Lecture 8: Gradient Descent

CSE 427: Machine Learning

Md Zahidul Hasan

Lecturer, Computer Science
BRAC University

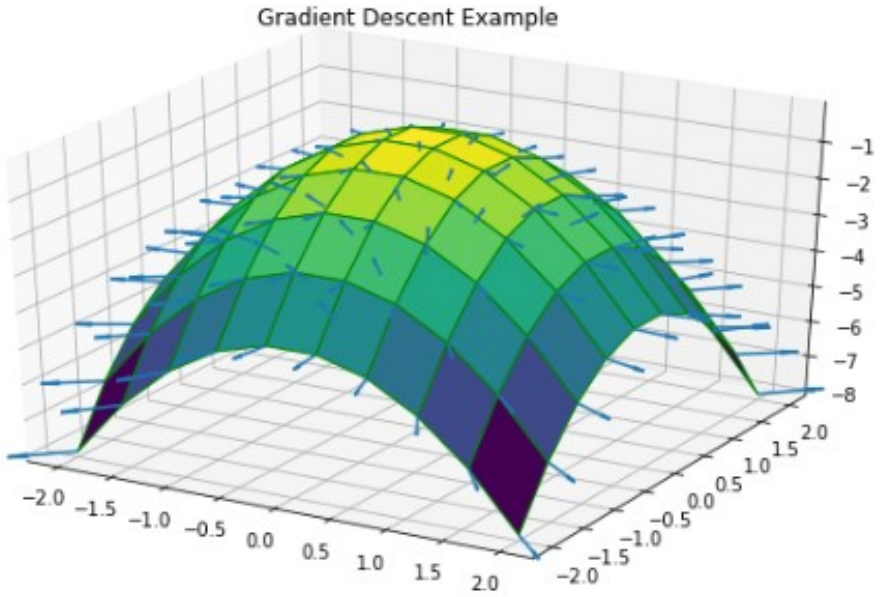
Spring 2023

Gradient

Definition

If $F(x_1, x_2, \cdots, x_n)$ is a scalar-valued function, then it's gradient is the following vector: $\nabla F = [\frac{\partial F}{\partial x_1}, \frac{\partial F}{\partial x_2}, \cdots, \frac{\partial F}{\partial x_n}]$

I plotted the function $f(x, y) = -x^2 - y^2$ in matplotlib. It's gradient is the vector $\nabla F = [-2x, -2y]$. Even though the graph is 3D, the gradient is 2D. It shows the direction of the steepest increase. We should change x and y in the prescribed direction.



Gradient Continued

Let's say, we make a small move according to the following vector $\Delta = [\Delta x, \Delta y]$. Then the change in the value of F should be as follows:

$\Delta F = \frac{\partial F}{\partial x}\Delta x + \frac{\partial F}{\partial y}\Delta y = (\nabla F)^T \Delta = |\nabla F||\Delta| \cos\theta$

where θ is the angle between the vectors Δ and ∇F. If the magnitude of our stride is fixed, then the only thing we can change is the angle θ. If we want to maximize ΔF which is the change in F, we can make $\theta = 0$. In that case, our movement Δ is in the same direction as the gradient ∇F. So, if we want to minimize the change in F, we should move in the opposite direction of the gradient by taking $\theta = 180^\circ$. So, gradient gives us the direction of the steepest increase. So, if we want to maximize a function. We should move in the direction of the gradient. If we want to minimize it, we should move in the opposite direction of the gradient. In the above graph, I have drawn vectors in the opposite direction of the gradient. In other words, I have plotted the vectors $[2x, 2y] = -\nabla F$.

Linear Regression With Gradient Descent

In the case of linear regression, our objective function is the squared error or the mean squared error.

$$E = \sum_{i=1}^{m} (y_i - h_w(x_i))^2.$$

where, $h_w(x) = w_1 x_1 + \cdots + w_n x_n + w_{n+1}$.
We can minimize this error using gradient descent. Since this is a quadratic function of the weights. It's convex, continuous and differentiable. We can find it's gradient with respect to the weights as follows: $\nabla E = [\frac{\partial F}{\partial w_1}, \cdots, \frac{\partial F}{\partial w_1}]$
For $1 \leq i \leq n$,

$$\frac{\partial F}{\partial w_i} = \sum_{j=1}^{m} 2(y_j - h_w(x_j))(-\frac{\partial h_w(x)}{\partial w_i}) = \sum_{j=1}^{m} 2(y_j - h_w(x_j))(-x_{j,i})$$

For $i = n + 1$,

$$\frac{\partial F}{\partial w_i} = \sum_{j=1}^{m} 2(y_j - h_w(x_j))(-\frac{\partial h_w(x)}{\partial w_i}) = \sum_{j=1}^{m} 2(y_j - h_w(x_j))(-1)$$

Linear Regression With Gradient Descent Continued

So, now we know in which direction to move. We can start with random values for the weights. For example, $w_1 = w_2 = \cdots = w_{n+1} = 0$. Then we can do the following until the values converge:

$$
\begin{aligned}
w_1 &= w_1 - \lambda \sum_{j=1}^{m} 2(y_j - h_w(x_j))(-x_{j,1}) \\
w_2 &= w_2 - \lambda \sum_{j=1}^{m} 2(y_j - h_w(x_j))(-x_{j,2}) \\
&\cdots \\
&\cdots \\
w_n &= w_n - \lambda \sum_{j=1}^{m} 2(y_j - h_w(x_j))(-x_{j,n}) \\
w_{n+1} &= w_{n+1} - \lambda \sum_{j=1}^{m} 2(y_j - h_w(x_j))(-1)
\end{aligned}
$$

Here λ is called the learning rate. It determines the length of our stride in the desired direction. The learning rate can change as the values approach the converging values. The above is an example of **batch gradient descent**.

Logistic Regression With Gradient Descent

In logistic regression, our objective to minimize is the following:

$$J = \sum_{i=1}^{m} y_i \log f(x_i) + (1 - y_i) \log (1 - f(x_i))$$

where $f(x) = g(h_w(x)) = \frac{1}{1+e^{-h_w(x)}}$. Now let's find the gradient.

$$\frac{\partial J}{\partial w_i} = \sum_{j=1}^{m} \frac{y_j}{f(x_j)} \frac{\partial f(x_j)}{\partial w_i} - \frac{1-y_j}{1-f(x_j)} \frac{\partial f(x_j)}{\partial w_i}$$

Using the chain rule of differentiation, we can easily see that, $\frac{\partial f(x)}{\partial w_i} = x_i f(x)(1 - f(x))$. Using this, we see that,

$$\frac{\partial J}{\partial w_i} = \sum_{j=1}^{m} [y_j - f(x_j)] x_{j,i}$$

So, the weight update rules will be:
For $1 \leq i \leq n$:

$$w_i = w_i - \lambda \sum_{j=1}^{m} [y_j - f(x_j)] x_{j,i}$$

For $i = n + 1$:

$$w_i = w_i - \lambda \sum_{j=1}^{m} [y_j - f(x_j)]$$

Softmax Regression With Gradient Descent

In softmax regression, we minimize the following objective function:

$$\sum_{i=1}^{m}\sum_{j=1}^{k} y_{i,j}\log \bar{y}_{i,j} \text{ where } \bar{y}_{i,j} = \frac{e^{h_{w_j}(x_i)}}{\sum_{l=1}^{k} e^{h_{w_l}(x_i)}}$$

Since y_i is a 1-hot vector. Only one of the $y_{i,j}$ is 1. The rest are 0.

$J_i = \sum_{j=1}^{k} y_{i,j}\log\frac{e^{h_{w_j}(x_i)}}{\sum_{l=1}^{k} e^{h_{w_l}(x_i)}}$

$= \sum_{j=1}^{k} y_{i,j}h_{w_j}(x_i) - \sum_{j=1}^{k} y_{i,j}\log\left(\sum_{l=1}^{k} e^{h_{w_l}(x_i)}\right)$

$= \sum_{j=1}^{k} y_{i,j}h_{w_j}(x_i) - \log\left(\sum_{l=1}^{k} e^{h_{w_l}(x_i)}\right)$

So, $\frac{\partial J_i}{\partial h_{w_q}(x_i)} = y_{i,q} - \frac{e^{h_{w_q}(x_i)}}{\sum_{l=1}^{k} e^{h_{w_l}(x_i)}} = y_{i,q} - \bar{y}_{i,q}.$

So, $\frac{\partial J_i}{\partial w_{q,j}} = \frac{\partial J_i}{\partial h_{w_q}(x_i)}\frac{\partial h_{w_q}(x_i)}{\partial w_{q,j}} = (y_{i,q} - \bar{y}_{i,q})x_{i,j}$

The update rules should be the following:

$$\text{For } 1 \le i \le n,\ w_{q,j} = w_{q,j} - \lambda\sum_{i=1}^{m}(y_{i,q} - \bar{y}_{i,q})x_{i,j}$$
$$\text{For } i = n+1,\ w_{q,j} = w_{q,j} - \lambda\sum_{i=1}^{m}(y_{i,q} - \bar{y}_{i,q})$$