Machine Learning for Finance (FIN 570)
Regression

Instructor: Jaehyuk Choi

Peking University HSBC Business School, Shenzhen, China

2018-19 Module 1 (Fall 2018)

Notations and conventions: vector and matrix

General rules (guess from the context)

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

Examples

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

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 ($j \le p$), row vector
- Y (output) may be K-dimensional (responses): Y_k ($k \le K$), row vector.
- The N observations of X or Y are stacked over as rows: $\boldsymbol{X}$ ($N \times p$ matrix), $\boldsymbol{Y}$ ($N \times K$ matrix)
- The i-th observation of the j-th feature: X_{ij} v.s. $X_j^{(i)}$ in **PML**
- The i-th observation set: $\boldsymbol{X}_{i*}$ ($1 \times p$ row vector) v.s. $X^{(i)}$ in **PML**
- All observation of j-th feature X_j: $\boldsymbol{X}_{*j}$ ($N \times 1$ row vector) v.s. X_j in **PML**
- $\boldsymbol{X} = (\boldsymbol{X}_{*1} \cdots \boldsymbol{X}_{*p})$ (column-wise concatenation)
- The weight vector, $\boldsymbol{\beta}$ or $\boldsymbol{w}$, are column vectors used interchangeably.

Simple Linear Regression (Ordinary Least Square)

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

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

For N observations $(x_1, y_1), \cdots, (x_N, y_N)$, the set of $(\hat\beta_0, \hat\beta_1)$ to minimize the residual sum of squares (RSS):

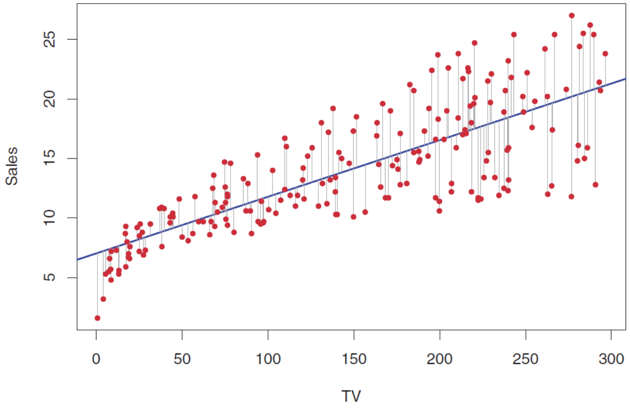
$$\mathsf{RSS}(\beta) = \sum_{i=1}^{N} (y_i - \beta_0 - \beta_1 x_i)^2 = (\boldsymbol{y} - \beta_0 - \beta_1 \boldsymbol{x})^T (\boldsymbol{y} - \beta_0 - \beta_1 \boldsymbol{x})$$

is given as

$$\hat\beta_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2} = \frac{\mathsf{Cov}(X, Y)}{\mathsf{Var}(X)},$$
$$\hat\beta_0 = \bar{y} - \hat\beta_1 \bar{x}$$

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

Figure 3.1 (p62) from **ISLR**



Multi-dimensional Linear Regression

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

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

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

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

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

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

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

which is the independent regressions on Y_j ($\boldsymbol{Y}_{*j}$) combined together,

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

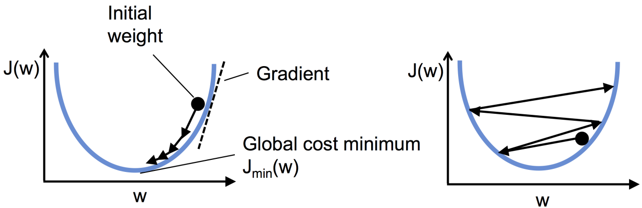
Gradient Descent

A first-order iterative optimization algorithm for finding the minimum of a function. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or of the approximate gradient) of the function at the current point

The minimum location $J(\boldsymbol{x})$ can be found by the following iteration:

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \eta \boldsymbol{\nabla} J(\boldsymbol{x}_n)$$

The constant η is called *learning rate*. Typically we use $0 < \eta < 1$ to avoid *overshooting*.



Gradient Descent of Weight: A Simple Case

How can we search for the right weight $\boldsymbol{w}$ to minize 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

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

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

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

The weight update rule, with *learning rate* η, is given by

$$\boldsymbol{w} := \boldsymbol{w} + \Delta\boldsymbol{w}$$

$$\Delta w_j = -\eta \frac{\partial}{\partial w_j} J(\boldsymbol{w}) = \eta \sum_i (y_i - \boldsymbol{X}_{i*}\boldsymbol{w}) X_{ij} = \eta \sum_i (y_i - \hat{y}_i) X_{ij}.$$

Perceptron's updating rule in **PML** Ch.2 is based on this result.
Note that one sample $(y_i, \boldsymbol{X}_{i*})$ contributes $(y_i - \hat{y}_i)X_{ij}$ to the update Δw_j. In **batch gradient descent**, $\boldsymbol{w}$ is updated from all i's. In **stochastic gradient descent** (iterative/online), however, $\boldsymbol{w}$ is updated from randomly selected single i.

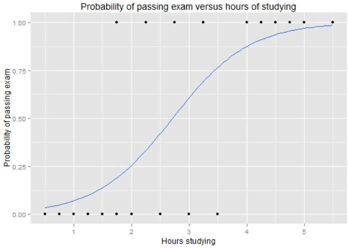
Logistic Regression (Classification)

- 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 (quantitative) is not proper
- Logistic (sigmoid) function: $\sigma(\text{logit}) = \text{quantile}$

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

- 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 the 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 $(\boldsymbol{X}, \boldsymbol{y})$ where $y_i \in \{0, 1\}$),

$$L(\boldsymbol{w}) = \prod_i P(y_i = \hat{y}_i) = \prod_{i:y_i=1} \phi(\boldsymbol{X}_{i*}\boldsymbol{w}) \prod_{i:y_i=0} \big(1 - \phi(\boldsymbol{X}_{i*}\boldsymbol{w})\big)$$

$$= \prod_i \phi(\boldsymbol{X}_{i*}\boldsymbol{w})^{y_i} \big(1 - \phi(\boldsymbol{X}_{i*}\boldsymbol{w})\big)^{1-y_i}$$

$$\log L(\boldsymbol{w}) = \sum_i y_i \log \phi(\boldsymbol{X}_{i*}\boldsymbol{w}) + (1 - y_i) \log \big(1 - \phi(\boldsymbol{X}_{i*}\boldsymbol{w})\big)$$

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

Gradient Descent of Weight: Logistic Regression

We use the properties of logistic function,

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

the gradient of error function is obtained as

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

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

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

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

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

An example from polynomial curve fitting:

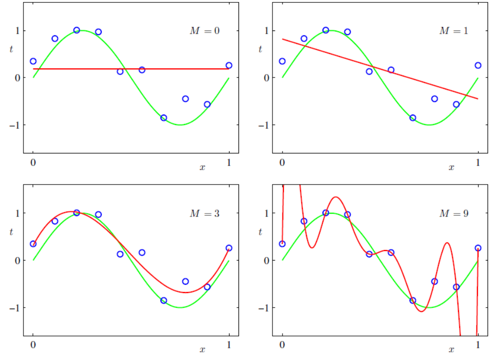


Figure 1.4 Plots of polynomials having various orders M, shown as red curves, fitted to the data set shown in Figure 1.2.

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