Optimización Continua y Aplicaciones (OCA)

(Obligatorio 3)

Descenso de gradiente estocástico, Regresión Lineal

Claudio Risso, Pablo Rodríguez Bocca

15 de septiembre de 2021

1) Mínimo local en una función.

Comenzaremos con un ejemplo sencillo de aplicación del algoritmo "Descenso por gradiente (GD)", basado en el trabajo de Daniel Newman (`https://www.dtnewman.com/`, idioma inglés). Buscaremos el mínimo local de la función $f(x) = x^3 - 2x^2 + 2$ cuando $0 \leq x \leq 2.5$.

Implementar la solución en `Octave/Matlab`.

```
function res = f(x)
  res = x.^3 -2*x.^2 + 2;
endfunction
```

A) Utilizando cálculo analítico encontrar el mínimo local x^* de la función $f()$.

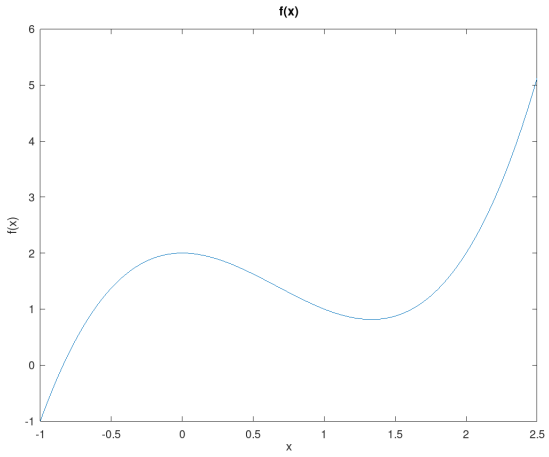

Figura 1: La función $f(x)$.

Supongamos para las siguientes partes que no conocemos el óptimo local, partiendo de $x_0 = 2$.

B) **Descenso por gradiente.** Aplicar el algortimo simple de descenso por gradiente para encontrar el óptimo local de $f(x)$. Utilizar un paso fijo (learning rate) de $\eta = 0.17$, y como criterio de parada utilizar solamente que el salto del paso sea mayor a una precisión de 0.0001. Este algoritmo de descenso puede resumirse en el siguiente pseudo-código:

Algorithm 1 Descenso por gradiente

Require: punto de partida x_0, learning rate η, precision p

$x_{n+1} \leftarrow x_0$

repeat

$x_n \leftarrow x_{n+1}$

$x_{n+1} \leftarrow x_n - \eta \nabla f(x_n)$

until $|x_n - x_{n-1}| > p$

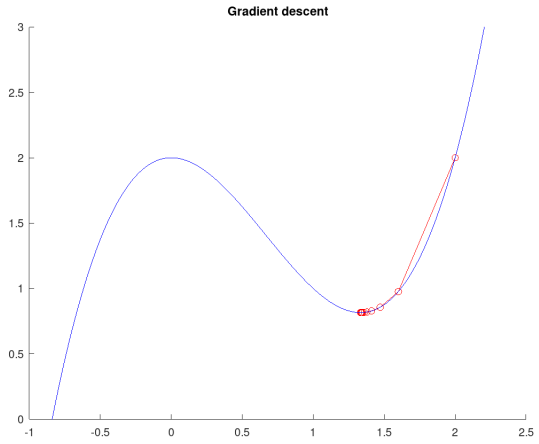

Figura 2: Aplicación de GD, luego de 17 pasos se encuentra un valor muy cercano al óptimo.

¿Qué sucede si se repite el cálculo cambiando $\eta = 0.0001$? ¿y cuándo $\eta = 0.6$?

C) **Descenso de gradiente, con paso óptimo.**

Como sabemos, el learning rate η es muy importante y determina en gran parte la convergencia del algoritmo GD. Si el paso es muy pequeño, entonces la convergencia es muy lenta; en cambio si el paso es muy grande el algoritmo puede diverger. En esta sección implementaremos un paso adaptativo óptimo, utilizando como ayuda la función `fmincon` del paquete `optim` de `Octave`. Mantener $x_0 = 2$, y una precisión de 0.0001 como criterio de parada.

Verificar que en muy pocos pasos se encuentra una solución, pero el tiempo de cómputo de cada paso aumenta considerablemente.

D) **Descenso de gradiente, con paso variable simple.**

Para evitar tener que resolver un problema de optimzación en cada paso, es posible definir otras formas más sencillas de paso variable. Por ejemplo considerar que el paso se decrementa en forma

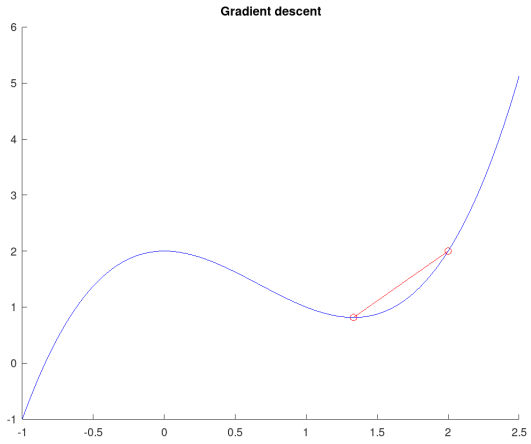

Figura 3: Aplicación de GD con paso óptimo. En 3 pasos se encuentra unas muy buena solución.

constante en cada iteración siguiendo la regla:

$$\eta(t+1) \leftarrow \eta(t)/(1+e \times d),$$

siendo e la época (iteración) y d una constante a definir.

Implementar el algoritmo GD con paso variable simple, y probarlo para $x_0 = 2$, una precisión de parada de 0.0001, y $d = 1$.

Verificar que esta versión del algoritmo requiere menos pasos que la versión con paso fijo, y es más rápida que la versión con paso variable óptimo.

2) Regresión Lineal.

Ahora buscaremos aplicar descenso por gradiente y descenso por gradiente estocástico en un ejemplo más complejo, en una regresión lineal.

Utilizaremos datos sintéticos para visualizar la escalabilidad de los algoritmos frente al crecimiento de los datos. Generaremos 500.000 puntos entorno a la línea $y = 2x + 17 + \epsilon$, para valores de $x \in [0, 100]$. A partir de estos 500.000 elegiremos una muestra de 500 que llamaremos (x_{mini}, y_{mini}).

```
function res = f(x)
  res = 2*x +17 + randn(size(x))*5;
endfunction}

# genero datos
m = 500000;
x = unifrnd(0,1, [m,1])*100;
y = f(x);

# creo una muestra de 500 valores
# como son aleatorios simplemente elijo los primeros...
m_mini = 500
```

```
x_mini = x(1:m_mini);
y_mini = y(1:m_mini);
```

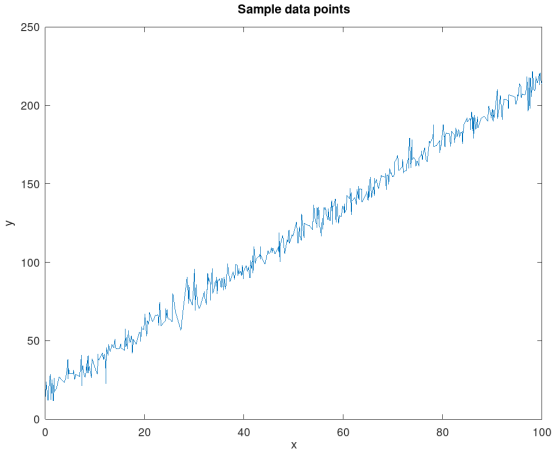

Figura 4: 500 puntos (x, y), elegidos aleatoriamente entorno a $y = 2x + 17 + \epsilon$.

A) **La regresión lineal.**

El modelo de regresión lineal puede escribirse como $h_\theta(x) = \theta_0 + \theta_1 x$. El problema de regresión lineal es encontrar los parámetros θ_0 y θ_1 que mejor ajusten a los datos de entrenamiento. La función de pérdida comunmente utilizada es el error cuadrático medio en los datos de entrenamiento (con m puntos (x, y)):

$$\mathcal{L}(x; \theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2.$$

Para los datos sintéticos, ¿cuáles son los valores que esperamos aprender de θ_0 y θ_1?

B) **Cálculo de gradiente.**

Verificar que en este caso, el gradiente de la función de pérdida respecto a los parámetros es:

$$\nabla_{\boldsymbol{\theta}} = \left(\frac{\partial \mathcal{L}}{\partial \theta_0}, \frac{\partial \mathcal{L}}{\partial \theta_1} \right),$$

$$\frac{\partial \mathcal{L}}{\partial \theta_0} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i),$$

$$\frac{\partial \mathcal{L}}{\partial \theta_1} = \frac{1}{m} \sum_{i=1}^{m} ((h_\theta(x_i) - y_i) \cdot x_i).$$

Implementar dichas funciones en `Octave/Matlab`.

C) **Descenso por gradiente.**

Al igual que en el ejemplo anterior, implementar el descenso por gradiente con paso fijo, el descenso por gradiente con paso óptimo, y el descenso de gradiente con paso variable sencillo en `Octave/Matlab`.

Probarlos con los datos de la muestra (x_{mini}, y_{mini}), y los parámetros:

- descenso por gradiente con paso fijo) $\theta_0 = 15$, $\theta_1 = 1$, $precision = 0.05$, $\eta = 0.0005$;
- descenso por gradiente con paso óptimo) $\theta_0 = 15$, $\theta_1 = 1$, $precision = 0.05$, $\eta = 0.0005$;
- descenso de gradiente con paso variable sencillo) $\theta_0 = 15$, $\theta_1 = 1$, $precision = 0.05$, $\eta = 0.0005$, $d = 0.00001$.

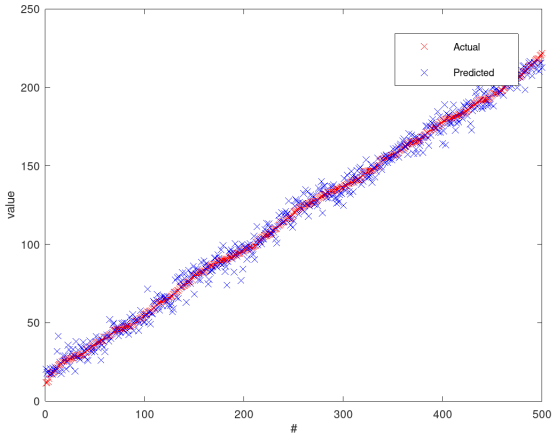

Figura 5: Predicción de la regresión lineal para los puntos de la muestra, usando GD con paso fijo.

¿Qué sucede con la velocidad de los algoritmos? ¿Convergen a los valores esperados? ¿Qué sucede si entrenamos estos algoritmos con todos los datos?

D) **Descenso por gradiente estocástico.**

Se observa que el algoritmo de descenso por gradiente (también llamado *batch gradient descent*.) utiliza todos los datos para calcular el gradiente, y esto puede ser prohibitivo para grandes conjuntos de datos.

En el descenso por gradiente estocástico, se actualizan los parámetros luego de observar cada punto de los datos, o más comunmente, luego de observar un conjunto pequeños de puntos aleatorios llamados *mini-batch*. Debido a esto, por lo general se utiliza un paso mucho más pequeño que en GD (puesto que el paso se aplica a cada *mini-batch*, sin embargo, la cantidad de épocas que se requiere en SGD dependerá de la cantidad de datos y del tamaño del *mini-batch*. Además, dado que cada iteración (sobre el *mini-batch*) lleva mucho menos tiempo que una iteración sobre todo el conjunto de datos, es posible ir observando la disminución en la pérdida y detener el algoritmo cuando se considere suficiente. En contrapartida es muy común ver una oscilación en la pérdida al acercarse al óptimo local debido a la aleatoriedad.

Implementar el algoritmo de descenso por gradiente estocástico con *mini-batchs* y paso fijo. El siguiente pseudo-código puede ayudar a definirlo:

Algorithm 2 Descenso por gradiente estocástico para la regresión lineal

Require: punto de partida $\boldsymbol{\theta}_n = (\theta_0, \theta_1)$, learning rate η, epocs E, minibatch size M
  for $e \in [1..E]$ **do**
    $(x_m, y_m) \leftarrow$ minibatch sample(x, y, M)
    $\boldsymbol{\theta}_{n+1} \leftarrow \boldsymbol{\theta}_n - \eta \nabla_{\boldsymbol{\theta}}(x_m, y_m, \boldsymbol{\theta}_n)$
  end for

Probar el algoritmo con los siguientes conjuntos de parámetros:

- datos de la muestra (x_{mini}, y_{mini}), $\theta_0 = 1$, $\theta_1 = 1$, $\eta = 0.0005$, $minibatch_{size} = 500$, $epocs = 30000$;
- datos totales (x, y), $\theta_0 = 1$, $\theta_1 = 1$, $\eta = 0.0005$, $minibatch_{size} = 500$, $epocs = 30000$;

¿Cuánto demora cada ejecución? ¿Obtienen buenos resultados?