Universidad Complutense de Madrid

IAAC - PRÁCTICA 6

Yaco Alejandro Santiago Pérez

Asignatura: INTELIGENCIA ARTIFICIAL APLICADA A INTERNET DE LAS COSAS

P6: Redes neuronales con Keras

Master IOT

27 de abril de 2020

Índice general

1. Introducción **1**

2. Objetivos **2**

3. Parte 1: Contacto con Keras **3**
3.1. Funciones propias 3
3.2. Ejecución 4
3.2.1. Problema 0: Problema lineal 4
3.2.2. Problema 1: Problema complejo: Lunas 5
3.2.3. Problema 3: Problema complejo: Círculos 6
3.2.4. Problema 3: Problema complejo: Espirales 8
3.3. Código 11

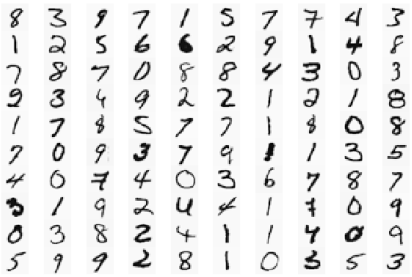
4. Parte 2:Práctica 4 con Keras **15**
4.1. Funciones 15
4.2. Ejecución 16
4.2.1. Modo 1 17
4.2.2. Modo 2 18
4.2.3. Modo 3 19
4.2.4. Modo 4 20
4.2.5. Modo 5 21
4.2.6. Modo 6 22
4.2.7. Conclusión 22
4.3. Código 23

Capítulo 1

Introducción

Esta práctica supone la primera puesta en contacto con **TensorFlow mediante Keras**, el *framework* de **Redes Neuronales**.

A lo largo de su desarrollo se aprenderá a formar distintas redes neuronales, en función del numero de capas y nodos y de las funciones de activación. También se modificará la practica 4 para utilizar una red neuronal construida con **Keras** para el reconocimiento de dígitos manuscritos.



Capítulo 2

Objetivos

En esta práctica, la cual se divide en dos partes:

En la **primera parte** se construyen redes neuronales simples para resolver problemas de clasificación de datos.

En la **segunda parte** se pretende solucionar la práctica **4** reconociendo dígitos con redes neuronales creadas con **keras**.

Como objetivos concretos se van a perseguir los siguientes:

- Calcular la regresión logística sobre dos dimensiones
- Calcular la regresión logística no lineal resolviendo el problema de las **lunas**
- Calcular la regresión logística no lineal resolviendo el problema de los **círculos**
- Calcular la regresión logística no lineal resolviendo el problema de la **espiral**
- Modificar la **práctica 4** para solventarlo mediante **Keras**

Capítulo 3

Parte 1: Contacto con Keras

Además de las funciones propias, se han utilizado las funciones proporcionadas en *helper.py*[1].

3.1. Funciones propias

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

- **menu(x):** Debido a que hay que realizar diferentes ejercicios en función de los datos y las redes neuronales configuradas este menú da la opción al usuario de elegir el caso que quiere ejecutar.
  Muestra la list ay solicita la opción al usuario.
- **problemaX(num, model, X, y, ver, ep, multi=False, yy=0):** Para no repetir código se ha generalizado las operaciones para todos los casos dentro de esta función. Como argumentos se le pasan el numero de ejercicio, las X, las Y, la versión del ejercicio, el numero de *epochs*, si se trata de multi-categoría, y por ultimo la y sin categorizar para los casos multi-categóricos.
- **main():**Función que hace la llamada al *menu()* y en función de la respuesta llama a *problemaX()* con los argumentos necesarios.

[1]Comentar que debido a que daba error he cambiado *"plt.cm.RdYlBu"* y todas las referencias similares por *"plt.cm.get_cmap(RdYlBu")"*

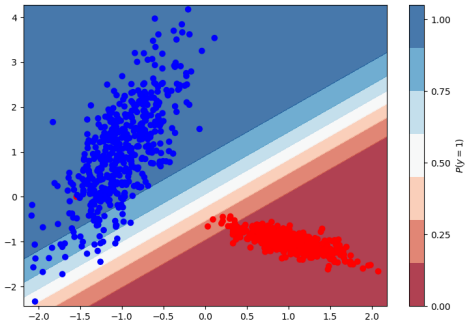
3.2. Ejecución

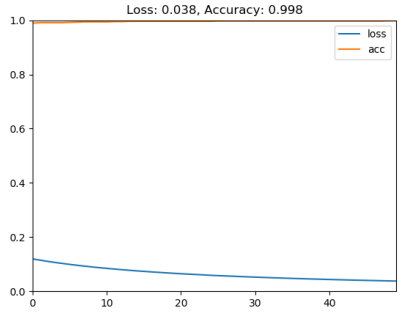
Para todas las ejecuciones se presenta el siguiente menú

```
Using TensorFlow backend.
---MENU---

Ejercicios:

(1) - [2 Dimensiones] Regresion logistica de dos dimensiones
(2) - [Lunas v1]
(3) - [Lunas v2] - Clasifica bien
(4) - [Circulos v1]
(5) - [Circulos v2] - Clasifica bien
(6) - [Espiral v1]
(7) - [Espiral v2] - Clasifica bien
(0) - Salir
Seleccione ejercicio a calcular (numero):
```

Se presentan las **gráficas** y en la consola se ven los **logs** obtenidos en las diferentes ejecuciones:

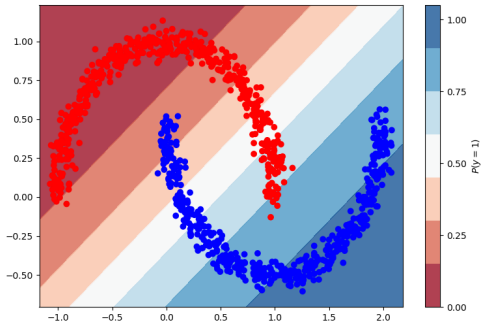
3.2.1. Problema 0: Problema lineal

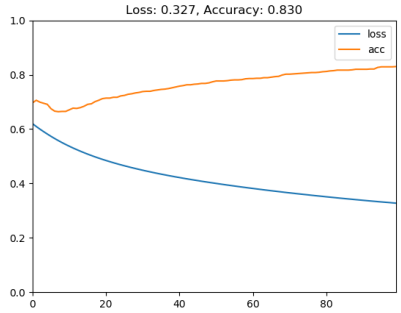




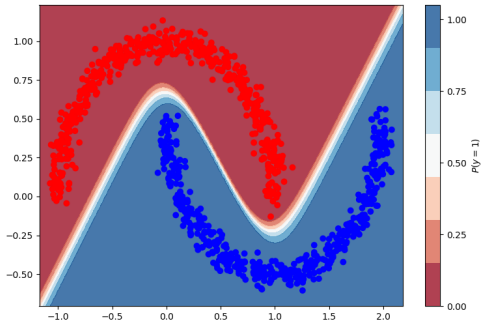

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       501
           1       1.00      1.00      1.00       499

    accuracy                           1.00      1000
   macro avg       1.00      1.00      1.00      1000
weighted avg       1.00      1.00      1.00      1000
```

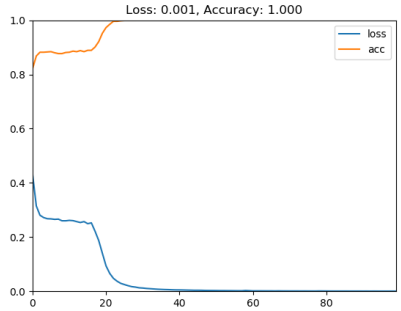
3.2.2. Problema 1: Problema complejo: Lunas

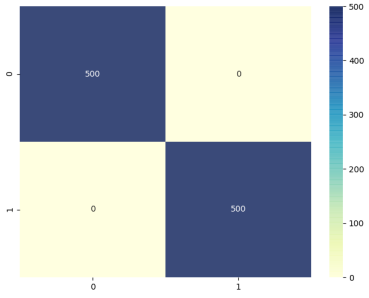




Tras cambiar la red por una con **3 capas**, de **4, 2 y 1** neuronas y funciones de activación *tanh*,*tanh* y *sigmoide* cada una, se ajusta **correctamente**. Obteniendo este resultado:

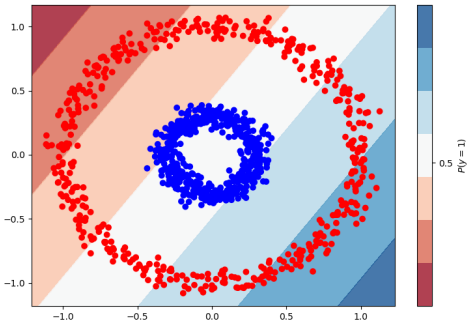


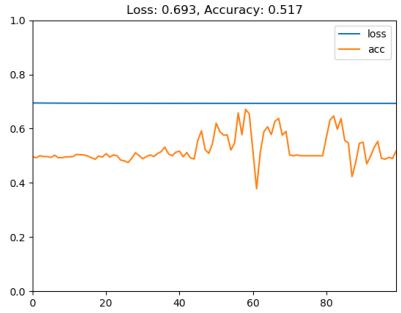


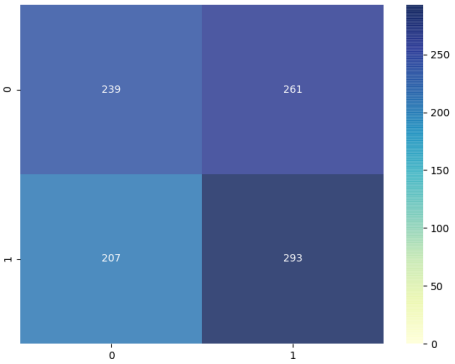


	precision	recall	f1-score	support
0	1.00	1.00	1.00	500
1	1.00	1.00	1.00	500
accuracy			1.00	1000
macro avg	1.00	1.00	1.00	1000
weighted avg	1.00	1.00	1.00	1000

3.2.3. Problema 3: Problema complejo: Círculos

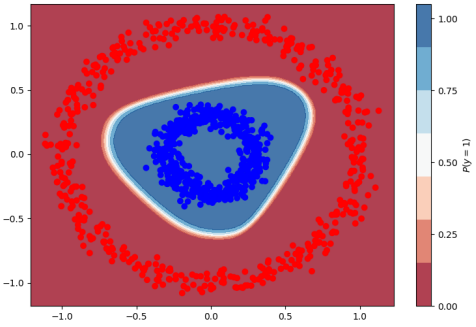


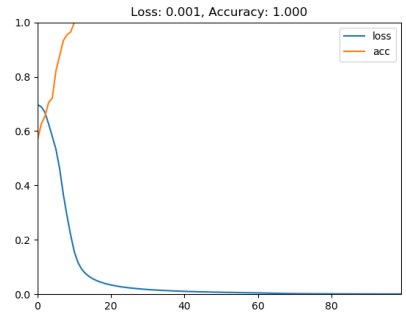


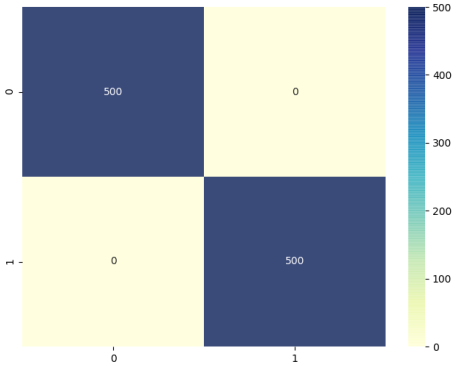


	precision	recall	f1-score	support
0	0.54	0.48	0.51	500
1	0.53	0.59	0.56	500
accuracy			0.53	1000
macro avg	0.53	0.53	0.53	1000
weighted avg	0.53	0.53	0.53	1000

Tras cambiar la red por una con **3 capas**, de **4, 2 y 1** neuronas y funciones de activación *tanh*,*tanh* y *sigmoide* cada una, se ajusta **correctamente**. Obteniendo este resultado:







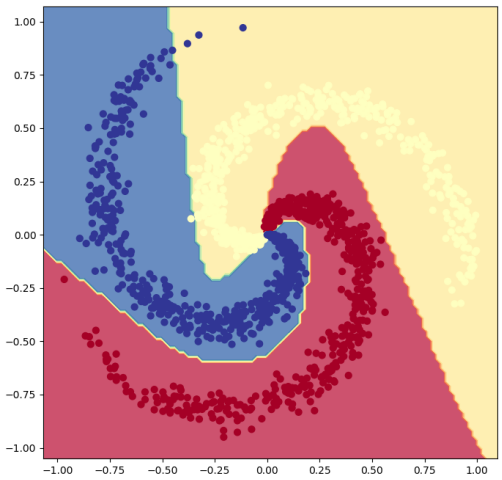
	precision	recall	f1-score	support
0	0.54	0.48	0.51	500
1	0.53	0.59	0.56	500
accuracy			0.53	1000
macro avg	0.53	0.53	0.53	1000
weighted avg	0.53	0.53	0.53	1000

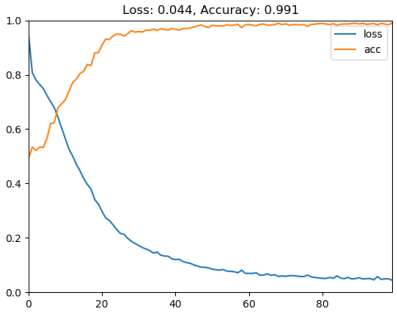
3.2.4. Problema 3: Problema complejo: Espirales

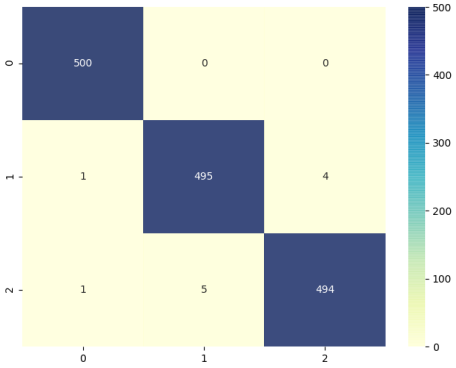
En este caso se trata de un modelo multi-clase, por lo que hay que realizar ciertas operaciones distintas con respecto a las vistas hasta el momento.

Para esta red neuronal, la función de activación de la tercera capa pasa a ser **softmax**, y en compilación se utiliza *loss='**categorical_crossentropy**'*.

Además, se va a ahcer uso de la función **plot_multiclass_decision_boundary** en vez de la usada hasta el momento **plot_decision_boundary**.



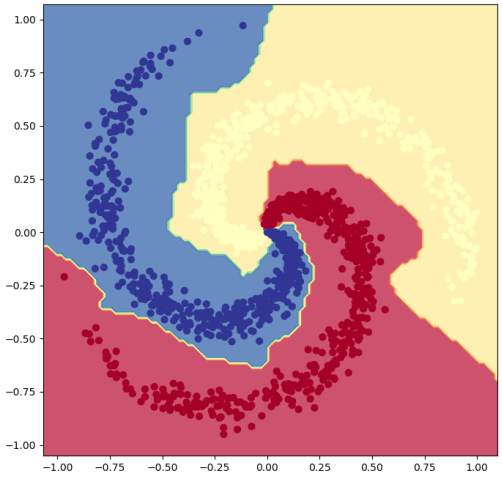


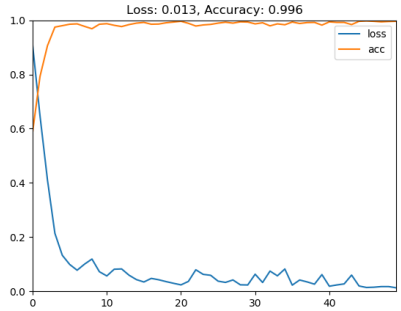


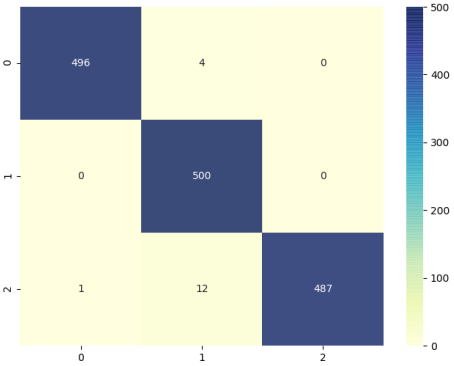
	precision	recall	f1-score	support
0	1.00	1.00	1.00	500
1	0.99	0.98	0.99	500
2	0.99	0.99	0.99	500
micro avg	1.00	0.99	0.99	1500
macro avg	1.00	0.99	0.99	1500
weighted avg	1.00	0.99	0.99	1500
samples avg	0.99	0.99	0.99	1500

Los resultados obtenidos son bastante aceptables, pero se puede intentar adaptarla más aún.

Se aumenta en una capa más la red y los nodos de cada capa aumentan de [4]-[2]-[3] a **[64]-[32]-[16]-[3]**







	precision	recall	f1-score	support
0	0.98	1.00	0.99	500
1	0.98	1.00	0.99	500
2	1.00	0.97	0.98	500
micro avg	0.99	0.99	0.99	1500
macro avg	0.99	0.99	0.99	1500
weighted avg	0.99	0.99	0.99	1500
samples avg	0.99	0.99	0.99	1500

3.3. Código

```
1  import helper as help
2  from sklearn.datasets import make_classification, make_moons, ↩
       make_circles
3  from sklearn.linear_model import LogisticRegression
4  from keras.models import Sequential
5  from keras.optimizers import Adam
6  from keras.layers import Dense
7  from sklearn.metrics import classification_report
8  from keras.utils.np_utils import to_categorical
9
10 def problemaX(num, model, X, y, ver, ep, multi=False, yy=0):
11     if multi==False:
12         help.plot_data(X,y)
13         help.plt.savefig("problema"+str(num)+"-"+str(ver)+"-graf0↩
               .png")
14
15     history = model.fit(x=X, y=y, verbose=0, epochs=ep)
16     help.plot_loss_accuracy(history)
17     if multi==False:
18         help.plt.close(2)
19     else:
20         help.plt.close(1)
21     help.plt.savefig("problema"+str(num)+"-"+str(ver)+"-graf1.png↩
           ")
22
23     #IF segun si es multiclass
24     if multi==False:
25         help.plot_decision_boundary(lambda x: model.predict(x), X↩
               , y)
26     else:
27         help.plot_multiclass_decision_boundary(model, X, yy)
28
29     help.plt.savefig("problema"+str(num)+"-"+str(ver)+"-graf2.png↩
           ")
30
31     if multi==False:
32         yy=y
33     help.plot_confusion_matrix(model, X, yy)
34     help.plt.savefig("problema"+str(num)+"-"+str(ver)+"-graf3.png↩
           ")
35
36     if multi==False:
```

```
37          pred= help.np.concatenate( (model.predict(X)>0.5), axis=0)
                ).astype(int)
38      else:
39          pred= (model.predict(X)>0.5).astype(int)
40
41      print(classification_report(y, pred))
42      help.plt.show()
43
44  def menu():
45      print("---MENU---\n")
46      print("Ejercicios:\n")
47      print("(1) - [2 Dimensiones] Regresi n logistica de dos
            dimensiones")
48      print("(2) - [Lunas v1] ")
49      print("(3) - [Lunas v2] - Clasifica bien")
50      print("(4) - [Circulos v1] ")
51      print("(5) - [Circulos v2] - Clasifica bien")
52      print("(6) - [Espiral v1] ")
53      print("(7) - [Espiral v2] - Clasifica bien")
54      print("(0) - Salir")
55
56      return int(input("Seleccione ejercicio a calcular (numero):")
            )
57
58  def main():
59      op = menu()
60      if op==0:
61          print("Saliendo...")
62      elif op==1:
63          #1 - Regresi n logistica de dos dimensiones (con keras)
64
65          X1, y1 = make_classification(n_samples=1000, n_features
                =2,
66                                       n_redundant=0, n_informative=2,
                                            random_state=7,
                                            n_clusters_per_class=1)
67          model10 = Sequential()
68          model10.add(Dense(units=1, input_shape=(2,), activation='
                sigmoid'))
69          model10.compile(optimizer='adam', loss='
                binary_crossentropy', metrics=['acc'])
70          problemaX(1,model10, X1,y1,0,50)
71
72      elif  op==2:
73          #2 - version 0 -Regresi n logistica no lineal - lunas
```

```
74          X2, y2 = make_moons(n_samples=1000, noise=0.05, ↩
               random_state=0)
75
76          model20 = Sequential()
77          model20.add(Dense(units=1, input_shape=(2,), activation='↩
               sigmoid'))
78
79          model20.compile(optimizer='adam', loss='↩
               binary_crossentropy', metrics=['acc'])
80          problemaX(2,model20, X2,y2,0,100)
81
82      elif  op==3:
83          #2 - version 1 - Cambiando el modelo para adaptarlo a las↩
                lunas
84          X2, y2 = make_moons(n_samples=1000, noise=0.05, ↩
               random_state=0)
85          model21 = Sequential()
86          model21.add(Dense(units=4, input_shape=(2,), activation='↩
               tanh'))
87          model21.add(Dense(units=2, activation='tanh'))
88          model21.add(Dense(units=1, activation='sigmoid'))
89
90          #model21.compile(optimizer='adam', loss='↩
               binary_crossentropy', metrics=['acc'])
91          model21.compile(Adam(lr=0.01), loss='binary_crossentropy'↩
               , metrics=['acc'])
92          problemaX(2,model21, X2,y2,1,100)
93
94      elif  op==4:
95          #3 - version 0 -Regresi n logistica no lineal - circulos
96          X3, y3 = make_circles(n_samples=1000, noise=0.05, factor↩
               =0.3, random_state=0)
97
98          model30 = Sequential()
99          model30.add(Dense(units=1, input_shape=(2,), activation='↩
               sigmoid'))
100
101         model30.compile(optimizer='adam', loss='↩
               binary_crossentropy', metrics=['acc'])
102         problemaX(3,model30, X3,y3,0,100)
103     elif  op==5:
104         #3 - version 1 -Regresi n logistica no lineal - circulos
105         X3, y3 = make_circles(n_samples=1000, noise=0.05, factor↩
               =0.3, random_state=0)
106
```

```
107         model31 = Sequential()
108         model31.add(Dense(units=4, input_shape=(2,), activation='↩
                tanh'))
109         model31.add(Dense(units=2, activation='tanh'))
110         model31.add(Dense(units=1, activation='sigmoid'))
111 
112         model31.compile(Adam(lr=0.01), loss='binary_crossentropy'↩
                , metrics=['acc'])
113         problemaX(3,model31, X3,y3,1,100)
114 
115     elif  op==6:
116         #4 - version 0 -Regresi n logistica no lineal - Espiral
117         X4, y4 = help.make_multiclass(K=3)
118 
119         model40 = Sequential()
120         model40.add(Dense(units=4, input_shape=(2,), activation='↩
                tanh'))
121         model40.add(Dense(units=2, activation='tanh'))
122         model40.add(Dense(units=3, activation='softmax'))
123 
124         model40.compile(Adam(lr=0.01), loss='↩
                categorical_crossentropy', metrics=['acc'])
125         y4_cat = to_categorical(y4)
126         problemaX(4,model40, X4,y4_cat,0,100, True,y4)
127 
128     elif  op==7:
129         #4 - version 1 -Regresi n logistica no lineal - Espiral ↩
                mejorada
130         X4, y4 = help.make_multiclass(K=3)
131 
132         model41 = Sequential()
133         model41.add(Dense(units=64, input_shape=(2,), activation=↩
                'tanh'))
134         model41.add(Dense(units=32, activation='tanh'))
135         model41.add(Dense(units=16, activation='softmax'))
136         model41.add(Dense(units=3, activation='softmax'))
137 
138         model41.compile(Adam(lr=0.01), loss='↩
                categorical_crossentropy', metrics=['acc'])
139         y4_cat = to_categorical(y4)
140         problemaX(4,model41, X4,y4_cat,1,50, True,y4)
141     else:
142         print("Opcion␣erronea")
143 main()
```

Código parte 1: p6.py

Capítulo 4

Parte 2:Práctica 4 con Keras

En esta parte se utilizará **Keras para solucionar la práctica 4 de reconocimiento de dígitos manuscritos.**

Figure 4.1: *Muestra de 100 números a identificar*

4.1. Funciones

Las funciones de la parte 2 son las siguientes:

- **def problemaX(num, model, X, y, ver, ep, y_noCat, X_test, y_test):** Esta función hace el entrenamiento, generando y pintando las gráficas correspondientes de *evolución del coste y la precisión* y la gráfica de *mapa de calor que representa la matriz de confusión del modelo.*
  Y seguidamente, testea el modelo con los datos de test, que suponen el 30 por ciento de las muestras.

  De esta manera se obtiene el **porcentaje de acierto** y el **coste** sobre la muestra de testeo.

- **menu():** El *menú* muestra las distintas opciones de configuración del modelo de la red neuronal para que el usuario pueda elegirlos y comparar los resultados.

- **main():**Función que carga los datos, divide en 70-30 por ciento los datos para entrenamiento y pruebas, a continuación convierte los arrays Y en formato de categorías y hace la llamada al *menu()* y en función de la respuesta llama a *problemaX()* con los argumentos pertinentes.

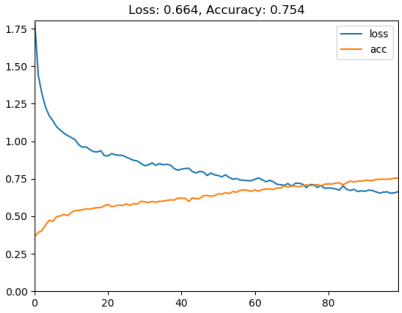
4.2. Ejecución

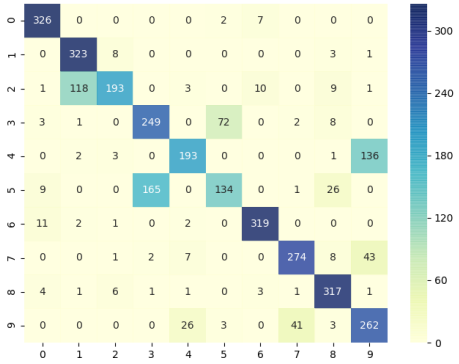
En la consola se va mostrar siempre la elección del modelo a utilizar. De esta manera:

```
---MENU---

Ejercicios:

(1) - [4-tanh]-[2-tanh]-[10-softmax]
(2) - [64-tanh]-[32-tanh]-[16-softmax]-[10-softmax]
(3) - [64-tanh]-[32-tanh]-[16-softmax]-[10-sigmoid]
(4) - [64-tanh]-[32-tanh]-[16-tanh]-[10-sigmoid]
(5) - [64-tanh]-[32-tanh]-[16-tanh]-[10-softmax]
(6) - [128-tanh]-[64-tanh]-[32-tanh]-[10-softmax]
(0) - Salir
Seleccione ejercicio a calcular (numero):
```

4.2.1. Modo 1

La estructura de la red es la siguiente:
[4-tanh]-[2-tanh]-[10-softmax]



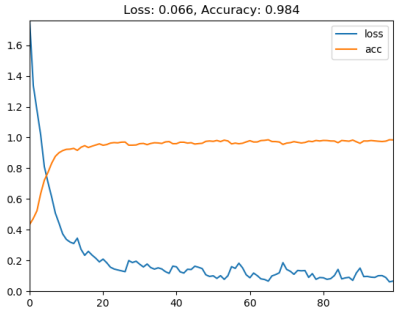


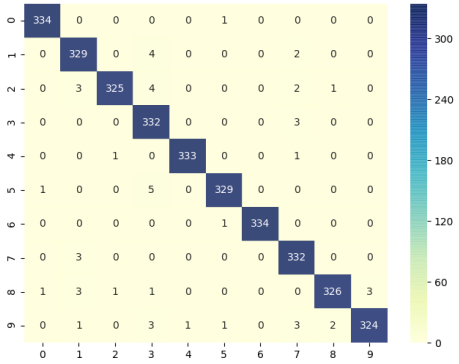
	precision	recall	f1-score	support
0	0.93	0.97	0.95	335
1	0.72	0.96	0.82	335
2	0.91	0.56	0.70	335
3	0.70	0.05	0.09	335
4	0.85	0.56	0.67	335
5	0.69	0.22	0.34	335
6	0.94	0.95	0.95	335
7	0.87	0.81	0.84	335
8	0.86	0.94	0.90	335
9	0.62	0.77	0.69	335
micro avg	0.82	0.68	0.74	3350
macro avg	0.81	0.68	0.69	3350
weighted avg	0.81	0.68	0.69	3350
samples avg	0.68	0.68	0.68	3350

```
Evaluando la parte de Test: Coste = 1.1946632780450763, Precision↩
    =67.5151526927948
```

4.2.2. Modo 2

La estructura de la red es la siguiente:
[64-tanh]-[32-tanh]-[16-softmax]-[10-softmax]



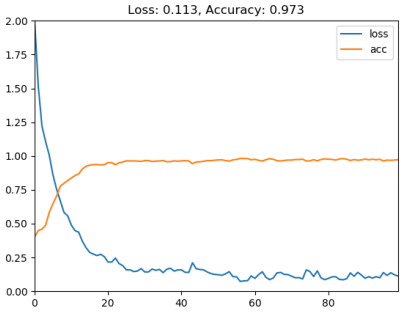


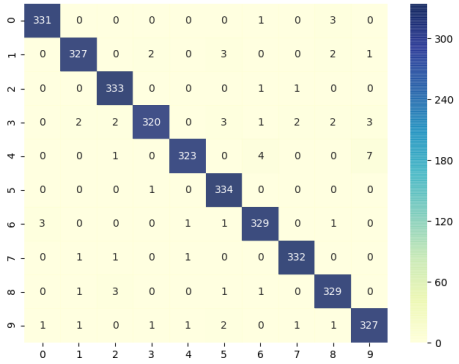
	precision	recall	f1-score	support
0	0.99	1.00	1.00	335
1	0.97	0.98	0.98	335
2	1.00	0.97	0.98	335
3	0.97	0.99	0.98	335
4	1.00	0.99	1.00	335
5	0.99	0.98	0.98	335
6	1.00	0.99	1.00	335
7	0.97	0.99	0.98	335
8	1.00	0.97	0.98	335
9	0.99	0.97	0.98	335
micro avg	0.99	0.98	0.99	3350
macro avg	0.99	0.98	0.99	3350
weighted avg	0.99	0.98	0.99	3350
samples avg	0.98	0.98	0.98	3350

```
Evaluando la parte de Test: Coste = 0.37549366558139974, ↩
   Precision =92.48484969139099
```

4.2.3. Modo 3

La estructura de la red es la siguiente:
[64-tanh]-[32-tanh]-[16-softmax]-[10-sigmoid]



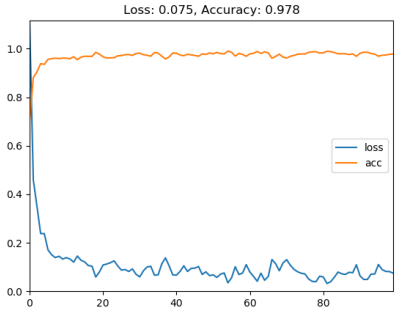


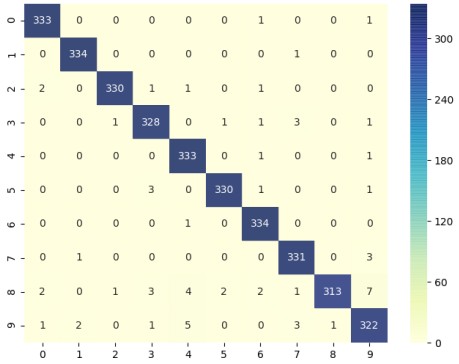
	precision	recall	f1-score	support
0	0.00	0.00	0.00	335
1	0.00	0.00	0.00	335
2	0.00	0.00	0.00	335
3	0.00	0.00	0.00	335
4	0.00	0.00	0.00	335
5	0.00	0.00	0.00	335
6	0.00	0.00	0.00	335
7	0.00	0.00	0.00	335
8	0.00	0.00	0.00	335
9	0.00	0.00	0.00	335
micro avg	0.00	0.00	0.00	3350
macro avg	0.00	0.00	0.00	3350
weighted avg	0.00	0.00	0.00	3350
samples avg	0.00	0.00	0.00	3350

```
Evaluando la parte de Test: Coste = 0.38735046766021036, ↩
   Precision =92.54545569419861
```

4.2.4. Modo 4

La estructura de la red es la siguiente:
[64-tanh]-[32-tanh]-[16-tanh]-[10-sigmoid]



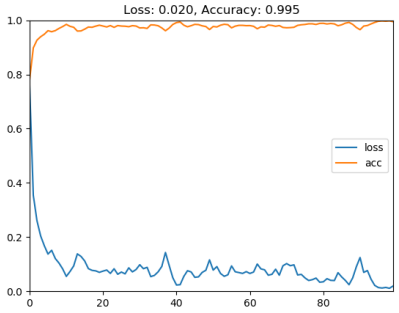


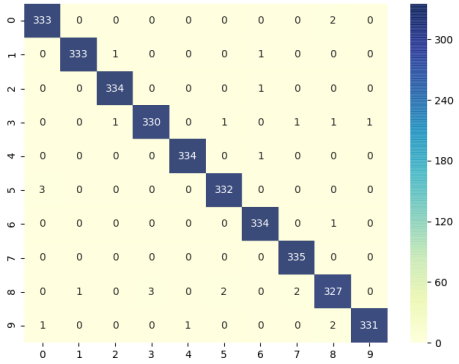
	precision	recall	f1-score	support
0	1.00	0.76	0.86	335
1	1.00	0.73	0.84	335
2	1.00	0.67	0.80	335
3	1.00	0.01	0.01	335
4	0.94	0.05	0.09	335
5	1.00	0.32	0.48	335
6	1.00	0.78	0.88	335
7	0.98	0.27	0.43	335
8	0.00	0.00	0.00	335
9	0.93	0.04	0.07	335
micro avg	1.00	0.36	0.53	3350
macro avg	0.88	0.36	0.45	3350
weighted avg	0.88	0.36	0.45	3350
samples avg	0.36	0.36	0.36	3350

```
Evaluando la parte de Test: Coste = 0.41202714880307517, ↩
   Precision =91.57575964927673
```

4.2.5. Modo 5

La estructura de la red es la siguiente:
[64-tanh]-[32-tanh]-[16-tanh]-[10-softmax]



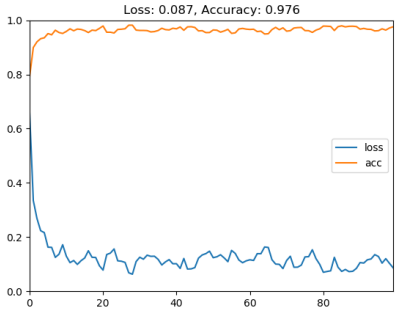


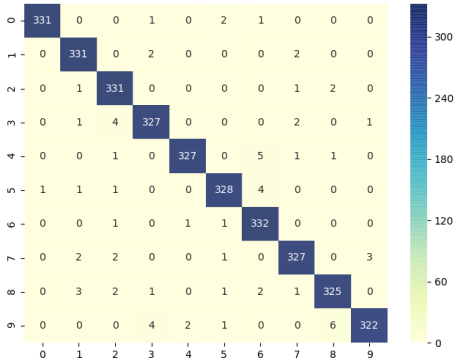
	precision	recall	f1-score	support
0	0.99	0.99	0.99	335
1	0.99	0.99	0.99	335
2	1.00	1.00	1.00	335
3	0.98	0.98	0.98	335
4	1.00	0.99	0.99	335
5	1.00	0.97	0.98	335
6	0.99	0.99	0.99	335
7	0.99	1.00	1.00	335
8	0.99	0.97	0.98	335
9	0.98	0.98	0.98	335
micro avg	0.99	0.99	0.99	3350
macro avg	0.99	0.99	0.99	3350
weighted avg	0.99	0.99	0.99	3350
samples avg	0.99	0.99	0.99	3350

```
Evaluando la parte de Test: Coste = 0.4649375308372758, Precision↩
    =90.84848761558533
```

4.2.6. Modo 6

La estructura de la red es la siguiente:
[128-tanh]-[64-tanh]-[32-tanh]-[10-softmax]





	precision	recall	f1-score	support
0	1.00	0.99	0.99	335
1	0.98	0.99	0.98	335
2	0.97	0.99	0.98	335
3	0.98	0.97	0.98	335
4	0.99	0.98	0.98	335
5	0.98	0.98	0.98	335
6	0.97	0.99	0.98	335
7	0.98	0.98	0.98	335
8	0.97	0.97	0.97	335
9	0.99	0.96	0.97	335
micro avg	0.98	0.98	0.98	3350
macro avg	0.98	0.98	0.98	3350
weighted avg	0.98	0.98	0.98	3350
samples avg	0.98	0.98	0.98	3350

```
Evaluando la parte de Test: Coste = 0.35749665272958353, ↩
   Precision =92.54545569419861
```

4.2.7. Conclusión

La **precisión más alta y coste más bajo** que he conseguido sobre los elementos de entrenamiento ha sido la del **modo 6**

(Coste = 0.35749665272958353,Precisión =92.54545569419861)
La cual es muy alta y aceptable, y aunque seguramente se podría alcanzar una precisión mayor, pero no he sido capaz de afinarlo más aún.

4.3. Código

Se ha importado **displayData.py** para poder pintar los números.

```
import helper as help
from scipy.io import loadmat
from displayData import displayData
from keras.models import Sequential
from keras.optimizers import Adam
from keras.layers import Dense
from sklearn.metrics import classification_report
from keras.utils.np_utils import to_categorical
from sklearn.model_selection import train_test_split

def problemaX(num, model, X, y, ver, ep, y_noCat, X_test, y_test):
    history = model.fit(x=X, y=y, verbose=0, epochs=ep)
    help.plot_loss_accuracy(history)
    help.plt.close(2)
    help.plt.savefig("problema"+str(num)+"-"+str(ver)+"-graf1.png")
    help.plot_confusion_matrix(model, X, y_noCat)
    help.plt.savefig("problema"+str(num)+"-"+str(ver)+"-graf3.png")

    pred= (model.predict(X)>0.5).astype(int)
    print(classification_report(y, pred))
    help.plt.show()

    [test_cost, test_acc] = model.evaluate(X_test, y_test,verbose=0)
    print("Evaluando la parte de Test: Coste = "+str(test_cost)+", Precision ="+str(test_acc*100))

def menu():
    print("---MENU---\n")
    print("Ejercicios:\n")
    print("(1) - [4-tanh]-[2-tanh]-[10-softmax]")
    print("(2) - [64-tanh]-[32-tanh]-[16-softmax]-[10-softmax]")
    print("(3) - [64-tanh]-[32-tanh]-[16-softmax]-[10-sigmoid]")
    print("(4) - [64-tanh]-[32-tanh]-[16-tanh]-[10-sigmoid]")
    print("(5) - [64-tanh]-[32-tanh]-[16-tanh]-[10-softmax]")
    print("(6) - [128-tanh]-[64-tanh]-[32-tanh]-[10-softmax]")
    print("(0) - Salir")

    return int(input("Seleccione ejercicio a calcular (numero):")
```

```
        )

def main():
    data = loadmat ('numbers.mat')
    y = data ['y']
    X = data ['X']
    y[y == 10] = 0
    sample = help.np.random.choice(X.shape[0], 100)
    fig, ax = displayData(X[sample, :])
    fig.savefig('numeros.png')

    X_train , X_test , y_train , y_test = train_test_split(X, y ,↩
       test_size=0.33 , stratify=y)
    y_cat_train = to_categorical(y_train)
    y_cat_test = to_categorical(y_test)

    op = menu()
    if op==0:
        print("Saliendo...")
    elif op==1:
        # [4-tanh]-[2-tanh]-[10-softmax]
        model = Sequential()
        model.add(Dense(units=4, input_shape=(400,), activation='↩
           tanh'))
        model.add(Dense(units=2, activation='tanh'))
        model.add(Dense(units=10, activation='softmax'))
        model.compile(Adam(lr=0.01), loss='↩
           categorical_crossentropy', metrics=['acc'])

    elif op==2:
        # [64-tanh]-[32-tanh]-[16-softmax]-[10-softmax]
        model = Sequential()
        model.add(Dense(units=64, input_shape=(400,), activation=↩
           'tanh'))
        model.add(Dense(units=32, activation='tanh'))
        model.add(Dense(units=16, activation='softmax'))
        model.add(Dense(units=10, activation='softmax'))
        model.compile(Adam(lr=0.01), loss='↩
           categorical_crossentropy', metrics=['acc'])

    elif  op==3:
        # [64-tanh]-[32-tanh]-[16-softmax]-[10-sigmoid]
        model = Sequential()
        model.add(Dense(units=64, input_shape=(400,), activation=↩
           'tanh'))
```

```
76          model.add(Dense(units=32, activation='tanh'))
77          model.add(Dense(units=16, activation='softmax'))
78          model.add(Dense(units=10, activation='sigmoid'))
79          model.compile(Adam(lr=0.01), loss='↩
               categorical_crossentropy', metrics=['acc'])
80
81      elif  op==4:
82          # [64-tanh]-[32-tanh]-[16-tanh]-[10-sigmoid]
83          model = Sequential()
84          model.add(Dense(units=64, input_shape=(400,), activation=↩
               'tanh'))
85          model.add(Dense(units=32, activation='tanh'))
86          model.add(Dense(units=16, activation='tanh'))
87          model.add(Dense(units=10, activation='sigmoid'))
88          model.compile(Adam(lr=0.01), loss='↩
               categorical_crossentropy', metrics=['acc'])
89
90      elif  op==5:
91          # [64-tanh]-[32-tanh]-[16-tanh]-[10-softmax]
92          model = Sequential()
93          model.add(Dense(units=64, input_shape=(400,), activation=↩
               'tanh'))
94          model.add(Dense(units=32, activation='tanh'))
95          model.add(Dense(units=16, activation='tanh'))
96          model.add(Dense(units=10, activation='softmax'))
97          model.compile(Adam(lr=0.01), loss='↩
               categorical_crossentropy', metrics=['acc'])
98
99      elif  op==6:
100         # [128-tanh]-[64-tanh]-[32-tanh]-[10-softmax]
101         model = Sequential()
102         model.add(Dense(units=128, input_shape=(400,), activation↩
               ='tanh'))
103         model.add(Dense(units=64, activation='tanh'))
104         model.add(Dense(units=32, activation='tanh'))
105         model.add(Dense(units=10, activation='softmax'))
106         model.compile(Adam(lr=0.01), loss='↩
               categorical_crossentropy', metrics=['acc'])
107
108     problemaX(5,model, X_train, y_cat_train,op,100, y_train, ↩
           X_test, y_cat_test)
109
110 main()
```

Código parte 2: p62.py