# 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.

```java
package eu.amidst.core.examples.datastream;


import eu.amidst.core.datastream.Attribute;
import eu.amidst.core.datastream.DataInstance;
import eu.amidst.core.datastream.DataOnMemory;
import eu.amidst.core.datastream.DataStream;
import eu.amidst.core.utils.DataSetGenerator;

/**
 * An example showing 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.
 */
public class DataStreamsExample {

    public static void main(String[] args) throws Exception {

        //We can open the data stream using the static class DataStreamLoader
        //DataStream<DataInstance> data = DataStreamLoader.open("datasetsTests/data.arff");

        //Generate the data stream using the class DataSetGenerator
        DataStream<DataInstance> data = DataSetGenerator.generate(1,1000,5,5);


        //Access to the attributes defining the data set
        System.out.println("Attributes defining the data set");
        for (Attribute attribute : data.getAttributes()) {
            System.out.println(attribute.getName());
        }
        Attribute discreteVar0 = data.getAttributes().getAttributeByName("DiscreteVar0");

        //1. Iterating over samples using a for loop
        System.out.println("1. Iterating over samples using a for loop");
        for (DataInstance dataInstance : data) {
            System.out.println("The value of attribute A for the current data instance is: " + dataInstance.getValue(discreteVar0));
        }


```

```
        //2. Iterating using streams. We need to restart the data again as a DataStream can only be used once.
        System.out.println("2. Iterating using streams.");
        data.restart();
        data.stream().forEach(dataInstance ->
                System.out.println("The value of attribute A for the current data instance is: " + dataInstance.getValue(discreteVar0))
        );


        //3. Iterating using parallel streams.
        System.out.println("3. Iterating using parallel streams.");
        data.restart();
        data.parallelStream(10).forEach(dataInstance ->
                System.out.println("The value of attribute A for the current data instance is: " + dataInstance.getValue(discreteVar0))
        );

        //4. Iterating over a stream of data batches.
        System.out.println("4. Iterating over a stream of data batches.");
        data.restart();
        data.streamOfBatches(10).forEach(batch -> {
            for (DataInstance dataInstance : batch)
                System.out.println("The value of attribute A for the current data instance is: " + dataInstance.getValue(discreteVar0));
        });

        //5. Iterating over a parallel stream of data batches.
        System.out.println("5. Iterating over a parallel stream of data batches.");
        data.restart();
        data.parallelStreamOfBatches(10).forEach(batch -> {
            for (DataInstance dataInstance : batch)
                System.out.println("The value of attribute A for the current data instance is: " + dataInstance.getValue(discreteVar0));
        });


        //6. Iterating over data batches using a for loop
        System.out.println("6. Iterating over data batches using a for loop.");
        for (DataOnMemory<DataInstance> batch : data.iterableOverBatches(10)) {
            for (DataInstance dataInstance : batch)
                System.out.println("The value of attribute A for the current data instance is: " + dataInstance.getValue(
```

```java
                discreteVar0));
        }
    }

}
```

## 1.2 Random Variables

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

```java
package eu.amidst.core.examples.variables;


import eu.amidst.core.variables.Variable;
import eu.amidst.core.variables.Variables;
import eu.amidst.core.variables.stateSpaceTypes.FiniteStateSpace;

import java.util.Arrays;

/**
 *
 * This example show the basic functionality of the classes Variables and Variable.
 *
 *
 * Created by andresmasegosa on 18/6/15.
 */
public class VariablesExample {

    public static void main(String[] args) throws Exception {

        //We first create an empty Variables object
        Variables variables = new Variables();

        //We invoke the "new" methods of the object Variables to create new variables.
        //Now we create a Gaussian variables
        Variable gaussianVar = variables.newGaussianVariable("Gaussian");

        //Now we create a Multinomial variable with two states
        Variable multinomialVar = variables.newMultinomialVariable("Multinomial", 2);

        //Now we create a Multinomial variable with two states: TRUE and FALSE
        Variable multinomialVar2 = variables.newMultinomialVariable("Multinomial2", Arrays.asList("TRUE, FALSE"));

```

```
        //For Multinomial variables we can iterate over their different states
        FiniteStateSpace states = multinomialVar2.getStateSpaceType();
        states.getStatesNames().forEach(System.out::println);

        //Variable objects can also be used, for example, to know if one variable can be set as parent of some other variable
        System.out.println("Can a Gaussian variable be parent of Multinomial variable? " +
                (multinomialVar.getDistributionType().isParentCompatible(gaussianVar)));

        System.out.println("Can a Multinomial variable be parent of Gaussian variable? " +
                (gaussianVar.getDistributionType().isParentCompatible(multinomialVar)));

    }
}
```

[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.

```
package eu.amidst.core.examples.models;

import eu.amidst.core.datastream.DataInstance;
import eu.amidst.core.datastream.DataStream;
import eu.amidst.core.io.BayesianNetworkWriter;
import eu.amidst.core.io.DataStreamLoader;
import eu.amidst.core.models.BayesianNetwork;
import eu.amidst.core.models.DAG;
import eu.amidst.core.variables.Variable;
import eu.amidst.core.variables.Variables;

/**
 * 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.
 * Created by andresmasegosa on 18/6/15.
 */
public class CreatingBayesianNetworks {

```

```java

    public static void main(String[] args) throws Exception {

        //We can open the data stream using the static class DataStreamLoader
        DataStream<DataInstance> data = DataStreamLoader.open("datasets/simulated/syntheticData.arff");


        /**
         * 1. Once the data is loaded, we create a random variable for each of the attributes (i.e. data columns)
         * in our data.
         *
         * 2. {@link Variables} is the class for doing that. It takes a list of Attributes and internally creates
         * all the variables. We create the variables using Variables class to guarantee that each variable
         * has a different ID number and make it transparent for the user.
         *
         * 3. We can extract the Variable objects by using the method getVariableByName();
         */
        Variables variables = new Variables(data.getAttributes());

        Variable a = variables.getVariableByName("A");
        Variable b = variables.getVariableByName("B");
        Variable c = variables.getVariableByName("C");
        Variable d = variables.getVariableByName("D");
        Variable e = variables.getVariableByName("E");
        Variable g = variables.getVariableByName("G");
        Variable h = variables.getVariableByName("H");
        Variable i = variables.getVariableByName("I");

        /**
         * 1. Once you have defined your {@link Variables} object, the next step is to create
         * a DAG structure over this set of variables.
         *
         * 2. To add parents to each variable, we first recover the ParentSet object by the method
         * getParentSet(Variable var) and then call the method addParent().
         */
        DAG dag = new DAG(variables);

        dag.getParentSet(e).addParent(a);
        dag.getParentSet(e).addParent(b);

        dag.getParentSet(h).addParent(a);
```

```java
        dag.getParentSet(h).addParent(b);

        dag.getParentSet(i).addParent(a);
        dag.getParentSet(i).addParent(b);
        dag.getParentSet(i).addParent(c);
        dag.getParentSet(i).addParent(d);

        dag.getParentSet(g).addParent(c);
        dag.getParentSet(g).addParent(d);

        /**
         * 1. We first check if the graph contains cycles.
         *
         * 2. We print out the created DAG. We can check that everything is as expected.
         */
        if (dag.containCycles()) {
            try {
            } catch (Exception ex) {
                throw new IllegalArgumentException(ex);
            }
        }

        System.out.println(dag.toString());


        /**
         * 1. We now create the Bayesian network from the previous DAG.
         *
         * 2. The BN object is created from the DAG. It automatically looks at the distribution tye
         * of each variable and their parents to initialize the Distributions objects that are stored
         * inside (i.e. Multinomial, Normal, CLG, etc). The parameters defining these distributions are
         * properly initialized.
         *
         * 3. The network is printed and we can have look at the kind of distributions stored in the BN object.
         */
        BayesianNetwork bn = new BayesianNetwork(dag);
        System.out.println(bn.toString());


        /**
         * 1. We iterate over the data set sample by sample.
         *
         * 2. For each sample or DataInstance object, we compute the log of the probability that the BN object
```

```
         * assigns to this observation.
         *
         * 3. We accumulate these log-probs and finally we print the log-prob of the data set.
         */
        double logProb = 0;
        for (DataInstance instance : data) {
            logProb += bn.getLogProbabiltyOf(instance);
        }
        System.out.println(logProb);

        BayesianNetworkWriter.save(bn, "networks/simulated/BNExample.bn");
    }
}
```

[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 parant of all the observable variables.

```
package eu.amidst.core.examples.models;
import eu.amidst.core.datastream.DataInstance;
import eu.amidst.core.datastream.DataStream;
import eu.amidst.core.io.BayesianNetworkWriter;
import eu.amidst.core.io.DataStreamLoader;
import eu.amidst.core.models.BayesianNetwork;
import eu.amidst.core.models.DAG;
import eu.amidst.core.variables.Variable;
import eu.amidst.core.variables.Variables;

import java.util.Arrays;

/**
 * In this example, we simply show how to create a BN model with latent variables. We simply
 * create a BN for clustering, i.e., a naive-Bayes like structure with a single common latent or hidden variable
 * acting as parent of all the observable variables.
 *
 * Created by andresmasegosa on 18/6/15.
 */
public class CreatingBayesianNetworksWithLatentVariables {
    public static void main(String[] args) throws Exception {

```

```
        //We can open the data stream using the static class
            DataStreamLoader
        DataStream<DataInstance> data = DataStreamLoader.open("
            datasets/simulated/syntheticData.arff");

        /**
         * 1. Once the data is loaded, we create a random variable for
             each of the attributes (i.e. data columns)
         * in our data.
         *
         * 2. {@link Variables} is the class for doing that. It takes a list of
             Attributes and internally creates
         * all the variables. We create the variables using Variables class
             to guarantee that each variable
         * has a different ID number and make it transparent for the user.
         *
         * 3. We can extract the Variable objects by using the method
             getVariableByName();
         */
        Variables variables = new Variables(data.getAttributes());

        Variable a = variables.getVariableByName("A");
        Variable b = variables.getVariableByName("B");
        Variable c = variables.getVariableByName("C");
        Variable d = variables.getVariableByName("D");
        Variable e = variables.getVariableByName("E");
        Variable g = variables.getVariableByName("G");
        Variable h = variables.getVariableByName("H");
        Variable i = variables.getVariableByName("I");

        /**
         * 1. We create the hidden variable. For doing that we make use of
              the method "newMultinomialVariable". When
         * a variable is created from an Attribute object, it contains all
             the information we need (e.g.
         * the name, the type, etc). But hidden variables does not have an
              associated attribute
         * and, for this reason, we use now this to provide this
             information.
         *
         * 2. Using the "newMultinomialVariable" method, we define a
             variable called HiddenVar, which is
         * not associated to any attribute and, then, it is a latent variable,
              its state space is a finite set with two elements, and its
         * distribution type is multinomial.
         *
         * 3. We finally create the hidden variable using the method "
             newVariable".
         */

```

```java
        Variable hidden = variables.newMultinomialVariable("HiddenVar", Arrays.asList("TRUE", "FALSE"));

        /**
         * 1. Once we have defined your {@link Variables} object, including the latent variable,
         * the next step is to create a DAG structure over this set of variables.
         *
         * 2. To add parents to each variable, we first recover the ParentSet object by the method
         * getParentSet(Variable var) and then call the method addParent(Variable var).
         *
         * 3. We just put the hidden variable as parent of all the other variables. Following a naive-Bayes
         * like structure.
         */
        DAG dag = new DAG(variables);

        dag.getParentSet(a).addParent(hidden);
        dag.getParentSet(b).addParent(hidden);
        dag.getParentSet(c).addParent(hidden);
        dag.getParentSet(d).addParent(hidden);
        dag.getParentSet(e).addParent(hidden);
        dag.getParentSet(g).addParent(hidden);
        dag.getParentSet(h).addParent(hidden);
        dag.getParentSet(i).addParent(hidden);

        /**
         * We print the graph to see if is properly created.
         */
        System.out.println(dag.toString());

        /**
         * 1. We now create the Bayesian network from the previous DAG.
         *
         * 2. The BN object is created from the DAG. It automatically looks at the distribution type
         * of each variable and their parents to initialize the Distributions objects that are stored
         * inside (i.e. Multinomial, Normal, CLG, etc). The parameters defining these distributions are
         * properly initialized.
         *
         * 3. The network is printed and we can have look at the kind of distributions stored in the BN object.
         */
        BayesianNetwork bn = new BayesianNetwork(dag);
```

```
        System.out.println(bn.toString());

        /**
         * Finally the Bayesian network is saved to a file.
         */
        BayesianNetworkWriter.save(bn, "networks/simulated/
            BNHiddenExample.bn");

    }
}
```

[Back to Top]

### 1.3.3 Modifiying Bayesian networks

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

```
package eu.amidst.core.examples.models;
import eu.amidst.core.distribution.Multinomial;
import eu.amidst.core.distribution.Normal_MultinomialParents;
import eu.amidst.core.models.BayesianNetwork;
import eu.amidst.core.utils.BayesianNetworkGenerator;
import eu.amidst.core.variables.Variable;

/**
 *
 * In this example we show how to access and modify the conditional
     probabilities of a Bayesian network model.
 * Created by andresmasegosa on 24/6/15.
 */
public class ModifiyingBayesianNetworks {

    public static void main (String[] args){

        //We first generate a Bayesian network with one multinomial, one
             Gaussian variable and one link
        BayesianNetworkGenerator.setNumberOfGaussianVars(1);
        BayesianNetworkGenerator.setNumberOfMultinomialVars(1,2);
        BayesianNetworkGenerator.setNumberOfLinks(1);

        BayesianNetwork bn = BayesianNetworkGenerator.
            generateBayesianNetwork();

        //We print the randomly generated Bayesian networks
        System.out.println(bn.toString());

        //We first access the variable we are interested in
        Variable multiVar = bn.getVariables().getVariableByName("
            DiscreteVar0");
```

```java

        //Using the above variable we can get the associated distribution and modify it
        Multinomial multinomial = bn.getConditionalDistribution(multiVar);
        multinomial.setProbabilities(new double[]{0.2, 0.8});

        //Same than before but accessing the another variable
        Variable normalVar = bn.getVariables().getVariableByName("GaussianVar0");

        //In this case, the conditional distribtuion is of the type "Normal given Multinomial Parents"
        Normal_MultinomialParents normalMultiDist = bn.getConditionalDistribution(normalVar);
        normalMultiDist.getNormal(0).setMean(1.0);
        normalMultiDist.getNormal(0).setVariance(1.0);

        normalMultiDist.getNormal(1).setMean(0.0);
        normalMultiDist.getNormal(1).setVariance(1.0);

        //We print modified Bayesian network
        System.out.println(bn.toString());
    }
}
```

[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.

```java
package eu.amidst.core.examples.io;


import eu.amidst.core.datastream.DataInstance;
import eu.amidst.core.datastream.DataStream;
import eu.amidst.core.io.DataStreamLoader;
import eu.amidst.core.io.DataStreamWriter;

/**
 *
 * 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)
 *
 * Created by andresmasegosa on 18/6/15.
 */
public class DataStreamIOExample {
```

```java

    public static void main(String[] args) throws Exception {

        //We can open the data stream using the static class DataStreamLoader
        DataStream<DataInstance> data = DataStreamLoader.open("datasets/simulated/syntheticData.arff");

        //We can save this data set to a new file using the static class DataStreamWriter
        DataStreamWriter.writeDataToFile(data, "datasets/simulated/tmp.arff");



    }
}
```

[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.

```java
package eu.amidst.core.examples.io;


import eu.amidst.core.io.BayesianNetworkLoader;
import eu.amidst.core.io.BayesianNetworkWriter;
import eu.amidst.core.models.BayesianNetwork;

import java.util.Random;

/**
 *
 * 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.
 *
 * Created by andresmasegosa on 18/6/15.
 */
public class BayesianNetworkIOExample {

    public static void main(String[] args) throws Exception {

        //We can load a Bayesian network using the static class BayesianNetworkLoader
```

```
        BayesianNetwork bn = BayesianNetworkLoader.loadFromFile("./
            networks/simulated/WasteIncinerator.bn");

        //Now we print the loaded model
        System.out.println(bn.toString());

        //Now we change the parameters of the model
        bn.randomInitialization(new Random(0));

        //We can save this Bayesian network to using the static class
            BayesianNetworkWriter
        BayesianNetworkWriter.save(bn, "networks/simulated/tmp.bn");

    }
}
```

[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 straigthfoward way to perform queries over a Bayesian network model. By the default the *VMP* inference method is invoked.

```
package eu.amidst.core.examples.inference;


import eu.amidst.core.inference.InferenceEngine;
import eu.amidst.core.io.BayesianNetworkLoader;
import eu.amidst.core.models.BayesianNetwork;
import eu.amidst.core.variables.Assignment;
import eu.amidst.core.variables.HashMapAssignment;
import eu.amidst.core.variables.Variable;

/**
 * This example show how to perform inference in a Bayesian network
     model using the InferenceEngine static class.
 * This class aims to be a straigthfoward way to perform queries over a
     Bayesian network model.
 *
 * Created by andresmasegosa on 18/6/15.
 */
public class InferenceEngineExample {

    public static void main(String[] args) throws Exception {

```

```
        //We first load the WasteIncinerator bayesian network which has multinomial and Gaussian variables.
        BayesianNetwork bn = BayesianNetworkLoader.loadFromFile("./networks/simulated/WasteIncinerator.bn");

        //We recover the relevant variables for this example: Mout which is normally distributed, and W which is multinomial.
        Variable varMout = bn.getVariables().getVariableByName("Mout");
        Variable varW = bn.getVariables().getVariableByName("W");

        //Set the evidence.
        Assignment assignment = new HashMapAssignment(1);
        assignment.setValue(varW,0);

        //Then we query the posterior of
        System.out.println("P(Mout|W=0) = " + InferenceEngine.getPosterior(varMout, bn, assignment));

        //Or some more refined queries
        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 ));

    }

}
```

[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).

```
package eu.amidst.core.examples.inference;

import eu.amidst.core.inference.InferenceAlgorithm;
import eu.amidst.core.inference.messagepassing.VMP;
import eu.amidst.core.io.BayesianNetworkLoader;
import eu.amidst.core.models.BayesianNetwork;
import eu.amidst.core.variables.Assignment;
import eu.amidst.core.variables.HashMapAssignment;
import eu.amidst.core.variables.Variable;

/**
 *
```

```java
 * This example we show how to perform inference on a general Bayesian network using the Variational Message Passing (VMP)
 * algorithm detailed in
 *
 * <i> Winn, J. M., and Bishop, C. M. (2005). Variational message passing. In Journal of Machine Learning Research (pp. 661-694). </i>
 *
 * Created by andresmasegosa on 18/6/15.
 */
public class VMPExample {

    public static void main(String[] args) throws Exception {

        //We first load the WasteIncinerator bayesian network which has multinomial and Gaussian variables.
        BayesianNetwork bn = BayesianNetworkLoader.loadFromFile("./networks/simulated/WasteIncinerator.bn");

        //We recover the relevant variables for this example: Mout which is normally distributed, and W which is multinomial.
        Variable varMout = bn.getVariables().getVariableByName("Mout");
        Variable varW = bn.getVariables().getVariableByName("W");

        //First we create an instance of a inference algorithm. In this case, we use the VMP class.
        InferenceAlgorithm inferenceAlgorithm = new VMP();
        //Then, we set the BN model
        inferenceAlgorithm.setModel(bn);

        //If exists, we also set the evidence.
        Assignment assignment = new HashMapAssignment(1);
        assignment.setValue(varW,0);
        inferenceAlgorithm.setEvidence(assignment);

        //Then we run inference
        inferenceAlgorithm.runInference();

        //Then we query the posterior of
        System.out.println("P(Mout|W=0) = " + inferenceAlgorithm.getPosterior(varMout));

        //Or some more refined queries
        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 ));

        //We can also compute the probability of the evidence
        System.out.println("P(W=0) = "+Math.exp(inferenceAlgorithm.
```

```
                getLogProbabilityOfEvidence()));


    }
}
```

[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.

```java
package eu.amidst.core.examples.inference;


import eu.amidst.core.inference.ImportanceSampling;
import eu.amidst.core.io.BayesianNetworkLoader;
import eu.amidst.core.models.BayesianNetwork;
import eu.amidst.core.variables.Assignment;
import eu.amidst.core.variables.HashMapAssignment;
import eu.amidst.core.variables.Variable;

/**
 *
 * This example we show how to perform inference on a general Bayesian
     network using an importance sampling
 * algorithm detailed in
 *
 * <i> Fung, R., and Chang, K. C. (2013). Weighing and integrating
     evidence for
 * stochastic simulation in Bayesian networks. arXiv preprint arXiv
     :1304.1504.
 * </i>
 *
 * Created by andresmasegosa on 18/6/15.
 */
public class ImportanceSamplingExample {

    public static void main(String[] args) throws Exception {

        //We first load the WasteIncinerator bayesian network which has
            multinomial and Gaussian variables.
        BayesianNetwork bn = BayesianNetworkLoader.loadFromFile("./
            networks/simulated/WasteIncinerator.bn");

```

```
        //We recover the relevant variables for this example: Mout which is normally distributed, and W which is multinomial.
        Variable varMout = bn.getVariables().getVariableByName("Mout");
        Variable varW = bn.getVariables().getVariableByName("W");

        //First we create an instance of a inference algorithm. In this case, we use the ImportanceSampling class.
        ImportanceSampling inferenceAlgorithm = new ImportanceSampling();
        //Then, we set the BN model
        inferenceAlgorithm.setModel(bn);

        System.out.println(bn.toString());

        //If it exists, we also set the evidence.
        Assignment assignment = new HashMapAssignment(1);
        assignment.setValue(varW,0);
        inferenceAlgorithm.setEvidence(assignment);

        //We can also set to be run in parallel on multicore CPUs
        inferenceAlgorithm.setParallelMode(true);

        //To perform more than one operation, data should be keep in memory
        inferenceAlgorithm.setKeepDataOnMemory(true);

        //Then we run inference
        inferenceAlgorithm.runInference();

        //Then we query the posterior of
        System.out.println("P(Mout|W=0) = " + inferenceAlgorithm.getPosterior(varMout));

        //Or some more refined queries
        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 ));

        //We can also compute the probability of the evidence
        System.out.println("P(W=0) = "+Math.exp(inferenceAlgorithm.getLogProbabilityOfEvidence()));

    }
}
```

[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,

```
package eu.amidst.core.examples.learning;



import eu.amidst.core.datastream.DataInstance;
import eu.amidst.core.datastream.DataOnMemory;
import eu.amidst.core.datastream.DataStream;
import eu.amidst.core.io.DataStreamLoader;
import eu.amidst.core.learning.parametric.ParallelMaximumLikelihood;
import eu.amidst.core.learning.parametric.ParameterLearningAlgorithm;
import eu.amidst.core.models.BayesianNetwork;
import eu.amidst.core.models.DAG;
import eu.amidst.core.variables.Variable;
import eu.amidst.core.variables.Variables;

/**
 *
 * This other example shows how to learn incrementally the parameters of
      a Bayesian network using data batches
 *
 * Created by andresmasegosa on 18/6/15.
 */
public class MaximimumLikelihoodByBatchExample {


    /**
     * This method returns a DAG object with naive Bayes structure for
          the attributes of the passed data stream.
     * @param dataStream object of the class DataStream<DataInstance>
     * @param classIndex integer value indicating the position of the class
     * @return object of the class DAG
     */
    public static DAG getNaiveBayesStructure(DataStream<DataInstance
        > dataStream, int classIndex){

        //We create a Variables object from the attributes of the data
            stream
        Variables modelHeader = new Variables(dataStream.getAttributes
            ());

        //We define the predicitive class variable
        Variable classVar = modelHeader.getVariableById(classIndex);

        //Then, we create a DAG object with the defined model header
```

```java
        DAG dag = new DAG(modelHeader);

        //We set the linkds of the DAG.
        dag.getParentSets().stream().filter(w -> w.getMainVar() != classVar).forEach(w -> w.addParent(classVar));

        return dag;
    }


    public static void main(String[] args) throws Exception {

        //We can open the data stream using the static class DataStreamLoader
        DataStream<DataInstance> data = DataStreamLoader.open("datasets/simulated/WasteIncineratorSample.arff");

        //We create a ParameterLearningAlgorithm object with the MaximumLikehood builder
        ParameterLearningAlgorithm parameterLearningAlgorithm = new ParallelMaximumLikelihood();

        //We fix the DAG structure
        parameterLearningAlgorithm.setDAG(getNaiveBayesStructure(data,0));

        //We should invoke this method before processing any data
        parameterLearningAlgorithm.initLearning();


        //Then we show how we can perform parameter learnig by a sequential updating of data batches.
        for (DataOnMemory<DataInstance> batch : data.iterableOverBatches(100)){
            parameterLearningAlgorithm.updateModel(batch);
        }

        //And we get the model
        BayesianNetwork bnModel = parameterLearningAlgorithm.getLearntBayesianNetwork();

        //We print the model
        System.out.println(bnModel.toString());

    }

}
```

[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.

```java
package eu.amidst.core.examples.learning;


import eu.amidst.core.datastream.DataInstance;
import eu.amidst.core.datastream.DataStream;
import eu.amidst.core.io.DataStreamLoader;
import eu.amidst.core.learning.parametric.ParallelMaximumLikelihood;
import eu.amidst.core.models.BayesianNetwork;

/**
 *
 * This example shows how to learn in parallel the parameters of a Bayesian network from a stream of data using maximum
 * likelihood.
 *
 * Created by andresmasegosa on 18/6/15.
 */
public class ParallelMaximumLikelihoodExample {


    public static void main(String[] args) throws Exception {

        //We can open the data stream using the static class DataStreamLoader
        DataStream<DataInstance> data = DataStreamLoader.open("datasets/simulated/WasteIncineratorSample.arff");

        //We create a ParallelMaximumLikelihood object with the MaximumLikehood builder
        ParallelMaximumLikelihood parameterLearningAlgorithm = new ParallelMaximumLikelihood();

        //We activate the parallel mode.
        parameterLearningAlgorithm.setParallelMode(true);

        //We fix the DAG structure
        parameterLearningAlgorithm.setDAG(MaximimumLikelihoodByBatchExample.getNaiveBayesStructure(data, 0));

        //We set the batch size which will be employed to learn the model in parallel
        parameterLearningAlgorithm.setWindowsSize(100);

        //We set the data which is going to be used for leaning the parameters
```

```
        parameterLearningAlgorithm.setDataStream(data);

        //We perform the learning
        parameterLearningAlgorithm.runLearning();

        //And we get the model
        BayesianNetwork bnModel = parameterLearningAlgorithm.
            getLearntBayesianNetwork();

        //We print the model
        System.out.println(bnModel.toString());

    }

}
```

[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.

```

package eu.amidst.core.examples.learning;




import eu.amidst.core.datastream.DataInstance;
import eu.amidst.core.datastream.DataOnMemory;
import eu.amidst.core.datastream.DataStream;
import eu.amidst.core.io.DataStreamLoader;
import eu.amidst.core.learning.parametric.bayesian.SVB;
import eu.amidst.core.models.BayesianNetwork;
import eu.amidst.core.utils.DAGGenerator;

/**
 *
 * 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
 *
```

```java
 * <i> 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). </i>
 *
 *
 * Created by andresmasegosa on 18/6/15.
 */
public class SVBByBatchExample {


    public static void main(String[] args) throws Exception {

        //We can open the data stream using the static class DataStreamLoader
        DataStream<DataInstance> data = DataStreamLoader.open("datasets/simulated/WasteIncineratorSample.arff");

        //We create a SVB object
        SVB parameterLearningAlgorithm = new SVB();

        //We fix the DAG structure
        parameterLearningAlgorithm.setDAG(DAGGenerator.getHiddenNaiveBayesStructure(data.getAttributes(),"H",2));

        //We fix the size of the window, which must be equal to the size of the data batches we use for learning
        parameterLearningAlgorithm.setWindowsSize(5);

        //We can activate the output
        parameterLearningAlgorithm.setOutput(true);

        //We should invoke this method before processing any data
        parameterLearningAlgorithm.initLearning();


        //Then we show how we can perform parameter learning by a sequential updating of data batches.
        for (DataOnMemory<DataInstance> batch : data.iterableOverBatches(5)){
            double log_likelhood_of_batch = parameterLearningAlgorithm.updateModel(batch);
            System.out.println("Log-Likelihood of Batch: "+ log_likelhood_of_batch);
        }

        //And we get the model
        BayesianNetwork bnModel = parameterLearningAlgorithm.getLearntBayesianNetwork();

```

```
        //We print the model
        System.out.println(bnModel.toString());

    }

}
```

[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).

```

package eu.amidst.core.examples.learning;


import eu.amidst.core.datastream.DataInstance;
import eu.amidst.core.datastream.DataStream;
import eu.amidst.core.io.DataStreamLoader;
import eu.amidst.core.learning.parametric.bayesian.ParallelSVB;
import eu.amidst.core.models.BayesianNetwork;
import eu.amidst.core.utils.DAGGenerator;

/**
 *
 * This example shows how to learn the parameters of a Bayesian network from a stream of data with a Bayesian
 * approach using a **parallel** version of the following algorithm
 *
 * <i> 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). </i>
 *
 *
 * Created by andresmasegosa on 18/6/15.
 */
public class ParallelSVBExample {

    public static void main(String[] args) throws Exception {

        //We can open the data stream using the static class DataStreamLoader
```

```
        DataStream<DataInstance> data = DataStreamLoader.open("datasets/simulated/WasteIncineratorSample.arff");

        //We create a ParallelSVB object
        ParallelSVB parameterLearningAlgorithm = new ParallelSVB();

        //We fix the number of cores we want to exploit
        parameterLearningAlgorithm.setNCores(4);

        //We fix the DAG structure, which is a Naive Bayes with a global latent binary variable
        parameterLearningAlgorithm.setDAG(DAGGenerator.getHiddenNaiveBayesStructure(data.getAttributes(), "H", 2));

        //We fix the size of the window
        parameterLearningAlgorithm.getSVBEngine().setWindowsSize(100);

        //We can activate the output
        parameterLearningAlgorithm.setOutput(true);

        //We set the data which is going to be used for leaning the parameters
        parameterLearningAlgorithm.setDataStream(data);

        //We perform the learning
        parameterLearningAlgorithm.runLearning();

        //And we get the model
        BayesianNetwork bnModel = parameterLearningAlgorithm.getLearntBayesianNetwork();

        //We print the model
        System.out.println(bnModel.toString());

    }

}
```

[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.

```java
/*
 *
 *
 * Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
 * See the NOTICE file distributed with this work for additional information regarding copyright ownership.
 * The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you may not use
 * this file except in compliance with the License. You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software distributed under the License is
 * distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and limitations under the License.
 *
 *
 */

package eu.amidst.core.examples.conceptdrift;


import eu.amidst.core.conceptdrift.NaiveBayesVirtualConceptDriftDetector;
import eu.amidst.core.datastream.DataInstance;
import eu.amidst.core.datastream.DataOnMemory;
import eu.amidst.core.datastream.DataStream;
import eu.amidst.core.io.DataStreamLoader;
import eu.amidst.core.variables.Variable;

/**
 * This example shows how to use the class NaiveBayesVirtualConceptDriftDetector to run the virtual concept drift
 * detector detailed in
 *
 * <i>Borchani et al. Modeling concept drift: A probabilistic graphical model based approach. IDA 2015.</i>
 *
 */
public class NaiveBayesVirtualConceptDriftDetectorExample {
    public static void main(String[] args) {

        //We can open the data stream using the static class DataStreamLoader
```

```java
        DataStream<DataInstance> data = DataStreamLoader.open("./datasets/DriftSets/sea.arff");

        //We create a NaiveBayesVirtualConceptDriftDetector object
        NaiveBayesVirtualConceptDriftDetector virtualDriftDetector = new NaiveBayesVirtualConceptDriftDetector();

        //We set class variable as the last attribute
        virtualDriftDetector.setClassIndex(-1);

        //We set the data which is going to be used
        virtualDriftDetector.setData(data);

        //We fix the size of the window
        int windowSize = 1000;
        virtualDriftDetector.setWindowsSize(windowSize);

        //We fix the so-called transition variance
        virtualDriftDetector.setTransitionVariance(0.1);

        //We fix the number of global latent variables
        virtualDriftDetector.setNumberOfGlobalVars(1);

        //We should invoke this method before processing any data
        virtualDriftDetector.initLearning();

        //Some prints
        System.out.print("Batch");
        for (Variable hiddenVar : virtualDriftDetector.getHiddenVars()) {
            System.out.print("\t" + hiddenVar.getName());
        }
        System.out.println();


        //Then we show how we can perform the sequential processing of
        // data batches. They must be of the same value than the window
        // size parameter set above.
        int countBatch = 0;
        for (DataOnMemory<DataInstance> batch : data.iterableOverBatches(windowSize)){

            //We update the model by invoking this method. The output
            // is an array with a value associated
            // to each fo the global hidden variables
            double[] out = virtualDriftDetector.updateModel(batch);

            //We print the output
            System.out.print(countBatch + "\t");
            for (int i = 0; i < out.length; i++) {
                System.out.print(out[i]+"\t");
```

```
            }
            System.out.println();
            countBatch++;
        }
    }
}
```

[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

```
package eu.amidst.core.examples.huginlink;

import COM.hugin.HAPI.Domain;
import COM.hugin.HAPI.ExceptionHugin;
import eu.amidst.core.models.BayesianNetwork;
import eu.amidst.huginlink.converters.BNConverterToAMIDST;
import eu.amidst.huginlink.converters.BNConverterToHugin;
import eu.amidst.huginlink.io.BNLoaderFromHugin;

/**
 * Created by rcabanas on 24/06/16.
 */
public class HuginConversionExample {
    public static void main(String[] args) throws ExceptionHugin {
        //We load from Hugin format
        Domain huginBN = BNLoaderFromHugin.loadFromFile("./
            networks/simulated/WasteIncinerator.bn");

        //Then, it is converted to AMIDST BayesianNetwork object
        BayesianNetwork amidstBN = BNConverterToAMIDST.
            convertToAmidst(huginBN);

        //Then, it is converted to Hugin Bayesian Network object
        huginBN = BNConverterToHugin.convertToHugin(amidstBN);

        System.out.println(amidstBN.toString());
        System.out.println(huginBN.toString());

    }
}
```

[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

```
package eu.amidst.core.examples.huginlink;

import COM.hugin.HAPI.Domain;
import COM.hugin.HAPI.ExceptionHugin;
import eu.amidst.core.models.BayesianNetwork;
import eu.amidst.huginlink.converters.BNConverterToAMIDST;
import eu.amidst.huginlink.io.BNLoaderFromHugin;
import eu.amidst.huginlink.io.BayesianNetworkWriterToHugin;

/**
 * Created by rcabanas on 24/06/16.
 */
public class HuginIOExample {
    public static void main(String[] args) throws ExceptionHugin {
        //We load from Hugin format
        Domain huginBN = BNLoaderFromHugin.loadFromFile("networks
            /asia.net");

        //We save a AMIDST BN to Hugin format
        BayesianNetwork amidstBN = BNConverterToAMIDST.
            convertToAmidst(huginBN);
        BayesianNetworkWriterToHugin.save(amidstBN,"networks/tmp.
            net");

    }
}
```

[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

```
package eu.amidst.core.examples.huginlink;

import eu.amidst.core.inference.InferenceAlgorithm;
import eu.amidst.core.io.BayesianNetworkLoader;
import eu.amidst.core.models.BayesianNetwork;
import eu.amidst.core.variables.Assignment;
import eu.amidst.core.variables.HashMapAssignment;
import eu.amidst.core.variables.Variable;
import eu.amidst.huginlink.inference.HuginInference;

import java.io.IOException;

```

```java
/**
 * Created by rcabanas on 24/06/16.
 */
public class HuginInferenceExample {
    public static void main(String[] args) throws IOException, ClassNotFoundException {
        //We first load the WasteIncinerator bayesian network
        //which has multinomial and Gaussian variables.
        BayesianNetwork bn = BayesianNetworkLoader.loadFromFile("./networks/WasteIncinerator.bn");

        //We recover the relevant variables for this example:
        //Mout which is normally distributed, and W which is multinomial.
        Variable varMout = bn.getVariables().getVariableByName("Mout");
        Variable varW = bn.getVariables().getVariableByName("W");

        //First we create an instance of a inference algorithm.
        //In this case, we use the ImportanceSampling class.
        InferenceAlgorithm inferenceAlgorithm = new HuginInference();

        //Then, we set the BN model
        inferenceAlgorithm.setModel(bn);

        //If exists, we also set the evidence.
        Assignment assignment = new HashMapAssignment(1);
        assignment.setValue(varW, 0);
        inferenceAlgorithm.setEvidence(assignment);

        //Then we run inference
        inferenceAlgorithm.runInference();

        //Then we query the posterior of
        System.out.println("P(Mout|W=0) = " + inferenceAlgorithm.getPosterior(varMout));

        //Or some more refined queries
        System.out.println("P(0.7<Mout<3.5 | W=0) = "
                + inferenceAlgorithm.getExpectedValue(varMout, v -> (0.7 < v && v < 3.5) ? 1.0 : 0.0));

    }
}
```

[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.

```java
package eu.amidst.core.examples.huginlink;

import eu.amidst.core.inference.InferenceAlgorithm;
import eu.amidst.core.io.BayesianNetworkLoader;
import eu.amidst.core.models.BayesianNetwork;
import eu.amidst.core.variables.Assignment;
import eu.amidst.core.variables.HashMapAssignment;
import eu.amidst.core.variables.Variable;
import eu.amidst.huginlink.inference.HuginInference;

import java.io.IOException;

/**
 * Created by rcabanas on 24/06/16.
 */
public class HuginInferenceExample {
    public static void main(String[] args) throws IOException, ClassNotFoundException {
        //We first load the WasteIncinerator bayesian network
        //which has multinomial and Gaussian variables.
        BayesianNetwork bn = BayesianNetworkLoader.loadFromFile("./networks/WasteIncinerator.bn");

        //We recover the relevant variables for this example:
        //Mout which is normally distributed, and W which is multinomial.
        Variable varMout = bn.getVariables().getVariableByName("Mout");
        Variable varW = bn.getVariables().getVariableByName("W");

```

```java
            //First we create an instance of a inference algorithm.
            //In this case, we use the ImportanceSampling class.
            InferenceAlgorithm inferenceAlgorithm = new HuginInference();

            //Then, we set the BN model
            inferenceAlgorithm.setModel(bn);

            //If exists, we also set the evidence.
            Assignment assignment = new HashMapAssignment(1);
            assignment.setValue(varW, 0);
            inferenceAlgorithm.setEvidence(assignment);

            //Then we run inference
            inferenceAlgorithm.runInference();

            //Then we query the posterior of
            System.out.println("P(Mout|W=0) = " + inferenceAlgorithm.getPosterior(varMout));

            //Or some more refined queries
            System.out.println("P(0.7<Mout<3.5 | W=0) = "
                    + inferenceAlgorithm.getExpectedValue(varMout, v -> (0.7 < v && v < 3.5) ? 1.0 : 0.0));

      }
}
```

[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.

```
java -Xmx512m -cp "../lib/*" -javaagent:../lib/sizeofag-1.0.0.jar
moa.DoTask EvaluatePrequential -l \(bayes.AmidstClassifier -g 1
-m 2\) -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.

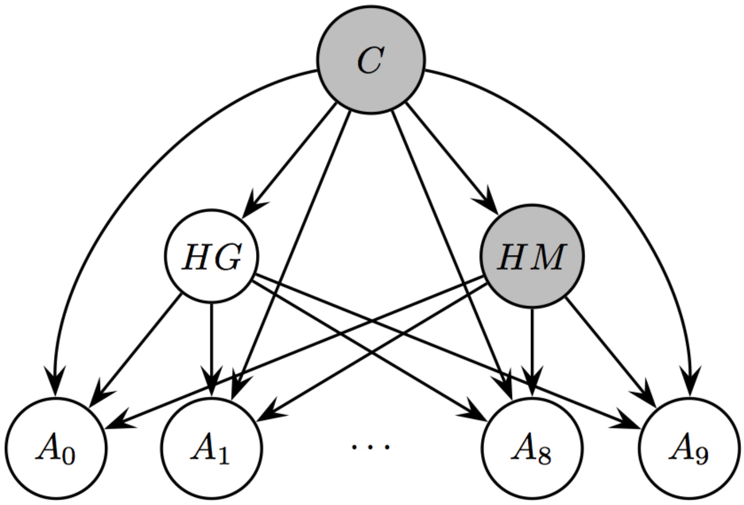


Figure 1: HODE example

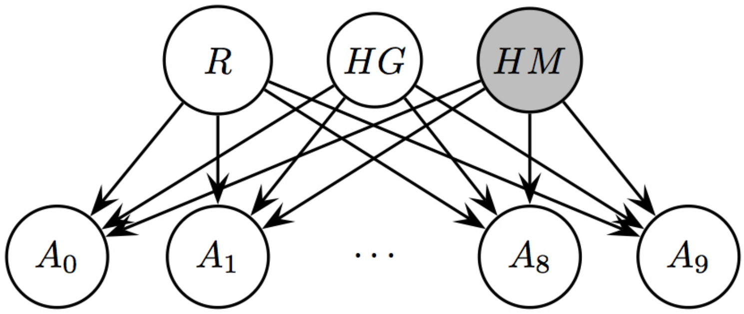


Figure 2: HODE regression example

```
java -Xmx512m -cp "../lib/*" -javaagent:../lib/sizeofag-1.0.0.jar
moa.DoTask EvaluatePrequentialRegression -l bayes.AmidstRegressor
-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]