**Redes Neuronales.**

**Repaso de Evaluación de modelos.**

**Objetivos**

Estos ejercicios tienen como objetivo que te familiarices con la librería Keras implementando modelos simples de Redes Neuronales, y adquieras confianza interpretando las métricas de evaluación de modelos.

**Preparación**

Desde una terminal de Anaconda ubicarse en el directorio de la práctica donde se encuentra los archivos con extensión “.ipynb” (notebooks) y escribir: “jupyter notebook”. Desde el entorno web, acceder a la dirección del servidor (generalmente <http://localhost:8888/>) para abrir los archivos.

**Ejercicio 1** En este ejercicio, haremos lo mismo que en el 2 de la práctica de **Regresión Logística**, pero ahora utilizando redes neuronales, es decir, varias capas. Probá variando la cantidad de capas y la función de activación de cada una.

Anotá los mejores resultados y la topología.

Podés escribir la topología de una red de tres capas, donde la primera tiene activación ReLU y 3 salidas, la segunda TanH y 5 salidas, y la tercera softmax y 2 salidas como:  
Dense(3,’relu’) - Dense(5,’tanh’) - Dense(2,’softmax’).

***Archivo*: Redes Neuronales con Keras y varios conjuntos de datos.ipynb**

C*onjuntos de datos*:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Archivo | Mejor accuracy | Mejor loss | Iteraciones | Tasa de aprendizaje | Topología |
| 2\_clases\_simple.csv | 1 | 0.0834 | 100 | 0.001 | Relu3-tanh5-softmax |
| 6\_clases\_dificil.csv | 0.9933 | 0.0172 | 2000 | 0.01 | Relu32-tanh48-softmax |
| circulos.csv | 1 | 0.0858 | 1000 | 0.001 | Relu3-relu5-softmax |
| diabetes.csv | 1 | 0.0010 | 2000 | 0.001 | Relu32-tanh48-softmax |
| iris.csv | 1 | 1.60e-4 | 2000 | 0.01 | Relu32-tanh48-softmax |
| moons\_mis-scaled.csv | 0.9850 | 0.0573 | 2000 | 0.001 | Relu32-tanh48-softmax |
| moons.csv | 1 | 0.093 | 2000 | 0.01 | Relu32-tanh48-softmax |

**Ejercicio 2 (Opcional) .** Implementá varias de las métricas vistas en clase utilizando NumPy.

**Archivo: Métricas de clasificación.ipynb**

**Ejercicio 3**. Entrená un modelo de redes neuronales que clasifique correctamente las clases en el archivo **6\_clases\_dificil.csv***.* Realizá un análisis de las diferentes métricas vistas en clase. Podés utilizar las funciones definidas en *AAPutils.py* para realizar más rápidamente los gráficos y reportes. A continuación se muestra un breve glosario de algunas funciones que te serán de utilidad.

|  |  |
| --- | --- |
| AAPutils.py | |
| X\_train, X\_test, Y\_train, Y\_test = dividir\_train\_test(X, Y, test\_size) | Divide el dataset de forma aleatoria en dos conjuntos. El parámetro “test\_ size” es un número entre 0 y 1 que indica el porcentaje a utilizar para el conjunto de evaluación. |
| plot\_frontera\_de\_decision\_2D(model, X\_train, Y\_train, X\_test, Y\_test) | Grafica en 2D el dataset junto con la frontera de decisión del modelo. Al pasar el conjunto de testeo se visualizan los dos conjuntos con diferentes marcadores. |
| plot\_training\_curves(history , acc=True) | Grafica las curvas de entrenamiento (loss) para el training y testing set. Si el parámetro “acc” es True grafica también la curva para el Accuracy. |
| plot\_confusion\_matrix(y\_true, y\_pred) | Grafica la matriz de confusión para los vectores de clases pasados como argumento. |
| print\_classification\_report(y\_true, y\_pred) | Imprime en consola un reporte del accuracy para los vectores de clases pasados como argumento. Si la clasificación es binaria se imprime además el precision, recall y f-measure. |
| plot\_ROC\_curve(modelo, x, y) | Grafica las curvas ROC y precision-recall para el modelo entrenado y para el conjunto pasado como argumento. |

**Ejercicio 4.** Realizá el mismo procedimiento que el ejercicio anterior, pero para los datasets descriptos a continuación. Recorda que las redes neuronales son sensibles a la escala de los datos. Intentá encontrar un balance adecuado entre precision y recall utilizando el parámetro class\_weight del método [fit](https://keras.io/api/models/model_training_apis/).

|  |  |
| --- | --- |
| Titanic.csv | Información sobre diferentes pasajeros del Titanic indicando si sobrevivió o no al viaje. |
| Diabetes.csv | Dataset con información sobre pacientes, indicando si cada uno tiene o no diabetes gestacional. |
| Ecoli.csv | Información de proteínas de pacientes, indicando si presenta o no una enfermedad. |

En los conjuntos de Diabetes y Ecoli, ¿cuál es el *precision* más alto que podés obtener, manteniendo el recall cercano a 1?

**Ejercicio 5** (*opcional,* solo si no tenés experiencia con imágenes en numpy). Los notebooks de Jupyter **Tutorial de numpy.ipynb** y **Tutorial de procesamiento de imágenes.ipynb** te introducen a como manipular matrices de numpy, en particular aquellas que representan imágenes.