Universidad Complutense de Madrid

IAAC - PRÁCTICA 3

Yaco Alejandro Santiago Pérez

Asignatura: INTELIGENCIA ARTIFICIAL APLICADA A INTERNET DE LAS COSAS

P3: Regresión logística multi-clase y redes neuronales

Master IOT

16 de marzo de 2020

Índice general

Capítulo 1

Introducción

Esta práctica trata de emplear la ***Regresión logística multi-clase*** sobre los datos que representan números escritos a mano con el objetivo de iniciarnos en el reconocimiento de imágenes.

Figure 1.1: *Selección aleatoria de 10 ejemplos de entrenamiento*

Para la segunda parte de la práctica se hará una primera toma de contacto con las **redes neuronales**.

Esta red está formada por **tres capas**, con 400 unidades en la primera capa (además de la primera fijada siempre a +1), 25 en la capa oculta y 10 en la capa de salida.

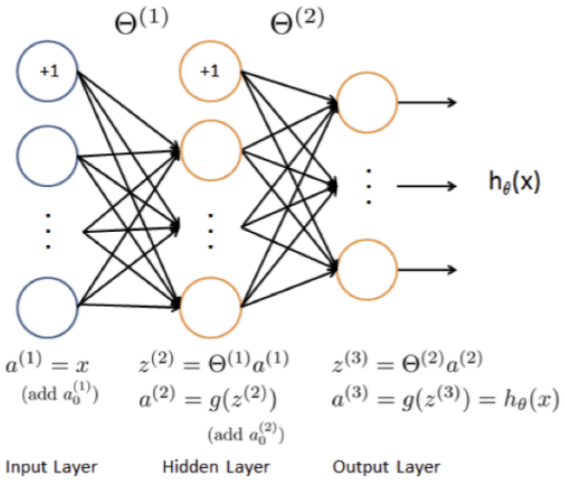


Figure 1.2: *Aspecto de la red neuronal*

Capítulo 2

Objetivos

En esta práctica, la cual se divide en dos partes, el objetivo es construir **modelos por regresión logística**.

- Emplear la regresión logística multi-clase
- Utilización de redes neuronales con distintos pesos

Para ello, deberemos alcanzar los siguientes objetivos más concretos:

- Obtener una imagen con 10 números aleatorios
- Vectorizar las funciones de coste y de gradiente en sus versiones normales y en las regularizadas.
- Calcular el coste
- Calcular el *gradiente*
- Calcular la *theta óptima*
- Desarrollar un método de clasificación de uno frente a todos
- Hacer una prueba sobre el modelo entrenado con datos aleatorios
- *Evaluar* el porcentaje de aciertos de la prueba
- Utilizar los pesos proporcionados para una red neuronal ya entrenada sobre ejemplos
- Evaluar su precisión sobre esos mismos ejemplos

Capítulo 3

Parte 1: Regresión logística multi-clase

3.1. Funciones

Las funciones empleadas, en orden de llamada, son las siguientes:

- **sigmoid(x):** Calcula el valor de la función **sigmoide**:

$$g(z) = \frac{1}{1 + e^{-z}}$$

- **cost(theta, X, Y):** Función de coste vectorizada sin regularización, como indica el enunciado, pero no se utiliza en la práctica.
- **gradient(theta, XX, Y):** Función de gradiente vectorizada sin regularización, como indica el enunciado, pero no se utiliza en la práctica.
- **cost_regul(theta, X, Y, lam):** Función de coste vectorizada con regularización. Que representa la siguiente fórmula.

$$J(\theta) = \left[\frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

- **gradient_regul(theta, XX, Y, lam):** unción de gradiente vectorizada con regularización. Que representa la siguiente fórmula.

$$\begin{aligned} \frac{\partial J(\theta)}{\partial \theta_0} &= \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} && \text{para } j = 0 \\ \frac{\partial J(\theta)}{\partial \theta_j} &= \left(\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}\right) + \frac{\lambda}{m}\theta_j && \text{para } j \geq 1 \end{aligned}$$

- **costeMinimo(XX, Y, lam):** Inicializa las thetas a 0 y realiza la llamada a *opt.fmin_tnc* para un *lamda* concreto. Para obtener el valor óptimo de *theta* para la versión regularizada de la función de coste.

- **oneVsAll(X, y, num_etiquetas, reg):** Función que devuelva una matriz donde cada fila de *theta* corresponde a los parámetros aprendidos para el clasificador de una de las clases, implementado en base a las indicaciones del guión.
- **clasificador(X, Thetas):** Clasifica de qué tipo es el dato evaluándolo con cada una de las *thetas* y devuelve el valor máximo del array.
- **exec(X, thetas,n):** Realiza una ejecución para un cierto número **n** de elementos de muestra y los evalúa uno a uno. Mostrando y guardando la imagen correspondiente e indicando en un texto a qué número se asemeja.
- **evaluaPorcentaje(X,Y,Theta):**Esta función evalúa la totalidad de la muestra y la compara con el resultado real, para obtener de esta manera el porcentaje de acierto.
- **main():** Función que hace todas las llamadas pertinentes.
  Carga los datos desde los archivos *.mat*.
  Como se indicaba al inicio del guión se coge una muestra aleatoria de diez números para visualizarlos por pantalla.

  Se obtienen las *thetas* llamando a la función **oneVsAll** y a continuación se llama a **exec** para realizar una ejecución sobre diez elementos aleatorios con una lamda de 0,1. Por ultimo, se evalúa el porcentaje de acierto con la llamad a la función **evaluaPorcentaje**.

3.2. Ejecución

En la consola se ven los valores obtenidos en la ejecución:

```
  NIT   NF   F                        GTG
    0    1  6.931471805599454E-01   2.72474253E+00
    1    6  7.898448817927257E-02   3.99787963E-03
    2   16  3.579320081118205E-02   7.19143620E-04
tnc: fscale = 37.29
    3   26  2.049484619438481E-02   5.23770150E-05
    4   34  1.509285602963753E-02   6.99123148E-06
    5   44  1.287137930822168E-02   8.48445345E-06
    6   54  1.215189676750747E-02   4.96691306E-07
tnc: fscale = 1418.92
    7   62  1.204047197633283E-02   4.38254079E-07
    8   71  1.192634285864231E-02   3.94903581E-08
    9   80  1.188588626009030E-02   9.84841993E-08
   10   89  1.187651376793313E-02   5.16321431E-08
   11   99  1.187267552847400E-02   7.44725635E-10
tnc: fscale = 36643.9
   12  107  1.187178163619536E-02   1.31249596E-09
   13  117  1.187111637706415E-02   2.27705362E-10
   14  127  1.187084801242321E-02   1.11458417E-10
   15  137  1.187069099379603E-02   1.02612112E-10
   16  147  1.187062075871555E-02   3.78510350E-11
   17  157  1.187058603831198E-02   1.12293772E-10
   18  167  1.187056635981914E-02   9.71990615E-12
tnc: |fn-fn-1] = 9.38645e-09 -> convergence
   19  176  1.187055697336479E-02   2.63054339E-11
tnc: Converged (|f_n-f_(n-1)| ~= 0)
  NIT   NF   F                        GTG
    0    1  6.931471805599454E-01   3.42944746E+00
    1    5  6.665873534124221E-02   5.19929005E-03
    2   14  3.750508051212356E-02   3.24407883E-04
tnc: fscale = 55.5206
    3   24  2.349515911017460E-02   3.04983643E-05
    4   27  2.173874414840160E-02   1.16103133E-05
tnc: stepmx = 1000
    5   46  1.604195157800859E-02   3.55445173E-05
    6   52  1.428695525655301E-02   1.38673401E-06
    7   55  1.412939565471885E-02   7.50262757E-07
tnc: fscale = 1154.5
    8   64  1.386294360793432E-02   9.37909151E-07
    9   74  1.373840016885895E-02   2.48172487E-08
   10   84  1.372104649017311E-02   2.90287205E-08
   11   93  1.371776918303841E-02   1.29302886E-08
```

```
   12  102  1.371579969523378E-02   1.46076052E-08
   13  111  1.371479937669860E-02   1.05258911E-09
tnc: fscale = 30822.7
   14  119  1.371439240174129E-02   6.28330082E-10
   15  131  1.371412309961251E-02   4.68591985E-11
   16  142  1.371405448218632E-02   7.24480525E-11
   17  158  1.371399310396701E-02   7.25500849E-11
   18  173  1.371397268389037E-02   3.29982371E-12
tnc: |fn-fn-1] = 9.18356e-09 -> convergence
   19  188  1.371396350033066E-02   1.33881082E-12
tnc: Converged (|f_n-f_(n-1)| ~= 0)
  NIT   NF   F                       GTG
    0    1  6.931471805599454E-01   2.82875449E+00
    1    5  2.407920405353954E-01   5.66459468E-02
    2   14  1.280451035783559E-01   8.63374404E-03
    3   17  1.137782028384112E-01   9.14732354E-04
tnc: fscale = 33.0638
    4   26  8.191033474397712E-02   1.46179904E-03
    5   30  7.403620082255173E-02   6.25616562E-05
    6   39  6.539266359824376E-02   3.42986617E-04
    7   49  5.969996974860015E-02   4.24575632E-05
    8   58  5.701640904902299E-02   3.85911660E-06
    9   68  5.584624147921290E-02   2.58591526E-06
   10   71  5.575705538041047E-02   2.07876329E-06
tnc: fscale = 693.581
   11   80  5.544045894901470E-02   9.09678891E-07
   12   90  5.507379765165393E-02   5.58213136E-07
   13   93  5.504092423866955E-02   1.02443494E-06
   14  102  5.483812673866345E-02   1.63750590E-06
   15  112  5.478934842981595E-02   6.22527886E-08
   16  121  5.477300747514784E-02   1.58732982E-07
   17  131  5.476070424168342E-02   2.30287129E-08
   18  141  5.475415525940604E-02   1.95965238E-08
   19  157  5.474700193881651E-02   1.34322225E-08
   20  166  5.474560219723280E-02   1.08771392E-08
   21  175  5.474415171136546E-02   5.04845721E-09
tnc: fscale = 14074.1
   22  189  5.474276699638923E-02   6.26753857E-09
   23  200  5.474241736600181E-02   4.79765408E-10
   24  210  5.474223144457163E-02   9.91231217E-10
   25  225  5.474201608329617E-02   1.34012409E-09
   26  234  5.474196800076315E-02   9.11857384E-10
   27  244  5.474193019328928E-02   7.82258447E-11
   28  261  5.474187286678250E-02   3.41445037E-11
   29  276  5.474185547675374E-02   4.33900289E-11
```

```
tnc: |fn-fn-1] = 7.84895e-09 -> convergence
   30  290  5.474184762780367E-02   2.47906442E-12
tnc: Converged (|f_n-f_(n-1)| ~= 0)
  NIT   NF   F                        GTG
    0    1  6.931471805599454E-01   2.81791427E+00
    1    5  2.577334650706678E-01   5.35638247E-02
    2   13  1.402788917675887E-01   5.81437434E-03
    3   16  1.297797241873513E-01   2.35271319E-03
tnc: fscale = 20.6165
    4   23  9.574136961774110E-02   7.73803092E-04
    5   29  8.523735694827429E-02   5.74222605E-05
    6   39  7.676039698459325E-02   1.67550685E-05
    7   49  7.291938790038884E-02   2.46141311E-05
    8   58  7.093064246169721E-02   3.77637584E-05
    9   67  6.981594313949885E-02   1.32128563E-05
   10   78  6.877516221675969E-02   1.96751708E-06
tnc: fscale = 712.92
   11   86  6.861171924093007E-02   1.83847896E-06
   12   96  6.832412554285286E-02   1.95524519E-06
   13  105  6.816083235290184E-02   1.68689030E-07
   14  114  6.810826839586437E-02   5.14581408E-08
   15  123  6.809472399972658E-02   5.22713367E-08
   16  132  6.809012130253861E-02   9.22132999E-08
   17  147  6.808498447723441E-02   7.15223375E-09
   18  156  6.808387662888558E-02   1.27937659E-08
   19  165  6.808313781419921E-02   9.62795553E-10
tnc: fscale = 32228
   20  173  6.808284365748071E-02   7.11233864E-09
   21  185  6.808254697972416E-02   2.65114604E-10
   22  196  6.808242110527363E-02   5.40856845E-10
   23  216  6.808228271548894E-02   6.33512303E-11
tnc: |fn-fn-1] = 5.83781e-09 -> convergence
   24  220  6.808227687768081E-02   9.73363303E-11
tnc: Converged (|f_n-f_(n-1)| ~= 0)
  NIT   NF   F                        GTG
    0    1  6.931471805599454E-01   3.04205786E+00
    1    5  2.074115187018571E-01   5.33846002E-02
    2   13  1.095657893195417E-01   1.00046468E-02
    3   19  7.122369578306477E-02   2.23756308E-04
tnc: fscale = 66.8517
    4   28  5.080039233607981E-02   2.07466867E-04
    5   36  4.264753149486246E-02   3.42872161E-04
    6   41  3.966714854255653E-02   2.76342867E-05
    7   51  3.576932026177618E-02   1.09157685E-05
    8   63  3.454602687956630E-02   1.10424416E-06
```

```
    9   72  3.419929121840975E-02   5.50414856E-07
tnc: fscale = 1347.89
   10   81  3.412118209082680E-02   1.33104551E-07
   11   91  3.403979473955124E-02   6.68768151E-08
   12  101  3.401057039854356E-02   1.02826621E-07
   13  111  3.400137886713391E-02   8.64389986E-09
   14  122  3.399628594332726E-02   3.63559480E-09
   15  125  3.399597070959856E-02   2.29234567E-09
   16  133  3.399431700318994E-02   2.82414197E-09
   17  143  3.399299262780816E-02   4.05188600E-09
   18  152  3.399220569147510E-02   4.51626909E-09
   19  161  3.399182501284029E-02   3.60391358E-09
   20  170  3.399157284675095E-02   1.15314338E-09
tnc: fscale = 29448.2
   21  178  3.399145079191456E-02   1.92207288E-09
   22  189  3.399134438414157E-02   1.04384922E-10
   23  199  3.399128136486621E-02   5.47708684E-10
   24  209  3.399124575770539E-02   9.51264481E-11
   25  226  3.399121220711967E-02   7.46287299E-11
tnc: |fn-fn-1] = 9.86727e-09 -> convergence
   26  241  3.399120233985221E-02   5.50028018E-11
tnc: Converged (|f_n-f_(n-1)| ~= 0)
  NIT   NF   F                        GTG
    0    1  6.931471805599454E-01   2.96332657E+00
    1    4  2.926627916958778E-01   1.52687546E-01
    2    8  1.834041288930853E-01   3.71557793E-02
    3   16  1.186429108109784E-01   8.73181998E-03
    4   19  1.130236135773379E-01   3.08687292E-04
tnc: fscale = 56.9168
    5   27  8.154313844284182E-02   1.37655249E-03
    6   35  6.847484057638166E-02   1.01228968E-03
    7   43  6.165646307216006E-02   2.12069759E-04
    8   51  5.798840675326609E-02   1.08196962E-04
    9   59  5.628633012581698E-02   4.38327238E-05
   10   67  5.542799153214299E-02   2.29439747E-05
   11   75  5.498143756568130E-02   2.19959063E-06
   12   84  5.476911037342628E-02   1.12100054E-06
   13   99  5.457849674569144E-02   2.53054187E-07
tnc: fscale = 1987.89
   14  107  5.456182567183528E-02   1.85755706E-07
   15  124  5.452835334675200E-02   1.05790513E-08
   16  139  5.452077260714306E-02   6.81259321E-08
   17  155  5.451786375417378E-02   2.00657964E-09
   18  170  5.451710392006536E-02   3.60203244E-09
   19  179  5.451695468077943E-02   7.34946051E-10
```

```
   20  194  5.451671926261487E-02   2.21723299E-09
   21  202  5.451667621227312E-02   9.45302802E-10
   22  211  5.451663837035899E-02   8.25368013E-11
tnc: fscale = 110072
   23  225  5.451658295837406E-02   2.26204157E-10
tnc: |fn-fn-1] = 1.22056e-08 -> convergence
   24  237  5.451657075275276E-02   5.64605922E-11
tnc: Converged (|f_n-f_(n-1)| ~= 0)
  NIT   NF   F                       GTG
    0    1  6.931471805599454E-01   2.89414573E+00
    1    5  1.102265148016889E-01   1.24776182E-02
    2   14  6.216725305569221E-02   1.48619620E-03
tnc: fscale = 25.9395
    3   21  4.376477931546813E-02   8.12052489E-04
    4   32  3.198656775998845E-02   2.70467825E-04
    5   39  2.713851082584029E-02   4.74143726E-06
    6   52  2.293041701016846E-02   2.55037896E-05
    7   62  2.114493619618927E-02   3.72551735E-06
tnc: fscale = 518.092
    8   70  2.076794344693134E-02   3.08677423E-06
    9   82  2.026608849036079E-02   4.63619323E-07
   10   91  2.018942959964367E-02   4.80770210E-08
   11  101  2.016352949963373E-02   5.71572076E-09
tnc: fscale = 13227.1
   12  109  2.015991456693108E-02   2.63732382E-08
   13  119  2.015714396581978E-02   2.99682082E-09
   14  130  2.015568242594623E-02   1.19200317E-09
   15  140  2.015515345875683E-02   3.42579303E-10
   16  155  2.015468926465705E-02   1.64186650E-09
   17  165  2.015458105218084E-02   8.62504922E-11
   18  175  2.015452012943916E-02   2.35734651E-11
   19  188  2.015447768816316E-02   1.01125873E-11
tnc: fscale = 314462
   20  197  2.015446008204367E-02   1.64697498E-11
tnc: |fn-fn-1] = 1.12327e-08 -> convergence
   21  209  2.015444884936876E-02   1.45231643E-11
tnc: Converged (|f_n-f_(n-1)| ~= 0)
  NIT   NF   F                       GTG
    0    1  6.931471805599454E-01   3.09413052E+00
    1    5  1.331079514505686E-01   1.52839646E-02
    2   14  7.414824428854713E-02   2.47287947E-03
tnc: fscale = 20.1094
    3   21  5.386675999463106E-02   2.70490397E-04
    4   27  4.577000555707196E-02   3.70996639E-05
    5   39  3.749512028940578E-02   5.45467000E-05
```

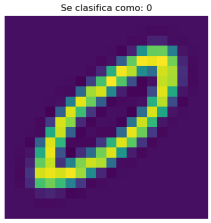
```
    6   50  3.363282236643216E-02   1.39011843E-05
    7   60  3.235791737906045E-02   4.52454236E-06
tnc: fscale = 470.124
    8   68  3.205143067987449E-02   5.74941540E-06
    9   82  3.157408719286951E-02   1.54340354E-07
   10   92  3.150915895488270E-02   5.71534674E-07
   11  102  3.148863246771048E-02   4.39567057E-08
   12  112  3.148137744231337E-02   6.38812347E-09
tnc: fscale = 12511.6
   13  129  3.147497328882636E-02   2.18653331E-09
   14  139  3.147417197479676E-02   1.21282821E-09
   15  149  3.147380405350714E-02   6.02234286E-10
   16  166  3.147350794746514E-02   1.21600040E-10
   17  183  3.147341582536742E-02   5.12528409E-11
   18  193  3.147339647205673E-02   6.18584465E-11
tnc: |fn-fn-1] = 1.36449e-08 -> convergence
   19  203  3.147338282714533E-02   1.64580859E-11
tnc: Converged (|f_n-f_(n-1)| ~= 0)
  NIT   NF   F                       GTG
    0    1  6.931471805599454E-01   2.65150865E+00
    1    4  3.120953360546006E-01   1.42828377E-01
    2    9  1.994585194083947E-01   1.30079054E-02
    3   18  1.549463808300993E-01   3.02157410E-03
    4   28  1.373033153147032E-01   3.32442733E-04
tnc: fscale = 54.8456
    5   34  1.299009967809901E-01   9.93641550E-05
    6   42  1.260508144856108E-01   4.21723895E-04
    7   50  1.220540709493017E-01   2.19621852E-04
    8   58  1.200903213888479E-01   6.36209568E-05
    9   74  1.182970201432051E-01   6.10938083E-06
   10   83  1.180370847159988E-01   6.51275753E-06
   11   93  1.178465995388484E-01   1.50143967E-06
   12  102  1.177472795362853E-01   2.06372353E-06
   13  111  1.176969041007960E-01   6.27224265E-07
tnc: fscale = 1262.67
   14  119  1.176836959914922E-01   2.49171484E-07
   15  129  1.176643696106126E-01   4.73027235E-07
   16  138  1.176543378179852E-01   7.39173640E-08
   17  147  1.176505818050412E-01   9.67270268E-08
   18  156  1.176480596467865E-01   2.79327007E-08
   19  165  1.176460231459407E-01   1.16943994E-07
   20  174  1.176442800718570E-01   6.51483409E-08
   21  190  1.176417542152133E-01   1.69096750E-08
   22  205  1.176407654760372E-01   2.23676695E-08
   23  220  1.176403811279267E-01   5.52436574E-10
```

```
tnc: fscale = 42546
   24  228  1.176402878469479E-01   2.95129119E-09
   25  239  1.176401922880832E-01   1.71889335E-09
   26  250  1.176401357969725E-01   8.67527990E-11
   27  261  1.176401085899494E-01   2.93627829E-10
   28  272  1.176400886079680E-01   4.13428969E-11
tnc: |fn-fn-1] = 1.19043e-08 -> convergence
   29  282  1.176400767036370E-01   4.80790552E-11
tnc: Converged (|f_n-f_(n-1)| ~= 0)
  NIT   NF   F                        GTG
    0    1  6.931471805599454E-01   2.91109822E+00
    1    4  3.044505421043397E-01   1.58302204E-01
    2    9  1.839572697886531E-01   4.39504396E-02
    3   13  1.272764407303535E-01   1.30180745E-03
tnc: fscale = 27.7157
    4   22  1.063432278034810E-01   5.66090460E-04
    5   30  9.602007316583769E-02   5.39992105E-04
    6   40  8.967402719182710E-02   3.63893330E-04
    7   48  8.666104174580500E-02   4.34480041E-05
    8   63  8.452560970124802E-02   3.65613595E-05
    9   71  8.422293393021235E-02   4.96745363E-06
   10   80  8.396718940082323E-02   6.10581657E-06
   11   89  8.382067026667704E-02   9.56400626E-06
   12   99  8.370991011974449E-02   3.02720205E-07
tnc: fscale = 1817.52
   13  108  8.367803068519619E-02   7.51285141E-08
   14  119  8.363273872109819E-02   4.62623526E-08
   15  130  8.362073196652599E-02   2.50825328E-08
   16  140  8.361830872394654E-02   1.27630805E-08
   17  156  8.361493303598548E-02   1.58986711E-08
   18  165  8.361446398006844E-02   1.21685005E-08
   19  175  8.361398545206930E-02   5.39740476E-09
   20  185  8.361365465079275E-02   1.25220776E-09
   21  195  8.361337993584436E-02   6.11909598E-10
tnc: fscale = 40425.6
   22  204  8.361316289829251E-02   1.37354308E-09
   23  221  8.361294171881899E-02   2.47616848E-10
   24  232  8.361288354472506E-02   2.20010440E-10
   25  242  8.361284181599120E-02   1.29913696E-10
   26  251  8.361281833015373E-02   1.29881768E-10
   27  261  8.361280004849345E-02   5.15370688E-10
tnc: |fn-fn-1] = 1.3995e-08 -> convergence
   28  271  8.361278605344641E-02   2.08482943E-10
tnc: Converged (|f_n-f_(n-1)| ~= 0)
Hay un 95.88 % de aciertos
```

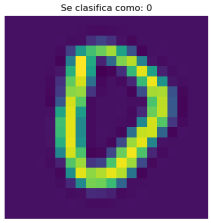
Hay un 95.88 % de aciertos con un *lamda* **0,1**

Se muestra y guarda los 10 distintos casos de prueba con la interpretación tomada sobre ellos:

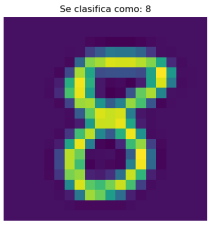
a)



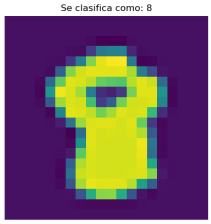
b)



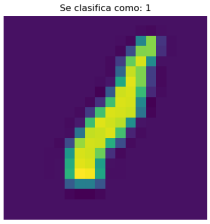
c)



d)



e)

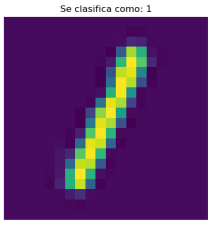


f)

g)

h)

i)



j)

3.3. Código

A continuación presento el código de la parte 1:

```
1
2  import numpy as np
3  import copy
4  from scipy.io import loadmat
5  import matplotlib.pyplot as plt
6  import scipy.optimize as opt
7
8
9  def sigmoid(x):
10     s = 1 / (1 + np.exp(-x))
11     return s
12
13 def cost(theta, X, Y):
14     H = sigmoid(np.matmul(X, theta))
15     cost = (- 1 / (len(X))) * (np.dot(Y, np.log(H)) + np.dot((1 -↩
           Y), np.log(1 - H)))
16     return cost
17
18 def gradient(theta, XX, Y):
19     H = sigmoid( np.matmul(XX, theta) )
20     grad = (1 / len(Y)) * np.matmul(XX.T, H - Y)
21     return grad
22
23 def cost_regul(theta, X, Y, lam):
24   m = len(X)
25   coste1 = -(1 / m) * np.sum(Y.T * np.log(sigmoid(np.matmul(X, ↩
        theta))) + (1 - Y.T) * np.log(1 - sigmoid(np.matmul(X, theta↩
        ))))
26   coste2 = (lam / (2 * m)) * np.sum(theta[1:]**2)
27   return coste1 + coste2
28
29 def gradient_regul(theta, XX, Y, lam):
30     H = np.array([sigmoid(np.dot(XX, theta))]).T
31     thetaNoZeroReg = np.insert(theta[1:], 0, 0)
32     thetaNoZeroReg =np.array([np.hstack([0, theta[1:]])]).T
33     gradient =  (np.dot(XX.T, (H - Y)) + lam * thetaNoZeroReg) / ↩
          len(Y)
34     return gradient
35
36
37 def costeMinimo(XX, Y, lam):
38     initialTheta = np.zeros(len(XX[0]))
```

```
39      result = opt.fmin_tnc(func=cost_regul, x0=initialTheta, ↩
           fprime=gradient_regul, args=(XX, Y, lam)) #regularizado
40      return result[0]
41
42  def oneVsAll(X, y, num_etiquetas, reg):
43      thetas = np.zeros((num_etiquetas, len(X[0])))
44      y[y == 10] = 0
45      for e in range(num_etiquetas):
46          yy = (y == e).astype('int')
47          thetas[e] = costeMinimo(X, yy, reg)
48      return thetas
49
50  def clasificador(X, Thetas):
51    res = [0] * len(Thetas)
52    i = 0
53    for t in Thetas:
54      res[i] = sigmoid(np.dot(t, X.T))
55      i += 1
56    return np.argmax(res[:10])
57
58  def exec(X, thetas,n):
59      for i in range(n):
60          sample = np.random.choice(X.shape[0], 1)
61          plt.imshow(X[sample, :].reshape(-1, 20).T)
62          plt.axis('off')
63          val = clasificador(X[sample, :], thetas)
64          plt.title("Se␣clasifica␣como:␣"+str(val))
65          plt.savefig("ejemplo"+str(i)+".png")
66          plt.show()
67
68  def evaluaPorcentaje(X,Y,Theta):
69      cont = 0
70      m = len(X)
71      for i in range(m):
72          clasificado = clasificador(X[i], Theta)
73          if (clasificado == Y[i]):
74              cont += 1
75      print("Hay␣un␣"+ str((cont/m)*100) + "␣%␣de␣aciertos")
76
77  def main():
78      data = loadmat ('ex3data1.mat')
79      y = data ['y']
80      X = data ['X']
81
82      sample=np.random.choice(X.shape[0], 10)
```

```
83      plt.imshow(X[sample, :].reshape(-1, 20).T)
84      plt.axis('off')
85      plt.show()
86
87      thetas = oneVsAll(X, y, 10, 0.1)
88      exec(X, thetas,10)
89      evaluaPorcentaje(X,y,thetas)
90
91  #######
92  main()
```

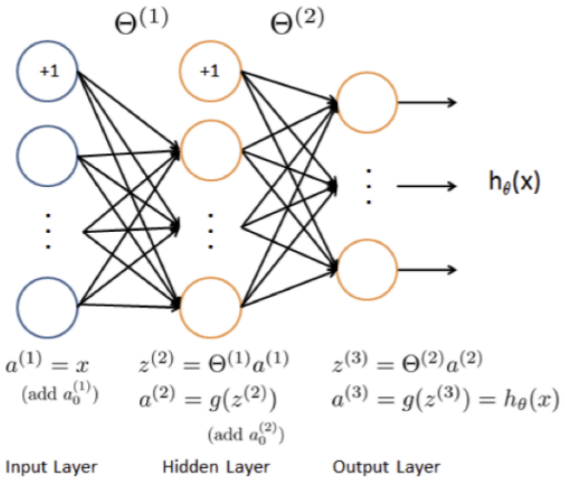
Código parte 1: p3.py

Capítulo 4

Parte 2: Redes neuronales

En esta parte se utilizarán los pesos proporcionados para una red neuronal ya entrenada sobre los ejemplos para evaluar su precisión sobre esos mismos ejemplos.

Como ya se indicó en la introducción, esta será la **estructura** de la red:



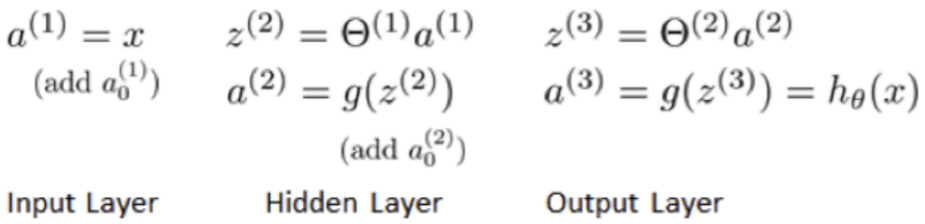
4.1. Funciones

Las funciones para la parte 2 son las siguientes:

- **sigmoid(x):** Calcula el valor de la función **sigmoide**:

$$g(z) = \frac{1}{1 + e^{-z}}$$

- **exec(X, theta1, theta2):** Esta función realiza los cálculos de las salidas de cada una de las dos capas tal y como lo indica la siguiente fórmula:



- **evaluaPorcentaje(H,Y):** Evalúa el número de aciertos. Con la ayuda de np.argmax() devuelve la hipótesis máxima de esa fila y la compara con la Y.

- **main():** Carga los pesos en *thetha1* y *theta2*, carga los datos en X e y y ahce las llamadas a *exec()* y a *evaluaPorcentaje()*.

4.2. Ejecución

En la consola se va a indicar cual ha sido el **porcentaje de acierto**. De esta manera:

```
Hay un 97.52 % de aciertos
```

Con el **97,52 %** se obtiene el porcentaje de acierto mencionado en el guión de la práctica.

4.3. Código

```
1  import numpy as np
2  import copy
3  from scipy.io import loadmat
4  import matplotlib.pyplot as plt
5  import scipy.optimize as opt
6
7  def sigmoid(x):
8      s = 1 / (1 + np.exp(-x))
9      return s
10
11 def exec(X, theta1, theta2):
12     m = len(X)
13     a1 = np.hstack([np.ones((m, 1)), X])
14
15     z2 = np.dot(theta1, a1.T)
16     a2 = sigmoid(z2)
17     a2 = np.hstack([np.ones((m, 1)), a2.T])
18     z3 = np.dot(theta2, a2.T)
19     return sigmoid(z3) #a3
20
21 def evaluaPorcentaje(H,Y):
22     cont = 0
23     m = len(Y)
24     for i in range(m):
25         if (np.argmax(H.T[i]) + 1) == Y[i, 0]:
26             cont += 1
27     print("Hay un "+ str((cont/m)*100) + "% de aciertos")
28
29 def main():
30     weights = loadmat('ex3weights.mat')
31     theta1, theta2 = weights['Theta1'], weights['Theta2']# Theta1↩
           es de dimensi n 25x401 ; Theta2 es de dimensi n 10x26
32
33     data = loadmat ('ex3data1.mat')
34     y = data ['y']
35     X = data ['X']
36
37     h = exec(X, theta1, theta2)
38     evaluaPorcentaje(h, y)
39
40 #######
41 main()
```

Código parte 2: p3.2.py