Linear Regression

Introduction to Artificial Intelligence with Mathematics
Lecture Notes

Ganguk Hwang

Department of Mathematical Sciences
KAIST

Ordinary Linear Regression

Linear Regression Model

Data: $(\mathbf{x}_i, y_i) = (x_{i1}, x_{i2}, \cdots, x_{ik}, y_i),\ i = 1, 2, \cdots, n$ for $n > k$

Model:

$$\begin{aligned} y_i &= \beta_0 + x_{i1}\beta_1 + x_{i2}\beta_2 + \cdots + x_{ik}\beta_k + \epsilon_i \\ &= \beta_0 + \sum_{j=1}^{k} x_{ij}\beta_j + \epsilon_i \end{aligned}$$

We want to find a linear regression line that explains the data set well.

In matrix form we have

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \epsilon$$

where

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1k} \\ 1 & x_{21} & x_{22} & \cdots & x_{2k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nk} \end{bmatrix}, \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix}, \mathbf{e} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}.$$

Our objective is to minimize

$$\sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} (y_i - (\beta_0 + \sum_{j=1}^{k} x_{ij}\beta_j))^2$$

which is equivalent to mininmize

$$f(\boldsymbol{\beta}) := ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||^2.$$

Note that

$$f(\boldsymbol{\beta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}).$$

Statistical Approach: Normal equations

In a statistical approach, we obtain

$$\begin{aligned}
\nabla_\beta f(\beta) &= \nabla_\beta (\mathbf{y} - \mathbf{X}\beta)^\top (\mathbf{y} - \mathbf{X}\beta) \\
&= \nabla_\beta (\mathbf{y}^\top - \beta^\top \mathbf{X}^\top)(\mathbf{y} - \mathbf{X}\beta) \\
&= \nabla_\beta (\mathbf{y}^\top \mathbf{y} - \beta^\top \mathbf{X}^\top \mathbf{y} - \mathbf{y}^\top \mathbf{X}\beta + \beta^\top \mathbf{X}^\top \mathbf{X}\beta) \\
&= -\mathbf{X}^\top \mathbf{y} - \mathbf{X}^\top \mathbf{y} + \mathbf{X}^\top \mathbf{X}\beta + \mathbf{X}^\top \mathbf{X}\beta \\
&= 2(\mathbf{X}^\top \mathbf{X}\beta - \mathbf{X}^\top \mathbf{y}).
\end{aligned}$$

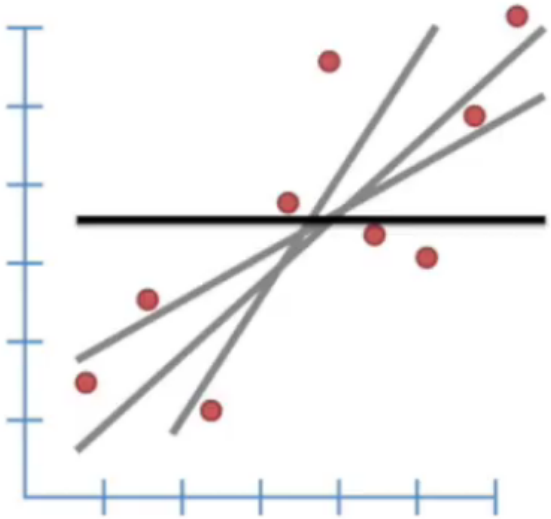
From $\nabla_\beta f(\beta) = 0$, the least square estimates (LSEs) $\hat{\beta}$ is given by

$$\hat{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}.$$

Linear Regression - Machine Learning Approach

Linear Regression

- The object is to find a linear line that explains the data set well.
- Which line is the “best”?

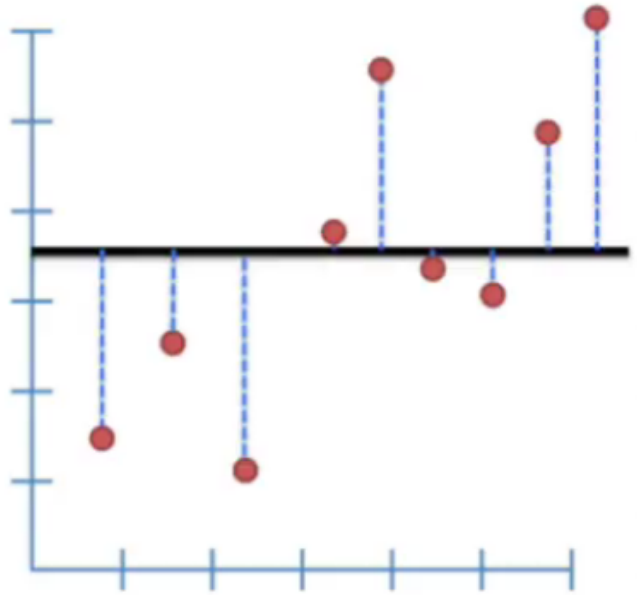


(Ref: StatQuest: Linear Models - Josh Starmer)

Sum of Squared Errors for Benchmark Line

The horizontal line is our benchmark line and we compute the sum of the squared errors (residuals).

$$(b - y_1)^2 + (b - y_2)^2 + (b - y_3)^2 + (b - y_4)^2 + (b - y_5)^2 + (b - y_6)^2 + (b - y_7)^2 + (b - y_8)^2 + (b - y_9)^2$$



= 24.62

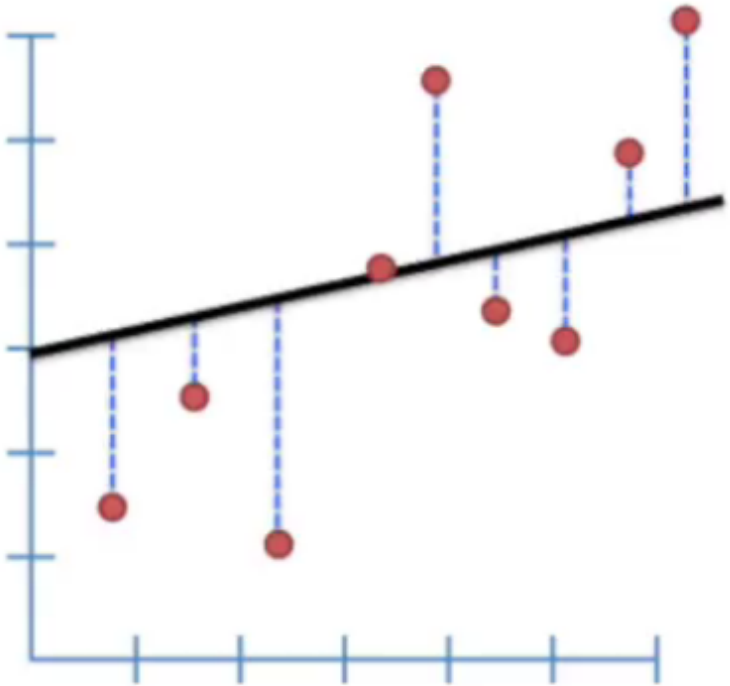
This is our measure of how well this line fits the data.

It's called the "sum of squared residuals, because the residuals are the differences between the real data and the line, and we are summing the square of these values.

(Ref: StatQuest: Linear Models - Josh Starmer)

Sum of Squared Errors for Rotated Line

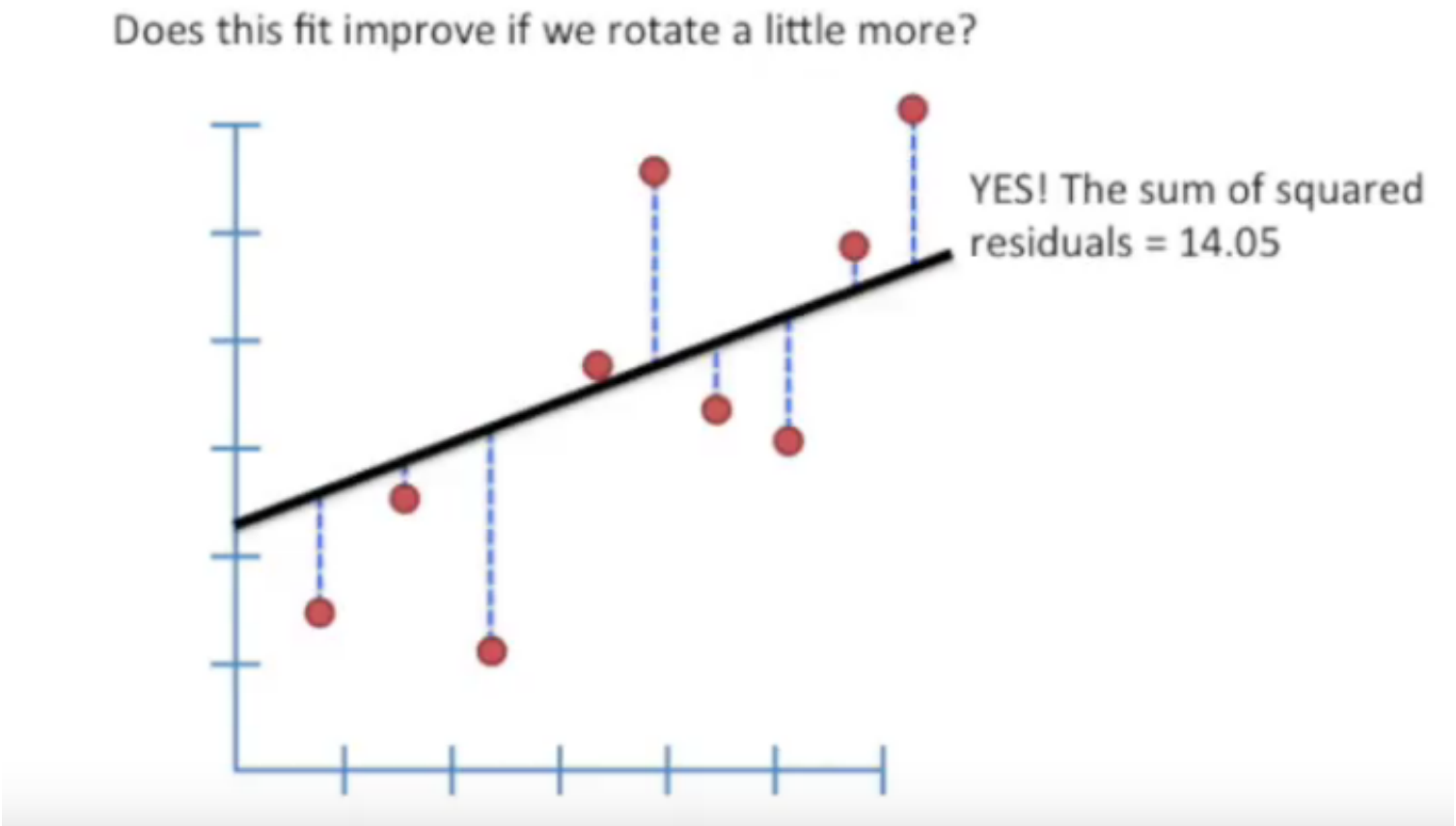
What happens if we slightly rotate the line?



The rotated line is better, but is it the best?
(Ref: StatQuest: Linear Models - Josh Starmer)

Sum of Squared Errors for Rotated Line

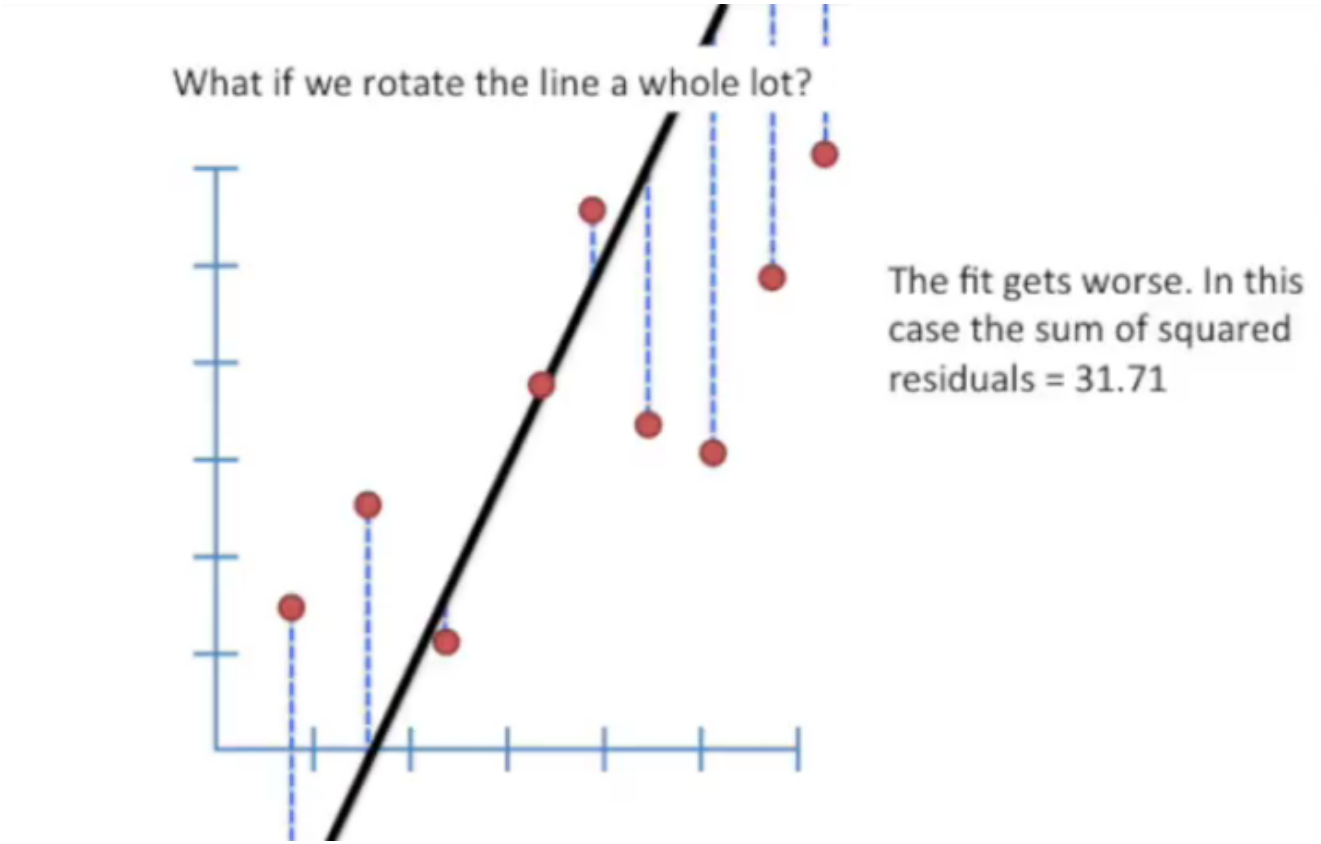
We rotate the line a bit more. Is it better?



It's better again, but is this the best?
(Ref: StatQuest: Linear Models - Josh Starmer)

Sum of Squared Errors for Rotated Line

If we rotate the line a lot, what happens?

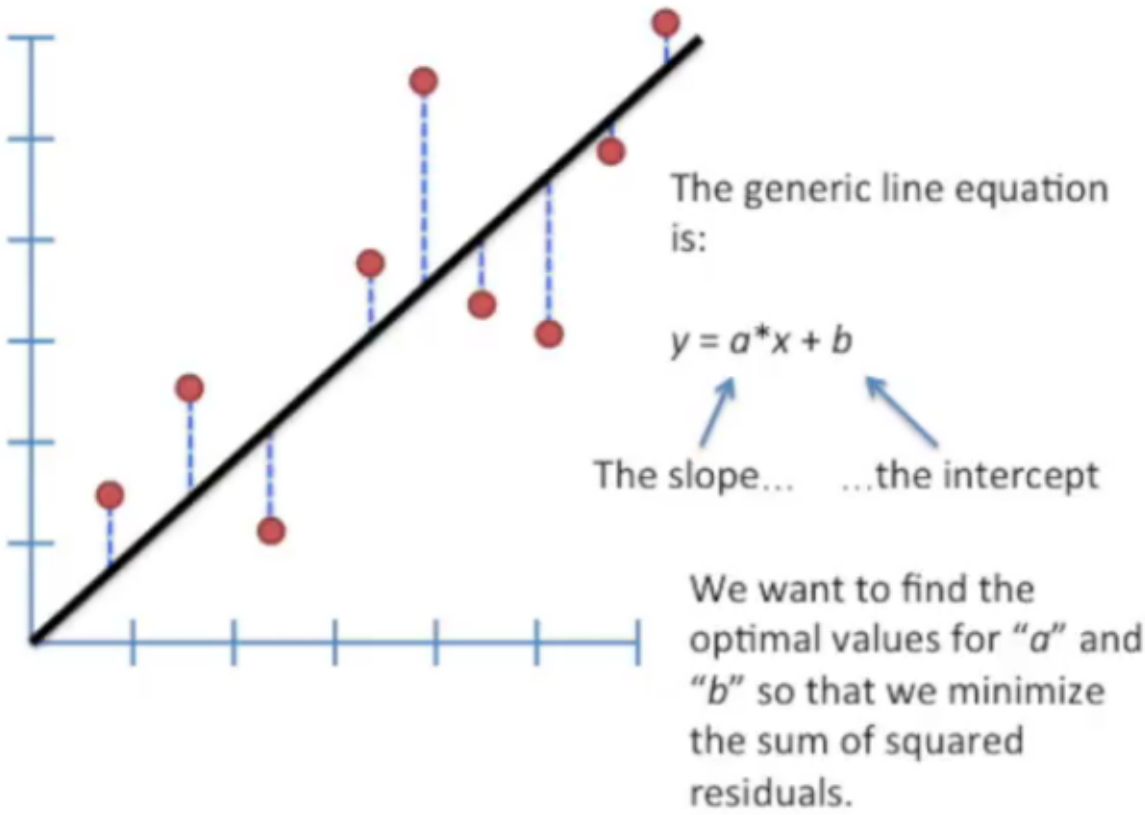


Rotating a lot is not good!

(Ref: StatQuest: Linear Models - Josh Starmer)

Optimization

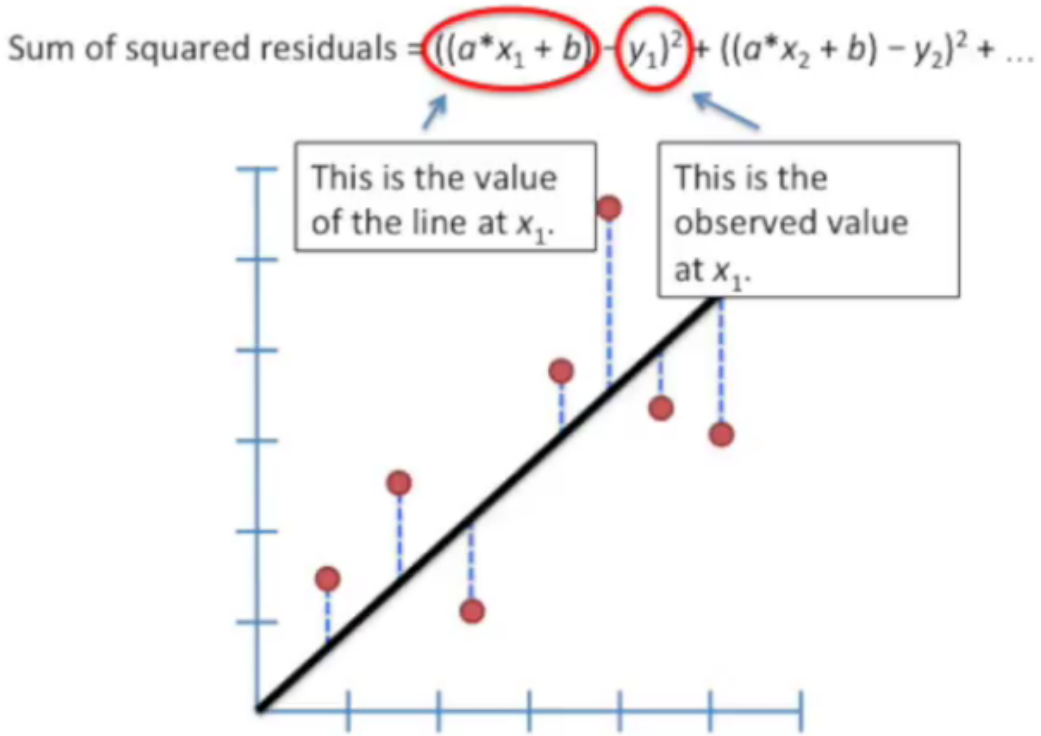
To find the best line, we consider an objective function.



(Ref: StatQuest: Linear Models - Josh Starmer)

Optimization

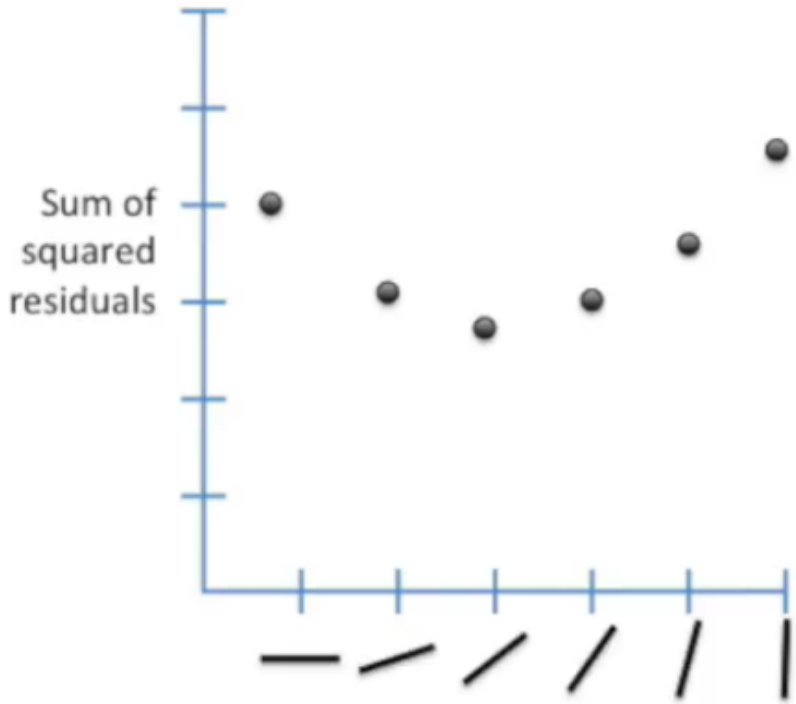
Obviously, the objective function is the sum of squared errors (SSE)!



(Ref: StatQuest: Linear Models - Josh Starmer)

Optimization

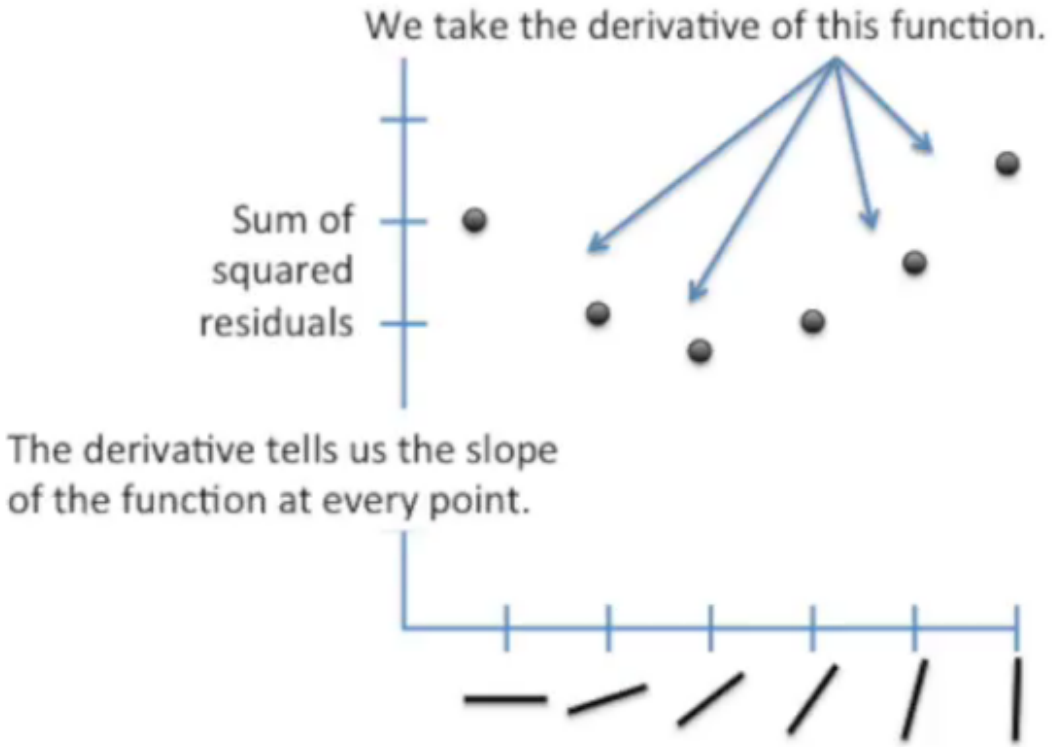
- If we plot the SSE for each rotated line, then it first decreases and then increases.
- We have to find the best line which corresponds to the minimum of the SSE.



(Ref: StatQuest: Linear Models - Josh Starmer)

Optimization

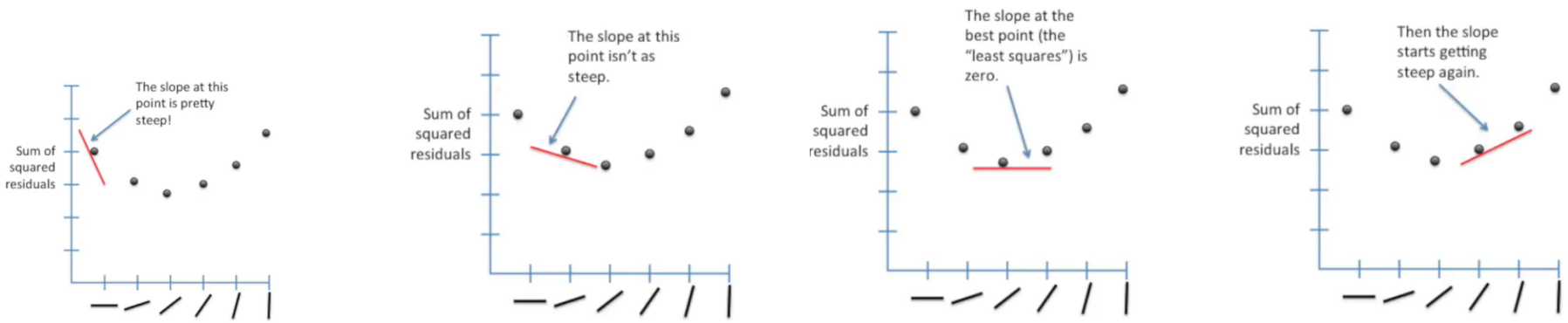
- We take the derivative of the SSE and it tells whether we change the line further or not.



(Ref: StatQuest: Linear Models - Josh Starmer)

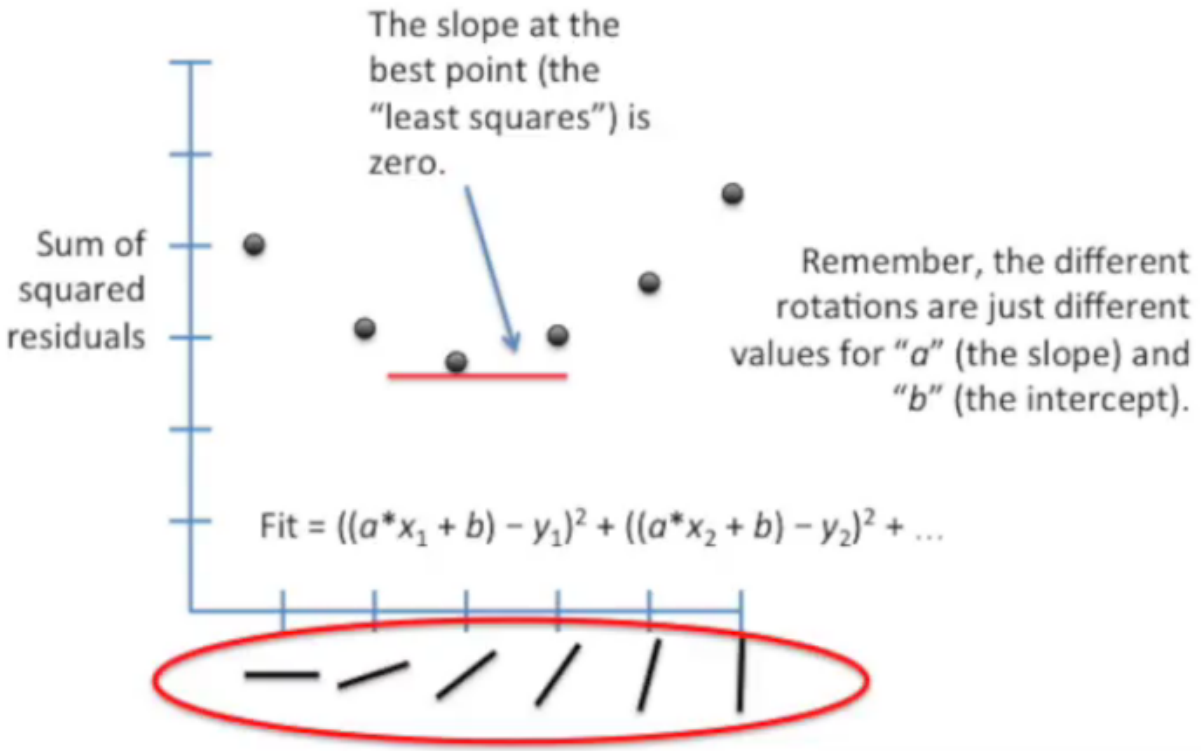
Optimization

- We take the derivative of the SSE and it tells whether we change the line further or not.



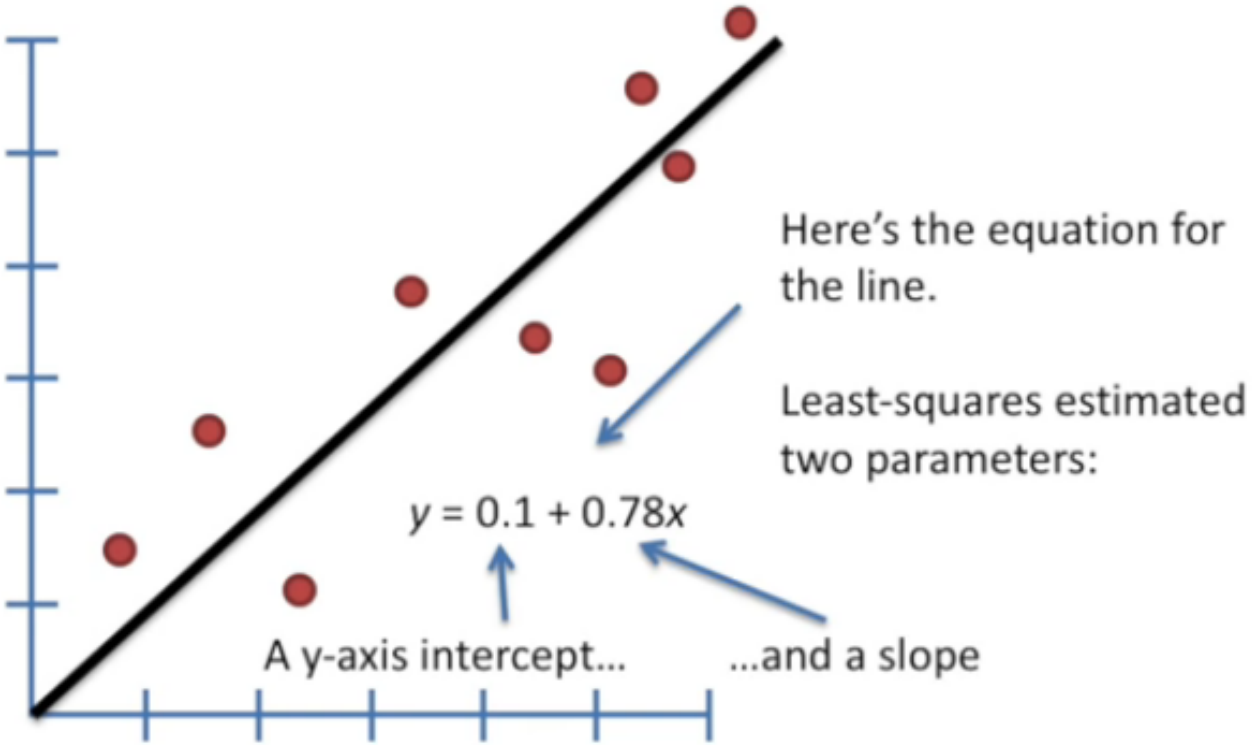
(Ref: StatQuest: Linear Models - Josh Starmer)

Optimization



(Ref: StatQuest: Linear Models - Josh Starmer)

Results



(Ref: StatQuest: Linear Models - Josh Starmer)

Our observation naturally leads to the Gradient Descent Algorithm!

Machine Learning Approach: Gradient Descent Algorithm

Recall that the objective function is

$$f(\boldsymbol{\beta}) := \sum_{i=1}^{n}(y_i - (\beta_0 + \sum_{j=1}^{k}\beta_j x_{ij}))^2.$$

The Gradient Descent Algorithm starts with an initial vector $\boldsymbol{\beta}$ and compute

$$\beta_l \leftarrow \beta_l - \alpha\frac{\partial}{\partial\beta_l}f(\boldsymbol{\beta})$$

where α is the *learning rate* and for $x_{i0} = 1, i = 1, 2, \cdots, n$,

$$\frac{\partial}{\partial\beta_l}f(\boldsymbol{\beta}) \quad = \quad -2\sum_{i=1}^{n}(y_i - (\beta_0 + \sum_{j=1}^{k}\beta_j x_{ij}))x_{il},\ 0 \leq l \leq k.$$

Gradient Descent Algorithm:

Repeat until convergence $\{$

$$\beta_l \leftarrow \beta_l + \alpha \sum_{i=1}^{n} (y_i - (\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij}))x_{il},\ 0 \leq l \leq k$$

$\}$

- This is a very natural algorithm that repeatedly takes a step in the direction of the steepest decrease of $f(\boldsymbol{\beta})$.
- However, the gradient descent algorithm has to scan through the entire training set before taking a single step.

Stochastic Gradient Descent Algorithm:

Loop {

for $i = 1$ to n, (randomly shuffle the indexes)

$$\beta_l \leftarrow \beta_l + \alpha(y_i - (\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij}))x_{il},\ 0 \leq l \leq k$$

}

- The stochastic descent algorithm can make progress with each data sample, so it often has faster convergence than the gradient descent algorithm.
- Each data sample is randomly selected from which the name *stochastic* comes.

  c.f. Mini-batch Gradient Decent Algorithm uses a mini-batch of small size.

Example:

We generate 200 samples from $y = 2x + 3 + \mathcal{N}(0, 0.5^2)$. The following figure shows the predicted line of linear regression with LSE.

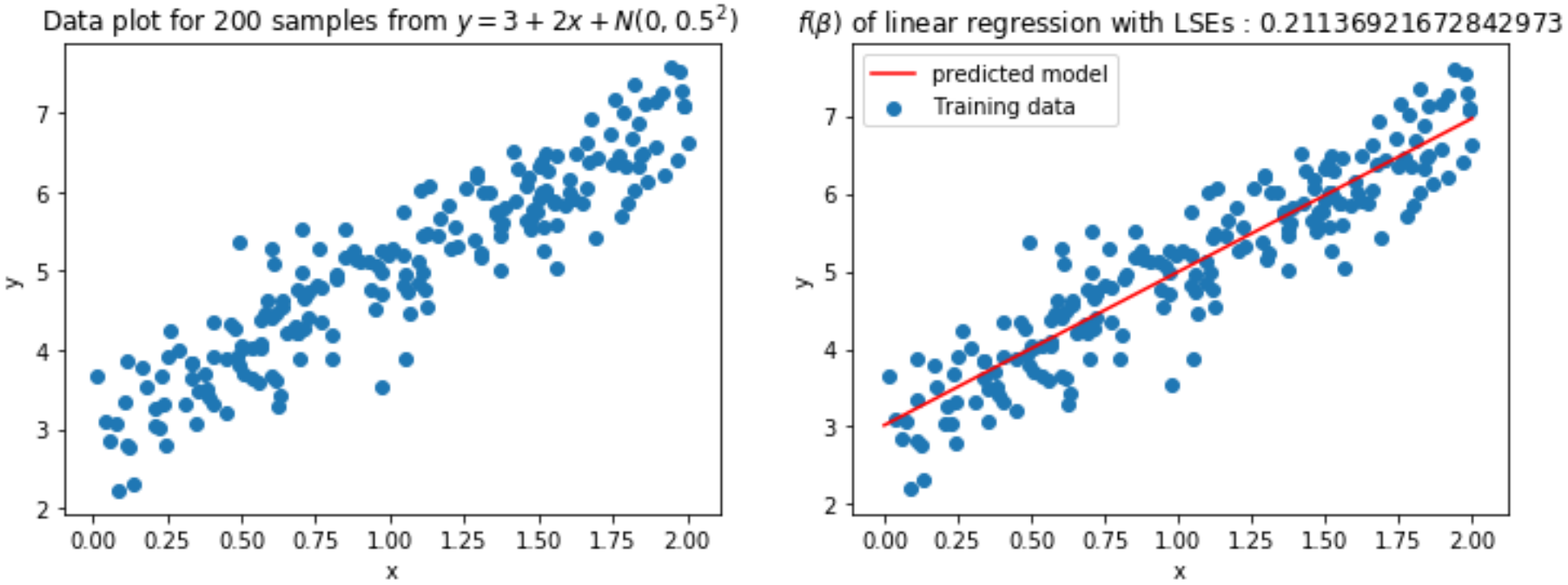


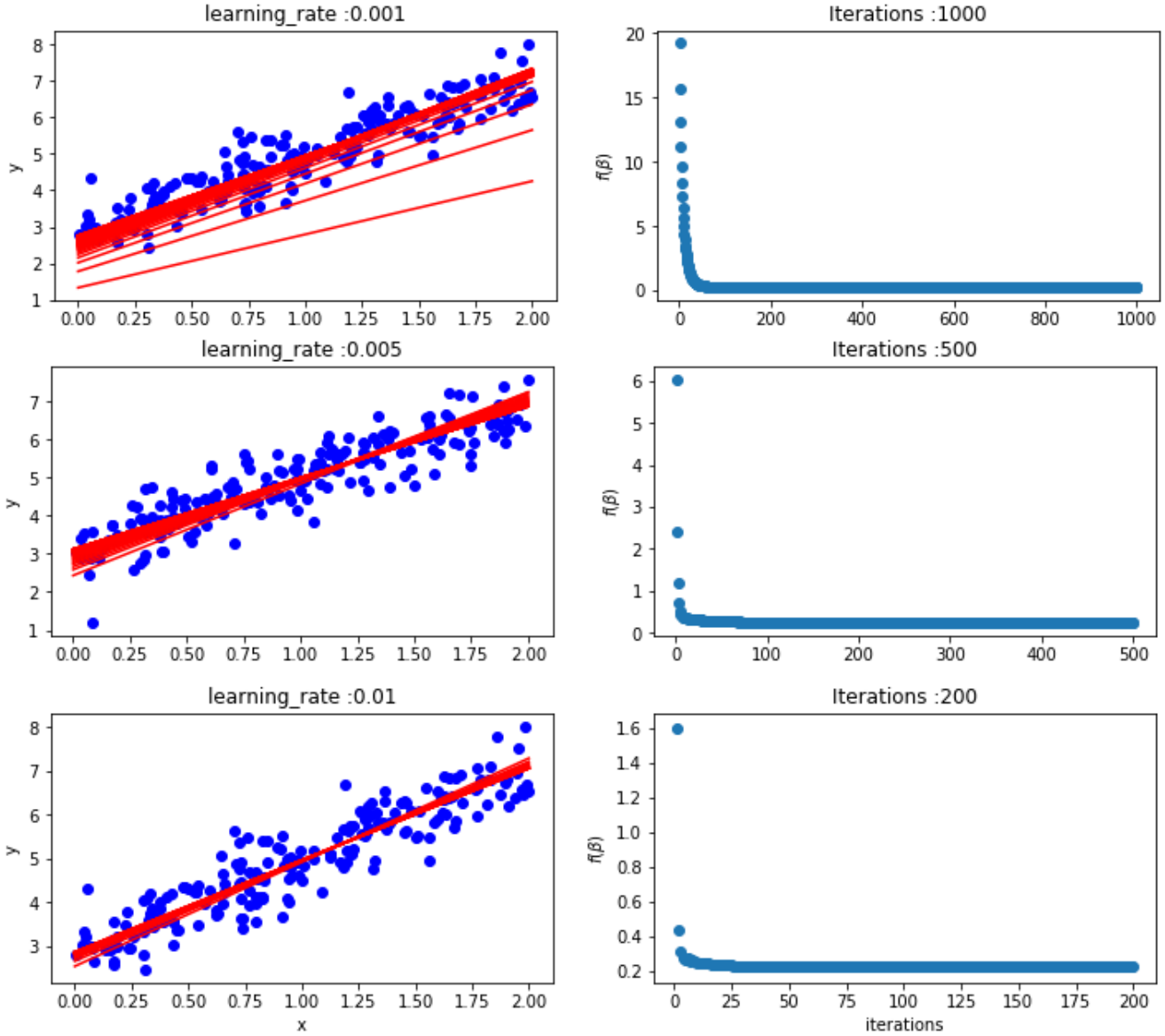
Figure: 200 samples and the predicted line

We can also use another objective function as follows:

$$f(\boldsymbol{\beta}) := \frac{1}{n}\sum_{i=1}^{n}\left(y_i - (\beta_0 + \sum_{j=1}^{k}\beta_j x_{ij})\right)^2$$

which is the mean squared error between the predicted linear model and sampled data.

The impact of learning rate and error behavior of SGD



Comparison between GD and SGD

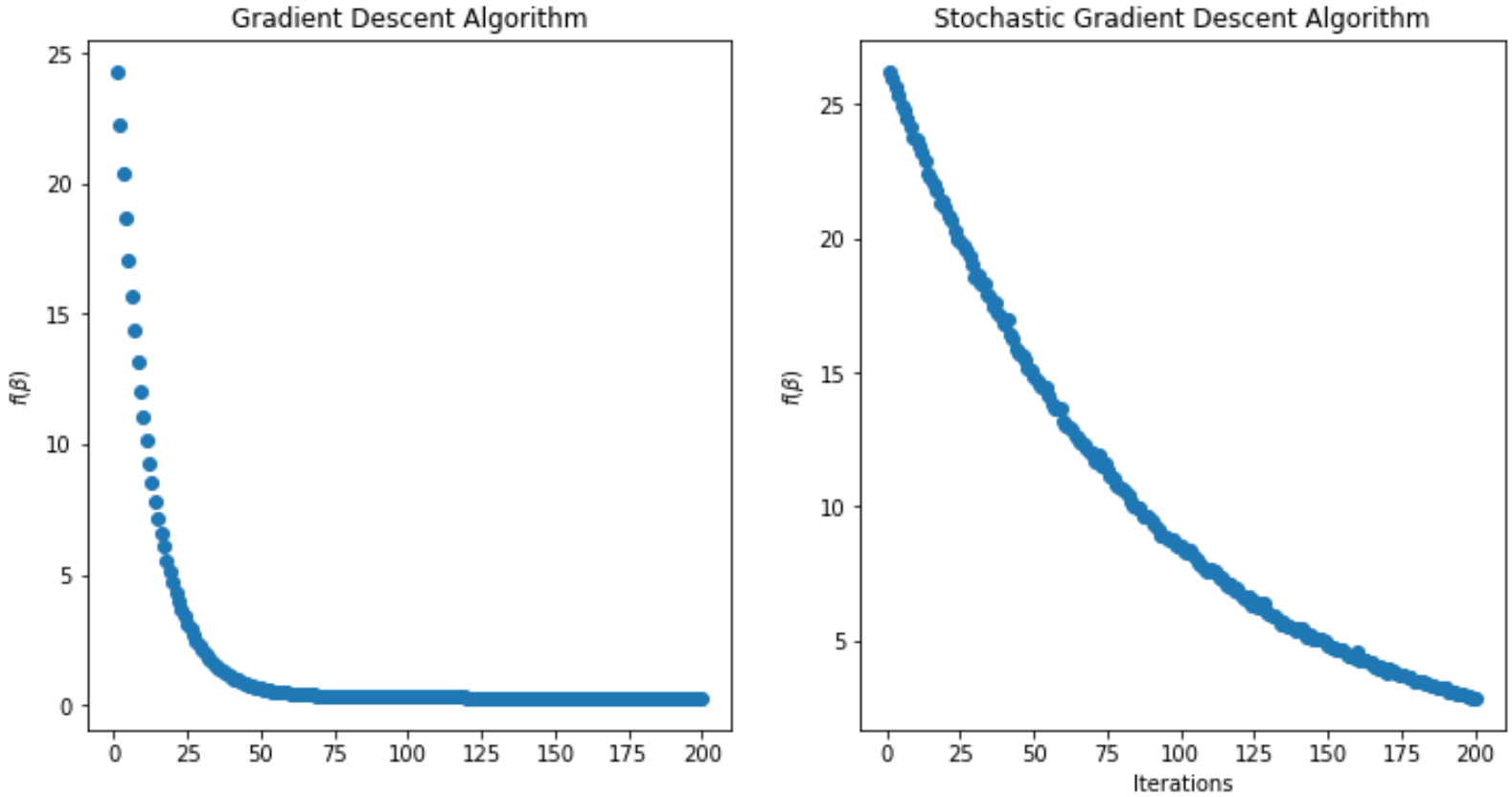


Figure: Learning rate $= 0.01$, mini-batch size $= 20$

Linear Regression - Probabilistic Approach

Probabilistic Approach

The response of an experiment can be predicted more adequately on the basis of a collection of input variables

$$\begin{aligned} Y &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k + \epsilon \\ &= \beta_0 + \sum_{j=1}^{k} \beta_j x_j + \epsilon \end{aligned}$$

where

- Y: dependent variable
- x_i: a set of input (independent) variables
- $\epsilon \sim \mathcal{N}(0, \sigma^2)$: a normal error with unknown σ^2

Let Y_i be the response corresponding to the set $\{x_{i1}, x_{i2}, \cdots, x_{ik}\}$.

We now consider the likelihood function of data sample $(y_1, y_2, \cdots, y_n)$

$$p(y_i, 1 \leq i \leq n | x_i, 1 \leq i \leq n, \beta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_i - h_i(\boldsymbol{\beta}))^2}{2\sigma^2}}$$

where $h_i(\boldsymbol{\beta}) = \beta_0 + \sum_{j=1}^{k} \beta_j x_{ij}$.

The log likelihood function is given by

$$\log p(y_i, 1 \leq i \leq n | x_i, 1 \leq i \leq n, \beta) = -n \log(\sqrt{2\pi}\sigma) - \sum_{i=1}^{n} \frac{(y_i - h_i(\boldsymbol{\beta}))^2}{2\sigma^2}.$$

To maximize the log likelihood function, it is sufficient to minimize

$$\sum_{i=1}^{n} (y_i - h_i(\boldsymbol{\beta}))^2$$

which is identical to $f(\boldsymbol{\beta})$.

From this point of view, the Least Square Estimators (LSEs) are also called the Maximum Likelihood Estimators (MLEs).

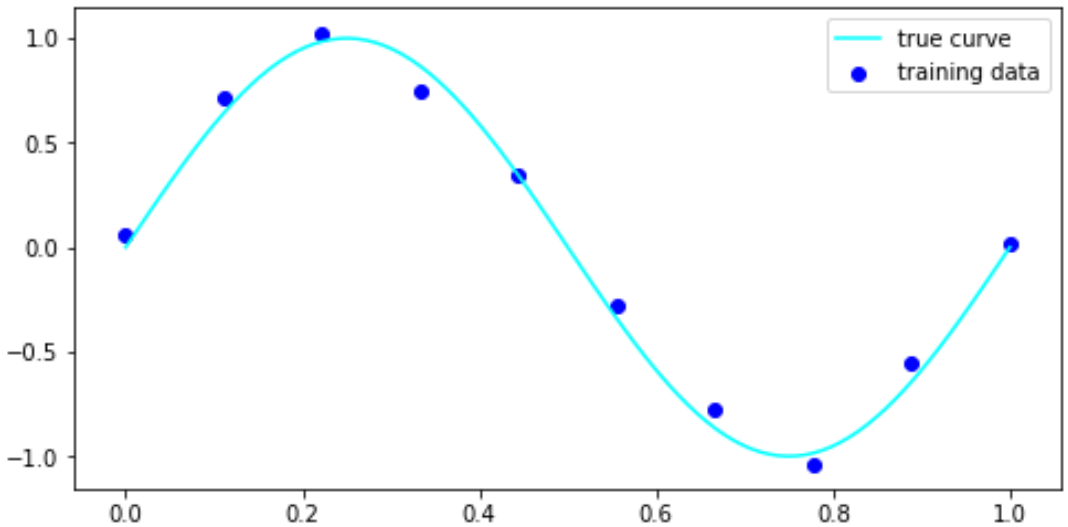
Linear Regression with Regularization

Regularization

Reference: Christopher M. Bishop, Pattern Recognition and Machine Learning, Springer 2006.

We have a data sample $\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$ and want to find a polynomial of degree M that fits the data sample well.

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_M x^M.$$



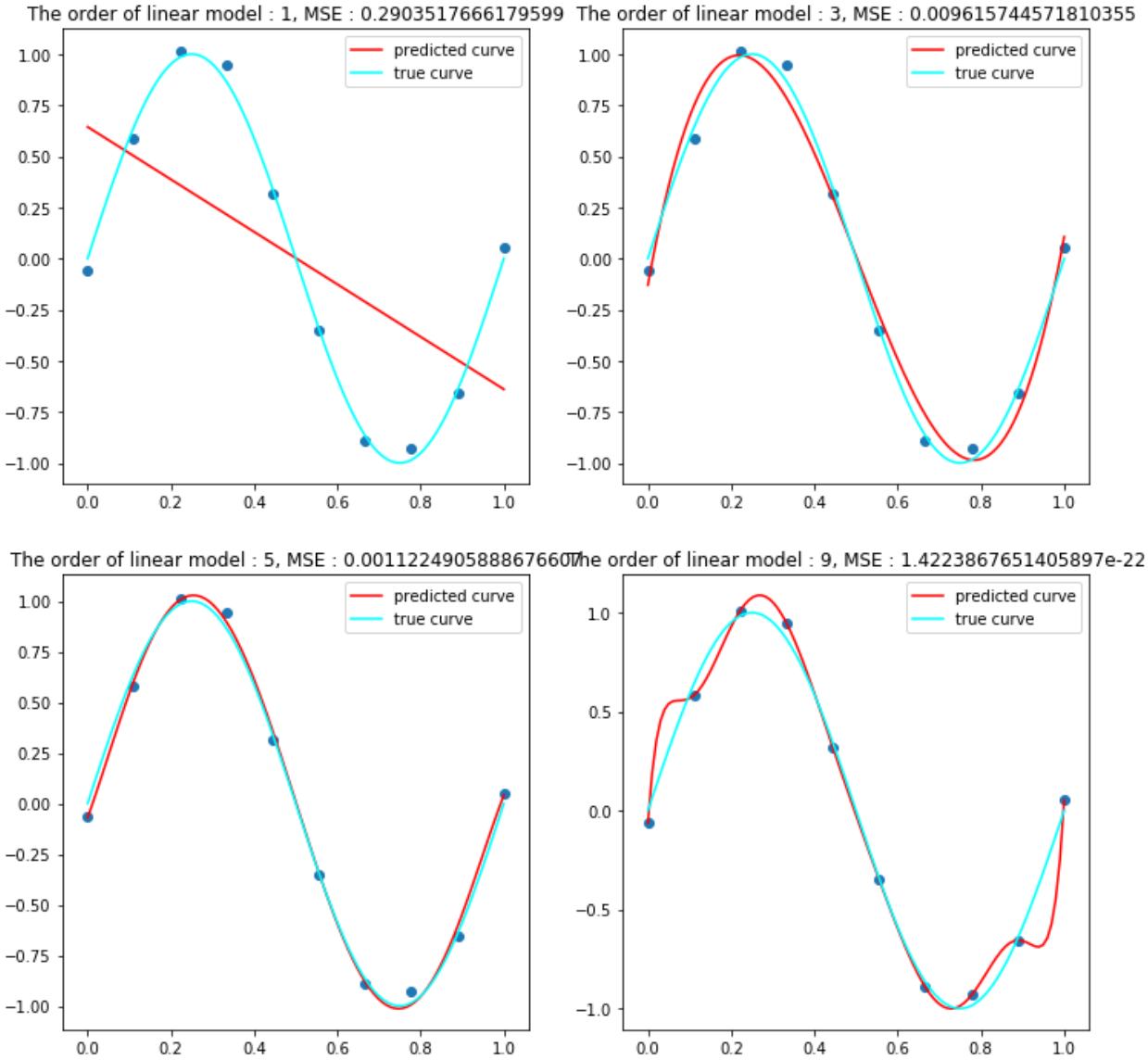
It is just a simple linear regression with the objective function

$$f(\boldsymbol{\beta}) := \|\mathbf{y} - X\boldsymbol{\beta}\|^2,$$

whose features are $\{1, x, x^2, \cdots, x^M\}$.

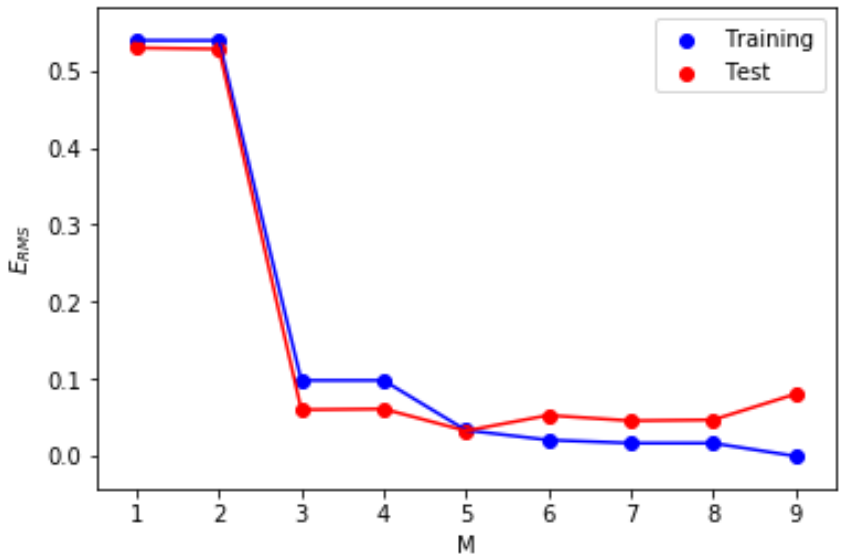
The following figures show the results of the polynomial regression for $M = 1, 3, 5, 9$.

Among $M = 1, 3, 5, 9$, the value of MSE $(= \frac{1}{n} f(\boldsymbol{\beta}))$ with respect to training data is minimized when $M = 9$. This is overfitting.


The order of linear model : 1, MSE : 0.2903517666179599
The order of linear model : 3, MSE : 0.009615744571810355
The order of linear model : 5, MSE : 0.0011224905888676607
The order of linear model : 9, MSE : 1.4223867651405897e-22
predicted curve
true curve

- We want to have good performace for a new data set. So generalization is important.
- We measure generalization using a separate data set (validation set) with Root Mean Squared (RMS) error

$$E_{RMS} = \sqrt{\frac{f(\boldsymbol{\beta})}{n}}$$

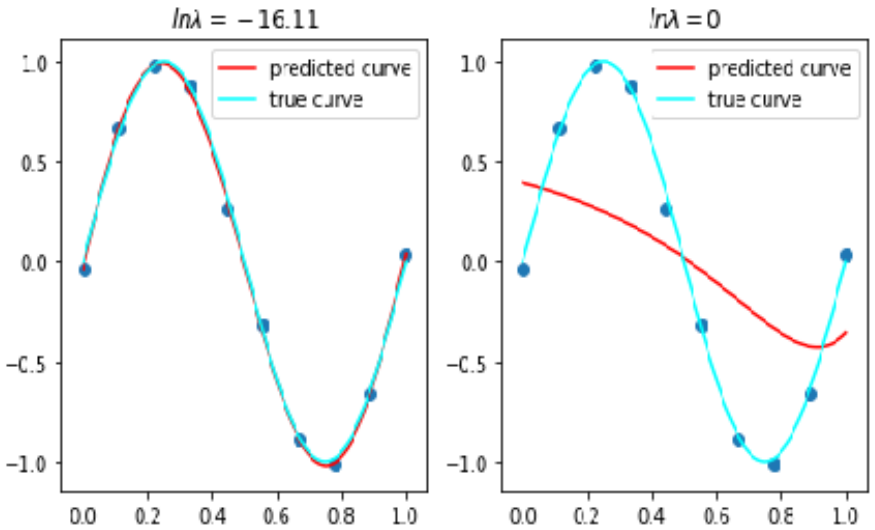


The difference between training curve and test (validation) curve increases as M increases for $M \geq 5$.

- One way to control the over-fitting is the regularization technique.
- We penalize large coefficients in the error function as

$$J[f] = \frac{1}{2} f(\boldsymbol{\beta}) + \frac{\lambda}{2} ||\boldsymbol{\beta}||^2.$$

- We now apply this to our polynomial regression with $M = 9$.



	$\ln\lambda = -36.8$	$\ln\lambda = -16.1$	$\ln\lambda = 0$
β_0	0.0595	0.4318	0.0518
β_1	-27.5030	8.5115	-0.4515
β_2	769.8333	-19.6029	-0.4321
β_3	-6789.6970	-9.8102	-0.2839
β_4	30751.1791	48.0982	-0.1423
β_5	-81239.8240	-29.9223	-0.0291
β_6	130036.20	-36.5120	0.0575
β_7	-12413.7379	29.0027	0.1229
β_8	65022.7529	58.4866	0.1726
β_9	-1437.6592	-48.2849	0.2105

- From this table, we can check that as $\lambda \to 0$, i.e, $\ln\lambda \to -\infty$, we have the over-fitting problem.

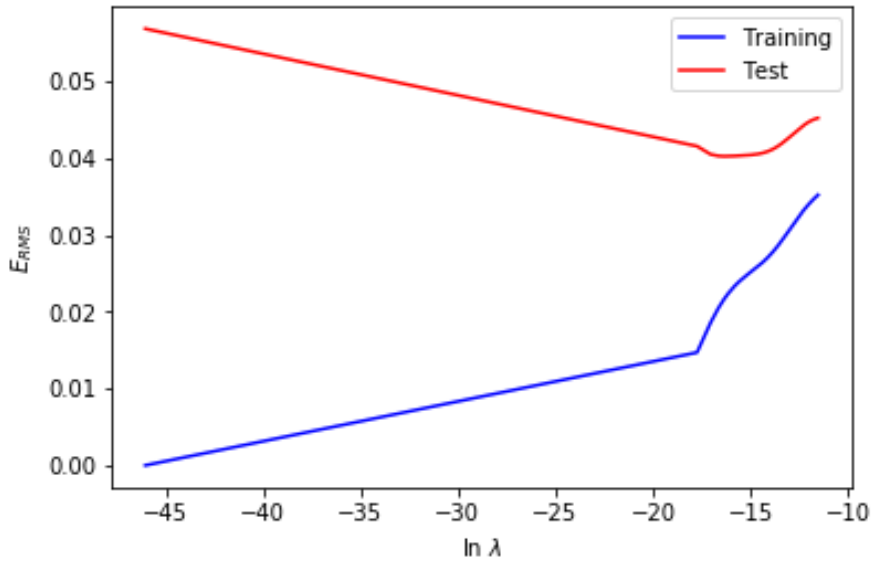
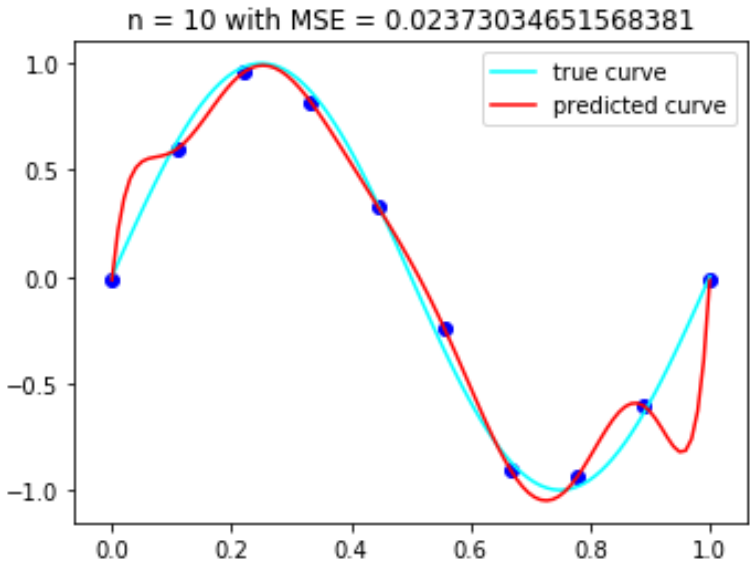
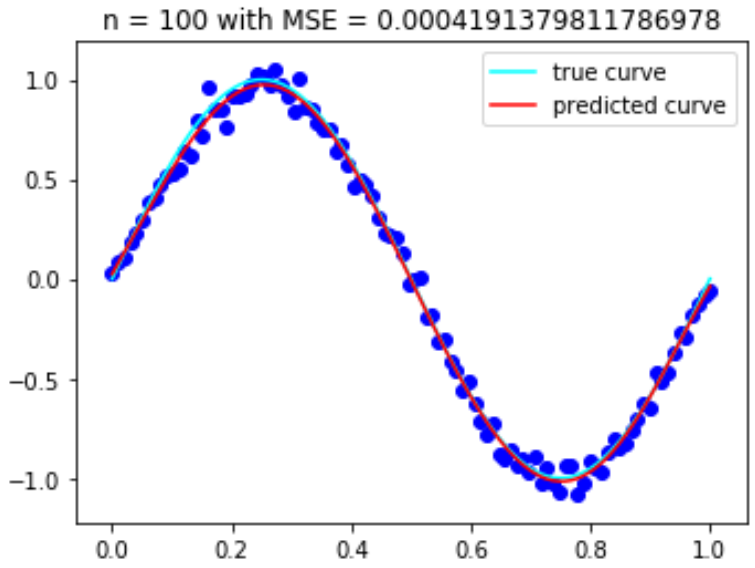


Figure: The RMS curves

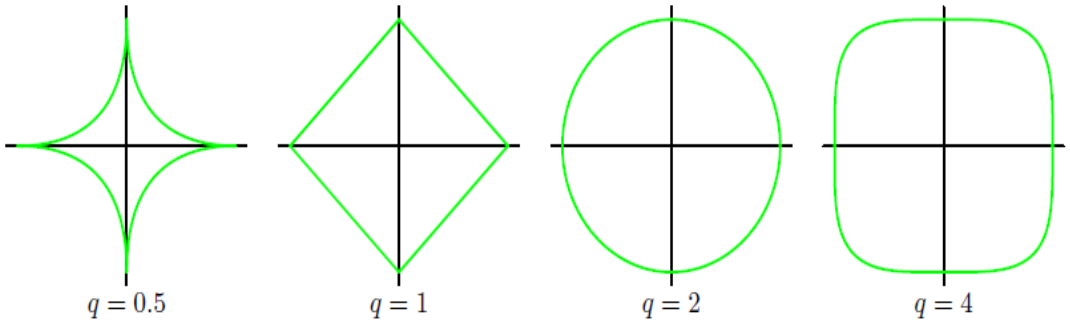
- With regularization the test (with validation set) and training errors show similar trend.

- On the other hand, the more data is given, the more complex model (i.e, higher M) can predict well.
- The following figures show a comparison between the cases when $n = 10$ and $n = 100$ for $M = 9$ and $\lambda = 0$.
- Rule of thumb: 10 data points for each parameter





Other Regularizers



We consider a number of regularizers

$$J[f] = \frac{1}{2} f(\boldsymbol{\beta}) + \frac{\lambda}{2} \sum_{j=0}^{M} |\beta_j|^q.$$

- $q = 1$: LASSO (Least Absolute Shrinkage and Selection Operator) regression
- $q = 2$: Ridge regression (aka Parameter Shrinkage Regression)

Analysis of Ridge Regression

For the objective function

$$\begin{aligned} J[f] &= \frac{1}{2} f(\boldsymbol{\beta}) + \frac{\lambda}{2} \sum_{j=0}^{M} |\beta_j|^2 \\ &= \frac{1}{2} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \frac{\lambda}{2} \boldsymbol{\beta}^\top \boldsymbol{\beta}, \end{aligned}$$

by letting

$$\begin{aligned} \nabla_{\boldsymbol{\beta}} J[f] &= \mathbf{X}^\top \mathbf{X}\boldsymbol{\beta} - \mathbf{X}^\top \mathbf{y} + \lambda \boldsymbol{\beta} \\ &= 0, \end{aligned}$$

we obtain

$$\hat{\boldsymbol{\beta}} = (\lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}.$$