</i>	The second key vector corresponding to <i>key_1_vec</i> .

Implements [LibTopoART.IByteAccess\\_TopoART\\_AM](#).

**5.7.3.9 Learn()** [2/2] `void LibTopoART.Fast_TopoART_AM.Learn (`  
`decimal[] key_1_vec,`  
`decimal[] key_2_vec )`

This method performs a single training step.

#### Parameters

<i>key_1_vec</i>	The first key vector to be learnt. The elements of the key vector are internally scaled from [0,255] to [0,1].
<i>key_2_vec</i>	The second key vector corresponding to <i>key_1_vec</i> . The elements of the key vector are internally scaled from [0,255] to [0,1].

Implements [LibTopoART.IDecimalAccess\\_TopoART\\_AM](#).

**5.7.3.10 RecallStep()** [1/2] `bool LibTopoART.Fast_TopoART_AM.RecallStep (`  
`out byte?[] recall_result,`  
`out decimal F3_activation )`

This method performs a single associative recall step.

#### Parameters

<i>recall_result</i>	Returns the recall output vector for the current step. The elements of the recall result are internally scaled from [0,1] to [0,255].
<i>F3_activation</i>	Returns the activation of the current F3 node.

#### Returns

A boolean result indicating whether the recall step was successfully completed or not.

Implements [LibTopoART.IByteAccess\\_Associative\\_recall](#).

**5.7.3.11 RecallStep()** [2/2] `bool LibTopoART.Fast_TopoART_AM.RecallStep (`  
`out decimal?[] recall_result,`  
`out decimal F3_activation )`

This method performs a single associative recall step.

#### Parameters

<i>recall_result</i>	Returns the recall output vector for the current step.
<i>F3_activation</i>	Returns the activation of the current F3 node.

**Returns**

A boolean result indicating whether the recall step was successfully completed or not.

Implements [LibTopoART.IDecimalAccess\\_Associative\\_recall](#).

**5.7.4 Property Documentation**

**5.7.4.1 FileFormatVersion** new decimal LibTopoART.Fast\_TopoART\_AM.FileFormatVersion [get]

Property FileFormatVersion returns the version of the file format used by class TopoART\_AM.

**5.7.4.2 Key\_1\_len** long LibTopoART.Fast\_TopoART\_AM.Key\_1\_len [get]

Property Key\_1\_len returns the length of the first key vector.

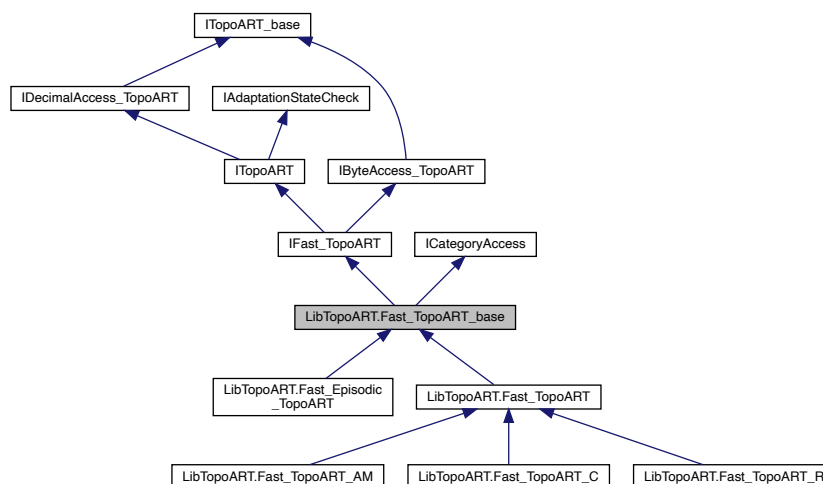
**5.7.4.3 Key\_2\_len** long LibTopoART.Fast\_TopoART\_AM.Key\_2\_len [get]

Property Key\_2\_len returns the length of the second key vector.

**5.8 LibTopoART.Fast\_TopoART\_base Class Reference**

Base class providing functionality common to several [TopoART](#) networks.

Inheritance diagram for LibTopoART.Fast\_TopoART\_base:



## Public Member Functions

- abstract void [Learn](#) (byte[] input)  
*This method performs a single training step.*
- abstract void [Learn](#) (decimal[] input)  
*This method performs a single training step.*
- void [Dispose](#) ()  
*Releases all resources used by the [LibTopoART.Fast\\_TopoART\\_base](#) object.*
- void [ComputeClusterIDs](#) ()  
*This method computes the cluster IDs for all neurons.*
- [F2\\_output\[\] GetBMOutput](#) (byte[] input)  
*This method finds the closest category for a given test input.*
- [F2\\_output\[\] GetBMOutput](#) (byte[] input, bool[]? mask\_vector)  
*This method finds the closest category for a given test input.*
- [F2\\_output\[\] GetBMOutput](#) (decimal[] input)  
*This method finds the closest category for a given test input.*
- [F2\\_output\[\] GetBMOutput](#) (decimal[] input, bool[]? mask\_vector)  
*This method finds the closest category for a given test input.*
- void [SaveText](#) (string path)  
*This method saves the entire network as a text file.*
- void [Save](#) (string path, CompressionLevel compression=CompressionLevel.Fastest)  
*This method saves the entire network as a binary file.*
- void [Save](#) (string path, bool compatibility\_mode, CompressionLevel compression=CompressionLevel.Fastest)  
*This method saves the entire network as a binary file.*
- void [ResetAdaptationState](#) ()  
*This method resets the adaptation state to `AdaptationState.NO_ADAPTATION`.*
- [AdaptationState GetAdaptationState](#) (decimal epsilon=0.001m)  
*This method returns the current adaptation state.*
- List< [CategoryInfo](#) >? [GetCategories](#) (long module\_index=FINAL\_MODULE)  
*This method collects information on the categories of a specified module.*

## Static Public Attributes

- const long [FINAL\\_MODULE](#) = LibTopoART\_info.FINAL\_MODULE  
*Instance variable `FINAL_MODULE` gives the value used for indicating that the [TopoART](#) module with the highest index is to be used.*

## Properties

- decimal [Alpha](#) [get, set]  
*Property `Alpha` represents the choice parameter alpha.*
- decimal [Beta\\_sbm](#) [get, set]  
*Property `Beta_sbm` represents the learning rate of the second best-matching nodes.*
- long[] [ClusterNum](#) [get]  
*Property `ClusterNum` represents the number of [TopoART](#) clusters found by each module.*
- long [InputLen](#) [get]  
*Property `InputLen` returns the length of the input vector.*
- long [LearningSteps](#) [get]  
*Property `LearningSteps` represents the total number of performed learning steps.*



- long [ModuleNum](#) [get]  
*Property `ModuleNum` represents the number of [TopoART](#) modules used. (The original [TopoART](#) uses two modules.)*
- long[] [NodeNum](#) [get]  
*Property `NodeNum` represents the number of [TopoART](#) nodes used by each module.*
- decimal [Rho\\_a](#) [get]  
*Property `Rho_a` represents the vigilance parameter of the first [TopoART](#) module (TA a).*
- long [Tau](#) [get, set]  
*Property `Tau` represents the parameter tau required for the removal of nodes and edges.*
- long [Phi](#) [get, set]
- long[] [Phis](#) [get, set]
- string [IntegerBaseType](#) = `Common.types[(int)integer_base_type_index]` [get]  
*Property `IntegerBaseType` returns a string containing the data type used for representing integer variables (IDs, parameters, counters, etc.) internally.*
- decimal [FileFormatVersion](#) [get]  
*Property `FileFormatVersion` returns the version of the file format used by class [Fast\\_TopoART\\_base](#).*
- string [FloatBaseType](#) = `Common.types[(int)float_base_type_index]` [get]  
*Property `FloatBaseType` returns a string containing the data type used for representing floating point variables (input, weights, etc.) internally.*
- decimal [TopoARTFileFormatVersion](#) [get]  
*Property `TopoARTFileFormatVersion` returns the version of the file format used by class [Fast\\_TopoART\\_base](#).*

### 5.8.1 Detailed Description

Base class providing functionality common to several [TopoART](#) networks.

### 5.8.2 Member Function Documentation

#### 5.8.2.1 ComputeClusterIDs() `void LibTopoART.Fast_TopoART_base.ComputeClusterIDs ( )`

This method computes the cluster IDs for all neurons.

Implements [LibTopoART.ITopoART\\_base](#).

#### 5.8.2.2 Dispose() `void LibTopoART.Fast_TopoART_base.Dispose ( )`

Releases all resources used by the [LibTopoART.Fast\\_TopoART\\_base](#) object.

Call [Dispose\(\)](#) when you are finished using the [LibTopoART.Fast\\_TopoART\\_base](#). The [Dispose\(\)](#) method leaves the [LibTopoART.Fast\\_TopoART\\_base](#) in an unusable state. After calling [Dispose\(\)](#), you must release all references to the [LibTopoART.Fast\\_TopoART\\_base](#) so the garbage collector can reclaim the memory that the [LibTopoART.Fast\\_TopoART\\_base](#) was occupying.

#### 5.8.2.3 GetAdaptationState() `AdaptationState LibTopoART.Fast_TopoART_base.GetAdaptationState ( decimal epsilon = 0.001m )`

This method returns the current adaptation state.

**Parameters**

<i>epsilon</i>	The threshold for weight adaptations to be considered.
----------------	--------------------------------------------------------

**Returns**

An enumeration describing the adaptation state.

**Exceptions**

<a href="#"><i>InvalidStateException</i></a>	Throws when the network is in an invalid state.
<a href="#"><i>InvalidNumberException</i></a>	Throws when the number of edges of an F2 node is greater than <code>int.MaxValue</code> .

Implements [LibTopoART.IAdaptationStateCheck](#).

**5.8.2.4 GetBMOutput()** [1/4] `F2_output [ ] LibTopoART.Fast_TopoART_base.GetBMOutput ( byte[] input )`

This method finds the closest category for a given test input.

**Parameters**

<i>input</i>	The input vector $x(t)$ . The input values are internally scaled from [0,255] to [0,1].
--------------	-----------------------------------------------------------------------------------------

**Returns**

An array of type `F2_output`. Each entry contains the ID of the best-matching node and the corresponding cluster ID for one [TopoART](#) module.

Implements [LibTopoART.IByteAccess\\_TopoART](#).

**5.8.2.5 GetBMOutput()** [2/4] `F2_output [ ] LibTopoART.Fast_TopoART_base.GetBMOutput ( byte[] input, bool?[] mask_vector )`

This method finds the closest category for a given test input.

**Parameters**

<i>input</i>	The input vector $x(t)$ . The input values are internally scaled from [0,255] to [0,1].
<i>mask_vector</i>	A mask vector excluding individual dimensions of $x(t)$ from the computation. (Setting an element of the mask vector to <code>true</code> , excludes the corresponding elements of $x(t)$ .)

**Returns**

An array of type [F2\\_output](#). Each entry contains the ID of the best-matching node and the corresponding cluster ID for one [TopoART](#) module.

**5.8.2.6 GetBMOutput()** [3/4] [F2\\_output](#) [ ] LibTopoART.Fast\_TopoART\_base.GetBMOutput (   
 decimal[ ] *input* )

This method finds the closest category for a given test input.

**Parameters**

<i>input</i>	The input vector x(t).
--------------	------------------------

**Returns**

An array of type [F2\\_output](#). Each entry contains the ID of the best-matching node and the corresponding cluster ID for one [TopoART](#) module.

Implements [LibTopoART.IDecimalAccess\\_TopoART](#).

**5.8.2.7 GetBMOutput()** [4/4] [F2\\_output](#) [ ] LibTopoART.Fast\_TopoART\_base.GetBMOutput (   
 decimal[ ] *input*,   
 bool? [ ] *mask\_vector* )

This method finds the closest category for a given test input.

**Parameters**

<i>input</i>	The input vector x(t).
<i>mask_vector</i>	A mask vector excluding individual dimensions of x(t) from the computation. (Setting an element of the mask vector to <code>true</code> , excludes the corresponding elements of x(t).)

**Returns**

An array of type [F2\\_output](#). Each entry contains the ID of the best-matching node and the corresponding cluster ID for one [TopoART](#) module.

**5.8.2.8 GetCategories()** List<[CategoryInfo](#)>? LibTopoART.Fast\_TopoART\_base.GetCategories (   
 long *module\_index* = [FINAL\\_MODULE](#) )

This method collects information on the categories of a specified module.

**Parameters**

<i>module_index</i>	The index of the module the categories of which are to be analysed.
---------------------	---------------------------------------------------------------------

**Returns**

A list containing information about the respective categories.

**Exceptions**

<a href="#"><i>InvalidModuleIndexException</i></a>	Throws when <i>module_index</i> is invalid.
<a href="#"><i>InvalidNumberException</i></a>	Throws when the number of nodes of a module is greater than <code>int.MaxValue</code> .

Implements [LibTopoART.ICategoryAccess](#).

**5.8.2.9 Learn() [1/2]** `abstract void LibTopoART.Fast_TopoART_base.Learn ( byte[] input ) [pure virtual]`

This method performs a single training step.

**Parameters**

<i>input</i>	The input vector to be learnt. The input values are internally scaled from [0,255] to [0,1].
--------------	----------------------------------------------------------------------------------------------

Implements [LibTopoART.IByteAccess\\_TopoART](#).

Implemented in [LibTopoART.Fast\\_TopoART\\_R](#), [LibTopoART.Fast\\_TopoART\\_C](#), [LibTopoART.Fast\\_TopoART](#), and [LibTopoART.Fast\\_Episodic\\_TopoART](#).

**5.8.2.10 Learn() [2/2]** `abstract void LibTopoART.Fast_TopoART_base.Learn ( decimal[] input ) [pure virtual]`

This method performs a single training step.

**Parameters**

<i>input</i>	The input vector to be learnt.
--------------	--------------------------------

Implements [LibTopoART.IDecimalAccess\\_TopoART](#).

Implemented in [LibTopoART.Fast\\_TopoART\\_R](#), [LibTopoART.Fast\\_TopoART\\_C](#), [LibTopoART.Fast\\_TopoART](#), and [LibTopoART.Fast\\_Episodic\\_TopoART](#).

**5.8.2.11 ResetAdaptationState()** `void LibTopoART.Fast_TopoART_base.ResetAdaptationState ( )`

This method resets the adaptation state to `AdaptationState.NO_ADAPTATION`.

**Exceptions**

<a href="#"><i>InvalidNumberException</i></a>	Throws when the number of edges of an F2 node is greater than <code>int.MaxValue</code> .
-----------------------------------------------	-------------------------------------------------------------------------------------------

Implements [LibTopoART.IAdaptationStateCheck](#).

**5.8.2.12 Save() [1/2]** `void LibTopoART.Fast_TopoART_base.Save (`  
`string path,`  
`bool compatibility_mode,`  
`CompressionLevel compression = CompressionLevel.Fastest )`

This method saves the entire network as a binary file.

**Parameters**

<i>path</i>	A <code>string</code> representing the path of the file to save.
<i>compatibility_mode</i>	If true, the file is saved in compatibility mode.
<i>compression</i>	Compression level of the save file (Compression is not supported by <a href="#">LibTopoART v0.93</a> and below.)

**5.8.2.13 Save() [2/2]** `void LibTopoART.Fast_TopoART_base.Save (`  
`string path,`  
`CompressionLevel compression = CompressionLevel.Fastest )`

This method saves the entire network as a binary file.

**Parameters**

<i>path</i>	A <code>string</code> representing the path of the file to save.
<i>compression</i>	Compression level of the save file (Compression is not supported by <a href="#">LibTopoART v0.93</a> and below.)

Implements [LibTopoART.ITopoART\\_base](#).

**5.8.2.14 SaveText()** `void LibTopoART.Fast_TopoART_base.SaveText (`  
`string path )`

This method saves the entire network as a text file.

## Parameters

<i>path</i>	A string representing the path of the file to save.
-------------	-----------------------------------------------------

Implements [LibTopoART.ITopoART\\_base](#).

### 5.8.3 Member Data Documentation

**5.8.3.1 FINAL\_MODULE** `const long LibTopoART.Fast_TopoART_base.FINAL_MODULE = LibTopoART_↵  
info.FINAL_MODULE [static]`

Instance variable `FINAL_MODULE` gives the value used for indicating that the [TopoART](#) module with the highest index is to be used.

### 5.8.4 Property Documentation

**5.8.4.1 FileFormatVersion** `decimal LibTopoART.Fast_TopoART_base.FileFormatVersion [get]`

Property `FileFormatVersion` returns the version of the file format used by class [Fast\\_TopoART\\_base](#).

**5.8.4.2 FloatBaseType** `string LibTopoART.Fast_TopoART_base.FloatBaseType = Common.types[(int)float↵  
_base_type_index] [get]`

Property `FloatBaseType` returns a string containing the data type used for representing floating point variables (input, weights, etc.) internally.

**5.8.4.3 IntegerBaseType** `string LibTopoART.Fast_TopoART_base.IntegerBaseType = Common.types[(int)integer↵  
_base_type_index] [get]`

Property `IntegerBaseType` returns a string containing the data type used for representing integer variables (IDs, parameters, counters, etc.) internally.

**5.8.4.4 Phi** `long LibTopoART.Fast_TopoART_base.Phi [get], [set]`

Property `Phi` represents the parameter `phi` required for the removal of nodes and edges as well as for the propagation of input to subsequent [TopoART](#) modules.

**5.8.4.5 Phis** `long [] LibTopoART.Fast_TopoART_base.Phis [get], [set]`

Property `Phis` constitutes an extension of property `Phi` that enables individual values of `phi` for each module. By this, the removal of nodes and edges as well as for the propagation of input to subsequent [TopoART](#) modules can be controlled in a task dependent manner.

## Exceptions

<a href="#">InvalidSizeException</a>	Throws when the array length does not fit the module number.
--------------------------------------	--------------------------------------------------------------

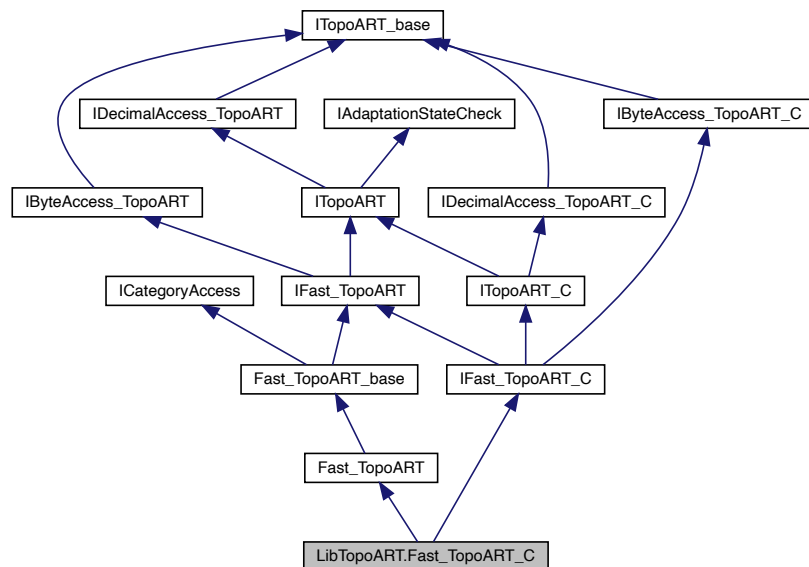
**5.8.4.6 TopoARTFileFormatVersion** decimal LibTopoART.Fast\_TopoART\_base.TopoARTFileFormat↔  
Version [get]

Property `TopoARTFileFormatVersion` returns the version of the file format used by class `Fast_TopoART_base`.

## 5.9 LibTopoART.Fast\_TopoART\_C Class Reference

Class `Fast_TopoART_C` provides an implementation of the TopoART-C neural network as proposed in "Marko Tscherepanow and Sören Riechers (2012). An Incremental On-line Classifier for Imbalanced, Incomplete, and Noisy Data. In Proceedings of the European Conference on Artificial Intelligence (ECAI), Workshop on Active and Incremental Learning (AIL), pp. 18-23. Montpellier, France."

Inheritance diagram for LibTopoART.Fast\_TopoART\_C:



### Public Member Functions

- `Fast_TopoART_C` (long input\_length, long module\_number, decimal rho\_a)  
*This constructor initialises a TopoART-C network.*
- `Fast_TopoART_C` (string path)  
*This constructor loads a saved TopoART-C network.*
- override void `Learn` (byte[] input)

- This method performs a single training step and sets the class ID corresponding to input to `UNDEFINED_CLASS_ID`.*

  - override void **Learn** (decimal[] input)

*This method performs a single training step and sets the class ID corresponding to input to `UNDEFINED_CLASS_ID`.*
- void **Learn** (byte[] input, long class\_ID)
- This method performs a single training step.*
- void **Learn** (decimal[] input, long class\_ID)
- This method performs a single training step.*
- long **Predict** (byte[] input, long nu)
- This method predicts the class ID.*
- long **Predict** (decimal[] input, long nu)
- This method predicts the class ID.*
- **TopoART\_C\_Prediction Predict** (byte[] input, bool[]? mask\_vector, long nu)
- This method predicts the class ID.*
- **TopoART\_C\_Prediction Predict** (decimal[] input, bool[]? mask\_vector, long nu)
- This method predicts the class ID.*

### Static Public Attributes

- const long **UNDEFINED\_CLASS\_ID** = -2
- Instance variable `UNDEFINED_CLASS_ID` gives the value used for indicating that an input sample was predict to belong to the undefined class; i.e, no class ID was provided for such input samples during training.*

### Properties

- new decimal **FileFormatVersion** [get]
- Property `FileFormatVersion` returns the version of the file format used by class `Fast_TopoART_C`.*

## 5.9.1 Detailed Description

Class `Fast_TopoART_C` provides an implementation of the TopoART-C neural network as proposed in "Marko Tscherepanow and Sören Riechers (2012). An Incremental On-line Classifier for Imbalanced, Incomplete, and Noisy Data. In Proceedings of the European Conference on Artificial Intelligence (ECAI), Workshop on Active and Incremental Learning (AIL), pp. 18-23. Montpellier, France."

Internally, real-valued data are mapped to `int` variables. Therefore, computations are accelerated but less accurate. As a consequence, the results may differ slightly from class `TopoART_C`.

Class `Fast_TopoART_C` requires all input except the class IDs to lie in the interval [0,1]. The class IDs are signed integer values.

## 5.9.2 Constructor & Destructor Documentation

### 5.9.2.1 **Fast\_TopoART\_C()** [1/2]

```
LibTopoART.Fast_TopoART_C.Fast_TopoART_C (
    long input_length,
    long module_number,
    decimal rho_a )
```

This constructor initialises a TopoART-C network.



## Parameters

<i>input_length</i>	The length of input vectors to be learnt.
<i>module_number</i>	The number of TopoART-C modules.
<i>rho_a</i>	The vigilance parameter of the first TopoART-C module (TopoART-C a).

### 5.9.2.2 Fast\_TopoART\_C() [2/2] `LibTopoART.Fast_TopoART_C.Fast_TopoART_C ( string path )`

This constructor loads a saved TopoART-C network.

## Parameters

<i>path</i>	The path of a binary TopoART-C file.
-------------	--------------------------------------

## Exceptions

<a href="#"><i>InvalidFileException</i></a>	Throws when the given file cannot be loaded.
---------------------------------------------	----------------------------------------------

## 5.9.3 Member Function Documentation

### 5.9.3.1 Learn() [1/4] `override void LibTopoART.Fast_TopoART_C.Learn ( byte[] input ) [virtual]`

This method performs a single training step and sets the class ID corresponding to *input* to `UNDEFINED_CLASS_ID`.

## Parameters

<i>input</i>	The input vector to be learnt. The input values are internally scaled from [0,255] to [0,1].
--------------	----------------------------------------------------------------------------------------------

Reimplemented from [LibTopoART.Fast\\_TopoART](#).

### 5.9.3.2 Learn() [2/4] `void LibTopoART.Fast_TopoART_C.Learn ( byte[] input, long class_ID )`

This method performs a single training step.

## Parameters

<i>input</i>	The input vector to be learnt. The elements of the input vector are internally scaled from [0,255] to [0,1].
<i>class_ID</i>	The class ID corresponding to <i>input</i> . (must be equal to or larger than 0)

## Exceptions

<a href="#"><i>InvalidClassIDException</i></a>	Throws when <i>class_ID</i> is less than 0.
------------------------------------------------	---------------------------------------------

Implements [LibTopoART.IByteAccess\\_TopoART\\_C](#).

**5.9.3.3 Learn() [3/4]** `override void LibTopoART.Fast_TopoART_C.Learn ( decimal[] input ) [virtual]`

This method performs a single training step and sets the class ID corresponding to *input* to `UNDEFINED_CLASS_ID`.

## Parameters

<i>input</i>	The input vector to be learnt.
--------------	--------------------------------

Reimplemented from [LibTopoART.Fast\\_TopoART](#).

**5.9.3.4 Learn() [4/4]** `void LibTopoART.Fast_TopoART_C.Learn ( decimal[] input, long class_ID )`

This method performs a single training step.

## Parameters

<i>input</i>	The input vector to be learnt.
<i>class_ID</i>	The class ID corresponding to <i>input</i> . (must be equal to or larger than 0)

## Exceptions

<a href="#"><i>InvalidClassIDException</i></a>	Throws when <i>class_ID</i> is less than 0.
------------------------------------------------	---------------------------------------------

Implements [LibTopoART.IDecimalAccess\\_TopoART\\_C](#).

**5.9.3.5 Predict()** [1/4] [TopoART\\_C\\_Prediction](#) LibTopoART.Fast\_TopoART\_C.Predict (   
     byte[] *input*,  
     bool?[] *mask\_vector*,  
     long *nu* )

This method predicts the class ID.

#### Parameters

<i>input</i>	The input vector the class ID of which is to be predicted. The elements of the input vector are internally scaled from [0,255] to [0,1].
<i>mask_vector</i>	The mask vector corresponding to <i>input</i> .
<i>nu</i>	The maximum cardinality of the set of enclosing categories E and the neighbourhood set N. (This parameter does not modify the network. It may be arbitrarily changed in each prediction step.)

#### Returns

An object of type [TopoART\\_C\\_Prediction](#) containing the predicted class ID and a corresponding confidence value.

**5.9.3.6 Predict()** [2/4] long LibTopoART.Fast\_TopoART\_C.Predict (   
     byte[] *input*,  
     long *nu* )

This method predicts the class ID.

#### Parameters

<i>input</i>	The input vector the class ID of which is to be predicted. The elements of the input vector are internally scaled from [0,255] to [0,1].
<i>nu</i>	The maximum cardinality of the set of enclosing categories E and the neighbourhood set N. (This parameter does not modify the network. It may be arbitrarily changed in each prediction step.)

#### Returns

The predicted class ID.

Implements [LibTopoART.IByteAccess\\_TopoART\\_C](#).

**5.9.3.7 Predict()** [3/4] [TopoART\\_C\\_Prediction](#) LibTopoART.Fast\_TopoART\_C.Predict (   
     decimal[] *input*,  
     bool?[] *mask\_vector*,  
     long *nu* )

This method predicts the class ID.

**Parameters**

<i>input</i>	The input vector the class ID of which is to be predicted.
<i>mask_vector</i>	The mask vector corresponding to <i>input</i> .
<i>nu</i>	The maximum cardinality of the set of enclosing categories E and the neighbourhood set N. (This parameter does not modify the network. It may be arbitrarily changed in each prediction step.)

**Returns**

An object of type [TopoART\\_C\\_Prediction](#) containing the predicted class ID and a corresponding confidence value.

**5.9.3.8 Predict()** [4/4] `long LibTopoART.Fast_TopoART_C.Predict (`  
`decimal[] input,`  
`long nu )`

This method predicts the class ID.

**Parameters**

<i>input</i>	The input vector the class ID of which is to be predicted.
<i>nu</i>	The maximum cardinality of the set of enclosing categories E and the neighbourhood set N. (This parameter does not modify the network. It may be arbitrarily changed in each prediction step.)

**Returns**

The predicted class ID.

Implements [LibTopoART.IDecimalAccess\\_TopoART\\_C](#).

**5.9.4 Member Data Documentation**

**5.9.4.1 UNDEFINED\_CLASS\_ID** `const long LibTopoART.Fast_TopoART_C.UNDEFINED_CLASS_ID = -2`  
`[static]`

Instance variable `UNDEFINED_CLASS_ID` gives the value used for indicating that an input sample was predict to belong to the undefined class; i.e, no class ID was provided for such input samples during training.

**5.9.5 Property Documentation**

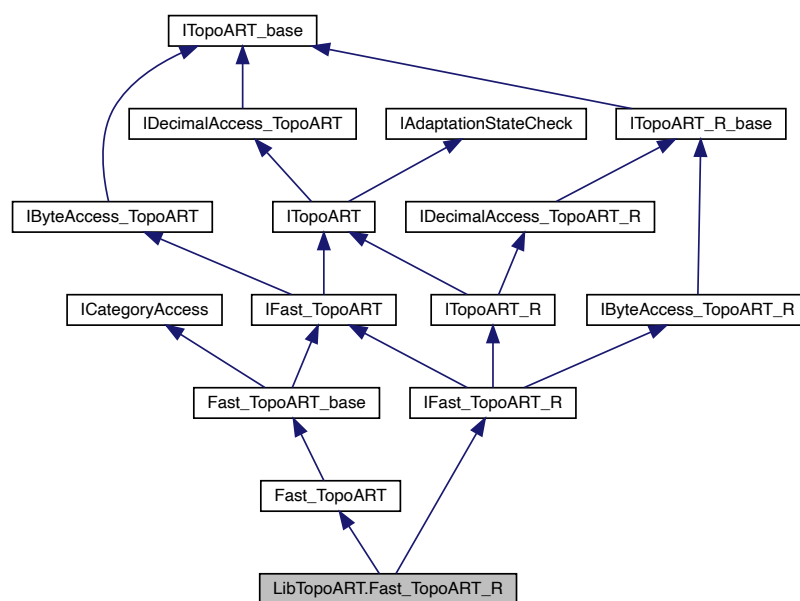
#### 5.9.5.1 FileFormatVersion `new decimal LibTopoART.Fast_TopoART_C.FileFormatVersion [get]`

Property `FileFormatVersion` returns the version of the file format used by class `Fast_TopoART_C`.

## 5.10 LibTopoART.Fast\_TopoART\_R Class Reference

Class `Fast_TopoART_R` provides an implementation of the TopoART-R neural network as proposed in "Marko Tscherepanow (2011). An Extended TopoART Network for the Stable On-Line Learning of Regression Functions. In Proceedings of the International Conference on Neural Information Processing (ICONIP), LNCS 7063, pp. 562–571. Berlin, Germany: Springer."

Inheritance diagram for `LibTopoART.Fast_TopoART_R`:



### Public Member Functions

- `Fast_TopoART_R` (long i\_length, long d\_length, long module\_number, decimal rho\_a)  
*This constructor initialises a TopoART-R network.*
- `Fast_TopoART_R` (string path)  
*This constructor loads a saved TopoART-R network.*
- override void `Learn` (byte[] input)  
*This method performs a single training step. The independent variables and the dependent variables are automatically separated.*
- override void `Learn` (decimal[] input)  
*This method performs a single training step. The independent variables and the dependent variables are automatically separated.*
- void `Learn` (byte[] i\_vec, byte[] d\_vec)  
*This method performs a single training step.*
- void `Learn` (decimal[] i\_vec, decimal[] d\_vec)  
*This method performs a single training step.*

- `byte[] Predict (byte[] i_vec, long nu=10)`  
*This method predicts the dependent variables.*
- `decimal[] Predict (decimal[] i_vec, long nu=10)`  
*This method predicts the dependent variables.*
- `TopoART_R_Prediction< byte > Predict (byte[] i_vec, bool[] m_i_vec, long nu=10)`  
*This method predicts the dependent variables for a given set of independent variables. Unknown values of independent variables can be signified by setting the corresponding value of `m_i_vec` to `true`.*
- `TopoART_R_Prediction< decimal > Predict (decimal[] i_vec, bool[] m_i_vec, long nu=10)`  
*This method predicts the dependent variables for a given set of independent variables. Unknown values of independent variables can be signified by setting the corresponding value of `m_i_vec` to `true`.*

## Properties

- long `D_len` [get]  
*Property `D_len` returns the length of the output vector (dependent variables).*
- new decimal `FileFormatVersion` [get]  
*Property `FileFormatVersion` returns the version of the file format used by class `Fast_TopoART_R`.*
- long `I_len` [get]  
*Property `I_len` returns the length of the input vector (independent variables).*

## Additional Inherited Members

### 5.10.1 Detailed Description

Class `Fast_TopoART_R` provides an implementation of the TopoART-R neural network as proposed in "Marko Tscherepanow (2011). An Extended TopoART Network for the Stable On-Line Learning of Regression Functions. In Proceedings of the International Conference on Neural Information Processing (ICONIP), LNCS 7063, pp. 562–571. Berlin, Germany: Springer."

Internally, real-valued data are mapped to `int` variables. Therefore, computations are accelerated but less accurate. As a consequence, the results may differ slightly from class `TopoART_R`.

Class `Fast_TopoART_R` requires all input and output to lie in the interval `[0,1]`.

### 5.10.2 Constructor & Destructor Documentation

**5.10.2.1 Fast\_TopoART\_R() [1/2]** `LibTopoART.Fast_TopoART_R.Fast_TopoART_R (`  
`long i_length,`  
`long d_length,`  
`long module_number,`  
`decimal rho_a )`

This constructor initialises a TopoART-R network.

#### Parameters

<code>i_length</code>	The length of the input vector (independent variables) to be learnt.
<code>d_length</code>	The length of the output vector (dependent variables) to be learnt.
<code>module_number</code>	The number of TopoART-R modules.
<code>rho_a</code>	The vigilance parameter of the first TopoART-R module (TopoART-R a).

**5.10.2.2 Fast\_TopoART\_R()** [2/2] `LibTopoART.Fast_TopoART_R.Fast_TopoART_R ( string path )`

This constructor loads a saved TopoART-R network.

#### Parameters

<i>path</i>	The path of a binary TopoART-R file.
-------------	--------------------------------------

#### Exceptions

<a href="#">InvalidFileException</a>	Throws when the given file cannot be loaded.
--------------------------------------	----------------------------------------------

### 5.10.3 Member Function Documentation

**5.10.3.1 Learn()** [1/4] `void LibTopoART.Fast_TopoART_R.Learn ( byte[] i_vec, byte[] d_vec )`

This method performs a single training step.

#### Parameters

<i>i_vec</i>	The input vector (independent variables) to be learnt. The elements of the input vector are internally scaled from [0,255] to [0,1].
<i>d_vec</i>	The output vector (dependent variables) corresponding to <i>i_vec</i> . The elements of the output vector are internally scaled from [0,255] to [0,1].

Implements [LibTopoART.IByteAccess\\_TopoART\\_R](#).

**5.10.3.2 Learn()** [2/4] `override void LibTopoART.Fast_TopoART_R.Learn ( byte[] input ) [virtual]`

This method performs a single training step. The independent variables and the dependent variables are automatically separated.

#### Parameters

<i>input</i>	The input vector to be learnt. The input values are internally scaled from [0,255] to [0,1].
--------------	----------------------------------------------------------------------------------------------

Reimplemented from [LibTopoART.Fast\\_TopoART](#).

**5.10.3.3 Learn()** [3/4] `void LibTopoART.Fast_TopoART_R.Learn (`  
`decimal[] i_vec,`  
`decimal[] d_vec )`

This method performs a single training step.

#### Parameters

<i>i_vec</i>	The input vector (independent variables) to be learnt.
<i>d_vec</i>	The output vector (dependent variables) corresponding to <i>i_vec</i> .

Implements [LibTopoART.IDecimalAccess\\_TopoART\\_R](#).

**5.10.3.4 Learn()** [4/4] `override void LibTopoART.Fast_TopoART_R.Learn (`  
`decimal[] input ) [virtual]`

This method performs a single training step. The independent variables and the dependent variables are automatically separated.

#### Parameters

<i>input</i>	The input vector to be learnt.
--------------	--------------------------------

Reimplemented from [LibTopoART.Fast\\_TopoART](#).

**5.10.3.5 Predict()** [1/4] `TopoART_R_Prediction<byte> LibTopoART.Fast_TopoART_R.Predict (`  
`byte[] i_vec,`  
`bool[] m_i_vec,`  
`long nu = 10 )`

This method predicts the dependent variables for a given set of independent variables. Unknown values of independent variables can be signified by setting the corresponding value of *m\_i\_vec* to `true`.

#### Parameters

<i>i_vec</i>	The input vector (independent variables). The elements of the input vector are internally scaled from [0,255] to [0,1].
<i>m_i_vec</i>	The mask vector corresponding to <i>i_vec</i> .
<i>nu</i>	The maximum cardinality of the neighbourhood set N. (In the original TopoART-R network, nu is fixed to 10. But task-specific adaptations might lead to an improved prediction accuracy. This parameter does not modify the network. It may be arbitrarily changed in each prediction step.)



**Returns**

An object of type [TopoART\\_R\\_Prediction](#) containing the predicted values for the unknown independent variables and all dependent variables. The elements of the predicted vectors are internally scaled from [0,1] to [0,255].

Implements [LibTopoART.IByteAccess\\_TopoART\\_R](#).

**5.10.3.6 Predict() [2/4]** `byte [] LibTopoART.Fast_TopoART_R.Predict (`  
`byte[] i_vec,`  
`long nu = 10 )`

This method predicts the dependent variables.

**Parameters**

<i>i_vec</i>	The input vector (independent variables). The elements of the input vector are internally scaled from [0,255] to [0,1].
<i>nu</i>	The maximum cardinality of the neighbourhood set N. (In the original TopoART-R network, nu is fixed to 10. But task-specific adaptations might lead to an improved prediction accuracy. This parameter does not modify the network. It may be arbitrarily changed in each prediction step.)

**Returns**

The predicted values for all dependent variables. The elements of the predicted output vector are internally scaled from [0,1] to [0,255].

Implements [LibTopoART.IByteAccess\\_TopoART\\_R](#).

**5.10.3.7 Predict() [3/4]** `TopoART_R_Prediction<decimal> LibTopoART.Fast_TopoART_R.Predict (`  
`decimal[] i_vec,`  
`bool[] m_i_vec,`  
`long nu = 10 )`

This method predicts the dependent variables for a given set of independent variables. Unknown values of independent variables can be signified by setting the corresponding value of *m\_i\_vec* to `true`.

**Parameters**

<i>i_vec</i>	The input vector (independent variables).
<i>m_i_vec</i>	The mask vector corresponding to <i>i_vec</i> .
<i>nu</i>	The maximum cardinality of the neighbourhood set N. (In the original TopoART-R network, nu is fixed to 10. But task-specific adaptations might lead to an improved prediction accuracy. This parameter does not alter the network. It may be arbitrarily changed in each prediction step.)

### Returns

An object of type [TopoART\\_R\\_Prediction](#) containing the predicted values for the unknown independent variables and all dependent variables.

Implements [LibTopoART.IDecimalAccess\\_TopoART\\_R](#).

```
5.10.3.8 Predict() [4/4] decimal [] LibTopoART.Fast_TopoART_R.Predict (
    decimal[] i_vec,
    long nu = 10 )
```

This method predicts the dependent variables.

### Parameters

<i>i_vec</i>	The input vector (independent variables).
<i>nu</i>	The maximum cardinality of the neighbourhood set N. (In the original TopoART-R network, nu is fixed to 10. But task-specific adaptations might lead to an improved prediction accuracy. This parameter does not modify the network. It may be arbitrarily changed in each prediction step.)

### Returns

The predicted values for all dependent variables.

Implements [LibTopoART.IDecimalAccess\\_TopoART\\_R](#).

## 5.10.4 Property Documentation

```
5.10.4.1 D_len long LibTopoART.Fast_TopoART_R.D_len [get]
```

Property `D_len` returns the length of the output vector (dependent variables).

```
5.10.4.2 FileFormatVersion new decimal LibTopoART.Fast_TopoART_R.FileFormatVersion [get]
```

Property `FileFormatVersion` returns the version of the file format used by class [Fast\\_TopoART\\_R](#).

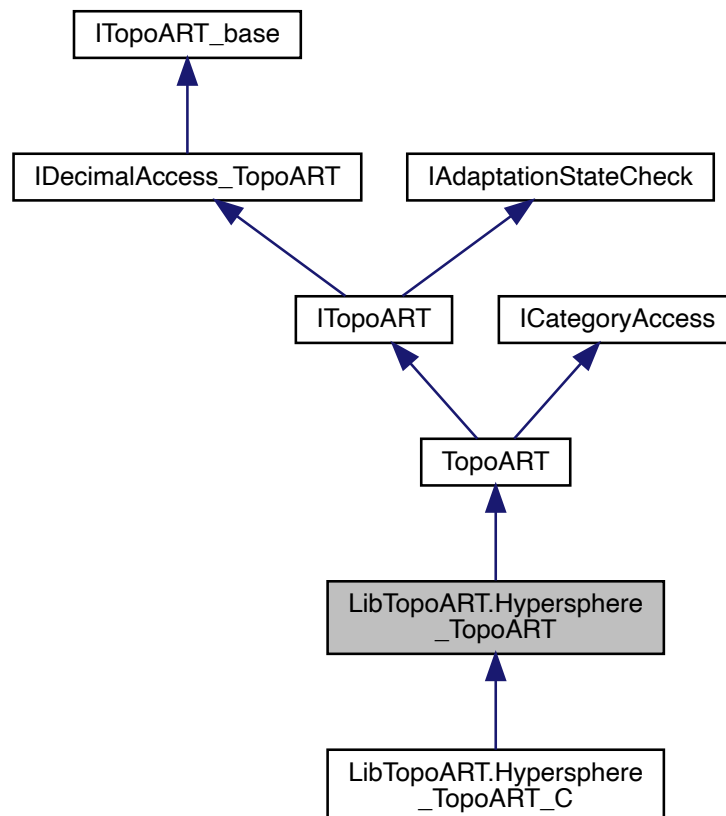
```
5.10.4.3 I_len long LibTopoART.Fast_TopoART_R.I_len [get]
```

Property `I_len` returns the length of the input vector (independent variables).

## 5.11 LibTopoART.Hypersphere\_TopoART Class Reference

Class [Hypersphere\\_TopoART](#) provides an implementation of the Hypersphere [TopoART](#) neural network as proposed in "Marko Tscherepanow (2012). Incremental On-line Clustering with a Topology-Learning Hierarchical ART Neural Network Using Hyperspherical Categories. In Poster and Industry Proceedings of the Industrial Conference on Data Mining (ICDM), pp. 22–34. Fockendorf, Germany: ibai-publishing."

Inheritance diagram for LibTopoART.Hypersphere\_TopoART:



### Public Member Functions

- [Hypersphere\\_TopoART](#) (long input\_length, long module\_number, decimal rho\_a)  
*This constructor initialises a Hypersphere [TopoART](#) network and sets the radial extend parameter to  $\text{Math}.\sqrt{\text{input\_length}}/2$ .*
- [Hypersphere\\_TopoART](#) (long input\_length, long module\_number, decimal rho\_a, decimal R)  
*This constructor initialises a Hypersphere [TopoART](#) network.*
- [Hypersphere\\_TopoART](#) (string path)  
*This constructor loads a saved Hypersphere [TopoART](#) network.*

## Properties

- new decimal [FileFormatVersion](#) [get]  
*Property `FileFormatVersion` returns the version of the file format used by class [Hypersphere\\_TopoART](#).*
- decimal [HypersphereTopoARTFileFormatVersion](#) [get]  
*Property `HypersphereTopoARTFileFormatVersion` returns the version of the file format used by class [Hypersphere\\_TopoART](#).*
- decimal [R](#) [get]  
*Property `R` represents the radial extend parameter  $R$ .*

## Additional Inherited Members

### 5.11.1 Detailed Description

Class [Hypersphere\\_TopoART](#) provides an implementation of the Hypersphere [TopoART](#) neural network as proposed in "Marko Tscherepanow (2012). Incremental On-line Clustering with a Topology-Learning Hierarchical ART Neural Network Using Hyperspherical Categories. In Poster and Industry Proceedings of the Industrial Conference on Data Mining (ICDM), pp. 22–34. Fockendorf, Germany: ibai-publishing."

In contrast to class [TopoART](#), class [Hypersphere\\_TopoART](#) does not require all input to lie in the interval  $[0,1]$ . The input range is controlled by the radial extend parameter  $R$ .

### 5.11.2 Constructor & Destructor Documentation

**5.11.2.1 [Hypersphere\\_TopoART\(\)](#) [1/3]** `LibTopoART.Hypersphere_TopoART.Hypersphere_TopoART ( long input_length, long module_number, decimal rho_a )`

This constructor initialises a Hypersphere [TopoART](#) network and sets the radial extend parameter to  $\text{Math}.\sqrt{\text{input\_length}}/2$ .

#### Parameters

<i>input_length</i>	The length of input vectors to be learnt.
<i>module_number</i>	The number of Hypersphere <a href="#">TopoART</a> modules.
<i>rho_a</i>	The vigilance parameter of the first Hypersphere <a href="#">TopoART</a> module (HTA a).

**5.11.2.2 [Hypersphere\\_TopoART\(\)](#) [2/3]** `LibTopoART.Hypersphere_TopoART.Hypersphere_TopoART ( long input_length, long module_number, decimal rho_a, decimal R )`

This constructor initialises a Hypersphere [TopoART](#) network.

## Parameters

<i>input_length</i>	The length of input vectors to be learnt.
<i>module_number</i>	The number of Hypersphere <a href="#">TopoART</a> modules.
<i>rho_a</i>	The vigilance parameter of the first Hypersphere <a href="#">TopoART</a> module (HTA a).
<i>R</i>	The radial extend parameter.

### 5.11.2.3 Hypersphere\_TopoART() [3/3] `LibTopoART.Hypersphere_TopoART.Hypersphere_TopoART ( string path )`

This constructor loads a saved Hypersphere [TopoART](#) network.

## Parameters

<i>path</i>	The path of a binary Hypersphere <a href="#">TopoART</a> file.
-------------	----------------------------------------------------------------

## Exceptions

<a href="#">InvalidFileException</a>	Throws when the given file cannot be loaded.
--------------------------------------	----------------------------------------------

### 5.11.3 Property Documentation

#### 5.11.3.1 FileFormatVersion `new decimal LibTopoART.Hypersphere_TopoART.FileFormatVersion [get]`

Property `FileFormatVersion` returns the version of the file format used by class [Hypersphere\\_TopoART](#).

#### 5.11.3.2 HypersphereTopoARTFileFormatVersion `decimal LibTopoART.Hypersphere_TopoART.Hypersphere↔TopoARTFileFormatVersion [get]`

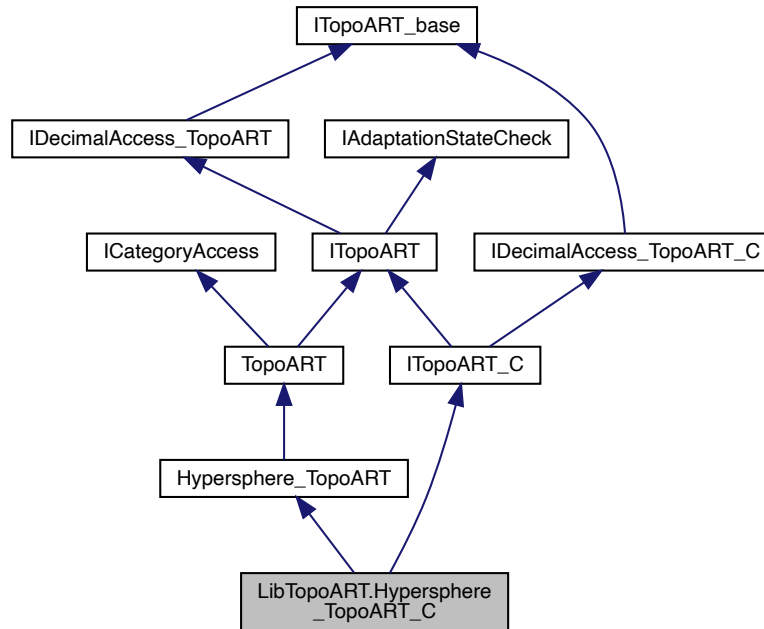
Property `HypersphereTopoARTFileFormatVersion` returns the version of the file format used by class [Hypersphere\\_TopoART](#).

## 5.12 LibTopoART.Hypersphere\_TopoART\_C Class Reference

Class [Hypersphere\\_TopoART\\_C](#) provides an implementation of the Hypersphere TopoART-C neural network. Hypersphere TopoART-C is a combination of Hypersphere [TopoART](#) as proposed in "Marko Tscherepanow (2012). Incremental On-line Clustering with a Topology-Learning Hierarchical ART Neural Network Using Hyperspherical Categories. In Poster and Industry Proceedings of the Industrial Conference on Data Mining (ICDM), pp. 22–34. Fockendorf, Germany: ibai-publishing." and TopoART-C as proposed in "Marko Tscherepanow and Sören Riechers (2012). An Incremental On-line Classifier for Imbalanced, Incomplete, and Noisy Data. In Proceedings of the

European Conference on Artificial Intelligence (ECAI), Workshop on Active and Incremental Learning (AIL), pp. 18-23. Montpellier, France."

Inheritance diagram for LibTopoART.Hypersphere\_TopoART\_C:



## Public Member Functions

- [Hypersphere\\_TopoART\\_C](#) (long input\_length, long module\_number, decimal rho\_a)  
This constructor initialises a Hypersphere TopoART-C network and sets the radial extend parameter to  $\text{Math}.\sqrt{\text{input\_length}}/2$ .
- [Hypersphere\\_TopoART\\_C](#) (long input\_length, long module\_number, decimal rho\_a, decimal R)  
This constructor initialises a Hypersphere TopoART-C network.
- [Hypersphere\\_TopoART\\_C](#) (string path)  
This constructor loads a saved Hypersphere TopoART-C network.
- override void [Learn](#) (decimal[] input)  
This method performs a single training step and sets the class ID corresponding to input to `UNDEFINED_CLASS_ID`.
- void [Learn](#) (decimal[] input, long class\_ID)  
This method performs a single training step.
- long [Predict](#) (decimal[] input, long nu)  
This method predicts the class ID.
- [TopoART\\_C\\_Prediction Predict](#) (decimal[] input, bool[]? mask\_vector, long nu)  
This method predicts the class ID.

## Static Public Attributes

- const long [UNDEFINED\\_CLASS\\_ID](#) = -2  
Instance variable `UNDEFINED_CLASS_ID` gives the value used for indicating that an input sample was predict to belong to the undefined class; i.e, no class ID was provided for such input samples during training.

## Properties

- new decimal [FileFormatVersion](#) [get]

Property `FileFormatVersion` returns the version of the file format used by class `Hypersphere_TopoART_C`.

### 5.12.1 Detailed Description

Class `Hypersphere_TopoART_C` provides an implementation of the Hypersphere TopoART-C neural network. Hypersphere TopoART-C is a combination of Hypersphere [TopoART](#) as proposed in "Marko Tscherepanow (2012). Incremental On-line Clustering with a Topology-Learning Hierarchical ART Neural Network Using Hyperspherical Categories. In Poster and Industry Proceedings of the Industrial Conference on Data Mining (ICDM), pp. 22–34. Fockendorf, Germany: ibai-publishing." and TopoART-C as proposed in "Marko Tscherepanow and Sören Riechers (2012). An Incremental On-line Classifier for Imbalanced, Incomplete, and Noisy Data. In Proceedings of the European Conference on Artificial Intelligence (ECAI), Workshop on Active and Incremental Learning (AIL), pp. 18-23. Montpellier, France."

In contrast to classes `TopoART_C` and `Fast_TopoART_C`, class `Hypersphere_TopoART_C` does not require all input to lie in the interval [0,1]. The input range is controlled by the radial extend parameter `R`.

### 5.12.2 Constructor & Destructor Documentation

**5.12.2.1 `Hypersphere_TopoART_C()` [1/3]** `LibTopoART.Hypersphere_TopoART_C.Hypersphere_TopoART_C (`  
`long input_length,`  
`long module_number,`  
`decimal rho_a )`

This constructor initialises a Hypersphere TopoART-C network and sets the radial extend parameter to `Math.Sqrt(input_length)/2`.

#### Parameters

<code>input_length</code>	The length of input vectors to be learnt.
<code>module_number</code>	The number of Hypersphere TopoART-C modules.
<code>rho_a</code>	The vigilance parameter of the first Hypersphere TopoART-C module (HTA-C a).

**5.12.2.2 `Hypersphere_TopoART_C()` [2/3]** `LibTopoART.Hypersphere_TopoART_C.Hypersphere_TopoART_C (`  
`long input_length,`  
`long module_number,`  
`decimal rho_a,`  
`decimal R )`

This constructor initialises a Hypersphere TopoART-C network.

## Parameters

<i>input_length</i>	The length of input vectors to be learnt.
<i>module_number</i>	The number of Hypersphere TopoART-C modules.
<i>rho_a</i>	The vigilance parameter of the first Hypersphere TopoART-C module (HTA-C a).
<i>R</i>	The radial extend parameter.

**5.12.2.3 Hypersphere\_TopoART\_C()** [3/3] `LibTopoART.Hypersphere_TopoART_C.Hypersphere_TopoART_C (`  
`string path )`

This constructor loads a saved Hypersphere TopoART-C network.

## Parameters

<i>path</i>	The path of a binary Hypersphere TopoART-C file.
-------------	--------------------------------------------------

## Exceptions

<a href="#"><i>InvalidFileException</i></a>	Throws when the given file cannot be loaded.
---------------------------------------------	----------------------------------------------

## 5.12.3 Member Function Documentation

**5.12.3.1 Learn()** [1/2] `override void LibTopoART.Hypersphere_TopoART_C.Learn (`  
`decimal[] input ) [virtual]`

This method performs a single training step and sets the class ID corresponding to *input* to `UNDEFINED_CLASS_ID`.

## Parameters

<i>input</i>	The input vector to be learnt.
--------------	--------------------------------

Reimplemented from [LibTopoART.TopoART](#).

**5.12.3.2 Learn()** [2/2] `void LibTopoART.Hypersphere_TopoART_C.Learn (`  
`decimal[] input,`  
`long class_ID )`

This method performs a single training step.



## Parameters

<i>input</i>	The input vector to be learnt.
<i>class_ID</i>	The class ID corresponding to <i>input</i> . (must be equal to or larger than 0)

## Exceptions

<a href="#"><i>InvalidClassIDException</i></a>	Throws when <i>class_ID</i> is less than 0.
------------------------------------------------	---------------------------------------------

Implements [LibTopoART.IDecimalAccess\\_TopoART\\_C](#).

**5.12.3.3 Predict()** [1/2] [TopoART\\_C\\_Prediction](#) LibTopoART.Hypersphere\_TopoART\_C.Predict (   
     decimal[] *input*,  
     bool?[] *mask\_vector*,  
     long *nu* )

This method predicts the class ID.

## Parameters

<i>input</i>	The input vector the class ID of which is to be predicted.
<i>mask_vector</i>	The mask vector corresponding to <i>input</i> .
<i>nu</i>	The maximum cardinality of the set of enclosing categories E and the neighbourhood set N. (This parameter does not modify the network. It may be arbitrarily changed in each prediction step.)

## Returns

An object of type [TopoART\\_C\\_Prediction](#) containing the predicted class ID and a corresponding confidence value.

**5.12.3.4 Predict()** [2/2] long LibTopoART.Hypersphere\_TopoART\_C.Predict (   
     decimal[] *input*,  
     long *nu* )

This method predicts the class ID.

## Parameters

<i>input</i>	The input vector the class ID of which is to be predicted.
<i>nu</i>	The maximum cardinality of the set of enclosing categories E and the neighbourhood set N. (This parameter does not modify the network. It may be arbitrarily changed in each prediction step.)

## Returns

The predicted class ID.

Implements [LibTopoART.IDecimalAccess\\_TopoART\\_C](#).

## 5.12.4 Member Data Documentation

**5.12.4.1 UNDEFINED\_CLASS\_ID** `const long LibTopoART.Hypersphere_TopoART_C.UNDEFINED_CLASS_ID = -2 [static]`

Instance variable `UNDEFINED_CLASS_ID` gives the value used for indicating that an input sample was predict to belong to the undefined class; i.e, no class ID was provided for such input samples during training.

## 5.12.5 Property Documentation

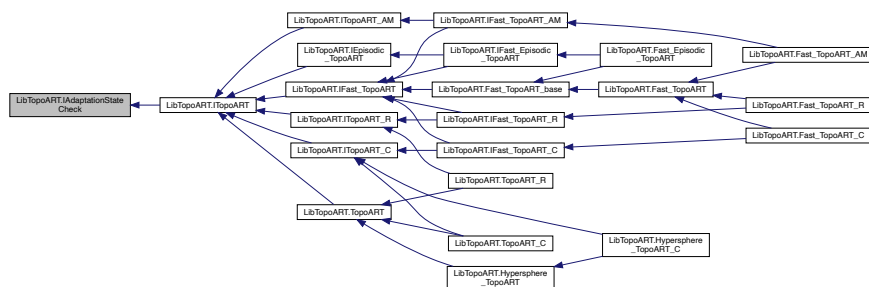
**5.12.5.1 FileFormatVersion** `new decimal LibTopoART.Hypersphere_TopoART_C.FileFormatVersion [get]`

Property `FileFormatVersion` returns the version of the file format used by class [Hypersphere\\_TopoART\\_C](#).

## 5.13 LibTopoART.IAdaptationStateCheck Interface Reference

Interface enabling checks whether certain adaptations of a network occurred.

Inheritance diagram for `LibTopoART.IAdaptationStateCheck`:



## Public Member Functions

- void [ResetAdaptationState](#) ()  
*This method resets the adaptation state to `AdaptationState.NO_ADAPTATION`.*
- [AdaptationState GetAdaptationState](#) (decimal epsilon=0.001m)  
*This method returns the current adaptation state.*

### 5.13.1 Detailed Description

Interface enabling checks whether certain adaptations of a network occurred.

### 5.13.2 Member Function Documentation

**5.13.2.1 GetAdaptationState()** `AdaptationState LibTopoART.IAdaptationStateCheck.GetAdaptationState ( decimal epsilon = 0.001m )`

This method returns the current adaptation state.

#### Parameters

<i>epsilon</i>	The threshold for weight adaptations to be considered.
----------------	--------------------------------------------------------

#### Returns

An enumeration describing the adaptation state.

Implemented in [LibTopoART.TopoART](#), and [LibTopoART.Fast\\_TopoART\\_base](#).

**5.13.2.2 ResetAdaptationState()** `void LibTopoART.IAdaptationStateCheck.ResetAdaptationState ( )`

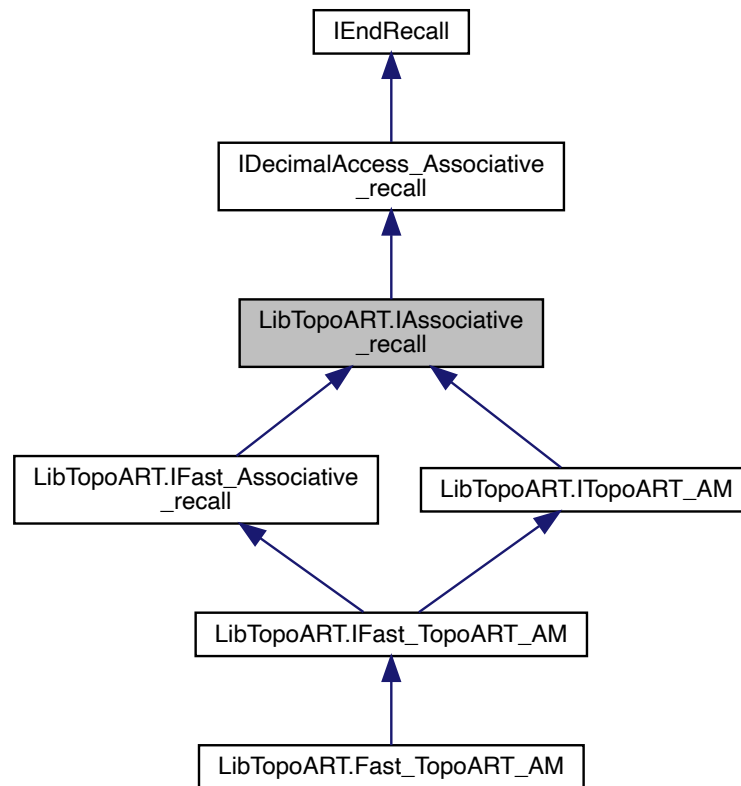
This method resets the adaptation state to `AdaptationState.NO_ADAPTATION`.

Implemented in [LibTopoART.TopoART](#), and [LibTopoART.Fast\\_TopoART\\_base](#).

## 5.14 LibTopoART.IAssociative\_recall Interface Reference

Interface summarising the associative recall functionality using stimulus elements and recall result elements of type `decimal`.

Inheritance diagram for LibTopoART.IAssociative\_recall:



## Additional Inherited Members

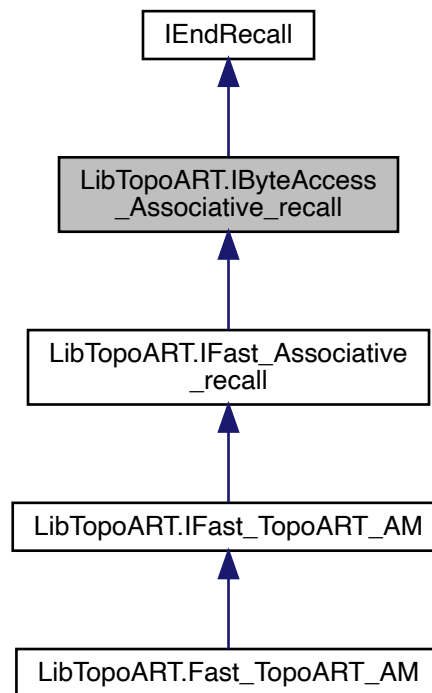
### 5.14.1 Detailed Description

Interface summarising the associative recall functionality using stimulus elements and recall result elements of type `decimal`.

## 5.15 LibTopoART.IByteAccess\_Associative\_recall Interface Reference

Interface providing access to the basic associative recall functionality using stimulus elements and recall result elements of type `byte`. Individual values may use the complete value range from 0 to 255. They are internally converted from/to the [TopoART](#) input interval [0,1].

Inheritance diagram for LibTopoART.IByteAccess\_Associative\_recall:



### Public Member Functions

- long [BeginRecallKey1](#) (byte[] key\_2\_vec, long module\_index=[Fast\\_TopoART\\_AM.FINAL\\_MODULE](#))  
*This method starts the recall process for the first key vector.*
- long [BeginRecallKey2](#) (byte[] key\_1\_vec, long module\_index=[Fast\\_TopoART\\_AM.FINAL\\_MODULE](#))  
*This method starts the recall process for the second key vector.*
- bool [RecallStep](#) (out byte[]? recall\_result, out decimal F3\_activation)  
*This method performs a single associative recall step.*

#### 5.15.1 Detailed Description

Interface providing access to the basic associative recall functionality using stimulus elements and recall result elements of type `byte`. Individual values may use the complete value range from 0 to 255. They are internally converted from/to the [TopoART](#) input interval [0,1].

#### 5.15.2 Member Function Documentation

**5.15.2.1 BeginRecallKey1()** long LibTopoART.IByteAccess\_Associative\_recall.BeginRecallKey1 ( byte[] key\_2\_vec, long module\_index = [Fast\\_TopoART\\_AM.FINAL\\_MODULE](#) )

This method starts the recall process for the first key vector.

**Parameters**

<i>key_2_vec</i>	The stimulus (second key vector) which is used to trigger recall.
<i>module_index</i>	Index of the TopoART-AM module to be used for recall. ( <a href="#">Fast_TopoART_AM.FINAL_MODULE</a> denotes the module with the highest index.)

**Returns**

The number of F3 nodes created.

Implemented in [LibTopoART.Fast\\_TopoART\\_AM](#).

```
5.15.2.2 BeginRecallKey2() long LibTopoART.IByteAccess_Associative_recall.BeginRecallKey2 (
    byte[] key_1_vec,
    long module_index = Fast\_TopoART\_AM.FINAL\_MODULE )
```

This method starts the recall process for the second key vector.

**Parameters**

<i>key_1_vec</i>	The stimulus (first key vector) which is used to trigger recall.
<i>module_index</i>	Index of the TopoART-AM module to be used for recall. ( <a href="#">Fast_TopoART_AM.FINAL_MODULE</a> denotes the module with the highest index.)

**Returns**

The number of F3 nodes created.

Implemented in [LibTopoART.Fast\\_TopoART\\_AM](#).

```
5.15.2.3 RecallStep() bool LibTopoART.IByteAccess_Associative_recall.RecallStep (
    out byte?[] recall_result,
    out decimal F3_activation )
```

This method performs a single associative recall step.

**Parameters**

<i>recall_result</i>	Returns the recall output vector for the current step.
<i>F3_activation</i>	Returns the activation of the current F3 node.

**Returns**

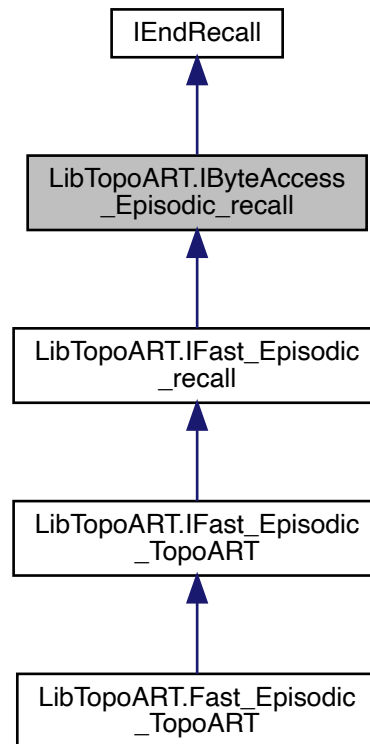
A boolean result indicating whether the recall step was successfully completed, or not.

Implemented in [LibTopoART.Fast\\_TopoART\\_AM](#).

## 5.16 LibTopoART.IByteAccess\_Episodic\_recall Interface Reference

Interface providing access to the basic episodic recall functionality using stimulus elements and recall result elements of type `byte`. Individual values may use the complete value range from 0 to 255. They are internally converted from/to the [TopoART](#) input interval [0,1].

Inheritance diagram for LibTopoART.IByteAccess\_Episodic\_recall:



### Public Member Functions

- long [BeginRecall](#) (byte[] stimulus)  
*This method starts the recall process. The stimulus elements are internally scaled from [0,255] to [0,1].*
- bool [InterEpisodeRecallStep](#) (out byte[]? recall\_result, out decimal F3\_activation)  
*This method performs a single inter-episode recall step and sets the starting point for intra-episode recall.*
- bool [IntraEpisodeRecallStep](#) (out byte[]? recall\_result)  
*This method performs a single intra-episode recall step.*

### 5.16.1 Detailed Description

Interface providing access to the basic episodic recall functionality using stimulus elements and recall result elements of type `byte`. Individual values may use the complete value range from 0 to 255. They are internally converted from/to the [TopoART](#) input interval [0,1].

## 5.16.2 Member Function Documentation

**5.16.2.1 BeginRecall()** `long LibTopoART.IByteAccess_Episodic_recall.BeginRecall ( byte[] stimulus )`

This method starts the recall process. The stimulus elements are internally scaled from [0,255] to [0,1].

### Parameters

<i>stimulus</i>	The stimulus (input) which is used to trigger recall.
-----------------	-------------------------------------------------------

### Returns

The number of F3 nodes created.

Implemented in [LibTopoART.Fast\\_Episodic\\_TopoART](#).

**5.16.2.2 InterEpisodeRecallStep()** `bool LibTopoART.IByteAccess_Episodic_recall.InterEpisode↔ RecallStep ( out byte?[] recall_result, out decimal F3_activation )`

This method performs a single inter-episode recall step and sets the starting point for intra-episode recall.

### Parameters

<i>recall_result</i>	Returns the recall output vector for the current step. The elements of the recall result are internally scaled from [0,1] to [0,255].
<i>F3_activation</i>	Returns the activation of the current F3 node.

### Returns

A boolean result indicating whether the recall step was successfully completed, or not.

Implemented in [LibTopoART.Fast\\_Episodic\\_TopoART](#).

**5.16.2.3 IntraEpisodeRecallStep()** `bool LibTopoART.IByteAccess_Episodic_recall.IntraEpisode↔ RecallStep ( out byte?[] recall_result )`

This method performs a single intra-episode recall step.



## Parameters

<i>recall_result</i>	Returns the recall output vector for the current step. The elements of the recall result are internally scaled from [0,1] to [0,255].
----------------------	---------------------------------------------------------------------------------------------------------------------------------------

## Returns

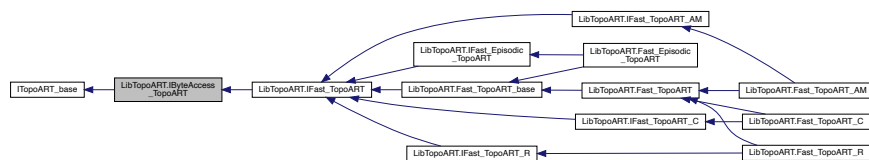
A boolean result indicating whether the recall step was successfully completed or not.

Implemented in [LibTopoART.Fast\\_Episodic\\_TopoART](#).

## 5.17 LibTopoART.IByteAccess\_TopoART Interface Reference

Interface providing access to the basic [TopoART](#) functionality using input elements of type `byte`. Individual values may use the complete value range from 0 to 255. They are internally converted to the [TopoART](#) input interval [0,1].

Inheritance diagram for LibTopoART.IByteAccess\_TopoART:



## Public Member Functions

- `F2_output[] GetBMOutput (byte[] input)`  
*This method finds the closest category for a given test input.*
- `F2_output[] GetBMOutput (byte[] input, bool[] mask_vector)`  
*This method finds the closest category for a given test input.*
- `void Learn (byte[] input)`  
*This method performs a single training step.*

## Additional Inherited Members

## 5.17.1 Detailed Description

Interface providing access to the basic [TopoART](#) functionality using input elements of type `byte`. Individual values may use the complete value range from 0 to 255. They are internally converted to the [TopoART](#) input interval [0,1].

## 5.17.2 Member Function Documentation

### 5.17.2.1 GetBMOutput() [1/2]

```

F2_output [] LibTopoART.IByteAccess_TopoART.GetBMOutput (
    byte[] input )

```

This method finds the closest category for a given test input.

**Parameters**

<i>input</i>	The input vector $x(t)$ . The input values are internally scaled from [0,255] to [0,1].
--------------	-----------------------------------------------------------------------------------------

**Returns**

An array of type `F2_output`. Each entry contains the ID of the best-matching node and the corresponding cluster ID for one `TopoART` module.

Implemented in `LibTopoART.Fast_TopoART_base`.

**5.17.2.2 GetBMOutput()** [2/2] `F2_output [] LibTopoART.IByteAccess_TopoART.GetBMOutput (`  
`byte[] input,`  
`bool[] mask_vector )`

This method finds the closest category for a given test input.

**Parameters**

<i>input</i>	The input vector $x(t)$ . The input values are internally scaled from [0,255] to [0,1].
<i>mask_vector</i>	A mask vector excluding individual dimensions of $x(t)$ from the computation. (Setting an element of the mask vector to <code>true</code> , excludes the corresponding elements of $x(t)$ .)

**Returns**

An array of type `F2_output`. Each entry contains the ID of the best-matching node and the corresponding cluster ID for one `TopoART` module.

**5.17.2.3 Learn()** `void LibTopoART.IByteAccess_TopoART.Learn (`  
`byte[] input )`

This method performs a single training step.

**Parameters**

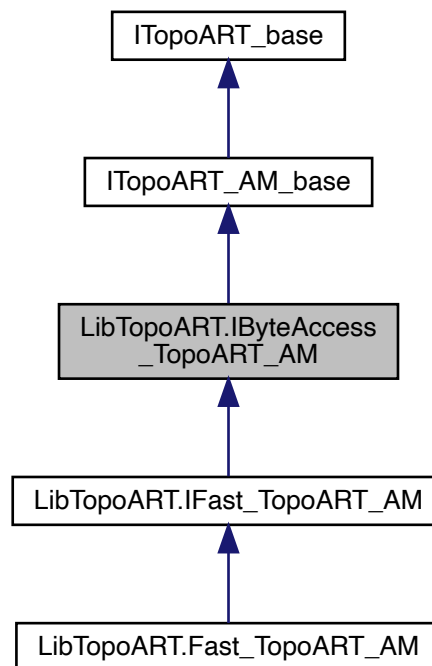
<i>input</i>	The input vector to be learnt. The input values are internally scaled from [0,255] to [0,1].
--------------	----------------------------------------------------------------------------------------------

Implemented in `LibTopoART.Fast_TopoART_R`, `LibTopoART.Fast_TopoART_C`, `LibTopoART.Fast_TopoART_base`, `LibTopoART.Fast_TopoART`, and `LibTopoART.Fast_Episodic_TopoART`.

## 5.18 LibTopoART.IByteAccess\_TopoART\_AM Interface Reference

Interface providing access to the basic TopoART-AM functionality using input elements of type `byte`. Individual values may use the complete value range from 0 to 255. They are internally converted to the `TopoART` input interval [0,1].

Inheritance diagram for LibTopoART.IByteAccess\_TopoART\_AM:



### Public Member Functions

- [F2\\_output\[\] GetBMOutput](#) (byte[] key\_1\_vec, byte[] key\_2\_vec)  
*This method finds the closest category for a given pair of keys.*
- void [Learn](#) (byte[] key\_1\_vec, byte[] key\_2\_vec)  
*This method performs a single training step.*

### Additional Inherited Members

#### 5.18.1 Detailed Description

Interface providing access to the basic TopoART-AM functionality using input elements of type `byte`. Individual values may use the complete value range from 0 to 255. They are internally converted to the [TopoART](#) input interval [0,1].

#### 5.18.2 Member Function Documentation

**5.18.2.1 GetBMOutput()** [F2\\_output](#) [ ] LibTopoART.IByteAccess\_TopoART\_AM.GetBMOutput ( byte[] key\_1\_vec, byte[] key\_2\_vec )

This method finds the closest category for a given pair of keys.

**Parameters**

<i>key_1_vec</i>	The first key vector. The elements of the key vector are internally scaled from [0,255] to [0,1].
<i>key_2_vec</i>	The second key vector corresponding to <i>key_1_vec</i> . The elements of the key vector are internally scaled from [0,255] to [0,1].

**Returns**

An array of type [F2\\_output](#). Each entry contains the ID of the best-matching node and the corresponding cluster ID for one TopoART-AM module.

Implemented in [LibTopoART.Fast\\_TopoART\\_AM](#).

```
5.18.2.2 Learn() void LibTopoART.IByteAccess_TopoART_AM.Learn (
    byte[] key_1_vec,
    byte[] key_2_vec )
```

This method performs a single training step.

**Parameters**

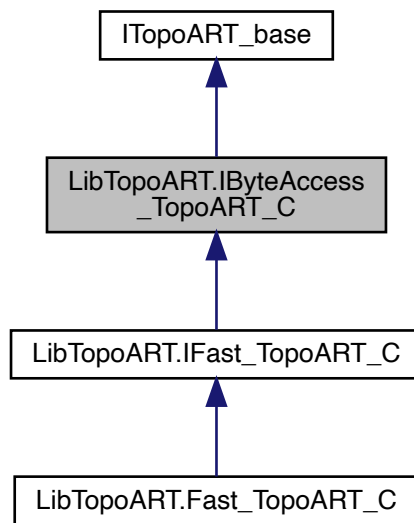
<i>key_1_vec</i>	The first key vector to be learnt. The elements of the key vector are internally scaled from [0,255] to [0,1].
<i>key_2_vec</i>	The second key vector corresponding to <i>key_1_vec</i> . The elements of the key vector are internally scaled from [0,255] to [0,1].

Implemented in [LibTopoART.Fast\\_TopoART\\_AM](#).

## 5.19 LibTopoART.IByteAccess\_TopoART\_C Interface Reference

Interface providing access to the basic TopoART-C functionality using input elements of type `byte`. Individual values may use the complete value range from 0 to 255. They are internally converted to the [TopoART](#) input interval [0,1].

Inheritance diagram for LibTopoART.IByteAccess\_TopoART\_C:



### Public Member Functions

- void [Learn](#) (byte[] input, long class\_ID)  
*This method performs a single training step.*
- long [Predict](#) (byte[] input, long nu)  
*This method predicts the class ID.*
- [TopoART\\_C\\_Prediction Predict](#) (byte[] input, bool[] mask\_vector, long nu)  
*This method predicts the class ID.*

### Additional Inherited Members

#### 5.19.1 Detailed Description

Interface providing access to the basic TopoART-C functionality using input elements of type `byte`. Individual values may use the complete value range from 0 to 255. They are internally converted to the [TopoART](#) input interval [0,1].

#### 5.19.2 Member Function Documentation

**5.19.2.1 Learn()** void LibTopoART.IByteAccess\_TopoART\_C.Learn (  
     byte[] input,  
     long class\_ID )

This method performs a single training step.

**Parameters**

<i>input</i>	The input vector to be learnt. The elements of the input vector are internally scaled from [0,255] to [0,1].
<i>class_ID</i>	The class ID corresponding to <i>input</i> .

Implemented in [LibTopoART.Fast\\_TopoART\\_C](#).

**5.19.2.2 Predict() [1/2]** [TopoART\\_C\\_Prediction](#) LibTopoART.IByteAccess\_TopoART\_C.Predict (   
     byte[] *input*,  
     bool[] *mask\_vector*,  
     long *nu* )

This method predicts the class ID.

**Parameters**

<i>input</i>	The input vector the class ID of which is to be predicted. The elements of the input vector are internally scaled from [0,255] to [0,1].
<i>mask_vector</i>	The mask vector corresponding to <i>input</i> .
<i>nu</i>	The maximum cardinality of the set of enclosing categories E and the neighbourhood set N. (This parameter does not modify the network. It may be arbitrarily changed in each prediction step.)

**Returns**

An object of type [TopoART\\_C\\_Prediction](#) containing the predicted class ID and a corresponding confidence value.

**5.19.2.3 Predict() [2/2]** long LibTopoART.IByteAccess\_TopoART\_C.Predict (   
     byte[] *input*,  
     long *nu* )

This method predicts the class ID.

**Parameters**

<i>input</i>	The input vector the class ID of which is to be predicted. The elements of the input vector are internally scaled from [0,255] to [0,1].
<i>nu</i>	The maximum cardinality of the set of enclosing categories E and the neighbourhood set N. (This parameter does not modify the network. It may be arbitrarily changed in each prediction step.)

**Returns**

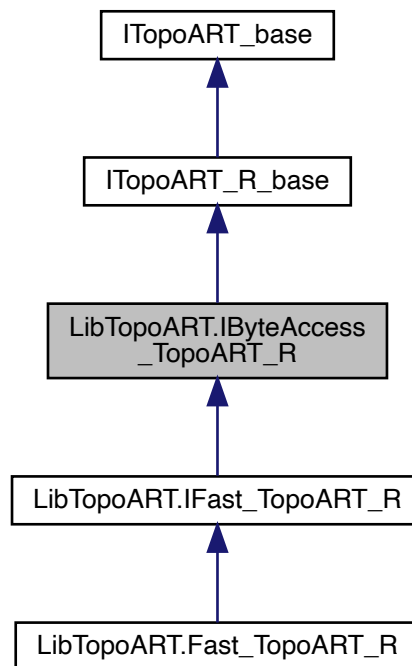
The predicted class ID.

Implemented in [LibTopoART.Fast\\_TopoART\\_C](#).

## 5.20 LibTopoART.IByteAccess\_TopoART\_R Interface Reference

Interface providing access to the basic TopoART-R functionality using input elements of type `byte`. Individual values may use the complete value range from 0 to 255. They are internally converted to the [TopoART](#) input interval [0,1].

Inheritance diagram for LibTopoART.IByteAccess\_TopoART\_R:



### Public Member Functions

- void [Learn](#) (byte[] i\_vec, byte[] d\_vec)  
*This method performs a single training step.*
- byte[] [Predict](#) (byte[] i\_vec, long nu=10)  
*This method predicts the dependent variables.*
- [TopoART\\_R\\_Prediction](#)< byte > [Predict](#) (byte[] i\_vec, bool[] m\_i\_vec, long nu=10)  
*This method predicts the dependent variables for a given set of independent variables. Unknown values of independent variables can be signified by setting the corresponding value of m\_i\_vec to `true`.*

### Additional Inherited Members

#### 5.20.1 Detailed Description

Interface providing access to the basic TopoART-R functionality using input elements of type `byte`. Individual values may use the complete value range from 0 to 255. They are internally converted to the [TopoART](#) input interval [0,1].

## 5.20.2 Member Function Documentation

**5.20.2.1 Learn()** `void LibTopoART.IByteAccess_TopoART_R.Learn (`  
`byte[] i_vec,`  
`byte[] d_vec )`

This method performs a single training step.

### Parameters

<i>i_vec</i>	The input vector (independent variables) to be learnt. The elements of the input vector are internally scaled from [0,255] to [0,1].
<i>d_vec</i>	The output vector (dependent variables) corresponding to <i>i_vec</i> . The elements of the output vector are internally scaled from [0,255] to [0,1].

Implemented in [LibTopoART.Fast\\_TopoART\\_R](#).

**5.20.2.2 Predict()** [1/2] `TopoART_R_Prediction<byte> LibTopoART.IByteAccess_TopoART_R.Predict (`  
`byte[] i_vec,`  
`bool[] m_i_vec,`  
`long nu = 10 )`

This method predicts the dependent variables for a given set of independent variables. Unknown values of independent variables can be signified by setting the corresponding value of *m\_i\_vec* to `true`.

### Parameters

<i>i_vec</i>	The input vector (independent variables). The elements of the input vector are internally scaled from [0,255] to [0,1].
<i>m_i_vec</i>	The mask vector corresponding to <i>i_vec</i> .
<i>nu</i>	The maximum cardinality of the neighbourhood set N. (In the original TopoART-R network, nu is fixed to 10. But task-specific adaptations might lead to an improved prediction accuracy. This parameter does not modify the network. It may be arbitrarily changed in each prediction step.)

### Returns

An object of type [TopoART\\_R\\_Prediction](#) containing the predicted values for the unknown independent variables and all dependent variables. The elements of the predicted vectors are internally scaled from [0,1] to [0,255].

Implemented in [LibTopoART.Fast\\_TopoART\\_R](#).

**5.20.2.3 Predict()** [2/2] `byte [] LibTopoART.IByteAccess_TopoART_R.Predict (`  
`byte[] i_vec,`  
`long nu = 10 )`

This method predicts the dependent variables.



## Parameters

<i>i_vec</i>	The input vector (independent variables). The elements of the input vector are internally scaled from [0,255] to [0,1].
<i>nu</i>	The maximum cardinality of the neighbourhood set N. (In the original TopoART-R network, nu is fixed to 10. But task-specific adaptations might lead to an improved prediction accuracy. This parameter does not modify the network. It may be arbitrarily changed in each prediction step.)

## Returns

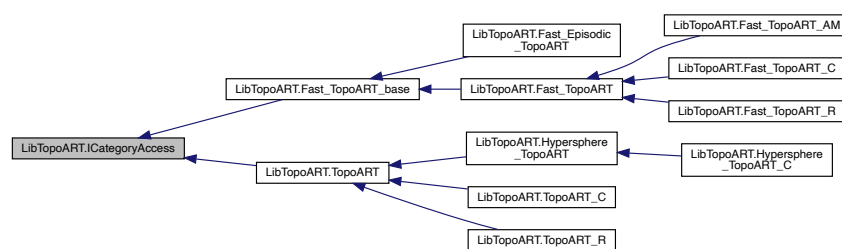
The predicted values for all dependent variables. The elements of the predicted output vector are internally scaled from [0,1] to [0,255].

Implemented in [LibTopoART.Fast\\_TopoART\\_R](#).

## 5.21 LibTopoART.ICategoryAccess Interface Reference

Interface providing access to the learnt categories, e.g for drawing.

Inheritance diagram for LibTopoART.ICategoryAccess:



## Public Member Functions

- List< [CategoryInfo](#) >? [GetCategories](#) (long module\_index=[LibTopoART\\_info.FINAL\\_MODULE](#))  
This method collects information on the categories of a specified module.

### 5.21.1 Detailed Description

Interface providing access to the learnt categories, e.g for drawing.

### 5.21.2 Member Function Documentation

**5.21.2.1 GetCategories()** List<[CategoryInfo](#)>? LibTopoART.ICategoryAccess.GetCategories ( long module\_index = [LibTopoART\\_info.FINAL\\_MODULE](#) )

This method collects information on the categories of a specified module.

## Parameters

<code>module_index</code>	The index of the module information on the categories of which is to be returned.
---------------------------	-----------------------------------------------------------------------------------

## Returns

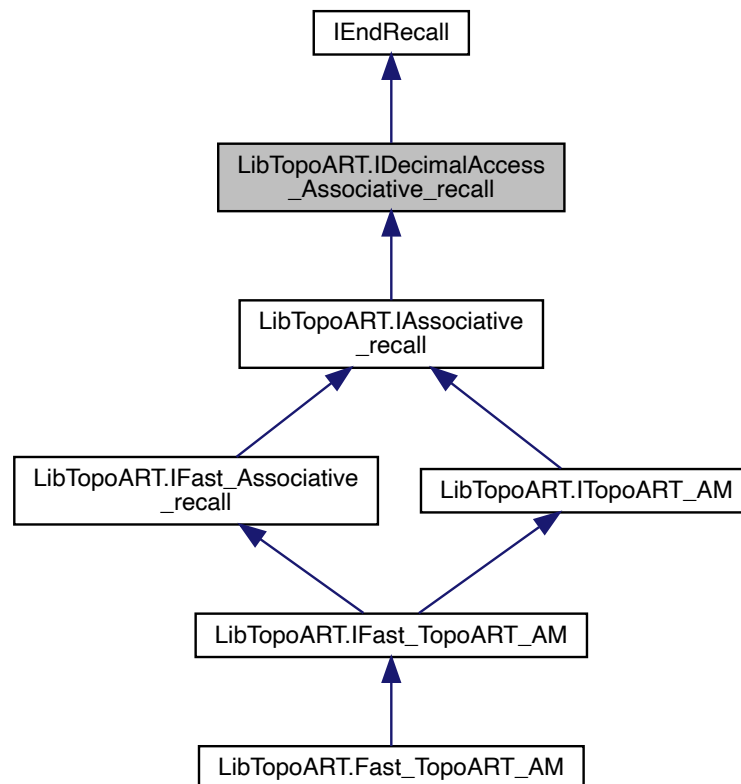
A list containing information about the respective categories.

Implemented in [LibTopoART.TopoART](#), and [LibTopoART.Fast\\_TopoART\\_base](#).

## 5.22 LibTopoART.IDecimalAccess\_Associative\_recall Interface Reference

Interface providing access to the basic associative recall functionality using stimulus elements and recall result elements of type `decimal`. Individual stimulus values must lie within the [TopoART](#) input interval [0,1].

Inheritance diagram for LibTopoART.IDecimalAccess\_Associative\_recall:



## Public Member Functions

- long [BeginRecallKey1](#) (decimal[] key\_2\_vec, long module\_index=[Fast\\_TopoART\\_AM.FINAL\\_MODULE](#))  
*This method starts the recall process for the first key vector.*
- long [BeginRecallKey2](#) (decimal[] key\_1\_vec, long module\_index=[Fast\\_TopoART\\_AM.FINAL\\_MODULE](#))  
*This method starts the recall process for the second key vector.*
- bool [RecallStep](#) (out decimal[]? recall\_result, out decimal F3\_activation)  
*This method performs a single associative recall step.*

### 5.22.1 Detailed Description

Interface providing access to the basic associative recall functionality using stimulus elements and recall result elements of type `decimal`. Individual stimulus values must lie within the [TopoART](#) input interval [0,1].

### 5.22.2 Member Function Documentation

**5.22.2.1 BeginRecallKey1()** `long LibTopoART.IDecimalAccess_Associative_recall.BeginRecallKey1 ( decimal[] key_2_vec, long module_index = Fast\_TopoART\_AM.FINAL\_MODULE )`

This method starts the recall process for the first key vector.

#### Parameters

<i>key_2_vec</i>	The stimulus (second key vector) which is used to trigger recall. The stimulus elements are internally scaled from [0,255] to [0,1].
<i>module_index</i>	Index of the TopoART-AM module to be used for recall. ( <a href="#">Fast_TopoART_AM.FINAL_MODULE</a> denotes the module with the highest index.)

#### Returns

The number of F3 nodes created.

Implemented in [LibTopoART.Fast\\_TopoART\\_AM](#).

**5.22.2.2 BeginRecallKey2()** `long LibTopoART.IDecimalAccess_Associative_recall.BeginRecallKey2 ( decimal[] key_1_vec, long module_index = Fast\_TopoART\_AM.FINAL\_MODULE )`

This method starts the recall process for the second key vector.

#### Parameters

<i>key_1_vec</i>	The stimulus (first key vector) which is used to trigger recall. The stimulus elements are internally scaled from [0,255] to [0,1].
<i>module_index</i>	Index of the TopoART-AM module to be used for recall. ( <a href="#">Fast_TopoART_AM.FINAL_MODULE</a> denotes the module with the highest index.)

#### Returns

The number of F3 nodes created.

Implemented in [LibTopoART.Fast\\_TopoART\\_AM](#).

**5.22.2.3 RecallStep()** `bool LibTopoART.IDecimalAccess_Associative_recall.RecallStep (`  
    `out decimal?[] recall_result,`  
    `out decimal F3_activation )`

This method performs a single associative recall step.

#### Parameters

<i>recall_result</i>	Returns the recall output vector for the current step. /// The elements of the recall result are internally scaled from [0,1] to [0,255].
<i>F3_activation</i>	Returns the activation of the current F3 node.

#### Returns

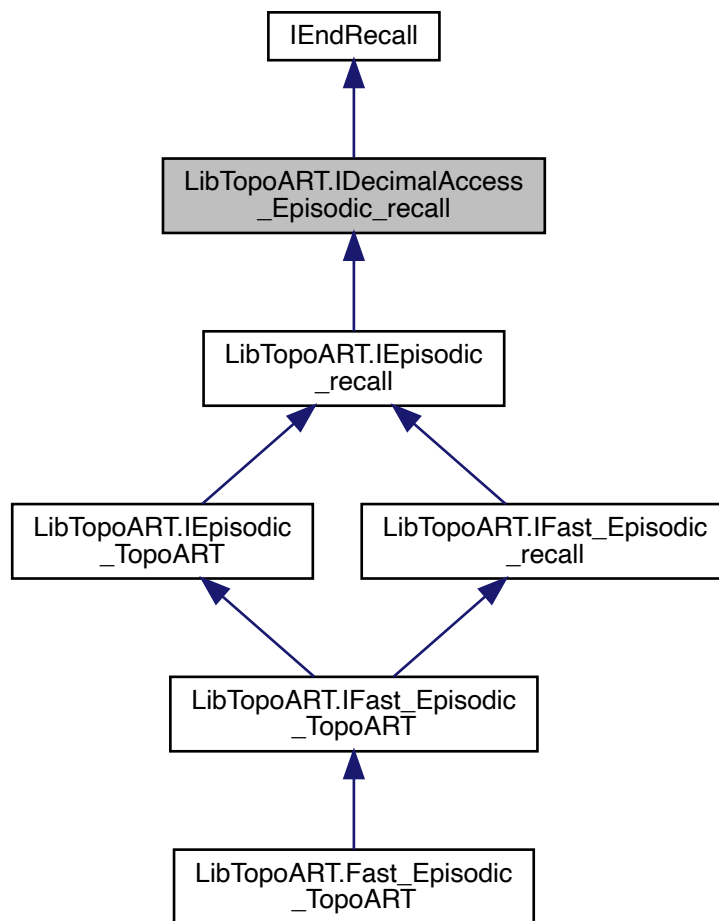
A boolean result indicating whether the recall step was successfully completed, or not.

Implemented in [LibTopoART.Fast\\_TopoART\\_AM](#).

## 5.23 LibTopoART.IDecimalAccess\_Episodic\_recall Interface Reference

Interface providing access to the basic episodic recall functionality using stimulus elements and recall result elements of type `decimal`. Individual stimulus values must lie within the [TopoART](#) input interval [0,1].

Inheritance diagram for LibTopoART.IDecimalAccess\_Episodic\_recall:



## Public Member Functions

- long [BeginRecall](#) (decimal[] stimulus)  
*This method starts the recall process.*
- bool [InterEpisodeRecallStep](#) (out decimal[]? recall\_result, out decimal F3\_activation)  
*This method performs a single inter-episode recall step and sets the starting point for intra-episode recall.*
- bool [IntraEpisodeRecallStep](#) (out decimal[]? recall\_result)  
*This method performs a single intra-episode recall step.*

### 5.23.1 Detailed Description

Interface providing access to the basic episodic recall functionality using stimulus elements and recall result elements of type `decimal`. Individual stimulus values must lie within the [TopoART](#) input interval [0,1].

## 5.23.2 Member Function Documentation

**5.23.2.1 BeginRecall()** `long LibTopoART.IDecimalAccess_Episodic_recall.BeginRecall ( decimal[] stimulus )`

This method starts the recall process.

### Parameters

<i>stimulus</i>	The stimulus (input) which is used to trigger recall.
-----------------	-------------------------------------------------------

### Returns

The number of F3 nodes created.

Implemented in [LibTopoART.Fast\\_Episodic\\_TopoART](#).

**5.23.2.2 InterEpisodeRecallStep()** `bool LibTopoART.IDecimalAccess_Episodic_recall.InterEpisode↵ RecallStep ( out decimal?[] recall_result, out decimal F3_activation )`

This method performs a single inter-episode recall step and sets the starting point for intra-episode recall.

### Parameters

<i>recall_result</i>	Returns the recall output vector for the current step.
<i>F3_activation</i>	Returns the activation of the current F3 node.

### Returns

A boolean result indicating whether the recall step was successfully completed, or not.

Implemented in [LibTopoART.Fast\\_Episodic\\_TopoART](#).

**5.23.2.3 IntraEpisodeRecallStep()** `bool LibTopoART.IDecimalAccess_Episodic_recall.IntraEpisode↵ RecallStep ( out decimal?[] recall_result )`

This method performs a single intra-episode recall step.

## Parameters

<code>recall_result</code>	Returns the recall output vector for the current step.
----------------------------	--------------------------------------------------------

## Returns

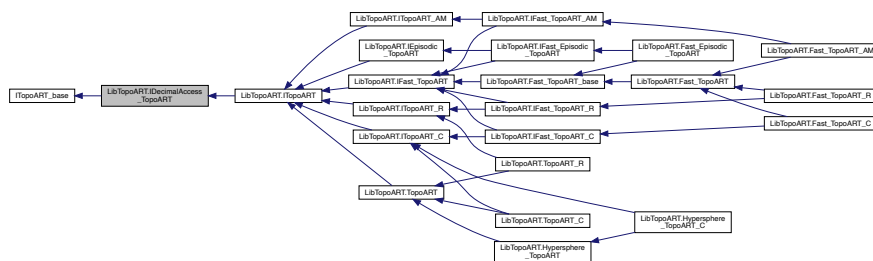
A boolean result indicating whether the recall step was successfully completed or not.

Implemented in [LibTopoART.Fast\\_Episodic\\_TopoART](#).

## 5.24 LibTopoART.IDecimalAccess\_TopoART Interface Reference

Interface providing access to the basic [TopoART](#) functionality using input elements of type `decimal`. Individual values must lie within the [TopoART](#) input interval [0,1].

Inheritance diagram for LibTopoART.IDecimalAccess\_TopoART:



## Public Member Functions

- `F2_output[] GetBMOutput (decimal[] input)`  
*This method finds the closest category for a given test input.*
- `F2_output[] GetBMOutput (decimal[] input, bool[] mask_vector)`  
*This method finds the closest category for a given test input.*
- `void Learn (decimal[] input)`  
*This method performs a single training step.*

## Additional Inherited Members

### 5.24.1 Detailed Description

Interface providing access to the basic [TopoART](#) functionality using input elements of type `decimal`. Individual values must lie within the [TopoART](#) input interval [0,1].

### 5.24.2 Member Function Documentation

#### 5.24.2.1 GetBMOutput() [1/2] `F2_output [] LibTopoART.IDecimalAccess_TopoART.GetBMOutput ( decimal[] input )`

This method finds the closest category for a given test input.

## Parameters

<i>input</i>	The input vector $x(t)$ .
--------------	---------------------------

## Returns

An array of type `F2_output`. Each entry contains the ID of the best-matching node and the corresponding cluster ID for one `TopoART` module.

Implemented in `LibTopoART.TopoART`, and `LibTopoART.Fast_TopoART_base`.

**5.24.2.2 GetBMOutput()** [2/2] `F2_output [] LibTopoART.IDecimalAccess_TopoART.GetBMOutput ( decimal[] input, bool[] mask_vector )`

This method finds the closest category for a given test input.

## Parameters

<i>input</i>	The input vector $x(t)$ .
<i>mask_vector</i>	A mask vector excluding individual dimensions of $x(t)$ from the computation. (Setting an element of the mask vector to <code>true</code> , excludes the corresponding elements of $x(t)$ .)

## Returns

An array of type `F2_output`. Each entry contains the ID of the best-matching node and the corresponding cluster ID for one `TopoART` module.

**5.24.2.3 Learn()** `void LibTopoART.IDecimalAccess_TopoART.Learn ( decimal[] input )`

This method performs a single training step.

## Parameters

<i>input</i>	The input vector to be learnt.
--------------	--------------------------------

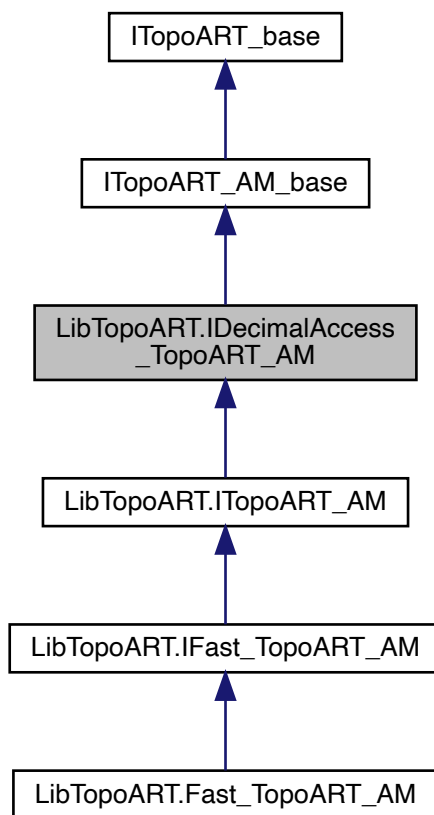
Implemented in `LibTopoART.Fast_TopoART_R`, `LibTopoART.TopoART_R`, `LibTopoART.Fast_TopoART_C`, `LibTopoART.TopoART_C`, `LibTopoART.TopoART`, `LibTopoART.Hypersphere_TopoART_C`, `LibTopoART.Fast_TopoART_base`, `LibTopoART.Fast_TopoART`, and `LibTopoART.Fast_Episodic_TopoART`.

## 5.25 LibTopoART.IDecimalAccess\_TopoART\_AM Interface Reference

Interface providing access to the basic TopoART-AM functionality using input elements of type `decimal`. Individual values must lie within the `TopoART` input interval  $[0,1]$ .



Inheritance diagram for LibTopoART.IDecimalAccess\_TopoART\_AM:



### Public Member Functions

- [F2\\_output\[\]](#) [GetBMOutput](#) (decimal[] key\_1\_vec, decimal[] key\_2\_vec)  
*This method finds the closest category for a given pair of keys.*
- void [Learn](#) (decimal[] key\_1\_vec, decimal[] key\_2\_vec)  
*This method performs a single training step.*

### Additional Inherited Members

#### 5.25.1 Detailed Description

Interface providing access to the basic TopoART-AM functionality using input elements of type `decimal`. Individual values must lie within the [TopoART](#) input interval [0,1].

#### 5.25.2 Member Function Documentation

**5.25.2.1 GetBMOutput()** `F2_output [ ] LibTopoART.IDecimalAccess_TopoART_AM.GetBMOutput ( decimal[ ] key_1_vec, decimal[ ] key_2_vec )`

This method finds the closest category for a given pair of keys.

#### Parameters

<i>key_1_vec</i>	The first key vector.
<i>key_2_vec</i>	The second key vector corresponding to <i>key_1_vec</i> .

#### Returns

An array of type `F2_output`. Each entry contains the ID of the best-matching node and the corresponding cluster ID for one TopoART-AM module.

Implemented in [LibTopoART.Fast\\_TopoART\\_AM](#).

**5.25.2.2 Learn()** `void LibTopoART.IDecimalAccess_TopoART_AM.Learn ( decimal[ ] key_1_vec, decimal[ ] key_2_vec )`

This method performs a single training step.

#### Parameters

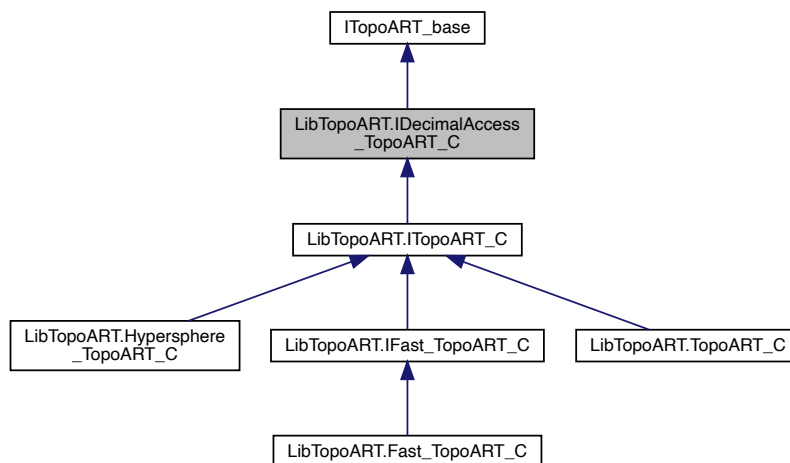
<i>key_1_vec</i>	The first key vector to be learnt.
<i>key_2_vec</i>	The second key vector corresponding to <i>key_1_vec</i> .

Implemented in [LibTopoART.Fast\\_TopoART\\_AM](#).

## 5.26 LibTopoART.IDecimalAccess\_TopoART\_C Interface Reference

Interface providing access to the basic TopoART-C functionality using input elements of type `decimal`. Individual values must lie within the [TopoART](#) input interval [0,1].

Inheritance diagram for LibTopoART.IDecimalAccess\_TopoART\_C:



### Public Member Functions

- void [Learn](#) (decimal[] input, long class\_ID)  
*This method performs a single training step.*
- long [Predict](#) (decimal[] input, long nu)  
*This method predicts the class ID.*
- [TopoART\\_C\\_Prediction Predict](#) (decimal[] input, bool[] mask\_vector, long nu)  
*This method predicts the class ID.*

### Additional Inherited Members

#### 5.26.1 Detailed Description

Interface providing access to the basic TopoART-C functionality using input elements of type `decimal`. Individual values must lie within the [TopoART](#) input interval [0,1].

#### 5.26.2 Member Function Documentation

**5.26.2.1 Learn()** void LibTopoART.IDecimalAccess\_TopoART\_C.Learn (  
    decimal[] input,  
    long class\_ID )

This method performs a single training step.

**Parameters**

<i>input</i>	The input vector to be learnt.
<i>class_ID</i>	The class ID corresponding to <i>input</i> .

Implemented in [LibTopoART.Fast\\_TopoART\\_C](#), [LibTopoART.TopoART\\_C](#), and [LibTopoART.Hypersphere\\_TopoART\\_C](#).

**5.26.2.2 Predict() [1/2]** [TopoART\\_C\\_Prediction](#) LibTopoART.IDecimalAccess\_TopoART\_C.Predict (   
     decimal[] *input*,  
     bool[] *mask\_vector*,  
     long *nu* )

This method predicts the class ID.

**Parameters**

<i>input</i>	The input vector the class ID of which is to be predicted.
<i>mask_vector</i>	The mask vector corresponding to <i>input</i> .
<i>nu</i>	The maximum cardinality of the set of enclosing categories E and the neighbourhood set N. (This parameter does not modify the network. It may be arbitrarily changed in each prediction step.)

**Returns**

An object of type [TopoART\\_C\\_Prediction](#) containing the predicted class ID and a corresponding confidence value.

**5.26.2.3 Predict() [2/2]** long LibTopoART.IDecimalAccess\_TopoART\_C.Predict (   
     decimal[] *input*,  
     long *nu* )

This method predicts the class ID.

**Parameters**

<i>input</i>	The input vector the class ID of which is to be predicted.
<i>nu</i>	The maximum cardinality of the set of enclosing categories E and the neighbourhood set N. (This parameter does not modify the network. It may be arbitrarily changed in each prediction step.)

**Returns**

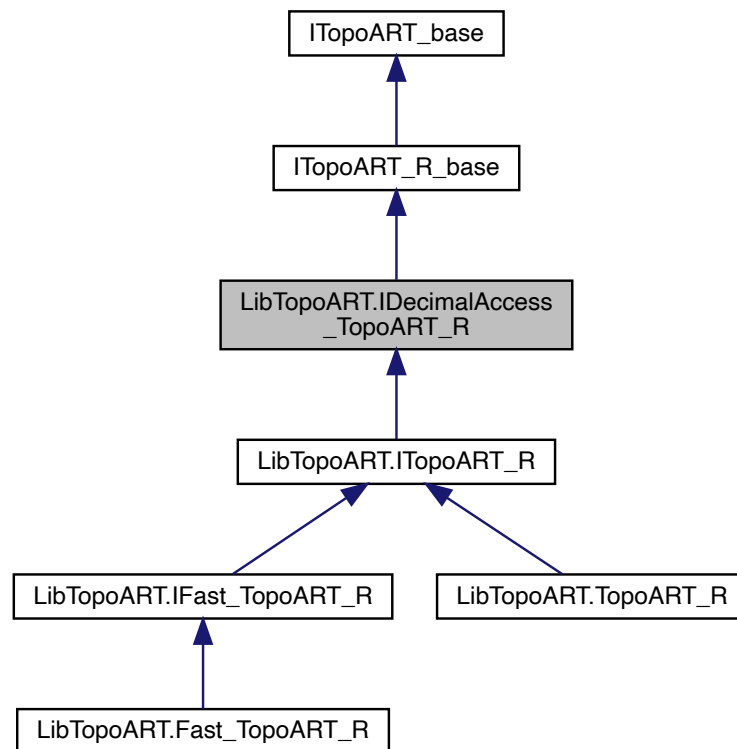
The predicted class ID.

Implemented in [LibTopoART.Fast\\_TopoART\\_C](#), [LibTopoART.TopoART\\_C](#), and [LibTopoART.Hypersphere\\_TopoART\\_C](#).

## 5.27 LibTopoART.IDecimalAccess\_TopoART\_R Interface Reference

Interface providing access to the basic TopoART-R functionality using input elements of type `decimal`. Individual values must lie within the [TopoART](#) input interval [0,1].

Inheritance diagram for LibTopoART.IDecimalAccess\_TopoART\_R:



### Public Member Functions

- void [Learn](#) (decimal[] i\_vec, decimal[] d\_vec)  
*This method performs a single training step.*
- decimal[] [Predict](#) (decimal[] i\_vec, long nu=10)  
*This method predicts the dependent variables.*
- [TopoART\\_R\\_Prediction](#)< decimal > [Predict](#) (decimal[] i\_vec, bool[] m\_i\_vec, long nu=10)  
*This method predicts the dependent variables for a given set of independent variables. Unknown values of independent variables can be signified by setting the corresponding value of `m_i_vec` to `true`.*

### Additional Inherited Members

#### 5.27.1 Detailed Description

Interface providing access to the basic TopoART-R functionality using input elements of type `decimal`. Individual values must lie within the [TopoART](#) input interval [0,1].

## 5.27.2 Member Function Documentation

**5.27.2.1 Learn()** `void LibTopoART.IDecimalAccess_TopoART_R.Learn (`  
`decimal[] i_vec,`  
`decimal[] d_vec )`

This method performs a single training step.

### Parameters

<i>i_vec</i>	The input vector (independent variables) to be learnt.
<i>d_vec</i>	The output vector (dependent variables) corresponding to <i>i_vec</i> .

Implemented in [LibTopoART.Fast\\_TopoART\\_R](#), and [LibTopoART.TopoART\\_R](#).

**5.27.2.2 Predict()** [1/2] `TopoART_R_Prediction<decimal> LibTopoART.IDecimalAccess_TopoART_R.↔`  
`Predict (`  
`decimal[] i_vec,`  
`bool[] m_i_vec,`  
`long nu = 10 )`

This method predicts the dependent variables for a given set of independent variables. Unknown values of independent variables can be signified by setting the corresponding value of *m\_i\_vec* to `true`.

### Parameters

<i>i_vec</i>	The input vector (independent variables).
<i>m_i_vec</i>	The mask vector corresponding to <i>i_vec</i> .
<i>nu</i>	The maximum cardinality of the neighbourhood set N. (In the original TopoART-R network, nu is fixed to 10. But task-specific adaptations might lead to an improved prediction accuracy. This parameter does not modify the network. It may be arbitrarily changed in each prediction step.)

### Returns

An object of type [TopoART\\_R\\_Prediction](#) containing the predicted values for the unknown independent variables and all dependent variables.

Implemented in [LibTopoART.Fast\\_TopoART\\_R](#), and [LibTopoART.TopoART\\_R](#).

**5.27.2.3 Predict()** [2/2] `decimal [] LibTopoART.IDecimalAccess_TopoART_R.Predict (`  
`decimal[] i_vec,`  
`long nu = 10 )`

This method predicts the dependent variables.

## Parameters

<i>i_vec</i>	The input vector (independent variables).
<i>nu</i>	The maximum cardinality of the neighbourhood set N. (In the original TopoART-R network, nu is fixed to 10. But task-specific adaptations might lead to an improved prediction accuracy. This parameter does not modify the network. It may be arbitrarily changed in each prediction step.)

## Returns

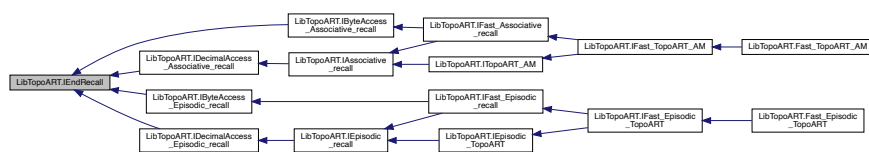
The predicted values for all dependent variables.

Implemented in [LibTopoART.Fast\\_TopoART\\_R](#), and [LibTopoART.TopoART\\_R](#).

## 5.28 LibTopoART.IEndRecall Interface Reference

Interface summarising the type-independent functionality to stop the recall process.

Inheritance diagram for LibTopoART.IEndRecall:



## Public Member Functions

- void [EndRecall](#) ()

*This method stops the recall process and frees temporary resources.*

### 5.28.1 Detailed Description

Interface summarising the type-independent functionality to stop the recall process.

### 5.28.2 Member Function Documentation

#### 5.28.2.1 EndRecall() void LibTopoART.IEndRecall.EndRecall ( )

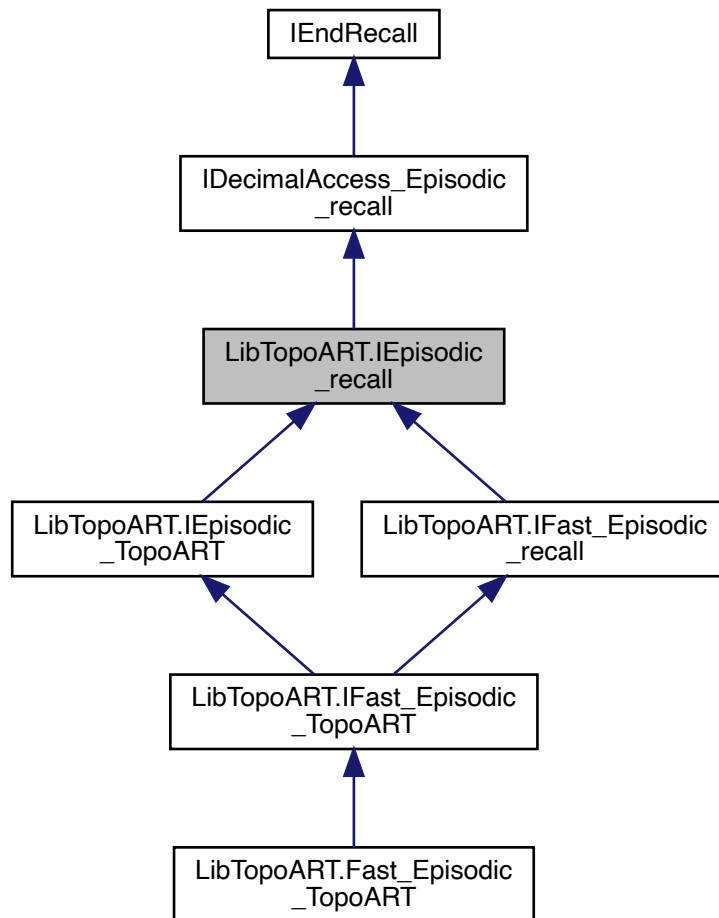
This method stops the recall process and frees temporary resources.

Implemented in [LibTopoART.Fast\\_TopoART\\_AM](#), and [LibTopoART.Fast\\_Episodic\\_TopoART](#).

## 5.29 LibTopoART.IEpisodic\_recall Interface Reference

Interface summarising the episodic recall functionality using stimulus elements and recall result elements of type `decimal`.

Inheritance diagram for LibTopoART.IEpisodic\_recall:



### Additional Inherited Members

#### 5.29.1 Detailed Description

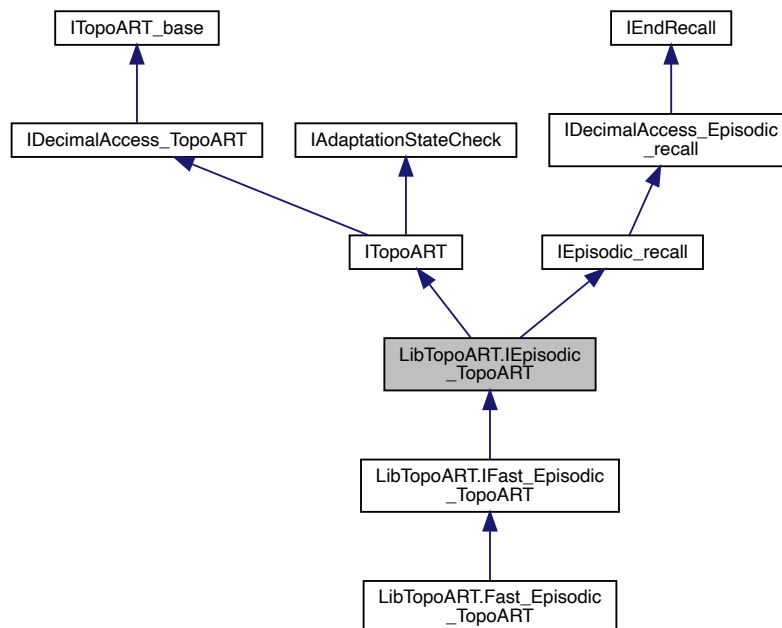
Interface summarising the episodic recall functionality using stimulus elements and recall result elements of type `decimal`.



## 5.30 LibTopoART.IEpisodic\_TopoART Interface Reference

Interface summarising the Episodic [TopoART](#) functionality including learning, prediction, episodic recall using input elements, stimulus elements, and recall result elements of type `decimal` as well as adaptation state control.

Inheritance diagram for LibTopoART.IEpisodic\_TopoART:



### Additional Inherited Members

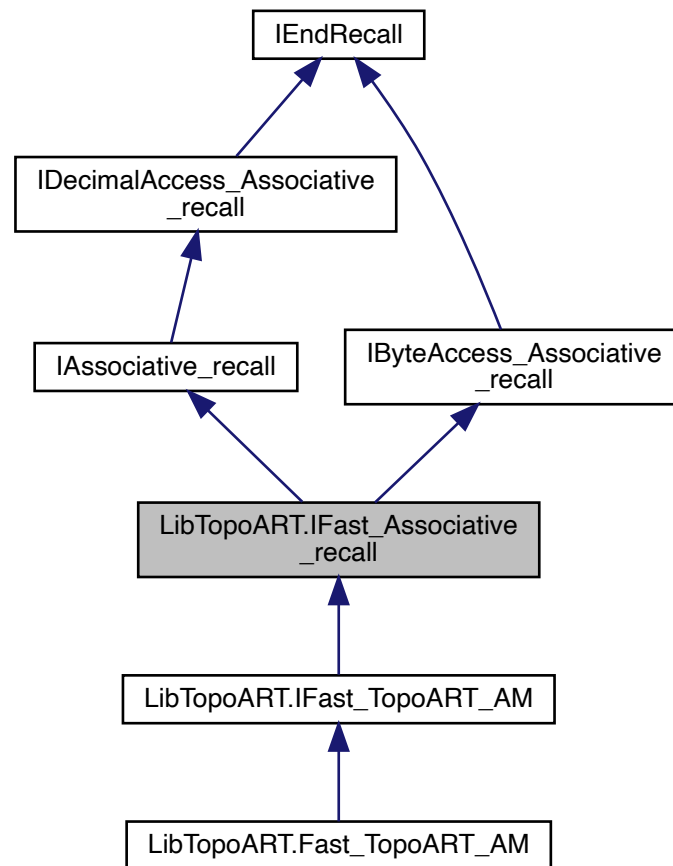
#### 5.30.1 Detailed Description

Interface summarising the Episodic [TopoART](#) functionality including learning, prediction, episodic recall using input elements, stimulus elements, and recall result elements of type `decimal` as well as adaptation state control.

## 5.31 LibTopoART.IFast\_Associative\_recall Interface Reference

Interface summarising the associative recall functionality using stimulus elements and recall result elements of type `byte` or of type `decimal`.

Inheritance diagram for LibTopoART.IFast\_Associative\_recall:



## Additional Inherited Members

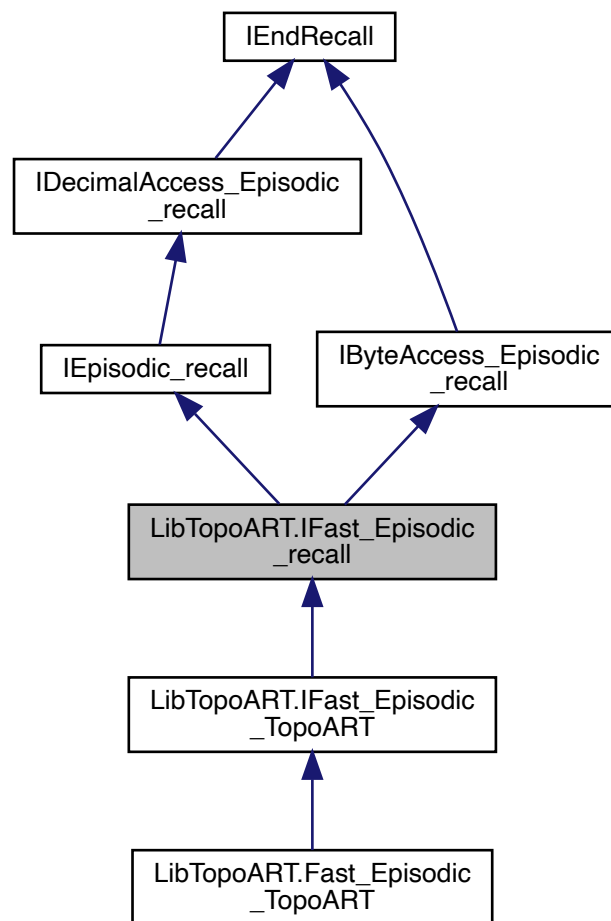
### 5.31.1 Detailed Description

Interface summarising the associative recall functionality using stimulus elements and recall result elements of type `byte` or of type `decimal`.

## 5.32 LibTopoART.IFast\_Episodic\_recall Interface Reference

Interface summarising the episodic recall functionality using stimulus elements and recall result elements of type `byte` or of type `decimal`.

Inheritance diagram for LibTopoART.IFast\_Episodic\_recall:



#### Additional Inherited Members

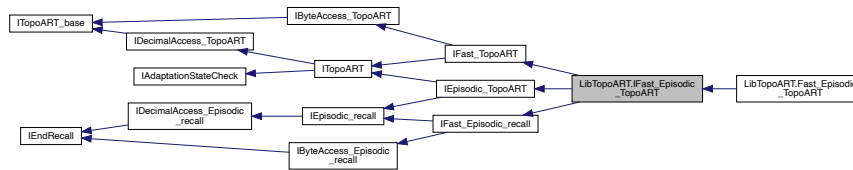
##### 5.32.1 Detailed Description

Interface summarising the episodic recall functionality using stimulus elements and recall result elements of type `byte` or of type `decimal`.

### 5.33 LibTopoART.IFast\_Episodic\_TopoART Interface Reference

Interface summarising the Episodic [TopoART](#) functionality including learning, prediction, episodic recall using input elements, stimulus elements, and recall result elements of type `byte` or of type `decimal` as well as adaptation state control.

Inheritance diagram for LibTopoART.IFast\_Episodic\_TopoART:



## Additional Inherited Members

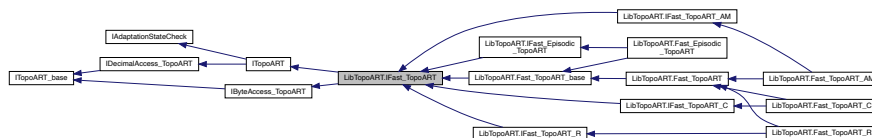
### 5.33.1 Detailed Description

Interface summarising the Episodic [TopoART](#) functionality including learning, prediction, episodic recall using input elements, stimulus elements, and recall result elements of type `byte` or of type `decimal` as well as adaptation state control.

## 5.34 LibTopoART.IFast\_TopoART Interface Reference

Interface summarising the [TopoART](#) functionality including learning and prediction using input elements of type `byte` or of type `decimal` as well as adaptation state control.

Inheritance diagram for LibTopoART.IFast\_TopoART:



## Additional Inherited Members

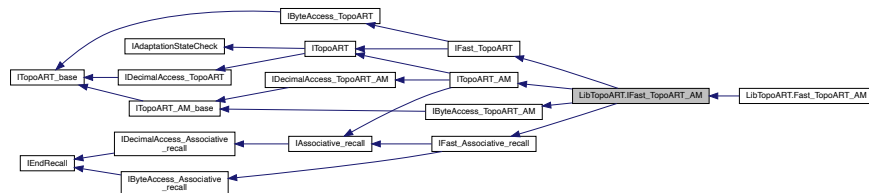
### 5.34.1 Detailed Description

Interface summarising the [TopoART](#) functionality including learning and prediction using input elements of type `byte` or of type `decimal` as well as adaptation state control.

### 5.35 LibTopoART.IFast\_TopopART\_AM Interface Reference

Interface summarising the Episodic **TopoART** functionality including learning, prediction, episodic recall using input elements, stimulus elements, and recall result elements of type `byte` or of type `decimal` as well as adaptation state control.

Inheritance diagram for LibTopoART.IFast\_TopoART\_AM:



### Additional Inherited Members

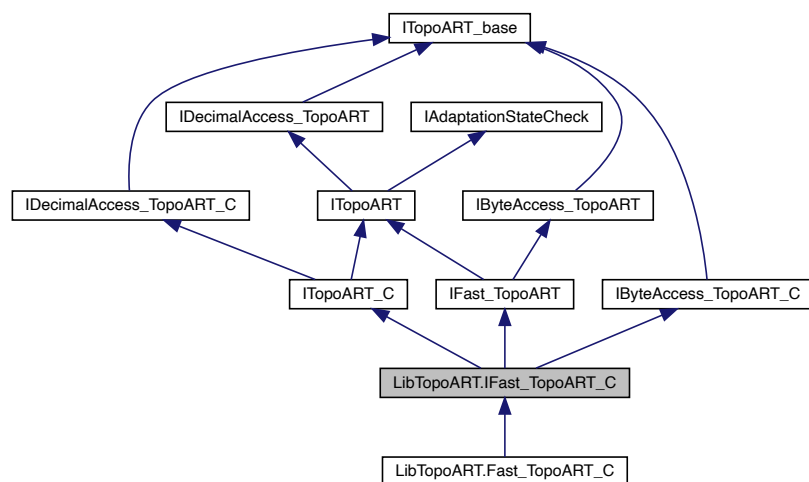
### 5.35.1 Detailed Description

Interface summarising the Episodic **TopoART** functionality including learning, prediction, episodic recall using input elements, stimulus elements, and recall result elements of type `byte` or of type `decimal` as well as adaptation state control.

### 5.36 LibTopoART.IFast\_TopoART\_C Interface Reference

Interface summarising the TopoART-C functionality including learning and prediction using input elements of type `byte` or of type `decimal` as well as adaptation state control.

Inheritance diagram for LibTopoART.IFast\_TopoART\_C:



## Additional Inherited Members

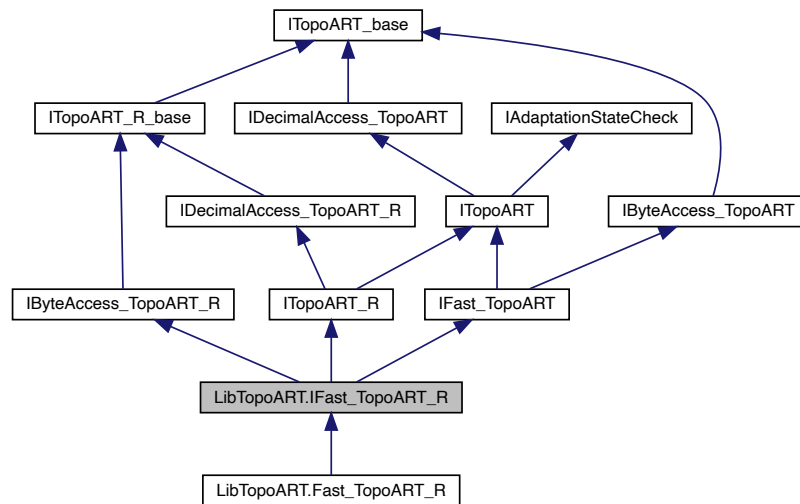
### 5.36.1 Detailed Description

Interface summarising the TopoART-C functionality including learning and prediction using input elements of type `byte` or of type `decimal` as well as adaptation state control.

## 5.37 LibTopoART.IFast\_TopoART\_R Interface Reference

Interface summarising the TopoART-R functionality including learning and prediction using input elements and output elements of type `byte` or of type `decimal` as well as adaptation state control.

Inheritance diagram for LibTopoART.IFast\_TopoART\_R:



## Additional Inherited Members

### 5.37.1 Detailed Description

Interface summarising the TopoART-R functionality including learning and prediction using input elements and output elements of type `byte` or of type `decimal` as well as adaptation state control.

## 5.38 LibTopoART.InvalidClassIDException Class Reference

Exception signalling an invalid class ID.

Inherits Exception.

#### 5.38.1 Detailed Description

Exception signalling an invalid class ID.

### 5.39 LibTopoART.InvalidFileException Class Reference

Exception signalling an invalid file.

Inherits Exception.

#### 5.39.1 Detailed Description

Exception signalling an invalid file.

### 5.40 LibTopoART.InvalidModuleIndexException Class Reference

Exception signalling an invalid module index.

Inherits Exception.

#### 5.40.1 Detailed Description

Exception signalling an invalid module index.

### 5.41 LibTopoART.InvalidNumberException Class Reference

Exception signalling an invalid number.

Inherits Exception.

#### 5.41.1 Detailed Description

Exception signalling an invalid number.

### 5.42 LibTopoART.InvalidSizeException Class Reference

Exception signalling an invalid size.

Inherits Exception.

#### 5.42.1 Detailed Description

Exception signalling an invalid size.

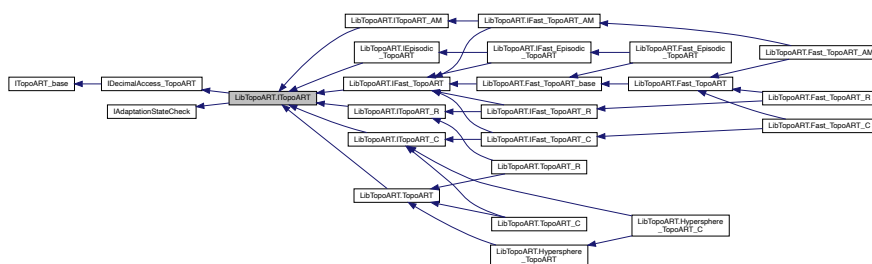
Inherits Exception.

### 5.43.1 Detailed Description

Exception signalling an invalid state of the neural network.

## 5.44 LibTopoART.ITopoART Interface Reference

Inheritance diagram for LibTopoART.ITopoART:

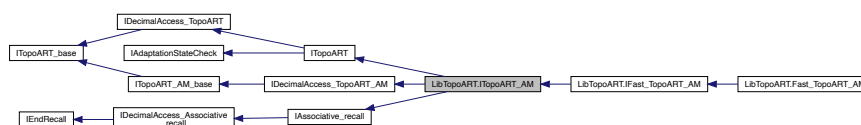


### Additional Inherited Members

### 5.44.1 Detailed Description

### 5.45 LibTopoART.ITopoART\_AM Interface Reference

Inheritance diagram for LibTopoART.ITopoART\_AM:





## Additional Inherited Members

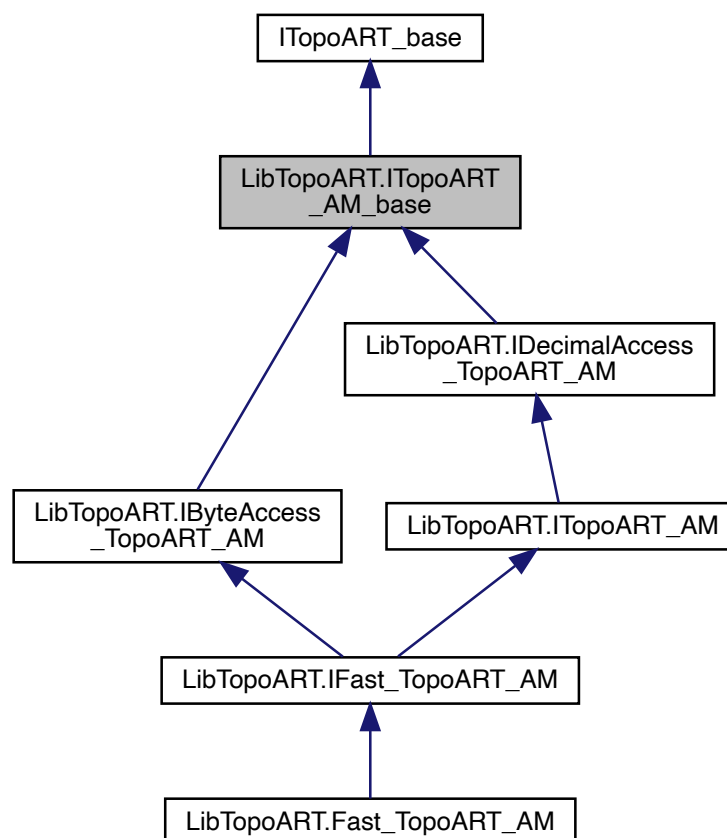
### 5.45.1 Detailed Description

Interface summarising the TopoART-AM functionality including learning, prediction, associative recall using input elements, stimulus elements, and recall result elements of type `decimal` as well as adaptation state control.

## 5.46 LibTopoART.ITopoART\_AM\_base Interface Reference

Interface summarising the basic TopoART-AM functionality excluding learning and prediction.

Inheritance diagram for LibTopoART.ITopoART\_AM\_base:



## Properties

- long `Key_1_len` [get]  
Property `K_1_len` returns the length of the first key vector.
- long `Key_2_len` [get]  
Property `K_2_len` returns the length of the second key vector.

Interface summarising the basic TopoART-AM functionality excluding learning and prediction.

Property `K_1_len` returns the length of the first key vector.

Property K 2 `len` returns the length of the second key vector.

Interface summarising the basic **TopoART** functionality excluding learning and prediction.

[illegible]

- void `ComputeClusterIDs` ()  
*This method computes the cluster IDs for all neurons.*
- void `SaveText` (string path)  
*This method saves the entire network as a text file.*
- void `Save` (string path, CompressionLevel compression=CompressionLevel.Fastest)  
*This method saves the entire network as a binary file.*

## Properties

- long[] [NodeNum](#) [get]  
*Property NodeNum represents the number of [TopoART](#) nodes used by each module.*
- long[] [ClusterNum](#) [get]  
*Property ClusterNum represents the number of [TopoART](#) clusters found by each module.*
- long [ModuleNum](#) [get]
- long [LearningSteps](#) [get]  
*Property LearningSteps represents the total number of performed learning steps.*
- decimal [Beta\\_sbm](#) [get, set]  
*Property Beta\_sbm represents the learning rate of the second best-matching nodes.*
- decimal [Rho\\_a](#) [get]  
*Property Rho\_a represents the vigilance parameter of the first [TopoART](#) module (TA a).*
- long [Tau](#) [get, set]  
*Property Tau represents the parameter tau required for the removal of nodes and edges.*
- long [Phi](#) [get, set]
- long[] [Phis](#) [get, set]
- decimal [Alpha](#) [get, set]  
*Property Alpha represents the choice parameter alpha.*

### 5.47.1 Detailed Description

Interface summarising the basic [TopoART](#) functionality excluding learning and prediction.

### 5.47.2 Member Function Documentation

#### 5.47.2.1 ComputeClusterIDs() `void LibTopoART.ITopoART_base.ComputeClusterIDs ( )`

This method computes the cluster IDs for all neurons.

Implemented in [LibTopoART.TopoART](#), and [LibTopoART.Fast\\_TopopART\\_base](#).

#### 5.47.2.2 Save() `void LibTopoART.ITopoART_base.Save ( string path, CompressionLevel compression = CompressionLevel.Fastest )`

This method saves the entire network as a binary file.

#### Parameters

<i>path</i>	A string representing the path of the file to save.
<i>compression</i>	Compression level of the save file (Compression is not supported by <a href="#">LibTopoART</a> v0.93 and below.)

Implemented in [LibTopoART.TopoART](#), and [LibTopoART.Fast\\_TopoART\\_base](#).

**5.47.2.3 SaveText()** `void LibTopoART.ITopoART_base.SaveText (`  
`string path )`

This method saves the entire network as a text file.

Parameters

<i>path</i>	A string representing the path of the file to save.
-------------	-----------------------------------------------------

Implemented in [LibTopoART.TopoART](#), and [LibTopoART.Fast\\_TopoART\\_base](#).

### 5.47.3 Property Documentation

**5.47.3.1 ModuleNum** `long LibTopoART.ITopoART_base.ModuleNum [get]`

Property `ModuleNum` represents the number of [TopoART](#) modules used. (The original [TopoART](#) uses two modules.)

**5.47.3.2 Phi** `long LibTopoART.ITopoART_base.Phi [get], [set]`

Property `Phi` represents the parameter `phi` required for the removal of nodes and edges as well as for the propagation of input to subsequent [TopoART](#) modules.

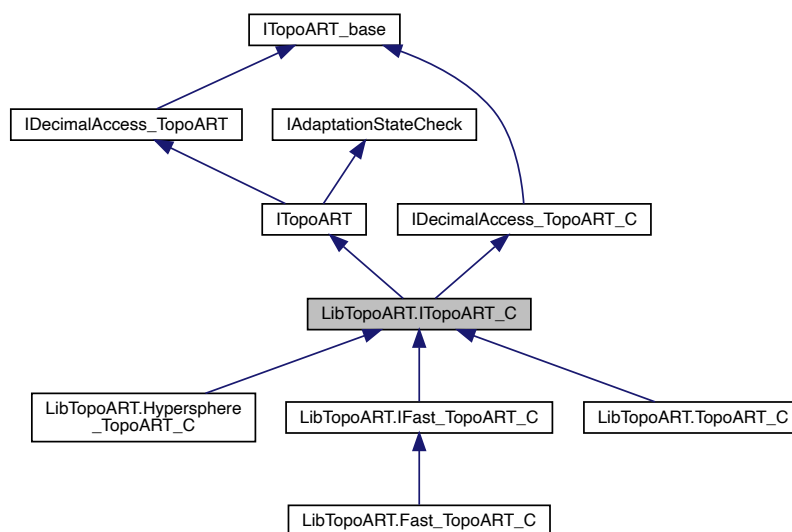
**5.47.3.3 Phis** `long [] LibTopoART.ITopoART_base.Phis [get], [set]`

Property `Phis` constitutes an extension of property `Phi` that enables individual values of `phi` for each module. By this, the removal of nodes and edges as well as for the propagation of input to subsequent [TopoART](#) modules can be controlled in a task dependent manner.

## 5.48 LibTopoART.ITopoART\_C Interface Reference

Interface summarising the TopoART-C functionality including learning and prediction using input elements of type `decimal` as well as adaptation state control.

Inheritance diagram for LibTopoART.ITopoART\_C:



### Additional Inherited Members

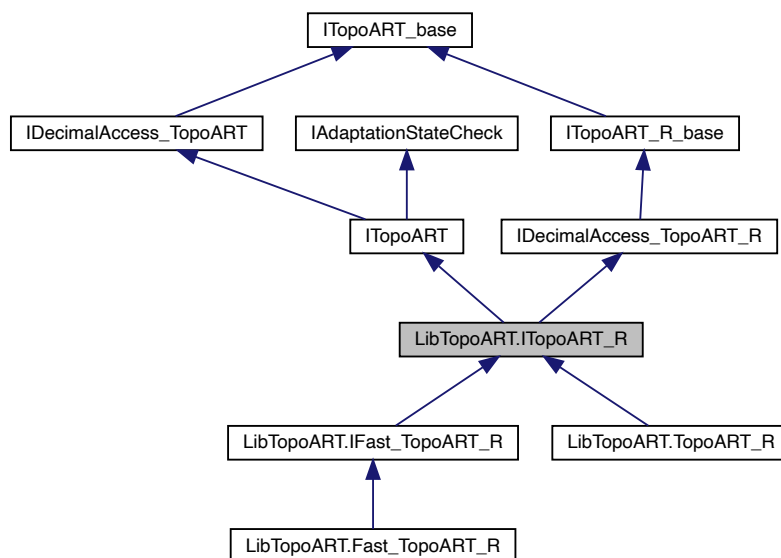
#### 5.48.1 Detailed Description

Interface summarising the TopoART-C functionality including learning and prediction using input elements of type `decimal` as well as adaptation state control.

### 5.49 LibTopoART.ITopoART\_R Interface Reference

Interface summarising the TopoART-R functionality including learning and prediction using input elements and output elements of type `decimal` as well as adaptation state control.

Inheritance diagram for LibTopoART.ITopoART\_R:



## Additional Inherited Members

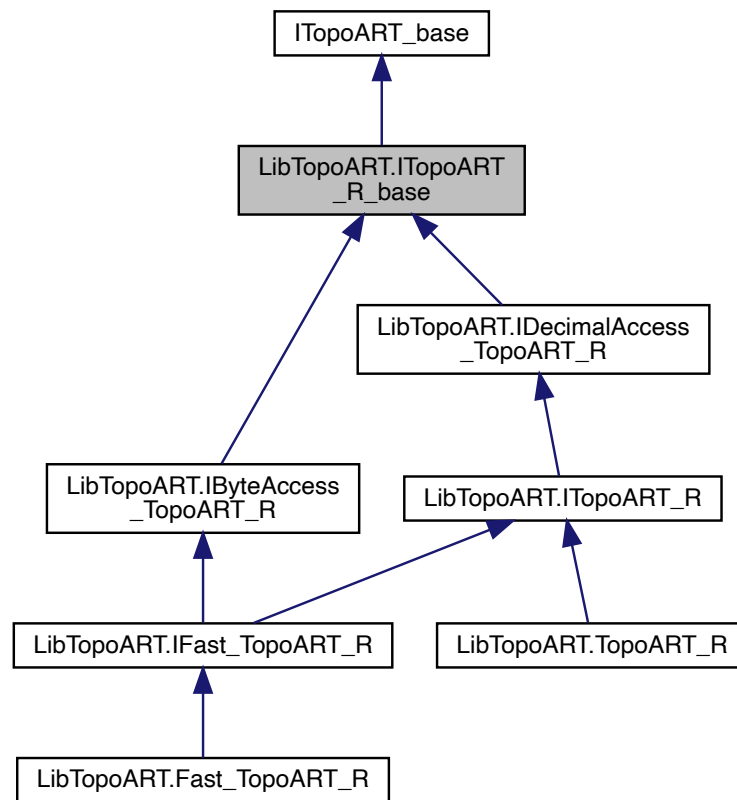
### 5.49.1 Detailed Description

Interface summarising the TopoART-R functionality including learning and prediction using input elements and output elements of type `decimal` as well as adaptation state control.

## 5.50 LibTopoART.ITopoART\_R\_base Interface Reference

Interface summarising the basic TopoART-R functionality excluding learning and prediction.

Inheritance diagram for LibTopoART.ITopoART\_R\_base:



## Properties

- long [D\\_len](#) [get]  
Property *D\_len* returns the length of the output vector (dependent variables).
- long [I\\_len](#) [get]  
Property *I\_len* returns the length of the input vector (independent variables).

## Additional Inherited Members

### 5.50.1 Detailed Description

Interface summarising the basic TopoART-R functionality excluding learning and prediction.

### 5.50.2 Property Documentation

**5.50.2.1 D\_len** `long LibTopoART.ITopoART_R_base.D_len [get]`

Property `D_len` returns the length of the output vector (dependent variables).

**5.50.2.2 I\_len** `long LibTopoART.ITopoART_R_base.I_len [get]`

Property `I_len` returns the length of the input vector (independent variables).

## 5.51 LibTopoART.LibTopoART\_control Struct Reference

Struct `LibTopoART_control` provides fields to control the general behaviour of `LibTopoART`.

### Static Public Attributes

- static `VerbosityLevel verbosity = VerbosityLevel.Verbose`  
*Instance variable `verbosity` enables controlling the number of messages issued by `LibTopoART`.*

### 5.51.1 Detailed Description

Struct `LibTopoART_control` provides fields to control the general behaviour of `LibTopoART`.

### 5.51.2 Member Data Documentation

**5.51.2.1 verbosity** `VerbosityLevel LibTopoART.LibTopoART_control.verbosity = VerbosityLevel.Verbose [static]`

Instance variable `verbosity` enables controlling the number of messages issued by `LibTopoART`.

## 5.52 LibTopoART.LibTopoART\_info Struct Reference

Struct `LibTopoART_info` provides some metainformation regarding the respective implementation of `LibTopoART`.

### Static Public Attributes

- const decimal `version = 0.94m`  
*Instance variable `version` represents the version of `LibTopoART`.*
- static readonly string[] `networks`  
*Instance variable `networks` provides a string array containing the networks implemented in the current version of `LibTopoART` and the corresponding class names.*
- const long `UNDEFINED = -1`  
*Instance variable `UNDEFINED` gives the value used for indicating undefined and uninitialised variables.*
- const long `FINAL_MODULE = UNDEFINED`  
*Instance variable `FINAL_MODULE` gives the value used for indicating that the `TopoART` module with the highest index is to be used.*



### 5.52.1 Detailed Description

Struct `LibTopoART_info` provides some metainformation regarding the respective implementation of `LibTopoART`.

### 5.52.2 Member Data Documentation

**5.52.2.1 FINAL\_MODULE** `const long LibTopoART.LibTopoART_info.FINAL_MODULE = UNDEFINED [static]`

Instance variable `FINAL_MODULE` gives the value used for indicating that the `TopoART` module with the highest index is to be used.

**5.52.2.2 networks** `readonly string [ ] LibTopoART.LibTopoART_info.networks [static]`

Initial value:

```
= {
    "Episodic TopoART (class Fast_Episodic_TopoART)",
    "Hypersphere TopoART (class Hypersphere_TopoART)",
    "Hypersphere TopoART-C (class Hypersphere_TopoART_C)",
    "TopoART (class TopoART, class Fast_TopoART)",
    "TopoART-AM (class Fast_TopoART-AM)",
    "TopoART-C (class TopoART_C, class Fast_TopoART_C)",
    "TopoART-R (class TopoART_R, class Fast_TopoART_R)"
}
```

Instance variable `networks` provides a string array containing the networks implemented in the current version of `LibTopoART` and the corresponding class names.

**5.52.2.3 UNDEFINED** `const long LibTopoART.LibTopoART_info.UNDEFINED = -1 [static]`

Instance variable `UNDEFINED` gives the value used for indicating undefined and uninitialised variables.

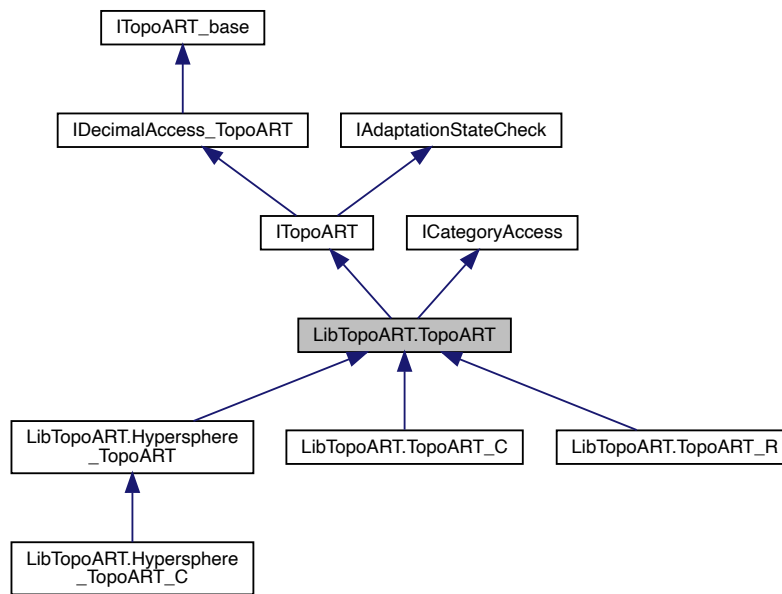
**5.52.2.4 version** `const decimal LibTopoART.LibTopoART_info.version = 0.94m [static]`

Instance variable `version` represents the version of `LibTopoART`.

### 5.53 LibTopoART.TopoART Class Reference

Class [TopoART](#) provides an implementation of the [TopoART](#) neural network as proposed in "Marko Tscherepanow (2010). [TopoART](#): A topology learning hierarchical ART network. In Proceedings of the International Conference on Artificial Neural Networks (ICANN), LNCS 6354, pp. 157–167. Berlin, Germany: Springer."

Inheritance diagram for LibTopoART.TopoART:



#### Public Member Functions

- [TopoART](#) (long input\_length, long module\_number, decimal rho\_a)  
*This constructor initialises a [TopoART](#) network.*
- [TopoART](#) (string path)  
*This constructor loads a saved [TopoART](#) network.*
- void [Dispose](#) ()  
*Releases all resources used by the [LibTopoART.TopoART](#) object.*
- void [ComputeClusterIDs](#) ()  
*This method computes the cluster IDs for all neurons.*
- [F2\\_output\[\] GetBMOutput](#) (decimal[] input)  
*This method finds the closest category for a given test input.*
- [F2\\_output\[\] GetBMOutput](#) (decimal[] input, bool[]? mask\_vector)  
*This method finds the closest category for a given test input.*
- virtual void [Learn](#) (decimal[] input)  
*This method performs a single training step.*
- void [SaveText](#) (string path)  
*This method saves the entire network as a text file.*
- void [Save](#) (string path, CompressionLevel compression=CompressionLevel.Fastest)  
*This method saves the entire network as a binary file.*
- void [ResetAdaptationState](#) ()

*This method resets the adaptation state to `AdaptationState.NO_ADAPTATION`.*

- [AdaptationState GetAdaptationState](#) (decimal epsilon=0.001m)

*This method returns the current adaptation state.*

- `List< CategoryInfo >? GetCategories` (long module\_index=[FINAL\\_MODULE](#))

*This method collects information on the categories of a specified module.*

### Static Public Attributes

- `const long FINAL\_MODULE = LibTopoART_info.FINAL_MODULE`

*Instance variable `FINAL_MODULE` gives the value used for indicating that the [TopoART](#) module with the highest index is to be used.*

### Properties

- decimal [Alpha](#) [get, set]

*Property `Alpha` represents the choice parameter alpha.*

- decimal [Beta\\_sbm](#) [get, set]

*Property `Beta_sbm` represents the learning rate of the second best-matching nodes.*

- long[] [ClusterNum](#) [get]

*Property `ClusterNum` represents the number of [TopoART](#) clusters found by each module.*

- long [InputLen](#) [get]

*Property `InputLen` returns the length of the input vector.*

- long [LearningSteps](#) [get]

*Property `LearningSteps` represents the total number of performed learning steps.*

- long [ModuleNum](#) [get]

*Property `ModuleNum` represents the number of [TopoART](#) modules used. (The original [TopoART](#) uses two modules.)*

- long[] [NodeNum](#) [get]

*Property `NodeNum` represents the number of [TopoART](#) nodes used by each module.*

- long [Phi](#) [get, set]

- long[] [Phis](#) [get, set]

- decimal [Rho\\_a](#) [get]

*Property `Rho_a` represents the vigilance parameter of the first [TopoART](#) module (TA a).*

- long [Tau](#) [get, set]

*Property `Tau` represents the parameter tau required for the removal of nodes and edges.*

- string [IntegerBaseType](#) = `Common.types[(int)integer_base_type_index]` [get]

*Property `IntegerBaseType` returns a string containing the data type used for representing integer variables (IDs, parameters, counters, etc.) internally.*

- decimal [FileFormatVersion](#) [get]

*Property `FileFormatVersion` returns the version of the file format used by class [TopoART](#).*

- string [FloatBaseType](#) = `Common.types[(int)float_base_type_index]` [get]

*Property `FloatBaseType` returns a string containing the data type used for representing floating point variables (input, weights, etc.) internally.*

- decimal [TopoARTFileFormatVersion](#) [get]

*Property `TopoARTFileFormatVersion` returns the version of the file format used by class [TopoART](#).*

#### 5.53.1 Detailed Description

Class [TopoART](#) provides an implementation of the [TopoART](#) neural network as proposed in "Marko Tscherepanow (2010). [TopoART](#): A topology learning hierarchical ART network. In Proceedings of the International Conference on Artificial Neural Networks (ICANN), LNCS 6354, pp. 157–167. Berlin, Germany: Springer."

Internally, real-valued data are stored in `decimal` variables. Hence, computations are rather slow but very accurate.

Class [TopoART](#) requires all input to lie in the interval [0,1].

## 5.53.2 Constructor & Destructor Documentation

**5.53.2.1 TopoART()** [1/2] `LibTopoART.TopoART.TopoART (`  
`long input_length,`  
`long module_number,`  
`decimal rho_a )`

This constructor initialises a [TopoART](#) network.

### Parameters

<i>input_length</i>	The length of input vectors to be learnt.
<i>module_number</i>	The number of <a href="#">TopoART</a> modules.
<i>rho_a</i>	The vigilance parameter of the first <a href="#">TopoART</a> module (TA a).

**5.53.2.2 TopoART()** [2/2] `LibTopoART.TopoART.TopoART (`  
`string path )`

This constructor loads a saved [TopoART](#) network.

### Parameters

<i>path</i>	The path of a binary <a href="#">TopoART</a> file.
-------------	----------------------------------------------------

### Exceptions

<a href="#">InvalidFileException</a>	Throws when the given file cannot be loaded.
--------------------------------------	----------------------------------------------

## 5.53.3 Member Function Documentation

**5.53.3.1 ComputeClusterIDs()** `void LibTopoART.TopoART.ComputeClusterIDs ( )`

This method computes the cluster IDs for all neurons.

Implements [LibTopoART.ITopoART\\_base](#).

**5.53.3.2 Dispose()** `void LibTopoART.TopoART.Dispose ( )`

Releases all resources used by the [LibTopoART.TopoART](#) object.

Call [Dispose\(\)](#) when you are finished using the [LibTopoART.TopoART](#). The [Dispose\(\)](#) method leaves the [LibTopoART.TopoART](#) in an unusable state. After calling [Dispose\(\)](#), you must release all references to the [LibTopoART.TopoART](#) so the garbage collector can reclaim the memory that the [LibTopoART.TopoART](#) was occupying.

**5.53.3.3 GetAdaptationState()** `AdaptationState LibTopoART.TopoART.GetAdaptationState ( decimal epsilon = 0.001m )`

This method returns the current adaptation state.

**Parameters**

<i>epsilon</i>	The threshold for weight adaptations to be considered.
----------------	--------------------------------------------------------

**Returns**

An enumeration describing the adaptation state.

**Exceptions**

<a href="#">InvalidStateException</a>	Throws when the network is in an invalid state.
<a href="#">InvalidNumberException</a>	Throws when the number of edges of an F2 node is greater than <code>int.MaxValue</code> .

Implements [LibTopoART.IAdaptationStateCheck](#).

**5.53.3.4 GetBMOutput()** `[1/2] F2_output [ ] LibTopoART.TopoART.GetBMOutput ( decimal[] input )`

This method finds the closest category for a given test input.

**Parameters**

<i>input</i>	The input vector x(t).
--------------	------------------------

**Returns**

An array of type [F2\\_output](#). Each entry contains the ID of the best-matching node and the corresponding cluster ID for one [TopoART](#) module.

Implements [LibTopoART.IDecimalAccess\\_TopoART](#).

**5.53.3.5 GetBMOutput()** [2/2] `F2_output [ ] LibTopoART.TopoART.GetBMOutput ( decimal[] input, bool?[] mask_vector )`

This method finds the closest category for a given test input.

#### Parameters

<i>input</i>	The input vector $x(t)$ .
<i>mask_vector</i>	A mask vector excluding individual dimensions of $x(t)$ from the computation. (Setting an element of the mask vector to <code>true</code> , excludes the corresponding elements of $x(t)$ .)

#### Returns

An array of type `F2_output`. Each entry contains the ID of the best-matching node and the corresponding cluster ID for one `TopoART` module.

**5.53.3.6 GetCategories()** `List<CategoryInfo>? LibTopoART.TopoART.GetCategories ( long module_index = FINAL_MODULE )`

This method collects information on the categories of a specified module.

#### Parameters

<i>module_index</i>	The index of the module the categories of which are to be analysed.
---------------------	---------------------------------------------------------------------

#### Returns

A list containing information about the respective categories.

#### Exceptions

<a href="#"><i>InvalidModuleIndexException</i></a>	Throws when <i>module_index</i> is invalid.
<a href="#"><i>InvalidNumberException</i></a>	Throws when the number of nodes of a module is greater than <code>int.MaxValue</code> .

Implements [LibTopoART.ICategoryAccess](#).

**5.53.3.7 Learn()** `virtual void LibTopoART.TopoART.Learn ( decimal[] input ) [virtual]`

This method performs a single training step.

#### Parameters

<i>input</i>	The input vector to be learnt.
--------------	--------------------------------

Implements [LibTopoART.IDecimalAccess\\_TopoART](#).

Reimplemented in [LibTopoART.TopoART\\_R](#), [LibTopoART.TopoART\\_C](#), and [LibTopoART.Hypersphere\\_TopoART\\_C](#).

### 5.53.3.8 ResetAdaptationState() `void LibTopoART.TopoART.ResetAdaptationState ( )`

This method resets the adaptation state to `AdaptationState.NO_ADAPTATION`.

#### Exceptions

<a href="#">InvalidNumberException</a>	Throws when the number of edges of an F2 node is greater than <code>int.MaxValue</code> .
----------------------------------------	-------------------------------------------------------------------------------------------

Implements [LibTopoART.IAdaptationStateCheck](#).

### 5.53.3.9 Save() `void LibTopoART.TopoART.Save (` `string path,` `CompressionLevel compression = CompressionLevel.Fastest )`

This method saves the entire network as a binary file.

#### Parameters

<i>path</i>	A <code>string</code> representing the path of the file to save.
<i>compression</i>	Compression level of the save file (Compression is not supported by <a href="#">LibTopoART</a> v0.93 and below.)

Implements [LibTopoART.ITopoART\\_base](#).

### 5.53.3.10 SaveText() `void LibTopoART.TopoART.SaveText (` `string path )`

This method saves the entire network as a text file.

#### Parameters

<i>path</i>	A <code>string</code> representing the path of the file to save.
-------------	------------------------------------------------------------------

Implements [LibTopoART.ITopoART\\_base](#).

## 5.53.4 Member Data Documentation

**5.53.4.1 FINAL\_MODULE** `const long LibTopoART.TopoART.FINAL_MODULE = LibTopoART_info.FINAL_MODULE [static]`

Instance variable `FINAL_MODULE` gives the value used for indicating that the [TopoART](#) module with the highest index is to be used.

### 5.53.5 Property Documentation

**5.53.5.1 FileFormatVersion** `decimal LibTopoART.TopoART.FileFormatVersion [get]`

Property `FileFormatVersion` returns the version of the file format used by class [TopoART](#).

**5.53.5.2 FloatBaseType** `string LibTopoART.TopoART.FloatBaseType = Common.types[(int)float_base_type_index] [get]`

Property `FloatBaseType` returns a string containing the data type used for representing floating point variables (input, weights, etc.) internally.

**5.53.5.3 IntegerBaseType** `string LibTopoART.TopoART.IntegerBaseType = Common.types[(int)integer_base_type_index] [get]`

Property `IntegerBaseType` returns a string containing the data type used for representing integer variables (IDs, parameters, counters, etc.) internally.

**5.53.5.4 Phi** `long LibTopoART.TopoART.Phi [get], [set]`

Property `Phi` represents the parameter `phi` required for the removal of nodes and edges as well as for the propagation of input to subsequent [TopoART](#) modules.

**5.53.5.5 Phis** `long [] LibTopoART.TopoART.Phis [get], [set]`

Property `Phis` constitutes an extension of property `Phi` that enables individual values of `phi` for each module. By this, the removal of nodes and edges as well as for the propagation of input to subsequent [TopoART](#) modules can be controlled in a task dependent manner.

#### Exceptions

<a href="#">InvalidSizeException</a>	Throws when the array length does not fit the module number.
--------------------------------------	--------------------------------------------------------------



**5.53.5.6 TopoARTFileFormatVersion** decimal LibTopoART.TopoART.TopoARTFileFormatVersion [get]

Property `TopoARTFileFormatVersion` returns the version of the file format used by class `TopoART`.

## 5.54 LibTopoART\_samples.TopoART\_AM\_sample1 Class Reference

Sample using TopoART-AM with synthetic two-dimensional data. [C#]

### 5.54.1 Detailed Description

Sample using TopoART-AM with synthetic two-dimensional data. [C#]

A TopoART-AM network is trained with the well-known Two Spirals dataset augmented with additional information. The resulting network maps two-dimensional points lying on each spiral (`key_1`) to their Euclidean distance from the origin and the corresponding spiral ID (`key_2`). Therefore, it can recall spiral points if a distance and a spiral ID are given, and vice versa.

The resulting network can be visualised using the R script `ShowTopoARTAMResults.R` provided in the sub-folder `R`.

## 5.55 LibTopoART\_samples.TopoART\_AM\_sample2 Class Reference

Learning of bidirectional associations between images. [F#]

### 5.55.1 Detailed Description

Learning of bidirectional associations between images. [F#]

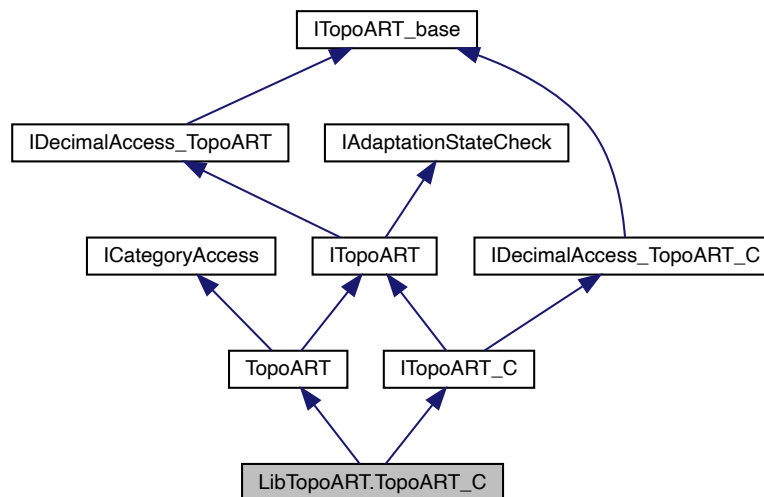
Similar to Section 4.2 of "Marko Tscherepanow, Marco Kortkamp, and Marc Kammer (2011). A Hierarchical ART Network for the Stable Incremental Learning of Topological Structures and Associations from Noisy Data. Neural Networks 24(8): 906-916. Elsevier.", a TopoART-AM network is trained with real-world image data. There are two kinds of images grouped into owners and objects. TopoART-AM learns a bi-directional mapping between images of these two groups. Each image has a size of about 34500 pixels. As each pixel comprises 3 color channels (RGB) and the input vector encompasses a key from each group, the total length of the input vector is larger than 200,000. After finishing training, recall is performed for a single input stimulus of each group. The recall results are saved in the folder `results/recall/ObjectsOwners_dataset_recall_results`.

/summary>

## 5.56 LibTopoART.TopoART\_C Class Reference

Class [TopoART\\_C](#) provides an implementation of the TopoART-C neural network as proposed in "Marko Tscherepanow and Sören Riechers (2012). An Incremental On-line Classifier for Imbalanced, Incomplete, and Noisy Data. In Proceedings of the European Conference on Artificial Intelligence (ECAI), Workshop on Active and Incremental Learning (AIL), pp. 18-23. Montpellier, France."

Inheritance diagram for LibTopoART.TopoART\_C:



### Public Member Functions

- [TopoART\\_C](#) (long input\_length, long module\_number, decimal rho\_a)  
*This constructor initialises a TopoART-C network.*
- [TopoART\\_C](#) (string path)  
*This constructor loads a saved TopoART-C network.*
- override void [Learn](#) (decimal[] input)  
*This method performs a single training step and sets the class ID corresponding to input to `UNDEFINED_CLASS_ID`.*
- void [Learn](#) (decimal[] input, long class\_ID)  
*This method performs a single training step.*
- long [Predict](#) (decimal[] input, long nu)  
*This method predicts the class ID.*
- [TopoART\\_C\\_Prediction Predict](#) (decimal[] input, bool[]? mask\_vector, long nu)  
*This method predicts the class ID.*

### Static Public Attributes

- const long [UNDEFINED\\_CLASS\\_ID](#) = -2  
*Instance variable `UNDEFINED_CLASS_ID` gives the value used for indicating that an input sample was predict to belong to the undefined class; i.e., no class ID was provided for such input samples during training.*

## Properties

- new decimal [FileFormatVersion](#) [get]

Property *FileFormatVersion* returns the version of the file format used by class *TopoART\_C*.

### 5.56.1 Detailed Description

Class [TopoART\\_C](#) provides an implementation of the TopoART-C neural network as proposed in "Marko Tscherepanow and Sören Riechers (2012). An Incremental On-line Classifier for Imbalanced, Incomplete, and Noisy Data. In Proceedings of the European Conference on Artificial Intelligence (ECAI), Workshop on Active and Incremental Learning (AIL), pp. 18-23. Montpellier, France."

Class [TopoART\\_C](#) requires all input except the class IDs to lie in the interval [0,1]. The class IDs are signed integer values.

### 5.56.2 Constructor & Destructor Documentation

**5.56.2.1 TopoART\_C() [1/2]** `LibTopoART.TopoART_C.TopoART_C (`  
     long *input\_length*,  
     long *module\_number*,  
     decimal *rho\_a* )

This constructor initialises a TopoART-C network.

#### Parameters

<i>input_length</i>	The length of input vectors to be learnt.
<i>module_number</i>	The number of TopoART-C modules.
<i>rho_a</i>	The vigilance parameter of the first TopoART-C module (TopoART-C a).

**5.56.2.2 TopoART\_C() [2/2]** `LibTopoART.TopoART_C.TopoART_C (`  
     string *path* )

This constructor loads a saved TopoART-C network.

#### Parameters

<i>path</i>	The path of a binary TopoART-C file.
-------------	--------------------------------------

#### Exceptions

<a href="#">InvalidFileException</a>	Throws when the given file cannot be loaded.
--------------------------------------	----------------------------------------------

### 5.56.3 Member Function Documentation

**5.56.3.1 Learn()** [1/2] `override void LibTopoART.TopoART_C.Learn ( decimal[] input ) [virtual]`

This method performs a single training step and sets the class ID corresponding to *input* to UNDEFINED\_CLASS\_ID.

#### Parameters

<i>input</i>	The input vector to be learnt.
--------------	--------------------------------

Reimplemented from [LibTopoART.TopoART](#).

**5.56.3.2 Learn()** [2/2] `void LibTopoART.TopoART_C.Learn ( decimal[] input, long class_ID )`

This method performs a single training step.

#### Parameters

<i>input</i>	The input vector to be learnt.
<i>class_ID</i>	The class ID corresponding to <i>input</i> . (must be equal to or larger than 0)

#### Exceptions

<a href="#">InvalidClassIDException</a>	Throws when <i>class_ID</i> is less than 0.
-----------------------------------------	---------------------------------------------

Implements [LibTopoART.IDecimalAccess\\_Top ART\\_C](#).

**5.56.3.3 Predict()** [1/2] `TopoART_C_Prediction LibTopoART.TopoART_C.Predict ( decimal[] input, bool?[] mask_vector, long nu )`

This method predicts the class ID.

#### Parameters

<i>input</i>	The input vector the class ID of which is to be predicted.
<i>mask_vector</i>	The mask vector corresponding to <i>input</i> .
<i>nu</i>	The maximum cardinality of the set of enclosing categories E and the neighbourhood set N. (This parameter does not modify the network. It may be arbitrarily changed in each prediction step.)

**Returns**

An object of type [TopoART\\_C\\_Prediction](#) containing the predicted class ID and a corresponding confidence value.

**5.56.3.4 Predict()** [2/2] `long LibTopoART.TopoART_C.Predict (`  
`decimal[] input,`  
`long nu )`

This method predicts the class ID.

**Parameters**

<i>input</i>	The input vector the class ID of which is to be predicted.
<i>nu</i>	The maximum cardinality of the set of enclosing categories E and the neighbourhood set N. (This parameter does not modify the network. It may be arbitrarily changed in each prediction step.)

**Returns**

The predicted class ID.

Implements [LibTopoART.IDecimalAccess\\_TopoART\\_C](#).

**5.56.4 Member Data Documentation**

**5.56.4.1 UNDEFINED\_CLASS\_ID** `const long LibTopoART.TopoART_C.UNDEFINED_CLASS_ID = -2 [static]`

Instance variable `UNDEFINED_CLASS_ID` gives the value used for indicating that an input sample was predict to belong to the undefined class; i.e, no class ID was provided for such input samples during training.

**5.56.5 Property Documentation**

**5.56.5.1 FileFormatVersion** `new decimal LibTopoART.TopoART_C.FileFormatVersion [get]`

Property `FileFormatVersion` returns the version of the file format used by class [TopoART\\_C](#).

**5.57 LibTopoART.TopoART\_C\_Prediction Struct Reference**

Struct [TopoART\\_C\\_Prediction](#) contains a prediction made by a TopoART-C network.

## Public Member Functions

- [TopoART\\_C\\_Prediction](#) (long [class\\_ID](#), decimal [confidence](#))

*This constructor sets the instance variables `class_ID` and `confidence` of struct [TopoART\\_C\\_Prediction](#).*

## Public Attributes

- readonly long [class\\_ID](#)

*Instance variable `class_ID` gives the predicted class ID.*

- readonly decimal [confidence](#)

*Instance variable `confidence` provides a confidence for the predicted class ID.*

### 5.57.1 Detailed Description

Struct [TopoART\\_C\\_Prediction](#) contains a prediction made by a TopoART-C network.

### 5.57.2 Constructor & Destructor Documentation

**5.57.2.1 [TopoART\\_C\\_Prediction\(\)](#)** `LibTopoART.TopoART_C_Prediction.TopoART_C_Prediction (`  
    long `class_ID`,  
    decimal `confidence` )

This constructor sets the instance variables `class_ID` and `confidence` of struct [TopoART\\_C\\_Prediction](#).

#### Parameters

<code>class_ID</code>	The class ID to be set.
<code>confidence</code>	The value of the confidence to be set.

### 5.57.3 Member Data Documentation

**5.57.3.1 `class_ID`**   readonly long `LibTopoART.TopoART_C_Prediction.class_ID`

Instance variable `class_ID` gives the predicted class ID.

**5.57.3.2 `confidence`**   readonly decimal `LibTopoART.TopoART_C_Prediction.confidence`

Instance variable `confidence` provides a confidence for the predicted class ID.

## 5.58 LibTopoART\_samples.TopoART\_C\_sample1 Class Reference

Simple classification sample. [C#]

### 5.58.1 Detailed Description

Simple classification sample. [C#]

This sample demonstrates training and several possibilities for prediction at the example of a simple classification task.

## 5.59 LibTopoART\_samples.TopoART\_C\_sample2 Class Reference

Classification sample using more complex synthetic two-dimensional data with associated class IDs. [C#]

### 5.59.1 Detailed Description

Classification sample using more complex synthetic two-dimensional data with associated class IDs. [C#]

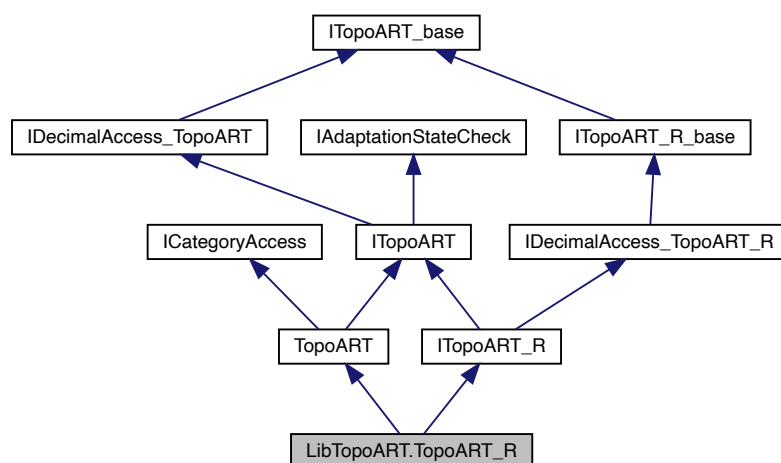
Train TopoART-C with a two-dimensional dataset similar to the one used in "Marko Tscherepanow and Sören Riechers (2012). An Incremental On-line Classifier for Imbalanced, Incomplete, and Noisy Data. In Proceedings of the European Conference on Artificial Intelligence (ECAI), Workshop on Active and Incremental Learning (AIL), pp. 18-23. Montpellier, France." This dataset comprises six clusters (each containing 15,000 samples) as well as 10,000 noise samples which were mixed randomly. The samples are divided into two classes. Each sample belonging to one of the six clusters is assigned a class ID depending on its position. In contrast, noise samples receive a random class ID.

The resulting neural network can be visualised using the R script `ShowTopoARTCResults.R` or the R script `ShowHypersphereTopoARTCResults.R`, respectively. Both R scripts are provided in the subfolder R.

## 5.60 LibTopoART.TopoART\_R Class Reference

Class `TopoART_R` provides an implementation of the TopoART-R neural network as proposed in "Marko Tscherepanow (2011). An Extended `TopoART` Network for the Stable On-Line Learning of Regression Functions. In Proceedings of the International Conference on Neural Information Processing (ICONIP), LNCS 7063, pp. 562–571. Berlin, Germany: Springer."

Inheritance diagram for LibTopoART.TopoART\_R:



## Public Member Functions

- [TopoART\\_R](#) (long i\_length, long d\_length, long module\_number, decimal rho\_a)  
*This constructor initialises a TopoART-R network.*
- [TopoART\\_R](#) (string path)  
*This constructor loads a saved TopoART-R network.*
- override void [Learn](#) (decimal[] input)  
*This method performs a single training step. The independent variables and the dependent variables are automatically separated.*
- void [Learn](#) (decimal[] i\_vec, decimal[] d\_vec)  
*This method performs a single training step.*
- decimal[] [Predict](#) (decimal[] i\_vec, long nu=10)  
*This method predicts the dependent variables.*
- [TopoART\\_R\\_Prediction](#)< decimal > [Predict](#) (decimal[] i\_vec, bool[] m\_i\_vec, long nu=10)  
*This method predicts the dependent variables for a given set of independent variables. Unknown values of independent variables can be signified by setting the corresponding value of m\_i\_vec to true.*

## Properties

- long [D\\_len](#) [get]  
*Property D\_len returns the length of the output vector (dependent variables).*
- new decimal [FileFormatVersion](#) [get]  
*Property FileFormatVersion returns the version of the file format used by class TopoART\_R.*
- long [I\\_len](#) [get]  
*Property I\_len returns the length of the input vector (independent variables).*

## Additional Inherited Members

### 5.60.1 Detailed Description

Class [TopoART\\_R](#) provides an implementation of the TopoART-R neural network as proposed in "Marko Tscherepanow (2011). An Extended [TopoART](#) Network for the Stable On-Line Learning of Regression Functions. In Proceedings of the International Conference on Neural Information Processing (ICONIP), LNCS 7063, pp. 562–571. Berlin, Germany: Springer."

Class [TopoART\\_R](#) requires all input and output to lie in the interval [0,1].

### 5.60.2 Constructor & Destructor Documentation

**5.60.2.1 TopoART\_R() [1/2]** `LibTopoART.TopoART_R.TopoART_R (`  
     long i\_length,  
     long d\_length,  
     long module\_number,  
     decimal rho\_a )

This constructor initialises a TopoART-R network.



## Parameters

<i>i_length</i>	The length of the input vector (independent variables) to be learnt.
<i>d_length</i>	The length of the output vector (dependent variables) to be learnt.
<i>module_number</i>	The number of TopoART-R modules.
<i>rho_a</i>	The vigilance parameter of the first TopoART-R module (TopoART-R a).

### 5.60.2.2 TopoART\_R() [2/2] `LibTopoART.TopoART_R.TopoART_R (string path )`

This constructor loads a saved TopoART-R network.

## Parameters

<i>path</i>	The path of a binary TopoART-R file.
-------------	--------------------------------------

## Exceptions

<a href="#"><i>InvalidFileException</i></a>	Throws when the given file cannot be loaded.
---------------------------------------------	----------------------------------------------

## 5.60.3 Member Function Documentation

### 5.60.3.1 Learn() [1/2] `void LibTopoART.TopoART_R.Learn (decimal[] i_vec, decimal[] d_vec )`

This method performs a single training step.

## Parameters

<i>i_vec</i>	The input vector (independent variables) to be learnt.
<i>d_vec</i>	The output vector (dependent variables) corresponding to <i>i_vec</i> .

Implements [LibTopoART.IDecimalAccess\\_TopoART\\_R](#).

### 5.60.3.2 Learn() [2/2] `override void LibTopoART.TopoART_R.Learn (decimal[] input ) [virtual]`

This method performs a single training step. The independent variables and the dependent variables are automatically separated.

## Parameters

<i>input</i>	The input vector to be learnt.
--------------	--------------------------------

Reimplemented from [LibTopoART.TopoART](#).

**5.60.3.3 Predict()** [1/2] [TopoART\\_R\\_Prediction](#)<decimal> LibTopoART.TopoART\_R.Predict (   
     decimal[] *i\_vec*,  
     bool[] *m\_i\_vec*,  
     long *nu* = 10 )

This method predicts the dependent variables for a given set of independent variables. Unknown values of independent variables can be signified by setting the corresponding value of *m\_i\_vec* to `true`.

## Parameters

<i>i_vec</i>	The input vector (independent variables).
<i>m_i_vec</i>	The mask vector corresponding to <i>i_vec</i> .
<i>nu</i>	The maximum cardinality of the neighbourhood set N. (In the original TopoART-R network, nu is fixed to 10. But task-specific adaptations might lead to an improved prediction accuracy. This parameter does not alter the network. It may be arbitrarily changed in each prediction step.)

## Returns

An object of type [TopoART\\_R\\_Prediction](#) containing the predicted values for the unknown independent variables and all dependent variables.

Implements [LibTopoART.IDecimalAccess\\_TopoART\\_R](#).

**5.60.3.4 Predict()** [2/2] decimal [] LibTopoART.TopoART\_R.Predict (   
     decimal[] *i\_vec*,  
     long *nu* = 10 )

This method predicts the dependent variables.

## Parameters

<i>i_vec</i>	The input vector (independent variables).
<i>nu</i>	The maximum cardinality of the neighbourhood set N. (In the original TopoART-R network, nu is fixed to 10. But task-specific adaptations might lead to an improved prediction accuracy. This parameter does not modify the network. It may be arbitrarily changed in each prediction step.)

## Returns

The predicted values for all dependent variables.

Implements [LibTopoART.IDecimalAccess\\_TopoART\\_R](#).

#### 5.60.4 Property Documentation

**5.60.4.1 D\_len** `long LibTopoART.TopoART_R.D_len [get]`

Property `D_len` returns the length of the output vector (dependent variables).

**5.60.4.2 FileFormatVersion** `new decimal LibTopoART.TopoART_R.FileFormatVersion [get]`

Property `FileFormatVersion` returns the version of the file format used by class [TopoART\\_R](#).

**5.60.4.3 I\_len** `long LibTopoART.TopoART_R.I_len [get]`

Property `I_len` returns the length of the input vector (independent variables).

### 5.61 LibTopoART.TopoART\_R\_Prediction<\_ElementType> Struct Template Reference

Struct [TopoART\\_R\\_Prediction](#) contains a prediction made by a TopoART-R network.

#### Public Member Functions

- [TopoART\\_R\\_Prediction](#) (`_ElementType[] i_vec_prediction, _ElementType[] d_vec_prediction`)  
*This constructor sets the instance variables `i_vec_prediction` and `d_vec_prediction` of struct [TopoART\\_R\\_Prediction](#).*
- `void PrintPredictions ()`  
*This method prints the predictions on the console.*

#### Public Attributes

- readonly `_ElementType` [NO\\_PREDICTION](#)  
*Instance variable `NO_PREDICTION` provides a default prediction for variables that are presented to the network; i.e., these variables are known and no prediction is computed for them. ATTENTION: `NO_PREDICTION` may be ambiguous depending on `_ElementType`.*
- readonly `_ElementType[]` [i\\_vec\\_prediction](#)  
*Instance variable `i_vec_prediction` represents predictions for unknown independent variables.*
- readonly `_ElementType[]` [d\\_vec\\_prediction](#)  
*Instance variable `d_vec_prediction` provides the predictions for the dependent variables.*

### 5.61.1 Detailed Description

Struct [TopoART\\_R\\_Prediction](#) contains a prediction made by a TopoART-R network.

#### Type Constraints

***\_ElementType* : struct**  
***\_ElementType* : IConvertible**

### 5.61.2 Constructor & Destructor Documentation

**5.61.2.1 TopoART\_R\_Prediction()** [LibTopoART.TopoART\\_R\\_Prediction](#)< [\\_ElementType](#) >.[TopoART\\_R\\_Prediction](#)  
(  
    [\\_ElementType](#)[] *i\_vec\_prediction*,  
    [\\_ElementType](#)[] *d\_vec\_prediction* )

This constructor sets the instance variables *i\_vec\_prediction* and *d\_vec\_prediction* of struct [TopoART\\_R\\_Prediction](#).

#### Parameters

<i>i_vec_prediction</i>	The prediction results for the independent variables to be set.
<i>d_vec_prediction</i>	The prediction results for the dependent variables to be set.

### 5.61.3 Member Function Documentation

**5.61.3.1 PrintPredictions()** void [LibTopoART.TopoART\\_R\\_Prediction](#)< [\\_ElementType](#) >.[PrintPredictions](#)  
( )

This method prints the predictions on the console.

### 5.61.4 Member Data Documentation

**5.61.4.1 d\_vec\_prediction** readonly [\\_ElementType](#) [] [LibTopoART.TopoART\\_R\\_Prediction](#)< [\\_ElementType](#) >.[d\\_vec\\_prediction](#)

Instance variable *d\_vec\_prediction* provides the predictions for the dependent variables.

**5.61.4.2 i\_vec\_prediction** readonly \_ElementType [] [LibTopoART.TopoART\\_R\\_Prediction](#)< \_ElementType >.i\_vec\_prediction

Instance variable `i_vec_prediction` represents predictions for unknown independent variables.

**5.61.4.3 NO\_PREDICTION** readonly \_ElementType [LibTopoART.TopoART\\_R\\_Prediction](#)< \_ElementType >.NO\_PREDICTION

Instance variable `NO_PREDICTION` provides a default prediction for variables that are presented to the network; i.e., these variables are known and no prediction is computed for them. ATTENTION: `NO_PREDICTION` may be ambiguous depending on `_ElementType`.

## 5.62 LibTopoART\_samples.TopoART\_R\_sample1 Class Reference

Regression sample using TopoART-R. (simplified version) [C#]

### 5.62.1 Detailed Description

Regression sample using TopoART-R. (simplified version) [C#]

This sample trains a TopoART-R network with 100 points sampled from a sine function. Then, sine values are predicted for 25 random values.

The predicted results can be visualised using the R script `ShowTopoARTResults.R` provided in the subfolder `R`.

## 5.63 LibTopoART\_samples.TopoART\_R\_sample2 Class Reference

Regression sample using TopoART-R. (advanced version) [C#]

### 5.63.1 Detailed Description

Regression sample using TopoART-R. (advanced version) [C#]

This sample trains a TopoART-R network with 100 points sampled from a sine function. Then, sine values are predicted for 25 random values.

The predicted results can be visualised using the R script `ShowTopoARTResults.R` provided in the subfolder `R`.

## 5.64 LibTopoART\_samples.TopoART\_R\_sample3 Class Reference

Pixel-wise regression analysis of an image using TopoART-R. [F#]

### 5.64.1 Detailed Description

Pixel-wise regression analysis of an image using TopoART-R. [F#]

This sample trains a TopoART-R network with a colour image depicting the Mandelbrot set. The pixel coordinates are used as independent variables and the corresponding colour values as dependent variables. The accuracy of the regression function can be controlled by means of the vigilance parameter `rho_a`.

The training image `Mandelbrot_reference.png` and the predicted image `TopoART-R_Mandelbrot_regression.png` are written into the subfolder `results/regression`.

## 5.65 LibTopoART\_samples.TopoART\_sample1 Class Reference

Simple clustering sample. [C#]

### 5.65.1 Detailed Description

Simple clustering sample. [C#]

First, a dataset comprised of 10 samples is learned by a TopoART network. Afterwards, the training samples are slightly modified by random values and used for predicting cluster labels.

## 5.66 LibTopoART\_samples.TopoART\_sample2 Class Reference

Clustering sample using more complex synthetic two-dimensional data. [C#]

### 5.66.1 Detailed Description

Clustering sample using more complex synthetic two-dimensional data. [C#]

Train TopoART or Hypersphere TopoART with a two-dimensional dataset similar to the one used in "Marko Tscherepanow and Sören Riechers (2012). An Incremental On-line Classifier for Imbalanced, Incomplete, and Noisy Data. In Proceedings of the European Conference on Artificial Intelligence (ECAI), Workshop on Active and Incremental Learning (AIL), pp. 18-23. Montpellier, France." This dataset comprises six clusters (each containing 15,000 samples) as well as 10,000 noise samples. These samples were mixed randomly.

The resulting neural network can be visualised using the R script `ShowTopoARTResults.R` or the R script `ShowHypersphereTopoARTResults.R`, respectively. Both R scripts are provided in the subfolder `R`.

## 5.67 LibTopoART\_samples.TopoART\_sample3 Class Reference

Clustering sample using very noisy synthetic two-dimensional data. [VB]

### 5.67.1 Detailed Description

Clustering sample using very noisy synthetic two-dimensional data. [VB]

Train TopoART or Hypersphere TopoART with a two-dimensional dataset similar to the one used in "Marko Tscherepanow and Sören Riechers (2012). An Incremental On-line Classifier for Imbalanced, Incomplete, and Noisy Data. In Proceedings of the European Conference on Artificial Intelligence (ECAI), Workshop on Active and Incremental Learning (AIL), pp. 18-23. Montpellier, France." The dataset applied here comprises 1,000,000 samples equally allotted to six clusters (each containing one sixth of the samples). Additionally, 1,000,000 uniformly distributed random samples are added. Finally, all samples are randomly shuffled.

Due to the randomness involved, the results differ between different runs of this program. However, they are qualitatively comparable: The first module creates a coarse clustering of the data while the second module refines it to the six clusters of the undisturbed portion of the dataset.

These results show the abilities of TopoART to cope with a high amount of noise data and produce stable results independent of the sample order.

The resulting neural network can be visualised using the R script `ShowTopoARTResults.R` or the R script `ShowHypersphereTopoARTResults.R`, respectively. Both R scripts are provided in the subfolder R.





## Index

- AdaptationState
  - LibTopoART, [12](#)
- ADAPTED\_NONPERMANENT\_WEIGHT
  - LibTopoART, [14](#)
- ADAPTED\_PERMANENT\_WEIGHT
  - LibTopoART, [14](#)
- ADDED\_EDGE\_CANDIDATE
  - LibTopoART, [14](#)
- ADDED\_NODE\_CANDIDATE
  - LibTopoART, [14](#)
- ADDED\_PERMANENT\_EDGE
  - LibTopoART, [14](#)
- ADDED\_PERMANENT\_NODE
  - LibTopoART, [14](#)
- ANY\_NONPERMANENT\_ADAPTATION\_MASK
  - LibTopoART, [14](#)
- ANY\_PERMANENT\_ADAPTATION\_MASK
  - LibTopoART, [14](#)
- BeginRecall
  - LibTopoART.Fast\_Episodic\_TopoART, [20, 21](#)
  - LibTopoART.IByteAccess\_Episodic\_recall, [66](#)
  - LibTopoART.IDecimalAccess\_Episodic\_recall, [80](#)
- BeginRecallKey1
  - LibTopoART.Fast\_TopoART\_AM, [28, 29](#)
  - LibTopoART.IByteAccess\_Associative\_recall, [63](#)
  - LibTopoART.IDecimalAccess\_Associative\_recall, [77](#)
- BeginRecallKey2
  - LibTopoART.Fast\_TopoART\_AM, [29, 30](#)
  - LibTopoART.IByteAccess\_Associative\_recall, [64](#)
  - LibTopoART.IDecimalAccess\_Associative\_recall, [77](#)
- bm\_cluster\_ID
  - LibTopoART.F2\_output, [18](#)
- bm\_node\_activation
  - LibTopoART.F2\_output, [18](#)
- bm\_node\_ID
  - LibTopoART.F2\_output, [18](#)
- bm\_permanent\_cluster\_ID
  - LibTopoART.F2\_output, [18](#)
- bm\_permanent\_node\_activation
  - LibTopoART.F2\_output, [18](#)
- bm\_permanent\_node\_ID
  - LibTopoART.F2\_output, [18](#)
- CategoryInfo
  - LibTopoART.CategoryInfo, [15](#)
- class\_ID
  - LibTopoART.CategoryInfo, [16](#)
  - LibTopoART.TopoART\_C\_Prediction, [120](#)
- cluster\_ID
  - LibTopoART.CategoryInfo, [16](#)
- ComputeClusterIDs
  - LibTopoART.Fast\_TopoART\_base, [35](#)
  - LibTopoART.ITopoART\_base, [101](#)
  - LibTopoART.TopoART, [110](#)
- confidence
  - LibTopoART.TopoART\_C\_Prediction, [120](#)
- D\_len
  - LibTopoART.Fast\_TopoART\_R, [52](#)
  - LibTopoART.ITopoART\_R\_base, [105](#)
  - LibTopoART.TopoART\_R, [125](#)
- d\_vec\_prediction
  - LibTopoART.TopoART\_R\_Prediction<\_Element-Type>, [126](#)
- Dispose
  - LibTopoART.Fast\_TopoART\_base, [35](#)
  - LibTopoART.TopoART, [110](#)
- EndRecall
  - LibTopoART.Fast\_Episodic\_TopoART, [21](#)
  - LibTopoART.Fast\_TopoART\_AM, [30](#)
  - LibTopoART.IEndRecall, [89](#)
- F2\_output
  - LibTopoART.F2\_output, [18](#)
- Fast\_Episodic\_TopoART
  - LibTopoART.Fast\_Episodic\_TopoART, [20](#)
- Fast\_TopoART
  - LibTopoART.Fast\_TopoART, [25](#)
- Fast\_TopoART\_AM
  - LibTopoART.Fast\_TopoART\_AM, [28](#)
- Fast\_TopoART\_C
  - LibTopoART.Fast\_TopoART\_C, [42, 43](#)
- Fast\_TopoART\_R
  - LibTopoART.Fast\_TopoART\_R, [48, 49](#)
- FileFormatVersion
  - LibTopoART.Fast\_Episodic\_TopoART, [24](#)
  - LibTopoART.Fast\_TopoART\_AM, [33](#)
  - LibTopoART.Fast\_TopoART\_base, [40](#)
  - LibTopoART.Fast\_TopoART\_C, [46](#)
  - LibTopoART.Fast\_TopoART\_R, [52](#)
  - LibTopoART.Hypersphere\_TopoART, [55](#)
  - LibTopoART.Hypersphere\_TopoART\_C, [60](#)
  - LibTopoART.TopoART, [114](#)
  - LibTopoART.TopoART\_C, [119](#)
  - LibTopoART.TopoART\_R, [125](#)
- FINAL\_MODULE
  - LibTopoART.Fast\_TopoART\_base, [40](#)
  - LibTopoART.LibTopoART\_info, [107](#)
  - LibTopoART.TopoART, [113](#)
- FloatBaseType
  - LibTopoART.Fast\_TopoART\_base, [40](#)
  - LibTopoART.TopoART, [114](#)
- GetAdaptationState
  - LibTopoART.Fast\_TopoART\_base, [35](#)
  - LibTopoART.IAdaptationStateCheck, [61](#)
  - LibTopoART.TopoART, [111](#)
- GetBMOutput

- LibTopoART.Fast\_TopoART\_AM, 30, 31
- LibTopoART.Fast\_TopoART\_base, 36, 37
- LibTopoART.IByteAccess\_TopoART, 67, 68
- LibTopoART.IByteAccess\_TopoART\_AM, 69
- LibTopoART.IDecimalAccess\_TopoART, 81, 82
- LibTopoART.IDecimalAccess\_TopoART\_AM, 83
- LibTopoART.TopoART, 111
- GetCategories
  - LibTopoART.Fast\_TopoART\_base, 37
  - LibTopoART.ICategoryAccess, 75
  - LibTopoART.TopoART, 112
- Hypersphere\_TopoART
  - LibTopoART.Hypersphere\_TopoART, 54, 55
- Hypersphere\_TopoART\_C
  - LibTopoART.Hypersphere\_TopoART\_C, 57, 58
- HypersphereTopoARTFileFormatVersion
  - LibTopoART.Hypersphere\_TopoART, 55
- I\_len
  - LibTopoART.Fast\_TopoART\_R, 52
  - LibTopoART.ITopoART\_R\_base, 106
  - LibTopoART.TopoART\_R, 125
- i\_vec\_prediction
  - LibTopoART.TopoART\_R\_Prediction< \_Element-  
Type >, 126
- Important
  - LibTopoART, 14
- IntegerBaseType
  - LibTopoART.Fast\_TopoART\_base, 40
  - LibTopoART.TopoART, 114
- InterEpisodeRecallStep
  - LibTopoART.Fast\_Episodic\_TopoART, 21, 22
  - LibTopoART.IByteAccess\_Episodic\_recall, 66
  - LibTopoART.IDecimalAccess\_Episodic\_recall, 80
- IntraEpisodeRecallStep
  - LibTopoART.Fast\_Episodic\_TopoART, 22, 23
  - LibTopoART.IByteAccess\_Episodic\_recall, 66
  - LibTopoART.IDecimalAccess\_Episodic\_recall, 80
- Key\_1\_len
  - LibTopoART.Fast\_TopoART\_AM, 33
  - LibTopoART.ITopoART\_AM\_base, 100
- Key\_2\_len
  - LibTopoART.Fast\_TopoART\_AM, 33
  - LibTopoART.ITopoART\_AM\_base, 100
- Learn
  - LibTopoART.Fast\_Episodic\_TopoART, 23
  - LibTopoART.Fast\_TopoART, 26
  - LibTopoART.Fast\_TopoART\_AM, 31, 32
  - LibTopoART.Fast\_TopoART\_base, 38
  - LibTopoART.Fast\_TopoART\_C, 43, 44
  - LibTopoART.Fast\_TopoART\_R, 49, 50
  - LibTopoART.Hypersphere\_TopoART\_C, 58
  - LibTopoART.IByteAccess\_TopoART, 68
  - LibTopoART.IByteAccess\_TopoART\_AM, 70
  - LibTopoART.IByteAccess\_TopoART\_C, 71
  - LibTopoART.IByteAccess\_TopoART\_R, 74
  - LibTopoART.IDecimalAccess\_TopoART, 82
  - LibTopoART.IDecimalAccess\_TopoART\_AM, 84
  - LibTopoART.IDecimalAccess\_TopoART\_C, 85
  - LibTopoART.IDecimalAccess\_TopoART\_R, 88
  - LibTopoART.TopoART, 112
  - LibTopoART.TopoART\_C, 118
  - LibTopoART.TopoART\_R, 123
- LibTopoART, 9
  - AdaptationState, 12
  - ADAPTED\_NONPERMANENT\_WEIGHT, 14
  - ADAPTED\_PERMANENT\_WEIGHT, 14
  - ADDED\_EDGE\_CANDIDATE, 14
  - ADDED\_NODE\_CANDIDATE, 14
  - ADDED\_PERMANENT\_EDGE, 14
  - ADDED\_PERMANENT\_NODE, 14
  - ANY\_NONPERMANENT\_ADAPTATION\_MASK, 14
  - ANY\_PERMANENT\_ADAPTATION\_MASK, 14
  - Important, 14
  - NO\_ADAPTATION, 14
  - Normal, 14
  - REMOVED\_EDGE\_CANDIDATE, 14
  - REMOVED\_NODE\_CANDIDATE, 14
  - Verbose, 14
  - VerbosityLevel, 14
- LibTopoART.CategoryInfo, 15
  - CategoryInfo, 15
  - class\_ID, 16
  - cluster\_ID, 16
  - spatial\_weights, 16
  - temporal\_weights, 16
- LibTopoART.F2\_output, 17
  - bm\_cluster\_ID, 18
  - bm\_node\_activation, 18
  - bm\_node\_ID, 18
  - bm\_permanent\_cluster\_ID, 18
  - bm\_permanent\_node\_activation, 18
  - bm\_permanent\_node\_ID, 18
  - F2\_output, 18
- LibTopoART.Fast\_Episodic\_TopoART, 19
  - BeginRecall, 20, 21
  - EndRecall, 21
  - Fast\_Episodic\_TopoART, 20
  - FileFormatVersion, 24
  - InterEpisodeRecallStep, 21, 22
  - IntraEpisodeRecallStep, 22, 23
  - Learn, 23
- LibTopoART.Fast\_TopoART, 24
  - Fast\_TopoART, 25
  - Learn, 26
- LibTopoART.Fast\_TopoART\_AM, 26
  - BeginRecallKey1, 28, 29
  - BeginRecallKey2, 29, 30
  - EndRecall, 30
  - Fast\_TopoART\_AM, 28
  - FileFormatVersion, 33
  - GetBMOOutput, 30, 31
  - Key\_1\_len, 33

- Key\_2\_len, 33
- Learn, 31, 32
- RecallStep, 32
- LibTopoART.Fast\_TopoART\_base, 33
  - ComputeClusterIDs, 35
  - Dispose, 35
  - FileFormatVersion, 40
  - FINAL\_MODULE, 40
  - FloatBaseType, 40
  - GetAdaptationState, 35
  - GetBMOutput, 36, 37
  - GetCategories, 37
  - IntegerBaseType, 40
  - Learn, 38
  - Phi, 40
  - Phis, 40
  - ResetAdaptationState, 38
  - Save, 39
  - SaveText, 39
  - TopoARTFileFormatVersion, 41
- LibTopoART.Fast\_TopoART\_C, 41
  - Fast\_TopoART\_C, 42, 43
  - FileFormatVersion, 46
  - Learn, 43, 44
  - Predict, 44–46
  - UNDEFINED\_CLASS\_ID, 46
- LibTopoART.Fast\_TopoART\_R, 47
  - D\_len, 52
  - Fast\_TopoART\_R, 48, 49
  - FileFormatVersion, 52
  - I\_len, 52
  - Learn, 49, 50
  - Predict, 50–52
- LibTopoART.Hypersphere\_TopoART, 53
  - FileFormatVersion, 55
  - Hypersphere\_TopoART, 54, 55
  - HypersphereTopoARTFileFormatVersion, 55
- LibTopoART.Hypersphere\_TopoART\_C, 55
  - FileFormatVersion, 60
  - Hypersphere\_TopoART\_C, 57, 58
  - Learn, 58
  - Predict, 59
  - UNDEFINED\_CLASS\_ID, 60
- LibTopoART.IAdaptationStateCheck, 60
  - GetAdaptationState, 61
  - ResetAdaptationState, 61
- LibTopoART.IAssociative\_recall, 61
- LibTopoART.IByteAccess\_Associative\_recall, 62
  - BeginRecallKey1, 63
  - BeginRecallKey2, 64
  - RecallStep, 64
- LibTopoART.IByteAccess\_Episodic\_recall, 65
  - BeginRecall, 66
  - InterEpisodeRecallStep, 66
  - IntraEpisodeRecallStep, 66
- LibTopoART.IByteAccess\_TopoART, 67
  - GetBMOutput, 67, 68
  - Learn, 68
- LibTopoART.IByteAccess\_TopoART\_AM, 68
  - GetBMOutput, 69
  - Learn, 70
- LibTopoART.IByteAccess\_TopoART\_C, 70
  - Learn, 71
  - Predict, 72
- LibTopoART.IByteAccess\_TopoART\_R, 73
  - Learn, 74
  - Predict, 74
- LibTopoART.ICategoryAccess, 75
  - GetCategories, 75
- LibTopoART.IDecimalAccess\_Associative\_recall, 76
  - BeginRecallKey1, 77
  - BeginRecallKey2, 77
  - RecallStep, 77
- LibTopoART.IDecimalAccess\_Episodic\_recall, 78
  - BeginRecall, 80
  - InterEpisodeRecallStep, 80
  - IntraEpisodeRecallStep, 80
- LibTopoART.IDecimalAccess\_TopoART, 81
  - GetBMOutput, 81, 82
  - Learn, 82
- LibTopoART.IDecimalAccess\_TopoART\_AM, 82
  - GetBMOutput, 83
  - Learn, 84
- LibTopoART.IDecimalAccess\_TopoART\_C, 84
  - Learn, 85
  - Predict, 86
- LibTopoART.IDecimalAccess\_TopoART\_R, 87
  - Learn, 88
  - Predict, 88
- LibTopoART.IEndRecall, 89
  - EndRecall, 89
- LibTopoART.IEpisodic\_recall, 90
- LibTopoART.IEpisodic\_TopoART, 91
- LibTopoART.IFast\_Associative\_recall, 91
- LibTopoART.IFast\_Episodic\_recall, 92
- LibTopoART.IFast\_Episodic\_TopoART, 93
- LibTopoART.IFast\_TopoART, 94
- LibTopoART.IFast\_TopoART\_AM, 95
- LibTopoART.IFast\_TopoART\_C, 95
- LibTopoART.IFast\_TopoART\_R, 96
- LibTopoART.InvalidClassIDException, 96
- LibTopoART.InvalidFileException, 97
- LibTopoART.InvalidModuleIndexException, 97
- LibTopoART.InvalidNumberException, 97
- LibTopoART.InvalidSizeException, 97
- LibTopoART.InvalidStateException, 98
- LibTopoART.ITopoART, 98
- LibTopoART.ITopoART\_AM, 98
- LibTopoART.ITopoART\_AM\_base, 99
  - Key\_1\_len, 100
  - Key\_2\_len, 100
- LibTopoART.ITopoART\_base, 100
  - ComputeClusterIDs, 101
  - ModuleNum, 102
  - Phi, 102
  - Phis, 102

- Save, 101
- SaveText, 102
- LibTopoART.ITopoART\_C, 102
- LibTopoART.ITopoART\_R, 103
- LibTopoART.ITopoART\_R\_base, 104
  - D\_len, 105
  - I\_len, 106
- LibTopoART.LibTopoART\_control, 106
  - verbosity, 106
- LibTopoART.LibTopoART\_info, 106
  - FINAL\_MODULE, 107
  - networks, 107
  - UNDEFINED, 107
  - version, 107
- LibTopoART.TopoART, 108
  - ComputeClusterIDs, 110
  - Dispose, 110
  - FileFormatVersion, 114
  - FINAL\_MODULE, 113
  - FloatBaseType, 114
  - GetAdaptationState, 111
  - GetBMOOutput, 111
  - GetCategories, 112
  - IntegerBaseType, 114
  - Learn, 112
  - Phi, 114
  - Phis, 114
  - ResetAdaptationState, 113
  - Save, 113
  - SaveText, 113
  - TopoART, 110
  - TopoARTFileFormatVersion, 114
- LibTopoART.TopoART\_C, 116
  - FileFormatVersion, 119
  - Learn, 118
  - Predict, 118, 119
  - TopoART\_C, 117
  - UNDEFINED\_CLASS\_ID, 119
- LibTopoART.TopoART\_C\_Prediction, 119
  - class\_ID, 120
  - confidence, 120
  - TopoART\_C\_Prediction, 120
- LibTopoART.TopoART\_R, 121
  - D\_len, 125
  - FileFormatVersion, 125
  - I\_len, 125
  - Learn, 123
  - Predict, 124
  - TopoART\_R, 122, 123
- LibTopoART.TopoART\_R\_Prediction< \_ElementType  
>, 125
  - d\_vec\_prediction, 126
  - i\_vec\_prediction, 126
  - NO\_PREDICTION, 127
  - PrintPredictions, 126
  - TopoART\_R\_Prediction, 126
- LibTopoART\_samples, 14
- LibTopoART\_samples.Episodic\_Top ART\_sample1, 16
- LibTopoART\_samples.Episodic\_Top ART\_sample2, 17
- LibTopoART\_samples.TopoART\_AM\_sample1, 115
- LibTopoART\_samples.TopoART\_AM\_sample2, 115
- LibTopoART\_samples.TopoART\_C\_sample1, 121
- LibTopoART\_samples.TopoART\_C\_sample2, 121
- LibTopoART\_samples.TopoART\_R\_sample1, 127
- LibTopoART\_samples.TopoART\_R\_sample2, 127
- LibTopoART\_samples.TopoART\_R\_sample3, 127
- LibTopoART\_samples.TopoART\_sample1, 128
- LibTopoART\_samples.TopoART\_sample2, 128
- LibTopoART\_samples.TopoART\_sample3, 128
- ModuleNum
  - LibTopoART.ITopoART\_base, 102
- networks
  - LibTopoART.LibTopoART\_info, 107
- NO\_ADAPTATION
  - LibTopoART, 14
- NO\_PREDICTION
  - LibTopoART.TopoART\_R\_Prediction< \_Element-  
Type >, 127
- Normal
  - LibTopoART, 14
- Phi
  - LibTopoART.Fast\_Top ART\_base, 40
  - LibTopoART.ITopoART\_base, 102
  - LibTopoART.TopoART, 114
- Phis
  - LibTopoART.Fast\_Top ART\_base, 40
  - LibTopoART.ITopoART\_base, 102
  - LibTopoART.TopoART, 114
- Predict
  - LibTopoART.Fast\_Top ART\_C, 44–46
  - LibTopoART.Fast\_Top ART\_R, 50–52
  - LibTopoART.Hypersphere\_Top ART\_C, 59
  - LibTopoART.IByteAccess\_Top ART\_C, 72
  - LibTopoART.IByteAccess\_Top ART\_R, 74
  - LibTopoART.IDecimalAccess\_Top ART\_C, 86
  - LibTopoART.IDecimalAccess\_Top ART\_R, 88
  - LibTopoART.TopoART\_C, 118, 119
  - LibTopoART.TopoART\_R, 124
- PrintPredictions
  - LibTopoART.TopoART\_R\_Prediction< \_Element-  
Type >, 126
- RecallStep
  - LibTopoART.Fast\_Top ART\_AM, 32
  - LibTopoART.IByteAccess\_Associative\_recall, 64
  - LibTopoART.IDecimalAccess\_Associative\_recall,  
77
- REMOVED\_EDGE\_CANDIDATE
  - LibTopoART, 14
- REMOVED\_NODE\_CANDIDATE
  - LibTopoART, 14
- ResetAdaptationState
  - LibTopoART.Fast\_Top ART\_base, 38
  - LibTopoART.IAdaptationStateCheck, 61

- LibTopoART.TopoART, [113](#)
- Save
  - LibTopoART.Fast\_TopoART\_base, [39](#)
  - LibTopoART.ITopoART\_base, [101](#)
  - LibTopoART.TopoART, [113](#)
- SaveText
  - LibTopoART.Fast\_TopoART\_base, [39](#)
  - LibTopoART.ITopoART\_base, [102](#)
  - LibTopoART.TopoART, [113](#)
- spatial\_weights
  - LibTopoART.CategoryInfo, [16](#)
- temporal\_weights
  - LibTopoART.CategoryInfo, [16](#)
- TopoART
  - LibTopoART.TopoART, [110](#)
- TopoART\_C
  - LibTopoART.TopoART\_C, [117](#)
- TopoART\_C\_Prediction
  - LibTopoART.TopoART\_C\_Prediction, [120](#)
- TopoART\_R
  - LibTopoART.TopoART\_R, [122](#), [123](#)
- TopoART\_R\_Prediction
  - LibTopoART.TopoART\_R\_Prediction< \_Element-  
Type >, [126](#)
- TopoARTFileFormatVersion
  - LibTopoART.Fast\_TopoART\_base, [41](#)
  - LibTopoART.TopoART, [114](#)
- UNDEFINED
  - LibTopoART.LibTopoART\_info, [107](#)
- UNDEFINED\_CLASS\_ID
  - LibTopoART.Fast\_TopoART\_C, [46](#)
  - LibTopoART.Hypersphere\_TopoART\_C, [60](#)
  - LibTopoART.TopoART\_C, [119](#)
- Verbose
  - LibTopoART, [14](#)
- verbosity
  - LibTopoART.LibTopoART\_control, [106](#)
- VerbosityLevel
  - LibTopoART, [14](#)
- version
  - LibTopoART.LibTopoART\_info, [107](#)