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 $P(\mathbf{x}, y) = P(y|\mathbf{x})P(\mathbf{x}) \simeq P(y|\mathbf{x})$. Their advantages include the wealth of information provided for significance of fit coefficients from rigorous formal results. 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

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

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 2016; Nguyen and Wood 2016), 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 (BB) 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 Sec. 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 $K \geq 2$ groups; e.g. $y = \texttt{case}, \texttt{control}$; $k = 1, \cdots, n$ denotes sample (or configuration) index. We also introduce frequency 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 data takes the form of one entry per obserbation, $w_k = 1$ 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 overall likelihood is

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

where the second summation is over all k for which $y_k = y$. The inference is first performed for each group y separately, maximizing L_y given by

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

where $p_y \equiv P(y)$ is the marginal distribution of y and $n_y = \sum_k w_k$ is the size of group y.

In parametrizing the first term in Eq. (5), 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 group-specific predictor distribution can then be 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.* 2019b). Its distribution (Jones *et al.* 2019a)

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

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):

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

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

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

with first derivatives

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

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

where

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

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

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 p.d.f. 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)

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

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

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

The statistic $q_i \sim \chi^2$ with d.f. $= (K-1)(L_i - 1)$.

2.5. Classification

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

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y)p_y}{\sum_{y'} P(\mathbf{x}|y')p_{y'}} = \frac{1}{1 + \sum_{y'\neq y} P(\mathbf{x}|y')p_{y'}/P(\mathbf{x}|y)p_y} = \frac{1}{1 + e^{-F_y(\mathbf{x})}}, \tag{28}$$

where

$$F_y(\mathbf{x}) = \ln\left[\frac{P(\mathbf{x}|y)p_y}{\sum_{y'\neq y} P(\mathbf{x}|y')p_{y'}}\right]. \tag{29}$$

For binary response coded as $y = 0, 1$, Eq. (29) reduces to

$$\begin{aligned} F_1(\mathbf{x}) &= \ln P(\mathbf{x}|y=1) - \ln P(\mathbf{x}|y=0) + \ln(p_1/p_0) \\ &= \sum_i \left[h_i^{(1)}(x_i) - h_i^{(0)}(x_i)\right] + \sum_{i<j} \left[J_{ij}^{(1)}(x_i, x_j) - J_{ij}^{(0)}(x_i, x_j)\right] + \ln\frac{Z_0 p_1}{Z_1 p_0}. \end{aligned} \tag{30}$$

Therefore, if $J_{ij}^{(y)}(x, x') = 0$ (naive Bayes), Eq. (28) 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. (28) exists for data with more than two groups.

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

$$\ln Z_y = \frac{1}{n_y}\sum_{k\in y}\sum_i \ln\left\{\sum_x \left[e^{h_i^{(y)}(x) + \sum_{j\neq i} J_{ij}(x, x_j^k)/2}\right]\right\}, \tag{31}$$

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:

$$\ln Z_y = -\ln \hat{f}^{(y)}(0) - \frac{1}{2}\sum_{i\neq j}\sum_{x,x'} J_{ij}(x,x')\hat{f}_i(x)\hat{f}_j(x'). \tag{32}$$

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

$$s = \frac{1}{n}\sum_k \mathbb{1}\left[\hat{y}(\mathbf{x}^k) = y_k\right], \tag{33}$$

where

$$\hat{y}(\mathbf{x}) = \arg\max_y P(y|\mathbf{x}). \tag{34}$$

If response is binary, the score defined by Eq. (33) is sensitive to marginal distributions of the two groups via Eq. (30). The area under curve (AUC) of receiver operating characteristic is a more robust performance measure independent of probability cutoff. In **bbl**, the prediction score given by Eqs. (33) and (34) is used in general with the option to use AUC for binary response using R package **pROC** (Robin, Turck, Hainard, Tiberti, Lisacek, Sanchez, and MÃijller 2011).

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
```

```
R> freq <- titanic$Freq
R> titanic <- titanic[,1:4]
```

Input data can either be of the form above with unique combinations of predictors in each row along with frequency 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> gfit0

Call:  glm(formula = Survived ~ Class + Sex + Age, family = binomial(),
    data = titanic, weights = freq)

Coefficients:
(Intercept)     Class2nd     Class3rd    ClassCrew    SexFemale
     0.6853      -1.0181      -1.7778      -0.8577       2.4201
   AgeAdult
    -1.0615

Degrees of Freedom: 23 Total (i.e. Null);  18 Residual
Null Deviance:            2769
Residual Deviance: 2210         AIC: 2222

R> summary(gfit0)

Call:
glm(formula = Survived ~ Class + Sex + Age, family = binomial(),
    data = titanic, weights = freq)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-18.505   -4.247    0.000    4.747   23.915

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   0.6853     0.2730   2.510   0.0121 *
Class2nd     -1.0181     0.1960  -5.194 2.05e-07 ***
Class3rd     -1.7778     0.1716 -10.362  < 2e-16 ***
ClassCrew    -0.8577     0.1573  -5.451 5.00e-08 ***
SexFemale     2.4201     0.1404  17.236  < 2e-16 ***
AgeAdult     -1.0615     0.2440  -4.350 1.36e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2769.5  on 23  degrees of freedom
Residual deviance: 2210.1  on 18  degrees of freedom
AIC: 2222.1

Number of Fisher Scoring iterations: 5
```

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> summary(gfit1)

Call:
glm(formula = Survived ~ (Class + Sex + Age)^2, family = binomial(),
    data = titanic, weights = freq)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-18.375   -4.565    0.000    3.286   24.014

Coefficients: (1 not defined because of singularities)
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)          14.77920  437.96770   0.034  0.97308
Class2nd              0.26021  549.29444   0.000  0.99962
Class3rd            -15.76960  437.96782  -0.036  0.97128
ClassCrew            -0.52215    0.18088  -2.887  0.00389 **
SexFemale             3.59619    0.74781   4.809 1.52e-06 ***
AgeAdult            -15.50683  437.96773  -0.035  0.97176
Class2nd:SexFemale   -0.06801    0.67120  -0.101  0.91929
Class3rd:SexFemale   -2.79995    0.56875  -4.923 8.52e-07 ***
ClassCrew:SexFemale  -1.13608    0.82048  -1.385  0.16616
Class2nd:AgeAdult    -1.93047  549.29453  -0.004  0.99720
Class3rd:AgeAdult    14.85629  437.96787   0.034  0.97294
ClassCrew:AgeAdult         NA         NA      NA       NA
SexFemale:AgeAdult    0.68679    0.52541   1.307  0.19116
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2769.5  on 23  degrees of freedom
Residual deviance: 2097.5  on 12  degrees of freedom
AIC: 2121.5

Number of Fisher Scoring iterations: 13
```

The summary of interacting model suggests that class status association arose from the linear effect of being `Crew` and interaction effect of `Female` `3rd`-class passengers.

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 `glm` model with interactions, omitting interaction `Class:Age` because it was rank-deficient, and make prediction on 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
```

3.2. Boltzmann Bayes learning

The logistic regression shown in the previous subsection 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 Sec. 3.1, the survival of Titanic passengers was seen to be associated with being `Crew` members and `Female 3rd`-class passengers. The corresponding linear regression coefficients, which have the same functional form as in Eq. (30), are measures of the *difference* in enrichment in the two response groups, while the two terms whose difference yielded the coefficient 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 BB 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 whose difference corresponds to the linear regression outcome.

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

```
R> bfit0 <- bbl(Survived ~ Class + Sex + Age, data=titanic, freq=freq)
```

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 `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, freq = 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
0.4446497 0.6881264 0.7047726

dh_[Class]^(Yes):
       2nd        3rd       Crew
-0.4104076 -0.9066249 -0.9579479

dh_[Sex]^(No):
    Male
1.075354

dh_[Sex]^(Yes):
     Male
-1.238318

dh_[Age]^(No):
     Child
-0.3603528

dh_[Age]^(Yes):
    Child
0.5181036
```

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, freq = freq)
 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      0.3134216  1.4635015 1.70604860
Yes    -0.5416357 -0.1312497 0.04332808
pooled -0.1312281  0.7753752 1.00127597
chisq = 180.3452, df = 3, Pr(>chisq) = 7.428894e-39

h_Sex:
             Male
No     2.37830103
Yes    0.06462923
pooled 1.30294740
chisq = 433.2369, df = 1, Pr(>chisq) = 3.209119e-96

h_Age:
           Child
No     -3.310543
Yes    -2.432087
pooled -2.950190
chisq = 17.09504, df = 1, Pr(>chisq) = 3.555496e-05
```

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, which clearly indicate the enrichment of lower `Class` status and `Male` among non-survivors. `Child` are few 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.

We now fit an interacting model using **bbl**:

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

Call:
 bbl(formula = Survived ~ Class * Sex + Sex * Age, data = titanic,
    freq = 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
1.486686 2.970870 1.554066

dh_[Class]^(Yes):
       2nd        3rd       Crew
-0.1035553 -0.7508587 -0.1143729

dh_[Sex]^(No):
    Male
3.154356

dh_[Sex]^(Yes):
    Male
-1.07586

dh_[Age]^(No):
    Child
0.3864377

dh_[Age]^(Yes):
    Child
-0.177389


dJ_[Class,Sex]^(No):
          Male
2nd  -1.232563
3rd  -2.720835
Crew -1.379435

dJ_[Class,Sex]^(Yes):
            Male
2nd  -0.90342729
3rd  -0.01467013
Crew -0.30251859

dJ_[Sex,Age]^(No):
          Child
Male -0.4941875
```

```
dJ_[Sex,Age]^(Yes):
        Child
Male 1.167093
```

The parameters printed include those for interactions. The `plot` method shows a barplot of bias parameters and a heatmap of interaction parameters (Fig. 1).

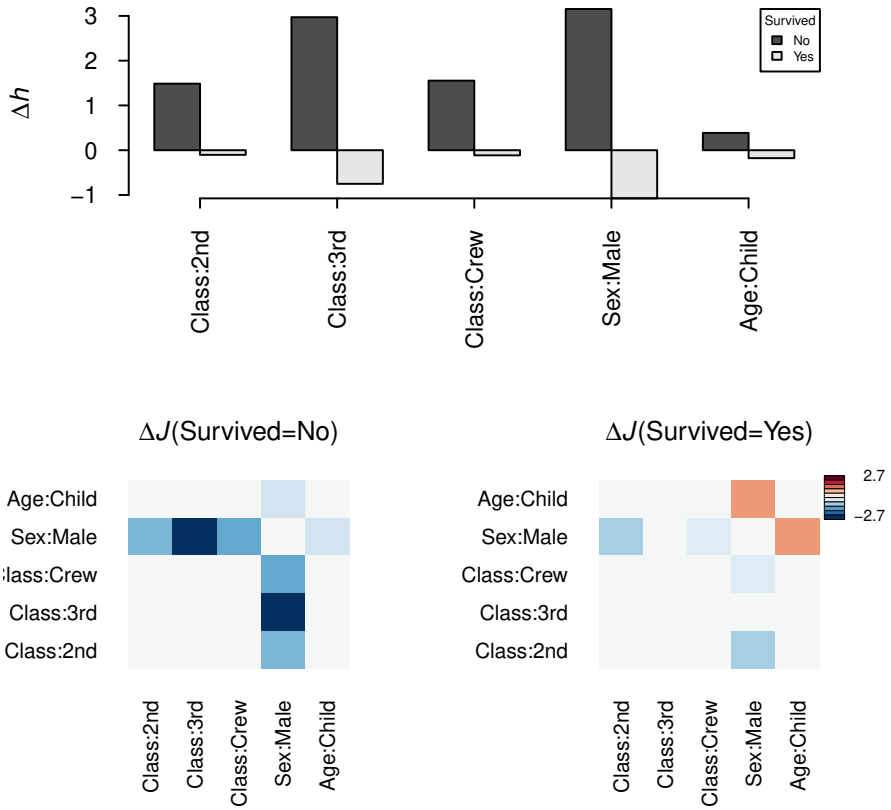

Figure 1: Plot of `bbl` object displays bias (top) and interaction parameters (bottom). All parameters are offset by their pooled (singe-group) values.

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, logit=FALSE)
R> head(pr)
```

```
         No       Yes yhat
1 0.8145593 0.1854407   No
```

```
2 0.8145593 0.1854407   No
3 0.8145593 0.1854407   No
4 0.8145593 0.1854407   No
5 0.8145593 0.1854407   No
6 0.8145593 0.1854407   No
```

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

```
Area under the curve: 0.7707
```

Here, Eq. (28) 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.

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 ~ Class*Sex + Sex*Age, data=dtrain,
+                 method='pseudo', lambda=10^seq(-5,-2,0.2), verbose=0)
R> cv
```

```
Optimal lambda = 0.0001584893
Max. score: 0.7226426

         lambda       auc
1  1.000000e-05 0.6566096
2  1.584893e-05 0.6877020
3  2.511886e-05 0.7016226
4  3.981072e-05 0.6987384
5  6.309573e-05 0.6578960
6  1.000000e-04 0.7040212
7  1.584893e-04 0.7226426
8  2.511886e-04 0.7070723
9  3.981072e-04 0.7050073
10 6.309573e-04 0.7114523
11 1.000000e-03 0.6912610
12 1.584893e-03 0.6770809
13 2.511886e-03 0.7081696
14 3.981072e-03 0.6605393
15 6.309573e-03 0.6719538
16 1.000000e-02 0.6423405
```

```
R> plot(cv, mar=c(4,4,3,3), tck=-0.04, las=1, ylab='AUC', bty='n')
```

It returns an object with a `data.frame` of AUCs for multiple `lambda` values and optimal values with maximum AUC. We use this information to make prediction as follows:

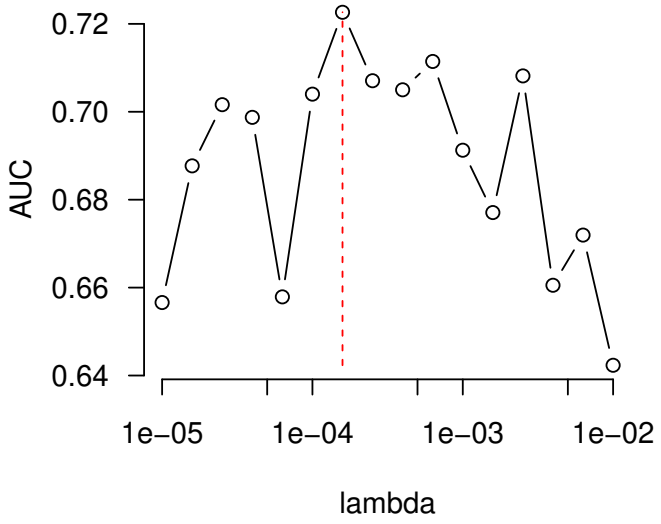

Figure 2: Cross-validation run of `Titanic` data in `bbl`.

```
R> model <- bbl(Survived ~ Class*Sex + Sex*Age, data=dtrain, lambda=cv$regstar)
R> pr2 <- predict(model, newdata=dtest)
R> bscore <- mean(dtest$Survived==pr2$yhat)
R> bscore

[1] 0.79

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

Area under the curve: 0.7707
```

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.

3.3. Simulated data

We next demonstrate the reliability of **bbl** inference using simulated data.

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

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: 42 iterations, likelihood = 0.718101
  Predictor 2: 32 iterations, likelihood = 0.96174
  Predictor 3: 45 iterations, likelihood = 0.963491
  Predictor 4: 46 iterations, likelihood = 0.923876
  Predictor 5: 45 iterations, likelihood = 0.989111

R> names(fit)

[1] "h"   "J"   "lkh" "lz"
```

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 3 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)
```

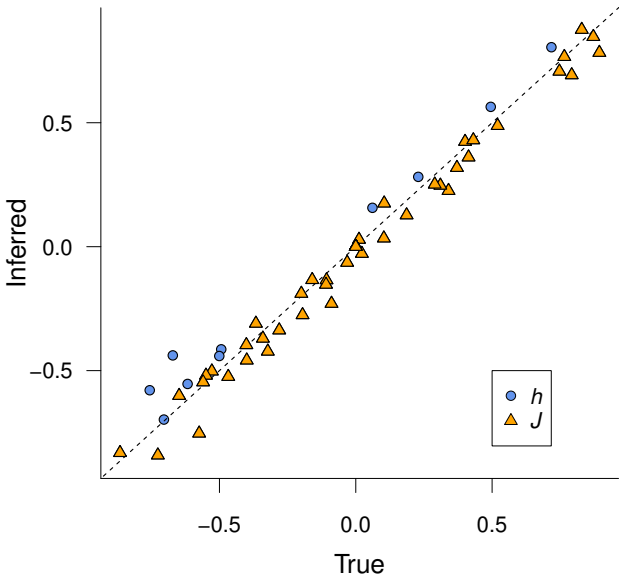

Figure 3: Comparison of true parameters and those inferred from pseudo-likelihood BB inference. See the text for conditions.

```
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)
```

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
1      0.0 0.7849546
2      0.1 0.8392593
3      0.2 0.8610941
4      0.3 0.8708767
5      0.4 0.8773411
6      0.5 0.8812357
7      0.6 0.8831850
8      0.7 0.8845219
9      0.8 0.8840456
```

```
10      0.9 0.8815880
11      1.0 0.8724978
```

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

As shown in Fig. 4**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)
```

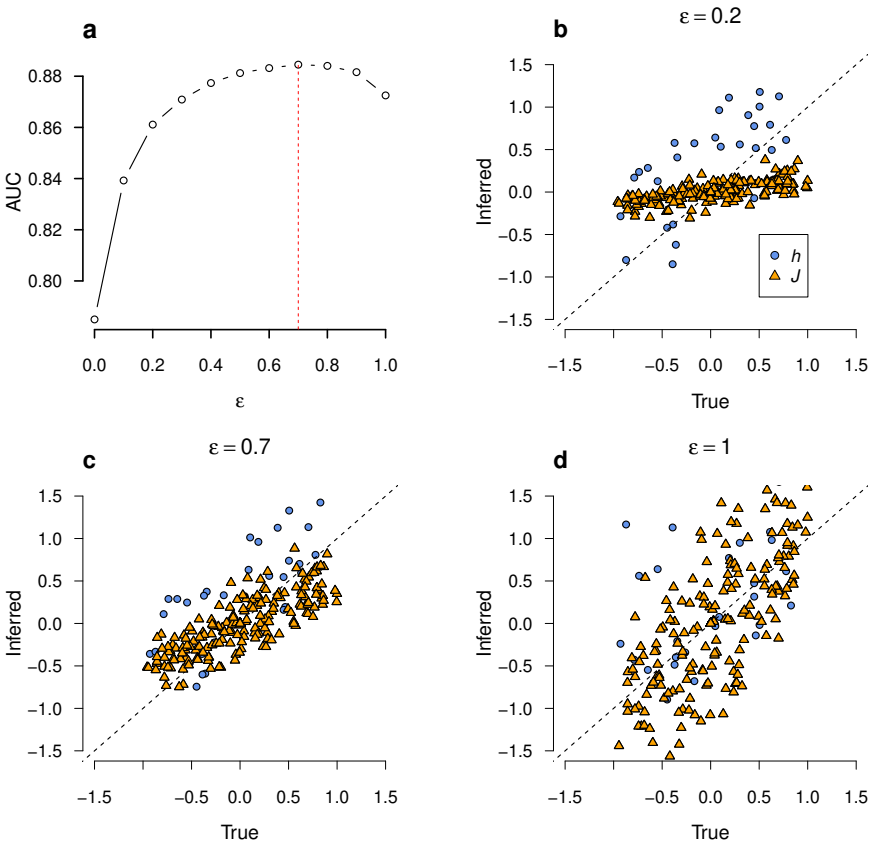

Figure 4: Regularized mean field inference using simulated data. (**a**) Cross-validation AUC with respect to regularization parameter (**b-d**) Comparison of true and inferred parameters under three Best fit is achieved when AUC is maximum.

Figure 4**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. 4**c**), as suggested by cross-validation AUC in Fig. 4**a**.

3.4. Genetic code

We consider a different learning task example with a much larger space of response groups, namely those of amino acides ($K = 21$, 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 discover this genetic code from data.

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

  6000-letter "DNAString" instance
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 score used is Eq. (33). The class `cv.bbl` extends `bbl` and stores the model with the optimal λ. In contrast to Sec. **??**, we do not refit the model under this λ because the score 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> score <- mean(ap==p$yhat)
R> score

[1] 1
```

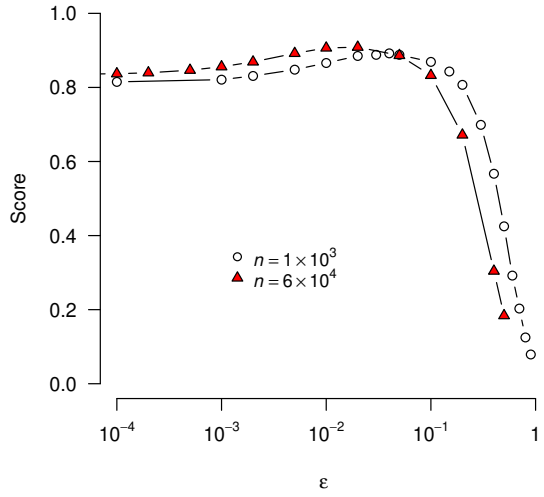

Figure 5: Cross-validation of BB inference on MNIST data using mean field Sample sizes are for down-sampled example and full data sets, respectively.

The trained model has perfect prediction score 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 the MNIST data set (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> dat <- read.csv(system.file('extdata/mnist_train.csv',package='bbl'))
R> dat <- removeConst(dat)
R> dat[1:5,1:10]
R> cv <- crossVal(y ~ .^2, data=dat, method='mf', eps=0.04)
```

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.

The above run will take a few minutes. By feeding a vector of ϵ values, one can obtain the profile shown in Fig. 5 (white symbols). The jump in performance under $\epsilon^* \sim 0.04$ 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. 4**a**, reflects the sparseness of image data.

We now retrain the model without cross-validation under ϵ^* and classify test set (also down-sampled to $n = 500$) images:

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 ($\epsilon = 0.05$)	8.4	**bbl**

Table 1: Performance comparison of BB inference and other models on MNIST data set. BB, Boltzmann Bayes; NN, neural network; RBM, restricted Boltzmann machine.

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

Since `mnist` dropped a subset of original predictors, the test data must be filtered accordingly. Note the increase in test score compared to cross-validation score because of the use of full training data. Set `progress.bar = TRUE` to monitor the progress in a slow `predict` run.

We performed similar cross-validation and test analyses of the full MNIST data (training $n = 60{,}000$ and test $n = 10{,}000$; Fig. 5, red symbols) and obtained the test score of 0.916 (classification error rate 8.4%), which compares favorably with some of the large-scale neural network algorithms (Table. 1).

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 from 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
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: 0.8530206

      lambda       auc
1 0.01000000 0.8492985
2 0.01584893 0.8519268
3 0.02511886 0.8530206
4 0.03981072 0.8527557
5 0.06309573 0.8517421
6 0.10000000 0.8501086

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
1     0.1 0.8523296
```

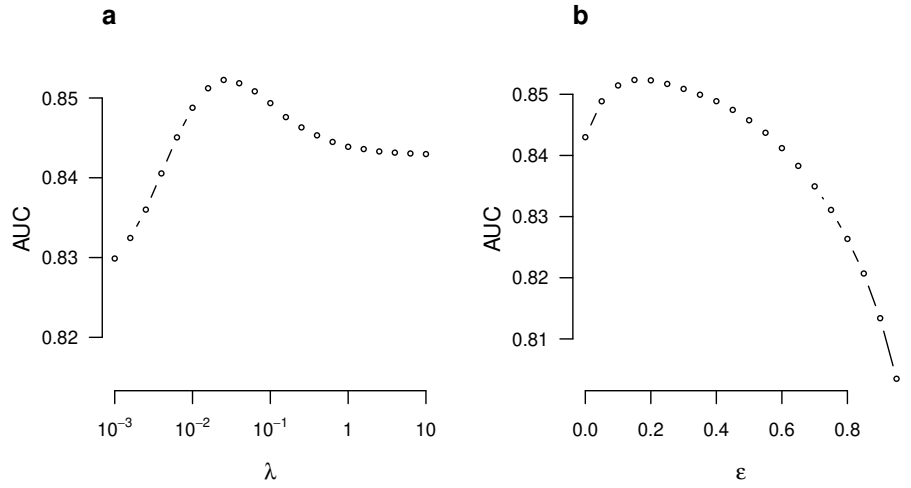

Figure 6: Cross-validation of transcription factor binding motif model using **bbl** with control sequences generated by 3 nucleotide mutations. Data set is from Khan *et al.* (2018) (sample ID MA0014.3; see text). (**a**) Pseudo-likelihood and (**b**) mean field inferences.

```
2     0.2 0.8530829
3     0.3 0.8518778
4     0.4 0.8498872
```

In both cases, there is an optimal, intermediate range of regularization with maximum AUC (Fig. 6). 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. 6**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.

We analyzed 684 data sets with at least 30 sequences from JASPAR database of varying sample sizes and segment lengths with the same protocol. Differences between fully optimized AUC scores and those from non-interacting models (naive Bayes) were most pronounced above the intermediate range of AUC, and were independent of segment lengths (Fig. 7**a**). Pseudo-likelihood results had better scores compared to mean field on avarge (Fig. 7**b**).

4. Summary

We introduced a user-friendly R package **bbl**, implementing general BB 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 **bbl** follows those of standard base R packages such as **glm**.

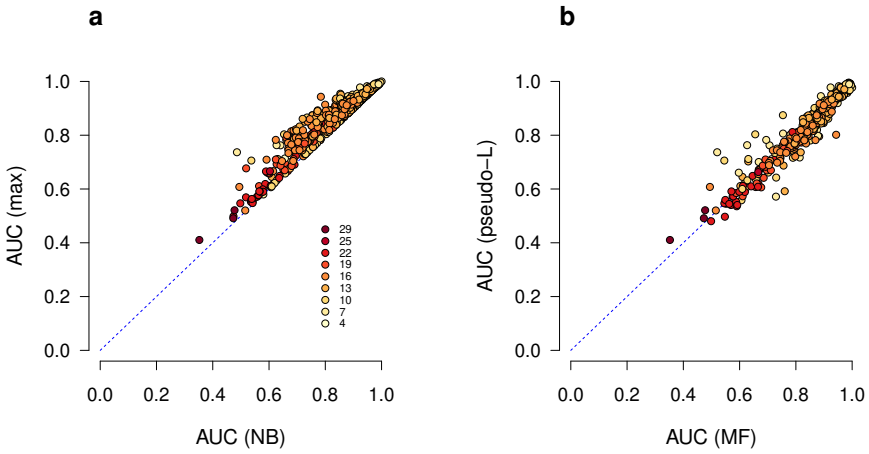

Figure 7: AUC scores of **bbl** model trained on 856 transcription factor binding site sequence data sets from JASPAR (Khan *et al.* 2018) under the same protocol as in Fig. 6. (**a**) Comparison of naive Bayes (NB; mean field with $\epsilon = 0$) and full mean field (MF) results. (**b**) Comparison of mean field (MF) and pseudo-likelihood maximization (pseudo-L) scores. The symbol colors show the segment length of each binding site data (color-map in **a**).

Compared to more widely applied restricted Boltzmann machine algorithms (Hinton 2012), the BB model explicitly infers interaction parameters for all pairs of predictors, making it possible to interpret trained models directly, as illustrated in Fig. 1. Tests on MNIST suggest performance scores similar to other deep layer neural network models in classification tests. However, BB 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 (Sec. 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 generally provides better performance.

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 3.6.1. R itself and all packages used are available from the CRAN at https://CRAN.R-project.org/ and at 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. ISSN 0364-0213. doi:https:

//doi.org/10.1016/S0364-0213(85)80012-4. URL http://www.sciencedirect.com/science/article/pii/S0364021385800124.

Besag J (1975). "Statistical Analysis of Non-Lattice Data." *Journal of the Royal Statistical Society. Series D (The Statistician)*, **24**(3), 179–195. ISSN 00390526, 14679884.

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

Hastie T, Tibshirani R, Friedman J (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd edition. New York. URL https://web.stanford.edu/~hastie/ElemStatLearn/.

Hinton GE (2012). *A Practical Guide to Training Restricted Boltzmann Machines*, pp. 599–619. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-642-35289-8. doi:10.1007/978-3-642-35289-8_32. URL https://doi.org/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. ISSN 0899-7667. doi:10.1162/neco.2006.18.10.2283. URL https://doi.org/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. ISSN 2475-9066. doi:10.21105/joss.01193. URL http://dx.doi.org/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**, D260–D266.

Lecun Y, Bottou L, Bengio Y, Haffner P (1998). "Gradient-Based Learning Applied to Document Recognition." In *Proceedings of the IEEE*, pp. 2278–2324.

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, USA*, **108**(49), E1293–E1301. ISSN 0027-8424. doi:10.1073/pnas.1111471108. https://www.pnas.org/content/108/49/E1293.full.pdf, URL https://www.pnas.org/content/108/49/E1293.

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. https://doi.org/10.1080/00018732.2017.1341604, URL https://doi.org/10.1080/00018732.2017.1341604.

Nguyen HD, Wood IA (2016). "Asymptotic Normality of the Maximum Pseudolikelihood Estimator for Fully Visible Boltzmann Machines." *IEEE Transactions on Neural Networks and Learning Systems*, **27**(4), 897–902. ISSN 2162-237X. doi:10.1109/TNNLS.2015.2425898.

Nguyen HD, Wood IA (2016). "A Block Successive Lower-bound Maximization Algorithm for the Maximum Pseudo-likelihood Estimation of Fully Visible Boltzmann Machines." *Neural Computation*, **28**(3), 485–492. ISSN 0899-7667. doi:10.1162/NECO_a_00813. URL http://dx.doi.org/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.

Robin X, Turck N, Hainard A, Tiberti N, Lisacek F, Sanchez JC, MÃijller M (2011). "pROC: An Open-Source Package for R and S+ to Analyze and Compare ROC Curves." *BMC Bioinformatics*, **12**, 77. URL https://cran.r-project.org/web/packages/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. URL http://proceedings.mlr.press/v5/salakhutdinov09a.html.

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

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**, 695.

Affiliation:

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