Technical details for the `bnclassify` package

Bojan Mihaljević, Concha Bielza, Pedro Larrañaga

January 12, 2018

Abstract

Many algorithms for learning discrete Bayesian network classifiers from data have been developed so far. Yet, only a handful are available for the `R` statistical environment. The `bnclassify` package helps filling this gap by providing such learning algorithms. Besides the naive Bayes, it implements a well-known adaptation of the Chow-Liu algorithm and multiple variants of the greedy hill-climbing search, producing one-dependence estimators and semi-naive Bayes models. These algorithms can maximize either data-intrinsic or wrapper network scores. `bnclassify` also implements fast prediction for complete data and cross-validation. This vignette provides details on the implemented functionalities and serves as a reference for understanding them.

Contents

1 Introduction

Many algorithms for learning Bayesian network classifiers [Friedman et al., 1997] from data have been developed over the last 40 years [Bielza and Larrañaga, 2014]. However, only a handful are available in the R environment for statistical computing [R Core Team, 2015]. The `bnclassify` package helps filling this gap by implementing state-of-the-art algorithms for learning discrete Bayesian network classifiers, including both structure and parameter learning methods, as well as functions for using these classifiers for prediction, assessing their predictive performance, and inspecting and analyzing their properties.

2 Package overview

Structure learning algorithms:

- Naive Bayes [Minsky, 1961]
- Chow-Liu's algorithm for one-dependence estimators (CL-ODE) [Friedman et al., 1997]
- Forward sequential selection and joining (FSSJ) [Pazzani, 1996]
- Backward sequential elimination and joining (BSEJ) [Pazzani, 1996]
- Hill-climbing tree augmented naive Bayes (TAN-HC) [Keogh and Pazzani, 2002]
- Hill-climbing super-parent tree augmented naive Bayes (TAN-HCSP) [Keogh and Pazzani, 2002]

Parameter learning methods:

- Bayesian and maximum likelihood estimation

- Model averaged naive Bayes (MANB) [Dash and Cooper, 2002]
- Weighting to Alleviate the Naive Bayes Independence Assumption (WANBIA) [Zaidi et al., 2013]
- Attribute-weighted naive Bayes (AWNB) [Hall, 2007]

Model evaluating:

- Cross-validated estimate of accuracy
- Log likelihood
- Akaike information criterion (AIC) [Akaike, 1974]
- Bayesian information criterion (BIC) [Schwarz, 1978]

Predicting:

- Fast prediction for augmented naive Bayes models with complete data

In addition, it implements functions for querying models properties, enables network structure plotting through the `Rgraphviz` package [Hansen et al., 2015], uses `gRain` for inference with incomplete data, and provides additional utility functions.

3 Background

3.1 Bayesian networks

A Bayesian network is a pair $\mathcal{B} = (\mathcal{G}, \theta)$ which encodes a joint distribution over a random vector $\mathbf{X} = (X_1, \ldots, X_m)$. The directed acyclic graph $\mathcal{G}$ (its structure), which has a node per each variable in $\mathbf{X}$, encodes conditional independences among triplets of variables (e.g., X_1 is independent of X_3 given X_2), breaking the joint distribution into multiple smaller, local ones, over subsets of variables. The parameters θ specify the local distributions. The joint probability distribution is factorized as

$$P(\mathbf{X}) = \prod_{i=1}^{m} P(X_i \mid \mathbf{Pa}(X_i)),$$

where $\mathbf{Pa}(X_i)$ is the set of parents of X_i in $\mathcal{G}$ and m is the number of variables.

When learning a Bayesian network from data we are given a data set $\mathcal{D} = \{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$ of N observations of $\mathbf{X}$ from which we aim to learn $\mathcal{G}$ and θ. Learning θ is generally straightforward given a structure. To learn the structure, $\mathcal{G}$, there are basically two methods: by testing for conditional independence among triplets of variables (e.g., is X_1 independent of X_3 given X_2?) and by searching in the space of structures guided by a network quality score. In general, learning

Bayesian networks by searching in the structure space is NP-hard [Chickering et al., 2004] and thus a plethora of algorithms exist for the task, both specific for Bayesian networks, such as the K2 [Cooper and Herskovits, 1992], and general purpose search algorithms, such as genetic algorithms.

Learning the parameters θ of a local conditional distribution $\theta_{ijk} = P(X_i = k \mid \mathbf{Pa}(X_i) = j)$ is relatively straightforward. Assuming a Dirichlet prior over θ with all hyperparameters equal to α, and fully observed data, we get the Bayesian parameter estimates in closed form,

$$\theta_{ijk} = \frac{N_{ijk} + \alpha}{N_{\cdot j \cdot} + r_i \alpha}, \tag{1}$$

where N_{ijk} is the number of instances in $\mathcal{D}$ in which $X_i = k$ and $\mathbf{Pa}(X_i) = j$, $N_{\cdot j \cdot}$ is the number of instances in which $\mathbf{Pa}(X_i) = j$, while r_i is the cardinality of X_i. When $\alpha = 0$, Equation (1) yields the maximum likelihood estimate of θ_{ijk}.

3.2 Bayesian network classifiers

A Bayesian network classifier is a Bayesian network that is used for classifying instances into classes (e.g., classifying images as tumor or not tumor). Thus, we distinguish between predictor variables (or features), which describe the objects to classify, and which we label as $\mathbf{X}$, and a discrete class variable C. In this setting we are learning from a supervised data set $\mathcal{D} = \{(\mathbf{x}^1, c^1), \ldots, (\mathbf{x}^N, c^N)\}$.

A Bayesian network classifiers encodes the joint over $\mathbf{X}$ and C as

$$P(\mathbf{X}, C) = P(C \mid \mathbf{Pa}(C)) \prod_{i=1}^{n} P(X_i \mid \mathbf{Pa}(X_i)),$$

where $n = m - 1$ is the number of predictors. The classifier assigns an instance $\mathbf{x}$ to the most probable class:

$$c^* = \arg\max_c P(c \mid \mathbf{x}) = \arg\max_c P(\mathbf{x}, c).$$

Many search algorithms that produce Bayesian network classifiers traverse a restricted search space, considering only structures where the class node is a root node and parent of each feature. These are augmented naive Bayes [Friedman et al., 1997] models and all structures in Figure 1 are examples of it. The best known example is the naive Bayes [Minsky, 1961] (Figure 1a), a special case with no augmenting arcs (i.e., arcs between the features). Thus, the augmented naive Bayes factorizes the joint distribution as

$$P(\mathbf{X}, C) = P(C) \prod_{i=1}^{n} P(X_i \mid \mathbf{Pa}(X_i)), \tag{2}$$

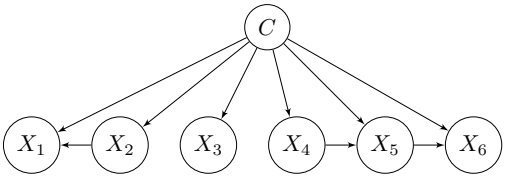
where $C \in \mathbf{Pa}(X_i)$ for all X_i.

4 Structure learning

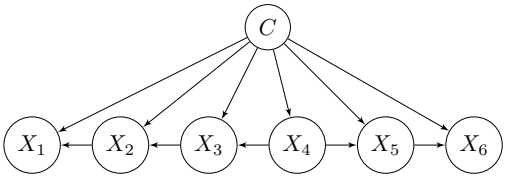
4.1 Models

All models produced by `bnclassify` are augmented naive Bayes models. These can be organized hierarchically according to their complexity.

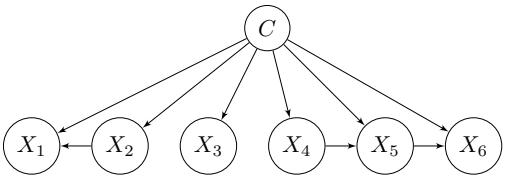
The simplest is the naive Bayes, which assumes that the predictors are independent given the class (Figure 1a). One-dependence estimators (ODE) allow each predictor to depend on at most one other predictor. A well-known example is the tree augmented naive Bayes (TAN) [Friedman et al., 1997], which has $n-1$ augmenting arcs, forming a tree in the predictors' subgraph (Figure 1b). Another variant is the forest augmented naive Bayes (FAN), with possibly less than $n-1$ augmenting arcs (Figure 1c). The semi-naive model (Semi) [Pazzani, 1996] allows for more complex dependencies among features. It partitions the predictor set into subsets that are fully dependent within them but mutually independent among them given the class. In other words, it forms a complete directed graph across each feature subset, with no arcs among the different subsets (Figure 1d).[1] The semi-naive Bayes model does not necessarily include all features (X_3 is omitted from the model in Figure 1d), i.e., it allows for feature selection [Liu and Motoda, 2007] embedded in the process of model construction.



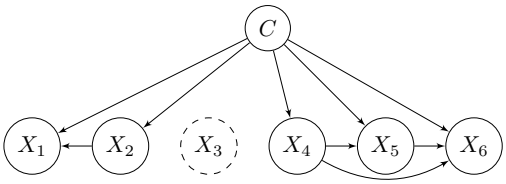
(a) $p(c, \mathbf{x}) = p(c)p(x_1|c)p(x_2|c)p(x_3|c)p(x_4|c)$ $p(x_5|c)p(x_6|c)$



(b) $p(c, \mathbf{x}) = p(c)p(x_1|c, x_2)p(x_2|c, x_3)p(x_3|c, x_4)p(x_4|c)$ $p(x_5|c, x_4)p(x_6|c, x_5)$



(c) $p(c, \mathbf{x}) = p(c)p(x_1|c, x_2)p(x_2|c)p(x_3|c)p(x_4|c)$ $p(x_5|c, x_4)p(x_6|c, x_5)$



(d) $p(c, \mathbf{x}) = p(c)p(x_1|c, x_2)p(x_2|c)p(x_4|c)$ $p(x_5|c, x_4)p(x_6|c, x_4, x_5)$

Figure 1: Augmented naive Bayes models of different complexity produced by the `bnclassify` package. (a) A naive Bayes; (b) a tree augmented naive Bayes; (c) a forest augmented naive Bayes; (d) a semi-naive Bayes. Here X_3 is omitted from the model.

4.2 Fixed structure

The naive Bayes has a fixed structure and its learning amounts to estimating its parameters.

[1] Note that, equivalently to this formulation, the original paper proposes forming 'supernodes' by joining features by means of the Cartesian product of their domains; these supernodes correspond to the complete subgraphs across predictor nodes in our formulation. There are two such supernodes in Figure 1d: (X_1, X_2) and (X_4, X_5, X_6).

4.3 Chow-Liu for one-dependence estimators

The CL-ODE algorithm by [Friedman et al., 1997] adapts the Chow-Liu [Chow and Liu, 1968] algorithm in order to find the maximum likelihood TAN model in time quadratic in n. Since the same method can be used to find ODE models which maximize decomposable penalized log-likelihood scores, `bnclassify` uses it to maximize Akaike's information criterion (AIC) [Akaike, 1974] and BIC [Schwarz, 1978]. While maximizing likelihood will always render a TAN, i.e., a network with $n-1$ augmenting arcs, maximizing penalized log-likelihood may render a FAN, since the inclusion of some arcs might degrade the penalized log-likelihood score.

Note that when data is incomplete `bnclassify` does not necessarily return the optimal (with respect to penalized log-likelihood) ODE. Namely, that requires the computationally expensive calculation of the sufficient statistics N_{ijk} which maximize parameter likelihood; instead, `bnclassify` approximates these statistics with the *available case analysis* heuristic (see Section 5).

4.4 Greedy hill-climbing

`bnclassify` implements four greedy hill-climbing algorithms for learning Bayesian network classifiers.

The following produce a TAN model:

- TAN-HC
- TAN-HCSP

Both algorithms start from a naive Bayes structure and add arcs until there is no improvement in network score. TAN-HCSP uses a reduced search space and should be less time-consuming.

The following algorithms produce a semi-naive Bayes model:

- BSEJ
- FSSJ

BSEJ starts from a naive Bayes structure and adds augmenting arcs and removes features from the model until no improvement in network score. On the contrary, FSSJ starts from a structure containing just the class node and proceeds by incorporating features features and augmenting arcs.

All greedy algorithms maximize the cross-validated estimate of accuracy as objective function, i.e., they are wrapper algorithms.

5 Parameter learning

5.1 Bayesian parameter estimation

With fully observed data, `bnclassify` estimates parameters with maximum likelihood or Bayesian estimation, according to Equation (1), with all prior hyperparameters $\alpha = a$ for a specified a

(provided by the user). When $N_{\cdot j\cdot} = 0$ and $a = 0$, that is, when the model's probability of observing $\mathbf{Pa}(X_i) = j$ is 0, `bnclassify` assigns a uniform distribution to $P(X_i \mid \mathbf{Pa}(X_i) = j)$.

When data is incomplete, the parameters of local distributions are no longer independent and we cannot separately maximize the likelihood of each one as in Equation 1. Optimizing (heuristically) the likelihood would require a time-consuming algorithm like expectation maximization [Dempster et al., 1977]. Instead, we use *available case analysis* [Pigott, 2001] and estimate the parameters independently, substituting $N_{\cdot j\cdot}$ in Equation 1 with $N_{ij\cdot} = \sum_{k=1}^{r_i} N_{ijk}$, i.e., with the count of instances in which $\mathbf{Pa}(X_i) = j$ and X_i is observed, regardless of the observation of other variables. That is,

$$\theta_{ijk} = \frac{N_{ijk} + \alpha}{N_{ij\cdot} + r_i \alpha}. \tag{3}$$

Thus, the parameter estimates are not maximum likelihood nor Bayesian when data is incomplete.

5.2 Exact model averaging for naive Bayes

The MANB parameter estimation method corresponds to exact Bayesian model averaging over the naive Bayes models obtained from all 2^n subsets of the n features, yet it is computed in time linear in n. The implementation in `bnclassify` follows the online appendix of Wei et al. [2011], extending it to the cases where $\alpha \neq 1$ in Equation (3).

The estimate for a particular parameter θ_{ijk}^{MANB} is:

$$\theta_{ijk}^{MANB} = \theta_{ijk} P(\mathcal{G}_{C \not\perp\!\!\!\perp X_i} \mid \mathcal{D}) + \theta_{ik} P(\mathcal{G}_{C \perp\!\!\!\perp X_i}),$$

where $P(\mathcal{G}_{C \not\perp\!\!\!\perp X_i} \mid \mathcal{D})$ is the local posterior probability of an arc from C to X_i, whereas $P(\mathcal{G}_{C \perp\!\!\!\perp X_i}) = 1 - P(\mathcal{G}_{C \not\perp\!\!\!\perp X_i} \mid \mathcal{D})$ is that of the absence of such an arc (which is equivalent to omitting X_i from the model), while θ_{ijk} and θ_{ik} are the Bayesian parameter estimates obtained with Equation (3) given the corresponding structures (i.e., with and without the arc from C to X_i).

Using Bayes' theorem,

$$P(\mathcal{G}_{C \not\perp\!\!\!\perp X_i} \mid \mathcal{D}) = \frac{P(\mathcal{G}_{C \not\perp\!\!\!\perp X_i}) P(\mathcal{D} \mid \mathcal{G}_{C \not\perp\!\!\!\perp X_i})}{P(\mathcal{G}_{C \not\perp\!\!\!\perp X_i}) P(\mathcal{D} \mid \mathcal{G}_{C \not\perp\!\!\!\perp X_i}) + P(\mathcal{G}_{C \perp\!\!\!\perp X_i}) P(\mathcal{D} \mid \mathcal{G}_{C \perp\!\!\!\perp X_i})}.$$

Assuming a Dirichlet prior with hyperparameter $\alpha = 1$ in Equation 3, Equation (6) and Equation (7) in the online appendix of Wei et al. [2011] give formulas for $P(\mathcal{D} \mid \mathcal{G}_{C \not\perp\!\!\!\perp X_i})$ and $P(\mathcal{D} \mid \mathcal{G}_{C \perp\!\!\!\perp X_i})$:

$$P(\mathcal{D} \mid \mathcal{G}_{C \not\perp\!\!\!\perp X_i}) = \prod_{j=1}^{r_C} \frac{(r_i - 1)!}{(N_{ij\cdot} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!,$$

$$P(\mathcal{D} \mid \mathcal{G}_{C \perp\!\!\!\perp X_i}) = \frac{(r_i - 1)!}{(N_i + r_i - 1)!} \prod_{k=1}^{r_i} N_{i\cdot k}!,$$

where $N_{i \cdot k} = \sum_{j=1}^{r_C} N_{ijk}$. Noting that the above are special cases of Equation (8) in Dash and Cooper [2002], we can generalize this for any hyperparameter $\alpha > 0$ as follows:

$$P(\mathcal{D} \mid \mathcal{G}_{C \not\perp X_i}) = \prod_{j=1}^{r_C} \frac{\Gamma(r_i \alpha)}{\Gamma(N_{ij} + r_i \alpha)} \prod_{k=1}^{r_i} \frac{\Gamma(N_{ijk} + \alpha)}{\Gamma(\alpha)},$$

and

$$P(\mathcal{D} \mid \mathcal{G}_{C \not\perp X_i}) = \frac{\Gamma(r_i \alpha)}{\Gamma(N_i + r_i \alpha)} \prod_{k=1}^{r_i} \frac{\Gamma(N_{ik} + \alpha)}{\Gamma(\alpha)}.$$

Following Wei et al. [2011], `bnclassify` asumes that the local prior probability of an arc from the class to a feature X_i, $P(\mathcal{G}_{C \not\perp X_i})$, is given by the user. The prior of a naive Bayes structure $\mathcal{G}$, with arcs from the class to a out of n, features and no arcs to the remaining $n - a$ features is, then,

$$P(\mathcal{G}) = P(\mathcal{G}_{C \not\perp X_i})^a (1 - P(\mathcal{G}_{C \not\perp X_i}))^{(n-a)}. \tag{4}$$

Note that `bnclassify` computes the above in logarithmic space to reduce numerical errors.

5.3 Weighting to Alleviate the Naive Bayes Independence Assumption

The WANBIA [Zaidi et al., 2013] method updates naive Bayes' parameters with a single exponent 'weight' per feature. The weights are computed by optimizing either the conditional log-likelihood or the mean root squared error of the predictions. `bnclassify` implements the conditional log-likelihood optimization as described in the original paper, namely optimizing it with the L-BFGS [Zhu et al., 1997] algorithm, with its gradient $\mathbf{g}$ given by

$$g_i = \sum_{j=1}^{N} \Big(\log P(X_i = x_i^{(j)} \mid c^{(j)}) - \sum_{c \in C} P(c \mid \mathbf{x}; \mathbf{w}) \log P(X_i = x_i^{(j)} \mid c) \Big), \tag{5}$$

where the probabilities are those estimated with maximum likelihood, i.e., without taking weights into account, whereas $P(c \mid \mathbf{x}; \mathbf{w})$ takes weights into account. This corresponds to discriminative learning of parameters, as a discriminative, rather than generative, objective function is optimized.

5.4 Attribute-weighted naive Bayes

The AWNB parameter estimation method is intended for the naive Bayes but in `bnclassify` it can be applied to any model. It exponentiates the conditional probability of a predictor,

$$P(\mathbf{X}, C) \propto P(C) \prod_{i=1}^{n} P(X_i \mid \mathbf{Pa}(X_i))^{w_i},$$

reducing or maintaining its effect on the class posterior, since $w_i \in [0, 1]$ (note that a weight $w_i = 0$ omits X_i from the model, rendering it independent from the class.). This is equivalent to updating parameters of θ_{ijk} given by Equation (3) as

$$\theta_{ijk}^{AWNB} = \frac{\theta_{ijk}^{w_i}}{\sum_{k=1}^{r_i} \theta_{ijk}^{w_i}},$$

and plugging those estimates into Equation (2). Weights w_i are computed as

$$w_i = \frac{1}{M} \sum_{t=1}^{M} \frac{1}{\sqrt{d_{ti}}},$$

where M is the number of bootstrap [Efron, 1979] subsamples from $\mathcal{D}$ and d_{ti} is the minimum testing depth of X_i in an unpruned classification tree learned from the t-th subsample ($d_{ti} = 0$ if X_i is omitted from t-th tree).

6 Cross-validation

`bnclassify` implements stratified cross-validation. It is possible to cross-validate the entire two-step learning process –learning the structure and the parameters– or parameter learning alone, keeping the structure fixed across the different subsamples.

7 Prediction

`bnclassify` implements prediction for augmented naive Bayes models with complete data. This amounts to multiplying the corresponding entries in the local distributions and is performed efficiently, with computations done in logarithmic space to reduce numerical error.

With incomplete data this cannot be done and therefore `bnclassify` uses the `gRain` package [Højsgaard, 2012] to perform exact inference. Such inference is time-consuming and, therefore, wrapper algorithms can be very slow when applied on incomplete data sets.

8 Future work

Future work involves handling real-valued data, via, for example, conditional Gaussian models. Straightforward extensions include additional variants of the greedy hill-climbing algorithm, such as the k-DB [Sahami, 1996] algorithm, or adding more flexibility to the hill-climbing algorithm, such as enabling the choice of the initial structure (e.g., whether a naive Bayes or a model with no arcs), when applicable. Useful parameter learning methods to implement include Zaidi et al. [2013].

References

H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

C. Bielza and P. Larrañaga. Discrete Bayesian network classifiers: A survey. *ACM Computing Surveys*, 47(1), 2014.

D. M. Chickering, D. Heckerman, and C. Meek. Large-sample learning of Bayesian networks is NP-hard. *The Journal of Machine Learning Research*, 5:1287–1330, 2004.

C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

D. Dash and G. F. Cooper. Exact model averaging with naive Bayesian classifiers. In *19th International Conference on Machine Learning (ICML-2002)*, pages 91–98, 2002.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979.

N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29: 131–163, 1997.

M. Hall. A decision tree-based attribute weighting filter for naive Bayes. *Knowledge-Based Systems*, 20(2):120–126, 2007.

K. D. Hansen, J. Gentry, L. Long, R. Gentleman, S. Falcon, F. Hahne, and D. Sarkar. *Rgraphviz: Provides plotting capabilities for R graph objects*, 2015. R package version 2.12.0.

S. Højsgaard. Graphical independence networks with the `gRain` package for `R`. *Journal of Statistical Software*, 46(10):1–26, 2012.

E. J. Keogh and M. J. Pazzani. Learning the structure of augmented Bayesian classifiers. *International Journal on Artificial Intelligence Tools*, 11(4):587–601, 2002.

H. Liu and H. Motoda. *Computational Methods of Feature Selection*. Chapman & Hall/CRC, Boca Raton, FL, 2007.

M. Minsky. Steps toward artificial intelligence. *Transactions on Institute of Radio Engineers*, 49: 8–30, 1961.

M. Pazzani. Constructive induction of Cartesian product attributes. In *Proceedings of the Information, Statistics and Induction in Science Conference (ISIS-1996)*, pages 66–77, 1996.

T. D. Pigott. A review of methods for missing data. *Educational research and evaluation*, 7(4): 353–383, 2001.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015. URL `http://www.R-project.org/`.

M. Sahami. Learning limited dependence Bayesian classifiers. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-1996)*, volume 96, pages 335–338, 1996.

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

W. Wei, S. Visweswaran, and G. F. Cooper. The application of naive Bayes model averaging to predict Alzheimer's disease from genome-wide data. *Journal of the American Medical Informatics Association*, 18(4):370–375, 2011.

N. A. Zaidi, J. Cerquides, M. J. Carman, and G. I. Webb. Alleviating naive Bayes attribute independence assumption by attribute weighting. *Journal of Machine Learning Research*, 14: 1947–1988, 2013.

C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560, 1997.