Inteligencia Artificial
Grado en Ingeniería Informática en Sistemas de Información
ENSEÑANZAS PRÁCTICAS Y DE DESARROLLO

EPD2.1: Machine Learning – Regresión Lineal univariable

Objetivos

- Implementación en Python de un algoritmo de Regresión Lineal univariable y su aplicación a datos.

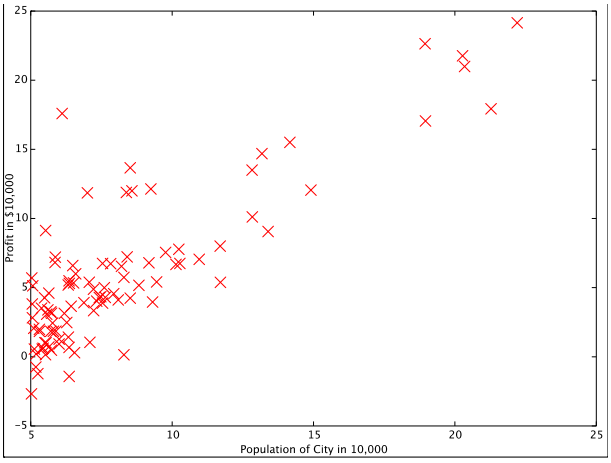
Bibliografía Básica

Machine Learning. Tom Mitchell. MacGraw-Hill, 1997

Ejercicios

Implementar y aplicar regresión lineal con una variable para predecir los beneficios de un restaurante ambulante. Ponte en el papel del Director Ejecutivo de una franquicia de este tipo de restaurantes y estás considerando diferentes ciudades para la apertura de un nuevo establecimiento. La cadena ya cuenta con restaurantes en varias ciudades, y dispone de los datos de los beneficios y las poblaciones de las ciudades. Deberías utilizar los datos como ayuda para seleccionar en qué ciudad poner el siguiente establecimiento.

El fichero `ex1data1.txt` contiene los datos para nuestro problema de regresión lineal. La primera columna es la población de una ciudad y la segunda es el beneficio de un establecimiento en esa ciudad. Un valor negativo de beneficio indica una pérdida.



Utilizar el script `main.py` para ir incorporando el código y las llamadas a las funciones que se piden en los siguientes ejercicios.

EJ1. Normalmente, antes de empezar cualquier tarea se suele visualizar el conjunto de datos. Añadir al fichero `main.py` el código necesario para cargar los datos, e implementar la función `plotData(X,y)` de `plotData.py` para representar los datos gráficamente. A la derecha se muestra un ejemplo del resultado de la visualización.

EJ2. El objetivo de la regresión lineal es minimizar el coste de la función:

$$J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Implementar la función `computeCost(X,y,theta)` de manera que devuelva el valor del coste. La matriz *X* es la matriz formada por la población de cada ciudad e *y* es el vector formado por los beneficios del establecimiento de cada ciudad. Para facilitar las operaciones vectorizadas, añada una columna con todos sus elementos a 1 a la matriz *X* como primera columna, de esta manera se podrá calcular la hipótesis *h* que viene dada por el modelo lineal:

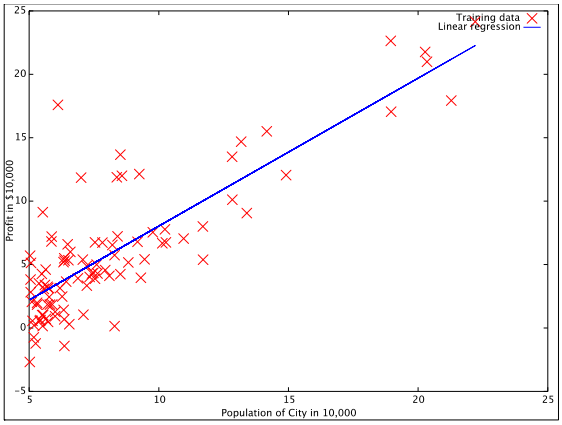
$$h_\theta(x) = \theta_0 + \theta_1 x$$

Para comenzar inicializar los parámetros `theta` a 0. El resultado del coste inicial debe ser 32,07.

EJ3. Implementar el método de descenso del gradiente en el fichero `gradientDescent.py` (ver documento de EB, p. 42). La función recibirá como parámetro, además de los datos X e y, los parámetros `theta`, `alpha` y el número de iteraciones. Estos dos últimos se pueden inicializar a 0,01 y 1500 respectivamente. La función deberá devolver los parámetros `theta` finales y un

histórico con el coste en cada iteración. Mostrar por pantalla los valores de `theta`. Crear la función `plotIterationsVsCost()` para comprobar que el descenso del gradiente para los parámetros indicados funciona correctamente. Se representará el número de iteraciones vs el coste.

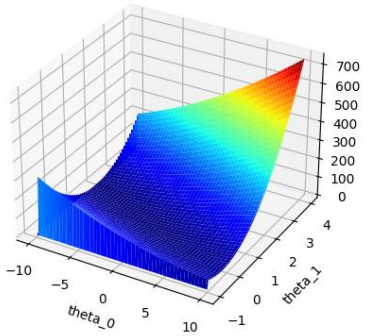
EJ4. Representar gráficamente los resultados obtenidos y calcular la predicción de beneficios para una población de 35000 habitantes y para otra de 70000 habitantes. Mantener la figura anterior para poder comprobar la diferencia con los datos originales como en la figura de la derecha.

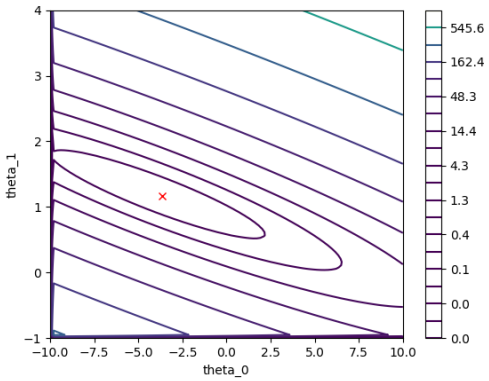


Problemas

P1. En el caso de estudio utilizado en los ejercicios anteriores, probar con diferentes valores de `alpha` (0.1, 0.03, 0.01, …). Representar y comparar los diferentes resultados obtenidos con la función `plotIterationsVsCost()` para comprobar qué valores de `alpha` son válidos y cuáles no. Finalmente, utilizar los mejores valores de `theta` para predecir de nuevo los beneficios para una población de 35000 y 70000 habitantes. Curiosidad: realizar también ambas predicciones utilizando diferentes modelos de regresión lineal de la librería sklearn (`sklearn.linear_model`) como por ejemplo: `LinearRegression()`, `Ridge()` o `Lasso()`.

P2. Incluir el código necesario en `main.py` para obtener las gráficas de la función de coste que se muestran a continuación. Para ello, utilizar la función `plotData_cost()` proporcionada que hace uso de los comandos plot_surface y contour.





P3. Utilizar la regresión lineal para predecir los precios de casas. Suponer que queremos vender nuestra casa y queremos saber cuál sería un buen precio de mercado. Una opción sería recolectar información de ventas de casas recientes y crear un modelo de precios de casas.

El fichero `ex1data2.txt` contiene un conjunto de entrenamiento de precios de casas en una ciudad estadounidense. La primera columna corresponde con el tamaño de la casa (en pies cuadrados), la segunda columna indica el número de dormitorios, y la tercera el precio de la casa.

Utilizando la función de coste y el método de descenso del gradiente antes implementados, indica cuál de los dos atributos determina mejor el precio de la casa. Considerar los valores 0,03 y 200 para `alpha` y el número de iteraciones respectivamente. Como en el ejercicio 4, visualizar los resultados obtenidos con cada una de las opciones o atributos.