

Instituto Tecnológico y de Estudios Superiores de Monterrey

Departamento de computación

Inteligencia Artificial avanzada para la ciencia de datos

Grupo 101

Módulo 2 Uso de framework o biblioteca de aprendizaje máquina para la implementación de una solución.

Luis Ignacio Ferro Salinas

A01378248

12 de septiembre de 2022

Al usar un framework para implementar mi red neuronal, tengo más hiperparámetros disponibles para variar. Usaré la misma técnica de dejar todos los hiperparámetros constantes excepto el que estoy estudiando para ver el efecto de su cambio sobre las métricas.

En este caso realizo un modelo para predecir si una persona va a sobrevivir el choque del Titanic o no dependiendo de algunas de sus características.

Hiperparámetros:

Arquitectura del modelo (numero de capas intermedias)

Batch size

Epochs

Funciones de activación intermedias

Arquitectura del modelo

batch\_size=256,

epochs=100,

funciones de activación intermedias=sigmoid

Una sola neurona (regresión logística) 9 x 1

Chart, line chart

Description automatically generated with medium confidence

En este caso se ve que están un poco separadas las métricas de entrenamiento y de validación, pero parece que van en la misma dirección. Todas las métricas tienen mejora al final de las epochs contra al inicio.

Una capa intermedia 9 x 16 x 1

Graphical user interface, diagram

Description automatically generated

En este caso, las métricas de pérdida y de accuracy van bastante juntas, pero la AUC se separa bastante. Las métricas tienen mejores valores que en el caso anterior. Todas las métricas mejoran al final de las epochs contra el inicio.

2 capas intermedias 9 x 32 x 8 x 1

Diagram

Description automatically generated

Parece que en este caso van bastante juntas las métricas de validación y training, excepto en la AUC. Las métricas tienen valores similiares contra la arquitectura anterior.

Batch\_size

Arquitectura 9 x 16 x 1

Epochs 100

Funciones de activación intermedias sigmoid

Batch\_size 800 (todas las muestras de entrenamiento)

Chart, line chart

Description automatically generated

Parece que la métrica de AUC se separa un poco entre training y validation, no hubo gran cambio en los valores de las métricas contra 256 muestras, ahora pruebo 64 muestras.

Batch\_size 64

Chart, line chart

Description automatically generated

En este caso ya vemos bastante diferencia entre la forma en que cambian las métricas a través de las epochs entre validation y training, posiblemente 64 muestras en este caso no son suficientes para guiar el descenso del gradiente de la mejor manera.

Epochs

Arquitectura del modelo 9 x 16 x 1

Batch size 256

Funciones de activación intermedias sigmoid

1000 epochs

Graphical user interface

Description automatically generated

Estas métricas se ven bastante bien a través de las 1000 epochs, porque tenemos un crecimiento recíproco entre validation y training, ósea en general crecen las métricas en ambos casos de training y de validation. Parece también que se comienzan a estabilizar las métricas, ya no cambian tanto al final.

Funciones de activación intermedias

Arquitectura del modelo 9 x 16 x 1

Batch size 256

Epochs 1000

Funciones de activación intermedias ReLU

Graphical user interface, application

Description automatically generated

WOW, esto está interesante porque ahora la pérdida en el training y validation convirgió extremadamente más rápido que con la función intermedia de sigmoid.

Funciones de activación intermedia tanh

Graphical user interface, diagram

Description automatically generated

Es un resultado similar al de ReLU, pero creo que el comportamiento de ReLU es tan bueno en la loss que puedo bajar significativamente el número de epochs

Configuración final

Arquitectura del modelo 9 x 16 x 1

Batch size 256

Epochs 300

Funciones de activación intermedias ReLU

Graphical user interface

Description automatically generated

Ahora vuelvo a hacer el Split de training y testing con conjuntos distintos a ver si este comportamiento es reproducible

Graphical user interface, diagram

Description automatically generated

Y los resultados son bastante similares aún con conjuntos de training y testing diferentes.

Ahora vuelvo a hacer el proceso anterior pero ahora plasmando los valores finales de las métricas en tablas para visualizarlo en un espacio más reducido

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Arquitectura | Perdida final | Perdida final validacion | Accuracy final | Accuracy final validación | AUC final | AUC final valifación |
| 9x1 | 48.61689376831055 | 45.798728942871094 | 0.6138888597488403 | 0.625 | 0.5272565484046936 | 0.502633273601532 |
| 9x16x1 | 0.6185844540596008 | 0.6147946715354919 | 0.6277777552604675 | 0.625 | 0.7423335909843445 | 0.7297563552856445 |
| 9x32x8x1 | 0.6177414059638977 | 0.6066555976867676 | 0.6277777552604675 | 0.637499988079071 | 0.7405641674995422 | 0.7679394483566284 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Batch size | Perdida final | | Perdida final validacion | | Accuracy final | | Accuracy final validación | | AUC final | | AUC final valifación | |
| 800 | .6252694129943848 | | 0.6237276792526245 | | 0.6347222328186035 | | 0.6499999761581421 | | 0.7562419176101685 | | 0.7435812950134277 | |
| 64 | | 0.5504095554351807 | | 0.536711573600769 | | 0.7180555462837219 | | 0.7749999761581421 | | 0.7886120080947876 | | 0.7870309352874756 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Epochs | Perdida final | Perdida final validacion | Accuracy final | Accuracy final validación | AUC final | AUC final valifación |
| 1000 | 0.3927774131298065 | 0.386232852935791 | 0.8277778029441833 | 0.875 | 0.386232852935791 | 0.8512178659439087 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Activación de capa intermedia | Perdida final | Perdida final validacion | Accuracy final | Accuracy final validación | AUC final | AUC final valifación |
| ReLU | 0.4429752826690674 | 0.4084412157535553 | 0.8013888597488403 | is 0.8500000238418579 | 0.8548538684844971 | 0.8459512591362 |
| Tanh | 0.42620909214019775 | 0.41985827684402466 | 0.8180555701255798 | 0.862500011920929 | 0.8606981039047241 | 0.8439762592315674 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Configuración final | Perdida final | Perdida final validacion | Accuracy final | Accuracy final validación | AUC final | AUC final valifación |
|  | 0.445425808429718 | 0.4179840683937073 | 0.7875000238418579 | 0.8374999761581421 | 0.8569480776786804 | 0.853522002696991 |

Predicciones

Hago 89 predicciones sobre las 89 muestras de test que el modelo nunca ha visto y parece que sigue bastante bien a lo que debió haber predicho.

Chart

Description automatically generated with low confidence

**Conclusión final**

En esta ocasión logré la implementación inicial en muy poco tiempo comparado con la realización del modelo sin framework, pero me tardé más recopilando los resultados y variando los hiperparámetros, ya que el framework me da muchas más opciones de personalización y tuneo del modelo. Ahora con la capacidad de incluir las métricas de Accuracy y de AUC pude visualizar un poco mejor el desempeño de mi modelo de clasificación binaria, y fue interesante que cuando logré mejoras en el desempeño en general todas las métricas mejoraban así tenía más seguridad de que en verdad mejoró mi modelo.

Algo muy importante fue que pude visualizar el desempeño de mi modelo con un conjunto de validación, porque así pude ver que en algunas combinaciones de hiperparámetros, mi modelo comenzaba a mejorar en métricas solamente para el conjunto de entrenamiento y no para el conjunto de validación, lo que me ayudó a detectar un posible overfitting y evitarlo.