2.1 - Training, Validation and Testing

Giovanni Della Lunga
giovanni.dellalunga@unibo.it

Introduction to Machine Learning for Finance

Bologna - February-March, 2025

Subsection 1

Training, Validation and Testing

Introduction

- Before delving into specific machine learning models, it is crucial to establish a rigorous understanding of how to **train**, **validate** and **test** models properly.
- This step is not merely a technical detail but a **fundamental prerequisite** for ensuring that any model we build is reliable, interpretable, and capable of generalizing to new data.
- Without a robust evaluation framework, even the most sophisticated models can yield misleading results, leading to overconfidence in predictions that do not hold up in real-world applications.

Introduction

- In the next slides we're going to introduce several key concepts that are **transversal to all of machine learning**.
- Among these, the notions of **overfitting** and **bias and variance** play a central role.
- In particular bias and variance, often in tension with each other, govern the fundamental trade-offs in model performance:
  - a model with high bias oversimplifies the problem and fails to capture essential patterns...
  - whereas a model with high variance is overly sensitive to training data and performs poorly on unseen examples
- Understanding how to balance these factors is **critical for designing robust machine learning systems**.

Introduction

- To develop intuition for these ideas, we will use a **simple regression model** as our primary tool.
- This choice is intentional: rather than getting caught up in the intricacies of a complex algorithm, we will focus on the core issues of **model training and validation, bias-variance tradeoff, and generalization**.
- Regression models provide a clear and interpretable framework for illustrating these concepts, making them an ideal starting point before extending these ideas to more advanced models.

Introduction

In the following sections, we will cover the essential components of model validation, including:

- The distinction between training, validation, and test sets
- Bias-variance decomposition and its implications
- Overfitting and underfitting: causes and remedies
- Regularization Methods

By mastering these principles early, we establish a solid foundation that will allow us to critically assess the performance of any machine learning model we encounter. With these concepts in place, we can confidently proceed to more advanced methods, knowing that we have the tools to rigorously evaluate their effectiveness.

Training, Validation and Testing

- When data is used for forecasting there is a danger that the machine learning model **will work very well for training data, but will not generalize well to other data**;
- An obvious point is that it is important that the data used in a machine learning model be representative of the situations to which the model is to be applied;
- It is also important to test a model **out-of-sample**, by this we mean that the model should be tested on data that is different from the sample data used to determine the parameters of the model;
- Data scientist refer to the sample data as the **training set** and the data used to determine the accuracy of the model as the **test set**;
- Often a **validation set** is used as well as we explain later;

Training, Validation and Testing

- We will illustrate the use of a training set and the test data set with a very simple Example (Hull, Chapter 1);
- We suppose that awe are interested in forecasting the salaries of people from their age;
- This simple regression model is an example of supervised learning...

Training Set

Training, Validation and Testing

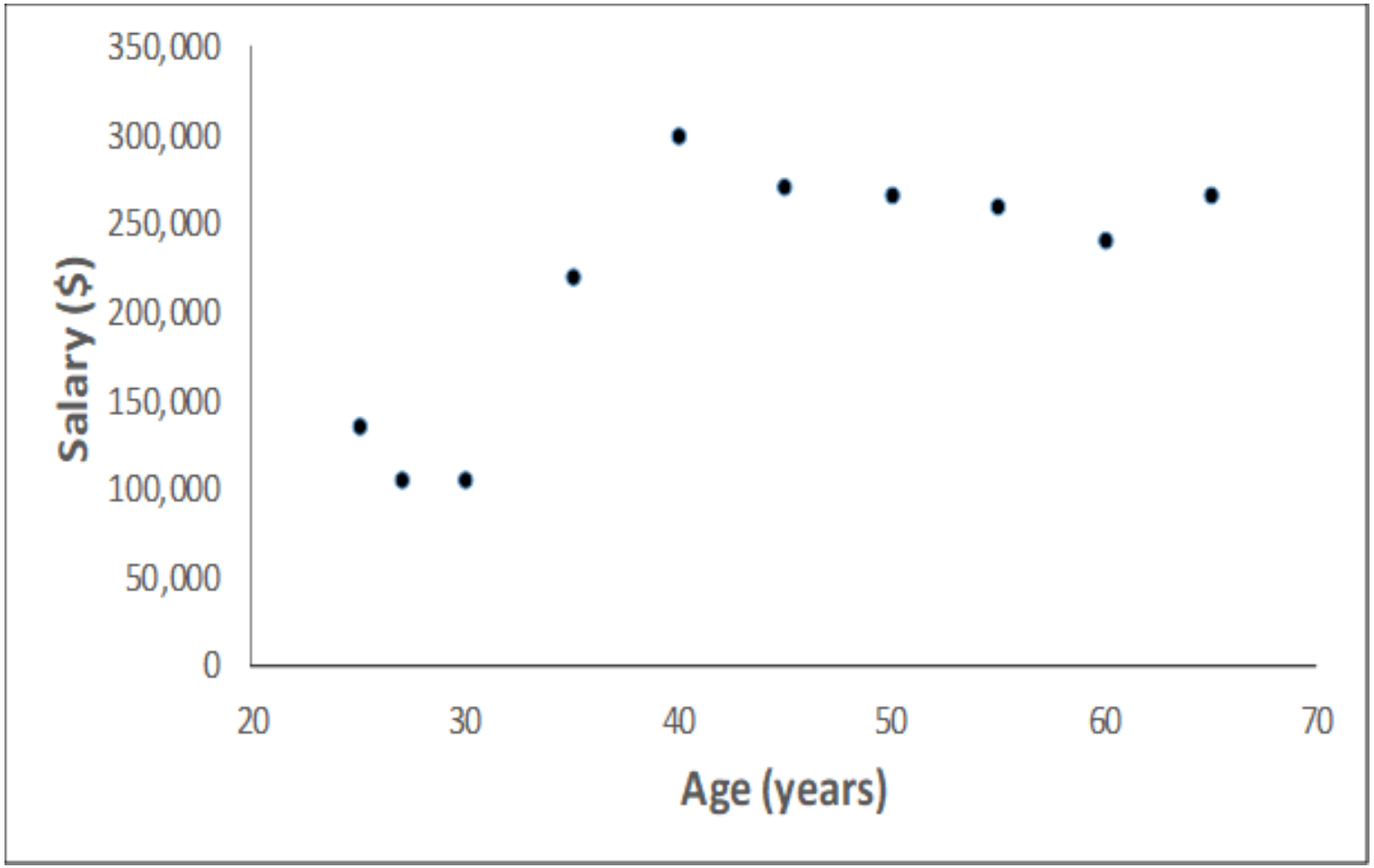
Table 1. Salary as a function of Age for a certain profession in a certain area)

Age (years)	Salary ($)
25	135,000
55	260,000
27	105,000
35	220,000
60	240,000
65	265,000
45	270,000
40	300,000
50	265,000
30	105,000

Training Set

Training, Validation and Testing

Figure 1. Scatter plot of Salary as a function of Age (see Table 1)

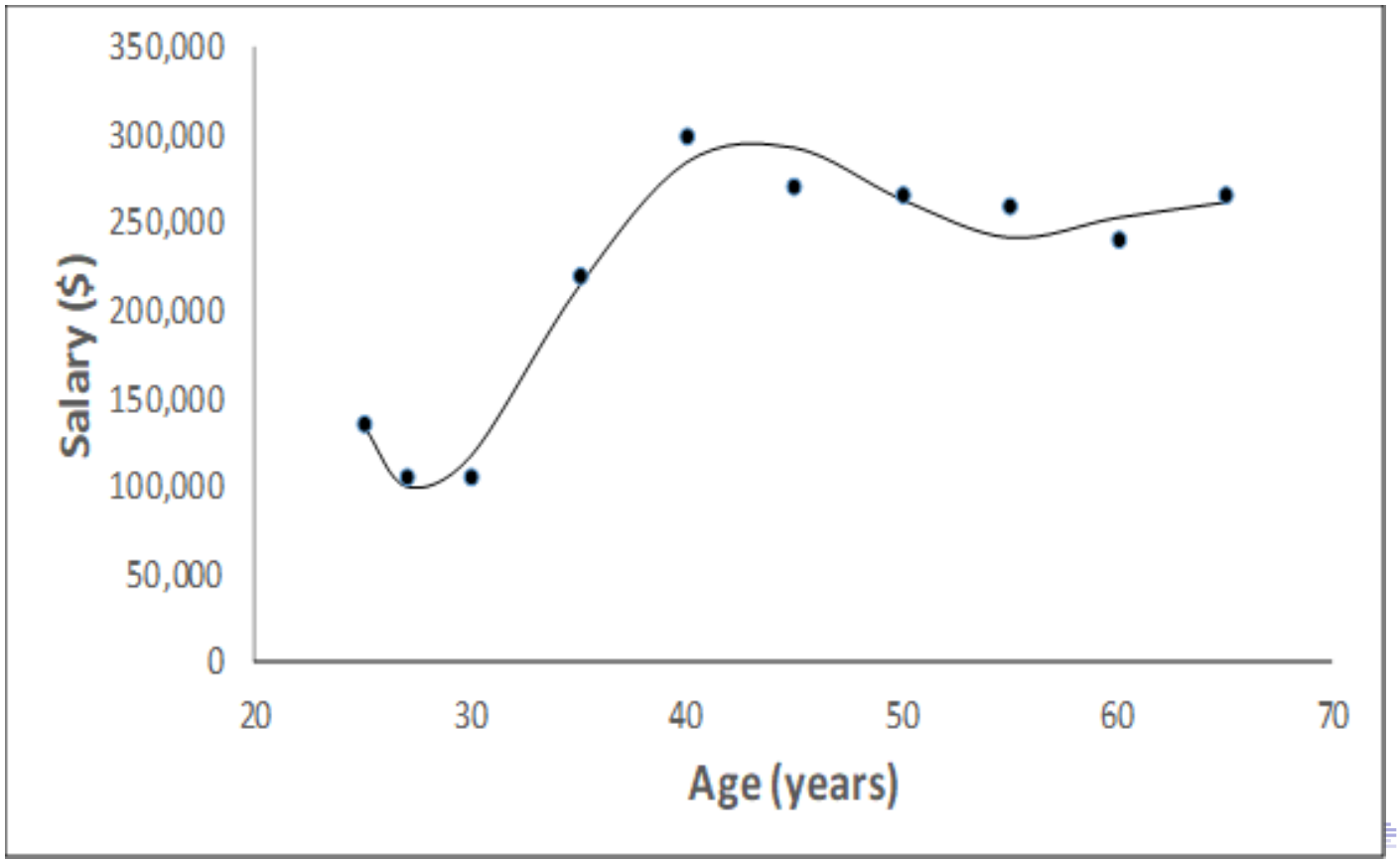


Training Set

Training, Validation and Testing

Figure 2. It's tempting to choose a model that fits the data really well for example with a polynomial of degree five (Y = Salary, X = Age):

$$Y = a + b_1X + b_2X^2 + b_3X^3 + b_4X^4 + b_5X^5$$



Discussion of Training Result

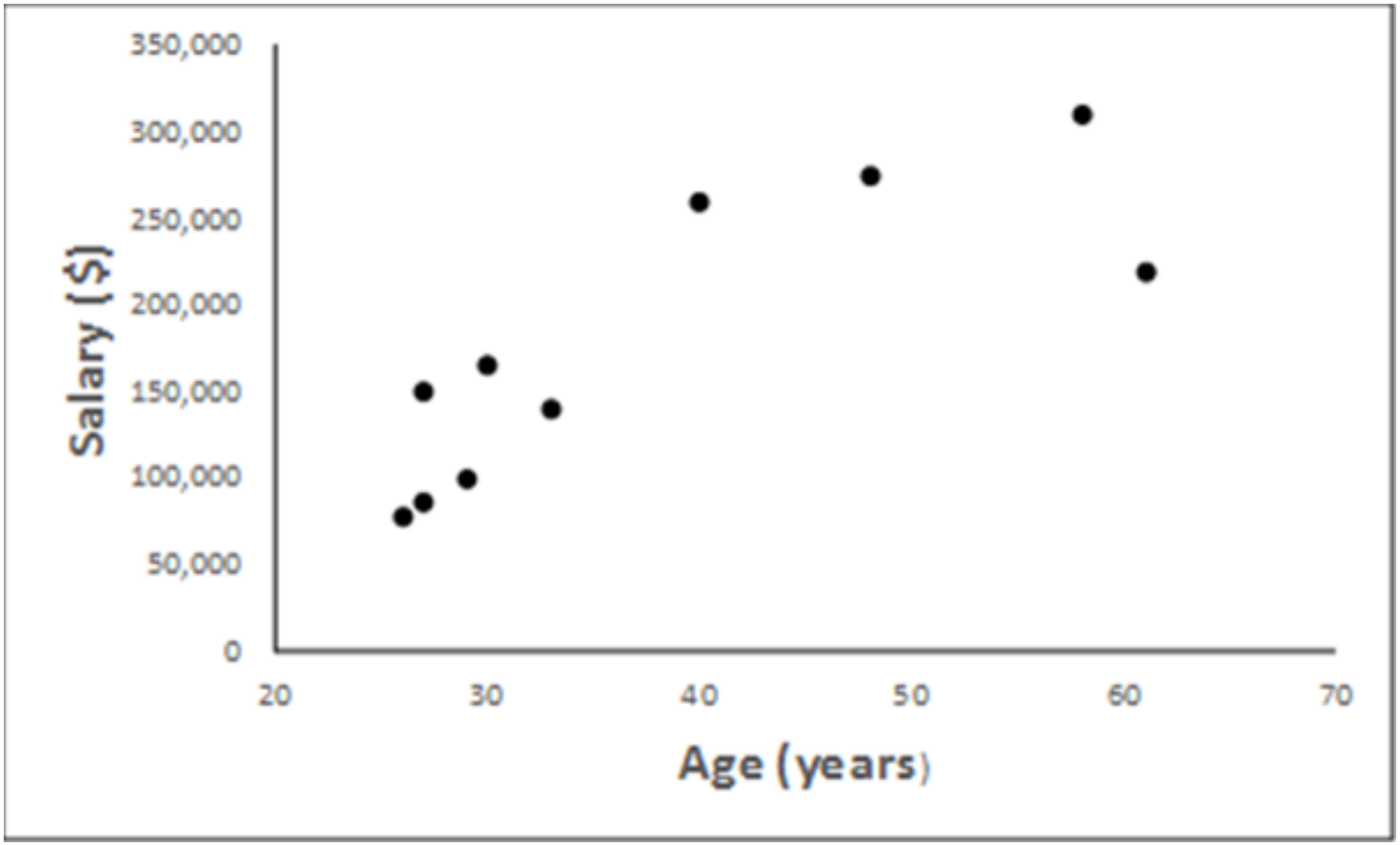
Training, Validation and Testing

- The model provides a good fit to the data;
- The standard deviation of the difference between the salary given by the model and the actual salary for the ten individuals in the training data set (which is referred to as the **root mean square error (rmse)**) is \$12902;
- However common sense would suggest that we many over-fitted the data;
- We need to check the model out-of-sample;
- To use the language of *data science* we need to determine whether the model generalizes well to a new data set that is different from the training set.

Validation Set

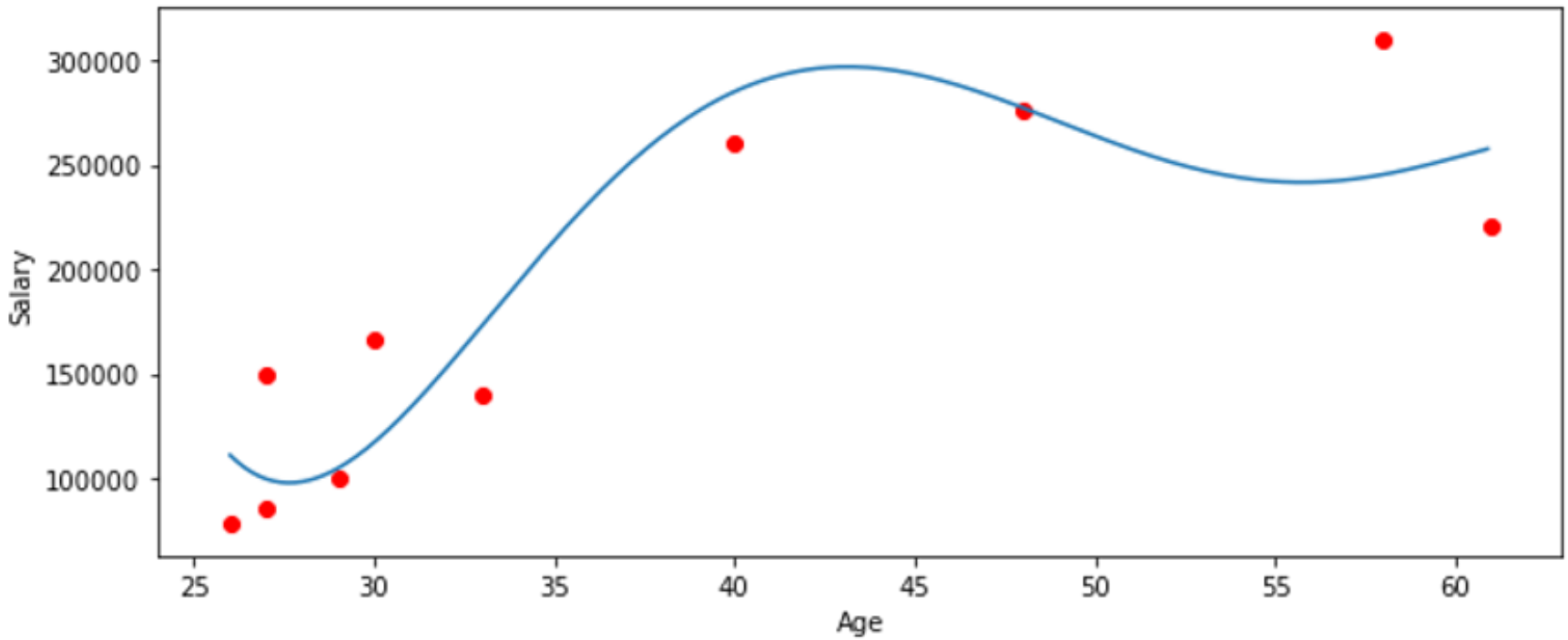
Training, Validation and Testing

Figure 3. An Out-of-Sample Validation Set



Validation and Testing: Validation Set

Figure 3. Scatter Plot for Validation Set



Validation and Testing: Discussion of Validation Result

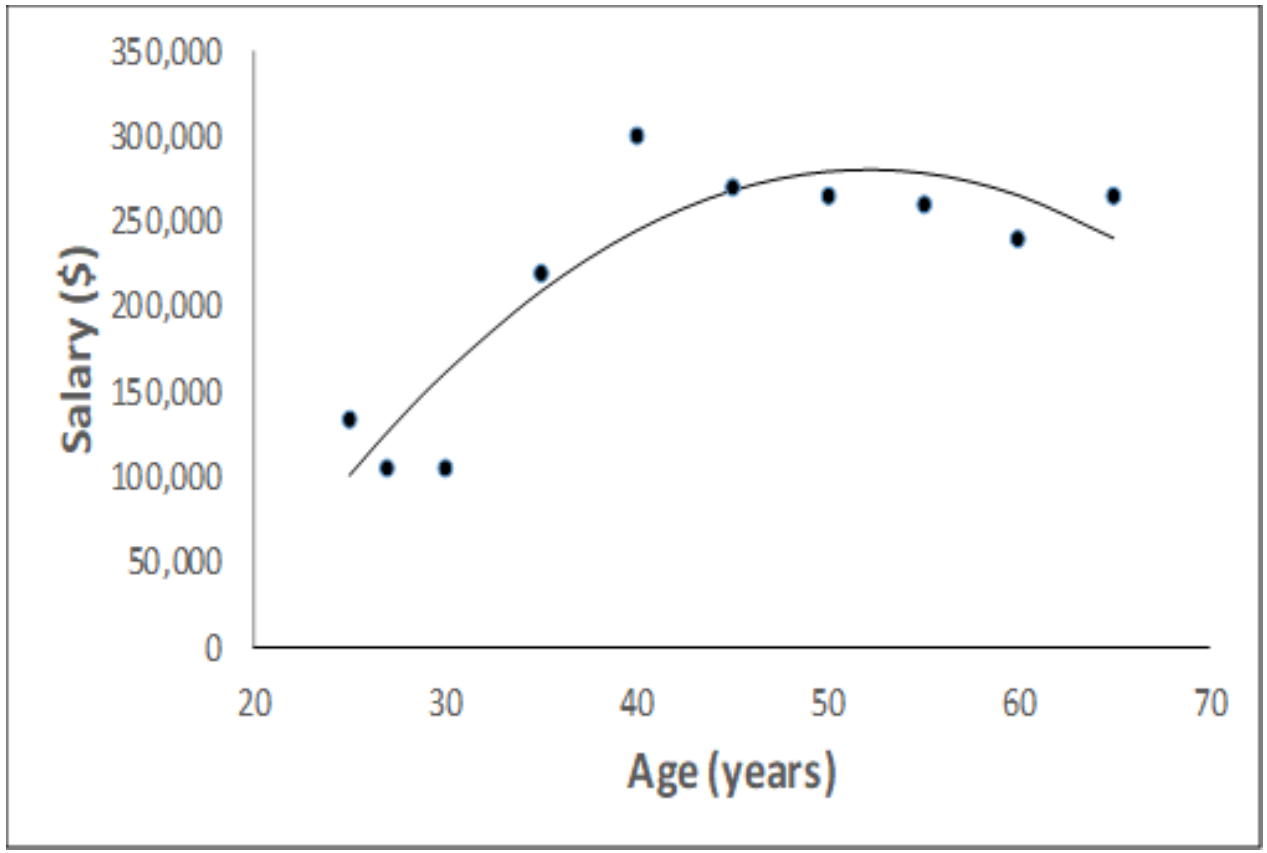
The Fifth Order Polynomial Model Does Not Generalize Well

- The root mean squared error (rmse) for the training data set is $12,902
- The rmse for the test data set is $38,794
- We conclude that the model overfits the data

Validation and Testing

Figure 4. A Simpler Quadratic Model

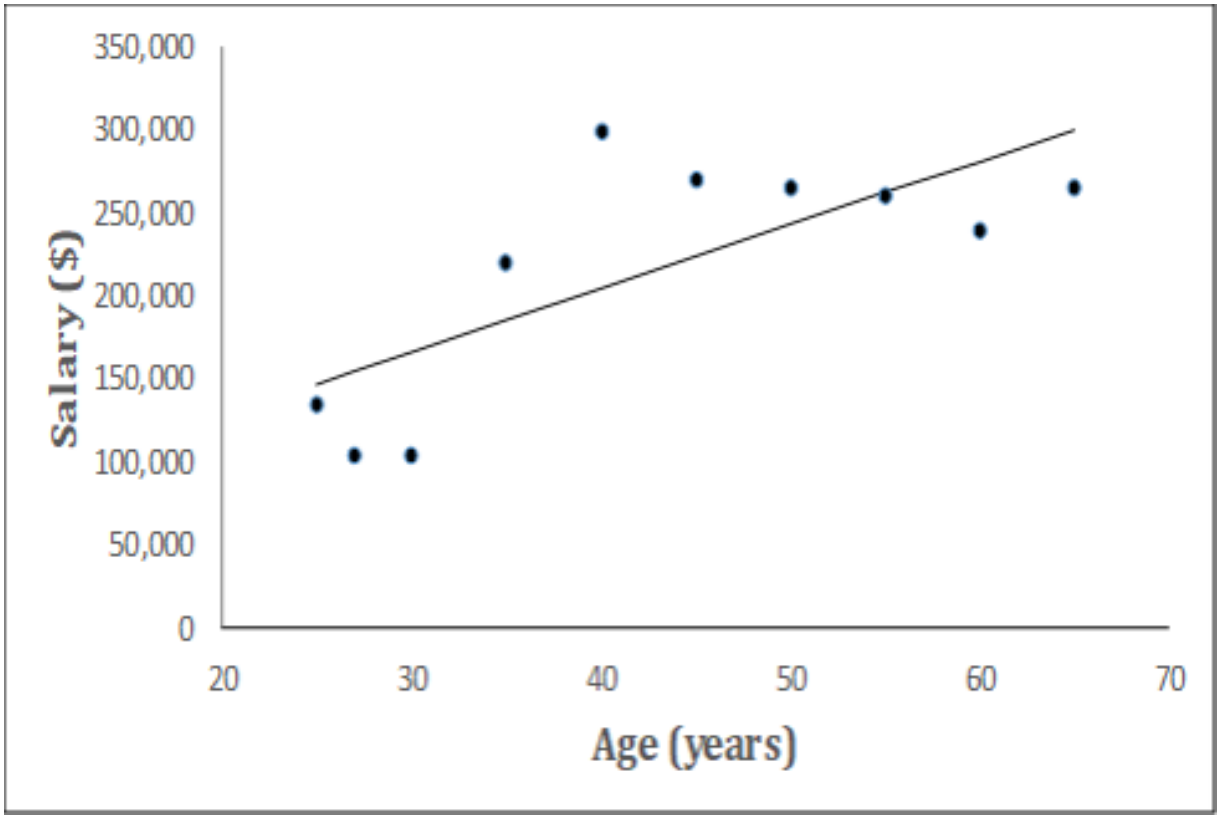
$$Y = a + b_1 X + b_2 X^2$$



Validation and Testing

Figure 5 . Linear Model

$$Y = a + b_1 X$$



Validation and Testing

Table 3. Summary of Results: The linear model under-fits while the 5th degree polynomial over-fits:

	Polynomial of degree 5	Quadratic model	Linear model
Training set	12, 902	32,932	49,731
Validation set	38,794	33,554	49,990

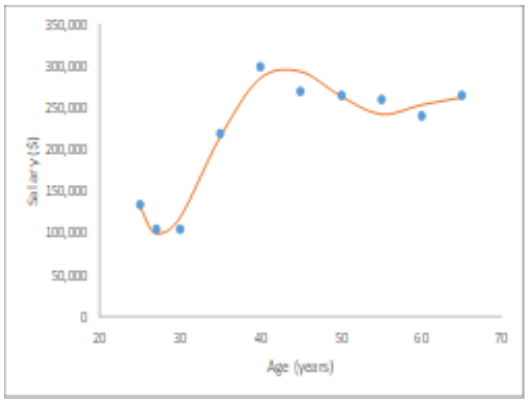
Validation and Testing

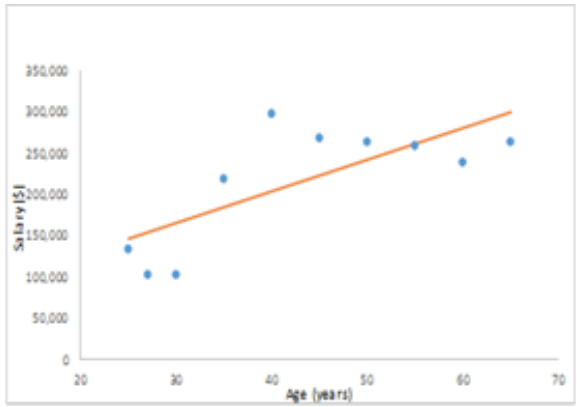
Table 3. Summary of Results: The linear model under-fits while the 5th degree polynomial over-fits:

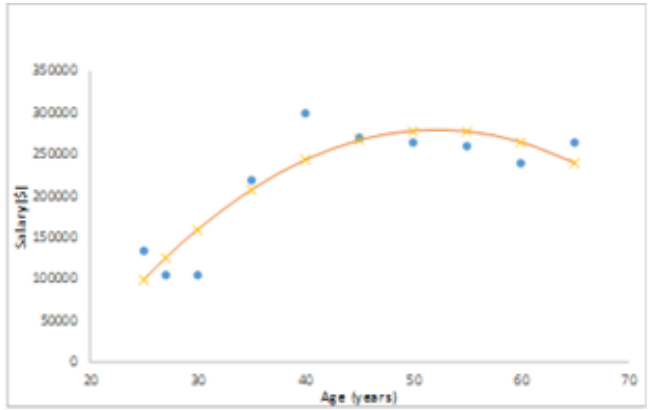
	Polynomial of degree 5	Quadratic model	Linear model
Training set	12,902	32,932	49,781
Validation set	38,794	33,554	49,990

Validation and Testing

Figure 6. Overfitting/Underfitting Example: predicting salaries for people in a certain profession in a certain area (only 10 observations)



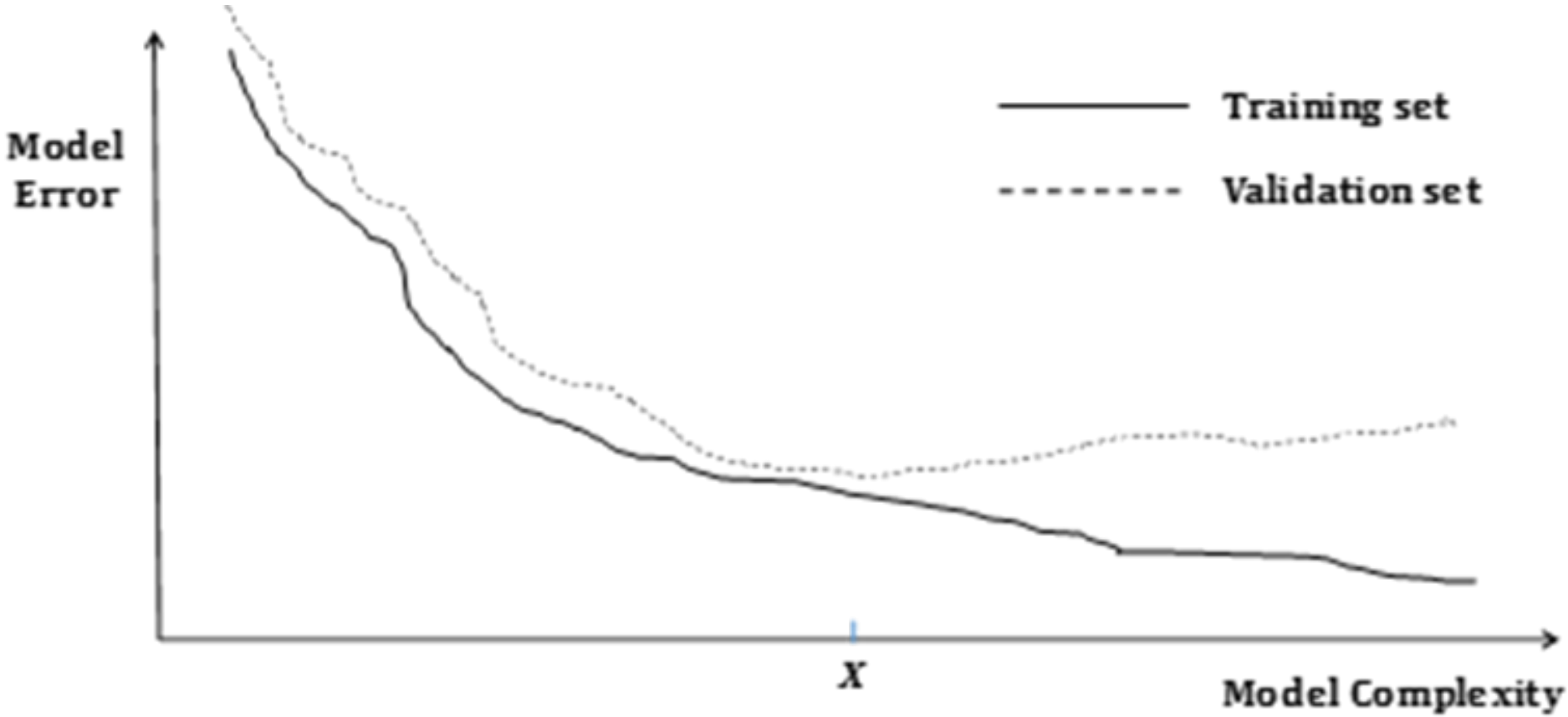




Overfitting ——— Underfitting ——— Best model?

Validation and Testing

Typical Pattern of Errors for Training Set and Validation Set



Validation and Testing

ML Good Practice

- Divide data into three sets
- Training set
- Validation set
- Test set
- Develop different models using the **training set** and compare them using the **validation set**;
- Rule of thumb: increase model complexity until model no longer generalizes well to the validation set;
- The **test set** is used to provide a final out-of-sample indication of how well the chosen model works;

Subsection 2

Bias and Variance

Bias and Variance

- Suppose there is a relationship between an independent variable x and a dependent variable y:

$$y = f(x) + \epsilon \tag{1}$$

where ϵ is an error term with mean zero and variance σ^2.

- The error term captures either genuine randomness in the data or noise due to measurement error.
- Suppose we find, with a Machine Learning technique, a deterministic model for this relationship:

$$y = \hat{f}(x) \tag{2}$$

Bias and Variance

- Now it comes a new data point x' not in the training set and we want to predict the corresponding y';
- The error we will observe in our model at point x' is going to be

$$\hat{f}(x') - f(x') - \epsilon \tag{3}$$

- There are two different sources of error in this equation.
- The first one is included in the factor ϵ;
- The second one, more interesting, is due to what is in our training set.
- A robust model should give us the same prediction whatever data we used for training out model.

Bias and Variance

- Let's look at the average error:

$$E\left[\hat{f}(x')\right] - f(x') \tag{4}$$

where the expectation is taken over random samples of training data (having the same distributio as the training data).

- This is the definition of the **bias**

$$\text{Bias}\left[\hat{f}(x')\right] = E\left[\hat{f}(x')\right] - f(x') \tag{5}$$

Bias and Variance

- We can also look at the mean square error

$$E\left[\left(\hat{f}(x') - f(x') - \epsilon\right)^2\right] = \left[\text{Bias}\left(\hat{f}(x')\right)\right]^2 + \text{Var}\left[\hat{f}(x')\right] + \sigma^2$$

  where we remember that $\hat{f}(x')$ and ϵ are independent.
- This show us that there are two important quantities, the **bias** and the **variance** that will affect our results and that we can control to some extent.

Bias and Variance

- **What is Bias?** It's the difference between the average prediction of our model and the correct value which we are trying to predict. Model with high bias pays very little attention to the training data and oversimplifies the model. It always leads to high error on training and test data.
- **What is Variance?** It's the variability of model prediction for a given data point or a value which tells us spread of our data. Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn't seen before. As a result, such models perform very well on training data but has high error rates on test data.

Bias and Variance

Low Bias and Low Variance

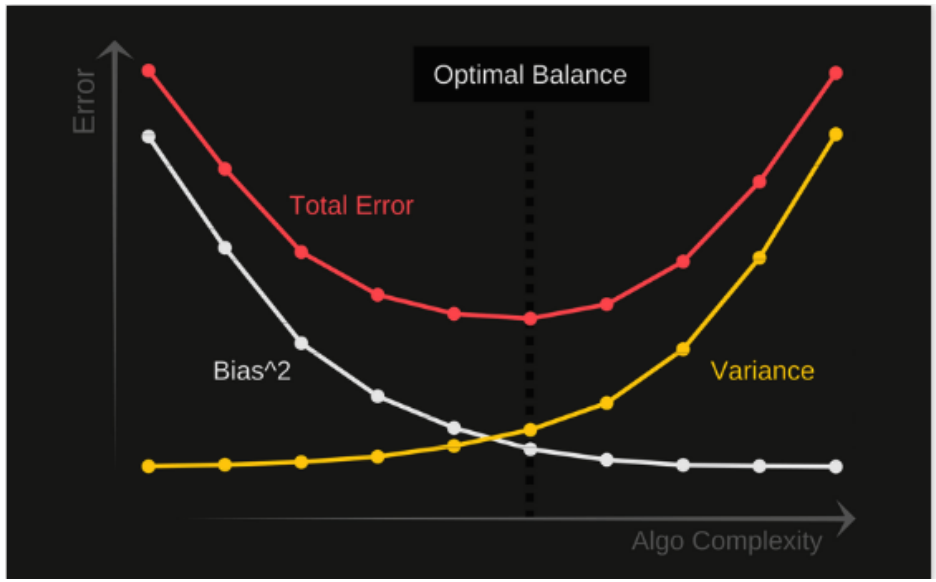
High Bias and Low Variance

Low Bias and High Variance

High Bias and High Variance

Bias and Variance

Unfortunately, we often find that there is a trade-off between bias and variance. As one is reduced, the other is increased. This is the matter of over- and under-fitting.



Subsection 3

Regularization

Regularization

- Linear regression can over-fit, particularly when there are a large number of correlated features.
- Results for validation set may not then be as good as for training set
- Regularization is a way of avoiding overfitting and reducing the number of features. Alternatives:
  - **Ridge Regression**
  - **Lasso Regression**
  - **Elastic net**
- We must first scale feature values

Transforming Polynomial Regression into Linear Regression

Regularization

- Polynomial regression allows modeling nonlinear relationships.
- Despite its name, it can be rewritten as a linear regression.
- This is done by redefining input features.

Polynomial Regression Model

General form of polynomial regression:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_d x^d + \epsilon \tag{6}$$

- y: dependent variable (target)
- x: independent variable (feature)
- $\beta_0, \beta_1, \ldots, \beta_d$: regression coefficients
- ϵ: error term

Transforming Polynomial Regression into Linear Regression

Regularization

Define new feature variables:

$$X_1 = x, \quad X_2 = x^2, \quad X_3 = x^3, \quad \ldots, \quad X_d = x^d$$

Rewritten as:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_d X_d + \epsilon \tag{7}$$

- Now, the model is linear in parameters β.
- We can apply standard linear regression techniques.

Transforming Polynomial Regression into Linear Regression

Regularization

Rewriting polynomial regression in matrix form:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{8}$$

Where:

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^d \\ 1 & x_2 & x_2^2 & \dots & x_2^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^d \end{bmatrix}$$

Transforming Polynomial Regression into Linear Regression

Regularization

Using Scikit-Learn to transform features and fit a linear model:

```
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

# Sample data
X = np.array([[1], [2], [3], [4], [5]])
y = np.array([2, 5, 10, 17, 26])  # Quadratic relationship

# Transform X to polynomial features (degree 2)
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)

# Fit linear regression on transformed features
model = LinearRegression()
model.fit(X_poly, y)
print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
```

Transforming Polynomial Regression into Linear Regression

Regularization

- Polynomial regression is linear in parameters, despite nonlinear terms.
- Transform features to apply linear regression techniques.
- Regularization (Ridge, Lasso) can help prevent overfitting.

Ridge Regression

- Ridge regression is a regularization technique where we change the function that is to be minimize;
- Reduce magnitude of regression coefficients by choosing a parameter λ and minimizing

$$\frac{1}{2N}\sum_{n=1}^{N}\left[h_\theta\left(x^{(n)}\right)-y^{(n)}\right]^2+\lambda\sum_{n=1}^{N}b_i^2 \tag{9}$$

- This change has the effect of encouraging the model to keep the weights b_j as small as possibile;
- The Ridge regression should only be used for determining model parameters using the training set. Once the model parameters have been determined the penalty term should be removed for prediction;
- What happens as λ increases?

Lasso Regression

- Lasso is short for *Least Absolute Shrinkage and Selection Operator*;
- Similar to ridge regression except we minimize

$$\frac{1}{2N}\sum_{n=1}^{N}\left[h_\theta\left(x^{(n)}\right)-y^{(n)}\right]^2+\lambda\sum_{n=1}^{N}|b_n| \tag{10}$$

- This function cannot be minimized analytically and so a variation on the gradient descent algorithm must be used;
- Lasso regression also has the effect of simplifying the model. It does this by setting the weights of unimportant features to zero. When there are a large number of features, Lasso can identify a relatively small subset of the features that form a good predictive model.

Lasso Regression: Geometrical Explanation

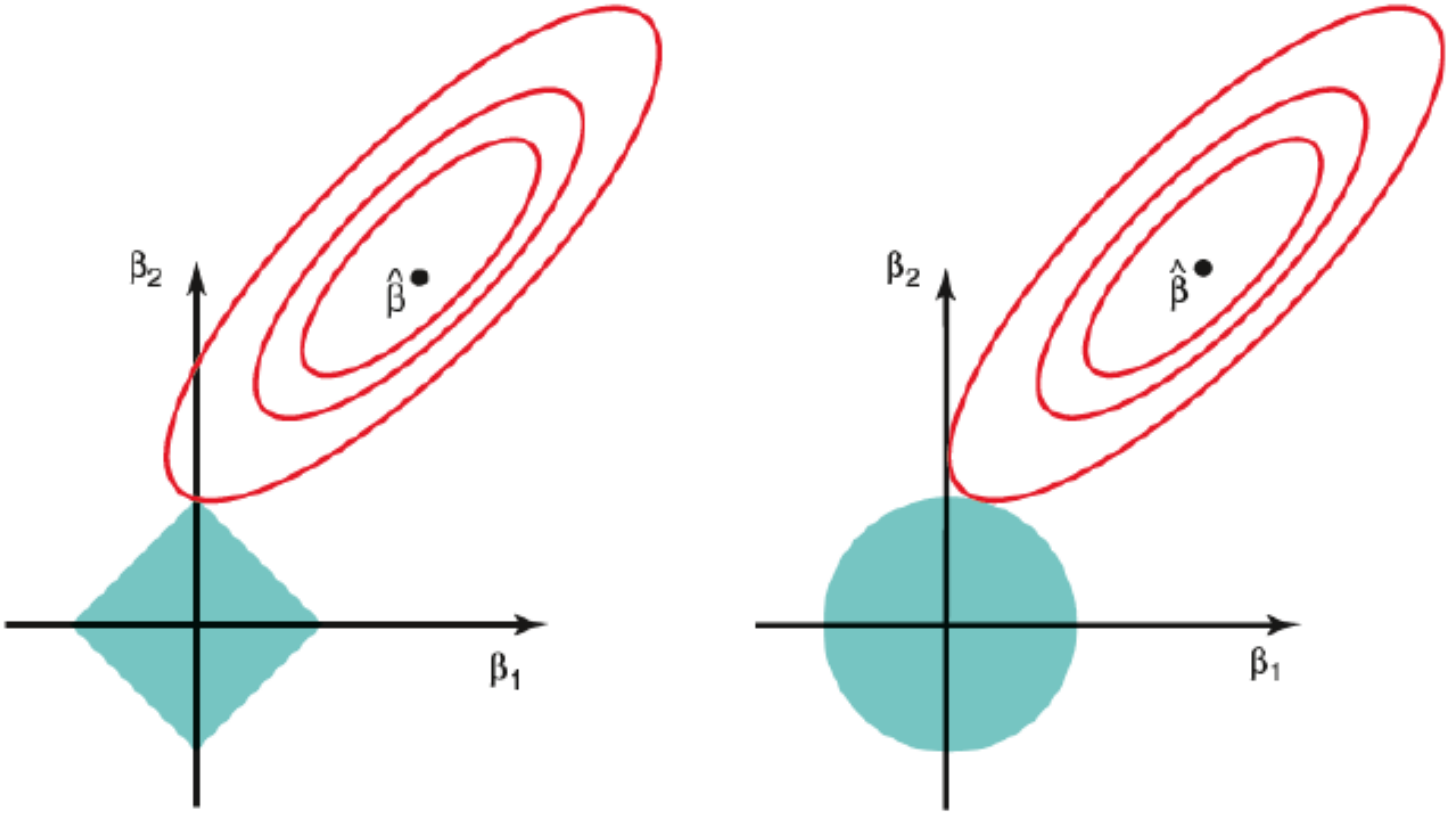


FIGURE 6.7. *Contours of the error and constraint functions for the lasso* (left) *and ridge regression* (right). *The solid blue areas are the constraint regions,* $|\beta_1| + |\beta_2| \leq s$ *and* $\beta_1^2 + \beta_2^2 \leq s$, *while the red ellipses are the contours of the RSS.*

Lasso Regression: Geometrical Explanation

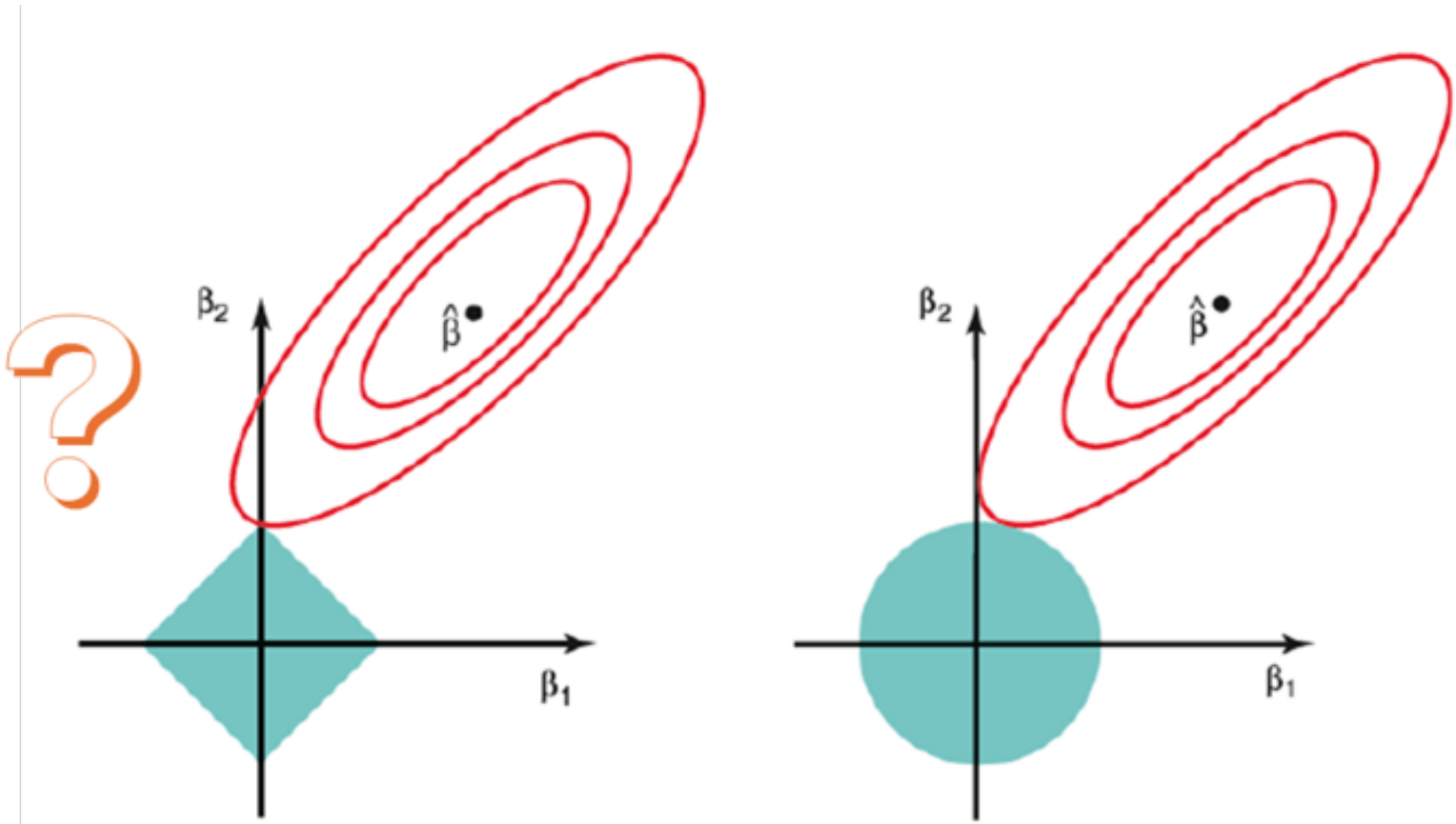


FIGURE 6.7. *Contours of the error and constraint functions for the lasso* (left) *and ridge regression* (right). *The solid blue areas are the constraint regions,* $|\beta_1| + |\beta_2| \leq s$ *and* $\beta_1^2 + \beta_2^2 \leq s$, *while the red ellipses are the contours of the RSS.*

L1 Norm Constraint as a Cross Polytope & Level Sets of the Objective Function

- **L1 Norm Constraint as a Cross Polytope:**
  - In 2D, the L1 norm constraint $\|\beta\|_1 \leq t$ (for some $t > 0$) forms a diamond shape (a square rotated by 45 degrees) with four corners at $(\pm t, 0)$ and $(0, \pm t)$.
  - In higher dimensions, this shape generalizes to a cross polytope with $2n$ corners in n-dimensional space.
- **Level Sets of the Objective Function:**
  - In many Lasso explanations, we visualize level sets (contours) of the unconstrained objective—often ellipses in the case of squared-error loss—expanding outward from a central point (e.g., the ordinary least squares solution).
  - These ellipses will eventually intersect the L1 diamond or polytope.

Corners Promote Sparsity & Why It May Seem Less Obvious in 2D

Corners Promote Sparsity:

- The "sparsity" effect arises because corners of the L1 region correspond to solutions where one or more coefficients are zero (i.e., the solution "hits" an axis).
- In higher dimensions, there are many more corners (since there are $2n$ vertices in an n-dimensional cross polytope), which increases the likelihood that the outward-expanding contours will meet the L1 ball at a corner, thereby setting some coefficients to zero.

Corners Promote Sparsity & Why It May Seem Less Obvious in 2D

Why It May Seem Less Obvious in 2D:

- In two dimensions, the diamond only has four vertices. Depending on the orientation of the contours (ellipses), it may not always appear as "corner-favoring"
- Even in 2D, as you vary the penalty parameter t, you can often see the solution snap to a corner (meaning one coordinate goes to zero).
- In higher dimensions, this effect is magnified by the large number of corners, making it more likely that some coefficients become exactly zero.

Elastic Net Regression (must use gradient descent)

- Middle ground between Ridge and Lasso
- Minimize

$$\frac{1}{2N}\sum_{n=1}^{N}\left[h_\theta\left(x^{(n)}\right)-y^{(n)}\right]^2+\lambda_1\sum_{n=1}^{N}b_n^2+\lambda_2\sum_{n=1}^{N}|b_n| \tag{11}$$

- In Lasso some weights are reduced to zero but others may be quite large. In Ridge, weights are small in magnitude but they are not reduced to zero. The idea underlying Elastic Net is that we may be able to get the best of both by making some weights zero while reducing the magnitude of the others.

Salary Vs Age Example: The Effect of Regularization

Age (years)	Salary ($)
25	135,000
55	260,000
27	105,000
35	220,000
60	240,000
65	265,000
45	270,000
40	300,000
50	265,000
30	105,000

We apply regularization to the model:

$$Y = a + b_1 X + b_2 X^2 + b_3 X^3 + b_4 X^4 + b_5 X^5 \tag{12}$$

where Y is salary and X is age.

Salary Vs Age Example: The Effect of Regularization

Data with Z-score scaling

Observ.	X	X^2	X^3	X^4	X^5
1	−1.290	−1.128	−0.988	−0.874	−0.782
2	0.836	0.778	0.693	0.592	0.486
3	−1.148	−1.046	−0.943	−0.850	−0.770
4	−0.581	−0.652	−0.684	−0.688	−0.672
5	1.191	1.235	1.247	1.230	1.191
6	1.545	1.731	1.901	2.048	2.174
7	0.128	−0.016	−0.146	−0.253	−0.333
8	−0.227	−0.354	−0.449	−0.511	−0.544
9	0.482	0.361	0.232	0.107	−0.004
10	−0.936	−0.910	−0.861	−0.803	−0.745

Salary Vs Age Example: The Effect of Regularization

Ridge Results, $\lambda = 0.02$ is similar to quadratic model

λ	a	b_1	b_2	b_3	b_4	b_5
0	216.5	−32,623	135,403	−215,493	155,315	−42,559
0.02	216.5	97.8	36.6	−8.5	35.0	−44.6
0.10	216.5	56.5	28.1	3.7	−15.1	−28.4

Salary Vs Age Example: The Effect of Regularization

Lasso Results, $\lambda = 1$ is similar to the quadratic model

λ	a	b_1	b_2	b_3	b_4	b_5
0	216.5	−32,623	135,403	−215,493	155,315	−42,559
0.02	216.5	−646.4	2,046.6	0.0	−3,351.0	2,007.9
0.1	216.5	355.4	0.0	−494.8	0.0	196.5
1	216.5	147.4	0.0	0.0	−99.3	0.0

Let's code ...