An Algorithm for Optimal Partitioning of Data on an Interval

Brad Jackson, Jeffrey D. Scargle, David Barnes, Sundararajan Arabhi, Alina Alt, Peter Gioumousis, Elyus Gwin, Paungkaew Sangtrakulcharoen, Linda Tan, and Tun Tao Tsai

Abstract— **Many signal processing problems can be solved by maximizing the fitness of a segmented model over all possible partitions of the data interval. This letter describes a simple but powerful algorithm that searches the exponentially large space of partitions of N data points in time $O(N^2)$. The algorithm is guaranteed to find the exact global optimum, automatically determines the model order (the number of segments), has a convenient real-time mode, can be extended to higher dimensional data spaces, and solves a surprising variety of problems in signal detection and characterization, density estimation, cluster analysis and classification.**

Index Terms— **signal detection, density estimation, optimization, Bayesian modeling, histograms, cluster analysis**

EDICS: 1.STAT

I. INTRODUCTION: THE PROBLEM

A variety of signal processing and related problems can be viewed as the search for an optimal partition of data given on a time interval I. For example, one may estimate a segmented model by maximizing some measure of model fitness[1] defined on partitions of I. Since the space of all partitions of a continuum is infinite, it is advantageous to discretize the interval. Often the data points themselves, say

$$X_n, n = 1, 2, \ldots, N, \qquad (1)$$

naturally subdivide I into subintervals – which we call *data cells*. We avoid a precise definition because many types of data cells are possible. Common examples are counts in bins, measurements at a set of sample times (evenly spaced or not), and event or point data. The underlying idea is that restricting consideration to the finite space of partitions whose elements are sets of data cells will result in no significant loss of information or of resolution in the independent variable.

A *partition* **P** of an interval I is a set of M *blocks*,

$$\mathbf{P}(I) = \{B_m, m \in \mathcal{M}\}, \mathcal{M} \equiv \{1, 2, \ldots M\}, \qquad (2)$$

---

[1]This concept goes by many names, including *fitness*, *cost*, *goodness of fit*, *loss*, *penalty*, *objective function*, *risk etc.* Here we assume that the data analysis problem can be phrased in terms of some fitness that is to be maximized.

where the blocks are sets of data cells defined by index sets $\mathcal{N}_m$:

$$B_m = \{X_n, n \in \mathcal{N}_m\} \qquad (3)$$

satisfying the usual conditions, $\bigcup_m B_m = I$ and $B_m \bigcap B_{m'} = \emptyset$ if $m \neq m'$. Here $\emptyset$ means a set of zero area, since the 1D boundaries are not relevant here. We impose connectedness too—i.e. that there be no gaps between the cells comprising a given block. M, the number of blocks, must satisfy $0 \leq M \leq N$. Partitions will be denoted in boldface, and refer to the interval I unless otherwise stated. Define P^* to be the (finite) set of all partitions of I into blocks.

Take as given an additive *fitness function* that assigns a value to any partition $\mathbf{P} \in P^*$ in the form

$$V(\mathbf{P}) = \sum_{m=1}^{M} g(B_m), \qquad (4)$$

where $g(B_m)$ is the fitness of block B_m. Computationally, the data cells must be represented by a data structure that contains *sufficient statistics* for the model – *i.e.* all information necessary to determine g for any block.

We exhibit an efficient $O(N^2)$ dynamic programming algorithm that finds an *optimal partition* $\mathbf{P}^{\max} \in P^*$: $V(\mathbf{P}^{\max}) \geq V(\mathbf{P})$ for all partitions $\mathbf{P} \in P^*$.

Scargle [21] proposed two greedy iterative algorithms for finding near-optimal partitions: one top-down (optimally divide I into two parts; recursively do the same to each such part) the other bottom-up (merge adjacent data cells). In both cases Bayesian model comparison provides effective fitness functions and halting criteria, implementing an $O(N^2)$ procedure for data spaces of 1, 2 and higher dimensions—hence the term *Bayesian Blocks* [21]. But in practice these greedy algorithms often find significantly suboptimal partitions, motivating the development reported here.

The dynamic programming idea for this kind of problem seems to have originated with Richard Bellman [6], a paper which influenced Kay's work [12], [13], [14]. An extensive discussion of precisely the same problem addressed here, but with a different approach to its solution, is in [3], [4]. Work by Hubert [10], [11], with applications to meteorology, influenced Kehagias and co-workers [8], [15], [16], [17], [18], [19], who developed a dynamic programming algorithm much like ours, for applications such as text segmentation (see also [9]), where the raw data are provided in the form of a similarity matrix. [22] gives an $O(kN^2)$ dynamic programming algorithm for finding the optimal partition of an interval into k blocks, for a

given k. See also [20], [2] for related work. Thus while similar algorithms have been previously developed, ours finds the exact global optimum for any block-additive fitness function, automatically determines the number of segments, and can be used in either real-time or retrospective analysis.

II. Dynamic Programming: Finding Optimal Partitions

We describe an $O(N^2)$ algorithm that is guaranteed to solve the above problem by finding an exact global optimum, for any fitness function V that is additive in the sense of Eq. (4). There is a large (2^{N-1}) but finite number of partitions in P^*. Dynamic programming [5] is an intelligent method of searching this space of all possible solutions. Our algorithm can be applied whenever any subpartition of an optimal partition is optimal.

Theorem 1 (Principle of Optimality): Let $\mathbf{P}^{\max}$ be an optimal partition of I and $\mathbf{P}_1 = \{B_m, m \in a\}$ be any subset of the blocks of $\mathbf{P}^{\max}$. Then $\mathbf{P}_1$ is an optimal partition of the part of I it covers, namely $I_1 = \bigcup_{m \in a} B_m$.

Intuitively this result follows from the contradiction that a better subpartition of I_1 could be used to construct a partition of I better than $\mathbf{P}^{\max}$. The proof relies on the fact that the block-additivity of the fitness function implies that it is also additive on subpartitions. To see this, divide partition $\mathbf{P}$ into any two disjoint parts, $\mathbf{P}_1 = \{B_m, m \in a\}$ and $\mathbf{P}_2 = \{B_m, m \in b\}$, with $\mathbf{P}_1 \bigcup \mathbf{P}_2 = \mathbf{P}$ and $a \bigcup b = \mathcal{M}$. Then the additivity of V yields

$$
\begin{aligned}
V(\mathbf{P}) &= \sum_{m=1}^{M} g(B_m) \\
&= \sum_{m \in a} g(B_m) + \sum_{m \in b} g(B_m) \\
&= V(\mathbf{P}_1) + V(\mathbf{P}_2).
\end{aligned} \tag{5}
$$

Proof 1: As above, denote by $\mathbf{P}_2$ the subpartition of $\mathbf{P}^{\max}$, consisting of the blocks $\{B_m, m \in \mathcal{M} - a\}$ in $\mathbf{P}^{\max}$ that are not in $\mathbf{P}_1$. Let $\mathbf{P}_3$ be any other partition of I_1. Since $\mathbf{P}^{\max}$ is an optimal partition of I and $\mathbf{P}_3 \cup \mathbf{P}_2$ is also a partition of I it follows that $V(\mathbf{P}^{\max}) = V(\mathbf{P}_1) + V(\mathbf{P}_2) \geq V(\mathbf{P}_3 \cup \mathbf{P}_2) = V(\mathbf{P}_3) + V(\mathbf{P}_2)$ so $V(\mathbf{P}_1) \geq V(\mathbf{P}_3)$. Thus $\mathbf{P}_1$ is an optimal partition of I_1.

Dynamic programming is a recursive procedure that can be used to efficiently find the solution to many kinds of combinatorial optimization problems. Our algorithm derives the optimal partition of the first $n + 1$ data points using previously obtained optimal partitions, *i.e.* those of the first $1, 2, \ldots n$ data points. At each iteration we must consider all possible starting locations j, $1 \leq j \leq n$ of the last block of the optimal partition. For each putative j the fitness function is – by the principle of optimality – the fitness of the optimal subpartition prior to j plus the fitness of the last block itself. The former was stored at previous iterations, and the latter is a simple evaluation of V. The desired new optimal partition corresponds to the maximum over all j.

More precisely, define $opt(n)$ to be the value of the fitness function of the optimal partition $\mathbf{P}_n^{\max}$ of the first n cells of I, for $1 \leq n \leq N$. The following dynamic programming algorithm finds the optimal partition $\mathbf{P}_N^{\max}$:

1) Define $opt(0) = 0$
2) Given that $opt(j)$ has been determined for $j = 0, 1, \ldots, n$:
   - Define $end(j, n + 1) = g(B_{j,n+1})$; $B_{j,n+1}$ is the union of cells $j, j + 1, \ldots, n + 1$
   - Then compute

     $$opt(n+1) = \operatorname*{Max}_{j}\{opt(j-1) + end(j, n+1)\}, \tag{6}$$

     for $j = 1, 2, \ldots, n + 1$.
   - The value of j where this maximum occurs is stored as $lastchange(n + 1)$.
3) Repeat 2 until $n + 1 = N$, when $opt(N)$, the optimal partition fitness for all N cells, has been obtained.
4) Backtrack using the *lastchange* vector to identify the start points of individual blocks of the optimal partition $\mathbf{P}^{\max}$ in the following way. Let $n_1 = lastchange(N), n_2 = lastchange(n_1 - 1)$, *etc*. Then the last block in $\mathbf{P}^{\max}$ contains cells $n_1, n_1 + 1, \ldots, N$, the next-to-last block in $\mathbf{P}^{\max}$ contains cells $n_2, n_2 + 1, \ldots, n_1 - 1$, and so on.

Theorem 2: This deterministic $O(N^2)$ dynamic programming algorithm finds the partition of I that maximizes the (additive) fitness function.

Proof 2: The proof is by mathematical induction. Clearly $opt(1) = \text{Max}\{0 + end(1, 1)\} = g(B_{1,1})$ is the fitness of the only possible (and therefore optimal) partition of the set comprising the first cell. At iteration $n + 1$, assume not only that we have found the optimum partition of $\mathbf{P}_n^{\max}$, but also that for $i = 1, 2, \ldots, n$, we have stored the corresponding fitness for this and all previous iterations in the array $opt(i)$, and the index of the cell beginning this partition's last block in array $lastchange(i)$. Let $F(j) = opt(j-1) + end(j, n+1)$; then the principle of optimality shows that when j indexes the first cell of the last block of the desired partition $\mathbf{P}_{n+1}^{\max}$, $F(j)$ is the corresponding maximum fitness. Further, for any j, $F(j)$ is the fitness of a legitimate partition of $\mathbf{P}_{n+1}^{\max}$, namely that consisting of the optimal partition of the cells prior to j followed by the single block $B_{j,n+1}$. These two facts combine to prove that the maximum of $F(j)$ specified in Eq. (6) gives the desired optimum partition at iteration $n + 1$. Identification of the corresponding optimal blocks – starting with the last one and working backwards, as in part (4) of the algorithm – can be validated with straightforward recursive application of the principle of optimality. Finally, since $end(j, n + 1) = g(B_{j,n+1})$ the algorithm requires $1 + 2 + \ldots + N = O(N^2)$ evaluations of the function g. It also requires $O(N^2)$ additions and $O(N^2)$ comparisons in determining the maximums.

III. Applications

These results apply to any segmented modeling of 1-dimensional data defined by a fitness function that satisfies Eq. (4). Piecewise constant, or step functions form the most natural model class. However the nature of the model depends on the application, and many other forms are possible, including

piecewise linear and piecewise exponential. The key is that the fitness function must not depend on any model parameters other than the changepoint locations. We have found excellent results with the posterior probability of the model for each segment, given the data in that segment, marginalized over all parameters but these locations [21]. Using logs turns the product posterior (resulting from statistical independence of the blocks) into the required additive form Eq.(4).

A comment about smoothness constraints is in order. With some fitness functions the algorithm produces the degenerate solution assigning a segment to each data point. In the Bayesian setting described above, a natural way to address this problem is to adopt a prior distribution for the number of segments, for example giving higher weight to smaller numbers. Indeed, the geometric prior [7] corresponds to a constant term in the fitness function for each block, so there is no problem with maintaining additivity. This artifice controls model complexity, but without imposing an explicit smoothness condition with concomitant loss of time resolution. Time series features on any time scale, no matter how short, can be found if the data support them in a statistically significant way.

Finally, we mention a few sample applications. Implementing density estimation with piecewise constant Poisson models yields histograms in which the bins are not constrained to be equal. The number of bins and their sizes and locations are determined by the data. The same model provides denoising and structure estimation for time series of events or counts of events in bins [21].

Further, almost all of the results described here can be easily extended – almost without change – to data of higher dimensionality, as will be described in future papers. In this setting cluster analysis can be effected as a post-processing of segmented models – piecing the blocks together into clusters – and similarly with unsupervised classification and other data mining procedures.

IV. Conclusion

As we have seen dynamic programming gives a good (polynomial) algorithm for finding an optimal partition of data on an interval for any fitness function V satisfying the additive property [see Eq.(4)]. Ironically it has the same $O(N^2)$ complexity as the greedy algorithm.

In comparing the use of our algorithm to detect and characterize clusters (collections of blocks) with some of the standard clustering techniques [1], we note that our method inherently compares partitions that have different numbers of blocks, so the number of blocks is automatically determined by the data. This is to be contrasted with most standard clustering techniques, in which k, the fixed number of clusters must be specified ahead of time. One often seeks to minimize the maximum diameter (defined as the maximum distance between any pair of points in the cluster) of the clusters, or to maximize the minimum separation between the clusters. In dimension 1, there are well-known $O(kN^2)$ dynamic programming algorithms for finding the best partitions into k clusters. For dimension 2 and higher it is known that these standard

problems are NP-complete. We don't yet know if our problem is NP-complete in dimension 2 and higher.

In addition, considered as a density estimation or signal detection technique, our approach does not introduce any explicit smoothing of the data. Structure on any time scale, no matter how short, will be detected if it is supported by the data. While the parameter in the geometric prior discussed above controls to some extent the number of blocks – and thus affects the roughness of the optimized model – it is not explicitly a smoothing parameter. Another feature is that the incremental way the algorithm operates on the data makes a real-time mode trivial to implement. This mode has found to be very useful in the rapid detection of change points in a data stream. And since $opt(i+1)$ is calculated from $opt(j), j = 1, 2, \ldots, i$, some of the necessary calculations can be performed as the data are still being collected. Also it is easy to modify the dynamic programming to yield the optimal partition with blocks of a minimum size (each block contains at least d data points, for a given positive integer d). These and other features are described in more detail at an algorithm repository at:

`http://astrophysics.arc.nasa.gov/~pgazis/CodeArchiveServer.html`

References

[1] C. J. Alpert and A. B. Kahng, Splitting Orderings into Multi-way Partitionings to Minimize the Maximum Diameter, *Journal of Classification*, **14**, 1997, 51-74.

[2] I. Auger and C. Lawrence, Algorithms for the optimal identification of segment neighborhoods, *Bull. of Math. Biology*, **51**, 1989, 39-54.

[3] D. Barry and J. A. Hartigan. Product partition models for change point problems. *Ann. of Stat.*, **20**, 1992, 260-279.

[4] D. Barry and J. A. Hartigan, A Bayesian analysis for change point problems. *J. Amer. Statist. Assoc.* **88**, 1993, 309-319.

[5] R. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, 1957.

[6] R. Bellman, On the Approximation of Curves by Line Segments Using Dynamic Programming. *Communications of the ACM*, **4**, 1961, 284.

[7] M. A. Coram *Nonparametric Bayesian Classification*, Ph.D. thesis, Department of Statistics, Stanford University, 2002.

[8] P. Fragkou, V. Petridis, and A. Kehagias, A Dynamic Programming Algorithm for Linear Text Segmentation, to appear in Journal of Intelligent Information Systems.

[9] O. Heinonen, Optimal Multi-Paragraph Text Segmentation by Dynamic Programming, Proceedings of COLING-ACL '98, pp. 1484-1486, Montreal, Canada `http://arXiv.org/abs/cs/9812005`

[10] P. Hubert, Change points in meteorological time series, in *Applications of Time Series Analysis in Astronomy and Meteorology*, T. Subba Rao, M. Priestley, and O. Lessi, eds., 1997, Chapman and Hall.

[11] P. Hubert, The segmentation procedure as a tool for discrete modeling of hydrometeorogical regimes, *Stoch. Env. Res. and Risk Ass.*, **14**, 2000, 297-304.

[12] S. Kay, Optimal Segmentation of Time Series Based on Dynamic Programming, unpublished manuscript, 1988?

[13] S. M. Kay, *Fundamentals of Statistical Signal Processing: Detection Theory*, Englewood Cliffs, NJ: Prentice-Hall, 1998 (see p. 449).

[14] S. Kay and X. Han, Optimal Segmentation of Signals Based on Dynamic Programming and Its Application to Image Denoising and Edge Detection, unpublished manuscript, 2002?

[15] A. Kehagias and V. Petridis, Time Series Segmentation using Predictive Modular Neural Networks, *Neural Computation*, **9**, 1997, 1691-1710.

[16] A. Kehagias, Hidden Markov model segmentation of hydrological and environmental time series (2002), technical report: `http://arxiv.org/abs/cs/0206039`

[17] A. Kehagias, P. Fragkou and V. Petridis, Linear Text Segmentation using a Dynamic Programming Algorithm, in Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics, Budapest, Hungary, 2003, 171-178.

[18] A. Kehagias, A hidden Markov model segmentation procedure for hydrological and environmental time series, *Stochastic Environmental Research and Risk Assessment*, **18**, 2004, 117-130.

[19] A. Kehagias, A. Nicolaou, P. Fragkou and V. Petridis, Text Segmentation by Product Partition Models and Dynamic Programming, *Mathematical and Computer Modeling*, **39**, 2004, 209-217.

[20] F. Quintana and P. Iglesias, Bayesian Clustering and Product Partition Models, *Journal of the Royal Statistical Society Series B*, **65**, 557 - 574, 2003.

[21] J. Scargle, *Studies in Astronomical Time Series Analysis. V. Bayesian Blocks, A New Method to Analyze Structure in Photon Counting Data*, The Astrophysical Journal, (504), 1998, 405.

[22] R. Vidal, *Optimal Partition of an Interval*, Applied Simulated Annealing, Springer-Verlag, New York, 1993, 291.