The Relation between Discrete Convolution/Correlation and String Matching, and Exploring the Possibility of a Deterministic Linear-Time Algorithm for Discrete Convolution/Correlation

Sumantra Dutta Roy
Department of Electrical Engineering
Indian Institute of Technology Delhi, INDIA
sumantra@cse.iitd.ac.in

Poonam Suryanarayan
Department of Electronics and Communication Engineering,
NIT Karnataka, Surathkal - 575 025, INDIA.
poonam.suryanarayan@gmail.com

Abstract

This paper explores the relation between discrete convolution/correlation and string matching for different specific classes of input signals. We examine the time complexity of computation for both convolution/correlation and string matching, and explore the possibility of a deterministic algorithm for discrete convolution/correlation.

Indexing Terms:

Fast Discrete Convolution/Correlation, Linear-Time, FFT, String Matching, KMP Linear-Time String Matching Algorithm, Floating Point and Boolean Convolution/Correlation.

1. Introduction

Discrete convolution/correlation are fundamental signal processing algorithms in the context of Linear Time-Invariant systems. In computer science, string matching is fundamental to many text processing operations. In this paper, we explore the relation between discrete convolution/correlation, and string matching. We consider the time complexity of different classes of algorithms for both domains.

The classical naive string matching algorithm has quadratic time complexity $\mathcal{O}(NM)$ (for strings of lengths M and N, explained in Section 2.2). The 1970s saw many algorithms devised for fast string matching. (Aho, Hopcroft and Ullman [1] and Cormen et al. [2] give an overview of the literature.) Karp and Rabin [3], and Boyer and Moore [4] give fast algorithms for string matching - however, both have a worst case time complexity that is quadratic: $\mathcal{O}(NM)$. A finite state machine-based pattern matcher (also known as a DFA-based pattern matcher) has time complexity $\mathcal{O}(N + M^3|\Sigma|)$, where Σ is the alphabet from which the string characters are drawn, and $|\Sigma|$ denotes the cardinality of this set i.e., the size of the alphabet. Using the structure of the pattern of length M to be found in a piece of text of length N - looking for points of repetition, and efficiently searching for skip factor to start the next matching point - give rise to linear-time string matching algorithms. Prominent among those are algorithms due to Knuth, Morris and Pratt [5] (commonly referred to as the 'KMP Algorithm'), and Galil and Seifras [6].

The asymptotic time complexity of convolution/correlation in the way they are defined, is $\mathcal{O}(NM)$ (explained in Section 2.1). A standard way of speeding up the above is to go through the frequency domain using the Fast Fourier Transform (FFT), which reduces the time complexity of $\mathcal{O}(N \log N)$. *However, a generic algorithm which reduces this to linear time has eluded researchers*, who have primarily concentrated on devising fast algorithms either for special cases of convolution/correlation, or do so at the cost of accuracy in the final result.

Fisher and Paterson [7] explore the relation between generalised string matching problems (such as

string matching with don't cares) and integer multiplication, and other products. There has been some work on generalised string matching problems such as two-dimensional approximate matching and tree pattern matching in addition to string matching with don't cares - including developing randomised algorithms for the same, e.g., Cole and Hariharan [8], Indyk [9], Muthuswamy and Palem [10]. Most of these works are based on the use of convolution for speeding up these generalised string matching problems. Simard et al. [11] propose a speed-up for convolution using an approximation for regions of the signal with low degree polynomials. The paper examines convolutions for feature extraction, where in cases, one can sacrifice precision for speed. Breitzman [12] considers convolution algorithms based on both the Chinese Remainder Theorem and the FFT, and examines the reduction in the number of actual computations in different cases through situation-specific procedures. He does not consider any general purpose reduction in the asymptotic time complexity of the number of operations. The same is the case with the work of Werman [13].

In this paper, we examine a possible relation between the two problems of discrete correlation/convolution and string/pattern matching, and explore the possibility of using ideas from the domain of string matching (which can be done in linear time), for performing fast convolution/correlation. We first examine the computational complexity issues for both the domains in Section 2: convolution and correlation in Section 2.1, and string matching in Section 2.2. We examine the general case of floating point convolution/correlation in Section 3. Section 4 examines the problem of Boolean convolution/correlation. Section 5 dwells on the relation between convolution/correlation and string matching. We show that it is not possible to perform convolution/correlation using a linear-time algorithm. Section 6 concludes the paper.

2. Basic Complexity Issues in Convolution/Correlation, and String Matching

In this section, we examine the basic definitions of the signal processing operations of convolution/correlation, and string matching, and examine time complexity issues.

2.1. Convolution and Correlation

Convolution is an operation fundamental to Linear Time-Invariant (LTI) systems (also known as Linear Shift-Invariant systems for two- and higher dimensional signals) since it allows a system to be represented in terms of its impulse response - a signal. The response of a system to a particular signal is its convolution with the impulse response of the system [14]. Convolution and Correlation are defined as infinite sums. The *aperiodic* convolution for two discrete signals $x[n]$ and $h[n]$ is defined as

$$y[n] = \sum_{k:-\infty}^{+\infty} x[k]h[n-k] \tag{1}$$

For practical time-limited signals, one often considers their *periodic* versions, so as to apply the DFT/FFT to them for equivalent analysis in the time and frequency domains. The DFT and its inverse are both periodic [15]. Given two discrete signals: an N-point signal $T[n]$ and an M-point signal $P[n]$, we define the convolution of $T[n]$ and $P[n]$ as:

$$D[k] = \sum_{k=0}^{M+N-2} T[k]P[n-k] \tag{2}$$

A related operation is Correlation:

$$C[k] = \sum_{k=0}^{M+N-2} T[k]P[n+k] \tag{3}$$

(These $M+N-1$ point signals are taken to be periodic with time period $M+N-1$ - this is sometimes referred to as Brigham's result [15].) The computation of both convolution and correlation require a signal to be shifted all around another; and a pair-wise multiplication of the overlapping scaled Kronecker delta functions (discrete impulses), and an addition. The main difference between the two is that for convolution, one of the signals is reflected - flipped around the y-axis, and this is shifted all around the other signal. *The physical significance of correlation is that the resultant value is high for the parts of the larger signal which resemble the smaller signal. (The same is true for convolution - only that we deal with a reflected version of one signal.)* We can abstract out the following two operations involved in convolution/correlation:

1. *A pair-wise operation* between corresponding points of two signals: multiplication, and
2. *An assimilation operation* for each result point: addition.

Computation-wise, the operation count in both cases is the same. Using the Big-Oh notation [2], the operation count is $\mathcal{O}(MN)$ in both cases. In the original domain (the time domain, for instance), the operation count in terms of the number of multiplications

(and additions, too) is quadratic as mentioned above: for each of the $M + N - 1$ terms of the convolution/correlation (Brigham's result [15]), the number of operations needed is N. One way to reduce the operation count is to operate in the frequency domain instead, via the Fast Fourier Transform (FFT). Without loss of generality, we may assume $N > M$. We can zero-pad the smaller signal to get both signals of length N. We convert both signals to the frequency domain (in time $\mathcal{O}(N \log N)$ using the FFT), perform N pairwise multiplications, and reconvert the signals back to the original domain (again in $\mathcal{O}(N \log N)$ time). The overall time complexity is $\mathcal{O}(N \log N)$. In this paper, we examine the feasibility of devising a linear time algorithm for correlation and convolution, since similar string operations are possible in linear time. We examine related string matching operations in the next section.

2.2. String Matching

From this section and the previous one, we discern a commonality in the two operations of convolution and correlation on the one hand, and string matching, on the other. Both involve shifting an array around another, and looking for regions of commonality between the smaller array, and regions of the larger array.

Naive string matching involves such an operation, which results in $(N - M + 1)M$ operations, since one can slide the pattern $P[1 \ldots M]$ and look at M possible pairwise matches at all $N-M+1$ positions of the overlap with text $T[1 \ldots N]$. For each of these $N - M - 1$ positions of possible matches, if the M next consecutive characters in $T[\,]$ are the same as those in $P[\,]$, we declare a match at that position. Fig. 1 summarises the steps in the naive string matching algorithm.

This comes closest to convolution/correlation - since we can remove the `break` statement in the inner loop in Fig. 1, and compute the convolution/correlation piecewise products for all the M overlapping time sampling points and sum them up to get the convolution/correlation sum at point i. We can declare a match if the convolution/correlation sum is greater than a predetermined threshold, and not, otherwise. (There are $2M - 2$ extra points of a possible partial match, where the convolution/correlation sum is computed - Sec. 3 has the details. However, this does not affect the overall argument being presented in this section.)

A lacuna associated with naive string matching is not moving the pattern ahead for a mismatch at a particular position - moving it ahead by not 1 position, but a larger amount. This is possible because we know the structure of the pattern $P[1 \ldots M]$ to be matched - this precludes some shift positions. Researchers have used this observation to devise linear-time algorithms for string matching. Prominent among these are the Knuth, Morris and Pratt algorithm [5] (also called, and referred to herewith as the 'KMP algorithm'), and that by Galil and Seifras [6]. We choose the former as a representative example of the class of linear-time string matching algorithms. The KMP algorithm performs string matching in linear time - $\mathcal{O}(N + M)$ [5]. This involves using a prefix function: to compute the largest shift from a current position of mismatch. This is the longest prefix of pattern $P[1 \ldots M]$ that is also a suffix of the match seen thus far. Fig. 2 summarises the steps in the algorithm [2]. The amortised time complexity [2] for `compute_prefix_function` is $\Theta(M)$, and that for `KMP_matcher` is $\Theta(N)$ [5]. (Θ denotes asymptotic tight bound: a stronger condition as compared to an asymptotic upper bound $\mathcal{O}$, though the latter suffices for our case.) The overall time complexity is thus, $\Theta(N + M)$.

```
ALGORITHM naive_matcher(T, P)
FOR i ← 1 TO N − M + 1 DO
  q ← 0; /* no of chars matched */
  FOR j ← 1 TO M DO
    /* check match */
    IF P[j] = T[i + j − 1] THEN
      q ← q + 1 ELSE break;
    IF q = M THEN
    print ''match at shift'' i;
```

Figure 1. Algorithm `naive_matcher`: adapted from [2]. This is the closest to convolution/correlation computation in terms of shifting a signal around the other (the smaller, for convenience), and looking for regions of commonality. (Details in text: Sec. 2.2.)

3. Floating Point Convolution/Correlation

Section 2.2 explores the similarity between string matching and convolution/correlation. Floating point convolution/correlation is different from string matching in the following ways:

1. String Matching is defined over an alphabet Σ, a *finite* set of symbols. Convolution/correlation between two real-valued signals involves floating point pairwise multiplications, and floating point additions as the assimilation operation. The set of real numbers $\mathcal{R}$ is *uncountably infinite*.

```
ALGORITHM KMP_matcher(T, P)
π ← compute_prefix_function(P);
q ← 0; /* no of chars matched */
FOR i ← 1 TO N DO /* scan L to R */
  WHILE q > 0 AND P[q+1] ≠ T[i] DO
    q ← π[q]; /* next doesn't match */
  IF P[q+1] = T[i] THEN
    q ← q + 1; /* next matches */
  IF q = M THEN
    print ''match at shift'' i - M;
    q ← π[q]; /* look for next match */
```

```
ALGORITHM compute_prefix_function(P)
π[1] ← 0; k ← 0;
FOR q ← 2 TO M DO
  WHILE k > 0 AND P[k + 1] ≠ P[q] DO
    k ← π[k];
  IF P[k + 1] = P[q] THEN
    k ← k + 1;
  π[q] ← k;
RETURN π;
```

Figure 2. Algorithm `KMP_matcher` for linear-time string matching due to Knuth, Morris and Pratt [5] (commonly referred to as the 'KMP algorithm'): adapted from [2]

2. The output of string matching is a Boolean string, of length $N - M + 1$, whereas the output of a convolution/correlation operation is a floating point array of size $N + M - 1$: there are $2M - 2$ extra calculations involved in convolution/correlation. If we could devise a linear-time algorithm for convolution/correlation, this will not affect the overall time complexity, though - these extra calculations would get absorbed into the overall time complexity.

3. The length of the Boolean string is $N - M + 1$, since the string matching problem is not defined for cases when the length of the overlap is less than M characters: there can be an instance of pattern $P[1 \ldots M]$ in text $T[1 \ldots N]$ if and only if the portion of the text window considered for matching has at least M characters. In convolution/correlation on the other hand, the minimum overlap needed is one signal point, since the product is defined for every case of overlap - whether it is one signal point, or all M.

4. In string matching, a check for an occurrence of the pattern $P[1 \ldots M]$ at a particular position i in the text $T[1 \ldots N]$ terminates the moment there is a mismatch - one need not match the rest of the characters out of the M in the pattern $P[1 \ldots M]$. In fact as mentioned in Section 2.2, algorithm KMP gets its speedup by skipping the next few characters to get the pattern template to the next position i' of a possible match - by removing all preceding positions of a guaranteed mismatch. In convolution/correlation on the other hand, the output is a floating point value at each position of the overlap (of length between 1 and M). Convolution/correlation requires that this be calculated for each of the possible $N + M - 1$ points of overlap.

In general, it is not possible to abort the computation of the floating point convolution/correlation based on a single value of the pairwise operation - a low value of a pairwise product at a particular time sampling point does not imply that the overall convolution sum (to be computed after this by taking into account other sampling points to the right) will be low - smaller than a pre-determined threshold. A current low match is no indicator of an overall good match. This is what prevents the formulation of a KMP-like linear-time algorithm for the same.

A string matching procedure has the point-wise operation as a similarity test - a match, or no match: a Boolean operation. In the next section, we explore whether we can adopt a KMP-like algorithm for a special case of convolution/correlation namely, Boolean convolution/correlation.

4. Boolean Convolution/Correlation

Boolean convolution and correlation are defined for Boolean (0/1) strings as the case when

- The pair-wise operation is an AND ($\cdot$), and
- The assimilation operation is an OR ($+$)

The Boolean case also does not admit to a KMP-like algorithm, since the pair-wise operation (AND) does not measure the similarity between two bits: $0 \cdot 1 = 1 \cdot 0 = 0$ and $1 \cdot 1 = 1$, but $0 \cdot 0 = 0$. Further, the assimilation operation does not stop at any intermediate point since the result of the convolution/correlation could be 1 if a future AND results in a 1. In Boolean algebra, 1 + 1 = 1, but 0 + x = x.

5. The Relation between Convolution/Correlation and String Matching

Based on the discussion in the previous sections, we can conclude the following: *String Matching is a spe-*

cial case of Boolean convolution/correlation where

- *The pair-wise operation is an X-NOR, and*
- *The assimilation operation is an AND.*

We have looked at the possibility of developing a KMP-like algorithm for a physically significant case of convolution/correlation - which gets to the *lower bound* of the process, since string matching can be looked upon as a special case of convolution/correlation. Thus, it is not possible to perform convolution/correlation in linear time, except in special cases where the length of the pattern $P[n]$ is a constant: $\mathcal{O}(1)$, and the time complexity reduces to $\mathcal{O}(N)$.

6. Conclusions

In this paper, we examine the relation between discrete convolution/correlation and string matching. We explore the use of a linear time string matching algorithm to speed up convolution/correlation. We show the relation between the two, and show that the physically significant cases of floating point and Boolean convolution/correlation cannot be performed in linear time.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1983.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, Prentice-Hall of India Private Limited, 2002.

[3] R. M. Karp and M. O. Rabin, "String-Matching and Other Products," Tech. Rep. TR-31-81, Aiken Computation Laboratory, Harvard University, 1981.

[4] R. S. Boyer and J. S. Moore, "A Fast String-Searching Algorithm," *Communications of the ACM*, vol. 21, no. 2, pp. 762 – 772, 1977.

[5] D. E. Knuth, J. H. Morris, Jr., and V. R. Pratt, "Fast Pattern Matching in Strings," *SIAM Journal on Computing*, vol. 6, no. 2, pp. 323 – 350, 1977.

[6] Z. Galil and J. Seifras, "Time-Space-Optimal String Matching," *Journal of Computer and System Sciences*, vol. 26, no. 3, pp. 280 – 294, 1983.

[7] M. J. Fischer and M. S. Paterson, "String-Matching and Other Products," Tech. Rep. MAC TM-41, Massachusetts Institute of Technology, January 1974.

[8] R. Cole and R. Hariharan, "Tree Pattern Matching to Subset Matching in Linear Time," *SIAM Journal on Computing*, vol. 32, no. 4, pp. 1056 – 1066, 2003.

[9] P. Indyk, "Faster Algorithms for String Matching Problems: Matching the Convolution Bound," in *Proc. 39th Symposium on Foundations on Computer Science*, 1998.

[10] S. Muthukrishnan and K. Palem, "Non-Standard Stringology: Algorithms and Complexity," in *Proc. Symposium on Theory of Computing*, 1994, pp. 770 – 779.

[11] P. Y. Simard, L. Bottou, P. Haffner, and Y. LeCun, "Boxlets: a Fast Convolution Algorithm for Signal Processing and Neural Networks," in *Advances in Neural Information Processing Systems*, M. Kearns, S. Solla, and D. Cohn, Eds., vol. 11, pp. 571 –577. MIT Press, 1999.

[12] A. F. Breitzman, *Automatic Derivation and Implementation of Fast Convolution Algorithms*, Ph.D. thesis, Drexel University, January 2003.

[13] M. Werman, "Fast Convolution," in *Proc. Int'l Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision*, 2003, vol. 11, pp. 528 –529.

[14] A. V. Oppenheim, A. S. Willsky, and S. Hamid Nawab, *Signals and Systems*, Prentice-Hall of India Private Limited, Second edition, 1998.

[15] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Addison Wesley Longman, Inc., 1992.