bbl: Boltzmann Bayes Learner for High-Dimensional Inference with Discrete Predictors in R

Jun Woo
University of Minnesota, Minneapolis

Jinhua Wang
University of Minnesota, Minneapolis

Abstract

Non-regression-based inferences, such as discriminant analysis, can account for the effect of predictor distributions that may be significant in big data modeling. We describe **bbl**, an R package for Boltzmann Bayes learning, which enables a comprehensive supervised learning of the association between a large number of categorical predictors and multi-level response variables. Its basic underlying statistical model is a collection of (fully visible) Boltzmann machines inferred for each distinct response level. The algorithm reduces to the naive Bayes learner when interaction is ignored. We illustrate example use cases for various scenarios, ranging from modeling of a relatively small set of factors with heterogeneous levels to those with hundreds or more predictors with uniform levels such as image or genomic data. We show how **bbl** explicitly quantifies the extra power provided by interactions via higher predictive performance of the model. In comparison to deep learning-based methods such as restricted Boltzmann machines, **bbl**-trained models can be interpreted directly via their bias and interaction parameters.

Keywords: Supervised learning, Boltzmann machine, naive Bayes, discriminant analysis, R.

1. Introduction

Many supervised learning tasks involve modeling discrete response variables y using predictors $\mathbf{x}$ that can occupy categorical factor levels (Hastie, Tibshirani, and Friedman 2009). Ideally, it would be best to model the joint distribution $P(\mathbf{x}, y)$ via maximum likelihood,

$$\hat{\Theta} = \arg\max_{\Theta} \left[\ln P(\mathbf{x}, y|\Theta)\right], \tag{1}$$

to find parameters Θ. Regression-based methods use $P(\mathbf{x}, y) = P(y|\mathbf{x})P(\mathbf{x}) \approx P(y|\mathbf{x})$. Many rigorous formal results known for regression coefficients facilitate interpretation of their significance. An alternative is to use $P(\mathbf{x}, y) = P(\mathbf{x}|y)P(y)$ and fit $P(\mathbf{x}|y)$. Since y is low-dimensional, this approach could capture extra information not accessible from regression when there are many covarying predictors. To make predictions for y using $P(\mathbf{x}|y)$, one uses the Bayes' formula. Examples include linear and quadratic discriminant analyses (Hastie *et al.* 2009, pp. 106-119) for continuous $\mathbf{x}$. For discrete $\mathbf{x}$, naive Bayes is the simplest approach, where the covariance among $\mathbf{x}$ is ignored via

$$P(\mathbf{x}|y) \approx \prod_i P(x_i|y) \tag{2}$$

with $\mathbf{x} = (x_1, \cdots, x_m)$.

In this paper, we focus on supervised learners taking into account the high-dimensional nature of $P(\mathbf{x}|y)$ beyond the naive Bayes-level description given by Eq. (2). Namely, a suitable parametrization is provided by the Boltzmann machine (Ackley, Hinton, and Sejnowski 1985), which for the simple binary predictor $x_i = 0, 1$,

$$P(\mathbf{x}|y) = \frac{1}{Z_y} \exp\left(\sum_i h_i^{(y)} x_i + \sum_{i<j} J_{ij}^{(y)} x_i x_j\right), \tag{3}$$

where Z_y is the normalization constant, or partition function. Equation (3) is the Gibbs distribution for Ising-type models in statistical mechanics (Chandler 1987). The two sets of parameters $h_i^{(y)}$ and $J_{ij}^{(y)}$ each represent single variable and two-point interaction effects, respectively. When the latter vanishes, the model leads to the naive Bayes classifier. Although exact inference of Eq. (3) from data is in general not possible, recent developments led to many accurate and practically usable approximation schemes (Hyvärinen 2006; Morcos, Pagnani, Lunt, Bertolino, Marks, Sander, Zecchina, Onuchic, Hwa, and Weigt 2011; Nguyen, Zecchina, and Berg 2017; Nguyen and Wood 2016a,b), making its use in supervised learning a viable alternative to regression methods. Two approximation methods available for use are pseudo-likelihood inference (Besag 1975) and mean field theory (Chandler 1987; Nguyen *et al.* 2017).

A recently described package **BoltzMM** can fit the ('fully visible') Boltzmann machine given by Eq. (3) to data using pseudo-likelihood inference (Jones, Nguyen, and Bagnall 2019b; Jones, Bagnall, and Nguyen 2019a). In contrast, classifiers based on this class of models remain largely unexplored. Supervised learners using statistical models of the type (3) usually take the form of the *restricted* Boltzmann machines instead (Hinton 2012), where (visible) predictors are augmented by hidden units and interactions are zero except between visible and hidden units. The main drawback of such layered Boltzmann machine learners, as is common in all deep learning algorithms, is the difficulty in interpreting trained models. In contrast, with the fully visible architecture, $J_{ij}^{(y)}$ in Eq. (3), if inferred with sufficient power while avoiding overfitting, has direct interpretation of interaction between two variables.

We refer to such learning/prediction algorithms using a generalized version of Eq. (3) as Boltzmann Bayes inference. An implementation specific to genomic single-nucleotide polymorphism (SNP) data (two response groups, e.g., case and control, and uniform three-level predictors, i.e., allele counts of $x_i = 0, 1, 2$) has been reported previously (Woo, Yu, Kumar, Gold, and Reifman 2016). However, this C++ software was geared specifically toward genome-wide association studies and is not suitable for use in more general settings. We introduce an R package **bbl** (Boltzmann Bayes Learner), which uses both R and C++ for usability and performance, allowing the user to train and test statistical models in a variety of different usage settings.

2. Model and algorithm

For completeness and for reference to software features described in Section 3, we summarize in this section key relevant formulas (Woo *et al.* 2016) used by **bbl**, generalized such that predictors each can have varying number of factor levels.

2.1. Model description

The discrete response y_k for an instance k takes factor values y among $G \geq 2$ groups; e.g., $y = \texttt{case}, \texttt{control}$ with $G = 2$; $k = 1, \cdots, n$ denotes sample (or configuration) index. We also introduce weights w_k, each of which is integral number of times each configuration was observed in data, such that $\sum_k w_k = n_s$ is the total sample size. If the data take the form of one entry per observation, $w_k = 1$ and $n = n_s$. The use of frequency w_k can lead to more efficient learning when the number of predictors is relatively small. We use symbol y for a particular factor value and generic response variables interchangeably.

The model attempts to connect response y to a set of predictors represented by $\mathbf{x}$ with elements x_i and the observed data for an instance k denoted by $\mathbf{x}^k$. We assume that predictor variables take discrete factor levels, each with distinct effect on responses, e.g., $x_i = \texttt{a}, \texttt{t}, \texttt{g}, \texttt{c}$ for DNA sequence data. The overall likelihood is

$$L = \sum_k w_k \ln P(\mathbf{x}^k, y_k) = \sum_y \sum_{k \in \mathbb{K}_y} w_k \ln P(\mathbf{x}^k, y) \equiv \sum_y L_y, \tag{4}$$

where the second summation restricts k to the set $\mathbb{K}_y$ of all k values for which $y_k = y$. The inference is first performed for each group y separately, maximizing L_y given by

$$L_y = \sum_{k \in \mathbb{K}_y} w_k \Big[\ln P(\mathbf{x}^k | y) + \ln P(y) \Big] = \sum_{k \in \mathbb{K}_y} w_k \ln P(\mathbf{x}^k | y) + n_y \ln p_y, \tag{5}$$

where $p_y \equiv P(y)$ is the marginal distribution of y and $n_y = \sum_{k \in \mathbb{K}_y} w_k$ is the size of group y. In the parametrization we adopt for the first term in Eq. (5), the group-specific predictor distribution is written as

$$P(\mathbf{x}|y) = \frac{1}{Z_y} \exp \left[\sum_i h_i^{(y)}(x_i) + \sum_{i<j} J_{ij}^{(y)}(x_i, x_j) \right]. \tag{6}$$

The number of parameters (d.f.) per group y in $\Theta_y = \{h_i^{(y)}(x), J_{ij}^{(y)}(x, x')\}$ is

$$\text{d.f.} = \sum_i (L_i - 1) + \sum_{i<j} (L_i - 1)(L_j - 1), \tag{7}$$

where L_i is the total number of levels in factor x_i, which contributes one less parameters to d.f. because one of the factors can be taken as reference with the rest measured against it. Internally, **bbl** orders factors, assigns codes $a_i = 0, \cdots, L_i - 1$, and set $h_i^{(y)}(a_i) = J_{ij}^{(y)}(a_i, a_j) = 0$ whenever $a_i = 0$ or $a_j = 0$. We refer to $h_i^{(y)}(x)$ and $J_{ij}^{(y)}(x, x')$ as bias and interaction parameters, respectively.

In the special case where predictor levels are binary ($x_i = 0, 1$), one may use the spin variables $s_i = 2x_i - 1 = \pm 1$, as in the package **BoltzMM** (Jones *et al.* 2019a,b). Its distribution (Jones *et al.* 2019a)

$$P(\mathbf{s}) \propto \exp \left(\frac{1}{2} \mathbf{s}^\top \mathbf{M}\, \mathbf{s} + \mathbf{b}^\top \mathbf{s} \right) \tag{8}$$

is then related to Eq. (3) by

$$b_i = \frac{h_i}{2} + \frac{1}{4} \sum_{j \neq i} J_{ij}, \tag{9a}$$

$$M_{ij} = \frac{1}{4} J_{ij}, \tag{9b}$$

where parameter superscripts were omitted because response group is not present.

2.2. Pseudo-likelihood inference

One option for fitting Eq. (6) to data is pseudo-likelihood maximization (Besag 1975):

$$L_y - n_y \ln p_y = \sum_{k\in\mathbb{K}_y} w_k \ln P(\mathbf{x}^k|y) \approx \sum_{k\in\mathbb{K}_y} w_k \sum_i \ln P_i(x_i^k|y, x_{j\setminus i}^k) \equiv \sum_i L_{iy}, \tag{10}$$

where the effective univariate distribution is conditional to all other predictor values:

$$P_i(x|y, x_{j\setminus i}) = \frac{e^{\bar{h}_i^{(y)}(x|x_{j\setminus i})}}{Z_{iy}(x_{j\setminus i})}, \tag{11}$$

$$Z_{iy}(x_{j\setminus i}) = \sum_x e^{\bar{h}_i^{(y)}(x|x_{j\setminus i})} = 1 + \sum_{a=1}^{L_i-1} e^{\bar{h}_i^{(y)}(a|x_{j\setminus i})}, \tag{12}$$

and

$$\bar{h}_i^{(y)}(x|x_{j\setminus i}) = h_i^{(y)}(x) + \sum_{j\neq i} J_{ij}^{(y)}(x, x_j). \tag{13}$$

Including L_2 penalizers (λ_h, λ), L_{iy} in Eq. (10) becomes

$$L_{iy} = \sum_{k\in\mathbb{K}_y} w_k \left[\bar{h}_i^{(y)}(x_i^k|x_{j\setminus i}^k) - \ln Z_{iy}(x_{j\setminus i}^k)\right] - \frac{\lambda_h}{2}\sum_x h_i^{(y)}(x)^2 - \frac{\lambda}{2}\sum_{j,x,x'} J_{ij}^{(y)}(x, x')^2 \tag{14}$$

with first derivatives

$$\frac{\partial L_{iy}/n_y}{\partial h_i^{(y)}(x)} = \hat{f}_i^{(y)}(x) - \frac{1}{n_y}\sum_{k\in\mathbb{K}_y} w_k P_i(x|y, x_{l\setminus i}^k) - \lambda_h h_i^{(y)}(x), \tag{15a}$$

$$\frac{\partial L_{iy}/n_y}{\partial J_{ij}^{(y)}(x, x')} = \hat{f}_{ij}^{(y)}(x, x') - \frac{1}{n_y}\sum_{k\in\mathbb{K}_y} w_k \mathbb{1}(x_j^k = x') P_i(x|y, x_{l\setminus i}^k) - \lambda J_{ij}^{(y)}(x, x'), \tag{15b}$$

where

$$\hat{f}_i^{(y)}(x) = \frac{1}{n_y}\sum_{k\in\mathbb{K}_y} w_k \mathbb{1}(x_i^k = x), \tag{16a}$$

$$\hat{f}_{ij}^{(y)}(x, x') = \frac{1}{n_y}\sum_{k\in\mathbb{K}_y} w_k \mathbb{1}(x_i^k = x)\mathbb{1}(x_j^k = x') \tag{16b}$$

are the first and second moments of predictor values and $\mathbb{1}(x)$ is the indicator function. In **bbl**, Eqs. (15) are solved in C++ functions using the quasi-Newton optimization function `gsl_multimin_fdfminimizer_vector_bfgs2` in GNU Scientific Library (`https://www.gnu.org/software/gsl`). By default, $\lambda_h = 0$ and only interaction parameters are penalized. As can be seen from the third equality of Eq. (10), the pseudo-likelihood inference decouples into individual predictors, and the inference for each i in **bbl** is performed sequentially. The resulting interaction parameters, however, do not satisfy the required symmetry,

$$J_{ij}(x, x') = J_{ji}(x', x). \tag{17}$$

After pseudo-likelihood inference, therefore, the interaction parameters are symmetrized as follows:

$$J_{ij}(x,x') \leftarrow \frac{1}{2}\left[J_{ij}(x,x') + J_{ji}(x',x)\right]. \tag{18}$$

In **bbl**, the input data are filtered such that predictors with only one factor level (no variation in observed data) are removed. Nevertheless, in cross-validation of the processed data, subdivisions into training and validation sets may lead to instances where factor levels observed for a given predictor within x_i in Eq. (15) are only a subset of those in the whole data. It is thus possible that optimization based on Eqs. (15) is ill-defined when any of the predictors are constant. In such cases, we augment the training data by an extra instance, in which constant predictors take other factor levels.

2.3. Mean field inference

The other option for predictor distribution inference is mean field approximation. In data-driven inference, the interaction parameters are approximated as (Nguyen *et al.* 2017)

$$\hat{J}_{ij}^{(y)}(x,x') = -\left[\mathsf{C}^{(y)}\right]^{-1}_{ij}(x,x'), \tag{19}$$

i.e., negative inverse of the covariance matrix,

$$\mathsf{C}_{ij}^{(y)}(x,x') = \hat{f}_{ij}(x,x') - \hat{f}_i(x)\hat{f}_j(x'). \tag{20}$$

Equation (19) can be interpreted as treating discrete $\mathbf{x}$ as if it were multivariate normal: Eq. (6) would then be the counterpart of the multivariate normal probability density function with $-J_{ij}^{(y)}(x,x')$ corresponding to the precision matrix. In real data where $n \sim$ d.f. or less, the matrix inversion is often ill-behaved. It is regularized by interpolation of $\mathsf{C}^{(y)}$ between non-interacting (naive Bayes) ($\epsilon = 0$) and fully interacting limits ($\epsilon = 1$):

$$\mathsf{C}^{(y)} \leftarrow \bar{\mathsf{C}}^{(y)} = (1-\epsilon)\frac{\operatorname{Tr}\mathsf{C}^{(y)}}{\operatorname{Tr}\mathsf{I}}\mathsf{I} + \epsilon\mathsf{C}^{(y)}, \tag{21}$$

where I is the identity matrix of the same dimension as $\mathsf{C}^{(y)}$. The parameter ϵ serves as a good handle for probing the relative importance of interaction effects.

The bias parameters are given in mean field by an analog of Eq. (13),

$$\hat{h}_i^{(y)}(x) = \bar{h}_i^{(y)}(x) - \sum_{j\neq i}\sum_{x'}\hat{J}_{ij}^{(y)}(x,x')\hat{f}_j^{(y)}(x'), \tag{22}$$

and

$$\bar{h}_i^{(y)}(x) = \ln\left[\hat{f}_i^{(y)}(x)/\hat{f}_i^{(y)}(0)\right], \tag{23}$$

where $\hat{f}_i^{(y)}(0)$ is the frequency of (reference) factor x_i for which the parameters are zero ($a_i = 0$). Equation (22) relates the effective bias for predictor x_i (the first term on the right) as the sum of univariate bias (left-hand side) and combined mean effects of interactions with other variables (the second term on the right) (Chandler 1987). The effective bias is related to frequency via Eq. (23) because

$$\hat{f}_i^{(y)}(x) = \frac{e^{\bar{h}_i^{(y)}(x)}}{Z_{iy}} = \hat{f}_i^{(y)}(0)\, e^{\bar{h}_i^{(y)}(x)} \tag{24}$$

where the fact that $\bar{h}_i^{(y)}(0) = 0$ was used in the second equality.

As in pseudo-likelihood maximization, mean field inference also may encounter non-varying predictors during cross-validation. To apply the same inference scheme using Eqs. (20), (22) and (23) to such cases, the single-variable frequency $\hat{f}_i^{(y)}(x)$ and covariance $\hat{f}_{ij}^{(y)}(x, x')$ are computed using data augmented by a prior count of 1 uniformly distributed among all L_i factor levels for each predictor.

2.4. Naive Bayes

When interaction is ignored ($J_{ij}^{(y)} = 0$), the model can be solved analytically. From Eqs. (22) and (23),

$$\hat{h}_i^{(y)}(x) = \ln\left[\hat{f}_i^{(y)}(x)/\hat{f}_i^{(y)}(0)\right] \tag{25}$$

and (Woo *et al.* 2016)

$$L_y - n_y \ln p_y = \sum_{k\in\mathbb{K}_y} w_k \ln P(\mathbf{x}^k|y) = n_y \sum_{i,x} \hat{f}_i^{(y)}(x) \ln \hat{f}_i^{(y)}(x). \tag{26}$$

The likelihood ratio statistic for each predictor, where the null hypothesis is $h_i^{(y)}(x) = h_i(x)$ with $h_i(x)$ the "pooled" inference parameters (same values for all response groups), is then

$$q_i = 2\sum_y n_y \sum_x \left[\hat{f}_i^{(y)}(x) \ln \hat{f}_i^{(y)}(x) - \hat{f}_i(x) \ln \hat{f}_i(x)\right]. \tag{27}$$

The statistic $q_i \sim \chi^2$ with d.f. $= (G-1)(L_i - 1)$. Another example of hypotheses that can be tested is $h_i^{(y)}(x) = h_i^{(y)}(A)$ for $x \in X_A$, where X_A is a subset A of predictor values (e.g., in `Titanic` model, the effects of `Class` are the same for `2nd` and `3rd Class`; see Sec. 3), for which

$$q_i = 2\sum_y n_y \sum_A \sum_{x\in X_A} \left[\hat{f}_i^{(y)}(x) \ln \hat{f}_i^{(y)}(x) - \hat{f}_i^{(y)}(A) \ln \hat{f}_i^{(y)}(A)\right] \tag{28}$$

with d.f. $= G(L_i - 1 - N_i)$, where N_i is the number of predictor levels with distinct parameter values.

2.5. Classification

For prediction, we combine predictor distributions for all response groups via Bayes formula:

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y)p_y}{\sum_{y'} P(\mathbf{x}|y')p_{y'}} = \frac{1}{1 + \sum_{y'\neq y} P(\mathbf{x}|y')p_{y'}/P(\mathbf{x}|y)p_y} = \frac{1}{1 + e^{-F_y(\mathbf{x})}}, \tag{29}$$

where

$$F_y(\mathbf{x}) = \ln\left[\frac{P(\mathbf{x}|y)p_y}{\sum_{y'\neq y} P(\mathbf{x}|y')p_{y'}}\right]. \tag{30}$$

For binary response coded as $y = 0, 1$, Eq. (30) reduces to

$$\begin{aligned} F_1(\mathbf{x}) &= \ln P(\mathbf{x}|y=1) - \ln P(\mathbf{x}|y=0) + \ln(p_1/p_0) \\ &= \alpha + \sum_i \beta_i(x_i) + \sum_{i<j} \gamma_{ij}(x_i, x_j), \end{aligned} \tag{31}$$

where

$$\begin{aligned} \alpha &= \ln \frac{Z_0 p_1}{Z_1 p_0}, \\ \beta_i(x) &= h_i^{(1)}(x) - h_i^{(0)}(x), \\ \gamma_{ij}(x, x') &= J_{ij}^{(1)}(x, x') - J_{ij}^{(0)}(x, x'). \end{aligned} \tag{32}$$

Therefore, if $J_{ij}^{(y)}(x, x') = 0$ (naive Bayes), Eq. (29) takes the form of the logistic regression formula. However, the actual naive Bayes parameter values differ from logistic regression fit. No expression for $P(y|\mathbf{x})$ simpler than Eq. (29) exists for data with more than two groups.

In pseudo-likelihood maximization inference, Z_y can be approximated by

$$\ln Z_y = \frac{1}{n_y} \sum_{k \in \mathbb{K}_y} \sum_i \ln \left\{ \sum_x \left[e^{h_i^{(y)}(x) + \sum_{j \neq i} J_{ij}(x, x_j^k)/2} \right] \right\}, \tag{33}$$

or with the same expression without the factor of 1/2 in the interaction term in the exponent (default). This quantity can be conveniently computed during the optimization process. With the mean field option, the following expression is used:

$$\ln Z_y = -\ln \hat{f}^{(y)}(0) - \frac{1}{2} \sum_{i \neq j} \sum_{x, x'} J_{ij}(x, x') \hat{f}_i(x) \hat{f}_j(x'). \tag{34}$$

For a test data set for which the actual group identity y_k of data instances are known, the accuracy may be defined as

$$s = \frac{1}{n} \sum_k \mathbb{1} \left[\hat{y}(\mathbf{x}^k) = y_k \right], \tag{35}$$

where

$$\hat{y}(\mathbf{x}) = \arg\max_y P(y|\mathbf{x}). \tag{36}$$

If response is binary, the accuracy defined by Eq. (35) is sensitive to marginal distributions of the two groups via Eq. (31). The area under curve (AUC) of receiver operating characteristic is a more robust performance measure independent of probability cutoff. In **bbl**, the accuracy given by Eqs. (35) and (36) is used in general with the option to use AUC for binary response using R package **pROC** (Robin, Turck, Hainard, Tiberti, Lisacek, Sanchez, and Müller 2011; Robin, Turck, Hainard, Tiberti, Lisacek, Sanchez, Müller, Siegert, and Doering 2019).

3. Software Usage and Tests

3.1. Logistic regression

To motivate the use of **bbl** and highlight differences, we first consider the use of logistic regression using `glm`. We use the base R data `Titanic` as an example:

```
R> titanic <- as.data.frame(Titanic)
R> titanic
```

```
   Class    Sex   Age Survived Freq
1    1st   Male Child       No    0
2    2nd   Male Child       No    0
3    3rd   Male Child       No   35
4   Crew   Male Child       No    0
5    1st Female Child       No    0
6    2nd Female Child       No    0
7    3rd Female Child       No   17
8   Crew Female Child       No    0
9    1st   Male Adult       No  118
10   2nd   Male Adult       No  154
11   3rd   Male Adult       No  387
12  Crew   Male Adult       No  670
13   1st Female Adult       No    4
14   2nd Female Adult       No   13
15   3rd Female Adult       No   89
16  Crew Female Adult       No    3
17   1st   Male Child      Yes    5
18   2nd   Male Child      Yes   11
19   3rd   Male Child      Yes   13
20  Crew   Male Child      Yes    0
21   1st Female Child      Yes    1
22   2nd Female Child      Yes   13
23   3rd Female Child      Yes   14
24  Crew Female Child      Yes    0
25   1st   Male Adult      Yes   57
26   2nd   Male Adult      Yes   14
27   3rd   Male Adult      Yes   75
28  Crew   Male Adult      Yes  192
29   1st Female Adult      Yes  140
30   2nd Female Adult      Yes   80
31   3rd Female Adult      Yes   76
32  Crew Female Adult      Yes   20
```

Although more detailed versions of the same data set are available [see, e.g., **titanic** (Hendricks 2015) or **stablelearner** (Philipp, Strobl, Zeileis, Rusch, and Hornik 2018b; Philipp, Rusch, Hornik, and Strobl 2018a)], the simpler version above only including factor variables suffices for our purposes because **bbl** requires discrete factors as predictors. Input data can either be of the form above with unique combinations of predictors in each row along with frequency (input to `weights` argument of `glm`) or raw data (one observation per row) we generate using the utility function `freq2raw`:

```
R> library('bbl')
R> titanic_raw <- freq2raw(data = titanic, freq = Freq)
R> head(titanic_raw)
```

```
  Class  Sex   Age Survived
1   3rd Male Child       No
```

```
2   3rd Male Child       No
3   3rd Male Child       No
4   3rd Male Child       No
5   3rd Male Child       No
6   3rd Male Child       No
```

```
R> summary(titanic_raw)
```

```
  Class         Sex          Age       Survived
 1st :325   Male  :1731   Child: 109   No :1490
 2nd :285   Female: 470   Adult:2092   Yes: 711
 3rd :706
 Crew:885
```

We train a logistic regression model using `glm`:

```
R> gfit0 <- glm(Survived ~ Class + Sex + Age, family = binomial(),
+    data = titanic, weights = Freq)
R> coef(summary(gfit0))
```

```
              Estimate Std. Error    z value     Pr(>|z|)
(Intercept)  0.6853195  0.2729942   2.510381 1.206011e-02
Class2nd    -1.0180950  0.1959975  -5.194428 2.053497e-07
Class3rd    -1.7777622  0.1715665 -10.361940 3.693921e-25
ClassCrew   -0.8576762  0.1573389  -5.451140 5.004800e-08
SexFemale    2.4200603  0.1404101  17.235662 1.434015e-66
AgeAdult    -1.0615424  0.2440256  -4.350127 1.360589e-05
```

The fit above included linear terms only. It indicates that survival was strongly associated with class status, sex (female heavily favored), and age. The model below includes all interactions:

```
R> gfit1 <- glm(Survived ~ (Class + Sex + Age)^2, family = binomial(),
+    data = titanic, weights = Freq)
R> coef(summary(gfit1))
```

```
                       Estimate  Std. Error       z value
(Intercept)          14.77919783 437.9677040  0.0337449490
Class2nd              0.26020943 549.2944350  0.0004737158
Class3rd            -15.76959654 437.9678231 -0.0360062902
ClassCrew            -0.52214898   0.1808848 -2.8866384473
SexFemale             3.59619056   0.7478095  4.8089662070
AgeAdult            -15.50683119 437.9677334 -0.0354063325
Class2nd:SexFemale   -0.06800887   0.6711978 -0.1013246346
Class3rd:SexFemale   -2.79994787   0.5687464 -4.9230165411
ClassCrew:SexFemale  -1.13607909   0.8204849 -1.3846434869
Class2nd:AgeAdult    -1.93047135 549.2945246 -0.0035144558
Class3rd:AgeAdult    14.85629331 437.9678705  0.0339209662
```

```
SexFemale:AgeAdult    0.68679086   0.5254120  1.3071473287
                        Pr(>|z|)
(Intercept)         9.730805e-01
Class2nd            9.996220e-01
Class3rd            9.712773e-01
ClassCrew           3.893814e-03
SexFemale           1.517128e-06
AgeAdult            9.717557e-01
Class2nd:SexFemale  9.192928e-01
Class3rd:SexFemale  8.522025e-07
ClassCrew:SexFemale 1.661615e-01
Class2nd:AgeAdult   9.971959e-01
Class3rd:AgeAdult   9.729402e-01
SexFemale:AgeAdult  1.911627e-01
```

A comparison of the linear coefficients and significance levels in the two models suggest that interaction plays important roles; in particular, marginal effects on the linear level remained significant only for the `Female` status.

To illustrate training and prediction, we divide the sample into train and test sets:

```
R> set.seed(159)
R> nsample <- NROW(titanic_raw)
R> flag <- rep(TRUE, nsample)
R> flag[sample(nsample, nsample/2)] <- FALSE
R> dtrain <- titanic_raw[flag,]
R> dtest <- titanic_raw[!flag,]
```

We train a `glm` model with interactions and make prediction on the test data:

```
R> gfit2 <- glm(Survived ~ Class * Sex + Sex * Age, family = binomial(),
+    data = dtrain)
R> prl <- predict(gfit2, newdata = dtest)
R> yhat <- ifelse(prl > 0, 'Yes', 'No')
R> mean(yhat == dtest$Survived)

[1] 0.7718182

R> gauc <- pROC::roc(response = dtest$Survived, predictor = prl,
+    direction = '<')$auc
R> gauc

Area under the curve: 0.7699
```

In the above, the interaction `Class:Age` was omitted because it was rank-deficient (no `Crew` among children) and prediction from a rank-deficient fit is ill-defined.

For comparison with **bbl**, which by default includes regularization, we also consider penalized logistic regression fit using **glmnet** (Friedman, Hastie, and Tibshirani 2010; Friedman, Hastie, Tibshirani, Narasimhan, Simon, and Qian 2019)

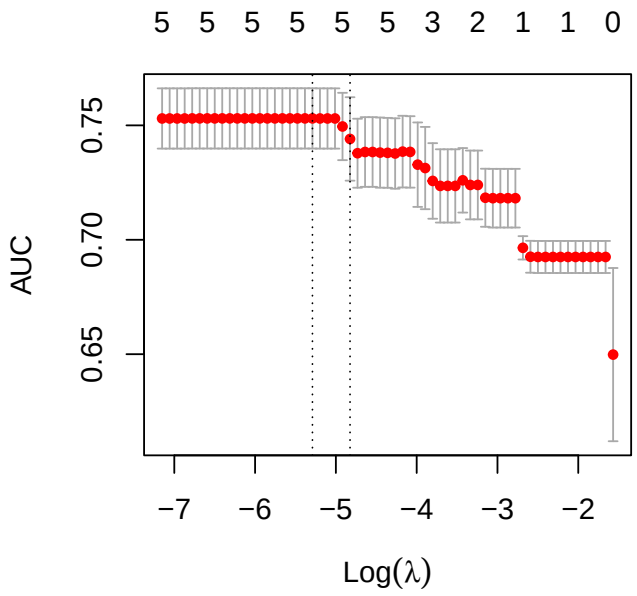

Figure 1: Cross-validation run of `glmnet` on `Titanic` data.

```
R> if(!require('glmnet'))
+    install.packages('glmnet')
R> library('glmnet')
R> xdat <- model.matrix(~ Class + Sex + Age, data = dtrain)[,-1]
R> y <- dtrain[, 4]
R> gnet <- cv.glmnet(x = xdat, y = y, family = 'binomial', alpha = 1,
+    nfolds = 5, type.measure = 'auc')
R> plot(gnet)
```

Note that the above fit used the non-interacting model of three predictors and L_1 penalization ($\alpha = 1$). The input matrix contains integer-coded terms in the linear model (columns):

```
R> head(xdat)
```

```
   Class2nd Class3rd ClassCrew SexFemale AgeAdult
4         0        1         0         0        0
5         0        1         0         0        0
7         0        1         0         0        0
8         0        1         0         0        0
13        0        1         0         0        0
14        0        1         0         0        0
```

Figure 1 indicates that the effect of regularization is minimal for this model.

3.2. Boltzmann Bayes learning

The logistic regression shown in Section 3.1 allowed for inference and significance testing of linear and interaction coefficients in association with the response variable. However, the regression fit did not provide any further information regarding the source of association: in the examples in Section 3.1, the survival of Titanic passengers was seen to be associated with being `Female` and not being `Crew` members. The corresponding linear regression coefficients, which have the same functional form as in Eq. (31) [$\beta_i(x)$ in Eq. (32) if interactions are neglected], are measures of the *difference* in coefficients $h_i^{(y)}$ between the two response groups [Eq. (32)]. The two terms, $h_i^{(1)}$ and $h_i^{(0)}$, whose difference yielded the coefficient $\beta_i(x)$ remained unknown. How were the sub-groups distributed among survivor and non-survivor groups? Were there very few `Female 3rd`-class passengers among the survivor group compared to non-survivor, or were they found in both groups but more so among non-survivors?

The **bbl** inference estimates the individual distributions of predictors in response groups separately and subsequently combines them to make predictions. For binary response, this inference provides estimates of the two coefficients [$h_i^{(1)}$, $h_i^{(0)}$ for linear effects and $J_{ij}^{(1)}$, $J_{ij}^{(0)}$ for interactions] in Eq. (31) whose difference corresponds to the logistic regression coefficients. More generally, the availability of the direct estimates of predictor distributions in each response group given by Eq. (6) facilitates model interpretations in a way not possible for regression-based models, as we show below in this section and Section 3.5.

With this comparison in mind, we use the same Titanic data below to illustrate the Boltzmann Bayes inference. As in **glm**, **bbl** uses formula input to train an `S3` object of class `bbl`:

```
R> bfit0 <- bbl(Survived ~ Class + Sex + Age, data = titanic, weights = Freq,
+    prior.count = 0)
```

which by default triggers a pair of pseudo-likelihood inferences, solving the maximum pseudo-likelihood equations (15) first under the alternative hypothesis (individual groups have distinct distributions) and then the null hypothesis (all samples have the same distribution).

The argument `prior.count` can be used to add prior counts to frequencies of occurrence of each predictor level. One may observe that when interaction is neglected, the naive Bayes model involves categorical distributions for each predictor. In this special case, therefore, the prior count can be regarded as the hyperparameter of the conjugate Dirichlet prior, making the overall treatment of the model a fully Bayesian extension.

The `print` method on `bbl` shows the structure of model and (subsets) of inferred parameters:

```
R> bfit0

Call:
 bbl(formula = Survived ~ Class + Sex + Age, data = titanic, weights = Freq,
    prior.count = 0)
 3 predictor states:
   Class = 1st 2nd 3rd Crew
   Sex = Female Male
   Age = Adult Child
 Responses:
```

```
   Survived = No Yes

Coefficients:
dh_[Class]^(No):
      2nd       3rd      Crew
0.4453027 0.6893738 0.7062034

dh_[Class]^(Yes):
       2nd        3rd       Crew
-0.4114330 -0.9075005 -0.9584320

dh_[Sex]^(No):
   Male
1.07819

dh_[Sex]^(Yes):
     Male
-1.239013

dh_[Age]^(No):
     Child
-0.3647851

dh_[Age]^(Yes):
    Child
0.5148763
```

where `dh` represents parameters $\Delta h_i^{(y)} = h_i^{(y)} - h_i$; i.e., individual group parameters offset by the pooled values. Internally, the parameters $h_i^{(y)}$ and $J_{ij}^{(y)}$ are stored as lists with argument order (y, i) and (y, i, j), respectively. The inner-most elements of the lists are vectors and matrices of dimension $L_i - 1 =$ `c(3,1,1)` and $(L_i - 1, L_j - 1)$, respectively. The `summary` method on `bbl` object prints out parameters and their significance test outcomes under the naive Bayes approximation (no interactions) as a rough overview of model under consideration:

```
R> summary(bfit0)

Call:
 bbl(formula = Survived ~ Class + Sex + Age, data = titanic, weights = Freq,
    prior.count = 0)
 3 predictor states:
   Class = 1st 2nd 3rd Crew
   Sex = Female Male
   Age = Adult Child
 Responses:
   Survived = No Yes
Fit method: mf
```

```
naive Bayes coefficients:
h_Class:
              2nd        3rd      Crew
No      0.3139728  1.4650752 1.7077243
Yes    -0.5425214 -0.1314224 0.0433803
pooled -0.1313360  0.7757901 1.0017625
chisq = 180.9014, df = 3, Pr(>chisq) = 5.633919e-39

h_Sex:
             Male
No     2.38189493
Yes    0.06472019
pooled 1.30372186
chisq = 434.4688, df = 1, Pr(>chisq) = 1.730842e-96

h_Age:
           Child
No     -3.319765
Yes    -2.440056
pooled -2.954528
chisq = 19.5606, df = 1, Pr(>chisq) = 9.745843e-06
```

The test results are those from likelihood ratio test applied to the naive Bayes result, Eq. (27), with the null hypothesis $h_i^{(y)}(a) = h_i(a)$. The tables of bias parameters shown above include those for two survival status groups. Their signs and magnitudes, along with the computed significance levels, clearly indicate the associations of lower `Class` status and being `Male` with non-survivors. There are few children among both survivors and non-survivors; hence highly negative bias parameters in all groups, although less so in survivor group, as expected. We note that the `summary` method displays naive Bayes results, for which simple analytic expressions for test results are available, even for models containing interactions.

One may compare the naive Bayes parameter $\beta_i(x)$ with the logistic regression coefficients:

```
R> cb0 <- coef(bfit0)
R> beta <- list(Class = cb0$h$Yes$Class - cb0$h$No$Class,
+    Sex = cb0$h$Yes$Sex - cb0$h$No$Sex, Age = cb0$h$Yes$Age - cb0$h$No$Age)
R> unlist(beta)

 Class.2nd  Class.3rd Class.Crew   Sex.Male  Age.Child
-0.8567357 -1.5968743 -1.6646354 -2.3172031  0.8796614

R> coef(summary(gfit0))[,'Estimate']

(Intercept)    Class2nd    Class3rd   ClassCrew   SexFemale
  0.6853195  -1.0180950  -1.7777622  -0.8576762   2.4200603
   AgeAdult
 -1.0615424
```

and observe that they are largely consistent (with different signs depending on which factor level was used as reference) but not identical.

We now fit an interacting model using `bbl`:

```
R> bfit <- bbl(Survived ~ Class * Sex + Sex * Age, data = titanic,
+    weights = Freq)
R> bfit

Call:
 bbl(formula = Survived ~ Class * Sex + Sex * Age, data = titanic,
    weights = Freq)
 3 predictor states:
   Class = 1st 2nd 3rd Crew
   Sex = Female Male
   Age = Adult Child
 Responses:
   Survived = No Yes

Coefficients:
dh_[Class]^(No):
     2nd      3rd     Crew
1.506746 2.993541 1.569273

dh_[Class]^(Yes):
       2nd        3rd       Crew
-0.1028470 -0.7492758 -0.1122300

dh_[Sex]^(No):
    Male
3.168334

dh_[Sex]^(Yes):
     Male
-1.074811

dh_[Age]^(No):
    Child
0.3879267

dh_[Age]^(Yes):
    Child
-0.175671


dJ_[Class,Sex]^(No):
          Male
2nd  -1.247307
```

```
3rd  -2.740353
Crew -1.399824

dJ_[Class,Sex]^(Yes):
            Male
2nd  -0.90438653
3rd  -0.01714753
Crew -0.30732860

dJ_[Sex,Age]^(No):
          Child
Male -0.4924723

dJ_[Sex,Age]^(Yes):
        Child
Male 1.165608
```

```
R> plot(bfit)
```

The parameters printed include those for interactions. The `plot` method shows a barplot of bias parameters and a heatmap of interaction parameters (Fig. 2). Note that `Male` members were predominant (bias parameters; top), while `Male` `3rd`-class passengers were underrepresented (interactions; bottom left), among non-survivors. In addition, `Male-Child` class had enhanced survival (bottom right).

We now fit the training data and make prediction on test data:

```
R> bfit2 <- bbl(Survived ~ Class * Sex + Sex * Age, data = dtrain)
R> pr <- predict(bfit2, newdata = dtest, type = 'prob')
R> head(pr)

         No       Yes yhat
1 0.8092695 0.1907305   No
2 0.8092695 0.1907305   No
3 0.8092695 0.1907305   No
4 0.8092695 0.1907305   No
5 0.8092695 0.1907305   No
6 0.8092695 0.1907305   No

R> auc <- pROC::roc(response = dtest$Survived, predictor = pr[, 2],
+    direction = '<')$auc
R> auc

Area under the curve: 0.7707
```

Here, Eq. (29) was used with `x` from the supplied `newdata`. The `predict` method returns a data frame containing predicted group probabilities and the most likely group for each row.

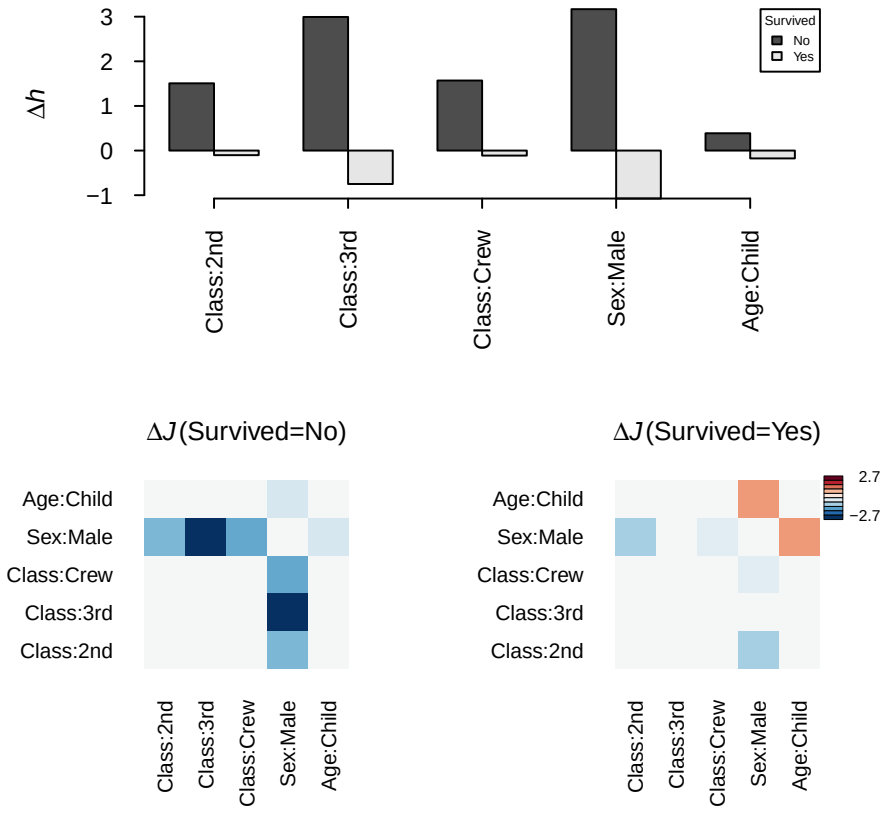

Figure 2: Plot of `bbl` object displays bias (top) and interaction parameters (bottom). All parameters are offset by their pooled (singe-group) values.

One can do cross-validation applied to `dtrain` data, dividing it into `nfold = 5` train/validation subsets of 4:1 proportion, and aggregating predictions for validation sets using the trained model:

```
R> cv <- crossVal(Survived ~ .^2, data = dtrain, method = 'pseudo',
+    lambda = 10^seq(-5, -2, 0.2), verbose = 0)
R> cv

Optimal lambda = 0.0002511886
Max. score: 0.7369487

         lambda       AUC       ci1       ci2
1  1.000000e-05 0.7184460 0.6861087 0.7507834
2  1.584893e-05 0.7162328 0.6838517 0.7486139
3  2.511886e-05 0.7275954 0.6955664 0.7596244
4  3.981072e-05 0.7289819 0.6965950 0.7613689
5  6.309573e-05 0.7280662 0.6956884 0.7604440
```

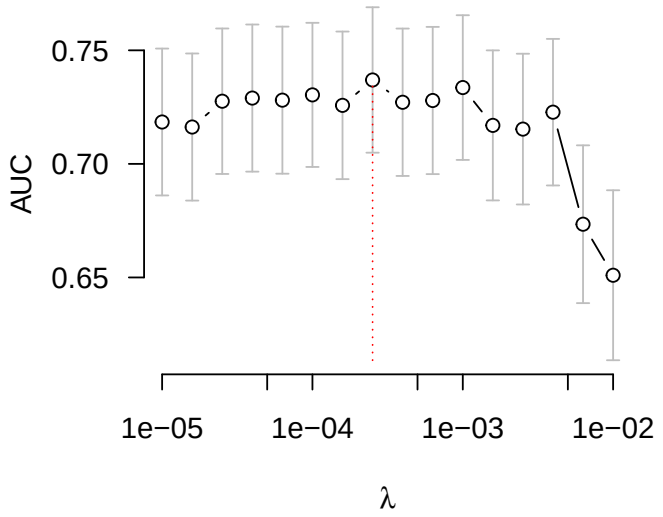

Figure 3: Cross-validation run of `Titanic` data in **bbl**.

```
6  1.000000e-04 0.7303795 0.6986485 0.7621106
7  1.584893e-04 0.7257752 0.6932851 0.7582653
8  2.511886e-04 0.7369487 0.7049321 0.7689653
9  3.981072e-04 0.7271394 0.6946901 0.7595888
10 6.309573e-04 0.7279105 0.6955033 0.7603178
11 1.000000e-03 0.7336048 0.7017480 0.7654616
12 1.584893e-03 0.7169557 0.6839247 0.7499867
13 2.511886e-03 0.7153134 0.6821272 0.7484996
14 3.981072e-03 0.7227723 0.6905092 0.7550355
15 6.309573e-03 0.6734404 0.6387032 0.7081776
16 1.000000e-02 0.6509709 0.6135566 0.6883852
```

```
R> plot(cv, mar=c(4, 4, 3, 3), tck = -0.04, bty = 'n')
```

Here, the model included all interaction terms and returned an object with a `data.frame` of AUCs for multiple `lambda` values as well as 95% confidence intervals and optimal values with maximum AUC. We use this information to make prediction as follows:

```
R> model <- bbl(Survived ~ .^2, data = dtrain, lambda = cv$regstar)
R> pr2 <- predict(model, newdata = dtest)
R> bscore <- mean(dtest$Survived == pr2$yhat)
R> bscore

[1] 0.7981818
```

```
R> bauc <- pROC::roc(response = dtest$Survived, predictor = pr2[,2],
+    direction = '<')$auc
R> bauc

Area under the curve: 0.7711
```

Alternatively, `predict(cv, ...)` will apply the optimal model within cross-validation to test data. The difference compared to the re-training step above is that the optimal model stored in `cv` was trained on 4/5 of the sample, while `model` above used the whole training set.

A major advantage of the `bbl` fit compared to regression is the availability of predictor distributions in each response group, $P(\mathbf{x}|y)$, given by Eq. (6). In addition to using the model to make predictions of response groups, one can also examine the predictor distributions and identify configurations dominant in each response group. Since the total number of configurations $\mathbf{x}$ grows exponentially with the number of predictors, Markov chain Monte Carlo (MCMC) sampling is necessary for exploration of these distributions except for very low dimensions. The function `mcSample` performs Gibbs sampling of the predictor distributions using `bbl` parameters and outputs the most likely configuration in each response group:

```
R> map <- mcSample(bfit, nstep = 1000, progress.bar = FALSE)
R> map

$xmax
      No      Yes
Class "3rd"   "1st"
Sex   "Male"  "Female"
Age   "Adult" "Adult"

$emax
      No      Yes
3.394166 0.000000
```

The return value is a list containing the predictor configurations with the highest probability in each response group (columns in `map$xmax` above) and the corresponding "energy" values, which are exponents of Eq. (6).

3.3. Simulated data

We next use simulated data to show the effect of penalizers on `bbl` inference as well as its usefulness under varying sample sizes.

```
R> predictors <- list()
R> m <- 5
R> L <- 3
R> for(i in 1:m) predictors[[i]] <- seq(0, L-1)
R> par <- randompar(predictors)
R> names(par)
```

```
[1] "h" "J"
```

The utility function `randompar` generates random parameters for predictors. We have set the total number of predictors as $m = 5$, each taking values $0, 1, 2$ ($L_i = L = 3$).

```
R> xi <- sample_xi(nsample = 10000, predictors = predictors, h = par$h,
+    J = par$J, code_out = TRUE)
R> head(xi)
```

```
1 1 1 0 1 2
2 1 1 0 0 0
3 1 1 0 1 1
4 0 1 2 0 1
5 2 0 0 0 0
6 1 1 0 1 0
```

The function `sample_xi` will list all possible predictor states and sample configurations based on the distribution (6). The total number of states here is $L^m = 3^5$, which is amenable for exhaustive enumeration. However, this is possible only for small m and L. If either are even moderately larger, `sample_xi` will hang.

Because there is only one response group, we call the main engine `mlestimate` of **bbl** inference directly instead of `bbl`:

```
R> fit <- mlestimate(xi = xi, method = 'pseudo', lambda = 0)
```

```
  Predictor 1: 25 iterations, likelihood = 0.796681
  Predictor 2: 27 iterations, likelihood = 1.03982
  Predictor 3: 23 iterations, likelihood = 0.946923
  Predictor 4: 22 iterations, likelihood = 0.827505
  Predictor 5: 25 iterations, likelihood = 1.01009
```

In contrast to `bbl` function, which fits a model of multiple response groups and predictors in factors, `mlestimate` is for a single group and requires input matrix `xi` whose elements are integral codes of factors: $a_i = 0, \cdots, L_i - 1$. Figure 4 compares the true and inferred parameters. Here, the sample size was large enough that no regularization was necessary.

We next simulate a full binary response data set with four-level predictors:

```
R> nt <- c('a', 'c', 'g', 't')
R> set.seed(135)
R> for(i in 1:m) predictors[[i]] <- nt
R> names(predictors) <- paste0('v', 1:m)
R> par <- list()
R> par[[1]] <- randompar(predictors)
R> par[[2]] <- randompar(predictors, h0 = 0.1, J0 = 0.1)
R> dat <- randomsamp(predictors, response = c('ctrl', 'case'), par = par,
+    nsample = 1000)
```

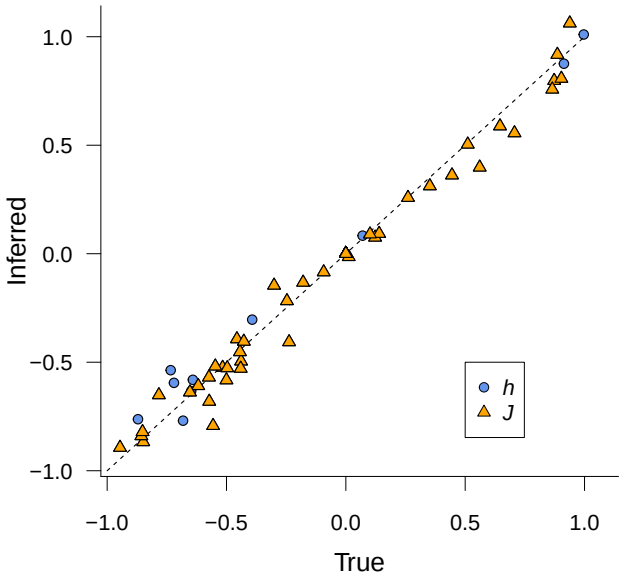

Figure 4: Comparison of true parameters and those inferred from pseudo-likelihood Boltzmann Bayes inference. See the text for conditions.

The function `randomsamp` generates random samples of `predictor-response` pairs using the supplied `par`. We perform a cross-validation using mean field inference,

```
R> cv <- crossVal(y ~ .^2, data = dat, method = 'mf', eps = seq(0, 1, 0.1),
+    verbose=0)
R> cv

Optimal epsilon = 0.7
Max. score: 0.8845219

   epsilon       AUC       ci1       ci2
1      0.0 0.7849546 0.7568076 0.8131017
2      0.1 0.8392593 0.8149947 0.8635240
3      0.2 0.8610941 0.8386831 0.8835051
4      0.3 0.8708767 0.8493991 0.8923543
5      0.4 0.8773411 0.8565066 0.8981756
6      0.5 0.8812357 0.8608067 0.9016647
7      0.6 0.8831850 0.8629906 0.9033795
8      0.7 0.8845219 0.8644845 0.9045594
9      0.8 0.8840456 0.8639740 0.9041172
10     0.9 0.8815880 0.8612527 0.9019232
11     1.0 0.8724978 0.8511909 0.8938047
```

Here, `bbl` is called inside `crossVal` as before but with `method = 'mf'`, which triggers mean field inference with Eqs. (19) and (22).

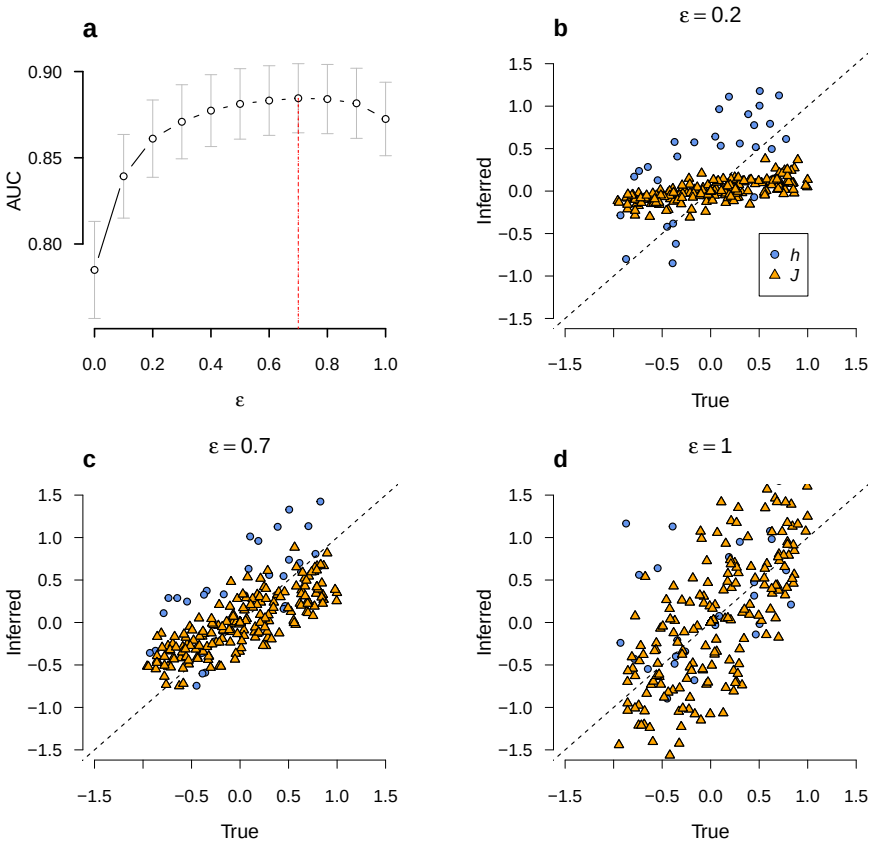

Figure 5: Regularized mean field inference using simulated data. (**a**) Cross-validation AUC with respect to regularization parameter ϵ. (**b-d**) Comparison of true and inferred parameters under three ϵ values. Best fit is achieved when AUC is maximum.

As shown in Fig. 5**a**, prediction AUC is optimized near $\epsilon = 0.7$. The difference between AUC at $\epsilon = 0$ (naive Bayes limit) and the maximum is a measure of the overall effect of interaction. We select three values of ϵ and examine the fit:

```
R> fit <- list()
R> eps <- c(0.2, 0.7, 1.0)
R> for(i in seq_along(eps))
+    fit[[i]] <- bbl(y ~ .^2, data = dat, method = 'mf', eps = eps[i],
+      verbose = 0)
```

Figure 5**b-d** compares the three inferred parameter sets (`coef(fit[[i]])$h, coef(fit[[i]])$J`) with the true values (`par[[iy]]$h, par[[iy]]$J`). As ϵ increases from 0 to 1, interaction parameter J grows from zero to large, usually overfit levels. We verify that the bias and variance strike the best balance under $\epsilon = 0.7$ (Fig. 5**c**), as suggested by cross-validation AUC in Fig. 5**a**.

3.4. Genetic code

We consider a different learning task example with a much larger space of response groups,

namely those of amino acids; $K = 21$, which include 20 amino acids plus stop signal ('*'), encoded by DNA sequences ($x_i = \mathtt{a}, \mathtt{c}, \mathtt{g}, \mathtt{t}$). In DNA sequences, three nucleotides combine to encode specific amino acids. We will train a model attempting to re-discover the mapping from nucleotide sequences to amino acids.

```
R> set.seed(351)
R> n <- 2000
R> dat <- data.frame(b1 = sample(nt, size = n, replace = TRUE),
+    b2 = sample(nt, size = n, replace = TRUE),
+    b3 = sample(nt, size = n, replace = TRUE))
R> head(dat)

  b1 b2 b3
1  t  a  g
2  g  t  c
3  t  a  a
4  c  g  g
5  a  a  c
6  c  t  g
```

In the above, we generated random instances of triplet codons for training. We use the package **Biostrings** (Pagès, Aboyoun, Gentleman, and DebRoy 2019) to translate it into amino acids:

```
R> if(!require('Biostrings')){
+    if(!require('BiocManager'))
+      install.packages('BiocManager')
+    BiocManager::install('Biostrings')
+  }
R> aa <- Biostrings::DNAString(paste(t(dat), collapse = ''))
R> aa

6000-letter DNAString object
seq: TAGGTCTAACGGAACCTGGCGATTATACTTG...AGTAAACTCGACAGTGACCGAAGGTACGGGC

R> aa <- strsplit(as.character(Biostrings::translate(aa)), split = '')[[1]]
R> xdat <- cbind(data.frame(aa = aa), dat)
R> head(xdat)

  aa b1 b2 b3
1  *  t  a  g
2  V  g  t  c
3  *  t  a  a
4  R  c  g  g
5  N  a  a  c
6  L  c  t  g
```

We now cross-validate using `bbl`:

```
R> cv <- crossVal(aa ~ .^2, data = xdat, lambda = 10^seq(-3, 1, 0.5),
+    verbose = 0)
R> cv

Optimal lambda = 0.3162278
Max. score: 1

        lambda  score
1  0.001000000 0.9195
2  0.003162278 0.9195
3  0.010000000 0.9875
4  0.031622777 0.9875
5  0.100000000 0.9925
6  0.316227766 1.0000
7  1.000000000 0.9930
8  3.162277660 0.9770
9 10.000000000 0.9770
```

Note that with the multinomial response group, the accuracy defined by Eq. (35) is used. The class `cv.bbl` extends `bbl` and stores the model with the optimal λ. In contrast to Section 3.2, we do not refit the model under this λ because accuracy is maximum. Testing can use all possible codon sequences ($4^3 = 64$ total):

```
R> panel <- expand.grid(b1 = nt, b2 = nt, b3 = nt)
R> head(panel)

  b1 b2 b3
1  a  a  a
2  c  a  a
3  g  a  a
4  t  a  a
5  a  c  a
6  c  c  a

R> dim(panel)

[1] 64  3

R> p <- predict(cv, panel)
R> ap <- Biostrings::DNAString(paste(t(panel), collapse = ''))
R> ap <- strsplit(as.character(Biostrings::translate(ap)), split = '')[[1]]
R> accuracy <- mean(ap == p$yhat)
R> accuracy

[1] 1
```

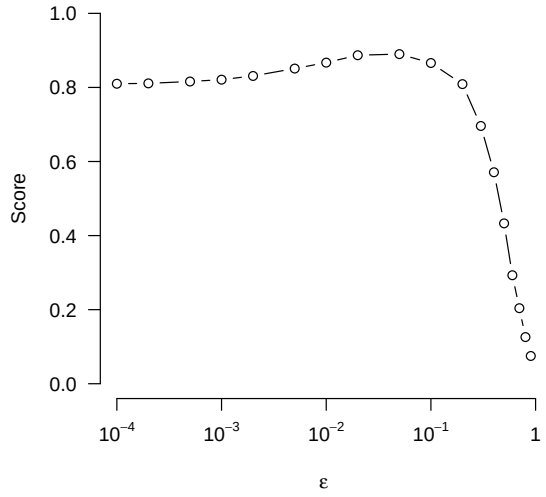

Figure 6: Cross-validation of Boltzmann Bayes inference on MNIST data using mean field option.

The trained model has perfect accuracy of 1 and will not make mistakes in any translation of DNA sequences.

3.5. Image data

We next consider learning examples with data sets containing predictors numbering ~ 100 or more. The MNIST data set (http://yann.lecun.com/exdb/mnist/), widely used for benchmarking classification algorithms (Lecun, Bottou, Bengio, and Haffner 1998), contains image data of grayscale levels ($x_i = [0, 255]$) derived from hand-written digits ($y_k = 0, \cdots, 9$) for $m = 28 \times 28 = 784$ pixels. We use down-sampled training ($n = 1{,}000$) and test ($n = 500$) data sets, where grayscale has been transformed into binary predictors ($x_i = 0, 1$):

```
R> dat0 <- read.csv(system.file('extdata/mnist_train.csv', package = 'bbl'))
R> dat <- removeConst(dat0)
R> dat[1:5, 1:10]
```

```
  y X40 X41 X45 X46 X47 X64 X65 X66 X67
1 9   0   0   0   0   0   0   0   0   0
2 7   0   0   0   0   0   0   0   0   0
3 1   0   0   0   0   0   0   0   0   0
4 8   0   0   0   0   0   0   0   0   0
5 1   0   0   0   0   0   0   0   0   0
```

```
R> cv <- crossVal(y ~ .^2, data = dat, method = 'mf', eps = 0.05)
```

Note that before calling `crossVal`, we removed predictors without factor variations (pixels that are always empty) using the utility function `removeConst`. By default, error will occur inside `crossVal` otherwise.

Algorithm	Method	Error rate (%)	Reference/package
Linear classifier	1-layer NN	12.0	Lecun *et al.* (1998)
K-nearest neighbors	Euclidean (L2)	5.0	Lecun *et al.* (1998)
2-layer NN	300 hidden units	4.7	Lecun *et al.* (1998)
RBM	2-layer	0.95	Salakhutdinov and Hinton (2009)
Naive Bayes	Mean field ($\epsilon = 0$)	15.7	**bbl**
BB	Mean field ($\epsilon = 0.05$)	8.5	**bbl**

Table 1: Performance comparison of BB inference and other models on MNIST data set. The **bbl** inferences used the full MNIST training and test data sets (see text). BB, Boltzmann Bayes; NN, neural network; RBM, restricted Boltzmann machine.

The above run will take a few minutes and yield a prediction score of 0.89. By feeding a vector of ϵ values, one can obtain the profile shown in Fig. 6. The jump in performance under $\epsilon^* \sim 0.05$ over $\epsilon \to 0$ (naive Bayes) limit gives a measure of interaction effects. The relatively small value of ϵ^* at the optimal condition, compared to e.g., Fig. 5**a**, reflects the sparseness of image data.

We now retrain the model without cross-validation under ϵ^* and classify test set images:

```
R> mnist <- bbl(y ~ .^2, data = dat, method = 'mf', eps = 0.05)
R> dtest <- read.csv(system.file('extdata/mnist_test.csv', package = 'bbl'))
R> dtest <- dtest[, colnames(dtest) %in% colnames(dat)]
R> pr <- predict(mnist, newdata = dtest[, -1], progress.bar = TRUE)
R> accuracy <- mean(pr$yhat == dtest$y)

R> accuracy

[1] 0.916
```

The test data must have the same set of predictors as those in `mnist`. Note the increase in accuracy compared to cross-validation value because of the use of full training data.

We performed similar cross-validation and test analyses of the full MNIST data (training $n = 60{,}000$ and test $n = 10{,}000$) and obtained the accuracy of 0.915 (classification error rate 8.5%), which compares favorably with other large-scale neural network algorithms (Table 1).

As with Titanic data, we leverage the unique advantage of `bbl` fit of providing predictor distributions and estimate dominant configurations of each response group (Fig. 7):

```
R> mnist_map <- mcSample(mnist, nstep = 20, progress.bar = TRUE)
R> oldpar <- par(mfrow = c(2, 5), mar = c(1, 1, 1, 1))
R> xvar <- colnames(dat0[, -1])
R> xmap <- apply(mnist_map$xmax, 1:2, as.numeric)
R> xf <- matrix(0, nrow = length(xvar), ncol = 10)
R> rownames(xf) <- xvar
R> for(i in 1:10) xf[rownames(xmap), i] <- xmap[, i]
R> for(i in 1:10){
+    mat <- matrix(t(xf[, i]), nrow = 28, ncol = 28)
```

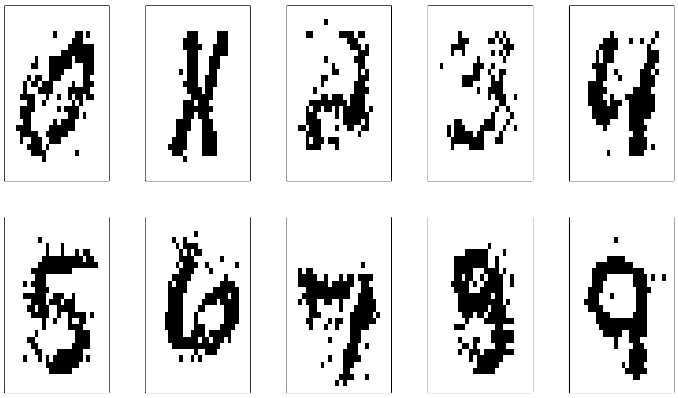

Figure 7: Maximum probability configurations of digits $(0, \cdots, 9)$ estimated from `bbl` fit coefficients using Gibbs sampling.

```
+    image(x = 1:28, y = 1:28, z = mat[, 28:1], col = c('white', 'black'),
+      xaxt = 'n', yaxt = 'n', xlab = '', ylab = '')
+  }
R> par(oldpar)
```

It is interesting to note that the model for handwritten digit "1" is a combination of two versions, one slated forward and the other backward. The images shown in Fig. 7 illustrate examples of model interpretation made possible by the Bayesian formulation used by **bbl**, a significant advantage compared to regression-based methods and other deep learning models whose interpretations are challenging (Montavon, Samek, and Müller 2018).

3.6. Transcription factor binding site data

One of machine learning tasks of considerable interest in biomedical applications is the detection of transcription factor binding sites within genomic sequences (Wasserman and Sandelin 2004). Transcription factors are proteins that bind to specific DNA sequence segments and regulate gene expression programs. Public databases, such as JASPAR (Khan, Fornes, Stigliani, Gheorghe, Castro-Mondragon, van der Lee, Bessy, Chéneby, Kulkarni, Tan, Baranasic, Arenillas, Sandelin, Vandepoele, Lenhard, Ballester, Wasserman, Parcy, and Mathelier 2018), host known transcription factors and their binding sequence motifs. Supervised learners allow users to leverage these data sets and search for binding motifs among candidate sequences.

Here, we illustrate such an inference using an example set (MA0014.3) of binding motif sequences from JASPAR (`http://jaspar.genereg.net`):

```
R> seq <- readFasta(system.file('extdata/MA0014.3.fasta', package = 'bbl'))
R> head(seq)
```

```
  1 2 3 4 5 6 7 8 9 10 11 12
1 G G G C G T G A C  T  T  C
2 C A G C G T G A C  G  C  G
```

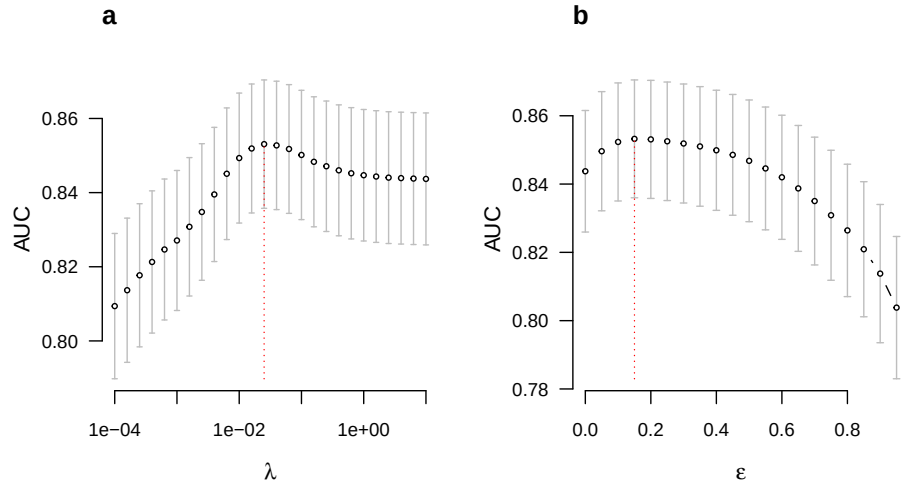

Figure 8: Cross-validation of transcription factor binding motif model using **bbl** with control sequences generated by 3 nucleotide mutations. Data set is from Khan *et al.* (2018) (sample ID MA0014.3; see text). (**a**) Pseudo-likelihood and (**b**) mean field inferences.

```
3 G C G C G T C A C  G  C  T
4 C A G C T T G A C  C  A  G
5 G A C C G T G A C  C  A  C
6 A G G C G C G A C  G  C  C
```

```
R> dim(seq)
```

```
[1] 948  12
```

The data set consists of common nucleotide segments from $n = 948$ raw sequences used for motif discovery. We simulate a training set by generating non-binding sequences with random mutation of 3 nucleotides:

```
R> set.seed(561)
R> nsample <- NROW(seq)
R> m <- NCOL(seq)
R> nt <- c('A', 'C', 'G', 'T')
R> ctrl <- as.matrix(seq)
R> for(k in seq_len(nsample))
+    ctrl[k, sample(m, 3)] <- sample(nt, 3, replace = TRUE)
R> colnames(ctrl) <- 1:m
R> data <- rbind(data.frame(y = rep('Binding', nsample), seq),
+    data.frame(y = rep('Non-binding', nsample), ctrl))
R> data <- data[sample(NROW(data)), ]
```

We assess the performance of pseudo-likelihood and mean field inferences below using cross-validation:

```
R> ps <- crossVal(y ~ .^2, data = data, method = 'pseudo',
+    lambda = 10^seq(-2, -1, 0.2), verbose = 0)
R> ps

Optimal lambda = 0.02511886
Max. score: 0.8530795

      lambda       AUC       ci1       ci2
1 0.01000000 0.8493230 0.8317931 0.8668530
2 0.01584893 0.8519034 0.8345098 0.8692970
3 0.02511886 0.8530795 0.8357665 0.8703926
4 0.03981072 0.8527357 0.8354177 0.8700538
5 0.06309573 0.8517654 0.8344032 0.8691276
6 0.10000000 0.8501564 0.8327169 0.8675960

R> mf <- crossVal(y ~ .^2, data = data, method = 'mf',
+    eps = seq(0.1, 0.4, 0.1), verbose = 0)
R> mf

Optimal epsilon = 0.2
Max. score: 0.8530829

  epsilon       AUC       ci1       ci2
1     0.1 0.8523296 0.8350451 0.8696140
2     0.2 0.8530829 0.8357946 0.8703712
3     0.3 0.8518778 0.8344548 0.8693008
4     0.4 0.8498872 0.8322967 0.8674777
```

In both cases, there is an optimal, intermediate range of regularization with maximum AUC (Fig. 8). The level of performance attainable with non-interacting models, such as position frequency matrix (Wasserman and Sandelin 2004), corresponds to the $\epsilon = 0$ limit in Fig. 8**b**. The AUC range obtained above is representative of the sensitivity and specificity levels one would get when scanning a genomic segment using a trained model for detection of a binding site to within resolution of ~ 3 base pairs.

4. Summary

We introduced a user-friendly R package **bbl**, implementing general Boltzmann Bayes classifiers applicable to heterogeneous, multifactorial predictor data associated with a discrete multi-class response variable. The currently available R package **BoltzMM** is limited to fitting data into a single fully visible Boltzmann distribution without reference to response variables, and assumes binary predictors. The package **bbl** employs a more general statistical distribution accommodating heterogeneous, factor-valued predictors via Eq. (6), embedding it in a Bayesian classifier to build supervised learning and prediction models. The basic implementation architecture of **bbl** follows those of standard base R packages such as **glm**.

Compared to more widely applied restricted Boltzmann machine algorithms (Hinton 2012), the Boltzmann Bayes model explicitly infers interaction parameters for all pairs of predictors, making it possible to interpret trained models directly, as illustrated in Figs. 2 and 7, the latter using MCMC sampling of predictor distributions. The **bbl** inference is especially suited to data types where a moderate number of unordered features (such as nucleotide sequences) combine to determine class identity, as in transcription factor binding motifs (Section 3.6). Among the two options for inference methods, mean field (`method = 'mf'`) is faster but can become memory intensive for models with a large number of predictors. Pseudo-likelihood maximization (`method = "pseudo"`) is slower but usually provides better performance measured in cross-validation accuracy or AUC.

Computational details

The current version of **bbl** is available at the Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.org/package=bbl. Installation of **bbl** requires the GNU Scientific library https://www.gnu.org/software/gsl installed. The results in this paper were obtained using R 4.4.1. R itself and all packages used are available from the CRAN at https://CRAN.R-project.org and Bioconductor at https://bioconductor.org.

References

Ackley DH, Hinton GE, Sejnowski TJ (1985). "A Learning Algorithm for Boltzmann Machines." *Cognitive Science*, **9**(1), 147–169. doi:10.1016/s0364-0213(85)80012-4.

Besag J (1975). "Statistical Analysis of Non-Lattice Data." *Journal of the Royal Statistical Society D*, **24**(3), 179–195. doi:10.2307/2987782.

Chandler D (1987). *Introduction to Modern Statistical Mechanics*. Oxford, New York.

Friedman J, Hastie T, Tibshirani R (2010). "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software*, **33**(1), 1–22. doi:10.18637/jss.v033.i01.

Friedman J, Hastie T, Tibshirani R, Narasimhan B, Simon N, Qian J (2019). ***glmnet**: Lasso and Elastic-Net Regularized Generalized Linear Models*. R package version 3.0-2, URL https://CRAN.R-project.org/package=glmnet.

Hastie T, Tibshirani R, Friedman J (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd edition. Springer-Verlag, New York. doi:10.1007/978-0-387-84858-7. URL https://web.stanford.edu/~hastie/ElemStatLearn/.

Hendricks P (2015). ***titanic**: Titanic Passenger Survival Data Set*. R package version 0.1.0, URL https://CRAN.R-project.org/package=titanic.

Hinton GE (2012). "A Practical Guide to Training Restricted Boltzmann Machines." In G Montavon, GB Orr, KR Müller (eds.), *Neural Networks: Tricks of the Trade: Second Edition*, pp. 599–619. Springer-Verlag, Berlin, Heidelberg. doi:10.1007/978-3-642-35289-8_32.

Hyvärinen A (2006). "Consistency of Pseudolikelihood Estimation of Fully Visible Boltzmann Machines." *Neural Computation*, **18**(10), 2283–2292. doi:10.1162/neco.2006.18.10.2283.

Jones A, Bagnall J, Nguyen H (2019a). "**BoltzMM**: An R Package for Maximum Pseudolikelihood Estimation of Fully-Visible Boltzmann Machines." *Journal of Open Source Software*, **4**(34), 1193. doi:10.21105/joss.01193.

Jones AT, Nguyen HD, Bagnall JJ (2019b). ***BoltzMM**: Boltzmann Machines with MM Algorithms*. R package version 0.1.4, URL https://CRAN.R-project.org/package=BoltzMM.

Khan A, Fornes O, Stigliani A, Gheorghe M, Castro-Mondragon JA, van der Lee R, Bessy A, Chéneby J, Kulkarni SR, Tan G, Baranasic D, Arenillas DJ, Sandelin A, Vandepoele K, Lenhard B, Ballester B, Wasserman WW, Parcy F, Mathelier A (2018). "JASPAR 2018: Update of the Open-Access Database of Transcription Factor Binding Profiles and Its Web Framework." *Nucleic Acid Research*, **46**(D1), D260–D266. doi:10.1093/nar/gkx1188.

Lecun Y, Bottou L, Bengio Y, Haffner P (1998). "Gradient-Based Learning Applied to Document Recognition." *Proceedings of the IEEE*, **86**(11), 2278–2324. doi:10.1109/5.726791.

Montavon G, Samek W, Müller KR (2018). "Methods for Interpreting and Understanding Deep Neural Networks." *Digital Signal Processing*, **73**, 1–15. doi:10.1016/j.dsp.2017.10.011.

Morcos F, Pagnani A, Lunt B, Bertolino A, Marks DS, Sander C, Zecchina R, Onuchic JN, Hwa T, Weigt M (2011). "Direct-Coupling Analysis of Residue Coevolution Captures Native Contacts across Many Protein Families." *Proceedings of the National Academy of Sciences of the United States of America*, **108**(49), E1293–E1301. doi:10.1073/pnas.1111471108.

Nguyen HC, Zecchina R, Berg J (2017). "Inverse Statistical Problems: From the Inverse Ising Problem to Data Science." *Advances in Physics*, **66**(3), 197–261. doi:10.1080/00018732.2017.1341604.

Nguyen HD, Wood IA (2016a). "Asymptotic Normality of the Maximum Pseudolikelihood Estimator for Fully Visible Boltzmann Machines." *IEEE Transactions on Neural Networks and Learning Systems*, **27**(4), 897–902. doi:10.1109/tnnls.2015.2425898.

Nguyen HD, Wood IA (2016b). "A Block Successive Lower-Bound Maximization Algorithm for the Maximum Pseudo-Likelihood Estimation of Fully Visible Boltzmann Machines." *Neural Computation*, **28**(3), 485–492. doi:10.1162/neco_a_00813.

Pagès H, Aboyoun P, Gentleman R, DebRoy S (2019). ***Biostrings**: Efficient Manipulation of Biological Strings*. R package version 2.52.0, URL https://bioconductor.org/packages/Biostrings.

Philipp M, Rusch T, Hornik K, Strobl C (2018a). "Measuring the Stability of Results From Supervised Statistical Learning." *Journal of Computational and Graphical Statistics*, **27**(4), 685–700. doi:10.1080/10618600.2018.1473779.

Philipp M, Strobl C, Zeileis A, Rusch T, Hornik K (2018b). ***stablelearner**: Stability Assessment of Statistical Learning Methods*. R package version 0.1-1, URL https://CRAN.R-project.org/package=stablelearner.

Robin X, Turck N, Hainard A, Tiberti N, Lisacek F, Sanchez JC, Müller M (2011). "**pROC**: An Open-Source Package for R and S-PLUS to Analyze and Compare ROC Curves." *BMC Bioinformatics*, **12**(1), 77. doi:10.1186/1471-2105-12-77.

Robin X, Turck N, Hainard A, Tiberti N, Lisacek F, Sanchez JC, Müller M, Siegert S, Doering M (2019). ***pROC**: Display and Analyze ROC Curves*. R package version 1.15.3, URL https://CRAN.R-project.org/package=pROC.

Salakhutdinov R, Hinton G (2009). "Deep Boltzmann Machines." In D van Dyk, M Welling (eds.), *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pp. 448–455. PMLR, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA.

Wasserman WW, Sandelin A (2004). "Applied Bioinformatics for the Identification of Regulatory Elements." *Nature Reviews Genetics*, **5**(4), 276–287. doi:10.1038/nrg1315.

Woo HJ, Yu C, Kumar K, Gold B, Reifman J (2016). "Genotype Distribution-Based Inference of Collective Effects in Genome-Wide Association Studies: Insights to Age-Related Macular Degeneration Disease Mechanism." *BMC Genomics*, **17**(1), 695. doi:10.1186/s12864-016-2871-3.

Affiliation:

Jun Woo[1] (*corresponding author*), Jinhua Wang
Institute for Health Informatics
and
Masonic Cancer Center
University of Minnesota
Minneapolis, Minnesota, USA
E-mail: wooh@mskcc.org

[1]Current address: Memorial Sloan Kettering Cancer Center, New York, New York, USA