# Practical ML for Engineers

Mathematical Foundations of Supervised Learning

# Learning Objectives

1. Master the notation used in ML
2. Know the different **ingredients** of supervised learning
3. Understand supervised learning as an optimization problem

**ML = Model space + Loss function (+ regularization) + optimization**

**Supervised Learning Setup**

We have a (labelled) dataset of input-output examples

$$\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)}) | i = 1, \dots, n\}$$

and we want to find a **model** $f$ that, given an unseen data pair $(x, y)$, performs well at predicting $y$:

$$f(x) \approx y$$

**⚠ Problems**

1. We have **no idea what $f$ should look like** (modelization)
2. We **only have examples** (from the past), but no idea about future $x$ we might see (generalization)

# Model Space and Parameters

- To solve problem nr. 1, and make the model search tractable, we restrict our search to a given **model space** $\mathcal{M}$.
- The learning task is then to **find the best** $f$ in $\mathcal{M}$.

- In most cases, our models are parametrized by a finite set of parameters

$$\boldsymbol{\theta} = \left(\theta_1, \ldots, \theta_p\right) \in \boldsymbol{\Theta},$$

that belong to a **parameter space** $\boldsymbol{\Theta}$.

# Example: Linear Regression Models

### 1D Linear regression model

Model space is the set of all linear functions of $x$:

$$\mathcal{M} = \{ f(x) = a \cdot x + b \quad | \quad a, b \in \mathbb{R} \}.$$

Since the model is completely parametrized by the slope $a$ and the intercept $b$, we can bundle these parameters in a 2d vector, so the parameter space is $\mathbb{R}^2$:

$$(a, b) =: \boldsymbol{\theta} \in \Theta := \mathbb{R}^2.$$

For a given paremeter $\boldsymbol{\theta} = (a, b)$, the corresponding model is

$$f_{\boldsymbol{\theta}}(x) = a \cdot x + b$$

### 1D Polynomial Regression Model

Model space is the set of all polynomial functions of $x$, up to degree $d$:

$$\mathcal{M} = \{ f(x) = a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_d \cdot x^d + b \quad | \quad a1, \ldots, a_d, b \in \mathbb{R} \}.$$

The model is fully parametrize by the $(d+1)$-dimensional vector:

$$(b, a_1, \ldots, a_d) =: \boldsymbol{\theta} \in \Theta := \mathbb{R}^{d+1}.$$

# Model Space: Why bother?

> **Important**
>
> The choice of a **model space** $\mathcal{M}$ affects subsequent learning in several major ways:
>
> - by restricting the search space, it makes the learning **tractable**
> - restricting the search space means that we are also **restricting the class of data that we are allowed to fit** (bias)
> - this can, in turn, allow us to incorporate **prior knowledge**
> - dimension of the parameter space (complexity) governs the **speed of the learning process** (curse of dimensionality)

## Examples

| Model | Functional Form | Parameters | Dimension |
|---|---|---|---|
| Linear | $f_\theta(x) = ax + b$ | $\theta \in \mathbb{R}^2$ | $2$ |
| Polynomial (degree $p$) | $f_\theta(x) = \sum_{j=0}^{p} \theta_j x^j$ | $\theta \in \mathbb{R}^{p+1}$ | $p+1$ |
| Neural Network | $\dots$ | Weights $W_i$ | $\dots$ |

# Model Space: Influence of complexity

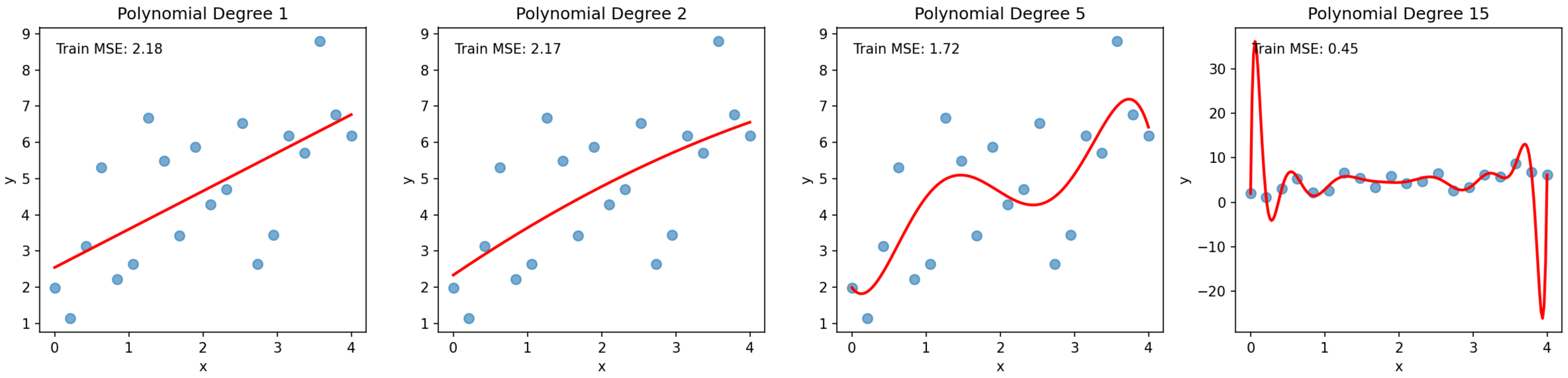
> A complex enough model can fit anything.

**Example: Polynomial regression**

$$f_\theta(x) = \theta_1 \cdot x + \theta_2 x^2 + \ldots + \theta_n x^n + \theta_{n+1}$$

**Bias-Variance Trade-off**

- **Simple models** (small $\mathcal{M}$): High bias (has difficulty fitting the data), low variance (low sensitivity to noise)
- **Complex models** (large $\mathcal{M}$): Low bias (can fit any data), high variance (too sensitive to data noise)




Will learn later how to **tune model complexity**.

**ML = Model space + Loss function (+ regularization) + optimization**

# Loss Functions

## Idea: Quantify how “wrong” our predictions are

- For each example $(\boldsymbol{x}, y)$ in the training dataset, we want to know how good our prediction $\hat{y} = f(x)$ is.
- Then pick the model $f$ that gives the best predictions.

| Features $x$ | | Target $y$ | | Prediction $\hat{y}$ |
|---|---|---|---|---|
| People in Office (Feature 1) $x_1$ | Salary (Feature 2) $x_2$ | Worked Minutes Week (Target Variable) | | Worked Minutes Week (Target Variable) |
| 4 | 4300 € | 2220 | | 2588 |
| 12 | 2700 € | 1800 | $\overset{?}{\approx}$ | 1644 |
| 5 | 3100 € | 1920 | | 1870 |

$\mathcal{D}_{\text{train}}$

- For a parametrized model $f_\theta$, corresponds to choosing the $\theta$ that minimizes an “aggregated” loss **on the training dataset**.

# Loss Functions

- Tells us, for a given prediction $\hat{y}$ how far we are from the true value $y$. - Provides a single number to optimize - Different cost functions emphasize different aspects

**L2 Loss**

$$L(y, \hat{y}) = (y - \hat{y})^2$$

- Penalizes large errors more than small ones
- Influenced by outliers
- Most common loss function for regression

**L1 Loss**

$$L(y, \hat{y}) = |y - \hat{y}|$$

- More robust to outliers
- Less statistically well-behaved

# Loss Functions for Learning

## Learning = Finding the best point in parameter space

**Central Idea: Empirical Risk Minimization**

- can sum all losses $L(y^{(i)}, \hat{y}^{(i)})$ over the training dataset
- for each possible value of the parameter $\boldsymbol{\theta}$ this give a measure of the “quality” of the model $f_{\boldsymbol{\theta}}$

$$\mathcal{J}(\boldsymbol{\theta}) = \sum_{i=1}^{n} L(y^{(i)}, f_{\boldsymbol{\theta}}(\boldsymbol{x})^{(i)})$$

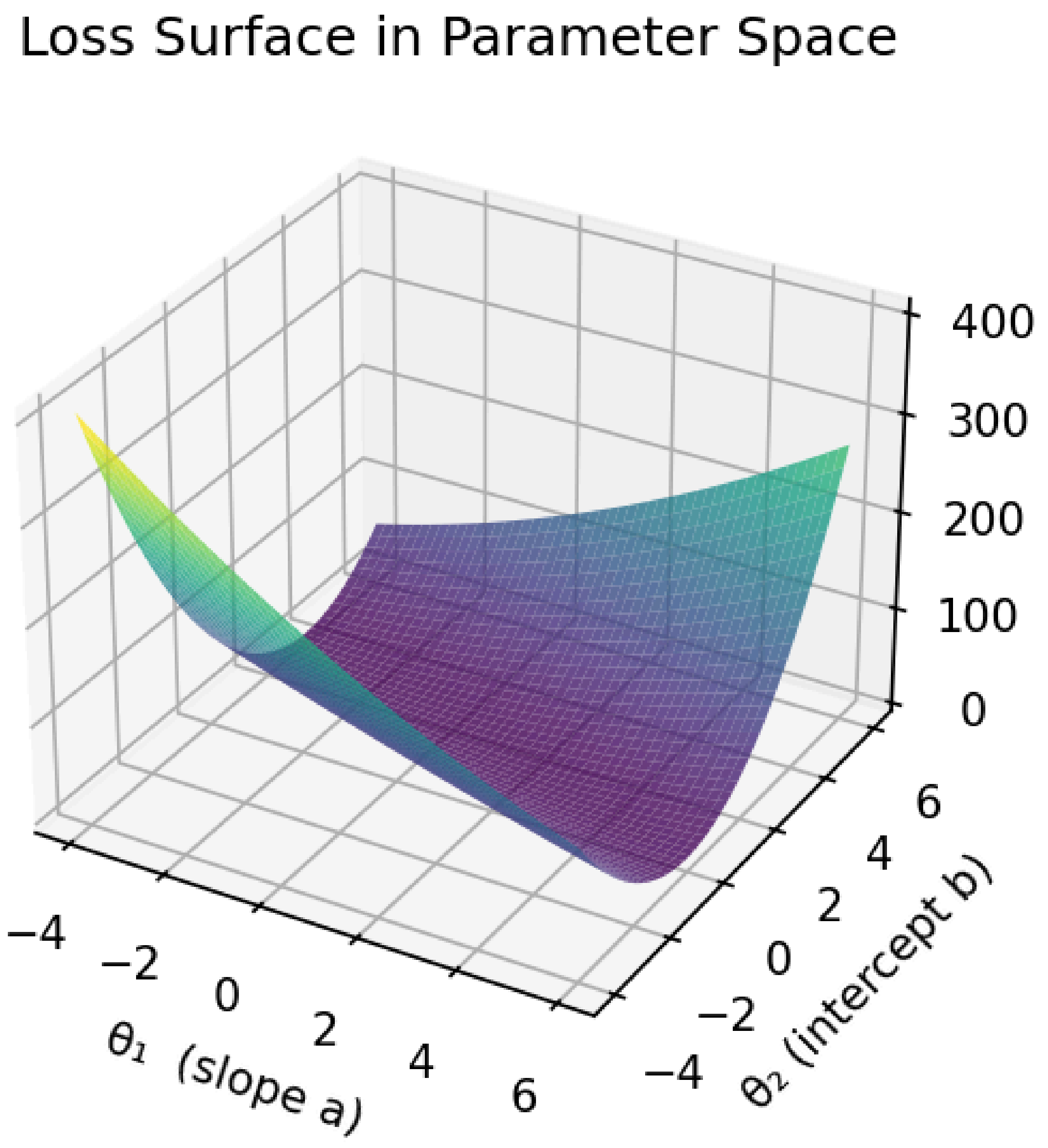
- tells us how well the model $f_{\boldsymbol{\theta}}$ fits the data for a given $\boldsymbol{\theta}$
- best model is the one with the smallest loss (smallest risk)

$$\boldsymbol{\theta}^* = \arg\min_{\theta \in \Theta} J(\boldsymbol{\theta})$$

# Empirical Risk Minimization: Worked Example

## 1D Linear regression

- **Model space**: All linear functions of the form $a \cdot x + b$
- **Parameters**: $\theta = (a, b)$
- Each $(a, b) = (\theta_1, \theta_2)$ pair defines one specific line



# Interactive Linear Regression & Loss Surface

### Parameters

**$\theta_1$ (Intercept):**

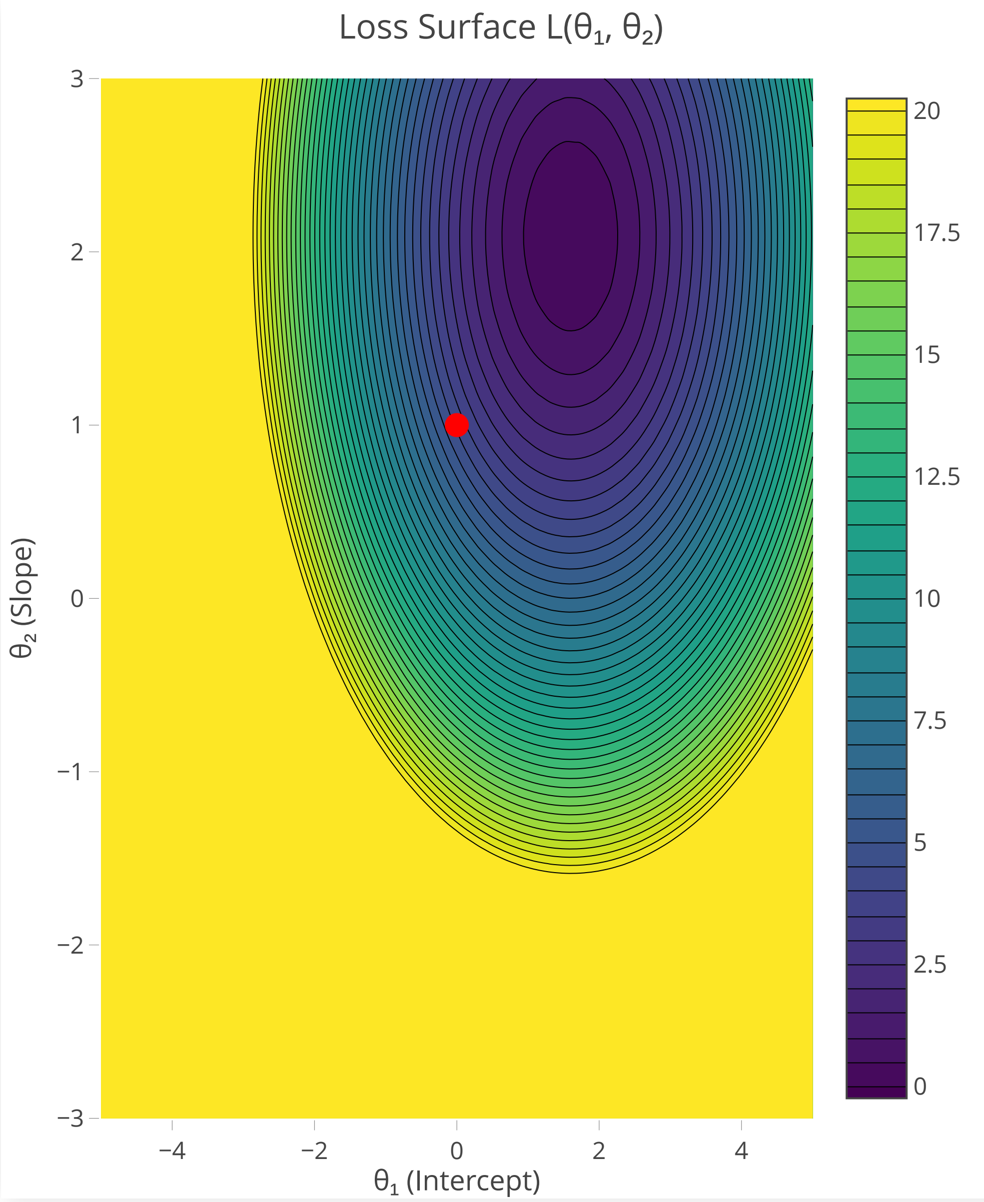
0.0

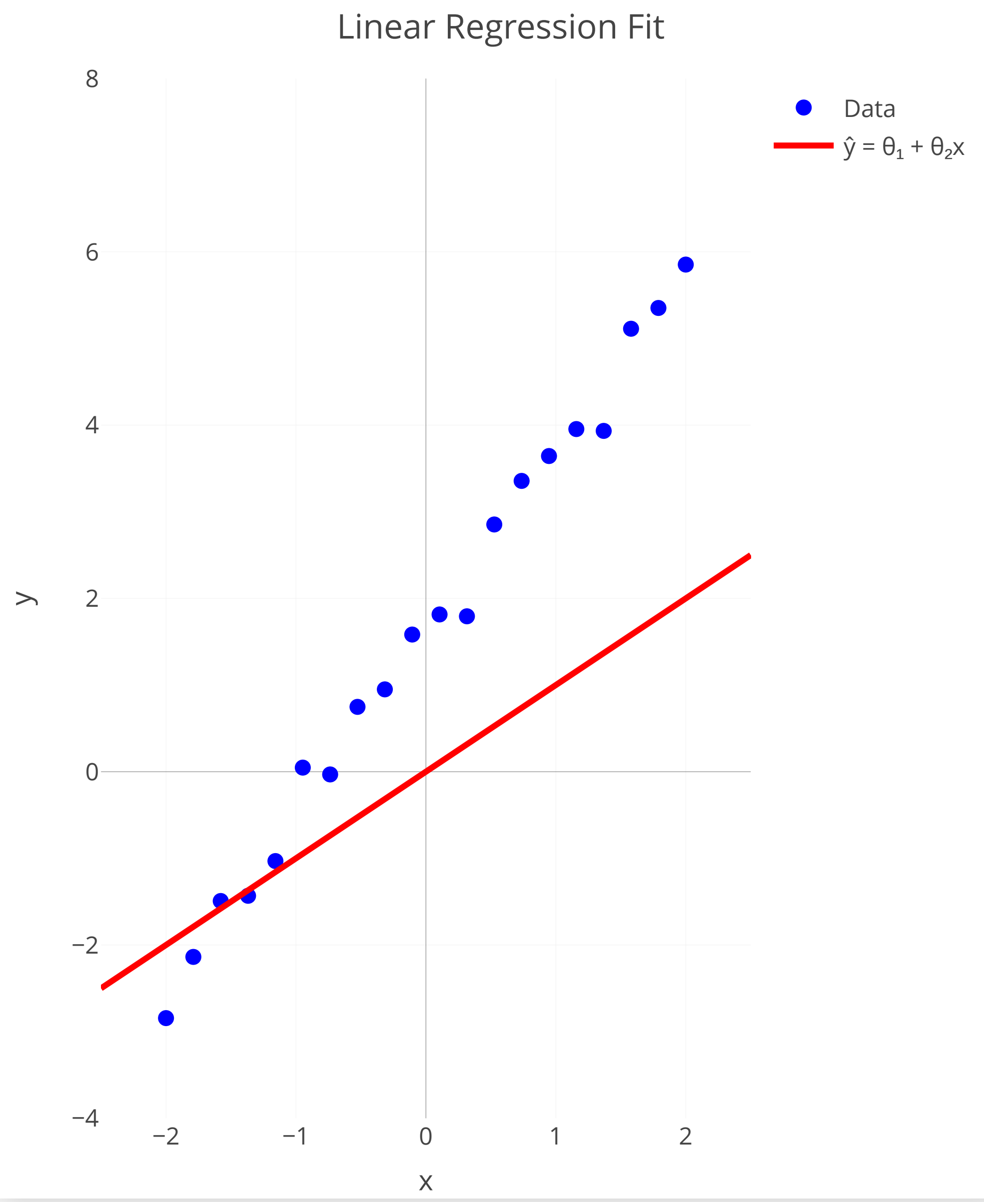
**$\theta_2$ (Slope):**

1.0

**Model:**
`f(x) = 0.0 + 1.0x`

**Current Loss:**
**4.375**



### Empirical Risk Minimization: Summary

- (*modelization*) Choose a (parametrized) model class $f_{\theta} \in \mathcal{M}$
- Choose a loss function $L(\cdot)$ (L1-loss, L2-loss, $\ldots$)
- For each given $\boldsymbol{\theta}$, can compute the empirical risk $\mathcal{J}(\boldsymbol{\theta})$
- (*model fitting*) Pick the model $f_{\boldsymbol{\theta}}^*$ that has the minimal risk
- (*prediction*) Use the fitted model $f_{\boldsymbol{\theta}}^*$ to make predictions at unseen points $\boldsymbol{x}$

## Question: How can we find the optimal parameter $\theta^*$ that minimizes the risk?

### Empirical Risk Minimization

- for each possible value of the parameter $\theta$ define the **empirical risk**:

$$\mathcal{J}(\theta) = \sum_{i=1}^{n} L(y^{(i)}, f_{\theta}(\boldsymbol{x})^{(i)})$$

- best model is the one with the smallest loss (smallest risk)

$$\theta^* = \arg\min_{\theta \in \Theta} J(\theta)$$

## How can we find the optimal parameter $\theta^*$?

- Some models have an **analytical solution** (e.g. linear regression)

````
```{python}
from sklearn.linear_model import LinearRegression
reg = LinearRegression().fit(X, y)
y_new = reg.predict(np.array(X_new)
```
````

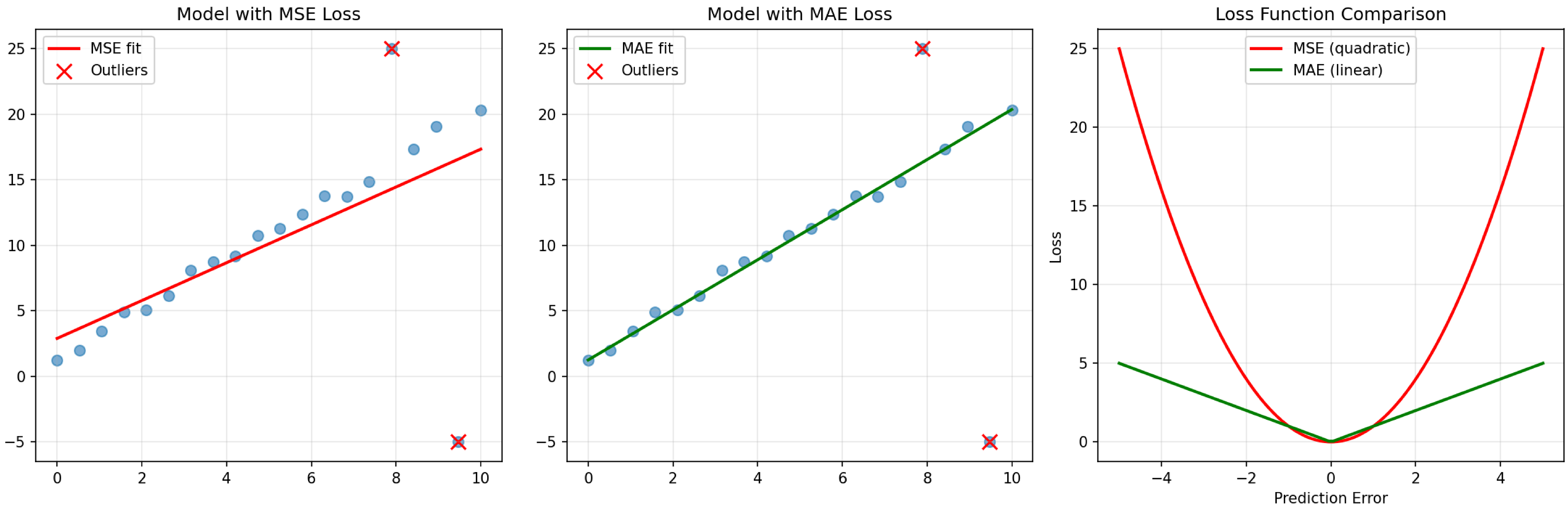
- In general, perform **iterative minimization** (next chapter). `scipy` provides iterative minimization routine:

````
```{python}
from scipy.optimize import minimize
res = minimize(f, x0,
               method='nelder-mead',
               options={'xatol': 1e-8, 'disp': True})
```
````

# How can we choose the Loss function?

# L2 vs L1 Loss Behavior with Outliers

```
MSE fit: y = 1.44x + 2.89
MAE fit: y = 1.91x + 1.25
```




Notice how MSE fit is more influenced by outliers

This is an instance of **overfitting** (more on this in the next chapter).