Centro de Investigación en Matemáticas (CIMAT). Unidad Monterrey

Tarea 2
Reporte de Resultados

Marcelo Alberto Sanchez Zaragoza

22 de septiembre de 2021

1. Planteamiento

Para los ejercicios vamos a usar los datos generados por el modelo:

$$f(x;\theta) = f(x;A,w,\phi) = Asin(wx+\phi)$$

y la función que mide el ajuste del modelo

$$L(\theta;X,y) = \frac{1}{n}\sum_{n}^{i=1}(f(x_i;\theta) - y_i)^2$$

donde $\mathbf{X} = (x_1, x_2, ..., x_n)$ y $\mathbf{y} = (y_1, y_2, ..., y_n)$ y los puntos (x_i, y_i) están generados por el siguiete código.

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

# Modelo
def fnc_f(x, theta):
A,w,phi = theta
return A*np.sin(w*x + phi)

A   = 13
w   = 1.2
phi = 0.6
# Generacion de los datos del problema, almacenados en los arreglos X,y
X     = np.random.rand(2**9)*6
theta = (A, w, phi)
y     = fnc_f(X, theta) + 2*np.random.randn(len(X))

```

El conjunto de datos que se generan se puede observar en la figura 1.1

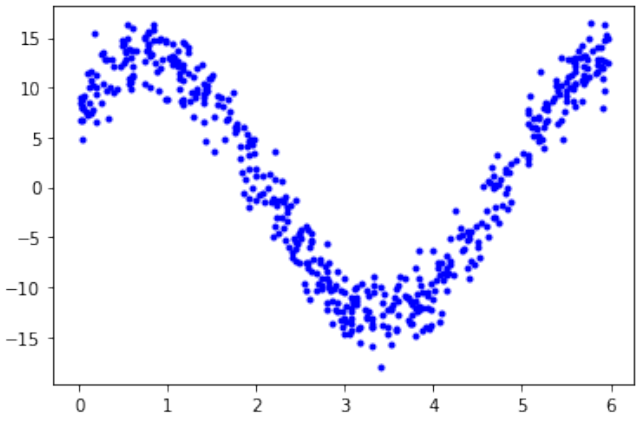

Figura 1.1: Gráfica del conjunto de datos generados.

Nos proporcionan un punto inicial $\theta_0 = (18,0,6,0)$, en la figura 1.2 se

observa este punto inicial.

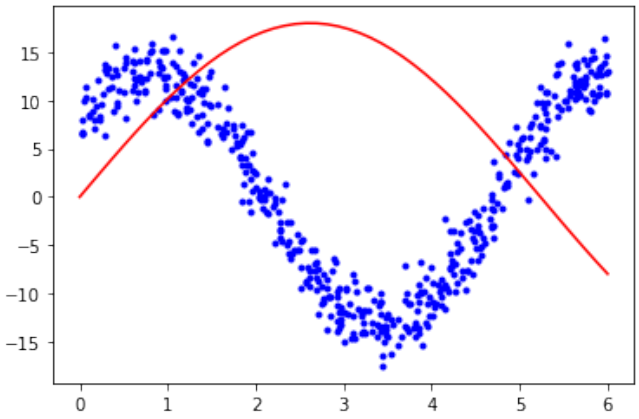

Figura 1.2: Gráfica con el punto inicial $\theta_0 = (18, 0{,}6, 0)$.

2. Ejercicio 1: SGD con momentum de Nesterov

La función implementada se anexa en un archivo Jupyter notebook, en este reporte solo mostraremos resultados y gráficos, así como las conclusiones que se nos solicitaron al principio.

Una vez que realizamos la implementación que se nos solicita del método de descenso de gradiente estocástico(SGD) con momentum de Nesterov, se nos solicita probar el algoritmo para el conjunto de datos que previamente se había generado, usando el punto inicial $\theta = (A, \omega, \phi) = (18, 0{,}6, 0)$, el valor $\alpha = 0{,}001$, una tolerancia $\tau_1 = 0{,}01$, $\tau_2 = 0{,}0001$, $v = (0, 0, 0)$, 5000 épocas (puede ajustar este valor si es necesario) y los tamaños de batch $m = 16$ y $m = 128$.
Para cada m, probar el algoritmo usando learning rate $\epsilon = 0{,}01$ y $\epsilon =$

0,0001

Resultados

Nuestro valor inicial fue $\theta = (A, \omega, \phi) = (18,\ 0.6,\ 0)$ que al evaluarlo en la función que mide el ajuste del modelo dio como resultado un valor de 311.2462, este resultado se espera sea menor una vez que tengamos la implementación.
Nos proporcionaron algunos parámetros de los cuales podiamos tomar varias opciones al momento de correr el programa, se observo que el mejor conjunto de parámetros fue:
Punto inicial $\theta = (18,\ 0.6,\ 0)$, el valor $\alpha = 0.001$, una tolerancia $\tau_1 = 0.01$, $\tau_2 = 0.0001$, $v = (0, 0, 0)$, 5000 épocas, tamaño de minibatch $m = 16$ y learning rate $\epsilon = 0.0001$.

Al ejecutar el programa obtuvimos un valor de $\theta_{final} = (13.0531,\ 1.2037,\ 0.60289)$. El valor en $L(\theta; \mathbf{X}, \mathbf{y})$ evaluado en el θ_{final} es de 4.097. La norma del gradiente de $L(\theta; \mathbf{X}, \mathbf{y})$ evaluado en el valor final de θ_{final} fue igual a 0.28641.
El número de iteraciones que se realizaron fue un total de 45088 con 1409 épocas. El tiempo de cómputo que se alcanzo es de 8.45494 segundos.
En la figura 2.1 se muestra la gráfica del modelo $f(x; \theta)$ usando el valor de θ final comparada con los datos generados al inicio.

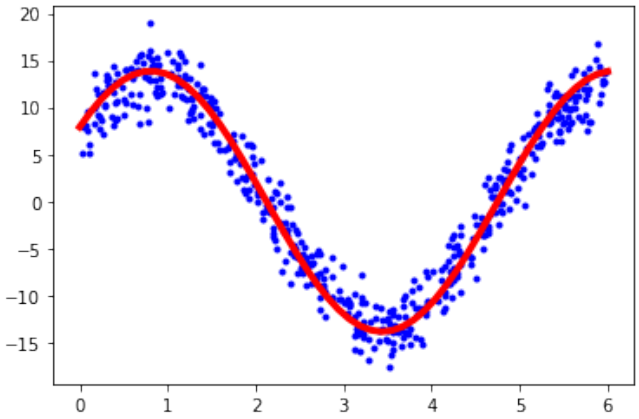

Figura 2.1: Gráfica con θ_{final}.

Observamos que el resultados de los parámetros con este algoritmo nos dio bueno resultados ya que la curva de color rojo si se ajusta muy bien a los datos.

En la figura 2.2 se muestra como el valor de $L(\theta; \mathbf{X}, \mathbf{y})$ fue disminuyendo con forme aumento el número de épocas. Al final observamos un tamaño en el error muy pequeño pero se podria conseguir proponiendo un nuevo τ_1 y τ_2.

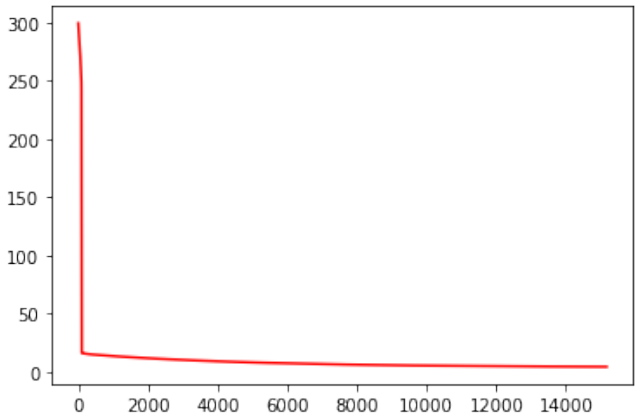

Figura 2.2: Gráfica de los valores de $L(\theta; \mathbf{X}, \mathbf{y})$.

En la figura 2.3 mostramos como los valores de los parámetros fueron disminuyendo con forme pasaron las iteraciones.

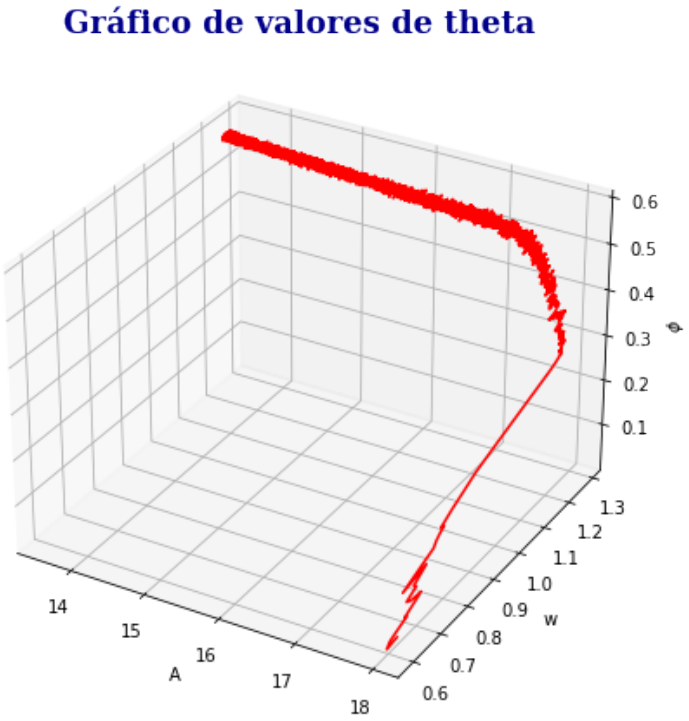

Figura 2.3: Gráfica de los valores de θ_k.

En esta primer vista se puede apreciar muy bien como fue disminuyendo el valor de ϕ pero para observar mejor los cambios se agrega la figura 2.4.

Figura 2.4: Gráfica de los valores de θ_k con rotación.

En la figura 2.5 se muestra como el valor del gradiente a lo largo de cada uno de los minibatch.

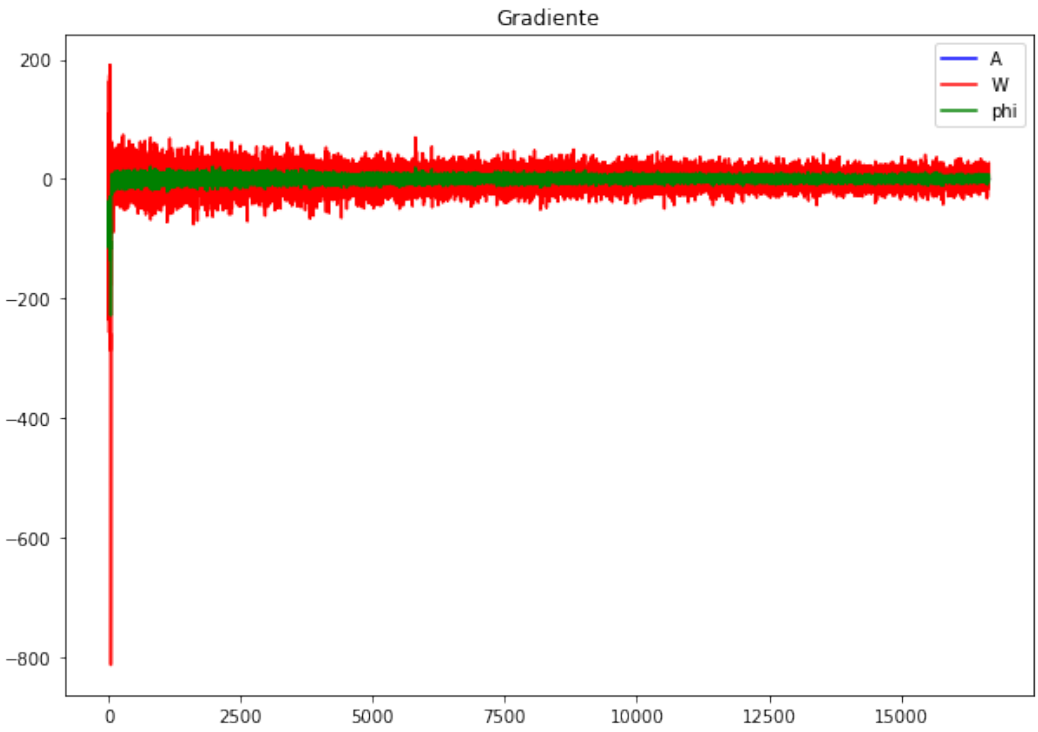

Figura 2.5: Gráfica del gradiente de L evaluada en los puntosθ_k.

Estos valores tienen cierta tendencia al valor cero pero no se observa muy bien todos ellos por lo que con fines ilustrativos agregamos en la figura 2.6 solo unos cuantos de ellos, en esta segunda gráfica se puede apreciar mejor los valores que nos dio el gradiente. Se fue tomando tercias de valores cada 50 datos para disminuir el número.

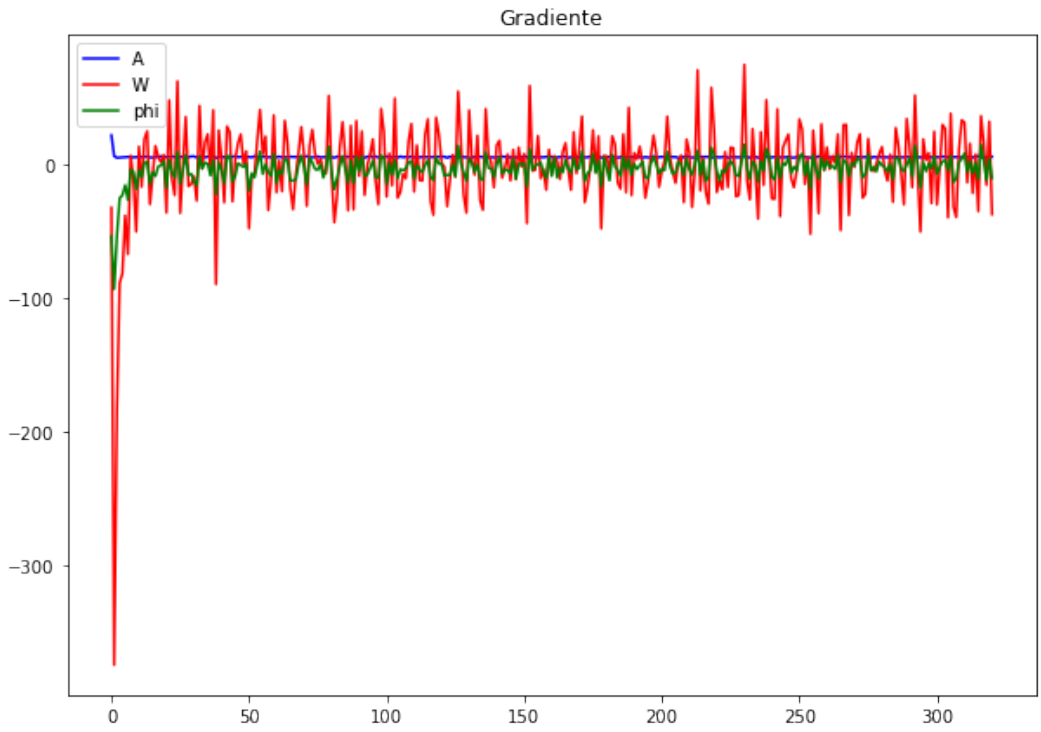

Figura 2.6: Gráfica del gradiente con pocos datos.

Como se menciono al principio el conjunto de parámetros nos podia ayudar a observar distintos casos por lo que muchos de ellos no se tomaron en cuenta y solo se mostro el que se considero mejor, con fines informativos en la siguiente tabla 2.1 se anexan los resultados más importante de cada conjunto de parámetros.
Conjunto 1: $m = 16$ y $\epsilon = 0.01$
Conjunto 2: $m = 128$ y $\epsilon = 0.0001$
Conjunto 3: $m = 128$ y $\epsilon = 0.01$

N.Conjunto	Valor de θ	Nor-Gra	Iter	Tiempo
Conjunto 1	(-0.4517, -28.4473, 10.5762)	6.1206	160000	29.4176
Conjunto 2	(14.067, 1.1974, 0.599)	0.9827	15528	21.3364
Conjunto 3	(1.57689, 16.4933, 3.81656)	6.2549	20000	28.5344

Cuadro 2.1: Posibles casos de parámetros

Dado los resultados podemos decir que el segundo conjunto es bueno pero el tiempo es bastante grande comparado con el que presentamos al principio, de los otros dos conjuntos no podemos decir mucho ya que el tiempo es mucho y sus resultados respecto a los valores de los parámetros no son buenos.

3. Ejercicio 2: Algoritmo RMSProp

En este ejercicio se hace la implementación del algoritmo RMSProp, nos piden que repitamos las pruebas que mostramos en el ejercicio 1, es decir, con los mismos parámetros que nos mencionaron pero usando $\rho = 0.9$.

Resultados

Nuestro valor inicial fue $\theta = (A, \omega, \phi) = (18,\ 0.6,\ 0)$ que al evaluarlo en la función que mide el ajuste del modelo dio como resultado un valor de 311.2462, este resultado no cambio debido a que se ocupo la misma función.

Al igual que en el ejercicio anterior solo mostraremos resultados del mejor conjunto de parámetros, lo cuales fueron: Punto inicial $\theta = (18,\ 0.6,\ 0)$, el valor $\rho = 0.9$, una tolerancia $\tau_1 = 0.01$, $\tau_2 = 0.0001$, $v = (0, 0, 0)$, 5000 épocas, tamaño de minibatch $m = 128$ y learning rate $\epsilon = 0.01$.
Al ejecutar el programa obtuvimos un valor de $\theta_{final} = (13.3612,\ 1.2066,\ 0.59828)$. El valor en $L(\theta; \mathbf{X}, \mathbf{y})$ evaluado en el θ_{final} es de 4.2669. La norma del gradiente de $L(\theta; \mathbf{X}, \mathbf{y})$ evaluado en el valor final de θ_{final} fue

igual a 6.95779.
El número de iteraciones que se realizaron fue un total de 32000 en 1000 épocas. El tiempo de cómputo que se alcanzo es de 5.8845 segundos.
En la figura 3.1 se muestra la gráfica del modelo $f(x;\theta)$ usando el valor de θ final comparada con los datos generados al inicio.

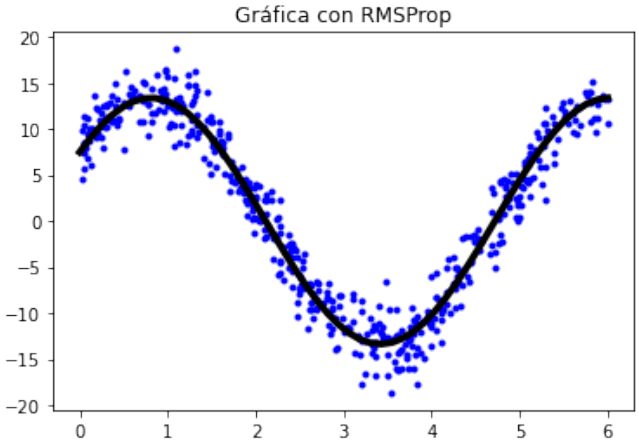

Figura 3.1: Gráfica con θ_{final}.

En la figura se puede apreciar que los parámetros proporcionados fueron los apropiados ya que nos ajusta bien la curva de color rojo.

En la figura 3.2 la curva de color rojo cae más rapido, es decir, se aproxima a cero en menos número de épocas, esto indica que el algoritmo dio buenos resultados en pocos pasos.

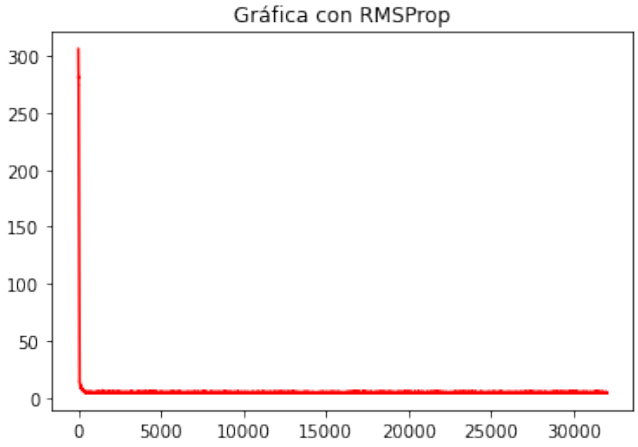

Figura 3.2: Gráfica de los valores de $L(\theta; \mathbf{X}, \mathbf{y})$.

En la figura 3.3 los parámetros fueron disminuyendo de valor pero a diferencia del anterior algoritmo los valores tienen más movimiento entre ciertos valores.

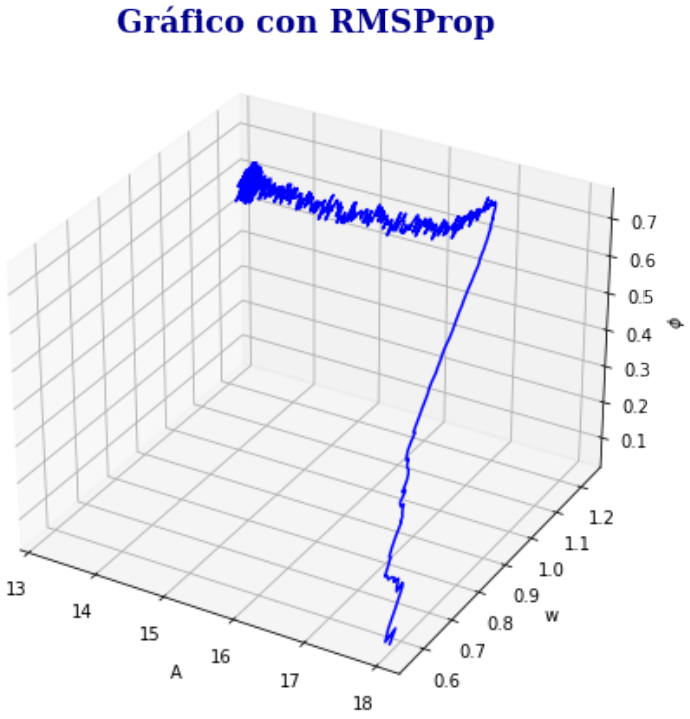

Figura 3.3: Gráfica de los valores de θ_k.

Se agrego igual un gráfico con ligera rotación en la figura 3.4, para como fue este cambio en los valores de los parámetros.

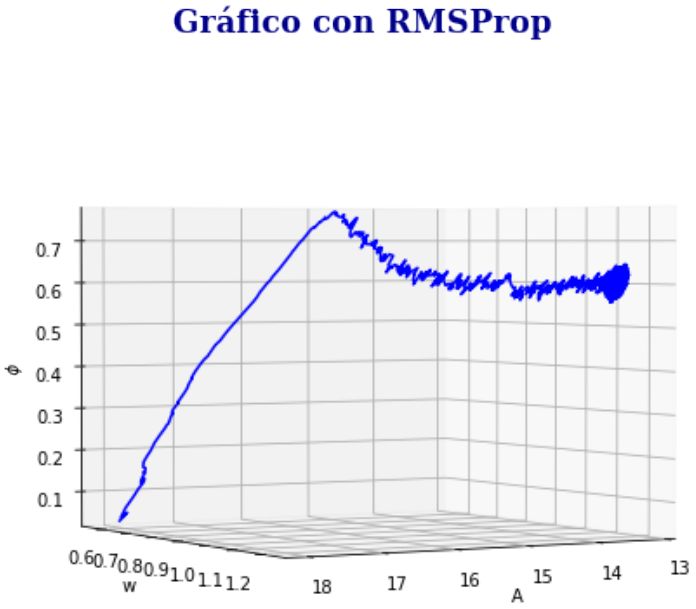

Figura 3.4: Gráfica de los valores de θ_k con rotación.

En la figura 3.5 mostramos como el valor del gradiente fue cambio a lo largo de todos los minibatches. Se observa que hay cierta tendencia al valor cero.

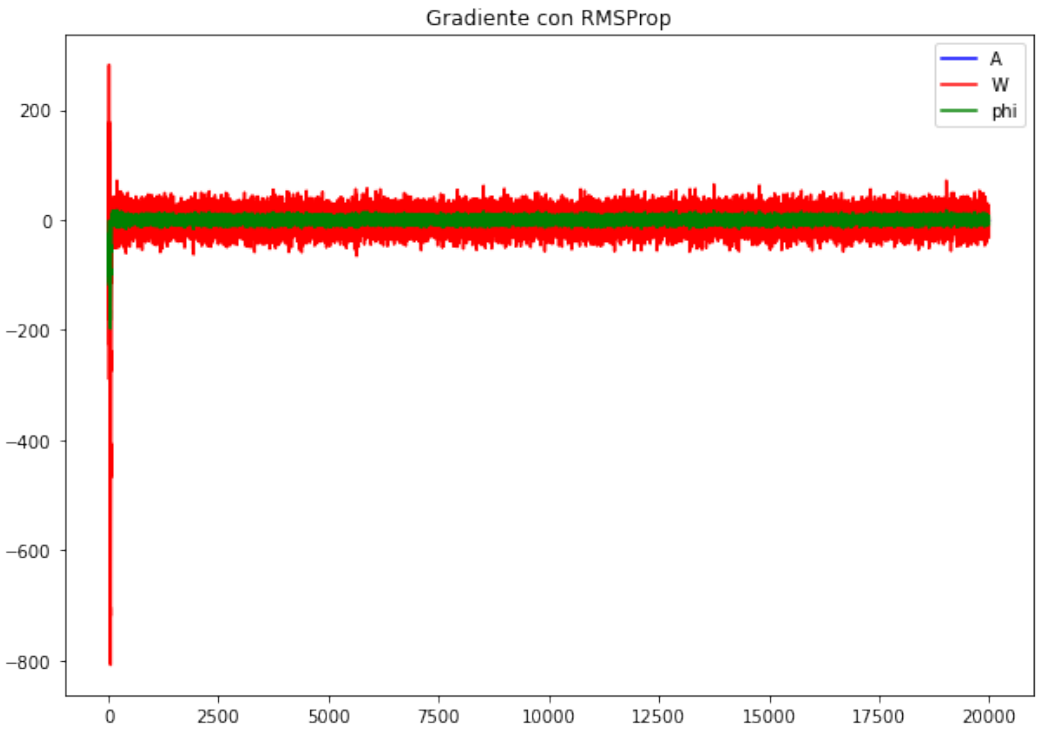

Figura 3.5: Gráfica del gradiente de L evaluada en los puntosθ_k.

Debido a que no es tan sencillo observar que sucede con los tres parámetros solo con fines ilustrativos anexamos una segunda gráfica en la figura 3.6 donde seleccionamos cierto datos.

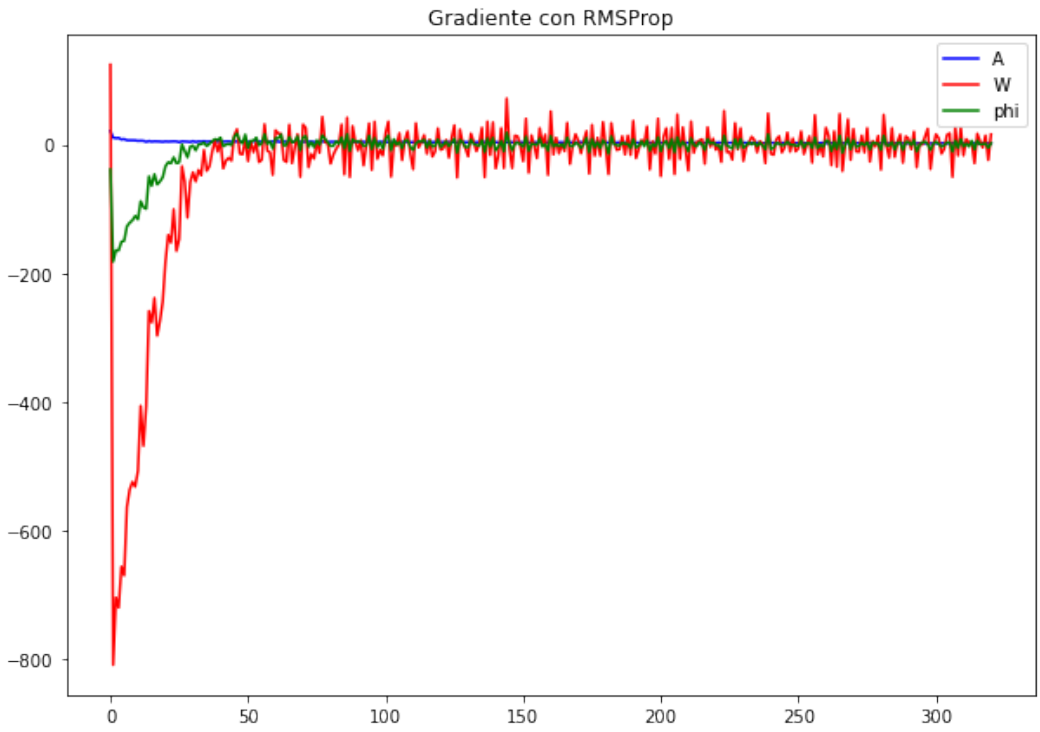

Figura 3.6: Gráfica del gradiente con pocos datos.

En la selección de los datos se fue tomando uno cada 50 para observar como se comportan los parámetros.
En la gráfica los tres parámetros tienden a cero y solo el valor del parámetro A es el que rápido se queda como una constante mientras que el valor de W si varia mucho, ϕ no tiene tantos cambios a lo largo de las iteraciones.

Como se menciono al principio el conjunto de parámetros nos podia ayudar a observar distintos casos por lo que muchos de ellos no se tomaron en cuenta y solo se mostro el que se considero mejor, con fines informativos

N.Conjunto	Valor de θ	Nor-Gra	Iter	Tiempo
Conjunto 1	(13.34, 1.214, 0.615)	5.91	32000	23.775
Conjunto 2	(17.60, 0.679, -0.339)	25.995	4000	5.7639
Conjunto 3	(14.8705, 0.7163, -0.5883)	48.60	32000	6.1703

Cuadro 3.1: Posibles casos de parámetros

en la siguiente tabla 3.1 se anexan los resultados más importante de cada conjunto de parámetros.
Conjunto 1: $m = 16$ y $\epsilon = 0.01$
Conjunto 2: $m = 128$ y $\epsilon = 0.0001$
Conjunto 3: $m = 16$ y $\epsilon = 0.0001$

Para los tres conjuntos observamos que los valores de los parámetros no fueron tan buenos más que para el conjunto 1, donde si se observa buenos valores pero comparados con el que se presento no fue tan bueno y el tiempo es más grande por lo que se podria decir que en segundo lugar queda este conjunto.

4. Conclusión

Observamos que entre los dos métodos tienen sus ventajas, en el primer método el número de iteraciones fue ligeramente mayor y al segundo no le tomo tantas iteraciones llegar a un punto donde se cumplan las condiciones de paro. Los gráficos reflejan que el segundo método disminuyo mucho el valor del error en las primeras iteraciones y el primero método no lo hizo tan rapido.
El algoritmo de RMSProp fue muy diferente respecto al algoritmo de SGD ya que en los conjuntos de parámetros necesito el tamaño del minibatch de 128 elementos y un tamaño de paso inicial de 0.01 y SGD necesito menos elementos en el minibatch y un tamaño de paso de 0.0001, esa diferencia

es importante ya que al correr las implementaciones en numerosas veces con los mismo conjuntos de parámetros se observo mucha diferencias en los resultados y se trato de observar resultados similares bajos mismos parámetros pero no fue así. Por lo que podemos decir que el segundo método(RMSProp) fue mejor que el primero pero tomando criterios como el tiempo, el valor de la función L a lo largo de las épocas y el número de iteaciones.