Report For Summer Work(first draft)

Zhuqi Wang

July 22, 2014

1 Describing the problem

This problem is about finding error data and missing data in a given sample with a given template. The given template is an integer stream with length N, and each of its elements is in [0,4095]. The given sample is should be a similar stream with length M. The whole process could be described as: Imagine that a transmitter sends the template recurrently and only sends an element for each time. Notice that we may start sending data from each possible position of template stream, which leads to the problem more complicated. Then there is also a receiver receiving these sent data continually. And this is not a stable process which means we have chance to get some wrong data or miss some of them. And the p denotes missing rate and q denotes error rate.

2 Difficulties of finding solution

In fact, this problem is an intractable problem, we will never find a perfect solution. An simple example to demonstrate this case is that if we miss some data in the front of the whole stream or in the end of the whole stream continually, we can do nothing to exam this kind of missing. Cause we may start receiving from any position and finish receiving in any position. In addition, the error data is also in [0,4095], which leads to a frustrating result of failing to detect error data.

3 Some auxiliary restriction

According to what we discuss above, we can conclude that it is impossible for us to solve such a problem without any other condition. So what i want to do is adding a necessary limiting condition. We should make the sample match the template with least reconstruction cost. About how to design the reconstruction cost function will be introduced in following sections. Now we have already got a more tractable problem.

4 Designing cost function

Recall that we want to match the sample and template with error rate and miss rate as little as possible. First of all, there is no doubt that we would neglect all missing data in the first part and the end part of sample stream. Based on these assumption, a simple idea is to design the cost function as num(error)+num(missing). We can solve this problem with some brutal algorithm, which is easy to realize but time-consuming. If we want to accelerate this process, we can use some pruning strategy.

In our solution framework, we focus on minimizing incorrect rate. So we find the case in which there is the least Levenshtein distance(edit distance) between sample and template. Also we can also design other cost function, which may be thought more reasonable. For example, we can take the distribution of incorrect data into consideration. We can assume that the missing rate and error rate is so low that it is unlikely to generate too many successive incorrect data. And we can design a particular cost function to prevent successive incorrect data from being produced in our model. If we use $S = \{s_1, s_2 \dots s_n\}$ represents incorrect data set and s_i means the ith successive incorrect data cluster. And the cost function= $e^{\lambda|s_1|} + e^{\lambda|s_2|} + \dots e^{\lambda|s_n|}$. λ is control parameter which is used as trade-off between incorrect rate and cluster rate. Generally, all function with property that $f(x + y) > f(x) + f(y)$ can be used for eliminating successive incorrect data.

5 Algorithm

If we only focus on the problem of matching sample and template, we can take dynamic programming strategy. Our problem looks extremely like edit distance problem and the only difference is the possible different measure way of distance. This problem is a classic dynamic programming problem which have had a existing method of solving with complexity of O(nm)(n, m are length of two strings). For more details about algorithm you can refer to wikipedia.

6 Drawbacks

The drawbacks of our model is obvious. The algorithm is not efficient enough to be applied in large data set. If we take all possible missing rate into consideration, the problem will become more time-consuming. However, the good news is that when consider different missing rates, we can easily combine the parallel algorithm with our model. Cause different missing rates do not interact with each other.

And another worthy problem is memory occupying, while the dynamic programming strategy uses trades memory for time in reality. In our solution we may use n*m*sizeof(int)(n, m are length of sample and length) bytes memory to record intermediate states and another n*m*sizeof(int)(n, m are length of sample and length) bytes memory to record editing details. If we perform our algorithm in huge large data, the memory will become the limiting factor.

7 Any probability model?

When solving large data problem, sometimes probability model is a state-of-art method. In this section, we will discuss whether we could construct a reasonable probability model for solving our problem.

Surely, probability model can solve many problems with large data, and our problem seems like to be highly related to probability. However, there are some

important issues need to be discussed.

First of all, if we want to solve problem with probability method, it is necessary to hold large scale of samples. In this problem, it is hardly to gather enough samples for training model.

Secondly, we should understand what we can get from constructing probability model. If we want to learn the missing rate or error rate, using probability model maybe a good choice. However, if want to detect the position of error bits or missing bits, using probability will not be helpful while these are events of equal probability.