, 2024, 1–8
doi:

Towards the Genome-scale Discovery of Bivariate Monotonic Classifiers

Océane Fourquet,[1,2] Martin S. Krejca,[3] Carola Doerr[2] and Benno Schwikowski[1,*]

[1]Computational Systems Biomedicine Lab, Institut Pasteur, Université Paris Cité, 25–28 Rue du Dr Roux, 75015, Paris, France, [2]LIP6, CNRS, Sorbonne Université, 4 Place Jussieu, 75005, Paris, France and [3]LIX, CNRS, École Polytechnique, Institut Polytechnique de Paris, Honoré d'Estienne d'Orves, 91120, Palaiseau, France

*Corresponding author. benno.schwikowski@pasteur.fr

FOR PUBLISHER ONLY Received on Date Month Year; revised on Date Month Year; accepted on Date Month Year

Abstract

Motivation: Bivariate monotonic classifiers (BMCs) are based on pairs of input features. Similar to many other models used for machine learning, they can capture nonlinear patterns in high-dimensional data. At the same time, they are simple and easy to interpret. Until now, the use of BMCs on a genome scale was hampered by the high computational complexity of the search for pairs of features with a high leave-one-out performance estimate.

Results: We introduce the *fastBMC* algorithm, which is an improved algorithm to determine ensembles of well-performing BMCs. The algorithm is based on a mathematical bound for the BMC performance estimate while maintaining optimality. We show empirically that fastBMC speeds up the computation by a factor of at least 15 already for a small number of features, compared to the traditional approach. For two of three clinical datasets we consider here, the resulting possibility of considering much larger sets of features translates into significantly improved classification performance.

Availability: We provide the first open-source implementation for learning BMCs, and an implementation of fastBMC in particular, at `https://github.com/oceanefrqt/fastBMC` .

Contact: benno.schwikowski@pasteur.fr

Key words: Systems Biology, Classification, Algorithms, Monotonic functions, Bivariate functions, Interpretability

1. Introduction

High-throughput RNA sequencing and new data analysis methods show great potential for the diagnosis and classification of disease (Thind et al. (2021) and Zararsız et al. (2017)). In particular, supervised machine learning methods are powerful tools for associating molecular profiles with disease states and variants. However, many advanced methods that allow for nonlinear associations represent "black boxes" that make it difficult to understand how individual features in large molecular profiles contribute to predictions. This makes it challenging to generate mechanistic and experimentally testable hypotheses and to interpret the classifier in light of other biomedical knowledge.

One class of machine learning methods with intrinsically better interpretability are bivariate classifiers that involve only pairs of features. These classifiers are more complex than univariate classifiers but are still simple enough to allow effective learning on small datasets. They also allow for graphical representation and analysis of the models and data in two dimensions. For example, bivariate tree methods (Bioch et al. (1997)) are able to determine decision trees based on pairs of features with good predictive power. Top-scoring pair classifiers (Geman et al. (2004); Tan et al. (2005)), are based on pairs of features whose relative magnitude is associated with the prediction target.

Bivariate monotonic classifiers (BMCs) use general monotonic functions as decision boundaries (Nikolayeva et al., 2018). Figure 1 shows one of the best BMCs for the prediction of dengue severity in the transcriptomic dataset from Nikolayeva et al. (2018). Interestingly, the known biological functions of both transcripts overlap in the highly specific context of T cell co-stimulation (Hong et al.,

© The Author 2024.

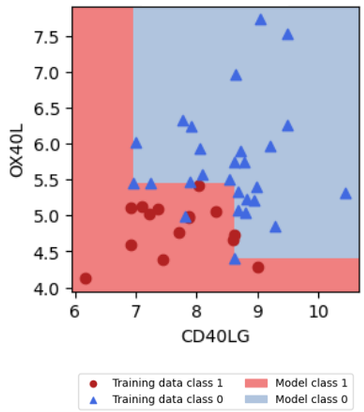

Figure 1: A bivariate monotonic classifier (BMC) is a function $f\colon \mathbb{R}^2 \to \{0,1\}$ that is monotonic in both dimensions. The function f thus divides $\mathbb{R}^2$ into two classes. This specific BMC, based on OX40 and CD40 ligand transcripts, is an example from Nikolayeva et al. (2018).

2015; Webb et al., 2016), which is known to be crucial in the response to dengue virus (Tian et al., 2019). Therefore, this classifier can be interpreted as a complex biological association, in which *simultaneously low* levels of OX40 and CD40 ligand transcripts are associated with severe dengue infection.

To capture more associations that involve more than two transcripts, multiple BMCs can be combined into an ensemble classifier (ensemble BMC). Nikolayeva et al. (2018) applied such an ensemble BMC to the classification of clinical dengue infections into *mild* and *severe*, according to the blood transcriptome at hospital admission. Their approach empirically performed as good as or better than other classifiers.

The ensemble BMC by Nikolayeva et al. (2018) was built from BMCs with the best empirical performance estimate, which requires the computation of this estimate for each possible pair of features. As this number grows quadratically with the number of features, this construction (which we refer to as *naïveBMC*) is computationally expensive and, in practice, can only be used on small subsets of typical genome-scale datasets.

Contribution. This paper addresses the construction of ensemble BMCs for larger genome-scale feature sets. We propose two contributions: (1) an efficient lower bound for the empirical error estimate of a given BMC (Theorem 1), and (2) the *Preselection Algorithm* that leverages this bound to accelerate the identification of suitable BMCs (Algorithm 1). This Preselection Algorithm is integrated into naïveBMC, producing a new version called *fastBMC*. This approach enables the computation of ensemble BMCs for significantly larger sets of features.

Outline. Section 2 provides formal definitions for the classification problem and the ensemble BMC approach (omitted in Nikolayeva et al. (2018)). The lower bound is then introduced in Section 3. We prove its correctness and provide details of the Preselection Algorithm in the same section. Finally, we present the results of our empirical evaluation in Section 4, which addresses the question of whether and how the testing of many more feature pairs results in better classification performance.

2. Formal Description of the Classification Problem

In this section, we provide a formal description of the classification problem that was introduced only incompletely at the formal level in Nikolayeva et al. (2018). Especially, Section 2.2 details the general approach that both naïveBMC and fastBMC follow.

2.1. Classification Setting and Bivariate Monotonic Classifiers

Formally, the input data in (Nikolayeva et al., 2018) is given as a vector of values for $m \in \mathbf{N}_{\geq 1}$ transcriptomic features, for each of $n \in \mathbf{N}_{\geq 1}$ patients. Each patient also has a disease label in $\{0,1\}$. The problem aims to predict, for a new patient, the disease label from the transcript values.

Nikolayeva et al. (2018) used *bivariate monotonic (binary) classifiers (BMCs)*, i.e., functions $f\colon \mathbb{R}^2 \to \{0,1\}$ that are monotonic in both dimensions.[1] Any function that increases monotonically in x or $-x$, and in y or $-y$ is a BMC. An *optimal* BMC f is one such that f minimizes the *classification error* (CE) of the input with respect to its labels $(v_i)_{i=1}^n \in \{0,1\}^n$, that is, $\sum_{i=1}^n |f(x_i, y_i) - v_i|$.

Since a CE-minimal f might not be unique, we follow the approach of Nikolayeva et al. (2018) and define an *optimal* BMC as one that classifies the most points in $\mathbf{R}^2$ as 1. Formally, a CE-minimal BMC f^* constructed over $S \subset \mathbf{R}^2$ is considered optimal if and only if (i) for all CE-minimal BMCs f constructed over S holds that $|\{x \in \mathbf{R}^2 \mid f^*(x) = 1\}| \geq |\{x \in \mathbf{R}^2 \mid f(x) = 1\}|$. The CE of the BMC in Figure 1 is 1, as exactly one blue triangle is within the red area and no red point falls in the blue area. The red area cannot be further increased without increasing the CE. Therefore, this BMC is optimal.

A CE-minimal BMC for n points can be efficiently computed by dynamic programming in time $\Theta(n \log^2 n)$ (Stout, 2013). Since this algorithm only constructs monotonically *increasing* BMCs, we compute four different CE-minimal BMCs for each transcriptomic feature pair – one for each possible orientation of each of the two axes. We then choose a CE-minimal BMC among these four options.

Note that although BMCs can be *described* as decision trees, common greedy algorithms for decision trees typically do not *identify* classifiers such as those in Figure 1. This may explain the poorer performance of the decision trees that were computed in Nikolayeva et al. (2018) for comparison.

2.2. Ensemble Bivariate Monotonic Classifiers

Nikolayeva et al. (2018) explain how to combine multiple BMCs into an ensemble classifier of at most $k_{\max}$ BMCs. Such an *ensemble BMC* is a set of $k \in \mathbf{N}_{\geq 1}$ BMCs, whose predicted label is the majority of the binary labels predicted by the k BMCs.[2]

The classification performance of any given BMC is estimated by removing a patient in turn from the input, training a BMC on the reduced dataset, and evaluating the absolute difference (0 or 1) between the predicted and the true label of the removed patient (this is called *leave-one-out cross-validation* (LOOCV)). The sum of these values is the BMC's *LOOCV error*, or *LOOCVE* (Section 2.3).

[1] Given a set of points $\{(x_i, y_i)\}_{i=1}^n \subset \mathbf{R}^2$, a *monotonically increasing* classifier (in x and in y) is any function $f\colon \mathbf{R}^2 \to \{0,1\}$ such that for all $i, j \in [1, n] \cap \mathbf{N}$ it holds that if $x_i \leq x_j$ and $y_i \leq y_j$, then $f(x_i, y_i) \leq f(x_j, y_j)$.

[2] In the case of a tie, the label is chosen to be 1.

The approach of Nikolayeva et al. (2018) is divided into three phases.

- **Phase 1: Estimate the Optimal Number k^* of BMCs**
  For each $k \in [1, k_{\max}] \cap \mathbf{N}$, estimate the classification performance of an optimal ensemble BMC, as follows.
  For each patient in turn, an ensemble BMC is constructed using all patients except the one evaluated (using a LOOCV). The error rates of these BMCs are calculated, and the top BMCs are selected for various values of k. The value of k whose model has the lowest LOOCVE is chosen as the optimal number k^* of BMCs.
- **Phase 2: Estimate Classification Performance with k^* BMCs**
  The second phase evaluates the performance of the ensemble BMC constructed with the k^* BMCs determined in Phase 1. Patient predictions are made using an ensemble BMC, and misclassification rates, calculated with LOOCV, are recorded. The performance of the ensemble BMC is quantified by estimating its error rate, calculated as the proportion of patients misclassified.
- **Phase 3: Constructing the Ensemble Classifier**
  To create the final ensemble BMC, we re-evaluate all BMCs using a LOOCV across all patients. This ensures robustness. The top k^* performing BMCs are then chosen to build the ensemble. This ensemble BMC aims to achieve the highest possible AUC score[3] by incorporating the most informative BMCs.

2.3. Prediction Error Estimate by Leave-One-Out Cross-Validation

Nikolayeva et al. (2018) used LOOCV to estimate the prediction error of a BMC from a dataset $\{\boldsymbol{x}_i\}_{i=1}^n =: S \subset \mathbf{R}^2$ of gene expression measurements for n patients, with labels $(v_i)_{i=1}^n \in \{0,1\}^n$. For a given pair of genes, for each $i \in [1, n] \cap \mathbf{N}$, they construct the optimal BMC f for $S \setminus \{\boldsymbol{x}_i\}$ and compute its prediction error $|f(\boldsymbol{x}_i) - v_i| =: \varepsilon_i$. They use the *LOOCV error* of the optimal BMC for S as $LOOCVE = \sum_{i=1}^n \varepsilon_i$ to estimate the predictive performance of the BMC on new data. Evaluating the LOOCVE of each BMC is computationally expensive, as it requires the construction of a new BMC for each left out patient. By computing the LOOCVE of all m pairs of transcriptomic features, the overall cost is mn, since, for each pair of transcriptomic features, each of the n patients needs to be left out in turn. In contrast, when using CE, the expensive LOOCV only needs to be performed for a subset of $\ell \leq m$ pairs of transcriptomic features. This results in an overall cost of $\ell n + m$, where the term m accounts for the evaluation of the CE. If $\ell \ll m$, then the overall cost of this new approach is much lower than that of the naïve approach.

3. The Preselection Algorithm and fastBMC

To reduce the cost of building the ensemble BMC, we introduce a Preselection Algorithm (Algorithm 1) that efficiently eliminates feature pairs that cannot appear in an optimal ensemble BMC. This algorithm integrates with naïveBMC proposed by Nikolayeva et al. (2018), resulting in an enhanced method that we call *fastBMC*. We first prove a mathematical bound underlying the Preselection Algorithm, guaranteeing its correctness. Afterwards, we detail the specific steps of the algorithm.

3.1. The Classification Error is a Lower Bound for the LOOCV Error

We prove in Theorem 1 that CE is a lower bound for LOOCVE. Theorem 1 shows that any point $\boldsymbol{x} \in \mathbf{R}^2$ that is correctly predicted by an optimal BMC constructed on a set of $n-1$ points is also correctly predicted by an optimal BMC constructed over the same $n-1$ points *and* $\boldsymbol{x}$.

In order to state the theorem and its proof concisely, we introduce some notation. Let $S \subset \mathbf{R}^2$ be a set of data points. For all $U \subseteq S$, let $\mathcal{C}(U)$ denote the set of all optimal BMCs over U. Furthermore, for all $U, V \subseteq S$ and all BMCs $C \in \mathcal{C}(U)$, let $\mathrm{MC}(C, V)$ denote the number of misclassified points among V according to C, that is, its CE.

Theorem 1 *Let $S \subset \mathbf{R}^2$, and let $\boldsymbol{x} \in S$. Furthermore, using the notation above, let $C_S \in \mathcal{C}(S)$ and $C_{S\setminus\{\boldsymbol{x}\}} \in \mathcal{C}(S \setminus \{\boldsymbol{x}\})$, and assume that $\boldsymbol{x} \in S$ is correctly classified by $C_{S\setminus\{\boldsymbol{x}\}}$. Then* $\mathrm{MC}(C_S, S) = \mathrm{MC}(C_S, S \setminus \{\boldsymbol{x}\})$.

Proof We show the equation to be proved via the following two cases.

Case $\mathbf{MC}(C_S, S) \geq \mathbf{MC}(C_S, S \setminus \{x\})$.
This inequality holds because eliminating a data point evaluated by C_S only keeps or reduces the number of misclassified points.

Case $\mathbf{MC}(C_S, S) \leq \mathbf{MC}(C_S, S \setminus \{x\})$.
Since $C_{S\setminus\{\boldsymbol{x}\}}$ is optimal (with respect to CE) to classify $S\setminus\{\boldsymbol{x}\}$, we have $\mathrm{MC}(C_S, S\setminus\{\boldsymbol{x}\}) \geq \mathrm{MC}(C_{S\setminus\{\boldsymbol{x}\}}, S\setminus\{\boldsymbol{x}\})$. Since $\boldsymbol{x}$ is correctly classified by $C_{S\setminus\{\boldsymbol{x}\}}$ by assumption, we have $\mathrm{MC}(C_{S\setminus\{\boldsymbol{x}\}}, S \setminus \{\boldsymbol{x}\}) \geq \mathrm{MC}(C_{S\setminus\{\boldsymbol{x}\}}, S)$. Moreover, as C_S is optimal for classifying S, it holds that $\mathrm{MC}(C_{S\setminus\{\boldsymbol{x}\}}, S) \geq \mathrm{MC}(C_S, S)$. By transitivity, this case follows.

Conclusion. Combining both cases concludes the proof. □

3.2. The Preselection Algorithm

Building on Theorem 1, we propose a Preselection Algorithm (Algorithm 1) that identifies a set of top-performing pairs of transcriptomic features for the BMCs (Section 2.1). We explain the inner workings of the algorithm, that is, how the CE threshold of the Preselection Algorithm changes over time and how this affects how many BMCs are eliminated.

3.2.1. Formal Implementation

Given an integer k, the Preselection Algorithm returns a set of BMCs that contain at least k pairwise disjoint pairs of transcriptomic features *(feature pairs)*. In order to reduce the expensive LOOCV computation, the main idea is to determine and maintain a CE threshold t, which allows removing BMCs based on their CE instead of their LOOCVE. On a high level, the Preselection Algorithm operates in three phases:

1. Evaluate the CE of all BMCs for all feature pairs and sort them with respect to increasing CE (lines 1 to 6).

[3] AUC stands for Area Under the Curve. The AUC ROC metric, abbreviated as AUC, consists of calculating the area under the Receiver Operating Characteristic curve, which plots all the values of the pair (1-Specificity, Sensitivity) according to the classification threshold.

2. Evaluate the LOOCVE of the BMCs with the lowest CE until the output contains at least k disjoint feature pairs, while also determining a threshold CE (lines 7 to 22).
3. Evaluate the LOOCVE of the remaining BMCs, adding better ones to the output, and update t (lines 23 to 41).

For Phase 1, we note that ties in the CE of different classifiers are broken arbitrarily (e.g., using lexicographic or random ordering).

Phase 2 chooses the CE threshold t as the maximum LOOCVE among all BMCs evaluated, which are stored in a search tree T. In this phase, we also store the number of disjoint feature pairs among the evaluated BMCs in the variable d. With each new BMC C that is evaluated, we iterate over all BMCs in T and determine their feature pairs. If the features of C do not appear in any existing feature pair, we increase d by 1, as we found a new feature pair. Otherwise, we proceed with the next iteration. This naïve method of checking the number of pairs of disjoint features could easily be improved. However, since it corresponds to the method of Nikolayeva et al. (2018) to select the best feature pairs, it is kept for consistency reasons.

In Phase 3, the remaining BMCs are considered in increasing order according to their CE. Due to Theorem 1, if a BMC has a CE greater than the current threshold t, its LOOCVE is also at least as bad. As the BMCs are ordered by their CE, we immediately skip the evaluation of any additional BMCs. Otherwise, we evaluate the LOOCVE of the BMC. During this evaluation, since we have a threshold t, if we see that the LOOCVE is worse than t, we stop the LOOCV and ignore the BMC. Otherwise, we add it to T. As T is increased, it might be the case that we can remove the BMCs that are too bad by now, recalling that we keep or eliminate all BMCs of the same LOOCVE. To this end, we check whether if we remove all BMCs with the worst LOOCVE (which is t), we still have at least k disjoint feature pairs. We check this in the same way as in Phase 2. If this is the case, we remove all BMCs with a LOOCVE of t, and we update t to the new worst LOOCVE among all BMCs that we still keep.

At the end of the Preselection Algorithm, we return all BMCs that were kept.

Note that after a run of the Preselection Algorithm, the BMCs are partitioned into three types: (1) those that are eliminated without computing their LOOCVE, (2) those for which the LOOCVE is (if possible, partially) computed but *not* selected in the output, and (3) those for which the LOOCVE is computed and that *are* part of the output. For example, for the dengue dataset (Figure 2), at the end of the algorithm, more than 97 % of all BMCs have been eliminated without computing their LOOCVE, while only a minimal percentage of BMCs are kept among the ones that are evaluated.

Algorithm 1 The Preselection Algorithm for identifying optimal BMCs. Let $m \in \mathbf{N}_{\geq 1}$ be the number of feature pairs, $n \in \mathbf{N}_{\geq 1}$ the number of patients, and let $\mathcal{S} := \{S_i\}_{i=1}^m$ be the feature pairs with their respective labels $\{V_i\}_{i=1}^m$ such that, for all $i \in [1, m] \cap \mathbf{N}$, it holds that $S_i \in (\mathbf{R}^2)^n$ and $V_i \in \{0, 1\}^n$. Further, let $k \in \mathbf{N}_{\geq 1}$ with $k \leq m$ be given. The Preselection Algorithm returns a set of BMCs with the lowest LOOCVE among $\mathcal{S}$ containing at least k disjoint feature pairs, if possible.

```
 1: Q ← empty min-priority queue
 2: for all i ∈ [1, m] ∩ N do
 3:     C ← optimal BMC for S_i with labels V_i
 4:     e ← CE of C (Section 3.1)
 5:     add (S_i, C) to Q with key e
 6: end for
 7: T ← empty search tree
 8: d ← 0
 9: t ← −∞
10: (S, C) ← minimum of Q
11: while Q not empty and d < k do
12:     remove (S, C) from Q
13:     q ← LOOCVE of C (Section 2.3)
14:     if q > t then
15:         t ← q
16:     end if
17:     if features of S are disjoint from all features in T then
18:         d ← d + 1
19:     end if
20:     add S to T with key q
21:     (S, C) ← minimum of Q
22: end while
23: e ← minimum key in Q
24: (S, C) ← minimum of Q
25: while Q is not empty do
26:     remove (S, C) from Q
27:     if e > t then
28:         break the loop
29:     end if
30:     q ← LOOCVE of C, aborting the evaluation if the
    LOOCVE gets larger than t
31:     if q ≤ t then
32:         add S to T with key q
33:         T′ ← T without the elements with key t
34:         if T′ contains at least k disjoint feature pairs then
35:             T ← T′
36:             t ← minimum key in T
37:         end if
38:     end if
39:     e ← minimum key in Q
40:     (S, C) ← minimum of Q
41: end while
42: return the BMCs of T
```

3.2.2. Update of the CE Threshold Inside of the Preselection Algorithm

Figure 3 shows the impact of updating the threshold t of the Preselection Algorithm. Initially, t is very high and does not allow us to eliminate BMCs according to their CE. This follows from the fact that there are no BMCs right to the vertical red dashed line in the leftmost plot. This bad initial value follows from the fact that CE is not always a very good estimate for LOOCVE. Recall that the Preselection Algorithm iterates on the BMCs in increasing order of CE, which means from left to right in the figures. Hence, the columns of points are evaluated from left to right. Phase 2 of the Preselection Algorithm stops once the output contains at least $k = 85$ disjoint feature pairs, which is not a small number. This means that a large number of BMCs must be evaluated first. Additional significant time can be saved by lowering this initial value of t during the evaluation process. More precisely, from the first update onwards—which already occurs after the first iteration in Phase 3—the threshold t drastically decreases (second graph from

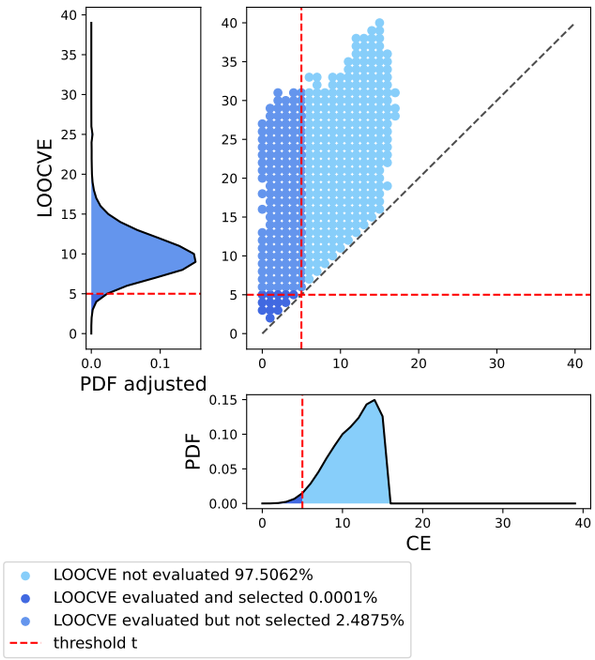

Figure 2: The reduction in the number of the LOOCVE evaluations using the Preselection Algorithm (Algorithm 1) on the dengue dataset (Section 4.1.1). Each point represents a set of potentially many BMCs with the CE and LOOCVE corresponding to the coordinates. The graphs along both axes represent the densities (probability density functions; PDFs) of the BMCs. The PDF at the bottom ranges over the whole dataset, the PDF at the left only over the BMCs that are evaluated (approximately 5 % of the overall data). The vertical dashed line represents the cutoff point at which no further LOOCVE evaluations were necessary. The horizontal dashed line shows the cutoff point beyond which no additional BMCs were selected among those whose LOOCVE was computed. Note that the update of the CE threshold t is not shown. See Figure 3 for an illustrated example.

the left), resulting in the elimination of most of the BMCs, as visible with the density functions in Figure 3. This is the case as more columns are evaluated, many of which contain BMCs with far better LOOCVE than the previously evaluated ones.

In several of the following iterations, the threshold is lowered again, leading to a final elimination of approximately 94 % of the BMCs for which the LOOCVE does not need to be calculated.

3.3. fastBMC

The fastBMC algorithm is an extension of naïveBMC that incorporates the Preselection Algorithm. The Preselection Algorithm is called within each of the three phases of naïveBMC mentioned in Section 2.2. Each of these phases conducts an exhaustive LOOCV. During each of these LOOCVs, immediately after every sample that is excluded for testing, the Preselection Algorithm is called. This means that the Preselection Algorithm is called multiple times per phase. The reason for calling the Preselection Algorithm repeatedly is that this ensures that we do not bias the overall LOOCV with respect to which patient we leave out. In more detail, the Preselection Algorithm does not introduce bias into the subsequent computations because it does not have access to the data point that is used for testing.

4. Does fastBMC Allow the Discovery of Better Classifiers?

Figures 2 and 3 suggest that fastBMC can drastically speed up the computation of ensemble BMCs, thus allowing for the analysis of significantly more features. But do the resulting classifiers also perform better on relevant biomedical datasets?

To answer this question, we compare the running times and classifier performances of fastBMC and the naïve approach on three clinical datasets from the literature, each consisting of transcriptome data and associated phenotypes.

In line with common pre-processing practice, we first eliminated those features with less than a fixed threshold of empirical variance, which is thought to reduce the complexity of the classification problem while retaining the information necessary for correct classification. Varying this threshold allowed us to [i] generate subdatasets of different sizes, [ii] to profile the speed-up obtained by fastBMC on these subdatasets, and [iii] to examine whether ability to process larger datasets using fastBMC results in improved classification performance.

4.1. Clinical Datasets

4.1.1. Dengue Dataset

This dataset comes from the study by Nikolayeva et al. (2018), in which an RNA signature predicted the severity of dengue in young patients. The set contains data from 42 patients, of whom 15 developed a severe form of dengue and 27 a non-severe form. In Nikolayeva et al. (2018), the 2 653 features with the highest variance were used.

4.1.2. Leukemia Dataset

This dataset comes from the study by Golub et al. (1999), showing how leukemia cases can be classified into acute myeloid leukemia and acute lymphoblastic leukemia. 38 samples were used for training and 34 for testing. The entire data comprises 7 129 features for 72 samples. It is accessible at `https://www.kaggle.com/datasets/crawford/gene-expression`.

4.1.3. Glioblastoma Dataset

This dataset comes from the study by Reifenberger et al. (2014) aimed at identifying transcriptomic markers of long-term glioblastoma survival. It comprises 23 samples from long-term and 16 from short-term survivors. Data were retrieved from the Gene Expression Omnibus (GEO) database at `http://www.ncbi.nlm.nih.gov/geo/` (accession number *GSE*53733). We retain only the 16 065 features corresponding to coding genes.

4.2. Empirical Evaluation: Speed-up

Figure 4 shows, for each of the clinical datasets, the running times of fastBMC and naïveBMC on the subdatasets of varying size.

Note that, interestingly, the running time of the fastBMC does not strictly increase with the number of input features: A newly added feature may lead to a reduction of the threshold t (as illustrated in Figure 3), leading fastBMC to perform fewer LOOCVE evaluations than without the newly added feature.

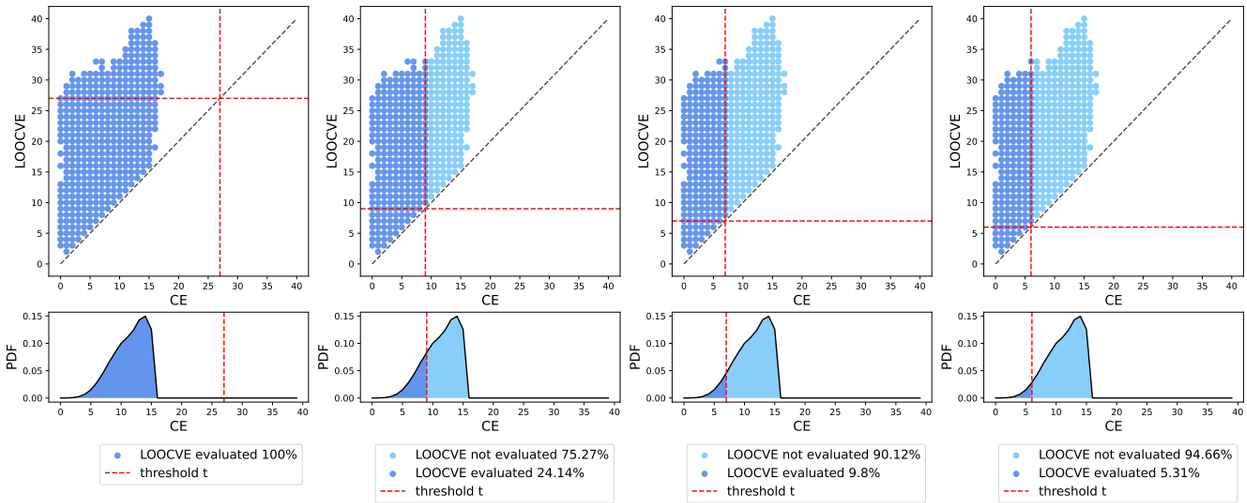

Figure 3: Successive updates of the threshold t of the Preselection Algorithm (Algorithm 1) in Phase 3 (lines 23–41). We ran the algorithm with $k = 85$ on the dengue dataset (Section 4.1.1). The series of panels at the bottom shows the different values of t over the course of Phase 3. With decreasing t, the number of BMCs for which the LOOCVE does not need to be computed (light blue area on the right) increases.

Table 1. Comparison of *user running time (URT)* between naïveBMC and fastBMC, on clinical datasets with 250 transcriptomic features. naïveBMC on the leukemia dataset was aborted after 1104 core-hours (23 cores × 48 hours.)

Clinical Dataset	URT fastBMC (h)	URT naïveBMC (h)	Speed-up
Dengue	30	449	15x
Glioblastoma	19	325	17x
Leukemia	170	–	–

More importantly, on subdatasets of 250 or more features, fastBMC is faster by at least a factor of 15 (Table 1), and Figure 4 suggests that this factor increases with increasing subdataset size.

4.3. Empirical Evaluation: Classification Performance

We now turn to the question of whether the increase in the number of features that can be processed by fastBMC also allows the discovery of better classifiers. To do this, we compute the AUC score of fastBMC and naïveBMC for varying subdatasets of increasing size across our three clinical datasets. As before, we aborted any analysis that took more than 1104 hours (user running time).

Figure 5 shows the result. First, we see that the performance of naïveBMC and fastBMC for the same input size is generally comparable. The eliminiation of certain low-performing models by the Preselection Algorithm does not appear to have a strong influence on classification performance. Second, we see that, for both approaches, a larger input size does not necessarily result in better performance. This observation is in line with the overfitting that can also be observed in other machine learning methods when too many features are provided. Third, for the dengue and leukemia datasets, the best performance is achieved for input sizes larger than what naïveBMC can handle within our time limit. However, for the glioblastoma dataset, the best performance is achieved using only a few features, which is within the capabilities of naïveBMC (albeit, using much more computation time).

4.4. Empirical Evaluation: Conclusion

Our empirical evaluation on three clinical datasets suggests that fastBMC produces models that perform similarly to naïveBMC, in drastically less time, thus allowing the analysis of significantly larger sets of features, which, in turn, is sometimes critical to discover better classifiers.

5. Discussion

In this article, we introduce an extension of an existing method by Nikolayeva et al. (2018), which we call fastBMC. This extension is based on a Preselection Algorithm that operates on a set of optimal bivariate monotonic classifiers *(BMCs)* for transcriptomic feature pairs and efficiently determines those with the lowest leave-one-out cross-validation error. Such sets are invaluable for constructing optimal ensembles of BMCs, which are not only highly accurate, but also easily interpretable. When combining the Preselection Algorithm with the approach by Nikolayeva et al. (2018) to compute BMCs, we empirically find that for datasets of 250 and more features, fastBMC is at least 15 times faster than the original approach (naïveBMC), and that this speed-up is also required in our empirical evaluation to discover optimal classifiers in reasonable computation time, in two out of three datasets. Thus, fastBMC represents a considerable advance for the discovery of BMCs especially in transcriptomic, but potentially also data-richer biological applications where even more features are available. The software we used for this study is the first open source library that allows the discovery of BMCs and their ensembles. It is available at `https://github.com/oceanefrqt/fastBMC`.

Moving forward, our efforts are directed towards further refinement of fastBMC and exploration of its applicability to biological and non-biological datasets. Currently, BMCs produce binary

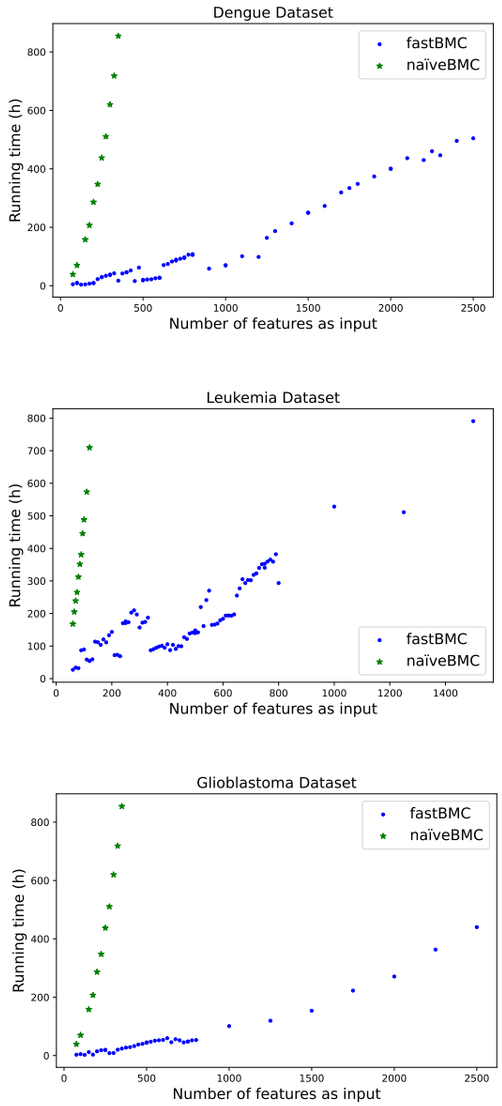

Figure 4: Running times in hours of naïveBMC and fastBMC on subdatasets of varying size, across three clinical datasets.

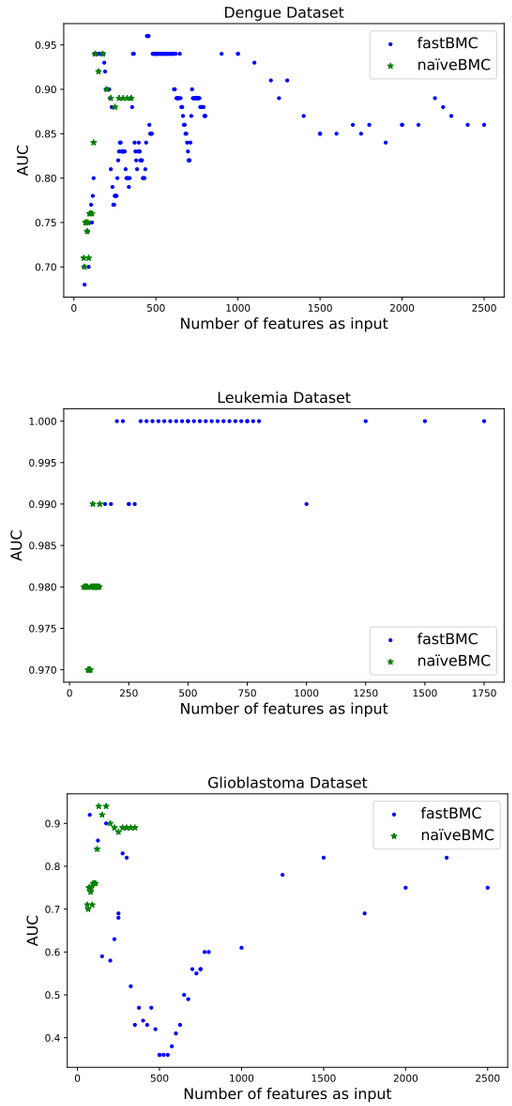

Figure 5: Comparison of AUC performance of ensemble BMCs constructed using fastBMC or naïveBMC for varying numbers of features.

output. To enhance the practical value of BMCs, it would be interesting to obtain the likelihood of a prediction. Such a likelihood could be defined, for example, on the basis of the distance of a point from the classification boundary by a method such as Platt scaling (Platt, 2000). Platt scaling is a calibration technique used in machine learning to convert the output of a classifier into well-calibrated probability scores, typically by applying a logistic function to model predictions.

Alternatively, a monotonic multiclass model could be used. Using any inherent order of training data, a multiclass approach represents a form of ordinal classification, typically used in bioinformatics for tasks such as cancer staging or prediction of disease progression. Beyond biology, ordinal classification is useful in other domains such as satisfaction surveys (Manderscheid and Lee (2023)), bank failure prediction (Manthoulis et al. (2020)), and quality control (Rosati et al. (2022)).

Acknowledgments

This work was supported by the Paris Île-de-France region, through the DIM RFSI project Opt4SysBio. Océane Fourquet and Benno Schwikowski have also received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 965193 for DECIDER.

References

J. C. Bioch, O. Meer, and R. Potharst. *Bivariate decision trees*, page 232–242. Springer Berlin Heidelberg, 1997. ISBN 9783540692362. doi: 10.1007/3-540-63223-9_122. URL `http://dx.doi.org/10.1007/3-540-63223-9_122`.

D. Geman, C. d'Avignon, D. Naiman, and R. Winslow. Classifying gene expression profiles from pairwise mRNA comparisons. *Statistical Applications in Genetics and Molecular Biology*, 3: 19–19, 2004. doi: 10.2202/1544-6115.1071.

T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. A. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286 5439:531–7, 1999. doi: 10.1126/science.286.5439.531.

G. U. Hong, J. Y. Lim, N. G. Kim, J.-H. Shin, and J. Y. Ro. IgE and IgA produced by OX40–OX40L or CD40–CD40L interaction in B cells–mast cells re-activate FcεRI or FcαRI on mast cells in mouse allergic asthma. *European Journal of Pharmacology*, 754:199–210, 2015. ISSN 0014-2999. doi: 10.1016/j.ejphar.2015.02.023. URL `https://www.sciencedirect.com/science/article/pii/S0014299915001302`.

E. Manderscheid and M. Lee. Predicting customer satisfaction with soft labels for ordinal classification. In S. Sitaram, B. Beigman Klebanov, and J. D. Williams, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 652–659, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-industry.62. URL `https://aclanthology.org/2023.acl-industry.62`.

G. Manthoulis, M. Doumpos, C. Zopounidis, and E. Galariotis. An ordinal classification framework for bank failure prediction: Methodology and empirical evidence for us banks. *European Journal of Operational Research*, 282(2):786–801, 2020. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2019.09.040. URL `https://www.sciencedirect.com/science/article/pii/S0377221719307970`.

I. Nikolayeva, P. Bost, I. Casademont, V. Duong, F. Koeth, M. Prot, U. Czerwinska, S. Ly, K. Bleakley, T. Cantaert, P. Dussart, P. Buchy, E. Simon-Lorière, A. Sakuntabhai, and B. Schwikowski. A blood RNA signature detecting severe disease in young dengue patients at hospital arrival. *The Journal of Infectious Diseases*, 217(11):1690–1698, 2018. doi: 10.1093/infdis/jiy086.

J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Adv. Large Margin Classif.*, 10, 06 2000.

G. Reifenberger, R. G. Weber, V. Riehmer, K. Kaulich, E. Willscher, H. Wirth, J. Gietzelt, B. Hentschel, M. Westphal, M. Simon, G. Schackert, J. Schramm, J. Matschke, M. C. Sabel, D. Gramatzki, J. Felsberg, C. Hartmann, J. P. Steinbach, U. Schlegel, W. Wick, B. Radlwimmer, T. Pietsch, J. C. Tonn, A. von Deimling, H. Binder, M. Weller, and M. Loeffler. Molecular characterization of long-term survivors of glioblastoma using genome- and transcriptome-wide profiling. *International journal of cancer*, 135(8):1822–1831, 10 2014. ISSN 1097-0215. doi: 10.1002/ijc.28836. URL `https://doi.org/10.1002/ijc.28836`.

R. Rosati, L. Romeo, V. M. Vargas, P. A. Gutiérrez, C. Hervás-Martínez, and E. Frontoni. A novel deep ordinal classification approach for aesthetic quality control classification. *Neural Computing and Applications*, 2022. ISSN 11625–11639. doi: https://doi.org/10.1007/s00521-022-07050-6.

Q. F. Stout. Isotonic regression via partitioning. *Algorithmica*, 66 (1):93–112, 2013. doi: 10.1007/s00453-012-9628-4.

A. C. Tan, D. Q. Naiman, L. Xu, R. L. Winslow, and D. Geman. Simple decision rules for classifying human cancers from gene expression profiles. *Bioinformatics*, 21(20):3896–3904, 2005. ISSN 1367-4803, 1460-2059. doi: 10.1093/bioinformatics/bti631. URL `https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/bti631`.

A. S. Thind, I. Monga, P. K. Thakur, P. Kumari, K. Dindhoria, M. Krzak, M. Ranson, and B. Ashford. Demystifying emerging bulk RNA-Seq applications: the application and utility of bioinformatic methodology. *Briefings in Bioinformatics*, 22(6): bbab259, 07 2021. ISSN 1477-4054. doi: 10.1093/bib/bbab259. URL `https://doi.org/10.1093/bib/bbab259`.

Y. Tian, A. Grifoni, A. Sette, and D. Weiskopf. Human T Cell Response to Dengue Virus Infection. *Frontiers in Immunology*, 10:2125, 2019. ISSN 1664-3224. doi: 10.3389/fimmu.2019.02125.

G. J. Webb, G. M. Hirschfield, and P. J. L. Lane. OX40, OX40L and Autoimmunity: a Comprehensive Review. *Clinical Reviews in Allergy & Immunology*, 50(3):312–332, 2016. ISSN 1559-0267. doi: 10.1007/s12016-015-8498-3.

G. Zararsız, D. Goksuluk, S. Korkmaz, V. Eldem, G. E. Zararsiz, I. P. Duru, and A. Ozturk. A comprehensive simulation study on classification of rna-seq data. *PLOS ONE*, 12(8):1–19, 08 2017. doi: 10.1371/journal.pone.0182507. URL `https://doi.org/10.1371/journal.pone.0182507`.