Heri Wijayanto,ST.,MT.
Information Technological Engineering, Engineering Faculty, University of Mataram
January, 16th 2013

Tutorial
Bayesian Network Programming by Java and NetBeans

Bayesian Network is applied widely in computer application especially for making decisions in uncertain domain that uses probabilistic Bayesian theorem. Predicting posterior state by one or more evident can be done by forward inference, or diagnosing some evidences by backward inference if posterior state is known. This tutorial gives explanation of the steps of implementing Bayesian Network by Java programming language and NetBeans IDE (Integrated Development Environment). Library that is used is JavaBayes which is available in http://www.cs.cmu.edu/~javabayes/ as an opensource.

After downloading JavaBayes bundle and making "jar" file, it is added to NetBeans library. In the javabayes classes folder, execute this command to make jar file.

Figure 1. Making JAR file

And then, create a new project in NetBeans and add this jar file to the library

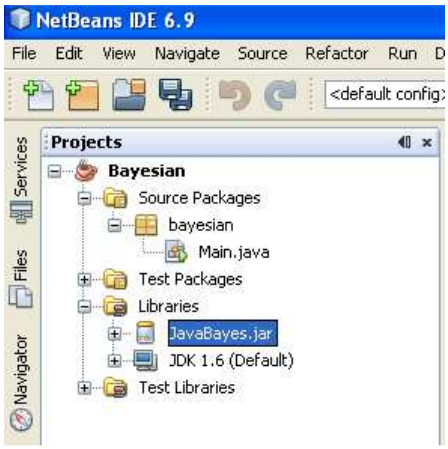

Figure 2. Adding Jar file to the NetBeans library

Adding JavaBayes.jar on the library is very simple, just click right button on the Libraries under Bayesian Project, click zip/jar, and then chose JavaBayes.jar file. Source code for testing BN can be downloaded from [Schweitzer] and that source code is shown below.

```
package bayesian;
import InferenceGraphs.*;
import java.util.Vector;
import QuasiBayesianInferences.*;
public class Main {
   public static void main(String[] args) {
      InferenceGraph ig = new InferenceGraph();
      // Setup nodes with states
      InferenceGraphNode hearBark = createNode(ig, "hearBark", "hearBark", "quiet");
      InferenceGraphNode dogOut = createNode(ig, "dogOut", "dogOut", "dogIn");
      InferenceGraphNode bowelProblem = createNode(ig, "bowelProblem", "bowelProblem", "noBowelProblem");
      InferenceGraphNode familyOut = createNode(ig, "familyOut", "familyOut", "familyIn");
      InferenceGraphNode lightOn = createNode(ig, "lightOn", "lightOn", "lightOff");
      // Add links (parent, child) where parent usually causes child to happen
      ig.create_arc(familyOut, lightOn);
      ig.create_arc(familyOut, dogOut);
      ig.create_arc(bowelProblem, dogOut);
      ig.create_arc(dogOut, hearBark);

      // Setup conditional probabilities
      setProbabilityValues(hearBark, "dogOut", .70, .30);
      setProbabilityValues(hearBark, "dogIn", .01, .99);

      setProbabilityValues(dogOut, "familyOut", "bowelProblem", .99, .01);
      setProbabilityValues(dogOut, "familyOut", "noBowelProblem", .90, .10);
      setProbabilityValues(dogOut, "familyIn", "bowelProblem", .97, .03);
      setProbabilityValues(dogOut, "familyIn", "noBowelProblem", .30, .70);

      setProbabilityValues(lightOn, "familyOut", .60, .40);
      setProbabilityValues(lightOn, "familyIn", .05, .95);

      // Setup 'leaf' probabilities
      setProbabilityValues(familyOut, .15, .85);
      setProbabilityValues(bowelProblem, .01, .99);

      double belief = getBelief(ig, lightOn);
      System.out.println("The probability of the light being on is " + belief);

      // Enter evidence, certain things that we observe
      hearBark.set_observation_value("hearBark");
      bowelProblem.set_observation_value("noBowelProblem");

      // Recalculate probability of light being on given evidence
      belief = getBelief(ig, lightOn);
      System.out.println("The probability of the light being on given a bark was heard "
                         + "and no bowel problem is " + belief);
   }
```

```
/**
* Helper function to create node since not as straightforward with JavaBayes
* to get a pointer back to the node that is being added
*/
private static InferenceGraphNode createNode(
                        InferenceGraph ig,
        String name,
        String trueVariable,
        String falseVariable) {
          ig.create_node(0, 0);
          InferenceGraphNode node = (InferenceGraphNode) ig.get_nodes().lastElement();

          node.set_name(name);
          ig.change_values(node, new String[] {trueVariable, falseVariable});

          return node;
}
private static void setProbabilityValues(InferenceGraphNode node, String firstParentVariable,
                        String secondParentVariable, double trueValue, double falseValue) {

          int variableIndex = (getVariableIndex(node, firstParentVariable) * 2) +
                                  getVariableIndex(node, secondParentVariable);
          int totalValues = getTotalValues(node, firstParentVariable) +
                                  getTotalValues(node, secondParentVariable);

          double[] probabilities = node.get_function_values();
          probabilities[variableIndex] = trueValue;
          probabilities[variableIndex + totalValues] = falseValue;
          node.set_function_values(probabilities);
}
/**
* Sets probabilities for a node that has a parent
*/
private static void setProbabilityValues(InferenceGraphNode node, String parentVariable,
                        double trueValue, double falseValue) {
          int variableIndex = getVariableIndex(node, parentVariable);
          int totalValues = getTotalValues(node, parentVariable);

          double[] probabilities = node.get_function_values();
          probabilities[variableIndex] = trueValue;
          probabilities[variableIndex + totalValues] = falseValue;
          node.set_function_values(probabilities);
}
/**
* Sets probabilities for a leaf node
*/
private static void setProbabilityValues(InferenceGraphNode node, double trueValue, double falseValue) {
          node.set_function_values(new double[] {trueValue, falseValue});
}
```

```
  /**
  * Returns the index of the variable for the parent that has the given variable
  */
  private static int getVariableIndex(InferenceGraphNode node, String parentVariable) {

        for (InferenceGraphNode parent : (Vector<InferenceGraphNode>) node.get_parents()) {
              int variableIndex = 0;

              for (String variable : parent.get_values()) {
                    if (variable.equals(parentVariable)) {
                          return variableIndex;
                    }

                    variableIndex++;
              }
        }

        return 0;
  }

  /**
  * Returns the total number of values for the parent that has the given variable
  */
  private static int getTotalValues(InferenceGraphNode node, String parentVariable) {
        for (InferenceGraphNode parent : (Vector<InferenceGraphNode>) node.get_parents()) {

              for (String variable : parent.get_values()) {
                    if (variable.equals(parentVariable)) {
                          return parent.get_number_values();
                    }
              }
        }

        return 0;
  }
  /**
  * Gets the belief/true result from the inference of the given node
  */
  private static double getBelief(InferenceGraph ig, InferenceGraphNode node) {
        QBInference qbi = new QBInference(ig.get_bayes_net(), false);
        qbi.inference(node.get_name());
        return qbi.get_result().get_value(0);
  }

}
```

The result of executing source code above which has output of calculating inference is 0.2365, is presented in figure 3 below.

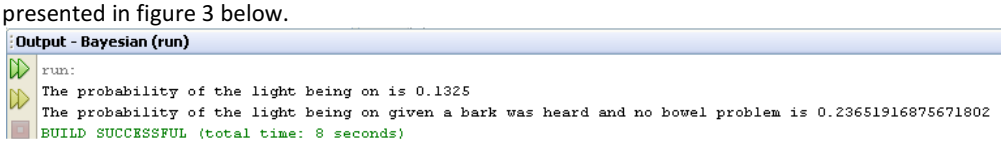

Figure 3. Result of executing source code

Reference

Schweitzer Joe; Simple Bayesian Network Inference Using Netica and JavaBayes http://www.dataworks-inc.com/site/blog/simple-bayesian-network-inference-using-netica-and-javabayes