University of BRISTOL

An introduction to regression

Learning functions which map feature vectors to continuous variables

Henry W J Reeve

henry.reeve@bristol.ac.uk

Statistical Computing & Empirical Methods (EMATM0061)

MSc in Data Science, Teaching block 1, 2021.

What will we 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 classification?

Learning a function $\phi : \mathcal{X} \to \mathcal{Y}$

which takes as input a feature vector $X \in \mathcal{X}$

and returns a categorical variable $\phi(X) \in \mathcal{Y}$

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

Features | Classification rule | Categorical output

What is regression?

Learning a function $\phi : \mathcal{X} \to \mathbb{R}$

which takes as input a feature vector $X \in \mathcal{X}$

and returns a continuous variable $\phi(X) \in \mathbb{R}$

$$X \in \mathcal{X} \longrightarrow \boxed{\phi} \longrightarrow \phi(X) \in \mathbb{R}$$

Features — Regression model — Continuous output

What is regression?

Example 1: Fuel usage

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

$$X \in \mathcal{X} \longrightarrow \boxed{\phi} \longrightarrow \phi(X) \in \mathbb{R}$$

Features — Regression model — Continuous output

What is regression?

Example 2: Weight of fish

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

$$X \in \mathcal{X} \longrightarrow \boxed{\phi} \longrightarrow \phi(X) \in \mathbb{R}$$

Features | Regression model | Continuous output

What is regression?

Example 3: Temperature prediction

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

$$X \in \mathcal{X} \longrightarrow \boxed{\phi} \longrightarrow \phi(X) \in \mathbb{R}$$

Features | Regression model | Continuous output

What is regression?

Learning a function $\phi : \mathcal{X} \rightarrow \mathbb{R}$ known as a **regression model**.

which takes as input a feature vector $X \in \mathcal{X}$

and returns a continuous variable $\phi(X) \in \mathbb{R}$ also known as a label.

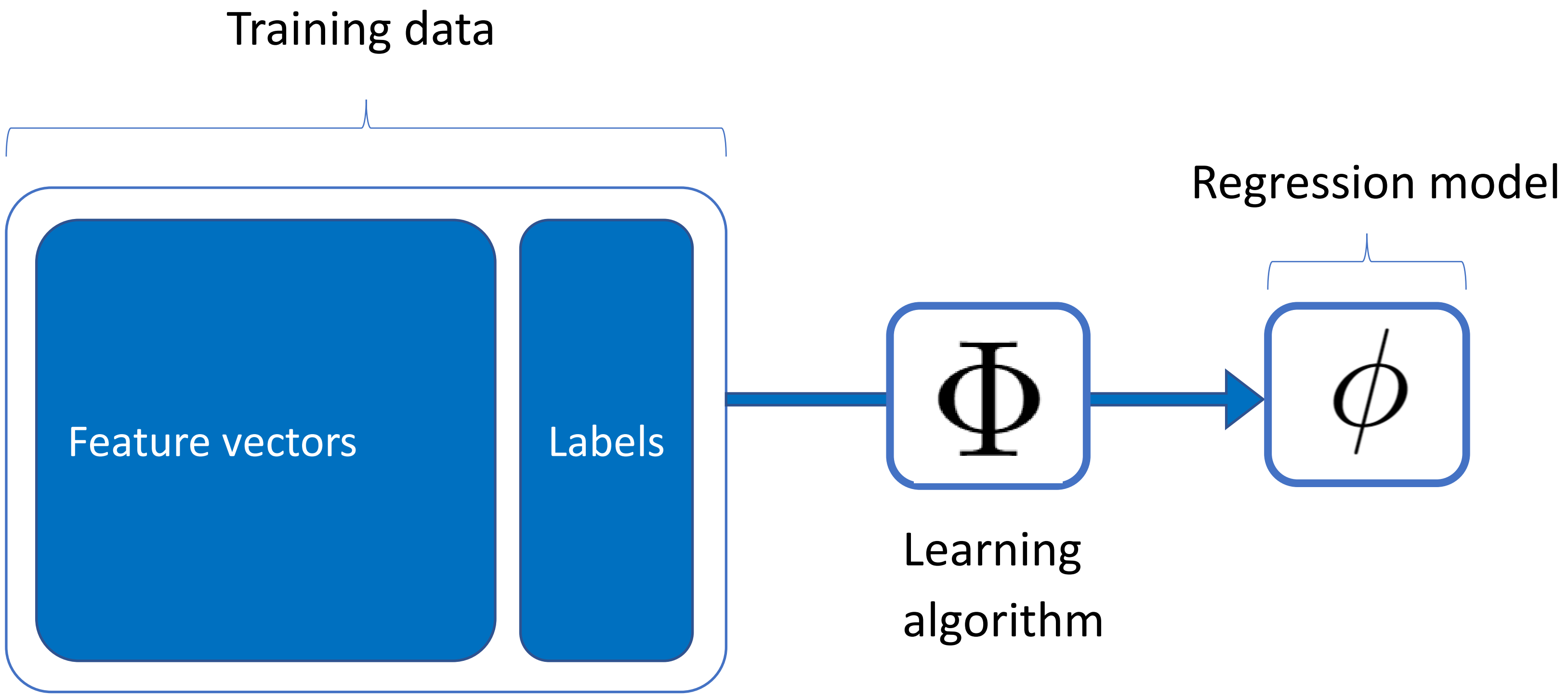
$$X \in \mathcal{X} \longrightarrow \boxed{\phi} \longrightarrow \phi(X) \in \mathbb{R}$$

Features | Regression model | Continuous output

Supervised learning

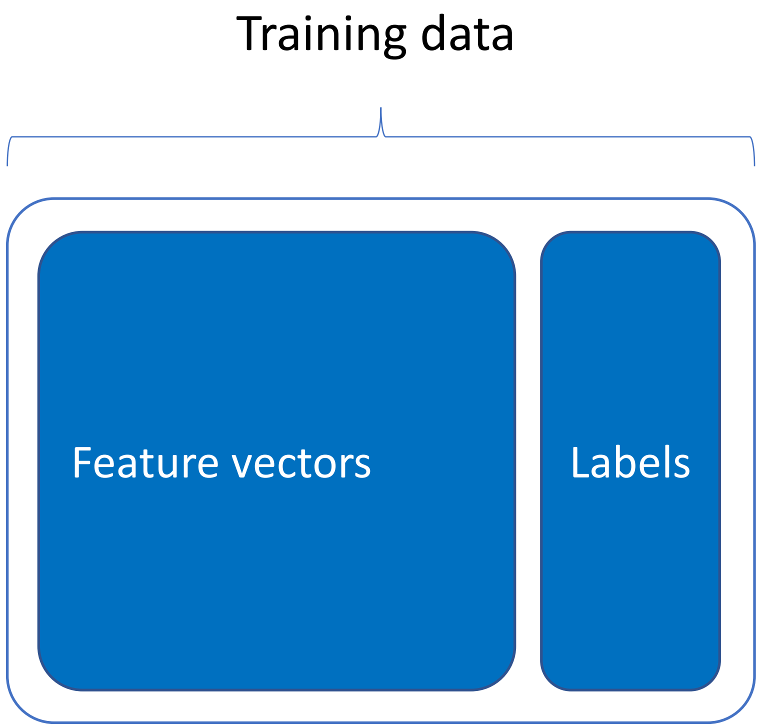
Supervised learning is the process of learning a function based on training data with feature vectors and labels.

We learn a **regression model** based on a set of training data $\mathcal{D}$.



Supervised learning

We learn a set of **regression model** based on a set of training data $\mathcal{D}$.



Training data consists of a set of labelled data

$$\mathcal{D} = ((X_1, Y_1), \cdots, (X_n, Y_n))$$

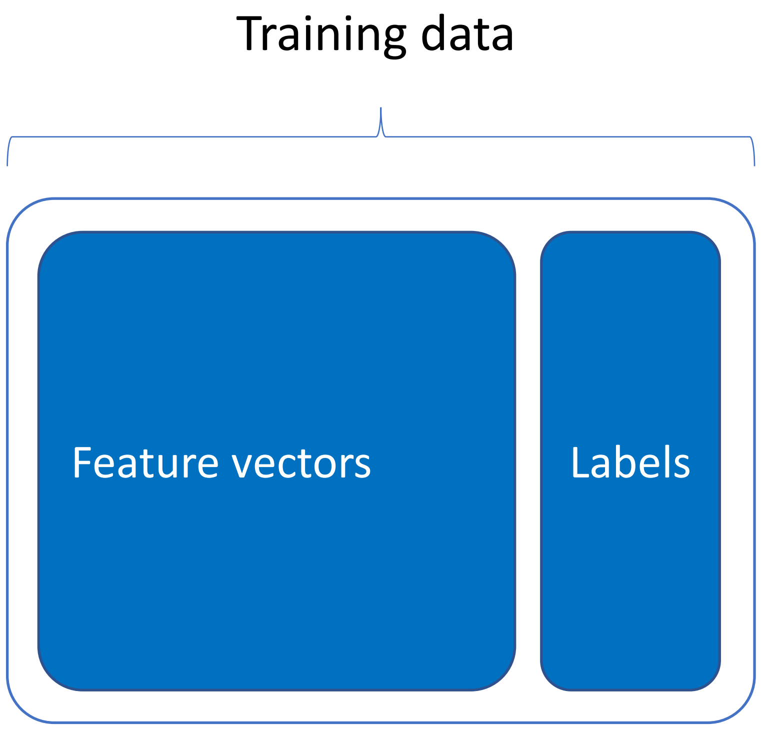
A sequence of ordered pairs (X_i, Y_i) .

X_i is a feature vector.

Y_i is a numerical label associated with X_i .

Supervised learning

We learn a set of **regression model** based on a set of training data $\mathcal{D}$.



Training data consists of a set of labelled data

$$\mathcal{D} = ((X_1, Y_1), \cdots, (X_n, Y_n))$$

<u>Example</u>

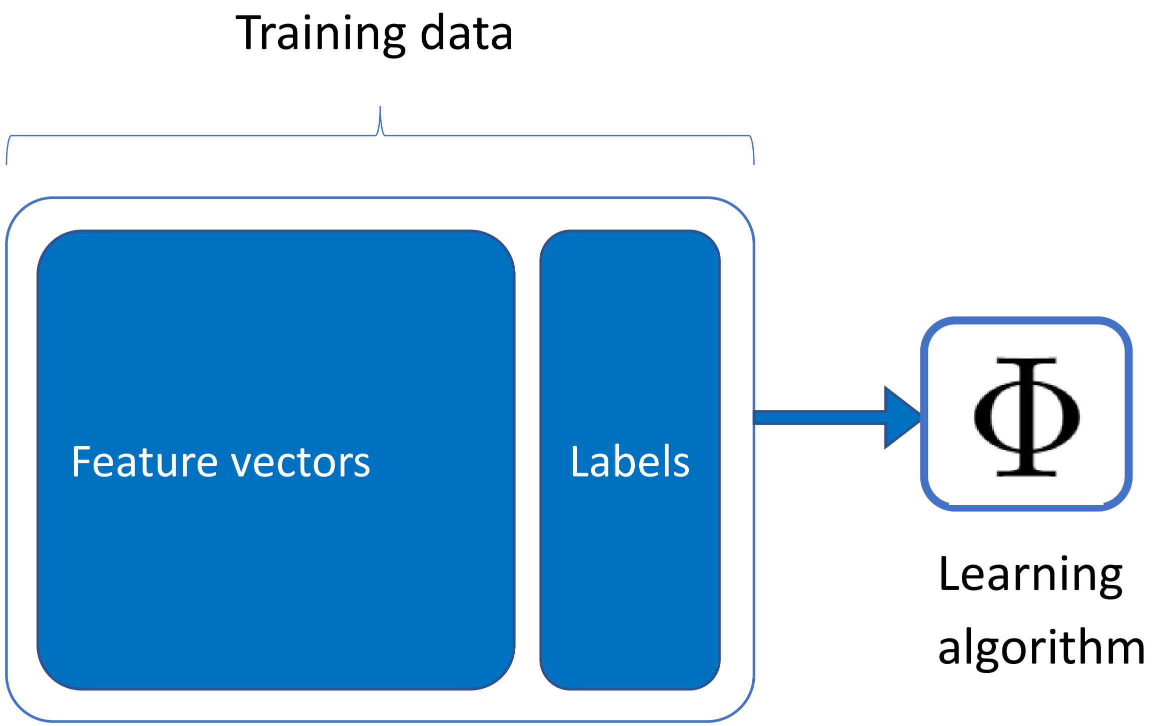
X_i is an image of a particular fish.

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

Supervised learning

We learn a **regression model** based on a set of training data $\mathcal{D}$.

Training data is passed to a **learning algorithm.**



The learning algorithm

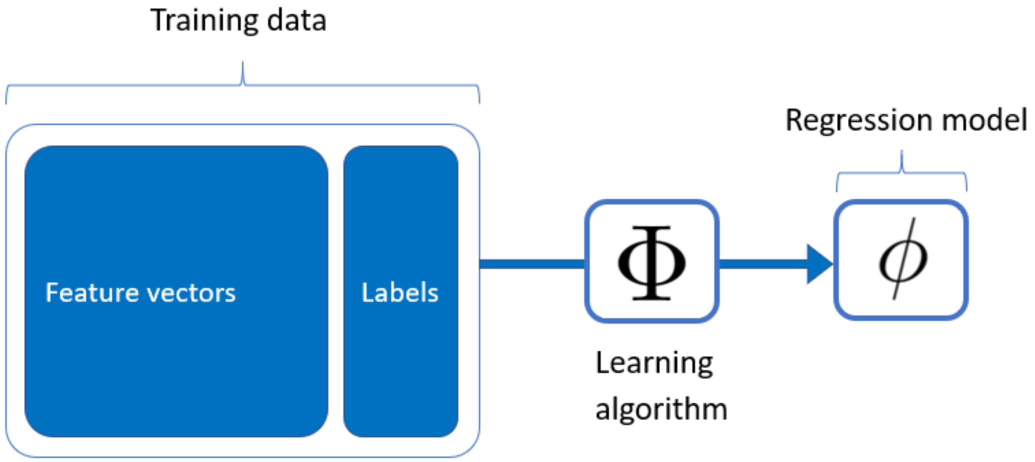
automatically identifies patterns

within the training data $\mathcal{D}$.

Supervised learning

We learn a set of **regression model** based on a set of training data $\mathcal{D}$.

Training data is passed to a **learning algorithm** which outputs a regression model.



The regression model rule is a mapping:

$$\phi : X \mapsto Y$$

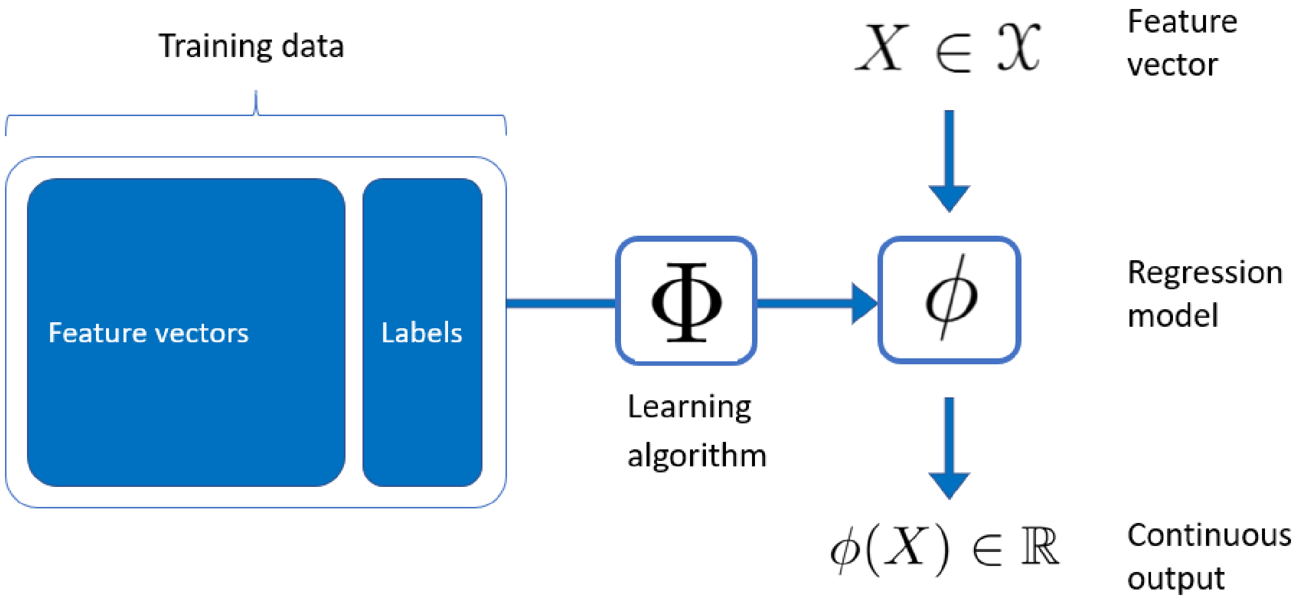
Should reflect the structure of the training data:

$$\mathcal{D} = ((X_1, Y_1), \cdots, (X_n, Y_n))$$

Learning vs. memorization

We learn a set of **regression model** based on a set of training data $\mathcal{D}$.

Training data is passed to a **learning algorithm** which outputs a regression model.

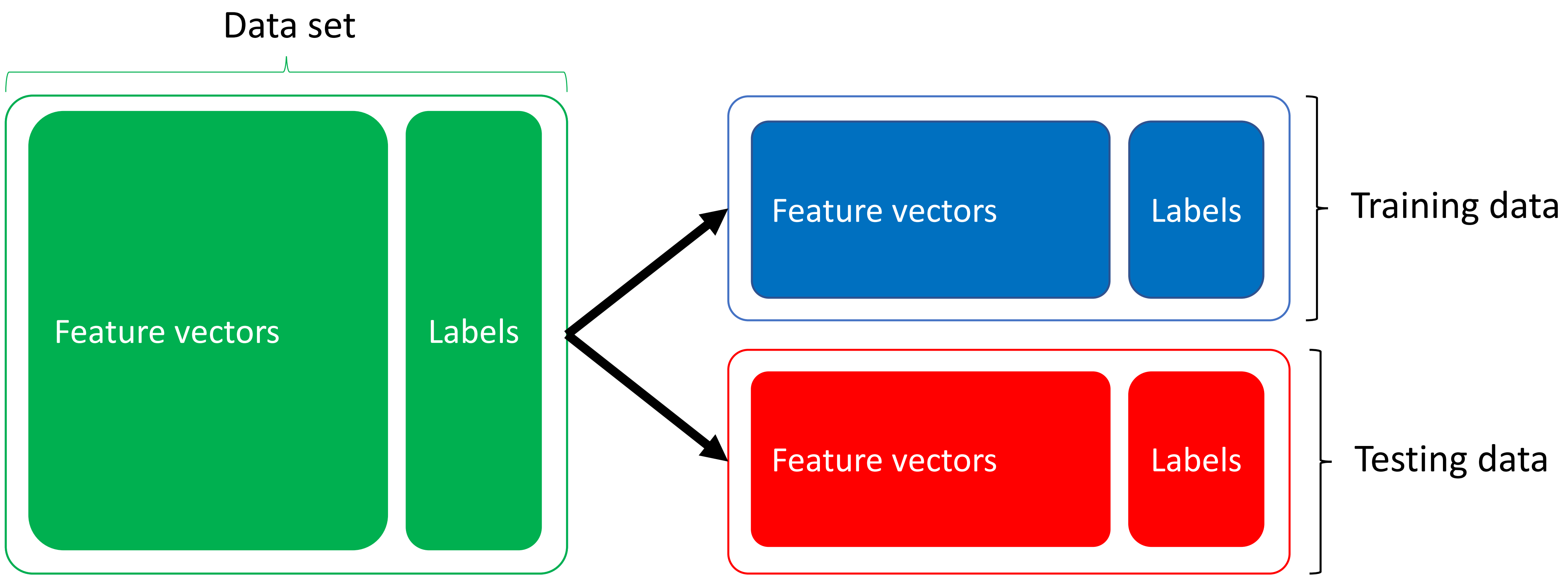


For the regression model to be successful it must perform well on **unseen** data.

The train test split

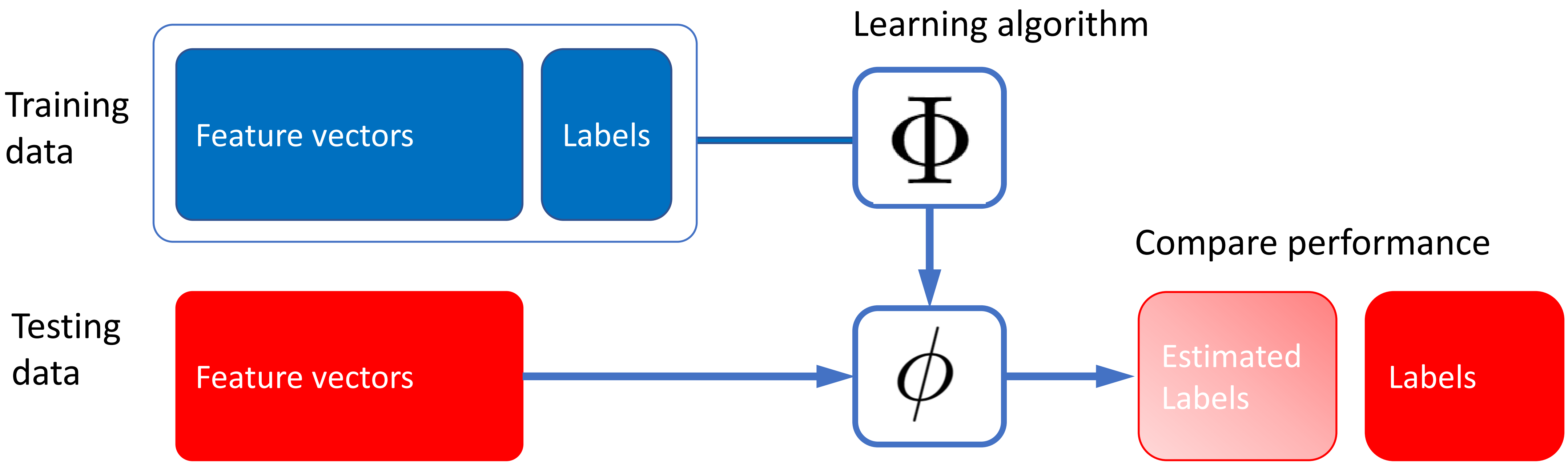
For the regression model to be successful it must perform well on **unseen** data.

In order to investigate learning algorithms, we always need to do a test train split.



The train test split

The goal of the test data is to see how well the model does on unseen data.



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

Now take a break!

A probabilistic model for regression

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

We then have random variables (X, Y).

X is a feature vector which takes values in $\mathcal{X}$.

Y is a label which takes values in $\mathbb{R}$.

The random variables $(X, Y) \sim \mathrm{P}$ have joint distribution P.

A probabilistic model for regression

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

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

In addition, we have some training data $\mathcal{D} = ((X_1, Y_1), \cdots, (X_n, Y_n))$.

We assume that examples $(X_i, Y_i) \sim \mathrm{P}$ are independent and identically distributed.

This will let us learn properties about the underlying distribution of $(X, Y) \sim \mathrm{P}$.

The mean squared error on the test data

We have random variables $(X, Y) \sim \mathrm{P}$ with joint distribution 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 $(X, Y) \sim \mathrm{P}$.

We quantify our performance with the **expected Mean Squared Error on test data**

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

In regression contexts, the mean squared error on the test 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 on the test data

We shall focus on the **mean squared error on the test data.**

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

Other performance metrics are also used e.g.

$$\mathcal{R}_p(\phi) := \mathbb{E}\left[|\phi(X) - Y|^p\right]$$

In practice we are also interested in computational issues:

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

The mean squared error on the training data

Our goal is to minimize the **mean squared error on the test data.**

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

We can't observe the test error – we do not know the underlying distribution P.

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

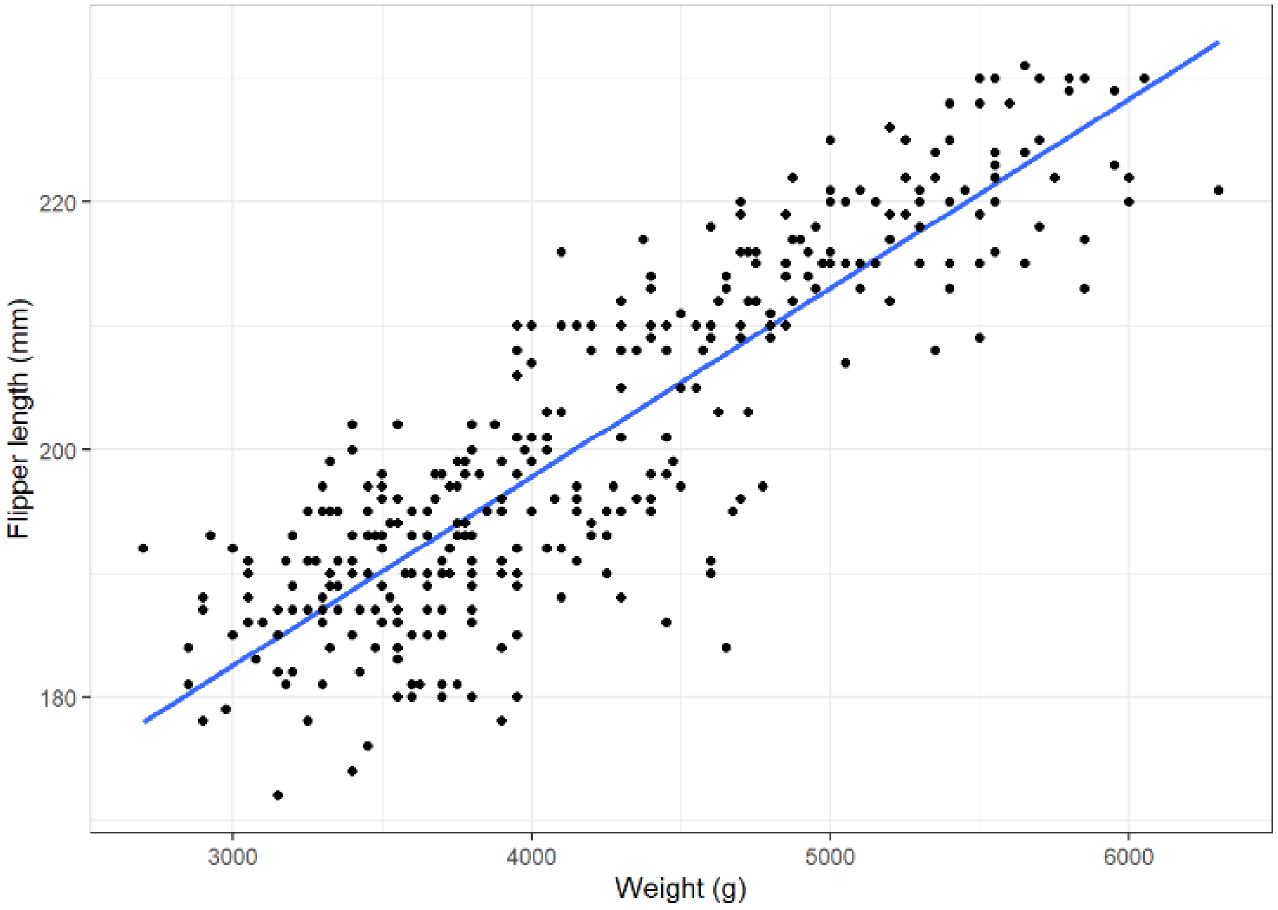
$$\hat{\mathcal{R}}_{\mathrm{MSE}}(\phi) := \frac{1}{n}\sum_{i=1}^{n}(\phi(X_i) - Y_i)^2$$

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

In regression contexts $\hat{\mathcal{R}}_{\mathrm{MSE}}(\phi)$ is referred to as the **training error** or the **empirical error**.

Linear regression models

Linear regression models fit a straight line to the data.



Linear regression models

Let's suppose we want to learn a linear regression model $\phi : \mathcal{X} \to \mathbb{R}$.

Let's suppose that our feature space $\mathcal{X} = \mathbb{R}^d$ has d continuous features,

$$X = (X^1, \cdots, X^d) \in \mathcal{X} = \mathbb{R}^d$$

<u>Example</u>: Penguin regression

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

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

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

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

Linear regression models

Suppose we have d continuous features $X = (X^1, \cdots, X^d) \in \mathcal{X} = \mathbb{R}^d$

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

$$\phi(x) = w^0 + w^1 \cdot x^1 + w^2 \cdot x^2 + \cdots + w^d \cdot x^d$$

For all $(x^1, x^2, \cdots, x^d) \in \mathbb{R}^d.$

Weights $w = (w^1, \cdots, w^d) \in \mathbb{R}^d$

Bias $w^0 \in \mathbb{R}$

Linear regression models

Suppose we have d continuous features $X = (X^1, \cdots, X^d) \in \mathcal{X} = \mathbb{R}^d$

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

$$\phi(x) = w^0 + w^1 \cdot x^1 + w^2 \cdot x^2 + \cdots + w^d \cdot x^d$$

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

with weights $w = (w^1, \cdots, w^d) \in \mathbb{R}^d$ and a bias $w^0 \in \mathbb{R}$

We can rewrite this as $\phi(x) = w\ x^\top + w^0$

where $(a^1, \cdots, a^d)\ (b^1, \cdots, b_d)^\top = a^1 \cdot b^1 + \cdots + a^d b^d.$

Now take a break!

Ordinary least squares

Our goal is to minimize the **mean squared error on the test data.**

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

Whilst we can't observe $\mathcal{R}_{\text{MSE}}(\phi)$ we can compute the **mean squared error on the training data**

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

The **Ordinary Least Squares** method (OLS) minimizes the training error over all possible linear models

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

Ordinary least squares

The ordinary least squares method (OLS) minimizes:

$$\hat{\mathcal{R}}_{\mathrm{MSE}}\left(\phi_{w,w^0}\right) = \frac{1}{n}\sum_{i=1}^{n}\left(\phi_{w,w^0}(X_i) - Y_i\right)^2$$
$$= \frac{1}{n}\sum_{i=1}^{n}\left(w\, X_i^\top + w^0 - Y_i\right)^2 .$$

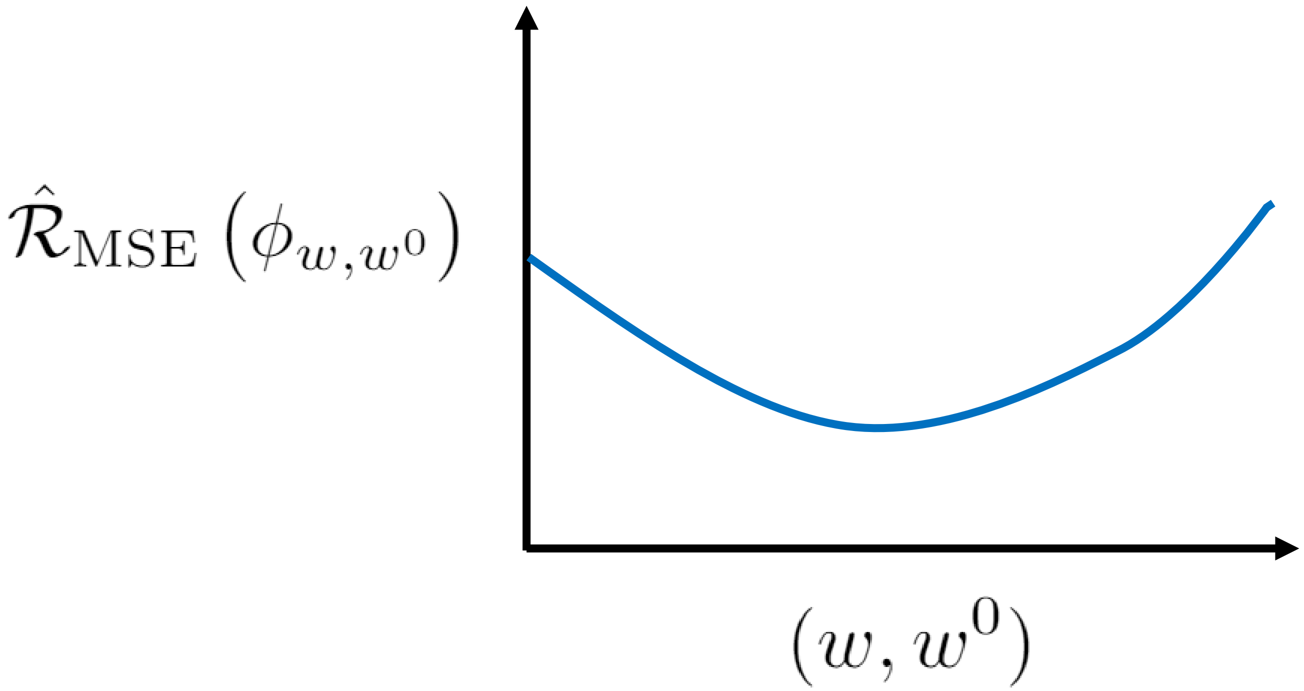
over all possible parameters $w = (w^1, \cdots, w^d) \in \mathbb{R}^d$ and $w^0 \in \mathbb{R}$.

This approach is known as empirical risk minimization.

Ordinary least squares

OLS objective:

Minimise
$$\hat{\mathcal{R}}_{\mathrm{MSE}}\left(\phi_{w,w^0}\right) = \frac{1}{n}\sum_{i=1}^{n}\left(w\, X_i^{\top} + w^0 - Y_i\right)^2 .$$



Ordinary least squares

Minimise
$$\hat{\mathcal{R}}_{\text{MSE}}\left(\phi_{w,w^0}\right) = \frac{1}{n}\sum_{i=1}^{n}\left(w\ X_i^\top + w^0 - Y_i\right)^2 .$$

First let
$$\overline{X} := \frac{1}{n}\sum_{i=1}^{n} X_i \quad \text{and} \quad \overline{Y} := \frac{1}{n}\sum_{i=1}^{n} Y_i$$

$$\frac{\partial}{\partial w^0}\left\{\hat{\mathcal{R}}_{\text{MSE}}\left(\phi_{w,w^0}\right)\right\} = \frac{2}{n}\sum_{i=1}^{n}\left(w\ X_i^\top + w^0 - Y_i\right) = 2\left(w\ \overline{X}^\top + w^0 - \overline{Y}\right).$$

At the minimum
$$\frac{\partial}{\partial w^0}\left\{\hat{\mathcal{R}}_{\text{MSE}}\left(\phi_{w,w^0}\right)\right\} = 0 \quad \Rightarrow \quad w^0 = \overline{Y} - w\ \overline{X}^\top .$$

Ordinary least squares

Minimise $\hat{\mathcal{R}}_{\text{MSE}}\left(\phi_{w,w^0}\right) = \frac{1}{n}\sum_{i=1}^{n}\left(w\, X_i^\top + w^0 - Y_i\right)^2$

At the minimum over (w, w^0) we have $w^0 = \overline{Y} - w\,\overline{X}^\top.$

It suffices to minimise

$$\begin{aligned}
\hat{\mathcal{R}}_{\text{MSE}}\left(\phi_{w,w^0}\right) &= \frac{1}{n}\sum_{i=1}^{n}\left(w\, X_i^\top + w^0 - Y_i\right)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}\left\{w\, X_i^\top + \left(\overline{Y} - w\,\overline{X}^\top\right) - Y_i\right\}^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}\left\{w\,\left(X_i - \overline{X}\right)^\top - \left(Y_i - \overline{Y}\right)\right\}^2.
\end{aligned}$$

Ordinary least squares

To minimize $\hat{\mathcal{R}}_{\mathrm{MSE}}\left(\phi_{w,w^0}\right)$ it suffices to minimize $\frac{1}{n}\sum_{i=1}^{n}\left\{w\ \left(X_i-\overline{X}\right)^\top-\left(Y_i-\overline{Y}\right)\right\}^2.$

Define $\Sigma_{X,X} := \frac{1}{n}\sum_{i=1}^{n}\left(X_i-\overline{X}\right)^\top\left(X_i-\overline{X}\right)$ and $\Sigma_{Y,X} := \frac{1}{n}\sum_{i=1}^{n}\left(Y_i-\overline{Y}\right)\left(X_i-\overline{X}\right)$.

We then compute

$$\begin{aligned}
&\frac{\partial}{\partial w}\left(\frac{1}{n}\sum_{i=1}^{n}\left\{w\ \left(X_i-\overline{X}\right)^\top-\left(Y_i-\overline{Y}\right)\right\}^2\right)\\
&=\frac{2}{n}\sum_{i=1}^{n}\left\{w\ \left(X_i-\overline{X}\right)^\top-\left(Y_i-\overline{Y}\right)\right\}\left(X_i-\overline{X}\right)\\
&=2\left(w\ \Sigma_{X,X}-\Sigma_{Y,X}\right).
\end{aligned}$$

Ordinary least squares

To minimize $\hat{\mathcal{R}}_{\mathrm{MSE}}\left(\phi_{w,w^0}\right)$ it suffices to minimize $\frac{1}{n}\sum_{i=1}^{n}\left\{w\ \left(X_i-\overline{X}\right)^{\top}-\left(Y_i-\overline{Y}\right)\right\}^2.$

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

At the minimum we have

$$0=\frac{\partial}{\partial w}\left(\frac{1}{n}\sum_{i=1}^{n}\left\{w\ \left(X_i^{\top}-\overline{X}\right)^{\top}-\left(Y_i-\overline{Y}\right)\right\}^2\right)=2\left(w\,\Sigma_{X,X}-\Sigma_{Y,X}\right).$$

The ordinary least squares estimate $\hat{w}$ satisfies $\hat{w}=\Sigma_{Y,X}\ \left(\Sigma_{X,X}\right)^{-1}$

Ordinary least squares

Minimise $$\hat{\mathcal{R}}_{\mathrm{MSE}}\left(\phi_{w,w^0}\right) = \frac{1}{n}\sum_{i=1}^{n}\left(w\, X_i^\top + w^0 - Y_i\right)^2 .$$

We define $$\overline{X} := \frac{1}{n}\sum_{i=1}^{n} X_i \quad , \qquad \Sigma_{X,X} := \frac{1}{n}\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^\top \left(X_i - \overline{X}\right)$$

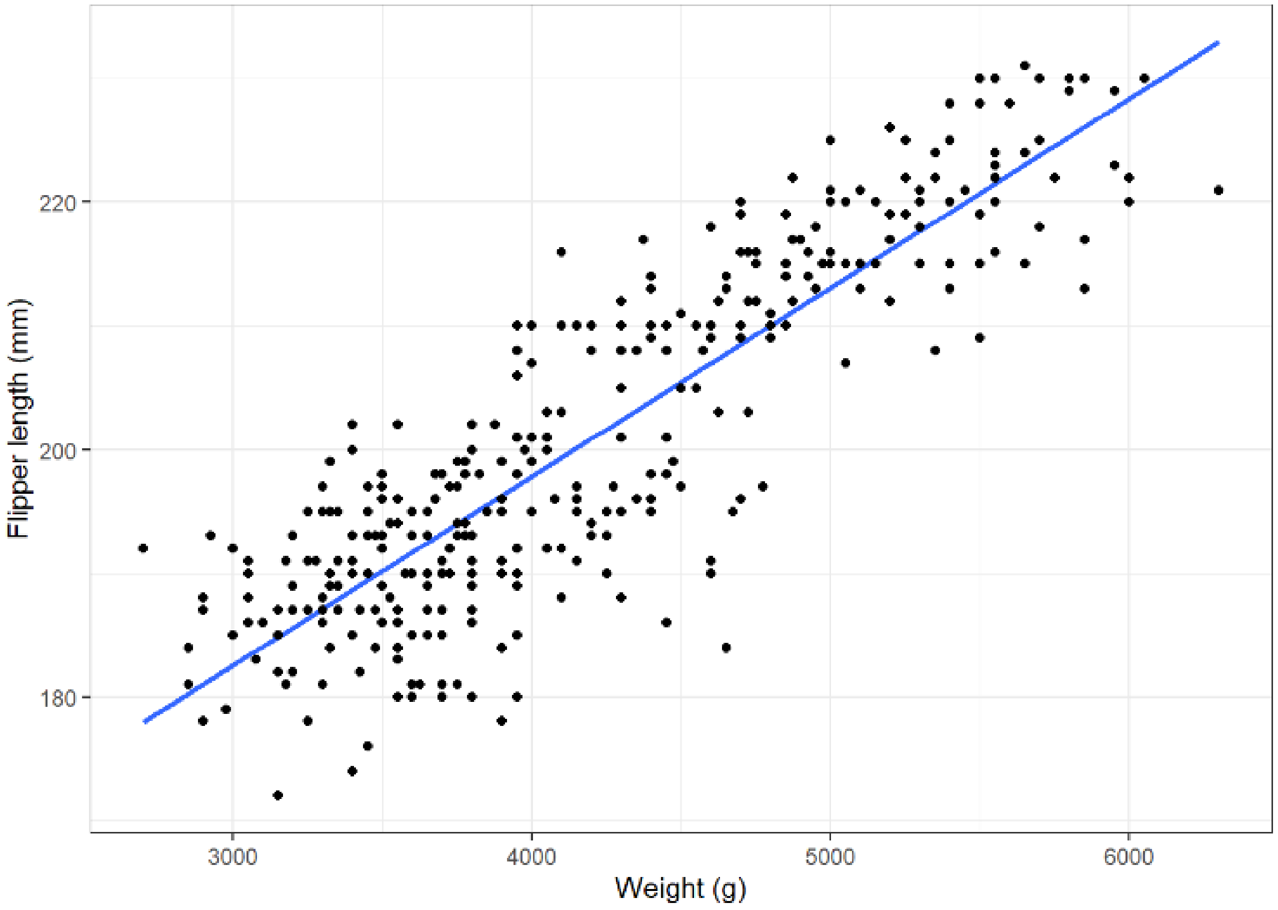
$$\overline{Y} := \frac{1}{n}\sum_{i=1}^{n} Y_i \quad \text{and} \qquad \Sigma_{Y,X} := \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \overline{Y}\right)\left(X_i - \overline{X}\right)$$

The Ordinary Least Squares solution is: $$\hat{\phi}_{\mathrm{OLS}}(x) = \hat{w}\; x^\top + \hat{w}^0$$

$$\hat{w} = \Sigma_{Y,X}\;\left(\Sigma_{X,X}\right)^{-1} \qquad \text{with} \qquad \hat{w}^0 = \overline{Y} - \hat{w}\;\overline{X}^\top .$$

Linear regression models

An ordinary least squares solution:



Now take a break!

Example: Penguin regression

Suppose we want to learn a regression model $\phi : \mathcal{X} \to \mathbb{R}$ which estimates the penguin's flipper length based on a feature vector of other morphological features.

Features:

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

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

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

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

Labels:

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

$$Y = \text{ the flipper length of the penguin (mm)}$$

Example: Penguin regression

Suppose we want to learn a regression model $\phi : \mathcal{X} \to \mathbb{R}$ which estimates the penguin's flipper length based on a feature vector of other morphological features.

```
library(tidyverse)
library(palmerpenguins)

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

Example: Penguin regression

Suppose we want to learn a regression model $\phi : \mathcal{X} \to \mathbb{R}$ which estimates the penguin's flipper length based on a feature vector of other morphological features.

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

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

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

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

Label $Y \in \mathbb{R}$

$Y =$ 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
```

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

The Ordinary Least Squares solution is: $$\hat{\phi}_{\mathrm{OLS}}(x) = \hat{w}\ x^{\top} + \hat{w}^{0}$$

where

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

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

$$\hat{w} = \Sigma_{Y,X}\ \left(\Sigma_{X,X}\right)^{-1} \qquad \hat{w}^{0} = \overline{Y} - \hat{w}\ \overline{X}^{\top}.$$

Ordinary least squares

The Ordinary Least Squares solution within R.

```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

}
```

Example: Penguin regression

Let’s apply our OLS function to the penguin data.

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

We can estimate the training error as follows.

```
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
```

We can also estimate the test error as follows.

```
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

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

We can estimate the training error as follows.

```
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
```

We can also estimate the test error as follows.

```
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
```

Now take a break!

A probabilistic perspective on OLS regression

Suppose we have training data $\mathcal{D} = ((X_1, Y_1), \cdots, (X_n, Y_n))$.

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

$$\hat{\mathcal{R}}_{\mathrm{MSE}}\left(\phi_{w,w^0}\right) = \frac{1}{n}\sum_{i=1}^{n}\left(w\ X_i^\top + w^0 - Y_i\right)^2 .$$

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 $w^0 \in \mathbb{R}$

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

A probabilistic perspective on OLS regression

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

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

A probabilistic perspective on OLS regression

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

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

$$\iff \quad Y_i - w\, X_i^\top - w^0 \sim \mathcal{N}(0, \sigma^2) \quad \text{with density} \quad f(z) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-z^2/(2\sigma^2)\right)$$

A probabilistic perspective on OLS regression

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

$Y_i = w\, X_i^\top + w^0 + \epsilon_i$ with $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ (i.i.d.)

$\Leftrightarrow$ $Y_i - w\, X_i^\top - w^0 \sim \mathcal{N}(0, \sigma^2)$ with density $f(z) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-z^2/(2\sigma^2)\right)$

$\Leftrightarrow$ Given data $\mathcal{D} = ((X_1, Y_1), \cdots, (X_n, Y_n))$ (i.i.d.) the likelihood is

$$\ell(w, w^0) = \prod_{i=1}^{n} \left\{ \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2} \left(w\, X_i^\top + w^0 - Y_i\right)^2 \right) \right\}.$$

A probabilistic perspective on OLS regression

Suppose we want to **maximize** the likelihood

$$\ell(w, w^0) = \prod_{i=1}^{n} \left\{ \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2} \left(w\, X_i^\top + w^0 - Y_i \right)^2 \right) \right\}.$$

This is the same as **maximizing** the log-likelihood

$$\log \ell(w, w^0) = -n \log(\sigma\sqrt{2\pi}) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(w\, X_i^\top + w^0 - Y_i \right)^2.$$

This is the same as **minimizing** the mean squared error on the training data

$$\hat{\mathcal{R}}_{\mathrm{MSE}}\left(\phi_{w,w^0}\right) = \frac{1}{n} \sum_{i=1}^{n} \left(w\, X_i^\top + w^0 - Y_i \right)^2.$$

A probabilistic perspective on OLS regression

Suppose we have training data $\mathcal{D} = ((X_1, Y_1), \cdots, (X_n, Y_n))$.

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

$$\hat{\mathcal{R}}_{\mathrm{MSE}}\left(\phi_{w,w^0}\right) = \frac{1}{n}\sum_{i=1}^{n}\left(w\, X_i^\top + w^0 - Y_i\right)^2 .$$

<u>CONCLUSION</u>: Minimising the squared training error is thethe same as **maximizing the likelihood**

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

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

A probabilistic perspective on OLS regression

Consider the Gaussian noise model

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

The OLS estimators $\hat{w} = \Sigma_{Y,X} \ (\Sigma_{X,X})^{-1}$ and $\hat{w}^0 = \overline{Y} - \hat{w}\, \overline{X}^\top$ are:

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

A probabilistic perspective on OLS regression

The OLS estimators $\hat{w} = \Sigma_{Y,X} \ (\Sigma_{X,X})^{-1}$ and $\hat{w}^0 = \overline{Y} - \hat{w} \ \overline{X}^\top$ are:

1. Maximum likelihood estimators of w and w^0.

2. Unbiased estimators with $\mathbb{E}[\hat{w}] = w$ and $\mathbb{E}[\hat{w}^0] = w^0$.

3. Minimum variance over all unbiased estimators.

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 today?

- 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 as an algorithm for linear regression.
- We saw advantages of OLS as unbiased, maximum likelihood estimators under a natural model.
- We also discussed how OLS can suffer from very variance, especially in high dimensions!

Thanks for listening!

henry.reeve@bristol.ac.uk

Include EMATM0061 in the subject of your email.

Supervised learning