Introduction to Machine Learning (IML)

LECTURE #1: INTRODUCTION

236756 – 2024 WINTER – TECHNION

LECTURER: YONATAN BELINKOV

חזקים אתכם

מוקדי סיוע ותמיכה לסטודנטים

לפניות סטודנטים בנושאים הבאים יש לסרוק את הברקוד

נושאים אקדמיים והתאמות למשרתי מילואים

כגון מועד מיוחד, קורס קדם, חזרה על קורס, קבוצות התמחות, פטורים, שיפור ציון, בקשת גמר לימודים, פגישה עם סגן הדיקן ללימודי הסמכה ועוד

נושאים אישיים

כגון סיוע בלמידה, סיוע רגשי, סיוע כלכלי, ועוד

הפקולטה למדעי המחשב
ע"ש הנרי ומרילין טאוב

הטכניון

Welcome to IML@2024w!

- **Course staff:**
  - Lecturer: Yonatan Belinkov
  - Head TA: Itay Evron & Arkadi Piven
  - TAs: Idan Kinderman, Shani Goren, Yonatan Elul
  - Graders: Natan Kaminski, Rotem Aviv
- **Website**: https://webcourse.cs.technion.ac.il/236756/
- **Piazza**: http://piazza.com/technion.ac.il/winter2024/236756

  Please direct all academic inquiries here.
- **Course email**: 236756ML@gmail.com

  Please direct all personal inquiries here.

Course structure

- **Teaching**:
  - Class (שיעור) – weekly, 2h
    Hybrid
  - Tutorial (תרגול) – weekly, 1h
    Four groups (3 frontal, 1 online)
- **Assignments**:
  - Wet exercises – total of 3
  - Dry exercises – total of 4
- **Exam**

Grading policy:

- **Wet exs:** 18% of final grade (6% each)
  - 3 programming exs
  - Mandatory
  - Done in pairs
- **Dry exs:** 8% of final grade
  - 4 pencil-and-paper assignments
  - Mandatory
  - Done individually
- **Exam:** 74% of final grade
  - Must score 55+ to pass the course

Format

- Following current Technion guidelines, our course will be given **in person**
- We will permit access to recordings from last year **to help students on reserve duty**
- Note there are some differences between semesters in content and emphasis
- **Obligatory course material** is that which is given in **current classes and recitations**

- **We urge you to physically attend class on a regular basis**
- Coming to class has added value that cannot be obtained otherwise
- This is clearly expressed – in grades, and in things that cannot be easily measured
- Use recorded content only when truly in need – relying on it to make life `easier' has a price

Overview

- **Machine learning is great!**
- Its also **highly competitive**
- Simply `knowing' the dry material will not get you far
- Success requires a deep understanding of how things work (and when they don't)

- Our **updated course:**
  - Is both **theoretical** and **applied**
  - Relies on **prior knowledge** – mostly probability and linear algebra
- Our course targets the **fundamentals** of machine learning
- Designed to give you an edge
- Requirements are set accordingly

Overview

- **This is not an easy course:**
  - it is intensive and fast-paced
  - makes use of and combines a variety of tools
  - has challenging homework
  - requires you to adapt a particular kind of ‘thinking’

- **To succeed:**
  - Come to class (if possible) and don’t rely on videos (especially not last minute!)
  - Do your homework – wet and dry
  - Focus on understanding (this is what we check for in the exam)
  - **Don’t take shortcuts**

Goals

- **Our goals for students:**
  - Understand learning inside-out and outside-in
  - Know how things work, why they do, and when they don't
  - Have clear mental model: plan, anticipate, debug
  - Understand assumptions: necessity, power, and limitations
  - Don’t be dazzled by the buzz
- ML is a fast-moving, fast-growing field – **be one step ahead!**

Goals

- **Our main goal – develop foundations**

- Conceptual framework
- Methods
- Tools

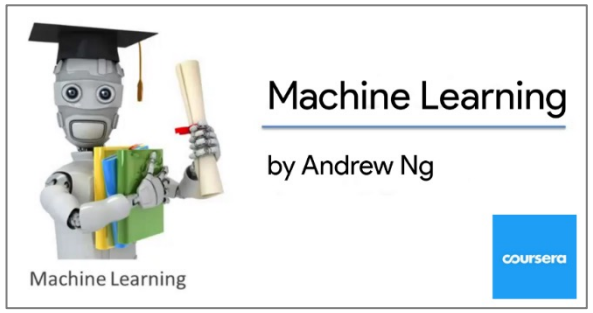
Goals

- **We want to differentiate you from this:**

Goals

- **We want to differentiate you from this:**

=



+ PyTorch + kaggle

Focus

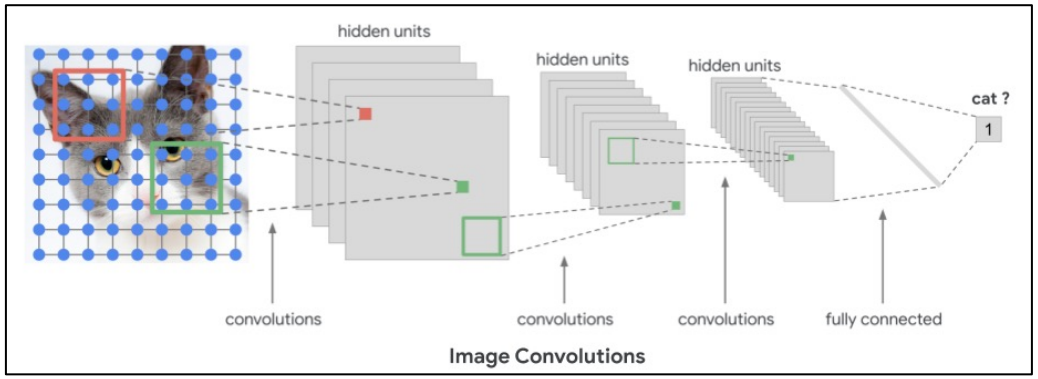
- **Two main means:**

1. **Theory:**

THEOREM 6.11 *Let $\mathcal{H}$ be a class and let $\tau_{\mathcal{H}}$ be its growth function. Then, for every $\mathcal{D}$ and every $\delta \in (0,1)$, with probability of at least $1-\delta$ over the choice of $S \sim \mathcal{D}^m$ we have*

$$|L_{\mathcal{D}}(h) - L_S(h)| \leq \frac{4 + \sqrt{\log(\tau_{\mathcal{H}}(2m))}}{\delta\sqrt{2m}}.$$

2. **Practice:**



Plan for today

1. What is *machine learning*?
2. What is *data*?
3. Towards a formal learning framework
4. (If time permits: simple classification problem)

What is machine learning?

What is machine learning?

Before we dive in – know this:

- Answer depends on who you ask!
- No single, “correct” answer
- Many possible, equally correct viewpoints

- **Our viewpoint**:
  - Narrow – admittedly and intentionally
  - Chosen for pedagogical reasons
  - Aligns with course objectives
  - Remember there is much, much more to learning

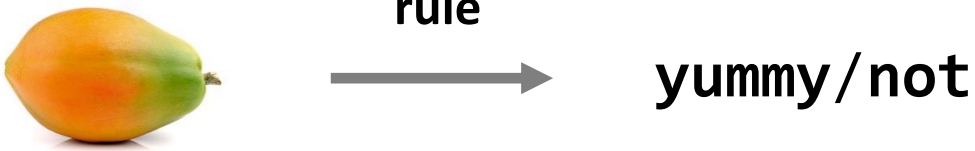
suppose you want to know:

is it yummy?

(without tasting)

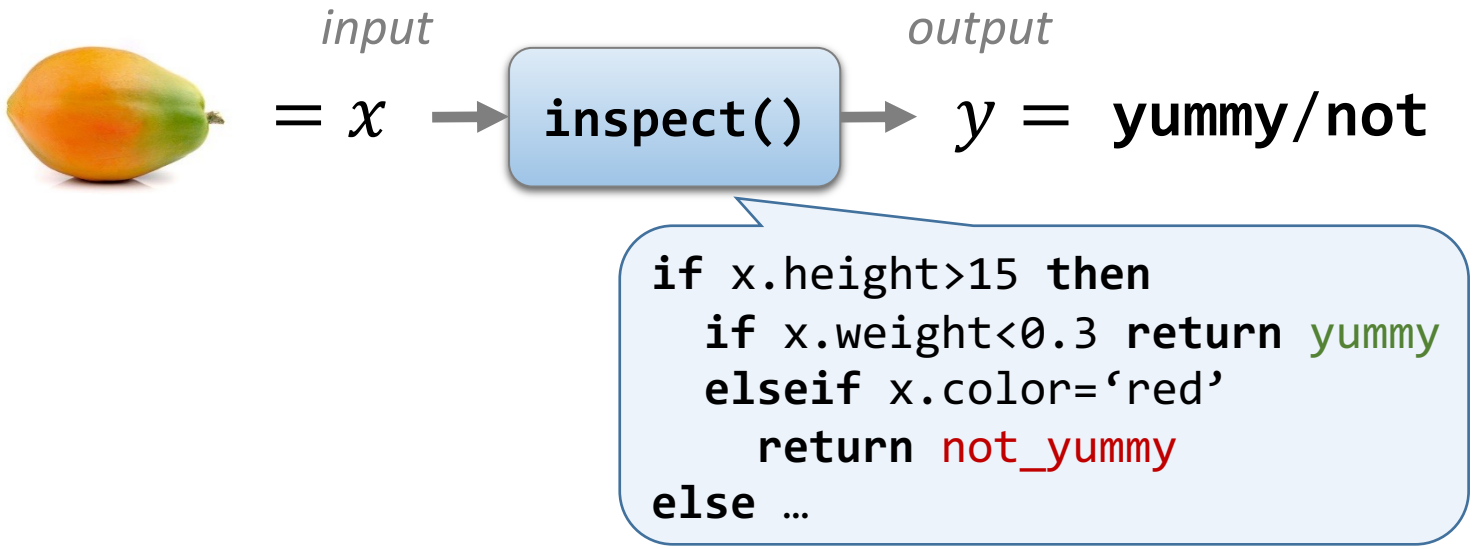
How to spot a good papaya

- **Goal**: design a rule that determines if papayas are yummy or not



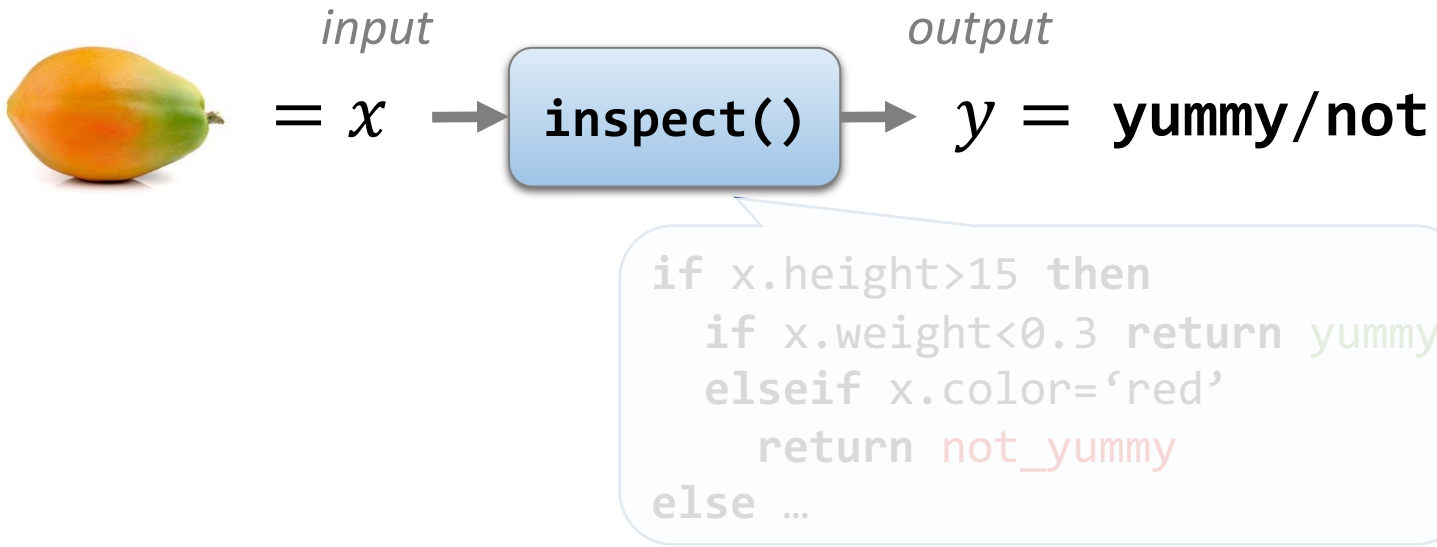
How to spot a good papaya

- **Goal**: design a rule that determines if papayas are yummy or not
- Think of a **rule** as an **algorithm** (often simple)



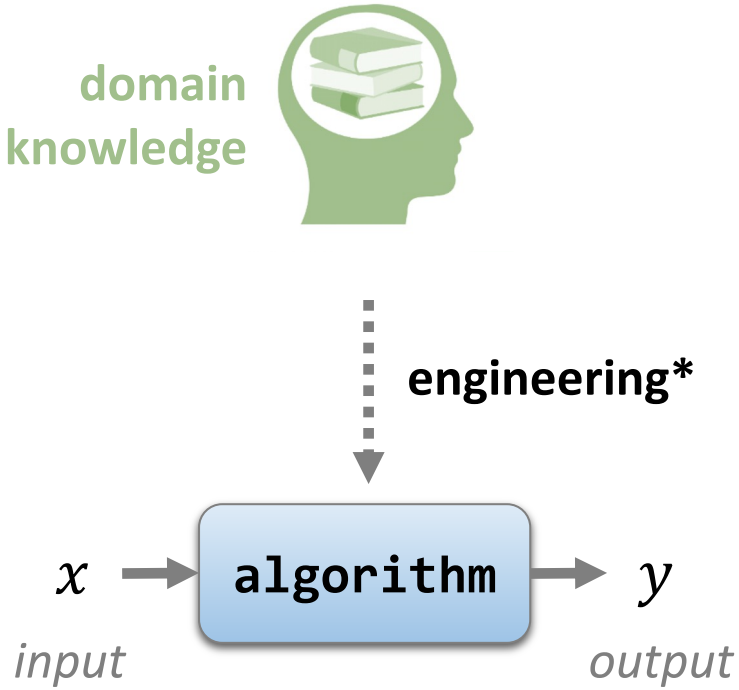
How to spot a good papaya

- **Goal**: design a rule that determines if papayas are yummy or not
- Think of a **rule** as an **algorithm** (often simple)
- How should we come up with this rule?
- **Classical approach**: engineer rule (from *knowledge*)



How to spot a good papaya

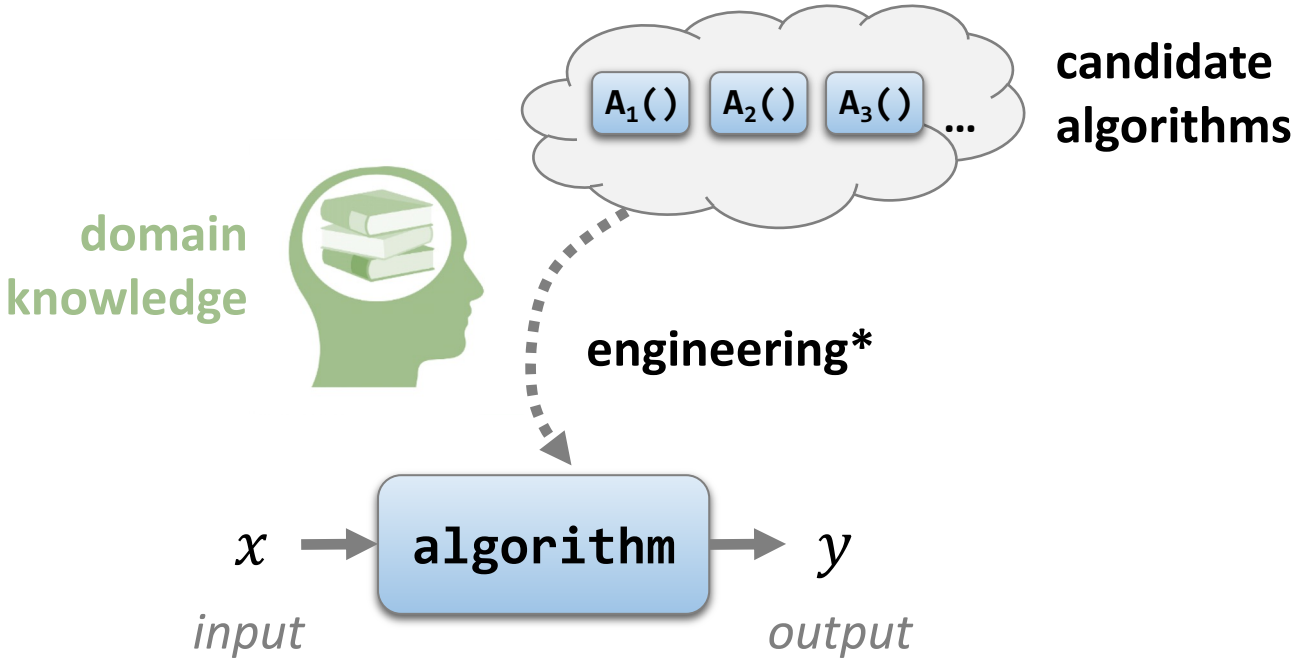
- **Classical approach**: engineer rule (from *knowledge*)



* this definition might make some engineers upset

How to spot a good papaya

- **Classical approach**: engineer rule (from *knowledge*)



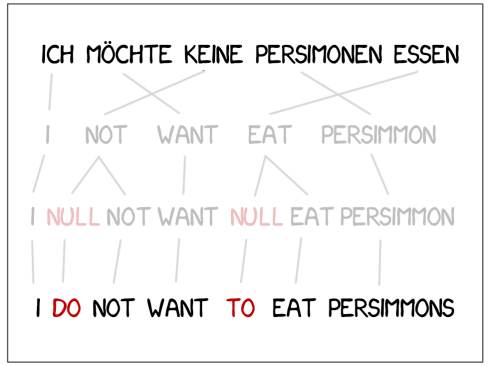
* this definition might make some engineers upset

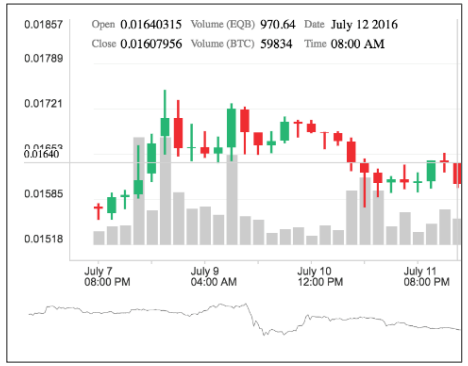
When should we engineer?

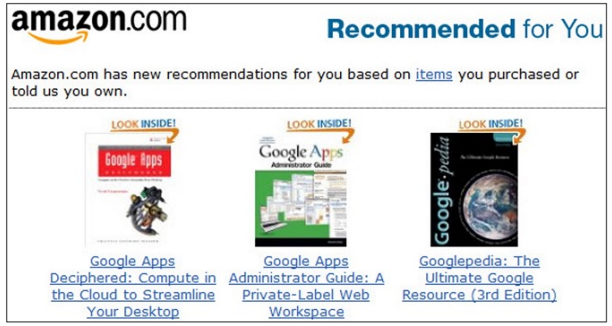
- Engineering works well when we have *sufficient, precise knowledge.*
- Typically results in *simple* rules.
- Is this true for determining yumminess of papayas? Maybe.

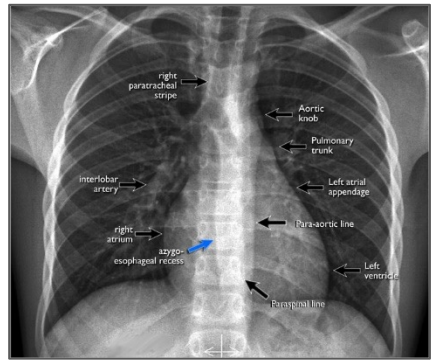
When should we engineer?

- Engineering works well when we have *sufficient, precise knowledge.*
- Typically results in *simple* rules.
- Is this true for determining yumminess of papayas? Maybe.
- But what about:









- Engineering good rules for these is **not an easy task**!

What is learning?

- **Classical approach**: engineer rule (from *knowledge*)
- **Alternative approach**: *learn* rule (from data)

data =

What is learning?

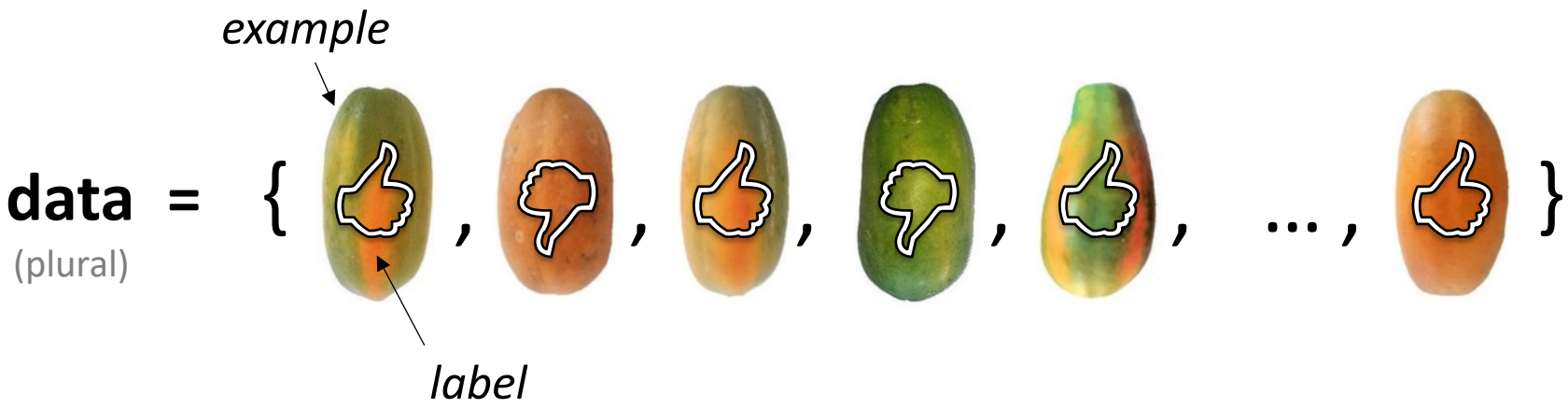
- **Classical approach**: engineer rule (from *knowledge*)
- **Alternative approach**: *learn* rule (from data)

datum = (singular)

= *input-output pair*

What is learning?

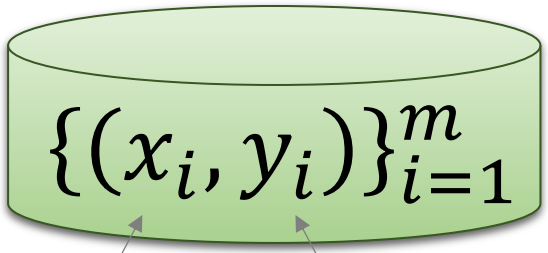
- **Classical approach**: engineer rule (from *knowledge*)
- **Alternative approach**: *learn* rule (from data)



What is learning?

- **Classical approach**: engineer rule (from *knowledge*)
- **Alternative approach**: *learn* rule (from data)

- For our purposes:

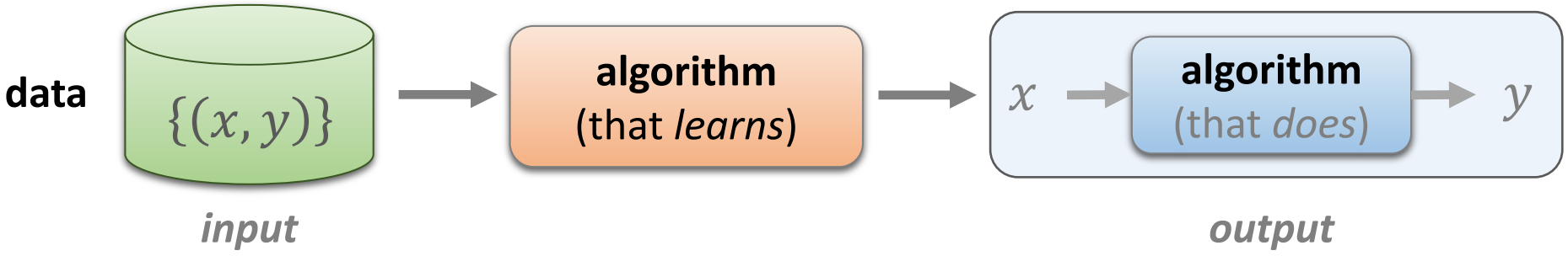
data =  = *set of input-output pairs*

What is learning?

- In learning we infer **rules** (algorithms) from data using …

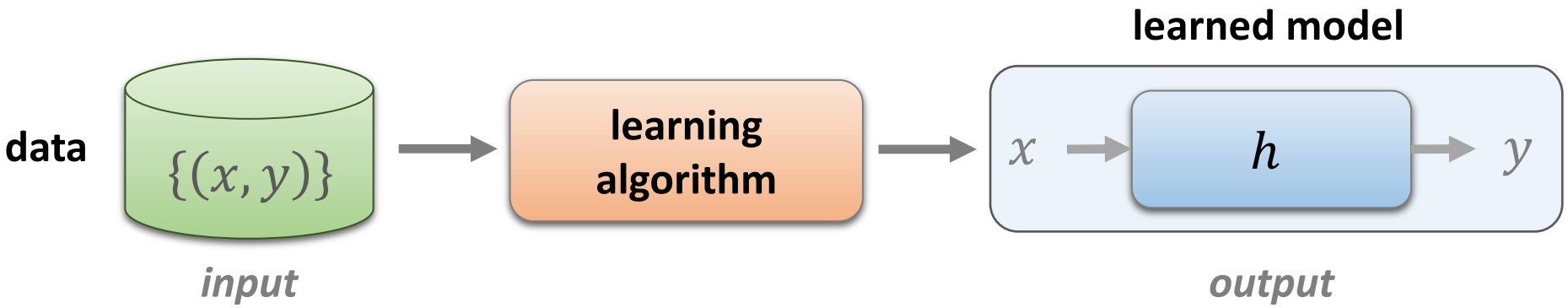
What is learning?

- In learning we infer **rules** (algorithms) from data using … **algorithms**!



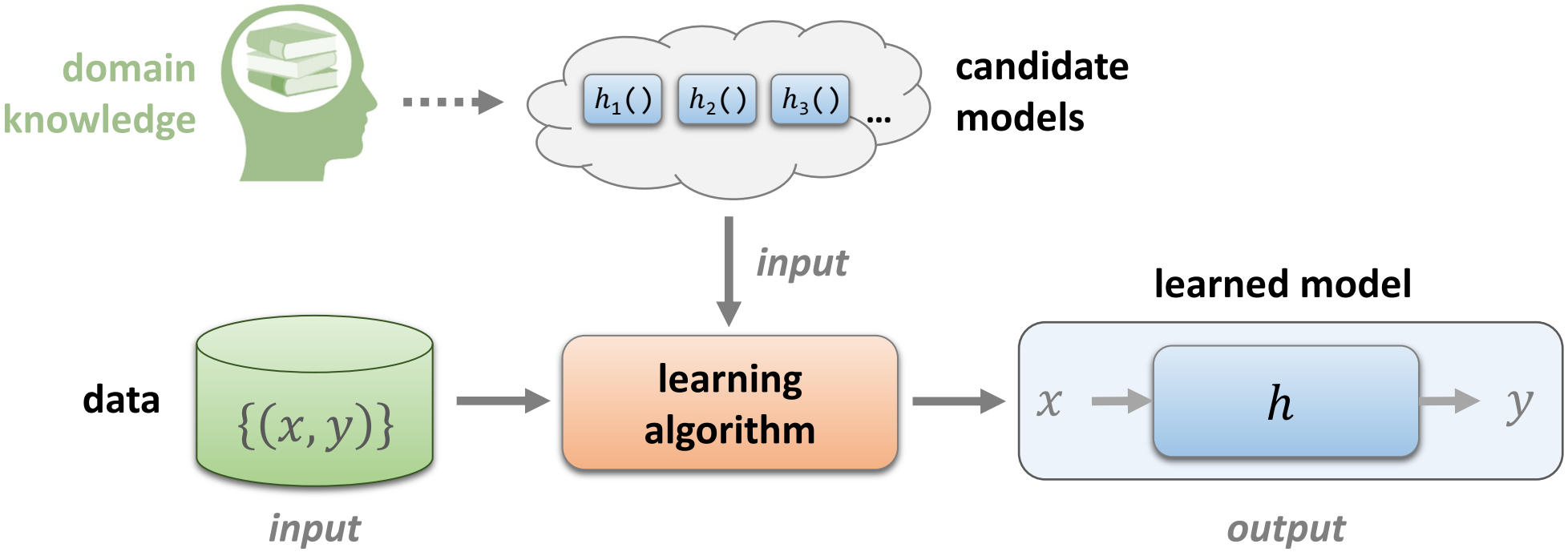
What is learning?

- In learning we infer **rules** (algorithms) from data using … **algorithms**!



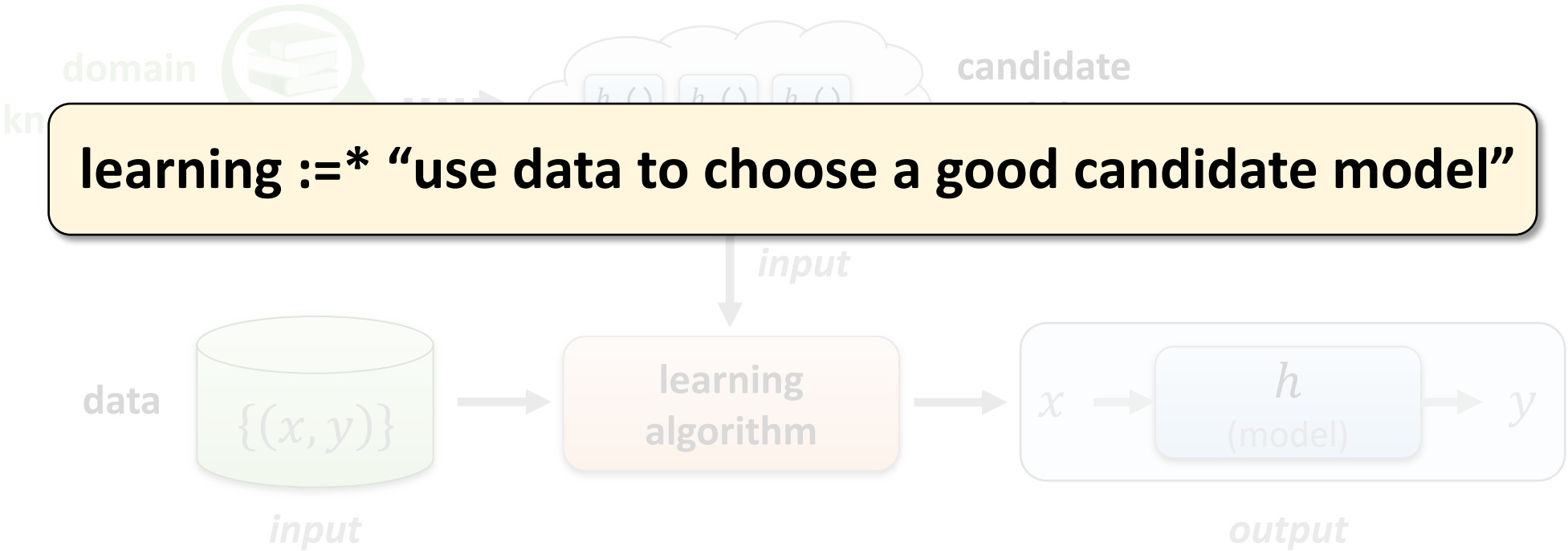
What is learning?

- In learning we infer **rules** (algorithms) from data using … **algorithms**!
- Does this mean we don't need knowledge? (of course not)



What is learning?

- In learning we infer **rules** (algorithms) from data using … **algorithms**!
- But what do these algorithms do?



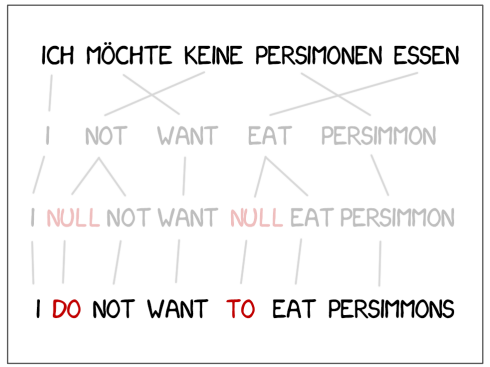
* for our purposes

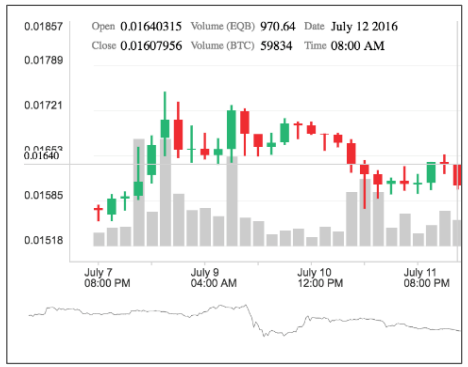
When should we learn?

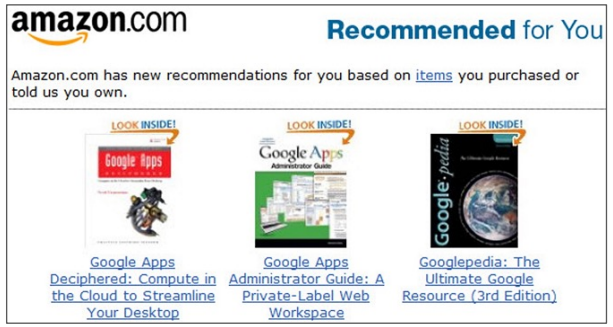
- Learning typically works well when we have:
  1. a sufficient understanding of the general *structure* of the problem
  2. lots and lots (and lots) of *data*

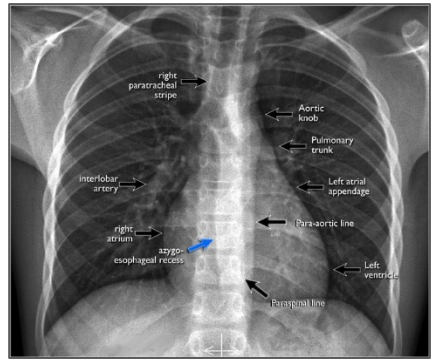
When should we learn?

- Learning typically works well when we have:
  1. a sufficient understanding of the general *structure* of the problem
  2. lots and lots (and lots) of *data*
- Just a small selection of real problems in which ML (arguably) does well:









- However, learning often produces *complex*, uninterpretable rules.
- Above conditions neither *sufficient* nor *necessary*; will be clearer as we progress.

Our three pillars of learning

- Back to our hand-wavy definition of learning:

learning := “use data to choose a good candidate model”

Our three pillars of learning

- Back to our hand-wavy definition of learning:

learning := “use data to choose a good candidate model”

optimization

Our three pillars of learning

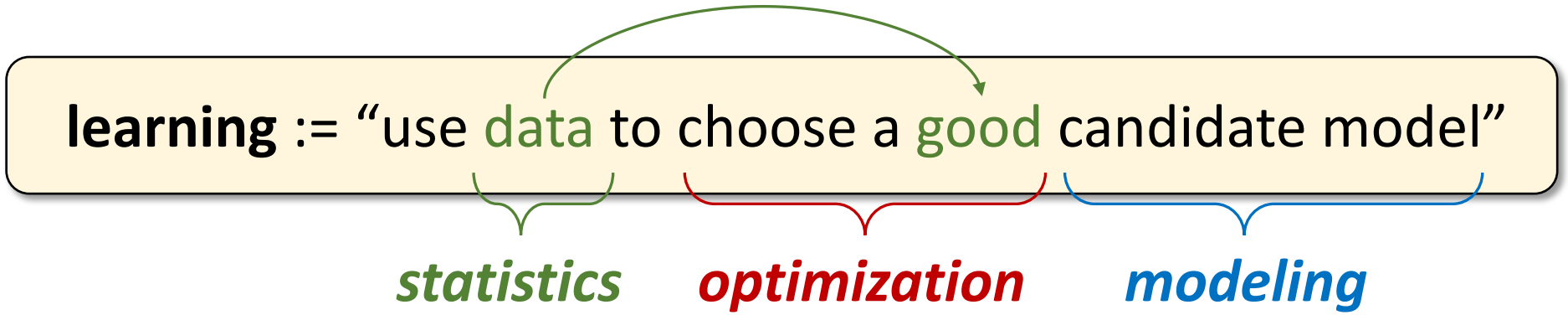
- Back to our hand-wavy definition of learning:

learning := “use data to choose a good candidate model”

optimization ***modeling***

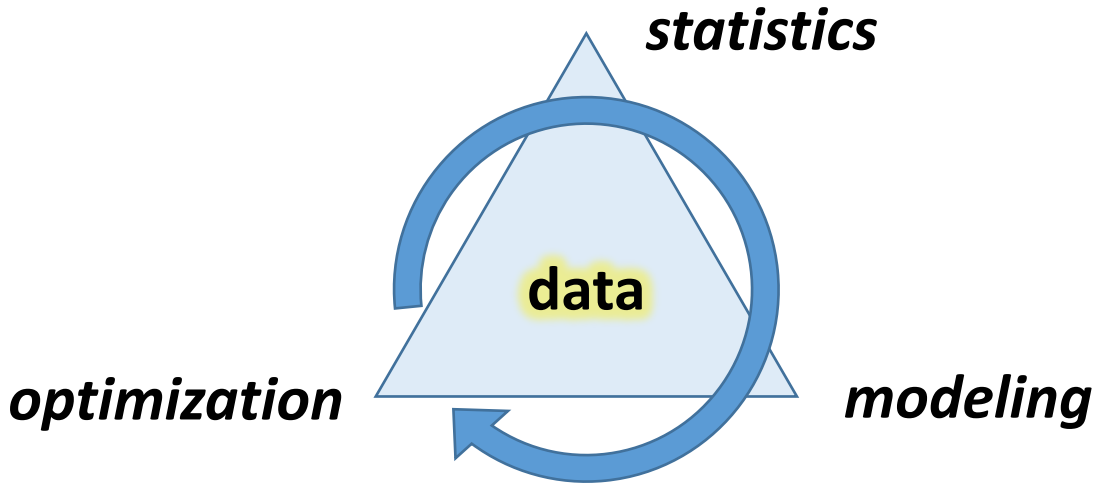
Our three pillars of learning

- Back to our hand-wavy definition of learning:



Our three pillars of learning

- **For our purposes, learning is:**



- To understand learning, we must understand each of these aspects
- But most importantly, we must understand how they interact
- Throughout our course we will visit and revisit all three aspects
- **Unifying element: *data***

What is data?

Data

- **Three questions:**

1. *What* is data?
2. *Why* do we need data?
3. *How* do we use data?

1. What is data?

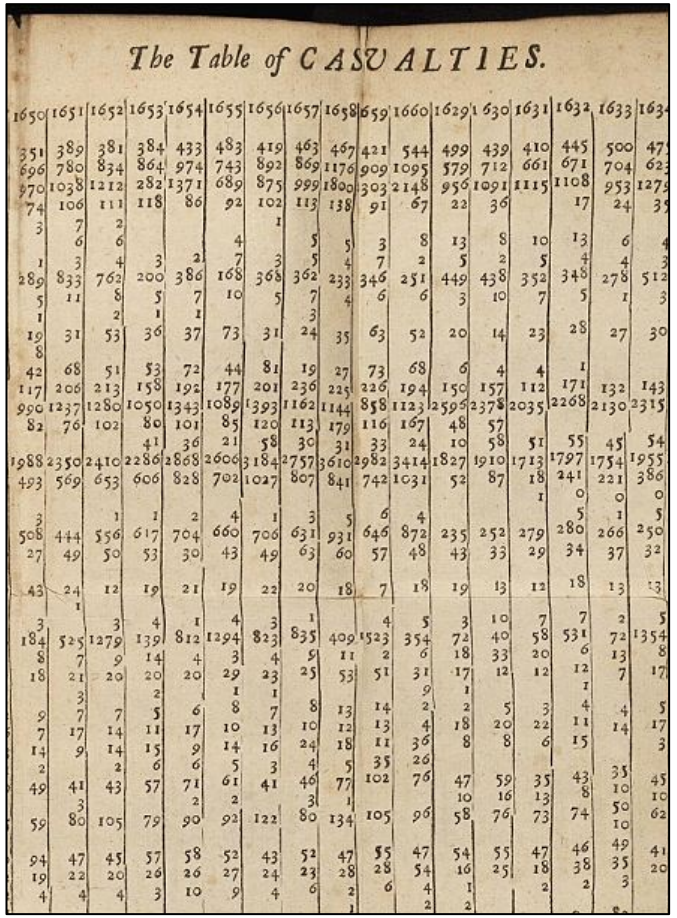
Definition:

data := “units of information that are collected through observation”

(wikipedia)

1. What is data?

The Ishango bone, thought to be an early form of tallying (C 18,000 BCE)



The Table of CASUALTIES.

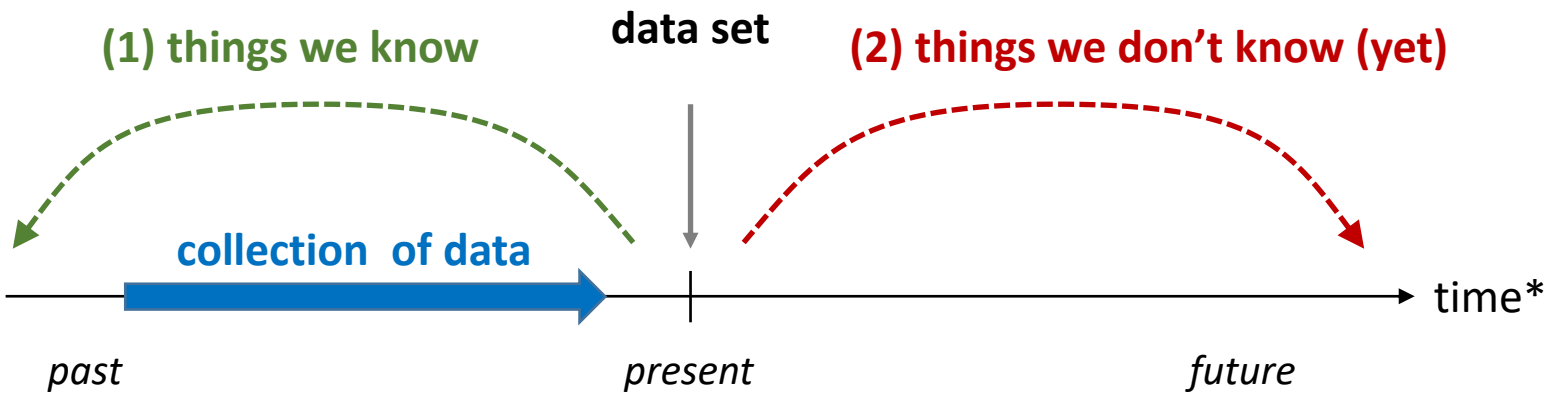
1650	1651	1652	1653	1654	1655	1656	1657	1658	1659	1660	1629	1630	1631	1632	1633	1634
351	389	381	384	433	483	419	463	467	421	544	499	439	410	445	500	475
696	780	834	864	974	743	892	869	1176	909	1095	979	712	661	671	704	623
970	1038	1212	282	1371	689	875	999	1800	303	2148	956	1091	1115	1108	953	1275
74	106	111	118	86	92	102	113	138	91	67	22	36		17	24	35
3	7	2				1										
	6	6			4		5	5	3	8	13	8	10	13	6	4
1	3	4	3	2	7	3	5	4	7	2	5	2	5	4	4	3
289	833	762	200	386	168	368	362	233	346	251	449	438	352	348	278	512
5	11	8	5	7	10	5	7	4	6	6	3	10	7	5	1	3
1		2	1	1			3									
19	31	53	36	37	73	31	24	35	63	52	20	14	23	28	27	30
8																
42	68	51	53	72	44	81	19	27	73	68	6	4	4	1		
117	206	213	158	192	177	201	236	225	226	194	150	157	112	171	132	143
990	1237	1280	1050	1343	1089	1393	1162	1144	858	1123	2596	2378	2035	2268	2130	2315
82	76	102	80	101	85	120	113	179	116	167	48	57				
			41	36	21	58	30	31	33	24	10	58	51	55	45	54
1988	2350	2410	2286	2868	2606	3184	2757	3610	2982	3414	1827	1910	1713	1797	1754	1955
493	569	653	606	828	702	1027	807	841	742	1031	52	87	18	241	221	386
													1	0	0	0
3		1	1	2	4	1	3	5	6	4				5	1	5
508	444	556	617	704	660	706	631	931	646	872	235	252	279	280	266	250
27	49	50	53	30	43	49	63	60	57	48	43	33	29	34	37	32
43	24	12	19	21	19	22	20	18	7	18	19	13	12	18	13	13
	1															
3		3	4	1	4	3	1		4	5	3	10	7	7	2	5
184	525	1279	139	812	1294	823	835	409	1523	354	72	40	58	531	72	1354
8	7	9	14	4	3	4	5	11	2	6	18	33	20	6	13	8
18	21	20	20	20	29	23	25	53	51	31	17	12	12	12	7	17
	3		2		1	1				9	1			1		
9	7	7	5	6	8	7	8	13	14	2	2	5	3	4	4	5
7	17	14	11	17	10	13	10	12	13	4	18	20	22	11	14	17
14	9	14	15	9	14	16	24	18	11	36	8	8	6	15		3
2		2	6	6	5	3	4	5	35	26					33	
49	41	43	57	71	61	41	46	77	102	76	47	59	35	43	10	45
	3			2	2		3	1			10	16	13	8	50	10
59	80	105	79	90	92	122	80	134	105	96	58	76	73	74	10	62
94	47	45	57	58	52	43	52	47	55	47	54	55	47	46	49	41
19	22	20	26	26	27	24	23	28	28	54	16	25	18	38	35	20
4	4	4	3	10	9	4	6	2	6	4	1		2	2	3	
										2	2					

"Natural and Political Observations on the Bills of Mortality" (John Graunt, 1662)

Modern data center (C 2021)

2. Why do we need data?

- Broadly, data can help us in two ways:



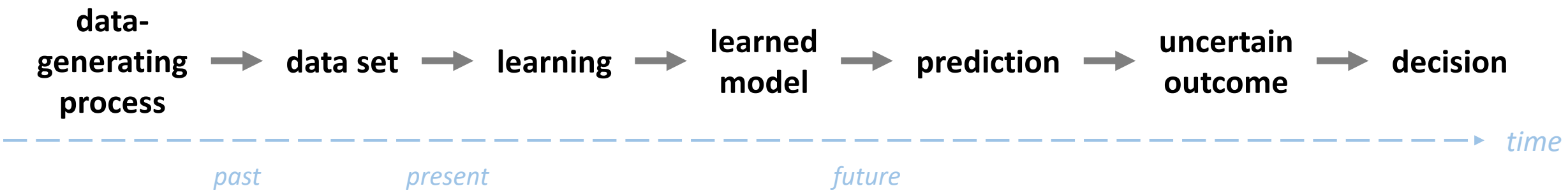
- Machine learning is especially useful for the latter.
- Can we always use the past to infer the future? Of course not.
- But under some conditions – yes, and for this, we need **assumptions**.
- We'll try to be precise about the assumptions we make (beware: not everyone is).

> *"It is difficult to make predictions, especially about the future"*

* *time* considers not when things happened, but rather, when we ask the question

2. Why do we need data?

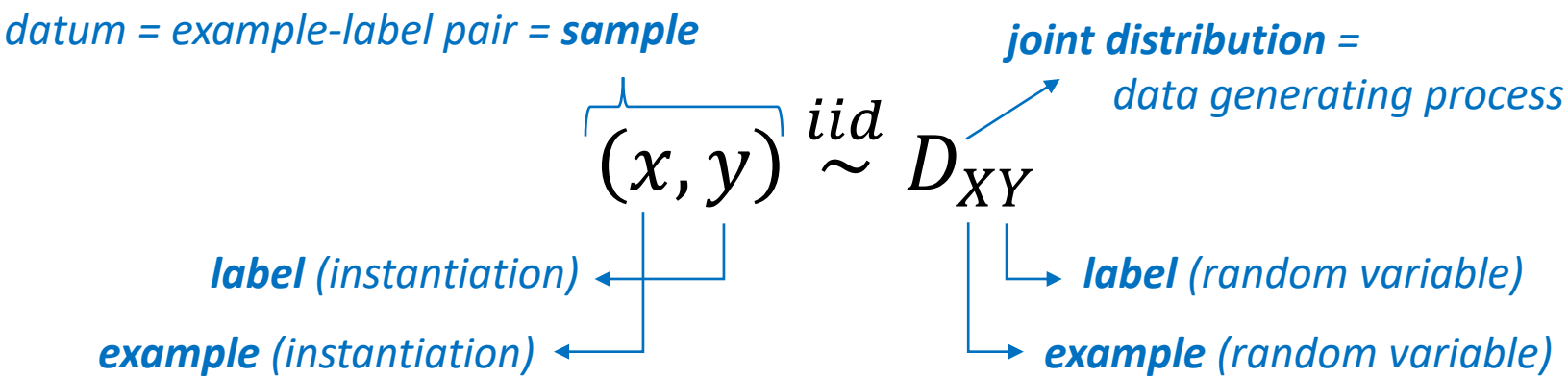
- So how will we use data?



- The main source of difficulty: **uncertainty**
- Learning **reduces uncertainty** by making use of **data**
- To reason about uncertainty, we will use **probability theory**

3. How do we use “data”?

- Probabilistic modeling lies at the heart of machine learning
- **Key assumption**: data is sampled from a fixed, unknown underlying distribution*



- This will be our *data generating process*.
- We will consider the simplest form of sampling: **i.i.d.**
- This is a **big assumption** – seldom realistic, but very effective.

* This will be our main assumption in most learning problems we will encounter

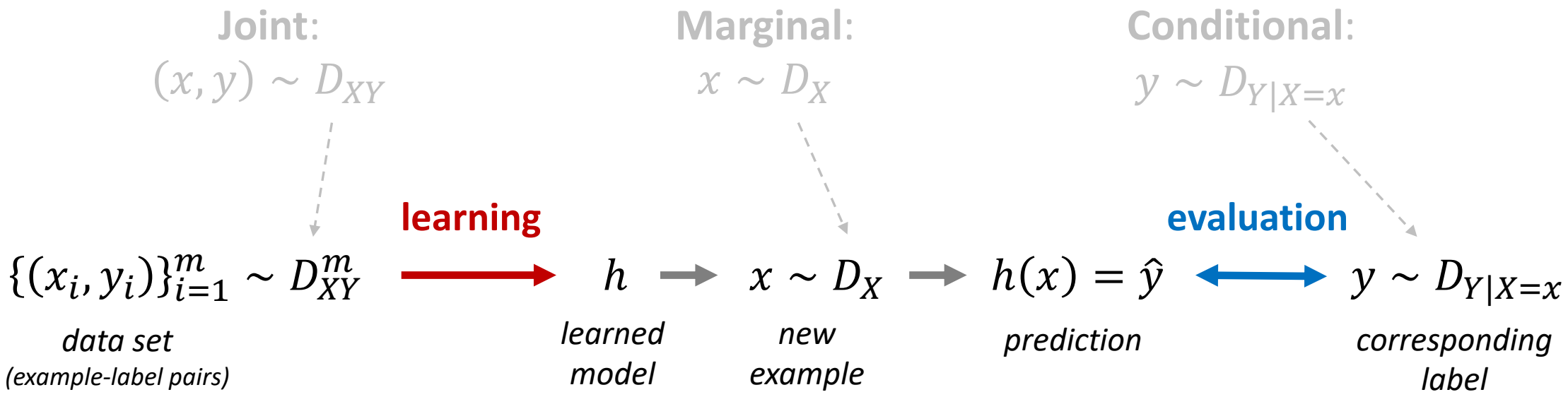
3. How do we use “data”?

Joint:
$(x, y) \sim D_{XY}$
(sample input-label pair)

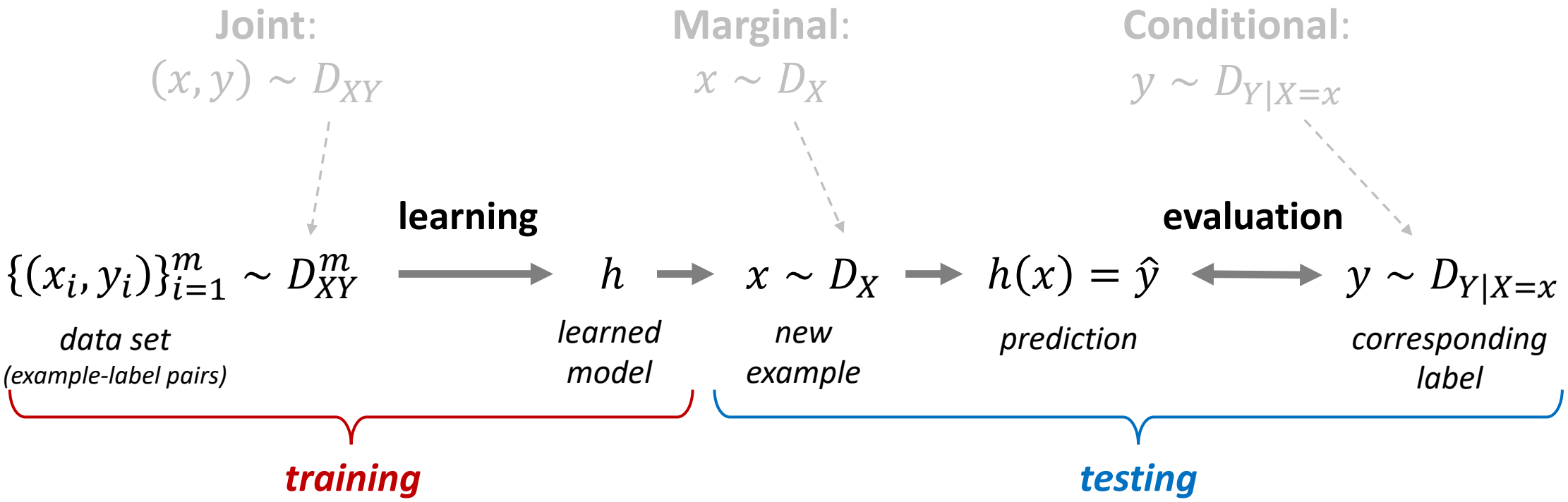
Marginal:
$x \sim D_X$
(sample input)
data uncertainty

Conditional:
$y \sim D_{Y|X=x}$
(sample label for given input)
label uncertaint(ies)

3. How do we use “data”?



3. How do we use “data”?



- Probability theory lets us compare *training time* to *test time*.
- Comparison is **sound** due to *chain rule*: $P(x, y) = P(x)P(y|x)$

3. How do we use “data”?

Joint:
$(x, y) \sim D_{XY}$
(sample input-label pair)

Marginal:
$x \sim D_X$
(sample input)
data uncertainty

Conditional:
$y \sim D_{Y|X=x}$
(sample label for given input)
label uncertaint(ies)

- Convenient abstraction of messy real-world data.
- Learning when examples *and* labels are observed is called *supervised learning*.
- Tells us what we observe – but not what we don’t.
- Remember that we never know the true D_{XY}! All we see are samples.

3. How do we use “data”?

Joint:
$(x, y) \sim D_{XY}$
(sample input-label pair)

Marginal:
$x \sim D_X$
(sample input)
data uncertainty

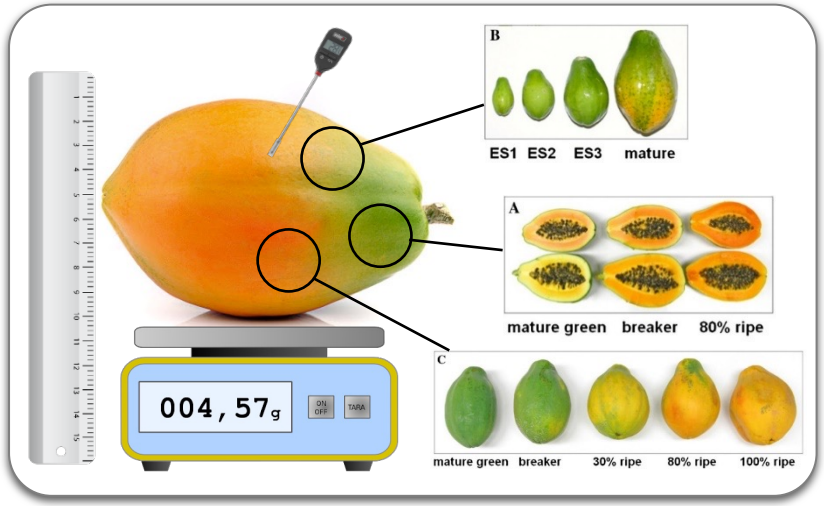
Conditional:
$y \sim D_{Y|X=x}$
(sample label for given input)
label uncertaint(ies)

- Most of our efforts will focus on reducing this form of uncertainty
- Throughout the course we will mostly consider *binary classification*: $y \in \{\pm 1\}$
- Labels y are distributed, but often we want accurate *deterministic* predictions
- This means we can, but don’t have to, model $P(Y|X = x)$ directly

3. How do we use “data”?

Joint:
$(x, y) \sim D_{XY}$
(sample input-label pair)

Marginal:
$x \sim D_X$
(sample input)
data uncertainty



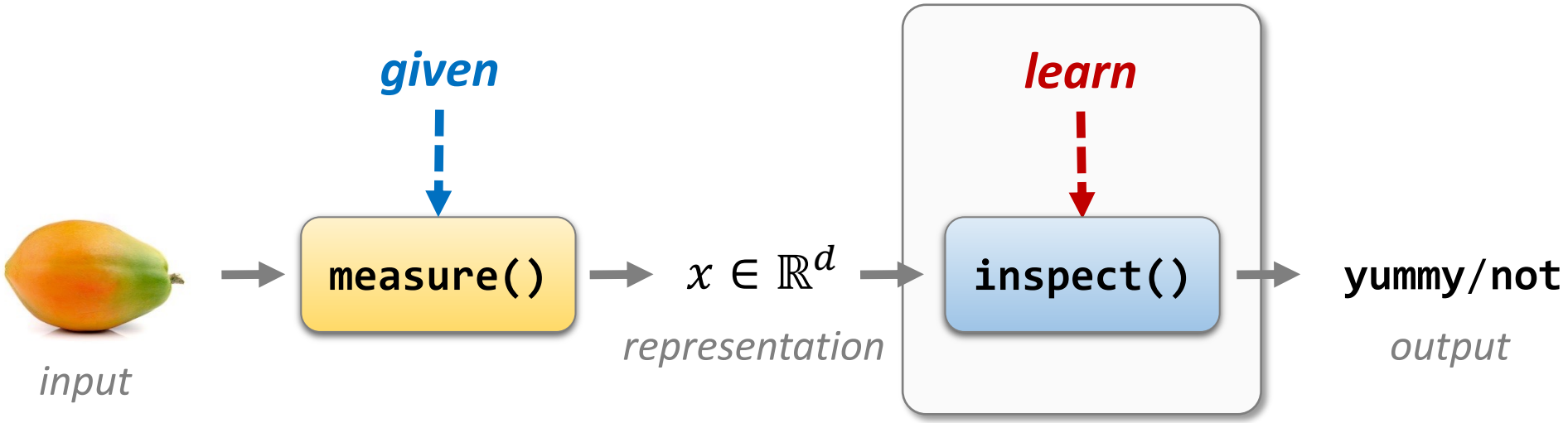
- What does it mean to “sample a papaya”?
- Think of x as *some representation* of underlying objects
- Conventional representation: numerical *feature vectors*
- Think of each x_i as a *measurement* or an *attribute*
- Modern representations: complex “natural” modalities *(images, text, sound, graphs, …)*

`measure()`

$x = (height, weight, color) \in \mathbb{R}^3$

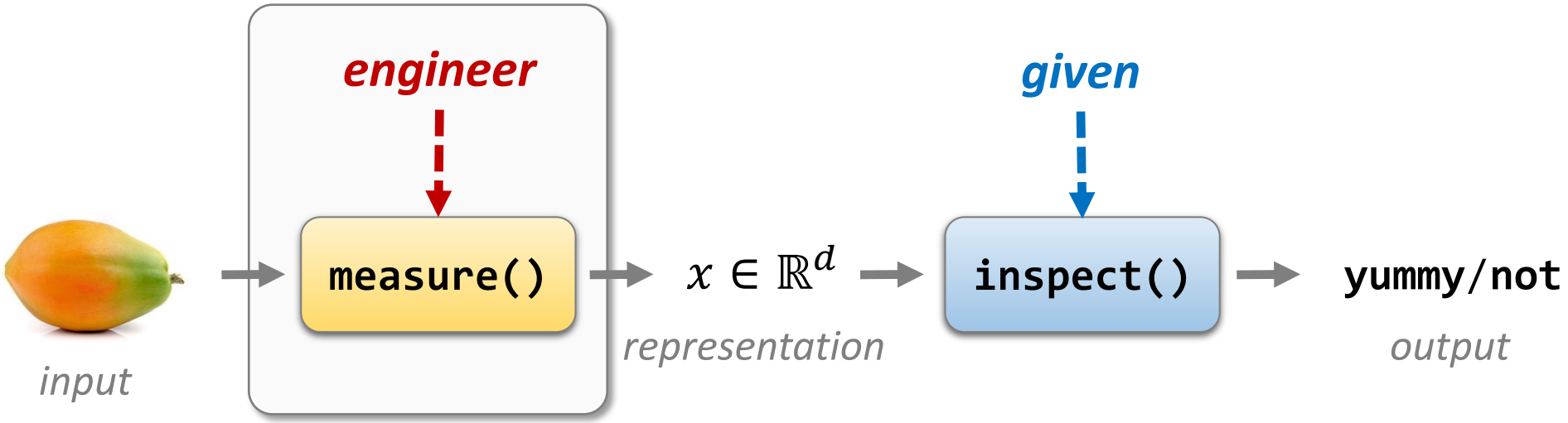
Data and us

- In class, we will mostly consider **representations as given** (and focus on *learning*)



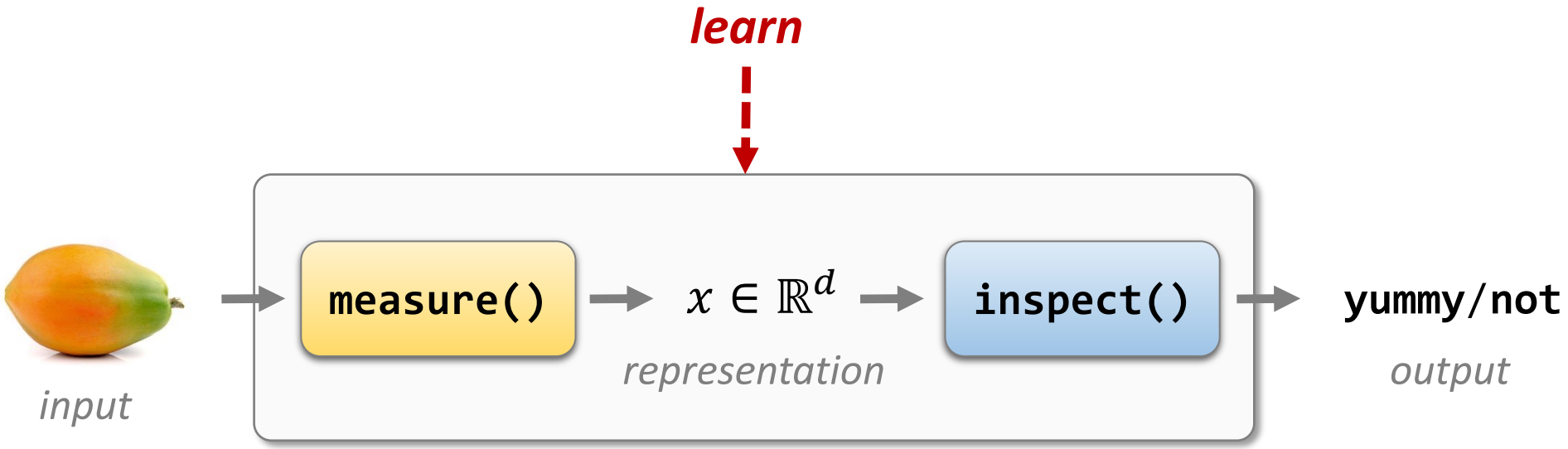
Data and us

- In class, we will mostly consider **representations as given** (and focus on *learning*)
- But successful learning often requires good hand-coded representations.
- In some exs you will work on **engineering representations**.



Data and us

- In class, we will mostly consider **representations as given** (and focus on *learning*)
- But successful learning often requires good hand-coded representations.
- In some exs you will work on **engineering representations**.
- Towards the end of our course, we will see how to **learn representations.**



Putting it all together:
A formal learning framework

Towards a learning algorithm

Notations:

- Features: $x \in \mathbb{R}^d$
- Labels: $y \in \{\pm 1\}$ *(classification)*
- Data dist.: $(x, y) \overset{iid}{\sim} D = D_{XY}$ *(supervised)*

Towards a learning algorithm

Notations:

- Features: $x \in \mathbb{R}^d$
- Labels: $y \in \{\pm 1\}$ *(classification)*
- Data dist.: $(x, y) \overset{iid}{\sim} D = D_{XY}$ *(supervised)*

- Model: h: $\mathbb{R}^d \to \{\pm 1\}$ *(classifier, hypothesis, …)*

- Expected error: $\mathbb{E}_{(x,y)\sim D}[\mathbb{1}\{y \neq h(x)\}]$

The learning problem:

- **Goal:** find a classifier $h(x)$ that is **accurate**
- We will define "accuracy" as:

$$\text{acc}(h) = \mathbb{P}_{(x,y)\sim D}[y = h(x)]$$

- We want to maximize accuracy
- **Equivalent goal**: minimize (expected) error

$$\begin{aligned}\text{err}(h) &= 1 - \text{acc}(h) \\ &= \mathbb{P}_{(x,y)\sim D}[y \neq h(x)] \\ &= \mathbb{E}_{(x,y)\sim D}[\mathbb{1}\{y \neq h(x)\}]\end{aligned}$$

= 0/1 loss

➢ Let's re-write this as an algorithmic problem

Towards a learning algorithm

Notations:

- Features: $x \in \mathbb{R}^d$
- Labels: $y \in \{\pm 1\}$ *(classification)*
- Data dist.: $(x, y) \overset{iid}{\sim} D = D_{XY}$ *(supervised)*
- Sample set: $S = \{(x_i, y_i)\}_{i=1}^m \sim D^m$
- Model: h: $\mathbb{R}^d \to \{\pm 1\}$ *(classifier, hypothesis, …)*
- Model class: $H = \{h_1, h_2, \dots\}$
- Expected error: $\mathbb{E}_{(x,y)\sim D}[\mathbb{1}\{y \neq h(x)\}]$

Template algorithm:

- **Input**:
  - Sample set $S = \{(x_i, y_i)\}_{i=1}^m$
  - Model class H *(candidate classifiers)*
- **Output**:
  classifier with lowest expected error

$$h^* = \underset{h \in H}{\operatorname{argmin}} \mathbb{E}_{(x,y)\sim D}[\mathbb{1}\{y \neq h(x)\}]$$

actually "$\in$"

= "learning objective"

Towards a learning algorithm

- **Naïve solution:** just compute h^*! (?)
- What's stopping us?
- **Fundamental problem**: D is unknown

Template algorithm:

- **Input**:
  - Sample set $S = \{(x_i, y_i)\}_{i=1}^{m}$
  - Model class H *(candidate classifiers)*
- **Output**:
  classifier with lowest expected error
  $h^* = \underset{h \in H}{\operatorname{argmin}}\, \mathbb{E}_{(x,y)\sim D}[\mathbb{1}\{y \neq h(x)\}]$

Towards a learning algorithm

- **Naïve solution:** just compute h^*! (?)
- What's stopping us?
- **Fundamental problem**: D is unknown
- One possible approach:

Empirical Risk Minimization (ERM)

- This is an *algorithmic problem* that entails *algorithmic questions.*
- But learning algorithms require us to ask additional questions.

ERM: (Empirical Risk Minimization)

- **Input**:
  - Sample set $S = \{(x_i, y_i)\}_{i=1}^m$
  - Model class H *(candidate classifiers)*
- **Objective**:
  classifier with lowest expected error
  $$h^* = \operatorname{argmin}_{h \in H} \mathbb{E}_{(x,y)\sim D}[\mathbb{1}\{y \neq h(x)\}]$$
- **Return**:
  classifier with lowest empirical error
  $$\hat{h} = \operatorname{argmin}_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\{y_i \neq h(x_i)\}$$

Towards a learning algorithm

Questions we should ask:

- **Algorithmic** (=optimization)
  - When can we find the argmin efficiently?
  - How much time/space does this require?
  - What are good approximation schemes?

ERM: (Empirical Risk Minimization)

- **Input**:
  - Sample set $S = \{(x_i, y_i)\}_{i=1}^{m}$
  - Model class H *(candidate classifiers)*
- **Objective:**
  classifier with lowest expected error
  $$h^* = \operatorname*{argmin}_{h \in H} \mathbb{E}_{(x,y)\sim D}[\mathbb{1}\{y \neq h(x)\}]$$
- **Return**:
  classifier with lowest empirical error
  $$\hat{h} = \operatorname*{argmin}_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\{y_i \neq h(x_i)\}$$

Towards a learning algorithm

Questions we should ask:

- **Algorithmic** (=optimization)
  - When can we find the argmin efficiently?
  - How much time/space does this require?
  - What are good approximation schemes?
- **Statistical** (ERM is *means*, not *ends*!)
  - When does low empirical error imply low expected error?
  - How many samples does this require?
  - Which model classes can be learned?

ERM: (Empirical Risk Minimization)

- **Input**:
  - Sample set $S = \{(x_i, y_i)\}_{i=1}^{m}$
  - Model class H *(candidate classifiers)*
- **Objective:**
  classifier with lowest expected error
  $$h^* = \operatorname*{argmin}_{h \in H} \mathbb{E}_{(x,y)\sim D}[\mathbb{1}\{y \neq h(x)\}]$$
- **Return**:
  classifier with lowest empirical error
  $$\hat{h} = \operatorname*{argmin}_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\{y_i \neq h(x_i)\}$$

Towards a learning algorithm

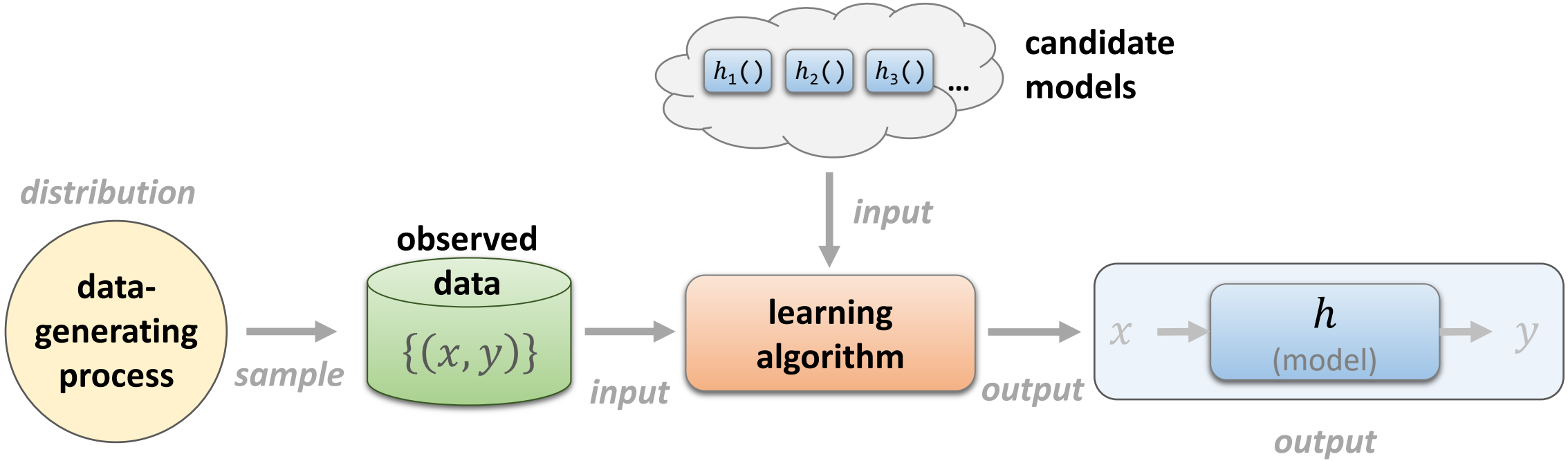
Questions we should ask:

- **Algorithmic** (=optimization)
  - When can we find the argmin efficiently?
  - How much time/space does this require?
  - What are good approximation schemes?
- **Statistical** (ERM is *means*, not *ends*!)
  - When does low empirical error imply low expected error?
  - How many samples does this require?
  - Which model classes can be learned?
- **Modeling**
  - How can we use our domain knowledge?
  - For a given task, what are good models?
  - Are we assuming too much? Not enough?

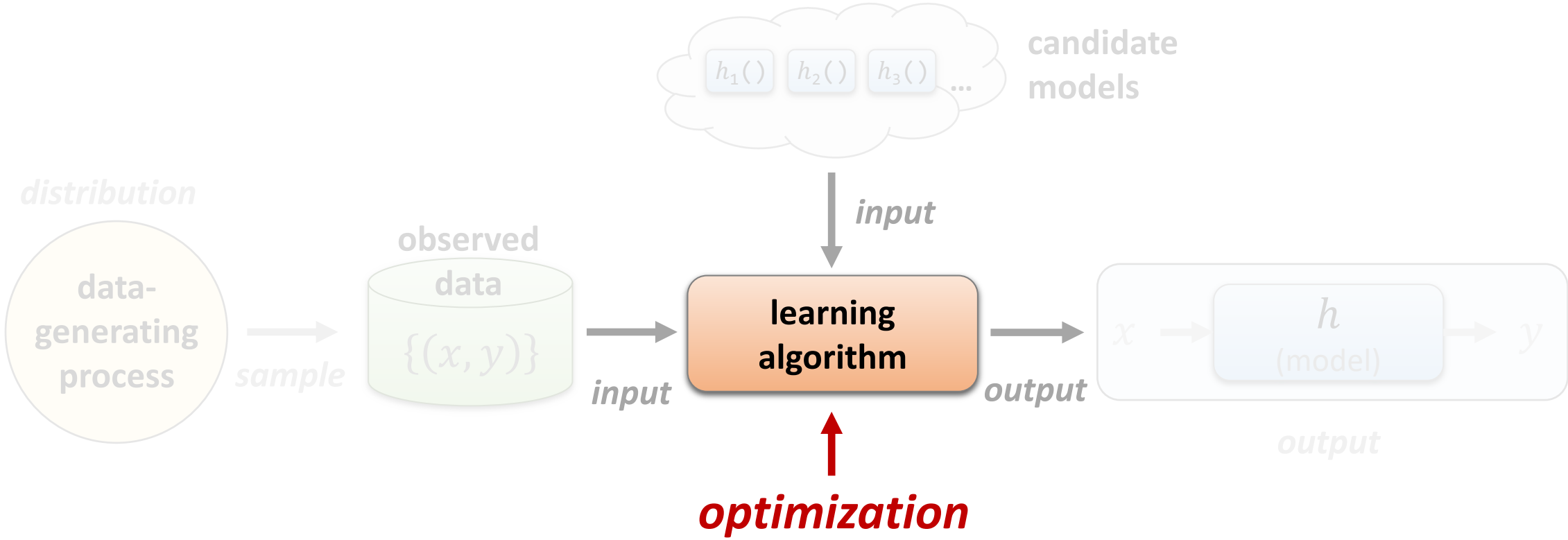
ERM: (Empirical Risk Minimization)

- **Input**:
  - Sample set $S = \{(x_i, y_i)\}_{i=1}^m$
  - Model class H *(candidate classifiers)*
- **Objective:**
  classifier with lowest expected error
  $$h^* = \operatorname*{argmin}_{h \in H} \mathbb{E}_{(x,y)\sim D}[\mathbb{1}\{y \neq h(x)\}]$$
- **Return**:
  classifier with lowest empirical error
  $$\hat{h} = \operatorname*{argmin}_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\{y_i \neq h(x_i)\}$$

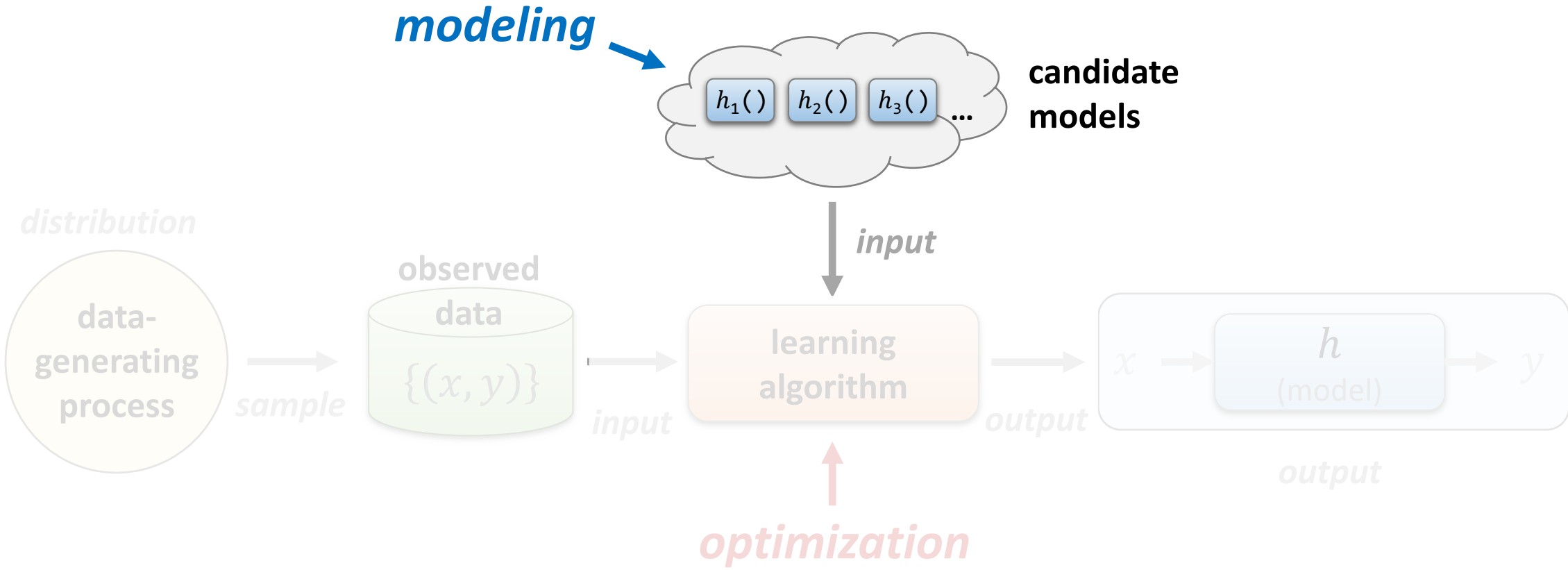
Recap:


$h_1()$
$h_2()$
$h_3()$
...
candidate models
distribution
data-generating process
sample
observed data
$\{(x, y)\}$
input
input
learning algorithm
output
x
h
(model)
y
output

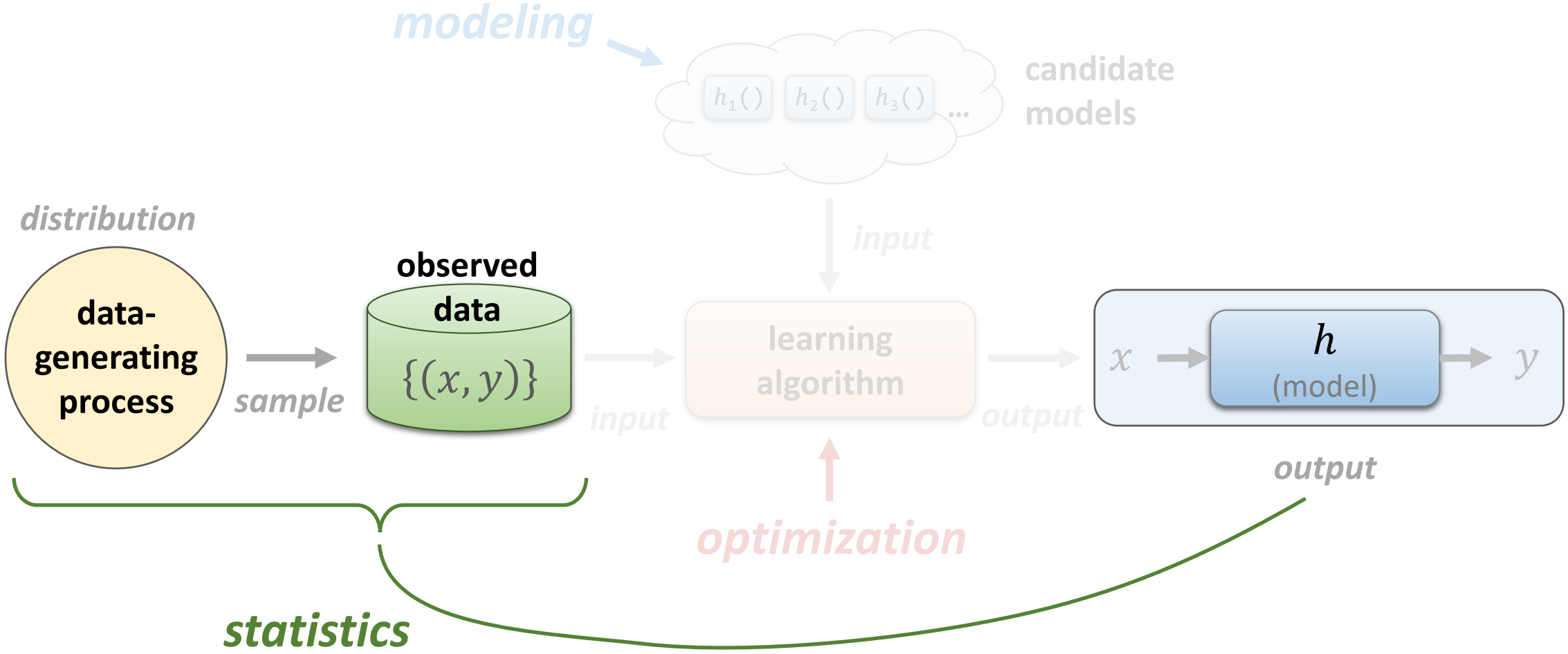
Recap:


$h_1()$
$h_2()$
$h_3()$
...
candidate models
input
distribution
observed data
data-generating process
sample
$\{(x, y)\}$
learning algorithm
input
output
x
h
(model)
y
output
optimization

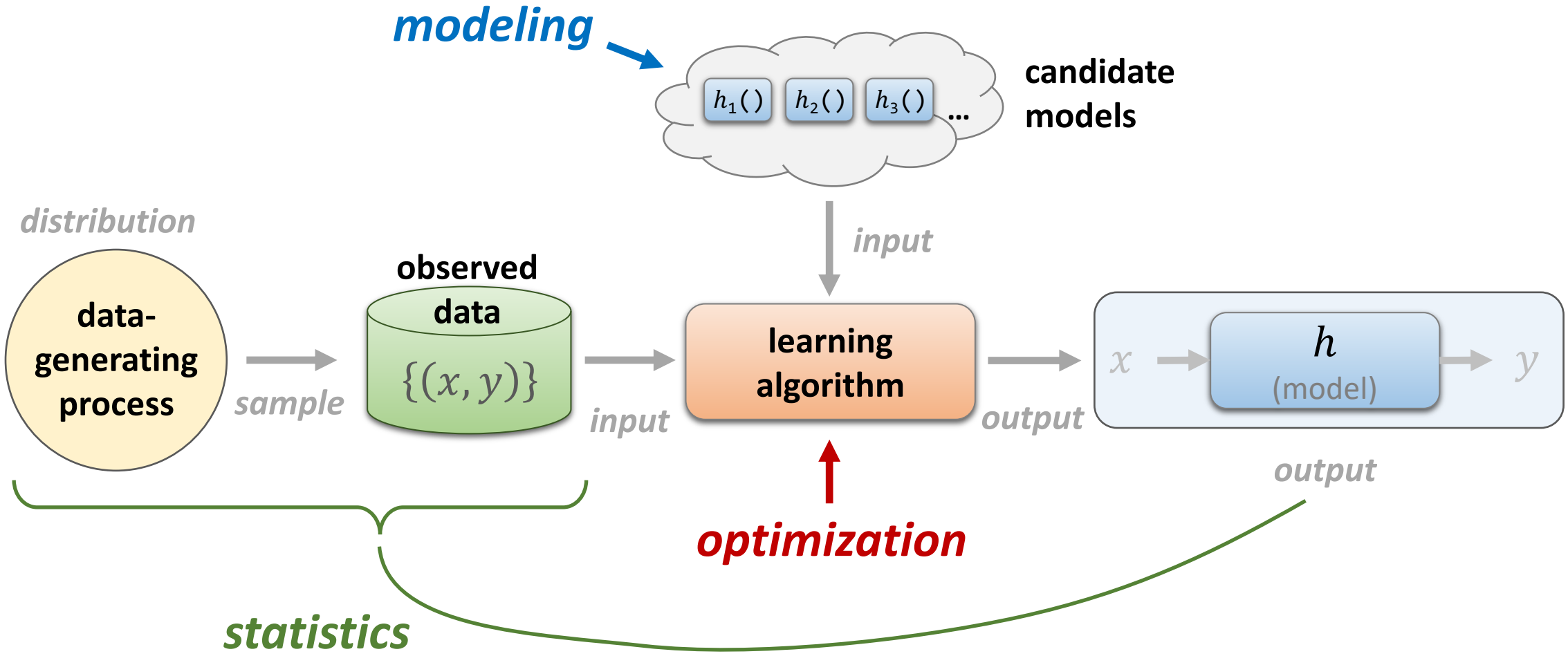
Recap:


modeling
$h_1()$
$h_2()$
$h_3()$
...
candidate models
input
distribution
data-generating process
sample
observed data
$\{(x, y)\}$
input
learning algorithm
output
x
h
(model)
y
output
optimization

Recap:

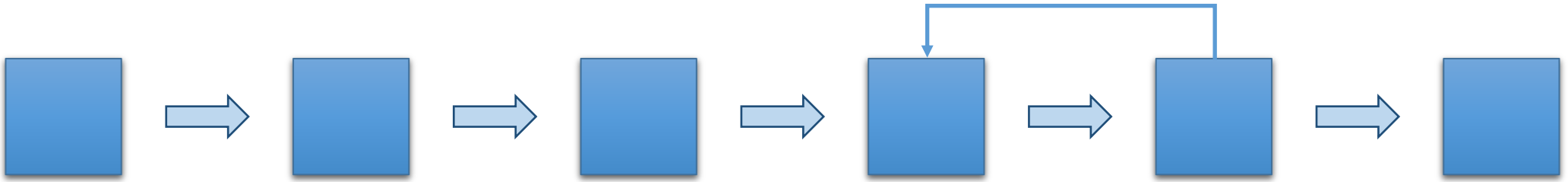

modeling
$h_1()$ $h_2()$ $h_3()$ ...
candidate models
input
distribution
data-generating process
sample
observed data
$\{(x, y)\}$
input
learning algorithm
output
x
h
(model)
y
output
optimization
statistics

Recap:


modeling
$h_1()$ $h_2()$ $h_3()$ ...
candidate models
input
distribution
observed data
$\{(x, y)\}$
data-generating process
sample
input
learning algorithm
output
x
h
(model)
y
output
optimization
statistics

The bigger picture

the learning pipeline



data collection
inputs x
labels y

preprocessing
clean
augment
select
transform

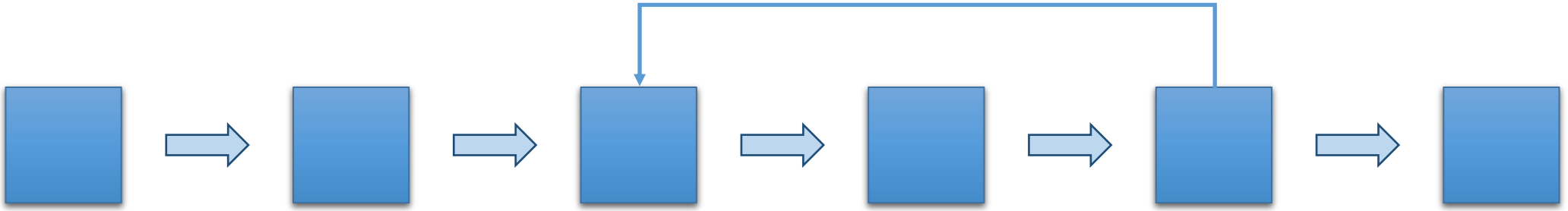
setup
model class
metrics
learning algorithm

learning
training
validation

evaluation
analysis
visualization

deployment
predictions
decisions
policy

the learning pipeline



data collection
inputs x
labels y

preprocessing
clean
augment
select
transform

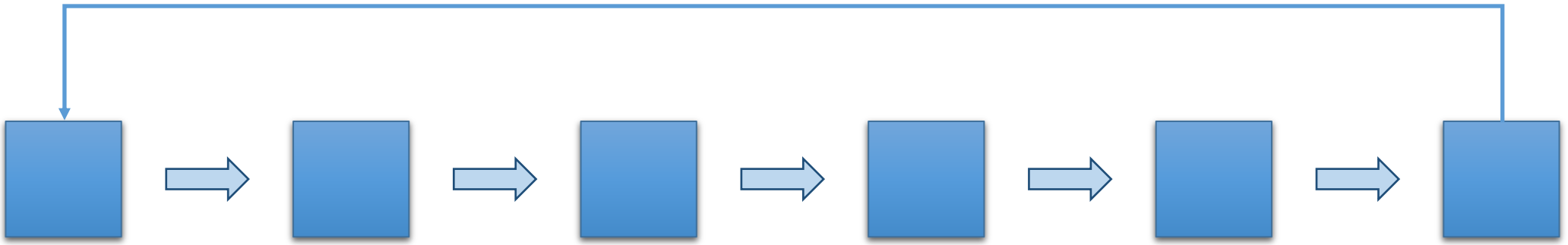
setup
model class
metrics
learning algorithm

learning
training
validation

evaluation
analysis
visualization

deployment
predictions
decisions
policy

the learning pipeline



data collection
inputs x
labels y

preprocessing
clean
augment
select
transform

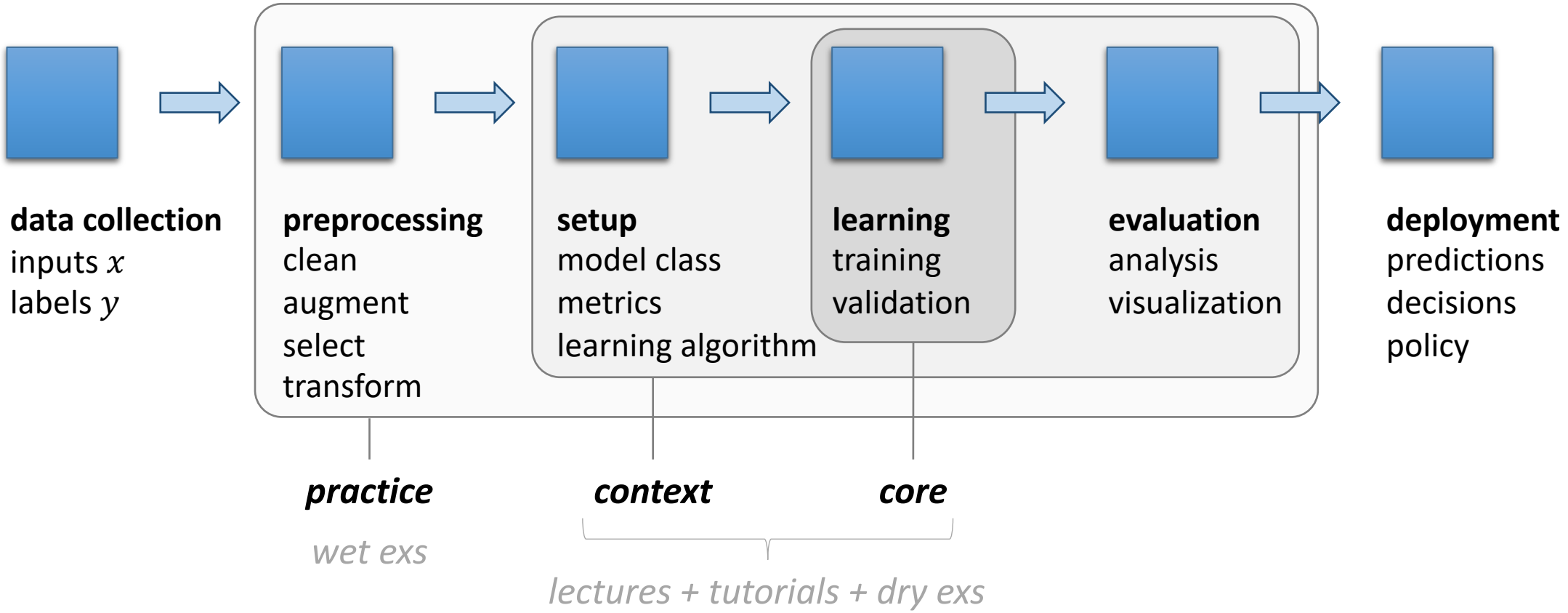
setup
model class
metrics
learning algorithm

learning
training
validation

evaluation
analysis
visualization

deployment
predictions
decisions
policy

the learning pipeline



types of learning

input:	**output:**	**supervision:**	**approach:**	**environment:**
feature vectors	binary	supervised	discriminative	batch learning

types of learning

input:
feature vectors
images
text
speech
behavior
graphs
time-series
…

output:
binary
multiclass
multilabel
structured
scalars
intervals
distributions
…

supervision:
supervised
unsupervised
semi-supervised
weakly supervised
zero-shot
…

approach:
discriminative
generative
hybrid
…

environment:
batch learning
online learning
reinforcement learning
active learning
metric learning
multi-task learning
learning to teach
meta-learning
…

Plan for semester

- **Part I:** Supervised binary classification
- **Part II:** Aspects of learning
- **Part III:** Supervised learning: revisited and expanded

Classification

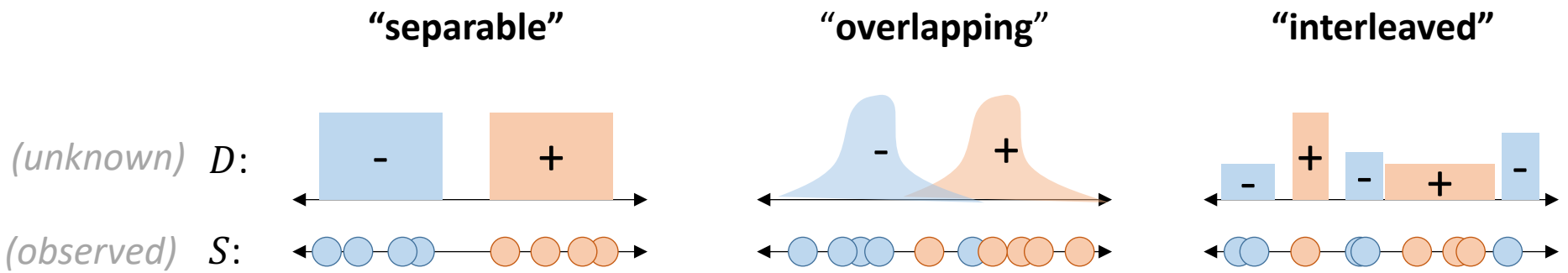
Recall:

- **Data generating process**: $(x, y) \overset{iid}{\sim} D,\ y \in \{\pm 1\}$ (classification)
- **Sample set**: $S = \{(x_i, y_i)\}_{i=1}^{m} \sim D^m$ (data)
- **Model class:** $H = \{h : h: \mathcal{X} \to \mathcal{Y}\}$
- **Expected error**: $L_D(h) = \mathbb{P}_D[y \neq h(x)] = \mathbb{E}_D[\mathbb{1}\{y \neq h(x)\}]$
- **Empirical error**: $L_S(h) = \mathbb{P}_S[y \neq h(x)] = \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}\{y_i \neq h(x_i)\}$
- **Goal**: learn $h \in H$ with low $L_D(h)$
- **Means**: return $\hat{h} = \underset{h \in H}{\operatorname{argmin}}\, L_S(h)$ (ERM)

Classification

Example:

- Learning on the real line, $\mathcal{X} = \mathbb{R}$
- Consider three "types" of distributions of increasing difficulty:



- For each, we will discuss the **three aspects of learning** we care about:

Modeling, Optimization, Statistics

Case I: Separable Data

Separable data

- **Definition (interim)***:

> D (over $\mathbb{R}$) is *separable* if exists $\theta^* \in \mathbb{R}$ such that $y = 1$ iff $x > \theta^*$

- Note this means labels are deterministic.

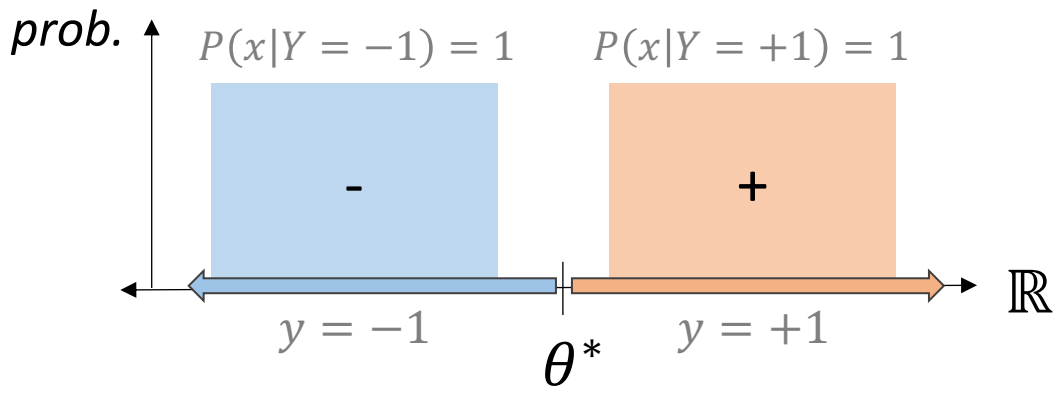
* This definition is not precise, and is in fact class-dependent. This will later be refined.

Separable data

- **Definition (interim)***:

> D (over $\mathbb{R}$) is *separable* if exists $\theta^* \in \mathbb{R}$ such that $y = 1$ iff $x > \theta^*$

- Note this means labels are deterministic.
- Example distribution:



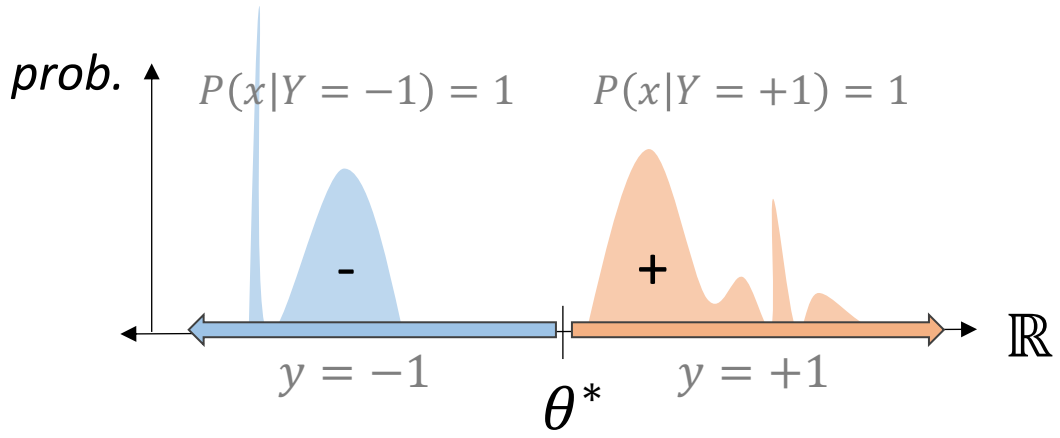
* This definition is not precise, and will later be refined.

Separable data

- **Definition (interim)***:

> D (over $\mathbb{R}$) is *separable* if exists $\theta^* \in \mathbb{R}$ such that $y = 1$ iff $x > \theta^*$

- Note this means labels are deterministic.
- Example distribution:



* This definition is not precise, and will later be refined.

Modeling

- Time to choose a model class!
- **Think** – what are good criteria for choosing?
- Let's try *threshold functions*:

$$H_{\text{thresh}} = \{h_\theta(x) = \text{sign}(x \geq \theta) : \theta \in \mathbb{R}\}$$

- **Q**: Why is this a good class?
- (**Remember**: we're assuming separability, but not any specific distribution)
- **A**: Because it's "just right" – intuitively:
  1. For any separable D there is some $h \in H$ with $L_S(h) = 0$
  2. And, it's not too large – no "redundant" h-s
- **Note:** $|H| = \infty$, suggesting that "large" is not the right measure (we'll return to this)

Optimization

- Recall the ERM template
- To actually learn, need to implement for H_{thresh}
- Plug $h_\theta(x) = \text{sign}(x \geq \theta)$ into formulas

ERM: (Empirical Risk Minimization)

- **Input**:
  - Sample set $S = \{(x_i, y_i)\}_{i=1}^m$
  - Model class H *(candidate classifiers)*
- **Objective**:
  classifier with lowest expected error
  $$h^* = \underset{h \in H}{\text{argmin}}\, \mathbb{E}_{(x,y)\sim D}[\mathbb{1}\{y \neq h(x)\}]$$
- **Return**:
  classifier with lowest empirical error
  $$\hat{h} = \underset{h \in H}{\text{argmin}} \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\{y_i \neq h(x_i)\}$$

Optimization

- Recall the ERM template
- To actually learn, need to implement for H_{thresh}
- Plug $h_\theta(x) = \text{sign}(x \geq \theta)$ into formulas
- We will call θ the **parameter**(s) of h_θ
- Optimizing over $h \equiv$ optimizing over θ
- This is an example of model class *structure*, which we will see is helpful (and necessary)
- Can we solve ERM now?

ERM: (Empirical Risk Minimization)

- **Input**:
  - Sample set $S = \{(x_i, y_i)\}_{i=1}^m$
  - Model class H *(candidate classifiers)*
- **Objective:**
  classifier with lowest expected error
  $$\theta^* = \underset{\theta \in \mathbb{R}}{\text{argmin}}\, \mathbb{E}_{(x,y)\sim D}[\mathbb{1}\{y \neq \text{sign}(x \geq \theta)\}]$$
- **Return**:
  classifier with lowest empirical error
  $$\hat{\theta} = \underset{\theta \in \mathbb{R}}{\text{argmin}} \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\{y_i \neq \text{sign}(x_i \geq \theta)\}$$

Optimization

Simple algorithm:

1. Sort x_i-s
1b. Validate separability
2. Find largest x_i with $y_i = -1$; call it x_{i^*}
3. Return $\hat{\theta} = x_{i^*}$

- **Congrats**! We just implemented our first ERM
- What do we need to prove?
- (Remember the algorithm is designed for *threshold classifiers* on *separable data)*

ERM: (Empirical Risk Minimization)

- **Input**:
  - Sample set $S = \{(x_i, y_i)\}_{i=1}^m$
  - Model class H *(candidate classifiers)*
- **Objective:**
  classifier with lowest expected error
  $\theta^* = \underset{\theta \in \mathbb{R}}{\operatorname{argmin}} \mathbb{E}_{(x,y)\sim D}[\mathbb{1}\{y \neq \operatorname{sign}(x \geq \theta)\}]$
- **Return**:
  classifier with lowest empirical error
  $\hat{\theta} = \underset{\theta \in \mathbb{R}}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\{y_i \neq \operatorname{sign}(x_i \geq \theta)\}$

Optimization

Simple algorithm:

1. Sort x_i-s $O(m \log m)$
1b. Validate separability $O(m)$
2. Find largest x_i with $y_i = -1$; call it x_{i^*} $O(m)$
3. Return $\hat{\theta} = x_{i^*}$

- **Congrats**! We just implemented our first ERM
- What do we need to prove?
- (Remember the algorithm is designed for *threshold classifiers* on *separable data)*

1. **Runtime**: $O(m \log m)$ – great!
- Recall $m = |S|$
- But H is also "input", and $|H| = \infty$! *How can this be?*

2. **Correctness**: $L_S(h_{\hat{\theta}}) = 0$
- Easy to show.
- Not only solution (what are others?)

3. ***Generalization***: $L_D(h_{\hat{\theta}}) = ?$
- How can we relate L_S and L_D?
- Enter *statistics.*