Software Libraries for PGMs

Kevin Rothi
Very popular tools for ML/NNs/Deep Learning...

- SciKit Learn
- Tensorflow
- Keras
- Torch
- CUDA
- Theano
- Caffe
No shortage of small libraries for graphical models…

http://www.cs.ubc.ca/~murphyk/Software/bnsoft.html

(Last updated 16 June 2014)

69 Libraries
<table>
<thead>
<tr>
<th>Name</th>
<th>Authors</th>
<th>Src</th>
<th>Cita</th>
<th>G2E</th>
<th>Flame</th>
<th>Shrew</th>
<th>Utility</th>
<th>Test</th>
<th>Under</th>
<th>Reference</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agglomerate</td>
<td>A. Agnesi</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>D</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Anomaly</td>
<td>G. R. H.</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>D</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Attribute</td>
<td>K. McAlister</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Behavioral</td>
<td>T. K. McLean</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Categorization</td>
<td>P. S.</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Classification</td>
<td>A. K. McHugh</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Clustering</td>
<td>B. V. McAuley</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Conceptual</td>
<td>C. F. McCallen</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Deduction</td>
<td>D. V. McAuley</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Decision</td>
<td>E. J. Mckean</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Evaluation</td>
<td>F. I. McMillan</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Feature</td>
<td>G. W. McSherry</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Generation</td>
<td>H. V. McWhorter</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Generation</td>
<td>I. R. McWright</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Graph</td>
<td>J. S. McKee</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Hierarchy</td>
<td>K. N. McKee</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Hypothesis</td>
<td>L. P. McLeod</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Knowledge</td>
<td>M. A. McIver</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Model</td>
<td>N. M. Mclendon</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Methodology</td>
<td>O. R. McLeod</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Metalearning</td>
<td>P. S. McKean</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Prediction</td>
<td>Q. S. McIver</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Process</td>
<td>R. E. McKee</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Query</td>
<td>S. W. McIver</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Reasoning</td>
<td>T. J. McConahay</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Representation</td>
<td>U. J.</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Rules</td>
<td>V. L. McKee</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Structure</td>
<td>W. J. McKee</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Strategy</td>
<td>X. S. McKee</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Technique</td>
<td>Y. T. McKee</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Tool</td>
<td>Z. R. McKee</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
</tbody>
</table>
Of these...

23 use junction trees for inference (some use Jtrees in addition to other algos)

5 use gibbs sampling

Many seem to be defunct, unsupported, or abandoned…

Why are there so many of these?
“It’s hard to strike a balance between generality and usability.” -Prof. Ihler
Positive qualities of software libraries… (CISQ)

Reliable

Efficient

Secure

Maintainable

 Appropriately Scoped (Size)

“CISQ has defined five major desirable characteristics of a piece of software needed to provide business value…” (https://en.wikipedia.org/wiki/Software_quality)
The rest of this talk will focus on the libraries that can begin to convincingly claim to fulfill these qualities (in my opinion)
graphical models library

Scholarly articles for graphical models library
... for discrete approximate inference in graphical models - Mooij - Cited by 256
... A program for analysis of Bayesian graphical models - Horik - Cited by 2332
Learning in graphical models - Jordan - Cited by 1767

GitHub - pgmmpy/pgmmpy: Python Library for Probabilistic Graphical...
https://github.com/pgmmpy/pgmmpy
pgmmpy is a python library for working with Probabilistic Graphical Models. Documentation and list of algorithms supported is at our official site http://pgmmpy.org/
Pgmmpy - Pgmmpy/pgmmpy_notebook - Issues 155

GitHub - opengm/opengm: A C++ Library for Discrete Graphical Models
https://github.com/opengm/opengm
OpenGM is a C++ template library for discrete factor graph models and distributive operations on these models. It includes state-of-the-art optimization and inference algorithms beyond message passing,...
The graphical model data structure, inference algorithms and different encodings...

machine learning - Good libraries for working with probabilistic...
https://stats.stackexchange.com/.../good-libraries-for-working-with-probabilistic-graph...
Nov 7, 2013 - A relatively new (and rapidly evolving) python library is pgmmpy (Github Link).

Edward - Home
edwardlib.org/
Edward is a Python library for probabilistic modeling, inference, and criticism. Directed graphical models; Neural networks (via libraries such as tf.layers and...
“Python library for Probabilistic Graphical Models”

- Details are sparse, but it seems that this library has its origins as a Google Summer of Code project. There appear to be 4 major contributors: Ankur Ankan from Radboud University, Yashu Seth, Abinash Panda, Utkarsh Khalibartan, and an unnamed GitHub user contributing under the handle “vivek425ster”.
- Open source
- Version 0.1.2
- Still under development (last commit on April 11)
- MIT License
- 48 contributors
Models

Bayesian Model
Markov Model
Factor Graph
Cluster Graph
Junction Tree
Markov Chain
NoisyOr Model
Naive Bayes
DynamicBayesianNetwork
Sampling Methods

Gibbs Sampler
Bayesian Model Samplers
Hamiltonian Monte Carlo
No U-Turn Sampler
Algorithms

Variable Elimination
Belief Propagation
MPLP
Dynamic Bayesian Network Inference
Positives

Very approachable (well documented)

Actively supported (bug fixes, features added)

Python
Negatives

Not backed by Big 4 company

Development seems to be slowing down (fewer commits over time)
2nd half of talk will focus on examples of what you can do with pgmpy...
Generality

OpenGM2

Usability
“A C++ Library for Discrete Graphical Models”

- Developed at The Heidelberg Collaboratory for Image Processing at the University of Heidelberg. There are 3 main developers: Bjoern Andres, Thorsten Beier, and Joerg H. Kappes.
- Open source
- Version 2.0.2
- Still under development (last commit on April 5)
- MIT License
- 38 contributors
- Wrappers for Python and Matlab
Models

Graphs of any order and structure, from second order grid graphs to irregular higher-order models
Algorithms

- Combinatorial/Global Optimal Methods
- Linear Programming Relaxations
- Message Passing Methods
- Move Making Methods
- Sampling
- Wrapped External Code for Discrete Graphical Models

(41 total by my count)
Positives

Highly general

C++

Extensive Documentation
Negatives

Not backed by a Big 4 company

Highly general

C++
Edward
“Edward is a Python library for probabilistic modeling, inference, and criticism. It is a testbed for fast experimentation and research with probabilistic models, ranging from classical hierarchical models on small data sets to complex deep probabilistic models on large data sets. Edward fuses three fields: Bayesian statistics and machine learning, deep learning, and probabilistic programming.”

“Formally, Edward is a Turing-complete probabilistic programming language.”

- Developed at Columbia University. Primary Developer: Dustin Tran
- Open source
- Version 1.3.5
- Still under development (last commit on June 1)
- MIT License
- 77 contributors
An abstraction over tensorflow

Directed graphical models
Neural networks (via libraries such as tf.layers and Keras)
Implicit generative models
Bayesian nonparametrics and probabilistic programs
Inference with...

Variational inference
- Black box variational inference
- Stochastic variational inference
- Generative adversarial networks
- Maximum a posteriori estimation

Monte Carlo
- Gibbs sampling
- Hamiltonian Monte Carlo
- Stochastic gradient Langevin dynamics

Compositions of inference
- Expectation-Maximization
- Pseudo-marginal and ABC methods
- Message passing algorithms
“Samlam is a comprehensive tool for modeling and reasoning with Bayesian networks”

- Developed at University of California, Los Angeles by the Automated Reasoning Group of Professor Adnan Darwiche.

- Closed source
Kevin’s notes on Samlam

I took a look at this tool. It’s impressive in the sense that the UI is very well designed and the fact that it’s a Java program means that it can run on any machine with a Java virtual machine implementation, but the project is not open source. I can call into the code, but I can neither see nor edit the code. In my opinion, this is a serious issue. Why not host the code on Github? Also, it’s not clear what the licensing is for this software. Can I use it in an industrial/commercial application? All of these factors limit Samlam’s utility, unfortunately.
Installation...

pip install if you’re on linux

Easy, fast, basically error-proof
(As an aside…)

There’s an R package called bnlearn ([http://www.bnlearn.com/](http://www.bnlearn.com/))

If you go to [http://www.bnlearn.com/bnrepository/](http://www.bnlearn.com/bnrepository/) there are Bayesian networks (large and small) to test with!
<table>
<thead>
<tr>
<th>Name</th>
<th>Nodes</th>
<th>Arcs</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASIA</td>
<td>8</td>
<td>8</td>
<td>18</td>
</tr>
<tr>
<td>CANCER</td>
<td>5</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>EARTHQUAKE</td>
<td>5</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>SACHS</td>
<td>11</td>
<td>17</td>
<td>178</td>
</tr>
<tr>
<td>SURVEY</td>
<td>6</td>
<td>6</td>
<td>21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Nodes</th>
<th>Arcs</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALARM</td>
<td>37</td>
<td>46</td>
<td>509</td>
</tr>
<tr>
<td>BARLEY</td>
<td>48</td>
<td>84</td>
<td>114005</td>
</tr>
<tr>
<td>CHILD</td>
<td>20</td>
<td>25</td>
<td>230</td>
</tr>
<tr>
<td>INSURANCE</td>
<td>27</td>
<td>52</td>
<td>984</td>
</tr>
<tr>
<td>MILDEW</td>
<td>35</td>
<td>46</td>
<td>540150</td>
</tr>
<tr>
<td>WATER</td>
<td>32</td>
<td>66</td>
<td>10083</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Nodes</th>
<th>Arcs</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>HAILCINDER</td>
<td>56</td>
<td>66</td>
<td>2666</td>
</tr>
<tr>
<td>HEDAR II</td>
<td>70</td>
<td>123</td>
<td>1453</td>
</tr>
<tr>
<td>WINGSPTS</td>
<td>76</td>
<td>112</td>
<td>574</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Nodes</th>
<th>Arcs</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANDES</td>
<td>223</td>
<td>338</td>
<td>1157</td>
</tr>
<tr>
<td>DIABETES</td>
<td>413</td>
<td>602</td>
<td>429409</td>
</tr>
<tr>
<td>LINK</td>
<td>724</td>
<td>1125</td>
<td>14211</td>
</tr>
<tr>
<td>MUHIN (4 subnetworks)</td>
<td>108–1041</td>
<td>273–1386</td>
<td>15622–80352</td>
</tr>
<tr>
<td>PATHFINDER</td>
<td>135</td>
<td>200</td>
<td>77195</td>
</tr>
<tr>
<td>PIGS</td>
<td>441</td>
<td>592</td>
<td>5618</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Nodes</th>
<th>Arcs</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>MUHIN (full network)</td>
<td>1041</td>
<td>1397</td>
<td>80592</td>
</tr>
<tr>
<td>MUHIN (4 subnetworks)</td>
<td>108–1041</td>
<td>273–1386</td>
<td>15622–80352</td>
</tr>
</tbody>
</table>
(As another aside…)
```python
from matplotlib import rc
rc("font", family="serif", size=12)
rc("text", usetex=True)

import daft

g = daft.PGM([3.6, 2.7], origin=[1.15, 0.65])
g.add_node(daft.Node("Pollution", r"Pollution", 2, 3, aspect=2))
g.add_node(daft.Node("Smoker", r"Smoker", 4, 3, aspect=2))
g.add_node(daft.Node("Cancer", r"Cancer", 3, 2, aspect=2.1))
g.add_node(daft.Node("Xray", r"Xray", 2, 1, aspect=2.4))
g.add_node(daft.Node("Dyspnoea", r"Dyspnoea", 4, 1, aspect=2.4))
g.add_edge("Pollution", "Cancer")
g.add_edge("Smoker", "Cancer")
g.add_edge("Cancer", "Xray")
g.add_edge("Cancer", "Dyspnoea")
g.render()
g.figure.savefig("wordy.pdf")
g.figure.savefig("wordy.png", dpi=150)
```
Back to pgmpy...

```python
# Starting with defining the network structure
from pgmpy.models import BayesianModel

cancer_model = BayesianModel([('Pollution', 'Cancer'),
                              ('Smoker', 'Cancer'),
                              ('Cancer', 'Xray'),
                              ('Cancer', 'Dyspnoea')])
```
from pgmpy.factors.discrete import TabularCPD

cpd_poll = TabularCPD(variable='Pollution', variable_card=2,
                      values=[[0.9], [0.1]])

cpd_smoke = TabularCPD(variable='Smoker', variable_card=2,
                       values=[[0.3], [0.7]])

cpd_cancer = TabularCPD(variable='Cancer', variable_card=2,
                         values=[[0.03, 0.05, 0.001, 0.02],
                                 [0.97, 0.95, 0.999, 0.98]],
                         evidence=['Smoker', 'Pollution'],
                         evidence_card=[2, 2])

cpd_xray = TabularCPD(variable='Xray', variable_card=2,
                       values=[[0.9, 0.2], [0.1, 0.8]],
                       evidence=['Cancer'], evidence_card=[2])

cpd_dysp = TabularCPD(variable='Dyspnoea', variable_card=2,
                       values=[[0.65, 0.3], [0.35, 0.7]],
                       evidence=['Cancer'], evidence_card=[2])
# Associating the parameters with the model structure.
cancer_model.add_cpds(cpd_poll, cpd_smoke, cpd_cancer, cpd_xray, cpd_dysp)

# Checking if the cpds are valid for the model.
cancer_model.check_model()

assert(cancer_model.is_active_trail('Pollution', 'Smoker') == False)
assert(cancer_model.is_active_trail('Pollution', 'Smoker', observed=['Cancer']) == True)

print(cancer_model.get_independencies())
print(cancer_model.get_independencies())
# Doing exact inference using Variable Elimination

```python
from pgmpy.inference import VariableElimination

# cancer_model is assumed to be defined

# Create an instance of VariableElimination

# Define the query and evidence

q = cancer_infer.query(variables=['Cancer'], evidence={'Pollution': 0, 'Smoker': 0})

# Print the result

print(q['Cancer'])
```

<table>
<thead>
<tr>
<th>Cancer</th>
<th>phi(Cancer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cancer_0</td>
<td>0.0300</td>
</tr>
<tr>
<td>Cancer_1</td>
<td>0.9700</td>
</tr>
</tbody>
</table>

```python
print(cancer_infer.induced_width(['Pollution', 'Smoker', 'Cancer', 'Xray', 'Dyspnoea']))
```
I hope this was helpful, interesting, or provided some ideas about potential future work.

Thank you!

Questions?