PRÁCTICA 1:

Búsqueda Iterativa de Óptimos y Regresión Lineal

Aprendizaje Automático

Ángel Cabeza Martín

1. Ejercicio sobre la búsqueda iterativa de óptimos

1.1. Implementar el algoritmo de gradiente descendente.

Para implementar este algoritmo, he usado su ecuación general:

$$w_j := w_j - \eta \frac{\partial E_{in}(\mathbf{w})}{\partial w_j}$$

En esta ecuación nos encontramos los siguientes elementos:

1. **wj:** Es el punto en el cual nos encontramos en la iteración actual del algoritmo. Al principio toma un valor asignado por el programador, más adelante veremos la importancia de elegir de manera correcta el punto de inicio.
2. **η:** Denominado tasa de aprendizaje (learning rate). También designada por el programador. Nos indica la "velocidad" con la que el algoritmo se va ajustando por iteración a lo que quiera optimizar.
3. $\frac{\partial Ein(w)}{\partial wj}$ **:** derivadas parciales de w respecto a sus dos parámetros (u,v en este ejercicio).

Un apunte muy importante en esta fórmula, es que el signo del producto de la tasa de aprendizaje y las derivadas parciales es negativo. Esto indica que estamos descendiendo por la función, es decir, queremos minimizar. Si el signo fuese positivo estaríamos ascendiendo y por tanto maximizando.

Una vez comprendida la ecuación, solo queda pensar en cómo pasarla a código. Lo primero que hay que decidir son los parámetros que le tendremos que pasar al algoritmo.Hay dos parámetros que son muy claros: la tasa de aprendizaje y el punto inicial. Estas dos variables las tiene que "saber" el algoritmo antes de iniciarse y son elegidas por el programador teniendo un cierto criterio (aunque sea por la experiencia del ensayo y error y no matemático) por lo tanto es lógico pensar que pueden variar según lo que optimizas y por tanto deben de ser dos parámetros del algoritmo. Los otros dos parámetros que he elegido tienen que ver con el criterio de parada del algoritmo, es decir, ¿cuándo debería parar el algoritmo?. He decidido que el algoritmo debe parar cuando alcance un número determinado de iteraciones para que no se tire de manera indefinida ejecutándose o cuándo se llegue a un error aceptable. Por ejemplo: si estamos minimizando una función y alcanzamos el mínimo absoluto de la función (pongamos que el mínimo es 0.5) en la iteración 50000 pero a partir de la iteración 1000 el algoritmo toma valores muy cercanos a 0.5, es posible que nos salga rentable ahorrarnos 4000 iteraciones ya que el error es tan pequeño que no tiene efectos prácticos visibles.

Teniendo en cuenta estos criterios comentados anteriormente, el pseudocódigo del algoritmo es el siguiente:

```
def gradient_descent (initial_point, learning rate, error2get, maxIter)

      while ( valor_funcion > error2get and iteracion < maxIter)
            w = w - learning_rate * derivada_parcial_de_w
      end

      return w, it

end
```

Donde valor_funcion será el valor que toma la función que estamos optimizando en el punto actual de la iteración y derivada_parcial_de_w contendrá el valor de las derivadas parciales respecto a u y v (x e y). Devolvemos el punto en el que hemos encontrado el mínimo de la función y el número de iteraciones porque esta información nos resulta útil para saber si tanto el learning rate como el initial point que hemos escogido son buenos (el mínimo se alcanza en un número razonable de iteraciones) y para saber cúal es el mínimo de la función (a partir del punto mínimo de la función podemos obtener el valor del mínimo).

1.2 Buscar un mínimo de una función

Considerar la función E(u,v) = $(u^3 e^{(v-2)} - 2v^2 e^{-u})^2$ **.Usar gradiente descendente para encontrar un mínimo de esta función, comenzando desde el punto(u,v) = (1,1) y usando una tasa de aprendizaje η= 0,1.**

a) Calcular analíticamente y mostrar la expresión del gradiente de la función E(u,v)

El gradiente de la función E(u,v) es la derivada respecto de u de E y la derivada de v de E. Por lo tanto para mostrar la expresión del gradiente necesitaremos calcular estas dos derivadas.

La derivada de E(u,v) respecto a u es la siguiente:

$$\frac{d}{d(u)} = 2(u^3 e^{(v-2)} - 2v^2 e^{-u}) * (3u^2 e^{(v-2)} + 2v^2 e^{-u})$$

La derivada de E(u,v) respecto a v es la siguiente:

$$\frac{d}{d(v)} = 2(u^3 e^{(v-2)} - 2v^2 e^{-u}) * (u^3 e^{(v-2)} - 4ve^{-u})$$

Al ser E una potencia de orden dos, he usado la regla de derivación de la potencia:

$$f(x) = u^k \Rightarrow f'(x) = k * u^{k-1} * u'$$

Como se puede observar en ambas derivadas encontramos un 2 (antes era la potencia) multiplicando a E(u,v) que sería $k * u^{k-1}$ y esto multiplicando a lo que sería u'.

Esta u' es lo que más cambia respecto a una derivada y otra pero en las dos derivadas se utilizan las mismas reglas de derivación; la de la potencia, la de la función exponencial y la del producto de dos funciones.

Derivada de la función exponencial base e:

$$f(x) = e^u \Rightarrow f'(x) = u' * e^u$$

Derivada del producto de funciones:

$$f(x) = u(x) * v(x) \Rightarrow f'(x) = u'(x) * v(x) + u(x) * v'(x)$$

Lo único que es resaltable es que al derivar e^{-u} ,respecto a u, el signo cambia a ser negativo y por eso cambia el signo en la derivada.

b)¿Cuántas iteraciones tarda el algoritmo en obtener por primera vez un valor de E(u,v) inferior a 10^{-14}. (Usar flotantes de 64 bits)

Para realizar este experimento lo que he hecho es igualar el error a buscar a 10^{-14} y la variable que indica el máximo de iteraciones igualarla a un número muy grande para que la condición de parada sea casi exclusivamente cuando alcanzamos el error indicado.

Mi algoritmo ha encontrado un valor de E(u,v) inferior a 10^{-14} en la iteración 10, ha necesitado pocas iteraciones para llegar a este valor por lo que podemos pensar que el learning rate que hemos escogido (0.1) es idóneo para el objetivo que nos hemos marcado, pero de esto hablaremos en los siguientes apartados en los que profundizaremos más en este tema.

c) ¿En qué coordenadas(u,v)se alcanzó por primera vez un valor igual o menor a 10^{-14} en el apartado anterior?.

Este valor se alcanzó con las siguientes coordenadas:

(1.1572888496465497 , 0.9108383657484797)

donde el primer término corresponde al valor de x (u en nuestro caso) y el segundo al valor de y (v en nuestro caso).

Hemos alcanzado el objetivo con una coordenada muy parecida al punto inicial, así que podemos asegurar que hemos hilado muy fino a la hora de escoger el punto de inicio del algoritmo. Si hubiéramos escogido este punto de inicio un poco más alejado (por ejemplo en el punto (1.5,1.5)) tardaríamos algunas iteraciones más. Para este caso se tardarían 75 iteraciones pero encontraríamos un mínimo algo mejor. Sin embargo como en este experimento estamos buscando un valor menor a 10^{-14} nos convendría más dejar el punto inicial donde está y ahorrarnos muchas iteraciones, ya que está muy cerca de nuestro objetivo.

1.3 Importancia de la tasa de aprendizaje y del punto inicial

Considerar ahora la función f(x,y) = (x+ 2)2+ 2(y−2)2+ 2sin(2πx)sin(2πy)

Ahora vamos a cambiar de función, así que si queremos hacer experimentos con esta función lo primero que tenemos que hacer es calcular la expresión del gradiente de la función, es decir, la derivada de la función respecto x e y (en el código (x = u) e (y = v) para ahorrarnos nuevas funciones):
.

Derivada respecto de x:

$$\frac{d}{d(x)} = 2(x + 2) + 4\pi cos(2\pi x) * sin(2\pi y)$$

Derivada respecto de y:

$$\frac{d}{d(y)} = 4(y - 2) + 4\pi sin(2\pi x) * cos(2\pi y)$$

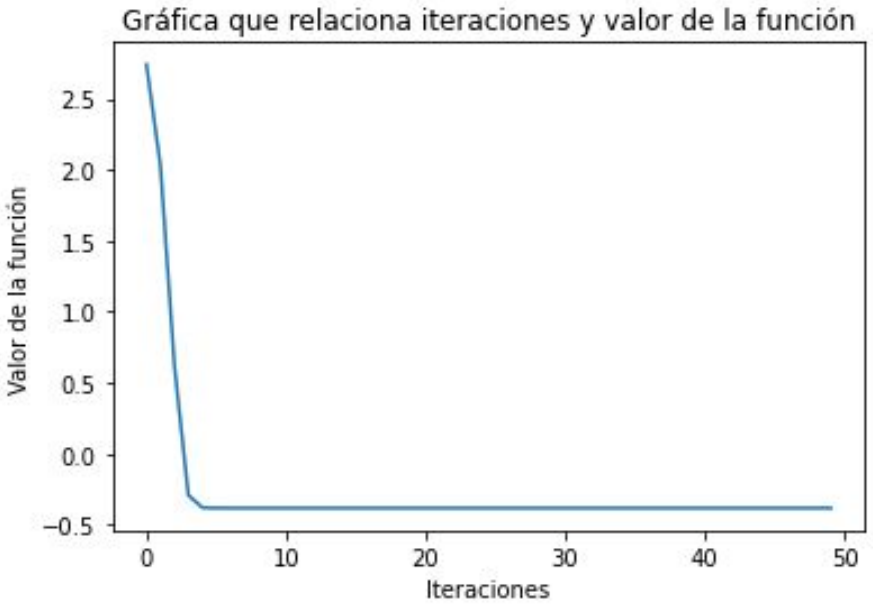
Para calcular estas derivadas hemos usado una regla de derivación nueva. La de la función seno, que se define así.

$$f(x) = sin(x) \Rightarrow f'(x) = cos(x)$$

a) Usar gradiente descendente para minimizar esta función. Usar como punto inicial (x0=−1,y0= 1), (tasa de aprendizaje η= 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 η= 0,1, comentar las diferencias y su dependencia de η.

Tras realizar estos dos experimentos he obtenido estas dos gráficas.

$\eta = 0,01$



$\eta = 0,1$

En la primera gráfica podemos ver que el algoritmo ha ido descendiendo por la gráfica a una velocidad considerable y ha encontrado un mínimo en la iteración 4. Sin embargo, a partir de ahí y hasta el final no ha sido capaz de salir de ese mínimo local. Esto se debe a que el learning rate escogido es pequeño y como vimos en teoría esto puede hacer que el algoritmo se quede estancado en un mínimo relativo y que le lleve muchas iteraciones salir de ahí o que directamente no salga.

En la segunda gráfica vemos como el valor de la función sube y baja con el paso de las iteraciones. Esto se debe a que el learning rate es alto y en vez de explorar la zona en la que está, el algoritmo se ha dedicado a explorar la función dejando atrás los mínimos que ha encontrado. Además gracias a esta gráfica podemos ver que el mínimo que encontramos con un learning rate de 0.01 es un mínimo local y no global lo que confirma nuestras sospechas.

b) 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. Comentar la dependencia del punto inicial.

Punto inicial	Mínimo	Coordenadas
(−0,5,−0,5)	2.4931912832664618	[-0.78365509,0.81314011]
(1,1)	6.4375695988659185	[0.67743878,1.29046913]
(2,1,−2,1)	12.490971442685037	[0.14880583, -0.0960677]
(−3,3)	-0.38124949743809955	[-2.73093565 ,2.71327913]
(−2,2)	-4.799231304517944e-31	[-2. 2]

En esta tabla podemos ver como la elección de un punto inicial influye en gran medida en los resultados obtenidos. Ya que dependiendo de este punto de inicio encontraremos un mínimo local o un mínimo global, menos cuando la función sea cóncava que siempre encontraremos un mínimo global.

1.4 Conclusión final

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

La dificultad de encontrar un mínimo global en una función arbitraria es muy alta, y depende principalmente de los siguientes factores:

1. Tasa de aprendizaje (learning rate): como hemos visto si el learning rate es pequeño, se quedará estancado en un mínimo local y si es muy grande explorará la función en vez de buscar un mínimo en una zona.
2. Punto de inicio: este parámetro nos indicará la zona que vamos a explorar, si no lo elegimos bien y empezamos en una zona muy alejada del mínimo global nos costará mucho encontrar el mínimo y probablemente solo encontremos un mínimo local.
3. De la función que estemos optimizando: además los dos factores anteriores también dependen de lo que estemos optimizando, según la función que optimicemos, escoger un learning rate grande o pequeño puede penalizar más o menos a la hora de buscar un mínimo.

Por lo tanto encontrar un mínimo global en una función arbitraria no es sencillo y se requiere un gran conocimiento del problema para poder configurar bien los parámetros del algoritmo.

2. Ejercicio sobre regresión lineal