Gradient descent: Understanding optimization algorithms

Benjamin P. Stager

Tulane University
Mathematics

March 23, 2023

Outline

What is gradient descent?

- We implement gradient descent to find a local minimum (or maximum) of any function that maps from $\mathbb{R}^n \to \mathbb{R}$
- The central idea is to begin with an initial guess and then 'descend' down the steepest path, which will lead you to the critical point
- The application of gradient descent falls under the category of machine learning, as we will see we must use an adaptive step size
- Gradient descent can see applications in PDE's, multidimensional functional analysis, statistics, linear modeling and more

An analogy

General application

- To understand gradient descent, it will be first important to understand the parameters of the method
- Consider a function F($\mathbf{x}$) such that $F : \mathbb{R}^n \rightarrow \mathbb{R}$
- We can 'descend' to find the location of the critical point using the scheme:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha_n \nabla F(\mathbf{x}_n) \tag{1}$$

- The key aspect of machine learning involved here is the idea of α_n. Many numerical methods set a constant stepsize, but we adapt α_n as we continue to descend to find the 'steepest' path
- We can find each α_n through the adaptive algorithm:

$$-\nabla F\Big((\mathbf{x}_n - \alpha_n \nabla F(\mathbf{x}_n))\Big)^T \nabla F(\mathbf{x}_n) = 0 \tag{2}$$

Other method for adaptive stepsizing

- The previous method involves a line search
- We can also implement the Barzilai-Borwein method, choosing our stepsize as a function of the current descent location and the previous descent location such that:

$$\alpha_n = \frac{|(\mathbf{x}_n - \mathbf{x}_{n-1})^T [\nabla F(\mathbf{x}_n) - \nabla F(\mathbf{x}_{n-1})]|}{||\nabla F(\mathbf{x}_n) - \nabla F(\mathbf{x}_{n-1})||^2} \tag{3}$$

- BB method will guarantee convergence of the method

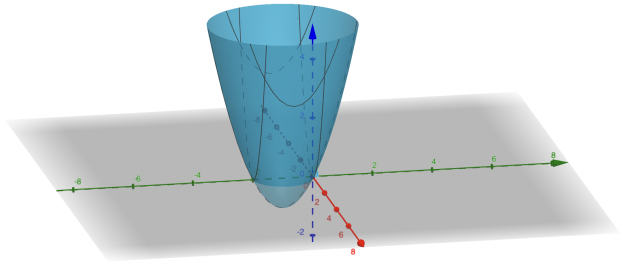
The algorithm

Algorithm 1: Gradient descent

Set initial guess $\mathbf{x}_0$, α_0;
while $\nabla F(\boldsymbol{x}_i) \geq \epsilon$ **do**
    $\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha_i \nabla F(\mathbf{x}_i)$;
    $i \rightarrow i + 1$
end

An example

- Consider the following function: $F(x, y) = x^2 + y^2 - 2y$



An example

- One can easily see from a 3-dimensional plot that the location of the minimum is $\mathbf{x} = (0, 1)$ - we will show a 'by hand' approach on how to begin the algorithm

$$\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial F}{\partial x} \\ \frac{\partial F}{\partial y} \end{bmatrix} = \begin{bmatrix} 2x \\ 2y - 2 \end{bmatrix}$$

- We will start at the point $\mathbf{x}_0 = (2, 2)$, and descend from there
- The next point is calculated as

$$\mathbf{x}_1 = \mathbf{x}_0 - \alpha_0 \nabla F(\mathbf{x}_0) \tag{4}$$

An example

- Recalling the expression we can find α_0 such that

$$G(\alpha) = -\nabla F(\mathbf{x}_0 - \alpha_0 \nabla F(\mathbf{x}_0))^T \nabla F(\mathbf{x}_0) = 0 \tag{5}$$

$$\implies -\nabla F((2,2) - \alpha_0(4,2))^T (4,2)^T = 0 \tag{6}$$

$$\implies -\nabla F(2 - 4\alpha_0, 2 - 2\alpha_0)^T (4,2)^T = 0 \tag{7}$$

$$\implies -4(4 - 8\alpha_0, 2 - 4\alpha_0)(4,2)^T = 0 \tag{8}$$

$$\implies -12 + 24\alpha_0 = 0 \implies \alpha_0 = 1/2 \tag{9}$$

An example

- Now that we have $\alpha_0 = 1/2$, we can find the corresponding $\mathbf{x}_0$

$$\mathbf{x}_1 = \mathbf{x}_0 - \frac{1}{2}\nabla F(\mathbf{x}_0) \tag{10}$$

$$\mathbf{x}_1 = (2,2) - (2,1) = (0,1) \quad (!!!!) \tag{11}$$

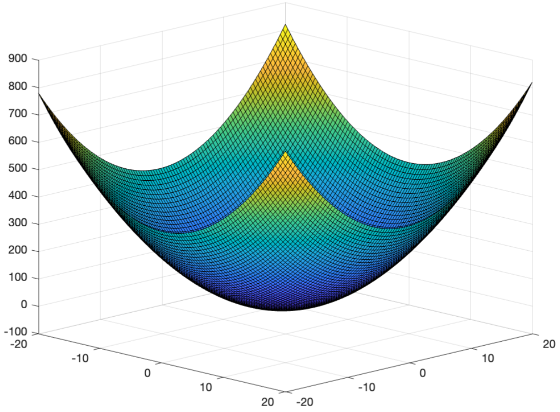
- Since this is a rather simple problem, it only took one iteration to find the critical point. We can verify this is a critical point with:

$$\nabla F(\mathbf{x} = (0,1)) = \tag{12}$$

$$\begin{bmatrix} 2 \cdot 0 \\ 2 \cdot 1 - 2 \end{bmatrix} = (0,0)^T$$

A numerical example

- Consider the equation: $f(x, y) = x^2 + y^2 - 2y + 3x$



A numerical example

```
function [X0, iter] = GD()

f = @(x,y) x.^2 + y.^2 - 2*y + 3*y;
% symbolic function to be used
syms fs xs ys
fs = xs.^2 + ys.^2 - 2*ys + 3*xs;


x = linspace(-20,20,100);
y = x';
z = f(x,y);
surf(x,y,z);


dfx = diff(fs,xs);
dfy = diff(fs,ys);
```

```
% converts symbolic functions to anonymous functions
gradx = matlabFunction(dfx);
grady = matlabFunction(dfy);

X0 = [5,3];
alpha = .5;
iter = 1;

for i = 1:20
    X0(1) = X0(1) - alpha * gradx(X0(1));
    X0(2) = X0(2) - alpha * grady(X0(2));
end


end
```

Using gradient descent to solve a linear system

We can use methods of gradient to solve linear systems. It can be an extremely efficient algorithm for complicated data sets

Consider the bivariate linear model:

$$Y = X\beta + \epsilon, \quad \beta \in \mathbb{R}^2 \tag{13}$$

We set β_0, $\beta_1 = 0$, and proceed with the following intermediaries

$$D_{\beta_0} = \frac{-2}{n} \sum_{i=0}^{n} (y_i - \hat{y}_i) \tag{14}$$

$$D_{\beta_1} = \frac{-2}{n} \sum_{i=0}^{n} x_i (y_i - \hat{y}_i) \tag{15}$$

Then update β_0, β_1 as follows:

$$\beta_0 \to \beta_0 - \alpha D_{\beta_0} \tag{16}$$

$$\beta_1 \to \beta_1 - \alpha D_{\beta_1} \tag{17}$$

gradient

2023-03-23

Load data

load(file = 'Davis.Rda') Davis = Davis[complete.cases(Davis),]

X = as.matrix(cbind(1,Davis[,'repwt']))

Y = as.matrix(Davis[,'weight'])

beta_start = t(cbind(0,0))

m = 0 c = 0

L = 0.0001 # The learning Rate

iter = 1000 # The number of iterations to perform gradient descent

n = length(X[,1]) # Number of elements in X

Performing Gradient Descent

for (i in 1:iter)

```
Y_pred = X %*% beta_start

D_m = (-2/n) * sum(X[,2] * (Y - Y_pred))

D_c = (-2/n) * sum(Y - Y_pred)

beta_start[1] = beta_start[1] - L * D_m

beta_start[2] = beta_start[1] - L * D_c
```

Conclusion of Davis data using gradient descent

- For a correct choice of learning rate α, we can find the coeffecients necessary to fit the linear model
- The Davis test confirms this
- If the learning rate is incorrectly fit or too high, the method will not converge, indicating the importance of the learning rate

References

Adarsh Menon. "Linear Regression Using Gradient Descent". In: *Towards Data Science* (2016)

Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *Aylien LTD.* (2017)