W3. Linear Models, Regularization, and Hyperparameter Tuning

Guang Cheng

University of California, Los Angeles

guangcheng@ucla.edu

Week 3

An Example: unemployment rate prediction

You have a dataset with various economic indicators and want to predict the unemployment rate Y. The dataset includes features such as:

- X_1: GDP (Gross Domestic Product)

An Example: unemployment rate prediction

You have a dataset with various economic indicators and want to predict the unemployment rate Y. The dataset includes features such as:

- X_1: GDP (Gross Domestic Product)
- X_2: Inflation rate

An Example: unemployment rate prediction

You have a dataset with various economic indicators and want to predict the unemployment rate Y. The dataset includes features such as:

- X_1: GDP (Gross Domestic Product)
- X_2: Inflation rate
- X_3: Education level of the population

An Example: unemployment rate prediction

You have a dataset with various economic indicators and want to predict the unemployment rate Y. The dataset includes features such as:

- X_1: GDP (Gross Domestic Product)
- X_2: Inflation rate
- X_3: Education level of the population
- X_4: Average income

An Example: unemployment rate prediction

You have a dataset with various economic indicators and want to predict the unemployment rate Y. The dataset includes features such as:

- X_1: GDP (Gross Domestic Product)
- X_2: Inflation rate
- X_3: Education level of the population
- X_4: Average income
- X_5: Public spending on infrastructure

Some interesting observations in this dataset

- Some of these features might be correlated, and this sort of multicollinearity could be impacting the performance of a traditional linear regression model;

Some interesting observations in this dataset

- Some of these features might be correlated, and this sort of multicollinearity could be impacting the performance of a traditional linear regression model;
- For better interpretability, you want to identify the most influential factors contributing to the unemployment rate.

Linear regression modelling

- We model the data in the following linear regression

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \epsilon$$

Linear regression modelling

- We model the data in the following linear regression

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \epsilon$$

- Meanwhile, we want to have a sparse estimate $\widehat{\boldsymbol{\beta}} = (\widehat{\beta}_0, \ldots, \widehat{\beta}_5)$ in the sense that some $\widehat{\beta}_i$'s are zero, while still maintaining accurate prediction of Y.

Linear regression modelling

- We model the data in the following linear regression

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \epsilon$$

- Meanwhile, we want to have a sparse estimate $\widehat{\boldsymbol{\beta}} = (\widehat{\beta}_0, \ldots, \widehat{\beta}_5)$ in the sense that some $\widehat{\beta}_i$'s are zero, while still maintaining accurate prediction of Y.
- We need spare estimate for two reasons:
  (i) remove those features that are highly correlated with each other;
  (ii) enhance the interpretability of the prediction model where only a few features determine the unemployment rate.

One solution: shrinkage methods

- Shrinkage methods, in the context of statistical modeling and machine learning, are techniques that involve intentionally reducing the impact or size of certain parameters or coefficients in a model. These methods are particularly useful for preventing overfitting, handling multicollinearity, and improving model interpretability.

One solution: shrinkage methods

- Shrinkage methods, in the context of statistical modeling and machine learning, are techniques that involve intentionally reducing the impact or size of certain parameters or coefficients in a model. These methods are particularly useful for preventing overfitting, handling multicollinearity, and improving model interpretability.
- One price to pay is that shrinkage/regularization methods often introduce some estimation bias;

Shrinkage methods

- Shrinkage methods are formulated as

$$(\hat{\beta}_0, \hat{\beta}) = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \beta_0 - \mathbf{x}_i^T \beta)^2 + \lambda J(\beta)$$

Shrinkage methods

- Shrinkage methods are formulated as

$$(\hat{\beta}_0, \hat{\beta}) = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \beta_0 - \mathbf{x}_i^T \beta)^2 + \lambda J(\beta)$$

- Various choices of $J(\beta)$ lead to different shrinkage methods and possess different properties

Shrinkage methods

- Shrinkage methods are formulated as

$$(\hat{\beta}_0, \hat{\beta}) = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \beta_0 - \mathbf{x}_i^T \beta)^2 + \lambda J(\beta)$$

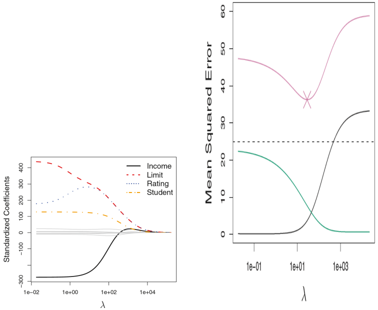
- Various choices of $J(\beta)$ lead to different shrinkage methods and possess different properties

- After centralization, it becomes

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \beta)^2 + \lambda J(\beta)$$

Deeper understanding of shrinkage estimate

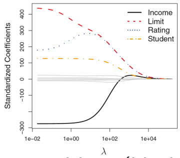
- In general, any shrinkage estimate $\hat{\beta}_\lambda$ is biased in the sense that its mean is not the same as true β.

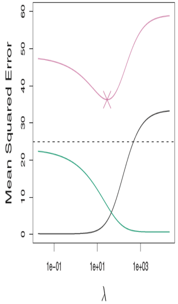


Right panel: squared bias (black), variance (green), test error (purple)

Deeper understanding of shrinkage estimate

- In general, any shrinkage estimate $\hat{\beta}_\lambda$ is biased in the sense that its mean is not the same as true β.
- As for the impact of the tuning parameter λ, we can see from the following two plots





Right panel: squared bias (black), variance (green), test error (purple)

Shrinkage method I: ridge regression

- Ridge regression uses an L_2-norm penalty, $\|\beta\|^2 = \sum_{j=1}^{p} \beta_j^2 = \beta^T \beta$,

$$\hat{\beta}_\lambda^{ridge} = \underset{\beta}{\operatorname{argmin}} \, (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) + \lambda\|\beta\|^2$$

Shrinkage method I: ridge regression

- Ridge regression uses an L_2-norm penalty, $\|\beta\|^2 = \sum_{j=1}^{p} \beta_j^2 = \beta^T\beta$,

$$\hat{\beta}_{\lambda}^{ridge} = \underset{\beta}{\operatorname{argmin}} \, (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) + \lambda\|\beta\|^2$$

- The second term, $\lambda\|\beta\|^2$, is a shrinkage penalty, which shrinks the estimates of β towards zero

Shrinkage method I: ridge regression

- Ridge regression uses an L_2-norm penalty, $\|\beta\|^2 = \sum_{j=1}^{p} \beta_j^2 = \beta^T\beta$,

$$\hat{\beta}_\lambda^{ridge} = \underset{\beta}{\operatorname{argmin}} \, (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) + \lambda\|\beta\|^2$$

- The second term, $\lambda\|\beta\|^2$, is a shrinkage penalty, which shrinks the estimates of β towards zero
- The tuning parameter $\lambda > 0$ controls the trade-off between regression fitting and coefficient shrinkage

Shrinkage method I: ridge regression

- Ridge regression uses an L_2-norm penalty, $\|\beta\|^2 = \sum_{j=1}^{p} \beta_j^2 = \beta^T\beta$,

$$\hat{\beta}_{\lambda}^{ridge} = \underset{\beta}{\operatorname{argmin}} \, (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) + \lambda\|\beta\|^2$$

- The second term, $\lambda\|\beta\|^2$, is a shrinkage penalty, which shrinks the estimates of β towards zero
- The tuning parameter $\lambda > 0$ controls the trade-off between regression fitting and coefficient shrinkage
- If $\lambda = 0$, ridge regression produces the oridnary linear regression; if $\lambda \to \infty$, all estimates of β_i's are zero

Solution of ridge regression

- Solution of the ridge regression is (by make first order derivative to be zero!). try to derive it in class

$$\hat{\beta}_{\lambda}^{ridge} = (\mathbf{X}^T\mathbf{X} + \lambda I_p)^{-1}\mathbf{X}^T\mathbf{y}$$

Solution of ridge regression

How to do the derivations ?

- Objective Function:

$$\hat{\beta}_{\lambda}^{ridge} = \underset{\beta}{\operatorname{argmin}} \, (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) + \lambda\|\beta\|^2$$

Solution of ridge regression

How to do the derivations ?

- Objective Function:

$$\hat{\beta}_{\lambda}^{ridge} = \underset{\beta}{\operatorname{argmin}} \, (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) + \lambda\|\beta\|^2$$

- Expanded First Term:

$$(\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) = \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\beta - \beta^T\mathbf{X}^T\mathbf{y} + \beta^T\mathbf{X}^T\mathbf{X}\beta$$

Solution of ridge regression

How to do the derivations ?

- Objective Function:

$$\hat{\beta}_{\lambda}^{ridge} = \underset{\beta}{\operatorname{argmin}} \, (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) + \lambda\|\beta\|^2$$

- Expanded First Term:

$$(\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) = \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\beta - \beta^T\mathbf{X}^T\mathbf{y} + \beta^T\mathbf{X}^T\mathbf{X}\beta$$

- Objective Function with Expanded Terms:

$$J(\beta) = \mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\mathbf{X}\beta + \beta^T\mathbf{X}^T\mathbf{X}\beta + \lambda\beta^T\beta$$

Solution of ridge regression

How to do the derivations ?

- Differentiation and Setting to Zero:

$$\frac{\partial J}{\partial \beta} = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\beta + 2\lambda\beta = 0$$

Solution of ridge regression

How to do the derivations ?

- Differentiation and Setting to Zero:

$$\frac{\partial J}{\partial \beta} = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\beta + 2\lambda\beta = 0$$

- Rearranging to Solve for β

$$(\mathbf{X}^T\mathbf{X} + \lambda I)\beta = \mathbf{X}^T\mathbf{y}$$

Solution of ridge regression

How to do the derivations ?

- Differentiation and Setting to Zero:

$$\frac{\partial J}{\partial \beta} = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\beta + 2\lambda\beta = 0$$

- Rearranging to Solve for β

$$(\mathbf{X}^T\mathbf{X} + \lambda I)\beta = \mathbf{X}^T\mathbf{y}$$

- Final Ridge Regression Estimate:

$$\hat{\beta}_\lambda^{ridge} = (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\mathbf{y}$$

Solution of ridge regression

- An equivalent formulation,

$$\hat{\beta}_\lambda^{ridge} = \operatorname*{argmin}_{\beta} (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta)$$
$$\text{subject to } \|\beta\|^2 \le s$$

Shrinkage method II: Lasso

- The lasso uses an L_1-norm penalty, $\|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|$,

$$\hat{\beta}^{lasso} = \underset{\beta}{\operatorname{argmin}} \, (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) + \lambda\|\beta\|_1$$

Shrinkage method II: Lasso

- The lasso uses an L_1-norm penalty, $\|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|$,

$$\hat{\beta}^{lasso} = \underset{\beta}{\operatorname{argmin}}\ (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) + \lambda\|\beta\|_1$$

- Or equivalently,

$$\begin{aligned} \hat{\beta}^{lasso} &= \underset{\beta}{\operatorname{argmin}}\ (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) \\ &\quad \text{subject to } \|\beta\|_1 \leq s \end{aligned}$$

Shrinkage method II: Lasso

- The lasso uses an L_1-norm penalty, $\|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|$,

$$\hat{\beta}^{lasso} = \underset{\beta}{\operatorname{argmin}} \, (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) + \lambda\|\beta\|_1$$

- Or equivalently,

$$\begin{aligned} \hat{\beta}^{lasso} &= \underset{\beta}{\operatorname{argmin}} \, (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) \\ & \quad \text{subject to } \|\beta\|_1 \le s \end{aligned}$$

- In comparison with ridge estimate, there is no explicit solution for Lasso. Rather, a quadratic programming (QP) algorithm can be used to solve the above optimization problem.

Example: Prostate cancer - background

- Prostate cancer is a type of cancer that occurs in the prostate, a small walnut-shaped gland in men that produces seminal fluid. Early detection and accurate diagnosis of prostate cancer are crucial for effective treatment and improving patient outcomes.

Example: Prostate cancer - background

- Prostate cancer is a type of cancer that occurs in the prostate, a small walnut-shaped gland in men that produces seminal fluid. Early detection and accurate diagnosis of prostate cancer are crucial for effective treatment and improving patient outcomes.
- Machine learning models applied to datasets containing clinical information can assist in predicting the likelihood of prostate cancer based on relevant features.

Example: Prostate cancer - background

- Clinical Features:

Example: Prostate cancer - background

- Clinical Features:
    - Age of the patient.

Example: Prostate cancer - background

- Clinical Features:
  - Age of the patient.
  - Prostate-specific antigen (PSA) levels in the blood.

Example: Prostate cancer - background

- Clinical Features:
    - Age of the patient.
    - Prostate-specific antigen (PSA) levels in the blood.
    - Biopsy Gleason scores, which characterize the aggressiveness of prostate cancer cells based on their microscopic appearance.

Example: Prostate cancer - background

- Clinical Features:
    - Age of the patient.
    - Prostate-specific antigen (PSA) levels in the blood.
    - Biopsy Gleason scores, which characterize the aggressiveness of prostate cancer cells based on their microscopic appearance.
    - Other features (we do not introduce each one of them)

Example: Prostate cancer - background

- Clinical Features:
    - Age of the patient.
    - Prostate-specific antigen (PSA) levels in the blood.
    - Biopsy Gleason scores, which characterize the aggressiveness of prostate cancer cells based on their microscopic appearance.
    - Other features (we do not introduce each one of them)
- Target Variable:

Example: Prostate cancer - background

- Clinical Features:
    - Age of the patient.
    - Prostate-specific antigen (PSA) levels in the blood.
    - Biopsy Gleason scores, which characterize the aggressiveness of prostate cancer cells based on their microscopic appearance.
    - Other features (we do not introduce each one of them)
- Target Variable:
    - Binary outcome indicating the presence or absence of prostate cancer.

Example: Prostate cancer - background

- Clinical Features:
  - Age of the patient.
  - Prostate-specific antigen (PSA) levels in the blood.
  - Biopsy Gleason scores, which characterize the aggressiveness of prostate cancer cells based on their microscopic appearance.
  - Other features (we do not introduce each one of them)
- Target Variable:
  - Binary outcome indicating the presence or absence of prostate cancer.
- In some cases, additional information may be available, such as the cancer stage.

Example: Prostate cancer - experiment

- A logistic regression model is trained to predict the diagnosis of prostate cancer based on the provided features.

Example: Prostate cancer - experiment

- A logistic regression model is trained to predict the diagnosis of prostate cancer based on the provided features.
  - In the logistic regression, Y would be the binary outcome indicating the presence or absence of cancer.

Example: Prostate cancer - experiment

- A logistic regression model is trained to predict the diagnosis of prostate cancer based on the provided features.
    - In the logistic regression, Y would be the binary outcome indicating the presence or absence of cancer.
    - X would be the set of predictors or features like age, PSA level, etc.

Example: Prostate cancer - experiment

- A logistic regression model is trained to predict the diagnosis of prostate cancer based on the provided features.
    - In the logistic regression, Y would be the binary outcome indicating the presence or absence of cancer.
    - X would be the set of predictors or features like age, PSA level, etc.
    - Lasso is used to identify the most important predictors and eliminate those that are not as important.

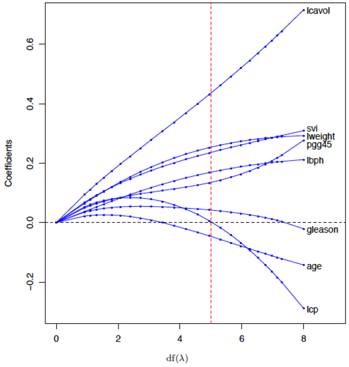
Example: Prostate cancer - experiment

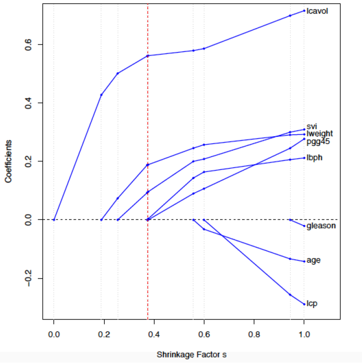
- A logistic regression model is trained to predict the diagnosis of prostate cancer based on the provided features.
  - In the logistic regression, Y would be the binary outcome indicating the presence or absence of cancer.
  - X would be the set of predictors or features like age, PSA level, etc.
  - Lasso is used to identify the most important predictors and eliminate those that are not as important.
  - Ridge is used to handle multicollinearity and enhance model prediction by shrinking the coefficients.

Example: Prostate cancer - experiment

- A logistic regression model is trained to predict the diagnosis of prostate cancer based on the provided features.
  - In the logistic regression, Y would be the binary outcome indicating the presence or absence of cancer.
  - X would be the set of predictors or features like age, PSA level, etc.
  - Lasso is used to identify the most important predictors and eliminate those that are not as important.
  - Ridge is used to handle multicollinearity and enhance model prediction by shrinking the coefficients.
- We use ridge regression and lasso regression to fit the data, and plot the coefficients of each under different regularized parameters.

Example: Prostate cancer - results





Left: ridge regression; Right: lasso regression

Example: Prostate cancer - explanation

- model's degree of freedom $df(\lambda)$

Example: Prostate cancer - explanation

- model's degree of freedom $df(\lambda)$
  - $df(\lambda)$ represents the effective degrees of freedom as a function of the regularization parameter λ.

$$df(\lambda) = \text{trace}(S) = \text{trace}[X(X^T X + \lambda I)^{-1} X^T]$$

Example: Prostate cancer - explanation

- model's degree of freedom $df(\lambda)$
  - $df(\lambda)$ represents the effective degrees of freedom as a function of the regularization parameter λ.

$$df(\lambda) = \text{trace}(S) = \text{trace}[X(X^TX + \lambda I)^{-1}X^T]$$

  - This definition reflects the fact that as λ increases, each predictor's influence on the response variable is reduced, thus reducing the effective degrees of freedom.

Example: Prostate cancer - Explanation

- Shrinkage factor s

Example: Prostate cancer - Explanation

- Shrinkage factor s
  - s represents the shrinkage factor of the regularization parameter λ.

$$s = \frac{1}{1+\lambda}$$

Example: Prostate cancer - Explanation

- Shrinkage factor s
  - s represents the shrinkage factor of the regularization parameter λ.

$$s = \frac{1}{1+\lambda}$$

  - As λ approaches infinity, the shrinkage factor s approaches 0, leading to maximal shrinkage. Conversely, as λ approaches 0, the shrinkage factor s approaches 1, implying minimal shrinkage and the model approaches the ordinary least squares solution.

Example: Prostate cancer - Interpretation

- For ridge regression, the regularized term will shrink the coefficients to some extent, but will not make the coefficients to become sparse.

Example: Prostate cancer - Interpretation

- For ridge regression, the regularized term will shrink the coefficients to some extent, but will not make the coefficients to become sparse.
- For lasso regression, the regularized term will make the coefficients sparse (many of them become zero when choosing appropriate regularized term).

Ridge vs Lasso

- Both lasso and ridge regression will shrink estimated coefficients while introducing some bias

Ridge vs Lasso

- Both lasso and ridge regression will shrink estimated coefficients while introducing some bias
- The lasso produces more sparse and more interpretable models that involve only a subset of predictors

Ridge vs Lasso

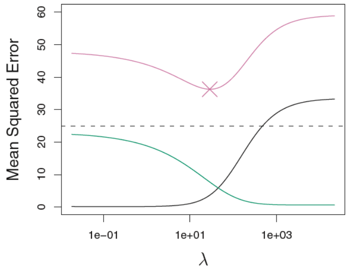
- Both lasso and ridge regression will shrink estimated coefficients while introducing some bias
- The lasso produces more sparse and more interpretable models that involve only a subset of predictors
- It is unclear which one leads to better prediction accuracy in general though. But we can compare in some special case, i.e., orthogonal case in the next page.

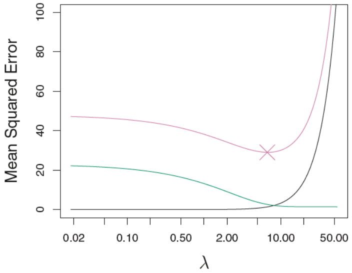
Ridge vs Lasso

- Both lasso and ridge regression will shrink estimated coefficients while introducing some bias
- The lasso produces more sparse and more interpretable models that involve only a subset of predictors
- It is unclear which one leads to better prediction accuracy in general though. But we can compare in some special case, i.e., orthogonal case in the next page.

Ridge vs Lasso

- Both lasso and ridge regression will shrink estimated coefficients while introducing some bias
- The lasso produces more sparse and more interpretable models that involve only a subset of predictors
- It is unclear which one leads to better prediction accuracy in general though. But we can compare in some special case, i.e., orthogonal case in the next page.





An orthogonal case

Consider a simple case with $n = p$ and $\mathbf{X} = \mathbf{I}_p$, then $\hat{\beta}_j^{ols} = y_j$,

- Ridge estimate:

$$\hat{\beta}_j^{ridge} = y_j/(1+\lambda)$$

An orthogonal case

Consider a simple case with $n = p$ and $\mathbf{X} = \mathbf{I}_p$, then $\hat{\beta}_j^{ols} = y_j$,

- Ridge estimate:
$$\hat{\beta}_j^{ridge} = y_j/(1+\lambda)$$
- Lasso estimate: $(a_+ = a$ for $a > 0$; $a_+ = 0$ otherwise)
$$\hat{\beta}_j^{lasso} = \mathrm{sign}(y_j)(|y_j| - \lambda/2)_+$$

An orthogonal case

Consider a simple case with $n = p$ and $\mathbf{X} = \mathbf{I}_p$, then $\hat{\beta}_j^{ols} = y_j$,

- Ridge estimate:
$$\hat{\beta}_j^{ridge} = y_j/(1+\lambda)$$
- Lasso estimate: $(a_+ = a$ for $a > 0$; $a_+ = 0$ otherwise)
$$\hat{\beta}_j^{lasso} = \mathrm{sign}(y_j)(|y_j| - \lambda/2)_+$$

An orthogonal case

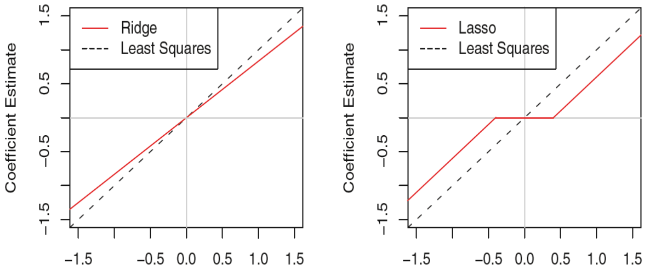
Consider a simple case with $n = p$ and $\mathbf{X} = \mathbf{I}_p$, then $\hat{\beta}_j^{ols} = y_j$,

- Ridge estimate:

$$\hat{\beta}_j^{ridge} = y_j/(1+\lambda)$$

- Lasso estimate: $(a_+ = a$ for $a > 0$; $a_+ = 0$ otherwise$)$

$$\hat{\beta}_j^{lasso} = \mathrm{sign}(y_j)(|y_j| - \lambda/2)_+$$



Elastic net: combination of ridge and lasso

- Elastic Net is a regularization technique that combines both Lasso Regression (L1 regularization) and Ridge Regression (L2 regularization) in an effort to leverage the benefits of both methods.

Elastic net: combination of ridge and lasso

- Elastic Net is a regularization technique that combines both Lasso Regression (L1 regularization) and Ridge Regression (L2 regularization) in an effort to leverage the benefits of both methods.
- It was introduced to address some limitations of Lasso Regression, such as its tendency to arbitrarily select one feature among a group of highly correlated features. Elastic Net introduces an additional hyperparameter to control the mixture of L1 and L2 regularization: its penalty is of the form

$$\lambda_1\|\beta\|_1 + \lambda_2\|\beta\|^2.$$

A general regularization: bridge estimators

With $L_r(\beta) = \sum_{j=1}^{p} |\beta_j|^r$,

$$\hat{\beta}^{bridge} = \underset{\beta}{\operatorname{argmin}} \| \mathbf{y} - \mathbf{X}\beta\|^2 + \lambda L_r(\beta)$$

- $L_0(\beta) = \sum_{j=1}^{p} I(\beta_j \neq 0)$; (Hard thresholding)

A general regularization: bridge estimators

With $L_r(\beta) = \sum_{j=1}^{p} |\beta_j|^r$,

$$\hat{\beta}^{bridge} = \underset{\beta}{\operatorname{argmin}} \| \mathbf{y} - \mathbf{X}\,\beta\|^2 + \lambda L_r(\beta)$$

- $L_0(\beta) = \sum_{j=1}^{p} I(\beta_j \neq 0)$; (Hard thresholding)
- $L_1(\beta) = \sum_{j=1}^{p} |\beta_j|$; (Lasso)

A general regularization: bridge estimators

With $L_r(\beta) = \sum_{j=1}^{p} |\beta_j|^r$,

$$\hat{\beta}^{bridge} = \operatorname*{argmin}_{\beta} \| \mathbf{y} - \mathbf{X}\beta\|^2 + \lambda L_r(\beta)$$

- $L_0(\beta) = \sum_{j=1}^{p} I(\beta_j \neq 0)$; (Hard thresholding)
- $L_1(\beta) = \sum_{j=1}^{p} |\beta_j|$; (Lasso)
- $L_2(\beta) = \sum_{j=1}^{p} \beta_j^2$; (Ridge regression)

A general regularization: bridge estimators

With $L_r(\beta) = \sum_{j=1}^{p} |\beta_j|^r$,

$$\hat{\beta}^{bridge} = \underset{\beta}{\operatorname{argmin}} \| \mathbf{y} - \mathbf{X}\beta\|^2 + \lambda L_r(\beta)$$

- $L_0(\beta) = \sum_{j=1}^{p} I(\beta_j \neq 0)$; (Hard thresholding)
- $L_1(\beta) = \sum_{j=1}^{p} |\beta_j|$; (Lasso)
- $L_2(\beta) = \sum_{j=1}^{p} \beta_j^2$; (Ridge regression)
- $L_\infty(\beta) = \max_j |\beta_j|$.

The penalty form

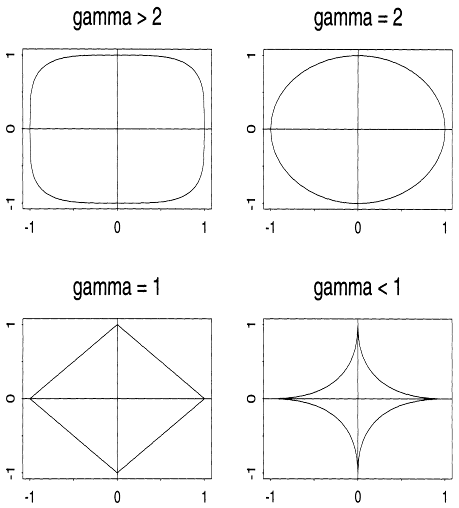


Figure 1. Constrained Areas of Bridge Regressions with t = 1.

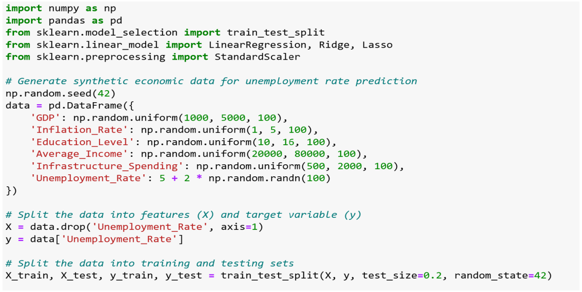
Revisit unemployment rate prediction example

- Back to the unemployment rate prediction example

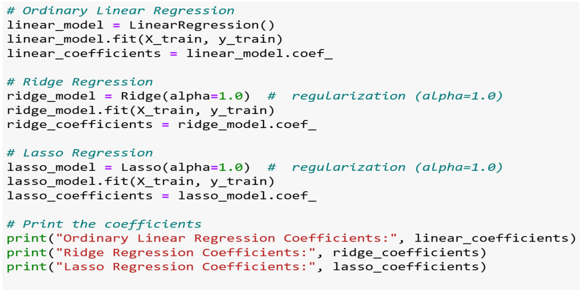
Revisit unemployment rate prediction example

- Back to the unemployment rate prediction example
- Let's take a look at the coefficients obtained using these three methods (ordinary, ridge and lasso regression) respectively.

Codes: Linear regression vs Ridge vs Lasso

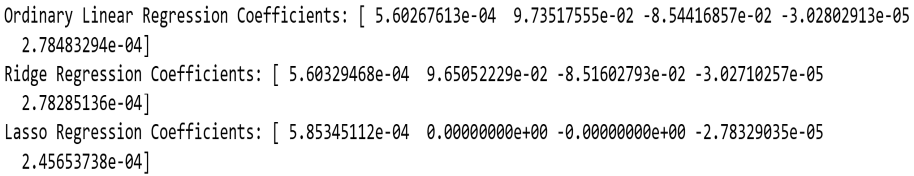


```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.preprocessing import StandardScaler

# Generate synthetic economic data for unemployment rate prediction
np.random.seed(42)
data = pd.DataFrame({
    'GDP': np.random.uniform(1000, 5000, 100),
    'Inflation_Rate': np.random.uniform(1, 5, 100),
    'Education_Level': np.random.uniform(10, 16, 100),
    'Average_Income': np.random.uniform(20000, 80000, 100),
    'Infrastructure_Spending': np.random.uniform(500, 2000, 100),
    'Unemployment_Rate': 5 + 2 * np.random.randn(100)
})

# Split the data into features (X) and target variable (y)
X = data.drop('Unemployment_Rate', axis=1)
y = data['Unemployment_Rate']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```



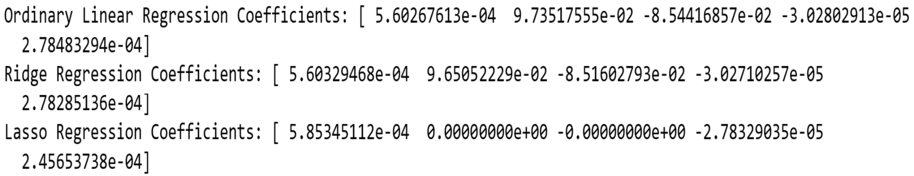
```python
# Ordinary Linear Regression
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)
linear_coefficients = linear_model.coef_

# Ridge Regression
ridge_model = Ridge(alpha=1.0)  #  regularization (alpha=1.0)
ridge_model.fit(X_train, y_train)
ridge_coefficients = ridge_model.coef_

# Lasso Regression
lasso_model = Lasso(alpha=1.0)  #  regularization (alpha=1.0)
lasso_model.fit(X_train, y_train)
lasso_coefficients = lasso_model.coef_

# Print the coefficients
print("Ordinary Linear Regression Coefficients:", linear_coefficients)
print("Ridge Regression Coefficients:", ridge_coefficients)
print("Lasso Regression Coefficients:", lasso_coefficients)
```

Results: Linear regression vs Ridge vs Lasso



```
Ordinary Linear Regression Coefficients: [ 5.60267613e-04  9.73517555e-02 -8.54416857e-02 -3.02802913e-05
  2.78483294e-04]
Ridge Regression Coefficients: [ 5.60329468e-04  9.65052229e-02 -8.51602793e-02 -3.02710257e-05
  2.78285136e-04]
Lasso Regression Coefficients: [ 5.85345112e-04  0.00000000e+00 -0.00000000e+00 -2.78329035e-05
  2.45653738e-04]
```

Results: Linear regression vs Ridge vs Lasso



```
Ordinary Linear Regression Coefficients: [ 5.60267613e-04  9.73517555e-02 -8.54416857e-02 -3.02802913e-05
  2.78483294e-04]
Ridge Regression Coefficients: [ 5.60329468e-04  9.65052229e-02 -8.51602793e-02 -3.02710257e-05
  2.78285136e-04]
Lasso Regression Coefficients: [ 5.85345112e-04  0.00000000e+00 -0.00000000e+00 -2.78329035e-05
  2.45653738e-04]
```

Conclusion

- Ridge regression slightly shrinks the fitting coefficients, while lasso regression makes the fitting coefficients sparse.

Basics in optimization algorithm

- **Question**: If you are asked to minimize or maximize a function $f : \mathbb{R} \to \mathbb{R}$ (such as loss function in our context), what would you do?

Basics in optimization algorithm

- **Question**: If you are asked to minimize or maximize a function $f : \mathbb{R} \to \mathbb{R}$ (such as loss function in our context), what would you do?
- **Answer**: Take the derivative with respect to x and solve the equation

$$\frac{d}{dx} f(x) = 0$$

Basics in optimization algorithm

- **Question**: If you are asked to minimize or maximize a function $f : \mathbb{R} \to \mathbb{R}$ (such as loss function in our context), what would you do?
- **Answer**: Take the derivative with respect to x and solve the equation

$$\frac{d}{dx} f(x) = 0$$

- **Question**: What if there is no explicit solution of the equation $\frac{d}{dx} f(x) = 0$?

Basics in optimization algorithm

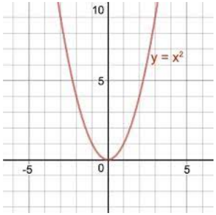
- **Question**: If you are asked to minimize or maximize a function $f : \mathbb{R} \to \mathbb{R}$ (such as loss function in our context), what would you do?
- **Answer**: Take the derivative with respect to x and solve the equation

$$\frac{d}{dx} f(x) = 0$$

- **Question**: What if there is no explicit solution of the equation $\frac{d}{dx} f(x) = 0$?
- **Answer**: In practice, we often adopt the optimization algorithm called as "Gradient ascent" (for maximization) or "Gradient descent" (for minimization).

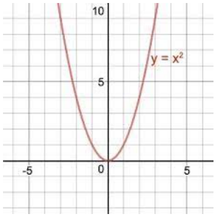
Gradient descent: an example

- Suppose we want to minimize $f(x) = x^2$



Gradient descent: an example

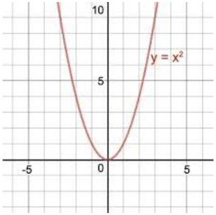
- Suppose we want to minimize $f(x) = x^2$



- First, we randomly choose an initial point $x^{(0)} = 3$

Gradient descent: an example

- Suppose we want to minimize $f(x) = x^2$

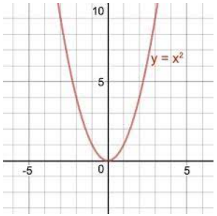


- First, we randomly choose an initial point $x^{(0)} = 3$
- Second, we calculate the derivative at the point $x^{(0)} = 3$

$$f'(3) = 6$$

Gradient descent: an example

- Suppose we want to minimize $f(x) = x^2$



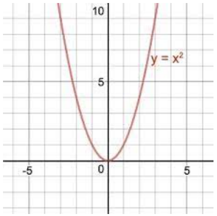
- First, we randomly choose an initial point $x^{(0)} = 3$
- Second, we calculate the derivative at the point $x^{(0)} = 3$

$$f'(3) = 6$$

- Update the new value $x^{(1)} = x^{(0)} - \alpha \cdot f'(3)$, α is the step size, also called as learning rate.

Gradient descent: an example

- Suppose we want to minimize $f(x) = x^2$



- First, we randomly choose an initial point $x^{(0)} = 3$
- Second, we calculate the derivative at the point $x^{(0)} = 3$

$$f'(3) = 6$$

- Update the new value $x^{(1)} = x^{(0)} - \alpha \cdot f'(3)$, α is the step size, also called as learning rate.
- Repeat the above process until convergence.

Gradient descent

- **Motivation of Gradient descent**: gradient provides information to minimize the objective function.

Gradient descent

- **Motivation of Gradient descent**: gradient provides information to minimize the objective function.
- **General form of gradient descent**: Let $f : \mathbb{R}^p \to \mathbb{R}$ be a p-variate function. Then gradient descent has the form

$$\boldsymbol{x}^{(t)} = \boldsymbol{x}^{(t-1)} - \lambda \nabla f(\boldsymbol{x}),$$

where $\nabla f(\boldsymbol{x}) = \left(\frac{\partial}{\partial x_1} f(\boldsymbol{x}), \frac{\partial}{\partial x_2} f(\boldsymbol{x}), \ldots, \frac{\partial}{\partial x_p} f(\boldsymbol{x}) \right)$

Gradient descent

- **Motivation of Gradient descent**: gradient provides information to minimize the objective function.
- **General form of gradient descent**: Let $f\colon \mathbb{R}^p \to \mathbb{R}$ be a p-variate function. Then gradient descent has the form

$$\boldsymbol{x}^{(t)} = \boldsymbol{x}^{(t-1)} - \lambda \nabla f(\boldsymbol{x}),$$

where $\nabla f(\boldsymbol{x}) = \left(\frac{\partial}{\partial x_1} f(\boldsymbol{x}), \frac{\partial}{\partial x_2} f(\boldsymbol{x}), \ldots, \frac{\partial}{\partial x_p} f(\boldsymbol{x}) \right)$

- **Question**: When does gradient descent stop?

Gradient descent

- **Motivation of Gradient descent**: gradient provides information to minimize the objective function.
- **General form of gradient descent**: Let $f : \mathbb{R}^p \to \mathbb{R}$ be a p-variate function. Then gradient descent has the form

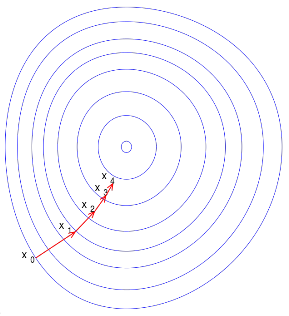
$$\boldsymbol{x}^{(t)} = \boldsymbol{x}^{(t-1)} - \lambda \nabla f(\boldsymbol{x}),$$

where $\nabla f(\boldsymbol{x}) = \left(\frac{\partial}{\partial x_1} f(\boldsymbol{x}), \frac{\partial}{\partial x_2} f(\boldsymbol{x}), \ldots, \frac{\partial}{\partial x_p} f(\boldsymbol{x}) \right)$

- **Question**: When does gradient descent stop?
- **Answer**: When $\nabla f(\boldsymbol{x}^{(t)}) \approx \mathbf{0}$ for a sufficiently large t.

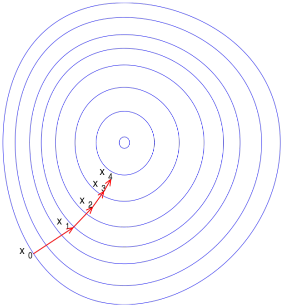
Application of gradient descent

- Gradient descent is usually employed, when optimization problem is non-convex or does not have **analytic solution** or high-dimensional ***x***, for example Deep neural networks (Non-convex)



Application of gradient descent

- Gradient descent is usually employed, when optimization problem is non-convex or does not have **analytic solution** or high-dimensional $\boldsymbol{x}$, for example Deep neural networks (Non-convex)



- However, if the loss function is convex, then the output of Gradient Descent is guaranteed to be the optimal solution.

How to choose learning rate in gradient descent?

- The gradient descent algorithm has different convergence rate with different choices of learning rate α

How to choose learning rate in gradient descent?

- The gradient descent algorithm has different convergence rate with different choices of learning rate α
- See the example in the next page.

Learning rate tuning in gradient descent: example

- Generate synthetic data for linear regression

Learning rate tuning in gradient descent: example

- Generate synthetic data for linear regression
- Compute the function of Mean Squared Error (MSE) loss

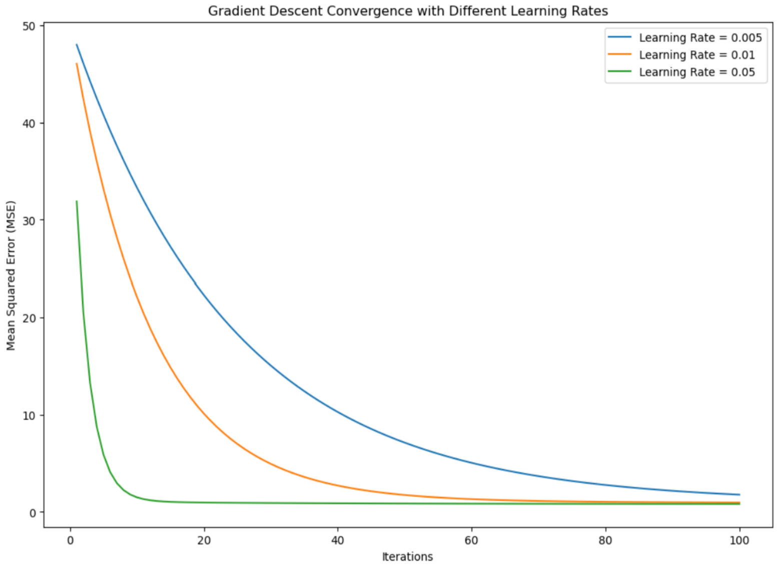
Learning rate tuning in gradient descent: example

- Generate synthetic data for linear regression
- Compute the function of Mean Squared Error (MSE) loss
- Use the gradient descent algorithm to find the minimal

Learning rate tuning in gradient descent: example

- Generate synthetic data for linear regression
- Compute the function of Mean Squared Error (MSE) loss
- Use the gradient descent algorithm to find the minimal
- Use the different learning rate ($\alpha = 0.005, 0.01, 0.05$) for training, and get the plot of MSE versus the iterations.

Learning rate tuning in Gradient Descent: plots



Exercise: Gradient Descent on a Simple Function

- Problem Setup: Consider a simple quadratic function: $f(x) = x^2$. Our goal is to find the value of x that that minimizes $f(x)$ using gradient descent.

Exercise: Gradient Descent on a Simple Function

- Problem Setup: Consider a simple quadratic function: $f(x) = x^2$. Our goal is to find the value of x that that minimizes $f(x)$ using gradient descent.
- Objective:

Exercise: Gradient Descent on a Simple Function

- Problem Setup: Consider a simple quadratic function: $f(x) = x^2$. Our goal is to find the value of x that that minimizes $f(x)$ using gradient descent.
- Objective:
  - Start with an initial guess for x, say $x = 10$.

Exercise: Gradient Descent on a Simple Function

- Problem Setup: Consider a simple quadratic function: $f(x) = x^2$. Our goal is to find the value of x that that minimizes $f(x)$ using gradient descent.
- Objective:
  - Start with an initial guess for x, say $x = 10$.
  - Use gradient descent to find a value of x that minimizes $f(x)$.

Exercise: Gradient Descent on a Simple Function

- Problem Setup: Consider a simple quadratic function: $f(x) = x^2$. Our goal is to find the value of x that that minimizes $f(x)$ using gradient descent.
- Objective:
  - Start with an initial guess for x, say $x = 10$.
  - Use gradient descent to find a value of x that minimizes $f(x)$.
  - Perform three iterations by hand.

Exercise: Gradient Descent on a Simple Function

- Problem Setup: Consider a simple quadratic function: $f(x) = x^2$. Our goal is to find the value of x that that minimizes $f(x)$ using gradient descent.
- Objective:
  - Start with an initial guess for x, say $x = 10$.
  - Use gradient descent to find a value of x that minimizes $f(x)$.
  - Perform three iterations by hand.
  - Assume the learning rate $\alpha = 0.1$.

Exercise: Gradient Descent on a Simple Function

Steps

- Calculate the gradient: The gradient of $f(x) = x^2$ with respect to x is $\frac{df}{dx} = 2x$

Exercise: Gradient Descent on a Simple Function

Steps

- Calculate the gradient: The gradient of $f(x) = x^2$ with respect to x is $\frac{df}{dx} = 2x$
- Update Rule: With our function, the update rule becomes $x = x - \alpha \cdot 2x$

Exercise: Gradient Descent on a Simple Function

Steps

- Calculate the gradient: The gradient of $f(x) = x^2$ with respect to x is $\frac{df}{dx} = 2x$
- Update Rule: With our function, the update rule becomes $x = x - \alpha \cdot 2x$
- Calculation:

Exercise: Gradient Descent on a Simple Function

Steps

- Calculate the gradient: The gradient of $f(x) = x^2$ with respect to x is $\frac{df}{dx} = 2x$
- Update Rule: With our function, the update rule becomes $x = x - \alpha \cdot 2x$
- Calculation:
    - After the 1st iteration: $x = 8.0$.

Exercise: Gradient Descent on a Simple Function

Steps

- Calculate the gradient: The gradient of $f(x) = x^2$ with respect to x is $\frac{df}{dx} = 2x$
- Update Rule: With our function, the update rule becomes $x = x - \alpha \cdot 2x$
- Calculation:
  - After the 1st iteration: $x = 8.0$.
  - After the 2nd iteration: $x = 6.4$.

Exercise: Gradient Descent on a Simple Function

Steps

- Calculate the gradient: The gradient of $f(x) = x^2$ with respect to x is $\frac{df}{dx} = 2x$
- Update Rule: With our function, the update rule becomes $x = x - \alpha \cdot 2x$
- Calculation:
    - After the 1st iteration: $x = 8.0$.
    - After the 2nd iteration: $x = 6.4$.
    - After the 3rd iteration: $x = 5.12$.

Exercise: Gradient Descent on a Simple Function

Steps

- Calculate the gradient: The gradient of $f(x) = x^2$ with respect to x is $\frac{df}{dx} = 2x$
- Update Rule: With our function, the update rule becomes $x = x - \alpha \cdot 2x$
- Calculation:
    - After the 1st iteration: $x = 8.0$.
    - After the 2nd iteration: $x = 6.4$.
    - After the 3rd iteration: $x = 5.12$.
    - .

Exercise: Gradient Descent on a Simple Function

Steps

- Calculate the gradient: The gradient of $f(x) = x^2$ with respect to x is $\frac{df}{dx} = 2x$
- Update Rule: With our function, the update rule becomes $x = x - \alpha \cdot 2x$
- Calculation:
  - After the 1st iteration: $x = 8.0$.
  - After the 2nd iteration: $x = 6.4$.
  - After the 3rd iteration: $x = 5.12$.
  - .
  - .

Exercise: Gradient Descent on a Simple Function

Steps

- Calculate the gradient: The gradient of $f(x) = x^2$ with respect to x is $\frac{df}{dx} = 2x$
- Update Rule: With our function, the update rule becomes $x = x - \alpha \cdot 2x$
- Calculation:
  - After the 1st iteration: $x = 8.0$.
  - After the 2nd iteration: $x = 6.4$.
  - After the 3rd iteration: $x = 5.12$.
  - .
  - .
  - .

Exercise: Gradient Descent on a Simple Function

Steps

- Calculate the gradient: The gradient of $f(x) = x^2$ with respect to x is $\frac{df}{dx} = 2x$
- Update Rule: With our function, the update rule becomes $x = x - \alpha \cdot 2x$
- Calculation:
    - After the 1st iteration: $x = 8.0$.
    - After the 2nd iteration: $x = 6.4$.
    - After the 3rd iteration: $x = 5.12$.
    - .
    - .
    - .
- Conclusion: the value of x keeps going down, and eventually converges to zero.

Problem with Original Gradient Descent (GD)

- **Computational Intensity**: GD requires the computation of gradients for the entire dataset (recall that the loss function is defined on the whole dataset) to perform a single update of the model parameters. This can be extremely computationally intensive and inefficient for large datasets.

Problem with Original Gradient Descent (GD)

- **Computational Intensity**: GD requires the computation of gradients for the entire dataset (recall that the loss function is defined on the whole dataset) to perform a single update of the model parameters. This can be extremely computationally intensive and inefficient for large datasets.
- **Memory Constraints**: Storing the entire dataset in memory for computation can be impractical or impossible with very large datasets, leading to memory constraints.

Problem with Original Gradient Descent (GD)

- **Redundant Calculations**: In real-world data, many samples may be similar or redundant. GD processes the entire dataset in each iteration, leading to redundant calculations that don't significantly contribute to learning.

Problem with Original Gradient Descent (GD)

- **Redundant Calculations**: In real-world data, many samples may be similar or redundant. GD processes the entire dataset in each iteration, leading to redundant calculations that don't significantly contribute to learning.
- **Convergence Speed**: The process of using the entire dataset for each update makes GD slow, particularly for large datasets. This slow convergence can be a major drawback in practice.

Problem with Original Gradient Descent (GD)

- **Redundant Calculations**: In real-world data, many samples may be similar or redundant. GD processes the entire dataset in each iteration, leading to redundant calculations that don't significantly contribute to learning.
- **Convergence Speed**: The process of using the entire dataset for each update makes GD slow, particularly for large datasets. This slow convergence can be a major drawback in practice.
- **Difficulty in Escaping Local Minima**: In high-dimensional and complex error landscapes (common in deep learning), GD can get stuck in local minima or saddle points, especially if the initial parameter values are not optimal.

Problem with Original Gradient Descent (GD)

- How to deal with the above drawbacks ?

Problem with Original Gradient Descent (GD)

- How to deal with the above drawbacks ?
- **Use stochastic gradient descent (SGD) !**

SGD - Basic Concept

- Unlike GD, which uses the entire dataset to compute the gradient of the cost function, SGD updates the model parameters using only a single data point (or a small batch of data points) chosen at random in each iteration.

SGD - Working Mechanism

- In SGD, a random sample (or a mini-batch of samples) is selected from the dataset, and the gradient of the cost function is computed based on this sample only. The model parameters are then updated accordingly. This process is repeated for each sample or mini-batch in the dataset.

SGD - Formula

- The parameter update rule in SGD is given by:

$$\theta = \theta - \alpha \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)})$$

SGD - Formula

- The parameter update rule in SGD is given by:

$$\theta = \theta - \alpha \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)})$$

- In this formula,

SGD - Formula

- The parameter update rule in SGD is given by:

$$\theta = \theta - \alpha \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)})$$

- In this formula,
    - θ represents the model parameters.

SGD - Formula

- The parameter update rule in SGD is given by:

$$\theta = \theta - \alpha \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)})$$

- In this formula,
  - θ represents the model parameters.
  - α is the learning rate.

SGD - Formula

- The parameter update rule in SGD is given by:

$$\theta = \theta - \alpha \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)})$$

- In this formula,
    - θ represents the model parameters.
    - α is the learning rate.
    - $\nabla_\theta J(\theta; x^{(i)}; y^{(i)})$ is the gradient of the loss function with respect to the parameters θ, evaluated at the single data point $(x^{(i)}, y^{(i)})$.

Other Choice: Mini-Batch SGD

- The parameter update rule in Mini-Batch SGD is given by:

$$\theta = \theta - \alpha \cdot \nabla_\theta J(\theta; X^{(i:i+n)}, Y^{(i:i+n)})$$

Other Choice: Mini-Batch SGD

- The parameter update rule in Mini-Batch SGD is given by:

$$\theta = \theta - \alpha \cdot \nabla_\theta J(\theta; X^{(i:i+n)}, Y^{(i:i+n)})$$

- In this formula,

Other Choice: Mini-Batch SGD

- The parameter update rule in Mini-Batch SGD is given by:

$$\theta = \theta - \alpha \cdot \nabla_\theta J(\theta; X^{(i:i+n)}, Y^{(i:i+n)})$$

- In this formula,
  - θ represents the model parameters.

Other Choice: Mini-Batch SGD

- The parameter update rule in Mini-Batch SGD is given by:

$$\theta = \theta - \alpha \cdot \nabla_\theta J(\theta; X^{(i:i+n)}, Y^{(i:i+n)})$$

- In this formula,
  - θ represents the model parameters.
  - α is the learning rate.

Other Choice: Mini-Batch SGD

- The parameter update rule in Mini-Batch SGD is given by:

$$\theta = \theta - \alpha \cdot \nabla_\theta J(\theta; X^{(i:i+n)}, Y^{(i:i+n)})$$

- In this formula,
  - θ represents the model parameters.
  - α is the learning rate.
  - $\nabla_\theta J(\theta; X^{(i:i+n)}, Y^{(i:i+n)})$ is the gradient of the loss function with respect to the parameters θ, computed over a mini-batch of data points. Here, $X^{(i:i+n)}$ and $Y^{(i:i+n)}$ represent the features and labels of the mini-batch, respectively, starting from the i-th data point to the $(i+n)$-th data point.

Why is SGD helpful for training deep neural network?

- **Efficiency with Large Datasets**: Deep learning often involves training on very large datasets. SGD is more efficient compared to traditional batch gradient descent, as it does not require the entire dataset to be loaded into memory or used for each parameter update. Instead, SGD updates parameters using only a small subset of data at a time, making it computationally feasible even with large-scale data.

Why is SGD helpful for training deep neural network?

- **Efficiency with Large Datasets**: Deep learning often involves training on very large datasets. SGD is more efficient compared to traditional batch gradient descent, as it does not require the entire dataset to be loaded into memory or used for each parameter update. Instead, SGD updates parameters using only a small subset of data at a time, making it computationally feasible even with large-scale data.
- **Faster Convergence**: By updating the model parameters more frequently, SGD can converge faster to a solution, especially in complex and high-dimensional spaces typical of deep neural networks. This is crucial in practical applications where training time is a critical factor.

Why is SGD helpful for training deep neural network?

- **Flexibility with Mini-Batch Sizes**: SGD allows for flexibility in choosing the size of mini-batches. This adaptability can lead to a balance between the computational efficiency of true stochastic updates (using very small batches) and the stability of gradient estimates (using larger batches).

Why is SGD helpful for training deep neural network?

- **Flexibility with Mini-Batch Sizes**: SGD allows for flexibility in choosing the size of mini-batches. This adaptability can lead to a balance between the computational efficiency of true stochastic updates (using very small batches) and the stability of gradient estimates (using larger batches).
- **Generalization and Regularization**: The stochastic nature of SGD, where each update is based on a subset of the data, can have a regularizing effect. This can potentially lead to better generalization on unseen data. In other words, models trained with SGD are often less likely to overfit to the training data.

Why is SGD helpful for training deep neural network?

- **Flexibility with Mini-Batch Sizes**: SGD allows for flexibility in choosing the size of mini-batches. This adaptability can lead to a balance between the computational efficiency of true stochastic updates (using very small batches) and the stability of gradient estimates (using larger batches).
- **Generalization and Regularization**: The stochastic nature of SGD, where each update is based on a subset of the data, can have a regularizing effect. This can potentially lead to better generalization on unseen data. In other words, models trained with SGD are often less likely to overfit to the training data.
- **Ability to Escape Local Minima**: In the complex, non-convex error landscapes typical of deep neural networks, SGD's stochasticity helps in escaping local minima. The randomness in selecting data points or batches introduces noise into the optimization process, which can aid in finding a more global minimum.