Gradient Descent

Lecture 9

Dr. Görkem Saygılı

Department of Biomedical Engineering
Ankara University

Numerical Methods, 2017-2018 Fall

Gradient Descent

Gradient Descent (GD) is a simple but effective optimization algorithm that is used to find the local minimum of a cost function by using derivative of the cost function with respect to the parameters.

When the algorithm converges, it returns the parameters that leads the cost function to the local minima.

Although the main goal is to find the global minimum, there is no guarantee that when the algorithm converges, it finds the global minimum.

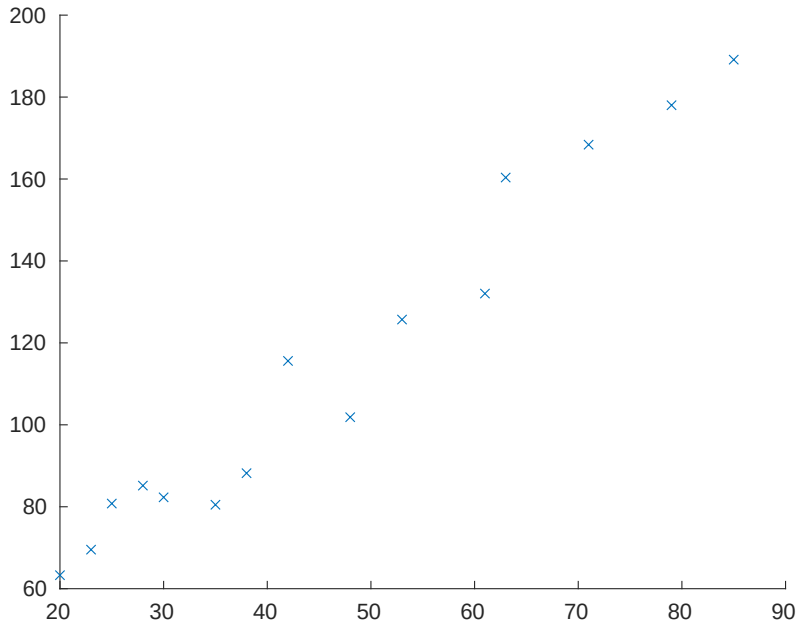
- Gradient Descent is a suitable optimization technique to be used with large amount of data.
- It has been extensively used in Machine Learning field, especially in the backpropagetion step of deep neural networks.
- It has different variants such as batch GD, stochastic GD, mini-batch GD. In batch GD, all of the samples are been used for the update whereas for stochastic GD, randomly selected samples are been used for the updates.

GD algorithm starts with initialization of the parameters, generally to zero. Then GD iteratively tunes these parameters until converges.

In the previous lectures, we learned about linear least squares which we use to fit a line on our data. We can use gradient descent similarly.

Linear Regression with GD

Let's observe an artificially created data:



We can see the linear characteristics in the data, hence we can use a linear representation to model our data.

Our aim here is to fit a line as we did for least squares linear regression. Let the equation of the fitted line be:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

We use mean square error as our cost function:

$$J_\theta(x) = \frac{1}{2} \sum_{i=1}^{M} (h_\theta(x^{(i)}) - y^{(i)})^2$$

where M is the total number of points (samples).

The objective of GD algorithm is to minimize $J_\theta(x)$ with respect to θ. Hence, the final result can be represented as:

$$\hat{\theta} = \arg\min_\theta J_\theta(x) = \arg\min_\theta (\frac{1}{2} \sum_{i=1}^{M} (h_\theta(x^{(i)}) - y^{(i)})^2)$$

At each iteration, the parameters are updated as:

$$\theta_i := \theta_i - \alpha \frac{\partial J(\theta)}{\partial \theta_i}$$

where $:=$ is the assignment operator representing the update and α is the learning rate.

Derivative of the cost with respect to θ_i can be calculated as:

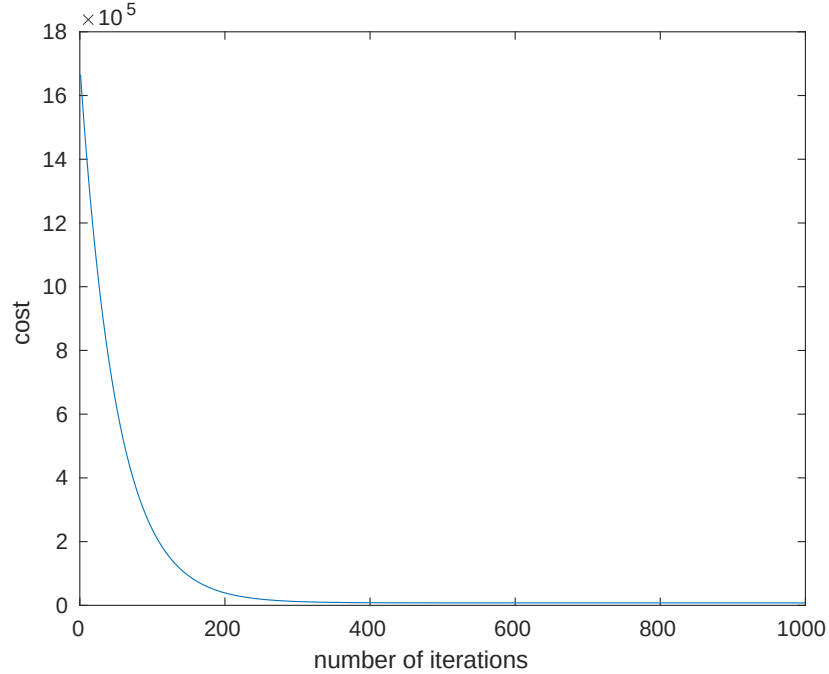
$$\frac{\partial J(\theta)}{\partial \theta_i} = \sum_{j=1}^{M} (h_\theta(x^{(j)}) - y^{(j)}) x_i^{(j)}$$

Hence we can compute θ_i at each update as:

$$\theta_i := \theta_i - \alpha \sum_{j=1}^{M} (h_\theta(x^{(j)}) - y^{(j)}) x_i^{(j)}$$

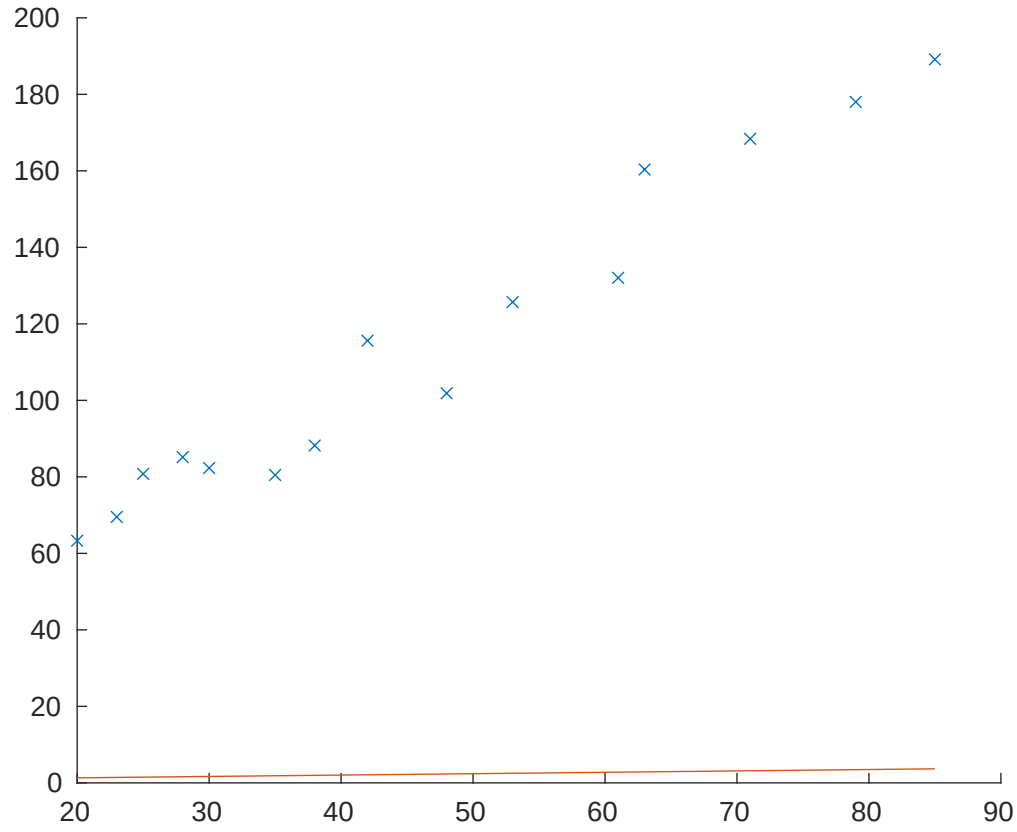
This algorithym is called Batch GD because it uses the entire data of M samples at each iteration.

At each iteration, we expect the cost to get smaller. This provides us a way to control the algorithm's convergence:

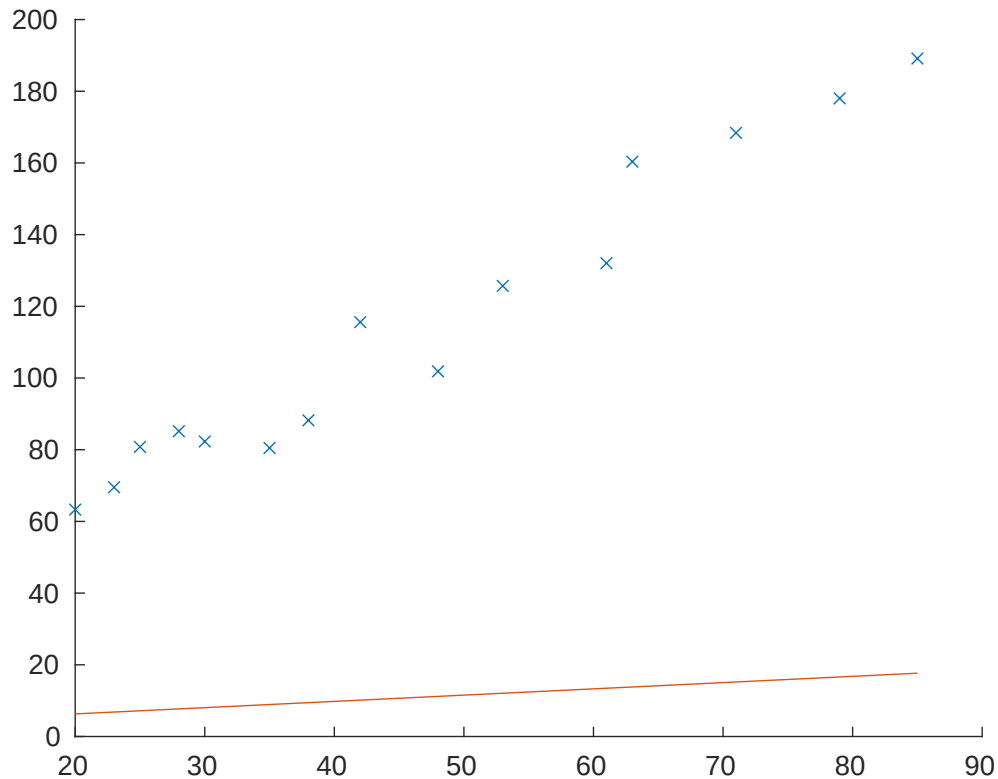


From the plot above, we can see that at each iteration, the cost gets smaller and converges before the total number of iterations reaches its maximum.

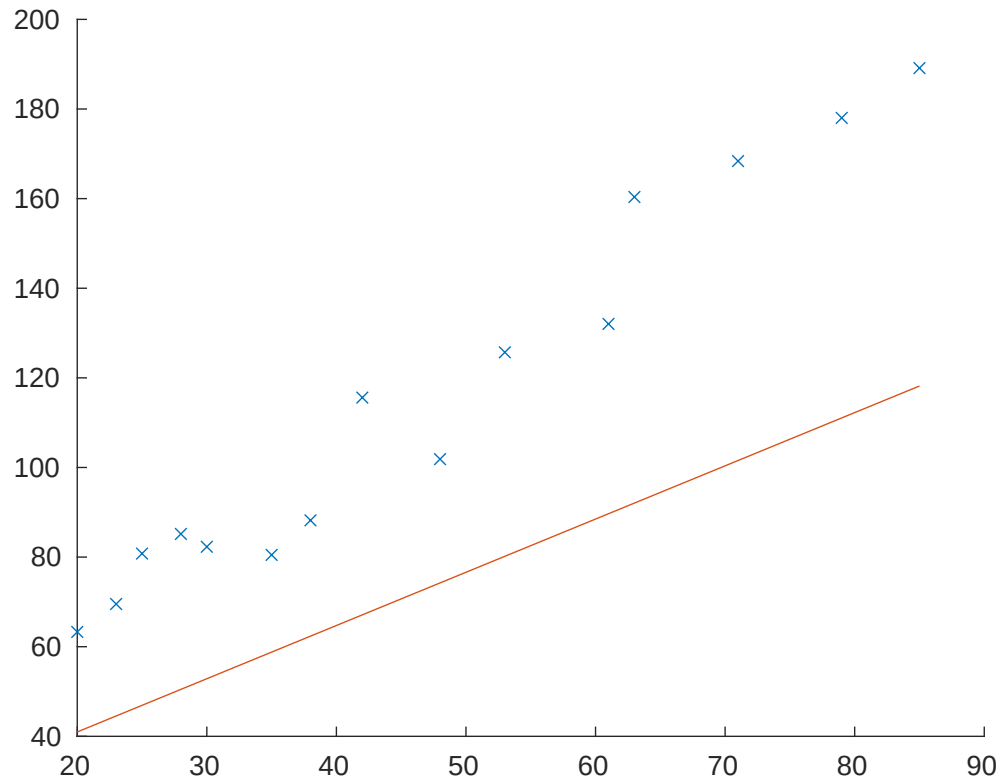
The fitted line at iteration 2 is:



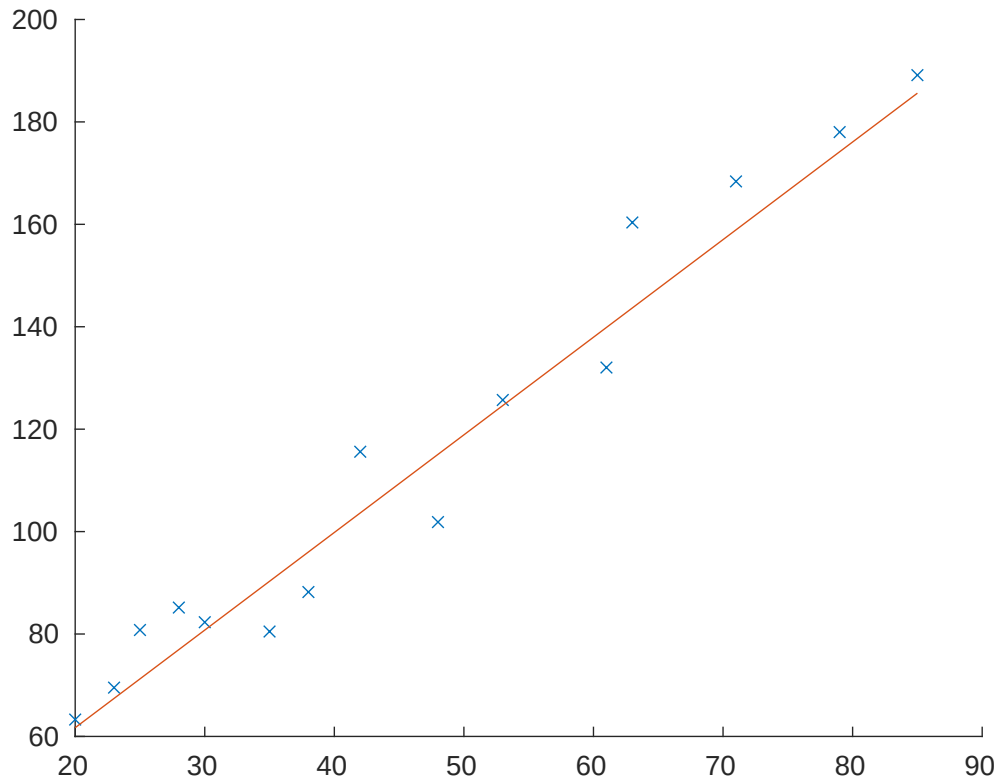
The fitted line at iteration 10 is:



The fitted line at iteration 100 is:



The fitted line at iteration 400 is:



Final predicted line is: