Universidad Simón Bolívar
Departamento de computación
Inteligencia aritifical CI5437

Proyecto 1
Regresión Lineal

Profesora :
Carolina Martínez

Grupo:
Carlos Farinha - 09-10270
Javier López - 11-10552
Nabil Márquez - 11-10683

20 de febrero del 2017

Introducción

El objetivo de este proyecto es introducir algunos aspectos teóricos el método numérico del gradiente descendente, la regresión lineal y presentar una breve implementación en Python. La técnica de la regresión lineal se clasifica en la categoría del aprendizaje mecánico supervisado y dentro de esta es de tipo aprendizaje mecánico por regresión. El objetivo de la regresión lineal es ajustar una aplicación afín a un conjunto de datos. La idea es poder utilizar esta estrategia para predecir posibles resultados en un marco general de observaciones, permitiendo asi tener una funcion que reduzca el error en cada uno de sus puntos a fin de tener una proyección de los resultados de manera eficiente.

Detalles de implementación/experimentación.

Se utilizaron las siguientes librerías:

- *numpy '1.11.3'*
- *matplotlib '1.4.2'*

Ejercicio 1

Para el primer ejercicio implementamos las siguientes funciones:

- *normalizacion*: Calcula media y desviación estándar y aplica normalización a las columnas deseadas. Si se provee una media y una desviación el algoritmo no lo recalcula (útil para aplicar sobre conjunto de pruebas).
- *costFuntionJ*: Calcula el costo a minimizar por *gradient descent*
- *gradientDescent*: Algoritmo de descenso de gradiente, entrena un vector *theta* de coeficientes reduciendo el error cuadrático medio entre $x \times \theta$ y y por tantas iteraciones como se indique.
- *normalizacion_l*: versión alternativa de normalización, agrega un vector de unos al inicio de la matriz.
- *eval_model*: implementa las métricas necesaria para el modelo del ejercicio3.

Ejercicio 2

Para este ejercicio se realizaron cambios ya que para ningún α utilizado se pudo obtener convergencia sin realizar antes normalización de los datos.

Se tuvieron ciertos problemas con la obteción de los datos para las gráficas, pero se rediseñaron las funciones de la pregunta 1 para obteneralos adecuadamente.

Ejercicio 3

Para realizar este ejercicio primero se limpiaron los datos, luego se tuvo que lidiar con los datos faltantes, para esto, en las columnas con datos de tipo nominal sustituimos por la moda de los datos y para el resto sustituimos con la media de tal manera que no modificara la media ni la desviación que posee. Luego de esto cambiamos lás columnas nominales por numéricas cambiando cada etiqueta por un valor diferente.

Posterior a esto separamos los datos en *training set* y *test set* haciendo *shuffle* de los datos, el resultado de esta mezcla de filas será siempre el mismo ya que mantenemos una semilla fija para este comportamiento "aleatorio" (*np.random.seed(42)*).

Luego de esto normalizamos del conjunto de entrenamiento y almacenamos la media y la desviación para normalizar, posteriormente hacemos uso de este μ y σ para normalizar el conjunto de pueba, hicimos esto ya

que en la vida real uno no puede tener acceso a la media real del universo de datos sino solo de la muestra así que se debe almacenar la escala que se le aplica al entrenamiento.

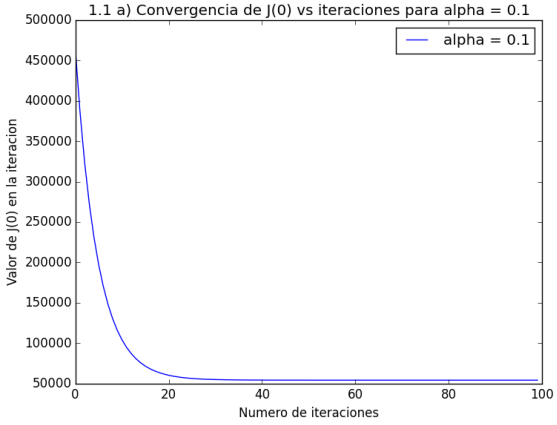
Para este ejercicio se realizaron varias pruebas hasta obtener un número de iteraciones adecuado, utilizando *113* como cantidad ideal de iteraciones.

Resultados y discusión

Ejercicio 2

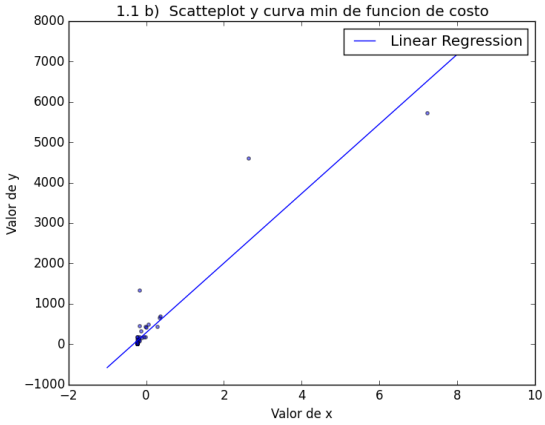
Parte 1

Para la matriz de pesos corporales obtuvimos la siguiente curva de convergencia con alpha = 1 y realizando normalización de los datos



Aquí se puede obserbar como el error desciende drasticamente en las primera 20 iteraciones y converge hasta su resultado final a las 100 iteraciones.

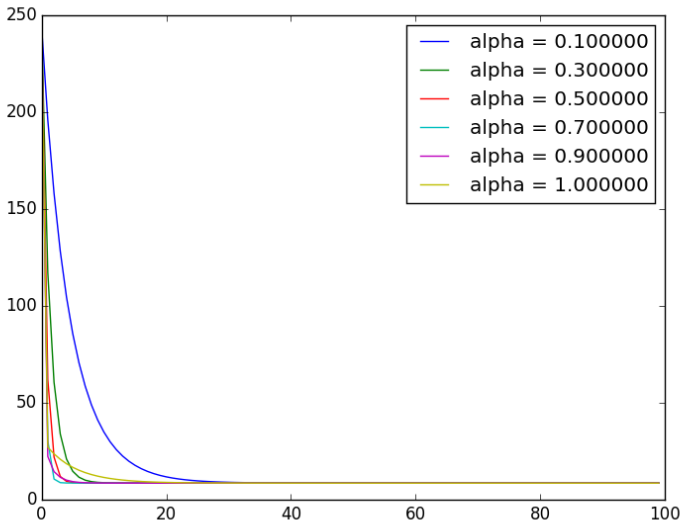
Graficando en los ejes x,y los puntos de los datos y graficando la función representada por los coeficientes de θ los cuales fueron entrenados por nuestro algoritmo de *gradient descent* obtenemos el siguiente *scatter plot*.



Aquí se puede observar como la mayoría de los datos se encuentran agrupados en la parte inferior y están distribuidos de forma similar a la de la función entrenada pero existen 2 ó 3 valores atípicos. Para adaptarse mejor a todos los puntos se podría usar una función de mayor grado pero se estaría corriendo riesgo de estar generando overfitting, una mejor decisión sería recolectar más datos y observar si los valores atípicos son despreciables o es necesario entrenar otro modelo

Parte 2

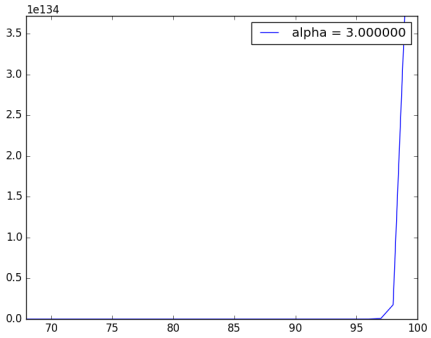
A continuación se muestra la comparación de las curvas de convergencia para todos los α haciendo uso de los datos normalizados

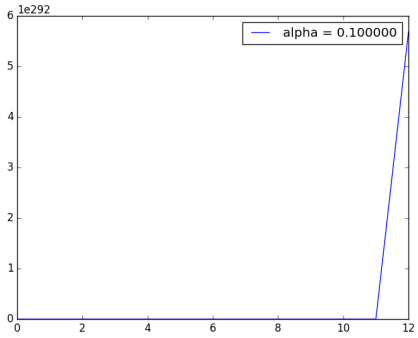


Con $\alpha = 0.1$ se obtuvo la convergencia más lenta y con $\alpha = 0.7$ la más rápida, además se puede observar que las curva para 0.7 y 0.9 tuvieron cambios más bruscos que las demás, esto es resultado de grandes reducciones del costo J en pocas iteraciones.

Por último es imporante destacar que $\alpha = 1$ no fue la convergencia más rápida a pesar de ser el mayor rate de entranamiento, esto puede ser debido a que el modelo en vez de acercarse lentamente a su valor valor final da pasos muy grandes pasando de errores negativos a positivos pero sin ser tan grandes como para llegar a ser divergentes, ese cambio de símbolos no podría ser apreciado en la gráfica ya que el error cuadrático medio por definición es una medida positiva.

Para mostrar esto con un gran *learning rate* ($\alpha = 3$) se puede observar la divergencia del error incluso con datos normalizados. A la derecha se puede observar el mismo conjunto de datos con un ratio de aprendizaje pequeño pero sin normalizar.





Ejercicio 3

La siguiente tabla representa los resultados para las métricas planteadas y además la función de costo. Esto se realizó para 113 iteraciones que es en donde logramos minimizar la función de costo para el conjunto de prueba.

Métrica	Training Set	Testing Set
Training set cost	0.0852	0.5293
Bias	0.2913	1.3246
Max Deviation	16.460	2.8914
Mean Absolute Deviation	0.3117	1.3247
Mean Square Error	0.1309	1.7923

Con más iteraciones se reduce el error del entrenamiento y aumenta el de prueba, mostrando así que la capacidad de generalización disminuye y el algoritmo está intentando memorizar los datos. Con 1000 iteraciones se obtiene la siguiente tabla con gran mejoría en el primer conjunto, y una diminución pequeña en el costo del segundo pero, aún así, mejorando algunas métricas un poco.

Métricas	Training Set	Testing Set
Training set cost	0.0060	0.5325
Bias	0.2913	1.3247
Max Deviation	2.0406	2.8640
Mean Absolute Deviation	0.3041	1.3247
Mean Square Error	0.1085	1.7802

Podemos observar que ambos casos y ambos conjuntos de datos se obtuvo un *bias* o *sesgo* positvo, lo cual sugiere que el modelo tiende a sobrestimar los precios, aún así los valores son pequeños y en el training set tiende a cero. La máxima desviación de nuestro objetivo a aproxima fue de 16 al usar 114 iteraciones, esta peor predicción fue mejorada en el segundo modelo. La media de desviaciones absolutas o error promedio fue muy pequeño para el training set y pequeños pero cercanos a 1 en el conjunto de pruebas. Por último tenemos el error cuadrático medio.

Conclusiones

- *Se puede observar como la regresion lineal en conjunto con el metodo de gradiente permiten conseguir un estimado proximo de una funcion que nos permita evaluar los datos de entrada generando un resultado proximo a lo esperado, validando asi los resultados de un conjunto de entrenamiento y siendo optimos para un grupo de pruebas.
- *La funcion de costo a lo largo de las iteraciones siempre disminuira pero dependiendo del coeficiente de aprendizaje que se toma esta puede variar drasticamente permitiendo una convergencia en menor numero de iteraciones.
- *Si el coeficiente de aprendizaje es bruscamente alterado la convergencia de la funcion puede ser comprometida generando que esta diverga y nunca logre alcanzar una aproximación cierta.
- *A la hora de seleccionar un conjunto de entrenamiento se debe tener en cuenta el error del mismo puede generar un fallo si el conjunto no converge tras un numero elevado de iteraciones.
- *Usualmente los datos deben ser normalizados para poder apreciar en un grafico con mayor detalle la convergencia de los mismos, esto devido a que si el error crece de manera desmesurada en alguno de los coeficientes de θ, esto podria generar un overflow.