Chapter ML:II

II. Machine Learning Basics

- Rule-Based Learning of Simple Concepts
- 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 $\alpha : O \to \mathbf{X}$, C a set of classes, $y : \mathbf{X} \to C$ a classifier, and $\gamma : O \to C$ 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 $c_{\mathbf{x}} = \gamma(o)$, $o \in O$.

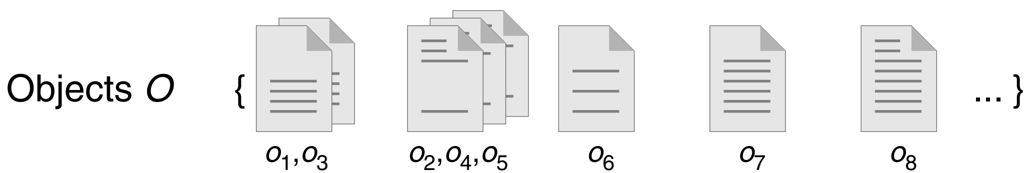
Evaluating Effectiveness

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 $\alpha : O \to \mathbf{X}$, C a set of classes, $y : \mathbf{X} \to C$ a classifier, and $\gamma : O \to C$ the ideal classifier to be approximated by $y()$.

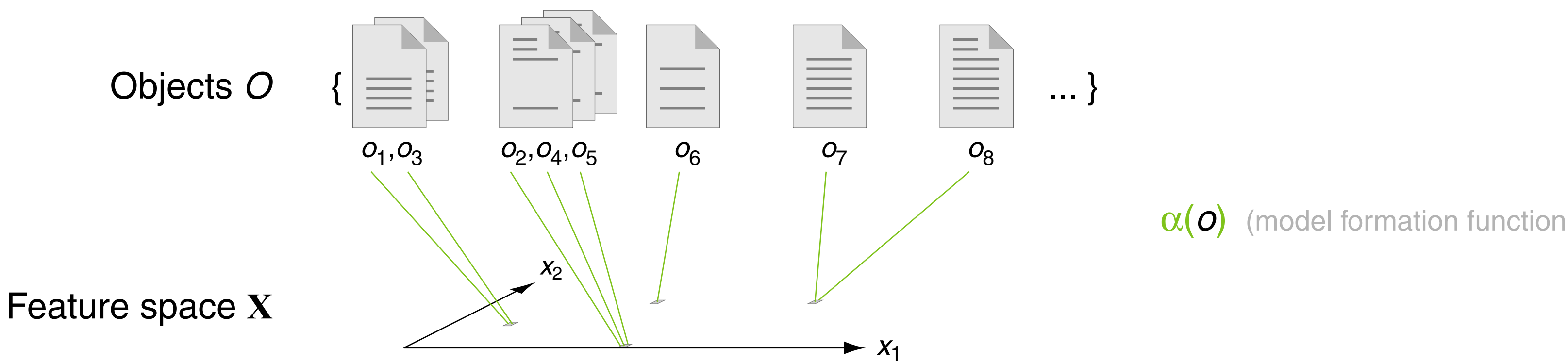
Let $X = \{\mathbf{x} \mid \mathbf{x} = \alpha(o), o \in O\}$ be a multiset of feature vectors and $c_\mathbf{x} = \gamma(o)$, $o \in O$.



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 $\alpha : O \to \mathbf{X}$, C a set of classes, $y : \mathbf{X} \to C$ a classifier, and $\gamma : O \to C$ the ideal classifier to be approximated by $y()$.

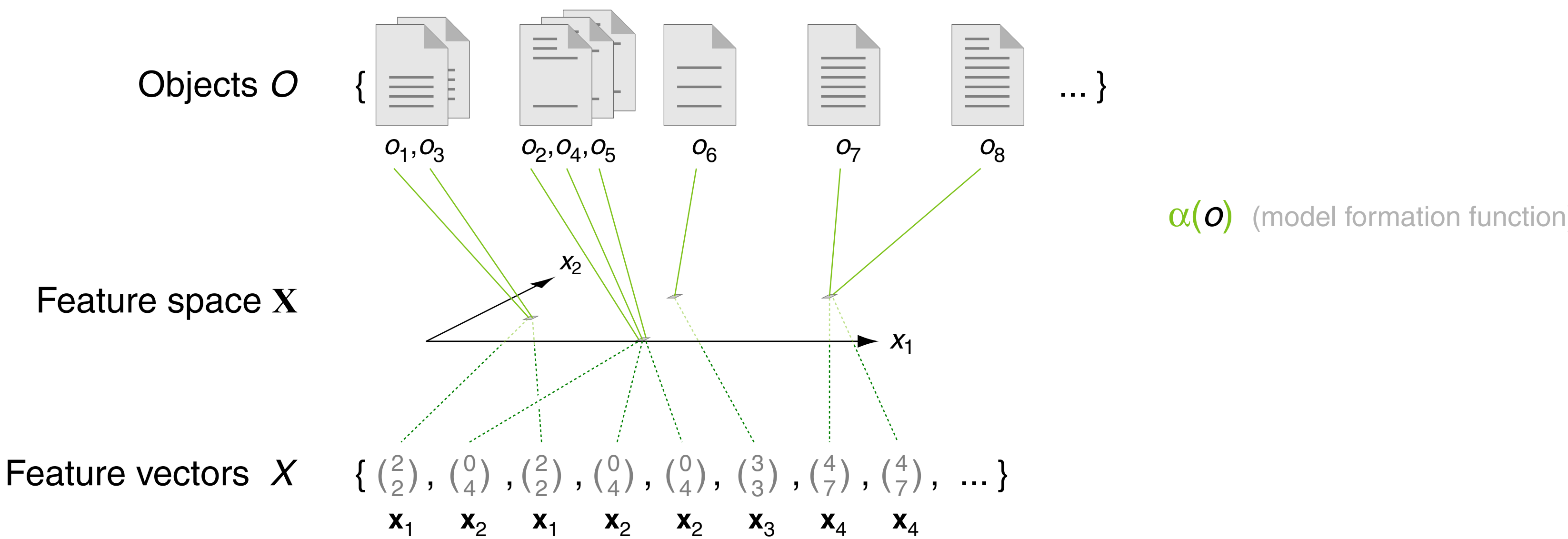
Let $X = \{\mathbf{x} \mid \mathbf{x} = \alpha(o), o \in O\}$ be a multiset of feature vectors and $c_{\mathbf{x}} = \gamma(o)$, $o \in O$.



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 $\alpha : O \to \mathbf{X}$, C a set of classes, $y : \mathbf{X} \to C$ a classifier, and $\gamma : O \to C$ 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 $c_{\mathbf{x}} = \gamma(o)$, $o \in O$.



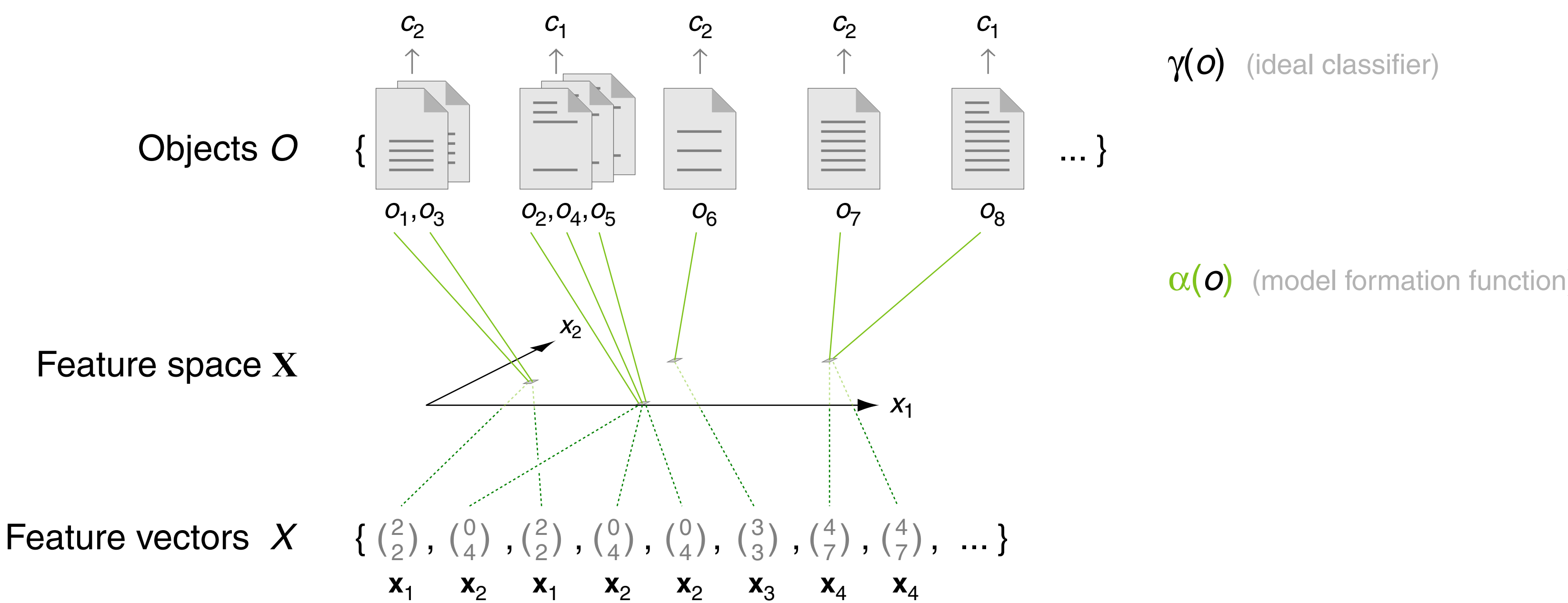
Evaluating Effectiveness

Misclassification Rate [probabilistic foundation]

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 $\alpha : O \to \mathbf{X}$, C a set of classes, $y : \mathbf{X} \to C$ a classifier, and $\gamma : O \to C$ 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 $c_{\mathbf{x}} = \gamma(o)$, $o \in O$.



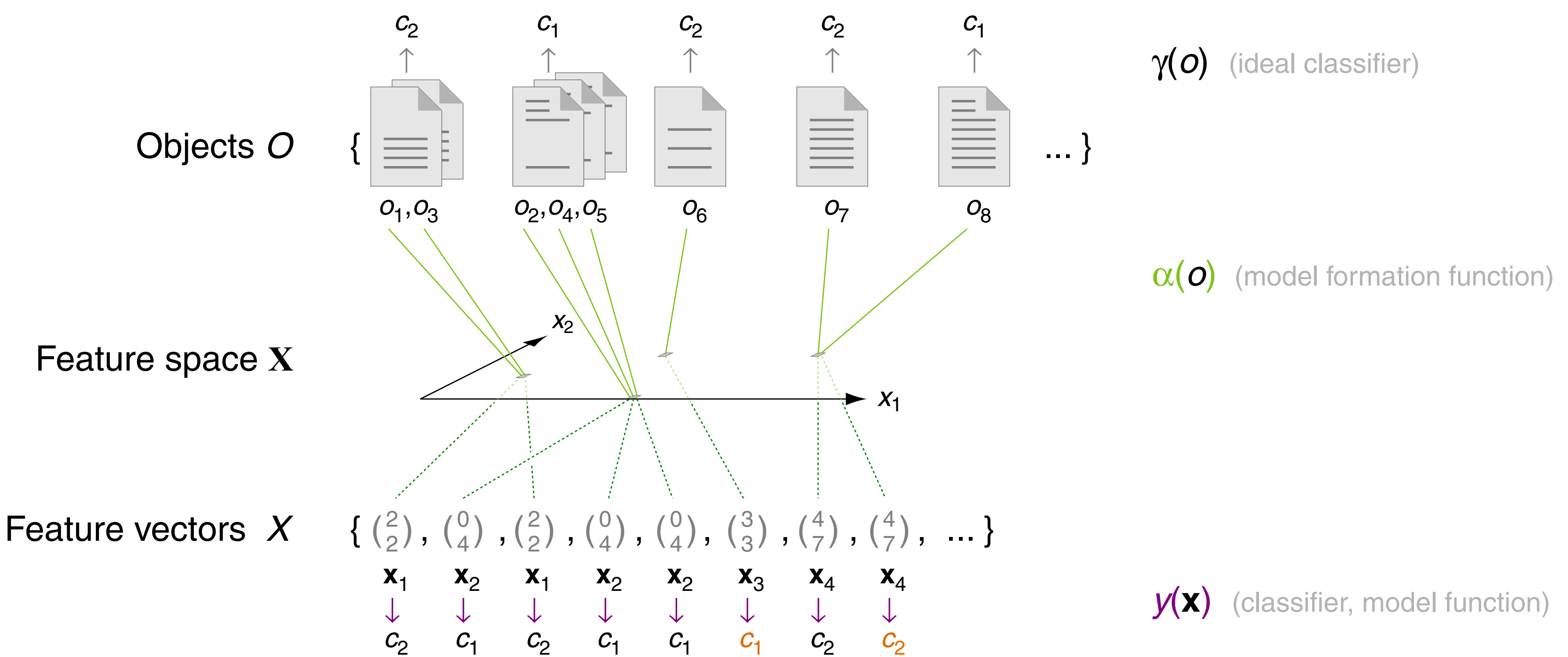
Evaluating Effectiveness

Misclassification Rate [probabilistic foundation]

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 $\alpha : O \to \mathbf{X}$, C a set of classes, $y : \mathbf{X} \to C$ a classifier, and $\gamma : O \to C$ 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 $c_{\mathbf{x}} = \gamma(o)$, $o \in O$.



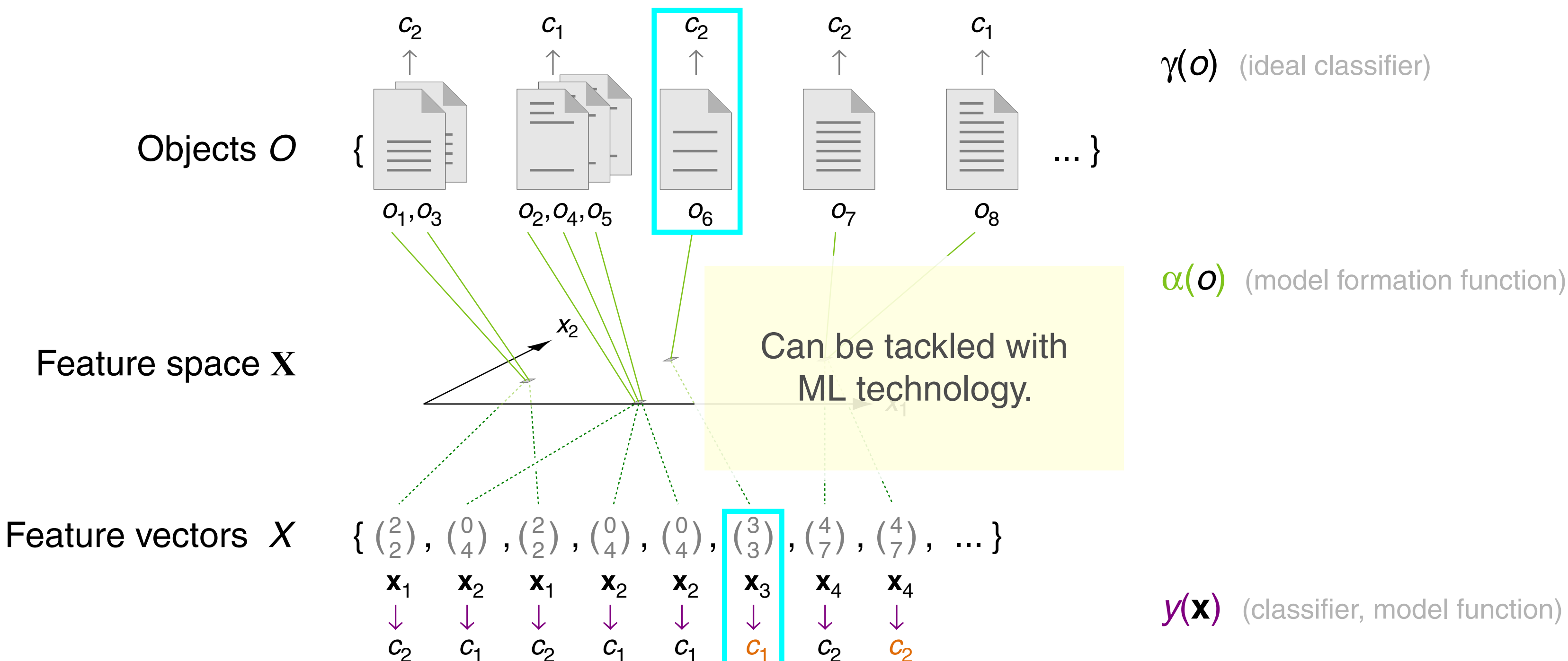
Evaluating Effectiveness

Misclassification Rate [probabilistic foundation]

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 $\alpha : O \to \mathbf{X}$, C a set of classes, $y : \mathbf{X} \to C$ a classifier, and $\gamma : O \to C$ 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 $c_{\mathbf{x}} = \gamma(o)$, $o \in O$.



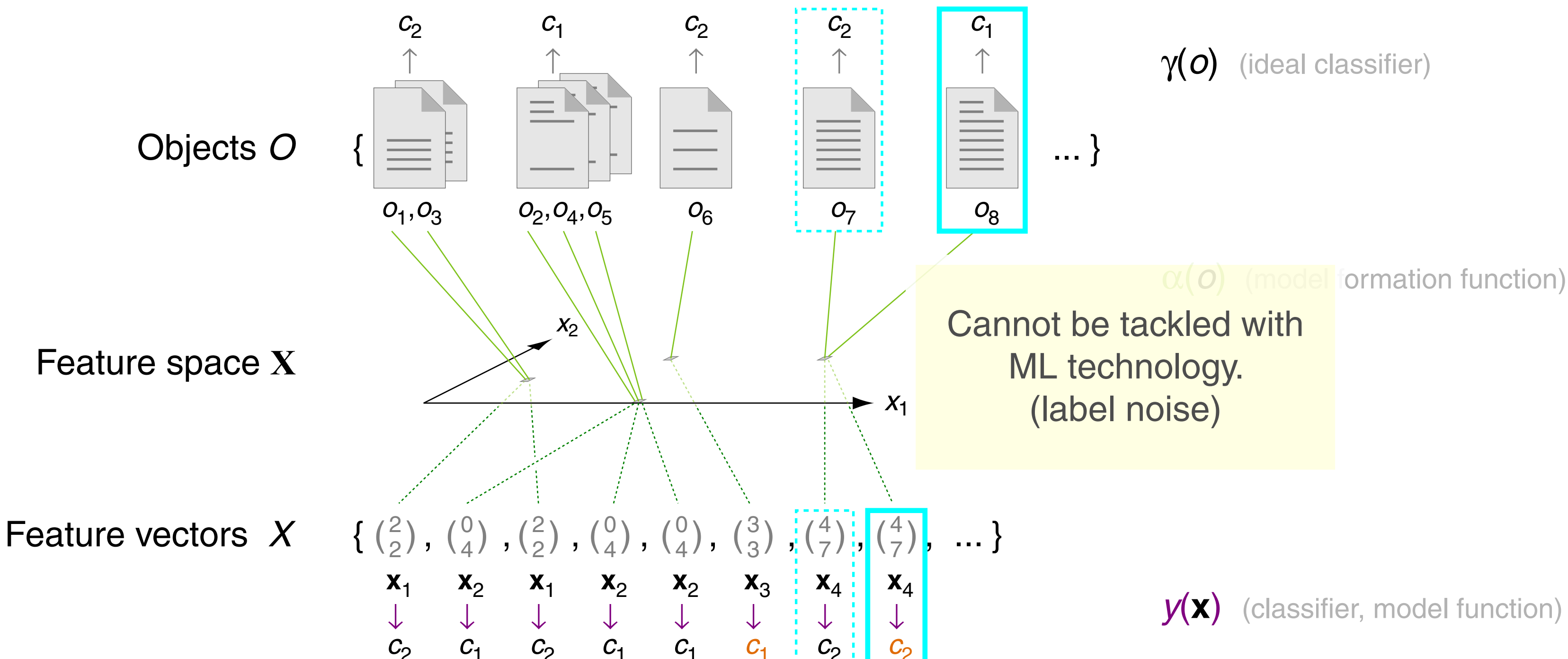
Evaluating Effectiveness

Misclassification Rate [probabilistic foundation]

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 $\alpha : O \to \mathbf{X}$, C a set of classes, $y : \mathbf{X} \to C$ a classifier, and $\gamma : O \to C$ 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 $c_{\mathbf{x}} = \gamma(o)$, $o \in O$.



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 $\alpha : O \to \mathbf{X}$, C a set of classes, $y : \mathbf{X} \to C$ a classifier, and $\gamma : O \to C$ 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 $y()$, denoted $Err^*(y())$, is defined as follows:

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

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 $\alpha : O \to \mathbf{X}$, C a set of classes, $y : \mathbf{X} \to C$ a classifier, and $\gamma : O \to C$ 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 $y()$, denoted $\mathit{Err}^*(y())$, is defined as follows:

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

Problem:

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

$\leadsto$ Based on a multiset of examples D, estimation of upper and lower bounds for $\mathit{Err}^*(y())$ 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.

- 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“.

Remarks: (continued)

- The previous definition of *Err*$^*(y())$ is "frequency-based": Information regarding the distribution of feature vectors and classes is estimated from the multiset of feature vectors, X, or examples, D, respectively.

  Instead of defining *Err*$^*(y())$ as the ratio of misclassified feature vectors in X or D, the definition of *Err*$^*(y())$ can be probabilistically founded via a probability measure P, that is, the explicit specification of a joint distribution of feature vectors and classes. In this regard, we introduce the following random variables:

  X : multivariate random variable whose instances are feature vectors

  C : random variable whose instances are class labels

- 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 = domain of the feature vectors

  X : multiset of feature vectors

Evaluating Effectiveness

Misclassification Rate (continued)

Definition 9 (Probabilistic Foundation of the True Misclassification Rate)

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

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

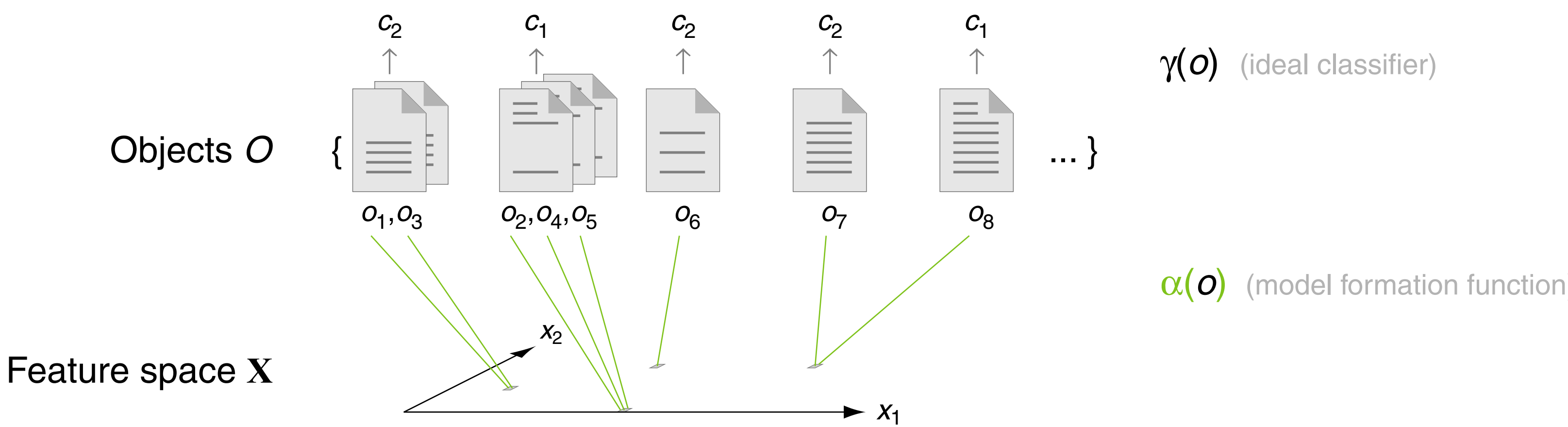
Evaluating Effectiveness

Misclassification Rate (continued)

Definition 9 (Probabilistic Foundation of the True Misclassification Rate)

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

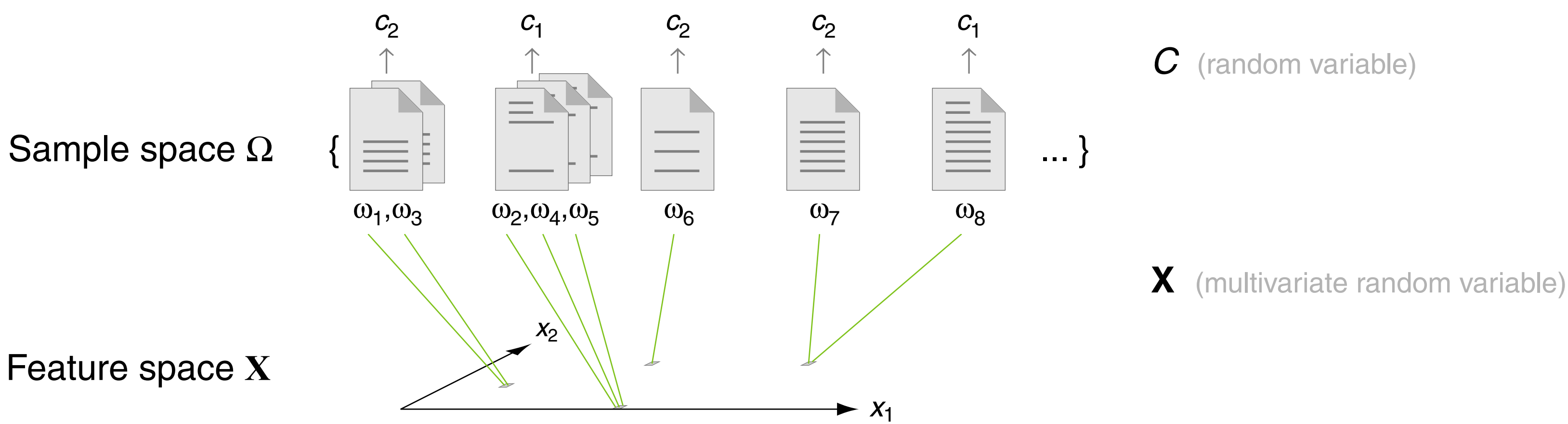
We consider two types of random variables, $\mathbf{X} : \Omega \to \mathbf{X}$, and $C : \Omega \to C$.



Definition 9 (Probabilistic Foundation of the True Misclassification Rate)

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

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



Probabilities $P(\mathbf{X}=\mathbf{x})$, $P(C=c)$, $P(\mathbf{X}=\mathbf{x}, C=c)$, $P(\mathbf{X}=\mathbf{x} \mid C=c)$

$p(\mathbf{x})$, $p(c)$, $p(\mathbf{x}, c)$, $p(\mathbf{x} \mid c)$

Definition 9 (Probabilistic Foundation of the True Misclassification Rate)

Let Ω be sample space, which corresponds to a set O of real-world objects, and P a probability measure defined on $\mathcal{P}(\Omega)$. Moreover, 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.

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

Then $p(\mathbf{x}, c)$, $p(\mathbf{x}, c) := P(\boldsymbol{X}{=}\mathbf{x}, C{=}c)$, 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 $y()$ can be expressed as follows:

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

Evaluating Effectiveness

Misclassification Rate (continued) [frequency-based foundation]

Definition 9 (Probabilistic Foundation of the True Misclassification Rate)

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

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

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 $y()$ can be expressed as follows:

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

Problem:

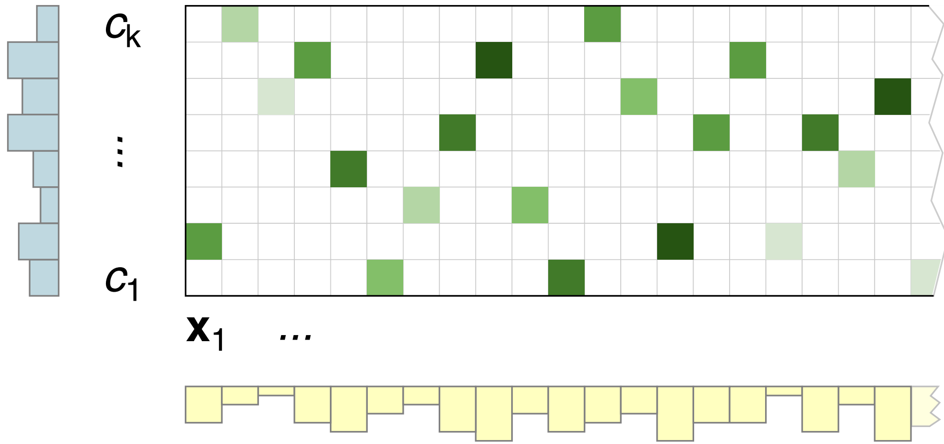
- Usually P and hence $p(\mathbf{x}, c)$ is unknown.

$\rightsquigarrow$ Based on D estimate $p(\mathbf{x} \mid c)$ under the Naive Bayes assumption.

Evaluating Effectiveness

Illustration 1: Label Noise

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



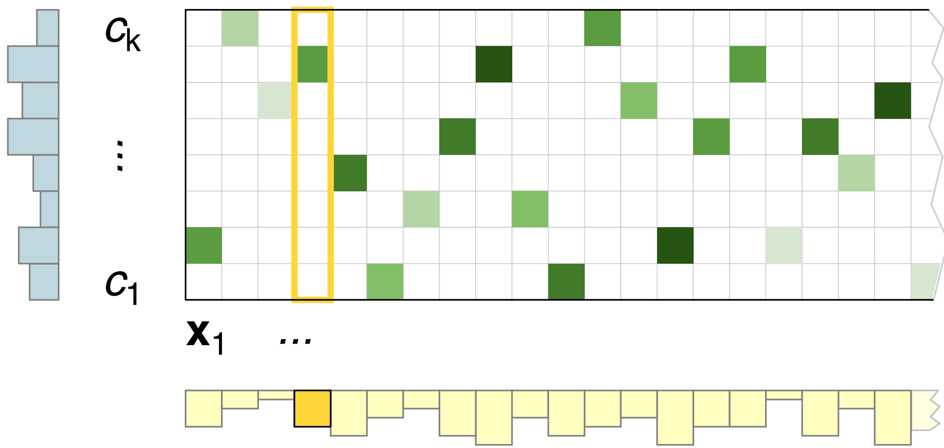
(no label noise $\rightarrow$ classes are unique)

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

Evaluating Effectiveness

Illustration 1: Label Noise

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



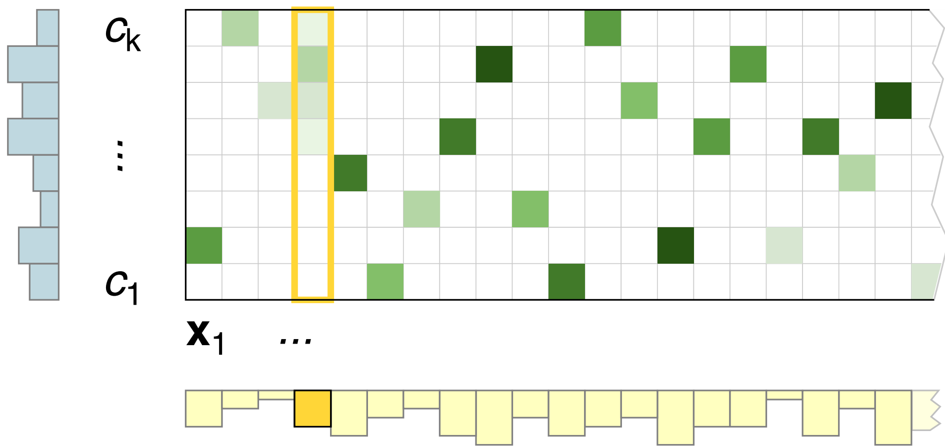
(no label noise $\rightarrow$ classes are unique)

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

Evaluating Effectiveness

Illustration 1: Label Noise

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



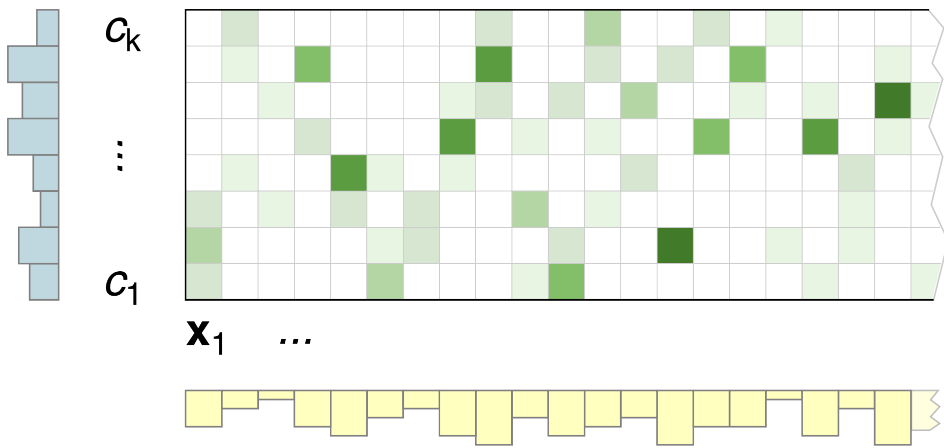
(label noise → classes are *not unique*)

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

Evaluating Effectiveness

Illustration 1: Label Noise

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



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

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

Remarks:

- **X** and *C* denote (multivariate) random variables with ranges $\mathbf{X}$ and C respectively.

  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, $c = \gamma(o)$.

- **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 **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 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 (probability basics) :

- $P()$ is a probability measure (see section Probability Basics in part Bayesian Learning) and its argument is an event. Examples for events are "**X**$=\mathbf{x}$", "**X**$=\mathbf{x}$, $C$$=c$", or "**X**$=\mathbf{x} \mid$ $C$$=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: $p()$ is defined via $P()$. [illustration]

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

- Since $p(\mathbf{x}, c)$ (and similarly $p(\mathbf{x})$, $p(\mathbf{x} \mid c)$, etc.) is defined as $P($**X**$=\mathbf{x}$, $C$$=c)$, the respective expressions for $p()$ and $P()$ can usually be used interchangeably. In this sense we have two parallel notations, arguing about realizations of random variables and events respectively.

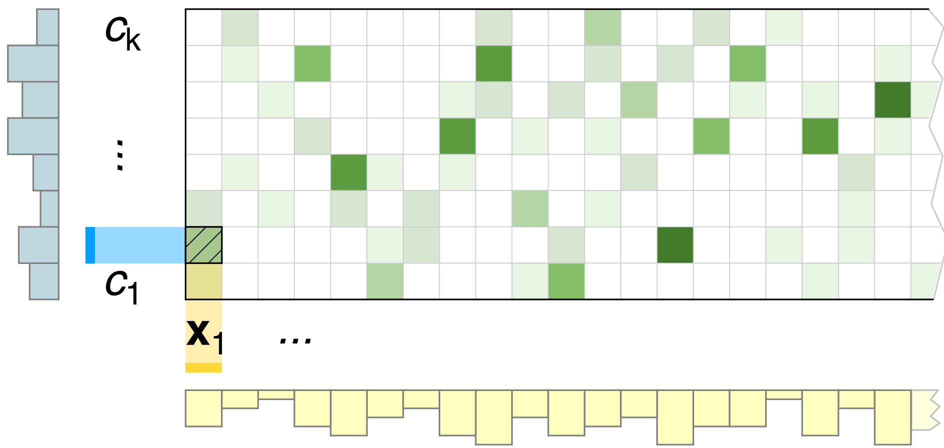
- Let A and B denote two events, e.g., $A =$ "**X**$=\mathbf{x}_9$" and $B =$ "$C$$=c_3$". Then the following expressions are equivalent notations for the probability of the joint event "A and B" : $P(A, B)$, $P(A \wedge B)$, $P(A \cap B)$.

- $I_{\neq}$ 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

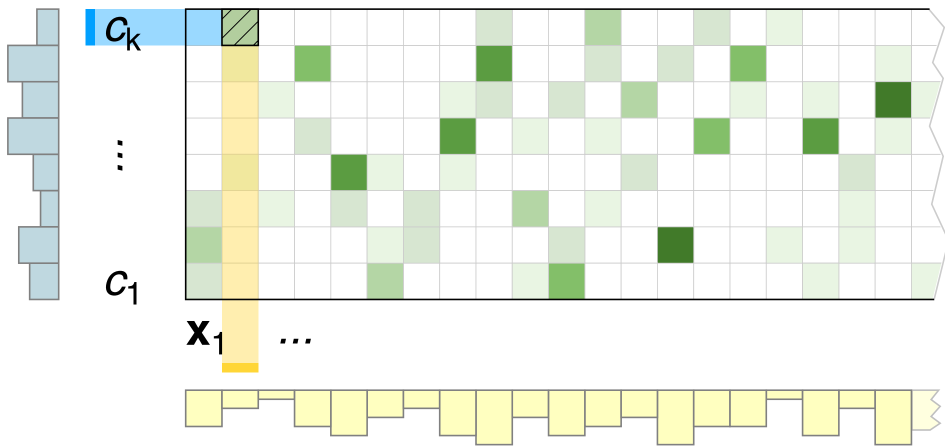
The Bayes classifier returns for $\mathbf{x}$ the class with the highest [posterior] probability:



Evaluating Effectiveness

Illustration 2: Bayes [Optimal] Classifier and Bayes Error

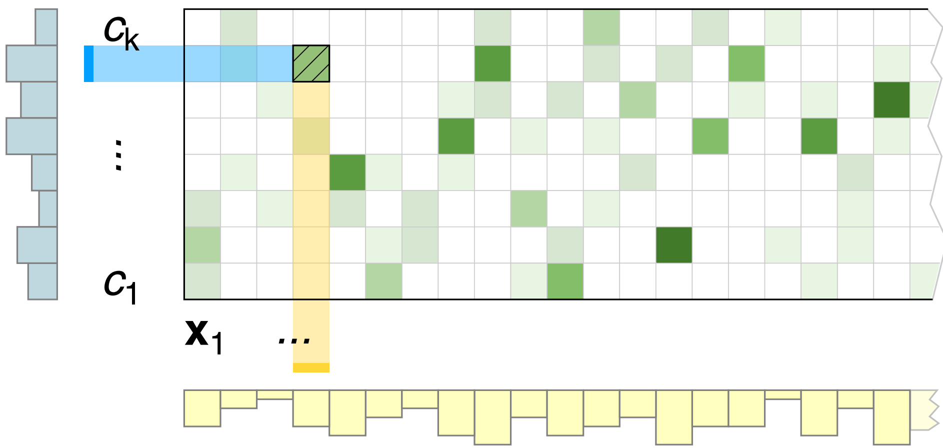
The Bayes classifier returns for $\mathbf{x}$ the class with the highest [posterior] probability:



Evaluating Effectiveness

Illustration 2: Bayes [Optimal] Classifier and Bayes Error

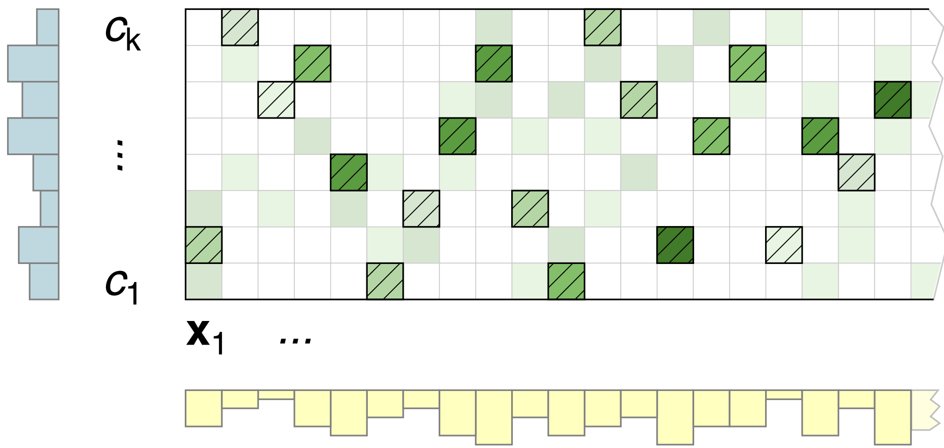
The Bayes classifier returns for $\mathbf{x}$ the class with the highest [posterior] probability:



Evaluating Effectiveness

Illustration 2: Bayes [Optimal] Classifier and Bayes Error

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

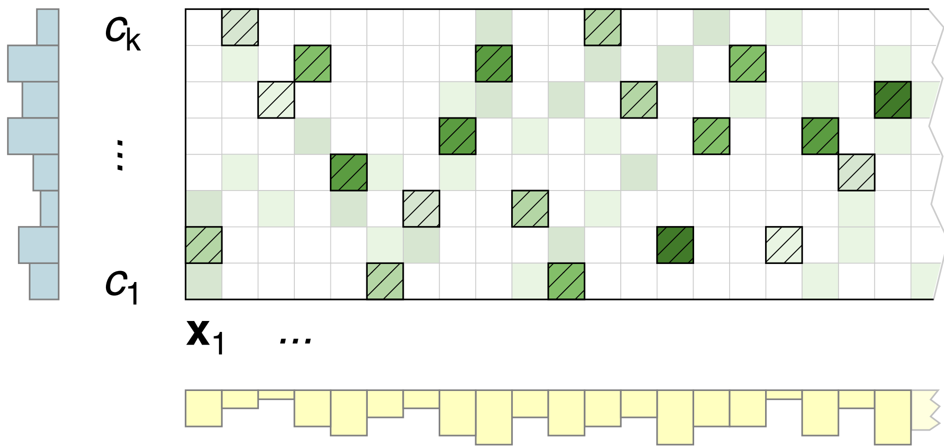


Bayes classifier: $y^*(\mathbf{x}) = \underset{c \in C}{\operatorname{argmax}}\, p(c, \mathbf{x}) = \underset{c \in C}{\operatorname{argmax}}\, p(c \mid \mathbf{x})$

Evaluating Effectiveness

Illustration 2: Bayes [Optimal] Classifier and Bayes Error

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



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

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

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 $p(c, \mathbf{x})$.

Remarks (Bayes classifier) : (continued)

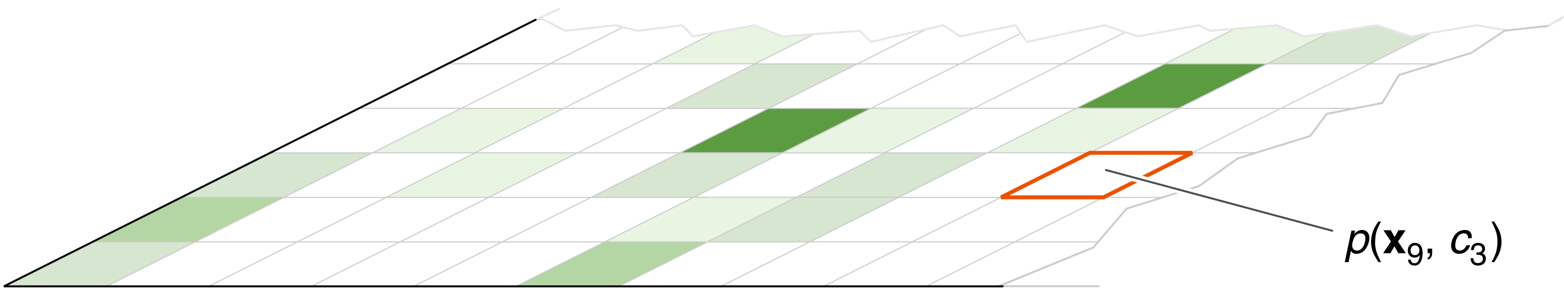
- Prerequisite to construct the Bayes classifier and to compute its error is knowledge about the joint probabilities, $p(c, \mathbf{x})$ or $p(c \mid \mathbf{x})$. In this regard the size of the available data, D, decides about the possibility and the quality for the estimation of the probabilities.

- 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

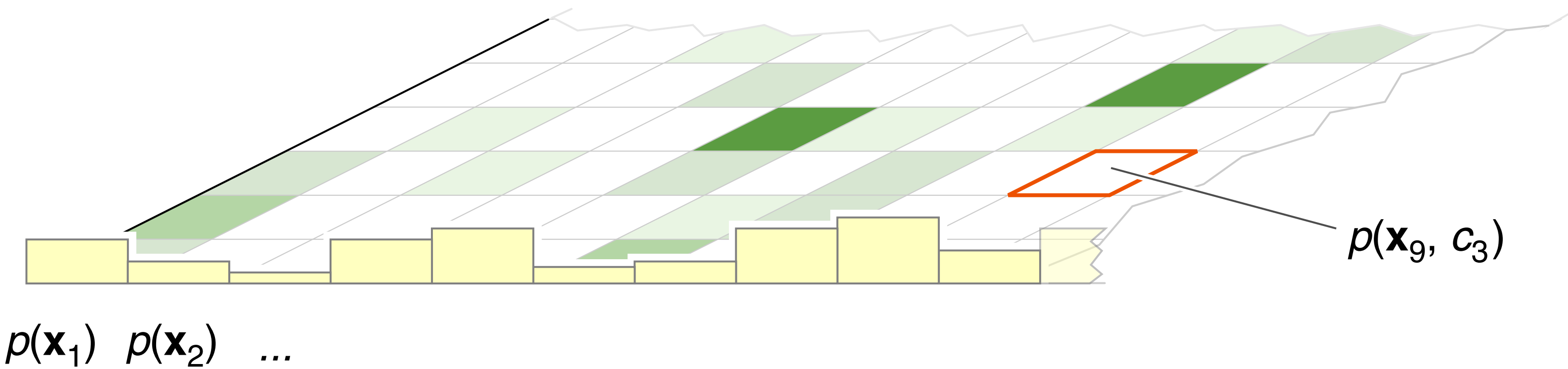
Joint probabilities $p(\mathbf{x}, c)$, shading indicates magnitude:



Evaluating Effectiveness

Illustration 3: Marginal and Conditional Distributions

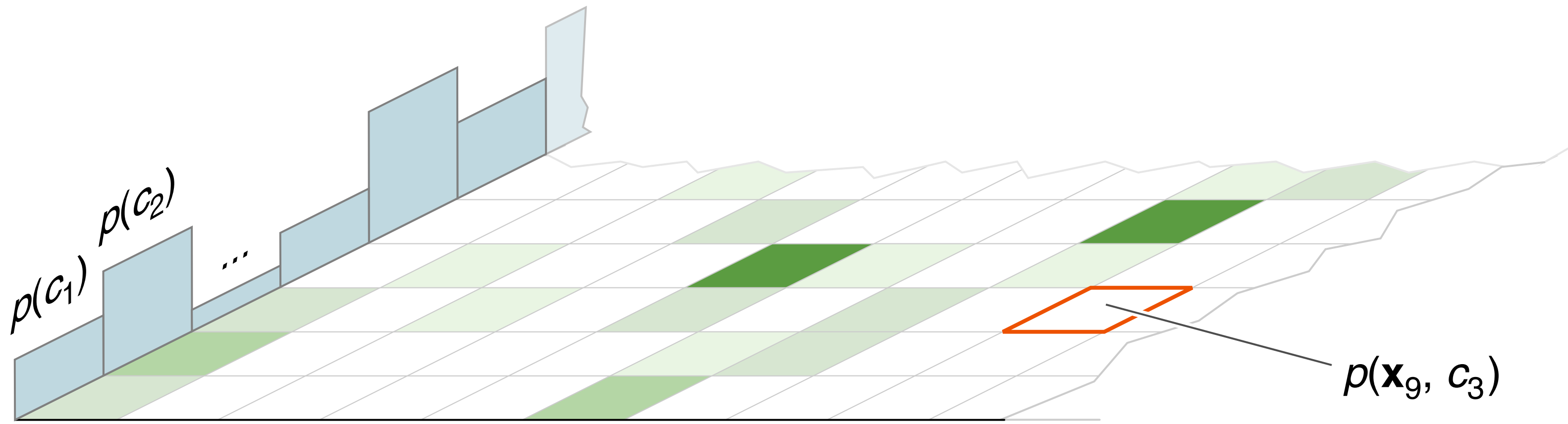
Marginal probabilities $p(\mathbf{x})$:



Evaluating Effectiveness

Illustration 3: Marginal and Conditional Distributions

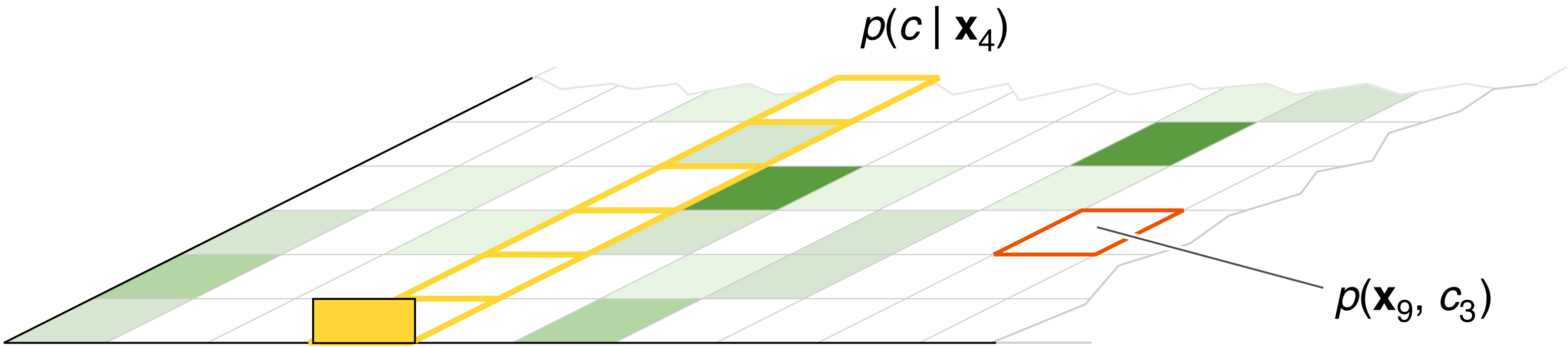
Marginal probabilities $p(c)$:



Evaluating Effectiveness

Illustration 3: Marginal and Conditional Distributions

Probabilities of the classes c under feature vector (the condition) $\mathbf{x}_4$, denoted as $p(c \mid \mathbf{x}_4)$:

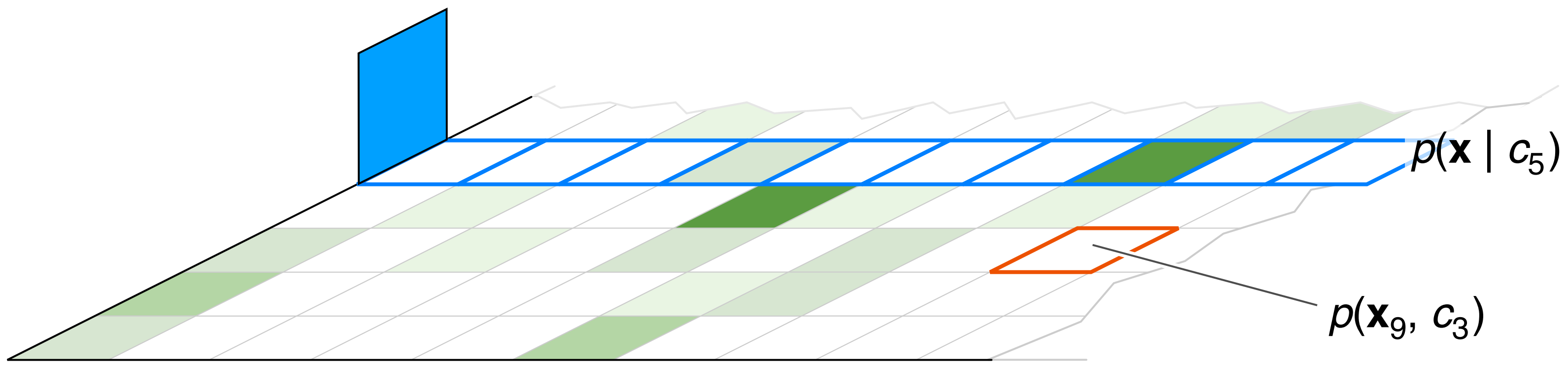


Conditional 'class probability function', CCPF.

Evaluating Effectiveness

Illustration 3: Marginal and Conditional Distributions

Probabilities of the feature vectors $\mathbf{x}$ under class (the condition) c_5, denoted as $p(\mathbf{x} \mid c_5)$:

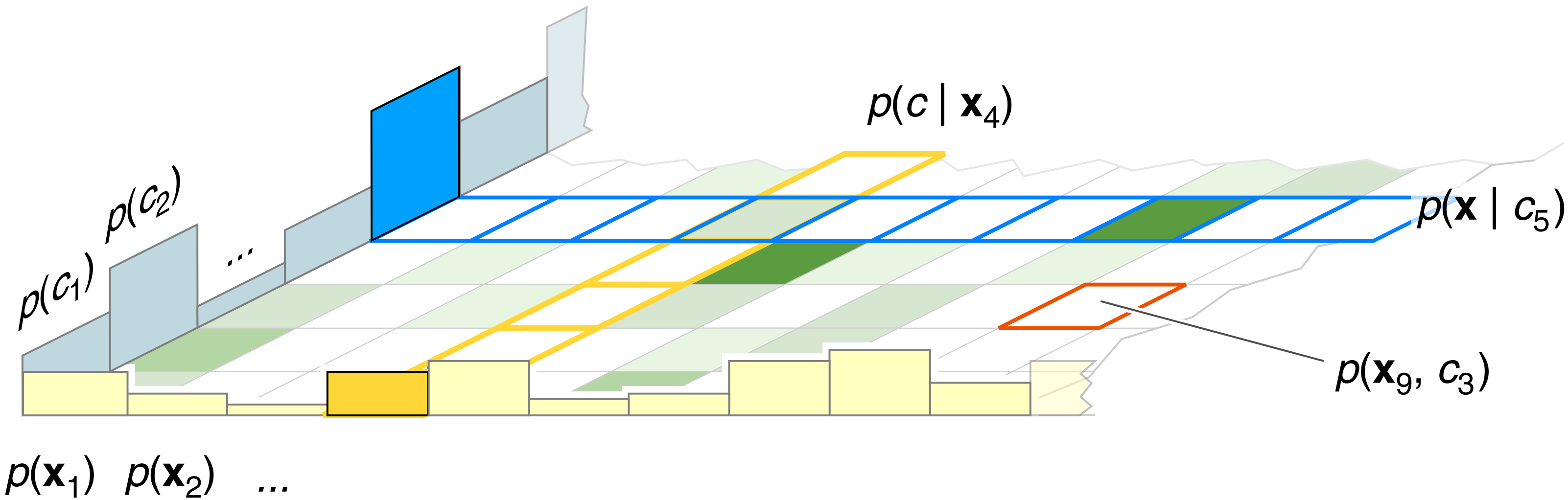


Class-conditional 'probability function', CPF.

Evaluating Effectiveness

Illustration 3: Marginal and Conditional Distributions

Overview:



Remarks:

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

  $p(c \mid \mathbf{x})$ 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 $p(\mathbf{x}_4)$. $p(c \mid \mathbf{x}_4)$ gives the probabilities of the c (consider the column) under feature vector $\mathbf{x}_4$ (= having normalized by $p(\mathbf{x}_4)$), i.e., $p(\mathbf{x}_4, c)/p(\mathbf{x}_4)$.

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

  $p(\mathbf{x} \mid c)$ 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 $p(c_5)$. $p(\mathbf{x} \mid c_5)$ gives the probabilities of the $\mathbf{x}$ (consider the row) under class c_5 (= having normalized by $p(c_5)$), i.e., $p(\mathbf{x}, c_5)/p(c_5)$.

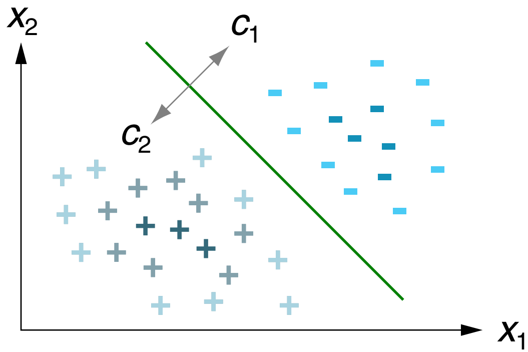
- $p(\mathbf{x}, c) = p(c, \mathbf{x}) = p(c \mid \mathbf{x}) \cdot p(\mathbf{x})$, where $p(\mathbf{x})$ is the prior probability for event $\textbf{X}{=}\mathbf{x}$, and $p(c \mid \mathbf{x})$ is the probability for event $\mathit{C}{=}c$ given event $\textbf{X}{=}\mathbf{x}$. Likewise, $p(\mathbf{x},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

- Let the events $\textbf{X}{=}\mathbf{x}$ and $\mathit{C}{=}c$ have occurred, and, let $\mathbf{x}$ be known and c be unknown. Then, $p(\mathbf{x} \mid c)$ is called *likelihood* (for event $\textbf{X}{=}\mathbf{x}$ given event $\mathit{C}{=}c$). [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 Classifiction Setting

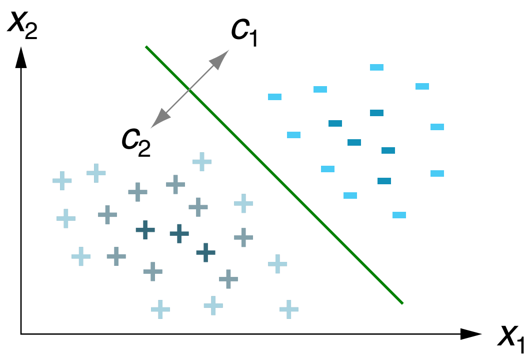


$$X = \left\{ \begin{pmatrix} x_{1_1} \\ x_{1_2} \end{pmatrix}, \begin{pmatrix} x_{2_1} \\ x_{2_2} \end{pmatrix}, \ \ldots \right\}, \quad \mathbf{X} = \mathbf{R}^2$$

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

Evaluating Effectiveness

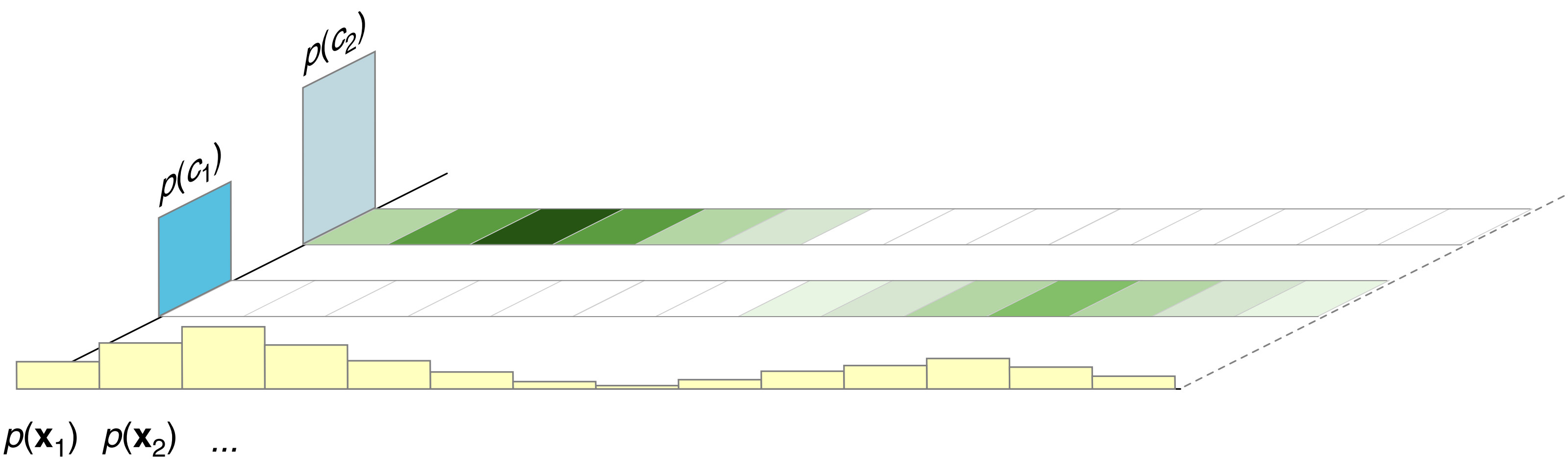
Illustration 4: Probability Distribution in a Classifiction Setting



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

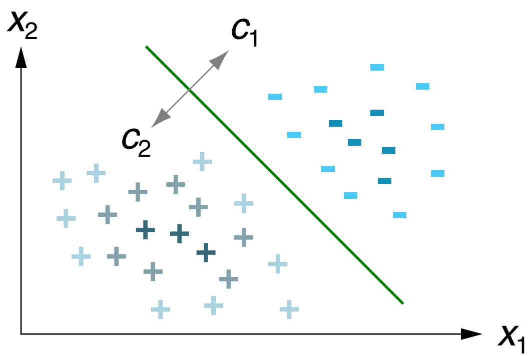
$$\mathbf{x}_1 \qquad \mathbf{x}_2 \qquad \ldots$$

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



Evaluating Effectiveness

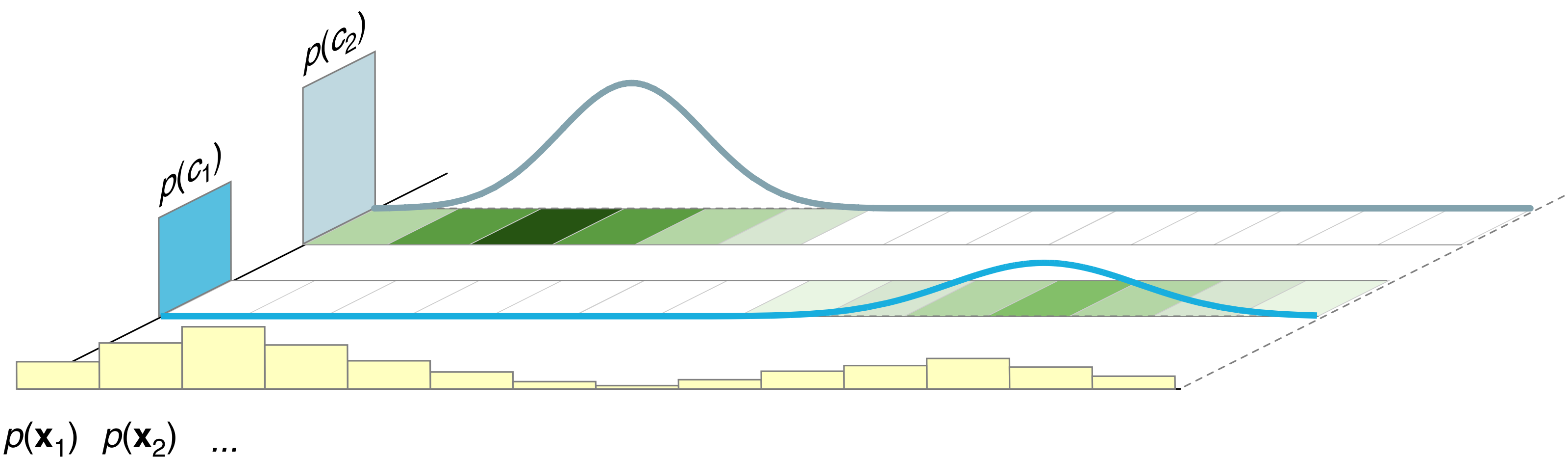
Illustration 4: Probability Distribution in a Classifiction Setting



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

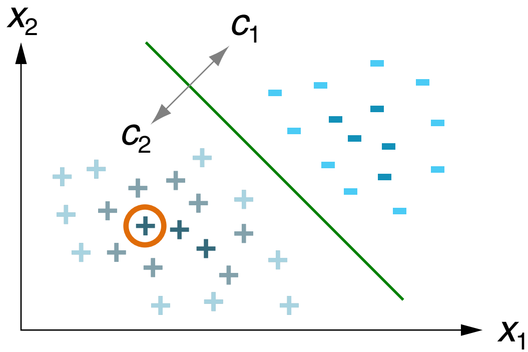
$$\mathbf{x}_1 \qquad \mathbf{x}_2 \qquad \ldots$$

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



Evaluating Effectiveness

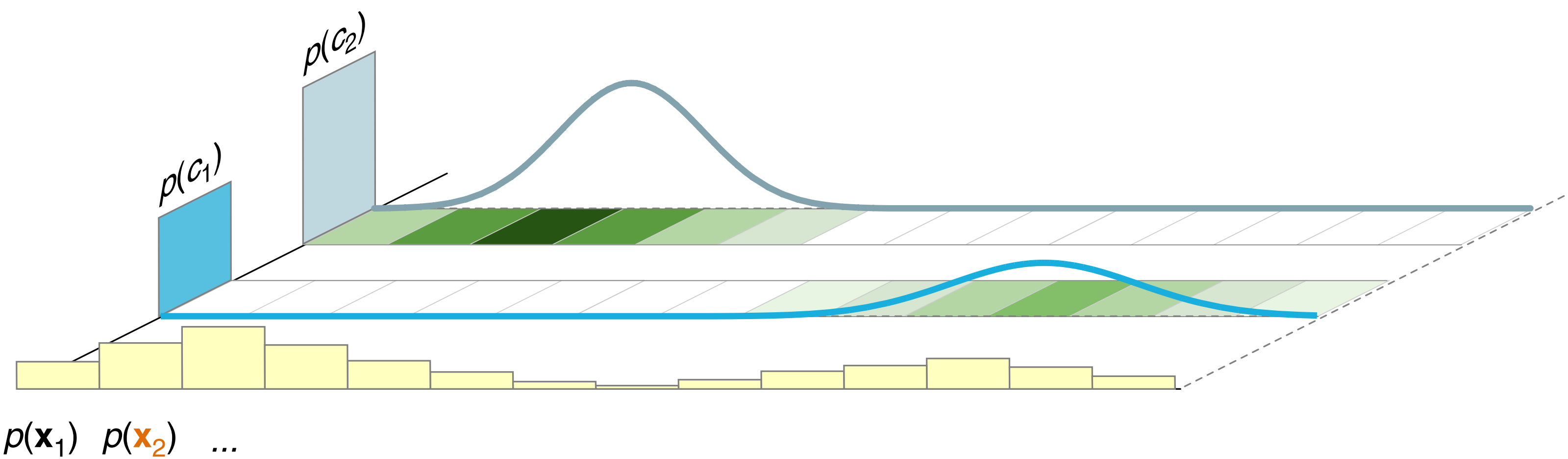
Illustration 4: Probability Distribution in a Classifiction Setting



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

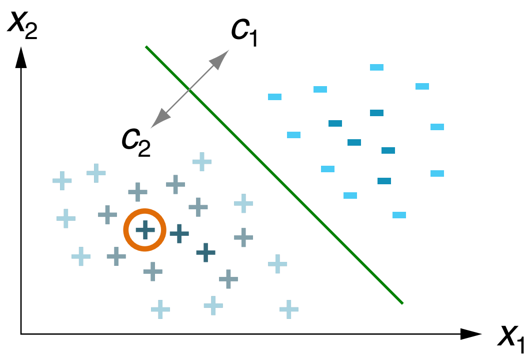
$$\mathbf{x}_1 \qquad \mathbf{x}_2 \qquad \ldots$$

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



Evaluating Effectiveness

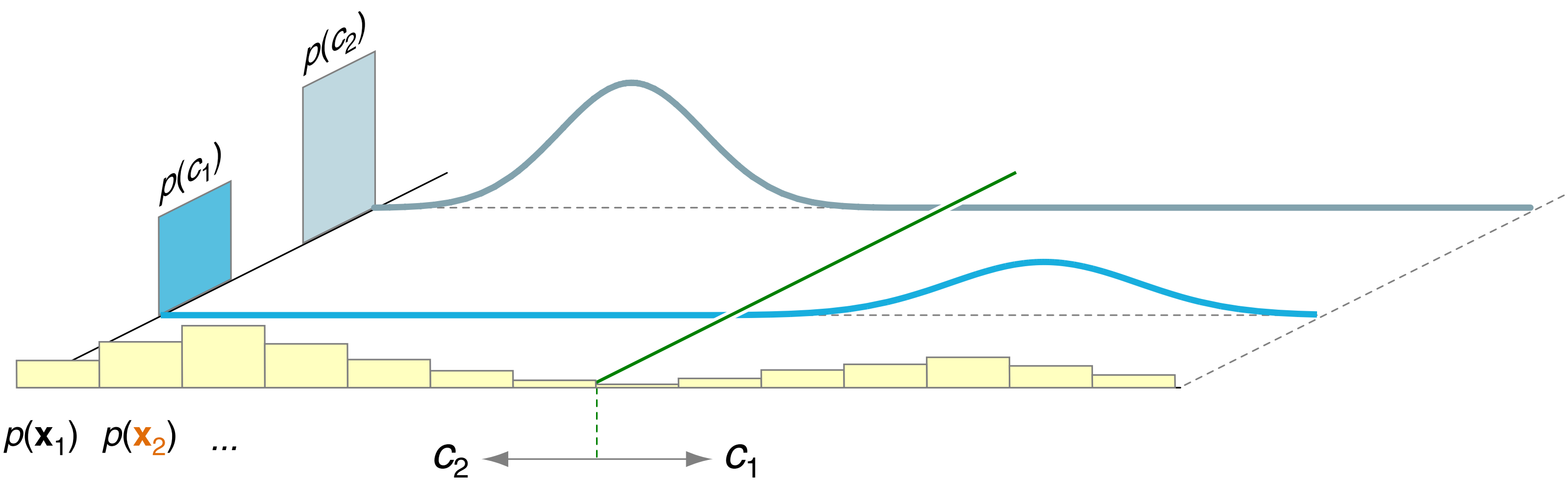
Illustration 4: Probability Distribution in a Classifiction Setting



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

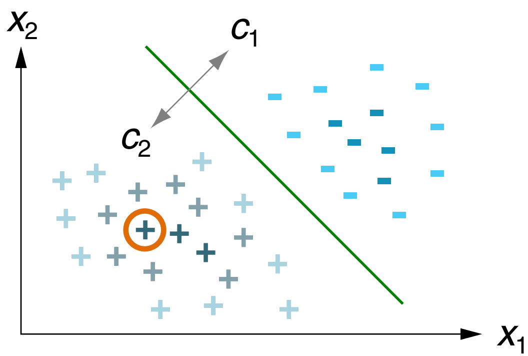
$\mathbf{x}_1 \quad \mathbf{x}_2 \quad \ldots$

Optimum hyperplane classifier:



Evaluating Effectiveness

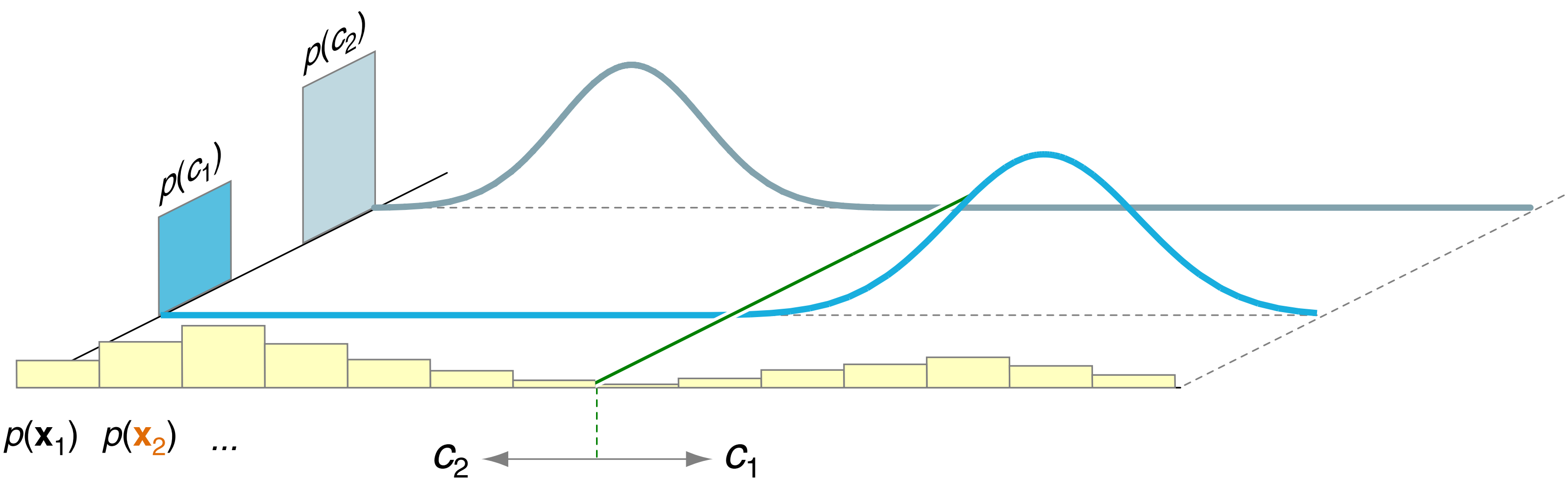
Illustration 4: Probability Distribution in a Classifiction Setting



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

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

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



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 $\mathbf{R}$, 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 [Comparing Model Variants]

Experiment setting:

- $D = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_n, c_n)\} \subseteq X \times C$ is a multiset of examples.
- $y()$ is the classifier trained on D.
- The true error *Err*$^*(y())$ measures the performance of $y()$ on X ("in the wild").
- Q: How can the true error *Err*$^*(y())$ be approximated?

Experiment setting:

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

The following relations typically hold:

Underestimation (likely)				Overestimation (unlikely)
Training error	Cross-validation error	Holdout error	True error	

$$\mathit{Err}(y(), D_{tr}) < \mathit{Err}(y(), D, k) \lesssim \mathit{Err}(y(), D_{test}) < \mathit{Err}^*(y()) < \mathit{Err}(y(), D, k) \lesssim \mathit{Err}(y(), D_{test})$$

Evaluating Effectiveness

Estimating Error Bounds [Comparing Model Variants]

Experiment setting:

- $D = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_n, c_n)\} \subseteq X \times C$ is a multiset of examples.
- $y()$ is the classifier trained on D.
- The true error $\mathit{Err}^*(y())$ measures the performance of $y()$ on X ("in the wild").
- Q: How can the true error $\mathit{Err}^*(y())$ be approximated?

The following relations typically hold:

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

$|D| \rightarrow |X|$

Evaluating Effectiveness

Estimating Error Bounds [Comparing Model Variants]

Experiment setting:

- $D = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_n, c_n)\} \subseteq X \times C$ is a multiset of examples.
- $y()$ is the classifier trained on D.
- The true error *Err*$^*(y())$ measures the performance of $y()$ on X ("in the wild").
- Q: How can the true error *Err*$^*(y())$ be approximated?

The following relations typically hold:

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

difference quantifies overfitting

Remarks:

- Notation of the different error estimation methods:

Training error: $Err(y(), D_{tr})$

data, used for training as well as to test

classifier

Holdout error: $Err(y(), D_{test})$, also: $Err(y())$

test data, different from training data

classifier

Cross-validation error: $Err(y(), D, k)$, also: $Err(y())$

split fraction $1/k$

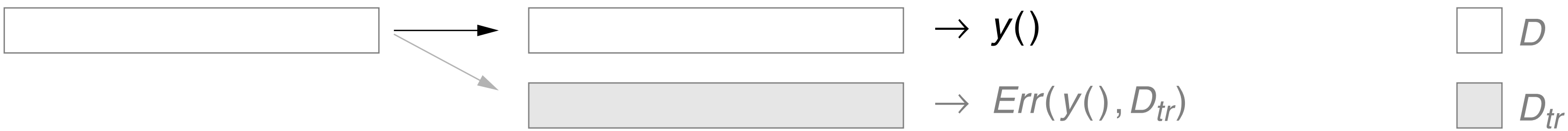
data, split into D_{tr} and D_{test} according to split fraction

classifier

- Relating the true error $Err^*(y())$ to the error estimations $Err(y(), D_{tr})$, $Err(y(), D, k)$, or $Err(y(), D_{test})$ is not straightforward but requires an in-depth analysis of the sampling strategy, the sample size D, and the set X of possible / typical / considered feature vectors.

Evaluating Effectiveness

Training Error



Evaluation setting:

- $y()$ is the classifier trained on $D_{tr} = D$.

Training error of $y()$:

- *Err*$(y(), D_{tr})$ $= \dfrac{|\{(\mathbf{x}, c) \in D_{tr} : y(\mathbf{x}) \neq c\}|}{|D_{tr}|}$

  $=$ misclassification rate of $y()$ on the training set.

Remarks (training error) :

- For the training error *Err*$(y(), D_{tr})$ holds that the same examples that are used for training $y()$ are also used to test $y()$. Hence *Err*$(y(), D_{tr})$ quantifies the *memorization* power of $y()$ but not its *generalization* power.

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

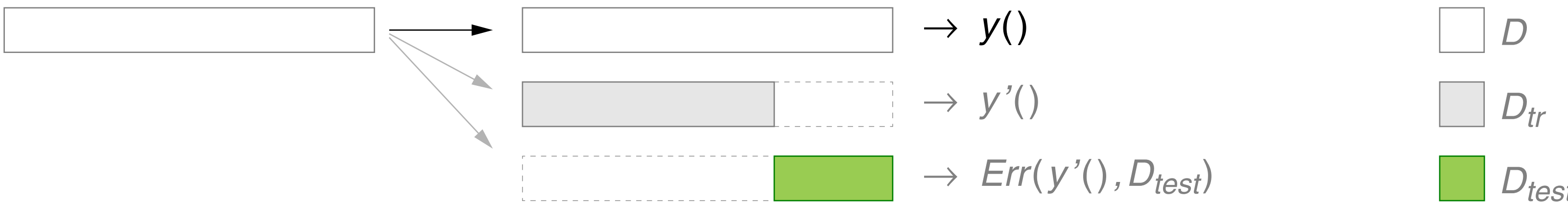
- The training error *Err*$(y(), D_{tr})$ is an optimistic estimation, i.e., it is always lower compared to the (unknown) true error *Err*$^*(y())$. With $D = X$ the training error *Err*$(y(), D_{tr})$ becomes the true error *Err*$^*(y())$.

- Note that the above discussion relate to the meaningfulness of *Err*$(y(), D_{tr})$ as an error estimate—and *not to the classifier* $y()$:

  Obviously, to get the maximum out of the data when training $y()$, D must be exploited completely. A classifier $y()$ trained on D will on average outperform every classifier $y'()$ trained on a subset of D. I.e., on average, *Err*$^*(y()) <$ *Err*$^*(y'())$.

Evaluating Effectiveness

Holdout Error

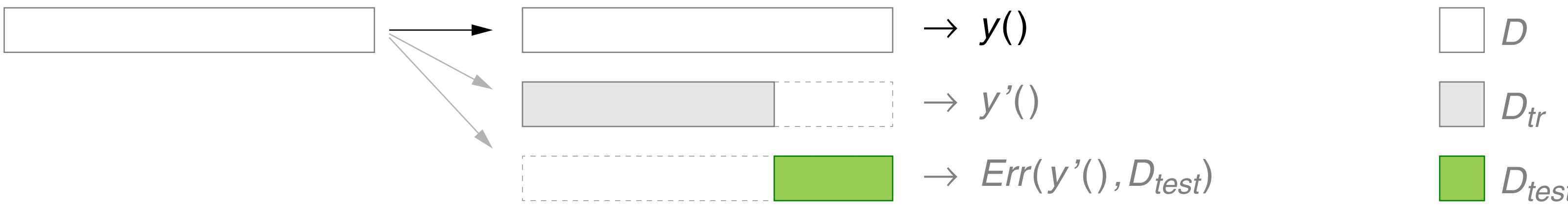


Evaluation setting:

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

Evaluating Effectiveness

Holdout Error



Evaluation setting:

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

Holdout error of $y'()$, $y()$:

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

- $\mathit{Err}(y(), D_{test}) := \mathit{Err}(y'(), D_{test})$

Evaluating Effectiveness

Holdout Error (continued) [LMS algorithm]

Principle: We build $y()$, and, in order to judge $y()$, we train and analyze $y'()$.

1. Training (D, η)

   initialize_random_weights$(\mathbf{w})$, $t = 0$
   REPEAT
   ⋮
   UNTIL(*convergence*$(D, y(), t)$)

2. Training (D_{tr}, η)

   initialize_random_weights$(\mathbf{w})$, $t = 0$
   REPEAT
   ⋮
   UNTIL(*convergence*$(D_{tr}, y'(), t)$)

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

Evaluating Effectiveness

Holdout Error (continued) [LMS algorithm]

Principle: We build $y()$, and, in order to judge $y()$, we train and analyze $y'()$.

1. Training (D, η)

   1. *initialize_random_weights*$(\mathbf{w})$, $t = 0$
   2. **REPEAT**

      ⋮

   10. **UNTIL**$($*convergence*$(D, y(), t)$ $\rightsquigarrow$ $y()$, $Err(y(), D_{tr})$ with $D_{tr} = D$

2. Training (D_{tr}, η)

   1. *initialize_random_weights*$(\mathbf{w})$, $t = 0$
   2. **REPEAT**

      ⋮

   10. **UNTIL**$($*convergence*$(D_{tr}, y'(), t)$ $\rightsquigarrow$ $y'()$, $Err(y'(), D_{tr})$ with $D_{tr} = D \setminus D_{test}$

3. Test $(D_{test}, y'())$ $\rightsquigarrow$ $Err(y(), D_{test}) := Err(y'(), D_{test})$

Remarks (holdout error) :

- We will use the prime symbol »'« to indicate whether a classifier is trained by withholding a test set. E.g., $y'()$ and $y_i'()$ denote classifiers trained by withholding the test sets $D_{\textit{test}}$ and $D_{\textit{test}_i}$ respectively.

- A holdout error of $y()$ cannot be computed if D is entirely used for training $y()$. Instead, *Err*$(y'(), D_{\textit{test}})$, the holdout error for $y'()$ is computed, where $y'()$ has been trained by witholding $D_{\textit{test}}$.

  Err$(y(), D_{\textit{test}})$, the holdout estimation of *Err*$^*(y())$ on $D_{\textit{test}}$, is defined as *Err*$(y'(), D_{\textit{test}})$.

  Recall in this regard that a classifier $y()$ trained on D will on average outperform every classifier $y'()$ trained on a subset of D. I.e., on average, *Err*$^*(y()) <$ *Err*$^*(y'())$.

- The difference between the training error *Err*$(\cdot\,, D_{\textit{tr}})$ and the holdout error *Err*$(\cdot\,, D_{\textit{test}})$ of a classifier quantifies the severity of a possible overfitting.

Remarks (holdout error) : (continued)

- When splitting D into $D_{\textit{tr}}$ and $D_{\textit{test}}$ 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*$(y(), D_{\textit{test}})$ 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 $D_{\textit{tr}} \subset D$ is called a (class-)stratified sample of D if it has the same class frequency distribution as D, i.e.:

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

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

Evaluating Effectiveness

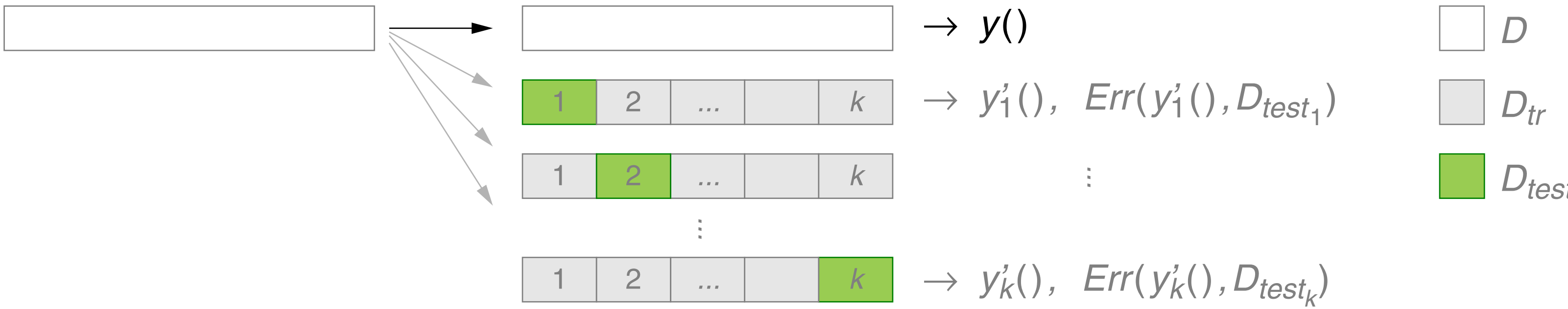
k-Fold Cross-Validation

Evaluation setting:

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

Evaluating Effectiveness

k-Fold Cross-Validation



Evaluation setting:

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

Cross-validation error of $y()$:

- $$Err(y(), D, k) := \frac{1}{k} \sum_{i=1}^{k} \frac{|\{(\mathbf{x}, c) \in D_{test_i} : y'_i(\mathbf{x}) \neq c\}|}{|D_{test_i}|}$$

$=$ averaged misclassification rate of the $y'_i()$ on the k test sets.

Remarks:

- n-fold cross-validation (aka “leave one out”) is the special case with $k = n$. Obviously singleton test sets ($|D_{\textit{test}_i}| = 1$) 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 $\textit{Err}^*(y())$ (see section Ensemble Methods > Bootstrap Aggregating in part Ensemble and Meta). [Wikipedia]

Evaluating Effectiveness

Comparing Model Variants [Estimating Error Bounds]

Experiment setting:

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

Remarks:

- In general, a hyperparameter π (with values $\pi_1, \pi_2, \ldots, \pi_m$) controls the learning process for a model's parameters, but is itself *not learned*. Instead, various hyperparameter settings are tried out.

  On the other hand, a regime in which knowledge (such as hyperparameter settings) *about* a machine learning process is learned is called *meta learning*.

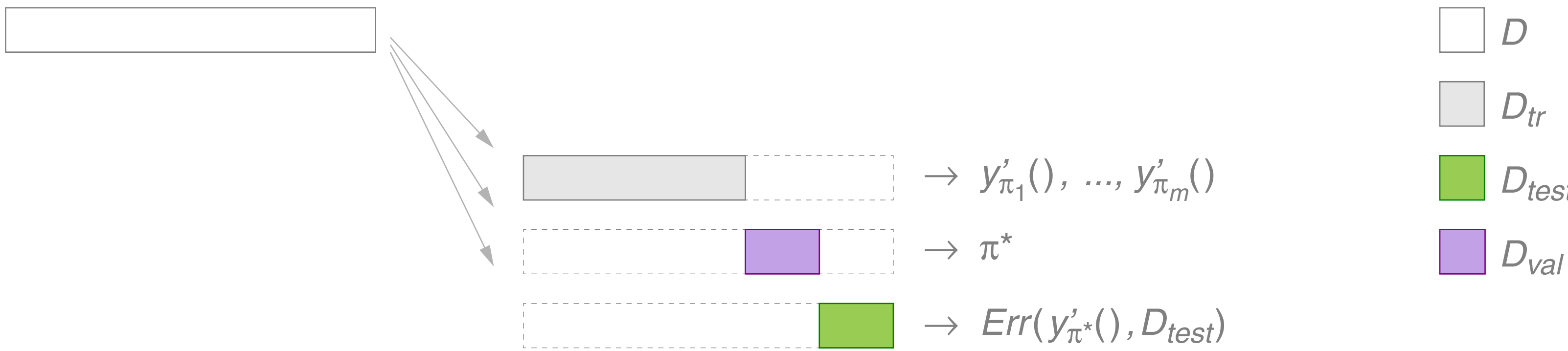
- Examples for hyperparameters in different kinds of model functions:
  - learning rate η in regression-based models fit via gradient descent
  - type of regularization loss used, e.g., $R_{||\vec{\mathbf{w}}||_2^2}$ or $R_{||\vec{\mathbf{w}}||_1}$
  - the term λ controlling the weighting of goodness-of-fit loss and regularization 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 [Holdout Error]

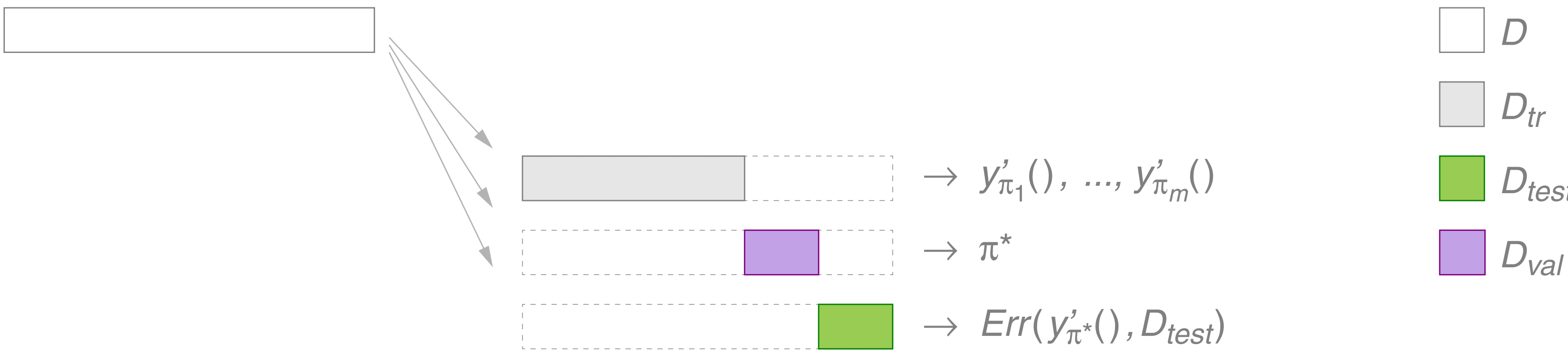


Evaluation setting:

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

Evaluating Effectiveness

Model Selection: Single Validation Set [Holdout Error]



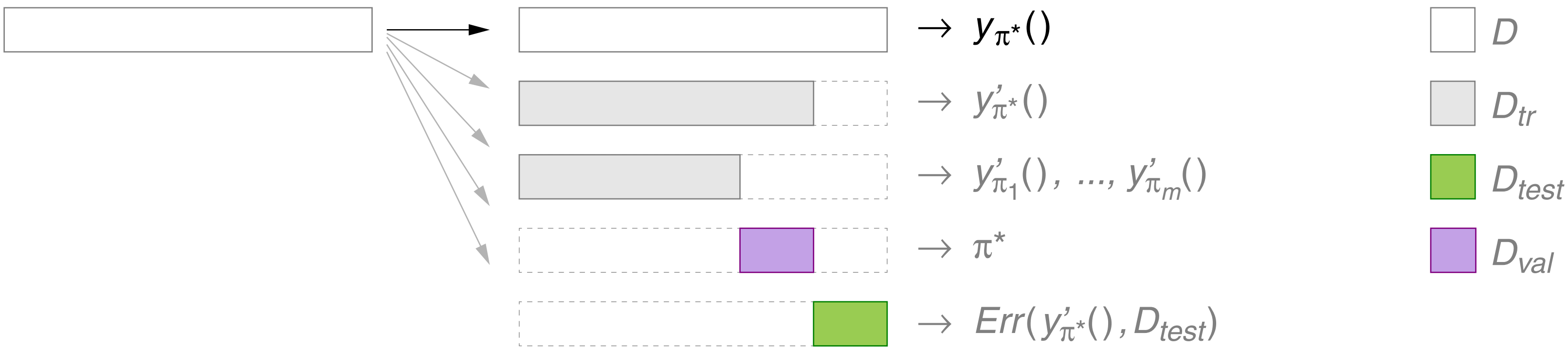
Evaluation setting:

- ❑ $D_{test} \subset D$ is a test set.
- ❑ $D_{val} \subset (D \setminus D_{test})$ is a validation set.
- ❑ $y'_{\pi_l}(),\ l = 1, \ldots, m,$ are the classifiers trained on $D_{tr} = D \setminus (D_{test} \cup D_{val})$.
- ❑ $\pi^* = \underset{\pi_l,\ l=1,\ldots,m}{\text{argmin}} \dfrac{|\{(\mathbf{x}, c) \in D_{val} : y'_{\pi_l}(\mathbf{x}) \neq c\}|}{|D_{val}|}$

↪p. 180

Evaluating Effectiveness

Model Selection: Single Validation Set (continued) [Holdout Error]



Evaluation setting:

⋮

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

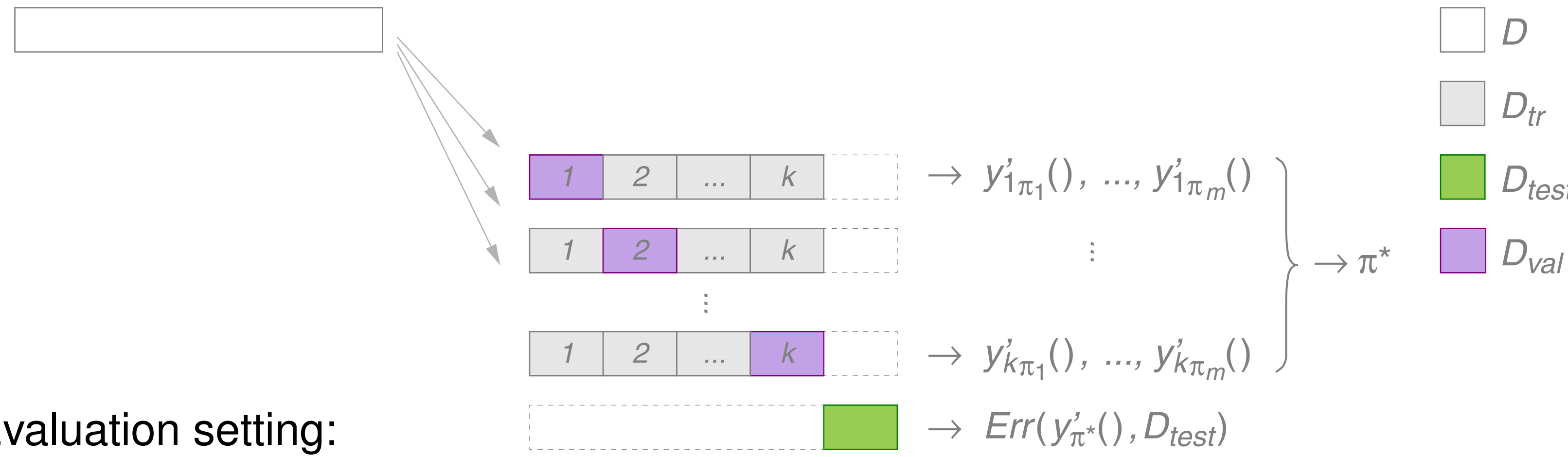
Holdout error of $y_{\pi^*}()$:

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

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

Evaluating Effectiveness

Model Selection: k validation sets [k-Fold Cross-Validation]

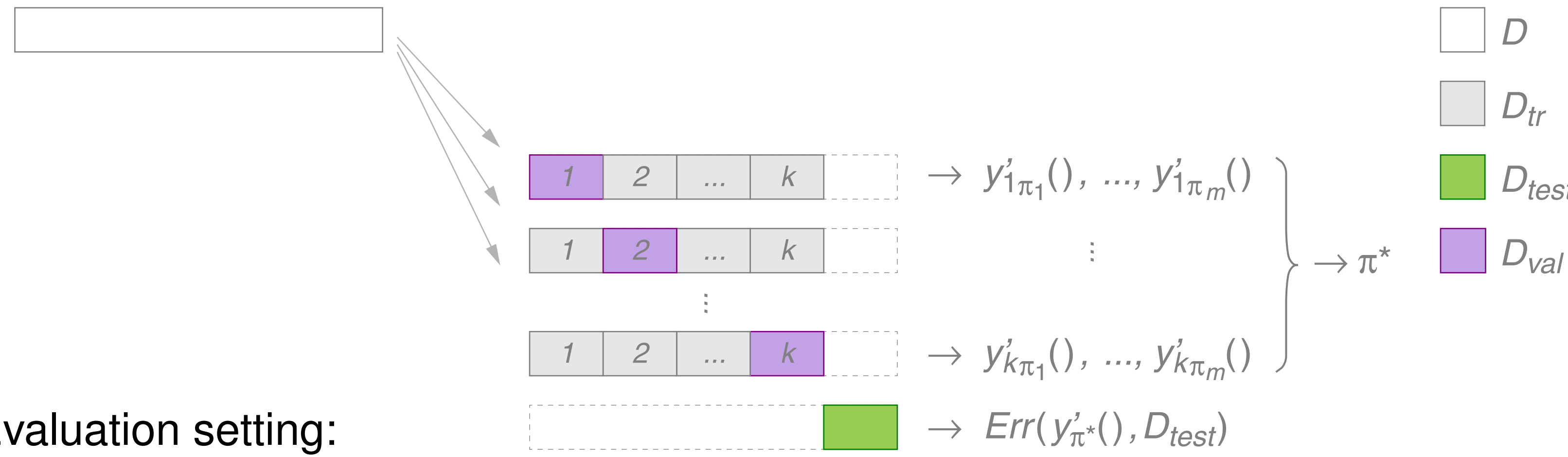


Evaluation setting:

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

Evaluating Effectiveness

Model Selection: k validation sets [k-Fold Cross-Validation]



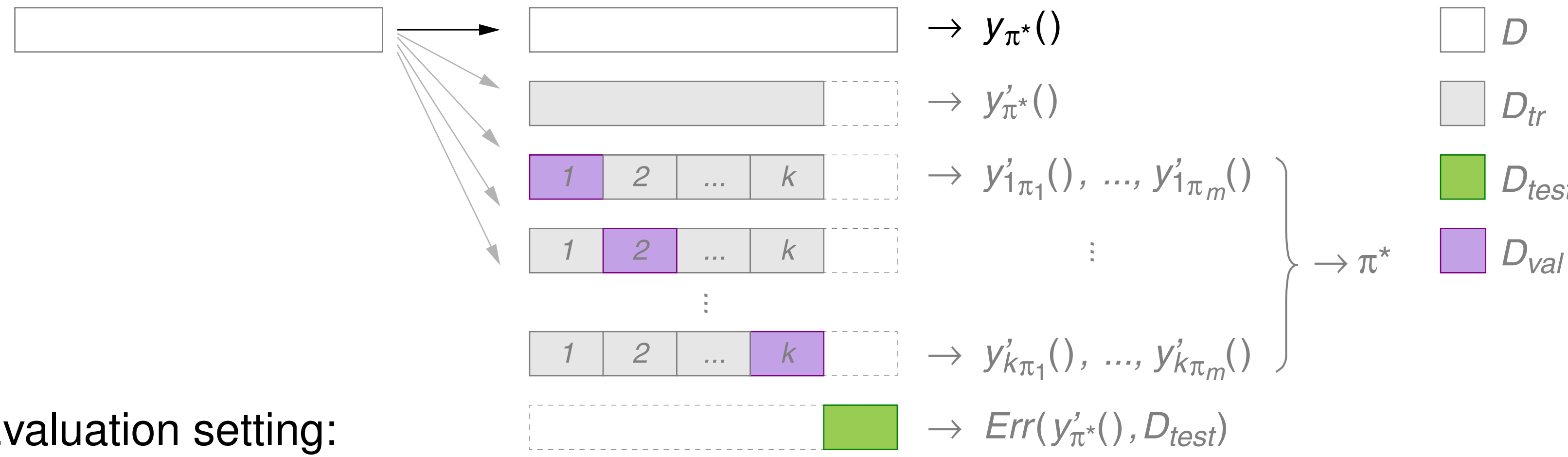
Evaluation setting:

- $D_{test} \subset D$
- k validation sets D_{val_i} by splitting $D \setminus D_{test}$ into k disjoint sets of similar size.
- $y'_{i_{\pi_l}}(),\ \ i = 1, \ldots, k,\ \ l = 1, \ldots, m,$ are the $k \cdot m$ classifiers trained on $D_{tr_i} = D \setminus (D_{test} \cup D_{val_i})$.
- $\pi^* = \underset{\pi_l,\ l=1,\ldots,m}{\text{argmin}} \sum_{i=1}^{k} \frac{|\{(\mathbf{x}, c) \in D_{val_i} : y'_{i_{\pi_l}}(\mathbf{x}) \neq c\}|}{|D_{val_i}|}$

↪p. 183

Evaluating Effectiveness

Model Selection: k validation sets (continued) [k-Fold Cross-Validation]



Evaluation setting:

⋮

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

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

Remarks:

- The validation set is also called “development set” or “dev set”. [Wikipedia]

Evaluating Effectiveness

Misclassification Costs

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

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

Holdout error of $y()$ based on misclassification costs:

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

  $=$ weighted misclassification rate of $y'()$ on the test set.

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 $I_{\neq}(y(), c)$:

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

