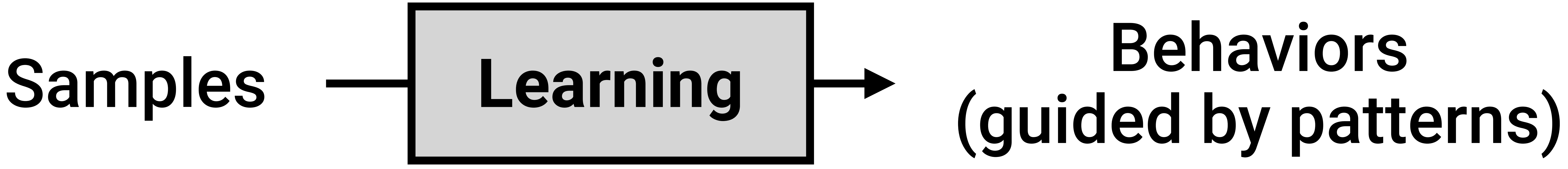
Elements of ML

Recap: What is learning?

- The process of extracting and utilizing **patterns** from the samples



Recap: What is learning?

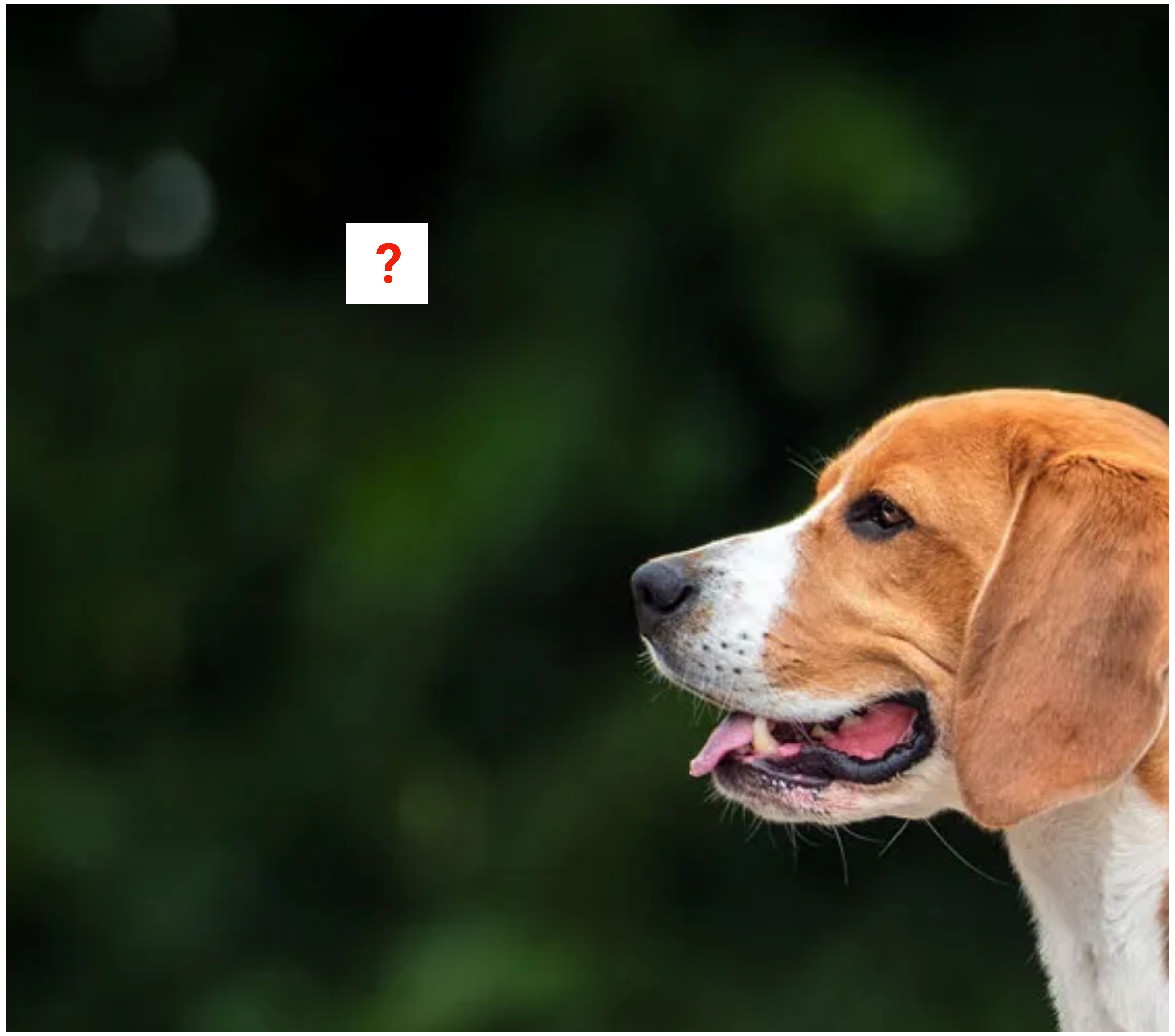
- **Today.** We formalize this concept:
  - What exactly is a pattern?
  - How can we program a machine to find one?

  - In particular, we provide some unified but hand-wavy perspectives:
    - starting next week, we look at individual ML algorithms

Patterns

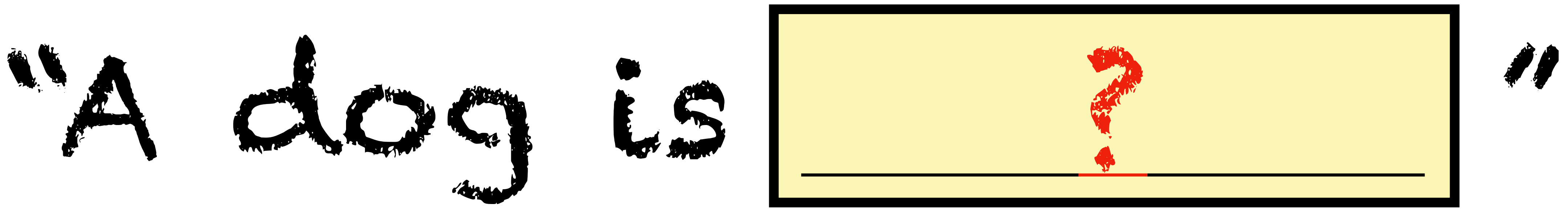
Patterns

- Associations of distinct variables
- **Example.** “Green pixels” are associated with “another green pixel”



Patterns

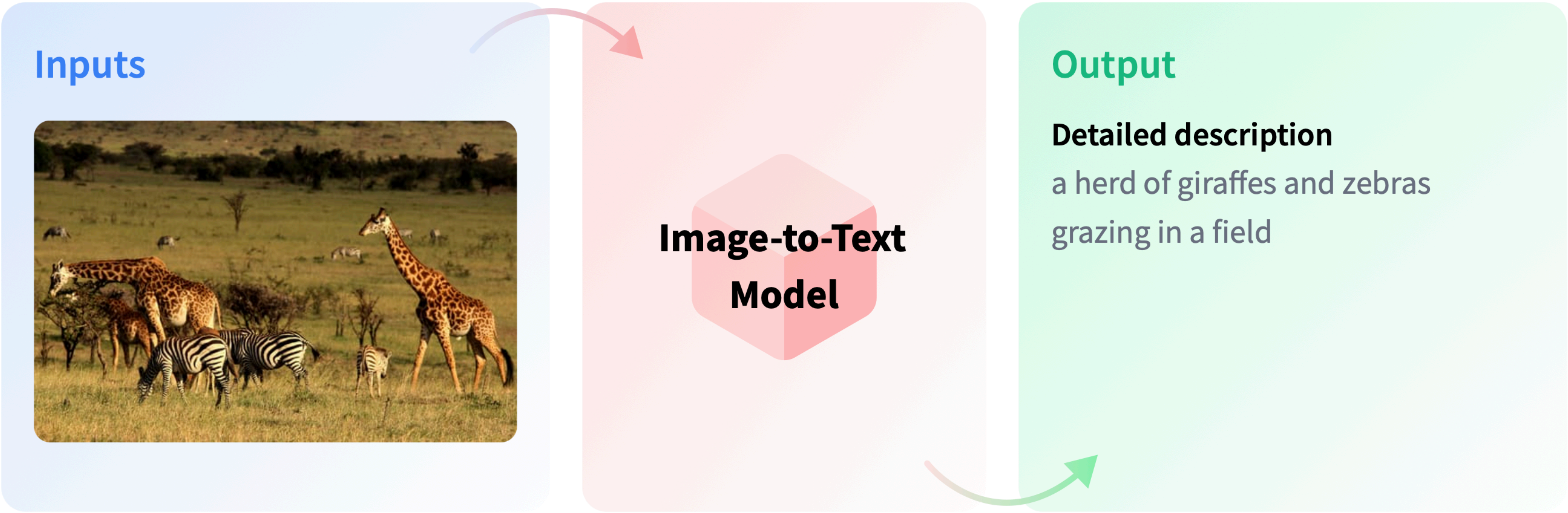
- Associations of distinct variables
- **Example.** The text “A dog is” is associated with the word “cute”



(Next word prediction — GPT is trained this way)

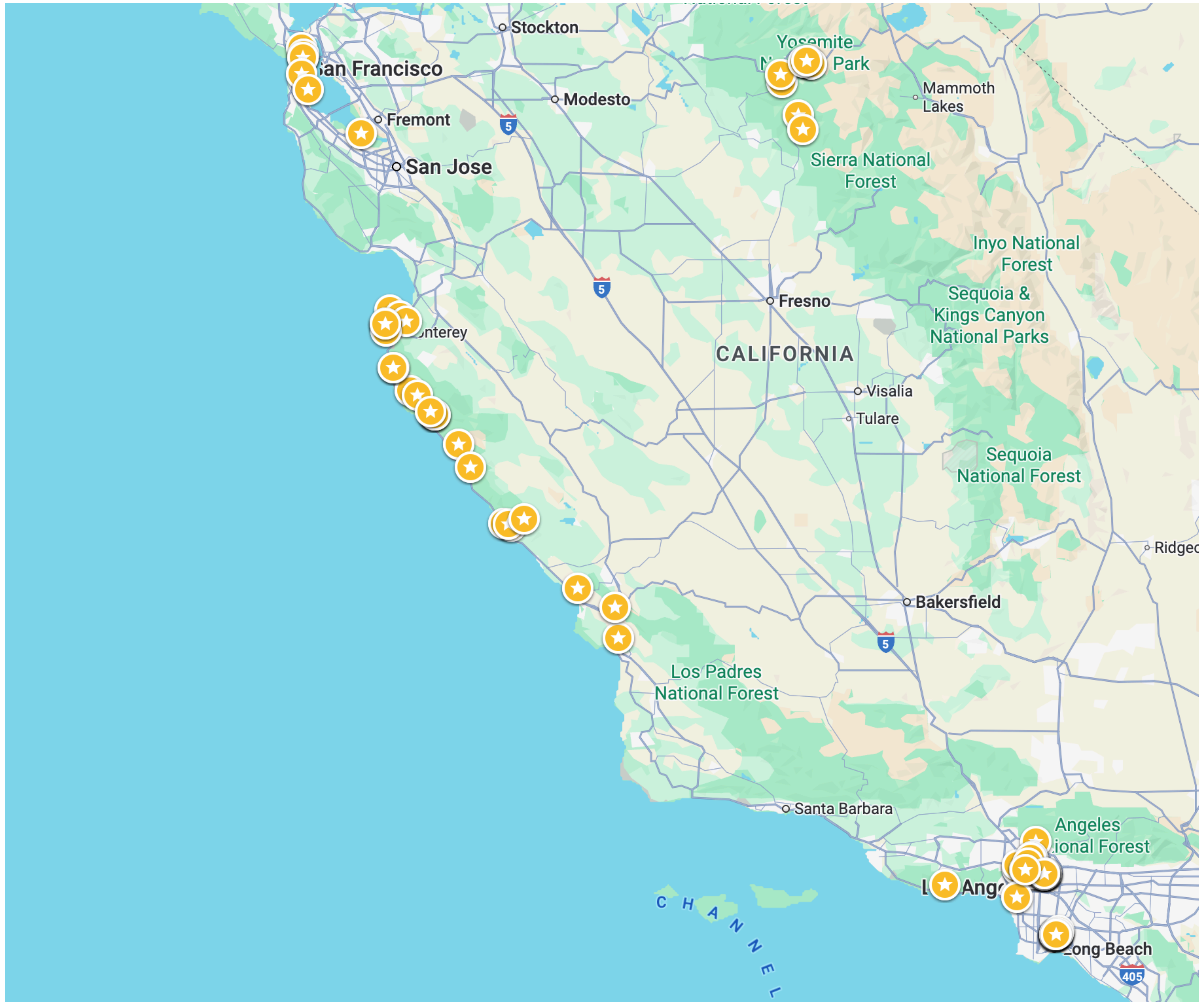
Patterns

- The variables need not be of same modality
- **Example.** An image is associated with its textual description



Patterns

- We can make associations with **imaginary variables**
- **Example.** "Locations" are associated with "Categories (imaginary)"



Categories

- Roughly, learning is about associating different random variables: X, Y
  - Jointly distributed as some probability distribution $P_{XY}(x, y)$
  - However, P_{XY} is not known to the learner
  - Instead, we have **training data**

Categories

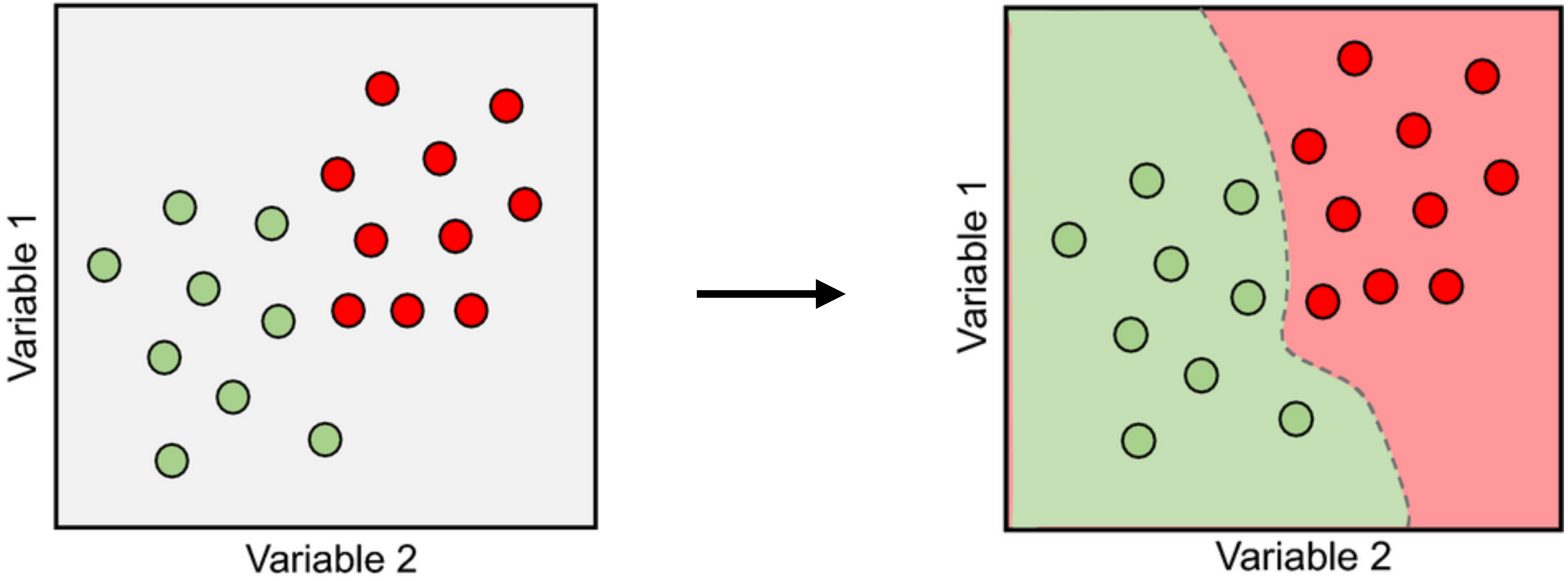
- Depending on the **type of data** available, learning can be categorized into:
  - Supervised learning
  - Unsupervised learning
  - Reinforcement learning

- Note: Of course, there are many other terminologies
  - semi-supervised, self-supervised, active, …

- Note: In this course, we focus on supervised & unsupervised
  - Reinforcement learning as a special session

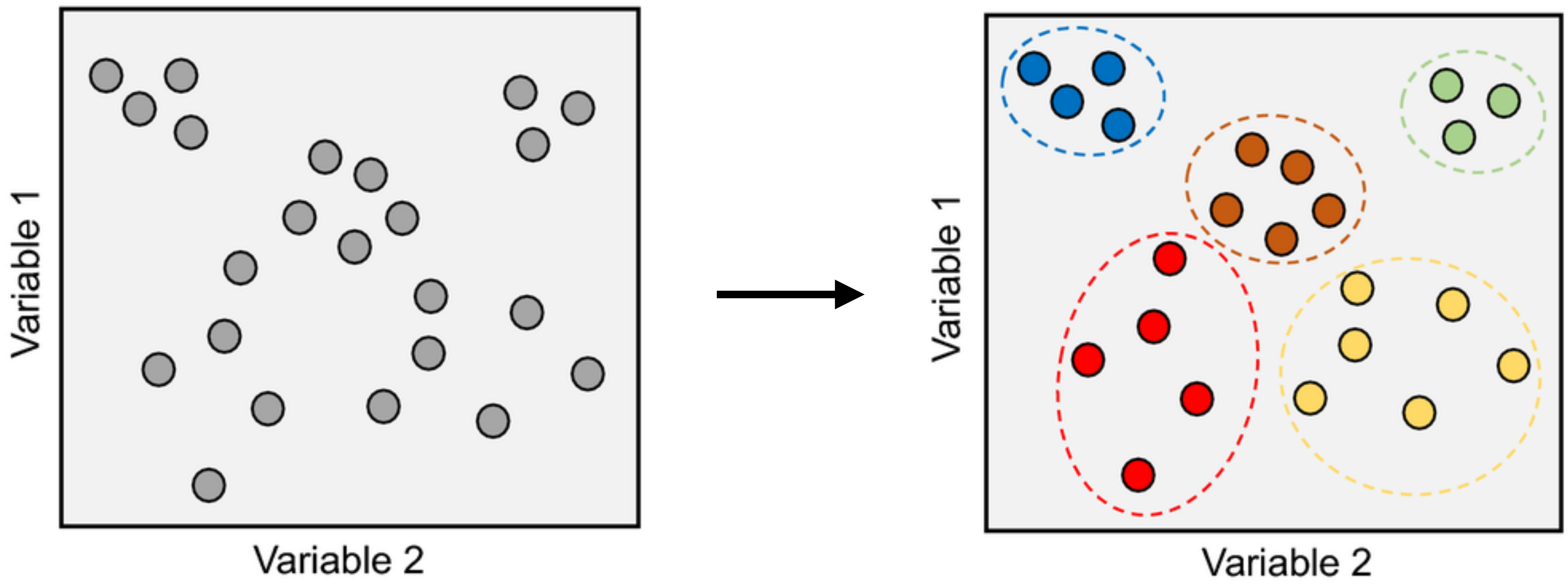
Categories: Supervised Learning

- We have many input-output pairs $D = \{(X_i, Y_i)\}_{i=1}^{n}, (X_i, Y_i) \sim P_{XY}$
  - Learn the input-to-output mapping
    (e.g., learning to predict the color of points)



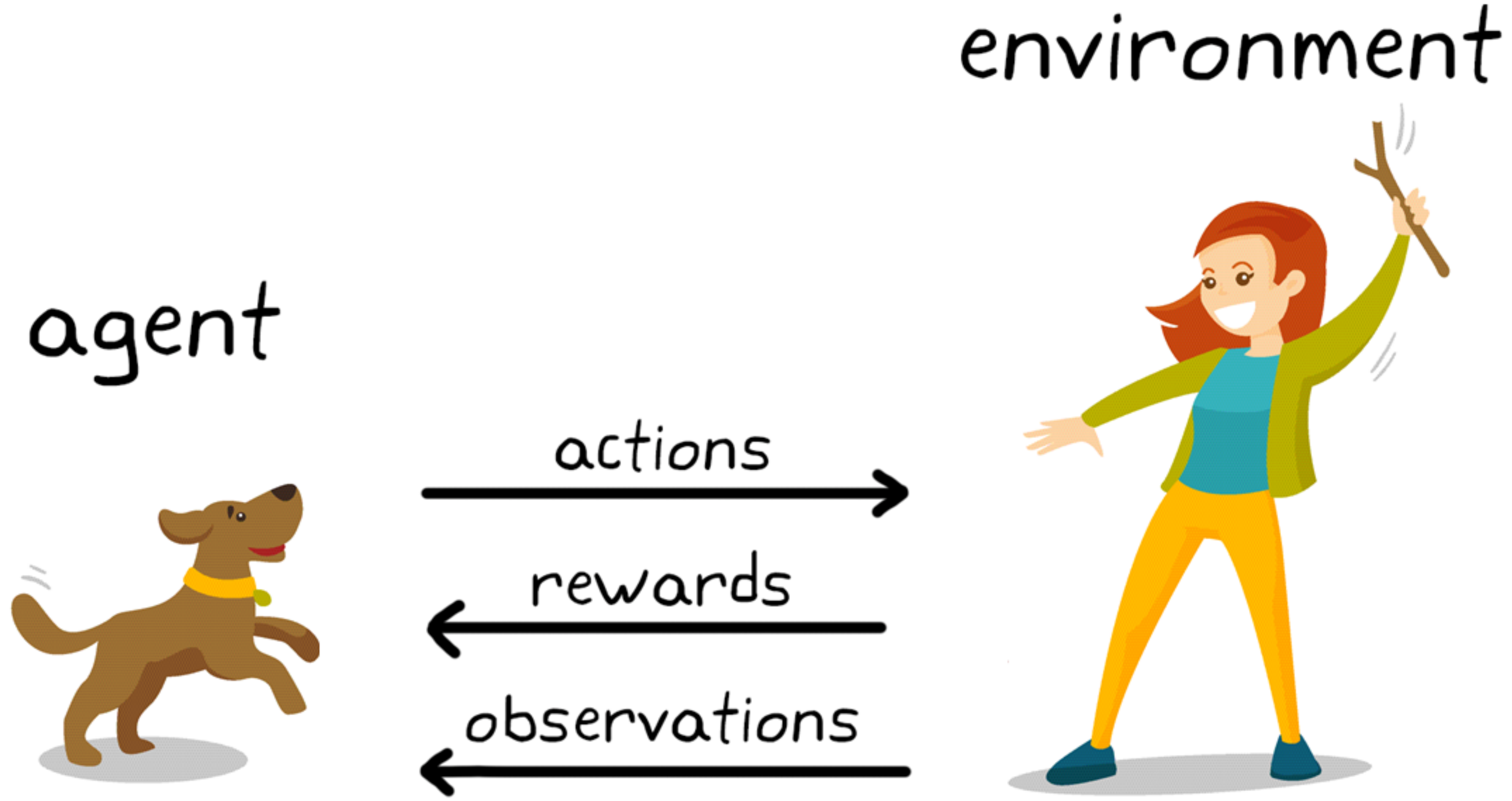
Categories: Unsupervised Learning

- We have many **unlabeled input data** $D = \{X_i\}_{i=1}^{n}, X_i \sim P_X$
  - Learn useful **structures of data** — or virtual labels (e.g., identifying clusters of data)
  - The structures may be useful for downstream tasks



Categories: Reinforcement Learning

- We have an **environment** that we can interact with:
  - Can collect **sequence of interactions** with the environment
    - Actions, Rewards, States
  - Learn an **interaction policy (i.e., agent)** that maximizes the reward
    - Somewhat specialized; we'll discuss this as a special topic later



Supervised Learning

Supervised Learning

- For now, let's focus on the **supervised learning** scenario:
  - We have two random variables: X, Y
    - Jointly distributed as some probability distribution $P_{XY}(x, y)$

- Consider a simple prediction task, where
  - X is easy to collect (called features)
    - e.g., natural images
  - Y is costly to acquire (called labels)
    - e.g., human-written labels

Supervised Learning

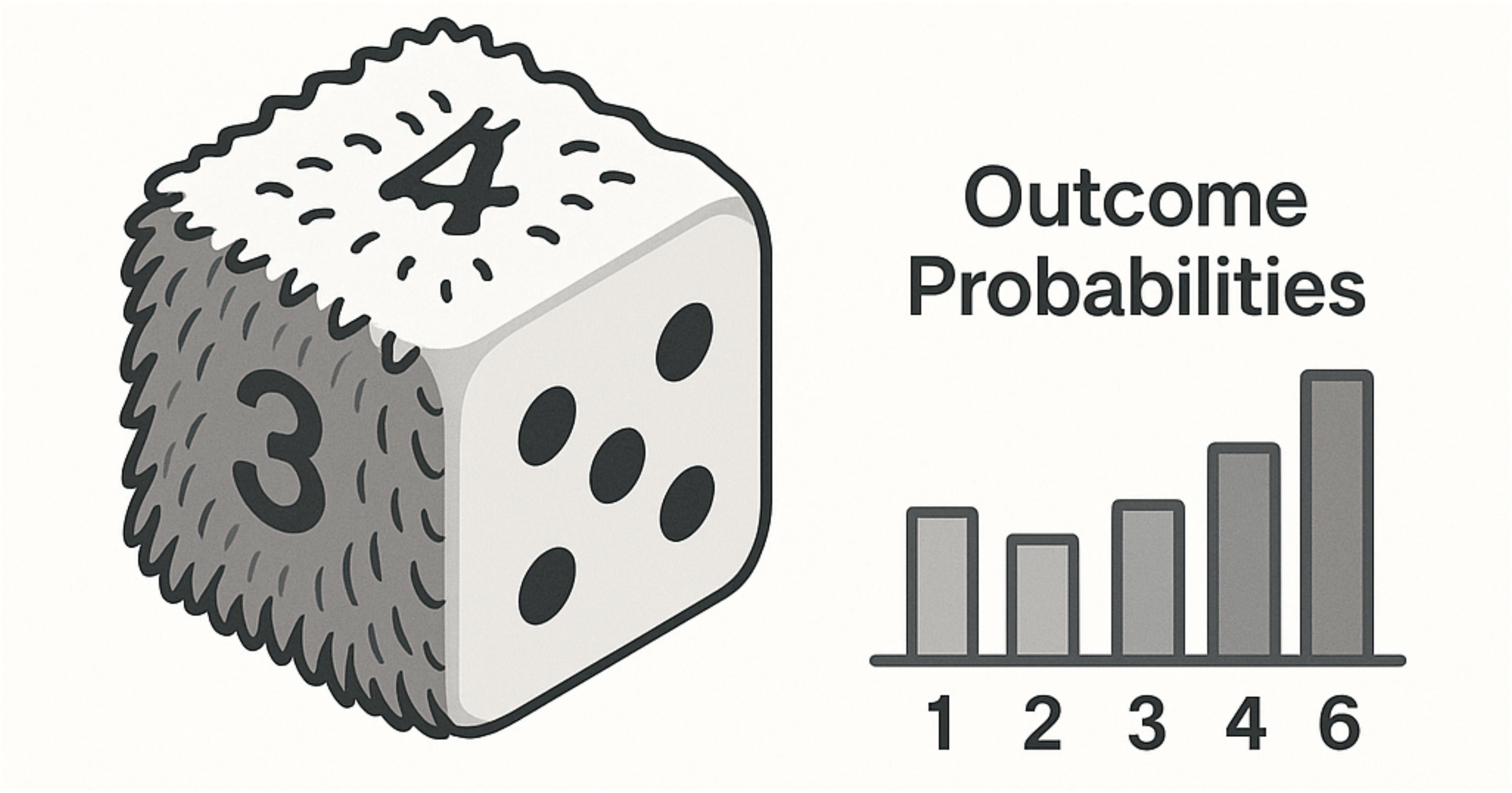
- **Goal.** Given some X, predict the associated label Y
- **Challenge.** We <u>do not know</u> the joint distribution $P_{XY}(x, y)$
  - Instead, we only have access to some data
  - If we knew: we can get the posterior distribution…

$$P_{Y|X}(y \mid x) = \frac{P_{XY}(x, y)}{P_X(x)} = \frac{P_{XY}(x, y)}{\int_y P_{XY}(x, y)\,\mathrm{d}y}$$

Warm-up quiz. Given $P_{Y|X}$, do you know what to do?

Quiz: Estimation Basics

- Imagine a **biased die**
  - Its face probabilities depend on the table condition, $X \in \mathbb{R}^d$
  - The event of each face coming up is represented by $Y \in \{1,2,\ldots,6\}$



Quiz: Estimation Basics

- We are given some weather ($X = x$)
  - Then, we can compute the probability of each face:

$$P_{Y|X}(1\,|\,x) = p_1, \quad P_{Y|X}(2\,|\,x) = p_2, \quad \ldots, \quad P_{Y|X}(6\,|\,x) = p_6$$

- **Question.** Given these, how will you predict the outcome $\hat{Y}$, if you want to:
  - Maximize the probability of being wrong?
  - Minimize the expected error $\mathbb{E}[(\hat{Y} - Y)^2]$?
  - Simulate the (random) outcome Y of the die?

Quiz: Estimation Basics

- **Answer.** we can construct, e.g.,
  - **Maximum a posteriori estimate (MAP)**, for discrete Y

$$\hat{y} = \arg\max_{y} P_{Y|X}(y \mid x)$$

  - **Minimum Mean-Squared Error (MMSE)**, for continuous Y

$$\hat{y} = \mathbb{E}[Y \mid X = x] = \int_{y} y \cdot P_{Y|X}(y \mid x) \, \mathrm{d}y$$

  - **Sampling** a solution, for diverse generation / prediction

$$\hat{y} \sim P_{Y|X}(\cdot \mid x)$$

- You can construct similar estimates, e.g., to minimize $\mathbb{E}[\,|\hat{Y} - Y|\,]$.

Two approaches in ML

- Now let's get back on track — in learning, we do not know P_{XY}

- Instead, we want to use **training data** to build an estimate

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

- **Question.** How can we do this in a principled way?
  - Generative
  - Discriminative

Two approaches in ML: Generative

- **Generative** approach aims to directly model P_{XY}
  - i.e., capturing the data generation process itself
  - Can be used to construct $\hat{P}_{Y|X}$
  - Examples: Naïve Bayes, VAE, GAN, Diffusion

👍 Once it works well, many other perks

- Generation (P_X), Conditional Generation ($P_{X|Y}$), quantify the uncertainty of prediction $P_{Y|X}(\hat{y} \mid x)$, …

👎 Very difficult to achieve — a lot of data, or heavy assumption

Two approaches in ML: Discriminative

- **Discriminative** approach aims to model $P_{Y|X}$
  - Often model the estimates based on $P_{Y|X}$ (e.g., MAP), not itself
  - Example: Logistic regression, SVM, neural net classifiers

👍 Can learn with relatively less samples

- Usually better accuracy on the target task

👎 Cannot generate data, potentially poor calibration, limited use case

Note: In DL, people used to work on D (~2019), then moved onto G

Learning as an optimization

Learning as an optimization

- Now, let's focus on the **discriminative** case for **supervised learning**
  - That is, we have a bunch of data

$$\{(X_i, Y_i)\}_{i=1}^n \sim P_{XY}$$

  - And our goal is to find a nice model

$$\hat{P}_{Y|X}(y \mid x)$$

    which fits this data the best.

- This optimization is what learning algorithm does

Learning as an optimization

- **Question.** How exactly do we do this?
  - **Answer.** Differs from algorithm to algorithm (sadly)

- There are two unified perspectives toward various ML algorithms
  - Statistical learning
  - Bayesian approach

- Note: These two are — to some degree — interchangeable

Learning as an optimization: Statistical learning

Statistical Learning

- Under the statistical learning paradigm, each learning algorithm is characterized by three elements:
  - Hypothesis space
  - Loss function
  - Search algorithm

Statistical Learning

- **Hypothesis space.** A bag of models

$$\mathcal{F} = \left\{ f_\theta(\,\cdot\,) \mid f_\theta : \mathcal{X} \to \mathcal{Y}, \quad \theta \in \Theta \right\}$$

  - $\mathcal{X}$: set of all possible X (e.g., set of all 256 x 256 images)
  - $\mathcal{Y}$: set of all possible Y (e.g., set of all labels)
  - θ: parameters (which we optimize for)

- **Example.** Set of all affine models

$$f_\theta(x) = Wx + b, \qquad \theta = (W, b), W \in \mathbb{R}^{m \times d}, b \in \mathbb{R}^m$$

Statistical Learning

- **Loss function.** A measure of "wrongness" of the model prediction:

$$\ell(\cdot,\cdot) : \mathscr{Y} \times \mathscr{Y} \to \mathbb{R}$$

  - If we get a sample (X^*, Y^*), the loss of a predictor f_θ is:

$$\ell(f_\theta(X), Y)$$

- **Example.** Squared loss

$$\ell(\hat{Y}, Y) = \|\hat{Y} - Y\|_2^2$$

  Zero-one loss

$$\ell(\hat{Y}, Y) = \mathbf{1}\{\hat{Y} = Y\}$$

Statistical Learning

- Before we describe the **search algo**, let us first formalize our final goal:

- **Objective.** Given the hypothesis space and the loss, our goal is to solve:

$$\min_{\theta \in \Theta} \mathbb{E}_{(X,Y)\sim P_{XY}}[\ell(f_\theta(X), Y)]$$

  - That is, we want to find the function f_θ which has the smallest average loss on the test sample (X, Y)

Statistical Learning

- **Problem.** We don't know P_{XY}

- **Idea.** We conduct Empirical Risk Minimization (ERM):
  - That is, we find the model which achieves the minimum **average loss on training dataset**:

$$\min_{\theta \in \Theta} \mathbb{E}_{(X,Y)\sim D}[\ell(f_\theta(X), Y)] = \min_{\theta \in \Theta} \left[\frac{1}{n}\sum_{i=1}^{n} \ell(f_\theta(X_i), Y_i)\right]$$

Statistical Learning

- **Rationale.** If we have enough samples, the law of large numbers say that

$$\frac{1}{n}\sum_{i=1}^{n} \ell(f_\theta(X_i), Y_i) \xrightarrow{n\to\infty} \mathbb{E}[\ell(f_\theta(X), Y)]$$

for any fixed θ

- Thus, the empirical loss can be a good proxy of the population loss
  - Caveat: LLN requires independent(-ish) draws of the samples!

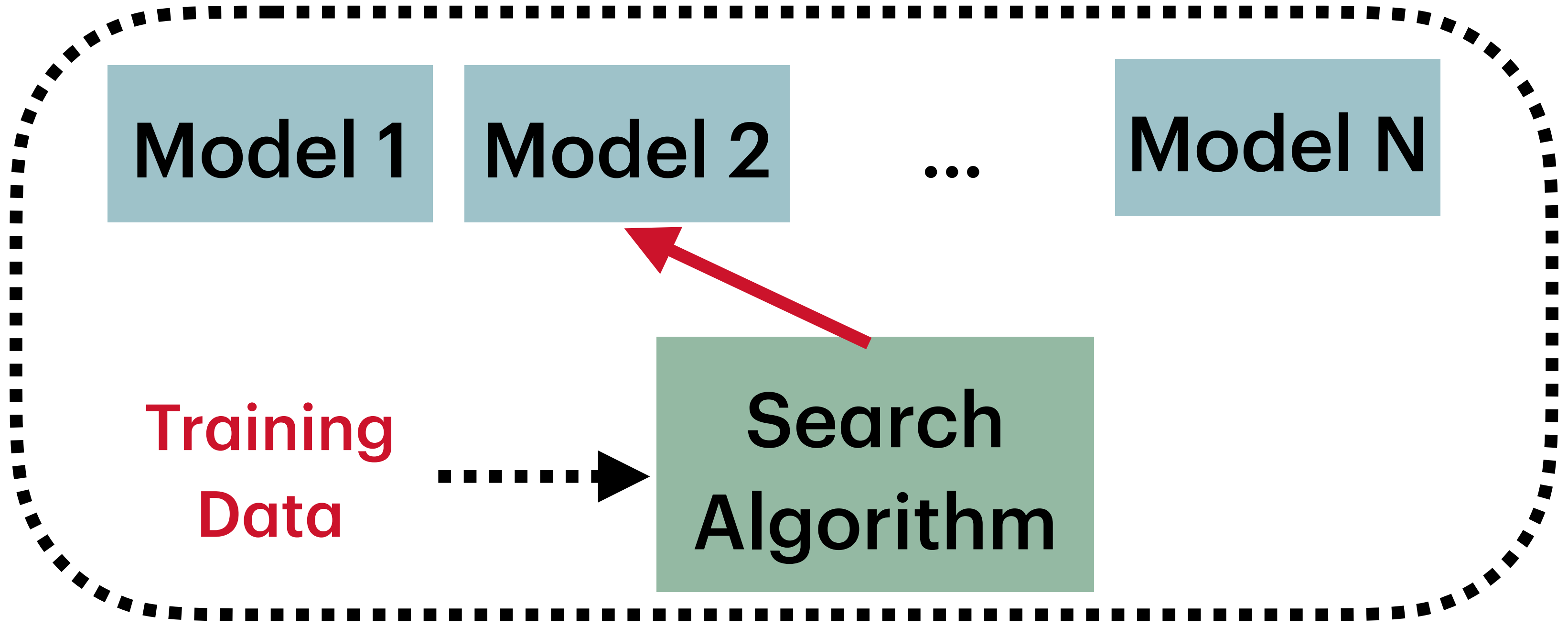
Statistical Learning

- **Search algorithm.** How we solve this ERM optimization

$$\min_{\theta \in \Theta} \mathbb{E}_{(X,Y)\sim D}[\ell(f_\theta(X), Y)] = \min_{\theta \in \Theta} \left[\frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(X_i), Y_i) \right]$$

- **Example.**
  - Analytical solution
  - Solve by iterative optimization (e.g., SGD)

(more on this later)



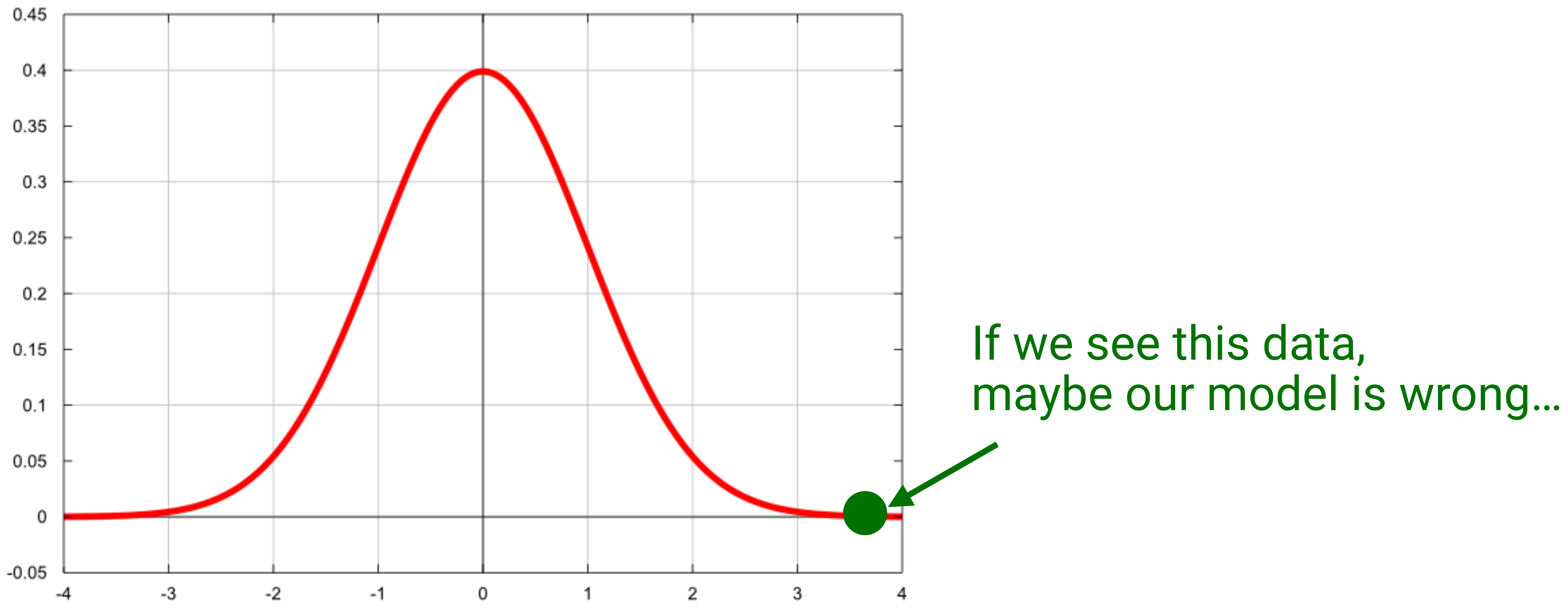
Learning as an optimization: Bayesian perspective

Bayesian approach

- Bayesians prefer a generative explanation:

“If our model is correct, the probability of our model generating the data would be high.”

(called the “maximum likelihood principle”)



Bayesian approach

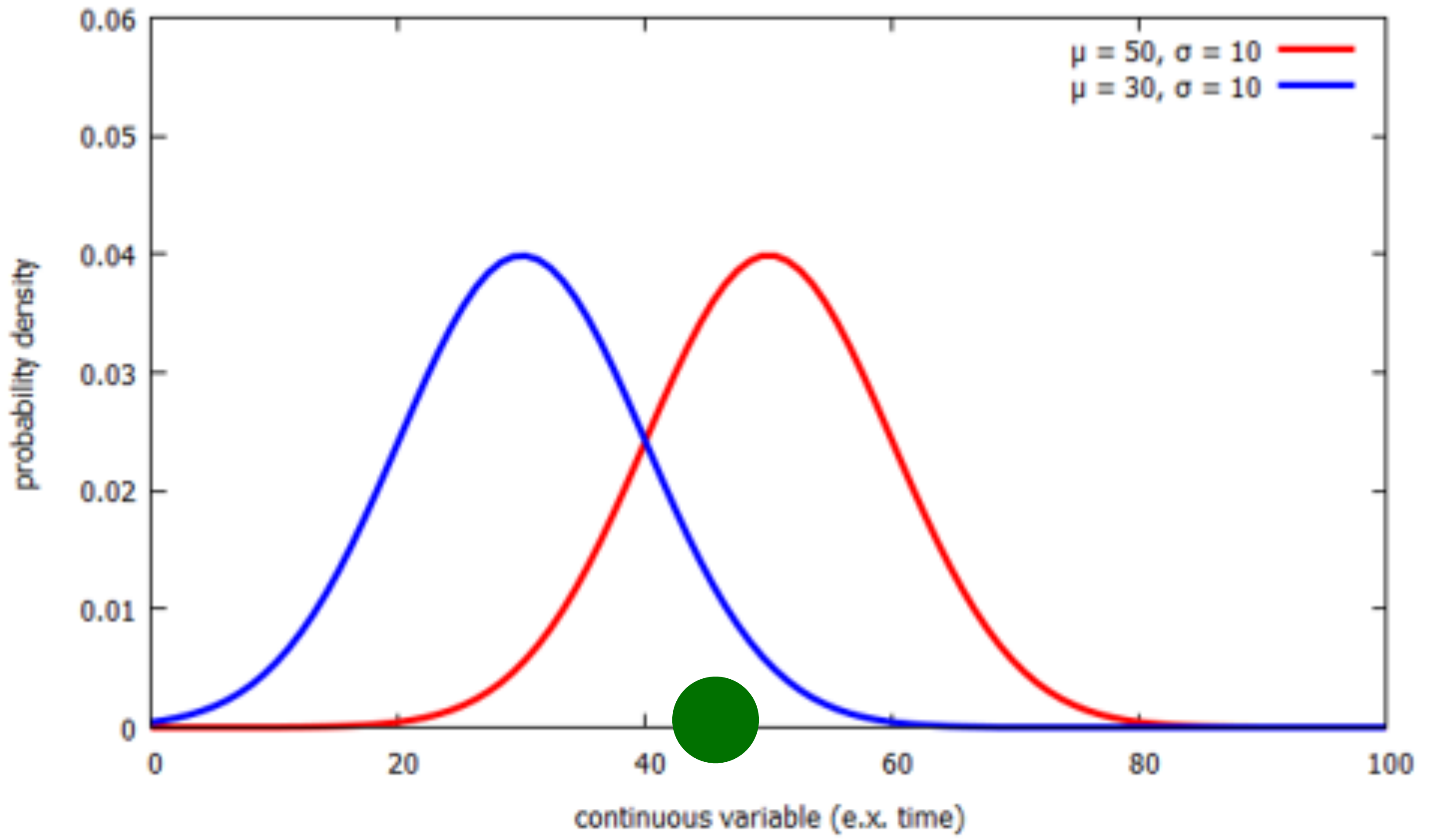
- This principle provides a mean to compare two models:
- **Example.** Suppose that we have two “models”

$$P^{(1)}_{XY}(x,y), \qquad P^{(2)}_{XY}(x,y)$$

  - Suppose that we are given one sample: (X^*, Y^*)
    - If we have

$$P^{(1)}_{XY}(X^*, Y^*) > P^{(2)}_{XY}(X^*, Y^*)$$

then $P^{(1)}$ is more likelier to be correct!



Bayesian approach

- Suppose that we have a family of parametrized joint distributions

$$P_\theta(x, y) = P_\theta(x) P_\theta(y \,|\, x), \qquad \theta \in \Theta$$

- Goal. Find θ maximizing the probability of generating all training data:

$$\max_\theta P_\theta((X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n))$$

  - If all training data are independently drawn, we know that this is:

$$\max_\theta \left(\prod_{i=1}^{n} P_\theta(X_i, Y_i) \right)$$

forgive me for the abuse of notation ;)

Bayesian approach

- We can apply $\log(\,\cdot\,)$ to make things look simpler:

$$\max_{\theta} \sum_{i=1}^{n} \log P_{\theta}(X_i, Y_i)$$

  - This is what we call the maximum log-likelihood solution

- We can break down P_{θ} and write:

$$\max_{\theta} \left(\sum_{i=1}^{n} \log P_{\theta}(Y_i \,|\, X_i) + \sum_{i=1}^{n} \log P_{\theta}(X_i) \right)$$

  - For simplicity, ignore the second term

Bayesian approach

- Notice that this is **similar to doing ERM:**

$$\max_{\theta} \left(\sum_{i=1}^{n} \log P_\theta(Y_i \mid X_i) \right) \Leftrightarrow \min_{\theta} \left(\frac{1}{n} \sum_{i=1}^{n} \log \frac{1}{P_\theta(Y_i \mid X_i)} \right)$$

- If we have a nice loss and f_θ such that

$$\log \frac{1}{P_\theta(y \mid x)} = \ell(f_\theta(x), y)$$

then Bayesian approach reduces to ERM!

(many loss functions — e.g., cross-entropy — have this origin)

Summing up

- Most ML algorithms are **ERM**, with different choice of
  - Hypothesis space
  - Loss
  - Search algorithm

- But why are there so many algorithms?

Considerations of building an ML algorithm

Which algorithm should we use?

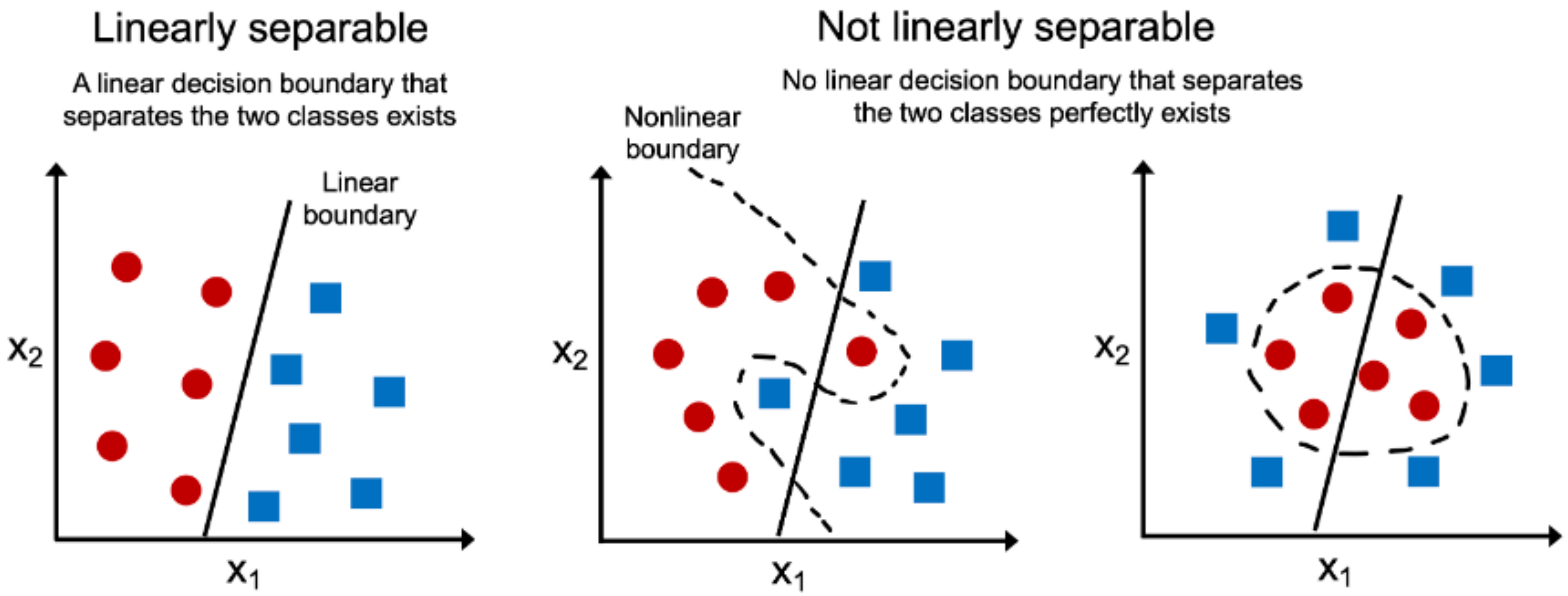
$$\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(X_i), Y_i) \quad \text{(+ regularizers)}$$

- Basically, designing the components of this optimization formula

Which algorithm should we use?

$$\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(X_i), Y_i) \quad \text{(+ regularizers)}$$

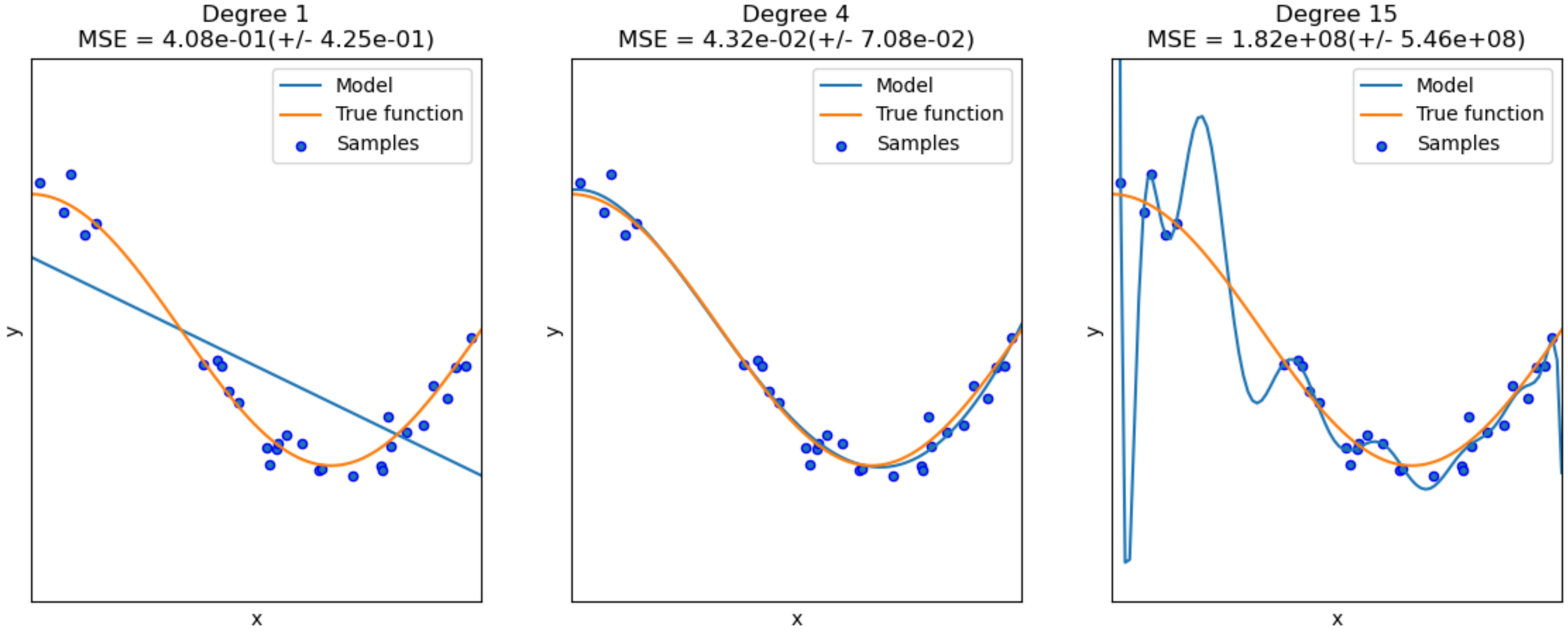
- **Model Size** (= Richness of $\mathscr{F}$)
  - If too small, even the best f_θ cannot fit the training data



Which algorithm should we use?

$$\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(X_i), Y_i) \quad \text{(+ regularizers)}$$

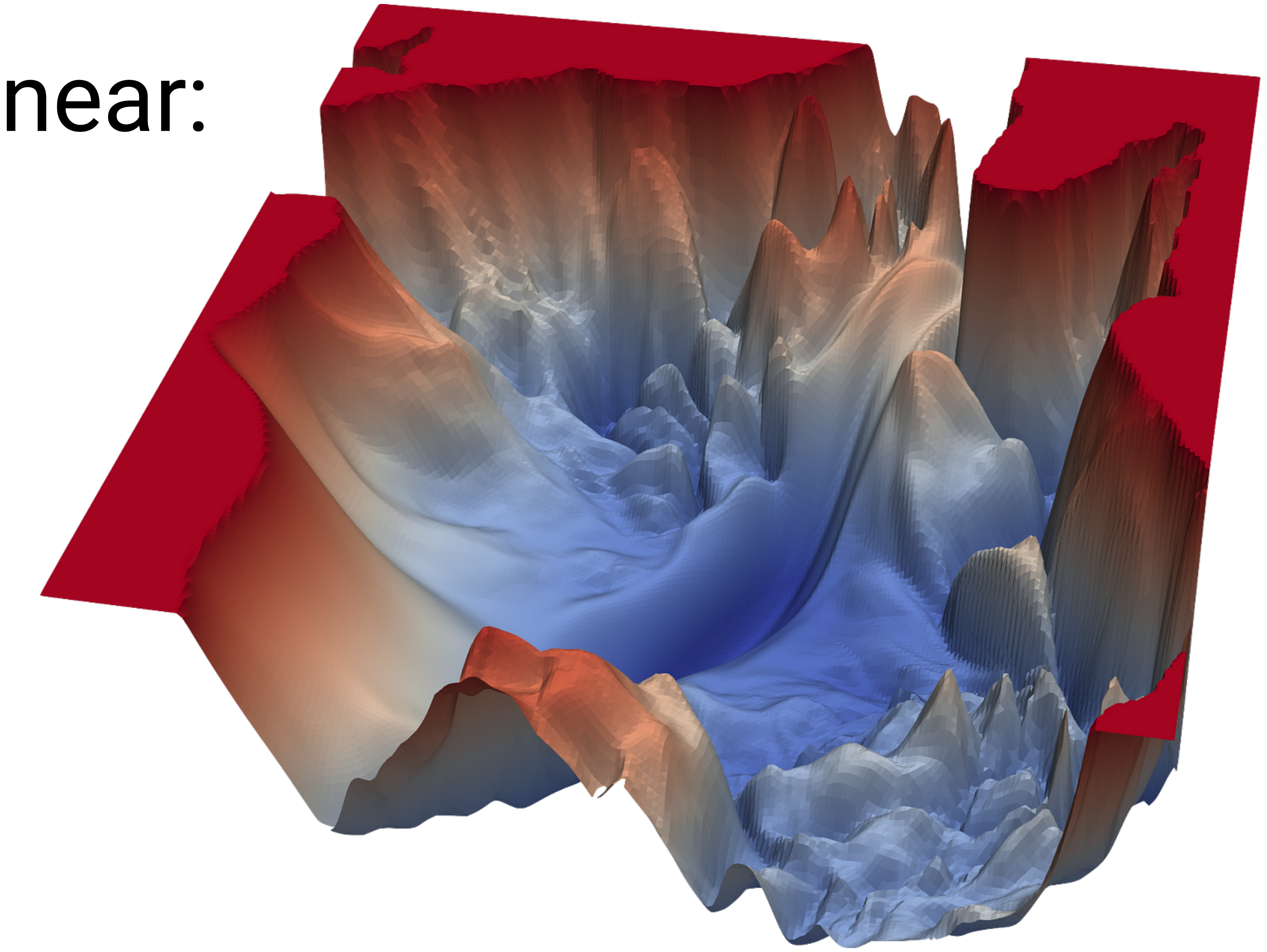
- **Model Size** (= Richness of $\mathscr{F}$)
  - If too large, can overfit the training data + large inference cost



Which algorithm should we use?

$$\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(X_i), Y_i) \quad \text{(+ regularizers)}$$

- Optimization (= Difficulty of solving ERM)
  - Need tailoring for each model class
  - If models are highly complicated & nonlinear:
    - Analytical solution unavailable
    - Takes a long time to solve



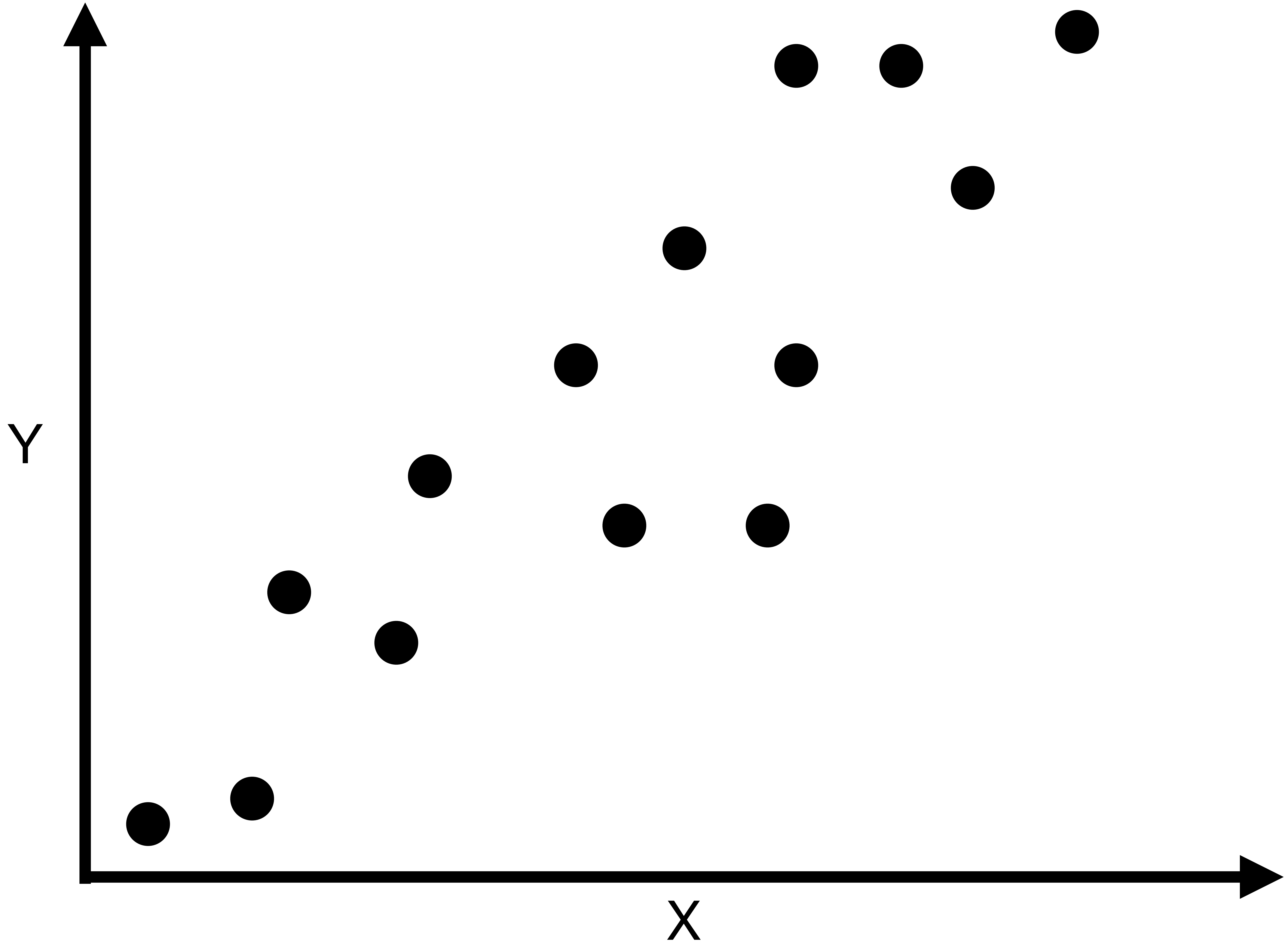
Which algorithm should we use?

$$\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(X_i), Y_i) \quad \text{(+ regularizers)}$$

- Loss & Regularizer
  - Affects the difficulty of optimization
    - e.g., non-continuous loss
  - Affects overfitting
    - e.g., penalizing model complexity

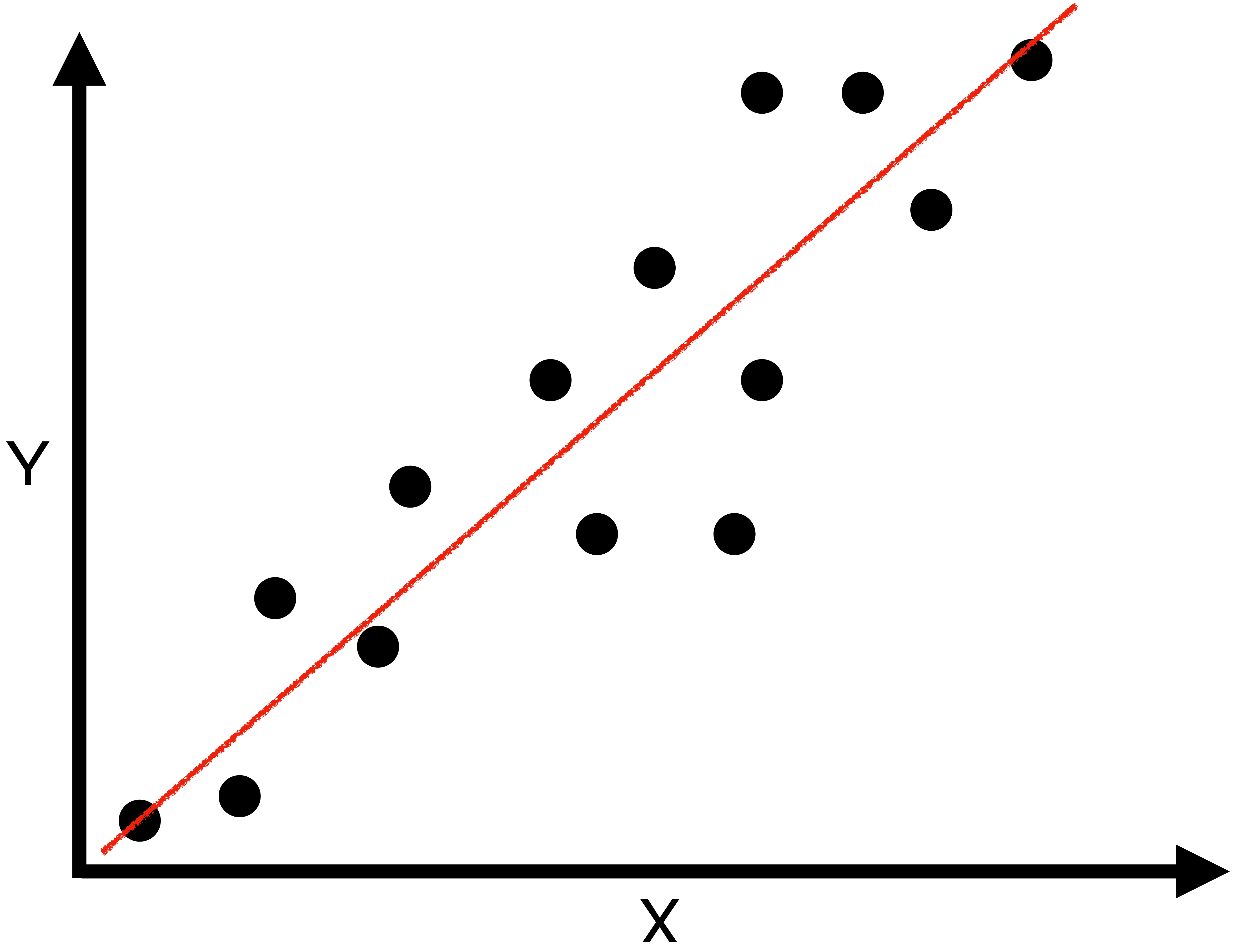
Designing the right model class

- Think about this data:



Designing the right model class

- We, as a human, may believe that this is a **straight line + noise**
  - This is due to our **inductive bias** —
    simpler solutions (in some sense) are more desirable



From the next class

- We study popular ML algorithms one-by-one
  - Each designed with different inductive bias
    - Different hypothesis space
    - Different optimization mechanism
    - Different loss / regularizer

- Note. Many of these choices heavily depend on tasks:
  - e.g., image vs text vs tabular