EL EFECTO DEL LÍMITE DE PRECISIÓN EN EL CÁLCULO DE LA MEDIA Y LA DESVIACIÓN ESTANDAR

Edgar Ruiz L.*

RESUMEN

Cuando se realizan cálculos en una computadora se encuentra con el problema de la precisión o *acuracidad* de estos. Para analizar este problema se examinan los cálculos de la media y la desviación estándar con un conjunto hipotético de datos de prueba. Se emplea el lenguaje de programación C++, utilizando dos algoritmos distintos en el cálculo; llegándose entre otros a concluir que la *acuracidad* depende del algoritmo empleado y de la capacidad de representación de los datos en una computadora.

Palabras clave : Acuracidad. Media. Desviación estándar. Varianza. Error absoluto.

ABSTRACT

When computer calculations are performed, the precision or *accuracy* problem arises. In order to analize this problem, calculations on the mean and standard deviation are examined with a hypothetical group of test data. The C++ programming language is used, employing two different algorithms in the calculation. It was concluded, among others, that accuracy depends on the algorithm used and on the capacity of performing a computer data representation.

Key words : Accuracy. Standard deviation. Variance. Absolute error.

INTRODUCCIÓN

Los ingenieros y científicos necesitan muchas veces conocer la exactitud de sus cálculos. Para ello utilizan herramientas que los ayudan en su labor, como calculadoras y computadoras; estas están sujetas a un límite de precisión, el cual es dado por su capacidad de representar los datos. El objetivo del presente artículo es mostrar las dificultades inherentes en los cálculos realizados en una computadora personal los cuales deben tenerse en cuenta cuando se requiere una muy buena precisión o *acuracidad* en los resultados.

LA MEDIA Y LA DESVIACIÓN ESTÁNDAR

En la estadística descriptiva dos de las medidas de tendencia central y dispersión mas comúnmente usadas son, la media y la desviación estándar. Dado un conjunto de números X_1, X_2,.....,X_n ; la media aritmética se define como:

$$\mu = \frac{X_1 + X_2 + ... + Xn}{n} \qquad (1)$$

La varianza esta definida como:

$$v = \frac{(X_1 - \mu)^2 + (X_2 - \mu)^2 + ... + (X_N - \mu)^2}{n} \qquad (2)$$

y la desviación estándar es:

$$\sigma = \sqrt{v} \qquad (3)$$

Haciendo uso de la ecuación (1) puede replantearse la ecuación (2) como:

$$v = \frac{X_1^2 + X_2^2 + ... + X_n^2}{n} - \mu^2 \qquad (4)$$

La carta N-S para un algoritmo que lee los valores X_1, X_2,..., X_n y calcula μ, v y σ usando las ecuaciones (1), (3) y (4) se muestra en la Figura 1. En ella la variable *suma* es el acumulador de los datos X_i leídos y la variable *sumacua* es el acumulador de los cuadrados de los X_i leídos. El valor de *contad* será él numero total de datos X_i leídos. TEST es un numero especial conocido como "centinela" el cual es escogido de tal forma que difiera notablemente de alguno de los valores X_i. Por ejemplo si calculamos la media y la desviación estándar de un conjunto de datos positivos, podemos entonces escoger TEST como –1 o cualquier otro número negativo. Si los datos están entre –100 y +100 podemos hacer que TEST sea –1000 o +1000.

*Ingeniero Industrial. Instituto de Investigación de la Facultad de Ingeniería Industrial. UNMSM.
E-mail : d260045@unmsm.edu.pe

Inicio
Variables x, suma,sumacua, contad son de tipo real media, var, desvstd son de tipo real
Escribir("Ingrese primer Valor de x")
Leer(x)
while X <> TEST
suma = suma + x
sumacua = sumacua + x^2
contad = contad + 1
Escribir("Ingrese siguiente valor (-1 para terminar)")
Leer(x)
media = suma / contad
suma = 0
var = sumacua / contad - media^2
desvstd = sqrt(var)
Escribir(media, var, desvstd)
Fin

Figura 1: Carta N-S para hallar le media, varianza y desviación estándar de un conjunto de datos

La prueba de la condición del while; x <> TEST hará que en cierto momento, cuando X=TEST se termine la adición de términos a *suma* y *sumacua*, cuando esto sucede los valores de *suma* y *sumacua* son

$$suma = \chi_1 + \chi_2 + ... + \chi_n \qquad (5)$$

$$sumacua = \chi_1^2 + \chi_2^2 + ... + \chi_n^2 \qquad (6)$$

y el contador *contad* tiene el valor *n*.

La técnica de usar un centinela permite al algoritmo indicar la finalización de datos en tiempo de ejecución del programa. Enseguida el algoritmo procede a calcular la media (ecuación (1)), la varianza (ecuación(3)) y la desviación estándar (ecuación(4)).

El programa en C++ que implementa[1] el algoritmo de la Figura 1, se muestra en la Figura 2. Obsérvese que *suma* y *sumacua* corresponden a las ecuaciones (5) y (6) respectivamente; las cuales permiten calcular la media, varianza y desviación estándar sin usar los datos individuales.

Si escogemos usar la ecuación (2) para él calculo de la varianza, las cosas no son tan simples. Debemos saber el valor de la media, μ, antes de iniciar él calculo de v. La carta N-S que hace esto se muestra en la Figura 3.

```
#include <iostream.h>
#include <stdlib.h> // para system()
#include <math.h>   // para sqrt()

const int TEST = -1;
int main()   //estadis0.cpp
{
    float x, suma, sumacua;
    int contad;
    float media, var, desvstd;

    suma = 0.0; sumacua = 0.0; contad = 0;
    cout<<«Ingrese primer valor de x «;
    cin>>x;
    while ( x != TEST )
```

```
    {   suma = suma + x;
        sumacua = sumacua + x*x;
        contad = contad + 1;
        //leer siguiente valor
        cout<<"Ingrese siguiente valor de x (-1 para terminar) ";
        cin>>x;
    }

    //calculo de media, varianza y desviacion estandar
    media = suma/contad;
    var = sumacua/contad - media*media;
    desvstd = sqrt(var);

    //impresion de resultados
    cout<<"Media = "<<media<<endl;
    cout<<"Varianza = "<<var<<endl;
    cout<<"Desviacion estandar = "<<desvstd<<endl;

    system("PAUSE");
    return 0;
}
```

Figura 2: Un programa en C++ correspondiente a la carta N-S de la Figura 1, para hallar la media, varianza y desviación estándar de un conjunto de números.

Inicio	
Variables x[MAX], suma, media, var, desvstd son de tipo real i, contad son de tipo entero	
Escribir("Ingrese primer Valor de x")	
Leer(x[i])	
while X <> TEST	
	suma = suma + x[i]
	contad = contad + 1
	i = i + 1
	Escribir("Ingrese siguiente valor (-1 para terminar)")
	Leer(x[i])
media = suma / contad	
suma = 0	
i = 0	
while (i < contad)	
	suma = suma + (x[i] - media)^2
	i = i+1
var = suma / contad	
desvstd = sqrt(var)	
Escribir(media, var, desvstd)	
Fin	

Figura 3: Carta N-S para un algoritmo diferente para hallar la media, varianza y desviación estándar de un conjunto de números, el cual requiere que la media sea calculada antes de calcular la varianza y además requiere el almacenamiento individual de los items de datos.

[1] Se utiliza el compilador Dev C++ 4.0; un compilador C/C++ del tipo GNU con licencia GPL

Si escogemos usar la ecuación (2) para él calculo de la varianza, las cosas no son tan simples. Debemos saber el valor de la media, μ, antes de iniciar él calculo de v. La carta N-S que hace esto se muestra en la Figura 3.

El programa que implementa la carta N-S de la Figura 3, se muestra en la figura 4. El programa cuenta él número de datos, pero estos se almacenan en un arreglo que no excederá de 200 (constante MAX). Por supuesto, que se podría almacenar un número de datos mayor y esto se logra cambiando el valor de la constante MAX. Se observa que el programa requiere un espacio de almacenamiento mayor (mas memoria) que el primero, puesto que se requiere memoria para cada dato. Adicionalmente, el segundo programa requiere mas aritmética que el primero.

```
#include <iostream.h>
#include <stdlib.h> // para system()
#include <math.h>   // para sqrt() y pow()

const int MAX = 200;
const int TEST = -1;

int main()   //estadis1.cpp
{
    float x[MAX], suma;
    int i = 0, contad;
    float media, var, desvstd;

    suma = 0; contad = 0;
    cout<<"Ingrese primer valor de x ";
    cin>>x[i];
    while ( x[i] != TEST )
    {
       suma = suma + x[i];
       contad = contad + 1;
       i++;
       //leer siguiente valor
       cout<<"Ingrese siguiente valor de x (=1 para terminar) ";
       cin>>x[i];
    }

    //calculo de media, varianza y desviacion estandar
    media = suma/contad;
    suma = 0;
    i = 0;
    while ( i < contad )
    {
       suma = suma + pow((x[i] - media),2);
       i++;
    }

    var =  suma/contad;
    desvstd = sqrt(var);

     //impresion de resutados
    cout<<"Media = "<<media<<endl;
    cout<<"Varianza = "<<var<<endl;
    cout<<"Desviacion estandar = "<<desvstd<<endl;

    system("PAUSE");
    return 0;
}
```

Figura 4: programa en C++ que implementa la carta N-S de la figura 3.

En el primer programa hay *n* multiplicaciones (cálculos de X^2) y *2n* adiciones (*suma* y *sumacua*) en el bucle; fuera de el hay dos divisiones, una adicción y una multiplicación; es decir. el primer método usando la ecuación (4) requiere:

$2n + 1$	adiciones
$n + 1$	multiplicaciones
2	Divisiones

No se han incluido las adiciones necesarias para contar él numero de datos que se han leído.

El segundo programa usa *n* adiciones en el primer bucle. Hay *n* multiplicaciones (calculo de $(X[i] - media)^2$) y *2n* adiciones (calculo de X[i] – media) y *suma + termino* al cuadrado en el segundo bucle. Adicionalmente hay una división antes del segundo while (media = suma/contad) y otra división para hallar la varianza (var = suma/contad). Entonces el segundo método usando la ecuación (2) requiere:

$3n$	adiciones
n	multiplicación
2	divisiones

Para valores grandes de *n* el segundo método requiere 50% mas adiciones y aproximadamente el mismo número de multiplicaciones y divisiones. Si se hubiera contado él numero de adiciones necesarias para el contador de los bucles el segundo método podría ser más ineficiente. Puede entonces hacerse uso de un método en lugar del otro?. La respuesta esta en la *acuracidad* relativa de los dos métodos: El segundo método tiene mayor *acuracidad* que el primero veamos por que.

Para mostrar las dificultades numéricas que puede acarrear el primer método considere el siguiente problema.

Se tiene 33 items de datos igualmente espaciados a una distancia $\Delta X = 1/16^4$ uno del otro. El valor más pequeño es $1-1/16^3 = 0.999756$ y él más grande es $1+1/16^3 = 1.00024$.

La media será 1; pues para cada numero más pequeño que 1 existe un correspondiente numero más grande que 1, y el promedio de estos dos números es 1. La varianza es pequeña. En verdad puede calcularse la varianza usando la ecuación (2). En (a).

El programa que se utiliza la carta N-S de la figura (1) y calcula media, la varianza y de la desviación estándar para este conjunto de 33 datos se presenta en la figura (5).

$$v = 2/33\left[\left(\frac{1}{16^4}\right)^2 + \left(\frac{2}{16^4}\right)^2 + \ldots\ldots\ldots + \left(\frac{15}{16^4}\right)^2 + \left(\frac{16}{16^4}\right)^2\right] = \frac{17}{3*16^7} = 2.110998*10^{-8} \qquad \text{(a)}$$

```
#include <iostream.h>
#include <math.h>
#include <stdlib.h>

void main() // estadis2.cpp
{
    float x, suma, sumacua;
    float media, var, desvstd;
    float del, cont;

    //inicializacion
    suma = 0.0;
    sumacua = 0.0;
    cont = 0.0;
    x = 1 - 1.0/pow(16.0,3);
    del = 1.0/pow(16.0,4);

    //generando los datos, su suma y suma de cuadrados
    while (cont < 33.0)
    {   suma = suma + x;
      sumacua = sumacua + x*x;
      cont = cont + 1.0;
      x = x + del;
    }

        //calculo de la media, varianza y desviacion estandar
    media = suma/cont;
    var = sumacua/cont - media*media;
    desvstd = sqrt(var);
    cout<<«Media : «<<media<<endl;
    cout<<«Var : «<<var<<endl;
    cout<<«Desv. std : «<<desvstd<<endl;

     system(«PAUSE»);
}
```

Figura 5: programa correspondiente a la carta N-S de la figura 1, excepto que los datos son generados dentro del mismo programa

La salida de este programa es:

Media:1
Var:0
Desv. Std:0

La media es correcta pero la varianza y la desviación estándar obviamente no lo son. Si volvemos a correr el programa haciendo que las variables sean de doble precisión (**double**) y no de precisión simple como los **float**, se tienen los siguientes resultados:

Media:1
Var: 2.111e-08
Desv. Std: 0.000145293

Los resultados ahora son extremadamente precisos.

Se han escogido potencias de 1/16 en los cálculos debido a que tales números pueden ser representados como términos fraccionales en una computadora hexadecimal como son las IBM PC y compatibles. Los errores en los resultados en la precisión simple se deben a la representación de los datos en la computadora y a los cálculos en sí mismos.

Para analizar esta dificultad supóngase una computadora decimal hipotética donde se usan potencias de 1/10 para evitar los errores inherentes a la data. Asúmase que tenemos 4 dígitos en la mantisa y se usan como datos los 21 números siguientes: .990, .991, ... ,999,1.0, 1.001, 1.002, ... ,1.010; de este modo $\Delta X = 1/10^3$. El número más pequeño es $1 - 1/10^2$ y el más grande es $1 + 1/10^2$. La media es 1 y la varianza es 0.36666×10^{-5}. Los números y sus sumas parciales en punto flotante son

X	Sumas Parciales
0.990	0.9900
0.991	1.9810
0.992	2.9730
0.993	3.9660
0.994	4.9600
0.995	5.9550
0.996	6.9510
0.997	7.9480
0.998	8.9460
0.999	9.9450
1.000	10.9450
1.001	11.9460
1.002	12.9480
1.003	13.9510
1.004	14.9550
1.005	15.9600
1.006	16.9660
1.007	17.9730
1.008	18.9810
1.009	19.9900
1.010	21.0000

Como puede verse la data se ha redondeado la data simétricamente. La suma de los 21 datos tal como se muestra es 21.00 y la media es 1.

Ahora sumemos los cuadrados de X_i, siendo los resultados:

X	X^2	Sumas Parciales
0.990	0.9801	0.9801
0.991	0.9821	1.9622
0.992	0.9841	2.9462
0.993	0.9860	3.9323
0.994	0.9880	4.9203
0.995	0.9900	5.9104
0.996	0.9920	6.9024
0.997	0.9940	7.8964
0.998	0.9960	8.8924
0.999	0.9980	9.8904
1.000	1.0000	10.8904
1.001	1.0020	11.8924
1.002	1.0040	12.8964
1.003	1.0060	13.9024
1.004	1.0080	14.9104
1.005	1.0100	15.9204
1.006	1.0120	16.9325
1.007	1.0140	17.9465
1.008	1.0161	18.9626
1.009	1.0181	19.9807
1.010	1.0201	21.0008

El último número en la última columna indica la suma de los cuadrados de los datos. Usando la ecuación (4), se tiene

$$v = \frac{21.00}{21} - (1.0)^2 = 0$$

Es decir; obtenemos una varianza de 0, tal como lo halla nuestro programa de computadora. Esto nos lleva a creer que si queremos analizar la dificultad en nuestra computadora hipotética debemos mirar por dentro la dificultad en una computadora real.

Demos una mirada a un dato particular; por ejemplo, el dato 0.996, siendo su cuadrado 0.992016; aunque en nuestra mantisa de cuatro dígitos este valor es representado como 0.9920. El error absoluto de redondeo es 0.000016 es decir 0.16×10^{-4}

Suponga que ahora usamos la ecuación (2) para calcular la varianza en vez de la ecuación (4), entonces antes de elevar al cuadrado el número 0.996, debemos restar la media. Ahora calculamos $(0.996 - 1.0)^2 = 0.000016 = 0.16 \times 10^{-4}$. Este es el último

número que se añadió a la suma en (2). Nótese que este es exactamente igual que el error de redondeo en él calculo del cuadrado de 0.996.

Veamos otro dato. Consideremos 1.007, su cuadrado es 1.014049. En nuestra computadora de cuatro dígitos es representado como 1.0140, así que el error de redondeo es $0.000049 = 0.49 \times 10^{-4}$. Si restamos la media, antes de elevar al cuadrado tenemos $(1.007 - 1.0)^2 = 0.000049 = 0.49 \times 10^{-4}$, el cual es nuevamente el error de redondeo en el cuadrado de 1.007.

Estos ejemplos nos llevan a suponer que si usamos (2), podremos conseguir resultados más precisos. El cálculo de la varianza usando (2) en nuestra computadora de 4 dígitos es como sigue:

X	$X - \mu$	$(X - \mu)^2$	Sumas Parciales
0.990	-0.0100	0.1000×10^{-3}	0.1000×10^{-3}
0.991	-0.0090	0.8100×10^{-4}	0.1810×10^{-3}
0.992	-0.0080	0.6400×10^{-4}	0.2450×10^{-3}
0.993	-0.0070	0.4900×10^{-4}	0.2940×10^{-3}
0.994	-0.0060	0.3600×10^{-4}	0.3300×10^{-3}
0.995	-0.0050	0.2500×10^{-4}	0.3550×10^{-3}
0.996	-0.0040	0.1600×10^{-4}	0.3710×10^{-3}
0.997	-0.0030	0.9000×10^{-5}	0.3800×10^{-3}
0.998	-0.0020	0.4000×10^{-5}	0.3840×10^{-3}
0.999	-0.0010	0.1000×10^{-5}	0.3850×10^{-3}
1.000	0.0000	0.0000×10^{-0}	0.3850×10^{-3}
1.001	0.0010	0.1000×10^{-5}	0.3860×10^{-3}
1.002	0.0020	0.4000×10^{-5}	0.3900×10^{-3}
1.003	0.0030	0.9000×10^{-5}	0.3990×10^{-3}
1.004	0.0040	0.1600×10^{-4}	0.4150×10^{-3}
1.005	0.0050	0.2500×10^{-4}	0.4400×10^{-3}
1.006	0.0060	0.3600×10^{-4}	0.4760×10^{-3}
1.007	0.0070	0.4900×10^{-4}	0.5250×10^{-3}
1.008	0.0080	0.6400×10^{-4}	0.5890×10^{-3}
1.009	0.0090	0.8100×10^{-4}	0.6700×10^{-3}
1.010	0.0100	0.1000×10^{-3}	0.7700×10^{-3}

Para calcular la varianza usamos (2) y obtenemos

$$\nu = \frac{0.000770}{21} = 0.00003667 = .3667x10^{-5}$$

Este es el resultado correcto redondeado simétricamente, para cuatro dígitos significativos.

En este caso (2) produce una mayor *acuracidad* que los resultados encontrados con (4).

CONCLUSIONES

Dos algoritmos que aparentemente conducirán al mismo resultado no siempre actuarán así. La ecuación (2) es la que tiene mayor *acuracidad* que la ecuación (4).

Con frecuencia el algoritmo que requiere menos operaciones aritméticas y menos espacio en memoria es el que tiene una menor *acuracidad*. Es decir es el *costo* entre el calculo realizado y la *acuracidad* del computo.

La representación de datos en las computadoras depende del número de bits utilizados. A mayor número de bits, mayor capacidad de representación. Así, se tiene para los **float** = 32 bit, los **double** = 64 bit y los **long double** = 80 bit.

BIBLIOGRAFÍA

1. **Chapra Seven C.; Canale Raymond P. 1999.** Métodos Numéricos para Ingenieros. 3ra. Ed., Mcgraw Hill Interamericana Editores, S.A. de C.V., México
2. **Mathews John. H.; Fink Kurtis. D. 2000.** Métodos Numéricos con MATLAB. 1ra. Ed., Editorial Prentice Hall Iberia S.R.L., España.