Machine Learning in Physics

Selected chapters on astrophysics

Pavel Baláž & Martin Žonda

Department of Condensed Matter Physics, Charles University, Prague

MACHINE LEARNING IN PHYSICS

What is Machine Learning?

Machine Learning (ML)

- Arthur Samuel (1959)

What is Machine Learning?

Machine Learning (ML)

Arthur Samuel (1959)

computer algorithms that improve automatically through experience

Tom Mitchell: Machine Learning (1997)

What is Machine Learning?

Machine Learning (ML)

- Arthur Samuel (1959)

- computer algorithms that improve automatically through experience
  - *Tom Mitchell: Machine Learning (1997)*

- part of artificial intelligence (AI)

What is Machine Learning?

Machine Learning (ML)

Arthur Samuel (1959)

- computer algorithms that improve automatically through experience

  Tom Mitchell: Machine Learning (1997)

- part of artificial intelligence (AI)

- ML builds a model based on sample data in order to make predictions or decisions without being *explicitly programmed* to do so

General AI

- hypothetical ability
  to understand or learn
  any intellectual task
  that a human being can

- Tests for confirming AI
  - The Turing test
  - The Coffee test
  - The Robot College Student Test
  - The Employment Test

Types of Artificial Intelligence

General AI

- hypothetical ability
  to understand or learn
  any intellectual task
  that a human being can

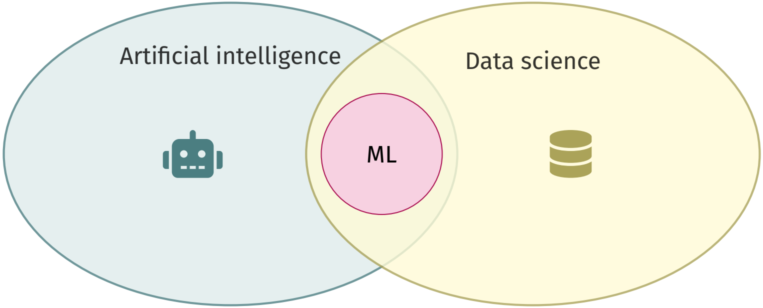
- Tests for confirming AI
  - The Turing test
  - The Coffee test
  - The Robot College Student Test
  - The Employment Test

Narrow AI

- weak AI / extended AI
- specializes in one area
- solves one particular problem

- Examples
  - spam email filtering
  - music/movies recommendations
  - autonomous vehicles

Machine learning: Part of Artificial Intelligence



How a machine can learn?

Supervised learning

- we have a dataset of examples with related targets (desired results)
- learning from examples

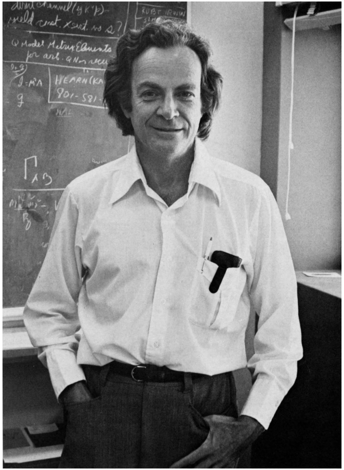
- classification
- regression

Unsupervised learning

- we have examples without any associated targets
- the model has to determine the data patterns

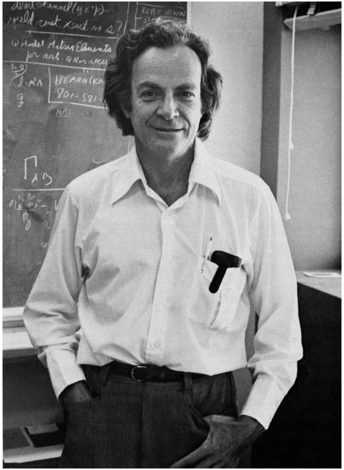
- clustering
- principal components analysis

What is Understanding? *Scientific process is analogous to discovering chess* ♝♜



Richard Feynman (1918 – 1988)

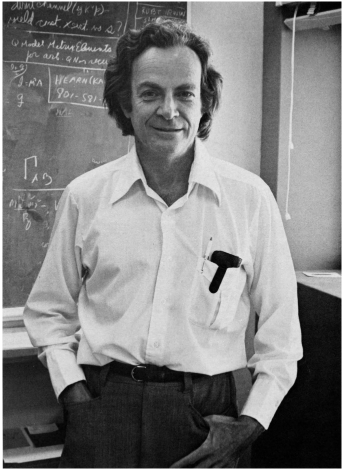
What is Understanding? *Scientific process is analogous to discovering chess ♗♜*



Richard Feynman (1918 – 1988)

- you don’t know the rules of the game, but you’re allowed to look at the board from time to time

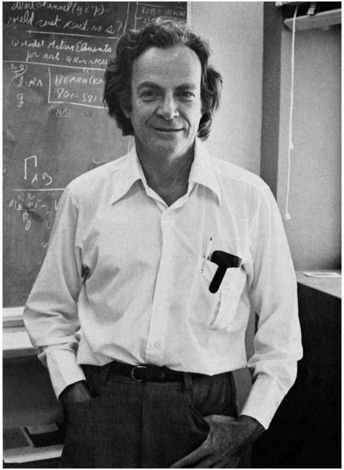
What is Understanding? *Scientific process is analogous to discovering chess*



Richard Feynman (1918 – 1988)

- you don’t know the rules of the game, but you’re allowed to look at the board from time to time
- from these observations, you try to figure out what the rules are

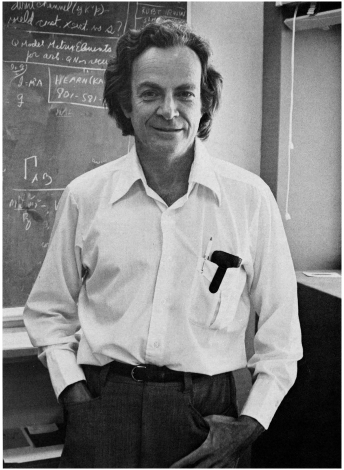
What is Understanding? *Scientific process is analogous to discovering chess*



Richard Feynman (1918 – 1988)

- you don't know the rules of the game, but you're allowed to look at the board from time to time
- from these observations, you try to figure out what the rules are
- You might discover that when there's only one bishop around on the board, that the bishop maintains its color or that it moves on a diagonal

What is Understanding? *Scientific process is analogous to discovering chess* ♗♖



Richard Feynman (1918 – 1988)

- ♞ you don't know the rules of the game, but you're allowed to look at the board from time to time
- ♝ from these observations, you try to figure out what the rules are
- ♚ You might discover that when there's only one bishop around on the board, that the bishop maintains its color or that it moves on a diagonal

Data-driven approach

We can learn physical laws from observations $\Longleftrightarrow$ data

AlphaGo

- Go is an abstract strategy board game for two players

- Go is an abstract strategy board game for two players
- in 2015 the original AlphaGo became the first computer program to beat a human professional Go player

AlphaGo

AlphaGo

- Go is an abstract strategy board game for two players
- in 2015 the original AlphaGo became the first computer program to beat a human professional Go player
- in 2017 AlphaGo beat the number one ranked player in the world in a three-game match

AlphaGo

- Go is an abstract strategy board game for two players
- in 2015 the original AlphaGo became the first computer program to beat a human professional Go player
- in 2017 AlphaGo beat the number one ranked player in the world in a three-game match

! AlphaGo algorithm finds its moves based on knowledge previously acquired by ML, specifically by an artificial neural network both from human and computer play

AlphaGo

- Go is an abstract strategy board game for two players
- in 2015 the original AlphaGo became the first computer program to beat a human professional Go player
- in 2017 AlphaGo beat the number one ranked player in the world in a three-game match

! AlphaGo algorithm finds its moves based on knowledge previously acquired by ML, specifically by an artificial neural network both from human and computer play

- a neural network is trained to identify the best moves and the winning percentages of these moves

AlphaGo

Machine Learning Lecture

Lecture plan

- Simple models:
  - linear regression
  - classification algorithms
- Unsupervised data processing:
  - principal components analysis (PCA)
  - clustering
- neural networks and deep learning:
  - feed forward neural network
  - convolutional neural network
  - autoencoder
- Machine Learning for time series
  - forecasting
  - classification
  - anomaly detection

Literatura

- Ian Goodfellow *et al.*: Deep Learning, MIT Press
  https://www.deeplearningbook.org/
- F. Chollet: Deep learning v jazyku Python. Knihovny Keras, Tensorflow, Grada (2019)
- A. Geron: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, O'Reilly Media (2019)
- T. M. Mitchell: Machine learning, McGraw-Hill Science/Engineering/Math (1997)

Literatura

- P. Mehta *et al*: A High-Bias, Low-Variance Introduction to Machine Learning for Physicists, Physics Reports **810**, 1 (2019)
  `https://arxiv.org/abs/1803.08823`
- S. Alexander *et al.*: The Physics of Machine Learning: An Intuitive Introduction for the Physical Scientist, preprint (2021)
  `https://arxiv.org/abs/2112.00851`
- G. Carleo *et al.*: Machine learning and the physical sciences, Rev. Mod. Phys. 91, 045002 (2019)
  `https://arxiv.org/abs/1903.10563`
- L. Zdeborová: Understanding deep learning is also a job for physicists, Nature Physics **16**, 602 (2020)
- A. Tanaka *et al.*: Deep Learning and Physics

Software

python™

NumPy

pandas

scikit learn

TensorFlow

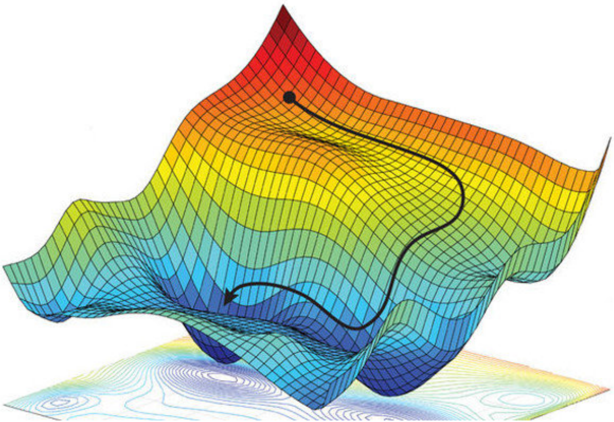
Software

- Jupyter
  `jupyter.org`
- Metacentrum
  `www.metacentrum.cz`
- JupyterHub
  `jupyter.cloud.metacentrum.cz`

Machine learning is an optimization problem

Optimization problem

- Find global minimum of a multimensional function $f(\boldsymbol{x})$, where $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$



Linear and Polynomial Regression

Linear regression

$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + \dots w_n x_n = f_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w}^\intercal \boldsymbol{x} \qquad \text{(model)}$$

- input: $\boldsymbol{x}^\intercal = (x_0, x_1, x_2, \dots, x_n)$, where $x_0 = 1$
- weigths: $\boldsymbol{w}^\intercal = (w_0, w_1, w_2, \dots, w_n)$, where w_0 is *bias*

Mean square error

$$\mathcal{L}_{\boldsymbol{w}}(\boldsymbol{X}, \boldsymbol{y}) = \frac{1}{m} \sum_{i=1}^{m} \left(\boldsymbol{w}^\intercal \boldsymbol{x}^{(i)} - y^{(i)} \right)^2 \qquad \text{(loss function)}$$

- trainig dataset: $\boldsymbol{X} = (\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(m)})$ is an input matrix, and $\boldsymbol{y} = (y^{(1)}, y^{(2)}, \dots, y^{(m)})$ is a vector of target values

➜ Our task: find w components in order to minimize $\mathcal{L}_w$ on the given training dataset $(\boldsymbol{X}, \boldsymbol{y})$

Machine learning terminology

$\hat{y} = f_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w}^\intercal \boldsymbol{x}$	model
$f_{\boldsymbol{w}}$	hypotesis function
$(\boldsymbol{X}, \boldsymbol{y})$	training dataset
$\boldsymbol{x}^{(k)}$	k-th sample / instance's feature vector
$x_i^{(k)}$	i-th feature
$y^{(k)}$	k-th target
$\boldsymbol{x}^{(k)} \in \mathbb{F} = \mathbb{I}_1 \times \mathbb{I}_2 \times \cdots \times \mathbb{I}_n$	$\mathbb{F}$ is the feature space
n	number of features / dimension of the feature space
$\hat{y}$	predicted value
w_j	j-th model parameter
w_0	bias term
$\mathcal{L}_{\boldsymbol{w}}(\boldsymbol{X}, \boldsymbol{y})$	loss function

Normal equation

$$\boldsymbol{w}^* = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal \boldsymbol{y}$$

- $\boldsymbol{w}^*$ minimizes the mean square error

Pseudoinverse

$$\boldsymbol{w}^* = \boldsymbol{X}^+ \boldsymbol{y}$$

- pseudoinverse or Moore-Penrose inverse $\boldsymbol{X}^+$
- Singular value decomposition (SVD): $\boldsymbol{X} = \boldsymbol{U} \boldsymbol{\Sigma} \boldsymbol{V}^\intercal$
- pseudoinverse $\boldsymbol{X}^+ = \boldsymbol{V} \boldsymbol{\Sigma}^+ \boldsymbol{U}^\intercal$

Linear Regression

Normal equation

$$\boldsymbol{w}^* = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal \boldsymbol{y}$$

- $\boldsymbol{w}^*$ minimizes the mean square error

Pseudoinverse

$$\boldsymbol{w}^* = \boldsymbol{X}^+ \boldsymbol{y}$$

- pseudoinverse or Moore-Penrose inverse $\boldsymbol{X}^+$
- Singular value decomposition (SVD): $\boldsymbol{X} = \boldsymbol{U}\,\boldsymbol{\Sigma}\,\boldsymbol{V}^\intercal$
- pseudoinverse $\boldsymbol{X}^+ = \boldsymbol{V}\,\boldsymbol{\Sigma}^+\,\boldsymbol{U}^\intercal$

Batch gradient descent

$$\frac{\partial \mathcal{L}_{\boldsymbol{w}}}{\partial w_j} = \frac{2}{m} \sum_{i=1}^{m} \left(\boldsymbol{w}^\intercal \boldsymbol{x}^{(i)} - y^{(i)} \right) x_j^{(i)}$$

$$\nabla_{\boldsymbol{w}} \mathcal{L}_{\boldsymbol{w}} = \begin{pmatrix} \frac{\partial \mathcal{L}_{\boldsymbol{w}}}{\partial w_0} \\ \frac{\partial \mathcal{L}_{\boldsymbol{w}}}{\partial w_1} \\ \vdots \\ \frac{\partial \mathcal{L}_{\boldsymbol{w}}}{\partial w_n} \end{pmatrix} = \frac{2}{m} \boldsymbol{X}^\intercal (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$$

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla_{\boldsymbol{w}} \mathcal{L}_{\boldsymbol{w}} \qquad \text{(update)}$$

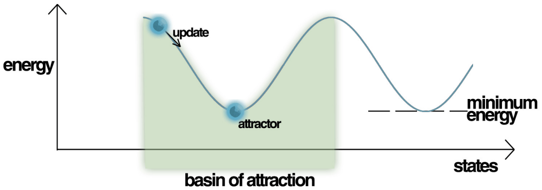
- Learning rate: $0 < \eta \ll 1$

Gradient Descent

- function $f_{\boldsymbol{w}}(\boldsymbol{x})$
- parameters $\boldsymbol{w} = (w_0, w_1, \ldots, w_n)$
- GD update

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \, \nabla_{\boldsymbol{w}} f_{\boldsymbol{w}} \qquad \text{(GD)}$$

- learning rate $0 < \eta \ll 1$



⚠ problem with local minima

Important Notes on Gradient Descent

- Linear Regression model is a convex function
  - there are no local minima, just one global minimum
  - it is a continuous function with a slope that never changes abruptly
- Gradient Descent is guaranteed to approach arbitrarily close the global minimum.

Important Notes on Gradient Descent

- Linear Regression model is a convex function
  - there are no local minima, just one global minimum
  - it is a continuous function with a slope that never changes abruptly
- Gradient Descent is guaranteed to approach arbitrarily close the global minimum.

- properly set the learning rate η: not too small, not too large

---

Important Notes on Gradient Descent

- Linear Regression model is a convex function
  - there are no local minima, just one global minimum
  - it is a continuous function with a slope that never changes abruptly
- Gradient Descent is guaranteed to approach arbitrarily close the global minimum.

- properly set the learning rate η: not too small, not too large

---

- When using regularization, always scale your data
- Standardize features by removing the mean and scaling to unit variance.
  - For any sample x of the training set calculate

$$z = (x - u)/s \qquad \text{(Standard Scaler)}$$

where u is the mean of the training samples and s is the standard deviation of the training samples

⚠ in Batch GD it takes a lot of computational time to calculate $\mathcal{L}_w(\boldsymbol{X}, \boldsymbol{y})$, especially, when the number of examples in $\boldsymbol{X}$ is large

Stochastic Gradient Descent (SGD)

- we do not use the whole batch for calculating $\mathcal{L}_w$
- we pick just one random instance from $\boldsymbol{X}$

(+) SGD is faster than Batch GD

(+) can avoid local minima (finds global minimum)

(−) less regular (never settles down in the minimum)

(−) some instances may be picked several times, while others may not be picked at all

💡 irregularity of SGD can be solved by learning schedule: gradually decrease η

Mini-Batch Gradient Descent

- calculate gradient on a small random set of instances of X

- ⊕ more regular than SGD
- ⊕ Mini-batch GD can be run in parallel
- ⊕ Mini-batch GD has an advantage on GPUs where matrix operations are optimized

Gradient Descent Methods

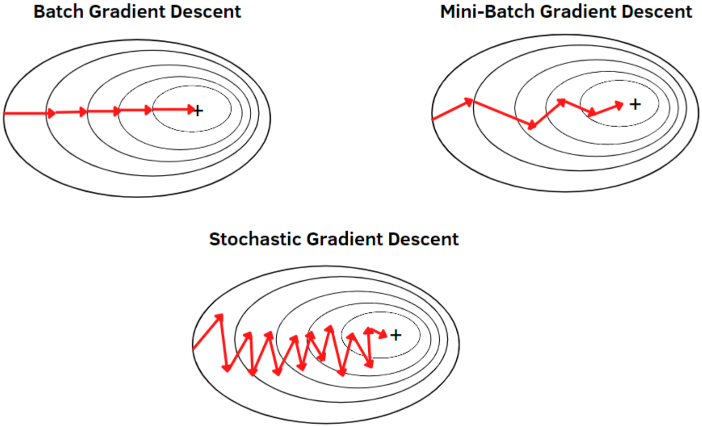
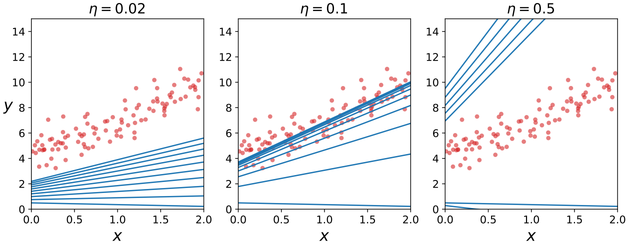


Figure source: analyticsvidhya.com

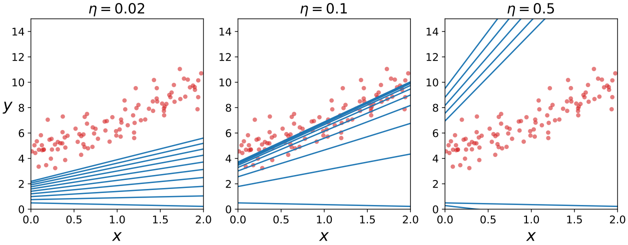
Linear Regression using Gradient Descent

- gradient descent

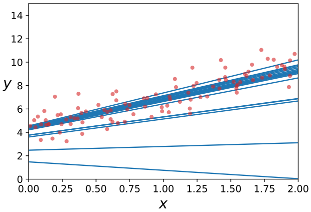


Linear Regression using Gradient Descent

- gradient descent



- stochastic gradient descent $\eta(t) = 5/(t + 50)$



Polynomial Regression: add polynomial features

- suppose a 1-dimensional dataset $\boldsymbol{X} = \{-3, -2, -1, 0, 1, 2, 3\}$ and $\boldsymbol{y}$ with measured values
- with Linear Regression one can fit the data as

$$y = a\,x + b \qquad \text{(1D Linear Regression)}$$

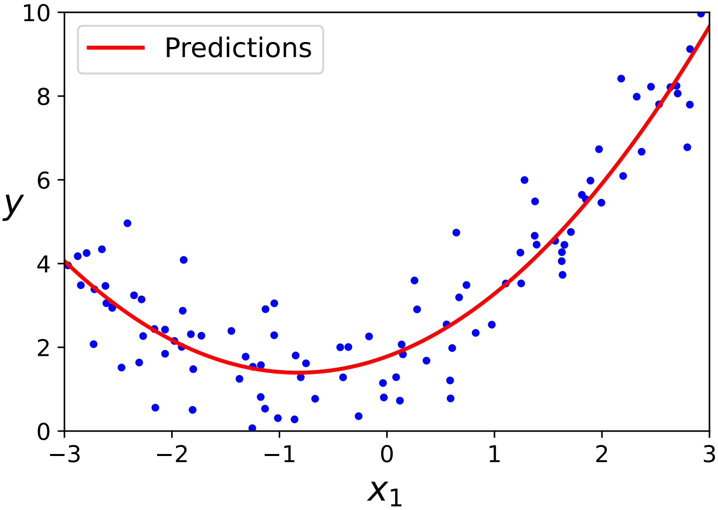
- if you assume that you need a higher degree polynomial curve to fit the data you can add polynomial feature
- in case of 2-nd degree polynomial one can extend the data to two dimensions $\boldsymbol{X}_2 = \{(-3, 9), (-2, 4), (-1, 1), (0, 0), (1, 1), (2, 4), (3, 9)\}$
- then we can fit the data using Linear Regression as

$$y = a\,x_1 + b\,x_2 + c \qquad \text{(2D Linear Regression)}$$

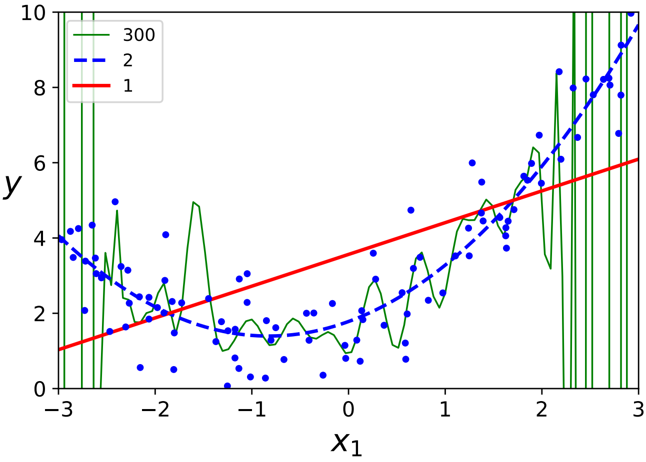
which is

$$y = b\,x^2 + a\,x + c \qquad \text{(Polynomial Regression)}$$

Polynomial Regression



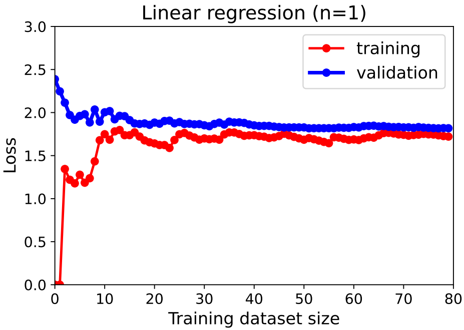
Higher degree polynomials



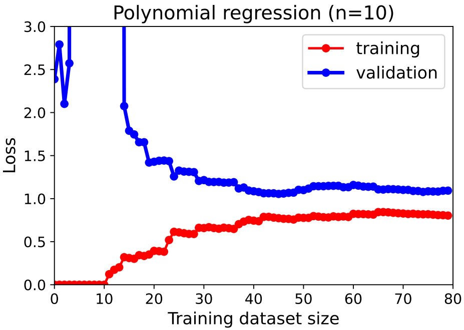
- ☹ $n = 1$ underfitting
- ☹ $n = 300$ overfitting
- ☺ $n = 2$ best fit

Learning Curves

underfitting



overfitting



The Bias/Variance Trade–off

- Model's generalization error can be expressed as a sum of three errors:

  - **Bias:** error due to a wrong assumption.
    A high-bias model is likely to underfit the training data.

  ↪ adding more training examples does not help. You need to use more complex model.

  - **Variance:** error due to the model's excessive sensitivity.
    Model with high variance is likely to overfit the trainig data.

  ↪ use more training data or reduce number of the fitting parameters.

  ↪ use regularized models.

  - **Irreducible error:** error due to noise in the data.

  ↪ It can be avoided by cleaning up the data (fix the data sources, such as broken detectors, or remove outliers).

Take Home Messages

- training a ML model means to solve an optimization problem
  - we minimize the loss function by setting the model's parameters
- Stochastic Gradient Descent is an efficient way of training
  - for Linear Regression it guarantees finding the global minimum
- we have to make a trade–off between bias and variance errors in order to decrease the generalization error
  - we need to analyze the the training process: learning curves