Instituto Tecnológico y de Estudios Superiores de Monterrey

Departamento de computación

Inteligencia artificial avanzada para la ciencia de datos I

Grupo 101

Módulo 2 Implementación de una técnica de aprendizaje máquina sin el uso de un framework.

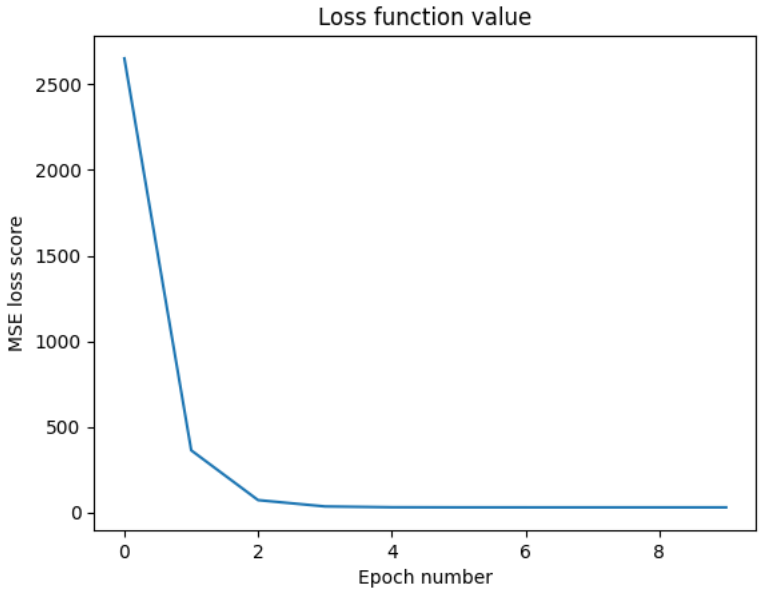
Luis Ignacio Ferro Salinas

A01378248

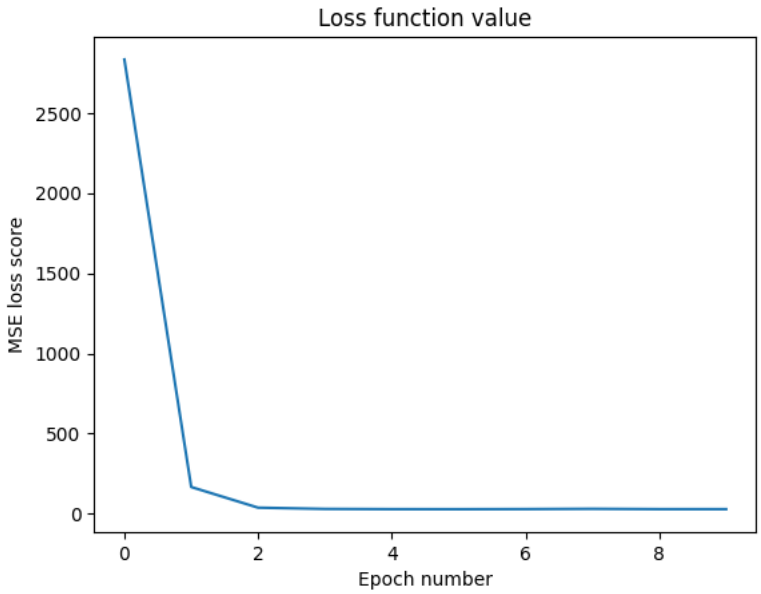
5 de septiembre de 2022

Analizo lo que le sucede a la función de error al variar los híperparámetros.

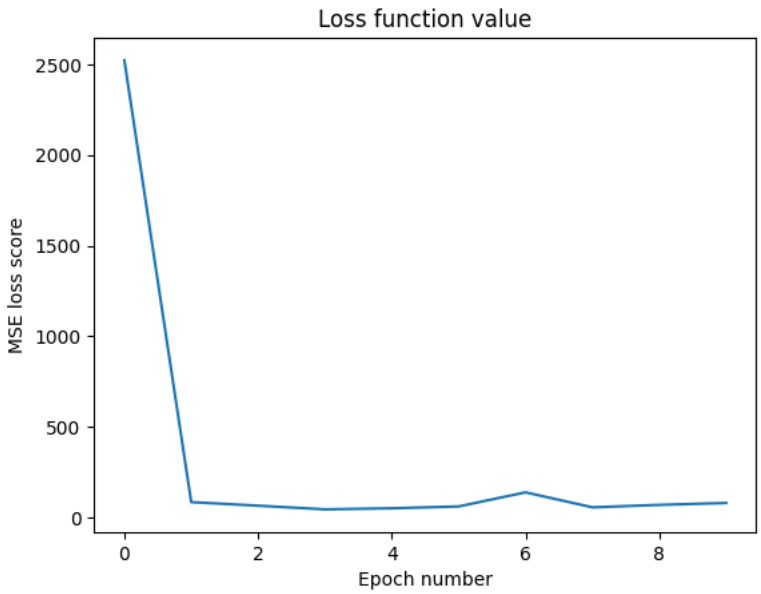
Primero al dejar el relative_batch_size como 1, ósea 100 % de los datos para calcular las gradientes y actualizar después, con learning rate 0.01 y número de épochs 10 constantes.



Ahora con el 50%



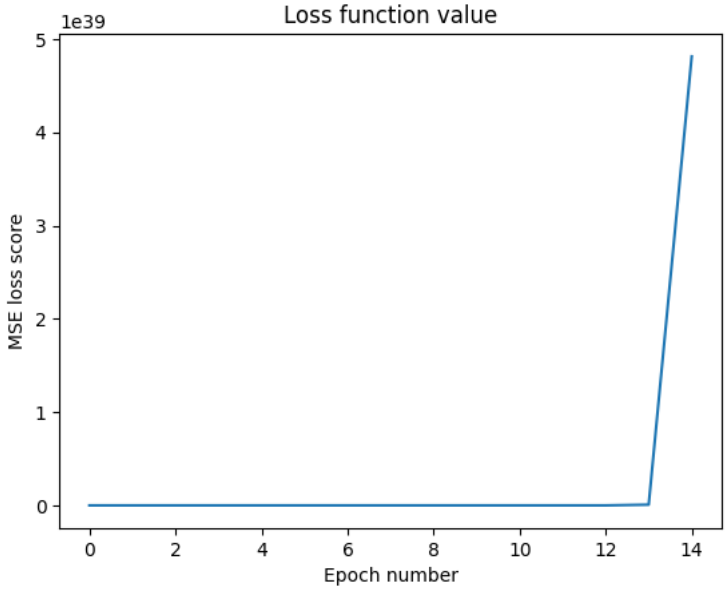
Parece que sigue convergiendo igual, ahora pruebo con el 7% de los datos.



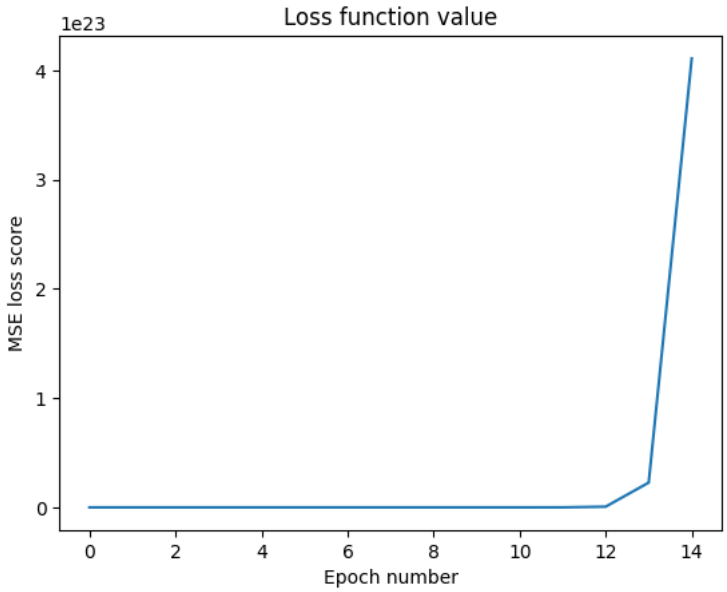
Esto está interesante porque aunque hay ligeras variaciones, el valor de la loss function que es verdaderamente mínimo solamente se obtiene al usar el 100% de los datos antes de actualizar los datos, pero aún usando el 7% de los datos, se obtienen resultados que reducen la función de pérdida.

Ahora veo que pasa al cambiar el learning rate dejando el relative_batch_size 50% y el número de épocas, 10, constantes.

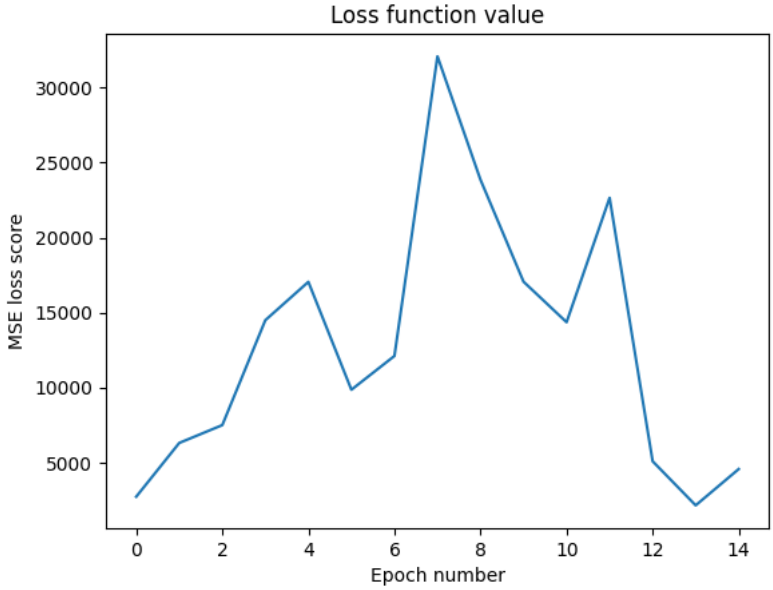
Pruebo inicialmente con learning_rate de 0.3, que vimos en clase que era como bastante conocido.



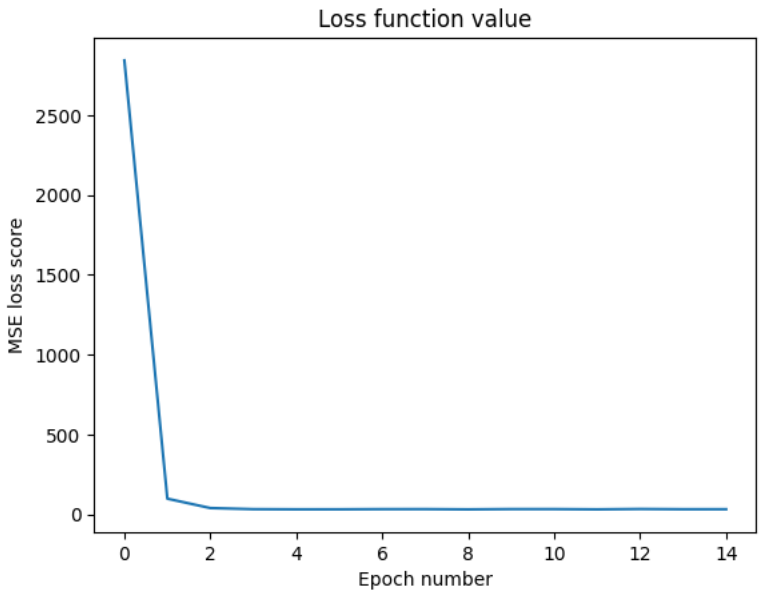
Parece que la función de loss se dispara, ahora pruebo con 0.1



Ahora un poco menos, ahora pruebo con 0.03



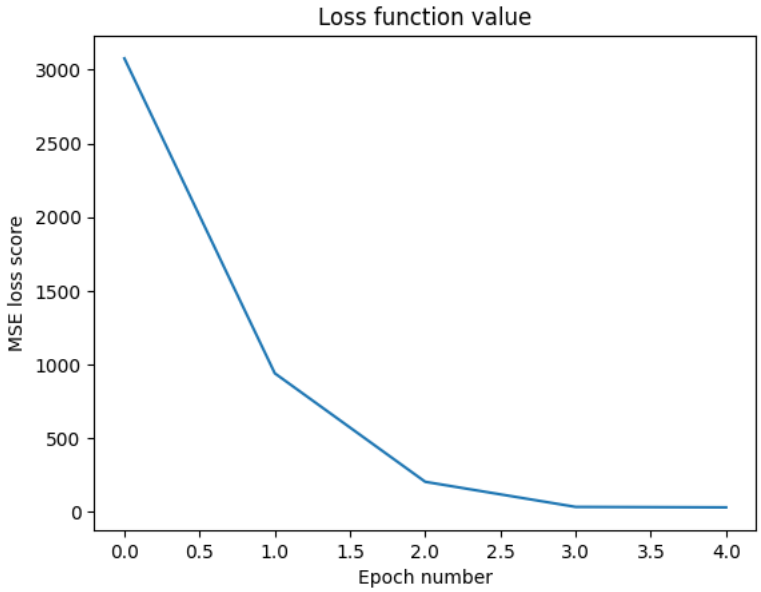
Un caso interesante, parece que se dispara y luego regresa un poco. Ahora pruebo con 0.01



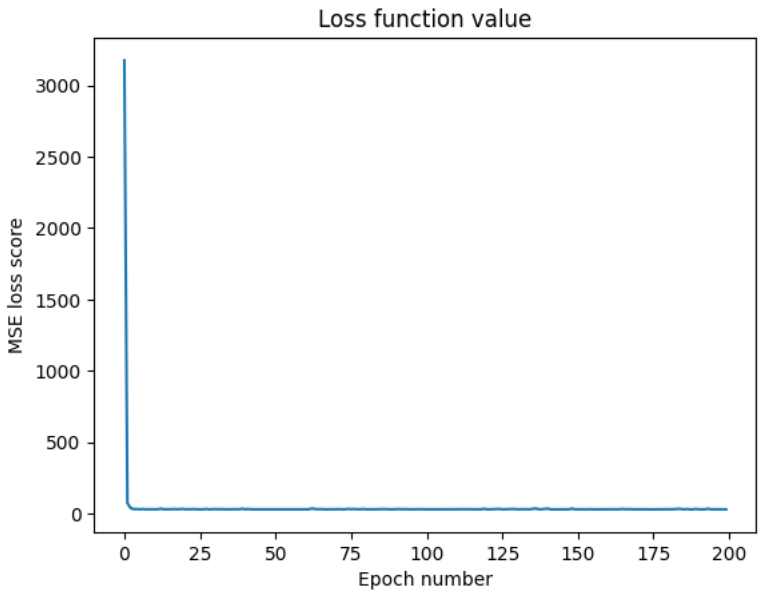
Aquí está, dejando 0.01 de learning rate la función converge casi inmediatamente en la primera epoch prácticamente.

Finalmente cambio el número de épocas dejando el relative_batch_size 50% y el learning_rate 0.01 constantes

Primero pruebo con 5 epochs



Parece que se estabiliza la loss, pero para asegurar pruebo con 200 epochs



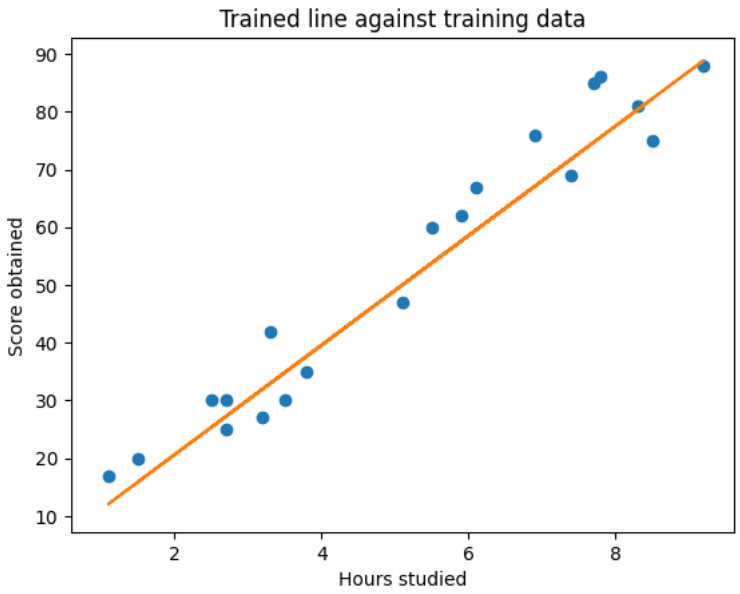
Ahora vuelvo a hacer el proceso anterior pero ahora registrando los valores finales de la función de pérdida en una tabla para visualizar todo en un mismo espacio.

Learning rate	Valor final de función de pérdida
0.3	1.3580128143728424e+27
0.1	1.5802830154709754e+17
0.01	30.284905372438992

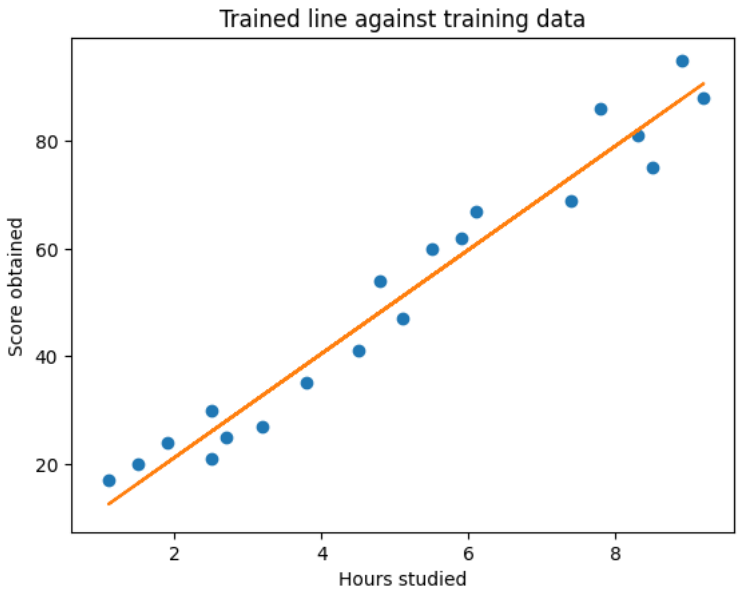
Relative batch size	Valor final de función de pérdida
100%	30.25378710272895
50%	30.723526383943558
7%	188.05887182239024

Epochs	Valor final de función de pérdida
5	32.63156651067349
200	30.60817409706852

Ahora sí, es claro que se estabiliza y esta es la línea que queda en los datos de entrenamiento y en las predicciones.

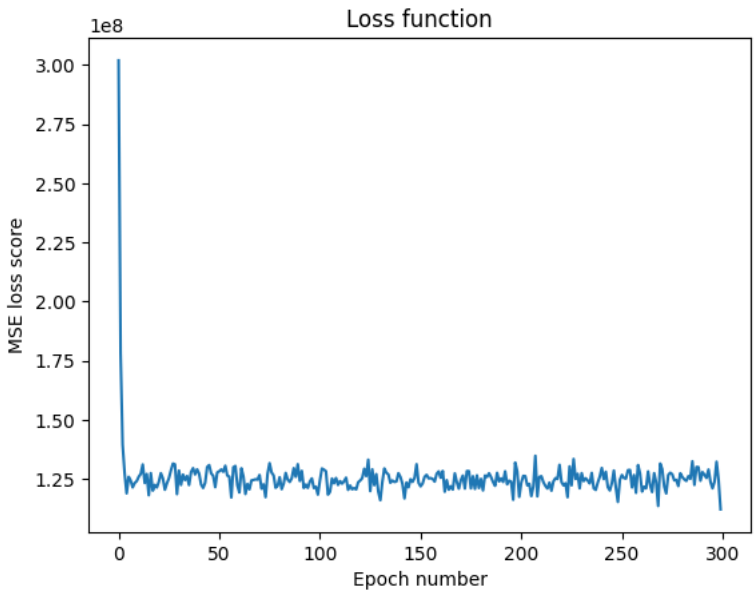


Como mi función que separa en training y test agarra datos aleatorios, simplemente vuelvo a hacer el proceso para que sean distintos los conjuntos de training y testing a ver si la línea final difiere.



Queda prácticamente igual.

También hago el proceso con un dataset de 3 variables independientes para ver cómo se comporta la función de pérdida, y tuneo los hiperparametros.



En este caso así logré que bajara la función de pérdida, y puedo hacer predicciones sobre los precios de un seguro médico para una persona dependiendo de su edad, índice de masa corporal y su número de hijos.

```
Your health insurance charges will be 7000.430674257332
```

```
In [497]:
```

Con mis datos salió de 7000 y encontré una instancia que tenía datos parecidos a los míos y de charges tenía 2585.85065, entonces tal vez no es la mejor predicción pero en mi opinión no está tan mal porque solamente son 4 parámetros y posiblemente no está lejos de la solución óptima.

Conclusión final

La verdad fue un poco retador realizar este modelo sin un framework porque debo de r evisar los detalles de la implementación para que sí se logre mejorar la función de pérdi da. Me gustó cuando logré que se redujera la función de pérdida porque al inicio se dis paraba y pensé que era porque había cometido algún error, pero después de probar va rios cambios en los hiperparámetros y de recordar que en clase vimos que se podía dis parar el error por un learning rate alto, entonces ya pude resolverlo. Hice variaciones en batch_size, en epochs y en learning rate para lograr reducir lo más posible la función d e pérdida en la menor cantidad de epochs y con lo que plasmé en las gráficas y tablas creo que lo logré lo mejor que pude.