Actividad 3 Neurona Lineal

Alumno: José Osvaldo Farfán de León

Codigo: 214796622

Materia: IA II

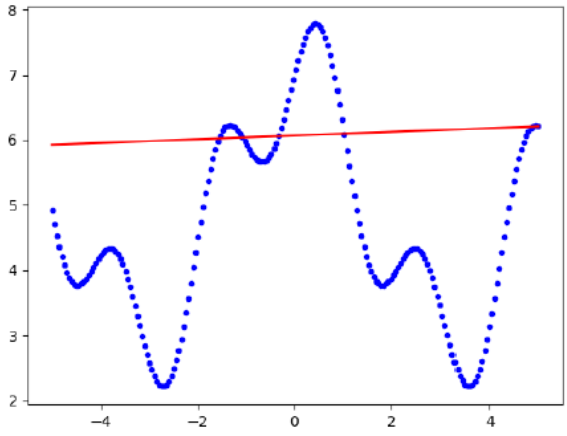
Profesor: Carlos Alberto Villaseñor Padilla

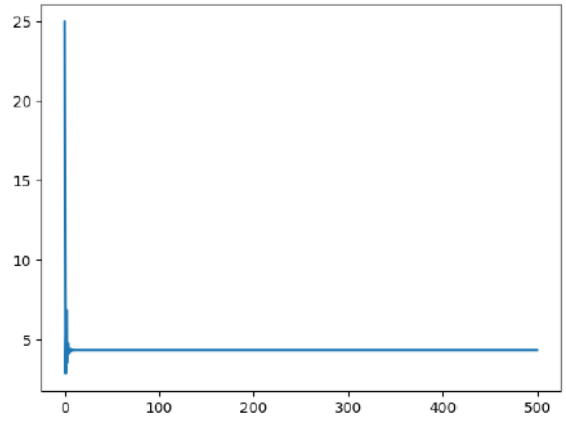
Sección: “D03”

Neurona lineal

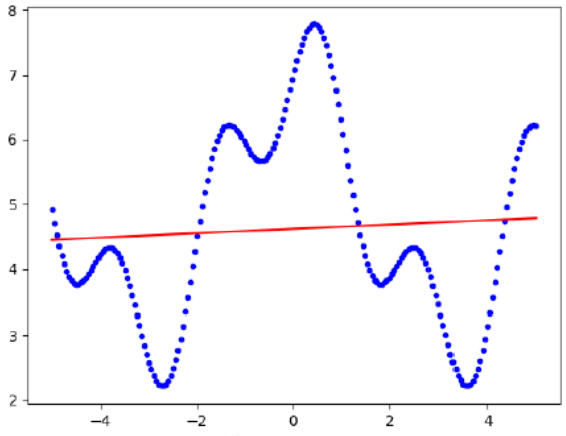
1. Tomando en cuenta lo visto en clase programa la neurona lineal (ADALINE) en tu lenguaje favorito.
2. Programa en una clase los siguientes tres métodos de SGD, BGD y el método directo (Pseudoinversa)
3. Con los datos anexos, entrena la red neuronal con los tres métodos y demuestra que da un buen resultado en los tres graficando los datos y la linea de regresión.

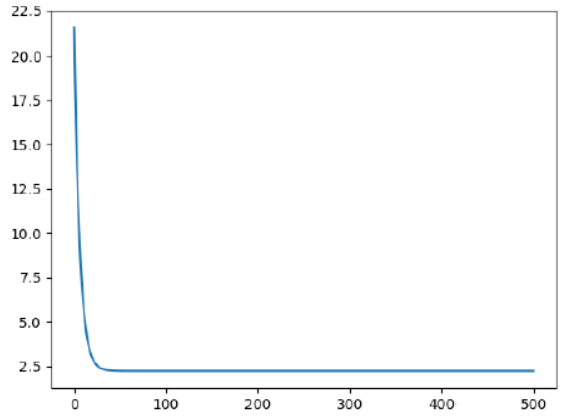
SGD



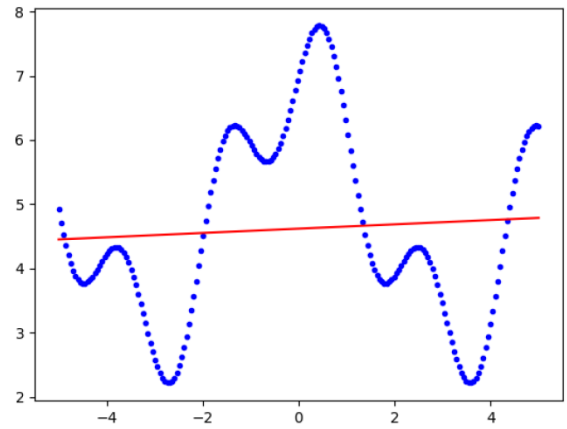


BGD

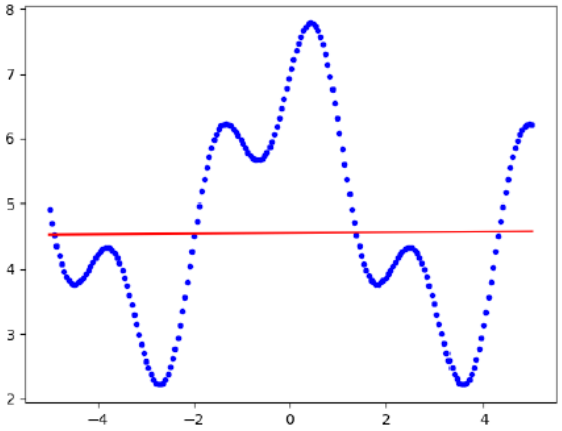


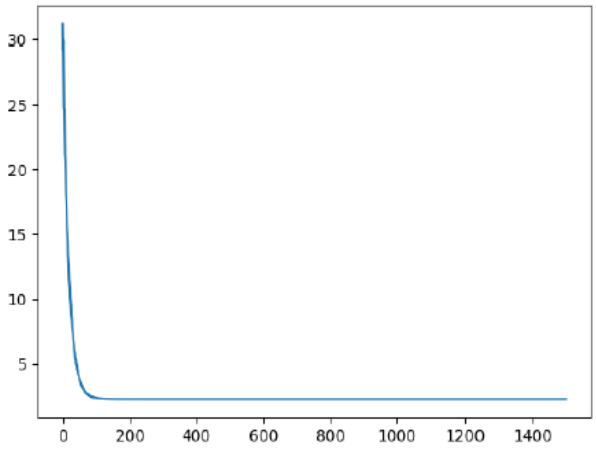


Método Directo



mBGD





Codes SGD, BGD y Directo

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

class LinearNeuron:
    def __class__init__(self, n_input):
        self.w = -1 + 2 * np.random.rand(n_input)
        self.b = -1 + 2 * np.random.rand()

    def predict(self, X):
        Y_est = np.dot(self.w, X) + self.b
        return Y_est

    def MSE(self, X, Y):
        p = X.shape[1]
        Y_est = self.predict(X)
        return (1/p) * np.sum((Y -Y_est) ** 2)
```

```
    def fit(self,X, Y, epochs=500, lr = 0.08, solver = 'BGD'):
        p = X.shape[1]
        error_history = []

        if solver == 'SGD':
            for _ in range(epochs):
                for i in range(p):
                    y_est = self.predict(X[:, i])
                    self.w += lr * (Y[:, i] - y_est) * X[:,i]
                    self.b += lr * (Y[:, i] - y_est)
                error_history.append(self.MSE(X, Y))

        elif solver == 'BGD':
            for _ in range(epochs):
                Y_est = self.predict(X)
                self.w += (lr/p) * ((Y-Y_est) @ X.T).ravel()
                self.b += (lr/p) * np.sum(Y - Y_est)
                error_history.append(self.MSE(X, Y))

        else:
            X_hat = np.concatenate((np.ones((1,p)), X), axis=0)
            W_hat = Y.reshape(1, -1) @ np.linalg.pinv(X_hat)
            self.w = W_hat[0, 1 :]
            self.b = W_hat[0,0]
            error_history.append(self.MSE(X,Y))
        return error_history

"""
Ejemplo
p=100
x= -1 + 2 * np.random.rand(p).reshape(1, -1)
y = -18 * x + 6 + 3 * np.random.rand(p)


"""
data = pd.read_csv("reg_problem.csv")
x = data["x"].values.reshape(1, -1)
y = data["y"].values.reshape(1, -1)

neurona = LinearNeuron(1)
error = neurona.fit(x, y, solver=':)')
```

```
#Dibujo
plt.plot(x,y, '.b')
xnew = np.array([[-5,5]])
ynew = neurona.predict(xnew)
plt.plot(xnew.ravel(), ynew, '-r')

plt.figure()
plt.plot(error)
plt.show()
```

mBGD

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

class LinearNeuron:
    def __class__init__(self, n_input):
        self.w = -1 + 2 * np.random.rand(n_input)
        self.b = -1 + 2 * np.random.rand()

    def predict(self, X):
        Y_est = np.dot(self.w, X) + self.b
        return Y_est

    def batcher(self, X, Y, batch_size):
        p = X.shape[1]
        li, ui = 0, batch_size
        while True:
            if li < p:
                yield X[:, li:ui], Y[:, li:ui]
                li, ui = li + batch_size, ui + batch_size
            else:
                return None

    def MSE(self, X, Y):
        p = X.shape[1]
        Y_est = self.predict(X)
        return (1/p) * np.sum((Y -Y_est) ** 2)
```

```
    def fit(self,X, Y, epochs=500, lr = 0.08, batch_size = 16):
        p = X.shape[1]
        error_history = []

        for _ in range(epochs):
            miniBatch = self.batcher(X, Y, batch_size)
            for mX, mY in miniBatch:
                mY_est = self.predict(mX)
                self.w += (lr/p) * ((mY-mY_est) @ mX.T).ravel()
                self.b += (lr/p) * np.sum(mY - mY_est)
                error_history.append(self.MSE(X, Y))
        return error_history



"""
Ejemplo
p=100
x= -1 + 2 * np.random.rand(p).reshape(1, -1)
y = -18 * x + 6 + 3 * np.random.rand(p)


"""
data = pd.read_csv("reg_problem.csv")
x = data["x"].values.reshape(1, -1)
y = data["y"].values.reshape(1, -1)

neurona = LinearNeuron(1)
error = neurona.fit(x, y, batch_size=77)

#Dibujo
plt.plot(x,y, '.b')
xnew = np.array([[-5,5]])
ynew = neurona.predict(xnew)
plt.plot(xnew.ravel(), ynew, '-r')

plt.figure()
plt.plot(error)
plt.show()
```

Conclusión:
Después de completar la actividad, podemos concluir que la mayoría de los métodos producen resultados prácticamente idénticos, pero aún así existen algunas diferencias notables.

Si examinamos el método SGD, podemos observar que en esta ocasión produjo un resultado con una mayor discrepancia en comparación con los otros métodos. También es evidente que realizó cambios más significativos en términos del error.

Mientras que los métodos directo y BGD arrojaron resultados prácticamente idénticos en esta ocasión. Es posible que pudiéramos detectar diferencias si realizamos pruebas con un conjunto de datos más grande.

Por último, aunque el método mBGD es bastante similar, se distingue por producir un resultado ligeramente diferente y, de alguna manera, parece ser más preciso. El error experimentó cambios más notorios, y la curva resultante parece más lineal que la de los otros métodos. Esto inspira mayor confianza, aunque no puedo precisar la razón detrás de esta observación.