Doing Statistics Blindfold

Louis J. M. Aslett (aslett@stats.ox.ac.uk), Pedro M. Esperança, and Chris C. Holmes
Department of Statistics, University of Oxford

i-like.org.uk

UNIVERSITY OF OXFORD

1. Privacy in statistics

The extensive use of private and personally identifiable information in modern statistical applications, especially in biomedical applications, can present serious privacy concerns.

Indeed, industry is on the brink on embarking on biomedical applications on a scale never before witnessed via the impending wave of so-called 'wearable devices' such as smart watches, which can monitor vital health signs round the clock, perhaps fitting classification models to alert on different health conditions. Such constrained devices will almost certainly leverage cloud services, uploading reams of private health diagnostics to corporate servers.

Is there any hope of honouring people's desire for privacy while still performing statistical analyses?

2. Homomorphic encryption : the blindfold

Traditional encryption schemes (AES, SSL, ...) secure data for archive or communication using a key k or keypair (k_p, k_s). Encrypt a *message*, $m \in M$, to a *ciphertext*, $c \in C$, with public key:

$$c \leftarrow \texttt{Enc}(k_p, m)$$

Decrypt with secret key:

$$m = \texttt{Dec}(k_s, c)$$

But if we want to compute, have to decrypt first because they are 'brittle':

$$\texttt{Dec}(k_s, f(c)) \neq f(m) \quad \forall\, f(\cdot) \neq \texttt{Id}(\cdot)$$

Homomorphic encryption

Rivest *et al.* (1978) hypothesised $\exists$ schemes allowing blindfold computation. Not until Gentry (2009) was it shown to be possible for arbitrary numbers of additions & multiplications. Homomorphic if:

$$\texttt{Dec}(k_s, \texttt{Enc}(k_p, m_1) \diamond \texttt{Enc}(k_p, m_2)) = m_1 \circ m_2$$

for a set of operations $\circ \in \mathcal{F}_M$ acting in M that have corresponding operations $\diamond \in \mathcal{F}_C$ acting in C. "Fully homomorphic" $\implies \mathcal{F}_M = \{+, \times\}$.

Fully homomorphic exciting if $M = \mathbb{Z}/2\mathbb{Z}$ because $+ \equiv \veebar$ and $\times \equiv \wedge$, so can reproduce arbitrary boolean logic (arbitrary computation).

Nirvana? Perhaps pergatory ...

Flurry of excitement, followed by dose of reality.

- C usually complex (e.g. polynomial ring)
  - very slow computation
  - size of $c \gg$ size of m
- $\therefore$ $M = \mathbb{Z}/2\mathbb{Z}$ impractical, but
  - $M = \mathbb{R}$ impossible
  - $M = \mathbb{Z}/n\mathbb{Z}$ for large n best
- ... but if integers not boolean circuits we'd like
  - division
  - comparisons $(<, \leq, >, \geq, =)$

  which are not possible!

Quick reality check: can just evaluate polynomials of integers (in practise of limited degree).

The challenge: fit meaningful statistical models within these constraints.

3. Fan and Vercauteren (2012)

Notation

$\mathbb{Z}_q = \{n : n \in \mathbb{Z}, -q/2 < n \leq q/2\}$
$[a]_q \in \mathbb{Z}_q$ st $[a]_q = a \mod q$
$\mathbb{Z}[x], \mathbb{Z}_q[x]$ polynomials with coeff $\in \mathbb{Z}$ and $\in \mathbb{Z}_q$
$\Phi_n(x)$ n^{th} cyclotomic poly, $\Phi_{2^d}(x) = x^{2^{d-1}} + 1$
$R = \mathbb{Z}[x]/\Phi_{2^d}(x)$ and $R_q = \mathbb{Z}_q[x]/\Phi_{2^d}(x)$
$a(x) = \underline{a} \in R_q$ polynomial ring elements
$[\underline{a}]_q \implies$ centred reduction of coeff in $\mathbb{Z}_q$
$\cdot \sim \chi$ random poly with discrete Gaussian coeff
$\cdot \sim R_q$ random poly uniformly from R_q

Parameters

- d, degree of polynomial rings M and C;
- t, q, magnitude of coefficient sets of M, C;
- σ, magnitude of injected noise.

Encryption scheme

$M = R_t, C = R_q \times R_q$

Keys: $\underline{k}_s \sim R_2$ and $k_p := ([-(\underline{a} \cdot \underline{k}_s + \underline{e})]_q, \underline{a})$
where $\underline{a} \sim R_q$ and $\underline{e} \sim \chi$.

Encrypt: first map $m \in \mathbb{Z} \to \mathring{m}(x) \in R_t$

$$m = \textstyle\sum_n a_n 2^n \to \mathring{m}(x) = \sum_{n=0}^{2^{d-1}-1} a_n x^n \in R_t$$
$$c := ([\underline{k}_{p1} \cdot \underline{u} + \underline{e}_1 + \Delta \cdot \underline{\mathring{m}}]_q, [\underline{k}_{p2} \cdot \underline{u} + \underline{e}_2]_q)$$

where $\underline{u}, \underline{e}_1, \underline{e}_2 \sim \chi$ and $\Delta = \left\lfloor \frac{q}{t} \right\rfloor$.

Add/mult: $c_1 + c_2 = ([\underline{c}_{11} + \underline{c}_{21}]_q, [\underline{c}_{12} + \underline{c}_{22}]_q)$

$$c_1 \times c_2 = \left(\left[\left\lfloor \frac{t(\underline{c}_{11} \cdot \underline{c}_{21})}{q} \right\rceil\right]_q, \left[\left\lfloor \frac{t(\underline{c}_{11} \cdot \underline{c}_{22} + \underline{c}_{12} \cdot \underline{c}_{21})}{q} \right\rceil\right]_q, \left[\left\lfloor \frac{t(\underline{c}_{12} \cdot \underline{c}_{22})}{q} \right\rceil\right]_q \right)$$

Decrypt c: $\underline{\mathring{m}} = \left[\left\lfloor \frac{t[\underline{c}_1 + \underline{c}_2 \cdot \underline{k}_s]_q}{q} \right\rceil\right]_t$

Then $m = \mathring{m}(2)$.

4. High performance R package

`HomomorphicEncryption` R package (Aslett, 2014) provides easy to use interface which hides all the complexity of homomorphic encryption.

Implementation is mostly high performance C++, with many operations setup to utilise multi-core parallelism without any end-user intervention.

Native support for vectors/matrices and all operators and common functions overloaded to run encrypted.

```
p <- parsHelp("FandV", lambda=80, L=8)
k <- keygen(p)
c <- enc(k$pk, matrix(1:9, nrow=3))
cres <- c[,1] %*% c
dec(k$sk, cres)
```

5. Secret Forests (SF)

Data representation

Create purely binary representation of categorical/continuous variables (e.g. quintile binning).

$$\begin{pmatrix} 1 & 1.7 \\ 2 & 1.9 \\ 3 & 1.6 \\ \vdots & \vdots \\ x_{i2} & x_{i3} \end{pmatrix} \to \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ \vdots & & & & \vdots & & & \\ \tilde{x}_{i21} & \tilde{x}_{i22} & \tilde{x}_{i23} & \tilde{x}_{i31} & \tilde{x}_{i32} & \tilde{x}_{i33} & \tilde{x}_{i34} & \tilde{x}_{i35} \end{pmatrix}$$

Note: now have partition of variable j, $\mathcal{K}_j$ st:

$$\sum_{k \in \mathcal{K}_j} \tilde{x}_{ijk}\, \tilde{x}_{ljk} = 1 \iff x_{ij} = x_{lj}$$

with equality in binned/partition sense.

Secret Forest algorithm

For each $t \in \{1, \ldots, T\}$, build a tree in forest:

i) Tree growth: For each $l \in \{1, \ldots, L\}$, build a level. Level l has 2^{l-1} branches. For $b \in \{1, \ldots, 2^{l-1}\}$, construct partitions:

- **Splitting variable:** Select variable p_{tlb} at random from P predictors. Due to the representation, this variable has a partition $\mathcal{K}_{p_{tlb}}$.
- **Split point:** Create partition of $\mathcal{K}_{p_{tlb}}$, $\mathcal{D}_{tlb} = \{D_1^{tlb}, D_2^{tlb}\}$ where $D_j^{tlb} = \bigcup k_i$ for some $k_i \in \mathcal{K}_{p_{tlb}}$, with $D_1^{tlb} \cap D_2^{tlb} = \varnothing$ and $D_1^{tlb} \cup D_2^{tlb} = \bigcup_{\forall\, i} k_i$.

ii) Tree fitting: Total # training obs from category c in randomly grown tree t at leaf $b \in \{1, \ldots, 2^L\}$ is:

$$\rho_{bc}^t = \sum_{i=1}^N \tilde{y}_{ic} \prod_{l=1}^L \left(\sum_{k \in D_{h(b,l)}^{tlg(b,l)}} \tilde{x}_{i,\, p_{tlg(b,l)},\, k} \right)$$

where

$$g(b,l) := \left\lceil \frac{b}{2^{L+1-l}} \right\rceil$$
$$h(b,l) := \left\lfloor \frac{(b-1) \mod 2^{L+1-l}}{2^{L-l}} \right\rfloor + 1$$

iii) Prediction: Given encrypted test obs $\tilde{x}_{jk}^\star$, then:

$$\hat{y}_c^\star = \sum_{t=1}^T \sum_{b=1}^{2^L} \rho_{bc}^t \prod_{l=1}^L \left(\sum_{k \in D_{h(b,l)}^{tlg(b,l)}} \tilde{x}_{p_{tlg(b,l)},\, k}^\star \right)$$

is # of votes for response category c.

Further points too big to fit on poster ...

- Forest calibration
- Natural variance reduction
- Encrypted unbiased stochastic fraction estimate
- Incremental/parallel computation

6. Results

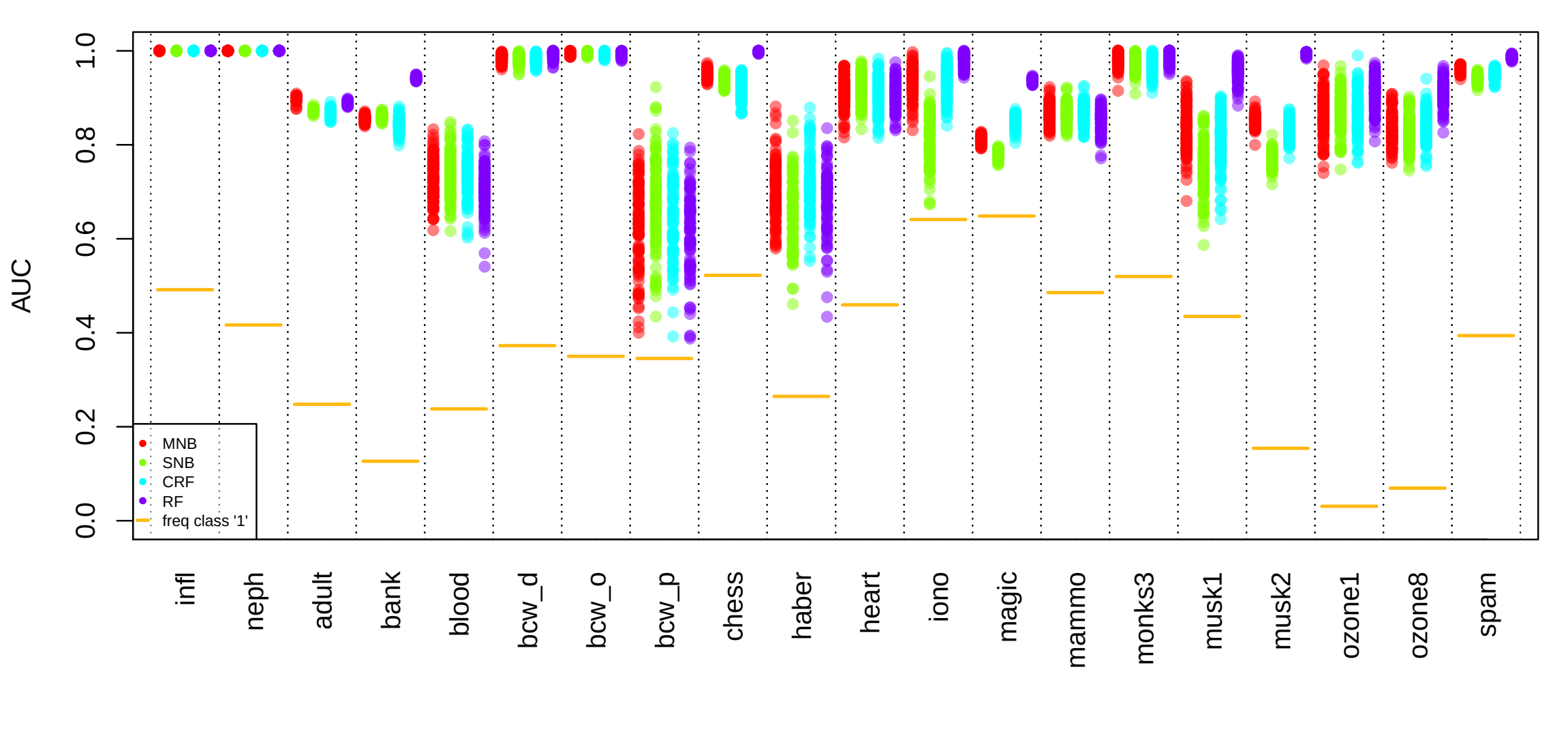
Other new crypto methods (see tech report soon)

- Semi-parametric naïve Bayes with linear logistic decision boundaries (SNB)
- One-step logistic regression (`LR-onestep`)

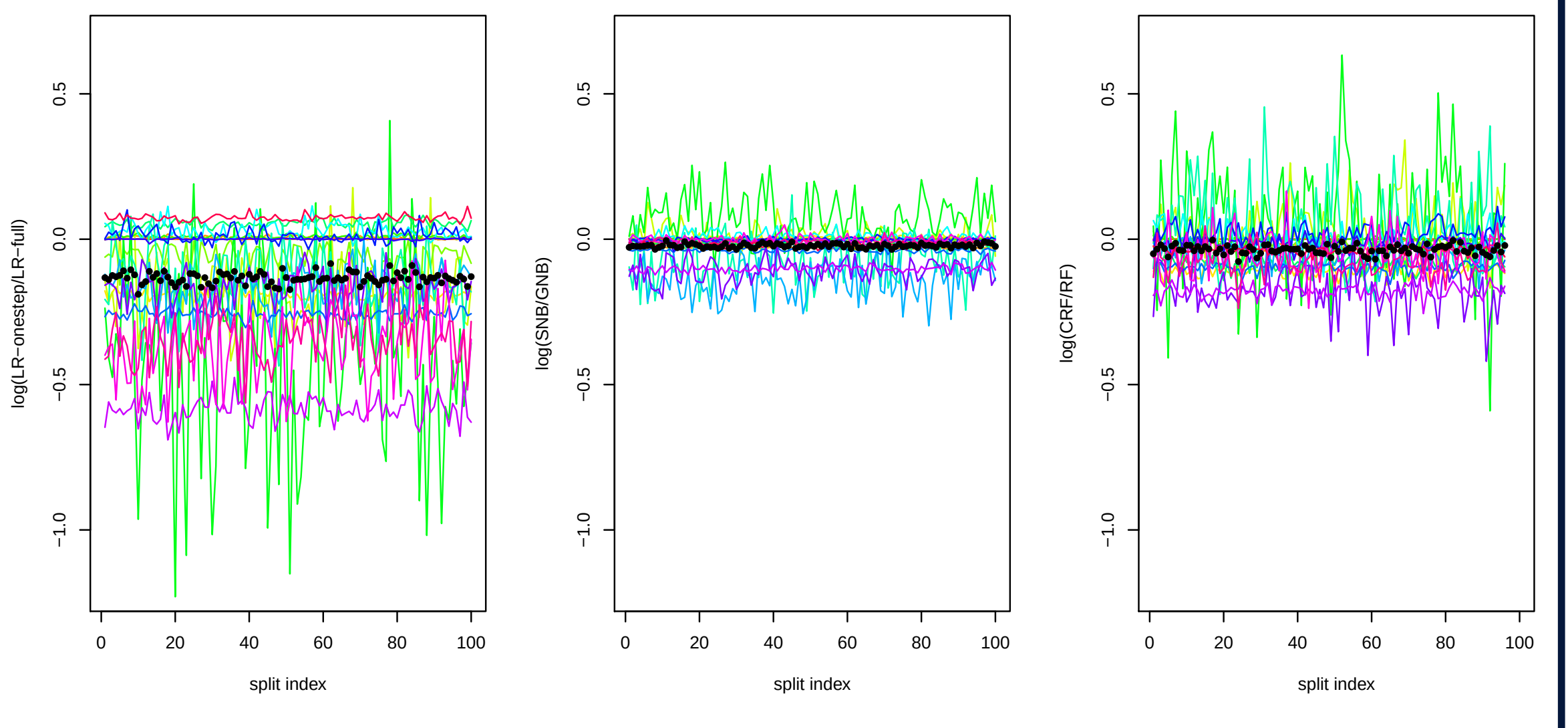
Tested on 20 different data sets from UCI repository.

Performance practicalities?

Full bcw_o data set, 100 tree SF with 3 levels deep fitted on Amazon EC2 cluster of 1152 CPU cores in 1 hour 36 minutes fully encrypted.
Total cost: less than US$ 24.



AUC for 100 randomisations of the train/test sets.



Ratio of encrypted method to traditional method AUC

References

Aslett, L. J. M. (2014), *HomomorphicEncryption: Fully Homomorphic Encryption*. R package version 0.1. `www.louisaslett.com/HomomorphicEncryption/`.
Fan, J. and Vercauteren, F. (2012), `Somewhat practical fully homomorphic encryption', *IACR Cryptology ePrint Archive* .
Gentry, C. (2009), A fully homomorphic encryption scheme, PhD thesis, Stanford University.
Rivest, R. L., Adleman, L. and Dertouzos, M. L. (1978), `On data banks and privacy homomorphisms', *Foundations of Secure Computation* **4**(11), 169--180.

Funding

EPSRC programme grant EP/K014463/1
`www.i-like.org.uk`