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 discrete factors 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 discrete 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 index with the total sample size n. We use symbol y for a particular factor value and generic response variables interchangeably. The overall likelihood is

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

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

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

where $p_y \equiv P(y)$ is the marginal distribution of y and n_y 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)

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

is then related to Eq. (3) by

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

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

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

2.2. Pseudo-likelihood inference

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

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

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

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

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

and

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

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

$$L_{iy} = \sum_{k\in y}\left[\bar{h}_i^{(y)}(x_i^k|x_{j\backslash i}^k) - \ln Z_{iy}(x_{j\backslash i}^k)\right] - \frac{1}{2}\left[\lambda_h \sum_x h_i^{(y)}(x)^2 + \lambda \sum_{j,x,x'} J_{ij}^{(y)}(x,x')^2\right] \tag{14}$$

with first derivatives

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

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

where $\hat{f}_i^{(y)}(x)$ and $\hat{h}_{ij}^{(y)}(x,x')$ 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,

$$J_{ij}(x,x') = J_{ji}(x',x). \tag{16}$$

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

$$J_{ij}(x,x') \leftarrow \frac{1}{2}\left[J_{ij}(x,x') + J_{ji}(x',x)\right]. \tag{17}$$

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)

$$\hat{J}_{ij}^{(y)}(x,x') = -\left[\mathsf{C}^{(y)}\right]^{-1}_{ij}(x,x'), \tag{18}$$

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

$$\mathsf{C}_{ij}^{(y)}(x,x') = \hat{f}_{ij}(x,x') - \hat{f}_i(x)\hat{f}_j(x'). \tag{19}$$

Equation (18) 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 ($\epsilon = 1$):

$$\mathsf{C}^{(y)} \leftarrow \bar{\mathsf{C}}^{(y)} = (1-\epsilon)\frac{\mathrm{Tr}\,\mathsf{C}^{(y)}}{\mathrm{Tr}\,\mathsf{I}}\mathsf{I} + \epsilon\mathsf{C}^{(y)}, \tag{20}$$

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),

$$\hat{h}_i^{(y)}(x) = \bar{h}_i^{(y)}(x) - \sum_{j\neq i}\sum_{x'}\hat{J}_{ij}^{(y)}(x,x')\hat{f}_j^{(y)}(x'), \tag{21}$$

and

$$\bar{h}_i^{(y)}(x) = \ln\left[\hat{f}_i^{(y)}(x)/\hat{f}_i^{(y)}(0)\right], \tag{22}$$

where $\hat{f}_i^{(y)}(0)$ is the frequency of (reference) factor x_i for which the parameters are zero ($a_i = 0$). Equation (21) 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. (22) because

$$\hat{f}_i^{(y)}(x) = \frac{e^{\bar{h}_i^{(y)}(x)}}{Z_{iy}} = \hat{f}_i^{(y)}(0)\, e^{\bar{h}_i^{(y)}(x)} \tag{23}$$

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. (19), (21) and (22) 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. Classification

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

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y)p_y}{\sum_{y'} P(\mathbf{x}|y')p_{y'}} = \frac{1}{1+\sum_{y'\neq y} P(\mathbf{x}|y')p_{y'}/P(\mathbf{x}|y)p_y} = \frac{1}{1+e^{-F_y(\mathbf{x})}}, \tag{24}$$

where

$$F_y(\mathbf{x}) = \ln\left[\frac{P(\mathbf{x}|y)p_y}{\sum_{y'\neq y} P(\mathbf{x}|y')p_{y'}}\right]. \tag{25}$$

For binary response coded as $y = 0, 1$, Eq. (25) reduces to

$$\begin{aligned} F_1(\mathbf{x}) &= \ln P(\mathbf{x}|y=1) - \ln P(\mathbf{x}|y=0) + \ln(p_1/p_0) \\ &= \sum_i \left[h_i^{(1)}(x_i) - h_i^{(0)}(x_i)\right] + \sum_{i<j}\left[J_{ij}^{(1)}(x_i,x_j) - J_{ij}^{(0)}(x_i,x_j)\right] + \ln\frac{Z_0 p_1}{Z_1 p_0}. \end{aligned} \tag{26}$$

Therefore, if $J_{ij}^{(y)}(x,x') = 0$ (naive Bayes), Eq. (24) 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. (24) exists for data with more than two groups.

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

$$\ln Z_y = \frac{1}{n_y}\sum_{k\in y}\sum_i \ln\left\{\sum_x \left[e^{h_i^{(y)}(x)+\sum_{j\neq i} J_{ij}(x,x_j^k)/2}\right]\right\}, \tag{27}$$

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:

$$\ln Z_y = -\ln \hat{f}^{(y)}(0) - \frac{1}{2}\sum_{i\neq j}\sum_{x,x'} J_{ij}(x,x')\hat{f}_i(x)\hat{f}_j(x'). \tag{28}$$

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

$$s = \frac{1}{n}\sum_k \mathbb{1}\left[\hat{y}(\mathbf{x}^k) = y_k\right], \tag{29}$$

where

$$\hat{y}(\mathbf{x}) = \arg\max_y P(y|\mathbf{x}). \tag{30}$$

If response is binary, the score defined by Eq. (29) is sensitive to marginal distributions of the two groups via Eq. (26). 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. (29) and (30) 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. Titanic data

We first illustrate BB inference for full multilevel response data sets by **bbl** using the base R data Titanic:

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

The frequency data can be converted into raw data (one observation per row) using the utility function freq2raw in **bbl**:

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

```
  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)
```

```
  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 first divide the sample into train and test sets,

```
R> set.seed(158)
R> nsample <- nrow(titanic)
R> flag <- rep(TRUE, nsample)
R> flag[sample(nsample, nsample/2)] <- FALSE
R> dtrain <- titanic[flag,]
R> dtest <- titanic[!flag,]
```

and apply linear regression using `glm`:

```
R> fit <- glm(Survived ~ ., family=binomial(), data=dtrain)
R> prl <- predict(fit, newdata=dtest)
R> pROC::roc(response=dtest$Survived, predictor=prl, direction='<')$auc

Area under the curve: 0.7615
```

The BB inference in **bbl** uses objects of `S4` class `bbl` instantiated by input training data:

```
R> model <- bbl(data=dtrain, y='Survived')
R> model

An object of class bbl
 3 predictor states:
   Class = 1st 2nd 3rd Crew
   Sex = Female Male
   Age = Adult Child
 Responses:
   Survived = No Yes
 Sample size: 1101
```

The argument `y` specifies the column name of the response variable.
We first try a single pseudo-likelihood inference by

```
R> model <- train(model, method='pseudo', lambda=0)

  Inference for class "Survived" = No:
  Maximum pseudo-likelihood = -1.457337

  Inference for class "Survived" = Yes:
  Maximum pseudo-likelihood = -1.894484

R> model@h

[[1]]
[[1]][[1]]
       2nd        3rd       Crew
 1.6062299  3.3293218 -0.6986764
```

```
[[1]][[2]]
    Male
3.294906

[[1]][[3]]
    Child
-9.061989


[[2]]
[[2]][[1]]
       2nd        3rd       Crew
-0.6471455 -0.6117188 -1.8110162

[[2]][[2]]
      Male
-0.9233562

[[2]][[3]]
    Child
-3.667611
```

R> head(model@J,n=1)

```
[[1]]
[[1]][[1]]
[[1]][[1]][[1]]
     2nd 3rd Crew
2nd    0   0    0
3rd    0   0    0
Crew   0   0    0

[[1]][[1]][[2]]
          Male
2nd  -1.265668
3rd  -2.001114
Crew  2.501538

[[1]][[1]][[3]]
         Child
2nd  -2.563383
3rd   8.047997
Crew -3.758009


[[1]][[2]]
[[1]][[2]][[1]]
```

```
           2nd       3rd     Crew
Male -1.265668 -2.001114 2.501538

[[1]][[2]][[2]]
     Male
Male    0

[[1]][[2]][[3]]
          Child
Male -0.7464861


[[1]][[3]]
[[1]][[3]][[1]]
            2nd      3rd      Crew
Child -2.563383 8.047997 -3.758009

[[1]][[3]][[2]]
            Male
Child -0.7464861

[[1]][[3]][[3]]
      Child
Child     0
```

The function `train` here solves the maximum pseudo-likelihood equations (15) using training data and stores the inferred parameters $h_i^{(y)}$ and $J_{ij}^{(y)}$ 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.

We predict the survival probability of test individuals using the trained model:

```
R> pr <- predict(model, newdata=dtest, logit=FALSE)

 predicting group probabilities...

R> head(pr)

         No       Yes
1 0.8310902 0.1689098
2 0.8310902 0.1689098
3 0.8310902 0.1689098
4 0.8310902 0.1689098
5 0.8310902 0.1689098
6 0.8310902 0.1689098

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

```
Area under the curve: 0.7649
```

Here, Eq. (24) was used with `x` from the supplied `newdata`. The `newdata` can either contain the response column as in the above example (the program will disregard it), or only the predictor columns. The predictor columns must either have the same order as in the training data or be labeled with column names.

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(model, method='pseudo', lambda=10^seq(-6,-2,0.5),
+                 verbose=0)
R> cv

        lambda       auc
1 1.000000e-06 0.6534036
2 3.162278e-06 0.6645917
3 1.000000e-05 0.6737016
4 3.162278e-05 0.6885623
5 1.000000e-04 0.6899207
6 3.162278e-04 0.6960352
7 1.000000e-03 0.7005476
8 3.162278e-03 0.7047930
9 1.000000e-02 0.6710430
```

It returns a `data.frame` of AUCs for multiple `lambda` values. There is a maximum AUC at $\lambda = 1 \times 10^{-3}$. We use this information to make prediction:

```
R> lstar <- cv[cv$auc==max(cv$auc),]$lambda
R> model <- train(model, method='pseudo', lambda=lstar)

  Inference for class "Survived" = No:
  Maximum pseudo-likelihood = -1.502807

  Inference for class "Survived" = Yes:
  Maximum pseudo-likelihood = -1.955395

R> pr2 <- predict(model, newdata=dtest, progress.bar=FALSE)

 predicting group probabilities...

R> yhat2 <- model@groups[apply(pr2,1,which.max)]
R> mean(dtest$Survived==yhat2)

[1] 0.7836364

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

```
Area under the curve: 0.7728
```

3.2. 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, dh=1, dJ=1, distr='unif')
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 2 3 4 5
1 0 2 0 0 1
2 0 0 2 2 2
3 0 2 1 0 1
4 2 1 2 1 1
5 0 2 2 2 2
6 2 1 0 1 0
```

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

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

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

```
  Predictor 1: 46 iterations, likelihood = 1.02771
  Predictor 2: 47 iterations, likelihood = 0.83911
  Predictor 3: 44 iterations, likelihood = 0.702856
  Predictor 4: 42 iterations, likelihood = 0.977994
  Predictor 5: 39 iterations, likelihood = 1.00459
```

```
R> names(fit)
```

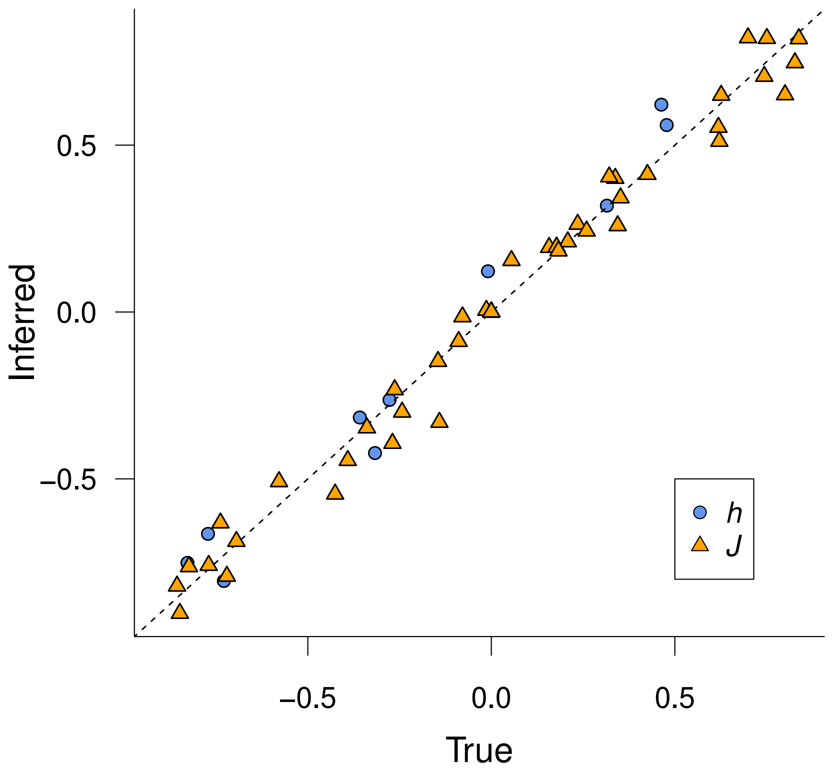

Figure 1: Comparison of true parameters and those inferred from pseudo-likelihood BB inference. See the text for conditions.

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

In contrast to `train`, which is designed for use with `bbl` object with 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 1 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> set.seed(135)
R> n <- 1000
R> for(i in 1:m) predictors[[i]] <- c('a','c','g','t')
R> par <- xi <- list()
R> for(iy in 1:2){
+    par[[iy]] <- randompar(predictors, h0=0.1*(iy-1), J0=0.1*(iy-1),
+                           distr='unif')
+    xi[[iy]] <- sample_xi(nsample=n, predictors=predictors, h=par[[iy]]$h,
+                          J=par[[iy]]$J)
+  }
```

```
R> dat <- cbind(rbind(xi[[1]],xi[[2]]), data.frame(y=c(rep('control',n),
+                                                        rep('case',n))))
R> model <- bbl(data=dat, groups=c('control','case'))
R> model

An object of class bbl
 5 predictor states:
   1 = a c g t
   2 = a c g t
   3 = a c g t
   ...
 Responses:
   y = control case
 Sample size: 2000
```

The explicit `groups` argument to `bbl` overrides the default detection and ordering of response groups from data. We now cross-validate using mean field inference,

```
R> cv <- crossval(model, method='mf', eps=seq(0,1,0.1),verbose=0)
R> head(cv)

  epsilon      auc
1     0.0 0.802728
2     0.1 0.846345
3     0.2 0.861013
4     0.3 0.866632
5     0.4 0.869644
6     0.5 0.871085
```

Here, `train` is called inside `crossval` as before but with `method = 'mf'`, which triggers mean field inference with Eqs. (18) and (21).

As shown in Fig. 2**a**, prediction AUC is optimized near $\epsilon = 0.8$. 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, 0.8, 1.0)
R> for(i in seq_along(eps))
+    fit[[i]] <- train(model, method='mf', eps=eps[i], verbose=0)
```

Figure 2**b-d** compares the three inferred parameter sets (`fit[[i]]@h, 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 $\epsilon = 0.8$ (Fig. 2**c**), as suggested by cross-validation AUC in Fig. 2**a**.

3.3. Image data

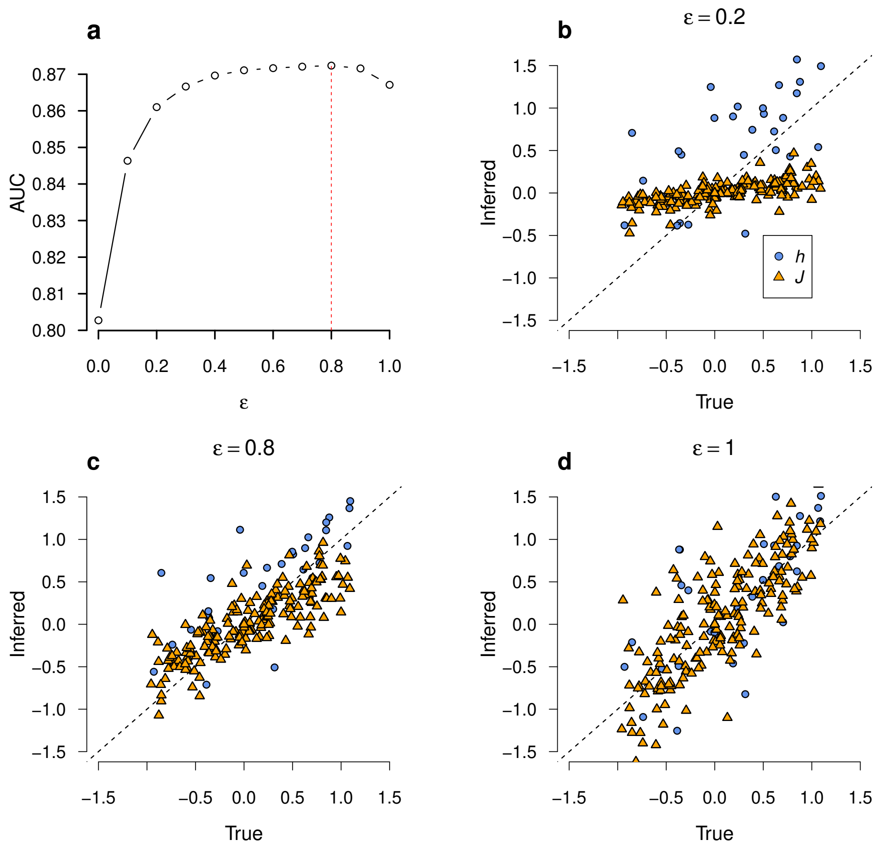

Figure 2: Regularized mean field inference using simulated data. (**a**) Cross-validation AUC with respect to regularization parameter ϵ. (**b-d**) Comparison of true and inferred parameters under three ϵ values. Best fit is achieved when AUC is maximum.

Advantages of **bbl** over regression become more apparent when dealing with large data sets and predictors numbering ~ 100 or more. Here, we consider the MNIST data set (yann.lecun.com/exdb/mnist/) widely used for benchmarking classification algorithms (Lecun, Bottou, Bengio, and Haffner 1998). Each sample in this data set contains grayscale levels ($x_i = [0, 255]$) derived from an image of 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[1:5,1:10]

  y X1 X2 X3 X4 X5 X6 X7 X8 X9
1 9  0  0  0  0  0  0  0  0  0
2 7  0  0  0  0  0  0  0  0  0
3 1  0  0  0  0  0  0  0  0  0
4 8  0  0  0  0  0  0  0  0  0
5 1  0  0  0  0  0  0  0  0  0

R> mnist <- bbl(data=dat)
```

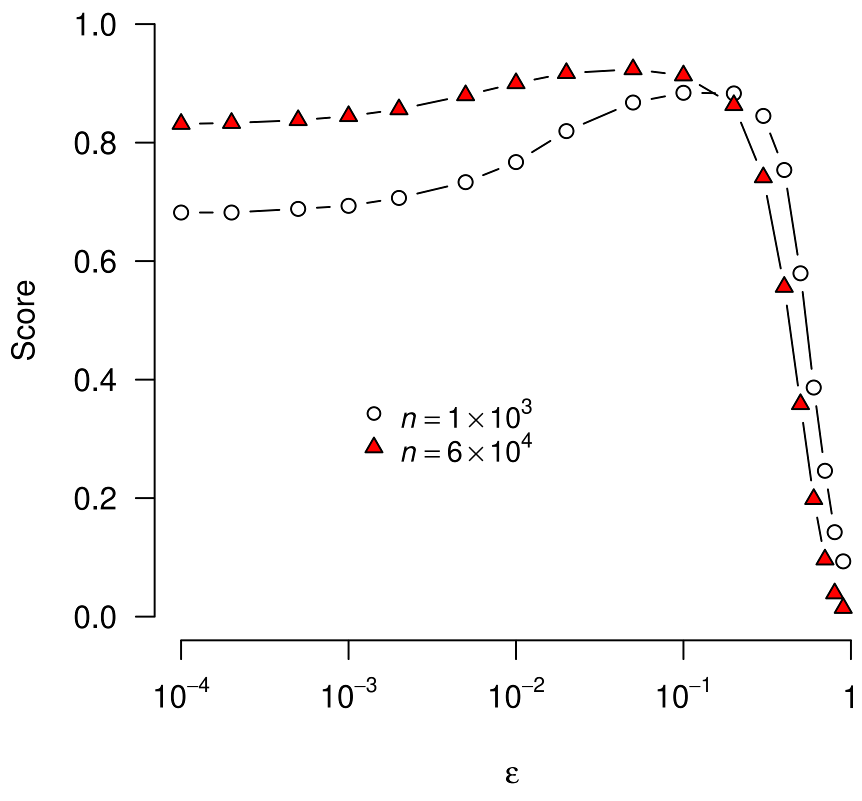

Figure 3: Cross-validation of BB inference on MNIST data using mean field option. Sample sizes are for down-sampled example and full data sets, respectively.

```
R> mnist
```

```
An object of class bbl
 584 predictor states:
   X40 = 0 1
   X41 = 0 1
   X45 = 0 1
   ...
 Responses:
   y = 0 1 2 3 4 5 6 7 8 9
 Sample size: 1000
```

Note that when the object is created, the predictors without factor variations (pixels that are always empty) are dropped from data.

```
R> cv <- crossval(mnist, method='mf', eps=0.1)
```

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 ($\epsilon = 0$)	16.2	**bbl**
BB	Mean field ($\epsilon = 0.05$)	6.8	**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.

The above run will take a few minutes. By feeding a vector of ϵ values, one can obtain the profile shown in Fig. 3 (white symbols). The substantial jump in performance under $\epsilon^* \sim 0.1$ 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. 2**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:

```
R> mnist <- train(mnist, method='mf', eps=0.1)
R> dtest <- read.csv(system.file('extdata/mnist_test.csv',package='bbl'))
R> dtest <- dtest[,colnames(dtest) %in% colnames(mnist@data)]
R> pr <- predict(mnist, newdata=dtest[,-1], progress.bar=FALSE)
R> yhat <- colnames(pr)[apply(pr, 1, which.max)]
R> mean(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. 3, red symbols) and obtained the test score of 0.932 (classification error rate 6.8%), which compares favorably with some of the best-performing large-scale neural network algorithms (Lecun *et al.* 1998; Salakhutdinov and Hinton 2009) (Table. 1).

3.4. 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 <- read.fasta(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. The function `read.fasta` will read a FASTA format file and turn it into a data frame. 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(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> model <- bbl(data=data)
R> model

An object of class bbl
 12 predictor states:
   X1 = A C G T
   X2 = A C G T
   X3 = A C G T
   ...
 Responses:
   y = Binding Non-binding
 Sample size: 1896
```

```
R> ps <- crossval(model, method='pseudo', lambda=10^seq(-1,-2,-0.2), verbose=0)
R> ps

      lambda       auc
1 0.10000000 0.8517130
2 0.06309573 0.8533535
3 0.03981072 0.8544531
4 0.02511886 0.8547420
5 0.01584893 0.8536301
6 0.01000000 0.8508037
```

```
R> mf <- crossval(model, method='mf', eps=seq(0.1,0.4,0.1),verbose=0)
R> mf

  epsilon       auc
1     0.1 0.8539235
2     0.2 0.8548070
3     0.3 0.8537745
4     0.4 0.8518373
```

In both cases, there is an optimal, intermediate range of regularization with maximum AUC (Fig. 4). 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. 4**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 the resolution of ~ 3 base pairs.

We analyzed 856 data sets 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. 5**a**). Pseudo-likelihood results had better scores compared to mean field on avarge (Fig. 5**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** extends the basic statistical distribution to accommodate heterogeneous, factor-valued predictors via Eq. (6), embedding it in a Bayesian classifier for supervised learning and prediction.

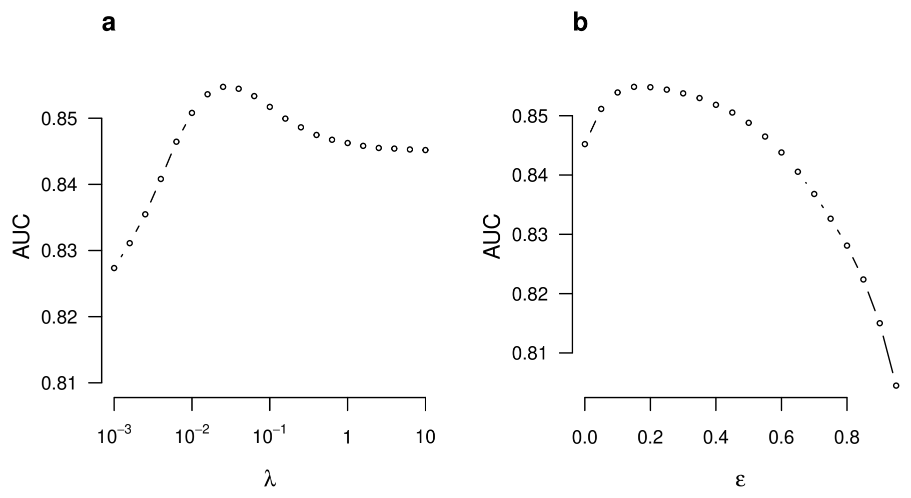

Figure 4: Cross-validation of discrete BB model on transcription factor binding motif data with control sequences generated by 3 nucleotide mutations. Data set is from Khan *et al.* (2018) (sample ID MA0014.3; see text). (**a**) Pseudo-likelihood and (**b**) mean field inferences.

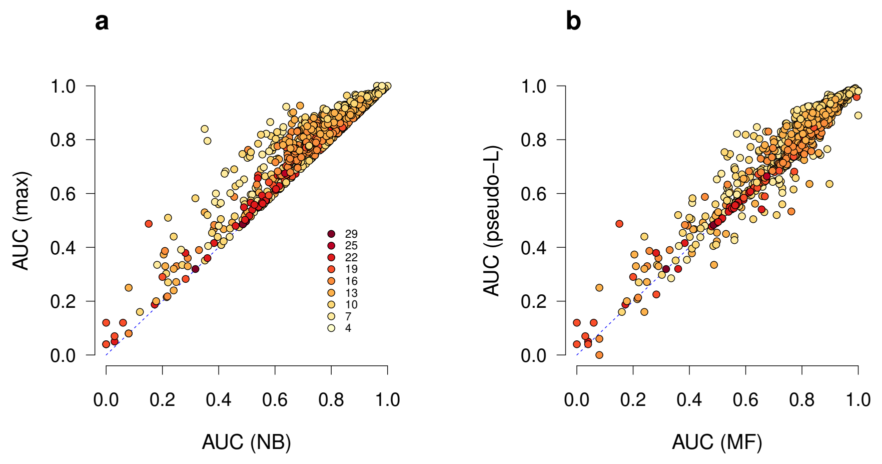

Figure 5: AUC scores of **bbl** model trained on 856 transcription factor binding site sequence data sets from JASPAR (Khan *et al.* 2018) under the same protocol as in Fig. 4. (**a**) Comparison of naive Bayes (NB; mean field with $\epsilon = 0$) and full mean field (MF) results. (**b**) Comparison 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. Tests on MNIST suggest performances comparable to other deep layer neural network models in classification tests. It 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. 3.4). 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 performs better.

Computational details

Installation of **bbl** requires the GNU Scientific library `https://www.gnu.org/software/gsl` installed. The results in this paper were obtained using R 3.6.0. R itself and all packages used are available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.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.

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`