UNIVERSIDAD
DE GRANADA

Aprendizaje Automático
grado en ingeniería informática

MEMORIA PRÁCTICA 1

Autor
Vladislav Nikolov Vasilev

Rama
Computación y Sistemas Inteligentes

Escuela Técnica Superior de Ingenierías Informática y de
Telecomunicación

Curso 2018-2019

Índice

1. Ejercicio sobre la búsqueda iterativa de óptimos

Apartado 1

Implementar el algoritmo de gradiente descendente.

A continuación se muestra el código en Python implementado:

In [3]:
```python
def descent_gradient(initial_w, function, gradient, eta=0.01, threshold=None, iterations=100):
    """
    Función para el cálculo del gradiente descendente

    :param initial_w: Pesos iniciales
    :param function: Función a evaluar
    :param gradient: Función gradiente a utilizar
    :param eta: Valor de la tasa de aprendizaje (por defecto 0.01)
    :param threshold: Valor umbral con el que parar (por defecto None)
    :param iterations: Número máximo de iteraciones que tiene que hacer el bucle
                       (por defecto 100)

    :returns: Devuelve el peso final (w), el número de iteraciones que ha llevado
              conseguir llegar hasta éste, un array con todos los w y un array con
              los valores de w evaluados en function
    """

    w = np.copy(initial_w)                  # Se copia initial_w para evitar modificarlo
    iter = 0                                # Se inicializan las iteraciones a 0
    w_list = []                             # Se inicializa una lista vacía con los valors de w
    func_values_list = []                   # Se inicializa una lista vacía con los valors de la función

    w_list.append(w)                        # Añadir valor inicial de w
    func_values_list.append(function(*w))   # Añadir valor inicial de w evaluado en function

    # Se realiza el cálculo de la gradiente descendente mientras no se superen
    # el número máximo de iteraciones.
    while iter < iterations:
        iter += 1
        w = w - eta * gradient(*w)              # Actualización de w con los nuevos valores

        w_list.append(w)                        # Añadir nuevo w
        func_values_list.append(function(*w))   # Añadir nueva evaluación de w en function

        # Si se ha especificado un umbral en el que parar y se ha pasado
        # se sale del bucle
        if threshold and function(*w) < threshold:
            break

    return w, iter, np.array(w_list), np.array(func_values_list)
```

Se ha intentado que esta implementación sea lo más general posible para poder utilizarla en los ejercicios posteriores, parametrizando la función que recibe (parámetro **function**), la cuál puede ser tanto $f(x, y)$ como $E(u, v)$. Con este motivo, también se ha parametrizado el error con el que se quiere ajustar, ya que en un caso no será necesario utilizar un error como criterio de parada (de ahí que su valor por defecto sea **None**). Y, adicionalmente, se ha parametrizado el gradiente (parámetro **gradient**), para que también se pueda especificar a la hora de la llamada cuál se usará.

Apartado 2

Considerar la función $E(u, v) = (u^2e^v - 2v^2e^{-u})^2$. Usar gradiente descendente para encontrar un mínimo de esta funcón, comenzando desde el punto $(u, v) = (1, 1)$ y usanto una tasa de aprendizaje $\eta = 0{,}01$.

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

Para calcular el gradiente, vamos a calcular antes $\frac{\partial E}{\partial u}$ y $\frac{\partial E}{\partial v}$. Las derivadas, al aplicar la regla de la cadena, quedarían de la siguiente forma:

$$\frac{\partial E}{\partial u} = \frac{\partial}{\partial u}\Big((u^2e^v - 2v^2e^{-u})^2\Big) = 2(u^2e^v - 2v^2e^{-u})\frac{\partial(u^2e^v - 2v^2e^{-u})}{\partial u} = \\ = 2(u^2e^v - 2v^2e^{-u})(2ue^v + 2v^2e^{-u}) \tag{1}$$

$$\frac{\partial E}{\partial v} = \frac{\partial}{\partial v}\Big((u^2e^v - 2v^2e^{-u})^2\Big) = 2(u^2e^v - 2v^2e^{-u})\frac{\partial(u^2e^v - 2v^2e^{-u})}{\partial v} = \\ = 2(u^2e^v - 2v^2e^{-u})(u^2e^v - 4ve^{-u}) \tag{2}$$

Con esto, tenemos que la expresión del gradiente es la siguiente:

$$\nabla E = \begin{bmatrix} \frac{\partial E}{\partial u} \\ \\ \frac{\partial E}{\partial v} \end{bmatrix} \tag{3}$$

$$\nabla E = \begin{bmatrix} 2(u^2e^v - 2v^2e^{-u})(2ue^v + 2v^2e^{-u}) \\ \\ 2(u^2e^v - 2v^2e^{-u})(u^2e^v - 4ve^{-u}) \end{bmatrix} \tag{4}$$

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)

El algoritmo tarda 33 iteraciones en encontrar el valor.

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

Las coordenadas donde se alcanzó un valor inferior a 10^{-14} son $(0{,}619, 0{,}968)$ (redondeadas a 3 cifras decimales).

Apartado 3

Considerar ahora la función $f(x, y) = x^2 + 2y^2 + 2\sin(2\pi x)\sin(2\pi y)$.

a) Usar gradiente descendente para minimizar esta función. Usar como punto inicial ($x_0 = 0{,}1, y_0 = 0{,}1$), tasa de aprendizaje $\eta = 0{,}01$ y un máximo de 50 iteraciones. Repetir el experimento pero usando $\eta = 0{,}1$, comentar las diferencias y su dependencia de η.

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

Apartado 4

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

2. Ejercicio sobre Regresión Lineal

3. Bonus

Referencias

[1] Texto referencia
`https://url.referencia.com`