Software for Generalized Bayesian Inference

An object-oriented $\text{R}$ implementation of generalized iLUCK-models

Gero Walter

Department of Statistics
Ludwig-Maximilians-University Munich

gero.walter@stat.uni-muenchen.de
www.stat.uni-muenchen.de/~walter

February 17th, 2009
Generalized Bayesian Inference – General Idea

Bayesian Inference on some parameter $\theta$:

- prior knowledge on $\theta$ + data $x$ $\rightarrow$ updated knowledge on $\theta$
- prior distribution $p(\theta)$ + likelihood $f(x \mid \theta)$ $\rightarrow$ posterior distribution $p(\theta \mid x)$

**set of** priors + likelihood $\rightarrow$ **set of** posteriors

Tractability: use **conjugate** priors, i.e.
choose $p(\theta)$ such that $p(\theta \mid x)$ is from the same parametric class
$\rightarrow$ update only parameters!
LUCK-models: Single Conjugate Prior

\( X \overset{iid}{\sim} \text{linear, canonical exponential family}, \) i.e.

\[
p(x \mid \theta) \propto \exp \left\{ \langle \psi, \tau(x) \rangle - nb(\psi) \right\} \quad \left[ \psi \text{ transformation of } \theta \right]
\]

\( \rightarrow \) conjugate prior:

\[
p(\theta) \propto \exp \left\{ n^{(0)} \left[ \langle \psi, y^{(0)} \rangle - b(\psi) \right] \right\}
\]

\( \rightarrow \) (conjugate) posterior:

\[
p(\theta \mid x) \propto \exp \left\{ n^{(1)} \left[ \langle \psi, y^{(1)} \rangle - b(\psi) \right] \right\}
\]

\[
y^{(1)} = \frac{n^{(0)}y^{(0)} + \tau(x)}{n^{(0)} + n} \quad \text{and} \quad n^{(1)} = n^{(0)} + n.
\]
Interpretation of $y^{(0)}$ and $n^{(0)}$

$y^{(0)}$: “main prior parameter”

- for samples from a $\mathcal{N}(\mu, 1)$, $p(\mu)$ is a $\mathcal{N}(y^{(0)}, \frac{1}{n^{(0)}})$
- for samples from a $\mathcal{M}(\theta)$, $p(\theta)$ is a $\text{Dir}(n^{(0)}, y^{(0)})$

$(y^{(0)}_j = t_j \equiv \text{prior probability for class } j, \ n^{(0)} = s)$

$n^{(0)}$: “prior strength” or “pseudocounts”

with $\tilde{\tau}(x) =: \frac{1}{n}\tau(x)$: $\tau(x) = \sum_{i=1}^{n} \tau(x_i)$

\[
y^{(1)} = \frac{n^{(0)}}{n^{(0)} + n} \cdot y^{(0)} + \frac{n}{n^{(0)} + n} \cdot \tilde{\tau}(x).
\]
sets of LUCK-models – iLUCK-models

iLUCK-model:

vary $y^{(0)}$ in $\mathcal{Y}^{(0)}$ [ $\mathcal{Y}^{(0)}$ convex ],
i.e. allow for ambiguity on the main prior parameter

→ prior credal set contains all finite convex mixtures of $p(\theta)$s with $y^{(0)} \in \mathcal{Y}^{(0)}$

→ posterior credal set easy to calculate:
all finite convex mixtures of $p(\theta \mid x)$s with

$$y^{(1)} \in \mathcal{Y}^{(1)} = \frac{n^{(0)}}{n^{(0)} + n} \cdot \mathcal{Y}^{(0)} + \frac{n}{n^{(0)} + n} \cdot \tilde{\tau}(x)$$

⚠️ unfavourable behavior in case of prior–data conflict! ⚠️
sets of LUCK-models – Generalized iLUCK-models

generalized iLUCK-model:

vary $y^{(0)}$ in $\mathcal{Y}^{(0)}$ and $n^{(0)}$ in $\mathcal{N}^{(0)}$, i.e. weigh prior information $\mathcal{Y}^{(0)}$ and sample information $\tilde{\tau}(x)$ more flexible in 

$$y^{(1)} = \frac{n^{(0)}}{n^{(0)} + n} \cdot y^{(0)} + \frac{n}{n^{(0)} + n} \cdot \tilde{\tau}(x)$$

$\rightarrow$ prior credal set contains all finite convex mixtures of $p(\theta)$s with $y^{(0)} \in \mathcal{Y}^{(0)}$ and $n^{(0)} \in \mathcal{N}^{(0)}$

$\rightarrow$ posterior credal set still quite easy to calculate: all finite convex mixtures of $p(\theta \mid x)$s with

$$\left\{ (n^{(1)}, y^{(1)}) \mid n^{(1)} = n^{(0)} + n, \ y^{(1)} = \frac{n^{(0)} y^{(0)} + \tau(x)}{n^{(0)} + n}, \ n^{(0)} \in \mathcal{N}^{(0)}, \ y^{(0)} \in \mathcal{Y}^{(0)} \right\}$$
The \texttt{R} project for Statistical Computing

- not just a (statistical) software package, rather a full-grown programming language
- open source implementation of the (award-winning) \texttt{S} language
- extremely widespread in university research (reference implementation of new methods are often in \texttt{R})
- extensions providing additional functionality can be made readily available as “packages”
- can be linked with \LaTeX{} (package \texttt{Sweave})
- can be used as imperative or as object-oriented language
Imperative vs. Object-oriented Programming

**imperative:** do this, then that
- functions (on arguments)

**object-oriented:** create ‘objects’, do things with them
- blueprints for objects called ‘classes’

Objects created according to a blueprint are called an ‘instance’

Example:
Banking company administrating their customers’ accounts

- **class:** BankAccount
- **instances:** bank account for customer A
  bank account for customer B
Object-oriented Programming: Class hierarchies

Some Class
- ‘slots’ (attributes)
- ‘methods’ (functions)

Subclass
- additional/specialized slots
- additional/specialized methods

Subsubclass 1
- add./spec. slots
- add./spec. methods

Subsubclass 2
- add./spec. slots
- add./spec. methods

Subsubclass 3
- add./spec. slots
- add./spec. methods

extends Some Class
inherits slots & methods
Implementation – Class Structure

LuckModel
- n0: matrix
- y0: matrix
- data: LuckModelData
- show()
- plot()
- unionHdi()

ScaledNormalLuckModel
- singleHdi()

PoissonLuckModel
- singleHdi()

...

