Práctica 1 – Regresión lineal

Realizada por Mario Blanco Domínguez y Juan Tecedor Roa

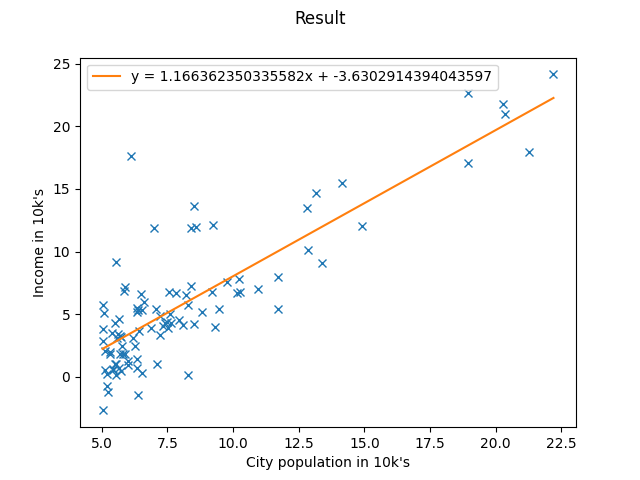
* Objetivo de la práctica

El objetivo de la práctica es aprender a aplicar la regresión lineal a un conjunto de datos, ya sea a un modelo con una variable o con varias.

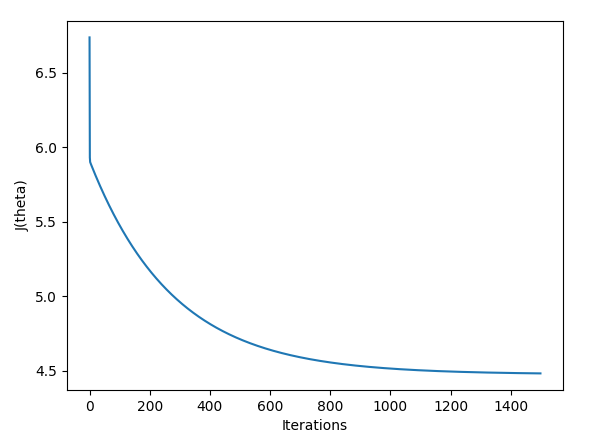
Se trata de aplicar el método de la regresión lineal a dos archivos csv. El primero de ellos representa los datos sobre beneficios de una compañía (la segunda columna) en base a la población (la primera columna). Tendremos que aplicar el método del descenso de gradiente para encontrar los parámetros Theta que definen la recta que se ajuste a los datos. Posteriormente, se aplicará la regresión lineal a unos datos con más variables: el precio de la casa, el tamaño de la casa y el número de habitaciones. También se aplicará el descenso de gradiente para optimizar los valores óptimos de la hipótesis de la regresión lineal.

* Código de la práctica: parte 1 (una variable)
* import numpy as np  
  import matplotlib.pyplot as plt  
  from pandas.io.parsers import read\_csv  
  from mpl\_toolkits.mplot3d import Axes3D  
  from matplotlib import cm  
  from matplotlib.ticker import LinearLocator, FormatStrFormatter  
    
    
  def getMat(file\_name):  
   return read\_csv(file\_name, header=None).to\_numpy().astype(float)  
    
    
  # Prediction of the value y. In this case using linear regression.  
  def hypothesis(X, theta\_0, theta\_1):  
   return X \* theta\_1 + theta\_0  
     
    
  # Cost function J(theta), measures how good are our guesses  
  def cost(X, Y, theta\_0, theta\_1):  
   return (1 / (2 \* len(X))) \* (((hypothesis(X, theta\_0, theta\_1) - Y)\*\*2).sum())  
    
    
  def gradientDescent(X, Y, iterations, alpha):  
   theta\_0 = theta\_1 = 0  
   costs = []  
    
   for i in range(iterations):  
   costs.append(cost(X, Y, theta\_0, theta\_1))  
   H = hypothesis(X, theta\_0, theta\_1)  
   temp\_0 = theta\_0 - (alpha / len(X)) \* (H - Y).sum()  
   temp\_1 = theta\_1 - (alpha / len(X)) \* ((H - Y) \* X).sum()  
   theta\_0 = temp\_0  
   theta\_1 = temp\_1  
    
   plt.suptitle('Cost')  
   plt.xlabel('Iterations')  
   plt.ylabel('J(theta)')  
   plt.plot(np.arange(0, iterations), costs)  
   plt.savefig('costs.png')  
   plt.show()  
   return theta\_0, theta\_1  
    
    
  def plot\_surfaces(theta\_0\_, theta\_1\_, theta\_0\_range, theta\_1\_range, X, Y):  
   step = .1  
   theta\_0 = np.arange(theta\_0\_range[0], theta\_0\_range[1], step)  
   theta\_1 = np.arange(theta\_1\_range[0], theta\_1\_range[1], step)  
   theta\_0, theta\_1 = np.meshgrid(theta\_0, theta\_1)  
   Cost = np.empty\_like(theta\_0)  
   for i\_x, i\_y in np.ndindex(theta\_0.shape):  
   Cost[i\_x, i\_y] = cost(X, Y, theta\_0[i\_x, i\_y], theta\_1[i\_x, i\_y])  
     
   fig = plt.figure()  
   ax = Axes3D(fig)  
   ax.plot\_surface(theta\_0, theta\_1, Cost, cmap=cm.coolwarm)  
   plt.show()  
     
   plt.contour(theta\_0, theta\_1, Cost, np.logspace(-2, 3, 20), colors='red')  
   plt.scatter(theta\_0\_, theta\_1\_)  
   plt.show()  
    
    
  def main():  
   data = getMat('./ex1data1.csv')  
   X = data[:, 0]  
   Y = data[:, 1]  
   alpha = 0.01  
   iterations = 1500  
    
   theta\_0, theta\_1 = gradientDescent(X, Y, iterations, alpha)  
   x = np.linspace(min(X), max(X), 100)  
   y = theta\_0 + theta\_1 \* x  
     
   plt.suptitle('Result')  
   plt.xlabel('City population in 10k\'s')  
   plt.ylabel('Income in 10k\'s')  
   plt.plot(X, Y, 'x')  
   plt.plot(x, y, label=('y = ' + str(theta\_1) + 'x + ' + str(theta\_0)))  
   plt.legend()  
   plt.savefig('result.png')  
   plt.show()  
    
   plot\_surfaces(theta\_0, theta\_1, [-10, 10], [-1, 4], X, Y)  
     
    
  main()
* Resultados de ejecución: parte 1 (una variable)

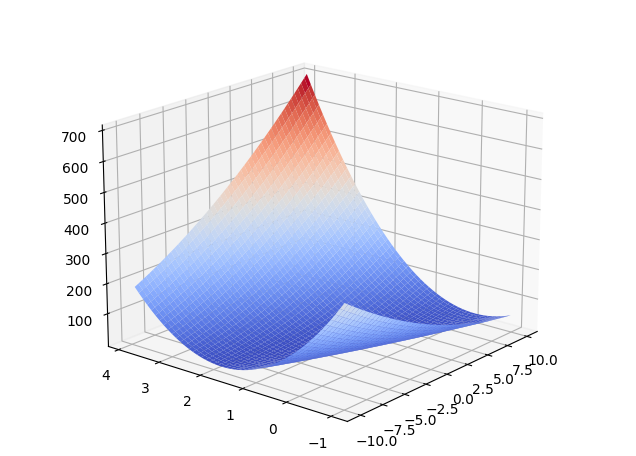
Se generan varias gráficas. En primer lugar, hemos obtenido la gráfica que los datos de entrada junto a la línea de predicción que mejor se ajusta con esta hipótesis.



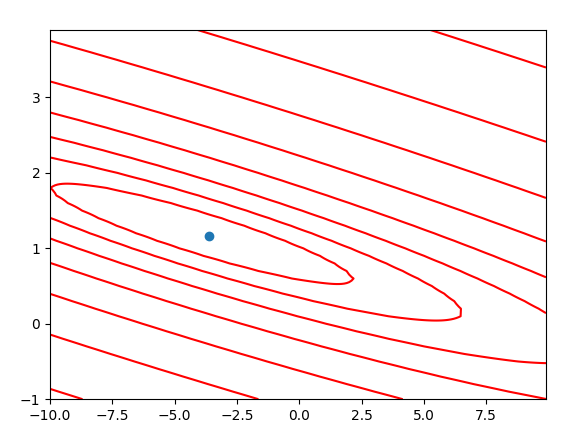
Además, hemos generado gráficas con respecto al coste. Podemos ver como el coste se va estabilizando rápidamente con Alpha = 0,01, y a la par que van aumentando las iteraciones disminuye lentamente.



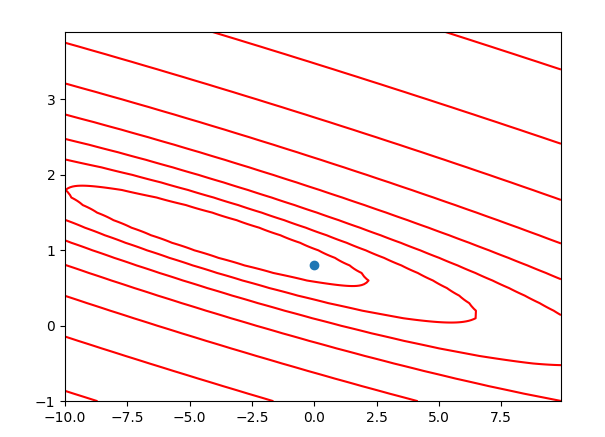
Se ha calculado también el coste dentro de los intervalos θ0 ∈ [−10, 10] y θ1 ∈ [−1, 4].



En la segunda gráfica, hemos pintado el mínimo de coste obtenido por el descenso de gradiente.



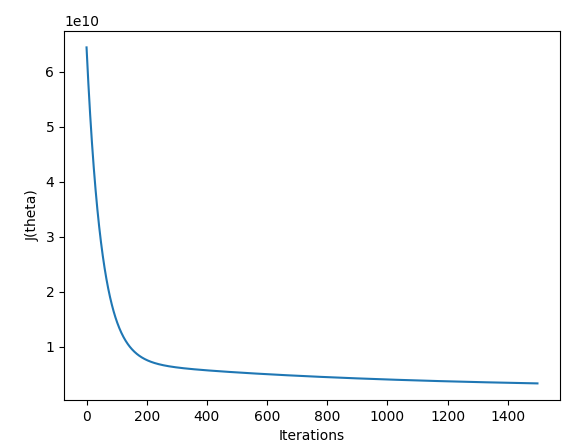
Con un valor Alpha menor, podemos comprobar que no encontramos un mejor mínimo, si no que obtenemos uno peor.



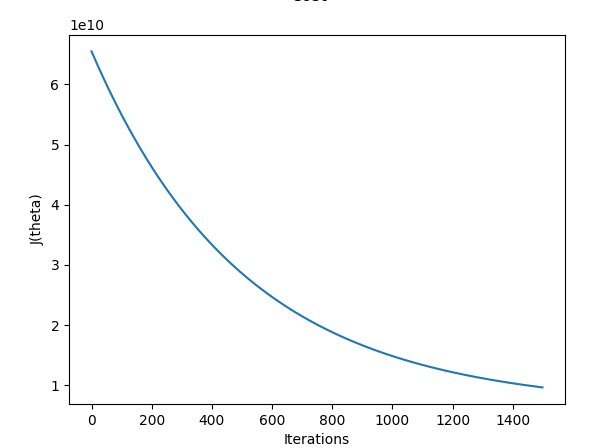
* Código de la práctica: parte 2 (varias variables)
* def getMat(file\_name):  
   return read\_csv(file\_name, header=None).to\_numpy().astype(float)  
    
    
  def gradient(X, Y, T, alpha):  
   new\_T = T  
   m = np.shape(X)[0]  
   n = np.shape(X)[1]  
   H = np.dot(X, T)  
   aux = (H - Y)  
   for i in range(n):  
   aux\_i = aux \* X[:, i]  
   new\_T[i] -= (alpha / m) \* aux\_i.sum()  
   return new\_T  
    
    
  def cost(X, Y, T):  
   XTY = np.matmul(X, T) - Y  
   return 1 / (2 \* np.shape(X)[0]) \* (np.matmul(np.transpose(XTY), XTY))  
    
  def normalize(X):  
   ranges = [ 0 ]  
   averages = [ 1 ]  
   XNorm = np.copy(X)  
   for i in range(1, np.shape(X)[1]):  
   col = XNorm[:, i]  
   ran = np.max(col) - np.min(col)  
   avg = np.average(col)  
   col -= avg  
   col /= ran  
   ranges.append(ran)  
   averages.append(avg)  
   averages = np.array(averages)  
   ranges = np.array(ranges)  
   return XNorm, averages, ranges  
    
  def normalize2(X):  
   averages = np.mean(X, axis=0)  
   ranges = np.std(X, axis=0)  
   return X/(averages - ranges), averages, ranges  
  def main():  
    
   data = getMat('./ex1data2.csv')  
   X = data[:, :-1]  
   X = np.hstack([np.ones([len(X), 1]), X])  
   Y = data[:, -1]  
   XNorm,averages,ranges = normalize(X)  
    
   costs = []  
   alpha = 0.01  
   iterations = 1500  
   T = np.zeros(np.shape(X)[1])  
   for i in range(iterations):  
   T = gradient(XNorm, Y, T, alpha)  
   costs.append(cost(XNorm, Y, T))  
    
   plt.suptitle('Cost')  
   plt.xlabel('Iterations')  
   plt.ylabel('J(theta)')  
   plt.plot(np.arange(0, iterations), costs)  
   plt.savefig('costs.png')  
   plt.show()  
    
   print(T)  
   for i in range(len(X)):  
   print(np.dot(np.transpose(T), X[i]))  
    
  main()
* Resultados de ejecución: parte 2 (varias variables)

Se ha generado la gráfica del coste teniendo en cuenta la tasa de aprendizaje y 1500 iteraciones.

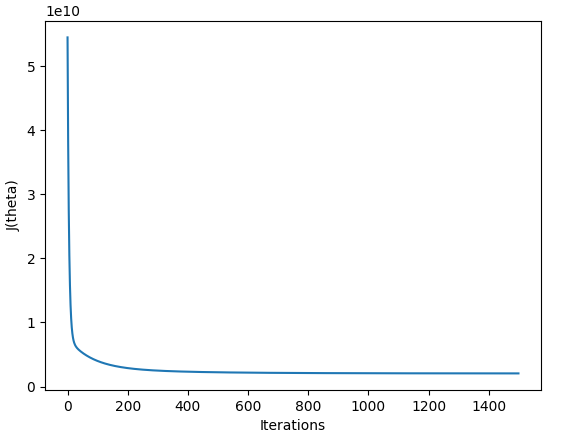
Esta gráfica muestra el coste J(theta) con una tasa de aprendizaje de 0,01.

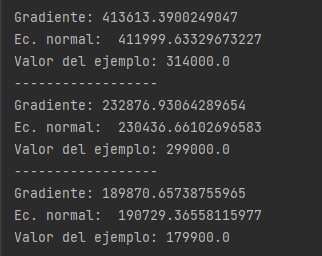


Si reducimos esa tasa de aprendizaje (por ejemplo, a 0,001), podemos ver que el coste no llega a converger en nuestras 1500 iteraciones, ya que da saltos muy pequeños



Si aumentamos la tasa (por ejemplo, a 0,1), obtenemos convergencia muy rápido, pero los valores se distancian, en ocasiones, bastante de los valores reales de los ejemplos de entrenamiento.





Estos valores mostrados, se han obtenido para todos los ejemplos del csv. Se comprueba que valor predicen nuestros dos algoritmos (tanto el implementado con el descenso gradiente, como el implementado mediante la ecuación normal). A continuación, se muestran los resultados para todos los valores del csv con 1500 iteraciones y alpha= 0,01.

Predicciones del CSV usando grandiente y la ecuacion normal:

Gradiente: 343568.70614447986

Ec. normal: 356283.11033889925

Valor del ejemplo: 399900.0

------------------

Gradiente: 311159.49006306095

Ec. normal: 286120.93063401605

Valor del ejemplo: 329900.0

------------------

Gradiente: 362602.6901922973

Ec. normal: 397489.4698481164

Valor del ejemplo: 369000.0

------------------

Gradiente: 278837.3668245447

Ec. normal: 269244.1857271005

Valor del ejemplo: 232000.0

------------------

Gradiente: 421675.2774980165

Ec. normal: 472277.8551463641

Valor del ejemplo: 539900.0

------------------

Gradiente: 356406.71733404783

Ec. normal: 330979.02101847425

Valor del ejemplo: 299900.0

------------------

Gradiente: 306915.4260523989

Ec. normal: 276933.0261488528

Valor del ejemplo: 314900.0

------------------

Gradiente: 300034.8980351136

Ec. normal: 262037.4840289668

Valor del ejemplo: 198999.0

------------------

Gradiente: 297012.6100275209

Ec. normal: 255494.58235013846

Valor del ejemplo: 212000.0

------------------

Gradiente: 304343.2660459371

Ec. normal: 271364.5991881477

Valor del ejemplo: 242500.0

------------------

Gradiente: 353513.0373267783

Ec. normal: 324714.54068768106

Valor del ejemplo: 239999.0

------------------

Gradiente: 336881.0901276791

Ec. normal: 341805.2002410662

Valor del ejemplo: 347000.0

------------------

Gradiente: 329807.6501099091

Ec. normal: 326492.0260991274

Valor del ejemplo: 329999.0

------------------

Gradiente: 537206.7769455726

Ec. normal: 669293.2122320869

Valor del ejemplo: 699900.0

------------------

Gradiente: 289810.5620094278

Ec. normal: 239902.98686016432

Valor del ejemplo: 259900.0

------------------

Gradiente: 376662.47738493467

Ec. normal: 374830.38333402626

Valor del ejemplo: 449900.0

------------------

Gradiente: 272664.1828090363

Ec. normal: 255879.9610214085

Valor del ejemplo: 299900.0

------------------

Gradiente: 287752.8340042584

Ec. normal: 235448.24529160035

Valor del ejemplo: 199900.0

------------------

Gradiente: 396532.4134348522

Ec. normal: 417846.4816054725

Valor del ejemplo: 499998.0

------------------

Gradiente: 423668.7015030244

Ec. normal: 476593.3860409105

Valor del ejemplo: 599000.0

------------------

Gradiente: 321898.258090039

Ec. normal: 309369.1131949595

Valor del ejemplo: 252900.0

------------------

Gradiente: 309188.8549007941

Ec. normal: 334951.62386341975

Valor del ejemplo: 255000.0

------------------

Gradiente: 311416.7060637071

Ec. normal: 286677.7733300865

Valor del ejemplo: 242900.0

------------------

Gradiente: 354927.7253303323

Ec. normal: 327777.1755160688

Valor del ejemplo: 259900.0

------------------

Gradiente: 458415.65043300006

Ec. normal: 604913.3741343784

Valor del ejemplo: 573900.0

------------------

Gradiente: 279007.4899822882

Ec. normal: 216515.5936252033

Valor del ejemplo: 249900.0

------------------

Gradiente: 302028.3220401215

Ec. normal: 266353.01492351317

Valor del ejemplo: 464500.0

------------------

Gradiente: 370704.99421265203

Ec. normal: 415030.01477433724

Valor del ejemplo: 469000.0

------------------

Gradiente: 349741.8901599882

Ec. normal: 369647.33504459134

Valor del ejemplo: 475000.0

------------------

Gradiente: 377842.73823058355

Ec. normal: 430482.39959029364

Valor del ejemplo: 299900.0

------------------

Gradiente: 306037.9588928784

Ec. normal: 328130.3008365561

Valor del ejemplo: 349900.0

------------------

Gradiente: 231596.71554854987

Ec. normal: 220070.56444809595

Valor del ejemplo: 169900.0

------------------

Gradiente: 359943.43734293286

Ec. normal: 338635.60808944365

Valor del ejemplo: 314900.0

------------------

Gradiente: 409994.7383113563

Ec. normal: 500087.73659910634

Valor del ejemplo: 579900.0

------------------

Gradiente: 345217.82130593894

Ec. normal: 306756.3637394074

Valor del ejemplo: 285900.0

------------------

Gradiente: 300677.938036729

Ec. normal: 263429.5907691431

Valor del ejemplo: 249900.0

------------------

Gradiente: 287945.74600474304

Ec. normal: 235865.87731365324

Valor del ejemplo: 229900.0

------------------

Gradiente: 365859.405357795

Ec. normal: 351442.9900990652

Valor del ejemplo: 345000.0

------------------

Gradiente: 499804.6376942942

Ec. normal: 641418.8240777791

Valor del ejemplo: 549000.0

------------------

Gradiente: 367788.5253626414

Ec. normal: 355619.31031959393

Valor del ejemplo: 287000.0

------------------

Gradiente: 294784.75886460795

Ec. normal: 303768.43288347166

Valor del ejemplo: 368500.0

------------------

Gradiente: 352185.4421661269

Ec. normal: 374937.3406572611

Valor del ejemplo: 329900.0

------------------

Gradiente: 393831.6454280673

Ec. normal: 411999.63329673227

Valor del ejemplo: 314000.0

------------------

Gradiente: 285437.8899984427

Ec. normal: 230436.66102696583

Valor del ejemplo: 299000.0

------------------

Gradiente: 242569.91073343306

Ec. normal: 190729.36558115977

Valor del ejemplo: 179900.0

------------------

Gradiente: 347854.2853125623

Ec. normal: 312464.00137413

Valor del ejemplo: 299900.0

------------------

Gradiente: 285630.8019989274

Ec. normal: 230854.2930490187

Valor del ejemplo: 239500.0

------------------

Para finalizar, se ha hecho la comparación entre el gradiente y la ecuación normal con el ejemplo concreto de 1500 pies cuadrados y 3 habitaciones.

Casa con una superficie de 1.650 pies cuadrados y 3 habitaciones:

Ec.normal: 293081.4643348973

Gradiente: 314374.6900711382