MISSOURI
S&T

Missouri University of Science and Technology

Scholars' Mine

Electrical and Computer Engineering Faculty Research & Creative Works

Electrical and Computer Engineering

01 Jan 2001

A Statistical Solution to a Text Decoding Challenge Problem

Xindi Cai

Rui Xu
Missouri University of Science and Technology

Donald C. Wunsch
Missouri University of Science and Technology, dwunsch@mst.edu

V. A. Samaranayake
Missouri University of Science and Technology, vsam@mst.edu

Follow this and additional works at: https://scholarsmine.mst.edu/ele_comeng_facwork

Part of the Electrical and Computer Engineering Commons

A statistical solution to a text decoding challenge problem

Xindi Cai*, Rui Xu*, V. A. Samaranayake** and Donald C. Wunsch II*

*Applied Computational Intelligence Laboratory
Dept. of Electrical and Computer Engineering
University of Missouri - Rolla
MO 65409-0249 USA

**Dept. of Mathematics and Statistics
202 Rolla Building
University of Missouri - Rolla
MO 65409-0020 USA

cai@umr.edu rxu@umr.edu vsam@umr.edu dwunsch@ece.umr.edu

Abstract:

Given an encoded unknown text message in the form of a three dimensional spatial series generated by the use of four smooth nonlinear functions, we use a novel method based on simple statistical reasoning to pick up samples for rebuilding the four functions. The estimated functions are then used to decode the sequence. The experimental results show that our method gives a nearly perfect decoding, enabling us to submit a 100% accurate solution to the IJCNN challenge problem.

1. Introduction

We are given a three-dimensional spatial series containing information about the encoding of an unknown text message. The encoding and the creation of the time series was carried out as follows:

1) An unknown English text was converted into a binary sequence according to the 8-bit ASCII table, with the character *SPACE* (i.e. 00100000), replaced by the character *NULL* (i.e. 00000000). Let $\{x(k)\}_{k=1}^{N}$ represent this binary sequence, where $x(k) = 0$ or 1, for $k = 1,2,...,N$. Note that k represents the bit number and N denotes the total number of bits in the sequence. We define $x(0) = 0$.

2) At each $k = 1,2,3,....,N$ a function $f_{x(k-1),x(k)} : [-1,1] \times [-1,1] \to \mathbb{R}$ was selected from among four pre-determined, smooth, nonlinear functions $f_{0,0}, f_{0,1}, f_{1,0}$, and $f_{1,1}$. The value y_k was then determined by the mapping

$$y_k = f_{x(k-1),x(k)}(u_1(k), u_2(k)), \qquad (1)$$

where the arguments $u_1(k)$ and $u_2(k)$ are independent random variables drawn from a Uniform (-1, 1) distribution.

3) The resulting spatial series $(u_1(k), u_2(k), y_k)$, $k=1,2,...,N$ was then provided for decoding.

This process can be called encoding since the text gets converted into a sequence of bits, which in turn gets transformed into the spatial series $(u_1(k), u_2(k), y_k)$, $k=1,2,...,N$ via the generation of uniform random variates and the mapping $f_{x(k-1),x(k)} : [-1,1] \times [-1,1] \to \mathbb{R}$. In [1], there are some good examples to show this encoding process. The problem is to utilize the information in this spatial series to obtain the original text.

We are given two data sets, A and B, created using the same encoding method, but using different sets of functions $f_{0,0}, f_{0,1}, f_{1,0}, f_{1,1}$. Set A contains the bit values $x(k)$ in addition to the three-dimensional series and is provided as a training set to help us develop a decoding method. This method is to be then applied to set B, which contains only the three dimensional series, and recovers the original English text.

Our approach to solving this problem employs a simple statistical property to estimate the most frequently used function out of the four functions $f_{0,0}, f_{0,1}, f_{1,0}, f_{1,1}$ and then moves on to estimate the remaining functions. We exploit the fact that the fraction of 1 bits in either of the binary sequences is approximately 0.135 and therefore the function $f_{0,0}$ is employed in approximately 75% of the mappings. We also employ a smoothing method based on the spatial statistics technique of Kriging [3]. We demonstrate that our method has high translation accuracy on set B.

The rest of the paper is organized as follows. Section 2 presents the proposed method. Experimental results and a generalization are given in Sections 3 and 4, respectively. Conclusions are given in Section 5.

2. Proposed Method

The propensity of zeros in the bit sequence and the smoothness of the four functions $f_{i,j}$, $i, j = 0$, 1,forms the foundation of our method. The smooth property of he functions implies continuity. Hence, given any $\alpha > 0$, we can always find a $\beta > 0$ such that,

$$\left| f_{i,j}(u_1, u_2) - f_{i,j}(u_1^*, u_2^*) \right| < \alpha .$$

if and only if $\left| (u_1, u_2) - (u_1^*, u_2^*) \right| < \beta$. (2)

Also, $u_1(k)$ and $u_2(k)$ are random variables from a uniform distribution, which means that the points $(u_1(k), u_2(k))$ will be spread across the entire square $[-1,1] \times [-1,1]$. Finally, as we observed before, approximately 75% of is the points $(u_1(k), u_2(k))$ will be transformed by the function $f_{0,0}$.

As mentioned earlier, the arguments $u_1(k)$ and $u_2(k)$ are independent, identically distributed random variables drawn from a uniform $(-1, 1)$ distribution. So, if we divide the area $[-1, 1] \times [-1, 1]$ into cells using a grid of equally spaced lines, every point $(u_1(k), u_2(k))$ will have an equal chance of falling into any one of the cells. Further, if we chose a fine enough grid, the smoothness property of the functions will ensure that given any two points (u_1, u_2), (u_1^*, u_2^*) within a cell, the difference $\left| f_{i,j}(u_1, u_2) - f_{i,j}(u_1^*, u_2^*) \right|$ will be very small. Moreover, except in situations where the functions cross one another, the values mapped by different functions would be separated within each cell. In short, we expect values mapped by each function to form separate clusters within each cell. We also know that approximately 75% of points $(u_1(k), u_2(k))$ are mapped by the function $f_{0,0}$. Since $u_1(k)$ and $u_2(k)$ are uniformly distributed, we can assume that, with high probability, over 50% of the points within each cell would also be mapped by $f_{0,0}$. Thus, the median of the y_k values of points within a cell must come

from the mapping $y_k = f_{0,0}(u_1(k), u_2(k))$, (see Figure 1). If at least one function crosses $f_{0,0}$ within the cell, then the median y_k may not come from a mapping involving $f_{0,0}$, but even then, this median value must fall within the range of values taken by $f_{0,0}$ inside the cell (see Figure 2).
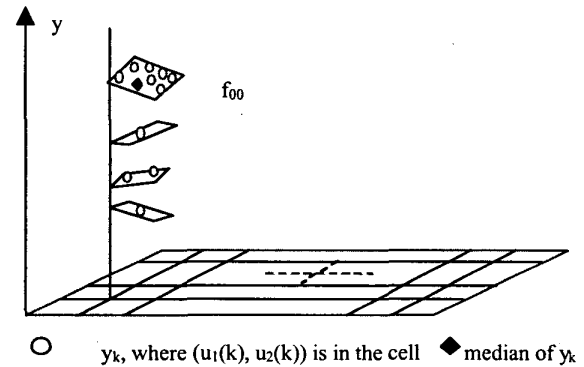

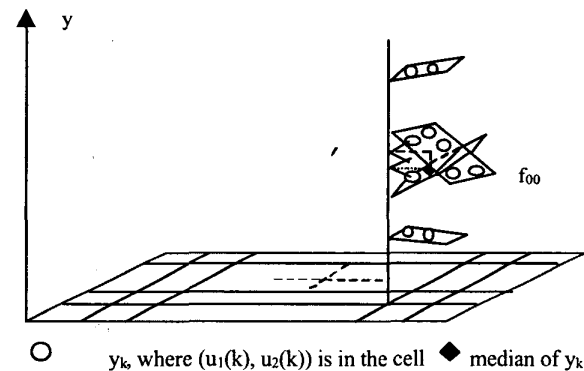**Figure 1:** Functions are separate within the cell

Figure 2: Functions cross each other within the cell

So we identify a subset of $f_{0,0}$ mappings (these are the medians) with a relatively high accuracy and the samples in this subset are spread out on the $[-1, 1] \times [-1, 1]$ square. We use this subset to estimate the function $f_{0,0}$ by a smoothing technique known as Kriging[3]. Now we can use this estimate $\hat{f}_{0,0}$ to pick up other points mapped by $f_{0,0}$ in the whole data set by using the criteria that if

$$\left| y_k - \hat{f}_{0,0}(u_1(k), u_2(k)) \right| < \delta \qquad (3)$$

1044

for some suitably chosen small $\delta > 0$, then y_k must be the result of a mapping by $f_{0,0}$; i.e. $(x(k-1), x(k)) = (0,0)$. We select a δ such that the number of y_k values picked as mappings of $f_{0,0}$ is roughly 75% of that of the whole data set. This approach enables us to identify almost all the bits that correspond to blank spaces. We assume that the "1" bits form only 13.5% of the total. We also assume that all the non-blank characters are capitals coded in ASCII. These assumptions, if true, allow us to estimate the function $f_{0,0}$ as described above and identify almost all the blank spaces in the set. We then proceed to estimate the functions $f_{0,1}$

and $f_{1,0}$ using the fact that the ASCII representation of every character begins with the bit sequence "010". Since the first two bits $(k-1, k)$ of every eight-bit word not identified as a space is 01 (with the exception of special characters), the data $(u_1(k), u_2(k), y_k)$ corresponding to these pairs of bits can be used to estimate $f_{0,1}$. The estimation is carried out using smoothing based on Kriging. We also estimate $f_{1,0}$ similarly, using the second and third bits of every non-blank character. Errors can creep into these estimates $\hat{f}_{0,1}$ and $\hat{f}_{1,0}$ because not every non-blank is an alphabetical character; they can be special characters such as ",", ".", "?". However, we know they are few and far apart in regular English text. Thus the estimated functions will represent the true functions fairly accurately. The forth function $f_{1,1}$ can now be estimated by picking the points $(u_1(k), u_2(k))$ transformed by $f_{1,1}$ using the criteria that if

$$\left| y_k - f_{i,j}(u_1(k), u_2(k)) \right| > \eta,$$
for all $(i,j) = (0,0), (0,1)$ and $(1,0)$ \qquad (4)

then y_k must be a the mapping of $(u_1(k), u_2(k))$ by $f_{1,1}$, where $\eta > 0$ is a suitably chosen constant. Once the four functions were estimated as indicated above, we evaluated the sequence $\{x(k)\}_{k=1}^{N}$ using the "least error criteria":

$$(x(k-1), x(k)) = (i, j) \text{ if}$$
$$\left| y_k - \hat{f}_{i,j}(u_1(k), u_2(k)) \right| \le \left| y_k - \hat{f}_{r,s}(u_1(k), u_2(k)) \right| \quad (6)$$

for all $r,s = 0,1$. Once the bit sequence is obtained, it is a trivial task to translate it to English text.

In order to improve translation accuracy and correct interpolation errors, we apply a consistency checking along with translation. The samples are generated by the four functions with a window of length two slid across the ASCII sequence to determine the function ID. When a sample is decoded, the corresponding function ID, which will form the ACSII bit of decoding text, should be consistent with the ID of the function just before it. For example, $f_{1,0}$ can follow neither $f_{0,0}$ nor $f_{1,0}$. The least error criteria may pick a wrong function at certain bit positions because of estimation errors. However, if consistency is met for several consecutive bits, the possibility of making a mistake drops dramatically. We also use the fact that every eight-bit word begins with a zero and hence the two-bit sequences containing the last bit of one word and the first bit of the next word cannot be 01 or 11. If consistency is violated at some bit, we go backward from the first bit of the next word and try to correct the error. If not, we write down all the possible letters and leave them for text content to fix. Or we may do another interpolation by picking samples at consistent bits. It is usually used for $f_{1,1}$, which has the least accuracy of the four, when necessary.

Procedure Consistency Checking
Begin
1. Initiate decoded_text_bit sequence T, N = number of samples of set B.
2. Forward and backward checking:
   for (I=1; I<N; I=I+8)
   { J=0;
     while (J<=7 && consistent)
     {
       Error_i = |y(I+J)-f_i(u_1(I+J),u_2(I+J))|;
                                          i=00,01,10,11.
       mark consistent bit T(I+J).
       J=J+1;
     } //end of while
     stop bit =J;
     if (stop bit <=7)
     { K=7;
       while(K>=stopbit && consistent)
       {
         Error_i = |y(I+J)-f_i(u_1(I+K),u_2(I+K))|;
                                          i=00,01,10,11.
         mark consistent bit T(I+K).
         K=K-1;
       } // end of while
       if (K<>J)
         mark all non-consistent bits T(I+J) ~ T(I+K);
     } // end of if
   } // end of for
3. Refine non-consistent bits.
4. Decode the bit sequence to ASCII.

5. If the text decoded is different from the previous one, then use all consistent bits to interpolate f_i and go to step1.

End of procedure

3. Experimental Results

Set B contains 30720 real-value triplets, which encodes an unknown text. We divide the $[-1, 1]\times[-1, 1]$ into $40\times40=1600$ cells. Using inverse-distance interpolation, provided by TECPLOT[2], we rebuild the four functions and decode the text. Finally, we point out the errors of inverse-distance interpolation by consistency checking and correct them with Kriging [3]. The final results are shown on Table 1.

Table 1: Decoding results of set B

	bit	space	character	letter (sp+ch)
Total in set B	30720	2564	1276	3840
Correctly decoded	30663	2564	1228	3792
percentage (%)	99.81	100	96.24	98.75

Moreover, we can correct all those 48 wrongly decoded characters if context is applied. Therefore, our submitted results to the text decoding competition [1] were 100% correct.

4. Generalization of Method

The method described above is based on several assumptions. First, we assumed that the "1" bits form only a small fraction of the bit sequence. Second, we assumed that all alphabetic characters are capitals and the special characters are few and far between. Third, the bit sequence was assumed to come from an ASCII representation of printable characters. We can generalize the method to work even when the first two assumptions are violated. We however, need the third assumption for this generalization to work.

In the original method, we had a majority of "00" bit pairs, which enabled us to estimate the function $f_{0,0}$. This approach is not feasible if we cannot assume the dominance of "00" pairs. Instead we make the following observations. If we take the bit pairs that represent the last bit of one eight-bit word and the first bit of the next word, then these can only be either 00 or 10. Also, if we look at the first two bits of each eight-bit word, they can only be 00 or 01 (under the last assumption). We first look at the subset of bit pairs forming the last bit of a word and the first bit of the next word. Assuming that either "00" or "10" is a majority (i.e.

there is no exact 50-50 split), the median in each cell (when limited to points in this subset) should represent either $f_{0,0}$ or $f_{1,0}$. Using these medians and Kriging we estimate the "Majority Function", say f_{MJ}. Similarly, we can estimate the Majority Function f^{*}_{MJ} using the medians obtained from the subset of data representing the bit pairs from the first two bits of each eight-bit word. This latter function will represent one of $f_{0,0}$ and $f_{0,1}$. Points in each subset that are not "close" to the Majority Function obtained for that subset are then used to estimate another function (using Kriging). We will call this function the "Minority Function". Let us denote these by f_{MI} and f^{*}_{MI}. The problem now is to assign these to $f_{0,0}, f_{0,1}$, and $f_{1,0}$. If f_{MJ} matches the data points that satisfy f^{*}_{MJ} or f^{*}_{MI}, then it must be $f_{0,0}$. Otherwise it should be $f_{1,0}$. Similarly, if f^{*}_{MJ} matches either f_{MJ} or f_{MI}, then it must be $f_{0,0}$; otherwise it is $f_{0,1}$. Having estimated the three functions $f_{0,0}, f_{0,1}$, and $f_{1,0}$, we can estimate $f_{1,1}$ using the points that do not match the three estimated functions (as was done in the original method). Once the four functions are estimated, we can used the least error and consistency checking described in Section 2 to the entire data set and obtain the bit sequence.

5. Conclusion

In this paper, we present an efficient method based on statistical reasoning to decode a real-value three-dimensional sequence into English text. The experimental results illustrate that our method works extremely well for ASCII code. We achieved 100% accuracy on the IJCNN decoding challenge problem [1] using this technique.

References:
[1] http://www.geocities.com/ijcnn/chal2.html
[2] TECPLOT, Amtec Engineering, Inc.
[3] Davis, J.C., *Statistics and Data Analysis in Geology*, Second Edition, John Wiley & Sons, Inc., New York, New York, 1986.