Fast Identification of Optimal Monotonic Classifiers

Océane Fourquet, Martin Krejca, Carola Doerr, Benno Schwikowski

, 2023, 1–7
doi:

Fast Identification of Optimal 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: Monotonic bivariate classifiers can describe simple patterns in high-dimensional data that may not be discernible using only elementary linear decision boundaries. Such classifiers are relatively simple, easy to interpret, and do not require large amounts of data to be effective. A challenge is that finding optimal pairs of features from a vast number of possible pairs tends to be computationally intensive, limiting the applicability of these classifiers.

Results: We prove a simple mathematical inequality and show how it can be exploited for the faster identification of optimal feature combinations. Our empirical results suggest speedups of 10x–20x, relative to the previous, naïve, approach in applications. This result thus greatly extends the range of possible applications for bivariate monotonic classifiers. In addition, we provide the first open-source code to identify optimal monotonic bivariate classifiers.

Availability: `https://gitlab.pasteur.fr/ofourque/mem_python`.

Contact: benno.schwikowski@pasteur.fr

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

1. Introduction

The use of high-throughput RNA sequencing and new computational methods to interpret the data collected have greatly enhanced the ability to diagnose and classify diseases, including cancer. A significant challenge in this area is the large number of data dimensions compared to the small number of samples, which can be addressed through the use of feature selection methods (Hastie et al. (2001)). These methods filter features as part of the process or in a dedicated preprocessing step, based on characteristics or statistical tests. While these feature selection methods can be effective, the resulting models are often either too simple to accurately describe the underlying biology, or too complex and difficult for biomedical researchers to interpret in the context of existing knowledge.

One approach to improving interpretability is the use of bivariate models that only involve pairs of features. These models are more complex than univariate models but still work with relatively little data and allow for the graphical representation and analysis of the models and data in two dimensions. An example of this is the top-scoring pairs (TSP) classifier (Geman et al. (2004)), which is based on pairs of genes whose expression significantly differs between two classes. TSP can be used with both linear models and more general monotonic modelsCano et al. (2019). However, in cases of many possible features, identifying the best bivariate model can be computationally expensive due to the need to test many possible feature pairs.

In this paper, we present a mathematical property for bivariate monotonic classifiers (Nikolayeva et al., 2018) that can be used to significantly speed up the identification of optimal monotonic feature pairs. We present a proof of the property (Theorem 1), provide an algorithm that takes advantage of it (Algorithm 1), and find that, across three biomedical use cases, the new algorithm speeds the running time up by a factor of 10–20 (Figure 4). Apart from (1) allowing to compute monotonic classification ensembles in much more reasonable time, our result (2) significantly broadens the range of possible applications (by allowing larger data sets as input), and (3) it allows to evaluate the obtained ensemble by statistical means.

2. System and Methods

Our mathematical inequality (Theorem 1) as well as the algorithm exploiting it (Algorithm 1) work for the task of binary, bivariate classification via monotonic models, as introduced by Nikolayeva et al. (2018). In this setting, we are given binary labeled data and aim to correctly predict the label of new data by constructing an *ensemble classifier* (or *ensemble* for short) from a set of gene pair classifiers, each of which operates on a combination of two features. The specific classifiers separate the data using bivariate, monotonic functions, which is why we call them *pair classifiers* (or just *classifiers* for short). The ensemble is constructed by selecting a good subset of pair classifiers, based on their prediction errors.

In the following, we describe this classification setting (Section 2.1), focusing on the pair classifiers (Section 2.2), their prediction error (Section 2.3), as well as a mathematical inequality (Theorem 1) that relates this computationally expensive error to a computationally cheaper one (Section 2.4). Theorem 1 forms the basis of our algorithm (Section 3).

2.1. Overview of the Classification Setting

In the setting of Nikolayeva et al. (2018), we are given binary-labeled disease data of $m \in \mathbf{N}_{\geq 1}$ pairs of gene expressions (the *gene pairs*) for $n \in \mathbf{N}_{\geq 1}$ patients, with labels in $\{0, 1\}$. For each gene pair, a (pair) classifier (Section 2.2) is constructed optimally, using the data for all n patients. The ensemble is a subset of $k^* \in \mathbf{N}_{\geq 1}$ classifiers, and the predicted label of the model is the majority of the labels of the k^* classifiers.[1] The choice of which classifiers to select is determined via leave-one-out cross-validation (LOOCV) over all n patients, called the classifier's *LOOCV error (LOOCVE)* (Section 2.3). In addition, this selection is given a bound $k \in \mathbf{N}_{\geq 1}$ with $k \leq m$ of how many classifiers to choose at most.

Existing Limitations of the Established Approach

Evaluating the LOOCVE of each classifier is very expensive, as it requires to build a new classifier for each patient left out. However, as we show in Theorem 1, the *regression error* (RE) of a classifiers, which can be computed without building new classifiers (Section 2.4), is a lower bound for a classifier's LOOCVE. This observation drastically reduces the number of candidate classifiers for which the LOOCVE needs to be computed, while still guaranteeing that the ensemble construction selects among the best possible classifiers.

Cost Reduction via Regression Error

We highlight the cost that is saved when utilizing the RE as a lower bound for a classifier's LOOCVE. To this end, we count the number of classifiers for which the LOOCVE needs to be computed, as this is by far the most expensive operation in this setting.

The approach of Nikolayeva et al. (2018) of computing the LOOCVE of all m gene pairs has an overall cost of mn, as, for each gene pair, each of the n patients need to be left out in turn. In contrast, when utilizing the RE, the expensive LOOCV only needs to be carried out for a subset of $\ell \leq m$ gene pairs. This results in an overall cost of $\ell n + m$, where the mterm accounts for the evaluation of the RE. If $\ell \ll m$, then the overall cost of this new approach is far lower than that of the previous one. In Section 4, we show that this is actually typically the case with real-world data.

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

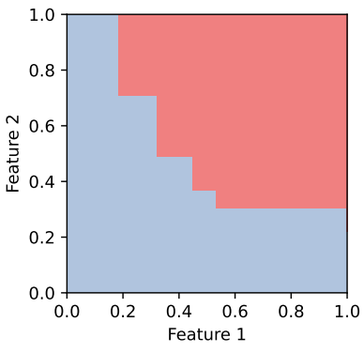

Figure 1: Example of a monotonic pair classifier (Section 2.2), a function f that is monotonic in both features and that separates the area into a red class (top right) and a blue class (bottom left).

2.2. Monotonic Classification and Pair Classifiers

Given a set of points $\{(x_i, y_i)\}_{i=1}^n =: S \subset \mathbf{R}^2$, a *monotonically increasing* classifier (in x and in y) is any function $f\colon \mathbf{R}^2 \to \{0, 1\}$ such that, for all $i, j \in [1, n] \cap \mathbf{N}$: $x_i \leq x_j$ and $y_i \leq y_j \implies f(x_i, y_i) \leq f(x_j, y_j)$. We call any function that is a monotonically increasing in x or $-x$, and in y or $-y$ a *monotonic pair classifier*, and visualize the two classes 0 and 1 using colors; see Figure 1 for an example.

We are interested in classifiers f that minimize the *regression error* (RE) of S with respect to its labels $(v_i)_{i=1}^n \in \{0, 1\}^n$, i.e. $\sum_{i=1}^n |f(x_i, y_i) - v_i|$. Since an RE-minimal f might not be unique, we follow the approach of Nikolayeva et al. (2018) and define an *optimal* classifier as one that classifies the most points in $\mathbf{R}^2$ as 1. Formally, an RE-minimal classifier f^* is optimal if and only if, for all RE-minimal monotonic classifiers f and all $\boldsymbol{x} \in \mathbf{R}^2$, it holds that $f^*(\boldsymbol{x}) \geq f(\boldsymbol{x})$ (see also Figure 2).

An RE-minimal monotonically increasing pair classifier for n points can be efficiently computed via dynamic programming (DP) in time $\Theta(n \log^2 n)$ (Stout (2013)). Since this algorithm only constructs monotonically *increasing* classifiers, we compute four different RE-minimal classifiers for each gene pair—one for each possible orientation of each of the two axes. In the end, we choose an RE-minimal classifier among these four options.

2.3. LOOCV Error of a Pair Classifier

Nikolayeva et al. (2018) estimate prediction errors of pair classifier from a data set $\{\boldsymbol{x}_i\}_{i=1}^n =: S \subset \mathbf{R}^2$ of gene expression measurements for n patients, with labels $(v_i)_{i=1}^n \in \{0, 1\}^n$ using *leave-one-out cross-validation* (LOOCV). For a given pair of genes, for each $i \in [1, n] \cap \mathbf{N}$, they construct the optimal pair classifier f for $S \setminus \{\boldsymbol{x}_i\}$ and compute the prediction error $|f(\boldsymbol{x}_i) - v_i| =: \varepsilon_i$. They use the *LOOCV error* of the optimal monotonic classifier for S as $LOOCVE = \sum_{i=1}^n \varepsilon_i$ to estimate the predictive performance of the classifier on new data.

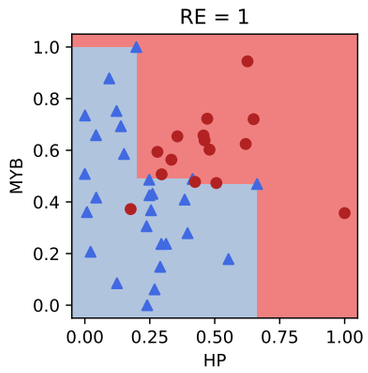

Figure 2: An optimal pair classifier (Section 2.2) for the shown point set. Blue triangles are labeled 0, red points are labeled 1. The background color (blue in the lower left area, red in the upper right) corresponds to the predicted labels of the classifier. The regression error (RE) (Section 2.4) of this model is 1, as exactly one red point is inside the blue area and no blue triangle falls in the red area. The red area cannot be increased any further without increasing the RE, hence this classifier is optimal.

2.4. Regression Error of a Pair Classifier and Relation to its LOOCV Error

The *regression error* (RE) of a classifier C trained on a set of data points $S \subset \mathbf{R}^2$ with labels $(v_{\boldsymbol{x}})_{\boldsymbol{x}\in S}$ is the number of misclassified points of C, that is, the cardinality of $\{\boldsymbol{x} \in S \mid C(\boldsymbol{x}) \neq v_{\boldsymbol{x}}\}$ (Figure 2).

We prove in Theorem 1 that the RE is a lower bound for the LOOCVE. Theorem 1 shows that any point $\boldsymbol{x} \in \mathbf{R}^2$ that is correctly predicted by a classifier constructed over a set of $n-1$ points is also correctly predicted by a classifier 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 classifiers over U. Further, for all $U, V \subseteq S$ and all $C \in \mathcal{C}(U)$, let $\mathrm{MC}(C, V)$ denote the number of misclassified points among V according to C, that is, its RE.

Theorem 1 *Let $S \subset \mathbf{R}^2$, and let $\boldsymbol{x} \in S$. Further, 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 prove via the following two cases.

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

Case $\mathbf{MC}(\boldsymbol{C_S}, \boldsymbol{S}) \leq \mathbf{MC}(\boldsymbol{C_S}, \boldsymbol{S} \setminus \{\boldsymbol{x}\})$.
Since $C_{S\setminus\{\boldsymbol{x}\}}$ is optimal (with respect to its RE) for classifying $S \setminus \{\boldsymbol{x}\}$, it holds that $\mathrm{MC}(C_S, S \setminus \{\boldsymbol{x}\}) \geq \mathrm{MC}(C_{S\setminus\{\boldsymbol{x}\}}, S \setminus \{\boldsymbol{x}\})$. Since $\boldsymbol{x}$ is correctly classified by $C_{S\setminus\{\boldsymbol{x}\}}$ by assumption, it holds that $\mathrm{MC}(C_{S\setminus\{\boldsymbol{x}\}}, S \setminus \{\boldsymbol{x}\}) \geq \mathrm{MC}(C_{S\setminus\{\boldsymbol{x}\}}, S)$. 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. Algorithm

Our algorithm (Algorithm 1) identifies a set of optimal pair classifiers (Section 2.2) for the setting of monotonic, bivariate classification (Section 2.1). The algorithm exploits the connection between the RE and the LOOCVE of a classifier provided by Theorem 1. Given an integer k, Algorithm 1 returns a set of classifiers that contain at least k pairwise disjoint gene pairs. Further, the classifiers that are returned have the lowest LOOCVE out of *all* classifiers for the set of provided gene pairs.[2] In order to reduce the expensive LOOCV computation, the main idea is to determine and maintain an RE threshold t, which allows to eliminate the pairs based on their RE instead of their LOOCVE. On a high level, the algorithm operates in three steps:

1. Evaluate the RE of all classifiers for all gene pairs and sort them with respect to increasing RE (lines 1 to 6).
2. Evaluate the LOOCVE of the classifiers with the lowest RE until the output contains at least k disjoint gene pairs while also determining an RE threshold (lines 7 to 22).
3. Evaluate the LOOCVE of the remaining classifiers, adding better ones to the output, and update t (lines 23 to 41).

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

Step 2 chooses the RE threshold t as the maximum LOOCVE among all the classifiers that are evaluated, which are stored in a search tree T. In this step, we also store the number of disjoint gene pairs among the evaluated classifiers in the variable d. With each new classifier C that is evaluated, we iterate over all gene pairs of classifiers in T. If the genes of C do not appear in any existing pair, we increase d by 1, as we found a new pair. Otherwise, we proceed with the next iteration.

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

At the very end of Algorithm 1, we return all of the classifiers that we kept.

Note that, after a run of Algorithm 1, the classifiers are partitioned into three types: those that are eliminated without computing their LOOCVE, those for which the LOOCVE is (if possible, partially) computed but that are not selected in the output, and the classifiers for which the LOOCVE is computed and that are part of the output (Figure 3).

[2] In case of ties, *all* classifiers with the same LOOCVE are kept, which means that the output can have a larger size than k.

Algorithm 1 The identification of optimal pair classifiers as described in Section 3. Let $m \in \mathbf{N}_{\geq 1}$ be the number of gene pairs, $n \in \mathbf{N}_{\geq 1}$ the number of patients, and let $\mathcal{S} \coloneqq \{S_i\}_{i=1}^m$ be the gene 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 algorithm returns a set of pair classifiers with the lowest LOOCVE among $\mathcal{S}$ containing at least k disjoint gene pairs, if possible.

$Q \leftarrow$ empty max-priority queue
for all $i \in [1, m] \cap \mathbf{N}$ **do**
    $C \leftarrow$ optimal pair classifier for S_i with labels V_i
    $e \leftarrow$ RE of C (Section 2.4)
    add (S_i, C) to Q with key e
end for
$T \leftarrow$ empty search tree
$d \leftarrow 0$
$t \leftarrow -\infty$
$(S, C) \leftarrow$ maximum of Q
while Q not empty and $d < k$ **do**
    remove (S, C) from Q
    $q \leftarrow$ LOOCVE of C (Section 2.3)
    if $q > t$ **then**
        $t \leftarrow q$
    end if
    if genes of S are disjoint from all genes in T **then**
        $d \leftarrow d + 1$
    end if
    add S to T with key q
    $(S, C) \leftarrow$ maximum of Q
end while
$e \leftarrow$ maximum key in Q
$(S, C) \leftarrow$ maximum of Q
while Q is not empty **do**
    remove (S, C) from Q
    if $e > t$ **then**
        break the loop
    end if
    $q \leftarrow$ LOOCVE of C, aborting the evaluation if the LOOCVE gets larger than t
    if $q \leq t$ **then**
        add S to T with key q
        $T' \leftarrow T$ without the elements with key t
        if T' contains at least k disjoint gene pairs **then**
            $T \leftarrow T'$
            $t \leftarrow$ maximum key in T
        end if
    end if
    $e \leftarrow$ maximum key in Q
    $(S, C) \leftarrow$ maximum of Q
end while
return the pair classifiers of T

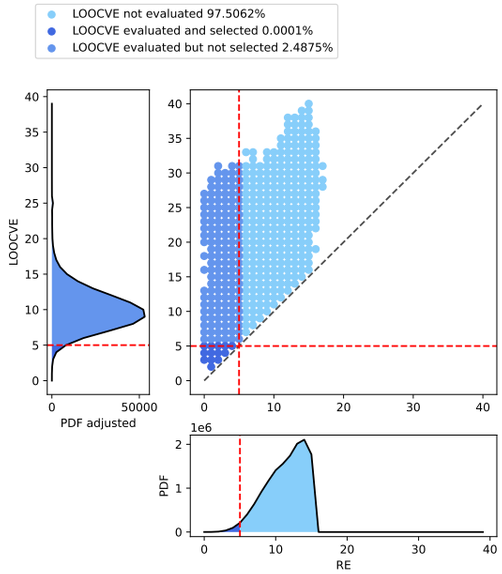

Figure 3: The reduction of LOOCVE evaluations using Algorithm 1 on the dengue severity classification problem (Section 4.1.1). Each point represents a set of potentially many pair classifiers with the RE and LOOCVE corresponding to the coordinates. The graphs along both axes represent the densities (PDFs) of the pair classifiers; the PDF at the bottom over the whole data set, the PDF on the left only over the pair classifiers that are evaluated (about 5 % of the overall data.). The vertical dashed line represents the cutoff at which no further LOOCVE evaluations were necessary. The horizontal dashed line shows the cutoff at which no further classifiers were selected among those whose LOOCVE was computed. Note that the update of the RE threshold t is not shown. See Figure 5 for an illustrated example.

4. Implementation

We evaluate empirically by how much Algorithm 1 reduces the number of LOOCVE computations (Section 2.3) for real-world data sets.

Our code is available online at `https://gitlab.pasteur.fr/ofourque/mem_python`.

In the following, we discuss the data sets that we use for our experiments (Section 4.1) and the experiment setup and our results (Section 4.2).

4.1. Data Sets

We evaluate Algorithm 1 on three real-world data sets, each containing gene expression data and binary labels from published studies. Each data set is from another disease. For each data set, we report the number of transcripts that we use for the experiments. Given x transcripts, this results in $\binom{x}{2}$ different gene pairs, which is the input for Algorithm 1. For each data set, we also eliminate transcripts with a low variance, in order to have a reasonable input size and to focus on more interesting transcripts.

4.1.1. Dengue Data Set

This data set comes from the study by Nikolayeva et al. (2018), determining a blood RNA signature detecting a severe form of dengue in young diseased patients. The set contains data of 42 patients, out of which 15 developed a severe form (label 1) of dengue and 27 a non-severe form (label 0). We follow the same filtering as in the study, which means that we eliminate the transcripts with a variance lower than 0.7. After this filtering, the data set comprises 2 653 transcripts.

4.1.2. Leukemia Data Set

This data set comes from the study by Golub et al. (1999), showing how new cases of cancer can be classified by gene expression, providing a general approach for identifying new cancer classes and assigning tumors to known classes. This data was used to classify patients with acute myeloid leukemia (label 1) and acute lymphoblastic leukemia (label 0). It is composed of two sub-datasets: the initial (training, 38 samples) and independent (test, 34 samples) data sets. We aggregate these two data sets into one, comprising 7 129 features for 72 samples. We filter the data set such that we only keep the 25 % of the features with highest variance, resulting in 1 783 remaining transcripts.

4.1.3. Bladder Cancer Data Set

This data set comes from the study by Urquidi et al. (2012), aiming to profile and differentiate tumoral urothelia samples from normal ones. This data set has been retrieved from CuMiDa (Feltes et al., 2019) and refers to $GSE31189$. It comprises 37 healthy samples (label 0) and 48 tumor samples (label 1). After only keeping the 25 % of the features with highest variances, we are left with 2 734 transcripts remaining.

4.2. Performance Evaluation

We investigate for how many classifiers Algorithm 1 does not compute the LOOCVE and for how many it only partially computes it. To this end, we run Algorithm 1 independently on each of the three data sets from Section 4.1 for various values of k, ranging from 5 to 285 in steps of 20. For the case of $k = 85$, we also log the RE threshold t in each iteration of step 3 of the algorithm (Section 3) in order to see how quickly it reduces.

We focus on two different aspects of our results. In Section 4.2.2, we discuss how the RE threshold of Algorithm 1 changes over time. In Section 4.2.1, we discuss how many LOOCVE computations Algorithm 1 saves.

4.2.1. Varying k of Algorithm 1

Figure 4 shows the percentage of classifiers for which Algorithm 1 does not compute the LOOCVE. This reduction is massive in all cases, reducing at least 80 % of the classifiers in the process. Since the running time of Algorithm 1 is mainly impacted by the LOOCVE calculations, this leads to heavily reduced running times compared to the naive approach of evaluating the LOOCV of all classifiers.

Not surprising, Figure 4 shows that, as k increases, fewer classifiers get eliminated, as the output size of Algorithm 1 is larger, hence requiring more classifiers to be fully evaluated via LOOCV. More interestingly is that the percentage reduces in steps. For example, for the dengue data set, except for the case $k = 5$, multiple values of k share the same amount of eliminated classifiers. This is the case because if Algorithm 1 contains a classifier, it contains *all* classifiers with the same LOOCVE. In order to guarantee this property, it evaluates the LOOCVE of all classifiers that share the same RE. Using the language of Figures 3 and 5, this means that the LOOCVE of an entire column of points is either (partially) computed or not at all. This leads to cases where increasing k does not result in evaluating an entirely new column, explaining the repeating numbers in Figure 4.

Among the different data sets, we see that Algorithm 1 eliminates the most classifiers for the leukemia data set while keeping the most for the bladder cancer data set. A larger amount of eliminated classifiers relates to more different RE values and/or the RE being close to the actual LOOCVE. In any case, for the same value of k, the order among the different data sets remains the same, suggesting that running Algorithm 1 with small values of k provides a good estimate of how many classifiers are eliminated for larger values of k.

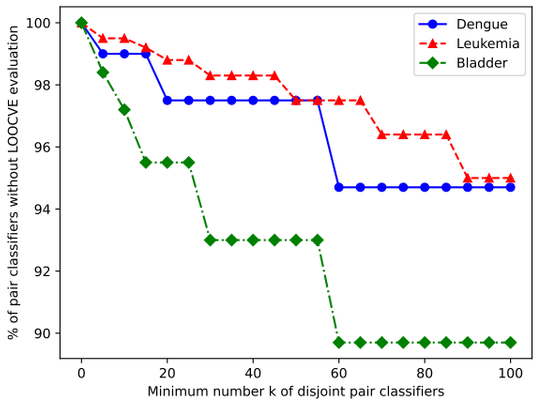

Figure 4: The percentage of pair classifiers whose LOOCVE is not evaluated after running Algorithm 1 with the specified value of k (on the x-axis) and the specified data set (the different curves; see also Section 4.1). With increasing k, the LOOCVE of more classifiers is evaluated. For typical values of k, at least 90 % of all possible pair classifiers are not evaluated, resulting in speedups of 10–20. The light blue points in the upper right of Figures 3 and 5 represent the pair classifiers without LOOCV evaluation in the dengue classification problem with $k = 85$.

4.2.2. Update of the RE Threshold Inside of Algorithm 1

Figure 5 shows the impact of updating the threshold t of Algorithm 1. Initially, t is very high and does not allow to eliminate pairs according to their regression error. This follows from there being no classifiers right to the vertical red, dashed line in the leftmost plot. This bad initial value follows from the RE error not always being a very good estimate for the LOOCVE. Recall that Algorithm 1 iterates over the classifiers in order of increasing RE, which means from left to right in the figures. Hence, the columns of points are evaluated from left to right. Step 2 of the algorithm stops once the output contains at least $k = 85$ disjoint gene pairs, which is not a small number. This means that a large number of classifiers need to 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—that already occurs after the first iteration in step 3—the threshold t drastically decreases (second plot from the left), resulting in the elimination of most of the classifiers, as visible with the density functions in Figure 5. This is the case as more columns are evaluated, many of which contain classifiers with far better LOOCVE than the previously evaluated ones.

In the following iterations, the threshold is lowered again (not always in each iteration tough), leading to a final elimination of

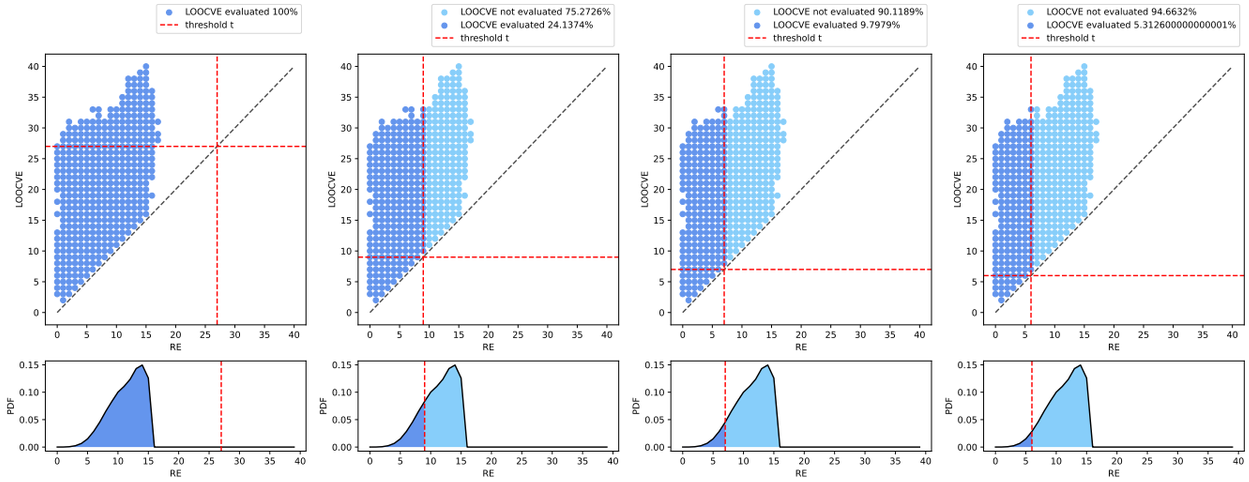

Figure 5: Successive updates of the threshold t of Algorithm 1 in step 3 (lines 23–41). We ran the algorithm with $k = 85$ on the dengue data set (Section 4.1.1). The series of panels at the bottom show the different values of t in the course of step 3. With decreasing t, the number of pair classifiers for which the LOOCVE (Section 2.3) does not need to be computed (light blue area on the right), increases.

about 94 % of the classifiers for which the LOOCVE does not need to be calculated.

5. Discussion

In this paper, we present a preprocessing method for the fast computation of optimal binary monotonic pair classifiers (Section 2). We prove that the regression error (RE) of a classifier is a lower bound for its leave-one-out cross-validation (LOOCV) error (Theorem 1). We provide an algorithm that uses the RE in a straightforward way as a lower bound for the identification of good pair classifiers. On biological data sets, we find that this approach typically avoids the calculation of LOOCV for 90 % or more of the possible pair classifiers that are evaluated in the naïve approach, corresponding to speedups of at least 10–20. The open-source implementation we provide with this paper (`https://gitlab.pasteur.fr/ofourque/mem_python`) implements this speedup.

We see impacts of the resulting drastic speedup in at least three areas. First, when the genes underlying a monotonic pair classifier are selected from a large number of possible gene pairs, as in Nikolayeva et al. (2018), it it usually interesting to statistically assess the performance of the resulting classifier. As straightforward approaches to this typically require an order of magnitude more calculations, the speedup presented here brings the computation of p-values from the infeasible to the practical in many cases.

Second, the speedup also facilitates the use of more ambitious types of models. One example would be the use of monotonic pair models for classification with more than two classes. A straightforward approach to this problem could be based on binary pair classifiers that distinguish one class against all others. Since each class would have to be treated separately, the speedup we report here may be critical for the practical feasibility of such an approach.

Finally, and importantly, the number of possible feature pairs scales quadratically with the number of data dimensions. The speedup enabled through our work thus also enables the use of monotonic pair classifiers for cases of higher data dimensionality, and ensures that this type of highly interpretable machine learning model will also remain practically applicable in potentially even more data-rich future biological applications.

Acknowledgments

Océane Fourquet and Benno Schwikowski have received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 965193 for DECIDER. Martin S. Krejca has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 945298-ParisRegionFP. Our work was also supported by the Paris Ile-de-France region, via DIM RFSI project Opt4SysBio.

References

J.-R. Cano, P. A. Gutiérrez, B. Krawczyk, M. Woźniak, and S. García. Monotonic classification: An overview on algorithms, performance measures and data sets. *Neurocomputing*, 341:168–182, 2019. ISSN 0925-2312. doi: 10.1016/j.neucom.2019.02.024.

B. C. Feltes, E. B. Chandelier, B. I. Grisci, and M. Dorn. Cumida: An extensively curated microarray database for benchmarking and testing of machine learning approaches in cancer research. *Journal of Computational Biology*, 26(4):376–386, 2019. doi: 10.1089/cmb.2018.0238.

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, 02 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.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer New York Inc., 2001. doi: 10.1007/978-0-387-84858-7.

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.

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

V. Urquidi, S. Goodison, Y. Cai, Y. Sun, and C. J. Rosser. A candidate molecular biomarker panel for the detection of bladder cancer. *Cancer Epidemiology, Biomarkers & Prevention*, 21 (12):2149–2158, 2012. doi: 10.1158/1055-9965.EPI-12-0428.