Readme for BatchTool 1.5

Instructions:

To unzip the file batchtool.zip:

1. Open batchtool.zip using the Windows utility WinZip, available from www.winzip.com.
2. Extract all the contents of batchtool.zip into the folder where SamIam v2.3 beta is installed.

To run from an MS DOS shell:

1. Change to the installation directory.
2. Execute the command "batchtool.bat <input file> <output file>"

To invoke via system call:

1. Call "batchtool.bat <input file> <output file>"

For questions or feedback, contact:

SamIam Tech Support
samiam@cs.ucla.edu

List of all files:

- batchtool.bat – MS DOS executable script to conveniently invoke BatchTool.
- batch.jar – Java archive file needed to run BatchTool.
- example_query.xml – example input file for the BatchTool program.
- batchquery.dtd – the formal DTD (document type definition) for the input and output XML files of BatchTool.
- samiam.jar, inflib.jar – BatchTool requires SamIam v2.3 beta.

System Requirements

- JRE (Java Runtime Environment) version 1.4 or later – java.exe must be available from the Windows PATH.
- SamIam version 2.3 beta

Overview

BatchTool is a command line interface to the same reasoning engine that powers SamIam v2.3 beta. BatchTool's purpose is to make it easy to execute many MAP, MPE, Pr(e), posterior probability and sensitivity queries programmatically, without opening the SamIam GUI. BatchTool requires as input a file that defines each query you would like BatchTool to execute, and outputs a file containing the results of those queries and any errors that occurred. The output file is identical to the input file, with the addition that BatchTool inserts the query results. The DTD **batchquery.dtd** formally defines the

XML format of the input/output files. We also provide an example query file that conforms to the formal definition: **example_query.xml**.

Command-Line Arguments

The simplest form of the command to invoke BatchTool is:

batchtool.bat <input.xml> <output.xml>

BatchTool also supports two optional command-line flags: –validate and –debug.

- **–validate**: ask BatchTool to use a validating XML parser to parse the input file. With the –validate option on, BatchTool will fail to run on an incorrectly formed input file.
- **–debug**: ask BatchTool to output verbose error messages to the console.

Here is an example of invoking BatchTool with the validate and debug flags on:

batchtool.bat –validate –debug example_query.xml results.xml

Input File: Informal Description

The name of the root element of the XML input file is <batch>. <batch> can have an arbitrary number of child elements named <query>. We require <query> elements to have an attribute @network, the value of which is the full path to the Hugin .net, Genie .dsl or .xdsl, Interchange .dsc, Netica .dne, Ergo .erg file that defines a Bayesian network. <query> can have an arbitrary number of child elements named <map>, <exactmap>, <mpe>, <probability>, and <sensitivity>, which define Maximum a Posteriori, complete Maximum a Posteriori, Most Probably Explanation, Pr(e)/posterior, and Sensitivity queries respectively.

<map> elements may have three optional attributes that define configuration parameters for the map query: @search, @initialization and @maxsteps. @search defines the search method, one of {HILL, TABOO} (default TABOO). @initialization defines the initialization method, one of {RANDOM, MPE, MAX_LIKELIHOODS, SEQUENTIAL} (default SEQUENTIAL). @maxsteps defines an integer bound on the number of search steps (default 25). <map> can have an arbitrary number of child elements named <id> which contain the variable identifiers of the MAP variables, and an arbitrary number of child elements named <evidence> which define known instantiations of network variables under which to execute MAP. <evidence> elements must have **exactly one** child element named <id>, which contains the variable identifier of the evidence variable, and **exactly one** child element named <state>, which contains the identifier of the discrete state of that variable to instantiate.

<exactmap> elements may have three optional attributes that define configuration parameters for the complete map query: @timelimit, @slop, and @widthbarrier. @timelimit defines a time out in seconds that limits the complete search. A value of zero gives the algorithm unlimited time, which is the default value for time limit. @slop

defines the threshold for a sloppy MAP search. If @slop is defined, BatchTool will return as many exact MAP results as exist with probability within @slop of the highest probability result. The structure of the children of <exactmap> is identical to that of <map>.

<mpe> elements do not contain attributes. They can have an arbitrary number of child elements named <evidence>, as described above, which define the evidence under which to execute MPE.

<probability> elements may have three optional configuration attributes: @algorithm, @cachefactor, @noposteriors and @prune. @algorithm defines the inference algorithm to use, one of { shenoy-shafer, hugin, zc-hugin, recursiveconditioning } (default shenoy-shafer). For backwards compatibility, @algorithm will also accept the value "jointree", equivalent to "shenoy-shafer". @cachefactor only applies when @algorithm = recursiveconditioning. It defines the proportion of memory the recursive conditioning engine should allocate out of the optimal amount (default 1.0). @noposteriors is a flag that turns off the calculation of posterior probabilities. @prune is valid only in the context of the join tree algorithms: { shenoy-shafer, hugin, zc-hugin }. @prune is a flag that, if true, instructs BatchTool to perform query based pruning of the join tree. Query based pruning is a technique to improve the memory and time cost of a query by eliminating unused portions of the join tree data structure. It will be most effective when the number of variables over which you ask for posterior probabilities is smaller than the total number of variables in the network. <probability> can have an arbitrary number of child elements named <id> which contain the variable identifiers of the variables over which to calculate posterior probabilities, and an arbitrary number of child elements named <evidence> which define known instantiations of network variables under which to calculate Pr(e) and the posteriors. If <probability> lacks @noposteriors and contains no children <id>, then the program default is to calculate posterior probabilities for every variable in the network.

<sensitivity> elements may have six optional configuration attributes: @algorithm, @inequality, @twoeventop, @constant, @oneparam, @onecpt. @algorithm defines the inference algorithm to use, one of {shenoy-shafer, zc-hugin} (default shenoy-shafer). @inequality defines the comparison operator part of the sensitivity constraint, one of {EQUALS, GTE, LTE} (default GTE). @twoeventop only applies to 2-event constraints. It defines the arithmetic operator part of the 2-event constraint, one of {DIFFERENCE, RATIO} (default DIFFERENCE). @constant defines the epsilon constant part of the constraint. It can be any decimal number. The default for 1-event and 2-event-difference constraints is 0. The default for 2-event-ratio constraints is 1. @oneparam is a flag attribute that can be used to turn off sensitivity suggestions of the form that pertain to single CPT parameters (default true). To turn off single CPT parameter suggestions, set oneparam="false". @onecpt is a flag attribute that can be used to turn off sensitivity suggestions of the form that pertain to multiple parameter suggestions within a single CPT (default true). To turn off multiple parameter suggestions, set onecpt="false". <sensitivity> elements must have at least one, but no more than two child elements of type <event>. <sensitivity> may also contain children <evidence> elements that define the evidence under which to perform the sensitivity query. In the case that <sensitivity> lacks <evidence> children, BatchTool will perform the query with no evidence set.

<event> elements define the events that are part of a sensitivity constraint. An event element may contain a single optional attribute: @name. @name is a way of naming events for a 2-event constraint. Named events will be considered in alphabetical order. In other words, BatchTool will treat the event with the name that occurs earlier in the alphabet as the left-hand operand in a 2-event constraint. In the case that you define a 2-event constraint with unnamed events, or events with the same name, BatchTool will order them arbitrarily. For clarity, we recommend that you name the left-hand event "1" and the right-hand event "2". Each <event> must have a single child <id> and a single child <state>.

Output File: Informal Description

The file BatchTool outputs will be identical in form and content to the input file, with the exception that some additional elements will be added to it.

BatchTool adds <result> elements to <map>, <exactmap>, <mpe>, <probability> and <sensitivity> elements. <result> elements will have an attribute named @date the value of which is the system date and time when the query was executed. BatchTool also adds an element <timing> with name "network_load" to <query> to report the time it took to load the network from the file.

<result> elements for <mpe> and <sensitivity> queries will have two additional attributes: @cpuMilliseconds and @systemElapsedMilliseconds. The values of these attributes represent the milliseconds it took BatchTool to calculate the answer to the query. @cpuMilliseconds represents a true cpu time profile, whereas @systemElapsedMilliseconds is the time that elapsed on the system clock. We provide @systemElapsedMilliseconds to users for whom @cpuMilliseconds may be unavailable, or for comparison purposes. @cpuMilliseconds will be available only on systems running the Java virtual machine profiler that comes with SamIam v2.3 beta. If that library is not present or disabled for some reason, the program will report the following value for @cpuMilliseconds: "NO PROFILER". <result> elements for <map>, non-sloppy <exactmap> and <mpe> queries may have an arbitrary number of child elements named <inst>, which each define a variable instantiation as part of the answer of the MAP or MPE query. These elements will also have a child element <score> that contains the probability value for the result instantiation jointed with the evidence. Non-sloppy <exactmap> will contain the child element <exact>. If the complete map search finished and the answer is exact, this element will contain the value "true".

<inst> elements must have exactly one child named <id> and one child named <state>. (<id> and <state> described above.)

<result> elements for <map> and <exactmap> will report detailed timings broken down by task. They will each include three child elements <timing>, that report both the cpu thread time and system elapsed time for each of the following phases of the MAP computation: (1) pruning, (2) initialization, (3) search. The @name for these <timing> elements will be "prune", "init", and "search."

<timing> elements contain three attributes. @name is a descriptive name that indicates what task the timing describes. @cpuMilliseconds is the profiled thread time, if available. @systemElapsedMilliseconds is the time that elapsed on the system clock.

<result> elements for <exactmap slop="0.XX"> queries will contain an arbitrary number of child elements <sloppyresult>. <sloppyresult> elements will contain attributes @score, @exact and @sloppynumber. The value of @score is the probability of the result instantiation jointed with the evidence. @exact will be "true" if the exact map query finished finding an exact answer within the specified time limit. The value of @sloppynumber is an integer that represents the position of the result when all results are sorted by score. @sloppynumber 0 is the highest scoring result. <sloppyresult> will contain an arbitrary number of <inst> that define the result instatiation.

<result> elements for <probability> queries will contain a child element <probabilityOfEvidence>, and may contain a child element <posteriors>.

<probabilityOfEvidence> elements will contain the attributes @value, @cpuMilliseconds, and @systemElapsedMilliseconds. @value is the calculated Pr(e) value. The other two attributes are the timing data for only the Pr(e) calculation, and their meanings correspond to those described above for <result>. <probabilityOfEvidence> will contain no child elements.

<posteriors> defines posterior probabilities over a set of variables. <posteriors> will contain the timing attributes @cpuMilliseconds, and @systemElapsedMilliseconds as described above, reporting the time to calculate only the posteriors. <posteriors> may contain an arbitrary number of child elements <variable>. <variable> has an attribute @id, which names the variable identifier, and contains at least one child element <posterior>. <posterior> has attributes @state and @value, which assign a posterior probability value to a state of the variable named by @id. <posterior> contains no child elements.

<result> elements for <sensitivity> queries may contain three types child element: <suggestion>, <satisfied> and <unsatisfiable>. In the case that the specified constraint is already satisfied by the input belief network, or is unsatisfiable, <result> will contain a single child element of type <satisfied> or <unsatisfiable> respectively. All result suggestions from a sensitivity query will be in the form of <suggestion>.

<suggestion> will contain the attribute @type which will identify the suggestion as "oneparam", meaning a suggestion pertaining to a single CPT parameter, or "onecpt", meaning a multiple parameter suggestion. <suggestion> will also contain the attribute @logoddschange, which declares the scalar log odds change for the suggestion. <suggestion> will contain child elements of type <parameter>. Single parameter suggestions will contain exactly one such child. Multiple parameter suggestions will contain an arbitrary number of <parameter> children, but at least one.

<parameter> elements may contain three attributes: @current – the current CPT value for the parameter, @suggested – the suggested change to that CPT value (its value will take the form of an interval), @absolutechange – the absolute change (its value will take the form of an interval). <parameter> contains exactly one child <prob_of> and one child <given>.

<prob_of> defines the variable instantiation for the joint variable. It contains exactly one child of type <inst>.

<given> defines the variable instantiation for the parent (conditioned) variables. It may contain an arbitrary number of <inst> elements, including none for root (parentless) variable parameters.

If BatchTool encounters errors while attempting to execute your queries, it will insert <error> elements. If an error occurred opening or parsing a Bayesian network file, then BatchTool will insert an <error> element as a child of the corresponding <query> element. If an error occurred in the definition of a MAP, MPE, Pr(e), posteriors, or sensitivity query, BatchTool will insert an <error> element as a child of the erroneous <map>, <exactmap>, <mpe>, <probability>, or <sensitivity> element. <error> elements contain only a textual description of the error that occurred.

Using –validate: Referring to the DTD

In order to use -validate, your xml input file must make reference to a DTD (xml document type definition). BatchTool v1.3 comes with a DTD file named "batchquery.dtd." In order to refer to that dtd file in your xml input, please add a <!DOCTYPE> element of the following form (substituting the absolute path to "batchquery.dtd" on your machine):

<!DOCTYPE batch SYSTEM "file:/c:\batchquery.dtd">

Please refer to the example query file that comes with BatchTool v1.3, "example_query.xml", for an example of a <!DOCTYPE> element. Once you insert that element, you can call BatchTool with the following command:

batchtool.bat -validate <input.xml> <output.xml>

With the "-validate" option, BatchTool will report errors in the input file format to the console. If it reports no errors, you can be certain that your input file conforms to the structure BatchTool expects. For further instructions on how to form the input correctly, you can also refer to the section titled "Input File: Informal Description" in this readme document.