University of Glasgow | School of Computing Science

CSC1103 Tutorial 4 5 : Control Structures and Functions

1. GAUSSIAN PROBABILITY DENSITY FUNCTION

1. Problem definition:

Write a C program to take in the input parameters of the mean μ and standard deviation σ of a gaussian distributed variable x that range from -20 to 20 in the increment of 0.5. Print out a table of the probability density function with respect to the input.

2. Problem Analysis

The normal/gaussian probability density function is given as

$$f(x) = \frac{1}{\sqrt[2]{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

with the given input μ , σ and x . The data requirement to compute $f(x)$ is as follow

Input variable

i. The mean (μ)of the normal variable x, mu (float mu)
ii. The standard deviation (σ) of the normal variable x, sigma (float sigma)
iii. The normal variable x , x (float x)
iv. The minimum value of the normal variable x, xmin (float xmin)
v. The maximum value of the normal variable x, xmax (float xmax)
vi. The step size increment of the normal variable x , delta (float delta)

Output variable

i. The probability density function $f(x)$ for different value of x, fx (double fx)

3. Algorithm

1. Set mu=1
2. Set sigma=2
3. Set delta =0.5
4. Set xmin= -20
5. Set xmax=20
6. Set x=xmin
7. While (x<=xmax) do the following

7.1 Compute $fx = \frac{1}{\sqrt[2]{2\pi sigma^2}} e^{-\frac{1}{2}\left(\frac{x-mu}{sigma}\right)^2}$
7.2 Increment x by delta, x=x+delta
7.3 Print the value of x, fx

<u>Pseudocode</u>

BEGIN
$mu \leftarrow 1$
$sigma \leftarrow 2$
$delta \leftarrow 0.5$
$xmin \leftarrow -20$
$xmax \leftarrow 20$
$x \leftarrow xmin$
WHILE $(x \leq xmax)$
$$f(x) \longleftarrow \frac{1}{\sqrt[2]{2\pi sigma^2}} e^{-\frac{1}{2}\left(\frac{x-mu}{sigma}\right)^2}$$
PRINT $x, f(x)$
$x \leftarrow x + delta$
END WHILE
END

2. EXPONENTIAL FUNCTION

1. Problem definition:

Write a C program to take in the input parameters x and evaluate the exponential function e^x using infinite series. Print out the sum and the increment to the sum for each term.

2. Problem Analysis

The exponential function using infinite series is given as

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots \frac{x^n}{n!}$$

The n^{th} term in the series can be obtained by multiplying the $(n-1)^{th}$ term by (x/n) given rise to

$$term_n = (term_{n-1})\left(\frac{x}{n}\right)$$

Where $n \geq 1$

when $n = 1$, $term_1 = (term_o)\left(\frac{x}{1}\right)$ $where\ (term_o) = 1$

when $n = 2$, $term_2 = (term_1)\left(\frac{x}{2}\right)$

when $n = 3$, $term_3 = (term_2)\left(\frac{x}{3}\right)$

Assumption: The infinite series can only take in positive number of x. If x is negative, need to change to positive x and perform infinite series first and then do an inverse of the final answer.

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots \frac{x^n}{n!} \qquad if\ x \geq 0$$

$$e^{-x} = 1/\, e^x \qquad if\ x < 0$$

Computation stops when the $term_n < 1 \times 10^{-6}$

<u>Input variable</u>

i. The x (double x)

Process variable

i. The limit of computation, limit (double `limit`)
ii. The x value regardless input x is positive or negative ,`absx` (double `absx`)
iii. The power limit, n (int n)

Output variable

i. Each computed term that required for summation, `term` (double `term`)
ii. Sum of each term for every round of computation, `sum` (double `sum`)

3. Algorithm

1. Read the value of x
2. Set $limit = 1 \times 10^{-6}$
3. Set $term = 1$
4. Set $sum = 1$
5. Set $n = 1$
6. If (x <0)
   6.1 $absx = -x$
   Else
   6.2 $absx = x$
7. Do the following
   7.1 $term = term * absx/n$
   7.2 $sum = sum + term$
   7.3 $n = n + 1$
   7.4 Print the value of sum and $term$
   while ($term \geq limit$)
8. If (x <0)
   8.1 $sum = 1/sum$
   Else
   8.2 $sum = sum$
9. Print sum

Pseudocode

```
BEGIN
        READ x
        limit ← 1 × 10^-6
        term, sum, n ← 1
        IF x < 0
                absx ← -x
        ELSE
                absx ← x
        ENDIF
        DO
                term ← term * absx/n
                sum ← sum + term
                n ← n + 1
                PRINT "sum", sum
                PRINT "term", term
        WHILE ( term ≥ limit)
        END WHILE
        IF x < 0
                sum ← 1/sum
        ELSE
                sum ← sum
        ENDIF
        PRINT "sum", sum
END
```

3. BINOMIAL THEORM

1. Problem definition:

Write a C program that takes in x, y, n and return the $(x+y)^n$ using series expansion and recursive function.

2. Problem Analysis

Using series expansion,

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k = \binom{n}{0} x^n + \binom{n}{1} x^{n-1} y^1 + \binom{n}{2} x^{n-2} y^2 + \cdots + \binom{n}{n} y^n$$

where

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n\times(n-1)\times(n-2)\cdots(n-k+1)}{1\times 2\cdots k}$$

Input variable

i. The input x, y and n (double x , double y and int n)

Process variable

i. The iterative variable, k (int k)

Output variable

ii. the result of series expansion of $(x+y)^n$, `result` (double, `result`)

3. Algorithm

The program is divided into following functions

a) **main ()** :
   i. to obtain user input on x, y, n
   ii. call the function **x_plus_y()** to compute the series expansion
   iii. print the result

b) **x_plus_y**():
   i. compute the series expansion by summing the series $\binom{n}{k} x^{n-k} y^k$ for $k = 0\ to\ n$ and return the sum back to **main()**
   ii. call the function **binom()** to compute the binomial coefficient $\binom{n}{k}$
   iii. call the function **power()** to compute the value of x^{n-k} and y^k

c) **binom()**
   i. compute the $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ and return the result back to **x_plus_y()**
   ii. call the recursive function **factorial ()** to compute the $n!$, $k!$ and $(n-k)!$

d) **factorial()**
   i. compute the factorial and return the factorial result.

e) **power()**
   i. compute the value of x and y raised to the power of $n-k$ and k respectively.

Algorithm for **main()**
1. Read the value of x
2. Read the value of y
3. Read the value of n
4. result= **x_plus_y(**x, y, n**)** to compute the series expansion
5. Print the `result`

Algorithm for **x_plus_y()**
1. Set $sum = 0$
2. For $k = 0\ to\ n$ do the following
   2.1 $sum = sum + \mathbf{binom}(n, k) * \mathbf{power}(x, n-k) * \mathbf{power}(y, k)$
3. Return the sum

Algorithm for **binom()**
1. Set $result = 1$
2. $n\_fact = factorial(n)$
3. $k\_fact = factorial(k)$
4. $n\_minus\_k\_fact = factorial(n-k)$
5. $result = n\_fact/(k\_fact \times n\_minus\_k\_fact)$
6. return $result$

Algorithm for **factorial()**

1. If m =0

   1.1 return 1

   Else

   1.1 return ($m * \text{factorial}\ (m - 1)$)

Algorithm for **power()**

1. Set $result = 1$
2. If $n = 0$

   2.1 return $result$
3. For $i = 1\ to\ abs(n\ )$ do the following

   3.1 $result = result * x$
4. If $n \geq 0$

   4.1 return $result$

   Else

   4.1 return 1/$result$

Pseudocode

BEGIN
    READ x, y, n
    $result \leftarrow \text{x\_plus\_y}(x, y, n)$
    PRINT "result", $result$
END

FUNCTION x_plus_y()
    $sum \leftarrow 0$
    FOR $k = 0\ to\ n$ do
        $sum \leftarrow sum + \text{binom}(n, k) * \text{power}(x, n - k) * \text{power}(y, k)$
    END FOR
    return sum
ENDFUNCTION

FUNCTION binom()
    $result \leftarrow 1$
    $n\_fact \leftarrow factorial(n)$
    $k\_fact \leftarrow factorial(k)$
    $n\_minus\_k\_fact \leftarrow factorial(n - k)$
    $result \leftarrow n\_fact/(k\_fact \times n\_minus\_k\_fact)$
    return $result$
ENDFUNCTION

```
FUNCTION factorial()
	IF (m=0)
		return 1
	ELSE
		return m*factorial(m−1)
	ENDIF
ENDFUNCTION

FUNCTION power()
	result ← 1
	IF (n = 0)
		return 1
	ENDIF
	FOR i = 1 to abs(n) do
		result ← result * x
	END FOR
	IF (n ≥ 0)
		return result
	ELSE
		return 1/result
	ENDIF
ENDFUNCTION
```

4. **Machine Learning: Linear Classification**

1. Problem definition:

Write a C program to train the weight and bias of the linear classifier to produce output y of the AND operation according to the input x_1 and x_2 and the truth table given. The user needs to key in different value of combination of input x_1 and x_2 to demonstrate the output y is achieved according to the truth table.

2. Problem Analysis

Based on the linear classifier formulae given,

For each case j where $j = 1 \cdots 4$, the estimated output

$$\hat{y}_j = \sum_{i=1}^{2} w_i x_i + b = w_1 x_1 + w_2 x_2 + b$$

where w_1 is the weight for x_1 and w_2 is the weight for x_2. b is bias of the linear classifier. The objective is to train the weights and bias such the desired output for each case j is achieved according to the truth table of AND operation as below.

Case j	x_1	x_2	AND operation (y_j)
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

This means that the aim is to achieve the error in output estimation in each case, $\mathcal{E}_j$ and total error for the four cases $\mathcal{E}$ to be zero or minimum as possible

$$\text{error in output estimation in each case } j, \mathcal{E}_j = \hat{y}_j - y_j = \left(\sum_{i=1}^{2} w_i\, x_{ji} + b\right) - y_j$$

$$\text{Total error in output estimation, } \mathcal{E} = \sum_{j=1}^{4} \mathcal{E}_j = \sum_{j=1}^{4} \hat{y}_j - y_j$$

In this case, error threshold is set to the zero where $\mathcal{E}_{threshold} = 0$ and $\mathcal{E} = \mathcal{E}_{threshold} = 0$. For each update or iteration t,new weight and bias for next iteration are updated as

$$w_1{}^{t+1} = w_1{}^t - \alpha \sum_{j=1}^{4} \mathcal{E}_j x_{j1}$$
$$w_2{}^{t+1} = w_2{}^t - \alpha \sum_{j=1}^{4} \mathcal{E}_j x_{j2}$$
$$b^{t+1} = b^t - \alpha\, \mathcal{E}$$

where α is the training speed. Set $\alpha = 0.5$ and initial weight w_1, w_2 and b =0

Input variable

i. The input x_1, and x_2 (int x_1 ,int x_2)
ii. The true output label, y (int y)

Process variable

i. The training speed, $alpha$ (float $alpha$)
ii. The error tolerance , $errortol$ (float $errortol$)
iii. The error due to input x_1, $errorx1$ (float $errorx1$)
iv. The error due to input x_1, $errorx2$ (float $errorx2$)
v. The estimated output, $yest$ (float $yest$)
vi. Number of iteration, $iter$ (int $iter$)
vii. The case number, row (int row)

Output variable

i. The weight for x_1, w_1 (float w_1)
ii. The weight for $x_{2\,,}$ w_2 (float w_2)
iii. The bias, b (float b)
iv. The total error for training output y, $errory$ (float $errory$)

3. Algorithm

The program is divided into following functions

a) **main ()** :
   i. call the function **selectdata ()** to select each training case j with correct input x_1, x_2 and true output label y_j
   ii. call the function **estimationerror ()** to compute error on output and due to input for each case j
   iii. call the function **parameterupdate ()** to compute new weights and bias for each iteration, $iter$

iv. print the total training iteration needed, finalized weights and bias
v. prompt the user to inputs x_1, x_2 and print the predicted output

b) **selectdata ()**:
i. Based on selected case j, output the training value of input x_1, x_2 and true output label y for training

c) **estimationerror ()**
i. Compute estimated output $yest = w_1 * x_1 + w_2 * x_2 + b$
ii. Update error in output $errory$ and error due to each input $errorx1$, $errorx2$

d) **parameterupdate ()**
i. Print the iteration run, $iter$ and its output error $errory$
ii. Update the weight and bias

Algorithm for **main()**
1. Set the $iter = 1$
2. Set the $row = 1$
3. Do the following
   3.1 Set $errory = 0$
   3.2 Set $errorx1 = 0$
   3.3 Set $errorx2 = 0$
   3.4 For $row = 1$ to 4 do the following
      3.4.1 **selectdata** (row)
      3.4.2 **estimationerror (**x_1, x_2, y**)**
   3.5 **parameterupdate** $(iter)$
   3.6 $iter = iter + 1$
   while $(errory > errortol)$
4. Print $iter, errory, w_1, w_2, b$
6. Read the value of x_1
7. Read the value of x_2
8. Print compute the $predicted\ output = w_1 * x_1 + w_2 * x_2 + b$

Algorithm for **selectdata ()**

1. If $(row = 1)$
   1.1 $x_1 = 0$
   1.2 $x_2 = 0$
   1.3 $y = 0$
2. If $(row = 2)$

2.1 $x_1 = 0$
2.2 $x_2 = 1$
2.3 $y = 0$

3. If $(row = 3)$
    3.1 $x_1 = 1$
    3.2 $x_2 = 0$
    3.3 $y = 0$
4. If $(row = 4)$
    4.1 $x_1 = 1$
    4.2 $x_2 = 1$
    4.3 $y = 1$

<u>Algorithm for **estimationerror()**</u>

1. $yest = w_1 * x_1 + w_2 * x_2 + b$
2. If $yest > 0$
    2.1. $yest = 1$

   Else
    2.2. $yest = 0$
3. $errory = errory + yest - y$
4. $errorx1 = errorx1 + (yest - y) * x_1$
5. $errorx2 = errorx2 + (yest - y) * x_2$

<u>Algorithm for **parameterupdate()**</u>

1. Print $iter, errory$
2. $w_1 = w_1 - alpha * errorx1$
3. $w_2 = w_2 - alpha * errorx2$
4. $b = b - alpha * errory$

Pseudocode

```
BEGIN
    iter ← 1
    row ← 1
    DO
        errory ← 0
        errorx1 ← 0
        errorx2 ← 0
        FOR row = 1 to 4 do
            selectdata(row)
            estimationerror(x1, x2, y)
        END FOR
        parameterupdate(iter)
        iter ← iter + 1
    WHILE ( abs(errory) > errortol)
    END WHILE
    PRINT "iter", iter
    PRINT "errory", errory
    PRINT "w1", w1
    PRINT "w2", w2
    PRINT  "b",    b
    READ x1, x2,
    PRINT "predicted output", w1 * x1 + w2 * x2 + b
END

FUNCTION selectdata ()
    IF (row = 1)
        x1 ← 0
        x2 ← 0
        y ← 0
    ENDIF
    IF (row = 2)
        x1 ← 0
        x2 ← 1
        y ← 0
    ENDIF
    IF (row = 3)
        x1 ← 1
        x2 ← 0
```

$y \leftarrow 0$
ENDIF
IF $(row = 4)$
$x_1 \leftarrow 1$
$x_2 \leftarrow 1$
$y \leftarrow 1$
ENDIF
ENDFUNCTION

FUNCTION estimationerror ()
$yest \leftarrow w_1 * x_1 + w_2 * x_2 + b$
IF $(yest > 0)$
$yest \leftarrow 1$
ELSE
$yest \leftarrow 0$
ENDIF
$errory \leftarrow errory + yest - y$
$errorx1 \leftarrow errorx1 + (yest - y) * x_1$
$errorx2 \leftarrow errorx2 + (yest - y) * x_2$
ENDFUNCTION

FUNCTION parameterupdate ()
PRINT “iter”, $iter$
PRINT “errory”, $errory$
$w_1 \leftarrow w_1 - alpha * errorx_1$
$w_2 \leftarrow w_2 - alpha * errorx_2$
$b \leftarrow b - alpha * errory$
ENDFUNCTION