Universidad de Huelva

Universidad de Huelva

Grado en Ingeniería Informática

Tema 2. Regresión Lineal

Resumen

Autor: Alberto Fernández Merchán
Asignatura: Aprendizaje Automático

1. Introducción

En esta asignatura nos centraremos en el **aprendizaje numérico**. En este tipo de aprendizaje, tanto la **entrada** como la **salida** son **valores numéricos**.

Existen dos tipos de aprendizaje numérico: la **regresión** y la **clasificación**. En este tema trataremos el método de regresión como **regresión lineal**, sin embargo, también existe regresión multisalida donde la salida es un vector de valores. Aunque el vector de datos de entrada ($\bar{x}$) pueda ser multidimensional, la regresión lineal genera un dato unidimensional como salida (y).

Un problema de aprendizaje numérico se puede enunciar como:

"dado un conjunto de datos de entrenamiento, debemos encontrar la función $h(x) = y'$, tal que $h(x)$ se ajuste lo máximo posible a los valores de y"

A la función $h(x)$ se le denomina **hipótesis**. Para averiguar dicha función, utilizaremos la **teoría lineal**

2. Teoría Lineal

2.1. Notación

- $m \equiv$ número de ejemplos de entrenamiento.
- $n \equiv$ número de atributos.
- $(X, y) \equiv$ ejemplo de entrenamiento.
- $(X^j, y^j) \equiv$ ejemplo de entrenamiento de la fila j.
- $X \equiv$ variable de entrada, característica, atributo...
- $y \equiv$ variable de salida, objetivo, variable independiente...

2.2. Minimización del Error

Para determinar la hipótesis, primero tendremos que suponer que la función será una **función lineal**. En general, para un dataset de n atributos, podemos escribir: $h(\bar{X}) = \sum_{i=0}^{n} \theta_i * x_i = \theta^T * \bar{X} \quad , (x_0 = 1)$. Tenemos que añadir un atributo más (x_0) que valdrá siempre uno para poder generalizar la fórmula.

Para encontrar la función de hipótesis, debemos determinar los coeficientes θ de forma que se minimice el error (ϵ). Para ello utilizaremos el concepto de derivada.

Podemos considerar diferentes definiciones del error:

- $\epsilon = Prediccion - ValorReal = h(x) - y(x)$: Es una mala definición para aplicarle el algoritmo Hill-Climbing.
- $\epsilon = |Prediccion - ValorReal| = |h(x) - y(x)|$: También es una mala definición para establecer el error porque no es derivable en todos sus puntos.
- $R = \frac{1}{2} \cdot (h(x) - y(x))^2$. Esta es una buena definición ya que nos permite utilizar el algoritmo de Hill Climbing.

Para obtener el error total, sumamos todos los ejemplos:

$$ErrorCuadraticoTotal = \frac{1}{2} \sum_{j=1}^{m} (h_\theta(x^j) - y^j)^2$$

Nuestro objetivo es **minimizar** el error total:

$$min_\theta(\frac{1}{2}\sum_{j=1}^{m}(h_\theta(x^j)-y^j)^2)$$

Podemos llamar a la expresión del error total $J(\theta)$, de esta forma tendremos que **buscar los valores de θ que minimicen $J(\theta)$.**

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

Buscaremos que el error se comporte de forma que describa una campana de Gauss:

- Gráfica Simétrica
- Media = Mediana = Centro

Utilizaremos el método de **DESCENSO POR GRADIENTE**.

2.3. Descenso por Gradiente

La idea de este algoritmo es:

1. Comenzamos inicializando los valores de θ con algún valor aleatorio, por ejemplo $\vec{\theta}=\vec{0}$.
2. Cambiaremos los valores de θ de forma que disminuya $J(\theta)$. Deberemos **asegurarnos** de que se disminuya J.
3. Repetimos el cambio de valores hasta alcanzar el mínimo (que ya no disminuya más el valor).

Para variar los coeficientes deberemos actualizarnos en la dirección del **máximo decremento**. Para saber dicha dirección utilizaremos el concepto de gradiente ($\vec{\nabla}_\theta J(\theta)$).

Usaremos la siguiente expresión para actualizar los coeficientes:

$$\theta_{nuevo}=\theta_{antiguo}+\alpha\vec{\nabla}_\theta J(\theta)$$

Donde α (Coeficiente de aprendizaje): modula la velocidad de acercamiento. Al minimizar utilizaremos un coeficiente negativo ($\alpha<0$).

Por definición, el gradiente es el vector de todas las derivadas parciales de J en las direcciones de θ.

$$\vec{\nabla}_\theta J(\theta)=(\frac{\partial J(\theta)}{\partial\theta_0},\frac{\partial J(\theta)}{\partial\theta_1},...,\frac{\partial J(\theta)}{\partial\theta_n})$$

A continuación, desarrollaremos la fórmula del gradiente para un dataset de 1 ejemplo para la componente iésima:

$$\frac{\partial}{\partial\theta_i}J(\theta)=\frac{\partial}{\partial\theta_i}(\frac{1}{2}(h_\theta(x)-y)^2)=(h_\theta(x)-y)\cdot\frac{\partial}{\partial\theta_i}(h_\theta(x)-y)=(h_\theta(x)-y)\cdot\frac{\partial}{\partial\theta_i}(\theta_0x_0+\theta_1x_1+...+\theta_nx_n-y)=$$

Calculamos la derivada: $\frac{\partial}{\partial\theta_i}(\theta_0x_0+\theta_1x_1+...+\theta_ix_i+...+\theta_nx_n-y)=(0+0+...+\theta_ix_i+...+0-0)=x_i$

La fórmula del gradiente quedaría tal que:

$$\frac{\partial}{\partial\theta_i}J(\theta)=(h_\theta(x)-y)x_i$$

La actualización de los parámetros para **m** elementos quedaría...

$$\theta_i'=\theta_i+\alpha\sum_{j=1}^{m}(h_\theta(x^j)-y^j)x_i^j$$

El descenso por gradiente funciona mejor cuando todos los datos se encuentran en una escala similar, por tanto, es necesario **normalizar** los datos.
Algunas características del descenso por gradiente son:

- La muestra debe ser normal, aleatoria y con varianzas homogéneas.
- La regresión no prueba causalidad.
- No debe extenderse más allá de los datos obtenidos.

El modelo de regresión parte de algunas presunciones acerca de las variables y de las muestras experimentales:

- La media de la población Y a un valor dado de X crece o decrece linealmente a medida que X aumenta.
- Para cada valor de X, los valores de Y se distribuyen normalmente.
- La desviación típica de la población de Y alrededor de la media de Y a un valor de X, es la misma para cada valor de X.

2.3.1. Algoritmo

```
theta[i] = init()
repeat until convergence {
   for i = 0 to n:
       theta[i] = Actualizacion[i]
}
```

Si la gráfica del error crece en algún punto, quiere decir que α es demasiado grande. La gráfica del error debe ser decreciente en todos sus puntos.

2.3.2. Características

- Este algoritmo procesa todos los ejemplos por cada nueva asignación de parámetro.
- No es aconsejable en *datasets* grandes porque realiza mxn actualizaciones.
- Para evitar mínimos locales se inicializa a valores aleatorios.

2.4. Descenso de Gradiente Estocástico

2.4.1. Características

- Actualiza los parámetros para cada uno de los ejemplos y no al final.
- El ejemplo es elegido al azar.
- Si el número de ejemplos es mucho mayor al de atributos, la solución converge más rápido.

2.5. Comparación

La comparación de ambos métodos es la siguiente:

Num.	Descenso por Gradiente	Descenso Estocástico
1.	Calcula el gradiente usando toda la muestra de entrenamiento	Calcula el gradiente usando una sola muestra de entrenamiento
2.	Algoritmo lento y computacionalmente costoso	Más rápido y menos computacionalmente costoso que Batch GD
3.	No sugerido para muestras de entrenamiento grandes.	Puede utilizarse para grandes muestras de entrenamiento.
4.	De naturaleza determinista.	De naturaleza estocástica.
5.	Proporciona una solución óptima con el tiempo suficiente para converger.	Da una buena solución pero no óptima.
6.	No se requiere una mezcla aleatoria de puntos.	La muestra de datos debe estar en un orden aleatorio, y es por eso que queremos mezclar el conjunto de entrenamiento para cada *epoch*.
7.	No se puede escapar fácilmente de los mínimos locales poco profundos.	puede escapar de los mínimos locales poco profundos más fácilmente.
8.	La convergencia es lenta.	Alcanza la convergencia mucho más rápido.

3. Ecuaciones Normales

Es una alternativa analítica al método iterativo. Consiste en buscar, de forma general, la resolución de la ecuación de minimización, sin embargo, no se puede aplicar en todas las ocasiones ya que, en algunos problemas, no existe la inversa de la matriz.

3.1. Fórmula General del Gradiente

$$\theta = \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}; \qquad \nabla J(\theta) = \begin{bmatrix} \frac{\partial J}{\partial \theta_0} \\ \vdots \\ \frac{\partial J}{\partial \theta_n} \end{bmatrix} \in \mathbb{R}^{n+1}$$

Donde $\mathbb{R}^{n+1}$ es el número de elementos que tiene cada vector teniendo en cuenta que hemos añadido un elemento más que vale 1 (x_0).

$$h(\theta) = X \cdot \theta = \begin{bmatrix} x_0^1 & \cdots & x_i^1 & \cdots & x_n^1 \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ x_0^m & \cdots & x_i^m & \cdots & x_n^m \end{bmatrix} \cdot \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_n \end{bmatrix} = \begin{bmatrix} x^1 \cdot \theta_0 \\ \vdots \\ x^m \cdot \theta_n \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{n} \theta_i^1 x_i^1 \\ \vdots \\ \sum_{i=0}^{n} \theta_i^m x_i^m \end{bmatrix} = \begin{bmatrix} h_\theta(x^1) \\ \vdots \\ h_\theta(x^m) \end{bmatrix} \in \mathbb{R}^m$$

Le añadimos la diferencia con la y:

$$X \cdot \theta - y = \begin{bmatrix} h(x^1) - y^1 \\ \vdots \\ h(x^m) - y^m \end{bmatrix}$$

Podemos transformar el cuadrado del error cuadrático en la multiplicación de la matriz por su traspuesta:

$$J(\theta) = \frac{1}{2}\sum_{j=1}^{m}(h(x^j) - y^j)^2 = \frac{1}{2}(h_\theta(x) - y)^t(h_\theta(x) - y)$$

$$J(\theta) = \tfrac{1}{2}(X\theta - y)^t(X\theta - y)$$

A continuación debemos obtener el mínimo de $J(\theta)$. Para ello se debe cumplir que $\nabla_\theta J(\theta) = 0$.

$$\nabla_\theta(\frac{1}{2}(X\theta - y)^t(X\theta - y)) = 0 \quad \rightarrow \quad \frac{1}{2}\nabla_\theta((X\theta - y)^t(X\theta - y)) = 0 \quad \rightarrow$$

Después de realizar las multiplicaciones se obtiene que:

$$\nabla_\theta J(\theta) = X^tX\theta - X^ty = 0 \quad \rightarrow \quad X^tX_\theta = X^ty \text{ (ec. normal)}$$

Si despejamos θ obtenemos su definición:

$$\theta = (\mathbf{X^tX})^{-1}\mathbf{X^ty} \quad \text{(En algunas ocasiones } X^tX \text{ no es invertible.)}$$

Con este método **NO** se necesita un método iterativo, ya que se actualiza con una simple orden.
Siempre hay un punto donde el error tendrá un mínimo. Aunque dicho punto exista, **NO SIEMPRE** se puede calcular con las ecuaciones normales