Chapter ML:I

I. Introduction

- Examples of Learning Tasks
- Specification of Learning Tasks
- Elements of Machine Learning

© STEIN/LETTMANN 2005-2020

Examples of Learning Tasks

Car Shopping Guide

?

What criteria form the basis of a decision?

© STEIN/LETTMANN 2005-2020

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
...	

© STEIN/LETTMANN 2005-2020

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
...	

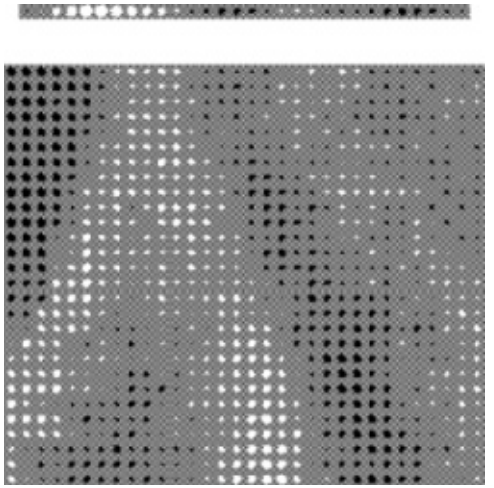
Learned rules:

```
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

...
```

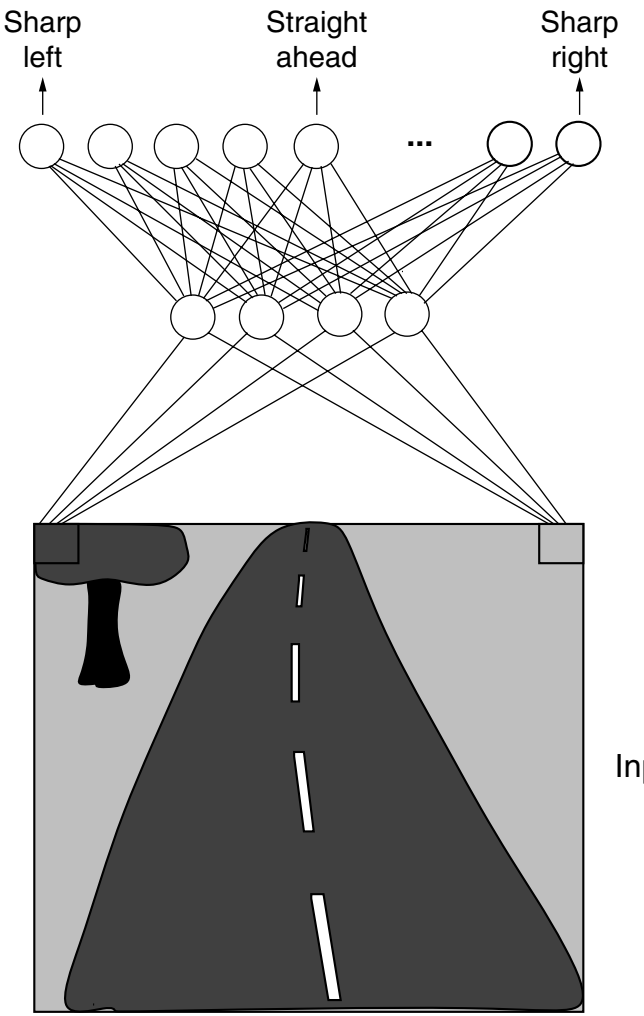

© STEIN/LETTMANN 2005-2020

Examples of Learning Tasks

Image Analysis [Mitchell 1997]



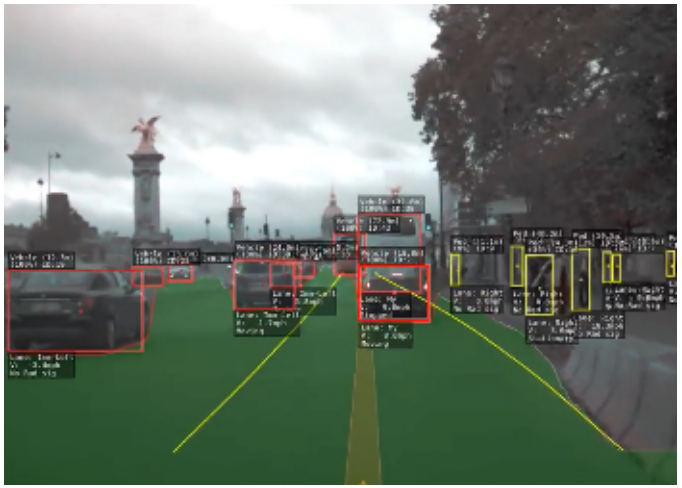
[1992]



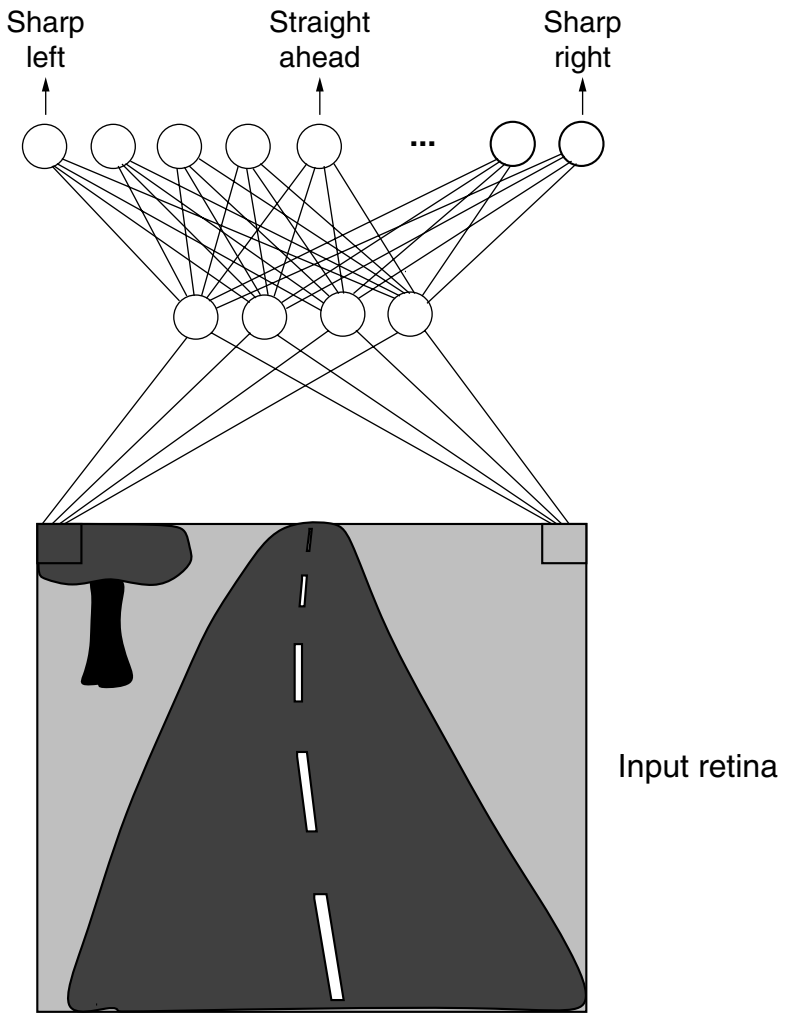
© STEIN/LETTMANN 2005-2020

Examples of Learning Tasks

Image Analysis [Mitchell 1997]



[2018]



© STEIN/LETTMANN 2005-2020

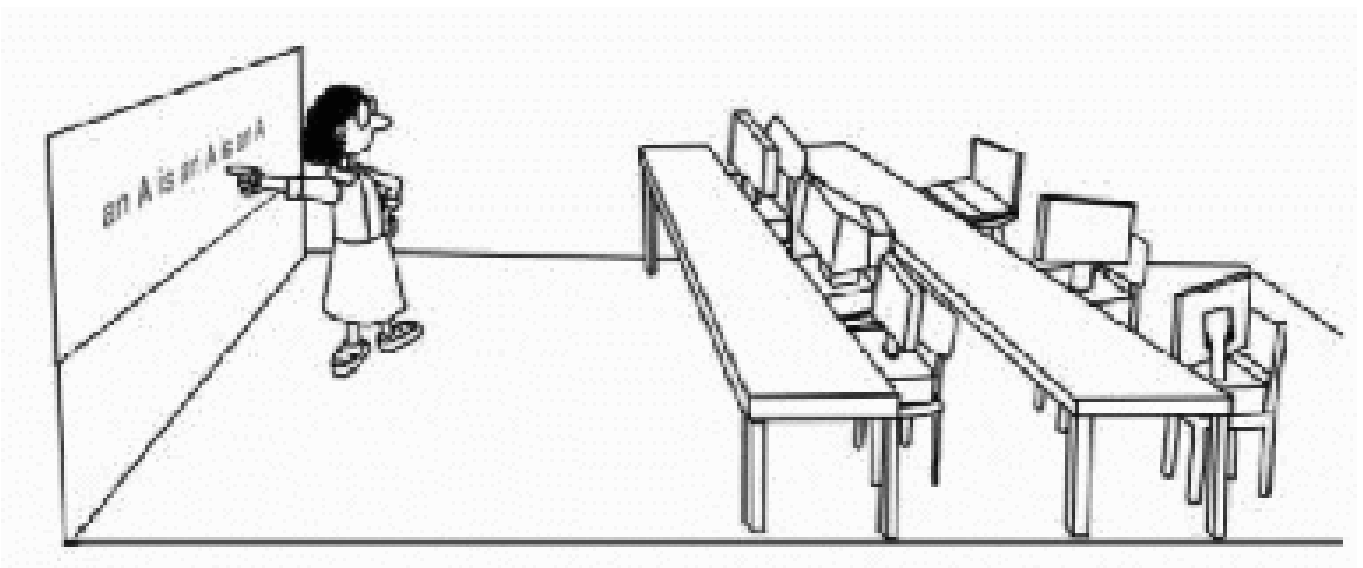
Specification of Learning Tasks

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.



©STEIN/LETTMANN 2005-2020

Remarks:

- Example: chess
  - task = playing chess
  - performance measure = number of games won during a world championship
  - experience = possibility to play 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 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]

© STEIN/LETTMANN 2005-2020

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.

© STEIN/LETTMANN 2005-2020

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.

© STEIN/LETTMANN 2005-2020

Specification of Learning Tasks

Example Chess: Kinds of Experience [Mitchell 1997]

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

©STEIN/LETTMANN 2005-2020

Specification of Learning Tasks

Example Chess: Kinds of Experience [Mitchell 1997]

1. Feedback
   - direct: for each board configuration 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 board for which the best move is unknown.

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

©STEIN/LETTMANN 2005-2020

Specification of Learning Tasks

Example Chess: Kinds of Experience [Mitchell 1997]

1. Feedback
   - direct: for each board configuration 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 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 may not be enough to become world class)

©STEIN/LETTMANN 2005-2020

Specification of Learning Tasks

Example Chess: Ideal Target Function γ [Mitchell 1997]

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

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

© STEIN/LETTMANN 2005-2020

Specification of Learning Tasks

Example Chess: Ideal Target Function γ [Mitchell 1997]

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

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

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

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

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: minimax strategy, α-β pruning. [Course on Search Algorithms, Stein 1998-2020]

© STEIN/LETTMANN 2005-2020

Specification of Learning Tasks

Example Chess: Real World $\rightarrow$ Model World

$$\gamma(o) \;\rightarrow\; y\big(\alpha(o)\big) \;\equiv\; y(\mathbf{x})$$ "model function"

©STEIN/LETTMANN 2005-2020

Specification of Learning Tasks

Example Chess: Real World $\rightarrow$ Model World

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

"model function"

"model formation function"

©STEIN/LETTMANN 2005-2020

Specification of Learning Tasks

Example Chess: Real World → Model World

$\gamma(o) \;\rightarrow\; y\big(\alpha(o)\big) \;\equiv\; y(\mathbf{x})$ "model function"

"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 on board o

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

$x_{5,\,6}$ = number of black / white pieces threatened on board o

D = $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, a set of board descriptions $\mathbf{x}_i$ with scores $y_i, y_i \in [-100; 100]$.

©STEIN/LETTMANN 2005-2020

Specification of Learning Tasks

Example Chess: Real World → Model World

$\gamma(o) \;\rightarrow\; y\big(\alpha(o)\big) \;\equiv\; y(\mathbf{x})$ "model function"

"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 on board o

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

$x_{5,\,6}$ = number of black / white pieces threatened on board o

D = $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, a set of board descriptions $\mathbf{x}_i$ with scores $y_i, y_i \in [-100; 100]$.

Other approaches to specify a model function y :

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

©STEIN/LETTMANN 2005-2020

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 consider 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, $y(\mathbf{x})$ is the abstracted and formalized counterpart of $\gamma(o)$.
- y is called *model function* or *model*. The choice and computation of a suited model function is a central aspect in the field of machine learning.
- α is called *model formation function*. The development of a suited model formation function is often not treated as part of machine learning but "outsourced" to respective domain experts. However, tackling most advanced learning tasks such as autonomous driving, automated debating, or playing chess requires a tight cooperation between the developers of α and y.
- The key difference between an ideal target function γ and a model function y lies in the complexity 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 $\mathbf{x}$, $\mathbf{x} = \alpha(o)$.

© STEIN/LETTMANN 2005-2020

Remarks (continued) :

- For automated chess playing a real-valued assessment function is needed; such kind of tasks 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 via 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 may be a sensible criterion; for classification problems the number of misclassified examples may be more relevant. Keywords: *regression loss* versus *classification loss*
- For classification problems, the ideal target function γ is also called ideal *classifier*; analogously, the model function y is called classifier.
- Decision problems are classification problems with two classes.
- The halting problem for Turing machines is an undecidable classification problem.

© STEIN/LETTMANN 2005-2020

Specification of Learning Tasks [model world]

Real World $\rightarrow$ Model World

Characterization 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 for O. (γ is a human expert)

Classification task:

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

©STEIN/LETTMANN 2005-2020

Specification of Learning Tasks [model world]

Real World → Model World

Characterization of the real world:

- O is a set of objects. (example: emails)
- C is a set of classes. (example: spam versus ham)
- $\gamma : O \to C$ is the ideal classifier for O. (γ is 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 $\mathbf{x} \in X$, where $\mathbf{x} = \alpha(o)$.
3. Construct examples as $(\mathbf{x}, c(\mathbf{x}))$, where $\mathbf{x} = \alpha(o)$ and $c(\mathbf{x}) \equiv \gamma(o)$.

©STEIN/LETTMANN 2005-2020

Specification of Learning Tasks [real world]

Real World $\rightarrow$ Model World (continued)

Characterization of the model world:

- X is a set of feature vectors, called feature space. (example: word frequencies)
- C is a set of classes. (as before: spam versus ham)
- $c : X \rightarrow C$ is the ideal classifier for X. (c is unknown)
- $D = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, c(\mathbf{x}_n))\} \subseteq X \times C$ is a set of examples.

Todo:

- Approximate c, which is implicitly given via D, by a model function y.

©STEIN/LETTMANN 2005-2020

Specification of Learning Tasks [real world]

Real World $\rightarrow$ Model World (continued)

Characterization of the model world:

- X is a set of feature vectors, called feature space. (example: word frequencies)
- C is a set of classes. (as before: spam versus ham)
- $c : X \rightarrow C$ is the ideal classifier for X. (c is unknown)
- $D = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, c(\mathbf{x}_n))\} \subseteq X \times C$ is a set of examples.

Todo:

- Approximate c, which is implicitly given via D, by a model function y.

Machine learning:

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

©STEIN/LETTMANN 2005-2020

Specification of Learning Tasks

Real World $\rightarrow$ Model World (continued)

$$\underset{O}{\text{Objects}} \xrightarrow{\gamma} \underset{C}{\text{Classes}}$$

Semantics:

γ Ideal classifier (a human) for real-world objects.

©STEIN/LETTMANN 2005-2020

Specification of Learning Tasks

Real World → Model World (continued)

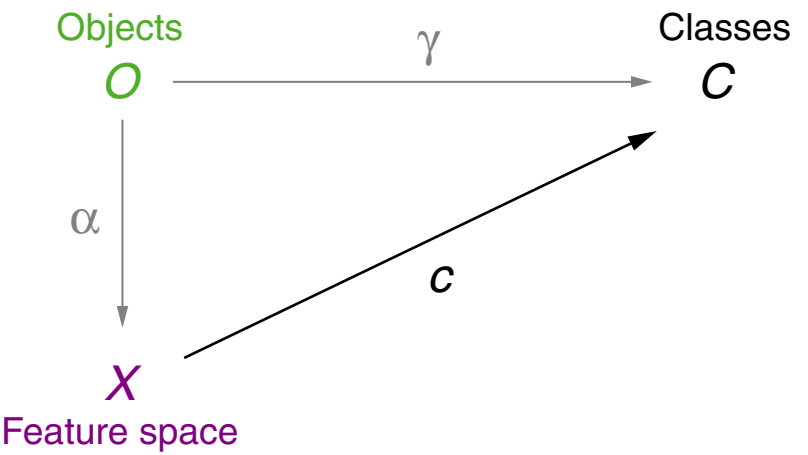
Semantics:

- γ Ideal classifier (a human) for real-world objects.
- α Model formation function.

©STEIN/LETTMANN 2005-2020

Specification of Learning Tasks

Real World → Model World (continued)



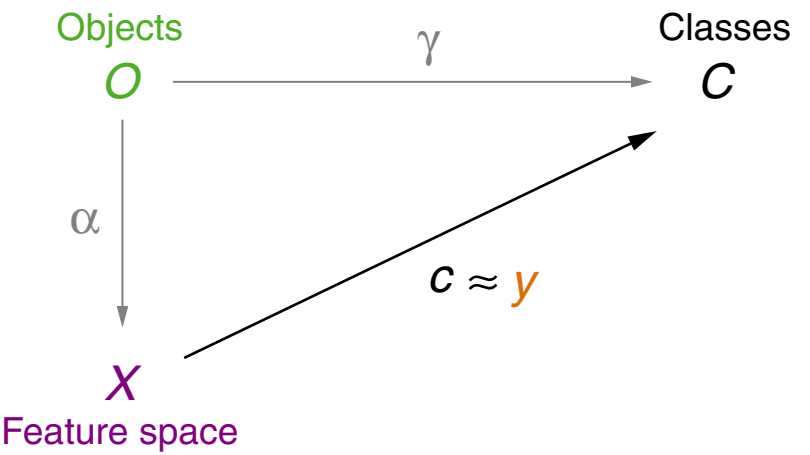
Semantics:

- γ Ideal classifier (a human) for real-world objects.
- α Model formation function.
- c Unknown ideal classifier for vectors from the feature space.

©STEIN/LETTMANN 2005-2020

Specification of Learning Tasks

Real World → Model World (continued)



Semantics:

γ	Ideal classifier (a human) for real-world objects.
α	Model formation function.
c	Unknown ideal classifier for vectors from the feature space.
y	Classifier (model function) to be learned.
$c \approx y$	c is approximated by y, based on a set of examples D.

© STEIN/LETTMANN 2005-2020

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 $\mathbf{x} = \alpha(o)$, c must be considered as *ideal* classifier, since $c(\mathbf{x})$ is defined as $\gamma(o)$, the true real-world class. I.e., c and γ have different domains each, but they return the same images.
- The function c is only implicitly given, in the form of the example set D. The representation of c as a closed function is unknown, it is approximated by y.
- $c(\mathbf{x})$ is often termed "ground truth" (for $\mathbf{x}$ and the underlying classification problem). Observe that this term is justified by the fact that $c(\mathbf{x}) \equiv \gamma(o)$.
- Note that in the chess example the scores y_i are not prescribed by $\gamma(o)$, since for $\gamma(o)$ only a recursive definition is given where $\gamma(o)$ *is unknown* for all boards o that fall under Point (4). I.e., for most chess boards o we *cannot provide the ground truth* $\gamma(o)$, say, we can neither give a statement whether o leads to a final board state that is won or lost if both sides play optimally nor provide the next optimum move.

© STEIN/LETTMANN 2005-2020

Specification of Learning Tasks

The LMS Algorithm for Fitting y [*IGD* Algorithm]

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

LMS(D, η)

1. *initialize_random_weights*$((w_0, w_1, \ldots, w_p))$
2. **REPEAT**
3. $\quad (\mathbf{x}, c(\mathbf{x})) =$ *random_select*(D)
4. $\quad y(\mathbf{x}) = w_0 + w_1 \cdot x_1 + \ldots + w_p \cdot x_p = \mathbf{w}^T\mathbf{x}$ // $\forall_{\mathbf{x} \in D} : \mathbf{x}|_{x_0} \equiv 1$
5. $\quad$ *error* $= c(\mathbf{x}) - y(\mathbf{x})$
6. $\quad \Delta\mathbf{w} = \eta \cdot$ *error* $\cdot \mathbf{x}$
7. $\quad \mathbf{w} = \mathbf{w} + \Delta\mathbf{w}$
8. **UNTIL**(*convergence*$(D, y(D))$)
9. *return*$((w_0, w_1, \ldots, w_p))$

© STEIN/LETTMANN 2005-2020

Remarks:

- The LMS weight adaptation corresponds to the incremental gradient descend [*IGD*] algorithm, and approximates the global direction of steepest error descent as used by the batch gradient descent [*BGD*] algorithm, for which more rigorous statements on convergence are possible.
- The *convergence* function may compute the global error quantified as the sum of the squared residuals, $\sum_{(\mathbf{x},c(\mathbf{x}))\in D} \left(c(\mathbf{x}) - y(\mathbf{x})\right)^2$, or employ an upper bound on the number of iterations.

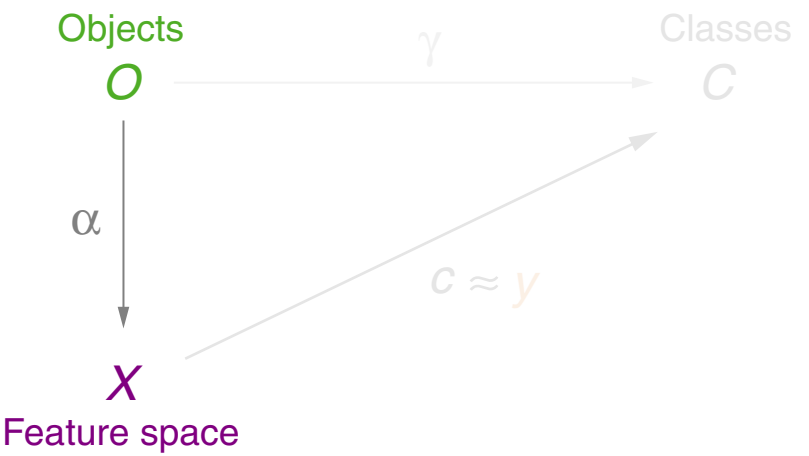
© STEIN/LETTMANN 2005-2020

Elements of Machine Learning

© STEIN/LETTMANN 2005-2020

Elements of Machine Learning

I. Model Formation: Real World → Model World



Related questions:

- From what kind of experience should be learned?
- Which level of fidelity is sufficient to solve a certain task?

© STEIN/LETTMANN 2005-2020

Elements of Machine Learning

II. Design of Supervised Learning Algorithms

Optimization approach

Performance function
Loss function [+ Regularization]

Model function $\rightsquigarrow$ Hypothesis space

Task

Data

© STEIN/LETTMANN 2005-2020

Elements of Machine Learning

II. Design of Supervised Learning Algorithms: LMS

Optimization approach

Performance function
Loss function [+ Regularization]

Model function $\rightsquigarrow$ Hypothesis space

Task

Data

Classification

$$D = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, c(\mathbf{x}_n))\} \subseteq X \times C$$

©STEIN/LETTMANN 2005-2020

Elements of Machine Learning

II. Design of Supervised Learning Algorithms: LMS (continued)

Optimization approach

Performance function
Loss function [+ Regularization]

Model function $\rightsquigarrow$ Hypothesis space

- Hypothesis space: Set of linear functions
- Linear model: $y(\mathbf{x}) = w_0 + \sum_{i=1}^{p} w_i x_i$

Task

Classification

Data

$$D = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, c(\mathbf{x}_n))\} \subseteq X \times C$$

©STEIN/LETTMANN 2005-2020

Elements of Machine Learning

II. Design of Supervised Learning Algorithms: LMS (continued)

Optimization approach

Performance function

Loss function [+ Regularization]

Model function $\rightsquigarrow$ Hypothesis space

Task

Data

- Performance: Minimize MSE
- Regularization: None
- Loss: Sum of squared residuals

- Hypothesis space: Set of linear functions
- Linear model: $y(\mathbf{x}) = w_0 + \sum_{i=1}^{p} w_i x_i$

Classification

$$D = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, c(\mathbf{x}_n))\} \subseteq X \times C$$

©STEIN/LETTMANN 2005-2020

Elements of Machine Learning

II. Design of Supervised Learning Algorithms: LMS (continued)

Optimization approach	Incremental/stochastic gradient descend
Performance function Loss function [+ Regularization]	❑ Performance: Minimize MSE ❑ Regularization: None ❑ Loss: Sum of squared residuals
Model function $\rightsquigarrow$ Hypothesis space	❑ Hypothesis space: Set of linear functions ❑ Linear model: $y(\mathbf{x}) = w_0 + \sum_{i=1}^{p} w_i x_i$
Task	Classification
Data	$D = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, c(\mathbf{x}_n))\} \subseteq X \times C$

©STEIN/LETTMANN 2005-2020

Related questions:

- What are useful classes of model functions?
- What are methods to fit (= learn) model functions?
- What are measures to assess the goodness of fit?

- How does (label) noise affect the learning process?
- How does the example number affect the learning process?
- How to deal with extreme class imbalance?

© STEIN/LETTMANN 2005-2020

The feature space is an inner product space.

- An inner product space (also called pre-Hilbert space) is a vector space with an additional structure called “inner product”.
- Example: Euclidean vector space equipped with the dot product.
- Enables algorithms such as gradient descent and support vector machines.

© STEIN/LETTMANN 2005-2020

Elements of Machine Learning

III. Structure of Feature Spaces (continued)

The feature space is an inner product space.

- An inner product space (also called pre-Hilbert space) is a vector space with an additional structure called “inner product”.
- Example: Euclidean vector space equipped with the dot product.
- Enables algorithms such as gradient descent and support vector machines.

The feature space is a σ-algebra.

- A σ-algebra on a set X is a collection of subsets of X that includes X itself, is closed under complement, and is closed under countable unions.
- Enables probability spaces and statistical learning, such as naive Bayes.

© STEIN/LETTMANN 2005-2020

Elements of Machine Learning

III. Structure of Feature Spaces (continued)

The feature space is an inner product space.

- An inner product space (also called pre-Hilbert space) is a vector space with an additional structure called “inner product”.
- Example: Euclidean vector space equipped with the dot product.
- Enables algorithms such as gradient descent and support vector machines.

The feature space is a σ-algebra.

- A σ-algebra on a set X is a collection of subsets of X that includes X itself, is closed under complement, and is closed under countable unions.
- Enables probability spaces and statistical learning, such as naive Bayes.

The feature space is a finite set of vectors with nominal dimensions.

- Requires concept learning via set splitting as done by decision trees.

© STEIN/LETTMANN 2005-2020

Remarks:

- The aforementioned examples for feature spaces are not meant to be complete but shall illustrate the range of different structures underlying the example sets we want to learn from.
- The structure of a feature space constrains the applicable learning algorithm. Usually, this structure is inherently determined by the application domain and cannot be chosen.

© STEIN/LETTMANN 2005-2020

Elements of Machine Learning

IV. Discriminative versus Generative Approaches

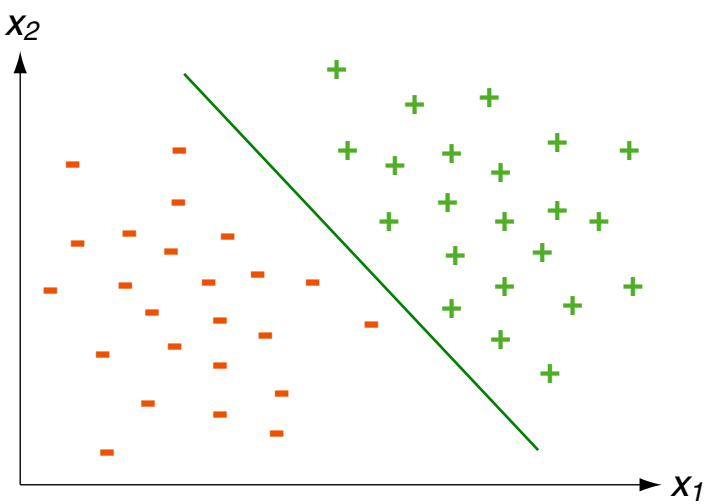
- Discriminative approaches learn a boundary between classes.
- Generative approaches exploit the distributions underlying the classes.

© STEIN/LETTMANN 2005-2020

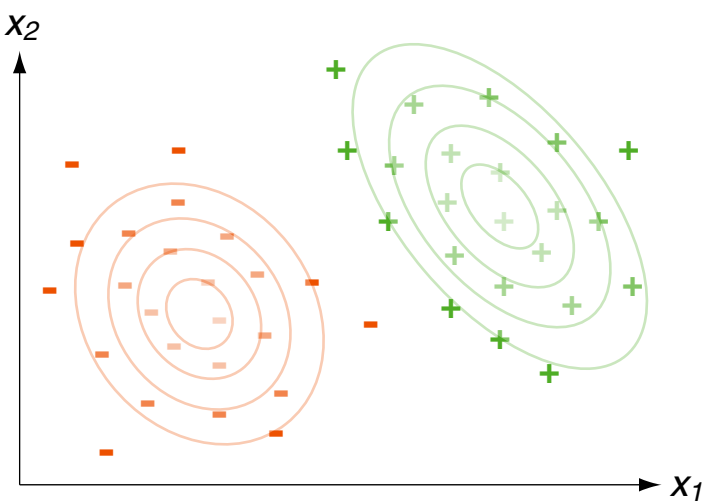
Elements of Machine Learning

IV. Discriminative versus Generative Approaches (continued)

- Discriminative approaches learn a boundary between classes.
- Generative approaches exploit the distributions underlying the classes.



discriminative
$\rightsquigarrow$ classification rule



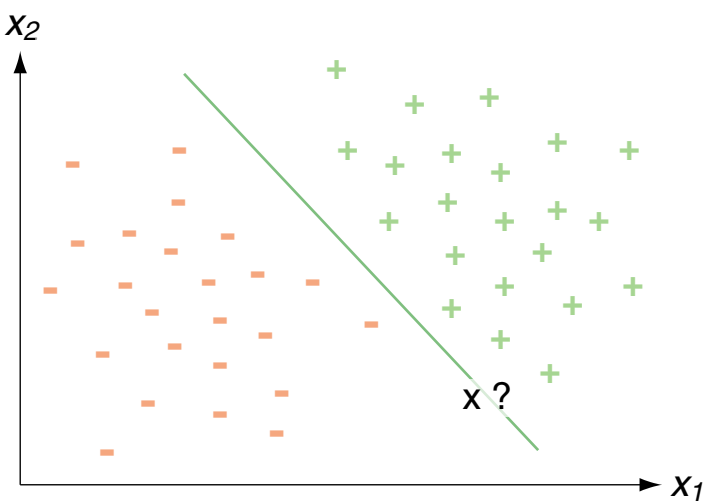
generative
$\rightsquigarrow$ class membership probability

© STEIN/LETTMANN 2005-2020

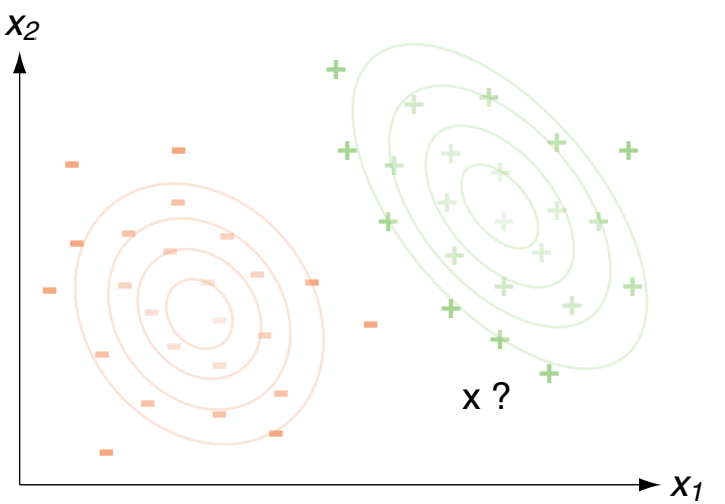
Elements of Machine Learning

IV. Discriminative versus Generative Approaches (continued)

- Discriminative approaches learn a boundary between classes.
- Generative approaches exploit the distributions underlying the classes.



discriminative
$\rightsquigarrow$ classification rule



generative
$\rightsquigarrow$ class membership probability

© STEIN/LETTMANN 2005-2020

Remarks:

- When classifying a new example (1) discriminative approaches apply a decision rule that was learned via minimizing the misclassification rate given training examples D, while (2) generative approaches maximize the probability of the combined event $P(\mathbf{x}, y)$, or, similarly, the a-posteriori probability $P(y \mid \mathbf{x})$, $y \in \{\ominus, \oplus\}$.
- The LMS algorithm computes "only" a decision boundary, i.e., it constructs a classifier according to the discriminative approach. A Bayes classifier is an example for a classifier constructed according to the generative approach.
- Yoav Freund provides an excellent video illustrating the pros and cons of discriminative and generative approaches respectively. [YouTube]

© STEIN/LETTMANN 2005-2020

Elements of Machine Learning

V. Frequentism versus Subjectivism

Frequentism:

- There is a (hidden) mechanism that generates D.
- To model this mechanism you may consider
  - a family of distributions,
  - a model function,
  - a combination of both,

  characterized by θ. The possible values for θ form the hypothesis space H.
- Select a most probable hypothesis $h_{\text{ML}} \in H$ by estimating θ using a sample $D' \subset D$. h_{ML} is called maximum likelihood hypothesis.

© STEIN/LETTMANN 2005-2020

Elements of Machine Learning

V. Frequentism versus Subjectivism (continued)

Frequentism:

- There is a (hidden) mechanism that generates D.
- To model this mechanism you may consider
  - a family of distributions,
  - a model function,
  - a combination of both,

  characterized by θ. The possible values for θ form the hypothesis space H.
- Select a most probable hypothesis $h_{\text{ML}} \in H$ by estimating θ using a sample $D' \subset D$. h_{ML} is called maximum likelihood hypothesis.

© STEIN/LETTMANN 2005-2020

Elements of Machine Learning

V. Frequentism versus Subjectivism (continued)

Frequentism:

- There is a (hidden) mechanism that generates D.
- To model this mechanism you may consider
  - a family of distributions,
  - a model function,
  - a combination of both,

  characterized by θ. The possible values for θ form the hypothesis space H.
- Select a most probable hypothesis $h_{\text{ML}} \in H$ by estimating θ using a sample $D' \subset D$. h_{ML} is called maximum likelihood hypothesis.

©STEIN/LETTMANN 2005-2020

Elements of Machine Learning

V. Frequentism versus Subjectivism (continued)

Frequentism:

- There is a (hidden) mechanism that generates D.
- To model this mechanism you may consider
  - a family of distributions,
  - a model function,
  - a combination of both,

  characterized by θ. The possible values for θ form the hypothesis space H.
- Select a most probable hypothesis $h_{\text{ML}} \in H$ by estimating θ using a sample $D' \subset D$. h_{ML} is called maximum likelihood hypothesis.

$$\theta \rightsquigarrow D', \qquad h_{\text{ML}} = \operatorname*{argmax}_{h \in H} P(D' \mid h)$$

© STEIN/LETTMANN 2005-2020

Remarks:

- θ is a parameter or a parameter vector that is considered as fixed (in particular: not as a random variable), but unknown.
- In the experiment of flipping a coin, one may suppose a Laplace experiment and consider the binomial distribution, $B(n, p)$.
- $P(D' \mid h)$ is the probability for observing D' under h. I.e., it is the probability for observing D' if the hidden mechanism that generates D' behaves according to the considered model whose parameter θ is set to h.

© STEIN/LETTMANN 2005-2020

Elements of Machine Learning

V. Frequentism versus Subjectivism (continued)

Subjectivism:

- Consider a model for the mechanism that has generated D.
- There are different beliefs about the parameter (vector) θ that characterizes the model. The possible values for θ form the hypothesis space H.
- Select a most probable hypothesis $h_{\text{MAP}} \in H$ by weighting the ML estimates under D with the priors. h_{MAP} is called maximum a-posteriori hypothesis.

Belief/Prior 1: $P(\underbrace{p = 0.5}_{\theta_1})$ = 0.95

Belief/Prior 2: $P(\underbrace{p = 0.75}_{\theta_2})$ = 0.50

©STEIN/LETTMANN 2005-2020

Elements of Machine Learning

V. Frequentism versus Subjectivism (continued)

Subjectivism:

- Consider a model for the mechanism that has generated D.
- There are different beliefs about the parameter (vector) θ that characterizes the model. The possible values for θ form the hypothesis space H.
- Select a most probable hypothesis $h_{\text{MAP}} \in H$ by weighting the ML estimates under D with the priors. h_{MAP} is called maximum a-posteriori hypothesis.

Belief/Prior 1: $P(\underbrace{p = 0.5}_{\theta_1})$ = 0.95

Belief/Prior 2: $P(\underbrace{p = 0.75}_{\theta_2})$ = 0.50

©STEIN/LETTMANN 2005-2020

Elements of Machine Learning

V. Frequentism versus Subjectivism (continued)

Subjectivism:

- Consider a model for the mechanism that has generated D.
- There are different beliefs about the parameter (vector) θ that characterizes the model. The possible values for θ form the hypothesis space H.
- Select a most probable hypothesis $h_{\text{MAP}} \in H$ by weighting the ML estimates under D with the priors. h_{MAP} is called maximum a-posteriori hypothesis.

Belief/Prior 1: $P(\underbrace{p = 0.5}_{\theta_1})$ = 0.95

Belief/Prior 2: $P(\underbrace{p = 0.75}_{\theta_2})$ = 0.50

©STEIN/LETTMANN 2005-2020

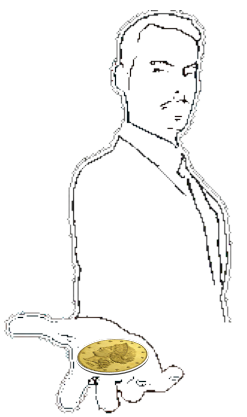
Elements of Machine Learning

V. Frequentism versus Subjectivism (continued)

Subjectivism:

- Consider a model for the mechanism that has generated D.
- There are different beliefs about the parameter (vector) θ that characterizes the model. The possible values for θ form the hypothesis space H.
- Select a most probable hypothesis $h_{\text{MAP}} \in H$ by weighting the ML estimates under D with the priors. h_{MAP} is called maximum a-posteriori hypothesis.

Belief/Prior 1: $P(\underbrace{p = 0.5}_{\theta_1})$ = 0.95



Belief/Prior 2: $P(\underbrace{p = 0.75}_{\theta_2})$ = 0.50

$$\left.\begin{aligned} \theta_1 + D &\rightarrow P(D \mid \theta_1) \\ \theta_2 + D &\rightarrow P(D \mid \theta_2) \end{aligned}\right\}$$

©STEIN/LETTMANN 2005-2020

Elements of Machine Learning

V. Frequentism versus Subjectivism (continued)

Subjectivism:

- Consider a model for the mechanism that has generated D.
- There are different beliefs about the parameter (vector) θ that characterizes the model. The possible values for θ form the hypothesis space H.
- Select a most probable hypothesis $h_{\text{MAP}} \in H$ by weighting the ML estimates under D with the priors. h_{MAP} is called maximum a-posteriori hypothesis.

Belief/Prior 1: $P(\underbrace{p = 0.5}_{\theta_1})$ = 0.95

Belief/Prior 2: $P(\underbrace{p = 0.75}_{\theta_2})$ = 0.50

$$\left.\begin{array}{l} \theta_1 + D \;\rightarrow\; P(D \mid \theta_1) \\ \theta_2 + D \;\rightarrow\; P(D \mid \theta_2) \end{array}\right\} \quad h_{\text{MAP}} = \underset{h \in \{\theta_1, \theta_2\}}{\text{argmax}}\, P(h \mid D) = \underset{h \in \{\theta_1, \theta_2\}}{\text{argmax}} \frac{P(D \mid h) \cdot P(h)}{P(D)}$$

© STEIN/LETTMANN 2005-2020

Remarks:

- θ is considered as random variable. There is prior knowledge about the distribution of θ.
- p is a parameter of the binomial distribution and denotes the success probability for each trial.
  - Belief 1: With a probability of 0.95 the coin is fair (both sides are equally likely).
  - Belief 2: With a probability of 0.5 the odds of prefering a particular side is 3:1.
- The subjectivistic approach is also called Bayesian interpretation of probability.
- The Bayesian interpretation of probability enables by design the integration of prior knowledge, background knowledge, and human expertise.
- Food for tought: Discuss the use of frequentist and subjectivist approaches to decision making if you had to develop an AI that plays poker.

© STEIN/LETTMANN 2005-2020

Chapter ML:II

II. Machine Learning Basics

- Linear Regression
- Concept Learning: Search in Hypothesis Space
- Concept Learning: Version Space
- Evaluating Effectiveness

©STEIN 2021

Linear Regression

Regression versus Classification

- X is a set of p-dimensional feature vectors:

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
...	

©STEIN 2021

Linear Regression

Regression versus Classification

- X is a set of p-dimensional feature vectors:

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
...	

Regression setting:

- $\mathbf{R}$ is the range of the regression function.
- y_i is the ground truth of the credit line value, $\mathbf{x}_i \in X$.
- $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \subseteq X \times \mathbf{R}$ is a set of examples.

©STEIN 2021

Linear Regression

Regression versus Classification

- X is a set of p-dimensional feature vectors:

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
...	

Regression setting:

- $\mathbf{R}$ is the range of the regression function.
- y_i is the ground truth of the credit line value, $\mathbf{x}_i \in X$.
- $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \subseteq X \times \mathbf{R}$ is a set of examples.

Classification setting:

- $C = \{-1, 1\}$ is a set of two classes. Similarly: $\{0, 1\}$, $\{\ominus, \oplus\}$, $\{\text{no}, \text{yes}\}$, etc.
- $c(\mathbf{x}_i)$ is the ground truth of the creditworthiness class, $\mathbf{x}_i \in X$.
- $D = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, c(\mathbf{x}_n))\} \subseteq X \times C$ is a set of examples.

©STEIN 2021

Linear Regression

The Linear Regression Model

- Given $\mathbf{x}$, predict a real-valued output under a linear model function:

$$y(\mathbf{x}) = w_0 + \sum_{j=1}^{p} w_j \cdot x_j$$

- Vector notation with $x_0 = 1$ and $\mathbf{w} = (w_0, w_1, \ldots, w_p)^T$:

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

- Given $\mathbf{x}_1, \ldots, \mathbf{x}_n$, assess goodness of fit as residual sum of squares:

$$\text{RSS}(\mathbf{w}) \;=\; \sum_{i=1}^{n} (y_i - y(\mathbf{x}_i))^2 \;=\; \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 \tag{1}$$

©STEIN 2021

Linear Regression

The Linear Regression Model

- Given $\mathbf{x}$, predict a real-valued output under a linear model function:

$$y(\mathbf{x}) = w_0 + \sum_{j=1}^{p} w_j \cdot x_j$$

- Vector notation with $x_0 = 1$ and $\mathbf{w} = (w_0, w_1, \ldots, w_p)^T$:

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

- Given $\mathbf{x}_1, \ldots, \mathbf{x}_n$, assess goodness of fit as residual sum of squares:

$$\text{RSS}(\mathbf{w}) \;=\; \sum_{i=1}^{n} (y_i - y(\mathbf{x}_i))^2 \;=\; \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 \tag{1}$$

- Estimate $\mathbf{w}$ by minimizing the residual sum of squares:

$$\hat{\mathbf{w}} \;=\; \underset{\mathbf{w} \in \mathbf{R}^{p+1}}{\text{argmin}} \; \text{RSS}(\mathbf{w}) \tag{2}$$

©STEIN 2021

Remarks:

- A *residual* is the difference between an observed value y_i and the estimated value $y(\mathbf{x}_i)$ of the model function.
- The residual sum of squares, RSS, is the sum of squares of the residuals. It is also known as the sum of squared residuals, SSR, or the sum of squared errors of estimates, SSE.
- The RSS term quantifies the regression error—or similarly, the goodness of fit—in the form of a single value.
- RSS provides several numerical and theoretical advantages, but it is not the only possibility to assess the goodness of fit (= error) between observed values and the model function. Alternative approaches for quantifying the error include absolute residual values or a polynomial in the residual values.
- The error computation is also called loss computation, cost computation, or generally, performance computation. Similarly, for the right-hand side of Equation (1) the following names are used: error function, loss function, cost function, or generally, performance term. Measures that quantify this kind of performance are called *effectiveness* measures. This term must not be confused with *efficiency* measures, which quantify the computational effort or runtime performance of a method.

©STEIN 2021

Remarks (continued) :

- From a statistical viewpoint, both $\mathbf{x}$ and y are *random variables* (vectorial and scalar respectively). Each feature vector, $\mathbf{x}_i$, and outcome, y_i, is the result of a random experiment and hence is governed by a—usually unknown—probability distribution.

- The distributions of the random variables y_i and $(y_i - y(\mathbf{x}_i))$ are identical.

- Equation (2): Estimating $\mathbf{w}$ by RSS minimization is based on the following assumptions:

  1. The random variables y_i are statistically independent. Actually, the conditional independence of the y_i under $\mathbf{x}_i$ is sufficient.

  2. The means $E(y_i)$ lie on a straight line, known as the true (population) regression line: $E(y_i) = \mathbf{w}^{*T}\mathbf{x}_i$. I.e., the relation between the observed $(\mathbf{x}, y)$ can be completely explained by a linear model function.

  3. The probability distributions $P(y_i \mid \mathbf{x}_i)$ have the same variance.

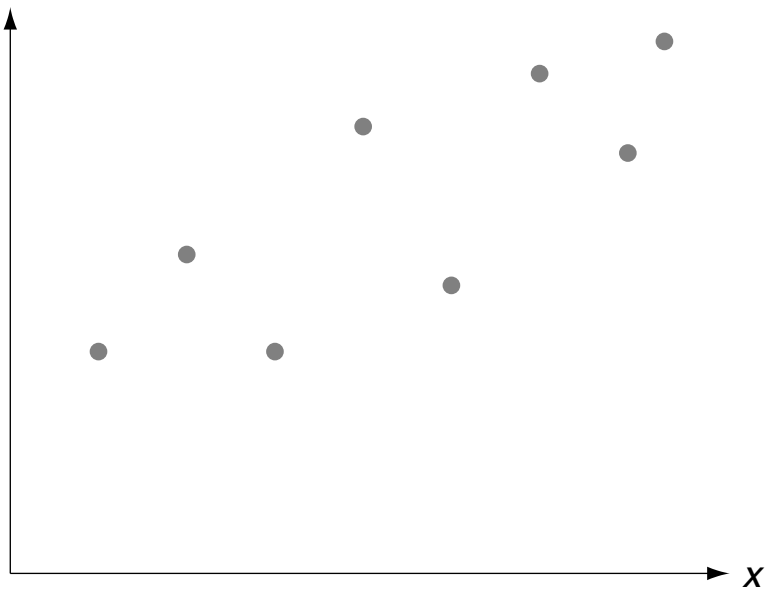
  The three assumptions are called the *weak set* (of assumptions). Along with a fourth assumption about the distribution shape of y_i they become the *strong set* of assumptions.

- To avoid cluttered notation, we won't use additional symbols to distinguish random variables from ordinary variables. I.e., whether $\mathbf{x}, x, y$, etc. denote a (vectorial or scalar) random variable will become clear from the context.

©STEIN 2021

Linear Regression

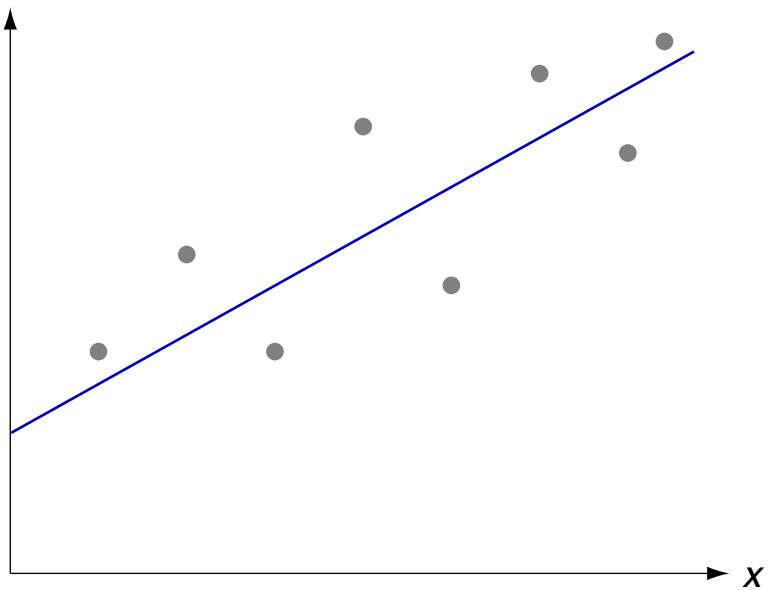
One-Dimensional Feature Space



©STEIN 2021

Linear Regression

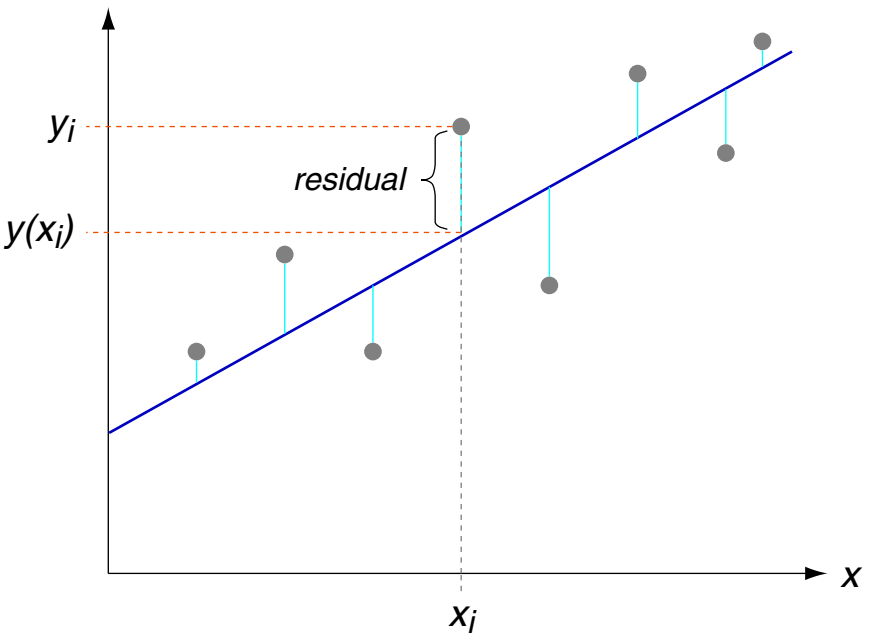
One-Dimensional Feature Space (continued)



©STEIN 2021

Linear Regression

One-Dimensional Feature Space (continued)

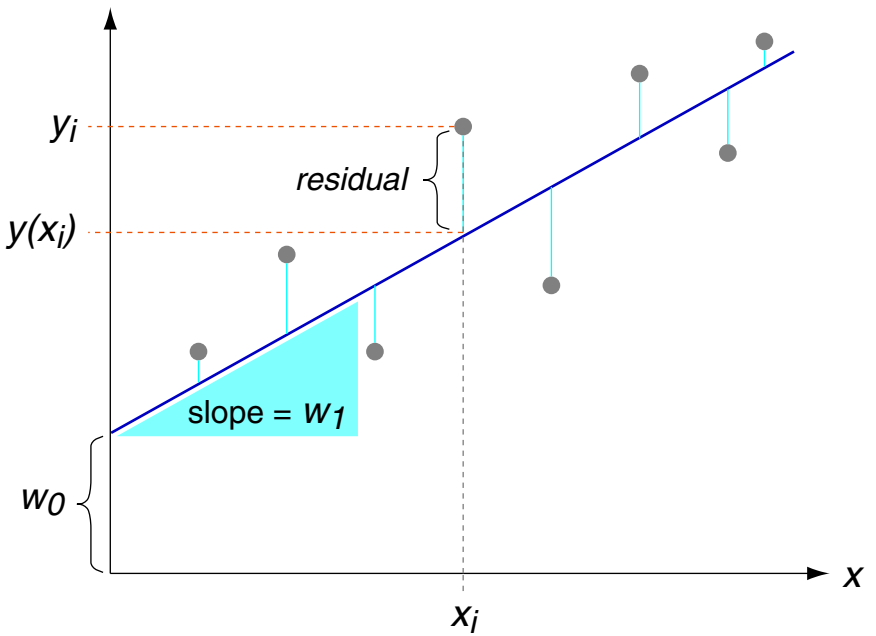


$$\text{RSS} = \sum_{i=1}^{n} (y_i - y(x_i))^2$$

©STEIN 2021

Linear Regression

One-Dimensional Feature Space (continued)



$$y(x) = w_0 + w_1 \cdot x, \qquad \text{RSS}(w_0, w_1) = \sum_{i=1}^{n} (y_i - w_0 - w_1 \cdot x_i)^2$$

©STEIN 2021

Linear Regression

One-Dimensional Feature Space (continued) [higher-dimensional]

Minimize RSS(w_0, w_1) by a direct method:

1. $$\frac{\partial}{\partial w_0} \sum_{i=1}^{n} (y_i - w_0 - w_1 \cdot x_i)^2 \;=\; 0$$

$$\rightsquigarrow \;\ldots\; \rightsquigarrow \quad w_0 \;=\; \frac{1}{n} \sum_{i=1}^{n} y_i \;-\; \frac{w_1}{n} \sum_{i=1}^{n} x_i \;=\; \bar{y} - w_1 \cdot \bar{x}$$

©STEIN 2021

Linear Regression

One-Dimensional Feature Space (continued) [higher-dimensional]

Minimize RSS(w_0, w_1) by a direct method:

1. $$\frac{\partial}{\partial w_0} \sum_{i=1}^{n} (y_i - w_0 - w_1 \cdot x_i)^2 \;=\; 0$$

$$\rightsquigarrow \;\ldots\; \rightsquigarrow \qquad w_0 \;=\; \frac{1}{n} \sum_{i=1}^{n} y_i \;-\; \frac{w_1}{n} \sum_{i=1}^{n} x_i \;=\; \bar{y} - w_1 \cdot \bar{x}$$

2. $$\frac{\partial}{\partial w_1} \sum_{i=1}^{n} (y_i - w_0 - w_1 \cdot x_i)^2 \;=\; 0$$

$$\rightsquigarrow \;\ldots\; \rightsquigarrow \qquad w_1 = \frac{\sum_{i=1}^{n} (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

©STEIN 2021

Linear Regression

One-Dimensional Feature Space (continued) [higher-dimensional]

Minimize RSS(w_0, w_1) by a direct method:

1. $$\frac{\partial}{\partial w_0} \sum_{i=1}^{n} (y_i - w_0 - w_1 \cdot x_i)^2 \;=\; 0$$

$$\rightsquigarrow \;\ldots\; \rightsquigarrow \quad \hat{w}_0 \;=\; \frac{1}{n} \sum_{i=1}^{n} y_i \;-\; \frac{w_1}{n} \sum_{i=1}^{n} x_i \;=\; \bar{y} - \hat{w}_1 \cdot \bar{x}$$

2. $$\frac{\partial}{\partial w_1} \sum_{i=1}^{n} (y_i - w_0 - w_1 \cdot x_i)^2 \;=\; 0$$

$$\rightsquigarrow \;\ldots\; \rightsquigarrow \quad \hat{w}_1 \equiv w_1 = \frac{\sum_{i=1}^{n} (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

©STEIN 2021

Linear Regression

Higher-Dimensional Feature Space

- Recall Equation (1) :

$$\text{RSS}(\mathbf{w}) = \sum_{(\mathbf{x}_i, y_i) \in D} (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

- Let X denote the $n \times (p+1)$ matrix where row i is the extended input vector $(1 \;\; \mathbf{x}_i^T)$ with $(\mathbf{x}_i, y_i) \in D$.

  Let $\mathbf{y}$ denote the n-vector of target values y_i in the training set D.

©STEIN 2021

Linear Regression

Higher-Dimensional Feature Space

- Recall Equation (1) :

$$\text{RSS}(\mathbf{w}) = \sum_{(\mathbf{x}_i, y_i) \in D} (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

- Let X denote the $n \times (p+1)$ matrix where row i is the extended input vector $(1 \;\; \mathbf{x}_i^T)$ with $(\mathbf{x}_i, y_i) \in D$.

  Let $\mathbf{y}$ denote the n-vector of target values y_i in the training set D.

$\rightsquigarrow$ $\text{RSS}(\mathbf{w}) = (\mathbf{y} - X\mathbf{w})^T (\mathbf{y} - X\mathbf{w})$

RSS$(\mathbf{w})$ is a quadratic function in $p+1$ parameters.

©STEIN 2021

Linear Regression

Higher-Dimensional Feature Space (continued) [one-dimensional]

Minimize RSS$(\mathbf{w})$ by a direct method:

$$\frac{\partial\,\text{RSS}}{\partial \mathbf{w}} = -2X^T(\mathbf{y} - X\mathbf{w}) = 0, \qquad \frac{\partial^2\,\text{RSS}}{\partial \mathbf{w}\partial \mathbf{w}^T} = -2X^T X$$

[Wikipedia 1, 2, 3]

$$\begin{aligned} & X^T(\mathbf{y} - X\mathbf{w}) &= 0 \\ \Leftrightarrow \quad & X^T X\mathbf{w} &= X^T\mathbf{y} \\ \rightsquigarrow \quad & \mathbf{w} &= (X^T X)^{-1} X^T\,\mathbf{y} \end{aligned}$$

©STEIN 2021

Linear Regression

Higher-Dimensional Feature Space (continued) [one-dimensional]

Minimize RSS$(\mathbf{w})$ by a direct method:

$$\frac{\partial\,\text{RSS}}{\partial \mathbf{w}} = -2X^T(\mathbf{y} - X\mathbf{w}) = 0, \qquad \frac{\partial^2\,\text{RSS}}{\partial \mathbf{w}\partial \mathbf{w}^T} = -2X^TX$$

[Wikipedia 1, 2, 3]

$$\begin{aligned} X^T(\mathbf{y} - X\mathbf{w}) &= 0 \\ \Leftrightarrow \quad X^TX\mathbf{w} &= X^T\mathbf{y} \\ \rightsquigarrow \quad \mathbf{w} &= \underbrace{(X^TX)^{-1}X^T}_{\text{Pseudoinverse of } X \text{ [Wikipedia]}}\mathbf{y} \end{aligned}$$

Normal equations.

If X has full column rank $p+1$.

©STEIN 2021

Linear Regression

Higher-Dimensional Feature Space (continued) [one-dimensional]

Minimize RSS$(\mathbf{w})$ by a direct method:

$$\frac{\partial\,\text{RSS}}{\partial \mathbf{w}} = -2X^T(\mathbf{y} - X\mathbf{w}) = 0, \qquad \frac{\partial^2\,\text{RSS}}{\partial \mathbf{w}\partial \mathbf{w}^T} = -2X^TX$$

[Wikipedia 1, 2, 3]

$$\begin{aligned} X^T(\mathbf{y} - X\mathbf{w}) &= 0 \\ \Leftrightarrow \quad X^TX\mathbf{w} &= X^T\mathbf{y} \\ \rightsquigarrow \quad \hat{\mathbf{w}} \equiv \mathbf{w} &= \underbrace{(X^TX)^{-1}X^T}_{\text{Pseudoinverse of } X \text{ [Wikipedia]}}\mathbf{y} \end{aligned}$$

Normal equations.

If X has full column rank $p+1$.

$$\hat{y}(\mathbf{x}_i) = \hat{\mathbf{w}}^T\mathbf{x}_i$$

Regression function with least squares estimator $\hat{\mathbf{w}}$.

$$\begin{aligned} \hat{\mathbf{y}} &= X\hat{\mathbf{w}} \\ &= X(X^TX)^{-1}X^T\mathbf{y} \end{aligned}$$

The n-vector of fitted values at the training input.

©STEIN 2021

Remarks:

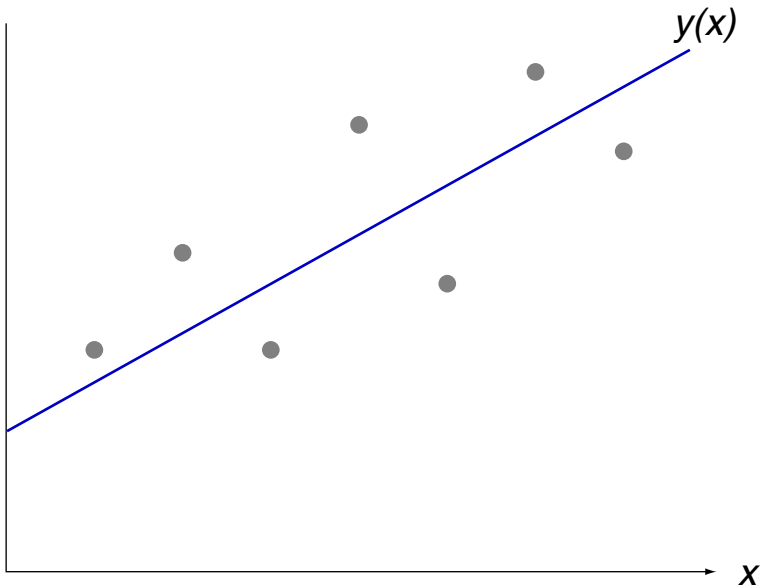
- A curve fitting (or regression) method that is based on the minimization of squared residuals is called a *method of least squares*.
- Various approaches for operationalizing the method of least squares have been devised, in particular for the case of linear model functions. From a numerical viewpoint one can distinguish iterative methods, such as the LMS algorithm, and direct methods, such as solving the normal equations via computing the pseudoinverse.
- More on direct methods. While solving the normal equations is usually fast, it suffers from several deficits: it is numerically unstable and requires singularity handling. Numerically more stable and more accurate methods are based on the QR decomposition and the singular value decomposition, SVD.
- QR decomposition can deal with problems of up to 10^4 variables, provided a dense problem structure. For significantly larger problems (additional 1-2 orders of magnitudes) as well as for sparse matrices iterative solvers are the choice. Even larger, dense problems may be tackled with Artificial Neural Networks.

©STEIN 2021

Linear Regression

Linear Regression for Classification (illustrated for $p = 1$)

Regression learns a real-valued function given as $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$.



$$y(x) = (w_0 \;\; w_1)\begin{pmatrix}1\\x\end{pmatrix}$$

©STEIN 2021

Linear Regression

Linear Regression for Classification (illustrated for $p = 1$)

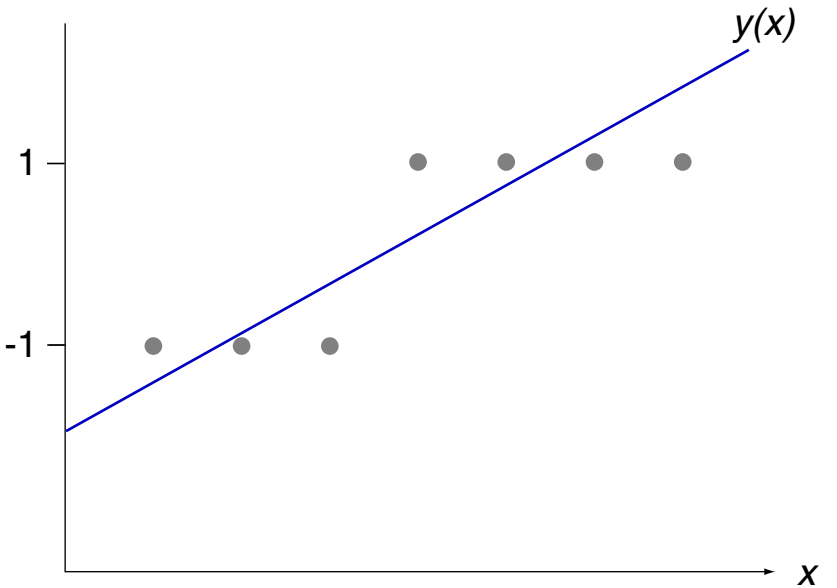
Binary-valued (± 1) functions are also real-valued.

©STEIN 2021

Linear Regression

Linear Regression for Classification (illustrated for $p = 1$)

Use linear regression to learn $\mathbf{w}$ from D, where $y_i = \pm 1 \approx y(\mathbf{x}_i) \overset{(\star)}{=} \mathbf{w}^T \mathbf{x}_i$.



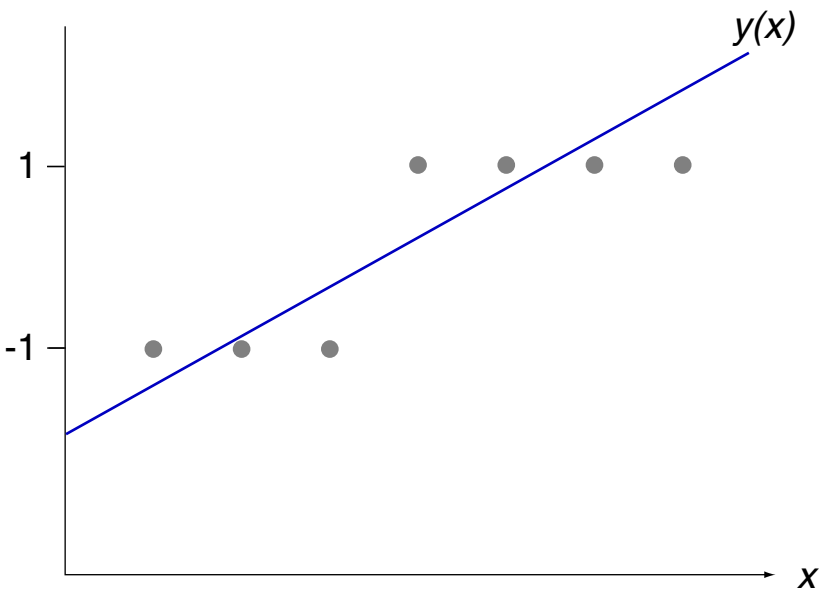
$$y(x) = (w_0 \;\; w_1) \begin{pmatrix} 1 \\ x \end{pmatrix}$$

©STEIN 2021

Linear Regression

Linear Regression for Classification (illustrated for $p = 1$)

Use linear regression to learn $\mathbf{w}$ from D, where $y_i = \pm 1 \approx y(\mathbf{x}_i) \overset{(\star)}{=} \mathbf{w}^T \mathbf{x}_i$.



The function "sign$(\mathbf{w}^T \mathbf{x}_i)$" is likely to agree with $y_i = \pm 1$.

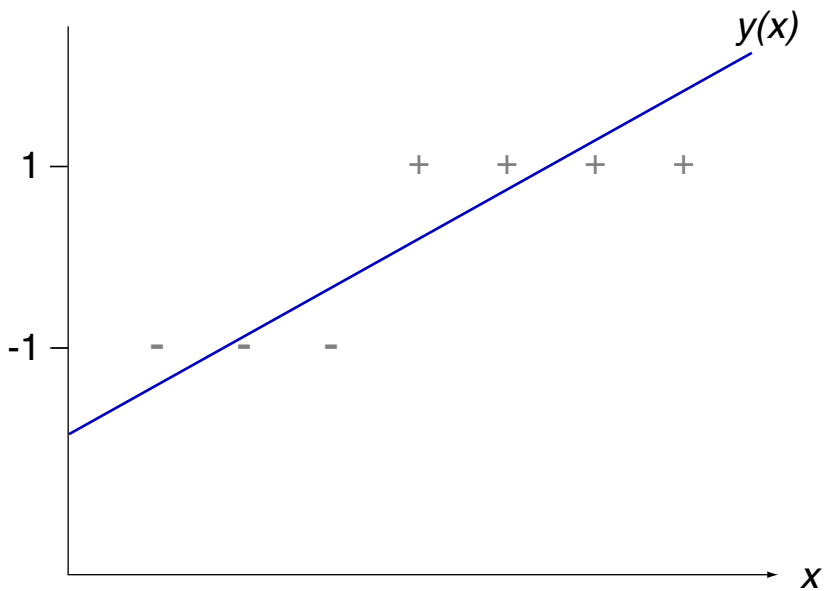
- Regression: $y(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$
- Classification: $y(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x})$

©STEIN 2021

Linear Regression

Linear Regression for Classification (illustrated for $p = 1$)

Use linear regression to learn $\mathbf{w}$ from D, where $y_i = \pm 1 \approx y(\mathbf{x}_i) \overset{(\star)}{=} \mathbf{w}^T \mathbf{x}_i$.



The function "sign$(\mathbf{w}^T \mathbf{x}_i)$" is likely to agree with $y_i = \pm 1$.

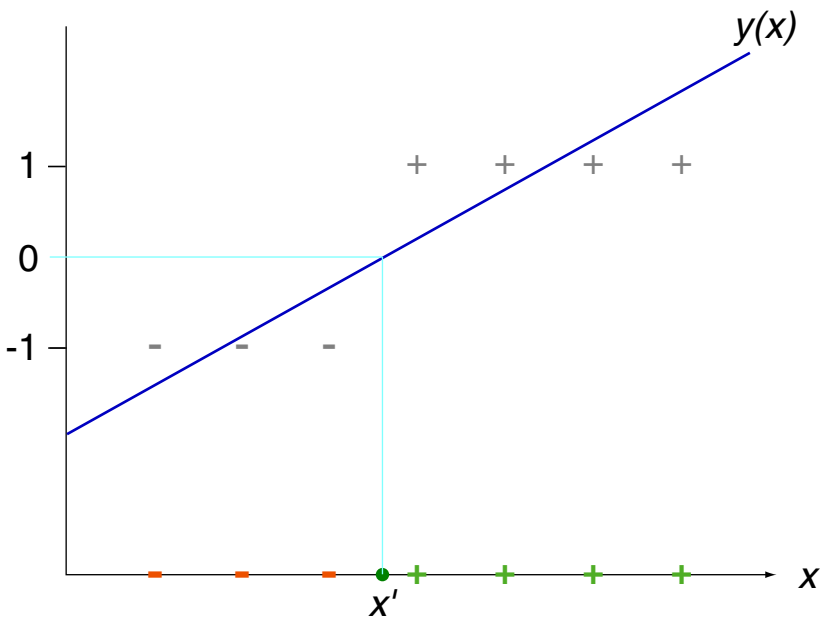
- Regression: $y(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$
- Classification: $y(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x})$

©STEIN 2021

Linear Regression

Linear Regression for Classification (illustrated for $p = 1$)

Use linear regression to learn $\mathbf{w}$ from D, where $y_i = \pm 1 \approx y(\mathbf{x}_i) \overset{(\star)}{=} \mathbf{w}^T \mathbf{x}_i$.



The function "sign$(\mathbf{w}^T \mathbf{x}_i)$" is likely to agree with $y_i = \pm 1$.

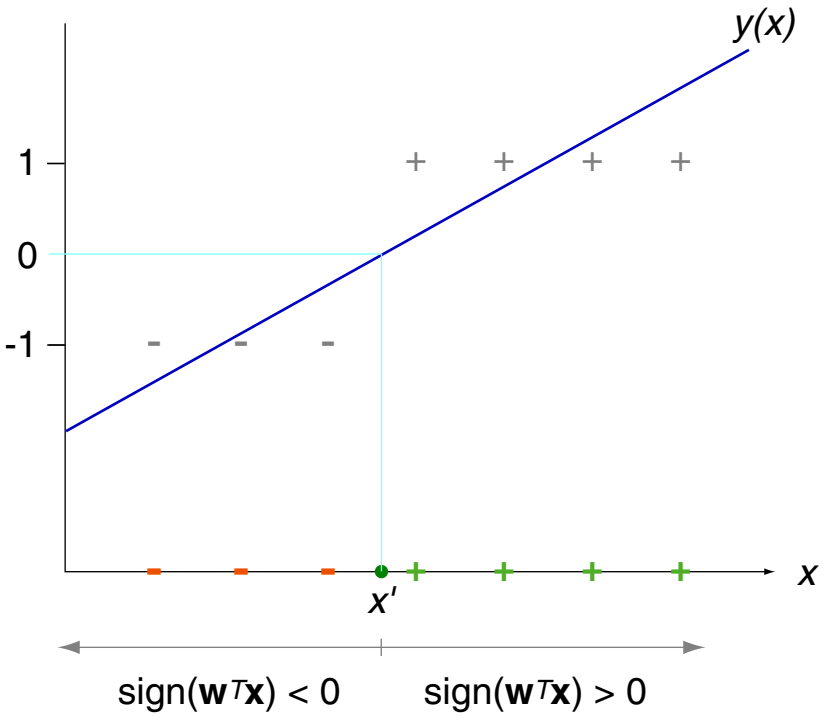
- Regression: $y(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$
- Classification: $y(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x})$

©STEIN 2021

Linear Regression

Linear Regression for Classification (illustrated for $p = 1$)

Use linear regression to learn $\mathbf{w}$ from D, where $y_i = \pm 1 \approx y(\mathbf{x}_i) \overset{(\star)}{=} \mathbf{w}^T \mathbf{x}_i$.



- The discrimination point, •, is defined by $w_0 + w_1 \cdot x' = 0$.
- For $p = 2$ we are given a discrimination *line*.

©STEIN 2021

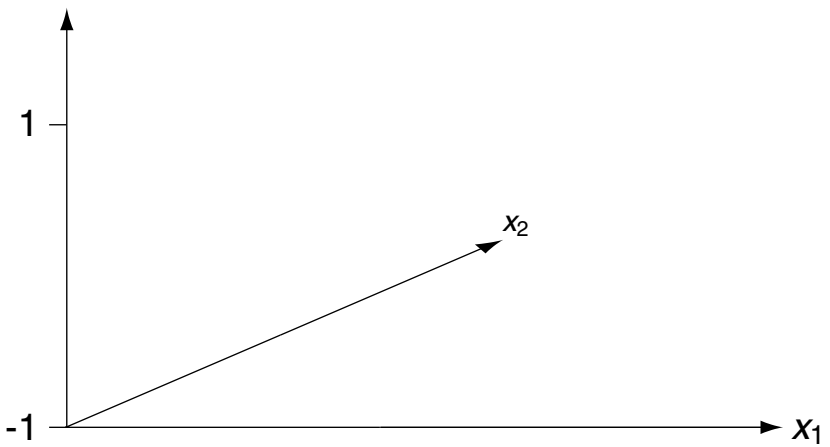
Remarks:

($\star$) When used as operand in a scalar product with the weight vector, $\mathbf{w}^T\mathbf{x}$, we consider the feature vector $\mathbf{x}$ in its extended form. I.e., $x_0 = 1$, and $\mathbf{x} = (1, x_1, \ldots, x_p)^T \in \mathbf{R}^{p+1}$.

©STEIN 2021

Linear Regression

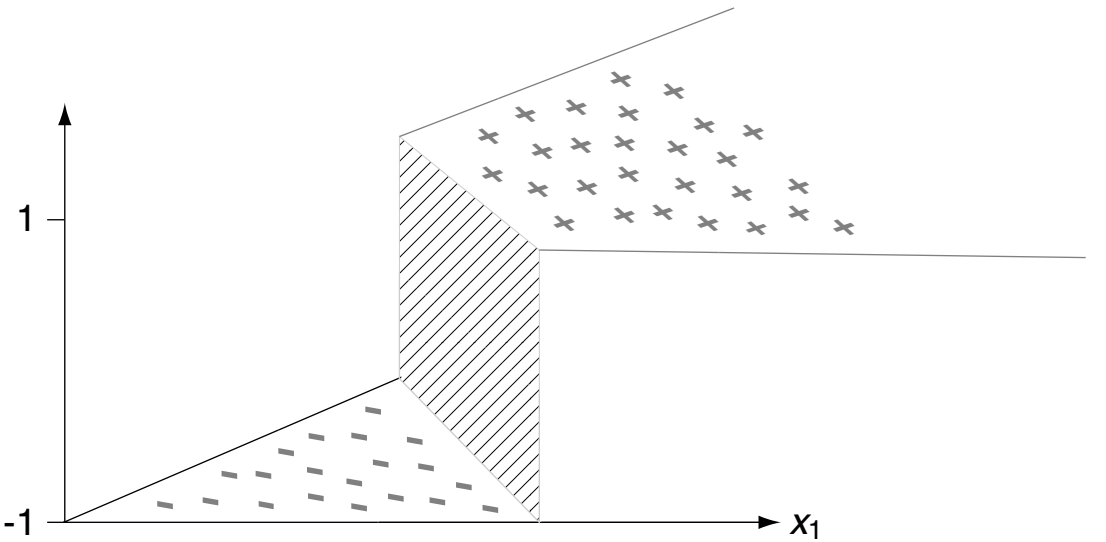
Linear Regression for Classification (illustrated for $p = 2$)



©STEIN 2021

Linear Regression

Linear Regression for Classification (illustrated for $p = 2$)

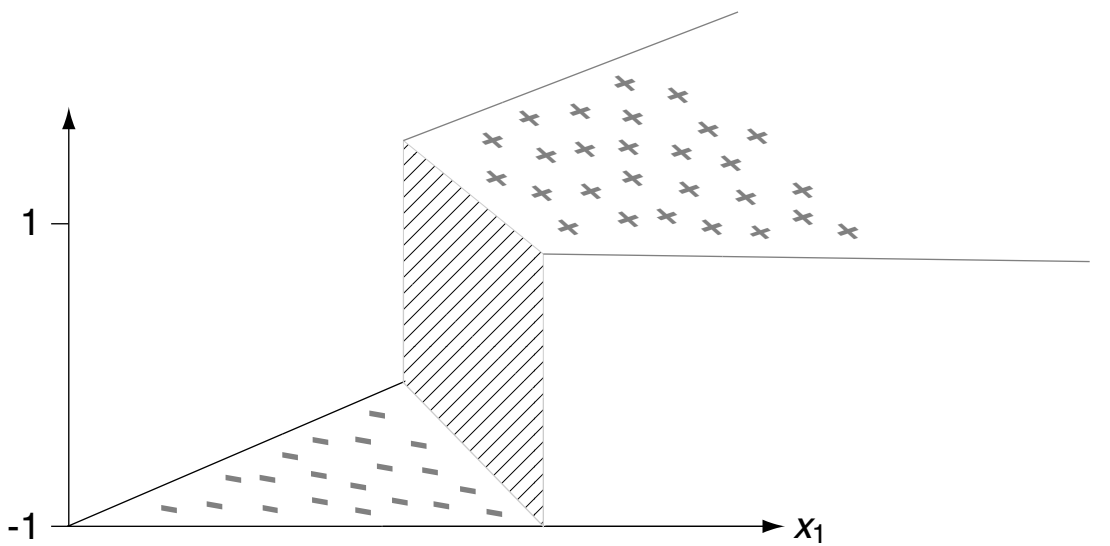


©STEIN 2021

Linear Regression

Linear Regression for Classification (illustrated for $p = 2$)

Use linear regression to learn $\mathbf{w}$ from D, where $y_i = \pm 1 \approx y(\mathbf{x}_i) \overset{(\star)}{=} \mathbf{w}^T \mathbf{x}_i$.



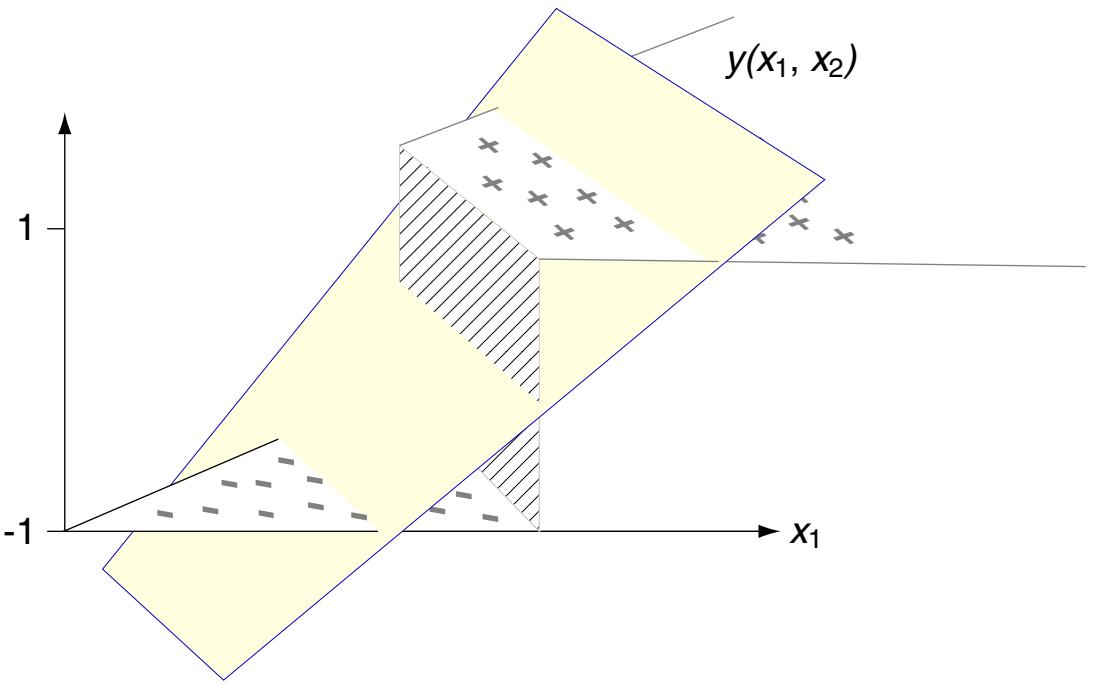
$$y(x_1, x_2) = (w_0 \;\; w_1 \;\; w_2) \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}$$

©STEIN 2021

Linear Regression

Linear Regression for Classification (illustrated for $p = 2$)

Use linear regression to learn $\mathbf{w}$ from D, where $y_i = \pm 1 \approx y(\mathbf{x}_i) \overset{(\star)}{=} \mathbf{w}^T \mathbf{x}_i$.



The function "sign$(\mathbf{w}^T \mathbf{x}_i)$" is likely to agree with $y_i = \pm 1$.

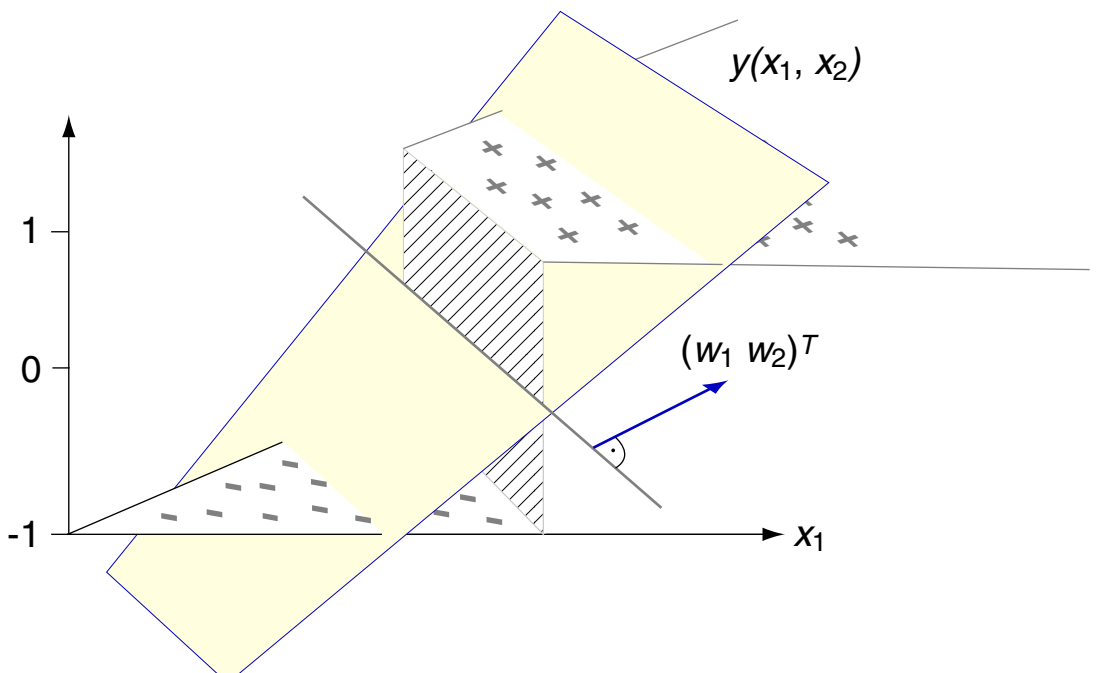
- Regression: $y(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$
- Classification: $y(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x})$

©STEIN 2021

Linear Regression

Linear Regression for Classification (illustrated for $p = 2$)

Use linear regression to learn $\mathbf{w}$ from D, where $y_i = \pm 1 \approx y(\mathbf{x}_i) \overset{(\star)}{=} \mathbf{w}^T \mathbf{x}_i$.



The function "sign$(\mathbf{w}^T \mathbf{x}_i)$" is likely to agree with $y_i = \pm 1$.

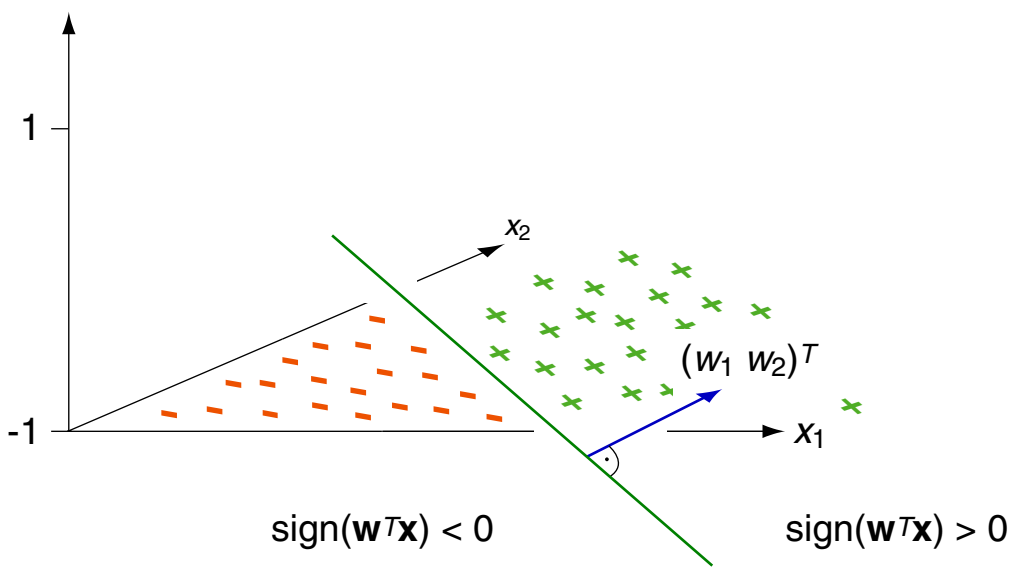
- Regression: $y(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$
- Classification: $y(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x})$

©STEIN 2021

Linear Regression

Linear Regression for Classification (illustrated for $p = 2$)

Use linear regression to learn $\mathbf{w}$ from D, where $y_i = \pm 1 \approx y(\mathbf{x}_i) \overset{(\star)}{=} \mathbf{w}^T \mathbf{x}_i$.



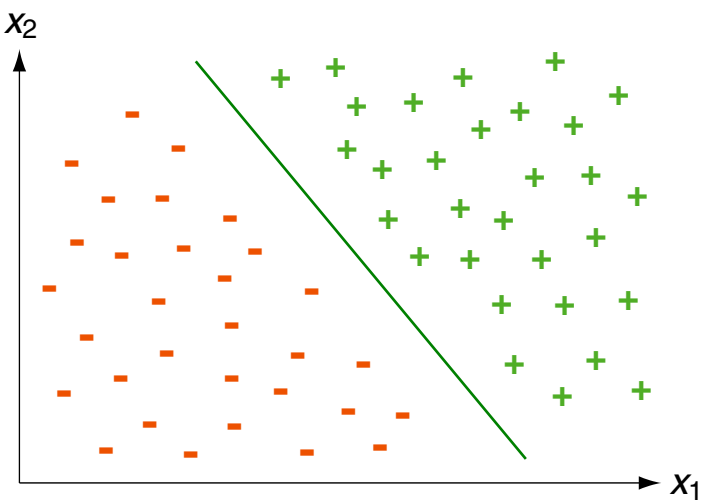
- The discrimination line, —, is defined by $w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 = 0$.
- For $p = 3$ $(p > 3)$ we are given a discriminating *(hyper)plane*.

©STEIN 2021

Linear Regression

Linear Regression for Classification (illustrated for $p = 2$)

Use linear regression to learn $\mathbf{w}$ from D, where $y_i = \pm 1 \approx y(\mathbf{x}_i) \overset{(\star)}{=} \mathbf{w}^T \mathbf{x}_i$.



- The discrimination line, —, is defined by $w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 = 0$.
- For $p = 3$ $(p > 3)$ we are given a discriminating *(hyper)plane*.

©STEIN 2021

Remarks:

- The shown figures illustrate how (linear) regression methods that are applied in the input-output space implicitly define a hyperplane in the input space.
  Note that, in general, linear regression is not the best choice to solve classification problems: imbalanced class distributions and outliers can severely impair the classification effectiveness.

- A suited regression method for classification is logistic regression, which estimates the probability of class membership. Note that also logistic regression is a linear classifier since its encoded hypothesis is a linear function in the parameters $\mathbf{w}$. An illustration of the input-output space of the logistic regression model along with the implicitly defined hyperplane is shown here.

©STEIN 2021

Linear Regression

Linear Model Function: Variants

The components (variables, random variables) of the input vector $\mathbf{x} = (x_1, \ldots, x_p)$ may stem from different sources [Hastie et al. 2001] :

1. quantitative inputs
2. transformations of quantitative inputs, such as $\log x_j$, $\sqrt{x_j}$
3. basis expansions, such as $x_j = (x_1)^j$
4. encoding of a qualitative variable g, $g \in \{1, \ldots, p\}$, as $x_j = I(g = j)$
5. interactions between variables, such as $x_3 = x_1 \cdot x_2$

©STEIN 2021

Linear Regression

Linear Model Function: Variants

The components (variables, random variables) of the input vector $\mathbf{x} = (x_1, \ldots, x_p)$ may stem from different sources [Hastie et al. 2001] :

1. quantitative inputs
2. transformations of quantitative inputs, such as $\log x_j$, $\sqrt{x_j}$
3. basis expansions, such as $x_j = (x_1)^j$
4. encoding of a qualitative variable g, $g \in \{1, \ldots, p\}$, as $x_j = I(g = j)$
5. interactions between variables, such as $x_3 = x_1 \cdot x_2$

No matter the source of the x_j, the model is still linear in its parameters $\mathbf{w}$:

$$y(\mathbf{x}) = w_0 + \sum_{j=1}^{p} w_j \cdot \phi_j(x_j)$$

©STEIN 2021

Linear Regression

Linear Model Function: Variants

The components (variables, random variables) of the input vector $\mathbf{x} = (x_1, \ldots, x_p)$ may stem from different sources [Hastie et al. 2001] :

1. quantitative inputs
2. transformations of quantitative inputs, such as $\log x_j$, $\sqrt{x_j}$
3. basis expansions, such as $x_j = (x_1)^j$
4. encoding of a qualitative variable g, $g \in \{1, \ldots, p\}$, as $x_j = I(g = j)$
5. interactions between variables, such as $x_3 = x_1 \cdot x_2$

No matter the source of the x_j, the model is still linear in its parameters $\mathbf{w}$:

$$y(\mathbf{x}) = w_0 + \sum_{j=1}^{p} w_j \cdot \phi_j(x_j)$$

- linear in the parameters: constant w_j and additive combination

©STEIN 2021

Linear Regression

Linear Model Function: Variants

The components (variables, random variables) of the input vector $\mathbf{x} = (x_1, \ldots, x_p)$ may stem from different sources [Hastie et al. 2001] :

1. quantitative inputs
2. transformations of quantitative inputs, such as $\log x_j$, $\sqrt{x_j}$
3. basis expansions, such as $x_j = (x_1)^j$
4. encoding of a qualitative variable g, $g \in \{1, \ldots, p\}$, as $x_j = I(g = j)$
5. interactions between variables, such as $x_3 = x_1 \cdot x_2$

No matter the source of the x_j, the model is still linear in its parameters $\mathbf{w}$:

$$y(\mathbf{x}) = w_0 + \sum_{j=1}^{p} w_j \cdot \phi_j(x_j)$$

- linear in the parameters: constant w_j and additive combination
- basis functions: input variables (space) become(s) feature variables (space)

©STEIN 2021

Linear Regression

Linear Model Function: Properties of the Solution

Theorem 1 (Gauss-Markov)

Let $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ be a set of examples to be fitted with a linear model function as $y(\mathbf{x}) \overset{(\star)}{=} \mathbf{x}^T \mathbf{w}$. Within the class of linear unbiased estimators for $\mathbf{w}$, the least squares estimator $\hat{\mathbf{w}}$ has minimum variance, i.e., is most efficient.

©STEIN 2021

Linear Regression

Linear Model Function: Properties of the Solution

Theorem 1 (Gauss-Markov)

Let $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ be a set of examples to be fitted with a linear model function as $y(\mathbf{x}) \overset{(\star)}{=} \mathbf{x}^T \mathbf{w}$. Within the class of linear unbiased estimators for $\mathbf{w}$, the least squares estimator $\hat{\mathbf{w}}$ has minimum variance, i.e., is most efficient.

Related follow-up issues:

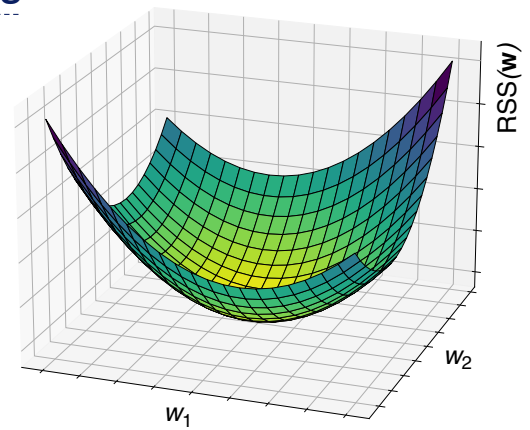
- mean and variance of $\hat{\mathbf{w}}$
- proof of the Gauss-Markov theorem
- weak set and strong set of assumptions
- efficiency and consistency of unbiased estimators
- rank deficiencies, where the feature number p exceeds $|D| = n$
- relation of mean least squares and the maximum likelihood principle

©STEIN 2021

Linear Regression

Methods of Least Squares: Iterative versus Direct Methods

$$\underset{\mathbf{w}}{\text{argmin}}\ \text{RSS}(\mathbf{w}), \quad \text{with } \text{RSS}(\mathbf{w}) = \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

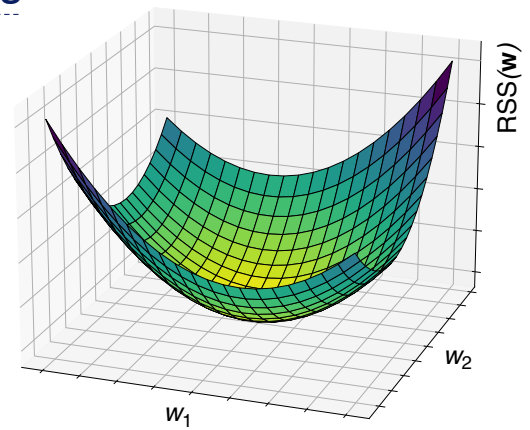


©STEIN 2021

Linear Regression

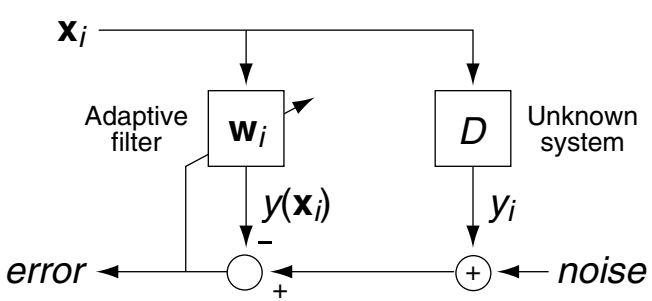
Methods of Least Squares: Iterative versus Direct Methods

$$\underset{\mathbf{w}}{\text{argmin}}\ \text{RSS}(\mathbf{w}), \quad \text{with } \text{RSS}(\mathbf{w}) = \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$



LMS algorithm:

- applicable as online algorithm
- robust algorithm structure
- unsatisfactory convergence
- allows stochastic sampling

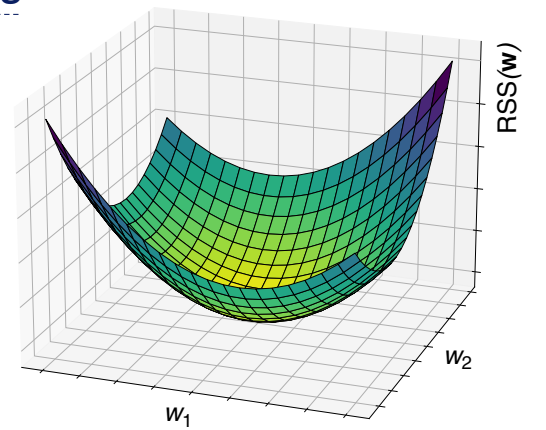


©STEIN 2021

Linear Regression

Methods of Least Squares: Iterative versus Direct Methods

$$\underset{\mathbf{w}}{\text{argmin}}\ \text{RSS}(\mathbf{w}), \quad \text{with } \text{RSS}(\mathbf{w}) = \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

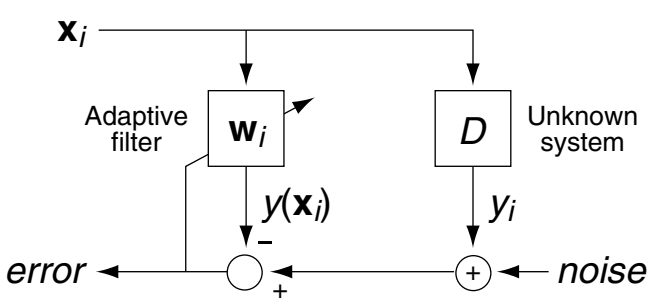


LMS algorithm:

- applicable as online algorithm
- robust algorithm structure
- unsatisfactory convergence
- allows stochastic sampling

Normal equations:

- needs complete data
- numerically unstable
- requires singularity handling
- hardly applicable to big data



$$\hat{\mathbf{w}} = (X^T X)^{-1} X^T \mathbf{y}$$

©STEIN 2021

Remarks:

- Recall the definitions of residual and method of least squares.
- The principle of RSS minimization is orthogonal to (= independent of) the type of the model function $y(\mathbf{x})$, i.e., independent of its dimensionality as well as its linearity or nonlinearity.
- To fit the parameters $\mathbf{w}$ of a (one-dimensional, multi-dimensional, linear, nonlinear) model function $y(\mathbf{x})$, both the LMS algorithm and direct methods exploit information about the derivative of the RSS term with respect to $\mathbf{w}$. I.e., even if *classification* and not regression is the goal, the distance to the decision boundary (and not the zero-one-loss) is computed, since the latter is not differentiable.
- For a linear model function $y(\mathbf{x})$, RSS$(\mathbf{w})$ is a convex function and hence a single, global optimum exists.
- Forthcoming: A main goal of machine learning approaches is to avoid overfitting. Overfitting, in turn, is caused by an inadequate (too high) model function complexity—or, similarly, by insufficient data. A means to reduce the model function complexity is *regularization*.
- Regularization will introduce additional constraints for the model function $y(\mathbf{x})$ or the parameter vector $\mathbf{w}$, respectively. With regularization the minimization expression (2) will consist of two summands: a performance term such as the RSS term, and a penalizing term such as a norm. As before, the first term captures the model function's goodness depending on $\mathbf{w}$, whereas the second term restricts the absolute values of the model function's parameters $\mathbf{w}$.

©STEIN 2021

Chapter ML:II (continued)

II. Machine Learning Basics

- Linear Regression
- Concept Learning: Search in Hypothesis Space
- Concept Learning: Version Space
- Evaluating Effectiveness

©STEIN 2021

Concept Learning: Search in Hypothesis Space

Simple Classification Problems

Setting:

- X is a set of feature vectors.
- C is a set of two classes: $\{0, 1\}$, $\{\text{yes}, \text{no}\}$, "belongs to a concept or not"
- $c : X \to C$ is the (unknown) ideal classifier for X.
- $D = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, c(\mathbf{x}_n))\} \subseteq X \times C$ is a set of examples.

Todo:

- Approximate $c(\mathbf{x})$, which is implicitly given via D, with a feature-value pattern.

©STEIN 2021

Concept Learning: Search in Hypothesis Space

An Exemplary Learning Task

The set X is comprised of vectors that describe the weather condition in the six dimensions "Sky", "Temperature", "Humidity", "Wind", "Water", and "Forecast". The set D are examples of weather conditions $\mathbf{x} \in X$ along with a statement whether or not our friend will enjoy her favorite sport:

Example	Sky	Temperature	Humidity	Wind	Water	Forecast	EnjoySport
1	sunny	warm	normal	strong	warm	same	yes
2	sunny	warm	high	strong	warm	same	yes
3	rainy	cold	high	strong	warm	change	no
4	sunny	warm	high	strong	cool	change	yes

- What is the concept behind "EnjoySport" ?
- What are possible hypotheses to formalize the concept "EnjoySport" ?

  Similarly: What are the elements of the set or class "EnjoySport" ?

©STEIN 2021

Remarks:

- Domains of the features in the learning task:

Sky	Temperature	Humidity	Wind	Water	Forecast
sunny	warm	normal	strong	warm	same
rainy	cold	high	light	cool	change
cloudy					

- A concept is a subset of a larger set of objects. In the exemplary learning task the larger object set is comprised of all possible weather conditions, while the subset (= concept) contains those weather conditions when surfing can be enjoyed.
- A hypothesis is expected to “capture a (target) concept”, to “explain a (target) concept”, or to “predict a (target) concept” in terms of the feature expressions of the objects.
- The “quality”, the “persuasiveness”, or the “power” of a hypothesis depends on its capability to represent (= to explain) a given set of observations, which are called examples here.
- A hypothesis cannot be inferred or proven by deductive reasoning. A hypothesis is a finding or an insight gained by inductive reasoning.

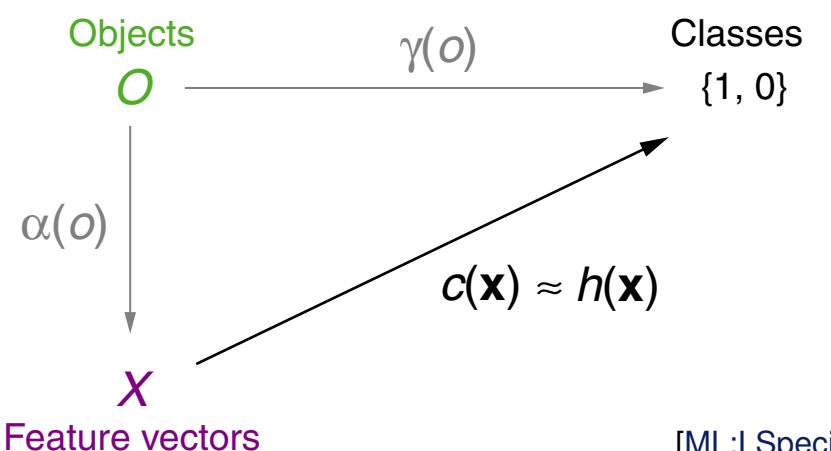
©STEIN 2021

Concept Learning: Search in Hypothesis Space

Definition 1 (Concept, Hypothesis, Hypothesis Space)

A concept is a subset of an object set O and hence determines a subset of $X = \alpha(O)$. Concept learning is the approximation of the ideal classifier $c : X \to \{0, 1\}$, where

$$c(\mathbf{x}) = \begin{cases} 1 & \text{if } \alpha^{-1}(\mathbf{x}) \text{ belongs to the concept,} \\ 0 & \text{otherwise.} \end{cases}$$



[ML:I Specification of Learning Problems]

©STEIN 2021

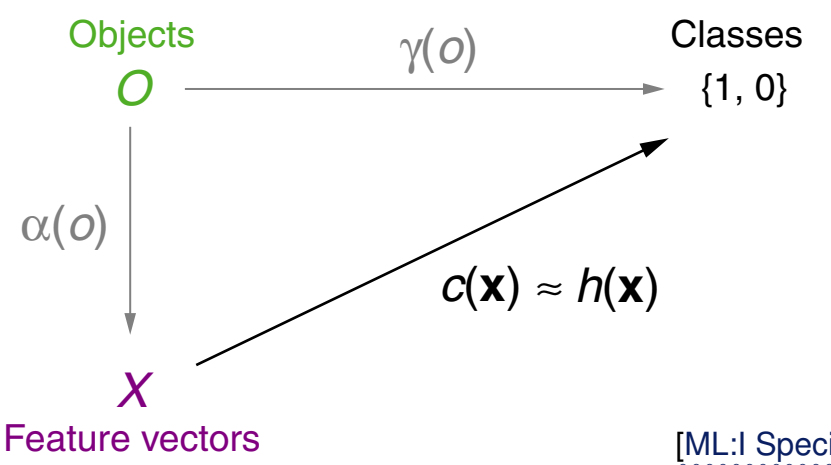
Concept Learning: Search in Hypothesis Space

Definition 1 (Concept, Hypothesis, Hypothesis Space)

A concept is a subset of an object set O and hence determines a subset of $X = \alpha(O)$. Concept learning is the approximation of the ideal classifier $c : X \to \{0, 1\}$, where

$$c(\mathbf{x}) = \begin{cases} 1 & \text{if } \alpha^{-1}(\mathbf{x}) \text{ belongs to the concept,} \\ 0 & \text{otherwise.} \end{cases}$$

A hypothesis is a function $h(\mathbf{x})$, $h : X \to \{0, 1\}$, to approximate $c(\mathbf{x})$. A hypothesis space is a set H of hypotheses among which $h(\mathbf{x})$ is searched.



[ML:I Specification of Learning Problems]

©STEIN 2021

Remarks:

- A hypothesis may also be called *model function* or *model*. Note however, that it is common practice to designate only the *parameters* of a model function, $\boldsymbol{\theta}$ or $\mathbf{w}$, as hypothesis (and not the function itself), especially if the setting focuses on a certain class of models, such as linear models, polynomials of a fixed degree, or Gaussian distributions.

- The subtle semantic distinction between the terms “model function” and “hypothesis” made in machine learning is that the former term is typically used to denote a function class or a particular class of computational approaches, while the latter term refers to a specific instance of that class.

- Depending on the learning task—more specifically: on the structure of the feature space—a hypothesis (model function) can take different forms and, accordingly, is denoted differently: $h(\mathbf{x})$ (as done here), $y(\mathbf{x})$ (in regression settings), T (for decision trees), $\prod P(A \mid B)$ (within statistical learning), etc.

©STEIN 2021

Concept Learning: Search in Hypothesis Space

The example set D, $D = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, c(\mathbf{x}_n))\}$, contains usually both positive ($c(\mathbf{x}) = 1$) and negative ($c(\mathbf{x}) = 0$) examples. [learning task]

Definition 2 (Hypothesis-Fulfilling, Consistency)

An example $(\mathbf{x}, c(\mathbf{x}))$ fulfills a hypothesis $h(\mathbf{x})$ iff $h(\mathbf{x}) = 1$.

A hypothesis $h(\mathbf{x})$ is consistent with an example $(\mathbf{x}, c(\mathbf{x}))$ iff $h(\mathbf{x}) = c(\mathbf{x})$.

A hypothesis $h(\mathbf{x})$ is consistent with a set D of examples, denoted as *consistent*(h, D), iff:

$$\forall (\mathbf{x}, c(\mathbf{x})) \in D : \; h(\mathbf{x}) = c(\mathbf{x})$$

©STEIN 2021

Remarks:

- The string “Iff” or “iff” is an abbreviation for “If and only if”, which means “necessary and sufficient”. It is a textual representation for the logical biconditional, also known as material biconditional or iff-connective. The respective symbol is “$\leftrightarrow$”. [Wolfram] [Wikipedia]
- The following terms are used synonymously: target concept, target function, ideal classifier.
- The fact that a hypothesis is consistent with an example can also be described the other way round: an example is consistent with a hypothesis.
- Given an example $(\mathbf{x}, c(\mathbf{x}))$, notice the difference between (1) hypothesis-fulfilling and (2) being consistent with a hypothesis. The former asks for $h(\mathbf{x}) = 1$, disregarding the actual target concept value $c(\mathbf{x})$. The latter asks for the identity between the target concept $c(\mathbf{x})$ and the hypothesis $h(\mathbf{x})$.
- The consistency of $h(\mathbf{x})$ can be analyzed for a single example as well as for a set D of examples. Given the latter, consistency requires for all elements in D that $h(\mathbf{x}) = 1$ iff $c(\mathbf{x}) = 1$. This is equivalent with the condition that $h(\mathbf{x}) = 0$ iff $c(\mathbf{x}) = 0$ for all $\mathbf{x} \in D$.
- Learning means to determine a hypothesis $h(\mathbf{x}) \in H$ that is consistent with D.
  Similarly: Machine learning means to systematically search the hypothesis space.

©STEIN 2021

Concept Learning: Search in Hypothesis Space

Simple Classification Problems (continued)

Structure of a hypothesis $h(\mathbf{x})$:

1. conjunction of feature-value pairs
2. three kinds of values: literal, ? (wildcard), $\perp$ (contradiction)

A hypothesis for EnjoySport [learning task] : $\langle$ *sunny, ?, ?, strong, ?, same* $\rangle$

©STEIN 2021

Concept Learning: Search in Hypothesis Space

Simple Classification Problems (continued)

Structure of a hypothesis $h(\mathbf{x})$:

1. conjunction of feature-value pairs
2. three kinds of values: literal, ? (wildcard), $\perp$ (contradiction)

A hypothesis for EnjoySport [learning task]: $\langle$ *sunny, ?, ?, strong, ?, same* $\rangle$

Definition 3 (Maximally Specific / General Hypothesis)

The hypotheses $s_0(\mathbf{x}) \equiv 0$ and $g_0(\mathbf{x}) \equiv 1$ are called maximally specific and maximally general hypothesis respectively. No $\mathbf{x} \in X$ fulfills $s_0(\mathbf{x})$, and all $\mathbf{x} \in X$ fulfill $g_0(\mathbf{x})$.

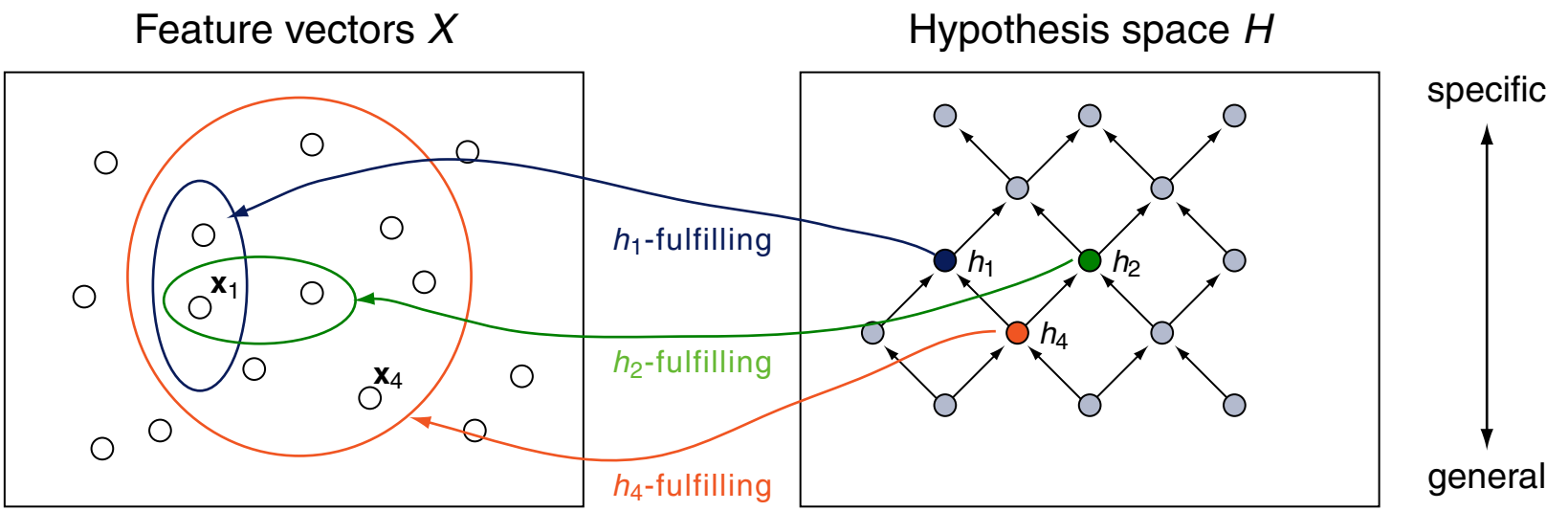
Maximally specific / general hypothesis in the example [learning task]:

- $s_0 = \langle \perp, \perp, \perp, \perp, \perp, \perp \rangle$ (never enjoy sport)
- $g_0 = \langle$ *?, ?, ?, ?, ?, ?* $\rangle$ (always enjoy sport)

©STEIN 2021

Concept Learning: Search in Hypothesis Space

Order of Hypotheses



$\mathbf{x}_1 = (\textit{sunny, warm, normal, strong, warm, same})$ $\quad h_1 = \langle$ *sunny, ?, normal, ?, ?, ?* $\rangle$

$h_2 = \langle$ *sunny, ?, ?, ?, warm, ?* $\rangle$

$\mathbf{x}_4 = (\textit{sunny, warm, high, strong, cool, change})$ $\quad h_4 = \langle$ *sunny, ?, ?, ?, ?, ?* $\rangle$

©STEIN 2021

Concept Learning: Search in Hypothesis Space

Order of Hypotheses

Definition 4 (More General Relation)

Let X be a set of feature vectors and let $h_1(\mathbf{x})$ and $h_2(\mathbf{x})$ be two boolean-valued functions with domain X. Then $h_1(\mathbf{x})$ is called more general than $h_2(\mathbf{x})$, denoted as $h_1(\mathbf{x}) \geq_g h_2(\mathbf{x})$, iff:

$$\forall \mathbf{x} \in X : \; (\; h_2(\mathbf{x}) = 1 \;\; \text{implies} \;\; h_1(\mathbf{x}) = 1 \;)$$

$h_1(\mathbf{x})$ is called stricly more general than $h_2(\mathbf{x})$, denoted as $h_1(\mathbf{x}) >_g h_2(\mathbf{x})$, iff:

$$(h_1(\mathbf{x}) \geq_g h_2(\mathbf{x})) \;\; \text{and} \;\; (h_2(\mathbf{x}) \ngeq_g h_1(\mathbf{x}))$$

©STEIN 2021

Concept Learning: Search in Hypothesis Space

Order of Hypotheses

Definition 4 (More General Relation)

Let X be a set of feature vectors and let $h_1(\mathbf{x})$ and $h_2(\mathbf{x})$ be two boolean-valued functions with domain X. Then $h_1(\mathbf{x})$ is called more general than $h_2(\mathbf{x})$, denoted as $h_1(\mathbf{x}) \geq_g h_2(\mathbf{x})$, iff:

$$\forall \mathbf{x} \in X : \ (\ h_2(\mathbf{x}) = 1 \ \text{ implies } \ h_1(\mathbf{x}) = 1 \ )$$

$h_1(\mathbf{x})$ is called stricly more general than $h_2(\mathbf{x})$, denoted as $h_1(\mathbf{x}) >_g h_2(\mathbf{x})$, iff:

$$(h_1(\mathbf{x}) \geq_g h_2(\mathbf{x})) \ \text{ and } \ (h_2(\mathbf{x}) \ngeq_g h_1(\mathbf{x}))$$

About the maximally specific / general hypothesis:

- $s_0(\mathbf{x})$ is minimum and $g_0(\mathbf{x})$ is maximum with regard to $\geq_g$: no hypothesis is more specific wrt. $s_0(\mathbf{x})$, and no hypothesis is more general wrt. $g_0(\mathbf{x})$.
- We will consider only hypothesis spaces that contain $s_0(\mathbf{x})$ and $g_0(\mathbf{x})$.

©STEIN 2021

Remarks:

- If $h_1(\mathbf{x})$ is more general than $h_2(\mathbf{x})$, then $h_2(\mathbf{x})$ can also be called being more specific than $h_1(\mathbf{x})$.

- $\geq_g$ and $>_g$ are independent of a target concept $c(\mathbf{x})$. They depend only on the fact that examples fulfill a hypothesis, i.e., whether $h(\mathbf{x}) = 1$. It is not required that $c(\mathbf{x}) = 1$.

- The $\geq_g$-relation defines a partial order on the hypothesis space H : $\geq_g$ is reflexive, anti-symmetric, and transitive. The order is *partial* since (unlike in a total order) not all hypothesis pairs stand in the relation. [Wikipedia partial, total]

  I.e., we are given hypotheses $h_i(\mathbf{x})$, $h_j(\mathbf{x})$, for which neither $h_i(\mathbf{x}) \geq_g h_j(\mathbf{x})$ nor $h_j(\mathbf{x}) \geq_g h_i(\mathbf{x})$ holds, such as the hypotheses $h_1(\mathbf{x})$ and $h_2(\mathbf{x})$ in the hypothesis space.

©STEIN 2021

Remarks: (continued)

- The semantics of the implication, in words "a implies b", denoted as $a \to b$, is as follows. $a \to b$ is true if either (1) a is true and b is true, or (2) if a is false and b is true, or (3) if a is false and b is false—in short: "if a is true then b is true as well", or, "the truth of a implies the truth of b".

- "$\to$" can be understood as "causality connective": Let a and b be two events where a is a cause for b. If we interpret the occurrence of an event as true and its non-occurrence as false, we will observe only occurrence combinations such that the formula $a \to b$ is true. The connective is also known as material conditional, material implication, material consequence, or simply, implication or conditional.

- Note in particular that the connective "$\to$" does not mean "entails", which would be denoted as either $\Rightarrow$ or $\models$. Logical entailment (synonymously: logical inference, logical deduction, logical consequence) allows to infer or to prove a fact. Consider for instance the More-General-Definition: From the fact $h_2(\mathbf{x}) = 1$ we cannot infer or prove the fact $h_1(\mathbf{x}) = 1$.

- In the More-General-Definition the implication specifies a condition that is to be fulfilled by the definiendum (= the thing to be defined). The implication is used to check whether or not a thing belongs to the set of things specified by the definiens (= the expression that defines): Each pair of functions, $h_1(\mathbf{x})$, $h_2(\mathbf{x})$, is a thing that belongs to the set of things specified by the definition of the $\geq_g$-relation (i.e., stands in the $\geq_g$-relation) if and only if the implication $h_2(\mathbf{x}) = 1 \;\to\; h_1(\mathbf{x}) = 1$ is true for all $\mathbf{x} \in X$.

©STEIN 2021

Remarks: (continued)

- In a nutshell: distinguish carefully between "α requires β", denoted as $\alpha \to \beta$, on the one hand, and "from α follows β", denoted as $\alpha \Rightarrow \beta$, on the other hand. $\alpha \to \beta$ is considered as a sentence from the *object language* (language of discourse) and stipulates a computing operation, whereas $\alpha \Rightarrow \beta$ is a sentence from the *meta language* and makes an assertion *about* the sentence $\alpha \to \beta$, namely: "$\alpha \to \beta$ is a tautology".

- Finally, consider the following sentences from the object language, which are synonymous: "$\alpha \to \beta$", "α implies β", "if α then β", "α causes β", "α requires β", "α is sufficient for β", "β is necessary for α".

©STEIN 2021

Concept Learning: Search in Hypothesis Space

Inductive Learning Hypothesis

"Any hypothesis found to approximate the target function well
over a sufficiently large set of training examples
will also approximate the target function well
over other unobserved examples." [p.23, Mitchell 1997]

©STEIN 2021

Concept Learning: Search in Hypothesis Space

Find-S Algorithm

```
1.  h(x) = s_0(x)  // h(x) is a maximally specific hypothesis in H.
2.  FOREACH (x, c(x)) ∈ D DO
      IF c(x) = 1 THEN // Use only positive examples.
        IF h(x) = 0 DO
          h = min_generalization(h, x) // Relax h(x) wrt. x.
        ENDIF
      ENDIF
    ENDDO
3.  return(h(x))
```


©STEIN 2021

Remarks:

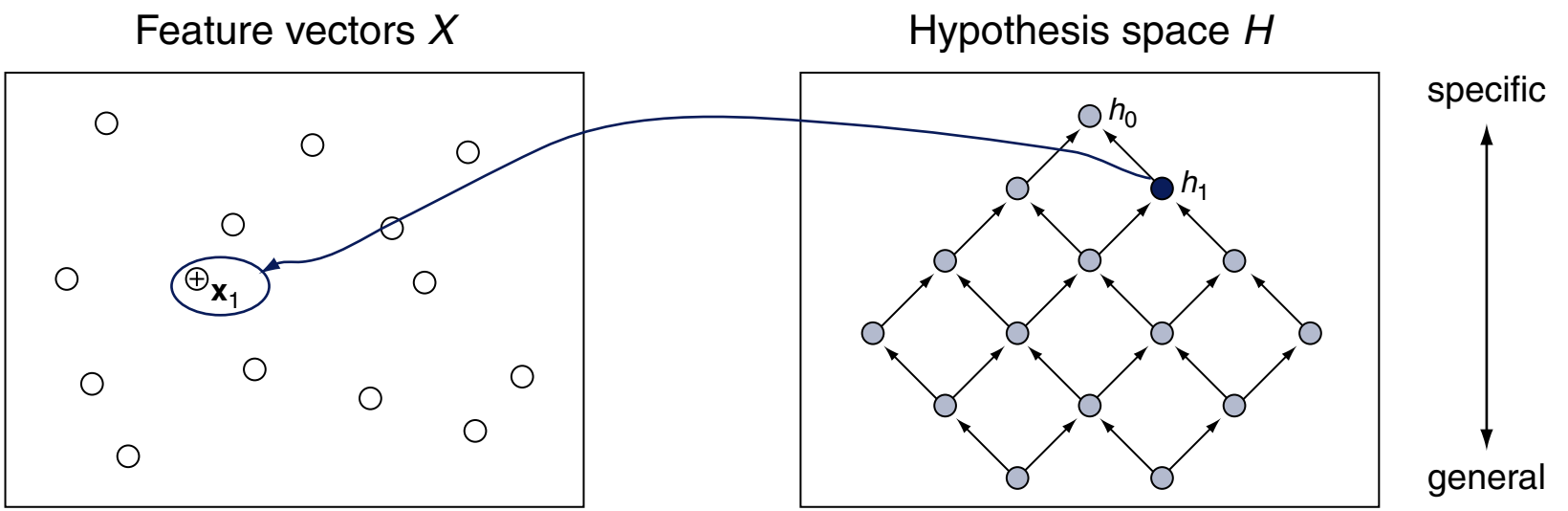
- Another term for “generalization” is “relaxation”.
- The function *min_generalization*$(h, \mathbf{x})$ returns a hypothesis $h'(\mathbf{x})$ that is minimally generalized wrt. $h(\mathbf{x})$ and that is consistent with $(\mathbf{x}, 1)$. Denoted formally: $h'(\mathbf{x}) \geq_g h(\mathbf{x})$ and $h'(\mathbf{x}) = 1$, and there is no $h''(\mathbf{x})$ with $h'(\mathbf{x}) >_g h''(\mathbf{x}) \geq_g h(\mathbf{x})$ with $h''(\mathbf{x}) = 1$.
- For more complex hypothesis structures the relaxation of $h(\mathbf{x})$, *min_generalization*$(h, \mathbf{x})$, may not be unique. In such a case one of the alternatives has to be chosen.
- If a hypothesis $h(\mathbf{x})$ needs to be relaxed towards some $h'(\mathbf{x})$ with $h'(\mathbf{x}) \notin H$, the maximally general hypothesis $g_0 \equiv 1$ can be added to H.
- Similar to *min_generalization*$(h, \mathbf{x})$, a function *min_specialization*$(h, \mathbf{x})$ can be defined, which returns a minimally specialized, consistent hypotheses for negative examples.

©STEIN 2021

Concept Learning: Search in Hypothesis Space

Find-S Algorithm

See the example set D for the concept *EnjoySport*.



$h_0 = s_0 = \langle \perp, \perp, \perp, \perp, \perp, \perp \rangle$

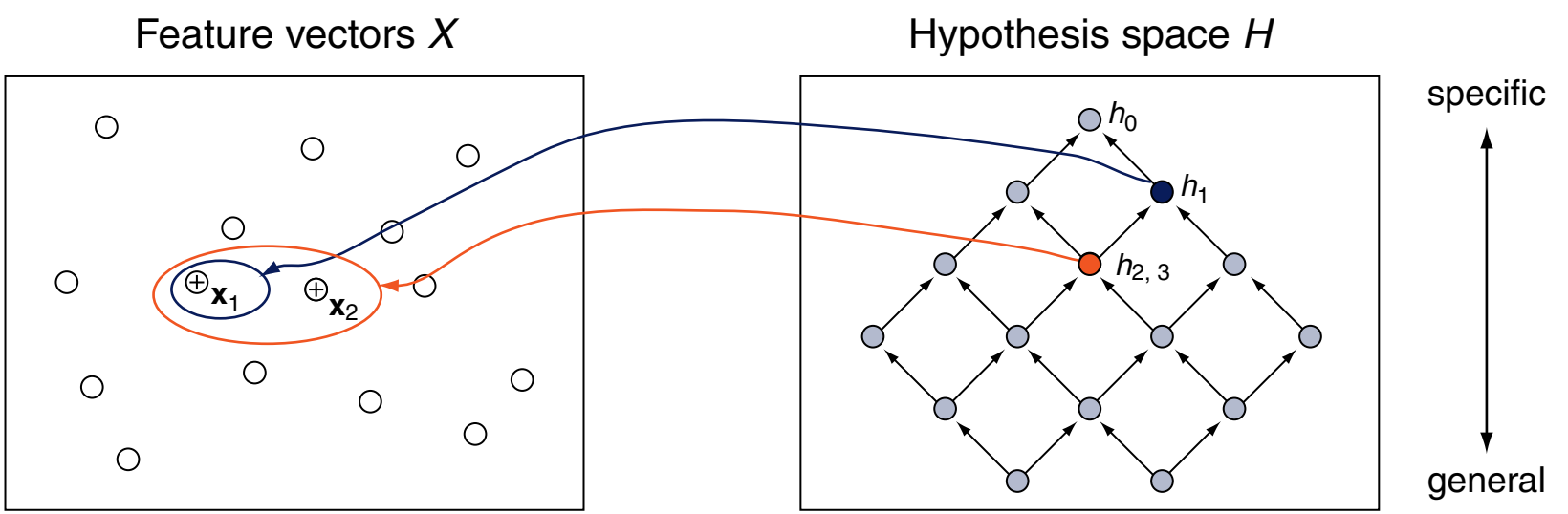
$\mathbf{x}_1 = (\textit{sunny, warm, normal, strong, warm, same})$ $\quad h_1 = \langle \textit{sunny, warm, normal, strong, warm, same} \rangle$

©STEIN 2021

Concept Learning: Search in Hypothesis Space

Find-S Algorithm

See the example set D for the concept *EnjoySport*.



$h_0 = s_0 = \langle\ \perp, \perp, \perp, \perp, \perp, \perp\ \rangle$

$\mathbf{x}_1 = (\textit{sunny, warm, normal, strong, warm, same})$ $\quad h_1 = \langle\ \textit{sunny, warm, normal, strong, warm, same}\ \rangle$

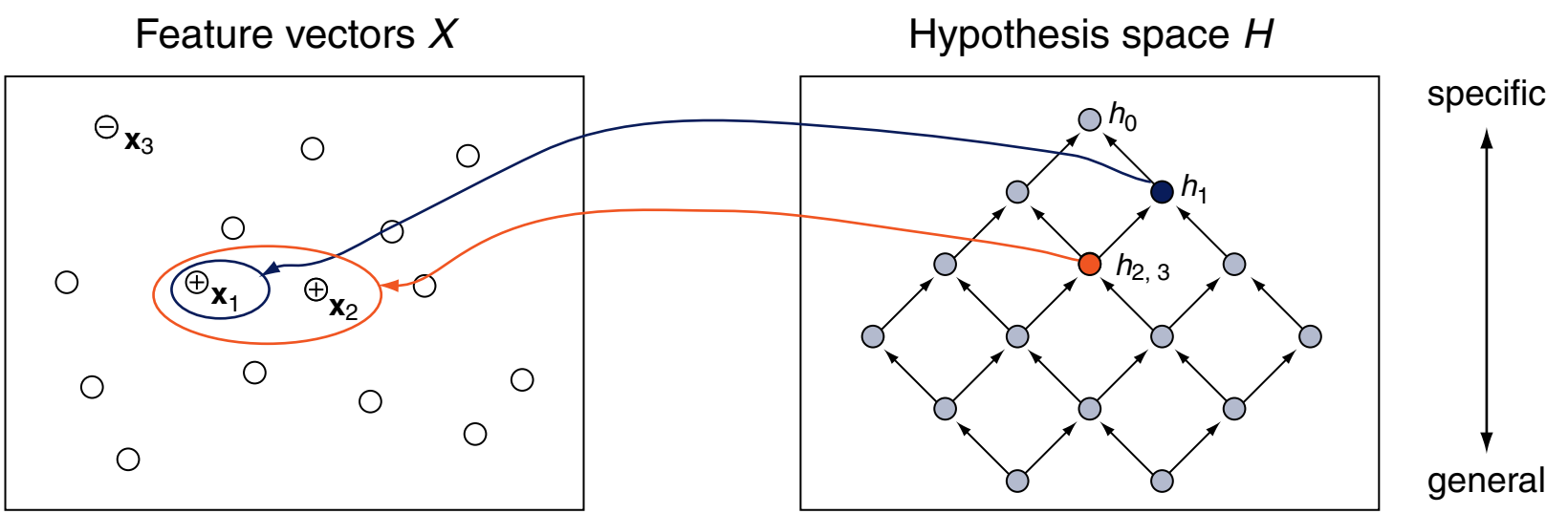
$\mathbf{x}_2 = (\textit{sunny, warm, high, strong, warm, same})$ $\quad h_2 = \langle\ \textit{sunny, warm, ?, strong, warm, same}\ \rangle$

©STEIN 2021

Concept Learning: Search in Hypothesis Space

Find-S Algorithm

See the example set D for the concept *EnjoySport*.



$h_0 = s_0 = \langle\, \perp, \perp, \perp, \perp, \perp, \perp \,\rangle$

$\mathbf{x}_1 = (\textit{sunny, warm, normal, strong, warm, same})$ $\quad h_1 = \langle\, \textit{sunny, warm, normal, strong, warm, same} \,\rangle$

$\mathbf{x}_2 = (\textit{sunny, warm, high, strong, warm, same})$ $\quad h_2 = \langle\, \textit{sunny, warm, ?, strong, warm, same} \,\rangle$

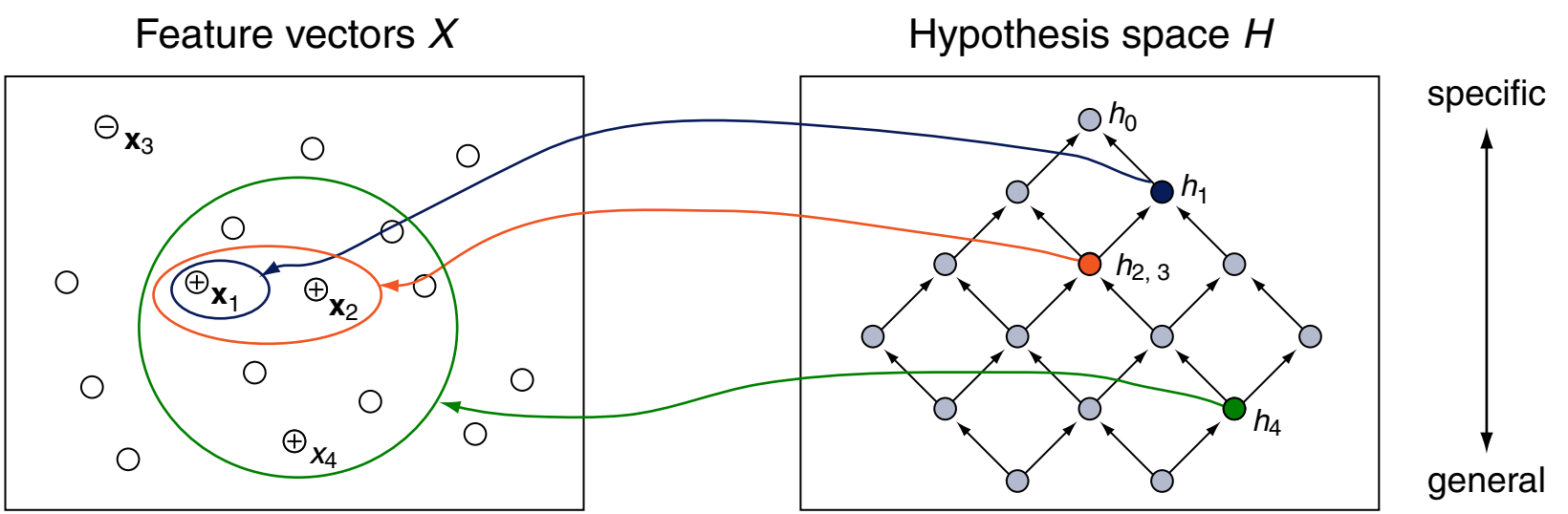
$\mathbf{x}_3 = (\textit{rainy, cold, high, strong, warm, change})$ $\quad h_3 = \langle\, \textit{sunny, warm, ?, strong, warm, same} \,\rangle$

©STEIN 2021

Concept Learning: Search in Hypothesis Space

Find-S Algorithm

See the example set D for the concept *EnjoySport*.



$h_0 = s_0 = \langle\, \perp, \perp, \perp, \perp, \perp, \perp \,\rangle$

$\mathbf{x}_1 = (\textit{sunny, warm, normal, strong, warm, same})$ $\quad h_1 = \langle\, \textit{sunny, warm, normal, strong, warm, same} \,\rangle$

$\mathbf{x}_2 = (\textit{sunny, warm, high, strong, warm, same})$ $\quad h_2 = \langle\, \textit{sunny, warm, ?, strong, warm, same} \,\rangle$

$\mathbf{x}_3 = (\textit{rainy, cold, high, strong, warm, change})$ $\quad h_3 = \langle\, \textit{sunny, warm, ?, strong, warm, same} \,\rangle$

$\mathbf{x}_4 = (\textit{sunny, warm, high, strong, cool, change})$ $\quad h_4 = \langle\, \textit{sunny, warm, ?, strong, ?, ?} \,\rangle$

©STEIN 2021

Concept Learning: Search in Hypothesis Space

Discussion of the Find-S Algorithm

1. Did we learn the only concept—or are there others?
2. Why should one pursuit the maximally specific hypothesis?
3. What if several maximally specific hypotheses exist?
4. Inconsistencies in the example set D remain undetected.
5. An inappropriate hypothesis structure or space H remains undetected.

©STEIN 2021

Concept Learning: Version Space

Definition 5 (Version Space)

The version space $V_{H,D}$ of a hypothesis space H and a example set D is comprised of all hypotheses $h(\mathbf{x}) \in H$ that are consistent with a set D of examples:

$$V_{H,D} = \{h(\mathbf{x}) \mid h(\mathbf{x}) \in H \ \wedge \ (\ \forall(\mathbf{x}, c(\mathbf{x})) \in D : \ h(\mathbf{x}) = c(\mathbf{x})\ )\ \}$$

©STEIN 2021

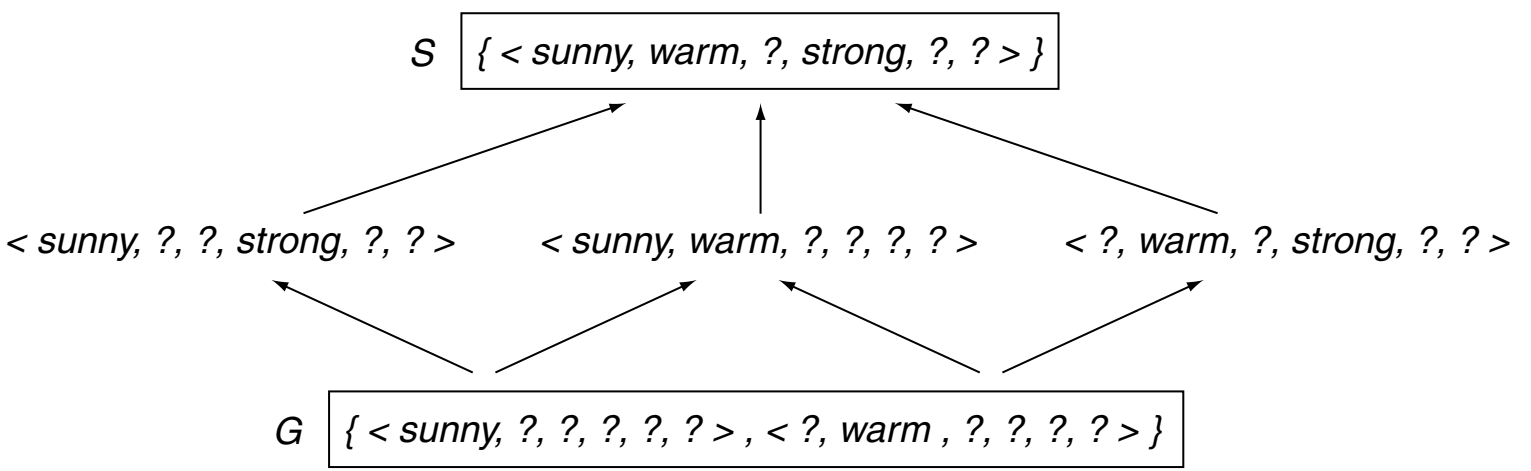
Concept Learning: Version Space

Definition 5 (Version Space)

The version space $V_{H,D}$ of a hypothesis space H and a example set D is comprised of all hypotheses $h(\mathbf{x}) \in H$ that are consistent with a set D of examples:

$$V_{H,D} = \{h(\mathbf{x}) \mid h(\mathbf{x}) \in H \ \wedge \ (\ \forall(\mathbf{x}, c(\mathbf{x})) \in D : \ h(\mathbf{x}) = c(\mathbf{x})\ )\ \}$$

Illustration of $V_{H,D}$ for the example set D:



©STEIN 2021

Remarks:

- The term “version space” reflects the fact that $V_{H,D}$ represents the set of all consistent versions of the target concept that are encoded in D.
- A naive approach for the construction of the version space is the following: (1) enumeration of all members of H, and, (2) elimination of those $h(\mathbf{x}) \in H$ for which $h(\mathbf{x}) \neq c(\mathbf{x})$ holds. This approach presumes a finite hypothesis space H and is feasible only for toy problems.

©STEIN 2021

Concept Learning: Version Space

Definition 6 (Boundary Sets of a Version Space)

Let H be hypothesis space and let D be set of examples. Then, based on the $\geq_g$-relation, the set of maximally general hypotheses, G, is defined as follows:

$$G = \{\ g(\mathbf{x}) \mid\ g(\mathbf{x}) \in H\ \wedge\ \textit{consistent}(g, D) \wedge \\ (\nexists g'(\mathbf{x}) : g'(\mathbf{x}) \in H\ \wedge\ g'(\mathbf{x}) >_g g(\mathbf{x})\ \wedge\ \textit{consistent}(g', D))\ \}$$

Similarly, the set of maximally specific (i.e., minimally general) hypotheses, S, is defined as follows:

$$S = \{s(\mathbf{x}) \mid\ s(\mathbf{x}) \in H\ \wedge\ \textit{consistent}(s, D) \wedge \\ (\nexists s'(\mathbf{x}) : s'(\mathbf{x}) \in H\ \wedge\ s(\mathbf{x}) >_g s'(\mathbf{x})\ \wedge\ \textit{consistent}(s', D))\ \}$$

©STEIN 2021

Concept Learning: Version Space

Theorem 7 (Version Space Representation)

Let X be a set of feature vectors and H a set of boolean-valued functions with domain X. Moreover, let $c(\mathbf{x}) : X \to \{0, 1\}$ be a target concept and D a set of examples of the form $(\mathbf{x}, c(\mathbf{x}))$.

Then, based on the $\geq_g$-relation, each member of the version space $V_{H,D}$ lies in between two members of G and S respectively:

$$V_{H,D} = \{h(\mathbf{x}) \mid \; h(\mathbf{x}) \in H \; \wedge \\ (\; \exists g(\mathbf{x}) \in G \;\; \exists s(\mathbf{x}) \in S : \; g(\mathbf{x}) \geq_g h(\mathbf{x}) \geq_g s(\mathbf{x}) \;) \; \}$$

©STEIN 2021

Concept Learning: Version Space

Candidate Elimination Algorithm [Mitchell 1997]

1. Initialization: $G = \{g_0\}, \; S = \{s_0\}$
2. If $\mathbf{x}$ is a positive example
   - Remove from G any hypothesis that is not consistent with $\mathbf{x}$
   - For each hypothesis s in S that is not consistent with $\mathbf{x}$
     - Remove s from S
     - Add to S all minimal generalizations h of s such that
       1. h is consistent with $\mathbf{x}$ and
       2. some member of G is more general than h
     - Remove from S any hypothesis that is less specific than another hypothesis in S

©STEIN 2021

Concept Learning: Version Space

Candidate Elimination Algorithm [Mitchell 1997]

1. Initialization: $G = \{g_0\}, \; S = \{s_0\}$
2. If $\mathbf{x}$ is a positive example
   - Remove from G any hypothesis that is not consistent with $\mathbf{x}$
   - For each hypothesis s in S that is not consistent with $\mathbf{x}$
     - Remove s from S
     - Add to S all minimal generalizations h of s such that
       1. h is consistent with $\mathbf{x}$ and
       2. some member of G is more general than h
     - Remove from S any hypothesis that is less specific than another hypothesis in S
3. If $\mathbf{x}$ is a negative example
   - Remove from S any hypothesis that is not consistent with $\mathbf{x}$
   - For each hypothesis g in G that is not consistent with $\mathbf{x}$
     - Remove g from G
     - Add to G all minimal specializations h of g such that
       1. h is consistent with $\mathbf{x}$ and
       2. some member of S is more specific than h
     - Remove from G any hypothesis that is less general than another hypothesis in G

©STEIN 2021

Remarks:

- The basic idea of Candidate Elimination is as follows:
  - A maximally specific hypothesis $s(\mathbf{x}) \in S$ restricts the positive examples in first instance. Hence, $s(\mathbf{x})$ needs to be relaxed (= generalized) with regard to each positive example that is not consistent with $s(\mathbf{x})$.
  - Conversely, a maximally general hypothesis $g(\mathbf{x}) \in G$ tolerates the negative examples in first instance. Hence, $g(\mathbf{x})$ needs to be constrained (= specialized) with regard to each negative example that is not consistent with $g(\mathbf{x})$.

©STEIN 2021

Concept Learning: Version Space

Candidate Elimination Algorithm (pseudo code)

```
1.  G = {g_0}  // G is the set of maximally general hypothesis in H.
    S = {s_0}  // S is the set of maximally specific hypothesis in H.

2.  FOREACH (x, c(x)) ∈ D DO
      IF c(x) = 1 THEN // x is a positive example.
        FOREACH g ∈ G DO IF g(x) ≠ 1 THEN G = G \ {g} ENDDO
        FOREACH s ∈ S DO
          IF s(x) ≠ 1 THEN
            S = S \ {s},  S^+ = min_generalizations(s, x)
            FOREACH s ∈ S^+ DO IF (∃g ∈ G : g ≥_g s) THEN S = S ∪ {s} ENDDO
            FOREACH s ∈ S DO IF (∃s' ∈ S : s' ≠ s ∧ s ≥_g s') THEN S = S \ {s} ENDDO
        ENDDO
      ELSE // x is a negative example.



      ENDIF
    ENDDO
3.  return(G, S)
```


©STEIN 2021

Concept Learning: Version Space

Candidate Elimination Algorithm (pseudo code)

```
1.  G = {g_0}  // G is the set of maximally general hypothesis in H.
    S = {s_0}  // S is the set of maximally specific hypothesis in H.

2.  FOREACH (x, c(x)) ∈ D DO
      IF c(x) = 1 THEN // x is a positive example.
        FOREACH g ∈ G DO IF g(x) ≠ 1 THEN G = G \ {g} ENDDO
        FOREACH s ∈ S DO
          IF s(x) ≠ 1 THEN
            S = S \ {s},  S^+ = min_generalizations(s, x)
            FOREACH s ∈ S^+ DO IF (∃g ∈ G : g ≥_g s) THEN S = S ∪ {s} ENDDO
            FOREACH s ∈ S DO IF (∃s' ∈ S : s' ≠ s ∧ s ≥_g s') THEN S = S \ {s} ENDDO
        ENDDO
      ELSE // x is a negative example.
        FOREACH s ∈ S DO IF s(x) ≠ 0 THEN S = S \ {s} ENDDO
        FOREACH g ∈ G DO
          IF g(x) ≠ 0 THEN
            G = G \ {g},  G^- = min_specializations(g, x)
            FOREACH g ∈ G^- DO IF (∃s ∈ S : g ≥_g s) THEN G = G ∪ {g} ENDDO
            FOREACH g ∈ G DO IF (∃g' ∈ G : g' ≠ g ∧ g' ≥_g g) THEN G = G \ {g} ENDDO
        ENDDO
      ENDIF
    ENDDO
3.  return(G, S)
```


©STEIN 2021

Concept Learning: Version Space

Candidate Elimination Algorithm (illustration)

$\boxed{\{ < \bot, \bot, \bot, \bot, \bot, \bot > \}}$ S_0

$\boxed{\{ < ?, ?, ?, ?, ?, ? > \}}$ G_0,

©STEIN 2021

Concept Learning: Version Space

Candidate Elimination Algorithm (illustration)

$\{ < \perp, \perp, \perp, \perp, \perp, \perp > \}$ S_0

$\{ <$ *sunny, warm, normal, strong, warm, same* $> \}$ S_1

$\{ < ?, ?, ?, ?, ?, ? > \}$ $G_0, G_1,$

$\mathbf{x}_1 = ($*sunny, warm, normal, strong, warm, same*$)$ *EnjoySport*$(\mathbf{x}_1) = 1$

©STEIN 2021

Concept Learning: Version Space

Candidate Elimination Algorithm (illustration)

$\{ < \perp, \perp, \perp, \perp, \perp, \perp > \}$ S_0

↓

$\{ <$ *sunny, warm, normal, strong, warm, same* $> \}$ S_1

↓

$\boxed{\{ <$ *sunny, warm, ?, strong, warm, same* $> \}}$ S_2,

$\boxed{\{ < ?, ?, ?, ?, ?, ? > \}}$ G_0, G_1, G_2

$\mathbf{x}_1 = ($*sunny, warm, normal, strong, warm, same*$)$ $\quad$ *EnjoySport*$(\mathbf{x}_1) = 1$

$\mathbf{x}_2 = ($*sunny, warm, high, strong, warm, same*$)$ $\quad$ *EnjoySport*$(\mathbf{x}_2) = 1$

©STEIN 2021

Concept Learning: Version Space

Candidate Elimination Algorithm (illustration)

$\{ < \bot, \bot, \bot, \bot, \bot, \bot > \}$ S_0

↓

$\{ <$ *sunny, warm, normal, strong, warm, same* $> \}$ S_1

↓

$\{ <$ *sunny, warm, ?, strong, warm, same* $> \}$ S_2, S_3

$\{ <$ *sunny, ?, ?, ?, ?, ?* $>, <$ *?, warm, ?, ?, ?, ?* $>, <$ *?, ?, ?, ?, ?, same* $> \}$ G_3

↑

$\{ <$ *?, ?, ?, ?, ?, ?* $> \}$ G_0, G_1, G_2

$\mathbf{x}_1 = ($*sunny, warm, normal, strong, warm, same*$)$ $\quad$ *EnjoySport*$(\mathbf{x}_1) = 1$

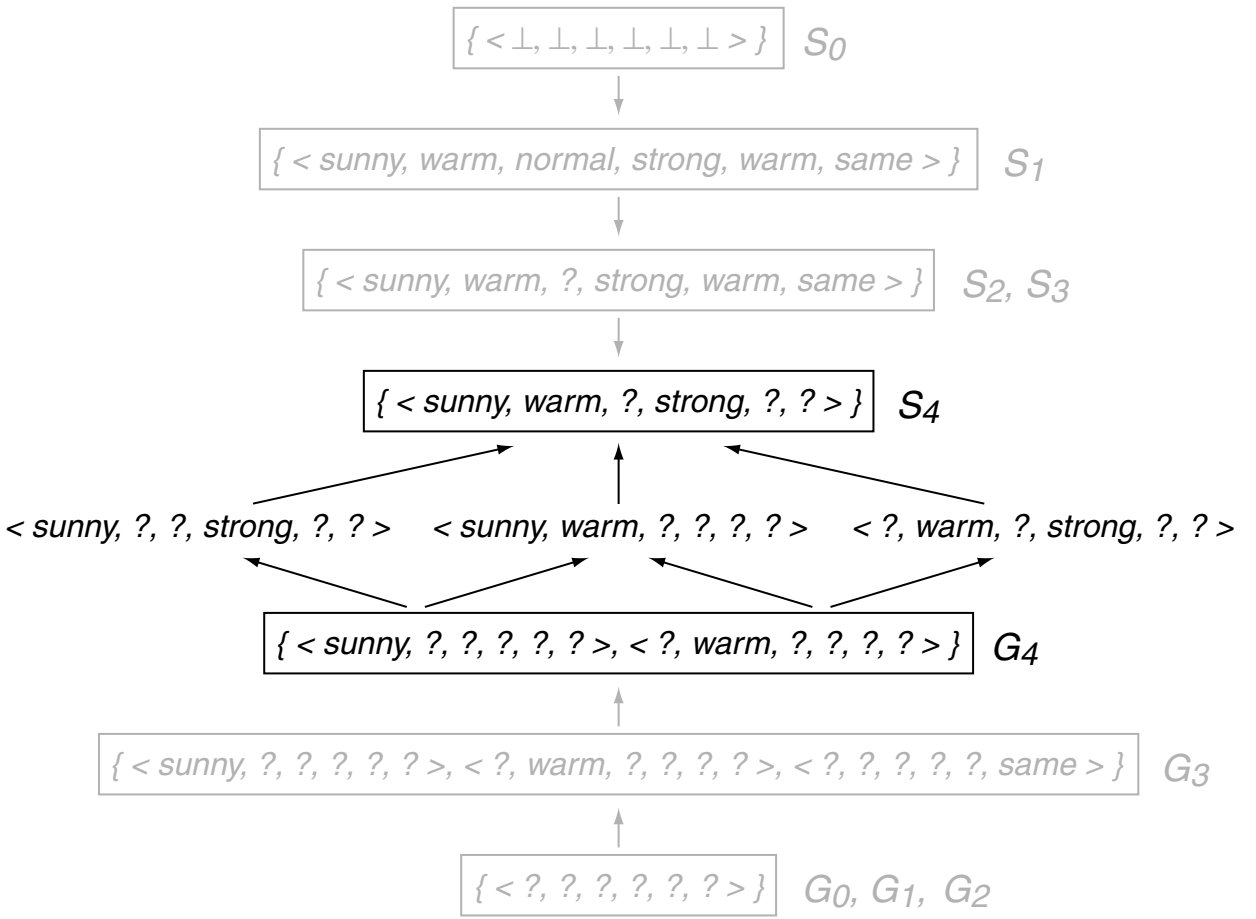
$\mathbf{x}_2 = ($*sunny, warm, high, strong, warm, same*$)$ $\quad$ *EnjoySport*$(\mathbf{x}_2) = 1$

$\mathbf{x}_3 = ($*rainy, cold, high, strong, warm, change*$)$ $\quad$ *EnjoySport*$(\mathbf{x}_3) = 0$ $\quad$ [feature domains] [algorithm]

©STEIN 2021

Concept Learning: Version Space

Candidate Elimination Algorithm (illustration)



$\mathbf{x}_1 = (\textit{sunny, warm, normal, strong, warm, same})$ $\quad$ $\textit{EnjoySport}(\mathbf{x}_1) = 1$

$\mathbf{x}_2 = (\textit{sunny, warm, high, strong, warm, same})$ $\quad$ $\textit{EnjoySport}(\mathbf{x}_2) = 1$

$\mathbf{x}_3 = (\textit{rainy, cold, high, strong, warm, change})$ $\quad$ $\textit{EnjoySport}(\mathbf{x}_3) = 0$ $\quad$ [feature domains] [algorithm]

$\mathbf{x}_4 = (\textit{sunny, warm, high, strong, cool, change})$ $\quad$ $\textit{EnjoySport}(\mathbf{x}_4) = 1$

©STEIN 2021

Concept Learning: Version Space

Discussion of the Candidate Elimination Algorithm

1. What about selecting examples from D according to a certain strategy?
   Keyword: active learning

2. What are partially learned concepts and how to exploit them?
   Keyword: ensemble classification

3. The version space as defined here is “biased”. What does this mean?

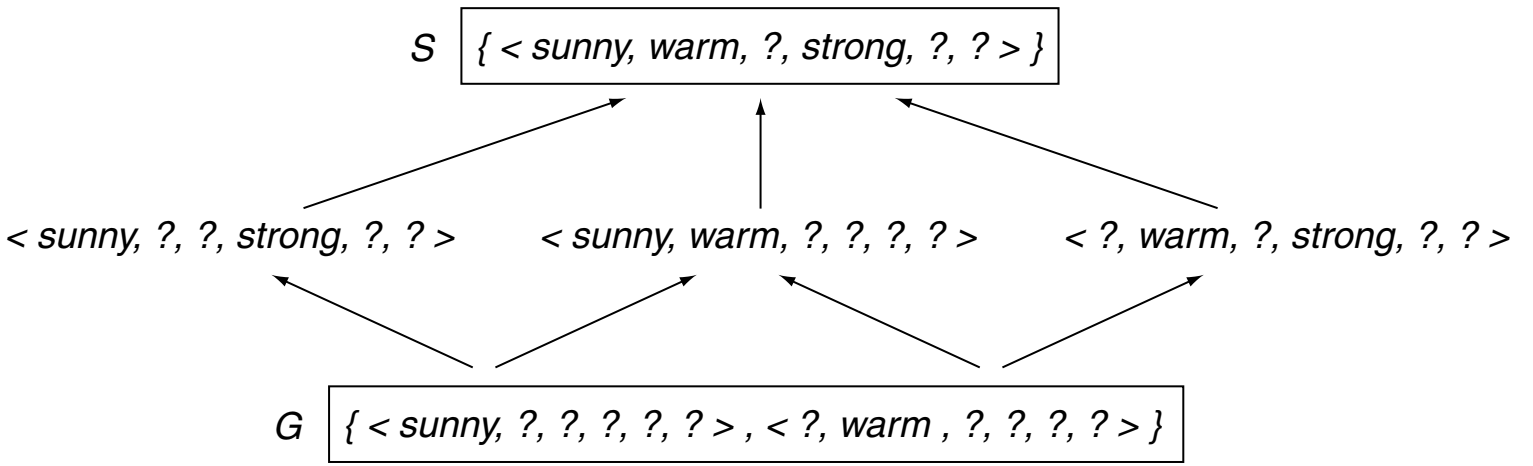
4. Will Candidate Elimination converge towards the correct hypothesis?

5. When does one end up with an empty version space?

©STEIN 2021

Concept Learning: Version Space

Question 1: Selecting Examples from D



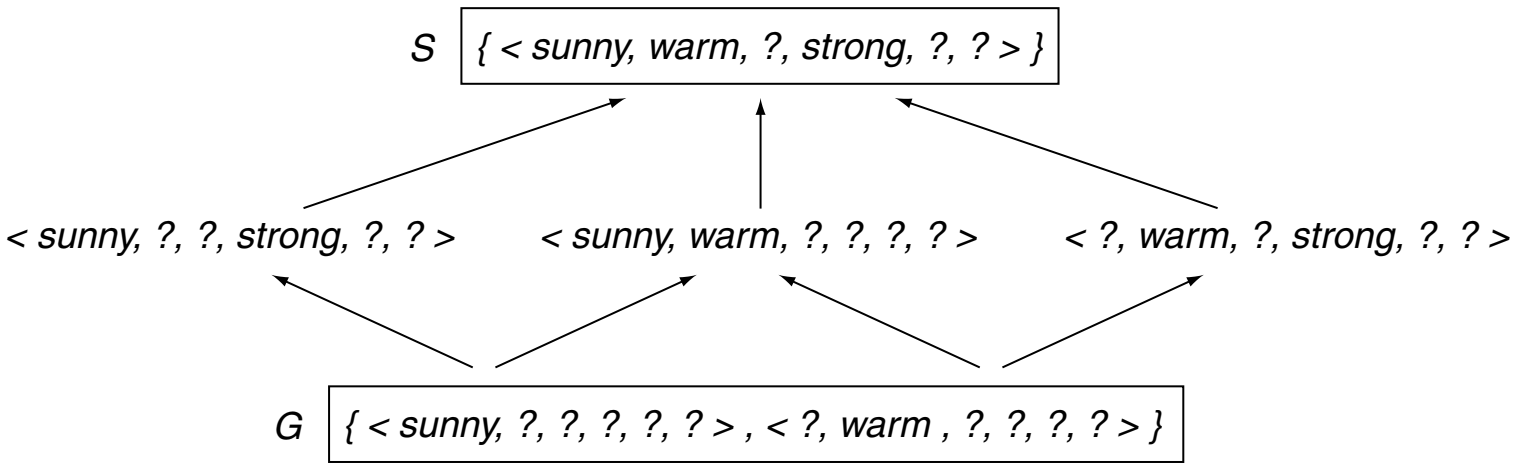
An example from which we can “maximally” learn:

$\mathbf{x}_7 = (\textit{sunny, warm, normal, light, warm, same})$

©STEIN 2021

Concept Learning: Version Space

Question 1: Selecting Examples from D



An example from which we can "maximally" learn:

$\mathbf{x}_7 = (\textit{sunny, warm, normal, light, warm, same})$

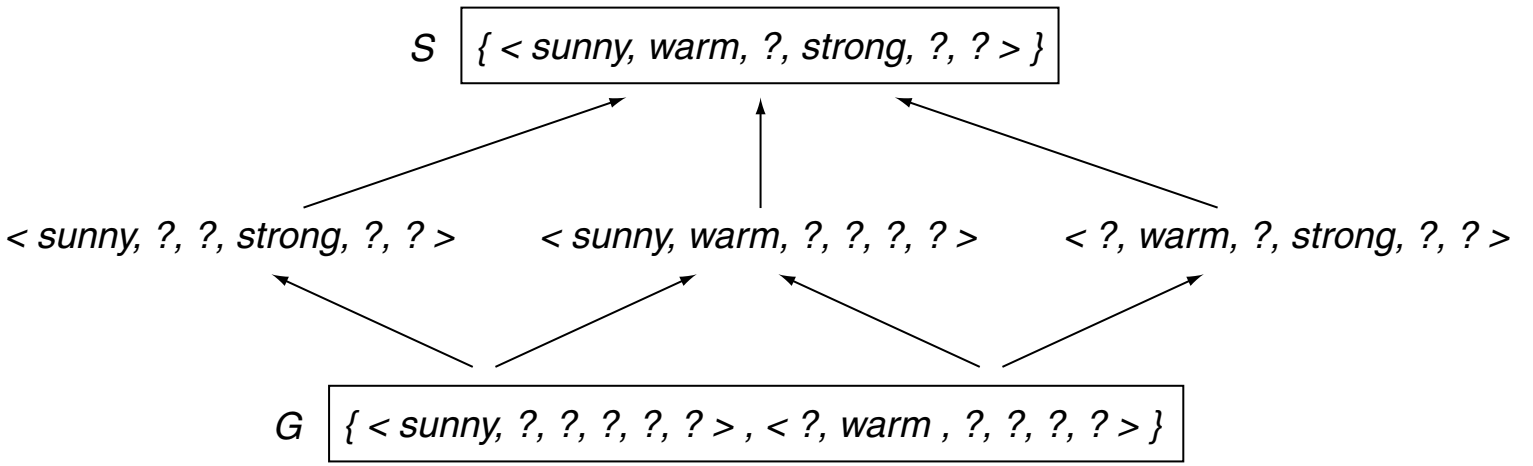
I.e., irrespective the value of $c(\mathbf{x}_7)$, the example $(\mathbf{x}_7, c(\mathbf{x}_7))$ will be consistent with three of the six hypotheses. It follows:

- If *EnjoySport*$(\mathbf{x}_7) = 1$ S can be further generalized.
- If *EnjoySport*$(\mathbf{x}_7) = 0$ G can be further specialized.

©STEIN 2021

Concept Learning: Version Space

Question 2: Partially Learned Concepts



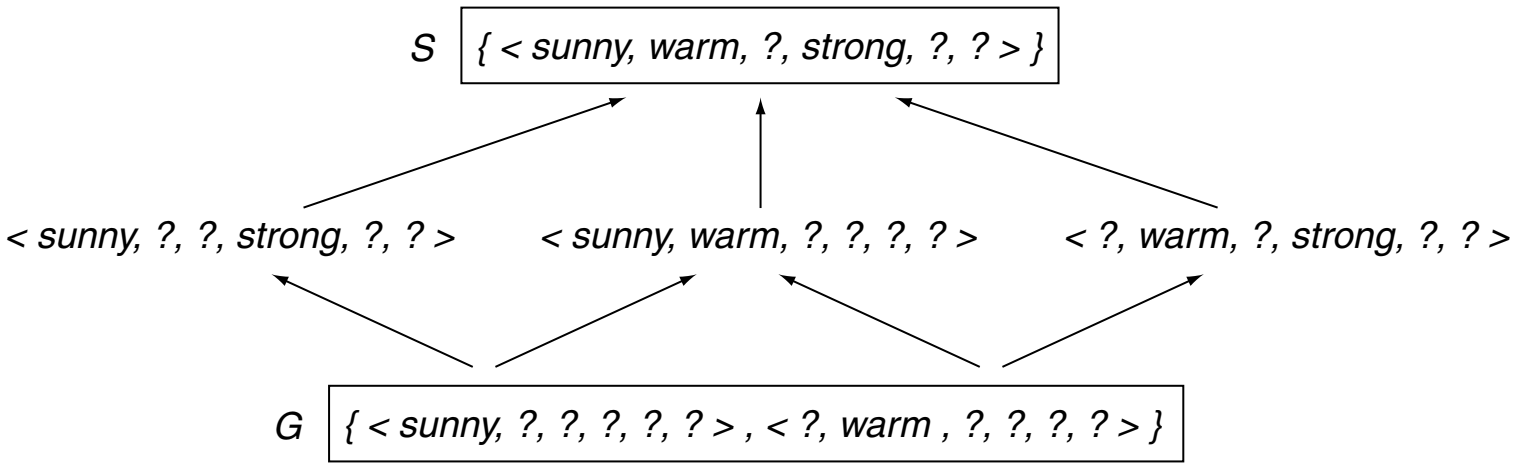
Combine the six classifiers in the version space:

Example	Sky	Temperature	Humidity	Wind	Water	Forecast	EnjoySport
5	sunny	warm	normal	strong	cool	change	
6	rainy	cold	normal	light	warm	same	
7	sunny	warm	normal	light	warm	same	
8	sunny	cold	normal	strong	warm	same	

©STEIN 2021

Concept Learning: Version Space

Question 2: Partially Learned Concepts



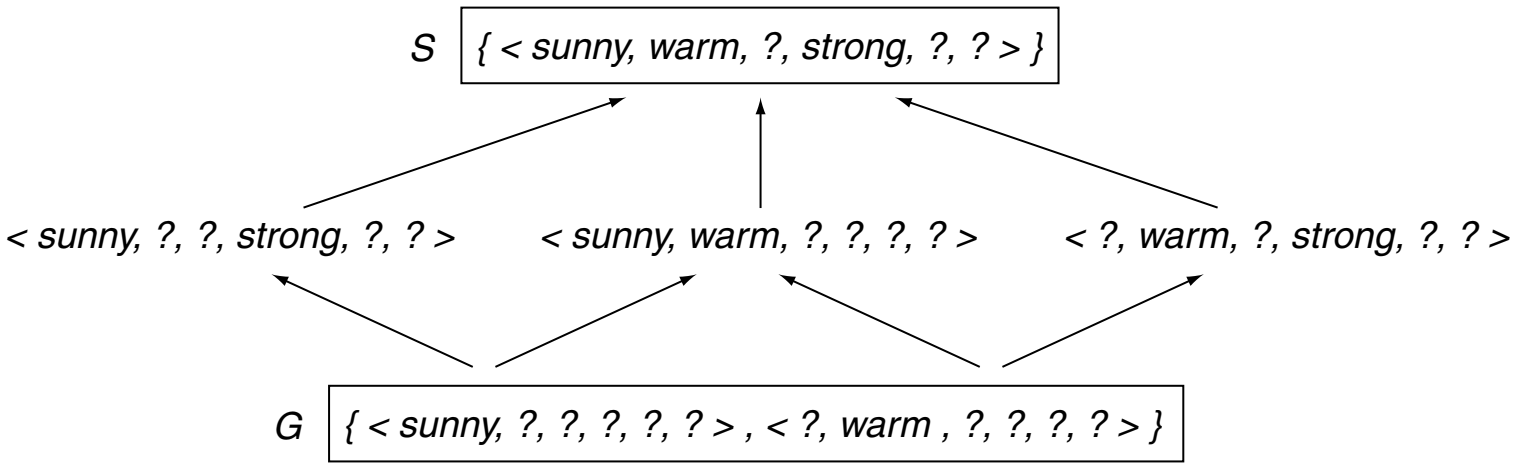
Combine the six classifiers in the version space:

Example	Sky	Temperature	Humidity	Wind	Water	Forecast	EnjoySport
5	sunny	warm	normal	strong	cool	change	6+ : 0–
6	rainy	cold	normal	light	warm	same	
7	sunny	warm	normal	light	warm	same	
8	sunny	cold	normal	strong	warm	same	

©STEIN 2021

Concept Learning: Version Space

Question 2: Partially Learned Concepts



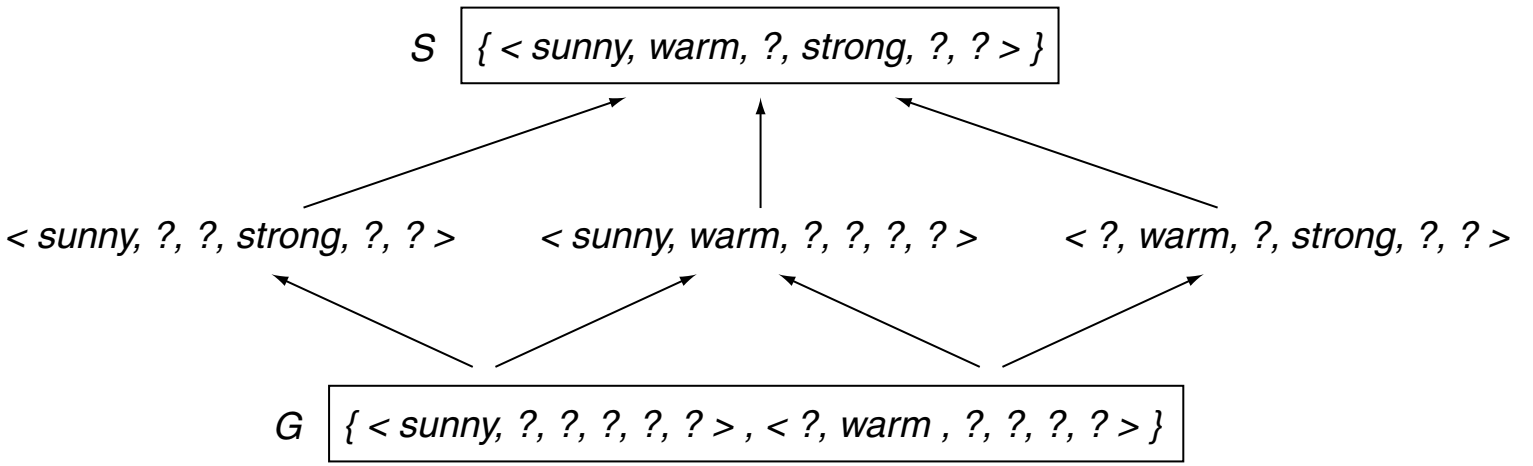
Combine the six classifiers in the version space:

Example	Sky	Temperature	Humidity	Wind	Water	Forecast	EnjoySport
5	sunny	warm	normal	strong	cool	change	6+ : 0–
6	rainy	cold	normal	light	warm	same	0+ : 6–
7	sunny	warm	normal	light	warm	same	
8	sunny	cold	normal	strong	warm	same	

©STEIN 2021

Concept Learning: Version Space

Question 2: Partially Learned Concepts



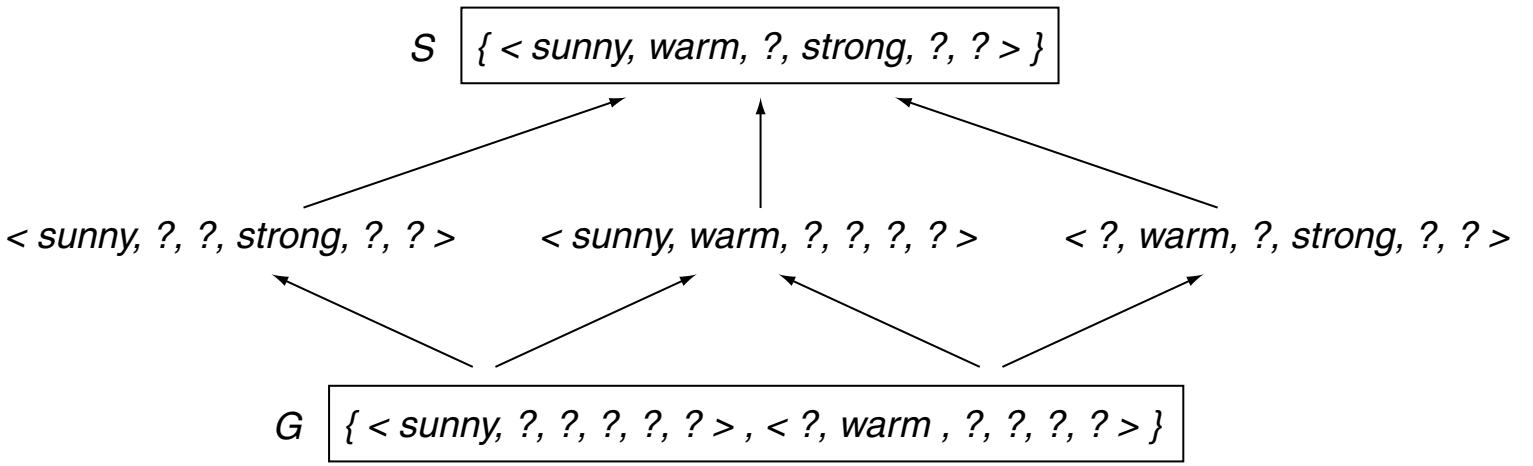
Combine the six classifiers in the version space:

Example	Sky	Temperature	Humidity	Wind	Water	Forecast	EnjoySport
5	sunny	warm	normal	strong	cool	change	6+ : 0–
6	rainy	cold	normal	light	warm	same	0+ : 6–
7	sunny	warm	normal	light	warm	same	3+ : 3–
8	sunny	cold	normal	strong	warm	same	

©STEIN 2021

Concept Learning: Version Space

Question 2: Partially Learned Concepts



Combine the six classifiers in the version space:

Example	Sky	Temperature	Humidity	Wind	Water	Forecast	EnjoySport
5	sunny	warm	normal	strong	cool	change	6+ : 0–
6	rainy	cold	normal	light	warm	same	0+ : 6–
7	sunny	warm	normal	light	warm	same	3+ : 3–
8	sunny	cold	normal	strong	warm	same	2+ : 4–

©STEIN 2021

Concept Learning: Version Space

Question 3: Inductive Bias

A new set of training examples D :

Example	Sky	Temperature	Humidity	Wind	Water	Forecast	EnjoySport
9	sunny	warm	normal	strong	cool	change	yes
10	cloudy	warm	normal	strong	cool	change	yes

➜ $S = \{ \langle$ *?, warm, normal, strong, cool, change* $\rangle \}$

©STEIN 2021

Concept Learning: Version Space

Question 3: Inductive Bias

A new set of training examples D :

Example	Sky	Temperature	Humidity	Wind	Water	Forecast	EnjoySport
9	sunny	warm	normal	strong	cool	change	yes
10	cloudy	warm	normal	strong	cool	change	yes

$\rightarrow \quad S = \{ \langle$ *?, warm, normal, strong, cool, change* $\rangle \}$

$\vdots$

11	rainy	warm	normal	strong	cool	change	no

$\rightarrow \quad S = \{ \}$

Discussion:

- What assumptions about the target concept are met by the learner a-priori?

©STEIN 2021

Concept Learning: Version Space

Question 3: Inductive Bias

A new set of training examples D :

Example	Sky	Temperature	Humidity	Wind	Water	Forecast	EnjoySport
9	sunny	warm	normal	strong	cool	change	yes
10	cloudy	warm	normal	strong	cool	change	yes

➜ $S = \{ \langle$ *?, warm, normal, strong, cool, change* $\rangle \}$

⋮

11	rainy	warm	normal	strong	cool	change	no

➜ $S = \{ \}$

Discussion:

- What assumptions about the target concept are met by the learner a-priori?

➜ Consequence: The hypothesis space H may be designed to contain more complex concepts, e.g., $\langle$ *sunny, ?, ?, ?, ?, ?* $\rangle \vee \langle$ *cloudy, ?, ?, ?, ?, ?* $\rangle$.

©STEIN 2021

Concept Learning: Version Space

Question 3: Inductive Bias (continued)

- In a binary classification problem the unrestricted (= unbiased) hypothesis space contains $|\mathcal{P}(X)| \equiv 2^{|X|}$ elements.
- A learning algorithm that considers all possible hypotheses as equally likely makes no a-priori assumption with regard to the target concept.
- A learning algorithm without a-priori assumptions has no "inductive bias".

> *"The policy by which a* [learning] *algorithm generalizes from observed training examples to classify unseen instances is its inductive bias.* [. . .] *Inductive bias is the set of assumptions that, together with the training data, deductively justify the classification by the learner to future instances."* [p.63, Mitchell 1997]

©STEIN 2021

Concept Learning: Version Space

Question 3: Inductive Bias (continued)

- In a binary classification problem the unrestricted (= unbiased) hypothesis space contains $|\mathcal{P}(X)| \equiv 2^{|X|}$ elements.
- A learning algorithm that considers all possible hypotheses as equally likely makes no a-priori assumption with regard to the target concept.
- A learning algorithm without a-priori assumptions has no "inductive bias".

> *"The policy by which a* [learning] *algorithm generalizes from observed training examples to classify unseen instances is its inductive bias.* [. . .] *Inductive bias is the set of assumptions that, together with the training data, deductively justify the classification by the learner to future instances."* [p.63, Mitchell 1997]

➜ A learning algorithm without inductive bias has no directive to classify unseen examples. Put another way: the learner cannot *generalize*.

➜ A learning algorithm without inductive bias can only *memorize*.

Which algorithm (Find-S, Candidate Elimination) has a stronger inductive bias?

©STEIN 2021

Chapter ML:II (continued)

II. Machine Learning Basics

- Linear Regression
- Concept Learning: Search in Hypothesis Space
- Concept Learning: Version Space
- Evaluating Effectiveness

©STEIN/LETTMANN 2021

Evaluating Effectiveness

Misclassification Rate

Definition 8 (True Misclassification Rate, True Error)

Let X be a set of feature vectors with a finite number of elements. Moreover, let C be a set of classes, let $y : X \to C$ be a classifier, and let $c(\mathbf{x})$ be the target concept to be learned. Then the true misclassification rate of $y(\mathbf{x})$, denoted by $Err^*(y)$, is defined as follows:

$$Err^*(y) = \frac{|\{\mathbf{x} \in X : y(\mathbf{x}) \neq c(\mathbf{x})\}|}{|X|}$$

©STEIN/LETTMANN 2021

Evaluating Effectiveness

Misclassification Rate

Definition 8 (True Misclassification Rate, True Error)

Let X be a set of feature vectors with a finite number of elements. Moreover, let C be a set of classes, let $y : X \to C$ be a classifier, and let $c(\mathbf{x})$ be the target concept to be learned. Then the true misclassification rate of $y(\mathbf{x})$, denoted by *Err*$^*(y)$, is defined as follows:

$$\mathit{Err}^*(y) = \frac{|\{\mathbf{x} \in X : y(\mathbf{x}) \neq c(\mathbf{x})\}|}{|X|}$$

Problem:

- Usually the *total function* $c(\mathbf{x})$ is unknown and hence *Err*$^*(y)$ is unknown.

Solution:

- Based on a set of examples D, estimation of upper and lower bounds for *Err*$^*(y)$ according to some sampling strategy.

©STEIN/LETTMANN 2021

Remarks:

- Instead of the term “true misclassification rate” we will also use the term “true misclassification error” or simply “true error”.
- The unknown function $c(\mathbf{x})$ returns for a feature vector $\mathbf{x}$ its true class $c \in C$. I.e., “$c(\mathbf{x})$” and “c” designate the same, namely, the true class of $\mathbf{x}$ (if the $\mathbf{x}$ associated with c is clear).
- The English word “rate” can denote both the mathematical concept of a *flow quantity* (a change of a quantity per time unit) as well as the mathematical concept of a *portion*, a *percentage*, or a *ratio*, which has a stationary (= time-independent) semantics. Note that the latter semantics is meant here when talking about the misclassification rate.
- The German word „Rate“ is often (mis)used to denote the mathematical concept of a portion, a percentage, or a ratio. Taking a precise mathematical standpoint, the correct German words are „Anteil“ or „Quote“. I.e., a semantically correct translation of misclassification rate is „Missklassifikationsanteil“, and not „Missklassifikationsrate“.

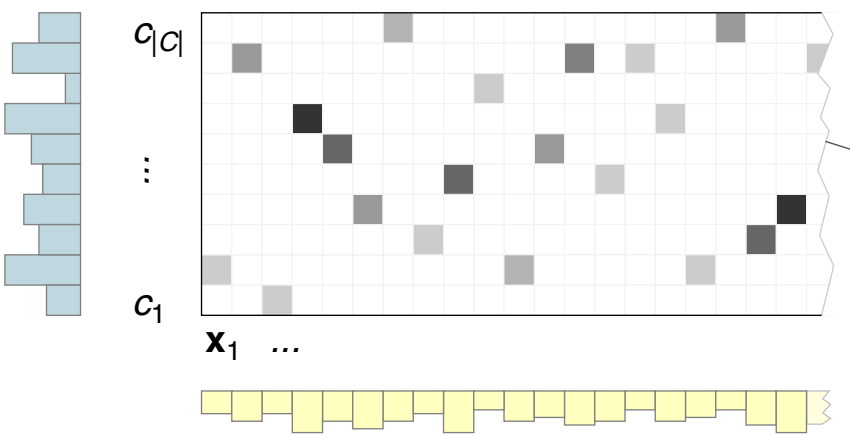
©STEIN/LETTMANN 2021

Evaluating Effectiveness

Misclassification Rate: Probabilistic Foundation

Let $\mathbf{X}$ be a feature space with a finite number of elements, C a set of classes, and P a probability measure on $\mathbf{X} \times C$. Then $p(\mathbf{x}, c)$, $p(\mathbf{x}, c) = P(\mathcal{X}{=}\mathbf{x}, \mathcal{C}{=}c)$, denotes the probability of the joint event (1) to observe the vector $\mathbf{x} \in \mathbf{X}$ and (2) that $\mathbf{x}$ belongs to class $c \in C$.

Illustration of P over $\mathbf{X} \times C$, $P \in [0; 1]$ (shading indicates magnitude) :



Joint probability $p(\mathbf{x}, c)$

(no label noise → classes unique)

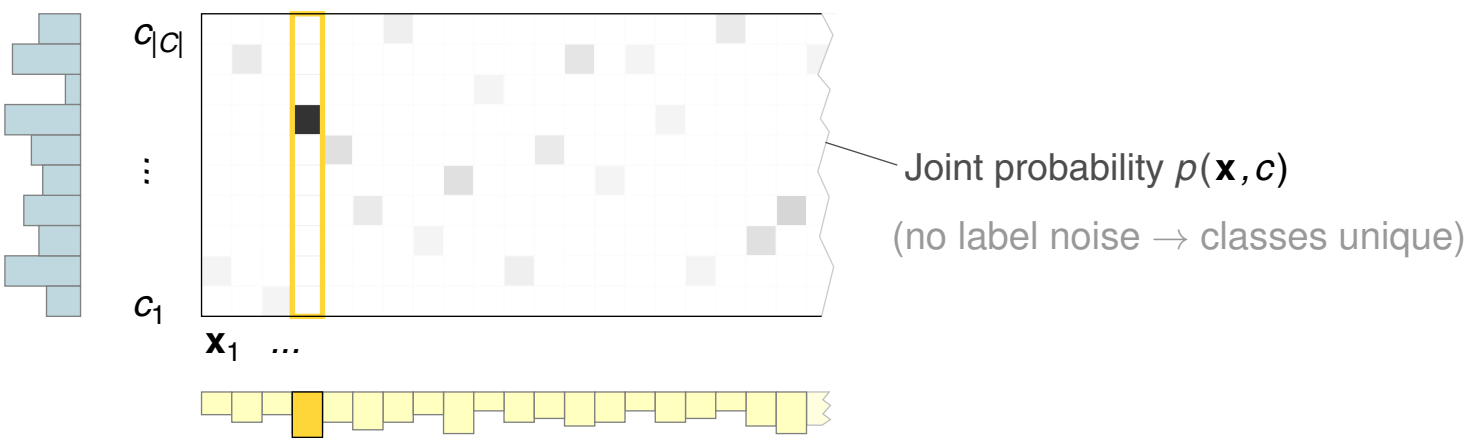
©STEIN/LETTMANN 2021

Evaluating Effectiveness

Misclassification Rate: Probabilistic Foundation (continued)

Let $\mathbf{X}$ be a feature space with a finite number of elements, C a set of classes, and P a probability measure on $\mathbf{X} \times C$. Then $p(\mathbf{x}, c)$, $p(\mathbf{x}, c) = P(\mathcal{X}{=}\mathbf{x}, \mathcal{C}{=}c)$, denotes the probability of the joint event (1) to observe the vector $\mathbf{x} \in \mathbf{X}$ and (2) that $\mathbf{x}$ belongs to class $c \in C$.

Illustration of P over $\mathbf{X} \times C$, $P \in [0; 1]$ (shading indicates magnitude) :



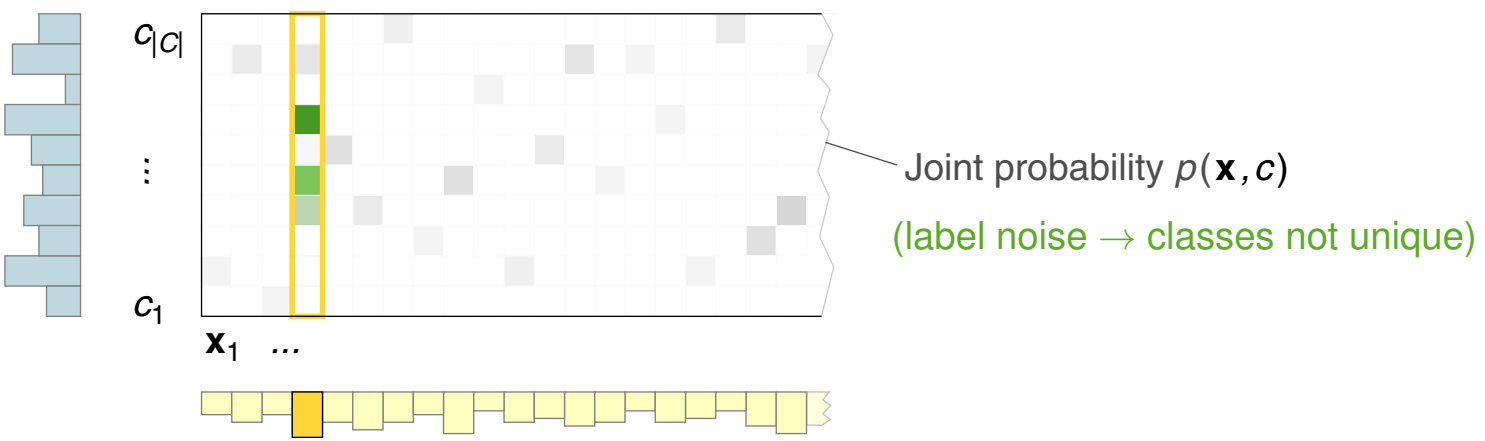
©STEIN/LETTMANN 2021

Evaluating Effectiveness

Misclassification Rate: Probabilistic Foundation (continued)

Let $\mathbf{X}$ be a feature space with a finite number of elements, C a set of classes, and P a probability measure on $\mathbf{X} \times C$. Then $p(\mathbf{x}, c)$, $p(\mathbf{x}, c) = P(\mathcal{X}{=}\mathbf{x}, \mathcal{C}{=}c)$, denotes the probability of the joint event (1) to observe the vector $\mathbf{x} \in \mathbf{X}$ and (2) that $\mathbf{x}$ belongs to class $c \in C$.

Illustration of P over $\mathbf{X} \times C$, $P \in [0; 1]$ (shading indicates magnitude) :



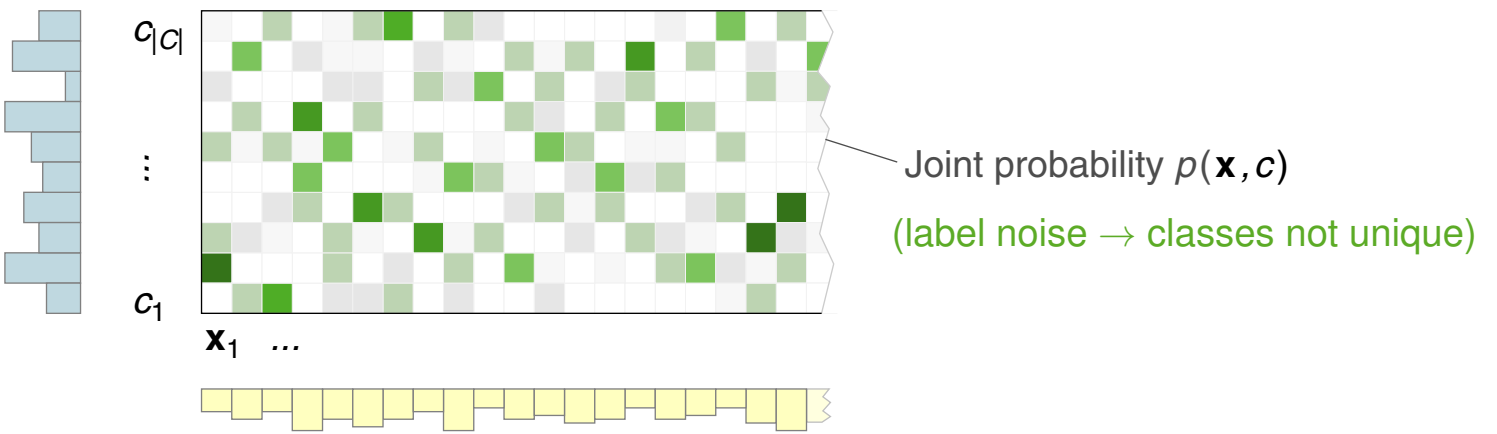
©STEIN/LETTMANN 2021

Evaluating Effectiveness

Misclassification Rate: Probabilistic Foundation (continued)

Let $\mathbf{X}$ be a feature space with a finite number of elements, C a set of classes, and P a probability measure on $\mathbf{X} \times C$. Then $p(\mathbf{x}, c)$, $p(\mathbf{x}, c) = P(\mathcal{X}{=}\mathbf{x}, \mathcal{C}{=}c)$, denotes the probability of the joint event (1) to observe the vector $\mathbf{x} \in \mathbf{X}$ and (2) that $\mathbf{x}$ belongs to class $c \in C$.

Illustration of P over $\mathbf{X} \times C$, $P \in [0; 1]$ (shading indicates magnitude) :



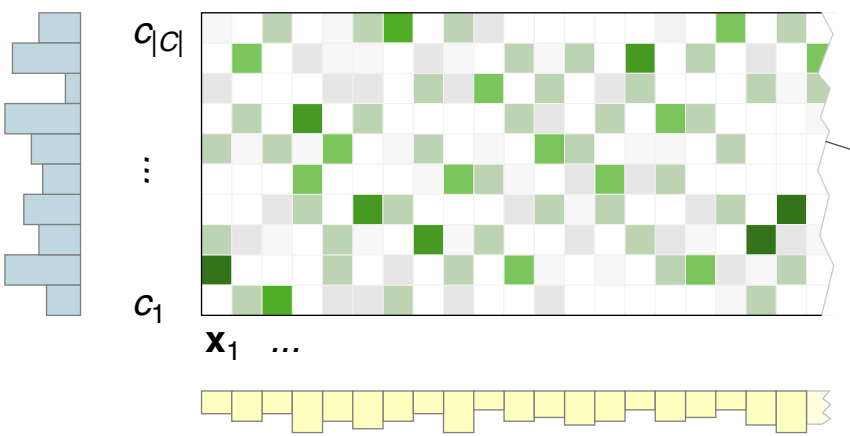
©STEIN/LETTMANN 2021

Evaluating Effectiveness

Misclassification Rate: Probabilistic Foundation (continued)

Let $\mathbf{X}$ be a feature space with a finite number of elements, C a set of classes, and P a probability measure on $\mathbf{X} \times C$. Then $p(\mathbf{x}, c)$, $p(\mathbf{x}, c) = P(\mathcal{X}{=}\mathbf{x}, \mathcal{C}{=}c)$, denotes the probability of the joint event (1) to observe the vector $\mathbf{x} \in \mathbf{X}$ and (2) that $\mathbf{x}$ belongs to class $c \in C$.

Illustration of P over $\mathbf{X} \times C$, $P \in [0; 1]$ (shading indicates magnitude) :



Joint probability $p(\mathbf{x}, c)$

(label noise → classes not unique)

$$\mathit{Err}^*(y) = \sum_{\mathbf{x} \in X} \sum_{c \in C} p(\mathbf{x}, c) \cdot I(y(\mathbf{x}), c), \quad \text{with } I(y(\mathbf{x}), c) = \begin{cases} 0 & \text{if } y(\mathbf{x}) = c \\ 1 & \text{otherwise} \end{cases}$$

©STEIN/LETTMANN 2021

Remarks:

- Recall from Section Specification of Learning Tasks in part Introduction the difference between the following concepts, denoted by glyph variants of the same letter:
  x (single feature), $\mathbf{x}$ (feature vector), X (set of feature vectors), $\mathbf{X}$ (feature space), $\mathcal{X}$ (random variable whose realizations are feature vectors).

- $\mathcal{X}$ and $\mathcal{C}$ denote random variables with domains $\mathbf{X}$ and C respectively. $\mathbf{X}$ may not be restricted to a finite set, giving rise to probability density functions (with continuous random variables) in the place of the probability mass functions (with discrete random variables). The illustrations in a continuous setting remain basically unchanged, presupposing a sensible discretization of the feature space $\mathbf{X}$. [continuous setting]

- $\mathcal{X}$ models the fact that each observation procedure is governed by a probability distribution, rendering certain observations more likely than others. Keyword: prior probability of [observing some] $\mathbf{x}$.
  Note that the (multi)set X of feature vectors in the definition of the true misclassification rate is governed by the distribution of $\mathcal{X}$: Objects o that are more likely will induce the respective multiplicity of feature vectors $\mathbf{x}$ in X and hence are considered with the appropriate weight in $Err^*(y)$.

- $\mathcal{C}$ models the fact that in the real world the classification $c(\mathbf{x})$ of a feature vector $\mathbf{x}$ may not be deterministic. Reasons for a probabilistic class assignment include: an incomplete feature set, imprecision and random errors within the feature measuring process, lack of care during data acquisition. Keyword: label noise.

©STEIN/LETTMANN 2021

Remarks: (continued)

- Since P is a probability measure (see section Probability Basics in part Statistical Learning), its argument is an event. Examples for events are "$\mathcal{X}{=}\mathbf{x}$", "$\mathcal{X}{=}\mathbf{x},\ \mathcal{C}{=}c$", or "$\mathcal{X}{=}\mathbf{x} \mid \mathcal{C}{=}c$".

- $p(\mathbf{x}, c)$, $p(\mathbf{x})$, $p(\mathbf{x} \mid c)$, etc. denote probabilty mass functions. A probabilty mass function, pmf, is defined for discrete random variables only. Its argument(s) is a (are) realization(s) of the respective random variable(s). The counterpart of $p()$ for continuous random variables is called probability density function, pdf, and is typically denoted by $f()$.

- Since $p(\mathbf{x}, c)$ is defined as $P(\mathcal{X}{=}\mathbf{x}, \mathcal{C}{=}c)$ the two expressions can be used interchangeably in most places.

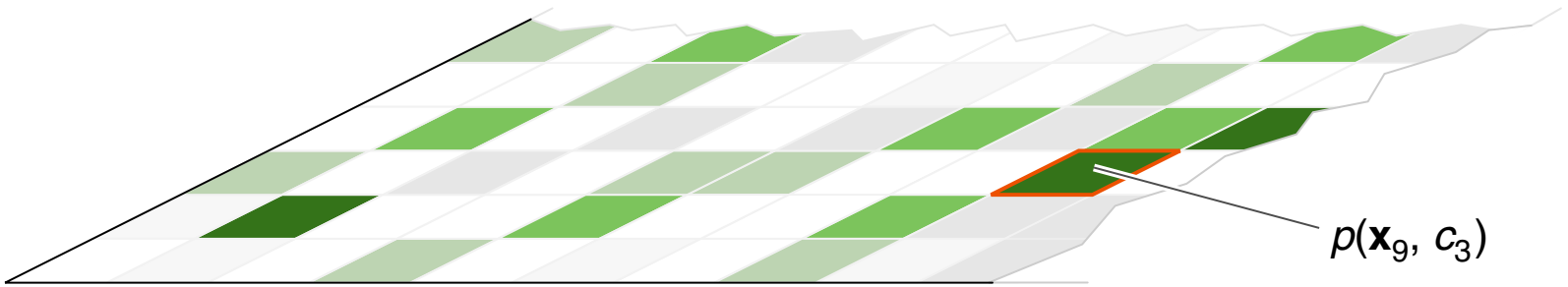
- Let A and B denote two events, e.g., $A =$ "$\mathcal{X}{=}\mathbf{x}_9$" and $B =$ "$\mathcal{C}{=}c_3$". Then the following expressions are equivalent notations for the probability of the joint event: $P(A, B)$, $P(A \text{ and } B)$, $P(A \wedge B)$.

©STEIN/LETTMANN 2021

Evaluating Effectiveness

Illustration 1: Generic Feature and Class Distributions [Wikipedia]

Joint probability $p(\mathbf{x}, c) = P(\mathcal{X}{=}\mathbf{x}, \mathcal{C}{=}c)$:

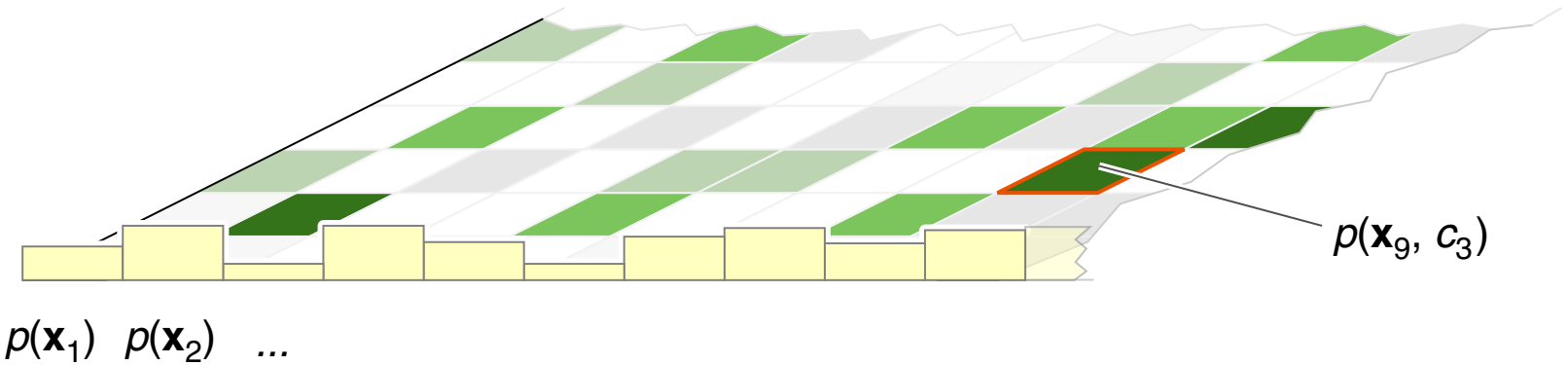


©STEIN/LETTMANN 2021

Evaluating Effectiveness

Illustration 1: Generic Feature and Class Distributions [Wikipedia] (continued)

Marginal probability $p(\mathbf{x}) = P(\mathcal{X}{=}\mathbf{x})$:

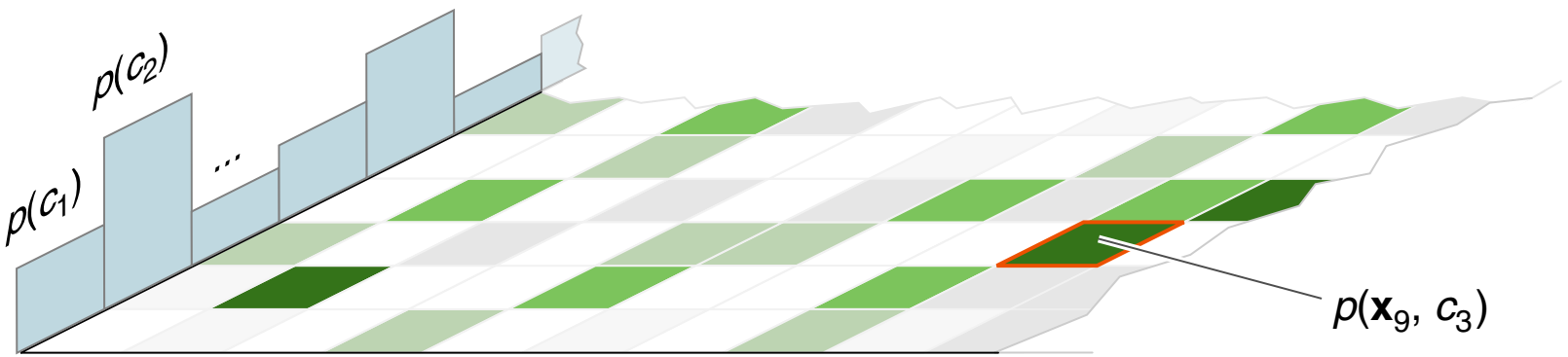


© STEIN/LETTMANN 2021

Evaluating Effectiveness

Illustration 1: Generic Feature and Class Distributions [Wikipedia] (continued)

Marginal probability $p(c) = P(\mathcal{C}{=}c)$:

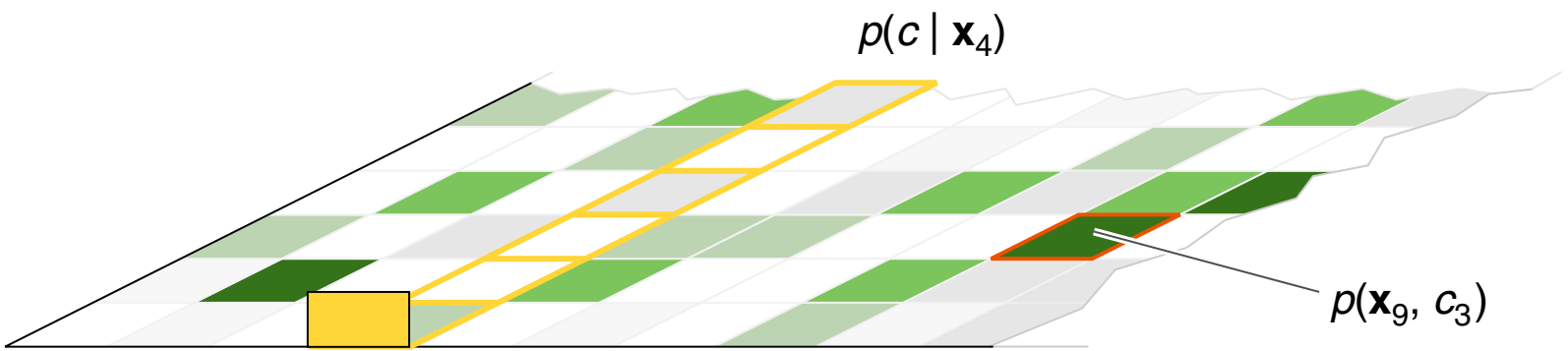


© STEIN/LETTMANN 2021

Evaluating Effectiveness

Illustration 1: Generic Feature and Class Distributions [Wikipedia] (continued)

Distribution of the classes c under feature vector $\mathbf{x}_4$,
denoted by $p(c \mid \mathbf{x}_4) = P(\mathcal{C}{=}c \mid \mathcal{X}{=}\mathbf{x}_4) \;\equiv\; P_{\mathcal{X}=\mathbf{x}_4}(\mathcal{C}{=}c)$:

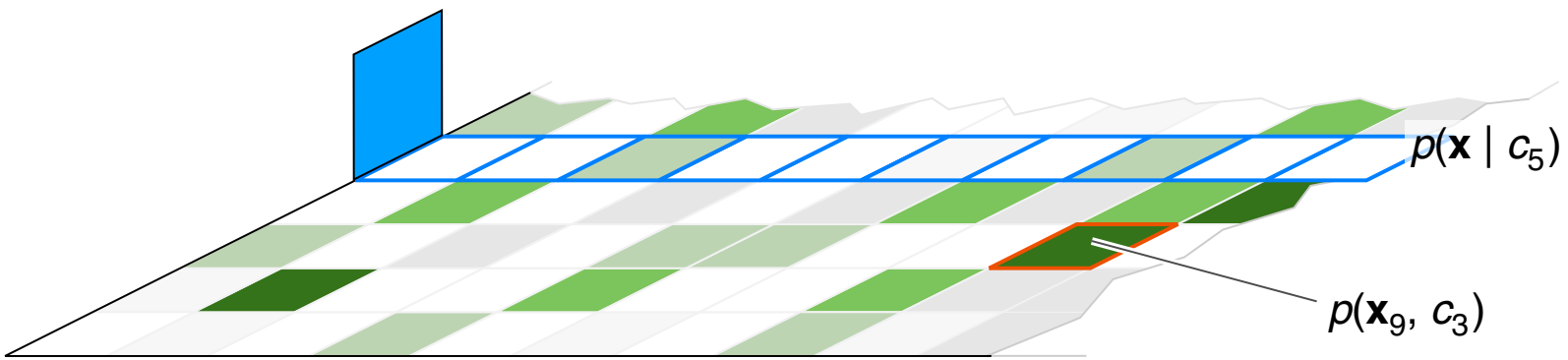


© STEIN/LETTMANN 2021

Evaluating Effectiveness

Illustration 1: Generic Feature and Class Distributions [Wikipedia] (continued)

Distribution of the feature vectors $\mathbf{x}$ und class c_5,
denoted by $p(\mathbf{x} \mid c_5) = P(\mathcal{X}{=}\mathbf{x} \mid \mathcal{C}{=}c_5) \;\equiv\; P_{\mathcal{C}=c_5}(\mathcal{X}{=}\mathbf{x})$:

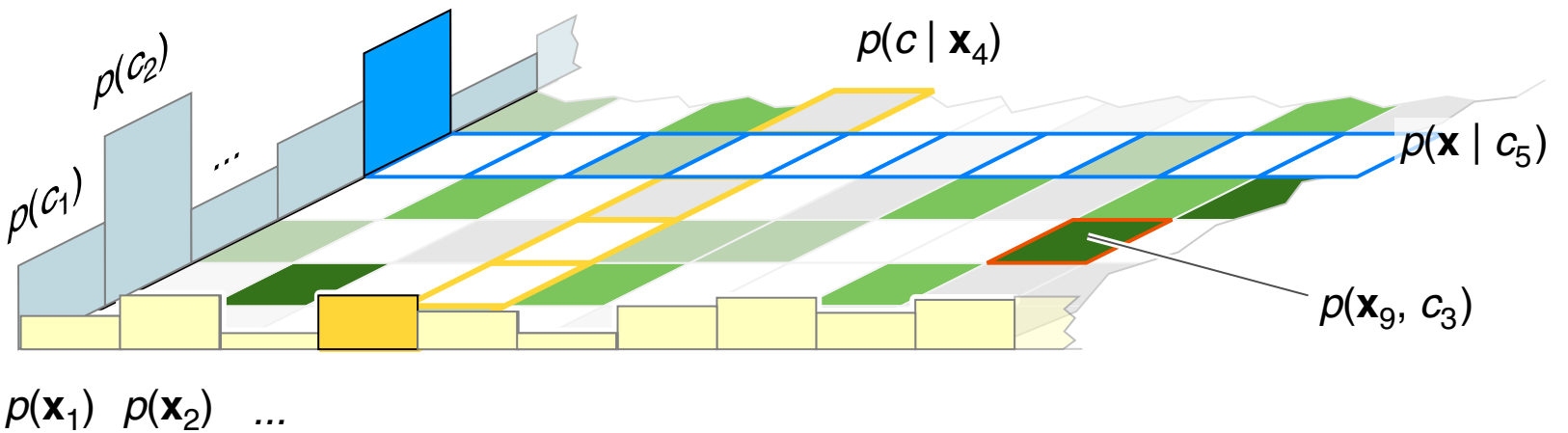


© STEIN/LETTMANN 2021

Evaluating Effectiveness

Illustration 1: Generic Feature and Class Distributions [Wikipedia] (continued)

Overview:



© STEIN/LETTMANN 2021

Remarks:

- $p(c \mid \mathbf{x}) = P(\mathcal{X}{=}\mathbf{x}, \mathcal{C}{=}c)/P(\mathcal{X}{=}\mathbf{x}) = P(\mathcal{C}{=}c \mid \mathcal{X}{=}\mathbf{x}) \;\equiv\; P_{\mathcal{X}=\mathbf{x}}(\mathcal{C}{=}c)$

  $p(c \mid \mathbf{x})$ is called "(feature-)conditional class probability" function.

  In the illustration: Summation over the $c \in C$ of the fourth column yields the marginal probability $p(\mathbf{x}_4) = P(\mathcal{X}{=}\mathbf{x}_4)$. $p(c \mid \mathbf{x}_4)$ is the distribution of the c (consider the column) under feature vector $\mathbf{x}_4$ (= having normalized by $p(\mathbf{x}_4)$), i.e., $p(\mathbf{x}_4, c)/p(\mathbf{x}_4)$.

- $p(\mathbf{x} \mid c) = P(\mathcal{X}{=}\mathbf{x}, \mathcal{C}{=}c)/P(\mathcal{C}{=}c) = P(\mathcal{X}{=}\mathbf{x} \mid \mathcal{C}{=}c) \;\equiv\; P_{\mathcal{C}=c}(\mathcal{X}{=}\mathbf{x})$

  $p(\mathbf{x} \mid c)$ is called "class-conditional (feature) probability [density]" function.

  In the illustration: Summation / integration over the $\mathbf{x} \in X$ of the fifth row yields the marginal probability $p(c_5) = P(\mathcal{C}{=}c_5)$. $p(\mathbf{x} \mid c_5)$ is the distribution of the $\mathbf{x}$ (consider the row) under class c_5 (= having normalized by $p(c_5)$), i.e., $p(\mathbf{x}, c_5)/p(c_5)$.

- $P(\mathcal{X}{=}\mathbf{x}, \mathcal{C}{=}c) = P(\mathcal{C}{=}c, \mathcal{X}{=}\mathbf{x}) = P(\mathcal{C}{=}c \mid \mathcal{X}{=}\mathbf{x}) \cdot P(\mathcal{X}{=}\mathbf{x})$, where $P(\mathcal{X}{=}\mathbf{x})$ is the prior probability for event $\mathcal{X}{=}\mathbf{x}$, and $P(\mathcal{C}{=}c \mid \mathcal{X}{=}\mathbf{x})$ is the probability for event $\mathcal{C}{=}c$ given event $\mathcal{X}{=}\mathbf{x}$.

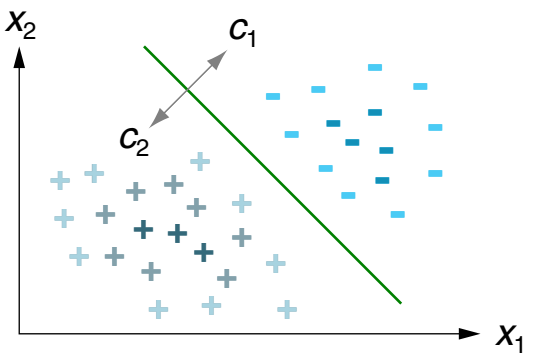
  Likewise, $P(\mathcal{X}{=}\mathbf{x}, \mathcal{C}{=}c) = P(\mathcal{X}{=}\mathbf{x} \mid \mathcal{C}{=}c) \cdot P(\mathcal{C}{=}c)$, where $P(\mathcal{C}{=}c)$ is the prior probability for event $\mathcal{C}{=}c$, and $P(\mathcal{X}{=}\mathbf{x} \mid \mathcal{C}{=}c)$ is the probability for event $\mathcal{X}{=}\mathbf{x}$ given event $\mathcal{C}{=}c$.

- Let the events $\mathcal{X}{=}\mathbf{x}$ and $\mathcal{C}{=}c$ have occurred, and, let $\mathbf{x}$ be known and c be unknown. Then, $P(\mathcal{X}{=}\mathbf{x} \mid \mathcal{C}{=}c)$ is called *likelihood* (for event $\mathcal{X}{=}\mathbf{x}$ given event $\mathcal{C}{=}c$). [Mathworld]

©STEIN/LETTMANN 2021

Evaluating Effectiveness

Illustration 2: Feature Distribution given Linearly Separable Classes

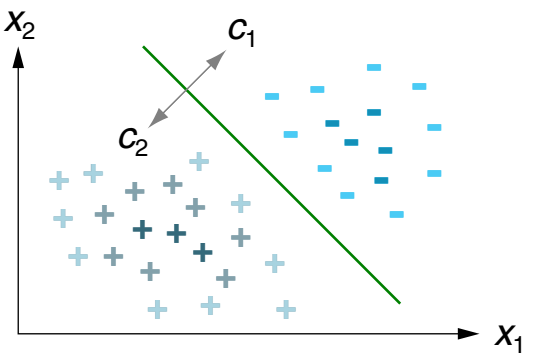


$$\mathbf{X} = \left\{ \underbrace{\begin{pmatrix} x_{1_1} \\ x_{1_2} \end{pmatrix}}_{\mathbf{x}_1}, \underbrace{\begin{pmatrix} x_{2_1} \\ x_{2_2} \end{pmatrix}}_{\mathbf{x}_2}, \underbrace{\ldots}_{\ldots} \right\} \subset \mathbf{R}^2$$

©STEIN/LETTMANN 2021

Evaluating Effectiveness

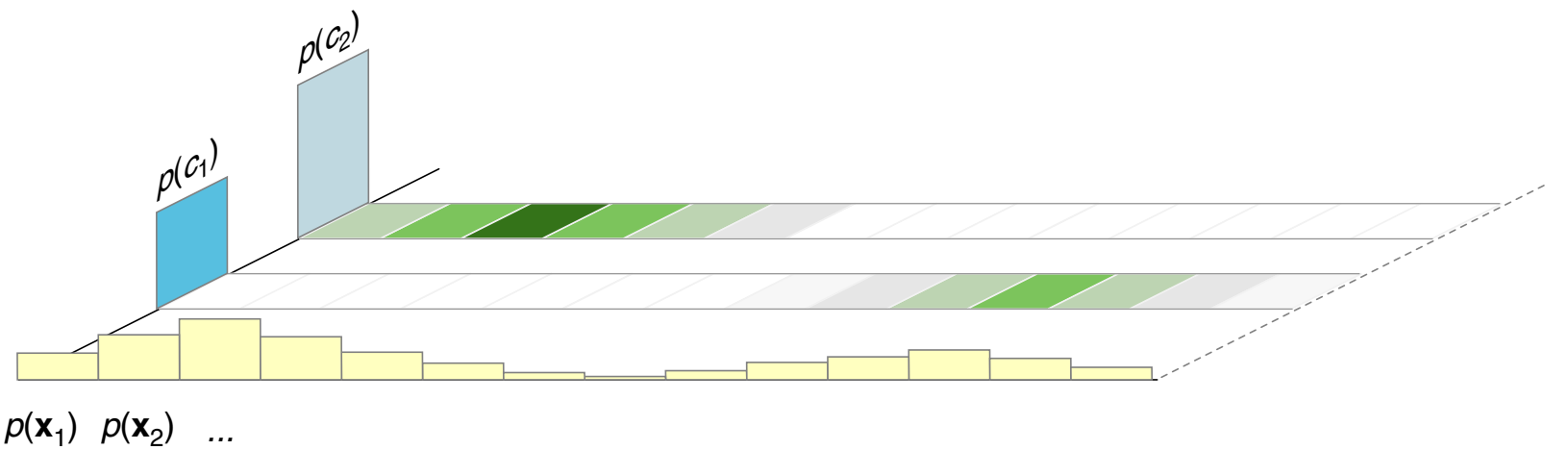
Illustration 2: Feature Distribution given Linearly Separable Classes (continued)



$$\mathbf{X} = \left\{ \begin{pmatrix} x_{1_1} \\ x_{1_2} \end{pmatrix}, \begin{pmatrix} x_{2_1} \\ x_{2_2} \end{pmatrix}, \ldots \right\} \subset \mathbf{R}^2$$

$$\mathbf{x}_1 \qquad \mathbf{x}_2 \qquad \ldots$$

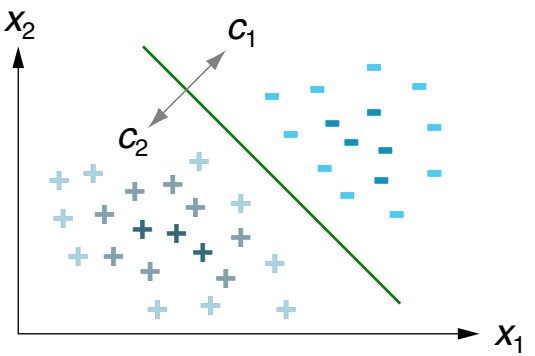
Joint and marginal probability functions $p(\mathbf{x}, c)$, $p(\mathbf{x})$, and $p(c)$:



© STEIN/LETTMANN 2021

Evaluating Effectiveness

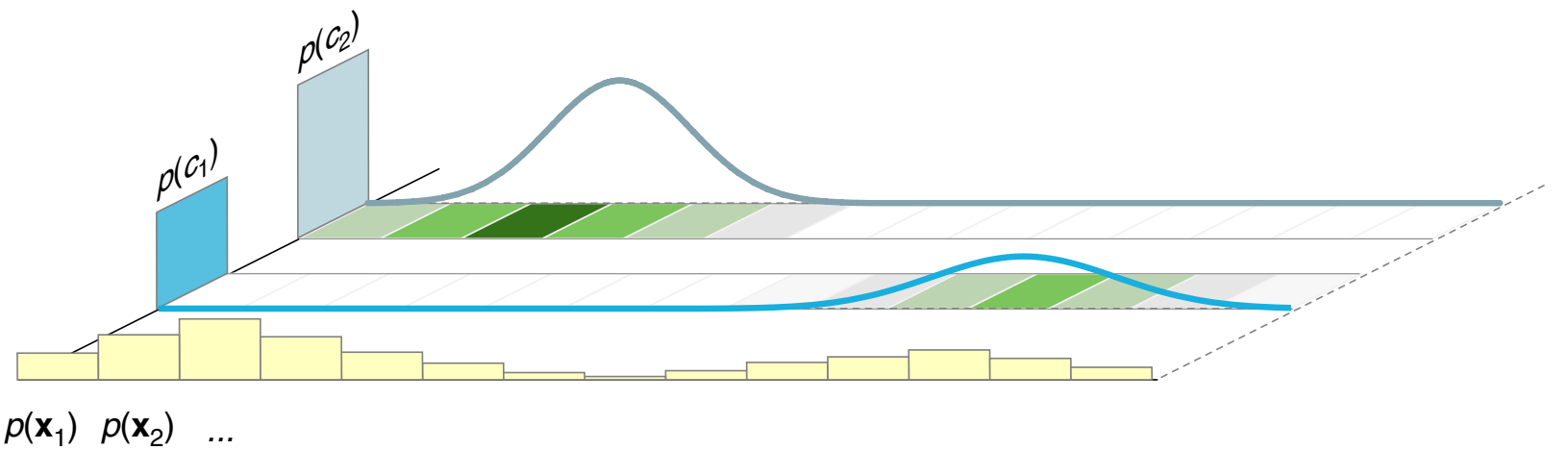
Illustration 2: Feature Distribution given Linearly Separable Classes (continued)



$$\mathbf{X} = \left\{ \begin{pmatrix} x_{1_1} \\ x_{1_2} \end{pmatrix}, \begin{pmatrix} x_{2_1} \\ x_{2_2} \end{pmatrix}, \ldots \right\} \subset \mathbf{R}^2$$

$$\mathbf{x}_1 \qquad \mathbf{x}_2 \qquad \ldots$$

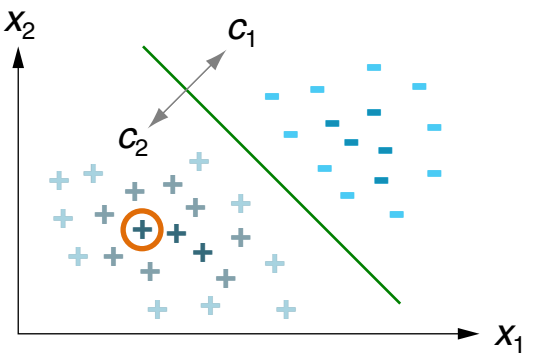
Joint and marginal probability functions $p(\mathbf{x}, c)$, $p(\mathbf{x})$, and $p(c)$:



© STEIN/LETTMANN 2021

Evaluating Effectiveness

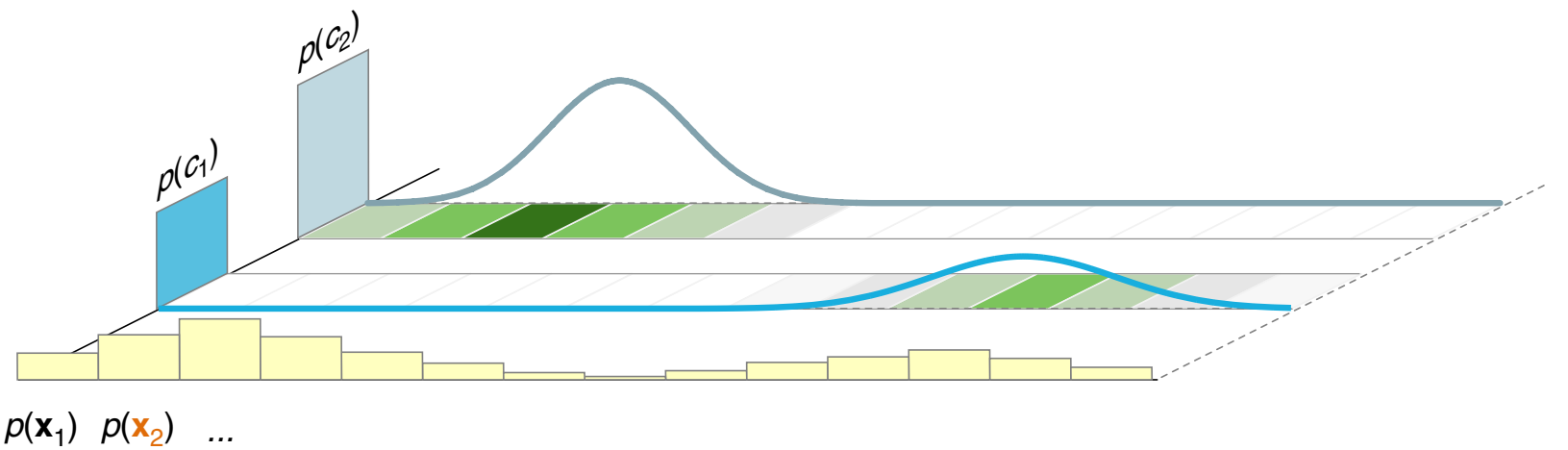
Illustration 2: Feature Distribution given Linearly Separable Classes (continued)



$$\mathbf{X} = \left\{ \begin{pmatrix} x_{1_1} \\ x_{1_2} \end{pmatrix}, \begin{pmatrix} x_{2_1} \\ x_{2_2} \end{pmatrix}, \ldots \right\} \subset \mathbf{R}^2$$

$$\mathbf{x}_1 \qquad \mathbf{x}_2 \qquad \ldots$$

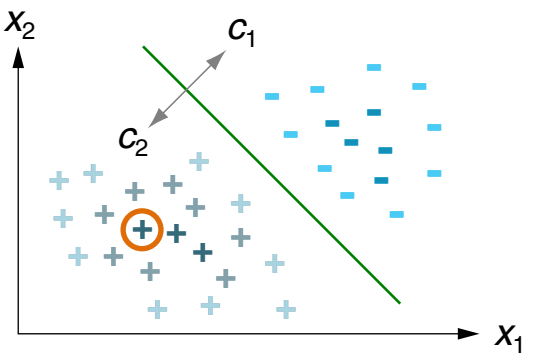
Joint and marginal probability functions $p(\mathbf{x}, c)$, $p(\mathbf{x})$, and $p(c)$:



© STEIN/LETTMANN 2021

Evaluating Effectiveness

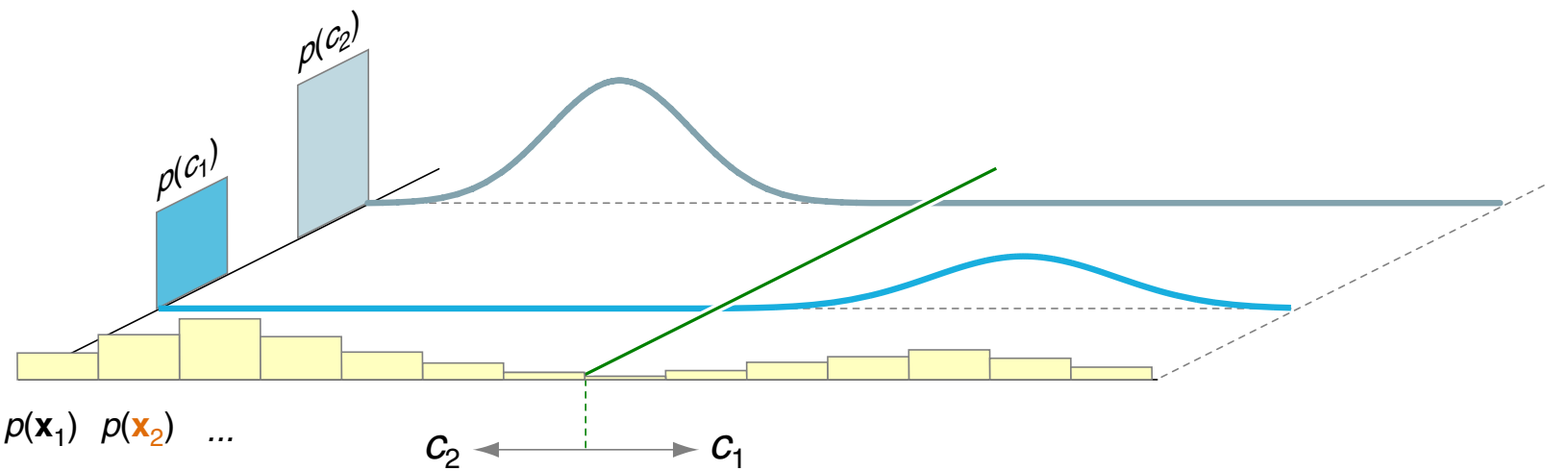
Illustration 2: Feature Distribution given Linearly Separable Classes (continued)



$$\mathbf{X} = \left\{ \begin{pmatrix} x_{1_1} \\ x_{1_2} \end{pmatrix}, \begin{pmatrix} x_{2_1} \\ x_{2_2} \end{pmatrix}, \ldots \right\} \subset \mathbf{R}^2$$

$$\mathbf{x}_1 \qquad \mathbf{x}_2 \qquad \ldots$$

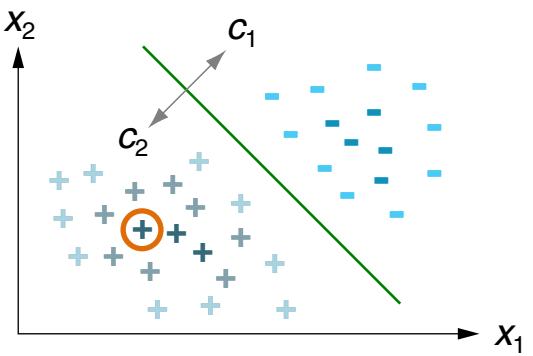
Optimum hyperplane classifier:



© STEIN/LETTMANN 2021

Evaluating Effectiveness

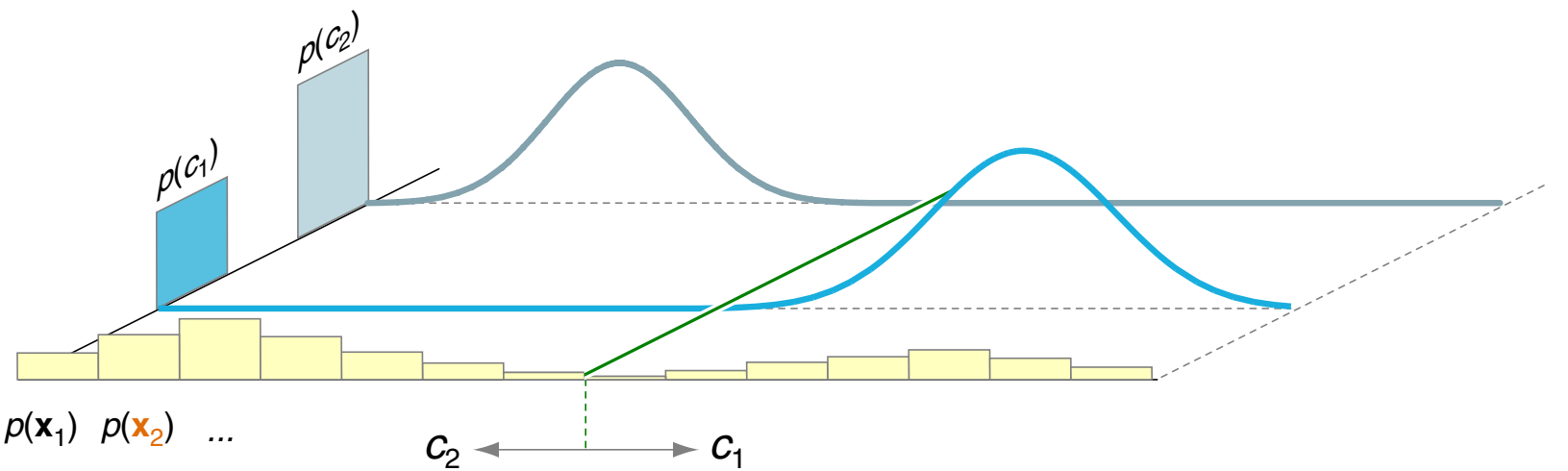
Illustration 2: Feature Distribution given Linearly Separable Classes (continued)



$$\mathbf{X} = \left\{ \begin{pmatrix} x_{1_1} \\ x_{1_2} \end{pmatrix}, \begin{pmatrix} x_{2_1} \\ x_{2_2} \end{pmatrix}, \ldots \right\} \subset \mathbf{R}^2$$

$$\mathbf{x}_1 \qquad \mathbf{x}_2 \qquad \ldots$$

Class-conditional probability functions $p(\mathbf{x} \mid c_1)$ and $p(\mathbf{x} \mid c_2)$:



© STEIN/LETTMANN 2021

Remarks:

- Actually, the two figures illustrate the discriminative approach (top) and the generative approach (bottom) to classification. See section Elements of Machine Learning in part Introduction.
- Linear separability requires a feature space with a particular structure. Here we assume that the feature space is a vector space over the scalar field of real numbers $\mathbb{R}$, equipped with the dot product.

© STEIN/LETTMANN 2021

Evaluating Effectiveness

Estimating Error Bounds

Classifier training:

- $D = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, c(\mathbf{x}_n))\} \subseteq X \times C$ is a set of examples.
- $y(\mathbf{x})$ is the classifier trained on D.
- The true error $\mathit{Err}^*(y)$ measures the performance of $y(\mathbf{x})$ on X ("in the wild").
- What can be said about the true error $\mathit{Err}^*(y)$ of $y(\mathbf{x})$?

©STEIN/LETTMANN 2021

Evaluating Effectiveness

Estimating Error Bounds

Classifier training:

- $D = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, c(\mathbf{x}_n))\} \subseteq X \times C$ is a set of examples.
- $y(\mathbf{x})$ is the classifier trained on D.
- The true error $Err^*(y)$ measures the performance of $y(\mathbf{x})$ on X ("in the wild").
- What can be said about the true error $Err^*(y)$ of $y(\mathbf{x})$?

The following relations typically hold:

Training error		Cross-validation error		Holdout error
$Err_{tr}(y)$	$<$	$Err_{cv}(y, k)$	$<$	$Err(y, D_{test})$

©STEIN/LETTMANN 2021

Evaluating Effectiveness

Estimating Error Bounds

Classifier training:

- $D = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, c(\mathbf{x}_n))\} \subseteq X \times C$ is a set of examples.
- $y(\mathbf{x})$ is the classifier trained on D.
- The true error $Err^*(y)$ measures the performance of $y(\mathbf{x})$ on X ("in the wild").
- What can be said about the true error $Err^*(y)$ of $y(\mathbf{x})$?

The following relations typically hold:

Training error		Cross-validation error		Holdout error		True error
$Err_{tr}(y)$	$<$	$Err_{cv}(y, k)$	$<$	$Err(y, D_{test})$	$<$	
						$Err^*(y)$

©STEIN/LETTMANN 2021

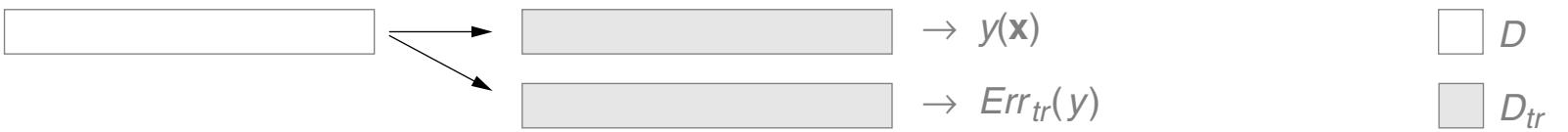
Remarks:

- Relating the true error $Err^*(y)$ to the aforementioned error assessments $Err_{tr}(y)$, $Err_{cv}(y,k)$, and $Err(y, D_{test})$ is not straightforward but requires an in-depth analysis of the sampling strategy, the sample size D, and the set X of feature vectors, among others.
- With $D = X$ the true error $Err^*(y)$ becomes the training error $Err_{tr}(y)$.

©STEIN/LETTMANN 2021

Evaluating Effectiveness

Training Error



Setting:

- No test set.
- $y(\mathbf{x})$ is the classifier trained on $D_{tr} = D$.

Training error of $y(\mathbf{x})$:

- $$Err_{tr}(y) = \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D_{tr} : y(\mathbf{x}) \neq c(\mathbf{x})\}|}{|D_{tr}|}$$

  $=$ misclassification rate of $y(\mathbf{x})$ on the training set.

©STEIN/LETTMANN 2021

Remarks:

- $\mathit{Err}_{tr}(y)$ is based on examples that are also exploited to train $y(\mathbf{x})$. It hence quantifies the *memorization* capability but not the *generalization* capability of $y(\mathbf{x})$.

- $\mathit{Err}_{tr}(y)$ is an optimistic estimation, i.e., it is constantly lower compared to the (unknown) true error $\mathit{Err}^*(y)$.

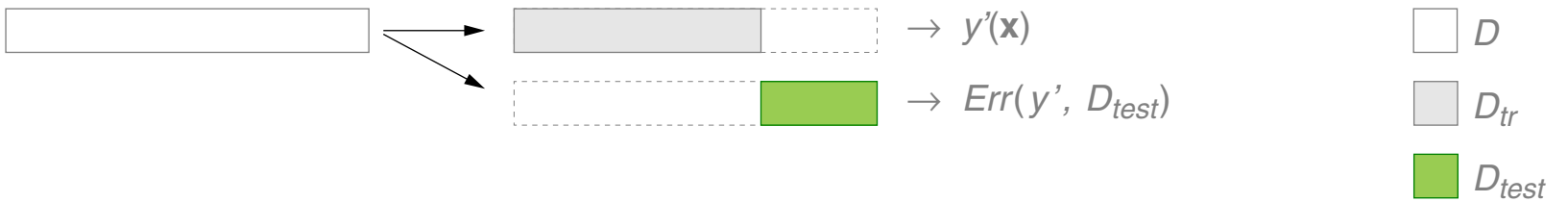
- Note that the above issues relate to the meaningfulness of $\mathit{Err}_{tr}(y)$ as an error estimate—and not to the classifier $y(\mathbf{x})$.
  Obviously, to get the maximum out of the data when training $y(\mathbf{x})$, D must be exploited completely: A classifier $y(\mathbf{x})$ trained on D will on average outperform any classifier $y'(\mathbf{x})$ trained on a subset of D. Hence, the error of $y'(\mathbf{x})$ may be considered as a lower bound for the (unknown) error of $y(\mathbf{x})$.

- Regarding notation: We will us the prime symbol »'« to indicate whether a classifier is trained by withholding a test set. E.g., $y'(\mathbf{x})$ and $y'_i(\mathbf{x})$ denote classifiers trained by withholding the test sets D_{test} and $D_{test\,i}$ respectively.

© STEIN/LETTMANN 2021

Evaluating Effectiveness

Holdout Error



Setting:

- $D_{test} \subset D$ is the test set.
- $y(\mathbf{x})$ is the classifier trained on D.
- $y'(\mathbf{x})$ is the classifier trained on $D_{tr} = D \setminus D_{test}$.

Holdout error of $y(\mathbf{x})$:

- $$\mathit{Err}(y, D_{test}) = \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D_{test} : y'(\mathbf{x}) \neq c(\mathbf{x})\}|}{|D_{test}|}$$

  $= \text{misclassification rate of } y'(\mathbf{x}) \text{ on the test set.}$

©STEIN/LETTMANN 2021

Remarks:

- When splitting D into D_{tr} and D_{test} one has to ensure that the underlying distribution is maintained, i.e., the examples have to be drawn independently and according to P. If this condition is not fulfilled then $Err(y, D_{test})$ cannot be used as an estimation of $Err^*(y)$.
  Keywords: stratification, sample selection bias

- D_{tr} and D_{test} should have similar sizes. A typical value for splitting D into training set D_{tr} and test set D_{test} is 2:1.

- The fact that random variables are both independent of each other and identically distributed is often abbreviated to "i.i.d."

©STEIN/LETTMANN 2021

Evaluating Effectiveness

k-Fold Cross-Validation

Setting:

- k test sets D_{test_i} by splitting D into k disjoint sets of similar size.

©STEIN/LETTMANN 2021

Evaluating Effectiveness

k-Fold Cross-Validation

Setting:

- k test sets D_{test_i} by splitting D into k disjoint sets of similar size.
- $i = 1, \ldots, k$:

  $y'_i(\mathbf{x})$ is the classifier trained on $D_{tr} = D \setminus D_{test_i}$,

  $$Err(y'_i, D_{test_i}) = \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D_{test_i} : y'_i(\mathbf{x}) \neq c(\mathbf{x})\}|}{|D_{test_i}|}$$

©STEIN/LETTMANN 2021

Evaluating Effectiveness

k-Fold Cross-Validation

Setting:

- k test sets D_{test_i} by splitting D into k disjoint sets of similar size.
- $i = 1, \ldots, k$:

  $y'_i(\mathbf{x})$ is the classifier trained on $D_{tr} = D \setminus D_{test_i}$,

  $$Err(y'_i, D_{test_i}) = \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D_{test_i} : y'_i(\mathbf{x}) \neq c(\mathbf{x})\}|}{|D_{test_i}|}$$

Cross-validation error of $y(\mathbf{x})$:

- $$Err_{cv}(y, k) = \frac{1}{k} \sum_{i=1}^{k} Err(y'_i, D_{test_i})$$

  $=$ averaged misclassification rate on the k test sets.

©STEIN/LETTMANN 2021

Remarks:

- For large k the set $D_{tr} = D \setminus D_{test_i}$ is of similar size as D. Hence $Err(y_i, D_{test_i})$—as well as $Err_{cv}(y, k)$—is close to $Err^*(y)$, since $Err^*(y)$ is the error of the classifier y trained on D.
- n-fold cross-validation (aka "leave one out") is the special case with $k = n$. Obviously singleton test sets ($|D_{test_i}| = 1$) are never stratified since they contain a single class only.
- n-fold cross-validation is a special case of exhaustive cross-validation methods, which learn and test on all possible ways to divide the original sample into a training and a validation set. [Wikipedia]

©STEIN/LETTMANN 2021

Evaluating Effectiveness

Comparing Model Variants

Classifier training:

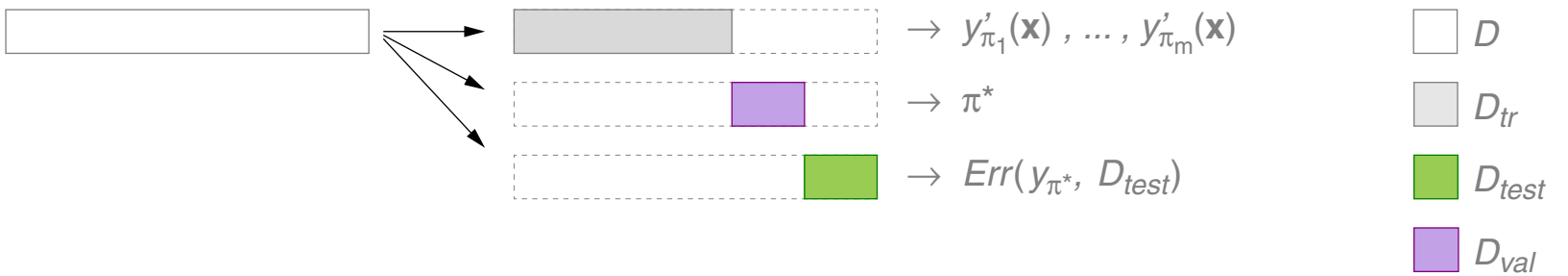
- $D = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, c(\mathbf{x}_n))\} \subseteq X \times C$ is a set of examples.
- m hyperparameter values $\pi_1, \pi_2, \ldots, \pi_m$,

  m classifiers $y_{\pi_1}(\mathbf{x}), y_{\pi_2}(\mathbf{x}), \ldots, y_{\pi_m}(\mathbf{x})$.
- $y_{\pi_j}(\mathbf{x})$ is the classifier trained on D.
- Which is the most effective among the m classifiers $y_{\pi_j}(\mathbf{x})$?

©STEIN/LETTMANN 2021

Evaluating Effectiveness

Model Selection (single validation set)



Setting:

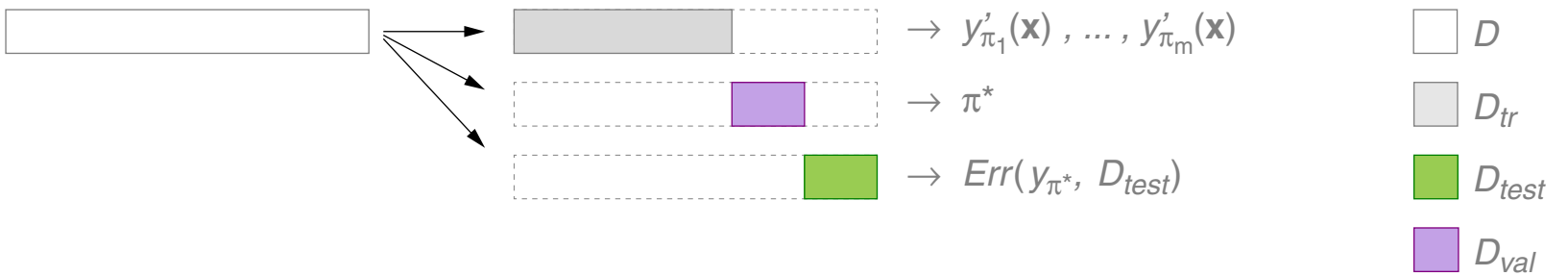
- $D_{test} \subset D$ is the test set,

  $D_{val} \subset (D \setminus D_{test})$ is the validation set.

©STEIN/LETTMANN 2021

Evaluating Effectiveness

Model Selection (single validation set)



Setting:

- $D_{test} \subset D$ is the test set,

  $D_{val} \subset (D \setminus D_{test})$ is the validation set.

- $j = 1, \ldots, m$:

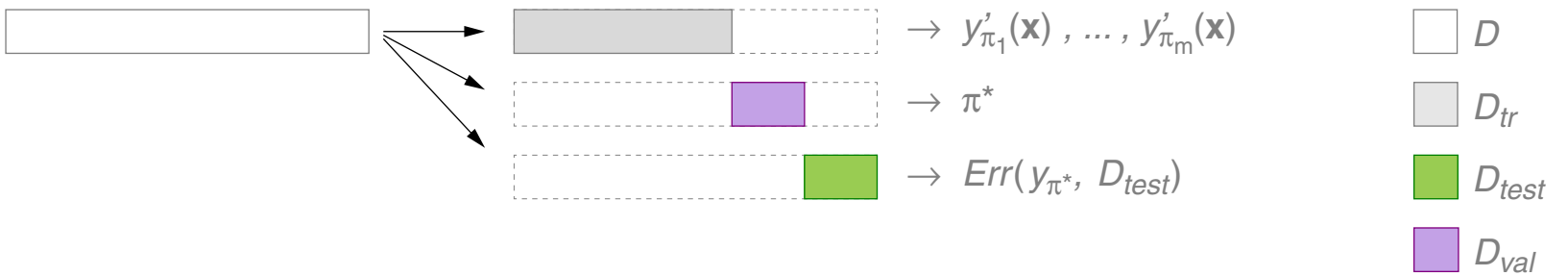
  $y'_{\pi_j}(\mathbf{x})$ is the classifier trained on $D_{tr} = D \setminus (D_{test} \cup D_{val})$,

$$\underline{Err(y'_{\pi_j}, D_{val})} = \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D_{val} : y'_{\pi_j}(\mathbf{x}) \neq c(\mathbf{x})\}|}{|D_{val}|}$$

©STEIN/LETTMANN 2021

Evaluating Effectiveness

Model Selection (single validation set) [Holdout Error]



(continued)

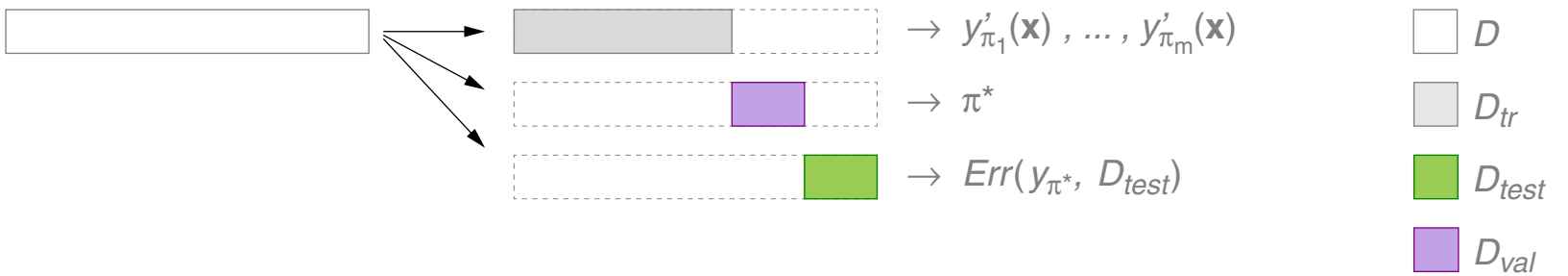
- $\pi^* = \underset{\pi_j,\ j=1,\ldots,m}{\text{argmin}} \ \ Err(y'_{\pi_j}, D_{val})$
- $y_{\pi^*}(\mathbf{x})$ is the classifier trained on D,

  $y'_{\pi^*}(\mathbf{x})$ is the classifier trained on $D_{tr} = D \setminus D_{test}$.

©STEIN/LETTMANN 2021

Evaluating Effectiveness

Model Selection (single validation set) [Holdout Error]



(continued)

- $\pi^* = \underset{\pi_j,\ j=1,\ldots,m}{\text{argmin}} \ \mathit{Err}(y'_{\pi_j}, D_{\mathit{val}})$
- $y_{\pi^*}(\mathbf{x})$ is the classifier trained on D,

  $y'_{\pi^*}(\mathbf{x})$ is the classifier trained on $D_{\mathit{tr}} = D \setminus D_{\mathit{test}}$.

Holdout error of $y_{\pi^*}(\mathbf{x})$:

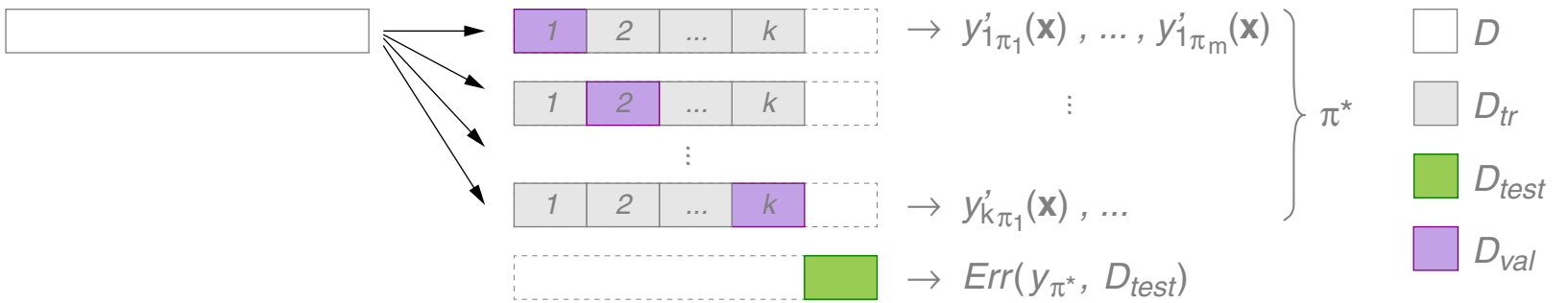
- $\mathit{Err}(y_{\pi^*}, D_{\mathit{test}}) = \dfrac{|\{(\mathbf{x}, c(\mathbf{x})) \in D_{\mathit{test}} : y'_{\pi^*}(\mathbf{x}) \neq c(\mathbf{x})\}|}{|D_{\mathit{test}}|}$

  $=$ misclassification rate of $y'_{\pi^*}(\mathbf{x})$ on the test set.

©STEIN/LETTMANN 2021

Evaluating Effectiveness

Model Selection (k validation sets)



Setting:

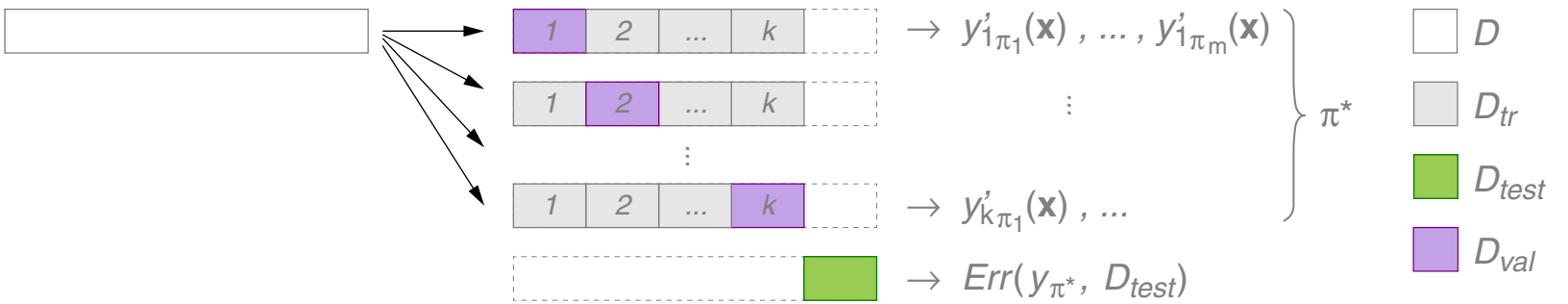
- $D_{test} \subset D$ is the test set,

  k validation sets D_{val_i} by splitting $D \setminus D_{test}$ into k disjoint sets of similar size.

©STEIN/LETTMANN 2021

Evaluating Effectiveness

Model Selection (k validation sets)



Setting:

- $D_{test} \subset D$ is the test set,

  k validation sets D_{val_i} by splitting $D \setminus D_{test}$ into k disjoint sets of similar size.

- $i = 1, \ldots, k$:

  $j = 1, \ldots, m$:

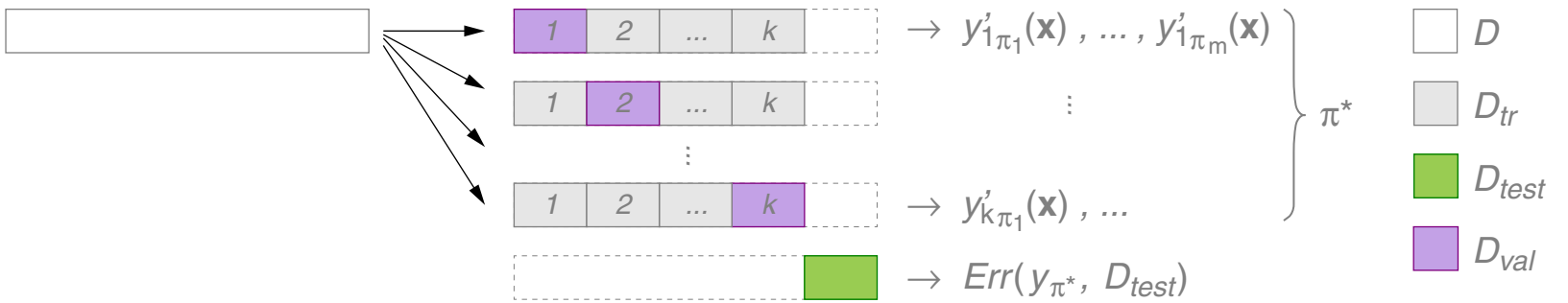
  $y'_{i_{\pi_j}}(\mathbf{x})$ is the classifier trained on $D_{tr} = D \setminus (D_{test} \cup D_{val_i})$,

$$Err(y'_{i_{\pi_j}}, D_{val_i}) = \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D_{val_i} : y'_{i_{\pi_j}}(\mathbf{x}) \neq c(\mathbf{x})\}|}{|D_{val_i}|}$$

©STEIN/LETTMANN 2021

Evaluating Effectiveness

Model Selection (k validation sets)



(continued)

- $\pi^* = \underset{\pi_j,\ j=1,\ldots,m}{\text{argmin}} \sum_{i=1}^{k} Err(y'_{i_{\pi_j}}, D_{val_i})$

- $y_{\pi^*}(\mathbf{x})$ is the classifier trained on D,

  $y'_{\pi^*}(\mathbf{x})$ is the classifier trained on $D_{tr} = D \setminus D_{test}$.

Holdout error of $y_{\pi^*}(\mathbf{x})$: computation as before.

©STEIN/LETTMANN 2021

Remarks:

- The validation set is also called “development set”.

©STEIN/LETTMANN 2021

Evaluating Effectiveness

Misclassification Costs

Use of a *cost measure* for the misclassification of a feature vector $\mathbf{x} \in X$ in a wrong class c' instead of in the correct class c:

$$\textit{cost}(c' \mid c) \begin{cases} \geq 0 & \text{if } c' \neq c \\ = 0 & \text{otherwise} \end{cases}$$

Holdout error of y based on misclassification costs:

$$\textit{Err}_{\textit{cost}}(y, D_{\textit{test}}) = \frac{1}{|D_{\textit{test}}|} \cdot \sum_{(\mathbf{x}, c(\mathbf{x})) \in D_{\textit{test}}} \textit{cost}(y(\mathbf{x}) \mid c(\mathbf{x}))$$

©STEIN/LETTMANN 2021

Remarks:

- The true error, $\mathit{Err}^*(y)$, is a special case of $\mathit{Err}_{\mathit{cost}}(y)$ with $\mathit{cost}(c' \mid c) = 1$ for $c' \neq c$. Consider in this regard the notation of $\mathit{Err}^*(y)$ in terms of the function $I(y(\mathbf{x}), c)$:

$$\mathit{Err}^*(y) \;=\; \frac{|\{\mathbf{x} \in X : y(\mathbf{x}) \neq c(\mathbf{x})\}|}{|X|} \;=\; \sum_{\mathbf{x} \in X} I(y(\mathbf{x}), c)$$

©STEIN/LETTMANN 2021