Python Data Products

Course 1: Basics

Lecture: Validation

Learning objectives

In this lecture we will...

- Introduce the concept of the **validation set**
- Explain the relationship between model **parameters** and **hyperparameters**
- Introduce the **training → validation → test** pipeline

Recap…

In the last few lectures we saw…

- How a **training set** can be used to evaluate model performance on seen data
- How a **test set** can be used to estimate **generalization performance**
- How we can use a **regularizer** to mitigate overfitting

Recap…

In particular, our **regularizer** "trades-off" between model accuracy and model complexity

$$\underbrace{\frac{1}{N}\sum_i (y_i - X_i \cdot \theta)^2}_{\text{MSE}} + \lambda \underbrace{\sum_k \theta_k^2}_{\text{regularizer}}$$

- We want a value of our regularization parameter that balances model accuracy (low MSE) with complexity (low sum of squared parameters)

Recap…

In particular, our **regularizer** "trades-off" between model accuracy and model complexity

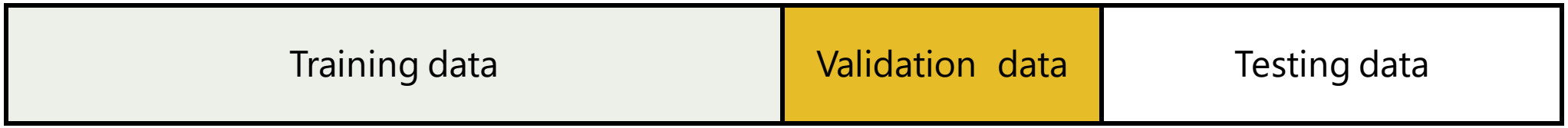
- If we only cared about **training error,** we'd always select the smallest possible value of lambda (i.e., lambda = 0)
- We could tune against our **test set,** but that would mean looking at the test set many times (which would be cheating!)

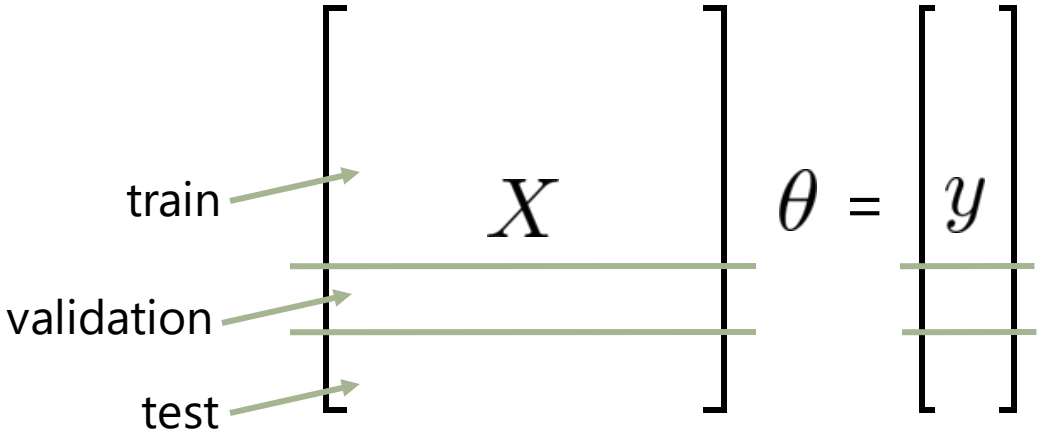
Recap…

In particular, our **regularizer** "trades-off" between model accuracy and model complexity

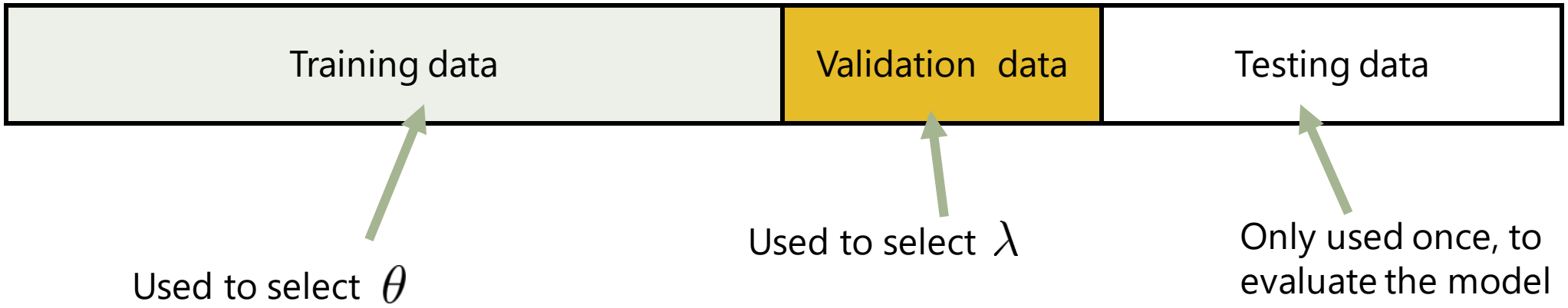
- So, we need a third partition of our data, which is similar to the test set, but which can be used to select hyperparameters like lambda
- This set is called the **validation set**

Training and test sets





Training and test sets



Summary of concepts

- We showed how a **validation set** can be used to tune parameters (or "hyperparameters") that cannot be selected using the training set (or the test set)
- In the following lecture, we'll explore more how this set can be used to optimize model performance