|  |
| --- |
| Asignatura: |
| Inteligencia Artificial II |

|  |
| --- |
| Título del documento: |
| **Laboratorio 2:**  **Implementación de modelos neuronales**  **supervisados** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | Fecha |
| Grupo: |  | 2 |  | 6/03/18 |
|  |  |  |  |  |
| Miembros: | 1- | Gonzalo de las Heras |  |  |
|  |  |  |  |  |
|  | 2- | Jorge de la Fuente |  |  |
|  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nombre de fichero: |  | Fecha: |  |  |  |  |
| LAB02-GRUPO02-MEMORIA.docx **(.pdf)** |  | **6/03/18** |  |  |  |  |

Índice

[1 Laboratorio 2. Implementación de modelos neuronales supervisados 5](#_Toc508136343)

[1.1 Práctica 1. Fundamentos de redes neuronales 5](#_Toc508136344)

[1.1.1 Objetivo 5](#_Toc508136345)

[1.1.2 Cuestiones 5](#_Toc508136346)

[1.1.2.1 Cuestión 1 5](#_Toc508136347)

[1.1.2.2 Cuestión 2 5](#_Toc508136348)

[1.2 Práctica 2. MLP con Keras 5](#_Toc508136349)

[1.2.1 Objetivo 5](#_Toc508136350)

[1.2.2 Cuestiones 6](#_Toc508136351)

[1.2.2.1 Cuestión 1 6](#_Toc508136352)

[1.2.2.2 Cuestión 2 6](#_Toc508136353)

[1.3 Práctica 3. Juego de Tronos con Keras 6](#_Toc508136354)

[1.3.1 Objetivo 6](#_Toc508136355)

[1.3.2 Cuestiones 6](#_Toc508136356)

[1.3.2.1 Cuestión 1 6](#_Toc508136357)

[1.3.2.2 Cuestión 2 6](#_Toc508136358)

[2 Resultados 7](#_Toc508136359)

[2.1 Práctica 1. Fundamentos de redes neuronales 7](#_Toc508136360)

[2.1.1 Cuestión 1 7](#_Toc508136361)

[2.1.2 Cuestión 2 8](#_Toc508136362)

[2.2 Práctica 2. MLP con Keras 10](#_Toc508136363)

[2.2.1 Cuestión 1 10](#_Toc508136364)

[2.2.2 Cuestión 2 12](#_Toc508136365)

[2.3 Práctica 3. MLP con Keras 15](#_Toc508136366)

[2.3.1 Cuestión 1 15](#_Toc508136367)

[3 Discusión general 18](#_Toc508136368)

[3.1 Práctica 1. Fundamentos de redes neuronales 18](#_Toc508136369)

[3.1.1 Cuestión 1 18](#_Toc508136370)

[3.1.2 Cuestión 2 18](#_Toc508136371)

[3.2 Práctica 2. MLP con Keras 20](#_Toc508136372)

[3.2.1 Cuestión 1 20](#_Toc508136373)

[3.2.2 Cuestión 2 21](#_Toc508136374)

[3.3 Práctica 3. Juego de Tronos con Keras 22](#_Toc508136375)

[3.3.1 Cuestión 1 22](#_Toc508136376)

[3.3.2 Cuestión 2 23](#_Toc508136377)

[4 Reflexiones finales 24](#_Toc508136378)

[5 Bibliografía 25](#_Toc508136379)

[6 ANEXO A. Documentos adjuntos 26](#_Toc508136380)

Índice de ilustraciones

[Ilustración 2‑1: Variación de loss con respecto a iteraciones Práctica 2 - Cuestión 1 (Adam). 11](#_Toc508136381)

[Ilustración 2‑2: Variación de loss con respecto a iteraciones Práctica 2 - Cuestión 1 (SGD). 11](#_Toc508136382)

[Ilustración 2‑3: Variación de loss con respecto a iteraciones Práctica 2 - Cuestión 2 (Adam). 14](#_Toc508136383)

[Ilustración 2‑4: Variación de loss con respecto a iteraciones Práctica 2 - Cuestión 2 (SGD). 14](#_Toc508136384)

[Ilustración 2‑5: Variación de loss con respecto a iteraciones Práctica 3. 15](#_Toc508136385)

[Ilustración 2‑6: Variación de loss con respecto a iteraciones Práctica 3 (red sin IsPopular). 16](#_Toc508136386)

[Ilustración 3‑1: Representación gráfica de la puerta lógica OR. 18](#_Toc508136387)

[Ilustración 3‑2: Representación gráfica de la puerta lógica XOR. 19](#_Toc508136388)

[Ilustración 3‑3: Representación gráfica del hiperplano generado (OR). 19](#_Toc508136389)

Índice de tablas

[Tabla 2‑1: Resultados Práctica 1 - Cuestión 1. 7](#_Toc508136390)

[Tabla 2‑2: Resultados entrenamiento perceptrón Práctica 1 - Cuestión 2 (parte 1). 8](#_Toc508136391)

[Tabla 2‑3: Resultados entrenamiento perceptrón Práctica 1 - Cuestión 2 (parte 2). 9](#_Toc508136392)

[Tabla 2‑4: Resultados entrenamiento MLP para resolver XOR Práctica 2 - Cuestión 1 (parte 1). 10](#_Toc508136393)

[Tabla 2‑5: Resultados entrenamiento MLP para resolver XOR Práctica 2 - Cuestión 1 (parte 1). 11](#_Toc508136394)

[Tabla 2‑6: Resultados entrenamiento MLP Práctica 2 - Cuestión 2 (Parte 1). 12](#_Toc508136395)

[Tabla 2‑7: Resultados entrenamiento MLP Práctica 2 - Cuestión 2 (Parte 1). 13](#_Toc508136396)

[Tabla 2‑8: Media de los pesos de la red MLP Práctica 3. 15](#_Toc508136397)

[Tabla 2‑9: Resultados predicción Práctica 3. 16](#_Toc508136398)

[Tabla 2‑10: Resultados predicción Práctica 3 (red sin IsPopular). 16](#_Toc508136399)

[Tabla 2‑11: Resultados predicción Práctica 3 (dato numDeadRelations de Daenerys modificado). 17](#_Toc508136400)

Listado de abreviaturas

|  |  |
| --- | --- |
| MLP | *MultiLayer Perceptron* (Perceptrón MultiCapa) |
| PMC | Perceptrón MultiCapa |
| GoT | Game of Thrones (Juego de Tronos) |
| SGD | Stochastic Gradient Descent |
|  |  |

# Laboratorio 2. Implementación de modelos neuronales supervisados

Desarrollar de forma autónoma distintas implementaciones de redes neuronales de aprendizaje supervisado que permitan resolver distintos casos de uso. La práctica comenzará con la implementación más sencilla del funcionamiento de una neurona sencilla y terminará con la construcción de un MLP capaz de hacer predicciones para un caso concreto planteado.

## Práctica 1. Fundamentos de redes neuronales

### Objetivo

Implementación de una red neuronal para resolver las funciones OR y XOR. La implementación se hará paso a paso, correspondiendo con los distintos apartados de la práctica. Como es necesario probar y testear el código según se desarrolla, utiliza los archivos propuestos en el apartado “Implementación” de esta práctica y responde a las preguntas que se plantean en “Cuestiones”. Debes de usar comentarios con profusión, incluyendo celdas específicas donde expliques los algoritmos que usas y el código que has programas. Cuantos más comentarios haya, mejor será la evaluación y, probablemente, menos preguntas serán necesarias en la defensa de la práctica.

### Cuestiones

#### Cuestión 1

Teniendo en cuenta los valores de salida del apartado 2 ¿Podemos saber cuál es la red neuronal que mejor predice? ¿Qué modelo predice mejor para la función ‘ReLU’? ¿Y para la ‘sigmoide’?

#### Cuestión 2

Recoge los resultados de las dos pruebas del apartado 3 en sendas tablas y discútelas en función de lo visto en teoría. Para los casos en los que sea posible, escribe de forma explícita la ecuación del hiperplano que se ha generado.

## Práctica 2. MLP con Keras

### Objetivo

Utiliza la librería Keras para construir y entrenar un MLP que resuelva problemas no lineales. En vez de usar la Regla Delta Generalizada, usaremos Adam como función de modificación de matriz de pesos (optimizer) de la forma que se indica en el apartado de “Implementación”. Responde a las preguntas que se plantean en “Cuestiones”.

### Cuestiones

#### Cuestión 1

Crea una tabla con los resultados del MLP que resuelve la función XOR que incluya el número de neuronas de la capa oculta, el número de epochs, los valores de salida y el último valor error para dicho learning rate. Cada fila de la tabla será una de las variaciones anteriores. Crear otro modelo cambiando las funciones de activación.

#### Cuestión 2

Crea una tabla con los resultados del MLP que resuelve la función XOR que incluya el número de neuronas de la capa oculta, el número de epochs, los valores de salida y el último valor error para dicho learning rate. Cada fila de la tabla será una de las variaciones anteriores. Crear otro modelo cambiando las funciones de activación.

## Práctica 3. Juego de Tronos con Keras

### Objetivo

Utiliza lo practicado en este laboratorio para resolver un caso práctico “real” usando la librería Keras y una arquitectura MLP para predecir la probabilidad de muerte de los personajes de Juego de Tronos (GoT).

### Cuestiones

#### Cuestión 1

Dibuja la arquitectura de red y escribe el ‘optimizer’ y las funciones de activación y de error usadas. Indica cual es el valor de error final y el valor de alive para todos los personajes. ¿Quién es el personaje con más posibilidades de morir?

#### Cuestión 2

Cambia el fichero predicciones\_got.csv para que sea otro el personaje con más probabilidades de morir y compruébalo. ¿Qué cambio has realizado? Justifícalo desde el punto de vista de las redes neuronales. ¿Todas las entradas son igual de relevantes? ¿Cómo lo puedes demostrar?

# Resultados

## Práctica 1. Fundamentos de redes neuronales

### Cuestión 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Red** | **Entradas** | **Pesos** | | | **Yr** | **Yd** | **Error** |
| **N1** | **N2** | **Salida** |
| ReLU 1 | 3 | 2 | 4 | 2 | 52 | 70 | 18 |
| 5 | 4 | -5 | 7 |
| ReLU 2 | 1 | 2 | 4 | 2 | 63 | 70 | 7 |
| -1 | 4 | -5 | 7 |
| Sigmoide 1 | 4 | -0,17 | 0,62 | 0,81 | 0,7034382608 | 1 | 0,2965617392 |
| 3 | 0,42 | 0,55 | 0,35 |
| Sigmoide 2 | -2 | -0,17 | 0,62 | 0,81 | 0,661236221502 | 1 | 0,338763778498 |
| 1 | 0,42 | 0,55 | 0,35 |

Tabla 2‑1: Resultados Práctica 1 - Cuestión 1.

### Cuestión 2

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Iteración** | **Entradas** | | **Pesos iniciales** | | **Salida real** | **Salida esperada** | **Error** | **Pesos finales** | |
| **X1** | **X2** | **W1** | **W2** | **W1** | **W2** |
| 1 | 0 | 0 | -0,7 | 0,7 | 0 | 0 | 0 | -0,7 | 0,7 |
| 0 | 1 | -0,7 | 0,7 | 1 | 1 | 0 | -0,7 | 0,7 |
| 1 | 0 | -0,7 | 0,7 | 0 | 1 | 1 | -0,6 | 0,7 |
| 1 | 1 | -0,6 | 0,7 | 0 | 1 | 1 | -0,5 | 0,8 |
| 2 | 0 | 0 | -0,5 | 0,8 | 0 | 0 | 0 | -0,5 | 0,8 |
| 0 | 1 | -0,5 | 0,8 | 1 | 1 | 0 | -0,5 | 0,8 |
| 1 | 0 | -0,5 | 0,8 | 0 | 1 | 1 | -0,4 | 0,8 |
| 1 | 1 | -0,4 | 0,8 | 1 | 1 | 0 | -0,4 | 0,8 |
| 3 | 0 | 0 | -0,4 | 0,8 | 0 | 0 | 0 | -0,4 | 0,8 |
| 0 | 1 | -0,4 | 0,8 | 1 | 1 | 0 | -0,4 | 0,8 |
| 1 | 0 | -0,4 | 0,8 | 0 | 1 | 1 | -0,3 | 0,8 |
| 1 | 1 | -0,3 | 0,8 | 1 | 1 | 0 | -0,3 | 0,8 |
| 4 | 0 | 0 | -0,3 | 0,8 | 0 | 0 | 0 | -0,3 | 0,8 |
| 0 | 1 | -0,3 | 0,8 | 1 | 1 | 0 | -0,3 | 0,8 |
| 1 | 0 | -0,3 | 0,8 | 0 | 1 | 1 | -0,2 | 0,8 |
| 1 | 1 | -0,7 | 0,7 | 0 | 0 | 0 | -0,7 | 0,7 |

Tabla 2‑2: Resultados entrenamiento perceptrón Práctica 1 - Cuestión 2 (parte 1).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Iteración** | **Entradas** | | **Pesos iniciales** | | **Salida real** | **Salida esperada** | **Error** | **Pesos finales** | |
| **X1** | **X2** | **W1** | **W2** | **W1** | **W2** |
| 5 | 0 | 0 | -0,2 | 0,8 | 0 | 0 | 0 | -0,2 | 0,8 |
| 0 | 1 | -0,2 | 0,8 | 1 | 1 | 0 | -0,2 | 0,8 |
| 1 | 0 | -0,2 | 0,8 | 0 | 1 | 1 | -0,1 | 0,8 |
| 1 | 1 | -0,1 | 0,8 | 1 | 1 | 0 | -0,1 | 0,8 |
| 6 | 0 | 0 | -0,1 | 0,8 | 0 | 0 | 0 | -0,1 | 0,8 |
| 0 | 1 | -0,1 | 0,8 | 1 | 1 | 0 | -0,1 | 0,8 |
| 1 | 0 | -0,1 | 0,8 | 0 | 1 | 1 | 0 | 0,8 |
| 1 | 1 | 0 | 0,8 | 1 | 1 | 0 | 0 | 0,8 |
| 7 | 0 | 0 | 0 | 0,8 | 0 | 0 | 0 | 0 | 0,8 |
| 0 | 1 | 0 | 0,8 | 1 | 1 | 0 | 0 | 0,8 |
| 1 | 0 | 0 | 0,8 | 0 | 1 | 1 | 0,1 | 0,8 |
| 1 | 1 | 0,1 | 0,8 | 1 | 1 | 0 | 0,1 | 0,8 |
| 8 | 0 | 0 | 0,1 | 0,8 | 0 | 0 | 0 | 0,1 | 0,8 |
| 0 | 1 | 0,1 | 0,8 | 1 | 1 | 0 | 0,1 | 0,8 |
| 1 | 0 | 0,1 | 0,8 | 0 | 1 | 1 | 0,2 | 0,8 |
| 1 | 1 | 0,2 | 0,8 | 1 | 1 | 0 | 0,2 | 0,8 |
| 9 | 0 | 0 | 0,2 | 0,8 | 0 | 0 | 0 | 0,2 | 0,8 |
| 0 | 1 | 0,2 | 0,8 | 1 | 1 | 0 | 0,2 | 0,8 |
| 1 | 0 | 0,2 | 0,8 | 1 | 1 | 0 | 0,2 | 0,8 |
| 1 | 1 | 0,2 | 0,8 | 1 | 1 | 0 | 0,2 | 0,8 |

Tabla 2‑3: Resultados entrenamiento perceptrón Práctica 1 - Cuestión 2 (parte 2).

## Práctica 2. MLP con Keras

### Cuestión 1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Optimizer** | **FA capa oculta** | **Coef. Aprendizaje** | **FA capa salida** | **F. de error** | **Neuronas capa oculta** | **Salida** | **Epochs** | **Score** |
| Adam | ReLU | 0,1 | Sigmoide | Mean Squared Error | 5 | 0,500322 | 60 | 0,2500006 |
| 0,499617 |
| 0,499428 |
| 0,498723 |
| Adam | ReLU | 0,1 | Sigmoide | Mean Squared Error | 10 | 0,036287 | 60 | 0,0003441 |
| 0,995387 |
| 0,99384 |
| 0,00077668 |
| Adam | ReLU | 0,1 | Sigmoide | Mean Squared Error | 15 | 0,000608481 | 60 | 0,0000024 |
| 0,997763 |
| 0,999616 |
| 0,00200143 |
| SGD | ReLU | 0,1 | Sigmoide | Mean Squared Error | 15 | 0,0870572 | 2000 | 0,0036450 |
| 0,945564 |
| 0,950399 |
| 0,0397183 |

Tabla 2‑4: Resultados entrenamiento MLP para resolver XOR Práctica 2 - Cuestión 1 (parte 1).

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Optimizer** | **FA capa oculta** | **Coef. Aprendizaje** | **FA capa salida** | **F. de error** | **Neuronas capa oculta** | **Salida** | **Epochs** | **Score** |
| SGD | ReLU | 0,1 | Sigmoide | Mean Squared Error | 25 | 0,0743679 | 2000 | 0,0030604 |
| 0,948539 |
| 0,952804 |
| 0,0428388 |
| SGD | ReLU | 0,1 | Sigmoide | Mean Squared Error | 35 | 0,0705402 | 2000 | 0,0027301 |
| 0,953185 |
| 0,954757 |
| 0,0413044 |

Tabla 2‑5: Resultados entrenamiento MLP para resolver XOR Práctica 2 - Cuestión 1 (parte 1).

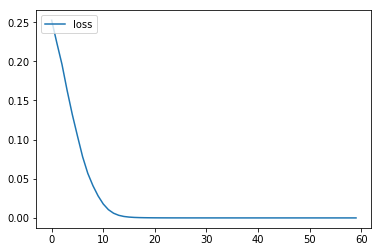


Ilustración 2‑1: Variación de loss con respecto a iteraciones Práctica 2 - Cuestión 1 (Adam).

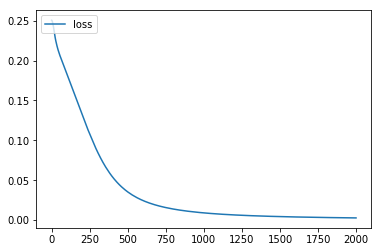


Ilustración 2‑2: Variación de loss con respecto a iteraciones Práctica 2 - Cuestión 1 (SGD).

### Cuestión 2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Optimizer** | **FA capa oculta** | **Coef. Aprendizaje** | **FA capa salida** | **F. de error** | **Neuronas capa oculta** | **Epochs** | **Error** |
| Adam | ReLU | 0,0000001 | Lineal | Mean Squared Error | 80 | 200 | 347,6674661 |
| Adam | ReLU | 0,01 | Lineal | Mean Squared Error | 80 | 200 | 166,1321063 |
| Adam | Lineal | 0,1 | ReLU | Mean Squared Error | 80 | 200 | 677,1621030 |
| Adam | Lineal | 1 | ReLU | Mean Squared Error | 80 | 200 | 677,1621030 |

Tabla 2‑6: Resultados entrenamiento MLP Práctica 2 - Cuestión 2 (Parte 1).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Optimizer** | **FA capa oculta** | **Coef. Aprendizaje** | **FA capa salida** | **F. de error** | **Neuronas capa oculta** | **Epochs** | **Error** |
| SGD | Lineal | 0,0000001 | ReLU | Mean Squared Error | 80 | 200 | 228,8709050 |
|
|
|
| SGD | Lineal | 0,01 | ReLU | Mean Squared Error | 80 | 200 | nan |
|
|
|
| SGD | ReLU | 0,1 | ReLU | Mean Squared Error | 80 | 200 | nan |
| SGD | ReLU | 1 | ReLU | Mean Squared Error | 80 | 200 | nan |

Tabla 2‑7: Resultados entrenamiento MLP Práctica 2 - Cuestión 2 (Parte 1).

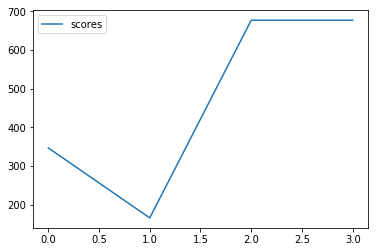


Ilustración 2‑3: Variación de loss con respecto a iteraciones Práctica 2 - Cuestión 2 (Adam).

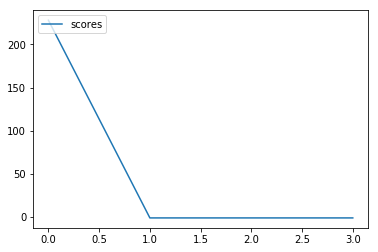


Ilustración 2‑4: Variación de loss con respecto a iteraciones Práctica 2 - Cuestión 2 (SGD).

## Práctica 3. MLP con Keras

### Cuestión 1

|  |  |  |
| --- | --- | --- |
| **Característica** | **Valor del peso** | **Rango de valores** |
| Male | -0,03003057 | {0, 1} |
| Book1 | -0,04740242 | {0, 1} |
| Book2 | -0,03694244 | {0, 1} |
| Book3 | -0,0261386 | {0, 1} |
| Book4 | 0,00123119 | {0, 1} |
| Book5 | -0,03290815 | {0, 1} |
| IsMarried | -0,02670936 | {0, 1} |
| IsNoble | -0,00386382 | {0, 1} |
| numDeadRelations | -0,05936071 | [0, ∞) |
| isPopular | 0,0004857 | {0, 1} |

Tabla 2‑8: Media de los pesos de la red MLP Práctica 3.



Ilustración 2‑5: Variación de loss con respecto a iteraciones Práctica 3.

|  |  |  |
| --- | --- | --- |
| **Posición** | **Personaje** | **Probabilidad de supervivencia** |
| 1º | Coldhands | 77,97% |
| 2º | Othell Yarwyck | 59,32% |
| 3º | Roland Crakehall (Kingsguard) | 47,07% |
| 4º | Daenerys Targaryen | 11,61% |
| 5º | Tommen Baratheon | 7,07% |

Tabla 2‑9: Resultados predicción Práctica 3.

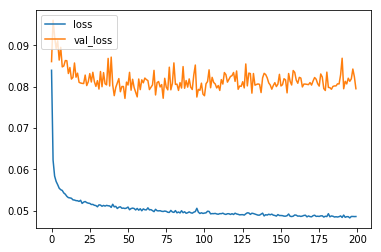


Ilustración 2‑6: Variación de loss con respecto a iteraciones Práctica 3 (red sin IsPopular).

|  |  |  |
| --- | --- | --- |
| **Posición** | **Personaje** | **Probabilidad de supervivencia** |
| 1º | Coldhands | 75,17% |
| 2º | Othell Yarwyck | 52,06% |
| 3º | Roland Crakehall (Kingsguard) | 42,43% |
| 4º | Daenerys Targaryen | 16,12% |
| 5º | Tommen Baratheon | 7,39% |

Tabla 2‑10: Resultados predicción Práctica 3 (red sin IsPopular).

|  |  |  |
| --- | --- | --- |
| **Posición** | **Personaje** | **Probabilidad de supervivencia** |
| 1º | Coldhands | 77,97% |
| 2º | Othell Yarwyck | 59,32% |
| 3º | Roland Crakehall (Kingsguard) | 47,07% |
| 4º | Daenerys Targaryen | 44,81% |
| 5º | Tommen Baratheon | 7,07% |

Tabla 2‑11: Resultados predicción Práctica 3 (dato numDeadRelations de Daenerys modificado).

# Discusión general

## Práctica 1. Fundamentos de redes neuronales

### Cuestión 1

Tras analizar los datos provistos para la construcción de la red neuronal (ficheros ‘rn1\_relu.csv’, ‘rn2\_relu.csv’, ‘rn1\_sigm.csv’ y ‘rn2\_sigm.csv’) hay que percatarse de que las redes con misma función de activación son iguales ya que tienen los mismos pesos exactos. Solo cambia la entrada, por lo que, en realidad, están prediciendo dos valores distintos y no tiene sentido compararlos emitiendo un juicio de cual es mejor.

Los datos obtenidos en este ejercicio se pueden consultar en la tabla 2-1.

### Cuestión 2

En el caso de la puerta lógica OR, nos encontramos con un problema linealmente separable. Un problema linealmente separable es aquel en el que existe un hiperplano que divide el espacio en dos regiones excluyentes, es decir, son clasificables. Este hecho, provoca directamente que el problema es resoluble por un perceptrón entrenado correctamente en un número finito de pasos.

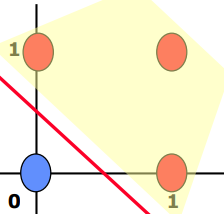


Ilustración 3‑1: Representación gráfica de la puerta lógica OR.

Los datos resultantes del entrenamiento se pueden consultar en las tablas 2-2 y 2-3.

En el segundo caso propuesto, de una puerta XOR, el problema no es linealmente separable. De modo que el problema no es resoluble por un perceptrón, ni con infinitos pasos de entrenamiento. Por este motivo, no se ha realizado el entrenamiento.

Sin embargo, este tipo de problemas se pueden resolver por un perceptrón multicapa (MLP) aunque existe un error, que se intenta minimizar, pero que sigue estando ahí.

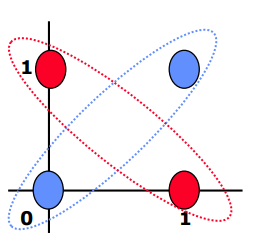


Ilustración 3‑2: Representación gráfica de la puerta lógica XOR.

Ecuación del hiperplano (OR):

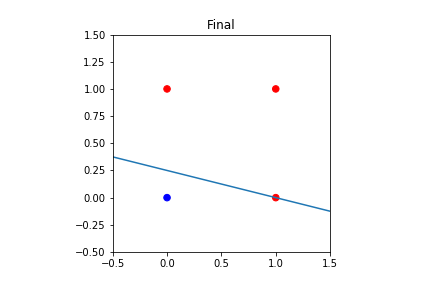


Ilustración 3‑3: Representación gráfica del hiperplano generado (OR).

## Práctica 2. MLP con Keras

### Cuestión 1

Para la implementación de una red neuronal que haga de predictor de la puerta lógica XOR, se ha diseñado un MLP con las siguientes características:

* Capa de entrada: 2 elementos.
* Capa oculta: 15 (Adam) y 35 (SGD) neuronas. En este caso, como nuestro conjunto de entrenamiento es igual al conjunto de posibles entradas, no tenemos problema de que la red neuronal memorice las entradas pues se contemplan todas las opciones.
* Función de error: Mean Squared Error.
* Función de activación de la capa oculta: ReLU (requisito del enunciado).
* Función de activación de la capa de salida: Sigmoide (requisito del enunciado y del problema a resolver).
* Iteraciones: 60 (Adam) y 2000 (SGD).
* Coeficiente de aprendizaje: 0,1 (Adam) y 0,01 (SGD).

Para el primer modelo, se ha usado el optimizador Adam, con el coeficiente de aprendizaje 0.1 y el resto de los hiperparámetros por defecto de Keras.

Para el segundo, se ha usado el optimizador SGD (Stochastic Gradient Descent), con el coeficiente de aprendizaje 0.1 y el resto de los parámetros por defecto de Keras.

Para cada modelo se han realizado 3 entrenamientos y los resultados pueden consultarse en las tablas 2.4 y 2.5.

Con los datos obtenidos, constatamos la superioridad del optimizador Adam con respecto a SGD, al proporcionar mejores predicciones con menos neuronas.

### Cuestión 2

Para la implementación de una red neuronal que haga de predictor de los datos suministrados en los ficheros ‘p2\_entrenamiento.csv’ y ‘p2\_objetivos.csv’, se ha diseñado un MLP con las siguientes características:

* Capa de entrada: 9 elementos.
* Capa oculta: 80 neuronas.
* Función de error: Mean Squared Error.
* Función de activación de la capa oculta: ReLU o Lineal.
* Función de activación de la capa de salida: ReLU o Lineal.
* Iteraciones: 200.
* Conjunto de entrenamiento: 80% de los datos proporcionados.
* Conjunto de validación: 20% de los datos proporcionados.
* Coeficiente de aprendizaje: 0,0000001, 0,01, 0,1 y 1 (requisito del enunciado).

Se han realizado distintos modelos alterando las funciones de activación, los optimizadores y los coeficientes de aprendizaje.

Los resultados pueden consultarse en las tablas 2.6 y 2.7.

Al tratar de predecir valores aleatorios es normal que el score de las redes sea muy alto (cuando más alto peor).

Como detalle, nos fijamos que un coeficiente de aprendizaje alto produce más error al ir alternando entre pendientes de la curva de error, pero nunca llegando a descender hasta un mínimo.

## Práctica 3. Juego de Tronos con Keras

### Cuestión 1

Para el proceso de búsqueda el mejor predictor, se ha empleado el método de validación cruzada. Para ello, se ha empleado el 90% de los datos proporcionados, como conjunto de entrenamiento, y el 10% restante, como validación.

Después del proceso de experimentación, el predictor más eficiente conseguido tiene entorno a un es score de 0,0838 (valor conseguido con el conjunto de validación). La red neuronal consta de las siguientes características:

* Capa de entrada: 10 elementos (los requeridos por el enunciado).
* Capa de oculta: 260 neuronas.
* Capa de salida: 1 neurona.
* Función de error: Mean Squared Error.
* Función de activación de la capa oculta: ReLU.
* Función de activación de la capa de salida: Sigmoide.
* Iteraciones: 200.
* Optimizador: Adam.
* Coeficiente de aprendizaje: 0,001.
* El resto de hiperparámetros del optimizador Adam son los propuestos por defecto por la librería Keras:
  + B1: 0,9.
  + B2: 0,999
  + E: 1e-08.
  + Decay: 0.

Los resultados de la predicción se pueden consultar en la tabla 2.9.

### Cuestión 2

Durante el proceso de aprendizaje, el MLP modifica los pesos de toda la red (aprendizaje global) para minimizar el error. De este modo, algunos pesos son inevitablemente más altos que otros y, además, algunos se acercan más al valor 0. Esto quiere decir, que existen entradas más relevantes para el cálculo del valor de salida de la neurona, que otros. Aquellos pesos que tiendan más a 0, serán menos relevantes, pues en la suma ponderada tendrán poca influencia. Por el contrario, para aquellos valores más altos, más influencia en la suma ponderada y más relevancia tendrá esa información para la predicción.

Para encontrar los pesos más y menos relevantes, se procederá al cálculo de la media de los pesos que salen cada entrada y al estudio comparativo de estos valores. Los resultados se pueden consultar en la tabla 2.8.

Además de los pesos, también hay que estudiar que rango de valores tiene cada entrada. De modo que, aunque el peso correspondiente a la característica “Book1”, por ejemplo, es alto con respecto al resto de pesos, su rango de valores es {0, 1} por lo que a la suma ponderada aportará como máximo el valor del peso (-0,04740242). Esto es así es todos los pesos menos en numDeadRelations. Esta última entrada, tiene como rango de valores [0, ∞), lo que significa que, en la suma ponderada, en cuanto la entrada sea mayor que 1, va a ser la característica que más va a aportar (aparte de que sea el peso más alejado de 0, -0,05936071).

Para demostrarlo, se ha modificado el valor de numDeadRelations para el personaje Daenerys Targaryen en el fichero de predicciones (15 en el fichero original, 0 en el modificado) y se ha realizado la predicción. El porcentaje de supervivencia con del fichero de datos sin alterar es de 11,61% y el porcentaje del fichero alterado es de 44,81%. Por lo tanto, deducimos que los personajes con más personas de su entorno que han muerto tienen más probabilidad de morir. Los resultados se pueden consultar en la tabla 2.11.

Por otra parte, la característica que tiene menos relevancia es IsPopular, con un peso de 0,0004857 y un rango de valores de {0, 1}.

Como demostración, se ha entrenado una red sin la característica IsPopular y como se puede apreciar en los resultados (tabla 2.10), las probabilidades de supervivencia apenas varían.

# Reflexiones finales

* Es importante una buena modelización del problema que se pretende resolver para diseñar correctamente la red neuronal que lo resuelva.
* Los datos provistos a la red neuronal no siempre son útiles para la predicción. Si previamente se eliminaran estos datos irrelevantes, el rendimiento del predictor mejoraría.
* Por mucho que se entrene la red en un MLP, siempre existirá un error, mínimo (local o global, dentro de la superficie de error) pero insalvable.
* La búsqueda de la mejor red neuronal es un proceso de experimentación.

# Bibliografía

**Brownlee, Jason.** Machine Learning Mastery. *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning.* [En línea] https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/.

**Chollet, François.** Keras: The Python Deep Learning library. [En línea] https://keras.io/.

**García-Tejedor, Álvaro.** Moodle UFV. Recursos de la asignatura. [En línea] http://moodleufv.ufv.es/moodle/.

# ANEXO A. Documentos adjuntos

* LAB02-GRUPO02-MEMORIA.docx (presente documento).
* Datos
* Práctica 1 – Fundamentos
  + Evolucion.mp4
  + L2P1\_apartado3\_OR.csv
  + L2P1-RRNN.ipnb
  + red\_neuronal.png
* Práctica 2 – Keras
  + L2P2-MLP.ipnb
* Práctica 3 – GOT
  + L2P3-GoT.ipnb