Chapter ML:II (continued)

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 $c_{\mathbf{x}} = \gamma(o)$, $o \in O$. Then, the true misclassification rate of $y()$, denoted *Err*$^*(y())$, is defined as follows:

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

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 $\alpha : O \to \mathbf{X}$, 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* $\gamma()$ is unknown and hence $\mathit{Err}^*(y())$ is unknown.

Solution:

- 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.

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 *Err*$^*(y())$ as the ratio of misclassified feature vectors in X, we can define *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 *Err*$^*(y())$ as the ratio of misclassified feature vectors in X, we can define *Err*$^*(y())$ as the *probability* that $y()$ misclassifies some 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 $y : \mathbf{X} \to C$ 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, **X** $: \Omega \to \mathbf{X}$, and *C* $: \Omega \to C$.

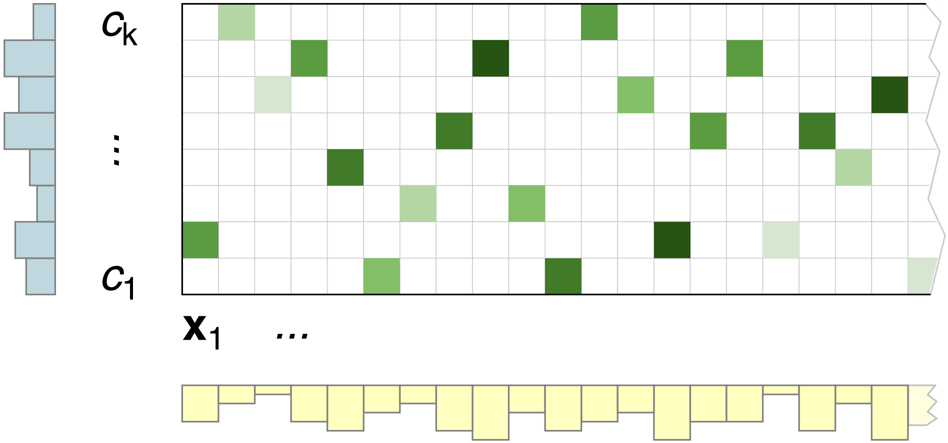
Then $p(\mathbf{x}, c)$, $p(\mathbf{x}, c) := P(\textbf{X}{=}\mathbf{x}, \mathit{C}{=}c)$, is the probability of the joint event (1) to get the vector $\mathbf{x} \in \mathbf{X}$, and, (2) that the respective object belongs to class $c \in C$. With $p(\mathbf{x}, c)$ the true misclassification rate of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

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

Evaluating Effectiveness

Illustration 1: Label Noise

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



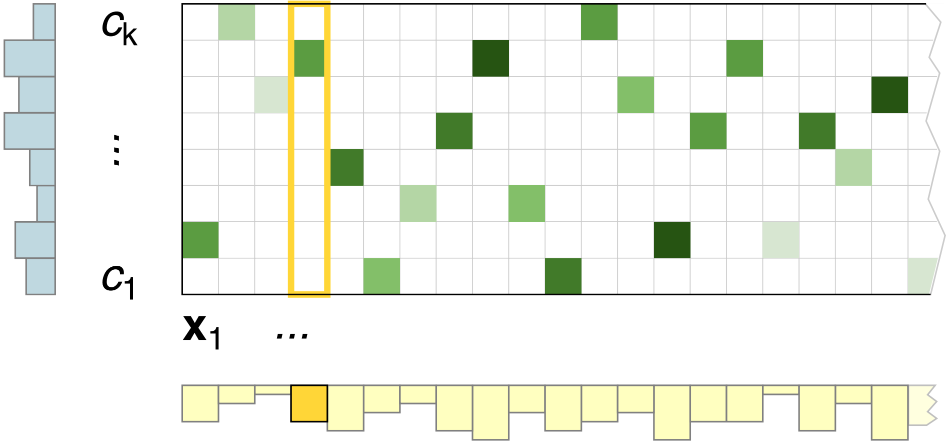
(no label noise → classes are unique)

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

Evaluating Effectiveness

Illustration 1: Label Noise (continued)

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



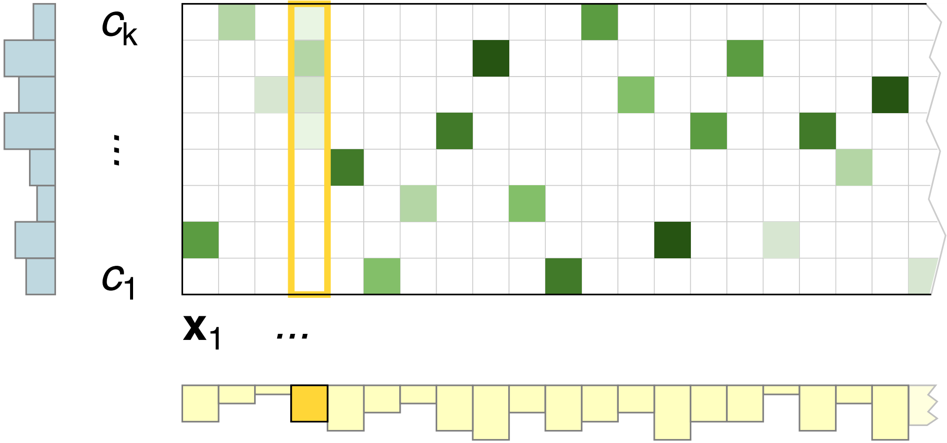
(no label noise → classes are unique)

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

Evaluating Effectiveness

Illustration 1: Label Noise (continued)

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



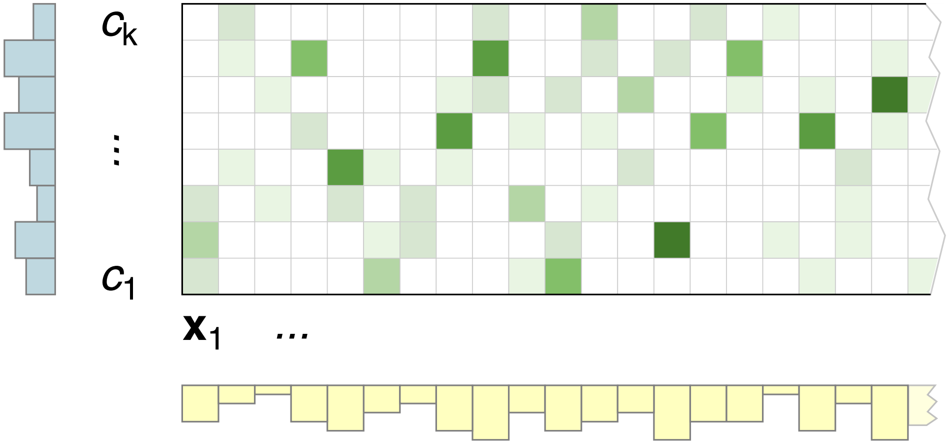
(label noise → classes are *not* unique)

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

Evaluating Effectiveness

Illustration 1: Label Noise (continued)

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



(label noise → classes are *not* unique)

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

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: (continued)

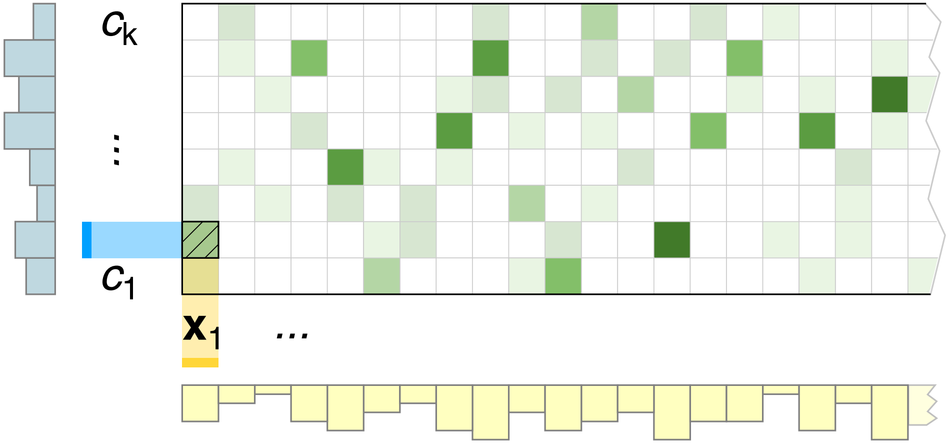
- $P()$ is a probability measure (see section Probability Basics in part Bayesian Learning) and its argument is an event. Examples for events are "**X**$=\mathbf{x}$", "**X**$=\mathbf{x}$, $C$$=c$", or "**X**$=\mathbf{x} \mid$ $C$$=c$".
- $p(\mathbf{x}, c)$, $p(\mathbf{x})$, or $p(\mathbf{x} \mid c)$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

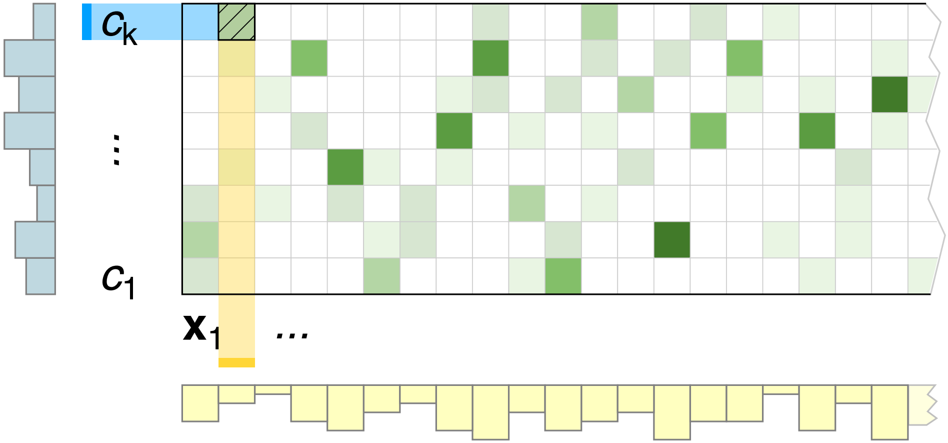
The Bayes classifier returns for $\mathbf{x}$ the class with the highest [posterior] probability:



Evaluating Effectiveness

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

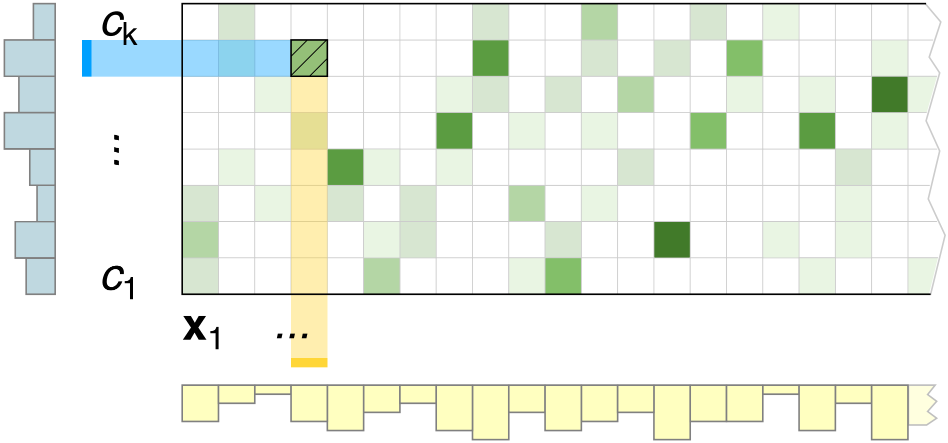
The Bayes classifier returns for $\mathbf{x}$ the class with the highest [posterior] probability:



Evaluating Effectiveness

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

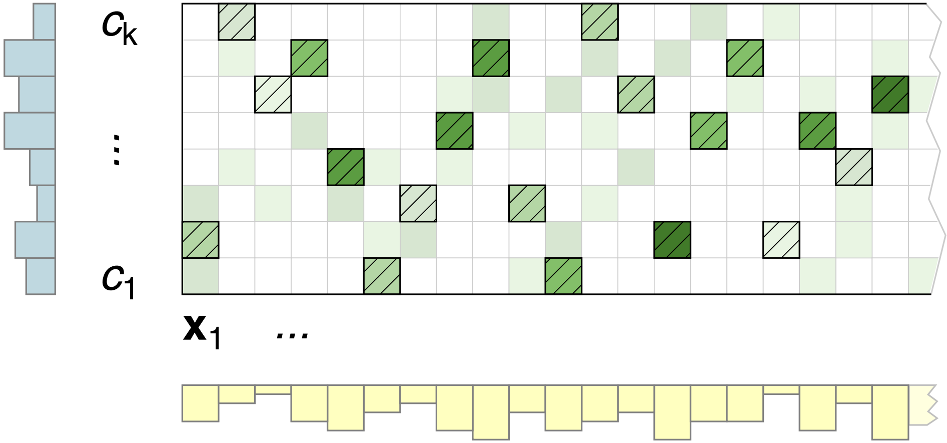
The Bayes classifier returns for $\mathbf{x}$ the class with the highest [posterior] probability:



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:

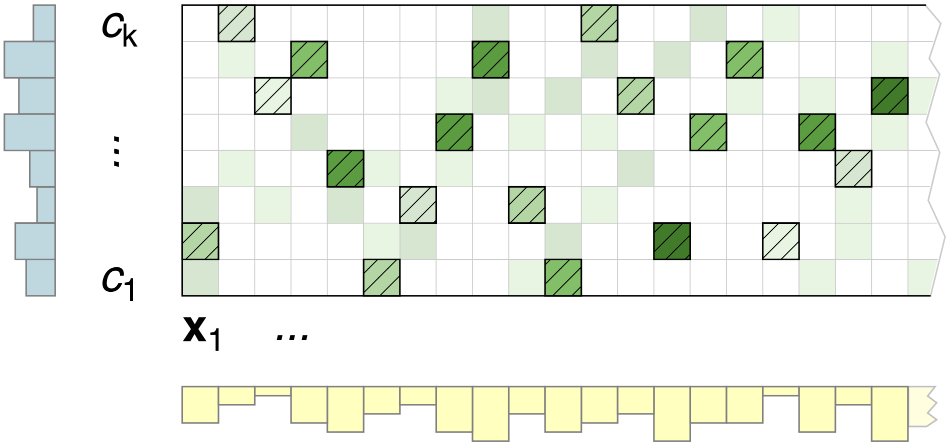


Bayes classifier: $y^*(\mathbf{x}) = \underset{c \in C}{\text{argmax}}\, p(c, \mathbf{x}) = \underset{c \in C}{\text{argmax}}\, p(c \mid \mathbf{x})$

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

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

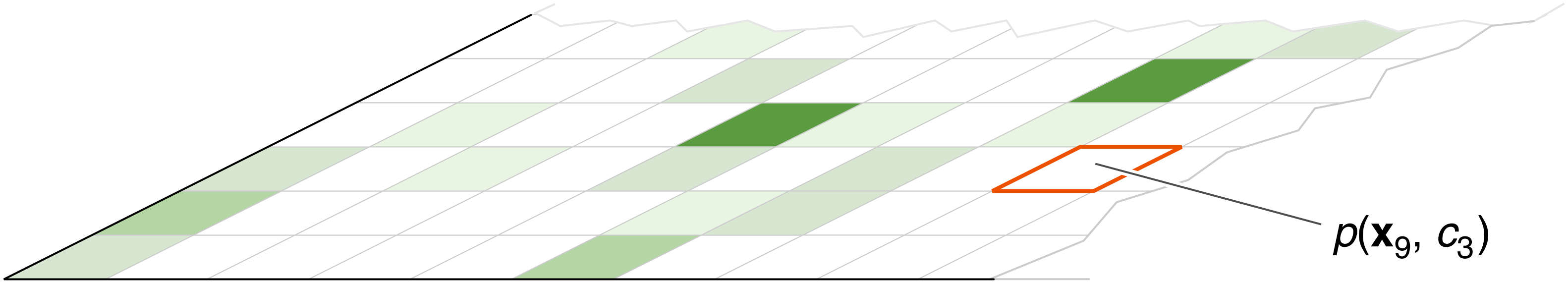
- Prerequisite to construct the Bayes classifier and to compute its error is knowledge about the joint probabilities, $p(c, \mathbf{x})$ or $p(c \mid \mathbf{x})$. In this regard the size of the available data, D, decides about the possibility and the quality for the estimation of the probabilities.

- 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

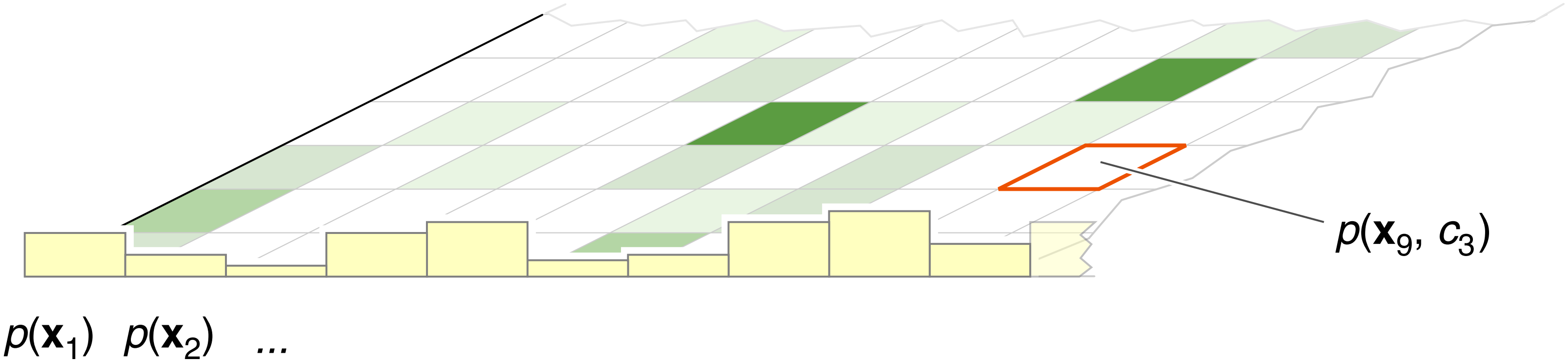
Joint probabilities $p(\mathbf{x}, c) := P(\mathbf{X}{=}\mathbf{x}, C{=}c)$:



Evaluating Effectiveness

Illustration 3: Marginal and Conditional Distributions (continued)

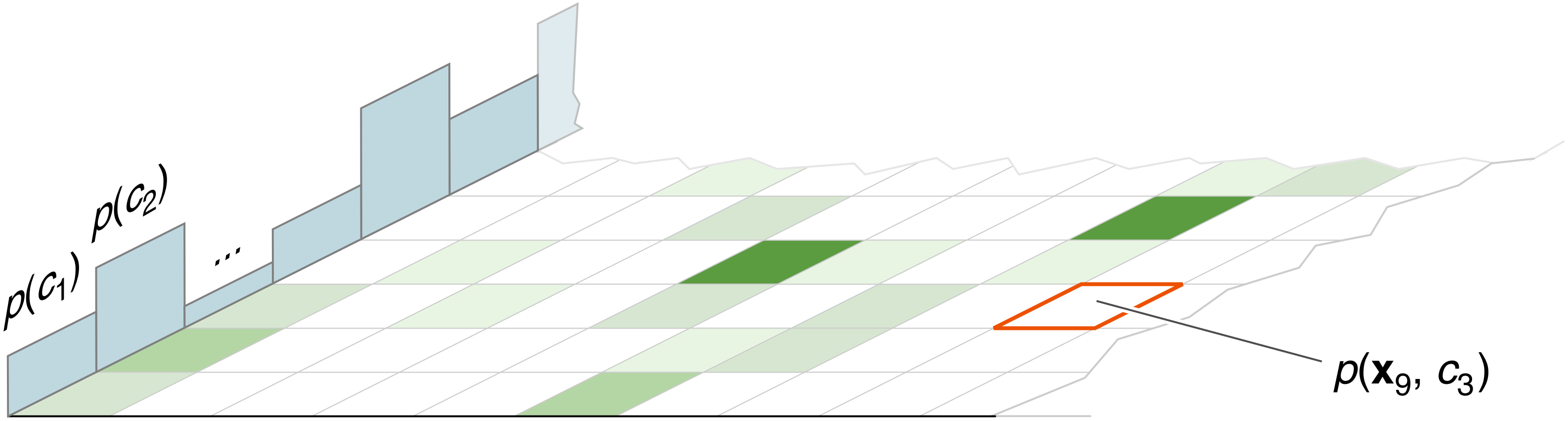
Marginal probabilities $p(\mathbf{x}) := P(\mathbf{X}{=}\mathbf{x})$:



Evaluating Effectiveness

Illustration 3: Marginal and Conditional Distributions (continued)

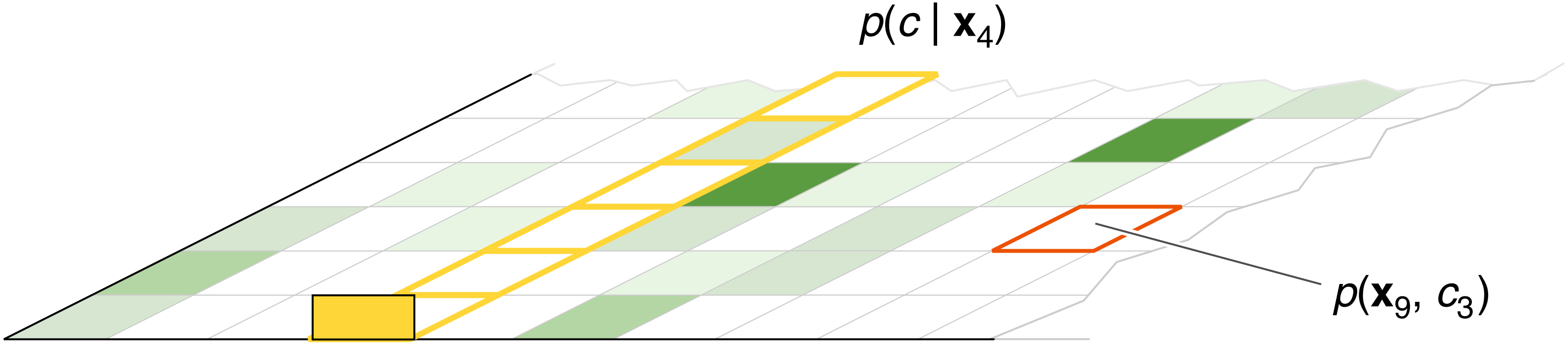
Marginal probabilities $p(c) := P(\mathbf{C}{=}c)$:



Evaluating Effectiveness

Illustration 3: Marginal and Conditional Distributions (continued)

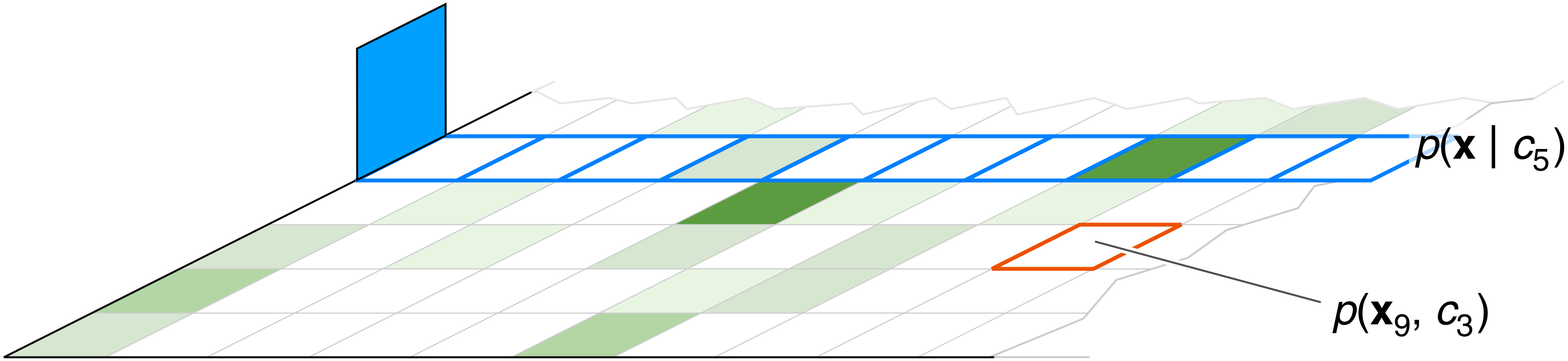
Probabilities of the classes c under feature vector (the condition) $\mathbf{x}_4$, denoted by $p(c \mid \mathbf{x}_4) := P(\mathbf{C}{=}c \mid \mathbf{X}{=}\mathbf{x}_4) \;\equiv\; P_{\mathbf{X}=\mathbf{x}_4}(\mathbf{C}{=}c)$:



Evaluating Effectiveness

Illustration 3: Marginal and Conditional Distributions (continued)

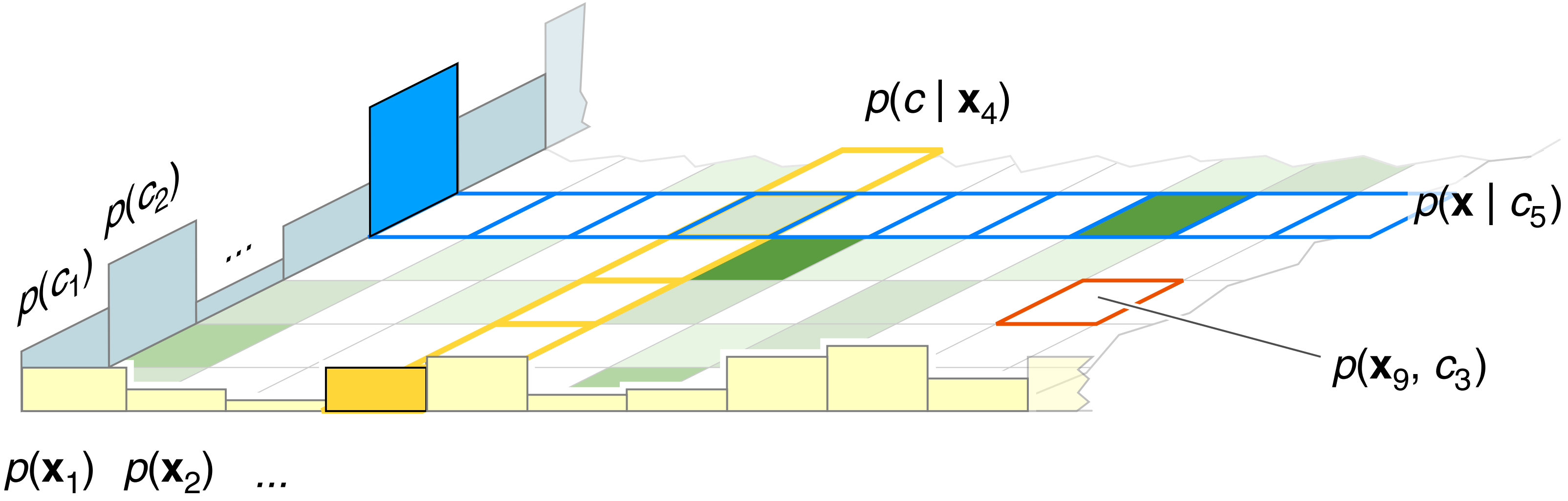
Probabilities of the feature vectors $\mathbf{x}$ under class (the condition) c_5, denoted by $p(\mathbf{x} \mid c_5) := P(\mathbf{X}{=}\mathbf{x} \mid C{=}c_5) \;\equiv\; P_{C=c_5}(\mathbf{X}{=}\mathbf{x})$:



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., $p(\mathbf{x}, c_5)/p(c_5)$.

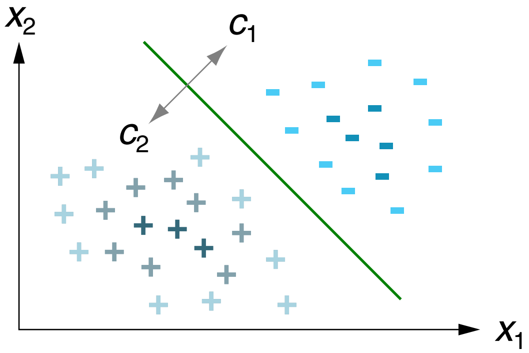
- $p(\mathbf{x}, c) = p(c, \mathbf{x}) = p(c \mid \mathbf{x}) \cdot p(\mathbf{x})$, where $p(\mathbf{x})$ is the prior probability for event $\mathbf{X}{=}\mathbf{x}$, and $p(c \mid \mathbf{x})$ is the probability for event $C{=}c$ given event $\mathbf{X}{=}\mathbf{x}$. Likewise, $p(\mathbf{x}, c) =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

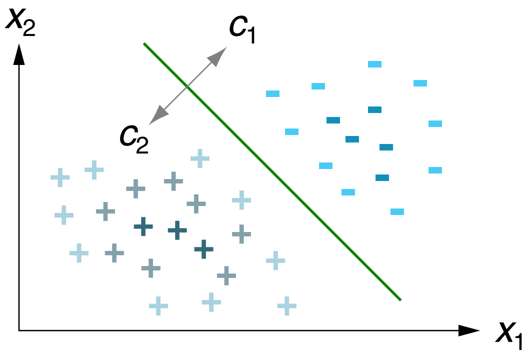


$$X = \left\{ \begin{pmatrix} x_{1_1} \\ x_{1_2} \end{pmatrix}, \begin{pmatrix} x_{2_1} \\ x_{2_2} \end{pmatrix}, \ \ldots \right\}, \quad \mathbf{X} = \mathbf{R}^2$$

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

Evaluating Effectiveness

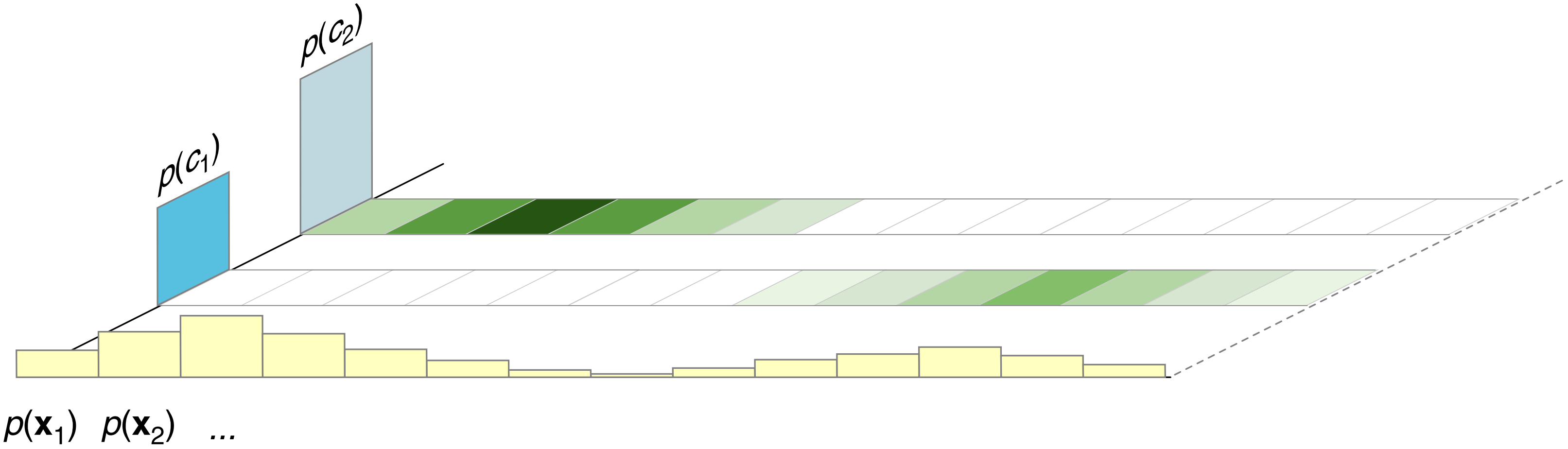
Illustration 4: Probability Distribution in a Regression Setting (continued)



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

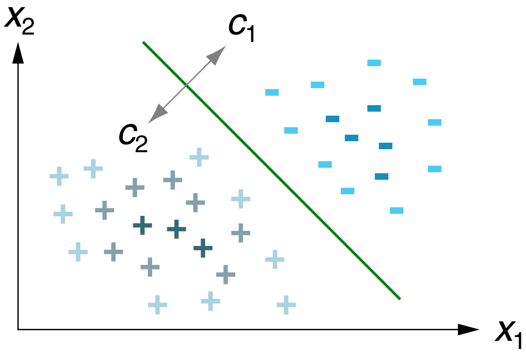
$$\mathbf{x}_1 \qquad \mathbf{x}_2 \qquad \ldots$$

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



Evaluating Effectiveness

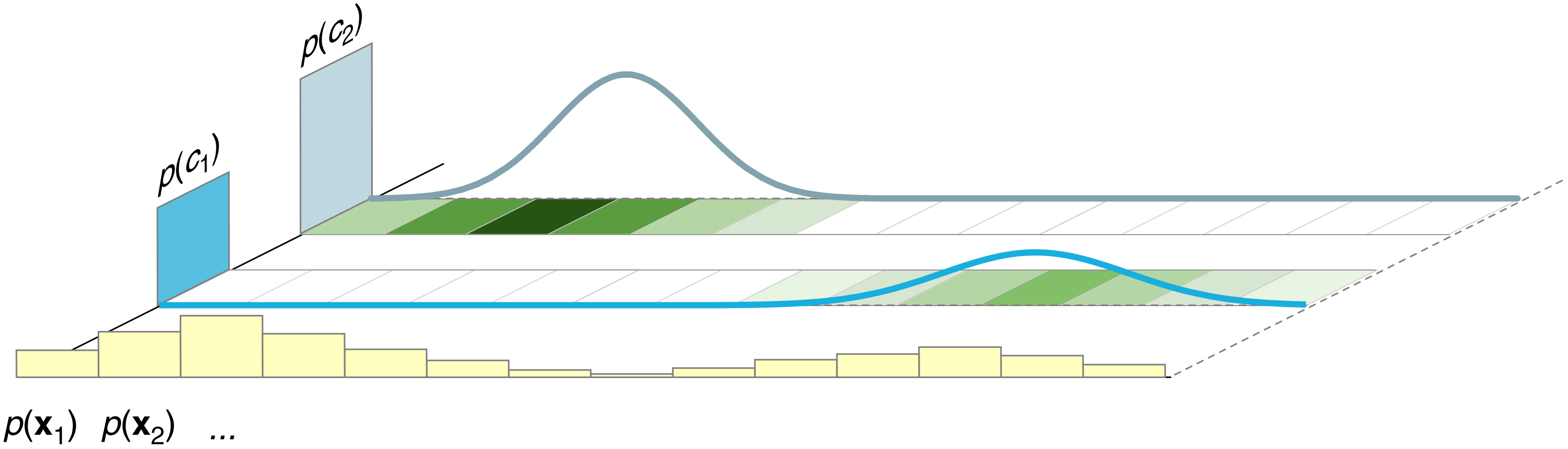
Illustration 4: Probability Distribution in a Regression Setting (continued)



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

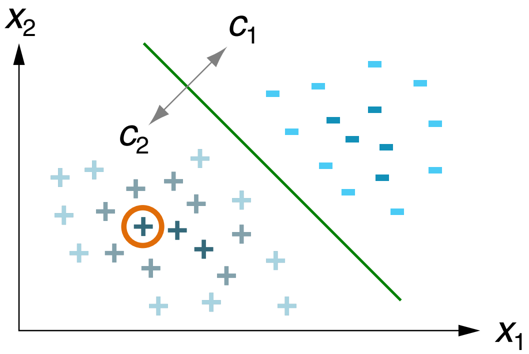
$$\mathbf{x}_1 \qquad \mathbf{x}_2 \qquad \ldots$$

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



Evaluating Effectiveness

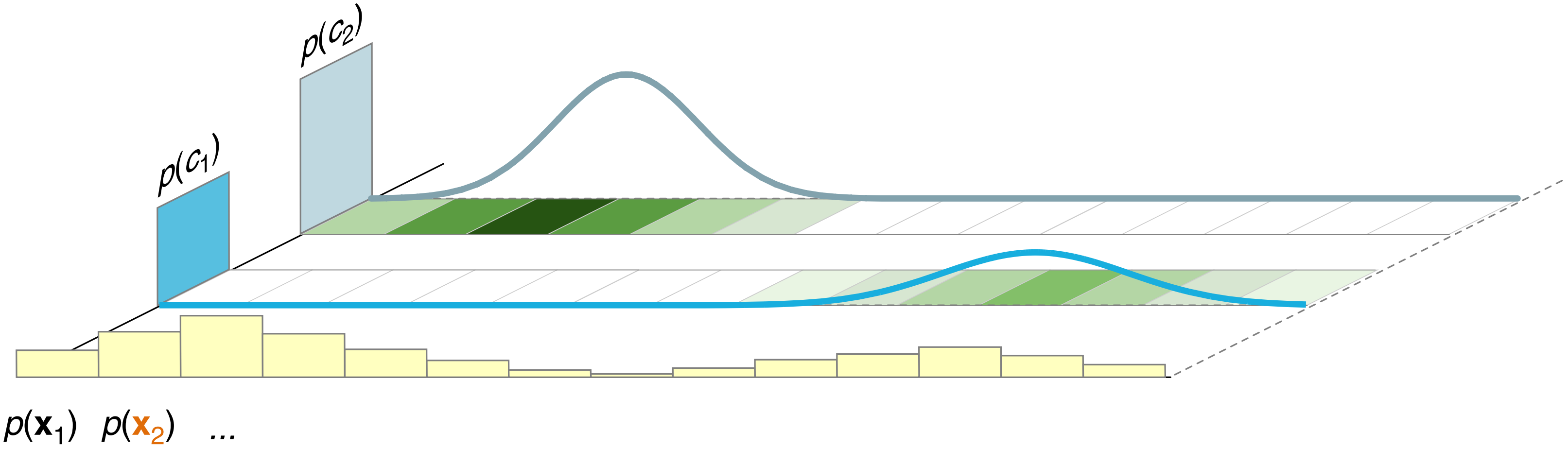
Illustration 4: Probability Distribution in a Regression Setting (continued)



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

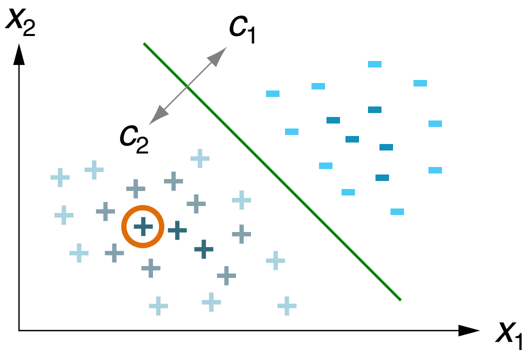
$\mathbf{x}_1$ $\mathbf{x}_2$ $\ldots$

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



Evaluating Effectiveness

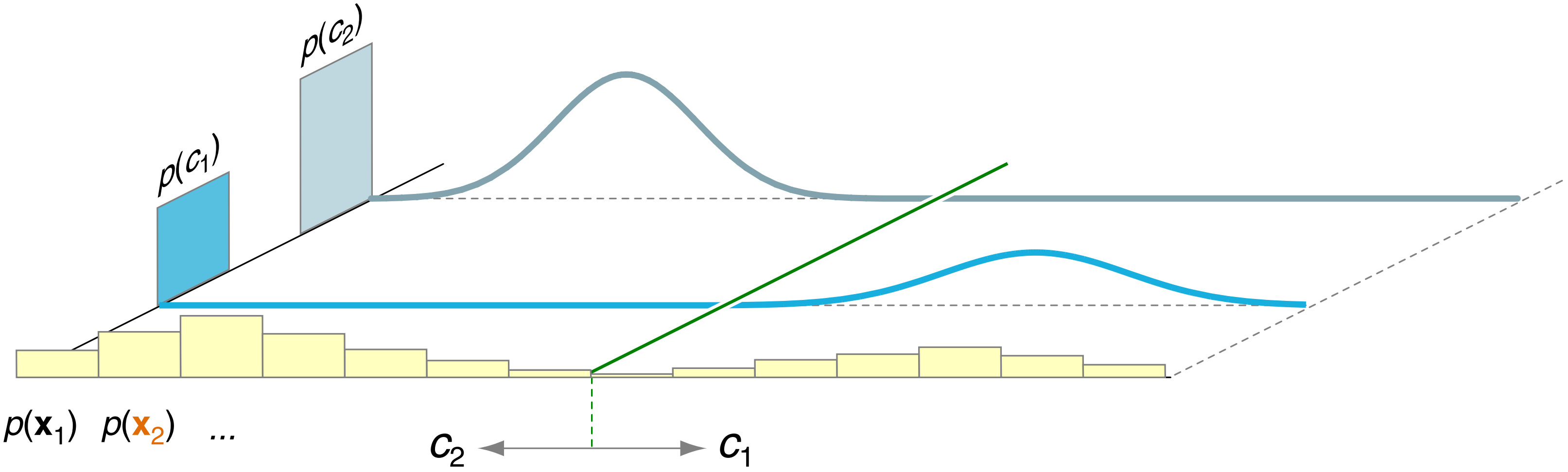
Illustration 4: Probability Distribution in a Regression Setting (continued)



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

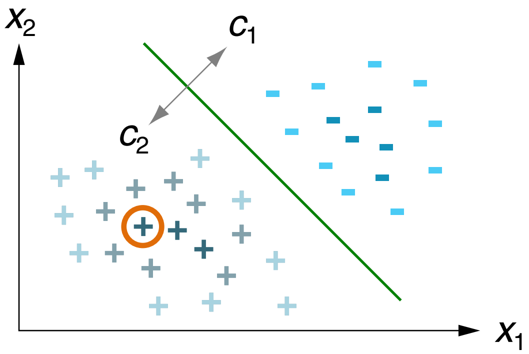
$\mathbf{x}_1 \quad \mathbf{x}_2 \quad \ldots$

Optimum hyperplane classifier:



Evaluating Effectiveness

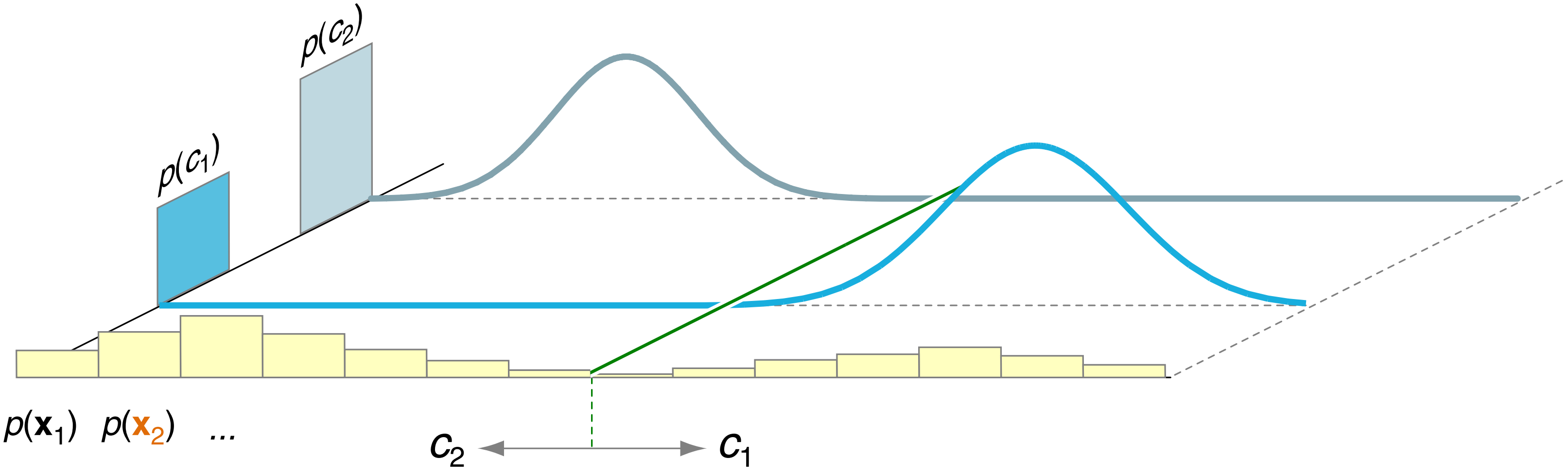
Illustration 4: Probability Distribution in a Regression Setting (continued)



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

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

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

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”).
- What can be said about the true error *Err*$^*(y())$?

Evaluating Effectiveness

Estimating Error Bounds (continued)

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”).
- What can be said about the true error $\mathit{Err}^*(y())$?

The following relations typically hold:

Underestimation (likely)	Overestimation (unlikely)

Training error   Cross-validation error   Holdout error   True error

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

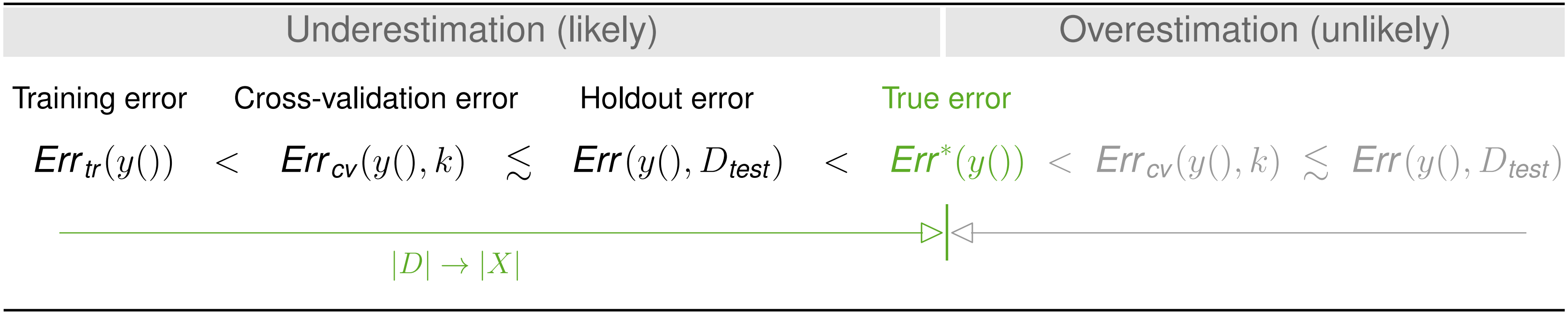
Evaluating Effectiveness

Estimating Error Bounds (continued)

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”).
- 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

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

$|D| \rightarrow |X|$

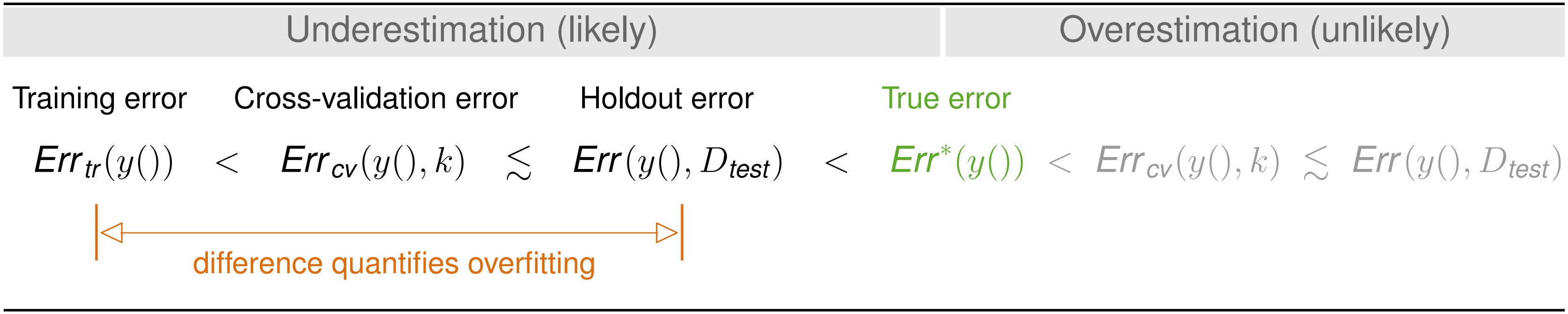
Evaluating Effectiveness

Estimating Error Bounds (continued)

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”).
- What can be said about the true error $\mathit{Err}^*(y())$?

The following relations typically hold:



Underestimation (likely) | Overestimation (unlikely)

Training error | Cross-validation error | Holdout error | True error

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

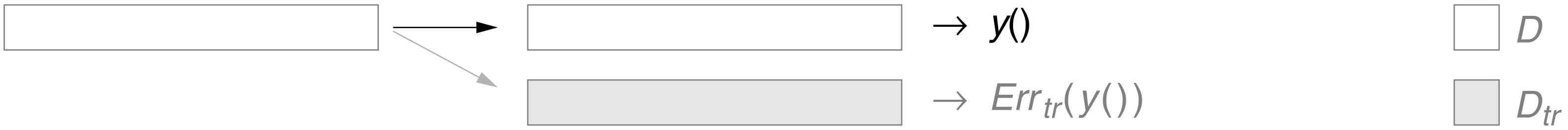
difference quantifies overfitting

Remarks:

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

Evaluating Effectiveness

Training Error



Evaluation setting:

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

Training error of $y()$:

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

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

Remarks:

- The estimation of $\mathit{Err}_{tr}(y())$ is based on $y()$ and tests against $D_{tr} = D$. I.e., the same examples that are used for training $y()$ are also used to test $y()$. Hence $\mathit{Err}_{tr}(y())$ 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 $\sim \mathbf{x}$, value $\sim c$), then $\mathit{Err}_{tr}(y())$ would be zero, which would tell us nothing about the failure of $y()$ in the wild.

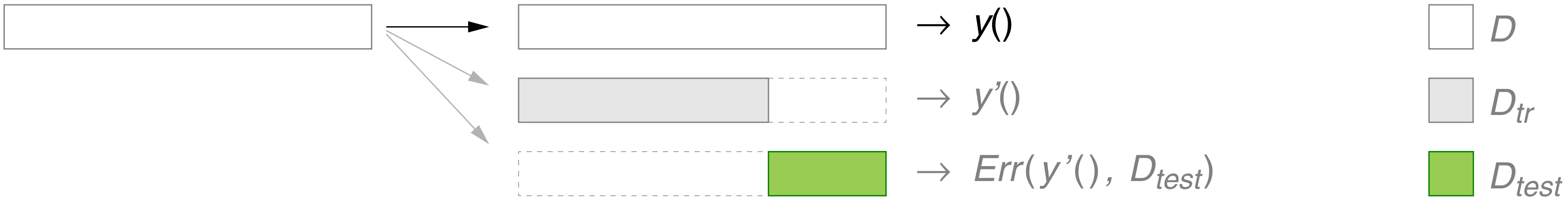
- $\mathit{Err}_{tr}(y())$ is an optimistic estimation, i.e., it is constantly lower compared to the (unknown) true error $\mathit{Err}^*(y())$. With $D = X$ the training error $\mathit{Err}_{tr}(y())$ becomes the true error $\mathit{Err}^*(y())$.

- Note that the above issues relate to the meaningfulness of $\mathit{Err}_{tr}(y())$ as an error estimate—and *not to the classifier* $y()$.

  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.

Evaluating Effectiveness

Holdout Error



Evaluation setting:

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

Holdout error of $y()$:

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

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

Evaluating Effectiveness

Holdout Error (continued)

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)

1. Training (D, η)

```
 1.  $\mathit{initialize\_random\_weights}(\mathbf{w})$,  $t = 0$
 2.  REPEAT                                              $\rightsquigarrow y()$
 ⋮
10.  UNTIL($\mathit{convergence}(D, y(), t)$)            $\rightsquigarrow \mathit{Err}_{tr}(y())$   (not used)
```

2. Training (D_{tr}, η)

```
 1.  $\mathit{initialize\_random\_weights}(\mathbf{w})$,  $t = 0$
 2.  REPEAT                                              $\rightsquigarrow y'()$
 ⋮
10.  UNTIL($\mathit{convergence}(D_{tr}, y'(), t)$)      $\rightsquigarrow \mathit{Err}_{tr}(y'())$
```

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

Evaluating Effectiveness

Holdout Error (continued)

1. Training (D, η)

```
    1.  $initialize\_random\_weights(\mathbf{w})$,   $t = 0$
    2.  REPEAT                                                  $\rightsquigarrow y()$
    ⋮
    10. UNTIL($convergence(D, y(), t)$)                         $\rightsquigarrow Err_{tr}(y())$   (not used)
```

2. Training (D_{tr}, η)

```
    1.  $initialize\_random\_weights(\mathbf{w})$,   $t = 0$
    2.  REPEAT                                                  $\rightsquigarrow y'()$
    ⋮
    10. UNTIL($convergence(D_{tr}, y'(), t)$)                   $\rightsquigarrow Err_{tr}(y'())$
```

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

Remarks:

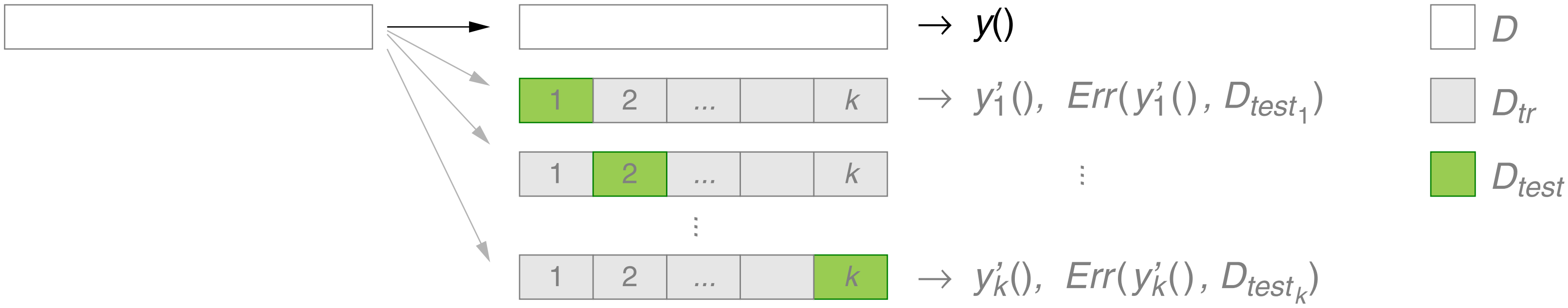
- The difference between the training error, *Err*$_{tr}(y())$ or *Err*$_{tr}(y'())$, and the holdout error, *Err*$(y(), D_{test})$, quantifies the severity of a possible overfitting.
- When splitting D into D_{tr} and D_{test} one has to ensure that the underlying distribution is maintained, i.e., the 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 »’« 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_{test} and D_{test_i} respectively.

Evaluating Effectiveness

k-Fold Cross-Validation

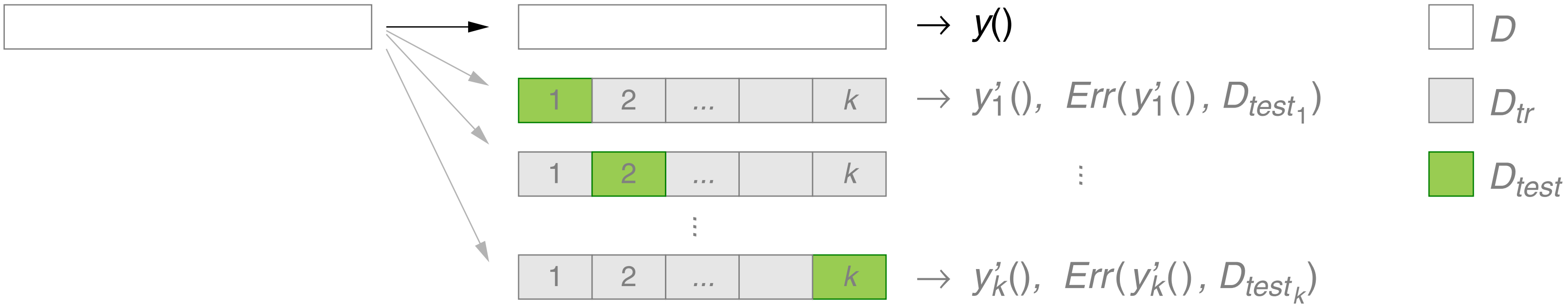


Evaluation setting:

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

Cross-validation error of $y()$:

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

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

Remarks:

- 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:

- $D = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_n, c_n)\} \subseteq X \times C$ 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.
- 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.

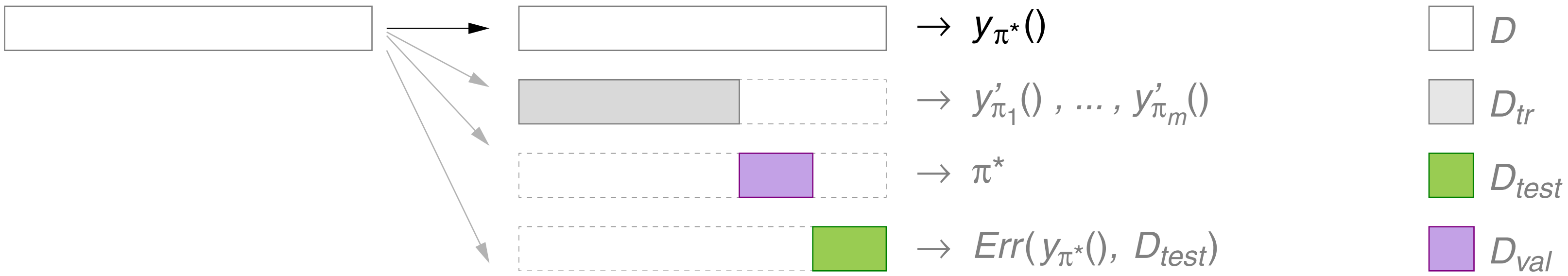
  A regime where knowledge (such as hyperparameter settings) *about* a machine learning process is learned is called meta learning.

- 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., $R_{||\vec{\mathbf{w}}||_2^2}$ or $R_{||\vec{\mathbf{w}}||_1}$
  - 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



→ $y_{\pi^*}()$

→ $y'_{\pi_1}()$, ... , $y'_{\pi_m}()$

→ π^*

→ $Err(y_{\pi^*}(), D_{test})$

□ D

▨ D_{tr}

🟩 D_{test}

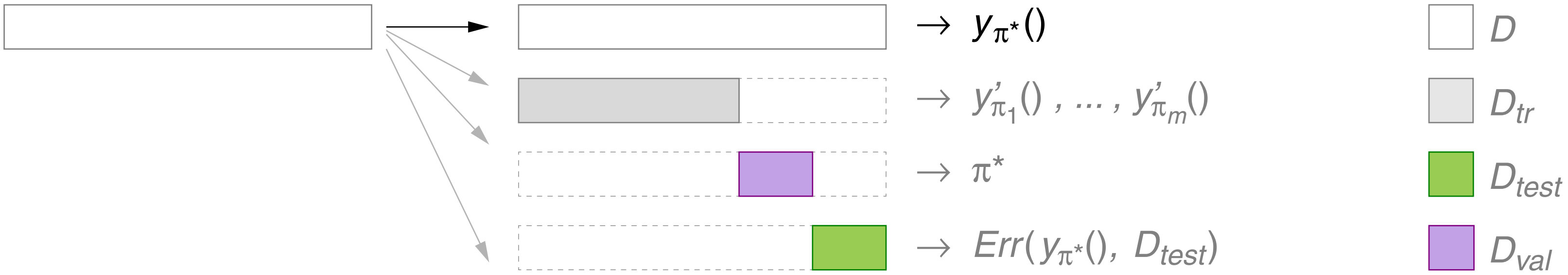
🟪 D_{val}

Evaluation setting:

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



→ $y_{\pi^*}()$
→ $y'_{\pi_1}()$, ... , $y'_{\pi_m}()$
→ π^*
→ $Err(y_{\pi^*}(), D_{test})$
D
D_{tr}
D_{test}
D_{val}

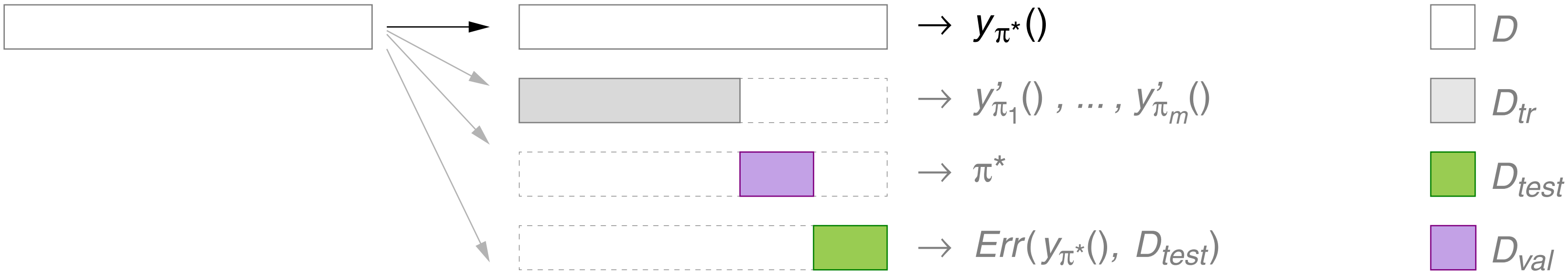
Evaluation setting:

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

⋮

Evaluating Effectiveness

Model Selection: Single Validation Set (continued)



→ $y_{\pi^*}()$
→ $y'_{\pi_1}()$, ... , $y'_{\pi_m}()$
→ π^*
→ $Err(y_{\pi^*}(), D_{test})$
D
D_{tr}
D_{test}
D_{val}

Evaluation setting:

⋮

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

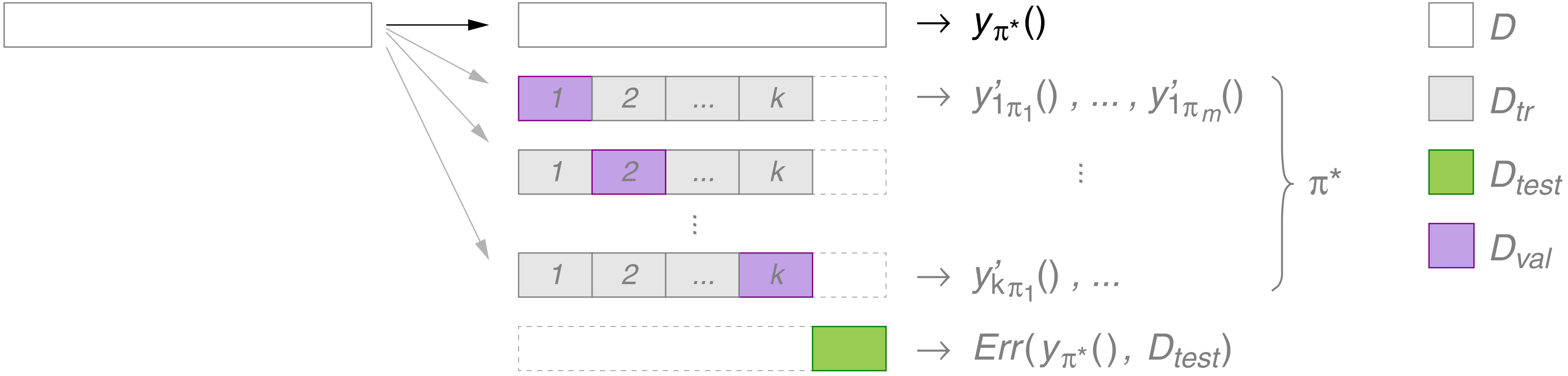
Holdout error of $y_{\pi^*}()$:

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

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

Evaluating Effectiveness

Model Selection: k validation sets



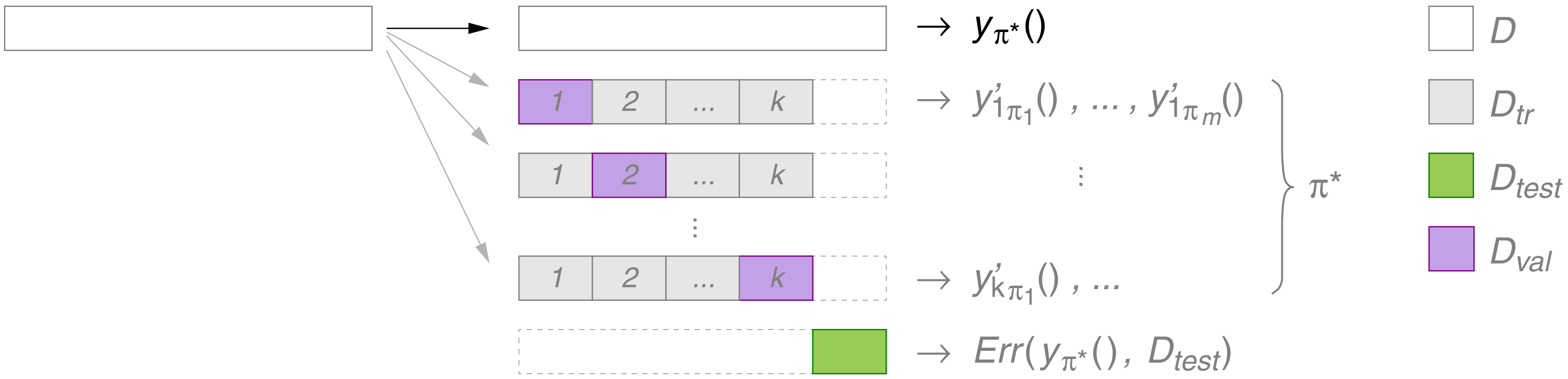
1
2
...
k
$\rightarrow y_{\pi^*}()$
$\rightarrow y'_{1\pi_1}()$, ... , $y'_{1\pi_m}()$
$\vdots$
$\rightarrow y'_{k\pi_1}()$, ...
π^*
$\rightarrow Err(y_{\pi^*}(), D_{test})$
D
D_{tr}
D_{test}
D_{val}

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.
- $y'_{i_{\pi_l}}()$, $i = 1, \ldots, k$, $l = 1, \ldots, m$, 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)



$\rightarrow y_{\pi^*}()$
1
2
...
k
$\rightarrow y'_{1\pi_1}()\,,\,\ldots\,,\, y'_{1\pi_m}()$
$\vdots$
$\vdots$
$\rightarrow y'_{k\pi_1}()\,,\,\ldots$
π^*
$\rightarrow Err(y_{\pi^*}(),\, D_{test})$
D
D_{tr}
D_{test}
D_{val}

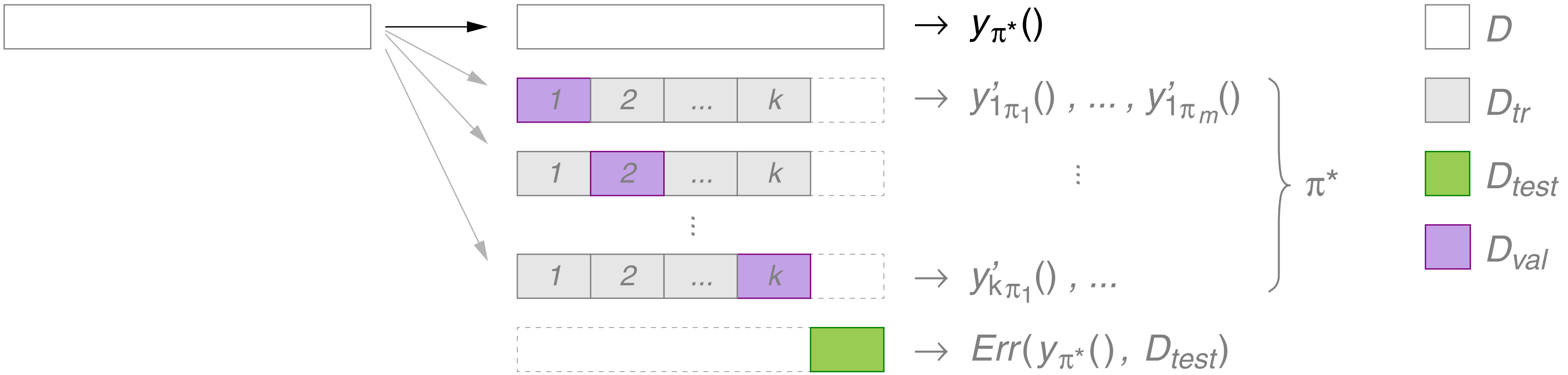
Evaluation setting:

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

$\vdots$

Evaluating Effectiveness

Model Selection: k validation sets (continued)



1
2
...
k
$\rightarrow y_{\pi^*}()$
$\rightarrow y'_{1\pi_1}(), \ldots, y'_{1\pi_m}()$
$\vdots$
$\rightarrow y'_{k\pi_1}(), \ldots$
π^*
$\rightarrow Err(y_{\pi^*}(), D_{test})$
D
D_{tr}
D_{test}
D_{val}

Evaluation setting:

$\vdots$

- ❑ $y_{\pi^*}()$ 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”.

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:

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

Remarks:

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

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