DS-412 Report

Measures of Dispersion

for

Streaming Data

Name: Riona Chakrabarti
Enrollment No. : BT18GCS163
Section : C4

Table of Contents:

Abstract:

The most popular and conventional methods of computing dispersion are usually limited to unidimensional data sets. With the evolution of IoT devices, streaming data is being generated by various sources. The velocity and volume of these types of data are uncontrollable and thus require processing without exceeding the available storage capacity. This establishes the need for one-pass algorithms that can be used to compute variability of such data. This report discusses methods like incremental covariance algorithm and Welford's online algorithm for computing variance in time-varying data sets.

1. Introduction:

While the value of a central tendency gives us a single-point representative value of a data set, dispersion or variation is essential in obtaining a complete sense of the data, as it captures the variability of the data.

The focus of this report is to discuss the different measures of dispersion and find one that gives the most optimal results for stream processing. Managing streaming data is fundamentally different from managing other transactional data as the insights derived are the most valuable shortly after their generation, it’s value diminishing with time. Utilising batch processing, when it comes to streaming data, is not feasible as it would require vast storage space. This process would also require the stream to stop at some point or else result in an unnecessary buildup.

In this report, section 2 explores the existing methods of computing standard deviation and variance of a data set and the challenges they pose. Section 3 discusses various methods of dispersion for both unidimensional and multidimensional data, and some streaming algorithms. Section 4 discusses the importance and application of streaming algorithms in the real world, followed by Section 5 which where the effectiveness of Welford's algorithm versus the textbook one-pass algorithm is demonstrated by means of simulations.

2. Literature Survey:

Data in streaming setting has the following features:

- The number of observations is unknown,
- Requires performing some stopping test after each sample,
- The number of observations is too large to allow storage and batch processing.

There are a variety of streaming algorithms that exist for the computation of mean and variance of streaming data, based on the method of incremental computation. However, since computation of variance involves sums of squares, limitations like numerical instability and arithmetic overflow caused by the algorithm, when dealing with large values, is common.

The most straightforward method for computing the variance of a data set is the standard two-pass algorithm[5],

$$S = \sum_{i=1}^{N} (x_i - \bar{x})^2$$

where, the sample mean

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

The problems posed by the two-pass algorithms mainly occur when the data sample is too large to be stored in the main memory or when the variance has to be calculated dynamically while the data is being collected.

The definition of S is manipulated into the following form to avoid the above limitations.

$$S = \sum_{i=1}^{N} x_i^2 - \frac{1}{N} (\sum_{i=1}^{N} x_i)^2$$

This formula is referred to as the ‘textbook one-pass algorithm’. The computation in this algorithm can result in being numerically unstable if the values of $\sum x_i^2$ and $\frac{1}{N}(\sum x_i)^2$ are very large. This results in some rounding error, in which case the two-pass algorithm can also be unstable.

To overcome these drawbacks the ‘updating algorithm’ suggested by Youngs and Cramer[3] and Welford’s online algorithms[4] can be used as alternative one-pass algorithms. Both of these are more stable and return the variance as the sum of non-negative values.

D.H.D. West [2] suggested an incremental algorithm that supports mean and variance computation for positive weighted samples, by replacing the simple counter with the sum of weights of the incoming data.

Pébay *et al* [9] proposed formulae for incremental and pairwise update of the covariance value, thus making it a possibility to use covariance matrices for the computation of variance in streaming data.

3. Main:

The measures of dispersion differ in construction, properties and the situations in which they are used. Unfortunately, the most popular and commonly used measures can be used only for one dimensional data. Since most of the contemporary datasets tend to be multivariate, this poses a hindrance. The univariate dispersion measures may be applied to multidimensional data, to each variable separately, but this way we lose information on possible relations between variables.

3.1 Unidimensional measures of dispersion:

The measures of dispersion are called the Averages of Second Order and represent the extent of the scattering of observations about the mean, in a particular distribution. The measures can be categorised as Absolute Measure of Dispersion or Relative Measure of Dispersion. These may be categorised as Absolute Measures or Relative Measures.

Absolute measures such as range, quartile deviations, mean and standard deviation produces a value which is in the same unit as the original data set, and the variability is expressed using the average of the deviation of observations. It helps understand the dispersion within the context of the current experiments and measurements. Range:

Out of these, the range and the quartile values do not take all the observations into consideration, and are thus not very reliable measures of dispersion. Since the calculations use the extreme values of the distribution, these measures are also highly prone to sampling fluctuations. The method of squaring the deviations in standard deviation and variance overcomes the drawback faced by mean deviations for ignoring the signs and taking the absolute value and is also least affected by the fluctuations in the observations.

$$S = \sqrt{\frac{\sum (x - \bar{x})^2}{N}}$$

Using standard deviation and variance as measures of dispersion can be used to detect skewness. They are used along with mean as the measure of central tendency in the case of symmetric, numerical data.

However, these measures are inappropriate in the case of skewed data. In case of ordinal or skewed numerical data, a measure such as the interquartile range is used along with the median.

Relative Measures such as coefficient of range, coefficient of variation, coefficient of mean deviation etc. produce the result in the form of a ratio or percentage, and thus, are

dimensionless. They are useful in comparing separate data sets generated during different experiments, since the units do not have any role in these calculations.

3.2 Covariance Matrix

While the variance is used to measure the variation of a single random variable throughout different observations, the covariance value is used to express how much two different random variables vary together. The covariance $\sigma(x, y)$ of two random variables x and y is given by

$$\sigma(x, y) = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$$

where the number of samples is n. The covariance σ_{2x} of a random variable x can be also expressed as $\sigma(x, x)$. This means that the value is the covariance of x with itself.

The covariance matrix is a square matrix giving the covariance between each pair of elements of a given random vector. Each entry of the matrix is $C_{i,j} = \sigma(x_i, x_j)$ where $C \in R_d$, d being the dimension or the number of features in the dataset. The main diagonal of the matrix consists of the variances of each element, or the covariance σ_{2x}.

This way, a data set of n-dimensional observations can be represented by a matrix of 2 numbers instead of a single real value of a desired measure of dispersion characterizing somehow the whole multidimensional sample.

3.3 Covariance Matrix in streaming data

The formula mentioned in Section 3.2 is a two pass algorithm which involve the following steps:

1. Computation of the sample means

$$\bar{x} = \sum_{i=1}^{n} \frac{x_i}{n}$$
$$\bar{y} = \sum_{i=1}^{n} \frac{y_i}{n}$$

2. Computation of the covariances

$$\sigma(x, y) = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$$

This is not a feasible option while stream processing as the cost of distributed memory access overshadows the computation costs.

As an alternative we can use the one-pass online algorithm for calculating incremental covariance matrix which allows direct updates while being as numerically stable. These one-pass algorithms are compatible with stream processing as the data does not need to be stored.

1. The sample means can be updated as follows:

$$\bar{x}_n = \bar{x}_{n-1} + \frac{x_n - \bar{x}_{n-1}}{n}$$

$$\bar{y}_n = \bar{y}_{n-1} + \frac{y_n - \bar{y}_{n-1}}{n}$$

2. The co-moment is computed:

$$C_n = C_{n-1} + (x_n - \bar{x}_n)(y_n - \bar{y}_{n-1})$$

$$C_n = C_{n-1} + (x_n - \bar{x}_{n-1})(y_n - \bar{y}_n)$$

3. Then finally, the covariance can be calculated as follows:

$$Cov_N(X, Y) = \frac{C_N}{N}$$

Pseudocode:

Input : Samples $[x_1, x_2, ..., x_n]$ *and* $[y_1, y_2, ..., y_n]$

Output: Population Covariance and Sample Covariance

1. *Mean_x :=0*
2. *Mean_y :=0*
3. *Moment := 0*
4. *Counter := 0*
5. *for each x, y in data1, data2:*
    a. *Counter ++*
    b. *dx = x - Mean_x*
    c. *Mean_x =Mean_x + dx / Counter*
    d. *Mean_y += (y - Mean_y) / Counter*
    e. *Moment += dx * (y - Mean_y)*
6. *Population_covariance = Moment / Counter*
7. *Sample_covariance = Moment / (Counter - 1)*

However, the size of a covariance matrix is equal to the square of the dimension of the data set. This might cause problems for real data sets that have a very high number of dimensions.

As a solution, one could compute a low rank factorization of the matrix, since the size of a factor is only rank times the data and not data dimension squared. However, the computation of low rank factorisation of a covariance matrix is a two-pass algorithm and requires resetting of the data stream.

3.4 Welford's algorithm:

Wellford's online algorithm is a stable method for the computation of variance in just a single pass. This would prevent the need to store the data values as each x_i value shall

be inspected only once, making it an ideal algorithm for streaming data. The value can be calculated by subtracting the sums of squared difference for N samples from that of $N-1$ samples.

The formula is used to update the variance of a distribution after taking into consideration the additional, incoming element x_n.

$\bar{x_n}$ is the sample mean for the first n samples.

The population variance, $\sigma_n^2 = \frac{(n-1)\sigma_{n-1}^2}{n} + \frac{(x_n - \bar{x_{n-1}})(x_n - \bar{x_n})}{n}$

could be written as $\sigma_n^2 = \sigma_{n-1}^2 + \frac{(x_n - \bar{x_{n-1}})(x_n - \bar{x_n}) - \sigma_{n-1}^2}{n}$

This could result in numerical instability due to the fact that a small number is repeatedly subtracted from a large value which scales with n. Thus, according to Welford's method, the sum squared of the differences from current mean is used for the process of updation:

$$M_{2,n} = \sum_{i=1}^{n} (x_i - \bar{x_n})^2$$

Updating $M_{2,n}$

$$M_{2,n} = M_{2,n-1} + (x_n - x_{n-1})(\bar{x_n} - \bar{x_n})$$

Thus the population variance :

$$\sigma_n^2 = \frac{M_{2,n}}{n}$$

Pseudocode:

Input : Array of the sample [x_1, x_2,....., x_N]
Output : Incremental variance of the data stream.

1. *Mean := 0*
2. *S:= 0*
3. *Counter:=0*
4. *for each x in sample:*
    a. *Counter++*
    b. *old_Mean := Mean*
    c. *Mean := Mean + (x - Mean)/Counter*
    d. *S := S + (x-Mean)*(x-old_Mean)*
5. *return S/N*

4. Application:

Incremental computation of mean and variance of data streams are an essential step in processing the data generated during network monitoring , intrusion detection, fraud detection, financial monitoring and analysis of e-commerce websites. Due to the constraints in storage space and computation power, streaming algorithms are used to aggregate the incoming data points, thus maintaining summaries or a synopsis of the nature of the time-varying data.

5. Simulation

The following code is runs a simulation of the textbook one-pass algorithm on an array of data

```
def naive(samples):
      S = 0
      S2 = 0
      c = 0
      for x in samples:
            S += x
            S2 += x*x
            c += 1
      M = S/c
      v = (S2 - (S*S)/c)/(c-1)
      return M,v
```

The data stream is created using the random function. The range of the data varies from 0-100 and the array will consist of 20 points.

```
randomlist = []
for i in range(0,100):
      n = random.randint(1,20)
      randomlist.append(n)
```

```
runfile('C:/Users/Riona/untitled1.py', wdir='C:/Users/Riona')
Input:  [6, 15, 9, 12, 7, 11, 17, 7, 8, 7, 20, 19, 17, 4, 18,.......]
```

```
print("Textbook one-pass: ",naive(randomlist))
print("Numpy:   ", (np.mean(randomlist), np.var(randomlist)))
```

```
Textbook one-pass:  (11.34, 34.852929292929296)
Numpy:      (11.34, 34.504400000000004)
```

The result obtained on running the naive algorithm on the data stream gives the same mean as numpy's `np.mean()` function. However, the variance obtained by the naive algorithm differs from numpy's at the first floating point. This is because the textbook one-pass algorithm plays badly with floating point precision.

The next simulation will include taking data with a higher range and more elements in the stream.

```
randomlist = []
for i in range(0,10000):
      n = random.randint(1,9999)
      randomlist.append(n)
```

On running the naive algorithm on this input, the variance significantly differs than the one obtained by numpy.

```
Textbook one-pass:  (5020.0716, 8429625.482621703)
Numpy:      (5020.0716, 8428782.52007344)
```

Now, the welford's online algorithm is implemented to compare it's precision with `numpy.var()`'s

```
def welford(samples):
      c = 0
      for x in samples:
            c += 1
            if c == 1:
                  M = x
                S = 0
            else:
                M_next = M + (x - M) / c
                S = S + (x - M)*(x - M_next)
                M = M_next
      v=S/(c-1)

      return M,v
```

The results obtained have slightly more precision than the naive algorithm but are not the same.

```
Welford: (5025.318999999975, 8276910.97313633)
Numpy:     (5025.319, 8276083.282039)
```

`numpy.var()` has a `ddof` (Delta degrees of freedom) argument whose value is 0 by default. On setting the `ddof=1`, the `var()` function will use $N-1$ as the divisor, rather than N while calculating the variance.

Thus on making the following changes:

```
print("Welford:", welford(randomlist))
print("Numpy:   ", (np.mean(randomlist), np.var(randomlist, ddof=1)))
```

the results obtained are :

```
Welford: (5008.9033999999965, 8355298.724940919)
Numpy:      (5008.9034, 8355298.724940932)
```

Here the value of the variance is the same upto the 7th floating point.

These results show that the textbook one-pass algorithm for the calculation of variance may be used in stream processing, but it is highly numerically unstable. On the other hand, Welford's algorithm is numerically stable, gives accurate results even in case of floating points, and is a one pass algorithm, thus not requiring the data to be stored.

Conclusion:

The popular and standard methods of computing dispersion in unidimensional data cannot be applied to data with higher dimensionality. A covariance matrix not just provides the variability of one random variable throughout a dataset, but also gives the variance between two random variables throughout the distribution. However, the size of a covariance matrix is the square of the number of dimensions of the dataset, thus causing problems for high dimensional datasets. Incremental calculation of mean and variance by means of a one-pass algorithm, such as the Wellford's online algorithm, was seen to be ideal for data stream processing.

References:

1. Kołacz, A. and Grzegorzewski, P., 2016. Measures of dispersion for multidimensional data. *European Journal of Operational Research*, 251(3), pp.930-937. https://doi.org/10.1016/j.ejor.2016.01.011
2. West, D., 1979. Updating mean and variance estimates. *Communications of the ACM*, 22(9), pp.532-535.https://doi.org/10.1145/359146.359153
3. Youngs, Edward, A., and Cramer, E., 1971 . Some Results Relevant to Choice of Sum and Sum-of-Product Algorithms.*Technometrics*, 13(3), pp. 657–665. https://doi.org/10.2307/1267176
4. Welford, B., 1962. Note on a Method for Calculating Corrected Sums of Squares and Products. *Technometrics*, 4(3), pp.419-420. https://doi.org/10.1080/00401706.1962.10490022
5. Chan, T., Golub, G. and Leveque, R., 1983. Algorithms for Computing the Sample Variance: Analysis and Recommendations. *The American Statistician*, 37(3), pp.242-247. www.jstor.org/stable/2683386.
6. Chan, T., Golub, G. and Leveque, R., 1979. Updating formulae and a pairwise algorithm for computing sample variances. Technical Report. Stanford University, Stanford, CA, USA. http://i.stanford.edu/pub/cstr/reports/cs/tr/79/773/CS-TR-79-773.pdf

7. Glen, S., 2021. *Statistics How To: Elementary Statistics for the rest of us!*. [online] Statistics How To. Available at: <https://www.statisticshowto.com/ > [Accessed 2 June 2021].
8. Manikandan, S., 2011. Measures of dispersion. *Journal of Pharmacology and Pharmacotherapeutics*, 2(4), p.315. https://doi.org/10.4103/0976-500X.85931
9. Pébay, P., Terriberry, T., Kolla, H. and Bennett, J., 2016. Numerically stable, scalable formulas for parallel and online computation of higher-order multivariate central moments with arbitrary weights. *Computational Statistics*, 31(4), pp.1305-1325.