10. BAYESIAN NETWORK CLASSIFIERS

RENATO LUI GEH
NUSP: 8536030

Abstract. This document contains the documentation to a Naive Bayes implementation.

1. LibDAI

We use LibDAI [Moo10] as an inference framework. LibDAI requires the following packages as dependencies:

- gcc
- make
- Boost (version 1.37+)
- GMP library
- graphviz

Under Linux they can easily be installed via the following command:

```
# Arch Linux
sudo pacman -S boost boost-libs make g++ gmp lib32-gmp
# Ubuntu/Debian
apt-get install g++ make graphviz libboost-dev libboost
    -graph-dev libboost-program-options-dev libboost-
    test-dev libgmp-dev
```

Package names should be similar for other Linux distributions'. Installation is straightforward:

```
git clone https://bitbucket.org/jorism/libdai.git
cd libdai
# Choose the Makefile configuration accordingly (LINUX
    for Linux, MACOSX for OS X, etc.)
cp Makefile.LINUX Makefile.conf
make
# The following are tests. They should run fine if the
    build was successful
examples/example test/alarm.fg
tests/testdai --aliases test/aliases.conf --filename
    tests/alarm.fg --methods JTREE_HUGIN BP_SEQMAX
# (Optional) Install LibDAI to your user path
```

```
cp -R include/dai /usr/include
cp -R lib/* /usr/lib/
```

We assume LibDAI is installed to your user path. If that is not the case, modify the Makefile in `/src` accordingly.

2. Compiling

To compile:

```
cd src
make
```

To compile with `DEBUG=true` (i.e. flag `-g`):

```
make clean
make DEBUG=true
```

3. Running an example

There is an example source code in `main.cc`. We use an example input dataset `input/spam-train.in`. To run:

```
./run input/spam-train.in
```

4. NaiveBayes

The implementation of a Naive Bayes model is in the files `src/nbayes.(cc/h)`. The class `NaiveBayes` represents a Naive Bayes model.

4.1. *Learning*

A `NaiveBayes` model can learned from a data file by calling the following static function:

```
NaiveBayes nb = NaiveBayes::Learn(const char* filename,
    int n);
```

The resulting model has `n` attribute nodes and one class node as specified in file `filename`. The input file must be structured as follows:

```
@relation name-of-dataset

@attribute attribute-1 {0,1}
@attribute attribute-2 {0,1,2}
⋮
```

```
@attribute attribute-n {0,1}
@attribute class {0,1}

@data
0,...,0
⋮ ⋱ ⋮
1,...,1
```

Empty lines are not optional. Internally, `NaiveBayes` learns its parameters through MLE.

4.2. *Inference*

Performing classification is done with two methods:

```
std::vector<dai::Var> GetVars(std::vector<int> labels);
size_t Classify(std::vector<dai::Var> atts, std::vector
   <int> val, bool output=true);
```

Method `GetVars` takes an ordered collection of variable labels and returns a vector of `dai::Var`. Method `Classify` takes a vector of attribute variables and their valuations and returns the most probable instantiation of the class node. The optional argument `output` outputs additional information on the classification.

A typical call for classification on variable labels `, 2, 5, 111, 2, 5, 11` and their instantiations `, 1, 1, 11, 1, 1, 1`:

```
Let nb be a NaiveBayes model
size_t c = nb.Classify(nb.GetVars({1, 2, 5, 11}, {1, 1,
    1, 1}));
```

4.3. *Output*

There are two `NaiveBayes` methods that can provide additional output on the model. Method `WriteToFile` outputs the Naive Bayes model in LibDAI format and `DrawGraph` outputs a `dot` file containing the underlying graph.

```
void NaiveBayes::WriteToFile(const char *name);
void NaiveBayes::DrawGraph(const char *name);
```

References

[Moo10] Joris M. Mooij. "libDAI: A Free and Open Source C++ Library for Discrete Approximate Inference in Graphical Models". In: *Journal of Machine Learning Research* 11 (Aug. 2010), pp. 2169–2173. URL: http://www.jmlr.org/papers/volume11/mooij10a/mooij10a.pdf.