An introduction to regression

Learning functions which map feature vectors to continuous variables

Statistical Computing and Empirical Methods
Unit EMATM0061, Data Science MSc

Rihuan Ke

rihuan.ke@bristol.ac.uk

Teaching Block 1, 2022

What we will cover today

We will begin by introducing the concept of regression.

We will see how regression fits within the supervised learning paradigm.

We will discuss the mean squared error as a metric for regression problems.

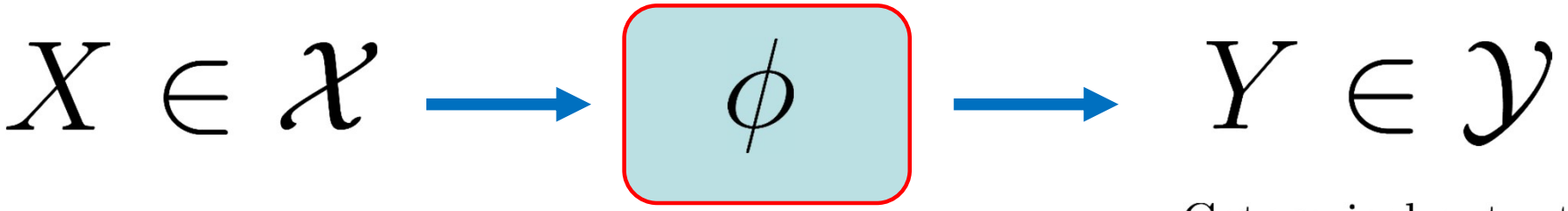
We will introduce the topic of linear regression.

We will discuss the ordinary least squares approach to linear regression.

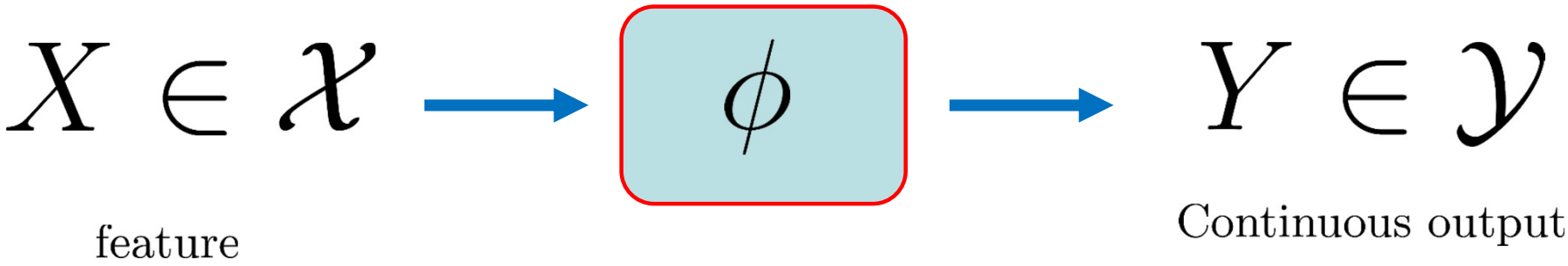
What is regression?

Classification vs Regression

In classification, we want to learn a function $\phi : \mathcal{X} \to \mathcal{Y}$, which takes as input a feature vector $X \in \mathcal{X}$, and returns a categorical variable $\phi(X) \in \mathcal{Y}$ which is also known as a label.



In regression, we want to learn a function $\phi : \mathcal{X} \to \mathbb{R}$, which takes as input a feature vector $X \in \mathcal{X}$, and returns a continuous variable $\phi(X) \in \mathbb{R}$.



Regression: Example

Example 1: Fuel usage

A logistics company wants to estimate the level of fuel consumed by a truck based on the distance travelled, the weight of goods transported, vehicle type etc.

$$X \in \mathcal{X} \longrightarrow \boxed{\phi} \longrightarrow Y \in \mathcal{Y}$$

feature

Continuous output

Regression: Example

Example 2: Weight of fish

A biologist wants to automatically estimate the weight of fish based on an image.

$$X \in \mathcal{X} \longrightarrow \boxed{\phi} \longrightarrow Y \in \mathcal{Y}$$

feature

Continuous output

Regression: Example

Example 3: Temperature prediction

A meteorologist wants to predict the average temperature for the following day based on weather history.

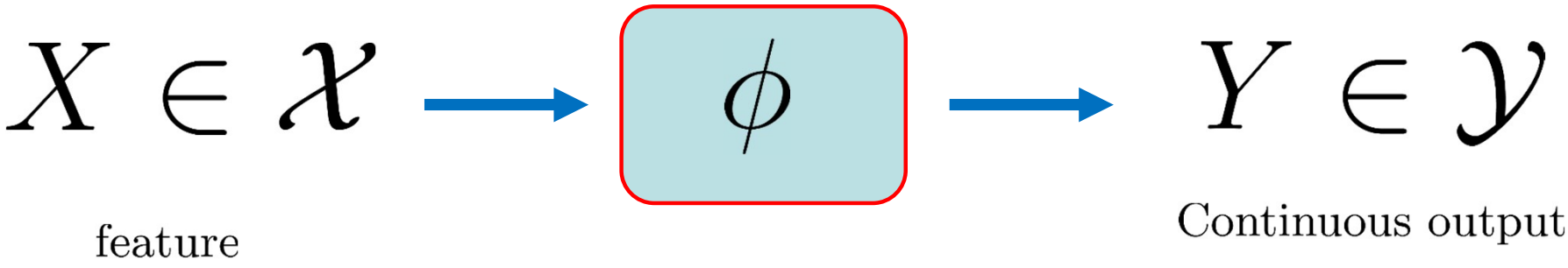
$$X \in \mathcal{X} \longrightarrow \boxed{\phi} \longrightarrow Y \in \mathcal{Y}$$

feature

Continuous output

Regression model

In regression, we want to learn a function $\phi : \mathcal{X} \to \mathbb{R}$, which takes as input a feature vector $X \in \mathcal{X}$, and returns a continuous variable $\phi(X) \in \mathbb{R}$.



The function ϕ is known as a regression model.

Supervised learning for regression

Supervised learning for regression

In regression, we want to learn a function $\phi : \mathcal{X} \to \mathbb{R}$, which takes as input a feature vector $X \in \mathcal{X}$, and returns a continuous variable $\phi(X) \in \mathbb{R}$.

We can find the function ϕ from data.

Supervised learning is the process of learning a function based on training data with feature vectors and labels

Suppose we have training data consisting of a set of labelled data $\mathcal{D} := ((X_1, Y_1), \cdots, (X_n, Y_n))$.

X_i is a feature vector Y_i is a numerical label associated with X_i

Example: Weight of fish

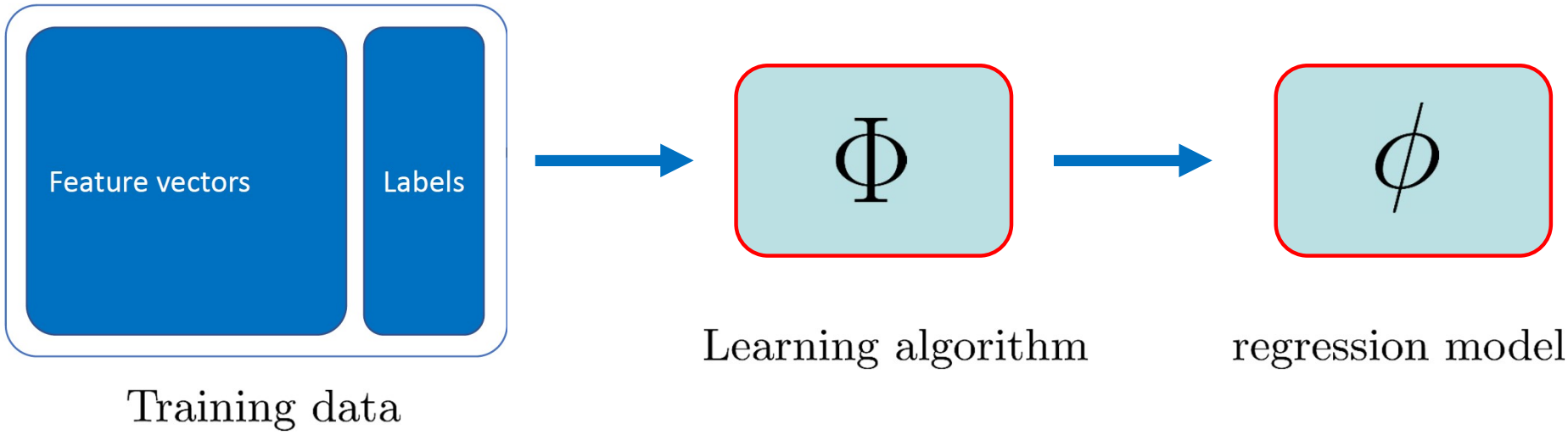
X_i is an image of a particular fish

Y_i is a label corresponding to the weight of the fish.

Supervised learning for regression

Suppose we have training data consisting of a set of labelled data
$\mathcal{D} := ((X_1, Y_1), \cdots, (X_n, Y_n))$.

Supervised learning: The training data is then passed to a learning algorithm. The output of the learning algorithm is a regression model.



The training datas contain information about what the function ϕ may look like.

A learned regression model should reflect the structure of the training data
$\mathcal{D} := \{(X_1, Y_1), \cdots, (X_n, Y_n)\}$.

Learning vs. memorization

Goal: our goal is to learn a regression model $\phi : \mathcal{X} \to \mathbb{R}$ from the data $\mathcal{D}$.

Algorithm: The training data is then passed to a learning algorithm. The output of the learning algorithm is the regression model that we want.

Good regression models ϕ should map a feature $X \in \mathcal{X}$ to its associated label $Y \in \mathbb{R}$

Key point: The model should perform well on unseen feature vectors $X \in \mathcal{X}$, i.e., the feature vectors that are not in $\mathcal{D}$.

- Knowing ϕ performs well on $\mathcal{D} := \{(X_1, Y_1), \cdots, (X_n, Y_n)\}$ is not sufficient.

This is the crucial difference between learning and memorization.

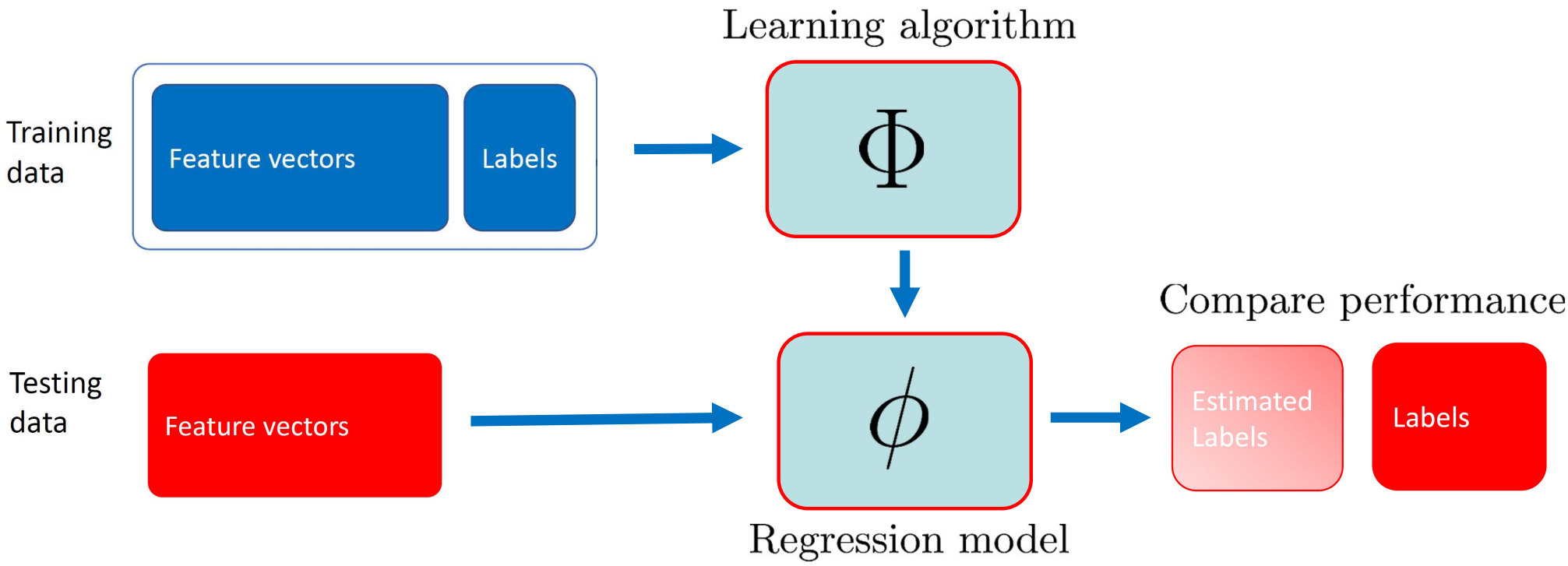
In order to investigate learning algorithms, we need test data apart from the training data.

We use the test dataset to assess our learning algorithms

Given a dataset, we can split it into a training dataset and a test dataset.

We use the training dataset to train our regression model.

We use the test dataset to assess our regression model (on unseen data).



Key point: Never use your test data to learn your regression model!

A probabilistic model and mean squared error

- quantifying the error of your regression model

A probabilistic model

In regression, we want to learn a function $\phi : \mathcal{X} \to \mathbb{R}$, which takes as input a feature vector $X \in \mathcal{X}$, and returns a continuous variable $\phi(X) \in \mathbb{R}$.

We begin with a feature space $\mathcal{X}$. In the simplest case this will be $\mathcal{X} = \mathbb{R}^d$.

We have $Y \in \mathbb{R}$.

Moreover, if we view a feature vector X as a random variable (vector) and Y as a random variable, then X and Y are dependent (the label is dependent on the feature vector).

Assume that $(X, Y) \sim \mathbf{P}$ with joint distribution $\mathbf{P}$.

- Then $\mathcal{D} := \{(X_1, Y_1), \cdots, (X_n, Y_n)\}$ is a sample drawn from $\mathbf{P}$.
- We often assume $(X_1, Y_1), \cdots, (X_n, Y_n)$ are i.i.d.

With the distribution $\mathbf{P}$, we can describe the error of the regression model (see next slide).

The mean squared error

We have random variables $(X, Y) \sim \mathbf{P}$ with joint distribution $\mathbf{P}$ on $\mathcal{X} \times \mathbb{R}$.

Our goal of the learn a regression model $\phi : \mathcal{X} \to \mathbb{R}$ such that $\phi(X) \approx Y$ for typical $(X, Y) \sim \mathbf{P}$.

We quantify our performance with the expected Mean Squared Error:

$$\mathcal{R}_{\text{MSE}}(\phi) := \mathbb{E}\big[(\phi(X) - Y)^2\big]$$

In regression contexts, the mean squared error is often simply referred to as the test error.

A good regression model $\phi : \mathcal{X} \to \mathbb{R}$ is one with a low expected test error.

The mean squared error

Our goal of the learn a regression model $\phi : \mathcal{X} \to \mathbb{R}$ such that $\phi(X) \approx Y$ for typical $(X, Y) \sim \mathbf{P}$.

We shall focus on the mean squared error $\mathcal{R}_{\text{MSE}}(\phi) := \mathbb{E}\big[(\phi(X) - Y)^2\big]$.

Other performance metrics are also used e.g. $\mathcal{R}_p(\phi) := \mathbb{E}[|\phi(X) - Y|^p]$

In practice we are also interested in computational issues:

- Time taken to train and test the model.
- Memory required to train and test the model.

The mean squared error on training data

Our goal of the learn a regression model $\phi : \mathcal{X} \to \mathbb{R}$ such that $\phi(X) \approx Y$ for typical $(X, Y) \sim \mathbf{P}$.

A good regression model $\phi : \mathcal{X} \to \mathbb{R}$ is one with a low expected test error

$$\mathcal{R}_{\text{MSE}}(\phi) := \mathbb{E}\left[(\phi(X) - Y)^2\right]$$

However, we can't observe the test error, because we do not know the underlying distribution $\mathbf{P}$

What we have is the training data (a sample drawn from $\mathbf{P}$).

We can compute the mean squared error on the training data $\mathcal{D} := \{(X_i, Y_i)\}$

$$\widehat{\mathcal{R}}_{\text{MSE}}(\phi) := \frac{1}{n} \sum_{i=1}^{n} (\phi(X_i) - Y_i)^2$$

The mean squared error on the training data is also known as the empirical mean squared error.

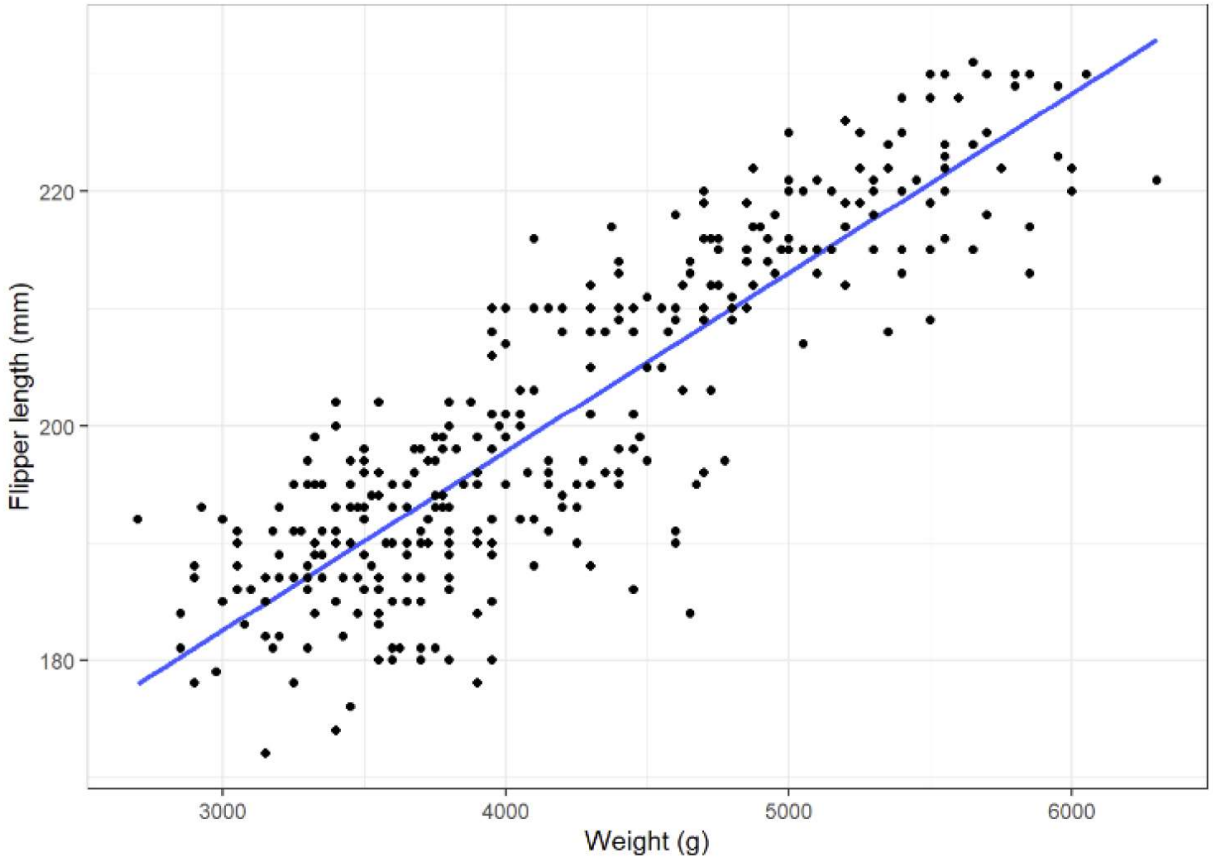
We can find a ϕ that has the smallest $\hat{R}_{\text{MSE}}(\phi)$ - this is a process of training.

In regression contexts $\widehat{R}_{\text{MSE}}(\phi)$ is referred to as the training error or the empirical error

Linear regression models

Linear regression models

Linear regression models fit a straight line to the data.



We want to find a linear model (represented by the blue line) to fit the data (represented by the black dots).

Linear regression models

Let's suppose that our feature vector $X \in \mathbb{R}^d$ has d continuous features $X = (X^1, \cdots, X^d) \in \mathcal{X} = \mathbb{R}^d$. For a given X we want to predict $Y \in \mathbb{R}$.

Example: Penguin regression

Predict Y the penguin's flipper length (mm) based on $X = (X^1, X^2, X^3) \in \mathbb{R}^3$ where

$$X^1 = \text{the weight of the penguin (grams)}$$

$$X^2 = \text{the bill length of the penguin (mm)}$$

$$X^3 = \text{the bill depth of the penguin (mm)}$$

Linear regression models

Let's suppose that our feature space $X \in \mathbb{R}^d$ has d continuous features $X = (X^1, \cdots, X^d) \in \mathcal{X} = \mathbb{R}^d$. For a given X we want to predict $Y \in \mathbb{R}$.

A linear regression model $\phi : \mathcal{X} \to \mathbb{R}$ is of the form

$$\phi(x) := w^0 + w^1 x^1 + \cdots w^d x^d$$

for $x := (x^1, x^2, \cdots, x^d) \in \mathbb{R}^d$.

Here ϕ has the following parameters: weights $w := (w^1, \cdots, w^d) \in \mathbb{R}^d$ and bias $w^0 \in \mathbb{R}$.

We can rewrite the model as $\phi(x) = wx^T + w^0$ where $(a^1, \cdots, a^d)(b^1, \cdots, b^d)^T = a^1 b^1 + \cdots + a^d b^d$.

To find a linear regression model, we need to decide the parameter w and w_0, from the data.

We can minimise the distance between $\phi(X)$ and Y.

Ordinary least squares method

- Minimising the training error

Ordinary least squares method

A good regression model $\phi : \mathcal{X} \to \mathbb{R}$ is one with a low expected test error

$$\mathcal{R}_{\text{MSE}}(\phi) := \mathbb{E}\left[(\phi(X) - Y)^2\right]$$

However, we can't observe the test error, because we do not know the underlying distribution $\mathbf{P}$

We can compute the mean squared error on the training data.

$$\widehat{\mathcal{R}}_{\text{MSE}}(\phi) := \frac{1}{n} \sum_{i=1}^{n} (\phi(X_i) - Y_i)^2$$

We can find a ϕ that has the smallest $\hat{R}_{\text{MSE}}(\phi)$ - this is a process of training.

The Ordinary Least Squares method (OLS) minimises $\widehat{R}_{\text{MSE}}$ over all possible linear models:

$$\phi_{w,w^0}(x) = wx^T + w^0$$

Next, we explore the closed form solution for w and w^0.

Minimising training error

We can compute the mean squared error on the training data.

$$\widehat{\mathcal{R}}_{\mathrm{MSE}}(\phi) := \tfrac{1}{n}\sum_{i=1}^{n}\left(\phi(X_i) - Y_i\right)^2$$

The Ordinary Least Squares method (OLS) minimises $\widehat{R}_{\mathrm{MSE}}$ over all possible linear models:

$$\phi_{w,w^0}(x) = wx^T + w^0$$

The ordinary least squares method (OLS) minimises:

$$\widehat{\mathcal{R}}_{\mathrm{MSE}}(\phi_{w,w_0}) := \frac{1}{n}\sum_{i=1}^{n}\left(\phi_{w,w^0}(X_i) - Y_i\right)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(wX_i^T + w^0 - Y_i\right)^2$$

This approach is known as empirical risk minimisation.

Observation: $\widehat{R}_{\mathrm{MSE}}$ is a quadratic function of w and w^0, hence we can find the closed-form solution for the minimiser. (see the next slide)

Minimising training error

$$\widehat{\mathcal{R}}_{\mathrm{MSE}}(\phi_{w,w_0}) := \frac{1}{n}\sum_{i=1}^{n}\left(\phi_{w,w^0}(X_i) - Y_i\right)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(wX_i^T + w^0 - Y_i\right)^2$$

To find the minimiser of $\hat{\mathcal{R}} := \hat{\mathcal{R}}_{\mathrm{MSE}}$, we take the derivatives and let them be zeros:

(1) $$0 = \frac{\partial}{\partial w}\widehat{\mathcal{R}} = \frac{2}{n}\sum_{i=1}^{n}\left(wX_i^TX_i + w^0X_i - Y_iX_i\right) = \frac{2}{n}\sum_{i=1}^{n}\left(wX_i^TX_i - Y_iX_i\right) + 2w^0\overline{X}$$

$\overline{X} := \frac{1}{n}\sum_i^n X_i$

(2) $$0 = \frac{\partial}{\partial w^0}\widehat{\mathcal{R}} = \frac{2}{n}\sum_{i=1}^{n}\left(wX_i^T + w^0 - Y_i\right) = \frac{2}{n}\sum_{i=1}^{n}\left(wX_i^T - Y_i\right) + 2w^0$$

(3) $$0 = \frac{\partial}{\partial w}\widehat{\mathcal{R}} - \left(\frac{\partial}{\partial w^0}\widehat{\mathcal{R}}\right)\overline{X} = \frac{2}{n}\sum_{i=1}^{n}\left[wX_i^T(X_i - \overline{X}) - Y_i(X_i - \overline{X})\right]$$ ← (combining (1) and (2))

(4) $$0 = \frac{2}{n}\sum_{i=1}^{n}\left[w(X_i - \overline{X})^T(X_i - \overline{X}) - (Y_i - \overline{Y})(X_i - \overline{X})\right]$$ ← (because $\sum_{i=1}^n a^T(X_i - \overline{X}) = 0$ for any vector a)

(5) $$\hat{w} = \Sigma_{Y,X}\Sigma_{X,X}^{-1}$$

$$\Sigma_{X,X} := \frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^T(X_i - \overline{X})$$

$$\Sigma_{Y,X} := \frac{1}{n}\sum_{i=1}^{n}(Y_i - \overline{Y})(X_i - \overline{X})$$

(6) $$\hat{w}^0 = \overline{Y} - \hat{w}\overline{X}^T$$ ← (because of (2) and (5))

Ordinary least squares method

$$\widehat{\mathcal{R}}_{\mathrm{MSE}}(\phi_{w,w_0}) := \frac{1}{n}\sum_{i=1}^{n}\left(\phi_{w,w^0}(X_i) - Y_i\right)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(wX_i^T + w^0 - Y_i\right)^2$$

Therefore,

The Ordinary Least Squares solution is:

$$\hat{\phi}_{\mathrm{OLS}}(x) = \hat{w}x^T + \hat{w}^0$$

where $\hat{w} = \Sigma_{Y,X}\Sigma_{X,X}^{-1}$ $\hat{w}^0 = \overline{Y} - \hat{w}\overline{X}^T$

$\overline{X} := \frac{1}{n}\sum_i^n X_i$ $\overline{Y} := \frac{1}{n}\sum_i^n Y_i$

$\Sigma_{X,X} := \frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^T(X_i - \overline{X})$ $\Sigma_{Y,X} := \frac{1}{n}\sum_{i=1}^{n}(Y_i - \overline{Y})(X_i - \overline{X})$

which is a minimiser of the training error $\hat{R}_{\mathrm{MSE}}$.

Example: Penguin regression

Example: Suppose we want to learn a classifier $\phi : \mathcal{X} \to \mathcal{Y}$ which takes a feature vector of morphological features and predicts whether a penguin belongs to either the Adelie species or the Chinstrap species.

Features: $X = (X^1, X^2, X^3) \in \mathcal{X} := \mathbb{R}^3$

X^1 = the weight of the penguin (grams)

X^2 = the bill length of the penguin (mm)

X^3 = the bill depth of the penguin (mm)

Labels: $Y \in \mathbb{R}$

Y = the flipper length of the penguin (mm)

Example: Penguin regression

Example: Suppose we want to learn a classifier $\phi : \mathcal{X} \to \mathcal{Y}$ which takes a feature vector of morphological features and predicts whether a penguin belongs to either the Adelie species or the Chinstrap species.

```
library(tidyverse)
library(palmerpenguins)

peng_flippers_total<-penguins%>%
  select(body_mass_g,bill_length_mm,bill_depth_mm,flipper_length_mm)%>%
  drop_na()
```

X^1 = the weight of the penguin (grams)

X^2 = the bill length of the penguin (mm)

X^3 = the bill depth of the penguin (mm)

Y = the flipper length of the penguin (mm)

Example: Penguin regression

Example: Suppose we want to learn a classifier $\phi : \mathcal{X} \to \mathcal{Y}$ which takes a feature vector of morphological features and predicts whether a penguin belongs to either the Adelie species or the Chinstrap species.

X Y

```
## # A tibble: 342 x 4
##    body_mass_g bill_length_mm bill_depth_mm flipper_length_mm
##          <int>          <dbl>         <dbl>             <int>
##  1        3750           39.1          18.7               181
##  2        3800           39.5          17.4               186
##  3        3250           40.3          18                 195
##  4        3450           36.7          19.3               193
##  5        3650           39.3          20.6               190
##  6        3625           38.9          17.8               181
##  7        4675           39.2          19.6               195
##  8        3475           34.1          18.1               193
##  9        4250           42            20.2               190
## 10        3300           37.8          17.1               186
## # ... with 332 more rows
```

feature vector $X = (X^1, X^2, X^3) \in \mathcal{X} := \mathbb{R}^3$

X^1 = the weight of the penguin (grams)

X^2 = the bill length of the penguin (mm)

X^3 = the bill depth of the penguin (mm)

labels $Y \in \mathbb{R}$

Y = the flipper length of the penguin (mm)

Example: Penguin regression

Let's perform a train test split of our data.

```
num_total<-peng_flippers_total%>%nrow() # total number of examples
num_train<-floor(num_total*0.75) # number of train examples
num_test<-num_total-num_train # number of test samples

set.seed(1) # set random seed for reproducibility
test_inds<-sample(seq(num_total),num_test) # random sample of test indicies
train_inds<-setdiff(seq(num_total),test_inds) # training data indicies

peng_flippers_train<-peng_flippers_total%>%filter(row_number() %in% train_inds) # train data
peng_flippers_test<-peng_flippers_total%>%filter(row_number() %in% test_inds) # test data
```

We extract feature vectors and labels.

```
peng_flippers_train_x<-peng_flippers_train%>%select(-flipper_length_mm) # train feature vectors
peng_flippers_train_y<-peng_flippers_train%>%pull(flipper_length_mm) # train labels

peng_flippers_test_x<-peng_flippers_test%>%select(-flipper_length_mm) # test feature vectors
peng_flippers_test_y<-peng_flippers_test%>%pull(flipper_length_mm) # test labels
```

Ordinary least squares method

$$\widehat{\mathcal{R}}_{\mathrm{MSE}}(\phi_{w,w_0}) := \frac{1}{n}\sum_{i=1}^{n}\left(\phi_{w,w^0}(X_i) - Y_i\right)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(wX_i^T + w^0 - Y_i\right)^2$$

Therefore,

The Ordinary Least Squares solution is:

$$\hat{\phi}_{\mathrm{OLS}}(x) = \hat{w}x^T + \hat{w}^0$$

where $\hat{w} = \Sigma_{Y,X}\Sigma_{X,X}^{-1}$ $\hat{w}^0 = \overline{Y} - \hat{w}\overline{X}^T$

$\overline{X} := \frac{1}{n}\sum_i^n X_i$ $\overline{Y} := \frac{1}{n}\sum_i^n Y_i$

$$\Sigma_{X,X} := \frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^T(X_i - \overline{X}) \qquad \Sigma_{Y,X} := \frac{1}{n}\sum_{i=1}^{n}(Y_i - \overline{Y})(X_i - \overline{X})$$

which is a minimiser of the training error $\hat{R}_{\mathrm{MSE}}$.

Ordinary least squares with R

The Ordinary Least Squares solution within R.

```
ols_predict_fn<-function(X,y){

  X_mn0<-scale(X,scale=FALSE) # subtract means of features
  y_mn0<-scale(y,scale=FALSE) # subtract means of labels

  Sig_XX<-t(X_mn0)%*%X_mn0 # compute features covariance
  Sig_YX<-t(y_mn0)%*%X_mn0 # compute label feature covariance

  weights<-Sig_YX%*%solve(Sig_XX) # compute weights
  intercept<-mean(y)-colMeans(X)%*%t(weights) # compute intercept

  predict_fn<-function(x){
  return((x%*%t(weights)+intercept[1])%>%as.numeric())}

  return(predict_fn) # extract prediction function

}
```

$X_i - \overline{X}$

$Y_i - \overline{Y}$

$\hat{\phi}_{\mathrm{OLS}}(x) = \hat{w}x^T + \hat{w}^0$

Example: Penguin regression

1. apply our OLS function to the penguin data.

```
ols_reg_model<-ols_predict_fn(peng_flippers_train_x%>%as.matrix(),peng_flippers_train_y)
```

2. compute the training error:

```
ols_train_predicted_y<-ols_reg_model(peng_flippers_train_x%>%as.matrix()) # extract predictions
ols_train_error<-mean((ols_train_predicted_y-peng_flippers_train_y)^2) # compute train error
ols_train_error
```

```
## [1] 31.86219
```

3. estimate the test error:

```
ols_test_predicted_y<-ols_reg_model(peng_flippers_test_x%>%as.matrix()) # extract predictions
ols_test_error<-mean((ols_test_predicted_y-peng_flippers_test_y)^2) # compute train error
ols_test_error
```

```
## [1] 37.79135
```

Example: Penguin regression

Alternatively, we can also use R's inbuilt linear model function:

```
ols_model<-lm(flipper_length_mm~.,peng_flippers_train) # train OLS model
ols_model
```

```
##
## Call:
## lm(formula = flipper_length_mm ~ ., data = peng_flippers_train)
##
## Coefficients:
##     (Intercept)     body_mass_g  bill_length_mm   bill_depth_mm
##       157.38535         0.01134         0.54465        -1.62205
```

Example: Penguin regression

Compute the training error with the OLS model:

```
ols_train_predicted_y<-predict(ols_model,peng_flippers_train_x) # extract predictions
ols_train_error<-mean((ols_train_predicted_y-peng_flippers_train_y)^2) # compute train error
ols_train_error
```

```
## [1] 31.86219
```

Compute the test error with the OLS model:

```
ols_test_predicted_y<-predict(ols_model,peng_flippers_test_x) # extract predictions
ols_test_error<-mean((ols_test_predicted_y-peng_flippers_test_y)^2) # compute train error
ols_test_error
```

```
## [1] 37.79135
```

A probabilistic perspective on OLS regression

A probabilistic perspective on OLS regression

Suppose we have training data consisting of a set of labelled data $\mathcal{D} := ((X_1, Y_1), \cdots, (X_n, Y_n))$.

We saw that the Ordinary Least Squares solution minimises the mean squared error on training data:

$$\widehat{\mathcal{R}}_{\text{MSE}}(\phi_{w,w_0}) := \frac{1}{n}\sum_{i=1}^{n}\left(\phi_{w,w^0}(X_i) - Y_i\right)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(wX_i^T + w^0 - Y_i\right)^2$$

However, we can also motivate this solution via a probabilistic model:

Suppose that for some $w = (w^1, \cdots, w^d) \in \mathbb{R}^d$ and $w^0 \in \mathbb{R}$,

$$Y_i = wX_i^T + w^0 + \epsilon_i \qquad \text{with} \qquad \epsilon_i \sim \mathcal{N}(0, \sigma^2) \text{ (i.i.d.)}$$

To find w, w^0 from our data, we can use the maximum likelihood principle (see the next slide).

Maximum likelihood

Suppose we have training data consisting of a set of labelled data $\mathcal{D} := ((X_1, Y_1), \cdots, (X_n, Y_n))$.

Suppose that for some $w = (w^1, \cdots, w^d) \in \mathbb{R}^d$ and $w^0 \in \mathbb{R}$,

$$Y_i = wX_i^T + w^0 + \epsilon_i \qquad \text{with} \qquad \epsilon_i \sim \mathcal{N}(0, \sigma^2) \text{ (i.i.d.)}$$

Likelihood function

because $Y_i - wX_i^T - w_0 = \epsilon_i \sim \mathcal{N}(0, \sigma^2)$

$$l(w, w^0) = \prod_{i=1}^{n} \left(\frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{\left(wX_i^T + w_0 - Y_i\right)^2}{2\sigma^2} \right) \right)$$

Log likelihood function

$$\log l(w, w^0) = -n \log(\sigma\sqrt{2\pi}) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(wX_i^T + w^0 - Y_i \right)^2$$

Therefore, maximising $\log l(w, w^0)$ is equivalent to minimising

$\frac{1}{n} \sum_{i=1}^{n} (wX_i^T + w^0 - Y_i)^2$ which is the same as $\hat{\mathcal{R}}_{\text{MSE}}$

A probabilistic perspective on OLS regression

Suppose we have training data consisting of a set of labelled data $\mathcal{D} := ((X_1, Y_1), \cdots, (X_n, Y_n))$.

We saw that the Ordinary Least Squares solution minimises the mean squared error on training data:

$$\widehat{\mathcal{R}}_{\text{MSE}}(\phi_{w,w_0}) := \frac{1}{n}\sum_{i=1}^{n}\left(\phi_{w,w^0}(X_i) - Y_i\right)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(wX_i^T + w^0 - Y_i\right)^2$$

Conclusion: Minimising the squared training error is the same as maximising the likelihood

$$Y_i = wX_i^T + w^0 + \epsilon_i \qquad \text{with} \qquad \epsilon_i \sim \mathcal{N}(0, \sigma^2) \text{ (i.i.d.)}$$

for parameters $w := (w^1, \cdots, w^d)$ and $w^0 \in \mathbb{R}$.

A probabilistic perspective on OLS regression

Conclusion: Minimising the squared training error is the same as maximising the likelihood

$$Y_i = wX_i^T + w^0 + \epsilon_i \qquad \text{with} \qquad \epsilon_i \sim \mathcal{N}(0, \sigma^2) \text{ (i.i.d.)}$$

for parameters $w := (w^1, \cdots, w^d)$ and $w^0 \in \mathbb{R}$.

The OLS estimators $\hat{w} = \Sigma_{Y,X}\Sigma_{X,X}^{-1}$ $\hat{w}^0 = \overline{Y} - \hat{w}\overline{X}^T$ are

1. Maximum likelihood estimators of w and w^0

2. Unbiased estimators with $\mathbb{E}(\hat{w})$ and $\mathbb{E}(\hat{w}^0)$

3. Minimum variance over all unbiased estimators (known as the Gauss Markov theorem).

A probabilistic perspective on OLS regression

The OLS estimators $\hat{w} = \Sigma_{Y,X}\Sigma_{X,X}^{-1}$ $\hat{w}^0 = \overline{Y} - \hat{w}\overline{X}^T$ are

1. Maximum likelihood estimators of w and w^0
2. Unbiased estimators with $\mathbb{E}(\hat{w})$ and $\mathbb{E}(\hat{w}^0)$
3. Minimum variance over all unbiased estimators (known as the Gauss Markov theorem).

4. However, the variance can still be extremely large, especially when the dimension is large compared to the sample size.

Ordinary least squares can still perform very poorly in high dimensional settings.

What have we covered?

We began by introducing the concept of regression with some examples.

We discussed the mean squared error on the test data as a performance metric.

We then considered the topic of linear regression.

We looked at Ordinary Least Squares (OLS) as an algorithm for linear regression.

We saw the advantages of OLS as unbiased, maximum likelihood estimators under a natural model.

We also discussed how OLS can suffer from very large variance, especially in high dimensions!

University of BRISTOL

Thanks for listening!

Dr. Rihuan Ke

rihuan.ke@bristol.ac.uk

Statistical Computing and Empirical Methods
Unit EMATM0061, MSc Data Science