Pattern Recognition Letters 3 (1985) 309–313
North-Holland

September 1985

Median strings

Teuvo KOHONEN
Helsinki University of Technology, Department of Technical Physics, SF-02150 Espoo 15, Finland

Received 26 February 1985

Abstract: The concept of 'median string' over a set of strings is introduced in this paper. The set median is defined to be that element in the set of strings which has the smallest sum of distances (with respect to an arbitrary measure) from the other elements, and the (generalized) median is a hypothetical, artificially constructed element which has the smallest sum of distances from all the elements of the given set. The applicability of the median strings in the recognition of garbled strings is demonstrated, where they mostly yield a higher accuracy than the multiple-similarity and the *k*-nearest-neighbour methods.

Key words: Error correction, generalized median, symbol strings, word recognition, text correction.

1. The scope of this paper

Output from optical character readers, phonetic transcriptions from speech-recognition devices, and sequences of character codes transmitted through noisy communication channels are examples of error-contaminated *symbol strings* which may be processed, e.g., by pattern-recognition methods. Another application in which automatically produced strings of discrete symbols are generated and subjected to automated analysis occurs in the syntactic recognition of images (e.g., Fu (1974)). On the other hand, although most numeric expressions are strings in themselves, too, an erroneous digit may radically change the value of the expression, whereby the methods discussed in this work are not directly applicable. The ideas propagated in this paper thus only relate to strings in which the symbol positions have equal weights; the various symbols, of course, may have different a priori probability of occurrence.

The three basic types of error in strings of discrete symbols are: (1) replacement, (2) insertion, (3) deletion of a symbol. (Interchange of two consecutive symbols is reduced to two of the above basic operations in many ways.) The central theme in this paper is smoothing or averaging over repeated erroneous strings. Averaging over sets of numbers is a familiar operation; it is much less obvious what kind of averaging methods ought to be applied onto strings which change their lengths and become 'warped' due to insertion and deletion errors. To get an idea of the problem, consider the following garbled versions of the word 'HELSINKI' shown in Table 1. These strings contain on the average 50 per cent errors of all the above types. How could we, conversely, deduce the correct form from the erroneous samples?

Table 1
Garbled versions of the word 'HELSINKI'

HLSQPKPK	HOELSVVKIG
THELSIFBJI	HELSSINI
EOMLSNI	DHELSIRIWKJII
HEHTLSINKI	QHSELINI
ZULSINKI	EVSDNFCKVM

An important case in which averages of strings would be needed occurs in the identification of isolated, partly garbled words: there, an unknown or test string is usually compared with a set of reference strings which belong to various classes

and represent, e.g., statistical samples of the erroneous forms. The speed of computation could be increased and the demand for memory reduced significantly if the multiple prototypes of a class could be replaced by a single hypothetical reference string which is some statistical average of that class. It is surprising, and we shall demonstrate it below, that even the classification accuracy may thereby be improved. On the other hand, if every class were represented by a multitude of prototypes, existence of a 'class average' would also facilitate an improved choice of representative samples since the 'outliers', i.e., the samples most distant from the average, can then be identified and discarded. (In statistics, the average computed for such a reduced set is called 'trimmed mean'.)

2. Measures of dissimilarity between strings

An insertion or deletion error changes the relative position of all symbols to the right of it, whereby, e.g., the Hamming distance is not applicable. There are two categories of distance measures which take into account the 'warping' of strings: (1) dynamic programming, which usually computes the minimum number of editing operations (replacements, insertions, and deletions of symbols) needed to change one string into another; these operations can be weighted in many ways, (2) comparison of strings by their local features, e.g., substrings of N consecutive symbols (N-grams), whereby the respective features are said to match only if their relative position in the two strings differs in no more than, say, p positions; the distance of the strings is then a function of their lengths and the matching score.

2.1. Levenshtein distances

The most familiar distance between strings may be the Levenshtein distance (LD), which for strings A and B is defined as

$$\mathrm{LD}(A, B) = \min\{a(i) + b(i) + c(i)\}. \quad (1)$$

Here B is obtained from A by $a(i)$ replacements, $b(i)$ insertions, and $c(i)$ deletions of a symbol. There exists an infinite number of combinations $\{a(i), b(i), c(i)\}$ to do this, and the minimum is sought, e.g., by a dynamic programming method (cf. Table 2).

Since the various types of error occur with different probabilities and depend on the occurring symbols, a classification decision based on distances is usually more reliable if, in the distance measure, the editing operations are provided with different weights. This then leads to the weighted Levenshtein distance (WLD)

$$\mathrm{WLD}(A, B) = \min\{pa(i) + qb(i) + rc(i)\} \quad (2)$$

where the coefficients p, q, and r may be obtained from the so-called confusion matrix of the alphabet, as the inverse probability for a particular type of error to occur.

Table 2
Algorithm for the computation of WLD

```
begin
D(0,0):=0;
for i:=1 step 1 until length(A) do D(i,0):=D(i-1,0)+r(A(i));
for j:=1 step 1 until length(B) do D(0,j):=D(0,j-1)+q(B(j));
for i:=1 step 1 until length(A) do
  for j:=1 step 1 until length(B) do begin
    m1:=D(i-1,j-1)+p(A(i),B(j));
    m2:=D(i,j-1)+q(B(j));
    m3:=D(i-1,j)+r(A(i));
    D(i,j)=min(m1,m2,m3);
  end
WLD:=D(length(A),length(B));
end
```

2.2. Maximum posterior probability distance

Closely related to WLD is the maximum posterior probability distance (MPR) which refers to the most probable sequence of events in which A is changed into B. It is defined as

$$\mathrm{MPR}(A, B) = \mathrm{prob}\{B \mid A\}. \quad (3)$$

If the errors can be assumed to occur independently, a dynamic-programming method resembling that of WLD (Table 2) is applicable to the computation of MPR, too. The algorithm is presented in Table 3.

Table 3
Algorithm for the computation of MPR

```
begin
D(0,0):=1.0;
for i:=1 step 1 until length(A) do D(i,0):=D(i-1,0)*r(A(i));
for j:=1 step 1 until length(B) do D(0,j):=D(0,j-1)*q(B(j));
for i:=1 step 1 until length(A) do
  for j:=1 step 1 until length(B) do begin
    m1:=D(i-1,j-1)*p(A(i),B(j));
    m2:=D(i,j-1)*q(B(j));
    m3:=D(i-1,j)*r(A(i));
    D(i,j)=m1+m2+m3;
  end
MPR:=D(length(A),length(B));
end
```

2.3. Feature distance

An N-gram is a substring of N consecutive symbols; usually $N=2$ (bigram or digram), or $N=3$ (trigram). It is a matter of choice whether the leading and/or trailing blanks are included in the original string, or whether some of the N-grams are defined end-around with respect to the original string. Here we do not consider end-around N-grams but the leading and trailing blanks are included.

In this work, the N-grams are said to match only if the relative displacement of an N-gram in string B is $+1$, 0, or -1 with respect to that in string A. Let L_a, L_b be the lengths of strings A and B, respectively, and let e be the number of matching N-grams in the above sense. The feature distance FD (Kohonen, 1984; Kohonen et al., 1984) is defined as

$$\mathrm{FD}(A, B) = \max(L_a, L_b) - e. \tag{4}$$

This measure has been studied extensively in the context of speech recognition where it has compared favourably with the WLD and MPR. While it may yield a slightly inferior classification accuracy, it has one significant merit over the latter two: There exists an extremely fast searching and comparing method based on the FD. This method, named redundant hash addressing (RHA) (loc. cit.), locates the best-matching candidates directly without scanning all the reference strings, and the speed of comparison can be orders of magnitude higher than if conventional methods were used.

3. Definition of generalized median and set median

Let us first recall a couple of results from elementary statistics. If $N=\{a(1), a(2), \ldots, a(n)\}$ is a set of n real scalar numbers, then the arithmetic mean of N, denoted m, satisfies the relation

$$\sum_{i=1}^{n} [a(i) - m]^2 = \min. \tag{5}$$

On the other hand, the median M (which, by definition, is an element of N such that if the $a(i)$ were ordered, M would be midmost) satisfies another relation

$$\sum_{i=1}^{n} [a(i) - M] = \min. \tag{6}$$

If n is odd and the $a(i)$ are different, M is unique; otherwise the solution may be multiple.

We may now generalize relation (6) for arbitrary elements and distance measures. If $d[x(i), x(j)]$ is the distance between elements $x(i), x(j) \in S$, then the *(generalized) median* M shall satisfy the relation

$$\sum_{i=1}^{n} d[x(i), M] = \min. \tag{7}$$

Notice that M may not need to belong to S. If M is restricted to the elements of S, then it may be called *set median*. (Notice that in the case of scalar numbers, the median was always the set median.)

4. Examples of median

The present method has been tested for natural as well as artificially generated data. Artificial errors of all the basic types, with controllable rates, can conveniently be generated by a random-number-controlled algorithm. Such strings are used in the present work.

The set median is found easily, by computing all the mutual distances between the given elements, and searching for that element which has the minimum sum of distances from the other elements. The (generalized) median is then found by systematically varying each of the symbol positions of the set median, making 'errors' of all the

Table 4
Medians of garbled strings. LD-Levenshtein distance; FD-feature distance

Correct string: HELSINKI
Garbled versions (50 per cent errors):

1. HLSQPKPK	6. HOELSVVKIG
2. THELSIFBJI	7. HELSSINI
3. EOMLSNI	8. DHELSIRIWKJII
4. HEHTLSINKI	9. QHSELINI
5. ZULSINKI	10. EVSDNFCKVM

Set median (LD): HELSSINI
Median (LD): HELSINKI

Set Median (FD): HELSSINI
Median (FD): HELSSINI

Correct string: RECOGNITION
Garbled versions (50 per cent errors):

1. REJOGNEITION	6. RXONUIUOK
2. UCGNRTION	7. RRCOOGEPIONN
3. RECOGUITPON	8. ECOBNFITIUOND
4. WEIOTNNIHTMIOJ	9. RUPCOWGNIPZTHUN
5. RRSCGNXIIUN	10. RCOGQNOIRTON

Set median (LD): RECOGUITPON
Median (LD): RECOGNITION

Set median (FD): RECOGUITPON
Median (FD): RECOGUITION

Correct string: IAPR
Garbled versions (75 per cent errors):

1. APRTX	6. IMPQHI
2. APWU	7. LAIPR
3. OFR	8. IVLP
4. KIIAMN	9. YIAIPR
5. VDPR	10. IWZAHPCC

Set median (LD): LAIPR
Median (LD): IAPR

Set median (FD): VDPR
Median (FD): IAPR

three types over the whole alphabet, and checking whether the sum of distances from the other elements is decreased. This may sound cumbersome, but the computing time is quite modest; unless the error rate is very high, the median is usually found in the vicinity of the set median in one cycle of variation, as seen from the examples.

Table 4 gives a few typical examples of sets of erroneous strings for which the median and set median have been computed, using the Levenshtein distance and the bigram distance method, respectively. It is obvious that the former gives a higher fidelity to the error-free string. On the other hand, if the purpose were to recognize strings, then the same distance measure ought to be used in the classification operations as well as in the formation of medians (cf. Section 5).

Notice that no causal reason exists for the median being equal to the correct string; the median is always computed relative to the error statistics.

Nonetheless, in almost half of tens of similar experiments (with error rates complying with those of Table 4), the LD median was correct, and in the rest of the cases it differed from the correct form in only one symbol position.

5. Recognition experiment

The present section demonstrates the power of the medians in defining a good single prototype for each string class.

This experiment simulates the postprocessing stage of, say, an isolated-word speech recognition system. Artificial strings were selected for the purpose of obtaining closely controllable data for the comparison of the methods. For simplicity, we used the bigram measure for medians as well as for the different classification methods. The median method was compared with two widely used algorithms: the multiple-similarity, and the k-nearest-neighbours (kNN) classification. In the multiple-similarity method we computed the sum of the distances of the test item from the prototypes of each class. In the kNN classification we used $k = 3$, since closer examination did not reveal any significant differences for higher values of k, and the results were rather stable.

The strings to be recognized, one hundred in number, we natural forenames from the beginning of the Finnish calendar. There were 27 four-letter names, 41 five-letter names, 19 six-letter names, and 13 longer ones. For each name, ten erroneous class prototypes (on which the classification was based) were generated. The rates of replacement,

insertion, and deletion errors were equal on the average, and the percentage of errors was varied between 5 and 22.5. Similarly generated, but statistically independent strings were used for comparison. The classification accuracies are given in Table 5.

Table 5
Recognition of names; percentage of errors vs. percentage of string errors. Med.-median method; MFD-multiple similarity (feature distance) method; 3NN-three-nearest-neighbours method

String errors, per cent	Recognition errors, per cent		
	Med.	MFD	3NN
5	1.36	1.36	1.09
7.5	2.29	2.62	2.15
10	3.51	4.48	3.53
12.5	4.79	6.61	4.95
15	6.14	8.51	7.15
17.5	7.87	11.47	9.25
20	9.62	13.8	12.51
22.5	13.27	17.6	15.64

It is surprisingly clear that the median method is superior over the other two. Moreover it is more robust than the latter since its relative accuracy compared with the other methods increases with the increasing error rate.

6. Conclusion

Definition of median over a set of strings seems rather useful, not only for the smoothing of erroneous versions, but in providing a robust single reference or prototype for string classification. This mode of use seems to be worth further investigation.

References

Fu, K.S. (1974). *Syntactic Methods in Pattern Recognition.* Academic Press, New York.

Kohonen, T. (1984). *Self-Organization and Associative Memory.* Springer, Berlin-Heidelberg-New York-Tokyo.

Kohonen, T., H. Riittinen, E. Reuhkala and S. Haltsonen (1984). *Information Sciences* 33, 3-30.