UNIVERSIDAD
DE GRANADA

Aprendizaje Automático. Búsqueda Iterativa de Óptimos y Regresión Lineal

Gradiente Descendente, Mínimos Cuadrados,

Ricardo Ruiz Fernández de Alba

Escuela Técnica Ingeniería Informática y Matemáticas

DECSAI

Universidad de Granada

31 de marzo de 2022

Índice general

1

Ejercicio sobre la búsqueda iterativa de óptimos

1.1 | Descenso de gradiente

El algoritmo de descenso de gradiente es una técnica general para minimizar funciones diferenciables. Dependiendo del punto de inicio, se puede llegar a un mínimo local o global. Si la función es convexa, puede converger al mínimo global.

1.2 | Formalización Matemática

Sea $w(0) \in \mathbb{R}^d$ un punto inicial, y $E : \mathbb{R}^d \longrightarrow \mathbb{R}$ una función diferenciable. Sea $\eta \in \mathbb{R}^+$. La actualización del peso en la t-ésima iteración viene definida por

$$w(t+1) = w(t) - \eta \nabla E_{in}(w(t))$$

donde al parámetro η se le conoce como tasa de aprendizaje. El algoritmo se basa en seguir la dirección opuesta al vector gradiente pues se maximiza el decrecimiento de la función en cada iteración. La convergencia no está garantizada en general y dependerá de la tasa de aprendizaje escogida así como de las características de la función.

Formalmente, esto se deduce de la expansión de Taylor de primer orden de la función: **?**

$$\begin{aligned} \Delta E_{in} &= E_{in}(w(0) + \eta\hat{v}) - E_{in}(w(0)) \\ &= \eta \nabla E_{in}(w(0))^T \hat{v} + O(\eta^2) \\ &=\geq -\eta \|\nabla E_{in}(w(0))\| \end{aligned} \tag{1.1}$$

con $\hat{v}$ vector unitario. La igualdad se da si y sólo si

$$\hat{v} = -\frac{\nabla E_{in}(w(0))}{\|\nabla E_{in}(w(0))\|}$$

Una tasa de aprendizaje (demasiado) pequeña es ineficiente lejos del mínimo local. Por otro lado, si es demasiado grande, las oscilaciones pueden afectan a la convergencia. Por ello, optamos por una tasa variable proporcional a la norma del gradiente.

$$\eta_t = \eta \|\nabla E_{in}\|$$

De esta manera, se realizan grandes pasos lejos del mínimo y pasos de menor tamaño cerca del mínimo (donde la norma del gradiente es menor). Además, η_t cancela con el denominador de $\hat{v}$ definido anteriormente lo que se sintetiza redefiniendo $\eta = \eta_t$ y hallando el opuesto del vector gradiente pero sin normalizar.

Como la convergencia no está garantizada en tiempo finito, necesitamos una condición de parada como que la función a minimizar quede por debajo de cierto umbral de error (i.e $E_{in} \leq \epsilon$) o simplemente un número máximo de iteraciones. Combinamos ambas en la implementación ofrecida an el archivo de código:

1.3 | Ejercicio 1

Implementar el algoritmo de gradiente descendente

Analizamos la definición de la función descrita en el código:

```
def gradient_descent(w_ini, lr, grad_fun, fun, epsilon, max_iters,
                     stop_cond,  hist=False):
```

Sobre la cabecera:

- `w_ini`: vector de pesos inicial
- `grad_fun`: gradiente de la función a minimizar
- `fun`: función a minimizar
- `epsilon`: cota de error para parar
- `max_iters`: número máximo de iteraciones para parar

- `stop_cond(it, w, epsilon, max_iters, grad_fun, fun)` : función que indica cuando parar el algoritmo

  `stop_cond_maxIter`: para cuando `it <= max_iters`

  `stop_cond_error`: para cuando `fun(w[0], w[1]) \leq epsilon`

- `hist`: True si se desea devolver el histórico para visualización

1.4 | Ejercicio 2.

Considerar la función $E(u,v) = \left(uv \cdot e^{-u^2-v^2}\right)^2$ Usar gradiente descendente para encontrar un mínimo de esta función, comenzando desde el punto $(u,v) = (0{,}5, -0{,}5)$ y usando una tasa de aprendizaje $\eta = 0{,}1$.

1.4.1 | Cálculo analítico de $\nabla E(u,v)$

Hallamos la primera derivada parcial:

$$\begin{aligned} \frac{\partial E}{\partial u}(u,v) = 2uve^{-u^2-v^2} \cdot v \left[e^{-u^2-v^2} + u \cdot e^{-u^2-v^2} \cdot 2(-u)\right] = \\ = 2uv^2e^{2(-u^2-v^2)} - 4u^3v^2e^{2(-u^2-v^2)} = -2u(2u^2-1)v^2e^{-2(u^2+v^2)} \end{aligned} \tag{1.2}$$

Y análogamente, $\frac{\partial E}{\partial u}(u,v) = -2v(2v^2-1)u^2e^{-2(u^2+v^2)}$. Luego,

$$\nabla E(u,v) = \begin{pmatrix} \frac{\partial E}{\partial u}(u,v) \\ \frac{\partial E}{\partial v}(u,v) \end{pmatrix} = \begin{pmatrix} -2u(2u^2-1)v^2e^{-2(u^2+v^2)} \\ -2v(2v^2-1)u^2e^{-2(u^2+v^2)} \end{pmatrix} \tag{1.3}$$

1.4.2 | Número de iteraciones con cota de error 10^{-8}

1.4.3 | Coordenadas obtenidas con cota de error 10^{-8}

Ejecutamos el algoritmo anteriormente descrito con los siguiente parámetros:

```
eta = 0.1
error2get = 1e-8
initial_point = np.array([0.5,-0.5])

w, it = gradient_descent(initial_point, eta, gradE, E,
                         error2get, maxIter, stop_cond_error)
```

La condición de parada fijada es que la función de error alcance un valor menor o igual a 10^{-8}, que se indica en `stop_cond_error`

- Numero de iteraciones: 25117
- Coordenadas obtenidas: $(0{,}0100, -0{,}0100)$

En la siguiente figura visualizamos el desplazamiento hacia el mínimo:

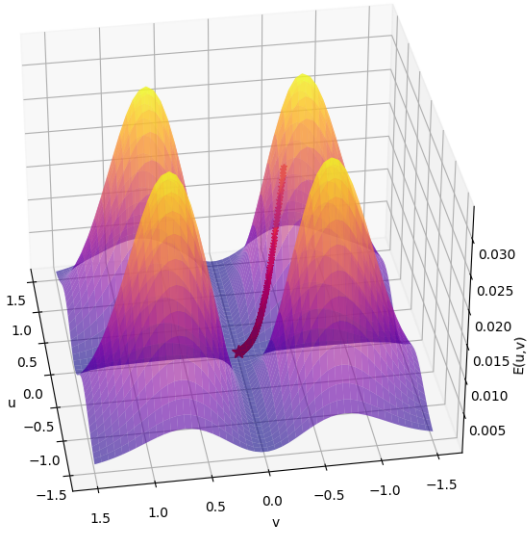

Figura 1.1: Descenso de gradiente sobre E

1.5 | Ejercicio 3

Consideramos la función $f(x,y) = x^2 + 2y^2 + 2\sin(2\pi x)\sin(\pi y)$. **Aplicar gradiente descendente para minimizar** f

1.5.1 | Dependencia de la tasa de aprendizaje

Usar como punto inicial $(x_0 = -1, y_0 = 1)$, tasa de aprendizaje $\eta = 0{,}01$ y un máximo de 50 iteraciones. Generar un gráfico de cómo desciende el valor de la función con las iteraciones. Repetir el experimento pero usando $\eta = 0{,}1$, comentar las diferencias y su dependencia de η.

En primer lugar, calculamos analíticamente el gradiente de f.

$$\nabla f(x,y) = \begin{pmatrix} \frac{\partial}{\partial x} f(x,y) \\ \frac{\partial}{\partial y} f(x,y) \end{pmatrix} = \begin{pmatrix} 2x + 4\pi sen(\pi y) cos(2\pi x) \\ 4y + 2\pi sen(\pi x) cos(\pi y) \end{pmatrix} \tag{1.4}$$

En este caso la condición de parada es el máximo de iteraciones, por lo que que ejecutamos el algoritmo con la función de condición de parada `stop_cond_maxIter`

```
eta = 0.01
maxIter = 50
initial_point = np.array([-1, 1])

ws, it = gradient_descent(initial_point, eta, gradf, f, None, maxIter,
                          stop_cond_maxIter, hist=True)
```

Así, para $\eta = 0{,}01$ y un máximo de 50 iteraciones, las coordinadas finales son $(-1{,}2178, 0{,}4134)$ donde f toma el valor $-0{,}0623$

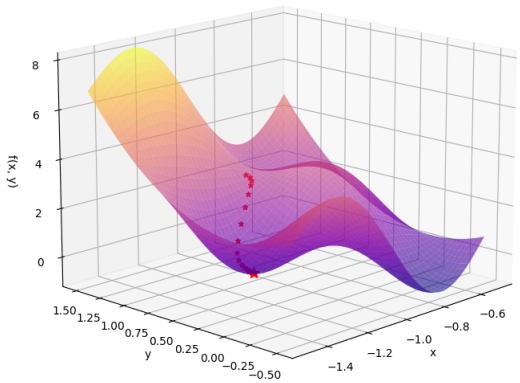

Figura 1.2: Descenso del gradiente sobre f con $\eta = 0{,}01$

De igual manera para $\eta = 0{,}1$ las coordenadas finales obtenidas son $(-0{,}1155, 0{,}1610)$ y visualizamos el descenso en la siguiente figura:

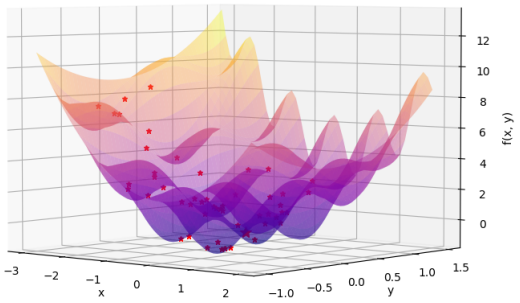

Figura 1.3: Descenso del gradiente sobre f con $\eta = 0{,}1$

Como se observa en la figura, la tasa de aprendizaje es demasiado alta, provocando oscilaciones que saltan el mínimo. De hecho, una inspección del vector de puntos `ws` revela que en la 43-ésima iteración alcanza el mínimo valor de las 50 iteraciones, pero que luego sigue oscilando y termina en la última iteración con un valor mayor.

La comparación entre las oscilaciones con $\eta = 0{,}1$ y el descenso adecuado con $\eta = 0{,}01$ quedan reflejadas en la siguiente figura.

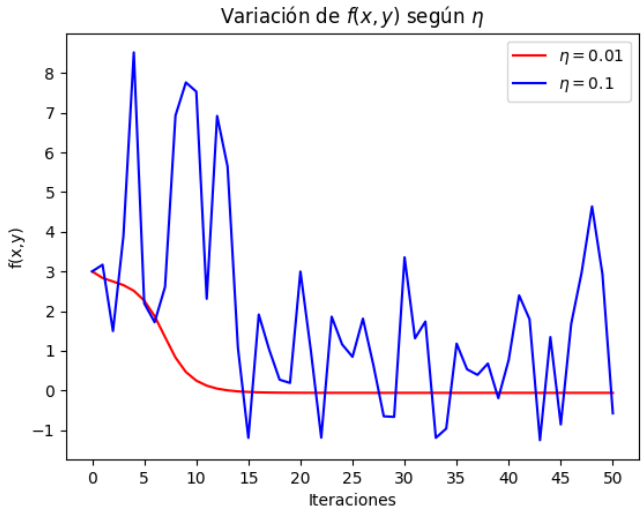

Figura 1.4: Comparación de $f(x, y)$ para $\eta = 0{,}01$ y $\eta = 0{,}1$

1.5.2 | Tabla de mínimos según punto inicial y tasa de aprendizaje

Obtener el valor mínimo y los valores de las variables (x, y) en donde se alcanzan cuando el punto de inicio se fija en: $(-0{,}5, -0{,}5)$, $(1, 1)$, $(2{,}1, -2{,}1)$, $(-3, 3)$, $(-2, 2)$. Generar

una tabla con los valores obtenidos, empleando el máximo número de iteraciones (50) y las tasas de aprendizaje del anterior apartado ($\eta = 0{,}01$ y $\eta = 0{,}1$). Comentar la dependencia del punto inicial.

Cuadro 1.1: Valor mínimo para $\eta = 0{,}1$

Punto inicial	Coordenadas mínimo	Valor mínimo f(x,y)
(-0.5, -0.5)	(0.2848,-0.5553)	-1.2250
(1, 1)	(0.3018,-0.4257)	-1.3902
(2.1, -2.1)	(-0.2320,0.2831)	-1.3293
(-3, 3)	(0.1816,-0.3152)	-1.2883
(-2, 2)	(0.2428,-0.3121)	-1.4061

Cuadro 1.2: Valor mínimo para $\eta = 0{,}01$

Punto inicial	Coordenadas mínimo	Valor mínimo f(x,y)
(-0.5, -0.5)	(-0.7308,-0.4144)	-1.0366
(1, 1)	(0.7308,0.4144)	-1.0366
(2.1, -2.1)	(1.6651,-1.1728)	4.6338
(-3, 3)	(-2.1888,0.5868)	3.6942
(-2, 2)	(-1.6643,1.1713)	4.6337

Es claro a partir de estos datos que el algoritmo de gradiente descendiente tiende hacia un mínimo local que dependerá del punto inicial. En general, en este caso, la tasa de aprendizaje mayor $\eta = 0{,}1$ proporciona un valor mínimo más regular (en torno a $-1{,}25$) que la tasa $\eta = 0{,}01$.

1.6 | Ejercicio 4

¿Cuál sería su conclusión sobre la verdadera dificultad de encontrar el mínimo global de una función arbitraria?

A partir del experimento anterior, confirmamos empíricamente que el mínimo local que encuentra el algoritmo de gradiente descendiente depende fuertemente tanto del punto inicial como de la tasa de aprendizaje. No podemos garantizar que este mínimo sea global a no ser que la función sea convexa. Esto último sucede en los casos de la función de error cuadrático medio de la regresión lineal así como en el error de entropía cruzada de la regresión logística. Esto significa que al minimizar estas medidas de error convexas con este algoritmo, no se estancará en un mínimo local.

2

Regresión Lineal

2.1 | Algoritmo de la Pseudo-inversa

El algoritmo de la Pseudo-inversa o mínimos cuadrados ordinarios consiste en minimizar el error cuadrático entre la estimación $h(x) = w^T x$ y el vector objetivo y

$$E_{in}(h) = \frac{1}{N}\sum_{n=1}^{N}(h(x_n) - y_n)^2 = \frac{1}{N}\|Xw - y\|^2 \tag{2.1}$$

Basta encontrar w que anule $\nabla E_{in}(w)$. Esto se verifica cuando $w_{lin} = X^\dagger y$ **?**. Donde $X^\dagger = (X^T X)^{-1} X^T$ es la **matriz pseudo-inversa** de X

Como el cálculo de la inversa $(X^T X)^{-1}$ puede ser costoso, aplicamos previamente la descomposición en valores singulares a X. Es decir, escribimos $X = UDV^T$ con $U \in \mathbb{R}^{N\times N}$, $V \in \mathbb{R}^{(d+1)\times(d+1)}$ matrices ortogonales y $D \in \mathbb{R}^{N\times(d+1)}$ matriz diagonal rectangular. Entonces $X^T X = VD^T DV^T$ y la pseudo-inversa se calculará como sigue:

$$\begin{aligned} X^\dagger = (X^T X)^{-1} X^T = (VD^T DV^T)^{-1} VD^T U^T = \\ = V(D^T D)^{-1} V^T VD^T U^T = V(D^T D)^{-1} D^T U^T = VD^\dagger U^T \end{aligned} \tag{2.2}$$

2.2 | Gradiente Descendente Estocástico

El algoritmo de gradiente descendente estocástico (SGD) es una versión secuencial del algoritmo de gradiente descendente descrito en el primer capítulo. En esta versión, usamos sólo una parte de la muestra para calcular el gradiente. Para ello, dividimos la muestra aleatoriamente en **mini-batches**. Recorreremos esta secuencia de lotes, actualizando los pesos en cada mini-batch de manera que sólo uno participa en cada actualización.

Formalmente, para cada mini-batch B_i

$$\frac{\partial E_{in}(w)}{\partial w_j} = \frac{2}{M} \sum_{n \in B_i} x_{nj}(h(x_n) - y_n)$$

2.3 | Ejercicio 1

Este ejercicio ajusta modelos de regresión a vectores de características extraídos a partir de imágenes de dígitos manuscritos. En particular, se extraen dos características concretas que miden el valor medio del nivel de gris y la simetría del dígito respecto de su eje vertical. Solo se seleccionaran para este ejercicio las imágenes de los números 1 y 5.

Estimar un modelo de regresión lineal, a partir de los datos proporcionados por los vectores de características dados, usando tanto el algoritmo de la pseudo-inversa como el gradiente descendente estocástico (SGD). Las etiquetas serán -1,1, una por cada vector de cada uno de los números. Pintar las soluciones obtenidas junto con los datos usados en el ajuste. Valorar la bondad del resultado usando E_{in} y E_{out} (para E_{out} calcular las predicciones usando los datos del fichero de test).

3

Método de Newton

This section should include a recipe of what you did (explain what you have done so if someone wants to reproduce the experiment, they can). A flow chart is typically helpful. Also, make sure to define all software that you used including version numbers and OS. Should also include a description of statistical methods used (if any).[1]

[1] For more information see: `http://rc.rcjournal.com/content/49/10/1229.short`

Bibliografía

Hsuan-Tien Lin Yaser S. Abu-Mostafa, Malik Magdon-Ismail. *Learing From Data. A Short Course*. AMLbook, 2012. URL `AMLbook.com`.