# Popular Machine Learning Methods: Idea, Practice and Math

## Part 2, Chapter 2, Section 2: Training Shallow Models

Yuxiao Huang

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

Spring 2021

# 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)
  - **PRML** Pattern Recognition and Machine Learning
  - **NND** Neural Network Design (2nd Edition)
  - **LFD** Learning From Data
  - **RL** Reinforcement Learning: An Introduction (2nd Edition)

- 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_training_shallow_models/code_example
- See related case study of Kaggle Competition in github repository: /p2_c2_s2_training_shallow_models/case_study

# Table of Contents

# Learning Objectives: Expectation

- It is expected to understand
  - the idea of Bias and Variance
  - the idea of Expected Test Error and its decomposition
  - the idea of Bias-Variance Tradeoff
  - the idea of Underfitting and Overfitting
  - the idea of Learning Curve
  - the takeaway of signs of underfitting and overfitting
  - the good practice for handling underfitting and overfitting
  - the idea of Regularization
  - the idea and implementation of popular regularization methods, including:
    - Lasso (a.k.a., L1 regularization)
    - Ridge (a.k.a., L2 regularization)
    - Elastic net
  - the good practice for using lasso / ridge / elastic net
  - the idea of Hyperparameter Tuning
  - the idea and usage of sklearn hyperparameter tuning tools, including:
    - `GridSearchCV`
    - `RandomizedSearchCV`
  - the good practice for using `GridSearchCV` and `RandomizedSearchCV`
  - the idea and implementation of model selection

# Learning Objectives: Recommendation

- It is recommended to understand
  - the math of the decomposition of expected test error
  - the math of popular regularization methods, including:
    - lasso
    - ridge
    - elastic net

# Kaggle Competition: Predicting House Price

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

- 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

# Motivation

- In /p2_c2_s1_linear_regression we discussed two methods for training linear regression:
  - the normal equation, which solves the optimal solution analytically
  - gradient descent, which estimates the optimal solution iteratively
- While the two methods are different in many ways, there is one thing in common: they both train linear regression by minimizing the training error (e.g., mean squared error).
- Unfortunately, if we only cared about minimizing the training error, we might learn a model that:
  - on the one hand, has low training error (i.e., performs well on training data)
  - but on the other hand, has high test error (i.e., generalizes poorly on test data)
- The *Learning Theory* tells us:
  - why this is the case
  - and more importantly, what we can be do to address this problem

# Bias

- In learning theory, *Bias* measures the average difference between the predicted target value and real target value:

$$\mathrm{Bias}(\widehat{\mathbf{y}}, \mathbf{y}) = E\left[\widehat{\mathbf{y}} - \mathbf{y}\right] = \frac{\sum_{i=1}^{m} (\widehat{y^i} - y^i)}{m}. \tag{1}$$

Here:

- $\widehat{\mathbf{y}}$ is the predicted target vector
- $\mathbf{y}$ is the real target vector
- $E\left[\widehat{\mathbf{y}} - \mathbf{y}\right]$ is the average of $\widehat{\mathbf{y}} - \mathbf{y}$
- $m$ is the number of samples in the data
- $\widehat{y^i}$ is the predicted target value of sample $i$
- $y^i$ is the real target value of sample $i$

## Variance

- Unlike bias that captures the difference between the predicted target value and real target value, *Variance* measures the difference between the predicted value themselves.
- More formally, variance is the average squared difference between the predicted target value and their mean:

$$\mathrm{Var}(\widehat{\mathbf{y}}) = E\left[(\widehat{\mathbf{y}} - E[\widehat{\mathbf{y}}])^2\right] = \frac{\sum_{i=1}^{m}(\widehat{y^i} - \frac{\sum_{i=1}^{m}\widehat{y^i}}{m})^2}{m}. \tag{2}$$

Here:

- $\widehat{\mathbf{y}}$ is the predicted target vector
- $E[\widehat{\mathbf{y}}]$ is the mean of the predicted target vector
- $E\left[(\widehat{\mathbf{y}} - E[\widehat{\mathbf{y}}])^2\right]$ is the average of $(\widehat{\mathbf{y}} - E[\widehat{\mathbf{y}}])^2$
- $m$ is the number of samples in the data
- $\widehat{y^i}$ is the predicted target value of sample $i$

## Expected Test Error

- Given a test sample, $\begin{bmatrix}\mathbf{x} & y\end{bmatrix}$, we:
  - draw $m$ training sets, $\begin{bmatrix}\mathbf{X}_1 & y_1\end{bmatrix}$ , ..., $\begin{bmatrix}\mathbf{X}_m & y_m\end{bmatrix}$, where the test sample and each training set come from the same distribution
  - train the same model $H$ on each training set and obtain $m$ models, $H_1, \ldots, H_m$
- **Q:** What is the expected test error (across the $m$ models)?

# Decomposition of Expected Test Error

- **A:** It turns out that we can decompose the expected test error (across the $m$ models) into the sum of squared bias and variance:

$$\underbrace{E\left[(\widehat{\mathbf{y}} - \mathbf{y})^2\right]}_{\text{Expected test error}} = \underbrace{(E[\widehat{\mathbf{y}} - \mathbf{y}])^2}_{\text{Bias}^2} + \underbrace{E\left[(\widehat{\mathbf{y}} - E[\widehat{\mathbf{y}}])^2\right]}_{\text{Variance}}. \tag{3}$$

  Here:
  - $\widehat{\mathbf{y}}$ / $\mathbf{y}$ is a $m \times 1$ predicted / real target vector across the $m$ models:
  $$\widehat{\mathbf{y}} = \begin{bmatrix}\widehat{y^1} & \dots & \widehat{y^m}\end{bmatrix}^\intercal \quad \text{and} \quad \mathbf{y} = \begin{bmatrix}y^1 & \dots & y^m\end{bmatrix}^\intercal, \tag{4}$$
  where $\widehat{y^i}$ is predicted by model $H_i$ and $y^i = y$ (with $y$ being the target value in the test sample)
  - bias is given in eq. (1)
  $$\text{Bias}(\widehat{\mathbf{y}}, \mathbf{y}) = E[\widehat{\mathbf{y}} - \mathbf{y}] = \frac{\sum_{i=1}^{m}(\widehat{y^i} - y^i)}{m} \tag{1}$$
  - variance is given in eq. (2)
  $$\text{Var}(\widehat{\mathbf{y}}) = E\left[(\widehat{\mathbf{y}} - E[\widehat{\mathbf{y}}])^2\right] = \frac{\sum_{i=1}^{m}(\widehat{y^i} - \frac{\sum_{i=1}^{m}\widehat{y^i}}{m})^2}{m} \tag{2}$$
- See the proof of eq. (3) in Appendix (pages 51 to 53).

# Bias-Variance Tradeoff

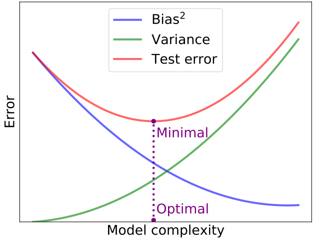


Figure 2: The bias-variance tradeoff.

- Fig. 2 shows the squared bias, variance and test error as a function of model complexity.
- Concretely, when the model complexity goes up
  - the squared bias goes down
  - the variance goes up
  - the test error, which can be decomposed into the sum of squared bias and variance (as shown in eq. (3)), first goes down then goes up
- The above relationship (between the squared bias / variance / test error and model complexity) is called the *Bias-Variance Tradeoff*.

# Underfitting VS Overfitting

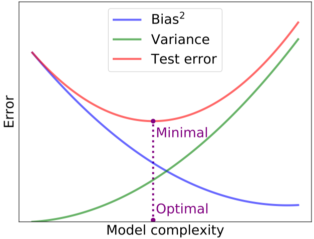


Figure 2: The bias-variance tradeoff.

- Fig. 2 also shows the minimal test error and the corresponding optimal model complexity.
- When model complexity < the optimal complexity, we call this *Underfitting*.
- When model complexity > the optimal complexity, we call this *Overfitting*.
- **Q:** Since the optimal complexity is usually unknown, how can we tell when we are underfitting and when we are overfitting?

# Underfitting VS Overfitting

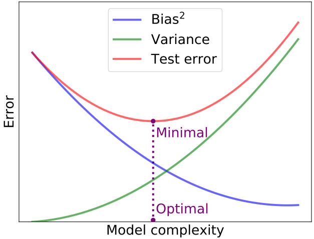


Figure 2: The bias-variance tradeoff.

- Fig. 2 also shows the minimal test error and the corresponding optimal model complexity.
- When model complexity $<$ the optimal complexity, we call this *Underfitting*.
- When model complexity $>$ the optimal complexity, we call this *Overfitting*.
- **Q:** Since the optimal complexity is usually unknown, how can we tell when we are underfitting and when we are overfitting?
- **A:** We can use the *Learning Curve* to do so.

# Learning Curve


Figure 3: Learning Curve showing underfitting (left) and overfitting (right).

- The *Learning Curve* shows the training and validation error as a function of the number of training iterations or samples.

**Takeaway**

- The left panel of fig. 3 shows the signs of underfitting:
  - training error is high
  - validation error is close to training error
- The right panel of fig. 3 shows the signs of overfitting:
  - training error is low
  - validation error is much higher than training error

# Handling Underfitting and Overfitting: The Idea

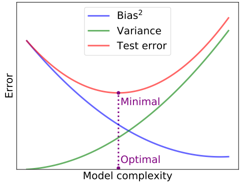


Figure 2: The bias-variance tradeoff.

- Underfitting indicates:
  - model complexity < the optimal complexity
  - we are on the left-hand side of the vertical dashed line in fig. 2
- Overfitting indicates:
  - model complexity > the optimal complexity
  - we are on the right-hand side of the vertical dashed line in fig. 2
- Both underfitting and overfitting result in higher test error (than the minimal).
- To handle underfitting, we should increase model complexity, so that we can significantly lower the squared bias and, in turn, the test error.
- To handle overfitting, we should decrease model complexity, so that we can significantly lower the variance and, in turn, the test error.

# Handling Underfitting and Overfitting: The Methods

**Good practice**

- Methods for handling underfitting:
  - use more complex model + regularization
  - boosting (see /p2_c2_s5_tree_based_models)
- Methods for handling overfitting:
  - regularization
  - bagging (see /p2_c2_s5_tree_based_models)
  - (allocate or collect) more data for training

# Motivation

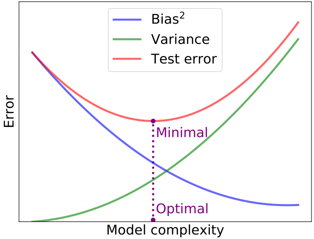


Figure 2: The bias-variance tradeoff.

- The idea of *Regularization* is handling overfitting by lowering model complexity.
- As shown in fig. 2, this allows us to significantly lower the variance and, in turn, lower the test error (i.e., the model will generalize better in reality).

# Popular Regularization Methods

- For both shallow and deep learning:
  - Lasso (a.k.a., L1 regularization)
  - Ridge (a.k.a., L2 regularization)
  - Elastic net
  - Early stopping (see /p2_c2_s5_tree_based_models)
- For deep learning only:
  - Batch normalization (see /p3_c2_s2_training_deep_neural_networks)
  - Drop out (see /p3_c2_s2_training_deep_neural_networks)
  - Data augmentation (see /p3_c2_s2_training_deep_neural_networks)
- For most regularization methods, we will use Mini-Batch Gradient Descent (MBGD) as the default for gradient descent, since as discussed in /p2_c2_s1_linear_regression:
  - in theory, MBGD reduces to Batch Gradient Descent (BGD) / Stochastic Gradient Descent (SGD) when the mini-batch contains all the samples / only one sample, so that we can slightly tweak the equations for MBGD (with respect to the mini-batch size) to get the equations for BGD and SGD
  - in practice, MBGD is more popular in deep learning

# Lasso, Ridge and Elastic Net: Similarity

- The idea of lasso, ridge and elastic net are very similar: all of them aim to push parameter values toward zero, by adding the parameter values to the loss function.
- We will use linear regression in eq. (5) to show why this will decrease model complexity and variance (and finally the test error):

$$\widehat{y^i} = b + w_1 x_1^i +, \ldots, + w_n x_n^i. \tag{5}$$

- Model complexity:
  - we can measure the complexity of linear equation as the number of features (i.e., $x$) in eq. (5)
  - based on eq. (5), the more weights (e.g., $w$) are zero, the fewer features remain in the equation, hence the lower the model complexity
- Variance:
  - the variance was given in eq. (2)

$$\mathrm{Var}(\widehat{\mathbf{y}}) = E\left[(\widehat{\mathbf{y}} - E[\widehat{\mathbf{y}}])^2\right] = \frac{\sum_{i=1}^{m}(\widehat{y^i} - \frac{\sum_{i=1}^{m}\widehat{y^i}}{m})^2}{m} \tag{2}$$

  - by substituting eq. (5) into eq. (2), we have

$$\mathrm{Var}(\widehat{\mathbf{y}}) = \frac{\sum_{i=1}^{m}\left(\sum_{j=1}^{n} w_j (x_j^i - E[\mathbf{x}_j])\right)^2}{m} \tag{6}$$

  - based on eq. (6), generally the lower the (absolute value of the) weights, the lower the variance

# Lasso, Ridge and Elastic Net: Difference

- While lasso, ridge and elastic net all add parameter values to the loss function, they do so in different ways.
- Lasso adds a weighted sum of the absolute value of the weights:

$$\alpha \sum_{j=1}^{n} |w_j|. \tag{7}$$

- Ridge adds a weighted sum of the squared value of the weights:

$$\frac{\alpha}{2} \sum_{j=1}^{n} w_j^2. \tag{8}$$

- Elastic net adds a weighted sum of the absolute value of the weights (first item in eq. (9)), and a weighted sum of the squared value of the weights (second item):

$$\alpha\gamma \sum_{j=1}^{n} |w_j| + \frac{\alpha(1-\gamma)}{2} \sum_{j=1}^{n} w_j^2. \tag{9}$$

- Here $\alpha$ (where $\alpha \geq 0$) and $\gamma$ (where $0 \leq \gamma \leq 1$) are the regularization parameters.
- The larger the $\alpha$, the stronger the regularization, in turn, the smaller the weights.
- The larger the $\gamma$, the similar the elastic net to lasso, whereas the smaller the $\gamma$, the similar the elastic net to ridge.
- Elastic net reduces to lasso / ridge when $\gamma$ is 1 / 0.

# MBGD + Lasso: Loss

- With the MBGD loss (second item in eq. (10)) and the regularization term of lasso (third item), the loss of MBGD + lasso is the sum of the two:

$$\underbrace{\mathcal{L}_{m+l_1}(\boldsymbol{\theta}^j)}_{\text{MBGD + lasso loss}} = \underbrace{\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}(y^i - \widehat{y^i})^2}_{\text{MBGD loss}} + \underbrace{\alpha\sum_{j=1}^{n}|w_j|}_{\text{lasso term}}. \qquad (10)$$

Here:

- $\boldsymbol{\theta}$ (where $\boldsymbol{\theta} = \begin{bmatrix} b & w_1 \cdots w_n \end{bmatrix}^\intercal$) is the parameter vector
- $|\mathbf{mb}^j|$ is the number of samples in mini-batch $\mathbf{mb}^j$
- $y^i$ / $\widehat{y^i}$ is the real / predicted target value of sample $i$, where

$$\widehat{y^i} = b + w_1 x_1^i +, \ldots, + w_n x_n^i = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \begin{bmatrix} b & w_1 \cdots w_n \end{bmatrix}^\intercal = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \boldsymbol{\theta} \qquad (11)$$

- $\alpha$ is the regularization parameter

# MBGD + Lasso: Updating Rule

- The updating rule of MBGD was given in eq. (12)

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j - \eta_k \mathbf{g}_k^j = \boldsymbol{\theta}_k - \eta_k \left. \nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal \right|_{\boldsymbol{\theta}^j = \boldsymbol{\theta}_k^j}, \tag{12}$$

  where the MBGD loss, $\mathcal{L}(\boldsymbol{\theta}^j)$, was given in eq. (13)

$$\mathcal{L}(\boldsymbol{\theta}^j) = \frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} (y^i - \widehat{y^i})^2. \tag{13}$$

- By replacing the MBGD loss in eq. (12), $\mathcal{L}(\boldsymbol{\theta}^j)$ (also the second item in eq. (10)), with MBGD + lasso loss, $\mathcal{L}_{m+l_1}(\boldsymbol{\theta}^j)$ (first item in eq. (10)), we can write the updating rule of MBGD + lasso as

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j - \eta_k \mathbf{g}_k^j = \boldsymbol{\theta}_k^j - \eta_k \left. \nabla \mathcal{L}_{m+l_1}(\boldsymbol{\theta}^j)^\intercal \right|_{\boldsymbol{\theta}^j = \boldsymbol{\theta}_k^j}. \tag{14}$$

# MBGD + Lasso: Updating Rule

- By deriving the gradient in eq. (14), $\nabla \mathcal{L}_{m+l_1}(\boldsymbol{\theta}^j)^\intercal \big|_{\boldsymbol{\theta}^j = \boldsymbol{\theta}_k^j}$, we can write eq. (14) as

$$\begin{aligned}\boldsymbol{\theta}_k^{j+1} &= \boldsymbol{\theta}_k^j + \eta_k \left( \frac{2}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} (y^i - \widehat{y^i}) - \alpha \begin{bmatrix} 0 & \mathrm{sgn}(w_1) \cdots \mathrm{sgn}(w_n) \end{bmatrix}^\intercal \right), \\ &= \boldsymbol{\theta}_k^j + \eta_k \left( \frac{2}{|\mathbf{mb}^j|} \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix}^\intercal (\mathbf{y}^j - \widehat{\mathbf{y}^j}) - \alpha \begin{bmatrix} 0 & \mathrm{sgn}(w_1) \cdots \mathrm{sgn}(w_n) \end{bmatrix}^\intercal \right).\end{aligned} \tag{15}$$

  Here
  - $\eta_k$ is the learning rate in epoch $k$
  - $|\mathbf{mb}^j|$ 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. (11)

    $$\widehat{y^i} = b + w_1 x_1^i +, \ldots, + w_n x_n^i = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \begin{bmatrix} b & w_1 \cdots w_n \end{bmatrix}^\intercal = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \boldsymbol{\theta}_k^j \tag{11}$$

    and $\mathbf{y}^j$ / $\widehat{\mathbf{y}^j}$ is the $|\mathbf{mb}^j| \times 1$ real / predicted target vector, where

    $$\widehat{\mathbf{y}^j} = b + w_1 \mathbf{x}_1 +, \ldots, + w_n \mathbf{x}_n = \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix} \begin{bmatrix} b & w_1 \cdots w_n \end{bmatrix}^\intercal = \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix} \boldsymbol{\theta}_k^j \tag{16}$$
  - $\mathbf{x}^i$ is the $1 \times n$ feature vector of sample $i$, and $\mathbf{X}^j$ the $|\mathbf{mb}^j| \times n$ feature matrix of $\mathbf{mb}^j$
  - $\mathrm{sgn}$ is the *Sign* function:

    $$\mathrm{sgn}(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{cases} \tag{17}$$

- See the proof of eq. (15) in Appendix (pages 54 to 58).

# MBGD + Lasso: The Implementation

- See /models/p2_shallow_learning:
  1. cell 4

# MBGD + Ridge: Loss

- With the MBGD loss (second item in eq. (18)) and the regularization term of ridge (third item), the loss of MBGD + ridge is the sum of the two:

$$\underbrace{\mathcal{L}_{m+l_2}(\boldsymbol{\theta}^j)}_{\text{MBGD + ridge loss}} = \underbrace{\frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}(y^i - \widehat{y^i})^2}_{\text{MBGD loss}} + \underbrace{\frac{\alpha}{2}\sum_{j=1}^{n} w_j^2}_{\text{ridge term}}. \quad (18)$$

Here:

- $\boldsymbol{\theta}$ (where $\boldsymbol{\theta} = \begin{bmatrix} b & w_1 \cdots w_n \end{bmatrix}^\intercal$) is the parameter vector
- $|\mathbf{mb}^j|$ 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. (11)

$$\widehat{y^i} = b + w_1 x_1^i +, \ldots, + w_n x_n^i = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \begin{bmatrix} b & w_1 \cdots w_n \end{bmatrix}^\intercal = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \boldsymbol{\theta} \quad (11)$$

- $\alpha$ is the regularization parameter

# MBGD + Ridge: Updating Rule

- The updating rule of MBGD was given in eq. (12)

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j - \eta_k \mathbf{g}_k^j = \boldsymbol{\theta}_k^j - \eta_k \left. \nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal \right|_{\boldsymbol{\theta}^j = \boldsymbol{\theta}_k^j}, \tag{12}$$

  where the MBGD loss, $\mathcal{L}(\boldsymbol{\theta}^j)$, was given in eq. (13)

$$\mathcal{L}(\boldsymbol{\theta}^j) = \frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} (y^i - \widehat{y^i})^2. \tag{13}$$

- By replacing the MBGD loss in eq. (12), $\mathcal{L}(\boldsymbol{\theta}^j)$ (also the second item in eq. (18)), with MBGD + ridge loss, $\mathcal{L}_{m+l_2}(\boldsymbol{\theta}^j)$ (first item in eq. (18)), we can write the updating rule of MBGD + lasso as

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j - \eta_k \mathbf{g}_k^j = \boldsymbol{\theta}_k^j - \eta_k \left. \nabla \mathcal{L}_{m+l_2}(\boldsymbol{\theta}^j)^\intercal \right|_{\boldsymbol{\theta}^j = \boldsymbol{\theta}_k^j}. \tag{19}$$

# MBGD + Ridge: Updating Rule

- By deriving the gradient in eq. (19), $\nabla \mathcal{L}_{m+l_2}(\boldsymbol{\theta}^j)^\intercal \big|_{\boldsymbol{\theta}^j = \boldsymbol{\theta}^j_k}$, we can write eq. (19) as

$$\begin{aligned}\boldsymbol{\theta}^{j+1}_k &= \boldsymbol{\theta}^j_k + \eta_k \left( \frac{2}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} (y^i - \widehat{y^i}) - \alpha \begin{bmatrix} 0 & w_1 \cdots w_n \end{bmatrix}^\intercal \right), \\ &= \boldsymbol{\theta}^j_k + \eta_k \left( \frac{2}{|\mathbf{mb}^j|} \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix}^\intercal (\mathbf{y}^j - \widehat{\mathbf{y}^j}) - \alpha \begin{bmatrix} 0 & w_1 \cdots w_n \end{bmatrix}^\intercal \right).\end{aligned} \tag{20}$$

  Here:
  - $\eta_k$ is the learning rate in epoch $k$
  - $|\mathbf{mb}^j|$ 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. (11)

    $$\widehat{y^i} = b + w_1 x^i_1 +, \ldots, + w_n x^i_n = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \begin{bmatrix} b & w_1 \cdots w_n \end{bmatrix}^\intercal = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \boldsymbol{\theta}^j_k \tag{11}$$

    and $\mathbf{y}^j$ / $\widehat{\mathbf{y}^j}$ is the $|\mathbf{mb}^j| \times 1$ real / predicted target vector, given in eq. (16)

    $$\widehat{\mathbf{y}^j} = b + w_1 \mathbf{x}_1 +, \ldots, + w_n \mathbf{x}_n = \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix} \begin{bmatrix} b & w_1 \cdots w_n \end{bmatrix}^\intercal = \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix} \boldsymbol{\theta}^j_k \tag{16}$$
  - $\mathbf{x}^i$ is the $1 \times n$ feature vector of sample $i$, and $\mathbf{X}^j$ the $|\mathbf{mb}^j| \times n$ feature matrix of $\mathbf{mb}^j$
- See the proof of eq. (20) in Appendix (pages 59 to 61).

# MBGD + Ridge: The Implementation

- See /models/p2_shallow_learning:
  1. cell 4

# MBGD + Elastic Net: Loss

- With the MBGD loss (second item in eq. (21)) and the regularization term of elastic net (third item), the loss of MBGD + elastic net is the sum of the two:

$$\underbrace{\mathcal{L}_{m+l_{12}}(\boldsymbol{\theta}^j)}_{\text{MBGD + elastic net loss}} = \underbrace{\frac{1}{|\mathbf{mb}^j|}\sum_{i \in \mathbf{mb}^j}(y^i - \widehat{y^i})^2}_{\text{MBGD loss}} + \underbrace{\alpha\gamma\sum_{j=1}^{n}|w_j| + \frac{\alpha(1-\gamma)}{2}\sum_{j=1}^{n}w_j^2}_{\text{elastic net term}}. \quad (21)$$

Here:

- $\boldsymbol{\theta}$ (where $\boldsymbol{\theta} = \begin{bmatrix} b & w_1 \cdots w_n \end{bmatrix}^\intercal$) is the parameter vector
- $|\mathbf{mb}^j|$ is the number of samples in mini-batch $\mathbf{mb}^j$
- $y^i$ / $\widehat{y^i}$ is the real / predicted target value of sample $i$, where

$$\widehat{y^i} = b + w_1 x_1^i +, \ldots, + w_n x_n^i = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \begin{bmatrix} b & w_1 \cdots w_n \end{bmatrix}^\intercal = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \boldsymbol{\theta} \quad (11)$$

- $\alpha$ and $\gamma$ are the regularization parameters

# MBGD + Elastic Net: Updating Rule

- The updating rule of MBGD was given in eq. (12)

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j - \eta_k \mathbf{g}_k^j = \boldsymbol{\theta}_k^j - \eta_k \left. \nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal \right|_{\boldsymbol{\theta}^j = \boldsymbol{\theta}_k^j}, \tag{12}$$

where the MBGD loss, $\mathcal{L}(\boldsymbol{\theta}^j)$, was given in eq. (13):

$$\mathcal{L}(\boldsymbol{\theta}^j) = \frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} (y^i - \widehat{y^i})^2. \tag{13}$$

- By replacing the MBGD loss in eq. (12), $\mathcal{L}(\boldsymbol{\theta}^j)$ (also the second item in eq. (21)), with MBGD + elastic net loss, $\mathcal{L}_{m+l_{12}}(\boldsymbol{\theta}^j)$ (first item in eq. (21)), we can write the updating rule of MBGD + elastic net as

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j - \eta_k \mathbf{g}_k^j = \boldsymbol{\theta}_k^j - \eta_k \left. \nabla \mathcal{L}_{m+l_{12}}(\boldsymbol{\theta}^j)^\intercal \right|_{\boldsymbol{\theta}^j = \boldsymbol{\theta}_k^j}. \tag{22}$$

# MBGD + Elastic Net: Updating Rule

- By deriving the gradient in eq. (22), $\nabla \mathcal{L}_{m+l_{12}}(\boldsymbol{\theta}^j)^\intercal \big|_{\boldsymbol{\theta}^j = \boldsymbol{\theta}^j_k}$, we can write eq. (22) as

$$\begin{aligned}\boldsymbol{\theta}^{j+1}_k &= \boldsymbol{\theta}^j_k + \eta_k \left( \frac{2\eta_k}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} (y^i - \widehat{y^i}) - \alpha\gamma \begin{bmatrix} 0 & \operatorname{sgn}(w_1) \cdots \operatorname{sgn}(w_n) \end{bmatrix}^\intercal - \alpha(1-\gamma) \begin{bmatrix} 0 & w_1 \cdots w_n \end{bmatrix}^\intercal \right) \\ &= \boldsymbol{\theta}^j_k + \eta_k \left( \frac{2\eta_k}{|\mathbf{mb}^j|} \begin{bmatrix} 1 & \mathbf{X}^j \end{bmatrix}^\intercal (\mathbf{y}^j - \widehat{\mathbf{y}^j}) - \alpha\gamma \begin{bmatrix} 0 & \operatorname{sgn}(w_1) \cdots \operatorname{sgn}(w_n) \end{bmatrix}^\intercal - \alpha(1-\gamma) \begin{bmatrix} 0 & w_1 \cdots w_n \end{bmatrix}^\intercal \right).\end{aligned} \tag{23}$$

  Here
  - $\eta_k$ is the learning rate in epoch $k$
  - $|\mathbf{mb}^j|$ 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. (11)

$$\widehat{y^i} = b + w_1 x^i_1 +, \ldots, + w_n x^i_n = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \begin{bmatrix} b & w_1 \cdots w_n \end{bmatrix}^\intercal = \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} \boldsymbol{\theta}^j_k \tag{11}$$

  and $\mathbf{y}^j$ / $\widehat{\mathbf{y}^j}$ is the $|\mathbf{mb}^j| \times 1$ real / predicted target vector, given in eq. (16)

$$\widehat{\mathbf{y}^j} = b + w_1 \mathbf{x}_1 +, \ldots, + w_n \mathbf{x}_n = \begin{bmatrix} 1 & \mathbf{X}^j \end{bmatrix} \begin{bmatrix} b & w_1 \cdots w_n \end{bmatrix}^\intercal = \begin{bmatrix} 1 & \mathbf{X}^j \end{bmatrix} \boldsymbol{\theta}^j_k \tag{16}$$

  - $\mathbf{x}^i$ is the $1 \times n$ feature vector of sample $i$, and $\mathbf{X}^j$ the $|\mathbf{mb}^j| \times n$ feature matrix of $\mathbf{mb}^j$
  - sgn is the sign function:

$$\operatorname{sgn}(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{cases} \tag{17}$$

- See the proof of eq. (15) in Appendix (pages 62 to 64).

# MBGD + Elastic Net: The Implementation

- See /models/p2_shallow_learning:
  1. cell 4

# Lasso VS Ridge VS Elastic Net

**Good practice**

- Ridge is a good default.
- However, if across all the features only a few of them are relevant:
  - use elastic net or lasso, because they tend to push parameter values of irrelevant features to exact zero
  - elastic net is preferred, because lasso may perform badly when
    - the number of features is higher than the number of samples (i.e., $n > m$)
    - some features are strongly correlated

# Parameters

- Parameters of a model or training method are the unknowns that are:
  - not fixed
  - but updated during training
- For example, $\boldsymbol{\theta} = \begin{bmatrix} b & w_1 \cdots w_n \end{bmatrix}^\intercal$ (bias and weights) are the parameters of linear regression in eq. (24)

$$\widehat{\mathbf{y}} = b + w_1\mathbf{x}_1 +, \ldots, +w_n\mathbf{x}_n = \begin{bmatrix} \mathbf{1} & \mathbf{X} \end{bmatrix} \begin{bmatrix} b & w_1 \cdots w_n \end{bmatrix}^\intercal = \begin{bmatrix} \mathbf{1} & \mathbf{X} \end{bmatrix} \boldsymbol{\theta}. \tag{24}$$

- These parameters are:
  - not fixed
  - but updated using, say, the updating rule of MBGD + ridge in eq. (20)

$$\begin{aligned}
\boldsymbol{\theta}_k^{j+1} &= \boldsymbol{\theta}_k^j + \eta_k \left( \frac{2}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} (y^i - \widehat{y^i}) - \alpha \begin{bmatrix} 0 & w_1 \cdots w_n \end{bmatrix}^\intercal \right), \\
&= \boldsymbol{\theta}_k^j + \eta_k \left( \frac{2}{|\mathbf{mb}^j|} \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix}^\intercal (\mathbf{y}^j - \widehat{\mathbf{y}^j}) - \alpha \begin{bmatrix} 0 & w_1 \cdots w_n \end{bmatrix}^\intercal \right).
\end{aligned} \tag{20}$$

# Hyperparameters

- Hyperparameters of a model or training method are the unknowns that are:
  - fixed
  - and not updated during training
- For example, $\eta_k$ (learning rate) and $\alpha$ (regularization parameter) are the hyperparameters of the updating rule of MBGD + ridge in eq. (20)

$$
\begin{aligned}
\boldsymbol{\theta}_k^{j+1} &= \boldsymbol{\theta}_k^{j} + \eta_k \left( \frac{2}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix} (y^i - \widehat{y^i}) - \alpha \begin{bmatrix} 0 & w_1 \cdots w_n \end{bmatrix}^\intercal \right), \\
&= \boldsymbol{\theta}_k^{j} + \eta_k \left( \frac{2}{|\mathbf{mb}^j|} \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix}^\intercal (\mathbf{y}^j - \widehat{\mathbf{y}^j}) - \alpha \begin{bmatrix} 0 & w_1 \cdots w_n \end{bmatrix}^\intercal \right).
\end{aligned} \tag{20}
$$

- These hyperparameters are:
  - fixed
  - and not updated during training
- It is worth noting that learning rate is not necessarily a hyperparameter:
  - we can use methods such as *Learning Rate Scheduling* to update it during training (see /p3_c2_s2_training_deep_neural_networks)
  - in this case, learning rate is a parameter rather than a hyperparameter

# Hyperparameter Tuning: Motivation

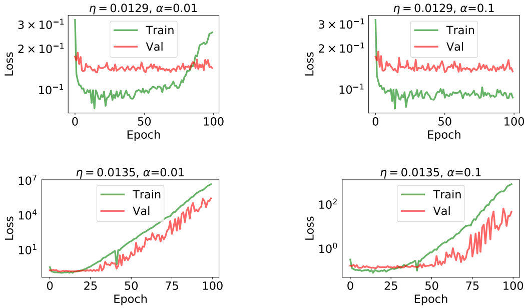


Figure 4: Training and validation loss of MBGD + ridge with different combinations of learning rate ($\eta$) and regularization parameter ($\alpha$).

- By comparing the two rows in each column of fig. 4, we can see that the training and validation loss can be quite sensitive to $\eta$.
- By comparing the two columns in each row of fig. 4, we can see that the training and validation loss can be quite sensitive to $\alpha$.
- The goal of hyperparameter tuning is finding hyperparameter values that lead to good validation performance (e.g., low validation loss).

# Hyperparameter Tuning: Idea

Table 1: Combinations of learning rate ($\eta$) and regularization parameter ($\alpha$) and their validation MSE. The best combination is highlighted in red.

| $\eta$ | $\alpha$ | Val MSE |
|---|---|---|
| 0.001 | 0.01 | $0.138$ |
| 0.001 | 0.1 | $0.139$ |
| 0.02 | 0.1 | $2.35 \times 10^{52}$ |
| 0.02 | 0.01 | $5.94 \times 10^{57}$ |

- Let us use table 1 as an example to illustrate the idea of hyperparameter tuning:
  1. we loop over each combination of $\eta$ and $\alpha$, and for each combination:
     1. we train the model (on the training data) using the combination as the hyperparameter values
     2. we get the validation MSE of the model (on the validation data)
  2. we pick the first combination (highlighted in red) as the best hyperparameter values since it leads to the lowest validation MSE
  3. we retrain the model (on the combined training and validation data) with the best hyperparameter values picked earlier

# Hyperparameter Tuning in Sklearn: Two Popular Methods

- There are two popular hyperparameter tuning methods in sklearn:
  - `GridSearchCV`
  - `RandomizedSearchCV`
- The key difference between the two methods lies in:
  - how they expect the user to propose values of a single hyperparameter
  - how they produce combinations of values of all the hyperparameters
- After producing the combinations of values, both methods:
  1. loop over each combination, and for each combination:
     1. train the model (on the training data) using the combination as the hyperparameter values
     2. get the validation performance of the model (on the validation data)
  2. pick the best hyperparameter values that lead to the best validation performance
  3. (when setting parameter `refit` as `True`) retrain the model (on the combined training and validation data) with the best hyperparameter values picked earlier

# Hyperparameter Tuning in Sklearn: Good Practice

**Good practice**

- It is recommended to set parameter `refit` as `True` when using `GridSearchCV` and `RandomizedSearchCV`.
- This allows us to retrain the model (i.e., its parameters) on the combined training and validation data with the best hyperparameter values.
- While retraining model requires extra computational cost, doing so will usually improve model performance (which is often preferred).

# GridSearchCV: Parameter Grid

Table 1: Combinations of learning rate ($\eta$) and regularization parameter ($\alpha$) and their validation MSE. The best combination is highlighted in red.

| $\eta$ | $\alpha$ | Val MSE |
|---|---|---|
| 0.001 | 0.01 | 0.138 |
| 0.001 | 0.1 | 0.139 |
| 0.02 | 0.1 | $2.35 \times 10^{52}$ |
| 0.02 | 0.01 | $5.94 \times 10^{57}$ |

- GridSearchCV expects a list of possible values for each hyperparameter.
- This list of values is also called *Parameter Grid* (hence the name of GridSearchCV).
- In table 1 we used the grid below for $\eta$ and $\alpha$ (in MBGD + ridge):
  - $\eta$: $[0.001, 0.02]$
  - $\alpha$: $[0.01, 0.1]$
- Based on the parameter grid of each hyperparameter, GridSearchCV produces all the possible combinations of hyperparameter values.
- With the grid of $\eta$ and $\alpha$ above, we will have four combinations, shown in table 1.

# GridSearchCV: Code Example

- See [/p2_c2_s2_training_shallow_models/code_example]:
  1. cells 54 to 56
  2. cells 57 to 61

# GridSearchCV: Pros and Cons

- Pros:
  - we have full control:
    - we can use parameter grid to specify the exact hyperparameter values we want to fine-tune
- Cons:
  - it is not scalable:
    - assume there are $n$ hyperparameters and for each hyperparameter we only fine-tune two values
    - the number of combination of hyperparameter values is $2^n$

# RandomizedSearchCV: Parameter Distribution

Table 2: Combinations of learning rate ($\eta$) and regularization parameter ($\alpha$) and their validation MSE. The best combination is highlighted in red.

| $\eta$ | $\alpha$ | Val MSE |
|---|---|---|
| 0.0124 | 0.0759 | $0.1350$ |
| 0.0040 | 0.024 | $0.1355$ |
| 0.0175 | 0.0152 | $5.31 \times 10^{40}$ |
| 0.0191 | 0.0437 | $1.11 \times 10^{50}$ |

- Unlike GridSearchCV that expects a list of possible values for each hyperparameter, RandomizedSearchCV expects a distribution for each hyperparameter.
- Possible values of a hyperparameter will then be randomly sampled from the distribution (hence the name of RandomizedSearchCV).
- In table 2 we used the distribution below for $\eta$ and $\alpha$ (in MBGD + ridge):
  - $\eta$: uniform(loc=0.01, scale=0.003)
  - $\alpha$: uniform(loc=0.01, scale=0.09)
- Based on the distribution of each hyperparameter, and parameter `n_iter`, RandomizedSearchCV produces `n_iter` combinations of hyperparameter values.
- With the distribution of $\eta$ and $\alpha$ above, and `n_iter` $= 4$, we could have four combinations, shown in table 2.

## RandomizedSearchCV: Code Example

- See /p2_c2_s2_training_shallow_models/code_example:
  1. cells 54 to 56
  2. cells 62 to 66

# RandomizedSearchCV: Pros and Cons

- Pros:
  - it is scalable:
    - the number of combination of hyperparameter values is not determined by the number of hyperparameters
    - instead, it is determined by parameter `n_iter` of RandomizedSearchCV
- Cons:
  - we do not have full control:
    - hyperparameter values we want to fine-tune are randomly sampled from the parameter distributions

# GridSearchCV VS RandomizedSearchCV: Good Practice

**Good practice**

- When there are many hyperparameters to fine-tune:
  - it is recommended to use `RandomizedSearchCV` (so that hyperparameter tuning can be scalable)
- When there are only a few hyperparameters to fine-tune:
  - it is recommended to use `GridSearchCV` (so that we can have full control of the hyperparameter values to fine-tune)

# Model Selection: Motivation

- For a problem, (in theory) there are usually many models we can use.
- Take linear regression for example, we have sklearn models such as:
  - `LinearRegression`
  - `SGDRegressor`
  - `Lasso`
  - `Ridge`
  - `ElasticNet`
- While for certain problems some models are favored over others, we may not know for sure which model actually works the best.
- As a result, we may have to:
  1. try many models
  2. select the top-1 model or ensemble of top-$k$ models for production
- The process of trying many models and selecting some of them is called *Model Selection*.

# Model Selection: Idea

- The idea of model selection is as follows:
  1. for each model:
     1. we fine-tune its hyperparameters and select the best combination of hyperparameter values (ones with the best validation performance)
     2. we retrain the model using the best combination selected earlier on the combined training and validation data
  2. we select the top-1 retrained model or ensemble of top-$k$ retrained models (based on the validation performance of the models)
  3. we test the selected retrained models on the test data to estimate how well they generalize in reality

# Model Selection: Code Example

- See /p2_c2_s2_training_shallow_models/code_example:
  1. cell 67
  2. cell 69

# Proof of Decomposition of Expected Test Error: Page 12

- The expected test error, $E\left[(\widehat{\mathbf{y}}-\mathbf{y})^2\right]$, can be written as

$$\underbrace{E\left[(\widehat{\mathbf{y}}-\mathbf{y})^2\right]}_{\text{Expected test error}} = E\left[\widehat{\mathbf{y}}^2 - 2\widehat{\mathbf{y}}\mathbf{y} + \mathbf{y}^2\right] = E\left[\widehat{\mathbf{y}}^2\right] - 2E\left[\widehat{\mathbf{y}}\mathbf{y}\right] + E\left[\mathbf{y}^2\right] \tag{25}$$

- Since $\widehat{\mathbf{y}}$ and $\mathbf{y}$ are independent, we can write eq. (25) as

$$\underbrace{E\left[(\widehat{\mathbf{y}}-\mathbf{y})^2\right]}_{\text{Expected test error}} = E\left[\widehat{\mathbf{y}}^2\right] - 2E[\widehat{\mathbf{y}}]E[\mathbf{y}] + E\left[\mathbf{y}^2\right]. \tag{26}$$

- Let $\mathbf{a}$ be a vector and $E[\mathbf{a}]$ the expectation of $\mathbf{a}$, then

$$\begin{aligned} E\left[(\mathbf{a}-E[\mathbf{a}])^2\right] &= E\left[\mathbf{a}^2 - 2\mathbf{a}E[\mathbf{a}] + E[\mathbf{a}]^2\right], \\ &= E\left[\mathbf{a}^2\right] - 2E\big[\mathbf{a}E[\mathbf{a}]\big] + E\left[E[\mathbf{a}]^2\right], \\ &= E\left[\mathbf{a}^2\right] - 2E[\mathbf{a}]^2 + E[\mathbf{a}]^2, \\ &= E\left[\mathbf{a}^2\right] - E[\mathbf{a}]^2. \end{aligned} \tag{27}$$

- Based on eq. (27), we have

$$E\left[\mathbf{a}^2\right] = E\left[(\mathbf{a}-E[\mathbf{a}])^2\right] + E[\mathbf{a}]^2. \tag{28}$$

# Proof of Decomposition of Expected Test Error: Page 12

- Based on eq. (28), we can write $E\left[\widehat{\mathbf{y}}^2\right]$ in eq. (25) as

$$E\left[\widehat{\mathbf{y}}^2\right] = \underbrace{E\left[\left(\widehat{\mathbf{y}} - E[\widehat{\mathbf{y}}]\right)^2\right]}_{\text{Variance}} + E[\widehat{\mathbf{y}}]^2. \tag{29}$$

- Similarly, based on eq. (28), we can write $E\left[\mathbf{y}^2\right]$ in eq. (25) as:

$$E\left[\mathbf{y}^2\right] = E\left[\left(\mathbf{y} - E[\mathbf{y}]\right)^2\right] + E[\mathbf{y}]^2. \tag{30}$$

- Based on eq. (4)

$$\mathbf{y} = \begin{bmatrix} y^1 & \ldots & y^m \end{bmatrix}^\intercal, \tag{4}$$

where $y^i = y$ and $y$ is the target value in the test sample, we have

$$E\left[\left(\mathbf{y} - E[\mathbf{y}]\right)^2\right] = 0. \tag{31}$$

- By substituting eq. (31) into eq. (30), we have

$$E\left[\mathbf{y}^2\right] = E[\mathbf{y}]^2. \tag{32}$$

# Proof of Decomposition of Expected Test Error: Page 12

- By substituting eqs. (29) and (32) into eq. (26), we have

$$\begin{aligned}
\underbrace{E\left[(\widehat{\mathbf{y}} - \mathbf{y})^2\right]}_{\text{Expected test error}} &= E\left[\widehat{\mathbf{y}}^2\right] - 2E[\widehat{\mathbf{y}}]E[\mathbf{y}] + E\left[\mathbf{y}^2\right], \\
&= \underbrace{E\left[(\widehat{\mathbf{y}} - E[\widehat{\mathbf{y}}])^2\right]}_{\text{Variance}} + \left(E[\widehat{\mathbf{y}}]^2 - 2E[\widehat{\mathbf{y}}]E[\mathbf{y}] + E[\mathbf{y}]^2\right), \\
&= \underbrace{E\left[\widehat{\mathbf{y}} - E[\widehat{\mathbf{y}}])^2\right]}_{\text{Variance}} + (E[\widehat{\mathbf{y}}] - E[\mathbf{y}])^2, \\
&= \underbrace{E\left[\widehat{\mathbf{y}} - E[\widehat{\mathbf{y}}])^2\right]}_{\text{Variance}} + \underbrace{(E[\widehat{\mathbf{y}} - \mathbf{y}])^2}_{\text{Bias}^2}.
\end{aligned} \tag{33}$$

which proves the claim in eq. (3) on page 12. $\square$

# Proof of Updating Rule: Page 24

- The MBGD + lasso loss can be written as

$$\underbrace{\mathcal{L}_{m+l_1}(\boldsymbol{\theta}^j)}_{\text{MBGD + lasso loss}} = \underbrace{\mathcal{L}(\boldsymbol{\theta}^j)}_{\text{MBGD loss}} + \underbrace{\alpha \sum_{j=1}^{n} |w_j|}_{\text{lasso term}}, \tag{34}$$

where the MBGD loss, $\mathcal{L}(\boldsymbol{\theta}^j)$, was given in eq. (13)

$$\mathcal{L}(\boldsymbol{\theta}^j) = \frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} (y^i - \widehat{y^i})^2. \tag{13}$$

- The gradient of MBGD + lasso loss, $\nabla \mathcal{L}_{m+l_1}(\boldsymbol{\theta}^j)^\intercal$, can be written as the sum of the gradient of MBGD loss (second item in eq. (34)) and the gradient of lasso term (third item):

$$\nabla \mathcal{L}_{m+l_1}(\boldsymbol{\theta}^j)^\intercal = \nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal + \left(\nabla \alpha \sum_{j=1}^{n} |w_j|\right)^\intercal. \tag{35}$$

# Proof of Updating Rule: Page 24

- The gradient of MBGD loss (second item in eq. (35)), $\nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal$, can be written as

$$\nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal = \left[ \frac{\partial}{\partial b}\mathcal{L}(\boldsymbol{\theta}^j) \quad \frac{\partial}{\partial w_1}\mathcal{L}(\boldsymbol{\theta}^j) \cdots \frac{\partial}{\partial w_n}\mathcal{L}(\boldsymbol{\theta}^j) \right]^\intercal . \tag{36}$$

- Based on eq. (13), we can write $\frac{\partial}{\partial b}\mathcal{L}(\boldsymbol{\theta}^j)$ as

$$\begin{aligned} \frac{\partial}{\partial b}\mathcal{L}(\boldsymbol{\theta}^j) &= \frac{\partial}{\partial b}\left( \frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left( y^i - \widehat{y^i} \right)^2 \right), \\ &= \frac{2}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left( y^i - \widehat{y^i} \right) \cdot \frac{\partial}{\partial b} \left( y^i - \widehat{y^i} \right), \\ &= \frac{2}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left( y^i - \widehat{y^i} \right) \cdot \frac{\partial}{\partial b} \left( y^i - (b + w_1 x_1^i +, \ldots, + w_n x_n^i) \right), \\ &= \frac{2}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left( y^i - \widehat{y^i} \right) \cdot (-1). \end{aligned} \tag{37}$$

# Proof of Updating Rule: Page 24

- Based on eq. (13), we can write $\frac{\partial}{\partial w_j} \mathcal{L}(\boldsymbol{\theta}^i)$ as

$$
\begin{aligned}
\frac{\partial}{\partial w_j} \mathcal{L}(\boldsymbol{\theta}^j) &= \frac{\partial}{\partial w_j} \left( \frac{1}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left( y^i - \widehat{y^i} \right)^2 \right), \\
&= \frac{2}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left( y^i - \widehat{y^i} \right) \cdot \frac{\partial}{\partial w_j} \left( y^i - \widehat{y^i} \right), \\
&= \frac{2}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left( y^i - \widehat{y^i} \right) \cdot \frac{\partial}{\partial w_j} \left( y^i - (b + w_1 x_1^i +, \ldots, + w_n x_n^i) \right), \\
&= \frac{2}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} \left( y^i - \widehat{y^i} \right) \cdot (-x_j^i).
\end{aligned} \tag{38}
$$

- By substituting eqs. (37) and (38) into eq. (36), we have

$$
\nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal = -\frac{2}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} (y^i - \widehat{y^i}) \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix}^\intercal = -\frac{2}{|\mathbf{mb}^j|} \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix}^\intercal (\mathbf{y}^j - \widehat{\mathbf{y}^j}). \tag{39}
$$

# Proof of Updating Rule: Page 24

- The gradient of the lasso term (third item in eq. (35)), $\nabla\alpha\sum_{j=1}^{n}|w_j|^\intercal$, can be written as

$$\left(\nabla\alpha\sum_{j=1}^{n}|w_j|\right)^\intercal=\alpha\left[\frac{\partial}{\partial b}\sum_{j=1}^{n}|w_j| \quad \frac{\partial}{\partial w_1}\sum_{j=1}^{n}|w_j|\cdots\frac{\partial}{\partial w_n}\sum_{j=1}^{n}|w_j|\right]^\intercal, \tag{40}$$

where

$$|w_j|=\begin{cases}-w_j, & w_j<0\\ 0, & w_j=0\\ w_j, & w_j>0.\end{cases} \tag{41}$$

- Based on eq. (41), we have

$$\frac{\partial}{\partial b}\sum_{j=1}^{n}|w_j|=0 \quad \text{and} \quad \frac{\partial}{\partial w_k}\sum_{j=1}^{n}|w_j|=\mathrm{sgn}(w_k)=\begin{cases}-1, & w_k<0\\ 0, & w_k=0\\ 1, & w_k>0\end{cases} \tag{42}$$

where $1\le k\le n$.

- By substituting eq. (42) into eq. (40), we have

$$\left(\nabla\alpha\sum_{j=1}^{n}|w_j|\right)^\intercal=\alpha\left[0 \quad \mathrm{sgn}(w_1)\cdots\mathrm{sgn}(w_n)\right]^\intercal. \tag{43}$$

# Proof of Updating Rule: Page 24

- By substituting eqs. (39) and (43) into eq. (35),

$$\nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal = -\frac{2}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} (y^i - \widehat{y^i}) \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix}^\intercal = -\frac{2}{|\mathbf{mb}^j|} \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix}^\intercal (\mathbf{y}^j - \widehat{\mathbf{y}^j}), \tag{39}$$

$$\left(\nabla \alpha \sum_{j=1}^{n} |w_j|\right)^\intercal = \alpha \begin{bmatrix} 0 & \mathrm{sgn}(w_1) \cdots \mathrm{sgn}(w_n) \end{bmatrix}^\intercal, \tag{43}$$

$$\nabla \mathcal{L}_{m+l_1}(\boldsymbol{\theta}^j)^\intercal = \nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal + \left(\nabla \alpha \sum_{j=1}^{n} |w_j|\right)^\intercal, \tag{35}$$

  we have

$$\nabla \mathcal{L}_{m+l_1}(\boldsymbol{\theta}^j)^\intercal = -\frac{2}{|\mathbf{mb}^j|} \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix}^\intercal (\mathbf{y}^j - \widehat{\mathbf{y}^j}) + \alpha \begin{bmatrix} 0 & \mathrm{sgn}(w_1) \cdots \mathrm{sgn}(w_n) \end{bmatrix}^\intercal. \tag{44}$$

- By substituting eq. (44) into eq. (14)

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j - \eta_k \mathbf{g}_k^j = \boldsymbol{\theta}_k^j - \eta_k \nabla \mathcal{L}_{m+l_1}(\boldsymbol{\theta}^j)^\intercal \Big|_{\boldsymbol{\theta}^j = \boldsymbol{\theta}_k^j}, \tag{14}$$

  we have

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j + \eta_k \left(\frac{2}{|\mathbf{mb}^j|} \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix}^\intercal (\mathbf{y}^j - \widehat{\mathbf{y}^j}) - \alpha \begin{bmatrix} 0 & \mathrm{sgn}(w_1) \cdots \mathrm{sgn}(w_n) \end{bmatrix}^\intercal\right), \tag{45}$$

  which proves the claim in eq. (15) on page 24. □

# Proof of Updating Rule: Page 28

- The MBGD + ridge loss can be written as

$$\underbrace{\mathcal{L}_{m+l_2}(\boldsymbol{\theta}^j)}_{\text{MBGD + ridge loss}} = \underbrace{\mathcal{L}(\boldsymbol{\theta}^j)}_{\text{MBGD loss}} + \underbrace{\frac{\alpha}{2}\sum_{j=1}^{n} w_j^2}_{\text{ridge term}}, \tag{46}$$

where the MBGD loss, $\mathcal{L}(\boldsymbol{\theta}^j)$, was given in eq. (13)

$$\mathcal{L}(\boldsymbol{\theta}^j) = \frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}(y^i - \widehat{y^i})^2. \tag{13}$$

- The gradient of MBGD + ridge loss, $\nabla\mathcal{L}_{m+l_2}(\boldsymbol{\theta}^j)^\intercal$, can be written as the sum of the gradient of MBGD loss (second item in eq. (46)) and the gradient of ridge term (third item):

$$\nabla\mathcal{L}_{m+l_2}(\boldsymbol{\theta}^j)^\intercal = \nabla\mathcal{L}(\boldsymbol{\theta}^j)^\intercal + \left(\nabla\frac{\alpha}{2}\sum_{j=1}^{n} w_j^2\right)^\intercal, \tag{47}$$

where the gradient of MBGD loss was given in eq. (39)

$$\nabla\mathcal{L}(\boldsymbol{\theta}^j)^\intercal = -\frac{2}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}(y^i - \widehat{y^i})\begin{bmatrix}1 & \mathbf{x}^i\end{bmatrix}^\intercal = -\frac{2}{|\mathbf{mb}^j|}\begin{bmatrix}\mathbf{1} & \mathbf{X}^j\end{bmatrix}^\intercal(\mathbf{y}^j - \widehat{\mathbf{y}^j}). \tag{39}$$

# Proof of Updating Rule: Page 28

- The gradient of the ridge term (third item in eq. (47)), $\nabla \alpha \sum_{j=1}^{n} |w_j|^\intercal$, can be written as

$$\left(\nabla \frac{\alpha}{2} \sum_{j=1}^{n} w_j^2\right)^\intercal = \frac{\alpha}{2} \left[\frac{\partial}{\partial b} \sum_{j=1}^{n} w_j^2 \quad \frac{\partial}{\partial w_1} \sum_{j=1}^{n} w_j^2 \cdots \frac{\partial}{\partial w_n} \sum_{j=1}^{n} w_j^2\right]^\intercal, \tag{48}$$

  where

$$\frac{\partial}{\partial b} \sum_{j=1}^{n} w_j^2 = 0 \quad \text{and} \quad \frac{\partial}{\partial w_k} \sum_{j=1}^{n} w_j^2 = 2w_k \tag{49}$$

  where $1 \leq k \leq n$.

- By substituting eq. (49) into eq. (48), we have

$$\left(\nabla \frac{\alpha}{2} \sum_{j=1}^{n} w_j^2\right)^\intercal = \alpha \left[0 \quad w_1 \cdots w_n\right]^\intercal. \tag{50}$$

# Proof of Updating Rule: Page 28

- By substituting eqs. (39) and (50) into eq. (47),

$$\nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal = -\frac{2}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} (y^i - \widehat{y^i}) \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix}^\intercal = -\frac{2}{|\mathbf{mb}^j|} \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix}^\intercal (\mathbf{y}^j - \widehat{\mathbf{y}^j}), \quad (39)$$

$$\left( \nabla \frac{\alpha}{2} \sum_{j=1}^{n} w_j^2 \right)^\intercal = \alpha \begin{bmatrix} 0 & w_1 \cdots w_n \end{bmatrix}^\intercal, \quad (50)$$

$$\nabla \mathcal{L}_{m+l_2}(\boldsymbol{\theta}^j)^\intercal = \nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal + \left( \nabla \frac{\alpha}{2} \sum_{j=1}^{n} w_j^2 \right)^\intercal, \quad (47)$$

we have

$$\nabla \mathcal{L}_{m+l_2}(\boldsymbol{\theta}^j)^\intercal = -\frac{2}{|\mathbf{mb}^j|} \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix}^\intercal (\mathbf{y}^j - \widehat{\mathbf{y}^j}) + \alpha \begin{bmatrix} 0 & w_1 \cdots w_n \end{bmatrix}^\intercal. \quad (51)$$

- By substituting eq. (51) into eq. (19)

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j - \eta_k \mathbf{g}_k^j = \boldsymbol{\theta}_k^j - \eta_k \left. \nabla \mathcal{L}_{m+l_2}(\boldsymbol{\theta}^j)^\intercal \right|_{\boldsymbol{\theta}^j = \boldsymbol{\theta}_k^j}, \quad (19)$$

we have

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j + \eta_k \left( \frac{2}{|\mathbf{mb}^j|} \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix}^\intercal (\mathbf{y}^j - \widehat{\mathbf{y}^j}) - \alpha \begin{bmatrix} 0 & w_1 \cdots w_n \end{bmatrix}^\intercal \right), \quad (52)$$

which proves the claim in eq. (20) on page 28. □

# Proof of Updating Rule: Page 32

- The MBGD + elastic net loss can be written as

$$\underbrace{\mathcal{L}_{m+l_{12}}(\boldsymbol{\theta}^j)}_{\text{MBGD + elastic net loss}} = \underbrace{\mathcal{L}(\boldsymbol{\theta}^j)}_{\text{MBGD loss}} \underbrace{+\alpha\gamma\sum_{j=1}^{n}|w_j| + \frac{\alpha(1-\gamma)}{2}\sum_{j=1}^{n}w_j^2}_{\text{elastic net term}}, \tag{53}$$

where the MBGD loss, $\mathcal{L}(\boldsymbol{\theta}^j)$, was given in eq. (13)

$$\mathcal{L}(\boldsymbol{\theta}^j) = \frac{1}{|\mathbf{mb}^j|}\sum_{i\in\mathbf{mb}^j}(y^i - \widehat{y^i})^2. \tag{13}$$

- The gradient of MBGD + elastic net loss, $\nabla\mathcal{L}_{m+l_{12}}(\boldsymbol{\theta}^j)^\intercal$, can be written as the sum of the gradient of MBGD loss (second item in eq. (53)) and the gradient of elastic net term (third item):

$$\nabla\mathcal{L}_{m+l_{12}}(\boldsymbol{\theta}^j)^\intercal = \nabla\mathcal{L}(\boldsymbol{\theta}^j)^\intercal + \left(\nabla\alpha\gamma\sum_{j=1}^{n}|w_j|\right)^\intercal + \left(\nabla\frac{\alpha(1-\gamma)}{2}\sum_{j=1}^{n}w_j^2\right)^\intercal. \tag{54}$$

# Proof of Updating Rule: Page 32

- The gradient of MBGD loss was given in eq. (39)

$$\nabla \mathcal{L}(\boldsymbol{\theta}^j)^\intercal = -\frac{2}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} (y^i - \widehat{y^i}) \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix}^\intercal = -\frac{2}{|\mathbf{mb}^j|} \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix}^\intercal (\mathbf{y}^j - \widehat{\mathbf{y}^j}). \tag{39}$$

- Based on the gradient of lasso term and ridge term given in eqs. (43) and (50)

$$\left(\nabla \alpha \sum_{j=1}^{n} |w_j|\right)^\intercal = \alpha \begin{bmatrix} 0 & \mathrm{sgn}(w_1) \cdots \mathrm{sgn}(w_n) \end{bmatrix}^\intercal, \tag{43}$$

$$\left(\nabla \frac{\alpha}{2} \sum_{j=1}^{n} w_j^2\right)^\intercal = \alpha \begin{bmatrix} 0 & w_1 \cdots w_n \end{bmatrix}^\intercal, \tag{50}$$

we can write the gradient of elastic net term as

$$\left(\nabla \alpha\gamma \sum_{j=1}^{n} |w_j|\right)^\intercal + \left(\nabla \frac{\alpha(1-\gamma)}{2} \sum_{j=1}^{n} w_j^2\right)^\intercal = \alpha\gamma \begin{bmatrix} 0 & \mathrm{sgn}(w_1) \cdots \mathrm{sgn}(w_n) \end{bmatrix}^\intercal + \alpha(1-\gamma) \begin{bmatrix} 0 & w_1 \cdots w_n \end{bmatrix}^\intercal. \tag{55}$$

# Proof of Updating Rule: Page 32

- By substituting eqs. (39) and (55) into eq. (54),

$$\nabla \mathcal{L}(\boldsymbol{\theta}^j)^\top = -\frac{2}{|\mathbf{mb}^j|} \sum_{i \in \mathbf{mb}^j} (y^i - \widehat{y^i}) \begin{bmatrix} 1 & \mathbf{x}^i \end{bmatrix}^\top = -\frac{2}{|\mathbf{mb}^j|} \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix}^\top (\mathbf{y}^j - \widehat{\mathbf{y}^j}), \tag{39}$$

$$\left(\nabla \alpha\gamma \sum_{j=1}^{n} |w_j|\right)^\top + \left(\nabla \frac{\alpha(1-\gamma)}{2} \sum_{j=1}^{n} w_j^2\right)^\top = \alpha\gamma \begin{bmatrix} 0 & \mathrm{sgn}(w_1) \cdots \mathrm{sgn}(w_n) \end{bmatrix}^\top + \alpha(1-\gamma) \begin{bmatrix} 0 & w_1 \cdots w_n \end{bmatrix}^\top, \tag{55}$$

$$\nabla \mathcal{L}_{m+l_{12}}(\boldsymbol{\theta}^j)^\top = \nabla \mathcal{L}(\boldsymbol{\theta}^j)^\top + \left(\nabla \alpha\gamma \sum_{j=1}^{n} |w_j|\right)^\top + \left(\nabla \frac{\alpha(1-\gamma)}{2} \sum_{j=1}^{n} w_j^2\right)^\top, \tag{54}$$

we have

$$\nabla \mathcal{L}_{m+l_2}(\boldsymbol{\theta}^j)^\top = -\frac{2}{|\mathbf{mb}^j|} \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix}^\top (\mathbf{y}^j + \widehat{\mathbf{y}^j}) + \alpha\gamma \begin{bmatrix} 0 & \mathrm{sgn}(w_1) \cdots \mathrm{sgn}(w_n) \end{bmatrix}^\top + \alpha(1-\gamma) \begin{bmatrix} 0 & w_1 \cdots w_n \end{bmatrix}^\top. \tag{56}$$

- By substituting eq. (56) into eq. (22)

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j - \eta_k \mathbf{g}_k^j = \boldsymbol{\theta}_k^j - \eta_k \left. \nabla \mathcal{L}_{m+l_2}(\boldsymbol{\theta}^j)^\top \right|_{\boldsymbol{\theta}^j = \boldsymbol{\theta}_k^j}, \tag{22}$$

we have

$$\boldsymbol{\theta}_k^{j+1} = \boldsymbol{\theta}_k^j + \eta_k \left( \frac{2\eta_k}{|\mathbf{mb}^j|} \begin{bmatrix} \mathbf{1} & \mathbf{X}^j \end{bmatrix}^\top (\mathbf{y}^j - \widehat{\mathbf{y}^j}) - \alpha\gamma \begin{bmatrix} 0 & \mathrm{sgn}(w_1) \cdots \mathrm{sgn}(w_n) \end{bmatrix}^\top - \alpha(1-\gamma) \begin{bmatrix} 0 & w_1 \cdots w_n \end{bmatrix}^\top \right), \tag{57}$$

which proves the claim in eq. (23) on page 32. □

# Bibliography