Reservoir Sampling

Anže Pečar, Miha Zidar

***Abstract*—In this article we are going to look at some of the techniques used for sampling continuous data. We will make an overview of the algorithms used in the past and present and present their shortcomings as well as their advantages.**

***Index Terms*—online learning, continuous data,**

I. Introduction

THE problem with random sampling is to select a random sample of size n from a set of size N. Many algorithms have been developed for this problem when the value of N is known beforehand. Some problems, that we encounter in the real world, do not have a specified N or N cannot be determined efficiently. Those kind of problems will be the focus of this paper. We shall take a look at older algorithms such as Algorithm R which was developed to efficiently and accurately create a random sample from a tape in one pass, as well as newer algorithms, such as online learning and some of it's improvements.

Many reservoir algorithms make the assumption that the data does not change over time. Algorithm R, for an example, samples starting data with a greater frequency than the data at the end of the tape, which is efficient but it fails to react to emerging trends in the data. If we are interested in trends a different kind of algorithm needs to be used. For this reason we shall describe a basic online learning algorithm and some improvements for it.

January 4, 2012

II. Algorithm R and its improvements

A. Motivation

In order to better understand the complex algorithms in use today we first need to understand the idea behind simpler algorithms used in the past. In this section we shall describe Algorithm R and its improvement - Algorithm Z.

B. Algorithm R

Algorithm R is a reservoir algorithm written by Alan Waterman. The basic idea behind reservoir algorithms is to select a sample of size $\geq n$, from which a random sample of size n can be generated. The purpose of Algorithm R is generating a random sample small enough to fit into working memory, from a continuous data stream such as a tape. The algorithm works as follows: we fill up our reservoir of n records with the first n records of the file we are processing. The $t+1$st record then has a $n/(t+1)$ chance of being in our reservoir of size n. The candidate it replaces is chosen randomly from the n candidates.

Below is the implementation of Algorithm R in $pseudocode$:

```
N = [0] * n # initial reservoir
for i in xrange(n):
        N[i] = READ_NEXT_RECORD()
t = n
while EXISTS_NEXT_RECORD():
        t += 1
        M = randrange(0, t, 1)
        if M < n:
                N[M] = READ_NEXT_RECORD()
        else:
                READ_NEXT_RECORD()
```

Explanation:
The list N is our reservoir in which we store a random sample of records. We start of with an empty list of length n in which we store the first n elements (the for loop). The function $READ\_NEXT\_RECORD()$ returns the next record in the stream. After we have inserted the first n elements into our reservoir, we enter the $while$ clause, which runs as long as there are still records in the stream. In each run of the $while$ clause we first increment the counter of records t and then calculate a random number between 0 and t. We store the random number in M. If M is lower than n we store the record at index M, otherwise we skip the record. [AM I EXPLAINING THE OBVIOUS?]

C. Algorithm Z

It turns out we do not need to go over every single record in order to get a random set in the reservoir. We can skip a random number of records each time just as long as the probability of a record being in the reservoir does not change. This is the idea behind Algorithm Z.

Below is the first part of Algorithm Z implemented in $pseudocode$:

```
N = [0] * n # initial reservoir
for i in xrange(n):
        N[i] = READ_NEXT_RECORD()
t = n
thresh = T * n
num = 0
while EXISTS_NEXT_RECORD() and t <= thresh:
        t += 1
        num += 1
        V = random()
        S = 0
        quot = num/t
        while quot > V:
                S += 1
```

```
            t += 1
            num += 1
            quot = (quot * num)/t

    SKIP_RECORDS(S)

    if EXISTS_NEXT_RECORD():
            M = randrange(0, n)
            N[M] = READ_NEXT_RECORD()
```

Similarly to Algorithm R, Algorithm Z starts of by putting the first n elements into the reservoir. It then proceeds into the while loop, which runs until the variable t is below the threshold variable $thresh$. The second while loop is used to calculate the number of skipped elements S. The value of S will be larger after each iteration of the first while loop, making the algorithm skip more and more elements each time.

The above algorithm works best when it's $thresh$ value T is between 10 and 40. After the t value reaches the $thres$ value the second part of the algorithm begins. The implementation of the second part of the algorithm, known as the rejection technique, is complicated, and we shall only describe the basic idea behind it. The reader can get the full algorithm in [1].

The rejection technique generates S even more slowly than in the first part of $Algorithm\ Z$, which makes the algorithm faster as there are even more records that we are able to skip.[TODO][WHY DO WE NEED IT?]

D. Conclusion[?]

As we can now see, the main difference between $Algorithm\ R$ and $Algorithm\ Z$ is that the former is slower as it computes the probability for every single record while the latter is much faster as it skips as many records as possible. Both algorithms give us a random sample, however, none of them is ideal for detecting emerging trends in the data. In the next chapter we shall focus on algorithms that do just that.

III. Online learning

A. Introduction

Online learning algorithms are used when we want our model to learn one instance at the time.

IV. Conclusion

The conclusion goes here.

Appendix A
Appendix title

Appendix one text goes here.

Appendix B

Appendix two text goes here.

Acknowledgment

The authors would like to thank...

References

[1] J. S. Vitter, *Random Sampling with a Reservoir*. Brown University, 1985.
[2] N. Littlestone, *Learning Quickly When Irrelevant Attributes Abound: A New Linear-threshold Algorithm*. University of California, 1988
[3] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

Anže Pečar Biography text here.

PLACE
PHOTO
HERE

Miha Zidar Biography text here.