

# Regression (Linear and Logistic)

## Machine Learning for Finance (FIN 570)

Instructor: Jaehyuk Choi

Peking University HSBC Business School, Shenzhen, China

2023-24 Module 3 (Spring 2024)

# Notations and conventions: vector and matrix

## General rules (guess from the context)

- Scalar (non-bold):  $x, y, X, Y$
- Vector (lowercase bold):  $\mathbf{x} = (x_i), \mathbf{y} = (y_i)$
- Matrix (uppercase bold):  $\mathbf{X} = (X_{ij}), \mathbf{Y} = (Y_{ij})$
- The  $(i, j)$  component of  $\mathbf{X}$ :  $X_{ij}$
- The  $i$ -th row vector of  $\mathbf{X}$ :  $\mathbf{X}_{i*} = (X_{i1}, X_{i2}, \dots, X_{ip})^T$
- The  $j$ -th column vector of  $\mathbf{X}$ :  $\mathbf{X}_{*j} = (X_{1j}, X_{2j}, \dots, X_{Nj})$

## Examples

- Dot product:  $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y}$
- Vector norm:  $|\mathbf{x}| = \sqrt{\mathbf{x}^T \mathbf{x}}$
- Matrix multiplication:  $\mathbf{Z} = \mathbf{X}\mathbf{Y} \rightarrow Z_{ij} = \mathbf{X}_{i*} \mathbf{Y}_{*j}$

# Notation and conventions: variables and observations

## General rules

- Generic (or representative) variables (uppercase non-bold):  $X$  (input),  $Y$  (output),  $G$  (classification output)
- The predictions:  $\hat{Y}$ ,  $\hat{G}$
- $X$  (input) may be  $p$ -dimensional (features/predictors):  $X_j$  ( $j \leq p$ ), **row** vector
- $Y$  (output) may be  $K$ -dimensional (responses):  $Y_k$  ( $k \leq K$ ), **row** vector.
- $N$  observations of  $X$  or  $Y$  are stacked over as **rows**:  
 $\mathbf{X}$  ( $N \times p$  matrix),  $\mathbf{Y}$  ( $N \times K$  matrix)
- The  $i$ -th observation of the  $j$ -th feature:  $X_{ij}$  v.s. **PML**:  $X_j^{(i)}$
- The  $i$ -th observation set:  $\mathbf{X}_{i*}$  ( $1 \times p$  row vector) v.s. **PML**:  $X^{(i)}$
- All observation of  $j$ -th feature  $X_j$ :  $\mathbf{X}_{*j}$  ( $N \times 1$  col. vector) v.s. **PML**:  $X_j$
- $\mathbf{X} = (\mathbf{X}_{*1} \cdots \mathbf{X}_{*p})$  (column-wise concatenation)
- The weight vectors,  $\beta$  or  $w$  (used interchangeably), are **columns** vectors.

# Simple Linear Regression (Ordinary Least Square)

For scalar predictor ( $X$ ) and response ( $Y$ ),

$$Y \approx \beta_0 + \beta_1 X \quad \longrightarrow \quad \hat{\mathbf{y}} = \beta_0 + \beta_1 \mathbf{x}.$$

For  $N$  observations  $(x_1, y_1), \dots, (x_N, y_N)$ , the set of  $(\hat{\beta}_0, \hat{\beta}_1)$  to minimize the residual sum of squares (RSS):

$$\text{RSS}(\beta) = \sum_{i=1}^N (y_i - \beta_0 - \beta_1 x_i)^2 = (\mathbf{y} - \beta_0 - \beta_1 \mathbf{x})^T (\mathbf{y} - \beta_0 - \beta_1 \mathbf{x})$$

is given as

$$\hat{\beta}_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2} = \frac{\text{Cov}(X, Y)}{\text{Var}(X)} \quad \text{and} \quad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

for  $\bar{x} = \sum x_i / N$  and  $\bar{y} = \sum y_i / N$ .

# Multi-dimensional Linear Regression

For  $(p + 1)$ -vector predictor ( $X$ ) and scalar response ( $Y$ ),

$$Y \approx X\beta \quad \longrightarrow \quad \hat{y} = X\beta,$$

where  $X_0 = 1$  ( $X_{*0} = \mathbf{1}$ ) and  $\beta$  is a  $(p + 1)$ -column vector.

$$\text{RSS}(\beta) = \frac{1}{2}(\mathbf{y} - X\beta)^T(\mathbf{y} - X\beta)$$

$$\frac{\partial}{\partial \beta} \text{RSS}(\beta) = -X^T(\mathbf{y} - X\beta) \quad \Rightarrow \quad \hat{\beta} = (X^T X)^{-1} X^T \mathbf{y}$$

For  $(p + 1)$ -vector predictor ( $X$ ) and  $K$ -vector response ( $Y$ ), the result is similarly given as

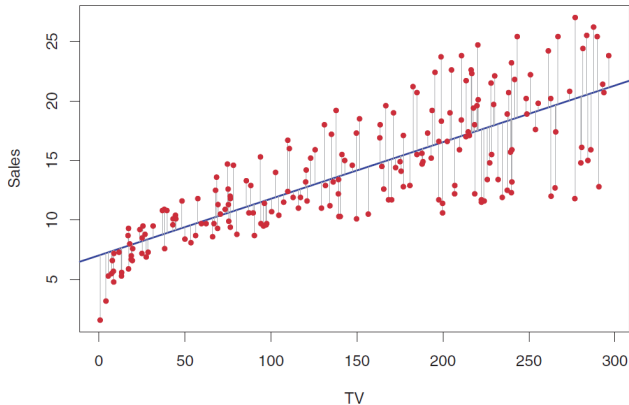
$$\hat{Y} = XB \quad \text{where} \quad \hat{B} = (X^T X)^{-1} X^T Y,$$

which is the independent regressions on  $Y_j$  ( $Y_{*j}$ ) combined together,

$$\hat{B}_{*j} = (X^T X)^{-1} X^T Y_{*j}$$

# An example

Figure 3.1 (p62) from **ISLR**

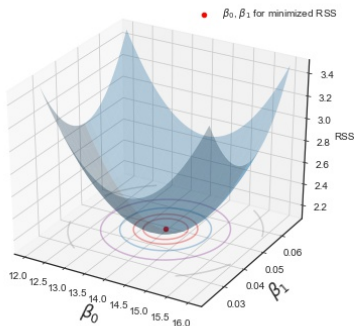
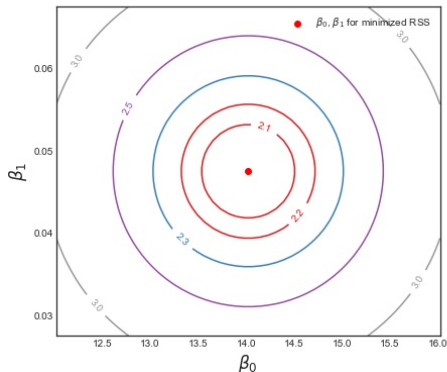


# The shape of RSS

The linear regression coefficients  $(\beta_0, \beta_1)$  indeed minimizes the RSS:

$$\text{RSS}(\beta) = \sum_{i=1}^N (y_i - \beta_0 - \beta_1 x_i)^2$$

RSS - Regression coefficients

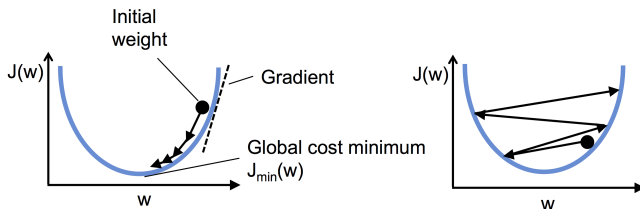


# Gradient Descent Method

- A first-order iterative optimization algorithm for finding a local minimum of the **objective/cost/loss function**,  $J(\mathbf{w})$ .
- The gradient,  $\nabla J(\mathbf{w})$ , is the direction of the steepest increase.
- We find the local minimum  $\mathbf{w}$  of a function  $J(\mathbf{w})$  by taking steps iteratively in the steepest descending direction on the current point:

$$\mathbf{w}^{(n+1)} = \mathbf{w}^{(n)} - \eta \nabla J(\mathbf{w}^{(n)})$$

- The constant  $\eta$ , called **learning rate**, controls how fast we descend. Typically we use  $0 < \eta < 1$  to avoid **overshooting**.





# Gradient Descent of Weight: A Simple Case

How can we search for the right weight  $w$  to minimize the error (or cost)?  
Imagine we fit a simple linear model  $y = xw$  to a single observations  $(x_1, y_1)$ . We need find  $w$  to minimize the RSS:

$$J(w) = \frac{1}{2}(y_1 - x_1 w)^2$$

Although we know the answer ( $w = y_1/x_1$ ), let's pretend that we need to improve  $w$  iteratively, such that

$$w := w + \Delta w \quad \text{with an initial guess } w^{(0)}.$$

The amount of update should be proportional to the derivative

$$\Delta w = -\eta \frac{d}{dw} J(w) = \eta(y_1 - x_1 w) x_1 = \eta(y_1 - \hat{y}_1) x_1.$$

The result is intuitive: the update is proportional to (i) the magnitude of error from  $(y_1 - x_1 w)$ , and (ii) the direction (sign) of update from  $x_1$ .  
You will see this equation often over the course!

# Gradient Descent of Weight: Linear Regression (Adaline)

Remind that the error (RSS) function and the gradient are

$$J(\mathbf{w}) = \frac{1}{2} \sum_i (y_i - \mathbf{X}_{i*} \mathbf{w})^2, \quad \frac{\partial}{\partial w_j} J(\mathbf{w}) = - \sum_i (y_i - \mathbf{X}_{i*} \mathbf{w}) X_{ij}.$$

The weight update rule, with **learning rate**  $\eta$ , is given by

$$\begin{aligned} \mathbf{w} &:= \mathbf{w} + \Delta \mathbf{w} \\ \Delta w_j &= -\eta \frac{\partial}{\partial w_j} J(\mathbf{w}) = \eta \sum_i (y_i - \mathbf{X}_{i*} \mathbf{w}) X_{ij} = \eta \sum_i (y_i - \hat{y}_i) X_{ij}. \end{aligned}$$

Perceptron's updating rule in **PML** Ch.2 is based on this result.

Note that one sample  $(y_i, \mathbf{X}_{i*})$  contributes  $(y_i - \hat{y}_i) X_{ij}$  to the update  $\Delta w_j$ . In **batch gradient descent**,  $\mathbf{w}$  is updated from all  $i$ 's. In **stochastic gradient descent** (iterative/online), however,  $\mathbf{w}$  is updated from randomly selected single  $i$ .

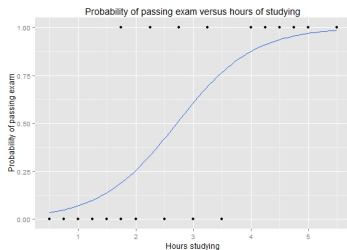
# Logistic Regression (Classification)

- Qualitative (categorical) response (binary dependent variable,  $Y \in \{0, 1\}$ )
- It is difficult to give numeric order to multiple categories: e.g., 0-1-2 vs 2-0-1?
- Linear regression (continuous variable) is not proper
- Logistic (sigmoid) function:  $\sigma(\text{logit}) = \text{quantile}$

$$p = \phi(t) = \frac{e^t}{1 + e^t} = \frac{1}{1 + e^{-t}} \quad \text{for } t = Xw \ (X_0 = 1)$$

- Logit function (the inverse): log odds

$$\phi^{-1}(p) = \log\left(\frac{p}{1-p}\right) = \log(p) - \log(1-p)$$



# Fitting of logistic regression

## Likelihood function

- For a given prediction model, measures the likelihood of a data set.
- The best prediction model/weight is the one that maximizes the likelihood of the dataset.

For a data set  $(\mathbf{X}, \mathbf{y})$  where  $y_i \in \{0, 1\}$ ,

$$\begin{aligned} L(\mathbf{w}) &= \prod_i P(y_i = \hat{y}_i) = \prod_{i:y_i=1} \phi(\mathbf{X}_{i*}\mathbf{w}) \prod_{i:y_i=0} (1 - \phi(\mathbf{X}_{i*}\mathbf{w})) \\ &= \prod_i \phi(\mathbf{X}_{i*}\mathbf{w})^{y_i} (1 - \phi(\mathbf{X}_{i*}\mathbf{w}))^{1-y_i} \\ \log L(\mathbf{w}) &= \sum_i y_i \log \phi(\mathbf{X}_{i*}\mathbf{w}) + (1 - y_i) \log (1 - \phi(\mathbf{X}_{i*}\mathbf{w})) \end{aligned}$$

The cost function (to minimize) is  $J(\mathbf{w}) = -\log L(\mathbf{w})$

# Gradient Descent of Weight: Logistic Regression

We use the properties of logistic function,

$$\frac{d}{dt}\phi(t) = \frac{e^{-t}}{(1 + e^{-t})^2} = \phi(t)(1 - \phi(t)), \quad \frac{\partial}{\partial w_j}\phi(\mathbf{X}_{i*}\mathbf{w}) = \phi(\cdot)(1 - \phi(\cdot))X_{ij},$$

the gradient of error function is obtained as

$$\begin{aligned}\frac{\partial}{\partial w_j}J(\mathbf{w}) &= \sum_i \left( -\frac{y_i}{\phi(\mathbf{X}_{i*}\mathbf{w})} + \frac{1 - y_i}{1 - \phi(\mathbf{X}_{i*}\mathbf{w})} \right) \frac{\partial}{\partial w_j}\phi(\mathbf{X}_{i*}\mathbf{w}) \\ &= \sum_i \left( -y_i(1 - \phi(\cdot)) + (1 - y_i)\phi(\cdot) \right) X_{ij} \\ &= -\sum_i (y_i - \phi(\cdot))X_{ij} = -\sum_i (y_i - \hat{y}_i)X_{ij}, \\ \Delta w_j &= \eta \sum_i (y_i - \hat{y}_i)X_{ij}.\end{aligned}$$

We get the exactly same weight updating rule as that of linear regression and Adaline!

# Regularization

To avoid **overfitting**, we do not want  $w$  to be too big. We add **penalty** for big  $w$ :

$$J(w) = -\log L(w) + \frac{\lambda}{2}|w|^2$$

$$J(w) = -C \log L(w) + \frac{1}{2}|w|^2 \quad (C = \frac{1}{\lambda}, \text{SciKit-Learn})$$

An example from polynomial curve fitting:

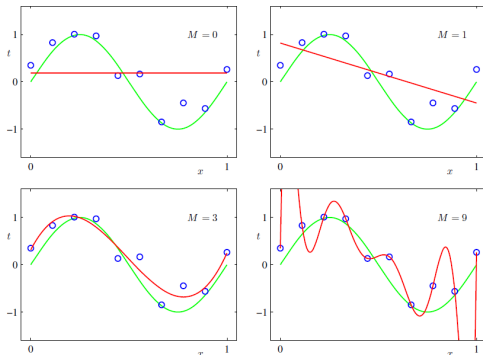


Figure 1.4 Plots of polynomials having various orders  $M$ , shown as red curves, fitted to the data set shown in Figure 1.2.

	$M = 0$	$M = 1$	$M = 6$	$M = 9$
$w_0^*$	0.19	0.82	0.31	0.35
$w_1^*$		-1.27	7.99	232.37
$w_2^*$			-25.43	-5321.83
$w_3^*$			17.37	48568.31
$w_4^*$				-231639.30
$w_5^*$				640042.26
$w_6^*$				-1061800.52
$w_7^*$				1042400.18
$w_8^*$				-557682.99
$w_9^*$				125201.43