Teoria degli Algoritmi

Corso di Laurea Magistrale in Matematica Applicata

a.a. 2020-21

SAPIENZA
UNIVERSITÀ DI ROMA

Gabriele Tolomei

Dipartimento di Informatica

Sapienza Università di Roma

tolomei@di.uniroma1.it

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 objects 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 objects 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 computing 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
  - Task/Problem: Find the maximum element of a list of 1 million unsorted 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 unsorted 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
- Example
  - Task/Problem: Find the maximum element of a list of 1 million unsorted 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

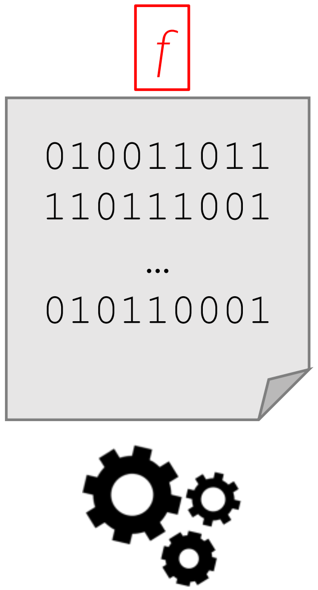
Problem

Solution/Algorithm
explicitly designed by human

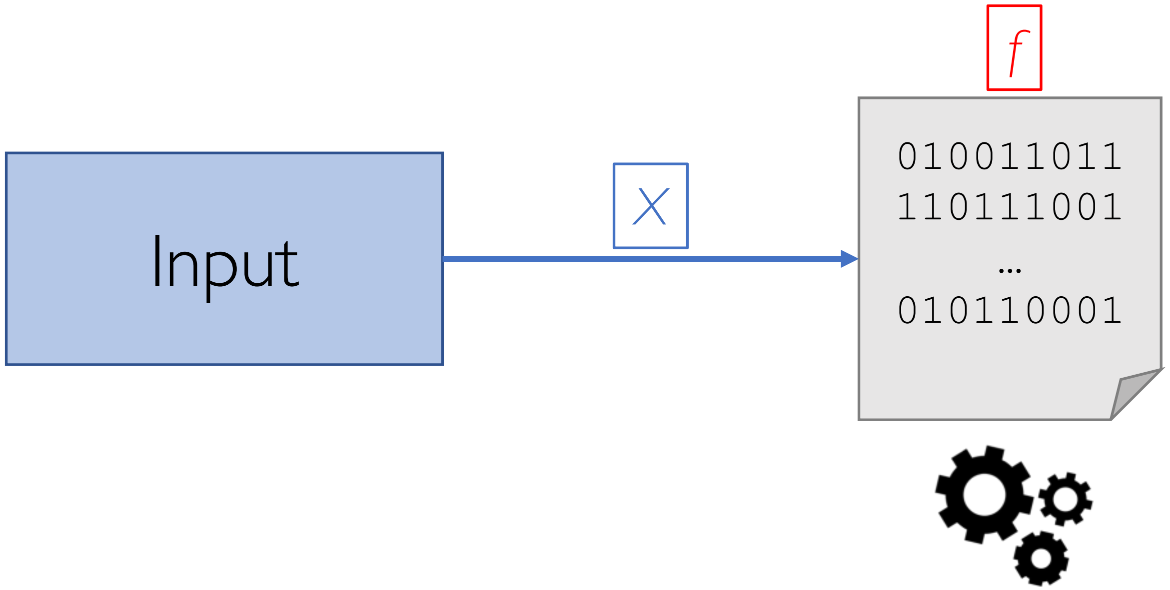
```
010011011
110111001
...
010110001
```

Encoding of
Computable Function f

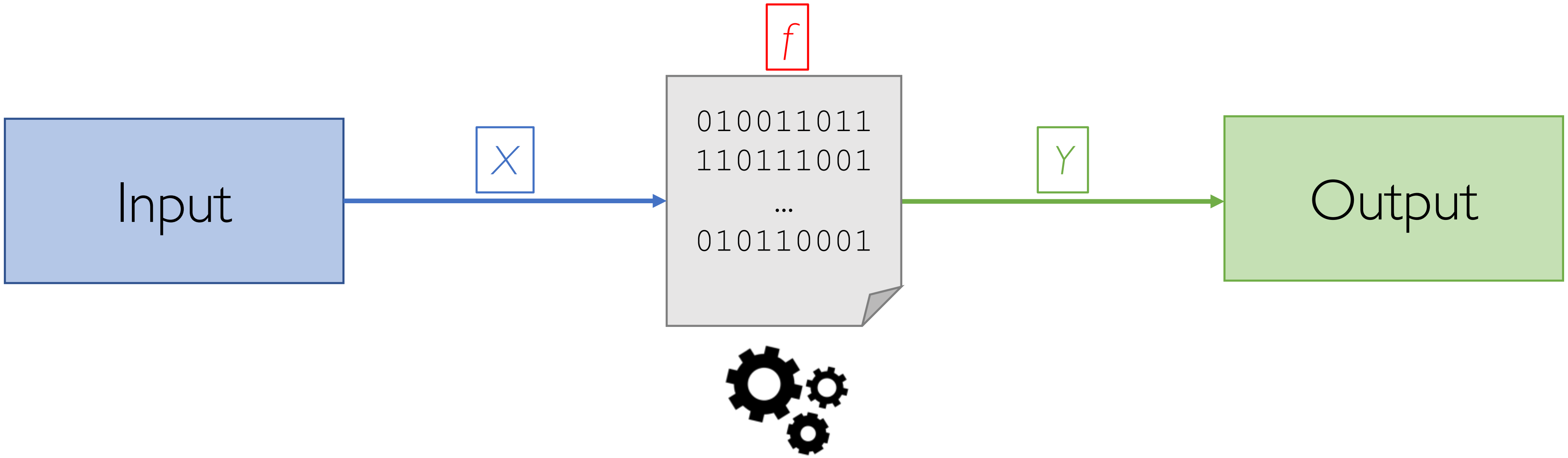
Programming a Computer



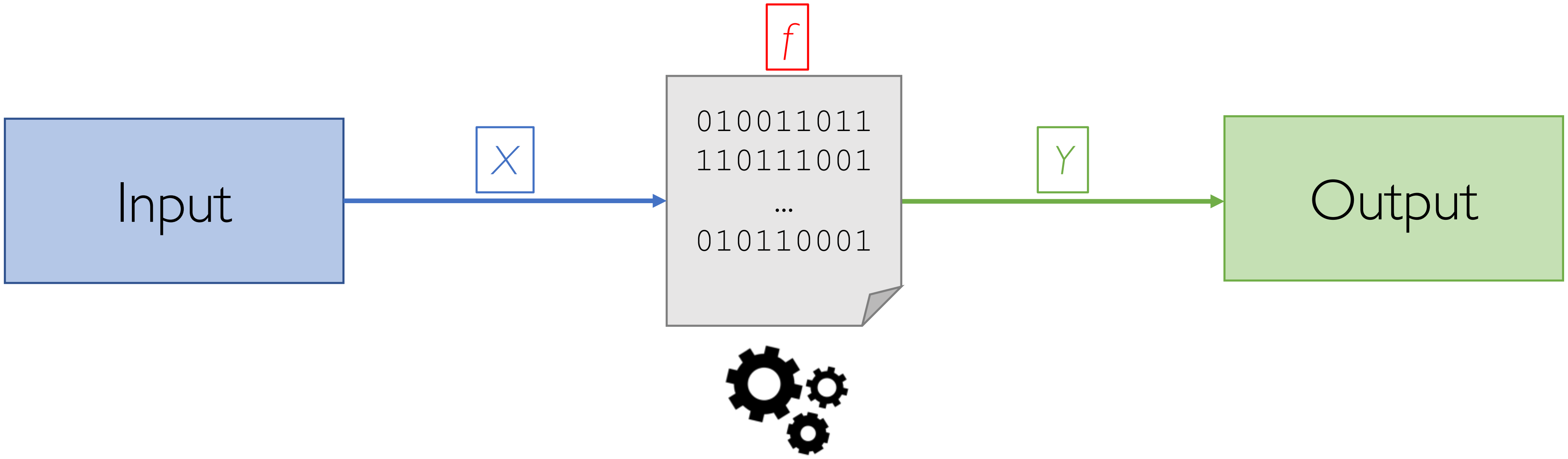
Programming a Computer



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

⇩

Programming vs. "Training" a Computer

"Training" a Computer

Problem

Labeled Data

"Training" a Computer

Problem

Labeled Data

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

"Training" a Computer

Problem

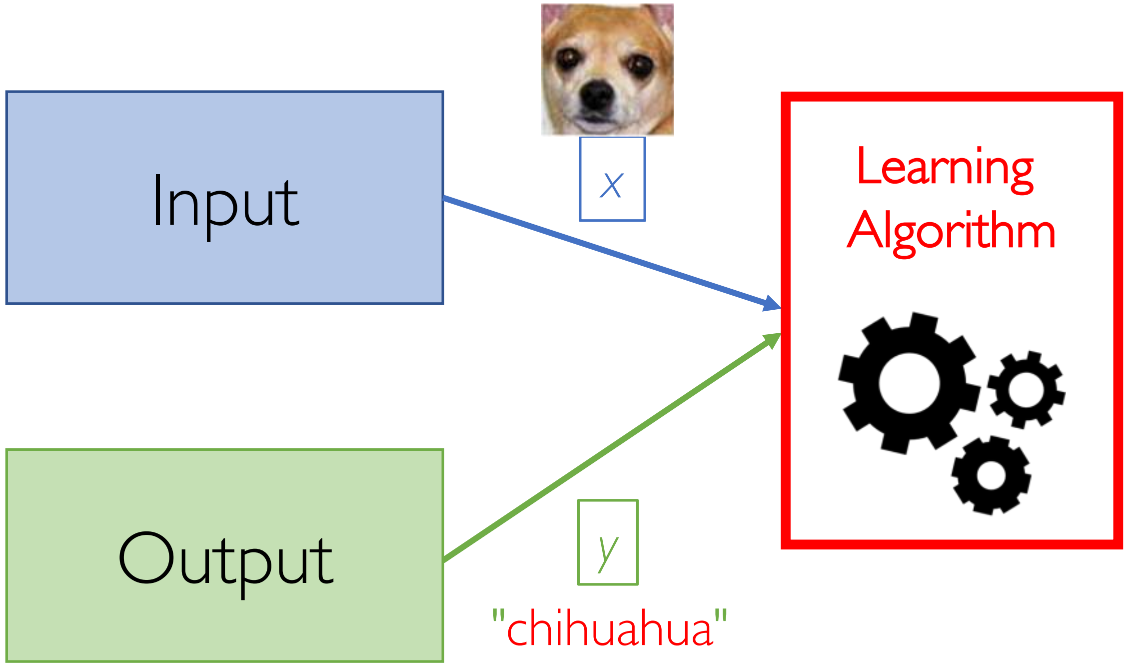
Labeled Data

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

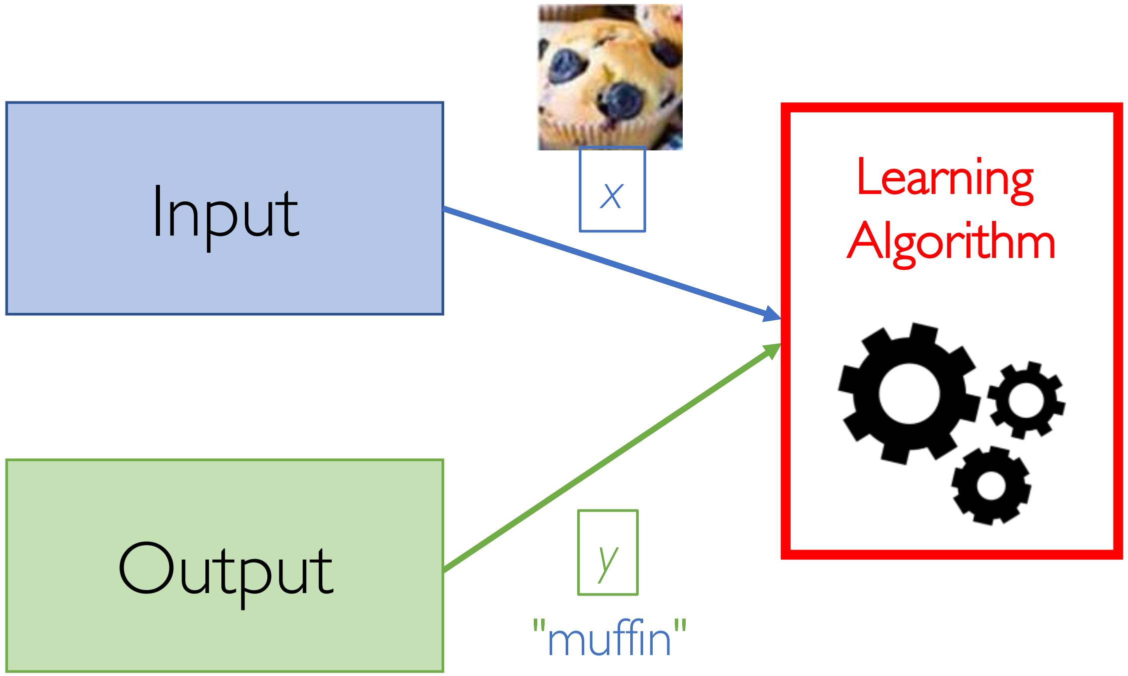
```
010011011
110111001
...
010110001
```

Encoding of
Computable Function f

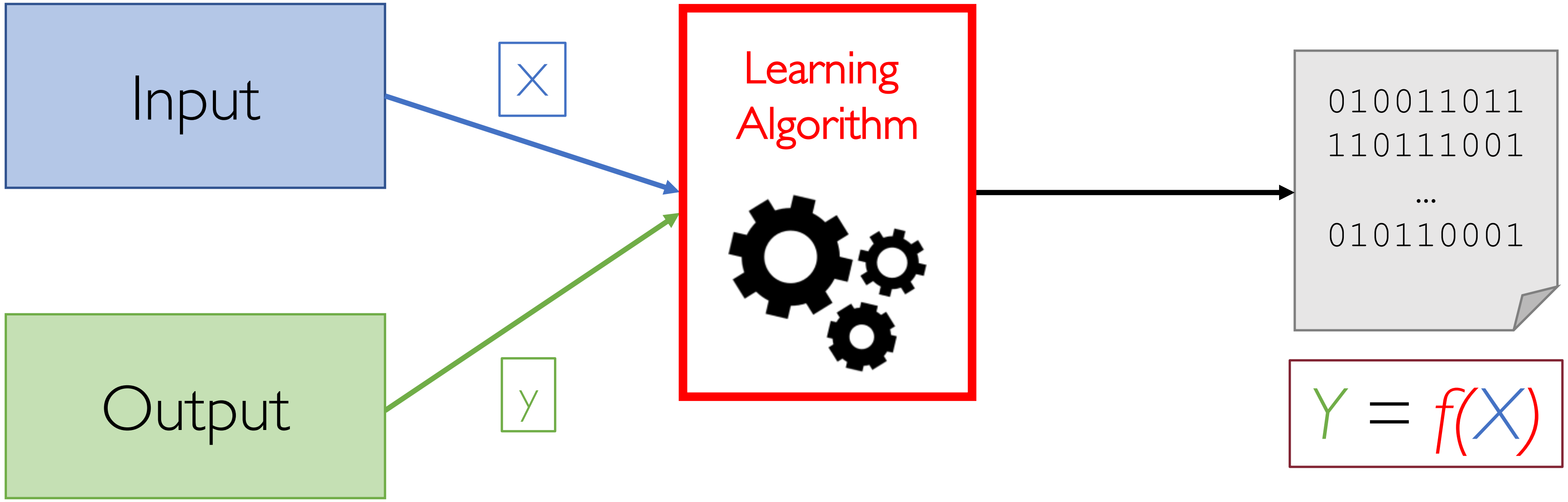
"Training" a Computer



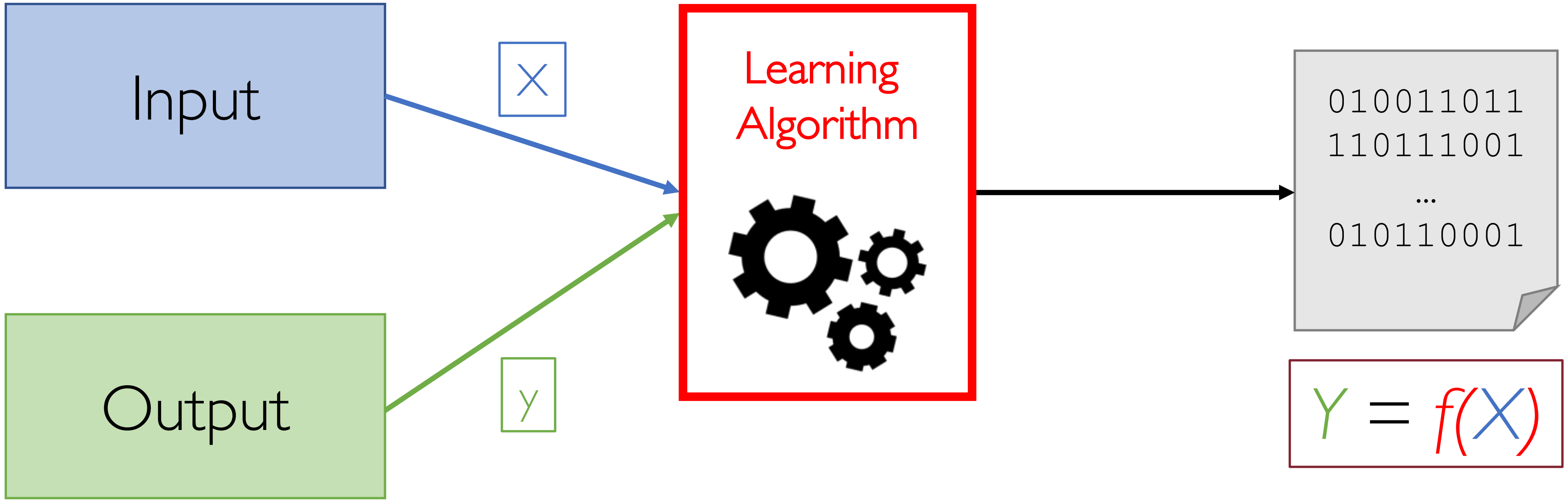
"Training" a Computer



"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 $\{(X, y)\}$

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 $\{(X, y)\}$

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 $\{(X, y)\}$

Reinforcement Learning

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

Supervised Learning: What Do We Predict?

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

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

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

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

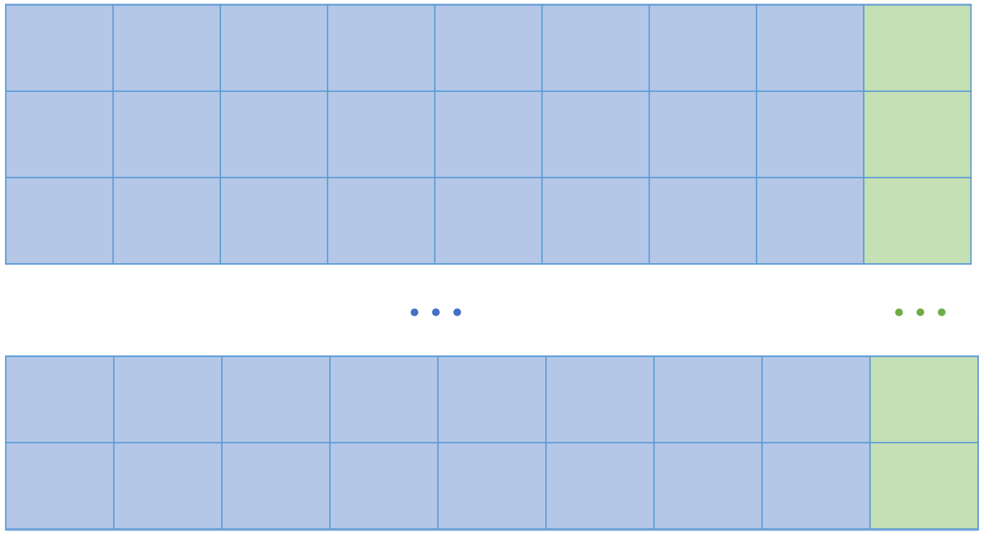
A Bit of Notation

$\mathcal{X} \subseteq \mathbb{R}^n$	input feature space
$\mathcal{Y}$	output space
$\mathcal{Y} \subseteq \mathbb{R}$	real-value label (regression)
$\mathcal{Y} = \{1, \ldots, k\}$	discrete-value label (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

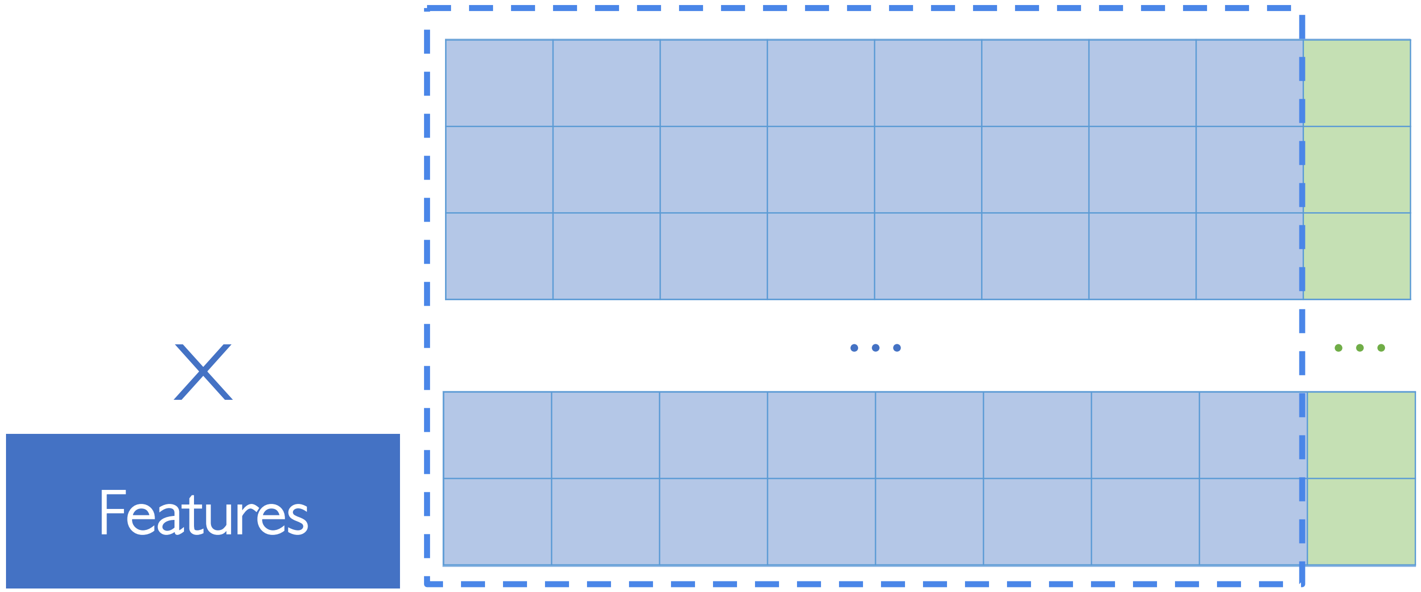
A Bit of Notation

$\mathcal{X} \subseteq \mathbb{R}^n$	input feature space
$\mathcal{Y}$	output space
$\mathcal{Y} \subseteq \mathbb{R}$	real-value label (regression)
$\mathcal{Y} = \{1, \ldots, k\}$	discrete-value label (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

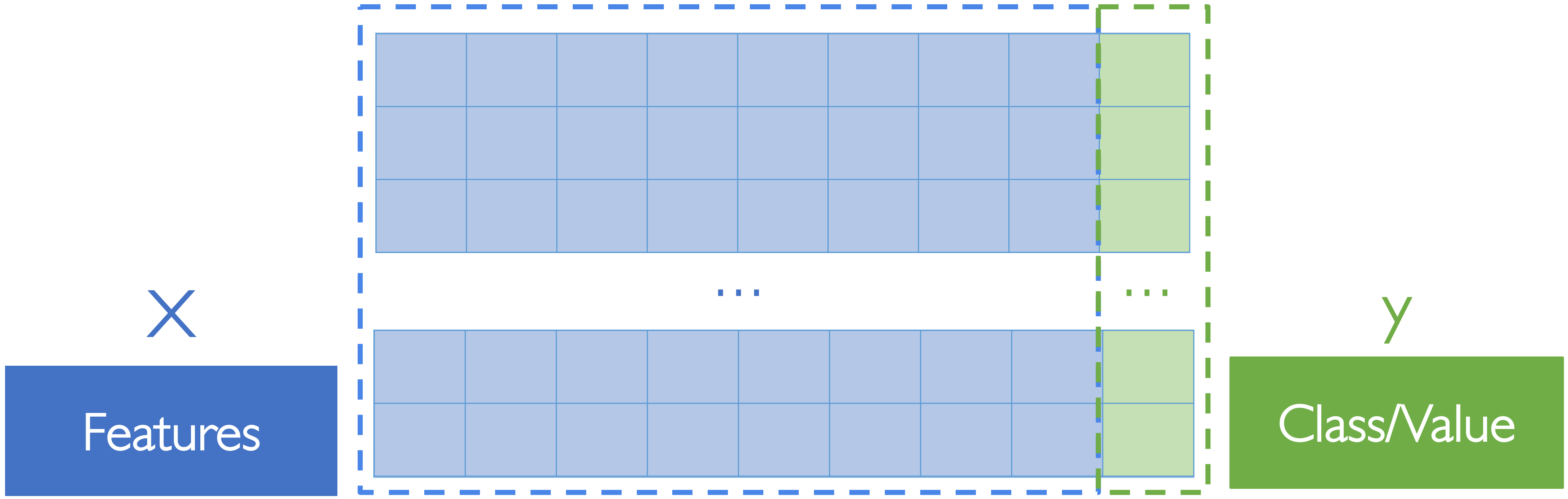
Model Training: Labeled Dataset



Model Training: Labeled Dataset

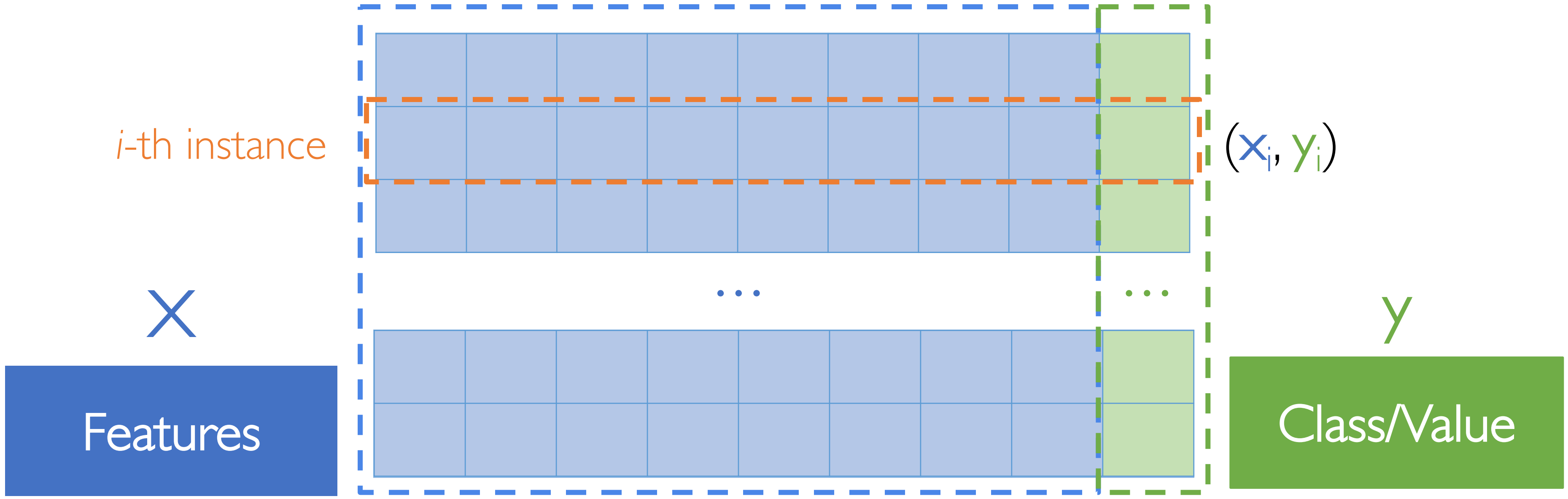


Model Training: Labeled Dataset



Model Training: Labeled Dataset

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



Model Training: Intuition

<u>Idea</u>

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

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

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 as measured on D

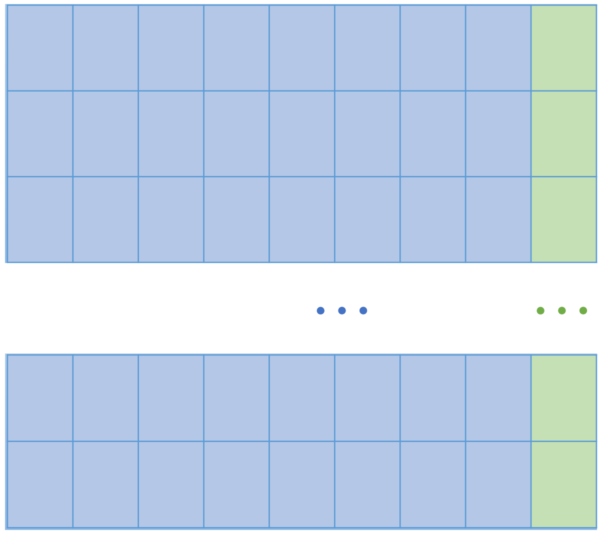
$$\boxed{h^* = \text{argmin}_{h \in \mathcal{H}} L(h, \mathcal{D})}$$

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

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

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

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

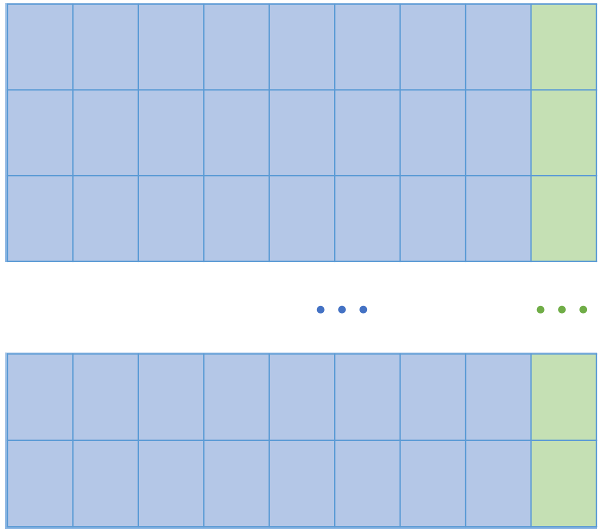


$(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$

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

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

$f = X \rightarrow Y$

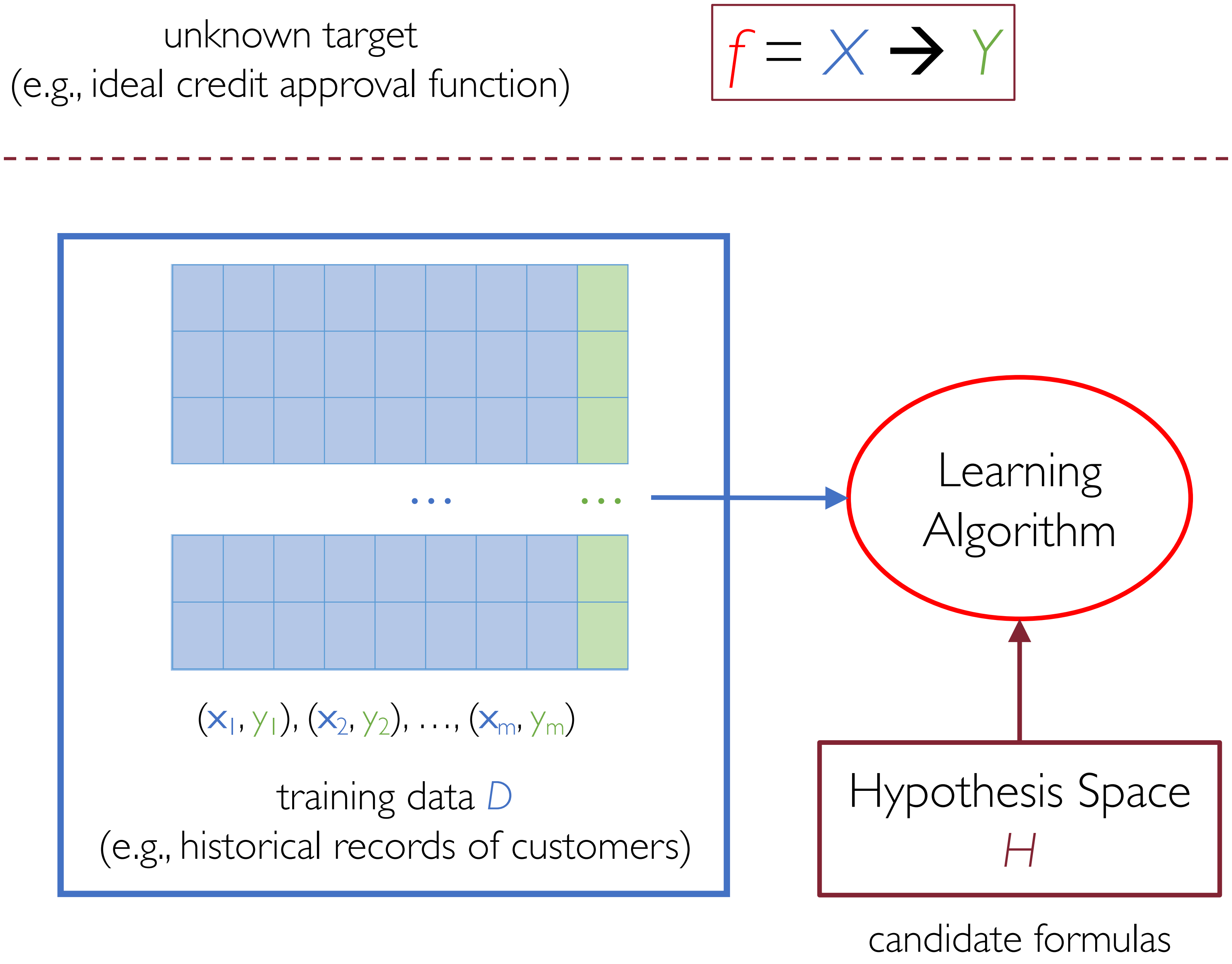


$(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$

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

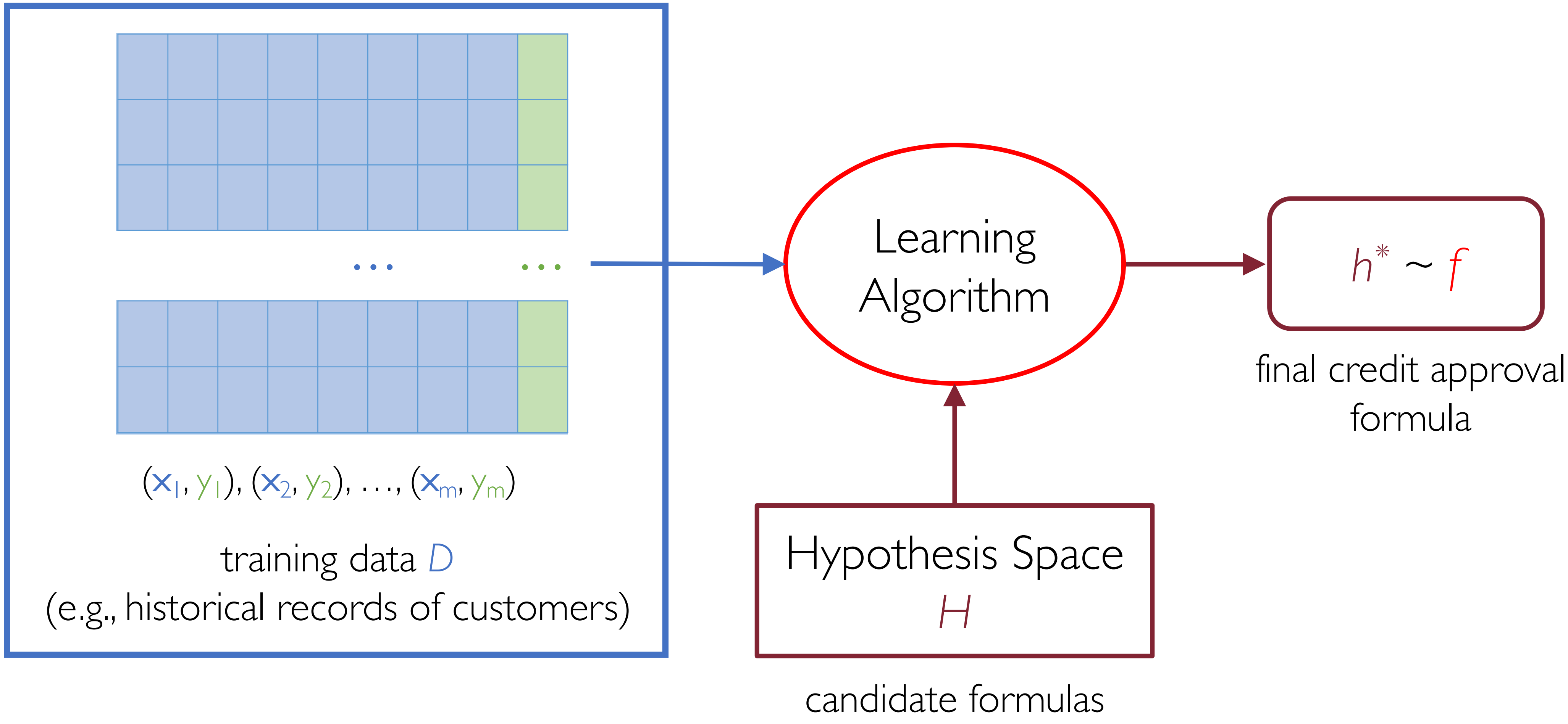
Hypothesis Space
H

candidate formulas


unknown target
(e.g., ideal credit approval function)
$f = X \rightarrow Y$
$(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$
training data D
(e.g., historical records of customers)
Learning Algorithm
Hypothesis Space
H
candidate formulas

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>!

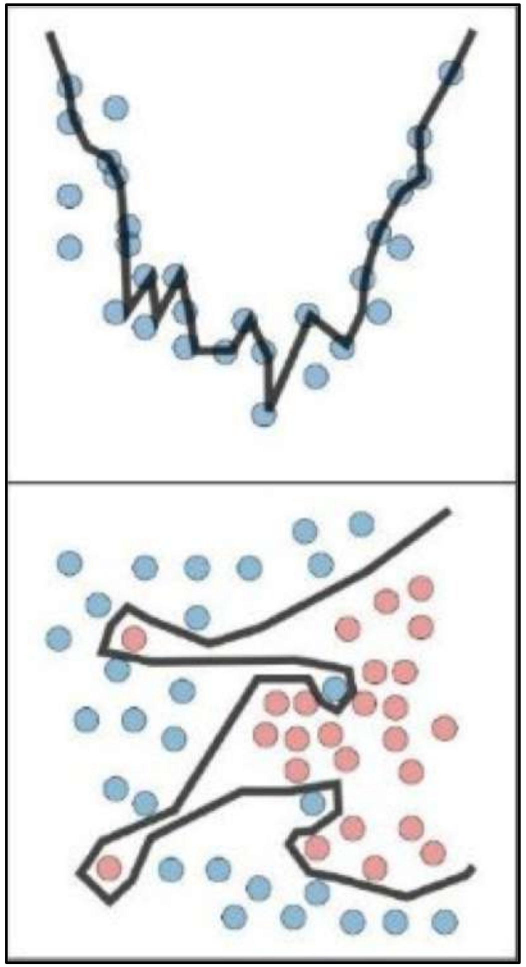
In-Sample vs. Out-of-Sample Error

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

Overfitting (High Variance)

Regression

Classification

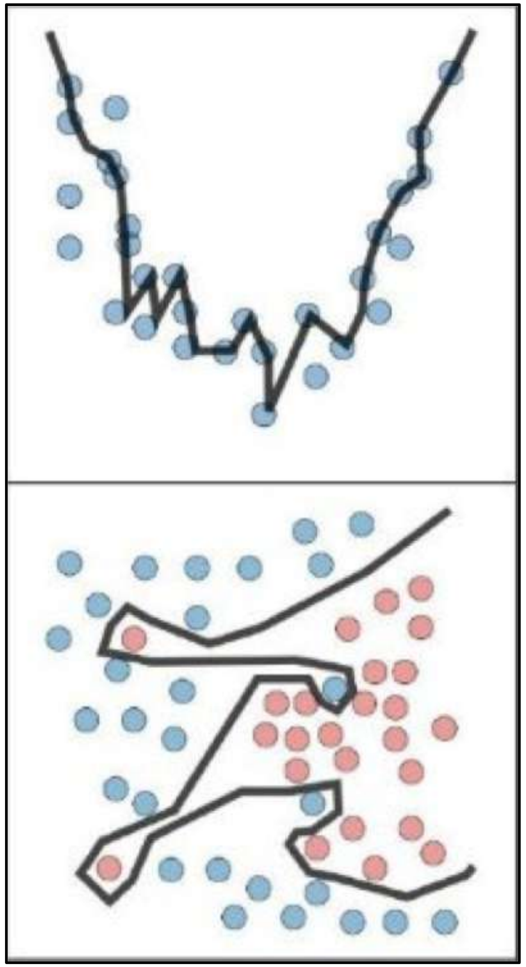


The hypothesis h^* is not learning the true f but it mimics its noise

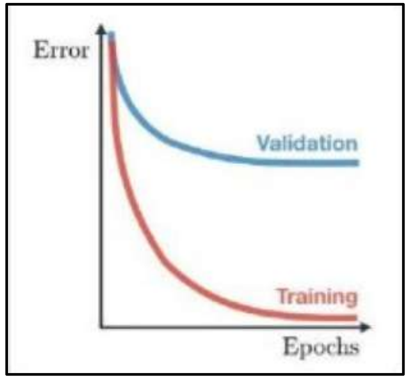
Overfitting (High Variance)

Regression

Classification



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

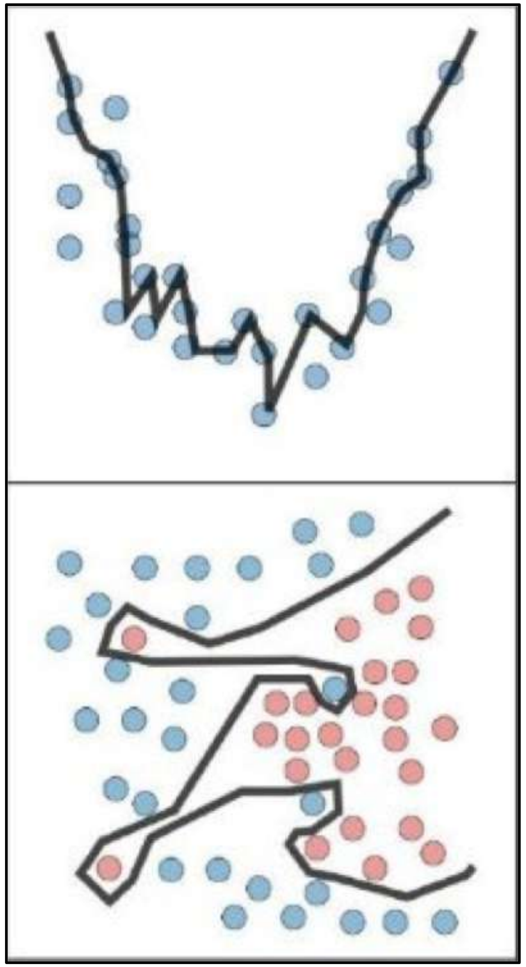


low in-sample error high out-of-sample error

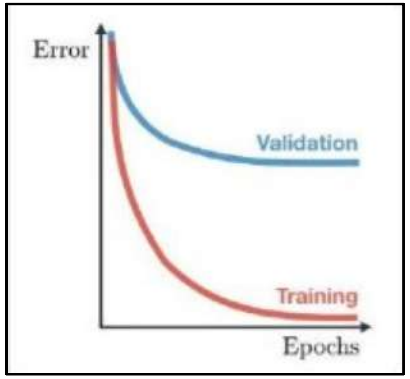
Overfitting (High Variance)

Regression

Classification



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



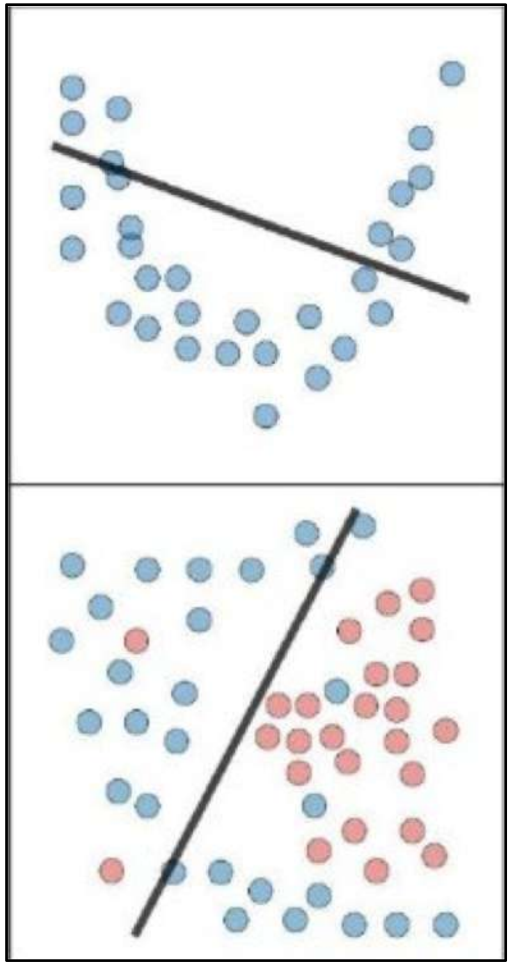
low in-sample error high out-of-sample error

- Regularization
- Get more data

Underfitting (High Bias)

Regression

Classification

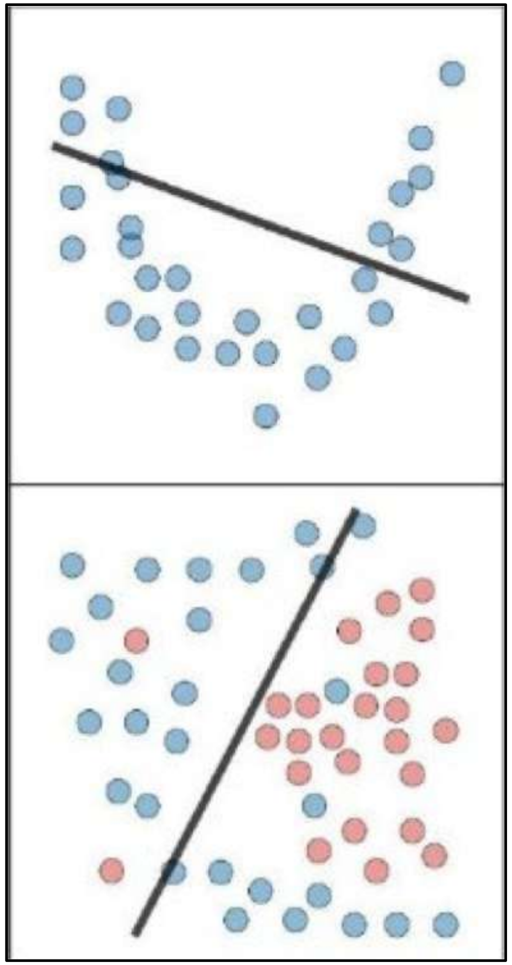


The hypothesis h^* is too "simple" for approximating the true f

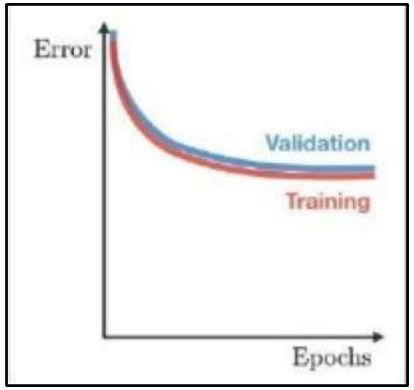
Underfitting (High Bias)

Regression

Classification



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

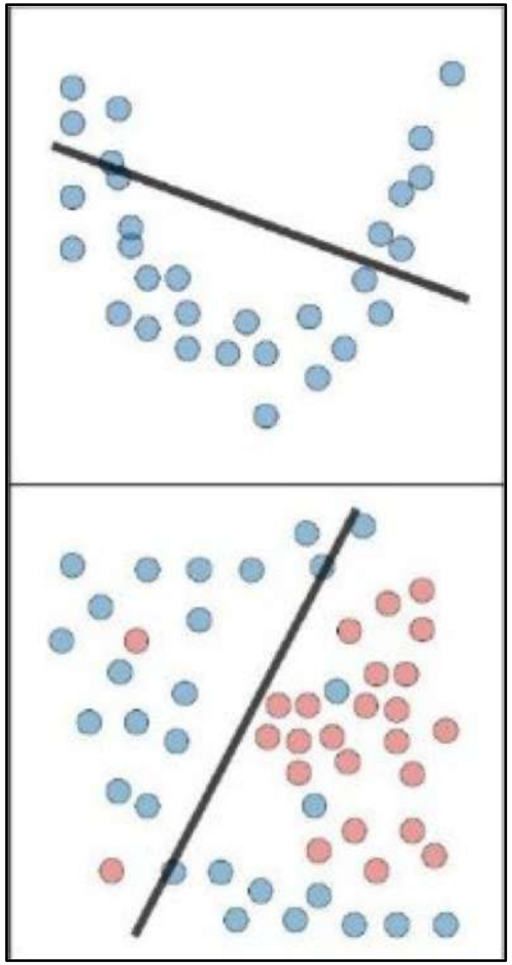


high in-sample error high out-of-sample error

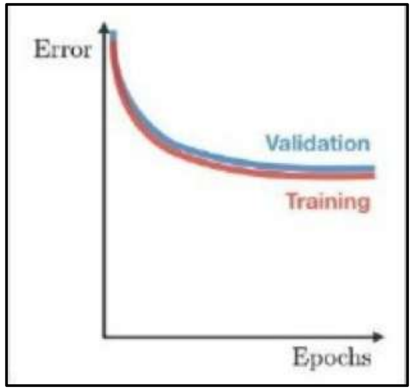
Underfitting (High Bias)

Regression

Classification



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



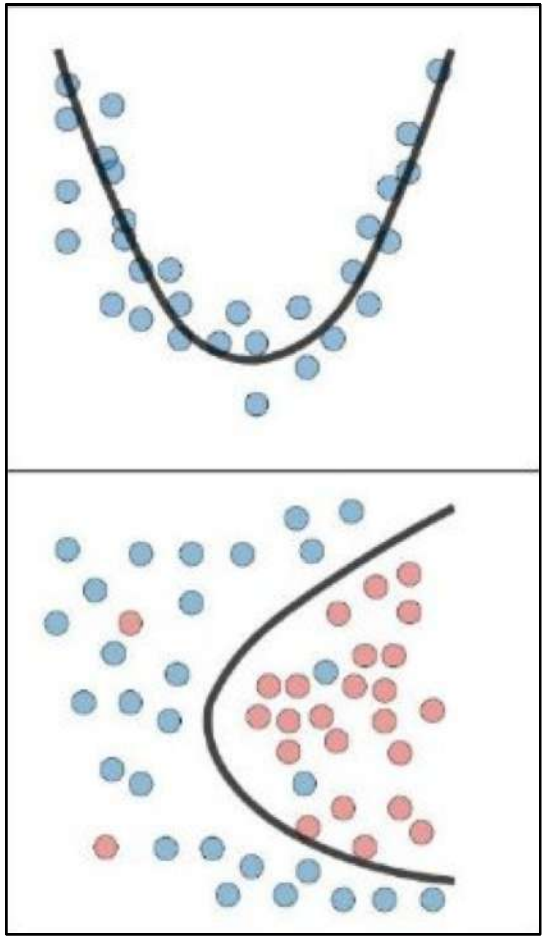
high in-sample error high out-of-sample error

- Increase model complexity
- Add more features

Bias-Variance Tradeoff

Regression

Classification



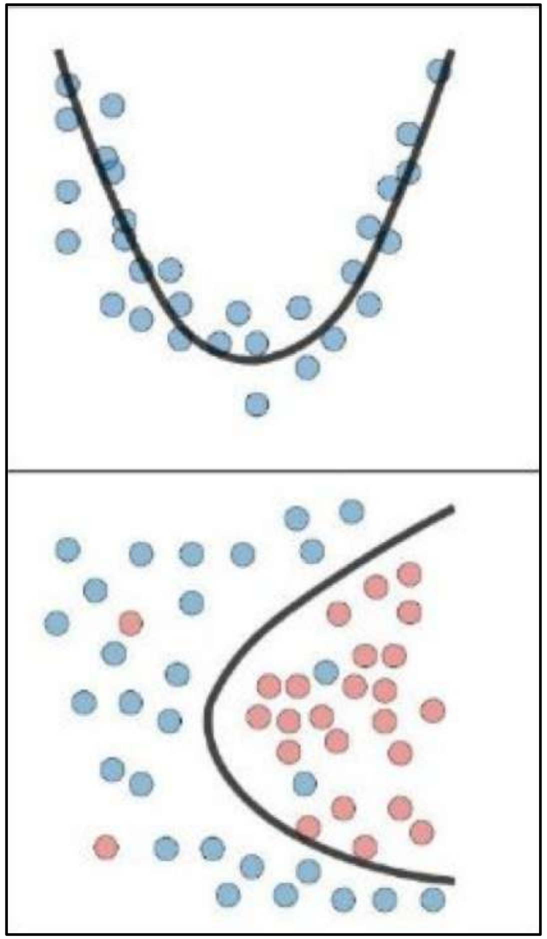
The hypothesis h^* is just right:
the simplest one explaining the data

Occam's razor

Bias-Variance Tradeoff

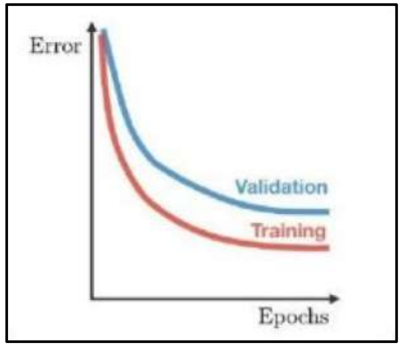
Regression

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 performance online may be too risky

Estimating Generalization Performance

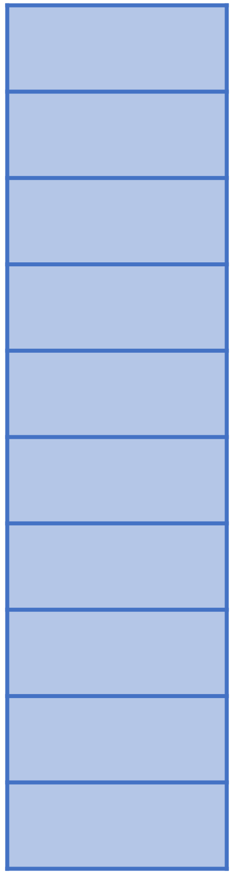
- Measuring the generalization performance online may be too risky
- Example: Don't want to deploy your new spam classifier in production knowing only its training (i.e., in-sample) performance

Estimating Generalization Performance

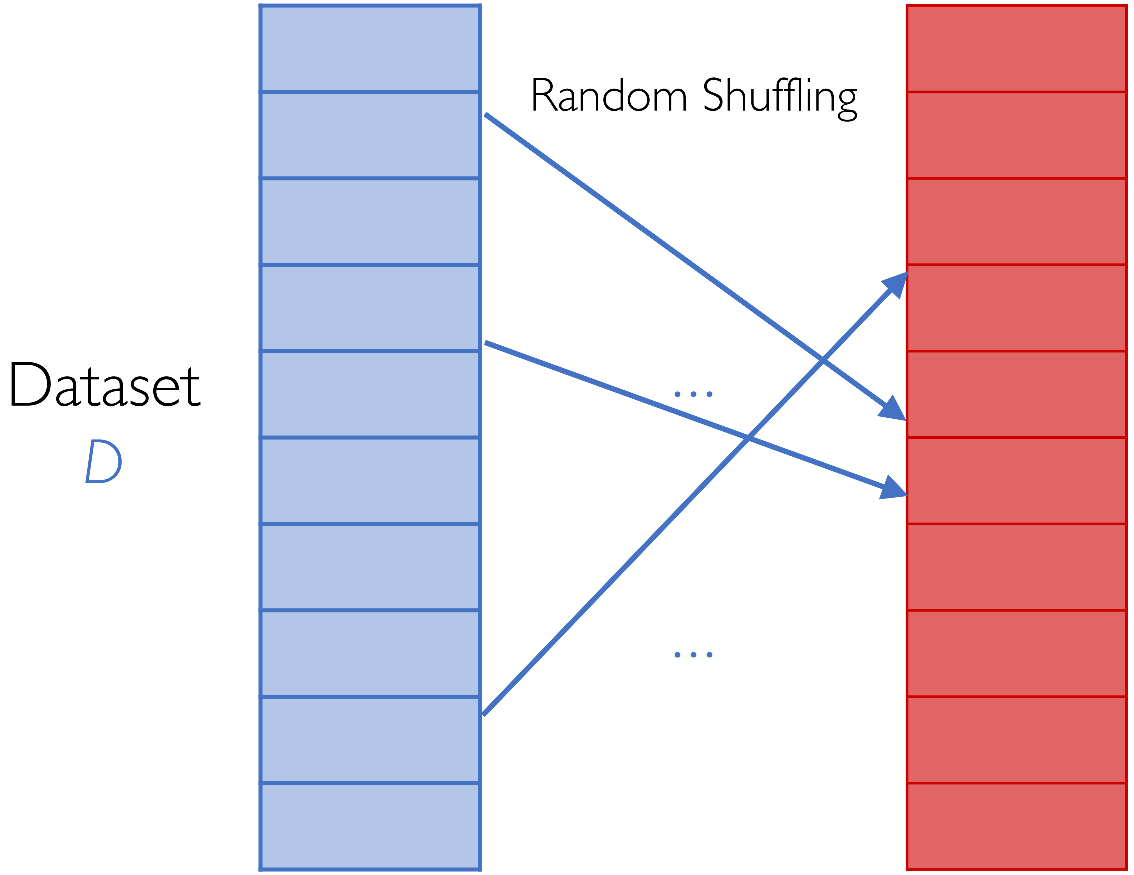
- Measuring the generalization performance online may be too risky
- Example: Don't want to deploy your new spam classifier in production knowing only its training (i.e., in-sample) performance
- Solution: Estimate the generalization performance using training set
  - As long as it holds true the assumption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

Dataset Splitting

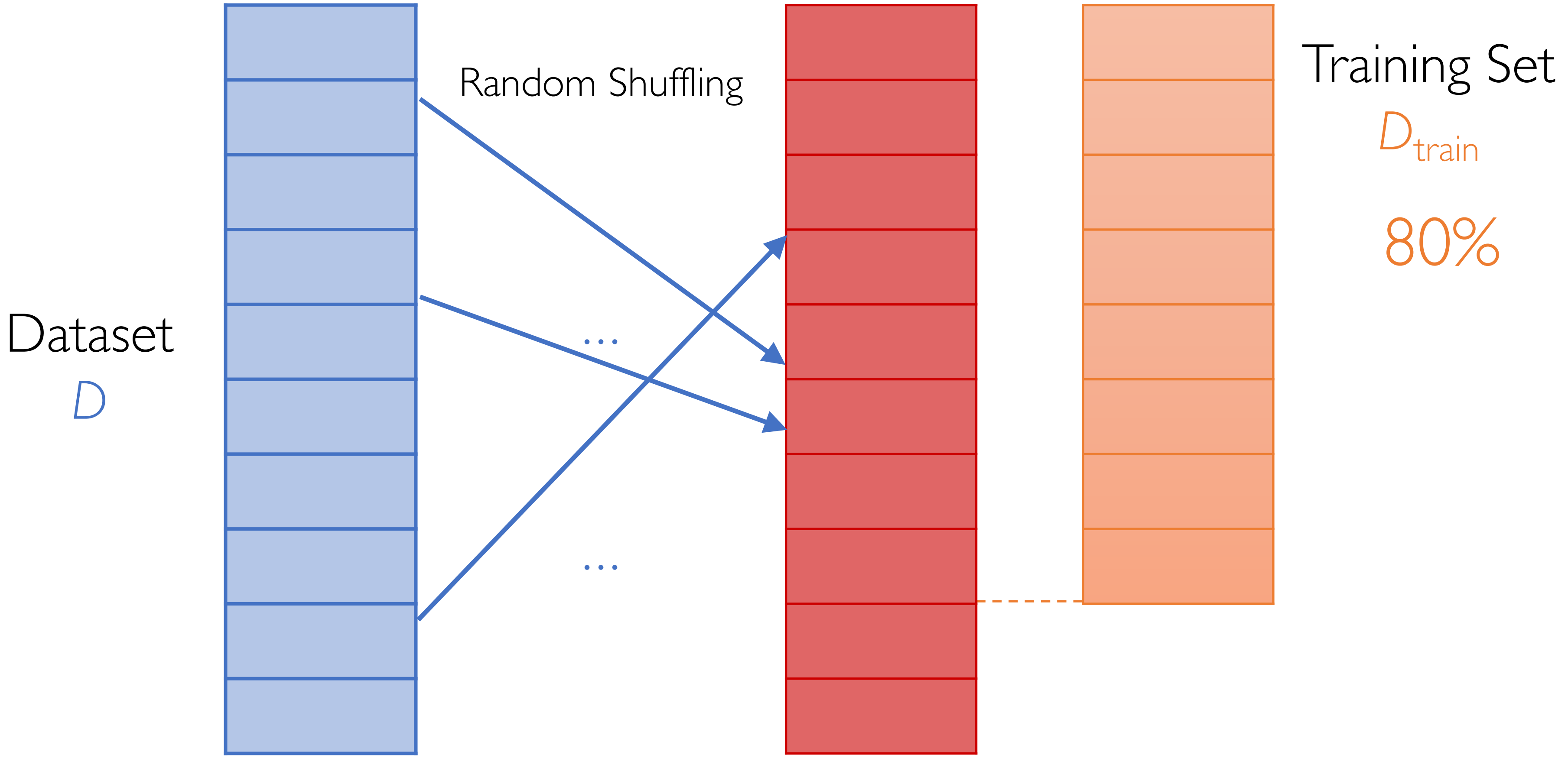
Dataset
D



Dataset Splitting

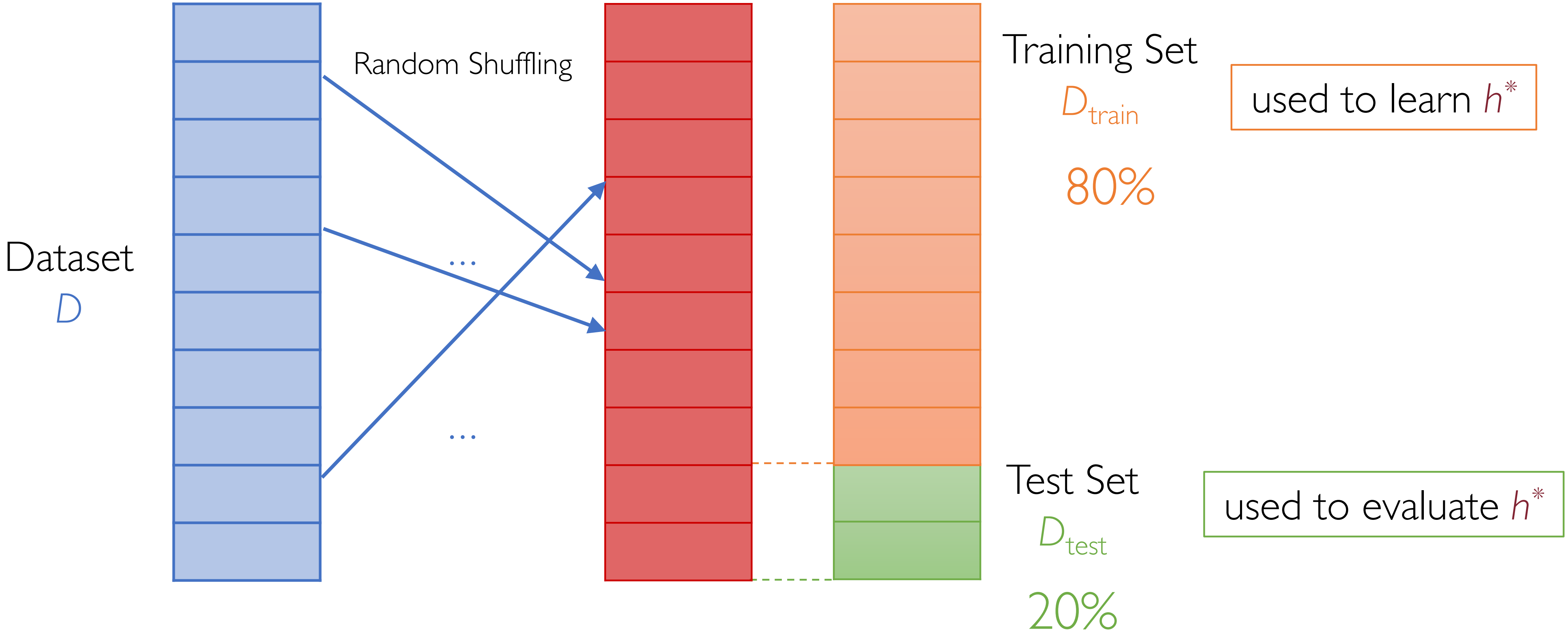


Dataset Splitting



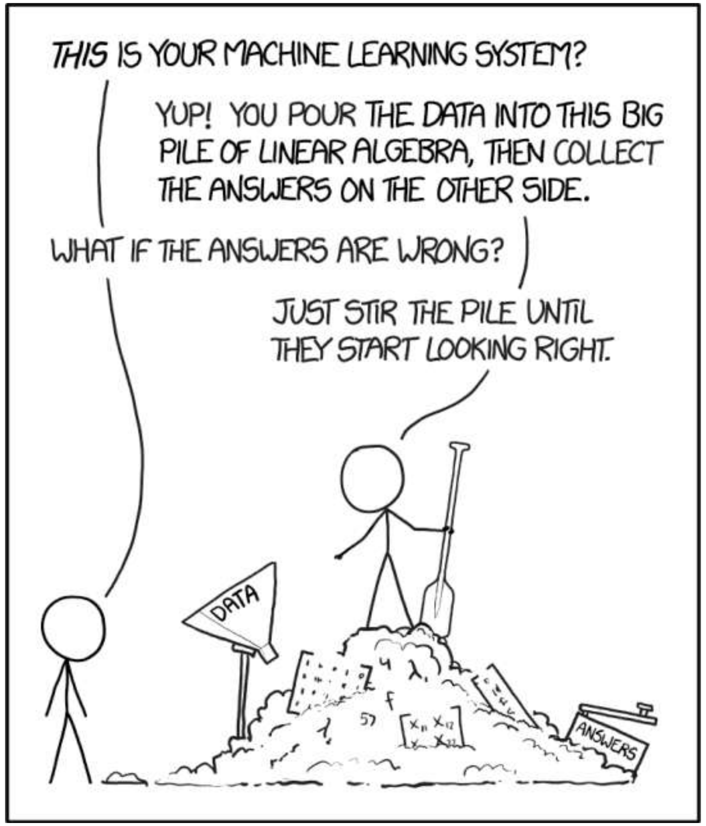
used to learn h^*

Dataset Splitting



How Much Data Do We Need?

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



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