Segugio; the flexible and efficient tool for reasoning on graphical models

Generated by Doxygen 1.8.15

Chapter 1

Preliminaries

1.1 Introduction

This Section will provide the notation used for the rest of this guide. Undirect Graphical models are networks made of variables and factors.
This library is intended for managing categorical variables V, i.e. random variable having a discrete domain:

$$\textsc{Domain}(V) = \{v_0, \cdots, v_n\} \tag{1.1}$$

The entire population of variables contained in a model is a set denoted as $\mathcal{V} = V_1, \cdots, V_m$.
Factors (sometimes also called potentials) are positive real functions describing the correlation among the variables in the network. The domain of a factor is the cartesian product of the domains of the variables involved in that factor. Suppose the generic factor Φ involves the set of variables: X, Y, Z, then $\Phi(X, Y, Z)$ is a function:

$$\Phi(X, Y, Z) : \textsc{Domain}(X) \times \textsc{Domain}(Y) \times \textsc{Domain}(Z) \longrightarrow \mathbb{R}^+ \tag{1.2}$$

Basically, the aim of Φ is to assume high values for those combinations $\{x, y, z\}$ that are probable and low values (at least a null value) for those being improbable. The population of factors of a network must be considered when computing the joint probability distribution of all the variables in the model $\mathbb{P}(V_{1,\cdots,m})$. Let be $\mathcal{D}_i \subset \mathcal{V}$ the subset of variables involved by the i^{th} factor Φ_i. The energy function E of a graph is the product of the factors:

$$E(V_{1,\cdots,m}) = \Phi_1(\mathcal{D}_1) \cdot \cdots \cdot \Phi_p(\mathcal{D}_p) = \prod_{i=1}^{p} \Phi_i(\mathcal{D}_i) \tag{1.3}$$

The joint probability distribution of an undirect graphical model is computable as follows:

$$\mathbb{P}(V_{1,\cdots,m}) = \frac{E(V_{1,\cdots,m})}{\mathcal{Z}} \tag{1.4}$$

$\mathcal{Z}$ is a normalization coefficient defined as follows:

$$\mathcal{Z} = \sum_{\tilde{V}_{1,\cdots,m} \in \textsc{Domain}(V_1) \times \cdots \times \textsc{Domain}(V_m)} E(\tilde{V}_{1,\cdots,m}) \tag{1.5}$$

Although the general theory behind graphical models supports the existance of generic multivaried factors, this library will address only two possible types:

- Binary potentials: they involve a pair of variables.
  $\textsc{Cardinality}(\mathcal{D}) = 2$

	b_0	b_1	b_2	b_3	b_4
a_0	1	4	0	0	0
a_1	0	1	0	0	0
a_2	0	0	5	0	1

a_0	a_1	a_2	b_3	b_4
0	2	0	2	1

- Unary potentials: they involve a single variable.
  $\textsc{Cardinality}(\mathcal{D}) = 1$

We can store the values in the codomain of a Binary potential in a two dimensional table. For instance, let be Φ_b a binary potential involving two variables A and B, whose domains contains 3 and 5 possible values respectively:

$$\begin{gathered} \textsc{Domain}(A) = \{a_1, a_2, a_3\} \\ \textsc{Domain}(B) = \{b_1, b_2, b_3, b_4, b_5\} \end{gathered} \tag{1.6}$$

The values assumed by $\Phi_b(A, B)$ are described by the table **??**. Essentially, $\Phi_b(A, B)$ tells us that the combinations $\{a_0, b_1\}$, $\{a_2, b_2\}$ are highly probable; $\{a_0, b_0\}$, $\{a_1, b_1\}$ and $\{a_2, b_4\}$ are moderately probable. Let be $\Phi_u(A)$ a Unary potential involving variable A. The values characterizing Φ_u can be stored in a simple vector, see table **??** Unary potentials can be adopted for expressing the prior knowledge about a variable.
Consider a graph for which $\Phi_b(A, B)$ is the only potential in the net, then the joint density $\mathbb{P}(A, B)$ will assume the following values:

$$\begin{aligned} \mathbb{P}(a_0, b_1) &= \frac{4}{\mathcal{Z}} = 0.3333 \\ \mathbb{P}(a_2, b_2) &= \frac{5}{\mathcal{Z}} = 0.4167 \\ \mathbb{P}(a_0, b_0) &= \mathbb{P}(a_1, b_1) = \mathbb{P}(a_2, b_4) = \frac{1}{\mathcal{Z}} = 0.0833 \end{aligned} \tag{1.7}$$

since $\mathcal{Z}$ is equal to:

$$\mathcal{Z} = \sum_{\forall i=0,1,2, \forall j=0,1,2,3,4} \Phi_b(A = a_i, B = b_j) = 12 \tag{1.8}$$

Both Unary and Binary potentials, can be of two possible classes:

- Simple shape, i.e. the basic case. The potential is simply described by the set of values assuming for the input combination. $\Phi_b(A, B)$ of the previous example is a Simple shape.
- Exponential shape. This kind of factors are indicated with Ψ_i and are defined as follows:

  $$\Psi_i = exp(w \cdot \Phi_i) \tag{1.9}$$

  where Φ_i is an underlying simple shape. The weight w, can be tunable or not. In the first case, it is a free parameter whose value is decided after training the model, otherwise is a constant . Exponential shapes with fixed weight will be denoted with $\overline{\Psi}_i$.

Figure 1.1 reports an example of undirected graph. Set $\mathcal{V}$ is made of 4 variables: A, B, C, D. There are 5 Binary potentials and 2 Unary ones. The notation adopted for Fig. 1.1 will be adopted for the rest of this guide. Weights α, β, γ and δ are assumed for respectively $\Psi_{AC}, \Psi_{AB}, \Psi_{CD}, \Psi_B$. For the sake of clarity, the joint probability of the variables in Fig. 1.1 is computable as follows:

$$\begin{aligned} \mathbb{P}(A, B, C, D) &= \frac{E(A, B, C)}{\mathcal{Z}(\alpha, \beta, \gamma, \delta)} \\ E(A, B, C) &= \Phi_A(A) \cdot exp(\alpha \Phi_{AC}(A, C)) \cdot exp(\beta \Phi_{AB}(A, B)) \cdots \\ \cdots &\quad \Phi_{BC}(B, C) \cdot exp(\gamma \Phi_{CD}(C, D)) \cdot \Phi_{BD}(B, D) \cdot exp(\delta \Phi_B(B)) \end{aligned} \tag{1.10}$$

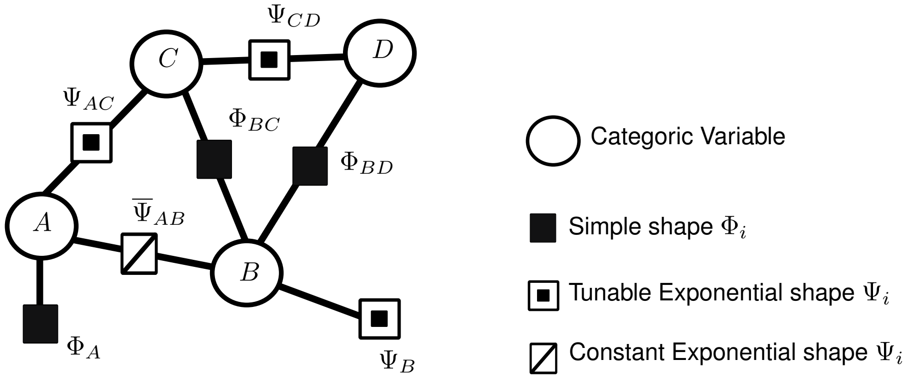


Figure 1.1 Example of graph: the legend of the right applies.

1.1.1 Query the graph

Graphical models are mainly used for performing belief propagation. Subset $\mathcal{O} = O_1, \cdots, O_l \subset \mathcal{V}$ is adopted for denoting the set of evidences: those variables in the net whose value become known. $\mathcal{O}$ can be dynamical or not, see the Section about he types of graphs METTERE. The hidden variables are contained in the complementary set $\mathcal{H} = H_1, \cdots, H_t$. Clearly $\mathcal{V} = \mathcal{O} \cup \mathcal{H}$ and $\emptyset = \mathcal{O} \cap \mathcal{H}$. Knowing the joint probability distribution of variables in $\mathcal{V}$ (equation 1.4) the conditional distribution of $H \in \mathcal{H}$ w.r.t. $O \in \mathcal{O}$ can be determined as follows:

$$\mathbb{P}(H|O) = \frac{\mathbb{P}(H,O)}{\sum_{\forall \hat{H} \in \mathcal{H}} \mathbb{P}(\hat{H},O)} = \frac{E(H,O)}{\sum_{\forall \hat{H}} E(\hat{H},O)} = \frac{E(H,O)}{\mathcal{Z}(O)} \tag{1.11}$$

The marginal probability $\mathbb{P}(H_i|O)$ of a variable in H is formally defined as follows:

$$\mathbb{P}(H_i|O) = \sum_{Y \in \{\mathcal{H} \setminus H_i\}} \mathbb{P}(H_i, Y|O) \tag{1.12}$$

The above marginal distribution is essentially the conditional distribution of H_i w.r.t. O, no matter the other variables in $\mathcal{H}$. The entire set of marginal distributions $\mathbb{P}(H_1|\mathcal{O}), \cdots, \mathbb{P}(H_t|\mathcal{O})$ can be easily computed by making use of the Message Passing algorithm, see METTERE.

Drawing samples for $\mathcal{H}$ from the conditional distribution $\mathbb{P}(\mathcal{H}|\mathcal{O})$ (without having to build the aforementioned distribution) can be done exploiting a Gibbs sampler, see Section METTERE.

1.1.2 Message Passing

TODO spiegare brevemente message passing e calcolo probabilità marginali.

1.1.3 Gibbs sampling

Spiegare MP algo.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

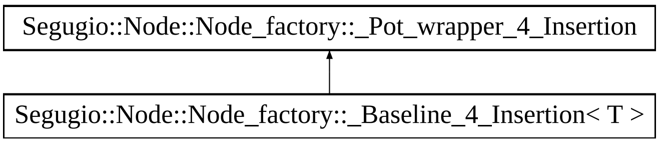
Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Class Documentation

4.1 Segugio::Node::Node_factory::_Baseline_4_Insertion< T > Struct Template Reference

Inheritance diagram for Segugio::Node::Node_factory::_Baseline_4_Insertion< T >:


Segugio::Node::Node_factory::_Pot_wrapper_4_Insertion
Segugio::Node::Node_factory::_Baseline_4_Insertion< T >

Public Member Functions

- **_Baseline_4_Insertion** (T ∗wrp)
- virtual const std::list< Categoric_var ∗ > ∗ **Get_involved_var_safe** ()
- virtual Potential ∗ **Get_Potential_to_Insert** (const std::list< Categoric_var ∗ > &var_involved, const bool &get_cloned)

Protected Attributes

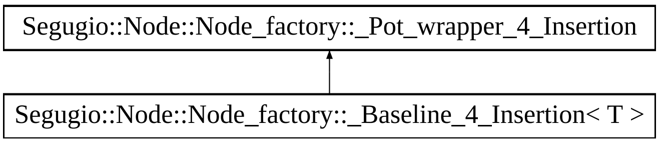
- T ∗ **wrapped**

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

- C:/Users/andre/Desktop/CRF/CRF/Header/Node.h

4.2 Segugio::Node::Node_factory::_Pot_wrapper_4_Insertion Struct Reference

Inheritance diagram for Segugio::Node::Node_factory::_Pot_wrapper_4_Insertion:


Segugio::Node::Node_factory::_Pot_wrapper_4_Insertion
Segugio::Node::Node_factory::_Baseline_4_Insertion< T >

Public Member Functions

- virtual const std::list< Categoric_var * > * **Get_involved_var_safe** ()=0
- virtual Potential * **Get_Potential_to_Insert** (const std::list< Categoric_var * > &var_involved, const bool &get_cloned)=0

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

- C:/Users/andre/Desktop/CRF/CRF/Header/Node.h

4.3 Segugio::Node::Node_factory::_SubGraph Class Reference

Public Member Functions

- _SubGraph (Node_factory *Originating, const std::list< Categoric_var * > &sub_set_to_consider)

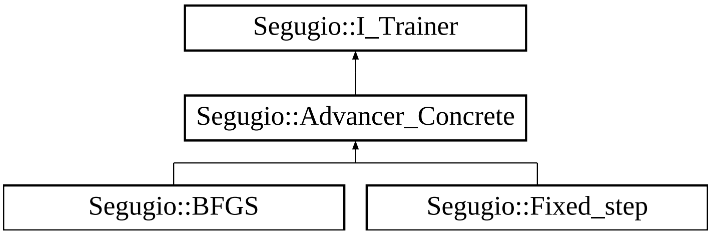
  Builds a reduction of the actual net, considering the actual observation values. The subgraph is not automatically updated w.r.t. modifications of the originating net: in such cases just create a novel subgraph with the same sub_set of variables involved.
- void **Get_marginal_prob_combination** (std::list< float > *result, const std::list< std::list< size_t >> &combinations, const std::list< Categoric_var * > &var_order_in_combination)
- void **Get_marginal_prob_combination** (std::list< float > *result, const std::list< size_t * > &combinations, const std::list< Categoric_var * > &var_order_in_combination)
- void **MAP** (std::list< size_t > *result)
- void **Gibbs_Sampling** (std::list< std::list< size_t >> *result, const unsigned int &N_samples, const unsigned int &initial_sample_to_skip)
- void **Get_All_variables** (std::list< Categoric_var * > *result)

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

- C:/Users/andre/Desktop/CRF/CRF/Header/Node.h
- C:/Users/andre/Desktop/CRF/CRF/Source/Subgraph.cpp

4.4 Segugio::Advancer_Concrete Class Reference

Inheritance diagram for Segugio::Advancer_Concrete:



Public Member Functions

- virtual void **Reset** ()
- void **Train** (Graph_Learnable ∗model_to_train, Training_set ∗Train_set, const unsigned int &Max_Iterations, std::list< float > ∗descend_story)
- virtual float **_advance** (Graph_Learnable ∗model_to_advance, const std::list< size_t ∗ > &comb_in_train_set, const std::list< Categoric_var ∗ > &comb_var)=0

Additional Inherited Members

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

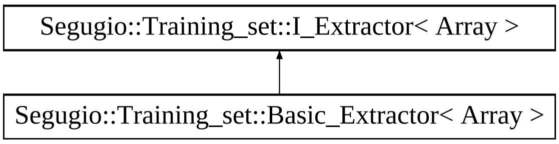
- C:/Users/andre/Desktop/CRF/CRF/Source/Trainer.cpp

4.5 Segugio::Training_set::Basic_Extractor< Array > Class Template Reference

Basic extractor, see Training_set(const std::list<std::string>& variable_names, std::list<Array> samples, I_Extractor<Array>∗ extractor)

```
#include <Training_set.h>
```

Inheritance diagram for Segugio::Training_set::Basic_Extractor< Array >:


Segugio::Training_set::I_Extractor< Array >
Segugio::Training_set::Basic_Extractor< Array >

Additional Inherited Members

4.5.1 Detailed Description

template<typename Array>
class Segugio::Training_set::Basic_Extractor< Array >

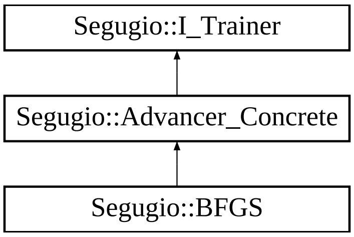
Basic extractor, see Training_set(const std::list<std::string>& variable_names, std::list<Array> samples, I_Extractor<Array>∗ extractor)

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

- C:/Users/andre/Desktop/CRF/CRF/Header/Training_set.h

4.6 Segugio::BFGS Class Reference

Inheritance diagram for Segugio::BFGS:



Public Member Functions

- void **Reset** ()

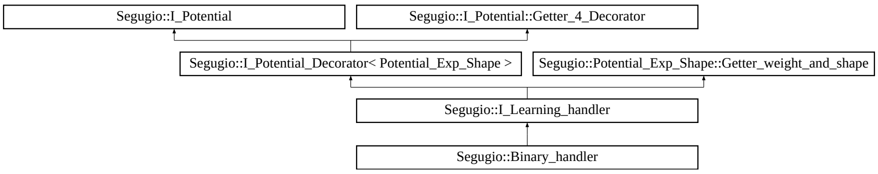
Additional Inherited Members

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

- C:/Users/andre/Desktop/CRF/CRF/Source/Trainer.cpp

4.7 Segugio::Binary_handler Class Reference

Inheritance diagram for Segugio::Binary_handler:


Segugio::I_Potential
Segugio::I_Potential::Getter_4_Decorator
Segugio::I_Potential_Decorator< Potential_Exp_Shape >
Segugio::Potential_Exp_Shape::Getter_weight_and_shape
Segugio::I_Learning_handler
Segugio::Binary_handler

Public Member Functions

- **Binary_handler** (Node ∗N1, Node ∗N2, Potential_Exp_Shape ∗pot_to_handle)

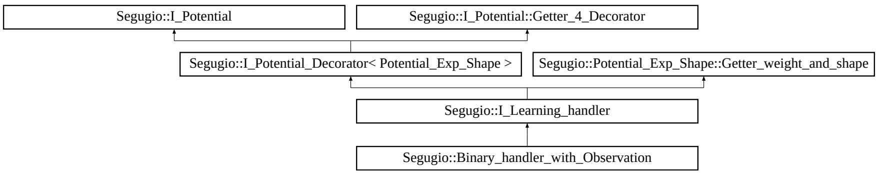
Additional Inherited Members

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

- C:/Users/andre/Desktop/CRF/CRF/Source/Graphical_model.cpp

4.8 Segugio::Binary_handler_with_Observation Class Reference

Inheritance diagram for Segugio::Binary_handler_with_Observation:


Segugio::I_Potential
Segugio::I_Potential::Getter_4_Decorator
Segugio::I_Potential_Decorator< Potential_Exp_Shape >
Segugio::Potential_Exp_Shape::Getter_weight_and_shape
Segugio::I_Learning_handler
Segugio::Binary_handler_with_Observation

Public Member Functions

- **Binary_handler_with_Observation** (Node ∗Hidden_var, size_t ∗observed_val, I_Learning_handler ∗handle_to_substitute)

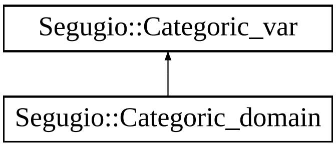
Additional Inherited Members

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

- C:/Users/andre/Desktop/CRF/CRF/Source/Graphical_model.cpp

4.9 Segugio::Categoric_domain Class Reference

Inheritance diagram for Segugio::Categoric_domain:



Public Member Functions

- const float & **operator[ ]** (const size_t &pos)

Additional Inherited Members

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

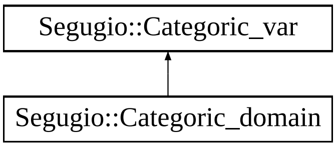
- C:/Users/andre/Desktop/CRF/CRF/Header/Potential.h
- C:/Users/andre/Desktop/CRF/CRF/Source/Potential.cpp

4.10 Segugio::Categoric_var Class Reference

Describes a categoric variable.

```
#include <Potential.h>
```

Inheritance diagram for Segugio::Categoric_var:



Public Member Functions

- Categoric_var (const size_t &size, const std::string &name)

  domain is assumed to be {0,1,2,3,...,size}
- **Categoric_var** (const Categoric_var &to_copy)
- const size_t & **size** () const
- const std::string & **Get_name** ()

Protected Attributes

- size_t **Size**
- std::string Name

4.10.1 Detailed Description

Describes a categoric variable.

, having a finite set as domain, assumed by default as {0,1,2,3,...,size}

4.10.2 Constructor & Destructor Documentation

4.10.2.1 Categoric_var()

```
Segugio::Categoric_var::Categoric_var (
             const size_t & size,
             const std::string & name )
```

domain is assumed to be {0,1,2,3,...,size}

Parameters

in	*size*	domain size of this variable
in	*name*	name to attach to this variable. It cannot be an empty string ""

4.10.3 Member Data Documentation

4.10.3.1 Name

```
std::string Segugio::Categoric_var::Name  [protected]
```

domain size

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

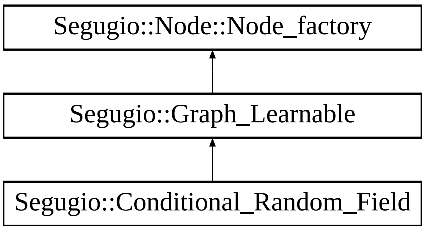
- C:/Users/andre/Desktop/CRF/CRF/Header/Potential.h
- C:/Users/andre/Desktop/CRF/CRF/Source/Potential.cpp

4.11 Segugio::Conditional_Random_Field Class Reference

This class describes Conditional Random fields.

`#include <Graphical_model.h>`

Inheritance diagram for Segugio::Conditional_Random_Field:


Segugio::Node::Node_factory
Segugio::Graph_Learnable
Segugio::Conditional_Random_Field

Public Member Functions

- Conditional_Random_Field (const std::string &config_xml_file, const std::string &prefix_config_xml_file="")

  The model is built considering the information contained in an xml configuration file.
- Conditional_Random_Field (const std::list< Potential_Exp_Shape ∗ > &potentials, const std::list< Categoric_var ∗ > &observed_var, const bool &use_cloning_Insert=true, const std::list< bool > &tunable_mask={}, const std::list< Potential_Shape ∗ > &shapes={})

  This constructor initializes the graph with the specified potentials passed as input, setting the variables passed as the one observed.
- void Set_Observation_Set_val (const std::list< size_t > &new_observed_vals)

  see Node::Node_factory::Set_Observation_Set_val(const std::list<size_t>& new_observed_vals)

Additional Inherited Members

4.11.1 Detailed Description

This class describes Conditional Random fields.

Set_Observation_Set_var is depracated: the observed set of variables cannot be changed after construction.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 Conditional_Random_Field() [1/2]

```
Segugio::Conditional_Random_Field::Conditional_Random_Field (
            const std::string & config_xml_file,
            const std::string & prefix_config_xml_file = "" )
```

The model is built considering the information contained in an xml configuration file.

TODO spiegare come e' fatto xml

Parameters

in	*configuration*	file
in	*prefix*	to use. The file prefix_config_xml_file/config_xml_file is searched.

4.11.2.2 Conditional_Random_Field() [2/2]

```
Segugio::Conditional_Random_Field::Conditional_Random_Field (
            const std::list< Potential_Exp_Shape * > & potentials,
            const std::list< Categoric_var * > & observed_var,
            const bool & use_cloning_Insert = true,
            const std::list< bool > & tunable_mask = {},
            const std::list< Potential_Shape * > & shapes = {} )
```

This constructor initializes the graph with the specified potentials passed as input, setting the variables passed as the one observed.

Parameters

in	*potentials*	the initial set of exponential potentials to insert (can be empty)
in	*observed_var*	the set of variables to assume as observations
in	*use_cloning_Insert*	when is true, every time an Insert of a novel potential is called (this includes the passed potentials), a copy of that potential is actually inserted. Otherwise, the passed potential is inserted as is: this can be dangerous, cause that potential cna be externally modified, but the construction of a novel graph is faster.

Parameters

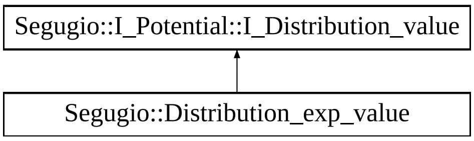
in	*tunable_mask*	when passed as non default value, it is must have the same size of potentials. Every value in this list is true if the corresponfing potential in the potentials list is tunable, i.e. has a weight whose value can vary with learning
in	*shapes*	A list of additional non learnable potentials to insert in the model

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

- C:/Users/andre/Desktop/CRF/CRF/Header/Graphical_model.h
- C:/Users/andre/Desktop/CRF/CRF/Source/Graphical_model.cpp

4.12 Segugio::Distribution_exp_value Struct Reference

Inheritance diagram for Segugio::Distribution_exp_value:


Segugio::I_Potential::I_Distribution_value
Segugio::Distribution_exp_value

Public Member Functions

- **Distribution_exp_value** (Distribution_value ∗to_wrap, float ∗weight)
- void **Set_val** (const float &v)
- void **Get_val** (float ∗result)
- size_t ∗ **Get_indeces** ()

Protected Attributes

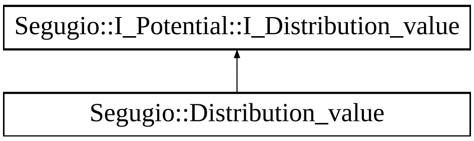
- float ∗ **w**
- Distribution_value ∗ **wrapped**

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

- C:/Users/andre/Desktop/CRF/CRF/Source/Potential.cpp

4.13 Segugio::Distribution_value Struct Reference

Inheritance diagram for Segugio::Distribution_value:


Segugio::I_Potential::I_Distribution_value
Segugio::Distribution_value

Public Member Functions

- **Distribution_value** (size_t ∗ind, const float &v=0.f)
- void **Set_val** (const float &v)
- void **Get_val** (float ∗result)
- size_t ∗ **Get_indeces** ()

Protected Attributes

- size_t ∗ **indices**
- float **val**

Friends

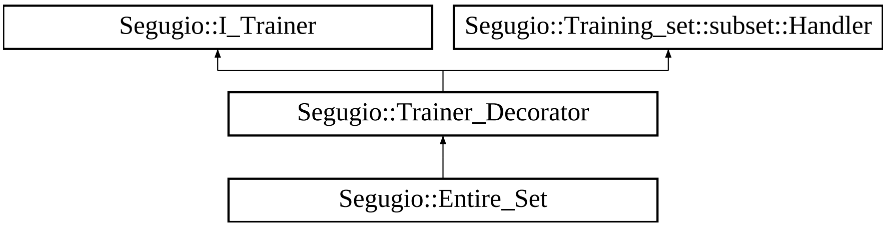
- struct **Distribution_exp_value**

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

- C:/Users/andre/Desktop/CRF/CRF/Source/Potential.cpp

4.14 Segugio::Entire_Set Class Reference

Inheritance diagram for Segugio::Entire_Set:


Segugio::I_Trainer
Segugio::Training_set::subset::Handler
Segugio::Trainer_Decorator
Segugio::Entire_Set

Public Member Functions

- **Entire_Set** (Advancer_Concrete ∗to_wrap)
- void **Train** (Graph_Learnable ∗model_to_train, Training_set ∗Train_set, const unsigned int &Max_Iterations, std::list< float > ∗descend_story)

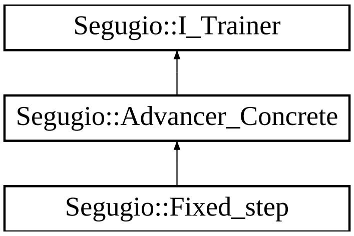
Additional Inherited Members

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

- C:/Users/andre/Desktop/CRF/CRF/Source/Trainer.cpp

4.15 Segugio::Fixed_step Class Reference

Inheritance diagram for Segugio::Fixed_step:



Public Member Functions

- **Fixed_step** (const float &step)

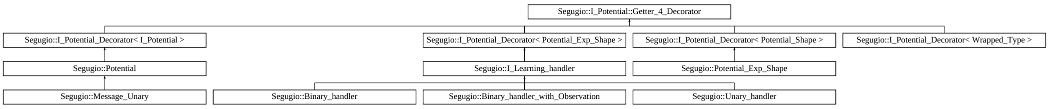
Additional Inherited Members

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

- C:/Users/andre/Desktop/CRF/CRF/Source/Trainer.cpp

4.16 Segugio::I_Potential::Getter_4_Decorator Struct Reference

Inheritance diagram for Segugio::I_Potential::Getter_4_Decorator:


Segugio::I_Potential::Getter_4_Decorator
Segugio::I_Potential_Decorator< I_Potential >
Segugio::I_Potential_Decorator< Potential_Exp_Shape >
Segugio::I_Potential_Decorator< Potential_Shape >
Segugio::I_Potential_Decorator< Wrapped_Type >
Segugio::Potential
Segugio::I_Learning_handler
Segugio::Potential_Exp_Shape
Segugio::Message_Unary
Segugio::Binary_handler
Segugio::Binary_handler_with_Observation
Segugio::Unary_handler

Static Protected Member Functions

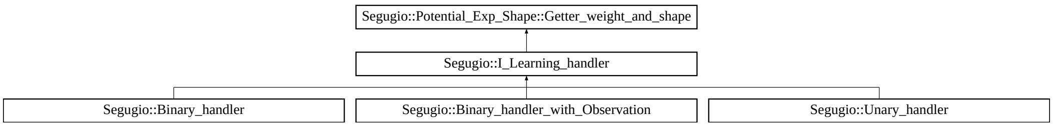
- static const std::list< Categoric_var ∗ > ∗ **Get_involved_var** (I_Potential ∗pot)
- static std::list< I_Distribution_value ∗ > ∗ **Get_distr** (I_Potential ∗pot)

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

- C:/Users/andre/Desktop/CRF/CRF/Header/Potential.h

4.17 Segugio::Potential_Exp_Shape::Getter_weight_and_shape Struct Reference

Inheritance diagram for Segugio::Potential_Exp_Shape::Getter_weight_and_shape:


Segugio::Potential_Exp_Shape::Getter_weight_and_shape
Segugio::I_Learning_handler
Segugio::Binary_handler
Segugio::Binary_handler_with_Observation
Segugio::Unary_handler

Static Protected Member Functions

- static float ∗ **Get_weight** (Potential_Exp_Shape ∗pot)
- static Potential_Shape ∗ **Get_shape** (Potential_Exp_Shape ∗pot)

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

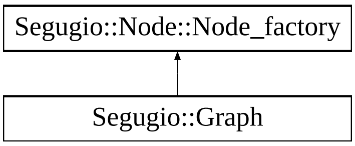
- C:/Users/andre/Desktop/CRF/CRF/Header/Potential.h

4.18 Segugio::Graph Class Reference

Interface for managing generic graphs.

```
#include <Graphical_model.h>
```

Inheritance diagram for Segugio::Graph:



Public Member Functions

- Graph (const bool &use_cloning_Insert=true)

  empty constructor
- Graph (const std::string &config_xml_file, const std::string &prefix_config_xml_file="")

  The model is built considering the information contained in an xml configuration file.
- Graph (const std::list< Potential_Shape ∗ > &potentials, const std::list< Potential_Exp_Shape ∗ > &potentials_exp, const bool &use_cloning_Insert=true)

  This constructor initializes the graph with the specified potentials passed as input.
- void Insert (Potential_Shape ∗pot)

  The model is built considering the information contained in an xml configuration file.
- void Insert (Potential_Exp_Shape ∗pot)

  The model is built considering the information contained in an xml configuration file.
- void Set_Observation_Set_var (const std::list< Categoric_var ∗ > &new_observed_vars)

  see Node::Node_factory::Set_Observation_Set_var(const std::list<Categoric_var∗>& new_observed_vars)
- void Set_Observation_Set_val (const std::list< size_t > &new_observed_vals)

  see Node::Node_factory::Set_Observation_Set_val(const std::list<size_t>& new_observed_vals)

Additional Inherited Members

4.18.1 Detailed Description

Interface for managing generic graphs.

Both Exponential and normal shapes can be included into the model. Learning is not possible: all belief propagation operations are performed assuming the mdoel as is. Every Potential_Shape or Potential_Exp_Shape is copied and that copy is inserted into the model.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 Graph() [1/3]

```
Segugio::Graph::Graph (
            const bool & use_cloning_Insert = true )  [inline]
```

empty constructor

Parameters

in	*use_cloning_Insert*	when is true, every time an Insert of a novel potential is called, a copy of that potential is actually inserted. Otherwise, the passed potential is inserted as is: this can be dangerous, cause that potential cna be externally modified, but the construction of a novel graph is faster.

4.18.2.2 Graph() [2/3]

```
Segugio::Graph::Graph (
            const std::string & config_xml_file,
            const std::string & prefix_config_xml_file = "" )
```

The model is built considering the information contained in an xml configuration file.

TODO spiegare come e' fatto xml

Parameters

in	*configuration*	file
in	*prefix*	to use. The file prefix_config_xml_file/config_xml_file is searched.

4.18.2.3 Graph() [3/3]

```
Segugio::Graph::Graph (
            const std::list< Potential_Shape * > & potentials,
            const std::list< Potential_Exp_Shape * > & potentials_exp,
            const bool & use_cloning_Insert = true )
```

This constructor initializes the graph with the specified potentials passed as input.

Parameters

in	*potentials*	the initial set of potentials to insert (can be empty)

Parameters

in	*potentials_exp*	the initial set of exponential potentials to insert (can be empty)
in	*use_cloning_Insert*	when is true, every time an Insert of a novel potential is called (this includes the passed potentials), a copy of that potential is actually inserted. Otherwise, the passed potential is inserted as is: this can be dangerous, cause that potential cna be externally modified, but the construction of a novel graph is faster.

4.18.3 Member Function Documentation

4.18.3.1 Insert() [1/2]

```
void Segugio::Graph::Insert (
            Potential_Shape * pot )   [inline]
```

The model is built considering the information contained in an xml configuration file.

Parameters

in	*the*	potential to insert. It can be a unary or a binary potential. In case it is binary, at least one of the variable involved must be already inserted to the model before (with a previous Insert having as input a potential which involves that variable).

4.18.3.2 Insert() [2/2]

```
void Segugio::Graph::Insert (
            Potential_Exp_Shape * pot )   [inline]
```

The model is built considering the information contained in an xml configuration file.

Parameters

in	*the*	potential to insert. It can be a unary or a binary potential. In case it is binary, at least one of the variable involved must be already inserted to the model before (with a previous Insert having as input a potential which involves that variable).

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

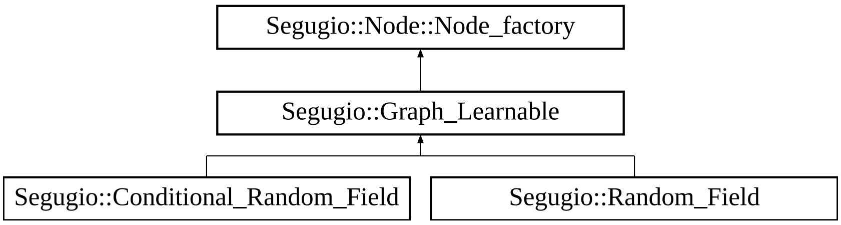
- C:/Users/andre/Desktop/CRF/CRF/Header/Graphical_model.h
- C:/Users/andre/Desktop/CRF/CRF/Source/Graphical_model.cpp

4.19 Segugio::Graph_Learnable Class Reference

Interface for managing learnable graphs, i.e. graphs for which it is possible perform learning.

```
#include <Graphical_model.h>
```

Inheritance diagram for Segugio::Graph_Learnable:


Segugio::Node::Node_factory
Segugio::Graph_Learnable
Segugio::Conditional_Random_Field
Segugio::Random_Field

Classes

- struct Weights_Manager

Public Member Functions

- size_t Get_model_size ()

  Returns the model size, i.e. the number of tunable parameters of the model, i.e. the number of weigths that can vary with learning.
- void **Get_Likelihood_estimation** (float ∗result, const std::list< size_t ∗ > &comb_train_set, const std::list< Categoric_var ∗ > &comb_var_order)
- void Get_structure (std::list< const Potential_Exp_Shape ∗ > ∗result)

  Returns the list of potentials constituting the net. Usefull for structural learning.

Protected Member Functions

- virtual _Pot_wrapper_4_Insertion ∗ **Get_Inserter** (Potential_Exp_Shape ∗pot, const bool &weight_tunability)
- **Graph_Learnable** (const bool &use_cloning_Insert)
- **Graph_Learnable** (const std::list< Potential_Exp_Shape ∗ > &potentials_exp, const bool &use_cloning_↩ Insert, const std::list< bool > &tunable_mask, const std::list< Potential_Shape ∗ > &shapes)

Protected Attributes

- std::list< I_Learning_handler ∗ > **Model_handlers**

4.19.1 Detailed Description

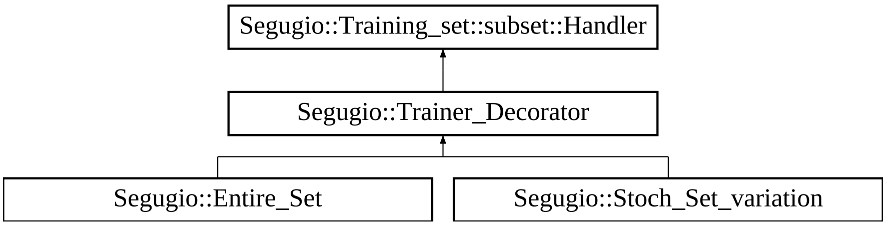
Interface for managing learnable graphs, i.e. graphs for which it is possible perform learning.

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

- C:/Users/andre/Desktop/CRF/CRF/Header/Graphical_model.h
- C:/Users/andre/Desktop/CRF/CRF/Source/Graphical_model.cpp

4.20 Segugio::Training_set::subset::Handler Struct Reference

Inheritance diagram for Segugio::Training_set::subset::Handler:


Segugio::Training_set::subset::Handler
Segugio::Trainer_Decorator
Segugio::Entire_Set
Segugio::Stoch_Set_variation

Static Protected Member Functions

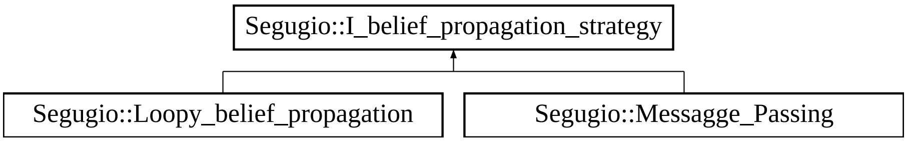
- static std::list< size_t ∗ > ∗ **Get_list** (subset ∗sub_set)
- static std::list< std::string > ∗ **Get_names** (subset ∗sub_set)
- static std::list< std::string > ∗ **Get_names** (Training_set ∗set)

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

- C:/Users/andre/Desktop/CRF/CRF/Header/Training_set.h

4.21 Segugio::I_belief_propagation_strategy Class Reference

Inheritance diagram for Segugio::I_belief_propagation_strategy:


Segugio::I_belief_propagation_strategy
Segugio::Loopy_belief_propagation
Segugio::Messagge_Passing

Static Public Member Functions

- static bool **Propagate** (std::list< Node ∗ > &cluster, const bool &sum_or_MAP=true, const unsigned int &Iterations=1000)

Protected Member Functions

- void **Instantiate_message** (Node::Neighbour_connection ∗outgoing_mex_to_compute, const bool &sum_or_MAP)
- void **Update_message** (float ∗variation_to_previous, Node::Neighbour_connection ∗outgoing_mex_to_compute, const bool &sum_or_MAP)
- void **Gather_incoming_messages** (std::list< Potential ∗ > ∗result, Node::Neighbour_connection ∗outgoing_mex_to_compute)
- std::list< Node::Neighbour_connection ∗ > ∗ **Get_Neighbourhood** (Node::Neighbour_connection ∗conn)
- Message_Unary ∗∗ **Get_Mex_to_This** (Node::Neighbour_connection ∗conn)
- Message_Unary ∗∗ **Get_Mex_to_Neigh** (Node::Neighbour_connection ∗conn)

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

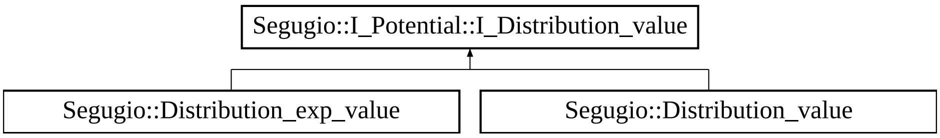
- C:/Users/andre/Desktop/CRF/CRF/Header/Node.h
- C:/Users/andre/Desktop/CRF/CRF/Source/Belief_propagation.cpp

4.22 Segugio::I_Potential::I_Distribution_value Struct Reference

Abstract interface for describing a value in the domain of a potential.

```
#include <Potential.h>
```

Inheritance diagram for Segugio::I_Potential::I_Distribution_value:


Segugio::I_Potential::I_Distribution_value
Segugio::Distribution_exp_value
Segugio::Distribution_value

Public Member Functions

- virtual void **Set_val** (const float &v)=0
- virtual void **Get_val** (float ∗result)=0
- virtual size_t ∗ **Get_indeces** ()=0

4.22.1 Detailed Description

Abstract interface for describing a value in the domain of a potential.

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

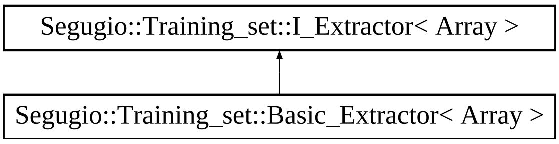
- C:/Users/andre/Desktop/CRF/CRF/Header/Potential.h

4.23 Segugio::Training_set::I_Extractor< Array > Class Template Reference

This class is adopted for parsing a set of samples to import as a novel training set. You have to derive yout custom extractor, implementing the two vritual method.

```
#include <Training_set.h>
```

Inheritance diagram for Segugio::Training_set::I_Extractor< Array >:


Segugio::Training_set::I_Extractor< Array >
Segugio::Training_set::Basic_Extractor< Array >

Public Member Functions

- virtual const size_t & **get_val_in_pos** (const Array &container, const size_t &pos)=0
- virtual size_t **get_size** (const Array &container)=0

4.23.1 Detailed Description

template<typename Array>
class Segugio::Training_set::I_Extractor< Array >

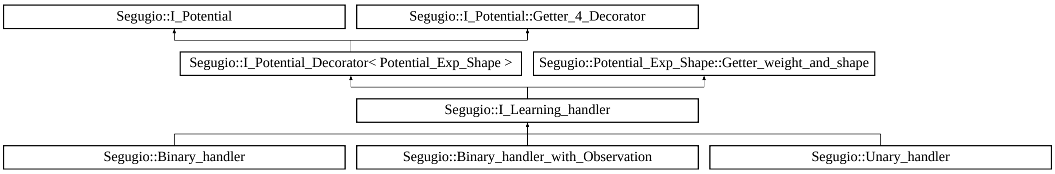
This class is adopted for parsing a set of samples to import as a novel training set. You have to derive yout custom extractor, implementing the two vritual method.

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

- C:/Users/andre/Desktop/CRF/CRF/Header/Training_set.h

4.24 Segugio::I_Learning_handler Class Reference

Inheritance diagram for Segugio::I_Learning_handler:


Segugio::I_Potential
Segugio::I_Potential::Getter_4_Decorator
Segugio::I_Potential_Decorator< Potential_Exp_Shape >
Segugio::Potential_Exp_Shape::Getter_weight_and_shape
Segugio::I_Learning_handler
Segugio::Binary_handler
Segugio::Binary_handler_with_Observation
Segugio::Unary_handler

Public Member Functions

- void **Get_weight** (float ∗w)
- void **Set_weight** (const float &w_new)
- void **Get_grad_alfa_part** (float ∗alfa, const std::list< size_t ∗ > &comb_in_train_set, const std::list< Categoric_var ∗ > &comb_var)
- virtual void **Get_grad_beta_part** (float ∗beta)=0
- const Potential_Exp_Shape ∗ **get_wrapped_exp_pot** ()

Protected Member Functions

- **I_Learning_handler** (Potential_Exp_Shape ∗pot_to_handle)
- **I_Learning_handler** (I_Learning_handler ∗other)

Protected Attributes

- float ∗ **pWeight**
- std::list< I_Distribution_value ∗ > **Extended_shape_domain**

Additional Inherited Members

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

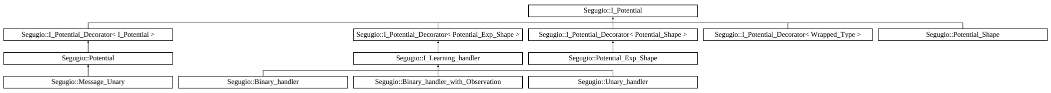
- C:/Users/andre/Desktop/CRF/CRF/Header/Graphical_model.h
- C:/Users/andre/Desktop/CRF/CRF/Source/Graphical_model.cpp

4.25 Segugio::I_Potential Class Reference

Abstract interface for potentials handled by graphs.

```
#include <Potential.h>
```

Inheritance diagram for Segugio::I_Potential:


Segugio::I_Potential
Segugio::I_Potential_Decorator< I_Potential >
Segugio::I_Potential_Decorator< Potential_Exp_Shape >
Segugio::I_Potential_Decorator< Potential_Shape >
Segugio::I_Potential_Decorator< Wrapped_Type >
Segugio::Potential_Shape
Segugio::Potential
Segugio::I_Learning_handler
Segugio::Potential_Exp_Shape
Segugio::Message_Unary
Segugio::Binary_handler
Segugio::Binary_handler_with_Observation
Segugio::Unary_handler

Classes

- struct Getter_4_Decorator
- struct I_Distribution_value

  Abstract interface for describing a value in the domain of a potential.

Public Member Functions

- **I_Potential** (const I_Potential &to_copy)
- void Print_distribution (std::ostream &f, const bool &print_entire_domain=false)

  when print_entire_domain is true, the entire domain is printed, even though the potential has a sparse distribution
- const std::list< Categoric_var ∗ > ∗ Get_involved_var_safe () const

  return list of references to the variables representing the domain of this Potential
- void Find_Comb_in_distribution (std::list< float > ∗result, const std::list< size_t ∗ > &comb_to_search, const std::list< Categoric_var ∗ > &comb_to_search_var_order)
- float max_in_distribution ()

  Returns the maximum value in the distribution describing this potential.

Static Public Member Functions

- static void Get_entire_domain (std::list< std::list< size_t >> ∗domain, const std::list< Categoric_var ∗ > &Vars_in_domain)

  get entire domain of a group of variables: list of possible combinations
- static void Get_entire_domain (std::list< size_t ∗ > ∗domain, const std::list< Categoric_var ∗ > &Vars_in_domain)

  Same as Get_entire_domain(std::list<std::list<size_t>>∗ domain, const std::list<Categoric_var∗>& Vars_in_domain), but adopting array internally allocated with malloc instead of list: remembre to delete combinations.

Protected Member Functions

- virtual const std::list< Categoric_var ∗ > ∗ **Get_involved_var** () const =0
- virtual std::list< I_Distribution_value ∗ > ∗ **Get_distr** ()=0

Static Protected Member Functions

- static void **Find_Comb_in_distribution** (std::list< I_Distribution_value ∗ > ∗result, const std::list< size_t ∗ > &comb_to_search, const std::list< Categoric_var ∗ > &comb_to_search_var_order, I_Potential ∗pot)
- static void **Find_Comb_in_distribution** (std::list< I_Distribution_value ∗ > ∗result, size_t ∗partial_comb_to_search, const std::list< Categoric_var ∗ > &partial_comb_to_search_var_order, I_Potential ∗pot)

4.25.1 Detailed Description

Abstract interface for potentials handled by graphs.

4.25.2 Member Function Documentation

4.25.2.1 Find_Comb_in_distribution()

```
void Segugio::I_Potential::Find_Comb_in_distribution (
            std::list< float > * result,
            const std::list< size_t * > & comb_to_search,
            const std::list< Categoric_var * > & comb_to_search_var_order )
```

Parameters

out	*result*	the list of values matching the combinations to find sent as input
in	*comb_to_search*	domain list of combinations (i.e. values of the domain) whose values are to find
in	*comb_to_search_var_order*	order of variables used for assembling the combinations to find

4.25.2.2 Get_entire_domain() [1/2]

```
void Segugio::I_Potential::Get_entire_domain (
            std::list< std::list< size_t >> * domain,
            const std::list< Categoric_var * > & Vars_in_domain ) [static]
```

get entire domain of a group of variables: list of possible combinations

Parameters

out	*domain*	the entire set of possible combinations
in	*Vars_in_domain*	variables involved whose domain has to be compute

4.25.2.3 Get_entire_domain() [2/2]

```
static void Segugio::I_Potential::Get_entire_domain (
            std::list< size_t * > * domain,
            const std::list< Categoric_var * > & Vars_in_domain ) [static]
```

Same as Get_entire_domain(std::list<std::list<size_t>>∗ domain, const std::list<Categoric_var∗>& Vars_in_domain), but adopting array internally allocated with malloc instead of list: remembre to delete combinations.

Parameters

out	*domain*	the entire set of possible combinations
in	*Vars_in_domain*	variables involved whose domain has to be compute

4.25.2.4 Print_distribution()

```
void Segugio::I_Potential::Print_distribution (
            std::ostream & f,
            const bool & print_entire_domain = false )
```

when print_entire_domain is true, the entire domain is printed, even though the potential has a sparse distribution

Parameters

in	*f*	out stream to target
in	*print_entire_domain*	

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

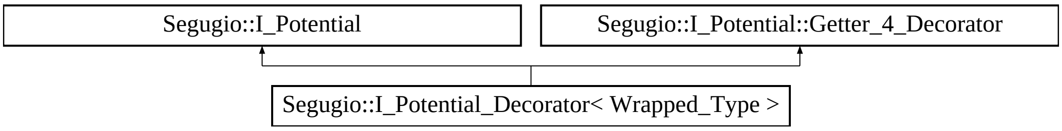
- C:/Users/andre/Desktop/CRF/CRF/Header/Potential.h
- C:/Users/andre/Desktop/CRF/CRF/Source/Potential.cpp

4.26 Segugio::I_Potential_Decorator< Wrapped_Type > Class Template Reference

Abstract decorator of a Potential, wrapping an Abstract potential.

`#include <Potential.h>`

Inheritance diagram for Segugio::I_Potential_Decorator< Wrapped_Type >:


Segugio::I_Potential
Segugio::I_Potential::Getter_4_Decorator
Segugio::I_Potential_Decorator< Wrapped_Type >

Protected Member Functions

- **I_Potential_Decorator** (Wrapped_Type ∗to_wrap)
- virtual const std::list< Categoric_var ∗ > ∗ **Get_involved_var** () const
- virtual std::list< I_Distribution_value ∗ > ∗ **Get_distr** ()

Protected Attributes

- bool **Destroy_wrapped**
- Wrapped_Type ∗ pwrapped

Additional Inherited Members

4.26.1 Detailed Description

template<typename Wrapped_Type>
class Segugio::I_Potential_Decorator< Wrapped_Type >

Abstract decorator of a Potential, wrapping an Abstract potential.

4.26.2 Member Data Documentation

4.26.2.1 pwrapped

```
template<typename Wrapped_Type>
Wrapped_Type* Segugio::I_Potential_Decorator< Wrapped_Type >::pwrapped  [protected]
```

when false, the wrapped abstract potential is wrapped also in another decorator, whihc is in charge of deleting the wrapped potential

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

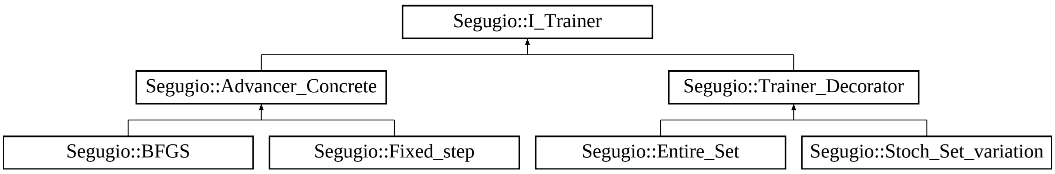
- C:/Users/andre/Desktop/CRF/CRF/Header/Potential.h

4.27 Segugio::I_Trainer Class Reference

This class is used by a Graph_Learnable, to perform training with an instance of a Training_set.

```
#include <Trainer.h>
```

Inheritance diagram for Segugio::I_Trainer:


Segugio::I_Trainer
Segugio::Advancer_Concrete
Segugio::Trainer_Decorator
Segugio::BFGS
Segugio::Fixed_step
Segugio::Entire_Set
Segugio::Stoch_Set_variation

Public Member Functions

- virtual void **Train** (Graph_Learnable ∗model_to_train, Training_set ∗Train_set, const unsigned int &Max_↩ Iterations=100, std::list< float > ∗descend_story=NULL)=0

Static Public Member Functions

- static I_Trainer ∗ Get_fixed_step (const float &step_size=0.1f, const float &stoch_grad_percentage=1.f)

  Creates a fixed step gradient descend solver.
- static I_Trainer ∗ Get_BFGS (const float &stoch_grad_percentage=1.f)

  Creates a BFGS gradient descend solver (`https://en.wikipedia.org/wiki/Broyden%E2%80%93Fletcher%E2%80%93Goldfarb%E2%80%93Shanno_algorithm`*)*

Protected Member Functions

- virtual void **Clean_Up** ()
- void **Get_w_grad** (Graph_Learnable ∗model, std::list< float > ∗grad_w, const std::list< size_t ∗ > &comb_in_train_set, const std::list< Categoric_var ∗ > &comb_var)
- void **Set_w** (const std::list< float > &w, Graph_Learnable ∗model)

Static Protected Member Functions

- static void **Clean_Up** (I_Trainer ∗to_Clean)

4.27.1 Detailed Description

This class is used by a Graph_Learnable, to perform training with an instance of a Training_set.

Instantiate a particular class of trainer to use by calling Get_fixed_step or Get_BFGS. That methods allocate in the heap a trainer to use later, for multiple tranining sessions. Remember to delete the instantiated trainer.

4.27.2 Member Function Documentation

4.27.2.1 Get_BFGS()

```
I_Trainer ∗ Segugio::I_Trainer::Get_BFGS (
             const float & stoch_grad_percentage = 1.f )  [static]
```

Creates a BFGS gradient descend solver (`https://en.wikipedia.org/wiki/Broyden%E2%80%93Fletcher%E2%80%93Goldfarb%E2%80%93Shanno_algorithm`)

Parameters

in	*stoch_grad_percentage*	percentage of the training set to use every time for evaluating the gradient

4.27.2.2 Get_fixed_step()

```
I_Trainer ∗ Segugio::I_Trainer::Get_fixed_step (
            const float & step_size = 0.1f,
            const float & stoch_grad_percentage = 1.f )  [static]
```

Creates a fixed step gradient descend solver.

Parameters

in	*step_size*	learinig degree
in	*stoch_grad_percentage*	percentage of the training set to use every time for evaluating the gradient

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

- C:/Users/andre/Desktop/CRF/CRF/Header/Trainer.h
- C:/Users/andre/Desktop/CRF/CRF/Source/Trainer.cpp

4.28 Segugio::info_neighbourhood::info_neigh Struct Reference

Public Attributes

- Potential ∗ **shared_potential**
- Categoric_var ∗ **Var**
- size_t **Var_pos**

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

- C:/Users/andre/Desktop/CRF/CRF/Source/Node.cpp

4.29 Segugio::info_neighbourhood Struct Reference

Classes

- struct info_neigh

Public Attributes

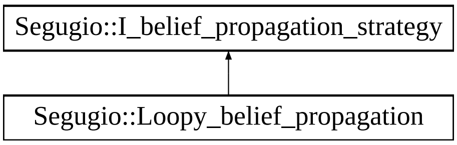
- size_t **Involved_var_pos**
- list< info_neigh > **Info**
- list< Potential ∗ > **Unary_potentials**

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

- C:/Users/andre/Desktop/CRF/CRF/Source/Node.cpp

4.30 Segugio::Loopy_belief_propagation Class Reference

Inheritance diagram for Segugio::Loopy_belief_propagation:


Segugio::I_belief_propagation_strategy
Segugio::Loopy_belief_propagation

Public Member Functions

- **Loopy_belief_propagation** (const int &max_iter)
- bool **_propagate** (std::list< Node ∗ > &cluster, const bool &sum_or_MAP)

Protected Attributes

- unsigned int **Iter**

Additional Inherited Members

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

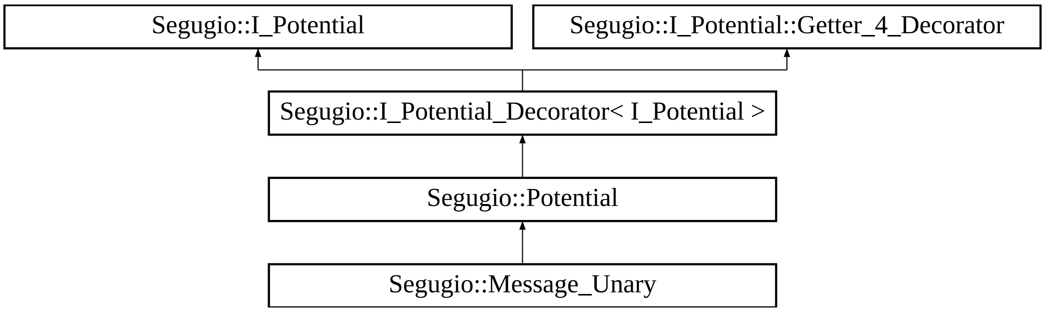
- C:/Users/andre/Desktop/CRF/CRF/Header/Belief_propagation.h
- C:/Users/andre/Desktop/CRF/CRF/Source/Belief_propagation.cpp

4.31 Segugio::Message_Unary Class Reference

This class is adopted by belief propagation algorithms. It is the message incoming to a node of the graph. Every node of a graph refers to a single Categorical variable. Internally it keeps track of the difference in time of the messages produced, in order to arrest loopy belief propagation.

```
#include <Potential.h>
```

Inheritance diagram for Segugio::Message_Unary:


Segugio::I_Potential
Segugio::I_Potential::Getter_4_Decorator
Segugio::I_Potential_Decorator< I_Potential >
Segugio::Potential
Segugio::Message_Unary

Public Member Functions

- Message_Unary (Categoric_var ∗var_involved)

  Creates a Message with all 1 as values for the image.
- Message_Unary (Potential ∗binary_to_merge, const std::list< Potential ∗ > &potential_to_merge, const bool &Sum_or_MAP=true)

  Firstly, all potential_to_merge are merged together using Potential::Potential(potential_to_merge, false) obtaining a merged potential. Secondly, the product of binary_to_merge and the merged potential is obtained. Finally the message is obtained by marginalizing from the second product, the variable of potential_to_merge, adopting a sum or a MAP. Exploited by message passing algorithms.
- Message_Unary (Potential ∗binary_to_merge, Categoric_var ∗var_to_marginalize, const bool &Sum_or_←MAP=true)

  Same as Message_Unary::Message_Unary(Potential∗ binary_to_merge, const std::list<Potential∗>& potential_←to_merge, const bool& Sum_or_MAP = true), but in the case potential_to_merge is empty.
- void Update (float ∗diff_to_previous, Potential ∗binary_to_merge, const std::list< Potential ∗ > &potential←_to_merge, const bool &Sum_or_MAP=true)

  Adopted by loopy belief propagation.
- void Update (float ∗diff_to_previous, Potential ∗binary_to_merge, Categoric_var ∗var_to_marginalize, const bool &Sum_or_MAP=true)

  Adopted by loopy belief propagation.

Additional Inherited Members

4.31.1 Detailed Description

This class is adopted by belief propagation algorithms. It is the message incoming to a node of the graph. Every node of a graph refers to a single Categorical variable. Internally it keeps track of the difference in time of the messages produced, in order to arrest loopy belief propagation.

4.31.2 Constructor & Destructor Documentation

4.31.2.1 Message_Unary() [1/2]

```
Segugio::Message_Unary::Message_Unary (
            Categoric_var ∗ var_involved )
```

Creates a Message with all 1 as values for the image.

Parameters

in	*var_involved*	the only variable in the domain

4.31.2.2 Message_Unary() [2/2]

```
Segugio::Message_Unary::Message_Unary (
            Potential * binary_to_merge,
            const std::list< Potential * > & potential_to_merge,
            const bool & Sum_or_MAP = true )
```

Firstly, all potential_to_merge are merged together using Potential::Potential(potential_to_merge, false) obtaining a merged potential. Secondly, the product of binary_to_merge and the merged potential is obtained. Finally the message is obtained by marginalizing from the second product, the variable of potential_to_merge, adopting a sum or a MAP. Exploited by message passing algorithms.

Parameters

in	*binary_to_merge*	binaty potential to consider
in	*potential_to_merge*	list of potentials to merge. The must be unary potentials

4.31.3 Member Function Documentation

4.31.3.1 Update() [1/2]

```
void Segugio::Message_Unary::Update (
            float * diff_to_previous,
            Potential * binary_to_merge,
            const std::list< Potential * > & potential_to_merge,
            const bool & Sum_or_MAP = true )
```

Adopted by loopy belief propagation.

Parameters

out	*diff_to_previous*	The difference with respect to the previous message camptation

4.31.3.2 Update() [2/2]

```
void Segugio::Message_Unary::Update (
            float * diff_to_previous,
            Potential * binary_to_merge,
            Categoric_var * var_to_marginalize,
            const bool & Sum_or_MAP = true )
```

Adopted by loopy belief propagation.

Parameters

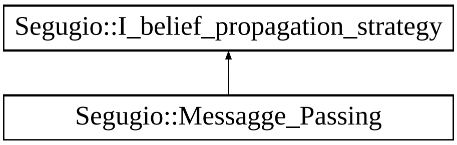
out	*diff_to_previous*	The difference with respect to the previous message camptation

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

- C:/Users/andre/Desktop/CRF/CRF/Header/Potential.h
- C:/Users/andre/Desktop/CRF/CRF/Source/Potential.cpp

4.32 Segugio::Messagge_Passing Class Reference

Inheritance diagram for Segugio::Messagge_Passing:


Segugio::I_belief_propagation_strategy
Segugio::Messagge_Passing

Public Member Functions

- bool **_propagate** (std::list< Node ∗ > &cluster, const bool &sum_or_MAP)

Additional Inherited Members

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

- C:/Users/andre/Desktop/CRF/CRF/Header/Belief_propagation.h
- C:/Users/andre/Desktop/CRF/CRF/Source/Belief_propagation.cpp

4.33 Segugio::Node::Neighbour_connection Struct Reference

Friends

- class **Node**
- class **I_belief_propagation_strategy**

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

- C:/Users/andre/Desktop/CRF/CRF/Header/Node.h
- C:/Users/andre/Desktop/CRF/CRF/Source/Node.cpp

4.34 Segugio::Node Class Reference

Classes

- struct Neighbour_connection
- class Node_factory

  Interface for describing a net: set of nodes representing random variables.

Public Member Functions

- Categoric_var ∗ **Get_var** ()
- void **Gather_all_Unaries** (std::list< Potential ∗ > ∗result)
- void **Append_temporary_permanent_Unaries** (std::list< Potential ∗ > ∗result)
- void **Append_permanent_Unaries** (std::list< Potential ∗ > ∗result)
- const std::list< Neighbour_connection ∗ > ∗ **Get_Active_connections** ()
- void **Compute_neighbour_set** (std::list< Node ∗ > ∗Neigh_set)
- void **Compute_neighbour_set** (std::list< Node ∗ > ∗Neigh_set, std::list< Potential ∗ > ∗binary_involved)
- void **Compute_neighbourhood_messages** (std::list< Potential ∗ > ∗messages, Node ∗node_involved_↩
in_connection)

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

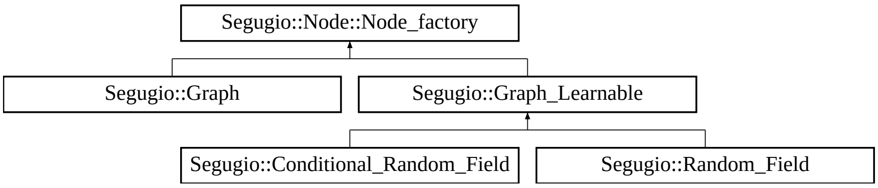
- C:/Users/andre/Desktop/CRF/CRF/Header/Node.h
- C:/Users/andre/Desktop/CRF/CRF/Source/Node.cpp

4.35 Segugio::Node::Node_factory Class Reference

Interface for describing a net: set of nodes representing random variables.

```
#include <Node.h>
```

Inheritance diagram for Segugio::Node::Node_factory:


Segugio::Node::Node_factory
Segugio::Graph
Segugio::Graph_Learnable
Segugio::Conditional_Random_Field
Segugio::Random_Field

Classes

- struct _Baseline_4_Insertion
- struct _Pot_wrapper_4_Insertion
- class _SubGraph

Public Member Functions

- Categoric_var ∗ Find_Variable (const std::string &var_name)

  Returns a pointer to the variable in this graph with that name.
- Categoric_var ∗ Find_Variable (Categoric_var ∗var_with_same_name)

  Returns a pointer to the variable in this graph with the same name of the variable passed as input.
- void Get_Actual_Hidden_Set (std::list< Categoric_var ∗ > ∗result)

  Returns the current set of hidden variables.
- void Get_Actual_Observation_Set (std::list< Categoric_var ∗ > ∗result)

  Returns the current set of observed variables.
- void Get_All_variables_in_model (std::list< Categoric_var ∗ > ∗result)

Returns the set of all variable contained in the net.

- void Get_marginal_distribution (std::list< float > ∗result, Categoric_var ∗var)

  Returns the marginal probabilty of the variable passed P(var|model, observations),.

- void MAP_on_Hidden_set (std::list< size_t > ∗result)

  Returns the Maximum a Posteriori estimation of the hidden set.

- void Gibbs_Sampling_on_Hidden_set (std::list< std::list< size_t >> ∗result, const unsigned int &N_↩samples, const unsigned int &initial_sample_to_skip)

  Returns a set of samples of the conditional distribution P(hidden variables | model, observed variables).

- unsigned int Get_Iteration_4_belief_propagation ()

  Returns the current value adopted when performing a loopy belief propagation.

- void Set_Iteration_4_belief_propagation (const unsigned int &iter_to_use)

  Returns the value to adopt when performing a loopy belief propagation.

- void Eval_Log_Energy_function (float ∗result, size_t ∗combination, const std::list< Categoric_var ∗ > &var↩_order_in_combination)

  Returns the logartihmic value of the energy function.

- void Eval_Log_Energy_function (float ∗result, const std::list< size_t > &combination, const std::list< Categoric_var ∗ > &var_order_in_combination)

  Same as Eval_Log_Energy_function(float∗ result, size_t∗ combination, const std::list<Categoric_var∗>& var_order_in_combination), passing a list instead of an array size_t∗, a list<size_t> for describing the combination for which you want to evaluate the energy.

- void Eval_Log_Energy_function (std::list< float > ∗result, const std::list< size_t ∗ > &combinations, const std::list< Categoric_var ∗ > &var_order_in_combination)

  Same as Eval_Log_Energy_function(float∗ result, size_t∗ combination, const std::list<Categoric_var∗>& var_order_in_combination), passing a list of combinations: don't iterate yourself many times using Eval_Log_Energy_function(float∗ result, size_t∗ combination, const but call this function.

- void Eval_Log_Energy_function_normalized (float ∗result, size_t ∗combination, const std::list< Categoric_var ∗ > &var_order_in_combination)

  Similar as Eval_Log_Energy_function(float∗ result, size_t∗ combination, const std::list<Categoric_var∗>& var_order_in_combination), but computing the Energy function normalized: E_norm = E(Y_1,2,....,n) / max possible { E }. E_norm is in [0,1]. The logarthmic value of E_norm is actually returned.

- void Eval_Log_Energy_function_normalized (float ∗result, const std::list< size_t > &combination, const std::list< Categoric_var ∗ > &var_order_in_combination)

  Similar as Eval_Log_Energy_function(float∗ result, const std::list<size_t>& combination, const std::list<Categoric_var∗>& var_order_i but computing the Energy function normalized.

- void Eval_Log_Energy_function_normalized (std::list< float > ∗result, const std::list< size_t ∗ > &combinations, const std::list< Categoric_var ∗ > &var_order_in_combination)

  Similar as Eval_Log_Energy_function(std::list<float>∗ result, const std::list<size_t∗>& combinations, const std::list<Categoric_var∗>& but computing the Energy function normalized.

- void Get_Observation_Set_val (std::list< size_t > ∗result)

  Returns the attual values set observations. This function can be invokated after a call to void Set_Observation_Set_val(const std::list<size

- void Get_structure (std::list< const Potential ∗ > ∗structure)

  Returns the list of potentials constituting the net. Usefull for structural learning.

- size_t Get_structure_size ()

  Returns the number of potentials constituting the graph, no matter of their type (simple shape, exponential shape fixed or exponential shape tunable)

Protected Member Functions

- **Node_factory** (const bool &use_cloning_Insert)
- void **Import_from_XML** (XML_reader ∗xml_data, const std::string &prefix_config_xml_file)
- void **Insert** (_Pot_wrapper_4_Insertion ∗element_to_add)
- void **Insert** (std::list< _Pot_wrapper_4_Insertion ∗ > &elements_to_add)
- virtual _Pot_wrapper_4_Insertion ∗ **Get_Inserter** (Potential_Exp_Shape ∗pot, const bool &weight_tunability)

- void **Insert** (Potential_Shape ∗pot)
- void **Insert** (Potential_Exp_Shape ∗pot, const bool &weight_tunability)
- Node ∗ **Find_Node** (const std::string &var_name)
- void Set_Observation_Set_var (const std::list< Categoric_var ∗ > &new_observed_vars)

  Set the values for the observations. Must call after calling Node_factory::Set_Observation_Set_val.
- void Set_Observation_Set_val (const std::list< size_t > &new_observed_vals)

  Set the observation set: which variables are treated like evidence when performing belief propagation.
- void **Belief_Propagation** (const bool &sum_or_MAP)
- size_t ∗ **Get_observed_val_in_case_is_in_observed_set** (Categoric_var ∗var)

4.35.1 Detailed Description

Interface for describing a net: set of nodes representing random variables.

4.35.2 Member Function Documentation

4.35.2.1 Eval_Log_Energy_function()

```
void Segugio::Node::Node_factory::Eval_Log_Energy_function (
             float ∗ result,
             size_t ∗ combination,
             const std::list< Categoric_var ∗ > & var_order_in_combination )
```

Returns the logartihmic value of the energy function.

Energy function E=Pot_1(Y_1,2,....,n) ∗ Pot_2(Y_1,2,....,n) .... ∗ Pot_m(Y_1,2,....,n). The combinations passed as input must contains values for all the variables present in this graph.

Parameters

out	*result*	
in	*combination*	set of values in the combination for which the energy function has to be eveluated
in	*var_order_in_combination*	order of variables considered when assembling combination. They must be references to the variables actually wrapped by this graph.

4.35.2.2 Find_Variable() [1/2]

```
Categoric_var ∗ Segugio::Node::Node_factory::Find_Variable (
             const std::string & var_name )
```

Returns a pointer to the variable in this graph with that name.

Returns NULL when the variable is not present in the graph.

Parameters

in	*var_name*	name to search

4.35.2.3 Find_Variable() [2/2]

```
Categoric_var* Segugio::Node::Node_factory::Find_Variable (
            Categoric_var * var_with_same_name )  [inline]
```

Returns a pointer to the variable in this graph with the same name of the variable passed as input.

Returns NULL when the variable is not present in the graph

Parameters

in	*var_with_same_name*	variable having the same of name of the variable to search

4.35.2.4 Get_marginal_distribution()

```
void Segugio::Node::Node_factory::Get_marginal_distribution (
            std::list< float > * result,
            Categoric_var * var )
```

Returns the marginal probabilty of the variable passed P(var|model, observations),.

on the basis of the last observations set (see Node_factory::Set_Observation_Set_var)

4.35.2.5 Gibbs_Sampling_on_Hidden_set()

```
void Segugio::Node::Node_factory::Gibbs_Sampling_on_Hidden_set (
            std::list< std::list< size_t >> * result,
            const unsigned int & N_samples,
            const unsigned int & initial_sample_to_skip )
```

Returns a set of samples of the conditional distribution P(hidden variables | model, observed variables).

Samples are obtained through Gibbs sampling. Calculations are done considering the last last observations set (see Node_factory::Set_Observation_Set_var)

Parameters

in	*N_samples*	number of desired samples
in	*initial_sample_to_skip*	number of samples to skip for performing Gibbs sampling
out	*result*	returned samples: every element of the list is a combination of values for the hidden set, with the same order returned when calling Node_factory::Get_Actual_Hidden_Set

4.35.2.6 MAP_on_Hidden_set()

```
void Segugio::Node::Node_factory::MAP_on_Hidden_set (
             std::list< size_t > * result )
```

Returns the Maximum a Posteriori estimation of the hidden set.

Values are ordered as returned by Node_factory::Get_Actual_Hidden_Set. Calculations are done considering the last last observations set (see Node_factory::Set_Observation_Set_var)

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

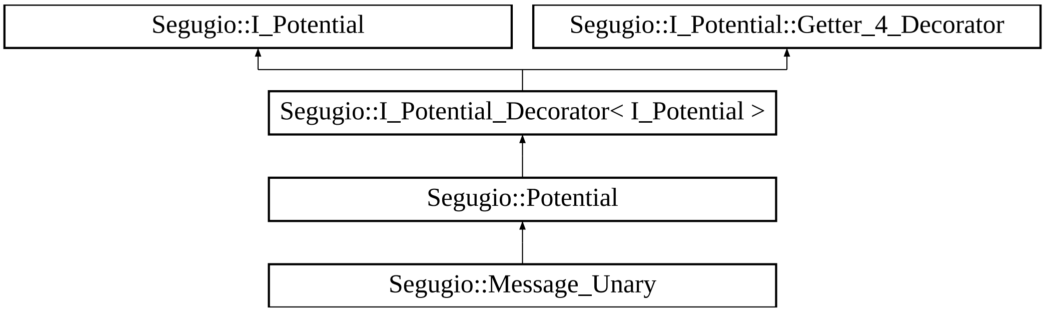
- C:/Users/andre/Desktop/CRF/CRF/Header/Node.h
- C:/Users/andre/Desktop/CRF/CRF/Source/Node.cpp

4.36 Segugio::Potential Class Reference

This class is mainly adopted for computing operations on potentials.

```
#include <Potential.h>
```

Inheritance diagram for Segugio::Potential:


Segugio::I_Potential
Segugio::I_Potential::Getter_4_Decorator
Segugio::I_Potential_Decorator< I_Potential >
Segugio::Potential
Segugio::Message_Unary

Public Member Functions

- Potential (Potential_Shape ∗pot)
- Potential (Potential_Exp_Shape ∗pot)
- Potential (const std::list< Potential ∗ > &potential_to_merge, const bool &use_sparse_format=true)

    The potential to create is obtained by merging a set of potentials referring to the same variables (i.e. values in the image are obtained as a product of the ones in the potential_to_merge set)
- Potential (const std::list< size_t > &val_observed, const std::list< Categoric_var ∗ > &var_observed, Potential ∗pot_to_reduce)

    The potential to create is obtained by marginalizing the observed variable passed as input.
- void Get_marginals (std::list< float > ∗prob_distr)

    Obtain the marginal probabilities of the variables in the domain of this potential, when considering this potential only.

Additional Inherited Members

4.36.1 Detailed Description

This class is mainly adopted for computing operations on potentials.

4.36.2 Constructor & Destructor Documentation

4.36.2.1 Potential() [1/4]

```
Segugio::Potential::Potential (
             Potential_Shape * pot )  [inline]
```

Parameters

in	*pot*	potential shape to wrap

4.36.2.2 Potential() [2/4]

```
Segugio::Potential::Potential (
             Potential_Exp_Shape * pot )  [inline]
```

Parameters

in	*pot*	exponential potential shape to wrap

4.36.2.3 Potential() [3/4]

```
Segugio::Potential::Potential (
             const std::list< Potential * > & potential_to_merge,
             const bool & use_sparse_format = true )
```

The potential to create is obtained by merging a set of potentials referring to the same variables (i.e. values in the image are obtained as a product of the ones in the potential_to_merge set)

Parameters

in	*potential_to_merge*	list of potential to merge, i.e. compute their product
in	*use_sparse_format*	when false, the entire domain is allocated even if some values are equal to 0

4.36.2.4 Potential() [4/4]

```
Segugio::Potential::Potential (
            const std::list< size_t > & val_observed,
            const std::list< Categoric_var * > & var_observed,
            Potential * pot_to_reduce )
```

The potential to create is obtained by marginalizing the observed variable passed as input.

Parameters

in	*pot_to_reduce*	the potential from which the variables observed are marginalized
in	*var_observed*	variables observed in pot_to_reduce
in	*val_observed*	values observed (same oreder of var_observed)

4.36.3 Member Function Documentation

4.36.3.1 Get_marginals()

```
void Segugio::Potential::Get_marginals (
            std::list< float > * prob_distr )
```

Obtain the marginal probabilities of the variables in the domain of this potential, when considering this potential only.

Parameters

in	*prob_distr*	marginals

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

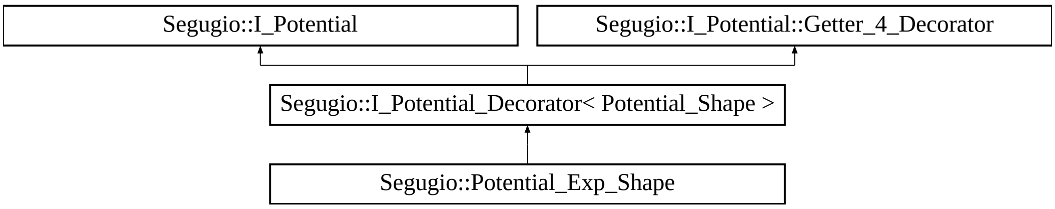
- C:/Users/andre/Desktop/CRF/CRF/Header/Potential.h
- C:/Users/andre/Desktop/CRF/CRF/Source/Potential.cpp

4.37 Segugio::Potential_Exp_Shape Class Reference

Represents an exponential potential, wrapping a normal shape one: every value of the domain are assumed as exp(mWeight ∗ val_in_shape_wrapped)

```
#include <Potential.h>
```

Inheritance diagram for Segugio::Potential_Exp_Shape:


Segugio::I_Potential
Segugio::I_Potential::Getter_4_Decorator
Segugio::I_Potential_Decorator< Potential_Shape >
Segugio::Potential_Exp_Shape

Classes

- struct Getter_weight_and_shape

Public Member Functions

- Potential_Exp_Shape (Potential_Shape ∗shape, const float &w=1.f)

  When building a new exponential shape potential, all the values of the domain are computed according to the new shape passed as input.
- Potential_Exp_Shape (const std::list< Categoric_var ∗ > &var_involved, const std::string &file_to_read, const float &w=1.f)

  When building a new exponential shape potential, all the values of the domain are computed according to the potential shape to wrap, which is instantiated in the constructor by considering the textual file provided, see also Potential_↩ Shape(const std::list<Categoric_var∗>& var_involved, const std::string& file_to_read)
- Potential_Exp_Shape (const Potential_Exp_Shape ∗to_copy, const std::list< Categoric_var ∗ > &var_↩ involved)
- void Substitute_variables (const std::list< Categoric_var ∗ > &new_var)

  Use this method for replacing the set of variables this potential must refer. Variables in new_var must be equal in number to the original set of variables and must have the same sizes.

Protected Member Functions

- virtual std::list< I_Distribution_value ∗ > ∗ **Get_distr** ()
- void **Wrap** (Potential_Shape ∗shape)

Protected Attributes

- float **mWeight**
- std::list< I_Distribution_value ∗ > Distribution

Additional Inherited Members

4.37.1 Detailed Description

Represents an exponential potential, wrapping a normal shape one: every value of the domain are assumed as exp(mWeight ∗ val_in_shape_wrapped)

4.37.2 Constructor & Destructor Documentation

4.37.2.1 Potential_Exp_Shape() [1/3]

```
Segugio::Potential_Exp_Shape::Potential_Exp_Shape (
            Potential_Shape * shape,
            const float & w = 1.f )
```

When building a new exponential shape potential, all the values of the domain are computed according to the new shape passed as input.

Parameters

in	*shape*	shape distribution to wrap
in	*w*	weight of the exponential

4.37.2.2 Potential_Exp_Shape() [2/3]

```
Segugio::Potential_Exp_Shape::Potential_Exp_Shape (
            const std::list< Categoric_var * > & var_involved,
            const std::string & file_to_read,
            const float & w = 1.f )
```

When building a new exponential shape potential, all the values of the domain are computed according to the potential shape to wrap, which is instantiated in the constructor by considering the textual file provided, see also Potential_Shape(const std::list<Categoric_var*>& var_involved, const std::string& file_to_read)

Parameters

in	*var_involved*	variables involved in the domain of this variables
in	*file_to_read*	textual file to read containing the values for the image
in	*w*	weight of the exponential

4.37.2.3 Potential_Exp_Shape() [3/3]

```
Segugio::Potential_Exp_Shape::Potential_Exp_Shape (
            const Potential_Exp_Shape * to_copy,
            const std::list< Categoric_var * > & var_involved )
```

Use this constructor for cloning an exponential shape, but considering a different set of variables. Variables in var_involved must be equal in number to those in the potential to clone and must have the same sizes of the variables involved in the potential to clone.

Parameters

in	*to_copy*	shape to clone
in	*var_involved*	new set of variables to consider when cloning

4.37.3 Member Function Documentation

4.37.3.1 Substitute_variables()

```
void Segugio::Potential_Exp_Shape::Substitute_variables (
            const std::list< Categoric_var * > & new_var )  [inline]
```

Use this method for replacing the set of variables this potential must refer. Variables in new_var must be equal in number to the original set of variables and must have the same sizes.

Parameters

in	*new_var*	variables to consider for the substitution

4.37.4 Member Data Documentation

4.37.4.1 Distribution

```
std::list<I_Distribution_value*> Segugio::Potential_Exp_Shape::Distribution  [protected]
```

Weight assumed for modulating the exponential (see description of the class)

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

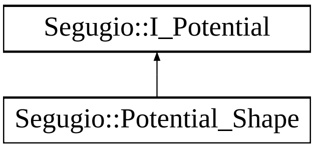
- C:/Users/andre/Desktop/CRF/CRF/Header/Potential.h
- C:/Users/andre/Desktop/CRF/CRF/Source/Potential.cpp

4.38 Segugio::Potential_Shape Class Reference

It's the only possible concrete potential. It contains the domain and the image of the potential.

```
#include <Potential.h>
```

Inheritance diagram for Segugio::Potential_Shape:



Public Member Functions

- Potential_Shape (const std::list< Categoric_var ∗ > &var_involved)

  When building a new shape potential, all values of the image are assumed as all zeros.
- Potential_Shape (const std::list< Categoric_var ∗ > &var_involved, const std::string &file_to_read)
- Potential_Shape (const std::list< Categoric_var ∗ > &var_involved, const bool &correlated_or_not)

  Returns simple correlating or anti_correlating shapes.
- Potential_Shape (const Potential_Shape ∗to_copy, const std::list< Categoric_var ∗ > &var_involved)
- **Potential_Shape** (const Potential_Shape &to_copy)
- void Import (const std::string &file_to_read)

  For populating the image of the domain with the values reported in the textual file.
- void Add_value (const std::list< size_t > &new_indeces, const float &new_val)

  Add a new value in the image set.
- void Set_ones ()

  All values in the image of the domain are set to 1.
- void Set_random (const float zeroing_threashold=1.f)

  All values in the image of the domain are randomly set.
- void Normalize_distribution ()

  All values in the image of the domain are multipled by a scaling factor, in order to to have maximal value equal to 1. Exploited for computing messages.
- void Substitute_variables (const std::list< Categoric_var ∗ > &new_var)

  Use this method for replacing the set of variables this potential must refer. Variables in new_var must be equal in number to the original set of variables and must have the same sizes.

Protected Member Functions

- void **Check_add_value** (const std::list< size_t > &indices)
- virtual const std::list< Categoric_var ∗ > ∗ **Get_involved_var** () const
- virtual std::list< I_Distribution_value ∗ > ∗ **Get_distr** ()

Additional Inherited Members

4.38.1 Detailed Description

It's the only possible concrete potential. It contains the domain and the image of the potential.

4.38.2 Constructor & Destructor Documentation

4.38.2.1 Potential_Shape() [1/4]

```
Segugio::Potential_Shape::Potential_Shape (
            const std::list< Categoric_var * > & var_involved )
```

When building a new shape potential, all values of the image are assumed as all zeros.

Parameters

in	*var_involved*	variables involved in the domain of this variables

4.38.2.2 Potential_Shape() [2/4]

```
Segugio::Potential_Shape::Potential_Shape (
            const std::list< Categoric_var * > & var_involved,
            const std::string & file_to_read )
```

Parameters

in	*var_involved*	variables involved in the domain of this variables
in	*file_to_read*	textual file to read containing the values for the image

4.38.2.3 Potential_Shape() [3/4]

```
Segugio::Potential_Shape::Potential_Shape (
            const std::list< Categoric_var * > & var_involved,
            const bool & correlated_or_not )
```

Returns simple correlating or anti_correlating shapes.

A simple correlating shape is a distribution having a value of 1 for every combinations {0,0,...,0}; {1,1,...,1} etc. and 0 for all other combinations. A simple anti_correlating shape is a distribution having a value of 0 for every combinations {0,0,...,0}; {1,1,...,1} etc. and 1 for all other combinations.

Parameters

in	*var_involved*	variables involved in the domain of this variables: they must have all the same size
in	*correlated_or_not*	when true produce a simple correlating shape, when false produce a anti_correlating function

4.38.2.4 Potential_Shape() [4/4]

```
Segugio::Potential_Shape::Potential_Shape (
            const Potential_Shape * to_copy,
            const std::list< Categoric_var * > & var_involved )
```

Use this constructor for cloning a shape, but considering a different set of variables. Variables in var_involved must be equal in number to those in the potential to clone and must have the same sizes of the variables involved in the potential to clone.

Parameters

in	*to_copy*	shape to clone
in	*var_involved*	new set of variables to consider when cloning

4.38.3 Member Function Documentation

4.38.3.1 Add_value()

```
void Segugio::Potential_Shape::Add_value (
            const std::list< size_t > & new_indeces,
            const float & new_val )
```

Add a new value in the image set.

Parameters

in	*new_indices*	combination related to the new value to add for the image
in	*new_val*	new val to insert

4.38.3.2 Import()

```
void Segugio::Potential_Shape::Import (
            const std::string & file_to_read )
```

For populating the image of the domain with the values reported in the textual file.

Parameters

in	*file_to_read*	textual file to read containing the values for the image

4.38.3.3 Substitute_variables()

```
void Segugio::Potential_Shape::Substitute_variables (
            const std::list< Categoric_var * > & new_var )
```

Use this method for replacing the set of variables this potential must refer. Variables in new_var must be equal in number to the original set of variables and must have the same sizes.

Parameters

in	*new_var*	variables to consider for the substitution

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

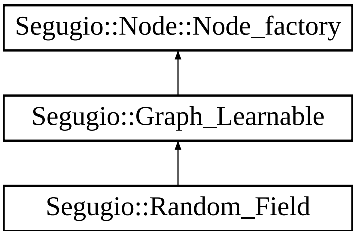
- C:/Users/andre/Desktop/CRF/CRF/Header/Potential.h
- C:/Users/andre/Desktop/CRF/CRF/Source/Potential.cpp

4.39 Segugio::Random_Field Class Reference

This class describes a generic Random Field, not having a particular set of variables observed.

```
#include <Graphical_model.h>
```

Inheritance diagram for Segugio::Random_Field:



Public Member Functions

- Random_Field (const bool &use_cloning_Insert=true)

  empty constructor
- Random_Field (const std::string &config_xml_file, const std::string &prefix_config_xml_file="")

  The model is built considering the information contained in an xml configuration file.
- Random_Field (const std::list< Potential_Exp_Shape * > &potentials_exp, const bool &use_cloning_Insert=true, const std::list< bool > &tunable_mask={}, const std::list< Potential_Shape * > &shapes={})

  This constructor initializes the graph with the specified potentials passed as input.
- void Insert (Potential_Shape *pot)

  Similar to Graph::Insert(Potential_Shape pot)*
- void Insert (Potential_Exp_Shape *pot, const bool &is_weight_tunable=true)

  Similar to Graph::Insert(Potential_Exp_Shape pot).*
- void Set_Observation_Set_var (const std::list< Categoric_var * > &new_observed_vars)

  see Node::Node_factory::Set_Observation_Set_var(const std::list<Categoric_var>& new_observed_vars)*
- void Set_Observation_Set_val (const std::list< size_t > &new_observed_vals)

  see Node::Node_factory::Set_Observation_Set_val(const std::list<size_t>& new_observed_vals)

Additional Inherited Members

4.39.1 Detailed Description

This class describes a generic Random Field, not having a particular set of variables observed.

4.39.2 Constructor & Destructor Documentation

4.39.2.1 Random_Field() [1/3]

```
Segugio::Random_Field::Random_Field (
            const bool & use_cloning_Insert = true )  [inline]
```

empty constructor

Parameters

in	*use_cloning_Insert*	when is true, every time an Insert of a novel potential is called, a copy of that potential is actually inserted. Otherwise, the passed potential is inserted as is: this can be dangerous, cause that potential cna be externally modified, but the construction of a novel graph is faster.

4.39.2.2 Random_Field() [2/3]

```
Segugio::Random_Field::Random_Field (
            const std::string & config_xml_file,
            const std::string & prefix_config_xml_file = "" )
```

The model is built considering the information contained in an xml configuration file.

TODO spiegare come e' fatto xml

Parameters

in	*configuration*	file
in	*prefix*	to use. The file prefix_config_xml_file/config_xml_file is searched.

4.39.2.3 Random_Field() [3/3]

```
Segugio::Random_Field::Random_Field (
            const std::list< Potential_Exp_Shape * > & potentials_exp,
            const bool & use_cloning_Insert = true,
            const std::list< bool > & tunable_mask = {},
            const std::list< Potential_Shape * > & shapes = {} )
```

This constructor initializes the graph with the specified potentials passed as input.

Parameters

in	*potentials_exp*	the initial set of exponential potentials to insert (can be empty)
in	*use_cloning_Insert*	when is true, every time an Insert of a novel potential is called (this includes the passed potentials), a copy of that potential is actually inserted. Otherwise, the passed potential is inserted as is: this can be dangerous, cause that potential cna be externally modified, but the construction of a novel graph is faster.
in	*tunable_mask*	when passed as non default value, it is must have the same size of potentials. Every value in this list is true if the corresponfing potential in the potentials list is tunable, i.e. has a weight whose value can vary with learning
in	*shapes*	A list of additional non learnable potentials to insert in the model

4.39.3 Member Function Documentation

4.39.3.1 Insert()

```
void Segugio::Random_Field::Insert (
            Potential_Exp_Shape * pot,
            const bool & is_weight_tunable = true )  [inline]
```

Similar to Graph::Insert(Potential_Exp_Shape∗ pot).

Parameters

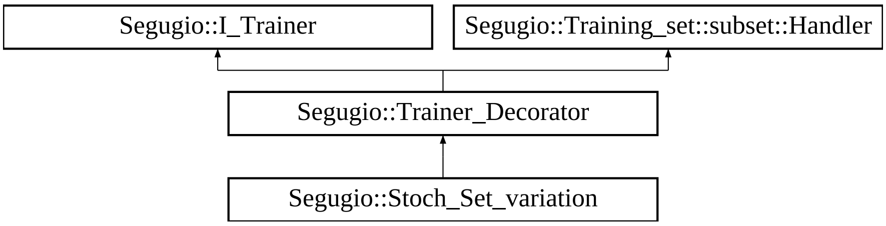
in	*is_weight_tunable*	When true, you are specifying that this potential has a weight learnable, otherwise the value of the weight is assumed constant.

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

- C:/Users/andre/Desktop/CRF/CRF/Header/Graphical_model.h
- C:/Users/andre/Desktop/CRF/CRF/Source/Graphical_model.cpp

4.40 Segugio::Stoch_Set_variation Class Reference

Inheritance diagram for Segugio::Stoch_Set_variation:


Segugio::I_Trainer
Segugio::Training_set::subset::Handler
Segugio::Trainer_Decorator
Segugio::Stoch_Set_variation

Public Member Functions

- **Stoch_Set_variation** (Advancer_Concrete ∗to_wrap, const float &percentage_to_use)
- void **Train** (Graph_Learnable ∗model_to_train, Training_set ∗Train_set, const unsigned int &Max_Iterations, std::list< float > ∗descend_story)

Additional Inherited Members

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

- C:/Users/andre/Desktop/CRF/CRF/Source/Trainer.cpp

4.41 Segugio::Training_set::subset Struct Reference

This class is describes a portion of a training set, obtained by sampling values in the original set. Mainly used by stochastic gradient computation strategies.

```
#include <Training_set.h>
```

Classes

- struct Handler

Public Member Functions

- subset (Training_set ∗set, const float &size_percentage=1.f)

4.41.1 Detailed Description

This class is describes a portion of a training set, obtained by sampling values in the original set. Mainly used by stochastic gradient computation strategies.

4.41.2 Constructor & Destructor Documentation

4.41.2.1 subset()

```
Segugio::Training_set::subset::subset (
            Training_set * set,
            const float & size_percentage = 1.f )
```

Parameters

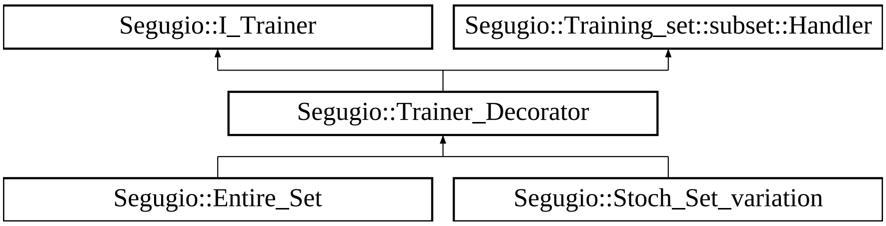
in	*set*	the training set from which this subset must be extracted
in	*size_percentage*	percentage to use for the extraction

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

- C:/Users/andre/Desktop/CRF/CRF/Header/Training_set.h
- C:/Users/andre/Desktop/CRF/CRF/Source/Training_set.cpp

4.42 Segugio::Trainer_Decorator Class Reference

Inheritance diagram for Segugio::Trainer_Decorator:


Segugio::I_Trainer
Segugio::Training_set::subset::Handler
Segugio::Trainer_Decorator
Segugio::Entire_Set
Segugio::Stoch_Set_variation

Public Member Functions

- **Trainer_Decorator** (Advancer_Concrete ∗to_wrap)
- void **Clean_Up** ()

Protected Member Functions

- void **__check_tunable_are_present** (Graph_Learnable ∗model_to_train)

Protected Attributes

- Advancer_Concrete ∗ **Wrapped**

Additional Inherited Members

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

- C:/Users/andre/Desktop/CRF/CRF/Source/Trainer.cpp

4.43 Segugio::Training_set Class Reference

This class is used for describing a training set for a graph.

```
#include <Training_set.h>
```

Classes

- class Basic_Extractor

  Basic extractor, see Training_set(const std::list<std::string>& variable_names, std::list<Array> samples, I_↩ Extractor<Array>∗ extractor)
- class I_Extractor

  This class is adopted for parsing a set of samples to import as a novel training set. You have to derive yout custom extractor, implementing the two vritual method.
- struct subset

  This class is describes a portion of a training set, obtained by sampling values in the original set. Mainly used by stochastic gradient computation strategies.

Public Member Functions

- Training_set (const std::string &file_to_import)
- template<typename Array >
  Training_set (const std::list< std::string > &variable_names, std::list< Array > &samples, I_Extractor< Array > ∗extractor)

  Similar to Training_set(const std::string& file_to_import),.
- template<typename Array >
  Training_set (const std::list< Categoric_var ∗ > &variable_in_the_net, std::list< Array > &samples, I_Extractor< Array > ∗extractor)

  Same as Training_set(const std::list<std::string>& variable_names, std::list<Array> samples, I_Extractor<Array>∗ extractor) passing the variables involved instead of the names.
- void Print (const std::string &file_name)

  This training set is reprinted in the location specified.

4.43.1 Detailed Description

This class is used for describing a training set for a graph.

A set is described in a textual file, where the first row must contain the list of names of the variables (all the variables) constituting a graph. All other rows are a single sample of the set, reporting the values assumed by the variables, with the order described by the first row

4.43.2 Constructor & Destructor Documentation

4.43.2.1 Training_set() [1/2]

```
Segugio::Training_set::Training_set (
            const std::string & file_to_import )
```

Parameters

in	*file_to_import*	file containing the set to import

4.43.2.2 Training_set() [2/2]

```
template<typename Array >
Segugio::Training_set::Training_set (
             const std::list< std::string > & variable_names,
             std::list< Array > & samples,
             I_Extractor< Array > * extractor )  [inline]
```

Similar to Training_set(const std::string& file_to_import),.

with the difference that the training set is not red from a textual file but it is imported from a list of container (generic can be list, vector or other) describing the samples of the set. You have to derived your own extractor for managing your particular container. Basic_Extractor is a baseline extractor that can be used for all those type having the method size() and the operator[].

Parameters

in	*variable_names*	the ordered list of variables to assume for the samples
in	*samples*	the list of generic Array representing the samples of the training set
in	*extractor*	the particular extractor to use, see I_Extractor

4.43.3 Member Function Documentation

4.43.3.1 Print()

```
void Segugio::Training_set::Print (
             const std::string & file_name )
```

This training set is reprinted in the location specified.

Parameters

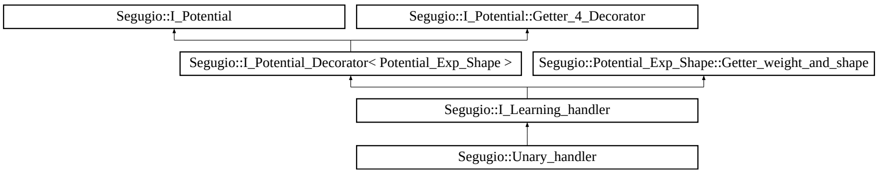
in	*file_name*	is the path of the file where the set must be printed

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

- C:/Users/andre/Desktop/CRF/CRF/Header/Training_set.h
- C:/Users/andre/Desktop/CRF/CRF/Source/Training_set.cpp

4.44 Segugio::Unary_handler Class Reference

Inheritance diagram for Segugio::Unary_handler:


Segugio::I_Potential
Segugio::I_Potential::Getter_4_Decorator
Segugio::I_Potential_Decorator< Potential_Exp_Shape >
Segugio::Potential_Exp_Shape::Getter_weight_and_shape
Segugio::I_Learning_handler
Segugio::Unary_handler

Public Member Functions

- **Unary_handler** (Node ∗N, Potential_Exp_Shape ∗pot_to_handle)

Additional Inherited Members

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

- C:/Users/andre/Desktop/CRF/CRF/Source/Graphical_model.cpp

4.45 Segugio::Graph_Learnable::Weights_Manager Struct Reference

Static Public Member Functions

- static void Get_tunable_w (std::list< float > ∗w, Graph_Learnable ∗model)
  Returns the values of the tunable weights, those that can vary when learning the model.

Friends

- class **I_Trainer**

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

- C:/Users/andre/Desktop/CRF/CRF/Header/Graphical_model.h
- C:/Users/andre/Desktop/CRF/CRF/Source/Graphical_model.cpp

Index