Práctica 1: FIFA Players Classification

Daniel Carmona Pedrajas

1 Arquitectura 1: One Layer Neural Network

La primera arquitectura que usaremos es tan simple como:

1. Capa densa con 512 neuronas.

1.1 Experimento 1: Configuración base arbitraria

Usamos la siguiente configuración:

Epochs	**Learning rate**	**Batch size**	**Activation**	**Loss**	**Optimizer**	**Regularization**
100	0.1	512	ReLU	Categorical Crossentropy	SGD	None

Table 1: Hiperparámetros para el Experimento 1 de la Arquitectura 1

Y entrenamos 5 veces para obtener los siguientes resultados:

	Train accuracy (%)	**Validation accuracy (%)**	**Bias (%)**	**Variance (%)**	**Training time (s)**
Mean	79.38	77.94	15.61	1.44	14
Std	0.05	0.14	0.05	0.19	0

Table 2: Resultados del Experimento 1 de la Arquitectura 1

Tener un *bias* alto y una *variance* baja significa que hay margen de mejora antes de llegar al *overfitting* y hay varias posibilidades para conseguir una mejor *accuracy*: añadir más neuronas, entrenar con más *epochs*, ...

1.2 Experimento 2: Aumentamos *epochs*

Tras el experimento anterior, nos decantamos por entrenar el modelo durante más *epochs* para reducir el *bias* usando la misma configuración.

Epochs	**Learning rate**	**Batch size**	**Activation**	**Loss**	**Optimizer**	**Regularization**
1000	0.1	512	ReLU	Categorical Crossentropy	SGD	None

Table 3: Hiperparámetros para el Experimento 2 de la Arquitectura 1

Tras 5 entrenamientos obtenemos los siguientes resultados: Con respecto al experimento anterior hemos

	Train accuracy (%)	**Validation accuracy (%)**	**Bias (%)**	**Variance (%)**	**Training time (s)**
Mean	84.02	81.5	10.97	2.47	199
Std	0.03	0.17	0.03	0.18	8.8

Table 4: Resultados del Experimento 2 de la Arquitectura 1

aumentado el *accuracy* tanto en el entrenamiento como en validación, reduciendo así el *bias* del modelo en

un 5% aunque ha aumentado ligeramente el *variance*. Como es lógico el tiempo de entrenamiento ha crecido, aunque no de forma lineal.
En la figura 1 vemos que a partir del epoch 400 no hay una mejora en *accuracy* para el conjunto de validación

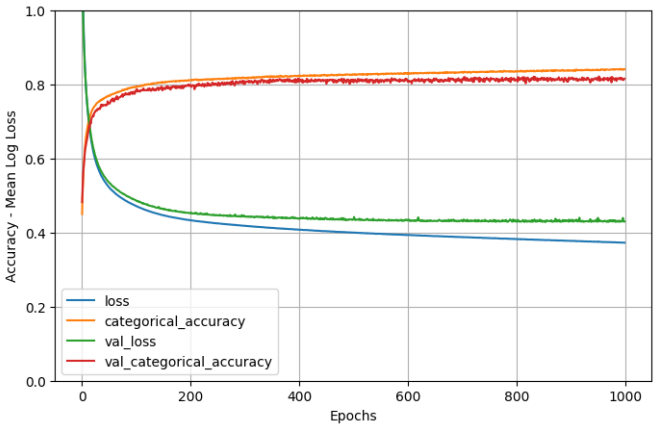

Figure 1: Entrenamiento durante el Experimento 2 de la Arquitectura 1

aunque sí para el conjunto de entrenamiento lo que nos indica que un número de *epochs* tan elevado como el que hemos usado en este experimento con esta arquitectura y configuración conduce a un *overfitting* del modelo, aunque por el momento no es excesivo como nos indica la *variance*.

1.3 Experimento 3: Cambiamos a *tanh* y reducimos *epochs*

Para este experimento decidimos reducir las *epochs* ya que como hemos visto en el experimento anterior, no hay una mejora significativa en validación con más epochs.
Además de esto, cambiaremos la función de activación a *tanh*. Hasta el momento hemos usado *ReLU* pero no hay razón para usarla para esta arquitectura porque resuelve el problema del *vanishing gradient* que se da en arquitecturas profundas.
La configuración que usamos para el experimento 3 es:

Epochs	**Learning rate**	**Batch size**	**Activation**	**Loss**	**Optimizer**	**Regularization**
400	0.1	512	**tanh**	Categorical Crossentropy	SGD	None

Table 5: Hiperparámetros para el Experimento 3 de la Arquitectura 1

Tras 5 entrenamientos obtenemos los siguientes resultados:

	Train accuracy (%)	**Validation accuracy (%)**	**Bias (%)**	**Variance (%)**	**Training time (s)**
Mean	79.58	78.3	15.41	1.28	72
Std	0.2	0.51	0.22	0.33	3.27

Table 6: Resultados del Experimento 3 de la Arquitectura 1

Con esta configuración el modelo ha vuelto a aumentar el *bias* y obtenemos unos resultados prácticamente idénticos al experimento uno con la función de activación *ReLU* aunque con más epochs. Parece que con la función *tanh*, el modelo se queda atrapado en mínimos globales como podemos apreciar en la figura 2 y necesita más epochs para escapar de ellos.

También observamos que no ha habido overfitting hasta la epoch 400, al contrario de lo que habíamos supuesto al inicio de este experimento.

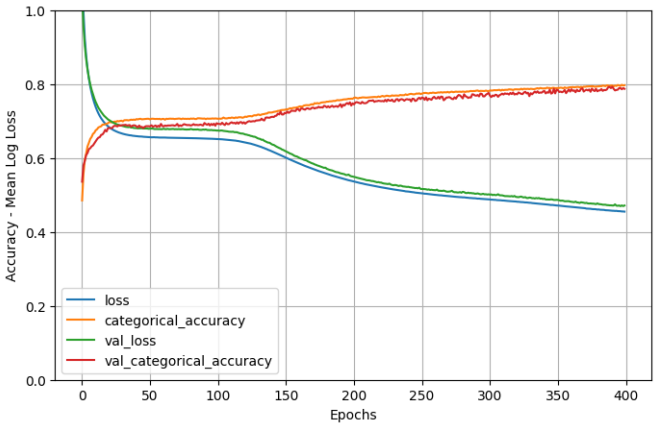

Figure 2: Entrenamiento durante el Experimento 3 de la Arquitectura 1

1.4 Experimento 4: Aumentamos *epochs*

El objetivo de este experimento es comprobar cuántas *epochs* podemos realizar antes de que el modelo comience a dirigirse hacia un *overfitting* por lo que la configuración es la misma que en la ejecución anterior, excepto que volvemos a incrementar las epochs a 1000:

Epochs	**Learning rate**	**Batch size**	**Activation**	**Loss**	**Optimizer**	**Regularization**
1000	0.1	512	tanh	Categorical Crossentropy	SGD	None

Table 7: Hiperparámetros para el Experimento 4 de la Arquitectura 1

Tras 5 entrenamientos obtenemos los siguientes resultados:

	Train accuracy (%)	**Validation accuracy (%)**	**Bias (%)**	**Variance (%)**	**Training time (s)**
Mean	81.17	80.23	13.82	0.94	189
Std	0.14	0.62	0.14	0.53	2.3

Table 8: Resultados del Experimento 4 de la Arquitectura 1

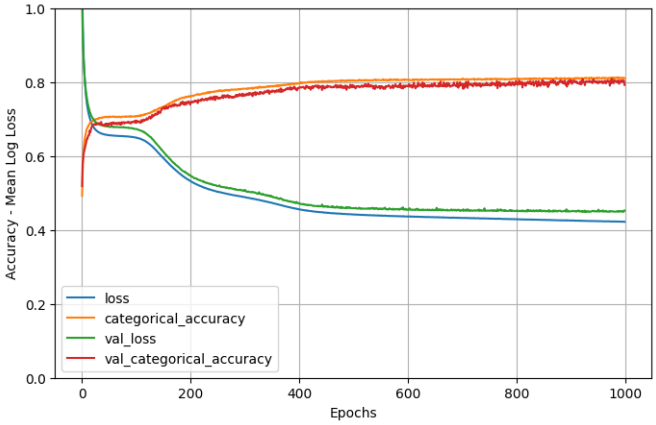

Figure 3: Entrenamiento durante el Experimento 4 de la Arquitectura 1

Aunque hemos doblado las *epochs* con respecto al experimento 3, la mejora ha sido de apenas un 2% en

accuracy. Por otra parte, como se muestra en la figura 3, el modelo no ha llegado al punto de *overfitting* aun habiendo usado un número tan alto de epochs.
Esto quiere decir que todavía hay margen de mejora si seguimos entrenando con más epochs aunque llevaría mucho tiempo porque el aprendizaje es lento.

1.5 Experimento 5: Reducimos *batch size*

Como se necesitaría aumentar exponencialmente el número de *epochs* para conseguir una mejora en el *accuracy*, decidimos reducir el *batch size* y comprobar si de esta forma el modelo consigue mejores resultados.

Epochs	**Learning rate**	**Batch size**	**Activation**	**Loss**	**Optimizer**	**Regularization**
1000	0.1	**128**	tanh	Categorical Crossentropy	SGD	None

Table 9: Hiperparámetros para el Experimento 5 de la Arquitectura 1

Tras 5 entrenamientos obtenemos los siguientes resultados:
Como vemos en la tabla 10 no conseguimos mejorar de forma significativa el *accuracy*. Aumentar el *batch size*

	Train accuracy (%)	**Validation accuracy (%)**	**Bias (%)**	**Variance (%)**	**Training time (s)**
Mean	82.92	80.84	12.07	2.08	517.2
Std	0.19	0.29	0.19	0.28	25.61

Table 10: Resultados del Experimento 4 de la Arquitectura 1

ha incrementado exponencialmente el tiempo de entrenamiento por lo que no merece la pena reducir el *batch size* en este caso. Lo que hemos conseguido reduciendo el *batch size* ha sido que el modelo escape rápidamente del primer mínimo local con el que se topa como vemos en la figura 4 así que concluimos que con esta técnica podemos obtener un resultado aceptable en poco tiempo.

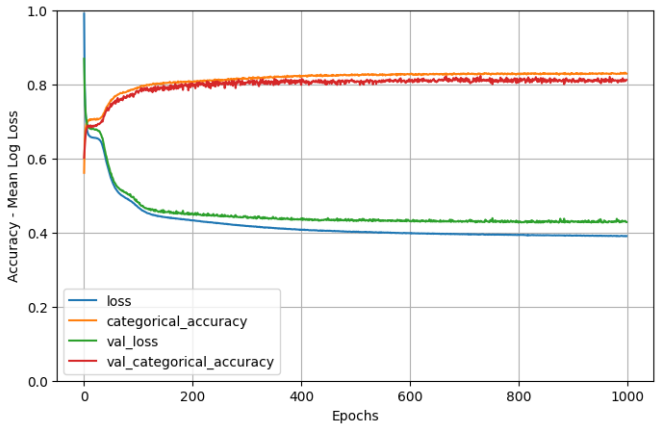

Figure 4: Entrenamiento durante el Experimento 5 de la Arquitectura 1

1.6 Conclusiones de la Arquitectura 1

- *ReLU* llega a un estado de *overfitting* con menos *epochs* que *tanh*
- *tanh* tiene un proceso de aprendizaje más lento que *ReLU*
- Reducir el *batch size* implica llegar a un óptimo de forma más rápida con *tanh*.

2 Arquitectura 2: