Ann Oper Res (2014) 216:307–325
DOI 10.1007/s10479-011-0886-3

A single interval based classifier

Heeyoung Kim · Xiaoming Huo · Jianjun Shi

Published online: 15 May 2011

Abstract In many applications, it is desirable to build a classifier that is bounded within an interval. Our motivating example is rooted in monitoring in a stamping process. A novel approach is proposed and examined in this paper. Our method consists of three stages: (1) A baseline of each class is estimated via convex optimization; (2) An "optimal interval" that maximizes the difference among the baselines is identified; (3) A classifier that is based on the "optimal interval" is constructed. We analyze the implementation strategy and properties of the derived algorithm. The derived classifier is named an *interval based classifier* (IBC) and can be computed via a low-order-of-complexity algorithm. Comparing to existing state-of-the-art classifiers, we illustrate the advantages of our approach. To showcase its usage in applications, we apply the IBC to a set of tonnage curves from stamping processes, and observed superior performance. This method can help identifying faulty situations in manufacturing. The computational steps of IBC take advantage of operations-research methodology. IBC can serve as a general data mining tool, when the features are based on single intervals.

Keywords Admissible length · Convex optimization · Interval based classifier · Support vector classifier

1 Introduction

We study a class of classification problems, in which it is known *a priori* that there exists an interval, such that a classifier built on this interval can achieve the best possible performance. An example can be seen in the tonnage data analysis (Jin and Shi 2005). Figure 1 plots 73

H. Kim · X. Huo (✉) · J. Shi
Georgia Institute of Technology, Atlanta, GA, USA
e-mail: xiaoming@isye.gatech.edu

H. Kim
e-mail: hykim@gatech.edu

J. Shi
e-mail: jianjun.shi@isye.gatech.edu

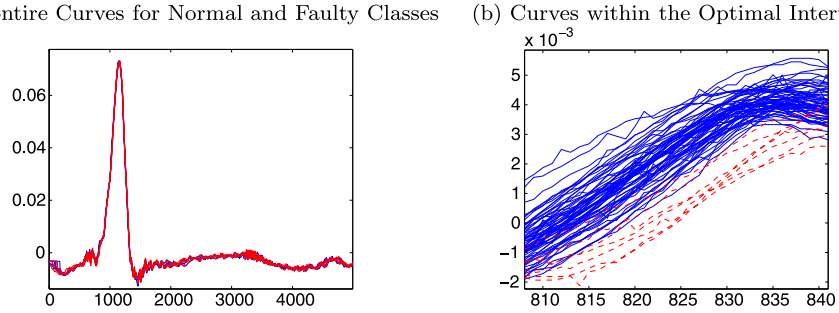

Fig. 1 (Color online) The curves are 4990 dimensional. There are 66 normal curves and 7 faulty curves. *Left figure* shows that a comparison based on the entire curves is insignificant. *Right figure* shows that the curves can be clearly separated over the interval [810, 845]

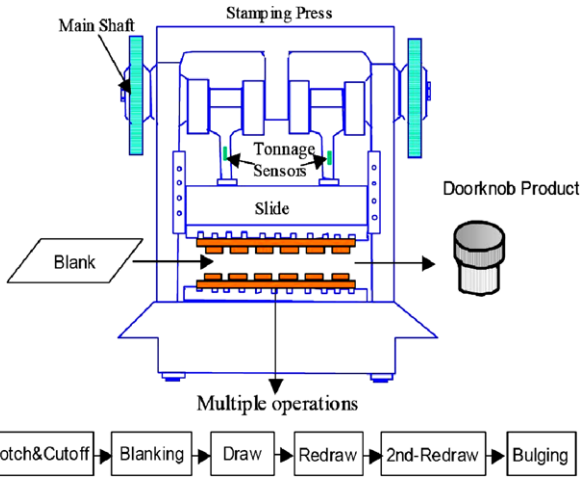

Fig. 2 Doorknob transfer die process

curves. Each curve is 4990 dimensional. Among these curves, 7 of them are considered faulty (in red) and 66 are considered normal (in blue). In Fig. 1(a), it is nearly impossible to distinguish these two classes. On the other hand, it is known that the two classes are well separated by segments restricted on the time interval [810, 845]. To classify faulty and normal curves, it is desirable to build a classifier that is only based on the aforementioned interval. Such an interval will be called an *optimal interval*. A classifier restricted on the optimal interval leads to faster and more effective prediction on new curves. A challenge is: we may know the existence of the optimal interval, however not the exact location and size of it.

The aforementioned tonnage data come from monitoring a transfer or progressive die process, which performs multiple operations by means of a die having several stations, each of which performs a different operation as the stock passes through the die. Figure 2 depicts a transfer die process that has six stations to produce a doorknob part. Generally, the press tonnage signal measurement is performed by the tonnage sensors (strain gage sensors) installed on the press linkages or columns. As an example of the doorknob process, two tonnage sensors are installed on the press linkages as shown in Fig. 2. The obtained press tonnage signal represents the total stamping force applied to all working stations in the press.

Press tonnage signals can be continuously collected during production by a computer-based tonnage monitoring system. To monitor the condition of each working station, it is required to know the signal profile of the stamping force at each station. Methodology has been developed (Jin and Shi 1999, 2005) to decompose the total stamping force into the individual forces generated from each station. For a given tonnage signal, it naturally breaks into various signal segments with each of them corresponding to specific stamping machine, die, and part interactions or conditions. Under normal production conditions, the tonnage signals are repeatable, or belong to the same baseline, for a given part in a fixed production condition. However, if some failures occur in production (machine failure, die broken or worn out, material properties change, part quality variation, etc.), the signals in some segments of the tonnage signals may change. The specific segments and their corresponding changes are different as the types of failures differ. However, the signals under given failure conditions are still repeatable, but belong to a different baseline that deviates from the normal conditions (as well as other failure conditions). In a production environment, because the tonnage signals are continually collected under various (unknown) system conditions, it is important to perform automatic classification to help grouping the tonnage signals into different clusters, which correspond to different production conditions. In the meantime, the optimal segments should be automatically determined to improve the sensitivity and performance of the classification.

We propose a classifier, which is built on an estimated optimal interval. We name our method an *interval based classifier* (IBC). The proposed method has three stages.

1. We estimate the *baseline* for each class by minimizing an objective function that is proportional to a weighted L_1 norm. Justification of such a criteria is provided. This step can be solved efficiently via standard convex optimization—a key component of operations research.
2. The optimal interval for classification is identified. We study the numerical details and demonstrate that it can be achieved via a low-order-of-complexity algorithm. Our analysis is based on studying the Lagrangian multiplier version of the original problem. The proposed method is fast: it takes a few minutes on a laptop to find the optimal classifier for a data set having nearly 100 curves and dimensionality of 4990.
3. A classifier that is based on the previously estimated optimal interval is proposed.

Our method is applied to a real data set from stamping processes. The outcome not only confirms our findings, but also improves our understanding of the data, which used to be rooted in the physical knowledge of the processes.

In this paper, we formulate a special classification problem from data mining into an optimization problem, and solve the problem with analytical study and demonstrate with real data. This paper makes several contributions. First of all, to the best of our knowledge, this is the first time that an interval-based classifier is proposed. Secondly, our method should be applicable in situations when features are interval based. Lastly, our derivation of the algorithm in the second stage generates a surprise, which consequently leads to an advantage of the IBC: it is easy to choose the algorithmic parameter in IBC. We explain more in the remaining of this paragraph. Recall IBC requires locating the optimal interval, which can be of any length. Note that when the dimension of the curve is large, one could have a large number of possibilities to examine. We found that one only needs to study those lengths that are associated with the vertices of a convex hull of an objective function. They, in our numerical experiments, form about 1% of all possible lengths. Hence IBC in practice can be done very efficiently—much more efficient than it appears. The above will become more evident in our description of numerical experiments.

We compare IBC with a state-of-the-art classifier—fused lasso support vector classifier (Tibshirani et al. 2005)—which utilizes stepwise constant baselines. The original goal of the fused lasso is to produce interval-like solutions, given that the coefficients are sparse or piecewise constant. (The major difference between IBC and fused lasso support vector classifier is that IBC is a classifier that is based on an interval, while fused lasso support vector classifier is a classifier whose coefficients resemble a step function—i.e., piecewise constant.) The fused lasso is a combination of lasso (Tibshirani 1996) and variable fusion (Land and Friedman 1996), and it penalizes both the coefficients and their successive differences. The fused lasso provides piecewise solution, however it is *not* guaranteed to generate the unique optimal interval, which is achieved in IBC. In particular in simulations, we run a comparison between IBC and an extension of the fused lasso—fused lasso support vector classifier. We demonstrate that for a real data set, the two methods produce similar solutions, but IBC is more stable and guarantees the unique optimal interval.

The rest of the paper is organized as follows. The IBC is described in Sect. 2. Numerical experiments in Sect. 3 demonstrate the effectiveness of IBC for both synthetic and real data. We conclude in Sect. 4.

2 Interval Based Classifier (IBC)

We give an overview of the proposed method for classification based on the most influential segment. In the remaining of this paper, we consider the "most influential segment" interchangeable with the "optimal interval". The former is more appropriate to be used when discrete algorithms are described; the latter is used when the problem is analyzed in continuum. Section 2.1 presents an estimator of the baseline of each class. Section 2.2 considers how to find the most influential segment with a prescribed size of the segment. Section 2.3 gives the main algorithm to find the all possible optimal intervals. Some physical justifications are provided in Sect. 2.4. Our classification rule is presented in Sect. 2.5. Finally, Sect. 2.6 describes a divide-and-conquer strategy, which can be used to deal with long curves.

2.1 Baseline estimation

The first step is to estimate the baseline of each class. Let $c_{i,j}$ denote the jth curve in the ith class, $i = 1, 2$ and $j = 1, 2, \ldots, n_i$, where n_i denotes the number of curves in class i. For each class, let b_i denote its baseline. Let $c_{i,j}(x)$ (respectively, $b_i(x)$) denote the value of the curve (respectively, the baseline) at location x, $1 \le x \le N$, where N is the length of the discrete curve (respectively, baseline). For class i, we estimate the baseline b_i by solving the following:

$$\begin{aligned} &\min_{b} \sum_{j=1}^{n_i} \sum_{x=1}^{N} |(c_{i,j}(x) - b(x))/\sigma_i(x)|, \\ &\text{subject to } b \in \mathcal{F}, \end{aligned} \tag{1}$$

where $\sigma_i(x)$ denotes an estimated standard deviation of errors in $c_{i,j}$'s at location x, $\mathcal{F}$ denotes a set of sequences that satisfies the following:

$$\mathcal{F} = \left\{ f : \sum_{x=1}^{N-1} |f(x+1) - f(x)| < C_1 \text{ and } \sum_{x=1}^{N-2} |f(x+2) - 2f(x+1) + f(x)| < C_2 \right\},$$

where C_1 and C_2 are predefined constants. There is a natural choice of C_1 and C_2: we compute the same quantities for the observed curves; then C_1 and C_2 are a constant (e.g., 1.5, to accommodate more functions) multiplied with the computed values. As for more general methods, cross validation (CV) can be used for the choice of C_1 and C_2. With the above specification, the optimization problem in (1) has a convex objective function and convex constraints. Efficient solvers are available. We used a general-purpose optimization toolbox: CVX (Grant et al. 2007). The execution time is quite impressive, referring to the section on numerical experiments.

2.2 Finding the most influential segment with a prescribed length

For a fixed length s of the target segment, we consider how to compute the most influential segment. Let $\hat{b}_1$ and $\hat{b}_2$ denote the estimates for b_1 and b_2. For a given length of the segment s, the most influential segment is the segment I^* that solves the following problem:

$$\max_{1\le i\le N-s+1} \left| \sum_{x=i}^{i+s-1} \frac{\hat{b}_1(x) - \hat{b}_2(x)}{\sigma(x)} \right|, \tag{2}$$

where $\sigma(x)$ is a pooled estimate of the standard deviation of errors at location x. If i^* is the maximizer of (2), we have $I^* = [i^*, i^* + 1, \ldots, i^* + s - 1]$. For simplicity, we denote the maximum value of the objective function in (2) as $h(s)$. The following theorem describes the complexity of solving problem (2). Proof of the theorem has been relegated to Appendix.

Theorem 1 *For a fixed segment length s, solving (2) can be done via an $O(N)$ algorithm.*

2.3 Convex hull and admissible lengths

If we try to solve for $h(s)$ for every $s (1 \le s \le N)$, the overall complexity can be $O(N^2)$. This order of complexity is too high for large N. It appears that we only need to compute $h(s)$ for a small subset of s. In particular, we only need to solve for $h(s)$ when $(s, h(s))$ is a vertex of the convex hull of the function $h(s)$. An illustration of the convex hull of function $h(s)$ is given in Fig. 3. The thick red line in the figure is a convex hull. Red circles are the vertices of the convex hull. We would argue that only the vertices of the convex hull worth

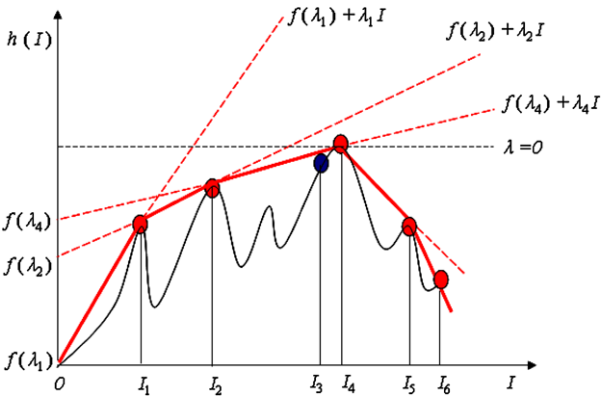

Fig. 3 (Color online) An illustration of the convex hull. Note that I or I_* denotes the segment lengths. Intercept $f(\lambda)$ is defined in the context

considering. If $(s, h(s))$ is *not* a vertex of the convex hull, we must have s_1 and s_2, such that $s_1 < s < s_2$ and

$$h(s) < \frac{s_2 - s}{s_2 - s_1} h(s_1) + \frac{s - s_1}{s_2 - s_1} h(s_2);$$

hence the most influential segment of length s can be substituted by a linear combination of two most influential segments of lengths s_1 and s_2, without sacrificing classification performance. The following theorem is an important observation in deriving our algorithm, which will be elaborated.

Theorem 2 *If $(s^*, h(s^*))$ is a vertex of the convex hull of the function $h(s)$, $1 \le s \le N$, then there must exist a constant λ, such that*

- *Segment I^* solves for*

$$\max_I \left| \sum_{x \in I} \frac{\hat{b}_1(x) - \hat{b}_2(x)}{\sigma(x)} \right| - \lambda |I|, \tag{3}$$

  where $|I|$ denotes the length of a segment I.
- *We have $s^* = |I^*|$ and $h(s^*) = |\sum_{x \in I^*} \frac{\hat{b}_1(x) - \hat{b}_2(x)}{\sigma(x)}|$.*

The proof of the above theorem utilizes standard arguments associated with the Lagrangian multipliers. We decide to omit a detailed proof. Note that in Fig. 3, $f(\lambda)$ denotes the maximum intercept associated with slope λ: i.e., for the I^* and λ in the setting of (3), we have $f(\lambda) = h(s^*) - \lambda s^*$. Recall problem (2) can be solved at $O(N)$. Using the same argument, one can verify that problem (3) can also be solved at $O(N)$.

If $(s, h(s))$ is a vertex on the convex hull of $h(s)$, we define s an *admissible length.* It turns out that admissible lengths are a small portion of all possible lengths. For example, for the tonnage data illustrated in Sect. 3.2, there are 4990 possible lengths; however, the number of admissible lengths is just 49 (roughly 1 percent). The inadmissible length is similar to the concept of inadmissible tree sizes in another statistical modeling problem (Huo et al. 2006). We now present a fast algorithm to search for all admissible lengths, as well as the corresponding most influential segments.

Algorithm

- **Initialization.** Take two initial values $0 < \lambda_{\min} < \lambda_{\max}$. Solve problem (3) with $\lambda_{\min}$ and $\lambda_{\max}$. Let s_1 and s_0 denote the lengths of the corresponding two most influential segments. It is evident that $s_0 < s_1$. (Ideally, we should have $s_0 = 1$ and $s_1 = N$.) Readers can get an intuition from Fig. 3. Initialize: OPEN $= \{pair(s_0, s_1)\}$; i.e., set OPEN contains one pair.
- **While** OPEN is not empty,

  Take a pair (s_1', s_2'), $s_1' < s_2'$, out of the set OPEN. Compute

$$\lambda = \frac{h(s_2') - h(s_1')}{s_2' - s_1'}.$$

  Solve problem (3) with the above λ. Let s_3' denote the length of the corresponding most influential segment. It is easy to verify that

$$s_1' \le s_3' \le s_2'.$$

If $s_1' < s_3'$, insert pair (s_1', s_3') into OPEN.
If $s_3' < s_2'$, insert pair (s_3', s_2') into OPEN.

End of While.

Because solving problem (3) requires $O(N)$ operations, the overall complexity of the above algorithm is the number of admissible lengths times $O(N)$. We will see that the number of admissible lengths is much smaller than N. In our numerical experiments with a real data set, the number of admissible lengths is roughly one percent of N.

2.4 Physical interpretation

The tonnage signal is the force measurement with x-axis being the crank angle (or equivalently as the travel distance by the upper die), and y-axis being the stamping force. The area covered under the tonnage signal is proportional to the forming energy. Under the same production conditions, the forming energy should be the same to form a designed part. The following interpretations come naturally:

- Minimizing in (1) is to find a baseline of a group of tonnage signals that have the similar forming energy in forming a part;
- Maximizing in (2) is to find the segment that can maximize the forming energy difference in the given segment. Such a segment reflects tonnage signals that are the most sensitive to the different forming conditions.

2.5 Classification rule

Using the estimated baselines ($\hat{b}_1$ and $\hat{b}_2$) and identified optimal interval (I^*), a new curve can be classified. Let c_{new} denote a new curve. The classification rule of IBC works as follows: If $\sum_{x=I_s^*}^{I_e^*}(|c_{\text{new}}(x) - \hat{b}_1(x)| - |c_{\text{new}}(x) - \hat{b}_2(x)|)/\sigma(x) < 0$, where I_s^* and I_e^* are the starting and ending points of I^*, respectively, and $\sigma(x)$ is a pooled estimate of the standard deviation at location x, then c_{new} is classified into class 1; otherwise, c_{new} is classified into class 2. Note that this classifier is a *nonlinear* one, unlike a support vector classifier that will be compared with later.

2.6 Divide and conquer strategy for long curves

The problem (1) of finding the baselines can be solved more efficiently by utilizing a divide-and-conquer strategy. The divide-and-conquer strategy works by dividing the original curve into several segments. Each segment of the curve is treated separately to find a baseline of the corresponding segment. All the separately estimated baselines are then concatenated to obtain the baseline of the original data. We cannot guarantee that the concatenated baseline is the same as the baseline that is estimated from the entire data. To resolve this problem, the neighboring points belonging to different segments are treated specially. Those special points are called the 'connecting-points group.'

We give a specific strategy to find the baseline with a divide-and-conquer strategy. Suppose that we divide the original data into n segments: $\{1, \ldots, p_1\}$, $\{p_1 + 1, \ldots, p_2\}$, $\ldots$, $\{p_{n-1} + 1, \ldots, p_n(= N)\}$, where p_i is the ending point of the ith segment, $i = 1, \ldots, n$. We define the ith connecting-points group as $\{p_i - 2, \ldots, p_i + 3\}$, where $i = 1, \ldots, n - 1$. The divide-and-conquer strategy can be presented as follows:

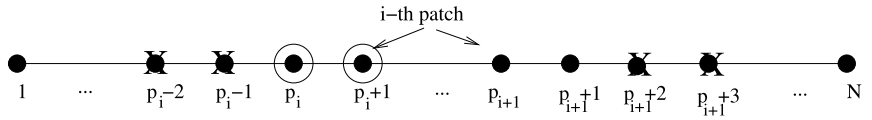

Fig. 4 An illustration for the divide-and-conquer strategy. *Circles* indicate constraining points used in the Initialization step. *Crosses* are the constraining points that are used in step 2

1. Initialization.
   (a) For $1 \le x \le p_1 + 1$, solve the problem (1). Let b_1^* denote the solution.
   (b) For $i = 2, \ldots, n-1$, for $p_{i-1} \le x \le p_i + 1$, solve the problem (1) with the two constraints: $b_i^*(p_{i-1}) = b_{i-1}^*(p_{i-1})$ and $b_i^*(p_{i-1}+1) = b_{i-1}^*(p_{i-1}+1)$, where b_i^* denotes the solution of the ith problem.
   (c) For $p_{n-1} \le x \le p_n (= N)$, solve the problem (1) with the two constraints: $b_n^*(p_{n-1}) = b_{n-1}^*(p_{n-1})$ and $b_n^*(p_{n-1}+1) = b_{n-1}^*(p_{n-1}+1)$. Let b_n^* denote the solution. Note that we separate (c) from (b) because when $i = n$, $p_i + 1$ should be changed to p_i.
2. Followup Iteration. For $i = 1, \ldots, n-1$, for the ith connecting-points group, $p_i - 2 \le x \le p_i + 3$, solve the problem (1) with the four constraints: $b_{c,i}^*(p_i - 2) = b_i^*(p_i - 2)$, $b_{c,i}^*(p_i - 1) = b_i^*(p_i - 1)$, $b_{c,i}^*(p_{i+1} + 2) = b_{i+1}^*(p_{i+1} + 2)$, $b_{c,i}^*(p_{i+1} + 3) = b_{i+1}^*(p_{i+1} + 3)$, where $b_{c,i}$ denotes the solution to the ith connecting-points group.
3. If $b_{c,i}^*(p_i)$ and $b_{c,i}^*(p_{i+1} + 1)$ are close enough to $b_i^*(p_i)$ and $b_i^*(p_{i+1} + 1)$ for all $i = 1, \ldots, n-1$, terminate. Otherwise, go to step 1 and repeat the above with the additional constraints $b_i^*(p_i) = b_{c,i}^*(p_i)$ and $b_i^*(p_i + 1) = b_{c,i}^*(p_i + 1)$, where $i = 1, \ldots, n-1$.

Figure 4 gives an illustration of the above divide-and-conquer strategy.

The above strategy is extremely useful when some computers do not have enough memory (RAM) to run the optimization toolbox CVX with high-dimensional curves. In fact in Sect. 3, to analyze 4990-dimensional tonnage data, we applied the above divide-and-conquer strategy with $p_1 = 1663$, $p_2 = 3326$, and $p_3 = 4990$. The result is quite satisfactory: in nearly all cases, the algorithm converges right after the first iteration.

3 Numerical experiments

We present a simulation study in Sect. 3.1. A real data case is presented in Sect. 3.2. In particular, IBC is compared with the fused lasso support vector classifier for the real data in Sect. 3.3.

3.1 Synthetic data

We consider two baseline functions: $f_1(t) = \frac{\sin(t)}{t}$ and $f_2(t) = -\frac{\sin(t)}{t}$. They are drawn in Fig. 5; the red (dashed) and blue (solid) curves are $f_1(t)$ and $f_2(t)$, respectively. To discretize, we assume equally spaced points in the domain $[-4\pi, 4\pi]$. We have two discrete baselines:

$$b_i(x) = f_i\left(-4\pi + \frac{x}{N} \cdot 8\pi\right), \quad x = 0, \ldots, N;\ i = 1, 2,$$

where N is the length of the curve. We generate the curves according to the following:

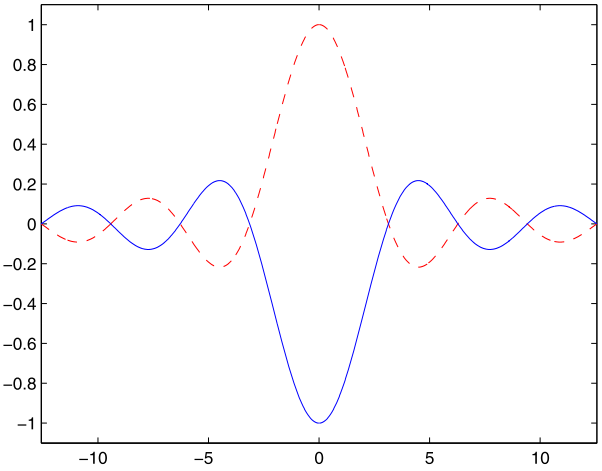

Fig. 5 (Color online) Two artificial baselines

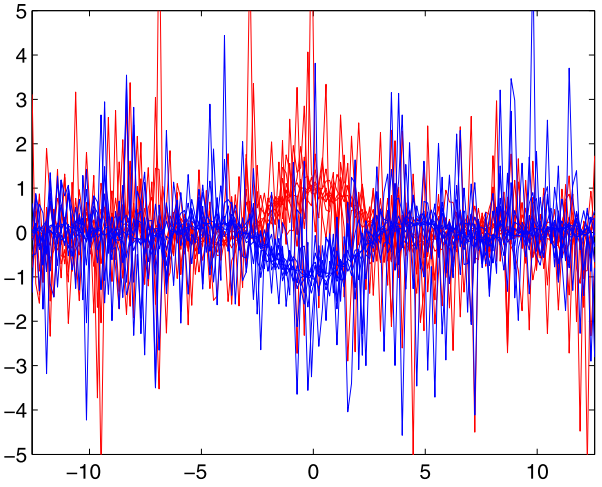

Fig. 6 Simulated data with $N = 155$, $\tau = 0.6$, $n = 10$. The curves are very noisy

1. Generate random variables $\sigma(x) \overset{\text{i.i.d.}}{\sim} \text{Exponential}(\tau),\ x = 0, 1, \ldots, N$;
2. Generate the curve $c(x) \overset{\text{i.i.d.}}{\sim} \text{Laplace}(b(x), \sigma(x)), x = 0, 1, \ldots, N$.

According to the above procedure, we generate n curves (i.e., $c(\cdot)$'s) for each group. Figure 6 illustrates the simulated data with $N = 155$, $\tau = 0.6$, $n = 10$. Note that the observed curves are extremely noisy. The proposed method is applied to this simulated data set. Figure 7 presents the two estimated baselines. Figure 8 shows the optimal value of objective function (i.e., $h(s)$) versus the length of interval (i.e., s), and the corresponding convex hull. Note that the number of admissible lengths is just 14 (out of 156). The full list of the admissible lengths and their corresponding optimal intervals is provided in Table 1. Note that when the length is 38, the objective value is maximized, and the optimal interval is $(-2.9997, 2.9997)$. This is exactly the "true" optimal interval.

We repeat the entire procedures for 40 times to verify the consistency. Figure 9 presents the optimal intervals for $n = 10, 30$, and 50; each horizontal line indicates the position of the computed optimal interval. Recall that visually, our simulated data (i.e., Fig. 6) can hardly be

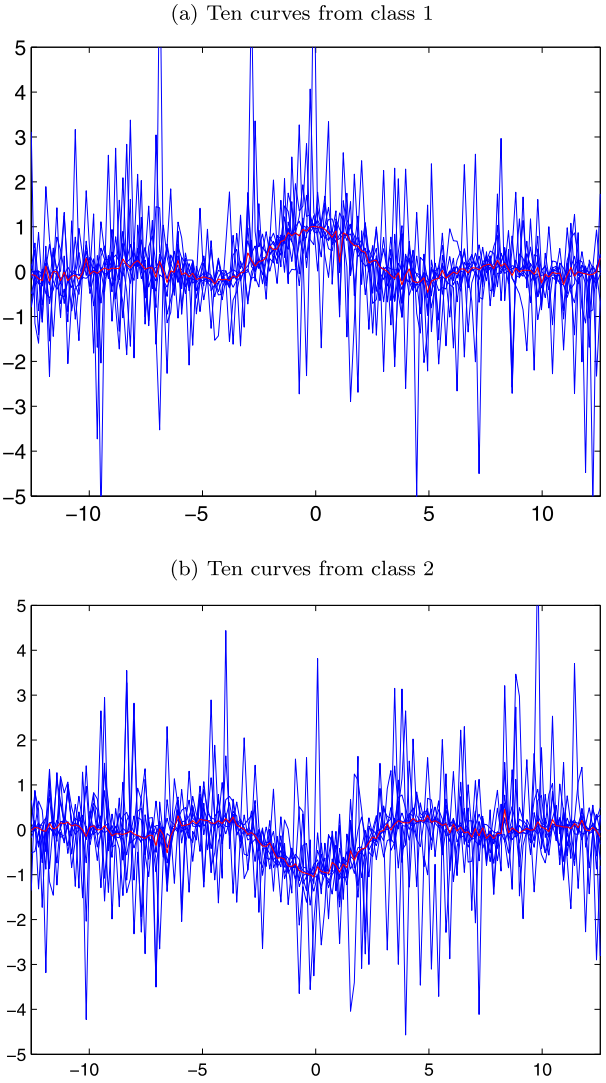

Fig. 7 (Color online) *Baseline curves* (*red*) for the two simulated groups. The curves are 156 dimensional. There are 10 curves in each group. In both classes, the estimated baselines are close to the true baselines

Table 1 Admissible lengths and the corresponding optimal intervals

Length	Start	End	Length	Start	End
1	84 (0.8918)	84 (0.8918)	28	65 (−2.1890)	92 (2.1890)
4	81 (0.4054)	84 (0.8918)	29	64 (−2.3511)	92 (2.1890)
5	80 (0.2432)	84 (0.8918)	30	63 (−2.5133)	92 (2.1890)
8	80 (0.2432)	87 (1.3782)	33	63 (−2.5133)	95 (2.6754)
20	68 (−1.7025)	87 (1.3782)	36	60 (−2.9997)	95 (2.6754)
22	66 (−2.0268)	87 (1.3782)	37	60 (−2.9997)	96 (2.8376)
23	65 (−2.1890)	87 (1.3782)	38	60 (−2.9997)	97 (2.9997)

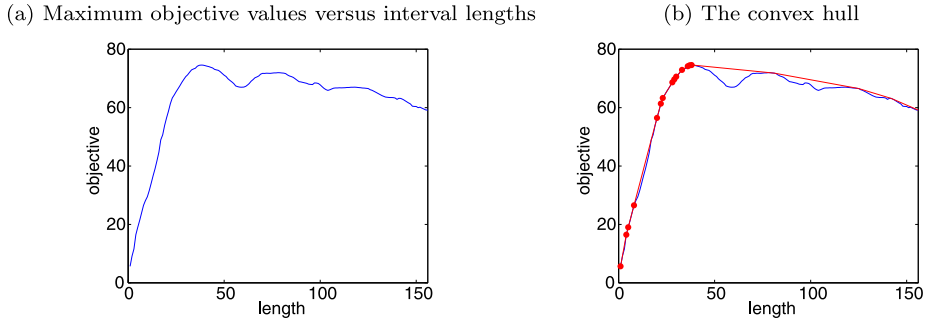

Fig. 8 The entire objective function (**a**) and its convex hull (**b**). Note that the number of vertices on the convex hull is much smaller than the maximum length

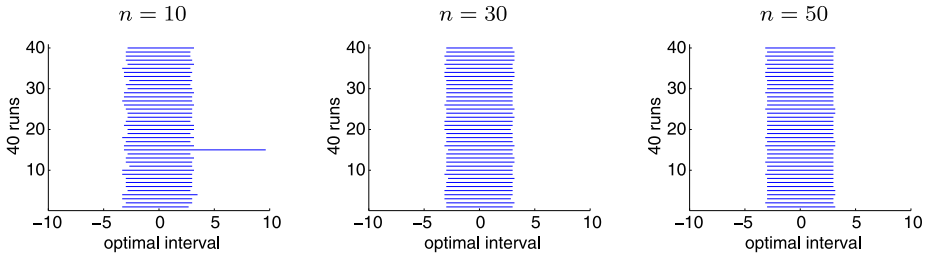

Fig. 9 Optimal intervals in 40 experiments with $\tau = 0.6$ and various n's. In most cases, the estimated optimal intervals are close to the true optimal interval

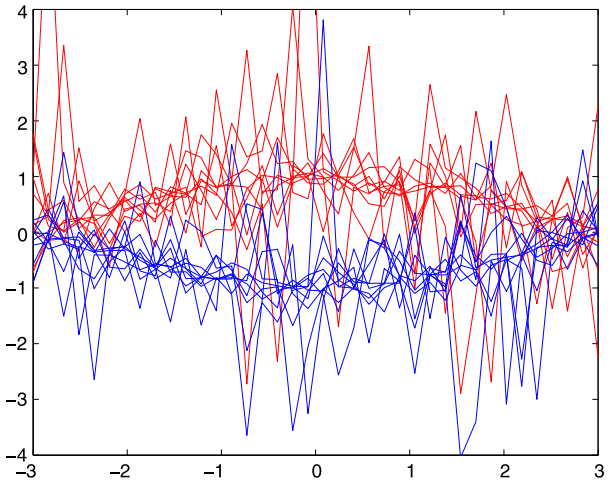

Fig. 10 Segments of curves within the computed optimal interval. We observe that the two classes of curves are clearly separated

distinguished. Figure 9 shows that if n is large enough, we can consistently find the optimal interval on which the curves are well separable. This demonstrates the consistency of our estimator. Figure 10 shows the separation of the two classes of simulated curves within the computed optimal interval. We observe that within the identified optimal interval, the two classes of curves are well separated.

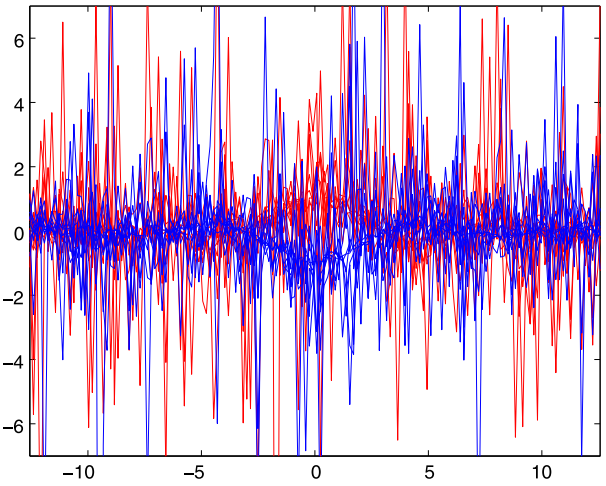

Fig. 11 Simulated data with $\tau = 1$, $n = 10$. This case is more difficult to classify than the previous one

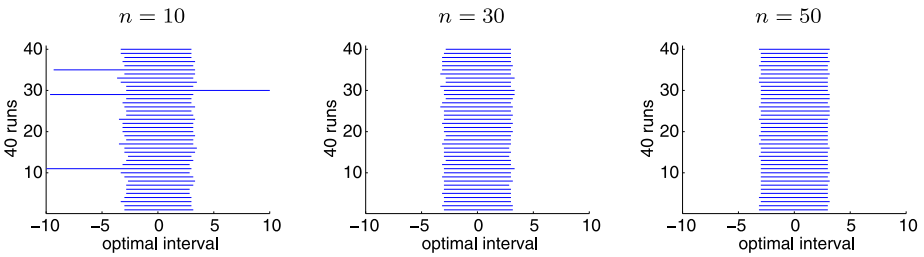

Fig. 12 Optimal intervals with $\tau = 1$ and various n's. Again the estimated optimal intervals are close to the true optimal interval

To check whether our method works well for worse case (i.e., noisier case), we increase the value of τ from 0.6 to 1. Figure 11 shows that it is more difficult to distinguish the curves than the previous case (i.e, $\tau = 0.6$). The same experiment is performed. Figure 12 shows that as n increases, our method becomes more consistent in locating the optimal interval for much noisier curves.

It will be also interesting to see the results for the case when N is large. For this purpose, we perform the same experiment for $N = 305, 605$, and 1205. We set $\tau = 1$ (i.e., very noisy case). The results on the optimal intervals are in Fig. 13.

Table 2 shows the execution time for our procedures. These procedures include all the steps, including finding baselines, computing the value of objective function, identifying an optimal interval. For 40 runs, we measure the execution time for each run, and then take the average. The execution time is quite impressive, considering the dimension of the curves and the numbers of samples.

3.2 A real case: tonnage data from a stamping process

The developed IBC has been tested with tonnage curves obtained from a real production environment. Two sets of curves under normal and faulty conditions are used. The stamped parts are manufactured from a progressive die. The total tonnage curve is measured as the summation of all forming operations within the press. The faulty condition is that one part

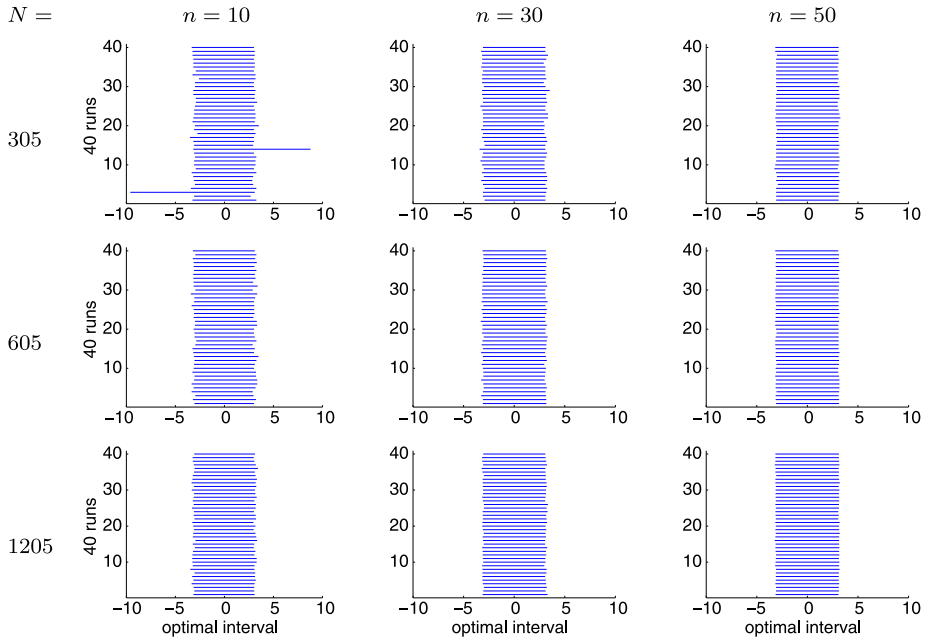

Fig. 13 Optimal intervals with $\tau = 1$ and various n's and N's

Table 2 Execution time (in seconds)

	$n = 10$	30	50
$N = 75$	0.8	2.1	4.2
155	1.3	4.3	9.0
305	2.4	9.1	20.4
605	4.6	16.6	39.9
1205	9.0	38.9	81.7

is missing in one of the die stations, which introduces some tonnage curve change in one of the segments. However, which segments (i.e., intervals) of the tonnage curve have the most sensitive difference is unknown. Such intervals are identified using the IBC. Figure 14 presents the baselines (in red) for the two classes. We highly recommend the readers to view this figure in color. It is more interesting to compare the two estimated baselines. Figure 15 presents a comparison. For the tonnage data, Fig. 16(a) shows the optimized objective value and the corresponding interval length, and Fig. 16(b) shows the convex hull. MATLAB offers built-in function "`convhull`" for calculating convex hull for a curve given by a set of points. Note that the number of admissible lengths is only 49, considering the number of possible lengths is 4990. A full list of admissible lengths and their corresponding optimal intervals is in Table 3. Note that if we are required to use intervals shorter than 35, then the optimal one is [808, 841] (with length 34; 35 is not admissible), which is very close to the interval [810, 845] that is a known optimal interval for separation based on physical knowledge. In this case, the output of IBC is consistent with the expert's knowledge.

To show the effectiveness of IBC, we compare the original curves and the segmented curves within the optimal interval in Fig. 1. Figure 1(a) plots all the curves ($N = 4990$

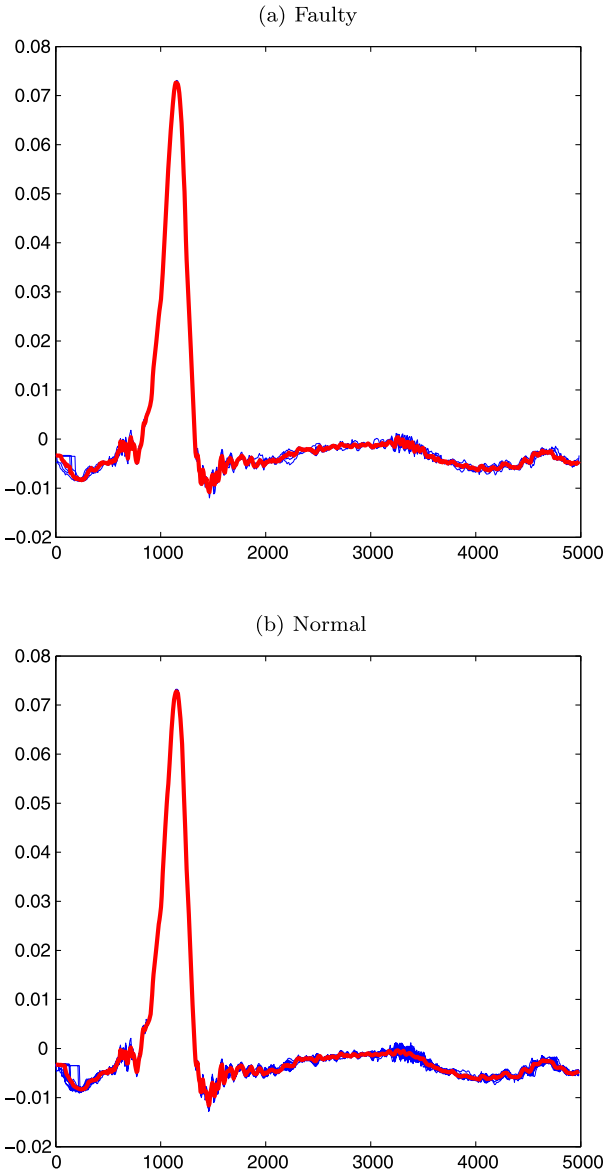

Fig. 14 (Color online) The computed baselines for faulty (**a**) and normal (**b**) tonnage curves respectively

dimensional) that are faulty (in red) and normal (in blue), respectively. Visually, the two classes are indistinguishable. However, in Fig. 1(b), the two classes have been observed to be well separated in interval (808, 841).

3.3 Comparison as a classifier

Using the identified optimal interval from the previous subsection and the classification rule as in Sect. 2.5, the tonnage curves are classified into two classes. The performance of IBC is verified in two ways: (a) We show that much lower misclassification error rate

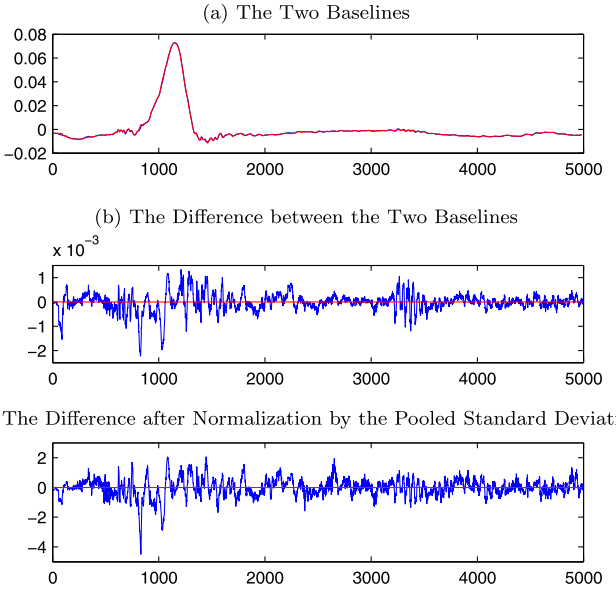

Fig. 15 A comparison of the two estimated baselines

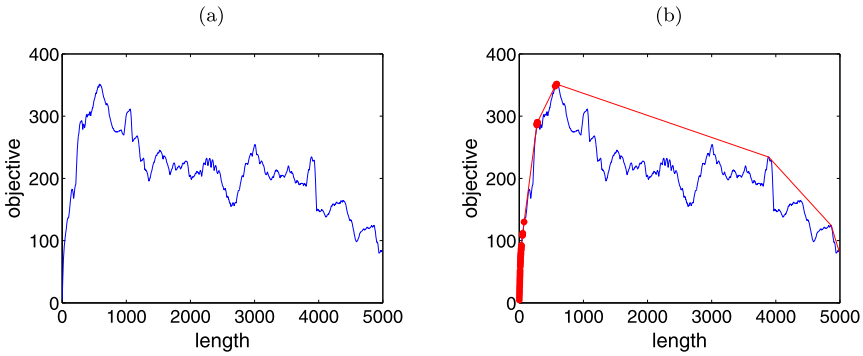

Fig. 16 (**a**) Maximum objective values versus interval lengths; (**b**) Convex hull

can be achieved by using data within the optimal interval rather than the entire data. (b) IBC produces similar solutions to those from the fused lasso support vector classifier, but is more stable, and guarantees the unique optimal interval.

We first examine the approach of using the optimal 'segment' of data. If we completely ignore the optimal interval, and carry out a classification on the entire domain, using the classification rule as in Sect. 2.5, the error rate can be much higher than doing so by combining with the optimal interval. In our experiment of 7-fold cross validation, all tonnage curves are divided into 7 groups. For each group, we use remaining 6 groups as training data, and the left-out group as testing data. In this scenario, 18 (out of 73) times the classification result is erroneous. Now if we insert the optimal interval selection immediately after the step of finding the baselines (i.e., before applying the classification rule), as shown in

Table 3 Admissible lengths and the corresponding optimal intervals

Length	Start	End	Length	Start	End	Length	Start	End
1	832	832	19	818	836	53	806	858
2	831	832	20	818	837	54	805	858
3	830	832	21	817	837	76	783	858
4	829	832	22	816	837	274	783	1056
5	828	832	23	815	837	275	782	1056
6	828	833	24	815	838	276	782	1057
7	827	833	25	814	838	278	780	1057
8	826	833	26	813	838	279	779	1057
9	826	834	28	811	838	280	779	1058
10	825	834	29	811	839	282	777	1058
11	824	834	30	811	840	283	777	1059
12	824	835	32	809	840	560	500	1059
13	823	835	33	808	840	581	479	1059
14	822	835	34	808	841	582	478	1059
16	820	835	50	808	857	585	478	1062
17	819	835	51	808	858			
18	818	835	52	807	858			

Table 4 Number of misclassified curves according to the various admissible lengths in IBC

Length	Number of errors	Length	Number of errors	Length	Number of errors
1	0	19	1	53	4
2	1	20	1	54	4
3	1	21	1	76	2
4	0	22	1	274	7
5	1	23	1	275	7
6	1	24	2	276	7
7	1	25	1	278	7
8	1	26	1	279	7
9	1	28	2	280	7
10	1	29	2	282	7
11	1	30	2	283	8
12	1	32	2	560	11
13	1	33	2	581	11
14	1	34	2	582	11
16	1	50	3	585	11
17	1	51	3		
18	1	52	4		

Table 4, the number of misclassified curves is much smaller. This experiment demonstrates that (at least in this setting), adding a step of selecting optimal interval is advantageous.

Secondly, IBC is compared with the fused lasso support vector classifier (SVC) that also produces interval-like solutions. For IBC, 7-fold cross-validation is repeated for every ad-

Table 5 Number of misclassified curves according to the various $C1$ and $C2$ in the fused lasso SVC. Note that the misclassification rate fluctuates, indicating difficulty in searching for a minimum

	$C2$									
C1	100	200	300	400	500	600	700	800	900	1000
100	34	37	46	16	34	7	7	7	7	7
200	34	34	43	46	37	19	55	46	16	7
300	28	25	7	7	52	46	28	46	28	28
400	16	25	16	16	16	34	25	46	37	37
500	7	25	16	16	7	34	16	34	16	34
600	16	16	16	16	19	16	7	16	34	34
700	6	24	7	16	16	7	16	7	7	7
800	4	4	24	23	4	4	4	4	4	4
900	3	3	12	3	2	3	35	13	3	3
1000	3	0	1	1	11	1	1	2	2	2

missible length (i.e., 49 times) so that we can choose the optimal interval that minimizes the number of misclassified curves. For the fused lasso SVC, the values of the two tuning parameters $C1$ and $C2$ (upper bounds for the L_1-norms of the coefficients and their successive differences, respectively) should be set in advance. To search the optimal values of $C1$ and $C2$, 7-fold cross-validation is repeated 100 times, as both $C1$ and $C2$ are varied from 100 to 1000 with increment 100. The values minimizing the number of misclassified curves are chosen as the optimal values of $C1$ and $C2$. Tables 4 and 5 show the results for IBC and the fused lasso SVC, respectively. We have two observations. One is that the results from IBC are very stable with various lengths, while those from the fused lasso SVC is rather bumpy. Indeed, when using IBC, among the 49 admissible lengths, 24 (roughly 50 percent) admissible lengths produce either 0 or 1 error. Moreover, most admissible lengths (37 out of 49) produce less than 4 errors. In contrast, the performance of the fused lasso SVC substantially depends on the values of $C1$ and $C2$, and therefore the choices of the two values are very critical. This implies bumpy response surface that is associated with the response in Table 5. Hence, finding optimal $(C1, C2)$ is not going to be easy.

Another observation is that while IBC produces the unique optimal interval given a length, the fused lasso SVC produces multiple intervals. In fact, with the optimal choices of $C1 = 1000$ and $C2 = 200$, the fused lasso SVC estimates the nonzero coefficients as: $\hat{\beta}_{824} = \cdots = \hat{\beta}_{833} = 50.79$, $\hat{\beta}_{834} = 1.05$, $\hat{\beta}_{1029} = \cdots \hat{\beta}_{1035} = 14.99$, $\hat{\beta}_{1082} = \cdots = \hat{\beta}_{1092} = -34.21$, $\hat{\beta}_{1093} = -9.71$. Note that the first interval (with the largest coefficients) is the same as the optimal interval from IBC with length 11, which suggests that (1) an underlying optimal interval does exist for this data set; (2) both methods (IBC & fused lasso SVC) recover such an interval in some sense. However, IBC explicitly searches for such an interval, while the fused lasso SVC does it without a guarantee. In some engineering problems, e.g., analyzing the stamping process, it is more convenient to adopt IBC instead of the fused lasso SVC.

4 Conclusion

We formulate a special class of classification problems (from data mining) into an optimization problem, and solve the problem with analytical study and demonstrate with real data.

In particular, we propose an interval based classifier. Its application is demonstrated by classifying some tonnage data. Fast numerical algorithms are derived. The proposed method is efficient—its order of complexity is nearly linear to the length of the curve. In applications of IBC, we demonstrate its speed and ability to deal with high-dimensional data. In many applications, problems that require interval based classifier may benefit from using IBC. Our computational method has a root in operations research. IBC should have many applications in data mining, when the features are based on single intervals. Our future research includes multiple (more than 2) classes scenario. In this case, a simple approach for identifying an optimal interval is to maximize the minimum of the pairwise distances of multiple baselines, while other criteria exist. However, if the number of class is large, this approach with a single optimal interval may not be effective; adopting multiple optimal intervals can be considered—this paper does not pursue further in this direction. The case of multiple optimal intervals and integration with other signal processing methods, e.g., wavelet transforms will be interesting future work.

Appendix: Proofs

Proof of Theorem 1 We show how an $O(N)$ algorithm can be constructed. First of all, the partial sums:

$$p_i = \sum_{x=1}^{i} \frac{\hat{b}_1(x) - \hat{b}_2(x)}{\sigma(x)}, \quad 1 \le i \le N,$$

can be done in $O(N)$, because we have

$$p_{i+1} = p_i + \frac{\hat{b}_1(i+1) - \hat{b}_2(i+1)}{\sigma(i+1)}, \quad 1 \le i \le N-1.$$

Secondly, differences $p_{i+s-1} - p_i$, $1 \le i \le N-s+1$ can be computed at $O(N)$. Evidently, taking the maximum among these differences can be done at $O(N)$. The above solves the following

$$\max_{1 \le i \le N-s+1} \sum_{x=i}^{i+s-1} \frac{\hat{b}_1(x) - \hat{b}_2(x)}{\sigma(x)}.$$

Now that compared to (2), there is no absolute value in the above objective function. To solve (2), we rerun the above for

$$\max_{1 \le i \le N-s+1} \sum_{x=i}^{i+s-1} \frac{\hat{b}_2(x) - \hat{b}_1(x)}{\sigma(x)};$$

(Note the switch of positions of $\hat{b}_1$ and $\hat{b}_2$ in the objective function) and take the maximum. □

References

Grant, M., Boyd, S., & Ye, Y. (2007). CVX: Matlab software for disciplined convex programming. Tech. rep., Stanford University, Stanford, CA. Version 1.1, Build 585, downloadable at http://www.stanford.edu/~boyd/cvx/.

Huo, X., Kim, S. B., Tsui, K. L., & Wang, S. (2006). FBP: A frontier-based tree pruning algorithm. *INFORMS Journal on Computing*, *18*(4), 494–505.

Jin, J., & Shi, J. (1999). Feature-preserving data compression of stamping tonnage information using wavelets. *Technometrics*, *41*(4), 327–339.

Jin, J., & Shi, J. (2005). Press tonnage signal decomposition and validation analysis for transfer or progressive die processes. *ASME Transactions, Journal of Manufacturing Science and Engineering*, *127*(1), 231–235.

Land, S. R., & Friedman, J. H. (1996). Variable fusion: a new method of adaptive signal regression. Tech. rep., Department of Statistics, Stanford University.

Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, *58*(1), 267–288.

Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., & Knight, K. (2005). Sparsity and smoothness via the fused Lasso. *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, *67*(1), 91–108.