© Copyright, Wray Buntine, 1995

Compiling Probabilistic Networks and Some Questions This Poses

Wray Buntine
Thinkbank
wray@Thinkbank.com

Presented at NIPS'95 Workshop: Learning in Bayesian Networks and Other Graphical Models. December, 2, 1995.

© Copyright, Wray Buntine, 1995

Probabilistic Networks

- **networks with a probabilistic semantics**
- **including:**
  - mixed real and discrete variables
  - missing and/or hidden values
  - deterministic nodes
  - arbitrary (but specified) conditional probabilities
  - directed/undirected arcs and different semantics

© Copyright, Wray Buntine, 1995

Outline

- **general framework: power-tools for data analysis**
- **compiling probability networks: an introduction**
- **arguments in favor of compilation and some lessons from (networks) prehistory**
- **some questions I'd like to see answered**

© Copyright, Wray Buntine, 1995

Power-tools for data analysis

Power-tools

e.g., GUI builders, lexical analysers, Matlab, etc.

- let you custom design software efficiently
- improves turn-around from concept to software;

Data analysis requires power-tools:

- there is no universal learning algorithm
- problem specific manipulation and structuring needed

i.e. need ways to understand and incorporate complex priors

- many general purpose algorithms exist that are applicable to specific contexts, and the space of possibilities has hardly been touched by researchers

Data analysis requires software engineering:

- need techniques to assist rapid prototype-refinement cycle

Quality power-tools already exist for other tasks:

- for data manipulation, visualization, etc.
- Matlab, S-Plus, PV-Wave, etc.

© Copyright, Wray Buntine, 1995

Programming with Probabilities:

(for prototyping learning algorithms by analysts/experts (NOT by users))

Probability networks: chain graphs, which model many network languages in neural networks and statistics

+

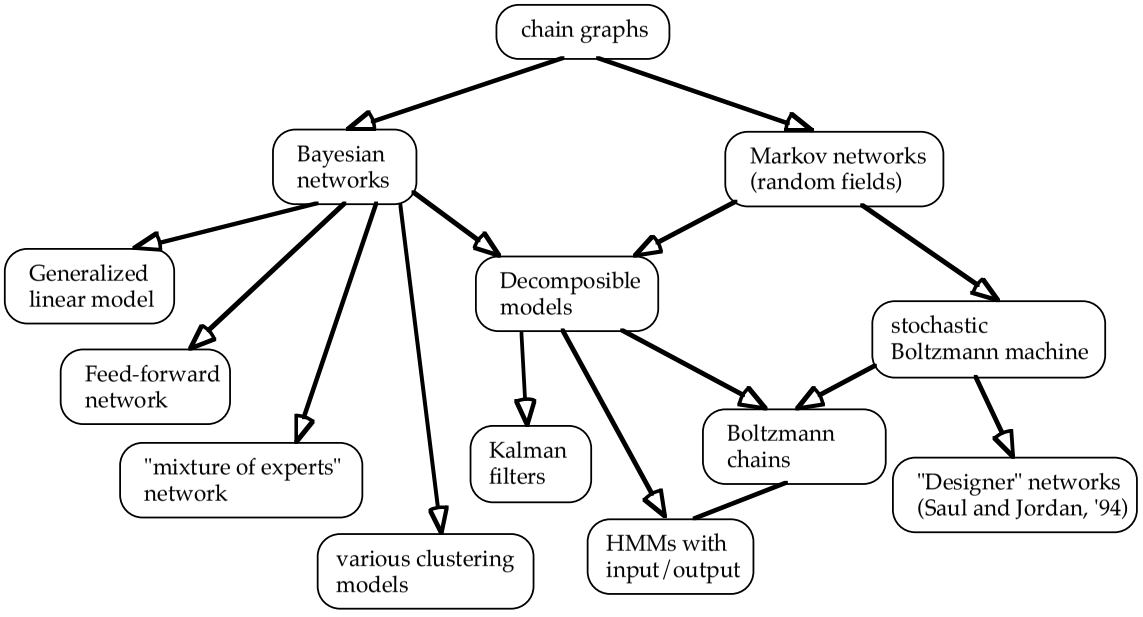
Operations on networks: decomposition, differentiation, maximization, MCMC sampling, optimizing compilation, etc.

+

Software environment: a flexible software environment to embed this in other systems.

© Copyright, Wray Buntine, 1995

Probability networks



A tangled hierarchy showing some different kinds of networks and their relationships.

NB Inference schemes for these network types have been developed in statistics, physics, artificial intelligence, and neural networks

© Copyright, Wray Buntine, 1995

Some algorithms of form: network + standard-operations

- Bayesian networks including missing/latent discrete/Gaussian variables, and so-called structure learning.
- Generalized linear models with variable subset selection, smoothing, etc. e.g., linear and logistic regression.
- Financial time-series models such as GARCH, and many variations, and the jump-diffusion model of volatility.
- Many variations of feed-forward networks and mixtures of experts, plus networks predicting error bar.
- Markov networks, stochastic Bolzmann networks, including some fast optimizations (e.g., "chains")
- Unsupervised learning including "multiple causes", hybrid Bayesian networks, correlation models, etc., principle components and various hybrids.
- Hidden Markov models, Kalman filters, etc., including new versions such as input/output HMMs.
- Knowledge refinement of the above, i.e. using subjective knowledge to "initialize" learning

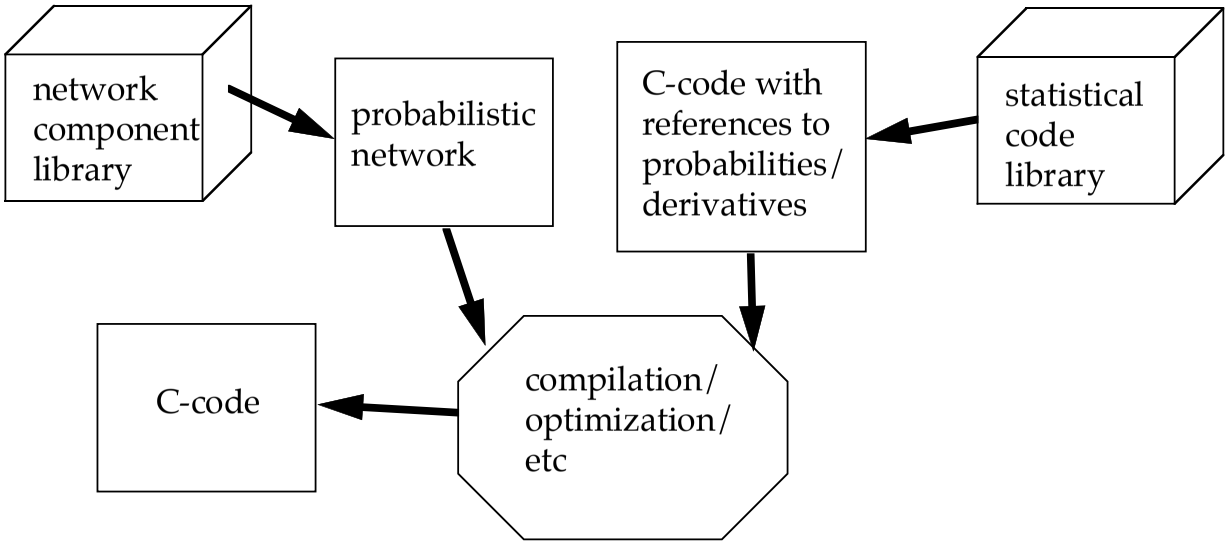
© Copyright, Wray Buntine, 1995

Outline

- **general framework: power-tools for data analysis**
- **compiling probability networks: an introduction**
- **arguments in favor of compilation and some lessons from (networks) prehistory**
- **some questions I'd like to see answered**

© Copyright, Wray Buntine, 1995

Compilers for data analysis



- could be implemented as a suite of C++ classes, an interpreter, or pre-processor to a standard compiler for C/C++

© Copyright, Wray Buntine, 1995

Short History of Compilers

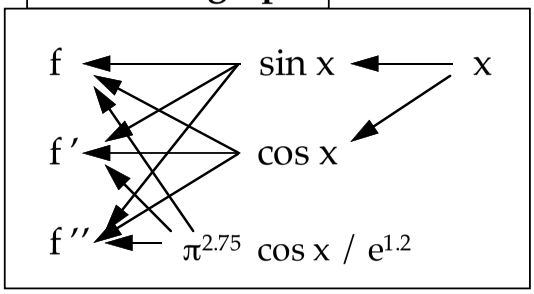
- **Bayesian analysis Using Gibbs Sampling, (BUGS)**
  - from Gilks, Spiegelhalter, and Thomas (MRC, Cambridge, UK), '91-present
  - takes a data analysis problem represented as a Bayesian Network with Plates and compiles a Gibbs sampler for the problem
  - ftp://ftp.mrc-bsu.cam.ac.uk/pub/methodology/bugs
- **Data flow graphs for compiling/interpreting probabilistic networks**
  - from implementation of Scott Roy's PhD thesis at Stanford, July, '95
  - – able to implement Bayesian Nearest Neighbor, Bayesian version of LVQ, Autoclass with correlation, variant of Jordan and Jacob's mixture networks, etc.
  - http://logic.stanford.edu/papers/
- **Symbolic/code-level differentiation of networks**
  - use of standard optimization routines can require exact derivatives when difference approximations are poor (Buntine & Weigend, '94)
- **Optimizing compiler techniques for exact inference on Bayes networks**
  - Hansson and Sharp (unpublished paper, 1994)
  - apply optimizing compiler techiques to compile algorithms from basic probabilistic expressions (not unlike D'Ambrosio's SPI, but at compile time)

© Copyright, Wray Buntine, 1995

Roy's data flow approach

Example: find zeroes of $f(x) = e^{\sin x} - \pi^{2.75} \cos x / e^{1.2}$

data flow graph



procedure

repeat
$x' = x - f'/f''$
$e = |x' - x|$
$x = x'$
until $e < 0.0001$

- data flow graph gives key variables and how they are to be computed
- **compiler/interpreter** merges data flow graph and procedure to generate efficient code,
  e.g. when computing f'/f'', this insures that (sin x) will only be computed once; and the constant is computed outside the loop
- this approach can be applied to probabilistic inference,
  e.g., to instantiate Jensen's algorithm for inference on Bayesian networks for a particular query, rather like D'Ambrosio's SPI

© Copyright, Wray Buntine, 1995

Outline

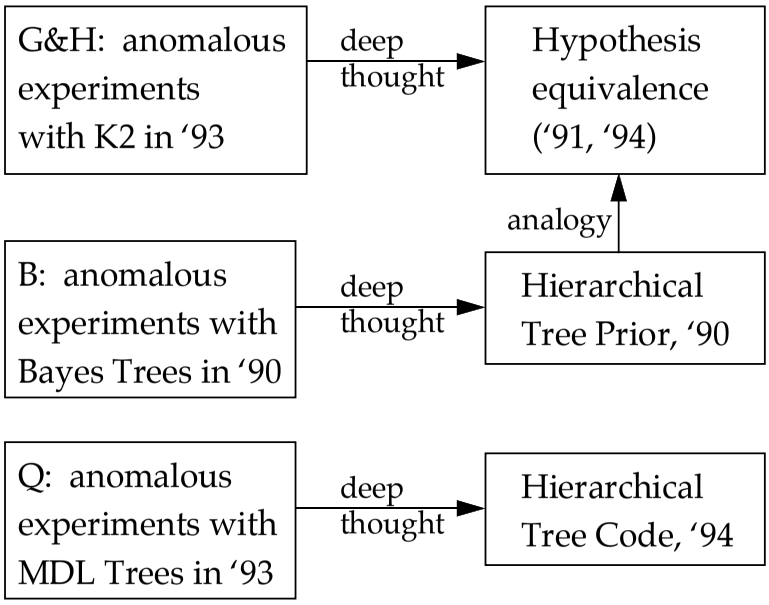
- **general framework: power-tools for data analysis**
- **compiling probability networks: an introduction**
- **arguments in favor of compilation and some lessons from (networks) prehistory**
- **some questions I'd like to see answered**

© Copyright, Wray Buntine, 1995

Discovering "Hypthesis Equivalence" for Bayesian networks

(see Geiger & Heckerman, UAI95)

- Geiger & Heckerman's discovery in '93-94 (see also Ayers '94)
- Buntine's discovery in '89-'91 (UAI'91)
- Quinlan's related discovery in '93 (ML'94)



LESSON: theoretical work in priors can require experimental investigation

© Copyright, Wray Buntine, 1995

What's a good tree system to implement my idea X in ?

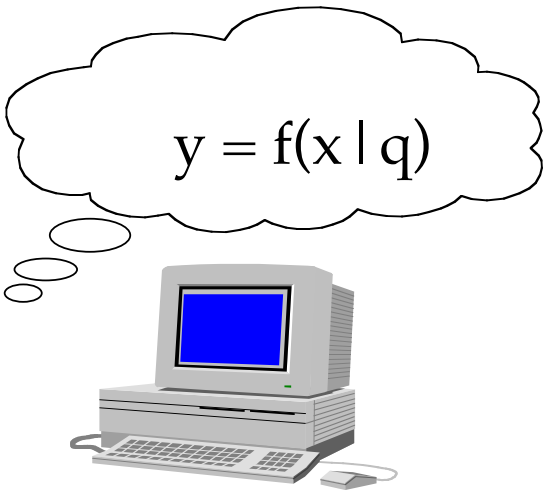
Consider any widely used classification tree package: Quinlan's, Buntine's, Fayyad's, Cestnik *et al.*'s, Breiman, Friedman *et al.*'s, etc.

- most authors receive repeated requests for help from people wanting to try out some new trick
  - with my IND at least, they often get turned down because my code is "a hacked up research prototype that grew like topsy"
- most packages have unique capabilities
- most only implement a small fraction of what the investigator know's how to do
- new researchers often reimplement their own system from scratch, and waste a large part of their resources doing so
- questions considered open to many may be "easy" to others – yet they can't demonstrate a solution because the implementation is too difficult in the above patchwork

LESSON: we need standards, shared implementation platforms, etc.

© Copyright, Wray Buntine, 1995

Data Analysis Today



3 minutes ...

... 3 months later

i.e., I turn away prospective customers because the effort required to do a prototype for them is not within the budget.

LESSON: we need better prototyping tools for learning algorithms

© Copyright, Wray Buntine, 1995

Many algorithms are novel combinations of a few simple tricks

(and we've hardly scratched the surface)

- APNs (Russell, Koller and Binder) is gradient descent (sometimes online) of an exponential family mixture
  - variation 1: use on-line gradient descent
  - variation 2: use sub-sampled version of EM
  - variation 3: hybrid combination of the two
- Multi-class (factorial) clustering (various, Ghahramani):
  - variation 1: EM and (slow) exact calculation of E-step
  - variation 2: EM and Gibb's approx. to E-step
  - variation 3: EM and mean field approx. to E-step

LESSON: we need faster ways to explore the space of possible algorithms

© Copyright, Wray Buntine, 1995

Outline

- **general framework: power-tools for data analysis**
- **compiling probability networks: an introduction**
- **arguments in favor of compilation and some lessons from (networks) prehistory**
- **some questions I'd like to see answered**

© Copyright, Wray Buntine, 1995

Some research questions to help in compilation

- For a given probabilistic network:
  - what's the sample complexity (for bounds on learning to determine when priors should be irrelevant)?
  - what's an approximation to the evidence/Bayes-factor?
  - what's a good approx. for the network, and how is it computed?
- What's the best way to control on-line vs. sub-sampling vs. batch versions of gradient descent for general cost function
- How should hybrids of gradient-descent/EM/Gibbs/k-means/etc. be applied to arbitrary missing/hidden values?

© Copyright, Wray Buntine, 1995

More questions

- Bayesian network compilation: given
  - Bayesian network on discrete variables
  - set of probability queries and evidence
  - procedural code giving potential order of queries

  compile code for evaluation, possibly into data flow graph

  NB. matrix multiplication is a special case
- Take your favorite/new MCMC algorithm:
  - how do we automatically instantiate it for an arbitrary probabilistic network?