ASSIGNMENT 3

Statistisk problemlösning med Excel och VBA

Oscar Lindquist
Elias Charalambidis
Teachers:
Leif Nilsson
May 21, 2024

Contents

1 Task 1

1.1 Problembeskrivning

Eftersom algoritmerna är beroende av matrisberäkningar så är det viktigt att veta hur man multiplicerar matriser effektivt i VBA. För att testa detta ska du börja med att skriva ett VBA-program för att jämföra tidsåtgången mellan två olika metoder för matrismultiplikation i VBA:

- Excels MMULT-funktion
- en egen funktion för matrismultiplikation

Gör denna jämförelse i en egen arbetsbok på följande sätt:

- Skapa två slumpmatriser av storlek $50 * 50$, med element som är likformigt fördelade i intervallet $[0, 100)$, i två separata flikar
- Läs in dessa två matriser till ett VBA-program som multiplicerar dem med varandra 10000 gånger, detta ska göras med bägge metoderna enligt ovan

1.2 Metod

Steg ett var att skapa två matriser med värden mellan 0-100 som har storlek 50X50 med hjälp av funktionen slump.mellan i excel. Därefter genomfördes ett vba-program (mmultmatris) med en for-loop som loopade 1000 ggr och genomförde matris multiplikation på dessa matriser med hjälp av MMult funktionen. När loopen var klar, så printades tiden som det tog att köra programmet.

Andra VBA-programmet (myMult) läser in matriserna och genomför matris multiplikationen förhand genom att ta rad gånger kolumn i flertalet for-loopar. När matris multiplikationen har genomförts 1000 gånger, så printas tiden som det tog att genomföra programmet.

1.3 Resultat

Resultatet presenteras som en tabell med våra program samt tiden att genomföra matrismultiplikationerna.

Funktion	Tid
MMult	6,668
Egen funktion	134,438

Table 1: Tid för våra olika funktioner

1.4 Slutsats

Vi ser en tydlig skillnad på resultatet. Den inbyggda matris multiplikations (MMULT) funktionen är mycket snabbare än den egenbyggda där vi gör matris multiplikationen förhand. Inför framtida uppgifter, så kommer därför MMult funktionen att användas för att effektivisera våra VBA-program.

2 Task 2

2.1 Problembeskrivning

I denna uppgift ska du med logistisk regression träna upp en modell som kan användas för att klassificera T-shirt (0) och skjorta (6). Modellen ska sedan utvärderas på ett testdata.

Datamaterialet `fashion_mnist_06` innehåller två flikar, en med träningsdata och en med testdata (testdata ska bara användas för att validera den modell du tränat upp med träningsdata).

Träningsdata och testdata består av 4000 respektive 2000 klädesplagg, där hälften är klassificerad som 0 (T-shirt) och hälften som 6 (skjorta).

Första kolumnen är märkt `label` och anger klassificering. Resterande 784 kolumner märkta pixel1, pixel2, ...,pixel784 anger pixelvärdena.

1. Börja med att dimensionera ned antalet feature-vektor (pixelkolumner) till 61 genom att bara ta med var trettonde med start med `pixel8` (`pixel8`, `pixel21`, ..., `pixel775`).
2. Standardisera pixelvärdena (dividera dem med 256) så att de ligger i intervallet $(0,1)$.
3. Träna sedan en logistisk regressionsmodell på det neddimensionerade träningsdata med hjälp av gradient descent med 1000 iterationer och en learning rate på 0.9.
4. Välj noll-vektorn som start-vektor.

a)

Skriv ut värdena på din kostnadsfunktion för var femtionde iteration (inklusive det första värdet) i en tabell som du lägger in i rapporten. Plotta sedan kostnadsfunktionen i ett linjediagram och infoga i rapporten.

b)

Sammanställ en tabell (confusion matrix) som anger hur många som klassificerats rätt och fel för respektive siffra. Beräkna andelen korrekt klassificerade (accuracy) angivet i procent. Gör detta för både tränings- och testdata.

2.2 Metod

2.2.1 a

- Läs in träningsdata i Excel med labels som en egen vektor
- Filtrerar data och plockar vart trettonde pixel samt dividerar med 256. Detta för att få en procentsats.
- Applicerar specificerade algoritmen på nya filtrerade datasetet
- Skapa nollvektor Θ_0
- Bestäm $Z_0 = X\Theta_0$
- Bestäm $P_0 = g(Z_0)$ (elementpris)
- Bestäm $\theta_1 = \Theta_0 - \frac{\alpha}{m} * X^T(P-Y)$

- Uppdatera för $n = 1, ..., 1000$:
- $Z_n = X\Theta_n$
- $P_n = g(Z_n)$ (elementvis)
- $\Theta_{n+1} = \Theta_n - \frac{\alpha}{m} * X^T(P_n - Y)$
- Kostnadsfunktionen beräknas enligt

$$L(p,y) = -y(ln(p)) - (1-y)ln(1-p)$$

När algoritmen har körts så får vi en tabell med varje 50 iteration samt kostnadsfunktions värdet, dessa presenteras i tabell samt i en graf.

2.2.2 b

Genomför en confusion matris på träningsdatat samt test datat där testdatat genomför en logistisk regression från nuvarande Theta för att få senaste iterationen. Sen genomförs en accuracy beräkning där vi tar totalt antal rätt genom totala.

2.3 Resultat

Resultatet av vår logistiska regression presenteras i Tabell 1 och 2 som confusion matriser nedan. Figure 1 visar kostnadsfunktionen för varje 50 iterationer som även presenteras i värden på figur 2.

Metod	0	1	
0	1752	248	
1	412	1588	
accuracy(%)			83.5

Table 2: confusion-matris för träningsdata

Metod	0	1	
0	886	114	
1	192	808	
accuracy(%)			83.5

Table 3: confusion-matris för testdata

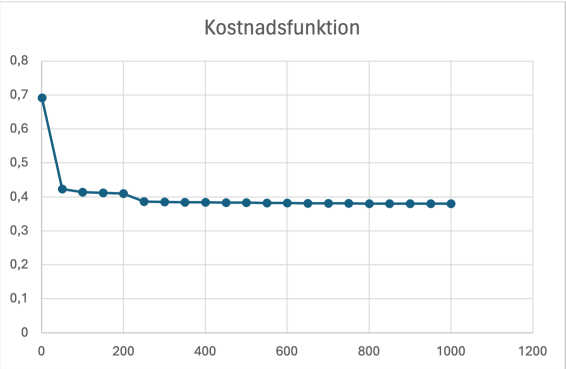

Figure 1: Kostnadsfunkion över antalet iterationer

iter	kostnadsfunktion
1	0,692100856
50	0,423069933
100	0,413964118
150	0,412353454
200	0,410184765
250	0,386537475
300	0,385639881
350	0,384551265
400	0,38378528
450	0,383144153
500	0,38289964
550	0,382130394
600	0,381745676
650	0,381365559
700	0,381057681
750	0,380768716
800	0,380517406
850	0,380275644
900	0,380081233
950	0,379988889
1000	0,379733601

Figure 2: Tabell för antal iterationer och motsvarande kostnadsfunktioner

2.4 Slutsats

Baserat på den data som vi fått tillgänglig, och efter den filtrering som krävdes enligt specifikation, får vår modell en accuracy på 83.5% på träningsdata, samt 84.7% på testdata. Detta anser vi är ett godtyckligt resultat, även det uppnår inte 95% vilket är en standardkvot. Detta då vi har byggt upp en relativt enkel modell med utrymme för förbättring. Exempelvis kunna hantera fler pixlar i beräkningarna. Accuracy:n beräknades för hand i excel. Baserat på Figure 1 kan man även dra slutsatsen att kostnadsfunktionen avtar snabbt när antalet iterationer är låga och konvergerar till ca 0.38 (se Figure 2) då antalet iterationer överstiger 200.

3 VBA-kod

3.1 Task 1

```
Sub mmultmatris()
    Dim matrix1 As Variant
    Dim matrix2 As Variant
    Dim matrix3 As Variant
    Dim seconds_elapsed As Double
    Dim start_time As Double

    start_time = Timer
    matrix1 = Sheets("Mat1").Range("A1:AX50").value
    matrix2 = Sheets("Mat2").Range("A1:AX50").value
    matrix3 = Application.WorksheetFunction.MMult(matrix1, matrix2)

    For i = 1 To 9999
        matrix3 = Application.WorksheetFunction.MMult(matrix1, matrix2)
    Next i

    Sheets("UPG1").Range("A1:AX50").value = matrix3
    seconds_elapsed = Timer - start_time
    Sheets("UPG1").Range("A52").value = seconds_elapsed
End Sub

Sub myMult()
    Dim matrix1 As Variant
    Dim matrix2 As Variant
    Dim matrix3 As Variant
    Dim seconds_elapsed As Double
    Dim start_time As Double
    Dim value As Double

    ReDim matrix3(1 To 50, 1 To 50)

    start_time = Timer
    matrix1 = Sheets("Mat1").Range("A1:AX50").value
    matrix2 = Sheets("Mat2").Range("A1:AX50").value
    For l = 1 To 10000
        For i = 1 To 50
            For h = 1 To 50
                value = 0
                For j = 1 To 50
                    value = value + matrix1(i, j) * matrix2(j, h)
                Next j
                matrix3(i, h) = value
            Next h
        Next i
    Next l
```

```
    Sheets("UPG1").Range("A55:AX104").value = matrix3
    seconds_elapsed = Timer - start_time
    Sheets("UPG1").Range("C52").value = seconds_elapsed
End Sub
```

3.2 Task 2

```
Sub LOGREG()
    Dim start_matrix As Variant
    Dim x(1 To 4000, 1 To 61) As Variant
    Dim y(1 To 4000, 1 To 1) As Variant
    Dim Teta(1 To 61, 1 To 1) As Variant
    Dim Teta_latest(1 To 61, 1 To 1) As Variant
    Dim Z As Variant
    Dim P(1 To 4000, 1 To 1) As Variant
    Dim PY(1 To 4000, 1 To 1) As Variant
    Dim F As Long 'eventuellt byta till variant
    Dim L(1 To 21, 1 To 1) As Variant
    Dim cost As Integer

    cost = 1
    start_matrix = Range("train").value

    For i = 1 To 4000
        x(i, 1) = 1
        y(i, 1) = start_matrix(i, 1)
        For j = 0 To 59
            x(i, j + 1) = start_matrix(i, 9 + 13 * j) / 256
        Next j
    Next i

    For i = 1 To 61
        Teta(i, 1) = 0
    Next i

    Z = WorksheetFunction.MMult(x, Teta)

    For i = 1 To 4000
        P(i, 1) = 1 / (1 + Exp(-Z(i, 1)))
        PY(i, 1) = P(i, 1) - y(i, 1)
    Next i

    Transpose_X = WorksheetFunction.Transpose(x)

    For i = 1 To 1000
        XPY = WorksheetFunction.MMult(Transpose_X, PY)
        For j = 1 To 61
            Teta_latest(j, 1) = Teta(j, 1) - (0.9 / 4000 * XPY)
        Next j
```

```
    Z = WorksheetFunction.MMult(x, Teta_latest)
    For j = 1 To 4000
        P(j, 1) = 1 / (1 + Exp(-Z(i, 1)))
    Next j

    F = 0
    If i = 1 Or i Mod 50 = 0 Then
        For j = 1 To 4000
            F = F - (y(i, 1) * Log(P(i, 1)) + (1 - y(i, 1)) * Log(1 - P(i, 1)))
        Next j
        L(cost, 1) = F / 4000
        cost = cost + 1
    End If

Next i
For i = 1 To 21
    Worksheets("UPG2").Cells(1 + i, 1).value = L(i, 1)
Next i

'b
T1 = 0
F1 = 0
T2 = 0
F2 = 0

For i = 1 To 4000
    If P(i, 1) < 0.5 Then
        If y(i, 1) = 0 Then
            T1 = T1 + 1
        Else
            F1 = F1 + 1
        End If

    Else
        If y(i, 1) = 1 Then
            T2 = T2 + 1
        Else
            F2 = F2 + 1
        End If
    End If
Next i

Range("F1") = T1
Range("F2") = F2
Range("G1") = F1
Range("G2") = T2


Dim x2(1 To 2000, 1 To 61) As Variant
```

```
Dim y2(1 To 2000, 1 To 1) As Variant
Dim P2(1 To 2000, 1 To 1) As Variant
Dim Z2 As Variant
Dim Test_matrix As Variant


Test_matrix = Range("TEST").value

For i = 1 To 2000
    x2(i, 1) = 1
    y2(i, 1) = Test_matrix(i, 1)
    For j = 0 To 59
        x2(i, j + 1) = Test_matrix(i, 9 + 13 * j) / 256
    Next j
Next i

Z2 = WorksheetFunction.MMult(x2, Teta_latest)

For i = 1 To 2000
    P2(i, 1) = 1 / (1 + Exp(-Z2(i, 1)))
Next i

T1 = 0
F1 = 0
T2 = 0
F2 = 0

For i = 1 To 2000
    If P2(i, 1) < 0.5 Then
        If y(i, 1) = 0 Then
            T1 = T1 + 1
        Else
            F1 = F1 + 1
        End If

    Else
        If y(i, 1) = 1 Then
            T2 = T2 + 1
        Else
            F2 = F2 + 1
        End If
    End If
Next i

Range("C1") = T1
Range("C2") = F2
Range("D1") = F1
Range("D2") = T2
```