Bauhaus-Universität Weimar

2 Fundamental Concepts of Machine Learning: Learning

Leon Herrmann

Stefan Kollmannsberger

Chair of Data Engineering in Construction

Bauhaus-Universität Weimar

Deep Learning in Computational Mechanics – an introductory course, Herrmann et al. 2025

website

book

Contents

2.5 Linear Regression – Optimization

$$\min_{\boldsymbol{w},b} C(\boldsymbol{w}, b) = \min_{\boldsymbol{w},b} \frac{1}{m} \sum_{i=1}^{m} \left(y_i - (\boldsymbol{w} \cdot \boldsymbol{x}_i + b)\right)^2$$

Note that $\boldsymbol{X}$ requires a column of ones for the bias b

For a more concise notation let us denote all learnable parameters in a vector $\boldsymbol{\Theta} = (\boldsymbol{w}, b)^T$

This allows to write the model function $\hat{y}_i = \boldsymbol{w}^T \boldsymbol{x}_i + b$ as $\hat{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\Theta}$ yielding the minimization

All predictions $\hat{y}_i$ are collected in the vector $\hat{\boldsymbol{y}}$.

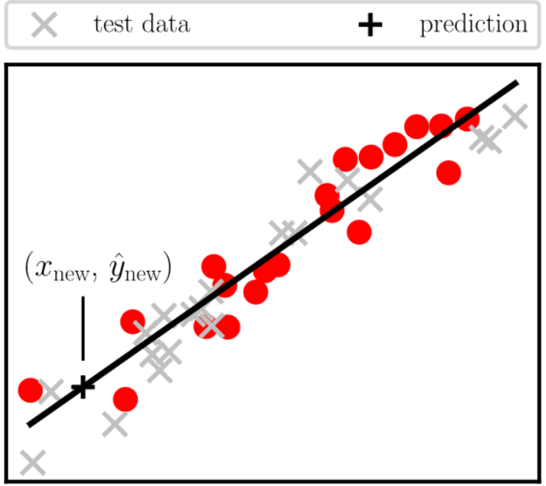
$$\min_{\boldsymbol{\Theta}} C(\boldsymbol{\Theta}) = \min_{\boldsymbol{\Theta}} (\tilde{\boldsymbol{y}} - \boldsymbol{X}\boldsymbol{\Theta})(\tilde{\boldsymbol{y}} - \boldsymbol{X}\boldsymbol{\Theta}) = \min_{\boldsymbol{\Theta}} (\tilde{\boldsymbol{y}}^T \tilde{\boldsymbol{y}} - 2\tilde{\boldsymbol{y}}^T \boldsymbol{X}\boldsymbol{\Theta} + (\boldsymbol{X}\boldsymbol{\Theta})^T \boldsymbol{X}\boldsymbol{\Theta})$$

The minimization is solved by setting the first derivative of C with respect to $\boldsymbol{\Theta}$ to zero (using $\boldsymbol{r} = \tilde{\boldsymbol{y}} - \boldsymbol{X}\boldsymbol{\Theta}$)

$$\frac{1}{2} \frac{\partial \boldsymbol{r}(\boldsymbol{\Theta})^2}{\partial \boldsymbol{\Theta}} = \frac{1}{2}(-2\boldsymbol{X}^T \tilde{\boldsymbol{y}} + 2\boldsymbol{X}^T \boldsymbol{X}\boldsymbol{\Theta}) = -\boldsymbol{X}^T \tilde{\boldsymbol{y}} + \boldsymbol{X}^T \boldsymbol{X}\boldsymbol{\Theta} = 0$$

$$\boldsymbol{X}^T \boldsymbol{X}\boldsymbol{\Theta} = \boldsymbol{X}^T \tilde{\boldsymbol{y}}$$

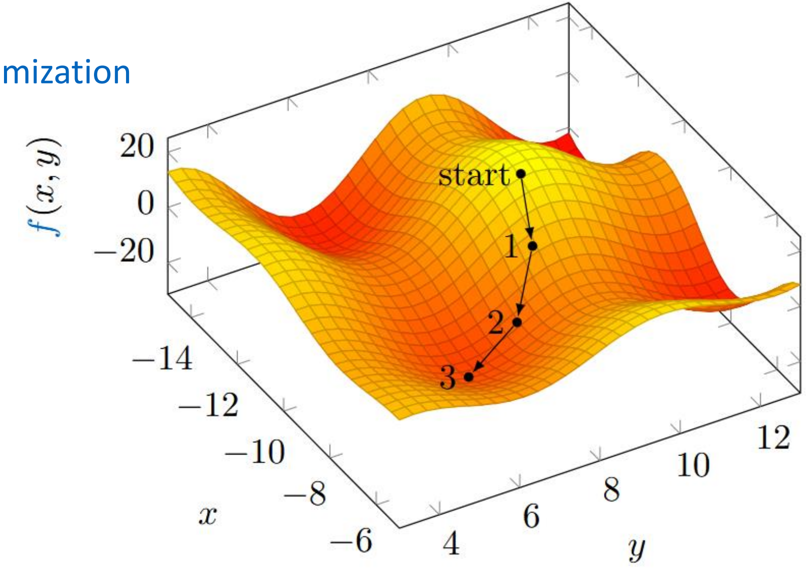
$$\boldsymbol{\Theta} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \tilde{\boldsymbol{y}}$$



Such a closed form solution is only possible if $\hat{\boldsymbol{y}}$ (or rather $\partial C / \partial \boldsymbol{\Theta}$) is **linear** with respect to $\boldsymbol{\Theta}$

2.8.1 Gradient Descent

Improve prediction $\hat{y}_i = \boldsymbol{w} \cdot \boldsymbol{x}_i + b$ via (iterative) cost function minimization



$$\min_{\boldsymbol{w},b} C(\boldsymbol{w}, b) = \min_{\boldsymbol{w},b} \frac{1}{m} \sum_{i=1}^{m} \left(\tilde{\boldsymbol{y}}_i - (\boldsymbol{w} \cdot \boldsymbol{x}_i + b)\right)^2$$

Partial derivatives of cost function with respect to each parameter

$$\frac{\partial C}{\partial \boldsymbol{w}} = \frac{1}{m} \sum_{i=1}^{m} -2x_i(\tilde{\boldsymbol{y}}_i - (\boldsymbol{w} \cdot \boldsymbol{x}_i + b))$$

$$\frac{\partial C}{\partial b} = \frac{1}{m} \sum_{i=1}^{m} -2(\tilde{\boldsymbol{y}}_i - (\boldsymbol{w} \cdot \boldsymbol{x}_i + b))$$

Each gradient descent iteration updates the parameters, such that the cost function decreases

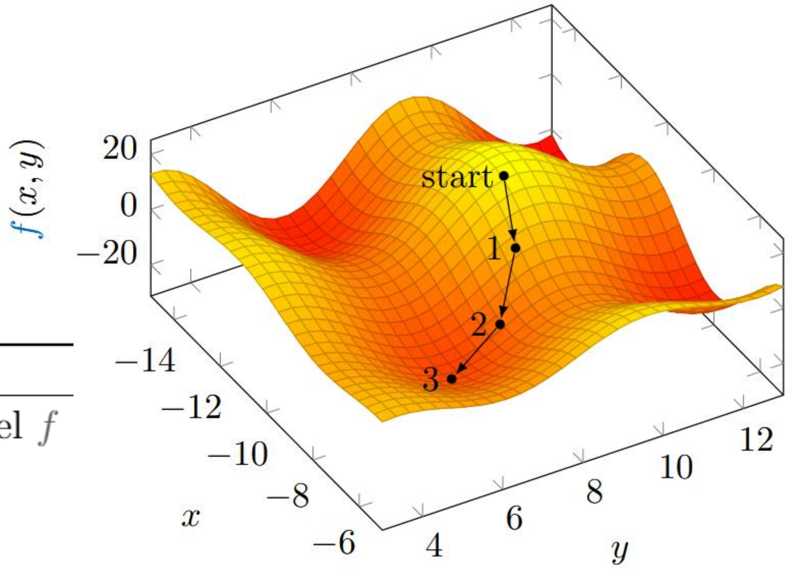
$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \frac{\partial C}{\partial \boldsymbol{w}}$$

$$b \leftarrow b - \alpha \frac{\partial C}{\partial b}$$

α (the **learning rate**) controls the step size

2.8.1 Gradient Descent

- Generalized gradient descent algorithm
- In machine learning:
  - Number of iterations is called **number of epochs**
  - Step size is called **learning rate**



Algorithm 1 Gradient descent

Require: dataset $\tilde{x}, \tilde{y}$, number of epochs n, step size α, model f
initialize the model $f(x; \Theta)$
for all n **do**
Compute the cost function $C(f(\tilde{x}; \Theta), \tilde{y})$
Compute the gradient $\nabla_{\Theta} C$
Update the model parameters $\Theta \leftarrow \Theta - \alpha \nabla_{\Theta} C$
end for

Exercises

- E.4 Linear Regression (P & C)
  - Perform a linear regression once by computing the weights directly and once using gradient descent. Do this by hand calculation and with a Python implementation.

2.8.1 Gradient Descent – Stochastic Gradient Descent

Gradient Descent = **Full-Batch Gradient Descent**

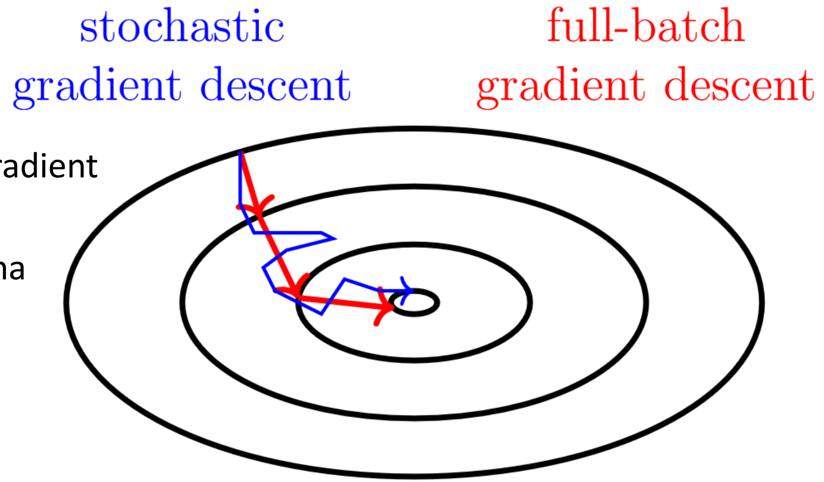
- All samples are considered during the gradient computation
- Accurate but expensive

Stochastic Gradient Descent (SGD)

- Only one (randomly selected) sample is used to compute the gradient
- Cheap but inaccurate gradients
- Inaccuracy induces stochasticity, enabling escape of local minima

Mini-Batch Stochastic Gradient Descent

- Samples are grouped in small batches of size k to approximate gradients more accurately



$$\frac{\partial C}{\partial \boldsymbol{w}} = \frac{1}{k}\sum_{i=1}^{k} -2\boldsymbol{x}_i(\tilde{y}_i - (\boldsymbol{w}\cdot\boldsymbol{x}_i + b))$$

$$\frac{\partial C}{\partial b} = \frac{1}{k}\sum_{i=1}^{k} -2(\tilde{y}_i - (\boldsymbol{w}\cdot\boldsymbol{x}_i + b))$$

Is it fair to compare the quality of a model after 100 iterations of Full-Batch, Mini-Batch, and Stochastic Gradient Descent?

- Batch size k is a hyperparameter, typically chosen as large as the GPU memory allows

2.8 Optimization Techniques

- 2.8.1 Gradient Descent
  - Stochastic Gradient Descent
  - Mini-Batch Stochastic Gradient Descent
- 2.8.2.1 **Gradient Descent with Momentum**
  - Uses a moving average of the gradient to improve the gradient estimation and avoid local minima
- 2.8.2.2 **AdaGrad**
  - Uses an accumulation of the squared gradients to normalize the updates and improve convergence
- 2.8.2.3 **RMSprop**
  - Extension of AdaGrad to avoid premature convergence by considering a moving average of the squared gradients
- 2.8.2.4 **Adam**
  - Combination of Gradient Descent with Momentum and RMSprop
- 2.8.3 **L-BFGS**
  - Leverages second order derivatives (Hessian) to improve convergence

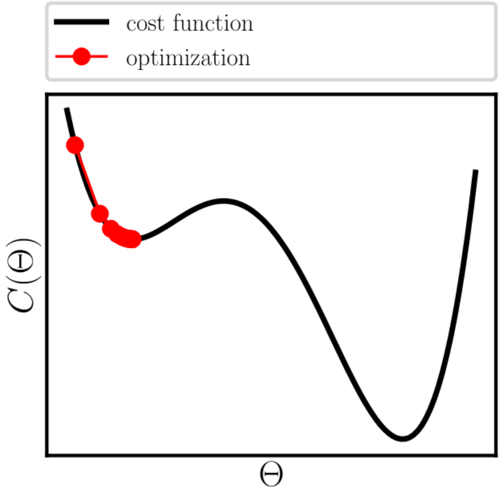
2.8 Optimization Techniques – GD with Momentum

Extension of update rule with **momentum** term $\boldsymbol{v}_t$

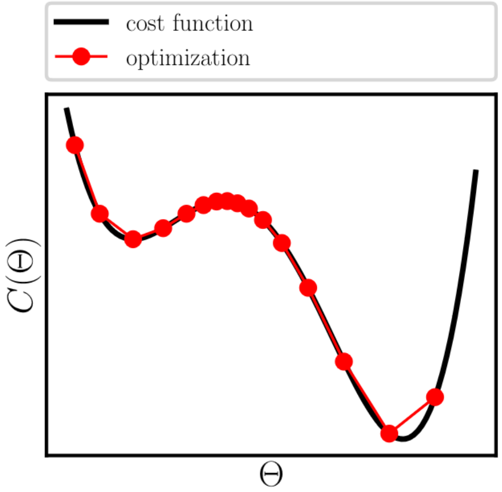
$$\boldsymbol{v}_{t+1} = \eta \boldsymbol{v}_t + \nabla_{\boldsymbol{\Theta}} C(\boldsymbol{\Theta})$$
$$\boldsymbol{\Theta}_{t+1} = \boldsymbol{\Theta}_t - \alpha \boldsymbol{v}_{t+1}$$

η is a hyperparameter, controlling the influence of previous gradients

$\boldsymbol{v}_t$ is analogous to the velocity towards the solution, and η is analogous to friction slowing that motion



Gradient descent



Gradient descent with momentum

Optimization Techniques - AdaGrad

Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, Duchi et al. 2011

AdaGrad introduces an adaptive learning rate to better reach optima

- Achieved by tracking the accumulated squared gradients

$$\bar{\boldsymbol{g}}_t = \sum_{\tau=1}^{t} \left[\nabla_{\boldsymbol{\Theta}_\tau} C(\boldsymbol{\Theta}_\tau)\right]^2$$

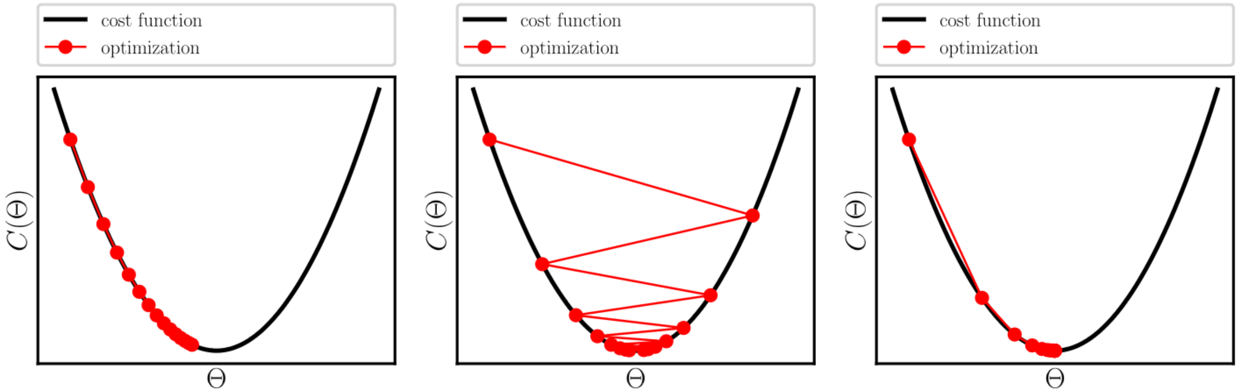
$$\boldsymbol{\Theta}_{t+1} = \boldsymbol{\Theta}_t - \frac{\alpha}{\sqrt{\bar{\boldsymbol{g}}_t^2 + \varepsilon}} \nabla_{\boldsymbol{\Theta}_t} C(\boldsymbol{\Theta}_t)$$

Each parameter is scaled individually

ε is small and prevents division by zero

- (Consistently) small gradients are amplified (indicates closeness to optimum)
- (Consistently) large gradients are supressed (indicates distance to optimum, instability, overshooting)

Too small α
Too large α
AdaGrad



2.8 Optimization Techniques – RMSprop

- AdaGrad can lead to a fast reduction in the learning rate → can prevent convergence
- **RMSprop** relies on a moving average of the squared gradients (via exponentially decaying average)

$$\tilde{\boldsymbol{g}}_t^2 = \rho \tilde{\boldsymbol{g}}_{t-1}^2 + (1-\rho)\left[\nabla_{\boldsymbol{\Theta}_t} C(\boldsymbol{\Theta}_t)\right]^2$$

$$\boldsymbol{\Theta}_{t+1} = \boldsymbol{\Theta}_t - \frac{\alpha}{\sqrt{\tilde{\boldsymbol{g}}_t^2 + \varepsilon}} \nabla_{\boldsymbol{\Theta}_t} C(\boldsymbol{\Theta}_t)$$

ε is small and prevents division by zero

- ρ is a hyperparameter controlling the decay rate
- Again each parameter is scaled individually, effectively yielding an individual learning rate for each parameter

2.8 Optimization Techniques – Adam

Adam: A Method for Stochastic Optimization, Kingma et al. 2014

Adam combines the strengths of **gradient descent with momentum** and **RMSprop**

- Momentum via first statistical moment

$$\boldsymbol{m}_t = \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\nabla_{\boldsymbol{\Theta}_t} C(\boldsymbol{\Theta}_t)$$

- Moving average of squared gradients via second statistical moment

$$\boldsymbol{n}_t = \beta_2 \boldsymbol{n}_{t-1} + (1-\beta_2)\left[\nabla_{\boldsymbol{\Theta}_t} C(\boldsymbol{\Theta}_t)\right]^2$$

- Bias correction due to initialization via $\boldsymbol{m}_0 = \boldsymbol{n}_0 = 0$

$$\widetilde{\boldsymbol{m}}_t = \frac{\boldsymbol{m}_t}{1-\beta_1^t}$$

$$\widetilde{\boldsymbol{n}}_t = \frac{\boldsymbol{n}_t}{1-\beta_2^t}$$

- Gradient update via corrected statistical moments

$$\boldsymbol{\Theta}_{t+1} = \boldsymbol{\Theta}_t - \frac{\alpha \widetilde{\boldsymbol{m}}_t}{\sqrt{\widetilde{\boldsymbol{n}}_t} + \varepsilon}$$

- β_1, β_2 are hyperparameters, typically chosen as $\beta_1 = 0.9, \beta_2 = 0.999$

Exercises

- E.6 Adam Optimizer (C)
  - Implement the Adam optimizer and find the optimum of the Rosenbrock function.

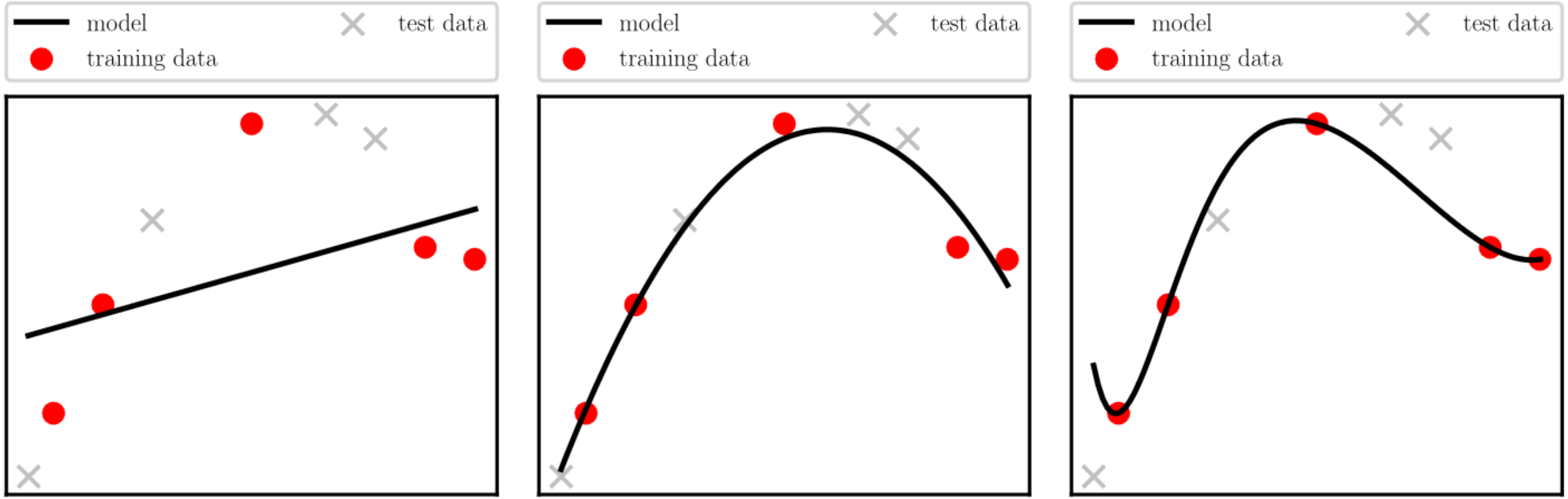
2.6 Overfitting Versus Underfitting

Underfitting: model capacity is too low

- Unable to fit the data

Overfitting: model capacity is too high

- Unable to generalize
- Is monitored with the test data, that is not used during training (and tuning)



Underfitting | Ideal fitting | Overfitting

2.6 Overfitting Versus Underfitting

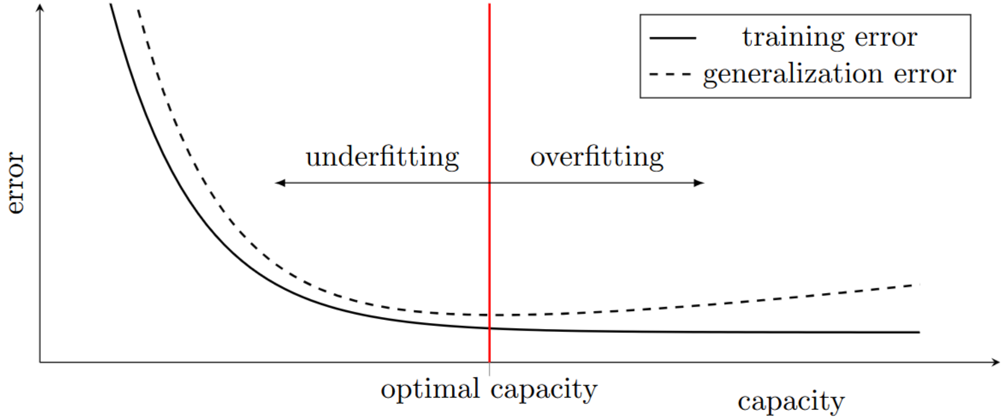
- **Underfitting**: model capacity is too low
- **Overfitting**: model capacity is too great

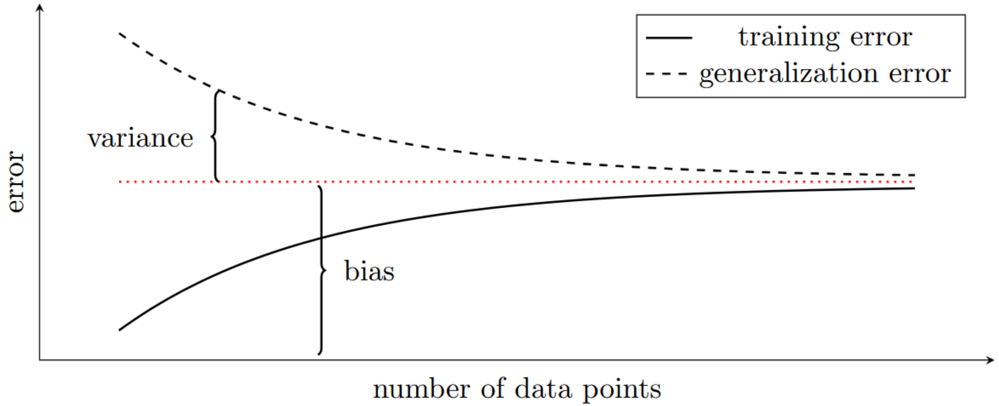
- Remedies (see Chapter 3 for more)
  - Cross-validation
  - More data
  - Data augmentation
  - Regularization
  - Early stopping

- **Variance** is related to the generalization error
- **Bias** is related to the training error

What is preferable:

- A high bias and low variance?
- Or a high variance and a low bias?



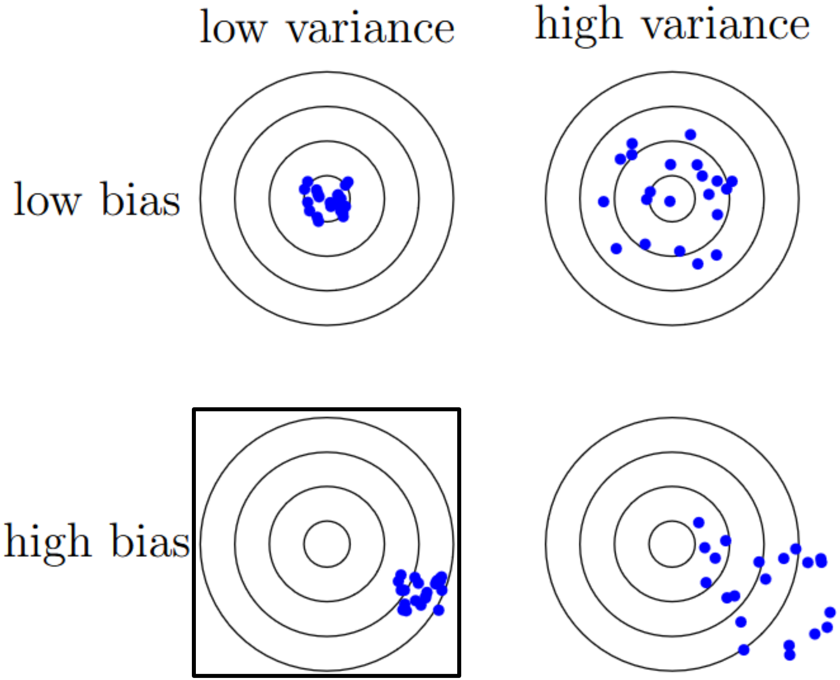


2.6 Overfitting Versus Underfitting

Low variance and high bias is preferable

Noise: a Flaw in Human Judgement, Kahneman et al. 2021

- A model/human should rather be consistently (but predictably) wrong, than inconsistent (and unpredictable).
- This is even worse in a machine learning model, where the best predictions are on datapoints close to the training data.



2.7 Regularization

Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error (low variance) but ideally not its training error (low bias)

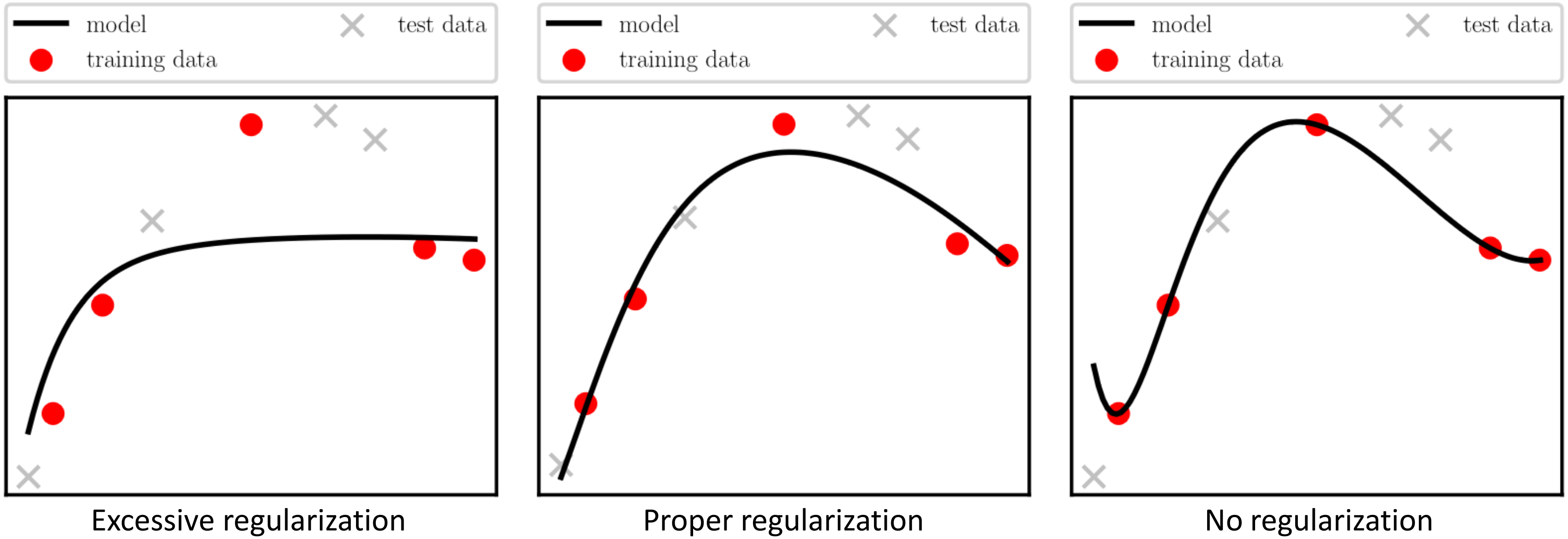
- Regularization is always a trade-off between **bias** and **variance**

- For linear regression $\hat{y} = \boldsymbol{w} \cdot \boldsymbol{x} + b$
  - L^1-regularization
$$\tilde{C}(\boldsymbol{w}, b) = C(\boldsymbol{w}, b) + \lambda||\boldsymbol{w}||_1$$
  - L^2-regularization
$$\tilde{C}(\boldsymbol{w}, b) = C(\boldsymbol{w}, b) + \lambda\boldsymbol{w}^T\boldsymbol{w}$$
  - Penalty factor λ is a hyperparameter that controls the penalty term
    - Punishes large coefficients, as seen in oscillations, where large slope coefficients occur
    - A small λ converges towards the initial regression
    - A large λ returns a simple or sparse model

In L^1: the derivative of $||\boldsymbol{w}||_1$ is constant pushing the unimportant weights to zero.
In L^2: the derivative of $\boldsymbol{w}^T\boldsymbol{w}$ is proportional to $\boldsymbol{w}$ resulting in small but non-zero unimportant weights.

2.7 Regularization

L^2-regularization



Exercises

- E.5 Higher-Order Regression (C)
  - Extend the linear regression Python implementation to higher-order regression. Experiment with underfitting, overfitting, and regularization.

Contents

2 Fundamental Concepts of Machine Learning: Learning

Leon Herrmann

Stefan Kollmannsberger

Chair of Data Engineering in Construction

Bauhaus-Universität Weimar

Deep Learning in Computational Mechanics – an introductory course, Herrmann et al. 2025

website

book