Supervised Learning

Chapter I: Introduction

Johannes Jurgovsky

1. Introduction

Introduction: Examples of Learning Tasks

Car Shopping Guide

👎

👍

?

Which criteria form the basis of a decision?

Introduction: Examples of Learning Tasks

Risk Analysis for Credit Approval

Customer 1	
house owner	yes
income (p.a.)	51 000 EUR
repayment (p.m.)	1 000 EUR
credit period	7 years
SCHUFA entry	no
age	37
married	yes
...	

...

Customer n	
house owner	no
income (p.a.)	55 000 EUR
repayment (p.m.)	1 200 EUR
credit period	8 years
SCHUFA entry	no
age	?
married	yes
...	

Introduction: Examples of Learning Tasks

Risk Analysis for Credit Approval

Customer 1	
house owner	yes
income (p.a.)	51 000 EUR
repayment (p.m.)	1 000 EUR
credit period	7 years
SCHUFA entry	no
age	37
married	yes
...	

...

Customer n	
house owner	no
income (p.a.)	55 000 EUR
repayment (p.m.)	1 200 EUR
credit period	8 years
SCHUFA entry	no
age	?
married	yes
...	

Learned rules:

```
IF     (income>40 000 AND credit_period<3) OR
       house_owner=yes
THEN   credit_approval=yes

IF     SCHUFA_entry=yes OR
       (income<20 000 AND repayment>800)
THEN   credit_approval=no
```

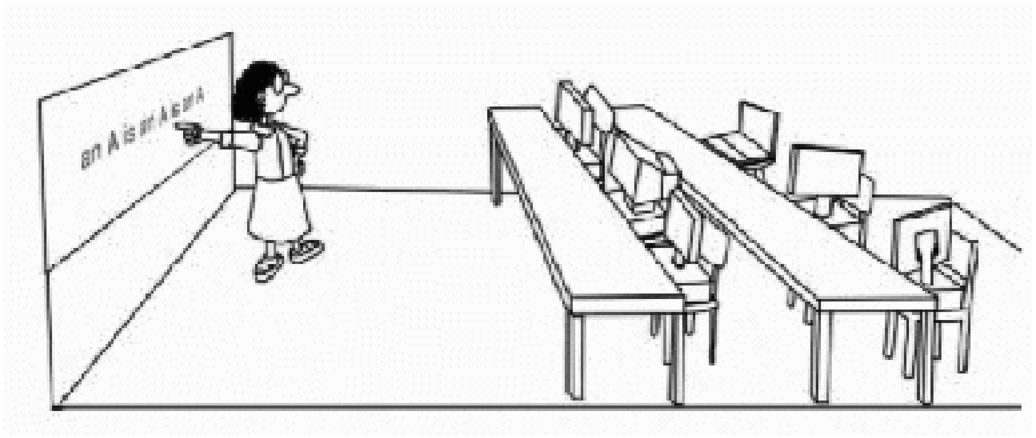
Introduction: Specification of Learning Problems

Definition 1 (Machine Learning [Mitchell 1997]**)**

A computer program is said to learn

- ❑ from experience
- ❑ with respect to some class of tasks and
- ❑ a performance measure,

if its performance at the tasks improves with the experience.

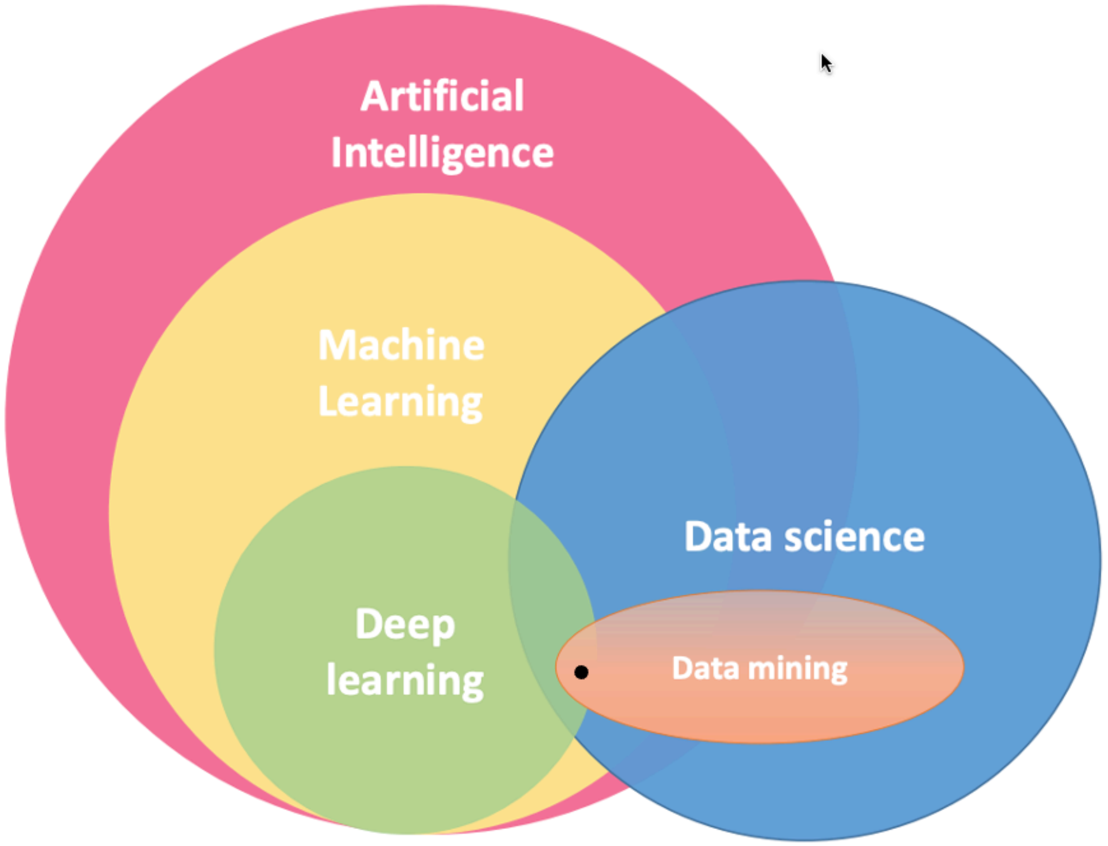


Remarks:

- ❑ Example chess
  - task = playing chess
  - performance measure = percentage of games won against opponent
  - experience = playing practice games against itself
- ❑ Example optical character recognition
  - task = isolation and classification of handwritten words in bitmaps
  - performance measure = percentage of correctly classified words
  - experience = collection of correctly classified, handwritten words
- ❑ A corpus with labeled examples forms a kind of “compiled experience”.
- ❑ Consider the different corpora that are exploited for different learning tasks in scikit-learn

Specification of Learning Problems

Learning Paradigms



- Agents in Environments
- Perceive, Think & Act
- Knowledge Representation
- Planning

- **<u>Supervised Learning</u>**
- Unsupervised Learning
- Reinforcement Learning

- Pre-Training
- Architectures
- Representation Learning

- Data Cleaning, Analysis, Visualization
- Knowledge Extraction
- Goal: Create value from data

[kulin2021]

Introduction: Specification of Learning Problems

Learning Paradigms

1. Supervised learning

2. Unsupervised learning

3. Reinforcement learning

Introduction: Specification of Learning Problems

Learning Paradigms

1. Supervised learning

   Learn a function from a set of input-output-pairs. An important branch of supervised learning is automated classification. Example: optical character recognition

2. Unsupervised learning

   Identify structures in data. Important subareas of unsupervised learning include automated categorization (e.g. via cluster analysis), parameter optimization (e.g. via expectation maximization), and feature extraction (e.g. via factor analysis).

3. Reinforcement learning

   Learn, adapt, or optimize a behavior strategy in order to maximize the own benefit by interpreting feedback that is provided by the environment. Example: development of behavior strategies for agents in a hostile environment.

Introduction: Specification of Learning Problems

Example Chess: Kind of Experience [Mitchell 1997]

1. Feedback
   - direct: for each board configuration the best move is given.
   - indirect: only a series of moves is given together with a final result (i.e. credit assignment problem).

Introduction: Specification of Learning Problems

Example Chess: Kind of Experience [Mitchell 1997]

1. Feedback
   - direct: for each board configuration the best move is given.
   - indirect: only a series of moves is given together with a final result (i.e. credit assignment problem).

2. Sequence and distribution of examples
   - A teacher presents important example problems along with a solution.
   - The learner chooses from the examples; e.g. pick a board for which the best move is unknown.
   - The selection of examples to learn from should follow the (expected) distribution of future problems.

Introduction: Specification of Learning Problems

Example Chess: Kind of Experience [Mitchell 1997]

1. Feedback
   - direct: for each board configuration the best move is given.
   - indirect: only a series of moves is given together with a final result (i.e. credit assignment problem).

2. Sequence and distribution of examples
   - A teacher presents important example problems along with a solution.
   - The learner chooses from the examples; e.g. pick a board for which the best move is unknown.
   - The selection of examples to learn from should follow the (expected) distribution of future problems.

3. Relevance under a performance measure
   - How far can we get with experience?
   - Can we master situations in the wild?
     (playing against itself will be not enough to become world class)

Introduction: Specification of Learning Problems

Example Chess: Ideal Target Function γ [Mitchell 1997]

(a) $\gamma : \mathit{Boards} \rightarrow \mathit{Moves}$

(b) $\gamma : \mathit{Boards} \rightarrow \mathbb{R}$

Introduction: Specification of Learning Problems

Example Chess: Ideal Target Function γ [Mitchell 1997]

(a) γ : *Boards* $\rightarrow$ *Moves*

(b) γ : *Boards* $\rightarrow \mathbb{R}$

A recursive definition of γ, following a kind of *means-ends analysis*:

Let be $o \in$ *Boards*.

1. $\gamma(o) = 100$, if o represents a final board state that is won.
2. $\gamma(o) = -100$, if o represents a final board state that is lost.
3. $\gamma(o) = 0$, if o represents a final board state that is drawn.
4. $\gamma(o) = \gamma(o^*)$ otherwise.

o^* denotes the best final state that can be reached if both sides play optimally.
Related: game playing, minimax strategy, α-β pruning.

Introduction: Specification of Learning Problems

Example Chess: From the Real World γ to a Model World h

$$\gamma(o) \;\rightsquigarrow\; h(\alpha(o)) \equiv h(\mathbf{x})$$

Introduction: Specification of Learning Problems

Example Chess: From the Real World γ to a Model World h

$$\gamma(o) \;\rightsquigarrow\; h(\alpha(o)) \equiv h(\mathbf{x})$$

$$h(\mathbf{x}) = w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3 + w_4 \cdot x_4 + w_5 \cdot x_5 + w_6 \cdot x_6$$

where

x_1	= number of black pawns on board o
x_2	= number of white pawns on board o
x_3	= number of black pieces on board o
x_4	= number of white pieces on board o
x_5	= number of black pieces threatened on board o
x_6	= number of white pieces threatened on board o

Introduction: Specification of Learning Problems

Example Chess: From the Real World γ to a Model World h

$$\gamma(o) \rightsquigarrow h(\alpha(o)) \equiv h(\mathbf{x})$$

$$h(\mathbf{x}) = w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3 + w_4 \cdot x_4 + w_5 \cdot x_5 + w_6 \cdot x_6$$

where

x_1	= number of black pawns on board o
x_2	= number of white pawns on board o
x_3	= number of black pieces on board o
x_4	= number of white pieces on board o
x_5	= number of black pieces threatened on board o
x_6	= number of white pieces threatened on board o

Other approaches to formulate h:

- case base
- set of rules
- neural network
- polynomial function of board features

Remarks:

- ❑ The *ideal target function* γ interprets the real world, say, a real-world object o, to "compute" $\gamma(o)$. This "computation" can be operationalized by a human or by some other (even arcane) mechanism of the real world.
- ❑ To simulate the interesting aspects of the real world by means of a computer, we define a model world. This model world is restricted to particular—typically easily measurable—features $\mathbf{x}$ that are derived from o, with $\mathbf{x} = \alpha(o)$. In the model world, $h(\mathbf{x})$ is the formalized counterpart of $\gamma(o)$.
- ❑ h is called *model function* or model, α is called *model formation function*.
- ❑ The key difference between an ideal target function γ and a model function h lies in the size and the representation of their respective domains. Examples:
  - A chess grand master assesses a board o in its entirety, both intuitively and analytically; a chess program is restricted to particular features $\mathbf{x}$, $\mathbf{x} = \alpha(o)$.
  - A human mushroom picker assesses a mushroom o with all her skills (intuitively, analytically, by tickled senses); a classification program is restricted to a few surface features $\mathbf{x}$, $\mathbf{x} = \alpha(o)$.

Remarks (continued) :

- For automated chess playing a real-valued assessment function is needed; such kind of problems form regression problems. If only a small number of values are to be considered (e.g. school grades), we are given a classification problem. A regression problem can be transformed into a classification problem by means of domain discretization.
- Regression problems and classification problems often differ with regard to assessing the achieved accuracy or goodness of fit. For regression problems the sum of the squared residuals may be a sensible criterion; for classification problems the number of misclassified examples may be more relevant.
- For classification problems, the ideal target function γ is also called ideal *classifier*; similarly, the model function h is called classifier.
- Decision problems are classification problems with two classes.
- The halting problem for Turing machines is an undecidable classification problem.

Introduction: Specification of Learning Problems [model world]

How to Construct a Classifier h

Characterization of the real world:

- O is a set of objects.
- Y is a set of possible assignments.
- $\gamma : O \to Y$ is the ideal classifier for O.

Introduction: Specification of Learning Problems

How to Construct a Classifier h

Characterization of the real world:

- O is a set of objects.
- Y is a set of possible assignments.
- $\gamma : O \rightarrow Y$ is the ideal classifier for O.

Assignment problem:

- Given some $o \in O$, determine its assignment $\gamma(o) \in C$.

Acquisition of classification knowledge:

1. Build a database of examples of the form $(o, \gamma(o))$, $o \in O_D$, $O_D \subseteq O$.
2. Abstract the objects $o \in O_D$ towards feature vectors $\mathbf{x} \in X$, with $\mathbf{x} = \alpha(o)$.
3. Compute $(\mathbf{x}, y(\mathbf{x}))$, with $\mathbf{x} = \alpha(o)$ and $y(\mathbf{x})$ defined as $\gamma(o)$, $o \in O_D$.

Introduction: Specification of Learning Problems

How to Construct a Classifier h (continued)

Characterization of the model world:

- X is a set of feature vectors, also called feature space.
- Y is a set of assignments.
- $y : X \to Y$ is the ideal assignment for X.
- $D = \{(\mathbf{x}_1, y(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, y(\mathbf{x}_n))\} \subseteq X \times Y$ is a set of examples.

Introduction: Specification of Learning Problems

How to Construct a Classifier h (continued)

Characterization of the model world:

- X is a set of feature vectors, also called feature space.
- Y is a set of assignments.
- $y : X \to Y$ is the ideal assignment for X.
- $D = \{(\mathbf{x}_1, y(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, y(\mathbf{x}_n))\} \subseteq X \times Y$ is a set of examples.

Definition 2 (Supervised Learning System)

Let $h_\Theta : X \to Y$ be a model function with parameters Θ and let $L_{D'}(h_\Theta, y)$ measure the loss between model function and the ideal assignment y over a set of examples $D' \subseteq D$. Then a supervised learning system aims to find the best parameters Θ of the modelling function h minimizing the loss $L_{D'}(h_\Theta, y)$, i.e.

$$h_\Theta = \underset{\Theta}{\text{argmin}}\, L_{D'}(h_\Theta, y)$$

Remarks:

- The feature space X comprises vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots$, which can be considered as abstractions of real-world objects $o_1, o_2, \ldots$, and which have been computed according to our view of the real world.

- The model formation function α determines the level of abstraction between o and $\mathbf{x}$, $\mathbf{x} = \alpha(o)$. I.e., α determines the representation fidelity, exactness, quality, or simplification.

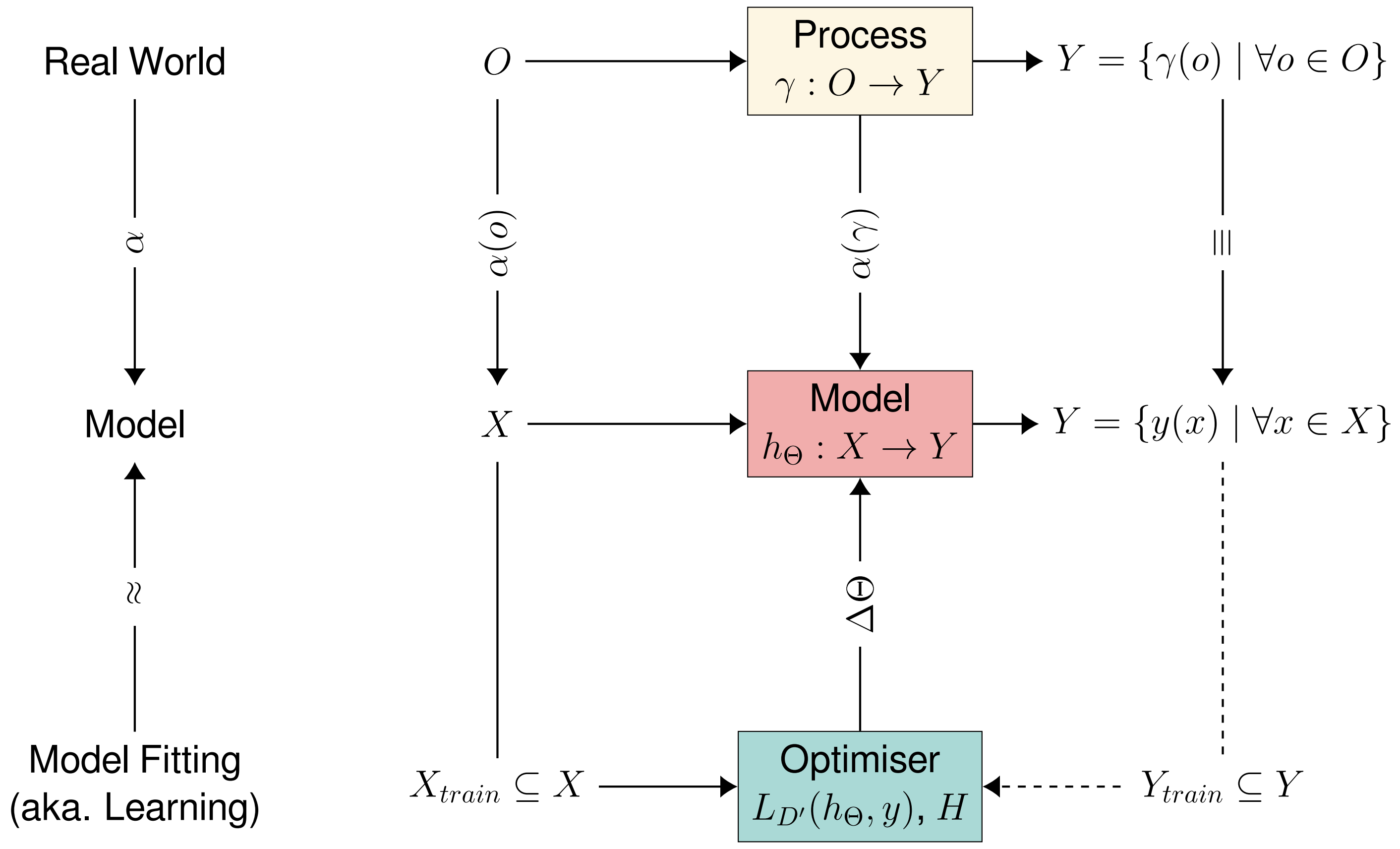
- Though α models an object $o \in O$ only imperfectly as $\mathbf{x} = \alpha(o)$, $y(\mathbf{x})$ must be considered as *ideal* classifier, since $y(\mathbf{x})$ is defined as $\gamma(o)$ and hence yields the true real-world classes. I.e., y and γ have different domains each, but they return the same images.

- $y(\mathbf{x})$ is often termed “ground truth” (for $\mathbf{x}$ and the underlying classification problem). Observe that this term is justified by the fact that $y(\mathbf{x}) \equiv \gamma(o)$.

- In a multi-class classification setting, the set of classes/labels Y contains k discrete class labels $\{1, \cdots, k\}$. In a binary classification setting, Y may be defined as $\{0, +1\}$ or $\{-1, +1\}$. In a regression setting, Y may cover entire $\mathbb{R}$ or some subset, e.g. $\mathbb{R}^+$.

Introduction: Specification of Learning Problems

Summary of a Supervised Learning System



Real World
α
Model
$\approx$
Model Fitting (aka. Learning)
O
$\alpha(o)$
X
$X_{train} \subseteq X$
Process
$\gamma : O \rightarrow Y$
$\alpha(\gamma)$
Model
$h_\Theta : X \rightarrow Y$
$\Delta\Theta$
Optimiser
$L_{D'}(h_\Theta, y)$, H
$Y = \{\gamma(o) \mid \forall o \in O\}$
$\equiv$
$Y = \{y(x) \mid \forall x \in X\}$
$Y_{train} \subseteq Y$

The optimizer is usually an algorithm taking in additional parameters H. Those parameters are called **Hyperparameters**.

Introduction: Specification of Learning Problems

Terminology & Assumptions

Parameterized classification function h_Θ:

- The classification function h is parameterized by a vector of parameters Θ
- These parameters are adjusted during learning on D such that $h \approx y$. (We include the subscript only in situations when it's relevant to the subject.)

Generalization error:

- Suppose we had some means to measure the error $E(h_\Theta, y)$ of h_Θ 's predictions.
- We denote the error induced on the data D used for parameter estimation as E_{train} and the error induced on new (yet unseen) data as E_{new}.
- The generalization error is defined as $E_{new} - E_{train}$ and it reflects the expected error on unseen data points.

Categorization of supervised learning problems:

- If Y is *qualitative/discrete*, we refer to the problem as Classification problem.
- If Y is *quantitative/continuous*, we refer to the problem as Regression problem.

Introduction: Specification of Learning Problems

Core Design Issues

Model Formation α

- ❑ What is a good representation X for the real world data O?
- ❑ What properties of O are important for γ ?
- ❑ What model function h_θ could approximate γ?

$\Rightarrow$ Modelling Error

Knowledge Acquisition D

- ❑ How well does our data set D represent the true distribution of objects O?
- ❑ What is the quality and quantity of our data set D?

$\Rightarrow$ Sampling Error

Modell Fitting

- ❑ How well does our loss function $L(h_\Theta, y)$ represent the true loss?
- ❑ Do we find the optimal parameters Θ?

$\Rightarrow$ Optimization Error / Generalization Error

Introduction: Specification of Learning Problems

Questions beyond a single Supervised Learning System

Model functions h:

- What are important classes of model functions?
- What are methods to fit (= learn) model functions?
- What are measures to assess the goodness of fit?
- How does the example number affect the learning process?
- How does noise affect the learning process?

Introduction: Specification of Learning Problems

Questions beyond a single Supervised Learning System (continued)

Generic learnability:

- What are the theoretical limits of learnability?
- How can we use nature as a model for learning?

Knowledge acquisition:

- How can we integrate background knowledge into the learning process?
- How can we integrate human expertise into the learning process?

Introduction: Example: Example Learning Algorithm

LMS Algorithm for Fitting h [IGD Algorithm]

Algorithm:	*LMS*	Least Mean Squares.
Input:	D	Training examples of the form $(\mathbf{x}, y(\mathbf{x}))$ with target function value $y(\mathbf{x})$ for $\mathbf{x}$.
	η	Learning rate, a small positive constant.
Internal:	$h(D)$	Set of $h(\mathbf{x})$-values computed from the elements $\mathbf{x}$ in D given some $\mathbf{w}$.
Output:	$\mathbf{w}$	Weight vector.

```
LMS$(D, \eta)$

 1. initialize_random_weights$((w_0, w_1, \ldots, w_p))$
 2. REPEAT
 3.   $(\mathbf{x}, y(\mathbf{x})) = $ random_select$(D)$
 4.   $h(\mathbf{x}) = w_0 + w_1 \cdot x_1 + \ldots + w_p \cdot x_p$
 5.   error $= y(\mathbf{x}) - h(\mathbf{x})$
 6.   FOR $j = 0$ TO $p$ DO
 7.     $\Delta w_j = \eta \cdot$ error $\cdot x_j$    // $\forall_{\mathbf{x} \in D} : \mathbf{x}|_{x_0} \equiv 1$
 8.     $w_j = w_j + \Delta w_j$
 9.   ENDDO
10. UNTIL(convergence$(D, y(D))$)
11. return$((w_0, w_1, \ldots, w_p))$
```

Introduction: Simple Example

A Simple Supervised Learning System

Lets make a system that learns to classify flowers

1. O - Flowers
2. $Y = \{"Rose","Iris"\}$
3. $\gamma(o)$ - The kind of flower $y \in Y$ for a collected flower o
4. α - Measure the properties of the flower: length, width, weight, diameter
5. X - properties of all flowers collected

	length	width	weight	diameter	flower
1	5.1	3.5	1.4	0.2	rose
2	4.9	3.	1.4	0.2	iris
3	4.7	3.2	1.3	0.2	iris
4	4.6	3.1	1.5	0.2	rose
...	...	...	...	...	...

6. Learning algorithm: LMS - Least Mean Square Fit
7. Loss function: Mean Squared Error

$$L_{D'}(h_\Theta, y) = \sum_{D'} (h_\Theta(x) - y(x))^2$$

Introduction: Machine Learning Principles

Machine Learning as Search in a Hypothesis Space

- Our classifier h depends on parameters that are learned from observed data (e.g. weight vector w).
- All possible values of the parameter are called the hypothesis space, usually denoted as $\mathcal{H}$, which contains all possible solution (according to our model).
- Machine learning can be understood as finding the “best” hypothesis $h_\Theta \in \mathcal{H}$ effectively.
  - **Exhaustive Search**: we can search the full hypothesis space $\mathcal{H}$. If the hypothesis space is small this can be done efficiently (there are some exceptions also for large hypothesis spaces)
  - **Heuristic search**: we take some assumption to only search parts of the solution space. We might miss the perfect solution, but are therefore efficient.
  - Except for very simple problems you can not have both: searching **all possible hypothesis** in an **efficient and robust manner**.

Introduction: Machine Learning Principles

Inductive Reasoning and Inductive Bias

- **Inductive Reasoning:** Machine learning aims to generalize over **unseen** examples based on seen examples. It is a formm of inductive reasoning, where we aim to find a general pattern (encoded as hypothesis from the hypothesis space $\mathcal{H}$) based on past observations, that are valid to predict the outcome of future observations.
  - Find the best weights $w \in \Re^d$ to judge the chess board configuration
  - Find the best rules to classify cars
- **Inductive Bias:** Are the assumptions we make (e.g. heuristic) by which the algorithm generalizes over the seen example. Without an inductive bias, a system could not generalize over unseen examples.
  - The defined hypothesis space is already an inductive bias
  - Further assumptions: noise, outliers, data distribution, heuristic, error distribution etc.

Introduction: Machine Learning Principles

Inductive Reasoning and Inductive Bias

? What is your inductive bias: How would you generalize the following observations:

1, 2, 3, 4, ?

Introduction: Machine Learning Principles

Bias-Variance Trade-Off

A model aims to generalize well over unseen example and therefore we need to make assumptions how the unseen data is distributed. Learning requires a trade-off between **fitting the seen data perfectly** and **sticking to prior-assumptions about the unseen data distribution**

- Bias determines an algorithms tendency to consistently favor particular outcomes independent of the data (i.e. the Stubborn Learner).
- Variance determiens an algorithms tendency to learn irregularities in data.
- Usually the higher the number of parameters Θ, the higher the variance of the model.
- A high-bias model tends to be more robust agaist noise in the observed data but might not be able to capture the important regularities. The model might **underfits** the data.
- A high-variance model captures patterns in data better, but is easier affected by noise or unimportant patterns. The model **overfits** the data.
- Loosely speaking: When applying machine learning techniques, there is always a trade-off between bias (underfitting due to a small hypothesis space) and variance (overfitting due to a large hypothesis space)

Introduction: Machine Learning Principles

Bias-Variance Trade-Off

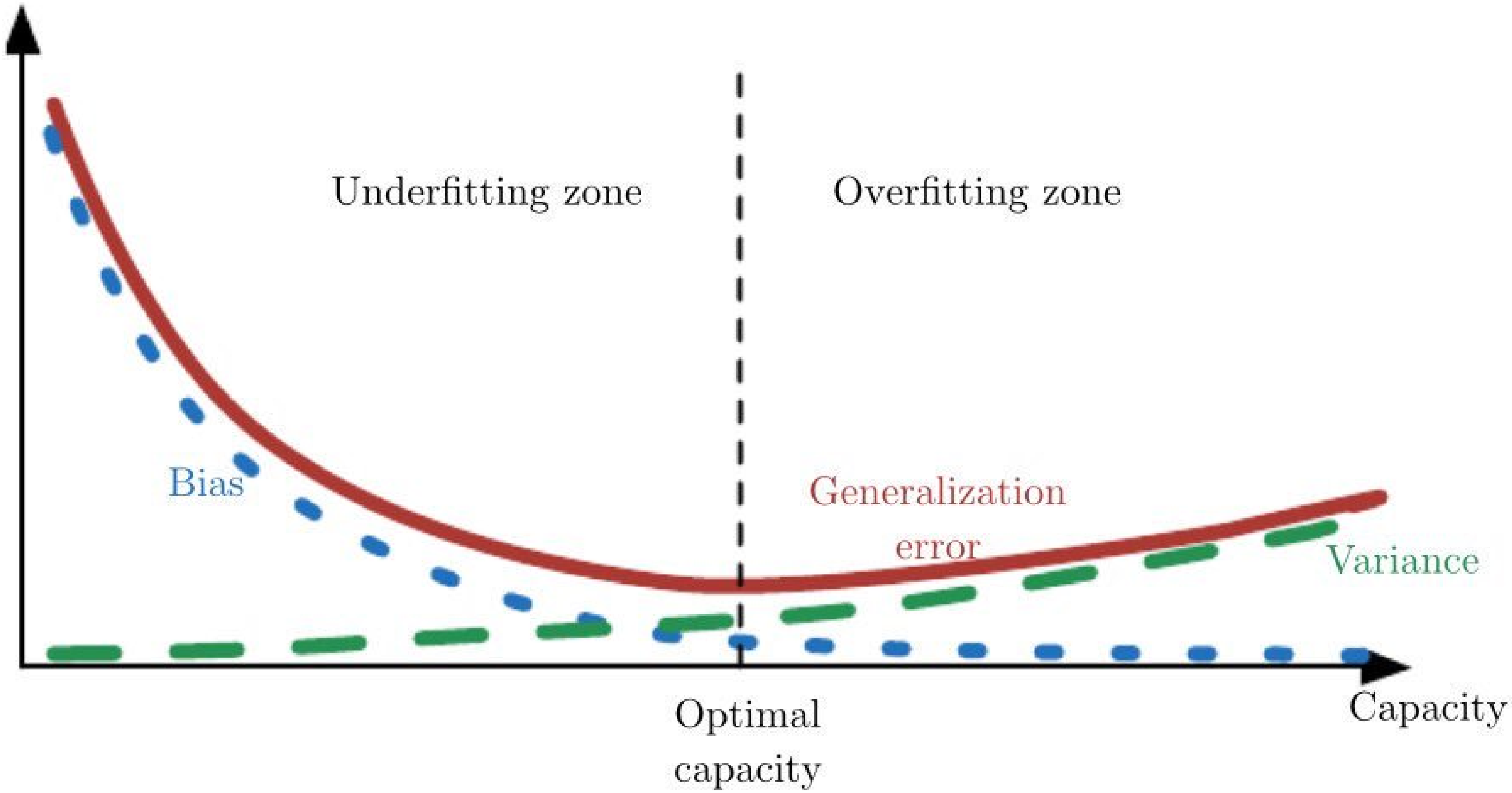


Image Credit: Ian Goodfellow and Yoshua Bengio and Aaron Courville, `http://deeplearningbook`.

Introduction: Machine Learning Principles

No-Free-Lunch

There is no single algorithm, that is on average better than another algorithms over all data distributions and target functions.

- "So when an algorithm performs well on a certain type of problems, it necessarily pays for that with degraded performance on the set of all remaining problems." [WM97]
- Analytical solutions, which algorithms perform well on which problems, is usually impossible.
- Usually applying machine learning in real-world setting requires data-driven trial-and-error.

References

[Dom12] Pedro Domingos. A few useful things to know about machine learning. *Commun. ACM*, 55(10):78–87, October 2012.

[KKDPM21] Merima Kulin, Tarik Kazaz, Eli De Poorter, and Ingrid Moerman. A survey on machine learning-based performance improvement of wireless networks: Phy, mac and network layer. *Electronics*, page 318, 01 2021.

[M$^+$97] Tom M Mitchell et al. *Machine learning*. McGraw-Hill Boston, MA, 1997.

[WM97] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.