

Chapter ML:II (continued)

II. Machine Learning Basics

- ❑ Concept Learning: Search in Hypothesis Space
- ❑ Concept Learning: Version Space
- ❑ From Regression to Classification
- ❑ Evaluating Effectiveness

Concept Learning: Search in Hypothesis Space

Simple Classification Problems

Setting:

- X is a multiset of feature vectors.
- $C = \{\text{no}, \text{yes}\}$ is a set of two classes.
Similarly: $\{0, 1\}$, $\{-1, 1\}$, $\{\ominus, \oplus\}$, “belongs to a concept or not”, etc.
- $D = \{(\mathbf{x}_1, c_1), \dots, (\mathbf{x}_n, c_n)\} \subseteq X \times C$ is a multiset of examples.

Learning task:

- Approximate D with a feature-value pattern.

Concept Learning: Search in Hypothesis Space

Example Learning Task

X contains vectors encoding weather in the six dimensions “Sky”, “Temperature”, “Humidity”, “Wind”, “Water”, and “Forecast”. D contains examples of weather conditions $\mathbf{x} \in X$ along with a statement whether or not our friend will enjoy her favorite sport (surfing):

Example	Sky	Temperature	Humidity	Wind	Water	Forecast	EnjoySport
1	sunny	warm	normal	strong	warm	same	yes 1
2	sunny	warm	high	strong	warm	same	yes 1
3	rainy	cold	high	strong	warm	change	no 0
4	sunny	warm	high	strong	cool	change	yes 1

- ❑ 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” ?

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 contains all possible weather conditions, while the subset (= the concept) contains those weather conditions when surfing is 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.
- ❑ In our learning setting, a hypothesis cannot be inferred or proven by deductive reasoning. A hypothesis is a finding or an insight gained by *inductive reasoning*.

Concept Learning: Search in Hypothesis Space

Simple Classification Problems (continued)

Definition 1 (Concept, Hypothesis, Hypothesis Space)

Let O be a set of objects, \mathbf{X} the feature space associated with a model formation function $\alpha : O \rightarrow \mathbf{X}$, and $X = \{\mathbf{x} \mid \mathbf{x} = \alpha(o), o \in O\}$ be a multiset of feature vectors.

A concept is a subset of O and induces a subset $X' \subseteq X$. Concept learning means learning the indicator function for X' , which returns 1 if $\mathbf{x} \in X'$ and 0 otherwise.

Concept Learning: Search in Hypothesis Space

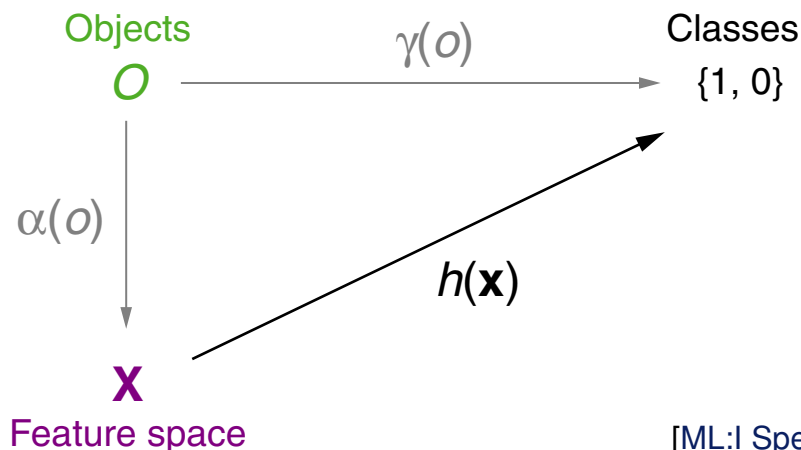
Simple Classification Problems (continued)

Definition 1 (Concept, Hypothesis, Hypothesis Space)

Let O be a set of objects, \mathbf{X} the feature space associated with a model formation function $\alpha : O \rightarrow \mathbf{X}$, and $X = \{\mathbf{x} \mid \mathbf{x} = \alpha(o), o \in O\}$ be a multiset of feature vectors.

A concept is a subset of O and induces a subset $X' \subseteq X$. Concept learning means learning the indicator function for X' , which returns 1 if $\mathbf{x} \in X'$ and 0 otherwise.

A hypothesis is a function $h(\mathbf{x})$, $h : X \rightarrow \{0, 1\}$, that approximates the indicator function for X' based on an example set D . The hypothesis space is a set H of hypotheses among which $h(\mathbf{x})$ is searched.



[ML:I Specification of Learning Problems]

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, \mathbf{w} , as hypothesis (and not the model 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, model) 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.

Concept Learning: Search in Hypothesis Space

Simple Classification Problems (continued)

The example set D , $D = \{(\mathbf{x}_1, c_1), \dots, (\mathbf{x}_n, c_n)\}$, contains usually both positive ($c = 1$) and negative ($c = 0$) examples. [\[learning task\]](#)

Definition 2 (Positive Classified, Consistent)

An example (\mathbf{x}, c) is positive classified by a hypothesis $h(\mathbf{x})$ iff $h(\mathbf{x}) = 1$.

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

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

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

Remarks:

- ❑ The string “lff” 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: concept, target concept, target function.
- ❑ 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) , notice the difference between (1) positive classified and (2) being consistent with a hypothesis. The former asks for $h(\mathbf{x}) = 1$, disregarding the actual target concept value c . The latter asks for the identity between the target concept c 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 that $h(\mathbf{x}) = 1$ iff $c = 1$, for all $(\mathbf{x}, c) \in D$. This is equivalent with the condition that $h(\mathbf{x}) = 0$ iff $c = 0$, for all $(\mathbf{x}, c) \in D$.
- ❑ 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.

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](#)]: $\langle \text{sunny}, ?, ?, \text{strong}, ?, \text{same} \rangle$

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]: $\langle \text{sunny}, ?, ?, \text{strong}, ?, \text{same} \rangle$

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 $\mathbf{x} \in X$ is positive classified by $s_0(\mathbf{x})$, and all $\mathbf{x} \in X$ are positive classified by $g_0(\mathbf{x})$.

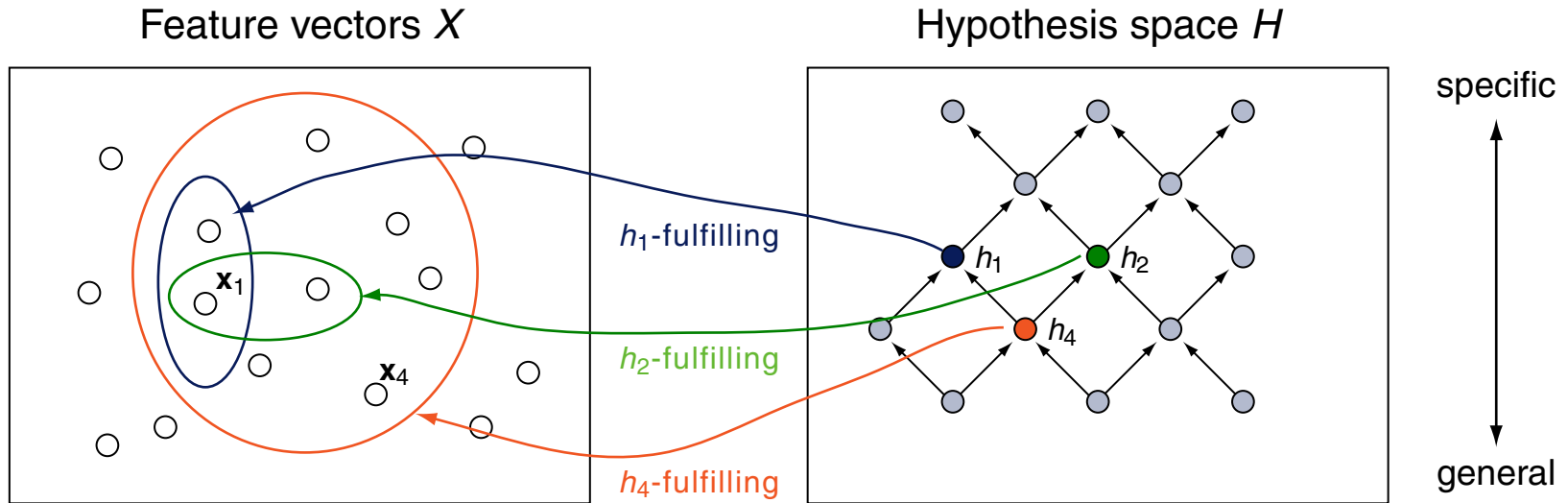
Maximally specific / general hypothesis in the example [learning task]:

$$\square \quad s_0 = \langle \perp, \perp, \perp, \perp, \perp, \perp \rangle \quad (\text{never enjoy sport})$$

$$\square \quad g_0 = \langle ?, ?, ?, ?, ?, ? \rangle \quad (\text{always enjoy sport})$$

Concept Learning: Search in Hypothesis Space

Order of Hypotheses



$x_1 = (\text{sunny}, \text{warm}, \text{normal}, \text{strong}, \text{warm}, \text{same})$

$x_4 = (\text{sunny}, \text{warm}, \text{high}, \text{strong}, \text{cool}, \text{change})$

$h_1 = \langle \text{sunny}, ?, \text{normal}, ?, ?, ? \rangle$

$h_2 = \langle \text{sunny}, ?, ?, ?, \text{warm}, ? \rangle$

$h_4 = \langle \text{sunny}, ?, ?, ?, ?, ? \rangle$

Concept Learning: Search in Hypothesis Space

Order of Hypotheses (continued)

Definition 4 (More General Relation)

Let X be a multiset 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 strictly more general than $h_2(\mathbf{x})$, denoted as $h_1(\mathbf{x}) >_g h_2(\mathbf{x})$, iff:

$$(h_1(\mathbf{x}) \geq_g h_2(\mathbf{x})) \text{ and } (h_2(\mathbf{x}) \not\geq_g h_1(\mathbf{x}))$$

In the illustration: $h_2(\mathbf{x}) = 1$ implies that $h_4(\mathbf{x}) = 1$. I.e., h_4 is more general than h_1 .

Concept Learning: Search in Hypothesis Space

Order of Hypotheses (continued)

Definition 4 (More General Relation)

Let X be a multiset 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 strictly more general than $h_2(\mathbf{x})$, denoted as $h_1(\mathbf{x}) >_g h_2(\mathbf{x})$, iff:

$$(h_1(\mathbf{x}) \geq_g h_2(\mathbf{x})) \text{ and } (h_2(\mathbf{x}) \not\geq_g h_1(\mathbf{x}))$$

In the illustration: $h_2(\mathbf{x}) = 1$ implies that $h_4(\mathbf{x}) = 1$. I.e., h_4 is more general than h_1 .

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 $g_0(\mathbf{x})$.

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 $h_1(\mathbf{x})$.
- ❑ The relations \geq_g and $>_g$ are independent of a target concept. They depend only on the fact that examples are positive classified by a hypothesis, i.e., whether $h(\mathbf{x}) = 1$, $(\mathbf{x}, c) \in D$. It is not required that $c = 1$.
- ❑ 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 [illustration](#).

Remarks on entailment:

- ❑ The semantics of the implication, in words “ a implies b ”, denoted as $a \rightarrow b$, is as follows. $a \rightarrow b$ 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 ”.
- ❑ “ \rightarrow ” 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 $a \rightarrow b$ 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 “ \rightarrow ” 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 formula β given a formula α .

Consider for instance the More-General-Definition: From the formula $\alpha = “h_2(\mathbf{x}) = 1”$ we cannot infer or prove the formula $\beta = “h_1(\mathbf{x}) = 1”$.

- ❑ 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 $h_2(\mathbf{x}) = 1 \rightarrow h_1(\mathbf{x}) = 1$ is true for all $\mathbf{x} \in X$.

Remarks on entailment: (continued)

- ❑ In a nutshell: distinguish carefully between “ α requires β ”, denoted as $\alpha \rightarrow \beta$, on the one hand, and “from α follows β ”, denoted as $\alpha \Rightarrow \beta$, on the other hand. $\alpha \rightarrow \beta$ 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 $\alpha \rightarrow \beta$, namely: “ $\alpha \rightarrow \beta$ is a tautology”.
- ❑ Finally, consider the following sentences from the object language, which are synonymous:
 - “ $\alpha \rightarrow \beta$ ”
 - “ α implies β ”
 - “if α then β ”
 - “ α causes β ”
 - “ α requires β ”
 - “ α is sufficient for β ”
 - “ β is necessary for α ”

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]

Concept Learning: Search in Hypothesis Space

Find-S Algorithm

1. $h(\mathbf{x}) = s_0(\mathbf{x})$ // $h(\mathbf{x})$ is a maximally specific hypothesis in H .
2. **FOREACH** $(\mathbf{x}, c) \in D$ **DO**
 IF $c = 1$ **THEN** // Learn only from positive examples.
 IF $h(\mathbf{x}) = 0$ **DO**
 $h = \text{min_generalization}(h, \mathbf{x})$ // Relax $h(\mathbf{x})$ wrt. \mathbf{x} .
 ENDIF
 ENDIF
ENDDO
3. *return*($h(\mathbf{x})$)

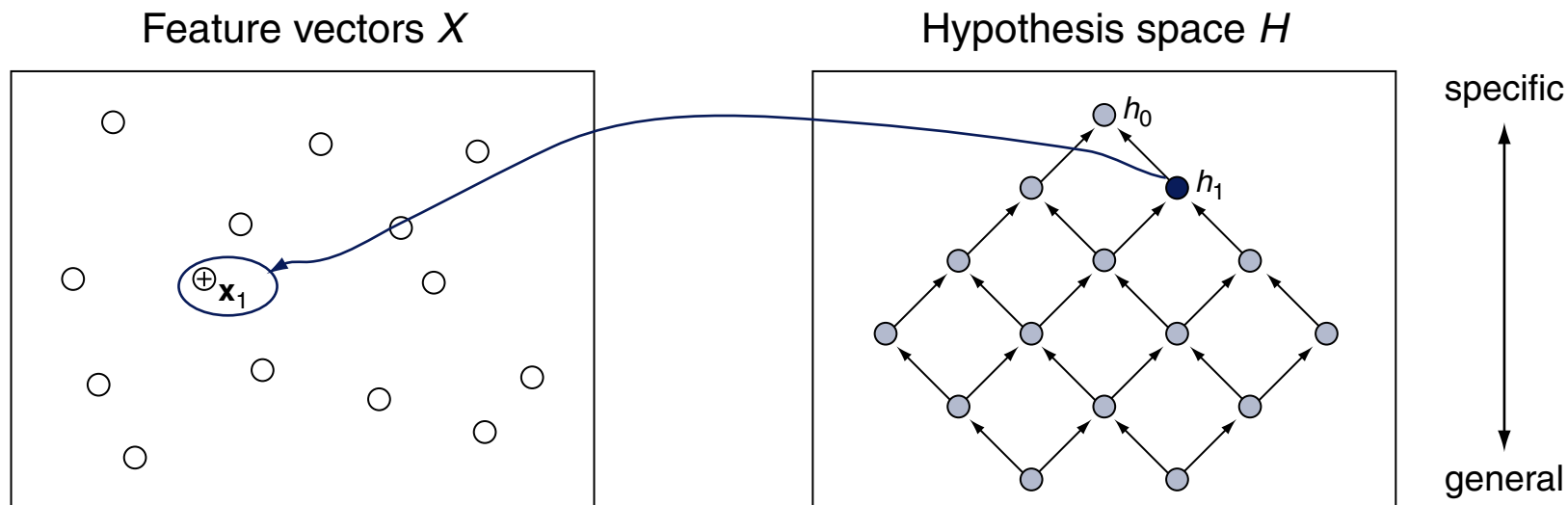
Remarks:

- ❑ Except for the first step, generalization means to substitute question marks (wildcards) for literals. Another term for “generalization” is “relaxation”.
- ❑ The function $\text{min_generalization}(h, \mathbf{x})$ 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 $h(\mathbf{x})$, $\text{min_generalization}(h, \mathbf{x})$, 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 $\text{min_generalization}(h, \mathbf{x})$, a function $\text{min_specialization}(h, \mathbf{x})$ can be defined, which returns a minimally specialized, consistent hypotheses for negative examples.

Concept Learning: Search in Hypothesis Space

Find-S Algorithm (continued)

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



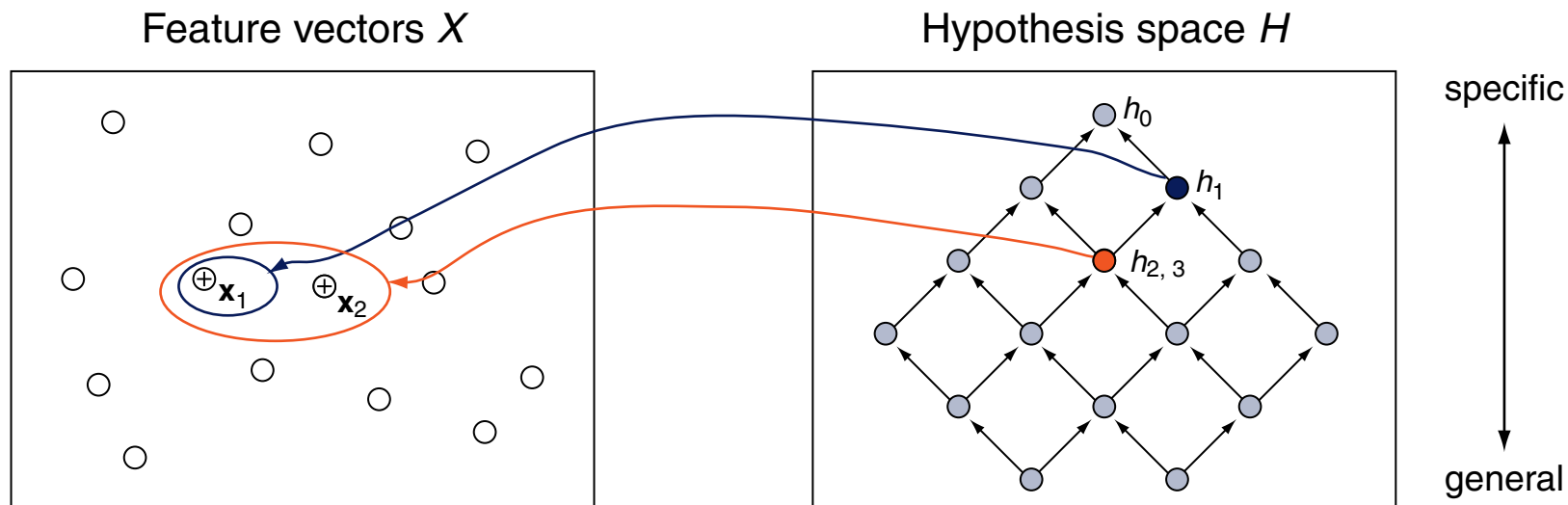
$$h_0 = \underline{s_0} = \langle \perp, \perp, \perp, \perp, \perp, \perp \rangle$$

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

Concept Learning: Search in Hypothesis Space

Find-S Algorithm (continued)

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



$$h_0 = \underline{s_0} = \langle \perp, \perp, \perp, \perp, \perp, \perp \rangle$$

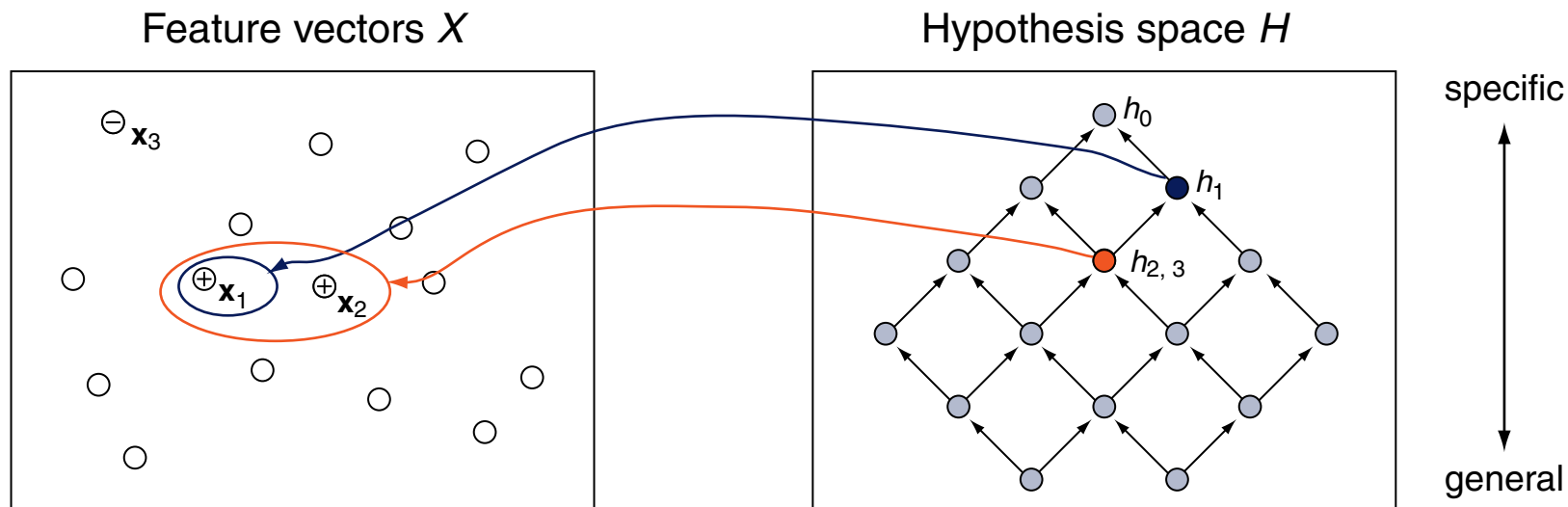
$$x_1 = (\text{sunny, warm, normal, strong, warm, same}) \quad h_1 = \langle \text{sunny, warm, normal, strong, warm, same} \rangle$$

$$x_2 = (\text{sunny, warm, high, strong, warm, same}) \quad h_2 = \langle \text{sunny, warm, ?, strong, warm, same} \rangle$$

Concept Learning: Search in Hypothesis Space

Find-S Algorithm (continued)

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



$$h_0 = \underline{s_0} = \langle \perp, \perp, \perp, \perp, \perp, \perp \rangle$$

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

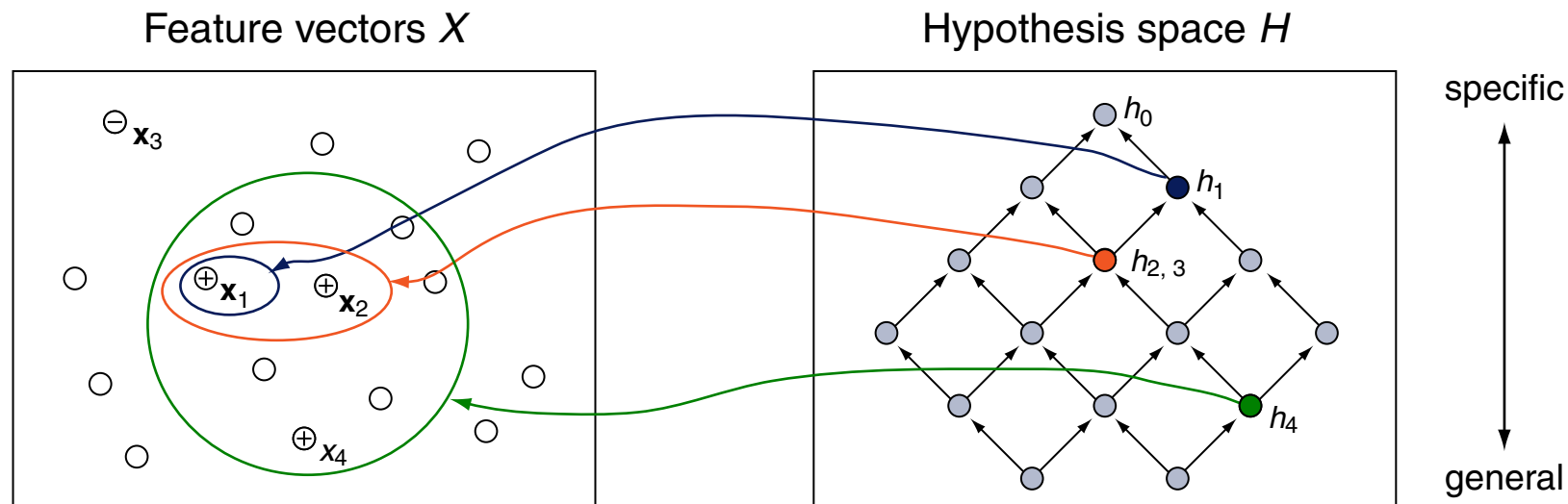
$$\mathbf{x}_2 = (\text{sunny, warm, high, strong, warm, same}) \quad h_2 = \langle \text{sunny, warm, ?, strong, warm, same} \rangle$$

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

Concept Learning: Search in Hypothesis Space

Find-S Algorithm (continued)

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



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

$$x_1 = (\text{sunny, warm, normal, strong, warm, same}) \quad h_1 = \langle \text{sunny, warm, normal, strong, warm, same} \rangle$$

$$x_2 = (\text{sunny, warm, high, strong, warm, same}) \quad h_2 = \langle \text{sunny, warm, ?, strong, warm, same} \rangle$$

$$x_3 = (\text{rainy, cold, high, strong, warm, change}) \quad h_3 = \langle \text{sunny, warm, ?, strong, warm, same} \rangle$$

$$x_4 = (\text{sunny, warm, high, strong, cool, change}) \quad h_4 = \langle \text{sunny, warm, ?, strong, ?, ?} \rangle$$

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 pursue 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.

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:

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

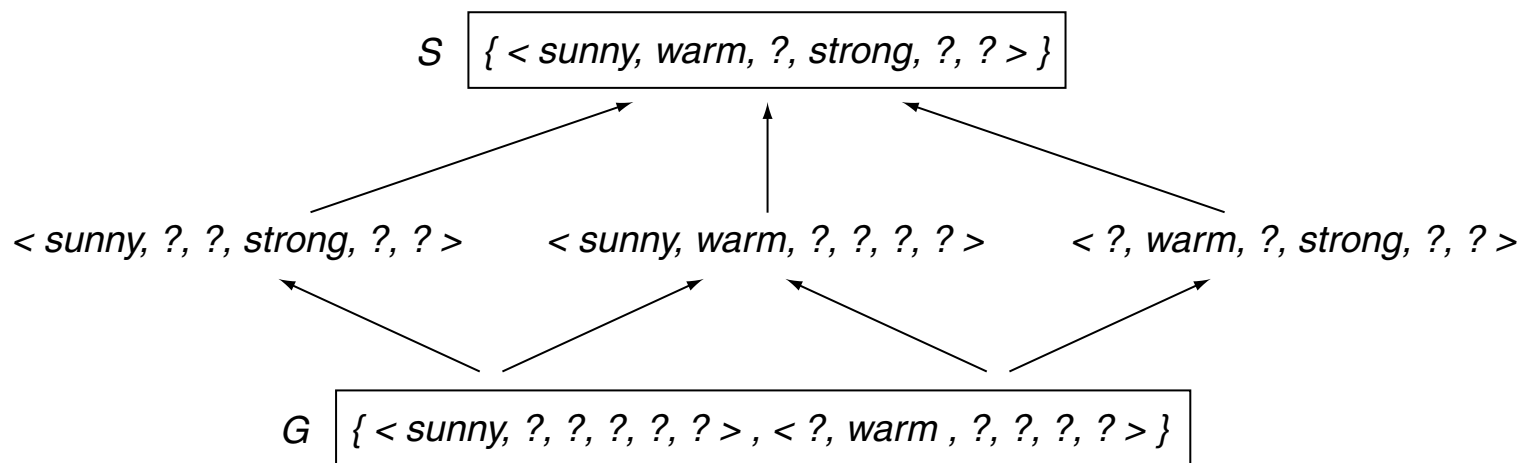
Concept Learning: Version Space

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:

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

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



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 $h(\mathbf{x}) \neq c$ holds. This approach presumes a finite hypothesis space H and is feasible only for toy problems.

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:

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

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

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

Concept Learning: Version Space

Theorem 7 (Version Space Representation)

Let X be a multiset of feature vectors, $C = \{0, 1\}$ be a set of classes, and H be a set of boolean-valued functions with domain X . Moreover, let $D \subseteq X \times C$ be a multiset of examples.

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

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

Remarks:

- The correctness of Theorem 7 is not obvious. The theorem allows us to characterize the set of all consistent hypotheses by the two boundary sets G and S .

Concept Learning: Version Space

Candidate Elimination Algorithm [Mitchell 1997]

1. Initialization: $G = \{g_0\}$, $S = \{s_0\}$
2. If x is a **positive** example
 - Remove from G any hypothesis that is not consistent with x
 - For each hypothesis s in S that is not consistent with x
 - Remove s from S
 - Add to S all minimal **generalizations** h of s such that
 1. h is consistent with 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

Concept Learning: Version Space

Candidate Elimination Algorithm [Mitchell 1997] (continued)

1. Initialization: $G = \{g_0\}$, $S = \{s_0\}$
2. If x is a **positive** example
 - Remove from G any hypothesis that is not consistent with x
 - For each hypothesis s in S that is not consistent with x
 - Remove s from S
 - Add to S all minimal **generalizations** h of s such that
 1. h is consistent with 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 x is a **negative** example
 - Remove from S any hypothesis that is not consistent with x
 - For each hypothesis g in G that is not consistent with x
 - Remove g from G
 - Add to G all minimal **specializations** h of g such that
 1. h is consistent with 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

Remarks:

- ❑ All hypothesis between G and S are consistent with all examples seen so far; i.e., they “accept” the positive examples and “reject” the negative examples.
- ❑ The basic idea of Candidate Elimination is as follows:
 - Deal with false positives. 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 $g(\mathbf{x})$.
 - Deal with false negatives. 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 $s(\mathbf{x})$.
- ❑ The G boundary of the version space summarizes the information from the previously encountered negative examples. The S boundary forms a summary of the previously encountered positive examples.

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** $(\mathbf{x}, c) \in D$ **DO**
 IF $c = 1$ **THEN** // \mathbf{x} is a positive example.
 FOREACH $g \in G$ **DO** **IF** $g(\mathbf{x}) \neq 1$ **THEN** $G = G \setminus \{g\}$ **ENDDO**
 FOREACH $s \in S$ **DO**
 IF $s(\mathbf{x}) \neq 1$ **THEN**
 $S = S \setminus \{s\}$, $S^+ = \text{min_generalizations}(s, \mathbf{x})$
 FOREACH $s \in S^+$ **DO** **IF** $(\exists g \in G : g \geq_g s)$ **THEN** $S = S \cup \{s\}$ **ENDDO**
 FOREACH $s \in S$ **DO** **IF** $(\exists s' \in S : s' \neq s \wedge s \geq_g s')$ **THEN** $S = S \setminus \{s\}$ **ENDDO**
 ENDDO
 ELSE // \mathbf{x} is a negative example.
 ENDIF
 ENDDO
3. **return** (G, S)

Concept Learning: Version Space

Candidate Elimination Algorithm (pseudo code) (continued)

```
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  $(\mathbf{x}, c) \in D$  DO  
   IF  $c = 1$  THEN //  $\mathbf{x}$  is a positive example.  
     FOREACH  $g \in G$  DO IF  $g(\mathbf{x}) \neq 1$  THEN  $G = G \setminus \{g\}$  ENDDO  
     FOREACH  $s \in S$  DO  
       IF  $s(\mathbf{x}) \neq 1$  THEN  
          $S = S \setminus \{s\}$ ,  $S^+ = \text{min\_generalizations}(s, \mathbf{x})$   
         FOREACH  $s \in S^+$  DO IF  $(\exists g \in G : g \geq_g s)$  THEN  $S = S \cup \{s\}$  ENDDO  
         FOREACH  $s \in S$  DO IF  $(\exists s' \in S : s' \neq s \wedge s \geq_g s')$  THEN  $S = S \setminus \{s\}$  ENDDO  
       ENDDO  
     ELSE //  $\mathbf{x}$  is a negative example.  
       FOREACH  $s \in S$  DO IF  $s(\mathbf{x}) \neq 0$  THEN  $S = S \setminus \{s\}$  ENDDO  
       FOREACH  $g \in G$  DO  
         IF  $g(\mathbf{x}) \neq 0$  THEN  
            $G = G \setminus \{g\}$ ,  $G^- = \text{min\_specializations}(g, \mathbf{x})$   
           FOREACH  $g \in G^-$  DO IF  $(\exists s \in S : g \geq_g s)$  THEN  $G = G \cup \{g\}$  ENDDO  
           FOREACH  $g \in G$  DO IF  $(\exists g' \in G : g' \neq g \wedge g' \geq_g g)$  THEN  $G = G \setminus \{g\}$  ENDDO  
         ENDDO  
       ENDIF  
     ENDDO  
  
3. return $(G, S)$ 
```

Concept Learning: Version Space

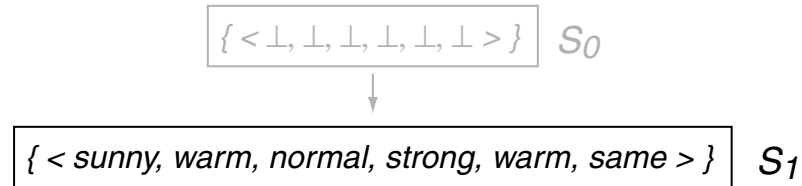
Illustration of the Candidate Elimination Algorithm

$$\boxed{\{ \langle \perp, \perp, \perp, \perp, \perp, \perp, \perp \rangle \}} S_0$$

$$\boxed{\{ \langle ?, ?, ?, ?, ?, ? \rangle \}} G_0,$$

Concept Learning: Version Space

Illustration of the Candidate Elimination Algorithm (continued)



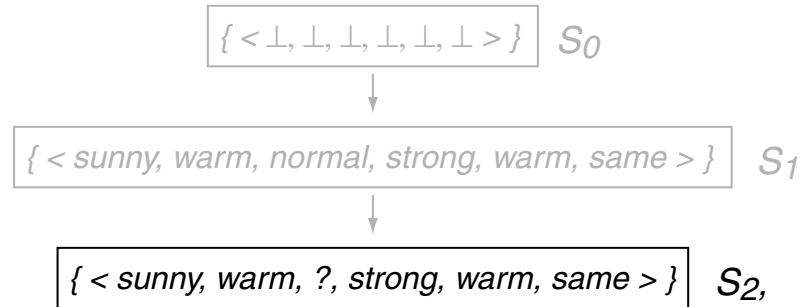
$$\{ \langle ?, ?, ?, ?, ?, ? \rangle \} \quad G_0, G_1,$$

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

$$\text{EnjoySport}(\mathbf{x}_1) = 1$$

Concept Learning: Version Space

Illustration of the Candidate Elimination Algorithm (continued)



$\{ \langle ?, ?, ?, ?, ?, ? \rangle \}$ G_0, G_1, G_2

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

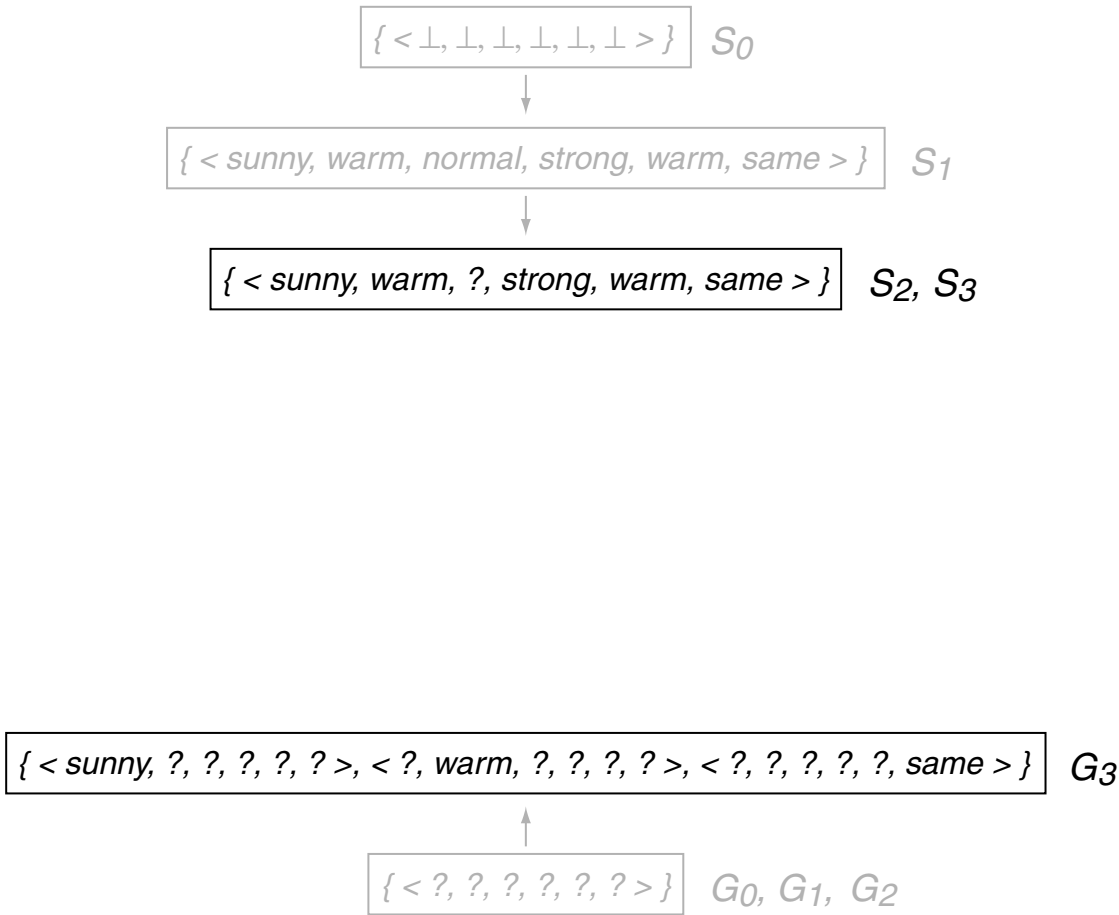
$\text{EnjoySport}(\mathbf{x}_1) = 1$

$\mathbf{x}_2 = (\text{sunny}, \text{warm}, \text{high}, \text{strong}, \text{warm}, \text{same})$

$\text{EnjoySport}(\mathbf{x}_2) = 1$

Concept Learning: Version Space

Illustration of the Candidate Elimination Algorithm (continued)



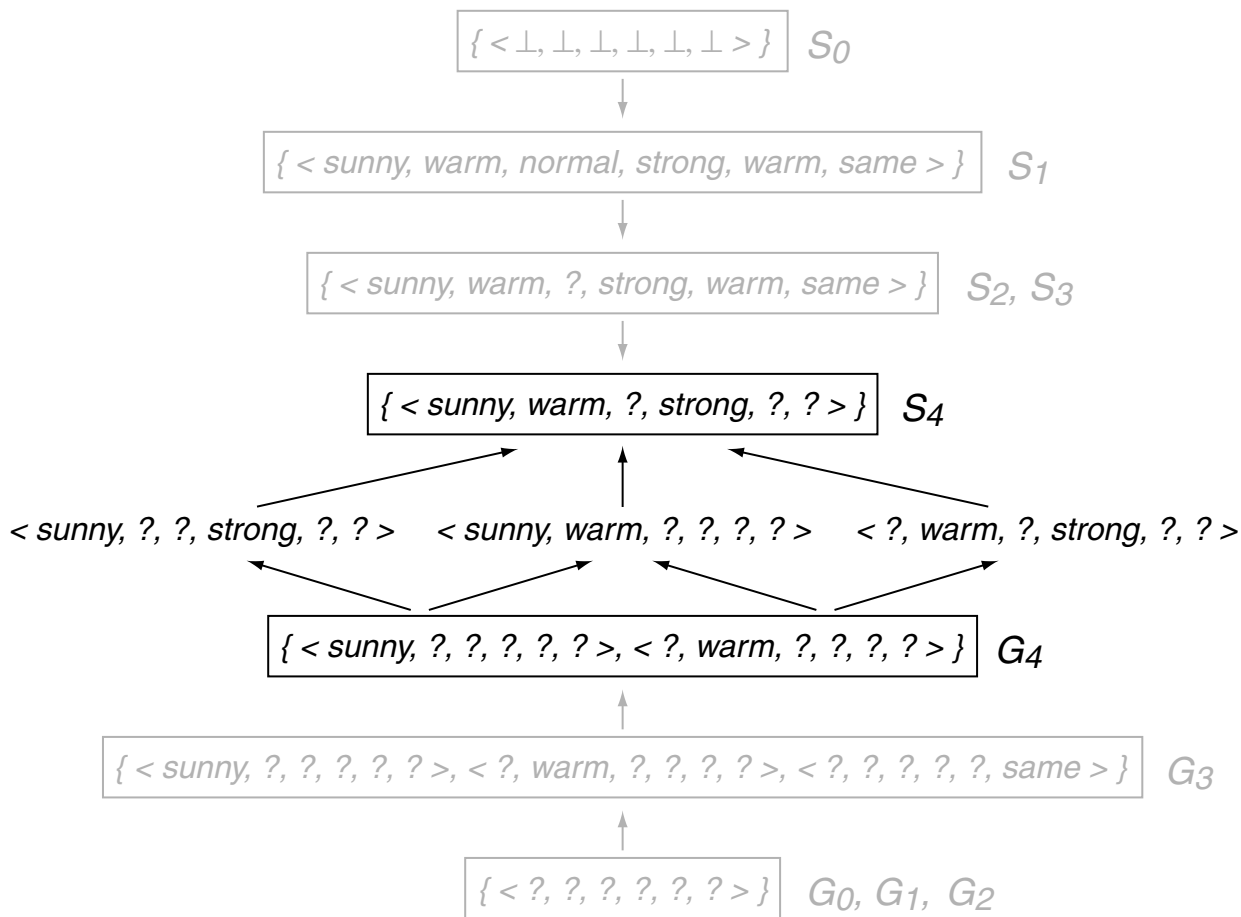
$\mathbf{x}_1 = (\text{sunny}, \text{warm}, \text{normal}, \text{strong}, \text{warm}, \text{same})$
 $\mathbf{x}_2 = (\text{sunny}, \text{warm}, \text{high}, \text{strong}, \text{warm}, \text{same})$
 $\mathbf{x}_3 = (\text{rainy}, \text{cold}, \text{high}, \text{strong}, \text{warm}, \text{change})$

$\text{EnjoySport}(\mathbf{x}_1) = 1$
 $\text{EnjoySport}(\mathbf{x}_2) = 1$
 $\text{EnjoySport}(\mathbf{x}_3) = 0$

[\[feature domains\]](#) [\[algorithm\]](#)

Concept Learning: Version Space

Illustration of the Candidate Elimination Algorithm (continued)



$\mathbf{x}_1 = (\text{sunny}, \text{warm}, \text{normal}, \text{strong}, \text{warm}, \text{same})$
 $\mathbf{x}_2 = (\text{sunny}, \text{warm}, \text{high}, \text{strong}, \text{warm}, \text{same})$
 $\mathbf{x}_3 = (\text{rainy}, \text{cold}, \text{high}, \text{strong}, \text{warm}, \text{change})$
 $\mathbf{x}_4 = (\text{sunny}, \text{warm}, \text{high}, \text{strong}, \text{cool}, \text{change})$

$\text{EnjoySport}(\mathbf{x}_1) = 1$
 $\text{EnjoySport}(\mathbf{x}_2) = 1$
 $\text{EnjoySport}(\mathbf{x}_3) = 0$
 $\text{EnjoySport}(\mathbf{x}_4) = 1$

[feature domains] [algorithm]

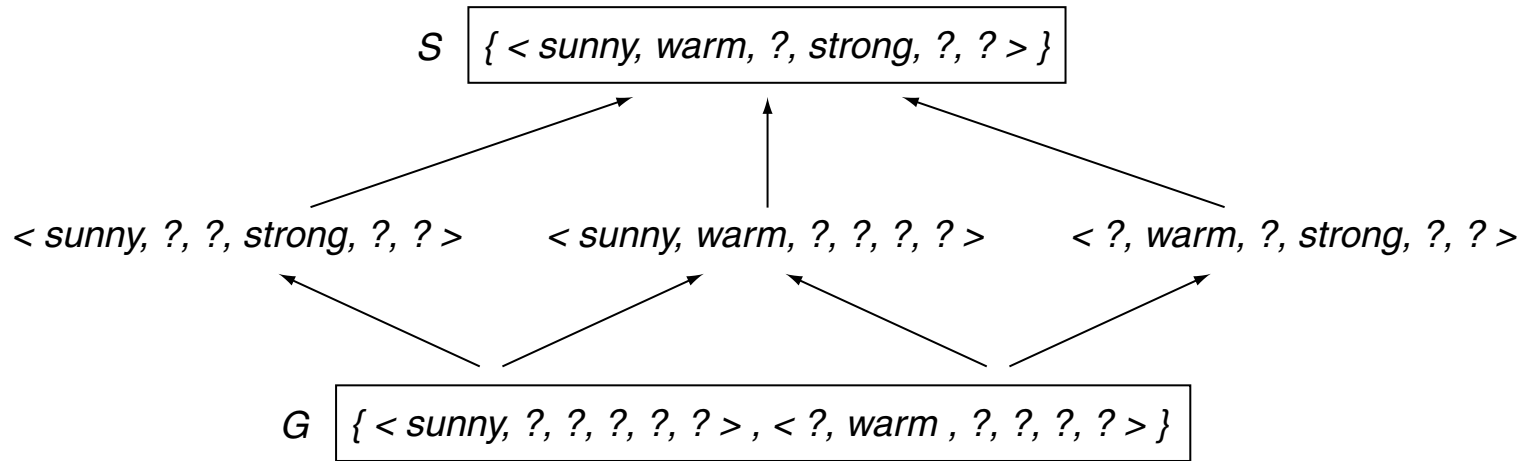
Concept Learning: Version Space

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?
Keywords: representation bias, search bias
4. Will Candidate Elimination converge towards the correct hypothesis?
5. When does one end up with an empty version space?

Concept Learning: Version Space

Question 1: Selecting Examples from D

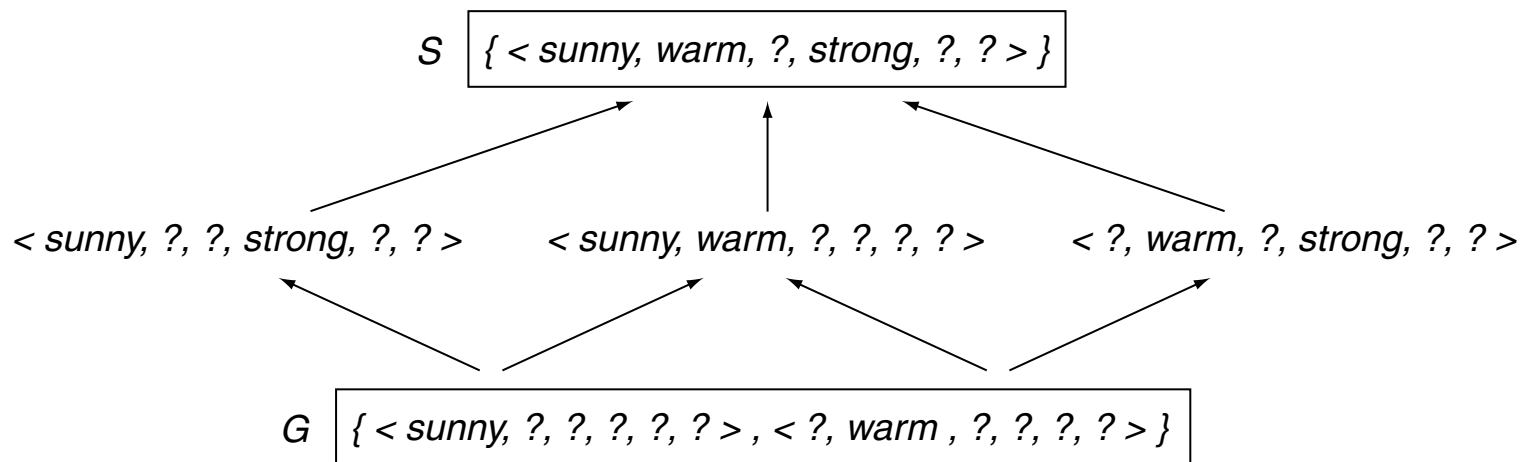


An example from which we can “maximally” learn:

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

Concept Learning: Version Space

Question 1: Selecting Examples from D (continued)



An example from which we can “maximally” learn:

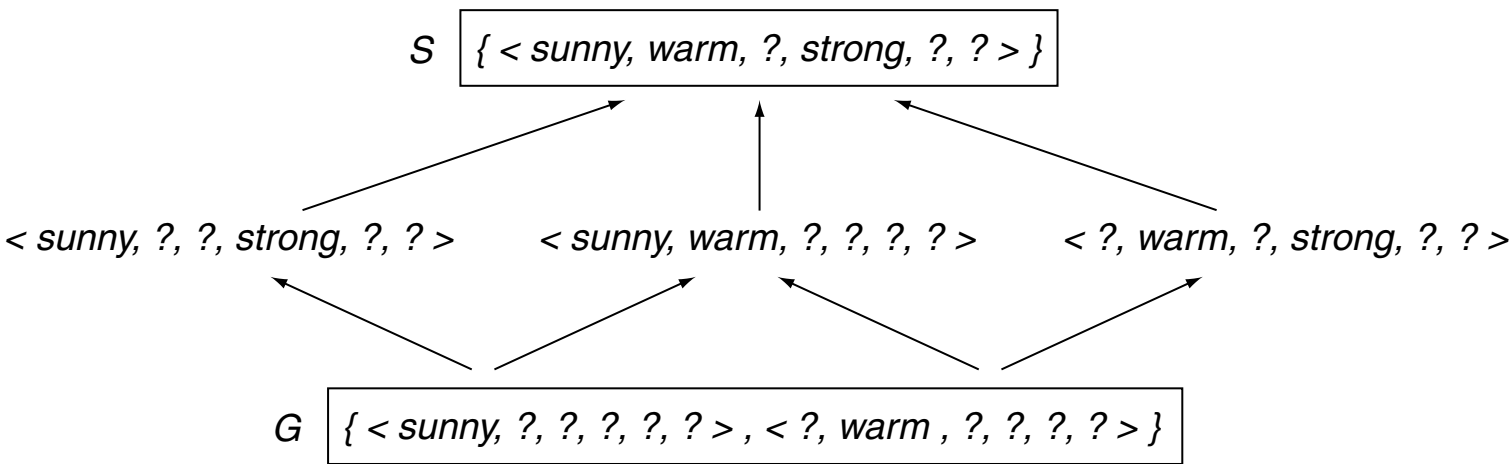
$\mathbf{x}_7 = (\text{sunny}, \text{warm}, \text{normal}, \text{light}, \text{warm}, \text{same})$

Irrespective the value of c , (\mathbf{x}_7, c) is consistent with 3 of the 6 hypotheses:

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

Concept Learning: Version Space

Question 2: Partially Learned Concepts

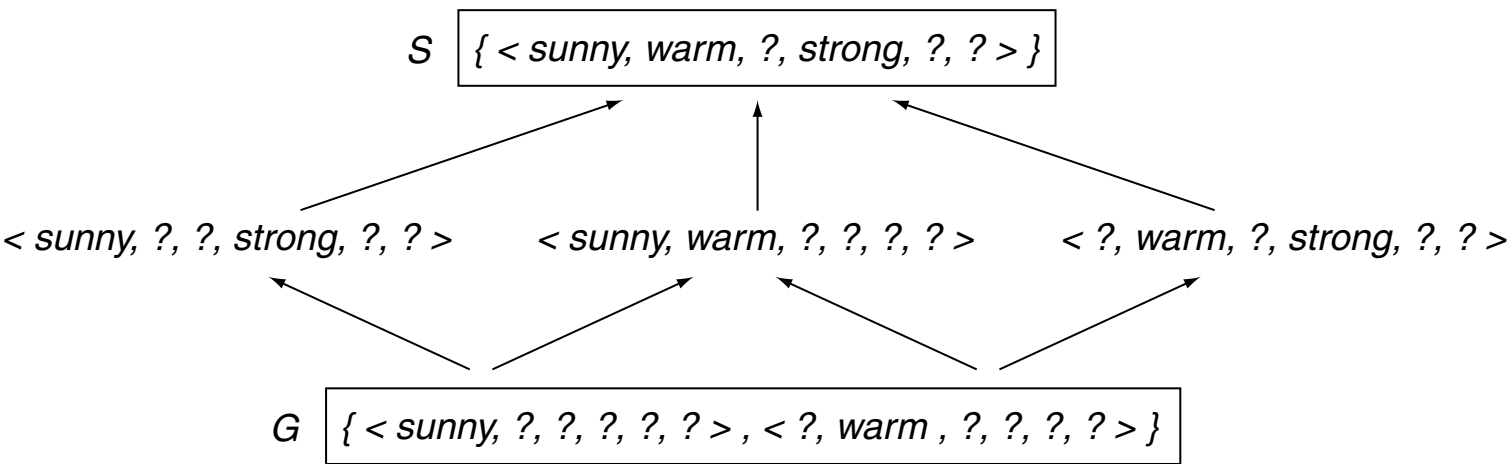


Combine the 6 classifiers in the version space to decide about new examples:

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	

Concept Learning: Version Space

Question 2: Partially Learned Concepts (continued)

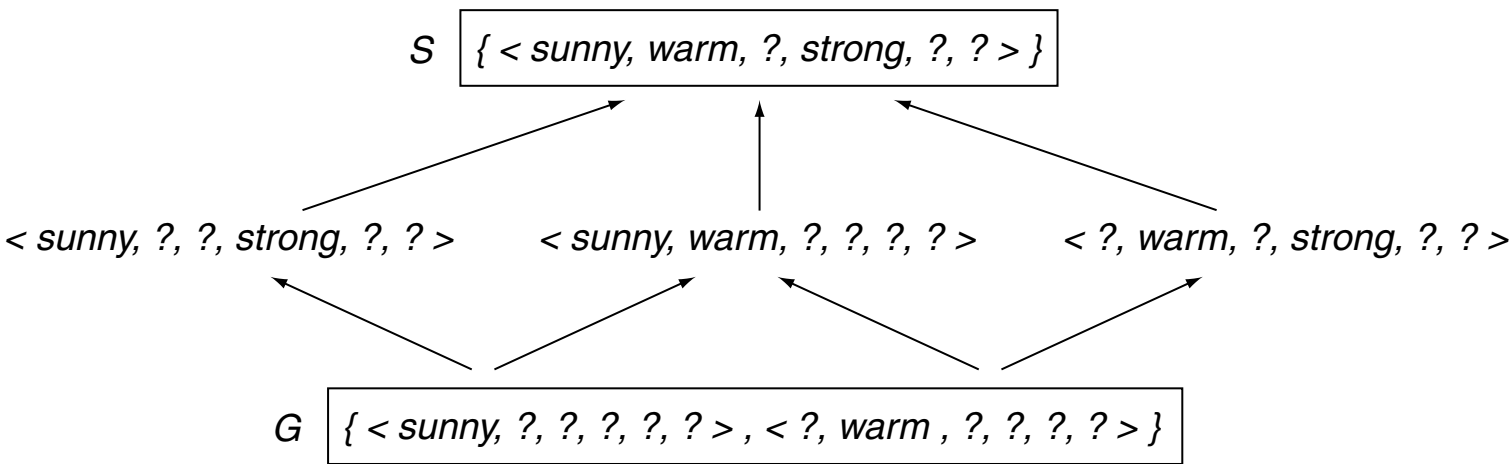


Combine the 6 classifiers in the version space to decide about new examples:

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	

Concept Learning: Version Space

Question 2: Partially Learned Concepts (continued)

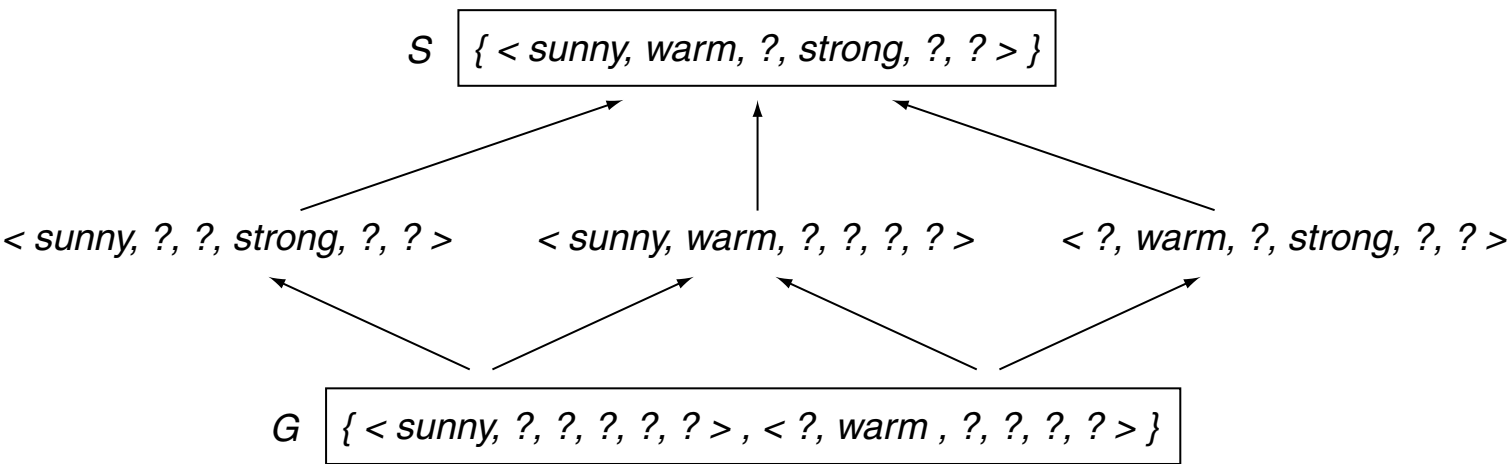


Combine the 6 classifiers in the version space to decide about new examples:

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	

Concept Learning: Version Space

Question 2: Partially Learned Concepts (continued)

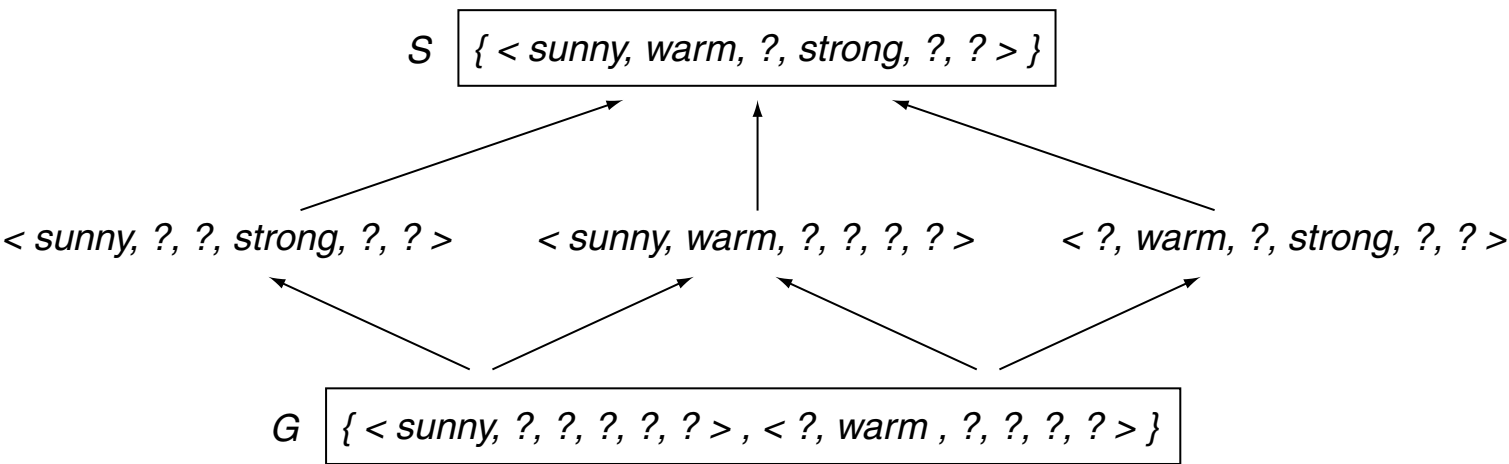


Combine the 6 classifiers in the version space to decide about new examples:

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	

Concept Learning: Version Space

Question 2: Partially Learned Concepts (continued)



Combine the 6 classifiers in the version space to decide about new examples:

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–

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

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

Concept Learning: Version Space

Question 3: Inductive Bias (continued)

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

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

\vdots

11	rainy	warm	normal	strong	cool	change	no
----	-------	------	--------	--------	------	--------	----

$\rightarrow S = \{ \}$

Discussion:

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

Concept Learning: Version Space

Question 3: Inductive Bias (continued)

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

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

⋮

11	rainy	warm	normal	strong	cool	change	no
----	-------	------	--------	--------	------	--------	----

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

Discussion:

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

→ H may be designed to contain more elaborate concepts:
 $\langle sunny, ?, ?, ?, ?, ? \rangle \vee \langle cloudy, ?, ?, ?, ?, ? \rangle$.

Concept Learning: Version Space

Question 3: Inductive Bias (continued)

“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.43, Mitchell 1997]

Concept Learning: Version Space

Question 3: Inductive Bias (continued)

- ❑ In a binary classification problem the unrestricted (= unbiased) hypothesis space contains $|\mathcal{P}(X)| = 2^{|X|}$ 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”.

Concept Learning: Version Space

Question 3: Inductive Bias (continued)

- ❑ In a binary classification problem the unrestricted (= unbiased) hypothesis space contains $|\mathcal{P}(X)| = 2^{|X|}$ 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”.
- 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?

Chapter ML:II

II. Machine Learning Basics

- ❑ Concept Learning: Search in Hypothesis Space
- ❑ Concept Learning: Version Space
- ❑ From Regression to Classification
- ❑ Evaluating Effectiveness

From Regression to Classification

Regression versus Classification

- X is a multiset 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
...	

From Regression to Classification

Regression versus Classification (continued)

- X is a multiset 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** for \mathbf{x}_i , $\mathbf{x}_i \in X$.
- $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\} \subseteq X \times \mathbf{R}$ is a multiset of examples.

From Regression to Classification

Regression versus Classification (continued)

- X is a multiset 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** for \mathbf{x}_i , $\mathbf{x}_i \in X$.
- $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\} \subseteq X \times \mathbf{R}$ is a multiset of examples.

Classification setting:

- $C = \{-1, 1\}$ is a set of two classes. Similarly: $\{0, 1\}$, $\{\ominus, \oplus\}$, $\{\text{no}, \text{yes}\}$, etc.
- c_i is the ground truth of the creditworthiness **class** for \mathbf{x}_i , $\mathbf{x}_i \in X$.
- $D = \{(\mathbf{x}_1, c_1), \dots, (\mathbf{x}_n, c_n)\} \subseteq X \times C$ is a multiset of examples.

From Regression to Classification

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 $\mathbf{w} = (w_0, w_1, \dots, w_p)^T$:

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

From Regression to Classification

The Linear Regression Model (continued)

- 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 $\mathbf{w} = (w_0, w_1, \dots, w_p)^T$:

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

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

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

From Regression to Classification

The Linear Regression Model (continued)

- 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 $\mathbf{w} = (w_0, w_1, \dots, w_p)^T$:

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

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

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

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

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

Remarks (residuals):

- ❑ A *residual* is the difference between a target value (ground truth, observation) 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 likelihood estimates.
- ❑ 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.
- ❑ Residual \neq Loss. Observe the subtle difference between the two concepts “residual” and “loss” (similarly: “error”, “cost”). The former denotes the difference between a target value (ground truth, observation) and its estimate, whereas the latter denotes the *interpretation of this difference*.

Remarks (randomness and distributions):

- ❑ The y_i are considered as realizations of n respective random variables Y_i . Btw., do not confuse the function $y()$ with a realization y_i .
- ❑ [Equation \(2\)](#): Estimating $\hat{\mathbf{w}}$ by RSS minimization is based on the following assumptions:
 1. The probability distributions of the Y_i have the same variance.
 2. The expectations $E[Y_i]$ of the 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 \mathbf{x}_i and the observed y_i can be explained completely by a linear model function.
 3. The random variables Y_i are statistically independent.

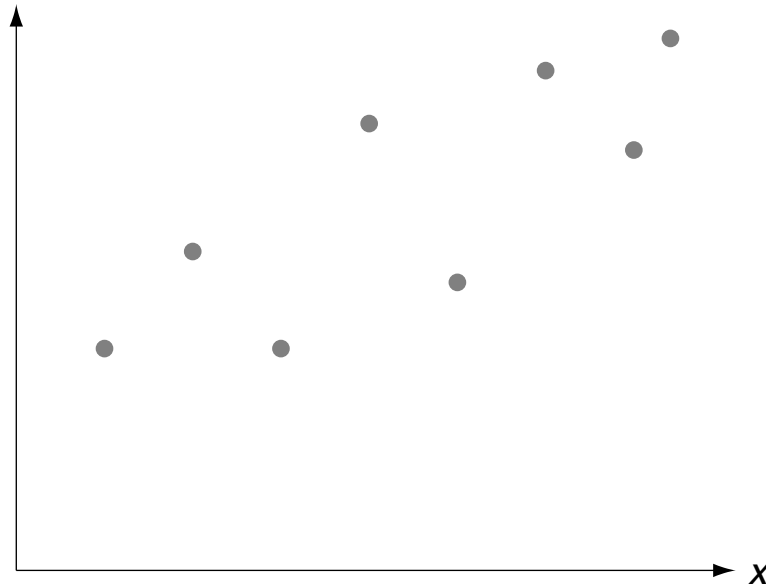
These assumptions are called the *weak set* (of assumptions). Along with a fourth assumption about the distribution shape of Y_i they become the *strong set* of assumptions.

- ❑ Y_i may also be defined as $y(\mathbf{x}_i) + E_i$, in which case the *disturbance term* E_i has the same distribution as Y_i but the mean 0 (while Y_i has the mean $\mathbf{w}^T \mathbf{x}_i$).
- ❑ Within the classical regression setting the variable \mathbf{x} , also called regressor, is a controlled variable. I.e., its instances \mathbf{x}_i , $i = 1, \dots, n$, are not considered as outcomes of a random experiment: the \mathbf{x}_i are given, chosen with intent, or constructed without any effect of chance.

Within the typical [machine learning setting](#), the occurrence of feature vectors—more general, the sample formation process underlying X —is governed by a probability distribution: certain observations may be more likely than others, and hence each feature vector \mathbf{x}_i is considered as the realization of a random vector \mathbf{X}_i .

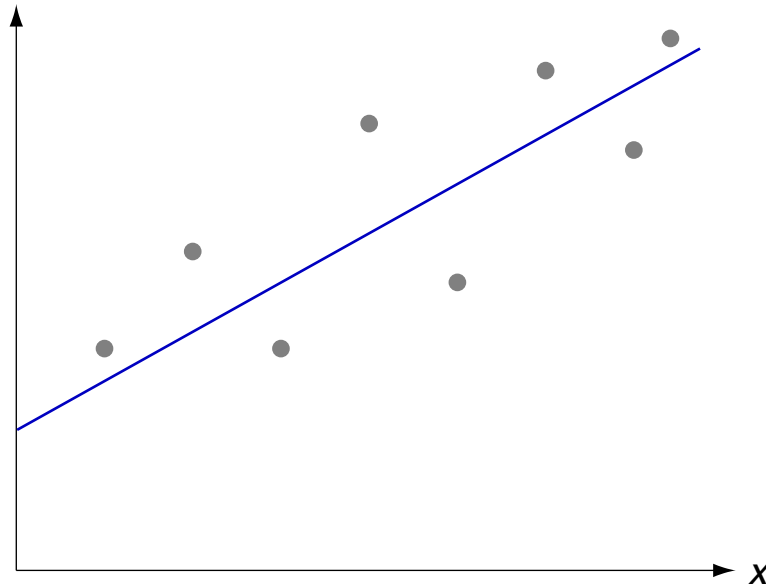
From Regression to Classification

One-Dimensional Feature Space



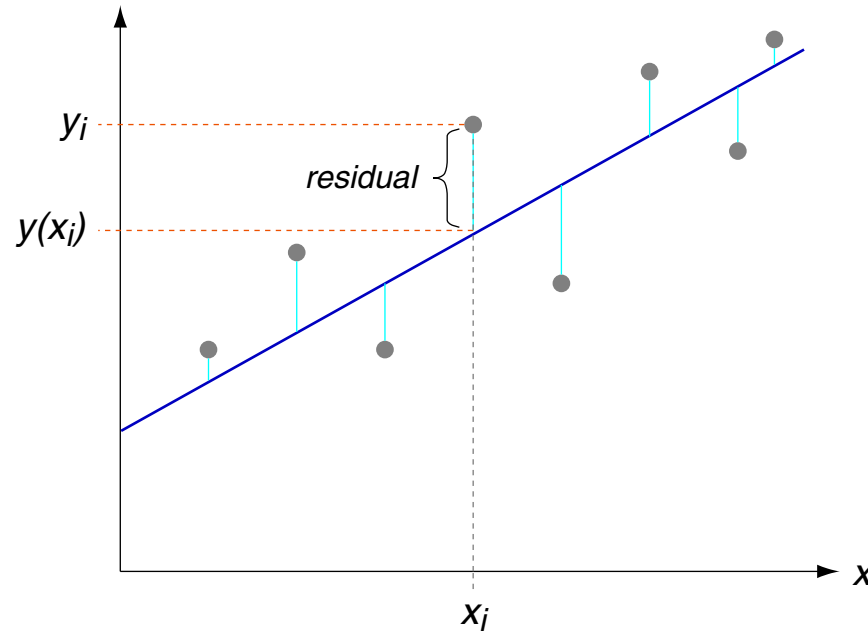
From Regression to Classification

One-Dimensional Feature Space (continued)



From Regression to Classification

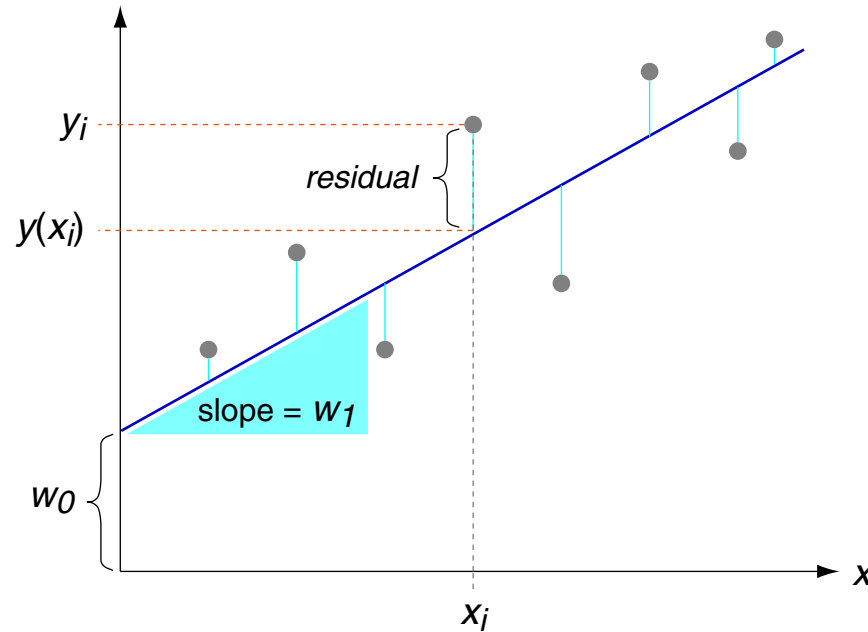
One-Dimensional Feature Space (continued)



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

From Regression to Classification

One-Dimensional Feature Space (continued)



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

From Regression to Classification

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

Minimize $\text{RSS}(w_0, w_1)$ via a direct method:

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

$$\leadsto \dots \leadsto w_0 = \frac{1}{n} \sum_{i=1}^n y_i - \frac{w_1}{n} \sum_{i=1}^n x_i = \bar{y} - w_1 \cdot \bar{x}$$

From Regression to Classification

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

Minimize $\text{RSS}(w_0, w_1)$ via a direct method:

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

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

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

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

From Regression to Classification

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

Minimize $\text{RSS}(w_0, w_1)$ via a direct method:

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

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

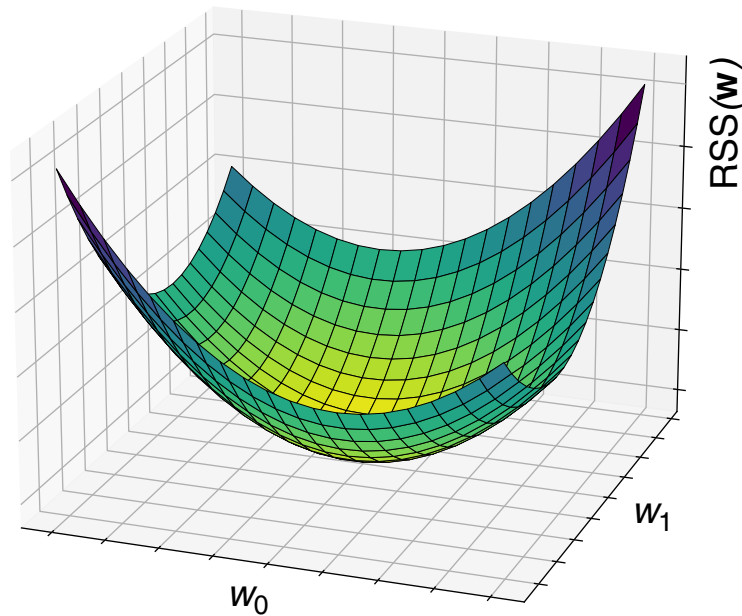
$$2. \quad \frac{\partial}{\partial w_1} \sum_{i=1}^n (y_i - w_0 - w_1 \cdot x_i)^2 = 0$$

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

From Regression to Classification

One-Dimensional Feature Space (continued)

Illustration of the task of minimizing $\text{RSS}(\mathbf{w}) = \sum_{i=1}^n (y_i - \mathbf{w}^T \mathbf{x}_i)^2$.



From Regression to Classification

Higher-Dimensional Feature Space

- Recall Equation (1) :

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

- 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 .

From Regression to Classification

Higher-Dimensional Feature Space (continued)

- Recall Equation (1) :

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

- 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 .

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

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

From Regression to Classification

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

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

$$\frac{\partial \text{RSS}}{\partial \mathbf{w}} = -2X^T(\mathbf{y} - X\mathbf{w}) = 0, \quad \frac{\partial^2 \text{RSS}}{\partial \mathbf{w} \partial \mathbf{w}^T} = -2X^T X \quad [\text{Wikipedia } \underline{1}, \underline{2}, \underline{3}]$$

$$X^T(\mathbf{y} - X\mathbf{w}) = 0$$

$$\Leftrightarrow X^T X \mathbf{w} = X^T \mathbf{y}$$

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

From Regression to Classification

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

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

$$\frac{\partial \text{RSS}}{\partial \mathbf{w}} = -2X^T(\mathbf{y} - X\mathbf{w}) = 0, \quad \frac{\partial^2 \text{RSS}}{\partial \mathbf{w} \partial \mathbf{w}^T} = -2X^T X \quad [\text{Wikipedia } \underline{1}, \underline{2}, \underline{3}]$$

$$X^T(\mathbf{y} - X\mathbf{w}) = 0$$

$$\Leftrightarrow X^T X \mathbf{w} = X^T \mathbf{y}$$

$$\leadsto \mathbf{w} = \underbrace{(X^T X)^{-1} X^T}_{\text{Pseudoinverse of } X} \mathbf{y}$$

Pseudoinverse of X [\[Wikipedia\]](#)

Normal equations.

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

From Regression to Classification

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

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

$$\frac{\partial \text{RSS}}{\partial \mathbf{w}} = -2X^T(\mathbf{y} - X\mathbf{w}) = 0, \quad \frac{\partial^2 \text{RSS}}{\partial \mathbf{w} \partial \mathbf{w}^T} = -2X^T X \quad [\text{Wikipedia } \underline{1}, \underline{2}, \underline{3}]$$

$$X^T(\mathbf{y} - X\mathbf{w}) = 0$$

$$\Leftrightarrow X^T X \mathbf{w} = X^T \mathbf{y}$$

$$\leadsto \hat{\mathbf{w}} \equiv \mathbf{w} = \underbrace{(X^T X)^{-1} X^T}_{\text{Pseudoinverse of } X} \mathbf{y}$$

Pseudoinverse of X [\[Wikipedia\]](#)

Normal equations.

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

$$\hat{y}(\mathbf{x}_i) = \hat{\mathbf{w}}^T \mathbf{x}_i \quad \text{Regression function with least squares estimator } \hat{\mathbf{w}}.$$

$$\begin{aligned} \hat{\mathbf{y}} &= X \hat{\mathbf{w}} && \text{The } n\text{-vector of fitted values at the training input.} \\ &= X(X^T X)^{-1} X^T \mathbf{y} \end{aligned}$$

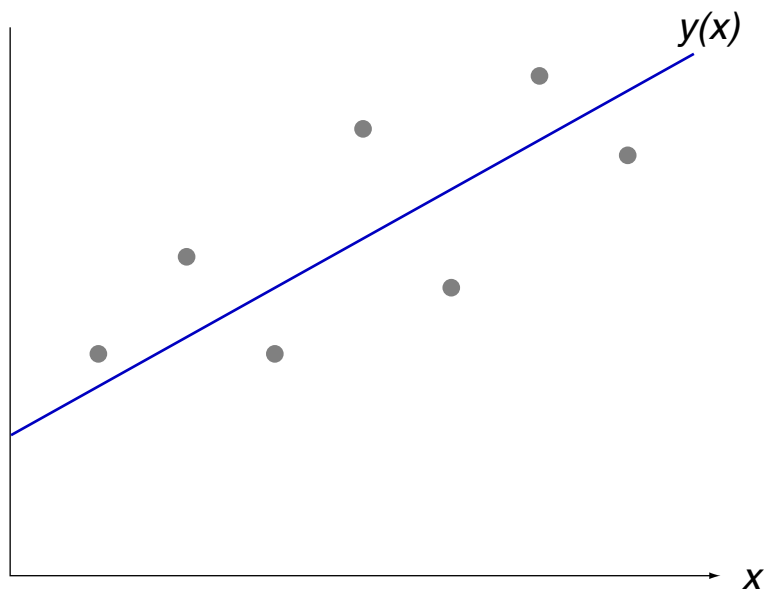
Remarks:

- ❑ 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.

From Regression to Classification

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

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



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

From Regression to Classification

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

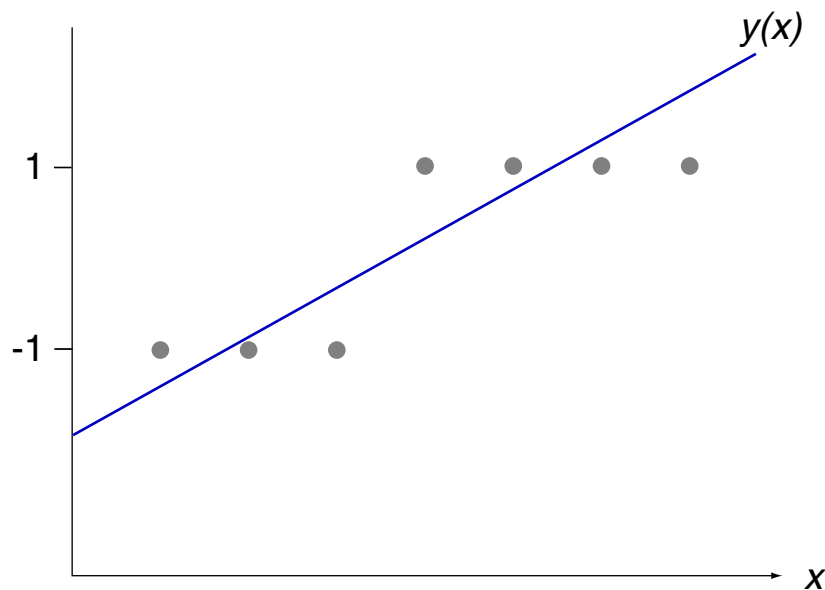
Binary-valued (± 1) functions are also real-valued.



From Regression to Classification

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

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

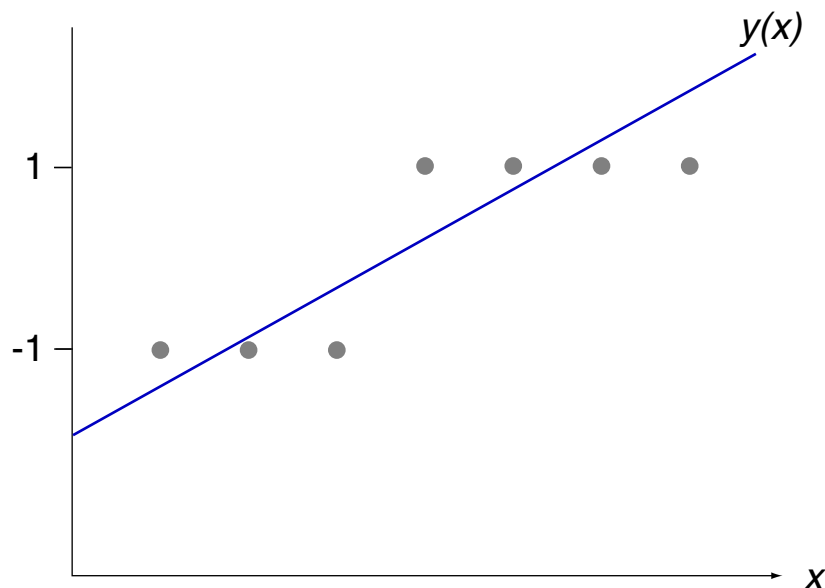


$$y(x) = (w_0 \ w_1) \begin{pmatrix} 1 \\ x \end{pmatrix}$$

From Regression to Classification

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

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



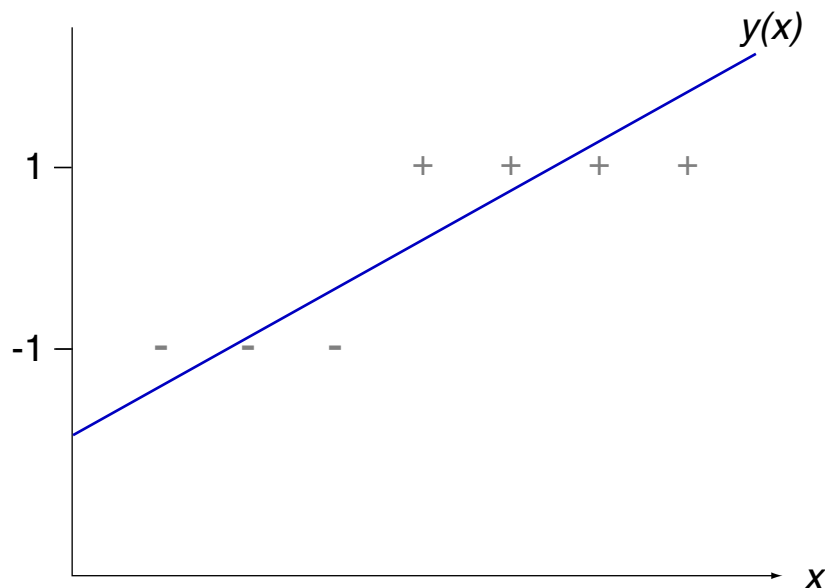
The function “ $\text{sign}(\mathbf{w}^T \mathbf{x}_i)$ ” is likely to agree with $y_i = \pm 1$.

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

From Regression to Classification

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

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



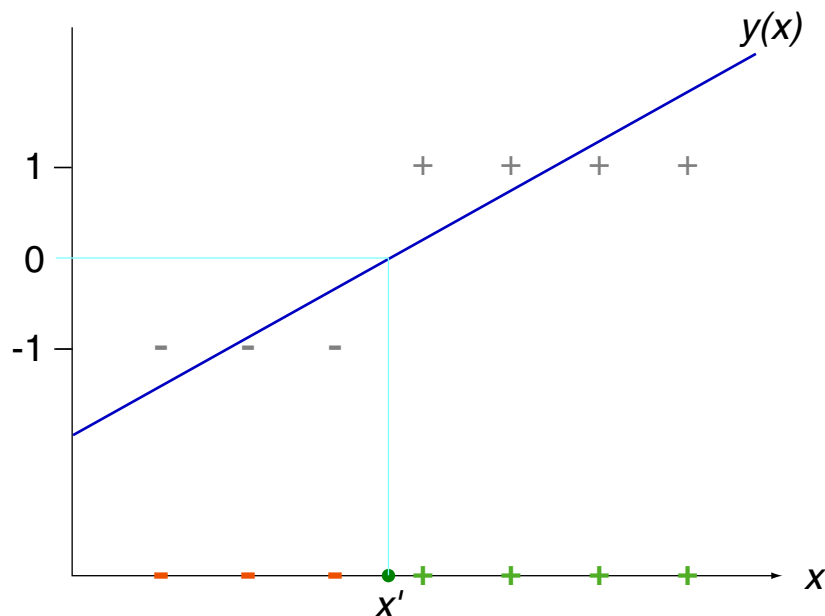
The function “ $\text{sign}(\mathbf{w}^T \mathbf{x}_i)$ ” is likely to agree with $y_i = \pm 1$.

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

From Regression to Classification

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

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



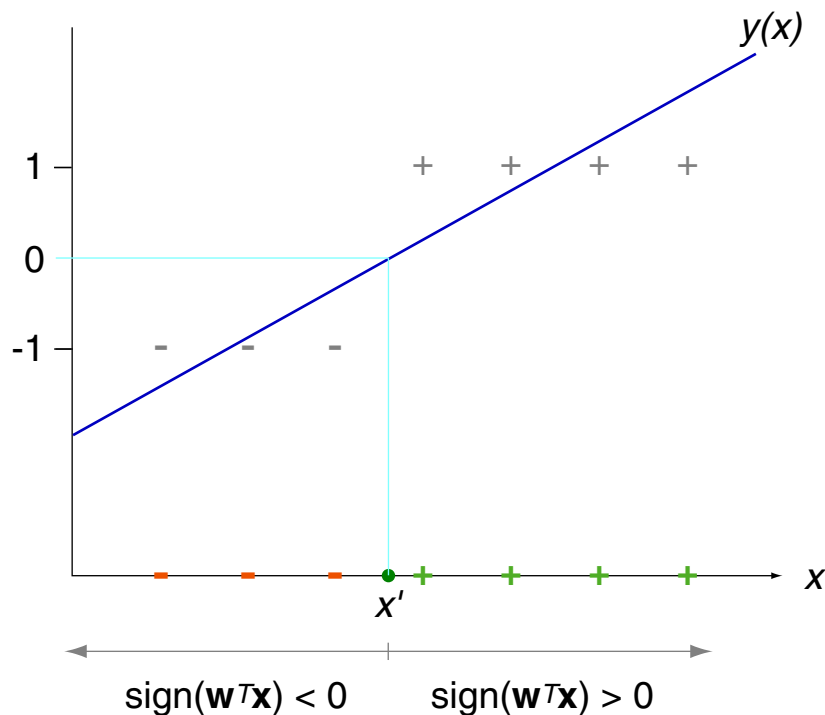
The function “ $\text{sign}(\mathbf{w}^T \mathbf{x}_i)$ ” is likely to agree with $y_i = \pm 1$.

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

From Regression to Classification

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

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



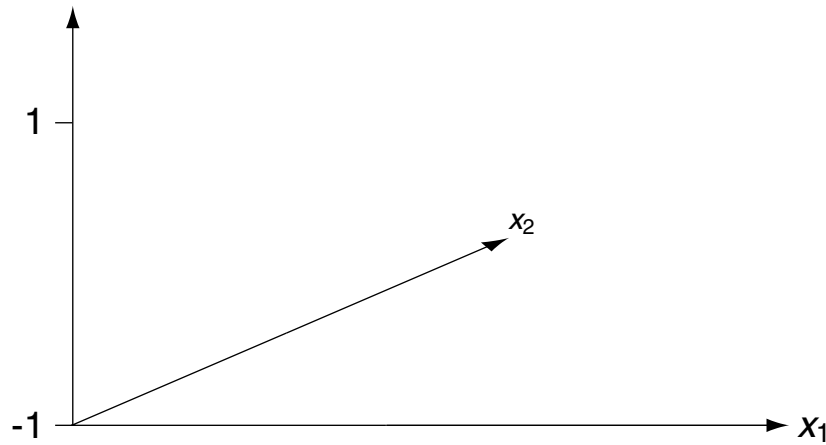
- ❑ The discrimination point, \bullet , is defined by $w_0 + w_1 \cdot x' = 0$.
- ❑ For $p = 2$ we are given a discrimination *line*.

Remarks:

- (★) We consider the feature vector \mathbf{x} in its extended form when used as operand in a scalar product with the weight vector, $\mathbf{w}^T \mathbf{x}$, and consequently, when noted as argument of the model function, $y(\mathbf{x})$. I.e., $\mathbf{x} = (1, x_1, \dots, x_p)^T \in \mathbf{R}^{p+1}$, and $x_0 = 1$.
- The [sign function](#) is three-valued, with $\text{sign}(z) = -1$ ($0, 1$) for $z < 0$ ($z = 0, z > 0$)—i.e., the case with $\mathbf{w}^T \mathbf{x} = 0$ needs special treatment. Without loss of generality we will label $y(0)$ with the “positive” class ($1, \oplus$, yes, etc.) and define $\text{sign}(0) = 1$ in the respective algebraic expressions.

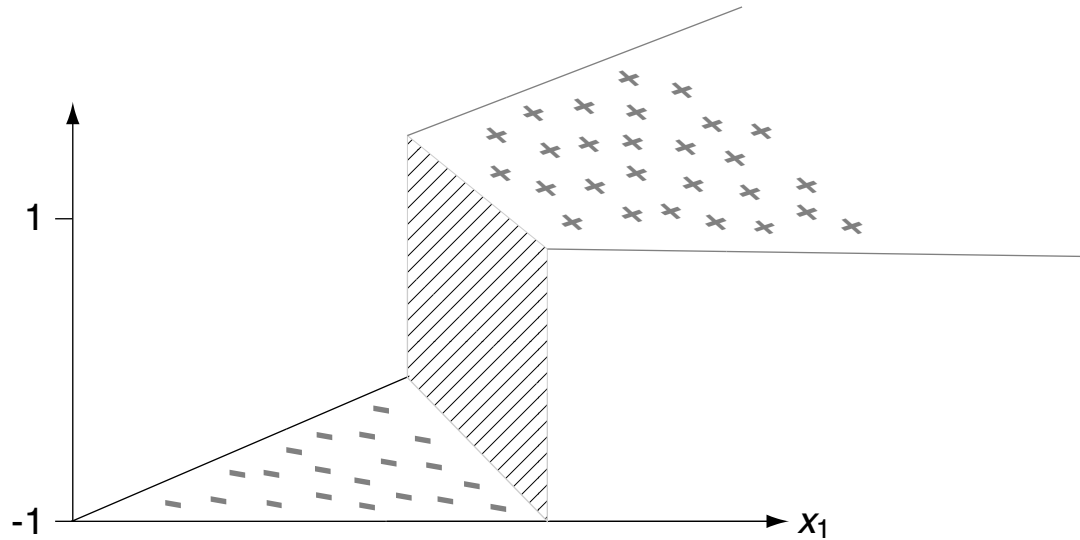
From Regression to Classification

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



From Regression to Classification

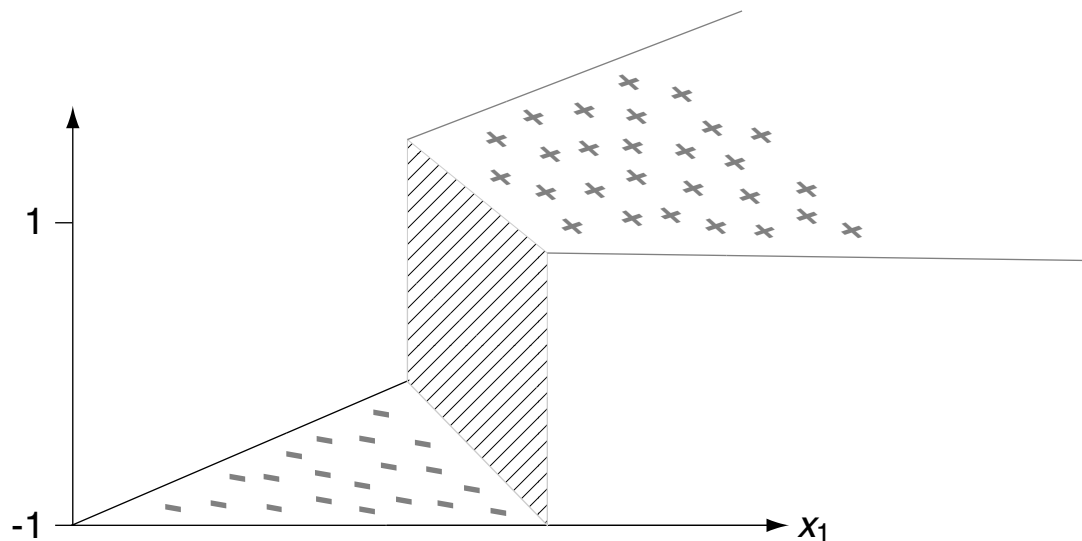
Linear Regression for Classification (illustrated for $p = 2$) (continued)



From Regression to Classification

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

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

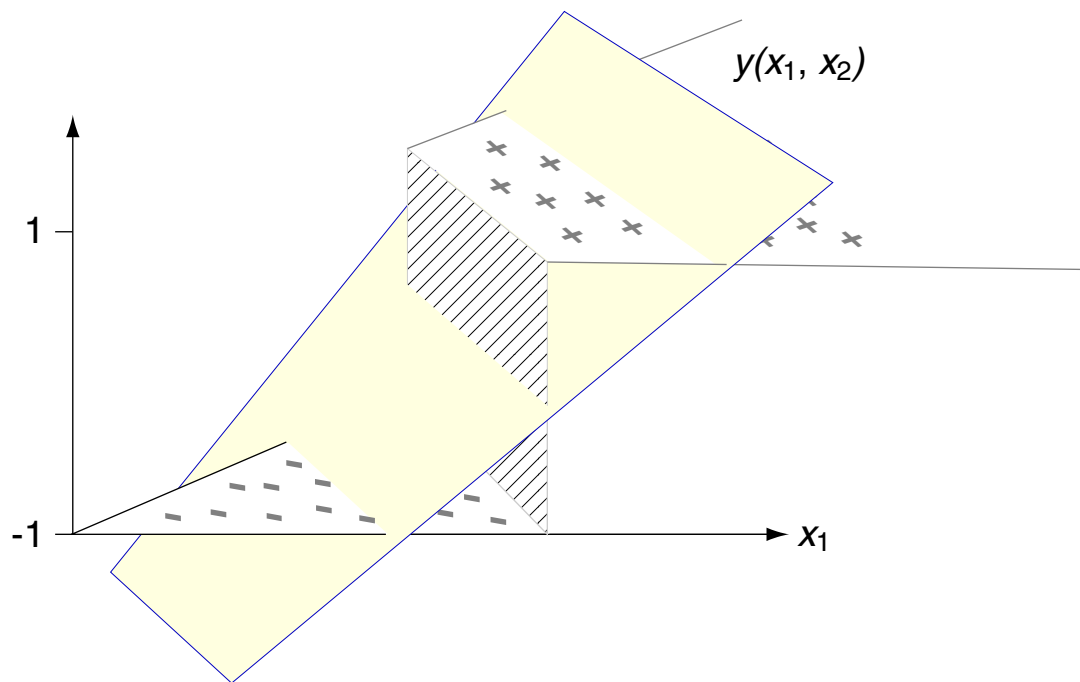


$$y(x_1, x_2) = (w_0 \ w_1 \ w_2) \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}$$

From Regression to Classification

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

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



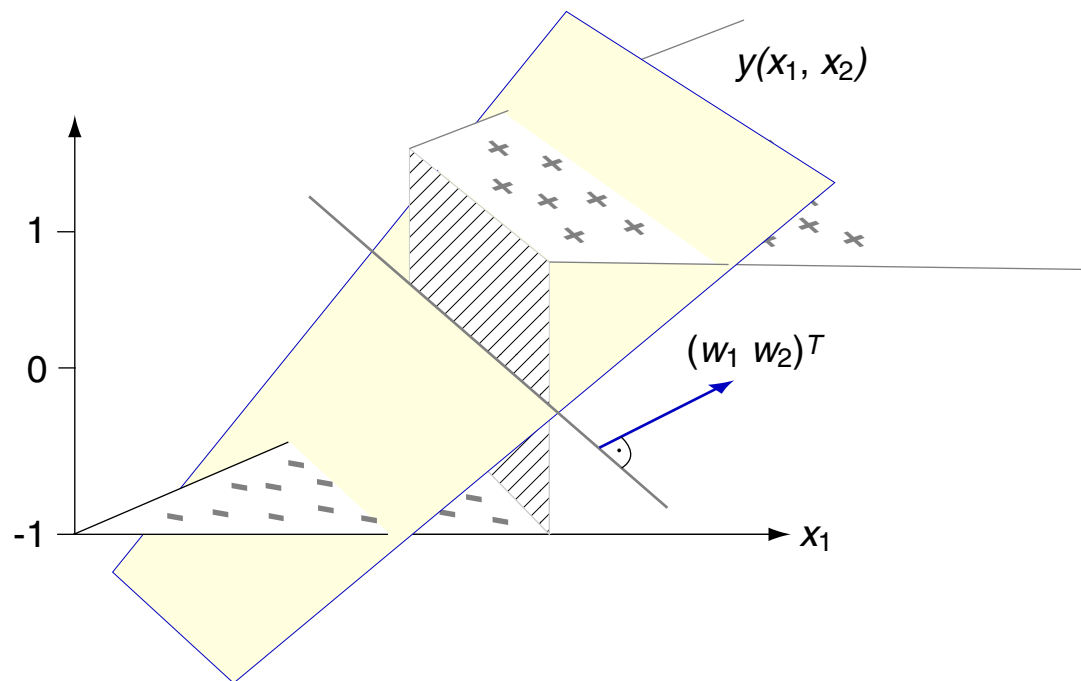
The function “ $\text{sign}(\mathbf{w}^T \mathbf{x}_i)$ ” is likely to agree with $y_i = \pm 1$.

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

From Regression to Classification

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

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



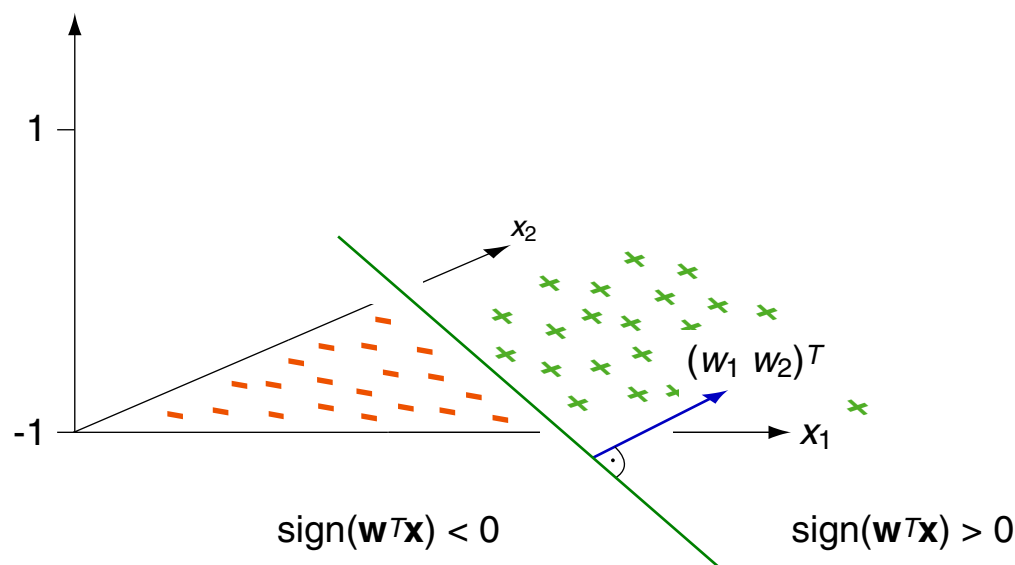
The function “ $\text{sign}(\mathbf{w}^T \mathbf{x}_i)$ ” is likely to agree with $y_i = \pm 1$.

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

From Regression to Classification

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

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

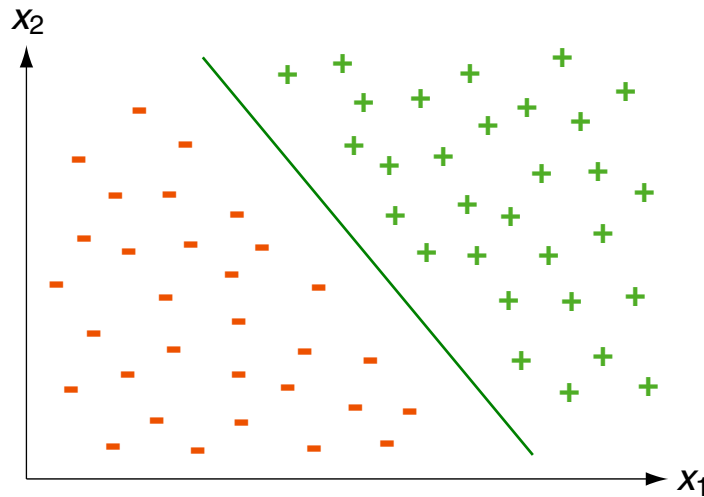


- The discrimination line, —, is defined by $w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 = 0$.
- For $p = 3$ ($p > 3$) we are given a discriminating (hyper)plane.

From Regression to Classification

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

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



- 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.

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.

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, introduced in the part Linear Models, 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} . The shown figures illustrate how (linear) regression methods that are applied in the input-output space implicitly define a hyperplane in the input space.

An illustration of the input-output space of the logistic regression model along with the implicitly defined hyperplane is shown [here](#).

From Regression to Classification

Linear Model Function Variants

The components (features) of the input vector $\mathbf{x} = (x_1, \dots, x_p)$ can 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 , $g \in \{1, \dots, p\}$, as $x_j = I(g = j)$
5. interactions between variables, such as $x_3 = x_1 \cdot x_2$

From Regression to Classification

Linear Model Function Variants (continued)

The components (features) of the input vector $\mathbf{x} = (x_1, \dots, x_p)$ can 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 $g, g \in \{1, \dots, p\}$, 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} :

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

From Regression to Classification

Linear Model Function Variants (continued)

The components (features) of the input vector $\mathbf{x} = (x_1, \dots, x_p)$ can 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 $g, g \in \{1, \dots, p\}$, 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: $y(\mathbf{w})$ is a linear function

From Regression to Classification

Linear Model Function Variants (continued)

The components (features) of the input vector $\mathbf{x} = (x_1, \dots, x_p)$ can 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 $g, g \in \{1, \dots, p\}$, as $x_j = I(g = j)$
5. interactions between variables, such as $x_3 = x_1 \cdot x_2$

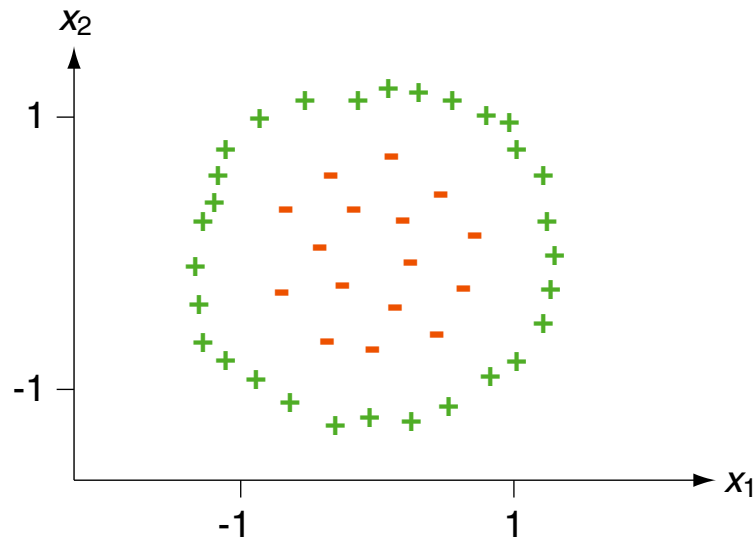
No matter the source of the x_j , the model is still linear in its parameters \mathbf{w} :

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

- linear in the parameters: $y(\mathbf{w})$ is a linear function
- basis functions: input variables (space) become(s) feature variables (space)

From Regression to Classification

Non-Linear Decision Boundaries [logistic regression]

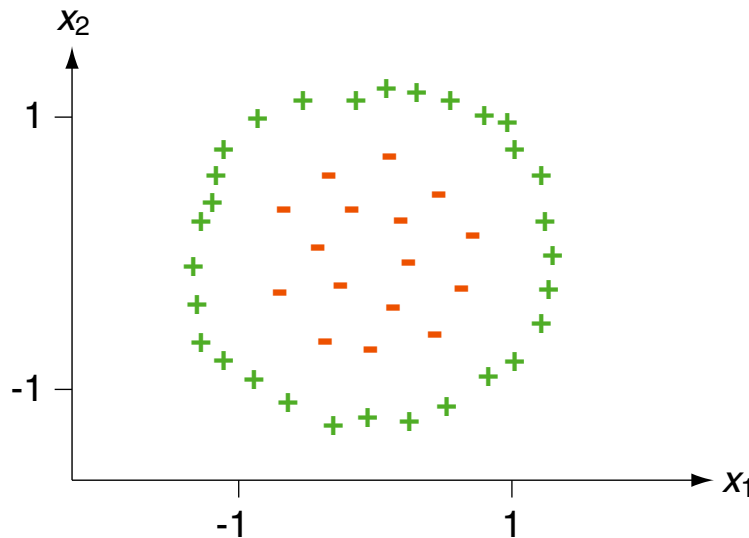


Higher order polynomial terms in the features (linear in the parameters):

$$y(\mathbf{x}) = w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_1^2 + w_4 \cdot x_2^2$$

From Regression to Classification

Non-Linear Decision Boundaries (continued) [logistic regression]

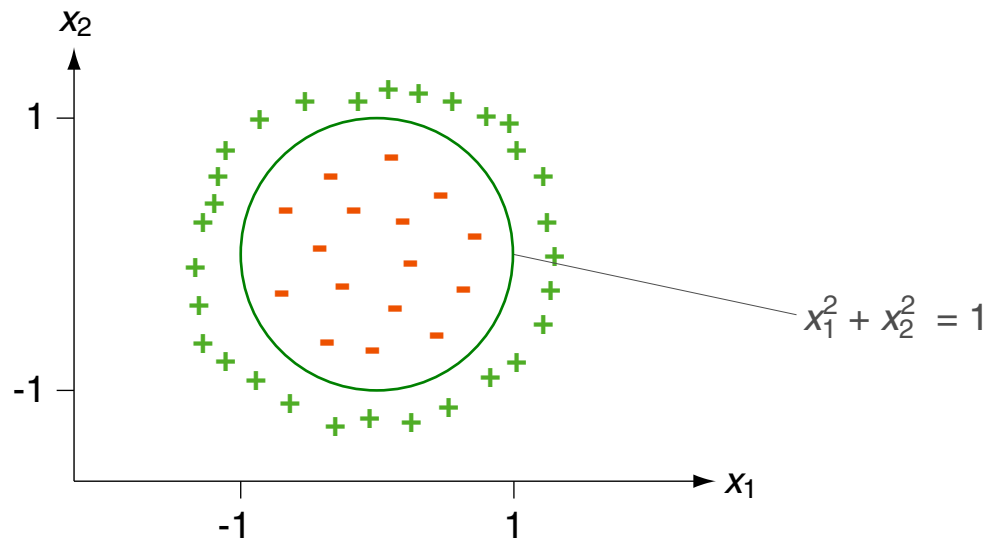


Higher order polynomial terms in the features (linear in the parameters):

$$y(\mathbf{x}) = w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_1^2 + w_4 \cdot x_2^2$$

From Regression to Classification

Non-Linear Decision Boundaries (continued) [logistic regression]



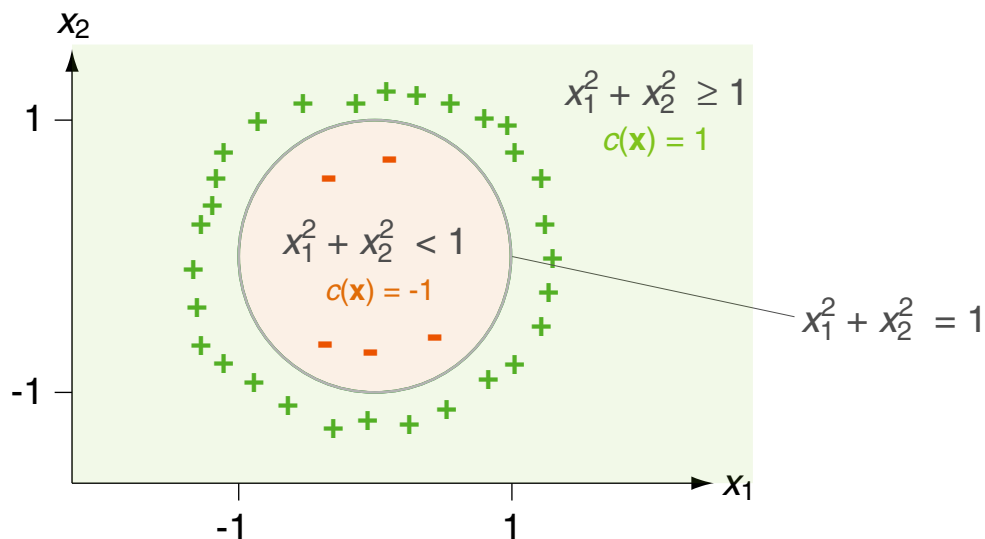
Higher order polynomial terms in the features (linear in the parameters):

$$y(\mathbf{x}) = w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_1^2 + w_4 \cdot x_2^2$$

$$\text{with } \mathbf{w} = \begin{pmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \leadsto y(\mathbf{x}) = -1 + x_1^2 + x_2^2$$

From Regression to Classification

Non-Linear Decision Boundaries (continued) [logistic regression]



Higher order polynomial terms in the features (linear in the parameters):

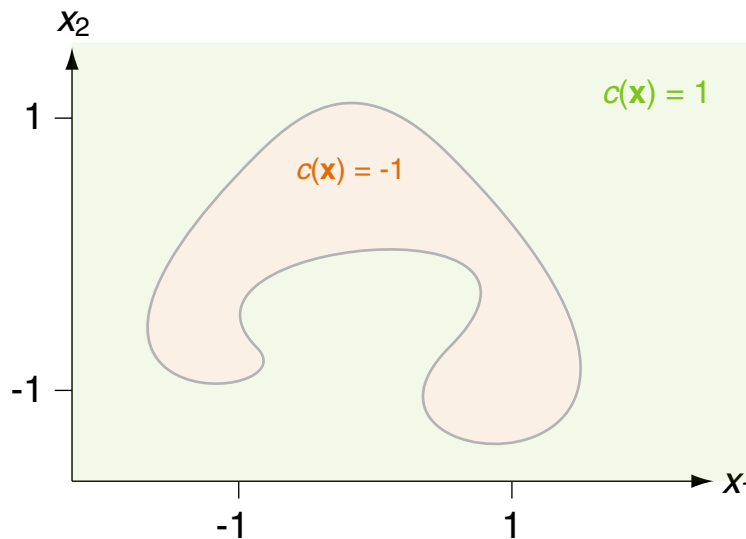
$$y(\mathbf{x}) = w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_1^2 + w_4 \cdot x_2^2$$

with $\mathbf{w} = \begin{pmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \leadsto y(\mathbf{x}) = -1 + x_1^2 + x_2^2$

Classification: Predict $\begin{cases} c = 1, & \text{if } x_1^2 + x_2^2 \geq 1 & \Leftrightarrow \mathbf{w}^T \mathbf{x} \geq 0 \\ c = -1, & \text{if } x_1^2 + x_2^2 < 1 & \Leftrightarrow \mathbf{w}^T \mathbf{x} < 0 \end{cases}$

From Regression to Classification

Non-Linear Decision Boundaries (continued) [logistic regression]



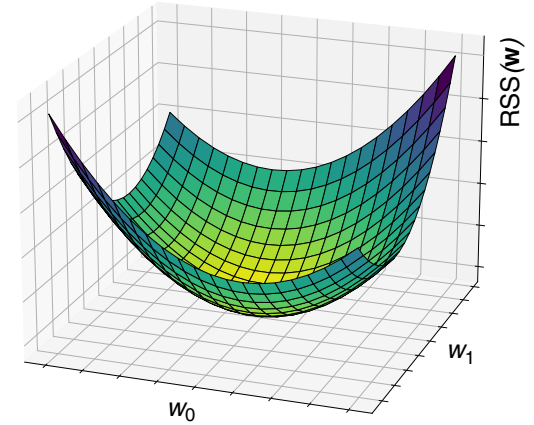
More complex polynomials entail more complex decision boundaries:

$$y(\mathbf{x}) = w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_1^2 + w_4 \cdot x_1^2 \cdot x_2 + w_5 \cdot x_1^2 \cdot x_2^2 + \dots$$

From Regression to Classification

Methods of Least Squares: Iterative versus Direct Methods

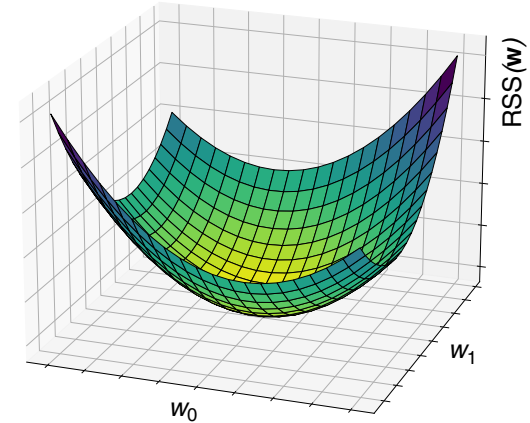
$$\operatorname{argmin}_{\mathbf{w}} \operatorname{RSS}(\mathbf{w}), \quad \text{with } \operatorname{RSS}(\mathbf{w}) = \sum_{i=1}^n (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$



From Regression to Classification

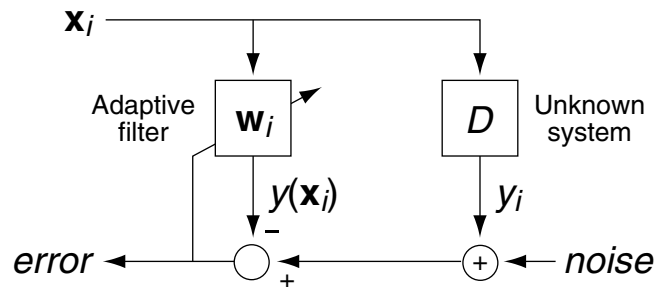
Methods of Least Squares: Iterative versus Direct Methods (continued)

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



LMS algorithm:

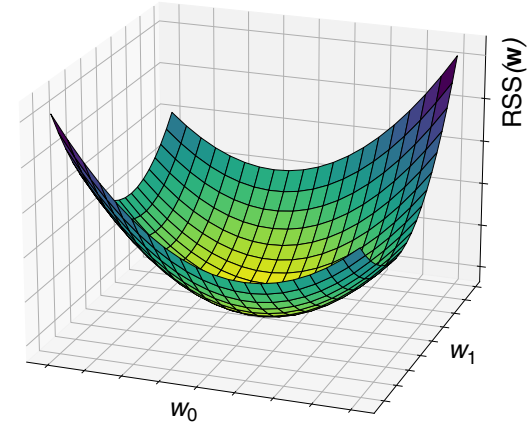
- ❑ applicable as online algorithm
- ❑ robust algorithm structure
- ❑ unsatisfactory convergence
- ❑ allows stochastic sampling



From Regression to Classification

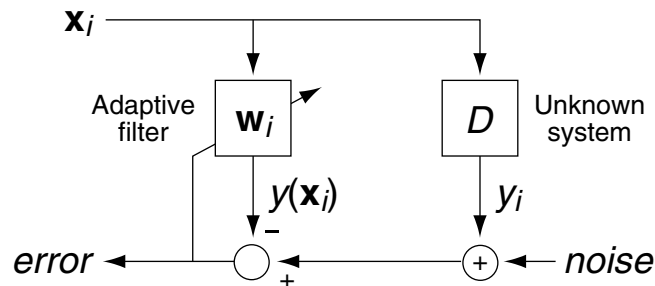
Methods of Least Squares: Iterative versus Direct Methods (continued)

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



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

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

Remarks:

- ❑ 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 zero-one-loss is non-differentiable.
- ❑ For a linear model function $y(\mathbf{x})$, $\text{RSS}(\mathbf{w})$ is a convex function and hence a single, global optimum exists.
- ❑ 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. Both topics are treated in the part Linear Models.
- ❑ Regularization will introduce additional constraints for the model function $y(\mathbf{x})$ or the parameter vector \mathbf{w} . With regularization the minimization expression (2) will have 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} .

From Regression to Classification

Properties of the Solution

Theorem 1 (Gauss-Markov)

Let $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ be a multiset of examples to be fitted with a linear model function as $y(\mathbf{x}) \stackrel{(\star)}{=} \mathbf{x}^T \mathbf{w}$. 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.

From Regression to Classification

Properties of the Solution (continued)

Theorem 1 (Gauss-Markov)

Let $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ be a multiset of examples to be fitted with a linear model function as $y(\mathbf{x}) \stackrel{(*)}{=} \mathbf{x}^T \mathbf{w}$. 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 between least squares and maximum likelihood estimators / methods

Remarks:

- ❑ The Gauss-Markov Theorem is important since it follows already from the weak set of assumptions.
- ❑ Under the strong set of assumptions the maximum likelihood estimates are identical to the least-squares estimates.

Chapter ML:II (continued)

II. Machine Learning Basics

- ❑ Concept Learning: Search in Hypothesis Space
- ❑ Concept Learning: Version Space
- ❑ From Regression to Classification
- ❑ Evaluating Effectiveness

Evaluating Effectiveness

Misclassification Rate

Definition 8 (True Misclassification Rate / True Error of a Classifier y)

Let O be a finite set of objects, \mathbf{X} the feature space associated with a model formation function $\alpha : O \rightarrow \mathbf{X}$, C a set of classes, $y : \mathbf{X} \rightarrow C$ a classifier, and $\gamma : O \rightarrow C$ the ideal classifier to be approximated by y .

Let $X = \{\mathbf{x} \mid \mathbf{x} = \alpha(o), o \in O\}$ be a multiset of feature vectors and $c_{\mathbf{x}} = \gamma(o)$, $o \in O$. Then, the true misclassification rate of $y(\mathbf{x})$, denoted $Err^*(y)$, is defined as follows:

$$\underline{Err^*(y)} = \frac{|\{\mathbf{x} \in X : y(\mathbf{x}) \neq c_{\mathbf{x}}\}|}{|X|} = \frac{|\{o \in O : y(\alpha(o)) \neq \gamma(o)\}|}{|O|}$$

Evaluating Effectiveness

Misclassification Rate (continued)

Definition 8 (True Misclassification Rate / True Error of a Classifier y)

Let O be a finite set of objects, \mathbf{X} the feature space associated with a model formation function $\alpha : O \rightarrow \mathbf{X}$, C a set of classes, $y : \mathbf{X} \rightarrow C$ a classifier, and $\gamma : O \rightarrow C$ the ideal classifier to be approximated by y .

Let $X = \{\mathbf{x} \mid \mathbf{x} = \alpha(o), o \in O\}$ be a multiset of feature vectors and $c_{\mathbf{x}} = \gamma(o)$, $o \in O$. Then, the true misclassification rate of $y(\mathbf{x})$, denoted $Err^*(y)$, is defined as follows:

$$\underline{Err^*(y)} = \frac{|\{\mathbf{x} \in X : y(\mathbf{x}) \neq c_{\mathbf{x}}\}|}{|X|} = \frac{|\{o \in O : y(\alpha(o)) \neq \gamma(o)\}|}{|O|}$$

Problem:

- Usually the *total function* $\gamma()$ is unknown and hence $Err^*(y)$ is unknown.

Solution:

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

Remarks:

- ❑ Alternative to “true misclassification rate” we will also use the term “true misclassification error” or simply “true error”.
- ❑ Since the total function $\gamma()$ is unknown, $c_{\mathbf{x}}$ is not given for all $\mathbf{x} \in X$. However, for some feature vectors $\mathbf{x} \in X$ we have knowledge about $c_{\mathbf{x}}$, namely for those in the multiset of examples D .
- ❑ If the mapping from feature vectors to classes is not unique, the multiset of examples D is said to contain (label) noise.

Remarks: (continued)

- ❑ 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 *proportion*, *percentage*, or *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 proportion, percentage, or ratio. Taking a precise mathematical standpoint, the correct German words are „Anteil“ or „Quote“. I.e., the correct translation of misclassification rate is „Missklassifikationsanteil“, and not „Missklassifikationsrate“.

- ❑ Finally, 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

\mathbf{X} : feature space

X : multiset of feature vectors

Evaluating Effectiveness

Misclassification Rate (continued)

Instead of defining $\text{Err}^*(y)$ as the ratio of misclassified features vectors in X , we can define $\text{Err}^*(y)$ as the *probability* that y misclassifies some \mathbf{x} , which depends on the joint distribution of the feature vectors and classes.

Evaluating Effectiveness

Misclassification Rate (continued)

Instead of defining $\underline{Err}^*(y)$ as the ratio of misclassified features vectors in X , we can define $\underline{Err}^*(y)$ as the *probability* that y misclassifies some \mathbf{x} , which depends on the joint distribution of the feature vectors and classes.

Definition 9 (Probabilistic Foundation of the True Misclassification Rate)

Let \mathbf{X} be a feature space with a finite number of elements, C a set of classes, and $y : \mathbf{X} \rightarrow C$ a classifier. Moreover, let Ω be a sample space, which corresponds to a set O of real-world objects, and P a probability measure defined on $\mathcal{P}(\Omega)$.

We consider two types of random variables, $\mathbf{X} : \Omega \rightarrow \mathbf{X}$, and $C : \Omega \rightarrow C$.

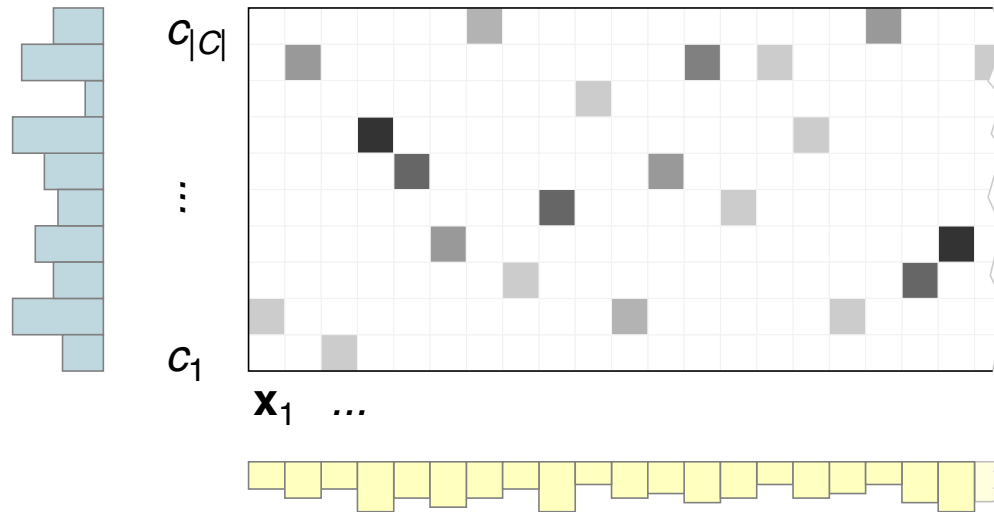
Then $p(\mathbf{x}, c)$, $p(\mathbf{x}, c) := P(\mathbf{X}=\mathbf{x}, C=c)$, is the probability of the joint event (1) to get the vector $\mathbf{x} \in \mathbf{X}$, and, (2) that the respective object belongs to class $c \in C$. With $p(\mathbf{x}, c)$ the true misclassification rate of $y(\mathbf{x})$ can be expressed as follows:

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

Evaluating Effectiveness

Illustration 1: Label Noise

Joint probabilities $p(\mathbf{x}, c) := P(\mathbf{X}=\mathbf{x}, \mathbf{C}=c)$ (shading indicates magnitude) :



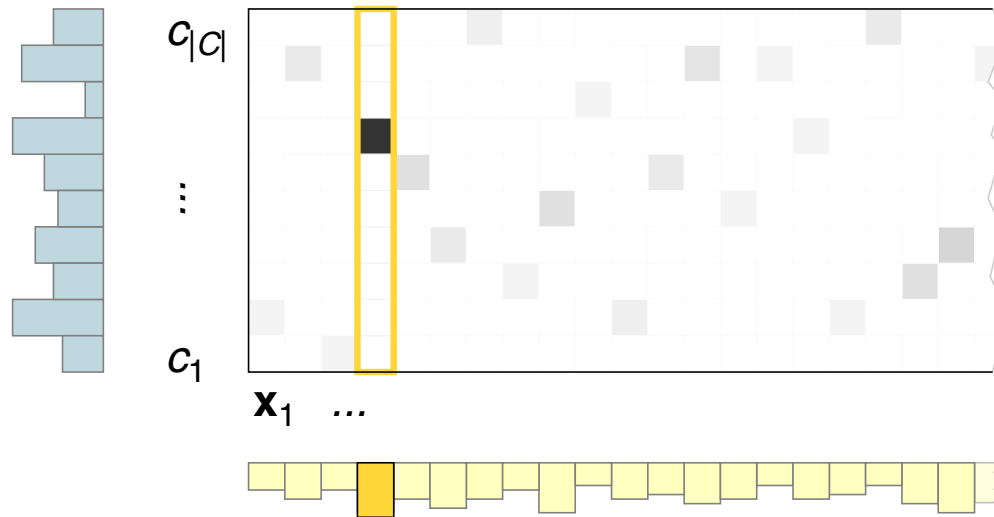
(no label noise \rightarrow classes are unique)

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

Evaluating Effectiveness

Illustration 1: Label Noise (continued)

Joint probabilities $p(\mathbf{x}, c) := P(\mathbf{X}=\mathbf{x}, \mathbf{C}=c)$ (shading indicates magnitude) :



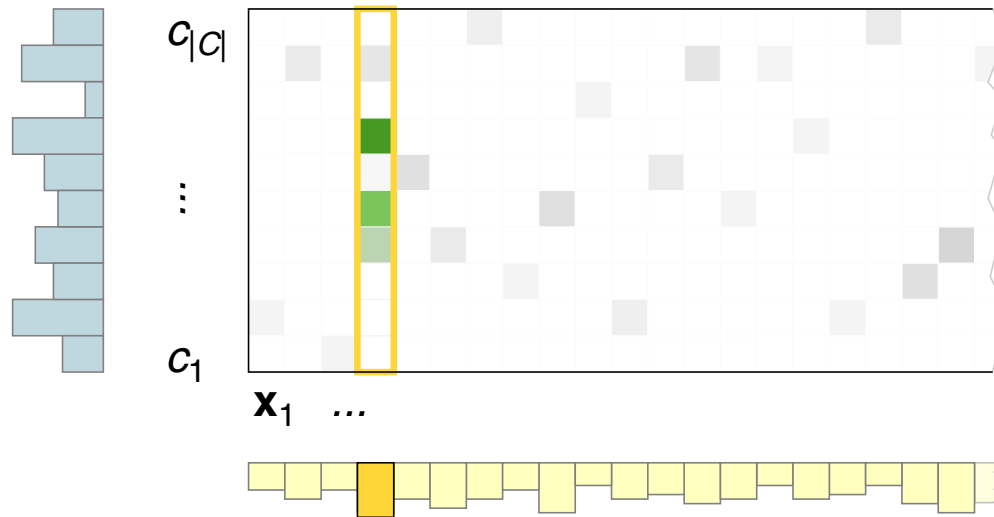
(no label noise \rightarrow classes are unique)

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

Evaluating Effectiveness

Illustration 1: Label Noise (continued)

Joint probabilities $p(\mathbf{x}, c) := P(\mathbf{X}=\mathbf{x}, \mathbf{C}=c)$ (shading indicates magnitude) :



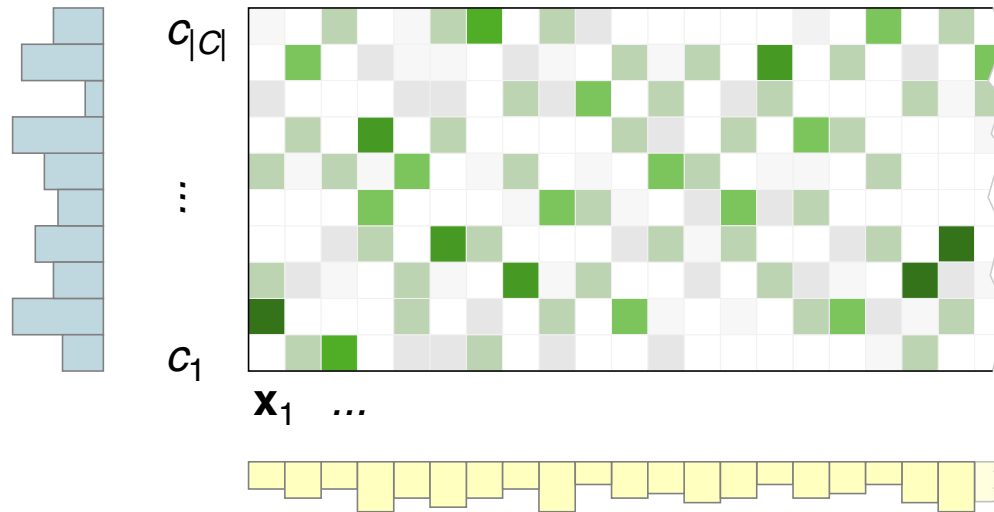
(label noise \rightarrow classes are *not* unique)

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

Evaluating Effectiveness

Illustration 1: Label Noise (continued)

Joint probabilities $p(\mathbf{x}, c) := P(\mathbf{X}=\mathbf{x}, \mathbf{C}=c)$ (shading indicates magnitude) :



(label noise \rightarrow classes are *not* unique)

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

Remarks:

- ❑ \mathbf{X} and C denote (multivariate) random variables with ranges X and C respectively.
 \mathbf{X} corresponds to a model formation function α , which returns for a real-world object $o \in O$ its feature vector \mathbf{x} , $\mathbf{x} = \alpha(o)$.
 C corresponds to an ideal classifier γ , which returns for a real-world object $o \in O$ its class c , $c = \gamma(o)$.
- ❑ \mathbf{X} models the fact that the occurrence of a feature vector is governed by a probability distribution, rendering certain observations more likely than others. Keyword: prior probability of [observing] \mathbf{x} .
Note that the multiset X of feature vectors in the true misclassification rate $Err^*(y)$ is governed by the distribution of \mathbf{X} : Objects in O that are more likely, but also very similar objects, will induce the respective multiplicity of feature vectors \mathbf{x} in X and hence are considered with the appropriate weight.
- ❑ C models the fact that the occurrence of a class is governed by a probability distribution, rendering certain classes more likely than others. Keyword: prior probability of c .

Remarks: (continued)

- ❑ The classification of a feature vector \mathbf{x} may not be deterministic: different objects in \mathcal{O} can be mapped to the same vector \mathbf{x} —but to [different classes](#). Reasons for a nondeterministic class assignment include: incomplete feature set, imprecision and random errors during feature measuring, lack of care during data acquisition. Keyword: label noise
- ❑ \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, presupposed a sensible discretization of the feature space \mathbf{X} .

[Wikipedia: [continuous setting](#), [illustration](#)]

Remarks: (continued)

- ❑ $P(\cdot)$ is a probability measure (see section [Probability Basics](#) in part Bayesian Learning) and its argument is an event. Examples for events are “ $\mathbf{X}=\mathbf{x}$ ”, “ $\mathbf{X}=\mathbf{x}, C=c$ ”, or “ $\mathbf{X}=\mathbf{x} \mid C=c$ ”.
- ❑ $p(\mathbf{x}, c)$, $p(\mathbf{x})$, or $p(\mathbf{x} \mid c)$ are examples for a [probability mass function](#), pmf. Its argument is a realization of a discrete random variable (or several discrete random variables), to which the pmf assigns a probability, based on a probability measure: $p(\cdot)$ is defined via $P(\cdot)$.

[\[illustration\]](#)

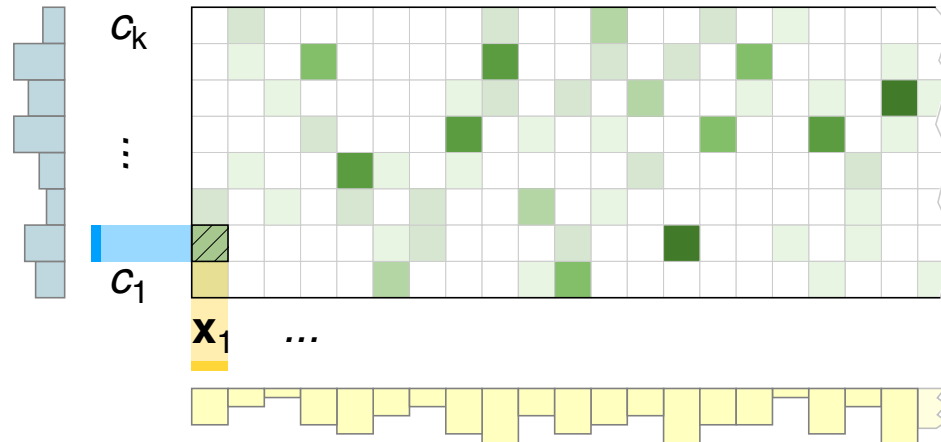
The counterpart of $p(\cdot)$ for a continuous random variable is called [probability density function](#), pdf, and is typically denoted by $f(\cdot)$.

- ❑ Since $p(\mathbf{x}, c)$ (and similarly $p(\mathbf{x})$, $p(\mathbf{x} \mid c)$, etc.) is defined as $P(\mathbf{X}=\mathbf{x}, C=c)$, the respective expressions for $p(\cdot)$ and $P(\cdot)$ can usually be used interchangeably. In this sense we have two parallel notations, arguing about realizations of random variables and events respectively.
- ❑ Let A and B denote two events, e.g., $A = “\mathbf{X}=\mathbf{x}_9”$ and $B = “C=c_3”$. Then the following expressions are equivalent notations for the probability of the joint event “ A and B ”: $P(A, B)$, $P(A \wedge B)$, $P(A \cap B)$.
- ❑ I_{\neq} is an indicator function that returns 1 if its arguments are *unequal* (and 0 if its arguments are equal).

Evaluating Effectiveness

Illustration 2: Bayes [Optimal] Classifier and Bayes Error

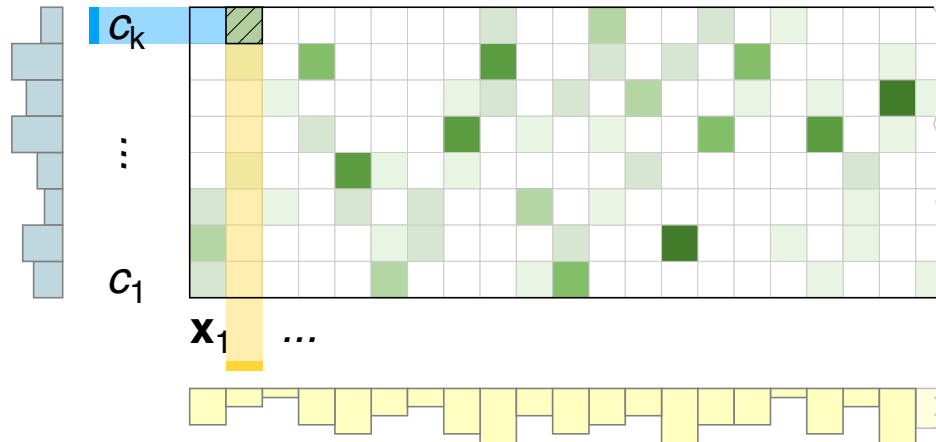
The Bayes classifier returns for \mathbf{x} the class with the highest [posterior] probability:



Evaluating Effectiveness

Illustration 2: Bayes [Optimal] Classifier and Bayes Error (continued)

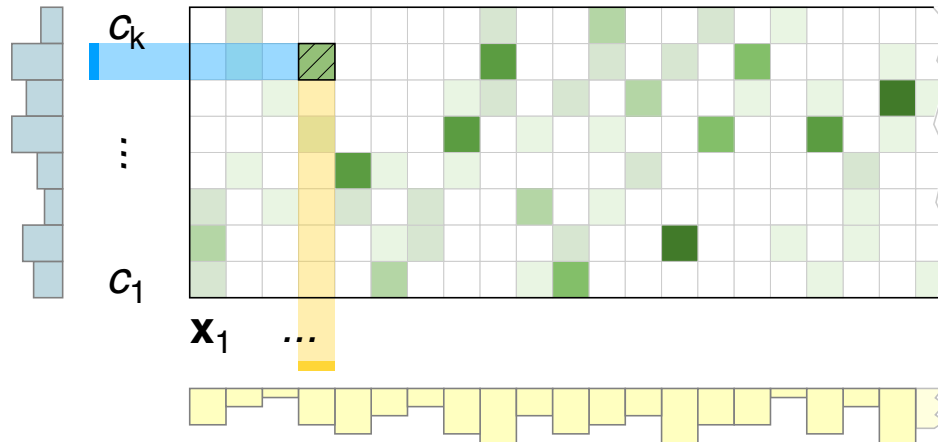
The Bayes classifier returns for \mathbf{x} the class with the highest [posterior] probability:



Evaluating Effectiveness

Illustration 2: Bayes [Optimal] Classifier and Bayes Error (continued)

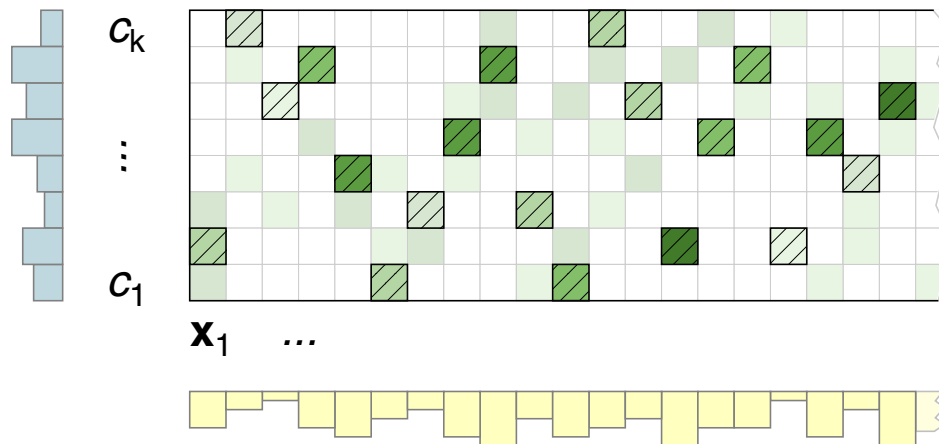
The Bayes classifier returns for \mathbf{x} the class with the highest [posterior] probability:



Evaluating Effectiveness

Illustration 2: Bayes [Optimal] Classifier and Bayes Error (continued)

The Bayes classifier returns for \mathbf{x} the class with the highest [posterior] probability:

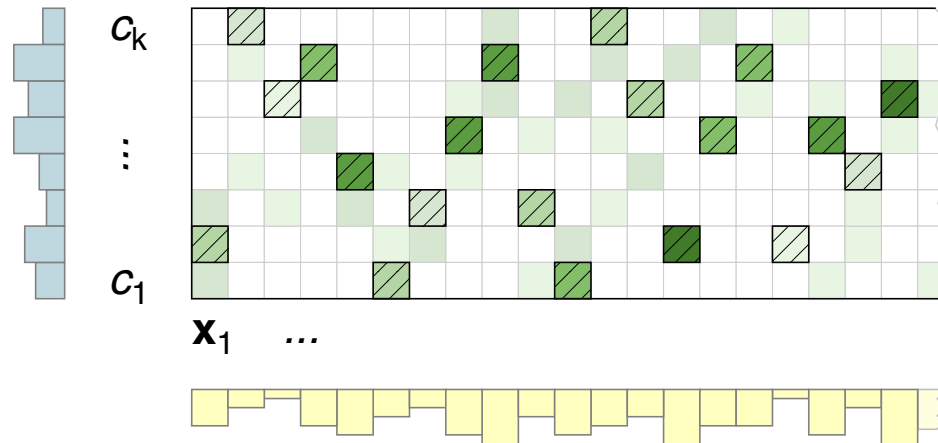


Bayes classifier:
$$y^*(\mathbf{x}) = \operatorname{argmax}_{c \in C} p(c, \mathbf{x}) = \operatorname{argmax}_{c \in C} p(c \mid \mathbf{x})$$

Evaluating Effectiveness

Illustration 2: Bayes [Optimal] Classifier and Bayes Error (continued)

The Bayes classifier returns for \mathbf{x} the class with the highest [posterior] probability:



Bayes classifier: $y^*(\mathbf{x}) = \operatorname{argmax}_{c \in C} p(c, \mathbf{x}) = \operatorname{argmax}_{c \in C} p(c \mid \mathbf{x})$

Bayes error: $\text{Err}^* = \sum_{\mathbf{x} \in \mathbf{X}} \sum_{c \in C} p(\mathbf{x}, c) \cdot I_{\neq}(y^*(\mathbf{x}), c) = \sum_{\mathbf{x} \in \mathbf{X}} (1 - \max_{c \in C} \{p(c, \mathbf{x})\})$

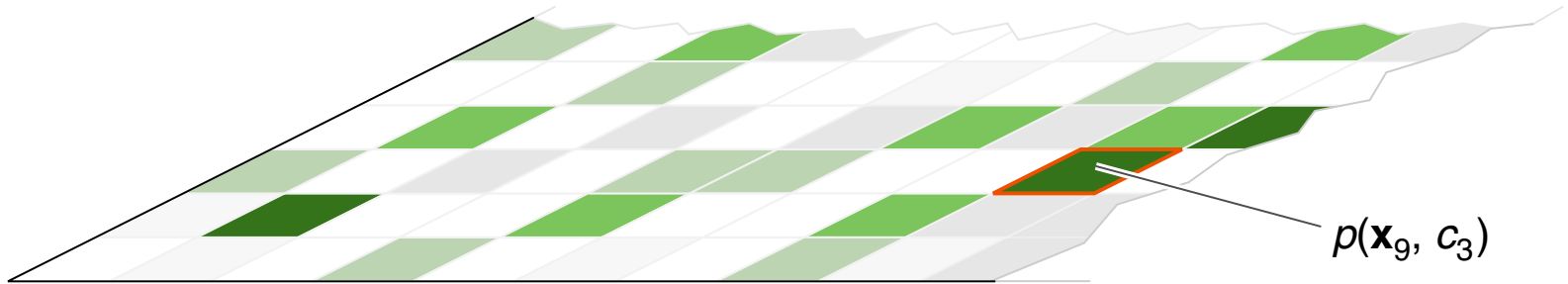
Remarks (Bayes classifier):

- ❑ The Bayes classifier (also: Bayes optimal classifier) maps each feature vector \mathbf{x} to the highest-probability class c according to the true joint probability distribution $p(c, \mathbf{x})$ that generates the data.
- ❑ The Bayes classifier incurs an error—the Bayes error—on feature vectors that have more than one possible class assignment with non-zero probability. This may be the case when the class assignment depends on additional (unobserved) features not recorded in \mathbf{x} , or when the relationship between objects and classes is inherently stochastic.
[\[Goodfellow et al. 2016, p.114\]](#) [\[Bishop 2006, p.40\]](#) [\[Daumé III 2017, ch.2\]](#) [\[Hastie et al. 2009, p.21\]](#)
- ❑ The Bayes error hence is the theoretically minimal error that can be achieved on average for a classifier learned from a multiset of examples D . It is also referred to as Bayes rate, irreducible error, or unavoidable error, and it forms a lower bound for the error of any model created without knowledge of the probability distribution $p(c, \mathbf{x})$.
- ❑ Prerequisite to construct the Bayes classifier and to compute its error is knowledge about the joint probabilities, $p(c, \mathbf{x})$ or $p(c | \mathbf{x})$. In this regard the size of the available data, D , decides about the possibility and the quality for the estimation of the probabilities.
- ❑ Do not mix up the following two issues: (1) The joint probabilities cannot be reliably estimated, (2) the joint probabilities can be reliably estimated but entail an unacceptably large Bayes error. The former issue can be addressed by enlarging D . The latter issue indicates the deficiency of the features, which can neither be repaired with more data nor with a (very complex) model function, but which requires the identification of new, more effective features: the model formation process is to be reconsidered.

Evaluating Effectiveness

Illustration 3: Marginal and Conditional Distributions

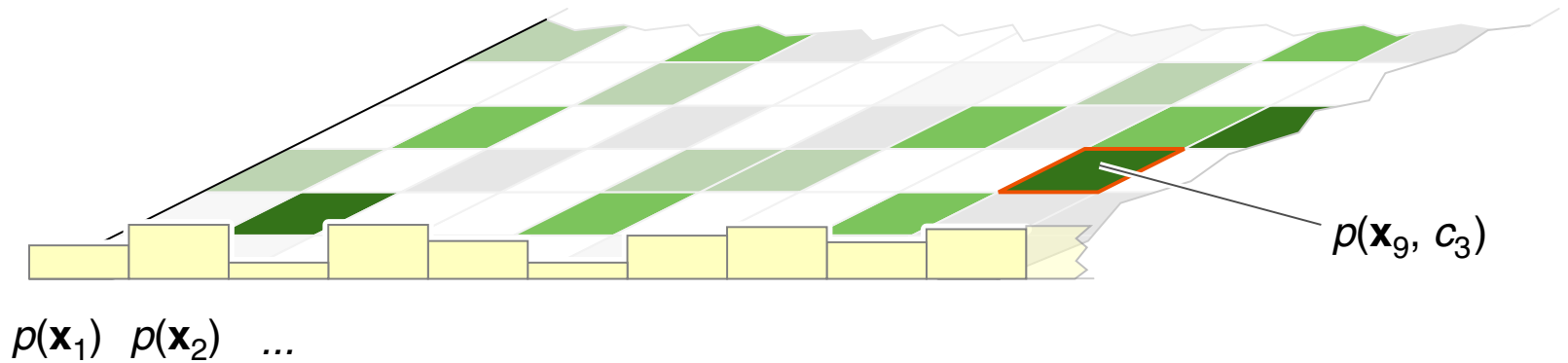
Joint probabilities $p(\mathbf{x}, c) := P(\mathbf{X}=\mathbf{x}, \mathbf{C}=c)$:



Evaluating Effectiveness

Illustration 3: Marginal and Conditional Distributions (continued)

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



Evaluating Effectiveness

Illustration 3: Marginal and Conditional Distributions (continued)

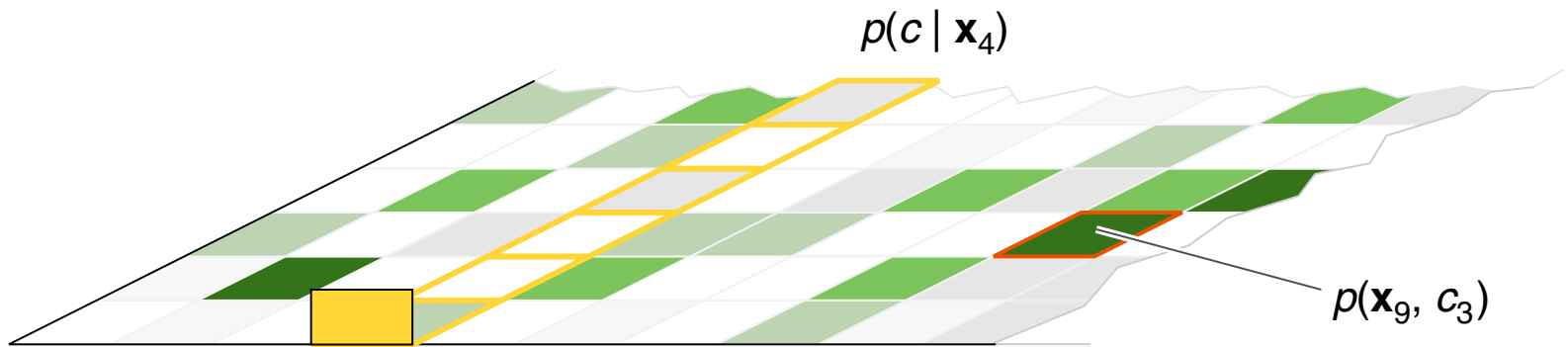
Marginal probabilities $p(c) := P(\mathbf{C}=c)$:



Evaluating Effectiveness

Illustration 3: Marginal and Conditional Distributions (continued)

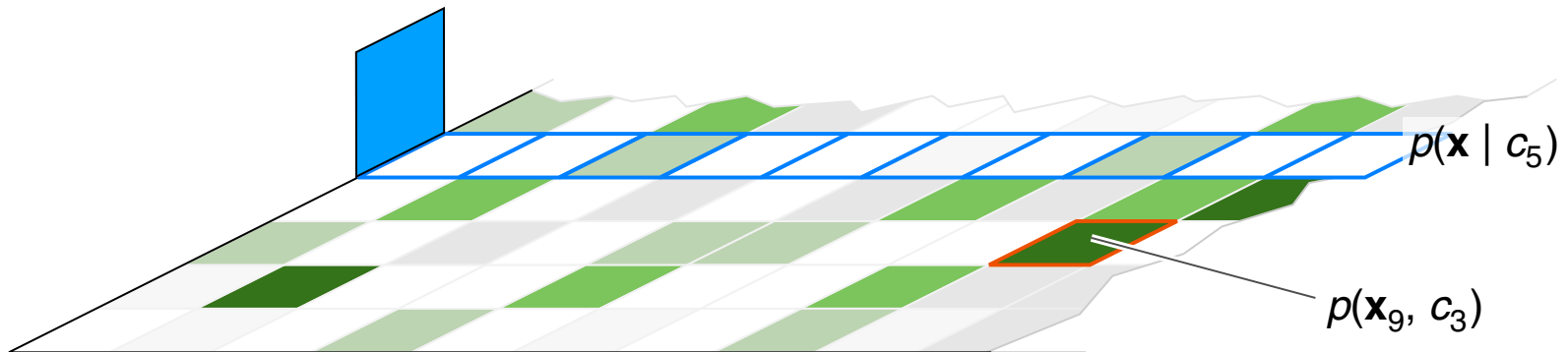
Probabilities of the classes c under feature vector (the condition) \mathbf{x}_4 , denoted by $p(c \mid \mathbf{x}_4) := P(\mathbf{C}=c \mid \mathbf{X}=\mathbf{x}_4) \equiv P_{\mathbf{X}=\mathbf{x}_4}(\mathbf{C}=c)$:



Evaluating Effectiveness

Illustration 3: Marginal and Conditional Distributions (continued)

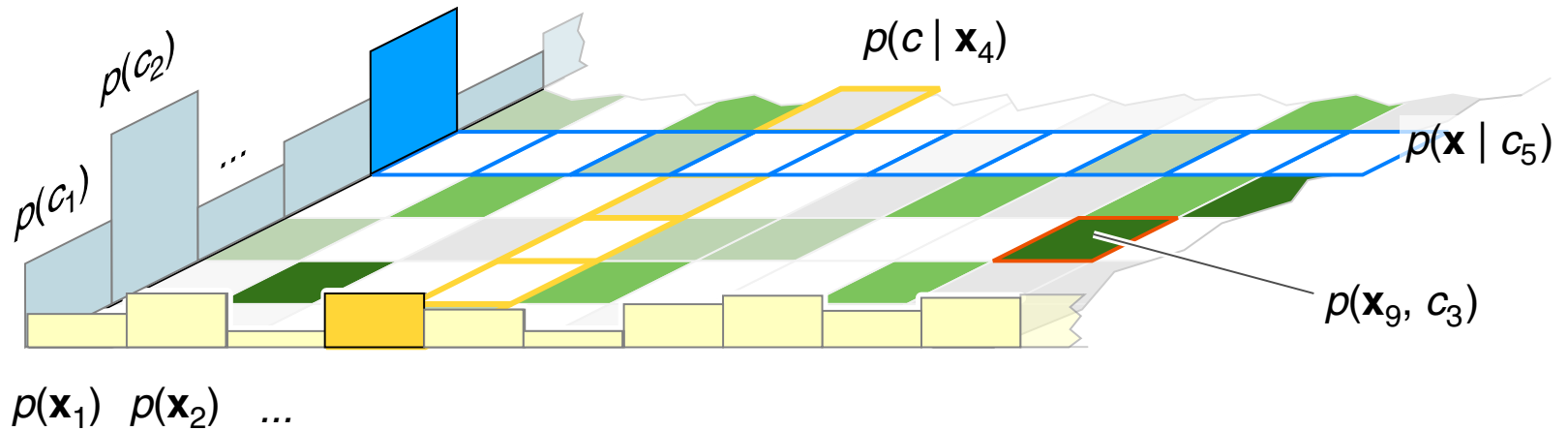
Probabilities of the feature vectors \mathbf{x} under class (the condition) c_5 , denoted by $p(\mathbf{x} \mid c_5) := P(\mathbf{X}=\mathbf{x} \mid \mathbf{C}=c_5) \equiv P_{\mathbf{C}=c_5}(\mathbf{X}=\mathbf{x})$:



Evaluating Effectiveness

Illustration 3: Marginal and Conditional Distributions (continued)

Overview:



Remarks:

□ $p(c \mid \mathbf{x}) := P(\mathbf{X}=\mathbf{x}, C=c)/P(\mathbf{X}=\mathbf{x}) = P(C=c \mid \mathbf{X}=\mathbf{x}) \equiv P_{\mathbf{X}=\mathbf{x}}(C=c)$

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

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

□ $p(\mathbf{x} \mid c) := P(\mathbf{X}=\mathbf{x}, C=c)/P(C=c) = P(\mathbf{X}=\mathbf{x} \mid C=c) \equiv P_{C=c}(\mathbf{X}=\mathbf{x})$

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

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

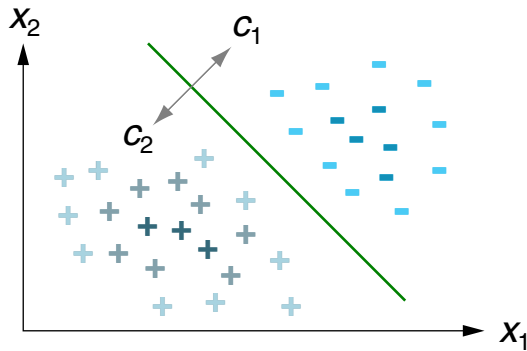
□ $p(\mathbf{x}, c) = p(c, \mathbf{x}) = p(c \mid \mathbf{x}) \cdot p(\mathbf{x})$, where $p(\mathbf{x})$ is the prior probability for event $\mathbf{X}=\mathbf{x}$, and $p(c \mid \mathbf{x})$ is the probability for event $C=c$ given event $\mathbf{X}=\mathbf{x}$. Likewise, $p(\mathbf{x}, c) = p(\mathbf{x}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

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

In the Bayes classification setting $p(c \mid \mathbf{x})$ is called “posterior probability”, i.e., the probability for c after we know that \mathbf{x} has occurred.

Evaluating Effectiveness

Illustration 4: Probability Distribution in a Regression Setting

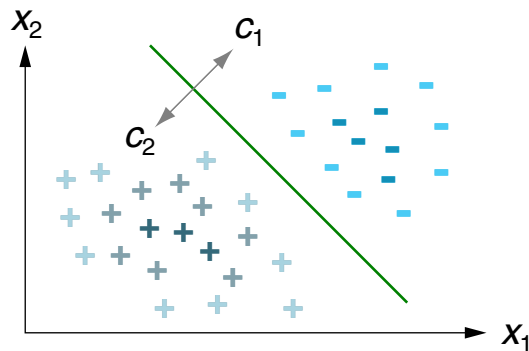


$$X = \left\{ \begin{pmatrix} x_{1_1} \\ x_{1_2} \end{pmatrix}, \begin{pmatrix} x_{2_1} \\ x_{2_2} \end{pmatrix}, \dots \right\}, \quad \mathbf{X} = \mathbf{R}^2$$

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

Evaluating Effectiveness

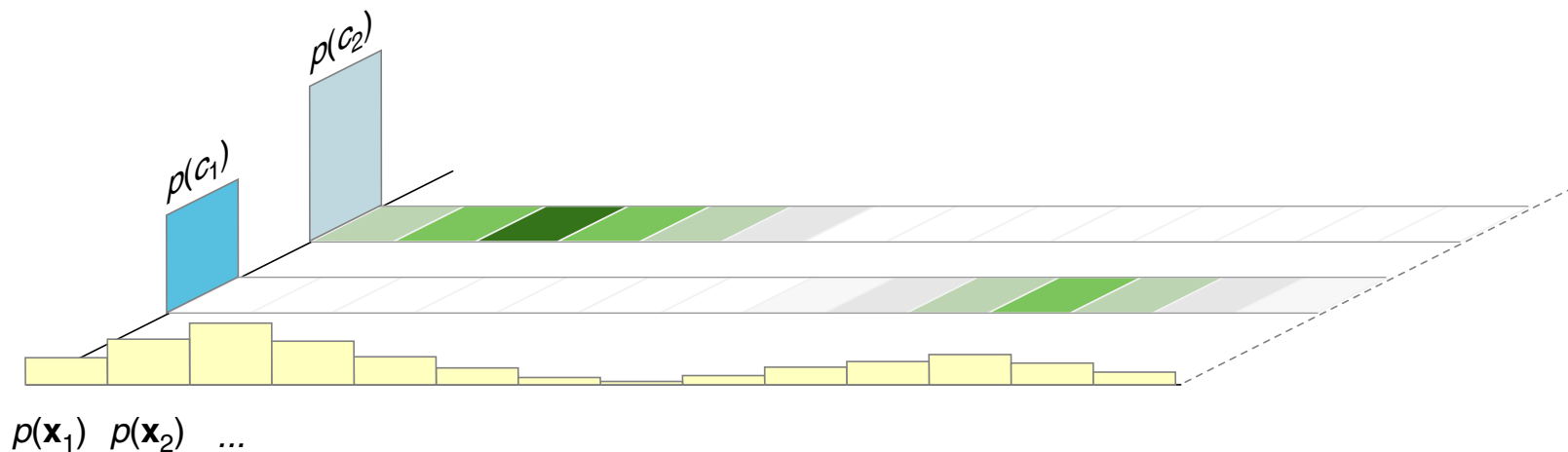
Illustration 4: Probability Distribution in a Regression Setting (continued)



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

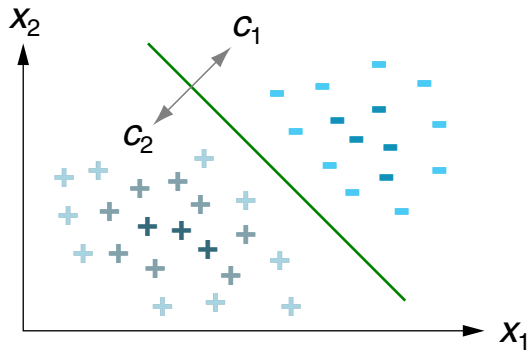
$\mathbf{x}_1 \quad \mathbf{x}_2 \quad \dots$

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



Evaluating Effectiveness

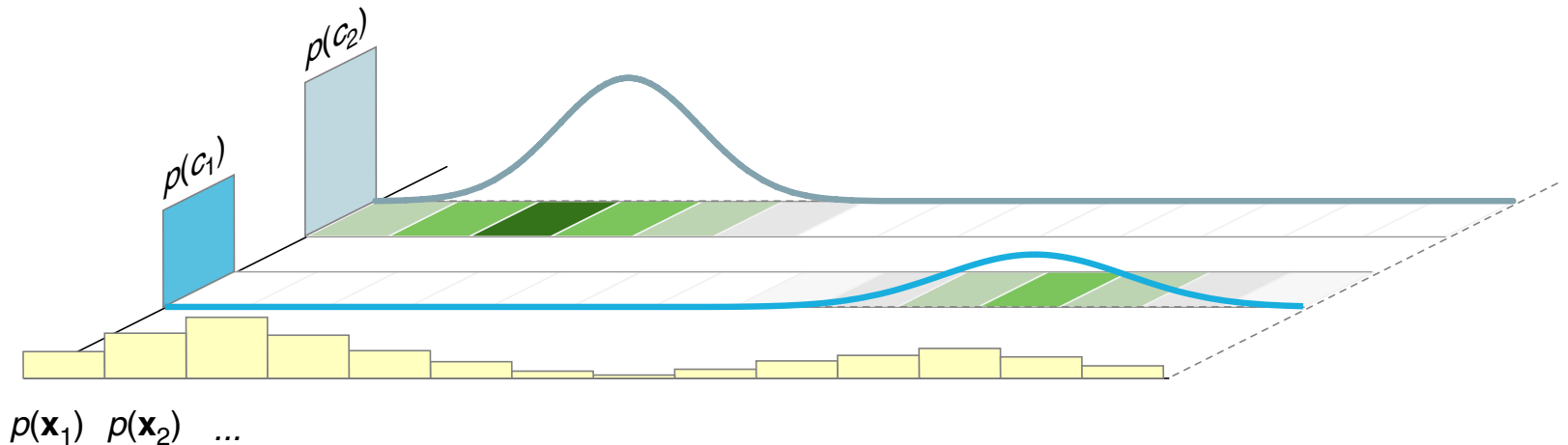
Illustration 4: Probability Distribution in a Regression Setting (continued)



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

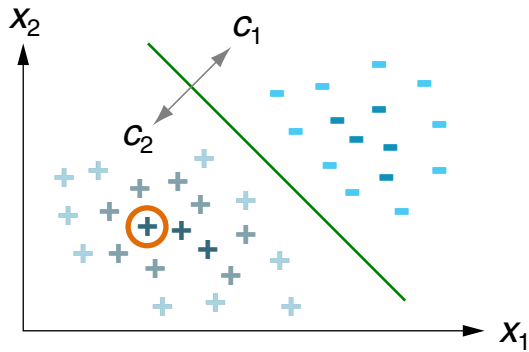
$\mathbf{x}_1 \quad \mathbf{x}_2 \quad \dots$

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



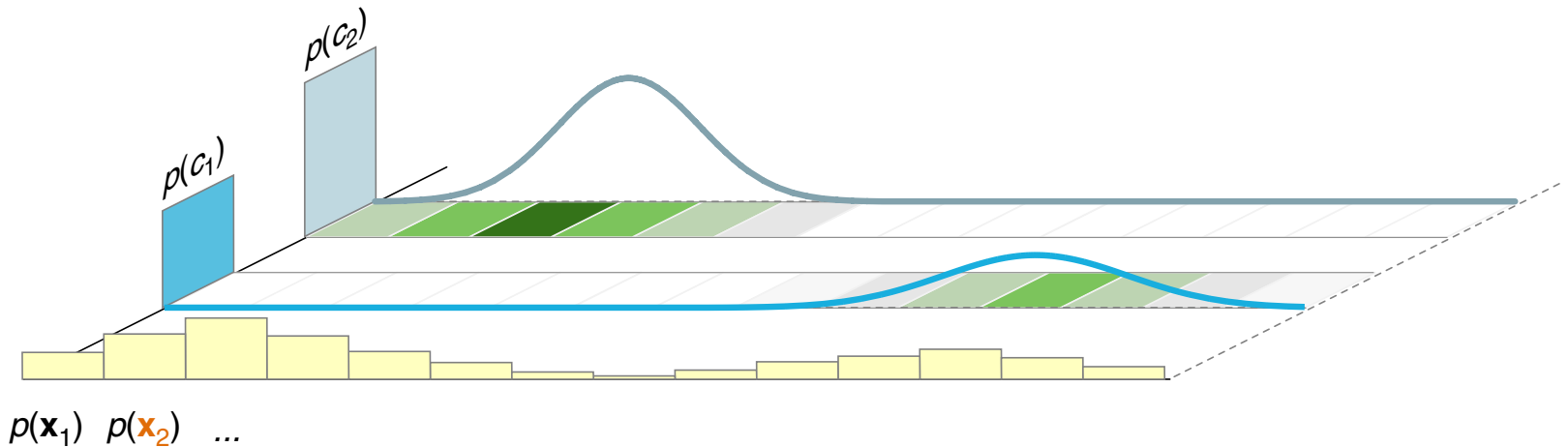
Evaluating Effectiveness

Illustration 4: Probability Distribution in a Regression Setting (continued)



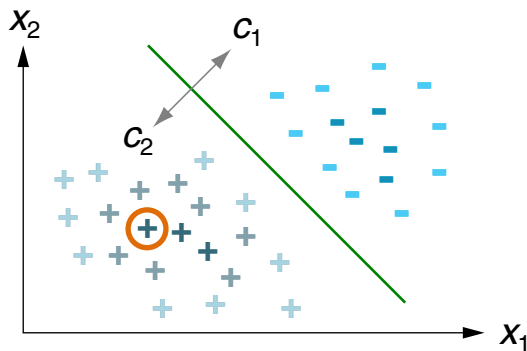
$$X = \left\{ \begin{pmatrix} x_{1_1} \\ x_{1_2} \end{pmatrix}, \begin{pmatrix} x_{2_1} \\ x_{2_2} \end{pmatrix}, \dots \right\}, \quad \mathbf{X} = \mathbf{R}^2$$

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



Evaluating Effectiveness

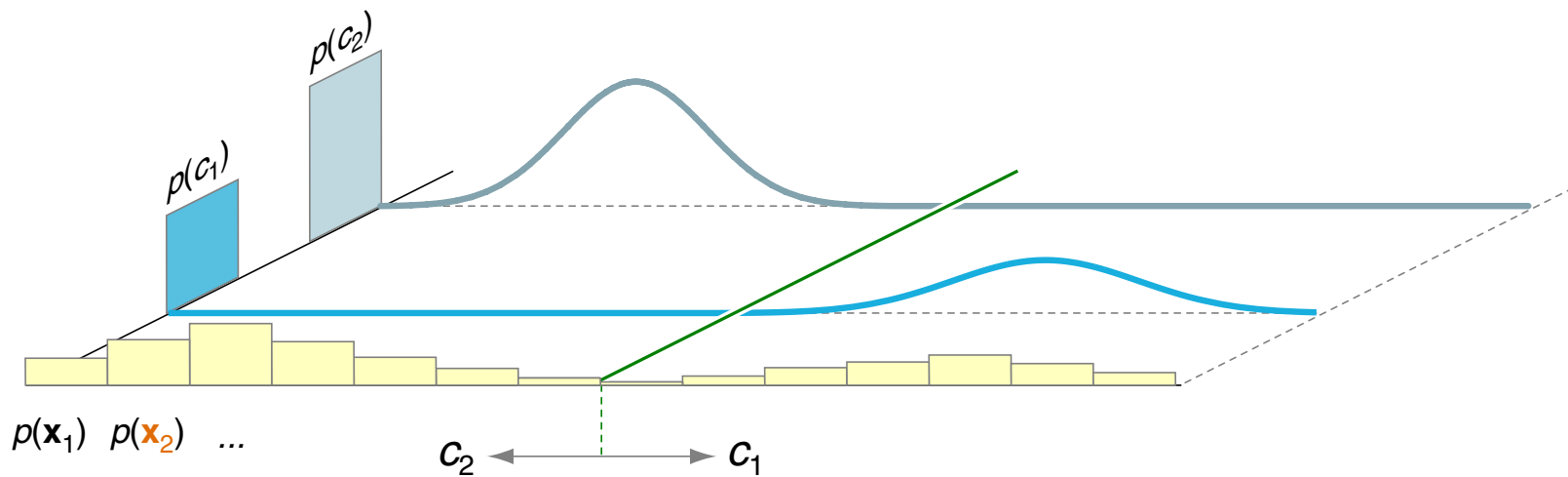
Illustration 4: Probability Distribution in a Regression Setting (continued)



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

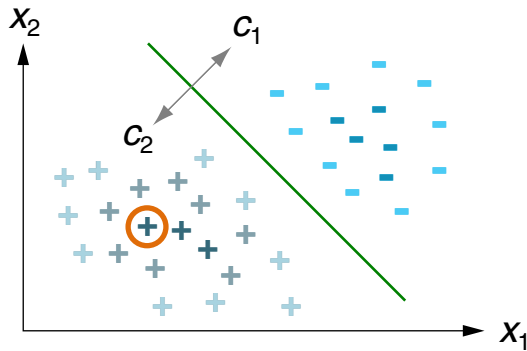
$\mathbf{x}_1 \quad \mathbf{x}_2 \quad \dots$

Optimum hyperplane classifier:



Evaluating Effectiveness

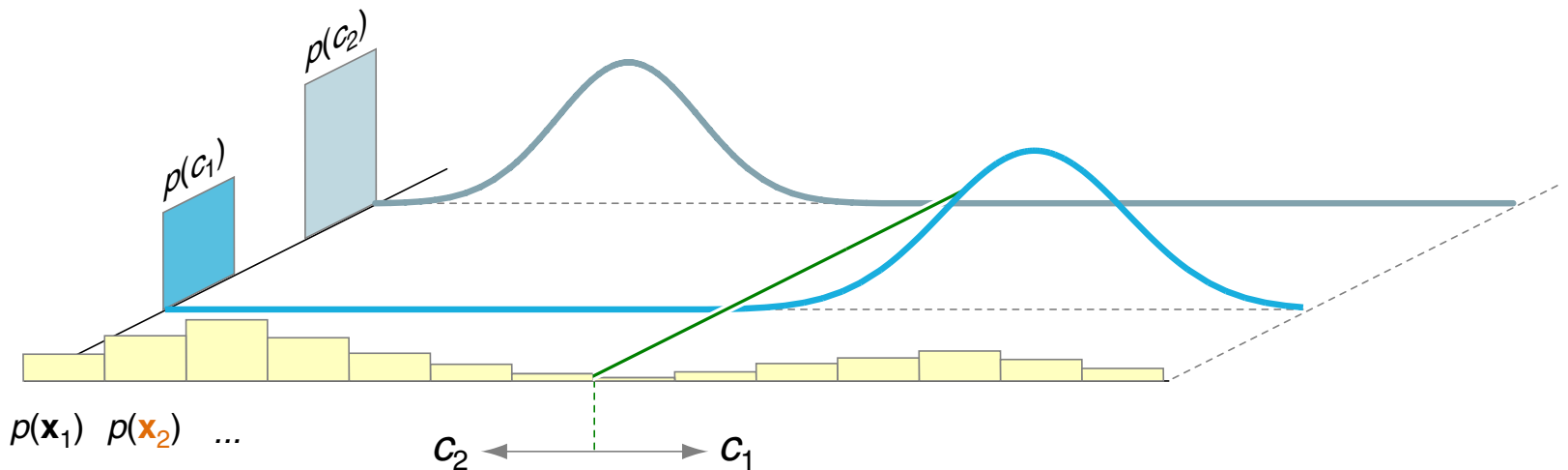
Illustration 4: Probability Distribution in a Regression Setting (continued)



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

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

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



Remarks:

- ❑ The illustration shows a classification task without label noise: each feature vector \mathbf{x} belongs to exactly one class. Moreover, the classification task can be reduced to solving a regression problem (e.g., via the [LMS algorithm](#)). Even more, for perfect classification the regression function needs to define a straight line only. Keyword: linear separability
- ❑ Solving classification tasks via regression 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.
- ❑ 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.

Evaluating Effectiveness

Estimating Error Bounds

Experiment setting:

- $D = \{(\mathbf{x}_1, c_1), \dots, (\mathbf{x}_n, c_n)\} \subseteq X \times C$ is a multiset of examples.
- $y(\mathbf{x})$ is the classifier trained on D .
- The true error $\text{Err}^*(y)$ measures the performance of $y(\mathbf{x})$ on X (“in the wild”).
- What can be said about the true error $\text{Err}^*(y)$?

Evaluating Effectiveness

Estimating Error Bounds (continued)

Experiment setting:

- $D = \{(\mathbf{x}_1, c_1), \dots, (\mathbf{x}_n, c_n)\} \subseteq X \times C$ is a multiset 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 $Err^*(y)$?

The following relations typically hold:

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


Evaluating Effectiveness

Estimating Error Bounds (continued)

Experiment setting:

- $D = \{(\mathbf{x}_1, c_1), \dots, (\mathbf{x}_n, c_n)\} \subseteq X \times C$ is a multiset of examples.
- $y(\mathbf{x})$ is the classifier trained on D .
- The true error $\text{Err}^*(y)$ measures the performance of $y(\mathbf{x})$ on X (“in the wild”).
- What can be said about the true error $\text{Err}^*(y)$?

The following relations typically hold:

Underestimation (likely)			Overestimation (unlikely)	
Training error	Cross-validation error	Holdout error	True error	
$\text{Err}_{tr}(y)$	$<$	$\text{Err}_{cv}(y, k)$	\lesssim	$\text{Err}(y, D_{test})$
		$<$	$\text{Err}^*(y)$	$<$
			$\text{Err}_{cv}(y, k)$	\lesssim
			$\text{Err}(y, D_{test})$	
$ D \rightarrow X $				


Evaluating Effectiveness

Estimating Error Bounds (continued)

Experiment setting:

- $D = \{(\mathbf{x}_1, c_1), \dots, (\mathbf{x}_n, c_n)\} \subseteq X \times C$ is a multiset of examples.
- $y(\mathbf{x})$ is the classifier trained on D .
- The true error $\text{Err}^*(y)$ measures the performance of $y(\mathbf{x})$ on X (“in the wild”).
- What can be said about the true error $\text{Err}^*(y)$?

The following relations typically hold:

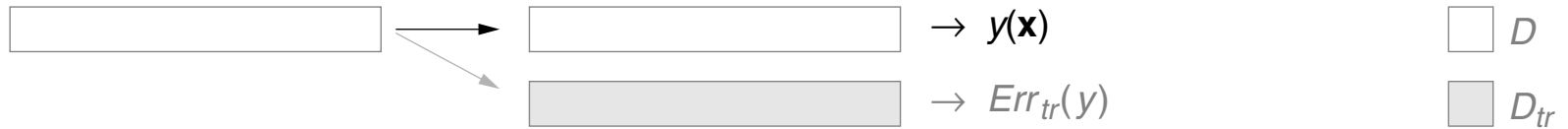
Underestimation (likely)			Overestimation (unlikely)	
Training error	Cross-validation error	Holdout error	True error	
$\text{Err}_{tr}(y)$	$< \text{Err}_{cv}(y, k) \lesssim$	$\text{Err}(y, D_{test})$	$< \text{Err}^*(y) < \text{Err}_{cv}(y, k) \lesssim$	$\text{Err}(y, D_{test})$
				

Remarks:

- ❑ Relating the true error $\text{Err}^*(y)$ to the aforementioned error assessments $\text{Err}_{tr}(y)$, $\text{Err}_{cv}(y, k)$, and $\text{Err}(y, D_{test})$ 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.
- ❑ The additional argument in the definitions of the error functions, k and D_{test} respectively, are necessary to completely specify the error computation. The set D is not specified as an argument since it is an integral and constant parameter of the learning procedure underlying $y(\mathbf{x})$.

Evaluating Effectiveness

Training Error



Evaluation setting:

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

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

- ❑ $\underline{Err_{tr}(y)} = \frac{|\{(\mathbf{x}, c) \in D_{tr} : y(\mathbf{x}) \neq c\}|}{|D_{tr}|}$
= misclassification rate of $y(\mathbf{x})$ on the training set.

Remarks:

- The estimation of $Err_{tr}(y)$ is based on $y(\mathbf{x})$ and tests against $D_{tr} = D$. I.e., the same examples that are used for training $y(\mathbf{x})$ are also used to test $y(\mathbf{x})$. Hence $Err_{tr}(y)$ quantifies the *memorization* power of $y(\mathbf{x})$ but not its *generalization* power.

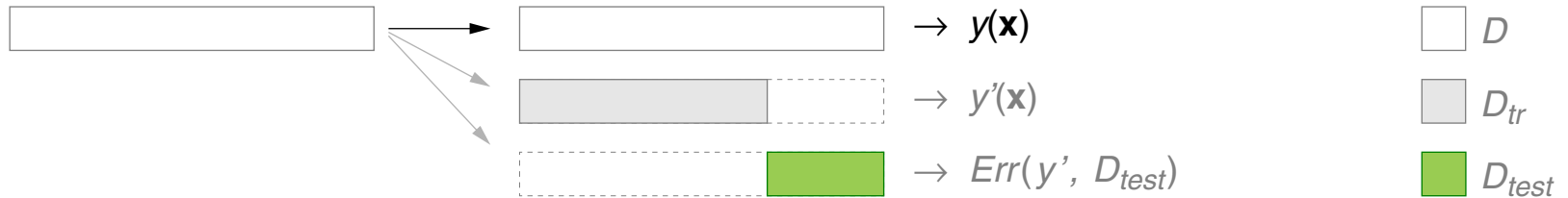
Consider the extreme case: If $y(\mathbf{x})$ stored D during “training” into a hashtable (key $\sim \mathbf{x}$, value $\sim c$), then $Err_{tr}(y)$ would be zero, which would tell us nothing about the failure of $y(\mathbf{x})$ in the wild.

- $Err_{tr}(y)$ is an optimistic estimation, i.e., it is constantly lower compared to the (unknown) true error $Err^*(y)$. With $D = X$ the training error $Err_{tr}(y)$ becomes the true error $Err^*(y)$.
- Note that the above issues relate to the meaningfulness of $Err_{tr}(y)$ 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 every classifier $y'(\mathbf{x})$ trained on a subset of D .

Evaluating Effectiveness

Holdout Error



Evaluation 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 $y(\mathbf{x})$:

- $\underline{Err}(y, D_{test}) = \frac{|\{(\mathbf{x}, c) \in D_{test} : y'(\mathbf{x}) \neq c\}|}{|D_{test}|}$
 $=$ misclassification rate of $y'(\mathbf{x})$ on the test set.

Evaluating Effectiveness

Holdout Error (continued)

1. Training (D, η)

```
1. initialize_random_weights( $\mathbf{w}$ ),  $t = 0$   
2. REPEAT  
   $\vdots$   
10. UNTIL(convergence( $D, y(\cdot), t$ )
```

2. Training (D_{tr}, η)

```
1. initialize_random_weights( $\mathbf{w}$ ),  $t = 0$   
2. REPEAT  
   $\vdots$   
10. UNTIL(convergence( $D_{tr}, y'(\cdot), t$ )
```

3. Test ($D_{test}, y'(\cdot)$)

Evaluating Effectiveness

Holdout Error (continued)

1. Training (D, η)

```
1. initialize_random_weights(w),  $t = 0$   
2. REPEAT  
   $\vdots$   
10. UNTIL(convergence( $D, y(\cdot), t$ )
```

$\leadsto y(\mathbf{x})$

$\leadsto Err_{tr}(y)$ (not used)

2. Training (D_{tr}, η)

```
1. initialize_random_weights(w),  $t = 0$   
2. REPEAT  
   $\vdots$   
10. UNTIL(convergence( $D_{tr}, y'(\cdot), t$ )
```

$\leadsto y'(\mathbf{x})$

$\leadsto Err_{tr}(y')$

3. Test ($D_{test}, y'(\cdot)$)

$\leadsto Err(y, D_{test})$

Evaluating Effectiveness

Holdout Error (continued)

1. Training (D, η)

```
1. initialize_random_weights( $\mathbf{w}$ ),  $t = 0$   
2. REPEAT  
   $\vdots$   
10. UNTIL(convergence( $D, y(\cdot)$ ),  $t$ )
```

$\leadsto y(\mathbf{x})$

$\leadsto Err_{tr}(y)$ (not used)

2. Training (D_{tr}, η)

```
1. initialize_random_weights( $\mathbf{w}$ ),  $t = 0$   
2. REPEAT  
   $\vdots$   
10. UNTIL(convergence( $D_{tr}, y'(\cdot)$ ),  $t$ )
```

$\leadsto y'(\mathbf{x})$

$\leadsto Err_{tr}(y')$

3. Test ($D_{test}, y'(\cdot)$)

$\leadsto Err(y, D_{test})$

Remarks:

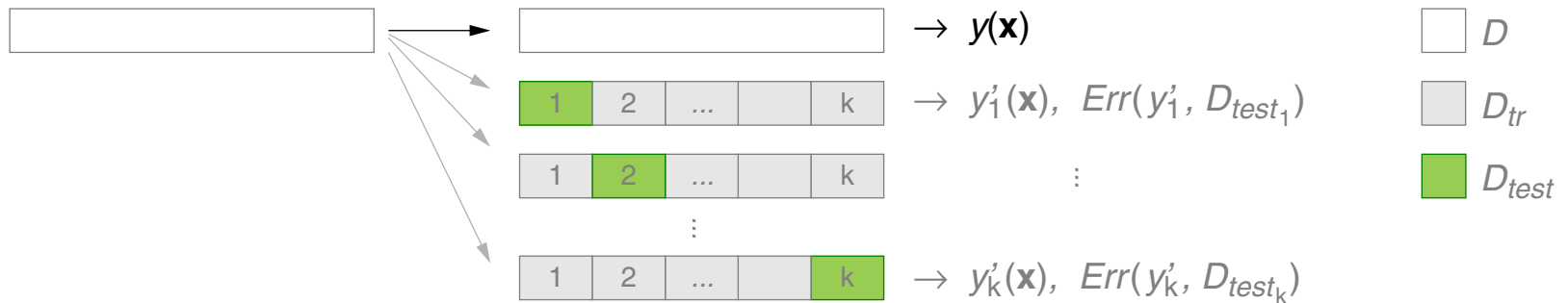
- ❑ The difference between the training error, $Err_{tr}(y)$ or $Err_{tr}(y')$, and the holdout error, $Err(y, D_{test})$, quantifies the severity of a possible overfitting.
- ❑ When splitting D into D_{tr} and D_{test} one has to ensure that the underlying distribution is maintained, i.e., the examples 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 $Err^*(y)$.
Keyword: sample selection bias
- ❑ An important aspect of the underlying data distribution specific to classification problems is the relative frequency of the classes. A sample $D_{tr} \subset D$ is called a (class-)stratified sample of D if it has the same class frequency distribution as D , i.e.:

$$\forall c_i \in C : \frac{|\{(\mathbf{x}, c) \in D_{tr} : c = c_i\}|}{|D_{tr}|} \approx \frac{|\{(\mathbf{x}, c) \in D : c = c_i\}|}{|D|}$$

- ❑ 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.”
- ❑ Regarding the notation: We will use the prime symbol $\gg' \ll$ 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.

Evaluating Effectiveness

k -Fold Cross-Validation

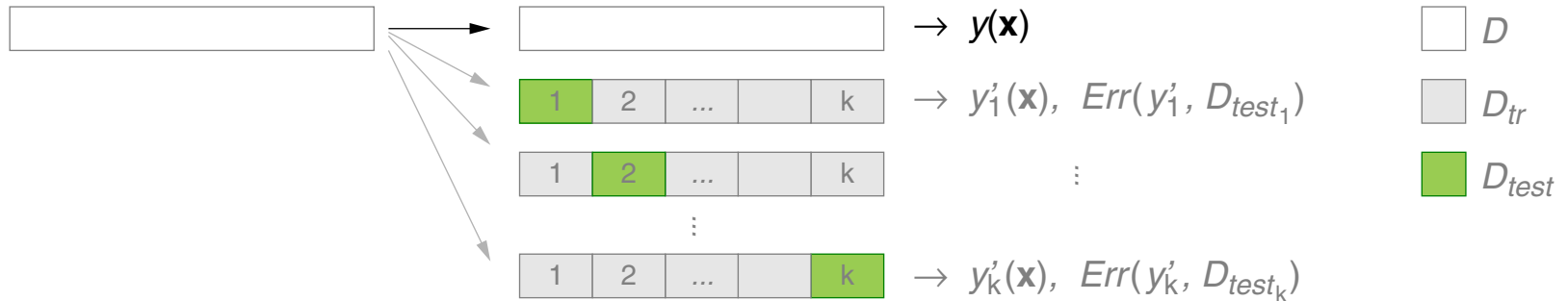


Evaluation setting:

- k test sets D_{test_i} by splitting D into k disjoint sets of similar size.
- $y(\mathbf{x})$ is the classifier trained on D .
- $y'_i(\mathbf{x}), i = 1, \dots, k$, are the classifiers trained on $D_{tr} = D \setminus D_{test_i}$.

Evaluating Effectiveness

k -Fold Cross-Validation (continued)



Evaluation setting:

- k test sets D_{test_i} by splitting D into k disjoint sets of similar size.
- $y(\mathbf{x})$ is the classifier trained on D .
- $y'_i(\mathbf{x}), i = 1, \dots, k$, are the classifiers trained on $D_{tr} = D \setminus D_{\text{test}_i}$.

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

$$\begin{aligned} \square \text{Err}_{cv}(y, k) &= \frac{1}{k} \sum_{i=1}^k \frac{|\{(\mathbf{x}, c) \in D_{\text{test}_i} : y'_i(\mathbf{x}) \neq c\}|}{|D_{\text{test}_i}|} \\ &= \text{averaged misclassification rate of the } y'_i(\mathbf{x}) \text{ on the } k \text{ test sets.} \end{aligned}$$

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. [\[Wikipedia\]](#)
- ❑ Instead of splitting D into disjoint subsets through sampling without replacement, it is also possible to generate folds by sampling *with* replacement; this results in a bootstrap estimate for $Err^*(y)$ (see section [Ensemble Methods > Bootstrap Aggregating](#) in part Ensemble and Meta). [\[Wikipedia\]](#)

Evaluating Effectiveness

Comparing Model Variants

Experiment setting:

- $D = \{(\mathbf{x}_1, c_1), \dots, (\mathbf{x}_n, c_n)\} \subseteq X \times C$ is a multiset of examples.
- m hyperparameter values $\pi_1, \pi_2, \dots, \pi_m$,
- $y_{\pi_1}(\mathbf{x}), y_{\pi_2}(\mathbf{x}), \dots, y_{\pi_m}(\mathbf{x})$ are the classifiers trained on D .
- Which is the most effective among the m classifiers $y_{\pi_l}(\mathbf{x})$?

Remarks:

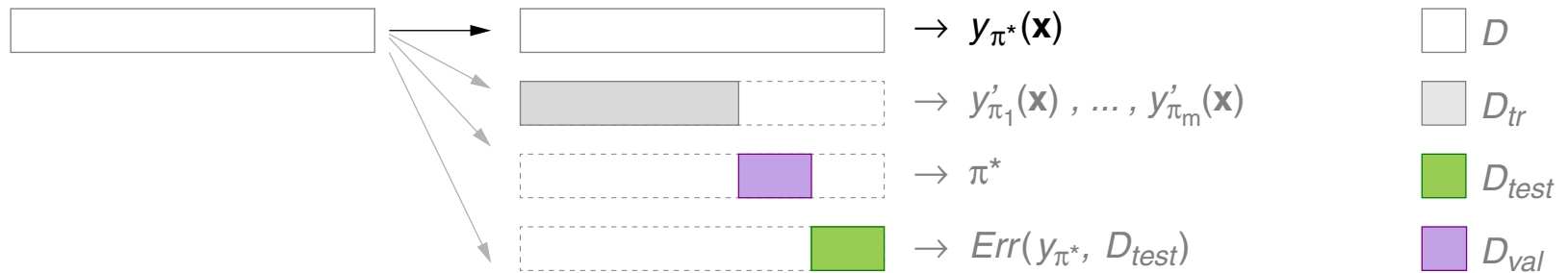
- ❑ In general, a hyperparameter π (with values $\pi_1, \pi_2, \dots, \pi_m$) controls the learning process for a model's parameters, but is itself not learned.

A regime where knowledge (such as hyperparameter settings) *about* a machine learning process is learned is called meta learning.

- ❑ Examples for hyperparameters in different kinds of model functions:
 - learning rate η in regression-based models fit via gradient descent
 - type of regularization loss used, e.g., $R_{||\vec{w}||_2^2}$ or $R_{||\vec{w}||_1}$
 - the term λ controlling the weighting of regularization loss and goodness-of-fit loss
 - number of hidden layers and the number of units per layer in multilayer perceptrons
 - choice of impurity function and pruning strategy in decision trees
 - architectural choices in deep-learning based models
- ❑ Different search strategies may be combined with cross-validation to find an optimal combination of hyperparameters for a given dataset and family of model functions. Depending on the size of the hyperparameter space, appropriate strategies can include both exhaustive grid search and approximation methods (metaheuristics) such as tabu search, simulated annealing, or evolutionary algorithms.

Evaluating Effectiveness

Model Selection: Single Validation Set

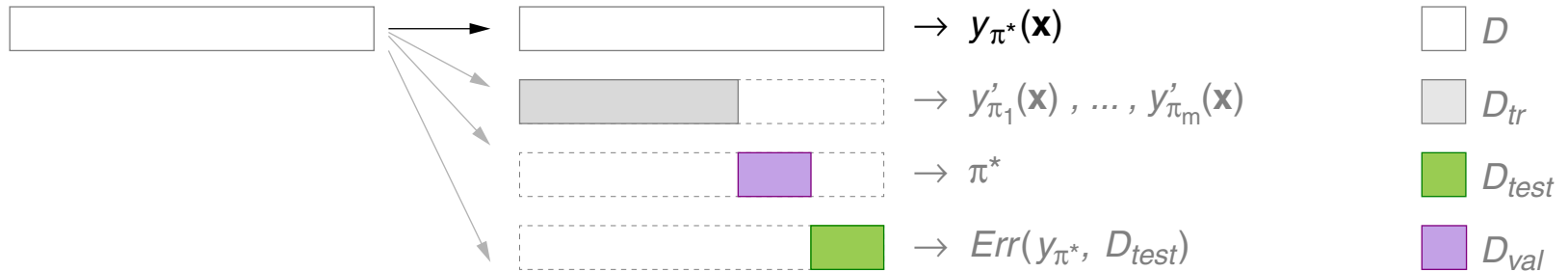


Evaluation setting:

- $D_{test} \subset D$ is the test set.
- $D_{val} \subset (D \setminus D_{test})$ is the validation set.
- $y_{\pi_l}'(\mathbf{x})$, $l = 1, \dots, m$, are the classifiers trained on $D_{tr} = D \setminus (D_{test} \cup D_{val})$.

Evaluating Effectiveness

Model Selection: Single Validation Set (continued)



Evaluation setting:

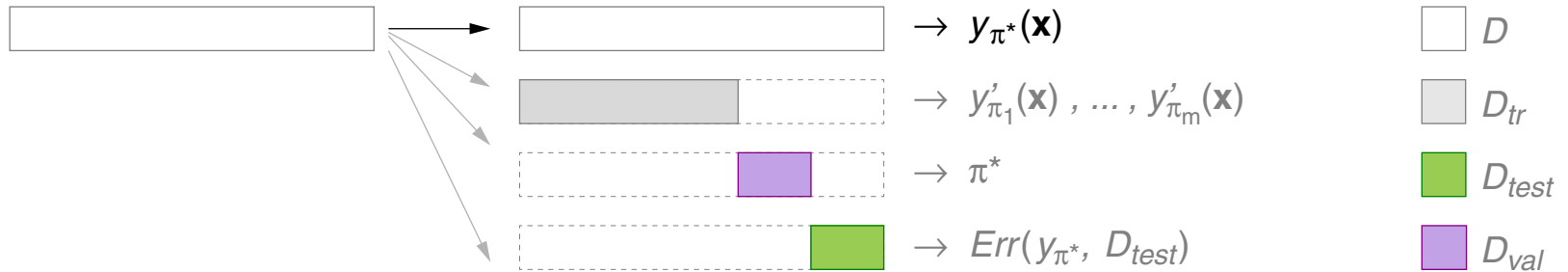
- $D_{test} \subset D$ is the test set.
- $D_{val} \subset (D \setminus D_{test})$ is the validation set.
- $y'_{\pi_l}(\mathbf{x}), l = 1, \dots, m$, are the classifiers trained on $D_{tr} = D \setminus (D_{test} \cup D_{val})$.

$$\square \pi^* = \underset{\pi_l, l=1, \dots, m}{\operatorname{argmin}} \frac{|\{(\mathbf{x}, c) \in D_{val} : y'_{\pi_l}(\mathbf{x}) \neq c\}|}{|D_{val}|}$$

⋮

Evaluating Effectiveness

Model Selection: Single Validation Set (continued)



Evaluation setting:

⋮

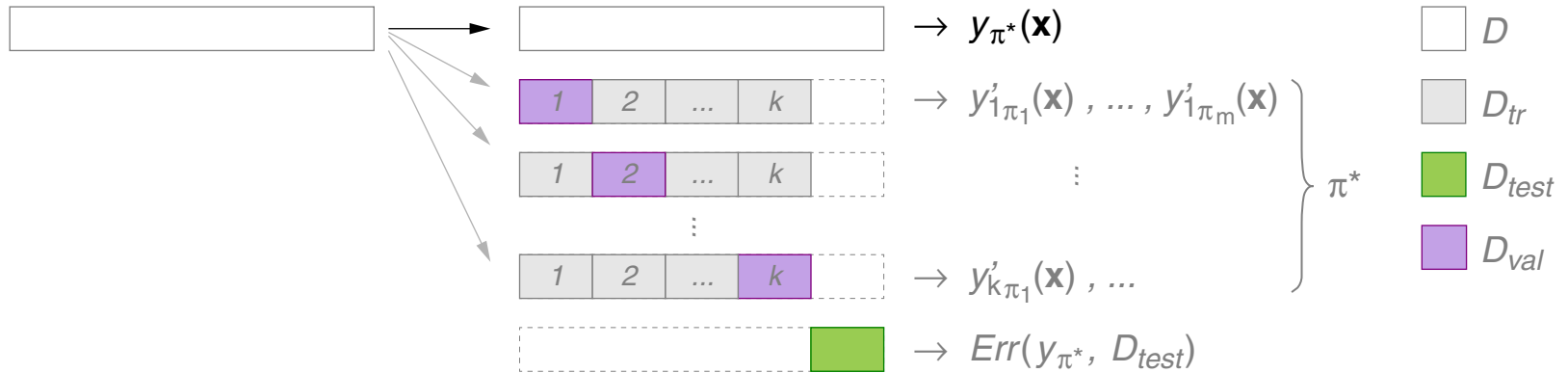
- $y_{\pi^*}(\mathbf{x})$ is the classifier trained on D .
- $y'_{\pi^*}(\mathbf{x})$ is the classifier trained on $D_{tr} = D \setminus D_{test}$.

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

$$\begin{aligned} \square \quad Err(y_{\pi^*}, D_{test}) &= \frac{|\{(\mathbf{x}, c) \in D_{test} : y'_{\pi^*}(\mathbf{x}) \neq c\}|}{|D_{test}|} \\ &= \text{misclassification rate of } y'_{\pi^*}(\mathbf{x}) \text{ on the test set.} \end{aligned}$$

Evaluating Effectiveness

Model Selection: k validation sets

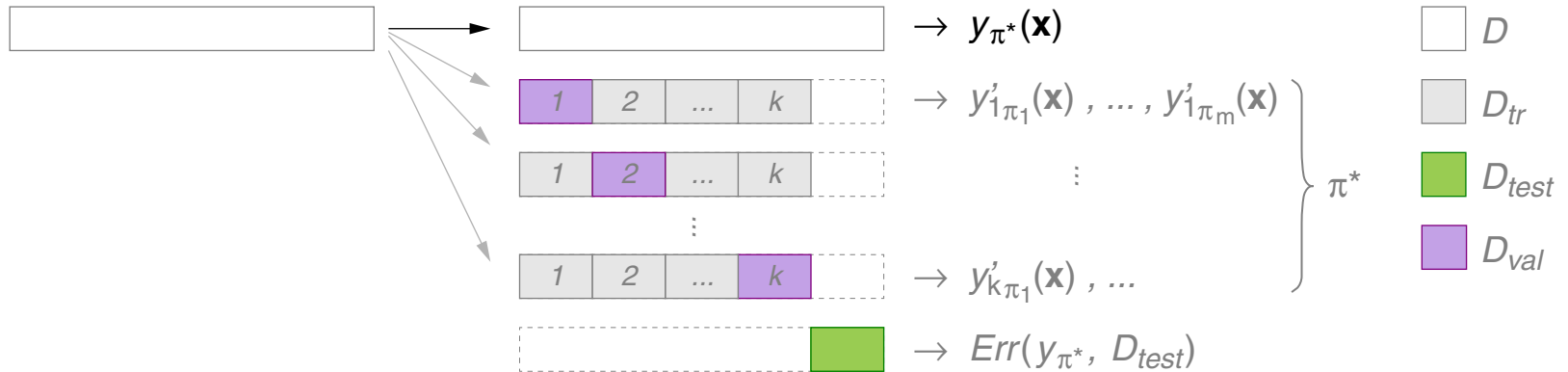


Evaluation setting:

- $D_{test} \subset D$ is the test set.
- k validation sets D_{val_i} by splitting $D \setminus D_{test}$ into k disjoint sets of similar size.
- $y'_{i\pi_l}(\mathbf{x})$, $i = 1, \dots, k$, $l = 1, \dots, m$, are the $k \cdot m$ classifiers trained on $D_{tr} = D \setminus (D_{test} \cup D_{val_i})$.

Evaluating Effectiveness

Model Selection: k validation sets (continued)



Evaluation setting:

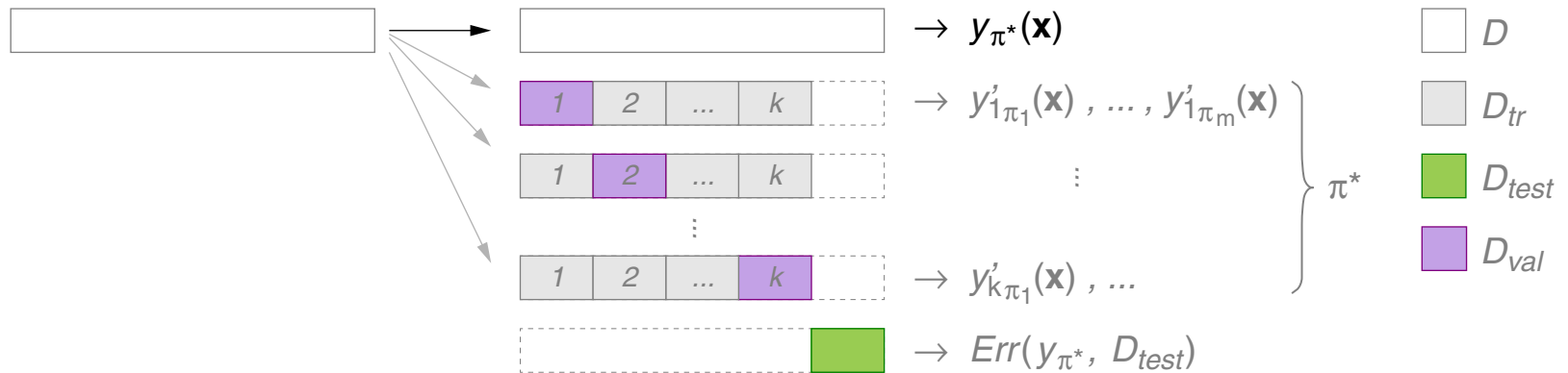
- $D_{test} \subset D$ is the test set.
- k validation sets D_{val_i} by splitting $D \setminus D_{test}$ into k disjoint sets of similar size.
- $y'_{i\pi_l}(\mathbf{x})$, $i = 1, \dots, k$, $l = 1, \dots, m$, are the $k \cdot m$ classifiers trained on $D_{tr} = D \setminus (D_{test} \cup D_{val_i})$.

$$\pi^* = \underset{\pi_l, l=1, \dots, m}{\operatorname{argmin}} \sum_{i=1}^k \frac{|\{(\mathbf{x}, c) \in D_{val_i} : y'_{i\pi_l}(\mathbf{x}) \neq c\}|}{|D_{val_i}|}$$

⋮

Evaluating Effectiveness

Model Selection: k validation sets (continued)



Evaluation setting:

- $y_{\pi^*}(\mathbf{x})$ 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.

Remarks:

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

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 :

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

Holdout error of y based on misclassification costs:

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

Remarks:

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

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