Regresión Lineal con Multiples Variables

Sebastián Marroquín

Octubre, 2018

1 Modelo General y Notación

Los modelos de regresión lineal están entre los modelos de predicción más simples para una salida única basada en múltiples variables de entrada o características.

Se supone que la salida es una función continua y lineal de todas las entradas consideradas. Específicamente, la función de hipótesis general relaciona la salida pronosticada $\hat{y}$ y n características $x_1, x_2, ..., x_n$ con la siguiente expresión,

$$\hat{y} = h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n \tag{1}$$

Una forma alternativa de la ecuación anterior (1) se puede expresar como el siguiente producto:

$$h_\theta(x) = \theta^T \cdot x \tag{2}$$

donde, $\theta = [\theta_0, \theta_1, \theta_2, ..., \theta_n]^T$ es un vector parametrico, y $x = [x_0, x_1, x_2, ..., x_n]^T$ representa todas las características observadas.

Ambas ecuaciones (1) y (2) muestran la solución del problema directo: encontrar un valor de predicción a partir de cualquier conjunto de características observadas. Por lo tanto, resolver el problema hacia adelante implica que la relación entre entrada y salida puede definirse "determinísticamente".

Considerando una colección de m ejemplos de capacitación independiente, por ejemplo, un conjunto de datos que comprenda un objetivo particular (observaciones de salida) y las entradas correspondientes. Supongamos ahora que ya se conoce la solución del problema de regresión lineal (se conocen el conjunto de valores de parámetros que "mejor describen" la entrada y el destino). Esto significa que para cada ejemplo de entrada de entrenamiento, el valor predicho por (1) es aproximadamente igual al valor objetivo respectivo, dicho de otra forma esto es:

$$y \approx X\theta \tag{3}$$

La expresión (3) es una forma informal de definir la característica de solución del problema de regresión lineal y, por lo tanto, no representa un sistema de ecuaciones lineales. Su lado derecho representa los valores predichos utilizando la forma vectorizada de la ecuación (2) al considerar todos los ejemplos de entrenamiento m. El término del lado izquierdo representa el objetivo en una forma vectorial.

La notación considerada en este trabajo es que las entradas de entrenamiento se almacenan en la matriz $X = m \cdot (n+1)$ y el objetivo está representado por el vector de columna y:

$$X = \begin{bmatrix} x_0^1 & x_1^1 & x_2^1 & ... & x_n^1 \\ x_0^2 & x_1^2 & x_2^2 & ... & x_n^2 \\ x_0^3 & x_1^3 & x_2^3 & ... & x_n^3 \\ ... & ... & ... & ... & ... \\ x_0^m & x_1^m & x_2^m & ... & x_n^m \end{bmatrix}, y = \begin{bmatrix} y^1 \\ y^2 \\ y^3 \\ ... \\ y^m \end{bmatrix}$$

En consecuencia, la i-ésima fila del vector $x^i = [x_0^i, x_1^i, ..., x_n^i]$, representa la i-ésima muestra de entrada, mientras que el j-ésimo vector de columna x_j indica todas las muestras recopiladas para esta característica específica.

2 Minimizando la función de costo

Una regresión lineal se puede declarar de manera equivalente como un problema de optimización en el que se debe minimizar una función de costo o una función objetivo. La elección directa para la función de costo es la suma cuadrática de todos los residuos de la salida y el valor previsto correspondiente como la siguiente expresión,

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2 \tag{4}$$

donde y_i es el i-ésimo ejemplo objetivo y $h_\theta(x^i)$ el i-ésimo valor predicho. La ecuación (4) es solo una medida de error general que captura la desviación general en el par de la predicción-objetivo.

La solución óptima está representada por el conjunto de parámetros $\theta_0, \theta_1, \theta_2, ..., \theta_n$ que minimiza la expresión (4). En términos formales, el problema de optimización a resolver es por lo tanto el siguiente:

$$\min_{\theta \in R^{n+1}} \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \tag{5}$$

La condición más general para la convergencia de algoritmos al mínimo es que $J(\theta)$ debe ser una función convexa.

2.1 Algoritmo del Descenso del Gradiente

El algoritmo de descenso de gradiente opera de manera iterativa, y en cada paso, una solución que está (en teoría) mcerca del mínimo global. La idea basicamente es: actualizar la solución anterior con la función de costo correspondiente frente a la tasa más alta:

$$\theta_{nueva} = \theta_{vieja} - \alpha \nabla [J(\theta_{vieja})] \tag{6}$$

donde el escalar α se conoce como la tasa de aprendizaje. La razón para adoptar un escalar (generalmente diferente de cero) es que la velocidad de aprendizaje es capaz de modular el desplazamiento o "saltos" permitiendo una convergencia más rápida (o más lenta) al mínimo global. Suponiendo que el modelo de regresión lineal general y la función de costo definida anteriormente, los componentes de la solución se actualizan específicamente a través de (6) como:

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)} \tag{7}$$

donde, el símbolo := denota la operación de asignación comúnmente utilizada para los algoritmos.

El algoritmo siguiente describe el procedimiento completo para encontrar una solución a una regresión lineal múltiple.

3 Método de la Ecuación Normal

Una solución analítica para el problema de regresión lineal multivariable es la ecuación normal que puede obtenerse explícitamente estableciendo los derivados de la función de costo con respecto a θ_j a cero, o usando argumentos de álgebra lineal. El siguiente resultado proporciona el valor óptimo teórico,

$$\theta_{min} = (X^T X)^{-1} X^T y \tag{8}$$

donde una vez más, $X \in R^{m \cdot (n+1)}, y \in R^m$ y $\theta_{min} \in R^{n+1}$. Aunque la aparente facilidad de aplicar la Ecuación Normal para encontrar los valores de los parámetros, es importante observar que este método también puede ser desventajoso en algunas situaciones. El problema principal aquí es cuando el número de características n es suficientemente grande, por lo que calcular el término de matriz inversa $(X^T X)^{-1}$ puede ser lento o incluso poco práctico.

4 Implementación en MATLAB

La siguiente implementación en MATLAB describe el procedimiento para encontrar una solución a la regresión lineal con multiples variables.

4.1 computeCostMulti.m

Calcula el costo de usar θ como el parámetro de regresión lineal para ajustar los puntos de datos en X y y. En este código notaremos lo siguiente:

Para resolver el calculo del costo con multiples variables la instrucción que debemos de poner es:

$$J = \frac{1}{2 \cdot m}(X \cdot \theta - y)^T \cdot (\theta \cdot X - y) \tag{9}$$

```
function J = computeCostMulti(X, y, theta)

% Initialize some useful values
        m = length(y); % number of training examples

% You need to return the following variables correctly
        J = 0;

% Esto es matricialmente:
        J = (1/(2 * m) * (X * theta - y)' * (X * theta - y)
            ↪ );
% =========================================
end
```

4.2 featureNormalize.m

Esta función devuelve una versión normalizada de X donde el valor medio de cada característica es θ y la desviación estándar es 1.

Este suele ser un buen paso de preprocesamiento cuando se trabaja con algoritmos de aprendizaje.

Como resolverlo:

1. Primero, para cada dimensión de la característica, calcule la media de la característica y réstela del conjunto de datos, almacenando el valor medio en mu.

2. Calculamos la desviación estándar de cada característica y divida cada característica por su desviación estándar, almacenando la desviación estándar en σ.

3. Debemos de tener en cuenta que X es una matriz donde cada columna es una entidad y cada fila es un ejemplo.

La variable μ es el promedio ó media de los datos que se calcula mediante:

$$X_n^i = \frac{X^i - \mu}{\sigma} \tag{10}$$

donde σ esta dado por $(max - min)$.

```
function [X_norm, mu, sigma] = featureNormalize(X)
               X_norm = X;
               mu = zeros(1, size(X, 2));
               sigma = zeros(1, size(X, 2));

               mu = mean(X);

               sigma = std(X);

               for i = 1:size(X, 2)
               Xmu = X(:, i) - mu(i);
               X_norm(:, i) = Xmu / sigma(i);
               end
end
```

4.3 normalEqn.m

Calcula la solución de forma cerrada a la regresión lineal usando las ecuaciones normales. Visto matemáticamente la solución de este problema es:

$$\theta = (X^T \cdot X)^{-1} \cdot (X^T \cdot y) \tag{11}$$

```
function [theta] = normalEqn(X, y)

       theta = zeros(size(X, 2), 1);

       % --------- Sample Solution -----------

       theta = inv(X'*X) * X' * y;
end
```

4.4 gradientDescentMulti.m (Algoritmo del Descenso más Pronunciado)

Obtenemos matricialmente la matriz θ tal que $J(\theta)$ sea mínimo, esto es:

$$\begin{pmatrix} \\ \theta_0 \\ \theta_1 \\ \theta_2 \\ \dots \\ \theta_n \end{pmatrix} = \frac{\alpha}{m}(X)^T \cdot (X \cdot \theta - y) \tag{12}$$

```
function [theta, J_history] = gradientDescentMulti(X, y, theta,
    ↪ alpha, num_iters)

       % Initialize some useful values
       m = length(y); % number of training examples
       J_history = zeros(num_iters, 1);

       for iter = 1:num_iters

       theta = theta - (alpha * (1/m) * X' * (X * theta - y));

       % ================================

       % Save the cost J in every iteration
       J_history(iter) = computeCostMulti(X, y, theta);
       end
end
```

5 Simulación en MATLAB

```
================ Feature Normalization ================
Loading data ...
First 10 examples from the dataset:
 x = [2104 3], y = 399900
 x = [1600 3], y = 329900
 x = [2400 3], y = 369000
 x = [1416 2], y = 232000
 x = [3000 4], y = 539900
 x = [1985 4], y = 299900
 x = [1534 3], y = 314900
 x = [1427 3], y = 198999
 x = [1380 3], y = 212000
 x = [1494 3], y = 242500
Program paused. Press enter to continue.
Normalizing Features ...
```

A este punto, se normalizan las caracteristicas (features) y comenzamos a calcular el descenso del gradiente, en el proceso de ejecución se nos pedira ingresar los valores tanto de α como el num_{iter}, estos valores respectivamente son de 0.01 y 500:

```
================ Gradient Descent ================
Running gradient descent ...
Give me the value of Alpha: 0.01
Give me the Number Of Iterations: 500

Theta computed from gradient descent:
 338175.983967
 103831.117371
 103.030733

Predicted price of a 1650 sq-ft, 3 br house (using gradient
    ↪ descent):
 $292335.049065
Program paused. Press enter to continue.
```

El algoritmo de Descenso del Gradiente calcula las θ optimas y predice el precio utilizando estos valores de θ.

Ahora lo comparamos con el método de las ecuaciones normales.

```
================ Normal Equations ================
Solving with normal equations...
Theta computed from the normal equations:
 89597.909543
 139.210674
 -8738.019112


=================== Predicted Price ====================
Predicted price of a 1650 sq-ft, 3 br house (using normal
    ↪ equations):
 $293081.464335
```

El grafico que nos entregan es el siguiente:

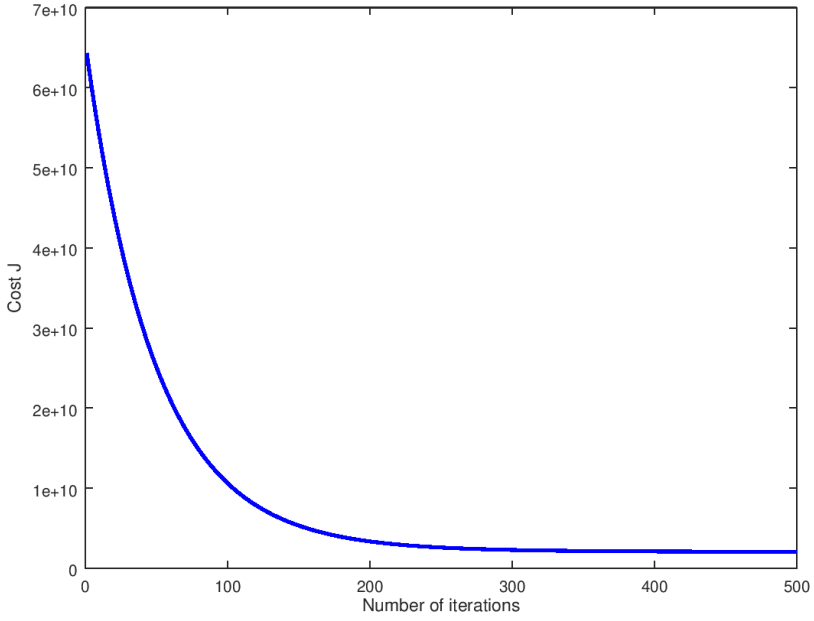

Figure 1: computeCostMulti con valores $\alpha = 0.01$ y $num_{iters} = 500$