Outline

- Regression Set-up
- Linear Regression
- Polynomial Regression
- Underfitting/Overfitting
- Regularization
- Gradient Descent Algorithm

Gradient Descent Algorithm

Optimization and Gradient Descent - Overview

- Optimization refers to finding **optimal** value of your unknown variables under some **constraints** on the variables.
  - Optimal value: usually, maximizing or minimizing the objective function.
  - Constraints: restricting the domain of our variable and are defined by imposing eqality or inequality constraints on the function of the variable.

- An optimization problem of finding a variable $\boldsymbol{\theta}$ is usually formulated as

$$\begin{aligned} \text{minimize} \quad & f_o(\boldsymbol{\theta}) \\ \text{subject to} \quad & f_i(\boldsymbol{\theta}) \leq 0, \quad i = 1, 2, \ldots, m \\ & h_j(\boldsymbol{\theta}) = 0, \quad , j = 1, 2, \ldots, p \end{aligned}$$

$f_o(\boldsymbol{\theta})$ - Objective function $f_i(\boldsymbol{\theta})$ - Inequality constraint functions $h_j(\boldsymbol{\theta})$ - equality constraint functions

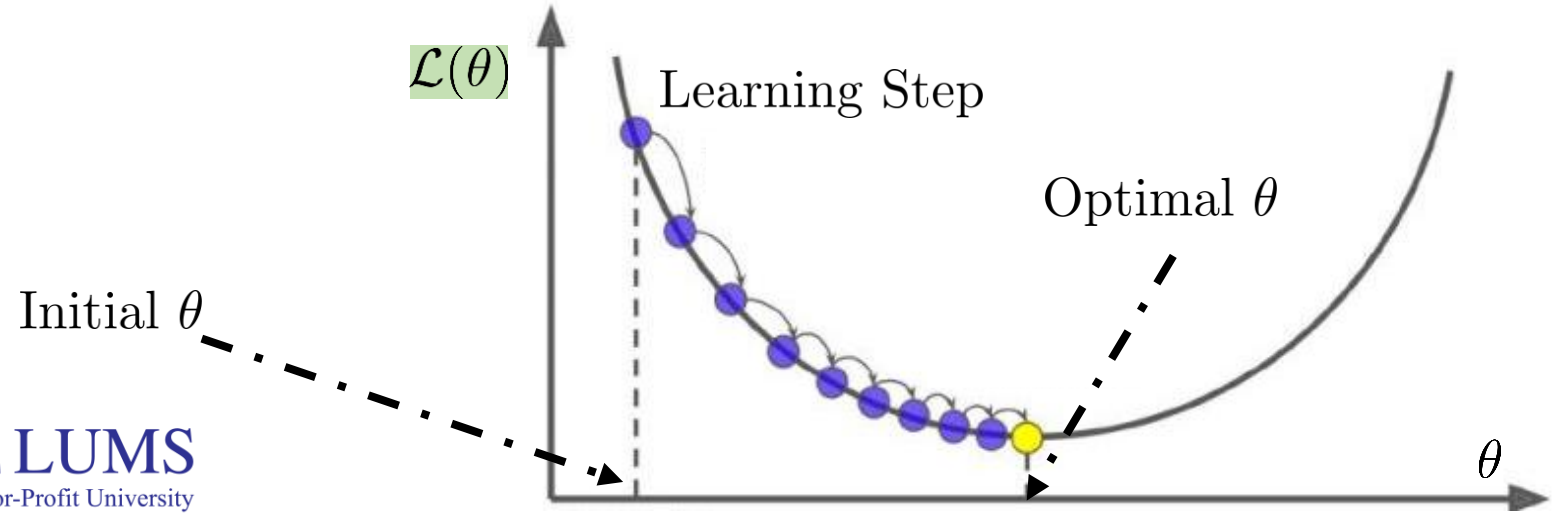
e.g., $\frac{1}{2}\|(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})\|_2^2$

- In ML, various algorithms (e.g., linear regression, neural network etc.) require us to solve an optimization problem.

Gradient Descent Algorithm

Optimization and Gradient Descent - Overview

- To solve the optimization problem, the gradient descent approach or algorithm is the most commonly used method.
- Gradient descent algorithm is best used when tha unknown variables cannot be determined analytically and need to be searched numerically.
- Gradient descent is an iterative algorithm in nature:
  - Initially, chose the coefficients to be something reasonabe (e.g., all zeros).
  - Iteratively update the coefficients in the direction of steepest descent until convergence.
  - Ensures that the new coefficients are better than the previous coefficients.



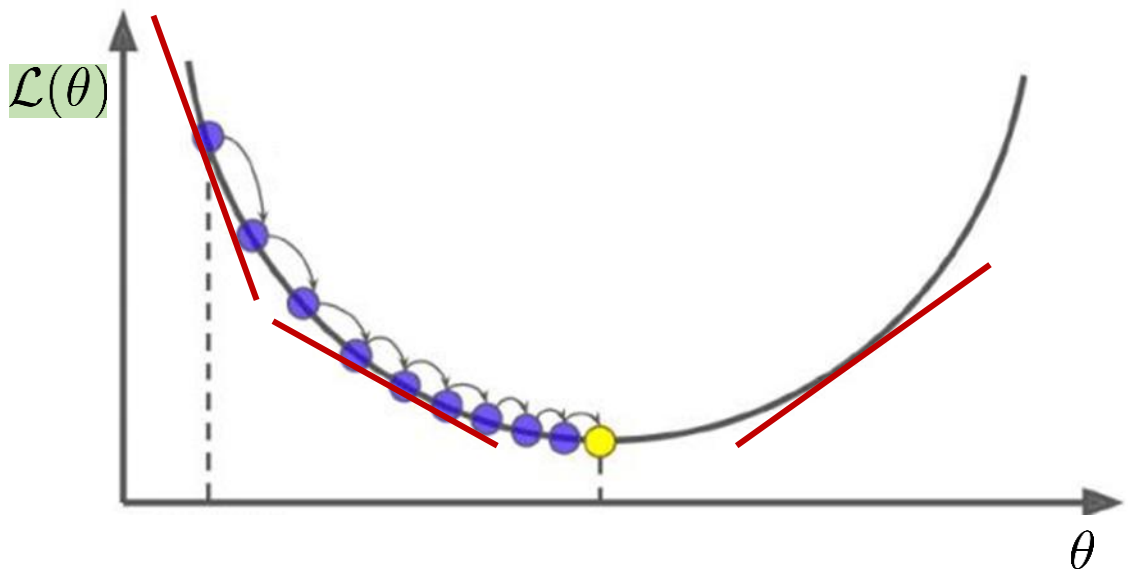
Gradient Descent Algorithm

Formulation:

- Loss function, denoted by $\mathcal{L}(\boldsymbol{\theta})$, required to be minimized. Assume $\boldsymbol{\theta} \in \mathbf{R}^d$.
- Interpretation of $\frac{\partial \mathcal{L}}{\partial \theta_i}$: Rate of change in the loss function with respect to θ_i
  - $\frac{\partial \mathcal{L}}{\partial \theta_i} > 0$: Increasing θ_i increases $\mathcal{L}$
  - $\frac{\partial \mathcal{L}}{\partial \theta_i} < 0$: Increasing θ_i decreases $\mathcal{L}$
- Noting this, the loss function is decreased with the following update:

$$\theta_i \leftarrow \theta_i - \alpha \frac{\partial \mathcal{L}}{\partial \theta_i}, \quad \alpha > 0$$

- This is the essence of gradient descent, that is, the step size in the direction of negative of the derivative.
- α is referred to as step size or learning rate.
  - Too small α: gradient descent can be slow.
  - Too large α: gradient descent can overshoot the minimum and it may fail to converge.



Gradient Descent Algorithm

Algorithm:

Overall:

- Start with some $\boldsymbol{\theta} \in \mathbf{R}^d$ and keep updating to reduce the loss function until we reach the minimum. Repeat until convergence

Pseudo-code:

- Initialize $\boldsymbol{\theta} \in \mathbf{R}^d$.
- Repeat until convergence:

$$\theta_i \leftarrow \theta_i - \alpha \frac{\partial \mathcal{L}}{\partial \theta_i}, \quad \text{for each} \quad i = 1, 2, \ldots, d$$

Equivalently,

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla \mathcal{L}(\boldsymbol{\theta})$$

Note: **Simultaneous update.**

Convergence and Step size:

- We stop updating $\boldsymbol{\theta}$ if $\nabla \mathcal{L}(\boldsymbol{\theta}) = 0$ or difference between the loss function in successive iterations is less than some threshold.
- We can have a constant step size α (typically 0.01, 0.05, 0.001) for each iteration or adjust it adpatively on each iteration.
- Algorithm converges for constant fixed rate as well due to the automatic smaller step size near the optimal solution.

Gradient Descent Algorithm

Linear Regression Case:

- We minimize mean-squared error (MSE) scaled by 1/2 factor:

$$\mathcal{L}(\theta_0, \boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^{n} \left(\theta_0 + \boldsymbol{\theta}^T \mathbf{x_i} - y_i\right)^2$$

- First we take a single feature regression; $\mathbf{x}$ is a scalar, $\mathbf{x_i} \equiv x_i$.

$$\mathcal{L}(\theta_0, \theta_1) = \frac{1}{2n} \sum_{i=1}^{n} \left(\theta_0 + \theta_1 x_i - y_i\right)^2$$

- We define partial derivatives as

$$\frac{\partial \mathcal{L}}{\partial \theta_0} = \frac{1}{n} \sum_{i=1}^{n} (\theta_0 + \theta_1 x_i - y_i) \qquad \frac{\partial \mathcal{L}}{\partial \theta_1} = \frac{1}{n} \sum_{i=1}^{n} (\theta_0 + \theta_1 x_i - y_i)\, x_i$$

Gradient Descent:

- Repeat until convergence:

$$\theta_0 \leftarrow \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^{n} (\theta_0 + \theta_1 x_i - y_i) \qquad \theta_1 \leftarrow \theta_1 - \alpha \frac{1}{n} \sum_{i=1}^{n} (\theta_0 + \theta_1 x_i - y_i)\, x_i$$

Note:

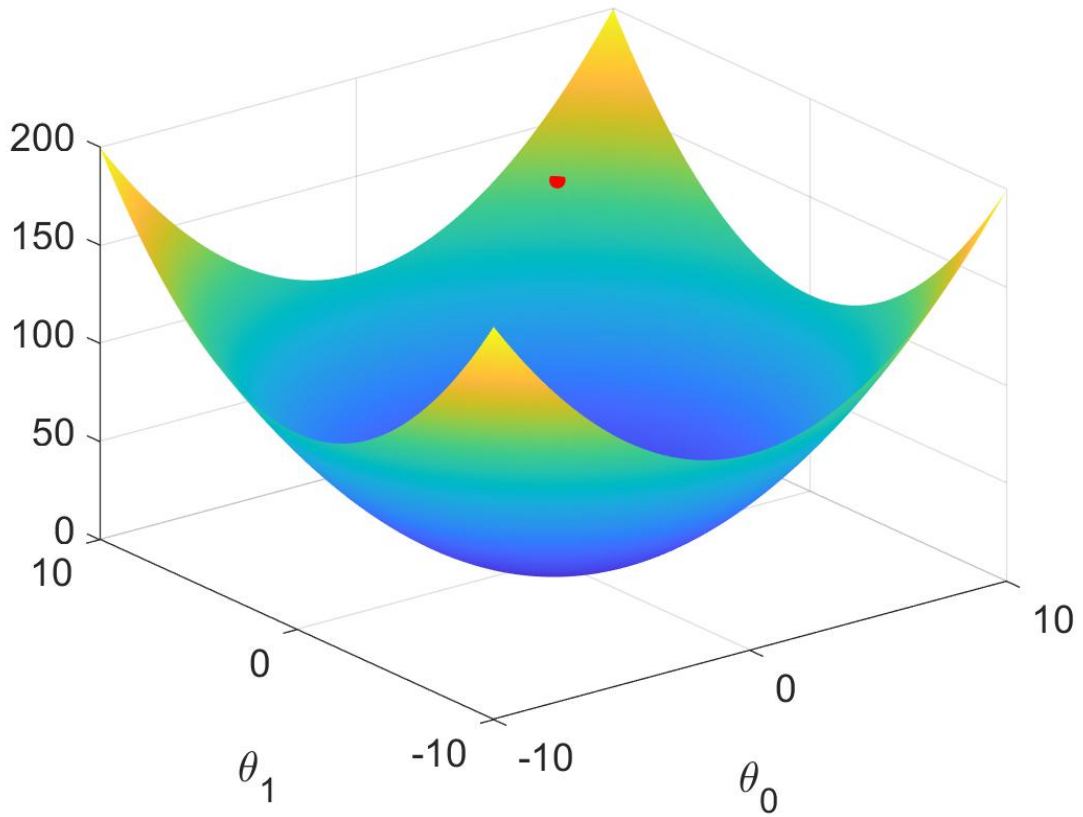
Simultaneous update.

Gradient Descent Algorithm

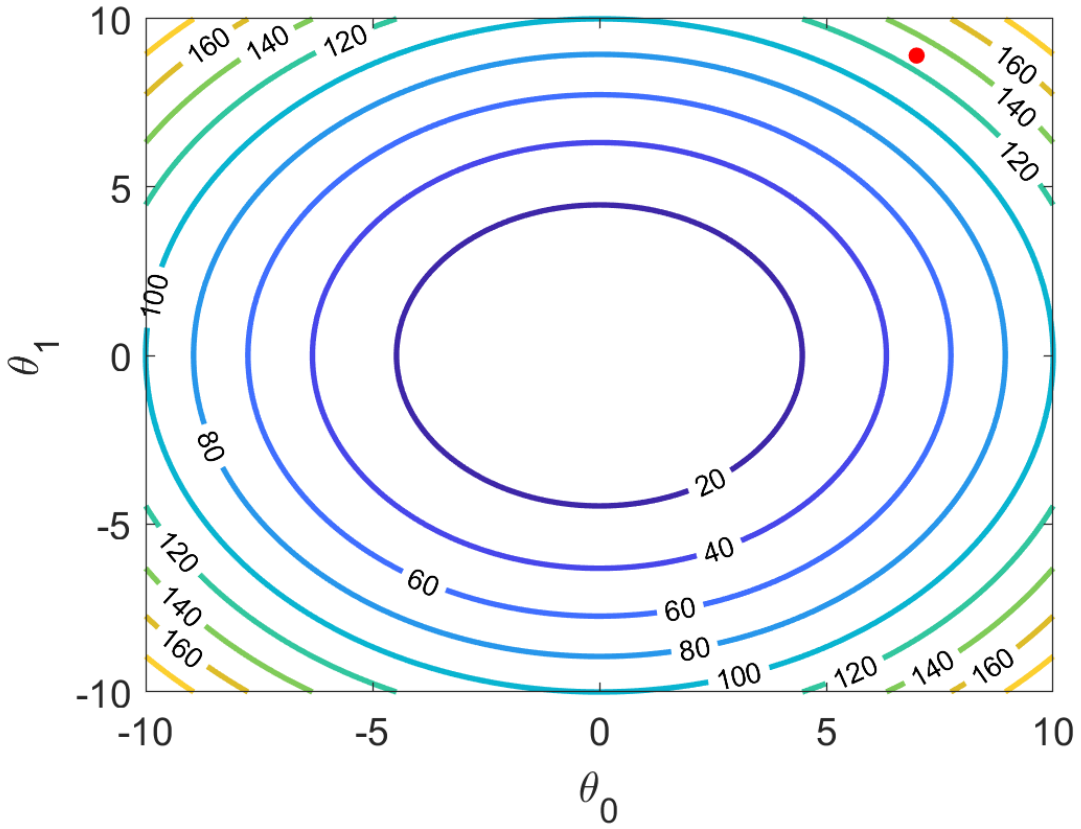
Linear Regression Case:

Visualization:

$$\mathcal{L}(\theta_0, \theta_1)$$



Surface plot



Contour plot

$$\alpha = 0.05,\ 0.2,\ 0.8,\ 1$$

Gradient Descent Algorithm

Linear Regression Case:

- For a multiple feature regression; $\mathbf{x}$ is a vector, $\mathbf{x_i} \in \mathbf{R}^d$.

$$\mathcal{L}(\theta_0, \boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^{n} \left(\theta_0 + \boldsymbol{\theta}^T \mathbf{x_i} - y_i\right)^2$$

- We define partial derivatives as

$$\frac{\partial \mathcal{L}}{\partial \theta_0} = \frac{1}{n} \sum_{i=1}^{n} (\theta_0 + \boldsymbol{\theta}^T \mathbf{x_i} - y_i) \qquad \frac{\partial \mathcal{L}}{\partial \theta_j} = \frac{1}{n} \sum_{i=1}^{n} (\theta_0 + \boldsymbol{\theta}^T \mathbf{x_i} - y_i) \mathbf{x_i}^{(j)}$$

where $\mathbf{x_i}^{(j)}$ denotes the j-th component of $\mathbf{x_i}$.

Gradient Descent:

- Repeat until convergence:

$$\theta_0 \leftarrow \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^{n} (\theta_0 + \boldsymbol{\theta}^T \mathbf{x_i} - y_i) \qquad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \frac{1}{n} \sum_{i=1}^{n} (\theta_0 + \boldsymbol{\theta}^T \mathbf{x_i} - y_i) \mathbf{x_i}$$

Note:

Simultaneous update.

Gradient Descent Algorithm

Notes:

- As we have taken all n points for updating at each step, we refer to the algorithm discussed here as **Batch Gradient Descent**.
- We also use the term 'epoch' to refer to one sweep of all the points in the data-set. So far, iteration is same as epoch as we have taken all the points at each step.
- We prefer to use gradient descent also for linear regression despite the fact that we can find the optimal solution analytically. **Why?**
- Gradient descent is easy to implement than the analytical solution.
- Gradient descent is computationally more efficient:
  - Closed-form (direct) solution: $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$
  - Size of $\mathbf{X}^T\mathbf{X}$ is $d \times d$, matrix inversion computational complexity is $\mathcal{O}(d^3)$.
  - Computational complexity of each update of gradient descent is $\mathcal{O}(n\,d)$.
  - $\mathcal{O}(n\,d)$ is better than $\mathcal{O}(d^3)$ when $d \gg 1$.

Stochastic Gradient Descent:

- For large data-sets such that the computation of gradient for all points in the data-set takes too much time, we use stochastic gradient descent.

Gradient Descent Algorithm

Stochastic Gradient Descent (SGD) - Rationale:

- Generalize the formulation by defining a loss function using model $\hat{f}(\mathbf{x_i}, \mathbf{w})$

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2n}\sum_{i=1}^{n}\left(\hat{f}(\mathbf{x_i}, \mathbf{w}) - y_i\right)^2 = \frac{1}{n}\sum_{i=1}^{n} g(\mathbf{x_i}, \mathbf{w}, y_i)$$

where

$$g(\mathbf{x_i}, \mathbf{w}, y_i) = \frac{1}{2}\left(\hat{f}(\mathbf{x_i}, \mathbf{w}) - y_i\right)^2$$

Quantifies the prediction error for a single input.

- In Batch (or Full) Gradient Descent, we update in each iteration as

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla \mathcal{L}(\mathbf{w})$$

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{1}{2n}\sum_{i=1}^{n} \nabla\left(\hat{f}(\mathbf{x_i}, \mathbf{w}) - y_i\right)^2$$

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{1}{n}\sum_{i=1}^{n} \nabla g(\mathbf{x_i}, \mathbf{w}, y_i)$$

- We are computing gradient for all n points.
  - n can be very large in practice.
  - Computationally expensive.

Gradient Descent Algorithm

Stochastic Gradient Descent (SGD):

- Stochastic gradient descent: update using one data point at each iteration

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla g(\mathbf{x_i}, \mathbf{w}, y_i)$$

- Also referred to as incremental or online gradient descent.
- This update tries to approximate the update of batch gradient descent.
- Q: How do we choose i in each iteration?
  - Stochastic selection: uniformly choose the index in each iteration.
  - Cyclic selection: choose $i = 1, 2, \ldots, n, 1, 2, \ldots, n, 1, 2, \ldots$
- Stochastic (random) selection, mostly used in practice, implies that SGD is using an unbiased estimate of the true gradient at each iteration.

Pros:

- Computationally efficient: iteration cost is independent of n.
- True gradient approximation can help in escaping the local minimum.

Gradient Descent Algorithm

SGD for Linear Regression Case:

- Using cyclic selection, we have the following SGD:

- Initialize $\theta_0 \in \mathbf{R}$ and $\boldsymbol{\theta} \in \mathbf{R}^d$.
- Repeat until convergence:

for $i = 1, 2, \ldots n$

$$\theta_0 \leftarrow \theta_0 - \alpha(\theta_0 + \boldsymbol{\theta}^T \mathbf{x_i} - y_i)$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha\,(\theta_0 + \boldsymbol{\theta}^T \mathbf{x_i} - y_i)\mathbf{x_i}$$

} Iteration

} Epoch

end for

- Even with cyclic seleciton, we shuffle the order in which we are using the data points after each epoch. Otherwise, algorithm can get stuck with the sequence of gradient updates that may cancel each other and conseqeuntly hinder learning.
- For online learning when the data points are arriving in a stream, we need to carry out predictions before we have all the data-points. In such a case, we use SGD for learning.

Gradient Descent Algorithm

Mini-batch Stochastic Gradient Descent (SGD) :

Batch Gradient Descent

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{1}{n} \sum_{i=1}^{n} \nabla g(\mathbf{x_i}, \mathbf{w}, y_i)$$

Stochastic Gradient Descent

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla g(\mathbf{x_i}, \mathbf{w}, y_i)$$

- Mini-batch Stochastic Gradient Descent: update using a subset of k data-points.
- From a set $\mathcal{D}$ of n points, we randomly select a subset, denoted by $\mathcal{S} \subseteq \mathcal{D}$ of $k \ll n$ points and use these k points to update the gradient as

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{1}{n} \sum_{i=1}^{k} \nabla g(\mathbf{x_i}, \mathbf{w}, y_i), \quad (\mathbf{x_i}, y_i) \in \mathcal{S}$$

- In one epoch, we divide the data into mini-batches and run mini-batch SGD on each subset iteratively.