Machine Learning & Signals Learning

4 Model Characterization

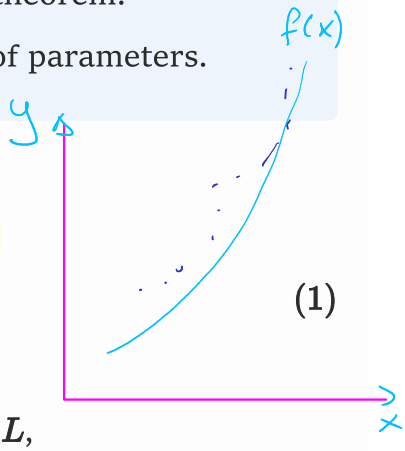
Goal:

- Interpret cross-validation results.
- Identify fundamental trade-offs and/or relations between:
  - Model complexity and generalization performance.
  - Model complexity and the size of dataset.

4.1 Uni-variate Polynomial Model

Goal:

- Extend a linear model "engine" to polynomial models. The polynomial model is very flexible, e.g. due to the Taylor expansion theorem.
- Provide an example of a model with variable number of parameters.



The L-degree uni-variate polynomial regression model is

$$\begin{aligned}\hat{y} = f(x;\mathbf{w}) &= w_0 + w_1 x + w_2 x^2 + \cdots + w_L x^L \\ &= \sum_{j=0}^{L} w_j x^j\end{aligned} \tag{1}$$

This problem is linear by the change of variables, $z_j = x^j,\ j = 0, \ldots, L$,

$$\mathbf{w}_{opt} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} \qquad \hat{y} = \sum_{j=0}^{L} w_j z_j \tag{2}$$

The corresponding prediction for the dataset $\{x_k, y_k\}_{k=1}^{M}$ can be easily written by using the matrix notation (also termed Vandermonde matrix),

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^L \\ 1 & x_2 & x_2^2 & \cdots & x_2^L \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_M & x_M^2 & \cdots & x_M^L \end{bmatrix} \tag{3}$$

The weights vector values are straightforward.

Hyper-parameter: A hyper-parameter is a parameter that defines the model structure or the learning process, and is set before training rather than learned from the data. In contrast to the model parameters $\mathbf{w}$, which are optimized during training, hyper-parameters are chosen by the designer (further discussed in Sec. 7.1.2).

The value of L in the polynomial model is an example of a hyper-parameter, as it determines the model complexity.

4.2 Generalization

Examples of questions of the significant importance:

- Is a model $\hat{y} = f(x; \mathbf{w})$ appropriate?
- How do we evaluate a model performance?
- How to select optimal hyper-parameter values, e.g. L of the polynomial model?

Goal: The goal is:

- Estimate some metrics of the model and understand the limitations on this estimate.
- Trial and error approach.
- Understand the limitations on the effectiveness of the model's ability to make inferences on unfamiliar data.

Let's assume that the dataset $(\mathbf{X}, \mathbf{y})$ are M samples drawn from some (unknown) joint probability distribution, $\mathcal{D}$. The theoretical model performance is given by some average model *metric* (not loss) over all (theoretically) possible points from $\mathcal{D}$,[1]

$$\bar{J}_{theory} = \lim_{M \to \infty} \frac{1}{M} \sum_{k=1}^{M} J(y_k, \hat{y}_k) \tag{4}$$

Generalization: The difference between:

- Performance metric over dataset that is used to train the model, $\bar{J}_M$.
- Theoretical performance metric $\bar{J}_{theory}$ from (4.4).

Better generalization means smaller difference between model performance and theoretical performance.

The problem is that the distribution $\mathcal{D}$ is unknown in most of the practical applications. Moreover, in practice, the value of M is (very) limited and $\bar{J}_M$ can significantly differ from $\bar{J}_{theory}$.

The generalization gap has two main sources:

- **Data-related**: limited dataset size M, sampling bias, noise in the data, and non-representative samples from $\mathcal{D}$.
- **Model-related**: model complexity mismatch (underfitting or overfitting), inappropriate model assumptions, and insufficient regularization.

In the following, methods to reduce both data-related and model-related gaps such that $\bar{J}_M \approx \bar{J}_{theory}$ will be discussed.

The generalization concept is illustrated in Fig. 4.1.

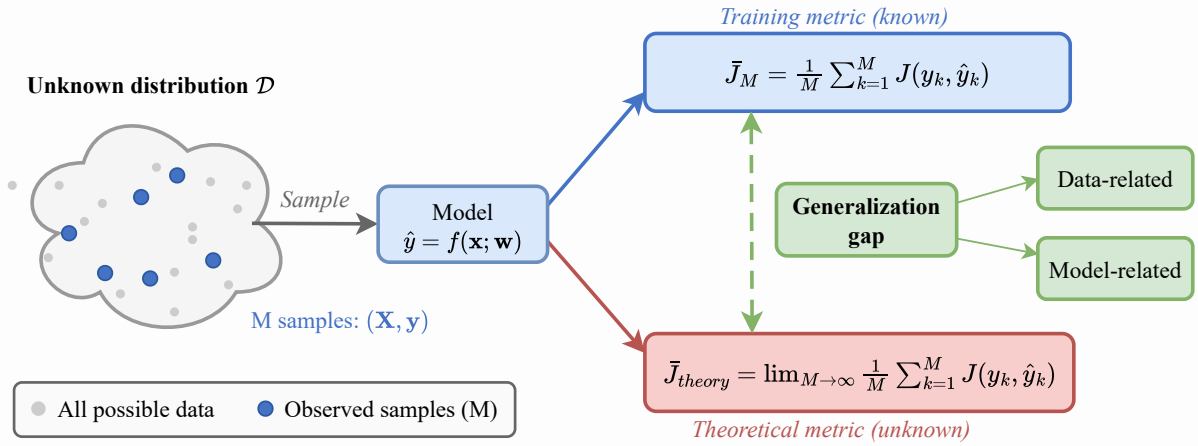


Figure 4.1: Generalization: the gap between the training metric $\bar{J}_M$ (computed on M observed samples) and the theoretical metric $\bar{J}_{theory}$ (over the entire unknown distribution $\mathcal{D}$ of possible inputs). The gap has data-related and model-related sources.

Table 4.1 summarizes the methods discussed in this chapter and their influence on the generalization gap. Each method primarily addresses either data-related or model-related sources of the gap.

Table 4.1: Chapter methods and their generalization influence.

Method	Data-related	Model-related
Cross-validation (Sec. 4.5)	✓	✓
Hyper-parameter tuning (Sec. 4.5)		✓
Normalization / Standardization (Sec. 4.6)	✓	

Dataset size increase (Sec. 4.7)	✓	
Regularization (Sec. 4.8)		✓

[1] Probabilistic formulation, $E_{\mathcal{D}}[J(y, \hat{y})]$

4.3 Bias-Variance Trade-Off

4.3.1 Bias and Variance

Noise deterministic function model The general model beneath the dataset model is

$$y_i = h(\mathbf{x}_i) + \epsilon_i, \tag{5}$$

where $h(\mathbf{x})$ is some unknown function and ϵ is some random noise with unknown distribution, zero-mean and variance of σ^2 ($\mathbb{E}[\epsilon] = 0, \mathbf{Var}[\epsilon] = \sigma^2$).

A model predicts outputs via $\hat{y}_i = f(\mathbf{x}_i)$ with error $e_i = \hat{y}_i - y_i$.

Bias The bias of the model is the expected difference between predictions and the true function,

$$\mathbf{Bias}[\hat{y}] = \mathbb{E}[\hat{y}] - \mathbb{E}[y] = \mathbb{E}[f(\mathbf{x})] - h(\mathbf{x}) \tag{6}$$

where the expectation is taken over all possible training samples.

For a specific dataset, the empirical bias can be estimated as:

$$\begin{aligned} \mathbf{Bias}_{\text{empirical}} = \bar{\mathbf{e}} &= \frac{1}{M}\sum_{i=1}^{M} e_i \\ &= \frac{1}{M}\sum_{i=1}^{M} \hat{y}_i - \frac{1}{M}\sum_{i=1}^{M} h(\mathbf{x}_i) \end{aligned} \tag{7}$$

Variance The variance of the model measures the variability of predictions across different training sets,

$$\mathbf{Var}[\hat{y}] = \mathbb{E}\Big[\big(\hat{y} - \mathbb{E}[\hat{y}]\big)^2\Big] \tag{8}$$

The bias and variance are illustrated conceptually in Fig. 4.4.

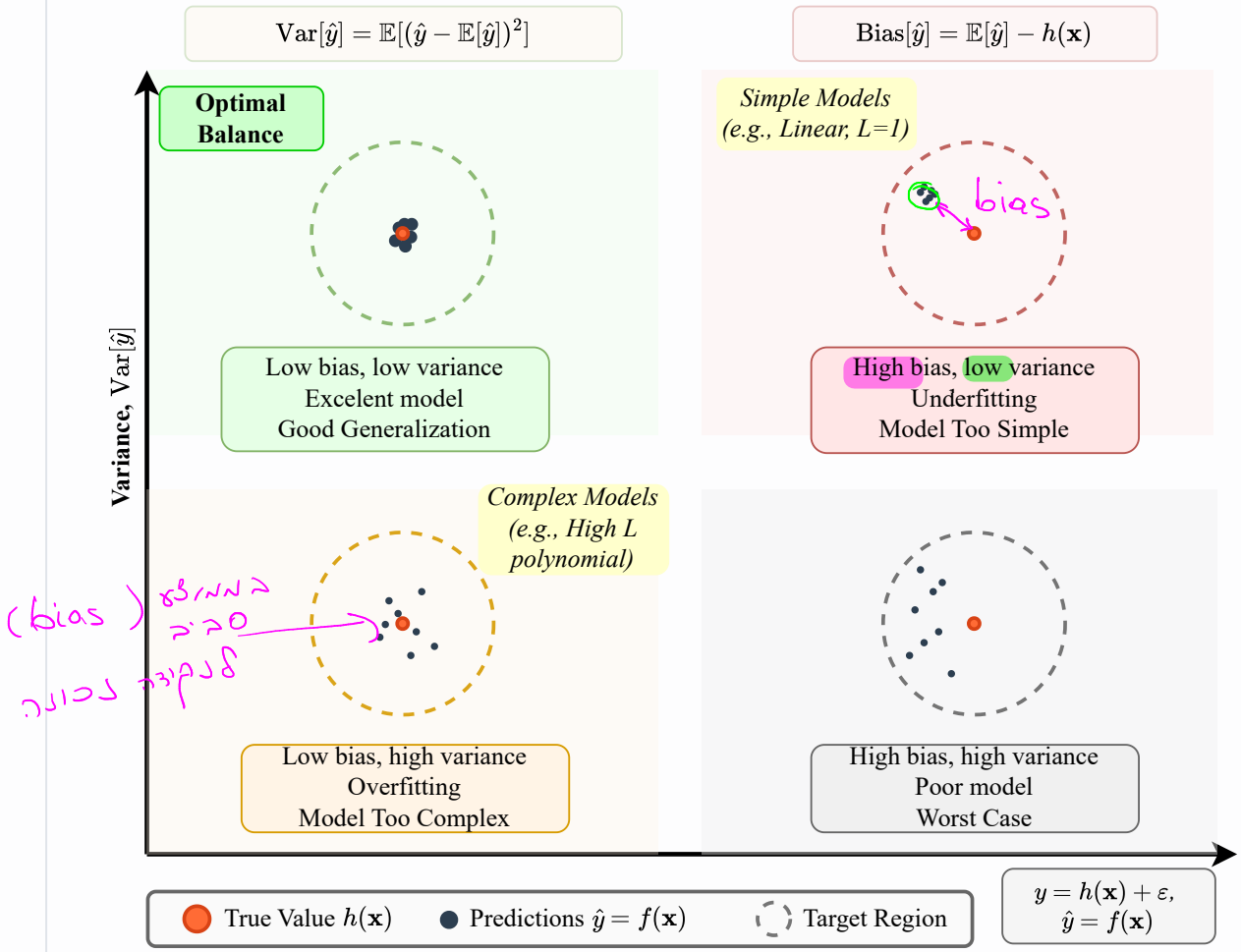


Figure 4.2: Illustration of bias and variance.

The Explicit Bias-Variance Trade-off Formula The core of model evaluation relates to understanding the total expected prediction error (MSE). For a new unseen point $\mathbf{x}$, the expected error is explicitly given by the sum of three fundamental components:

$$\mathbb{E}\left[(y - \hat{y})^2\right] = \left(\mathrm{Bias}[\hat{y}]\right)^2 + \mathrm{Var}\left[\hat{y}\right] + \sigma^2 \tag{9}$$

where:

- *Squared Bias* $\left(\mathrm{Bias}[\hat{y}]\right)^2$: Error introduced by approximating a real-world problem with a simplified model.
- *Variance* $\mathrm{Var}\left[\hat{y}\right]$: Error introduced by the model's sensitivity to small fluctuations or noise in the training set.
- *Irreducible Noise* σ^2: The inherent variance of the data itself that no model can capture.

The total prediction error consists of these three components (formally derived in Sec. 4.9). The inherent bias-variance trade-off is presented in Fig. 4.5. Underfitting is characterized by low variance and high bias, whereas overfitting is characterized by high variance and low bias. The best generalized model performance balances this inherent

bias-variance trade-off to minimize the overall expected error. However, note that some poorly chosen models can simultaneously exhibit both high bias and high variance.

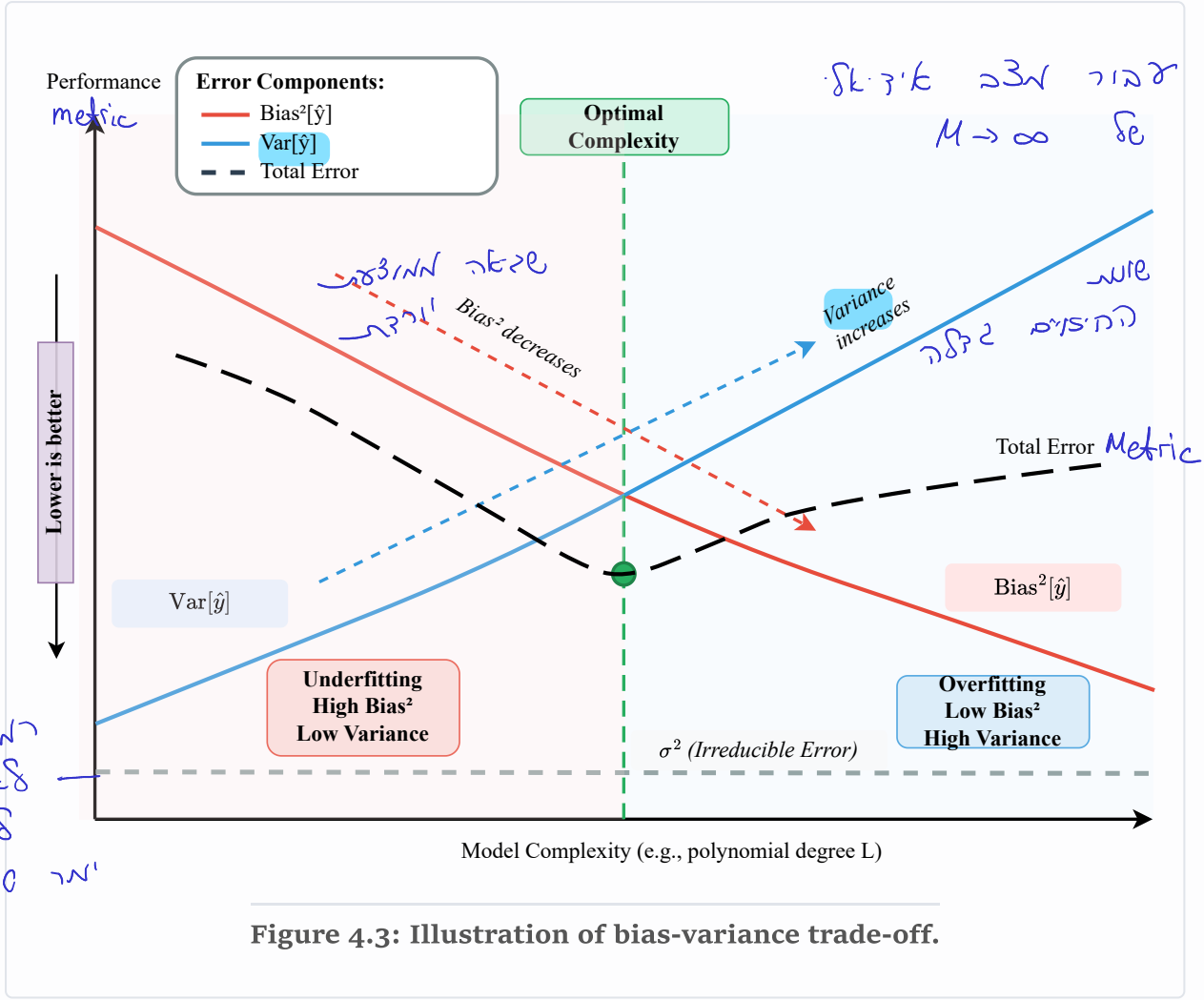


Figure 4.3: Illustration of bias-variance trade-off.

4.3.2 Overfitting and underfitting

> **Goal:** Understand underfitting and overfitting as two extremes of model complexity.

The numerical example of underfitting and overfitting trade-off from the polynomial model (Sec. 4.1) is presented in Fig. 4.3.

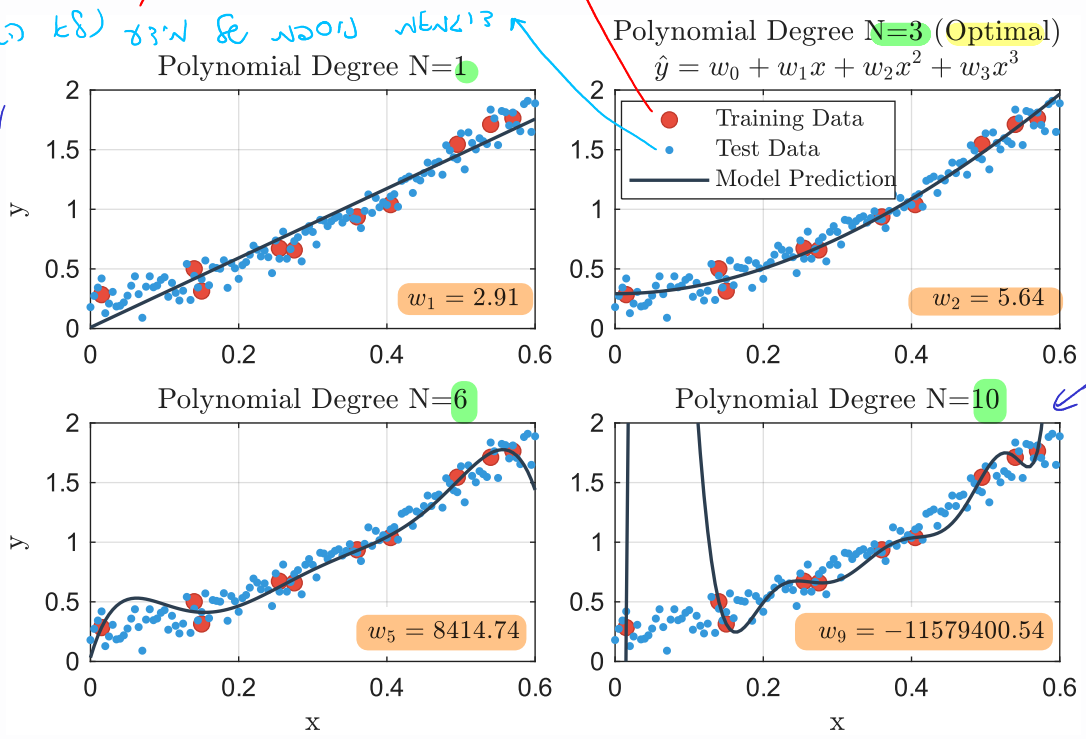


Figure 4.4: Overfitting and underfitting polynomial example.

To evaluate a model, the available dataset is split into a *training* subset (used to fit the model) and a *test* subset (used to evaluate it on unseen data). The formal splitting procedures are discussed in Sec. 4.5. The relationship between model complexity and performance on these two subsets reveals two failure modes.

Overfitting is when the model is too complex, i.e. have too many parameters.

- Too many parameters relative to the number of observations. Theoretically, we expect at least an order of magnitude more observations than parameters.
- Follow the training data very closely.
- Fail to generalize well to unseen data.

Significant performance difference between train and test data.

Underfitting happens when a model is too simple.

- Unable to capture the underlying pattern of the data and hence misses the trends in the data.
- Performs poorly on the training data and fail to generalize.

Poor performance on both train and test datasets.

Overfitting and underfitting are complementary and balancing between them is key to building robust machine learning models that perform well on new, unseen data, i.e. generalize well. This principle is illustrated in Fig. 4.2

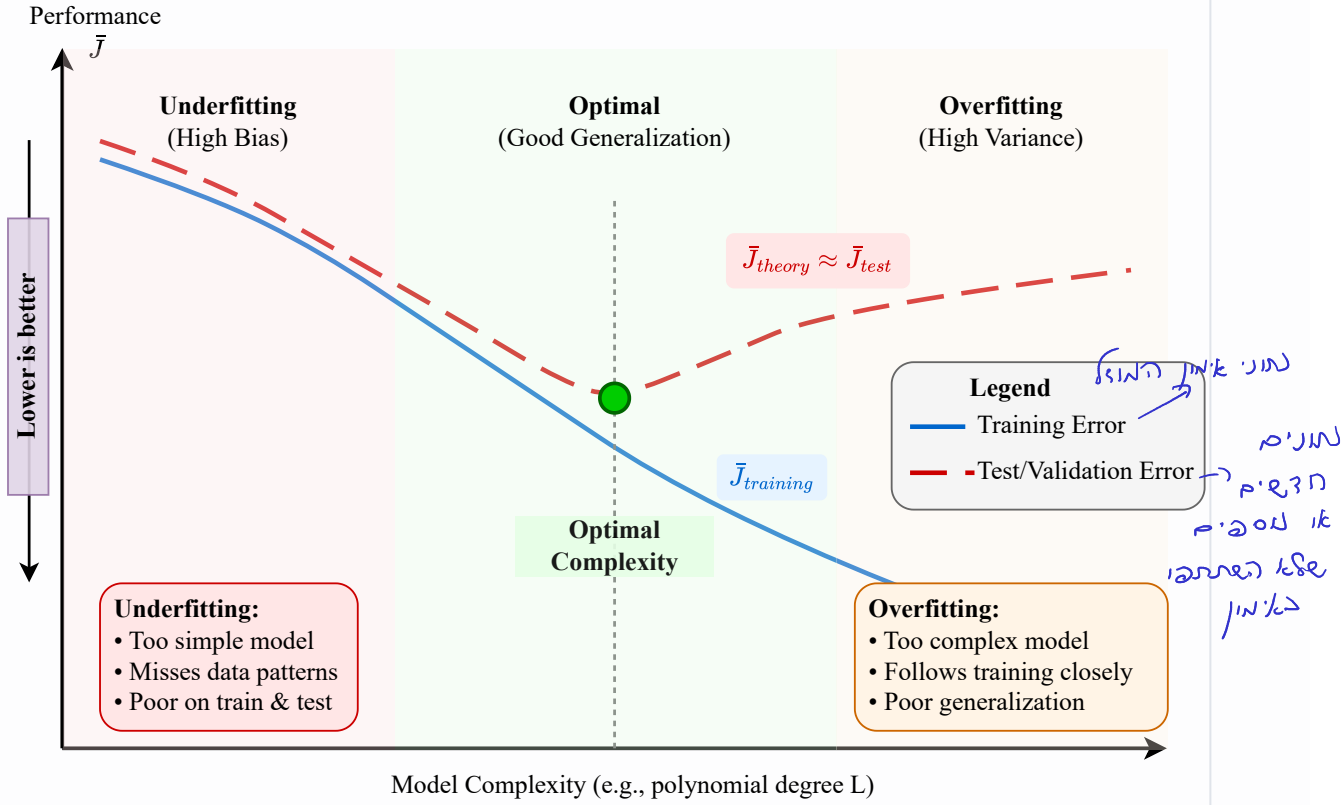


Figure 4.5: Increasing model complexity improves the prediction error for training data, but from a certain point of transitioning from underfitting to overfitting, it increases the error for test data.

4.4 Cross-validation

Goal: Trial and error approach to quantify generalization performance. The cross-validation is also termed performance assessment. Outcomes:

- Approximated generalization performance, $\bar{J}_M \approx \bar{J}_{theory}$.
- Hyper-parameters selection guideline.

The cross-validation methods are illustrated in Fig. 4.6.

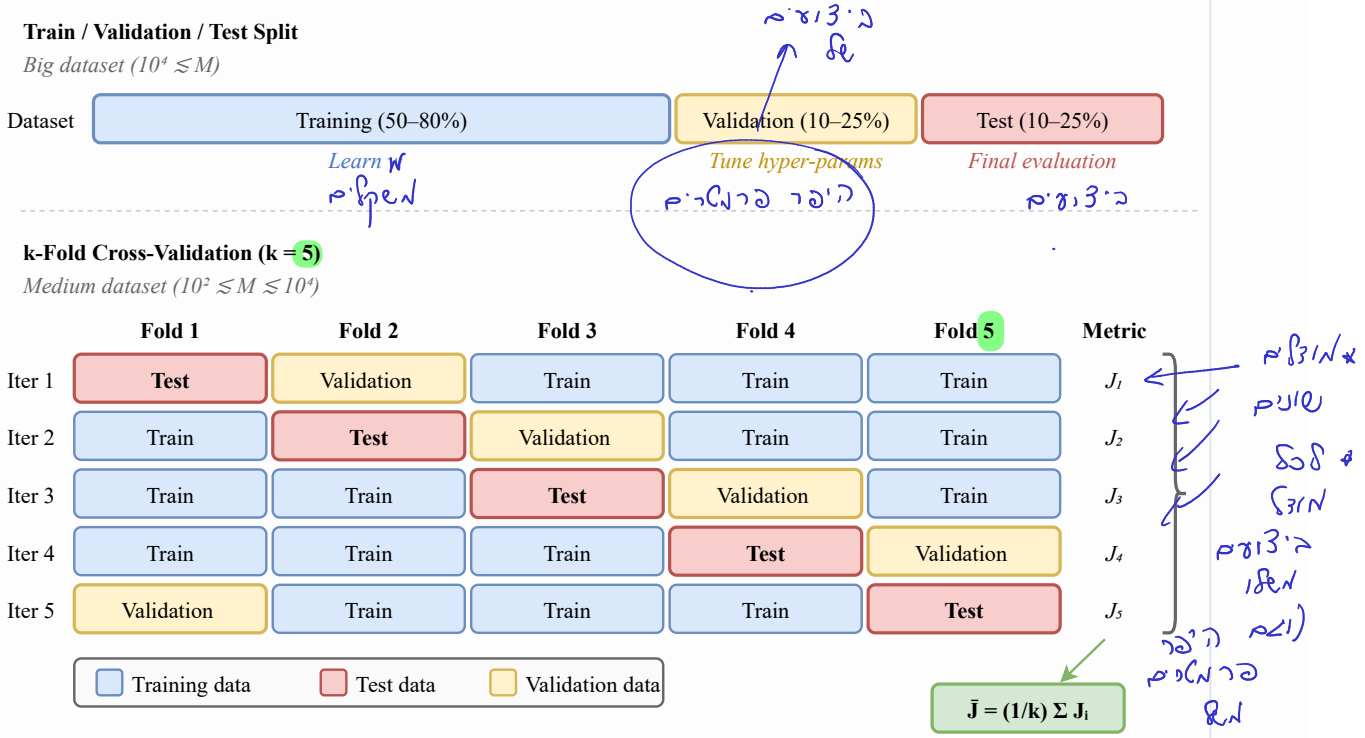


Figure 4.6: Cross-validation illustration: train/validation/test split for big datasets (top) and nested *k*-fold cross-validation for medium datasets (bottom).

Big dataset $\left(10^4 \lesssim M\right)$: train/validation/test

First step is to resample the dataset into the **random order**. Then, split it into three *distinctive* datasets:

- *Training dataset* (50-80%): used for learning of model parameters, e.g. weights $\mathbf{w}$.
- *Validation dataset* (10-25%): used for assessment of model hyper-parameters influence.
- *Test dataset* (10-25%): performance assessment that is supposed to be sufficiently close to $\bar{J}_{theory}$.

Hyper-parameter selection proceeds as follows:

1. For each candidate hyper-parameter value (e.g., polynomial degree L, regularization λ), train the model on the *training* set and evaluate the metric on the *validation* set.
2. Select the hyper-parameter value that yields the best validation metric.
3. Retrain the model on the combined training and validation sets using the selected hyper-parameter, and report the final performance on the held-out *test* set.

Medium datasets $\left(10^2 \lesssim M \lesssim 10^4\right)$: Nested k-fold Cross-Validation

> **Goal:** Simultaneously tune hyper-parameters (when applicable) and obtain an unbiased performance estimate.

The first step is to resample the dataset into a **random order**. After randomizing, we apply a nested loop procedure to separate two main objectives: hyper-parameter selection and performance assessment. Conflating these objectives into a single loop leads to optimistic performance estimates because the test data would indirectly influence model selection.

Nested cross-validation separates these objectives into two loops (Fig. 4.6):

- *Outer loop* (k_{out} folds for Performance Assessment): The available dataset is divided into k_{out} subsets (folds) of approximately equal size. In each of the k_{out} iterations, one fold (red in Fig. 4.6) is held out as the test set for unbiased performance evaluation. The remaining $k_{\text{out}} - 1$ folds form the outer training set.

- *Inner loop* (k_{in} folds for Hyper-parameter Tuning): Within each outer training set, a further k_{in}-fold split creates validation folds (yellow in Fig. 4.6). For each candidate hyper-parameter value (e.g., polynomial degree L, regularization parameter λ), the model is trained on the inner training folds (blue) and evaluated on the inner validation fold. The hyper-parameter value that maximizes the average inner validation metric is selected.

After the inner loop selects the best hyper-parameter, the model is retrained on the entire outer training set using that hyper-parameter, and its performance is evaluated on the outer test fold. The final performance estimate is the average over the k_{out} outer test metrics:

$$\bar{J} = \frac{1}{k_{\text{out}}} \sum_{i=1}^{k_{\text{out}}} J_i \tag{10}$$

Usually, k_{out} defaults to 5 or 10.

> The hyper-parameter selected may differ across outer folds. This is expected — each outer fold sees a slightly different training distribution. The purpose of nested CV is to produce an unbiased performance estimate, not a single set of hyper-parameters.

When hyper-parameter optimization is not required the inner loop is completely omitted. The procedure simply collapses into a standard, single-loop k-fold cross-validation. The model is trained on the $k_{\text{out}} - 1$ training folds and evaluated directly on the held-out test fold. This process is repeated k_{out} times, and the performance is averaged.

Very small datasets $\left(M \lesssim 10^2\right)$: Leave-One-Out Cross-Validation (LOOCV)

Uses k-fold with $k = M$, which means that each fold will contain only one data point.

> LOOCV is generally recommended only for very small datasets where k-fold would leave too few samples per fold. For medium and large datasets, 5- or 10-fold cross-

validation typically provides a better bias-variance trade-off in the performance estimate itself.

4.5 Normalization and Standardization

> **Goal:** Data pre-processing to reduce possible overfitting.

Values in different columns in $\mathbf{X}$ (feature columns $\tilde{\mathbf{x}}_j$) may be different by orders of magnitudes, i.e. $\|\tilde{\mathbf{x}}_i\| \gg \|\tilde{\mathbf{x}}_j\|$. This results in:

- Some columns have significantly higher influence on $\hat{\mathbf{y}}$.
- Numerical instabilities.

Standardization (z-score normalization)

Mapping all the input values such that they follow a distribution with zero mean and unit variance.

$$z_{std} = \frac{\mathbf{x} - \bar{\mathbf{x}}}{\sigma_{\mathbf{x}}}, \qquad (11)$$

where

$$\bar{\mathbf{x}} = \frac{1}{M}\sum_{j=1}^{M} x_j$$

$$\sigma_{\mathbf{x}}^2 = \frac{1}{M}\sum_{j=1}^{M} (x_j - \bar{\mathbf{x}})^2$$

Implementation steps:

1. On **train** dataset, evaluate $\bar{\mathbf{x}}$ and $\sigma_{\mathbf{x}}$.
2. Apply normalization on **train** dataset, using $\bar{\mathbf{x}}$ and $\sigma_{\mathbf{x}}$.
3. Apply normalization on **test** dataset, using **same** $\bar{\mathbf{x}}$ and $\sigma_{\mathbf{x}}$ (no recalculation).

When normalization is applied to $\mathbf{y}$, the output of the model is transformed back, $\hat{\mathbf{y}} = \hat{\mathbf{y}}_{std}\sigma_{\mathbf{y}} + \bar{\mathbf{y}}$.

Example: `zscore` command in Matlab and `sklearn.preprocessing.StandardScaler` in Python.

Normalization

Mapping all values of a feature to be in the range $[0, 1]$ by the transformation[2]

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \qquad (12)$$

Implementation steps for normalization are similar to standardization.

[2] In Python: `sklearn.preprocessing.MinMaxScaler`

Beware, normalization and standardization are used interchangeably.

Pros:

- Improves convergence speed of gradient-based optimization.
- Prevents features with large magnitudes from dominating distance-based models.
- Standardization of both $\mathbf{X}$ and $\mathbf{y}$ results in $w_0 = 0$ and removes the requirement for $\mathbf{1}$ column in $\mathbf{X}$.

Cons:

- Min-max normalization is sensitive to outliers.
- Scaling parameters must be computed on the training set and applied consistently to the test set.

4.6 Dataset Size

> **Goal:** Dataset size influence on underfitting-overfitting trade-off.
> Understanding when more data helps and when it does not.

By the law of large numbers, the generalization gap $\bar{J}_M - \bar{J}_{theory}$ shrinks as the number of samples M grows. However, the benefit of additional data depends on whether the model is overfitting or underfitting.

Overfitting (complex models) More data reduces variance. The training error rises (the model can no longer memorize every sample) while the test error drops — both converge toward the theoretical performance $\bar{J}_{theory}$. This is illustrated in Fig. 4.7.

Underfitting (simple models) More data does *not* help since the model lacks the capacity to capture the underlying pattern regardless of M. Both training and test errors remain high and close to each other.

Therefore, increasing the dataset size is a **data-related** method (see Sec. 4.2) that primarily reduces *variance*, not *bias*.

Beyond a certain dataset size, performance improvement may be almost negligible or non-existent.

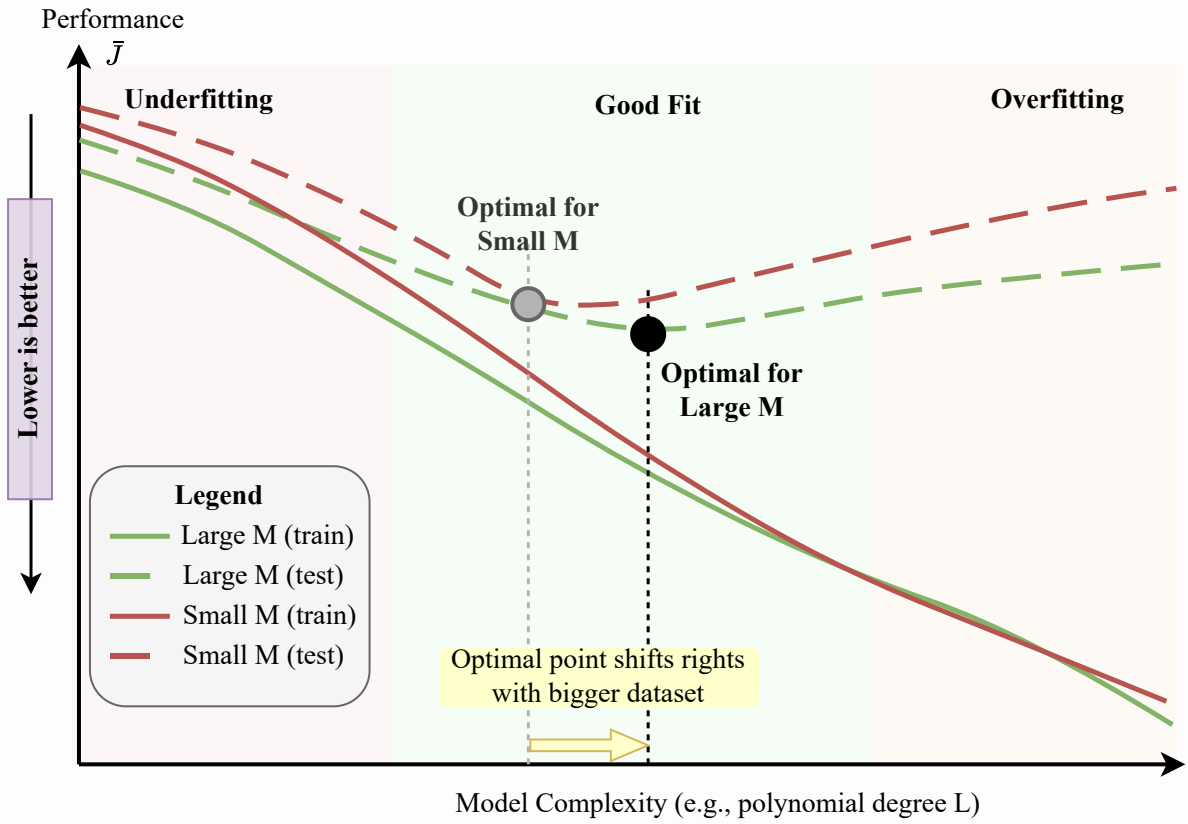


Figure 4.7: Increase in dataset size improves the over-complex model performance.

4.7 Regularization

> **Goal:** Loss function tweak that influences on underfitting-overfitting trade-off.

Regularization: Penalty to the loss function

$$\mathcal{L}_{\text{new}} = \mathcal{L} + \lambda g(\mathbf{w}) \tag{13}$$

where λ is termed *regularization parameter*.

L_2-regularization: Special case of

$$g(\mathbf{w}) = \frac{1}{2M} \| \mathbf{w} \|^2 = \frac{1}{2M} \sum_{i=1}^{N} w_i^2 \tag{14}$$

where vector of weights $\mathbf{w}$ does not include w_0 weight. Moreover, when normalization is used, no w_0 weight is needed. Two special cases are:

- $\lambda \to 0$ gets the original loss function (overfitting).
- $\lambda \to \infty$ makes loss function independent on $\mathbf{w}$ (underfitting).

Polynomial regression example The resulting coefficients w_i will be significantly higher for higher i (Fig. 4.3). Reducing them results in smooth $\hat{y}$ prediction. The illustration of λ influence is presented in Fig. 4.8.

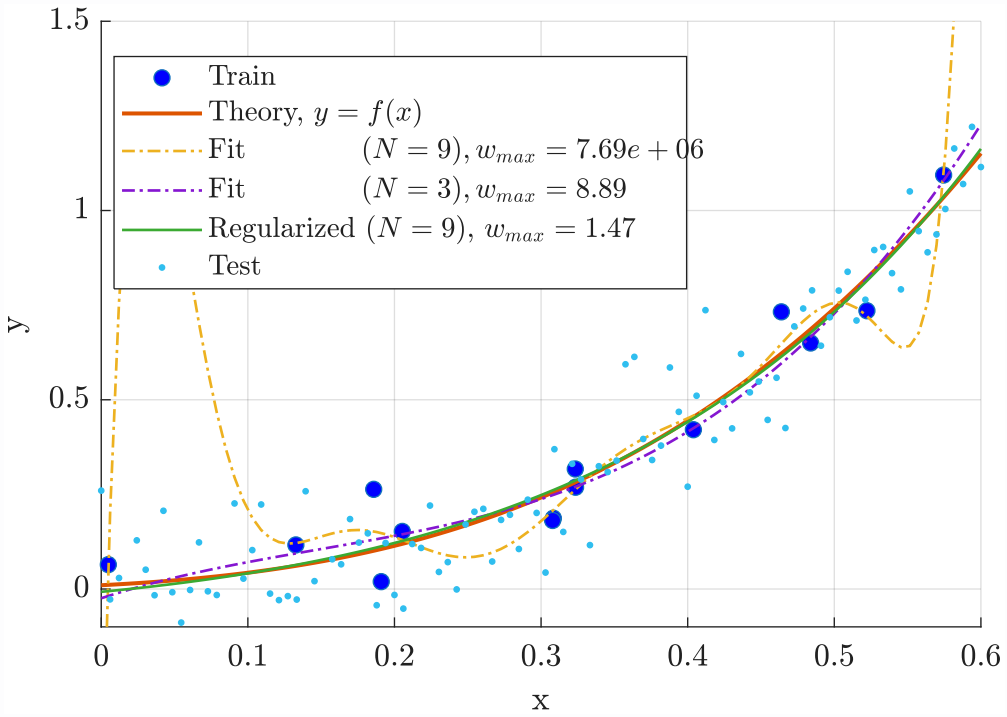


Figure 4.8: Illustration of regularization influence on the polynomial model.

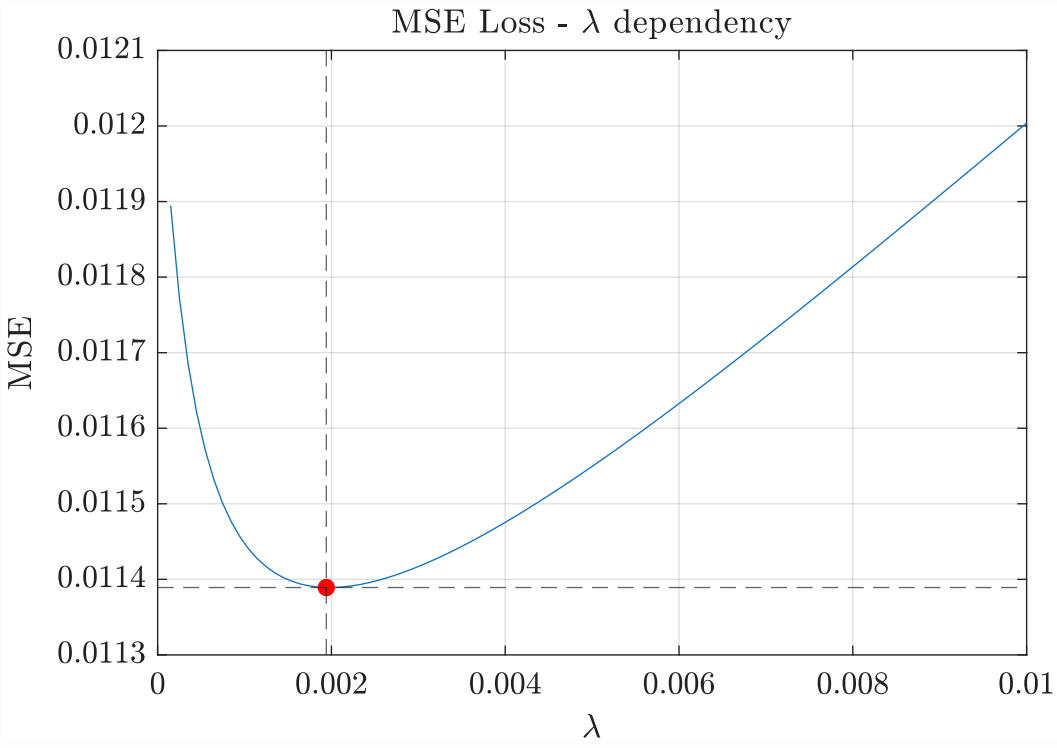


Figure 4.9: Dependence of MSE on λ for regularization in Fig. 4.8 (polynomial regression). Overfitting is on the left $(\lambda \to 0)$ and undefitting on the right

$(\lambda \to \infty)$.

The corresponding underfitting-overfitting λ-related trade-off is presented in Fig. 4.9.

4.7.1 Ridge Regression

Ridge regression: L_2-regularized linear regression.

$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= \frac{1}{2M}\|\mathbf{y}-\mathbf{X}\mathbf{w}\|^2 + \frac{\lambda}{2M}\underbrace{\|\mathbf{w}\|^2}_{\sum_{i=1}^N w_i^2} \\ &= \frac{1}{2M}(\mathbf{y}-\mathbf{X}\mathbf{w})^T(\mathbf{y}-\mathbf{X}\mathbf{w}) + \frac{\lambda}{2M}\mathbf{w}^T\mathbf{w}\end{aligned} \tag{15}$$

Derivative

$$\nabla\mathcal{L}(\mathbf{w}) = \frac{1}{M}\left(-\mathbf{X}^T(\mathbf{y}-\mathbf{X}\mathbf{w}) + \lambda\mathbf{w}\right) = 0 \tag{16}$$

Solution

$$\begin{aligned}-\mathbf{X}^T(\mathbf{y}-\mathbf{X}\mathbf{w}) + \lambda\mathbf{w} &= -\mathbf{X}^T\mathbf{y} + \mathbf{X}^T\mathbf{X}\mathbf{w} + \lambda\mathbf{w} = 0 \\ \mathbf{X}^T\mathbf{y} &= \mathbf{X}^T\mathbf{X}\mathbf{w} + \lambda\mathbf{w}\end{aligned}$$

Finally, the regularized weights are given by

$$\mathbf{w} = \left(\lambda\mathbf{I} + \mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} \tag{17}$$

Can be viewed as special case of linear regression, of the form

$$\begin{aligned}\tilde{y} &= \tilde{\mathbf{X}}\mathbf{w} \\ \begin{bmatrix}\mathbf{y}\\0\\\vdots\\0\end{bmatrix} &= \begin{bmatrix}- & \mathbf{X} & - \\ \sqrt{\lambda} & & 0 \\ & \ddots & \\ 0 & & \sqrt{\lambda}\end{bmatrix}\begin{bmatrix}w_1\\\vdots\\w_N\end{bmatrix} + w_0\end{aligned} \tag{18}$$

Slope interpretation Higher value of λ reduces (mostly) the highest slopes (weights w_i) $\Rightarrow$ the dependency on the parameters with these weights is reduced.

Eigenvalues interpretation This calculation limits the smallest eigenvalues to $> \frac{1}{\lambda}$ and thus improve the numerical stability.

Bias-Variance Trade-off The bias-variance trade-off for ridge regression is illustrated in Fig. 4.10.

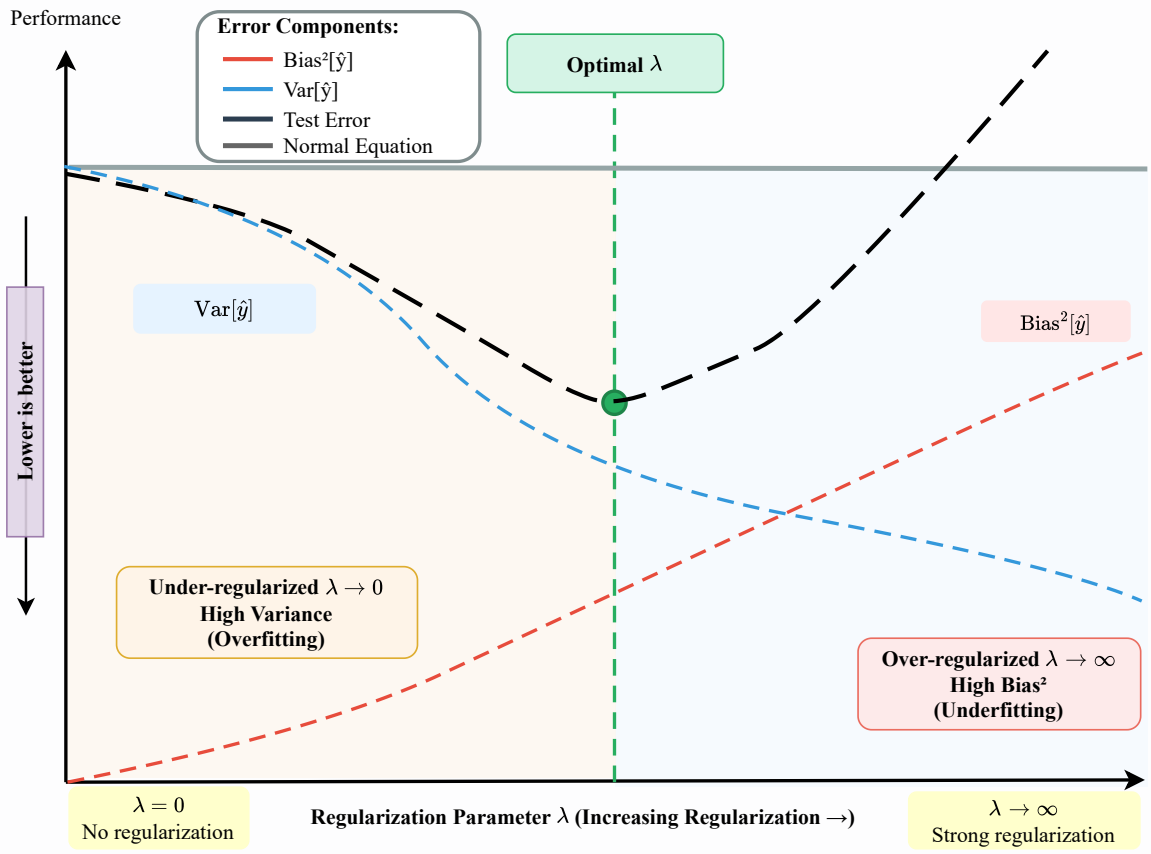


Figure 4.10: Regularization trade-off for ridge regression: Test error $\bar{J}_{\text{test}}$ is minimized at optimal λ. Note, the trends are opposite to Fig. 4.5.

GD Solution

Tailored GD (Sec. 3.4) solution is

$$\begin{aligned} \mathbf{w}_{n+1} &= \mathbf{w}_n - \alpha \nabla \mathcal{L}(\mathbf{w}) \\ &= \mathbf{w}_n - \frac{\alpha}{M}\left[\mathbf{X}^T(\mathbf{X}\mathbf{w}_n - \mathbf{y}) + \lambda \mathbf{w}_n\right] \\ &= \mathbf{w}_n\left(1 - \frac{\alpha}{M}\lambda\right) - \frac{\alpha}{M}\mathbf{X}^T(\mathbf{X}\mathbf{w}_n - \mathbf{y}) \end{aligned} \tag{19}$$

The α value is typically chosen such that

$$0.99 \gtrsim \left(1 - \frac{\alpha}{M}\lambda\right) \gtrsim 0.95 \tag{20}$$

4.8 Bias-Variance Decomposition (*)

> **Goal:** Discussion of inherent underfitting and overfitting trade-off in (??).

The expected prediction error (MSE), taken over all possible training sets and noise realizations, decomposes into three irreducible components:

$$\mathbb{E}\left[(\hat{y} - y)^2\right] = \left(\mathbb{E}\left[\hat{y}\right] - h(\mathbf{x})\right)^2 + \mathrm{Var}\left[\hat{y}\right] + \sigma^2 \tag{21}$$

where $\left(\mathrm{Bias}[\hat{y}]\right)^2 = \left(\mathbb{E}\left[\hat{y}\right] - h(\mathbf{x})\right)^2$.

Proof. Starting from:

$$\mathrm{MSE} = \mathbb{E}\left[(\hat{y} - y)^2\right] = \mathbb{E}\left[\hat{y}^2\right] - 2\mathbb{E}\left[y\hat{y}\right] + \mathbb{E}\left[y^2\right] \tag{22}$$

We evaluate each term separately. For the first term, by the variance identity:

$$\mathbb{E}\left[\hat{y}^2\right] = \mathrm{Var}\left[\hat{y}\right] + \mathbb{E}^2[\hat{y}] \tag{23}$$

For the cross-term, since $y = h(\mathbf{x}) + \epsilon$ and ϵ is independent of $\hat{y}$:

$$\mathbb{E}\left[y\hat{y}\right] = \mathbb{E}\left[(h(\mathbf{x}) + \epsilon)\hat{y}\right] = h(\mathbf{x})\mathbb{E}\left[\hat{y}\right] + \cancelto{0}{\mathbb{E}[\epsilon]}\mathbb{E}\left[\hat{y}\right] = h(\mathbf{x})\mathbb{E}\left[\hat{y}\right] \tag{24}$$

For the last term:

$$\mathbb{E}\left[y^2\right] = \mathbb{E}\left[(h(\mathbf{x}) + \epsilon)^2\right] = h^2(\mathbf{x}) + 2h(\mathbf{x})\cancelto{0}{\mathbb{E}[\epsilon]} + \mathbb{E}\left[\epsilon^2\right] = h^2(\mathbf{x}) + \sigma^2 \tag{25}$$

Combining all three terms:

$$\begin{aligned}\mathbb{E}\left[(\hat{y} - y)^2\right] &= \mathrm{Var}\left[\hat{y}\right] + \mathbb{E}^2[\hat{y}] - 2h(\mathbf{x})\mathbb{E}\left[\hat{y}\right] + h^2(\mathbf{x}) + \sigma^2 \\ &= \left(\mathbb{E}\left[\hat{y}\right] - h(\mathbf{x})\right)^2 + \mathrm{Var}\left[\hat{y}\right] + \sigma^2\end{aligned} \tag{26}$$

Empirical estimate For a specific dataset of M samples, the MSE can be similarly decomposed:

$$\mathrm{MSE} = \frac{1}{M}\sum_{i=1}^{M}(\hat{y}_i - y_i)^2 = \mathrm{bias}^2_{\mathrm{emp}} + \mathrm{variance}_{\mathrm{emp}} + \mathrm{noise}_{\mathrm{emp}} \tag{27}$$

where

$$\mathrm{bias}^2_{\mathrm{emp}} = \left(\bar{\hat{y}} - \bar{h}\right)^2, \qquad \bar{\hat{y}} = \frac{1}{M}\sum_{i=1}^{M}\hat{y}_i, \quad \bar{h} = \frac{1}{M}\sum_{i=1}^{M}h(\mathbf{x}_i) \tag{28}$$

$$\mathrm{variance}_{\mathrm{emp}} = \frac{1}{M}\sum_{i=1}^{M}(\hat{y}_i - \bar{\hat{y}})^2 \tag{29}$$

$$\mathrm{noise}_{\mathrm{emp}} = \frac{1}{M}\sum_{i=1}^{M}\epsilon_i^2 \tag{30}$$

4.9 Summary

- **Overfitting and underfitting** are two extremes of model complexity: overfitting follows training data too closely, underfitting fails to capture the underlying pattern.

• **Bias-variance trade-off** provides the theoretical framework explaining why overfitting (high variance, low bias) and underfitting (high bias, low variance) occur.

• **Cross-validation** mitigates data-related issues by evaluating model performance across different data splits, and guides model-related decisions such as hyper-parameter selection.

• **Hyper-parameter tuning** (e.g., polynomial degree L, regularization λ) adjusts model complexity to balance underfitting and overfitting.

• **Normalization and standardization** are data pre-processing steps that reduce the influence of feature scale imbalance and improve numerical stability.

• **Increasing dataset size** directly reduces the data-related gap by making $\bar{J}_M$ closer to $\bar{J}_{theory}$, and reduce overfitting in complex models.

• **Regularization** constrains model complexity via the penalty parameter λ, controlling the bias-variance trade-off from the model side.