

Big Data Computing

Master's Degree in Computer Science

2021-2022

Gabriele Tolomei

Department of Computer Science

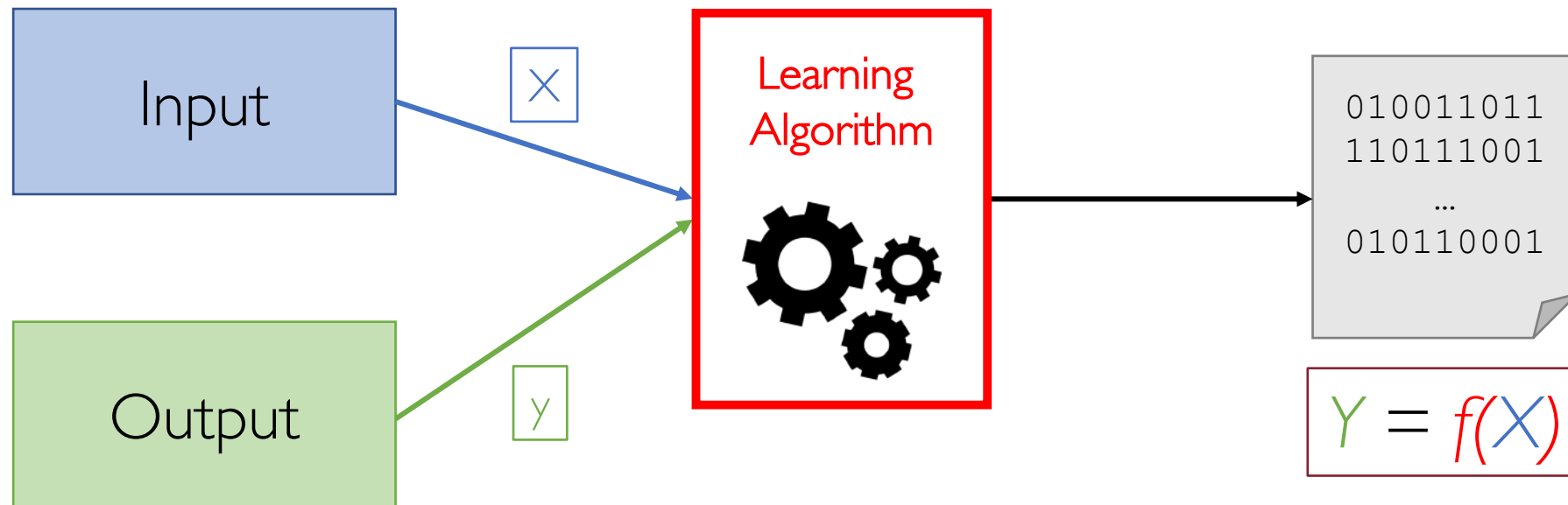
Sapienza Università di Roma

tolomei@di.uniroma1.it



SAPIENZA
UNIVERSITÀ DI ROMA

"Training" a Computer



Eventually, the function f is **learned** by the learning algorithm from a (large) set of **labeled data**

A Bit of Notation

$$\mathcal{X} \subseteq \mathbb{R}^n$$

input feature space

A Bit of Notation

$$\mathcal{X} \subseteq \mathbb{R}^n$$

\mathcal{Y}

input feature space

output space

A Bit of Notation

$$\mathcal{X} \subseteq \mathbb{R}^n$$

input feature space

$$\mathcal{Y}$$

output space

$$\mathcal{Y} \subseteq \mathbb{R}$$

real-value label (**regression**)

$$\mathcal{Y} = \{1, \dots, k\}$$

discrete-value label (**k-ary classification**)

A Bit of Notation

$$\mathcal{X} \subseteq \mathbb{R}^n$$

input feature space

$$\mathcal{Y}$$

output space

$$\mathcal{Y} \subseteq \mathbb{R}$$

real-value label (**regression**)

$$\mathcal{Y} = \{1, \dots, k\}$$

discrete-value label (**k-ary classification**)

$$(\mathbf{x}_i, y_i)$$

i -th labeled instance

A Bit of Notation

$$\mathcal{X} \subseteq \mathbb{R}^n$$

input feature space

$$\mathcal{Y}$$

output space

$$\mathcal{Y} \subseteq \mathbb{R}$$

real-value label (**regression**)

$$\mathcal{Y} = \{1, \dots, k\}$$

discrete-value label (**k-ary classification**)

$$(\mathbf{x}_i, y_i)$$

i -th labeled instance

$$\mathbf{x}_i = (x_{i,1}, \dots, x_{i,n}) \in \mathcal{X}$$

n -dimensional feature vector of the i -th instance

A Bit of Notation

$$\mathcal{X} \subseteq \mathbb{R}^n$$

input feature space

$$\mathcal{Y}$$

output space

$$\mathcal{Y} \subseteq \mathbb{R}$$

real-value label (**regression**)

$$\mathcal{Y} = \{1, \dots, k\}$$

discrete-value label (**k-ary classification**)

$$(\mathbf{x}_i, y_i)$$

i -th labeled instance

$$\mathbf{x}_i = (x_{i,1}, \dots, x_{i,n}) \in \mathcal{X}$$

n -dimensional feature vector of the i -th instance

$$y_i \in \mathcal{Y}$$

label of the i -th instance

A Bit of Notation

$$\mathcal{X} \subseteq \mathbb{R}^n$$

input feature space

$$\mathcal{Y}$$

output space

$$\mathcal{Y} \subseteq \mathbb{R}$$

real-value label (**regression**)

$$\mathcal{Y} = \{1, \dots, k\}$$

discrete-value label (**k-ary classification**)

$$(\mathbf{x}_i, y_i)$$

i -th labeled instance

$$\mathbf{x}_i = (x_{i,1}, \dots, x_{i,n}) \in \mathcal{X}$$

n -dimensional feature vector of the i -th instance

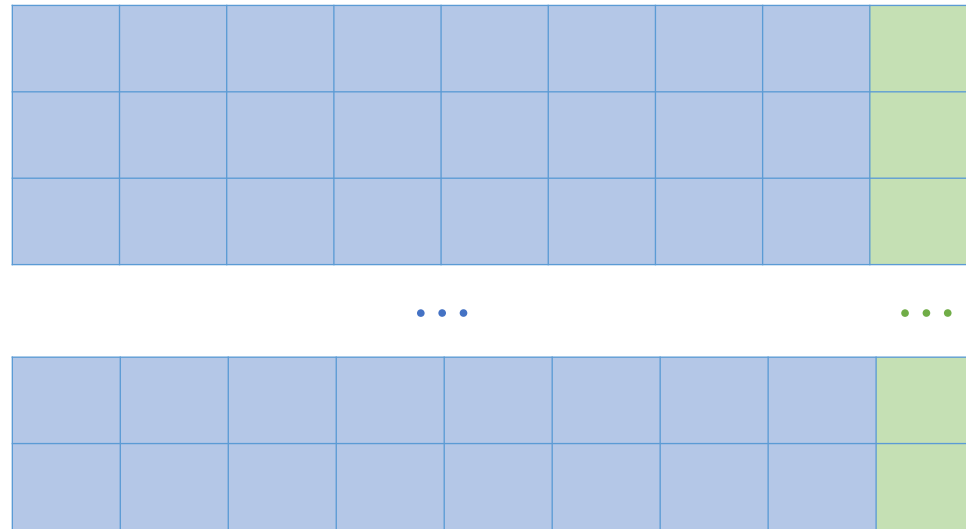
$$y_i \in \mathcal{Y}$$

label of the i -th instance

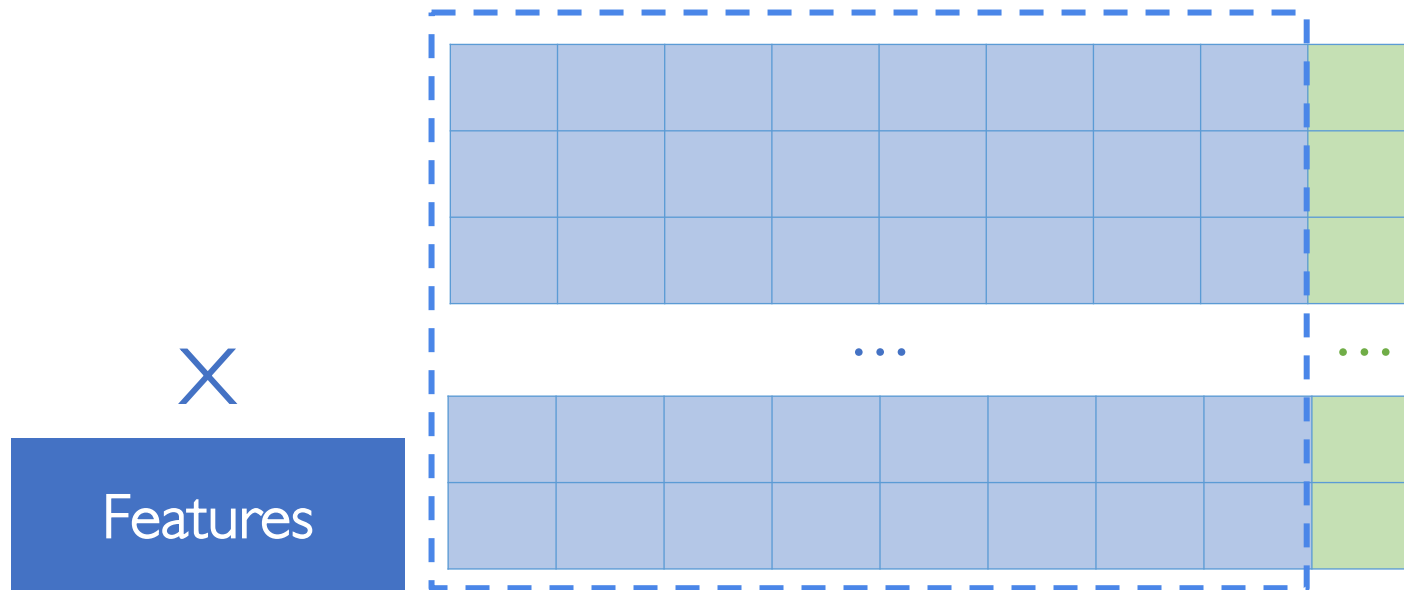
$$\mathcal{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$$

dataset of m **i.i.d.** labeled instances

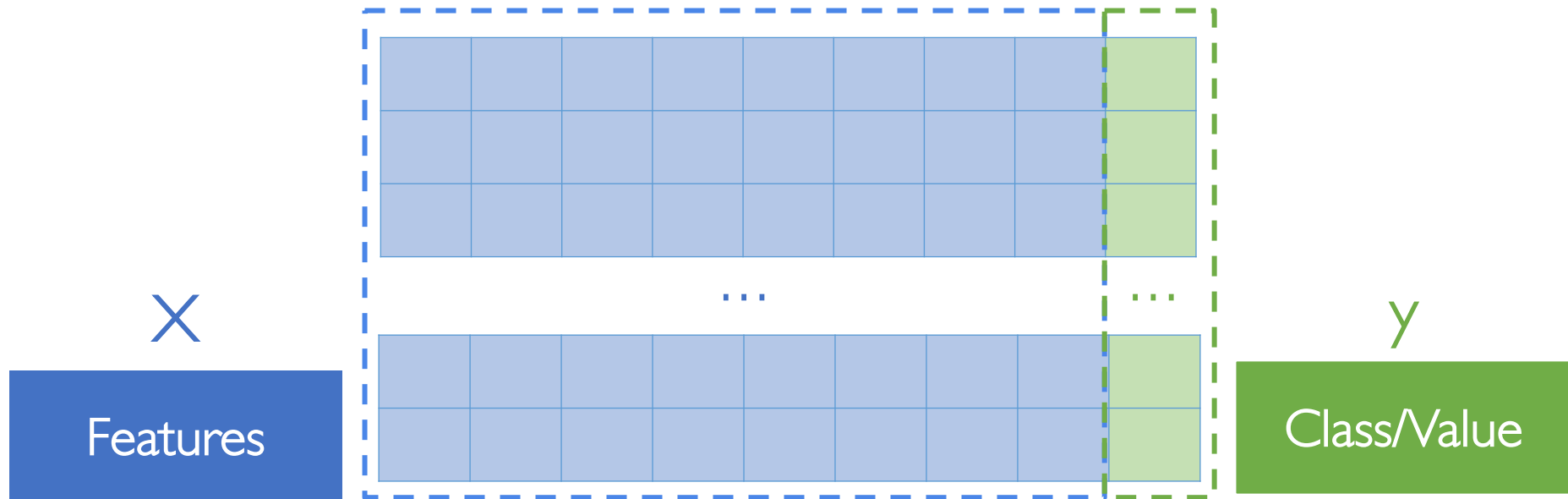
Model Training: Labeled Dataset



Model Training: Labeled Dataset

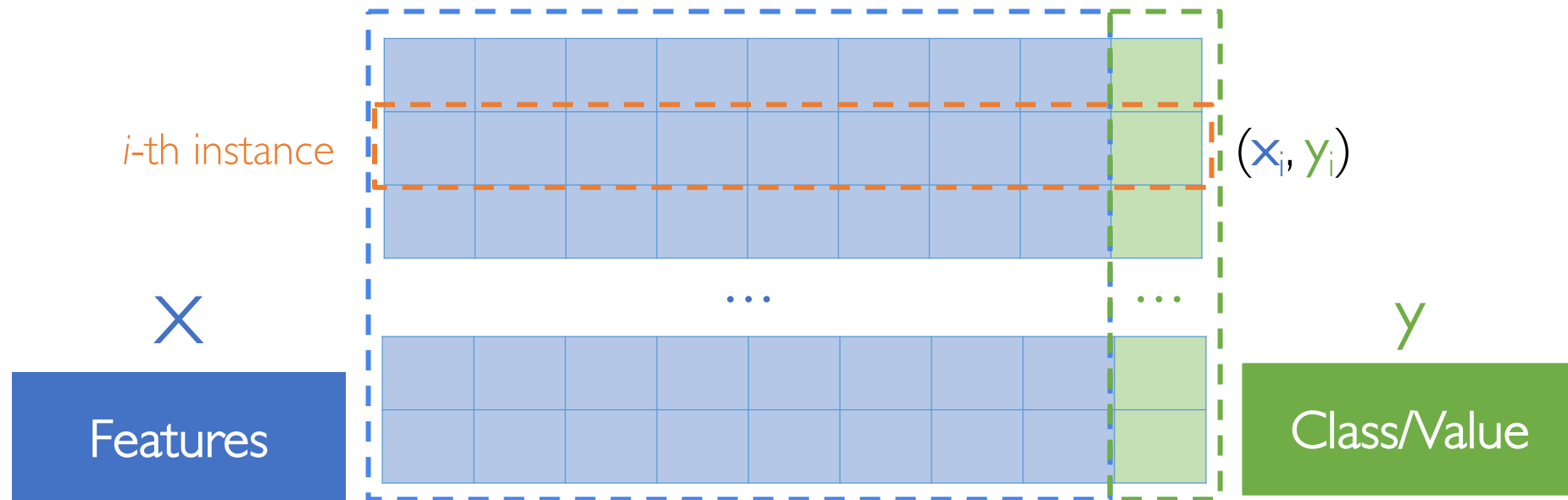


Model Training: Labeled Dataset



Model Training: Labeled Dataset

Each instance comes with the **class label** (**classification**) or the **value** (**regression**) we want to predict



Model Training: Intuition

Idea

There is an **unknown target function** f which puts in a relationship elements of X with elements of Y

Model Training: Intuition

Idea

There is an **unknown target function** f which puts in a relationship elements of X with elements of Y



$$f = X \rightarrow Y$$

Model Training: Intuition

Idea

There is an **unknown target function** f which puts in a relationship elements of X with elements of Y



$$f = X \rightarrow Y$$



Problem

We cannot write down an algorithm which just implements f

Model Training: Learning f from Labeled Data

- Learning f means "finding" another function h^* which best approximates f using the data we observed

Model Training: Learning f from Labeled Data

- Learning f means "finding" another function h^* which best approximates f using the data we observed
- h^* is chosen among a family of functions H called **hypothesis space** by specifying two components:

Model Training: Learning f from Labeled Data

- Learning f means "finding" another function h^* which best approximates f using the data we observed
- h^* is chosen among a family of functions H called **hypothesis space** by specifying two components:
 - **loss function**: measures the error of using h^* instead of the true f

Model Training: Learning f from Labeled Data

- Learning f means "finding" another function h^* which best approximates f using the data we observed
- h^* is chosen among a family of functions H called **hypothesis space** by specifying two components:
 - **loss function**: measures the error of using h^* instead of the true f
 - **learning algorithm**: explores the hypothesis space to pick the function which minimizes the loss on the observed data

The Hypothesis Space H



- The set of functions the learning algorithm will search through to pick the hypothesis h^* which best approximates the true target f

The Hypothesis Space H



- The set of functions the learning algorithm will search through to pick the hypothesis h^* which best approximates the true target f
- The larger the hypothesis space:
 - the larger will be the set of functions that can be represented



The Hypothesis Space H

- The set of functions the learning algorithm will search through to pick the hypothesis h^* which best approximates the true target f
- The larger the hypothesis space:
 - the larger will be the set of functions that can be represented 
 - the harder will be for the learning algorithm to pick h^* 

The Hypothesis Space H

- The set of functions the learning algorithm will search through to pick the hypothesis h^* which best approximates the true target f
- The larger the hypothesis space:
 - the larger will be the set of functions that can be represented 
 - the harder will be for the learning algorithm to pick h^* 

Trade-off

Put some constraints on H , e.g., limit the search space only to **linear functions**

The Loss Function

- Measures the error we would make if a hypothesis h is used instead of the true (yet unknown) mapping f

The Loss Function

- Measures the error we would make if a hypothesis h is used instead of the true (yet unknown) mapping f
- It can be computed only on the data we observed, therefore depends on the hypothesis and the dataset

$$L : \mathcal{H} \times \mathcal{D} \mapsto \mathbb{R}$$

The Loss Function

- Measures the error we would make if a hypothesis h is used instead of the true (yet unknown) mapping f
- It can be computed only on the data we observed, therefore depends on the hypothesis and the dataset

$$L : \mathcal{H} \times \mathcal{D} \mapsto \mathbb{R}$$

- This **in-sample error** (a.k.a. empirical loss) is an estimate of the **out-of-sample error** (a.k.a. expected loss or risk)

The Learning Algorithm

- Defines the strategy we use to search the hypothesis space H for picking our **best** hypothesis h^*

The Learning Algorithm

- Defines the strategy we use to search the hypothesis space H for picking our **best** hypothesis h^*
- Here, "**best**" means the hypothesis that **minimizes** the loss function on the observed data (**Empirical Risk Minimization**)

The Learning Algorithm

- Defines the strategy we use to search the hypothesis space H for picking our **best** hypothesis h^*
- Here, "**best**" means the hypothesis that **minimizes** the loss function on the observed data (**Empirical Risk Minimization**)
- In other words, among all the hypotheses specified by H the learning algorithm will pick the one that minimizes L

The Learning Algorithm

- Defines the strategy we use to search the hypothesis space H for picking our **best** hypothesis h^*
- Here, "**best**" means the hypothesis that **minimizes** the loss function on the observed data (**Empirical Risk Minimization**)
- In other words, among all the hypotheses specified by H the learning algorithm will pick the one that minimizes L

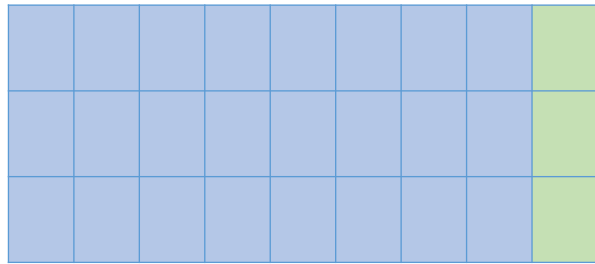
$$h^* = \operatorname{argmin}_{h \in \mathcal{H}} L(h, \mathcal{D})$$

unknown target
(e.g., ideal credit approval function)

$$f = X \rightarrow Y$$

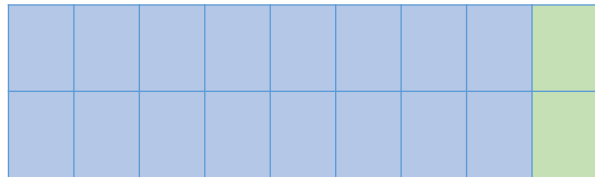
unknown target
(e.g., ideal credit approval function)

$$f = X \rightarrow Y$$



A diagram showing a grid of training data. It consists of two rows of three blue squares each, followed by a vertical ellipsis. To the right of the blue squares is a column of three green squares, followed by a horizontal ellipsis. This represents a dataset with multiple features (blue) and a target variable (green).

...



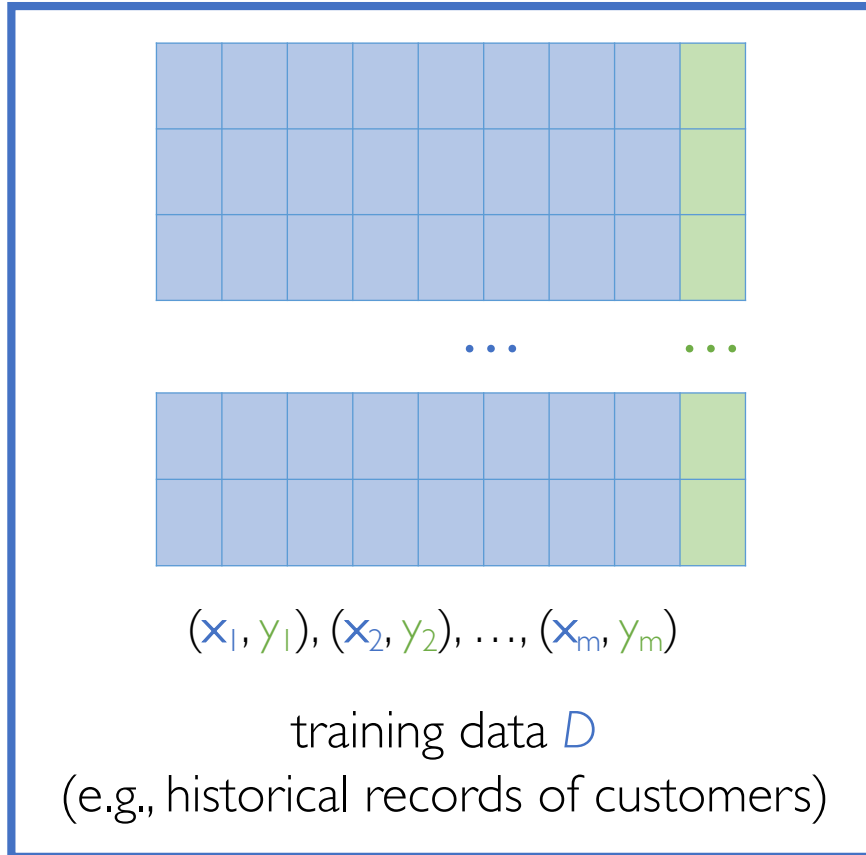
A second instance of the training data grid, identical to the one above, showing two rows of three blue squares and a column of three green squares.

$(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)$

training data D
(e.g., historical records of customers)

unknown target
(e.g., ideal credit approval function)

$$f = X \rightarrow Y$$



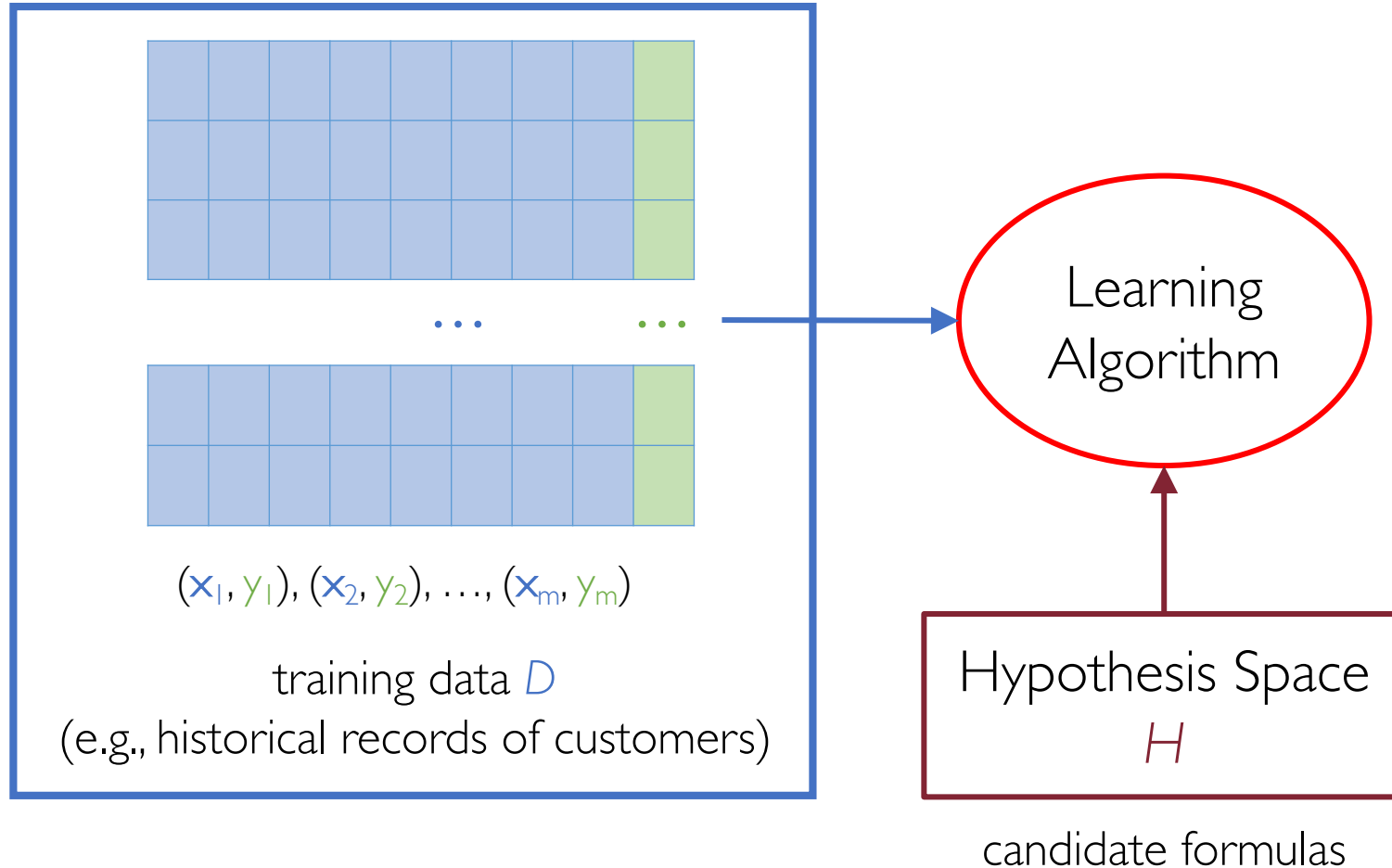
Hypothesis Space

H

candidate formulas

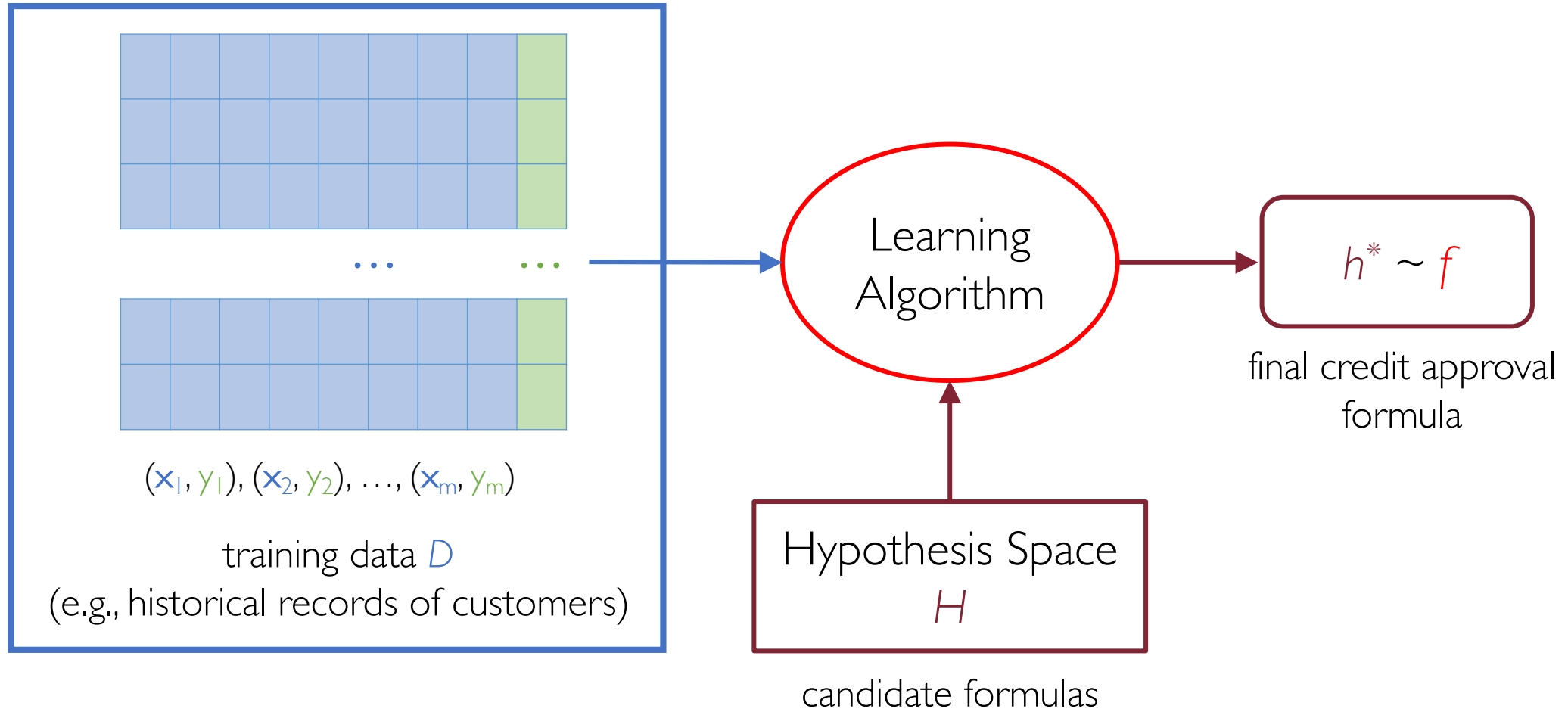
unknown target
(e.g., ideal credit approval function)

$$f = X \rightarrow Y$$



unknown target
(e.g., ideal credit approval function)

$$f = X \rightarrow Y$$



Learning f as an Optimization Problem

- We define the supervised learning problem as an optimization one

Learning f as an Optimization Problem

- We define the supervised learning problem as an optimization one
- By plugging in different loss functions combined with various hypothesis spaces we must solve a specific optimization problem

Learning f as an Optimization Problem

- We define the supervised learning problem as an optimization one
- By plugging in different loss functions combined with various hypothesis spaces we must solve a specific optimization problem
- Those choices are usually "mathematically convenient": e.g., **convex objective functions** are guaranteed to have a unique global minimum

Learning f as an Optimization Problem

- We define the supervised learning problem as an optimization one
- By plugging in different loss functions combined with various hypothesis spaces we must solve a specific optimization problem
- Those choices are usually "mathematically convenient": e.g., **convex objective functions** are guaranteed to have a unique global minimum
- Even though closed-form solutions to the optimization problem rarely exist, there are numerical methods which work: e.g., **gradient descent**

In-Sample vs. Out-of-Sample Error

- Minimizing the loss function on the observed data D just limits the in-sample error

In-Sample vs. Out-of-Sample Error

- Minimizing the loss function on the observed data D just limits the in-sample error
- Our ultimate hypothesis is to pick h^* which is able to generalize to **unseen instances** (i.e., minimize the out-of-sample error)

In-Sample vs. Out-of-Sample Error

- Minimizing the loss function on the observed data D just limits the in-sample error
- Our ultimate hypothesis is to pick h^* which is able to generalize to **unseen instances** (i.e., minimize the out-of-sample error)
- If we pick a hypothesis which just memorizes all the training instances, we will obtain a 0 in-sample error but this is not learning!

In-Sample vs. Out-of-Sample Error

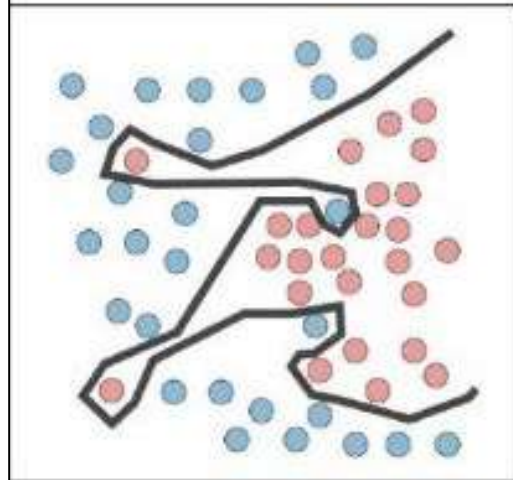
- Minimizing the loss function on the observed data D just limits the in-sample error
- Our ultimate hypothesis is to pick h^* which is able to generalize to **unseen instances** (i.e., minimize the out-of-sample error)
- If we pick a hypothesis which just memorizes all the training instances, we will obtain a 0 in-sample error but this is not learning!
- At the same time we do not want h^* to perform poorly on D

Overfitting (High Variance)

Regression



Classification



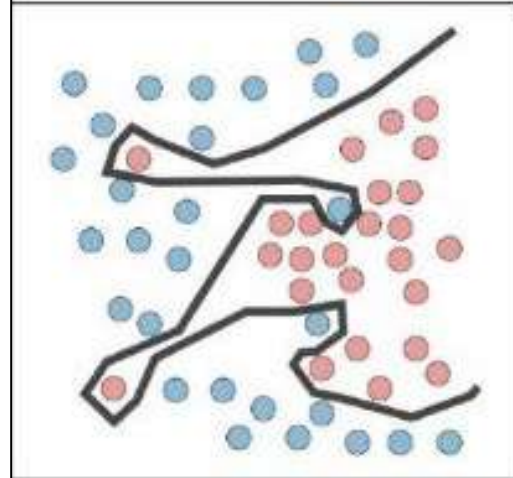
The hypothesis h^* is not learning the true f but it mimics its noise

Overfitting (High Variance)

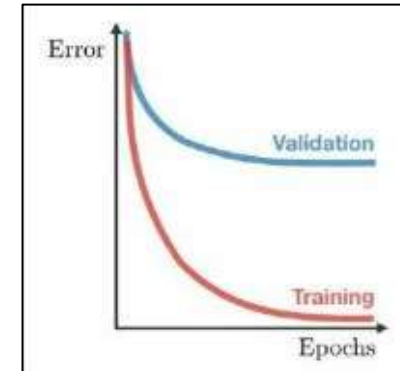
Regression



Classification



The hypothesis h^* is not learning the true f but it mimics its noise



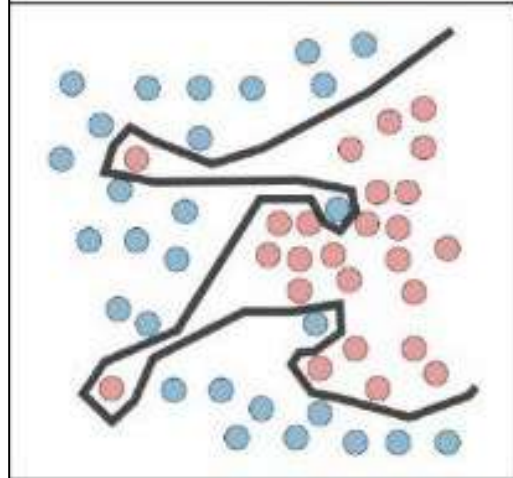
low in-sample error high out-of-sample error

Overfitting (High Variance)

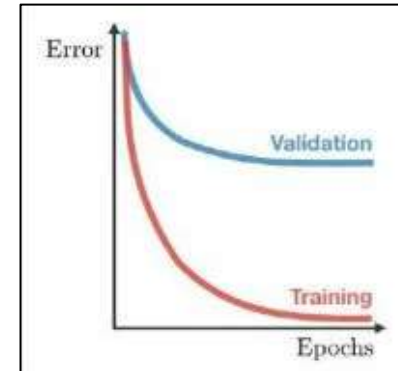
Regression



Classification



The hypothesis h^* is not learning the true f but it mimics its noise

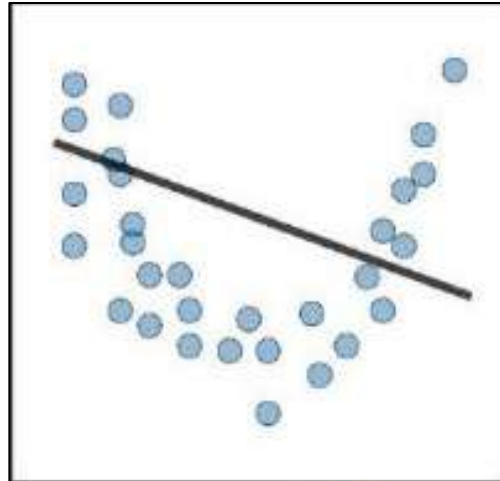


low in-sample error high out-of-sample error

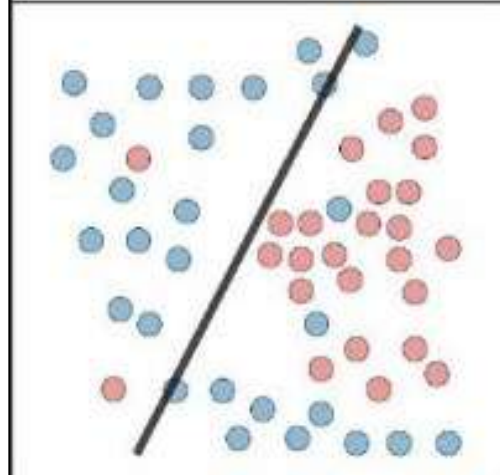
- Regularization
- Get more data

Underfitting (High Bias)

Regression



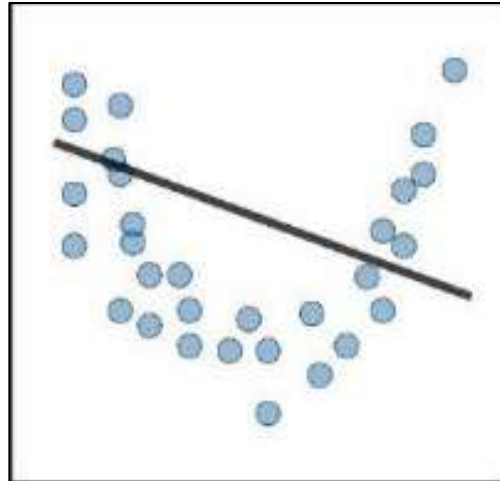
Classification



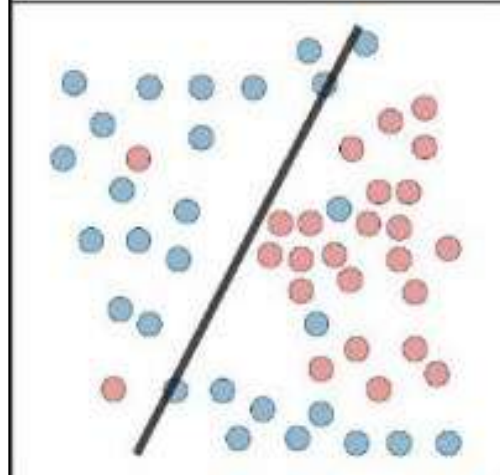
The hypothesis h^* is too "simple" for approximating the true f

Underfitting (High Bias)

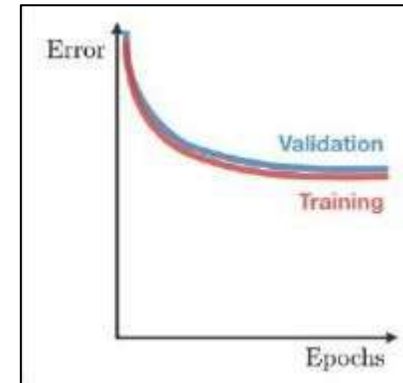
Regression



Classification



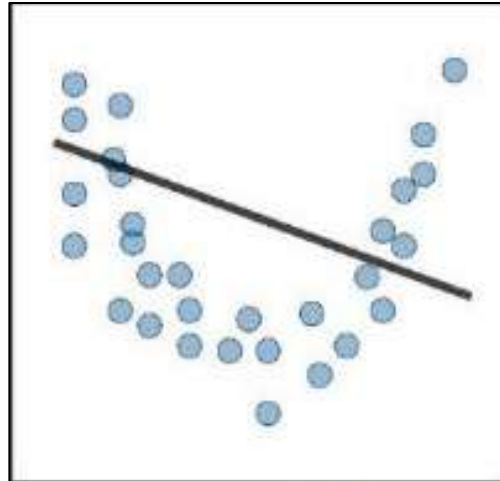
The hypothesis h^* is too "simple" for approximating the true f



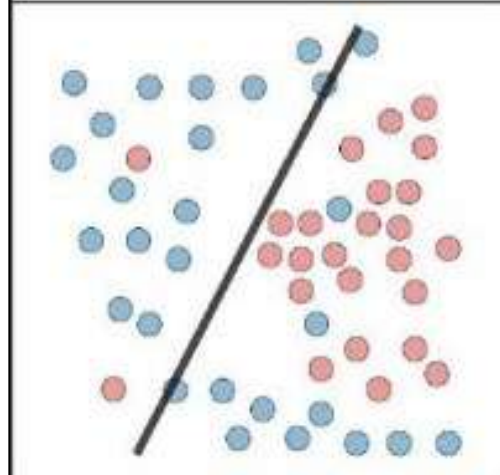
high in-sample error high out-of-sample error

Underfitting (High Bias)

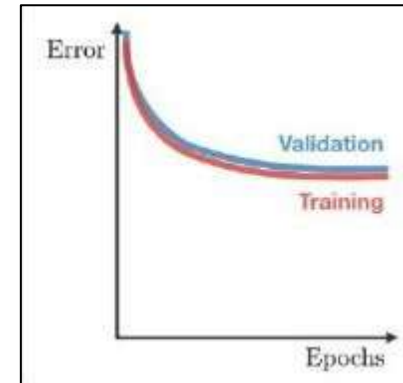
Regression



Classification



The hypothesis h^* is too "simple" for approximating the true f



high in-sample error high out-of-sample error

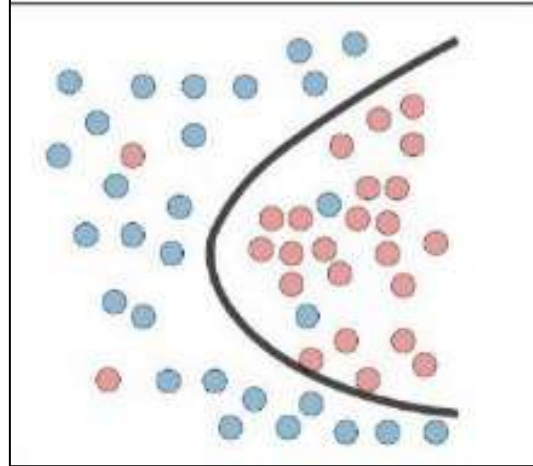
- Increase model complexity
- Add more features

Bias-Variance Tradeoff

Regression



Classification



The hypothesis h^* is just right:
the simplest one explaining the data

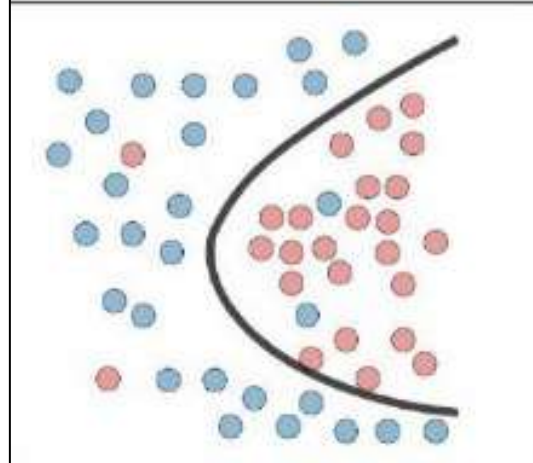
Occam's razor

Bias-Variance Tradeoff

Regression

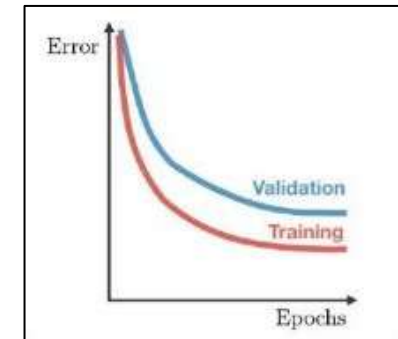


Classification



The hypothesis h^* is just right:
the simplest one explaining the data

Occam's razor



low in-sample error low out-of-sample error

Estimating Generalization Performance

- Measuring the generalization (i.e., out-of-sample) performance online may be too risky

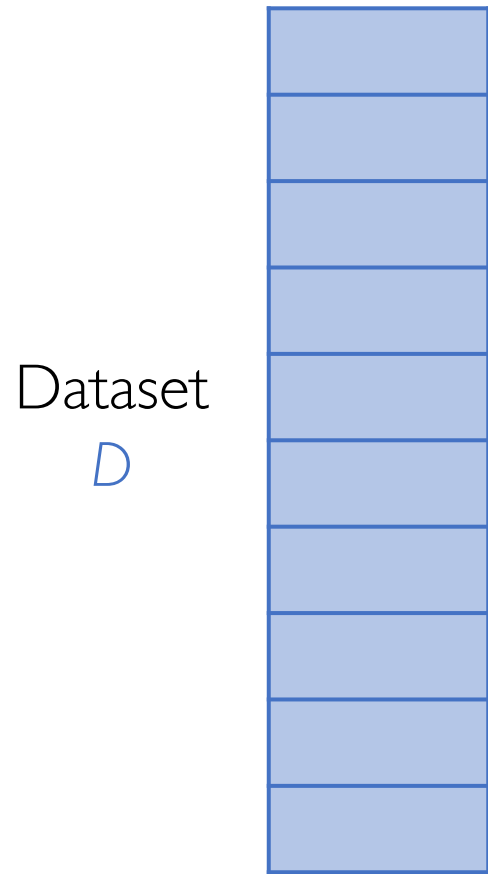
Estimating Generalization Performance

- Measuring the generalization (i.e., out-of-sample) performance online may be too risky
- **Example:** Don't want to deploy your new spam classifier in production knowing only its training (i.e., in-sample) performance

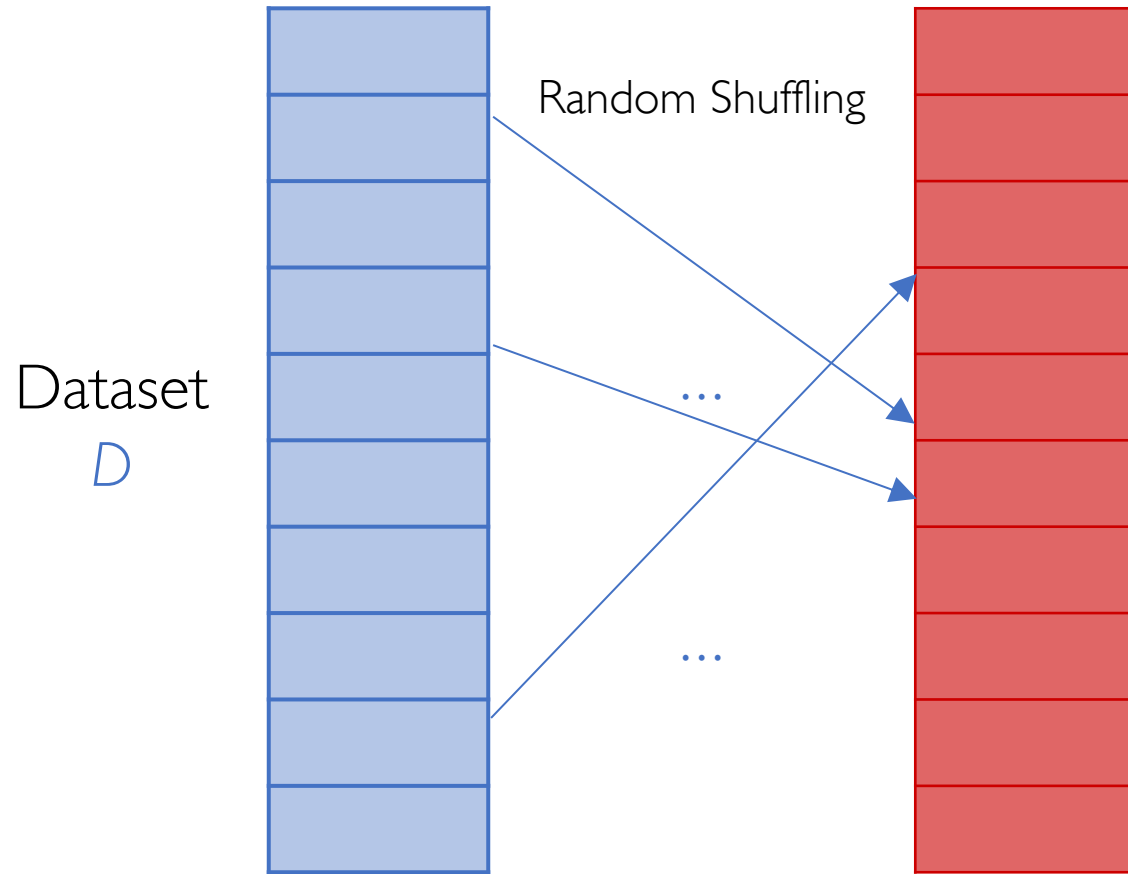
Estimating Generalization Performance

- Measuring the generalization (i.e., out-of-sample) performance online may be too risky
- **Example:** Don't want to deploy your new spam classifier in production knowing only its training (i.e., in-sample) performance
- **Solution:** Estimate the generalization performance using training set
 - As long as it holds true the assumption that training and test instances are both drawn from the same probability distribution (**i.i.d. assumption**)

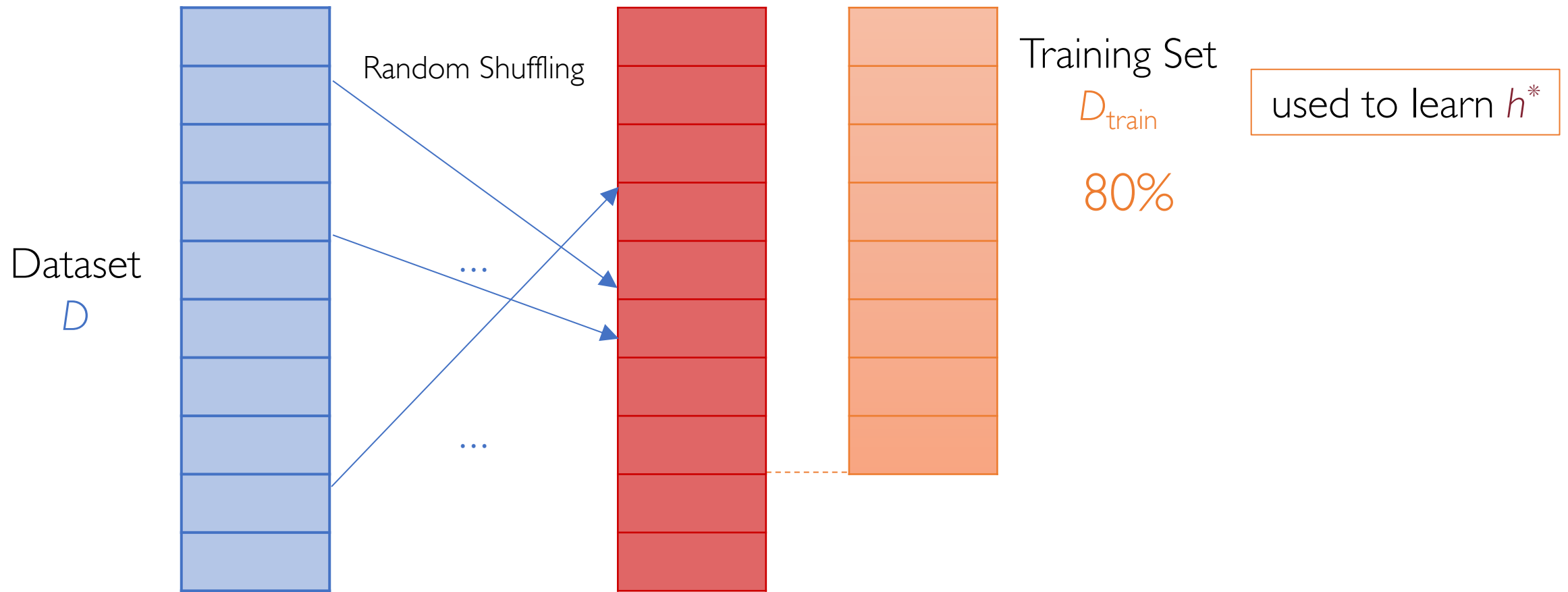
Dataset Splitting



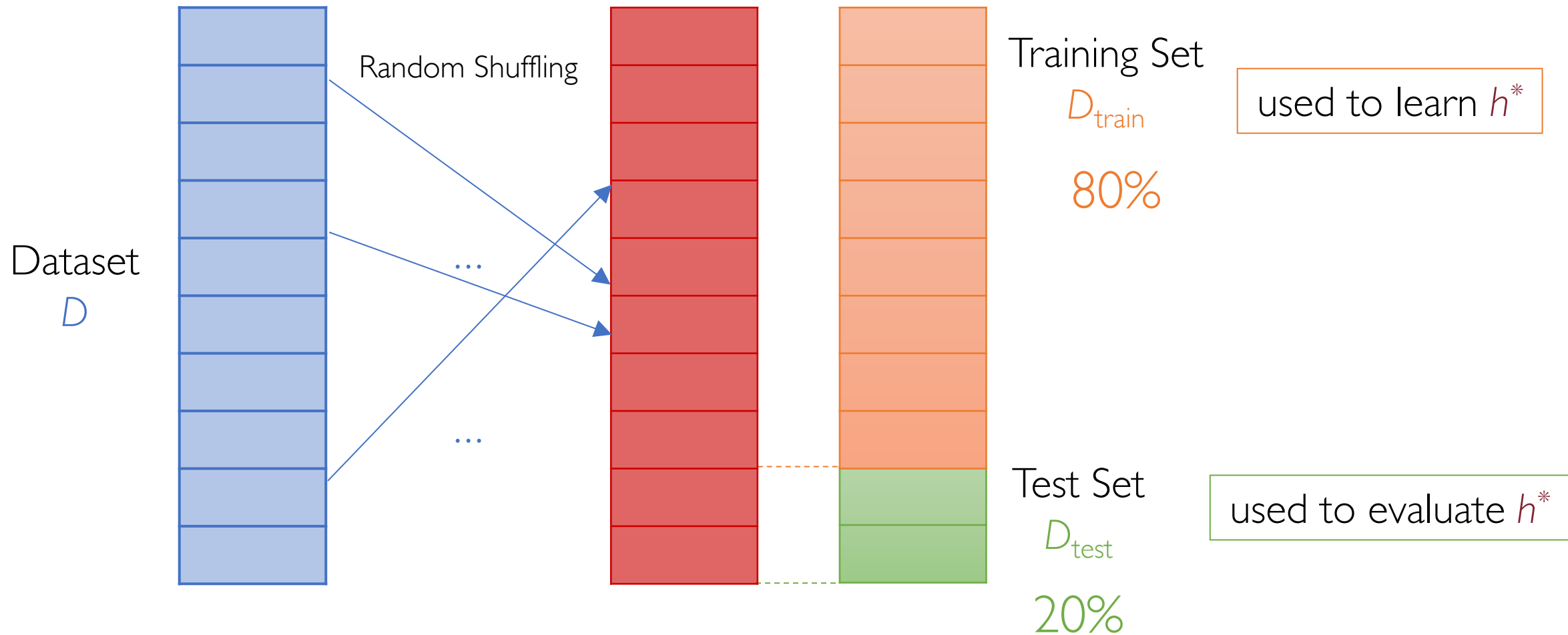
Dataset Splitting



Dataset Splitting

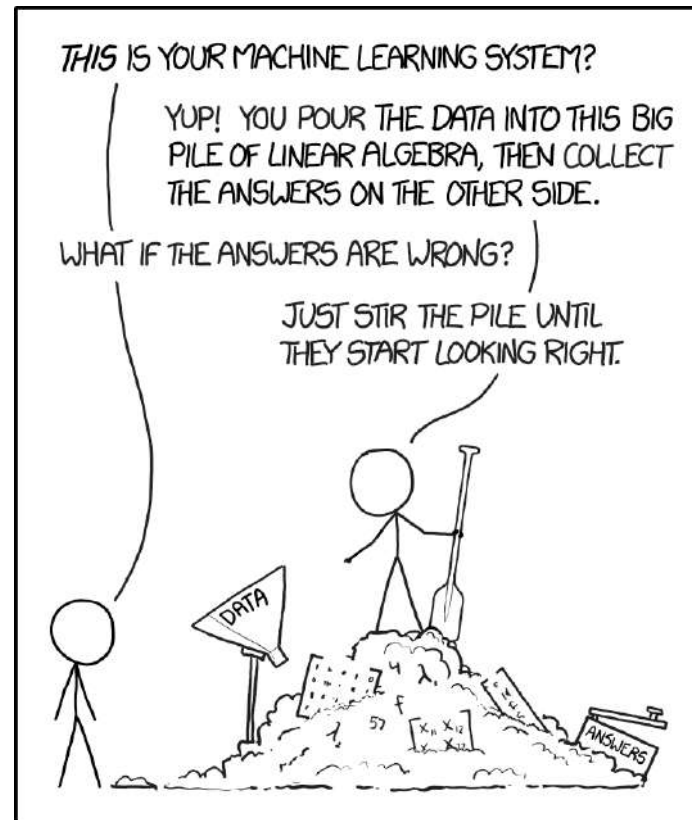


Dataset Splitting



How Much Data Do We Need?

In general, the more data we have the better we learn



K-Fold Cross Validation

- A generalization of the training/test splitting seen before

K-Fold Cross Validation

- A generalization of the training/test splitting seen before
- Pick a value for K (e.g., $K=5$ or 10)

K-Fold Cross Validation

- A generalization of the training/test splitting seen before
- Pick a value for K (e.g., $K=5$ or 10)
- Divide your dataset D into K distinct folds

K-Fold Cross Validation

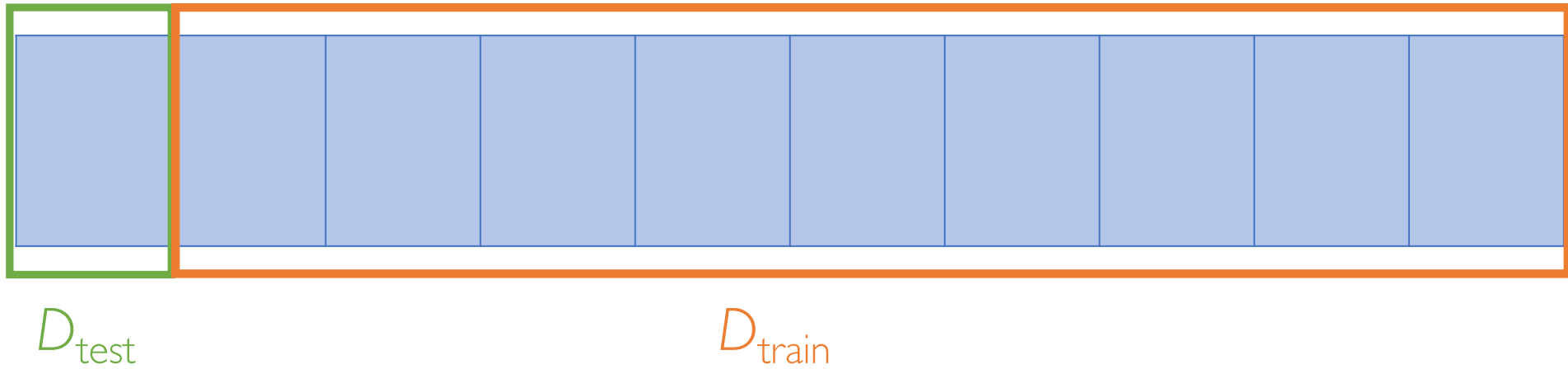
- A generalization of the training/test splitting seen before
- Pick a value for K (e.g., $K=5$ or 10)
- Divide your dataset D into K distinct folds
- Perform K rounds where h^* is:
 - learned from $K-1$ training folds
 - evaluated on 1 remaining test fold

K-Fold Cross Validation

- A generalization of the training/test splitting seen before
- Pick a value for K (e.g., $K=5$ or 10)
- Divide your dataset D into K distinct folds
- Perform K rounds where h^* is:
 - learned from $K-1$ training folds
 - evaluated on 1 remaining test fold
- The estimate of generalization error is the average across the K test folds of all the K rounds

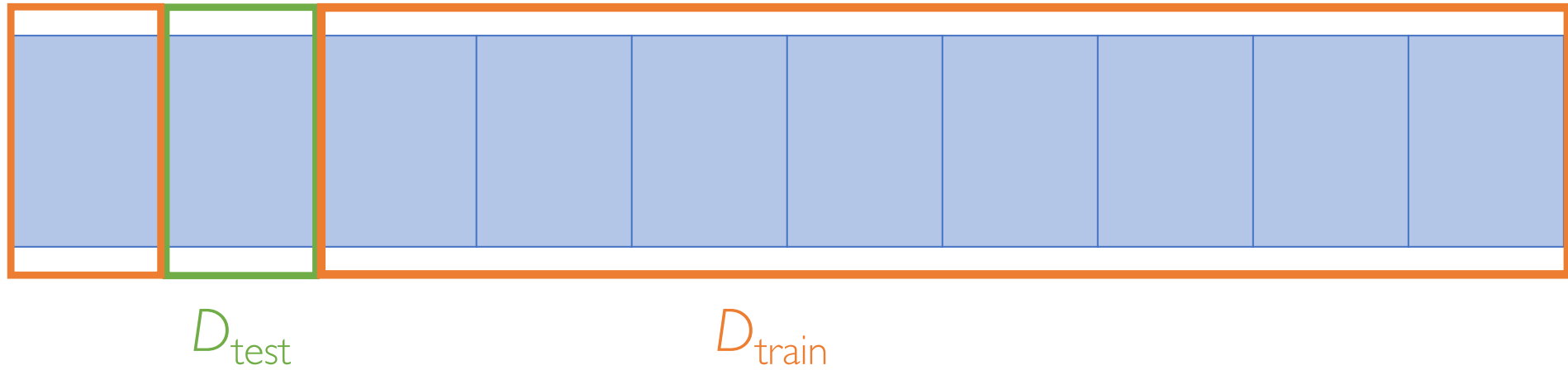
K-Fold Cross Validation

Round $k = 1$



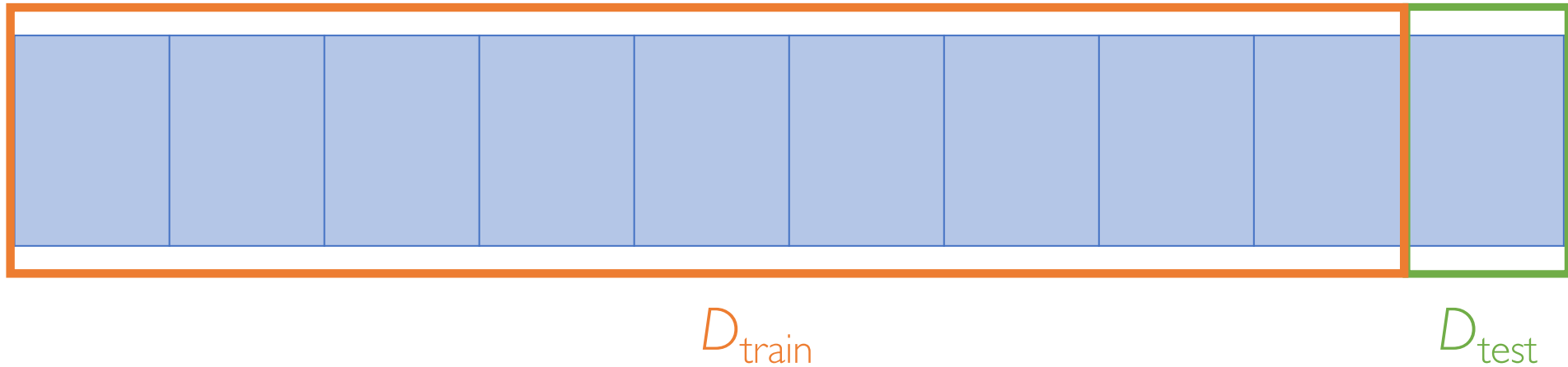
K-Fold Cross Validation

Round $k = 2$



K-Fold Cross Validation

Round $k = 10$



Model Selection/Evaluation

Several different learning models to achieve the same task



Model Selection/Evaluation

Several different learning models to achieve the same task



Each learning model has its own set of **hyperparameters** (e.g., the number k of neighbors in kNN)

Model Selection/Evaluation

Several different learning models to achieve the same task



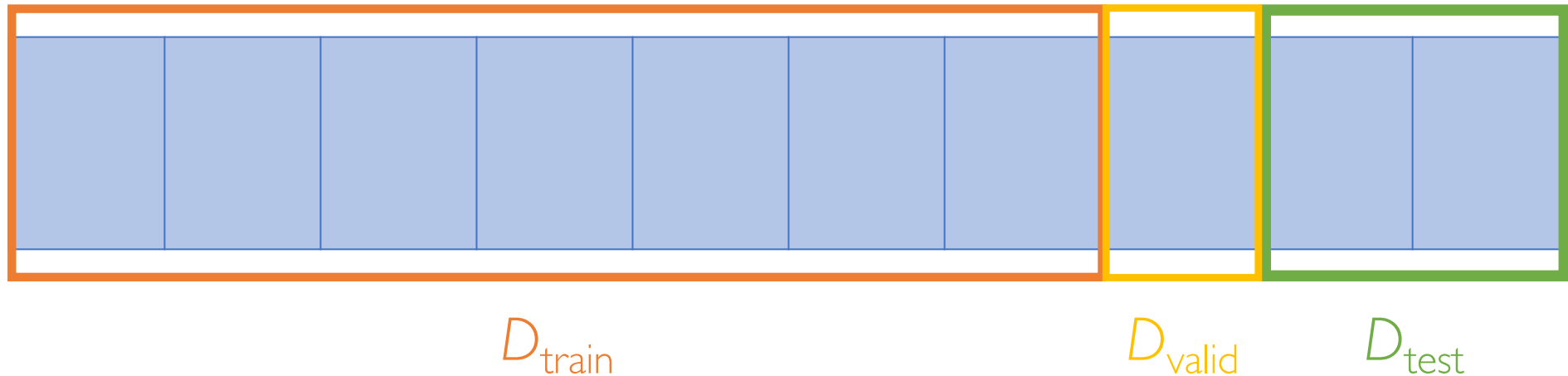
Each learning model has its own set of **hyperparameters** (e.g., the number k of neighbors in kNN)

How do we select the best model?

Model Selection/Evaluation: Validation Set

Separate hyperparameter selection from model evaluation

D_{valid} is used to validate hyperparameters



Model Selection/Evaluation: Example

Select which value of $k = \{2, 5, 10\}$ of a k NN gives the best performance

Model Selection/Evaluation: Example

Select which value of $k = \{2, 5, 10\}$ of a k NN gives the best performance

- 1) Train a separate model for each value of k on the training set (e.g., 70%)

Model Selection/Evaluation: Example

Select which value of $k = \{2, 5, 10\}$ of a k NN gives the best performance

- 1) Train a separate model for each value of k on the training set (e.g., 70%)
- 2) Measure the error of each model on the validation set (e.g., 10%)

Model Selection/Evaluation: Example

Select which value of $k = \{2, 5, 10\}$ of a k NN gives the best performance

- 1) Train a separate model for each value of k on the training set (e.g., 70%)
- 2) Measure the error of each model on the validation set (e.g., 10%)
- 3) Select the model whose value of k gives the best performance on the validation set (e.g., $k = 5$)

Model Selection/Evaluation: Example

Select which value of $k = \{2, 5, 10\}$ of a k NN gives the best performance

- 1) Train a separate model for each value of k on the training set (e.g., 70%)
- 2) Measure the error of each model on the validation set (e.g., 10%)
- 3) Select the model whose value of k gives the best performance on the validation set (e.g., $k = 5$)
- 4) Re-train only this model on the training + validation set

Model Selection/Evaluation: Example

Select which value of $k = \{2, 5, 10\}$ of a k NN gives the best performance

- 1) Train a separate model for each value of k on the training set (e.g., 70%)
- 2) Measure the error of each model on the validation set (e.g., 10%)
- 3) Select the model whose value of k gives the best performance on the validation set (e.g., $k = 5$)
- 4) Re-train only this model on the training + validation set
- 5) Measure the performance on the test set (e.g., 20%)

Model Selection/Evaluation: Example

Select which value of $k = \{2, 5, 10\}$ of a k NN gives the best performance

- 1) Train a separate model for each value of k on the training set (e.g., 70%)
- 2) Measure the error of each model on the validation set (e.g., 10%)
- 3) Select the model whose value of k gives the best performance on the validation set (e.g., $k = 5$)
- 4) Re-train only this model on the training + validation set
- 5) Measure the performance on the test set (e.g., 20%)

Note:

The strategy above can also be extended to K-fold Cross Validation

Take-Home Message of Today

- Supervised Learning as an optimization problem
 - Hypothesis space (assumption)
 - Loss Function (objective)
 - Learning Algorithm (optimizer)

Take-Home Message of Today

- Supervised Learning as an optimization problem
 - Hypothesis space (assumption)
 - Loss Function (objective)
 - Learning Algorithm (optimizer)
- Regression vs. Classification

Take-Home Message of Today

- Supervised Learning as an optimization problem
 - Hypothesis space (assumption)
 - Loss Function (objective)
 - Learning Algorithm (optimizer)
- Regression vs. Classification
- Bias-Variance Tradeoff

Take-Home Message of Today

- Supervised Learning as an optimization problem
 - Hypothesis space (assumption)
 - Loss Function (objective)
 - Learning Algorithm (optimizer)
- Regression vs. Classification
- Bias-Variance Tradeoff
- Model selection vs. Model evaluation

Take-Home Message of Today

- Supervised Learning as an optimization problem
 - Hypothesis space (assumption)
 - Loss Function (objective)
 - Learning Algorithm (optimizer)
- Regression vs. Classification
- Bias-Variance Tradeoff
- Model selection vs. Model evaluation

Suggested reading: <https://homes.cs.washington.edu/~pedrod/papers/cacml2.pdf>