

# Introduction to Algorithms for Data Mining and Machine Learning

## Chapter 4: Data Fitting and Regression

Xin-She Yang

For details, please read the book:

Xin-She Yang, [Introduction to Algorithms for Data Mining and Machine Learning](#),  
Academic Press/Elsevier, (2019).

# Sample Mean and Variance

For a given set of  $n$  independent samples/observations  $x_1, x_2, \dots, x_n$  of a random variable  $X$ , the sample mean can be calculated by

$$\bar{x} \equiv \langle x \rangle = \frac{1}{n}(x_1 + x_2 + \dots + x_n) = \frac{1}{n} \sum_{i=1}^n x_i,$$

which is essentially the arithmetic average of the values  $x_i$ .

## Sample Variance

The sample variance  $S^2$  is defined by

$$S^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2.$$

It is worth pointing out that the factor is  $1/(n-1)$ , not  $1/n$ . Loosely speaking, we need at least one sample to estimate the mean, but we need at least two samples to estimate the variance.

## Example

The measurements of a quantity such as the noise level. The readings in dB are :

66, 73, 73, 74, 83, 70, 69, 77, 72, 75.

From the data, we know that  $n = 10$  and the mode is 73 as 73 appears twice (all the rest only appears once). The sample mean is

$$\begin{aligned}\bar{x} &= \frac{1}{10}(x_1 + x_2 + \dots + x_{10}) \\ &= \frac{1}{10}(66 + 73 + 73 + 74 + 83 + 70 + 69 + 77 + 72 + 75) = \frac{732}{10} = 73.2.\end{aligned}$$

The corresponding sample variance can be calculated by

$$\begin{aligned}S^2 &= \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2 \\ &= \frac{1}{10-1} \sum_{i=1}^{10} (x_i - 73.2)^2 = \frac{1}{9} [(66 - 73.2)^2 + (73 - 73.2)^2 + \dots + (75 - 73.2)^2] \\ &= \frac{1}{9} [(-7.2)^2 + (-0.2)^2 + \dots + (1.8)^2] = \frac{195.6}{9} \approx 21.73.\end{aligned}$$

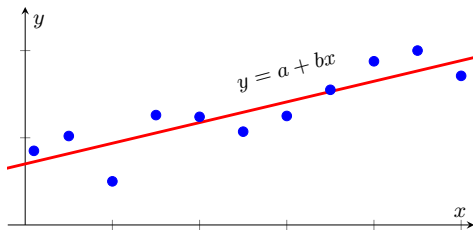
Thus, the standard derivation is  $S = \sqrt{S^2} \approx \sqrt{21.73} \approx 4.662$ .

# Regression

The main idea of regression is to fit the data points  $(x_i, y_i)$  to a function  $y = f(x)$  (e.g.,  $y = a + bx$  is a straight line). The best fit should minimize the sum squares of errors  $\sum_{i=1}^n [y_i - f(x_i)]^2$ . That is

$$\min \Psi = \sum_{i=1}^n [y_i - f(x_i)]^2,$$

which is the **method of least squares**.



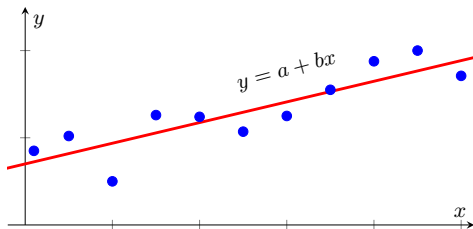
Best fit line for a simple linear model.

# Regression

The main idea of regression is to fit the data points  $(x_i, y_i)$  to a function  $y = f(x)$  (e.g.,  $y = a + bx$  is a straight line). The best fit should minimize the sum squares of errors  $\sum_{i=1}^n [y_i - f(x_i)]^2$ . That is

$$\min \Psi = \sum_{i=1}^n [y_i - f(x_i)]^2,$$

which is the **method of least squares**.



Best fit line for a simple linear model.

From  $\frac{\partial \Psi}{\partial a} = 0$  and  $\frac{\partial \Psi}{\partial b} = 0$ , we have

$$\sum_{i=1}^n [y_i - (a + bx_i)] = 0, \quad \sum_{i=1}^n x_i [y_i - (a + bx_i)] = 0,$$

which can determine  $a$  and  $b$  uniquely. After some algebra, we have

$$a = \frac{1}{n} \left[ \sum_{i=1}^n y_i - b \sum_{i=1}^n x_i \right] = \bar{y} - b\bar{x}, \quad b = \frac{n \sum_{i=1}^n x_i y_i - (\sum_{i=1}^n x_i)(\sum_{i=1}^n y_i)}{n \sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2},$$

where  $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$  and  $\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$  are the sample means.

# Correlation Coefficient

The correlation coefficient measures the degree of correlation between  $x$  and  $y$ .

$$r = r_{x,y} = \frac{n \sum_{i=1}^n x_i y_i - \sum_{i=1}^n x_i \sum_{i=1}^n y_i}{\sqrt{\left[ n \sum_{i=1}^n x_i^2 - \left( \sum_{i=1}^n x_i \right)^2 \right] \left[ n \sum_{i=1}^n y_i^2 - \left( \sum_{i=1}^n y_i \right)^2 \right]}},$$

which can also be written compactly as

$$r = \frac{nK_{xy} - K_x K_y}{\sqrt{(nK_{xx} - K_x^2)(nK_{yy} - K_y^2)}},$$

$$K_x = \sum_{i=1}^n x_i, \quad K_y = \sum_{i=1}^n y_i, \quad K_{xx} = \sum_{i=1}^n x_i^2, \quad K_{yy} = \sum_{i=1}^n y_i^2, \quad K_{xy} = \sum_{i=1}^n x_i y_i.$$

$$-1 \leq r \leq +1$$

- $r > 0$ , it is positive correlation. That is,  $y$  increases as  $x$  increases. In case  $r = 1$ , it is a perfect, increasing linear relationship.
- $r = 0$  (or near zero), there is no correlation between  $y$  and  $x$ .
- $r < 0$ , it is negative correlation.  $r = -1$  corresponds to a perfect, decreasing, linear relationship.

However, if  $|r| > 1$ , there must be some mistake in calculations!

For given the following given data:

$x_i$	1	2	3	4	5
$y_i$	1.6	2.1	2.5	2.9	3.7

We have

$$K_x = \sum_{i=1}^5 x_i = 1 + 2 + 3 + 4 + 5 = 15, \quad K_y = \sum_{i=1}^5 y_i = 1.6 + 2.1 + 2.5 + 2.9 + 3.7 = 12.8$$

$$K_{xx} = \sum_{i=1}^5 x_i^2 = 1^2 + \dots + 5^2 = 55, \quad K_{yy} = \sum_{i=1}^5 y_i^2 = 1.6^2 + \dots + 3.7^2 = 35.32$$

$$K_{xy} = \sum_{i=1}^5 x_i y_i = 1 \times 1.6 + \dots + 5 \times 3.7 = 43.4$$

So we have

$$a = \frac{K_{xx}K_y - K_xK_{xy}}{nK_{xx} - K_x^2} = \frac{55 \times 12.8 - 15 \times 43.4}{5 \times 55 - (15)^2} = 1.06,$$

$$b = \frac{nK_{xy} - K_xK_y}{nK_{xx} - K_x^2} = \frac{5 \times 43.4 - 15 \times 12.8}{5 \times 55 - (15)^2} = 0.5.$$

So the best-fit line is

$$y = a + bx = 1.06 + 0.5x.$$

The correlation coefficient is

$$r = \frac{nK_{xy} - K_xK_y}{\sqrt{(nK_{xx} - K_x^2)(nK_{yy} - K_y^2)}} = 0.98976.$$

Exercise: Carry out the above regression in R and compare the results.

# Linearization

Some model functions look nonlinear, but they can be converted into a linear form.

## Example

- Nonlinear function

$$f(x) = \alpha e^{-\beta x},$$

can be transformed into a linear form by taking logarithms of both sides

$$\ln f(x) = \ln(\alpha) - \beta x \implies y = a + bx, \quad y = \ln f(x), \quad a = \ln(\alpha), \quad b = -\beta.$$

- Function  $f(x) = \alpha x^\beta$  can also be transformed into

$$\ln[f(x)] = \ln(\alpha) + \beta \ln(x),$$

which becomes  $y = a + bv$  between  $y = \ln[f(x)]$  and  $v = \ln(x)$  with  $a = \ln(\alpha)$  and  $b = \beta$ .

- Similarly,

$$y = ax \exp(-x/b), \quad x, y > 0$$

can be converted to

$$\frac{y}{x} = ae^{-x/b}, \quad x \neq 0,$$

and then

$$\ln\left(\frac{y}{x}\right) = \ln a - \frac{1}{b}x,$$

which is linear between  $\ln(y/x)$  and  $x$ .



# Generalized Linear Regression

Fitting to a polynomial of degree  $p$

$$y(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \dots + \alpha_p x^p,$$

is probably the most widely used. This is equivalent to the regression to the linear combination of the **basis functions**  $1, x, x^2, \dots, x^p$ .

For given data points  $(x_i, y_i)$  ( $i = 1, 2, \dots, n$ ), the minimization of the sum of the errors squared leads to

$$\begin{pmatrix} \sum_{i=1}^n 1 & \sum_{i=1}^n x_i & \dots & \sum_{i=1}^n x_i^p \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 & \dots & \sum_{i=1}^n x_i^{p+1} \\ \vdots & & \ddots & \\ \sum_{i=1}^n x_i^p & \sum_{i=1}^n x_i^{p+1} & \dots & \sum_{i=1}^n x_i^{2p} \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_p \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i y_i \\ \vdots \\ \sum_{i=1}^n x_i^p y_i \end{pmatrix},$$

which is a matrix equation. In principle,  $\alpha = (\alpha_1, \dots, \alpha_p)^T$  can be obtained by inversion, though the Gauss-Newton method is more efficient.

## Example: Quadratic Regression

 $x : -0.98, 1.00, 2.02, 3.03, 4.00$ 
 $y : 2.44, -1.51, -0.47, 2.54, 7.52$ 

For the formula  $y = \alpha_0 + \alpha_1 x + \alpha_2 x^2$ , we have

$$\begin{pmatrix} n & \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i^3 \\ \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i^3 & \sum_{i=1}^n x_i^4 \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i y_i \\ \sum_{i=1}^n x_i^2 y_i \end{pmatrix}.$$

Using the data set, we have  $n = 5$ ,  $\sum_{i=1}^n x_i = 9.07$  and  $\sum_{i=1}^n y_i = 10.52$ .

Other quantities can be calculated in a similar way. Therefore, we have

$$\begin{pmatrix} 5.0000 & 9.0700 & 31.2217 \\ 9.0700 & 31.2217 & 100.119 \\ 31.2217 & 100.119 & 358.861 \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{pmatrix} = \begin{pmatrix} 10.52 \\ 32.9256 \\ 142.5551 \end{pmatrix}.$$

By direct inversion, we have

$$\begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{pmatrix} = \begin{pmatrix} -0.5055 \\ -2.0262 \\ 1.0065 \end{pmatrix}.$$

The best fit equation is

$$y(x) = -0.5055 - 2.0262x + 1.0065x^2.$$

Exercise: Implement this using R or Python and compare the results.

# Goodness of Fit

In the preceding example, we fit the data using  $p = 2$  (quadratic). If we plot out the curve, it is well fit. However, without visualizing the data, **how do we know it is well fit by choosing  $p = 2$ ?** In fact,  $p$  is hyper-parameter (that we have to supply a value).

Using different  $p = 1, 2, 3, 4$ , we have  $f_p(x)$ :

$$f_1(x) = 0.9373x + 0.4038 \text{ for } p = 1.$$

$$f_2(x) = 1.0065x^2 - 2.0262x - 0.5055 \text{ (best fit).}$$

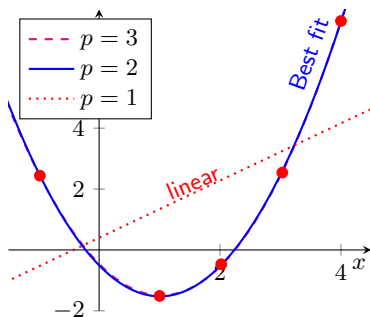
$$f_3(x) = 0.0080x^3 + 0.9694x^2 - 2.0131x - 0.4580.$$

$$f_4(x) = 0.0101x^4 - 0.0610x^3 + 1.0778x^2 - 1.9571x - 0.5798.$$

The fit error is the **residual sum of squares (RSS)**

$$\text{RSS}(p) = \sum_{i=1}^n [y_i - f_p(x_i)]^2 = \sum_{i=1}^n [y_i - \hat{y}_i(x_i)]^2,$$

where  $\hat{y}_i(x_i) = f_p(x_i)$  are the predicted values.



The best fit is a quadratic function ( $p = 2$ ).

## Goodness of fit

Order	$p = 1$	$p = 2$	$p = 3$	$p = 4$
RSS	36.3485	0.0045	0.0080	$6.9 \times 10^{-30}$

$p = 4$  leads to overfitting.

# Over-fitting

However, **higher-order polynomials** tend to **overfit the data**, and oscillations become an issue for higher  $p$ . So, RSS is not a good criterion for measuring the goodness of the fit.

To choose the proper order  $p$  (hyperparameter) to avoid overfitting, better criteria are Bayesian information criterion (BIC) or Akaike information criterion (AIC).

## Information Criteria: Minimization of AIC or BIC

For a model  $y = f(x)$  with  $k$  parameters to fit  $n$  data points, the AIC is defined by

$$\text{AIC} = 2k + n \ln(\text{RSS}/n).$$

For small number of samples (e.g.,  $n/k < 40$ ), the corrected AIC (i.e., AICc) becomes

$$\text{AICc} = 2k + n \ln(\text{RSS}/n) + \frac{2k(k+1)}{n-k-1}.$$

With the same assumptions of errors obeying Gaussian distributions, we have

$$\text{BIC} = k \ln n + n \ln \left( \frac{\text{RSS}}{n} \right).$$

**Both AIC and BIC are useful criteria, and their global minima give the best fit ( $k$  value).** It is difficult to say which one is better, which may also depend on the types of problems.

Let us revisit the previous example for fitting  $n = 5$  data points with polynomials.

Using the AIC criterion, and we have  $n = 5$  and  $k = p + 1 = 2$  (for  $p = 1$ )

$$\text{AIC} = 2 \times 2 + 5 \ln(36.3485/5) = 13.64.$$

Similarly, we have

$$\text{AIC} = -29.07, \quad -24.19,$$

for  $p = 2, 3$ , respectively. Among these three values,  $p = 2$  has the lowest AIC and AICc. Since the values start to increase for  $p = 3$ , we can conclude that the best degree of fit is  $p = 2$  with  $k = 3$  parameters. The results of AIC and AICc values are summarized in the Table below.

Order	$p = 1$	$p = 2$	$p = 3$
$k = p + 1$	2	3	4
RSS	36.3485	0.0045	0.0080
AIC	13.92	-29.07	-24.19
AICc	15.92	-17.07	Inf (division by zero)
BIC	13.14	-30.24	-25.75

We also list the BIC values and the conclusion  $p = 2$  is consistent with AIC/AICc.

# Nonlinear Least Squares

## Curve-fitting is optimization

Linear models are rare/special cases. In general, for  $n$  data points  $(x_i, y_i)$  ( $i = 1, 2, \dots, n$ ) to fit a model  $f(x_i, \mathbf{a})$  with  $\mathbf{a} = (a_0, a_1, \dots, a_m)$  (i.e.,  $m$  parameters), we have the method of nonlinear least squares to minimize the  $L_2$ -norm of the residuals  $R_i = y_i - f(x_i, \mathbf{a})$ . That is to minimize the sum of the fit errors/residuals:

$$\text{Minimize } E(\mathbf{a}) = \sum_{i=1}^n R_i^2(\mathbf{a}) = \sum_{i=1}^n [y_i - f(x_i, \mathbf{a})]^2 = \|\mathbf{R}_i(\mathbf{a})\|_2^2,$$

which can be solved by optimization techniques such as the Gauss-Newton Algorithm.

## Gauss-Newton Algorithm

Let  $\mathbf{J}$  denote the Jacobian matrix in the form

$$\mathbf{J} = [J_{ij}] = \frac{\partial R_i}{\partial a_j}, \quad i = 1, 2, \dots, n, \quad j = 0, 1, 2, \dots, m,$$

which is an  $n \times (m + 1)$  matrix. Then the gradient of the objective (error) function is

$$\frac{\partial E}{\partial a_j} = 2 \sum_{i=1}^n \frac{\partial R_i}{\partial a_j} R_i, \quad j = 0, 1, 2, \dots, m.$$

# Gauss-Newton Algorithm

This gradient can be written in the vector form  $\nabla E$  with  $m + 1$  components as

$$\nabla E(\mathbf{a}) = 2\mathbf{J}^T \mathbf{R},$$

where the residual vector  $\mathbf{R}$  is given by  $\mathbf{R}(\mathbf{a}) = [R_1(\mathbf{a}), R_2(\mathbf{a}), \dots, R_n(\mathbf{a})]^T$ . The Hessian matrix  $\mathbf{H}$  [of size  $(m + 1) \times (m + 1)$ ] of  $E$  can be written as

$$\mathbf{H} = \nabla^2 E = 2 \sum_{i=1}^n [\nabla R_i \nabla R_i^T + R_i \nabla^2 R_i] = 2\mathbf{J}^T \mathbf{J} + 2 \sum_{i=1}^n R_i \nabla^2 R_i,$$

which can be approximated by (after ignoring all the higher-order terms)

$$\mathbf{H} \approx 2\mathbf{J}^T \mathbf{J}.$$

The minimization of  $E(\mathbf{a})$  can be solved by Newton's method (starting with an initial guess  $\mathbf{a}_0$ ), and we have

$$\mathbf{a}_{t+1} = \mathbf{a}_t - \frac{\nabla E}{\nabla^2 E} = \mathbf{a}_t - \frac{\nabla E}{\mathbf{H}} = \mathbf{a}_t - \frac{2\mathbf{J}^T \mathbf{R}}{2\mathbf{J}^T \mathbf{J}} = \mathbf{a}_t - (\mathbf{J}^T \mathbf{J})^{-1} \mathbf{J}^T \mathbf{R}(\mathbf{a}_t).$$

This is the Gauss-Newton method, which requires  $\mathbf{J}$  to be full rank [so  $(\mathbf{J}^T \mathbf{J})^{-1}$  exists].

Many software packages have implemented this method.

As an example, let us use the following data

$$x : 0.10, 0.50, 1.0, 1.5, 2.0, 2.5$$

$$y : 0.10, 0.28, 0.40, 0.40, 0.37, 0.32$$

to fit a model

$$y = \frac{x}{a + bx^2},$$

where  $a$  and  $b$  are the coefficients to be determined by the data. The objective is to minimize the sum of the residual squares


$$S = \sum_{i=1}^6 R_i^2 = \sum_{i=1}^6 \left[ 1 - \frac{x}{a + bx^2} \right]^2, \quad R_i = y_i - \frac{x_i}{a + bx_i^2}.$$

Since

$$\frac{\partial R_i}{\partial a} = \frac{x_i}{(a + bx_i^2)^2}, \quad \frac{\partial R_i}{\partial b} = \frac{x_i^3}{(a + bx_i^2)^2}.$$

If we use the initial guess  $a = 1$  and  $b = 1$ , then the initial residuals are

$$\mathbf{R} = (0.0010 \quad -0.1200 \quad -0.1000 \quad -0.0615 \quad -0.0300 \quad -0.0248)^T.$$

Exercise: Implement this example using R or Python. 



The initial Jacobian matrix is

$$\mathbf{J} = \begin{pmatrix} 0.0980 & 0.0010 \\ 0.3200 & 0.0800 \\ 0.2500 & 0.2500 \\ 0.1420 & 0.3195 \\ 0.0800 & 0.3200 \\ 0.0476 & 0.2973 \end{pmatrix}.$$

Thus, the first iteration using the Gauss-Newton algorithm gives

$$\begin{pmatrix} a \\ b \end{pmatrix}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} - (\mathbf{J}^T \mathbf{J})^{-1} \mathbf{J} \mathbf{R} = \begin{pmatrix} 1.3449 \\ 1.0317 \end{pmatrix}.$$

Then, by updating the new Jacobian and residuals, we have

$$a = 1.4742, \quad b = 1.0059,$$

after the second iteration. Similarly, we have

$$a = 1.4852, \quad b = 1.0022 \quad (\text{third iteration}),$$

$$a = 1.4854, \quad b = 1.0021 \quad (\text{fourth iteration}).$$

It converges quickly (parameters almost remain the same values after 10 iterations).

# Levenberg-Marquardt Algorithm

The Levenberg-Marquardt algorithm is more robust by using a damping term in the approximation of the Hessian, that is

$$\mathbf{H} \approx 2[\mathbf{J}^T \mathbf{J} + \mu \mathbf{I}],$$

where  $\mu > 0$  is the damping coefficient, also called Marquardt parameter, and  $\mathbf{I}$  is the identity matrix of the same size as  $\mathbf{H}$ . Thus, the iteration formula becomes

$$\mathbf{a}_{t+1} = \mathbf{a}_t - \frac{\mathbf{J}^T \mathbf{R}(\mathbf{a}_t)}{\mathbf{J}^T \mathbf{J} + \mu \mathbf{I}} = \mathbf{a}_t - (\mathbf{J}^T \mathbf{J} + \mu \mathbf{I})^{-1} \mathbf{J}^T \mathbf{R}(\mathbf{a}_t),$$

which is equivalent to the step size  $\mathbf{s}_t$  in the line search given by

$$(\mathbf{J}^T \mathbf{J} + \mu \mathbf{I}) \mathbf{s}_t = -\mathbf{J}^T \mathbf{R}(\mathbf{a}_t), \quad \mathbf{a}_{t+1} = \mathbf{a}_t + \mathbf{s}_t.$$

Mathematically speaking, a large  $\mu$  will effectively reduce the step size (in comparison with those in the Gauss-Newton algorithm) and damps the moves with the right amount.

- If the reduction in  $E$  is sufficient, we can either keep this value of  $\mu$  or reduce it. However, if the reduction  $E$  is not sufficient, we can increase  $\mu$ .
- If  $\mu = 0$ , this algorithm reduces to the standard Gauss-Newton algorithm.
- Since  $\mu \neq 0$ ,  $(\mathbf{J}^T \mathbf{J} + \mu \mathbf{I})$  is always full rank (thus invertible).

# Weighted Least Squares

The methods so far implicitly assume that all data points have **equal variance  $\sigma^2$** . To consider the possible different variances  $\sigma_i^2$ , we have the weighted least squares.

## Weighted Least Squares

$$\text{Minimize } \sum_{i=1}^n w_i R_i^2 = \sum_{i=1}^n \frac{R_i^2}{\sigma_i^2} = \left\| \frac{R_i}{\sigma_i} \right\|_2^2, \quad R_i = y_i - f(x_i, \mathbf{a}).$$

Here,  $w_i = 1/\sigma_i^2$  and  $\sigma_i^2$  is the variance associated with data point  $(x_i, y_i)$ .

By defining a weight matrix  $\mathbf{W}$  as

$$\mathbf{W} = \text{diag}(w_i) = \begin{pmatrix} w_1 & 0 & \dots & 0 \\ 0 & w_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & w_n \end{pmatrix},$$

and following the similar derivations as above, steps/increments will become

$$\mathbf{a}_{t+1} = \mathbf{a}_t - (\mathbf{J}^T \mathbf{W} \mathbf{J})^{-1} (\mathbf{J}^T \mathbf{W} \mathbf{R}),$$

which is equivalent to approximating the Hessian by  $\mathbf{H} = 2\mathbf{J}^T \mathbf{W} \mathbf{J}$  and the gradient by  $\nabla E = 2\mathbf{J}^T \mathbf{W} \mathbf{R}$ .

# Regularization

Regularization is another approach to deal with overfitting. In case of multi-variate cases with  $p$  components  $\mathbf{Z} = (Z_1, Z_2, \dots, Z_p)^T$ , we have the linear regression model

$$y = \beta_0 + \beta_1 Z_1 + \beta_2 Z_2 + \dots + \beta_p Z_p = \mathbf{Z}^T \boldsymbol{\beta} + \beta_0,$$

where  $\beta_0$  is the bias, while  $\boldsymbol{\beta} = (\beta_1, \beta_2, \dots, \beta_p)^T$  is the coefficient vector.

The standard method of least squares is to minimize the residual sum of squares (RSS)

$$\text{Minimize } \sum_{i=1}^n (y_i - \mathbf{Z}_i^T \boldsymbol{\beta} - \beta_0)^2.$$

## Ridge Regression

The Ridge regression uses a penalized RSS in the form

$$\text{Minimize } \sum_{i=1}^n (y_i - \mathbf{Z}_i^T \boldsymbol{\beta} - \beta_0)^2 + \lambda \sum_{j=1}^p \beta_j^2,$$

where  $\lambda$  is the penalty coefficient. The above formula can be written compactly as

$$\text{Minimize } \|\mathbf{y} - \beta_0 \mathbf{1} - \mathbf{Z}^T \boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_2^2,$$

where  $\|\cdot\|_2$  is the  $L_2$ -norm, similar to Tikhonov regularization.

Here the bias  $\beta_0$  is not part of the penalty or regularization term [we can always pre-process the data  $y_i$  (by subtracting their mean) so that the bias  $\beta_0$  becomes zero].

# Lasso Method

## Lasso

The Lasso method uses an  $L_1$ -norm in the regularization term

$$\text{Minimize } ||y - \beta_0 - \mathbf{Z}^T \boldsymbol{\beta}||_2^2 + \lambda ||\boldsymbol{\beta}||_1,$$

which is equivalent to the following minimization problem:

$$\text{Minimize } \sum_{i=1}^n (y_i - \beta_0 - \mathbf{Z}_i^T \boldsymbol{\beta})^2,$$

subject to  $||\boldsymbol{\beta}||_1 = |\beta_1| + |\beta_2| + \dots + |\beta_p| \leq \delta$  where  $\delta > 0$  is a predefined hyper-parameter. [Again,  $\beta_0$  is not penalized.]

## Hybrid Regularization

A hybrid method is the elastic net regularization or regression, which combines the Ridge and Lasso methods into a hybrid as

$$\text{Minimize } ||y - \beta_0 - \mathbf{Z}^T \boldsymbol{\beta}||_2^2 + \lambda_1 ||\boldsymbol{\beta}||_1 + \lambda_2 ||\mathbf{b}||_2^2.$$

Both the  $L_1$ -norm and  $L_2$ -norm are used with two hyper-parameters  $\lambda_1$  and  $\lambda_2$ .

These regularization methods have been implemented in many software packages.

# References

## References

- Xin-She Yang, **Introduction to Algorithms for Data Mining and Machine Learning**, Academic Press/Elsevier, (2019).
- Xin-She Yang, **Optimization Techniques and Applications with Examples**, John Wiley & Sons, (2018).

## Notes on Software

- Matlab: Matlab has some well-tested curve-fitting tools `fit` and `polyfit`, nonlinear least squares `lsqnonlin` (including Levenberg-Marquardt method), generalized least squares `fitnlm` and `lsqcurvefit`, Lasso method `lassglm`, and many others.
- Octave: Octave has linear least squares `lsqin`, exponential fit `expfit`, Levenberg-Marquardt nonlinear regression `leqsqr`, polynomial fit `polyfitinf`, and nonlinear least squares `lsqnonlin`.
- R: R has many functions for processing and visualizing data, including linear regression `lm()`, nonlinear least squares `fit nls` and many other statistical functionalities.
- Python: Python has at least two modules for regression `statsmodels` and `Scikit-learn`. The module `Scikit learn sklearn` is mainly for data mining and machine learning, which can do k-means clustering and basic clustering as well as various statistical analysis.

Any questions?

Thank you :)