Lecture 04: Regression

[SCS4049-02] Machine Learning and Data Science

Seongsik Park (s.park@dgu.edu)

AI Department, Dongguk University

Linear regression

Linear regression

Linear model

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \tag{1}$$

In this equation,

- $\hat{y}$ is the predicted value (for true y)
- n is the number of features
- x_i is the i-th feature value
- θ_j is the j-th model parameter (including the bias term θ_0 and the feature weights $\theta_1, \theta_2, \cdots, \theta_n$)

Linear regression

This can be written much more concisely using a vectorized form,

$$\hat{} = \quad _0 + \quad _1 \, ' \, _1 + \quad _2 \, ' \, _2 + \qquad + \quad _{} \, ' \, _{} \tag{2}$$

$$= \quad _{} \; (^{2}) = \qquad ^{2} \tag{3}$$

In this equation,

- is the model's [illegible] , containing the bias term $_0$ and the feature weights $_1$ to $_{}$
- ^{2} is the instance's [illegible] , containing $' _0$ to $' _{}$ always equal to 1
- ^{2} is the dot product of the vectors and ^{2}, which is of course equal to $_0 \, ' \, _0 + \quad _1 \, ' \, _1 + \qquad + \quad _{} \, ' \, _{}$
- $_{}$ is the hypothesis function, using the model parameter

Linear regression

That's the linear regression model – but how do we train it?

Recall that training a model means setting its parameters so that the model best fits the training set.

We first need a measure of how well (or poorly) the model fits the training data.

The most common performance measure of a regression model is the Root Mean Square Error (RMSE).

$$\mathrm{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h(\mathbf{x}^{(i)}) - y^{(i)}\right)^2} \tag{4}$$

Linear regression

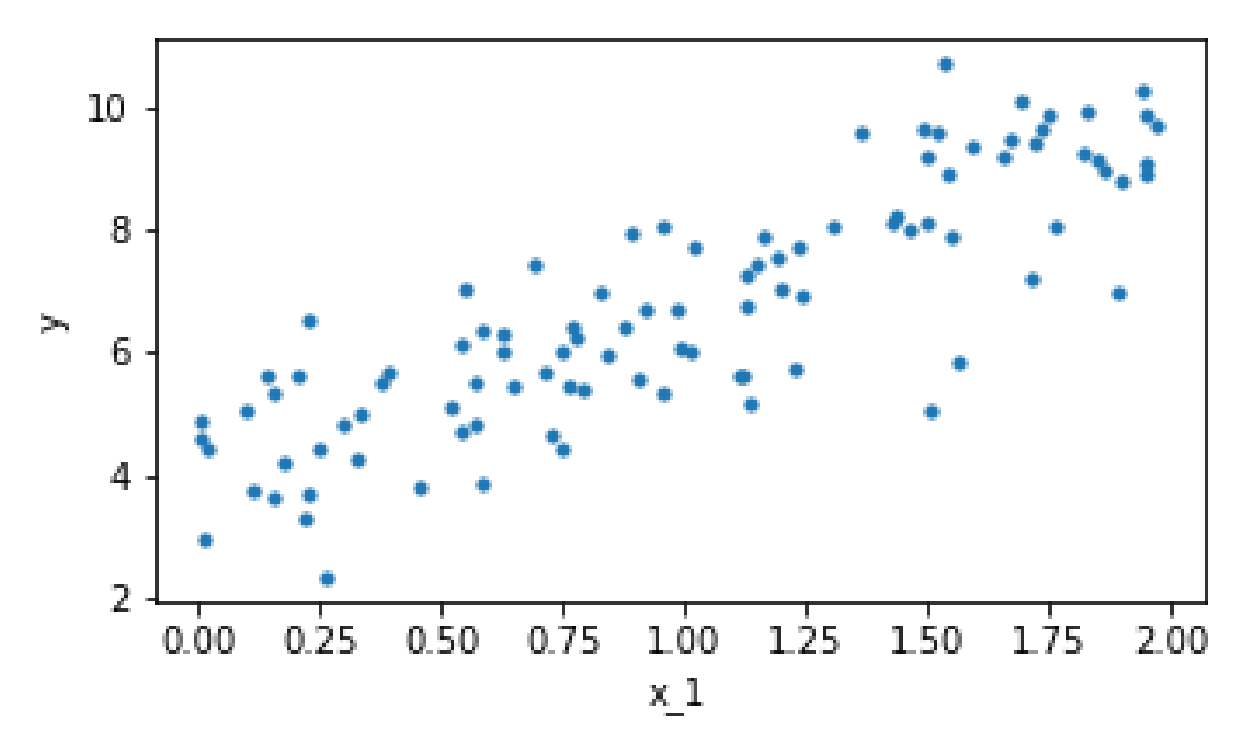


Figure 1: Linear regression: training dataset

Generating training dataset

$$\text{‘} \quad {}_0 + {}_1\text{’} \tag{5}$$

$$\text{‘} = {}_0 + {}_1\text{’} + \quad \text{where} \quad N(0; \ {}^2) \tag{6}$$

Linear regression: design matrix

Design matrix (regressor matrix, model matrix, data matrix)

- ﺌﮩ Ȁ ḫ ᙖ ɭ (sample, example) ᙖ m ƙ
- feature vector ² u ﻭٖ ⁴ ᙖ n ᒪ ẋ *
- m ƙ u sample row vector ̏E ﻨﺰ design matrix ± ̏E ﻳﺞ ` ﻨﻢ ̈ב Կ

$$
\acute{} = \begin{bmatrix} \text{ˀ}^{(1)} \\ \text{ˀ}^{(2)} \\ \vdots \\ \text{ˀ}^{(g)} \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ \vdots \\ \text{ђ} \end{bmatrix} \tag{7}
$$

$$
\pm = \begin{bmatrix} \text{1st sample } \sim^{(1)} \\ \text{2nd sample } \sim^{(2)} \\ \\ \text{m-th sample } \sim^{(g)} \end{bmatrix} = \begin{bmatrix} \sim_1^{(1)} & \sim_2^{(1)} & & \sim_{\text{ђ}}^{(1)} \\ \sim_1^{(2)} & \sim_2^{(2)} & & \sim_{\text{ђ}}^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ \sim_1^{(g)} & \sim_2^{(g)} & & \sim_{\text{ђ}}^{(g)} \end{bmatrix} \tag{8}
$$

Linear regression: design matrix

Ẏ ẇ* ^ = İ₂ u m ƙ u sample Ꮭ ḏ يم ą Ö Ʀ Ŀ نمصحيج ב Ч

$$\hat{y} = X\theta \tag{9}$$

$$\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(m)} & x_2^{(m)} & \cdots & x_n^{(m)} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \tag{10}$$

$$\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} = \begin{bmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \\ \vdots \\ \mathbf{x}^{(m)} \end{bmatrix} \tag{11}$$

Design matrix ± u Ꝑ ꟿ ẫ₂ Ớ ə ﻤﻤ ڽْ

$$\hat{y} = \pm = \begin{bmatrix} \tilde{x}_1 & \tilde{x}_2 & \cdots & \tilde{x}_p \end{bmatrix} \begin{bmatrix} {}_1 \\ {}_2 \\ \vdots \\ {}_p \end{bmatrix} \quad (12)$$

$$= {}_1\tilde{x}_1 + {}_2\tilde{x}_2 + \cdots + {}_p\tilde{x}_p \quad (13)$$

* ^ ʊ $\tilde{x}_1; \tilde{x}_2; \ldots; \tilde{x}_p$ ᗜ ◕ ᵻ ﻤﻤ ʊ hyperplane ◄ ꟿ ʟ ᑎ ﻨﯽ

Residual Ǎ ʊ error u ﺻ $\|\ \|$ ^k Ǖ ﭺ ﻜﺨ ﻤﻤ Ä ڽْ= u hyperplane Ö error ^ Ѣ ♭ Ё ב (orthogonal) ﻴﻢ ꟿ ﻨﯽ

Residual Ǎ ʊ error u ڝ, *k*´ ^*k* ȕ ڿ ݣ ࢫ ʺA ẇ= ˜ u hyperplane Ö error ´ ^ ƀ ƅ ̏E ב ̗ (orthogonal) ࢬ ˧ ݭ

̗ * ẇ ṭ column vector ˧ ḏ ࢬ ƅ

$$ꝍ(´ \; ^) = 0 \tag{14}$$

Ⴆ Ⱡ Ὼ ࢬ ˧ ݭ

ꞑ ̗ m ƙ u sample ˧ ḏ ࢬ ƅ

$$\pm^{i}(´ \; \pm \;) = 0 \quad Q) \quad \pm^{i} \pm \; = \pm^{i} ´ \tag{15}$$

$$^ = (\pm^{i} \pm)^{1} \pm^{i} ´ \tag{16}$$

Normal equation: geometric interpretation

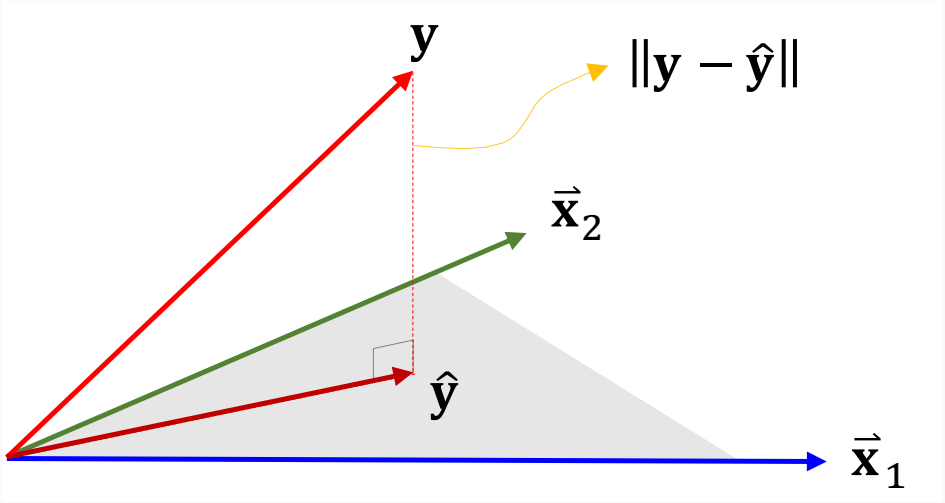


Figure 2: Normal equation: geometric interpretation

Normal equation

$$\wedge = (\pm^{\dot{I}} \pm)^{\ 1} \pm^{\dot{I}} \text{,} \tag{17}$$

Projection matrix

$$\wedge = \pm \wedge = \pm(\pm^{\dot{I}} \pm)^{\ 1} \pm^{\dot{I}} \text{,} \tag{18}$$

Normal equation: analytic approach

Sum of squared error (SSE)

$$\sim\sim \mathrm{N} = \sum_{\dot{0}=1} (\text{‘}^{(\dot{0})} \quad \text{ʔ}^{(\dot{0})})^2 \tag{19}$$

$$= k\acute{} \quad \wedge k^2 = (\acute{} \quad \wedge)^{\dot{I}}(\acute{} \quad \wedge) \tag{20}$$

Using $\wedge = \pm$

$$\sim\sim \mathrm{N}(\ ) = \acute{}^{\dot{I}}\acute{} \quad 2\acute{}^{\dot{I}}\pm \ + (\pm \ )^{\dot{I}}(\pm \ ) = \acute{}^{\dot{I}}\acute{} \quad 2\acute{}^{\dot{I}}\pm \ + \ ^{\dot{I}}\pm^{\dot{I}}\pm \tag{21}$$

$$\frac{@\sim\sim \mathrm{N}(\ )}{@} = 2(\pm^{\dot{I}}\pm \quad \pm\acute{}) = 0 \quad \mathrm{Q}) \quad \pm^{\dot{I}}\pm \ = \pm^{\dot{I}}\acute{} \tag{22}$$

Hence, we have

$$\wedge = (\pm^{\dot{I}}\pm)^{\ 1}\pm^{\dot{I}}\acute{} \tag{23}$$

$$\wedge = \pm(\pm^{\dot{I}}\pm)^{\ 1}\pm^{\dot{I}}\acute{} \tag{24}$$

Linear regression: cost function

We need to find the value of that minimizes the RMSE. In practice, it is simpler to minimize the sum of squared error (SSE) than the MSE or the RMSE.

$$\hat{} = 9\}\mathrm{X}\ \mathrm{h\_j}\ \mid \mathrm{g} \sim \mathrm{N}(\pm; \dot{q}_{\cdot}) = 9\}\mathrm{X}\ \mathrm{h\_j}\ \sqrt{\frac{1}{g}\sum_{\ddot{\emptyset}=1}^{\mathbb{X}} \dot{I}_{2}(\ddot{\emptyset}) \; \text{‘}(\ddot{\emptyset})^{2}} \tag{25}$$

$$= 9\}\mathrm{X}\ \mathrm{h\_j}\ \mathrm{g} \sim \mathrm{N}(\pm; \dot{q}_{\cdot}) = 9\}\mathrm{X}\ \mathrm{h\_j}\ \frac{1}{g}\sum_{\ddot{\emptyset}=1}^{\mathbb{X}} \dot{I}_{2}(\ddot{\emptyset}) \; \text{‘}(\ddot{\emptyset})^{2} \tag{26}$$

$$= 9\}\mathrm{X}\ \mathrm{h\_j} \sim\sim \mathrm{N}(\pm; \dot{q}_{\cdot}) = 9\}\mathrm{X}\ \mathrm{h\_j}\ \sum_{\ddot{\emptyset}=1}^{\mathbb{X}} \dot{I}_{2}(\ddot{\emptyset}) \; \text{‘}(\ddot{\emptyset})^{2} \tag{27}$$

Linear regression: closed from solution

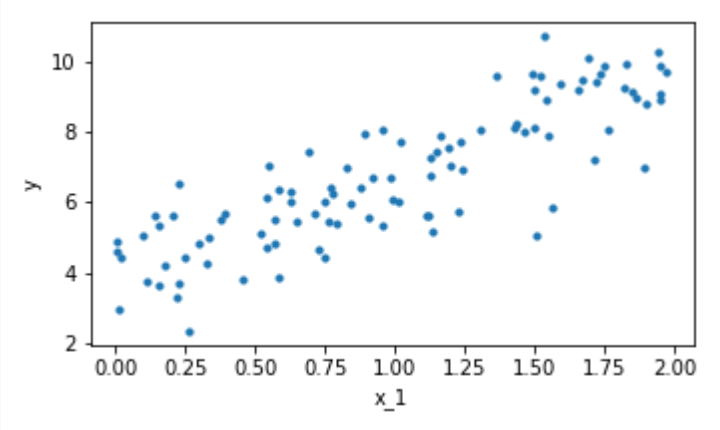


Figure 3: Linear regression: training dataset

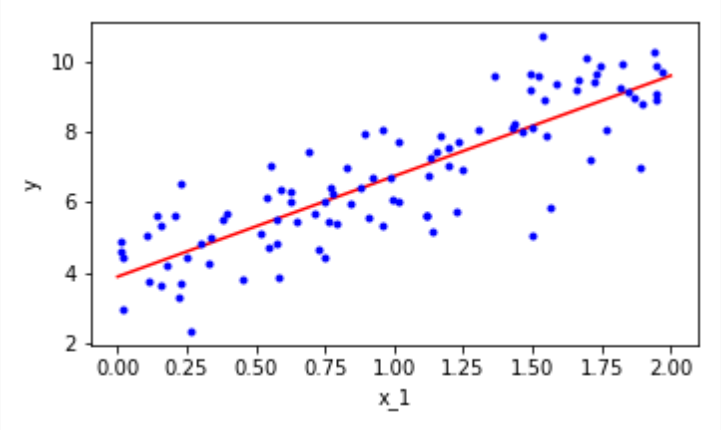


Figure 4: Linear regression: closed form solution

Linear regression: computational complexity

Normal equation u Æ▓

- Normal equation ɻ u ﺘﺮ ɦ ݣ ݳ ʌ́

$$\acute{\wedge} = \pm(\pm^{\dot{I}}\pm)^{1}\pm^{\dot{I}}\,\acute{} \qquad (28)$$

- $\pm\ 2\ R^{g\quad \hbar}$
- $\pm^{\dot{I}}\pm\ 2\ R^{\hbar\quad \hbar}$
- $(\pm^{\dot{I}}\pm)^{1}$ u Æ▓ꞁ Ω l̠ $= \ddot{e}(\hbar^{2:4}) \quad \ddot{e}(\hbar^{3})$
- Feature ꟾ u ꟷ ł h Й ݳ̤̈ E Æ▓ ` ꞓ Ƅ ˜ ƅ

Gradient descent

Gradient descent

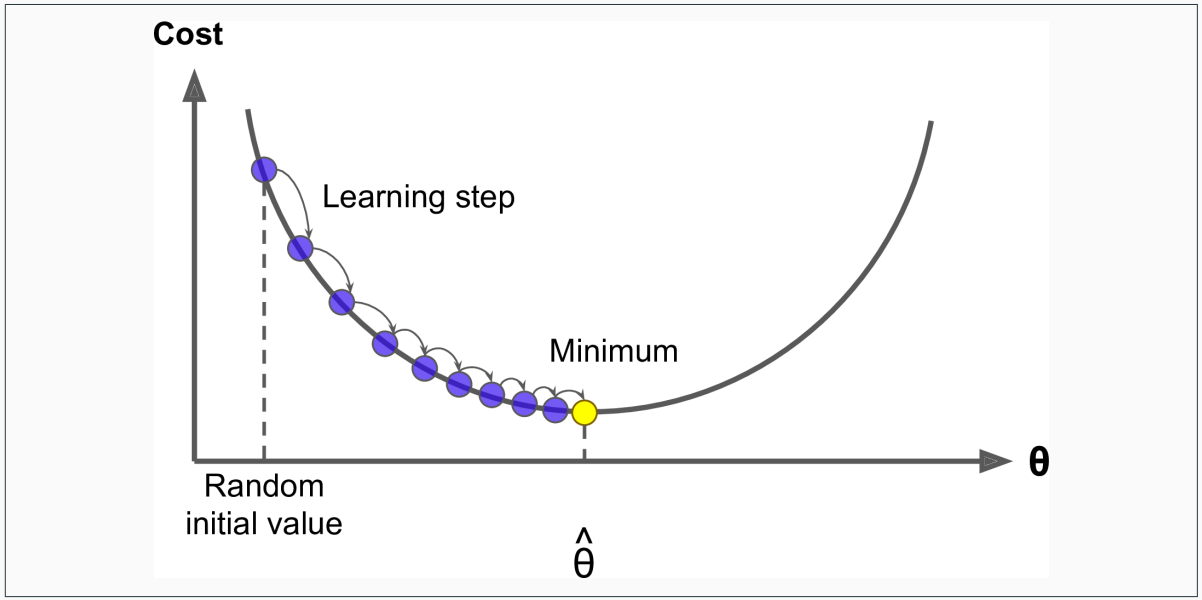


Figure 4-3. Gradient Descent

Gradient descent

Derivative

- ’ u ″ ݣ ⅃ d̰ ڗ ĭ(’) u Ŷ Ҕ !
- ڗ ɦ ’ ⅃ ƅ u ¸ ǘ¸
- ĭ : scalar ! scalar

$$f'(\theta) = \frac{d f(\theta)}{d\theta} \tag{29}$$

Gradient

- Derivative ǚ ą ″ ב ڼב ̏E Ը ̀Ωݣ
- ڗ ɦ x ⅃ ƅ u ڿ d̰ ¸ ǘ¸
- ĭ : vector ! scalar

$$\nabla f(\theta) = \frac{\partial f(\theta)}{\partial \theta} = \begin{bmatrix} \frac{\partial f}{\partial \theta_1} \\ \frac{\partial f}{\partial \theta_2} \\ \vdots \\ \frac{\partial f}{\partial \theta_n} \end{bmatrix} \tag{30}$$

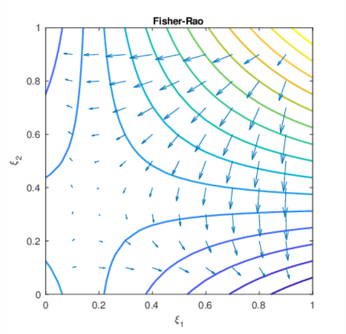


Figure 5: Gradient field

Gradient descent

Find the value of θ where $J(\theta)$ is minimum (local or global)

- Find the value θ at which $J'(\theta) = 0$

$$\frac{\partial J(\theta)}{\partial \theta} = 0 \tag{31}$$

- Sure it's minimum?

$$\frac{\partial^2 J(\theta)}{\partial \theta^2} > 0 \tag{32}$$

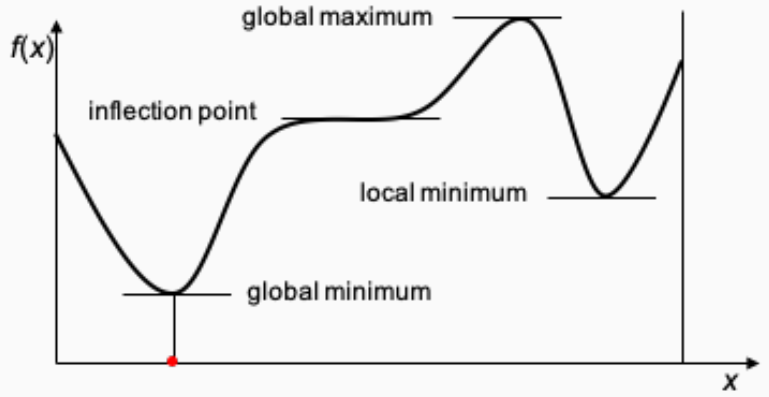


Figure 6: Local and global minimum

Gradient descent: optimization method

- [illegible] (cost function, loss function)
  - [illegible] (ground truth) [illegible]
  - [illegible] utility, [illegible]
  - e.g., sum of squared error (SSE), cross-entropy [illegible]
- [illegible] (learning algorithm)
  - [illegible] (training data) [illegible]
- [illegible]
  - [illegible] normal equation [illegible] close-form solution [illegible]
  - [illegible]

Gradient descent: method

ꓒ h ȣ ᷍ = ʾ ꓷ d̯ يم ƅ SSE ȕ̃ cost ̏E ▌ ȉ́ مم ʊ Ä ȕ̏

$$\mathring{A}(\cdot) = \cdot = \sum_{\mathring{i}=1}^{g} (\cdot^{(\mathring{i})} \cdot^{(\mathring{i})})^2 \tag{33}$$

$$\mathring{A}'(\cdot) = 2 \sum_{\mathring{i}=1}^{g} \left(\cdot^{(\mathring{i})} \cdot^{(\mathring{i})}\right) \cdot^{(\mathring{i})} \tag{34}$$

L u u ḩ ꓷ ƅ $\mathring{A}'(\cdot)$ Ҭ ꭗ ḩ ꓷ ƅ $\mathring{A}(\cdot)$ ƅ ƅ Ч ↑ ȕ̆ I Ƃ مم ʊ بحÖ
صِ ȕ̃ ᵁ ع ᵖ

$$\cdot^1 = \cdot^0 \quad \mathring{A}'(\cdot^0) \tag{35}$$

$$\cdot^2 = \cdot^1 \quad \mathring{A}'(\cdot^1) \tag{36}$$

$$\vdots \qquad \vdots \tag{37}$$

$j\mathring{A}(\cdot) \quad \mathring{A}(\cdot^{-1})j$ (ꭗ ȍ ȕ̏ Ư ꭗ ב) ƅ ȍ̈ ẋ ̔ ῼ مم ɹ چ Ŀ يم u ▌Ҭ
ძنی

ɹ ƅ ʊ نر ̃ ̏O (learning rate) ꭗ update e ϧ ɲ ɘ مم ʊ hyperparameter (قٶב) u Ը :

Gradient descent: learning rate

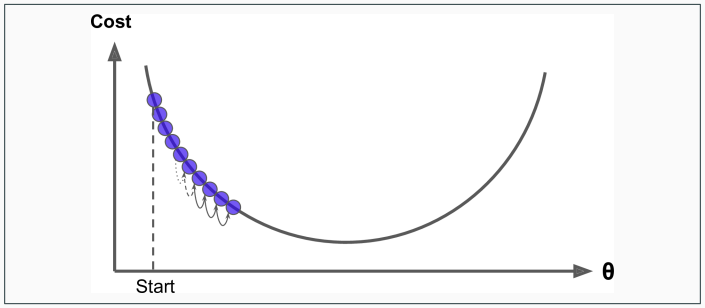


Figure 4-4. Learning rate too small

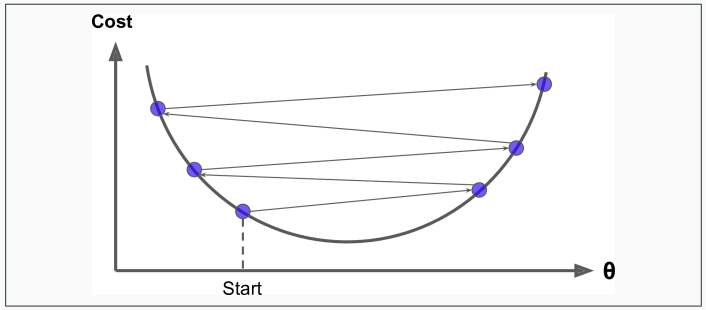


Figure 4-5. Learning rate too large

- (iteration)
- (iteration)
- Gradient vector (norm) (tolerance)
- convex batch gradient descent
- tolerance 1/10
- 10

Batch gradient descent

Linear regression model $\wedge = \pm$

$$Å(\ ) = \sim\sim \mathrm{N}(\ ) = k' \quad \pm \quad k^2 \tag{38}$$

$$r \quad Å(\ ) = \frac{@}{@ \ 2} Å(\ ) = 2 \pm \dot{I}^{(} \pm \quad ') \tag{39}$$

$$r \quad Å(\ ) = \begin{matrix} 2 \\ 6 \\ 6 \\ 6 \\ 4 \end{matrix} \begin{matrix} \frac{@}{@ \ 1} Å(\ ) \\ \frac{@}{@ \ 2} Å(\ ) \\ \vdots \\ \frac{@}{@ \ ŋ} Å(\ ) \end{matrix} \begin{matrix} 3 \\ 7 \\ 7 \\ 7 \\ 5 \end{matrix} = \begin{matrix} 2 \\ 6 \\ 6 \\ 6 \\ 4 \end{matrix} \begin{matrix} 2 \sum\limits_{ð=1}^{g} , (ð) \ \dot{I}_{\mathbf{2}}(ð) \ ' (ð) \\ 2 \sum\limits_{ð=1}^{g} , (ð) \ \dot{I}_{\mathbf{2}}(ð) \ ' (ð) \\ \vdots \\ 2 \sum\limits_{ð=1}^{g} , (ð) \ \dot{I}_{\mathbf{2}}(ð) \ ' (ð) \end{matrix} \begin{matrix} 3 \\ 7 \\ 7 \\ 7 \\ 5 \end{matrix} \tag{40}$$

Gradient descent step

$$^{-}+1 = \quad ^{-} \quad r \quad Å(\ ^{-}) \tag{41}$$

where iteration number $^{-}$ and arbitrary initial value

Batch gradient descent

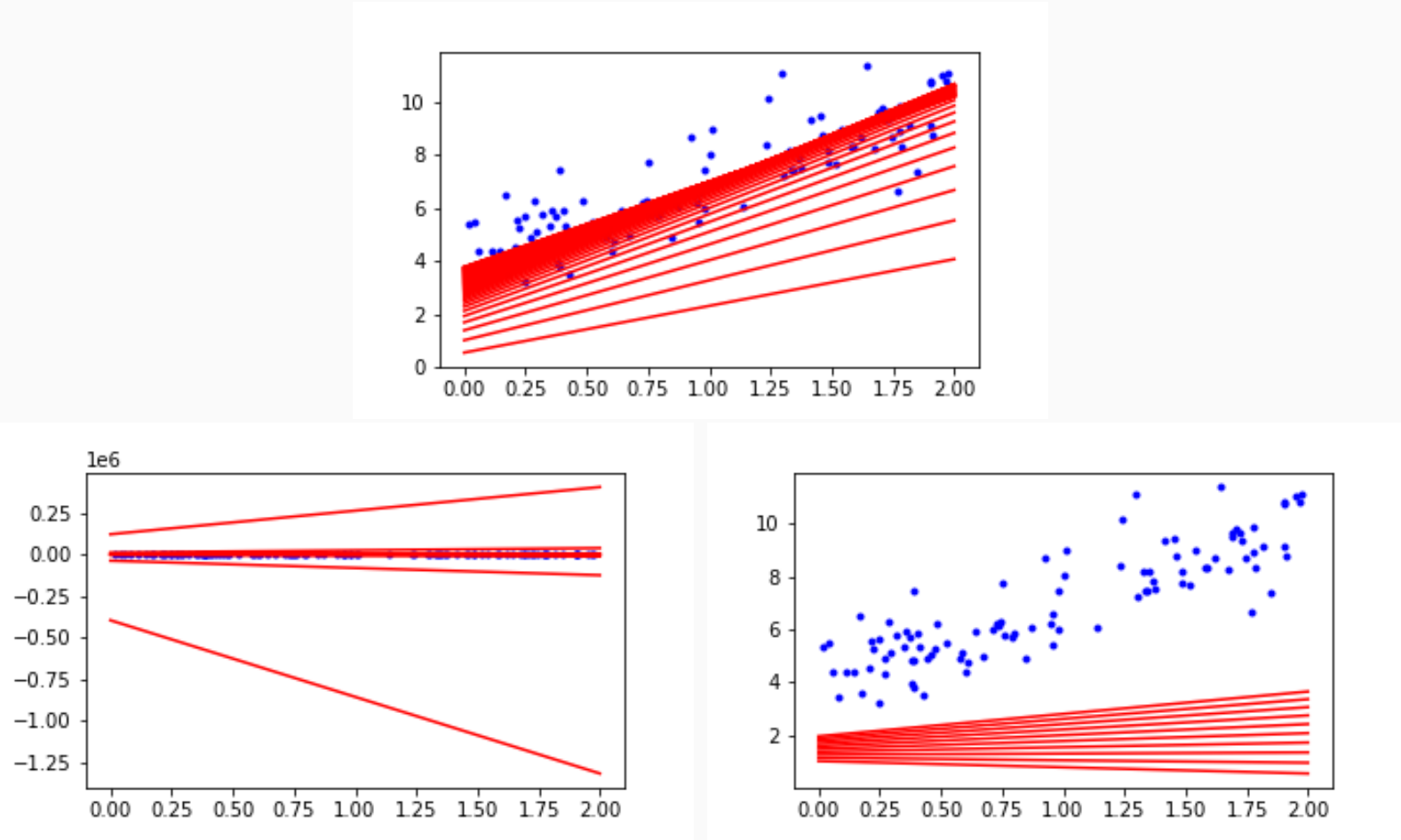


Figure 7: Learning rate = 0.001, 0.01, 0.0001

Batch gradient descent algorithm

- ὺ ʻ ى῭Ωą batch ῃ ز ⅎ ḓ نز Æ▓ كيὓ
- ḫ ㅌ ɻ ɧ ㅌ ۦ ˤ ῳ ˉ ḻ ƅ ˈ ΆƳ
- Normal equation: feature ב ⅎ ẁỚ Æ▓ˉ ḻ ƅ ˤ ב Ŀ ẓ῭Ε ˈ ΆƳ
- Gradient descent: feature ב ƅ ᵘ ᛇ ḻ ص ا ″ مٴ ᛌ
- Learning rate
- Hyperparameter Ŀ نز ˜ ΌΟ(learning rate) ƅ ᵡ ˜ Ʊ א Ä ὓ ˋ ꝑ ㅌ ⅎ Ủ ц ῶ
- ᵡ ˜ ض Ä ὓ چ Ŀ يم ὓ ˤ ᵁ ڗ يم ὓ ۴ ˤ ῼ نم ב Ч

Learning schedule

- Constant learning rate
  - 0.1, 0.01
- Time-based decay

$$= \frac{0}{(1 + j^{-})} \tag{42}$$

$_0$: j : hyperparameter, $^{-}$: iteration

- Step decay
  - epoch
  - 5 epoch 20 epoch 1/10
  - Epoch: epoch
- Exponential decay

$$= {}_0\bar{a}^{\,j^{-}} \tag{43}$$

$_0$: j : hyperparameter, $^{-}$: iteration

Stochastic gradient descent

For our linear regression model ^ = İ 2

$$Å(\ ) = \sim\sim N(\ ) = \sum_{0=1} \left((0) \quad İ\, 2 (0) \right)^2 = \sum_{0=1} Å_0(\ ) \qquad (44)$$

$$^{+1} = \ ^{-}\ 2\ ( ) \quad (\ )^{İ}\, 2 ( ) \quad 2 ( ) \qquad (45)$$

- ̃ Ữ ̈E ♠ ق نز نز ƙ u sample Ɉ d̪ يم ƅ ̈Ω gradient ŭ̃ Æ▓ مم ɺ parameter ŭ̃ update
- sequential learning or online learning
- d̪˙ ῷ ḫ Ƅ Ɩ ḩ پῶ مم ʊ ḫ ̃ ῶ
- ♠ ق مم ʊ ▌ ĕ̋ u ̃ Ữ ŧ ʔ̈E Ŋ ˛ L Ƅ ə˙ ه
- BGD Ɉ e يم local optimum Ɉ ƅ ʔ I ↑ n ᵁ Ɉ ב Ч
- چ Ŀ يم Ɉ I̱ Ḇ مم ˊ ̈Ω ˊ ˉ Ŀ ʔ̈E ŭ̈ ṅ
- BGD̆ ̈Ω ى ƅ ˊ ̈E global optimum Ƅ Ớ ʊ ꞁ Ч Ƅ Ɉ

Mini-batch gradient descent

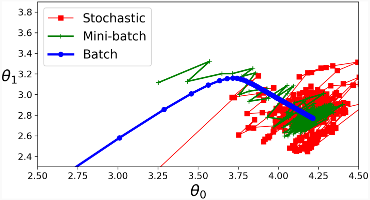


Figure 4-11. Gradient Descent paths in parameter space

- ﺌﻬ Ā ɧ ᕫ ɭ ɧ Մ ℵ ﺼ u Մ a o ٣ ﻧﻲ Ⴤ E ᵁ ð ᠰ ƅ gradient ǚ
  ᕲ ﻤﻣ ᴜ "
- ᠢ 100,000 ƙ u ɧ ᕫ ɭ = (mini-batch size 100) (1,000 mini-batches)
- Batch gradient descent stochastic gradient descent(SGD) u ᵽ ﮋ
- SGD ą ə ﻫ ﻧﺰ Ŋ L ᕫ Ḣ ﻨﯽ
- SGD ą local minimum ᠰ ƅ ↑ n ᵁ ᠢ ƅ ◀ ɖ Ŀ Ⴤ E ĝ ᠰ Ā Ŋ
- GPU ǚ . ﻧﺰ ὺ ﺠﺋ ώ ᠰ ▓ u ỊǚʆᏑ ᑳ ᑫ

Linear regression comparison

Table 4-1. Comparison of algorithms for Linear Regression

Algorithm	Large *m*	Out-of-core support	Large *n*	Hyperparams	Scaling required	Scikit-Learn
Normal Equation	Fast	No	Slow	0	No	n/a
SVD	Fast	No	Slow	0	No	`LinearRegression`
Batch GD	Slow	No	Fast	2	Yes	`SGDRegressor`
Stochastic GD	Fast	Yes	Fast	≥2	Yes	`SGDRegressor`
Mini-batch GD	Fast	Yes	Fast	≥2	Yes	`SGDRegressor`

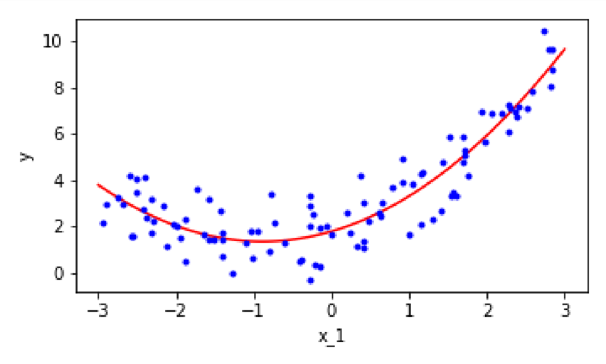
There is almost no difference after training: all these algorithms end up with very similar models and make predictions in exactly the same way.

Polynomial regression

Polynomial regression

- ḫ ᕲ ⅃ ṯ ᕲ e ♠ض ϴÆ ǘ ᕲ ȋ ə Կ ẋ
- polynomial u Ƥ ﱞ ‴ ƙ u feature‶E ꝑ د ﱞ ♠ض ῤ Ḯ Ö ˜
- ᛂ ȣ ’$_1$ ᛂ ḓ ﱞ 2 ۋ polynomial

$$\hat{} = {}_0 + {}_1 {}'_1 + {}_2 {}'^2_1 \quad (46)$$



Polynomial regression: closed form solution

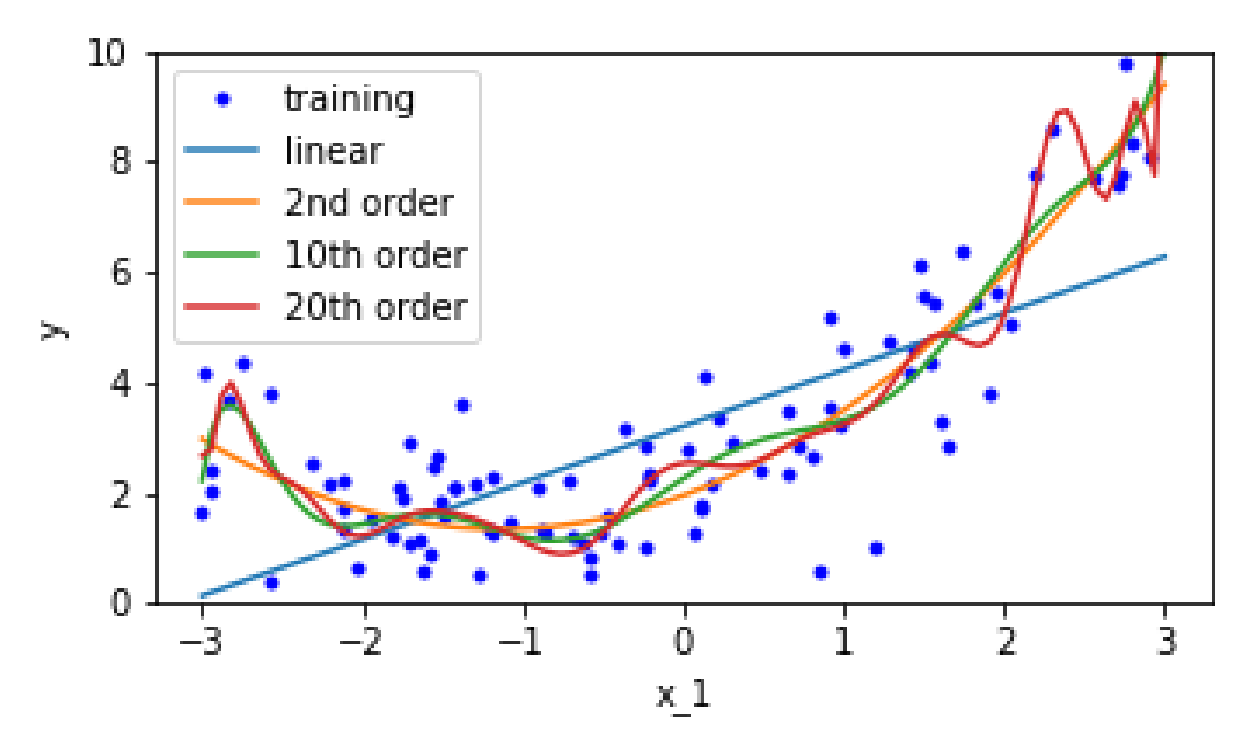


Figure 9: Polynomial regression: closed form solution

Learning curves: model selection

Learning curves

- ﻬٔ Ά ḫ Ե ꞁ ḩ u ﺻ Ǻ ʊ Ὼ ő ʊ ﻬٔ Ά u ﻬ ב ᚷ ẁ ṹ training error validation error ṹ ῴ Կ
- Ᵽ error u ◂ ḏ Ŀ Ŀ ″ ﻜ ṹ ꞁ ə ῴ Ἰ ﻠﻲ ƀ ﻨﻣ ב Կ

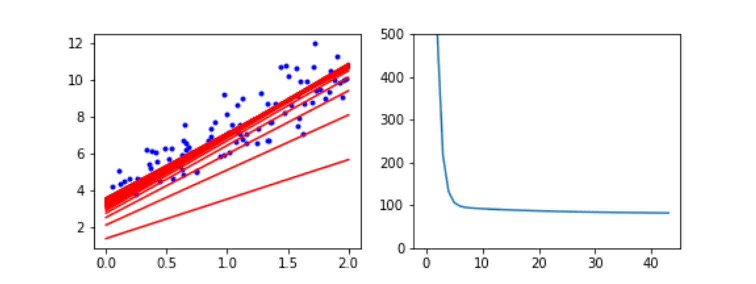


Figure 10: Linear regression with learning curve (SSE)

Learning curves: model selection

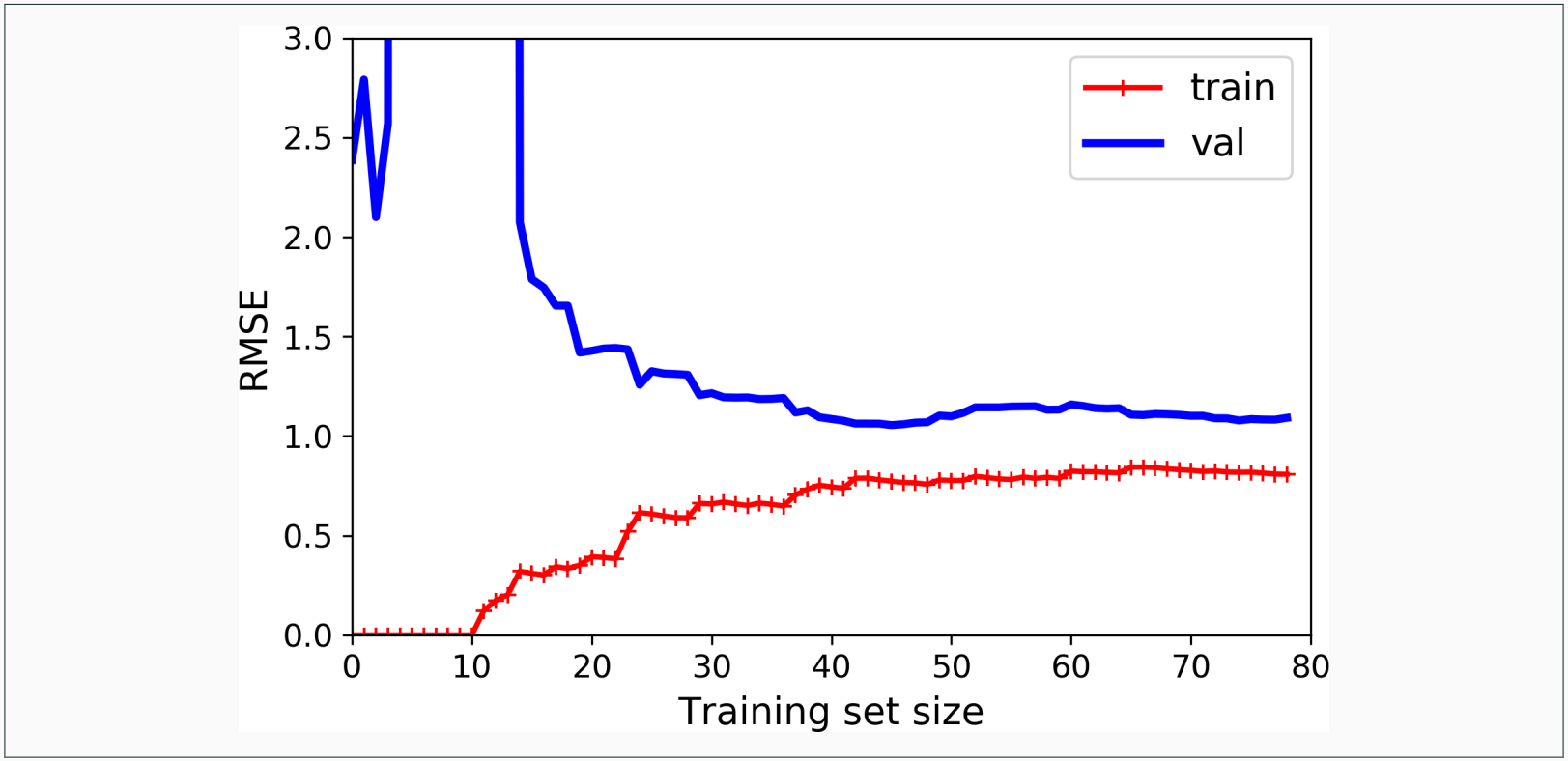


Figure 4-16. Learning curves for the polynomial model

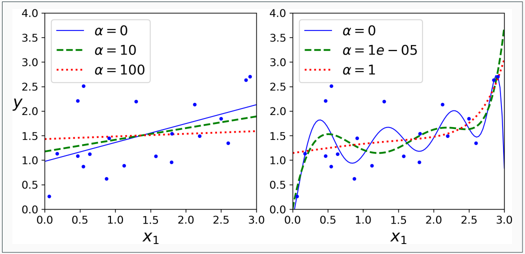
Regularized linear model

SSE e ť نی ב 4 model parameter 4 d̰ نز penalty Ů يد ƀ

- Ö Ŀ پہ ٔ مہ ۉ ƅ model parameter Ů ƀ з نز Ư I ˝Ω ṯ Ị ˝E ىن

$$\mathring{A}(\ ) = \sim\sim N(\ ) + \frac{}{2} k\ \ k^2 \qquad (47)$$

- Hyperparameter ˝E cost u Ṕ يز Ώ ▌Ե u ◀ d̰ Ŀ Ŀ e ظِ
  ɲ ɵ مہ ב Կ
- Closed form solution $\hat{} = (\pm^{\dot{I}} \pm + \ \ \hat{}_{\text{ɧ}})^{\ 1} \pm^{\dot{I}}$ ´



Fiaure 4-17. Ridae Rearession

Lasso regression

SSE e ἵ ﻰﻨﺑ ⅎ model parameter ⅎ ḏ ﺰﻧ penalty ȕ̃ ﺪﻳ ƀ

- Least Absolute Shrinkage and Selection Operator Regression (LASSO)
- ﻅ ȕ̈ ﻢﻣ ٔ ١ א feature ṯ u parameter ȕ̃ ɨ ŋ ﻲﻋ 0 ɣ̍ ̏E ̏Ω ṫ ʊ Ä ﺞﺑ

$$\mathring{A}(\ ) = \sim\sim N(\ ) + \sum_{\ddot{0}=1}^{f} j\ \ddot{0}j \qquad (48)$$

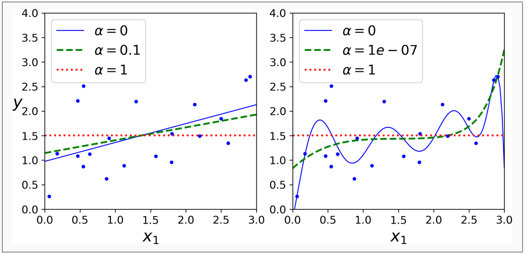


Figure 4-18. Lasso Regression

Appendix

Reference and further reading

- “Chap 4” of A. Geron, Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow

Due: 9 20 Ꮭ Ꮭ ǚ Ꮭ* 23` 59 o

- python جم Ꮭ Ƅ ᵁ jupyter notebook جم Ꮭ ᗅ ̈E i يم د Ⱡ ǚ, (ᗅ ة جم Ꮭ Ị Ƅ з)
- ƠƄ* Ỵ ϖ̈ ǚ چ ḓ نز ▌ỉ مم ᘃ ə ˛ ɭ ɗ يم ⁵ ą,
- ƠƄ* Ỵ ϖ̈ ǚ ▌ỉ نز Ä ǚ ᵘ ˛ ɭ ɗ نز ҚÖ e ꭓ يم ⁵ ą,

ȝ u Ѯ i 28 ր u Figure 9 ǚ ɗ صممⱠ ǚ,

1. 2 ۈ polynomial ẁ ǚ ᵘ ḫ Ƅ ɭ 200 ꝁ ǚ ◕ Ⱡ مم Ⱡ ǚ, (10 ɧ)
2. Closed-form ꓘ ̈E 1, 2, 10, 20 ۈ u يم ǚ ɗ مم Ⱡ ǚ, (10 ɧ)
3. Ᵽ ۈ ב ᚩ ḓ يم batch gradient descent ǚ . يم يم ǚ ɗ مم Ⱡ ǚ, (10 ɧ)
4. Ᵽ ۈ ב ᚩ ḓ يم BGD u iteration ̈Ω ą SSE ǚ ɗ مم ˛ Ủ ثي ̈E ᵁ ع ᵒ Ⱡ ǚ, (10 ɧ)