Universidad Autónoma de Querétaro

Facultad de Ingeniería
División de Investigación y Posgrado

Reporte 3

Gradiente Descendente

Maestría en Ciencias en Inteligencia Artificial
Optativa de especialidad II - Deep Learning

Aldo Cervantes Marquez
Expediente: 262775
Profesor: Dr. Sebastián Salazar Colores

Santiago de Querétaro, Querétaro, México
Semestre 2022-2
15 de Agosto de 2022

Índice

1. Introducción

La presente práctica consiste en realizar regresión lineal y gradiente descendente para obtener el error menor posible mediante esta técnica y tener los valores que mejor se ajusten al modelo solicitado. Los modelos constan de dos bases de datos, archivadas con los nombres: *advertising.csv* y *Articulos_ml.csv*. El primero consta de una tabla que muestra la inversión de la publicidad por distintos medios (televisión, radio y papel periódico) y sus ventas obtenidas [1]. El segundo consta de la popularidad de los artículos con base en sus palabras, cantidad de imágenes, cantidad de likes, comentarios, y la edad del articulo.
Todos los elementos teóricos de esta sección fueron obtenidos de [2, 3] y de los apuntes de la clase.

1.1. Regresión Lineal

Consiste en el uso de una función lineal (grado 1) para poder aproximar un conjunto de puntos relacionados entre sí (véase Figura 1).

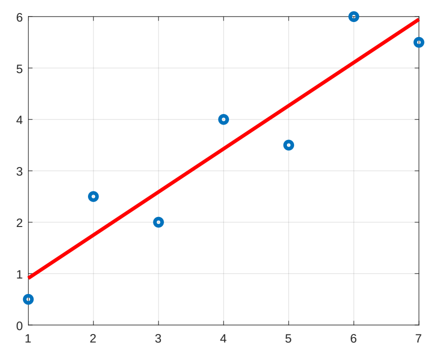


Figura 1: Regresión lineal

Obteniendo una ecuación de recta como se muestra a continuación.

$$h_\theta(x) = \theta_1 x + \theta_0 \tag{1}$$

Por lo que los parámetros a obtener son θ_0 y θ_1 con el fin de ajustar la pendiente y ordenada al origen de la recta.

Existen varios indicadores de la correlación de nuestra base de datos, en donde podremos observar que tan relacionadas se encuentran los datos antes de realizar el modelo.

1.2. Forma Matricial de la Hipótesis de Ecuación Lineal

Partiendo de la ecuación (1) podemos acomodar las variables θ_0 y θ_1 de la siguiente manera:

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \tag{2}$$

Del mismo modo podemos agrupar las características de x en cualquier dimensión a partir de la siguiente matriz:

$$X = \begin{bmatrix} x_1^0 & x_1^1 & . & . & x_1^n \\ x_2^0 & x_2^1 & . & . & x_2^n \\ . \\ . \\ . \\ x_m^0 & x_m^1 & . & . & x_m^n \end{bmatrix} \tag{3}$$

Donde $x^0_{1,2,3,..,n}$ es igual a 1 puesto que representa al termino independiente de la ecuación (θ_0). Por lo que generalizando y por producto de matrices, es posible simplificar la operación de la siguiente manera la hipótesis h_θ:

$$h_\theta = \begin{bmatrix} x_1^0 & x_1^1 & . & . & x_1^n \\ x_2^0 & x_2^1 & . & . & x_2^n \\ . \\ . \\ . \\ x_m^0 & x_m^1 & . & . & x_m^n \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ . \\ . \\ . \\ \theta_n \end{bmatrix} = \begin{bmatrix} h_0 \\ h_1 \\ . \\ . \\ . \\ h_m \end{bmatrix} \tag{4}$$

1.3. Indicadores Estadísticos

Primeramente se tiene la covarianza que indica una variación conjunta de los datos y se representa en la siguiente ecuación.

$$Cov(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - E[X])(y_i - E[Y]) \tag{5}$$

Mientras su valor absoluto sea mayor, mayor correlación habrá, ya sea positiva o negativa (la pendiente de la rectas.

Posteriormente se tiene el coeficiente de correlación de Pearson, el cual normaliza los datos y se puede obtener un valor normalizado entre -1 y 1 donde 1 significa que hay una correlación positiva fuerte y -1 una correlación negativa fuerte, mientras se aproxima a 0 no existe correlación lineal entre los datos. Se representa mediante la siguiente ecuación.

$$\rho_{X,Y} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y} = \frac{\mathrm{Cov}(X,Y)}{\sqrt{\mathrm{Var}(X)\,\mathrm{Var}(Y)}} \tag{6}$$

donde:

- σ_{XY} es la covarianza de (X, Y)
- σ_X es la desviación estándar de la variable X
- σ_Y es la desviación estándar de la variable Y

1.3.1. Cálculo del error

Existen criterios para observar que tan bueno es el ajuste hecho, y se basan en los errores residuales de los datos disponibles evaluados en los puntos que se tienen. Se tiene el error medio cuadrático (MSE), el cual se muestra a continuación.

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2 \tag{7}$$

También se tiene el error absoluto medio (MAE).

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} |h_\theta(x_i) - y_i| \tag{8}$$

Donde h_0 es la predicción realizada en el valor x_i

La idea principal es minimizar este error a 0 y por lo tanto tener el mejor ajuste posible.

1.4. Gradiente Descendente

El uso de gradiente descendente es método para obtener valores mínimos de funciones mediante el uso de criterios de derivadas (criterio de primera derivada) donde se sabe que los ceros (o raíces) de una función (véase Figura 2).

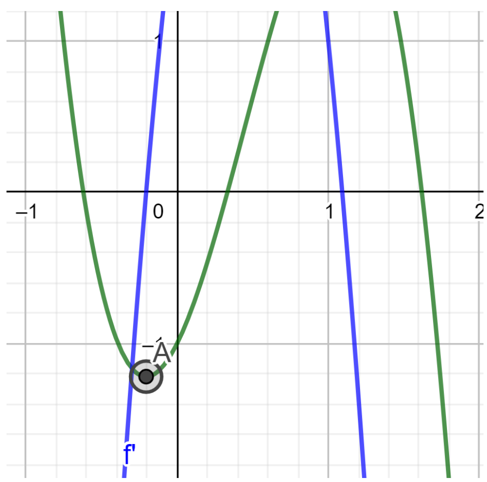


Figura 2: Criterio de derivada.

Como se observa, la función verde $f(x)$ en el punto A se tiene un mínimo y en ese mismo punto se tiene en la función derivada azul $f'(x)$ tiene un cruce con el eje x.

A continuación, se muestra un pseudocódigo del algoritmo desarrollado.

Algoritmo 1 Proceso de gradiente descendiente.

Inicio
Valor arbitrario de θ_0 y θ_1
Repetir (**hasta** que se cumpla **condicion** de paro)
Buscar dirección $\boldsymbol{v}$ de decrecimiento en el campo $\boldsymbol{x}$
Variación valores de la dirección
$\boldsymbol{x} \rightarrow \boldsymbol{x} + \boldsymbol{\varepsilon v}$

Gráficamente se puede observar como la tasa de cambio de la derivada y su movimiento a través de la función desplazándose cierta cantidad de unidades hacia el mínimo (véase Figura 3).

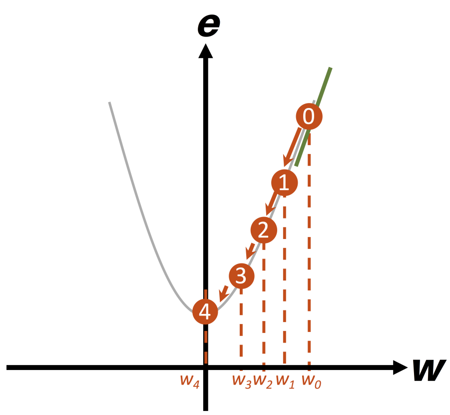


Figura 3: Criterio de derivada.

Matemáticamente y computacionalmente se puede definir la formula para un numero de épocas (iteraciones) determinado.

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \tag{9}$$

donde α es la taza de aprendizaje.

1.5. Gradiente Descendiente con matrices.

En este caso se usará el gradiente descendiente con el criterio de minimizar la función de costo $J(\theta)$ que será el MSE mencionado anteriormente.

Sabiendo la derivada de MSE se obtiene la ecuación (10)

$$\begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_0} \\ \frac{\partial J(\theta)}{\partial \theta_1} \\ . \\ . \\ . \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{bmatrix} = \frac{1}{m}(h_\theta - Y)X^T = \frac{1}{m}(X.\theta - Y)X^T \tag{10}$$

La cual describe la derivada parcial con respecto a cada θ_n. Debido a las operaciones matriciales, es más cómodo calcular para n dimensiones de θ.

La formula completa implica el uso de la actualización de θ, por lo que al agregarle el factor α se obtiene la ecuación (11).

$$\begin{bmatrix}\theta_0\\ \theta_1\\ .\\ .\\ .\\ \theta_n\end{bmatrix} = \begin{bmatrix}\theta_0\\ \theta_1\\ .\\ .\\ .\\ \theta_n\end{bmatrix} - \alpha \begin{bmatrix}\frac{\partial J(\theta)}{\partial \theta_0}\\ \frac{\partial J(\theta)}{\partial \theta_1}\\ .\\ .\\ .\\ \frac{\partial J(\theta)}{\partial \theta_n}\end{bmatrix} \tag{11}$$

1.6. Método de Mínimos Cuadrados

Este método permite obtener la distancia entre puntos de sus derivadas y se igualan a 0 para resolver un sistema de ecuaciones que se simplifican en las siguientes formulas.

$$a_1 = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2} \tag{12}$$

$$a_0 = \bar{y} - a_1\bar{x} \tag{13}$$

donde:

- n es la cantidad de valores.
- $\bar{y}$ es la media aritmética de la variable dependiente.
- $\bar{x}$ es la media aritmética de la variable independiente.

2. Justificación

El uso del análisis de datos para conocer el comportamiento y hacer predicciones de los mismos se ha convertido en un tema de gran importancia para realizar acciones y tomas de decisiones importantes, y de esta manera, obtener resultados favorables para el problema que se esté resolviendo. Obteniendo una posibilidad de relacionar datos que tal vez, a simple vista, no se puede encontrar relación [4].

Estos métodos de regresión requieren de minimizadores y técnicas que permitan obtener los mejores parámetros de ajuste del modelo. Por lo que el uso de algoritmos de inteligencia artificial para minimizar los errores e incrementar la correlación entre los elementos, es una opción viable y el algoritmo de gradiente descendente es útil para la resolución de problemas lineales con una variable independiente.

3. Resultados

Los resultados se observan de manera más concreta con en el anexo del código. Sin embargo, se mostrarán los resultados de los análisis estadísticos de las dos bases de datos con una corta interpretación de los mismos.

3.1. Visualización de datos

En esta parte se muestran los diagramas de dispersión de las dos bases de datos, en la base de datos de *Advertising* (Figura 4) se grafican las variables con respecto de 'Sales'. En el diagrama de dispersión de la base de datos *Artículos* (Figura 5) se grafican las variables con respecto a la variable *# Shares*.

3.1.1. Advertising

En la Figura 4 se observa por simple inspección que las primeras dos graficas (*TV vs Sales* y *Radio vs Sales*) tienen cierta tendencia a la alza conforme se incrementa el eje x (una relación positiva). Sin embargo en la gráfica de *Newspaper vs Sales* se observa poca o nula correlación entre las variables.

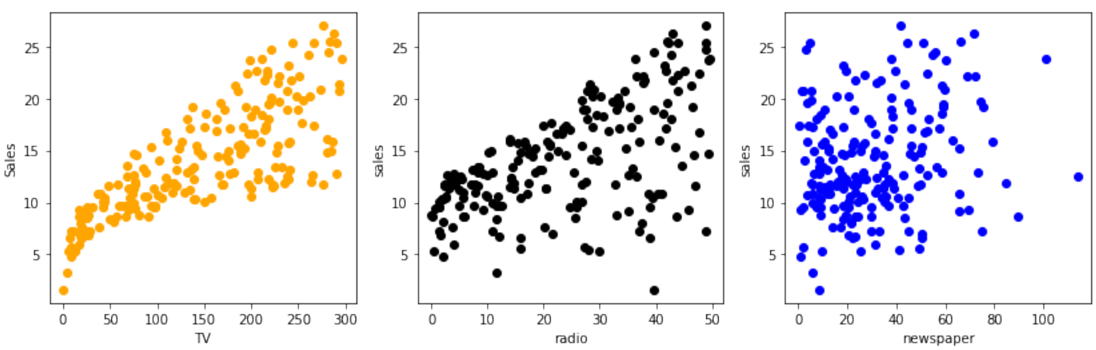


Figura 4: Diagrama de dispersión advertising.

3.1.2. Artículos

En la Figura 5 se observa por simple inspección una nula correlación de los datos y mucha dispersión de los mismos.

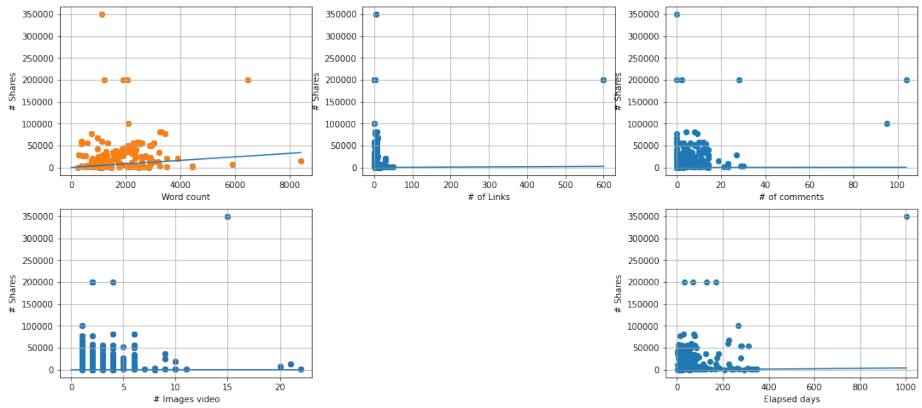


Figura 5: Diagrama de dispersión artículos.

3.2. Covarianza

Para el calculo de este indicador fue necesario aplicar la ecuación (5) y se realizaron tablas para poder observar variable a variable sus correlaciones de una manera ordenada.

3.2.1. Advertising

En la Tabla 1 se observa que los valores entre las mismas variables son bastante altos, a excepción de *Sales*, sin embargo esto es lógico debido a que al compararse entre las mismas debe haber una relación 1:1. Ente las demás variables con respecto a *Sales* se observa que únicamente *TV* tiene una covarianza alta. Aun así, no es concluyente saber si las covarianzas implican una fuerte correlación.

Tabla 1: Covarianza de las variables de Advertising.

	TV	Radio	Newspaper	Sales
TV	7370.9498	69.8624	105.9194	350.3901
Radio	69.8624	220.4277	114.4969	44.6356
Newspaper	105.91945	114.4969	474.3083	25.9413
Sales	350.3901	44.6356	25.9413	27.2218

3.2.2. Artículos

De igual manera, en la Tabla 2 se observa que los valores entre las mismas variables son demasiado altos, además de que las covarianzas de las variables con *# Shares* es muy alto, por lo que se esperaría una buena correlación entre variables independientes y la dependiente. Sin embargo, esto aún no es concluyente para saber lo bien que la propuesta de modelo puede ajustarse.

Tabla 2: Covarianza de las variables de Artículos.

	Word count	# of Links	# of comments	# Images video	Elapsed days	# Shares
Word count	1303979.8815	18716.6434	1533.877	1810.273	-21851.3013	7025616.2211
# of Links	18716.6434	2234.6065	67.4285	3.40733	319.7388	591633.7413
# of comments	1533.8777	67.4285	150.5486	-1.3751	52.7640	143677.2432
# Images video	1810.2739	3.4073	-1.3751	11.6847	82.0598	9224.9277
Elapsed days	-21851.3013	319.7388	52.7640	82.0598	13073.0719	1614728.9877
# Shares	7025616.2211	591633.731	143677.2432	9224.9277	1614728.9877	1884255057.70

3.3. Correlación de Pearson

Con la formula de la ecuación (6) fue posible obtener los siguientes diagramas (véase Figuras 6 y 7).

3.3.1. Advertising

Como se observa en la Figura 6 fue posible apreciar la alta correlación entre *TV* y *Sales* y una correlación muy baja con *Newspaper*, esto permitirá darnos una idea de los resultados del ajuste. La correlación entre las variables independientes (*TV,Radio,Newspaper*) es prácticamente nula tendiendo a 0.

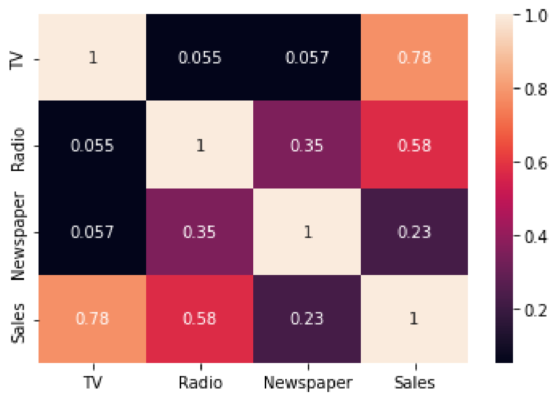


Figura 6: Diagrama de calor de las variables de advertising.

3.3.2. Artículos

En la Figura 7 se obseva una correlación muy pequeña entre todas las variables con respecto a *# Shares* siendo el mayor 0.33, algo muy pequeño comparado con la otra base de datos, por lo que se puede decir que no se podrá ajustar el modelo tan bien como en la otra base de datos.

3.4. Comparación de valores obtenidos en la regresión

Cabe mencionar que manualmente fueron ajustados los datos observando los errores como se comportaban conforme se iban moviendo. Las tablas cuentan con apartados de cada método utilizado,

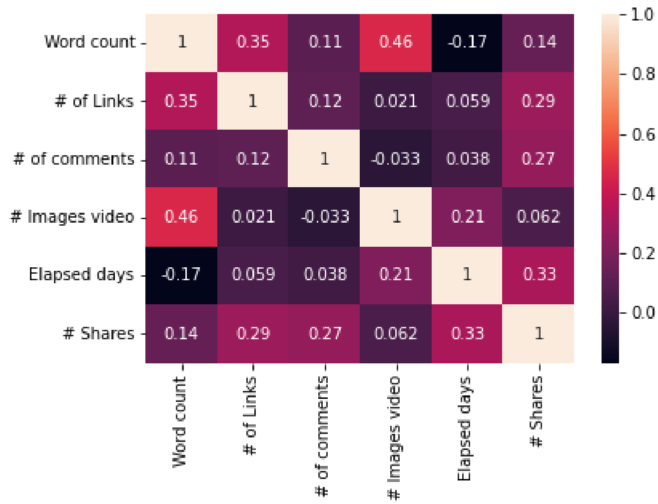


Figura 7: Diagrama de calor de las variables de artículos.

en este caso se utilizó el método de mínimos cuadrados como comparativa alterna a una estimación de la reducción de la función de coste.

Mediante la ecuación (9) de aplicaron las derivadas parciales y se obtuvieron las siguientes ecuaciones.
Tomando en cuenta que: $h_\theta(x) = \theta_1 x + \theta_0$

Derivadas Parciales de MSE

$$\theta_0 := \theta_0 - \alpha \frac{1}{2m} \frac{\partial}{\partial \theta_0} \sum_{i=1}^{m} |h_\theta(x_i) - y_i| \tag{14}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{2m} \frac{\partial}{\partial \theta_1} \sum_{i=1}^{m} |h_\theta(x_i) - y_i| \tag{15}$$

Derivadas Parciales de MAE

$$\theta_0 := \theta_0 - \alpha \frac{1}{2m} \sum_{i=1}^{m} \frac{h_\theta(x_i) - y_i}{|h_\theta(x_i) - y_i|} \tag{16}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{2m} \sum_{i=1}^{m} \frac{h_\theta(x_i) - y_i}{|h_\theta(x_i) - y_i|} \tag{17}$$

3.4.1. Advertising

Para α=0.001 comparado con los otros métodos Tabla 3.

Tabla 3: Resultados de MSE y MAE con $\alpha = 0.001$ y comparado con otros métodos.

TV vs Sales			
	Manual	Gradiente Descendiente	Mínimos cuadrados
θ_0	0.05	0.05	7.03259
θ_1	0.07	0.07	0.0475
iteraciones n	-	500	-
Taza de aprendizaje α	-	0.001	-
MSE	13.8761	13.8761	5.2563
θ_0	0.05	6.3058	7.03259
θ_1	0.07	0.0521	0.0475
iteraciones n	-	1000	-
MAE	2.3473	1.26847	1.2749

Radio vs Sales			
	Manual	Gradiente Descendiente	Mínimos cuadrados
θ_0	7	9.31	9.3116
θ_1	0.27	0.2034	0.20249
iteraciones n		400	-
Taza de aprendizaje α	-	0.001	-
MSE	9.820	9.0465	9.0461
θ_0	7	9.7928	9.3116
θ_1	0.27	0.2235	0.20249
iteraciones n	-	500	-
MAE	1.8139	1.6195	1.6601

	Newspaper vs Sales		
	Manual	Gradiente Descendiente	mínimos cuadrados
θ_0	5	12.6584	12.3514
θ_1	0.3	0.04721	0.0546
iteraciones n	-	4000	-
Taza de aprendizaje α	-	0.001	-
MSE	27.0468	12.6584	12.83701
θ_0	5	10.6069	12.3514
θ_1	0.3	0.08515	0.0546
iteraciones n	-	3000	-
MAE	2.8329	2.0464	2.07327

Para α=0.01 y α=0.1

Tabla 4: Resultados de MSE y MAE con $\alpha = 0.01$ y $\alpha = 0.1$.

TV vs Sales		
	Gradiente Descendiente	Gradiente Descendiente
θ_0	0.05	0.7
θ_1	0.07	0.01
iteraciones n	500	100
Taza de aprendizaje α	0.01	0.1
MSE	13.8761	80.6590
θ_0	0.77	0.034
θ_1	0.07516	0.04
iteraciones n	1000	1000
MAE	1.9805	4.0533

Radio vs Sales		
	Gradiente Descendiente	Gradiente Descendiente
θ_0	7	0.7726
θ_1	0.27	0.43142
iteraciones n	800	400
Taza de aprendizaje α	0.01	0.1
MSE	9.04652	19.955
θ_0	9.6917	12.83927
θ_1	0.2275	0.11596
iteraciones n	500	500
MAE	1.6194	1.87285

Newspaper vs Sales		
	Gradiente Descendiente	Gradiente Descendiente
θ_0	0.9714	0.65302
θ_1	0.16875	0.32826
iteraciones n	500	500
Taza de aprendizaje α	0.01	0.1
MSE	47.0730	36.07408
θ_0	12.6456	13.00891
θ_1	0.0370	0.04511
iteraciones n	500	500
MAE	2.07428	2.1007

3.4.2. Artículos

Tabla 5: Resultados de MSE y MAE con $\alpha = 0.01$ y $\alpha = 0.1$.

TV vs Sales		
	Gradiente Descendiente	Gradiente Descendiente
θ_0	0.05	0.7
θ_1	0.07	0.01
iteraciones n	500	100
Taza de aprendizaje α	0.01	0.1
MSE	13.8761	80.6590
θ_0	0.77	0.034
θ_1	0.07516	0.04
iteraciones n	1000	1000
MAE	1.9805	4.0533

Radio vs Sales		
	Gradiente Descendiente	Gradiente Descendiente
θ_0	7	0.7726
θ_1	0.27	0.43142
iteraciones n	800	400
Taza de aprendizaje α	0.01	0.1
MSE	9.04652	19.955
θ_0	9.6917	12.83927
θ_1	0.2275	0.11596
iteraciones n	500	500
MAE	1.6194	1.87285

Newspaper vs Sales		
	Gradiente Descendiente	Gradiente Descendiente
θ_0	0.9714	0.65302
θ_1	0.16875	0.32826
iteraciones n	500	500
Taza de aprendizaje α	0.01	0.1
MSE	47.0730	36.07408
θ_0	12.6456	13.00891
θ_1	0.0370	0.04511
iteraciones n	500	500
MAE	2.07428	2.1007

Tabla 6: Resultados de MSE y MAE con $\alpha = 0.001$ en base de datos de artículos.

Word count vs # Shares			
	Manual	Gradiente Descendiente	Mínimos cuadrados
θ_0	4	0.56462	18205.753
θ_1	14	0.526300	5.3878256
iteraciones n	-	1000	-
Taza de aprendizaje α	-	0.001	-
MSE	968979423.0434	1297175239.60	917466962.77
θ_0	4	0.324004	18205.753
θ_1	14	9.07553	5.3878256
iteraciones n	-	100	-
MAE	11961.695	11546.92	12294.816

# of Links vs # Shares			
	Manual	Gradiente Descendiente	Mínimos cuadrados
θ_0	15	20512.028	25369.8186
θ_1	100	275.7189	264.759
iteraciones n	-	100	-
Taza de aprendizaje α	-	0.001	-
MSE	1251989102.102	869861511.48	858441882.613
θ_0	15	37.15386	25369.8186
θ_1	100	334.2224	264.759
iteraciones n	-	500	-
MAE	13568.499	12834.96127	238297.0974

# of comments vs # Shares			
	Manual	Gradiente Descendiente	mínimos cuadrados
θ_0	10	22489.1972	21232.27
θ_1	100	1068.292	954.35786
iteraciones n	-	100	-
Taza de aprendizaje α	-	0.001	-
MSE	1293607682.58	871232088.352	868141888.33
θ_0	10	253.2806	21232.27
θ_1	100	1473.8691	954.35786
iteraciones n	-	1500	-
MAE	13665.217	12102.95794	859505.9579

# Images video vs # Shares			
	Manual	Gradiente Descendiente	Mínimos cuadrados
θ_0	4000	1123.7842	25050.2914
θ_1	-1000	4136.49876	789.4874
iteraciones n	-	10	-
Taza de aprendizaje α	-	0.001	-
MSE	986368778.795	1065447269.598	932656940.725
θ_0	4000	977.1053	25050.2914
θ_1	-1000	2668.7454	789.4874
iteraciones n	-	2500	-
MAE	14184.8695	12673.3034	712350.5868

Elapsed days vs # Shares			
	Manual	Gradiente Descendiente	Mínimos cuadrados
θ_0	800	0.423051	15828.4705
θ_1	250	0.44598	123.5156
iteraciones n	-	100	-
Taza de aprendizaje α	-	0.001	-
MSE	944522033.76	1324882556.92	837173046.711
θ_0	800	61.57293	15828.4705
θ_1	250	70.72181	123.5156
iteraciones n	-	1000	-
MAE	14627.6583	13137.3758	107053.9893

4. Conclusiones

El uso del método de gradiente descendiente permitió disminuir los valores de la función de coste y se observaron varios puntos que fueron de interés al momento de realizar la práctica:

1. Se observó que se obtenian resultados más estables si α era un valor aun más pequeño

2. Si se disminuye la tasa de aprendizaje, entonces se debe aumentar la cantidad de épocas o iteraciones

3. El punto inicial (es decir $\theta_{0inicial}$,$\theta_{1inicial}$) influye mucho en la cantidad de épocas necesarias para llegar a una solución semejante a que si iniciara mas cerca (numéricamente) a que si se pusieran los resultados manuales y en el resultado final.

4. En la base de datos de *Articulos* habían datos que estaban como **nan** entonces se decidió llenar esos espacios vacíos con 0. Sin embargo, también era una opción viable eliminar las filas o intercambiar los valores por el promedio de la columna. Por lo tanto el resultado se podría ver afectado.

5. En los casos con poca correlación de Pearson, se observó que las funciones MSE y MAE fueron muy altas debido a las no linealidades de los conjuntos de datos.

6. El algoritmo cuando tiende a oscilar, es decir, que el comportamiento de MSE o MAE a lo largo de las épocas es muy variable, se optó por tomar el valor menor se la sucesión. De este modo al meno se aseguraba tener el valor menor de todos los valores calculados.

7. En comparación con calcular manualmente los valores, es mas fiable este método siempre y cuando se conozcan los coeficientes de Pearson y considero, que ese valor es muy útil para saber que tan bien podrá ajustarse la recta al conjunto de puntos.

8. En comparación con el método de mínimos cuadrados, creo que hay mucho parentesco en cuanto a su uso, pero el algoritmo de gradiente descendiente tiene la ventaja de que puede obtener diferentes resultados, que pueden ser mejores que los calculados por los mínimos cuadrados, por lo que considero mas flexible al algoritmo de gradiente descendiente.

Esta práctica es un complemento de la practica 2, por lo que se añadirán un par de conclusiones extra.

1. El uso de matrices reduce ampliamente las líneas de código, así como también evita ocupar ciclos for para los cálculos.

2. Es posible añadir más variables independientes y de este modo, se incrementan las θ solución, por lo que es más facil visualizar y aplicar los conceptos anteriormente mencionados.

3. Se obtuvieron los mismos resultados que programándolo con funciones.

4. Los programas se encuentran en la Notebook anexa y en el anexo de código, donde se muestra como se aplicaron las formulas. Uno para evaluar la función y la otra para realizar el gradiente descendiente.

Referencias

[1] "Conjunto de datos Advertising.csv: analiza la relación entre la inversión en medios publicitarios y las ventas - programador clic." https://programmerclick.com/article/7701963942/.

[2] "Introducción a Machine Learning." https://unidad.gdl.cinvestav.mx/doc/investigacion/computacion/Introduccion-Machine-Learning.pdf.

[3] S. C. Chapra, R. P. Canale, and ProQuest, *Métodos numéricos para ingenieros*. Distrito Federal: McGraw-Hill Interamericana, 5 ed., 2007. OCLC: 1105521785.

[4] J. Gil, "Tableau: La importancia del análisis de datos." https://www.hiberus.com/crecemos-contigo/tableau-la-importancia-del-analisis-de-datos/, Feb. 2021.

regresión lineal con matrices

August 22, 2022

Coodigo de covarianza relacion lineal

[2]:
```
from google.colab import drive

drive.mount('/content/drive', force_remount=True)
```

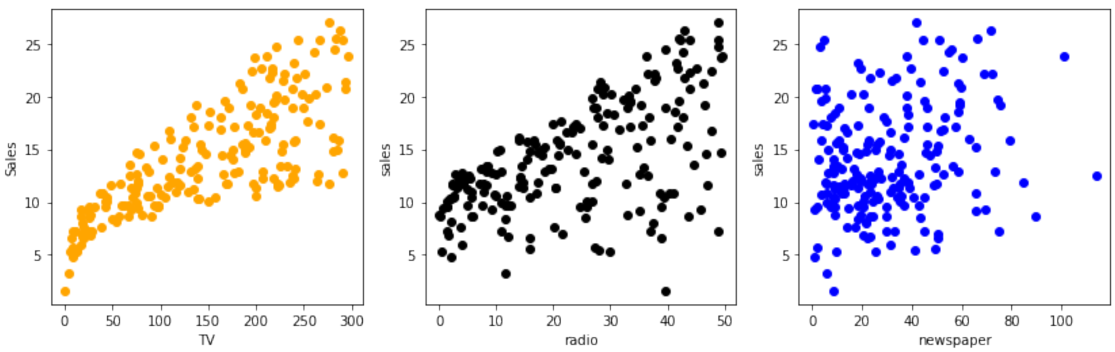
```
Mounted at /content/drive
```

[3]:
```
import pandas as pd, numpy as np
data = pd.read_csv('/content/drive/MyDrive/Programacion/Practica_regresion/
↪Advertising.csv')

print(data)# comentario
```

```
     Unnamed: 0     TV  Radio  Newspaper  Sales
0             1  230.1   37.8       69.2   22.1
1             2   44.5   39.3       45.1   10.4
2             3   17.2   45.9       69.3    9.3
3             4  151.5   41.3       58.5   18.5
4             5  180.8   10.8       58.4   12.9
..          ...    ...    ...        ...    ...
195         196   38.2    3.7       13.8    7.6
196         197   94.2    4.9        8.1    9.7
197         198  177.0    9.3        6.4   12.8
198         199  283.6   42.0       66.2   25.5
199         200  232.1    8.6        8.7   13.4

[200 rows x 5 columns]
```

[4]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(14,4))

plt.subplot(1,3,1)
plt.scatter(data['TV'], data['Sales'], color='orange')
plt.xlabel('TV')
plt.ylabel('Sales')

plt.subplot(1,3,2)
```

```
plt.scatter(data['Radio'], data['Sales'], color = 'black')
plt.xlabel('radio')
plt.ylabel('sales')

plt.subplot(1,3,3)
plt.scatter(data['Newspaper'], data['Sales'], color = 'blue')
plt.xlabel('newspaper')
plt.ylabel('sales')


plt.show()
```



La regresión lineal encuentra los parámetros de la línea que mejor se ajusta a los datos, es decir, la pendiente (θ_1) y la constante o intercepto (θ_0) en este caso.

$$h_\theta(x) = \theta_1 x + \theta_0$$

La covarianza entre dos variables aleatorias X e Y se define como:

$$Cov(X,Y) = E[(X - E[X])(Y - E[Y])$$

Dado que las variables aleatorias X e Y son discretas y están definidas como $X = \{x_1, x_2, ..., x_n\}$ y $Y = \{y_1, y_2, ..., y_n\}$ con igual probabilidad para todos sus elementos entonces la covarianza se puede definir como:

$$Cov(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - E[X])(y_i - E[Y])$$

[5]:
```
col_namesx=['TV','Radio','Newspaper']

def covarianza(a,b):
  sum=0
  for i in range(len(a)):
    sum=sum+ (a[i]-a.mean())*(b[i]-b.mean())
```

```
  res=sum/len(a)
  return res

print(covarianza(data['TV'],data['Sales']))
print(covarianza(data['Radio'],data['Sales']))
print(covarianza(data['Newspaper'],data['Sales']))
print('Función------------------')
from numpy import cov
for col in col_namesx:
  print(np.cov(data[col],data['Sales'],bias=True)[0,1])
```

```
348.6382437499999
44.412509999999976
25.811684999999997
Función------------------
348.63824374999996
44.412509999999976
25.811684999999997
```

Todos con todos

[6]:
```
col_namesxy=['TV','Radio','Newspaper','Sales']
cov_todos=np.array([['    ']+col_namesxy])
#print(cov_todos)
#f=pd.DataFrame(m,columns=[col_namesx])
for col in col_namesxy:
  #fila=[]
  fila=[col]
  for colm in col_namesxy:
    fila.append(cov(data[col],data[colm],bias=False)[0,1])
    #print(fila)
  cov_todos=np.append(cov_todos,[fila],axis=0)

r=pd.DataFrame(cov_todos)
print(r)
```

```
           0                  1                  2                  3  \
0                            TV              Radio          Newspaper
1         TV   7370.949893216087   69.8624924623115  105.91945226130643
2      Radio    69.8624924623115 220.42774271356785  114.49697889447243
3  Newspaper  105.91945226130643 114.49697889447243  474.30832562814095
4      Sales  350.39019472361804  44.635688442211034   25.94139195979899

                    4
0               Sales
1  350.39019472361804
2  44.635688442211034
3   25.94139195979899
```

```
4    27.22185301507536
```

1 Pearson correlación

Dado un par de variables aleatorias (X, Y), el coeficiente de correlación poblacional de Pearson (también denotado por $\rho_{X,Y}$) se define como

$$\rho_{X,Y} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y} = \frac{\operatorname{Cov}(X,Y)}{\sqrt{\operatorname{Var}(X)\operatorname{Var}(Y)}}$$

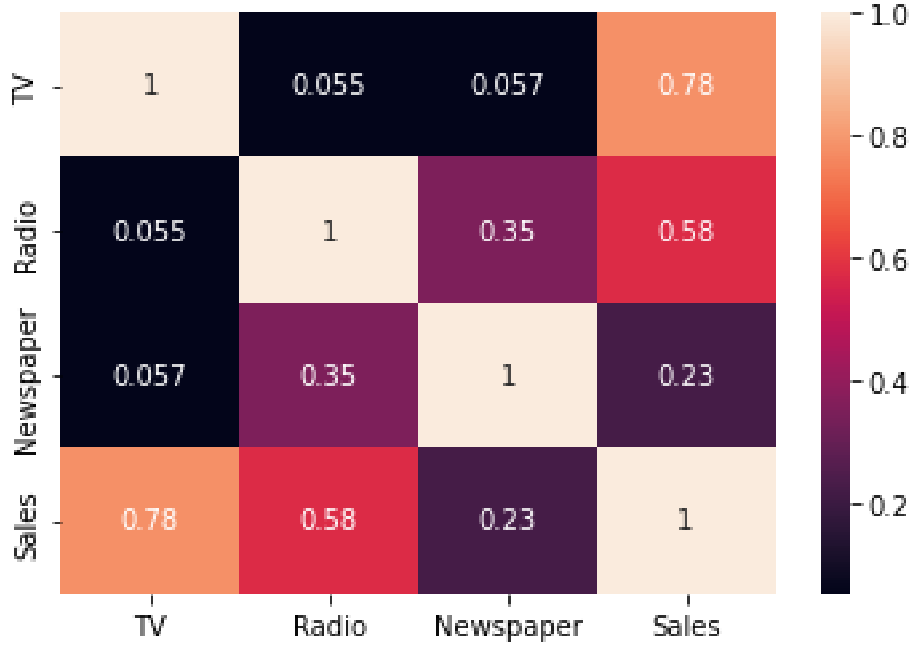
donde

- σ_{XY} es la covarianza de (X, Y)
- σ_X es la desviación estándar de la variable X
- σ_Y es la desviación estándar de la variable Y

[7]:
```
import seaborn as sns
def pearson(x,y):
  p=(cov(x,y)[0,1])/((np.std(x))*(np.std(y)))
  return p

for c in col_namesx:
  print(c,pearson(data[c],data['Sales']))

sns.heatmap(data[col_namesxy].corr(),annot=True)
```

```
TV 0.786155200865936
Radio 0.5791181653980452
Newspaper 0.22944625766448773
```

[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f40155cfb10>
```



2 Error cuadrático medio (MSE) y Error Absoluto Medio (MAE)

En nuestro caso función de costo es una función que calcula el error entre los valores de ventas (objetivo) predichos por nuestra hipótesis h y la publicidad realizada en TV, radio o periódicos en el conjunto de entrenamiento.

- Error Cuadrático Medio (MSE, por sus sigles en inglés)

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

- Error Absoluto Medio (MAE, por sus sigles en inglés)

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} |h_\theta(x_i) - y_i|$$

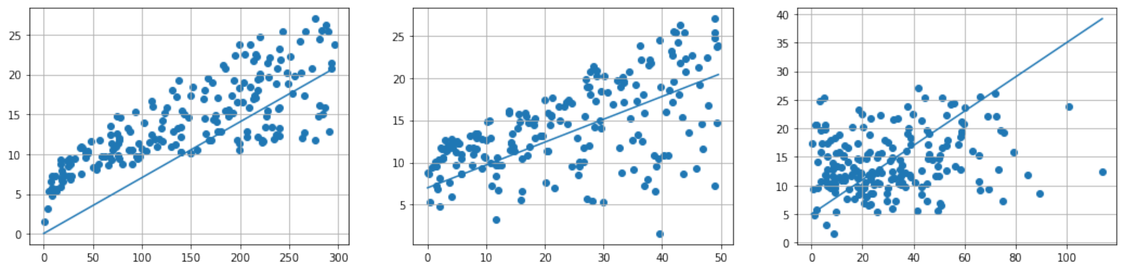
Aquí h es el valor predicho por nuestro modelos m es el número de muestras.

[8]:
```
t1=0.05
t2=0.07
def hipotesis(x,t0,t1):
  return t1*x +t0
def MSE(x,y,t1,t2): #Error Cuadratico Medio
  j0=0
  for i in range(len(x)):
```

```
    j0=j0+((hipotesis(x[i],t1,t2))-y[i])**2
  j0=j0*(1/(2*len(x)))
  return j0

def MAE(x,y,t1,t2): #error absoluto medio
  j0=0
  for i in range(len(x)):
    j0=j0+abs((hipotesis(x[i],t1,t2))-y[i])
  j0=j0*(1/(2*len(x)))
  return j0
#costo cal_costo



def plotpl(X,Y,t1,t0):
  x=np.linspace(0,X.max(),len(X))
  plt.scatter(X,Y)
  plt.plot(x,hipotesis(x,t1,t0))
  plt.grid()

plt.figure(figsize=(18,4))
plt.subplot(1,3,1)
plotpl(data['TV'],data['Sales'],t1,t2)
print('TV vs Sales')
print('MSE= ',MSE(data['TV'],data['Sales'],t1,t2))
print('MAE= ',MAE(data['TV'],data['Sales'],t1,t2))
plt.subplot(1,3,2)
t0r=7
t1r=0.27
print('Radio vs Sales')
print('MSE= ',MSE(data['Radio'],data['Sales'],t0r,t1r))
print('MAE= ',MAE(data['Radio'],data['Sales'],t0r,t1r))
plotpl(data['Radio'],data['Sales'],t0r,t1r)
plt.subplot(1,3,3)
t0n=5
t1n=0.3
print('Newspaper vs Sales')
print('MSE= ',MSE(data['Newspaper'],data['Sales'],t0n,t1n))
print('MAE= ',MAE(data['Newspaper'],data['Sales'],t0n,t1n))
plotpl(data['Newspaper'],data['Sales'],t0n,t1n)
```

```
TV vs Sales
MSE=  13.876180027499993
MAE=  2.3473775000000003
Radio vs Sales
MSE=  9.820615985
MAE=  1.8139000000000005
Newspaper vs Sales
```

```
MSE=  27.04684649999998
MAE=  2.8329500000000003
```



3 Mínimos cuadrados

[9]:
```
def min_cuad(x,y):
  n=len(x)
  x_sum=sum(x)
  y_sum=sum(y)
  xy_sum=sum(x.mul(y))
  x2_sum=sum(x.mul(x))
  a1=((n*xy_sum)-(x_sum*y_sum))/((n*x2_sum)-(x_sum)**2)
  a0=y.mean()-(a1*x.mean())
  return [a0,a1]
```

Minimos cuadrados advertising

[10]:
```
r_adv=min_cuad(data['TV'],data['Sales'])
print(r_adv)
print(MSE(data['TV'],data['Sales'],r_adv[0],r_adv[1]))
print(MAE(data['TV'],data['Sales'],r_adv[0],r_adv[1]))

r_adv=min_cuad(data['Radio'],data['Sales'])
print(r_adv)
print(MSE(data['Radio'],data['Sales'],r_adv[0],r_adv[1]))
print(MAE(data['Radio'],data['Sales'],r_adv[0],r_adv[1]))

r_adv=min_cuad(data['Newspaper'],data['Sales'])
print(r_adv)
print(MSE(data['Newspaper'],data['Sales'],r_adv[0],r_adv[1]))
print(MAE(data['Newspaper'],data['Sales'],r_adv[0],r_adv[1]))
```

```
[7.032593549127709, 0.04753664043301965]
5.2563264578283775
1.274903019463743
[9.31163809515829, 0.20249578339243937]
9.046198872562716
```

```
1.6601093988210331
[12.35140706927817, 0.054693098472273056]
12.837011360279845
2.0732798719192447
```

repetir hasta convergencia o se es alcanzado un número de épocas determinado {

$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$ (para todos los j)

}

Donde α es la tasa de aprendizaje

4 Gradiente desendiente

5 Uso de Gradiente Desendiente TV vs Sales

La intencion es minimizar la función de coste para poder obtener un mejor ajuste

MSE algoritmo de gradiente desendiente

$$\theta_0 := \theta_0 - \alpha \frac{1}{2m} \frac{\partial}{\partial \theta_0} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$
$$\theta_1 := \theta_1 - \alpha \frac{1}{2m} \frac{\partial}{\partial \theta_1} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

Tomando en cuenta que:

$$h_\theta(x) = \theta_1 x + \theta_0$$

entonces resulta:

$$\theta_0 := \theta_0 - \frac{\alpha}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)$$

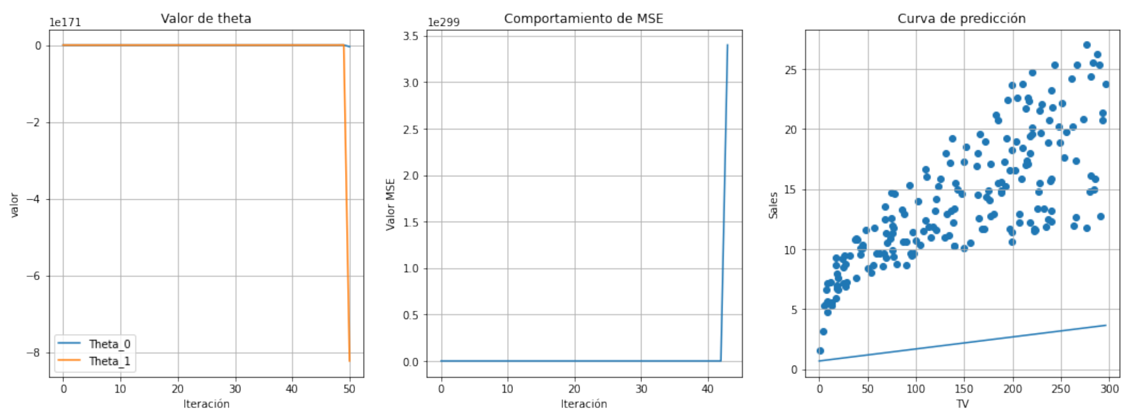
$$\theta_1 := \theta_1 - \frac{\alpha}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i) x_i$$

[11]:
```
#Uso de gradiente descendiente  COLOCAR TITULOS Y EJES
import random
alpha=0.1
theta1=0
t0=0.7
t1=0.01
def gradiente_mse(alfa,x,y,iteraciones,t0,t1):

  t0_list=[]
```

```
  t1_list=[]
  t0_list.append(t0)
  t1_list.append(t1)
  it=[]
  it.append(0)
  error_c=[]
  gd0=0
  gd1=0
  m=len(x)
  error_c.append(MSE(x,y,t0,t1))
  for i in range(1,iteraciones+1):
    for j in range(len(x)):
      gd1=gd1+((hipotesis(x[j],t0,t1)-y[j])*x[j])
      gd0=gd0+(hipotesis(x[j],t0,t1)-y[j])


    it.append(i)
    t0=t0-((alpha/m)*gd0)
    t1=t1-((alpha/m)*gd1)
    t0_list.append(t0)
    t1_list.append(t1)
    error_c.append(MSE(x,y,t0,t1))
  return [it,t0_list,t1_list,error_c]

resulgd=gradiente_mse(alpha,data['TV'],data['Sales'],50,t0,t1)
#print()
itr=resulgd[0]
t0_r=resulgd[1]
t1_r=resulgd[2]
mse_r=resulgd[3]

plt.figure(figsize=(18,6))

plt.subplot(1,3,1)
plt.plot(resulgd[0],resulgd[1])
plt.plot(resulgd[0],resulgd[2])
plt.xlabel('Iteración')
plt.ylabel('valor')
plt.title('Valor de theta')
plt.legend(['Theta_0','Theta_1'])
plt.grid()
plt.subplot(1,3,2)

plt.plot(resulgd[0],resulgd[3])
plt.xlabel('Iteración')
plt.ylabel('Valor MSE')
plt.title('Comportamiento de MSE')
```

```
plt.grid()

plt.subplot(1,3,3)
plotpl(data['TV'],data['Sales'],t0_r[mse_r.index(min(mse_r))],t1_r[mse_r.
 ↪index(min(mse_r))])
plt.xlabel('TV')
plt.ylabel('Sales')
plt.title('Curva de predicción')


print('El minimo encontrado fue un MSE de: ',min(mse_r))
print('Con un t0= ',t0_r[mse_r.index(min(mse_r))])
print('Con un t1= ',t1_r[mse_r.index(min(mse_r))])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: RuntimeWarning:
overflow encountered in double_scalars


El minimo encontrado fue un MSE de:  80.65903509750002
Con un t0=  0.7
Con un t1=  0.01
```



6 Uso de Gradiente descendiente TV vs Sales

MAE

$$\theta_0 := \theta_0 - \alpha \frac{1}{2m} \frac{\partial}{\partial \theta_0} \sum_{i=1}^{m} |h_\theta(x_i) - y_i|$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{2m} \frac{\partial}{\partial \theta_1} \sum_{i=1}^{m} |h_\theta(x_i) - y_i|$$

Tomando en cuenta que:

$$h_\theta(x) = \theta_1 x + \theta_0$$

entonces resulta:

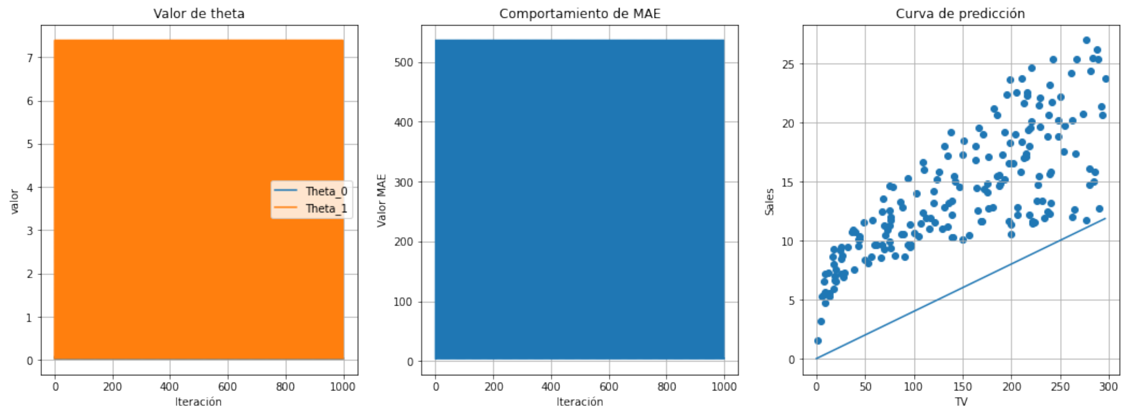
$$\theta_0 := \theta_0 - \alpha \frac{1}{2m} \sum_{i=1}^{m} \frac{h_\theta(x_i) - y_i}{|h_\theta(x_i) - y_i|}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{2m} \sum_{i=1}^{m} \frac{h_\theta(x_i) - y_i}{|h_\theta(x_i) - y_i|} x_i$$

[12]:
```python
import random
alpha=0.1
theta1=0
t0=0.034#0.05
t1=0.04#0.07
def gradiente_mae(alfa,x,y,iteraciones,t0,t1):

  t0_list=[]
  t1_list=[]
  t0_list.append(t0)
  t1_list.append(t1)
  it=[]
  it.append(0)
  error_c=[]
  gd0=0
  gd1=0
  m=len(x)
  error_c.append(MAE(x,y,t0,t1))
  for i in range(1,iteraciones+1):
    for j in range(len(x)):
      gd1=gd1+(((hipotesis(x[j],t0,t1)-y[j])/
 ↪(abs(hipotesis(x[j],t0,t1)-y[j])))*x[j])

      gd0=gd0+(((hipotesis(x[j],t0,t1)-y[j]))/(abs(hipotesis(x[j],t0,t1)-y[j])))


    it.append(i)
    t0=t0-((alpha/(2*m))*gd0)
    t1=t1-((alpha/(2*m))*gd1)
    t0_list.append(t0)
    t1_list.append(t1)
    error_c.append(MAE(x,y,t0,t1))
  return [it,t0_list,t1_list,error_c]

resulgd=gradiente_mae(alpha,data['TV'],data['Sales'],1000,t0,t1) #
```

```
#print()
itr=resulgd[0]
t0_r=resulgd[1]
t1_r=resulgd[2]
mae_r=resulgd[3]

plt.figure(figsize=(18,6))

plt.subplot(1,3,1)
plt.plot(resulgd[0],resulgd[1])
plt.plot(resulgd[0],resulgd[2])
plt.xlabel('Iteración')
plt.ylabel('valor')
plt.title('Valor de theta')
plt.legend(['Theta_0','Theta_1'])
plt.grid()
plt.subplot(1,3,2)
plt.plot(resulgd[0],resulgd[3])
plt.xlabel('Iteración')
plt.ylabel('Valor MAE')
plt.title('Comportamiento de MAE')
plt.grid()

plt.subplot(1,3,3)
plotpl(data['TV'],data['Sales'],t0_r[mae_r.index(min(mae_r))],t1_r[mae_r.
 ↪index(min(mae_r))])
plt.xlabel('TV')
plt.ylabel('Sales')
plt.title('Curva de predicción')


print('El minimo encontrado fue un MAE de: ',min(mae_r))
print('Con un t0= ',t0_r[mae_r.index(min(mae_r))])
print('Con un t1= ',t1_r[mae_r.index(min(mae_r))])
```

```
El minimo encontrado fue un MAE de:  4.053399999999997
Con un t0=  0.034
Con un t1=  0.04
```



7 Uso de Gradiente descendiente

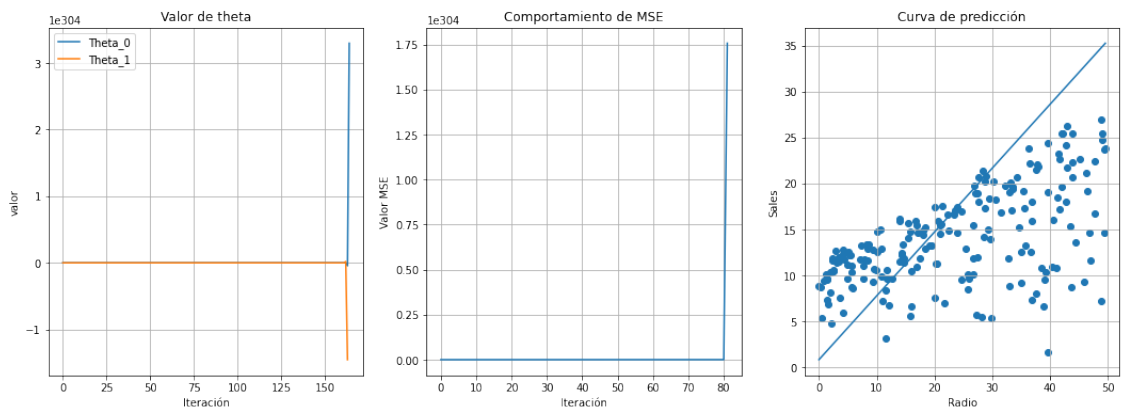
MSE Radio vs sales

[13]:
```
alpha=0.1
t0r=random.random()#7
t1r=random.random()#0.27

resulgd=gradiente_mse(alpha,data['Radio'],data['Sales'],400,t0r,t1r)
#print()
itr=resulgd[0]
t0_r=resulgd[1]
t1_r=resulgd[2]
mse_r=resulgd[3]

plt.figure(figsize=(18,6))

plt.subplot(1,3,1)
plt.plot(resulgd[0],resulgd[1])
plt.plot(resulgd[0],resulgd[2])
plt.xlabel('Iteración')
plt.ylabel('valor')
plt.title('Valor de theta')
plt.legend(['Theta_0','Theta_1'])
plt.grid()
plt.subplot(1,3,2)

plt.plot(resulgd[0],resulgd[3])
plt.xlabel('Iteración')
plt.ylabel('Valor MSE')
plt.title('Comportamiento de MSE')
plt.grid()
```

```
plt.subplot(1,3,3)
plotpl(data['Radio'],data['Sales'],t0_r[mse_r.index(min(mse_r))],t1_r[mse_r.
 ↪index(min(mse_r))])
plt.xlabel('Radio')
plt.ylabel('Sales')
plt.title('Curva de predicción')


print('El minimo encontrado fue un MSE de: ',min(mse_r))
print('Con un t0= ',t0_r[mse_r.index(min(mse_r))])
print('Con un t1= ',t1_r[mse_r.index(min(mse_r))])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: RuntimeWarning:
overflow encountered in double_scalars

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:22: RuntimeWarning:
overflow encountered in double_scalars
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: RuntimeWarning:
invalid value encountered in double_scalars
  after removing the cwd from sys.path.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:22: RuntimeWarning:
invalid value encountered in double_scalars
```

```
El minimo encontrado fue un MSE de:  39.92837007399209
Con un t0=  0.8288557834800366
Con un t1=  0.6941945115745622
```



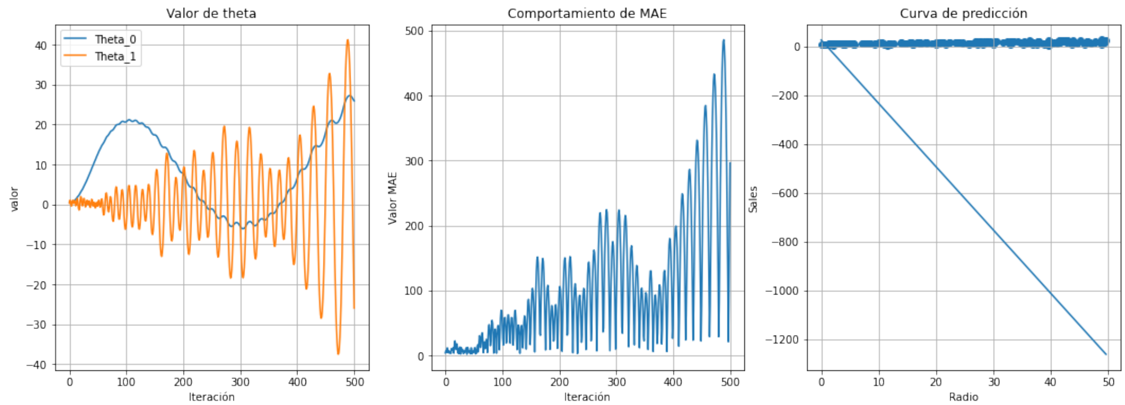
Gradiente decendiente

MAE Radio vs sales

[14]:
```
alpha=0.1
t0r=random.random()#7
```

```
t1r=random.random()#0.27

resulgd=gradiente_mae(alpha,data['Radio'],data['Sales'],500,t0r,t1r) #
#print()
itr=resulgd[0]
t0_r=resulgd[1]
t1_r=resulgd[2]
mae_r=resulgd[3]

plt.figure(figsize=(18,6))

plt.subplot(1,3,1)
plt.plot(resulgd[0],resulgd[1])
plt.plot(resulgd[0],resulgd[2])
plt.xlabel('Iteración')
plt.ylabel('valor')
plt.title('Valor de theta')
plt.legend(['Theta_0','Theta_1'])
plt.grid()
plt.subplot(1,3,2)
plt.plot(resulgd[0],resulgd[3])
plt.xlabel('Iteración')
plt.ylabel('Valor MAE')
plt.title('Comportamiento de MAE')
plt.grid()

plt.subplot(1,3,3)
plotpl(data['Radio'],data['Sales'],t0_r[-1],t1_r[-1])
plt.xlabel('Radio')
plt.ylabel('Sales')
plt.title('Curva de predicción')


print('El minimo encontrado fue un MAE de: ',min(mae_r))
print('Con un t0= ',t0_r[mae_r.index(min(mae_r))])
print('Con un t1= ',t1_r[mae_r.index(min(mae_r))])
```

```
El minimo encontrado fue un MAE de:  1.6469923431580864
Con un t0=  8.5337278187935
Con un t1=  0.2731191641043418
```



8 Uso del gradiente descendiente

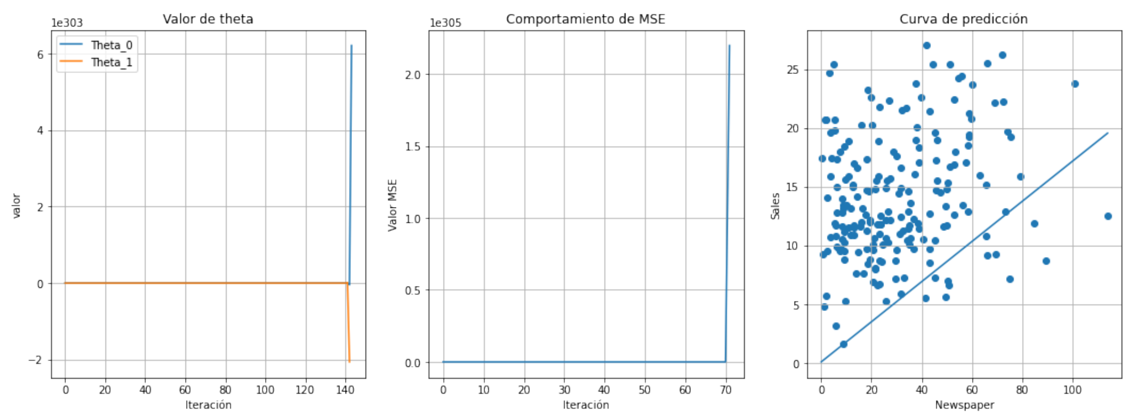
MSE Newspaper vs sales

[15]:
```
alpha=0.1
t0n=random.random()#5
t1n=random.random()#0.3

resulgd=gradiente_mse(alpha,data['Newspaper'],data['Sales'],500,t0n,t1n)
#print()
itr=resulgd[0]
t0_r=resulgd[1]
t1_r=resulgd[2]
mse_r=resulgd[3]

plt.figure(figsize=(18,6))

plt.subplot(1,3,1)
plt.plot(resulgd[0],resulgd[1])
plt.plot(resulgd[0],resulgd[2])
plt.xlabel('Iteración')
plt.ylabel('valor')
plt.title('Valor de theta')
plt.legend(['Theta_0','Theta_1'])
plt.grid()
plt.subplot(1,3,2)

plt.plot(resulgd[0],resulgd[3])
plt.xlabel('Iteración')
plt.ylabel('Valor MSE')
plt.title('Comportamiento de MSE')
plt.grid()
```

```
plt.subplot(1,3,3)
plotpl(data['Newspaper'],data['Sales'],t0_r[mse_r.index(min(mse_r))],t1_r[mse_r.
↪index(min(mse_r))])
plt.xlabel('Newspaper')
plt.ylabel('Sales')
plt.title('Curva de predicción')


print('El minimo encontrado fue un MSE de: ',min(mse_r))
print('Con un t0= ',t0_r[mse_r.index(min(mse_r))])
print('Con un t1= ',t1_r[mse_r.index(min(mse_r))])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: RuntimeWarning:
overflow encountered in double_scalars

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:22: RuntimeWarning:
overflow encountered in double_scalars
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:22: RuntimeWarning:
invalid value encountered in double_scalars
```

```
El minimo encontrado fue un MSE de:  53.93363546867626
Con un t0=  0.10114318338599904
Con un t1=  0.17057851279892167
```



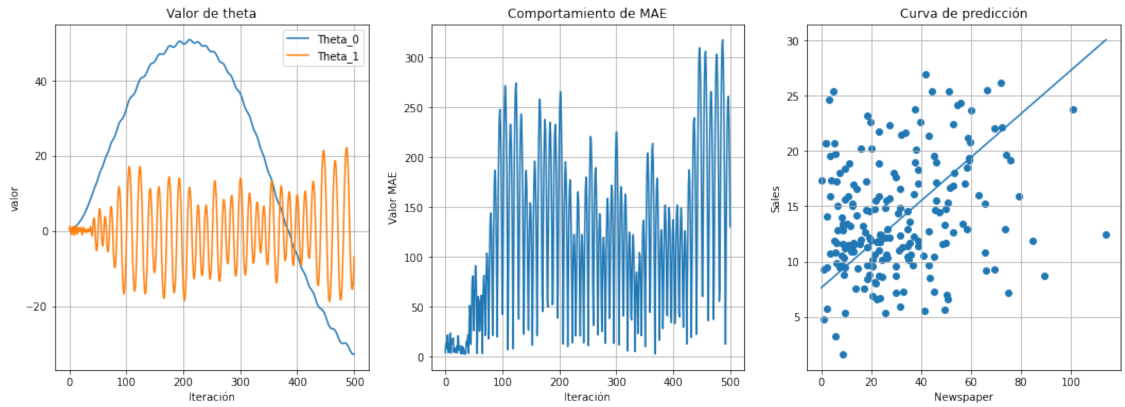
Uso de Gradiente decendiente

MAE Newspaper vs Sales

[16]:
```
alpha=0.1
t0n=random.random()#5
t1n=random.random()#0.3
```

```
resulgd=gradiente_mae(alpha,data['Newspaper'],data['Sales'],500,t0n,t1n) #
#print()
itr=resulgd[0]
t0_r=resulgd[1]
t1_r=resulgd[2]
mae_r=resulgd[3]

plt.figure(figsize=(18,6))

plt.subplot(1,3,1)
plt.plot(resulgd[0],resulgd[1])
plt.plot(resulgd[0],resulgd[2])
plt.xlabel('Iteración')
plt.ylabel('valor')
plt.title('Valor de theta')
plt.legend(['Theta_0','Theta_1'])
plt.grid()
plt.subplot(1,3,2)
plt.plot(resulgd[0],resulgd[3])
plt.xlabel('Iteración')
plt.ylabel('Valor MAE')
plt.title('Comportamiento de MAE')
plt.grid()

plt.subplot(1,3,3)
plotpl(data['Newspaper'],data['Sales'],t0_r[mae_r.index(min(mae_r))],t1_r[mae_r.
 ↪index(min(mae_r))])
plt.xlabel('Newspaper')
plt.ylabel('Sales')
plt.title('Curva de predicción')


print('El minimo encontrado fue un MAE de: ',min(mae_r))
print('Con un t0= ',t0_r[mae_r.index(min(mae_r))])
print('Con un t1= ',t1_r[mae_r.index(min(mae_r))])
```

```
El minimo encontrado fue un MAE de:  2.308489014157994
Con un t0=  7.621018093855884
Con un t1=  0.19705008321323914
```



9 Base de datos articulos

Analisis estadistico

[17]:
```
art=pd.read_csv('/content/drive/MyDrive/Programacion/Practica_regresion/
 ↪articulos_ml.csv')
print(art)
art['Word count'] = art['Word count'].fillna(0) #se llena con 0 los nan values
art['# of Links'] = art['# of Links'].fillna(0)
art['# of comments'] = art['# of comments'].fillna(0)
art['# Images video'] = art['# Images video'].fillna(0)
art['Elapsed days'] = art['Elapsed days'].fillna(0)
art['# Shares'] = art['# Shares'].fillna(0)
```

```
                                                Title  \
0    What is Machine Learning and how do we use it ...
1     10 Companies Using Machine Learning in Cool Ways
2    How Artificial Intelligence Is Revolutionizing...
3    Dbrain and the Blockchain of Artificial Intell...
4    Nasa finds entire solar system filled with eig...
..                                                ...
156  [Log] 83: How Google Uses Machine Learning And...
157  [Log] 84: Zuck Knows If You've Been Bad Or Goo...
158  [Log] 85: Microsoft Improves Windows Phone Voi...
159  [Log] 86: How Google's Acquisition Of DNNresea...
160   [Log] 87: Google's Cloud Is Eating Apple's Lunch

                                                  url  Word count  \
0    https://blog.signals.network/what-is-machine-l...        1888
1                                                 NaN        1742
2                                                 NaN         962
3                                                 NaN        1221
4                                                 NaN        2039
```

```
..                                                    ...        ...
156  [Log] 83: http://feedproxy.google.com/~r/Techc...        3239
157  [Log] 84: http://feedproxy.google.com/~r/Techc...        2566
158  [Log] 85: http://feedproxy.google.com/~r/Techc...        2089
159  [Log] 86: http://feedproxy.google.com/~r/Techc...        1530
160  [Log] 87: http://feedproxy.google.com/~r/Techc...         953

     # of Links  # of comments  # Images video  Elapsed days  # Shares
0             1            2.0               2            34    200000
1             9            NaN               9             5     25000
2             6            0.0               1            10     42000
3             3            NaN               2            68    200000
4             1          104.0               4           131    200000
..          ...            ...             ...           ...       ...
156           3           11.0               1            84      3239
157           3            8.0               4            85     25019
158           4            4.0               1            86     49614
159           4           12.0               3            87     33660
160           6           13.0               2            88      5956

[161 rows x 8 columns]
```

Covarianza valores vs shares

[18]:
```
print('Word count vs # Shares: ',covarianza(art['Word count'],art['# Shares']))
print('# of Links vs # Shares:',covarianza(art['# of Links'],art['# Shares']))
print('# of comments vs # Shares:',covarianza(art['# of comments'],art['#␣
 ↪Shares']))
print('# Images video vs # Shares:',covarianza(art['# Images video'],art['#␣
 ↪Shares']))
print('Elapsed days vs # Shares:',covarianza(art['Elapsed days'],art['#␣
 ↪Shares']))
```

```
Word count vs # Shares:  6981978.853362141
# of Links vs # Shares: 587958.9975695381
# of comments vs # Shares: 142784.83796921413
# Images video vs # Shares: 9167.630029705639
Elapsed days vs # Shares: 1604699.6151768835
```

Covarianza entre todos

[19]:
```
col_namesxy=['Word count','# of Links','# of comments','# Images video','Elapsed␣
 ↪days','# Shares']
col_namesx=['Word count','# of Links','# of comments','# Images video','Elapsed␣
 ↪days']
cov_todos=np.array([['    ']+col_namesxy])
#print(cov_todos)
#f=pd.DataFrame(m,columns=[col_namesx])
for col in col_namesxy:
```

```
  #fila=[]
  fila=[col]
  for colm in col_namesxy:
    fila.append(cov(art[col],art[colm],bias=False)[0,1])
    #print(fila)
  cov_todos=np.append(cov_todos,[fila],axis=0)

r=pd.DataFrame(cov_todos)
print(r)
```

```
                 0                    1                   2  \
0                            Word count          # of Links
1       Word count     1303979.88152174  18716.643478260885
2       # of Links   18716.643478260885   2234.606521739144
3    # of comments   1533.8777173913045   67.42853260869555
4   # Images video   1810.2739130434788   3.407336956521741
5     Elapsed days  -21851.301358695655  319.73885869565225
6         # Shares    7025616.221195655   591633.7413043476

                    3                   4                    5  \
0       # of comments      # Images video         Elapsed days
1  1533.8777173913045  1810.2739130434788  -21851.301358695655
2   67.42853260869555   3.407336956521741   319.73885869565225
3  150.54860248447213  -1.375155279503107    52.76409161490682
4  -1.375155279503107  11.684704968944084    82.05989906832292
5   52.76409161490682   82.05989906832292   13073.071972049693
6   143677.2432065217   9224.927717391301   1614728.9877717388

                    6
0            # Shares
1   7025616.221195655
2   591633.7413043476
3   143677.2432065217
4   9224.927717391301
5  1614728.9877717388
6  1884255057.7032604
```

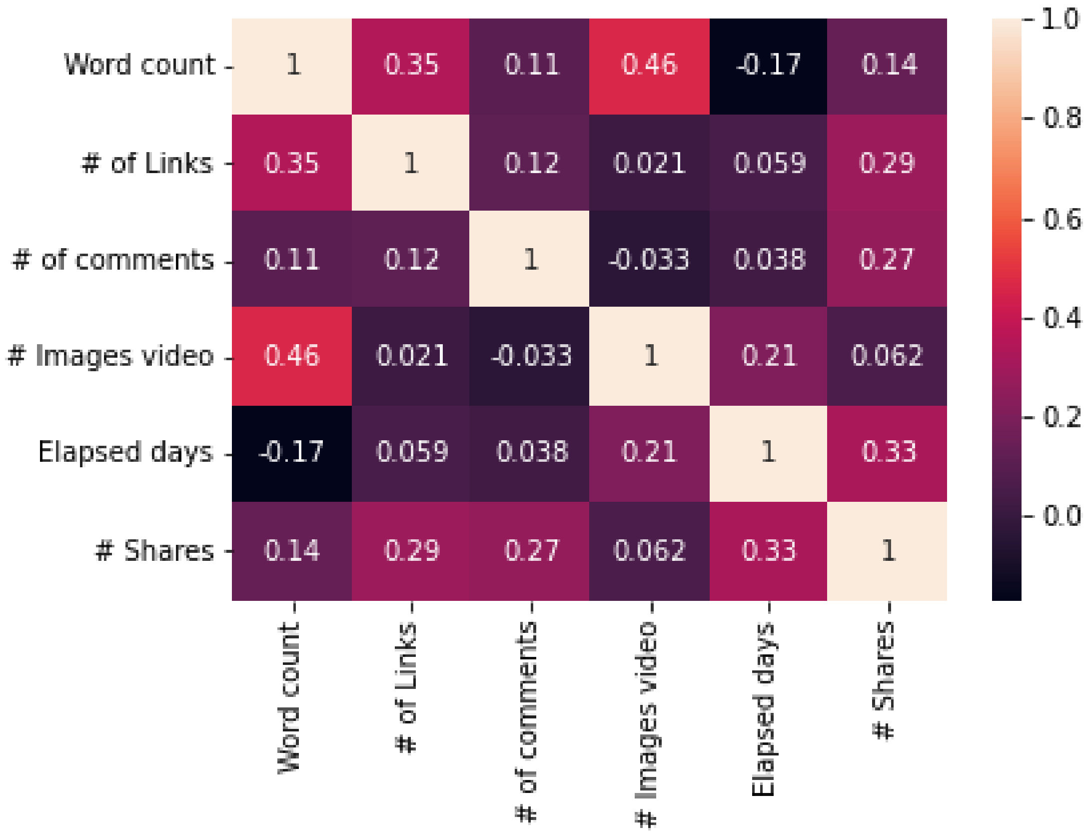
Correlación de Pearson

[20]:
```
import seaborn as sns

def pearson(x,y):
  p=(cov(x,y)[0,1])/((np.std(x))*(np.std(y)))
  return p

for c in col_namesx:
  print(c,pearson(art[c],art['# Shares']))
```

```
sns.heatmap(art[col_namesxy].corr(),annot=True) #mapa de calor de relación de␣
↪los valores todos con todos
```

```
Word count 0.14262150847304
# of Links 0.29012720337832265
# of comments 0.27144739034075294
# Images video 0.0625590499668279
Elapsed days 0.3273760198137792
```

[20]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4012489150>



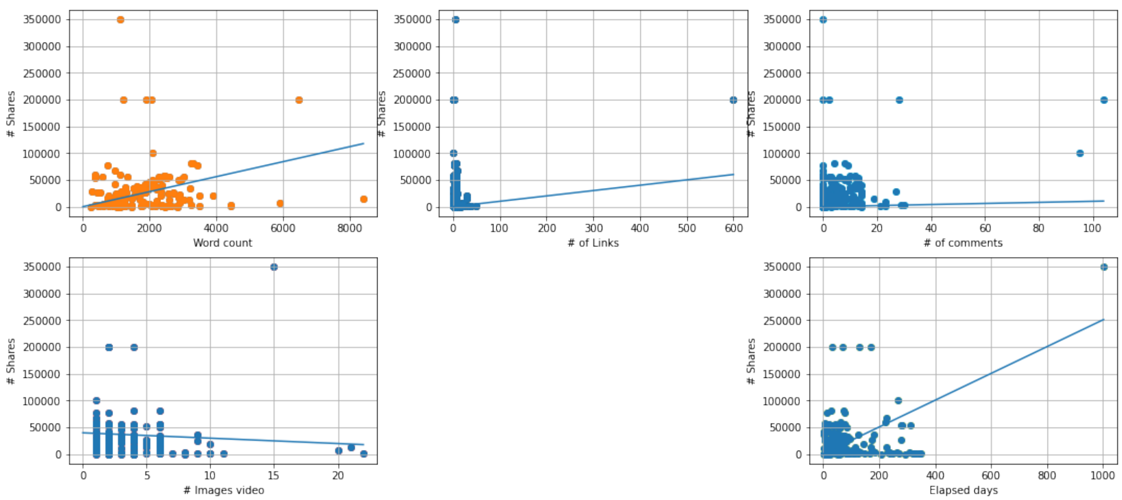
Gráficas

[21]:
```
plt.figure(figsize=(18,8))
plt.subplot(2,3,1)
plt.scatter(art['Word count'],art['# Shares'])
#plt.grid()
t0wc=4
t1wc=14
plotpl(art['Word count'],art['# Shares'],t0wc,t1wc)
plt.xlabel('Word count')
```

```
plt.ylabel('# Shares')
print('Word count vs # Shares')
print('MSE= ',MSE(art['Word count'],art['# Shares'],t0wc,t1wc))
print('MAE= ',MAE(art['Word count'],art['# Shares'],t0wc,t1wc))


plt.subplot(2,3,2)
plt.scatter(art['# of Links'],art['# Shares'],color='black')
plt.xlabel('# of Links')
plt.ylabel('# Shares')
t0of=15
t1of=100
plotpl(art['# of Links'],art['# Shares'],t0of,t1of)
print('# of Links vs # Shares')
print('MSE= ',MSE(art['# of Links'],art['# Shares'],t0of,t1of))
print('MAE= ',MAE(art['# of Links'],art['# Shares'],t0of,t1of))

plt.subplot(2,3,3)
plt.scatter(art['# of comments'],art['# Shares'],color='cyan')
plt.xlabel('# of comments')
plt.ylabel('# Shares')
t0oc=10
t1oc=100
plotpl(art['# of comments'],art['# Shares'],t0oc,t1oc)
print('# of comments vs # Shares')
print('MSE= ',MSE(art['# of comments'],art['# Shares'],t0oc,t1oc))
print('MAE= ',MAE(art['# of comments'],art['# Shares'],t0oc,t1oc))

plt.subplot(2,3,4)
plt.scatter(art['# Images video'],art['# Shares'],color='red')
plt.xlabel('# Images video')
plt.ylabel('# Shares')
t0iv=40000
t1iv=-1000
plotpl(art['# Images video'],art['# Shares'],t0iv,t1iv)
print('# Images video vs # Shares')
print('MSE= ',MSE(art['# Images video'],art['# Shares'],t0iv,t1iv))
print('MAE= ',MAE(art['# Images video'],art['# Shares'],t0iv,t1iv))

plt.subplot(2,3,6)
plt.scatter(art['Elapsed days'],art['# Shares'],color='yellow')
plt.xlabel('Elapsed days')
plt.ylabel('# Shares')
t0ed=800
t1ed=250
plotpl(art['Elapsed days'],art['# Shares'],t0ed,t1ed)
print('Elapsed days vs # Shares')
```

```
print('MSE= ',MSE(art['Elapsed days'],art['# Shares'],t0ed,t1ed))
print('MAE= ',MAE(art['Elapsed days'],art['# Shares'],t0ed,t1ed))
```

```
Word count vs # Shares
MSE=  968979423.0434783
MAE=  11961.695652173912
# of Links vs # Shares
MSE=  1251989102.1024845
MAE=  13568.499999999998
# of comments vs # Shares
MSE=  1293607682.5838509
MAE=  13665.217391304346
# Images video vs # Shares
MSE=  986368778.795031
MAE=  14184.86956521739
Elapsed days vs # Shares
MSE=  944522033.7639751
MAE=  14627.658385093167
```



Uso de minimos cuadrados con Articulos

[22]:
```
t_r_wc=min_cuad(art['Word count'],art['# Shares'])
print(t_r_wc)
print(MSE(art['Word count'],art['# Shares'],t_r_wc[0],t_r_wc[1]))
print(MAE(art['Word count'],art['# Shares'],t_r_wc[0],t_r_wc[1]))

t_r_wc=min_cuad(art['# of Links'],art['# Shares'])
print(t_r_wc)
print(MSE(art['# of Links'],art['# Shares'],t_r_wc[0],t_r_wc[1]))
print(MAE(art['Word count'],art['# Shares'],t_r_wc[0],t_r_wc[1]))
```

```
t_r_wc=min_cuad(art['# of comments'],art['# Shares'])
print(t_r_wc)
print(MSE(art['# of comments'],art['# Shares'],t_r_wc[0],t_r_wc[1]))
print(MAE(art['Word count'],art['# Shares'],t_r_wc[0],t_r_wc[1]))

t_r_wc=min_cuad(art['# Images video'],art['# Shares'])
print(t_r_wc)
print(MSE(art['# Images video'],art['# Shares'],t_r_wc[0],t_r_wc[1]))
print(MAE(art['Word count'],art['# Shares'],t_r_wc[0],t_r_wc[1]))

t_r_wc=min_cuad(art['Elapsed days'],art['# Shares'])
print(t_r_wc)
print(MSE(art['Elapsed days'],art['# Shares'],t_r_wc[0],t_r_wc[1]))
print(MAE(art['Word count'],art['# Shares'],t_r_wc[0],t_r_wc[1]))
```

```
[18205.753574904906, 5.387825625803971]
917466962.7774607
12294.816571093977
[25369.818621895305, 264.75969507324993]
858441882.6139004
238297.097446539
[21232.27662261109, 954.3578673959524]
868141888.3395554
859505.9579851676
[25050.291477019782, 789.4874318101781]
932656940.7259738
712350.5868034082
[15828.470524693535, 123.51565043197498]
837173046.711687
107053.98933115955
```

Uso de gradiente decendiente

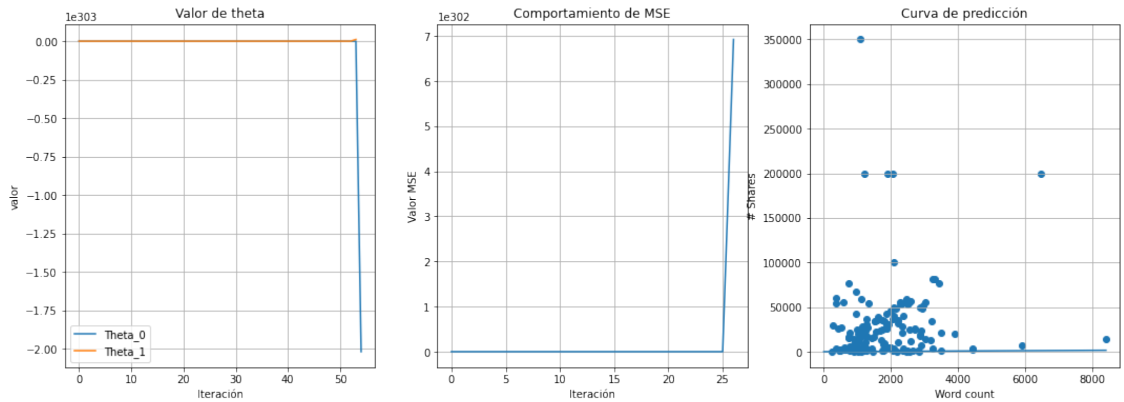
MSE Word count vs shares

[23]:
```
alpha=0.1
t0wc=random.random()#5
t1wc=random.random()#0.3

resulgd=gradiente_mse(alpha,art['Word count'],art['# Shares'],1000,t0wc,t1wc)
#print()
itr=resulgd[0]
t0_r=resulgd[1]
t1_r=resulgd[2]
mse_r=resulgd[3]

plt.figure(figsize=(18,6))
```

```
plt.subplot(1,3,1)
plt.plot(resulgd[0],resulgd[1])
plt.plot(resulgd[0],resulgd[2])
plt.xlabel('Iteración')
plt.ylabel('valor')
plt.title('Valor de theta')
plt.legend(['Theta_0','Theta_1'])
plt.grid()
plt.subplot(1,3,2)

plt.plot(resulgd[0],resulgd[3])
plt.xlabel('Iteración')
plt.ylabel('Valor MSE')
plt.title('Comportamiento de MSE')
plt.grid()

plt.subplot(1,3,3)
plotpl(art['Word count'],art['# Shares'],t0_r[mse_r.
 ↪index(min(mse_r))],t1_r[mse_r.index(min(mse_r))])
plt.xlabel('Word count')
plt.ylabel('# Shares')
plt.title('Curva de predicción')


print('El minimo encontrado fue un MSE de: ',min(mse_r))
print('Con un t0= ',t0_r[mse_r.index(min(mse_r))])
print('Con un t1= ',t1_r[mse_r.index(min(mse_r))])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: RuntimeWarning:
overflow encountered in double_scalars

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:22: RuntimeWarning:
overflow encountered in double_scalars
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:22: RuntimeWarning:
invalid value encountered in double_scalars
```

```
El minimo encontrado fue un MSE de:  1316149449.9015152
Con un t0=  0.5473666444466252
Con un t1=  0.18682222422830408
```



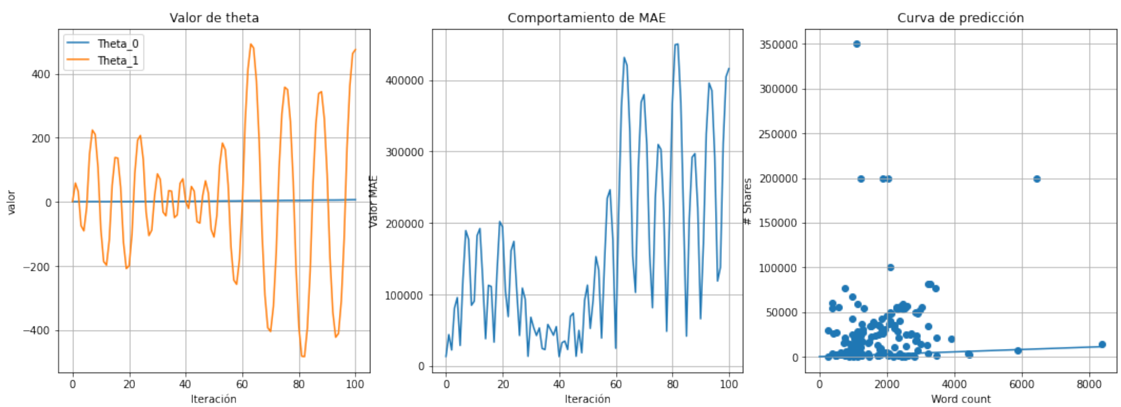
Uso de gradiente decendiente

MAE word count vs shares

[24]:
```
alpha=0.1
t0wc=random.random()
t1wc=random.random()


resulgd=gradiente_mae(alpha,art['Word count'],art['# Shares'],100,t0wc,t1wc) #
#print()
itr=resulgd[0]
t0_r=resulgd[1]
t1_r=resulgd[2]
mae_r=resulgd[3]

plt.figure(figsize=(18,6))

plt.subplot(1,3,1)
plt.plot(resulgd[0],resulgd[1])
plt.plot(resulgd[0],resulgd[2])
plt.xlabel('Iteración')
plt.ylabel('valor')
plt.title('Valor de theta')
plt.legend(['Theta_0','Theta_1'])
plt.grid()
plt.subplot(1,3,2)
plt.plot(resulgd[0],resulgd[3])
plt.xlabel('Iteración')
plt.ylabel('Valor MAE')
plt.title('Comportamiento de MAE')
plt.grid()
```

```
plt.subplot(1,3,3)
plotpl(art['Word count'],art['# Shares'],t0_r[mae_r.
 ↪index(min(mae_r))],t1_r[mae_r.index(min(mae_r))])
plt.xlabel('Word count')
plt.ylabel('# Shares')
plt.title('Curva de predicción')


print('El minimo encontrado fue un MAE de: ',min(mae_r))
print('Con un t0= ',t0_r[mae_r.index(min(mae_r))])
print('Con un t1= ',t1_r[mae_r.index(min(mae_r))])
```

```
El minimo encontrado fue un MAE de:  13090.555264355022
Con un t0=  1.845752579698811
Con un t1=  1.3226612259235822
```



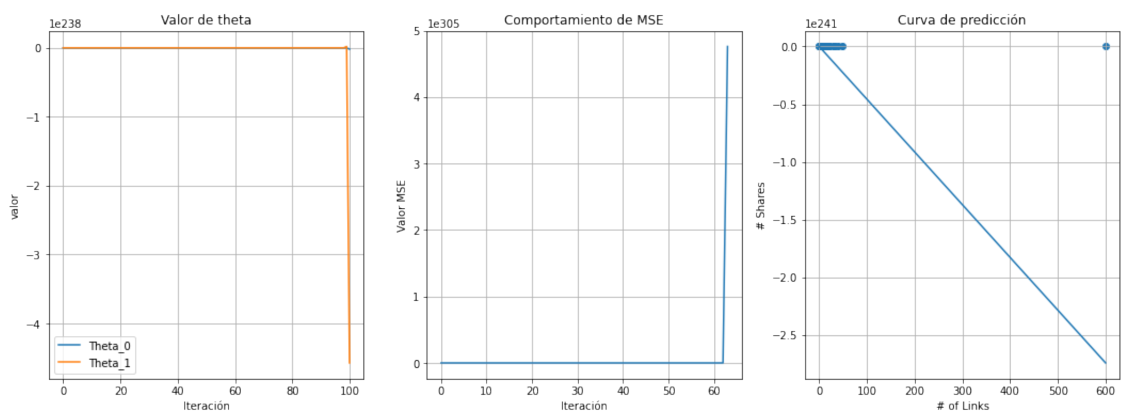
Uso de Gradiente decendiente

MSE # of Links vs # Shares

[25]:
```
alpha=0.1
t0ol=random.random()#5
t1ol=random.random()#0.3

resulgd=gradiente_mse(alpha,art['# of Links'],art['# Shares'],100,t0ol,t1ol)
#print()
itr=resulgd[0]
t0_r=resulgd[1]
t1_r=resulgd[2]
mse_r=resulgd[3]

plt.figure(figsize=(18,6))

plt.subplot(1,3,1)
```

```
plt.plot(resulgd[0],resulgd[1])
plt.plot(resulgd[0],resulgd[2])
plt.xlabel('Iteración')
plt.ylabel('valor')
plt.title('Valor de theta')
plt.legend(['Theta_0','Theta_1'])
plt.grid()
plt.subplot(1,3,2)

plt.plot(resulgd[0],resulgd[3])
plt.xlabel('Iteración')
plt.ylabel('Valor MSE')
plt.title('Comportamiento de MSE')
plt.grid()

plt.subplot(1,3,3)
plotpl(art['# of Links'],art['# Shares'],t0_r[-1],t1_r[-1])
plt.xlabel('# of Links')
plt.ylabel('# Shares')
plt.title('Curva de predicción')


print('El minimo encontrado fue un MSE de: ',min(mse_r))
print('Con un t0= ',t0_r[mse_r.index(min(mse_r))])
print('Con un t1= ',t1_r[mse_r.index(min(mse_r))])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: RuntimeWarning:
overflow encountered in double_scalars


El minimo encontrado fue un MSE de:  1326138233.405965
Con un t0=  0.36822966603055773
Con un t1=  0.7941465809752191
```

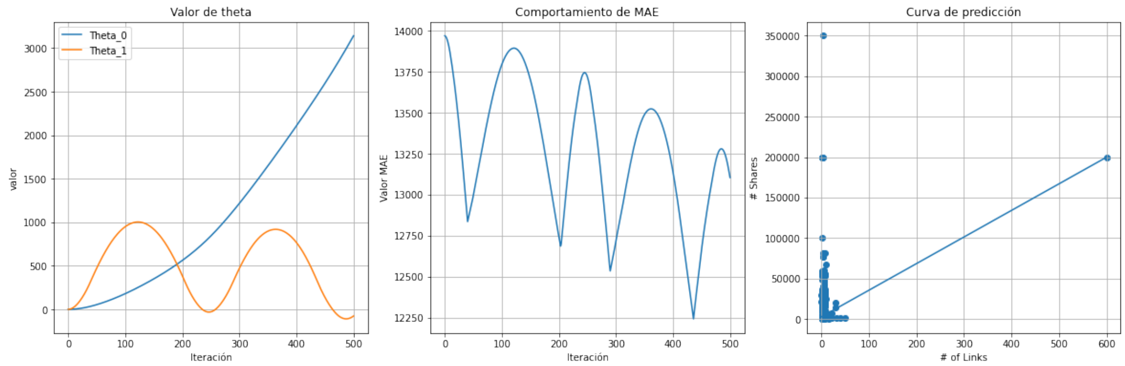


Uso de Gradiente decendiente

MAE # of Links vs # Shares

[26]:
```
alpha=0.1
t0ol=random.random()#5
t1ol=random.random()#0.3

resulgd=gradiente_mae(alpha,art['# of Links'],art['# Shares'],500,t0ol,t1ol) #
#print()
itr=resulgd[0]
t0_r=resulgd[1]
t1_r=resulgd[2]
mae_r=resulgd[3]

plt.figure(figsize=(20,6))

plt.subplot(1,3,1)
plt.plot(resulgd[0],resulgd[1])
plt.plot(resulgd[0],resulgd[2])
plt.xlabel('Iteración')
plt.ylabel('valor')
plt.title('Valor de theta')
plt.legend(['Theta_0','Theta_1'])
plt.grid()
plt.subplot(1,3,2)
plt.plot(resulgd[0],resulgd[3])
plt.xlabel('Iteración')
plt.ylabel('Valor MAE')
plt.title('Comportamiento de MAE')
plt.grid()

plt.subplot(1,3,3)
plotpl(art['# of Links'],art['# Shares'],t0_r[mae_r.
 ↪index(min(mae_r))],t1_r[mae_r.index(min(mae_r))])
plt.xlabel('# of Links')
plt.ylabel('# Shares')
plt.title('Curva de predicción')


print('El minimo encontrado fue un MAE de: ',min(mae_r))
print('Con un t0= ',t0_r[mae_r.index(min(mae_r))])
print('Con un t1= ',t1_r[mae_r.index(min(mae_r))])
```

```
El minimo encontrado fue un MAE de:  12242.617739175415
Con un t0=  2457.1390076734697
Con un t1=  328.94101755291064
```



Uso de Gradiente decendiente

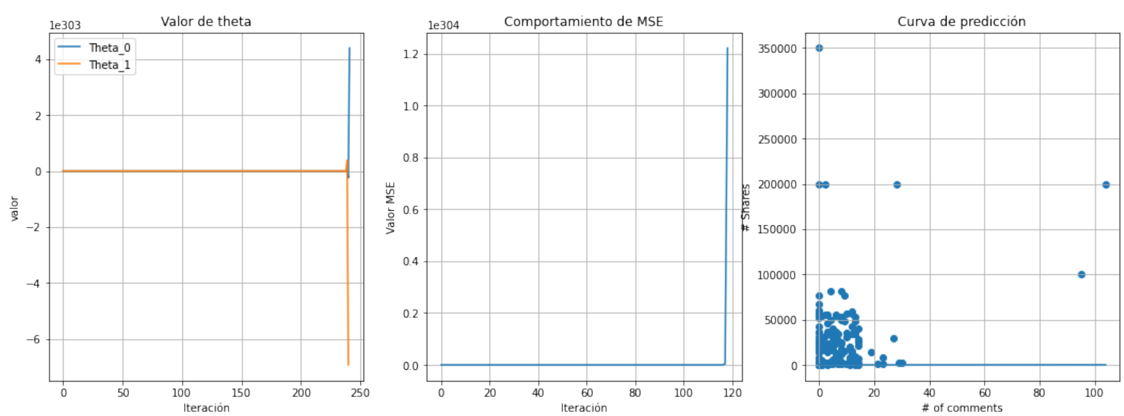
MSE # of comments vs # Shares

[27]:
```
alpha=0.1
t0oc=random.random()#5
t1oc=random.random()#0.3

resulgd=gradiente_mse(alpha,art['# of comments'],art['# Shares'],300,t0oc,t1oc)
#print()
itr=resulgd[0]
t0_r=resulgd[1]
t1_r=resulgd[2]
mse_r=resulgd[3]

plt.figure(figsize=(18,6))

plt.subplot(1,3,1)
plt.plot(resulgd[0],resulgd[1])
plt.plot(resulgd[0],resulgd[2])
plt.xlabel('Iteración')
plt.ylabel('valor')
plt.title('Valor de theta')
plt.legend(['Theta_0','Theta_1'])
plt.grid()
plt.subplot(1,3,2)

plt.plot(resulgd[0],resulgd[3])
plt.xlabel('Iteración')
plt.ylabel('Valor MSE')
plt.title('Comportamiento de MSE')
plt.grid()

plt.subplot(1,3,3)
```

```
plotpl(art['# of comments'],art['# Shares'],t0_r[mse_r.
 ↪index(min(mse_r))],t1_r[mse_r.index(min(mse_r))])
plt.xlabel('# of comments')
plt.ylabel('# Shares')
plt.title('Curva de predicción')


print('El minimo encontrado fue un MSE de: ',min(mse_r))
print('Con un t0= ',t0_r[mse_r.index(min(mse_r))])
print('Con un t1= ',t1_r[mse_r.index(min(mse_r))])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: RuntimeWarning:
overflow encountered in double_scalars

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:22: RuntimeWarning:
overflow encountered in double_scalars
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: RuntimeWarning:
invalid value encountered in double_scalars
  after removing the cwd from sys.path.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:22: RuntimeWarning:
invalid value encountered in double_scalars
```

```
El minimo encontrado fue un MSE de:  1326541580.7108095
Con un t0=  0.6695593888311785
Con un t1=  0.7972892538564829
```



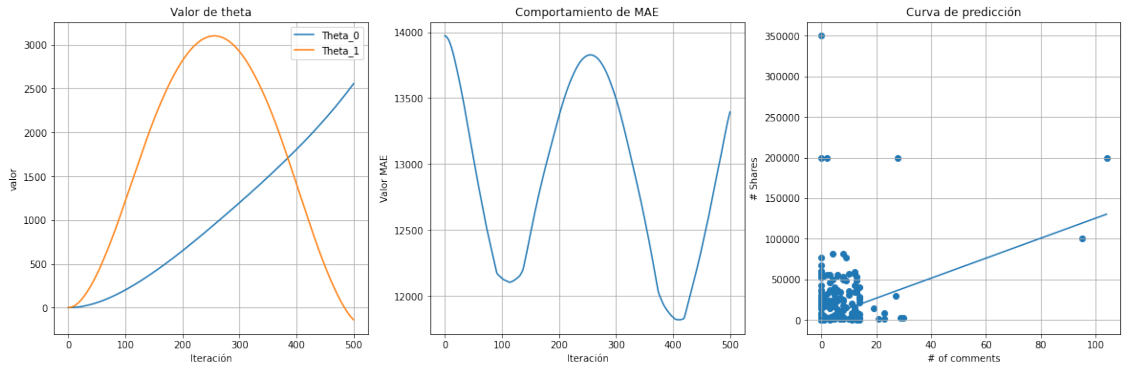
Uso de Gradiente decendiente

MAE # of comments vs # Shares

[28]:
```
alpha=0.1
t0oc=random.random()#5
t1oc=random.random()#0.3
```

```
resulgd=gradiente_mae(alpha,art['# of comments'],art['# Shares'],500,t0oc,t1oc) #
#print()
itr=resulgd[0]
t0_r=resulgd[1]
t1_r=resulgd[2]
mae_r=resulgd[3]

plt.figure(figsize=(20,6))

plt.subplot(1,3,1)
plt.plot(resulgd[0],resulgd[1])
plt.plot(resulgd[0],resulgd[2])
plt.xlabel('Iteración')
plt.ylabel('valor')
plt.title('Valor de theta')
plt.legend(['Theta_0','Theta_1'])
plt.grid()
plt.subplot(1,3,2)
plt.plot(resulgd[0],resulgd[3])
plt.xlabel('Iteración')
plt.ylabel('Valor MAE')
plt.title('Comportamiento de MAE')
plt.grid()

plt.subplot(1,3,3)
plotpl(art['# of comments'],art['# Shares'],t0_r[mae_r.
 ↪index(min(mae_r))],t1_r[mae_r.index(min(mae_r))])
plt.xlabel('# of comments')
plt.ylabel('# Shares')
plt.title('Curva de predicción')


print('El minimo encontrado fue un MAE de: ',min(mae_r))
print('Con un t0= ',t0_r[mae_r.index(min(mae_r))])
print('Con un t1= ',t1_r[mae_r.index(min(mae_r))])
```

```
El minimo encontrado fue un MAE de:  11818.984010985349
Con un t0=  1864.4341207738973
Con un t1=  1232.7899106974203
```



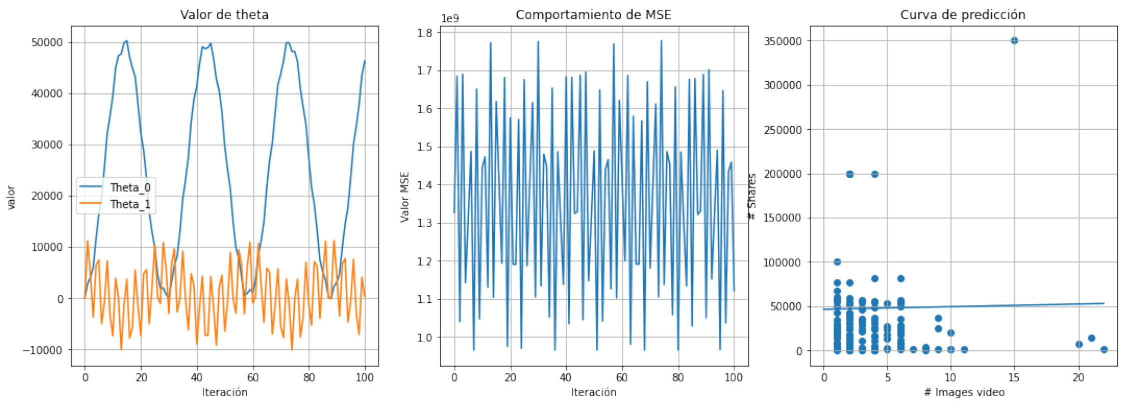
Uso de Gradiente decendiente

MSE # Images video vs Shares

[29]:
```
alpha=0.1
t0iv=random.random()#5
t1iv=random.random()#0.3

resulgd=gradiente_mse(alpha,art['# Images video'],art['# Shares'],100,t0iv,t1iv)
#print()
itr=resulgd[0]
t0_r=resulgd[1]
t1_r=resulgd[2]
mse_r=resulgd[3]

plt.figure(figsize=(18,6))

plt.subplot(1,3,1)
plt.plot(resulgd[0],resulgd[1])
plt.plot(resulgd[0],resulgd[2])
plt.xlabel('Iteración')
plt.ylabel('valor')
plt.title('Valor de theta')
plt.legend(['Theta_0','Theta_1'])
plt.grid()
plt.subplot(1,3,2)

plt.plot(resulgd[0],resulgd[3])
plt.xlabel('Iteración')
plt.ylabel('Valor MSE')
plt.title('Comportamiento de MSE')
plt.grid()

plt.subplot(1,3,3)
```

```
plotpl(art['# Images video'],art['# Shares'],t0_r[-1],t1_r[-1])
plt.xlabel('# Images video')
plt.ylabel('# Shares')
plt.title('Curva de predicción')


print('El minimo encontrado fue un MSE de: ',min(mse_r))
print('Con un t0= ',t0_r[mse_r.index(min(mse_r))])
print('Con un t1= ',t1_r[mse_r.index(min(mse_r))])
```

```
El minimo encontrado fue un MSE de:  965103300.7893695
Con un t0=  36075.24136278364
Con un t1=  -1410.2540585519728
```



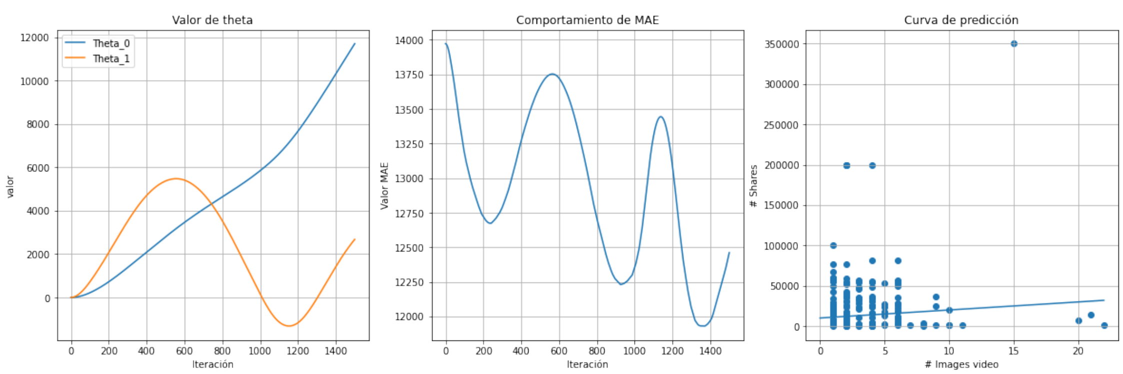
Uso de Gradiente decendiente

MAE # Images video vs Shares

[30]:
```
alpha=0.1
t0iv=random.random()#5
t1iv=random.random()#0.3

resulgd=gradiente_mae(alpha,art['# Images video'],art['#␣
 ↪Shares'],1500,t0iv,t1iv) #
#print()
itr=resulgd[0]
t0_r=resulgd[1]
t1_r=resulgd[2]
mae_r=resulgd[3]

plt.figure(figsize=(20,6))

plt.subplot(1,3,1)
plt.plot(resulgd[0],resulgd[1])
```

```
plt.plot(resulgd[0],resulgd[2])
plt.xlabel('Iteración')
plt.ylabel('valor')
plt.title('Valor de theta')
plt.legend(['Theta_0','Theta_1'])
plt.grid()
plt.subplot(1,3,2)
plt.plot(resulgd[0],resulgd[3])
plt.xlabel('Iteración')
plt.ylabel('Valor MAE')
plt.title('Comportamiento de MAE')
plt.grid()

plt.subplot(1,3,3)
plotpl(art['# Images video'],art['# Shares'],t0_r[mae_r.
 ↪index(min(mae_r))],t1_r[mae_r.index(min(mae_r))])
plt.xlabel('# Images video')
plt.ylabel('# Shares')
plt.title('Curva de predicción')


print('El minimo encontrado fue un MAE de: ',min(mae_r))
print('Con un t0= ',t0_r[mae_r.index(min(mae_r))])
print('Con un t1= ',t1_r[mae_r.index(min(mae_r))])
```

```
El minimo encontrado fue un MAE de:  11930.709780425448
Con un t0=  9905.673187401937
Con un t1=  996.1995655218662
```



Uso de Gradiente decendiente

MSE Elapsed days vs Shares

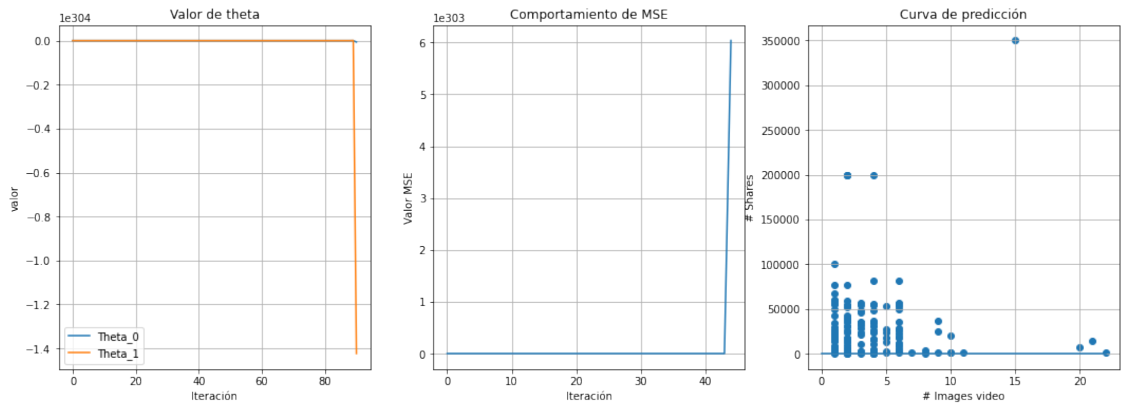
[31]:
```
alpha=0.1
t0ed=random.random()#5
```

```
t1ed=random.random()#0.3

resulgd=gradiente_mse(alpha,art['Elapsed days'],art['# Shares'],1000,t0ed,t1ed)
#print()
itr=resulgd[0]
t0_r=resulgd[1]
t1_r=resulgd[2]
mse_r=resulgd[3]

plt.figure(figsize=(18,6))

plt.subplot(1,3,1)
plt.plot(resulgd[0],resulgd[1])
plt.plot(resulgd[0],resulgd[2])
plt.xlabel('Iteración')
plt.ylabel('valor')
plt.title('Valor de theta')
plt.legend(['Theta_0','Theta_1'])
plt.grid()
plt.subplot(1,3,2)

plt.plot(resulgd[0],resulgd[3])
plt.xlabel('Iteración')
plt.ylabel('Valor MSE')
plt.title('Comportamiento de MSE')
plt.grid()

plt.subplot(1,3,3)
plotpl(art['# Images video'],art['# Shares'],t0_r[mse_r.
 ↪index(min(mse_r))],t1_r[mse_r.index(min(mse_r))])
plt.xlabel('# Images video')
plt.ylabel('# Shares')
plt.title('Curva de predicción')


print('El minimo encontrado fue un MSE de: ',min(mse_r))
print('Con un t0= ',t0_r[mse_r.index(min(mse_r))])
print('Con un t1= ',t1_r[mse_r.index(min(mse_r))])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: RuntimeWarning:
overflow encountered in double_scalars

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:22: RuntimeWarning:
overflow encountered in double_scalars
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:23: RuntimeWarning:
overflow encountered in double_scalars
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:22: RuntimeWarning:
invalid value encountered in double_scalars
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:23: RuntimeWarning:
invalid value encountered in double_scalars
```

```
El minimo encontrado fue un MSE de:  1323915728.486139
Con un t0=  0.10499728151359855
Con un t1=  0.6710932332521365
```



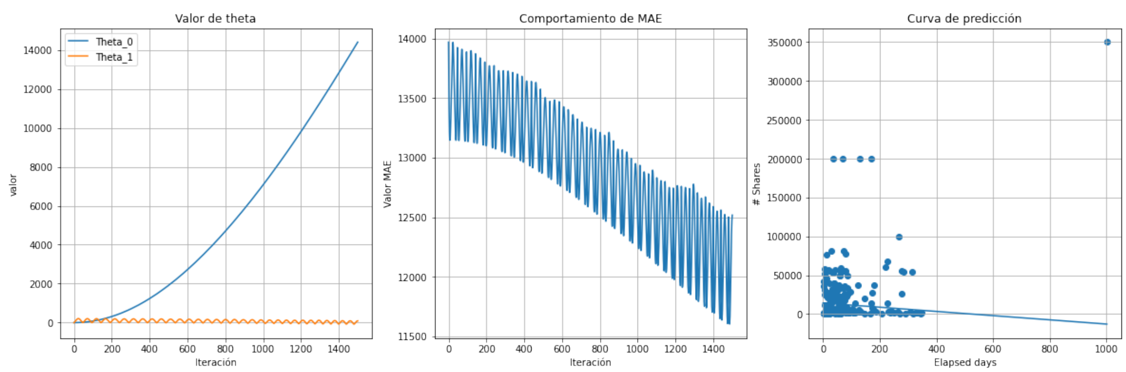
Uso de Gradiente decendiente

MAE Elapsed days vs Shares

[32]:
```
alpha=0.1
t0ed=random.random()#5
t1ed=random.random()#0.3

resulgd=gradiente_mae(alpha,art['Elapsed days'],art['# Shares'],1500,t0ed,t1ed) #
#print()
itr=resulgd[0]
t0_r=resulgd[1]
t1_r=resulgd[2]
mae_r=resulgd[3]

plt.figure(figsize=(20,6))

plt.subplot(1,3,1)
plt.plot(resulgd[0],resulgd[1])
plt.plot(resulgd[0],resulgd[2])
plt.xlabel('Iteración')
plt.ylabel('valor')
plt.title('Valor de theta')
plt.legend(['Theta_0','Theta_1'])
plt.grid()
plt.subplot(1,3,2)
plt.plot(resulgd[0],resulgd[3])
```

```
plt.xlabel('Iteración')
plt.ylabel('Valor MAE')
plt.title('Comportamiento de MAE')
plt.grid()

plt.subplot(1,3,3)
plotpl(art['Elapsed days'],art['# Shares'],t0_r[mae_r.index(min(mae_r))],t1_r[mae_r.index(min(mae_r))])
plt.xlabel('Elapsed days')
plt.ylabel('# Shares')
plt.title('Curva de predicción')


print('El minimo encontrado fue un MAE de: ',min(mae_r))
print('Con un t0= ',t0_r[mae_r.index(min(mae_r))])
print('Con un t1= ',t1_r[mae_r.index(min(mae_r))])
```

```
El minimo encontrado fue un MAE de:  11601.547823754303
Con un t0=  14170.805422682686
Con un t1=  -27.06079003197277
```



10 Forma matricial de la hipótesis de ecuación lineal

Empecemos con la hipótesis:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Los parámetros:

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$$

Y nuestras características X:

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ . & \\ . & \\ . & \\ 1 & x_m \end{bmatrix}$$

$^*x^0_{\{1,\ldots,m\}} = 1$

$^*x^1_{\{1,\ldots,m\}} = x_{\{1,\ldots,m\}}$

Con lo cuál:

$$h_\theta = X.\theta$$

Los parámetros:

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ . \\ . \\ . \\ \theta_n \end{bmatrix}$$

Las características:

$$X = \begin{bmatrix} x^0_1 & x^1_1 & . & . & x^n_1 \\ x^0_2 & x^1_2 & . & . & x^n_2 \\ . & & & & \\ . & & & & \\ . & & & & \\ x^0_m & x^1_m & . & . & x^n_m \end{bmatrix}$$

$^*x^0_{\{1,\ldots,m\}} = 1$

La hipótesis $h_\theta = X.\theta$:

$$h_\theta = \begin{bmatrix} x^0_1 & x^1_1 & . & . & x^n_1 \\ x^0_2 & x^1_2 & . & . & x^n_2 \\ . & & & & \\ . & & & & \\ . & & & & \\ x^0_m & x^1_m & . & . & x^n_m \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ . \\ . \\ . \\ \theta_n \end{bmatrix} = \begin{bmatrix} h_0 \\ h_1 \\ . \\ . \\ . \\ h_m \end{bmatrix}$$

$^*x^0_{\{1,\ldots,m\}} = 1$

[106]:
```
def mat_hip(x,thetas):
  mat_x=x.to_numpy()
  oness=np.ones((len(mat_x),1))
  #mat_y=y.to_numpy()
  mat_x=np.hstack((np.ones((len(mat_x),1)),mat_x))
  #thetas=np.random.rand(len(mat_x[0]),1) # en caso de requerir valores␣
 ↪aleatorios de inicio
  return mat_x.dot(thetas) # Uso de X.Theta


#print(np.ones((1,2)))
print(mat_hip(data[['TV','Radio']],np.random.rand(3,1)))
```

```
[[117.25652579]
 [ 38.13978965]
 [ 29.4377982 ]
 [ 85.06943045]
 [ 83.63432835]
 [ 27.16323409]
 [ 40.73887647]
 [ 61.62581265]
 [  5.58619959]
 [ 88.03116134]
 [ 32.0133451 ]
 [104.28483503]
 [ 27.30636743]
 [ 46.34339612]
 [103.82204386]
 [106.89100257]
 [ 46.91630709]
 [140.14346204]
 [ 40.11022285]
 [ 75.25721086]
 [107.57829083]
 [105.34926687]
 [ 13.91395263]
 [106.86584102]
 [ 33.5082439 ]
 [115.57791348]
 [ 75.84993259]
 [111.84774717]
 [120.37403812]
 [ 38.64163448]
 [139.88893853]
 [ 57.47457106]
 [ 43.40761123]
 [124.33102565]
 [ 42.71660698]
```

```
[127.80783051]
[135.84108728]
[ 55.77291792]
[ 31.7401729 ]
[116.30752501]
[ 98.25668617]
[ 92.39926464]
[139.91385712]
[ 93.75286833]
[ 23.54014776]
[ 86.56686999]
[ 44.0477377 ]
[123.1729464 ]
[105.88670551]
[ 35.07210131]
[ 88.26122582]
[ 48.5106373 ]
[113.16010773]
[100.69688296]
[127.13317718]
[109.17820161]
[ 16.99055991]
[ 68.32166879]
[114.38715876]
[105.09556735]
[ 24.84692774]
[132.92697772]
[110.95159704]
[ 58.70220313]
[ 76.98774154]
[ 34.87077971]
[ 25.785969  ]
[ 67.49204483]
[115.65615546]
[114.34438912]
[100.61377611]
[ 54.71518917]
[ 27.63007932]
[ 59.18596712]
[104.0019199 ]
[ 28.29651623]
[ 13.48302611]
[ 65.84995862]
[ 17.00180298]
[ 54.34430232]
[ 46.05898084]
[105.92112408]
[ 42.64116188]
```

```
[ 50.80932237]
[112.51129209]
[ 92.46323689]
[ 46.38408458]
[ 67.20357549]
[ 50.62375747]
[ 70.12950915]
[ 60.9248357 ]
[ 13.91000687]
[109.9460428 ]
[125.60223819]
[ 53.54516433]
[ 85.68011153]
[ 87.49929009]
[ 90.09062004]
[144.95476235]
[ 78.24457604]
[ 98.53125022]
[145.07494998]
[126.0536648 ]
[ 89.63211271]
[119.12902773]
[ 81.56816655]
[ 16.7332527 ]
[ 39.9314957 ]
[  6.73895441]
[123.11997427]
[101.78773353]
[122.33648468]
[ 83.55795095]
[100.527426  ]
[ 56.08156245]
[ 49.31905866]
[ 67.35701961]
[ 34.14164092]
[ 71.95101185]
[ 16.62592977]
[ 74.01161943]
[ 18.99066827]
[ 98.34499598]
[ 69.77473025]
[114.46782   ]
[ 43.84699713]
[ 22.17494973]
[ 35.40751584]
[118.1960266 ]
[ 32.0711821 ]
[ 19.44408095]
```

```
[116.29082284]
[ 17.0494375 ]
[110.84903069]
[ 34.54974322]
[ 43.31676069]
[ 29.87485997]
[131.90912663]
[ 31.32907031]
[100.6275731 ]
[ 40.30574729]
[100.50042623]
[111.01198797]
[ 48.52211016]
[ 49.09733209]
[ 62.12441423]
[107.522535  ]
[128.04289453]
[ 35.80495609]
[ 32.01404761]
[128.01715194]
[ 56.81636375]
[ 96.60984339]
[ 92.84710993]
[ 91.38361621]
[  8.02245061]
[ 61.31404635]
[ 65.93328207]
[ 22.93166933]
[ 66.01859158]
[ 83.42431759]
[ 54.24510069]
[ 90.26122588]
[ 88.15878094]
[ 58.08119808]
[103.32006437]
[ 25.91972384]
[ 92.23745625]
[104.40177941]
[128.04670563]
[ 27.75918589]
[ 81.2727248 ]
[ 18.59845733]
[ 76.59992273]
[ 98.11713416]
[142.48768727]
[121.62844019]
[ 77.69600262]
[120.95967914]
```

```
 [ 76.73031285]
 [ 69.45541049]
 [ 97.36041456]
 [ 27.7103893 ]
 [144.37386473]
 [119.85526135]
 [109.82261192]
 [ 61.87244464]
 [ 96.29957723]
 [130.29612138]
 [ 14.53043089]
 [ 36.81805061]
 [ 38.35593566]
 [ 10.2044079 ]
 [ 91.97043248]
 [ 81.67270579]
 [ 19.0502352 ]
 [ 43.68206697]
 [ 81.31015683]
 [142.19375884]
 [104.68074878]]
```

11 Uso de Gradiente descendiente de forma matricial

Se usa el gradiente descendiente para minimizar el error de MSE

Recordemos que MSE está definida como:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2$$

Puede expresarse como:

$$J(\theta) = \frac{1}{2m}(X.\theta - Y)^T.(X.\theta - Y)$$

donde

$$Y = \begin{bmatrix} y_0 \\ y_1 \\ . \\ . \\ . \\ y_m \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_0} \\ \frac{\partial J(\theta)}{\partial \theta_1} \\ . \\ . \\ . \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{bmatrix} = \frac{1}{m}(h_\theta - Y)X^T = \frac{1}{m}(X.\theta - Y)X^T$$

Finalmente, es posible expresar de una manera general al gradiente descendente con MSE:

$$\begin{bmatrix} \theta_0 \\ \theta_1 \\ . \\ . \\ . \\ \theta_n \end{bmatrix} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ . \\ . \\ . \\ \theta_n \end{bmatrix} - \alpha \begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_0} \\ \frac{\partial J(\theta)}{\partial \theta_1} \\ . \\ . \\ . \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{bmatrix}$$

[197]:
```
def gradiente_mat(x,y,alpha,thetas,iter):
  xm=x.to_numpy()
  xm=np.hstack((np.ones([len(xm),1]),xm))
  ym=y.to_numpy()
  ym=np.reshape(ym,(len(ym),1))
  tm=thetas#.to_numpy()
  tm=np.reshape(tm,(len(thetas),1))
 #h=x.dot(thetas)
  m=len(ym)
  dj=0
  j=0
  resultados=[]
  it=[]
  ts=np.empty([len(tm),1])

  #ts=np.array()
  for aa in range(iter):
    h=xm.dot(tm)
    j=(1/(2*m))*(h-ym).T.dot(h-ym)
    ts=np.hstack((ts,tm))
    #print(ts.shape)
    dj=(1/m)*((xm.T).dot(h-ym))
    tm=tm-(alpha*dj)
    it.append(aa)
    resultados.append(float(j))


  ts=np.delete(ts,0,axis=1)
  plt.figure(figsize=(14,6))
```

```
  plt.subplot(1,2,1)
  plt.plot(it,resultados)
  plt.title('MSE')
  plt.xlabel('Epocas')
  plt.grid()
  plt.subplot(1,2,2)
  for yy in range (len(ts)):
    plt.plot(it,ts[yy,:])
  plt.xlabel('Epocas')
  plt.ylabel('valor theta')
  plt.title('Valores de theta')
  plt.grid()
  return tm

thetas_res=gradiente_mat(data[['TV','Radio']],data['Sales'],0.0000007,np.random.rand(3,1),400)
print(thetas_res)
```

```
[[ 0.14205425]
 [-0.00183827]
 [ 0.69499611]]
```