Lecture 6

Introduction to Machine Learning

Spring 2023

Bill Freeman, Vincent Sitzmann, Mina Konaković Luković

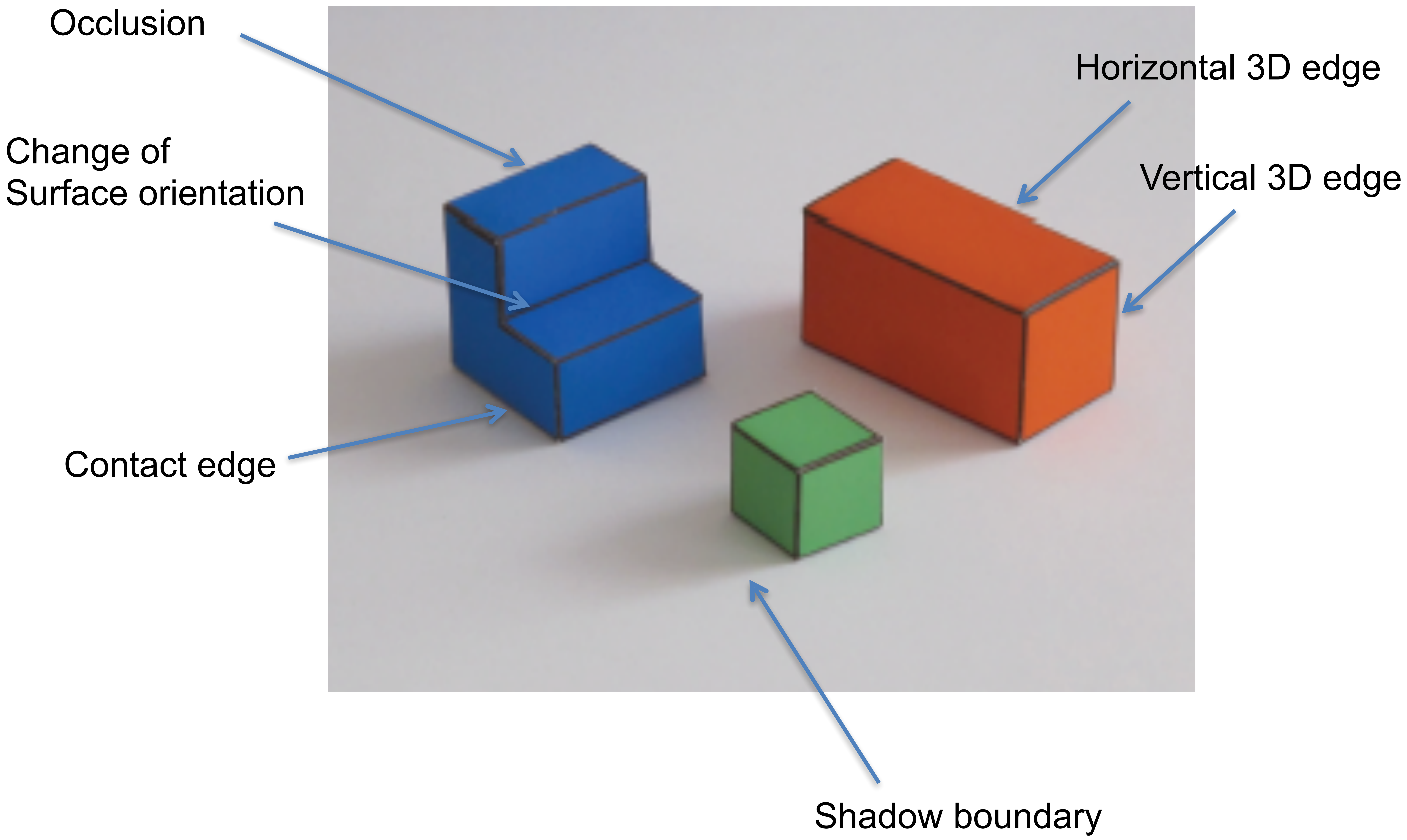
Announcements

- Pset 2 due on Thursday
- PyTorch tutorials Tues 4-5pm in 34-101 and Thurs 10-11am 34-101

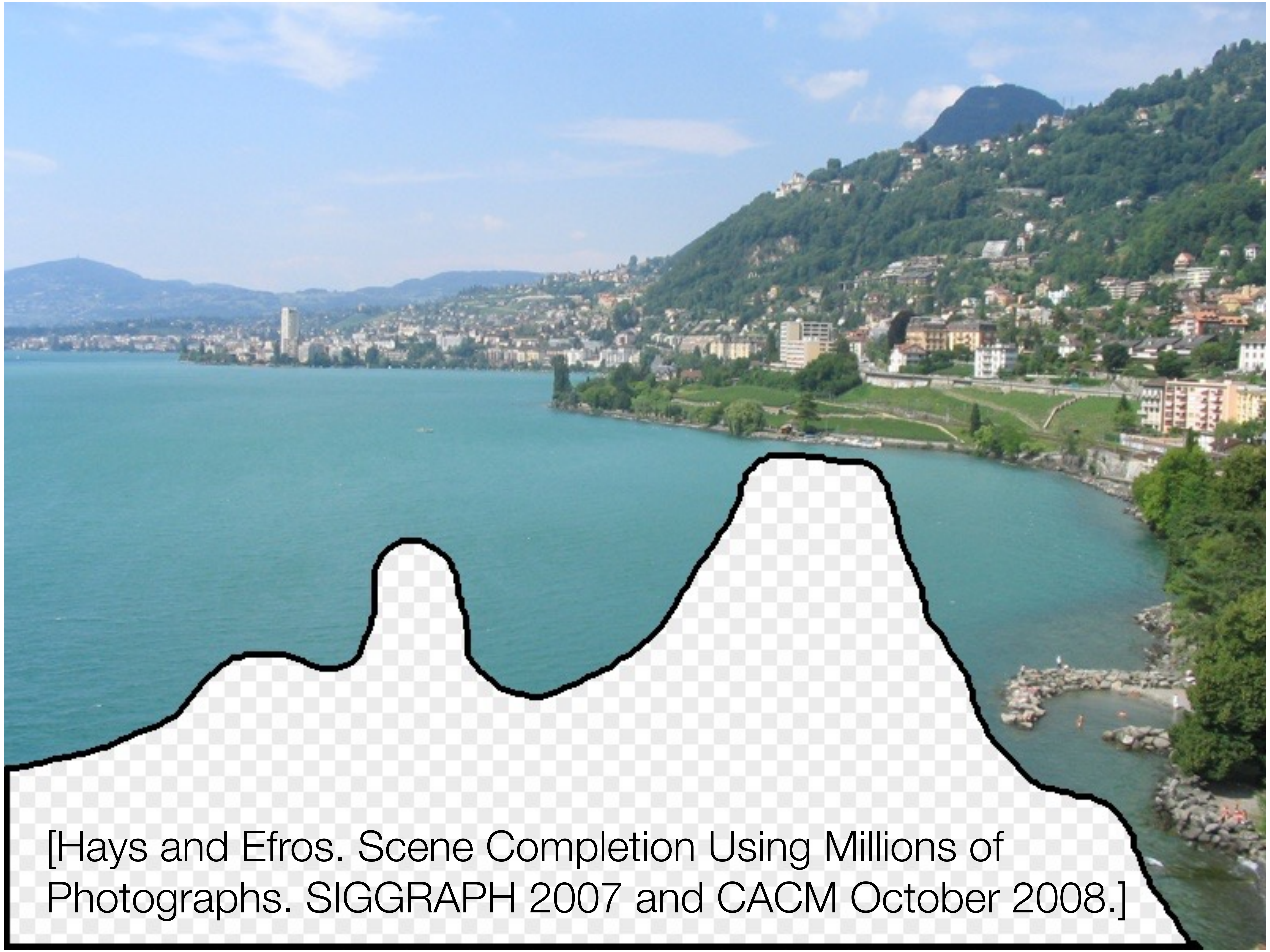
6. Introduction to Machine Learning

- "It's all about the data!"
- Formalisms of learning *(Data, Compute, Objective Function, Hypothesis Space, Optimizer)*
- Case study #1: Linear least-squares
    - Empirical risk minimization
- Case study #2: Image classification
    - Softmax regression
- The problem of generalization

Edges

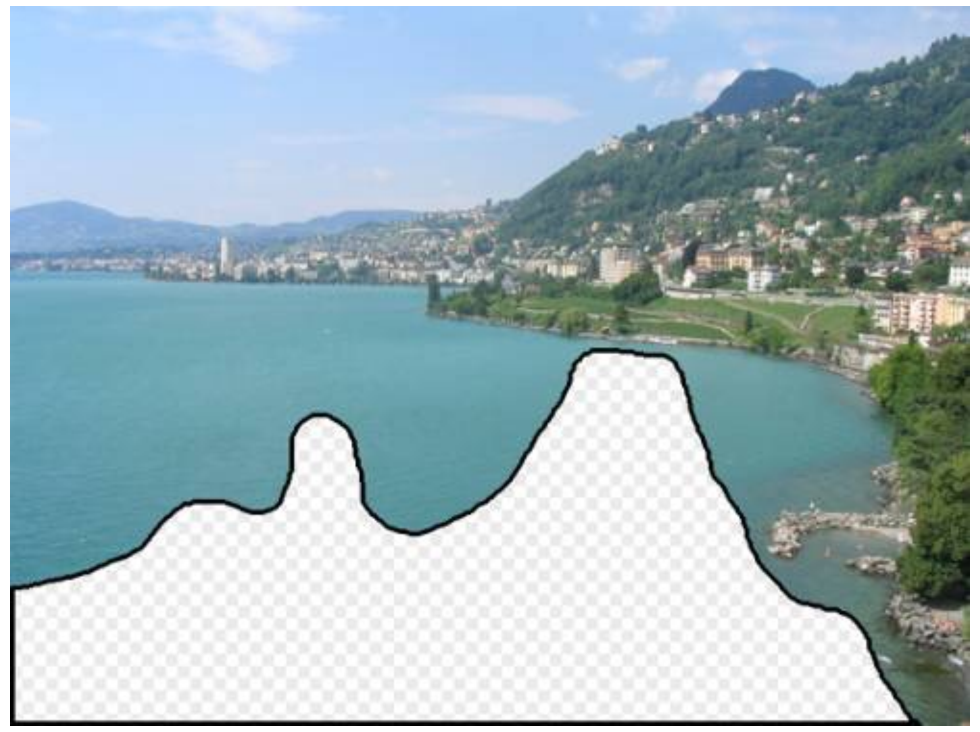

Occlusion
Change of
Surface orientation
Contact edge
Horizontal 3D edge
Vertical 3D edge
Shadow boundary

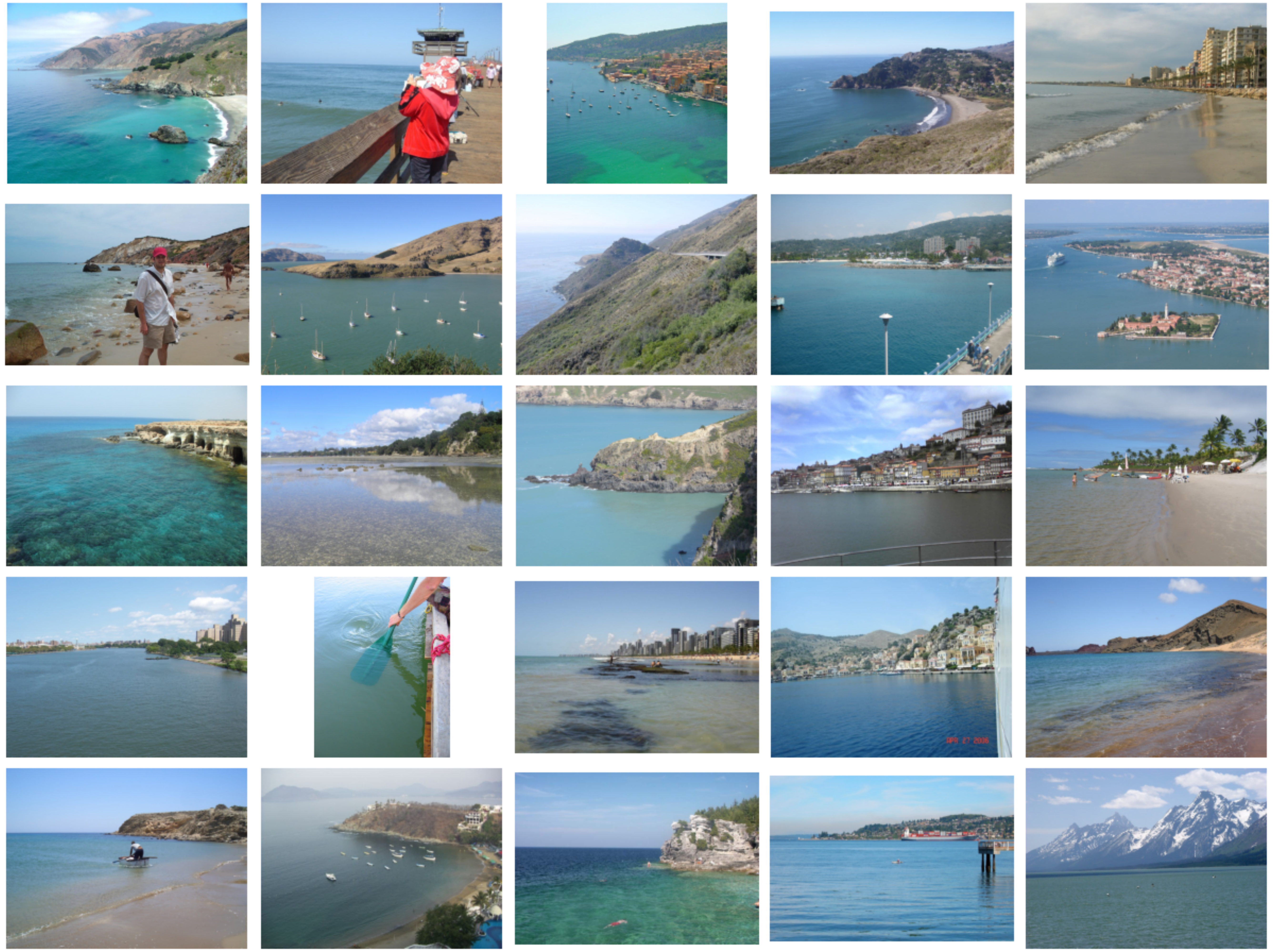
[Slide credit: Alexei Efros]



[Slide credit: Alexei Efros]

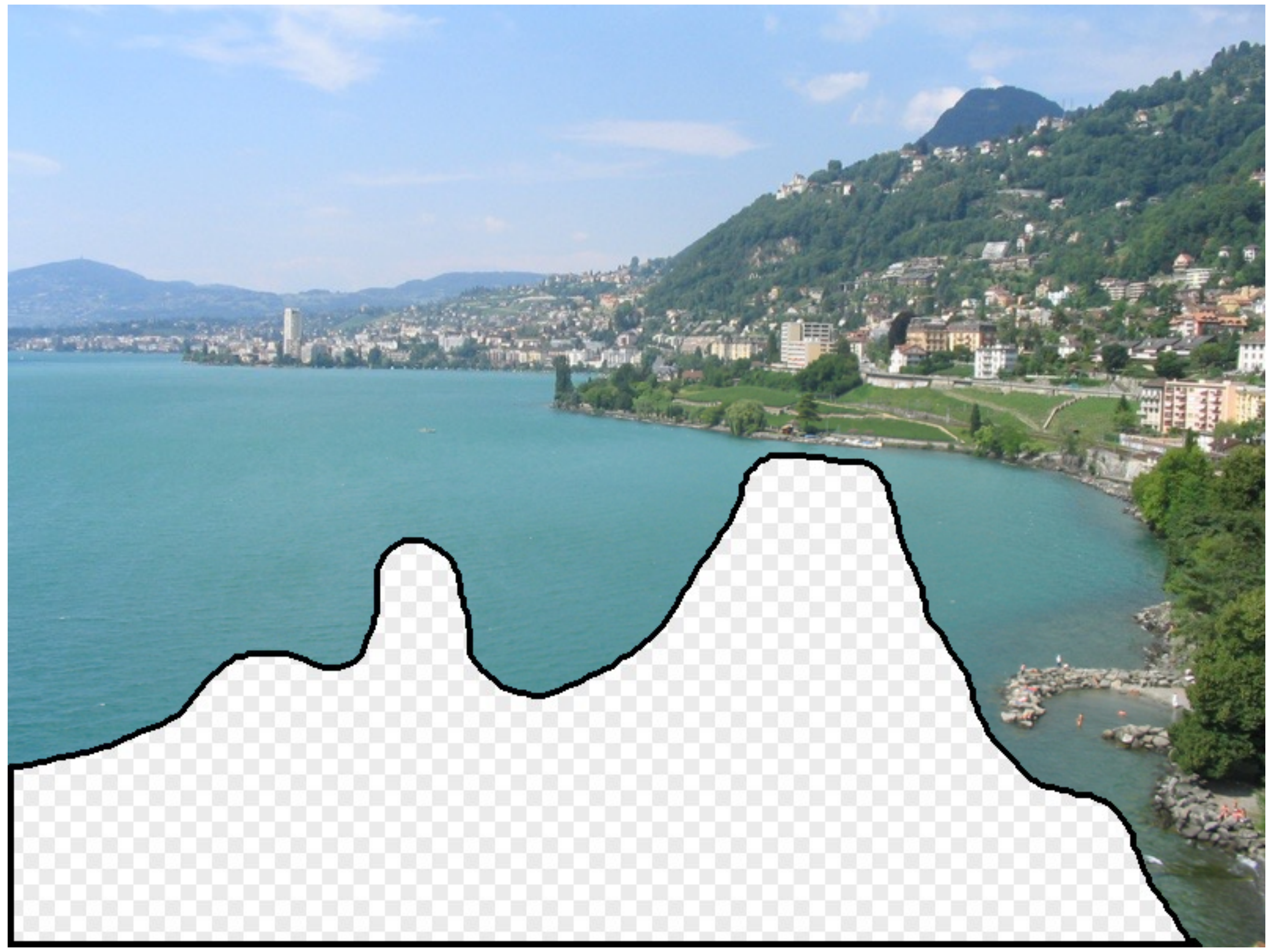
[Slide credit: Alexei Efros]





… 200 total

[Slide credit: Alexei Efros]



[Slide credit: Alexei Efros]

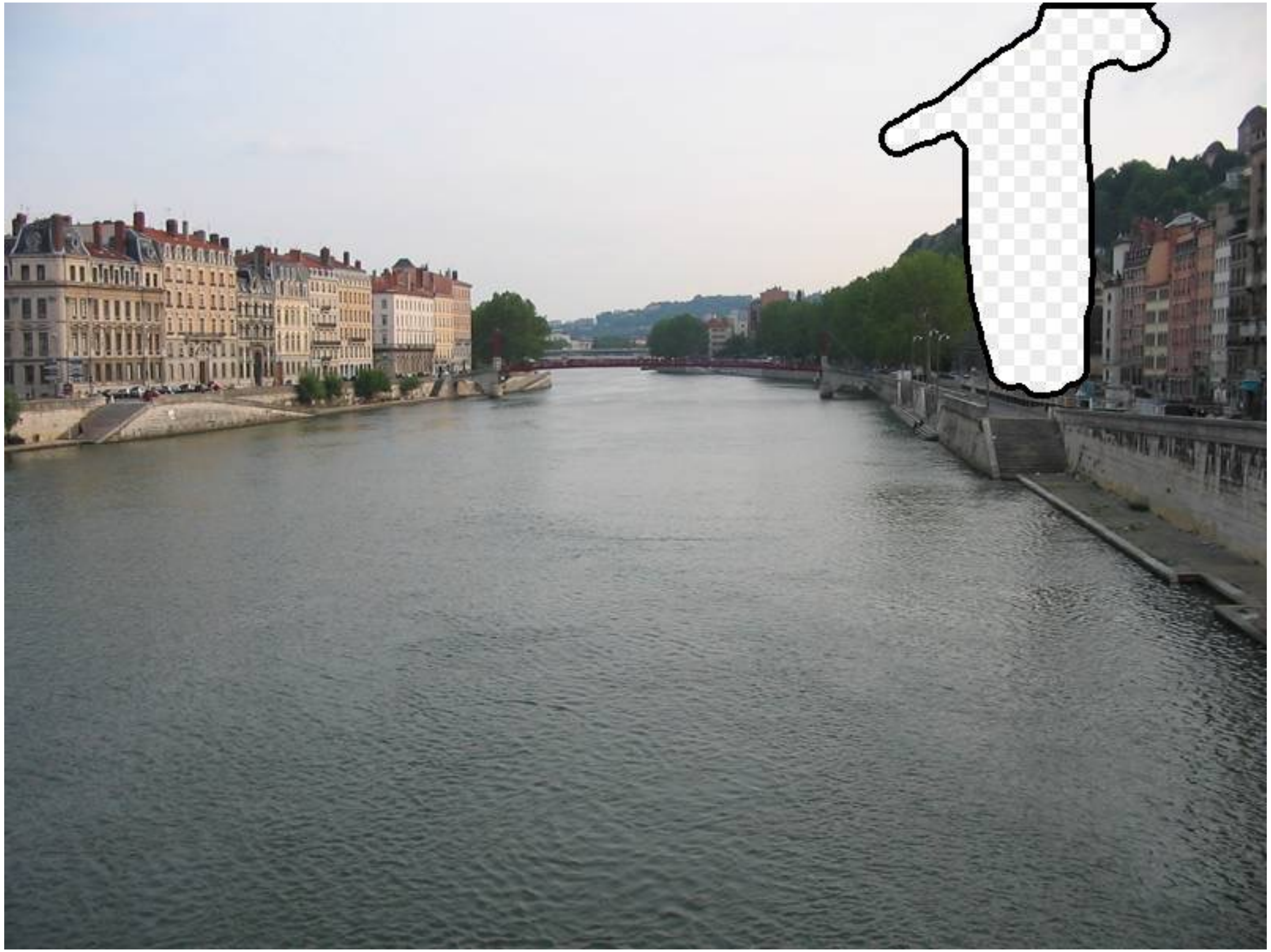
[Slide credit: Alexei Efros]

[Slide credit: Alexei Efros]

[Slide credit: Alexei Efros]

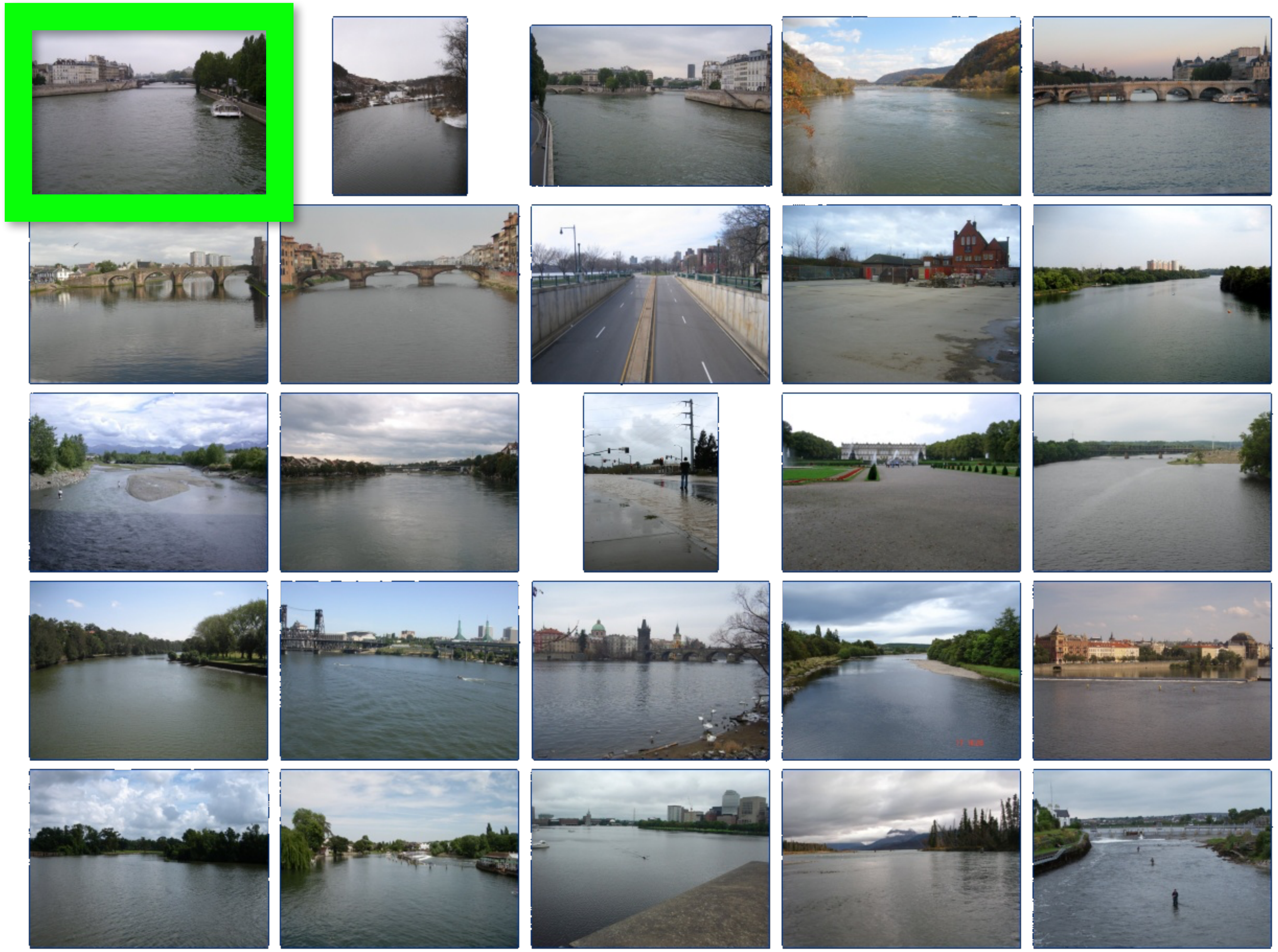
[Slide credit: Alexei Efros]

[Slide credit: Alexei Efros]



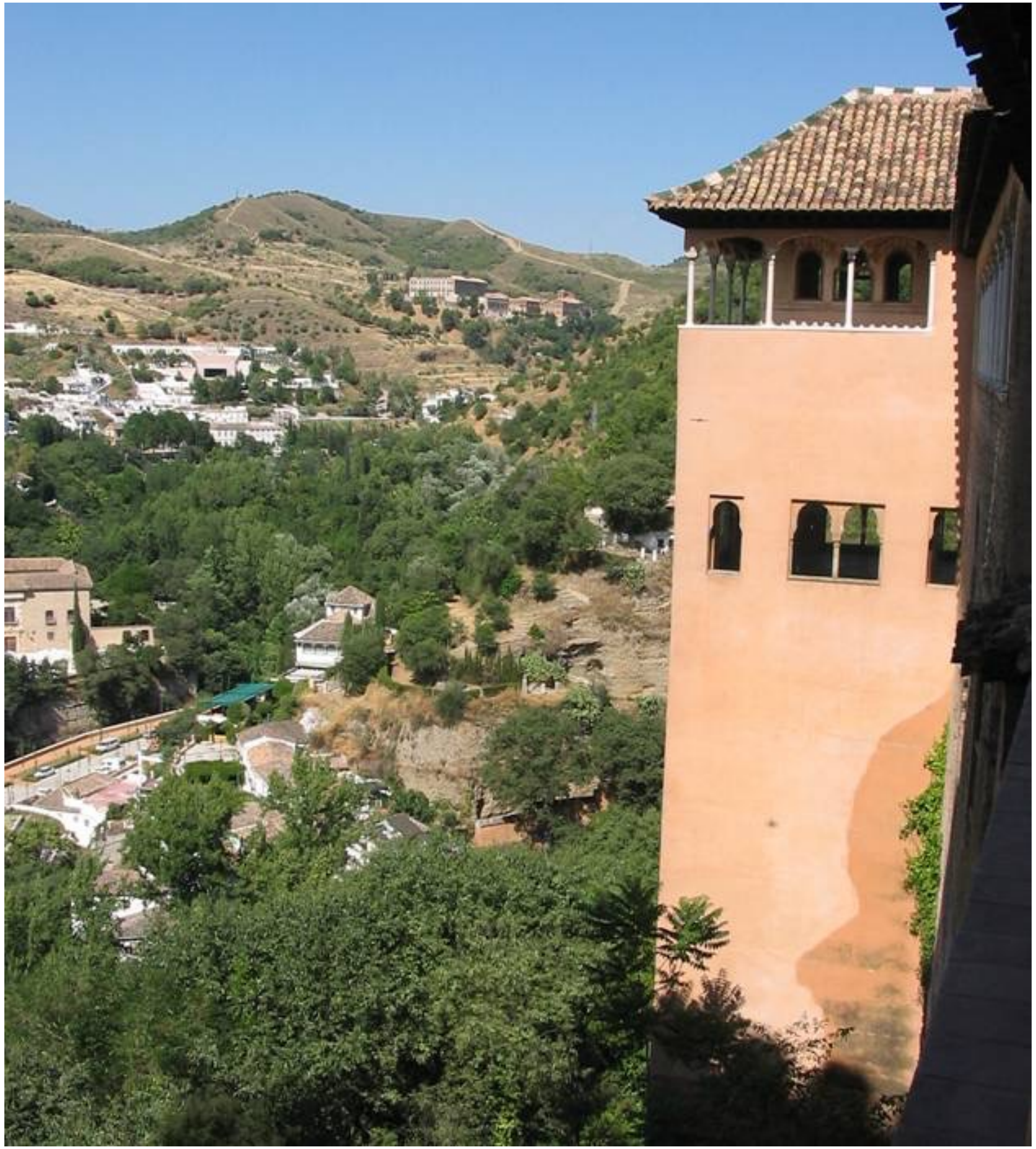
[Slide credit: Alexei Efros]

[Slide credit: Alexei Efros]

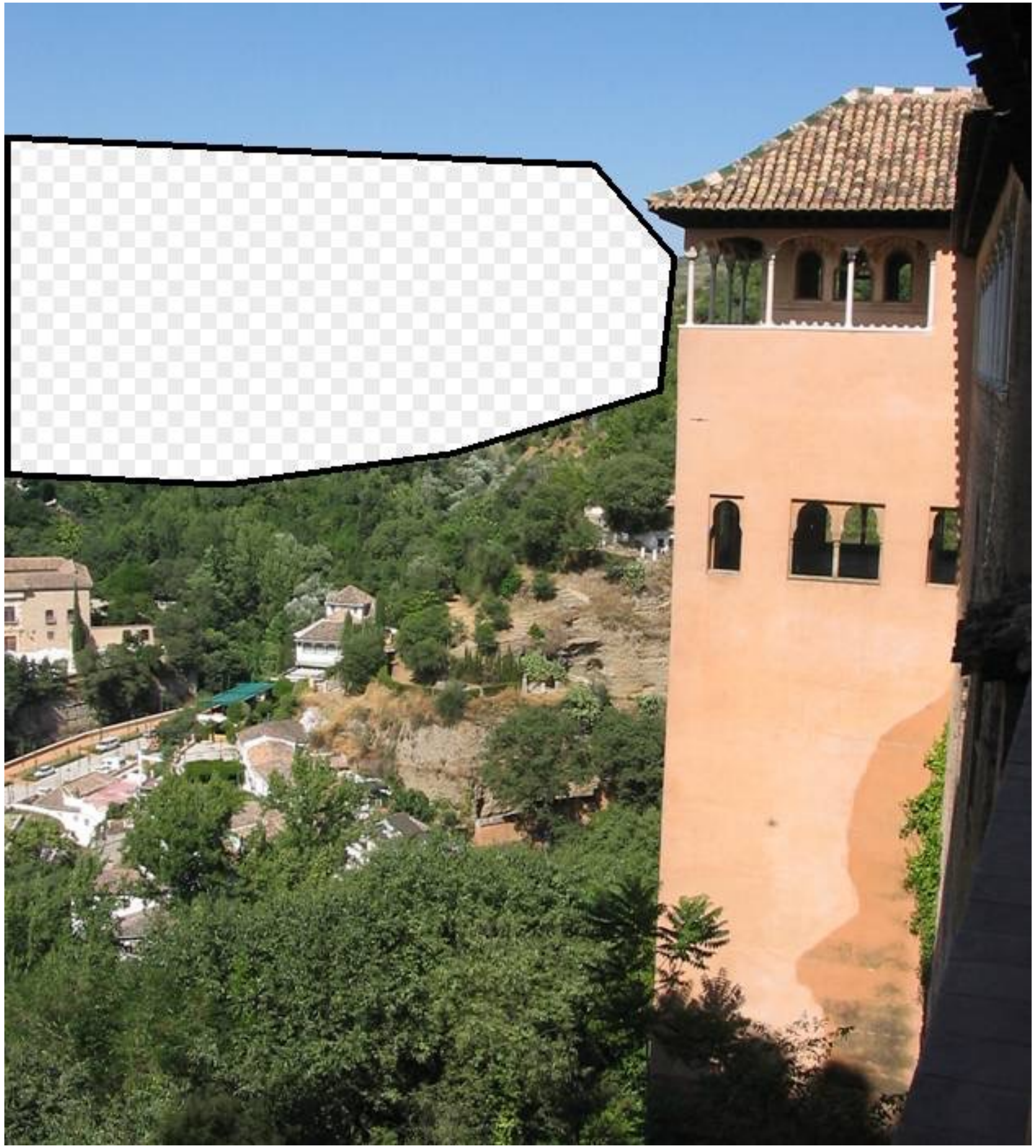


... 200 scene matches

[Slide credit: Alexei Efros]



[Slide credit: Alexei Efros]

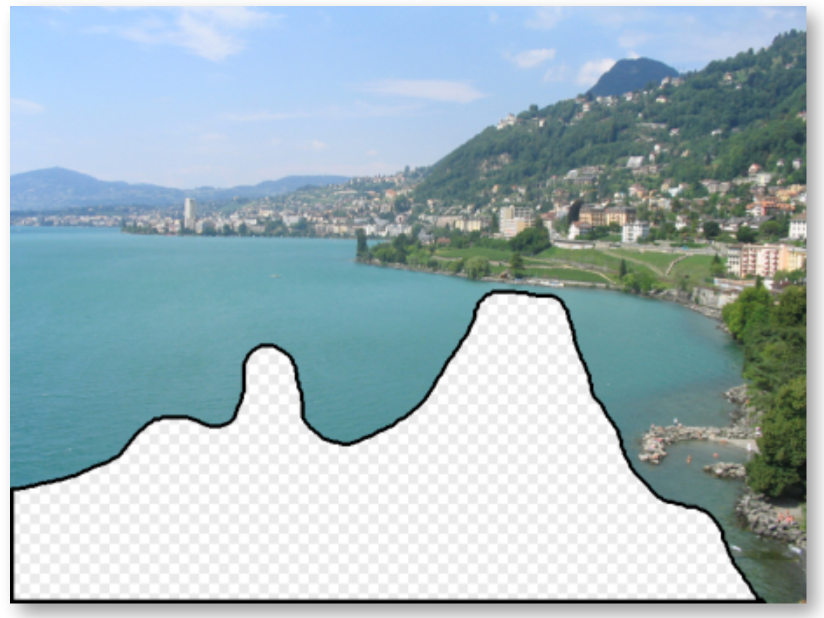


[Slide credit: Alexei Efros]

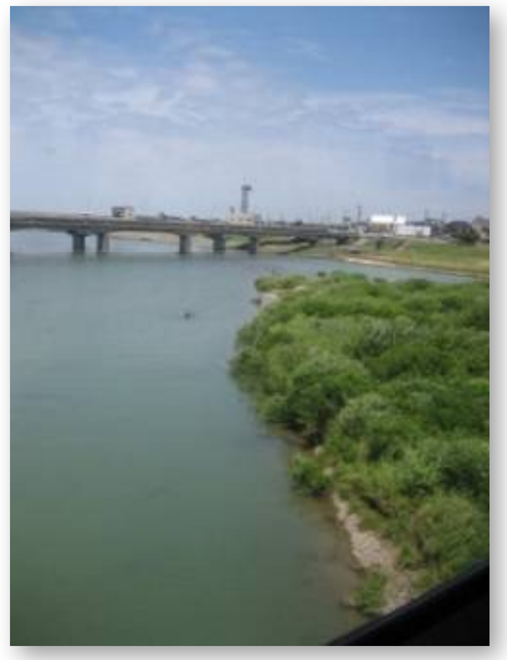
[Slide credit: Alexei Efros]

Why does it work?

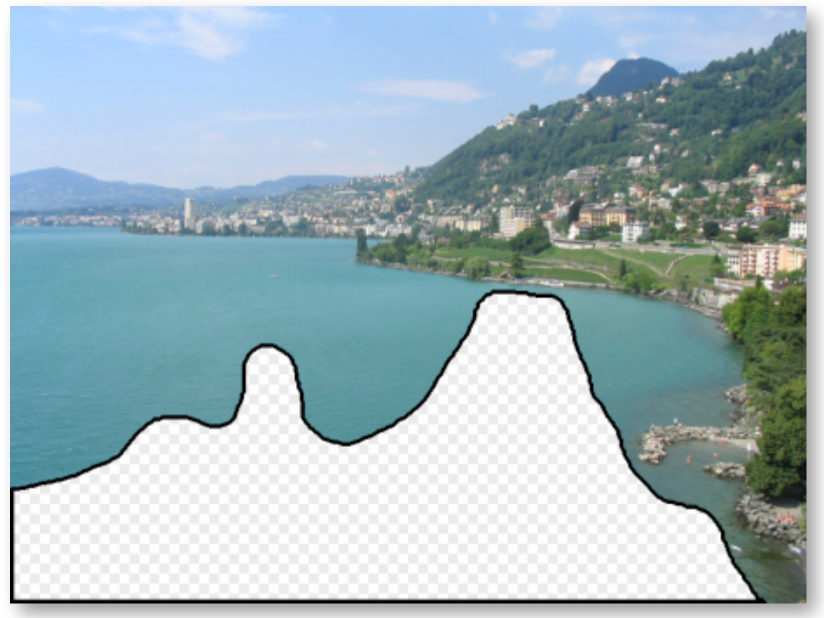
[Slide credit: Alexei Efros]



[Slide credit: Alexei Efros]



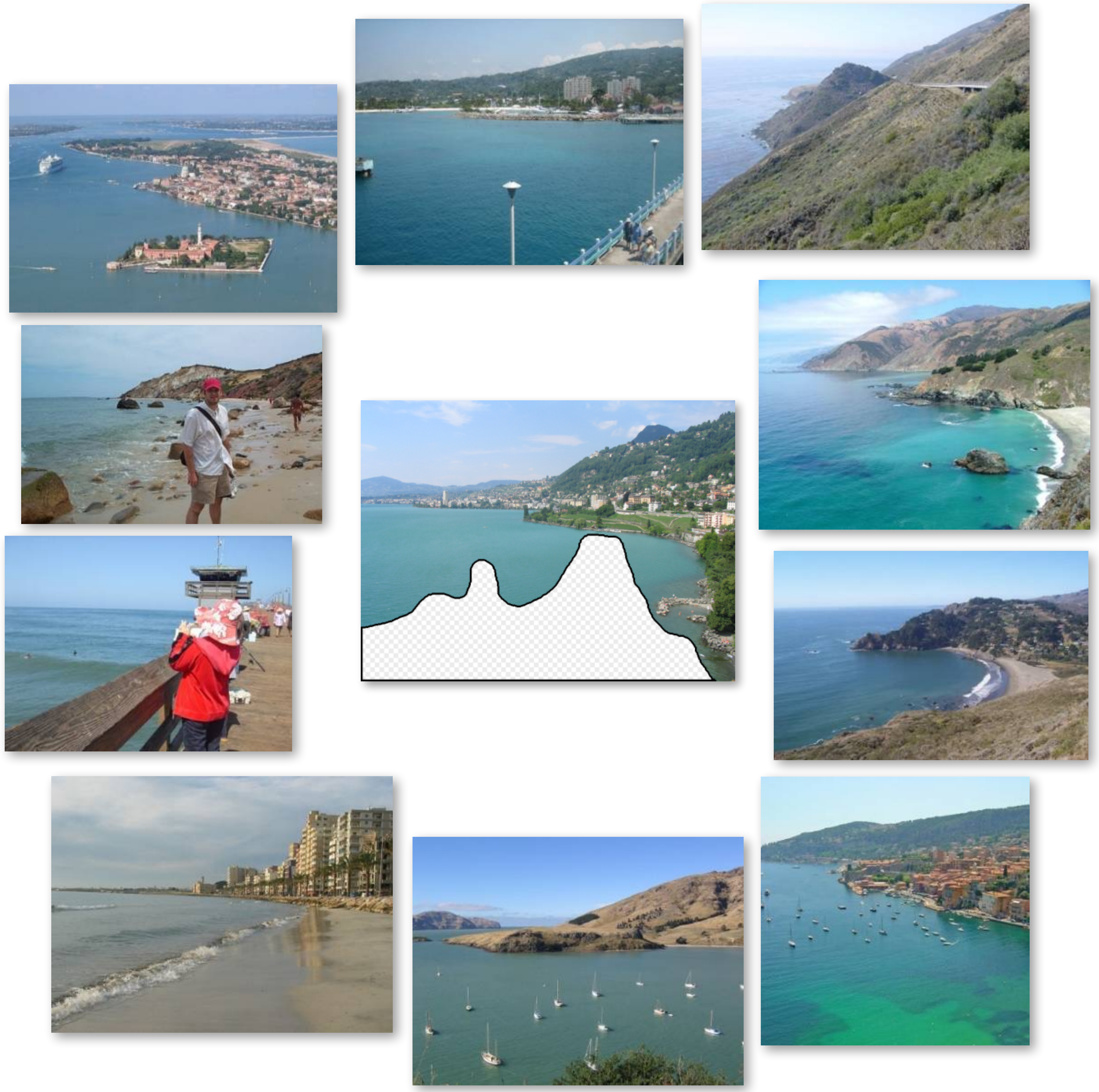





Nearest neighbors from a collection of 20 thousand images

[Slide credit: Alexei Efros]



Nearest neighbors from a collection of 2 million images

[Slide credit: Alexei Efros]

“Unreasonable Effectiveness of Data”

[Halevy, Norvig, Pereira 2009]

Parts of our world can be explained by elegant mathematics

physics, chemistry, astronomy, etc.

But much cannot

psychology, economics, genetics, etc.

Enter The Data!

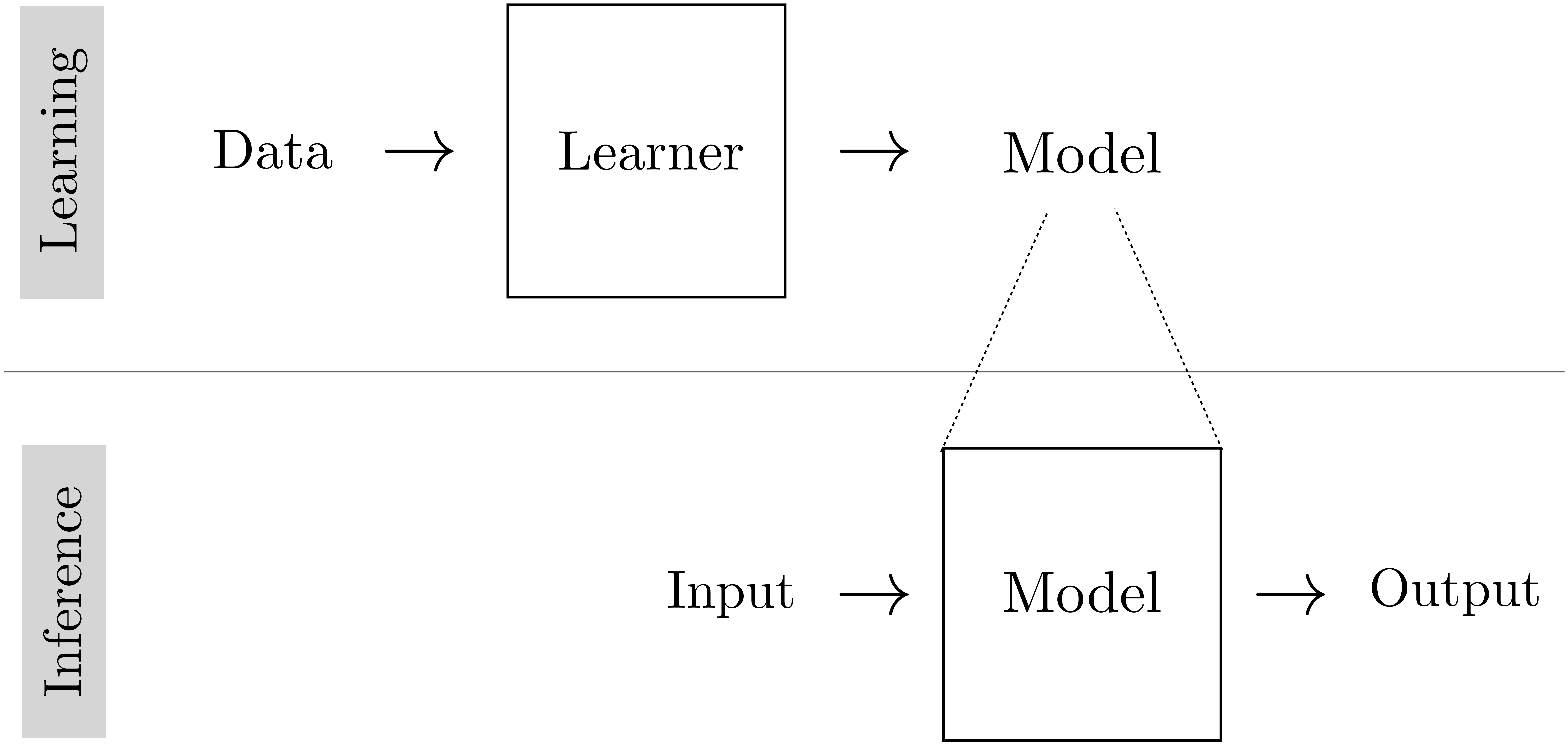
Great advances in several fields:

e.g., speech recognition, machine translation

Case study: Google

> “For many tasks, once we have a billion or so examples, we essentially have a closed set that represents (or at least approximates) what we need…”

[Slide credit: Alexei Efros]


Learning
Data
Learner
Model
Inference
Input
Model
Output

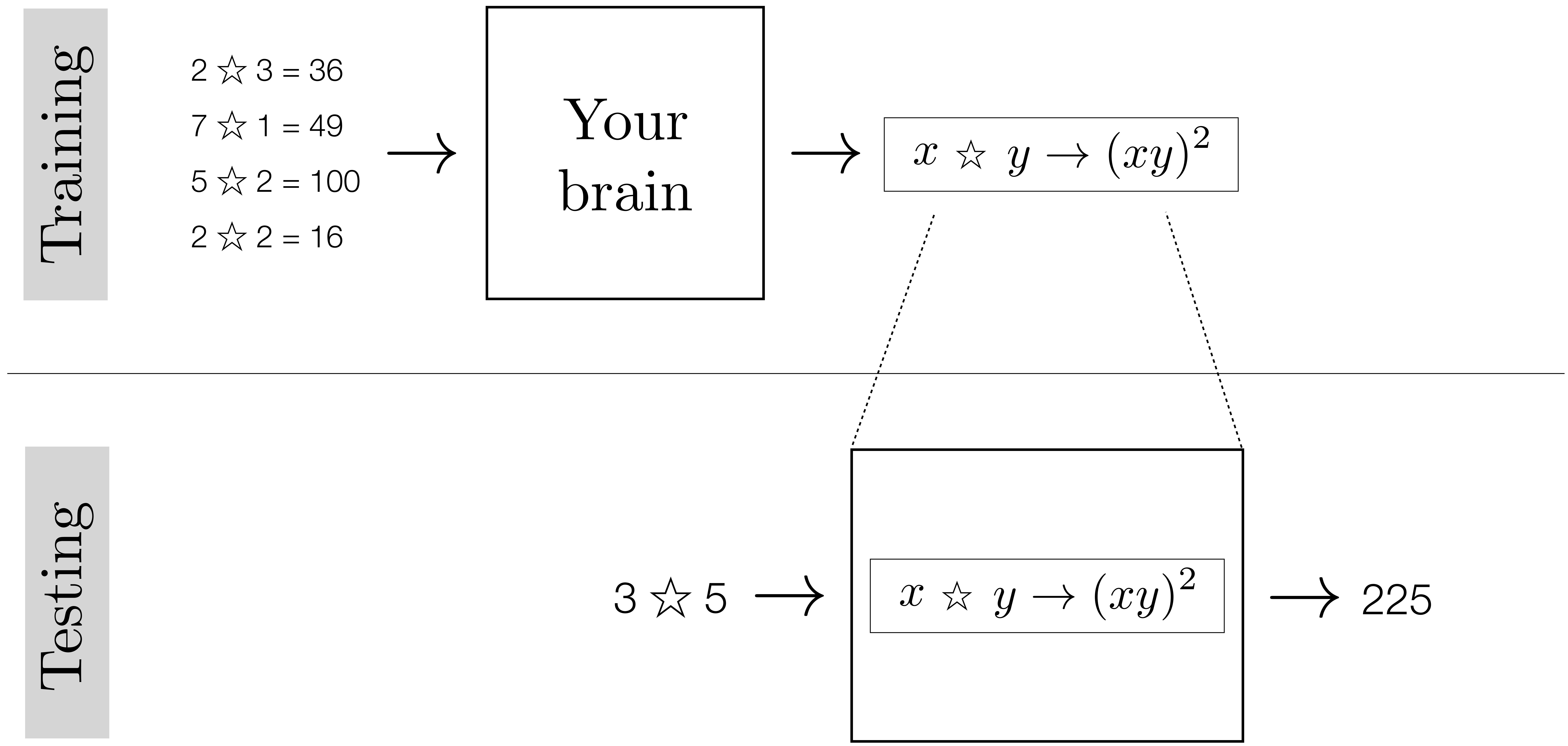
What does ☆ do?

2 ☆ 3 = 36

7 ☆ 1 = 49

5 ☆ 2 = 100

2 ☆ 2 = 16


Training
2 ☆ 3 = 36
7 ☆ 1 = 49
5 ☆ 2 = 100
2 ☆ 2 = 16
Your brain
$x \star y \to (xy)^2$
Testing
3 ☆ 5
$x \star y \to (xy)^2$
225

Learning from examples

(aka **supervised learning**)

Training data

$\{\texttt{input}: [2, 3], \texttt{output}: 36\}$

$\{\texttt{input}: [7, 1], \texttt{output}: 49\}$

$\{\texttt{input}: [5, 2], \texttt{output}: 100\}$

$\{\texttt{input}: [2, 2], \texttt{output}: 16\}$

$\rightarrow$ Learner $\rightarrow$ f

Learning from examples

(aka **supervised learning**)

Training data

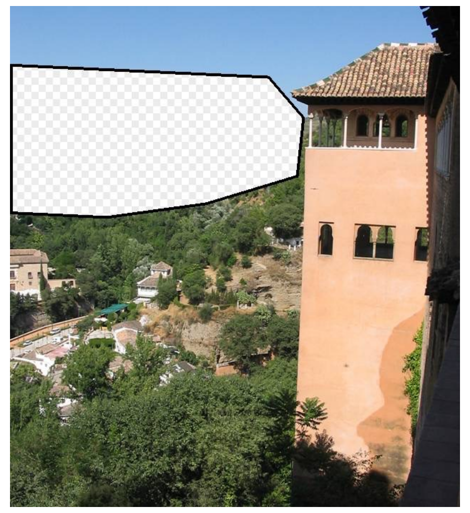
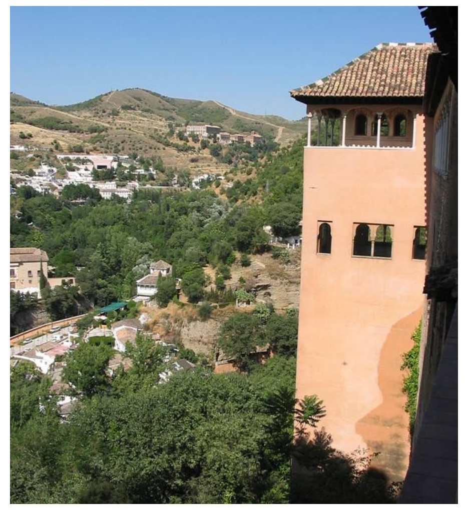
$$\{x^{(1)}, y^{(1)}\}$$
$$\{x^{(2)}, y^{(2)}\} \quad \rightarrow \quad \text{Learner} \quad \rightarrow \quad f : X \rightarrow Y$$
$$\{x^{(3)}, y^{(3)}\}$$
$$\ldots$$

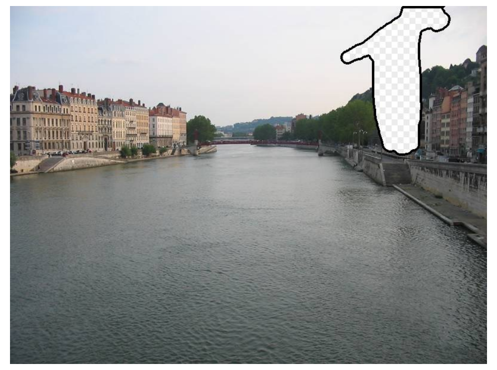
Learning from examples

(aka **supervised learning**)

Training data

X Y

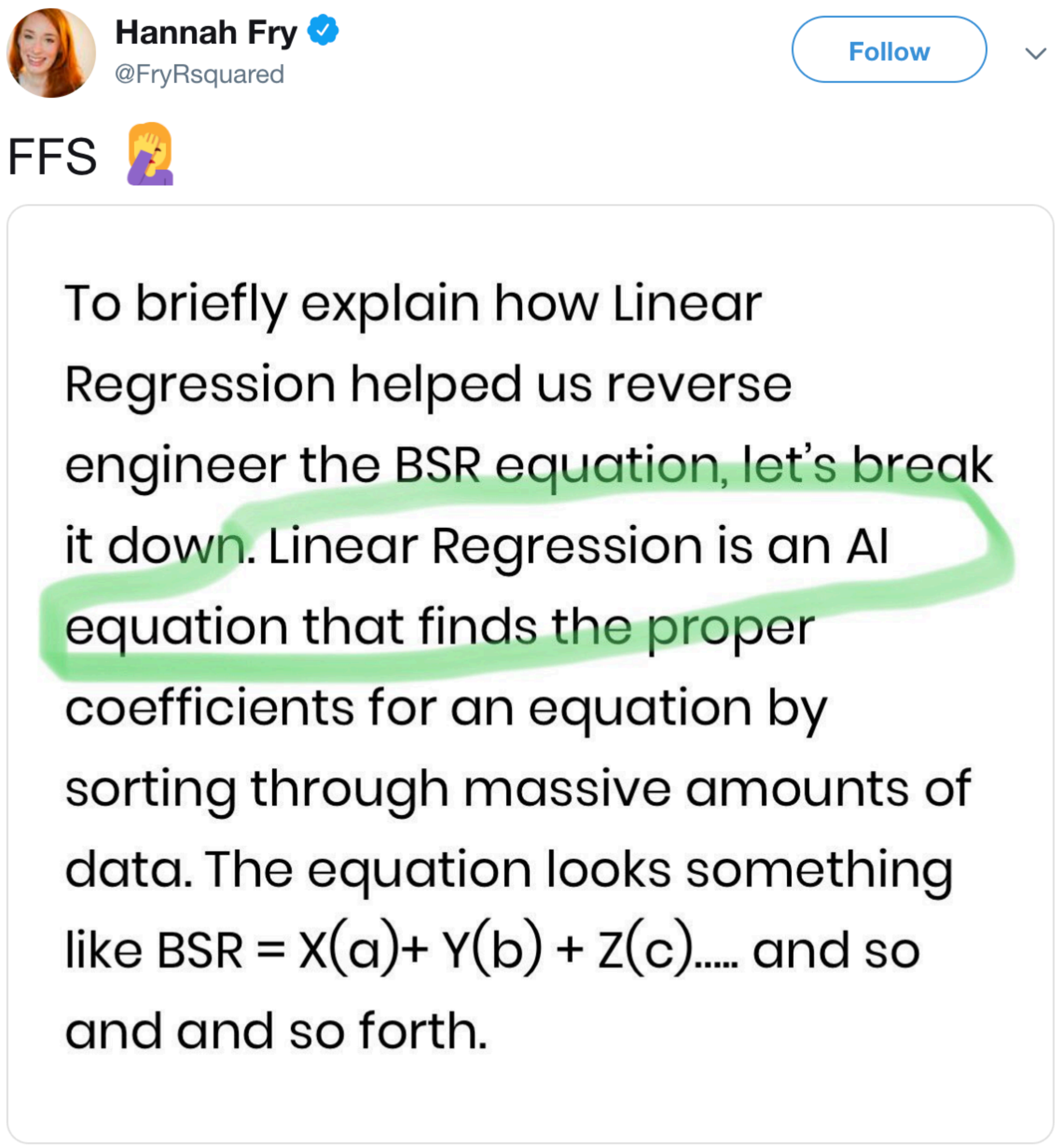
$\{$  ,  $\}$

$\{$  ,  $\}$

$\vdots$

$\rightarrow$ Learner $\rightarrow$ $f : X \rightarrow Y$

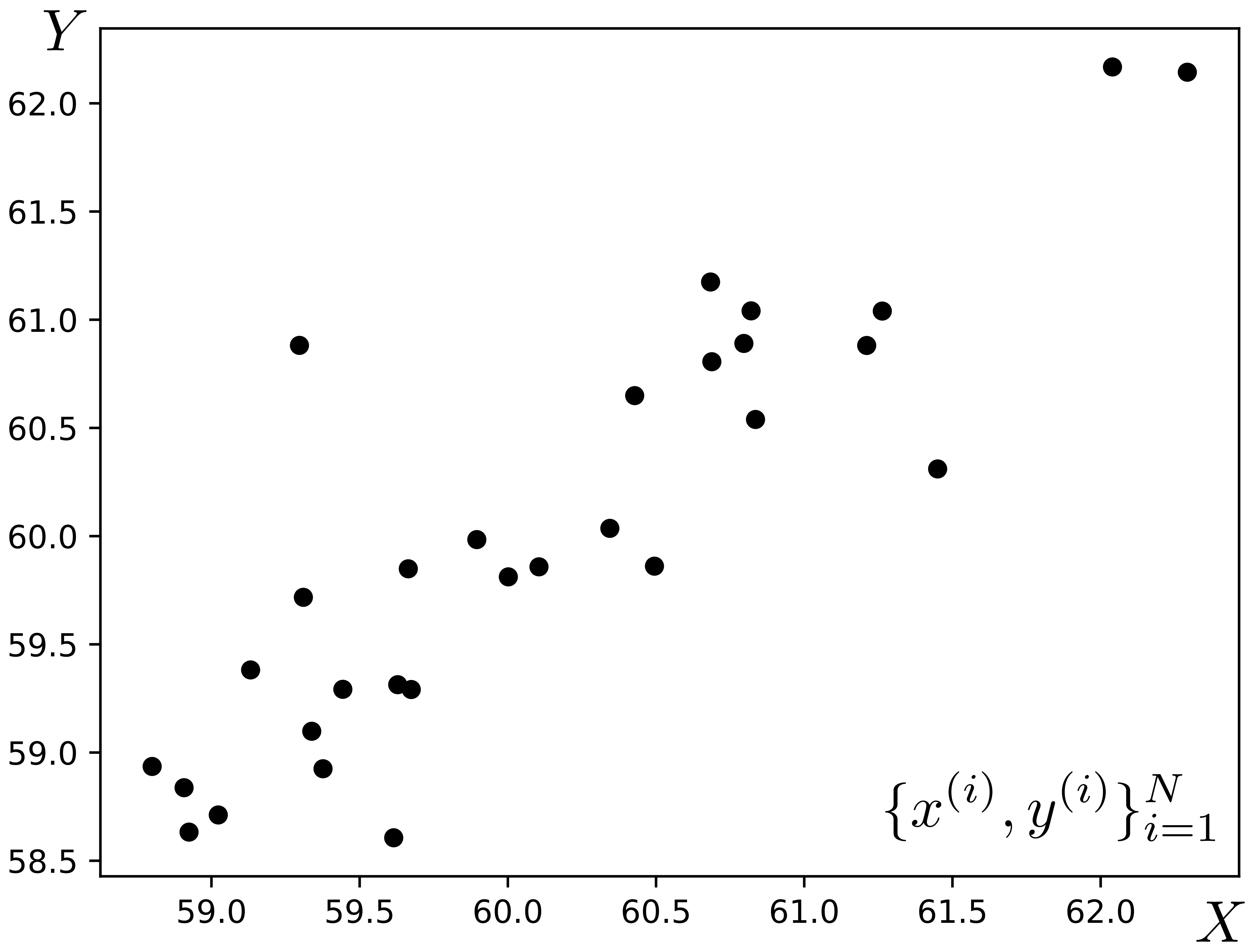
Case study #1: Linear least squares


Hannah Fry
@FryRsquared
Follow

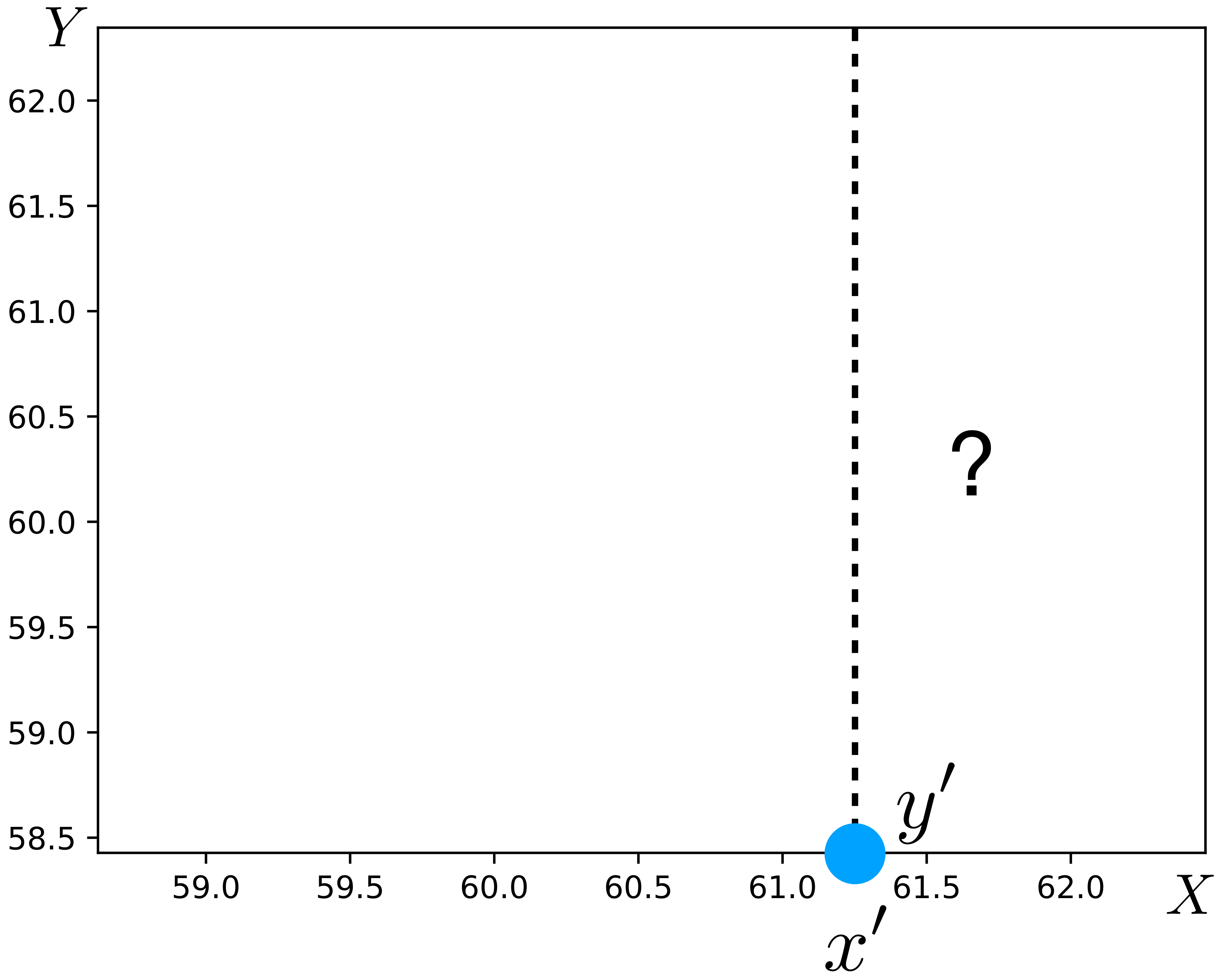
FFS 🤦‍♀️

To briefly explain how Linear Regression helped us reverse engineer the BSR equation, let's break it down. Linear Regression is an AI equation that finds the proper coefficients for an equation by sorting through massive amounts of data. The equation looks something like BSR = X(a)+ Y(b) + Z(c)..... and so and and so forth.

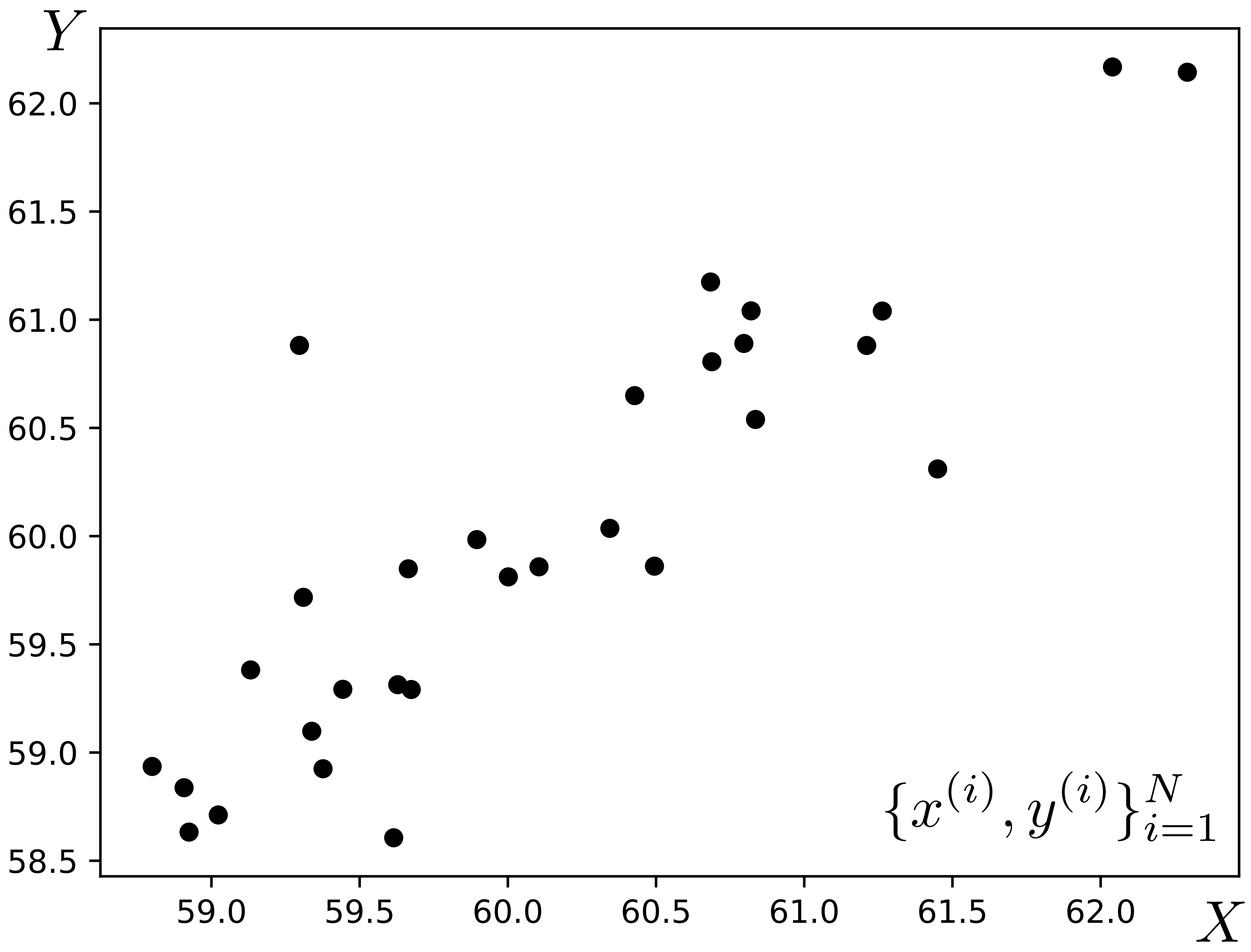
Training data



Test query



Training data

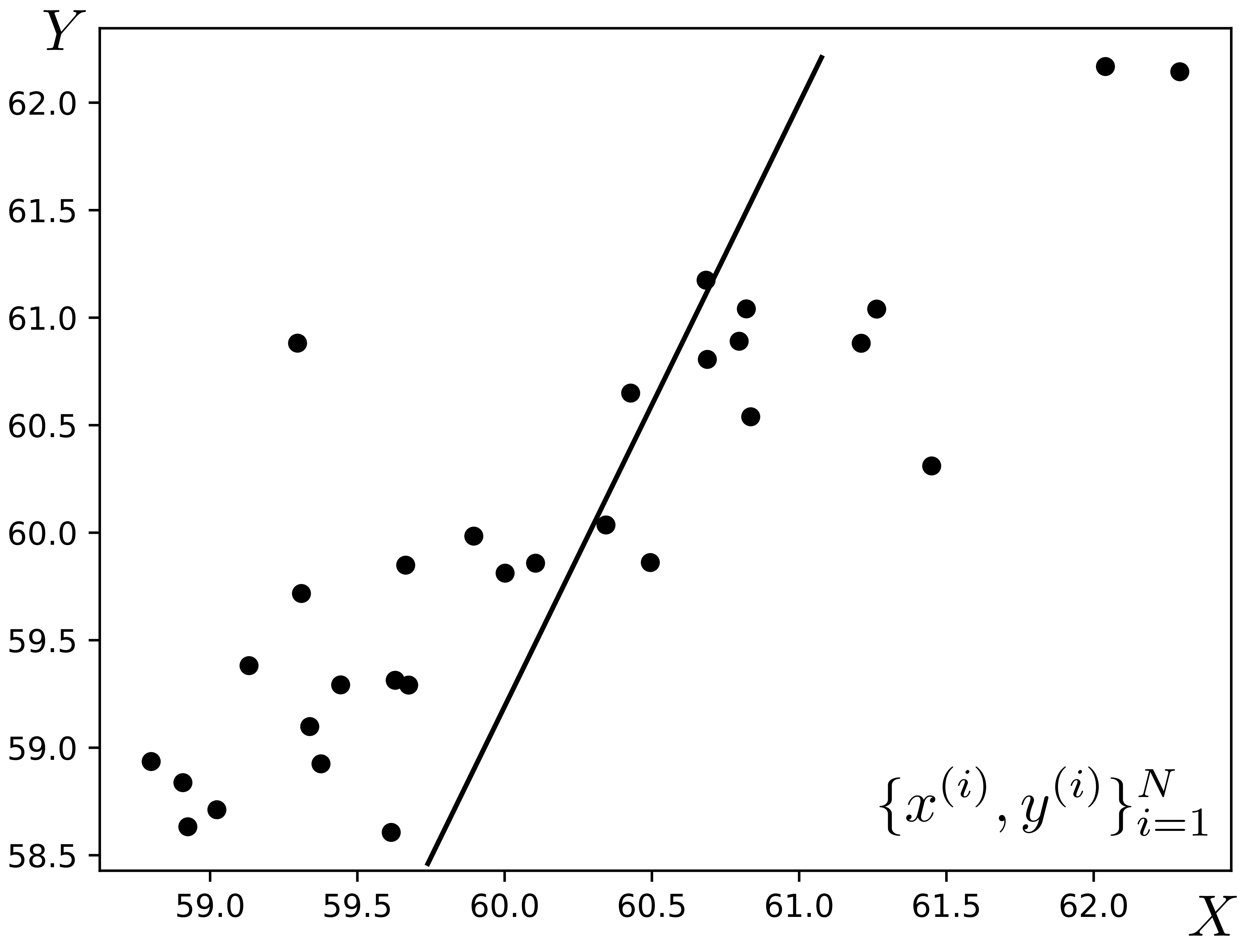


$\rightarrow$ Learner $\rightarrow$ $f_\theta(x) = \theta_1 x + \theta_0$

Hypothesis space

The relationship between X and Y is roughly linear: $y \approx \theta_1 x + \theta_0$

Training data

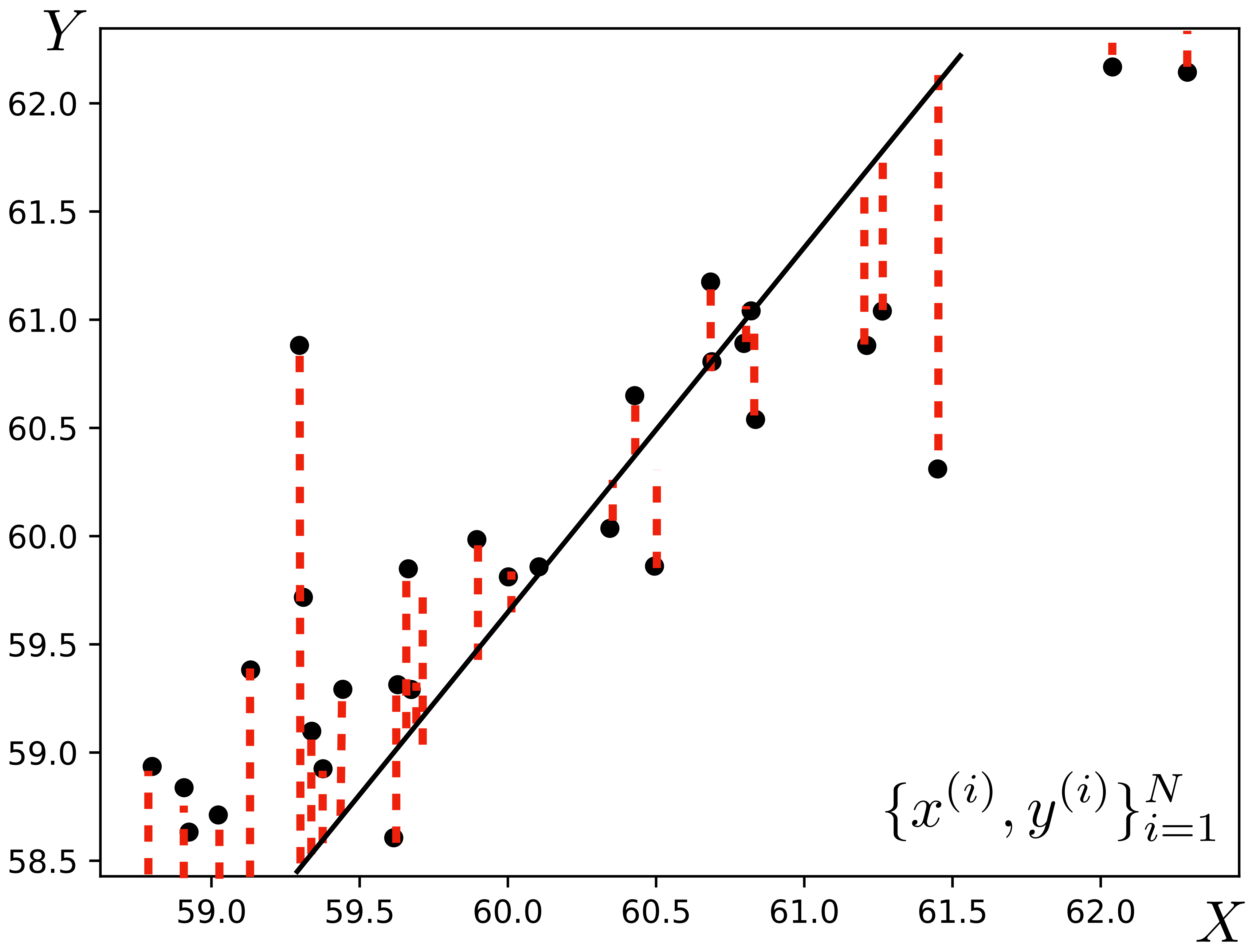


Search for the **parameters**, $\theta = \{\theta_0, \theta_1\}$, that best fit the data.

$$f_\theta(x) = \theta_1 x + \theta_0$$

Best fit in what sense?

Training data



Search for the **parameters**, $\theta = \{\theta_0, \theta_1\}$, that best fit the data.

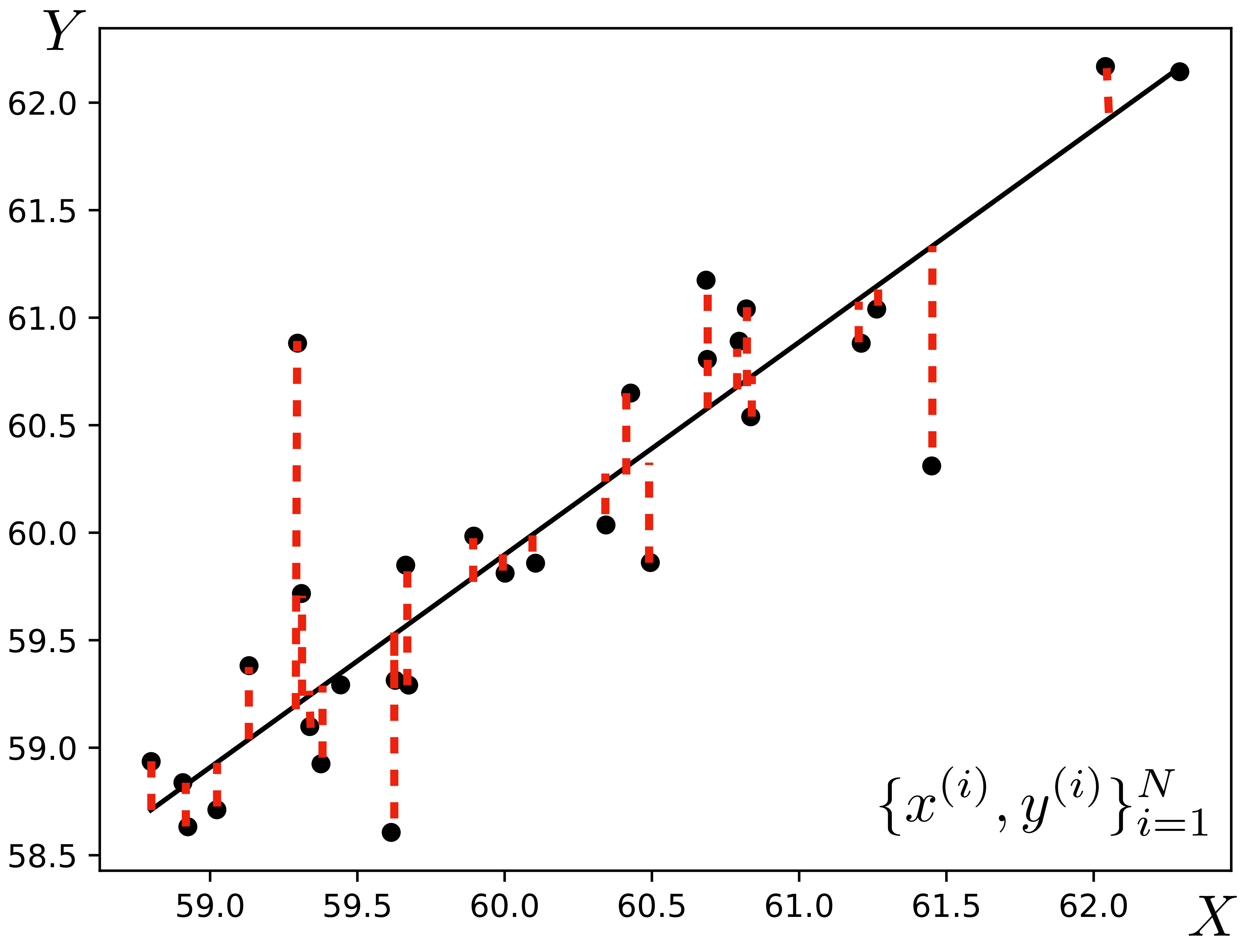
$$f_\theta(x) = \theta_1 x + \theta_0$$

Best fit in what sense?

The least-squares **objective** (aka **loss**) says the best fit is the function that minimizes the squared error between predictions and target values:

$$\mathcal{L}(\hat{y}, y) = (\hat{y} - y)^2 \quad \hat{y} \equiv f_\theta(x)$$

Training data



Search for the **parameters**, $\theta = \{\theta_0, \theta_1\}$, that best fit the data.

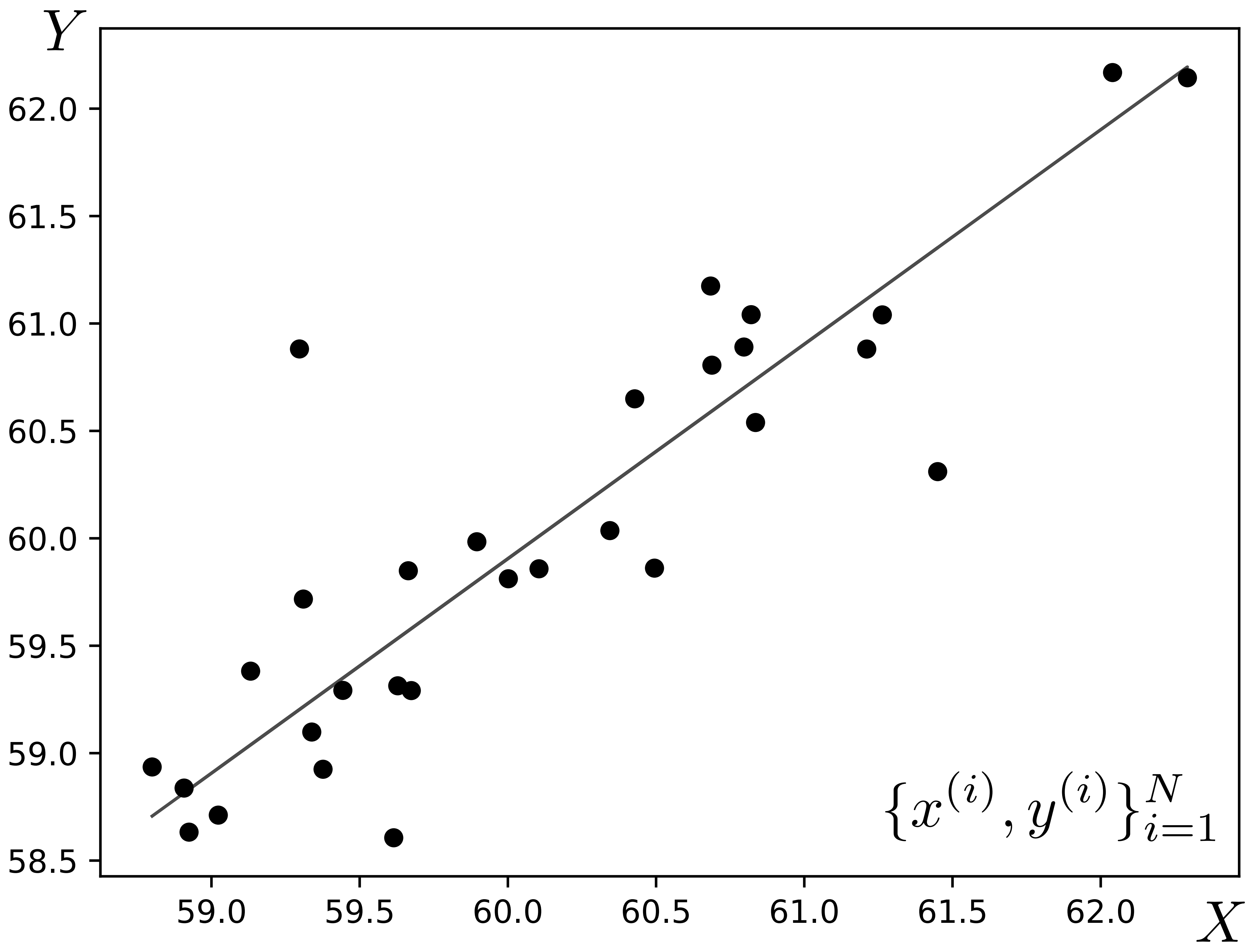
$$f_\theta(x) = \theta_1 x + \theta_0$$

Best fit in what sense?

The least-squares **objective** (aka **loss**) says the best fit is the function that minimizes the squared error between predictions and target values:

$$\mathcal{L}(\hat{y}, y) = (\hat{y} - y)^2 \qquad \hat{y} \equiv f_\theta(x)$$

Training data

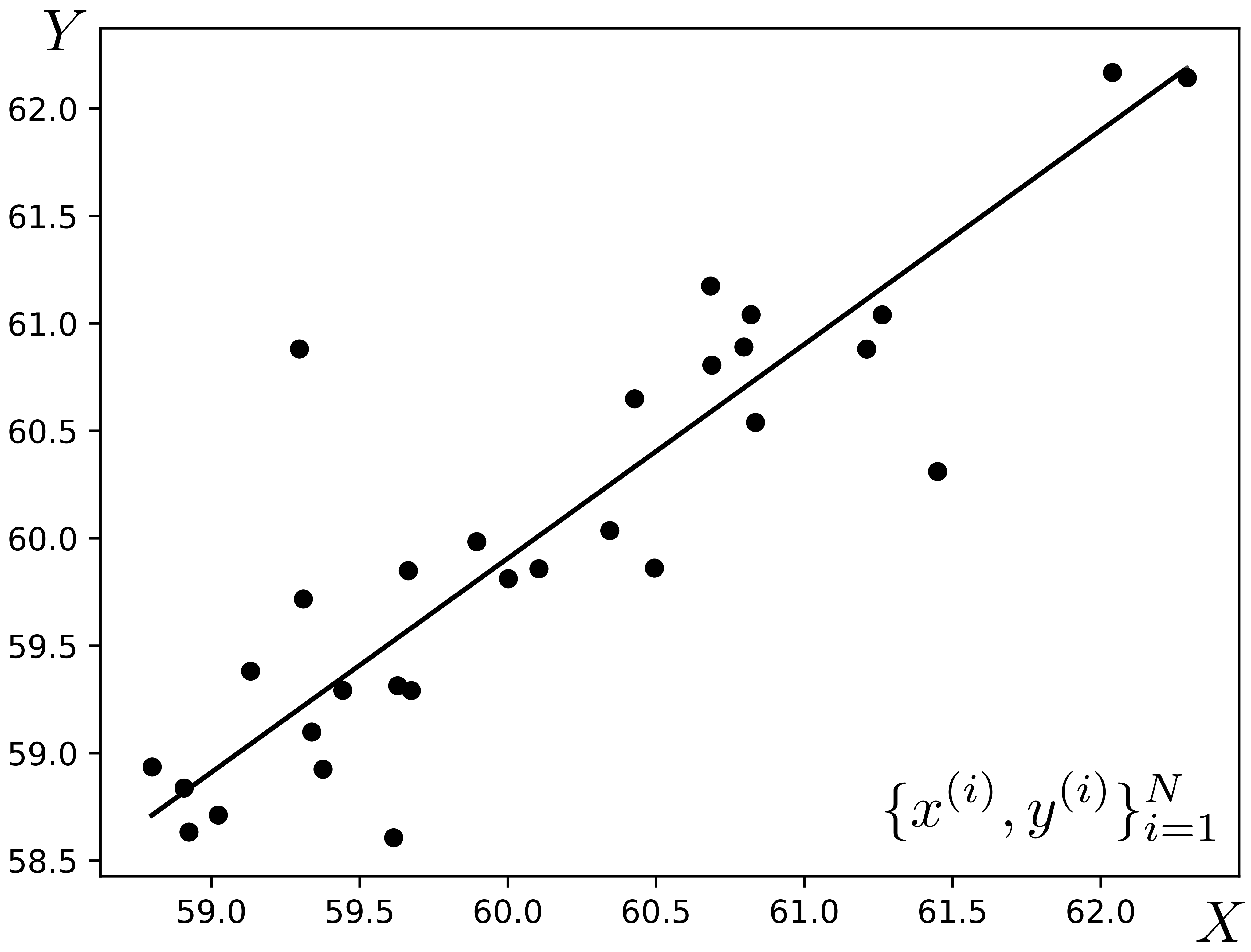


Complete learning problem:

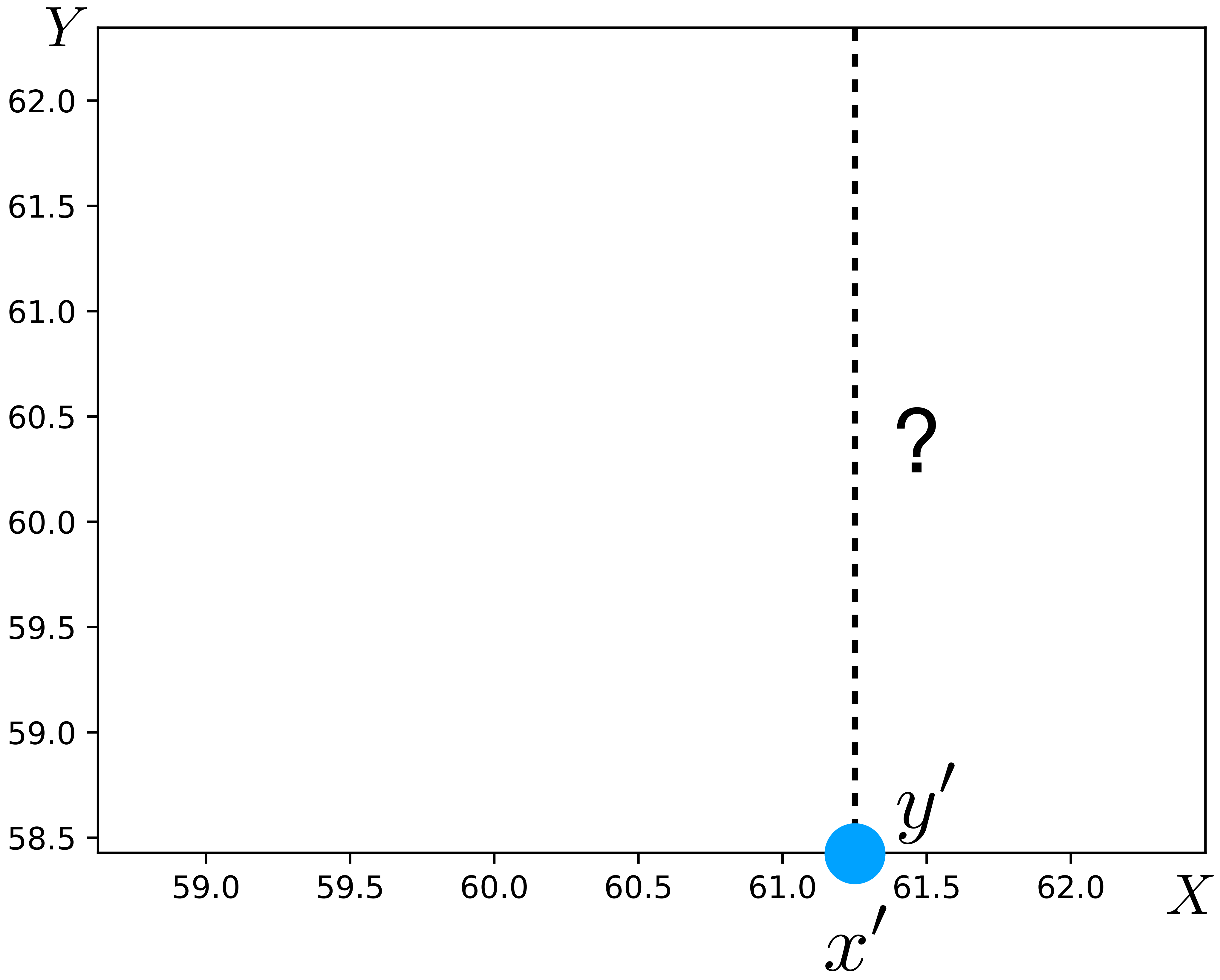
$$\theta^* = \arg\min_\theta \sum_{i=1}^N (f_\theta(x^{(i)}) - y^{(i)})^2$$

$$= \arg\min_\theta \sum_{i=1}^N (\theta_1 x^{(i)} + \theta_0 - y^{(i)})^2$$

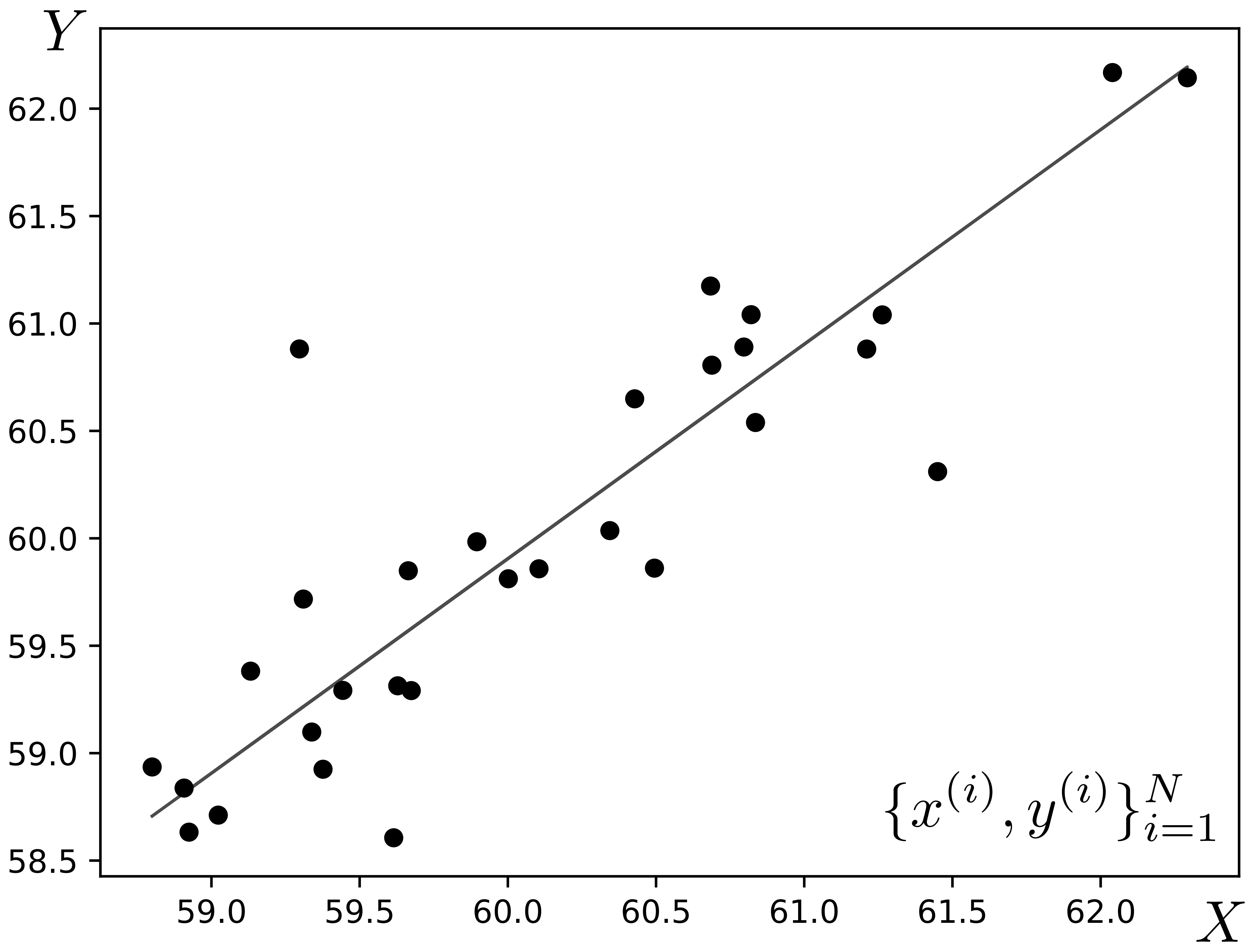
Training data



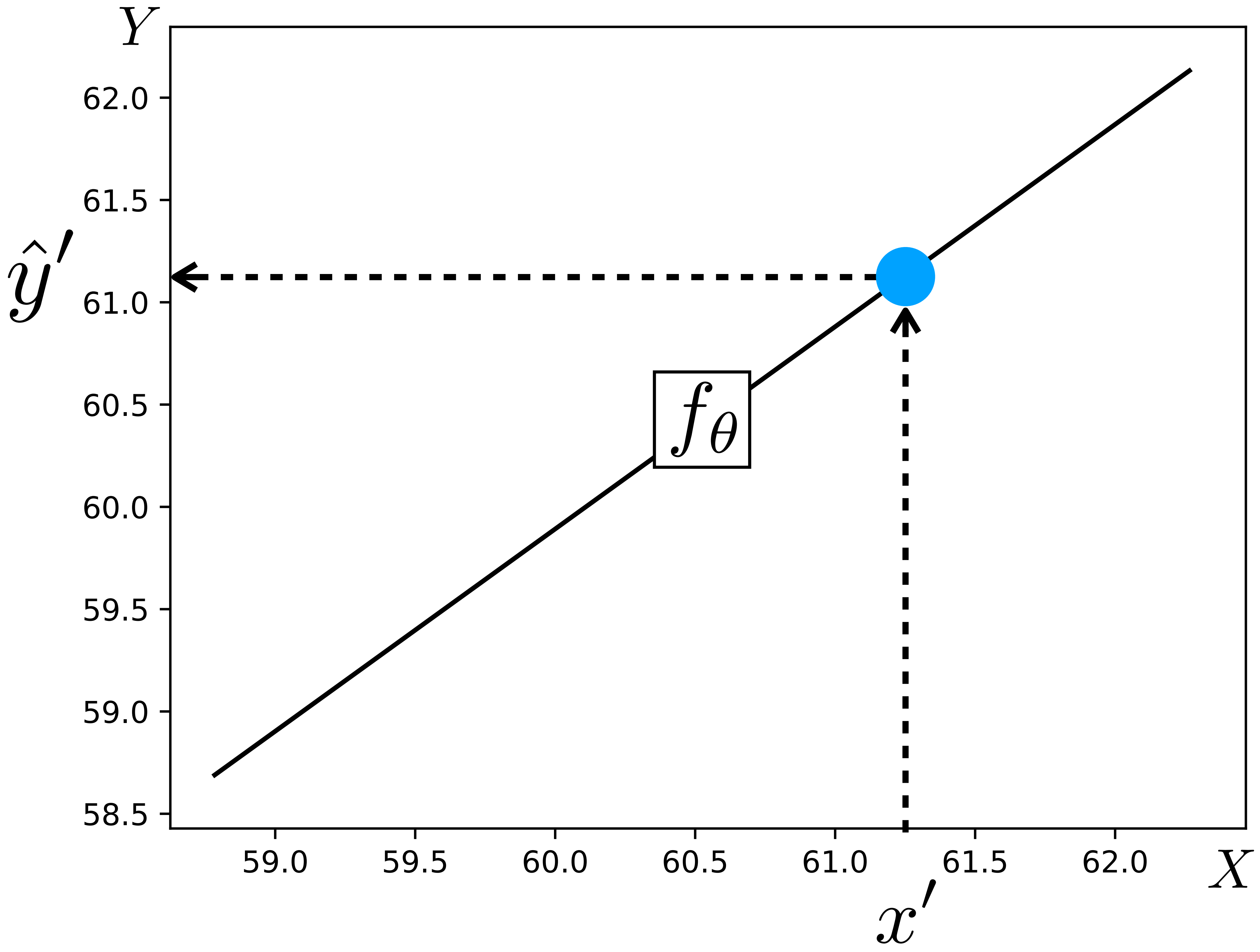
Test query



Training data



Test query

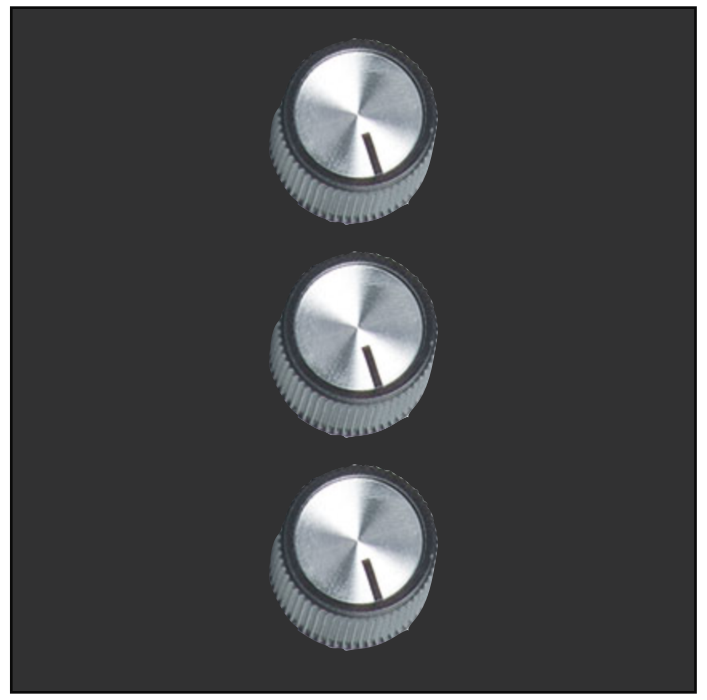


$$x' \dashrightarrow \boxed{f_\theta} \dashrightarrow \hat{y}'$$

How to minimize the objective w.r.t. θ?

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{N} (f_\theta(x^{(i)}) - y^{(i)})^2$$

Use an **optimizer**!

Input →  → **Output** → **Score**

Machine with knobs

How to minimize the objective w.r.t. θ?

In the linear case:

Learning problem

$$\theta^* = \arg\min_\theta \sum_{i=1}^{N} (\theta_1 x^{(i)} + \theta_0 - y^{(i)})^2$$

$$J(\theta) = \sum_{i=1}^{N} (\theta_1 x^{(i)} + \theta_0 - y^{(i)})^2$$
$$= (\mathbf{y} - \mathbf{X}\theta)^T (\mathbf{y} - \mathbf{X}\theta)$$

$$\mathbf{X} = \begin{pmatrix} x^{(1)} & 1 \\ x^{(2)} & 1 \\ \vdots & \vdots \\ x^{(N)} & 1 \end{pmatrix} \quad \theta = \begin{pmatrix} \theta_1 & \theta_0 \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{pmatrix}$$

$$\theta^* = \arg\min_\theta J(\theta)$$

$$\frac{\partial J(\theta)}{\partial \theta} = 0$$

$$\frac{\partial J(\theta)}{\partial \theta} = 2(\mathbf{X}^T\mathbf{X}\theta - \mathbf{X}^T\mathbf{y})$$

$$2(\mathbf{X}^T\mathbf{X}\theta^* - \mathbf{X}^T\mathbf{y}) = 0$$

$$\mathbf{X}^T\mathbf{X}\theta^* = \mathbf{X}^T\mathbf{y}$$

$$\boxed{\theta^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}}$$

Solution

Empirical Risk Minimization

(formalization of supervised learning)

Linear least squares learning problem

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{N} (\theta_1 x^{(i)} + \theta_0 - y^{(i)})^2$$

Empirical Risk Minimization

(formalization of supervised learning)

Objective function
(loss)

$$f^* = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{N} \mathcal{L}(f(\mathbf{x}^{(i)}), \mathbf{y}^{(i)})$$

Hypothesis space

Training data

Case study #1: Linear least squares

Data

$\{x^{(i)}, y^{(i)}\}_{i=1}^{N} \rightarrow$

Learner

Objective

$$\mathcal{L}(f_\theta(x), y) = (f_\theta(x) - y)^2$$

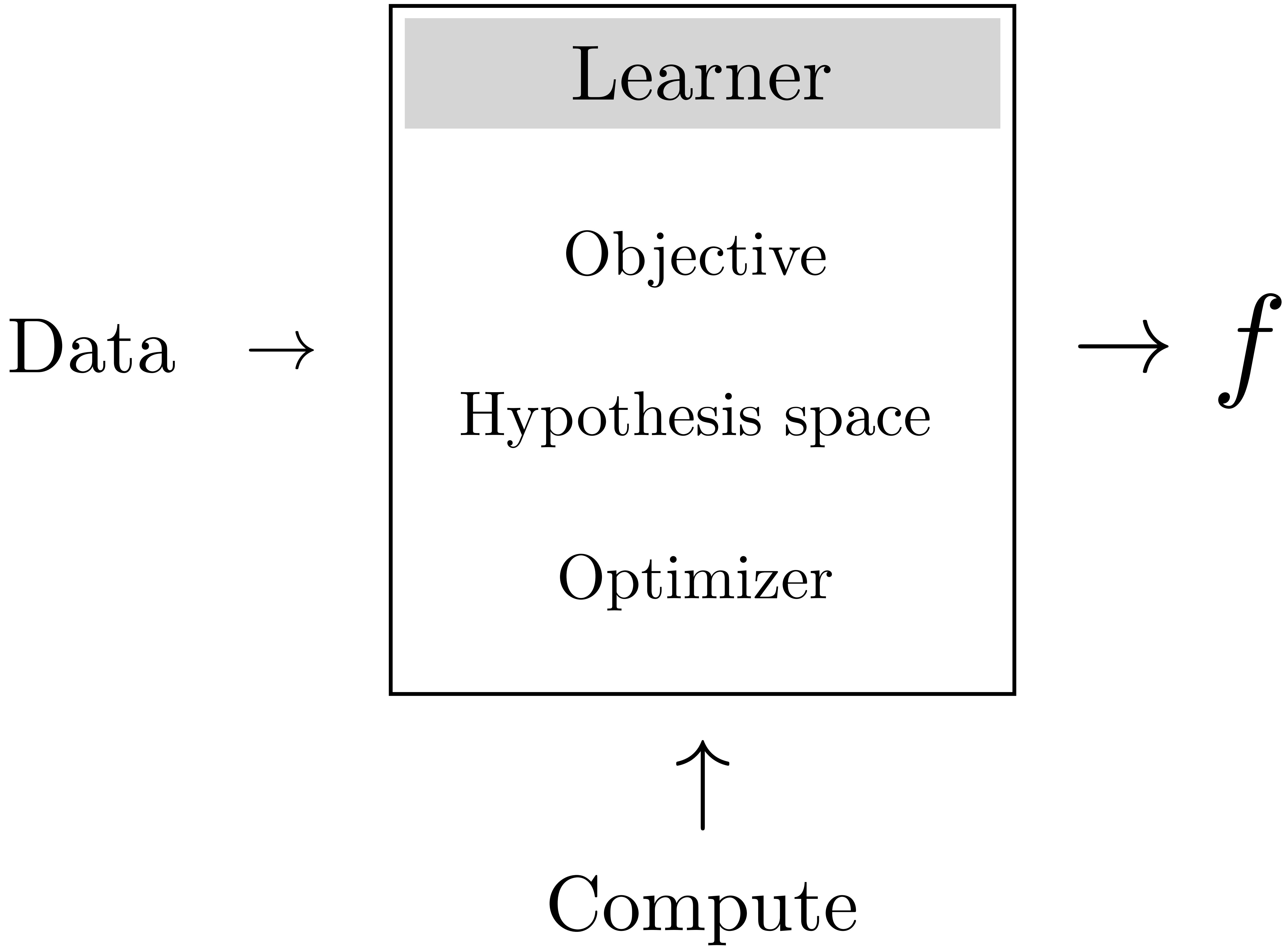
Hypothesis space

$$f_\theta(x) = \theta_1 x + \theta_0$$

Optimizer

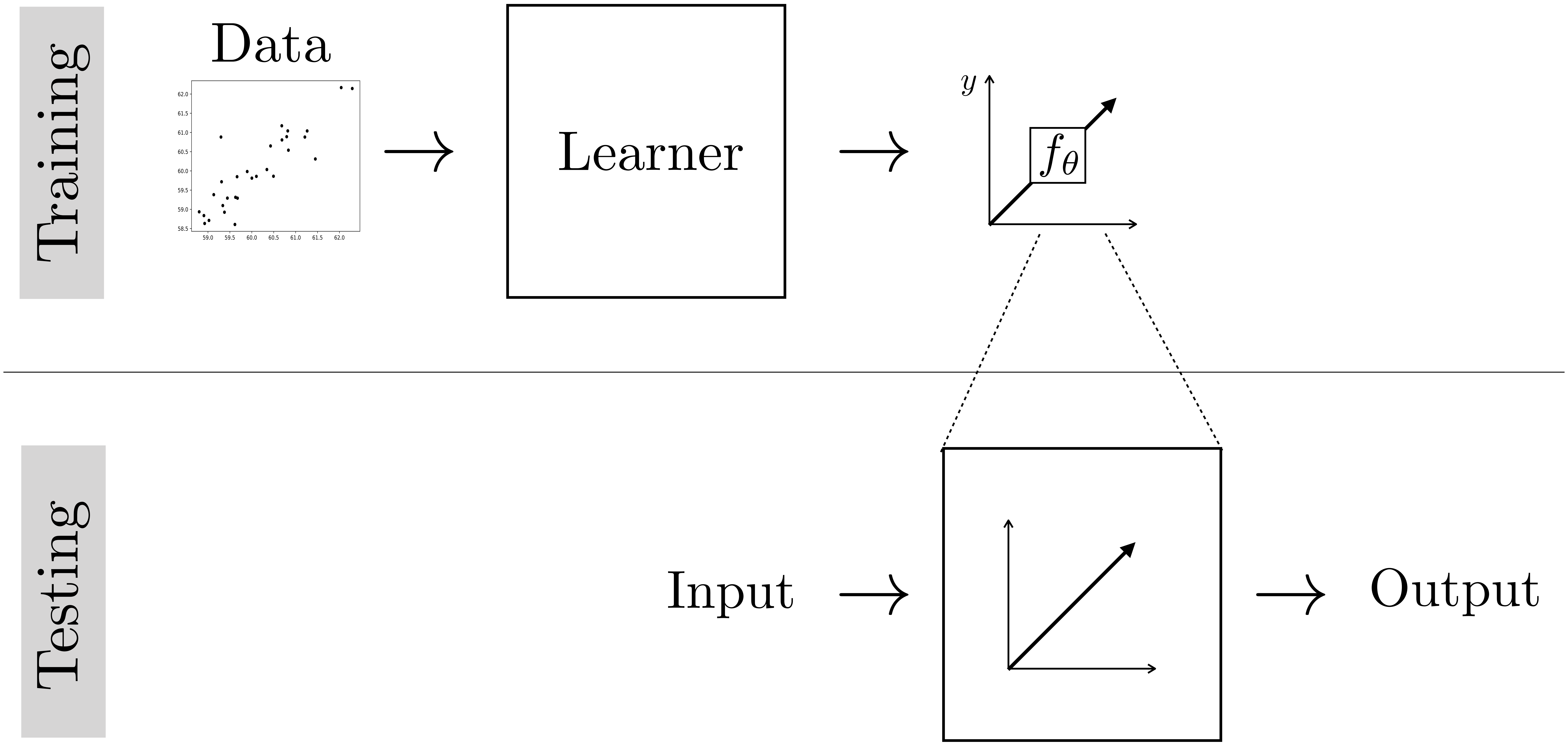
$$\theta^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

$\rightarrow f$

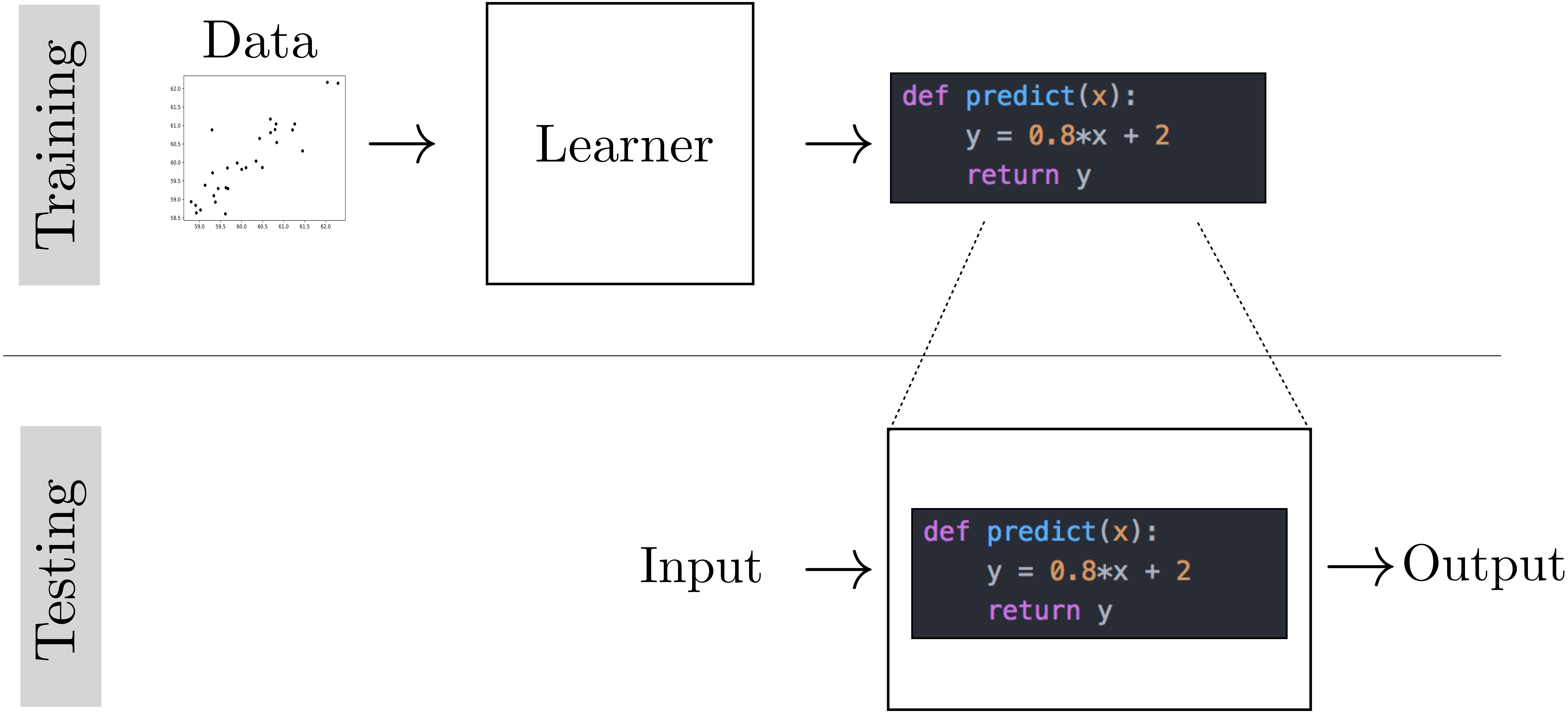


Learner
Objective
Hypothesis space
Optimizer
Data →
→ f
↑
Compute

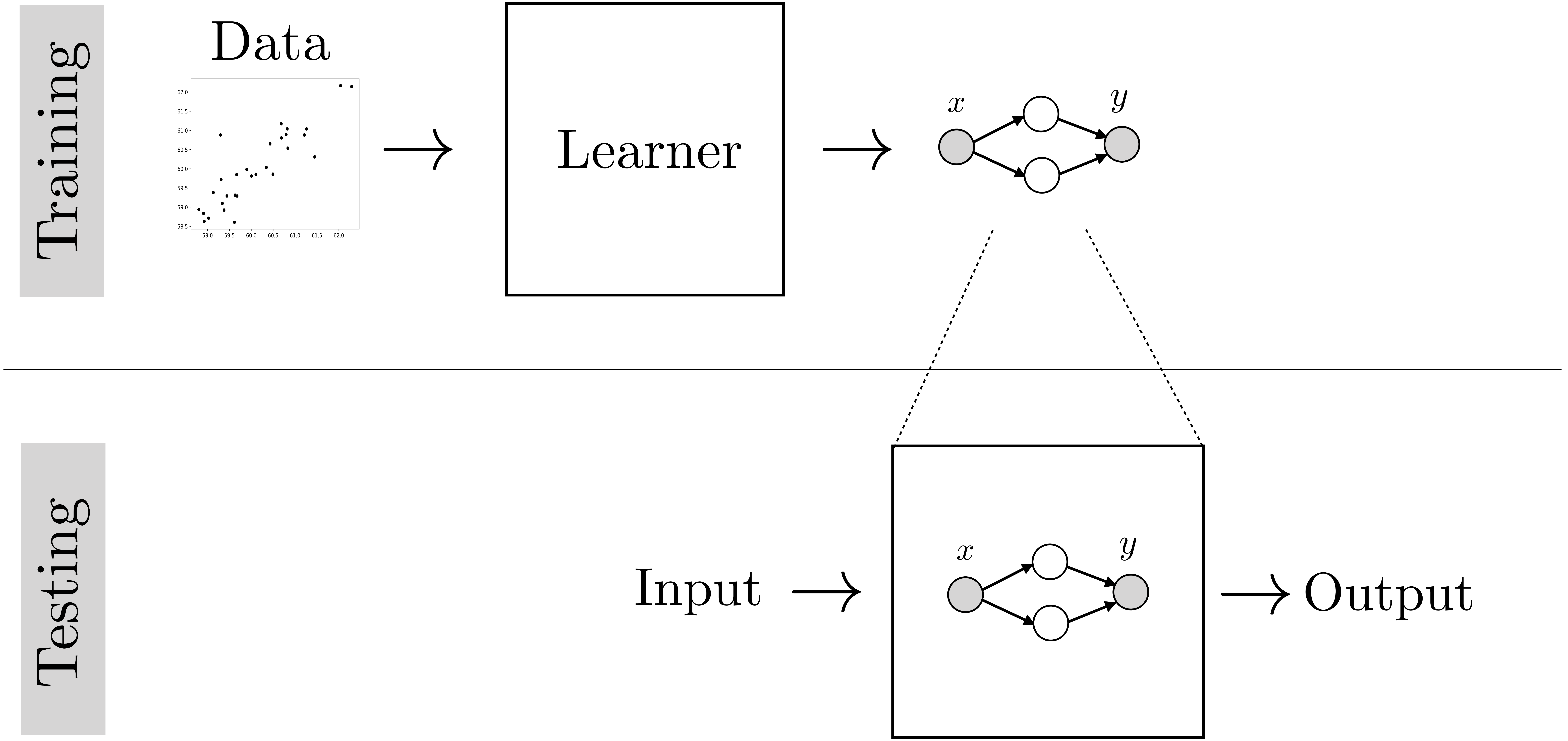
Example 1: Linear least squares

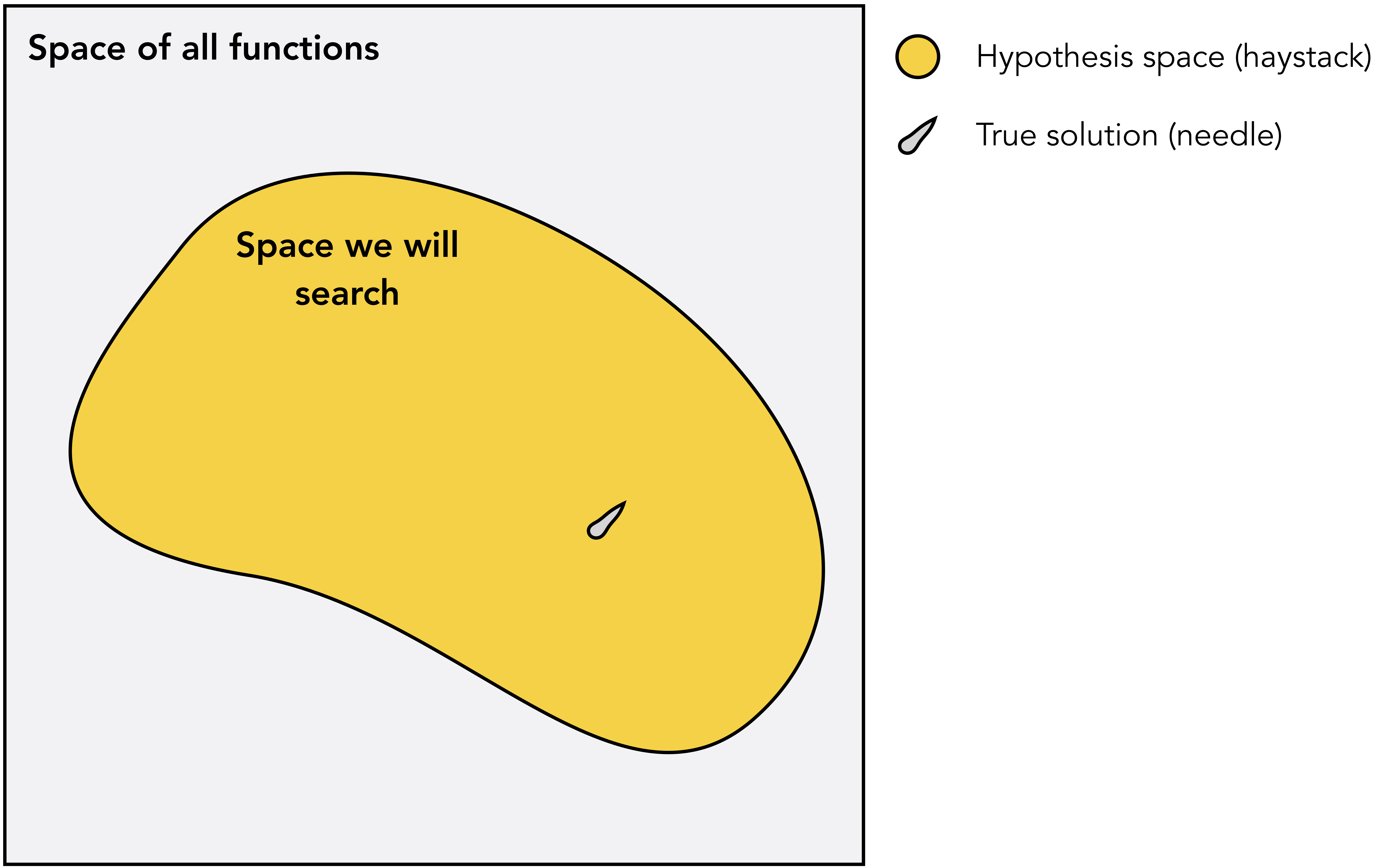


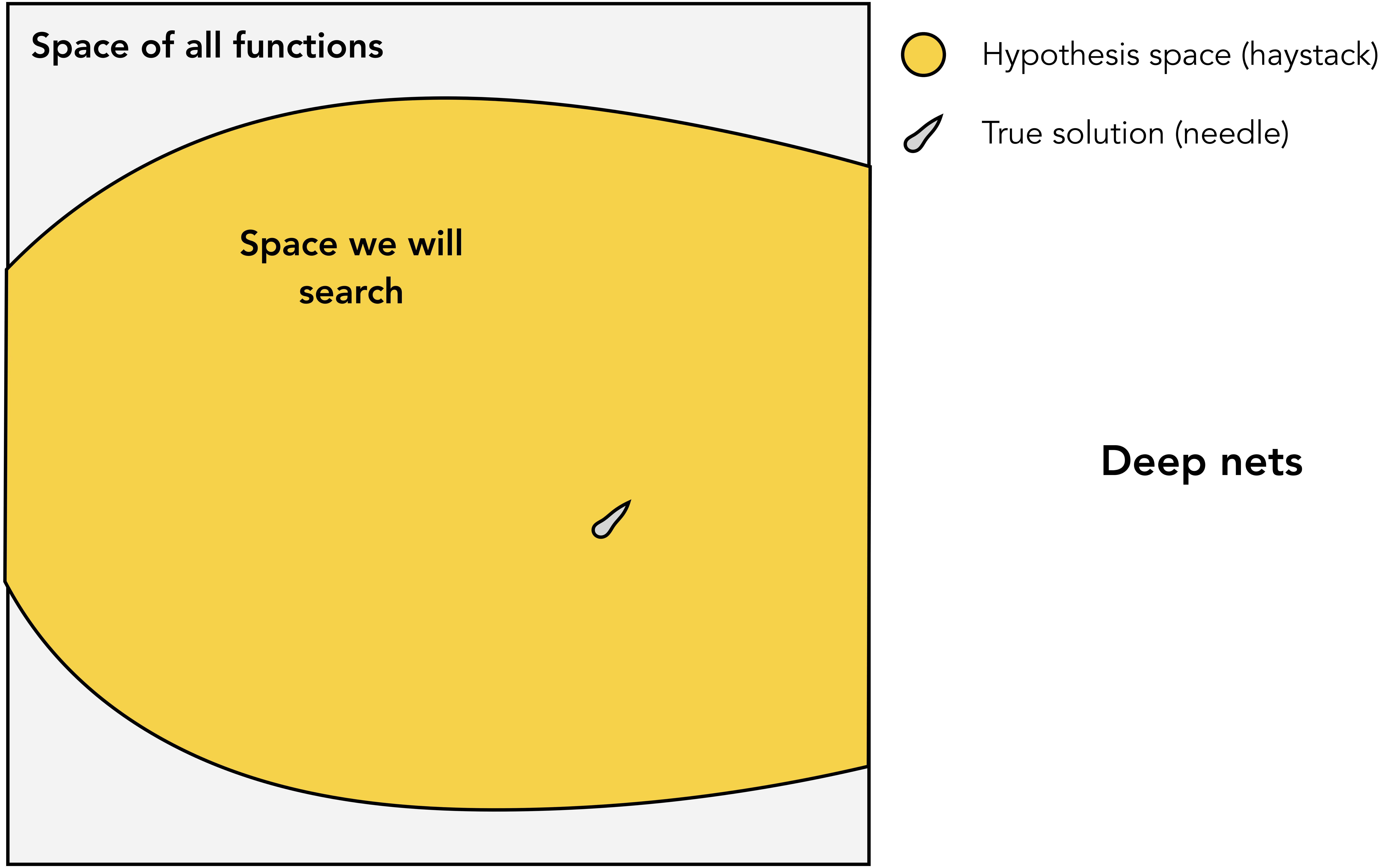
Example 2: Program induction



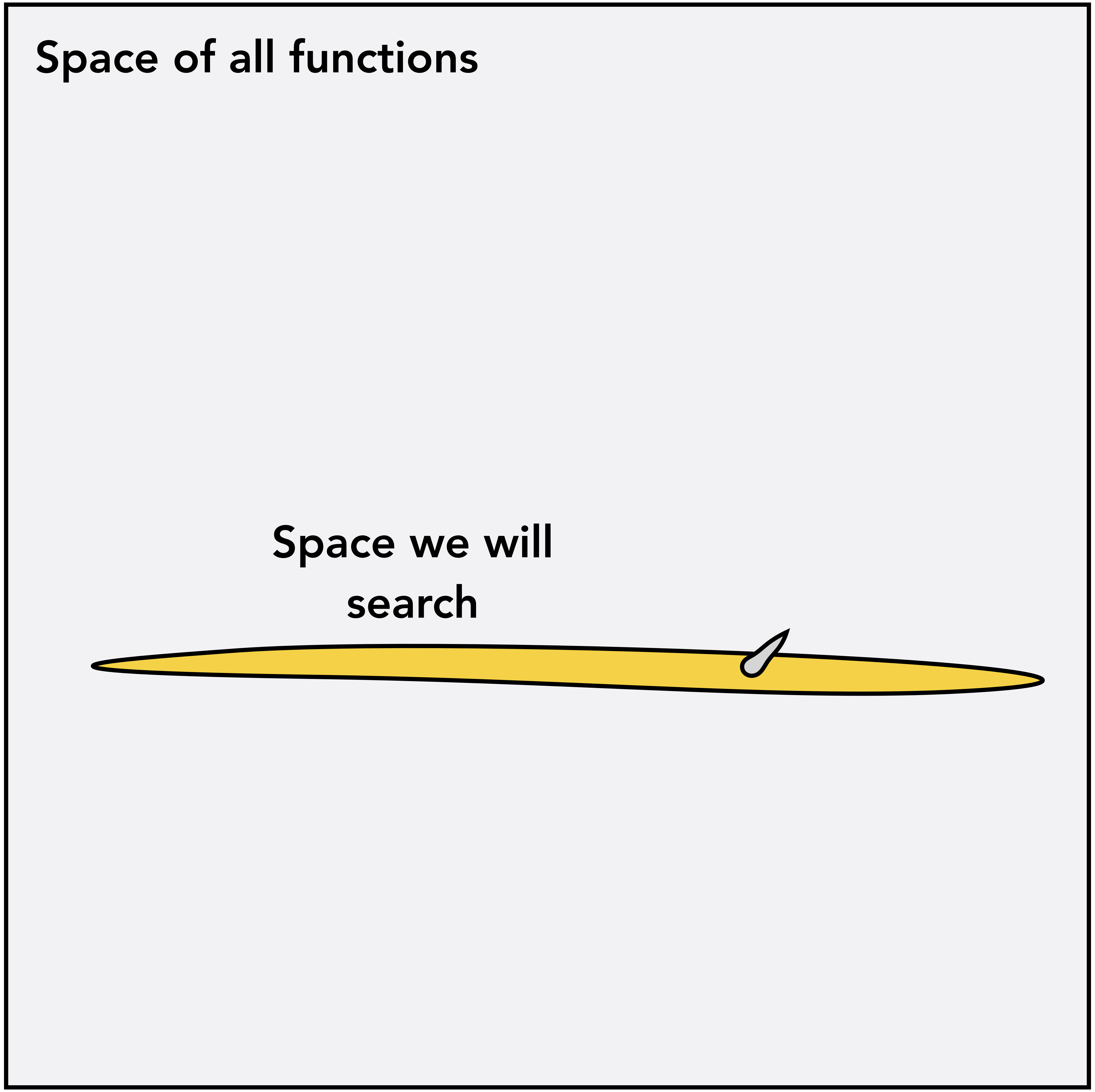
Example 3: Deep learning




Space of all functions
Space we will search
Hypothesis space (haystack)
True solution (needle)


Space of all functions
Space we will search
Hypothesis space (haystack)
True solution (needle)

Deep nets

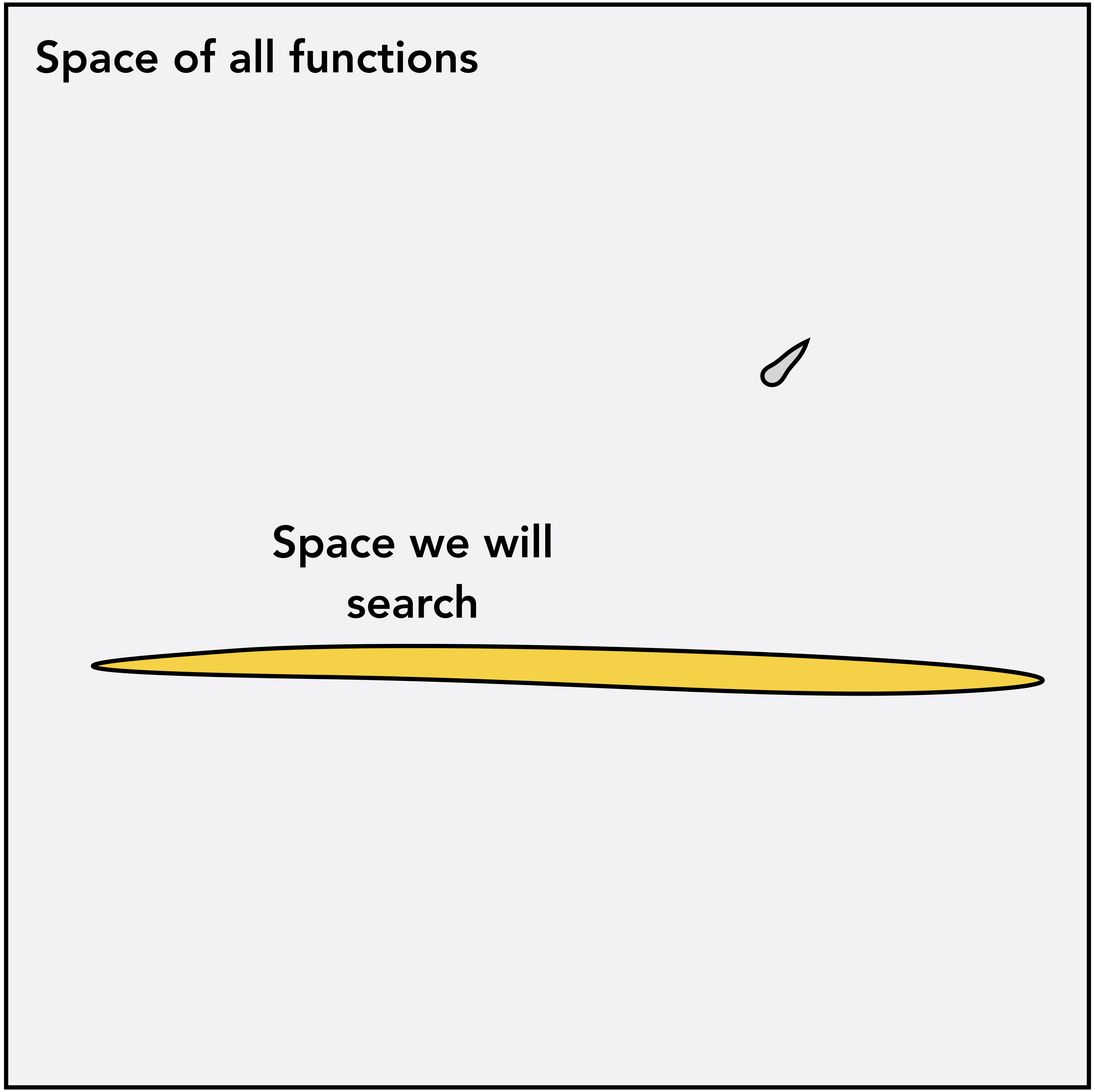


Hypothesis space (haystack)

True solution (needle)

Linear functions

True solution is linear



Hypothesis space (haystack)

True solution (needle)

Linear functions

True solution is nonlinear

Learning for vision

Big questions:

1. How do you represent the input and output?

2. What is the objective?

3. What is the hypothesis space? (e.g., linear, polynomial, neural net?)

4. How do you optimize? (e.g., gradient descent, Newton's method?)

5. What data do you train on?

Case study #2: Image classification

1. **How do you represent the input and output?**
2. **What is the objective?**
3. Assume hypothesis space is sufficienly expressive
4. Assume we optimize perfectly
5. Assume we train on exactly the data we care about

Image classification

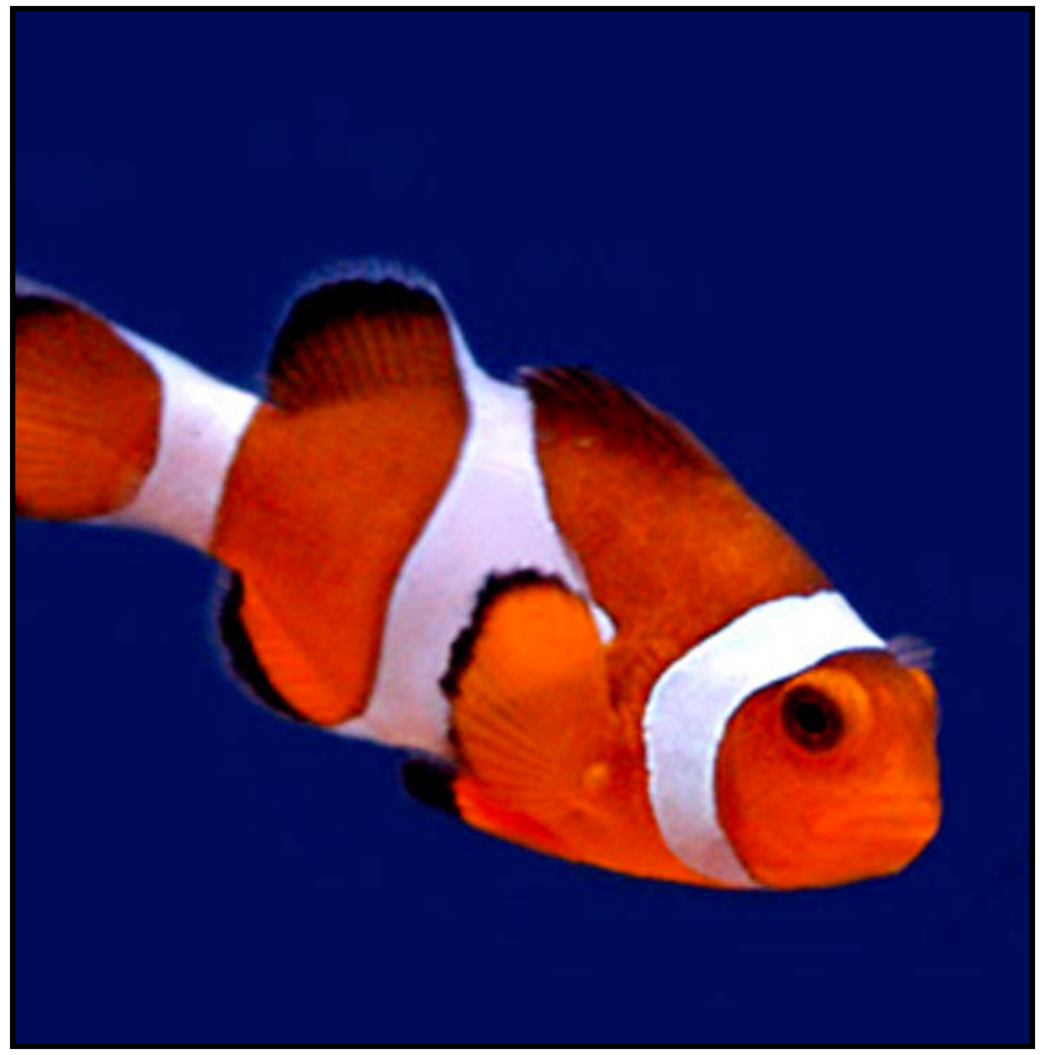


image **x**

Classifier

“Fish”

label y

Image classification

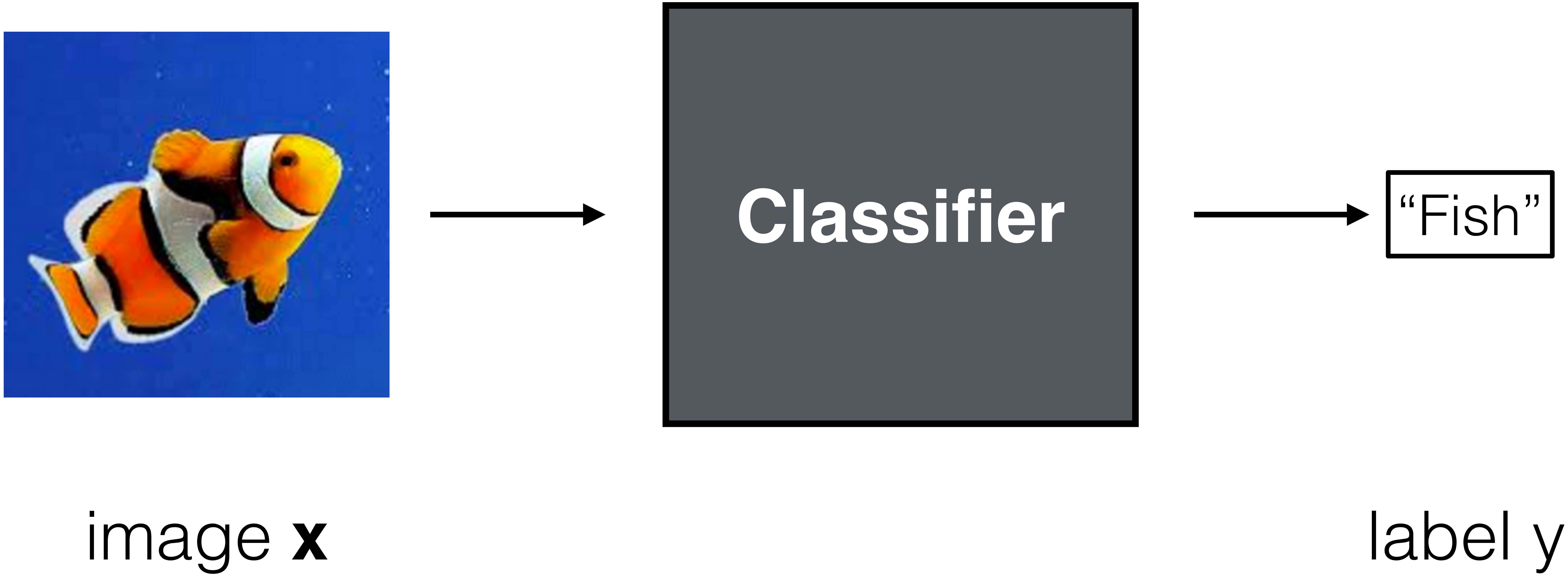


Image classification

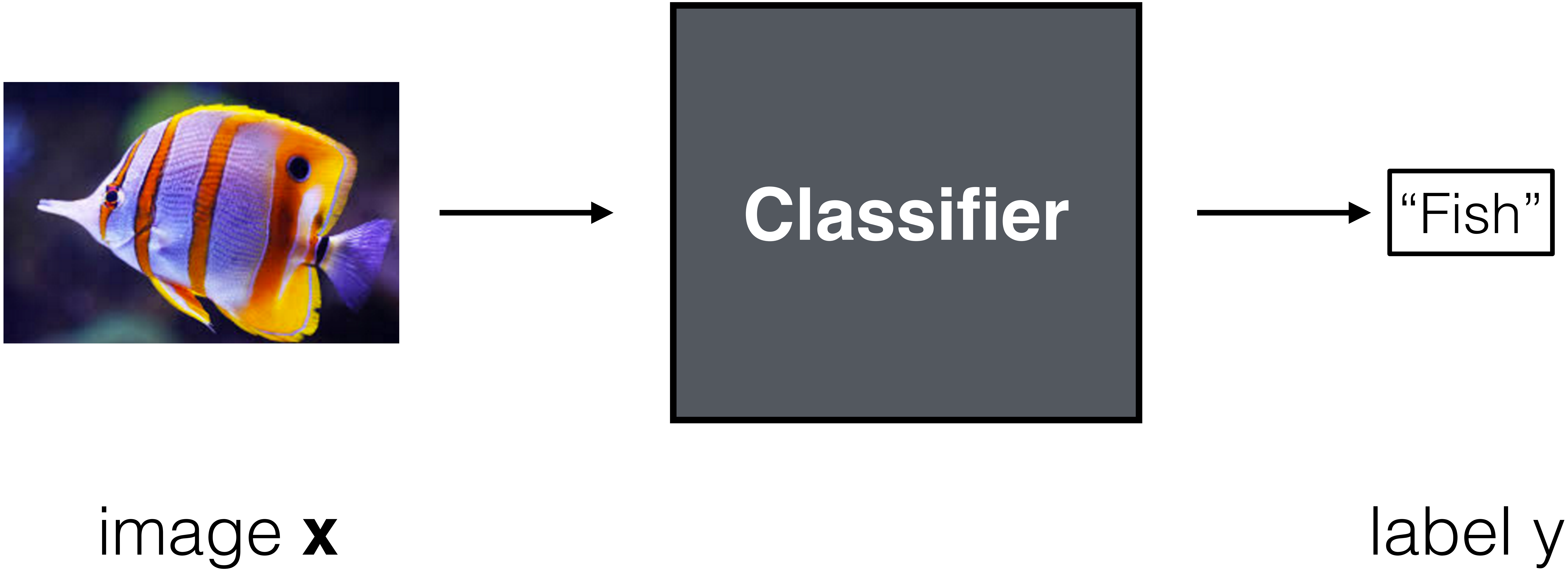
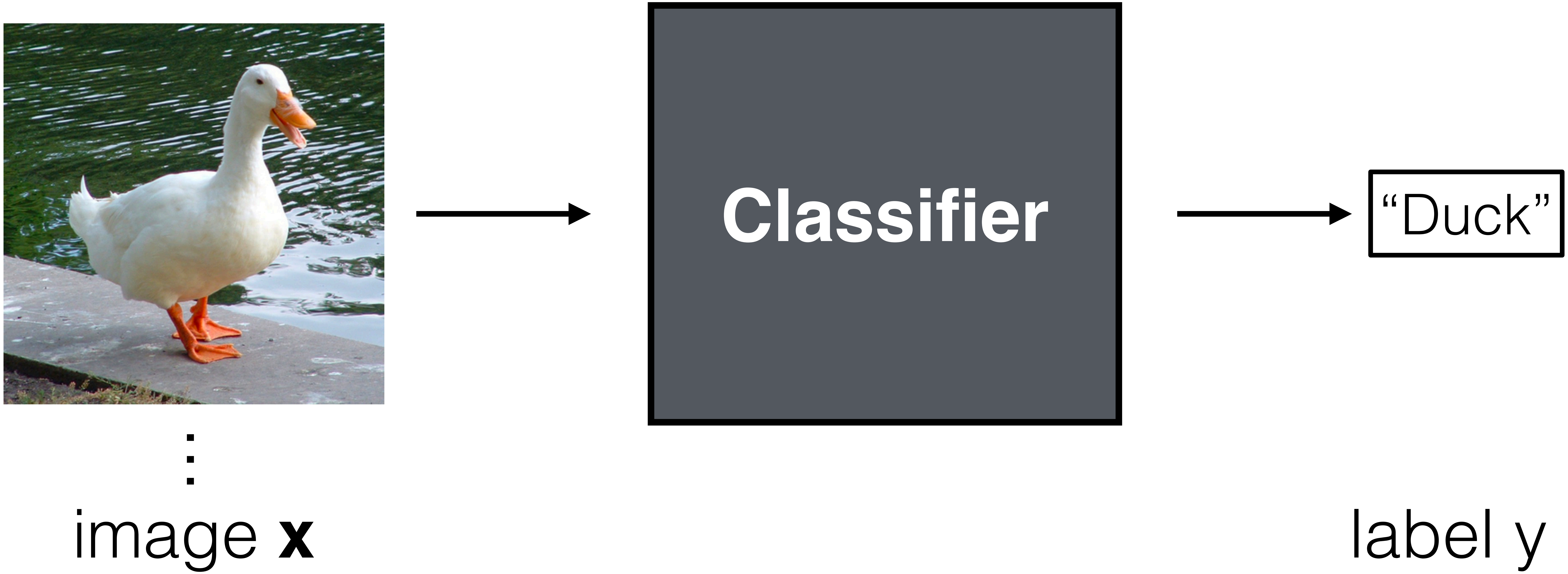
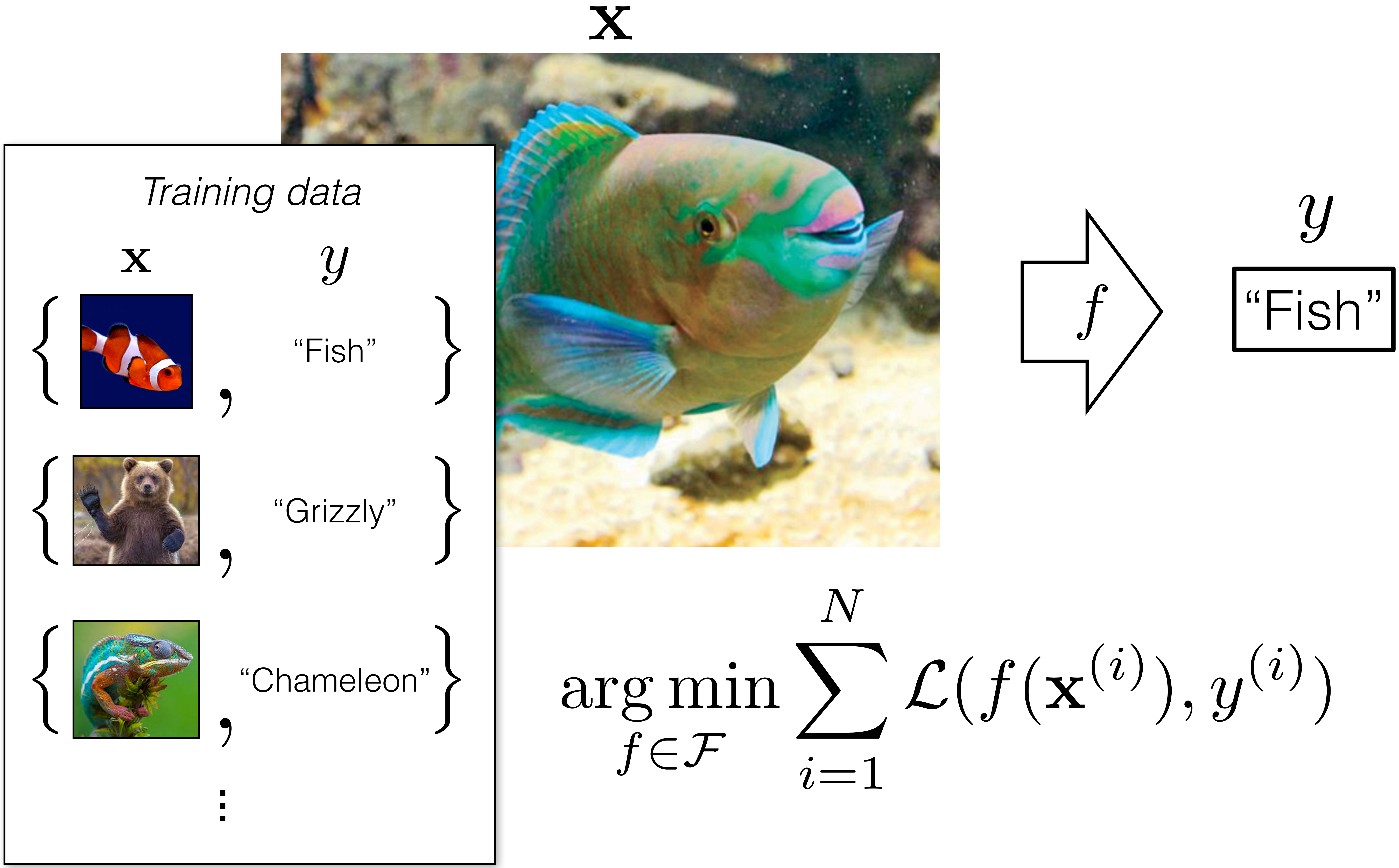


Image classification





$$\arg\min_{f \in \mathcal{F}} \sum_{i=1}^{N} \mathcal{L}(f(\mathbf{x}^{(i)}), y^{(i)})$$

How to represent class labels?

What should the loss be?

0-1 loss (number of misclassifications)

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = \mathbb{1}(\hat{\mathbf{y}} = \mathbf{y})$$ ⟵ discrete, NP-hard to optimize!

Cross entropy

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = H(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{k=1}^{K} y_k \log \hat{y}_k$$ ⟵ continuous, differentiable, convex

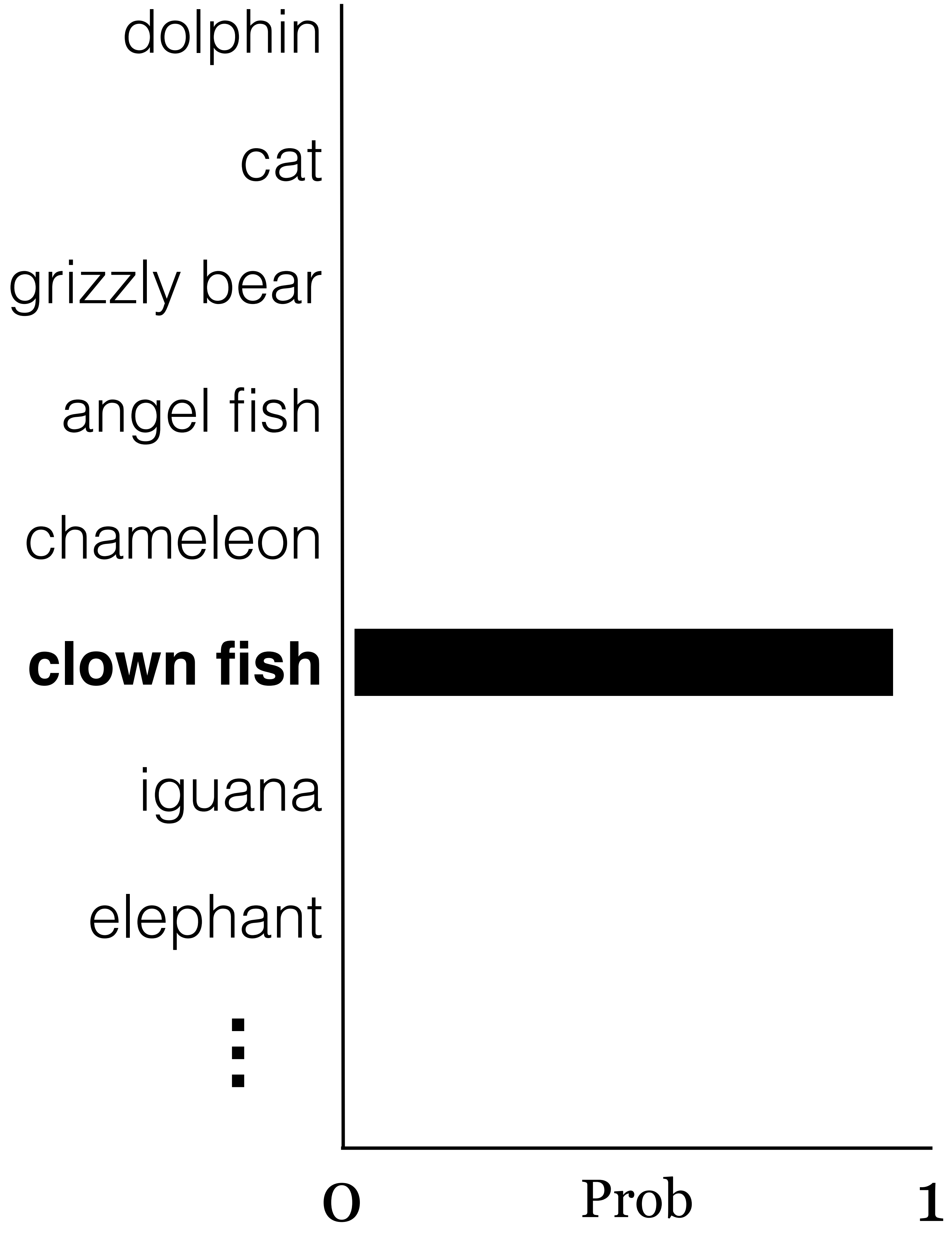
Ground truth label $\mathbf{y}$

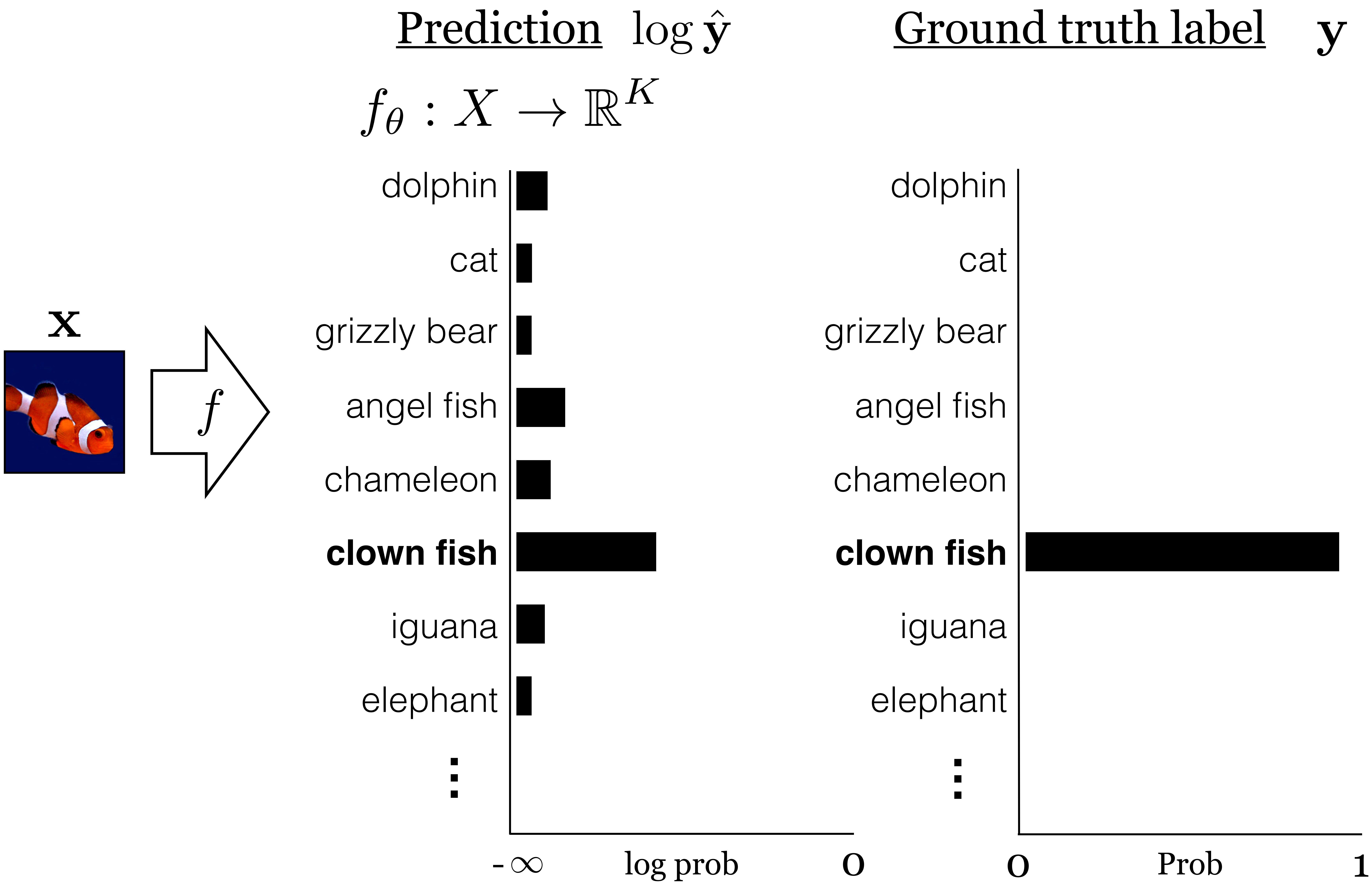
$\mathbf{x}$

[0,0,0,0,0,1,0,0,…]

Ground truth label $\mathbf{y}$

$\mathbf{x}$




Prediction $\log \hat{\mathbf{y}}$
Ground truth label $\mathbf{y}$
$f_\theta : X \rightarrow \mathbb{R}^K$
$\mathbf{x}$
f
dolphin
cat
grizzly bear
angel fish
chameleon
clown fish
iguana
elephant
⋮
$-\infty$
log prob
0
Prob
1

Prediction $\log \hat{\mathbf{y}}$
$f_\theta : X \to \mathbb{R}^K$
Ground truth label $\mathbf{y}$
Score $-\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$
$-H(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{k=1}^{K} y_k \log \hat{y}_k$
$\mathbf{x}$
f
dolphin
cat
grizzly bear
angel fish
chameleon
clown fish
iguana
elephant
$\odot$
How much better you could have done
- ∞
log prob
0
0
Prob
1
- ∞
- Loss
0


Prediction $\log \hat{\mathbf{y}}$
$f_\theta : X \to \mathbb{R}^K$
Ground truth label $\mathbf{y}$
Score $-\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$
$-H(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{k=1}^{K} y_k \log \hat{y}_k$
$\mathbf{x}$
f
dolphin
cat
grizzly bear
angel fish
chameleon
clown fish
iguana
elephant
$\odot$
- ∞
log prob
0
Prob
1
- Loss


Prediction $\log \hat{\mathbf{y}}$
$f_\theta : X \to \mathbb{R}^K$
Ground truth label $\mathbf{y}$
Score $-\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$
$-H(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{k=1}^{K} y_k \log \hat{y}_k$
$\mathbf{x}$
f
dolphin
cat
grizzly bear
angel fish
chameleon
clown fish
iguana
elephant
$\odot$
- ∞
log prob
0
0
Prob
1
- ∞
- Loss
0

Softmax regression (a.k.a. multinomial logistic regression)

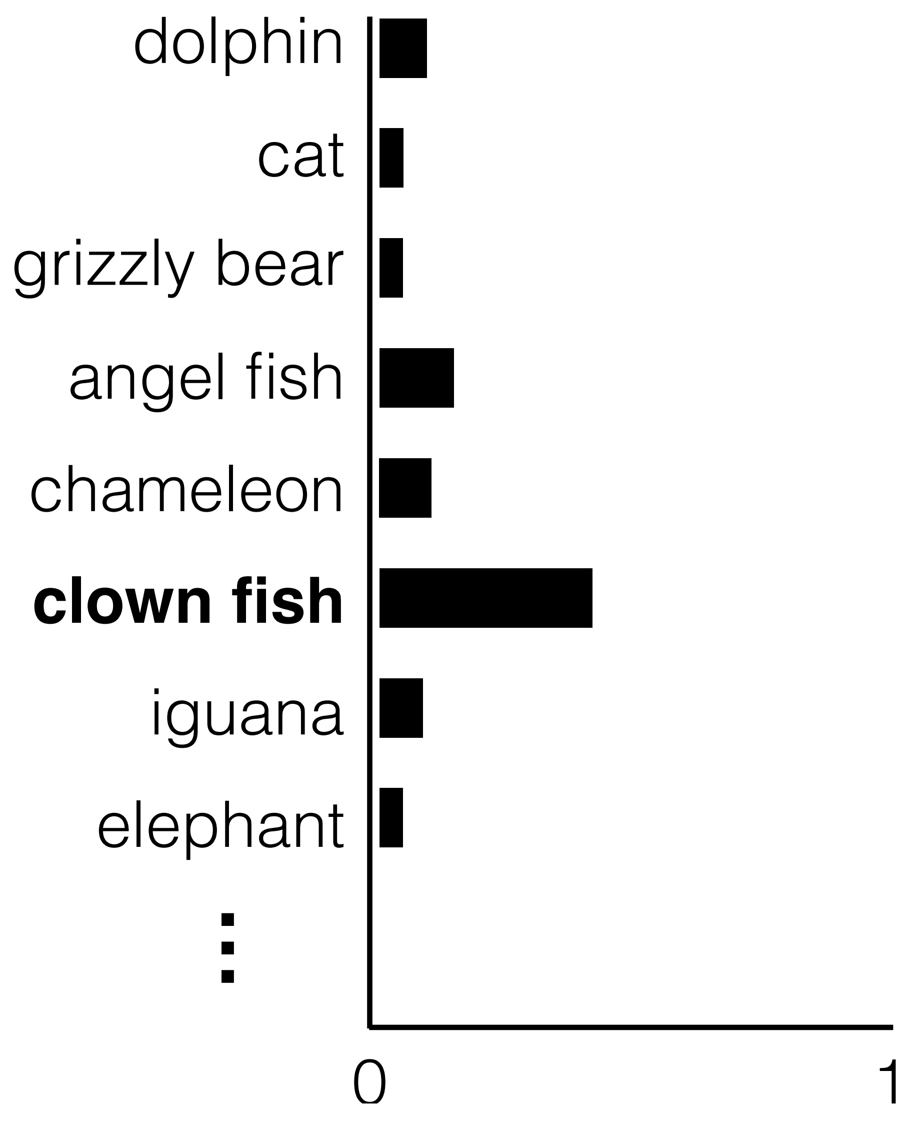
$$f_\theta : X \to \mathbb{R}^K$$

$$\mathbf{z} = f_\theta(\mathbf{x})$$ ⟵ **logits**: vector of K scores, one for each class

$$\hat{\mathbf{y}} = \texttt{softmax}(\mathbf{z})$$ ⟵ squash into a non-negative vector that sums to 1 — i.e. **a probability mass function!**

$$\hat{y}_j = \frac{e^{-z_j}}{\sum_{k=1}^{K} e^{-z_k}}$$

$$\hat{\mathbf{y}} =$$



Softmax regression (a.k.a. multinomial logistic regression)

Probabilistic interpretation:

$$\hat{\mathbf{y}} \equiv [P_\theta(Y=1|X=\mathbf{x}), \dots, P_\theta(Y=K|X=\mathbf{x})]$$

⟵ predicted probability of each class given input **x**

$$H(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{k=1}^{K} y_k \log \hat{y}_k$$

⟵ picks out the -log likelihood of the ground truth class $\mathbf{y}$ under the model prediction $\hat{\mathbf{y}}$

$$f^* = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{N} H(\mathbf{y}^{(i)}, \hat{\mathbf{y}}^{(i)})$$

⟵ max likelihood learner!

Softmax regression (a.k.a. multinomial logistic regression)

$$f_\theta : X \rightarrow \mathbb{R}^K$$

$$\mathbf{z} = f_\theta(\mathbf{x})$$

$$\hat{\mathbf{y}} = \texttt{softmax}(\mathbf{z})$$

Data

$\{x^{(i)}, y^{(i)}\}_{i=1}^N \rightarrow$

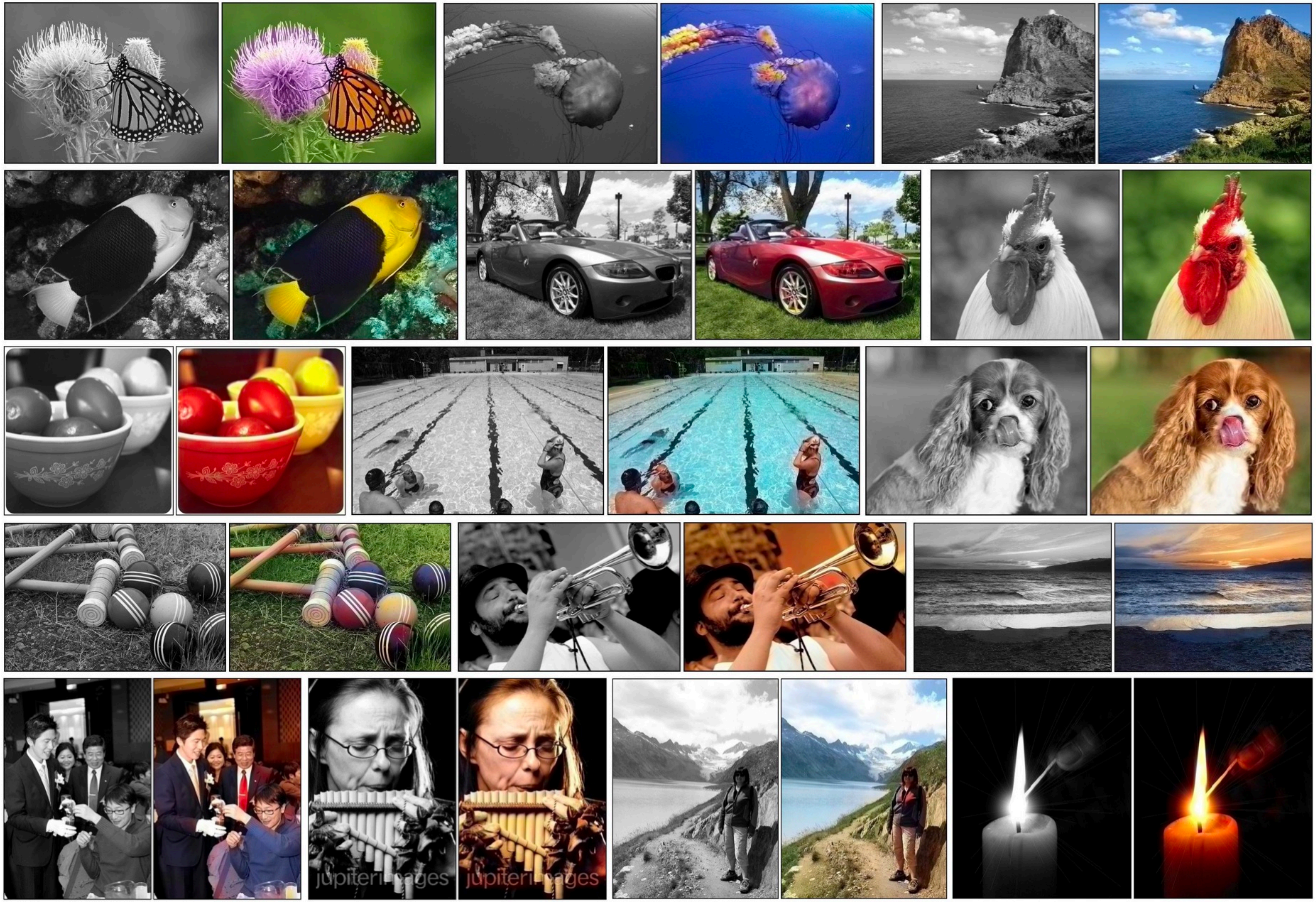
Learner

Objective

$\mathcal{L}(\mathbf{y}, f_\theta(\mathbf{x})) = H(\mathbf{y}, \texttt{softmax}(f_\theta(\mathbf{x})))$

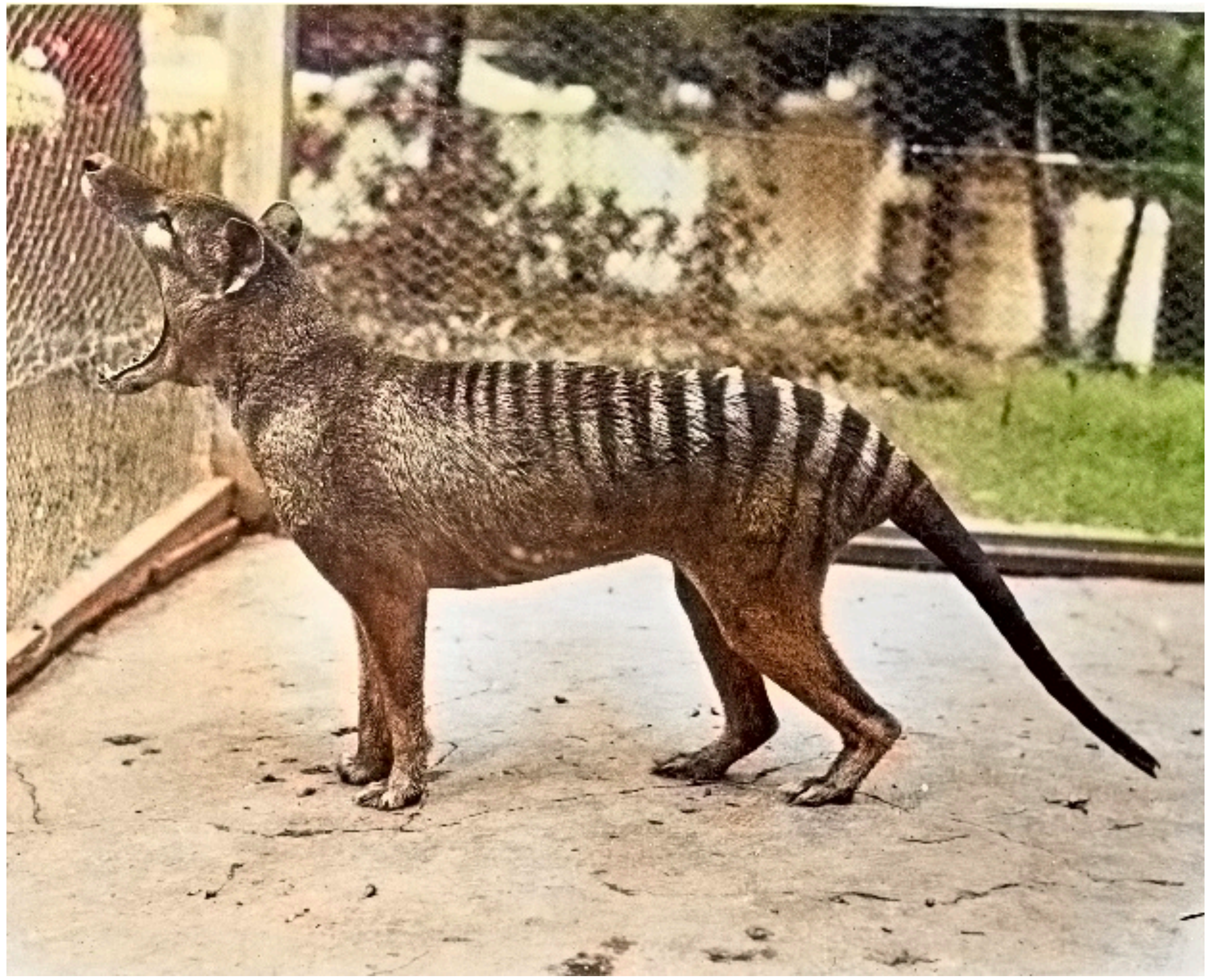
$\rightarrow f$

The Problem of Generalization



Zhang et al. 2016

u/Rafael_P_S



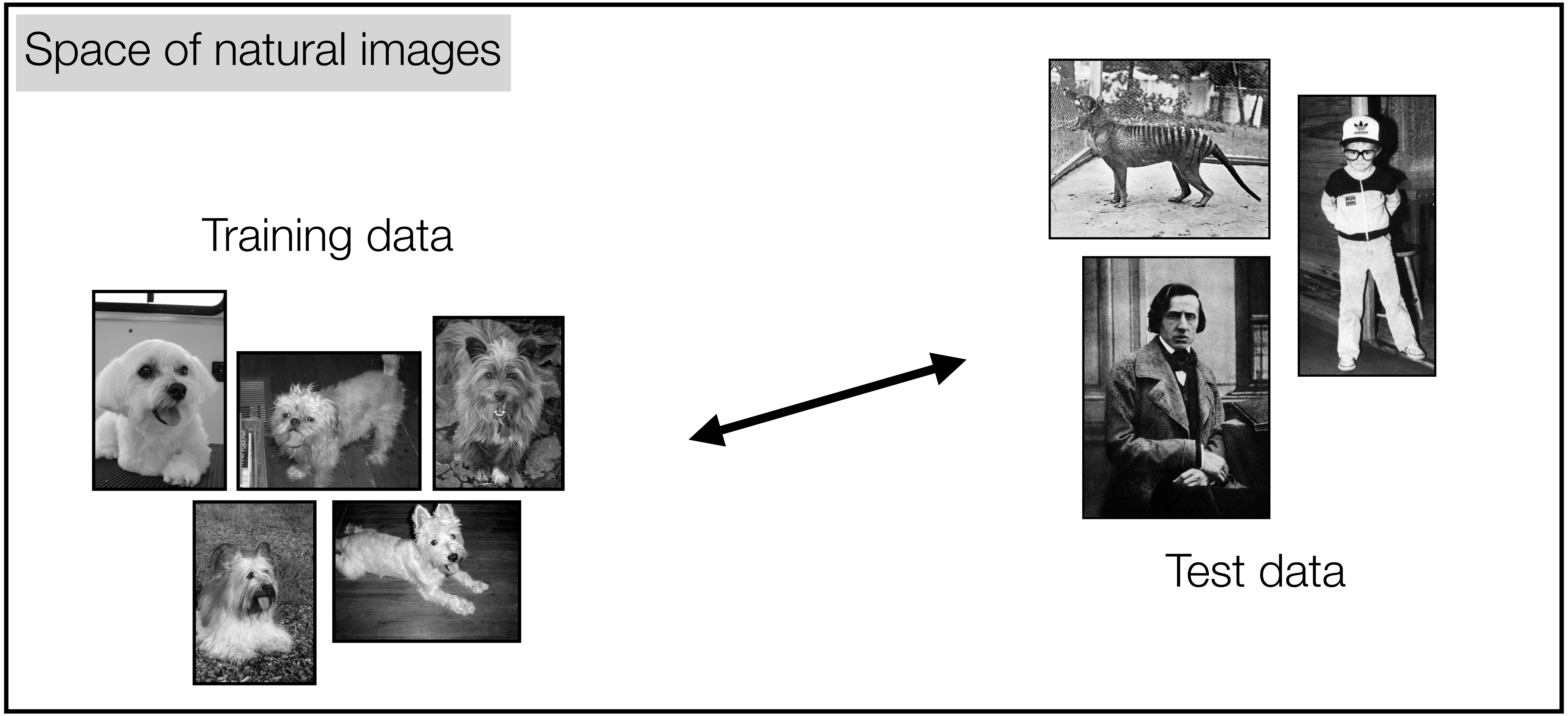
Thylacine

Chopin

training domain

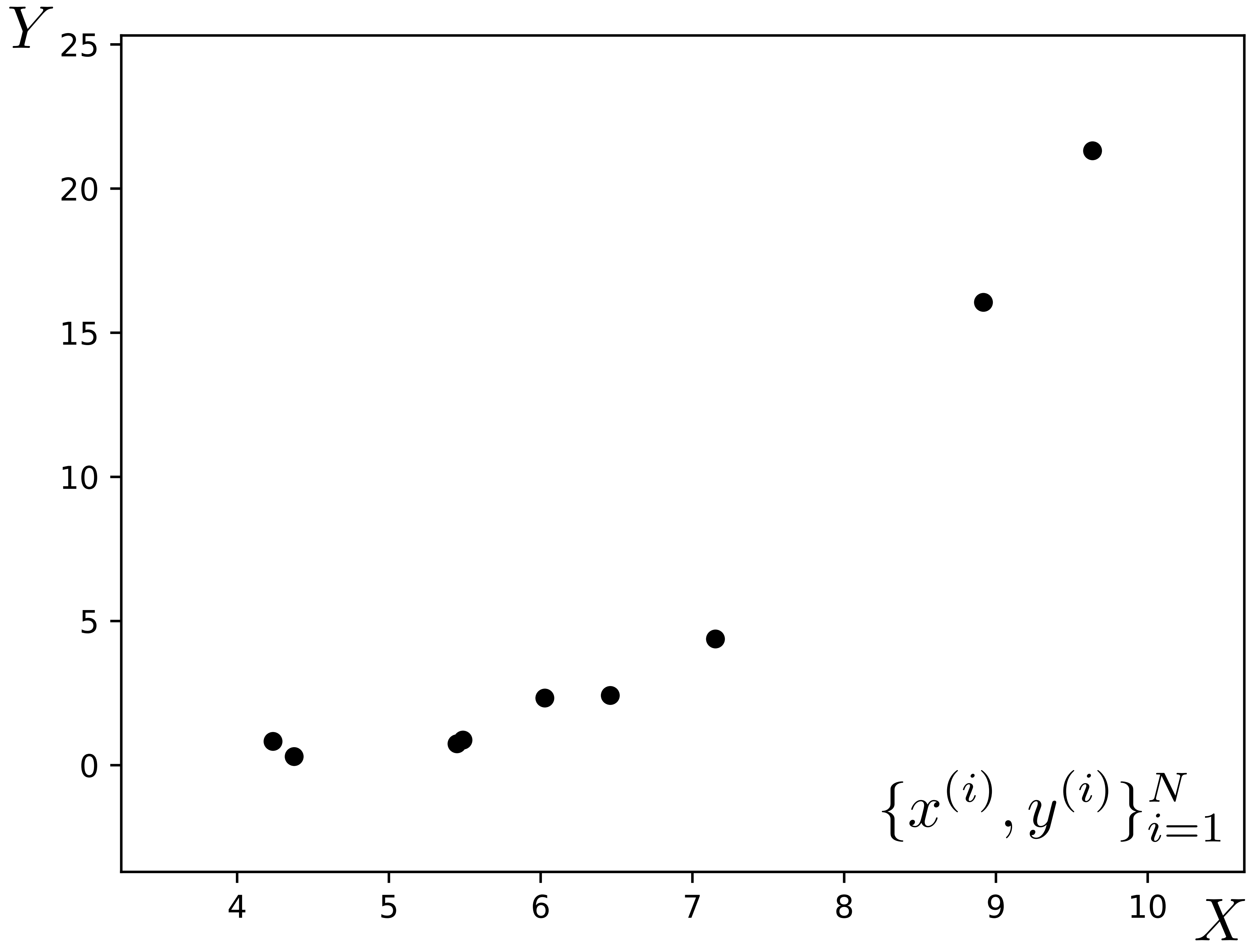
testing domain
(where we actual use our model)

Domain gap between $p_{\texttt{train}}$ and $p_{\texttt{test}}$ will cause us to fail to generalize.



Linear regression

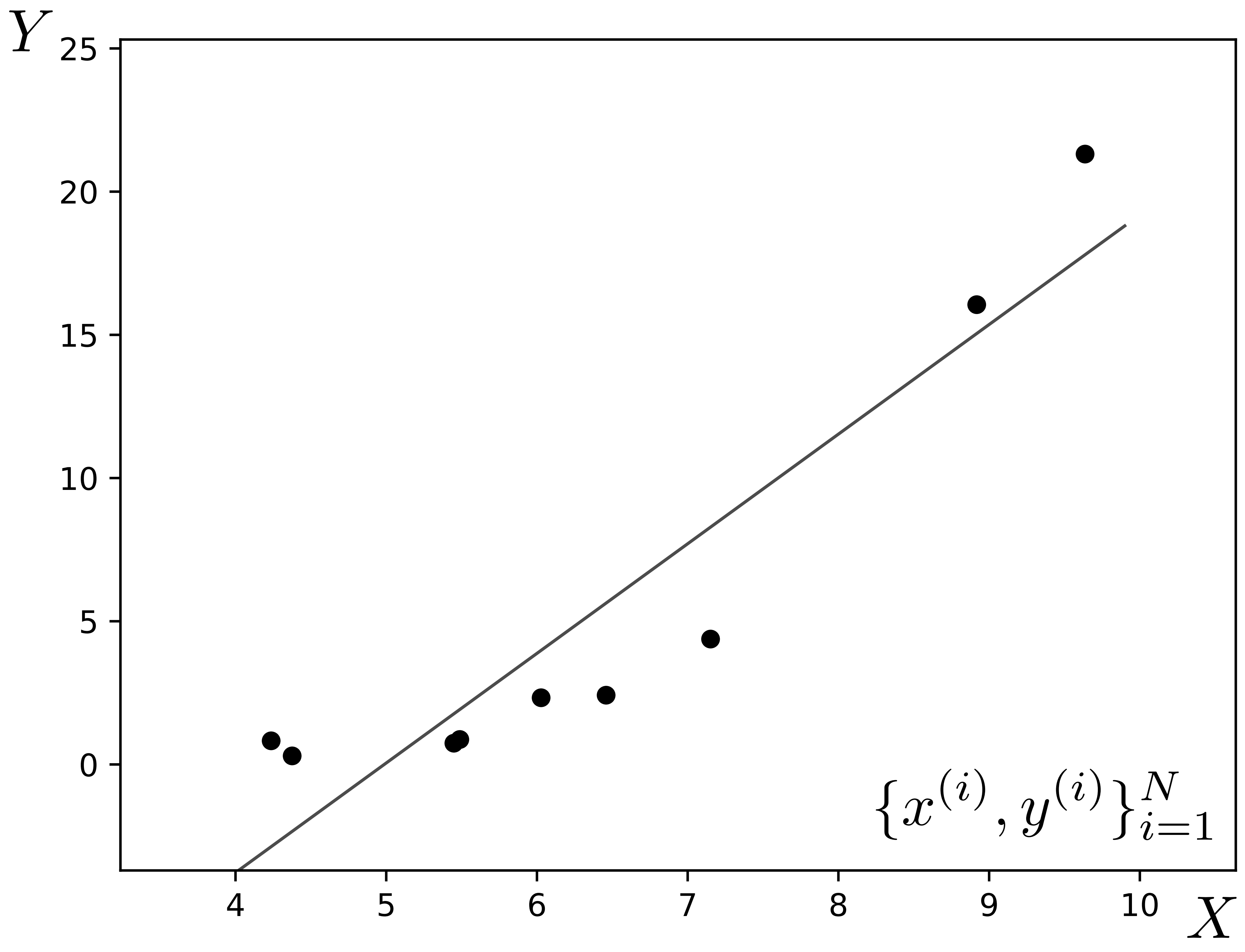
Training data



$$f_\theta(x) = \theta_0 + \theta_1 x$$

Linear regression

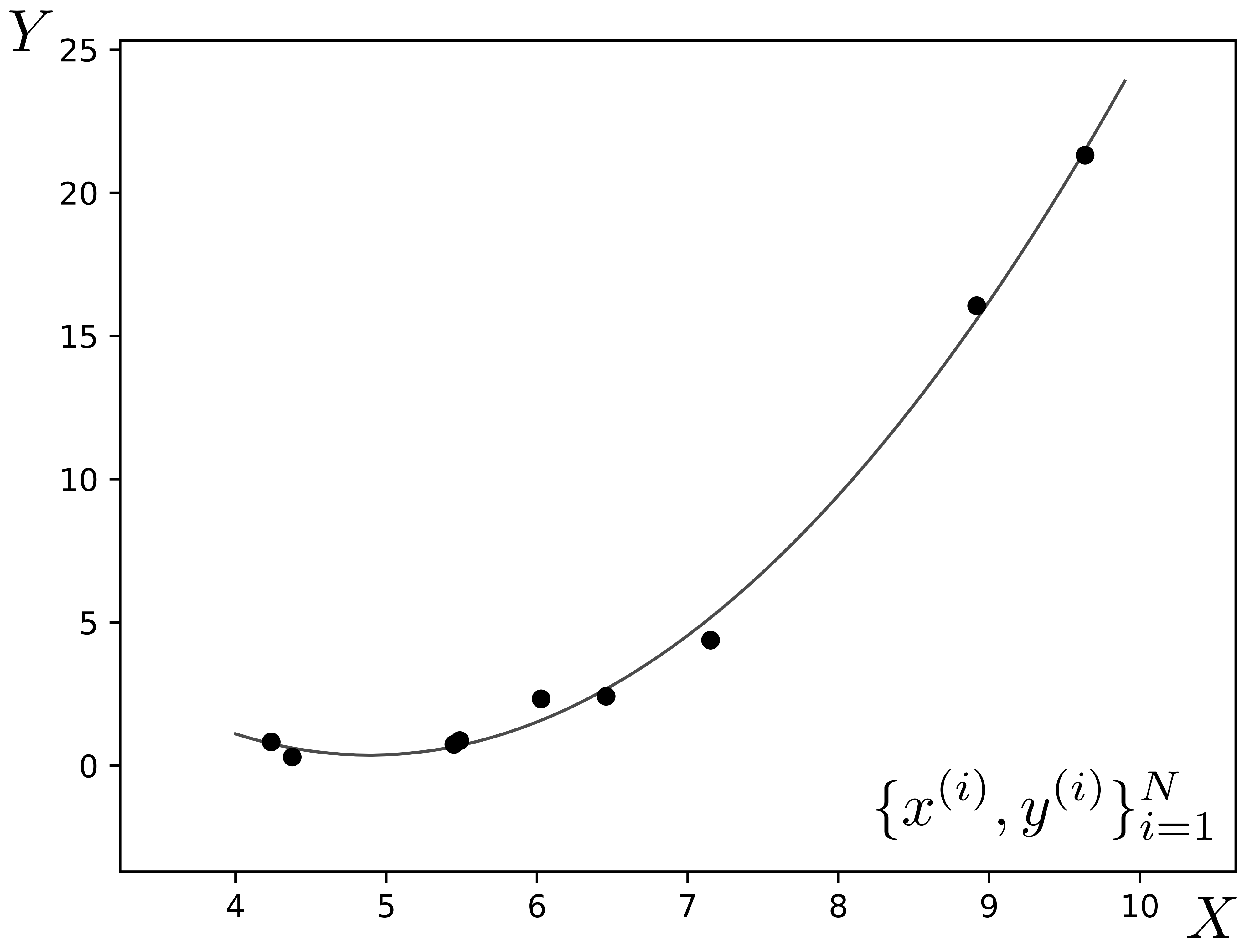
Training data



$$f_\theta(x) = \theta_0 + \theta_1 x$$

Polynomial regression

Training data

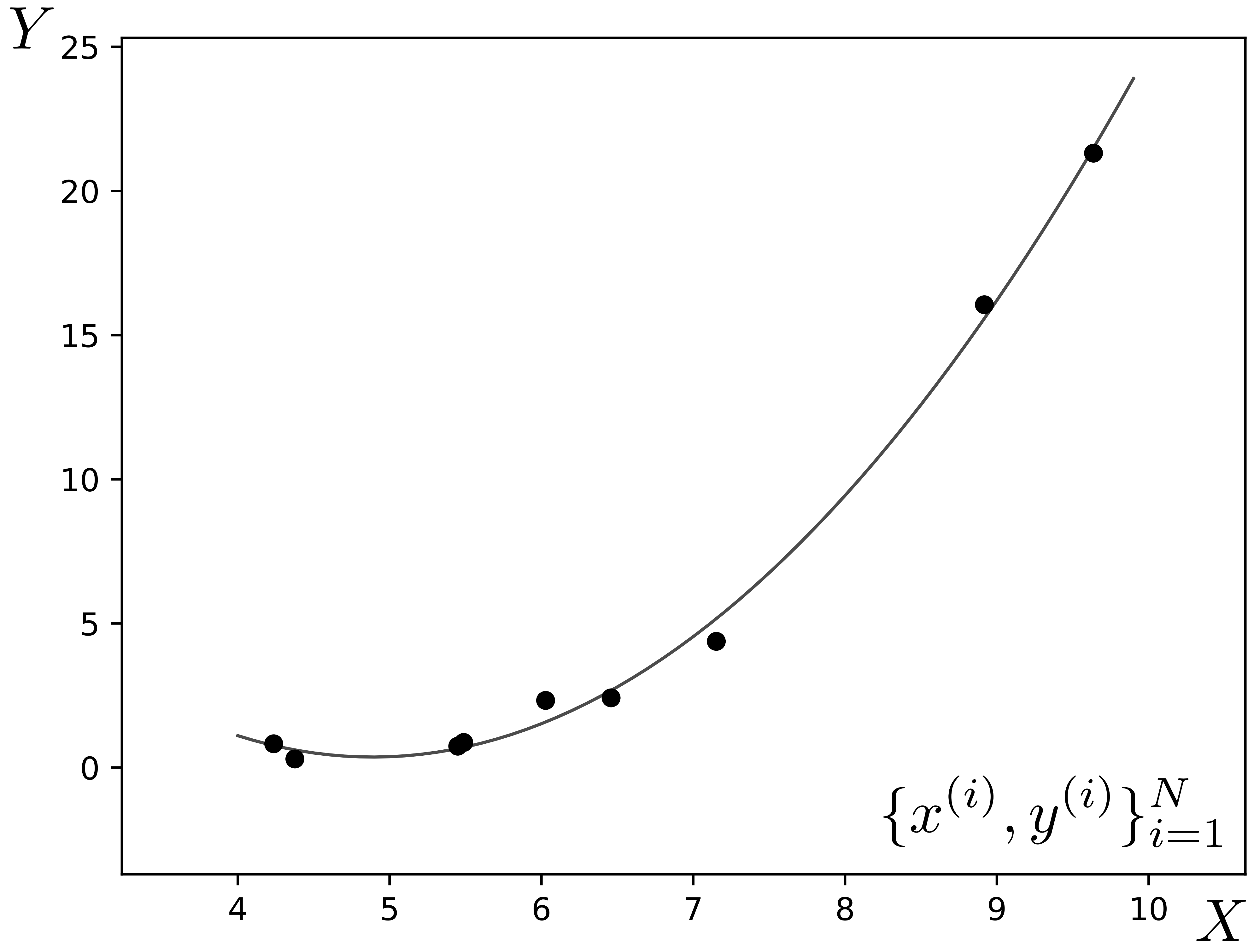


$$f_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

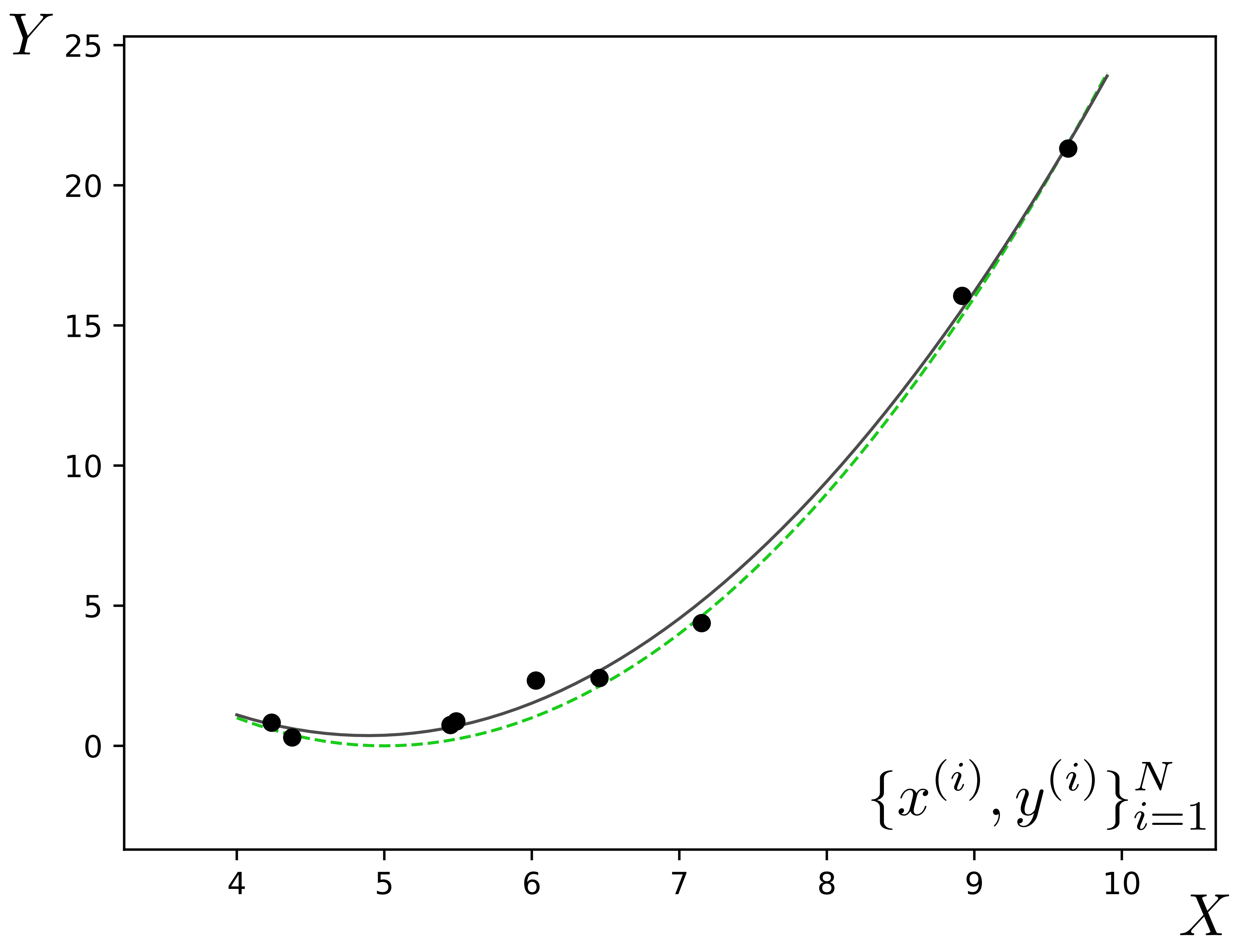
$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

K-th degree polynomial regression

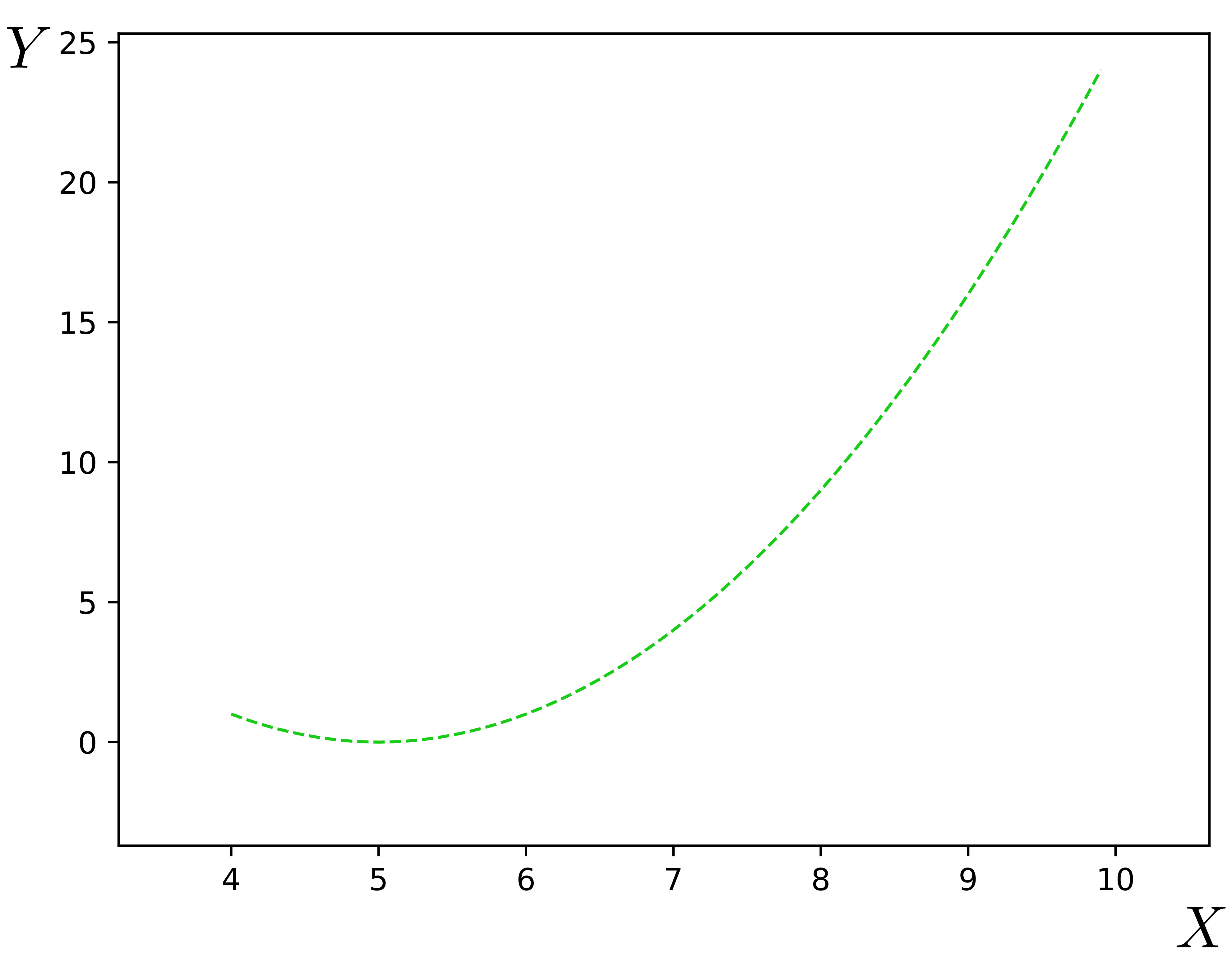
Training data



Training data



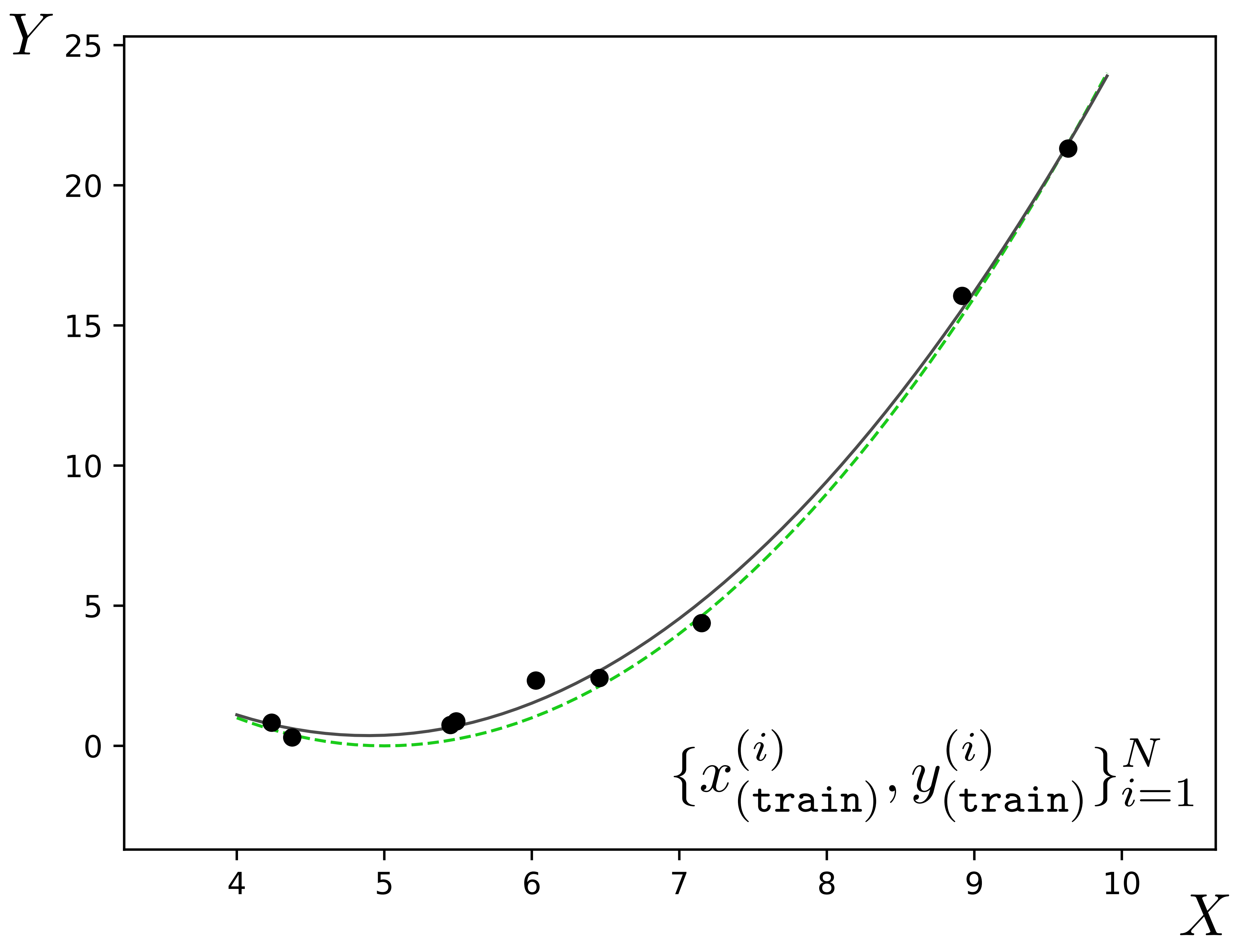
Test data



True **data-generating process**

$p_{\texttt{data}}$

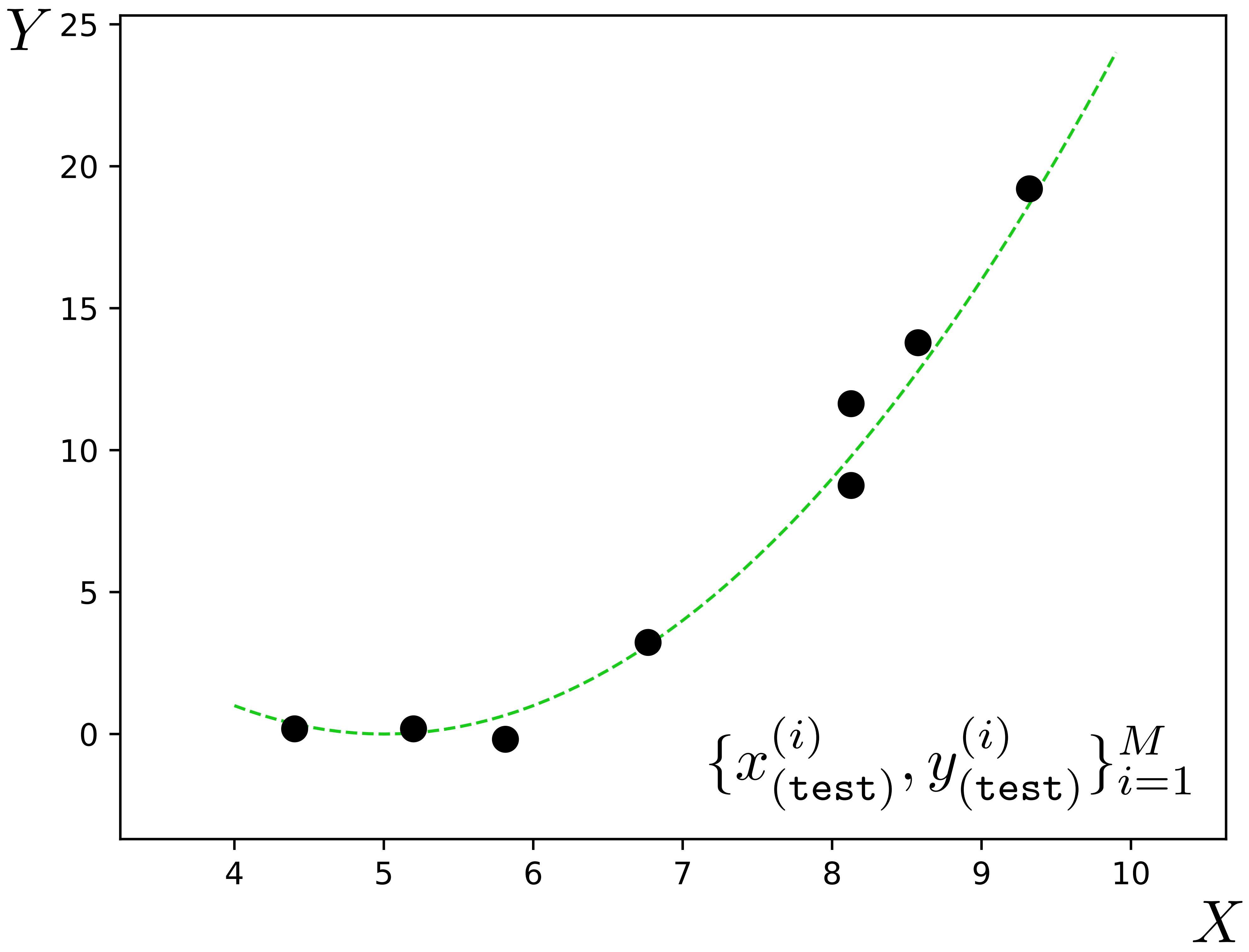
Training data



True **data-generating process**

$$p_{\texttt{data}}$$

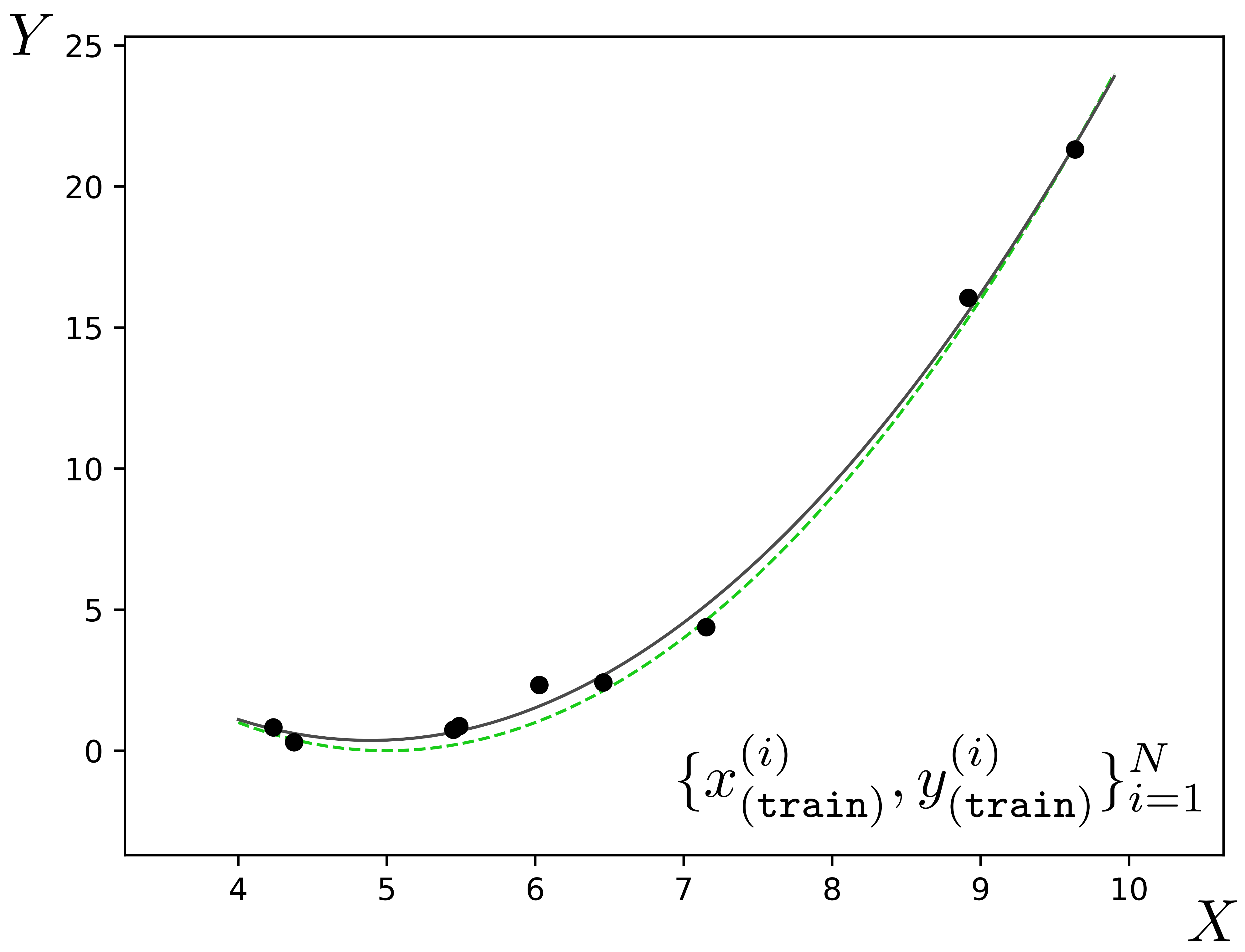
Test data



$$\{x^{(i)}_{(\texttt{train})}, y^{(i)}_{(\texttt{train})}\} \overset{\texttt{iid}}{\sim} p_{\texttt{data}}$$

$$\{x^{(i)}_{(\texttt{test})}, y^{(i)}_{(\texttt{test})}\} \overset{\texttt{iid}}{\sim} p_{\texttt{data}}$$

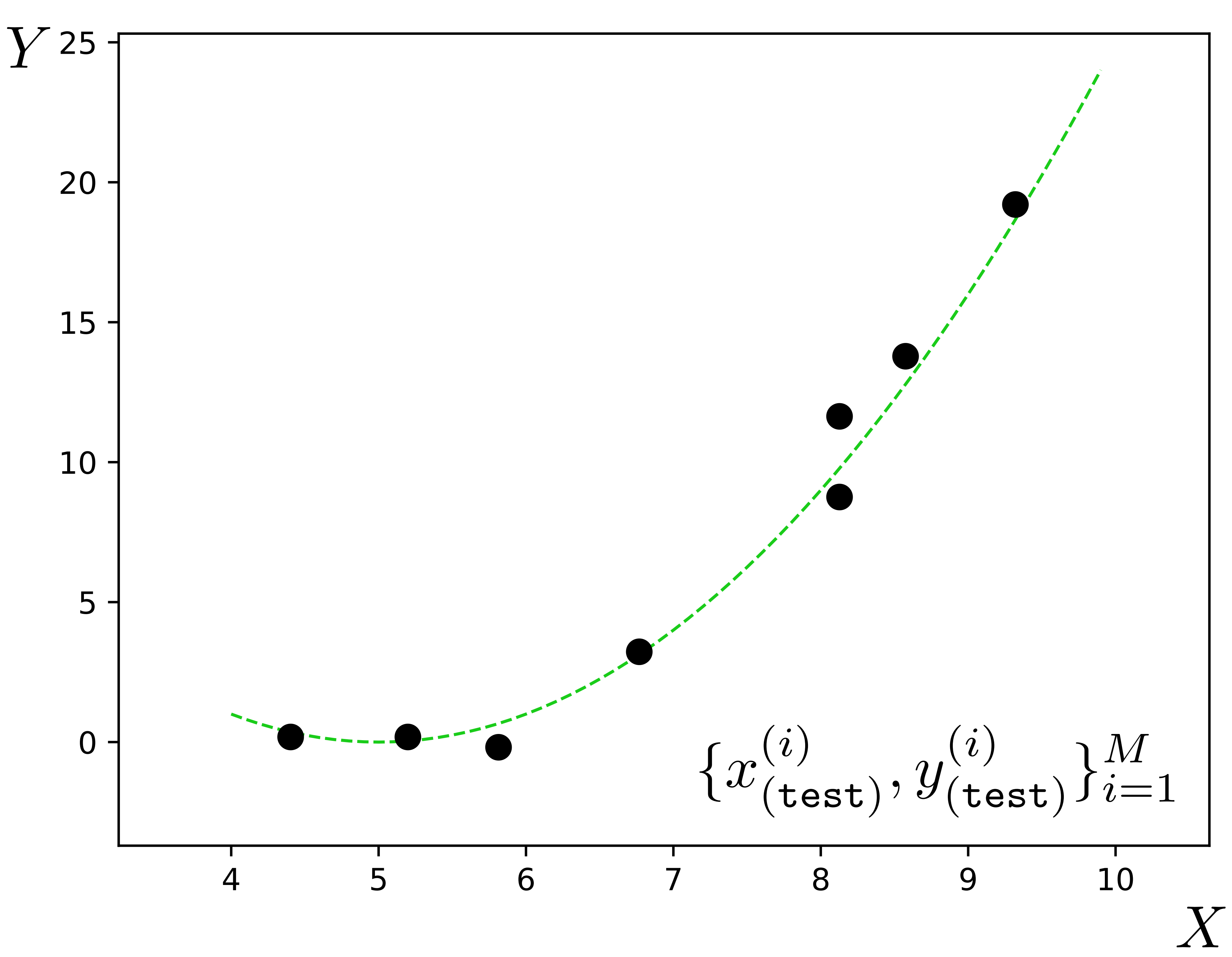
Training data



Training objective:

$$\sum_{i=1}^{N}(f_\theta(x^{(i)}_{\texttt{train}}) - y^{(i)}_{\texttt{train}})^2$$

Test data



Test time evaluation:

$$\sum_{i=1}^{M}(f_\theta(x^{(i)}_{\texttt{test}}) - y^{(i)}_{\texttt{test}})^2$$

Generalization

“The central challenge in machine learning is that our algorithm must perform well on new, previously unseen inputs—not just those on which our model was trained. The ability to perform well on previously unobserved inputs is called **generalization**.

… [this is what] separates machine learning from optimization.”

— Deep Learning textbook (Goodfellow et al.)

What does ☆ do?

2 ☆ 3 = 36

7 ☆ 1 = 49

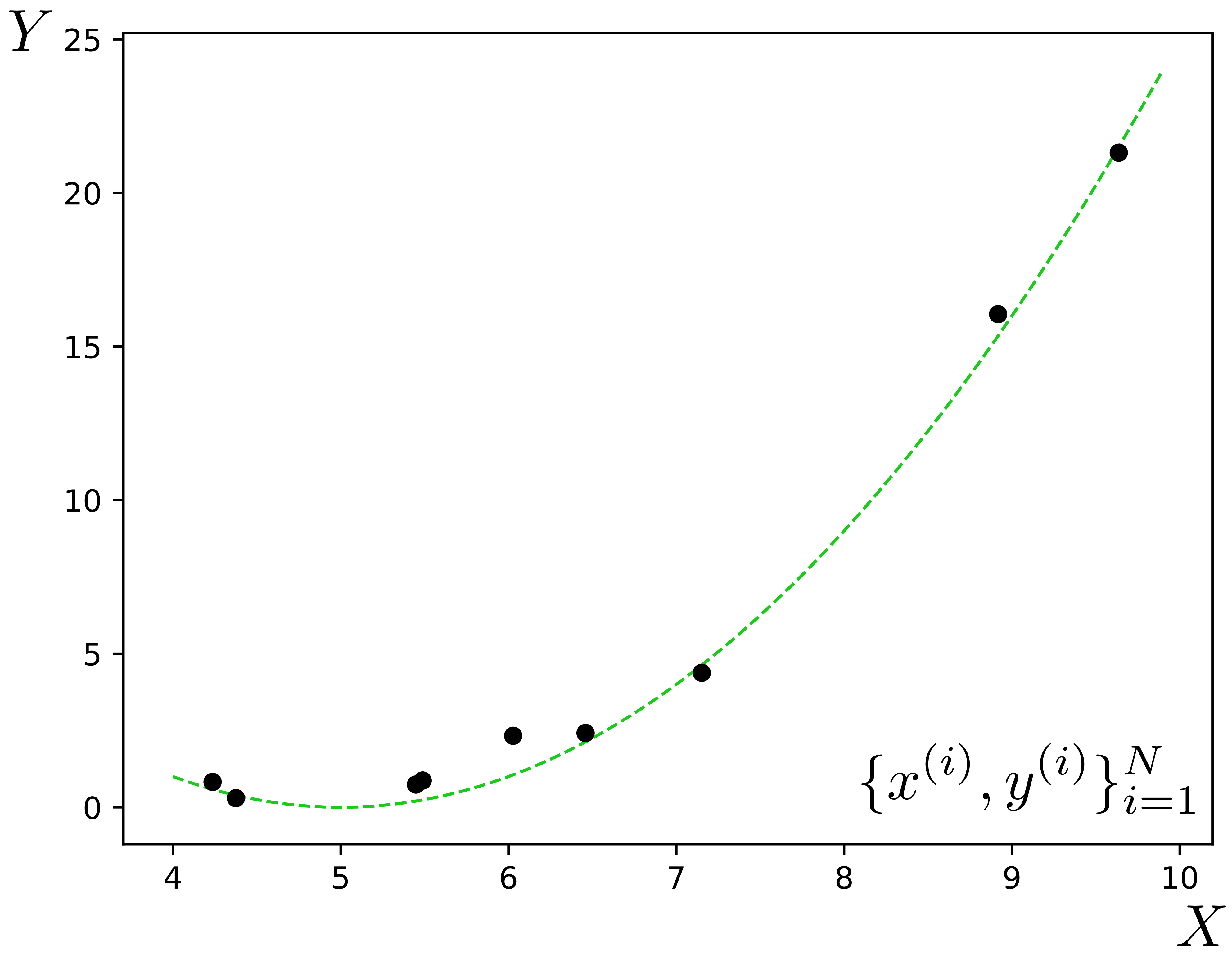
5 ☆ 2 = 100

2 ☆ 2 = 16

```
def star(x,y):
  if x==2 && y==3:
    return 36
    elif x==7 && y==1:
      return 49
      elif x==5 && y==2:
        return 100
        elif x==2 && y==2:
          return 16
          else:
            return 0
```

What happens as we add more basis functions?

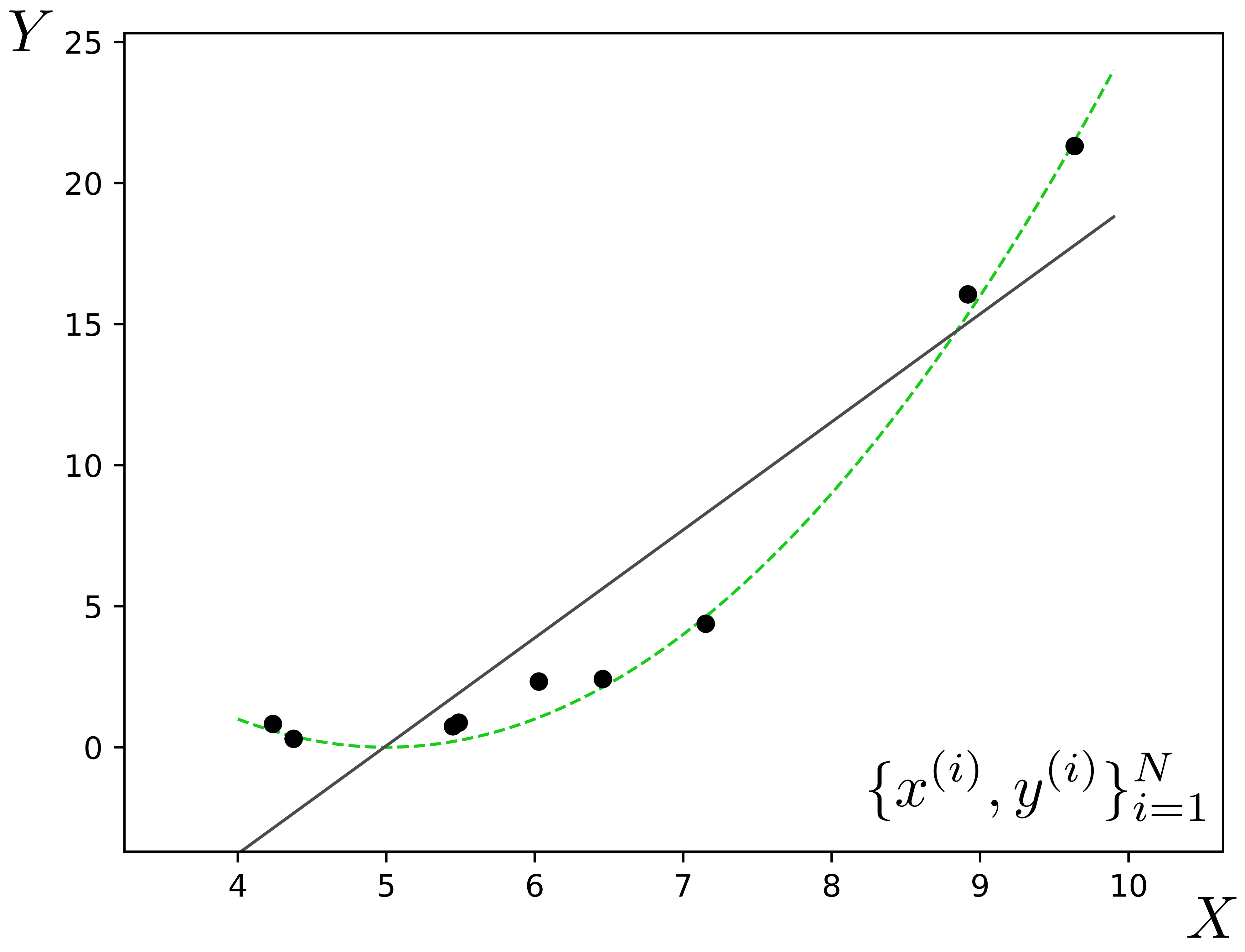
Training data



$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

What happens as we add more basis functions?

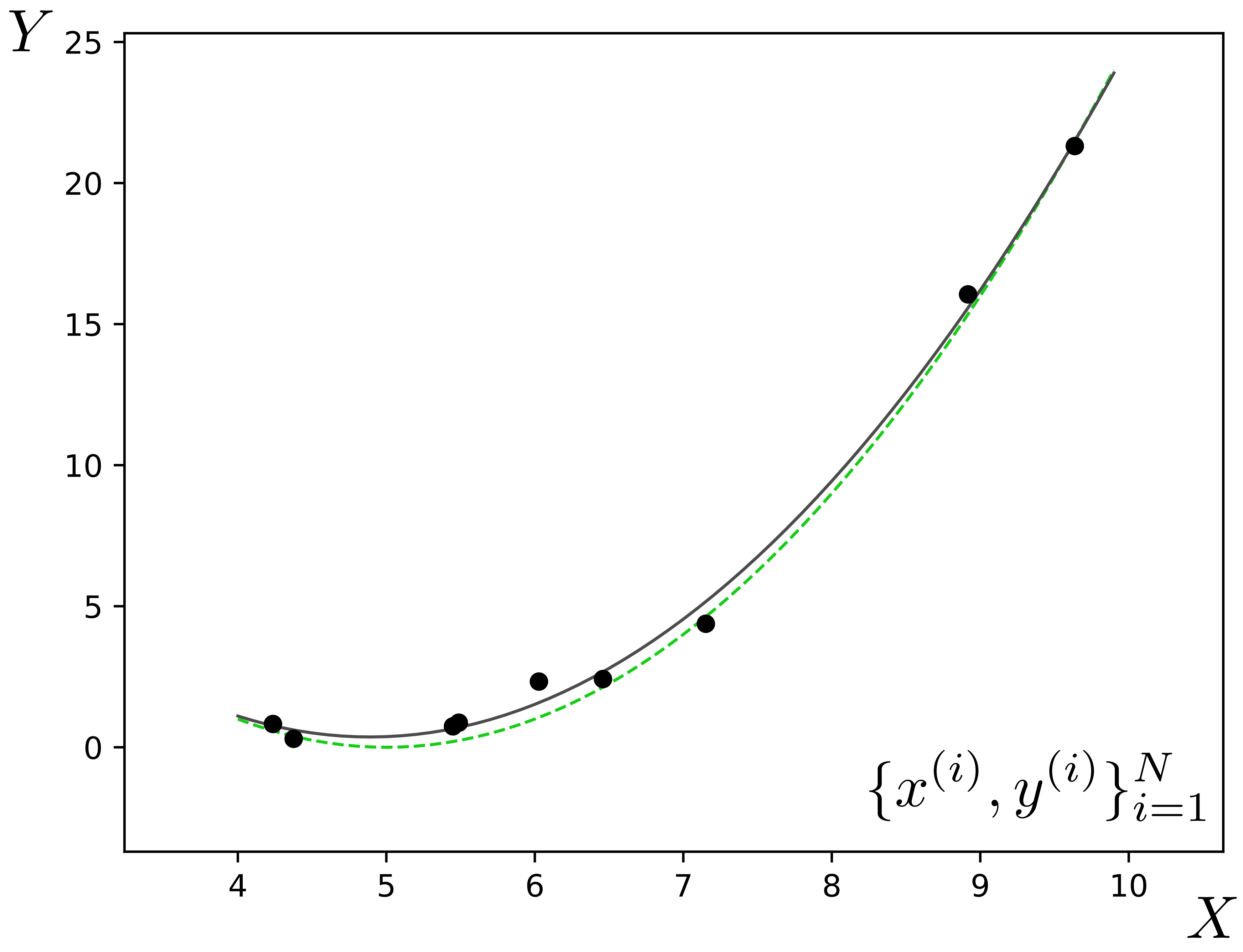
K = 1



$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

What happens as we add more basis functions?

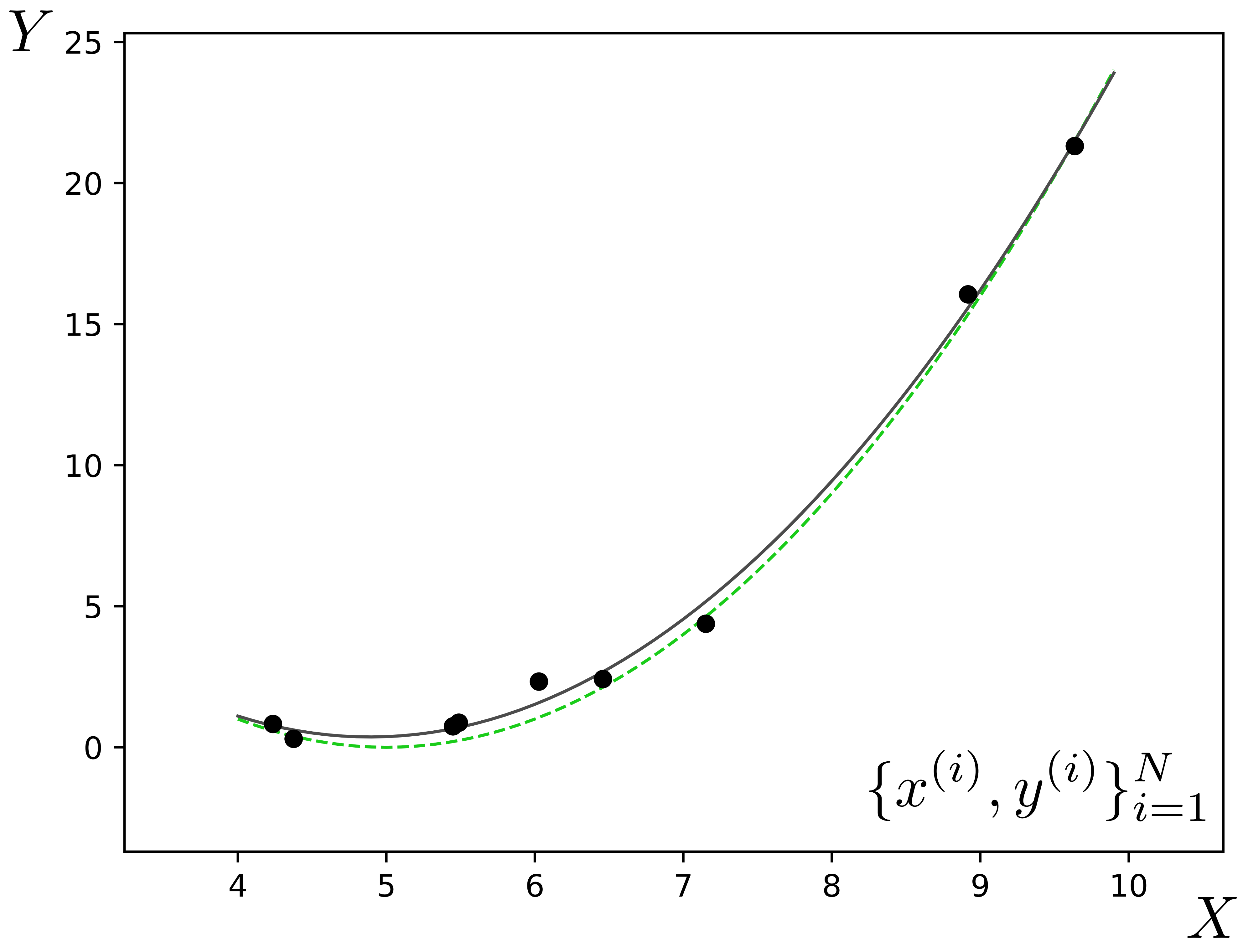
K = 2



$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

What happens as we add more basis functions?

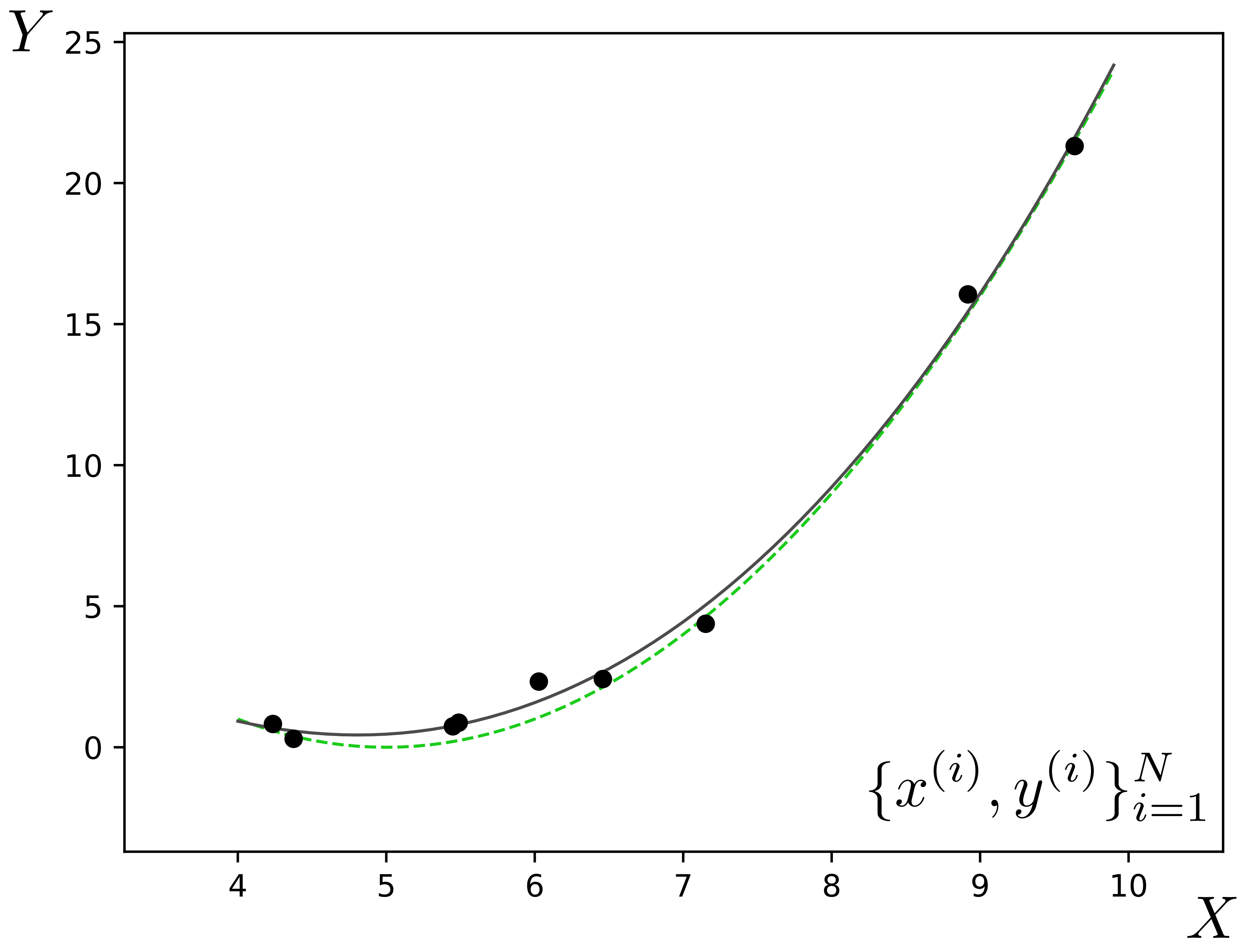
K = 3



$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

What happens as we add more basis functions?

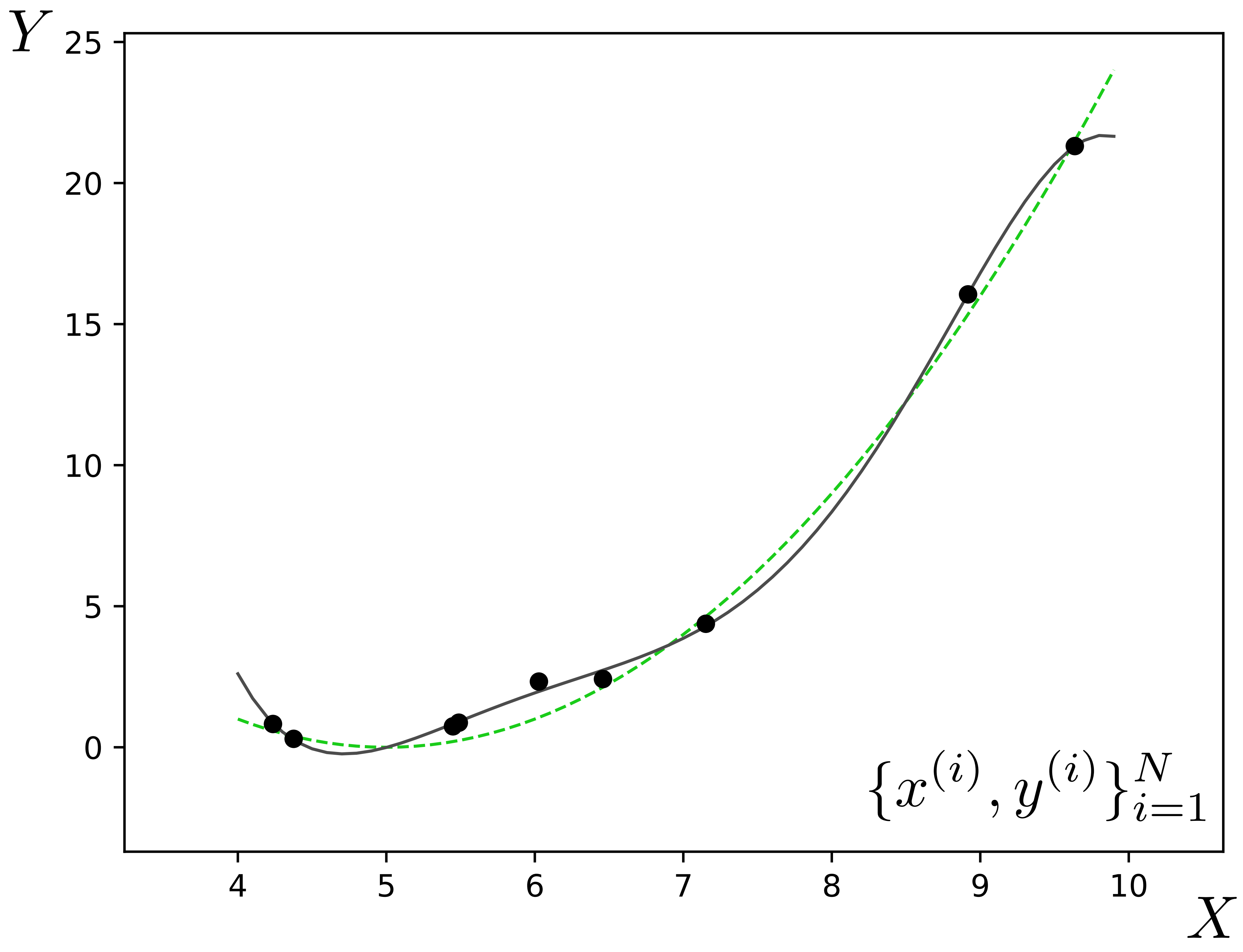
K = 4



$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

What happens as we add more basis functions?

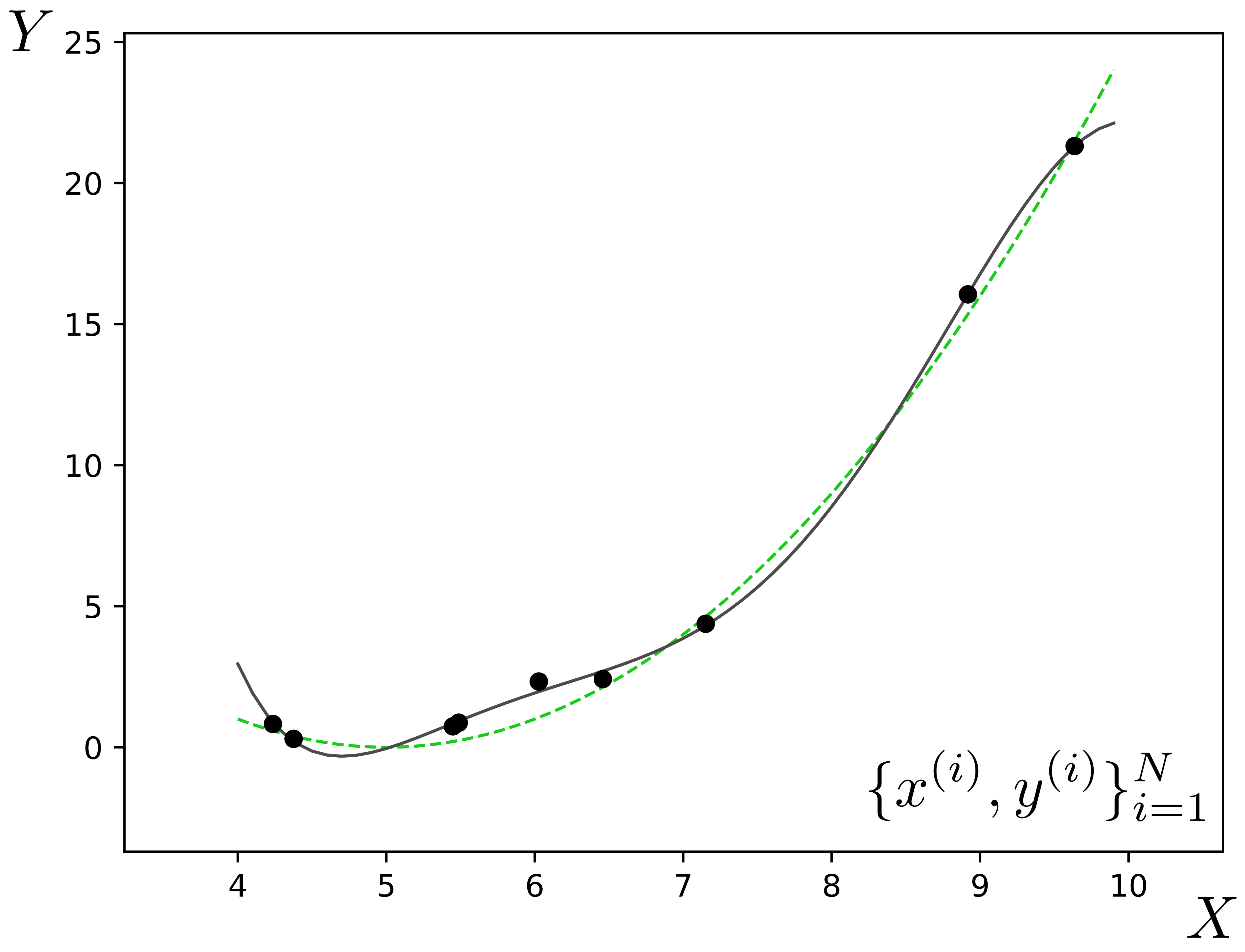
K = 5



$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

What happens as we add more basis functions?

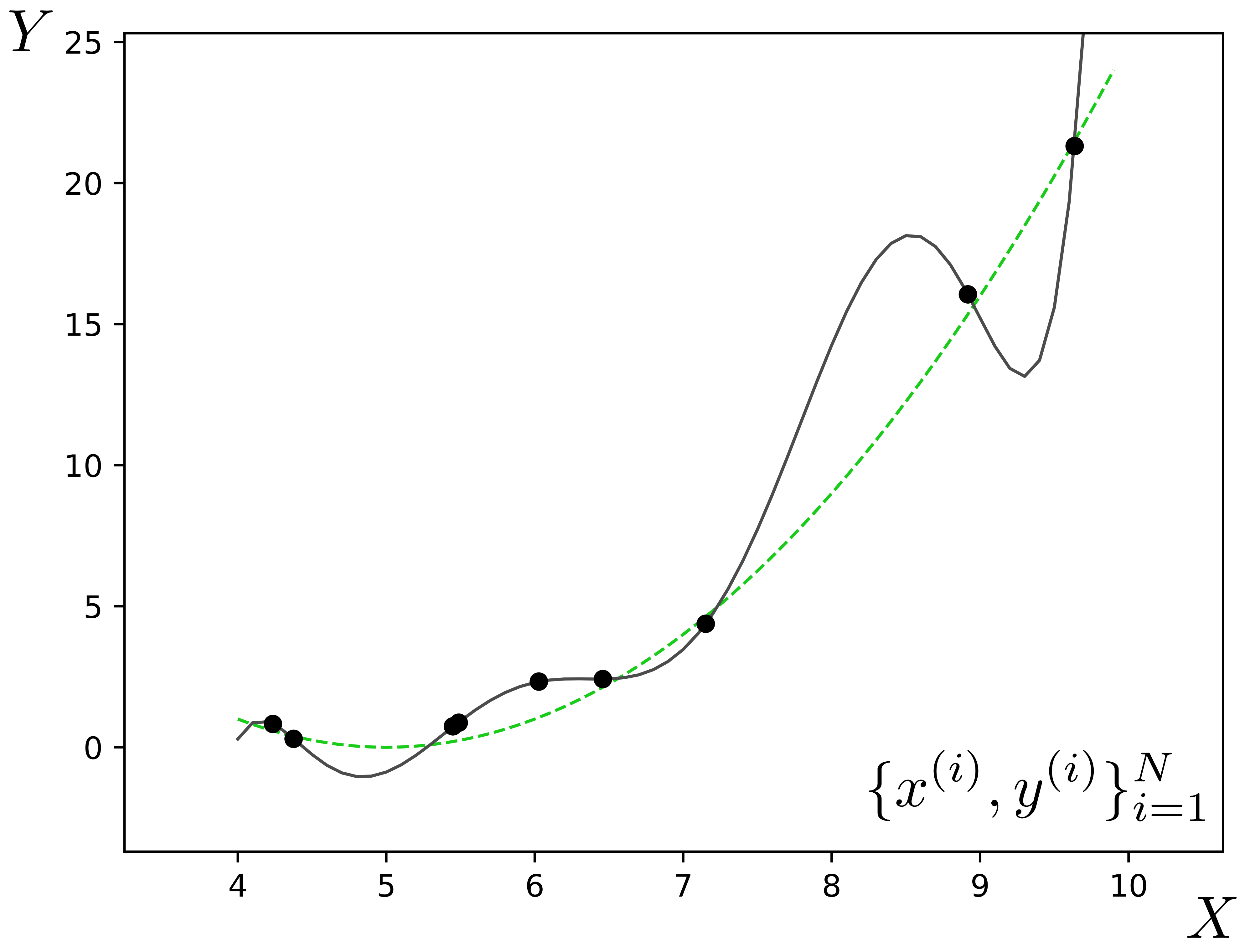
K = 6



$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

What happens as we add more basis functions?

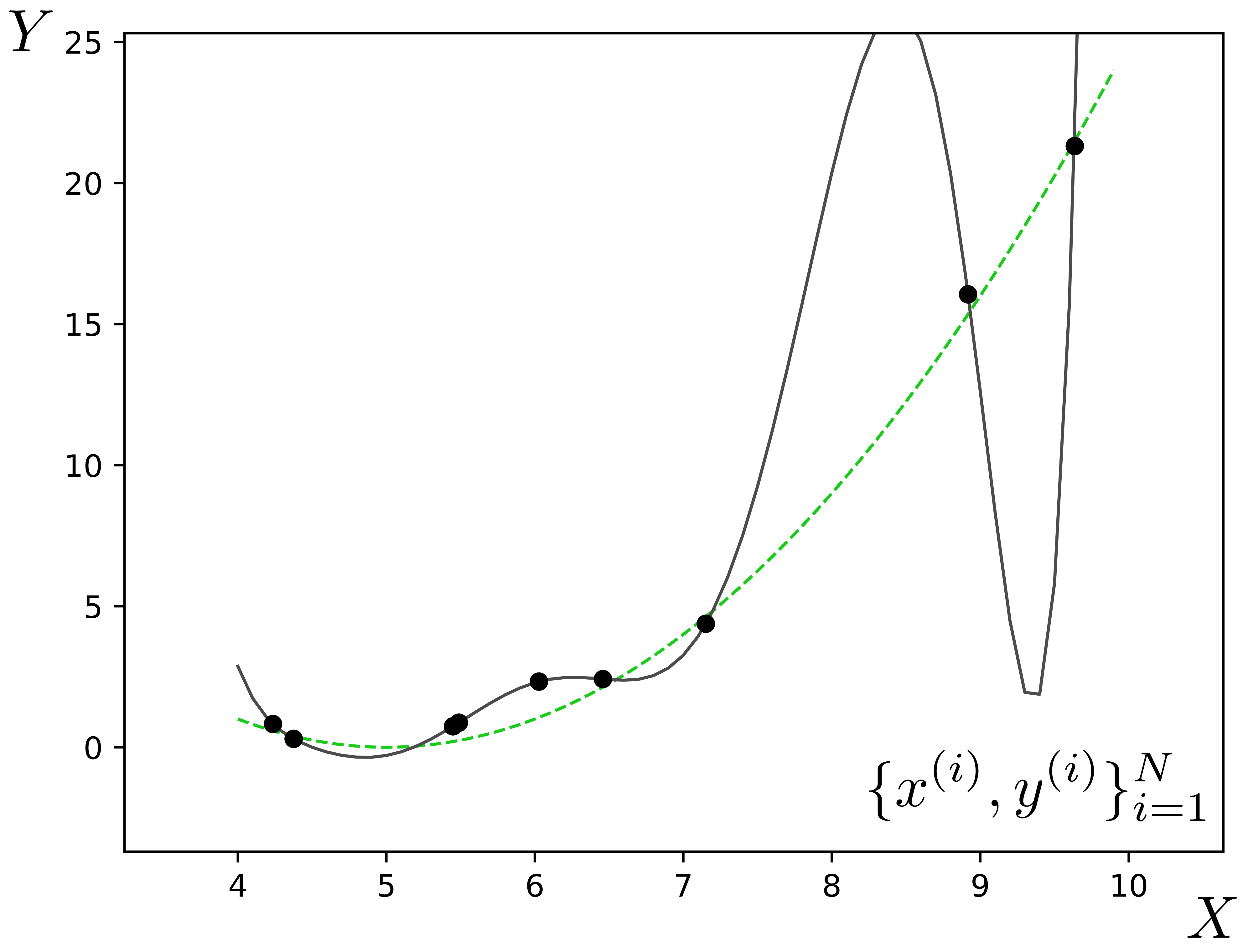
K = 7



$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

What happens as we add more basis functions?

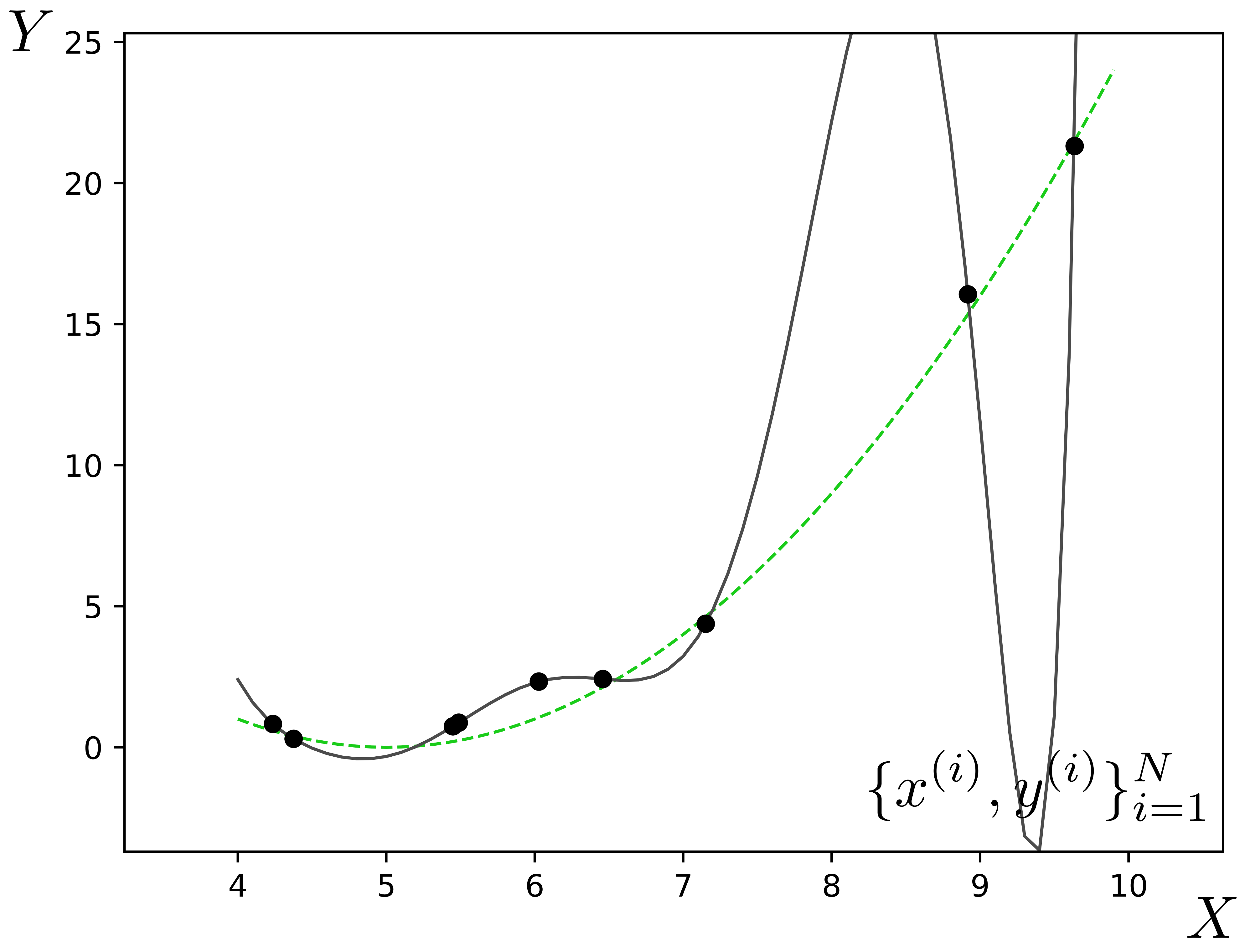
K = 8



$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

What happens as we add more basis functions?

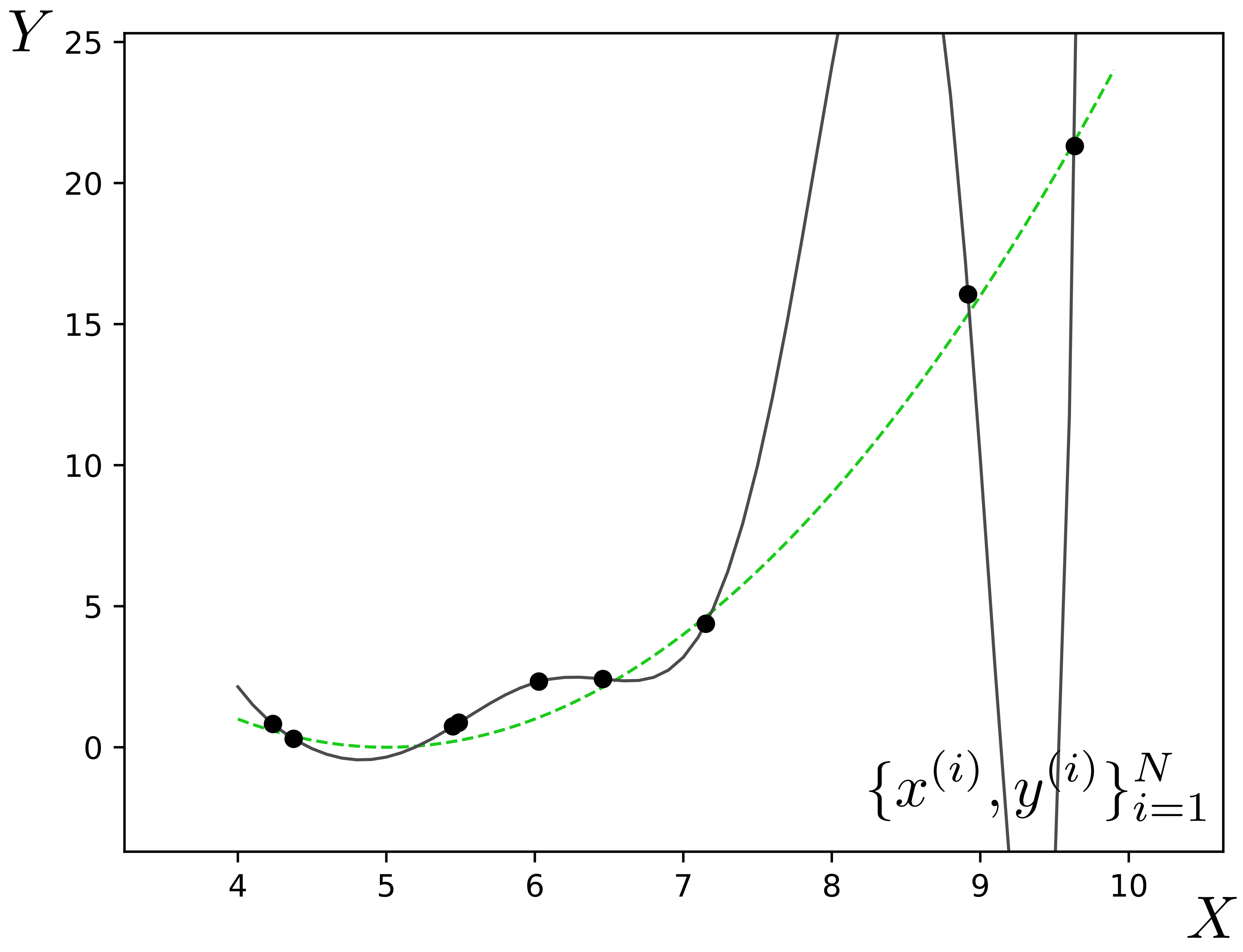
K = 9



$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

What happens as we add more basis functions?

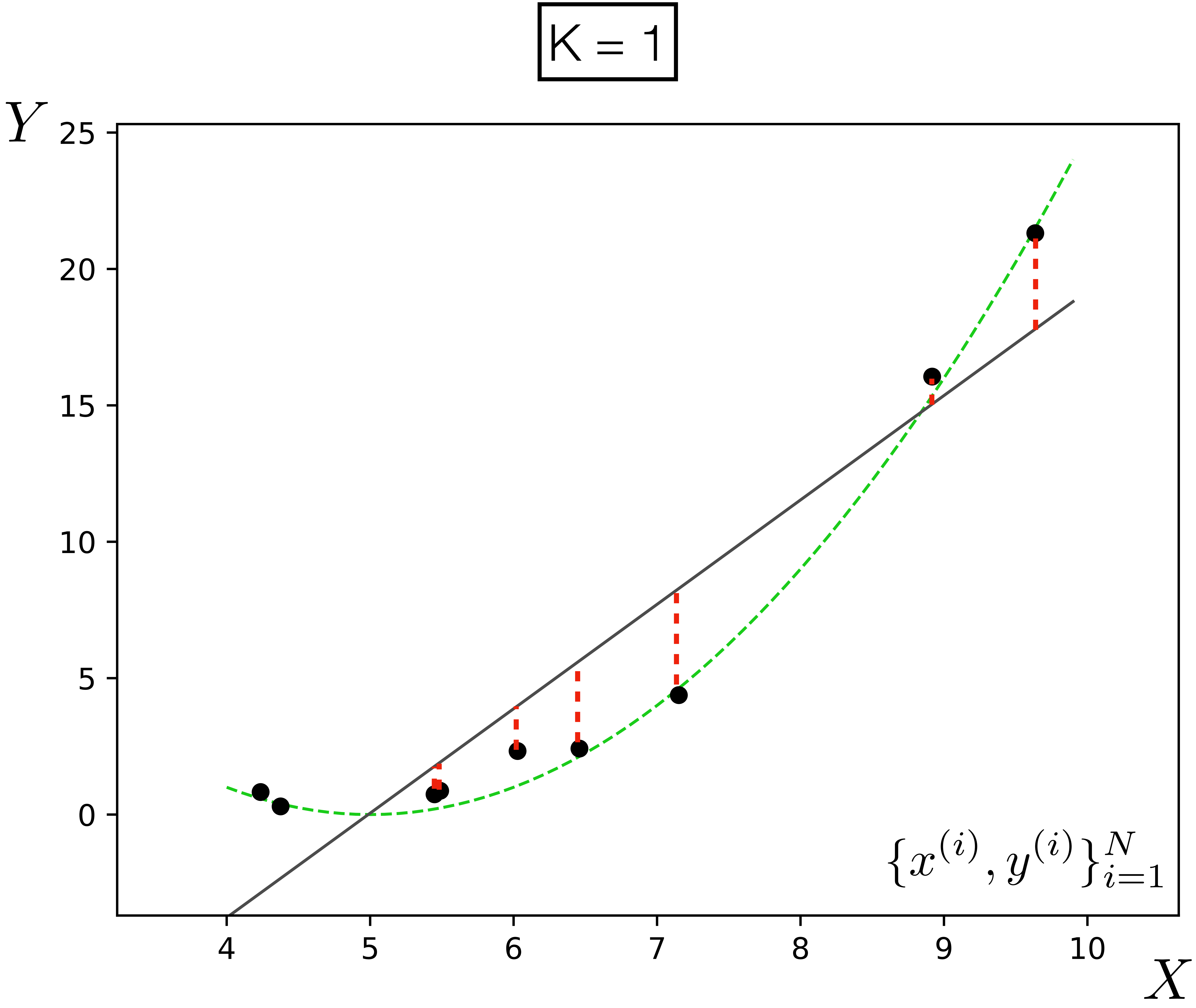
K = 10



$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

This phenomenon is called **overfitting**.

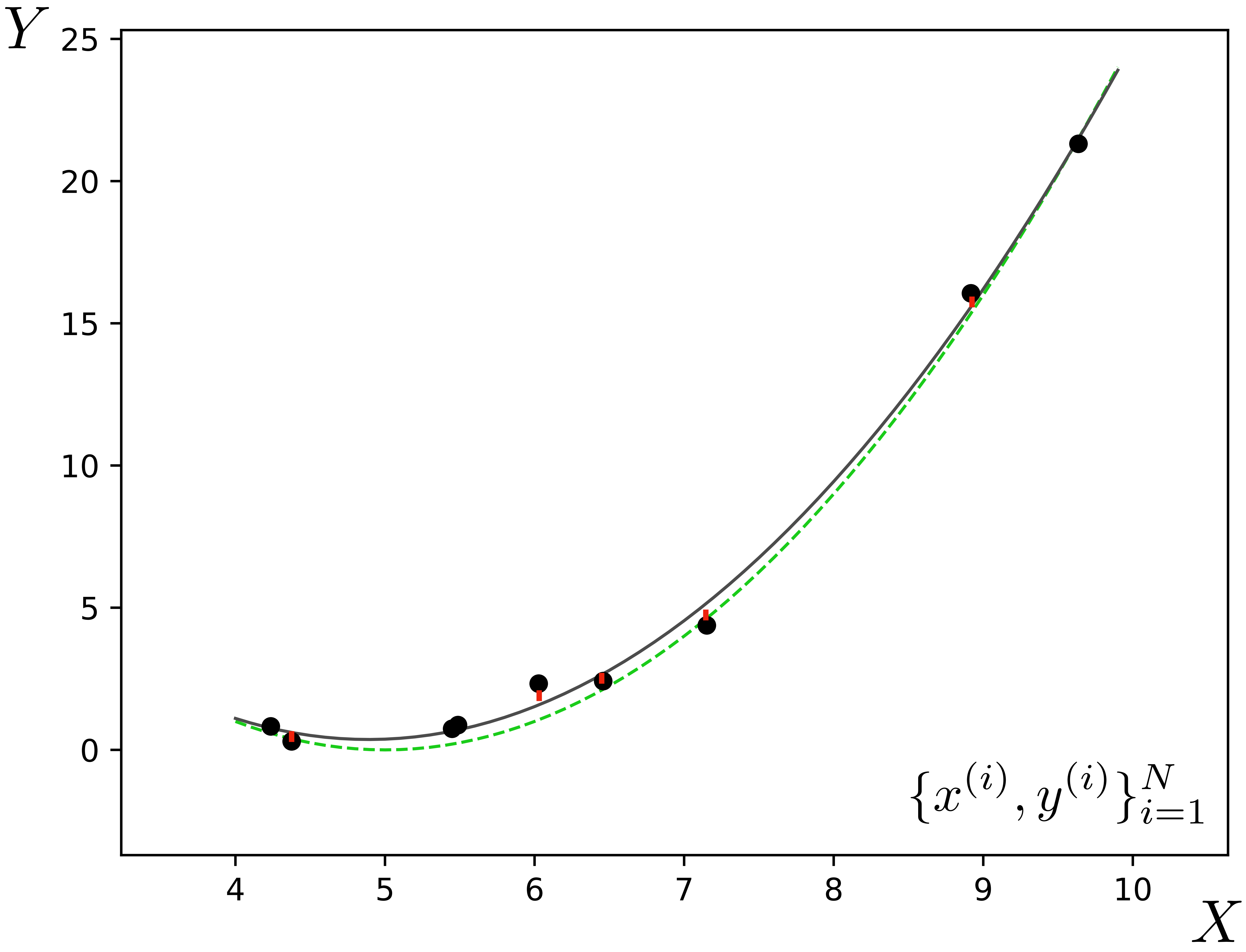
It occurs when we have too high **capacity** a model, e.g., too many free parameters, too few data points to pin these parameters down.



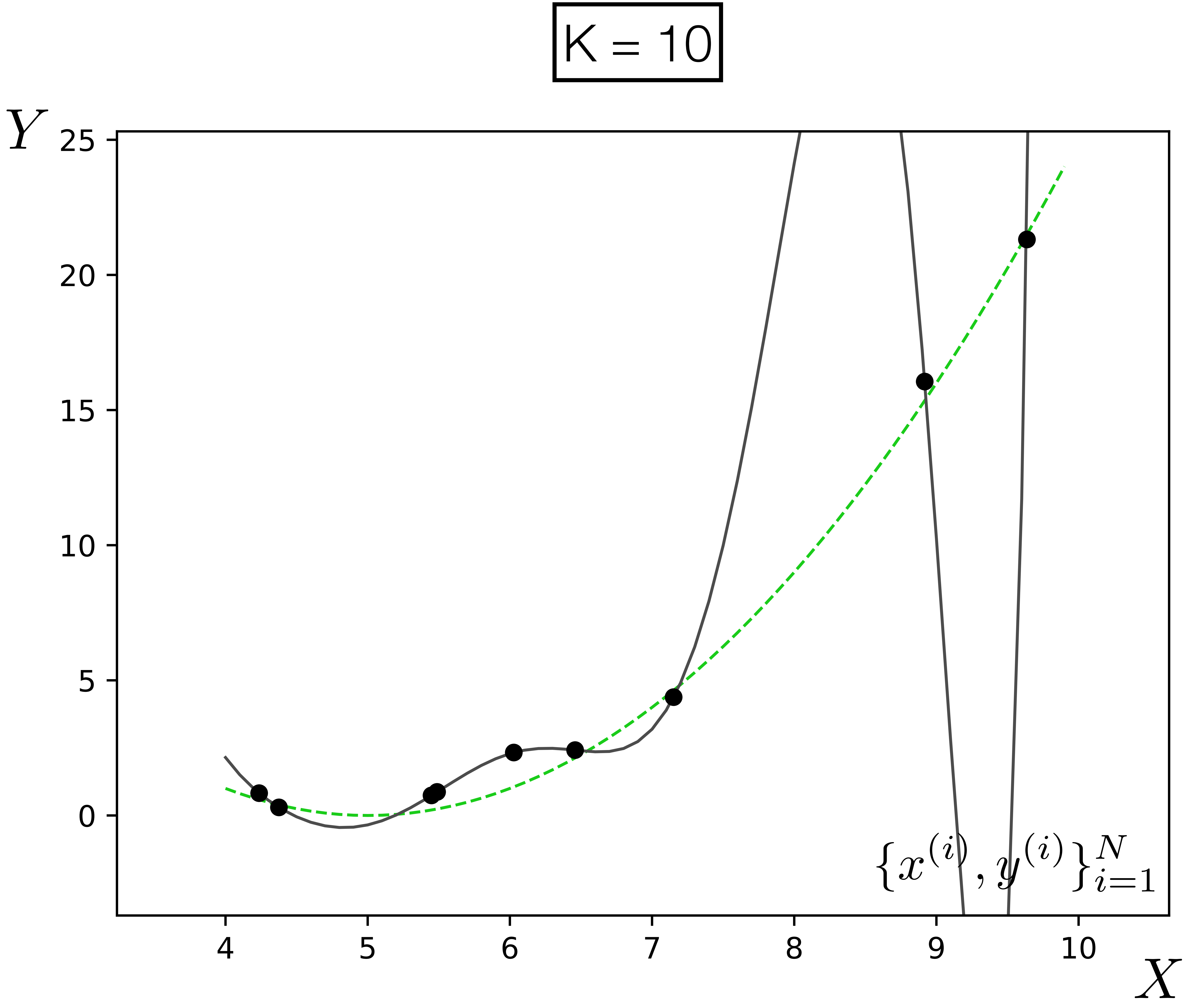
When the model does not have the capacity to capture the true function, we call this **underfitting**.

An underfit model will have high error on the training points. This error is known as **approximation error**.

K = 2



The true function is a quadractic, so a quadractic model (K=2) fits quite well.

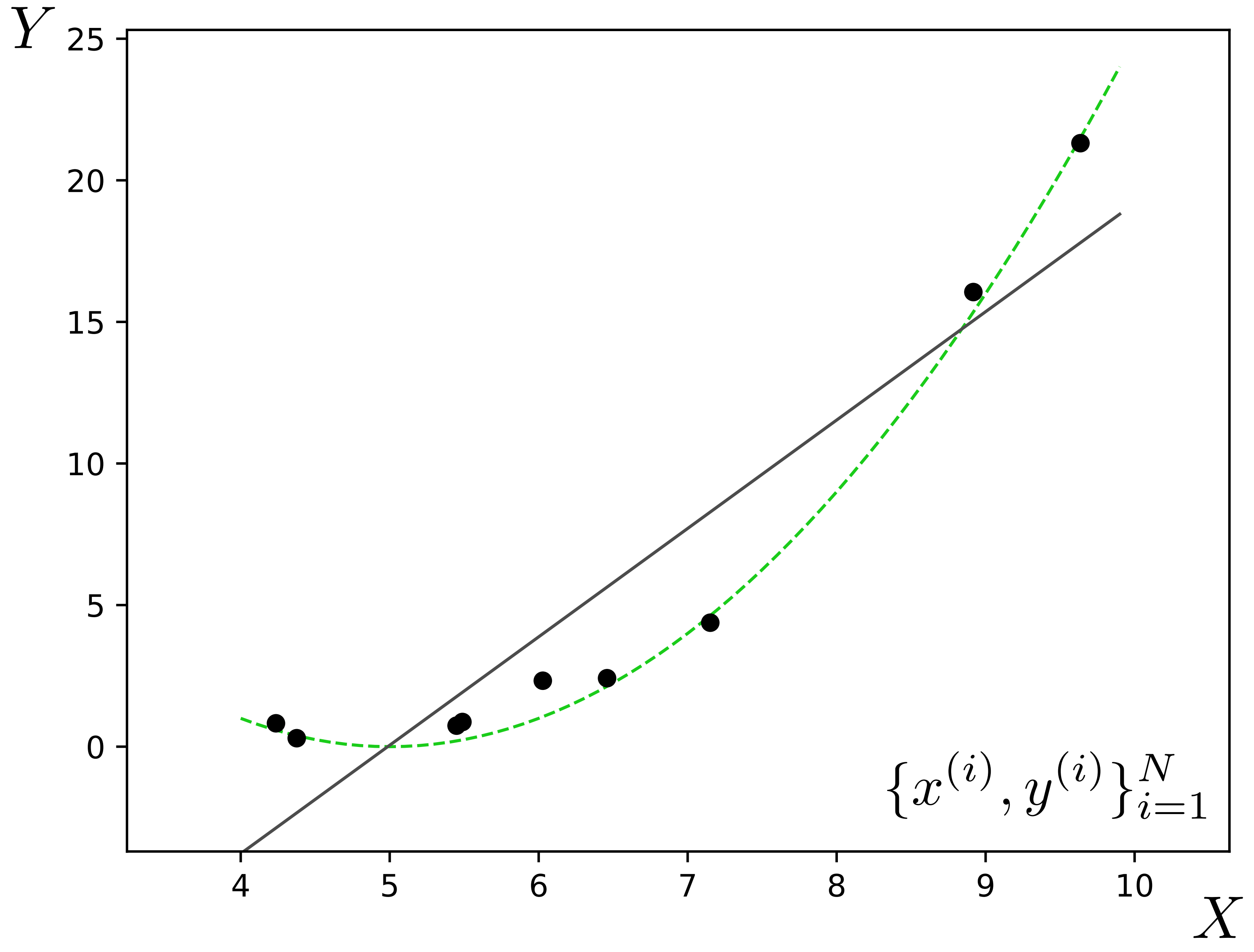


Now we have zero approximation error — the curve passes exactly through each training point.

But we have high **generalization error**, reflected in the gap between the true function and the fit line. We want to do well on *novel* queries, which will be sampled from the green curve (plus noise).

Underfitting

K = 1

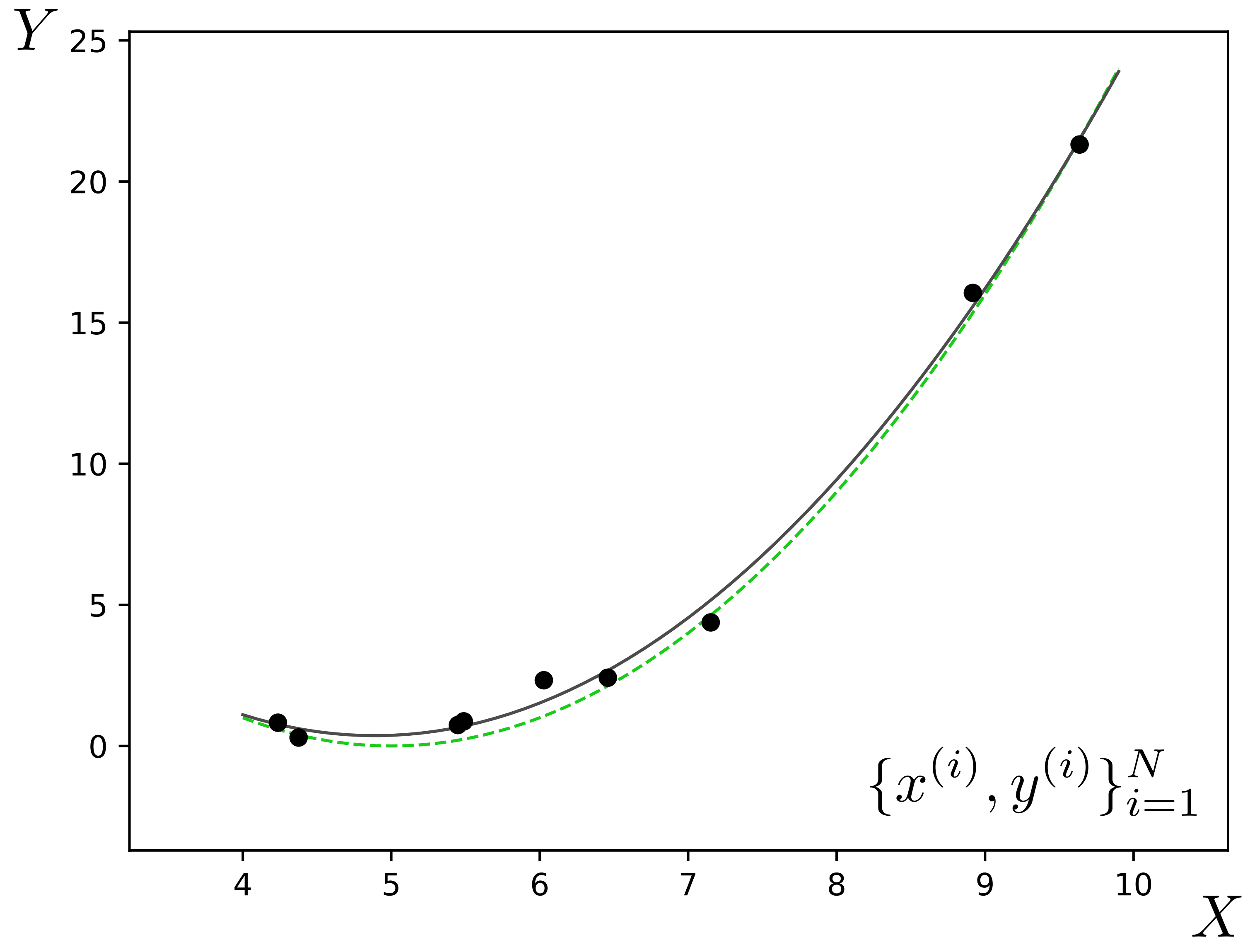


High error on train set

High error on test set

Appropriate model

K = 2

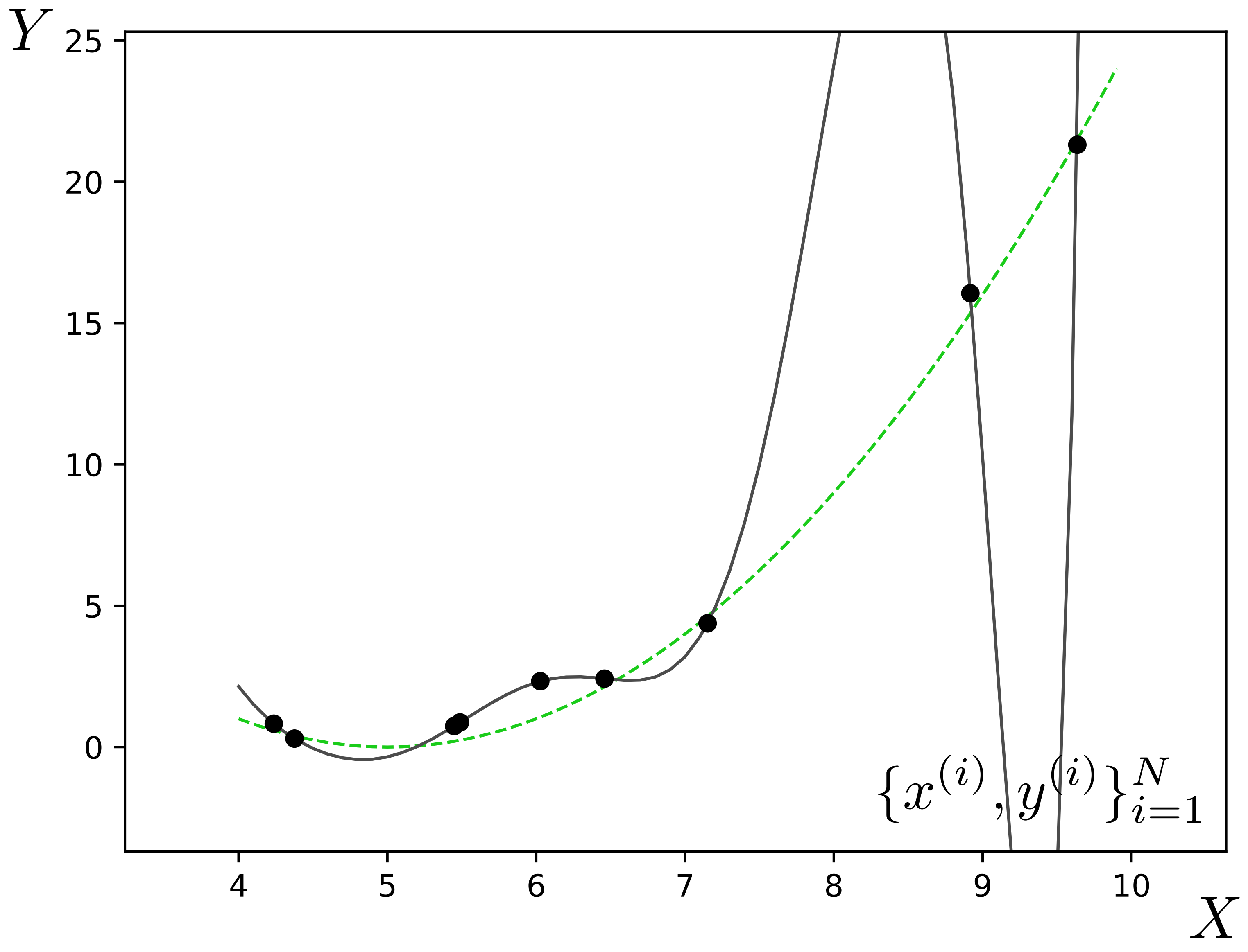


Low error on train set

Low error on test set

Overfitting

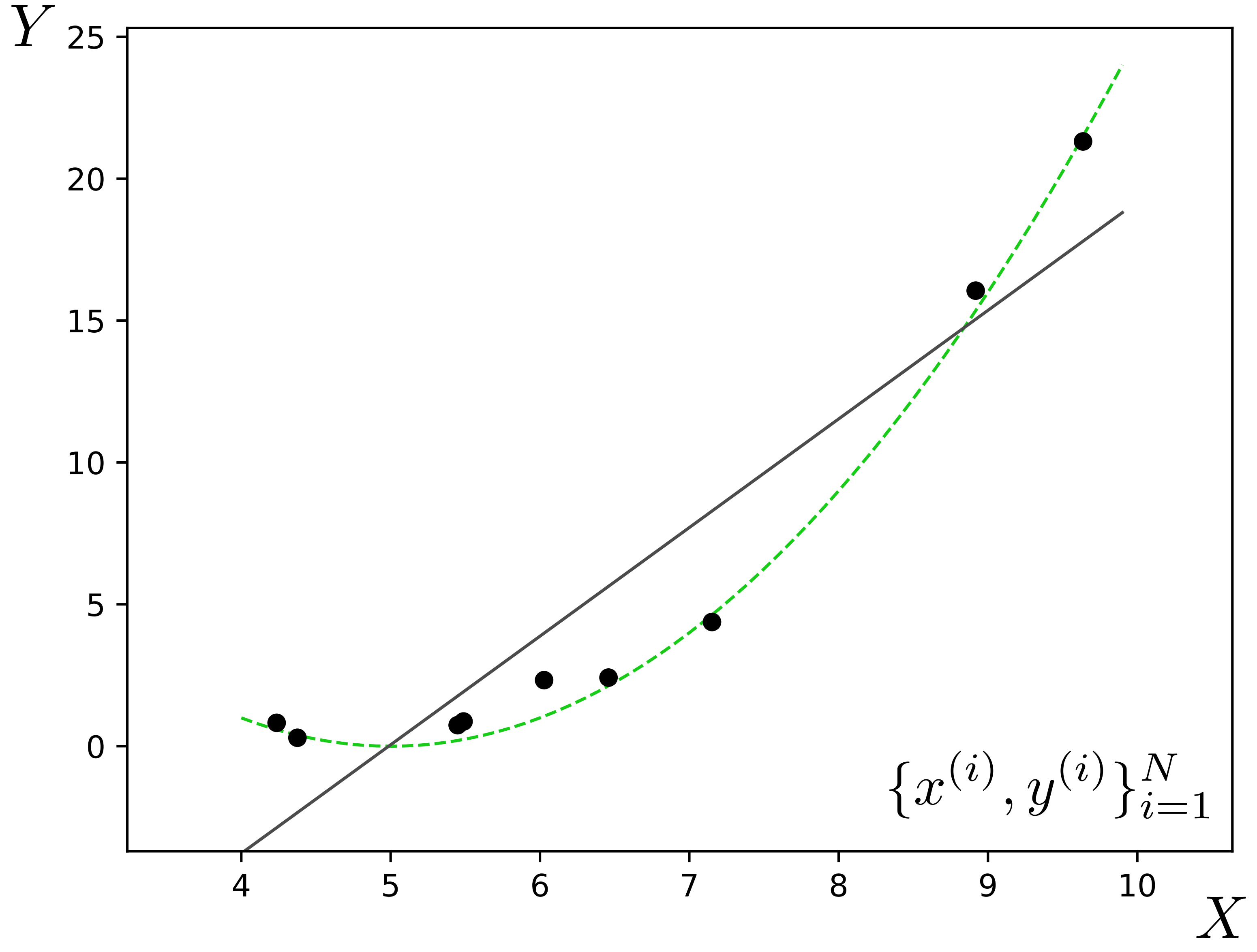
K = 10

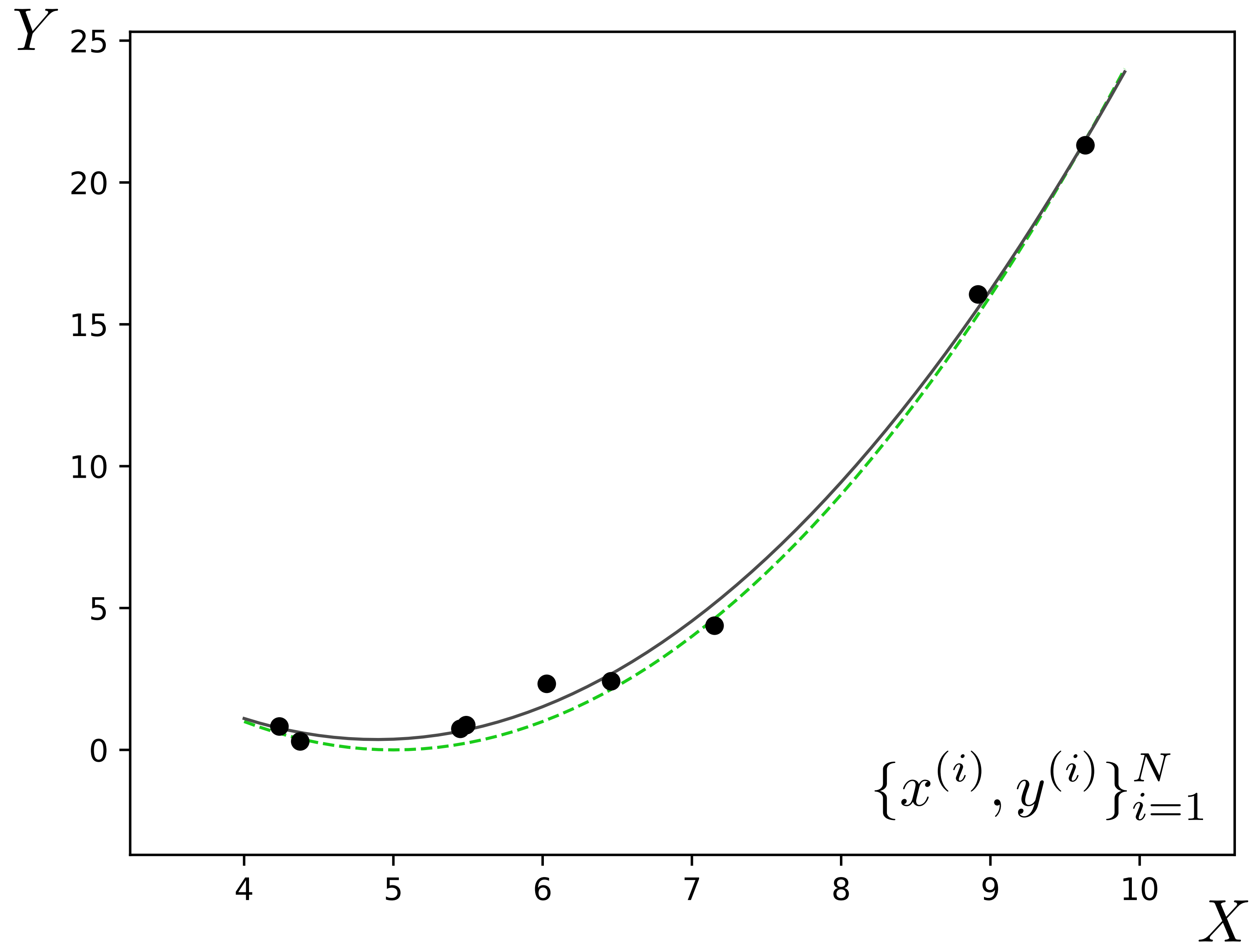


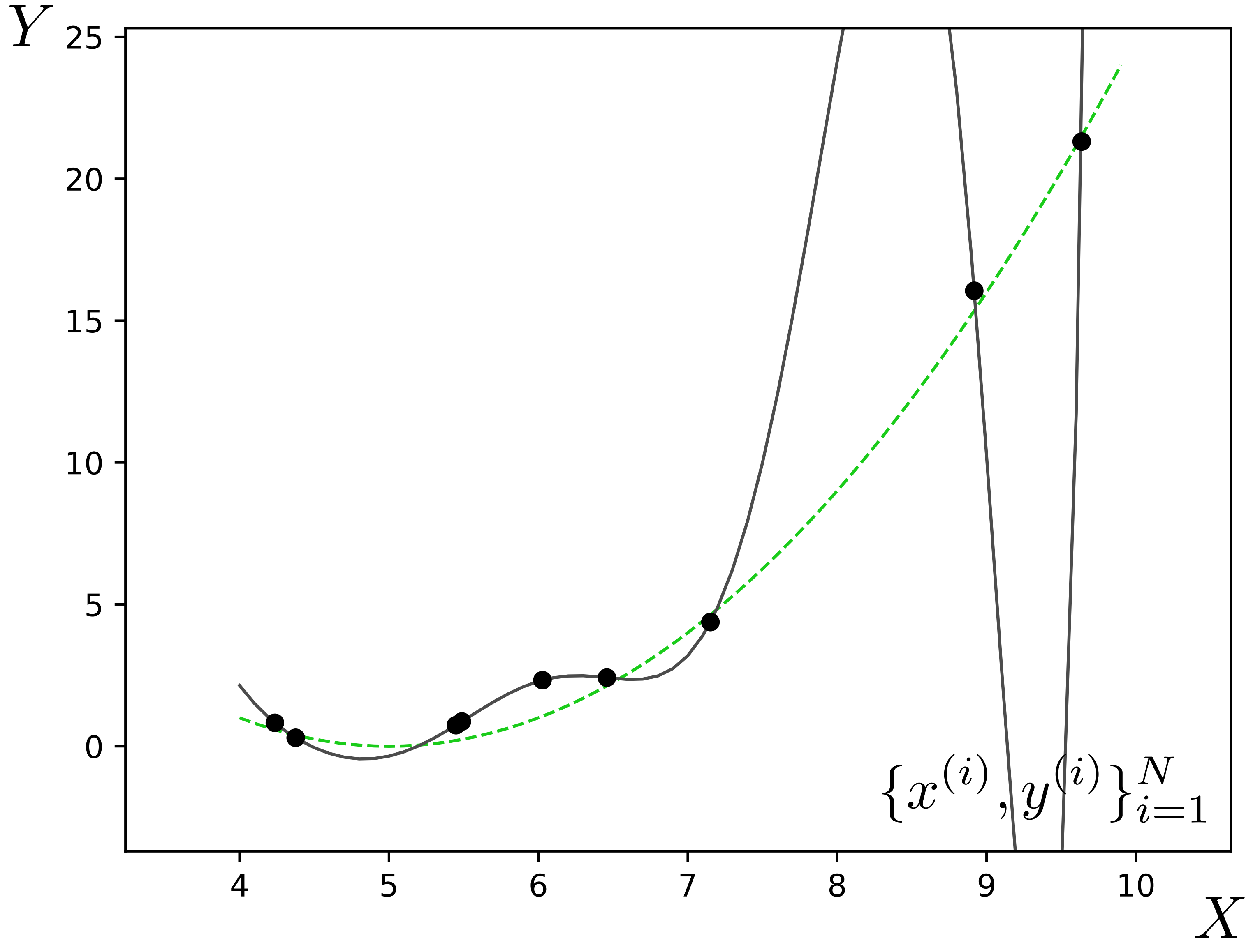
Lowest error on train set

High error on test set

Underfitting	Appropriate model	Overfitting
K = 1	K = 2	K = 10



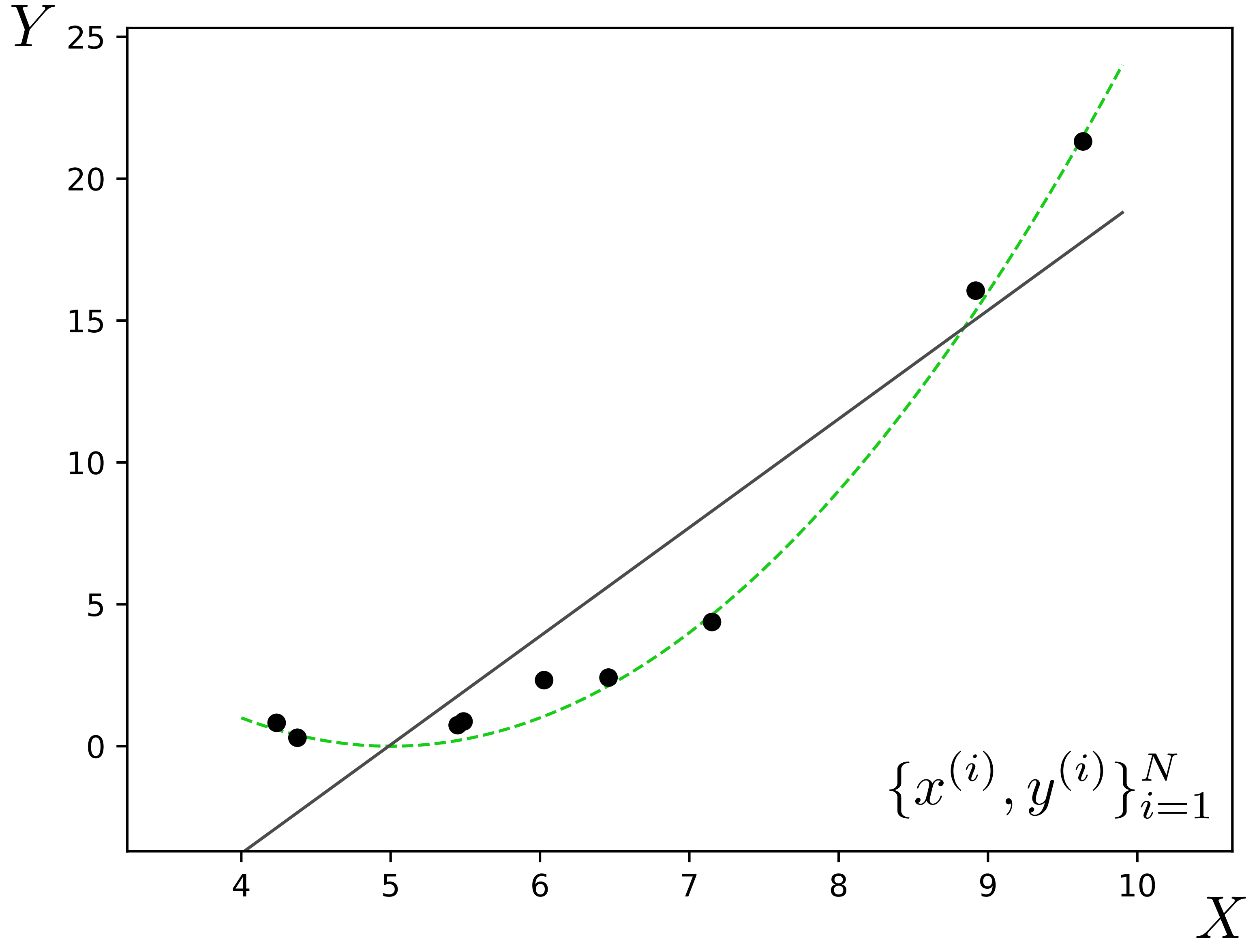




Simple model	Simple model	Complex model
Doesn't fit the training data	Fits the training data	Fits the training data

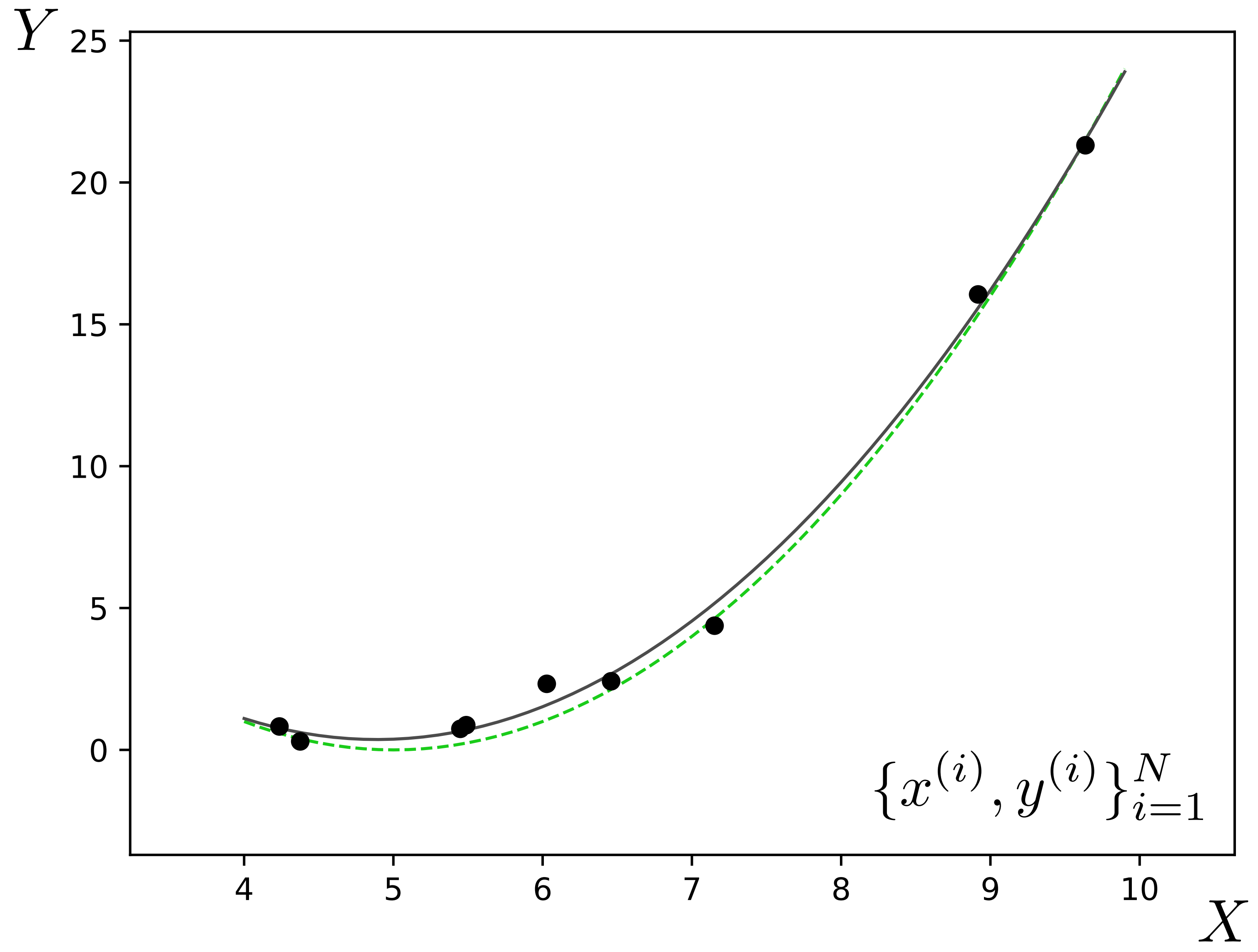
Underfitting

K = 1



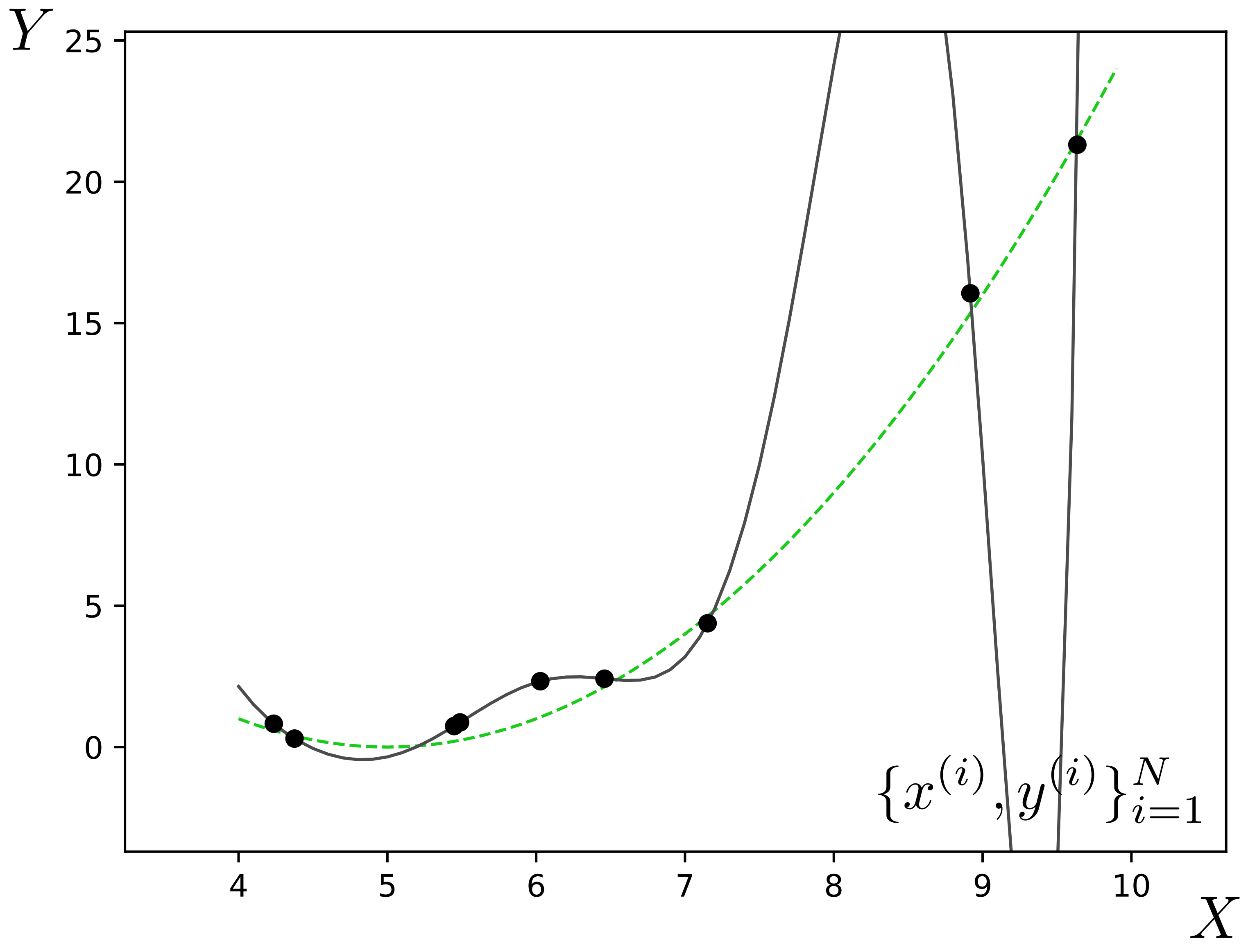
Appropriate model

K = 2



Overfitting

K = 10



We need to control the **capacity** of the model (e.g., use the appropriate number of free parameters).

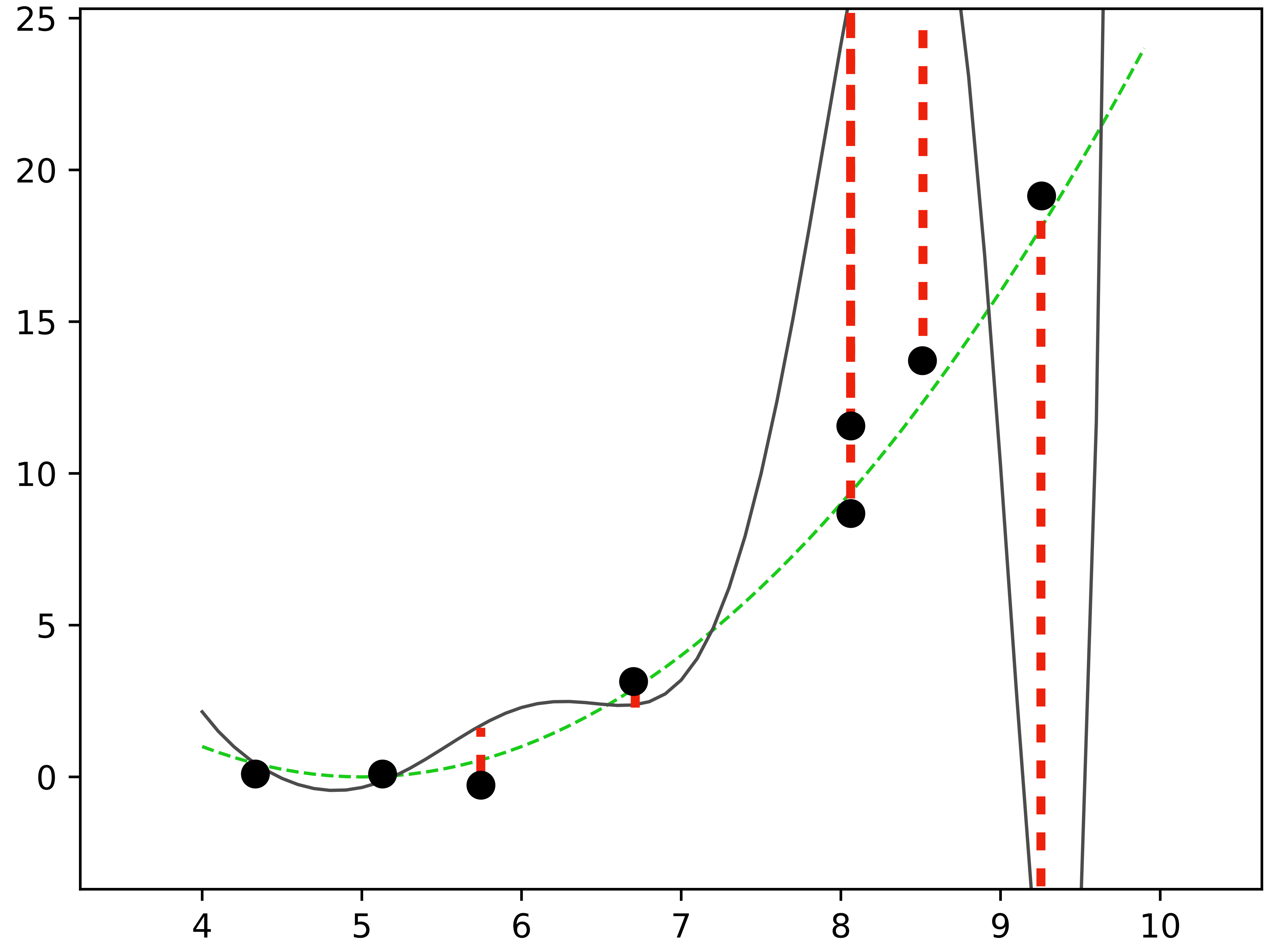
The capacity may be defined as the number of hypotheses under consideration in the hypothesis space.

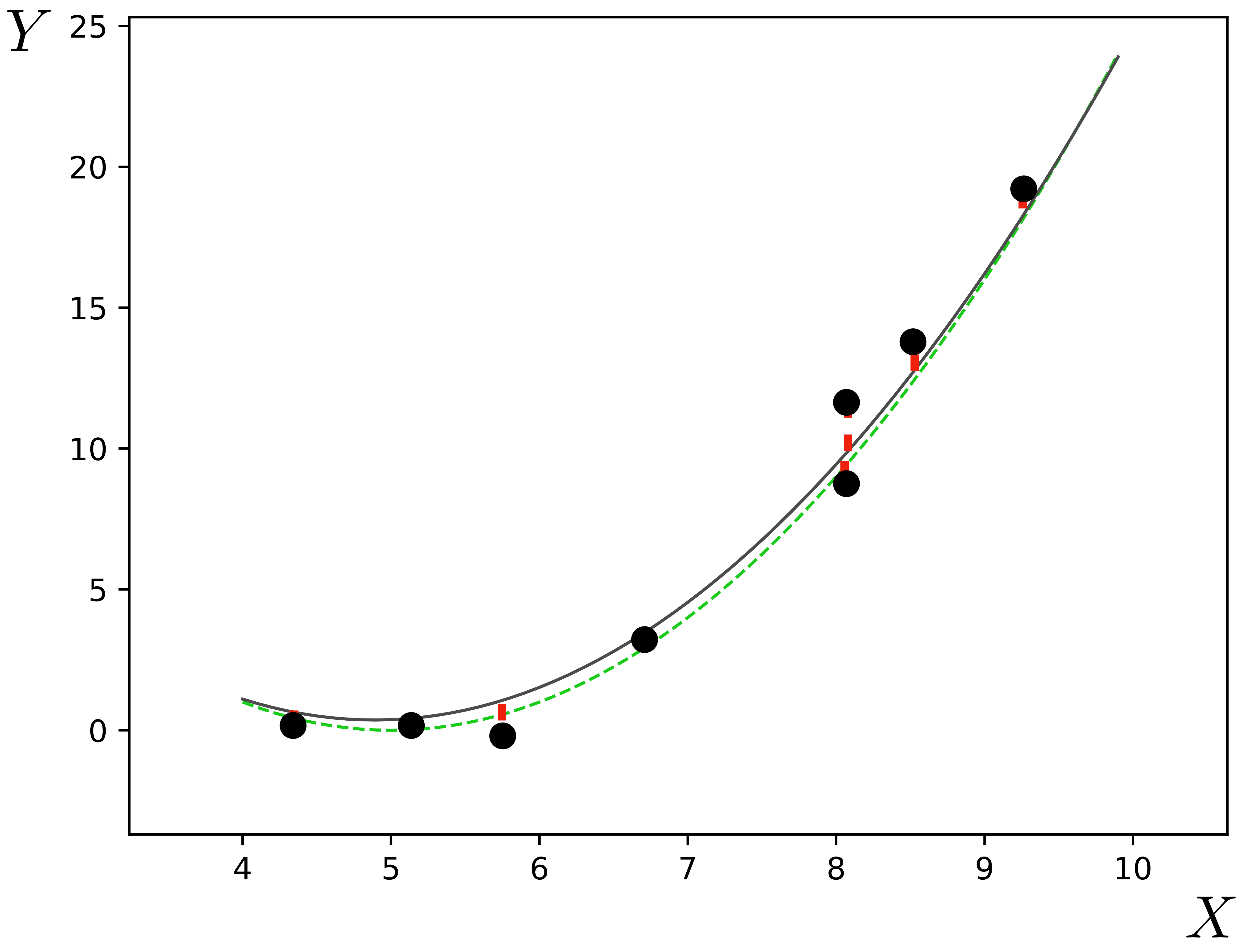
Complex models with many free parameters have high capacity.

Simple models have low capacity.

How do we know if we are underfitting or overfitting?

Validation data $\{x^{(i)}_{(\texttt{val})}, y^{(i)}_{(\texttt{val})}\}_{i=1}^{V}$





Cross validation: measure prediction error on validation data

Fitting just right

Underfitting?

1. add more parameters (more features, more layers, etc.)

Overfitting?

1. remove parameters
2. add **regularizers**

Selecting a *hypothesis space* of functions with just the right capacity is known as **model selection**

Regularization

Empirical risk minimization:

$$f^* = \arg\min_{\theta} \sum_{i=1}^{N} \mathcal{L}(f(\mathbf{x}^{(i)}), \mathbf{y}^{(i)}) + R(f)$$

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{N} \mathcal{L}(f_{\theta}(\mathbf{x}^{(i)}), \mathbf{y}^{(i)}) + R(\theta)$$

Regularized least squares

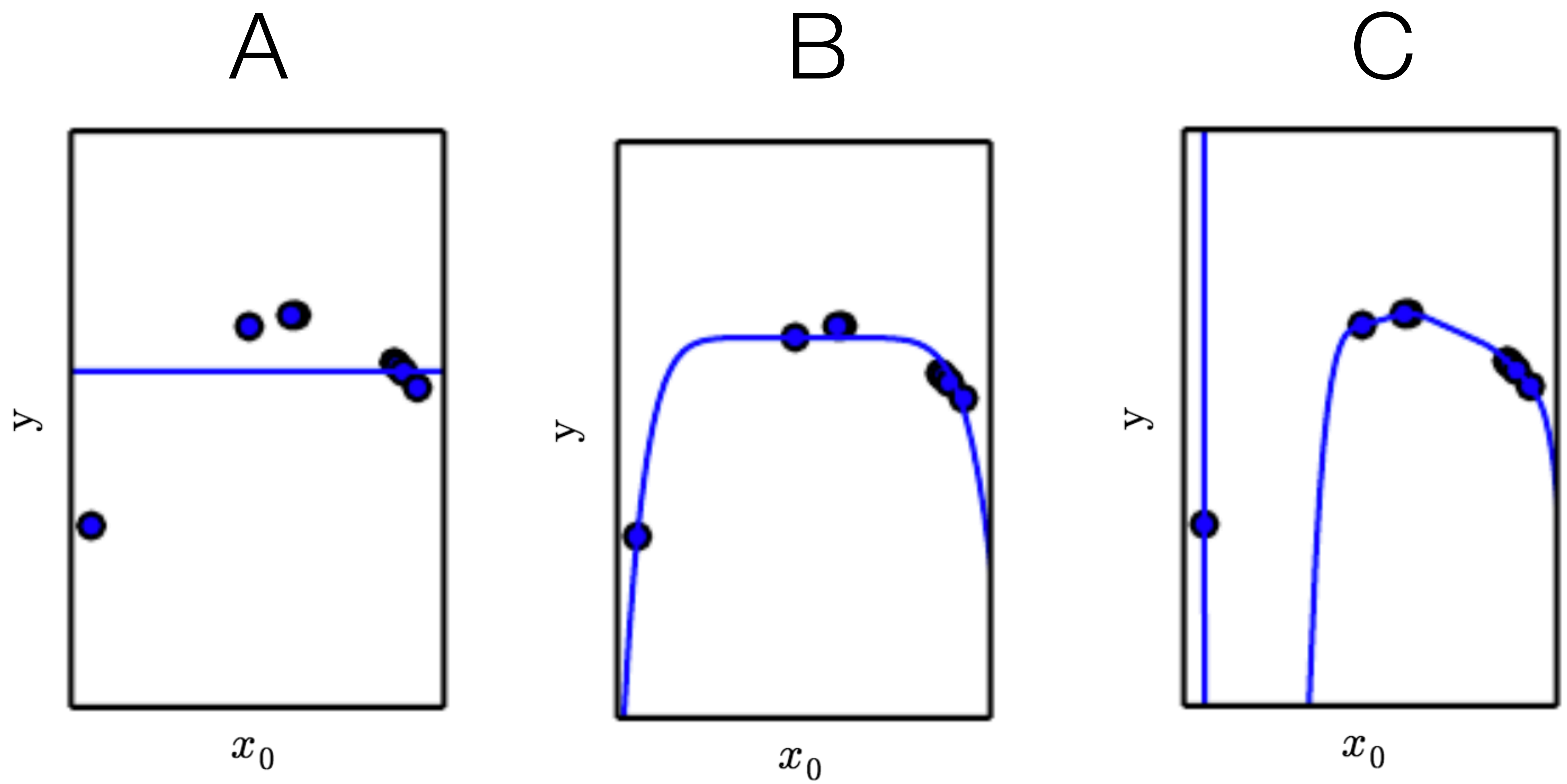
$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

$$R(\theta) = \lambda \left\|\theta\right\|_2^2$$ ← Only use polynomial terms if you really need them! Most terms should be zero

ridge regression, a.k.a., **Tikhonov regularization**

(Probabilistic interpretation: R is a Gaussian **prior** over values of the parameters.)

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{N} \mathcal{L}(f_\theta(\mathbf{x}^{(i)}), \mathbf{y}^{(i)}) + \lambda \|\theta\|_2^2$$



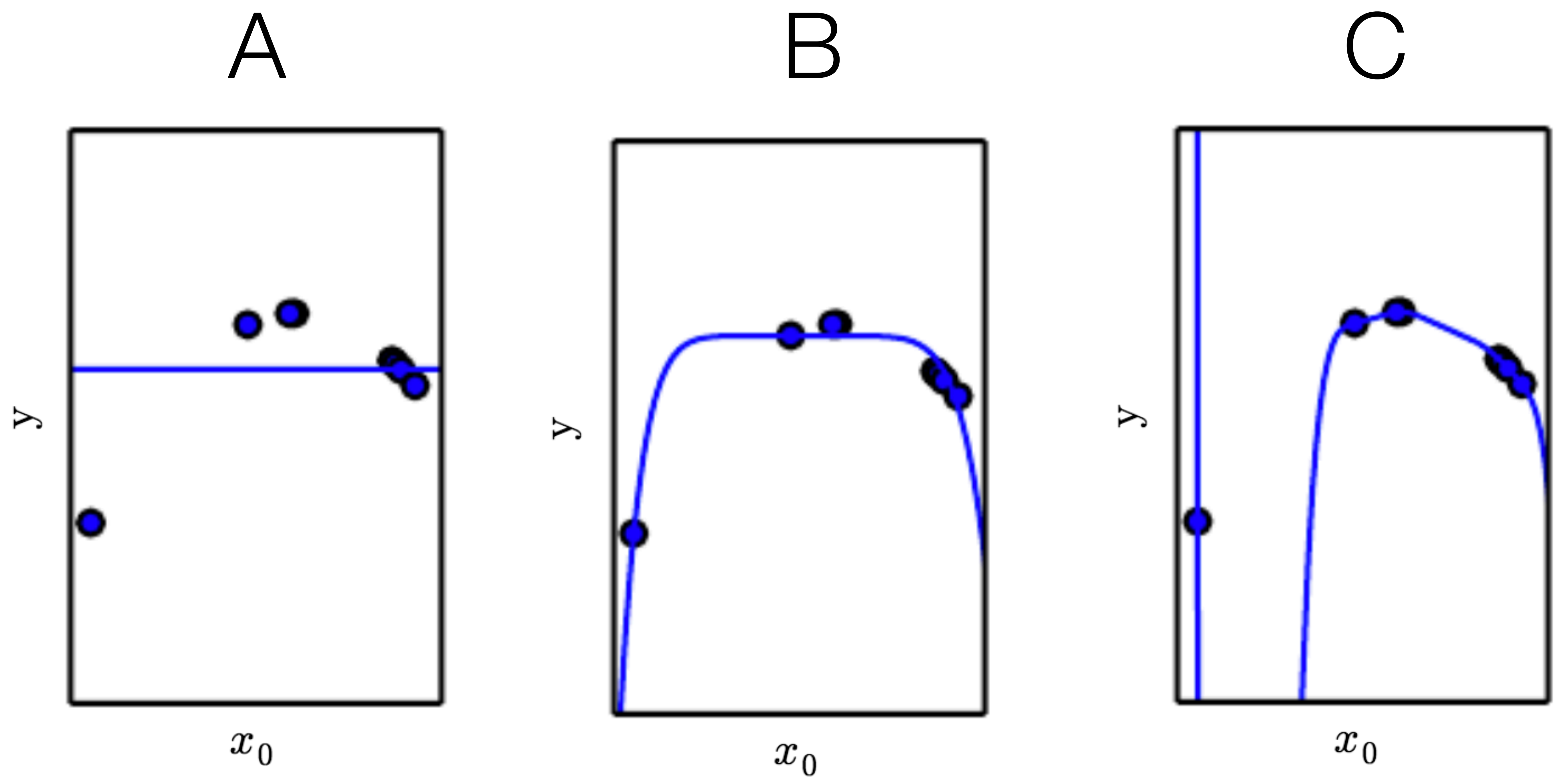
Low λ — ?

Medium λ — ?

High λ — ?

[Adapted from “Deep Learning”, Goodfellow et al.]

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{N} \mathcal{L}(f_\theta(\mathbf{x}^{(i)}), \mathbf{y}^{(i)}) + \lambda \|\theta\|_2^2$$



Low λ — C

Medium λ — B

High λ — A

[Adapted from “Deep Learning”, Goodfellow et al.]

Regularized polynomial least squares regression

Data

$\{x^{(i)}, y^{(i)}\}_{i=1}^{N} \rightarrow$

Learner

Objective

$$\sum_{i=1}^{N}(f_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \left\|\theta\right\|_2^2$$

Hypothesis space

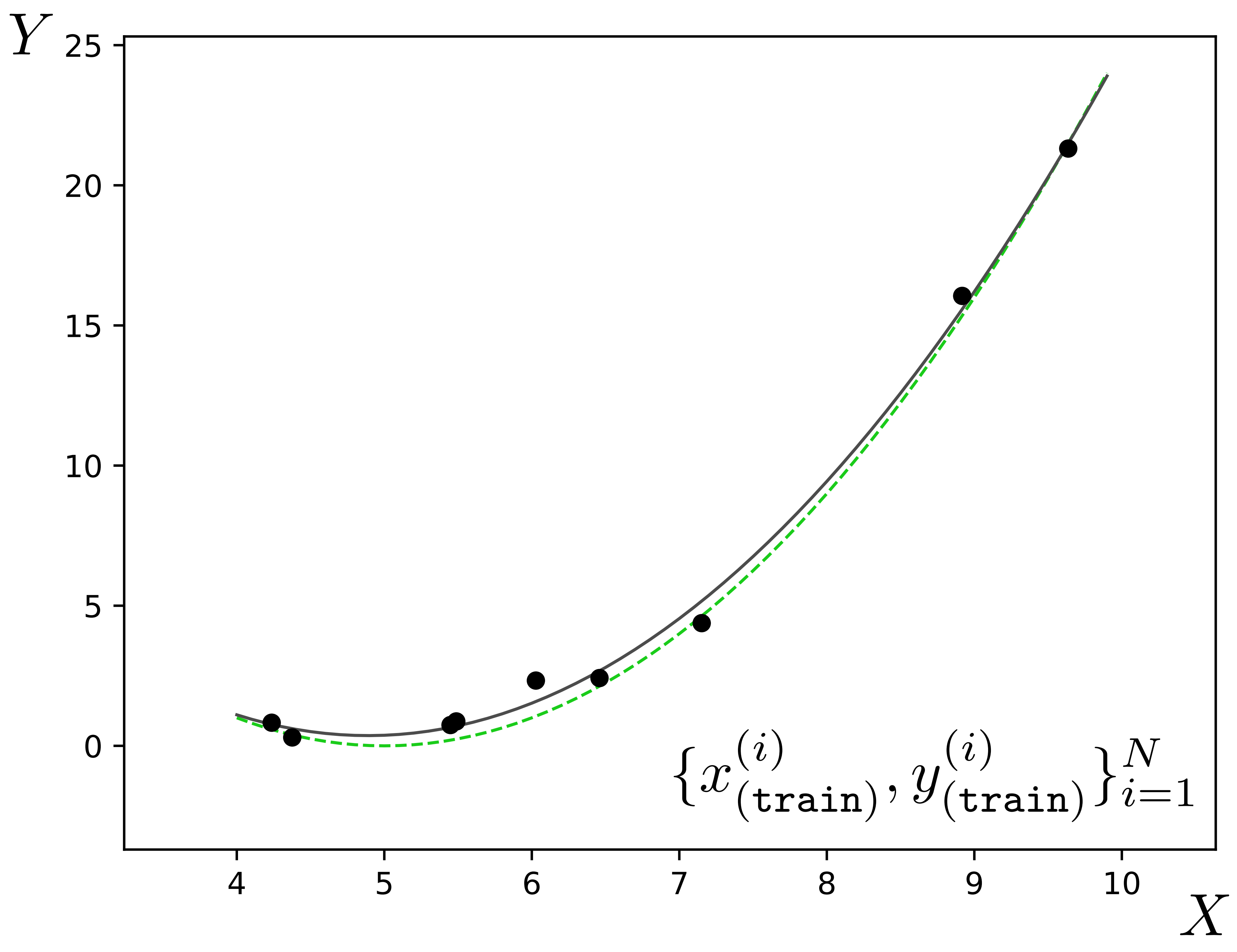
$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

Optimizer

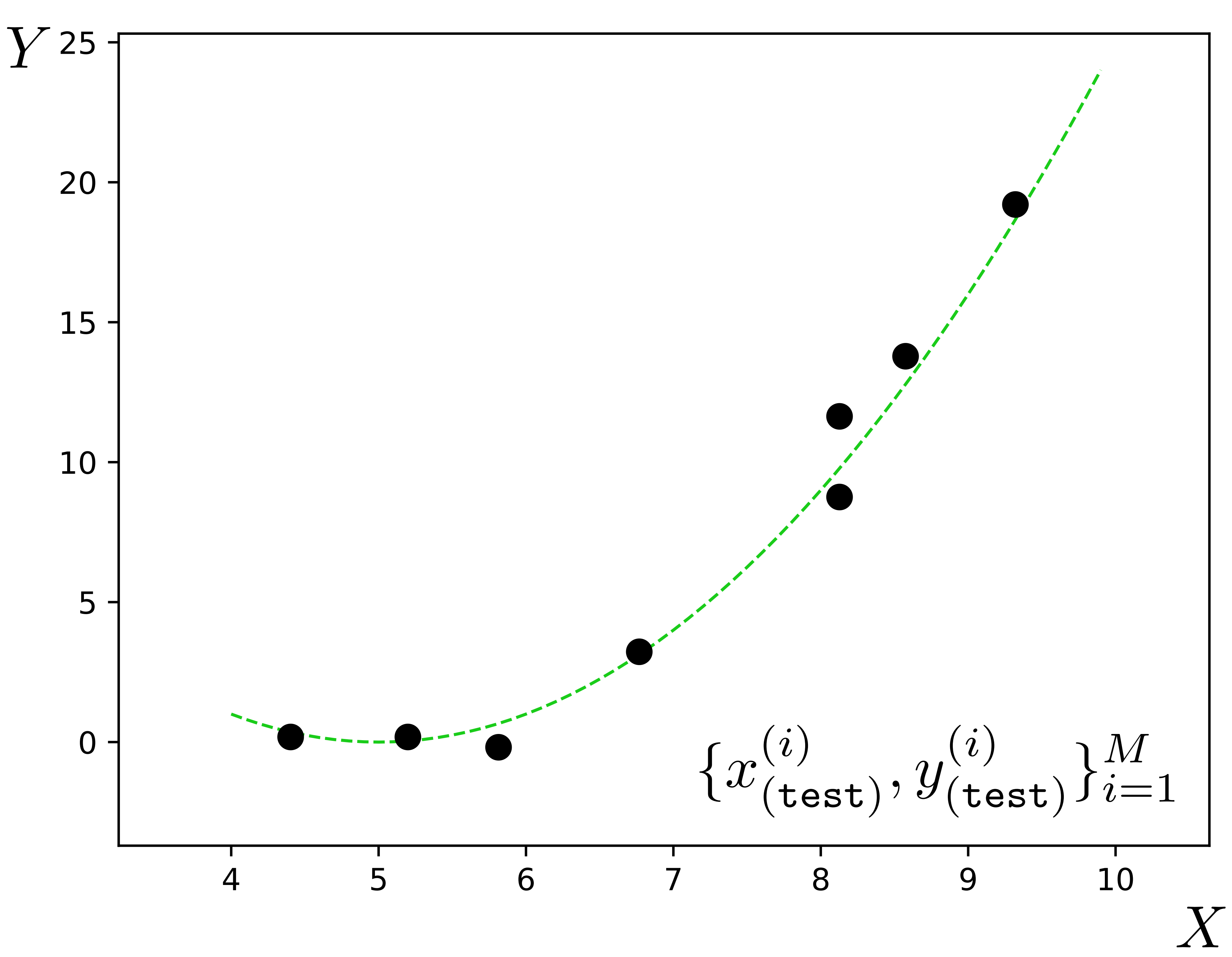
$$\theta^* = (\mathbf{\Phi}^T\mathbf{\Phi} + \lambda I)^{-1}\mathbf{\Phi}^T\mathbf{y}$$

$\rightarrow f$

Training data



Test data



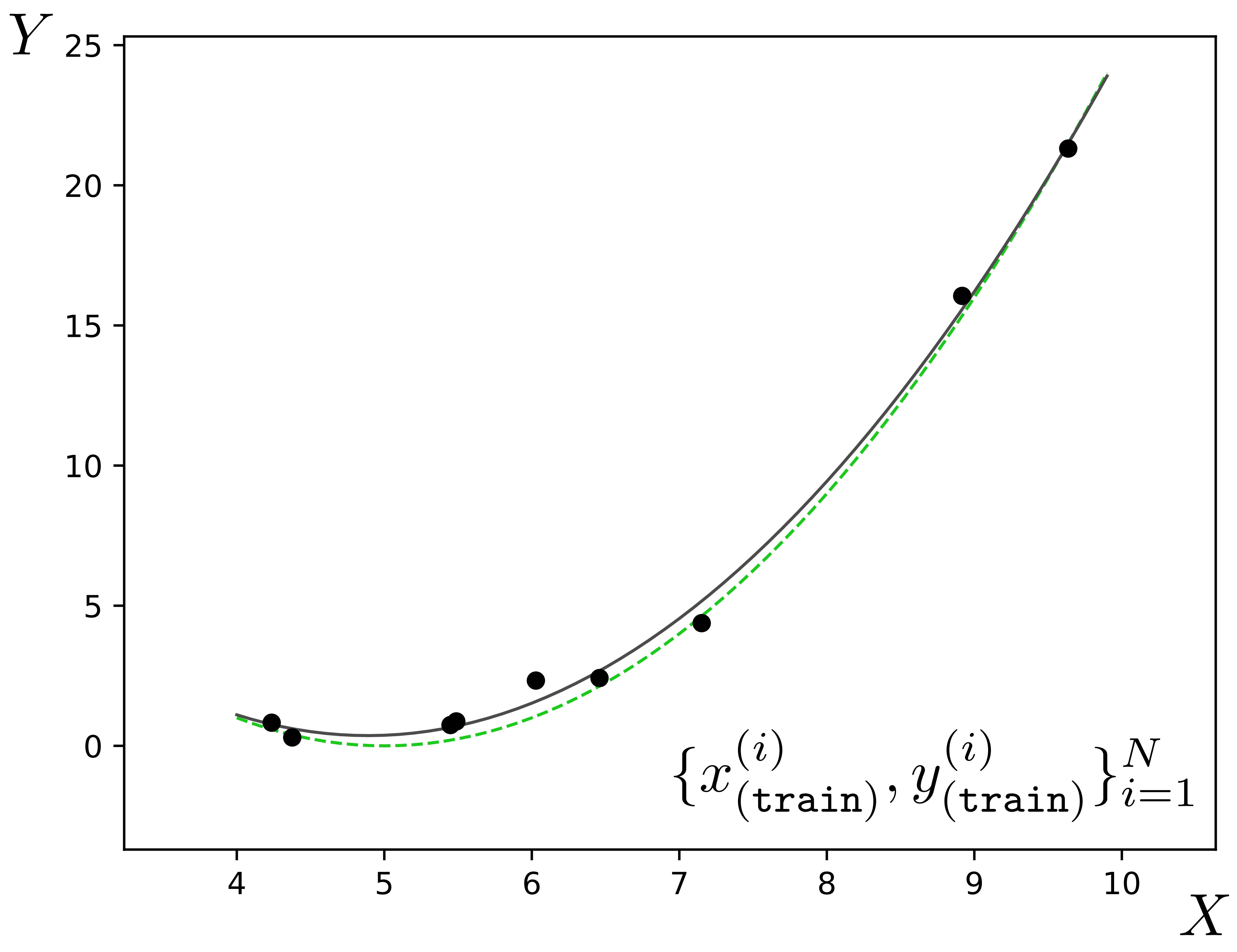
True **data-generating process**

$p_{\mathtt{data}}$

$$\{x^{(i)}_{(\mathtt{train})}, y^{(i)}_{(\mathtt{train})}\} \overset{\mathtt{iid}}{\sim} p_{\mathtt{data}}$$

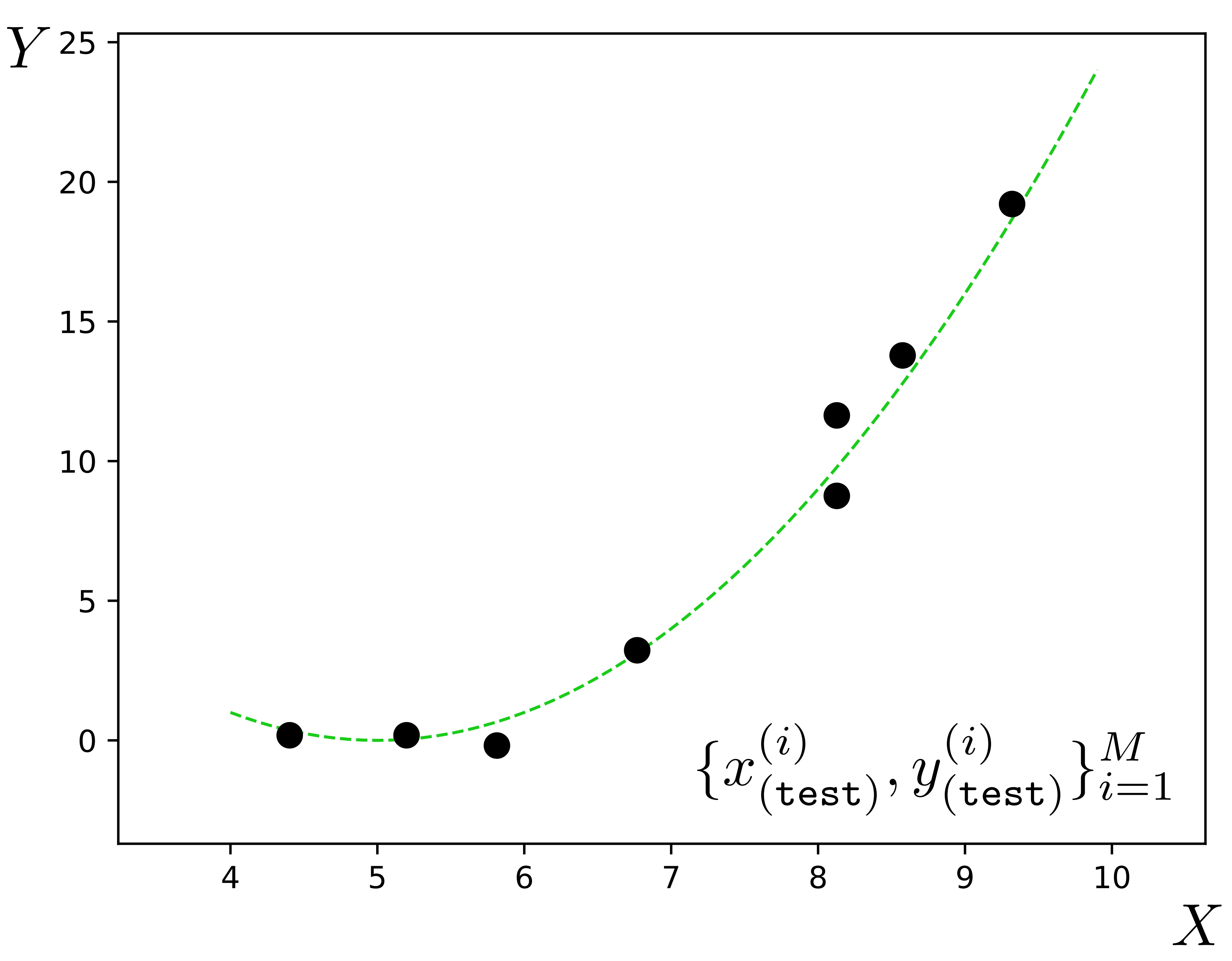
$$\{x^{(i)}_{(\mathtt{test})}, y^{(i)}_{(\mathtt{test})}\} \overset{\mathtt{iid}}{\sim} p_{\mathtt{data}}$$

Training data



This is a huge assumption!
Almost never true in practice!

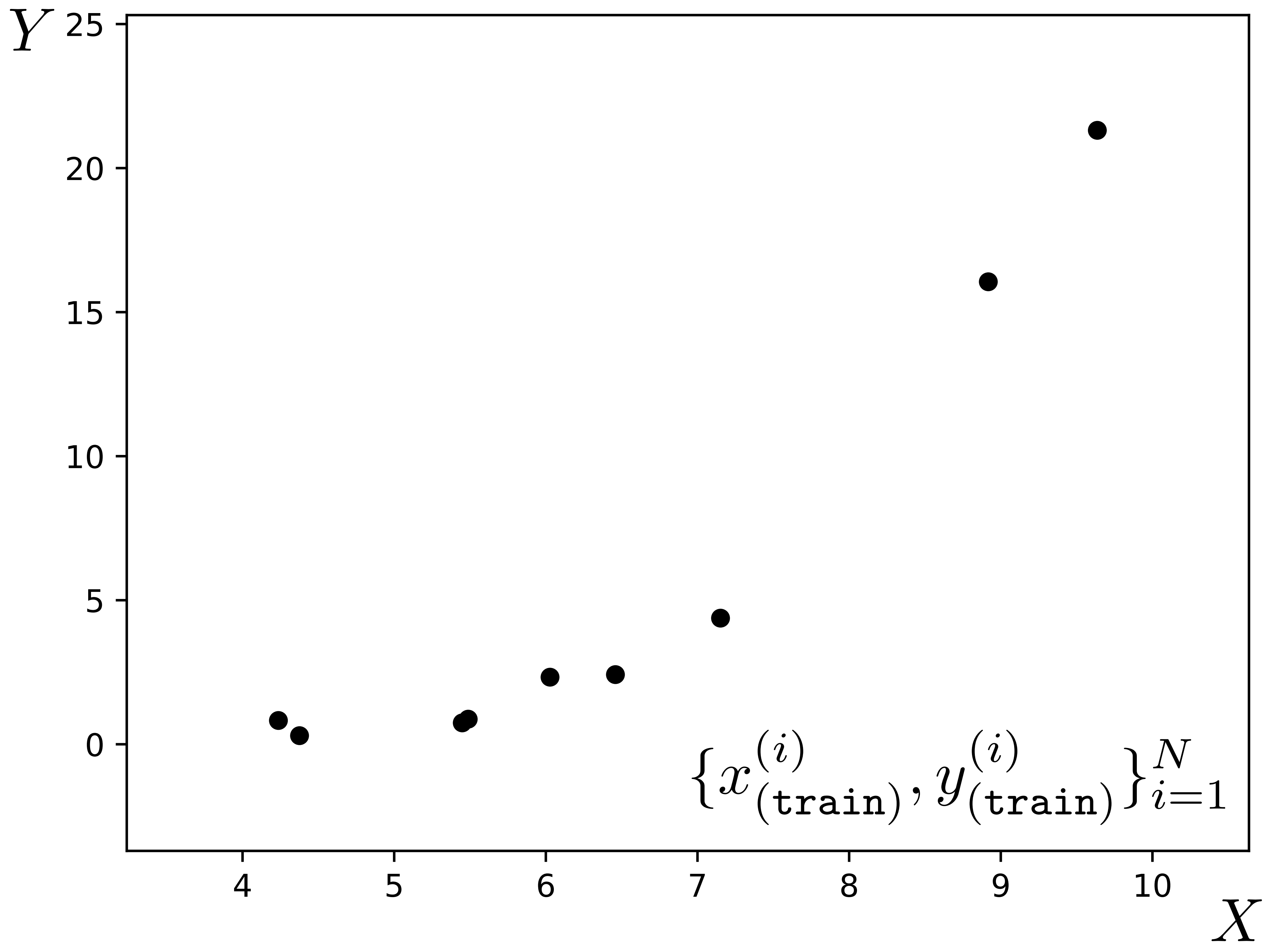
Test data



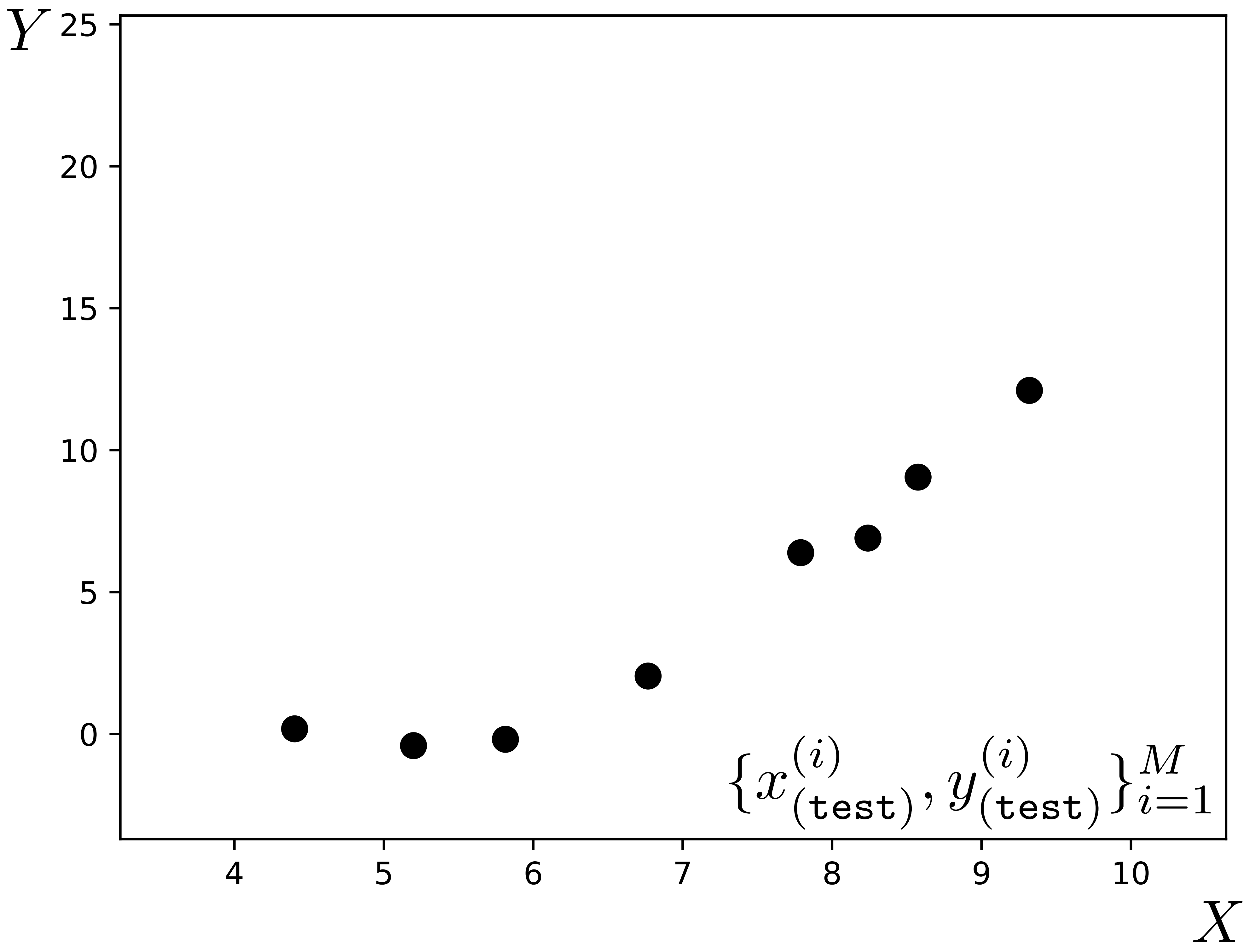
$$\{x^{(i)}_{(\mathtt{train})}, y^{(i)}_{(\mathtt{train})}\} \overset{\mathtt{iid}}{\sim} p_{\mathtt{data}}$$

$$\{x^{(i)}_{(\mathtt{test})}, y^{(i)}_{(\mathtt{test})}\} \overset{\mathtt{iid}}{\sim} p_{\mathtt{data}}$$

Training data



Test data

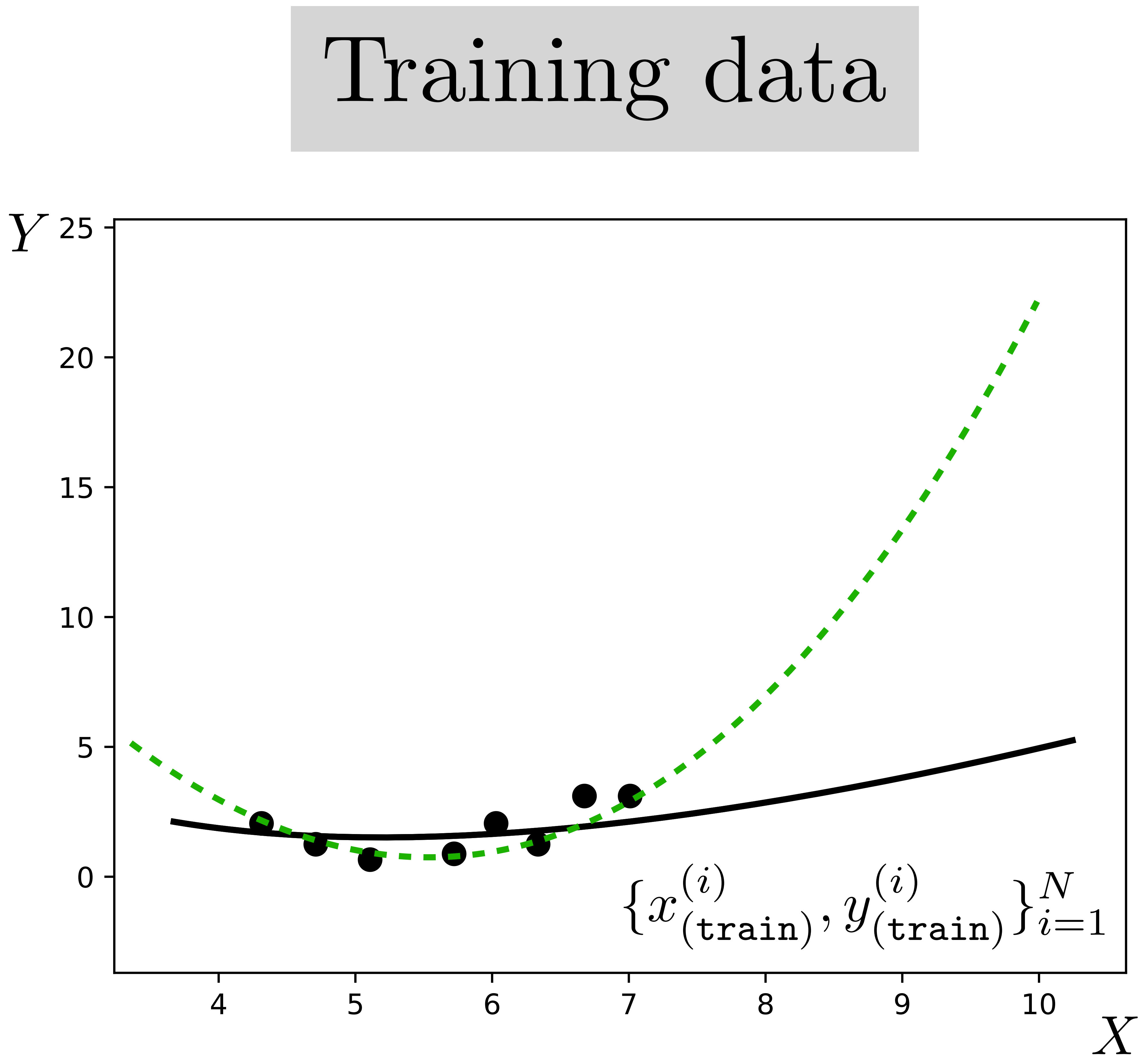


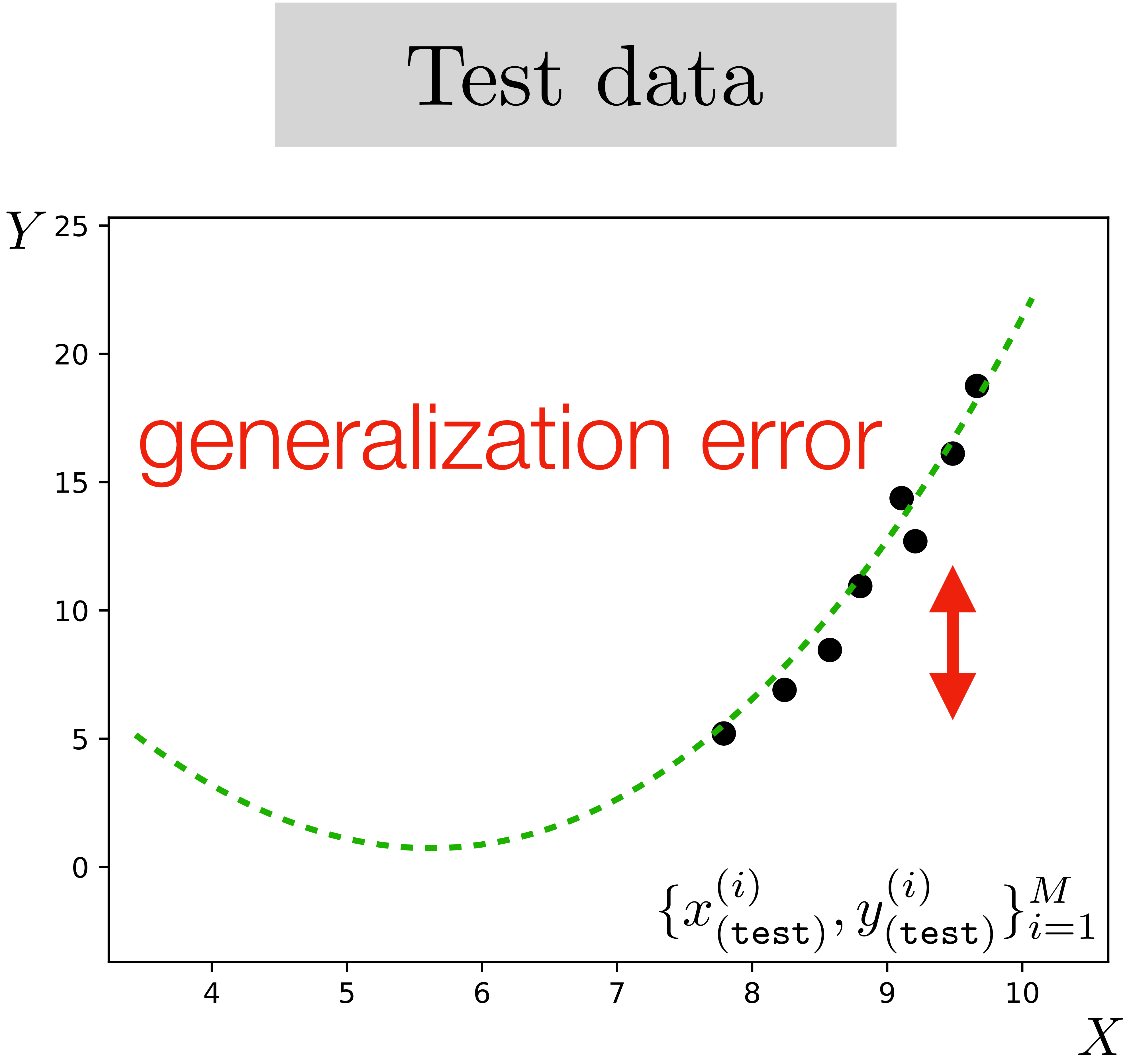
Much more commonly, we have

$$p_{\texttt{train}} \neq p_{\texttt{test}}$$

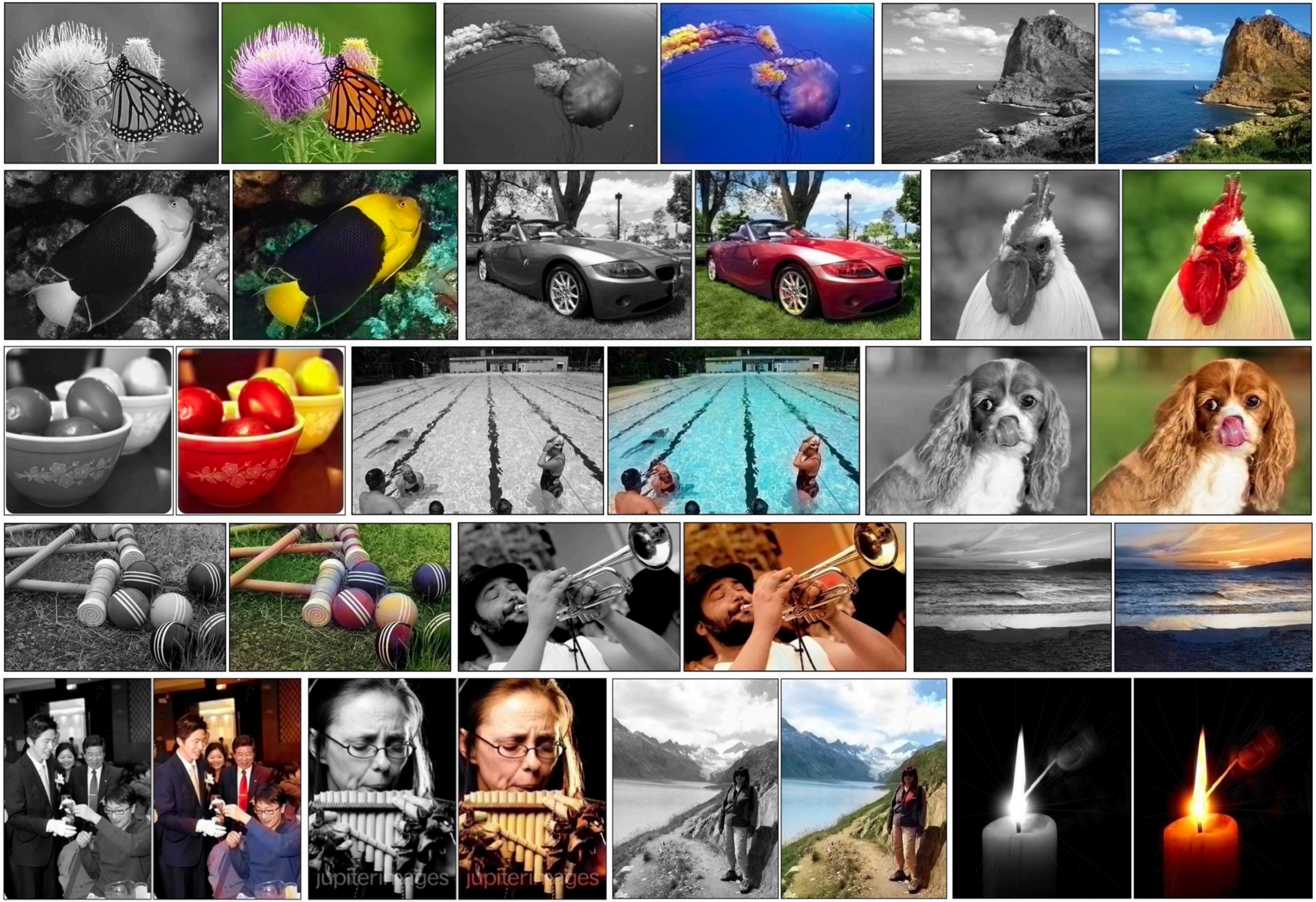
$$\{x^{(i)}_{(\texttt{train})}, y^{(i)}_{(\texttt{train})}\} \overset{\texttt{iid}}{\sim} p_{\texttt{train}}$$

$$\{x^{(i)}_{(\texttt{test})}, y^{(i)}_{(\texttt{test})}\} \overset{\texttt{iid}}{\sim} p_{\texttt{test}}$$



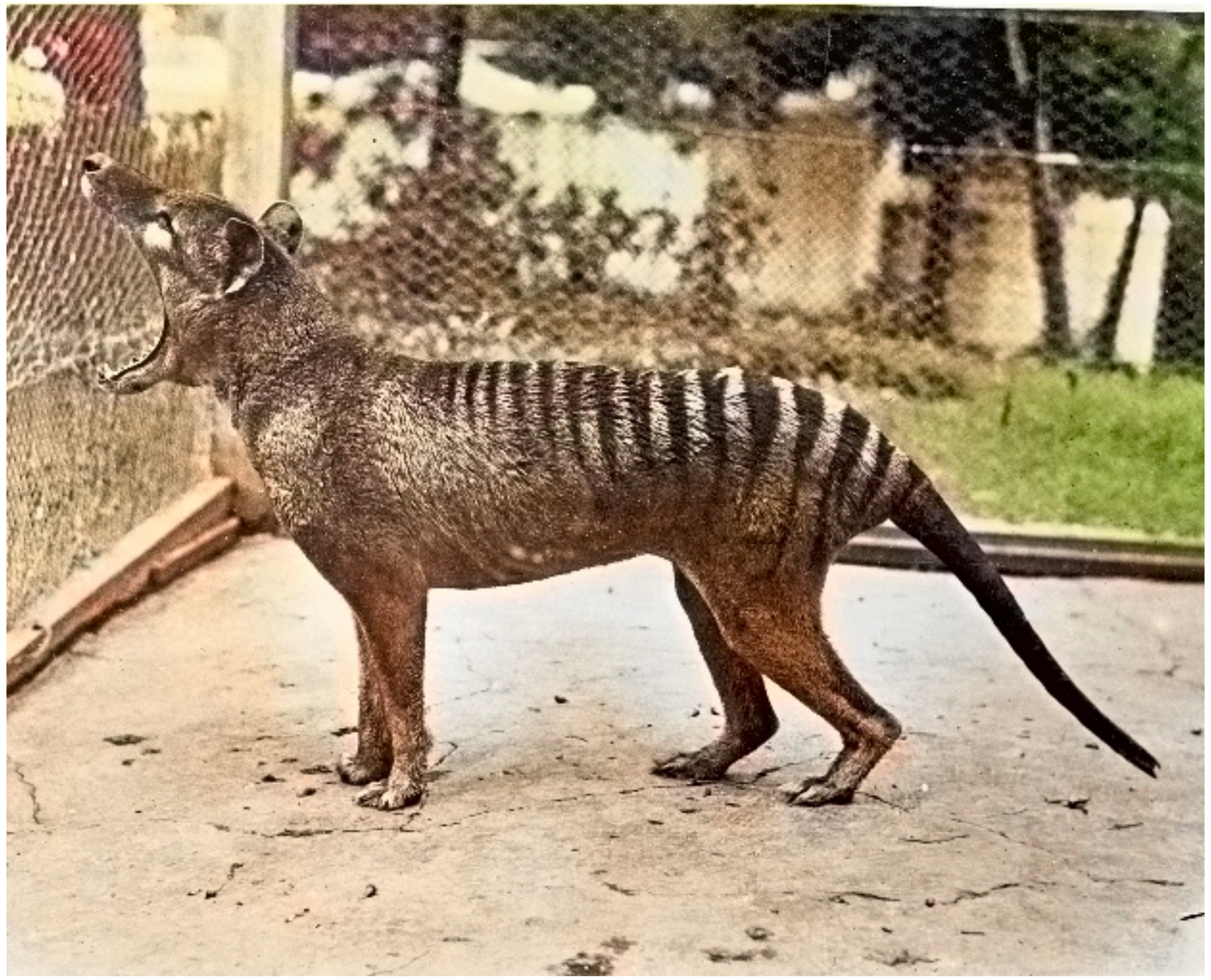


Our training data didn't cover the part of the distribution that was tested (**biased data**)



Zhang et al. 2016

u/Rafael_P_S



Thylacine

Chopin

6. Introduction to Machine Learning

- "It's all about the data!"
- Formalisms of learning *(Data, Compute, Objective Function, Hypothesis Space, Optimizer)*
- Case study #1: Linear least-squares
  - Empirical risk minimization
- Case study #2: Image classification
  - Softmax regression
- The problem of generalization