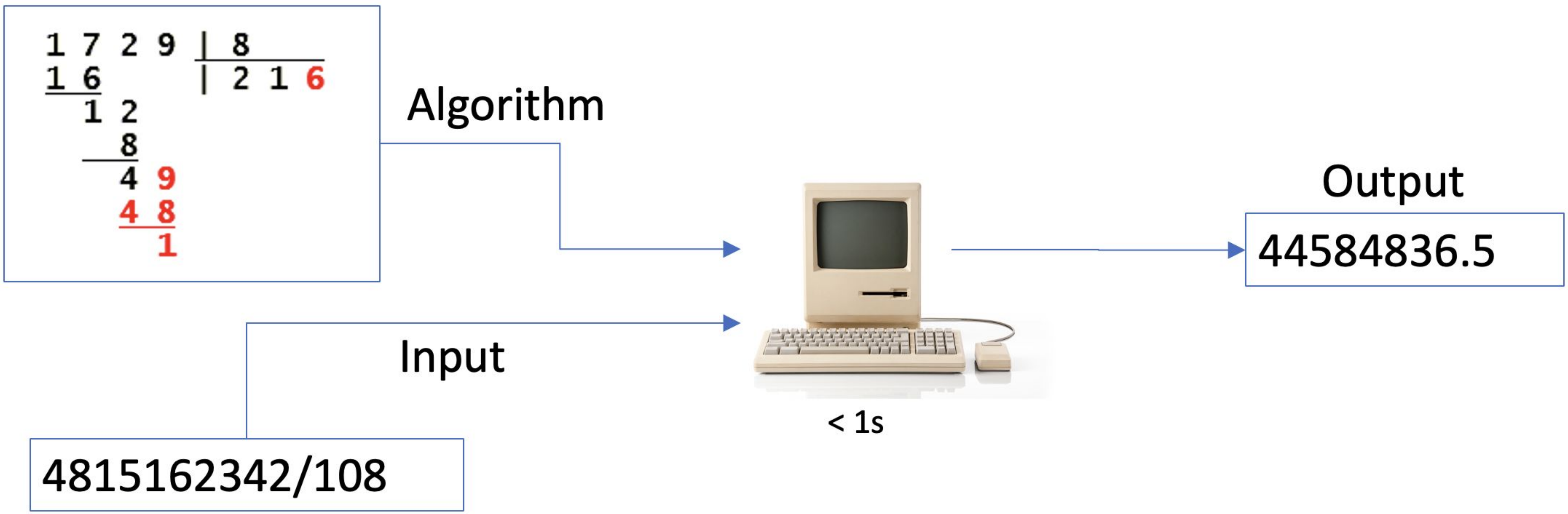
Introduction to Machine Learning Basics

Part 1:
Shifting the paradigm

What is Machine Learning?

- Algorithm: a clear and unambiguous description of a set of steps for solving a problem



A concrete Example

- Can you guess the difference between 2 classes of images:
  - Chartreux
  - Persian

A concrete Example

- Can you guess the difference between 2 classes of images:
  - Chartreux
  - Persian

A concrete Example

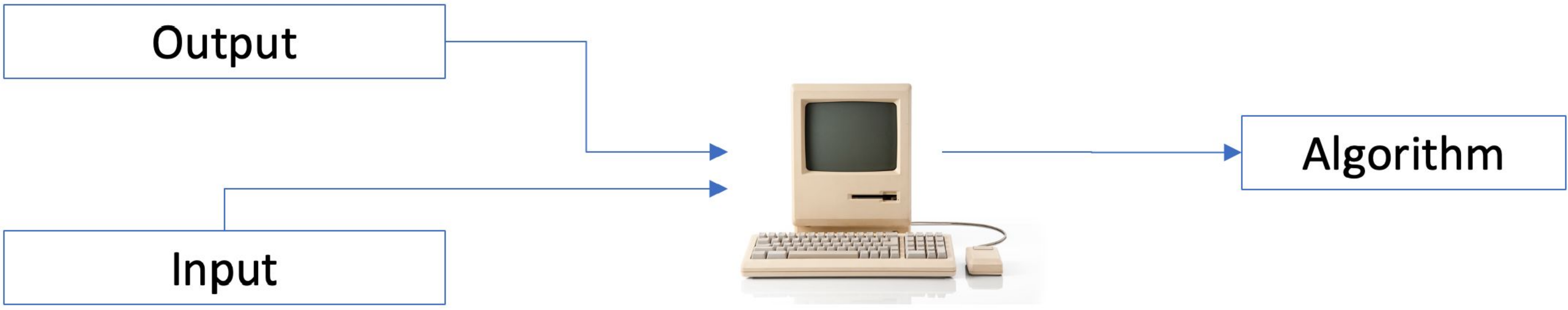
- With some examples “labeled” you can solve the task with an high performance
- But listing examples in a programming way is very difficult
  - programming = set of instructions (if-else)
  - listing all type of pattern combinations (e.g., from pixel) is impossible in real-life
    - potential infinite “pictures”
    - time consuming
  - Hard to formalize

A concrete Example

- Challenges with real-life tasks
  - The "data" you are using for the prediction might not be fully ideal
    - e.g., a picture can be noisy
    - e.g., wrong angle, it does not capture the details you need
    - e.g., data might be ambiguous
  - You need a lot of data to "**generalize**"
  - So what if .. rather than we design an algorithm with properties to classify images
    - We write an algorithm that **finds pattern automatically** from the data

Example of a Machine Learning Algorithm

- Idea: let the computer look for the patterns
- Ex. Input = an image; Output = 1 if there is a certosino, 0 otherwise
- Automatically search for patterns that correlate with class 1 or 0



When to use Machine Learning

- When we use ML?
  - The problem is difficult to formalize the problem, easy to provide examples
  - Presence of noise

- And the ML system should
  - adapt to each sample in in order to compute the correct answer
  - find and discover new **regularities** from empirical data

- The level of information and knowledge acquire highly depends on the data quality used for the training
  - if you only show black Chartreux cat, it might learn that “black” is an essential condition for that class
  - same for a sofa
  - the more data we have, the more likely we **generalize**
    - the ML model learns the actual “concept” of that class

Machine Learning Algorithm

Machine Learning is the study of algorithms that

- improve their performance P
- at some task T
- with experience E

A well-defined learning task is given by <P, T, E>

Machine Learning Algorithm

- A task is usually described in terms of how the machine learning algorithm should process an *example (i.e. what the output should be)*

Machine Learning Algorithm

- A task is usually described in terms of how the machine learning algorithm should process an *example (i.e. what the output should be)*

- Examples of questions
  - How much or how many? (regression)
  - Which category? (classification)
  - Which group? (clustering)
  - Is this weird? (anomaly detection)
  - Which option should be taken? (recommendation)

Machine Learning Algorithm

- A task is usually described in terms of how the machine learning algorithm should process an *example (i.e. what the output should be)*
- In this case: classification task
- Now, how we represent data?

Machine Learning Algorithm

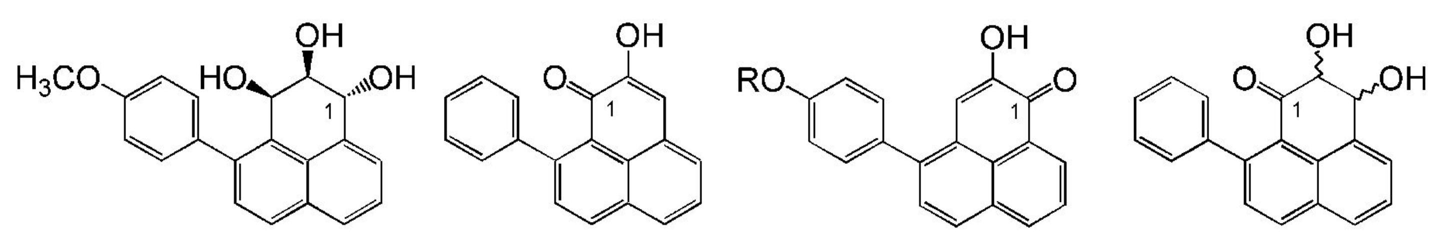
- A task is usually described in terms of how the machine learning algorithm should process an *example (i.e. what the output should be)*
- In this case: classification task
- Now, how we represent data?
    - Raw: RGB representation
    - Features Engineering: “colour of eyes”, “shape of the ear”

The Performance Measure

- How good is the learning algorithm ?
- We need to measure its performance, i.e. how accurate is the function/model returned by it!
- The performance measure depends on the task, e.g.:
    - **Classification** -> ***accuracy***, proportion of examples for which the model produces the correct output
    - It can also depend on the type of task (e.g., identifying animals rather than cancer)
        - We might have different metrics

The Performance Measure

- How good is the learning algorithm ?
- We need to measure its performance, i.e. how accurate is the function/model returned by it!
- The performance measure depends on the task, e.g.:
    - **Regression** -> ***mean squared error (MSE)***, the *average* of the squares of the *errors*



predicted toxicity index →	1.2	0.9	0.75	1.1
real toxicity index →	1.0	1.1	0.8	1.2
squared error →	0.04	0.04	0.0025	0.01

MSE: 0.023125

The Experience

- The dataset
- Which kind of data ?
  - real-valued features
  - discrete features
  - mixed features
- How do we get data ?
  - obtained once for all (batch learning)
  - acquired incrementally by interacting with the environment (on-line learning)
- How can data be used ?
  - Learning paradigms

Main Learning Paradigms

Different paradigms

- Supervised Learning
- Unsupervised learning
- Reinforcement learning
- .. and many others.

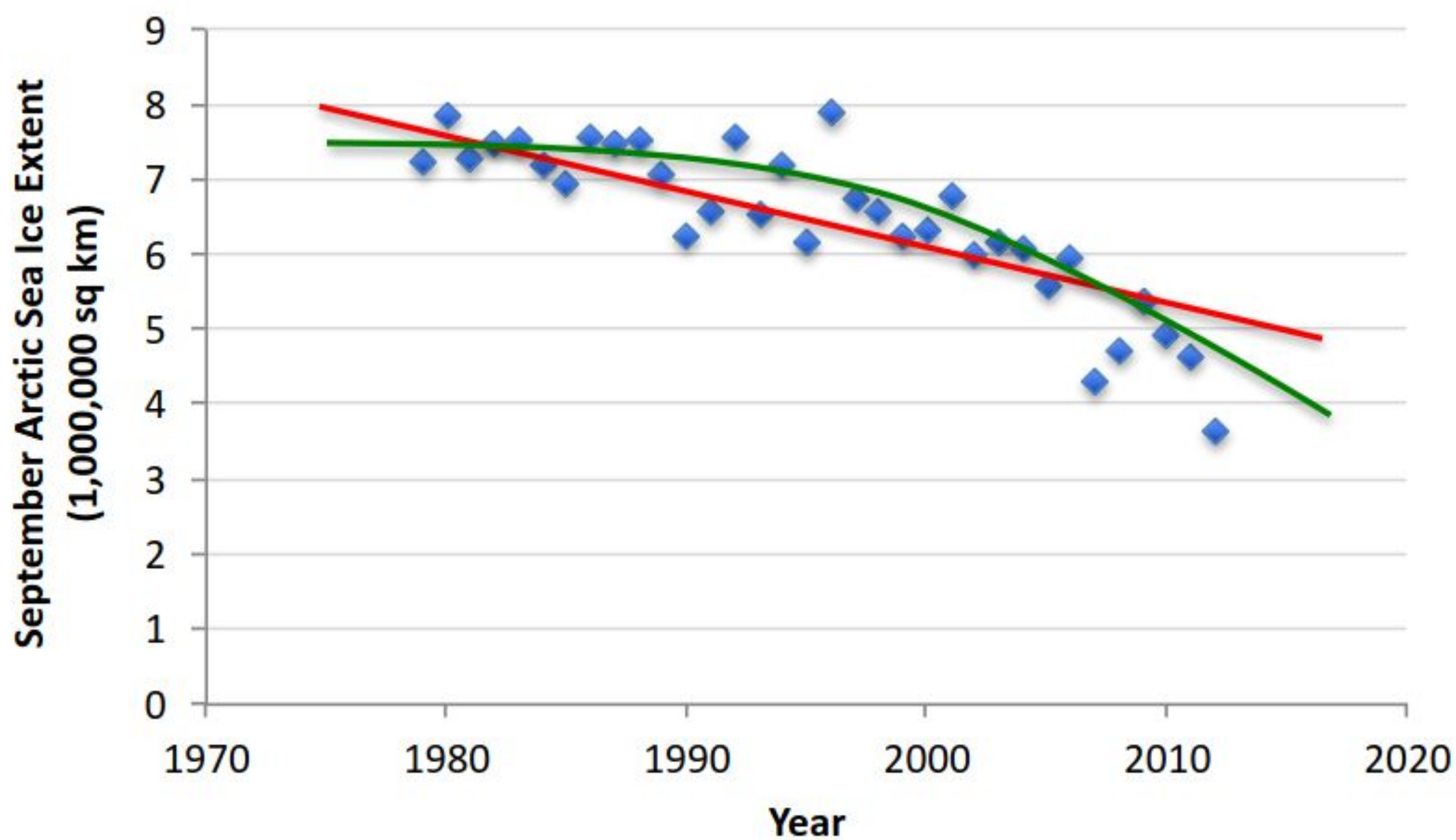
Main Learning Paradigms

Different paradigms

- Supervised (inductive) Learning
  - Given: training data + desired outputs (labels)
- Unsupervised learning
  - Given: training data (without desired outputs)
- Reinforcement learning
  - Given: Rewards from sequence of actions
- .. and many others.

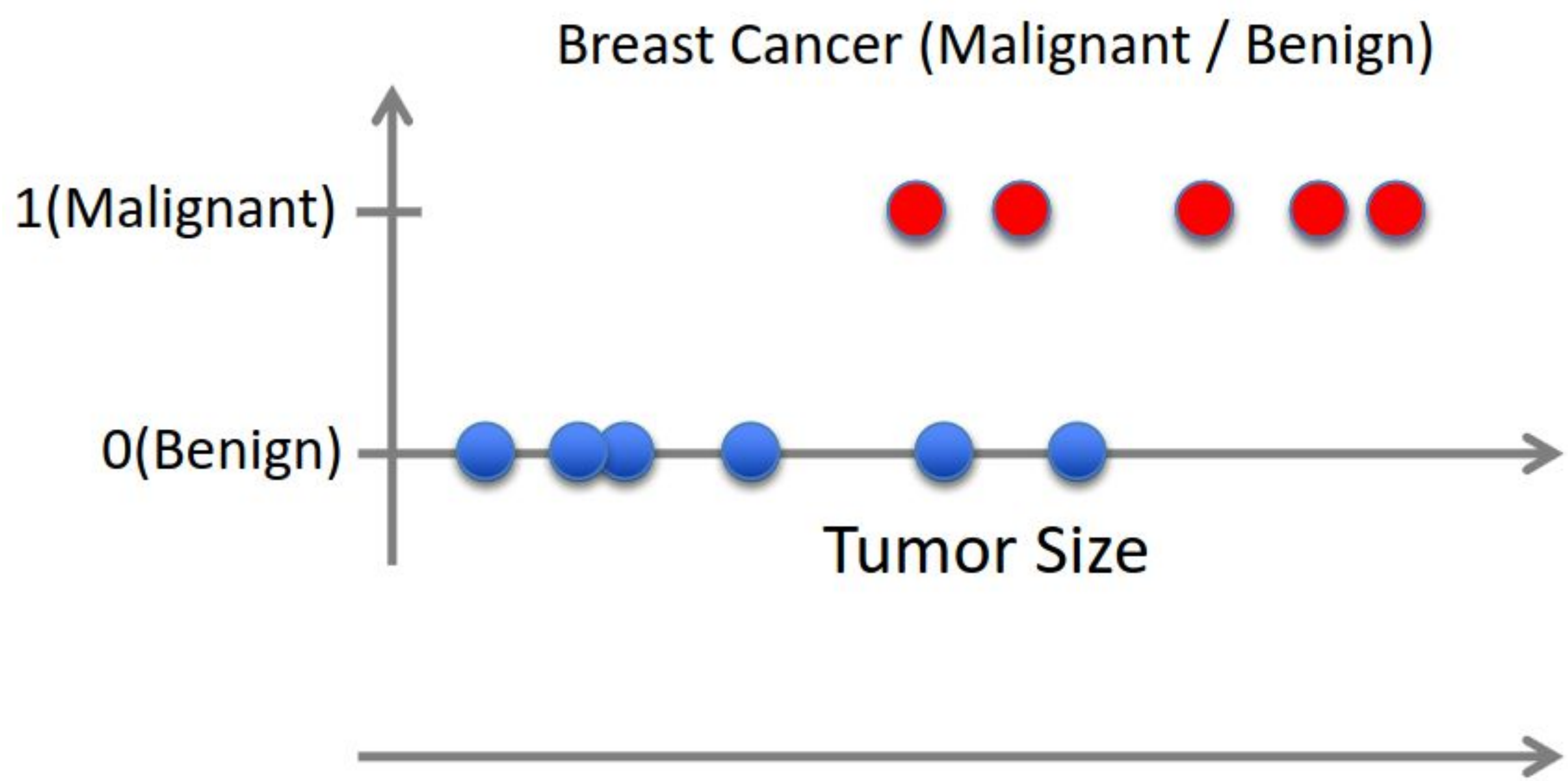
Supervised Learning: Regression

- Given a set of points $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$
- Learn a function h(x) to predict y given x
  - y is continuous -> regression
- As you can see, **there are** many f that we can use



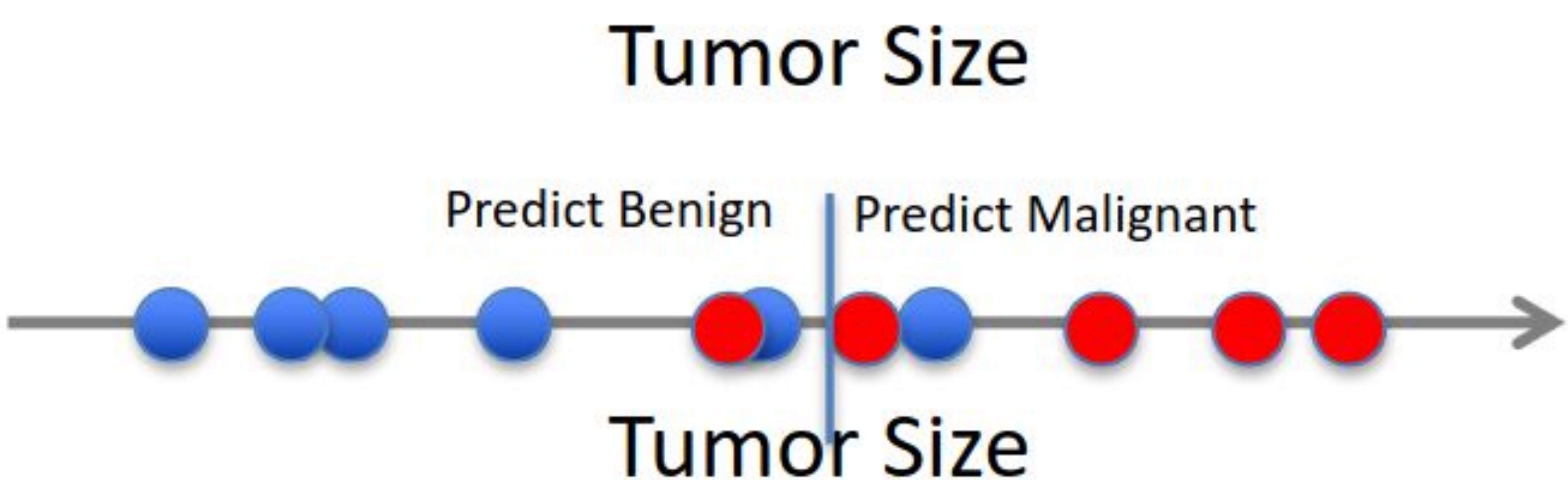
Supervised Learning: Classification

- Given a set of points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
- Learn a function h(x) to predict y given x
  - y is discrete -> classification
- As you can see, **there are** many f that we can use



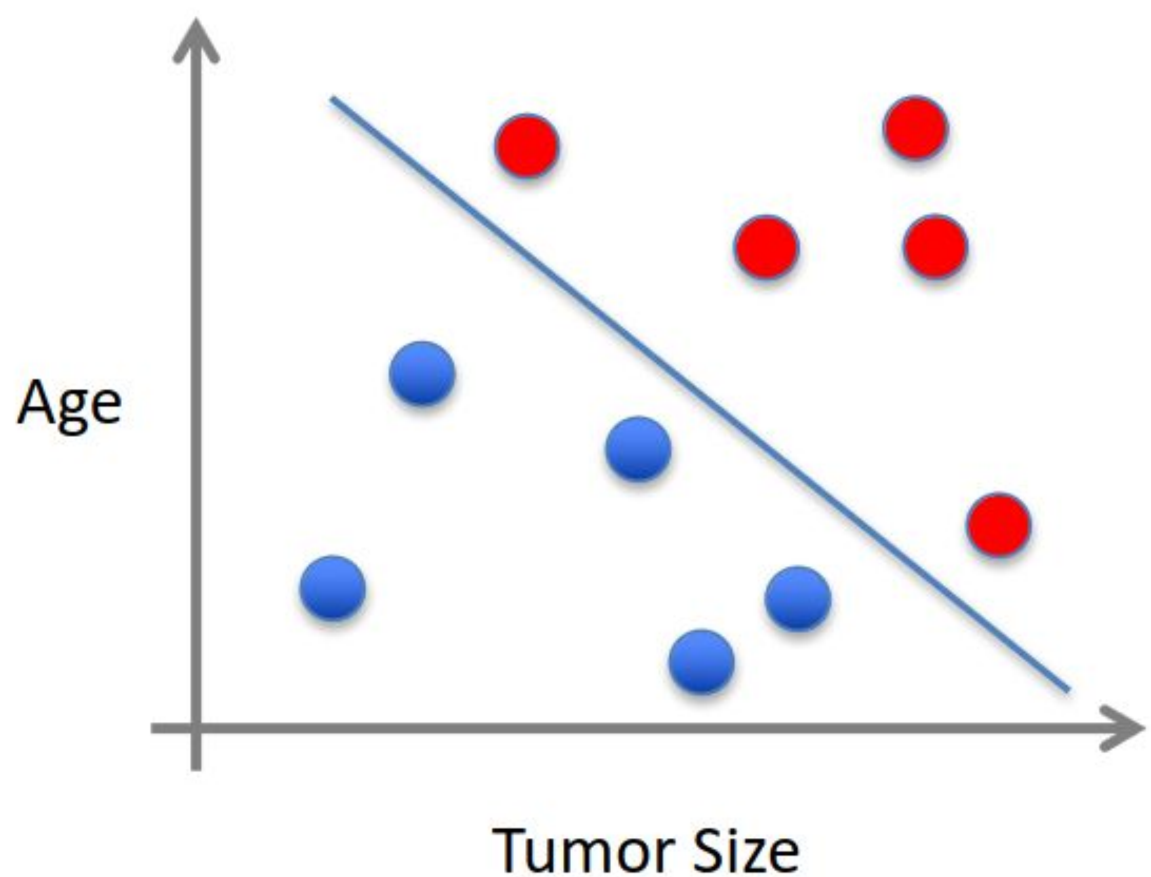
Supervised Learning: Classification

- Given a set of points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
- Learn a function h(x) to predict y given x
  - y is discrete -> classification
- As you can see, **there are** many f that we can use



Supervised Learning: Classification

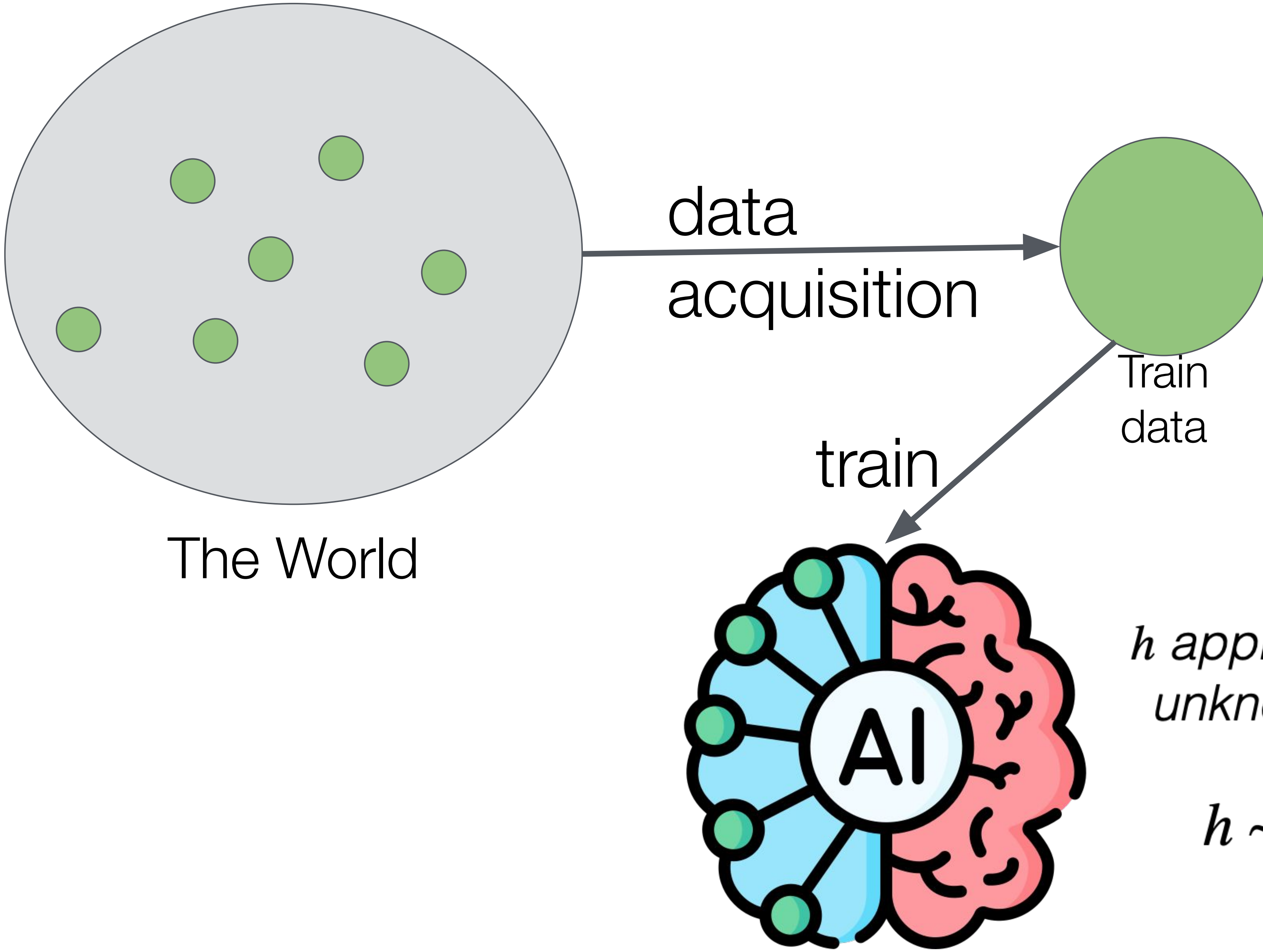
- Given a set of points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
- Learn a function h(x) to predict y given x
  - y is discrete -> classification
- As you can see, **there are** many f that we can use



Supervised Learning

- Training Set is drawn from “the entire world”
  - drawn: how we collect the data
  - the entire world: all possible data that there exists
    - impossible to have
- There **exists** a function f that solve the task
  - f is *unknown*
    - if it is known, we design an algorithm

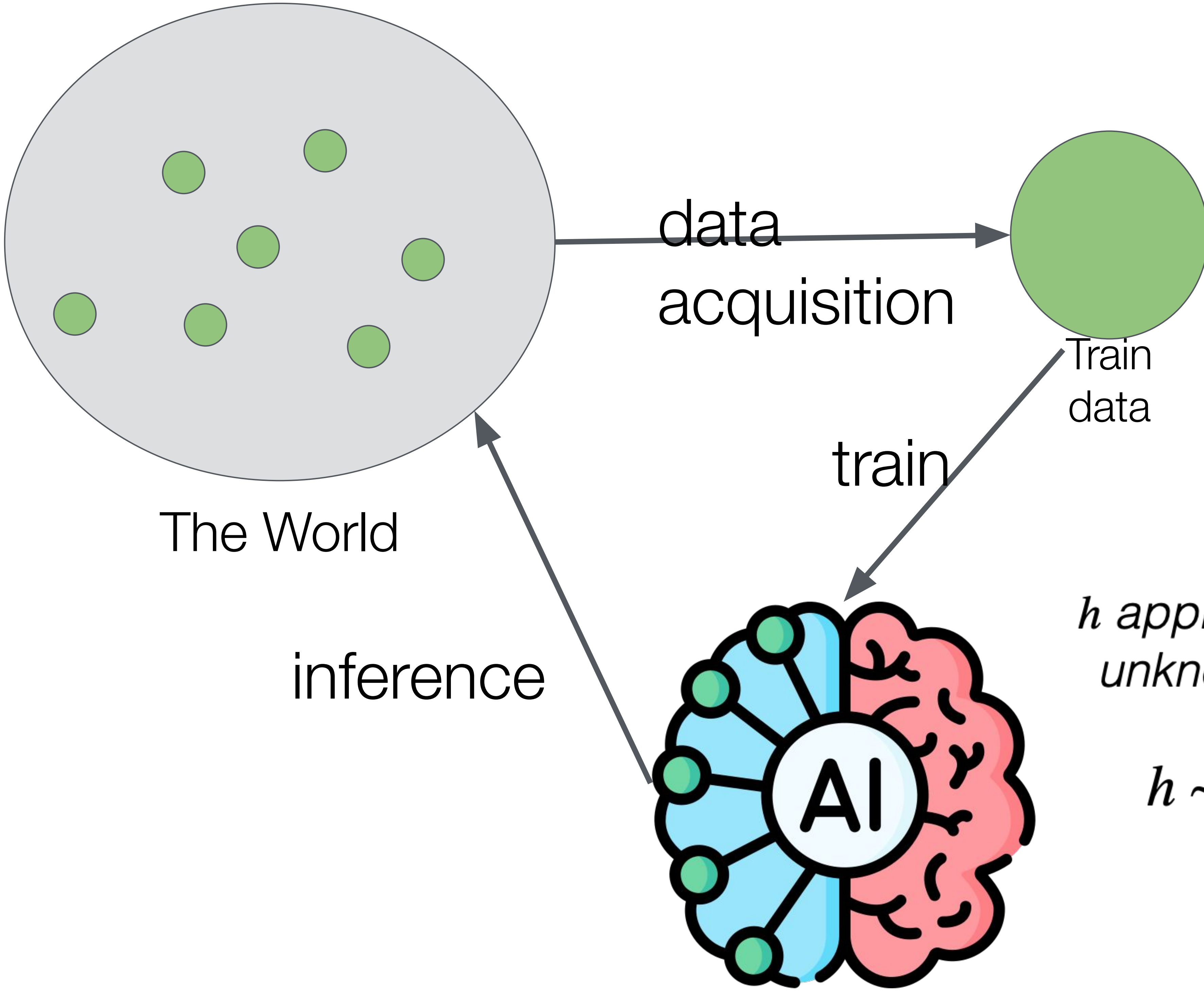
Supervised Learning



h approximates the unknown target f

$$h \sim f: X \rightarrow Y$$

Supervised Learning



h approximates the unknown target f

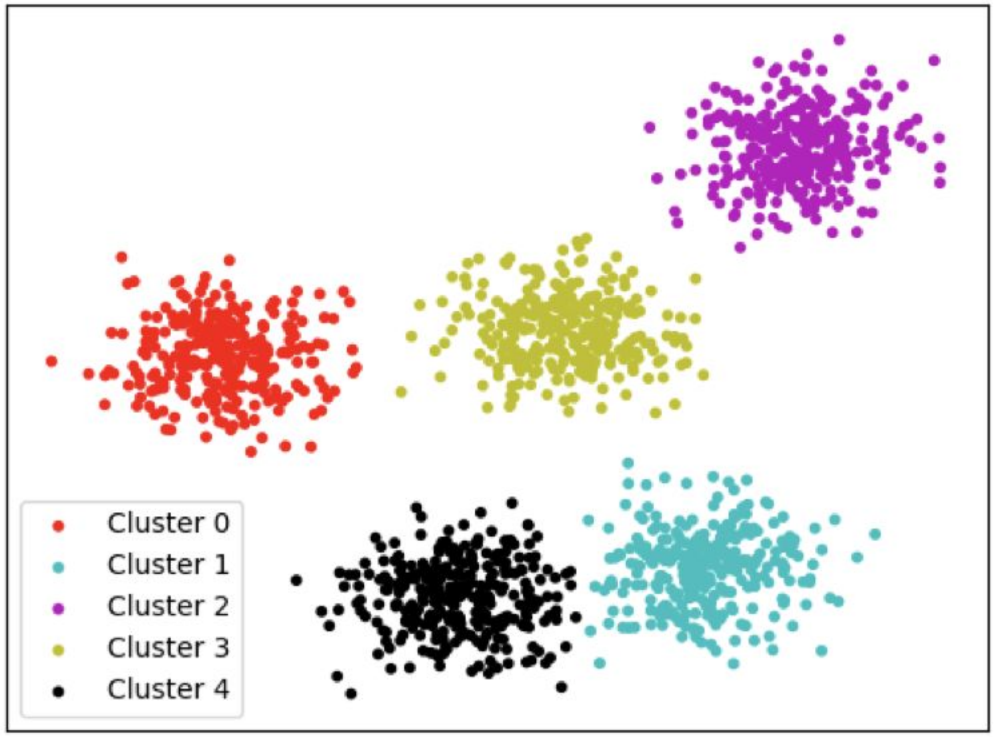
$$h \sim f: X \rightarrow Y$$

Supervised Learning

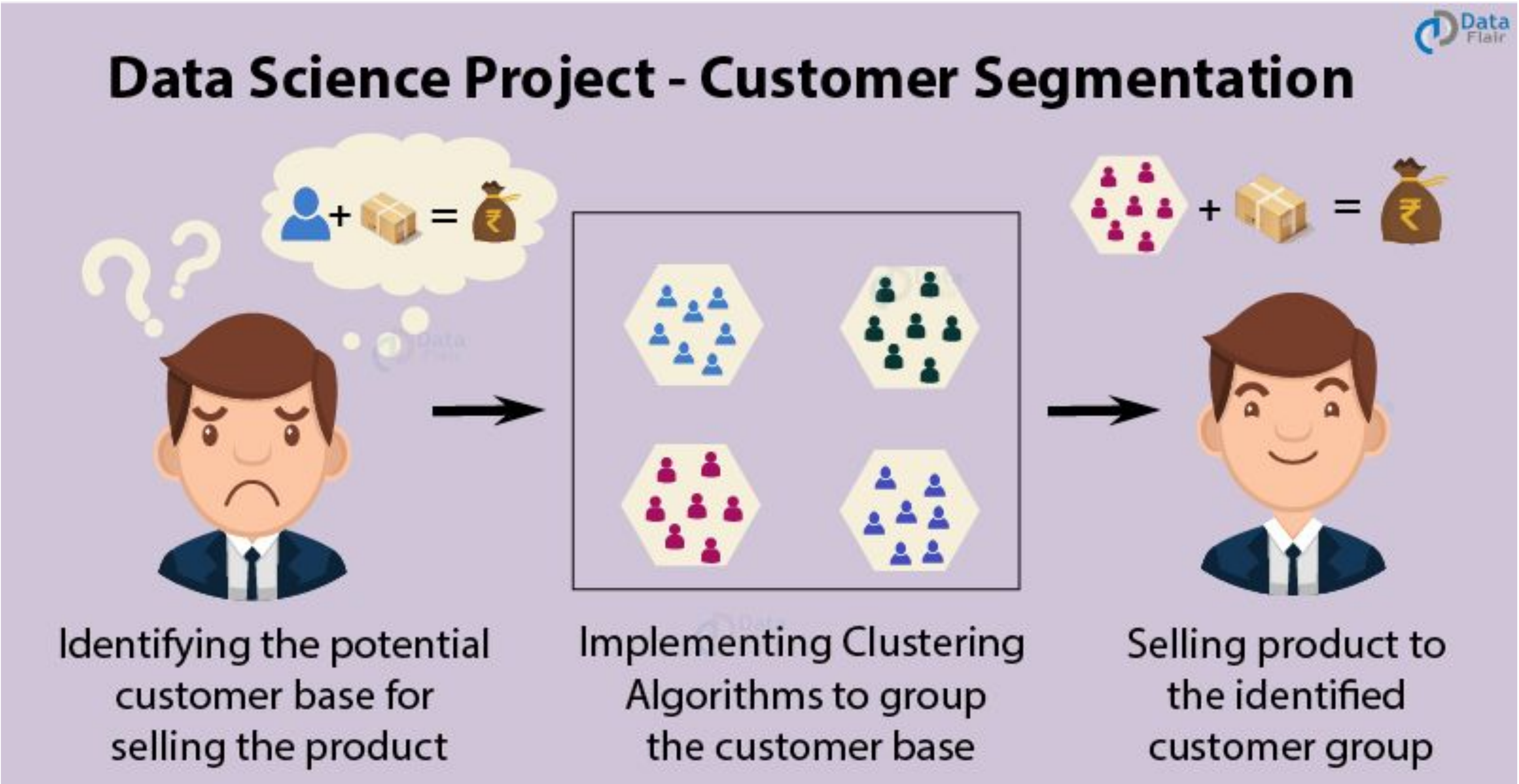
- Since h is an approximation
  - you might not solve the task **perfectly**
- This happens for many reasons such as
  - The training data you have does not allow the algorithm to generalize to the entire world
    - you might need more samples
    - there might be **bias** in your data
    - or the data “representation” does not allow you to solve the task
      - e.g., see example of breast cancer classification
    - your function h is not suitable to learn the insight (or all) to solve the task
  - usually, in real life, you have all of these problem together
    - with maybe different “degrees” of impact

Unsupervised Learning

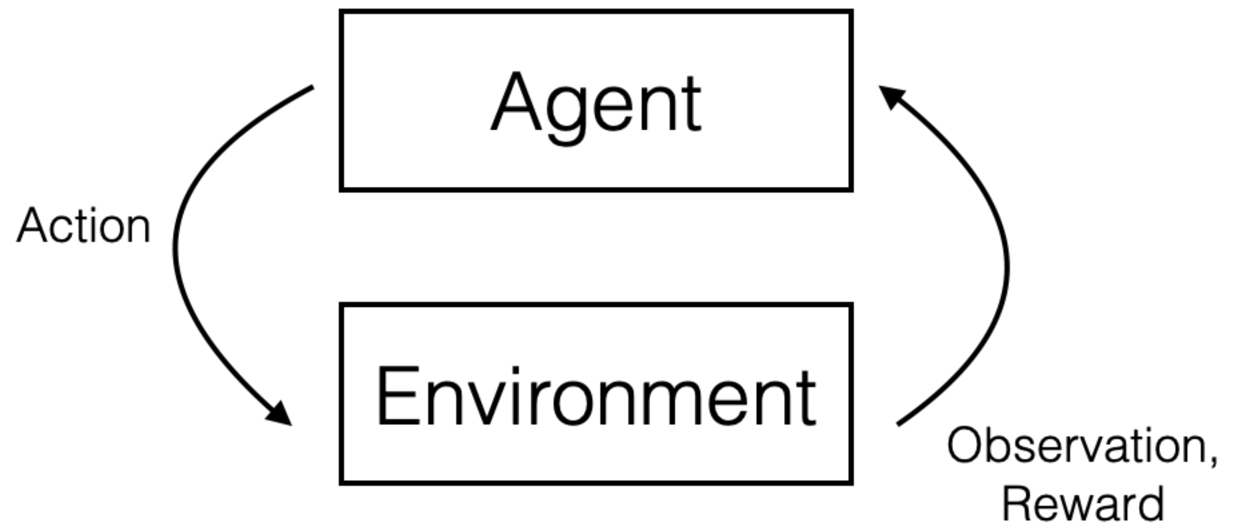
- **Goal:** find regularities / patterns on the data
- Given examples $\{x^{(i)}\}$, discover regularities on the whole input domain
- There is no expert (*i.e.* no supervision)



Clustering application example



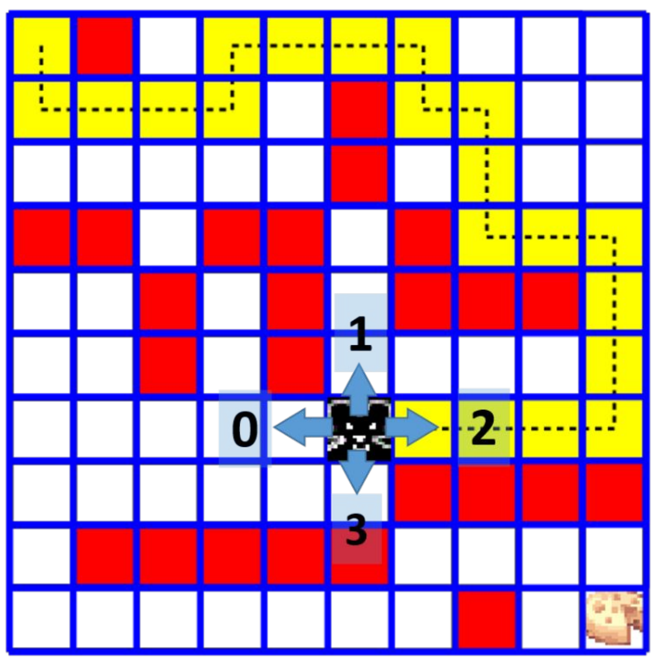
Reinforcement Learning



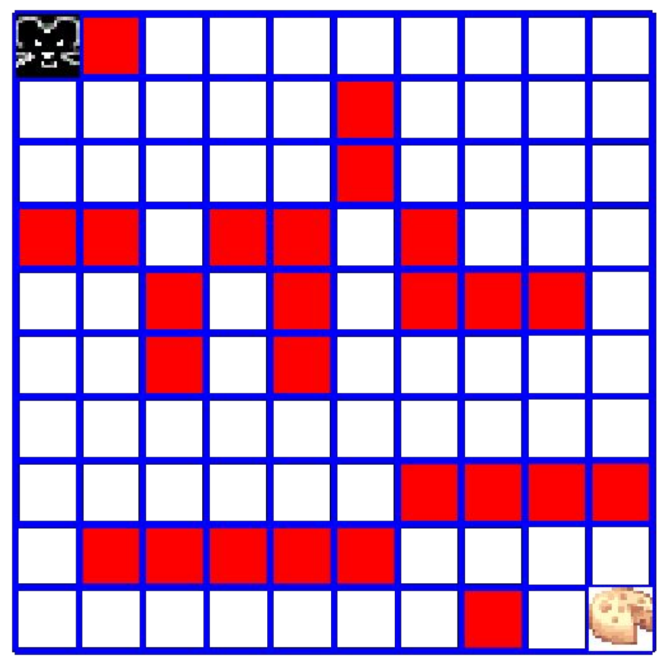
- Agent which may
  - be in state s
  - execute action a
    (among the ones admissible in state s)
- and operates in an environment e, which in response to action a in the state s returns
  - the next state and a reward *r* (which can be positive, negative or neutral)
- The goal of the agent is to maximize a function of the rewards

Example of Reinforcement Learning

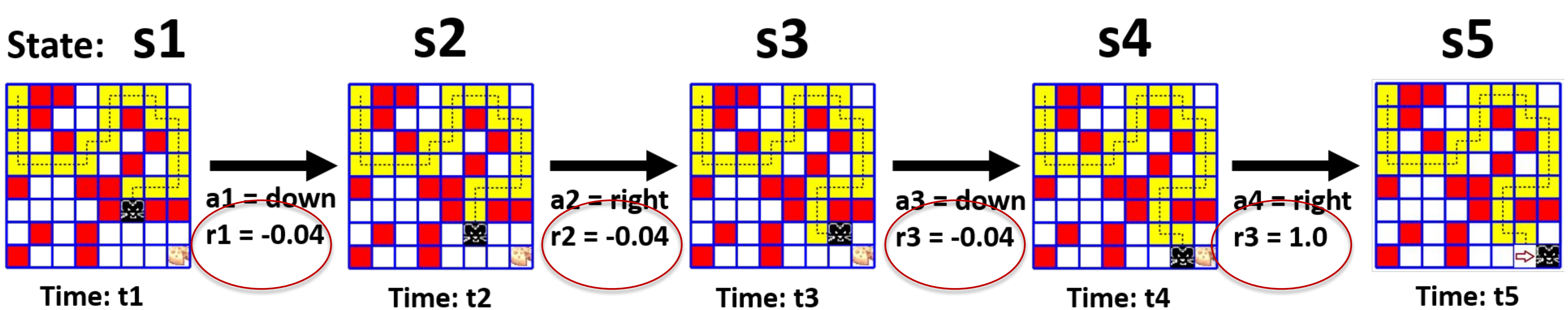
4 actions



An action takes the agent to a different state..

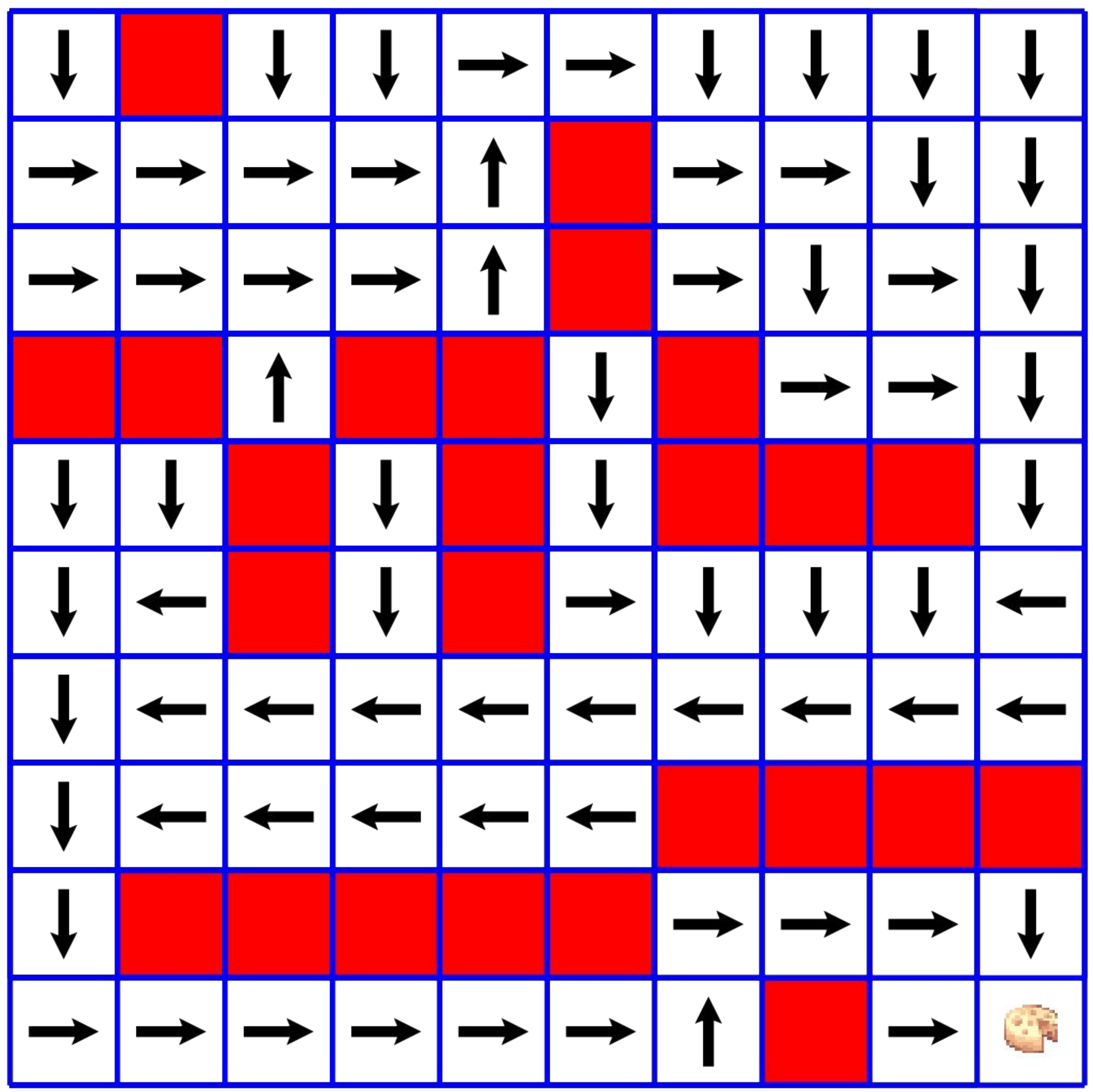


…and provides a reward



Example of RL 2

- The agent learns a policy: a mapping from states to actions, that maximizes the long-term reward



Part 2: Learning

Ingredients

- Training Data
  - drawn from the **Instance Space** X
- Hypothesis Space H
  - set of functions that can be implemented by the machine learning algorithm
- f (the target function) is unknown
  - f can be represented by the hypotheses in H
  - there exist $h \in \mathcal{H}$ s.t. h is similar to f

- Therefore, **learning** means finding the function h that approximate the most f

Inductive Bias

- Can we have H s.t. it contains all the possible functions?
    - No! Potentially infinite!
- **Inductive Bias** = all the assumptions about the “nature” of the target function and its selection
- Two type of bias:
    - Restriction: limit the hypothesis space
    - Preference: impose ordering on hypothesis space

Concept Learning

- A concept in an instance space X is defined as a boolean function over X , $c : X \rightarrow \{0, 1\}$

- An example in X is defined as
  - (x, c(x)), where $x \in X$ and c() is a boolean function over x

- Let $h : X \rightarrow \{0, 1\}$ a boolean function in X
  - h satisfies $x \in X$ if $h(x) = 1$ (true)

- h is consistent with an example x if $h(x) = c(x)$
  - h is consistent with Tr is h is consistent with any training example in Tr

Concept Learning

Conjunction of m literals

- Instance Space $\rightarrow$ strings of m bits: $X = \{s | s \in \{0,1\}^m\}$
- Hypothesis Space $\rightarrow$ all the logic sentences involving literals $l_1, \ldots, l_m$ (any boolean variable l_i or its negation $\neg l_i$) and just containing the operator $\wedge$ (**and**):
  $\mathcal{H} = \{f_{\{i_1,\ldots,i_j\}}(s) | f_{\{i_1,\ldots,i_j\}}(s) \equiv L_{i_1} \wedge L_{i_2} \wedge \cdots \wedge L_{i_j}$, where $L_{i_k} = l_{i_k}$ or $\neg l_{i_k}, \{i_1, \ldots, i_j\} \subseteq \{1, \ldots, 2m\}\}$

Notice that if in a formula a literal occurs together with its negation, then the formula is always *false* (unsatisfiable formula)
So, all the formulas containing a literal and its negation, are equivalent to *false*

Learning Conjunctions of Literals

Find-S Algorithm

/* it finds the most specific hypothesis which is consistent with the training set */

- input: training set Tr
- initialize h to the most specific
  $h \equiv l_1 \wedge \neg l_1 \wedge l_2 \wedge \neg l_2 \wedge \cdots \wedge l_m \wedge \neg l_m$
- for each positive training instance $(x, true) \in Tr$
  - remove from h any literal which is not satisfied by x
- returns h

Example of application: *m=5*

(positive) Example	current hypothesis
	$h_0 \equiv l_1 \wedge \neg l_1 \wedge l_2 \wedge \neg l_2 \wedge l_3 \wedge \neg l_3 \wedge l_4 \wedge \neg l_4 \wedge l_5 \wedge \neg l_5$
1 1 0 1 0	$h_1 \equiv l_1 \wedge l_2 \wedge \neg l_3 \wedge l_4 \wedge \neg l_5$
1 0 0 1 0	$h_2 \equiv l_1 \wedge \neg l_3 \wedge l_4 \wedge \neg l_5$
1 0 1 1 0	$h_3 \equiv l_1 \wedge l_4 \wedge \neg l_5$
1 0 1 0 0	$h_4 \equiv l_1 \wedge \neg l_5$
0 0 1 0 0	$h_5 \equiv \neg l_5$

Notice that $\boxed{h_0 \leq_g h_1 \leq_g h_2 \leq_g h_3 \leq_g h_4 \leq_g h_5}$

Moreover, at every step the current hypothesis h_i is substituted by hypothesis h_{i+1} which constitutes a *minimal generalization* of h_i consistent with the current example.

Thus **Find-S** returns the most specific hypothesis which is consistent with Tr

Inductive Bias

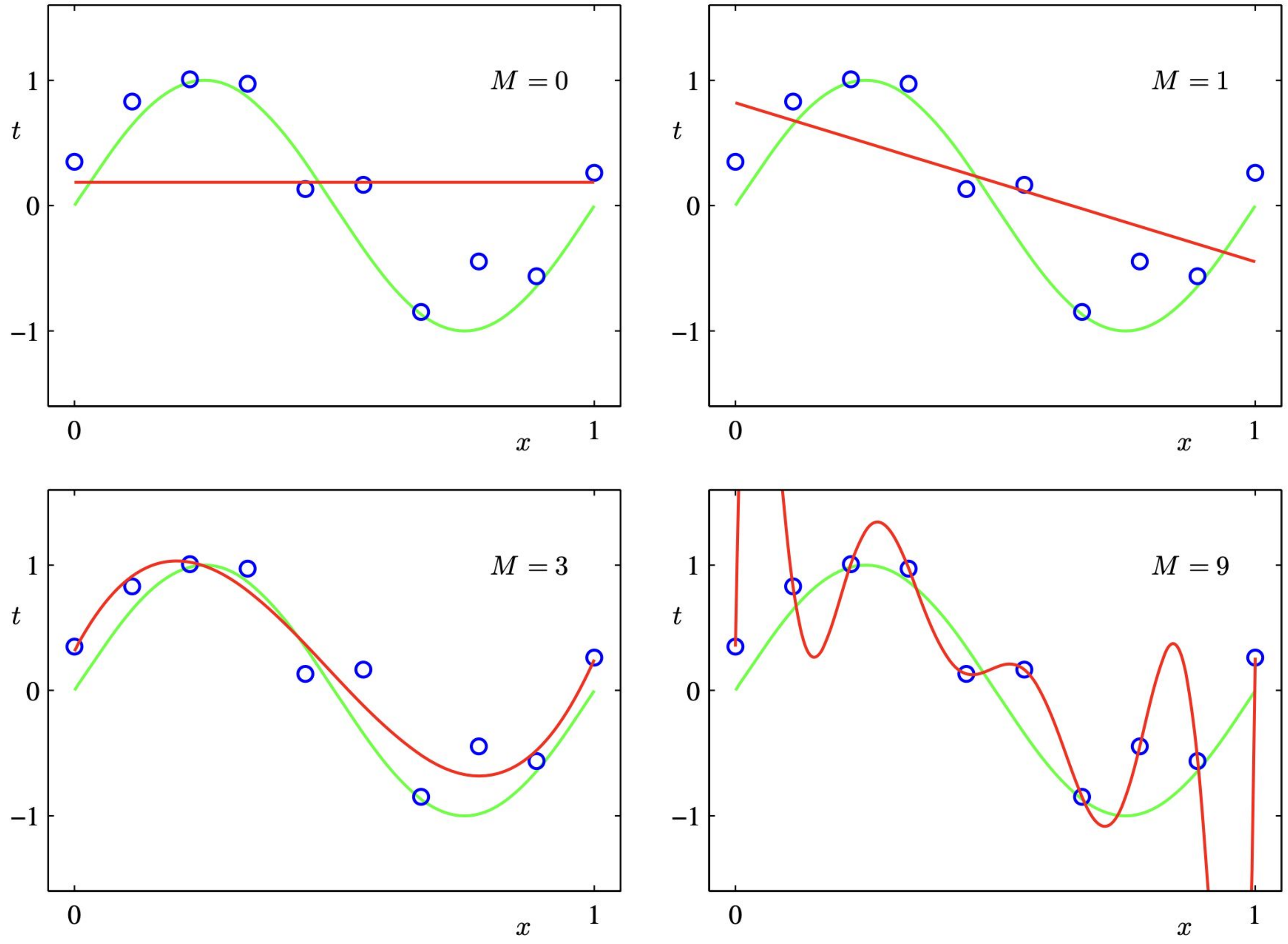
due to inductive bias

Hypothesis	Satisfied Instances
h_0	
h_1	10010
h_2	11010, 10010
h_3	11010, 10010, 10110, 11110,…
h_4	11010, 10010, 10110, 10100, 11110, 10000,11000,11100,…
h_5	11010, 10010, 10110, 10100, 00100, 11110, 10000,11000,11100, 01010,…

Part 3:
Error

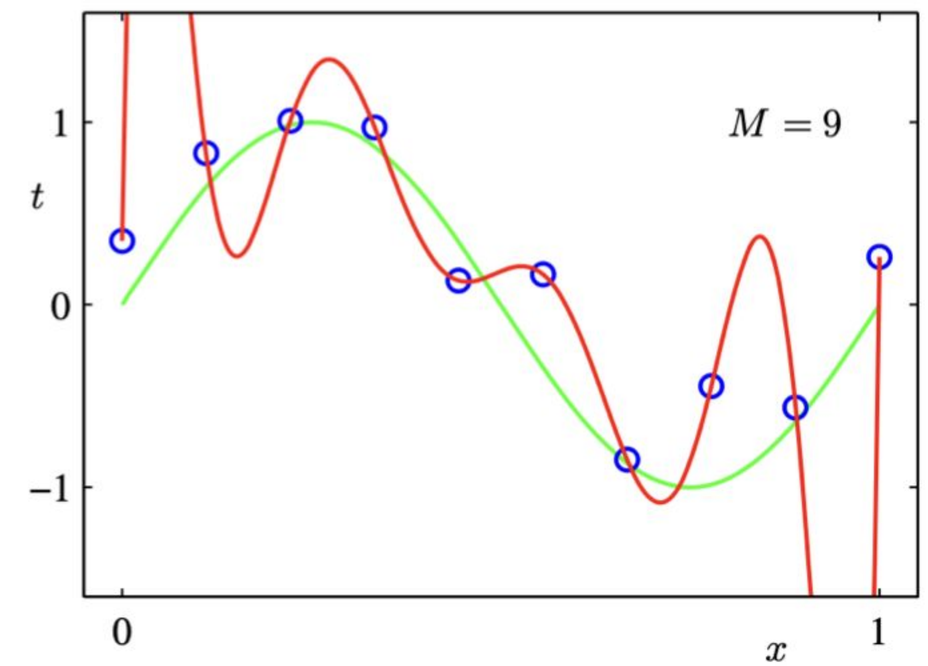
Hypothesis Spaces Example

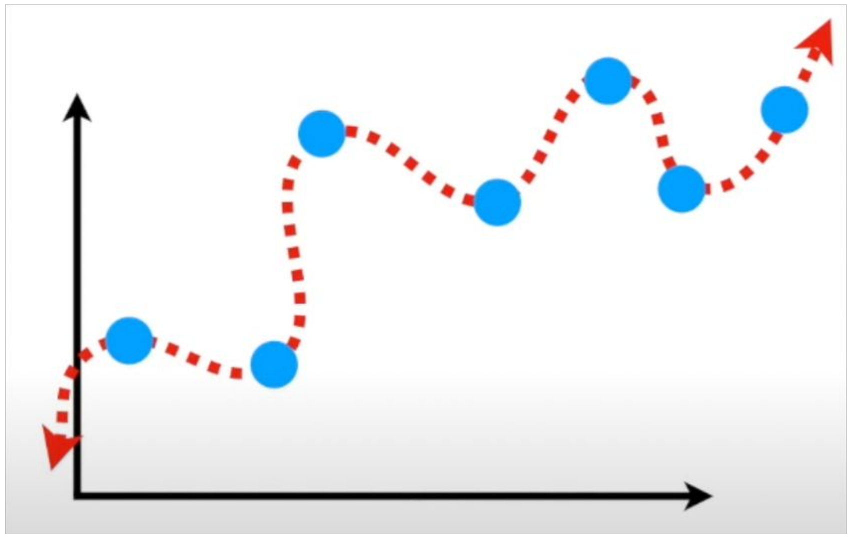
- Regression Task; function f in green, examples with noise added; Different polynomials of degree M as Hypothesis spaces

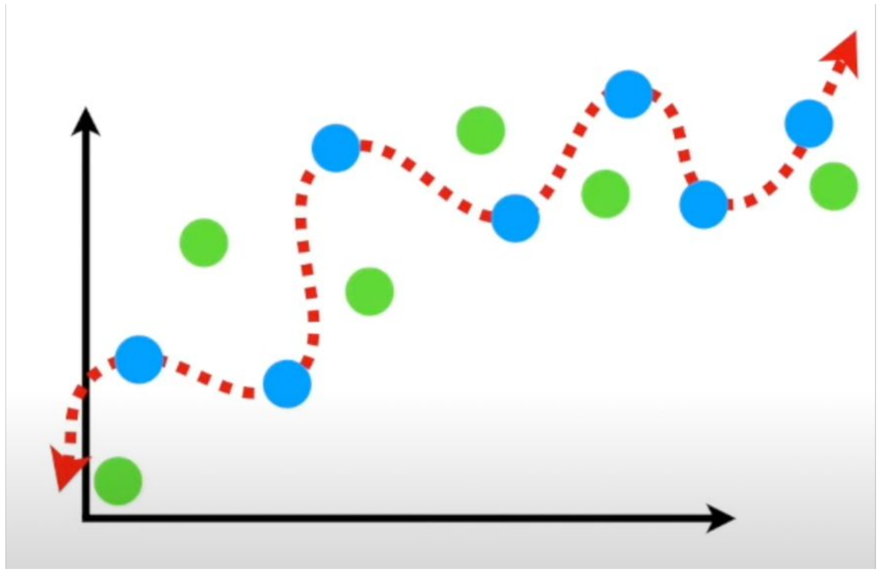


Variance

- M=9 adapts "too well" to the data: it is so powerful that can model the noise as well!
- M=9 has high variance/sensitivity (if we select a different set of training points, the fitting curve changes a lot; it would not happen to M=1)
- High variance is undesirable because…

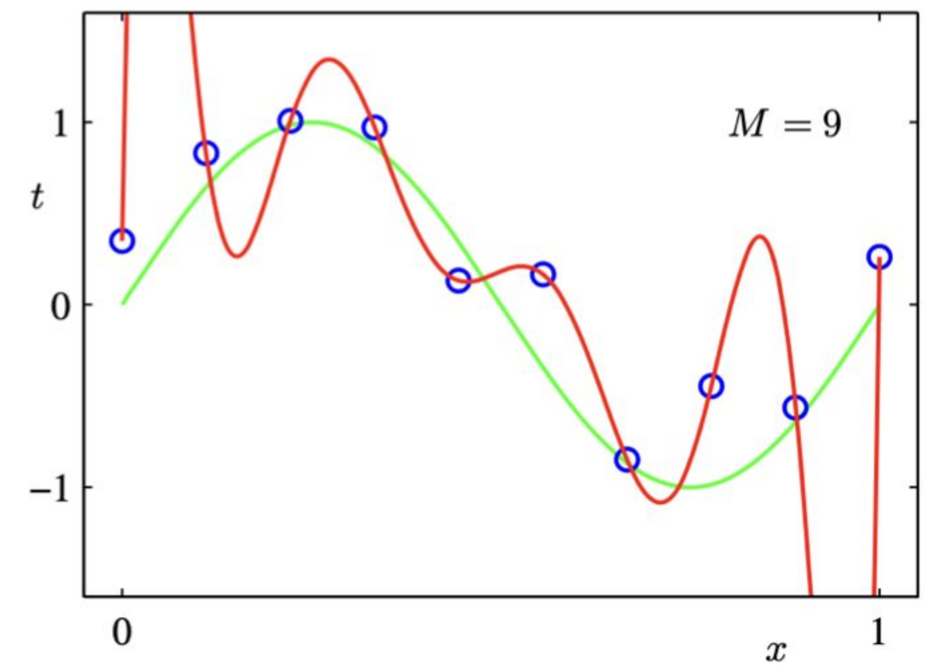


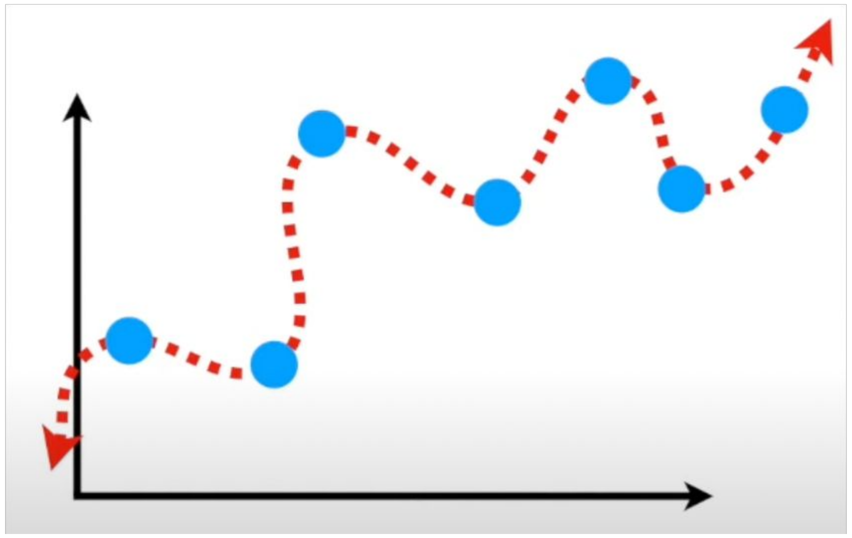


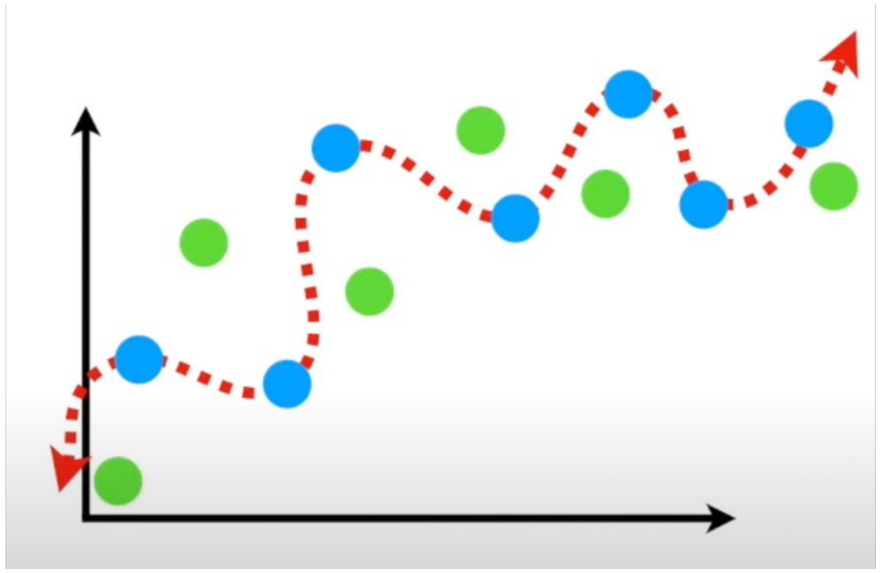


Variance

- High variance is undesirable:
- Consider two function with different complexity
  - f() (simpler) that does not change a lot
  - h() (more complex) that change a lot
  - what can we say about the error h() will do on unseen examples?

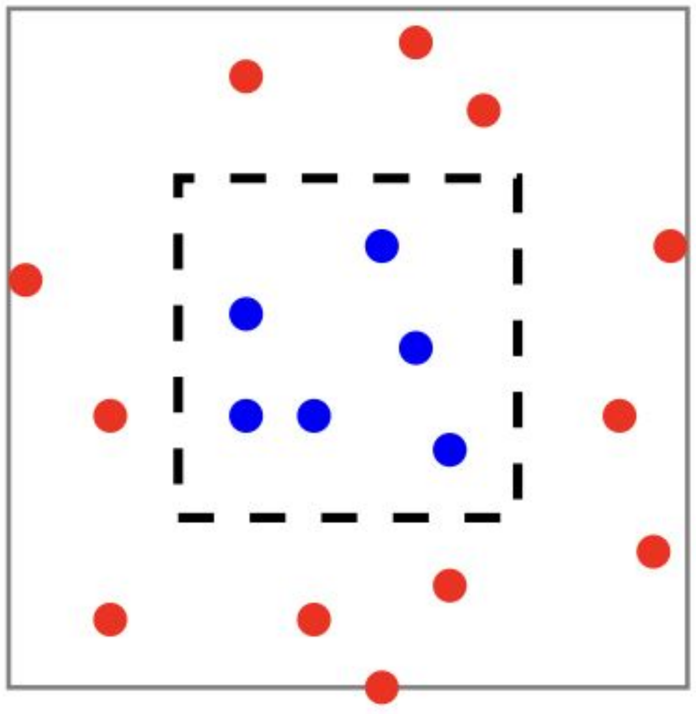






Complex Models - Overfitting

- *h(x) = 1 if x=x_blue, 0 otherwise*
- x is classified as blue only if it coincides with a blue point, i.e. it “memorizes” the training set
- Zero error on the training set but it is not learning anything!



Notions of Statistical Learning Theory

-

- The dataset we have is a random sample identically and independently distributed according to some probability distribution $\mathcal{D}$

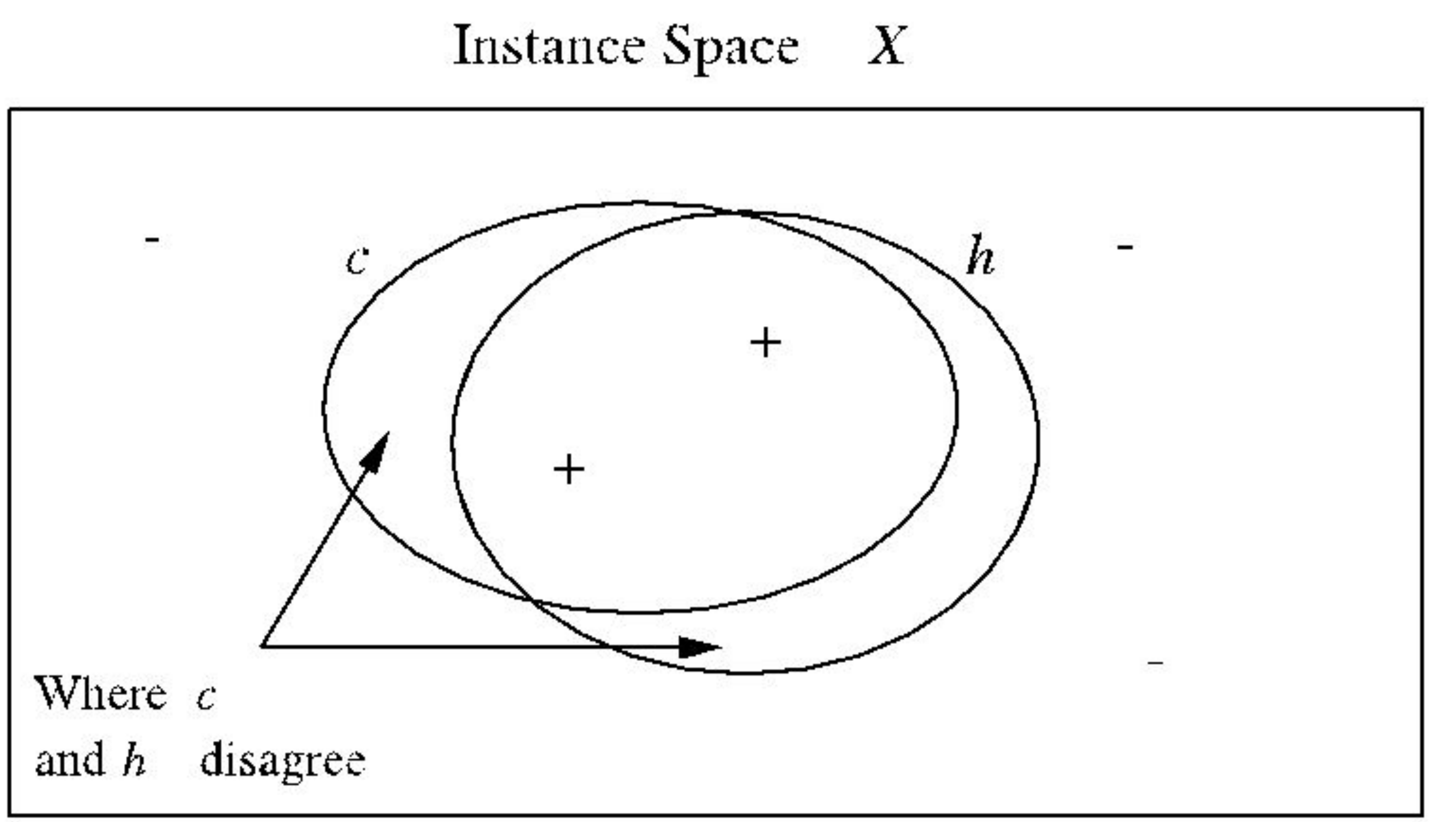
- In general, we are interested in generalization!

- E.g. Emotion detection system from faces.

  - Training set: pictures of your faces expressing different emotions

  - Goal: classify emotions of other people!

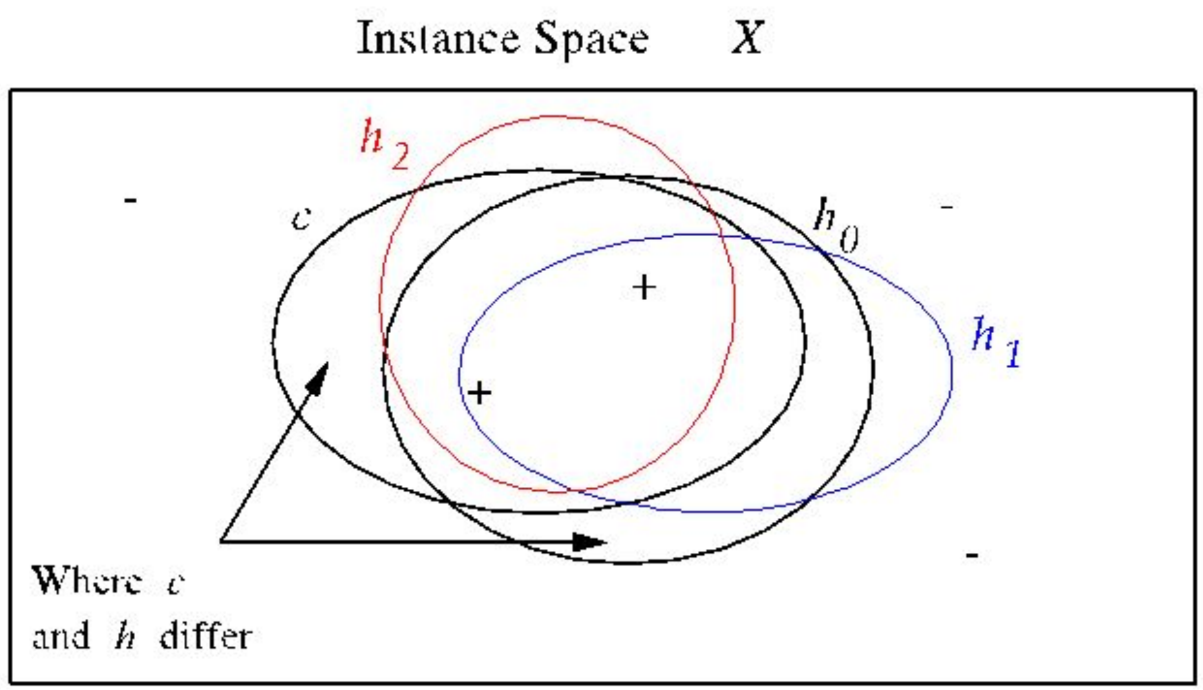
True Error



Def: The **True Error** ($error_{\mathcal{D}}(h)$) of hypothesis h with respect to target concept c and distribution $\mathcal{D}$ (to observe an input instance $x \in X$) is the probability that h will misclassify an instance drawn at random according to $\mathcal{D}$:

$$error_{\mathcal{D}}(h) \equiv \Pr_{x \in \mathcal{D}}[c(x) \neq h(x)]$$

Empirical Error

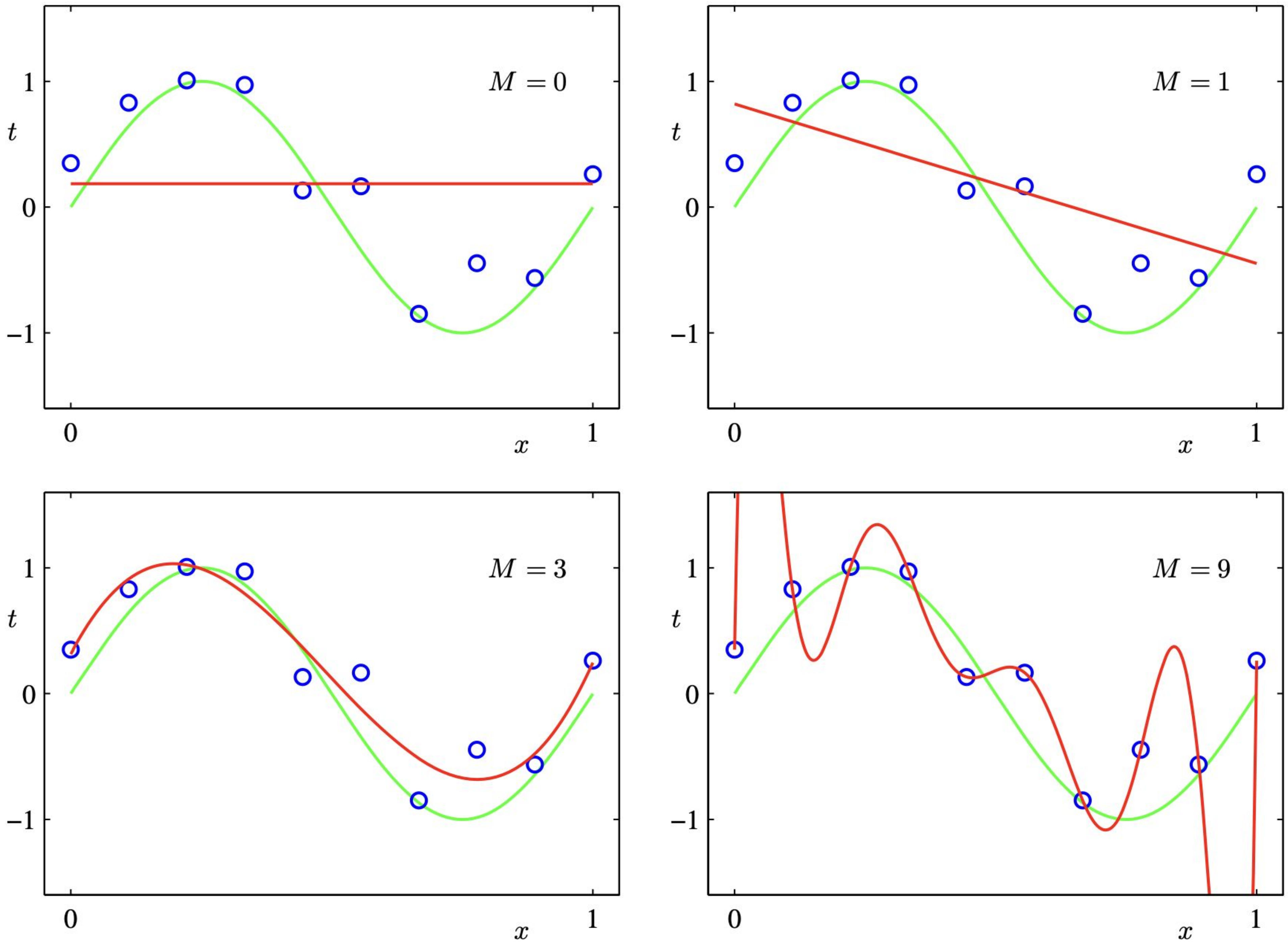


Def: The **Empirical Error** ($error_{Tr}(h)$) of hypothesis h with respect to Tr is the number of examples that h misclassifies:

$$error_{Tr}(h) = Pr_{(x,f(x))\in Tr}\left[f(x) \neq h(x)\right] = \frac{\left|\left\{\left(x, f(x)\right) \in Tr \;\middle|\; f(x) \neq h(x)\right\}\right|}{|Tr|}$$

Def: $h \in \mathcal{H}$ overfits Tr if $\exists h' \in \mathcal{H}$ such that $error_{Tr}(h) < error_{Tr}(h')$, but $error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$.

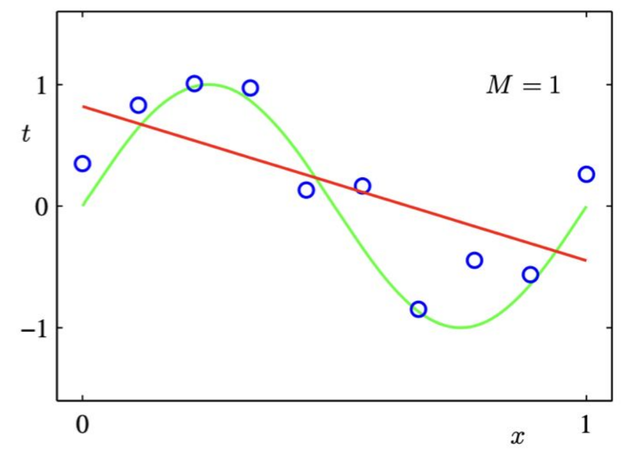
Overfitting

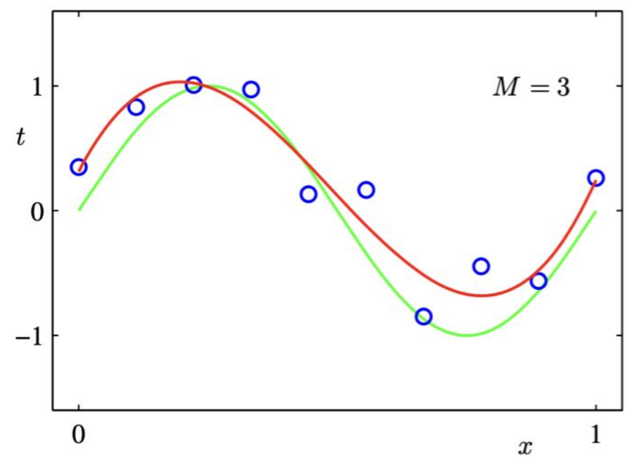


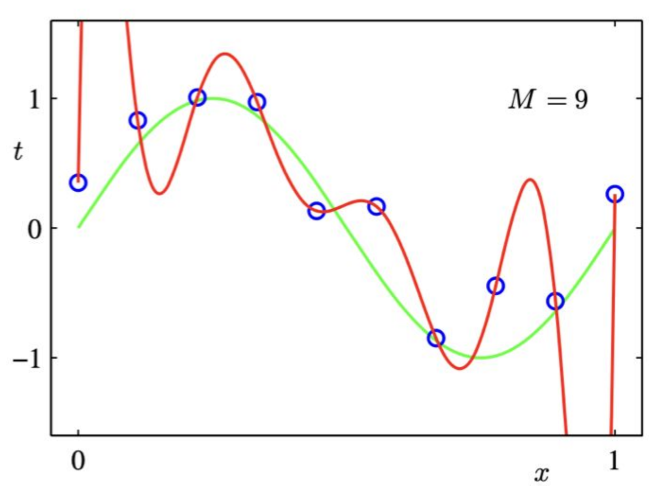
Bias-Variance Tradeoff

- The bias error is produced by weak assumptions in the learning algorithm
  - High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting)
- The variance is an error produced by an over-sensitivity to small fluctuations in the training set
  - High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs (overfitting)

Bias-Variance trade-off

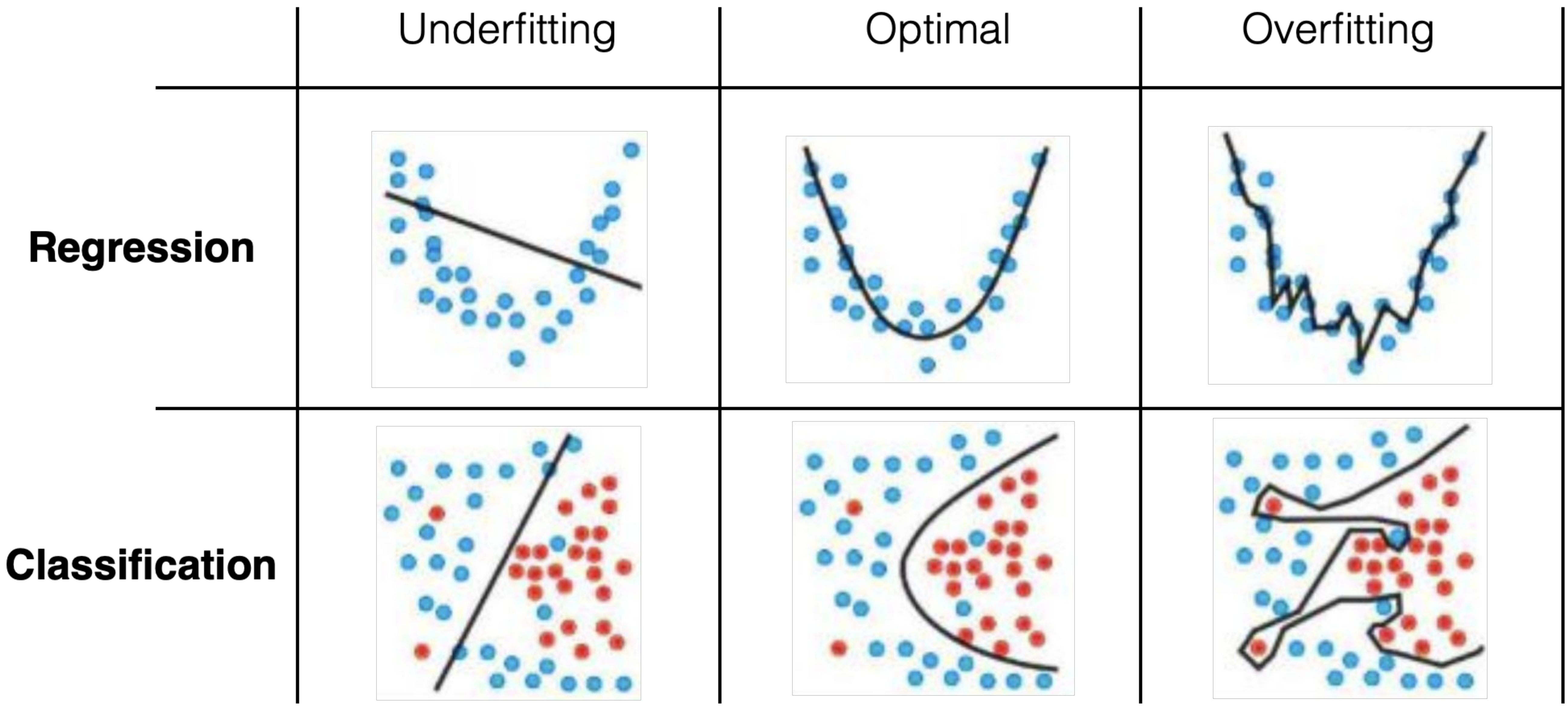






- Function too simple - High Bias - risk of underfitting (no function in H has high error on the training set –> high true error!)
- right complexity for this problem - good balance between bias and variance
  - Small empirical error
  - Small True error
- H too “powerful” - might model noise - high variance
  - Very low empirical error (error on the training set)
  - High true error!

Bias-Variance Tradeoff



Estimating the True error

- Minimizing the error on the training set (Empirical Risk Minimization) may not be the best option (see overfitting later)

We want to minimize the true error!

$$error_{\mathcal{D}}(h) = error_{Tr}(h) + generalization(h)$$

2 ways: bounds and estimation

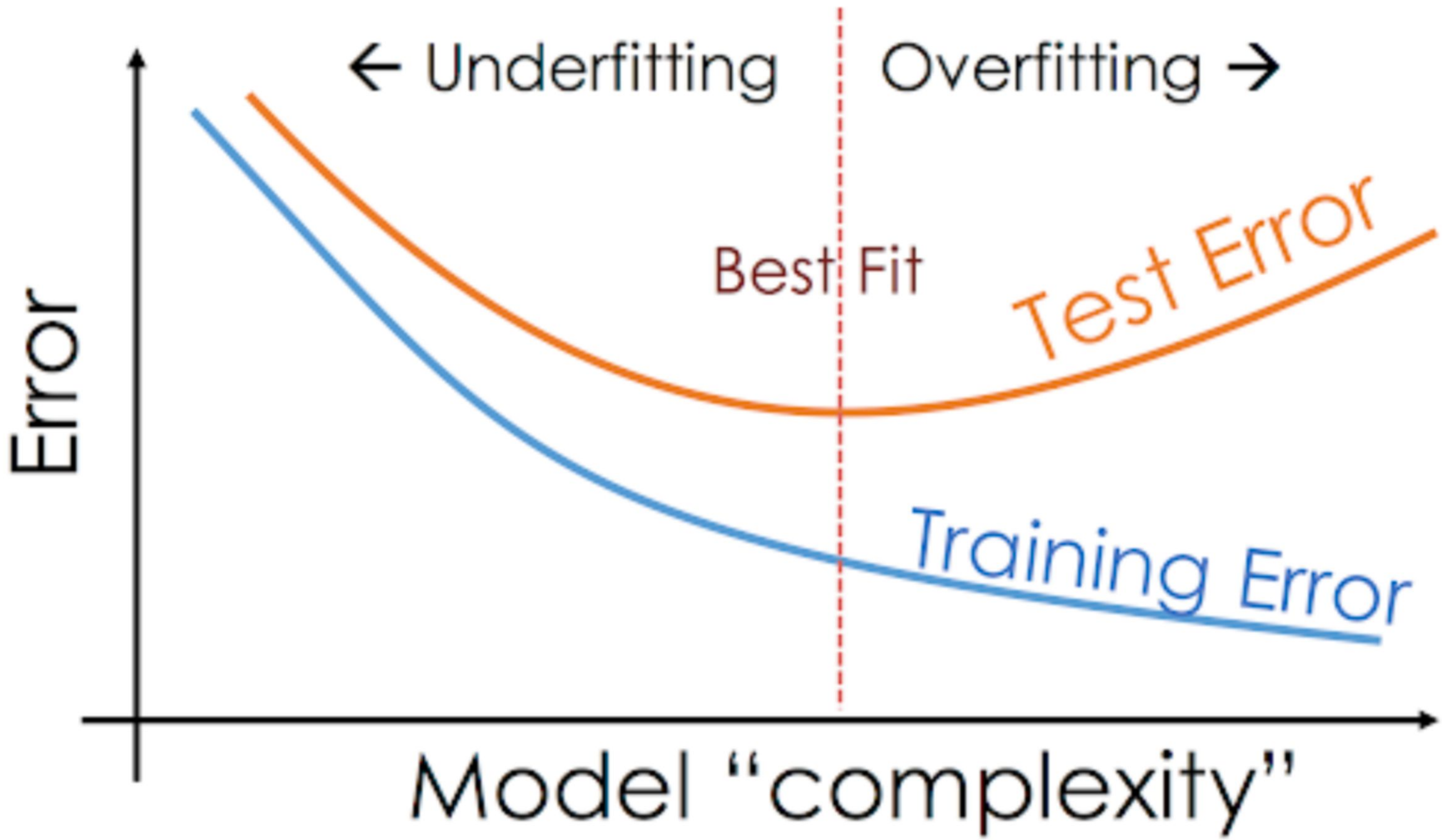
1. relate the Empirical error and the true error with generalization bounds:

$$error_{\mathcal{D}}(h) \leq error_{Tr}(h) + complexityMeasure(\mathcal{H})$$

   with $\mathrm{h} \in \mathcal{H}$, exploiting some complexity measure of the hypothesis space

2. compute the error on unseen data (TEST set)

Overfitting - 2



No Free Lunch Theorem

- No Free Lunch Theorem: there is no “best” learning algorithm

- Each learning algorithm defines an inductive bias, we can constrict a problem for which his inductive bias does not result in the best bias-variance tradeoff

- This is one of the reason why there are so many learning algorithm