Introduction to Machine Learning

Chapter 5: Linear regression models

Bernd Bischl, Christoph Molnar, Daniel Schalk, Fabian Scheipl

Department of Statistics – LMU Munich

LINEAR REGRESSION: REPRESENTATION

We want to predict a numerical target variable by a *linear transformation* of the features $x \in \mathbb{R}^p$.

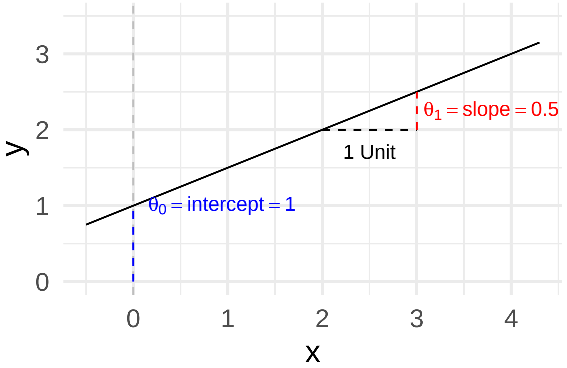
So with $\theta \in \mathbb{R}^p$ this mapping can be written as:

$$\begin{aligned} y = f(x) &= \theta_0 + \theta^T x \\ &= \theta_0 + \theta_1 x_1 + \cdots + \theta_p x_p \end{aligned}$$

This restricts the hypothesis space H to all linear functions in θ:

$$H = \{\theta_0 + \theta^T x \mid (\theta_0, \theta) \in \mathbb{R}^{p+1}\}$$

LINEAR REGRESSION: REPRESENTATION



$$y = \theta_0 + \theta \cdot x$$

LINEAR REGRESSION: REPRESENTATION

Given observed labeled data $\mathcal{D}$, how to find (θ, θ_0)?

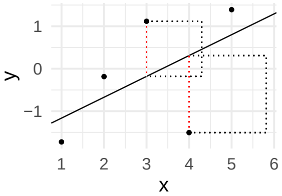
This is **learning** or parameter estimation, the learner does exactly this by **empirical risk minimization**.

NB: We assume from now on that θ_0 is included in θ.

LINEAR REGRESSION: EVALUATION

We could measure training error as the sum of squared prediction errors (SSE). This is also called the **L2 loss**, or L2 risk:

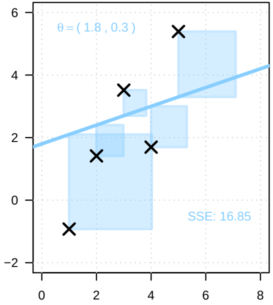
$$\mathcal{R}_{\text{emp}}(\theta) = \text{SSE}(\theta) = \sum_{i=1}^{n} L\left(y^{(i)}, f\left(x^{(i)} | \theta\right)\right) = \sum_{i=1}^{n} \left(y^{(i)} - \theta^T x^{(i)}\right)^2$$



Minimizing the squared error is computationally much simpler than minimizing the absolute differences (this would be called **L1 loss**).

LINEAR MODEL: OPTIMIZATION

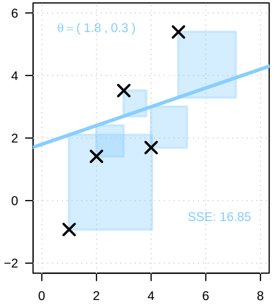
We want to find the parameters θ of the linear model, i.e., an element of the hypothesis space H that fits the data optimally.
So we evaluate different candidates for θ.
A first (random) try yields a rather large SSE: (**Evaluation**).

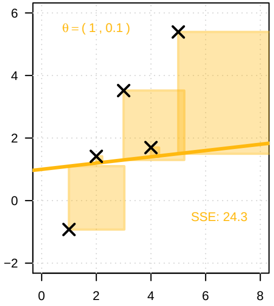


LINEAR MODEL: OPTIMIZATION

We want to find the parameters of the LM / an element of the hypothesis space H that best suits the data. So we evaluate different candidates for θ.

Another line yields an even bigger SSE (**Evaluation**). Therefore, this one is even worse in terms of empirical risk.

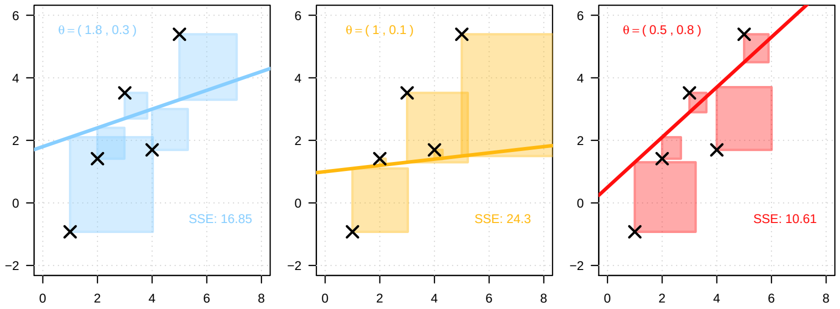




LINEAR MODEL: OPTIMIZATION

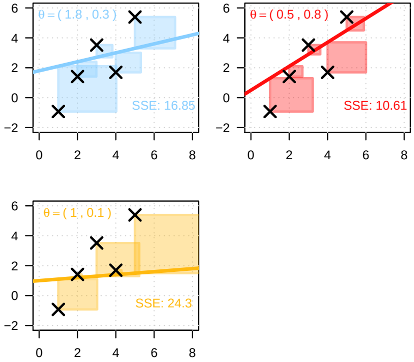
We want to find the parameters of the LM / an element of the hypothesis space *H* that best suits the data. So we evaluate different candidates for θ.

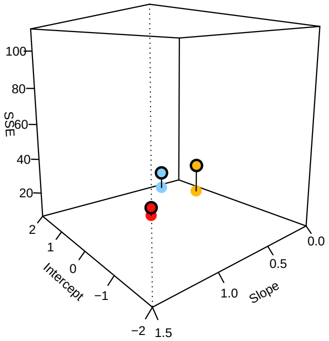
Another line yields an even bigger SSE (**Evaluation**). Therefore, this one is even worse in terms of empirical risk. Let's try again:



LINEAR MODEL: OPTIMIZATION

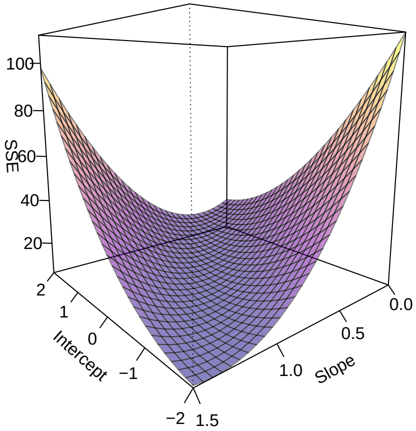
Since every θ results in a specific value of $\mathcal{R}_{\text{emp}}(\theta)$, and we try to find $\arg\min_\theta \mathcal{R}_{\text{emp}}(\theta)$, let's look at what we have so far:





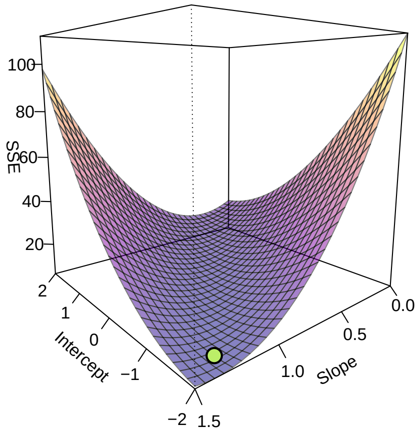
LINEAR MODEL: OPTIMIZATION

Instead of guessing, we use **optimization** to find the best θ:



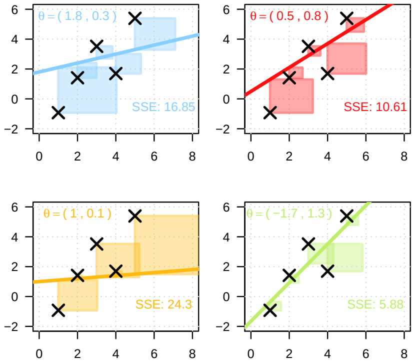
LINEAR MODEL: OPTIMIZATION

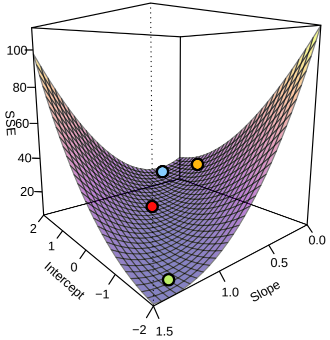
Instead of guessing, we use **optimization** to find the best θ:



LINEAR MODEL: OPTIMIZATION

Instead of guessing, we use **optimization** to find the best θ:





LINEAR MODEL: OPTIMIZATION

For L2 regression, we can find this optimal value analytically:

$$\begin{aligned}\hat{\theta} &= \arg\min_{\theta} \mathcal{R}_{\text{emp}}(\theta) = \sum_{i=1}^{n} \left(y^{(i)} - \theta^T x^{(i)}\right)^2 \\ &= \arg\min_{\theta} \|y - X\theta\|_2^2\end{aligned}$$

where $X = \begin{pmatrix} 1 & x_1^{(1)} & \dots & x_p^{(1)} \\ 1 & x_1^{(2)} & \dots & x_p^{(2)} \\ \vdots & \vdots & & \vdots \\ 1 & x_1^{(n)} & \dots & x_p^{(n)} \end{pmatrix}$ is the $n \times (p+1)$-**design matrix.**

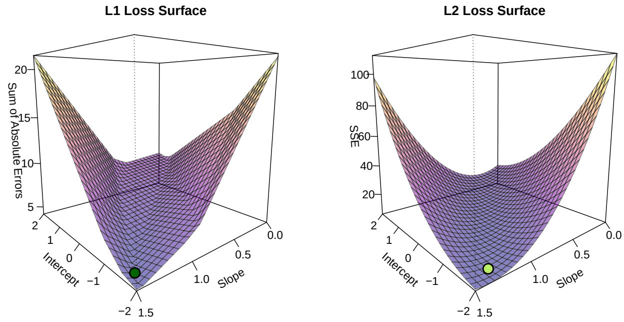
This yields the so called normal equations for the LM:

$$\frac{\delta}{\delta\theta}\mathcal{R}_{\text{emp}}(\theta) = 0 \qquad \Rightarrow \qquad \hat{\theta} = \left(X^T X\right)^{-1} X^T y$$

EXAMPLE: REGRESSION WITH L1 VS L2 LOSS

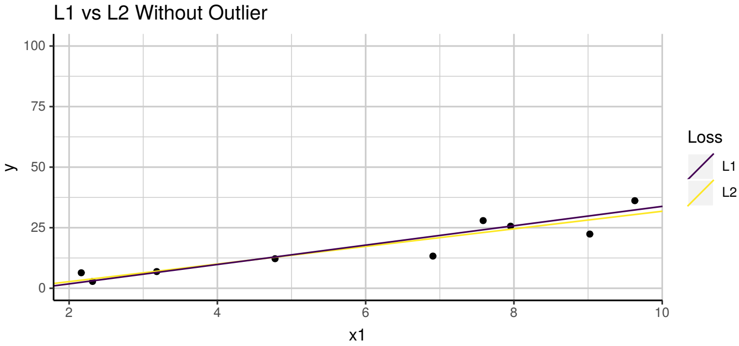
We could also minimize the L1 loss. This changes the evaluation and optimization step:

$$\mathcal{R}_{\text{emp}}(\theta) = \sum_{i=1}^{n} L\left(y^{(i)}, f\left(x^{(i)}|\theta\right)\right) = \sum_{i=1}^{n} \left|y^{(i)} - \theta^T x^{(i)}\right| \qquad \text{(Evaluation)}$$



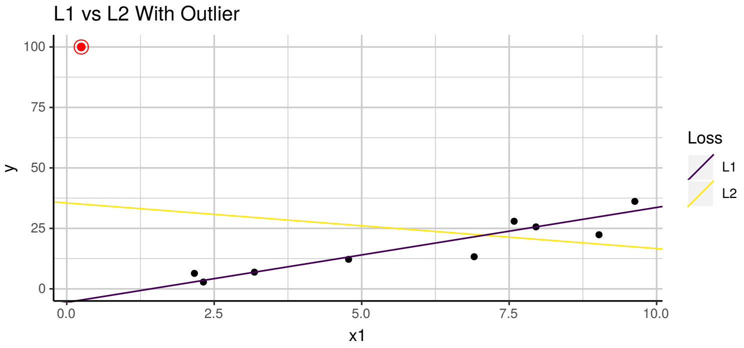
L1 loss is harder to optimize, but the model is less sensitive to outliers.

EXAMPLE: REGRESSION WITH L1 VS L2 LOSS



EXAMPLE: REGRESSION WITH L1 VS L2 LOSS

Adding an outlier (highlighted red) pulls the line fitted with L2 into the direction of the outlier:



LINEAR REGRESSION

Representation: Design matrix X, coefficients θ.

Evaluation: Any regression loss function.

Optimization: Direct analytic solution for L2-loss, numerical optimization for L1 and others.

REGRESSION: POLYNOMIALS

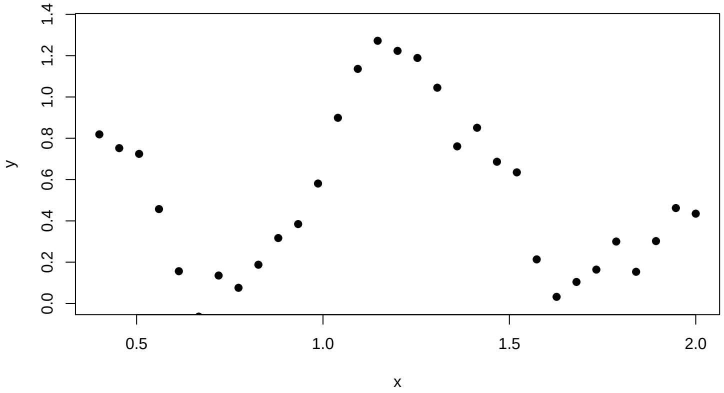
We can make regression models much more flexible by using *polynomials* x_j^d – or any other *derived features* like $\sin(x_j)$ or $(x_j \cdot x_k)$ – as additional features.

The optimization and evaluation for the learner remains the same.

Only the representation of the learner changes: it now includes the derived features as additional columns in the design matrix as well as the coefficients associated with them.

REGRESSION: POLYNOMIALS

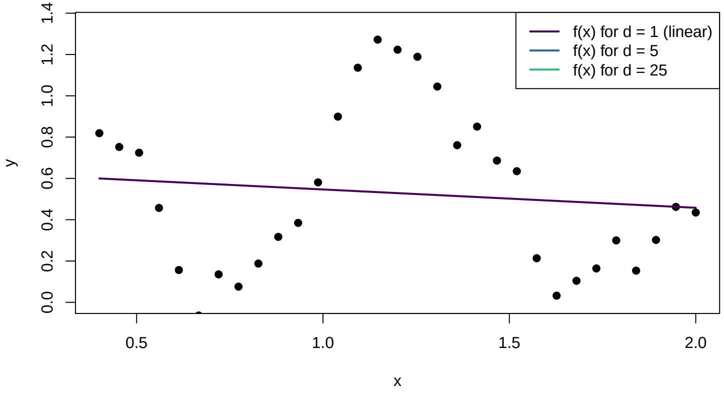
Polynomial regression example



REGRESSION: POLYNOMIALS

Polynomial regression example

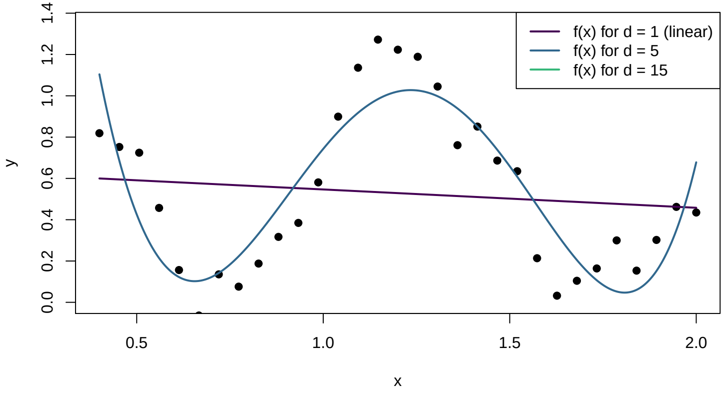
Models of different *complexity*, i.e. of different polynomial order *d*, are fitted to the data:



REGRESSION: POLYNOMIALS

Polynomial regression example

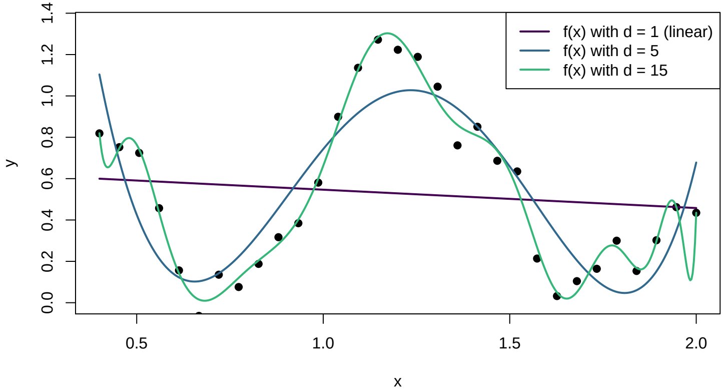
Models of different *complexity*, i.e. of different polynomial order *d*, are fitted to the data:



REGRESSION: POLYNOMIALS

Polynomial regression example

Models of different *complexity*, i.e. of different polynomial order d, are fitted to the data:



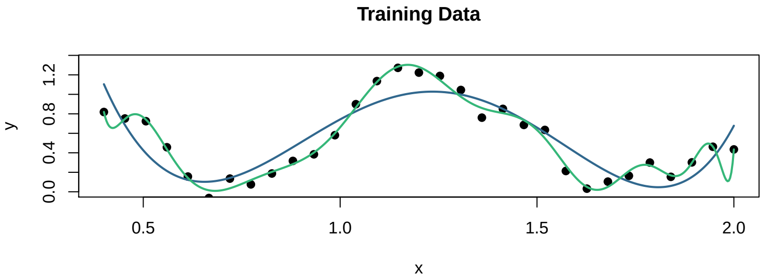
REGRESSION: POLYNOMIALS

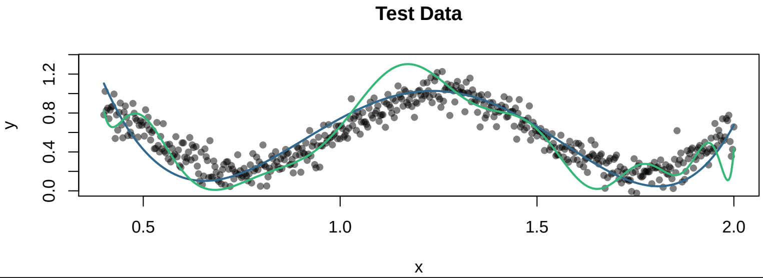
The higher d is, the more **capacity** the learner has to learn complicated functions of x, but this also increases the danger of **overfitting**:

The model space H contains so many complex functions that we are able to find one that approximates the training data arbitrarily well.

However, predictions on new data are not as successful because our model includes too much of the "noise" of the training data (much, much more on this later).

REGRESSION: POLYNOMIALS


Training Data
y
x


Test Data
y
x