Finding recurrent sources in sequences

Aristides Gionis*
Department of Computer Science
Stanford University
Stanford, CA, 94305, USA
gionis@cs.stanford.edu

Heikki Mannila
HIIT Basic Research Unit
Department of Computer Science
University of Helsinki
P.O. Box 26, Teollisuuskatu 23
FIN-00014 Helsinki, Finland
Heikki.Mannila@cs.helsinki.fi

ABSTRACT

Many genomic sequences and, more generally, (multivariate) time series display tremendous variability. However, often it is reasonable to assume that the sequence is actually generated by or assembled from a small number of sources, each of which might contribute several segments to the sequence. That is, there are h hidden sources such that the sequence can be written as a concatenation of $k > h$ pieces, each of which stems from one of the h sources. We define this (k, h)-segmentation problem and show that it is NP-hard in the general case. We give approximation algorithms achieving approximation ratios of 3 for the L_1 error measure and $\sqrt{5}$ for the L_2 error measure, and generalize the results to higher dimensions. We give empirical results on real (chromosome 22) and artificial data showing that the methods work well in practice.

Categories and Subject Descriptors

F.2.2 [**Analysis of algorithms and problem complexity**]: Nonnumerical algorithms and problems—*Computations on discrete structures, Geometrical problems and computations*; J.3 [**Life and medical sciences**]: Biology and genetics

General Terms

Algorithms

1. INTRODUCTION

Many genomic sequences and, more generally, (multivariate) time series display tremendous variability. In many cases it is interesting to find out whether the sequence can be viewed as consisting of segments coming from a small number of different sources. That is, we want to examine whether there are h hidden sources such that the sequence T can be written as a concatenation of $k > h$ pieces each of which can be viewed as coming from one of the h sources.

For genomic analyses, the segments might be viral or microbial inserts, stemming from a small number of possible sources. For instance, Azad et al [2] try to identify a coarse–grained description of the given DNA string in terms of a smaller set of distinct domain labels. The biological hypothesis behind this is that a mosaic organization of DNA sequences could have originated from the insertion of fragments of one genome (the parasite) inside another (the host). For time series type of data, the segments would reflect the generating process having h different states, with arbitrary (but relatively rare) transitions between the states.

The segmentation problem for genome sequences and time series has been discussed widely; see, e.g., [10, 3, 7, 9, 5, 16, 14, 15, 11, 4, 2]. For a wide variety of score functions, dynamic programming can be used to obtain the best segmentation of an n-element sequence into k pieces in time $O(n^2k)$; this idea goes back at least to Bellman [3] in 1961. The problem with this approach is that the descriptions for each segment are independent. That is, we get as many sources as there are segments. In many applcations it makes more sense to assume that the number of sources is (much) smaller than the number of segments. A complex dynamic process can have a small set of underlying states, and the number of state transitions is far larger than the number of states. Similarly, a genome can have an evolutionary structure with only a few different sources, each source contributing several segments.

Our problem is thus to find a good way of segmenting an n-element sequence into k segments, each of which comes from one of h different sources. We call this the (k, h)-*segmentation problem*. The normal dynamic programming approaches to sequence segmentation ignore the constraint of having less sources than segments and solve the (k, k)-segmentation problem. When $h < k$, the (k, h)-segmentation problem cannot be solved using that method: the restriction to h different sources means that decisions made early in the sequence have an impact later, invalidating the optimality condition needed for dynamic programming. The (k, h)-segmentation problem turns out to be algorithmically quite interesting.

Formally, the (k, h)-segmentation problem is defined as follows. Suppose we are given a sequence $T = (t_1, \ldots, t_n)$,

*Supported by a Microsoft Research Fellowship. This work was done while the author was visiting the HIIT Basic Research Unit, Department of Computer Science, University of Helsinki, Finland.

RECOMB'03, April 10–13, 2003, Berlin, Germany.

where $t_i \in \Sigma$. A segmentation $\mathcal{M}$ of T is defined by $k+1$ segment boundaries $1 = b_1 < b_2 < \cdots < b_k < b_{k+1} = n+1$, yielding segments $S_1, \ldots, S_k$ where $S_i = (t_{b_i}, \ldots, t_{b_{i+1}-1})$. Denote the possible sources (or states of the process) by Γ. Given a source $\gamma \in \Gamma$ and a segment S_i, denote by $P(S_i \mid \gamma)$ the likelihood that the sequence S_i comes from source γ.

The problem is to find h sources $\gamma_1, \ldots, \gamma_h$, a decomposition of T into k segments $S_1, \ldots, S_k$, and for each segment S_i an assignment of a source $\gamma_{j_i} \in \{\gamma_1, \ldots, \gamma_h\}$ such that the total probability $\prod_{i=1}^{k} P(S_i \mid \gamma_{j_i})$ is maximized.

The above formulation is in terms of an arbitrary likelihood function. We mostly consider the case of sequence of points from R^d and the problem of finding (k,k)-segmentations for them. We next discuss how the two formulations are related. If the sequence consists of n points in R^d, we can take Γ to be R^d, and say that the loglikelihood of an observation t_i stemming from source γ (a point in R^d) is proportional to $-||t_i - \gamma||^2$. Then, given an interval $[a,b]$, the loglikelihood of the subsequence $S[a,b]$ is proportional to $-\sum_{i=a}^{b} ||t_i - \gamma||^2$. Thus the best source for the interval is the one that maximizes the probability $P(S[a,b] \mid \gamma_{j_i})$, or equivalently the minimizer of the sum of squares error $\sum_{i=a}^{b} ||t_i - \gamma||^2$. The latter turns out to be the mean of the data points in the interval $S[a,b]$.

Denote by $Var(S_i)$ the variance of segment S_i, and by $|S_i|$ the length (number of points) of S_i. Then the (k,k)-segmentation problem is equivalent to finding the k segments such that the sum $\sum_{i=1}^{k} |S_i| Var(S_i)$ is minimized; the best source for each segment is simply the mean of the segment, and the error per point is the variance of the segment. Instead of the L_2-metric, we can of course consider arbitrary L_p-metrics. For each segment, the (k,k)-segmentation problem can use the source that minimizes the error, whereas in the (k,h)-segmentation problem with $h < k$ several segments have to use the same source.

While the (k,k)-segmentation problem has been considered often, to the best of our knowledge the problem of (k,h)-segmentation has not been studied extensively. In 1989 Churchill [6] stated a related problem for the purpose of partitioning genomic sequences into segments. In his formulation he uses hidden Markov states to model different compositional properties within each DNA segment, and his solution uses the Viterbi algorithm to determine the most probable sequence of states. More recently, Azad et al [2] formulate a similar problem to (k,h)-segmentation but their solution is based on greedy "split and merge" and it does not provide any theoretical guarantee.

An alternative way to view the (k,h)-segmentation problem is as a clustering problem with added constraints: the task is to cluster the points into h clusters, with the restriction that along the dimension of the sequence the cluster may change at most $k-1$ times.

In this paper we study the (k,h)-segmentation problem and apply it to several domains. First, we show that the (k,h)-segmentation for sequences of points from R^d is NP-hard for $d > 1$, both for the L_1 and L_2 metrics. For the case $d = 1$, it is not known whether the problem is NP-hard or not. Nevertheless, for $d = 1$ we give a 3-approximation algorithms for the L_1 error metric and a $\sqrt{5}$-approximation algorithm for the L_2 error metric. For general d, we get a $3+\epsilon$ approximation ratio for L_1 and $\alpha+2$ for L_2, where α is the best approximation factor for the k-means clustering problem.

These approximation algorithms are simple and easy to implement, while the analysis is slightly complex. The methods are based on using instances of dynamic programming for the polynomially solvable cases of the problem. The algorithms can be applied to any a polynomial-time computable likelihood function $P(S \mid \gamma)$, as long as we can, given a segment S from the sequence, find the source γ maximizing $P(S \mid \gamma)$; the details of the approximation guarantees depend on the exact form of the likelihood function.

We give empirical data both for artificial and for real sequences. The artificial data is generated from a process with a small number of hidden states; even for 0-1 sequences, our algorithms produce very good approximations of the hidden states. A typical empirical approximation error is about 4%.

It seems to us that one of the main applications of the (k,h)-segmentation problem is in genomic sequences. As an example, we study the distribution of w-letter words (e.g. $w = 2, 3$) in blocks of length 500 kb in the human genome. That is, given the genome sequence, we look at each 500 kb block in it, and compute for each block the frequency of each of the w-letter words over A, C, G, T. These word frequencies then form our sequence of dimension 4^w (16 or 64), and we search for recurrent states in this sequence. We apply the Bayesian information criterion (BIC, see [17, 10]) to search for the values of k and h yielding the best segmentation. In addition, we apply our segmentation algorithms on the chromosome sequence that describes the densities of SNPs and genes in each block. We also examine combinations of both of the above densities giving to each an equal weight. The results show that the methods provide useful segmentations that can be applied in the search for large scale structure in the genome (see Section 5.2).

The rest of this paper is organized as follows. In Section 2 we define the (k,h)-segmentation problem formally, give some basic observations, and show that the problem is NP-hard in general. In Section 3 we give three algorithms for the problem and describe the approximability results; the proofs are in Section 4. Experimental results are described in Section 5, and Section 6 is a short conclusion.

2. PRELIMINARIES

As noted in the introduction, our algorithms can be applied to finding optimal (k,h)-segmentations for any likelihood function satisfying the following two mild conditions: First, for every set of points coming from a single source we should be able to compute the optimal source that maximizes the likelihood function. Second, the total likelihood for a segmentation of the whole sequence should be an associative combination function (e.g. sum or product) of the individual likelihoods on each segment.

For simplicity of notation, we consider in the sequel the case of multidimensional real-valued sequences with the L_p distances. Let $T = (t_1, \ldots, t_n)$ be a sequence with n data points from R^d. The L_p distance between any two data points t_i and t_j of the sequence is defined as

$$d_p(t_i, t_j) = (\sum_{k=1}^{d} |t_{ik} - t_{jk}|^p)^{\frac{1}{p}}$$

where $(t_{i1}, \ldots, t_{id})$ and $(t_{j1}, \ldots, t_{jd})$ are the coordinates of the two d-dimensional points t_i and t_j. The most commonly used distances are the L_1 (Manhattan), the L_2 (Euclidean), and the L_∞ (L-infinity). In the one dimension all of the

above distances are simply $|t_i - t_j|$. In the rest of the paper when the subscript p is ommited the Euclidean distance is implied, i.e. $d(t_i, t_j) = d_2(t_i, t_j)$, but most of our algorithms can be easily modified so that other distance functions can also be used.

A *k-segmentation* $\mathcal{M}$ of a sequence T is defined by $k+1$ indices $\{b_1, b_2, \ldots, b_k, b_{k+1}\}$ that divides T into k segments, i.e., with $1 = b_1 < b_2 < \ldots < b_k < b_{k+1} = n+1$. We write $\mathcal{M} \in \text{Segm}_k(T)$ to denote that $\mathcal{M}$ is a k-segmentation of T. A k-segmentation $\mathcal{M}$ is also described by the sequence of segments to which it partitions the original sequence, that is, $\mathcal{M} = (S_1, \ldots, S_k)$, where $S_i = (t_{b_i}, \ldots, t_{b_{i+1}} - 1)$ for all $i = 1, \ldots, k$. We write $S \in \mathcal{M}$ for a segment in segmentation $\mathcal{M}$, and $t \in S$ for a point in segment S.

Given the original sequence T, our goal is to find a segmentation $\mathcal{M}$ such that each of the segments $S \in \mathcal{M}$ is homogeneous with respect to an error measure. Two natural choices for error measures within a segment S are the 1-error measure:

$$E_1(S) = \min_{x \in R^d} \sum_{t \in S} d(t, x)$$

and the 2-error measure:

$$E_2(S) = \min_{x \in R^d} \sqrt{\sum_{t \in S} d(t, x)^2}$$

Minimizing these error measures can also be obtained by representing the segment S with a single representative $\bar{t}_S$, chosen so that it minimizes the error measure. In other words, $E_1(S) = \sum_{t \in S} d(t, \bar{t}_S)$ and $E_2^2(S) = \sum_{t \in S} d(t, \bar{t}_S)^2$. It is easy to see that for $E_1(S)$, the representative $\bar{t}_S$ will be the *median* of the points in S, while for $E_2(S)$ it will be the *mean* of the points in S. We write $E_p(\cdot)$ when we don't want to distinguish between $E_1(\cdot)$ and $E_2(\cdot)$. The traditional sequence segmentation problem can now be formalized as follows.

PROBLEM 1 ((k,k)-SEGMENTATION$_p$). *For a sequence T, a k-segmentation $\mathcal{M}$ of T, and $p = 1, 2$, we define the p-error of a segmentation $\mathcal{M}$ to be*

$$E_p(T, \mathcal{M}) = \left(\sum_{S \in \mathcal{M}} E_p^p(S) \right)^{\frac{1}{p}} = \left(\sum_{S \in \mathcal{M}} \sum_{t \in S} d(t, \bar{t}_S)^p \right)^{\frac{1}{p}}$$

The task is to find the segmentation $\mathcal{M}$ that minimizes the above error measure. The minimum error is denoted by $Seg_p(T, k)$, i.e.

$$Seg_p(T, k) = \min_{\mathcal{M} \in \text{Segm}_k(T)} E_p(T, \mathcal{M})$$

This problem can, of course, be solved in polynomial time using dynamic programming as shown in 1961 by Bellman [3].

As noted in the introduction, the drawback in this formulation is that the segments are completely independent: we have no way of preferring solutions that would use only a few types of segments. That is, in (k,k)-segmentation$_p$, the representative $\bar{t}_S$ can take a distinct value for each segment S. We would like to restrict the segment representatives to use at most h different values. Given a set of $L = \{l_1, \ldots, l_h\}$ of h values and a segmentation $\mathcal{M}$, the minimal error for segment $S \in \mathcal{M}$ is the l_i that minimizes the error measure, i.e.,

$$E_p^p(S|L) = \min_{l \in L} \sum_{t \in S} d(t, l)^p.$$

The minimizer of the error measure in the set L for a segment S is denoted by l_S. The problem of finding recurrent states in sequences is formalized as follows.

PROBLEM 2 ((k,h)-SEGMENTATION$_p$). *Given a sequence T, a k-segmentation $\mathcal{M}$ of T, a set L of h values, and $p = 1, 2$, the p-error of the segmentation $\mathcal{M}$ with respect to L is*

$$E_p(T, \mathcal{M}, L) = \left(\sum_{S \in \mathcal{M}} E_p^p(S|L) \right)^{\frac{1}{p}} = \left(\sum_{S \in \mathcal{M}} \sum_{t \in S} d(t, l_S)^p \right)^{\frac{1}{p}}$$

The task is to find the segmentation $\mathcal{M}$ and the set of values L that minimizes $E_p(T, \mathcal{M}, L)$. The minimum error achieved is denoted by $Seg_p(T, k, h)$, i.e.

$$Seg_p(T, k, h) = \min_{\mathcal{M} \in \text{Segm}_k(T),\ L:|L|=h} E_p(T, \mathcal{M}, L)$$

The (k,k)-segmentation$_p$ problem is a special case of (k,h)-segmentation$_p$, where the value of h is left unconstrained. This means the following.

OBSERVATION 1. *For a sequence T and any numbers k and h we have $Seg_p(T, k) \leq Seg_p(T, k, h)$.*

Also notice that for fixed h the value of $Seg_p(T, k, h)$ decreases as k increases. Since the largest value of k is n, we get the following

OBSERVATION 2. *For a sequence T of size n and any k and h we have $Seg_p(T, n, h) \leq Seg_p(T, k, h)$.*

The case of $k = n$, in Observation 2, can be viewed as ignoring the constraint that some consecutive points in the sequence should be represented by the same value. Thus, the (n,h)-segmentation$_p$ problem is the well-studied *clustering problem*. For $p = 1$ it is the Euclidean k-median problem, and for $p = 2$ it is k-means problem of finding the k points such that the sum of distances to the closest point is minimized.[1] Both of these problems can be solved in polynomial time for 1-dimensional data [13], and both are NP–hard for dimensions $d \geq 2$ [12].

LEMMA 1. *For $p = 1, 2$, and dimension $d \geq 2$, the problem (k,h)-segmentation$_p$ is NP–hard.*

PROOF. For $p = 1$ and $p = 2$, the k-median and k-means problems, respectively, are special cases of (k,h)-segmentation$_p$ with $k = n$. □

3. ALGORITHMS

In this section we describe three algorithms for solving the (k,h)-segmentation problem. The algorithms are based on solving the polynomial sub-cases and combining the results in different ways.

3.1 SEGMENTS2LEVELS algorithm

The first of our algorithms for the (k,h)-segmentation problem is called SEGMENTS2LEVELS. The algorithm is as follows. First solve the (k,k)-segmentation problem to obtain a k-segmentation $\mathcal{M}$. Then solve the (n,h)-segmentation

[1]Note that "k-means" is often used for one approximate algorithm for this problem.

problem to obtain a set L of h level values. Then assign each segment $S \in \mathcal{M}$ to the level in L which is closest to $\bar{t}_S$.

We show in Section 4 that this simple algorithm provides a 3-approximation to the (k,h)-segmentation problem for $p=1$ and for $p=2$, for dimension 1. For higher dimensions, the approximation guarantees are $3+\epsilon$ for $p=1$, and $\alpha+2$ for $p=2$, where α is the best approximation factor for the k-means problem.

The algorithm runs in time $O(n^2(k+h))$, as the running time of the dynamic programming method is quadratic in the number of points and linear in the number of segments.

3.2 ClusterSegments algorithm

The next algorithm, called ClusterSegments, yields approximation rations 5 and $\sqrt{5}$, for the 1-error and 2-error measures, respectively.

The first step of the algorithm is the same with the first step of Segments2Levels; it solves (k,k)-segmentation and decides about the segments of the solution. Then, each segment S in the optimal segmentation of (k,k)-segmentation is represented with the single value $\bar{t}_S$ and weight $|S|$. A set of h levels L is produced by clustering those k weighted points into h clusters. The final solution is produced by assigning each segment S to the level $l_S \in L$ which is the closest to $\bar{t}_S$ among all levels in L. The running time of the method is again $O(n^2(k+h))$.

3.3 Iterative algorithm

Our third algorithm is inspired by the EM algorithm and it is based on the following observations.

1. If we know the set L of levels that are to be used, then we can find the k optimal segments. This follows from simple application of dynamic programming. This step corresponds to the *E-step* of the EM algorithm.

2. Given an existing segmentation we can readjust the values of the levels and possibly improve the value of the solution. This is done as the *M-step* of the EM algorithm: for each level we obtain a possibly better value by computing the mean or the median (depending on the error that we are optimizing for) of all the points in the segments that are assigned to this level.

The Iterative algorithm uses these observations. It starts from an initial segmentation obtained by, e.g., either of the previous algorithms. Then, given the current levels, the optimal segmentation for those levels is found using the above observation. After that, a new set of h levels is computed by using the second observation. These two steps are iterated until the error does not decrease any more. We denote by Iterative(SL) the version that starts with the output of Segments2Levels, and by Iterative(CS) the version that starts with the output of ClusterSegments.

The Iterative algorithm produces at least as good approximations as the previous two methods. The running time of the algorithm is $O(In^2(k+h))$, where I is the number of iterations. In most of our experiments the initial solutions provided by Segments2Levels and ClusterSegments were very good, so the number of iterations remained below 3 or 4.

4. APPROXIMATION GUARANTEES

In this section we prove the approximation guarantees for the Segments2Levels and ClusterSegments algorithms. For the Iterative algorithm we only know that it never is worse that the other two methods. The empirical results in the next section show that in practice the algorithms work far better than what the worst-case bounds suggest.

4.1 Analysis of the Segments2Levels algorithm

Theorem 1. *Consider a 1-dimensional sequence T and integers k and h. Let $Opt_1 = Seg_1(T,k,h)$ be the value of the optimal segmentation for the 1-error measure. Denote by $SL_1(T,k,h)$ be the value of the solution provided by the algorithm* Segments2Levels. *Then $SL_1(T,k,h) \leq 3 \cdot Opt_1$.*

Proof. Recall from the description of the algorithm in Section 3.1, that there are two initial steps. The first step is to solve the (n,h)-segmentation$_1$ problem. Since we are in the 1-dimensional case, this can be solved in polynomial time using dynamic programming. Thus we obtain a set of levels L for which we have $Seg_1(T,n,h) \leq Opt_1$. The second step is to solve the (k,k)-segmentation$_1$, again by dynamic programming. So, we obtain a k-segmentation $\mathcal{M}$, for which $Seg_1(T,k) \leq Opt_1$.

Now consider a data point t of the sequence. Let S be the segment in $\mathcal{M}$ which point t belongs to, and let $l_S \in L$ be the level which segment S is assigned to by Segments2Levels. Also define $l_t \in L$ to be the value which point t is assigned to by the the solution of (n,h)-segmentation$_1$.

The error that point t contributes to the total error is $d(t,l_S)$ and by triangle inequality

$$d(t,l_S) \leq d(t,\bar{t}_S) + d(\bar{t}_S,l_S)$$

where, as before, $\bar{t}_S$ is the median of all points in segment S. However, since l_S is the closest to $\bar{t}_S$, among all levels in L, we have

$$d(\bar{t}_S,l_S) \leq d(\bar{t}_S,l_t)$$

and with a second application of the triangle inequality

$$d(\bar{t}_S,l_t) \leq d(\bar{t}_S,t) + d(t,l_t)$$

Combining the previous inequalities gives

$$d(t,l_S) \leq 2d(t,\bar{t}_S) + d(t,l_t)$$

and summing over all points in T we obtain the desired result:

$$\begin{aligned} &SL_1(T,k,h) \\ &\quad = \sum_{t\in T} d(t,l_S) \\ &\quad \leq 2 \cdot \sum_{t\in T} d(t,\bar{t}_S) + \sum_{t\in T} d(t,l_t) \\ &\quad \leq 2 \cdot Seg_1(T,k) + Seg_1(T,n,h) \leq 3 \cdot Opt_1. \end{aligned}$$

□

Theorem 2. *Let T be a 1-dimensional sequence and let k and h be integers. Let $Opt_2 = Seg_2(T,k,h)$ be the value of the optimal segmentation for (k,h)-segmentation$_2$ problem for the $p=2$ error measure. Let $SL_2(T,k,h)$ be the value of the solution provided by the algorithm* Segments2Levels. *Then $SL_2(T,k,h) \leq 3 \cdot Opt_2$.*

PROOF. The proof is very similar to the one for the 1-error measure. Again in the 1-dimensional case the problem (n,h)-segmentation$_2$ can be solved in polynomial time, so $Seg_2(T,n,h) \leq Opt_2$. Also by solving (k,k)-segmentation$_2$, we obtain segmentation $\mathcal{M}$ for which $Seg_2(T,k) \leq Opt_2$.

Using the same notation as in Theorem 1 we have that the error that a single point t contributes to the square of the total error is

$$d(t,l_S)^2 \leq (d(t,\bar{t}_S) + d(\bar{t}_S,l_S))^2$$

Combining with $d(\bar{t}_S,l_S) \leq d(\bar{t}_S,t) + d(t,l_t)$ we get

$$d(t,l_S)^2 \leq 4d(t,\bar{t}_S)^2 + d(t,l_t)^2 + 4d(t,\bar{t}_S)d(t,l_t)$$

and summing over all points in T we obtain

$$\begin{aligned}
&SL_2(T,k,h)^2 \\
&= \sum_{t\in T} d(t,l_S)^2 \\
&\leq 4\sum_{t\in T} d(t,\bar{t}_S)^2 + \sum_{t\in T} d(t,l_t)^2 + 4\sum_{t\in T} d(t,\bar{t}_S)d(t,l_t) \\
&\leq 4\sum_{t\in T} d(t,\bar{t}_S)^2 + \sum_{t\in T} d(t,l_t)^2 \\
&\quad +4\sqrt{\sum_{t\in T} d(t,\bar{t}_S)^2}\sqrt{\sum_{t\in T} d(t,l_t)^2} \quad \text{(by Cauchy-Schwarz)} \\
&\leq 4\cdot Seg_2(T,k)^2 + Seg_2(T,n,h)^2 \\
&\quad +4\cdot Seg_2(T,k)\cdot Seg_2(T,n,h) \\
&= 9\cdot Opt_2^2
\end{aligned}$$

which, by taking square roots, provides the claimed approximation ratio. □

4.1.1 Higher dimensions

The only place that the 1-dimensionality is used in Theorems 1 and 2 is in solving exactly the clustering problem (n,h)-segmentation$_p$, and claiming that $Seg_p(T,n,h) \leq Opt_p$. In higher dimensions, as we already mentioned, those clustering problems become NP–hard. However, we can approximate the solutions by polynomial algorithms. The k-median problem (i.e., (n,h)-segmentation$_1$) admits a PTAS [1], so we can find a set of levels L for which $Seg_1(T,n,h) \leq (1+\epsilon)Opt_1$. Plugging this in the proof of Theorem 1 we obtain an approximation ratio of $3+\epsilon$ for the 1-error measure. Similarly, k-means can be approximated in polynomial time. Calling α the best approximation ratio for k-means, and using that in the proof of Theorem 2 we obtain an approximation ratio of $\alpha+2$ for the 2-error measure. The current α is $9+\epsilon$ by Kanungo et al [8], yielding a factor of $11+\epsilon$ to our problem. The clustering algorithms (for (n,h)-segmentation) used to obtain these approximability results for higher dimensions might not be very efficient in practice.

4.2 Analysis of the ClusterSegments algorithm

For analyzing the performance of the ClusterSegments algorithm we need the following lemmas. The first is a weaker version of the triangle inequality in the case of the square of distances between points.

LEMMA 2 (DOUBLE TRIANGLE INEQUALITY). *For points x, y, and z we have*

$$d(x,y)^2 \leq 2\cdot d(x,z)^2 + 2\cdot d(z,y)^2$$

PROOF. The lemma can be shown with a simple application of the triangle inequality, i.e.,

$$\begin{aligned}
d(x,y)^2 &\leq (d(x,z)+d(z,y))^2 \\
&= d(x,z)^2 + d(z,y)^2 + 2\cdot d(x,z)d(z,y) \\
&\leq 2\cdot d(x,z)^2 + 2\cdot d(z,y)^2
\end{aligned}$$

since $2\cdot d(x,z)d(z,y) \leq d(x,z)^2 + d(z,y)^2$. □

The next lemma is a bias-variance result, which we state and prove for completeness.

LEMMA 3. *Let $\{t_1,\ldots,t_m\}$ be a set of m points and let $\bar{t}$ be their mean, i.e. the minimizer of the 2-error measure. For any point x it is true that*

$$m\cdot d(\bar{t},x)^2 \leq \sum_{i=1}^{m} d(t_i,x)^2 = m\cdot d(\bar{t},x)^2 + \sum_{i=1}^{m} d(t_i,\bar{t})^2$$

PROOF. We will only show the equality part of the lemma, since this implies the inequality, as well. For simplicity we will assume that the points are one-dimensional, i.e. $d(x,y) = |x-y|$. The extension to the multidimensional case can be done by applying the equality in each coordinate separately and summing in all coordinates. By using the definition of the mean $\sum_{i=1}^{m} t_i = m\bar{t}$, we have

$$\begin{aligned}
&m(\bar{t}-x)^2 + \sum_{i=1}^{m}(t_i-\bar{t})^2 \\
&= m\bar{t}^2 + mx^2 - 2mx\bar{t} + \sum_{i=1}^{m} t_i^2 + m\bar{t}^2 - 2m\bar{t}\sum_{i=1}^{m} t_i \\
&= mx^2 - 2mx\bar{t} + \sum_{i=1}^{m} t_i^2 \\
&= \sum_{i=1}^{m}(t_i-x)^2
\end{aligned}$$

□

Now we show that the solution found by ClusterSegments is within a factor of $\sqrt{5}$ of the optimal solution for the 2-error measure. Using a similar line of arguments, one can show that for the 1-error the approximation ratio of ClusterSegments is 5. Since we have already shown a ratio of 3 for that case using the Segments2Levels algorithm, we omit the analysis.

THEOREM 3. *For a sequence T, numbers k and h, and the 2-error measure, let $Opt_2 = Seg_2(T,k,h)$ be the value of the optimal segmentation for (k,h)-segmentation$_2$. If $CS_2(T,k,h)$ is the value of the solution provided by the algorithm* ClusterSegments*, then $CS_2(T,k,h) \leq \sqrt{5}\cdot Opt_2$.*

PROOF. By solving (k,k)-segmentation$_2$ we obtain a segmentation $\mathcal{M}$ for which $Seg_2(T,k) \leq Opt_2$. Let L be the set of h points found by clustering the k weighted points $\bar{t}_S$, for $S \in \mathcal{M}$. Let l_S be the cluster center point which segment S is assigned to.

Now, the error contributed by a specific segment S to $CS_2(T,k,h)^2$ is $\sum_{t\in S} d(t,l_S)^2$. Summing over all segments,

and applying Lemma 3 we obtain

$$\begin{aligned}
&CS_2(T,k,h)^2 \\
&\quad= \sum_{S\in\mathcal{M}} \sum_{t\in S} d(t,l_S)^2 \\
&\quad= \sum_{S\in\mathcal{M}} \left(|S|\cdot d(\bar{t}_S,l_S)^2 + \sum_{t\in S} d(t,\bar{t}_S)^2 \right) \\
&\quad= \sum_{S\in\mathcal{M}} |S|\cdot d(\bar{t}_S,l_S)^2 + Seg_2(T,k)^2 \\
&\quad\le \sum_{S\in\mathcal{M}} |S|\cdot d(\bar{t}_S,l_S)^2 + Opt_2^2
\end{aligned}$$

The next step is to bound the term $\sum_{S\in\mathcal{M}} |S|\cdot d(\bar{t}_S,l_S)^2$. Consider the set of optimal level points M for the original problem. For each segment S, let μ_S be the point in M which is the closest to $\bar{t}_S$. For the 1-dimensional case, the clustering of the k weighted points can be done optimally, by dynamic programming. So by claiming optimality for L we get

$$\sum_{S\in\mathcal{M}} |S|\cdot d(\bar{t}_S,l_S)^2 \le \sum_{S\in\mathcal{M}} |S|\cdot d(\bar{t}_S,\mu_S)^2$$

Assume that the optimal segmentation partitions a segment S of $\mathcal{M}$ into r subsegments S_j, each of them having size s_j and mean $\bar{t}_j$. Also assume that the optimal solution assigns each of those segments at a level $\mu_j \in M$, for $j = 1,\ldots,r$. Since μ_S is the closest to $\bar{t}_S$, we have

$$\begin{aligned}
&|S|\cdot d(\bar{t}_S,\mu_S)^2 \\
&\quad\le \sum_{j=1}^{r} s_j\cdot d(\bar{t}_S,\mu_j)^2 \\
&\quad\le 2\left(\sum_{j=1}^{r} s_j\cdot d(\bar{t}_S,\bar{t}_j)^2 + \sum_{j=1}^{r} s_j\cdot d(\bar{t}_j,\mu_j)^2\right) \text{(by Lemma 2)} \\
&\quad\le 2\sum_{j=1}^{r}\sum_{t\in S_j} d(t,\bar{t}_S)^2 + 2\sum_{j=1}^{r}\sum_{t\in S_j} d(t,\mu_j)^2 \text{(by Lemma 3)} \\
&\quad= 2\sum_{t\in S} d(t,\bar{t}_S)^2 + 2\sum_{t\in S} d(t,\mu_t)^2
\end{aligned}$$

where, by μ_t we denote the level to which the point t is assigned by the optimal solution. Summing over all segments in $\mathcal{M}$, it is clear that the term $\sum_{S\in\mathcal{M}} |S|\cdot d(\bar{t}_S,\mu_S)^2$ is bounded by $4\cdot Opt_2^2$. Combining the above inequalities we have $CS_2(T,k,h)^2 \le 5\cdot Opt_2^2$, which yields the desired bound. □

Similarly with the SEGMENTS2LEVELS algorithm, the proof can be adapted for higher dimensions giving ratios of $5+\epsilon$ for $p=1$, and $\sqrt{4\alpha^2+1}$ for $p=2$. As before, α is the best approximation factor for the k-means problem.

Two points should be discussed here regarding the extension of the algorithms in higher dimensions. First, one should note that CLUSTERSEGMENTS achieves a better approximation ratio in the one dimension where $\alpha = 1$, while SEGMENTS2LEVELS is better in higher dimensions, and in general as long as $\alpha \ge 1.86$.

The second observation is that the clustering subroutine used by the CLUSTERSEGMENTS algorithm takes as input only k points, instead of n. This means that for small values of k, say small constants, one could possibly afford to use a brute force algorithm in order to find the optimal clustering. This would require time exponential in k (still constant however), but the approximation ratio remains $\sqrt{5}$ independently of the dimension.

5. EXPERIMENTAL EVALUATION

5.1 Experiments on synthetic data

To test the performance of the methods we generated artificial data sets that display the recurrent state behavior that is of interest to us.

The data sets were generated as follows. First we created a Markov model with 5 states with transition probabilities between the states chosen randomly from in the interval $[0, \frac{1}{500}]$, implying that the average length of stay in a given state is about 250 steps. For the 0-1 data, each state i had a probability p_i of emitting a 1. We simulated the state transitions and emissions for n=5K, 10K, 15K and 20K steps.

Given each point in the sequence, we know what the generating probability p of a 1 has been at that point. When applying the algorithms for (k,h)-segmentation described in the previous sections, we obtain an estimate of the generated probabilities[2]. We then computed the average error between the true value of p and the estimate produced by the algorithms.

The other artificial data set was generated by having a real-valued hidden source x at each state. When the generating process was at that state, the generated point was chosen randomly from the normal distribution with mean x and variance σ^2, with σ varying from 0.02 to 0.4.

The results for these two sets of artificial data are shown in Figures 1 and 2, respectively. One can see that the error is typically very low, below 4% for the best algorithms, CLUSTERSEGMENTS and ITERATIVE(CS). The SEGMENTS2LEVELS and the ITERATIVE(SL) algorithms perform worse. This is only to be expected, as for 0-1 data the clustering algorithm provides only two sources. The accuracy of CLUSTERSEGMENTS and ITERATIVE(CS) solution improves nicely when the size of the sequence grows. For normally distributed data, the error remains very small even for large variance. The number of states used does not have to be exactly the correct one, as seen in Figure 3. Note that one could also reconstruct the transition probabilities for sequences that are long enough. We obtained similar results for the case of 10 underlying states.

5.2 Experiments on genome data

We used the *sliding window* technique to collect statistics from the human genome sequence, in particular from chromosome 22. Our window size was 500 kb, and the sliding step was set to 50 kb. The known area of the chromosome is about 35 Mb, so our windowing parameters yield sequences of about 700 points. Within each window we compute the frequency of each of the w-letter words over A, C, G, T, for $w = 2$ and 3. Therefore, we obtain sequences of dimensions 16 and 64. The same windowing technique was used on data indicating the positions of SNPs and genes on chromosome 22. For each window we computed the frequency of SNPs,

[2]Note that given a segment $S = (t_1,\ldots,t_m)$ containing m_1 1s and $m_0 = m - m_1$ 0s, the p that maximizes the Bernoulli likelihood $\prod_i p^{t_i}(1-p)^{1-t_i}$ is m_1/m, i.e., the p that minimizes the L_2 distance.

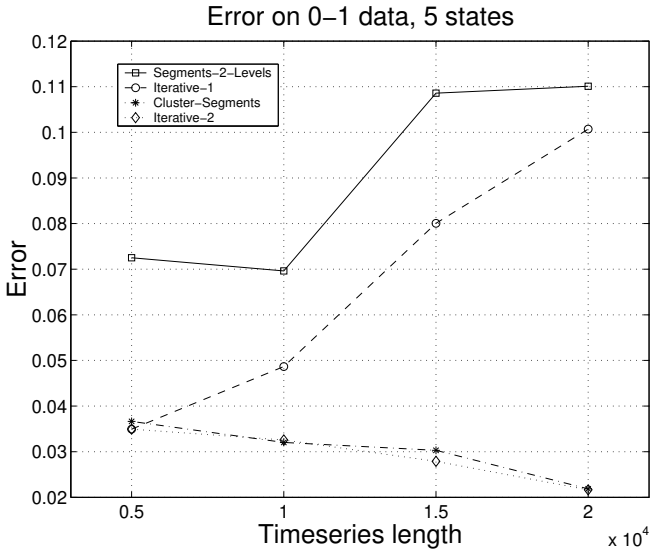

Figure 1: Average error of the estimated level from the generating level for artificial 0-1 data as a function of the length of the input sequence.

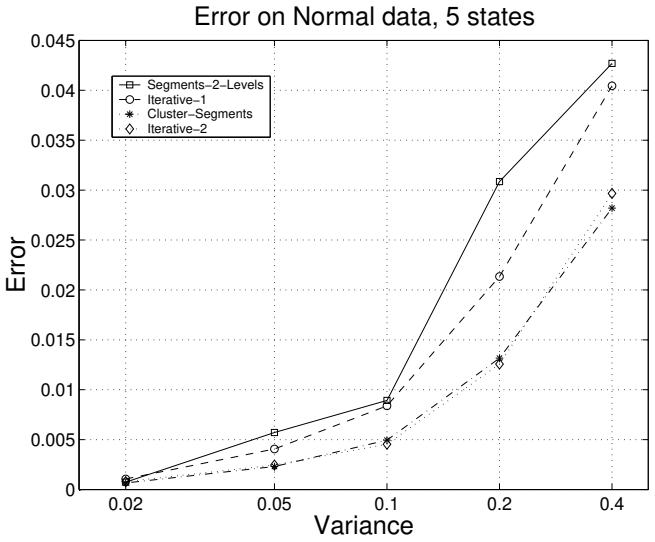

Figure 2: Average error of the estimated level from the generating level for normally distributed data as a function of the variance in the generated data.

the frequency of genes, and the combined frequency (2-d points).

For the sequences we described above, we used our method to search for segmentations with recurrent states. We applied the Bayesian information criterion (see [17, 10]) to search for the values of k and h yielding the best segmentation. Our results are shown in Figure 4. The results indicate that the (k, h)-segmentation methods produce intuitively appealing results; the BIC-optimal number of segments is reasonably small and the number of sources is clearly smaller. The results can be viewed as showing that, with respect to the statistics used, there are segments in the chromosome that show the same characteristics. The statistics used for segmenting the sequences could, of course, be chosen differently; the presented results are just examples. We are, however, exploring the relationship of the obtained segments with, e.g., the human-mouse synteny maps.

6. CONCLUSIONS

We have introduced the (k, h)-segmentation problem, where the task is to find a segmentation of a sequence into k segments stemming from h different sources. The problem is NP-complete in the general case. We gave simple approximation algorithms for the problem, proved bounds on the approximation quality, and gave empirical results on real and artificial data.

Several open problems remain. On the theoretical side the foremost is finding out whether the (k, h)-segmentation problem is NP-hard for 1-dimensional data. The analysis of the approximation qualities seem to be fairly tight, but it would be interesting to know whether the results can be improved.

In this paper we gave only simple examples of the applications of the method. For genomic analyses the main task is to select good characteristics over which the segmentation problem is then solved. The ones we used, gene and SNP density and distribution of short words, are just examples: basically any characteristic of small blocks of sequences can be used. An interesting issue is measuring the correlation between two segmentations.

7. REFERENCES

[1] S. Arora, P. Raghavan, and S. Rao. Approximation schemes for euclidean k -medians and related problems. In *ACM Symposium on Theory of Computing*, pages 106 – 113, 1998.

[2] R. K. Azad, J. S. Rao, W. Li, and R. Ramaswamy. Simplifying the mosaic description of DNA sequences. *Physical Review E, 66, article 031913*, 2002.

[3] R. Bellman. On the approximation of curves by line segments using dynamic programming. *Commun. ACM*, 4(6), 1961.

[4] K. Bennett. Determination of the number of zones in a biostratigraphical sequence. *New Phytol.*, 132:155–170, 1996.

[5] A. Cantoni. Optimal curve fitting with piecewise linear functions. *IEEE Transactions on Computers*, C-20(1):59–67, 1971.

[6] G. A. Churchill. Stochastic models for heterogeneous DNA sequences. *Bulletin of Mathematical Biology*, 51:79 – 94, 1989.

[7] J. Himberg, K. Korpiaho, H. Mannila, J. Tikanmäki, and H. T. Toivonen. Time series segmentation for context recognition in mobile devices. In *The 2001 IEEE International Conference on Data Mining (ICDM'01)*, pages 203 – 210, 2001.

[8] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R.Silverman, and A. Wu. A local search approximation algorithm for k-means clustering. In *Proceedings of the 18th Annual Symposium on Computational Geometry*, pages 10 – 18, 2002.

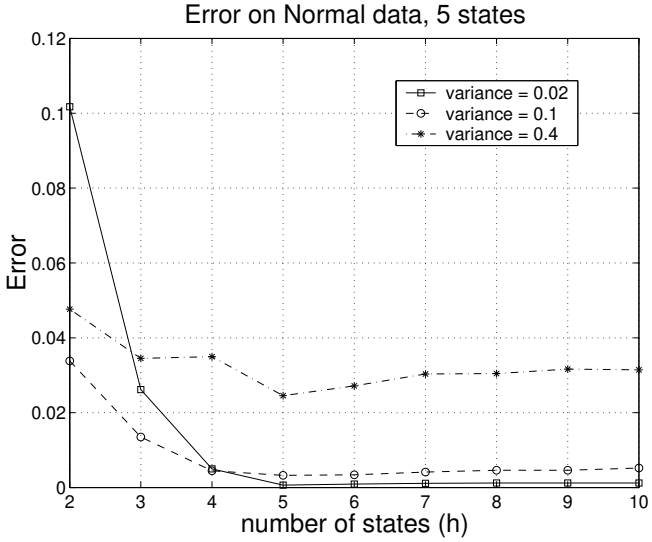

Figure 3: Average error of the estimated level from the generating level for normally distributed data as a function of the number h of states used by the algorithm.

[9] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. In *IEEE International Conference on Data Mining*, pages 289 – 296, 2001.

[10] W. Li. DNA segmentation as a model selection process. In *RECOMB 2001*, pages 204 – 210, 2001.

[11] J. Liu and C. Lawrence. Bayesian inference on biopolymer models. *Bioinformatics*, 15(1):38–52, 1999.

[12] N. Megiddo and K. J. Supowit. On the complexity of some common geometric location problems. *SIAM Journal on Computing*, 13(1):182–196, 1984.

[13] N. Megiddo, E. Zemel, and S. L. Hakimi. The maximum coverage location problem. *SIAM Journal on Algebraic and Discrete Methods*, 4:253–261, 1983.

[14] A. Pavlicek, J. Paces, O. Clay, and G. Bernardi. A compact view of isochores in the draft human genome sequence. *FEBS Letters*, 511:165–169, 2002.

[15] V. Ramensky, V. Makeev, M. Roytberg, and V. Tumanyan. DNA segmentation through the bayesian approach. *Journal of Computational Biology*, 7(1/2):215–231, 2000.

[16] M. Salmenkivi, J. Kere, and H. Mannila. Genome segmentation using piecewise constant intensity models and reversible jump MCMC. In *European Conference on Computational Biology (ECCB2003)*, 2003. To appear.

[17] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 7(2):461 – 464, 1978.

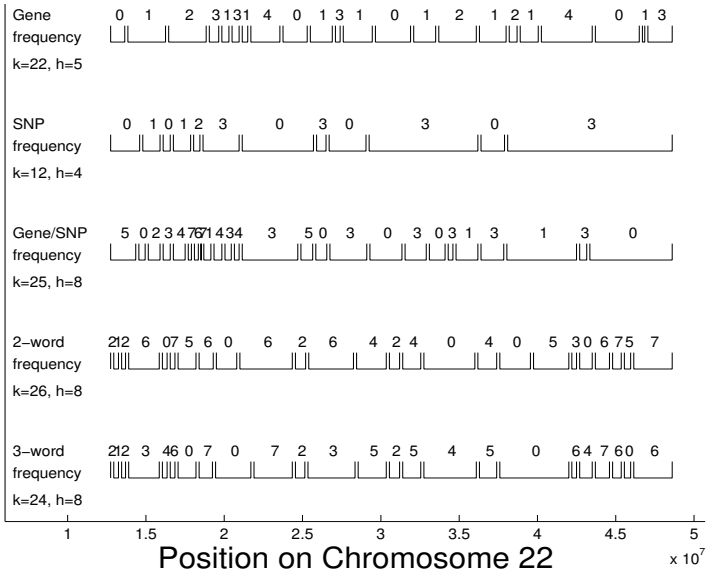

Figure 4: BIC-optimal (k, h)-segmentations for sequences of frequencies of genes, SNPs, combined genes and SNPs, 2-words, and 3-words. Each sequence has about 700 points; the dimensions are 1, 1, 2, 16, and 64, respectively. The numbers labeling each segment indicate the recurrent states.