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 $x$ that can occupy categorical factor levels (Hastie, Tibshirani, and Friedman 2009). Ideally, it would be best to model the joint distribution $P(x, y)$ via maximum likelihood,

$$\hat{\Theta} = \arg \max_\Theta \{ \ln P(x, y|\Theta) \},$$

(1)

to find parameters $\Theta$. Regression-based methods use $P(x, y) = P(y|x)P(x) \approx P(y|x)$. Many rigorous formal results known for regression coefficients facilitate interpretation of their significance. An alternative is to use $P(x, y) = P(x|y)P(y)$ and fit $P(x|y)$. 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 $P(x|y)$, one uses the Bayes’ formula. Examples include linear and quadratic discriminant analyses (Hastie et al. 2009, pp. 106-119) for continuous $x$. For discrete $x$, naive Bayes is the simplest approach, where the covariance among $x$ is ignored via

$$P(x|y) \approx \prod_i P(x_i|y)$$

(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 $x_i = 0, 1$,

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

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., $y = \text{case, control}$ with $G = 2$; $k = 1, \ldots, n$ 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 $x$ with elements $x_i$ and the observed data for an instance $k$ denoted by $x^k$. We assume that predictor variables take discrete factor levels, each with distinct effect on responses, e.g., $x_i = a, t, g, c$ for DNA sequence data. The overall likelihood is

$$L = \sum_k w_k \log P(x^k, y_k) = \sum_y \sum_{k \in \mathcal{K}_y} w_k \log P(x^k, y) \equiv \sum_y L_y,$$

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

$$L_y = \sum_{k \in \mathcal{K}_y} w_k \left[ \log P(x^k|y) + \log P(y) \right] = \sum_{k \in \mathcal{K}_y} w_k \log P(x^k|y) + n_y \log p_y,$$

where $p_y \equiv P(y)$ is the marginal distribution of $y$ and $n_y = \sum_{k \in \mathcal{K}_y} w_k$ 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

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

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

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

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 $a_i = 0, \ldots, L_i - 1$, 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)

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

is then related to Eq. (3) by

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

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

(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):

$$L_y - n_y \ln p_y = \sum_{k \in K_y} w_k \ln P(x^k | y) \approx \sum_{k \in K_y} w_k \sum_i \ln P_i(x^k_i | y, x^k_{j, i}) \equiv \sum_i L_{iy},$$

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

$$P_i(x | y, x^k_{j, i}) = \frac{e^{h_i(y|x|x^k_{j, i})}}{Z_{iy}(x^k_{j, i})},$$

$$Z_{iy}(x^k_{j, i}) = \sum_x e^{h_i(y|x|x^k_{j, i})} = 1 + \sum_{a=1}^{L_i-1} e^{h_i(y|a|x^k_{j, i})},$$

and

$$\tilde{h}_i(y | x^k_{j, i}) = h_i(y | x) + \sum_{j \neq i} J_{ij}^y(x, x_j).$$

Including $L_2$ penalizers ($\lambda_h, \lambda$), $L_{iy}$ in Eq. (10) becomes

$$L_{iy} = \sum_{k \in K_y} w_k \left[ \tilde{h}_i^y(x^k_i | x^k_{j, i}) - \ln Z_{iy}(x^k_{j, i}) \right] - \frac{\lambda_h}{2} \sum_x h_i^y(x)^2 - \frac{\lambda}{2} \sum_{j, x, x'} J_{ij}^y(x, x')^2$$

with first derivatives

$$\frac{\partial L_{iy}/n_y}{\partial h_i^y(x)} = \hat{f}_i^y(x) - \frac{1}{n_y} \sum_{k \in K_y} w_k P_i(x | y, x^k_{j, i}) - \lambda_h h_i^y(x),$$

$$\frac{\partial L_{iy}/n_y}{\partial J_{ij}^y(x, x')} = \hat{f}_{ij}^y(x, x') - \frac{1}{n_y} \sum_{k \in K_y} w_k \mathbb{1}(x^k_j = x') P_i(x | y, x^k_{j, i}) - \lambda J_{ij}^y(x, x'),$$

where

$$\hat{f}_i^y(x) = \frac{1}{n_y} \sum_{k \in K_y} w_k \mathbb{1}(x^k_i = x),$$

$$\hat{f}_{ij}^y(x, x') = \frac{1}{n_y} \sum_{k \in K_y} w_k \mathbb{1}(x^k_i = x) \mathbb{1}(x^k_j = x'),$$

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,

$$J_{ij}(x, x') = J_{ji}(x', x).$$
After pseudo-likelihood inference, therefore, the interaction parameters are symmetrized as follows:
\[
J_{ij}(x, x') \leftarrow \frac{1}{2} \left[ J_{ij}(x, x') + J_{ji}(x', x) \right].
\]

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)
\[
\hat{J}_{ij}^y(x, x') = -\left[ C^y \right]_{ij}^{-1}(x, x'),
\]
i.e., negative inverse of the covariance matrix,
\[
C_{ij}^y(x, x') = \hat{f}_{ij}(x, x') - \hat{f}_i(x) \hat{f}_j(x').
\]

Equation (19) can be interpreted as treating discrete \( x \) as if it were multivariate normal: Eq. (6) would then be the counterpart of the multivariate normal probability density function with \(- \hat{J}_{ij}^y(x, x')\) 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 \( C^y \) between non-interacting (naive Bayes) \((\epsilon = 0)\) and fully interacting limits \((\epsilon = 1)\):
\[
C^y \leftarrow \tilde{C}^y = (1 - \epsilon) \frac{\text{Tr} C^y}{\text{Tr} I} + \epsilon C^y,
\]
where \( I \) is the identity matrix of the same dimension as \( C^y \). The parameter \( \epsilon \) 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),
\[
\tilde{h}_i^y(x) = \tilde{h}_i^y(x) - \sum_{j \neq i} \sum_{x'} \hat{J}_{ij}^y(x, x') \hat{f}_j(x'),
\]
and
\[
\tilde{h}_i^y(x) = \ln \left[ \hat{f}_i^y(x) / \hat{f}_i^y(0) \right],
\]
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
\[
\hat{f}_i^y(x) = \frac{e^{\tilde{h}_i^y(x)}}{Z_{iy}} = \hat{f}_i^y(0) e^{\tilde{h}_i^y(x)}
\]
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),

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

(25)

and (Woo et al. 2016)

$$L_y - n_y \ln p_y = \sum_{k \in K_y} w_k \ln P(x^k | y) = n_y \sum_i \hat{f}_i^{(y)}(x) \ln \hat{f}_i^{(y)}(x).$$

(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

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

(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

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

(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:

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

(29)

where

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

(30)

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

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

(31)
where

\[ \alpha = \ln \frac{Z_0 p_1}{Z_1 p_0}, \]

\[ \beta_i(x) = h_i^{(1)}(x) - h_i^{(0)}(x), \]

\[ \gamma_{ij}(x, x') = J_{ij}^{(1)}(x, x') - J_{ij}^{(0)}(x, x'). \]  

(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 \( P(y|x) \) simpler than Eq. (29) exists for data with more than two groups.

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

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

(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:

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

(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

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

(35)

where

\[ \hat{y}(x) = \arg \max_y P(y|x). \]  

(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:

\[
R> \text{titanic} <- \text{as.data.frame(Titanic)} \\
R> \text{titanic}
\]
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
R> library('bbl')
R> titanic_raw <- freq2raw(data = titanic, freq = Freq)
R> head(titanic_raw)

     Class      Sex    Age    Survived Survived
   1 3rd Male Child  No 0
   2 3rd 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
```
We train a logistic regression model using glm:

```R
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
Class2nd:SexFemale -0.0680088  0.6711978  -0.101324  1.000000e+00
Class3rd:SexFemale -2.7999478  0.5687464  -4.923016  5.000000e-07
ClassCrew:SexFemale -1.1360791  0.8204849  -1.384643  1.000000e+00
Class2nd:AgeAdult -1.9304713  0.5492945  -3.500000  9.519086e-04
Class3rd:AgeAdult  3.3562933  0.7397809   4.570000  2.988760e-06
```

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

```
Estimate  Std. Error    z value
(Intercept)  14.77919784  437.9677046  0.033744949
Class2nd -0.86597950  549.2945250 -0.001574553
Class3rd -15.76959654  437.9678237 -0.035604023
ClassCrew -0.52214898  0.1808848  -2.900270097
SexFemale  3.59619056  0.2440256  14.779197841
AgeAdult -15.50683120  437.9677339 -0.035406332
Class2nd:SexFemale -0.0680088  0.6711978  -0.101324634
Class3rd:SexFemale -2.7999478  0.5687464  -4.923016541
ClassCrew:SexFemale -1.1360791  0.8204849  -1.384643864
Class2nd:AgeAdult -1.9304713  0.5492945  -3.500000000
Class3rd:AgeAdult  3.3562933  0.7397809   4.570000000
```
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)
nsample <- NROW(titanic_raw)
flag <- rep(TRUE, nsample)
flag[sample(nsample, nsample/2)] <- FALSE
dtrain <- titanic_raw[flag,]
dtest <- titanic_raw[!flag,]

glm model with interactions and make prediction on the test data:

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

```
[1] 0.7718182
```

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

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
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
R> head(xdat)
```

<table>
<thead>
<tr>
<th>Class2nd</th>
<th>Class3rd</th>
<th>ClassCrew</th>
<th>SexFemale</th>
<th>AgeAdult</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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) \( \beta(x) \) 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 \( \beta(x) \) 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 \texttt{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 \texttt{glm}, \texttt{bbl} uses formula input to train an \texttt{S3} object of class \texttt{bbl}:

\begin{verbatim}
R> bfit0 <- bbl(Survived ~ Class + Sex + Age, data = titanic, weights = Freq, + prior.count = 0)
\end{verbatim}

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 \texttt{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 \texttt{print} method on \texttt{bbl} shows the structure of model and (subsets) of inferred parameters:

\begin{verbatim}
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:
\end{verbatim}
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:

\texttt{h\_Class:}

<table>
<thead>
<tr>
<th></th>
<th>2nd</th>
<th>3rd</th>
<th>Crew</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>0.3139728</td>
<td>1.4650752</td>
<td>1.7077243</td>
</tr>
<tr>
<td>Yes</td>
<td>-0.5425214</td>
<td>-0.1314224</td>
<td>0.0433803</td>
</tr>
<tr>
<td>pooled</td>
<td>-0.1313360</td>
<td>0.7757901</td>
<td>1.0017625</td>
</tr>
</tbody>
</table>

\texttt{chisq = 180.9014, df = 3, Pr(>chisq) = 5.633919e-39}

\texttt{h\_Sex:}

\texttt{Male}

|        | 2.3818949 |
| No     | 0.06472019 |
| pooled | 1.30372186 |

\texttt{chisq = 434.4688, df = 1, Pr(>chisq) = 1.730842e-96}

\texttt{h\_Age:}

\texttt{Child}

|        | -3.319765 |
| No     | -2.440056 |
| pooled | -2.954528 |

\texttt{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 $h_i(y) = h_{i}(a)$. 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 \texttt{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
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 -0.8576762 2.4200603

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

(Intercept) Class2nd Class3rd ClassCrew SexFemale Age.Adult
0.6853195 -1.0180950 -1.7777622 -0.8576762 2.4200603
-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
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 2.993540 1.569273
  dh_[Class]^(Yes):
    2nd    3rd    Crew  
-0.1028469 -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
```
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 under-represented (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 = 'ts1prob')
R> head(pr)

<table>
<thead>
<tr>
<th></th>
<th>No</th>
<th>Yes</th>
<th>yhat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.8092692</td>
<td>0.1907308</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>0.8092692</td>
<td>0.1907308</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>0.8092692</td>
<td>0.1907308</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>0.8092692</td>
<td>0.1907308</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>0.8092692</td>
<td>0.1907308</td>
<td>No</td>
</tr>
<tr>
<td>6</td>
<td>0.8092692</td>
<td>0.1907308</td>
<td>No</td>
</tr>
</tbody>
</table>

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 $\texttt{bb1}$ object displays bias (top) and interaction parameters (bottom). All parameters are offset by their pooled (single-group) values.

One can do cross-validation applied to $\texttt{dtrain}$ data, dividing it into $n_{\text{fold}} = 5$ train/validation subsets of 4:1 proportion, and aggregating predictions for validation sets using the trained model:

\begin{verbatim}
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
\end{verbatim}
Figure 3: Cross-validation run of Titanic data in bbl.

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
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 `bb1` fit compared to regression is the availability of predictor distributions in each response group, $P(x|y)$, 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 $x$ 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 `bb1` 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 `bb1` 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)
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 (\( L_i = L = 3 \)).

\[ R > \text{xi} \leftarrow \text{sample_xi}(\text{nsample} = 10000, \text{predictors} = \text{predictors}, h = \text{par}$h$, + \quad J = \text{par}$J$, code_out = \text{TRUE}) \]
\[ R > \text{head(xi)} \]

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. 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`:

\[ R > \text{fit} \leftarrow \text{mlestimate}(\text{xi} = \text{xi}, \text{method} = \text{'pseudo'}, \text{lambda} = 0) \]

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: \( a_i = 0, \ldots, L_i - 1 \). 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:

\[ R > \text{nt} \leftarrow \text{c('a', 'c', 'g', 't')} \]
\[ R > \text{set.seed(135)} \]
\[ R > \text{for(i in 1:m) predictors[[i]]} \leftarrow \text{nt} \]
\[ R > \text{names(predictors)} \leftarrow \text{paste0('v', 1:m)} \]
\[ R > \text{par} \leftarrow \text{list()} \]
\[ R > \text{par[[1]]} \leftarrow \text{randompar(predictors)} \]
\[ R > \text{par[[2]]} \leftarrow \text{randompar(predictors}, h0 = 0.1, J0 = 0.1) \]
\[ R > \text{dat} \leftarrow \text{randomsamp(predictors, response = c('ctrl', 'case'), par = par, + \quad nsample = 1000)} \]
Figure 4: Comparison of true parameters and those inferred from pseudo-likelihood Boltzmann Bayes inference. See the text for conditions.

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

\begin{verbatim}
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

<table>
<thead>
<tr>
<th>epsilon</th>
<th>AUC</th>
<th>ci1</th>
<th>ci2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0</td>
<td>0.7849546</td>
<td>0.7568076</td>
</tr>
<tr>
<td>2</td>
<td>0.1</td>
<td>0.8392593</td>
<td>0.8149947</td>
</tr>
<tr>
<td>3</td>
<td>0.2</td>
<td>0.8610941</td>
<td>0.8386831</td>
</tr>
<tr>
<td>4</td>
<td>0.3</td>
<td>0.8708767</td>
<td>0.8493991</td>
</tr>
<tr>
<td>5</td>
<td>0.4</td>
<td>0.8773411</td>
<td>0.8565066</td>
</tr>
<tr>
<td>6</td>
<td>0.5</td>
<td>0.8812357</td>
<td>0.8608067</td>
</tr>
<tr>
<td>7</td>
<td>0.6</td>
<td>0.8831850</td>
<td>0.8629906</td>
</tr>
<tr>
<td>8</td>
<td>0.7</td>
<td>0.8845219</td>
<td>0.8644845</td>
</tr>
<tr>
<td>9</td>
<td>0.8</td>
<td>0.8840456</td>
<td>0.8639740</td>
</tr>
<tr>
<td>10</td>
<td>0.9</td>
<td>0.8815880</td>
<td>0.8612527</td>
</tr>
<tr>
<td>11</td>
<td>1.0</td>
<td>0.8724978</td>
<td>0.8511909</td>
</tr>
</tbody>
</table>
\end{verbatim}

Here, \texttt{bbl} is called inside \texttt{crossVal} as before but with \texttt{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 $\epsilon$. (b-d) Comparison of true and inferred parameters under three $\epsilon$ values. Best fit is achieved when AUC is maximum.

As shown in Fig. 5a, 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 $\epsilon$ and examine the fit:

```R
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 = 'ts1mf', eps = eps[i],
+                     verbose = 0)
```

Figure 5b-d compares the three inferred parameter sets ($\text{coef(fit[[i]])}$h, $\text{coef(fit[[i]])}$J) with the true values ($\text{par[[iy]]}h$, $\text{par[[iy]]}J$). As $\epsilon$ 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. 5c), as suggested by cross-validation AUC in Fig. 5a.

### 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 ($x_i = a, c, g, t$). 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'))
  +     BiocManager::install('Biostrings')
  + }
R> aa <- Biostrings::DNAString(paste(t(dat), collapse = ''))
R> aa

6000-letter DNAString object
seq: TAGTCTAACGGAACCTGGCGATTATACTTG...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:
\begin{verbatim}
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

<table>
<thead>
<tr>
<th>lambda</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.001</td>
<td>0.9195</td>
</tr>
<tr>
<td>0.003</td>
<td>0.9195</td>
</tr>
<tr>
<td>0.010</td>
<td>0.9875</td>
</tr>
<tr>
<td>0.032</td>
<td>0.9875</td>
</tr>
<tr>
<td>0.100</td>
<td>0.9925</td>
</tr>
<tr>
<td>0.316</td>
<td>1.0000</td>
</tr>
<tr>
<td>1.000</td>
<td>0.9930</td>
</tr>
<tr>
<td>3.162</td>
<td>0.9770</td>
</tr>
<tr>
<td>10.0</td>
<td>0.9770</td>
</tr>
</tbody>
</table>

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 $\lambda$. In contrast to Section 3.2, we do not refit the model under this $\lambda$ 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
\end{verbatim}
3.5. Image data

We next consider learning examples with data sets containing predictors numbering \( \sim 100 \) 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
R> dat0 <- read.csv(system.file('extdata/mnist_train.csv', package = 'bb1'))
R> dat <- removeConst(dat0)
R> dat[1:5, 1:10]
```

<table>
<thead>
<tr>
<th></th>
<th>X40</th>
<th>X41</th>
<th>X45</th>
<th>X46</th>
<th>X47</th>
<th>X64</th>
<th>X65</th>
<th>X66</th>
<th>X67</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

```r
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.
**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 $\epsilon$ values, one can obtain the profile shown in Fig. 6. The jump in performance under $\epsilon^* \sim 0.05$ over $\epsilon \rightarrow 0$ (naive Bayes) limit gives a measure of interaction effects. The relatively small value of $\epsilon^*$ at the optimal condition, compared to e.g., Fig. 5a, reflects the sparseness of image data.

We now retrain the model without cross-validation under $\epsilon^*$ 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)
```
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, Baranasić, 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 T T C

Figure 7: Maximum probability configurations of digits (0, ⋯, 9) estimated from bbl fit coefficients using Gibbs sampling.
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
R> dim(seq)
[1] 948 12
```

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

```r
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:
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. 8b. 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 $\sim$3 base pairs.

4. Summary

We introduced a user-friendly R package \texttt{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 \texttt{BoltzMM} is limited to fitting data into a single fully visible Boltzmann distribution without reference to response variables, and assumes binary predictors. The package \texttt{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 \texttt{bbl} follows those of standard base R packages such as \texttt{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 4.1.2. R itself and all packages used are available from the CRAN at https://CRAN.R-project.org and Bioconductor at https://bioconductor.org.

**References**


Affiliation:

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

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