

1 Bayesian Networks: Code Examples

- Data Streams
- Random Variables
- Random Variables
- Models
 - Creating BNs
 - Creating Bayesian networks with latent variables
 - Modifying BNs
- Input/Output
 - I/O of data streams
 - I/O of BNs
- Input/Output
 - The inference engine
 - Variational Message Passing
 - Importance Sampling
- Learning Algorithms
 - Maximum Likelihood
 - Parallel Maximum Likelihood
 - Streaming Variational Bayes
 - Parallel Streaming Variational Bayes
- Concept Drift Method
 - Naive Bayes with Virtual Concept Drift Method
- HuginLink
 - Models conversion between AMiDST and Hugin
 - I/O of Bayesian Networks with Hugin net format
 - Invoking Hugin's inference engine
 - Invoking Hugin's Parallel TAN
- MoaLink
 - AMIDST Classifiers from MOA

1.1 Data Streams

In this example we show how to use the main features of a `DataStream` object. More precisely, we show six different ways of iterating over the data samples of a `DataStream` object.

```
1 package eu.amidst.core.examples.datastream;
2
3
4 import eu.amidst.core.datastream.Attribute;
5 import eu.amidst.core.datastream.DataInstance;
6 import eu.amidst.core.datastream.DataOnMemory;
7 import eu.amidst.core.datastream.DataStream;
8 import eu.amidst.core.utils.DataSetGenerator;
9
10 /**
11  * An example showing how to use the main features of a DataStream
12  * object. More precisely, we show six different
13  * ways of iterating over the data samples of a DataStream object.
14  */
15 public class DataStreamsExample {
16
17     public static void main(String[] args) throws Exception {
18
19         //We can open the data stream using the static class
20         //DataStreamLoader
21         DataStream<DataInstance> data = DataStreamLoader.open("
22             datasetsTests/data.arff");
23
24         //Generate the data stream using the class DataSetGenerator
25         DataStream<DataInstance> data = DataSetGenerator.generate
26             (1,1000,5,5);
27
28         //Access to the attributes defining the data set
29         System.out.println("Attributes defining the data set");
30         for (Attribute attribute : data.getAttributes()) {
31             System.out.println(attribute.getName());
32         }
33         Attribute discreteVar0 = data.getAttributes().
34             getAttributeByName("DiscreteVar0");
35
36         //1. Iterating over samples using a for loop
37         System.out.println("1. Iterating over samples using a for loop");
38         for (DataInstance dataInstance : data) {
39             System.out.println("The value of attribute A for the current
40                 data instance is: " + dataInstance.getValue(discreteVar0
41                     ));
42         }
43     }
44 }
```

```

39      //2. Iterating using streams. We need to restart the data again as
        a DataStream can only be used once.
40      System.out.println("2. Iterating using streams.");
41      data.restart();
42      data.stream().forEach(dataInstance ->
43          System.out.println("The value of attribute A for
            the current data instance is: " +
            dataInstance.getValue(discreteVar0))
44      );
45
46
47      //3. Iterating using parallel streams.
48      System.out.println("3. Iterating using parallel streams.");
49      data.restart();
50      data.parallelStream(10).forEach(dataInstance ->
51          System.out.println("The value of attribute A for
            the current data instance is: " +
            dataInstance.getValue(discreteVar0))
52      );
53
54      //4. Iterating over a stream of data batches.
55      System.out.println("4. Iterating over a stream of data batches.");
56      data.restart();
57      data.streamOfBatches(10).forEach(batch -> {
58          for (DataInstance dataInstance : batch)
59              System.out.println("The value of attribute A for the
                current data instance is: " + dataInstance.getValue(
                discreteVar0));
60      });
61
62      //5. Iterating over a parallel stream of data batches.
63      System.out.println("5. Iterating over a parallel stream of data
        batches.");
64      data.restart();
65      data.parallelStreamOfBatches(10).forEach(batch -> {
66          for (DataInstance dataInstance : batch)
67              System.out.println("The value of attribute A for the
                current data instance is: " + dataInstance.getValue(
                discreteVar0));
68      });
69
70
71      //6. Iterating over data batches using a for loop
72      System.out.println("6. Iterating over data batches using a for
        loop.");
73      for (DataOnMemory<DataInstance> batch : data.
        iterableOverBatches(10)) {
74          for (DataInstance dataInstance : batch)
75              System.out.println("The value of attribute A for the
                current data instance is: " + dataInstance.getValue(

```

```

76         discreteVar0));
77     }
78 }
79 }

```

1.2 Random Variables

This example show the basic functionality of the classes Variables and Variable.

```

1  package eu.amidst.core.examples.variables;
2
3
4  import eu.amidst.core.variables.Variable;
5  import eu.amidst.core.variables.Variables;
6  import eu.amidst.core.variables.stateSpaceTypes.FiniteStateSpace;
7
8  import java.util.Arrays;
9
10 /**
11  *
12  * This example show the basic functionality of the classes Variables and
13  * Variable.
14  *
15  * Created by andresmasegosa on 18/6/15.
16  */
17 public class VariablesExample {
18
19     public static void main(String[] args) throws Exception {
20
21         //We first create an empty Variables object
22         Variables variables = new Variables();
23
24         //We invoke the "new" methods of the object Variables to create
25         new variables.
26         //Now we create a Gaussian variables
27         Variable gaussianVar = variables.newGaussianVariable("Gaussian")
28         ;
29
30         //Now we create a Multinomial variable with two states
31         Variable multinomialVar = variables.newMultinomialVariable("
32         Multinomial", 2);
33
34         //Now we create a Multinomial variable with two states: TRUE
35         and FALSE
36         Variable multinomialVar2 = variables.newMultinomialVariable("
37         Multinomial2", Arrays.asList("TRUE,_FALSE"));
38
39     }
40 }

```

```

34      //For Multinomial variables we can iterate over their different
        states
35      FiniteStateSpace states = multinomialVar2.getStateSpaceType();
36      states.getStatesNames().forEach(System.out::println);
37
38      //Variable objects can also be used, for example, to know if one
        variable can be set as parent of some other variable
39      System.out.println("Can a Gaussian variable be parent of a
        Multinomial variable?" +
40          (multinomialVar.getDistributionType().
            isParentCompatible(gaussianVar)));
41
42      System.out.println("Can a Multinomial variable be parent of a
        Gaussian variable?" +
43          (gaussianVar.getDistributionType().isParentCompatible(
            multinomialVar)));
44
45    }
46 }

```

[\[Back to Top\]](#)

1.3 Models

1.3.1 Creating BNs

In this example, we take a data set, create a BN and we compute the log-likelihood of all the samples of this data set. The numbers defining the probability distributions of the BN are randomly fixed.

```

1  package eu.amidst.core.examples.models;
2
3  import eu.amidst.core.datastream.DataInstance;
4  import eu.amidst.core.datastream.DataStream;
5  import eu.amidst.core.io.BayesianNetworkWriter;
6  import eu.amidst.core.io.DataStreamLoader;
7  import eu.amidst.core.models.BayesianNetwork;
8  import eu.amidst.core.models.DAG;
9  import eu.amidst.core.variables.Variable;
10 import eu.amidst.core.variables.Variables;
11
12 /**
13  * In this example, we take a data set, create a BN and we compute the
        log-likelihood of all the samples
14  * of this data set. The numbers defining the probability distributions of
        the BN are randomly fixed.
15  * Created by andresmasegosa on 18/6/15.
16  */
17 public class CreatingBayesianNetworks {
18

```

```

19
20 public static void main(String[] args) throws Exception {
21
22     //We can open the data stream using the static class
23     DataStreamLoader
24     DataStream<DataInstance> data = DataStreamLoader.open("
25         datasets/simulated/syntheticData.arff");
26
27     /**
28      * 1. Once the data is loaded, we create a random variable for
29      * each of the attributes (i.e. data columns)
30      * in our data.
31      * 
32      * 2. {@link Variables} is the class for doing that. It takes a list of
33      * Attributes and internally creates
34      * all the variables. We create the variables using Variables class
35      * to guarantee that each variable
36      * has a different ID number and make it transparent for the user.
37      * 
38      * 3. We can extract the Variable objects by using the method
39      * getVariableByName();
40      */
41     Variables variables = new Variables(data.getAttributes());
42
43     Variable a = variables.getVariableByName("A");
44     Variable b = variables.getVariableByName("B");
45     Variable c = variables.getVariableByName("C");
46     Variable d = variables.getVariableByName("D");
47     Variable e = variables.getVariableByName("E");
48     Variable g = variables.getVariableByName("G");
49     Variable h = variables.getVariableByName("H");
50     Variable i = variables.getVariableByName("I");
51
52     /**
53      * 1. Once you have defined your {@link Variables} object, the
54      * next step is to create
55      * a DAG structure over this set of variables.
56      * 
57      * 2. To add parents to each variable, we first recover the
58      * ParentSet object by the method
59      * getParentSet(Variable var) and then call the method addParent
60      * ().
61      */
62     DAG dag = new DAG(variables);
63
64     dag.getParentSet(e).addParent(a);
65     dag.getParentSet(e).addParent(b);
66
67     dag.getParentSet(h).addParent(a);

```

```

60     dag.getParentSet(h).addParent(b);
61
62     dag.getParentSet(i).addParent(a);
63     dag.getParentSet(i).addParent(b);
64     dag.getParentSet(i).addParent(c);
65     dag.getParentSet(i).addParent(d);
66
67     dag.getParentSet(g).addParent(c);
68     dag.getParentSet(g).addParent(d);
69
70     /**
71      * 1. We first check if the graph contains cycles.
72      *
73      * 2. We print out the created DAG. We can check that
74         everything is as expected.
75     */
76     if (dag.containCycles()) {
77         try {
78             catch (Exception ex) {
79                 throw new IllegalArgumentException(ex);
80             }
81         }
82
83         System.out.println(dag.toString());
84
85     /**
86      * 1. We now create the Bayesian network from the previous DAG
87      *
88      * 2. The BN object is created from the DAG. It automatically
89         looks at the distribution type
90      * of each variable and their parents to initialize the Distributions
91         objects that are stored
92      * inside (i.e. Multinomial, Normal, CLG, etc). The parameters
93         defining these distributions are
94      * properly initialized.
95      *
96      * 3. The network is printed and we can have look at the kind of
97         distributions stored in the BN object.
98     */
99     BayesianNetwork bn = new BayesianNetwork(dag);
100    System.out.println(bn.toString());
101
102    /**
103     * 1. We iterate over the data set sample by sample.
104     *
105     * 2. For each sample or DataInstance object, we compute the log
106        of the probability that the BN object

```

```

103         * assigns to this observation.
104         *
105         * 3. We accumulate these log-probs and finally we print the log-
           prob of the data set.
106     */
107     double logProb = 0;
108     for (DataInstance instance : data) {
109         logProb += bn.getLogProbabilityOf(instance);
110     }
111     System.out.println(logProb);
112
113     BayesianNetworkWriter.save(bn, "networks/simulated/BNExample
           .bn");
114 }
115 }

```

[\[Back to Top\]](#)

1.3.2 Creating Bayesian networks with latent variables

In this example, we simply show how to create a BN model with hidden variables. We simply create a BN for clustering, i.e., a naive-Bayes like structure with a single common hidden variable acting as parent of all the observable variables.

```

1  package eu.amidst.core.examples.models;
2  import eu.amidst.core.datastream.DataInstance;
3  import eu.amidst.core.datastream.DataStream;
4  import eu.amidst.core.io.BayesianNetworkWriter;
5  import eu.amidst.core.io.DataStreamLoader;
6  import eu.amidst.core.models.BayesianNetwork;
7  import eu.amidst.core.models.DAG;
8  import eu.amidst.core.variables.Variable;
9  import eu.amidst.core.variables.Variables;
10
11  import java.util.Arrays;
12
13  /**
14   * In this example, we simply show how to create a BN model with latent
           variables. We simply
15   * create a BN for clustering, i.e., a naive-Bayes like structure with a
           single common latent or hidden variable
16   * acting as parent of all the observable variables.
17   *
18   * Created by andresmasegosa on 18/6/15.
19   */
20  public class CreatingBayesianNetworksWithLatentVariables {
21      public static void main(String[] args) throws Exception {
22

```



```

23      //We can open the data stream using the static class
24      DataStreamLoader
25      DataStream<DataInstance> data = DataStreamLoader.open("
26          datasets/simulated/syntheticData.arff");
27
28      /**
29       * 1. Once the data is loaded, we create a random variable for
30       * each of the attributes (i.e. data columns)
31       * in our data.
32       *
33       * 2. {@link Variables} is the class for doing that. It takes a list of
34       * Attributes and internally creates
35       * all the variables. We create the variables using Variables class
36       * to guarantee that each variable
37       * has a different ID number and make it transparent for the user.
38       *
39       * 3. We can extract the Variable objects by using the method
40       * getVariableByName();
41       */
42      Variables variables = new Variables(data.getAttributes());
43
44      Variable a = variables.getVariableByName("A");
45      Variable b = variables.getVariableByName("B");
46      Variable c = variables.getVariableByName("C");
47      Variable d = variables.getVariableByName("D");
48      Variable e = variables.getVariableByName("E");
49      Variable g = variables.getVariableByName("G");
50      Variable h = variables.getVariableByName("H");
51      Variable i = variables.getVariableByName("I");
52
53      /**
54       * 1. We create the hidden variable. For doing that we make use of
55       * the method "newMultinomialVariable". When
56       * a variable is created from an Attribute object, it contains all
57       * the information we need (e.g.
58       * the name, the type, etc). But hidden variables does not have an
59       * associated attribute
60       * and, for this reason, we use now this to provide this
61       * information.
62       *
63       * 2. Using the "newMultinomialVariable" method, we define a
64       * variable called HiddenVar, which is
65       * not associated to any attribute and, then, it is a latent variable,
66       * its state space is a finite set with two elements, and its
67       * distribution type is multinomial.
68       *
69       * 3. We finally create the hidden variable using the method "
70       * newVariable".
71       */

```

```

60     Variable hidden = variables.newMultinomialVariable("HiddenVar",
61         Arrays.asList("TRUE", "FALSE"));
62
63     /**
64      * 1. Once we have defined your {@link Variables} object,
65      *    including the latent variable,
66      * 2. the next step is to create a DAG structure over this set of
67      *    variables.
68      *
69      * 2. To add parents to each variable, we first recover the
70      *    ParentSet object by the method
71      *    getParentSet(Variable var) and then call the method addParent
72      *    (Variable var).
73      *
74      * 3. We just put the hidden variable as parent of all the other
75      *    variables. Following a naive-Bayes
76      *    like structure.
77      */
78     DAG dag = new DAG(variables);
79
80     dag.getParentSet(a).addParent(hidden);
81     dag.getParentSet(b).addParent(hidden);
82     dag.getParentSet(c).addParent(hidden);
83     dag.getParentSet(d).addParent(hidden);
84     dag.getParentSet(e).addParent(hidden);
85     dag.getParentSet(g).addParent(hidden);
86     dag.getParentSet(h).addParent(hidden);
87     dag.getParentSet(i).addParent(hidden);
88
89     /**
90      * We print the graph to see if is properly created.
91      */
92     System.out.println(dag.toString());
93
94     /**
95      * 1. We now create the Bayesian network from the previous DAG
96      *
97      * 2. The BN object is created from the DAG. It automatically
98      *    looks at the distribution type
99      *    of each variable and their parents to initialize the Distributions
100     objects that are stored
101     inside (i.e. Multinomial, Normal, CLG, etc). The parameters
102     defining these distributions are
103     properly initialized.
104     *
105     * 3. The network is printed and we can have look at the kind of
106     distributions stored in the BN object.
107     */
108     BayesianNetwork bn = new BayesianNetwork(dag);

```

```

99         System.out.println(bn.toString());
100
101         /**
102          * Finally the Bayesian network is saved to a file.
103          */
104         BayesianNetworkWriter.save(bn, "networks/simulated/
            BNHiddenExample.bn");
105
106     }
107 }

```

[\[Back to Top\]](#)

1.3.3 Modifying Bayesian networks

In this example we show how to access and modify the conditional probabilities of a Bayesian network model.

```

1 package eu.amidst.core.examples.models;
2 import eu.amidst.core.distribution.Multinomial;
3 import eu.amidst.core.distribution.Normal_MultinomialParents;
4 import eu.amidst.core.models.BayesianNetwork;
5 import eu.amidst.core.utils.BayesianNetworkGenerator;
6 import eu.amidst.core.variables.Variable;
7
8 /**
9  *
10 * In this example we show how to access and modify the conditional
    probabilities of a Bayesian network model.
11 * Created by andresmasegosa on 24/6/15.
12 */
13 public class ModifyingBayesianNetworks {
14
15     public static void main (String[] args){
16
17         //We first generate a Bayesian network with one multinomial, one
            Gaussian variable and one link
18         BayesianNetworkGenerator.setNumberOfGaussianVars(1);
19         BayesianNetworkGenerator.setNumberOfMultinomialVars(1,2);
20         BayesianNetworkGenerator.setNumberOfLinks(1);
21
22         BayesianNetwork bn = BayesianNetworkGenerator.
            generateBayesianNetwork();
23
24         //We print the randomly generated Bayesian networks
25         System.out.println(bn.toString());
26
27         //We first access the variable we are interested in
28         Variable multiVar = bn.getVariables().getVariableByName("
            DiscreteVar0");

```

```

29
30      //Using the above variable we can get the associated distribution
      and modify it
31      Multinomial multinomial = bn.getConditionalDistribution(
          multiVar);
32      multinomial.setProbabilities(new double[]{0.2, 0.8});
33
34      //Same than before but accessing the another variable
35      Variable normalVar = bn.getVariables().getVariableByName("
          GaussianVar0");
36
37      //In this case, the conditional distribtuion is of the type "Normal
      given Multinomial Parents"
38      Normal_MultinomialParents normalMultiDist = bn.
          getConditionalDistribution(normalVar);
39      normalMultiDist.getNormal(0).setMean(1.0);
40      normalMultiDist.getNormal(0).setVariance(1.0);
41
42      normalMultiDist.getNormal(1).setMean(0.0);
43      normalMultiDist.getNormal(1).setVariance(1.0);
44
45      //We print modified Bayesian network
46      System.out.println(bn.toString());
47  }
48 }

```

[Back to Top]

1.4 Input/Output

1.4.1 I/O of data streams

In this example we show how to load and save data sets from .arff files.

```

1  package eu.amidst.core.examples.io;
2
3
4  import eu.amidst.core.datastream.DataInstance;
5  import eu.amidst.core.datastream.DataStream;
6  import eu.amidst.core.io.DataStreamLoader;
7  import eu.amidst.core.io.DataStreamWriter;
8
9  /**
10   *
11   * In this example we show how to load and save data sets from ".arff"
      files (http://www.cs.waikato.ac.nz/ml/weka/arff.html)
12   *
13   * Created by andresmasegosa on 18/6/15.
14   */
15  public class DataStreamIOExample {

```

```

16
17     public static void main(String[] args) throws Exception {
18
19         //We can open the data stream using the static class
20         DataStreamLoader
21         DataStream<DataInstance> data = DataStreamLoader.open("
22             datasets/simulated/syntheticData.arff");
23
24         //We can save this data set to a new file using the static class
25         DataStreamWriter
26         DataStreamWriter.writeDataToFile(data, "datasets/simulated/tmp
27             .arff");
28     }

```

[Back to Top]

1.4.2 I/O of BNs

In this example we show how to load and save Bayesian networks models for a binary file with ".bn" extension. In this toolbox Bayesian networks models are saved as serialized objects.

```

1  package eu.amidst.core.examples.io;
2
3
4  import eu.amidst.core.io.BayesianNetworkLoader;
5  import eu.amidst.core.io.BayesianNetworkWriter;
6  import eu.amidst.core.models.BayesianNetwork;
7
8  import java.util.Random;
9
10 /**
11  *
12  * In this example we show how to load and save Bayesian networks
13  * models for a binary file with ".bn" extension. In
14  * this toolbox Bayesian networks models are saved as serialized objects.
15  *
16  * Created by andresmassegosa on 18/6/15.
17  */
18 public class BayesianNetworkIOExample {
19
20     public static void main(String[] args) throws Exception {
21
22         //We can load a Bayesian network using the static class
23         BayesianNetworkLoader

```

```

22         BayesianNetwork bn = BayesianNetworkLoader.loadFromFile("./
           networks/simulated/WasteIncinerator.bn");
23
24         //Now we print the loaded model
25         System.out.println(bn.toString());
26
27         //Now we change the parameters of the model
28         bn.randomInitialization(new Random(0));
29
30         //We can save this Bayesian network to using the static class
           BayesianNetworkWriter
31         BayesianNetworkWriter.save(bn, "networks/simulated/tmp.bn");
32
33     }
34 }

```

[Back to Top]

1.5 Inference

1.5.1 The inference engine

This example show how to perform inference in a Bayesian network model using the InferenceEngine static class. This class aims to be a straightfoward way to perform queries over a Bayesian network model. By the default the *VMP* inference method is invoked.

```

1  package eu.amidst.core.examples.inference;
2
3
4  import eu.amidst.core.inference.InferenceEngine;
5  import eu.amidst.core.io.BayesianNetworkLoader;
6  import eu.amidst.core.models.BayesianNetwork;
7  import eu.amidst.core.variables.Assignment;
8  import eu.amidst.core.variables.HashMapAssignment;
9  import eu.amidst.core.variables.Variable;
10
11  /**
12   * This example show how to perform inference in a Bayesian network
           model using the InferenceEngine static class.
13   * This class aims to be a straightfoward way to perform queries over a
           Bayesian network model.
14   *
15   * Created by andresmasegosa on 18/6/15.
16   */
17  public class InferenceEngineExample {
18
19      public static void main(String[] args) throws Exception {
20

```

```

21      //We first load the WasteIncinerator bayesian network which has
        multinomial and Gaussian variables.
22      BayesianNetwork bn = BayesianNetworkLoader.loadFromFile("./
        networks/simulated/WasteIncinerator.bn");
23
24      //We recover the relevant variables for this example: Mout which
        is normally distributed, and W which is multinomial.
25      Variable varMout = bn.getVariables().getVariableByName("Mout"
        );
26      Variable varW = bn.getVariables().getVariableByName("W");
27
28      //Set the evidence.
29      Assignment assignment = new HashMapAssignment(1);
30      assignment.setValue(varW,0);
31
32      //Then we query the posterior of
33      System.out.println("P(Mout|W=0) = " + InferenceEngine.
        getPosterior(varMout, bn, assignment));
34
35      //Or some more refined queries
36      System.out.println("P(0.7 < Mout < 6.59 | W=0) = " +
        InferenceEngine.getExpectedValue(varMout, bn, v -> (0.7 <
        v && v < 6.59) ? 1.0 : 0.0 ));
37
38    }
39
40 }

[Back to Top]

```

1.5.2 Variational Message Passing

This example we show how to perform inference on a general Bayesian network using the Variational Message Passing (VMP) algorithm detailed in

Winn, J. M., Bishop, C. M. (2005). Variational message passing.
In Journal of Machine Learning Research (pp. 661-694).

```

1  package eu.amidst.core.examples.inference;
2
3  import eu.amidst.core.inference.InferenceAlgorithm;
4  import eu.amidst.core.inference.messagepassing.VMP;
5  import eu.amidst.core.io.BayesianNetworkLoader;
6  import eu.amidst.core.models.BayesianNetwork;
7  import eu.amidst.core.variables.Assignment;
8  import eu.amidst.core.variables.HashMapAssignment;
9  import eu.amidst.core.variables.Variable;
10
11 /**
12  *

```

```

13  * This example we show how to perform inference on a general Bayesian
    * network using the Variational Message Passing (VMP)
14  * algorithm detailed in
15  *
16  * <i> Winn, J. M., and Bishop, C. M. (2005). Variational message
    * passing. In Journal of Machine Learning Research (pp. 661–694). </i>
    >
17  *
18  * Created by andresmassego on 18/6/15.
19  */
20  public class VMPExample {
21
22      public static void main(String[] args) throws Exception {
23
24          //We first load the WasteIncinerator bayesian network which has
            multinomial and Gaussian variables.
25          BayesianNetwork bn = BayesianNetworkLoader.loadFromFile("./
            networks/simulated/WasteIncinerator.bn");
26
27          //We recover the relevant variables for this example: Mout which
            is normally distributed, and W which is multinomial.
28          Variable varMout = bn.getVariables().getVariableByName("Mout"
            );
29          Variable varW = bn.getVariables().getVariableByName("W");
30
31          //First we create an instance of a inference algorithm. In this case
            , we use the VMP class.
32          InferenceAlgorithm inferenceAlgorithm = new VMP();
33          //Then, we set the BN model
34          inferenceAlgorithm.setModel(bn);
35
36          //If exists, we also set the evidence.
37          Assignment assignment = new HashMapAssignment(1);
38          assignment.setValue(varW,0);
39          inferenceAlgorithm.setEvidence(assignment);
40
41          //Then we run inference
42          inferenceAlgorithm.runInference();
43
44          //Then we query the posterior of
45          System.out.println("P(Mout|W=0)="_ + inferenceAlgorithm.
            getPosterior(varMout));
46
47          //Or some more refined queries
48          System.out.println("P(0.7<Mout<6.59_|_W=0)="_ +
            inferenceAlgorithm.getExpectedValue(varMout, v -> (0.7 < v
                && v < 6.59) ? 1.0 : 0.0 ));
49
50          //We can also compute the probability of the evidence
51          System.out.println("P(W=0)="_ + Math.exp(inferenceAlgorithm.

```



```

52         getLogProbabilityOfEvidence());
53
54     }
55 }

```

[Back to Top]

1.5.3 Importance Sampling

This example we show how to perform inference on a general Bayesian network using an importance sampling algorithm detailed in

Fung, R., Chang, K. C. (2013). Weighing and integrating evidence for stochastic simulation in Bayesian networks. arXiv preprint arXiv:1304.1504.

```

1  package eu.amidst.core.examples.inference;
2
3
4  import eu.amidst.core.inference.ImportanceSampling;
5  import eu.amidst.core.io.BayesianNetworkLoader;
6  import eu.amidst.core.models.BayesianNetwork;
7  import eu.amidst.core.variables.Assignment;
8  import eu.amidst.core.variables.HashMapAssignment;
9  import eu.amidst.core.variables.Variable;
10
11 /**
12  *
13  * This example we show how to perform inference on a general Bayesian
14  *   network using an importance sampling
15  *
16  * <i> Fung, R., and Chang, K. C. (2013). Weighing and integrating
17  *   evidence for
18  * stochastic simulation in Bayesian networks. arXiv preprint arXiv
19  * :1304.1504.
20  * </i>
21  *
22  * Created by andresmasegosa on 18/6/15.
23  */
24 public class ImportanceSamplingExample {
25
26     public static void main(String[] args) throws Exception {
27
28         //We first load the WasteIncinerator bayesian network which has
29         //multinomial and Gaussian variables.
30         BayesianNetwork bn = BayesianNetworkLoader.loadFromFile("./
31         networks/simulated/WasteIncinerator.bn");
32
33     }
34 }

```

```

29      //We recover the relevant variables for this example: Mout which
30      is normally distributed, and W which is multinomial.
31      Variable varMout = bn.getVariables().getVariableByName("Mout"
32      );
33      Variable varW = bn.getVariables().getVariableByName("W");
34
35      //First we create an instance of a inference algorithm. In this case
36      , we use the ImportanceSampling class.
37      ImportanceSampling inferenceAlgorithm = new
38      ImportanceSampling();
39      //Then, we set the BN model
40      inferenceAlgorithm.setModel(bn);
41
42      System.out.println(bn.toString());
43
44      //If it exists, we also set the evidence.
45      Assignment assignment = new HashMapAssignment(1);
46      assignment.setValue(varW,0);
47      inferenceAlgorithm.setEvidence(assignment);
48
49      //We can also set to be run in parallel on multicore CPUs
50      inferenceAlgorithm.setParallelMode(true);
51
52      //To perform more than one operation, data should be keep in
53      memory
54      inferenceAlgorithm.setKeepDataOnMemory(true);
55
56      //Then we run inference
57      inferenceAlgorithm.runInference();
58
59      //Then we query the posterior of
60      System.out.println("P(Mout|W=0) = " + inferenceAlgorithm.
61      getPosterior(varMout));
62
63      //Or some more refined queries
64      System.out.println("P(0.7<Mout<6.59|W=0) = " +
65      inferenceAlgorithm.getExpectedValue(varMout, v -> (0.7 < v
66      && v < 6.59) ? 1.0 : 0.0 ));
67
68      //We can also compute the probability of the evidence
69      System.out.println("P(W=0) = " + Math.exp(inferenceAlgorithm.
70      getLogProbabilityOfEvidence()));
71
72  }
73 }

```

[Back to Top]

1.6 Learning Algorithms

1.6.1 Maximum Likelihood

This other example shows how to learn incrementally the parameters of a Bayesian network using data batches,

```
1 package eu.amidst.core.examples.learning;
2
3
4
5 import eu.amidst.core.datastream.DataInstance;
6 import eu.amidst.core.datastream.DataOnMemory;
7 import eu.amidst.core.datastream.DataStream;
8 import eu.amidst.core.io.DataStreamLoader;
9 import eu.amidst.core.learning.parametric.ParallelMaximumLikelihood;
10 import eu.amidst.core.learning.parametric.ParameterLearningAlgorithm;
11 import eu.amidst.core.models.BayesianNetwork;
12 import eu.amidst.core.models.DAG;
13 import eu.amidst.core.variables.Variable;
14 import eu.amidst.core.variables.Variables;
15
16 /**
17  *
18  * This other example shows how to learn incrementally the parameters of
19  * a Bayesian network using data batches
20  *
21  * Created by andresmasegosa on 18/6/15.
22  */
23 public class MaximimumLikelihoodByBatchExample {
24
25     /**
26      * This method returns a DAG object with naive Bayes structure for
27      * the attributes of the passed data stream.
28      * @param dataStream object of the class DataStream<DataInstance>
29      * @param classIndex integer value indicating the position of the class
30      * @return object of the class DAG
31      */
32     public static DAG getNaiveBayesStructure(DataStream<DataInstance>
33         > dataStream, int classIndex){
34
35         //We create a Variables object from the attributes of the data
36         //stream
37         Variables modelHeader = new Variables(dataStream.getAttributes
38             ());
39
40         //We define the predictive class variable
41         Variable classVar = modelHeader.getVariableById(classIndex);
42
43         //Then, we create a DAG object with the defined model header
```

```

40     DAG dag = new DAG(modelHeader);
41
42     //We set the linkds of the DAG.
43     dag.getParentSets().stream().filter(w -> w.getMainVar() !=
44         classVar).forEach(w -> w.addParent(classVar));
45
46     return dag;
47 }
48
49 public static void main(String[] args) throws Exception {
50
51     //We can open the data stream using the static class
52     DataStreamLoader
53     DataStream<DataInstance> data = DataStreamLoader.open("
54         datasets/simulated/WasteIncineratorSample.arff");
55
56     //We create a ParameterLearningAlgorithm object with the
57     MaximumLikelihood builder
58     ParameterLearningAlgorithm parameterLearningAlgorithm = new
59         ParallelMaximumLikelihood();
60
61     //We fix the DAG structure
62     parameterLearningAlgorithm.setDAG(getNaiveBayesStructure(
63         data,0));
64
65     //We should invoke this method before processing any data
66     parameterLearningAlgorithm.initLearning();
67
68
69     //Then we show how we can perform parameter learnig by a
70     sequential updating of data batches.
71     for (DataOnMemory<DataInstance> batch : data.
72         iterableOverBatches(100)){
73         parameterLearningAlgorithm.updateModel(batch);
74     }
75
76     //And we get the model
77     BayesianNetwork bnModel = parameterLearningAlgorithm.
78         getLearntBayesianNetwork();
79
80     //We print the model
81     System.out.println(bnModel.toString());
82 }

```

[Back to Top]

1.6.2 Parallel Maximum Likelihood

This example shows how to learn in parallel the parameters of a Bayesian network from a stream of data using maximum likelihood.

```
1 package eu.amidst.core.examples.learning;
2
3
4 import eu.amidst.core.datastream.DataInstance;
5 import eu.amidst.core.datastream.DataStream;
6 import eu.amidst.core.io.DataStreamLoader;
7 import eu.amidst.core.learning.parametric.ParallelMaximumLikelihood;
8 import eu.amidst.core.models.BayesianNetwork;
9
10 /**
11  *
12  * This example shows how to learn in parallel the parameters of a
13  * Bayesian network from a stream of data using maximum
14  * likelihood.
15  *
16  * Created by andresmasegosa on 18/6/15.
17  */
18 public class ParallelMaximumLikelihoodExample {
19
20     public static void main(String[] args) throws Exception {
21
22         //We can open the data stream using the static class
23         DataStreamLoader
24         DataStream<DataInstance> data = DataStreamLoader.open("
25             datasets/simulated/WasteIncineratorSample.arff");
26
27         //We create a ParallelMaximumLikelihood object with the
28         MaximumLikelihood builder
29         ParallelMaximumLikelihood parameterLearningAlgorithm = new
30             ParallelMaximumLikelihood();
31
32         //We activate the parallel mode.
33         parameterLearningAlgorithm.setParallelMode(true);
34
35         //We fix the DAG structure
36         parameterLearningAlgorithm.setDAG(
37             MaximumLikelihoodByBatchExample.
38             getNaiveBayesStructure(data, 0));
39
40         //We set the batch size which will be employed to learn the model
41         in parallel
42         parameterLearningAlgorithm.setWindowsSize(100);
43
44         //We set the data which is going to be used for leaning the
45         parameters
```

```

38         parameterLearningAlgorithm.setDataStream(data);
39
40         //We perform the learning
41         parameterLearningAlgorithm.runLearning();
42
43         //And we get the model
44         BayesianNetwork bnModel = parameterLearningAlgorithm.
            getLearntBayesianNetwork();
45
46         //We print the model
47         System.out.println(bnModel.toString());
48
49     }
50
51 }

```

[Back to Top]

1.6.3 Streaming Variational Bayes

This example shows how to learn incrementally the parameters of a Bayesian network from a stream of data with a Bayesian approach using the following algorithm,

Broderick, T., Boyd, N., Wibisono, A., Wilson, A. C., and Jordan, M. I. (2013). Streaming variational Bayes. In Advances in Neural Information Processing Systems (pp. 1727-1735).

In this second example we show a alternative implementation which explicitly updates the model by batches by using the class SVB.

```

1
2 package eu.amidst.core.examples.learning;
3
4
5
6
7 import eu.amidst.core.datastream.DataInstance;
8 import eu.amidst.core.datastream.DataOnMemory;
9 import eu.amidst.core.datastream.DataStream;
10 import eu.amidst.core.io.DataStreamLoader;
11 import eu.amidst.core.learning.parametric.bayesian.SVB;
12 import eu.amidst.core.models.BayesianNetwork;
13 import eu.amidst.core.utils.DAGGenerator;
14
15 /**
16  *
17  * This example shows how to learn incrementally the parameters of a
18  * Bayesian network from a stream of data with a Bayesian
19  * approach using the following algorithm
20  *

```

```

20 * <i> Broderick, T., Boyd, N., Wibisono, A., Wilson, A. C., and Jordan,
    M. I. (2013). Streaming variational bayes.
21 * In Advances in Neural Information Processing Systems (pp. 1727–1735)
    . </i>
22 *
23 *
24 * Created by andresmasegosa on 18/6/15.
25 */
26 public class SVBByBatchExample {
27
28
29     public static void main(String[] args) throws Exception {
30
31         //We can open the data stream using the static class
32         DataStreamLoader
33         DataStream<DataInstance> data = DataStreamLoader.open("
34             datasets/simulated/WasteIncineratorSample.arff");
35
36         //We create a SVB object
37         SVB parameterLearningAlgorithm = new SVB();
38
39         //We fix the DAG structure
40         parameterLearningAlgorithm.setDAG(DAGGenerator.
41             getHiddenNaiveBayesStructure(data.getAttributes(),"H",2));
42
43         //We fix the size of the window, which must be equal to the size
44         //of the data batches we use for learning
45         parameterLearningAlgorithm.setWindowsSize(5);
46
47         //We can activate the output
48         parameterLearningAlgorithm.setOutput(true);
49
50         //We should invoke this method before processing any data
51         parameterLearningAlgorithm.initLearning();
52
53         //Then we show how we can perform parameter learning by a
54         //sequential updating of data batches.
55         for (DataOnMemory<DataInstance> batch : data.
56             iterableOverBatches(5)){
57             double log_likelihood_of_batch = parameterLearningAlgorithm.
58                 updateModel(batch);
59             System.out.println("Log-Likelihood_of_Batch:_" +
60                 log_likelihood_of_batch);
61         }
62
63         //And we get the model
64         BayesianNetwork bnModel = parameterLearningAlgorithm.
65             getLearntBayesianNetwork();
66     }
67 }

```

```

59         //We print the model
60         System.out.println(bnModel.toString());
61
62     }
63
64 }

```

[Back to Top]

1.6.4 Parallel Streaming Variational Bayes

This example shows how to learn in the parameters of a Bayesian network from a stream of data with a Bayesian approach using the parallel version of the SVB algorithm,

Broderick, T., Boyd, N., Wibisono, A., Wilson, A. C., and Jordan, M. I. (2013). Streaming variational Bayes. In Advances in Neural Information Processing Systems (pp. 1727-1735).

```

1
2 package eu.amidst.core.examples.learning;
3
4
5 import eu.amidst.core.datastream.DataInstance;
6 import eu.amidst.core.datastream.DataStream;
7 import eu.amidst.core.io.DataStreamLoader;
8 import eu.amidst.core.learning.parametric.bayesian.ParallelSVB;
9 import eu.amidst.core.models.BayesianNetwork;
10 import eu.amidst.core.utils.DAGGenerator;
11
12 /**
13  *
14  * This example shows how to learn the parameters of a Bayesian network
15  * from a stream of data with a Bayesian
16  * approach using a parallel version of the following algorithm
17  *
18  * Broderick, T., Boyd, N., Wibisono, A., Wilson, A. C., and Jordan,
19  * M. I. (2013). Streaming variational Bayes.
20  * In Advances in Neural Information Processing Systems (pp. 1727–1735)
21  * . </i>
22  *
23  * Created by andresmasegosa on 18/6/15.
24  */
25 public class ParallelSVBExample {
26
27     public static void main(String[] args) throws Exception {
28
29         //We can open the data stream using the static class
30         DataStreamLoader

```



```

28     DataStream<DataInstance> data = DataStreamLoader.open("
        datasets/simulated/WasteIncineratorSample.arff");
29
30     //We create a ParallelSVB object
31     ParallelSVB parameterLearningAlgorithm = new ParallelSVB();
32
33     //We fix the number of cores we want to exploit
34     parameterLearningAlgorithm.setNCores(4);
35
36     //We fix the DAG structure, which is a Naive Bayes with a global
        latent binary variable
37     parameterLearningAlgorithm.setDAG(DAGGenerator.
        getHiddenNaiveBayesStructure(data.getAttributes(), "H", 2));
38
39     //We fix the size of the window
40     parameterLearningAlgorithm.getSVBEngine().setWindowSize
        (100);
41
42     //We can activate the output
43     parameterLearningAlgorithm.setOutput(true);
44
45     //We set the data which is going to be used for leaning the
        parameters
46     parameterLearningAlgorithm.setDataStream(data);
47
48     //We perform the learning
49     parameterLearningAlgorithm.runLearning();
50
51     //And we get the model
52     BayesianNetwork bnModel = parameterLearningAlgorithm.
        getLearntBayesianNetwork();
53
54     //We print the model
55     System.out.println(bnModel.toString());
56
57 }
58
59 }

```

[\[Back to Top\]](#)

1.7 Concept Drift Methods

1.7.1 Naive Bayes with Virtual Concept Drift Detection

This example shows how to use the class `NaiveBayesVirtualConceptDriftDetector` to run the virtual concept drift detector detailed in

Borchani et al. Modeling concept drift: A probabilistic graphical model based approach. IDA 2015.

```

1  /*
2   *
3   *
4   * Licensed to the Apache Software Foundation (ASF) under one or more
      contributor license agreements.
5   * See the NOTICE file distributed with this work for additional
      information regarding copyright ownership.
6   * The ASF licenses this file to You under the Apache License, Version 2.0
      (the "License"); you may not use
7   * this file except in compliance with the License. You may obtain a copy
      of the License at
8   *
9   * http://www.apache.org/licenses/LICENSE-2.0
10  *
11  * Unless required by applicable law or agreed to in writing, software
      distributed under the License is
12  * distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
      CONDITIONS OF ANY KIND, either express or implied.
13  * See the License for the specific language governing permissions and
      limitations under the License.
14  *
15  *
16  */
17
18  package eu.amidst.core.examples.conceptdrift;
19
20
21  import eu.amidst.core.conceptdrift.
      NaiveBayesVirtualConceptDriftDetector;
22  import eu.amidst.core.datastream.DataInstance;
23  import eu.amidst.core.datastream.DataOnMemory;
24  import eu.amidst.core.datastream.DataStream;
25  import eu.amidst.core.io.DataStreamLoader;
26  import eu.amidst.core.variables.Variable;
27
28  /**
29   * This example shows how to use the class
      NaiveBayesVirtualConceptDriftDetector to run the virtual concept
      drift
30   * detector detailed in
31   *
32   * <i>Borchani et al. Modeling concept drift: A probabilistic graphical
      model based approach. IDA 2015.</i>
33   *
34   */
35  public class NaiveBayesVirtualConceptDriftDetectorExample {
36      public static void main(String[] args) {
37
38          //We can open the data stream using the static class
      DataStreamLoader

```

```

39      DataStream<DataInstance> data = DataStreamLoader.open("./
40         datasets/DriftSets/sea.arff");
41
42         //We create a NaiveBayesVirtualConceptDriftDetector object
43         NaiveBayesVirtualConceptDriftDetector virtualDriftDetector =
44             new NaiveBayesVirtualConceptDriftDetector();
45
46         //We set class variable as the last attribute
47         virtualDriftDetector.setClassIndex(-1);
48
49         //We set the data which is going to be used
50         virtualDriftDetector.setData(data);
51
52         //We fix the size of the window
53         int windowSize = 1000;
54         virtualDriftDetector.setWindowSize(windowSize);
55
56         //We fix the so-called transition variance
57         virtualDriftDetector.setTransitionVariance(0.1);
58
59         //We fix the number of global latent variables
60         virtualDriftDetector.setNumberOfGlobalVars(1);
61
62         //We should invoke this method before processing any data
63         virtualDriftDetector.initLearning();
64
65         //Some prints
66         System.out.print("Batch");
67         for (Variable hiddenVar : virtualDriftDetector.getHiddenVars()) {
68             System.out.print("\t" + hiddenVar.getName());
69         }
70         System.out.println();
71
72         //Then we show how we can perform the sequential processing of
73         // data batches. They must be of the same value than the window
74         // size parameter set above.
75         int countBatch = 0;
76         for (DataOnMemory<DataInstance> batch : data.
77             iterableOverBatches(windowSize)){
78
79             //We update the model by invoking this method. The output
80             // is an array with a value associated
81             // to each fo the global hidden variables
82             double[] out = virtualDriftDetector.updateModel(batch);
83
84             //We print the output
85             System.out.print(countBatch + "\t");
86             for (int i = 0; i < out.length; i++) {
87                 System.out.print(out[i]+"\t");

```

```

86         }
87         System.out.println();
88         countBatch++;
89     }
90 }
91 }

```

[\[Back to Top\]](#)

1.8 HuginLink

1.8.1 Models conversion between AMiDST and Hugin

This example shows how to use the class BNConverterToAMIDST and BNConverterToHugin to convert a Bayesian network models between Hugin and AMIDST formats

```

1  package eu.amidst.core.examples.huginlink;
2
3  import COM.hugin.HAPI.Domain;
4  import COM.hugin.HAPI.ExceptionHugin;
5  import eu.amidst.core.models.BayesianNetwork;
6  import eu.amidst.huginlink.converters.BNConverterToAMIDST;
7  import eu.amidst.huginlink.converters.BNConverterToHugin;
8  import eu.amidst.huginlink.io.BNLoaderFromHugin;
9
10 /**
11  * Created by rcabanais on 24/06/16.
12  */
13 public class HuginConversionExample {
14     public static void main(String[] args) throws ExceptionHugin {
15         //We load from Hugin format
16         Domain huginBN = BNLoaderFromHugin.loadFromFile("./
            networks/simulated/WasteIncinerator.bn");
17
18         //Then, it is converted to AMIDST BayesianNetwork object
19         BayesianNetwork amidstBN = BNConverterToAMIDST.
            convertToAmidst(huginBN);
20
21         //Then, it is converted to Hugin Bayesian Network object
22         huginBN = BNConverterToHugin.convertToHugin(amidstBN);
23
24         System.out.println(amidstBN.toString());
25         System.out.println(huginBN.toString());
26
27     }
28 }

```

[\[Back to Top\]](#)

1.8.2 I/O of Bayesian Networks with Hugin net format

This example shows how to use the class BNLoaderFromHugin and BNWriterToHugin classes to load and write Bayesian networks in Hugin format

```
1 package eu.amidst.core.examples.huginlink;
2
3 import COM.hugin.HAPI.Domain;
4 import COM.hugin.HAPI.ExceptionHugin;
5 import eu.amidst.core.models.BayesianNetwork;
6 import eu.amidst.huginlink.converters.BNConverterToAMIDST;
7 import eu.amidst.huginlink.io.BNLoaderFromHugin;
8 import eu.amidst.huginlink.io.BayesianNetworkWriterToHugin;
9
10 /**
11  * Created by rcabanas on 24/06/16.
12  */
13 public class HuginIOExample {
14     public static void main(String[] args) throws ExceptionHugin {
15         //We load from Hugin format
16         Domain huginBN = BNLoaderFromHugin.loadFromFile("networks
17             /asia.net");
18
19         //We save a AMIDST BN to Hugin format
20         BayesianNetwork amidstBN = BNConverterToAMIDST.
21             convertToAmidst(huginBN);
22         BayesianNetworkWriterToHugin.save(amidstBN,"networks/tmp.
23             net");
24     }
25 }
```

[\[Back to Top\]](#)

1.8.3 Invoking Hugin's inference engine

This example we show how to perform inference using Hugin inference engine within the AMiDST toolbox

```
1 package eu.amidst.core.examples.huginlink;
2
3 import eu.amidst.core.inference.InferenceAlgorithm;
4 import eu.amidst.core.io.BayesianNetworkLoader;
5 import eu.amidst.core.models.BayesianNetwork;
6 import eu.amidst.core.variables.Assignment;
7 import eu.amidst.core.variables.HashMapAssignment;
8 import eu.amidst.core.variables.Variable;
9 import eu.amidst.huginlink.inference.HuginInference;
10
11 import java.io.IOException;
12
```

```

13  /**
14   * Created by rcabanias on 24/06/16.
15   */
16  public class HuginInferenceExample {
17      public static void main(String[] args) throws IOException,
          ClassNotFoundException {
18          //We first load the WasteIncinerator bayesian network
19          //which has multinomial and Gaussian variables.
20          BayesianNetwork bn = BayesianNetworkLoader.loadFromFile("./
              networks/WasteIncinerator.bn");
21
22          //We recover the relevant variables for this example:
23          //Mout which is normally distributed, and W which is
              multinomial.
24          Variable varMout = bn.getVariables().getVariableByName("Mout"
              );
25          Variable varW = bn.getVariables().getVariableByName("W");
26
27          //First we create an instance of a inference algorithm.
28          //In this case, we use the ImportanceSampling class.
29          InferenceAlgorithm inferenceAlgorithm = new HuginInference();
30
31          //Then, we set the BN model
32          inferenceAlgorithm.setModel(bn);
33
34          //If exists, we also set the evidence.
35          Assignment assignment = new HashMapAssignment(1);
36          assignment.setValue(varW, 0);
37          inferenceAlgorithm.setEvidence(assignment);
38
39          //Then we run inference
40          inferenceAlgorithm.runInference();
41
42          //Then we query the posterior of
43          System.out.println("P(Mout|W=0) = " + inferenceAlgorithm.
              getPosterior(varMout));
44
45          //Or some more refined queries
46          System.out.println("P(0.7 < Mout < 3.5 | W=0) = "
47              + inferenceAlgorithm.getExpectedValue(varMout, v ->
                  (0.7 < v && v < 3.5) ? 1.0 : 0.0));
48
49      }
50  }

```

[Back to Top]

1.8.4 Invoking Hugin's Parallel TAN

This example we show how to perform inference using Hugin inference engine within the AMIDST toolbox.

This example shows how to use Hugin's functionality to learn in parallel a TAN model. An important remark is that Hugin only allows to learn the TAN model for a data set completely loaded into RAM memory. The case where our data set does not fit into memory, it solved in AMIDST in the following way. We learn the structure using a smaller data set produced by Reservoir sampling and, then, we use AMIDST's ParallelMaximumLikelihood to learn the parameters of the TAN model over the whole data set.

For further details about the implementation of the parallel TAN algorithm look at the following paper:

Madsen, A.L. et al. A New Method for Vertical Parallelisation of TAN Learning Based on Balanced Incomplete Block Designs. Probabilistic Graphical Models. Lecture Notes in Computer Science Volume 8754, 2014, pp 302-317.

```
1 package eu.amidst.core.examples.huginlink;
2
3 import eu.amidst.core.inference.InferenceAlgorithm;
4 import eu.amidst.core.io.BayesianNetworkLoader;
5 import eu.amidst.core.models.BayesianNetwork;
6 import eu.amidst.core.variables.Assignment;
7 import eu.amidst.core.variables.HashMapAssignment;
8 import eu.amidst.core.variables.Variable;
9 import eu.amidst.huginlink.inference.HuginInference;
10
11 import java.io.IOException;
12
13 /**
14  * Created by rcabanas on 24/06/16.
15  */
16 public class HuginInferenceExample {
17     public static void main(String[] args) throws IOException,
18         ClassNotFoundException {
19         //We first load the WasteIncinerator bayesian network
20         //which has multinomial and Gaussian variables.
21         BayesianNetwork bn = BayesianNetworkLoader.loadFromFile("./
22             networks/WasteIncinerator.bn");
23
24         //We recover the relevant variables for this example:
25         //Mout which is normally distributed, and W which is
26         //multinomial.
27         Variable varMout = bn.getVariables().getVariableByName("Mout"
28             );
29         Variable varW = bn.getVariables().getVariableByName("W");
```

```

27      //First we create an instance of a inference algorithm.
28      //In this case, we use the ImportanceSampling class.
29      InferenceAlgorithm inferenceAlgorithm = new HuginInference();
30
31      //Then, we set the BN model
32      inferenceAlgorithm.setModel(bn);
33
34      //If exists, we also set the evidence.
35      Assignment assignment = new HashMapAssignment(1);
36      assignment.setValue(varW, 0);
37      inferenceAlgorithm.setEvidence(assignment);
38
39      //Then we run inference
40      inferenceAlgorithm.runInference();
41
42      //Then we query the posterior of
43      System.out.println("P(Mout|W=0)_" + inferenceAlgorithm.
44                          getPosterior(varMout));
45
46      //Or some more refined queries
47      System.out.println("P(0.7<Mout<3.5_|W=0)_"
48                          + inferenceAlgorithm.getExpectedValue(varMout, v ->
49                          (0.7 < v && v < 3.5) ? 1.0 : 0.0));
50  }

```

[Back to Top]

1.9 MoaLink

1.9.1 AMIDST Classifiers from MOA

The following command can be used to learn a Bayesian model with a latent Gaussian variable (HG) and a multinomial with 2 states (HM), as displayed in figure below. The VMP algorithm is used to learn the parameters of these two non-observed variables and make predictions over the class variable.

```

1 java -Xmx512m -cp "../lib/*" -javaagent:../lib/sizeofag-1.0.0.jar
2 moa.DoTask EvaluatePrequential -l \ (bayes.AmidstClassifier -g 1
3 -m 2\ ) -s generators.RandomRBFGenerator -i 10000 -f 1000 -q 1000

```

[Back to Top]

1.9.2 AMIDST Classifiers from MOA

It is possible to learn an enriched naive Bayes model for regression if the class label is of a continuous nature. The following command uses the model in Figure 2 on a toy dataset from WEKA's collection of regression problems.

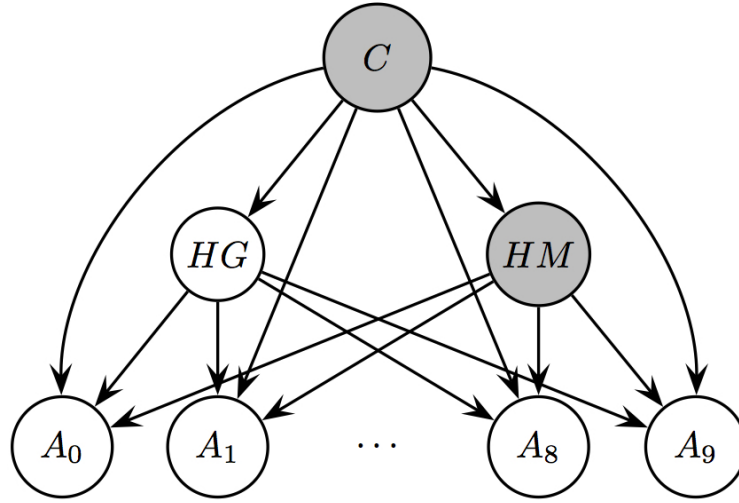


Figure 1: HODE example

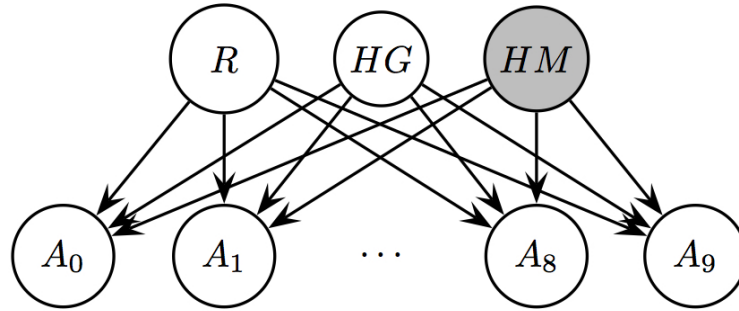


Figure 2: HODE regression example

```

1 java -Xmx512m -cp "../lib/*" -javaagent:../lib/sizeofag-1.0.0.jar
2 moa.DoTask EvaluatePrequentialRegression -l bayes.AmidstRegressor
3 -s (ArffFileStream -f ./quake.arff)

```

Note that the simpler the dataset the less complex the model should be. In this case, `quake.arff` is a very simple and small dataset that should probably be learn with a more simple classifier, that is, a high-bias-low-variance classifier, in order to avoid overfitting. This aims at providing a simple running example. [\[Back to Top\]](#)