Big Data Computing

Master's Degree in Computer Science

2019-2020

Gabriele Tolomei

Department of Computer Science

Sapienza Università di Roma

tolomei@di.uniroma1.it

SAPIENZA
UNIVERSITÀ DI ROMA

Recap from Last Lecture(s)

2 unsupervised learning techniques to extract "structural" patterns from raw data

Recap from Last Lecture(s)

2 unsupervised learning techniques to extract "structural" patterns from raw data

Clustering

- Group together similar object according to a specific distance function
- Formalized as an NP-hard optimization problem
- K-means and its variants as effective heuristics that work in practice

Recap from Last Lecture(s)

2 unsupervised learning techniques to extract "structural" patterns from raw data

Clustering

- Group together similar object according to a specific distance function
- Formalized as an NP-hard optimization problem
- K-means and its variants as effective heuristics that work in practice

Principal Component Analysis (PCA)

- Reduce data dimensionality
- Automatically extract features from raw data
- Resort to compute the eigenvectors and eigenvalues of the covariance matrix

SUPERVISED LEARNING

Human vs. Computer

- Computers are designed to be **programmed** by humans in order to solve a task/problem quicker and better than humans

Human vs. Computer

- Computers are designed to be **programmed** by humans in order to solve a task/problem quicker and better than humans
- **Example**

Human vs. Computer

- Computers are designed to be **programmed** by humans in order to solve a task/problem quicker and better than humans
- **Example**
  - Task/Problem: Find the maximum element of a list of 1 million numbers

Human vs. Computer

- Computers are designed to be **programmed** by humans in order to solve a task/problem quicker and better than humans
- **Example**
  - Task/Problem: Find the maximum element of a list of 1 million numbers
  - Solution/Algorithm: Scan all the numbers in the set and keep track of the largest found "so far"

Human vs. Computer

- Computers are designed to be **programmed** by humans in order to solve a task/problem quicker and better than humans
- **<u>Example</u>**
  - Task/Problem: Find the maximum element of a list of 1 million numbers
  - Solution/Algorithm: Scan all the numbers in the set and keep track of the largest found "so far"
  - Code/Program: Encode the algorithm above into one specific programming language (e.g., C/C++, Java, Python)

Programming a Computer

Problem

Programming a Computer

Problem

Solution/Algorithm
explicitly designed by human

Programming a Computer

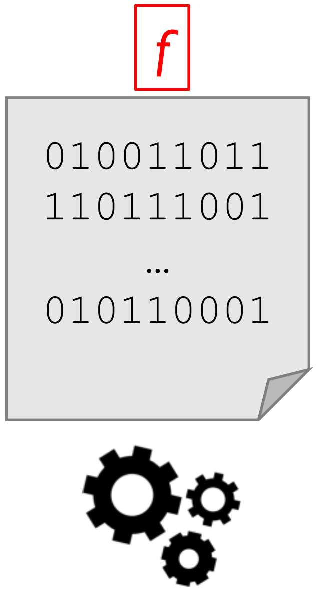
Problem

Solution/Algorithm
explicitly designed by human

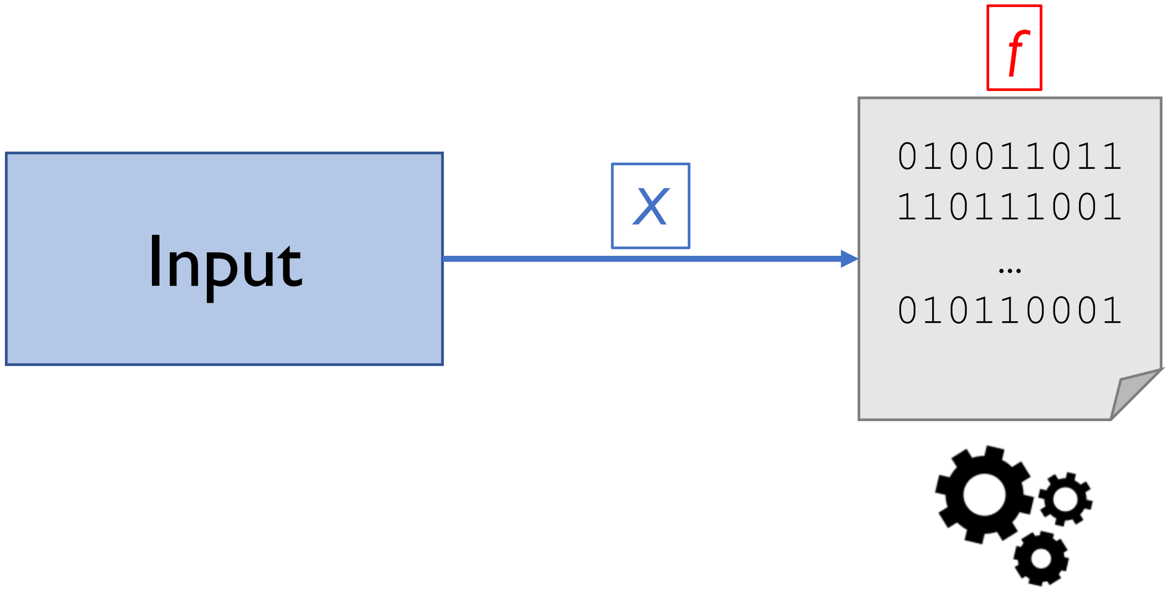
010011011
110111001
…
010110001

Encoding of
Computable Function f

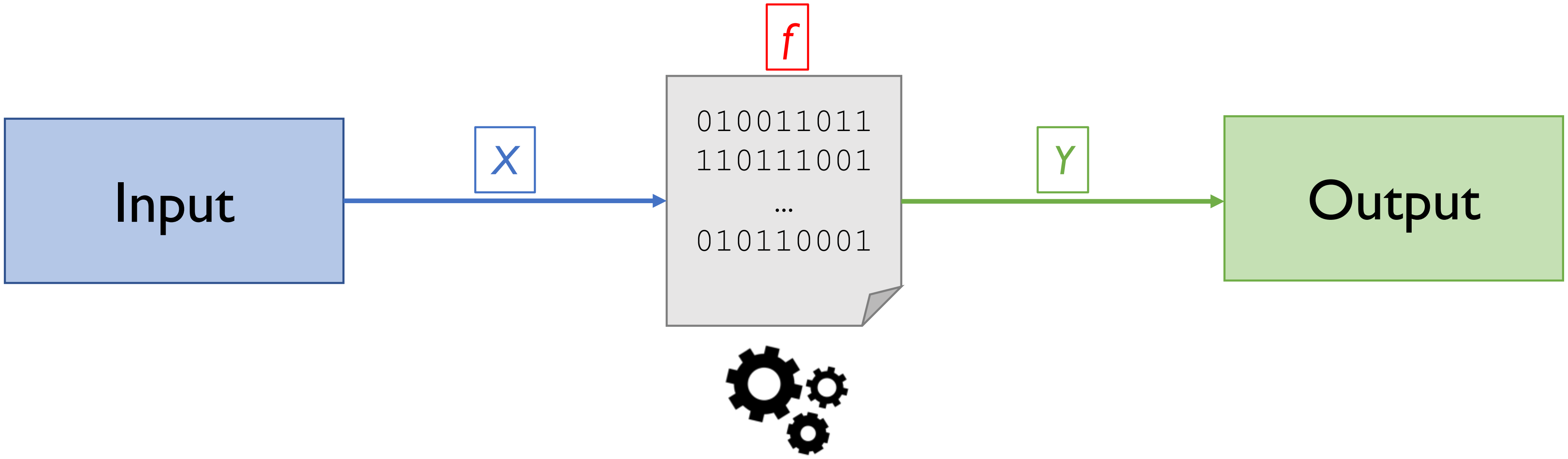
Programming a Computer



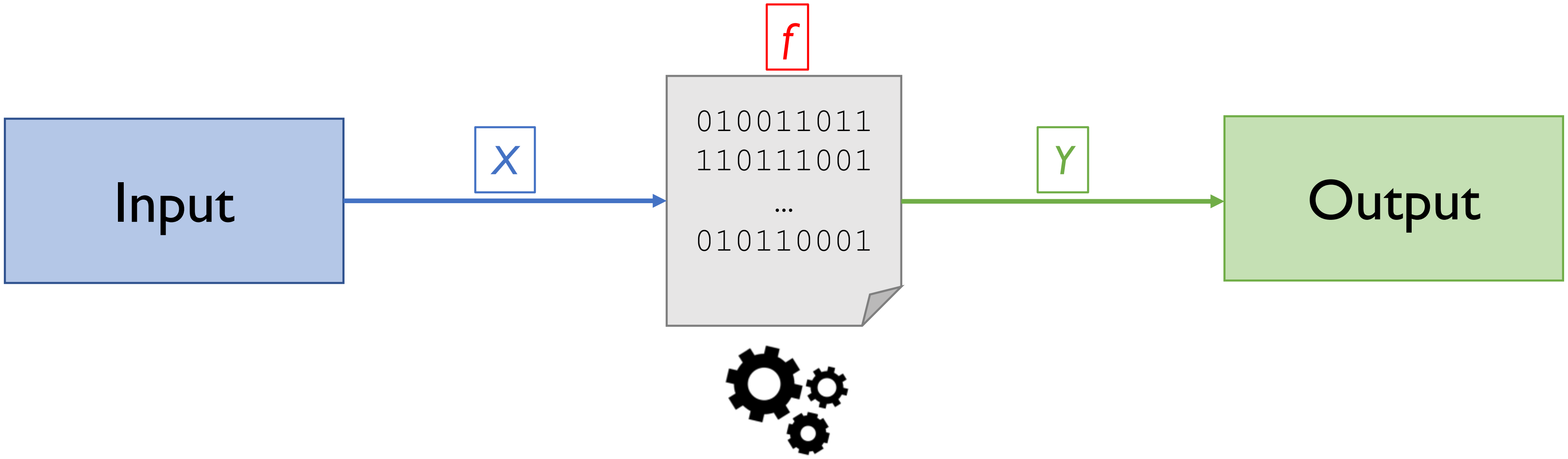
Programming a Computer



Programming a Computer



Programming a Computer



$$\boxed{Y = f(X)}$$

Can We Always Do That?

Chihuahua or Muffin?

[Copyright @teenybiscuit]

Chihuahua

Muffin

Programming vs. "Training" a Computer

- There exist some problems like the chihuahua vs. muffin above which are too hard to be solved directly

Programming vs. "Training" a Computer

- There exist some problems like the chihuahua vs. muffin above which are too hard to be solved directly
- Hard to design an algorithm which is general enough to capture all the nuances of the problem and gives the correct output for any input

Programming vs. "Training" a Computer

- There exist some problems like the chihuahua vs. muffin above which are too hard to be solved directly
- Hard to design an algorithm which is general enough to capture all the nuances of the problem and gives the correct output for any input

⇩

$\boxed{\text{Programming vs. "Training" a Computer}}$

"Training" a Computer

Problem

Labeled Data

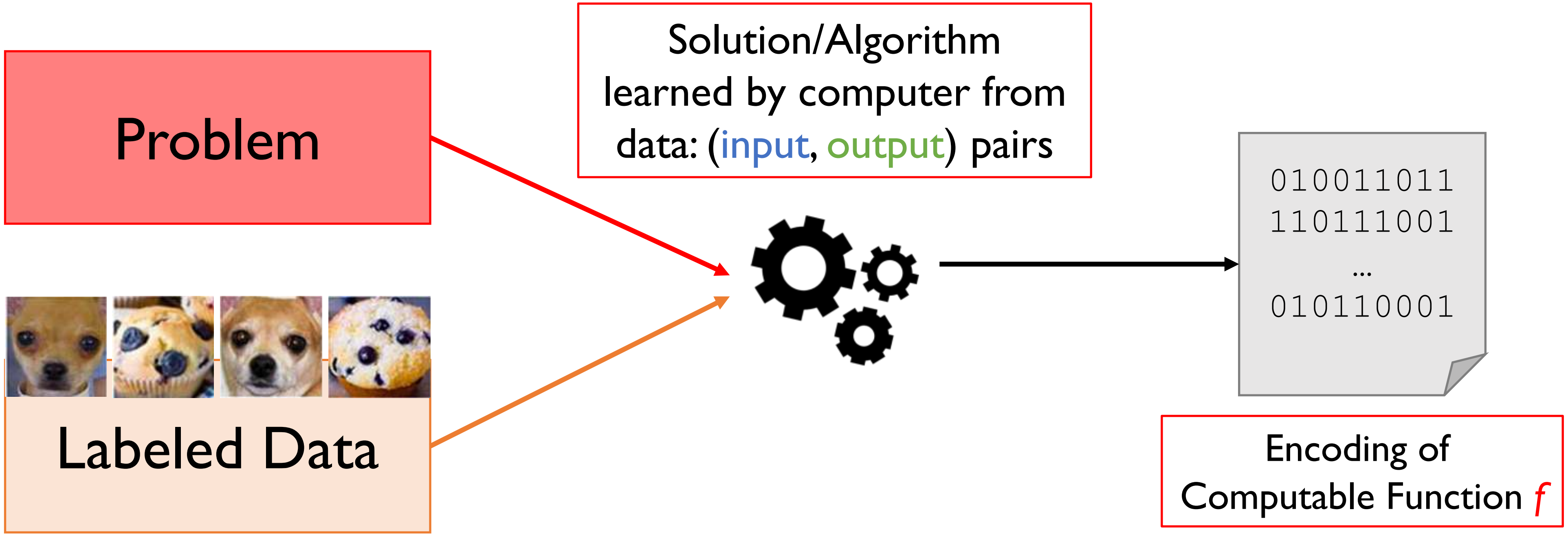
"Training" a Computer

Problem

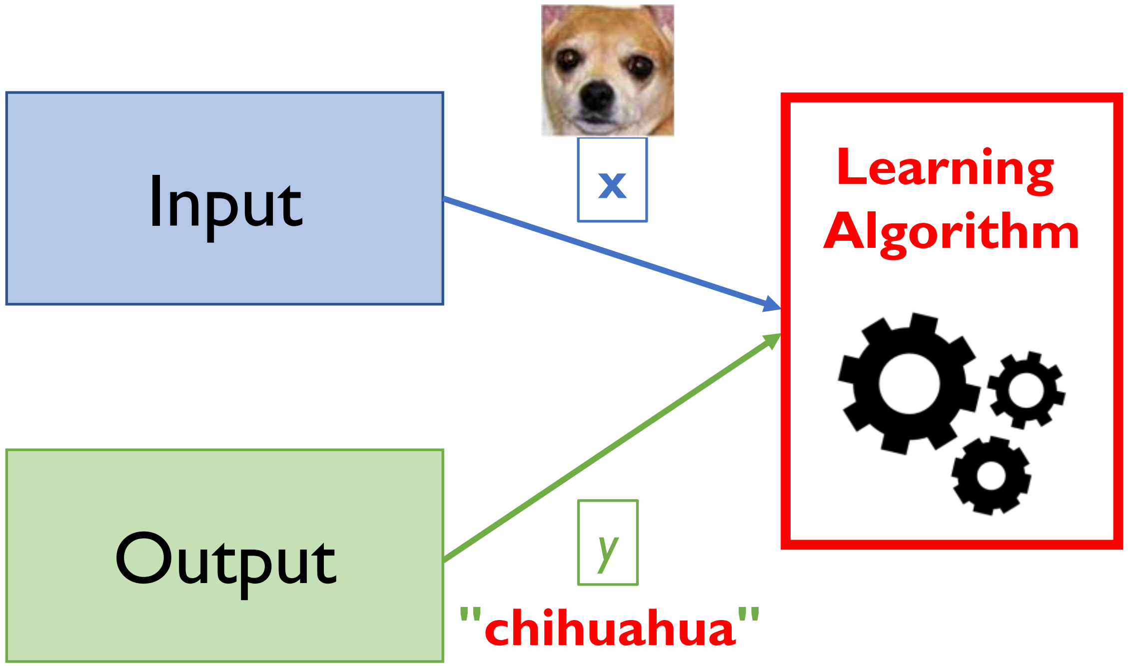
Solution/Algorithm learned by computer from data: (input, output) pairs

Labeled Data

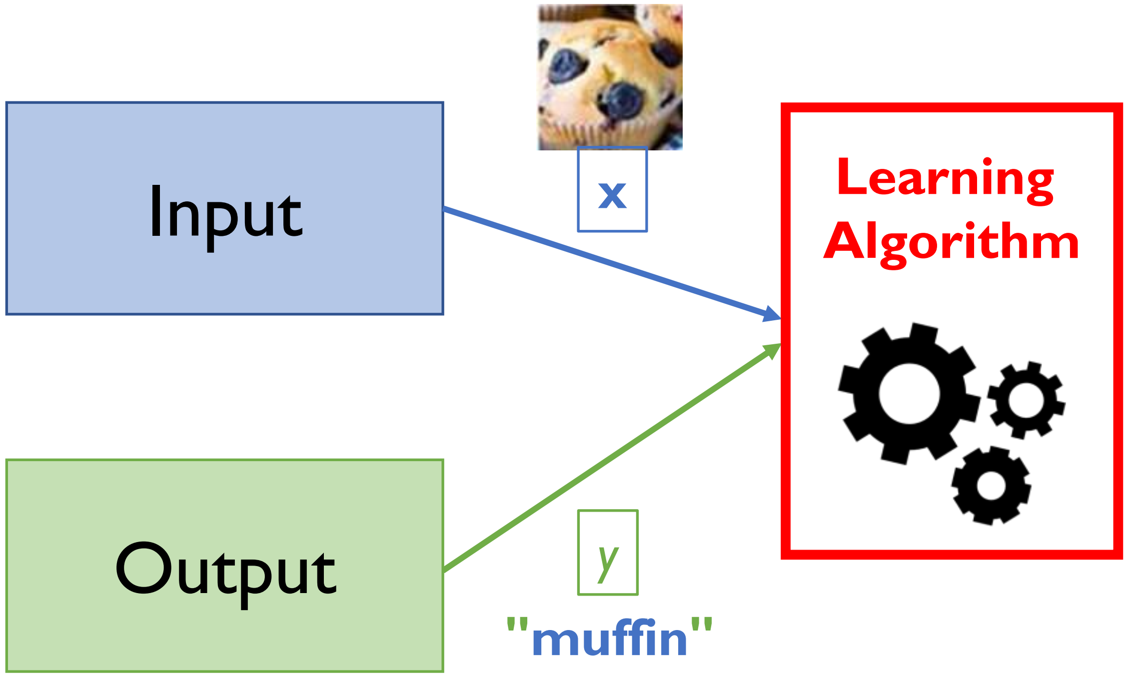
"Training" a Computer



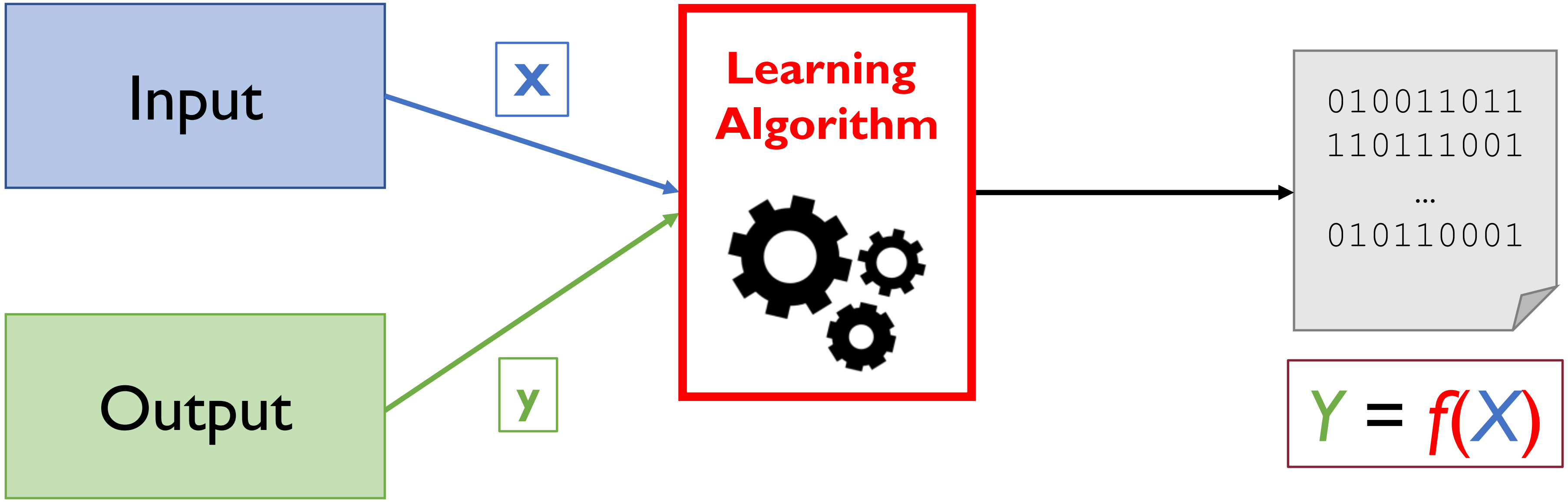
"Training" a Computer



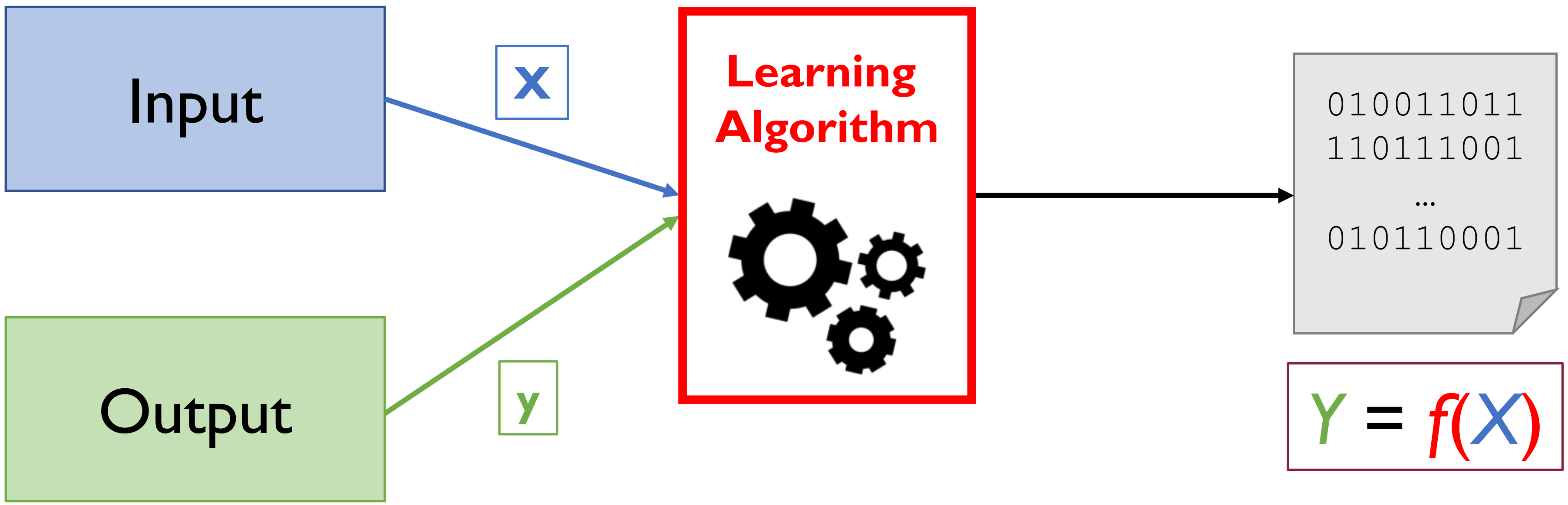
"Training" a Computer



"Training" a Computer



"Training" a Computer



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

Machine Learning

- A broad discipline concerned with how to teach machines to learn (i.e., extract knowledge) from data

Machine Learning

- A broad discipline concerned with how to teach machines to learn (i.e., extract knowledge) from data
- 2 main definitions of it:

Machine Learning

- A broad discipline concerned with how to teach machines to learn (i.e., extract knowledge) from data
- 2 main definitions of it:

"The field of study that gives computers the ability to learn without being explicitly programmed"

Arthur Samuel

Machine Learning

- A broad discipline concerned with how to teach machines to learn (i.e., extract knowledge) from data
- 2 main definitions of it:

"The field of study that gives computers the ability to learn without being explicitly programmed"

Arthur Samuel

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E"

Tom Mitchell

Machine Learning: Taxonomy

Machine Learning

Machine Learning: Taxonomy

Machine Learning

Unsupervised Learning

Extract patterns from input data without any information on the output (target) variable

Machine Learning: Taxonomy

Machine Learning

Unsupervised Learning

Extract patterns from input data without any information on the output (target) variable

Supervised Learning

Approximate a function $f: X \rightarrow Y$ from a set of observed labeled examples $\{(\mathbf{X}, \mathbf{y})\}$

Machine Learning: Taxonomy

Machine Learning

Unsupervised Learning

Extract patterns from input data without any information on the output (target) variable

Supervised Learning

Approximate a function $f: X \rightarrow Y$ from a set of observed labeled examples $\{(\mathbf{X}, \mathbf{y})\}$

Reinforcement Learning

Use a Reward-Feedback loop to continuously learn and update the hidden behavior or pattern

Machine Learning: Taxonomy

Machine Learning

Unsupervised Learning

Extract patterns from input data without any information on the output (target) variable

Supervised Learning

Approximate a function $f: X \rightarrow Y$ from a set of observed labeled examples $\{(\mathbf{X}, \mathbf{y})\}$

Reinforcement Learning

Use a Reward-Feedback loop to continuously learn and update the hidden behavior or pattern

Supervised Learning: What Do We Predict?

$\boxed{\textbf{Supervised Learning}}$

Supervised Learning: What Do We Predict?

Supervised Learning

Regression

The target y we want to predict is a **continuous real value**
e.g., y = price of a house

Supervised Learning: What Do We Predict?

Supervised Learning

Regression

The target y we want to predict is a **continuous real value**
e.g., y = price of a house

Classification

The target y we want to predict is a **discrete value**
e.g., y = spam/non-spam

The Supervised Learning Pipeline

The Stages of Supervised Learning

0. Be sure your problem needs <u>actually</u> to be tackled using Machine Learning techniques

(i.e. there is no point in adopting any fancy ML solution if it can be solved "directly"!)

The Stages of Supervised Learning

0. Be sure your problem needs actually to be tackled using Machine Learning techniques
(i.e. there is no point in adopting any fancy ML solution if it can be solved "directly"!)

1. Data collection: get data from your domain of interest

The Stages of Supervised Learning

0. Be sure your problem needs actually to be tackled using Machine Learning techniques
(i.e. there is no point in adopting any fancy ML solution if it can be solved "directly"!)

1. Data collection: get data from your domain of interest

2. Feature engineering: represent data in a "machine-friendly" format

The Stages of Supervised Learning

0. Be sure your problem needs actually to be tackled using Machine Learning techniques
(i.e. there is no point in adopting any fancy ML solution if it can be solved "directly"!)

1. Data collection: get data from your domain of interest

2. Feature engineering: represent data in a "machine-friendly" format

3. Model training: "build" one (or more) learning models

The Stages of Supervised Learning

0. Be sure your problem needs actually to be tackled using Machine Learning techniques
(i.e. there is no point in adopting any fancy ML solution if it can be solved "directly"!)

1. Data collection: get data from your domain of interest

2. Feature engineering: represent data in a "machine-friendly" format

3. Model training: "build" one (or more) learning models

4. Model selection/evaluation: pick the best-performing model according to some quality metrics

Data Collection

- Any ML technique needs data to operate on!

Data Collection

- Any ML technique needs data to operate on!
- Supervised Learning requires **labeled data** which may be even harder to get
  - e.g., emails + spam/non-spam tags

Data Collection

- Any ML technique needs data to operate on!
- Supervised Learning requires **labeled data** which may be even harder to get
  - e.g., emails + spam/non-spam tags
- Might involve combining multiple and heterogeneous data sources

Feature Engineering

Domain Objects

Feature Engineering

Collected data need to be encoded with a machine-readable format

Domain Objects

Feature Engineering

Collected data need to be encoded with a machine-readable format

Domain Objects

i-th object

0.27	3.8	1.12	…	4.75	…	…	56.9
0.0	3.65	0.04	…	6.81	…	…	1.0

…

6.94	5.4	0.9	…	3.92	…	…	8.2

Each domain object is translated into a n-dimensional vector of features

Feature Engineering

Collected data need to be encoded with a machine-readable format

Domain Objects

i-th object

0.27	3.8	1.12	…	4.75	…	…	56.9
0.0	3.65	0.04	…	6.81	…	…	1.0
				…			
6.94	5.4	0.9	…	3.92	…	…	8.2

j-th feature

Each domain object is translated into a *n*-dimensional vector of features

Feature Engineering

- Each feature is a **property** of an instance of our domain
  - e.g., **`number_of_bedrooms`** in the case our domain objects are "houses"

Feature Engineering

- Each feature is a **property** of an instance of our domain
  - e.g., **`number_of_bedrooms`** in the case our domain objects are "houses"
- Each feature can be either derived locally from an instance
  - e.g., **`annual_income`** of a person

Feature Engineering

- Each feature is a **property** of an instance of our domain
  - e.g., **`number_of_bedrooms`** in the case our domain objects are "houses"
- Each feature can be either derived locally from an instance
  - e.g., **`annual_income`** of a person
- Or it can be the result of more complex computation involving the whole data collection
  - e.g., **`tf-idf`** of a word of a document w.r.t. a corpus

Feature Engineering

- Traditionally done manually by human experts

Feature Engineering

- Traditionally done manually by human experts
- Require in-depth knowledge of the specific domain of application
  - e.g., text, images, finance, etc.

Feature Engineering

- Traditionally done manually by human experts
- Require in-depth knowledge of the specific domain of application
  - e.g., text, images, finance, etc.
- Tedious and time-consuming process

Feature Engineering

- Traditionally done manually by human experts
- Require in-depth knowledge of the specific domain of application
  - e.g., text, images, finance, etc.
- Tedious and time-consuming process
- Techniques to **automatically** learn data representation (i.e., features):
  - K-means clustering or PCA (**unsupervised**)
  - Deep Neural Network (**supervised**)

Feature Engineering: Challenges and Solutions

Collected (raw) data is far from being perfect!

Feature Engineering: Challenges and Solutions

Collected (raw) data is far from being perfect!

Many challenges need to be addressed before taking any further step down to the ML pipeline

Feature Engineering: Challenges and Solutions

Collected (raw) data is far from being perfect!

Many challenges need to be addressed before taking any further step down to the ML pipeline

Data Preprocessing

Feature Engineering: Challenges and Solutions

Challenge	Description	
Missing values	A feature value may not be available for one or more instances	

Feature Engineering: Challenges and Solutions

Challenge	**Description**	**Solution**
Missing values	A feature value may not be available for one or more instances	Replace missing values with the **median** (*continuous*) or the **mode** (*categorical*) of the existing values

Feature Engineering: Challenges and Solutions

Challenge	Description	
Sparsity	Most of the instances contain just a small subset of the features	

Feature Engineering: Challenges and Solutions

Challenge	Description	Solution
Sparsity	Most of the instances contain just a small subset of the features	Use "sparse-friendly" data structures (e.g., DOK)

Feature Engineering: Challenges and Solutions

Challenge	Description	
Outliers	One or more instances have out-of-range values for one or more features	

Feature Engineering: Challenges and Solutions

Challenge	**Description**	**Solution**
Outliers	One or more instances have out-of-range values for one or more features	Retention vs. Exclusion (*trimming* or *winsorising*)

Feature Engineering: Challenges and Solutions

Challenge	Description	
Mix of continuous and discrete values	Feature set contains both numerical and categorical values	

Feature Engineering: Challenges and Solutions

Challenge	Description	Solution
Mix of continuous and discrete values	Feature set contains both numerical and categorical values	Transform categorical features using *one-hot encoding*

Feature Engineering: Challenges and Solutions

Challenge	**Description**	
Multiple feature magnitudes	Feature set contains very wide range of values	

Feature Engineering: Challenges and Solutions

Challenge	Description	Solution
Multiple feature magnitudes	Feature set contains very wide range of values	Standardization (min-max, z-scores)

Feature Engineering: Challenges and Solutions

Challenge	Description	
Class imbalance	Instances labeled with the class of interest represents a tiny fraction of the total	

Feature Engineering: Challenges and Solutions

Challenge	Description	Solution
Class imbalance	Instances labeled with the class of interest represents a tiny fraction of the total	Over-/Under-sampling, cost-sensitive learning

Feature Engineering: Challenges and Solutions

Challenge	Description	
Strong multicollinearity	Linear relationship between one or more features	

Feature Engineering: Challenges and Solutions

Challenge	Description	Solution
Strong multicollinearity	Linear relationship between one or more features	Dimensionality reduction (PCA)

A Bit of Notation

$\mathcal{X} \subseteq \mathbb{R}^n$ input feature space

A Bit of Notation

$\mathcal{X} \subseteq \mathbb{R}^n$ input feature space

$\mathcal{Y}$ output space

A Bit of Notation

$\mathcal{X} \subseteq \mathbb{R}^n$ input feature space

$\mathcal{Y}$ output space

$\mathcal{Y} \in \mathbb{R}$ real-value label of the *i*-th instance (regression)

$\mathcal{Y} \in \{1, \ldots, k\}$ discrete-value label of the *i*-th instance (k-ary classification)

A Bit of Notation

$\mathcal{X} \subseteq \mathbb{R}^n$	input feature space
$\mathcal{Y}$	output space
$\mathcal{Y} \in \mathbb{R}$	real-value label of the *i*-th instance (regression)
$\mathcal{Y} \in \{1, \ldots, k\}$	discrete-value label of the *i*-th instance (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

$\mathcal{Y} \in \mathbb{R}$ real-value label of the *i*-th instance (regression)

$\mathcal{Y} \in \{1, \ldots, k\}$ discrete-value label of the *i*-th instance (k-ary classification)

$(\mathbf{x}_i, y_i)$ *i*-th labeled instance

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

$\mathcal{Y} \in \mathbb{R}$ real-value label of the *i*-th instance (regression)

$\mathcal{Y} \in \{1, \dots, k\}$ discrete-value label of the *i*-th instance (k-ary classification)

$(\mathbf{x}_i, y_i)$ *i*-th labeled instance

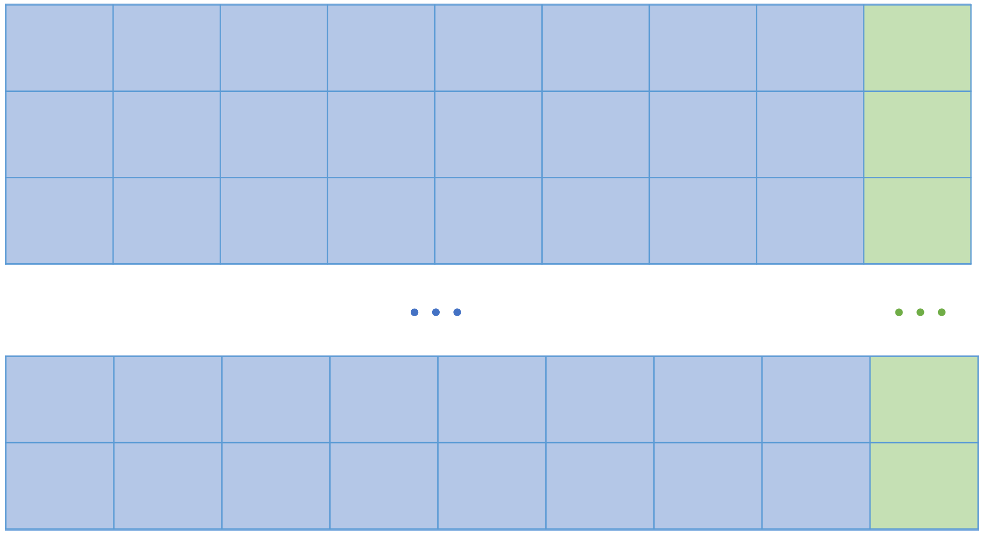
$\mathbf{x}_i = (x_{i,1}, \dots, x_{i,n}) \in \mathcal{X}$ *n*-dimensional feature vector of the *i*-th instance

$y_i \in \mathcal{Y}$ label of the *i*-th instance

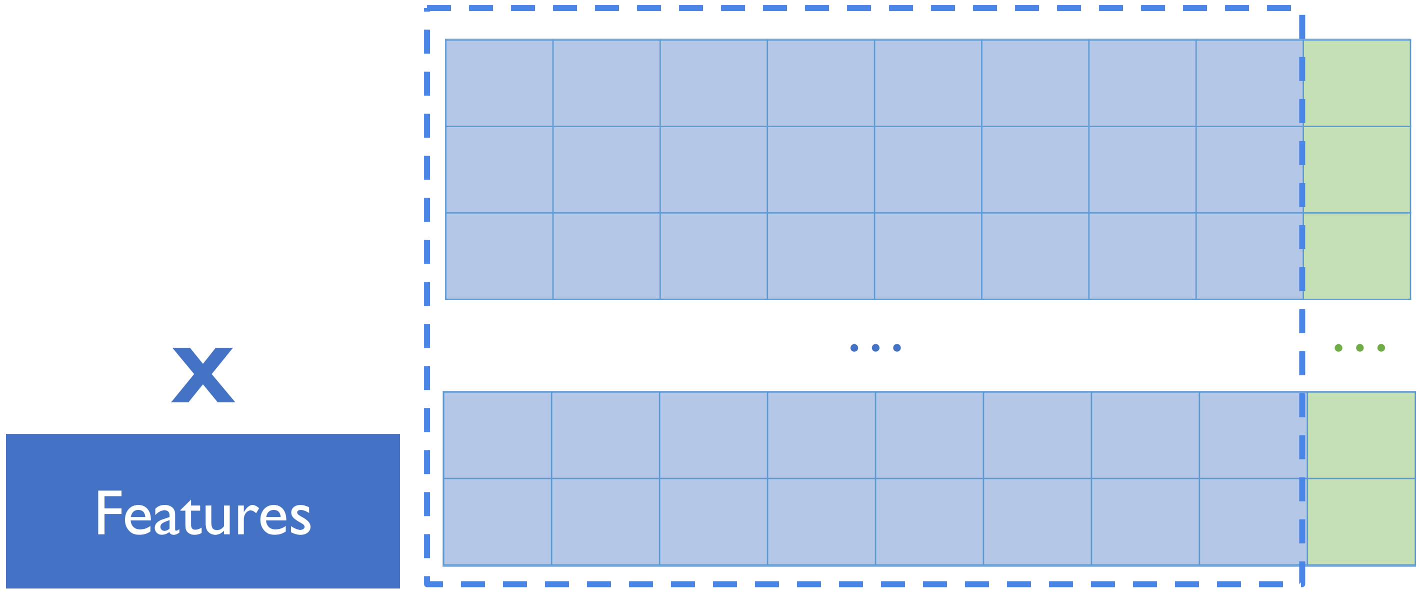
A Bit of Notation

$\mathcal{X} \subseteq \mathbb{R}^n$	input feature space
$\mathcal{Y}$	output space
$\mathcal{Y} \in \mathbb{R}$	real-value label of the *i*-th instance (regression)
$\mathcal{Y} \in \{1, \ldots, k\}$	discrete-value label of the *i*-th instance (k-ary classification)
$(\mathbf{x}_i, y_i)$	*i*-th labeled instance
$\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,n}) \in \mathcal{X}$	*n*-dimensional feature vector of the *i*-th instance
$y_i \in \mathcal{Y}$	label of the *i*-th instance
$\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$	dataset of *m* i.i.d. labeled instances

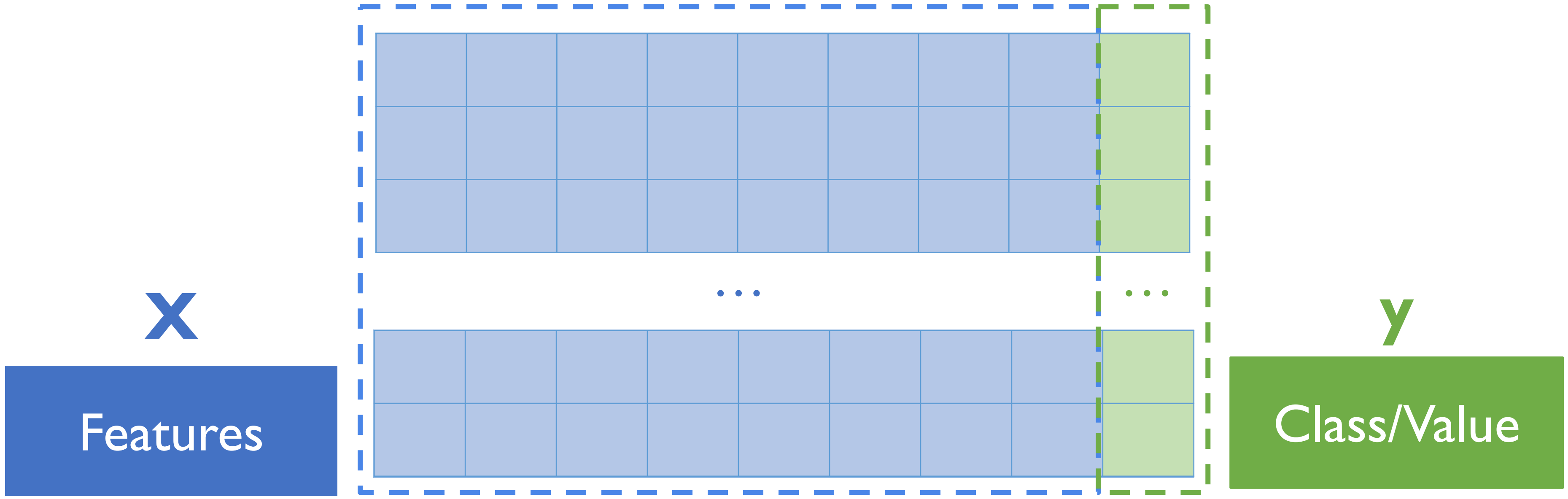
Model Training: Labeled Dataset



Model Training: Labeled Dataset

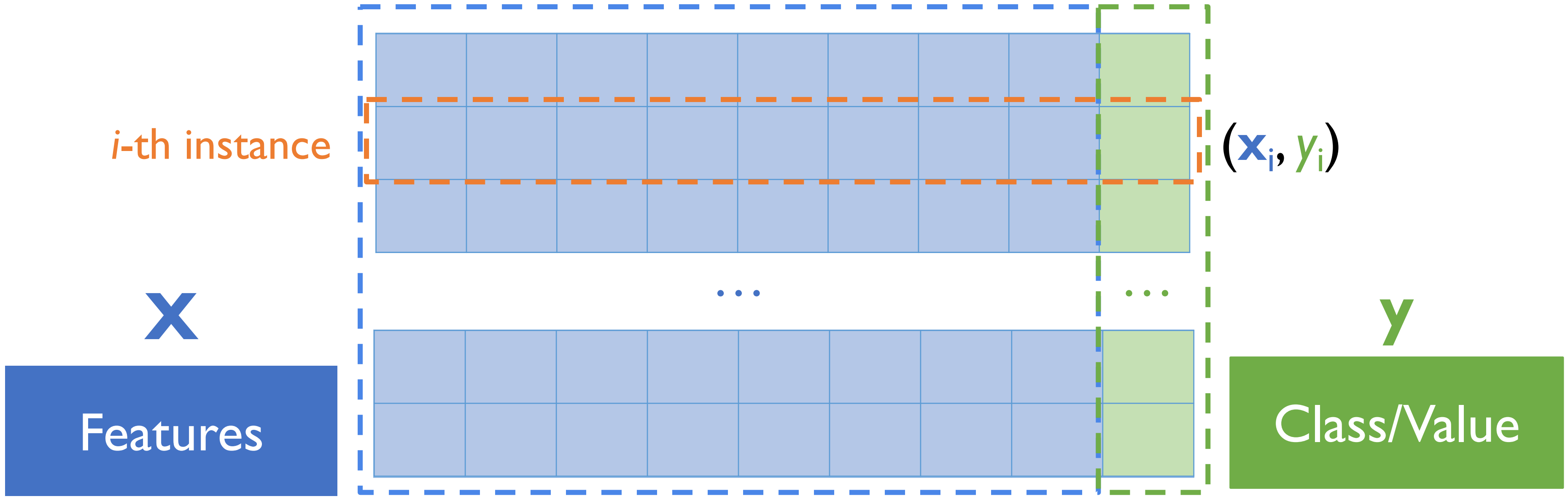


Model Training: Labeled Dataset



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*

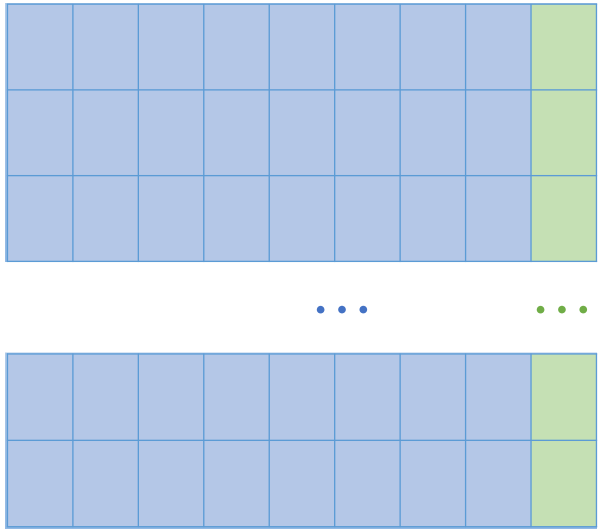
$$\boxed{h^* = \text{argmin}_{h \in \mathcal{H}} L(h, \mathcal{D})}$$

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

$$\boxed{f = X \rightarrow Y}$$

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

$f = X \rightarrow Y$

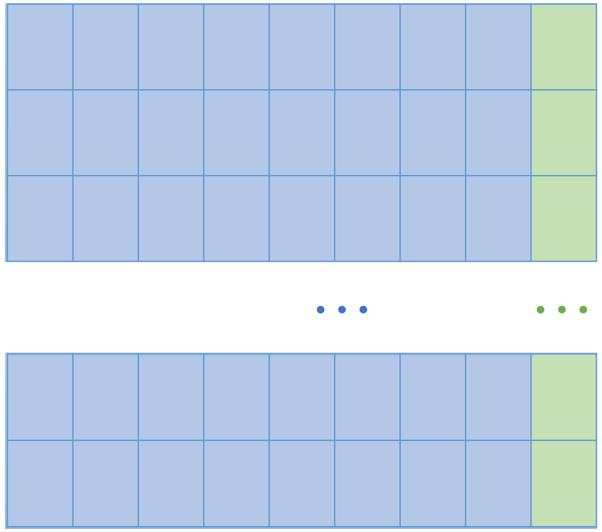


$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)$

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

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

$\boxed{f = X \rightarrow Y}$



$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)$

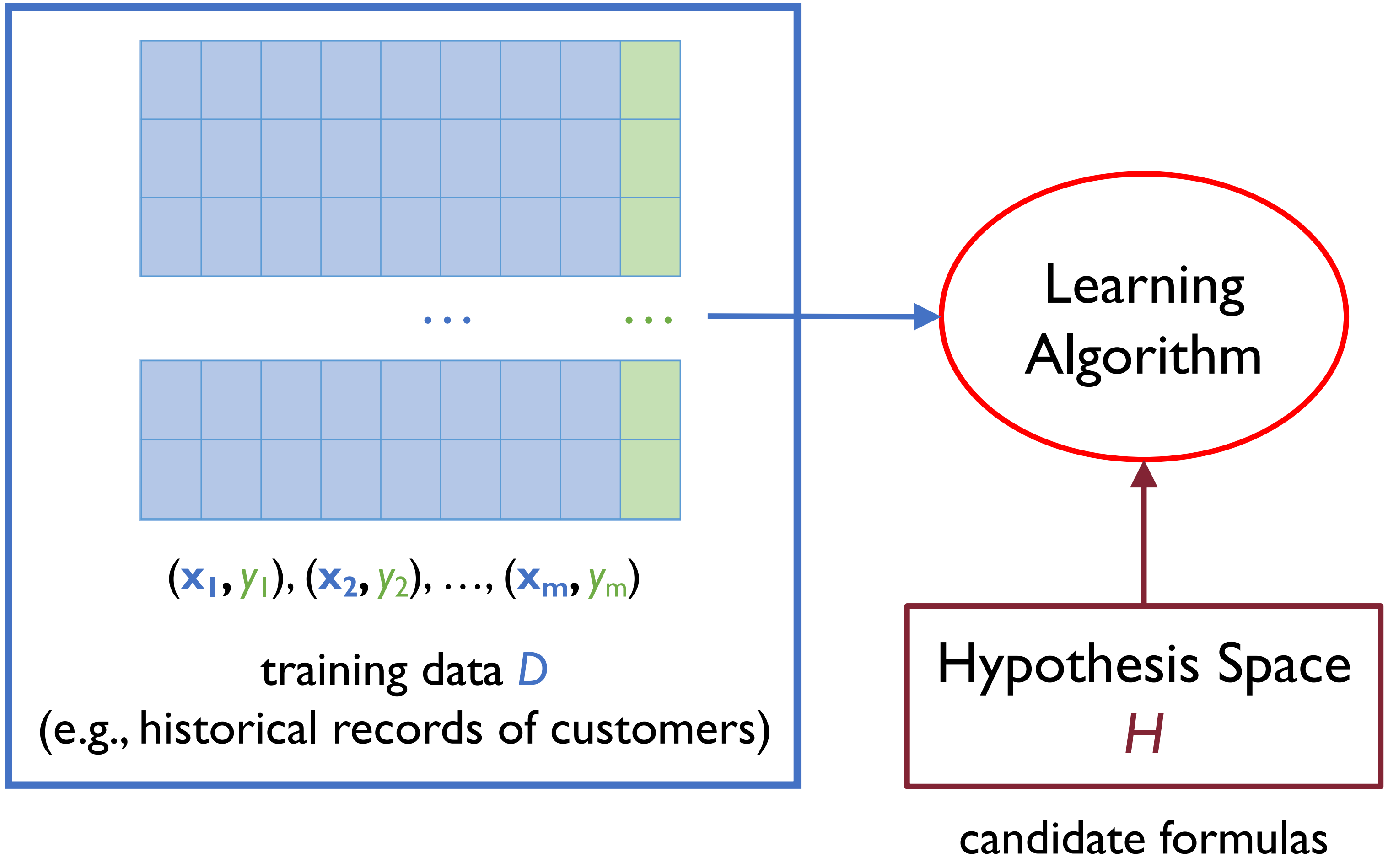
training data D
(e.g., historical records of customers)

Hypothesis Space
H

candidate formulas

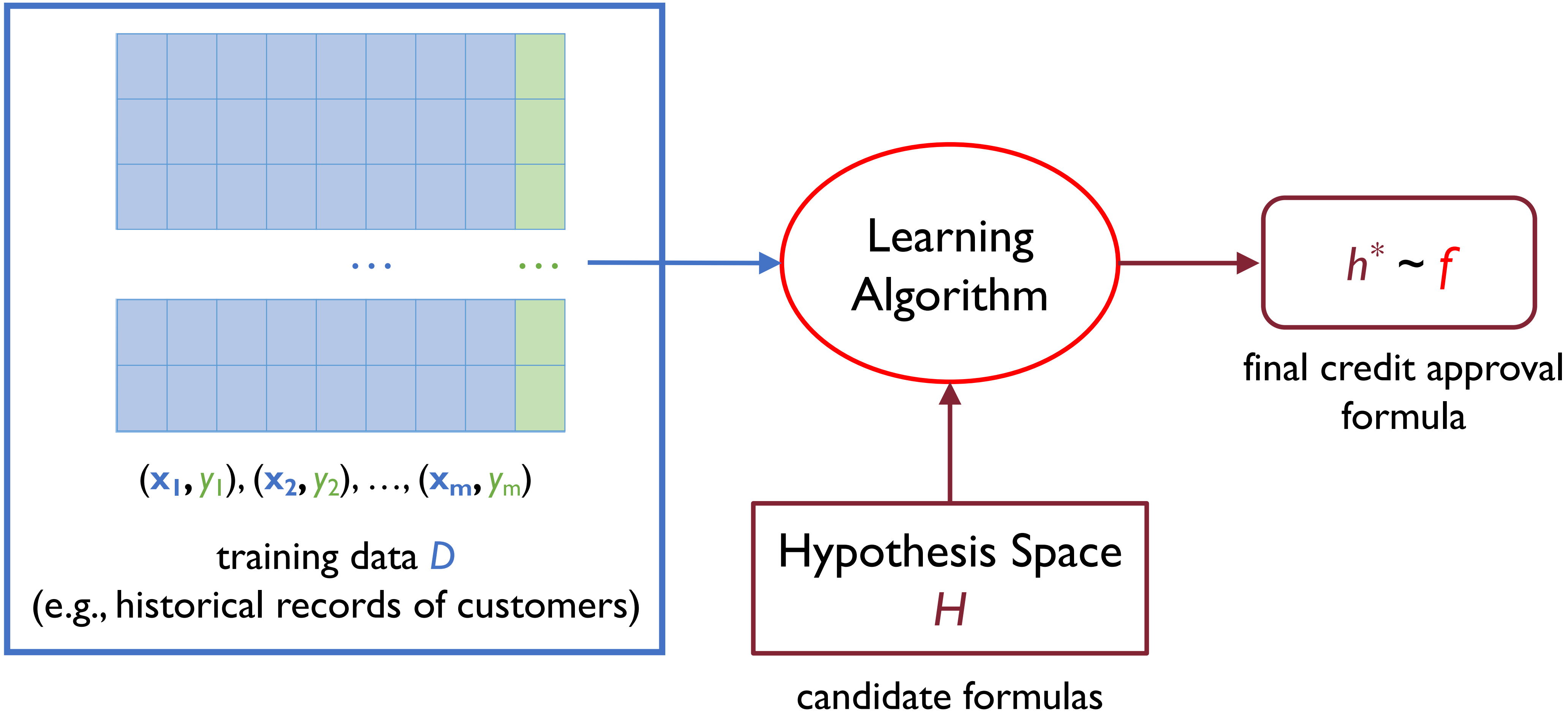
unknown target
(e.g., ideal credit approval function)

$$\boxed{f = X \rightarrow Y}$$



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

$$\boxed{f = X \rightarrow Y}$$



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 <u>this is not learning</u>!

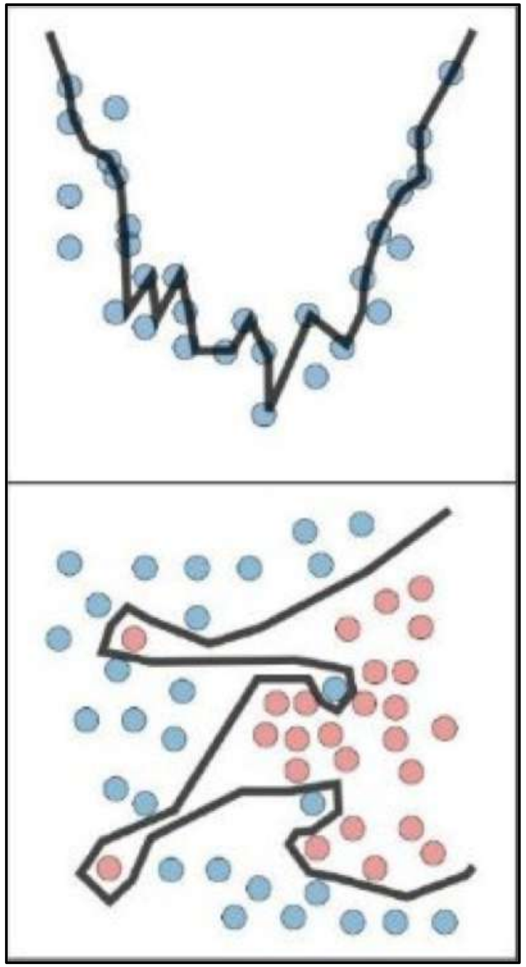
In-Sample vs. Out-of-Sample Error

- 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 <u>this is not learning</u>!
- At the same time we do not want h^* to perform poorly on D

Overfitting (High Variance)

Regression

Classification

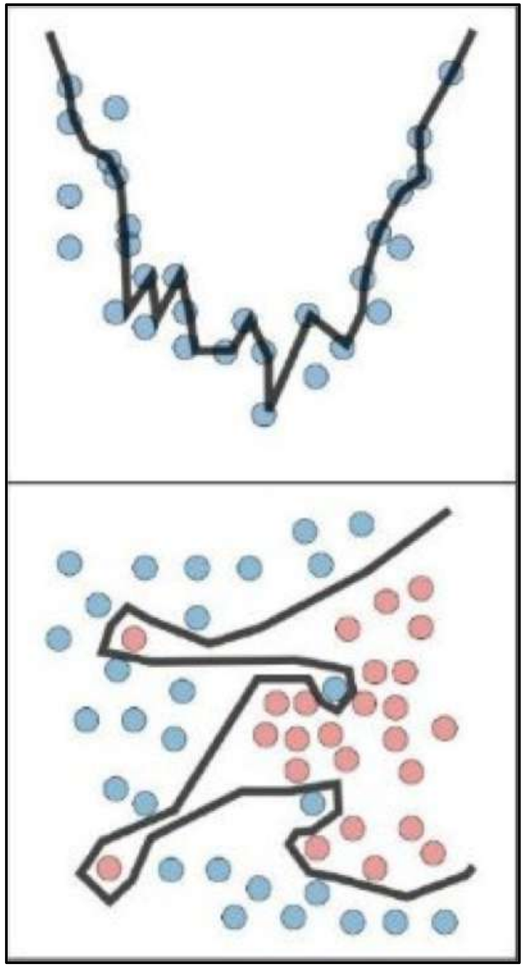


The hypothesis h^* is not learning the true f but it mimics its noise

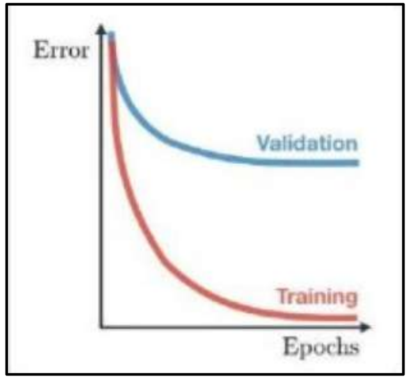
Overfitting (High Variance)

Regression

Classification



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

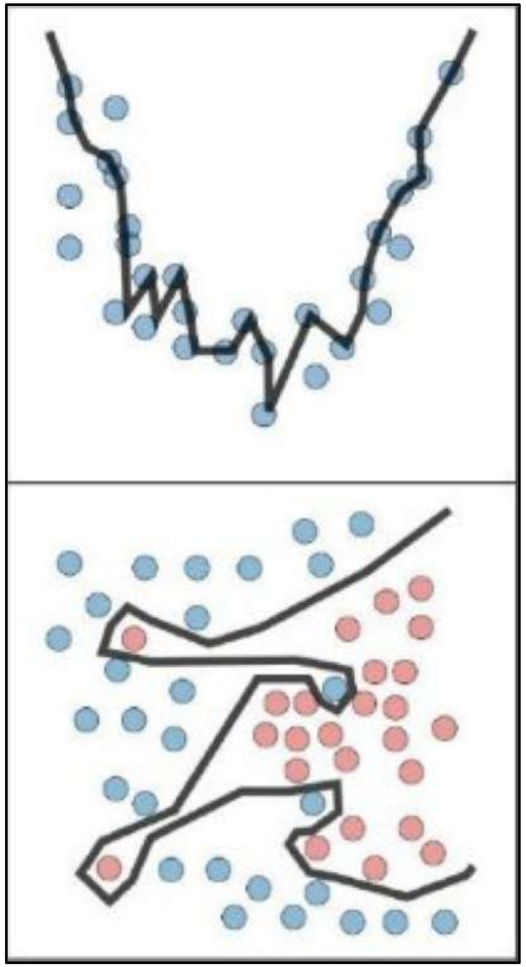


low in-sample error high out-of-sample error

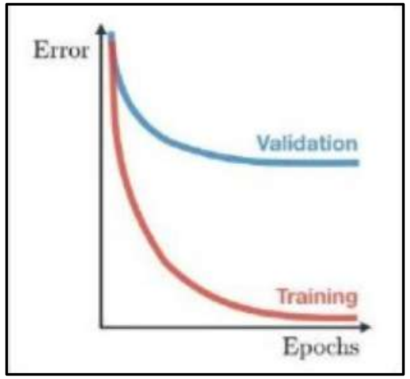
Overfitting (High Variance)

Regression

Classification



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

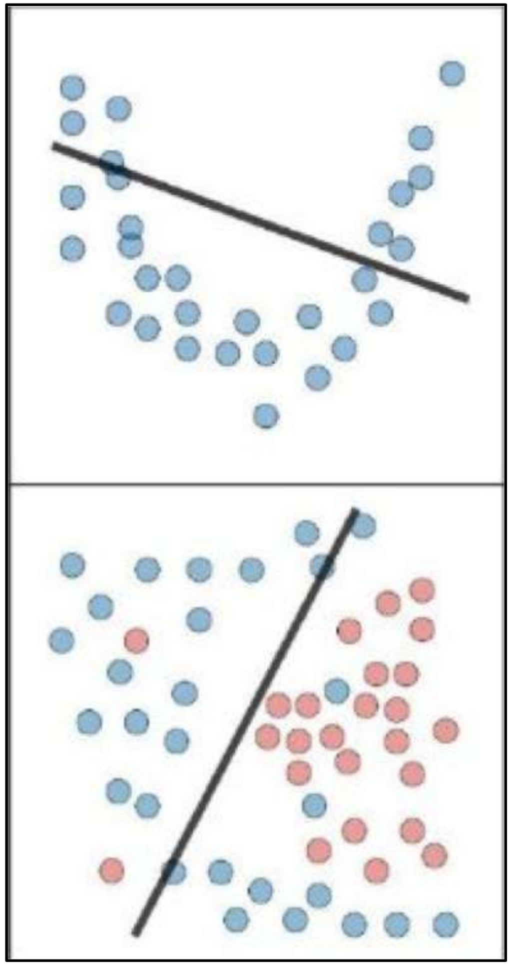


low in-sample error high out-of-sample error

- Regularization
- Get more data

Underfitting (High Bias)

Regression



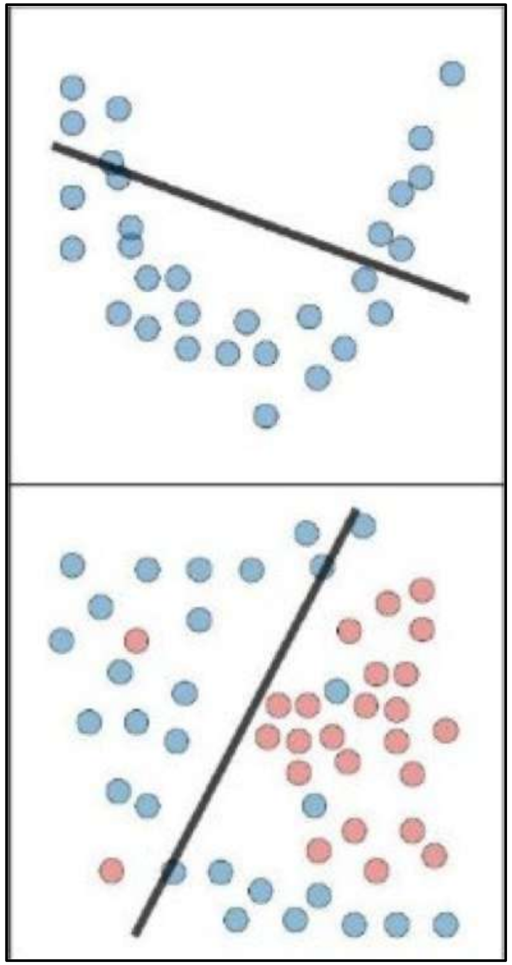
Classification

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

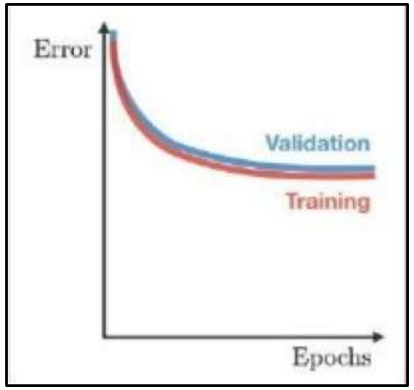
Underfitting (High Bias)

Regression

Classification



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

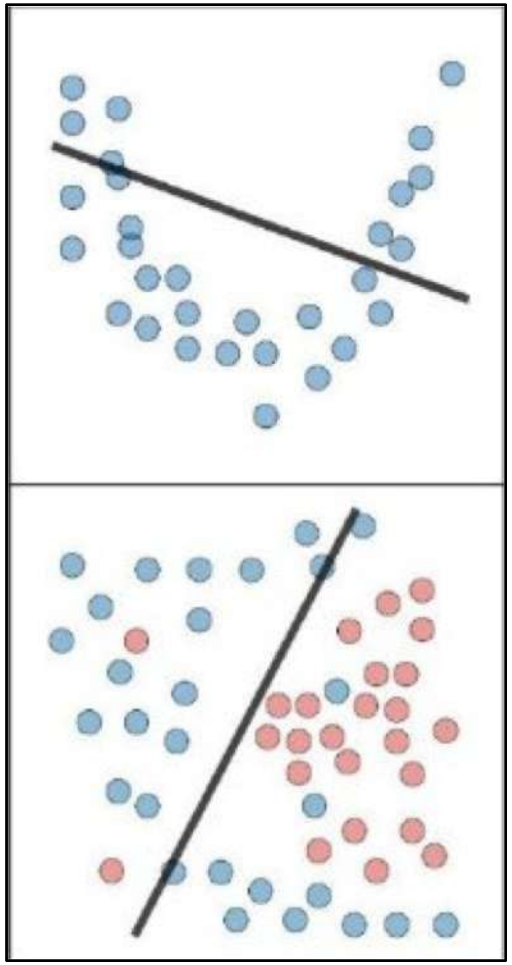


high in-sample error high out-of-sample error

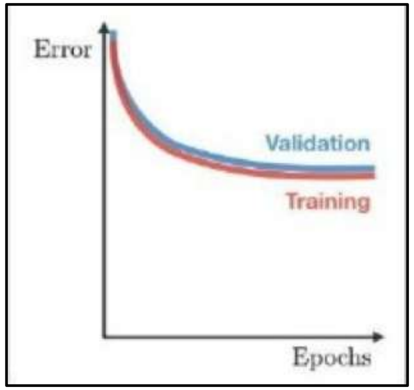
Underfitting (High Bias)

Regression

Classification



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



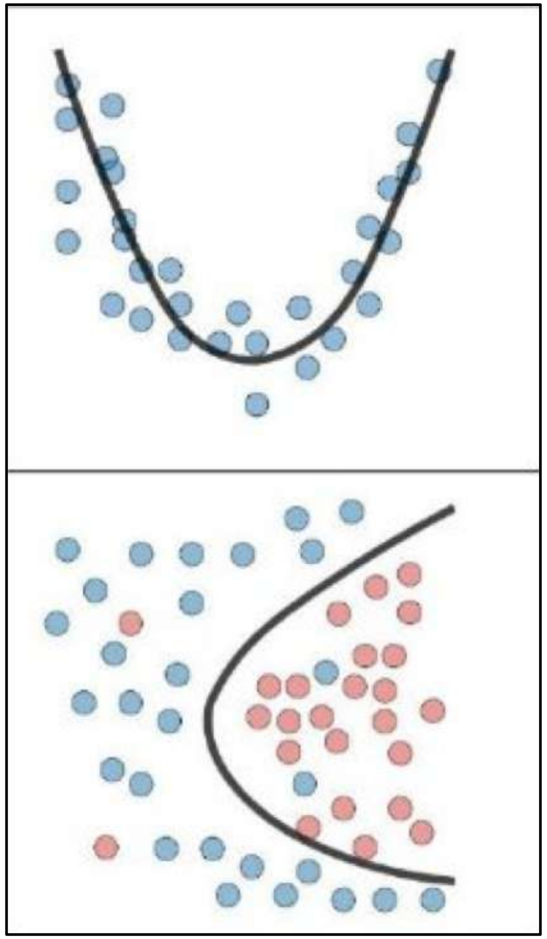
high in-sample error high out-of-sample error

- Increase model complexity
- Add more features

Bias-Variance Tradeoff

Regression

Classification

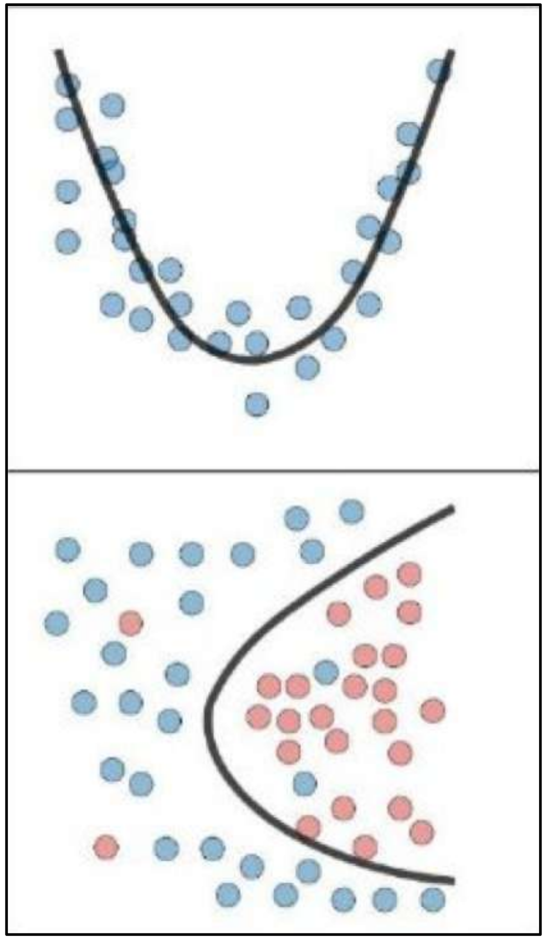


The hypothesis h^* is just right:
the simplest one explaining the data

Occam's razor

Bias-Variance Tradeoff

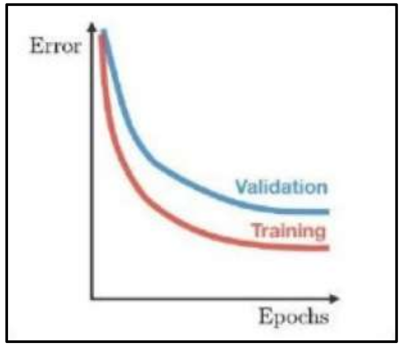
Regression



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
  - e.g., don't want to deploy your new spam classifier in production knowing only its training (i.e., in-sample) performance

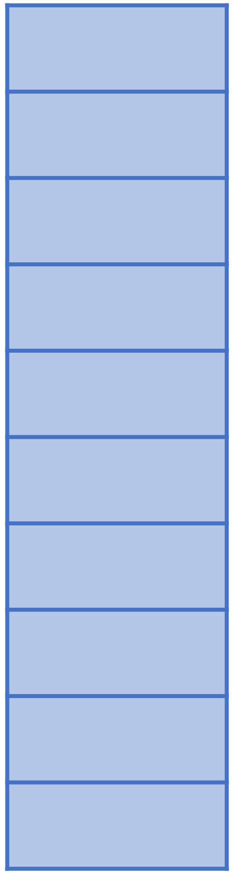
Estimating Generalization Performance

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

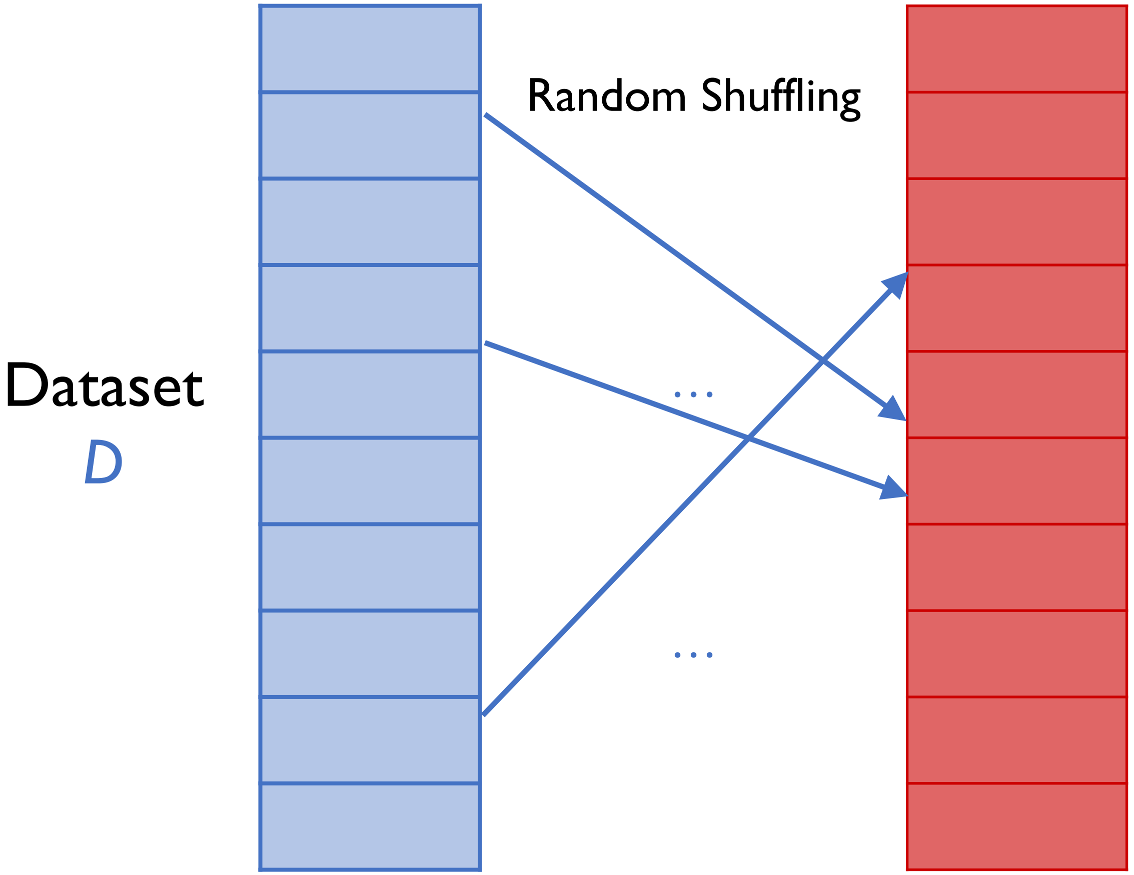
Dataset Splitting

Dataset

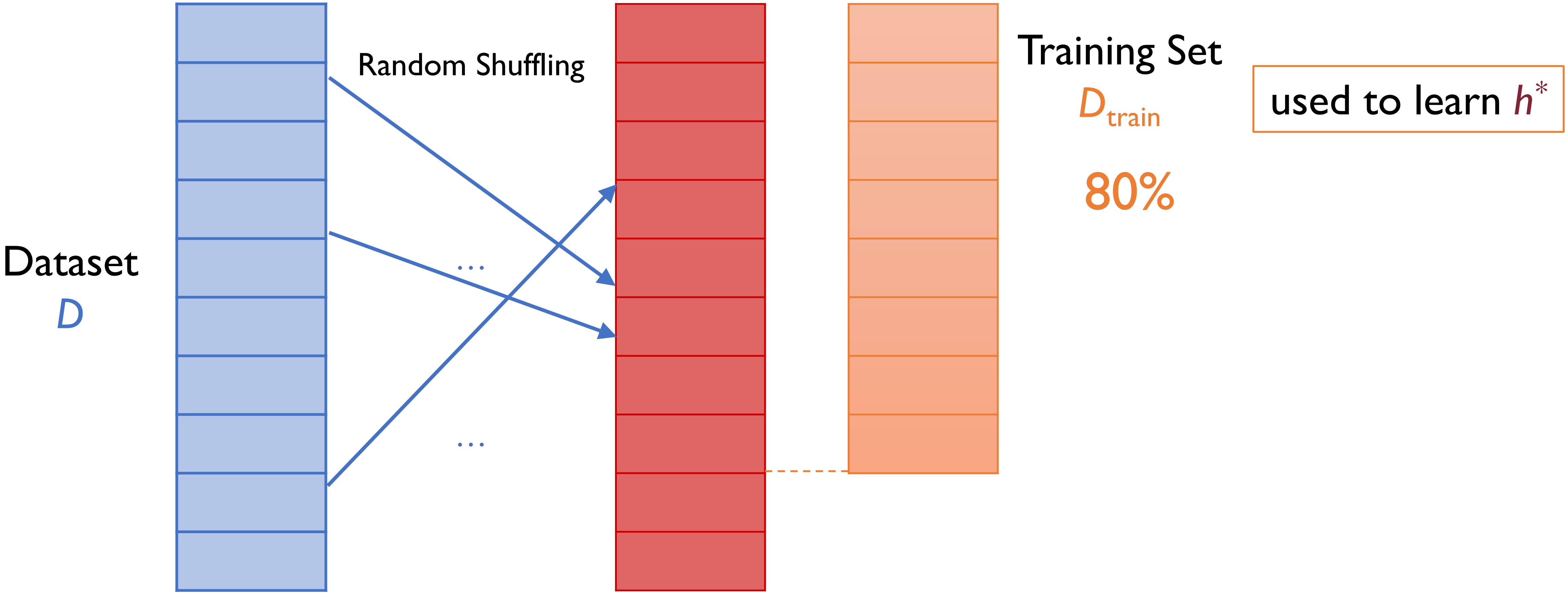
D



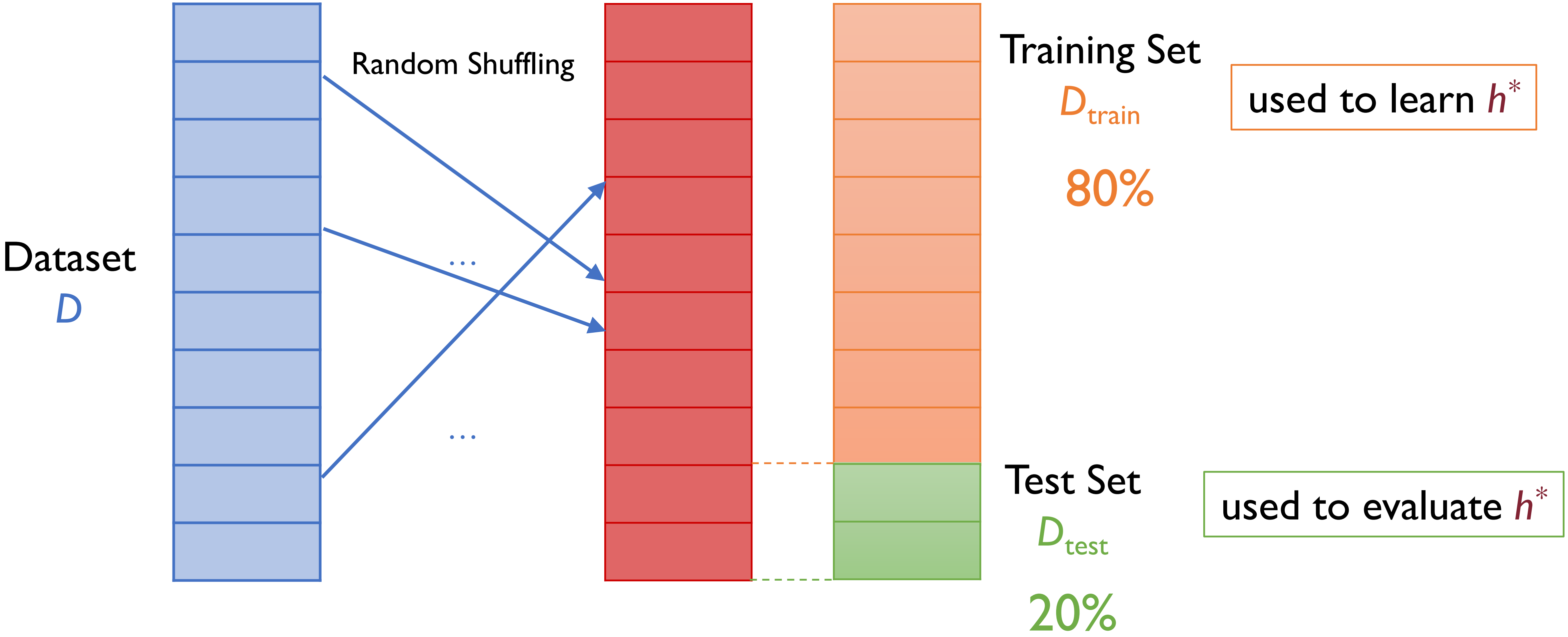
Dataset Splitting



Dataset Splitting

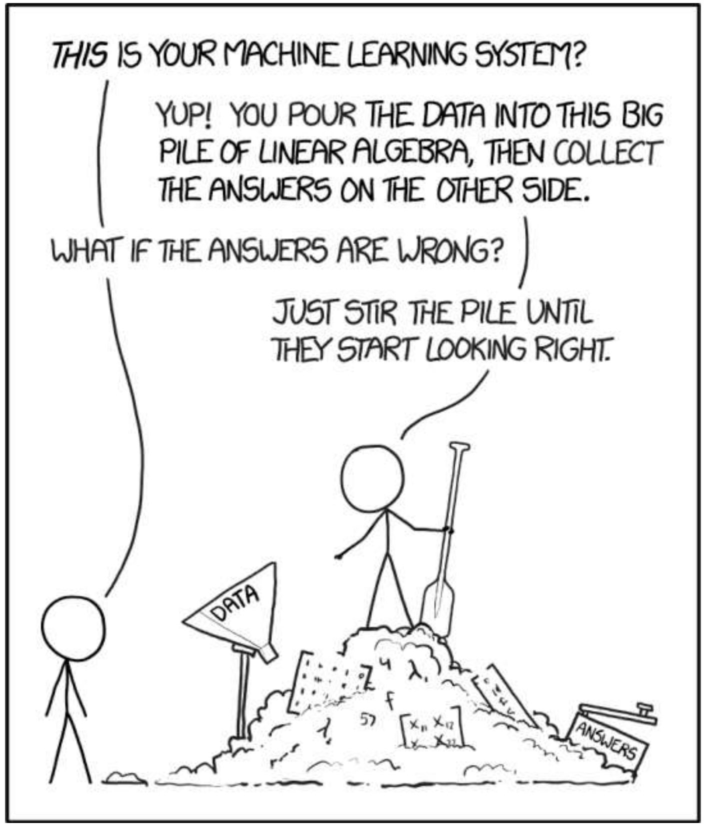


Dataset Splitting



How Much Data Do We Need?

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



source: https://xkcd.com/1838/

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

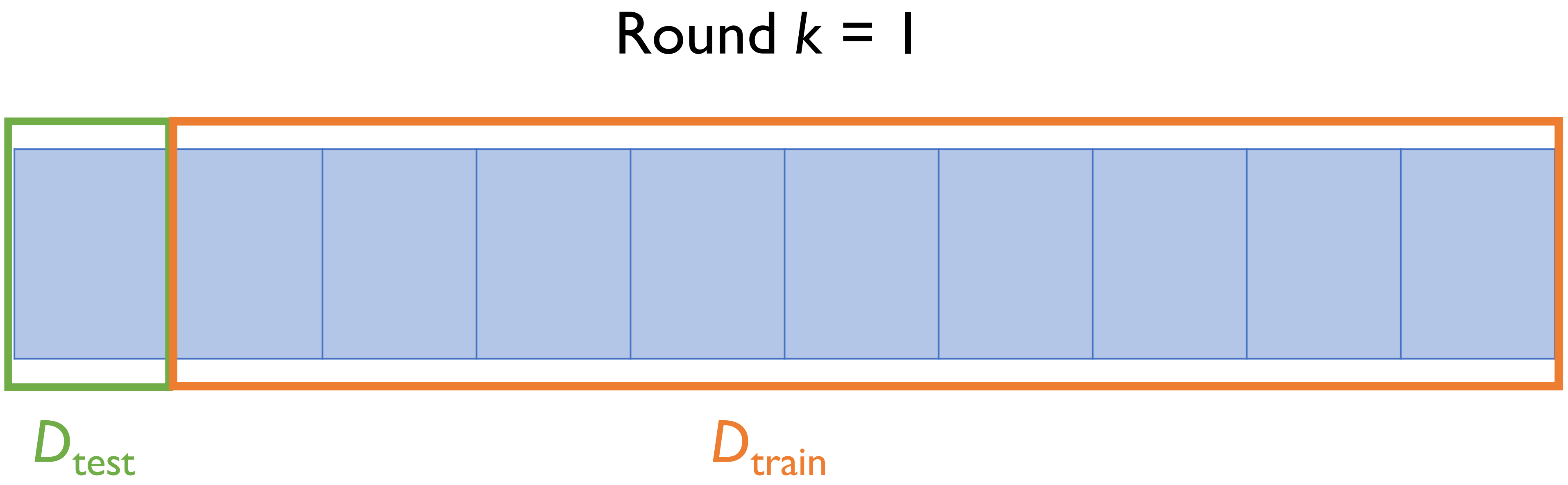
K-Fold Cross Validation

- 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:
  - leaned from K-1 training folds
  - evaluated on 1 remaining test fold

K-Fold Cross Validation

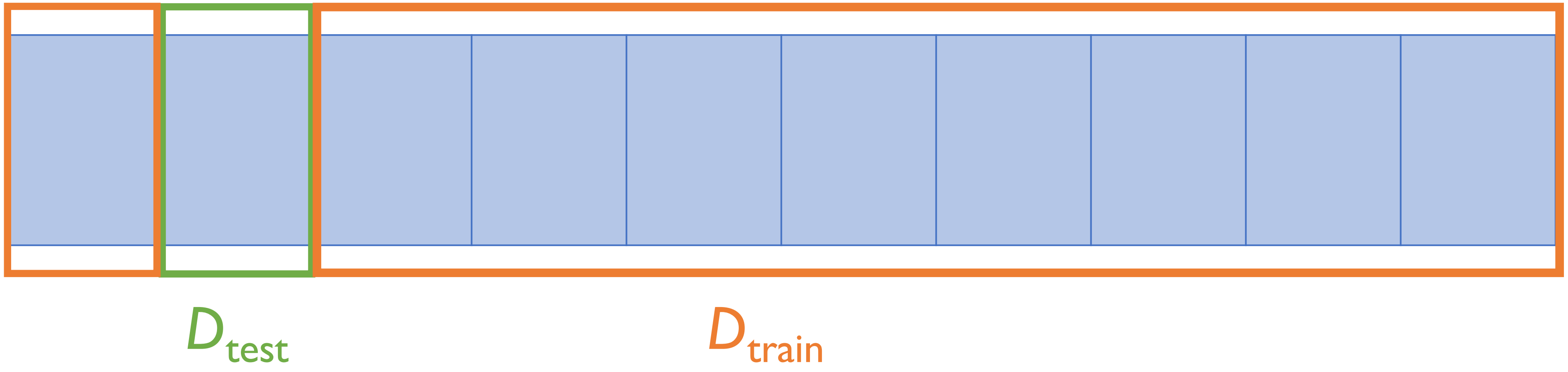
- A generalization of the training/test splitting seen before
- Pick a value for K (e.g., K=5 or 10)
- Divide your dataset D into K distinct folds
- Perform K rounds where h^* is:
  - leaned from K-1 training folds
  - evaluated on 1 remaining test fold
- The estimate of generalization error is the average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

K-Fold Cross Validation



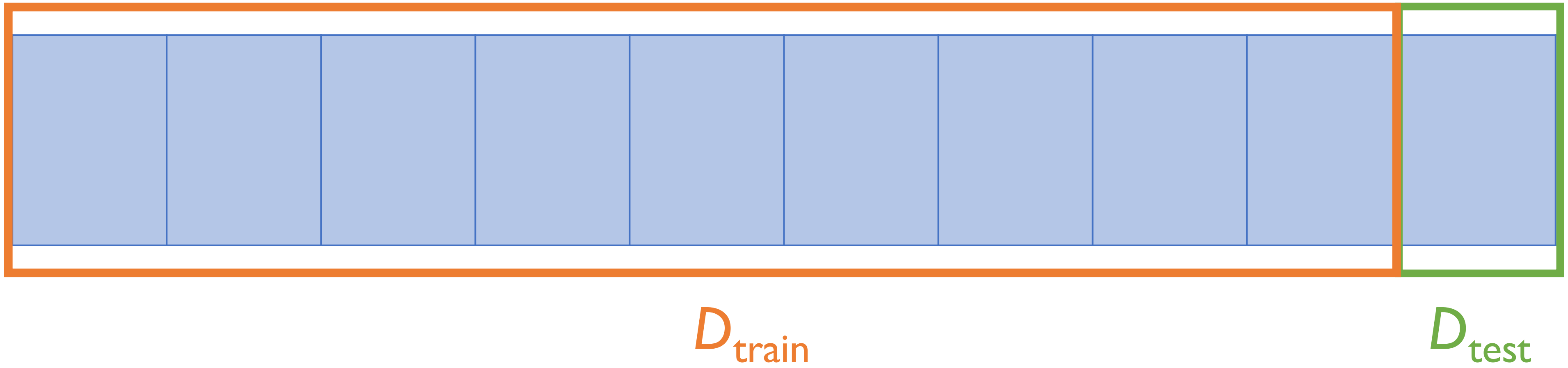
K-Fold Cross Validation

Round k = 2



K-Fold Cross Validation

Round k = 10



Model Selection/Evaluation

Several different learning models to achieve the same task

Decision Trees

Random Forests

Logistic Regression

*k*NN

SVM

MLP

Model Selection/Evaluation

Several different learning models to achieve the same task

Decision Trees

Random Forests

Logistic Regression

*k*NN

SVM

MLP

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

Model Selection/Evaluation

Several different learning models to achieve the same task

Decision Trees

Random Forests

Logistic Regression

*k*NN

SVM

MLP

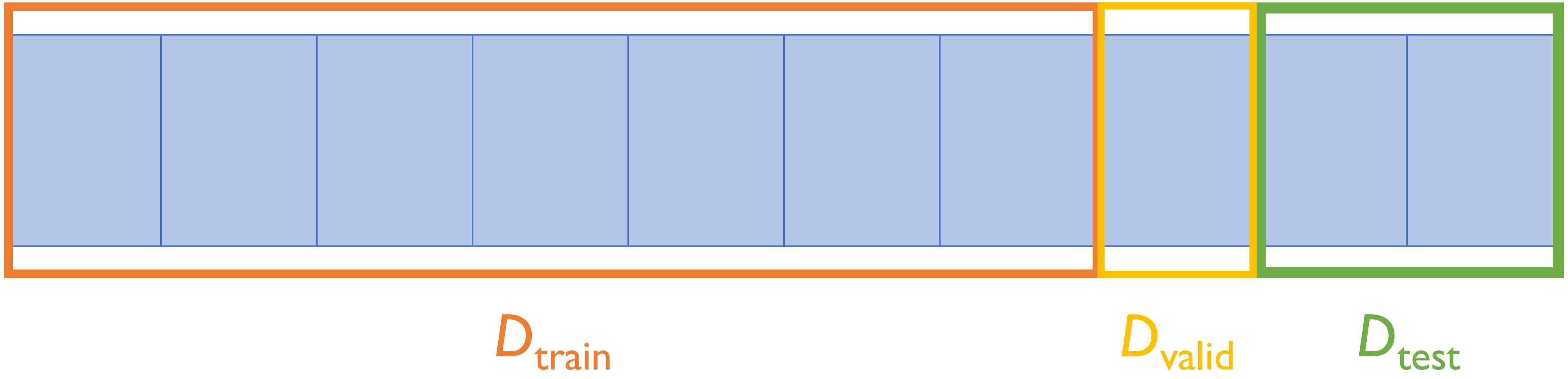
Each learning model has its own set of **hyperparameters** (e.g., the number *k* of neighbours in *k*NN)

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 kNN 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/cacm12.pdf