Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI-5438 Inteligencia Artificial II
Trimestre Septiembre - Diciembre 2023

Informe Proyecto 1

Estudiantes:

Br. Gamboa, Roberto

Br. Blanco, Luis

Carnés:

16-10394

17-10066

Profesor

Carlos Infante

3 de noviembre de 2023

Índice

Introducción

El descenso de gradiente es un algoritmo utilizado en la Inteligencia Artificial para el entrenamiento óptimo de Machine Learning y Redes Neuronales. Con el paso del tiempo, los datos con los que se entrenan estos modelos ayudan a que cada vez se obtengan mejores resultados y que la función sea mucho más precisa.

El objetivo principal de este proyecto es la implementación de dicho algoritmo, para luego realizar el entrenamiento con un set de datos y estudiar su precisión. El set de datos a utilizar debe ser preprocesado primero, para luego poder entrenar el modelo con él.

Link al repositorio del proyecto

El proyecto se encuentra alojado en el siguiente enlace:

`https://github.com/rcgamboan/CI5438-Proyecto1/tree/main`

Implementación

El algoritmo fue implementado en el lenguaje de programación Python. Se siguió el pseudocódigo dado en clases y se complementó con los códigos que se encuentran en:

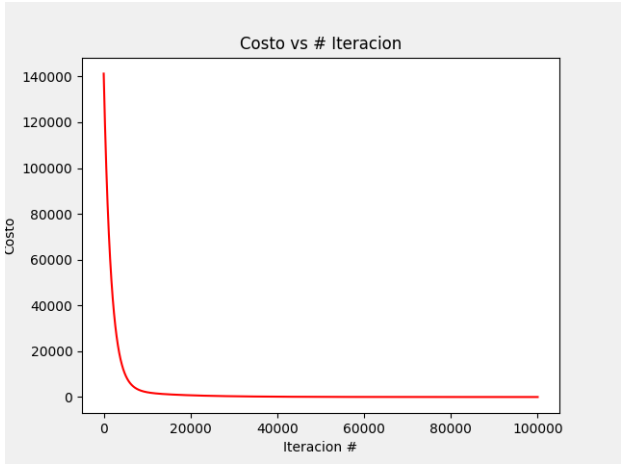
- "Multivariate Linear Regression w. Gradient Descent"
- "How to implement a gradient descent in Python to find a local minimum ?"

Se utilizó la librería numpy para realizar los cálculos requeridos en el algoritmo de una forma cómoda y sencilla. Además para realizar las gráficas se usó matplotlib.

El algoritmo de descenso de gradiente cuenta con un criterio de convergencia que compara el resultado de cada iteración con el de la iteración anterior, deteniendo su ejecución si la diferencia entre ambos es mínima. En cada iteración del algoritmo se calcula el costo de la iteración y se guarda en un arreglo, que posteriormente es utilizado para representar gráficamente la evolución del costo.

Dentro del archivo `src/descenso.py` se realiza una prueba al algoritmo generando 1000 puntos aleatorios mediante la función generar_data y utilizando una función de la forma $f(x_1, x_2) = w_1x_1 + w_2x_2 + w_0$ para calcular el valor de y. Al obtener la hipótesis para el modelo con algoritmo de descenso de gradiente y comparar los valores obtenidos al evaluar los 1.000 puntos en la hipótesis con los valores reales en y, se obtienen valores muy cercanos a los reales. Además se estudió la precisión del modelo calculando el Error Cuadrático Medio (RMSE), dando como resultado un RMSE de aproximadamente 3,80, siendo este un valor pequeño y dentro de un rango aceptable. Al ejecutar el algoritmo de descenso de gradiente en este caso se fijó una tasa de aprendizaje de 0,00000005, ya que valores mayores para este modelo ocasionaban que el algoritmo no convergiera a ninguna solución, y valores mas pequeños resultaban en hipótesis muy desviadas de los valores reales. Se fijó también un máximo de 10.000 iteraciones pero el modelo converge a una solución luego de 6.000 iteraciones.

En la siguiente gráfica se evidencia el costo de cada iteración vs la cantidad de iteraciones totales realizadas:



Se evidencia en la gráfica que se minimiza el valor del costo por cada iteración.

Preprocesamiento de datos

Para el preprocesamiento de los datos se utilizó la librería pandas. En primer lugar se lee el archivo `CarDekho.csv` y se crea un `DataFrame` con los datos que contiene el archivo, luego se eliminan las columnas que no se utilizarán para el entrenamiento del modelo, dejando solo las que fueron recomendadas en el enunciado del proyecto.

En segundo lugar, se deben tratar las filas que contienen datos nulos, para ello se implementaron dos métodos: en el primero se eliminan las filas con valores nulos; en el segundo se rellenan las celdas vacías con el dato que más se repita en la columna correspondiente.

En tercer lugar, se debe realizar la categorización de la data. Aquellos atributos que tengan valores no numéricos se deben convertir en numéricos. Se deben crear tantos atributos como categorías haya en cada atributo y luego si el valor pertenece al nuevo atributo se coloca un 1, en caso contrario 0. Todo este proceso se realizó con ayuda del método one-hot encoding de la librería `sklearn` y la librería `category_encoders`.

Por último se realiza la normalización de los datos con valores numéricos muy altos con la fórmula que fue dada en el enunciado.

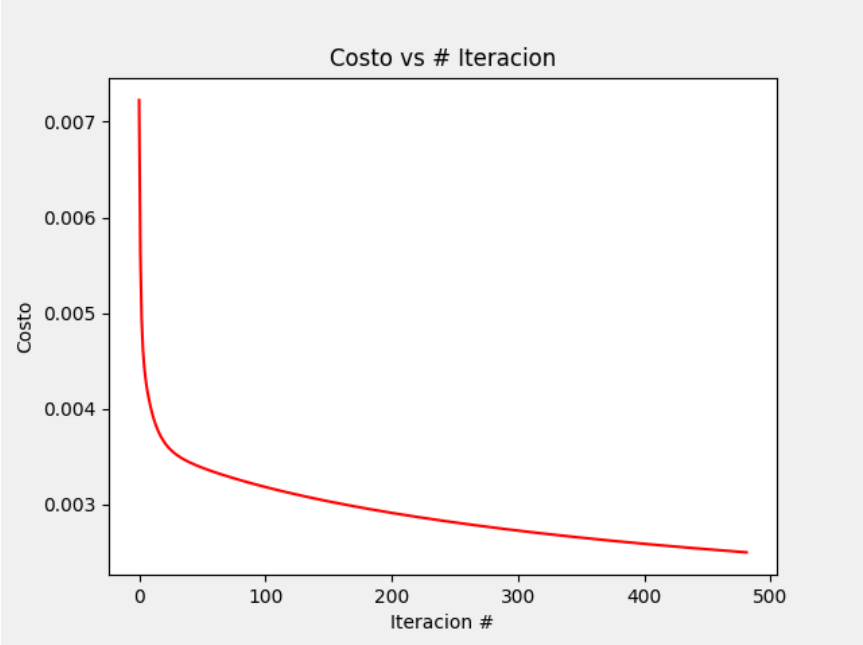
Entrenamiento del modelo

El entrenamiento del modelo se puede encontrar en el archivo `src/entrenamiento.py`. Primero se obtienen los datos a ser utilizados para entrenar el modelo y se dividen en dos lotes: un lote de entrenamiento y un lote de prueba, los cuales conforman en 80 % y 20 % respectivamente del conjunto total de datos. Luego se establecen la tasa de aprendizaje a ser utilizada y la cantidad de iteraciones máxima para hacer la llamada al algoritmo de descenso de gradiente.

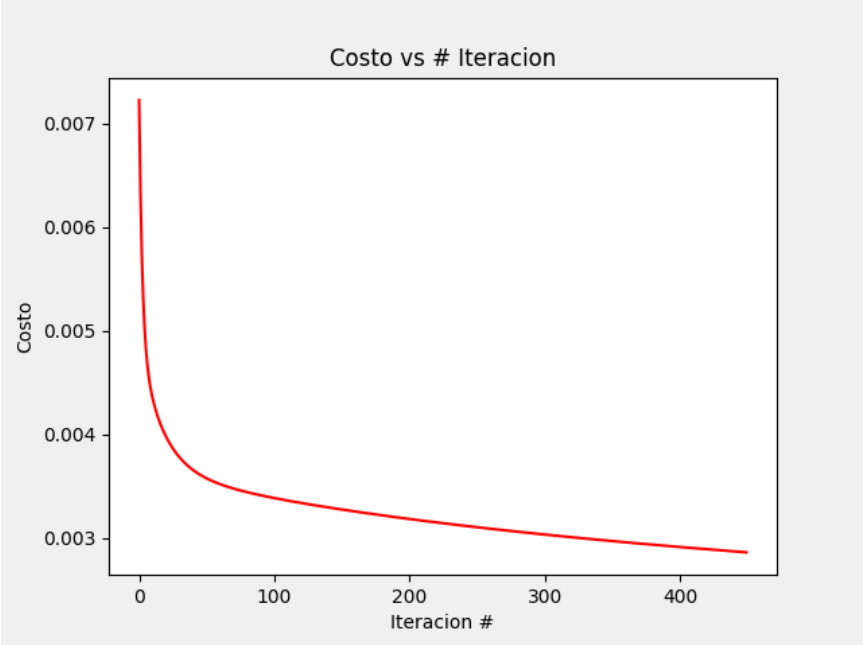
Al obtener la hipótesis por medio del algoritmo de descenso, se calculan las predicciones para el conjunto de prueba, así como el error. Por último se desnormalizan los valores del precio en el conjunto de prueba, para así poder comparar los precios reales con los obtenidos por el modelo.

Para la ejecución del algoritmo de descenso de gradiente se fijó una tasa de aprendizaje de 0.05 y un máximo de 100.000 iteraciones pero el algoritmo converge luego de 400 iteraciones. Se experimentó con diversos valores para la tasa de aprendizaje, siendo las gráficas obtenidas las siguientes:

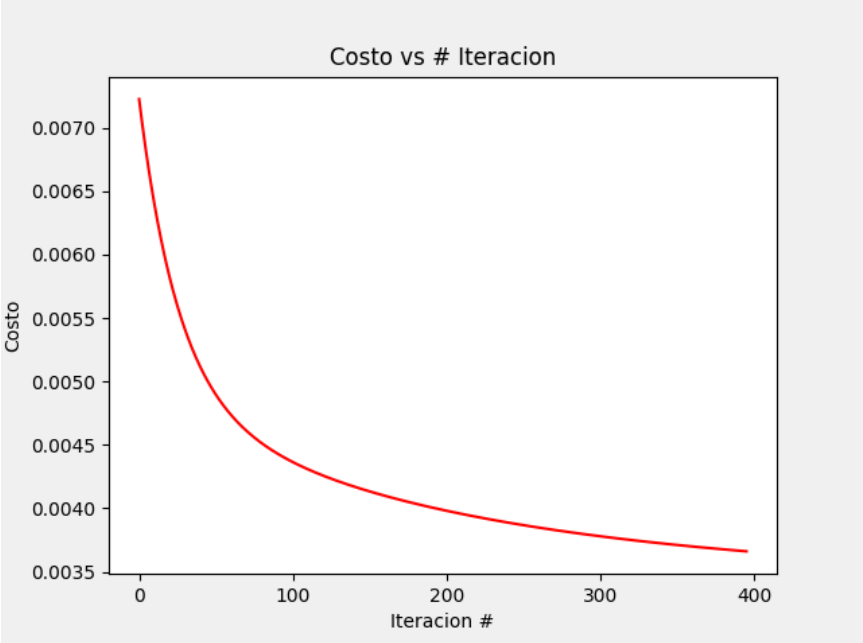
- 0,1:



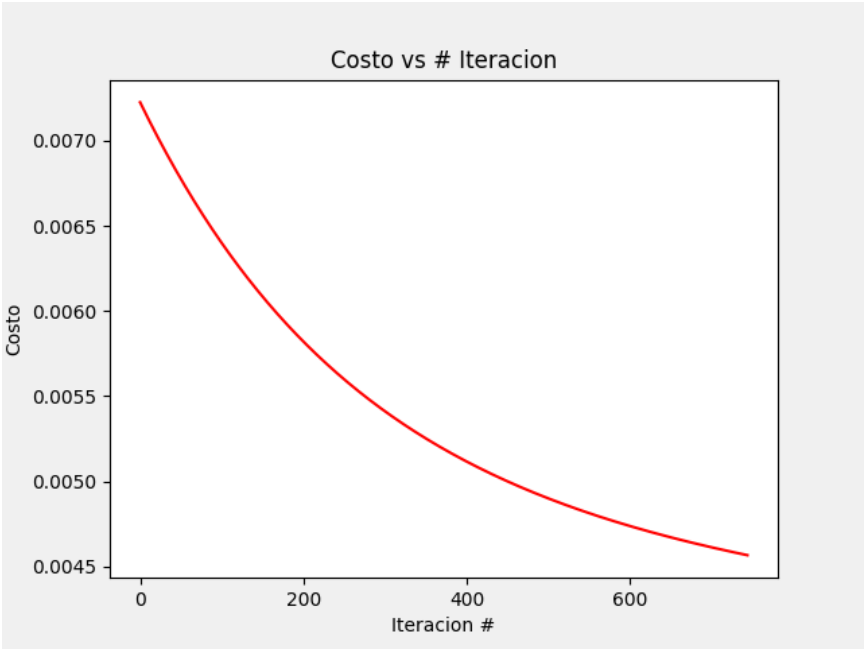
- 0,05:



- 0,005:



- 0,0005:



Se puede apreciar mediante las gráficas que una tasa de aprendizaje óptima para el modelo se encuentra entre 0,1 y 0,05, ya que que con estas tasas se minimiza en mayor medida el costo en cada iteración.

Para tasas de aprendizaje menores a 0,0005 el algoritmo no converge a ninguna solución con el modelo dado.

Los precios obtenidos al evaluar los datos del conjunto de pruebas con la hipótesis obtenida, y su comparación con los precios reales se pueden consultar en el directorio `doc/Datos informe`

Ejecución del proyecto

El proyecto fue realizado en el lenguaje de programación Python, en su versión 3.11.2. Para ejecutar el proyecto inicialmente se requiere instalar las librerías necesarias, especificadas en el archivo `requirements.txt` ubicado en la carpeta raíz del repositorio del proyecto, las mismas pueden instalarse mediante el comando

```
$ pip install -r requirements.txt
```

Posteriormente basta con acceder al directorio `src` y ejecutar el archivo entrenamiento.py con el comando

```
$ python3 entrenamiento.py
```

Adicionalmente, si se desea probar la implementación del algoritmo de descenso de gradiente, se puede ejecutar el archivo descenso.py con el comando

```
$ python3 descenso.py
```

Conclusión

En este proyecto se ha implementado y probado el algoritmo de descenso de gradiente para el entrenamiento de un modelo de datos.

Según lo observado al probar el algoritmo de descenso de gradiente al generar los 1.000 puntos aleatorios y al entrenar el modelo con los datos suministrados, es evidente que su ejecución es fuertemente influenciada por el valor de la tasa de aprendizaje.

La tasa de aprendizaje depende a su vez de cada modelo y no existe una forma certera para calcularla en cada caso, provocando que se deba probar el algoritmo para distintos valores de tasa de aprendizaje en cada ocasión. Valores muy altos para un modelo pueden ocasionar que el algoritmo genere números de muy grande magnitud y hacer overflow en el computador, mientras que el mismo valor para otro conjunto de datos puede funcionar perfectamente y dar como resultado una hipótesis que se ajuste satisfactoriamente a los datos. En otros modelos una tasa de aprendizaje muy baja puede causar que el algoritmo no converja pero en otros modelos sí.

En general, el algoritmo de descenso de gradiente es una herramienta poderosa, pero es importante tener en cuenta la dependencia del modelo con el valor de la tasa de aprendizaje para obtener el resultado deseado.