Popular Machine Learning Methods: Idea, Practice and Math

Part 2, Chapter 2, Section 1: Linear Regression

Yuxiao Huang

Data Science, Columbian College of Arts & Sciences
George Washington University

Fall 2020

Reference

- This set of slices was largely built on the following 7 wonderful books and a wide range of fabulous papers:

HML	Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (2nd Edition)
PML	Python Machine Learning (3rd Edition)
ESL	The Elements of Statistical Learning (2nd Edition)
LFD	Learning From Data
NND	Neural Network Design (2nd Edition)
NNDL	Neural Network and Deep Learning
RL	Reinforcement Learning: An Introduction

- For most materials covered in the slides, we will specify their corresponding books and papers for further reference.

Code Example & Case Study

- See related code example in github repository:
  /p2_c2_s2_linear_regression/code_example
- See related case study of Kaggle Competition in github repository:
  /p2_c2_s2_linear_regression/case_study

Table of Contents

1. Learning Objectives
2. Linear Regression
3. Linear Regression in Sklearn
4. Training Linear Regression
5. Appendix
6. Reference

Learning Objectives

- It is **expected** to understand
  - the idea of linear regression
  - the good practice for using sklearn LinearRegression and SGDRegressor
- It is **recommended** to understand
  - the time complexity of sklearn LinearRegression and SGDRegressor

Kaggle Competition: Predicting House Price

Figure 1: Kaggle competition: predicting house price. Picture courtesy of Kaggle.

- The House Prices (Advanced Regression Techniques) dataset:
    - features: 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa
    - target: the sale price of homes

The House Prices Dataset

Table 1: The first 7 features and target (SalePrice) of the House Prices Dataset.

Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	SalePrice
1	60	RL	65.0	8450	Pave	NaN	208500
2	20	RL	80.0	9600	Pave	NaN	181500
3	60	RL	68.0	11250	Pave	NaN	223500
4	70	RL	60.0	9550	Pave	NaN	140000
5	60	RL	84.0	14260	Pave	NaN	250000

- The goal of this Kaggle competition is using the features (see the first 7 in table 1) to predict the target, SalePrice.
- Since SalePrice can take infinite number of values, it is a *Continuous* variable.
- We call this kind of prediction (where the target is continuous) *Regression*.
- We will apply the simplest regression model, *Linear Regression*, to this competition.

Linear Regression

- A key reason for linear regression being the simplest regression model is that, it assumes a linear relationship between the features and target (hence the name).
- Concretely, in linear regression the target is modeled as a weighted sum of features:

$$\widehat{\mathbf{y}} = b + w_1\mathbf{x}_1 +, \ldots, +w_n\mathbf{x}_n. \tag{1}$$

  Here:
  - $\widehat{\mathbf{y}}$ is the predicted target vector
  - $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are the feature vectors
  - b is the bias
  - $w_1, \ldots, w_n$ are the weights of $\mathbf{x}_1, \ldots, \mathbf{x}_n$
- We can also write eq. (1) in matrix form:

$$\widehat{\mathbf{y}} = \begin{bmatrix} \mathbf{1} & \mathbf{X} \end{bmatrix} \boldsymbol{\theta}. \tag{2}$$

  Here:
  - $\mathbf{X}$ is a $m \times n$ feature matrix (with m/n being the number of samples/features):

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \cdots & \mathbf{x}_n \end{bmatrix} \tag{3}$$

  - $\mathbf{1}$ is a $m \times 1$ vector, full of 1s
  - $\boldsymbol{\theta}$ is the parameter vector:

$$\boldsymbol{\theta} = \begin{bmatrix} b & w_1 & \cdots & w_n \end{bmatrix}^\intercal \tag{4}$$

A 2D Example: Simple Linear Regression

Table 1: The first 7 features and target (SalePrice) of the House Prices Dataset.

Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	SalePrice
1	60	RL	65.0	8450	Pave	NaN	208500
2	20	RL	80.0	9600	Pave	NaN	181500
3	60	RL	68.0	11250	Pave	NaN	223500
4	70	RL	60.0	9550	Pave	NaN	140000
5	60	RL	84.0	14260	Pave	NaN	250000

- Suppose we want to use only one feature say, LotFrontage, to predict the target, SalePrice. Then linear regression composed of the two variables is

$$\widehat{\mathrm{SalePrice}} = b + w \times \mathrm{LotFrontage}. \tag{5}$$

  Here:

  - $\widehat{\mathrm{SalePrice}}$ is the predicted target vector
  - LotFrontage is the feature vector
  - b is the bias
  - w is the weight of LotFrontage

- Liner regression with only one feature is also called *Simple Linear Regression*.

Visualizing the 2D Data

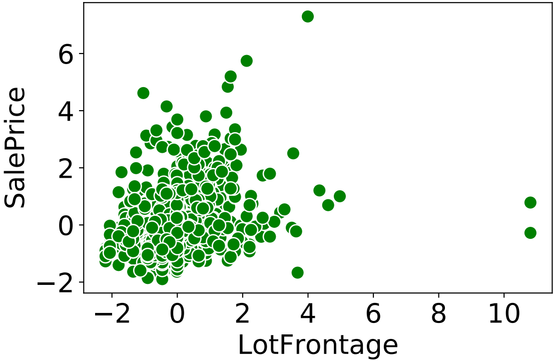

Figure 2: The scatter plot between feature LotFrontage and target SalePrice.

Visualizing the Simple Linear Regression

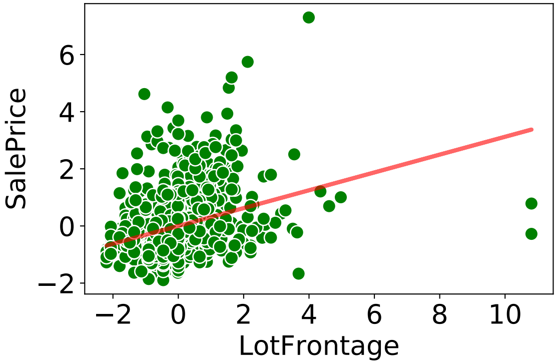

Figure 3: Simple linear regression with feature LotFrontage and target SalePrice.

A 3D Example: Multiple Linear Regression

Table 1: The first 7 features and target (SalePrice) of the House Prices Dataset.

Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	SalePrice
1	60	RL	65.0	8450	Pave	NaN	208500
2	20	RL	80.0	9600	Pave	NaN	181500
3	60	RL	68.0	11250	Pave	NaN	223500
4	70	RL	60.0	9550	Pave	NaN	140000
5	60	RL	84.0	14260	Pave	NaN	250000

- Suppose now we want to use two features say, LotFrontage and LotArea, to predict the target, SalePrice. Then linear regression composed of the three variables is

$$\widehat{\mathrm{SalePrice}} = b + w_1 \times \mathrm{LotFrontage} + w_2 \times \mathrm{LotArea}. \tag{6}$$

  Here:

  - $\widehat{\mathrm{SalePrice}}$ is the predicted target vector
  - LotFrontage and LotArea are the feature vectors
  - b is the bias
  - w_1 and w_2 are the weight of LotFrontage and LotArea
- Liner regression with multiple features is also called *Multiple Linear Regression*.

Visualizing the 3D Data

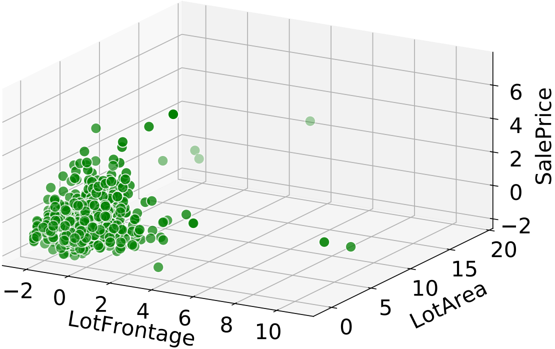

Figure 4: The scatter plot between feature LotFrontage, LotArea and target SalePrice.

Visualizing the Multiple Linear Regression

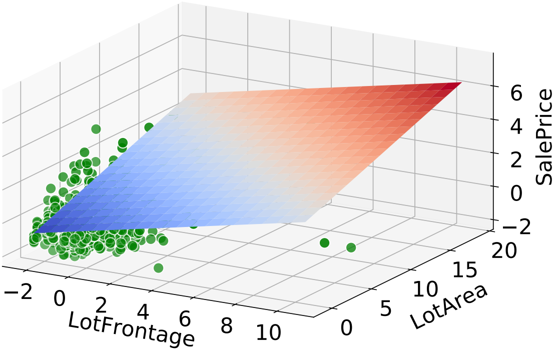

Figure 5: Multiple linear regression with feature LotFrontage, LotArea and target SalePrice.

Sklearn LinearRegression: Code Example

- See /p2_c2_s1_linear_regression/code_example:
  1. cell

Sklearn SGDRegressor: Code Example

- See /p2_c2_s1_linear_regression/code_example:
  1. cell

LinearRegression VS SGDRegressor

- **Q:** Since there are two sklearn implementations for linear regression, which one shall we use?

LinearRegression VS SGDRegressor

- **Q:** Since there are two sklearn implementations for linear regression, which one shall we use?
- **A:** It is largely based on two factors:
  - the number of samples in the data, m
  - the number of features in the data, n

Good practice

- When m is very large (e.g., millions):
  - it is recommended to use SGDRegressor
- When m is relatively small (e.g., thousands):
  - when $m \gg n$:
    - it is recommended to use LinearRegression (which is faster)
  - when $m \ll n$:
    - it is recommended to use SGDRegressor (which is faster)

- See the explanation of the good practice in Appendix (pages 62 to 64).

The Goal

- The goal of training linear regression is tweaking the parameters of the model in such a way that we can improve the model.
- There are two key elements in training linear regression:
  - Loss Function
  - Optimization
- The *Loss Function* of linear regression:
  - is a function of the parameters of the model
  - is a measurement of the *badness* of the model
- The *Optimization* for linear regression is a method that tweaks the parameters by minimizing the loss function (since it measures how bad the model is).

Mean Squared Error

- One popular loss function of linear regression is the *Mean Squared Error* (MSE), $\mathcal{L}(\boldsymbol{\theta})$, which measures the average squared difference between the real target value and the predicted target value, across all the samples:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}(y^i - \widehat{y^i})^2 \quad \text{where} \quad \widehat{y^i} = b + w_1 x_1^i +, \ldots, +w_n x_n^i. \tag{7}$$

  Here:

  - m is the number of samples in the data
  - y^i / $\widehat{y^i}$ is the real / predicted target value of sample i
  - $x_1^i, \ldots, x_n^i$ are the feature values of sample i
  - $\boldsymbol{\theta} = \begin{bmatrix} b & w_1 & \cdots & w_n \end{bmatrix}^\intercal$ are the parameters

- We can decompose eq. (7) as

$$\mathcal{L}(\boldsymbol{\theta}) = \overbrace{\frac{1}{m}\overbrace{\sum_{i=1}^{m}\underbrace{(\underbrace{y^i - \widehat{y^i}}_{\text{error}})^2}_{\text{squared error}}}^{\text{sum of squared error}}}^{\text{mean squared error}}. \tag{8}$$

Surface and Contour Plot of MSE

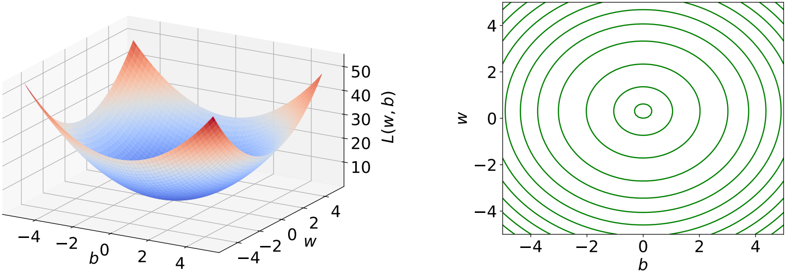

Figure 6: The surface and contour plot of training MSE of the linear regression in eq. (5).

- Fig. 6 shows the surface and contour plot of MSE of the linear regression in eq. (5):

$$\widehat{\text{SalePrice}} = b + w \times \text{LotFrontage}. \tag{5}$$

- Both linear regression and MSE are function of parameters b and w.

Optimal Solution

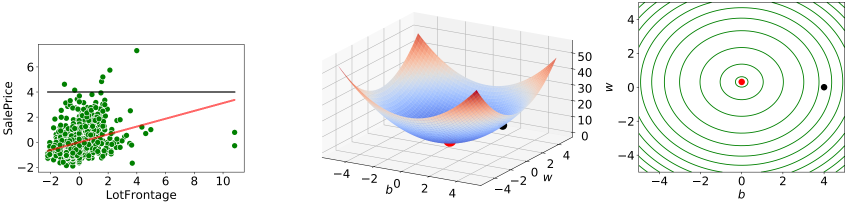

Figure 7: Two linear regression models (left) and their training MSE (right).

- The black line in the left panel of fig. 7 and the black dot in the middle and right panel correspond to the following parameter setting:
  - $b = 4$, $w = 0$
- The red line in the left panel of fig. 7 and the red dot in the middle and right panel correspond to the following parameter setting:
  - $b = 0$, $w = 0.3$
- The parameter setting that leads to the lowest MSE are sometimes called the *Optimal Solution*.
- Based on the position of the red dot in the middle and right panel of fig. 7, in this case $b = 0$ and $w = 0.3$ are the optimal solution.

Optimization

- The goal of optimization is finding the optimal solution.
- When the loss (e.g., MSE) measures how bad the model is,
  - the lower the loss, the better the model
  - as a result, optimization entails *minimizing* the loss
- When the loss (e.g., likelihood, see P2_C2_S3_Logistic_Regression) measures how good the model is,
  - the higher the loss, the better the model
  - as a result, optimization entails *maximizing* the loss
- Interestingly, we may want to maximizes the loss with respect to one part of the model and, at the same time, minimizes the loss with respect to the other part of the model.
- One such model is the *Generative Adversarial Networks* (see P3_C3_S2_Generative_Adversarial_Networks).

First-Order and Second-Order Optimization

- For linear regression, we will discuss two training methods, both belong to *First-Order Optimization*, which, as the name suggests, is based on the *first-order* derivative of the loss.
- For logistic regression (see P2_C2_S3_Logistic_Regression), we will discuss a training method that belongs to *Second-Order Optimization*, which, as the name suggests, is based on the *second-order* derivative of the loss.
- It turns out that both first-order and second-order optimization estimate the optimal solution (that leads to the minimum loss) by exploring the relationship between the loss and optimal solution.
- To capture such relationship, we need to introduce a very important concept named *Taylor Series Expansion*.

Taylor Series Expansion

- For an analytic function (i.e., a function that is infinitely differentiable), $F(\theta)$, its Taylor series expansion about a point, θ^*, is

$$\begin{aligned} F(\theta) = F(\theta^*) &+ \left.\frac{d}{d\theta}F(\theta)\right|_{\theta=\theta^*}(\theta-\theta^*) \\ &+ \frac{1}{2}\left.\frac{d^2}{d\theta^2}F(\theta)\right|_{\theta=\theta^*}(\theta-\theta^*)^2 \\ &+ \cdots \\ &+ \frac{1}{n!}\left.\frac{d^n}{d\theta^n}F(\theta)\right|_{\theta=\theta^*}(\theta-\theta^*)^n \\ &+ \cdots \end{aligned} \tag{9}$$

Here $n!$ is the factorial of n:

$$n! = \prod_{i=1}^{n} i, \tag{10}$$

and $\left.\frac{d^n}{d\theta^n}F(\theta)\right|_{\boldsymbol{\theta}=\theta^*}$ the nth-order derivative of $F(\theta)$ evaluated at point θ^*.

- To have the equal sign in eq. (9), we need to include all the derivatives of $F(\theta)$ in the equation (this is why we require $F(\theta)$ to be infinitely differentiable).

Nth-Order Approximation

- If we do not include all the derivatives of $F(\theta)$, but instead, only include up to nth-order derivative in the equation, then the following Taylor series expansion is the *Nth-Order Approximation* of $F(\theta)$:

$$\begin{aligned} F(\theta) \approx F(\theta^*) &+ \frac{d}{d\theta}F(\theta)\bigg|_{\theta=\theta^*}(\theta-\theta^*) \\ &+ \frac{1}{2}\frac{d^2}{d\theta^2}F(\theta)\bigg|_{\theta=\theta^*}(\theta-\theta^*)^2 \\ &+ \cdots \\ &+ \frac{1}{n!}\frac{d^n}{d\theta^n}F(\theta)\bigg|_{\theta=\theta^*}(\theta-\theta^*)^n. \end{aligned} \tag{11}$$

- In eq. (11), function $F(\theta)$ only has one parameter, θ. In reality, $F(\boldsymbol{\theta})$ may have multiple parameters: $F(\boldsymbol{\theta}) = F(\begin{bmatrix}\theta_1 & \theta_2 & \cdots & \theta_n\end{bmatrix}^\intercal)$.
- When this is the case, we need to write eq. (11) in matrix form:

$$\begin{aligned} F(\boldsymbol{\theta}) \approx F(\boldsymbol{\theta}^*) &+ \nabla F(\boldsymbol{\theta})^\intercal|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*}(\boldsymbol{\theta}-\boldsymbol{\theta}^*) \\ &+ \frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\theta}^*)^\intercal\nabla^2 F(\boldsymbol{\theta})\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*}(\boldsymbol{\theta}-\boldsymbol{\theta}^*) \\ &+ \cdots \end{aligned} \tag{12}$$

Gradient

- In eq. (12)

$$\begin{aligned} F(\boldsymbol{\theta}) \approx F(\boldsymbol{\theta}^*) &+ \nabla F(\boldsymbol{\theta})^\intercal|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*}(\boldsymbol{\theta}-\boldsymbol{\theta}^*) \\ &+ \frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\theta}^*)^\intercal \nabla^2 F(\boldsymbol{\theta})\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*}(\boldsymbol{\theta}-\boldsymbol{\theta}^*) \\ &+ \cdots \end{aligned} \tag{12}$$

  $\nabla F(\boldsymbol{\theta})$ is the *Gradient* of $F(\boldsymbol{\theta})$:

$$\nabla F(\boldsymbol{\theta}) = \left[\frac{\partial}{\partial\theta_1}F(\boldsymbol{\theta}) \quad \frac{\partial}{\partial\theta_2}F(\boldsymbol{\theta}) \quad \cdots \quad \frac{\partial}{\partial\theta_n}F(\boldsymbol{\theta})\right]^\intercal, \tag{13}$$

  where the ith element, $\frac{\partial}{\partial\theta_i}F(\boldsymbol{\theta})$, is the first-order *partial* derivative of function $F(\boldsymbol{\theta})$ with respect to parameter θ_i.

- Eq. (13) shows that the gradient, $\nabla F(\boldsymbol{\theta})$, measures how fast the function, $F(\boldsymbol{\theta})$, changes with the parameters, $\boldsymbol{\theta}$:
  - if $\frac{\partial}{\partial\theta_i}F(\boldsymbol{\theta}) > 0$, the higher $\frac{\partial}{\partial\theta_i}F(\boldsymbol{\theta})$ the faster $F(\boldsymbol{\theta})$ increases when θ_i increases
  - if $\frac{\partial}{\partial\theta_i}F(\boldsymbol{\theta}) < 0$, the lower $\frac{\partial}{\partial\theta_i}F(\boldsymbol{\theta})$ the faster $F(\boldsymbol{\theta})$ decreases when θ_i increases
  - if $\frac{\partial}{\partial\theta_i}F(\boldsymbol{\theta}) = 0$, $F(\boldsymbol{\theta})$ does not change when θ_i changes

First-Order Approximation

- Based on eq. (12),

$$F(\boldsymbol{\theta}) \approx F(\boldsymbol{\theta}^*) + \nabla F(\boldsymbol{\theta})^\intercal|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*} (\boldsymbol{\theta} - \boldsymbol{\theta}^*) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^\intercal \nabla^2 F(\boldsymbol{\theta})\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*} (\boldsymbol{\theta} - \boldsymbol{\theta}^*) + \cdots \tag{12}$$

  the *First-Order Approximation* (which only includes up to the first-order derivative) of $F(\boldsymbol{\theta})$ about a point $\boldsymbol{\theta}^*$ is

$$F(\boldsymbol{\theta}) \approx F(\boldsymbol{\theta}^*) + \nabla F(\boldsymbol{\theta})^\intercal|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*} (\boldsymbol{\theta} - \boldsymbol{\theta}^*). \tag{14}$$

- By replacing the function in eq. (14) ($F(\boldsymbol{\theta})$) with MSE ($\mathcal{L}(\boldsymbol{\theta})$), the first-order approximation of $\mathcal{L}(\boldsymbol{\theta})$ about the optimal solution (which minimizes $\mathcal{L}(\boldsymbol{\theta})$), $\boldsymbol{\theta}^*$, is

$$\mathcal{L}(\boldsymbol{\theta}) \approx \mathcal{L}(\boldsymbol{\theta}^*) + \nabla \mathcal{L}(\boldsymbol{\theta})^\intercal|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*} (\boldsymbol{\theta} - \boldsymbol{\theta}^*). \tag{15}$$

LinearRegression and SGDRegressor: Open the Blackbox

- Earlier we discussed two sklearn implementations of linear regression:
    - LinearRegression
    - SGDRegressor
- However, we basically treated them as a blackbox.
- Let us open the blackbox!
- It turns out that, the two tools use different methods for training:
    - LinearRegression uses a method named the *Normal Equation*
    - SGDRegressor uses a method named *Gradient Descent* (a.k.a., *Steepest Descent*)
- While the normal equation and gradient descent use different ways to find the optimal solution (which minimizes MSE), they do have something in common.
- Concretely, both of them:
    - belong to first-order optimization
    - are closely related to the first-order estimation of MSE (eq. (15)):

$$\mathcal{L}(\boldsymbol{\theta}) \approx \mathcal{L}(\boldsymbol{\theta}^*) + \nabla\mathcal{L}(\boldsymbol{\theta})^\intercal|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*} (\boldsymbol{\theta} - \boldsymbol{\theta}^*) \tag{15}$$

    - estimate the optimal solution by exploring its relationship with MSE

First-Order Condition

- The normal equation makes use of the following relationship between MSE ($\mathcal{L}(\boldsymbol{\theta})$) and the optimal solution ($\boldsymbol{\theta}^*$):

$$\nabla \mathcal{L}(\boldsymbol{\theta})^\intercal |_{\boldsymbol{\theta}=\boldsymbol{\theta}^*} = 0. \tag{16}$$

- Eq. (16) says that the first-order derivative (i.e., gradient) of MSE ($\mathcal{L}(\boldsymbol{\theta})$) evaluated at the optimal solution ($\boldsymbol{\theta}^*$) is zero.
- This equation is also called the *First-Order Condition*.
- See the proof of first-order condition in Appendix (page 67).

The Normal Equation

- Eq. (16) shows the first-order condition:

$$\nabla \mathcal{L}(\boldsymbol{\theta})^\intercal |_{\boldsymbol{\theta}=\boldsymbol{\theta}^*} = 0. \tag{16}$$

- It turns out that we can solve the first-order condition analytically to obtain the optimal solution, $\boldsymbol{\theta}^*$:

$$\boldsymbol{\theta}^* = \left(\begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}^\intercal \begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}\right)^{-1} \begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}^\intercal \mathbf{y}. \tag{17}$$

  Here:

  - $\boldsymbol{\theta}^*$ is the $(n+1) \times 1$ optimal solution vector:

$$\boldsymbol{\theta} = \begin{bmatrix}b^* & w_1^* & \cdots & w_n^*\end{bmatrix}^\intercal \tag{18}$$

  - $\mathbf{X}$ is a $m \times n$ feature matrix (with m/n being the number of samples/features):

$$\mathbf{X} = \begin{bmatrix}\mathbf{x}_1 & \cdots & \mathbf{x}_n\end{bmatrix} \tag{19}$$

  - $\mathbf{1}$ is a $m \times 1$ vector, full of 1s
  - $\mathbf{y}$ is a $m \times 1$ target vector

- Eq. (17) is called the *Normal Equation*.
- See the proof of the normal equation in Appendix (pages 68 to 70).

The Normal Equation: Code Example

- See /p2_c2_s1_linear_regression/code_example:
  1. cell

Gradient Descent: The Motivation

- Unlike the normal equation which directly gives the closed-form expression of the optimal solution, *Gradient Descent* iteratively updates the parameters to approach the optimal solution.
- **Q:** Why do we need this iterative approach to estimate the optimal solution, when we already know its closed-form expression?

Gradient Descent: The Motivation

- Unlike the normal equation which directly gives the closed-form expression of the optimal solution, *Gradient Descent* iteratively updates the parameters to approach the optimal solution.
- **Q:** Why do we need this iterative approach to estimate the optimal solution, when we already know its closed-form expression?
- **A:** It is closely related to the good practice discussed on page 17 (see the explanation of the good practice in Appendix (pages 62 to 64)).

Good practice

- When m is very large (e.g., millions):
  - it is recommended to use SGDRegressor
- When m is relatively small (e.g., thousands):
  - when $m \gg n$:
    - it is recommended to use LinearRegression (which is faster)
  - when $m \ll n$:
    - it is recommended to use SGDRegressor (which is faster)

Batch / Stochastic / Mini-Batch Gradient Descent

- Based on the amount of training data we use in each epoch (i.e., iteration) of updating the parameters, we can divide gradient descent into three categories:
  - *Batch Gradient Descent* (BGD): use all the training data
  - *Stochastic Gradient Descent* (SGD): use only one training sample
  - *Mini-batch Gradient Descent* (MBGD): use a *Mini-Batch* of training samples (something between all the training data and only one training sample)
- We can think of BGD and SGD as two extremes:
  - BGD uses as many training data as possible
  - SGD uses as few training data as possible
- In that sense, MBGD is a trade-off between the two extremes.

BGD: The Idea

- The idea of BGD is iteratively updating the parameters in such a way that MSE gets minimized.
- This is realized by the following updating rule, which repeatedly updates the parameters $\boldsymbol{\theta}$ (where $\boldsymbol{\theta} = \begin{bmatrix} b & w_1 & \cdots & w_n \end{bmatrix}^\intercal$):

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \eta_k \mathbf{p}_k. \tag{20}$$

  Here
  - $\boldsymbol{\theta}_k$ are the parameters in epoch k (the current round)
  - $\boldsymbol{\theta}_{k+1}$ are the parameters in epoch $k+1$ (the next round)
  - $\mathbf{p}_k$ is a direction in epoch k along which we search for $\boldsymbol{\theta}_{k+1}$ from $\boldsymbol{\theta}_k$
  - η_k is the *Learning Rate* (a positive scalar) that determines the step size in epoch k (how far we move from $\boldsymbol{\theta}_k$ along $\mathbf{p}_k$ to search for $\boldsymbol{\theta}_{k+1}$)
- Once we know the learning rate (η_k) and direction ($\mathbf{p}_k$), we can use eq. (20) to go from $\boldsymbol{\theta}_k$ to $\boldsymbol{\theta}_{k+1}$.

BGD: Two Analogies

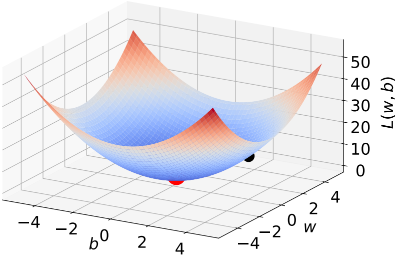

Figure 8: The surface plot of MSE of the linear regression in eq. (5).

$$\widehat{\text{SalePrice}} = b + w \times \text{LotFrontage}. \tag{5}$$

- Analogy 1: Based on fig. 8, we can think of MSE as a bowl-shaped valley.
- Analogy 2: Since the idea of BGD is iteratively updating the parameters in such a way that MSE gets minimized (e.g., from the black dot in fig. 8 to the red one), we can think of BGD as going down the bowl-shaped valley.

BGD: Direction and Learning Rate

- Based on the two analogies:
  - the direction matters because it determines where we go:
    - we could go uphill (i.e., increasing MSE) rather than downhill (decreasing MSE) if the direction were wrong
  - the learning rate matters because it determines the step size:
    - we could overshoot the bottom of the valley (i.e., the minimum of MSE) if the step size were too large
- It turns out that:
  - knowing the direction is easy
  - knowing the learning rate is difficult
- As a result, we will:
  - focus on how to find the direction for now
  - come back to how to find the learning rate later

BGD: Finding the Direction

- The goal is to find the direction, $\mathbf{p}_k$, in the updating rule in eq. (20)

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \eta_k \mathbf{p}_k. \tag{20}$$

  Here

  - $\boldsymbol{\theta}_k$ are the parameters in epoch k (the current round)
  - $\boldsymbol{\theta}_{k+1}$ are the parameters in epoch $k + 1$ (the next round)
  - $\mathbf{p}_k$ is a direction in epoch k along which we search for $\boldsymbol{\theta}_{k+1}$ from $\boldsymbol{\theta}_k$
  - η_k is the learning rate (a positive scalar) that determines the step size in epoch k (how far we move from $\boldsymbol{\theta}_k$ along $\mathbf{p}_k$ to search for $\boldsymbol{\theta}_{k+1}$)

- It turns out that $\mathbf{p}_k$ is the direction that leads to the steepest descent of MSE.
- It makes sense since, again, the idea of BGD is iteratively updating the parameters so that MSE gets minimized.

BGD: Finding the Direction

- The *Gradient*, $\mathbf{g}_k$, is the first-order derivative of MSE, $\mathcal{L}(\boldsymbol{\theta})$, with respect to the parameters in epoch k, $\boldsymbol{\theta}_k$:

$$\mathbf{g}_k = \nabla \mathcal{L}(\boldsymbol{\theta})^\intercal|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k} . \tag{21}$$

- The gradient, $\mathbf{g}_k$, is also the direction that leads to the steepest ascent of MSE, $\mathcal{L}(\boldsymbol{\theta})$.
- Since the direction, $\mathbf{p}_k$, is the direction that leads to the steepest descent of MSE, it is the opposite direction of gradient:

$$\mathbf{p}_k = -\mathbf{g}_k = - \nabla \mathcal{L}(\boldsymbol{\theta})^\intercal|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k} . \tag{22}$$

- See the proof of eq. (22) in Appendix (pages 71 and 72).

BGD: The Math

- The updating rule of BGD was given in eq. (20)

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \eta_k \mathbf{p}_k. \tag{20}$$

- Based on eq. (22)

$$\mathbf{p}_k = -\mathbf{g}_k = - \nabla \mathcal{L}(\boldsymbol{\theta})^\intercal |_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}, \tag{22}$$

  we can write eq. (20) as

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta_k \mathbf{g}_k = \boldsymbol{\theta}_k - \eta_k \nabla \mathcal{L}(\boldsymbol{\theta})^\intercal |_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}. \tag{23}$$

- By deriving the gradient, $\nabla \mathcal{L}(\boldsymbol{\theta})^\intercal |_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}$, we can write eq. (23) as

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \frac{2\eta_k}{m} \sum_{i=1}^{m} (y^i - \widehat{y^i}) \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix}^\intercal = \boldsymbol{\theta}_k + \frac{2\eta_k}{m} \begin{bmatrix} \mathbf{1} & \mathbf{X} \end{bmatrix}^\intercal (\mathbf{y} - \widehat{\mathbf{y}}). \tag{24}$$

  Here:
  - m is the number of samples in the data
  - y^i / $\widehat{y^i}$ is the real / predicted target value of sample i, where

$$\widehat{y^i} = b + w_1 x_1^i +, \ldots, + w_n x_n^i = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \begin{bmatrix} b & w_1 & \cdots & w_n \end{bmatrix}^\intercal = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \boldsymbol{\theta} \tag{25}$$

    and $\mathbf{y}$ / $\widehat{\mathbf{y}}$ is the real / predicted target vector, where

$$\widehat{\mathbf{y}} = b + w_1 \mathbf{x}_1 +, \ldots, + w_n \mathbf{x}_n = \begin{bmatrix} \mathbf{1} & \mathbf{X} \end{bmatrix} \begin{bmatrix} b & w_1 & \cdots & w_n \end{bmatrix}^\intercal = \begin{bmatrix} \mathbf{1} & \mathbf{X} \end{bmatrix} \boldsymbol{\theta} \tag{26}$$

  - $\mathbf{x}^i$ is the feature vector of sample i, and $\mathbf{X}$ the feature matrix
- See the proof of eq. (24) in Appendix (pages 73 and 74).

BGD: Code Example

- See /p2_c2_s1_linear_regression/code_example:
  1. cell

BGD: The Loss

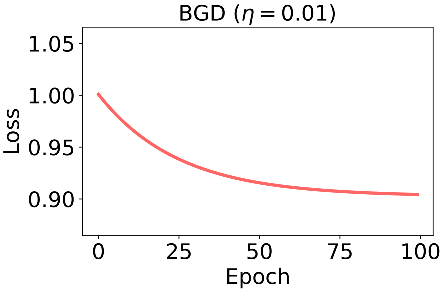

Figure 9: The training loss in BGD as a function of epoch.

- Fig. 9 shows that the training loss in BGD keeps decreasing when training proceeds.
- However, this does not mean the model keeps improving (more on this later).

BGD: Pros and Cons

- Pros:
  - utilizes parallel computing to the most extent (making it efficient)
  - does not require shuffling the data (making it even more efficient)
  - relies on gradient calculated using all the training data (easier to converge to the optimal solution)
- Cons:
  - requires processing all the training data to complete one update of the parameters (not suitable for large datasets)

Takeaway

- BGD is suitable for small datasets.

SGD: The Idea

- Unlike BGD that requires processing all the training data to complete one update of the parameters, Stochastic Gradient Descent (SGD) only requires processing one training sample to complete one update.
- This is realized by the following updating rule, which, in each epoch k, updates the parameters $\boldsymbol{\theta}$ (where $\boldsymbol{\theta} = \begin{bmatrix} b & w_1 & \cdots & w_n \end{bmatrix}^\intercal$) after processing each training sample i:

$$\boldsymbol{\theta}_k^{i+1} = \boldsymbol{\theta}_k^{i} + \eta_k \mathbf{p}_k^{i}. \tag{27}$$

  Here

  - $\boldsymbol{\theta}_k^{i}$ are the parameters in epoch k after the update using sample i
  - $\boldsymbol{\theta}_k^{i+1}$ are the parameters in epoch k after the update using sample $i + 1$
  - $\mathbf{p}_k^{i}$ is a direction in epoch k along which we search for $\boldsymbol{\theta}_k^{i+1}$ from $\boldsymbol{\theta}_k^{i}$
  - η_k is the learning rate that determines the step size in epoch k (how far we move from $\boldsymbol{\theta}_k^{i}$ along $\mathbf{p}_k^{i}$ to search for $\boldsymbol{\theta}_k^{i+1}$)

SGD: Squared Error

- Since SGD updates the parameters using only one training sample, the loss function is the *Squared Error*, $\mathcal{L}(\boldsymbol{\theta}^i)$, which measures the squared difference between the real target value and the predicted target value, with respect to sample i:

$$\mathcal{L}^i(\boldsymbol{\theta}) = (y^i - \widehat{y^i})^2. \tag{28}$$

Here:

  - y^i is the real target value of sample i
  - $\widehat{y^i}$ is the predicted target value of sample i, given in eq. (25):

$$\widehat{y^i} = b + w_1 x_1^i +, \ldots, + w_n x_n^i = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \begin{bmatrix} b & w_1 & \cdots & w_n \end{bmatrix}^\intercal = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \boldsymbol{\theta} \tag{25}$$

SGD: Finding the Direction

- The goal is to find the direction, $\mathbf{p}_k^i$, in the updating rule in eq. (27)

$$\boldsymbol{\theta}_k^{i+1} = \boldsymbol{\theta}_k^i + \eta_k \mathbf{p}_k^i. \tag{27}$$

  where $\mathbf{p}_k^i$ is a direction in epoch k along which we search for $\boldsymbol{\theta}_k^{i+1}$ from $\boldsymbol{\theta}_k^i$.

- It turns out that $\mathbf{p}_k^i$ is the direction that leads to the steepest descent of SE, $\mathcal{L}(\boldsymbol{\theta}^i)$, given in eq. (28):

$$\mathcal{L}(\boldsymbol{\theta}^i) = (y^i - \widehat{y^i})^2. \tag{28}$$

- It this sense, we can also think of SGD as an approximation of BGD:
  - BGD aims to minimize MSE, $\mathcal{L}(\boldsymbol{\theta})$, by taking the direction of the steepest descent of MSE, $\mathcal{L}(\boldsymbol{\theta})$
  - SGD also aims to minimize MSE, $\mathcal{L}(\boldsymbol{\theta})$, but by taking the direction of the steepest descent of SE, $\mathcal{L}(\boldsymbol{\theta}^i)$

SGD: Finding the Direction

- The gradient, $\mathbf{g}_k^i$, is the first-order derivative of SE, $\mathcal{L}(\boldsymbol{\theta}^i)$, with respect to the parameters in epoch k, updated using sample i, $\boldsymbol{\theta}_k^i$:

$$\mathbf{g}_k^i = \nabla \mathcal{L}(\boldsymbol{\theta}^i)^\intercal \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k^i} . \tag{29}$$

- The gradient, $\mathbf{g}_k^i$, is also the direction that leads to the steepest ascent of SE.
- Since the direction, $\mathbf{p}_k^i$, is the direction that leads to the steepest descent of SE, it is the opposite direction of gradient:

$$\mathbf{p}_k^i = -\mathbf{g}_k^i = - \nabla \mathcal{L}(\boldsymbol{\theta}^i)^\intercal \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k^i} . \tag{30}$$

- See the proof of eq. (30) in Appendix (pages 75 and 76).

SGD: The Math

- The updating rule of SGD was given in eq. (27)

$$\boldsymbol{\theta}_k^{i+1} = \boldsymbol{\theta}_k^i + \eta_k \mathbf{p}_k^i. \tag{27}$$

- Based on eq. (30)

$$\mathbf{p}_k^i = -\mathbf{g}_k^i = -\ \nabla \mathcal{L}(\boldsymbol{\theta}^i)^\intercal \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k^i}, \tag{30}$$

  we can write eq. (27) as

$$\boldsymbol{\theta}_k^{i+1} = \boldsymbol{\theta}_k^i - \eta_k \mathbf{g}_k^i = \boldsymbol{\theta}_k - \eta_k\ \nabla \mathcal{L}(\boldsymbol{\theta}^i)^\intercal \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k^i}. \tag{31}$$

- By deriving the gradient, $\nabla \mathcal{L}(\boldsymbol{\theta}^i)^\intercal \big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k^i}$, we can write eq. (31) as

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + 2\eta_k (y^i - \widehat{y^i}) \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix}^\intercal. \tag{32}$$

  Here:
  - y^i is the real target value of sample i
  - $\widehat{y^i}$ is the predicted target value of sample i, given in eq. (25)

$$\widehat{y^i} = b + w_1 x_1^i +, \ldots, + w_n x_n^i = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \begin{bmatrix} b & w_1 & \cdots & w_n \end{bmatrix}^\intercal = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \boldsymbol{\theta} \tag{25}$$

  - $\mathbf{x}^i$ is the feature vector of sample i
- See the proof of eq. (32) in Appendix (pages 77 and 78).

SGD: The Implementation

- See /p2_c2_s1_linear_regression/code_example:
  1. cell

SGD: The Loss

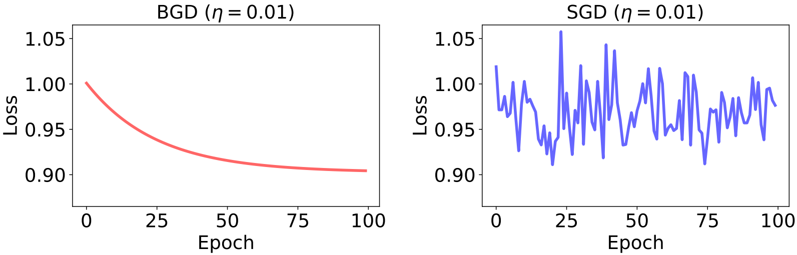

Figure 10: The training loss of BGD and SGD with learning rate 0.01.

- Fig. 10 shows that the loss of BGD is much smoother than the loss of SGD.
- This is not surprising since, as we mentioned earlier, SGD (taking the steepest descent of SE) is an approximation of BGD (taking the steepest descent of MSE).

SGD: Sklearn SGDRegressor

- See /p2_c2_s1_linear_regression/code_example:
  1. cell

SGD: Pros and Cons

- Pros:
  - allows processing only one training sample to complete one update of the parameters, making it more suitable for large datasets (compared to BGD)
  - relies on gradient calculated using each training sample, so that the loss of SGD is bumpier than the loss of BGD, making SGD easier to escape local minimum (compared to BGD)
- Cons:
  - utilizes parallel computing to the least extent, making it less efficient (than BGD)
  - requires shuffling the data in the beginning of each epoch, making it less efficient (than BGD)
  - relies on gradient calculated using each training sample, so that the loss of SGD is bumpier than the loss of BGD, making it more difficult to converge to the global minimum (compared to BGD)

BGD VS SGD

Takeaway

- BGD is suitable for small datasets.
- SGD is suitable for large datasets.

MBGD: The Idea

- Unlike BGD / SGD that uses all the training data / only one training sample to complete one update of the parameters, MBGD (Mini-Batch Gradient Descent) uses a *Mini-Batch* of training samples (something between all the training data and only one training sample) to do so.
- This is realized by the following updating rule, which, in each epoch k, updates the parameters $\boldsymbol{\theta}$ (where $\boldsymbol{\theta} = \begin{bmatrix} b & w_1 & \cdots & w_n \end{bmatrix}^\intercal$) after processing each mini-batch:

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j + \eta_k \mathbf{p}_k^j. \tag{33}$$

Here

- $\boldsymbol{\theta}_k^j$ are the parameters in epoch k after the update using mini-batch $\mathbf{mb}^j$
- $\boldsymbol{\theta}_k^{j+1}$ are the parameters in epoch k after the update using mini-batch $\mathbf{mb}_{j+1}$
- $\mathbf{p}_k^j$ is a direction in epoch k along which we search for $\boldsymbol{\theta}_k^{j+1}$ from $\boldsymbol{\theta}_k^j$
- η_k is the learning rate that determines the step size in epoch k (how far we move from $\boldsymbol{\theta}_k^j$ along $\mathbf{p}_k^j$ to search for $\boldsymbol{\theta}_k^{j+1}$)

MBGD: Mean Squared Error

- Since MBGD updates the parameters using a mini-batch of training samples, the loss function is MSE, $\mathcal{L}(\boldsymbol{\theta}^j)$, which measures the average squared difference between the real target value and the predicted target value, across the samples in mini-batch $\mathbf{mb}^j$:

$$\mathcal{L}(\boldsymbol{\theta}^j) = \frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} (y^i - \widehat{y^i})^2. \tag{34}$$

Here:

  - $|\mathbf{mb}^j|$ is the number of samples in mini-batch $\mathbf{mb}^j$
  - y^i is the real target value of sample i
  - $\widehat{y^i}$ is the predicted target value of sample i, given in eq. (25)

$$\widehat{y^i} = b + w_1 x_1^i +, \ldots, + w_n x_n^i = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \begin{bmatrix} b & w_1 & \cdots & w_n \end{bmatrix}^\intercal = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \boldsymbol{\theta} \tag{25}$$

MBGD: Finding the Direction

- The goal is to find the direction, $\mathbf{p}_k^j$, in the updating rule in eq. (33)

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j + \eta_k \mathbf{p}_k^j, \tag{33}$$

  where $\mathbf{p}_k^j$ is a direction in epoch k along which we search for $\boldsymbol{\theta}_k^{j+1}$ from $\boldsymbol{\theta}_k^j$.
- It turns out that $\mathbf{p}_k^i$ is the direction that leads to the steepest descent of MSE with respect to mini-batch $\mathbf{mb}^j$, $\mathcal{L}(\boldsymbol{\theta}^j)$.
- It this sense, MBGD is a better approximation of BGD (than SGD):
  - BGD aims to minimize MSE, $\mathcal{L}(\boldsymbol{\theta})$, by taking the direction of the steepest descent of MSE, $\mathcal{L}(\boldsymbol{\theta})$
  - SGD aims to minimize MSE, $\mathcal{L}(\boldsymbol{\theta})$, by taking the direction of the steepest descent of SE, $\mathcal{L}(\boldsymbol{\theta}^i)$
  - MBGD aims to minimize MSE, $\mathcal{L}(\boldsymbol{\theta})$, by taking the direction of the steepest descent of MSE with respect to mini-batch $\mathbf{mb}^j$, $\mathcal{L}(\boldsymbol{\theta}^j)$

MBGD: Finding the Direction

- The gradient, $\mathbf{g}_k^j$, is the first-order derivative of MSE, $\mathcal{L}(\boldsymbol{\theta}^j)$, with respect to the parameters in epoch k, updated using mini-batch $\mathbf{mb}^j$, $\boldsymbol{\theta}_k^j$:

$$\mathbf{g}_k^j = \nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k^j} . \tag{35}$$

- The gradient, $\mathbf{g}_k^j$, is also the direction that leads to the steepest ascent of $\mathcal{L}(\boldsymbol{\theta}^j)$.
- Since the direction, $\mathbf{p}_k^j$, is the direction that leads to the steepest descent of $\mathcal{L}(\boldsymbol{\theta}^j)$, it is the opposite direction of gradient:

$$\mathbf{p}_k^j = -\mathbf{g}_k^j = - \nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k^j} . \tag{36}$$

- See the proof of eq. (30) in Appendix (pages 79 and 80).

MBGD: The Math

- The updating rule of MBGD was given in eq. (33)

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j + \eta_k \mathbf{p}_k^j. \tag{33}$$

- Based on eq. (36)

$$\mathbf{p}_k^j = -\mathbf{g}_k^j = -\left.\nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k^j}, \tag{36}$$

  we can write eq. (33) as

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j - \eta_k \mathbf{g}_k^j = \boldsymbol{\theta}_k - \eta_k \left.\nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k^j}. \tag{37}$$

- By deriving the gradient, $\left.\nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k^j}$, we can write eq. (37) as

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \frac{2\eta_k}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} (y^i - \widehat{y^i}) \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix}^\intercal = \boldsymbol{\theta}_k + \frac{2\eta_k}{|\mathbf{mb}^j|} \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix}^\intercal (\mathbf{y}^j - \widehat{\mathbf{y}^j}). \tag{38}$$

  Here:
  - $|\mathbf{mb}^j|$ is the number of samples in mini-batch $\mathbf{mb}^j$
  - y^i / $\widehat{y^i}$ is the real / predicted target value of sample i, given in eq. (25)

$$\widehat{y^i} = b + w_1 x_1^i +, \ldots, +w_n x_n^i = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \begin{bmatrix} b & w_1 & \cdots & w_n \end{bmatrix}^\intercal = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \boldsymbol{\theta} \tag{25}$$

    and $\mathbf{y}^j$ / $\widehat{\mathbf{y}^j}$ is the real / predicted target vector of $\mathbf{mb}^j$, where

$$\widehat{\mathbf{y}^j} = \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix} \begin{bmatrix} b & w_1 & \cdots & w_n \end{bmatrix}^\intercal = \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix} \boldsymbol{\theta} \tag{39}$$

  - $\mathbf{x}^i$ is the feature vector of sample i, and $\mathbf{X}^j$ the feature matrix of $\mathbf{mb}^j$
- See the proof of eq. (38) in Appendix (pages 81 and 82).

MBGD: The Implementation

- See /p2_c2_s1_linear_regression/code_example:
  1. cell

Good practice

> - It is recommended to use small batches (from 2 to 32) in MBGD [2].

MBGD: The Loss

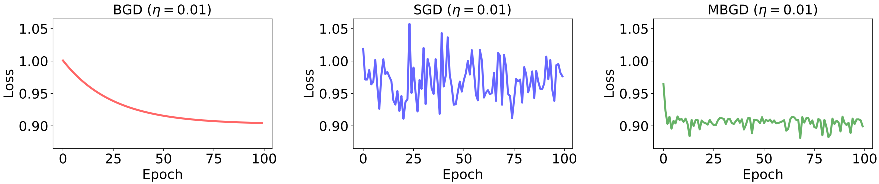

Figure 11: The training loss of BGD, SGD and MBGD with learning rate 0.01.

- Fig. 11 shows that:
  - the loss of MBGD is bumpier than the loss of BGD
  - the loss of MBGD is smoother than the loss of SGD
- This is not surprising since, as we mentioned earlier:
  - BGD takes the steepest descent of MSE
  - SGD takes the steepest descent of SE
  - MBGD takes the steepest descent of mini-batch MSE
- Compared to SGD, MBGD is a better approximation of BGD.

MBGD: Pros and Cons

- Pros:
  - allows processing only a mini-batch of training samples to complete one update of the parameters, making it more suitable for large datasets (compared to BGD)
  - relies on gradient calculated using a mini-batch of training samples, making it easier to:
    - escape local minimum (compared to BGD)
    - converge to the global minimum (compared to SGD)
  - utilizes parallel computing to some extent, making it more efficient (than SGD)
- Cons:
  - utilizes parallel computing to some extent, making it less efficient (than BGD)
  - requires shuffling the data in the beginning of each epoch, making it less efficient (than BGD)
  - relies on gradient calculated using a mini-batch of training samples, making it more difficult to:
    - converge to the global minimum (compared to BGD)
    - escape local minimum (compared to SGD)

BGD VS SGD VS MBGD

Takeaway

- BGD is suitable for small datasets.
- Both SGD and MBGD are suitable for large datasets.
- MBGD is more popular in deep learning.

Time Complexity: Page 17

- In simple words, the time complexity of an algorithm is the theoretical run time of the algorithm.
- Since the run time increases when the size of the input increases (e.g., counting to 100 takes more time than counting to 10), we would like to know the *Asymptotic Order of Growth* of the time, which measures a bound of the run time, when the input is large.
- One such bound is the upper bound, denoted by big O (see a rigorous discussion in Chapter 3 of [1]).
- For example, when we say the time complexity of an algorithm is, say $O(n^3)$ (where n is the size of the input), what we mean is that the theoretical run time is lower than n^3 (since $O(n^3)$ is an upper bound), when n is large.

LinearRegression VS SGDRegressor: Page 17

- With m being the number of training samples, n the number of features and k the number of epochs:
    - the time complexity of LinearRegression (which solves the normal equation) is

$$\max(O(n^2m), O(n^3)) = \begin{cases} O(n^2m), & \text{if} \quad m > n, \\ O(n^3), & \text{otherwise}, \end{cases} \tag{40}$$

        - if we only want to update the parameters c times, the time complexity of doing so is still the same as those in eq. (40)
        - this is because we still have to solve the normal equation, with complexity in eq. (40)
    - the time complexity of SGDRegressor (which uses SGD) updating the parameters km times is

$$O(kmn), \tag{41}$$

        - if we only want to update the parameters c times, the time complexity of doing so is

$$O(cn). \tag{42}$$

- See the proof of eqs. (40), (41) and (42) in Appendix (pages 65 and 66).

LinearRegression VS SGDRegressor: Page 17

- With the time complexities in eqs. (40), (41) and (42) , we can explain (see the text in red) the good practice on page 17.

Good practice

- When m is very large (e.g., millions):
  - it is recommended to use SGDRegressor
  - because we can use SGDRegressor to update the parameters c times to have a reasonably good estimation of the optimal solutions, whose complexity (eq. (42)) could be much lower than that of LinearRegression (eq. (40))
- When m is relatively small (e.g., thousands):
  - when $m \gg n$:
    - it is recommended to use LinearRegression (which is faster)
    - because the complexity of LinearRegression (eq. (40)) could be lower than that of SGDRegressor (eq. (41)), when $k > n$
  - when $m \ll n$:
    - it is recommended to use SGDRegressor (which is faster)
    - because the complexity of SGDRegressor (eq. (41)) could be lower than the complexity of LinearRegression (eq. (40)), when $km < n^2$

Proof of Time Complexity: Page 63

- To solve the normal equation in eq. (17)

$$\boldsymbol{\theta}^* = \left(\begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}^\intercal \begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}\right)^{-1} \begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}^\intercal \mathbf{y}, \tag{17}$$

  we need the following steps (from left to right):

  1. calculating $\begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}^\intercal \begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}$, with $O(n^2 m)$ time complexity
  2. calculating $\left(\begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}^\intercal \begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}\right)^{-1}$, with typically $O(n^3)$ time complexity
  3. calculating $\left(\begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}^\intercal \begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}\right)^{-1} \begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}^\intercal$, with $O(n^2 m)$ time complexity
  4. calculating $\left(\begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}^\intercal \begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}\right)^{-1} \begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}^\intercal \mathbf{y}$, with $O(nm)$ time complexity

- According to the *Additivity Property* of the big O notation (which says the overall time complexity of an algorithm is the largest one across the individual steps of the algorithm), the time complexity of solving the normal equation is

$$\max(O(n^2 m), O(n^3)), \tag{43}$$

  which proves the claim in eq. (40) on page 63. □

Proof of Time Complexity: Page 63

- SGD uses the updating rule in eq. (32) to update the parameters once:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + 2\eta_k(y^i - \widehat{y^i}) \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix}^\intercal . \tag{32}$$

- The time complexity of calculating $2\eta_k(y^i - \widehat{y^i}) \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix}^\intercal$ in eq. (32) is

$$O(cn). \tag{44}$$

- If we update the parameters c times, the time complexity is

$$O(n), \tag{45}$$

  which proves the claim in eq. (42).

- If there are k epochs and m training samples, we update the parameters km times, the time complexity is

$$O(kmn), \tag{46}$$

  which proves the claim in eq. (41). □

Proof of First-Order Condition: Page 29

- Eq. (15) shows the first-order approximation of $\mathcal{L}(\boldsymbol{\theta})$ about $\boldsymbol{\theta}^*$

$$\mathcal{L}(\boldsymbol{\theta}) \approx \mathcal{L}(\boldsymbol{\theta}^*) + \nabla \mathcal{L}(\boldsymbol{\theta})^\intercal|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*} (\boldsymbol{\theta} - \boldsymbol{\theta}^*). \tag{15}$$

- Let $\boldsymbol{\theta} = \boldsymbol{\theta}^* + \Delta\boldsymbol{\theta}$ (where $\Delta\boldsymbol{\theta} > 0$), then based on eq. (15) we have

$$\mathcal{L}(\boldsymbol{\theta}^* + \Delta\boldsymbol{\theta}) \approx \mathcal{L}(\boldsymbol{\theta}^*) + \nabla \mathcal{L}(\boldsymbol{\theta})^\intercal|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*} \Delta\boldsymbol{\theta}. \tag{47}$$

- Let $\boldsymbol{\theta} = \boldsymbol{\theta}^* - \Delta\boldsymbol{\theta}$ (where $\Delta\boldsymbol{\theta} > 0$), then based on eq. (15) we have

$$\mathcal{L}(\boldsymbol{\theta}^* - \Delta\boldsymbol{\theta}) \approx \mathcal{L}(\boldsymbol{\theta}^*) - \nabla \mathcal{L}(\boldsymbol{\theta})^\intercal|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*} \Delta\boldsymbol{\theta}. \tag{48}$$

- Since $\boldsymbol{\theta}^*$ is the optimal solution (that minimizes $\mathcal{L}(\boldsymbol{\theta})$), we have

$$\mathcal{L}(\boldsymbol{\theta}^* + \Delta\boldsymbol{\theta}) - \mathcal{L}(\boldsymbol{\theta}^*) \geq 0 \leq \mathcal{L}(\boldsymbol{\theta}^* - \Delta\boldsymbol{\theta}) - \mathcal{L}(\boldsymbol{\theta}^*). \tag{49}$$

- Based on eqs. (47) to (49) we have

$$\nabla \mathcal{L}(\boldsymbol{\theta})^\intercal|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*} \geq 0 \geq \nabla \mathcal{L}(\boldsymbol{\theta})^\intercal|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*}, \tag{50}$$

which proves the first-order condition in eq. (16) on page 29. □

Proof of the Normal Equation: Page 30

- Eq. (7) shows the loss function (MSE) of linear regression:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}(y^i - \widehat{y^i})^2 \quad \text{where} \quad \widehat{y^i} = b + w_1 x_1^i +, \ldots, +w_n x_n^i. \tag{7}$$

- We can write eq. (7) in matrix form:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{m}(\mathbf{y} - \widehat{\mathbf{y}})^\intercal(\mathbf{y} - \widehat{\mathbf{y}}). \tag{51}$$

- Eq. (16) shows the first-order condition:

$$\nabla\mathcal{L}(\boldsymbol{\theta})^\intercal|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*} = 0. \tag{16}$$

- By substituting eq. (51) into eq. (16) we have

$$\nabla\mathcal{L}(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*} = \nabla\frac{1}{m}(\mathbf{y} - \widehat{\mathbf{y}})^\intercal(\mathbf{y} - \widehat{\mathbf{y}})\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*} = 0. \tag{52}$$

Proof of the Normal Equation: Page 30

- Since eq. (52) still holds after removing $\frac{1}{m}$, then

$$\nabla(\mathbf{y} - \widehat{\mathbf{y}})^\intercal(\mathbf{y} - \widehat{\mathbf{y}})|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*} = 0. \tag{53}$$

Here $\widehat{\mathbf{y}}$ is the predicted target vector, given in eq. (2):

$$\widehat{\mathbf{y}} = \begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}\boldsymbol{\theta}. \tag{2}$$

- By substituting eq. (2) into eq. (53) we have

$$\nabla\left(\mathbf{y} - \begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}\boldsymbol{\theta}\right)^\intercal\left(\mathbf{y} - \begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}\boldsymbol{\theta}\right)\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*} = 0, \tag{54}$$

which can be written as

$$\nabla\Big(\mathbf{y}^\intercal\mathbf{y} - 2\boldsymbol{\theta}^\intercal\begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}^\intercal\mathbf{y} + \boldsymbol{\theta}^\intercal\begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}^\intercal\begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}\boldsymbol{\theta}\Big)\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*} = 0. \tag{55}$$

Proof of the Normal Equation: Page 30

- By calculating the derivation in eq. (55) we have

$$-2\begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}^\intercal \mathbf{y} + 2\begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}^\intercal \begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix} \boldsymbol{\theta}^* = 0, \tag{56}$$

  which can be written as

$$\begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}^\intercal \begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix} \boldsymbol{\theta}^* = \begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}^\intercal \mathbf{y}. \tag{57}$$

- By multiplying $\left(\begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}^\intercal \begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}\right)^{-1}$ on both sides of eq. (57) we have

$$\boldsymbol{\theta}^* = \left(\begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}^\intercal \begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}\right)^{-1} \begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}^\intercal \mathbf{y}, \tag{58}$$

  which proves the normal equation in eq. (17) on page 30. □

Proof of the Search Direction: Page 38

- The first-order approximation of MSE, $\mathcal{L}(\boldsymbol{\theta})$, about the optimal solution (which minimizes $\mathcal{L}(\boldsymbol{\theta})$), $\boldsymbol{\theta}^*$, is given in eq. (15)

$$\mathcal{L}(\boldsymbol{\theta}) \approx \mathcal{L}(\boldsymbol{\theta}^*) + \nabla \mathcal{L}(\boldsymbol{\theta})^\intercal|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*} (\boldsymbol{\theta} - \boldsymbol{\theta}^*). \tag{15}$$

- Similar to eq. (15), the first-order approximation of MSE in epoch $k+1$, $\mathcal{L}(\boldsymbol{\theta}_{k+1})$, about the parameters in epoch k, $\boldsymbol{\theta}_k$, is

$$\mathcal{L}(\boldsymbol{\theta}_{k+1}) \approx \mathcal{L}(\boldsymbol{\theta}_k) + \nabla \mathcal{L}(\boldsymbol{\theta})^\intercal|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k} (\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k) = \mathcal{L}(\boldsymbol{\theta}_k) + \mathbf{g}_k(\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k). \tag{59}$$

- In BGD, the parameters are updated iteratively using the updating rule in eq. (20)

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \eta_k \mathbf{p}_k, \tag{20}$$

which can be written as

$$\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k = \eta_k \mathbf{p}_k. \tag{60}$$

- By substituting eq. (60) into eq. (59) we have

$$\mathcal{L}(\boldsymbol{\theta}_{k+1}) \approx \mathcal{L}(\boldsymbol{\theta}_k) + \eta_k \mathbf{g}_k \mathbf{p}_k, \tag{61}$$

which can be written as

$$\mathcal{L}(\boldsymbol{\theta}_{k+1}) - \mathcal{L}(\boldsymbol{\theta}_k) \approx \eta_k \mathbf{g}_k \mathbf{p}_k. \tag{62}$$

Proof of the Search Direction: Page 38

- Since $\mathbf{g}_k \mathbf{p}_k$ in eq. (62) is the dot product between $\mathbf{g}_k$ and $\mathbf{p}_k$, we can write eq. (62) as

$$\mathcal{L}(\boldsymbol{\theta}_{k+1}) - \mathcal{L}(\boldsymbol{\theta}_k) \approx \eta_k \|\mathbf{g}_k\| \|\mathbf{p}_k\| \cos(\alpha). \tag{63}$$

  Here

  - η_k is the learning rate in epoch k
  - $\|\mathbf{g}_k\|$ and $\|\mathbf{p}_k\|$ are the magnitude of $\mathbf{g}_k$ and $\mathbf{p}_k$
  - α is the angle between $\mathbf{g}_k$ and $\mathbf{p}_k$

- Since the goal of BGD is finding the optimal solution, $\boldsymbol{\theta}^*$, which minimizes MSE, we want $\boldsymbol{\theta}_{k+1}$ to be as close to $\boldsymbol{\theta}^*$ as possible.
- In other words, we want to minimize the difference in eq. (63), $\mathcal{L}(\boldsymbol{\theta}_{k+1}) - \mathcal{L}(\boldsymbol{\theta}_k)$.
- Since the difference is the minimum when $\alpha = 180°$:

$$\arg\min_{\alpha} \eta_k \|\mathbf{g}_k\| \|\mathbf{p}_k\| \cos(\alpha) = 180°, \tag{64}$$

  $\mathbf{p}_k$ takes the opposite direction of $\mathbf{g}_k$.

- Assume $\mathbf{p}_k$ and $\mathbf{g}_k$ have the same magnitude (which is fine due to η_k), we have

$$\mathbf{p}_k = -\mathbf{g}_k, \tag{65}$$

  which proves the claim in eq. (22) on page 38. □

Proof of the Updating Rule: Page 39

- We can write the gradient (i.e., first-order derivative) of MSE, $\nabla \mathcal{L}(\boldsymbol{\theta})^\intercal$, as

$$\nabla \mathcal{L}(\boldsymbol{\theta})^\intercal = \left[\tfrac{\partial}{\partial b} \mathcal{L}(\boldsymbol{\theta}) \quad \tfrac{\partial}{\partial w_1} \mathcal{L}(\boldsymbol{\theta}) \quad \cdots \quad \tfrac{\partial}{\partial w_n} \mathcal{L}(\boldsymbol{\theta}) \right]^\intercal, \tag{66}$$

where MSE, $\mathcal{L}(\boldsymbol{\theta})$, is given in eq. (7)

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} (y^i - \widehat{y^i})^2 \quad \text{where} \quad \widehat{y^i} = b + w_1 x_1^i +, \ldots, +w_n x_n^i. \tag{7}$$

- Based on eq. (7), we can write $\frac{\partial}{\partial b} \mathcal{L}(\boldsymbol{\theta})$ as

$$\begin{aligned} \frac{\partial}{\partial b} \mathcal{L}(\boldsymbol{\theta}) &= \frac{\partial}{\partial b} \left(\frac{1}{m} \sum_{i=1}^{m} \left(y^i - \widehat{y^i} \right)^2 \right), \\ &= \frac{2}{m} \sum_{i=1}^{m} \left(y^i - \widehat{y^i} \right) \cdot \frac{\partial}{\partial b} \left(y^i - \widehat{y^i} \right), \\ &= \frac{2}{m} \sum_{i=1}^{m} \left(y^i - \widehat{y^i} \right) \cdot \frac{\partial}{\partial b} \left(y^i - (b + w_1 x_1^i +, \ldots, +w_n x_n^i) \right), \\ &= \frac{2}{m} \sum_{i=1}^{m} \left(y^i - \widehat{y^i} \right) \cdot (-1). \end{aligned} \tag{67}$$

Proof of the Updating Rule: Page 39

- Based on eq. (7), we can write $\frac{\partial}{\partial w_j}\mathcal{L}(\boldsymbol{\theta})$ as

$$\begin{aligned}\frac{\partial}{\partial w_j}\mathcal{L}(\boldsymbol{\theta}) &= \frac{\partial}{\partial w_j}\left(\frac{1}{m}\sum_{i=1}^{m}\left(y^i - \widehat{y^i}\right)^2\right),\\ &= \frac{2}{m}\sum_{i=1}^{m}\left(y^i - \widehat{y^i}\right)\cdot\frac{\partial}{\partial w_j}\left(y^i - (b + w_1 x_1^i +, \dots, +w_n x_n^i)\right), \qquad (68)\\ &= \frac{2}{m}\sum_{i=1}^{m}\left(y^i - \widehat{y^i}\right)\cdot(-x_j^i).\end{aligned}$$

- By substituting eqs. (67) and (68) into eq. (66), we have

$$\nabla\mathcal{L}(\boldsymbol{\theta})^\intercal = -\frac{2}{m}\sum_{i=1}^{m}(y^i - \widehat{y^i})\begin{bmatrix}1 & \mathbf{x}^i\end{bmatrix}^\intercal = -\frac{2}{m}\begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}^\intercal(\mathbf{y} - \widehat{\mathbf{y}}). \qquad (69)$$

- By substituting eq. (69) into eq. (23), we have

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \frac{2\eta_k}{m}\sum_{i=1}^{m}(y^i - \widehat{y^i})\begin{bmatrix}1 & \mathbf{x}^i\end{bmatrix}^\intercal = \boldsymbol{\theta}_k + \frac{2\eta_k}{m}\begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}^\intercal(\mathbf{y} - \widehat{\mathbf{y}}), \qquad (70)$$

which proves the claim in eq. (24) on page 39. □

Proof of the Search Direction: Page 46

- Similar to eq. (15), the first-order approximation of SE in epoch k after the update using sample $i+1$, $\mathcal{L}(\boldsymbol{\theta}_k^{i+1})$, about the parameters in epoch k after the update using sample i, $\boldsymbol{\theta}_k^i$, is

$$\mathcal{L}(\boldsymbol{\theta}_k^{i+1}) \approx \mathcal{L}(\boldsymbol{\theta}_k^i) + \nabla\mathcal{L}(\boldsymbol{\theta})^\intercal|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k^i} (\boldsymbol{\theta}_k^{i+1} - \boldsymbol{\theta}_k^i) = \mathcal{L}(\boldsymbol{\theta}_k^i) + \mathbf{g}_k^i(\boldsymbol{\theta}_k^{i+1} - \boldsymbol{\theta}_k^i). \tag{71}$$

- In SGD, the parameters are updated iteratively using the updating rule in eq. (27)

$$\boldsymbol{\theta}_k^{i+1} = \boldsymbol{\theta}_k^i + \eta_k \mathbf{p}_k^i. \tag{27}$$

which can be written as

$$\boldsymbol{\theta}_k^{i+1} - \boldsymbol{\theta}_k^i = \eta_k \mathbf{p}_k^i. \tag{72}$$

- By substituting eq. (72) into eq. (71) we have

$$\mathcal{L}(\boldsymbol{\theta}_k^{i+1}) \approx \mathcal{L}(\boldsymbol{\theta}_k^i) + \eta_k \mathbf{g}_k^i \mathbf{p}_k^i, \tag{73}$$

which can be written as

$$\mathcal{L}(\boldsymbol{\theta}_k^{i+1}) - \mathcal{L}(\boldsymbol{\theta}_k^i) \approx \eta_k \mathbf{g}_k^i \mathbf{p}_k^i. \tag{74}$$

Proof of the Search Direction: Page 46

- Since $\mathbf{g}_k^i \mathbf{p}_k^i$ in eq. (74) is the dot product between $\mathbf{g}_k^i$ and $\mathbf{p}_k^i$, we can write eq. (74) as

$$\mathcal{L}(\boldsymbol{\theta}_k^{i+1}) - \mathcal{L}(\boldsymbol{\theta}_k^i) \approx \eta_k \|\mathbf{g}_k^i\| \|\mathbf{p}_k^i\| \cos(\alpha). \tag{75}$$

  Here

  - η_k is the learning rate in epoch k
  - $\|\mathbf{g}_k^i\|$ and $\|\mathbf{p}_k^i\|$ are the magnitude of $\mathbf{g}_k^i$ and $\mathbf{p}_k^i$
  - α is the angle between $\mathbf{g}_k^i$ and $\mathbf{p}_k^i$
- Since the goal of SGD is finding the optimal solution, $\boldsymbol{\theta}^*$, which minimizes MSE, we want $\boldsymbol{\theta}_k^i$ to be as close to $\boldsymbol{\theta}^*$ as possible.
- In other words, we want to minimize the difference in eq. (75), $\mathcal{L}(\boldsymbol{\theta}_k^{i+1}) - \mathcal{L}(\boldsymbol{\theta}_k^i)$.
- Since the difference is the minimum when $\alpha = 180°$:

$$\arg\min_{\alpha} \eta_k \|\mathbf{g}_k^i\| \|\mathbf{p}_k^i\| \cos(\alpha) = 180°, \tag{76}$$

  $\mathbf{p}_k^i$ takes the opposite direction of $\mathbf{g}_k^i$.
- Assume $\mathbf{p}_k^i$ and $\mathbf{g}_k^i$ have the same magnitude (which is fine due to η_k), we have

$$\mathbf{p}_k^i = -\mathbf{g}_k^i, \tag{77}$$

  which proves the claim in eq. (30) on page 46. □

Proof of the Updating Rule: Page 47

- We can write the gradient (i.e., first-order derivative) of SE, $\nabla \mathcal{L}(\boldsymbol{\theta}^i)^\intercal$, as

$$\nabla \mathcal{L}(\boldsymbol{\theta}^i)^\intercal = \begin{bmatrix} \frac{\partial}{\partial b}\mathcal{L}(\boldsymbol{\theta}^i) & \frac{\partial}{\partial w_1}\mathcal{L}(\boldsymbol{\theta}^i) & \cdots & \frac{\partial}{\partial w_n}\mathcal{L}(\boldsymbol{\theta}^i) \end{bmatrix}^\intercal, \tag{78}$$

where SE, $\mathcal{L}(\boldsymbol{\theta}^i)$, is given in eq. (28)

$$\mathcal{L}^i(\boldsymbol{\theta}) = (y^i - \widehat{y^i})^2 \quad \text{where} \quad \widehat{y^i} = b + w_1 x_1^i +, \ldots, +w_n x_n^i. \tag{28}$$

- Based on eq. (28), we can write $\frac{\partial}{\partial b}\mathcal{L}(\boldsymbol{\theta}^i)$ as

$$\begin{aligned}
\frac{\partial}{\partial b}\mathcal{L}(\boldsymbol{\theta}^i) &= \frac{\partial}{\partial b}\left(y^i - \widehat{y^i}\right)^2, \\
&= 2\left(y^i - \widehat{y^i}\right) \cdot \frac{\partial}{\partial b}\left(y^i - \widehat{y^i}\right), \\
&= 2\left(y^i - \widehat{y^i}\right) \cdot \frac{\partial}{\partial b}\left(y^i - (b + w_1 x_1^i +, \ldots, +w_n x_n^i)\right), \\
&= 2\left(y^i - \widehat{y^i}\right) \cdot (-1).
\end{aligned} \tag{79}$$

Proof of the Updating Rule: Page 47

- Based on eq. (28), we can write $\frac{\partial}{\partial w_j}\mathcal{L}(\boldsymbol{\theta}^i)$ as

$$\begin{aligned}\frac{\partial}{\partial w_j}\mathcal{L}(\boldsymbol{\theta}^i) &= \frac{\partial}{\partial w_j}\left(y^i - \widehat{y^i}\right)^2, \\ &= 2\left(y^i - \widehat{y^i}\right) \cdot \frac{\partial}{\partial w_j}\left(y^i - (b + w_1 x_1^i +, \ldots, +w_n x_n^i)\right), \\ &= 2\left(y^i - \widehat{y^i}\right) \cdot (-x_j^i).\end{aligned} \tag{80}$$

- By substituting eqs. (79) and (80) into eq. (78), we have

$$\nabla\mathcal{L}(\boldsymbol{\theta}^i)^\intercal = -2(y^i - \widehat{y^i})\begin{bmatrix}1 & \mathbf{x}^i\end{bmatrix}^\intercal. \tag{81}$$

- By substituting eq. (81) into eq. (31), we have

$$\boldsymbol{\theta}_k^{i+1} = \boldsymbol{\theta}_k^i + 2\eta_k(y^i - \widehat{y^i})\begin{bmatrix}1 & \mathbf{x}^i\end{bmatrix}^\intercal, \tag{82}$$

which proves the claim in eq. (32) on page 47. □

Proof of the Search Direction: Page 56

- Similar to eq. (15), the first-order approximation of MSE in epoch k after the update using mini-batch $\mathbf{mb}_{j+1}$, $\mathcal{L}(\boldsymbol{\theta}_k^{j+1})$, about the parameters in epoch k after the update using mini-batch $\mathbf{mb}^j$, $\boldsymbol{\theta}_k^j$, is

$$\mathcal{L}(\boldsymbol{\theta}_k^{j+1}) \approx \mathcal{L}(\boldsymbol{\theta}_k^j) + \nabla\mathcal{L}(\boldsymbol{\theta})^\intercal|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k^i} (\boldsymbol{\theta}_k^{j+1} - \boldsymbol{\theta}_k^j) = \mathcal{L}(\boldsymbol{\theta}_k^j) + \mathbf{g}_k^j(\boldsymbol{\theta}_k^{j+1} - \boldsymbol{\theta}_k^j). \tag{83}$$

- In MBGD, the parameters are updated iteratively using the updating rule in eq. (33)

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j + \eta_k \mathbf{p}_k^j. \tag{33}$$

which can be written as

$$\boldsymbol{\theta}_k^{j+1} - \boldsymbol{\theta}_k^j = \eta_k \mathbf{p}_k^j. \tag{84}$$

- By substituting eq. (84) into eq. (83) we have

$$\mathcal{L}(\boldsymbol{\theta}_k^{j+1}) \approx \mathcal{L}(\boldsymbol{\theta}_k^j) + \eta_k \mathbf{g}_k^j \mathbf{p}_k^j, \tag{85}$$

which can be written as

$$\mathcal{L}(\boldsymbol{\theta}_k^{j+1}) - \mathcal{L}(\boldsymbol{\theta}_k^j) \approx \eta_k \mathbf{g}_k^j \mathbf{p}_k^j. \tag{86}$$

Proof of the Search Direction: Page 56

- Since $\mathbf{g}_k^j \mathbf{p}_k^j$ in eq. (86) is the dot product between $\mathbf{g}_k^j$ and $\mathbf{p}_k^j$, we can write eq. (86) as

$$\mathcal{L}(\boldsymbol{\theta}_k^{j+1}) - \mathcal{L}(\boldsymbol{\theta}_k^j) \approx \eta_k \|\mathbf{g}_k^j\| \|\mathbf{p}_k^j\| \cos(\alpha). \tag{87}$$

  Here

  - η_k is the learning rate in epoch k
  - $\|\mathbf{g}_k^j\|$ and $\|\mathbf{p}_k^j\|$ are the magnitude of $\mathbf{g}_k^j$ and $\mathbf{p}_k^j$
  - α is the angle between $\mathbf{g}_k^j$ and $\mathbf{p}_k^j$
- Since the goal of SGD is finding the optimal solution, $\boldsymbol{\theta}^*$, which minimizes MSE, we want $\boldsymbol{\theta}_k^j$ to be as close to $\boldsymbol{\theta}^*$ as possible.
- In other words, we want to minimize the difference in eq. (87), $\mathcal{L}(\boldsymbol{\theta}_k^{j+1}) - \mathcal{L}(\boldsymbol{\theta}_k^j)$.
- Since the difference is the minimum when $\alpha = 180°$:

$$\arg\min_{\alpha} \eta_k \|\mathbf{g}_k^j\| \|\mathbf{p}_k^j\| \cos(\alpha) = 180°, \tag{88}$$

  $\mathbf{p}_k^j$ takes the opposite direction of $\mathbf{g}_k^j$.
- Assume $\mathbf{p}_k^j$ and $\mathbf{g}_k^j$ have the same magnitude (which is fine due to η_k), we have

$$\mathbf{p}_k^j = -\mathbf{g}_k^j, \tag{89}$$

  which proves the claim in eq. (36) on page 56. □

Proof of the Updating Rule: Page 57

- We can write the gradient (i.e., first-order derivative) of MSE with respect to mini-batch $\mathbf{mb}^j$, $\nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal$, as

$$\nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal = \left[\frac{\partial}{\partial b} \mathcal{L}(\boldsymbol{\theta}^j) \quad \frac{\partial}{\partial w_1} \mathcal{L}(\boldsymbol{\theta}^j) \quad \cdots \quad \frac{\partial}{\partial w_n} \mathcal{L}(\boldsymbol{\theta}^j) \right]^\intercal, \tag{90}$$

where MSE, $\mathcal{L}(\boldsymbol{\theta}^j)$, is given in eq. (34)

$$\mathcal{L}(\boldsymbol{\theta}^j) = \frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} (y^i - \widehat{y^i})^2. \tag{34}$$

- Based on eq. (34), we can write $\frac{\partial}{\partial b} \mathcal{L}(\boldsymbol{\theta}^j)$ as

$$\begin{aligned} \frac{\partial}{\partial b} \mathcal{L}(\boldsymbol{\theta}^j) &= \frac{\partial}{\partial b} \left(\frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left(y^i - \widehat{y^i} \right)^2 \right), \\ &= \frac{2}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left(y^i - \widehat{y^i} \right) \cdot \frac{\partial}{\partial b} \left(y^i - \widehat{y^i} \right), \\ &= \frac{2}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left(y^i - \widehat{y^i} \right) \cdot \frac{\partial}{\partial b} \left(y^i - (b + w_1 x_1^i +, \ldots, + w_n x_n^i) \right), \\ &= \frac{2}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left(y^i - \widehat{y^i} \right) \cdot (-1). \end{aligned} \tag{91}$$

Proof of the Updating Rule: Page 57

- Based on eq. (34), we can write $\frac{\partial}{\partial w_j}\mathcal{L}(\boldsymbol{\theta}^i)$ as

$$\begin{aligned}\frac{\partial}{\partial w_j}\mathcal{L}(\boldsymbol{\theta}^j) &= \frac{\partial}{\partial w_j}\left(\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(y^i - \widehat{y^i}\right)^2\right),\\ &= \frac{2}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(y^i - \widehat{y^i}\right)\cdot\frac{\partial}{\partial w_j}\left(y^i - \widehat{y^i}\right),\\ &= \frac{2}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(y^i - \widehat{y^i}\right)\cdot\frac{\partial}{\partial w_j}\left(y^i - (b + w_1x_1^i +,\ldots,+w_nx_n^i)\right),\\ &= \frac{2}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}\left(y^i - \widehat{y^i}\right)\cdot(-x_j^i).\end{aligned} \tag{92}$$

- By substituting eqs. (91) and (92) into eq. (90), we have

$$\nabla\mathcal{L}(\boldsymbol{\theta}^i)^\intercal = -\frac{2}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}(y^i - \widehat{y^i})\begin{bmatrix}1 & \mathbf{x}^i\end{bmatrix}^\intercal = -\frac{2}{|\mathbf{mb}^j|}\begin{bmatrix}\mathbf{1} & \mathbf{X}^j\end{bmatrix}^\intercal(\mathbf{y}^j - \widehat{\mathbf{y}^j}). \tag{93}$$

- By substituting eq. (93) into eq. (37), we have

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \frac{2\eta_k}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}(y^i - \widehat{y^i})\begin{bmatrix}1 & \mathbf{x}^i\end{bmatrix}^\intercal = \boldsymbol{\theta}_k + \frac{2\eta_k}{|\mathbf{mb}^j|}\begin{bmatrix}\mathbf{1} & \mathbf{X}^j\end{bmatrix}^\intercal(\mathbf{y}^j - \widehat{\mathbf{y}^j}), \tag{94}$$

which proves the claim in eq. (38) on page 57. □

References

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein.
Introduction to Algorithms.
MIT press, 2009.

D. Masters and C. Luschi.
Revisiting Small Batch Training for Deep Neural Networks.
arXiv preprint arXiv:1804.07612, 2018.