Chapter ML:I

I. Introduction

- Examples of Learning Tasks
- Specification of Learning Tasks
- Elements of Machine Learning
- Notation Overview
- Classification Approaches Overview

Examples of Learning Tasks

(1) Car Shopping Guide

?

What criteria form the basis of a decision?

Examples of Learning Tasks

(2) 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
...	

Examples of Learning Tasks

(2) 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
...	

...

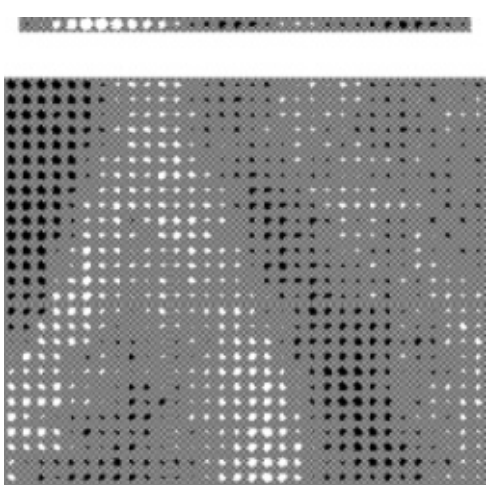
Customer n	
house owner	no
income (p.a.)	55 000 EUR
repayment (p.m.)	1 200 EUR
credit period	8 years
SCHUFA entry	no
age	?
married	yes
...	

Learned rules:

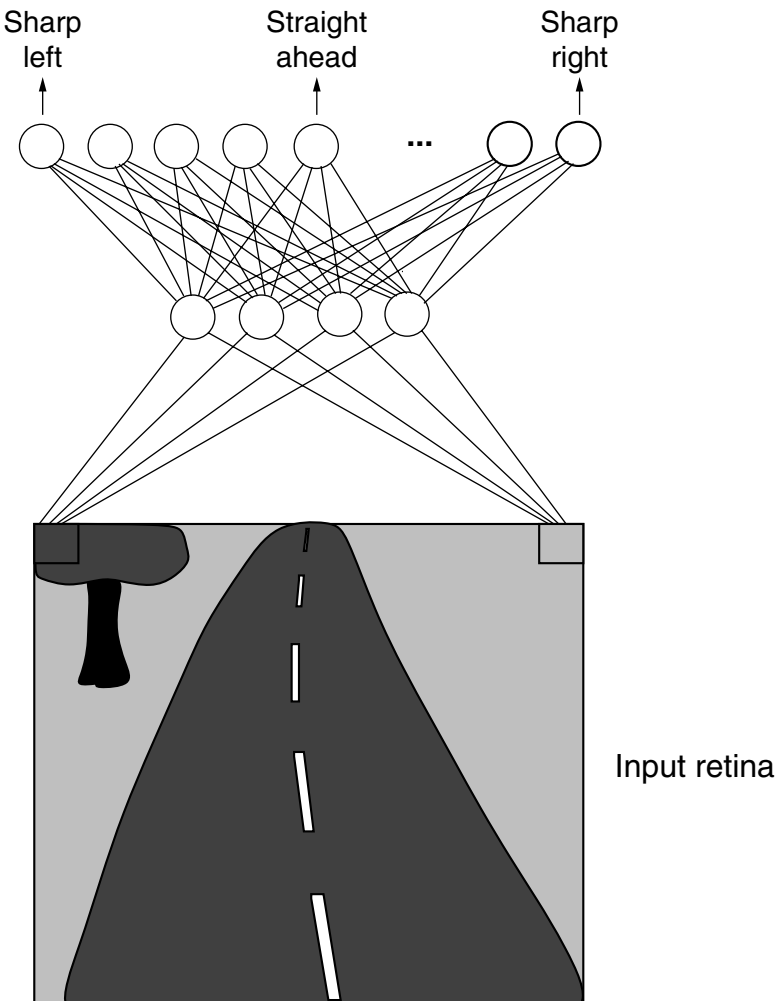
```
IF    ( income>40 000 AND credit_period<3 )  OR  house_owner=yes
THEN  credit_approval=yes

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

...
```

Examples of Learning Tasks

(3) Image Analysis [Mitchell 1997, p.84]

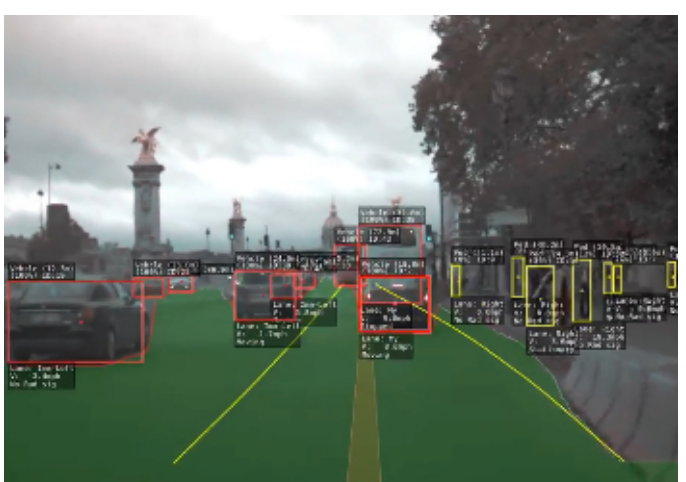


[1992]

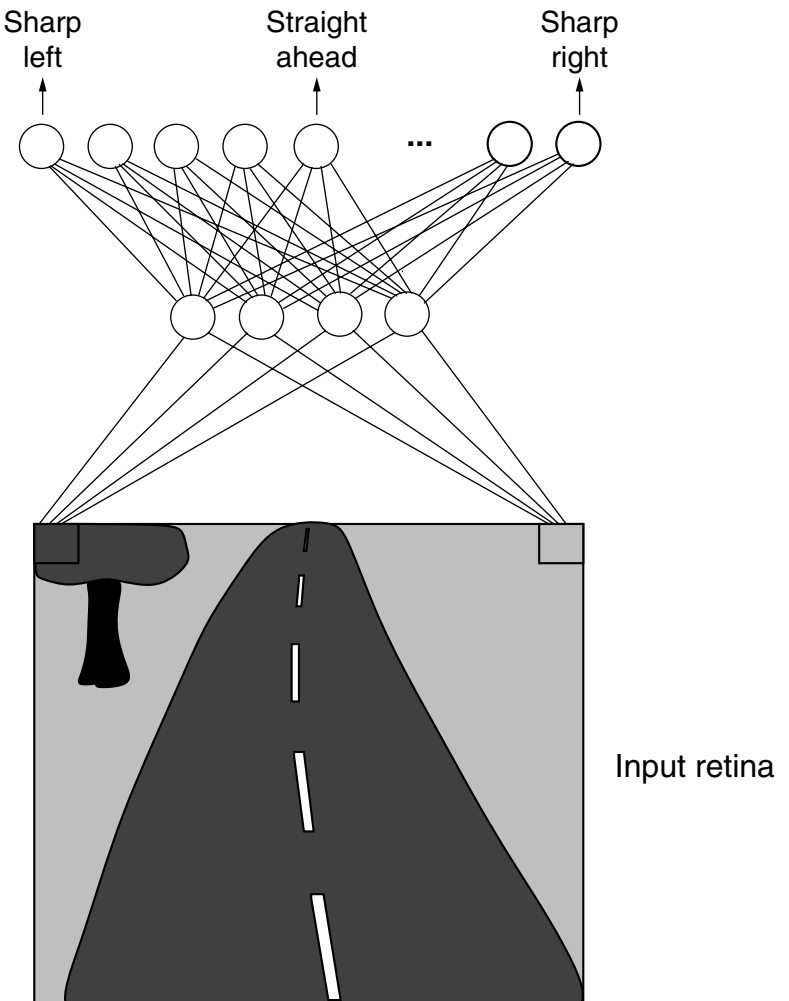


Examples of Learning Tasks

(3) Image Analysis [Mitchell 1997, p.84]



[2018]



Chapter ML:I

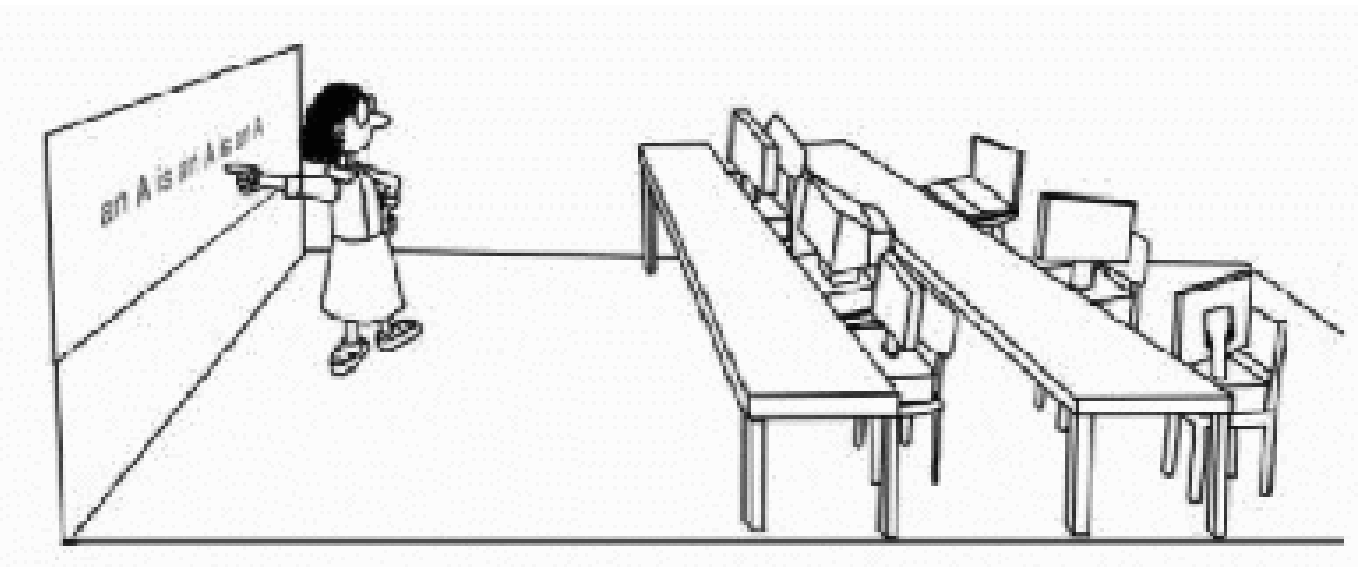
Specification of Learning Tasks

Definition 1 (Machine Learning [Mitchell 1997, p.2]**)**

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 = points scored in a tournament
  - experience = possibility to do self play
- 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
- In this context, the term “performance” refers to the effectiveness of the computer program to solve the task correctly and not to its efficiency in terms of runtime.
- A data set (a corpus) with labeled examples forms a kind of “compiled experience”.
- Consider the different data sets that are developed and exploited for different learning tasks in the Webis Group. [webis.de/data.html]

Specification of Learning Tasks

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).

   Example: intrusion detection in a network data stream

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 in a hostile environment

Specification of Learning Tasks

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).

   Example: intrusion detection in a network data stream

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 in a hostile environment

Specification of Learning Tasks

(4) Example Chess: Kinds of Experience [Mitchell 1997, p.5]

1. Feedback
   - direct: for each position, the best move is given.
   - indirect: only the final result is given after a series of moves.

Specification of Learning Tasks

(4) Example Chess: Kinds of Experience [Mitchell 1997, p.5]

1. Feedback
   - direct: for each position, the best move is given.
   - indirect: only the final result is given after a series of moves.

2. Sequence and distribution of examples
   - A teacher presents important example problems along with a solution.
   - The learner chooses from the examples; e.g., picks a position for which the best move is unknown.

The selection of examples to learn from should follow the (expected) distribution of future problem instances.

Specification of Learning Tasks

(4) Example Chess: Kinds of Experience [Mitchell 1997, p.5]

1. Feedback
   - direct: for each position, the best move is given.
   - indirect: only the final result is given after a series of moves.

2. Sequence and distribution of examples
   - A teacher presents important example problems along with a solution.
   - The learner chooses from the examples; e.g., picks a position for which the best move is unknown.

   The selection of examples to learn from should follow the (expected) distribution of future problem instances.

3. Relevance under a performance measure
   - How far can we get with experience?
   - Can we master situations in the wild?
     (self play may not be enough for a human to become a grandmaster)

Specification of Learning Tasks

(4) Example Chess: Ideal Target Function $\gamma()$ [Mitchell 1997, p.7]

a) $\gamma : \textit{Positions} \rightarrow \textit{Moves}$

b) $\gamma : \textit{Positions} \rightarrow \mathbf{R}$

Specification of Learning Tasks

(4) Example Chess: Ideal Target Function $\gamma()$ [Mitchell 1997, p.7]

a) $\gamma : \mathit{Positions} \rightarrow \mathit{Moves}$

b) $\gamma : \mathit{Positions} \rightarrow \mathbf{R}$

A recursive definition of $\gamma()$, following a kind of *means-end analysis*:

Let be $o \in \mathit{Positions}$.

1. $\gamma(o) = \quad 1$, if o represents a final position that is won.
2. $\gamma(o) = -1$, if o represents a final position that is lost.
3. $\gamma(o) = \quad 0$, if o represents a final position that is drawn.
4. $\gamma(o) = \gamma(o^*)$, with $\gamma(o^*) \in (-1; 1)$, otherwise.

o^* denotes the best final state that can be reached if both sides play optimally.
Related: minimax strategy, α-β pruning. [Course on Search Algorithms]

Specification of Learning Tasks

(4) Example Chess: Real World $\gamma()$ $\rightarrow$ Model World $y()$

$\gamma(o) \rightarrow y\big(\alpha(o)\big) \equiv y(\mathbf{x})$

$y()$ is called "model function"

$\alpha()$ is called "model formation function"

Specification of Learning Tasks

(4) Example Chess: Real World $\gamma()$ $\rightarrow$ Model World $y()$

$\gamma(o) \rightarrow y\big(\alpha(o)\big) \equiv y(\mathbf{x})$

$y()$ is called "model function"

$\alpha()$ is called "model formation function"

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

where

$x_{1,\,2}$ = number of black / white pawns in position o

$x_{3,\,4}$ = number of black / white pieces in position o

$x_{5,\,6}$ = number of black / white pieces attacked in position o

D = $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, a multiset of positions $\mathbf{x}_i$ with scores $y_i, y_i \in [-1; 1]$.

Specification of Learning Tasks

(4) Example Chess: Real World $\gamma()$ $\rightarrow$ Model World $y()$

$\gamma(o) \rightarrow y\big(\alpha(o)\big) \equiv y(\mathbf{x})$

$y()$ is called "model function"

$\alpha()$ is called "model formation function"

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

where

- $x_{1,\,2}$ = number of black / white pawns in position o
- $x_{3,\,4}$ = number of black / white pieces in position o
- $x_{5,\,6}$ = number of black / white pieces attacked in position o
- D = $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, a multiset of positions $\mathbf{x}_i$ with scores $y_i, y_i \in [-1; 1]$.

Other approaches to specify a model function:

- case base
- set of rules
- neural network
- higher order polynomial of board features

Remarks:

- $\gamma : O \to \mathbf{R}$, $o \mapsto \gamma(o)$, is called *ideal target function*. The ideal target function interprets the real world, say, a real-world object o, to "compute" $\gamma(o)$. This "computation" can be done 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 consider a model world. This model world is restricted to particular and typically easily measurable features $\mathbf{x}$, which are derived from o, with $\mathbf{x} = \alpha(o)$. In the model world, $y(\mathbf{x})$ is the abstracted and formalized counterpart of $\gamma(o)$.
- $y : \mathbf{X} \to \mathbf{R}$, $\mathbf{x} \mapsto y(\mathbf{x})$, is called *model function* or *model*. The choice and computation of a suitable model function is a central aspect in the field of machine learning.
- $\alpha : O \to \mathbf{X}$, $o \mapsto \alpha(o)$, is called *model formation function*. The development of a suitable model formation function is often not treated as part of machine learning but "outsourced" to respective domain experts. However, tackling advanced learning tasks such as autonomous driving, automated debating, or playing chess requires a tight cooperation among the developers of $\alpha()$ and $y()$.
- The difference between an ideal target function $\gamma()$ and a model function $y()$ lies in the complexity and the representation of their respective domains. Examples:
  - A chess grandmaster assesses a position 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 is needed; such kind of tasks form regression problems. If only a small set of values, here denoted as C, 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 domain discretization.
- Regression problems and classification problems differ in the way how an achieved accuracy or goodness of fit is assessed. For regression problems the sum of the squared residuals, RSS, may be a sensible criterion; for classification problems the number of misclassified examples is more relevant. Keywords: *regression loss* versus *classification loss*
- For classification problems, the ideal target function, $\gamma : O \to C$, is also called *ideal classifier*; analogously, the model function, $y : \mathbf{X} \to C$, is called classifier.
- Decision problems are classification problems with two classes.
- The halting problem for Turing machines is an undecidable classification problem.

Specification of Learning Tasks [model world]

Real World $\rightarrow$ Model World

Setting of the real world:

- O is a set of objects. (example: emails)
- C is a set of classes. (example: spam versus ham)
- $\gamma : O \rightarrow C$ is the ideal classifier. ($\gamma()$ is operationalized by a human expert)

Classification task:

- Given some $o \in O$, determine its class $\gamma(o) \in C$. (example: is an email spam?)

Specification of Learning Tasks [model world]

Real World $\rightarrow$ Model World

Setting of the real world:

- O is a set of objects. (example: emails)
- C is a set of classes. (example: spam versus ham)
- $\gamma : O \rightarrow C$ is the ideal classifier. ($\gamma()$ is operationalized by a human expert)

Classification task:

- Given some $o \in O$, determine its class $\gamma(o) \in C$. (example: is an email spam?)

Acquisition of classification knowledge D:

1. Collect real-world examples of the form $(o, \gamma(o)), \ o \in O$.
2. Abstract the objects towards feature vectors $\mathbf{x} \in \mathbf{X}$, where $\mathbf{x} := \alpha(o)$.
3. Form examples as $(\mathbf{x}, c)$, where $\mathbf{x} = \alpha(o)$ and $c := \gamma(o)$.

Specification of Learning Tasks [real world]

Real World → Model World (continued)

Setting of the model world:

- X is a multiset of feature vectors. (example: word frequency vectors)
- C is a set of classes. (as before: spam versus ham)
- $D = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_n, c_n)\} \subseteq X \times C$ is a multiset of examples.

Learning task:

- Fit D using a suited model function $y()$.

Specification of Learning Tasks [real world]

Real World → Model World (continued)

Setting of the model world:

- X is a multiset of feature vectors. (example: word frequency vectors)
- C is a set of classes. (as before: spam versus ham)
- $D = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_n, c_n)\} \subseteq X \times C$ is a multiset of examples.

Learning task:

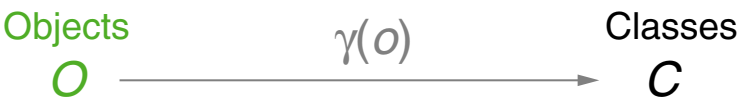
- Fit D using a suited model function $y()$.

Machine learning:

1. Formulate a model function $y : \mathbf{X} \to C,\ \mathbf{x} \mapsto y(\mathbf{x})$ ($y()$ needs to be fitted)
2. Apply statistics, theory, and algorithms from the field of machine learning to maximize the goodness of fit between $(\mathbf{x}, c)$ and $(\mathbf{x}, y(\mathbf{x}))$, $(\mathbf{x}, c) \in D$.

Specification of Learning Tasks

Real World $\rightarrow$ Model World (continued)

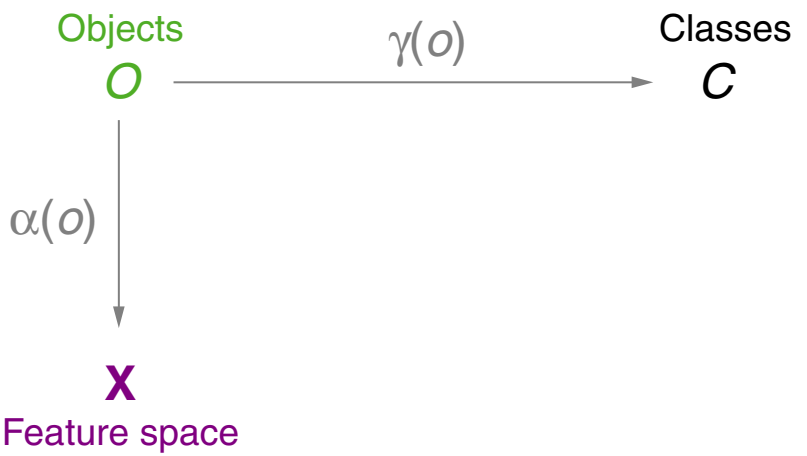


Semantics:

$\gamma(o)$ Ideal classifier (a human) for real-world objects.

Specification of Learning Tasks

Real World $\rightarrow$ Model World (continued)

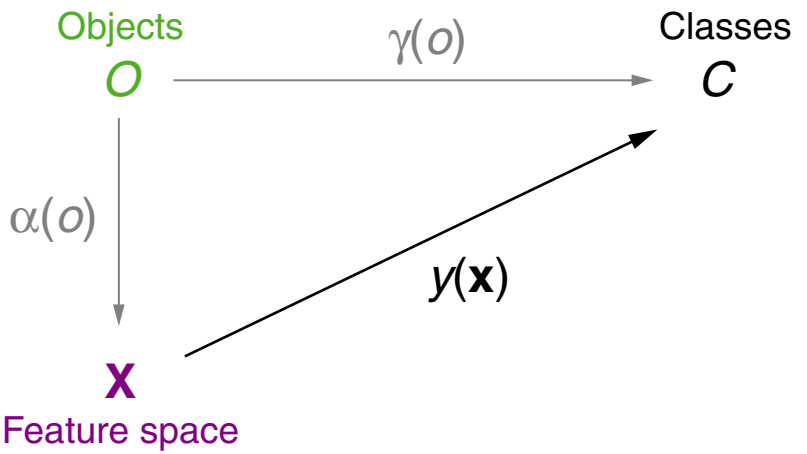


Semantics:

$\gamma(o)$	Ideal classifier (a human) for real-world objects.
$\alpha(o)$	Model formation function.

Specification of Learning Tasks

Real World → Model World (continued)



Semantics:

$\gamma(o)$	Ideal classifier (a human) for real-world objects.
$\alpha(o)$	Model formation function.
$y(\mathbf{x})$	Classifier (model function) to be learned.

Remarks:

- Feature vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots$, are abstractions of (or "proxies" for) real-world objects $o_1, o_2, \ldots$.
  While the real-world objects can be considered as unique, the feature vectors are typically not, as similar objects can be mapped onto the same feature vector. The set X thus actually is a multiset, comprised of feature vectors resulting from the analysis of a sequence of unique real-world objects.
- To be distinguished from the multiset X of feature vectors is the space $\mathbf{X}$, which is the Cartesian product of the domains of the different features (dimensions) of a feature vector $\mathbf{x}$.
  In the chess example, $\mathbf{X}$ is the $\mathbf{N}^6$, and, in the upcoming regression settings, $\mathbf{X}$ is the $\mathbf{R}^p$.
- $\mathbf{X}$ is called input space, feature space, feature domain, domain, or similar. The structure of the feature space (consider a pre-Hilbert space, a σ-algebra, or even no structure) is an important determinant of the learning problem.
- Heads-up: Actually, we will distinguish between the two terms input space and feature space (used synonymously in the item before) if an explicit feature transformation step is done such as basis expansion (regression), multilayer perceptron processing (neural networks), or non-linear kernelization (support vector machines).

Remarks (continued):

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

  A suitable "representation" $\mathbf{x}$ for an object o can also be algorithmically computed, with embedding approaches for example. Keyword: *representation learning*

- The value c in an example $(\mathbf{x}, c)$, $(\mathbf{x}, c) \in D$, is termed "target value", "ground truth", or "observation" (for $\mathbf{x}$ and the underlying classification problem). Observe that these terms are justified by the fact that $c := \gamma(o)$.

- Note that, in the chess example, $\gamma()$ defines the scores $y_i \in \mathbf{R}$ only for three types of positions (won, lost, drawn); $\gamma()$ is unknown for all positions o that fall under Point (4) of the recursive definition. This means that for most chess positions o, we *cannot provide the ground truth* $\gamma(o)$, say, we can neither give a statement whether o leads to a final position that is won or lost if both sides play optimally nor provide the next optimum move.

Specification of Learning Tasks

The LMS Algorithm for Fitting $y(\mathbf{x})$ [algorithms: LMS, BGD$_\sigma$, BGD, IGD, PT]

Algorithm:	LMS	Least Mean Squares.
Input:	D	Training examples $(\mathbf{x}, c)$ with $\mathbf{x} \in \mathbf{R}^p$ and target class c.
	η	Learning rate, a small positive constant.
Output:	$\mathbf{w}$	Weight vector from $\mathbf{R}^{p+1}$. (= hypothesis)

LMS(D, η)

1. *initialize_random_weights*$((w_0, w_1, \ldots, w_p))$, $t = 0$
2. **REPEAT**
3. $\quad t = t + 1$
4. $\quad (\mathbf{x}, c) =$ *random_select*(D)
5. $\quad y(\mathbf{x}) = w_0 + w_1 \cdot x_1 + \ldots + w_p \cdot x_p \overset{(\star)}{=} \mathbf{w}^T\mathbf{x}$
6. $\quad \delta = c - y(\mathbf{x})$
7. $\quad \Delta\mathbf{w} \overset{(\star)}{=} \eta \cdot \delta \cdot \mathbf{x}$ // $-\delta \cdot \mathbf{x}$ is the derivative of loss $l_2(c, y(\mathbf{x}))$ wrt. $\mathbf{w}$.
8. $\quad \mathbf{w} = \mathbf{w} + \Delta\mathbf{w}$
9. **UNTIL**(*convergence*$(D, y(), t)$)
10. *return*$((w_0, w_1, \ldots, w_p))$

Specification of Learning Tasks

The LMS Algorithm for Fitting $y(\mathbf{x})$ [algorithms: LMS, BGD$_\sigma$, BGD, IGD, PT]

Algorithm:	LMS	Least Mean Squares.
Input:	D	Training examples $(\mathbf{x}, c)$ with $\mathbf{x} \in \mathbf{R}^p$ and target class c.
	η	Learning rate, a small positive constant.
Output:	$\mathbf{w}$	Weight vector from $\mathbf{R}^{p+1}$. (= hypothesis)

LMS(D, η)

1. *initialize_random_weights*$((w_0, w_1, \ldots, w_p))$, $t = 0$
2. **REPEAT**
3. $t = t + 1$
4. $(\mathbf{x}, c) =$ *random_select*(D)
5. `Model function evaluation.`
6. `Calculation of residual.`
7. `Calculation of derivative of the loss.`
8. `Parameter vector update ≙ one gradient step down.`
9. **UNTIL**(*convergence*$(D, y(), t)$)
10. *return*$((w_0, w_1, \ldots, w_p))$

Remarks:

- A hypothesis is a proposed explanation for a phenomenon. [Wikipedia]

  Here, a hypothesis "explains" (= fits) the data D. Hence, a concrete model function $y()$, $\mathbf{y}()$, or, if the function type is clear from the context, its parameters $\mathbf{w}$ or $\boldsymbol{\theta}$ are called "hypothesis". The variable name h (similarly: h_1, h_2, h_i, h', etc.) may be used to refer to a specific instance of a model function or its parameters.

($\star$) We consider the feature vector $\mathbf{x}$ in its extended form when used as operand in a scalar product with the weight vector, $\mathbf{w}^T\mathbf{x}$, and consequently, when noted as argument of the model function, $y(\mathbf{x})$. I.e., $\mathbf{x} = (1, x_1, \ldots, x_p)^T \in \mathbf{R}^{p+1}$, and $x_0 = 1$.

- Line 7: The derivative of the pointwise squared loss $l_2(c, y(\mathbf{x}))$ is used to update the weight vector. See section Gradient Descent in Detail of part Neural Networks for a derivation of the update term.

- Line 9: The function *convergence*() can analyze the global loss quantified as sum of squared residuals (RSS), $\sum_{(\mathbf{x},c)\in D} (c - y(\mathbf{x}))^2$, or the norm of the loss gradient, $||\nabla \text{RSS}||$, and compare it to a small positive bound ε. Consider in this regard the vectors of observed and computed classes, $D|_c$ and $y(D|_\mathbf{x})$ respectively. In addition, the function may check via t an upper bound on the number of iterations.

Remarks (continued):

- The LMS algorithm approximates the solution of a least squares problem. It is a specialization of the SGD algorithm (stochastic gradient descent)—more specifically, an SGD that uses a squared loss (cost) function.

  The SGD algorithm, in turn, is a stochastic approximation of gradient descent optimization; it replaces the true gradient (calculated from the entire data set) by an estimate thereof (calculated from a randomly selected subset of the data).

- The LMS algorithm is hence very similar to the IGD algorithm (incremental gradient descent) equipped with a squared loss (cost) function. They differ in their selection of examples:

  [LMS] Line 4: $(\mathbf{x}, c) = \textit{random\_select}(D)$

  [IGD] Line 4: FOREACH $(\mathbf{x}, c) \in D$ DO

  The IGD algorithm approximates the global direction of steepest loss descent as used by the BGD algorithm (batch gradient descent), for which more rigorous statements on convergence are possible.

- The LMS algorithm was proposed in 1959 by Bernard Widrow and Ted Hoff.