

Geo Data Science

Polynomial Regression & Regularization

Prof. Dr. Martin Kada

Chair Methods of Geoinformation Science (GIS)
Institute of Geodesy and Geoinformation Science

Copyright Notice

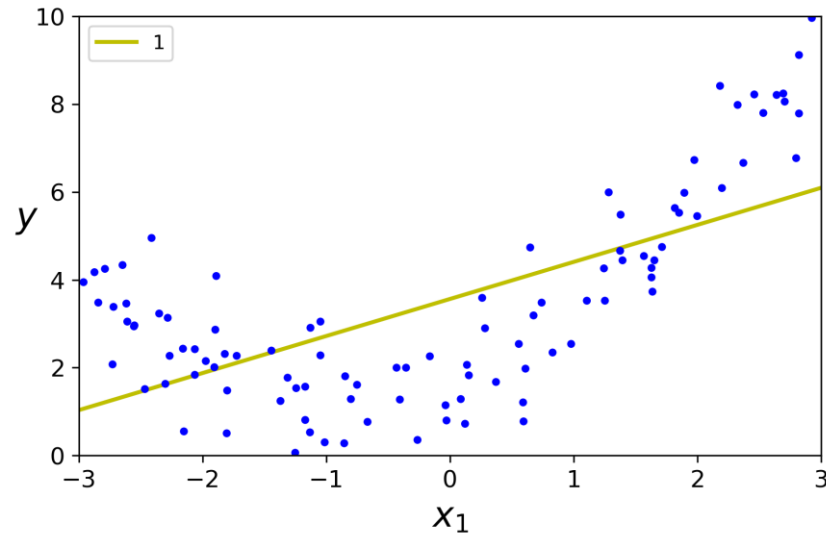
The teaching materials for this course and all elements contained therein are protected by international copyright laws. They may only be used for study purposes for the corresponding course.

Any reproduction and redistribution of the course materials without written permission is prohibited, other than the following: You may print or download them for your own personal use while attending the course.

Motivation

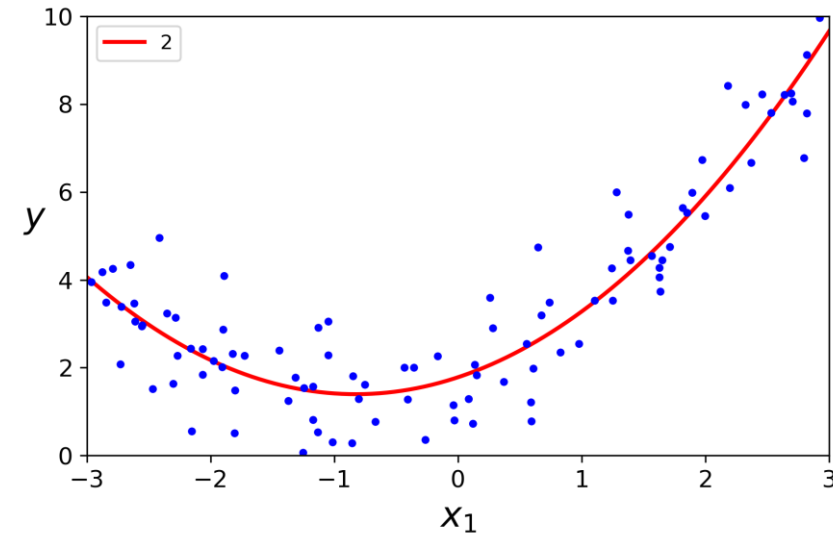
Simple linear model

$$y = \theta_0 + \theta_1 x_1$$



Polynomial model

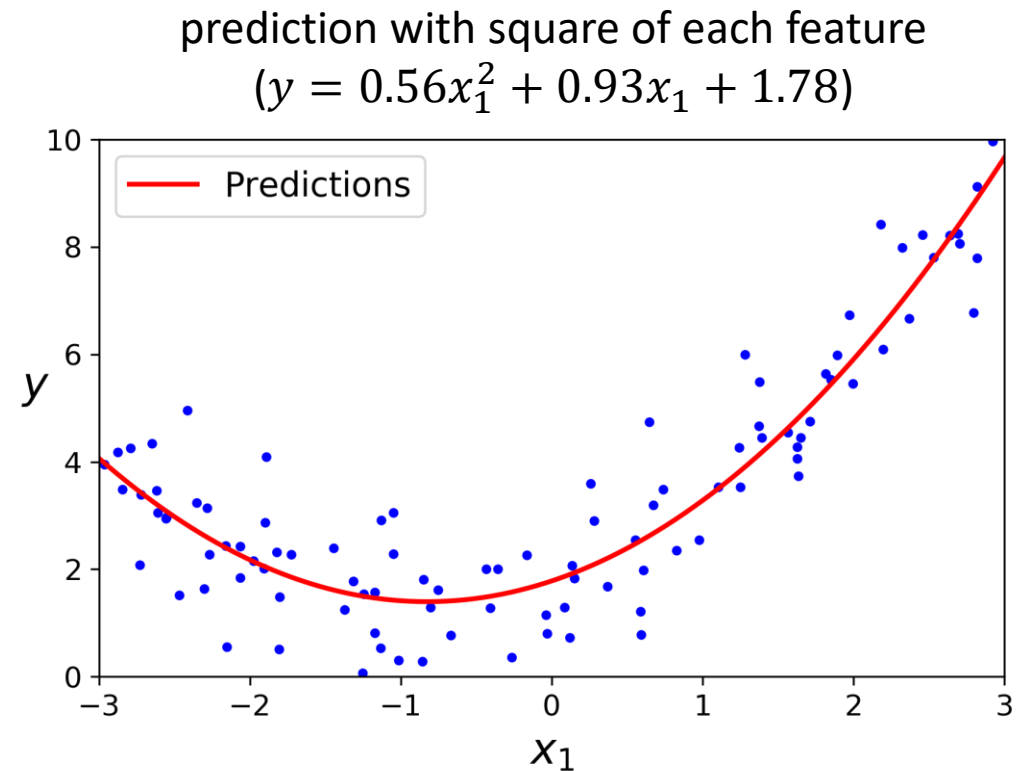
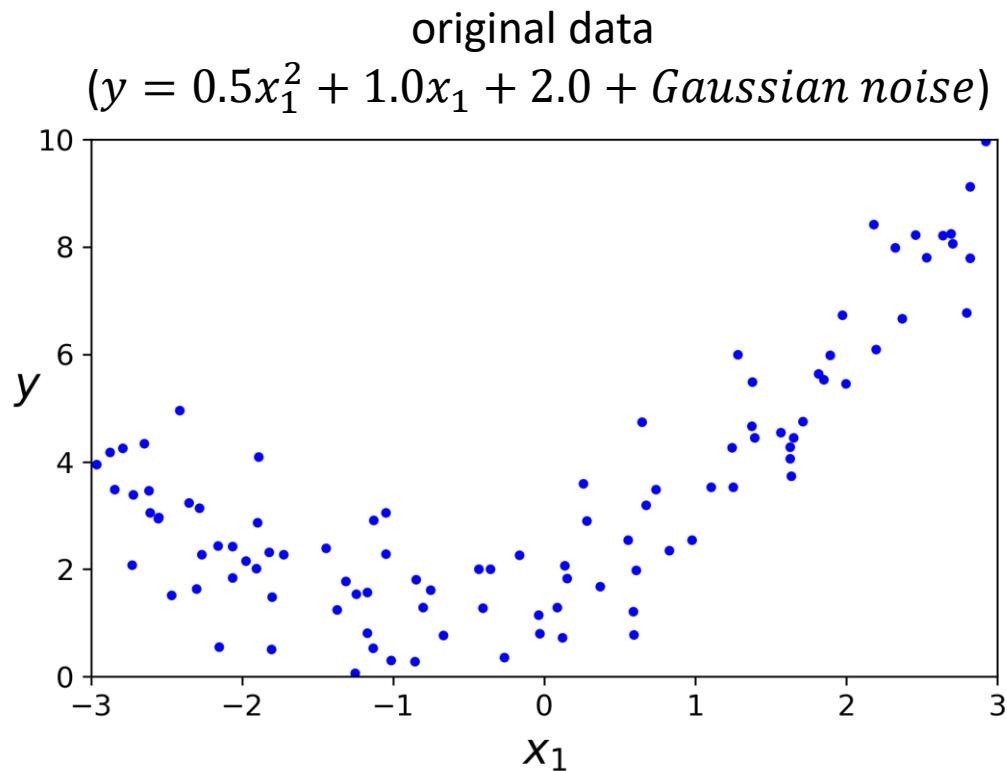
$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$$



- When there is no linear correlation between the **dependent variable** (target value) and the **independent variables** (input features), then a **polynomial regression** can be used to model a nonlinear correlation

Polynomial Regression

- A linear model that also includes the powers of each feature to fit nonlinear data



Polynomial Regression

- Adds all combinations of the n features up to a given degree d ,
 - Results in $\frac{(n+d)!}{d!n!}$ features

θ_0	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9
------------	------------	------------	------------	------------	------------	------------	------------	------------	------------

Example for 2 features
(x_1 and x_2) and
polynomial degree 3

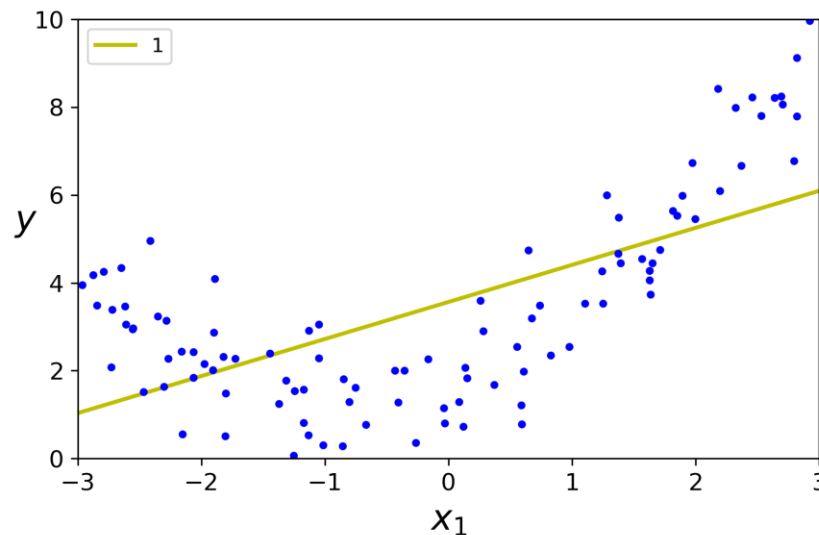
1
x_1
x_2
x_1^2
x_2^2
x_1^3
x_2^3
$x_1 x_2$
$x_1^2 x_2$
$x_1 x_2^2$

=

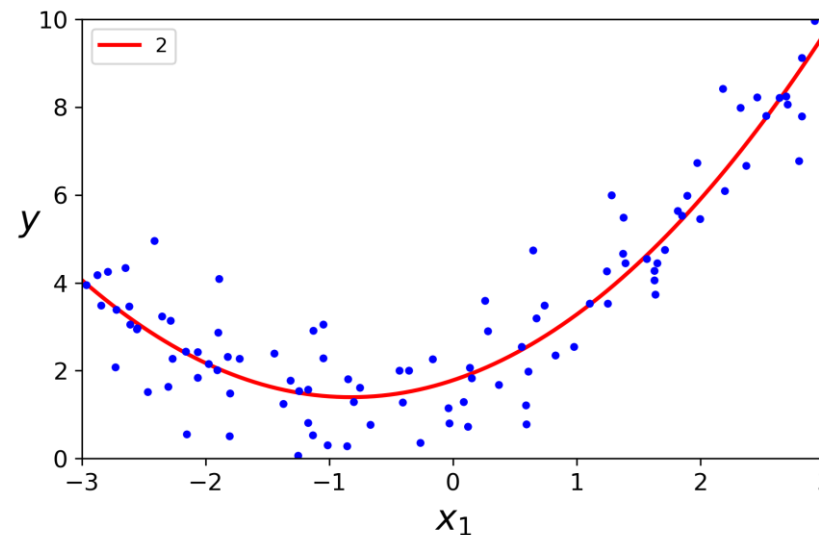
\hat{y}

Polynomial Regression

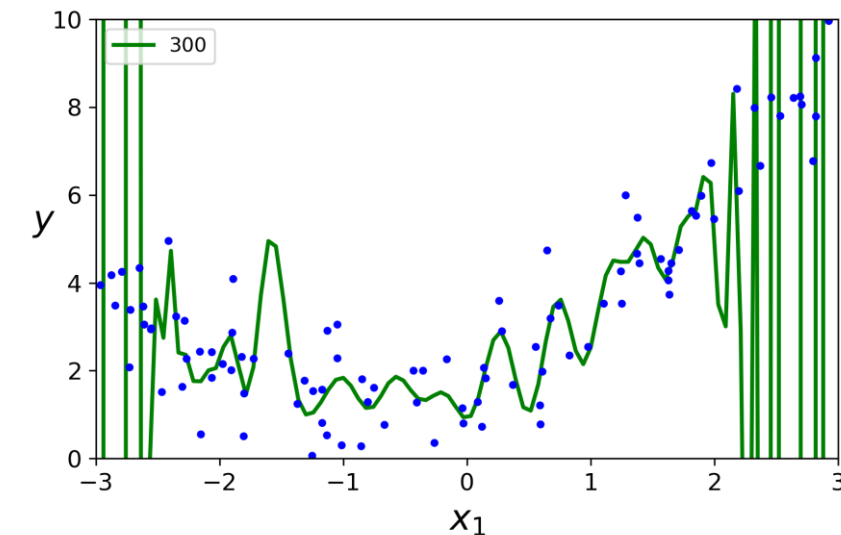
- High-degree Polynomial Regression will likely fit the training data much better than Linear Regression



linear model is underfitting
the training data



quadratic model
generalizes best to new data

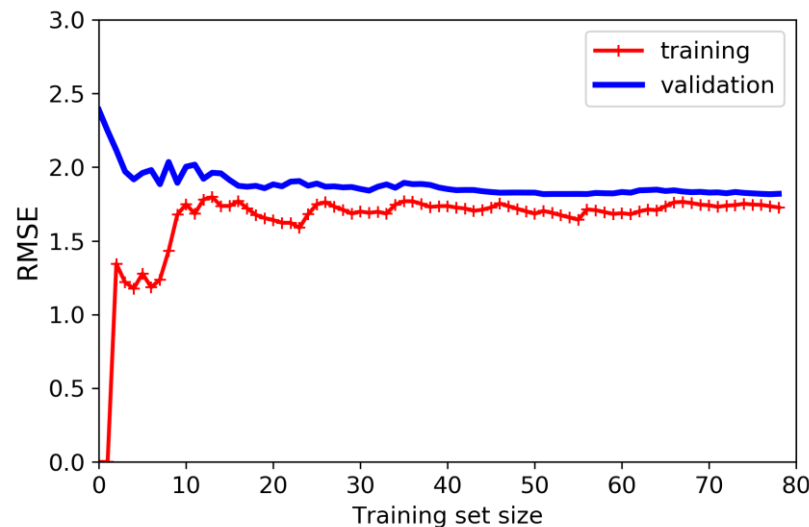


300-degree polynomial model is
severely overfitting the training data

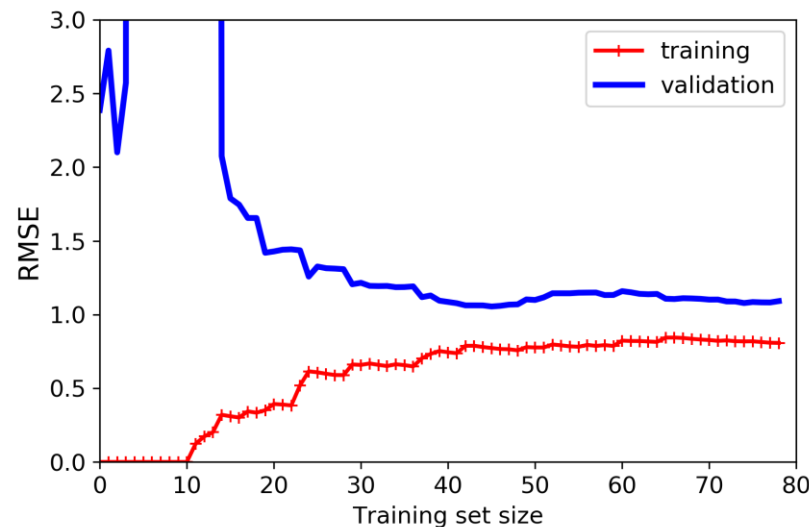
- How to tell that the model is overfitting or underfitting the data?
 - Perform validation with reserved data not used for training (**validation data**) by observing the **learning curves** based on some performance metric (e.g. loss value) between the training data and validation data over time (epochs)
 - Use cross validation if not much training data is available
 - If a model performs well on training data, but **generalizes** poorly on validation data according to the validation metrics → model is **overfitting**
 - If a model performs poorly on both → model is **underfitting**

Recognize Over- and Underfitting

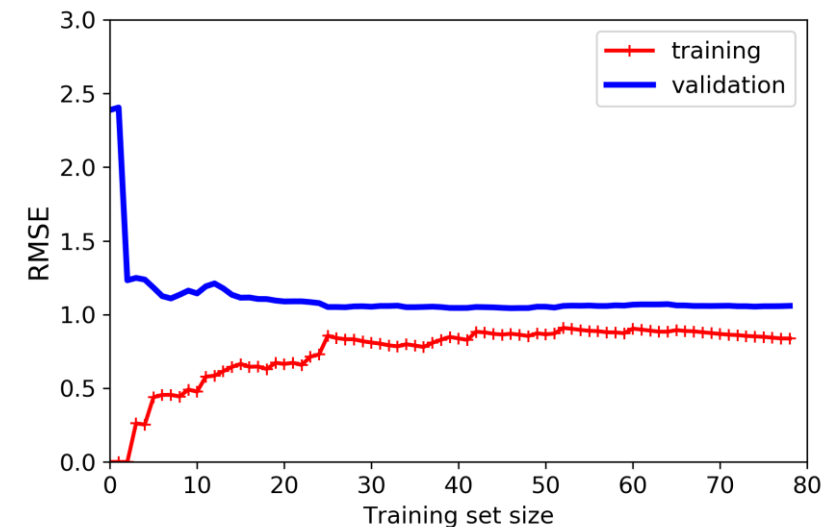
- How to tell that the model is overfitting or underfitting the data?
 - Inspect the learning curves of the model's performance on the training set and the validation set as a function of the training set size



learning curves for
Linear Regression model:
high overall loss → underfitting



learning curves for 10th degree
Polynomial Regression model:
gap between curves → overfitting



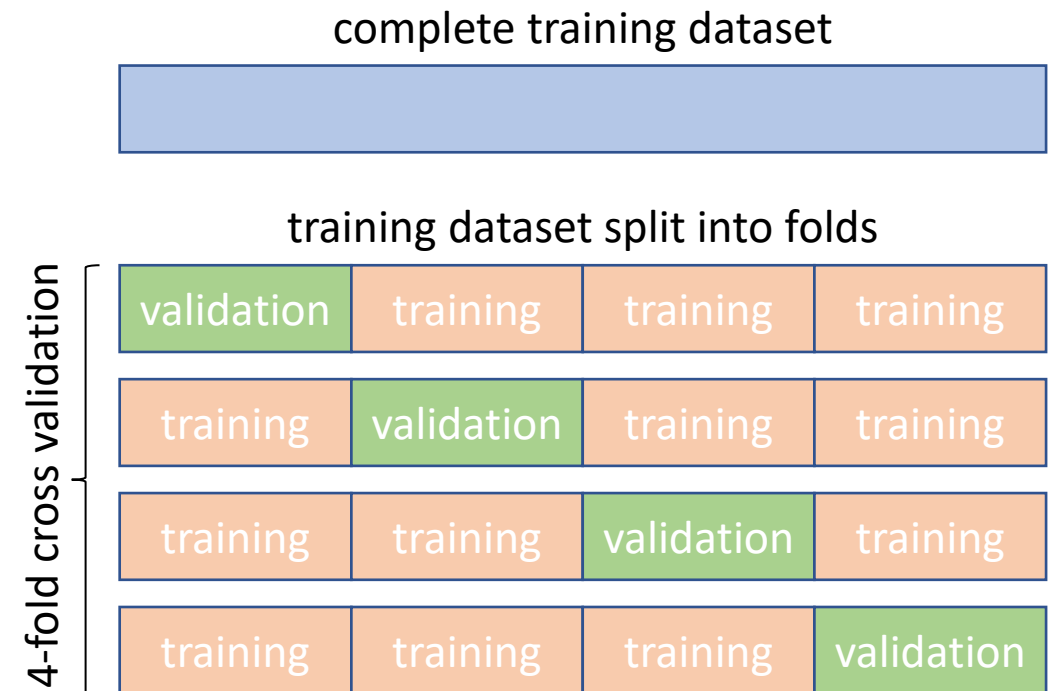
learning curves for 2nd degree
Polynomial Regression model:
low overall loss → good

Cross-Validation

- Training data is often rare, but there is a need to validate and fine tune the hyperparameters of the model with unseen data
 - Keep a small part of the training data for evaluating the model that has not been used during the training process

- Cross-validation:

- Split the training dataset into complementary subsets (called folds)
- Train models against different combinations of these subsets
→ as many models as there are folds
- Validate each model against the remaining (validation) fold
- Average the computed validation values



- Reduce overfitting by **regularizing** (i.e. constraining) the model
 - The fewer degrees of freedom, the less likely to overfit the data
 - The polynomial degree of the Polynomial Regression model is a hyperparameter, so the algorithm cannot reduce the number of polynomial degrees itself while training the model
 - Constrain the weights of the linear model:
 - Ridge Regression
 - Lasso Regression
 - Elastic Net

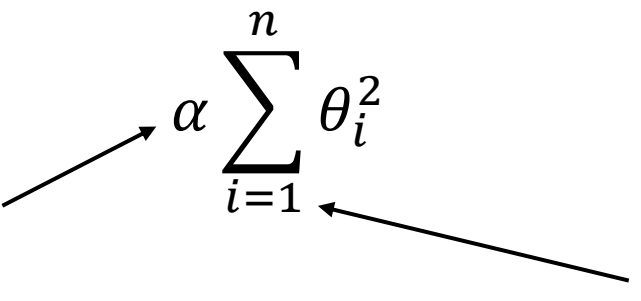
Ridge Regression

- In Ridge Regression (also called Tikhonov regularization), a regularization term equal to the **sum of the squared weights** is added to the cost function

hyperparameter that controls how much the model is regularized

$$\alpha \sum_{i=1}^n \theta_i^2$$

bias θ_0 is not part of the regularization



- Keeps the model weights as small as possible while fitting the data
- Regularization term is used only for training, but omitted for evaluating the model's performance and for predictions

Ridge Regression

- Ridge Regression cost function:

$$J(\theta) = \text{MSE}(\theta) + \alpha \frac{1}{2} \sum_{i=1}^n \theta_i^2$$

$\frac{1}{2}$ does not change the regularization term,
but simplifies the gradient computation

- If θ denotes the feature weight vector (θ_1 to θ_n), then the regularization term is equal to $\frac{1}{2} (\|\theta\|_2)^2$, where $\|\cdot\|_2$ represents the ℓ_2 norm of the weight vector
- For Gradient Descent, $\alpha\theta$ is added to the MSE gradient vector

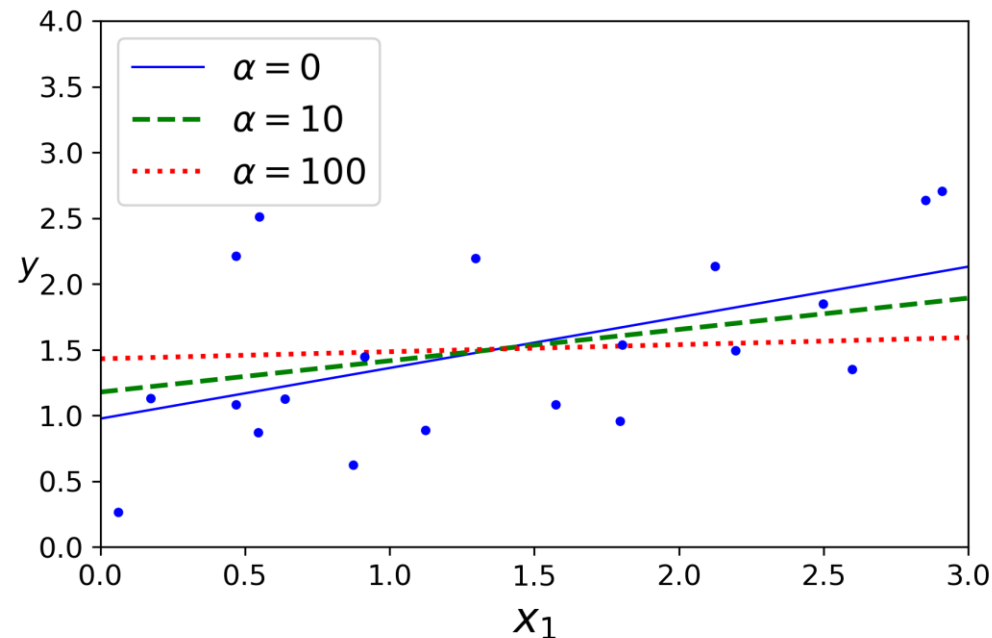
- Ridge Regression closed-form:

$$\hat{\theta} = (\mathbf{X}^T \cdot \mathbf{X} + \alpha \mathbf{A})^{-1} \cdot \mathbf{X}^T \cdot \mathbf{y}$$

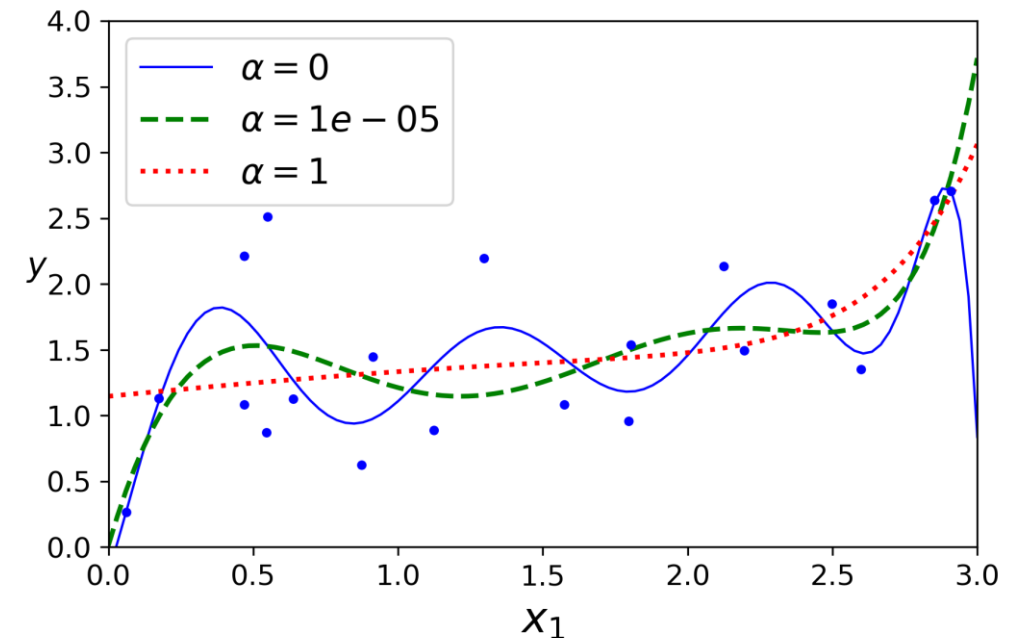


$n \times n$ identity matrix
with the exception of a 0 in the top-left cell
(corresponding to the bias term)

Ridge Regression



Ridge Regression (without polynomials) on linear data leading to linear predictions



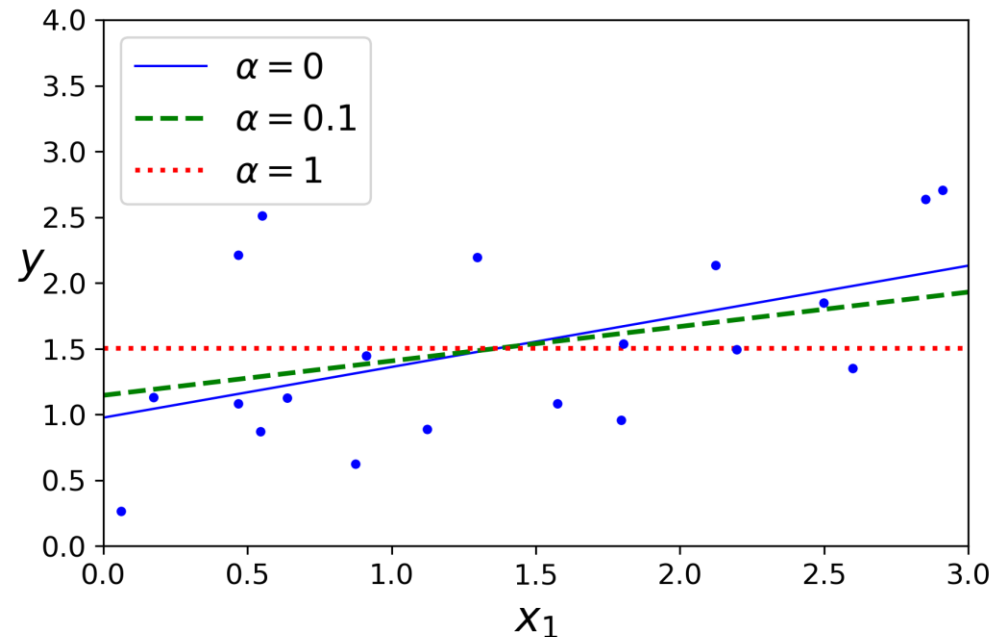
Ridge Regression on linear data that was first expanded using 10th degree polynomial features, then scaled, and finally the Ridge models applied (Polynomial Regression with Ridge Regularization)

- In Least Absolute Shrinkage and Selection Operator Regression (simply called Lasso Regression), the ℓ_1 norm of the weight vector is added to the cost function as a regularization term:

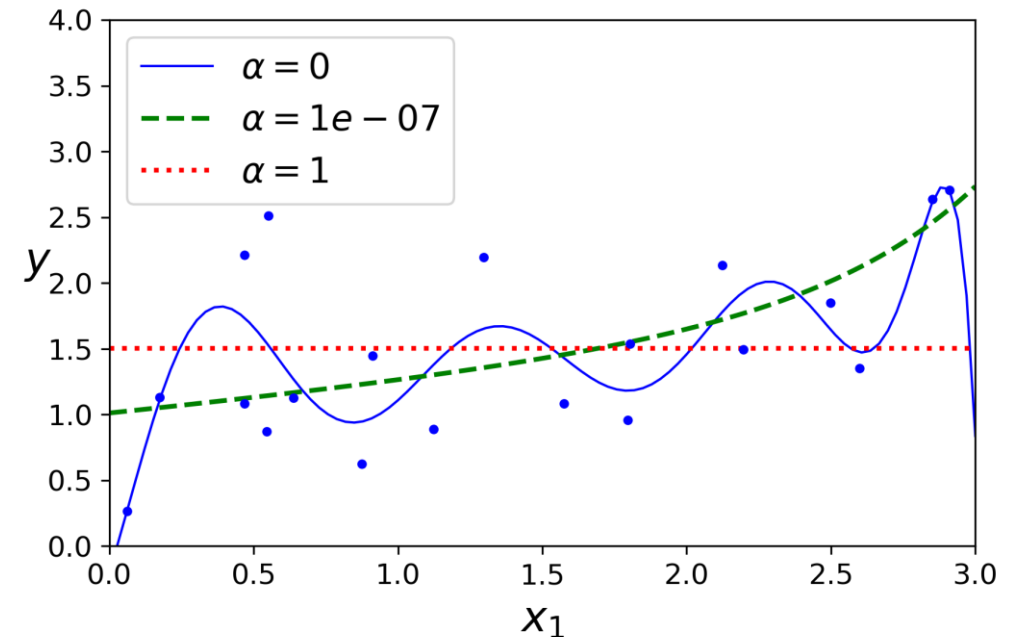
$$J(\theta) = \text{MSE}(\theta) + \alpha \sum_{i=1}^n |\theta_i|$$

- Lasso regularization tends to completely eliminate the weights of the least important features by setting them to zero
 - Performs feature selection and outputs a sparse model with few nonzero weights

Lasso Regression



Lasso Regression (without polynomials) on linear data leading to linear predictions



Lasso Regression on linear data that was first expanded using 10th degree polynomial features, then scaled, and finally the Lasso models applied (Polynomial Regression with Lasso Regularization)

Lasso Regression

- The Lasso cost function is not differentiable at $\theta_i = 0$ (for $i = 1, 2, \dots, n$), but Gradient Descent still works if a sub-gradient vector \mathbf{g} is used instead
- Lasso Regression sub-gradient vector:

$$g(\theta, J) = \nabla_{\theta} \text{MSE}(\theta) + \alpha \begin{pmatrix} \text{sign}(\theta_1) \\ \text{sign}(\theta_2) \\ \vdots \\ \text{sign}(\theta_n) \end{pmatrix} \quad \text{where} \quad \text{sign}(\theta_i) = \begin{cases} -1 & \text{if } \theta_i < 0 \\ 0 & \text{if } \theta_i = 0 \\ 1 & \text{if } \theta_i > 0 \end{cases}$$

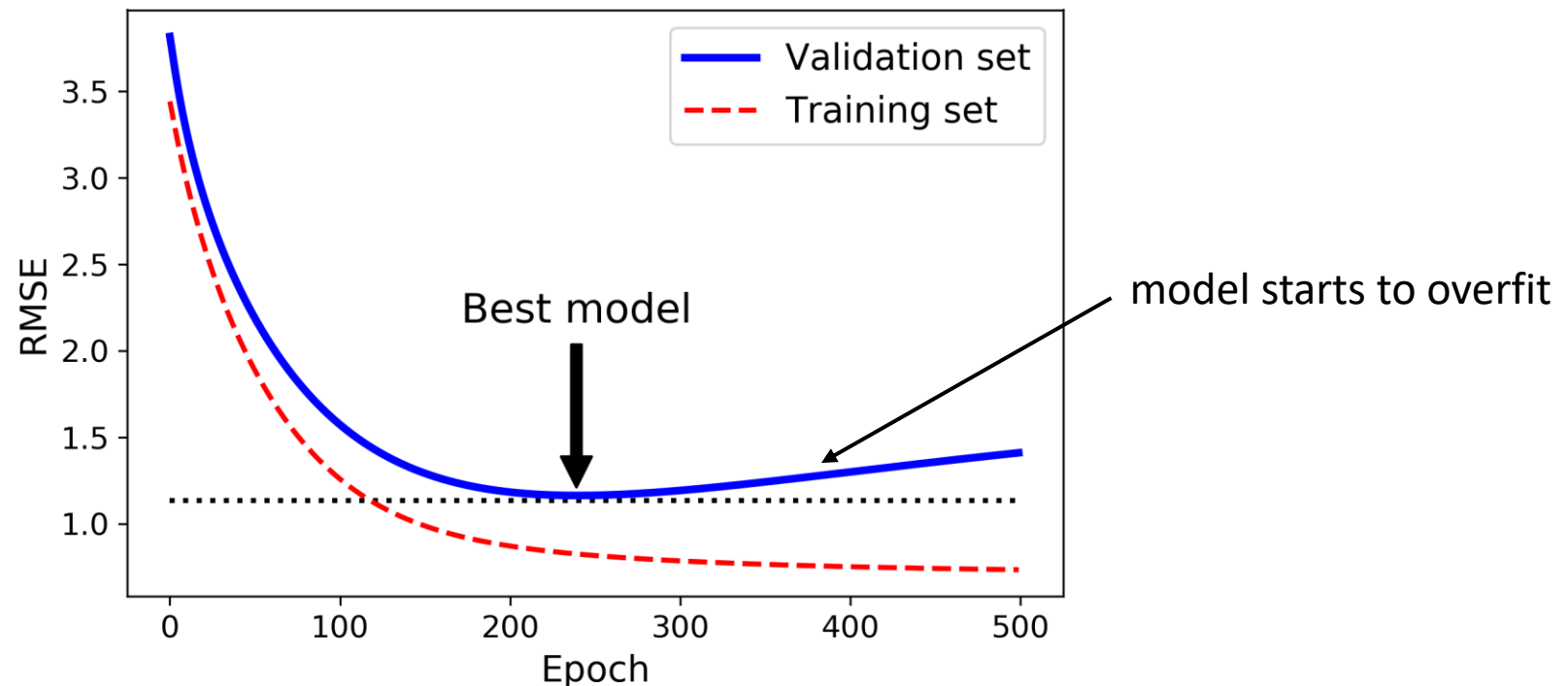
- Regularization term is a mix of both Ridge and Lasso's regularization terms controlled by the mix ratio r ($0 \leq r \leq 1$)
 - If $r = 0$, it is equivalent to Ridge Regression
 - If $r = 1$, it is equivalent to Lasso Regression
- Elastic Net cost function:

$$J(\theta) = \text{MSE}(\theta) + r\alpha \sum_{i=1}^n |\theta_i| + \frac{1-r}{2}\alpha \sum_{i=1}^n \theta_i^2$$

- When to use which regularization method?
 - Polynomial Regression without regularization should be avoided
 - Ridge Regression is a good default
 - Lasso or Elastic Net if only a few features are actually useful
 - Elastic Net is preferred over Lasso since Lasso may behave erratically when the number of features is greater than the number of training instances or when several features are strongly correlated

Early Stopping

- When using an iterative learning algorithm such as Gradient Descent, another way to regularize is to stop training as soon as the validation error reaches a minimum



Thank you for your attention!