

Lecture 8

Introduction to Machine Learning

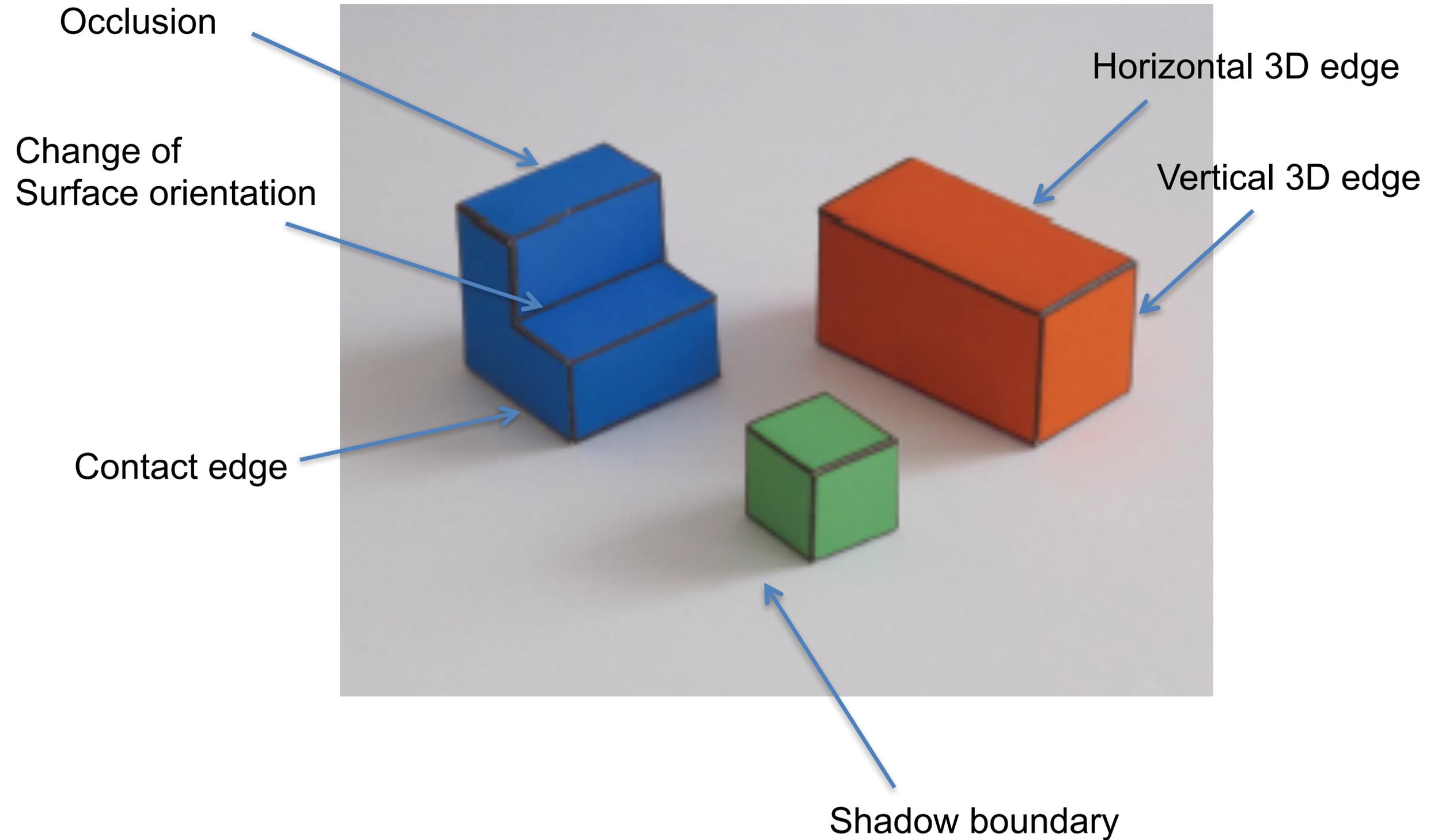
Announcements

- Pset 3 due next Tuesday (in 1 week)
- Pytorch tutorials Tues 4-6pm and Thurs 4-6pm

8. Introduction to Machine Learning

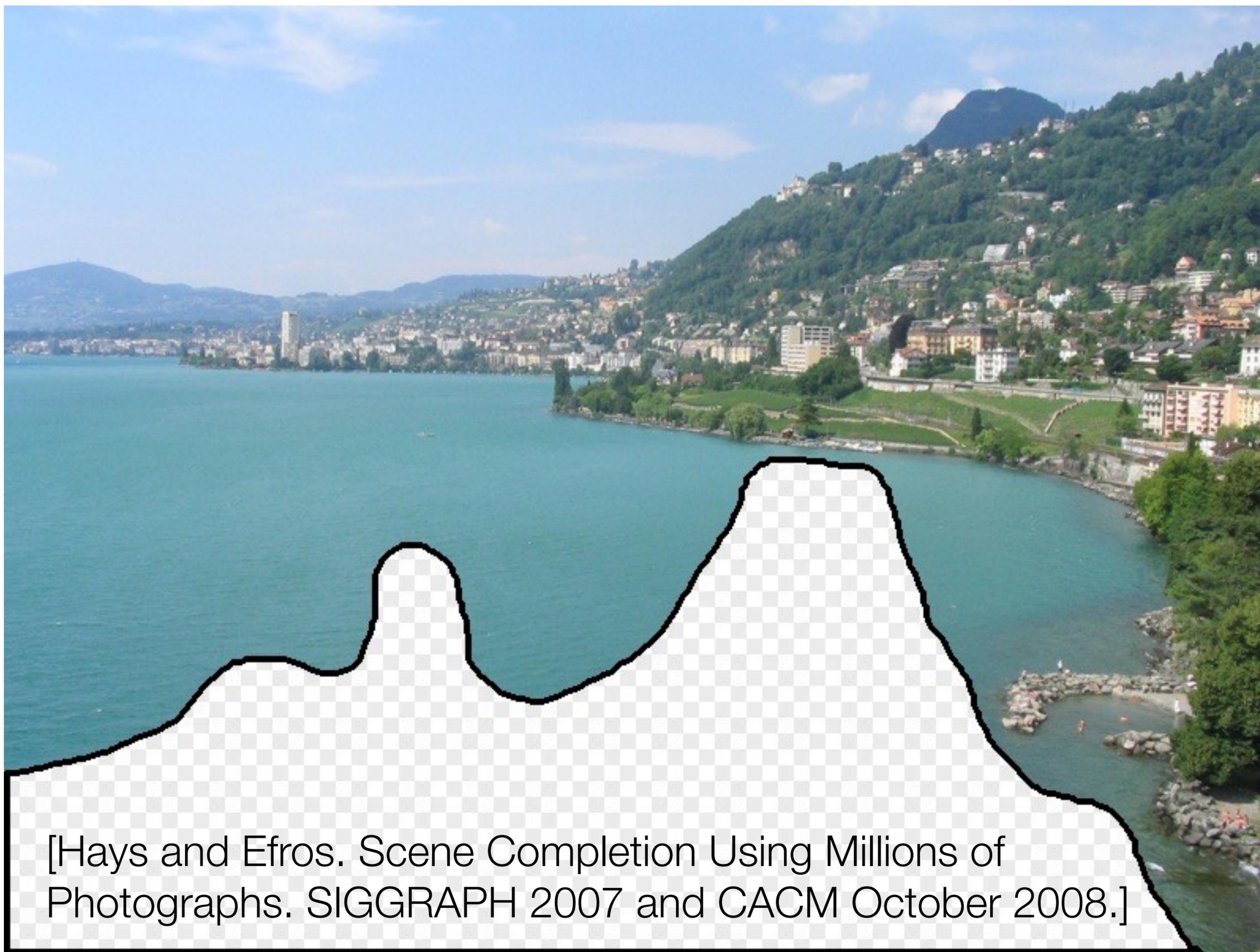
- “It’s all about the data!”
- Formalisms of learning (*Data, Compute, Objective Function, Hypothesis Space, Optimizer*)
- Case study #1: Linear least-squares
 - Empirical risk minimization
- Case study #2: Image classification
 - Softmax regression
- The problem of generalization

Edges





[Slide credit: Alexei Efros]



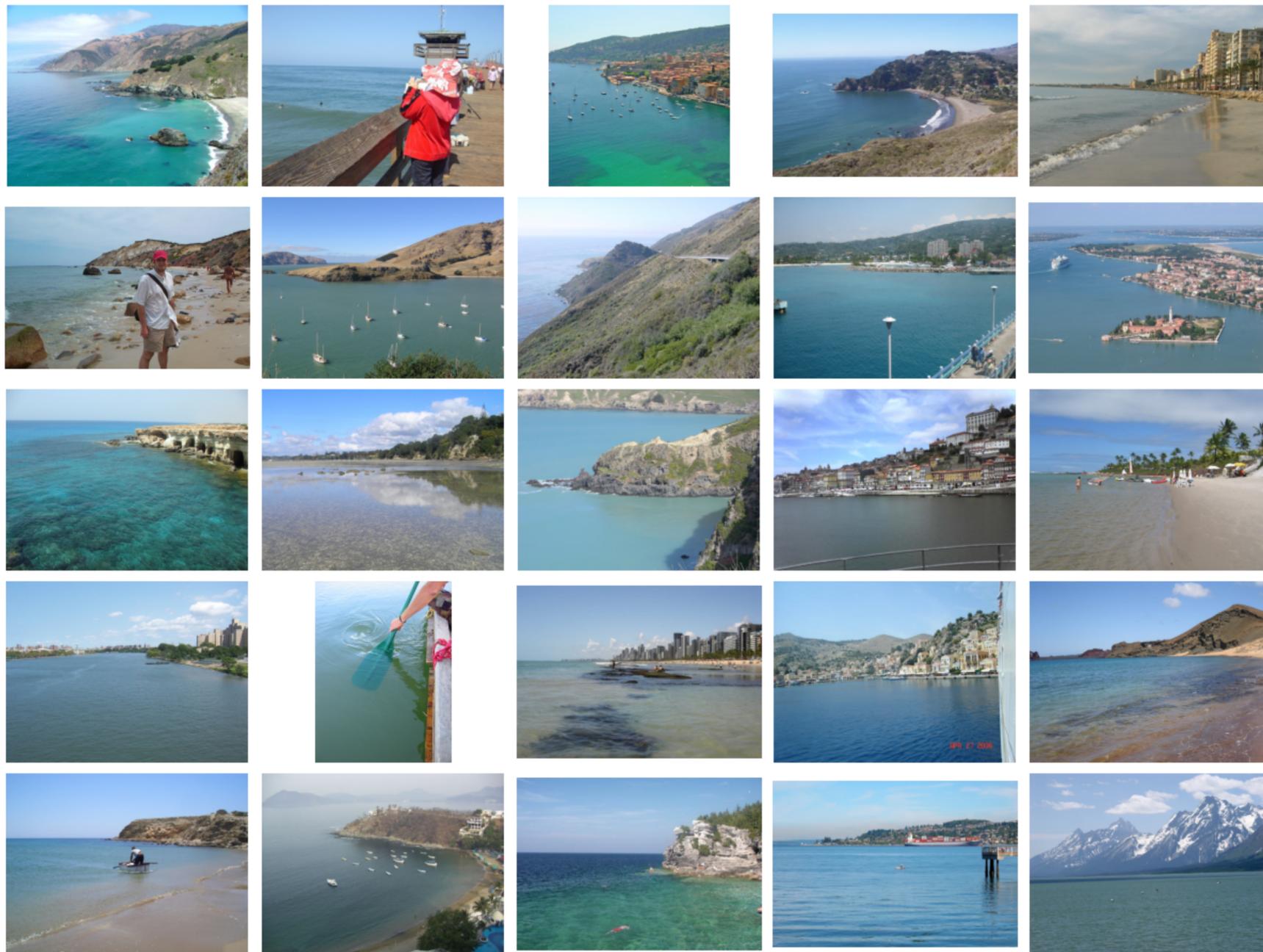
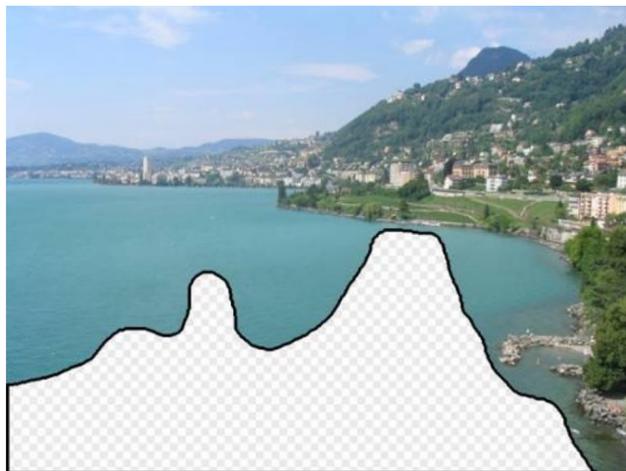
[Hays and Efros. Scene Completion Using Millions of Photographs. SIGGRAPH 2007 and CACM October 2008.]

[Slide credit: Alexei Efros]

2 Million Flickr Images

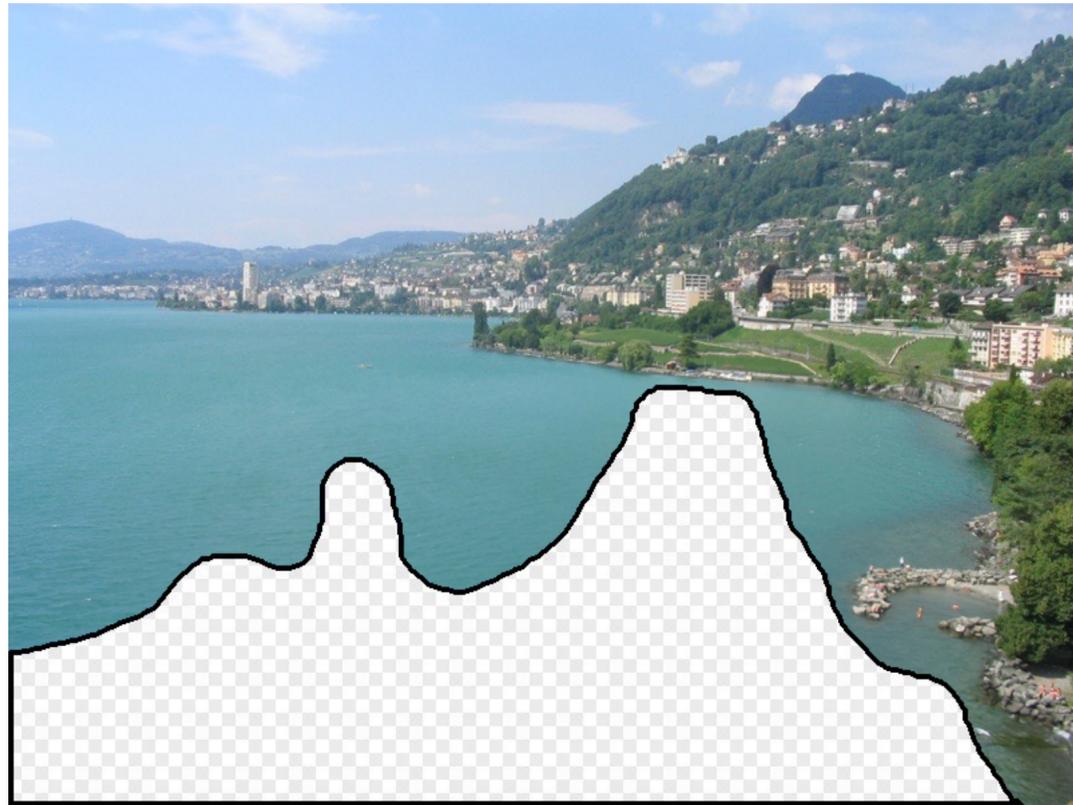


[Slide credit: Alexei Efros]



... 200 total

[Slide credit: Alexei Efros]



[Slide credit: Alexei Efros]





[Slide credit: Alexei Efros]



[Slide credit: Alexei Efros]



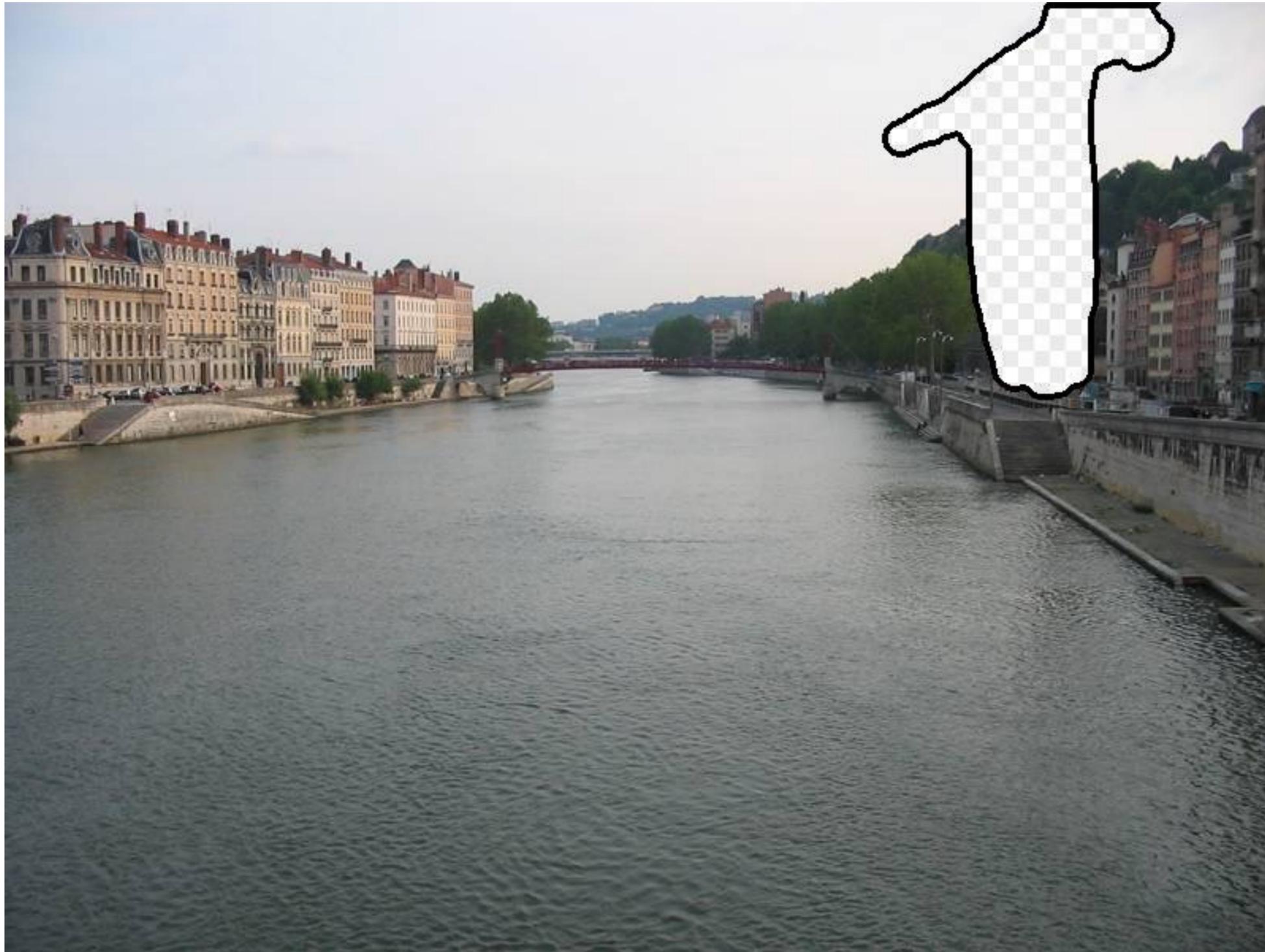
[Slide credit: Alexei Efros]



[Slide credit: Alexei Efros]



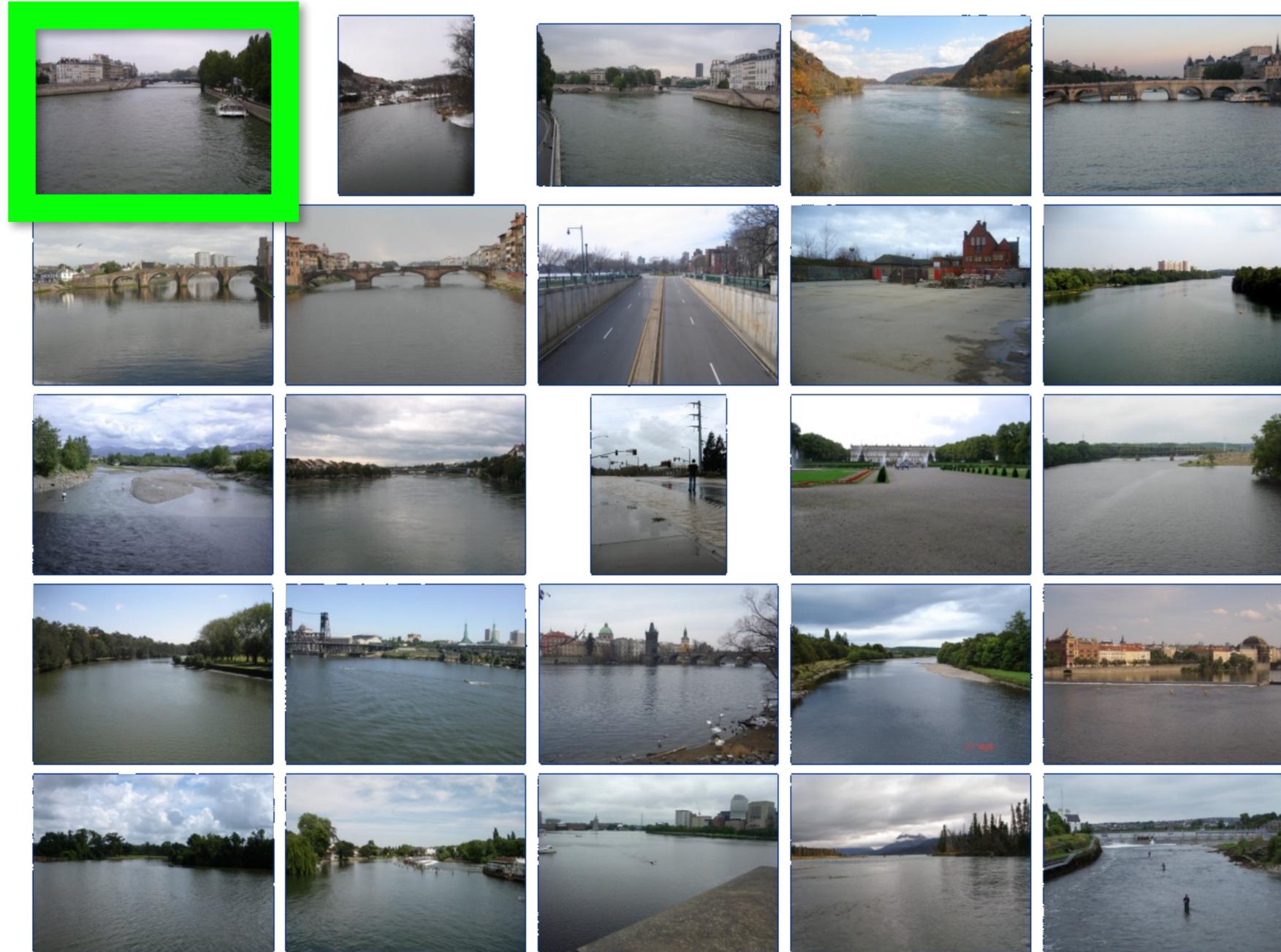
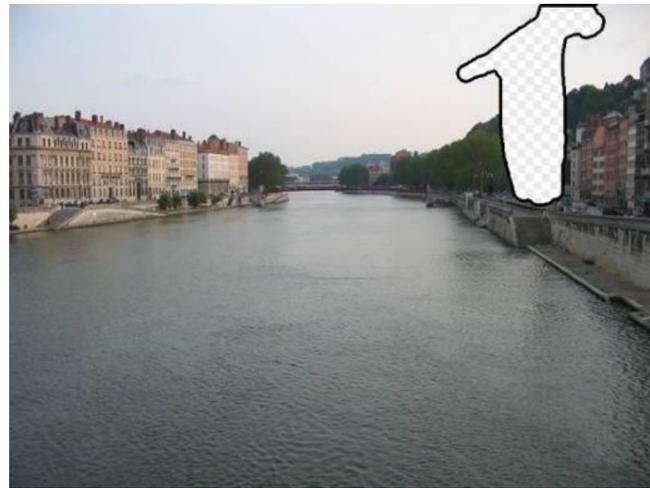
[Slide credit: Alexei Efros]



[Slide credit: Alexei Efros]



[Slide credit: Alexei Efros]

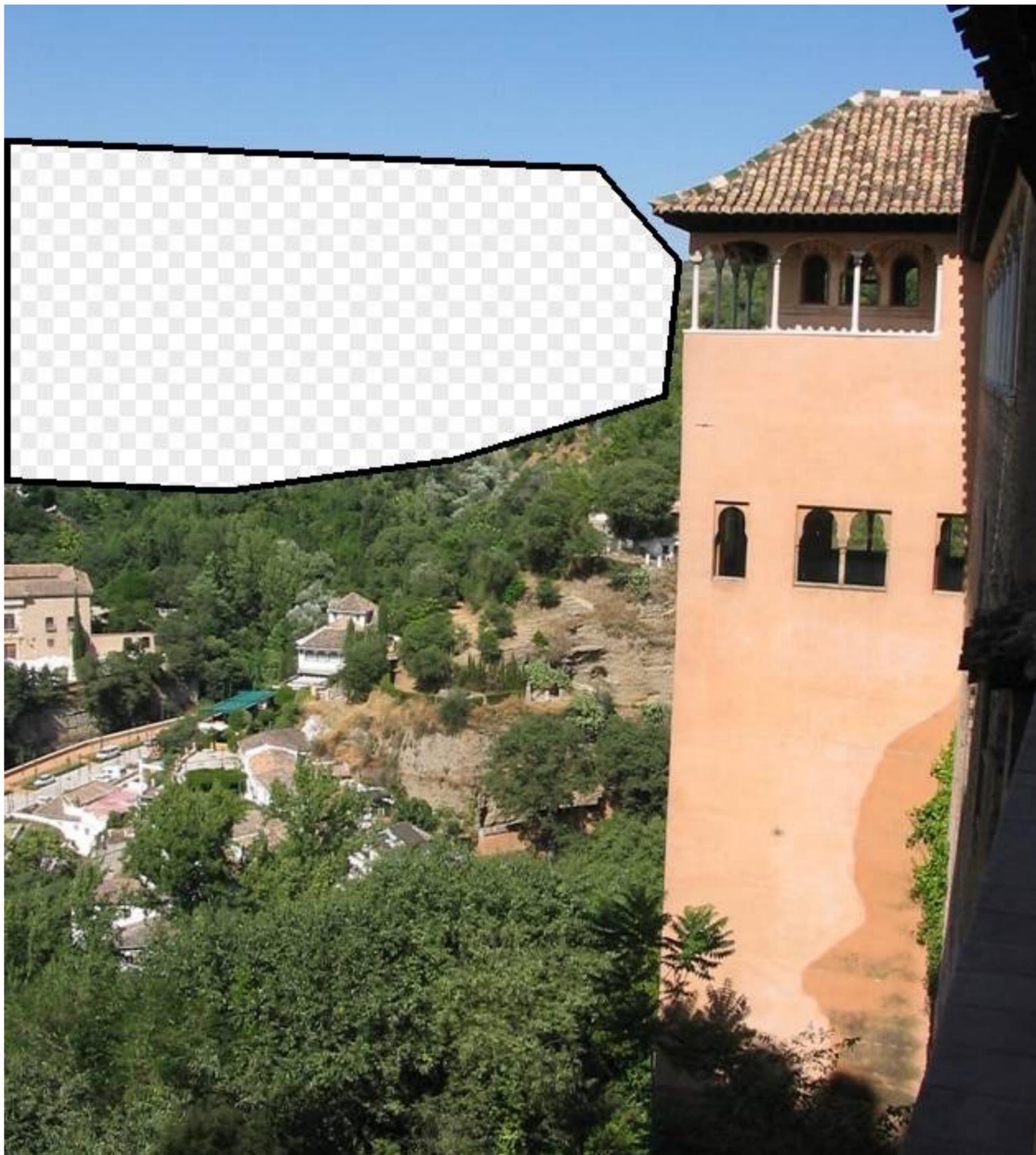


... 200 scene matches

[Slide credit: Alexei Efros]



[Slide credit: Alexei Efros]

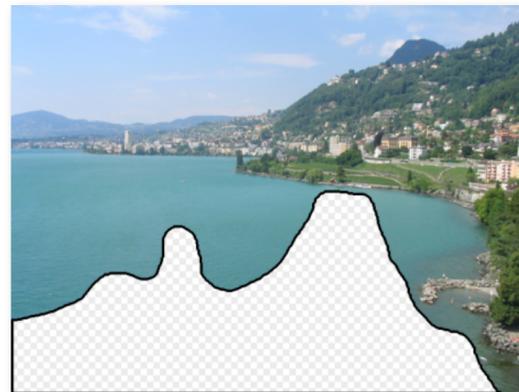


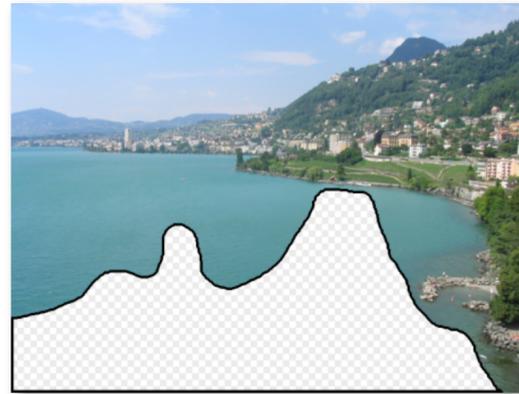
[Slide credit: Alexei Efros]



[Slide credit: Alexei Efros]

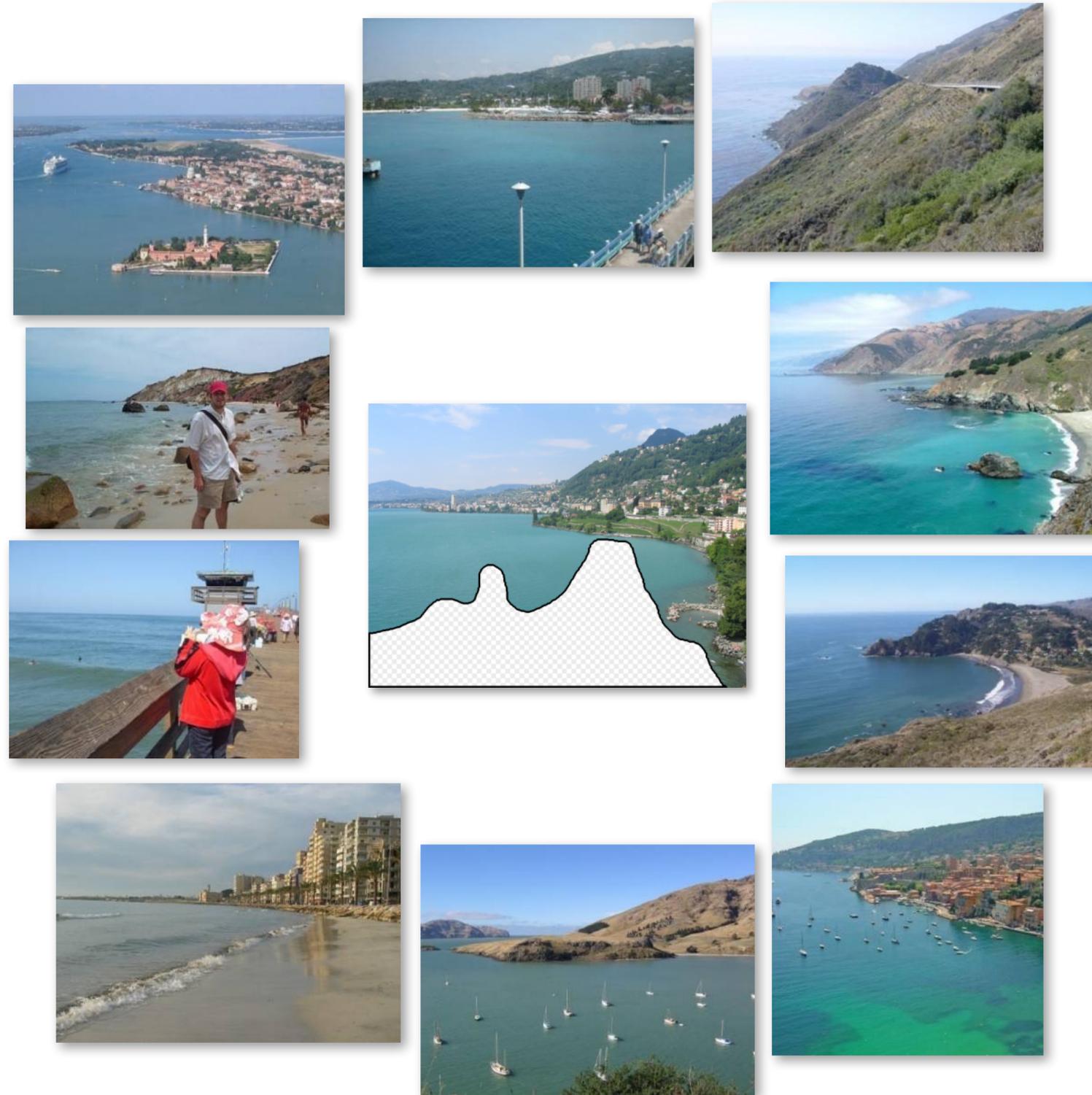
Why does it work?





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:

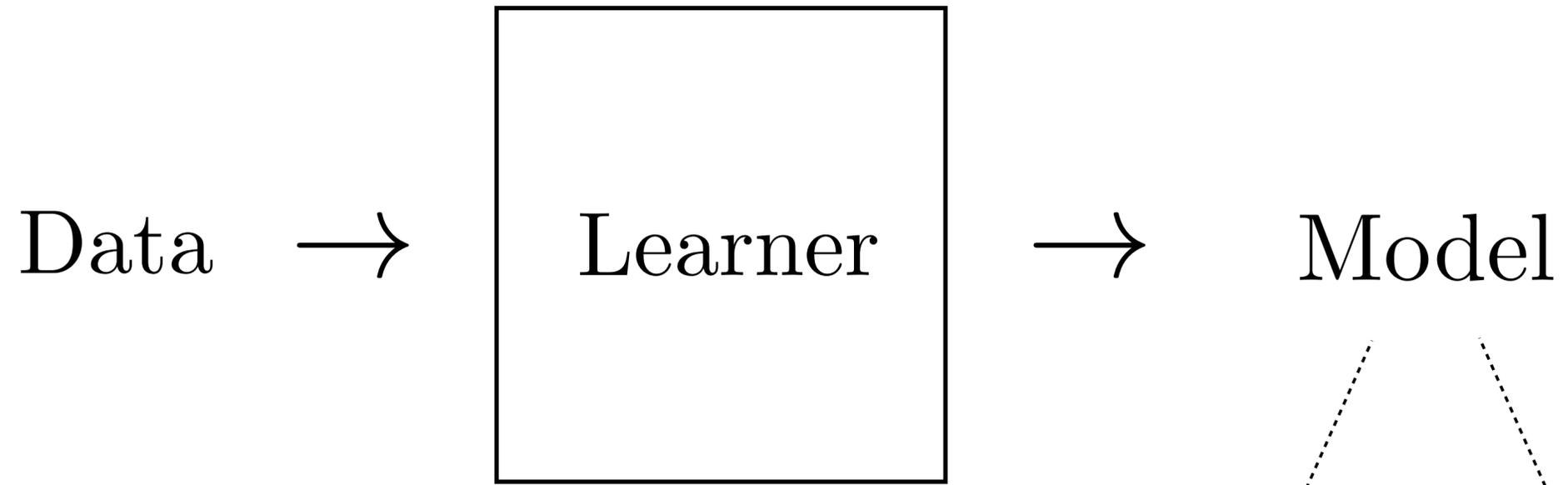
e.g., speech recognition, machine translation

Case study: Google

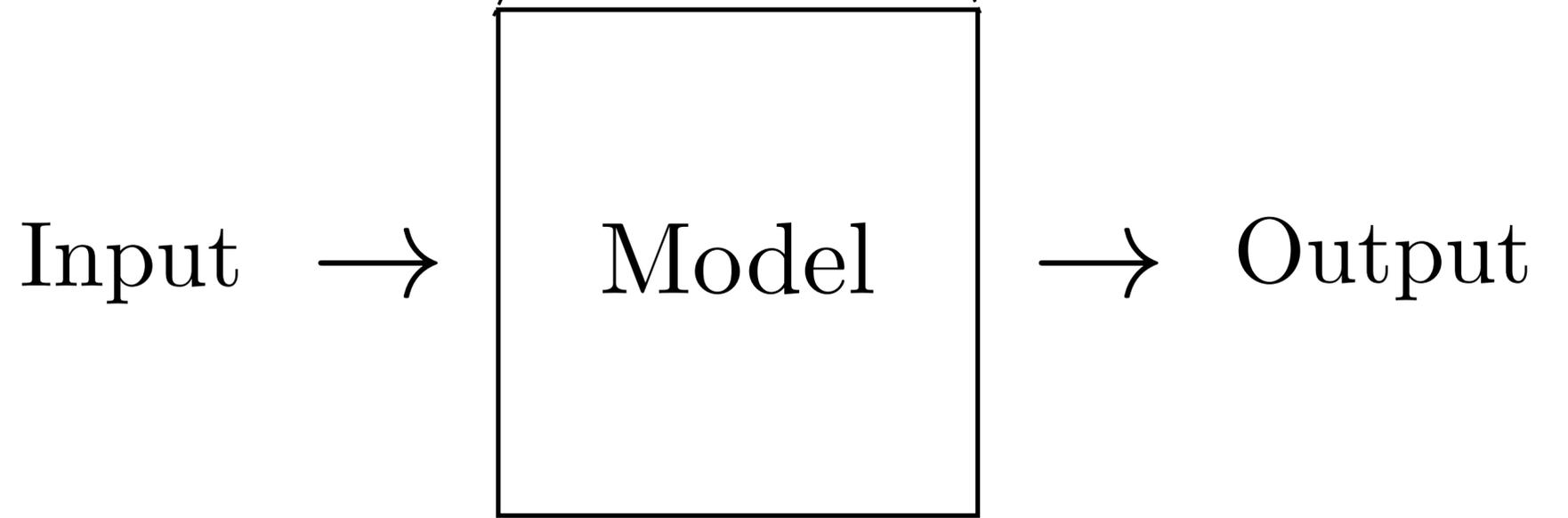
“For many tasks, once we have a billion or so examples, we essentially have a closed set that represents (or at least approximates) what we need...”

[Slide credit: Alexei Efros]

Learning



Inference



What does ☆ do?

$$2 \star 3 = 36$$

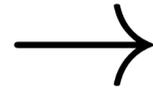
$$7 \star 1 = 49$$

$$5 \star 2 = 100$$

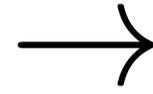
$$2 \star 2 = 16$$

Training

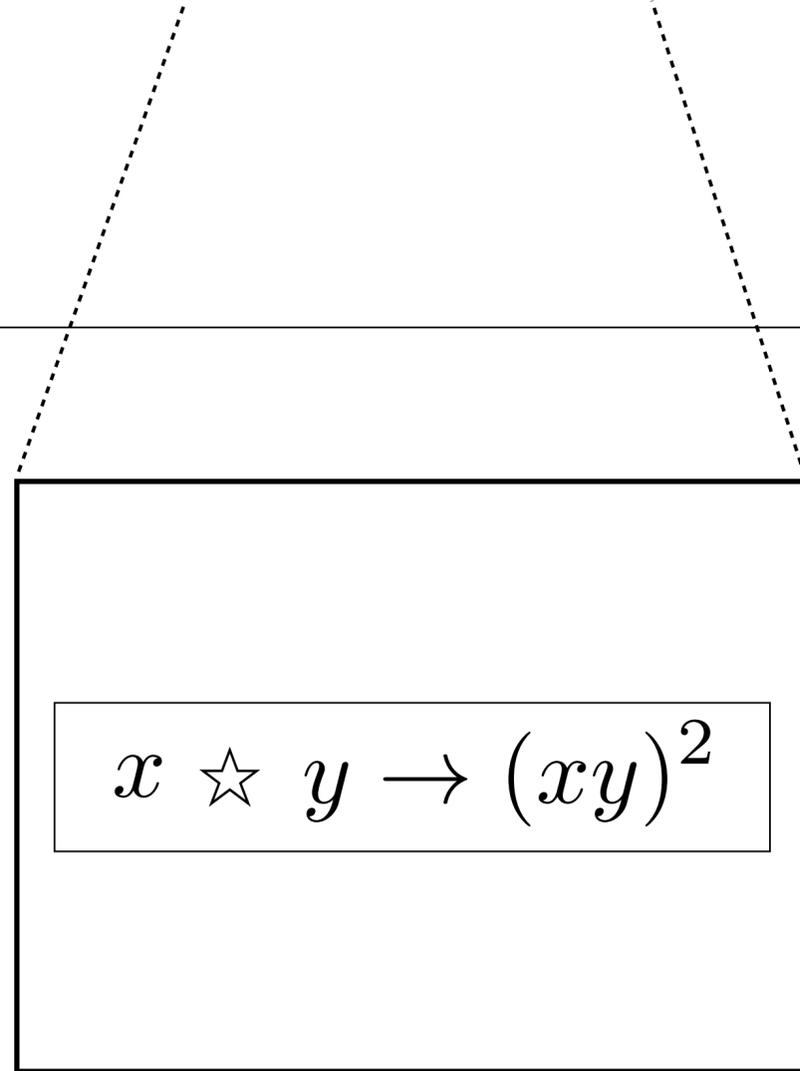
$$\begin{aligned} 2 \star 3 &= 36 \\ 7 \star 1 &= 49 \\ 5 \star 2 &= 100 \\ 2 \star 2 &= 16 \end{aligned}$$



Your
brain



$$x \star y \rightarrow (xy)^2$$



Testing

$$3 \star 5 \rightarrow$$

$$x \star y \rightarrow (xy)^2$$

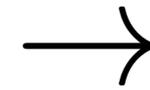
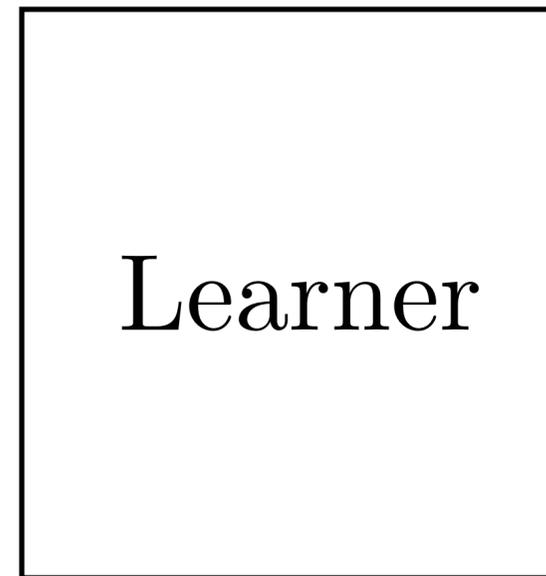
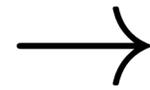
$$\rightarrow 225$$

Learning from examples

(aka **supervised learning**)

Training data

$\{\text{input}: [2, 3], \text{output}: 36\}$
 $\{\text{input}: [7, 1], \text{output}: 49\}$
 $\{\text{input}: [5, 2], \text{output}: 100\}$
 $\{\text{input}: [2, 2], \text{output}: 16\}$



f

Learning from examples

(aka **supervised learning**)

Training data

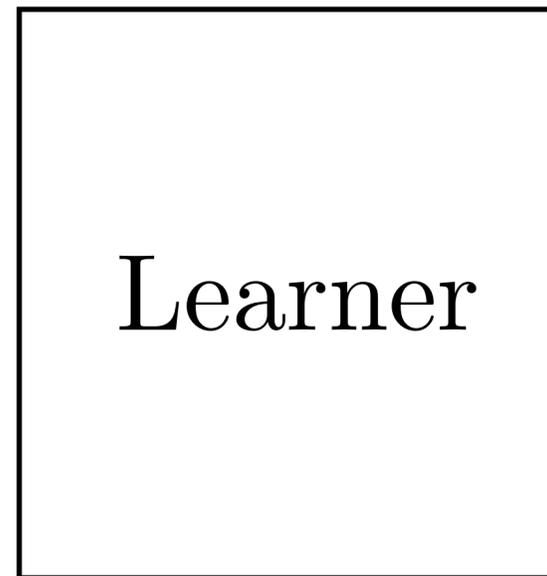
$\{x^{(1)}, y^{(1)}\}$

$\{x^{(2)}, y^{(2)}\}$

$\{x^{(3)}, y^{(3)}\}$

...

→



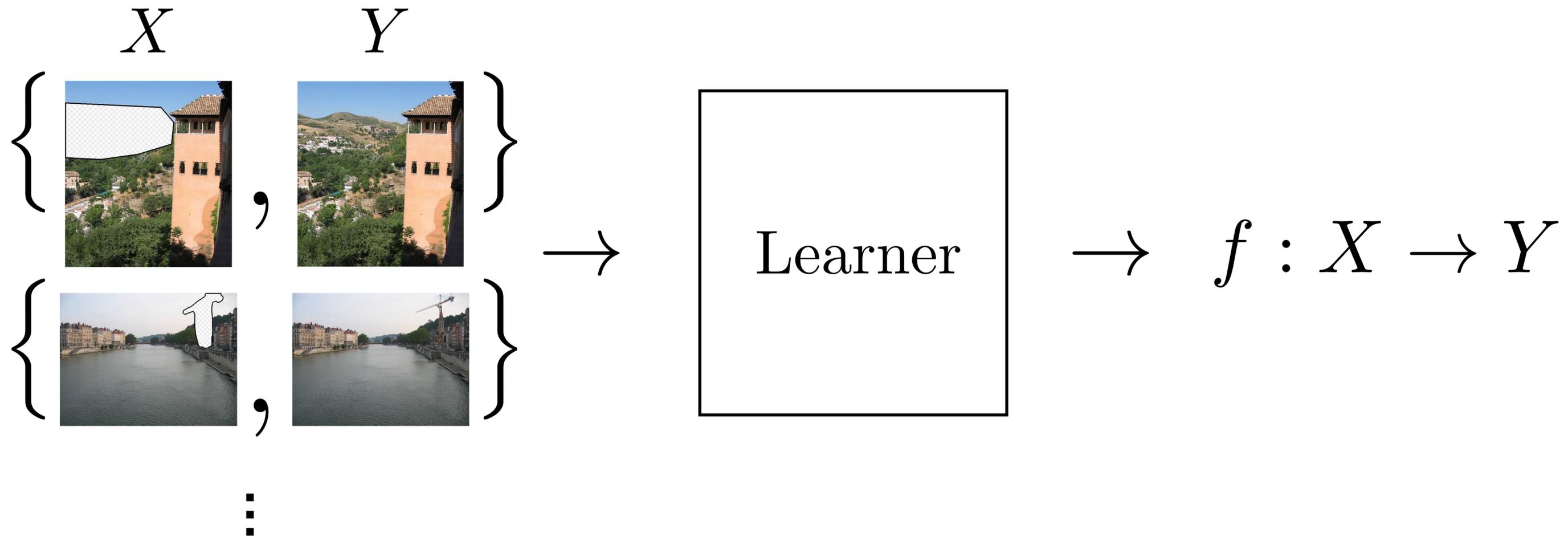
→

$f : X \rightarrow Y$

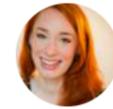
Learning from examples

(aka **supervised learning**)

Training data



Case study #1: Linear least squares



Hannah Fry 

@FryRsquared

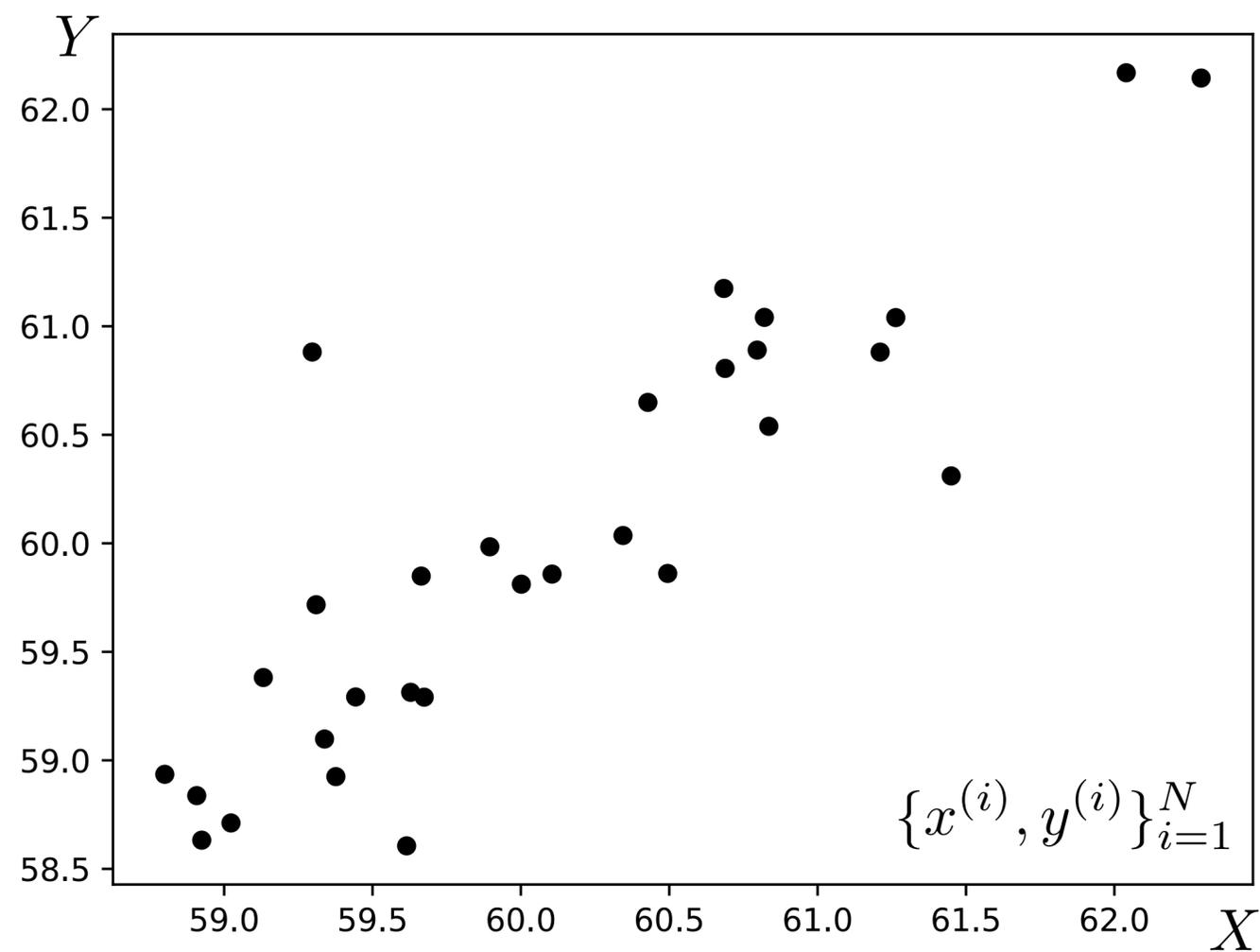
Follow



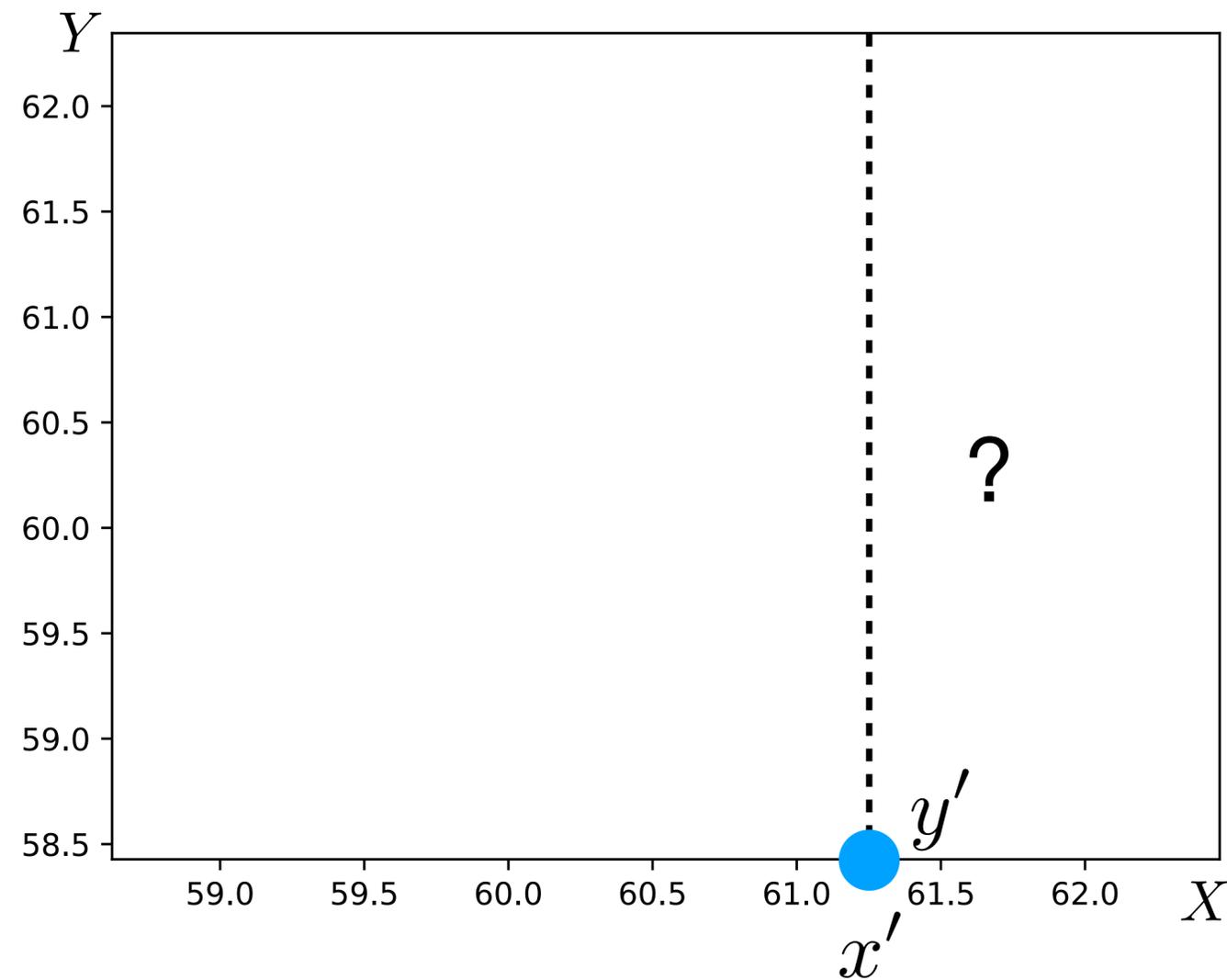
FFS 

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 $BSR = X(a) + Y(b) + Z(c) \dots$ and so and so forth.

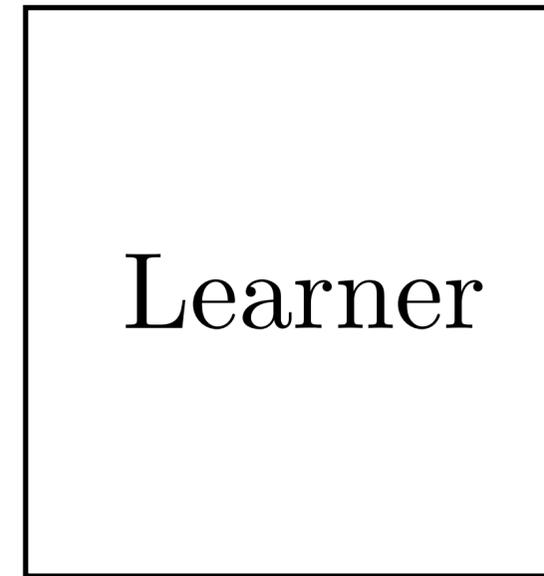
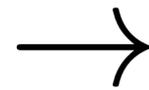
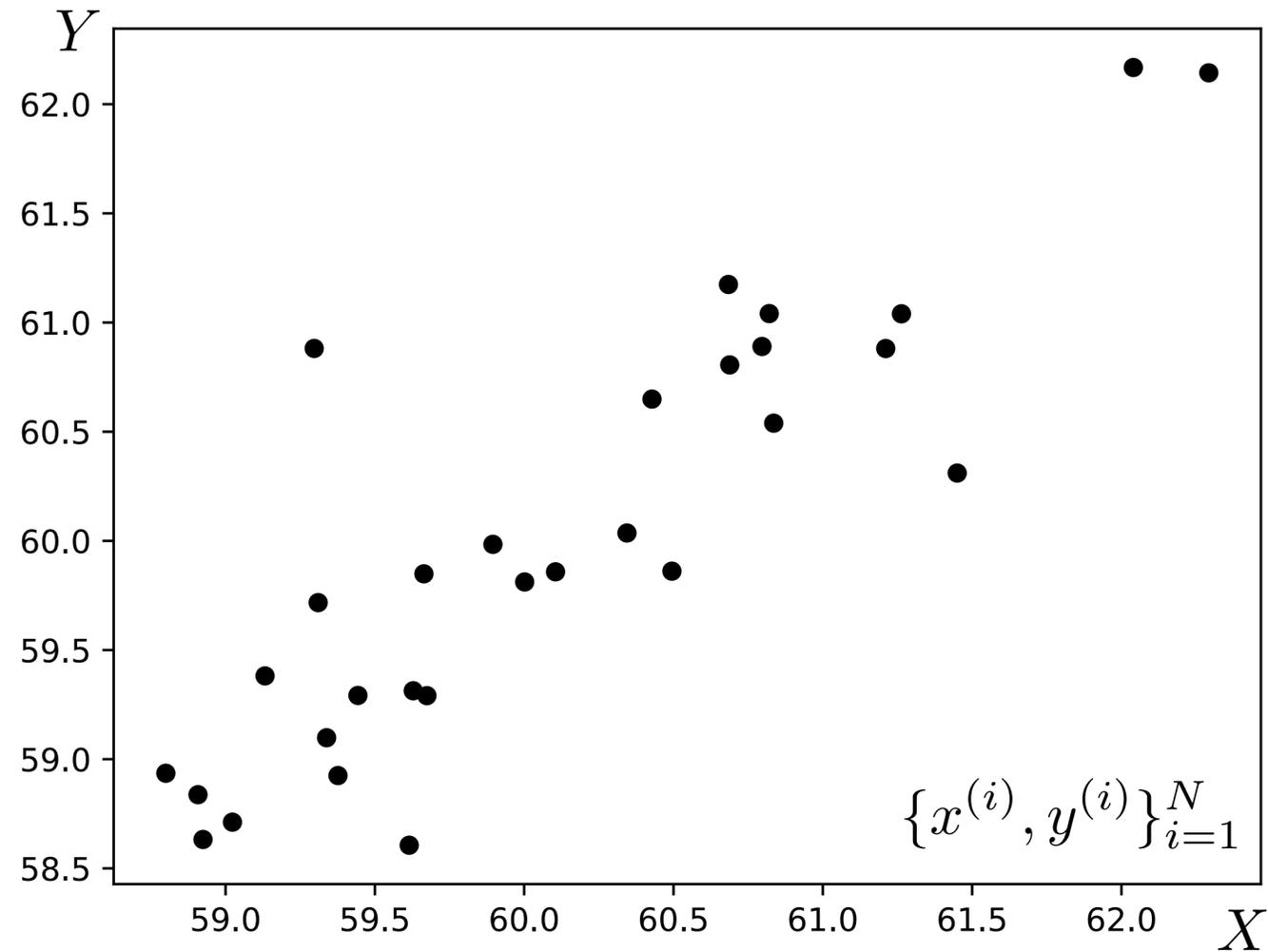
Training data



Test query



Training data

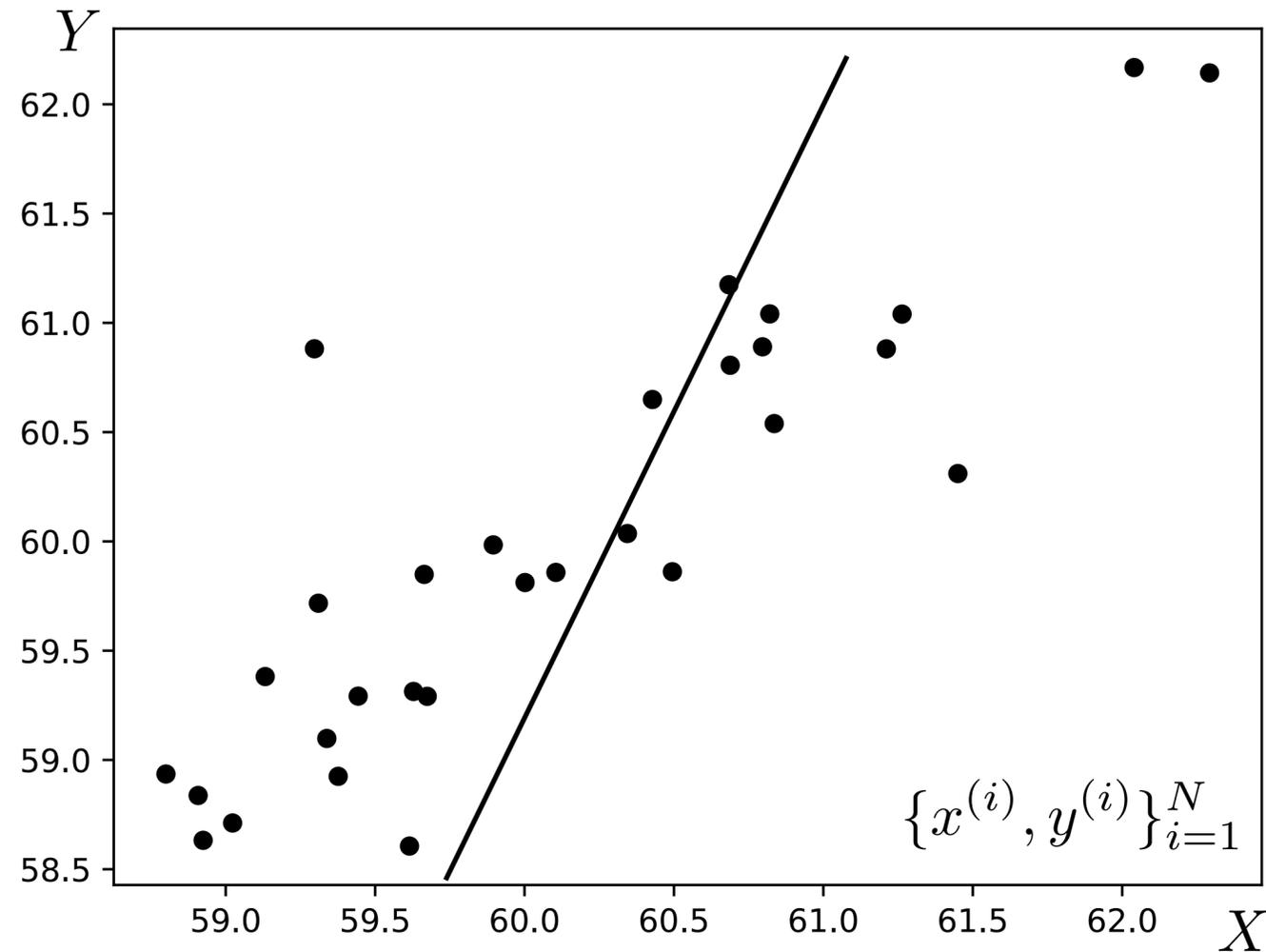


$$f_{\theta}(x) = \theta_1 x + \theta_0$$

Hypothesis space

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

Training data

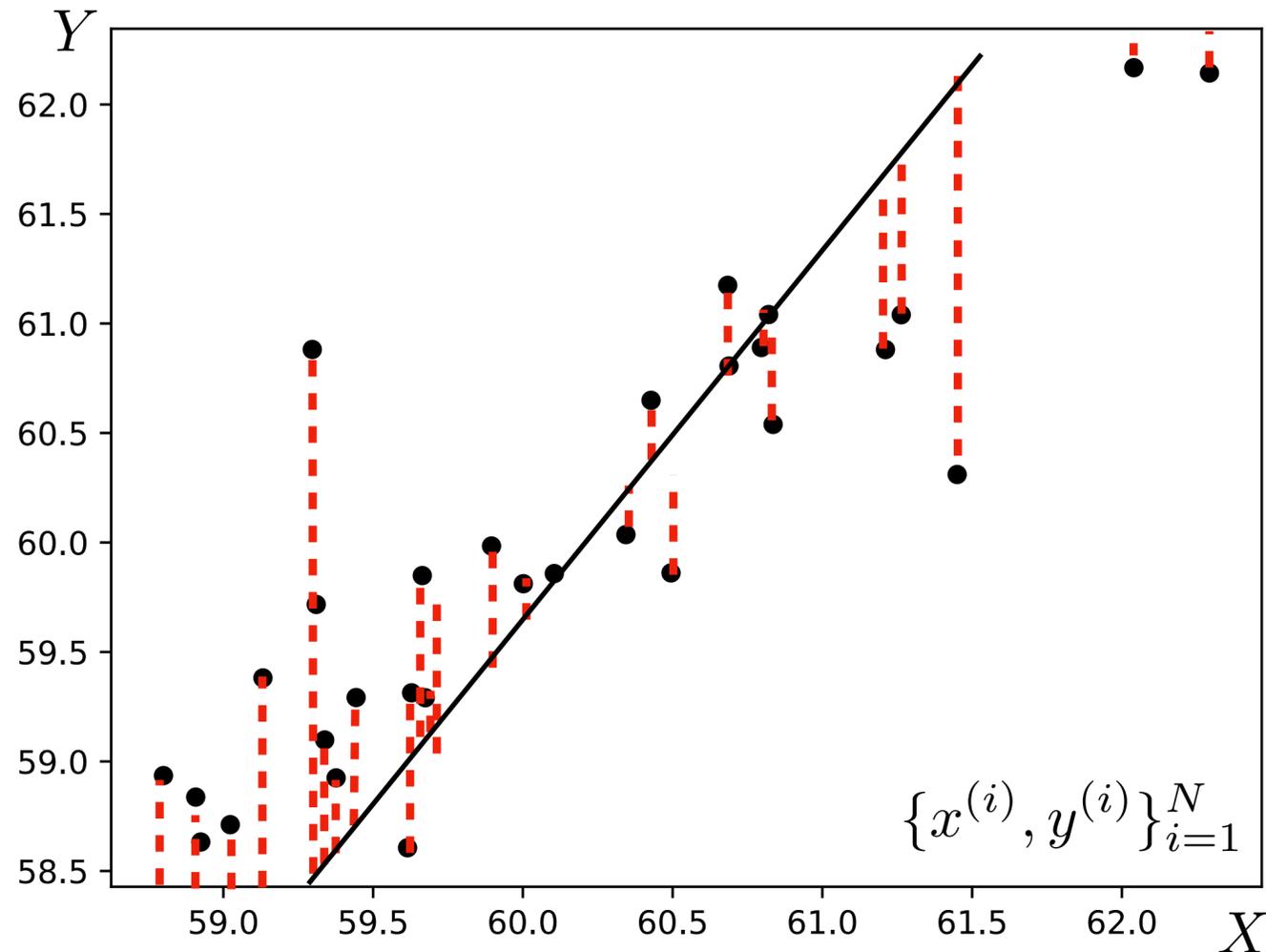


Search for the **parameters**, $\theta = \{\theta_0, \theta_1\}$, that best fit the data.

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

Best fit in what sense?

Training data



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

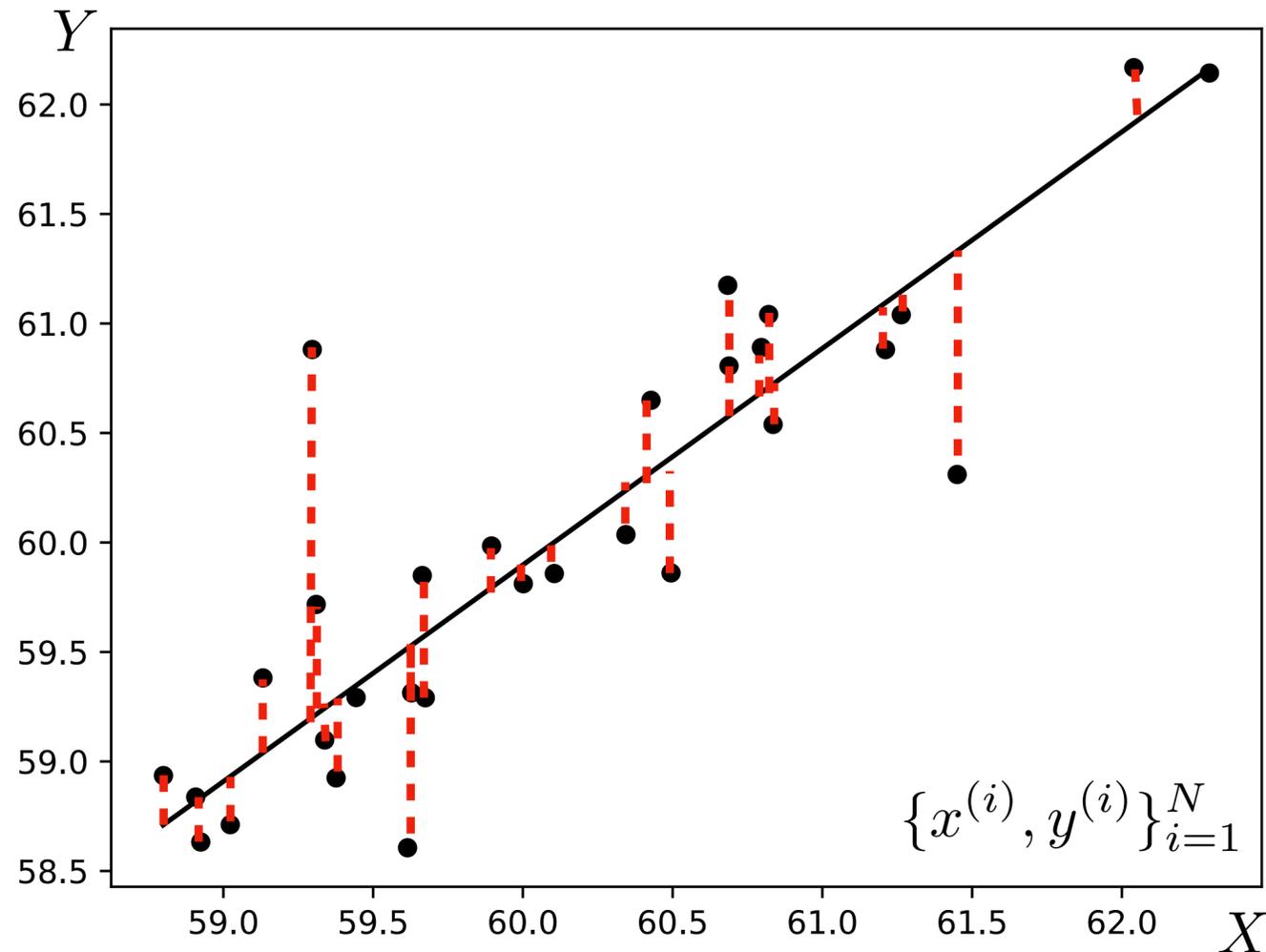
$$f_{\theta}(x) = \theta_1 x + \theta_0$$

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.

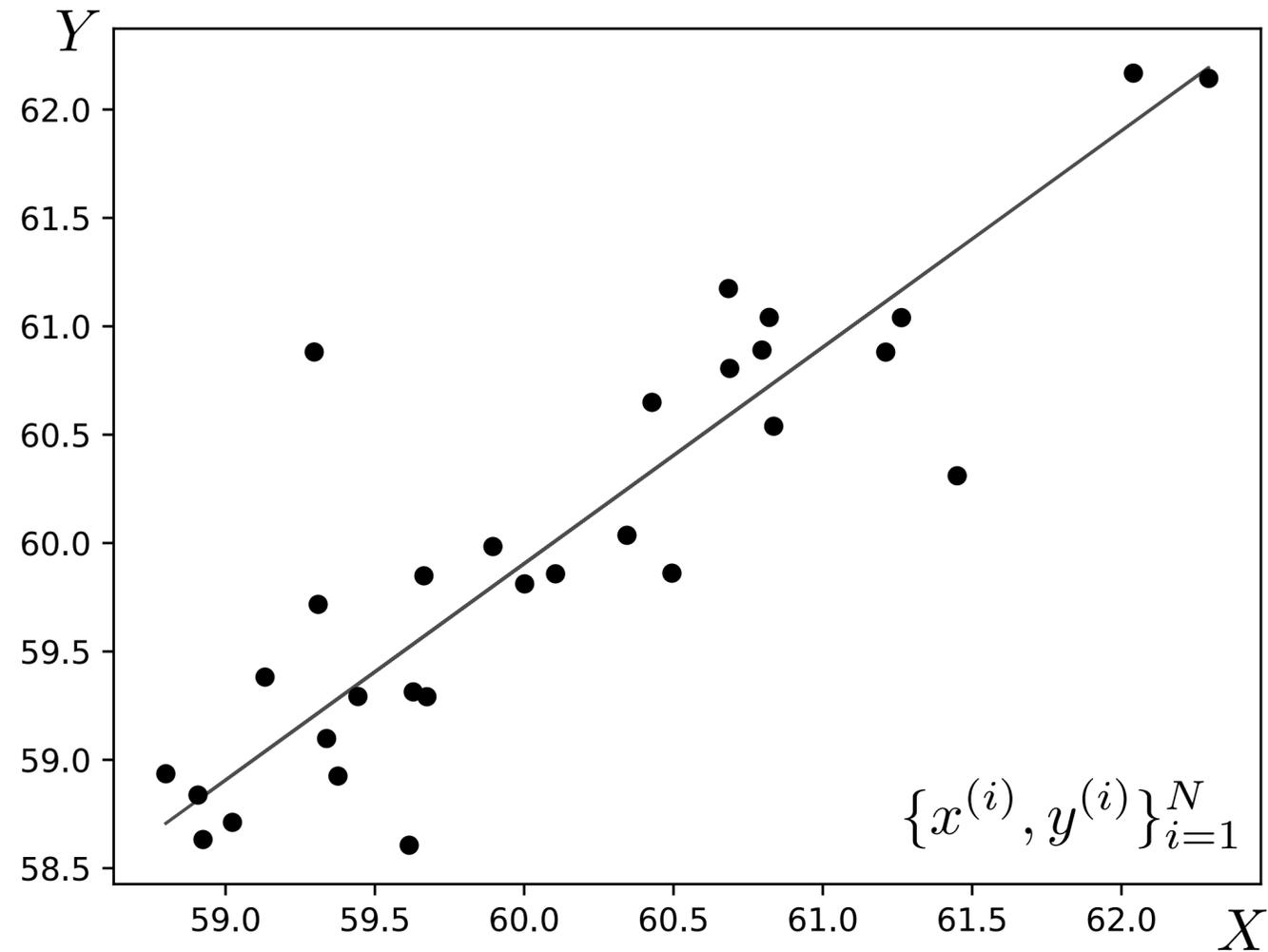
$$f_{\theta}(x) = \theta_1 x + \theta_0$$

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:

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

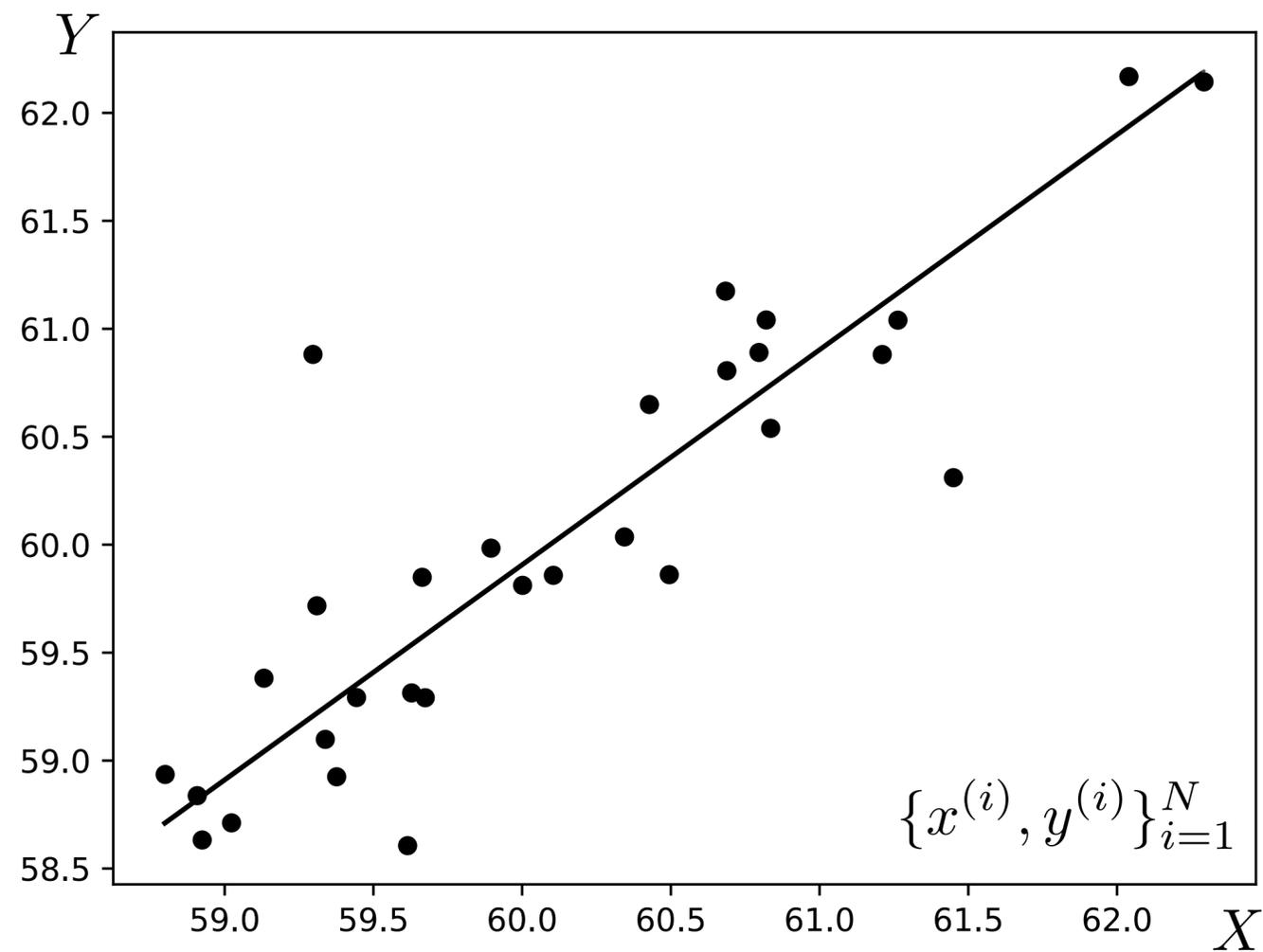
Training data



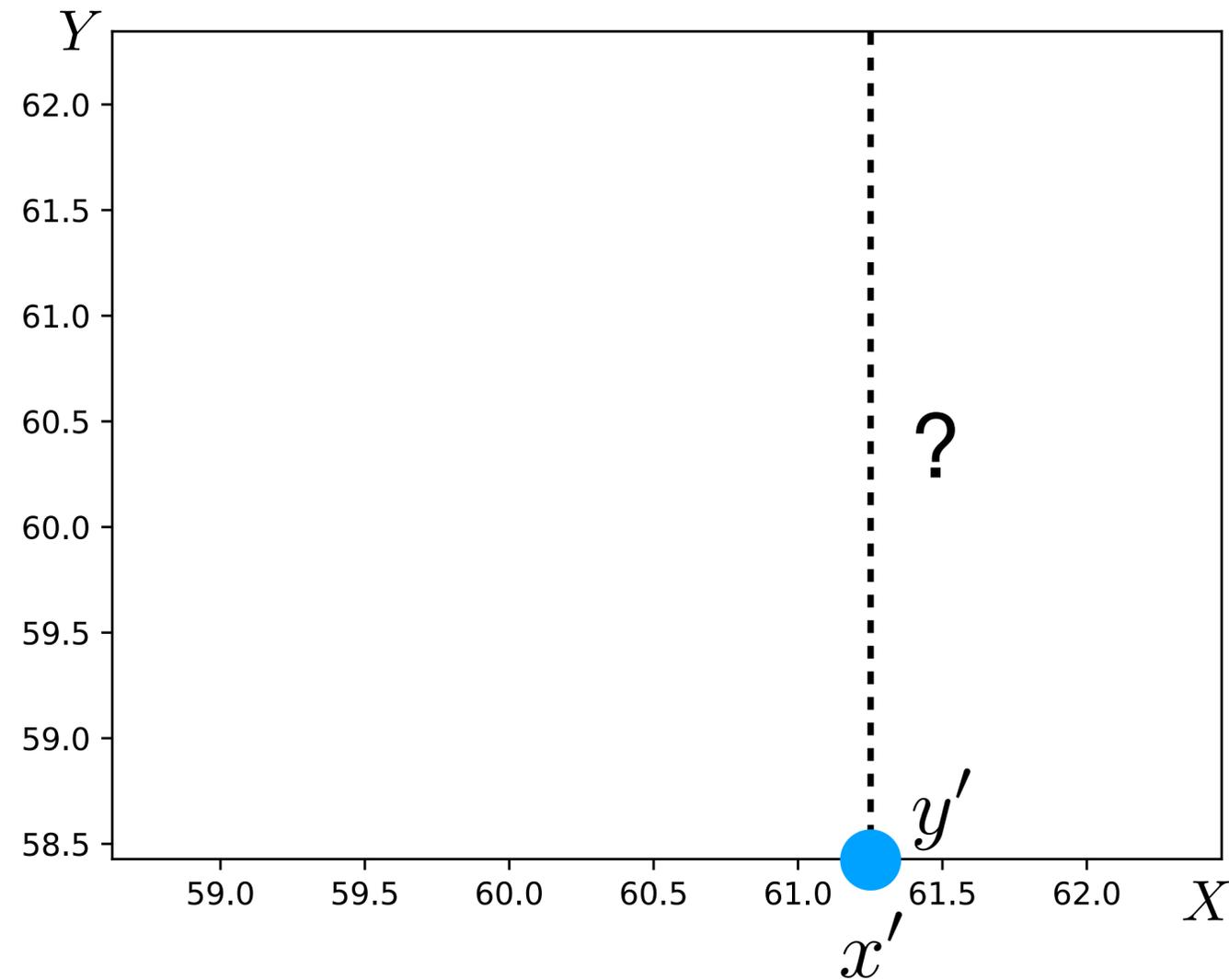
Complete learning problem:

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

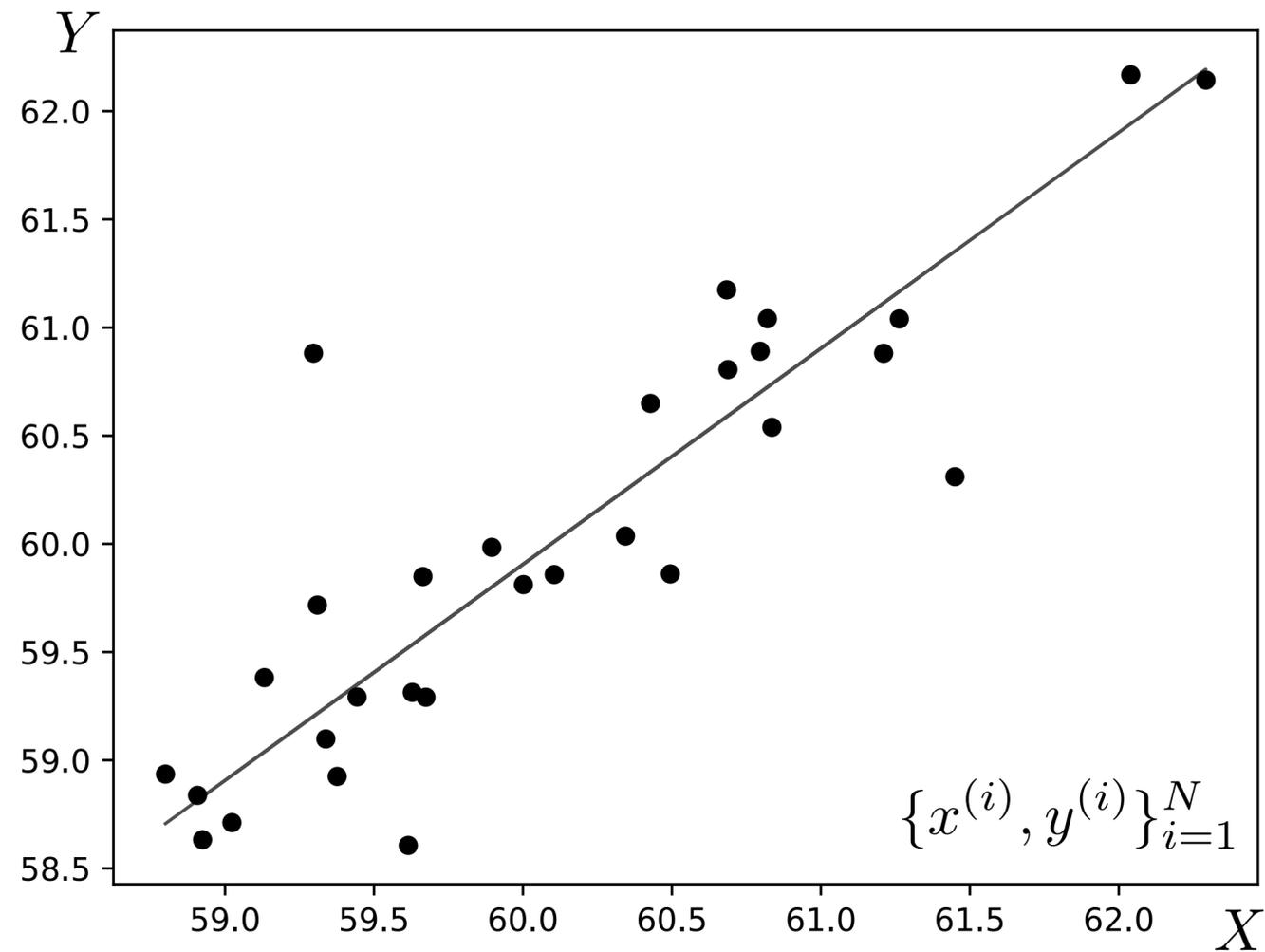
Training data



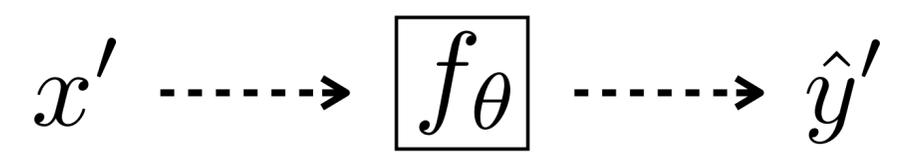
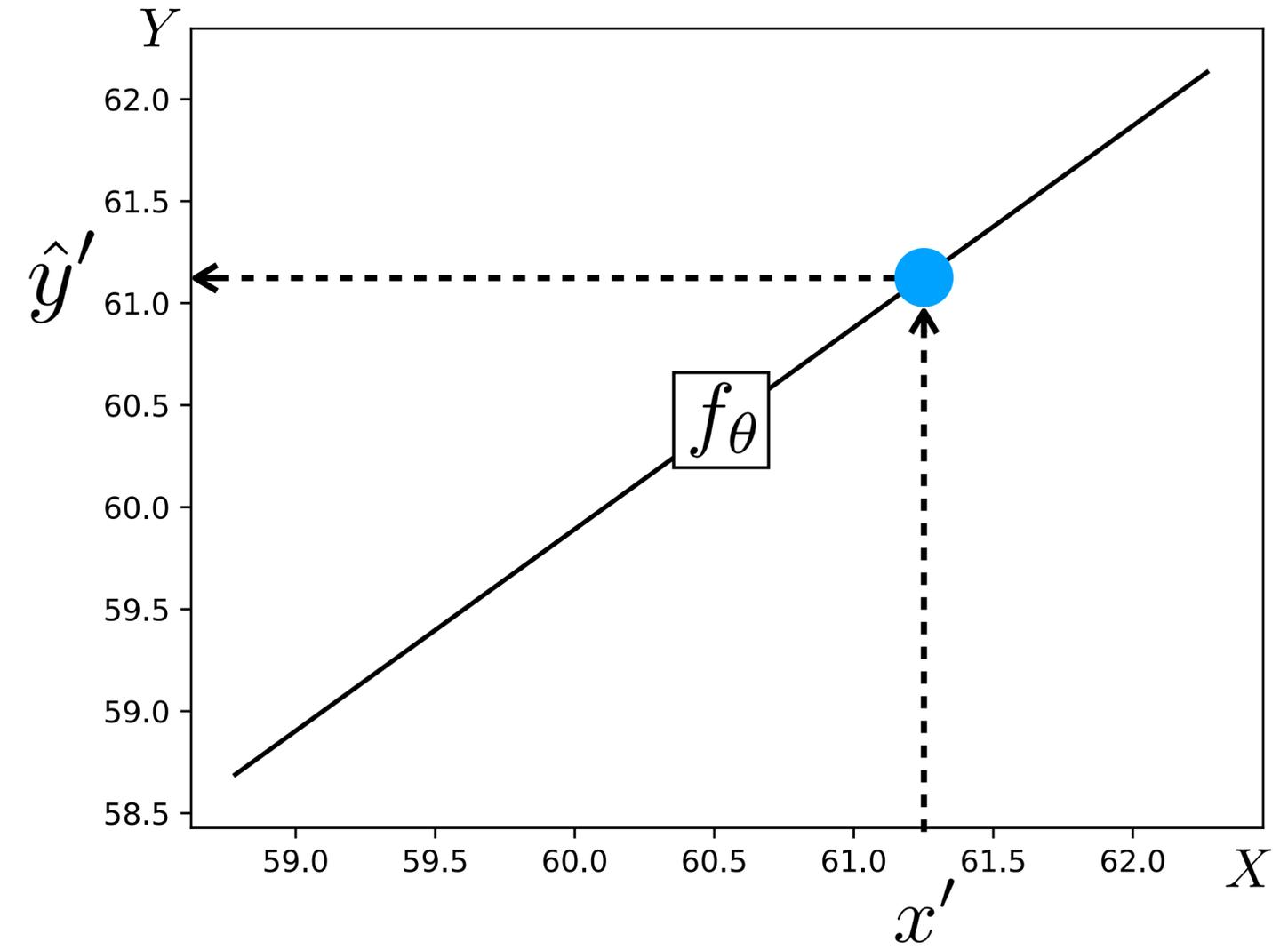
Test query



Training data



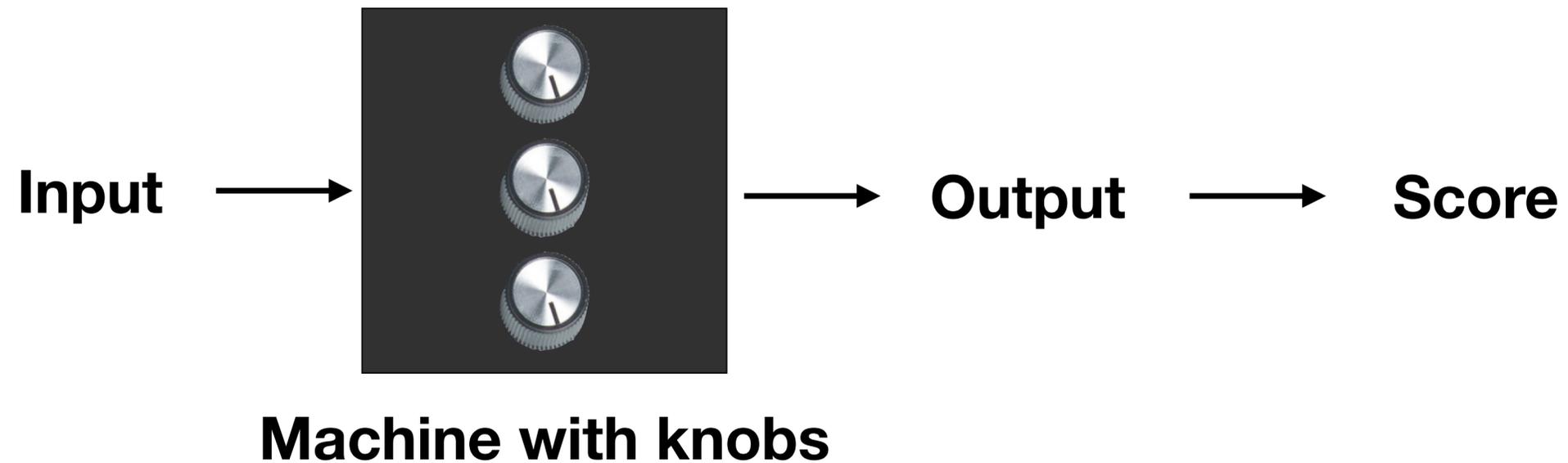
Test query



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!**



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

In the linear case:

Learning problem

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

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

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

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

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

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

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

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

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

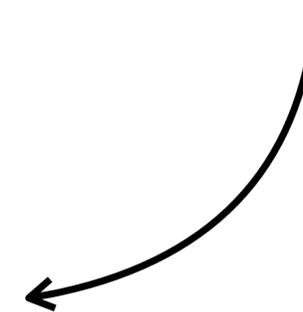
Solution

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)

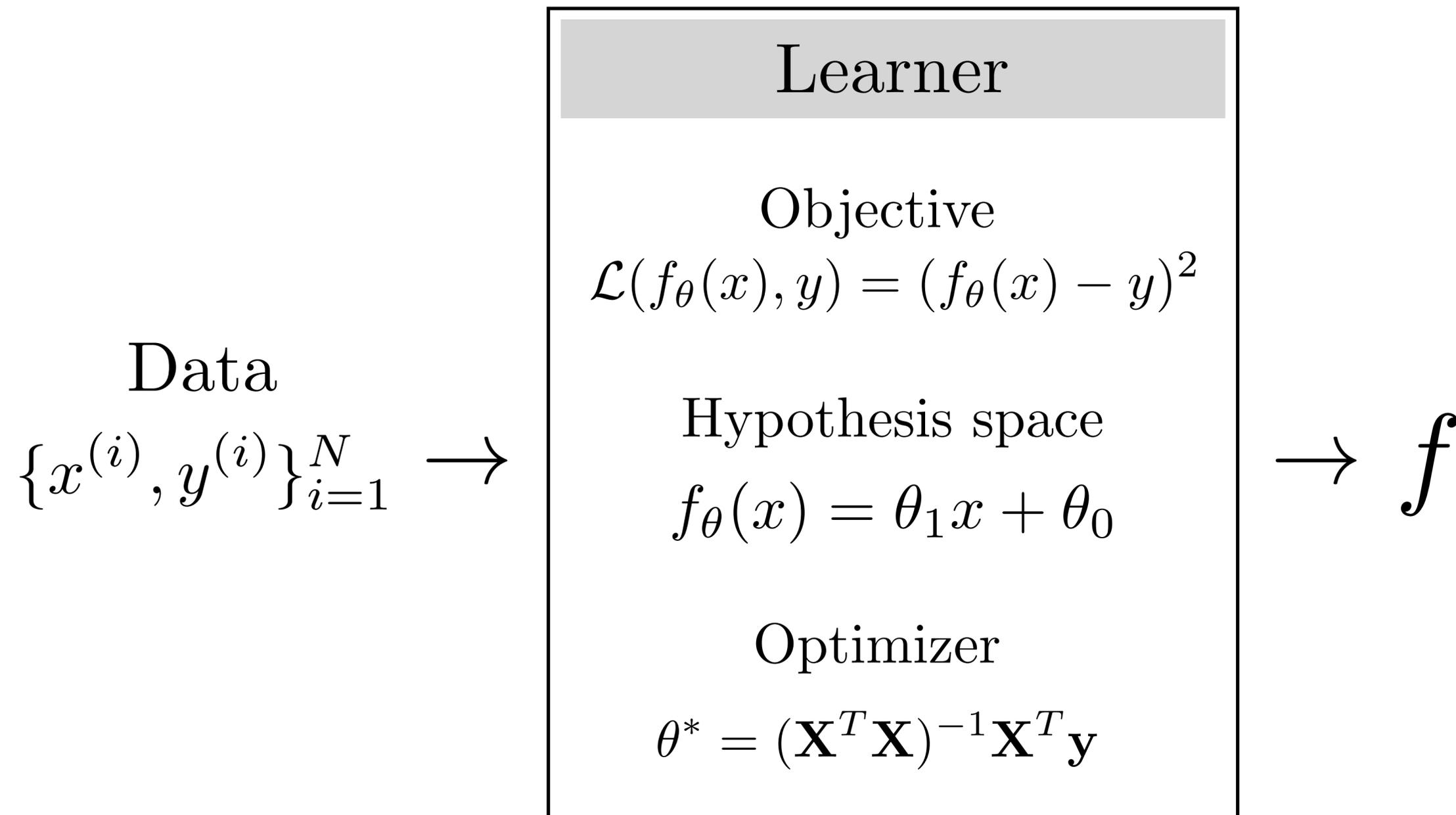
$$f^* = \arg \min_{f \in \mathcal{F}} \sum_{i=1}^N \mathcal{L}(f(\mathbf{x}^{(i)}), \mathbf{y}^{(i)})$$

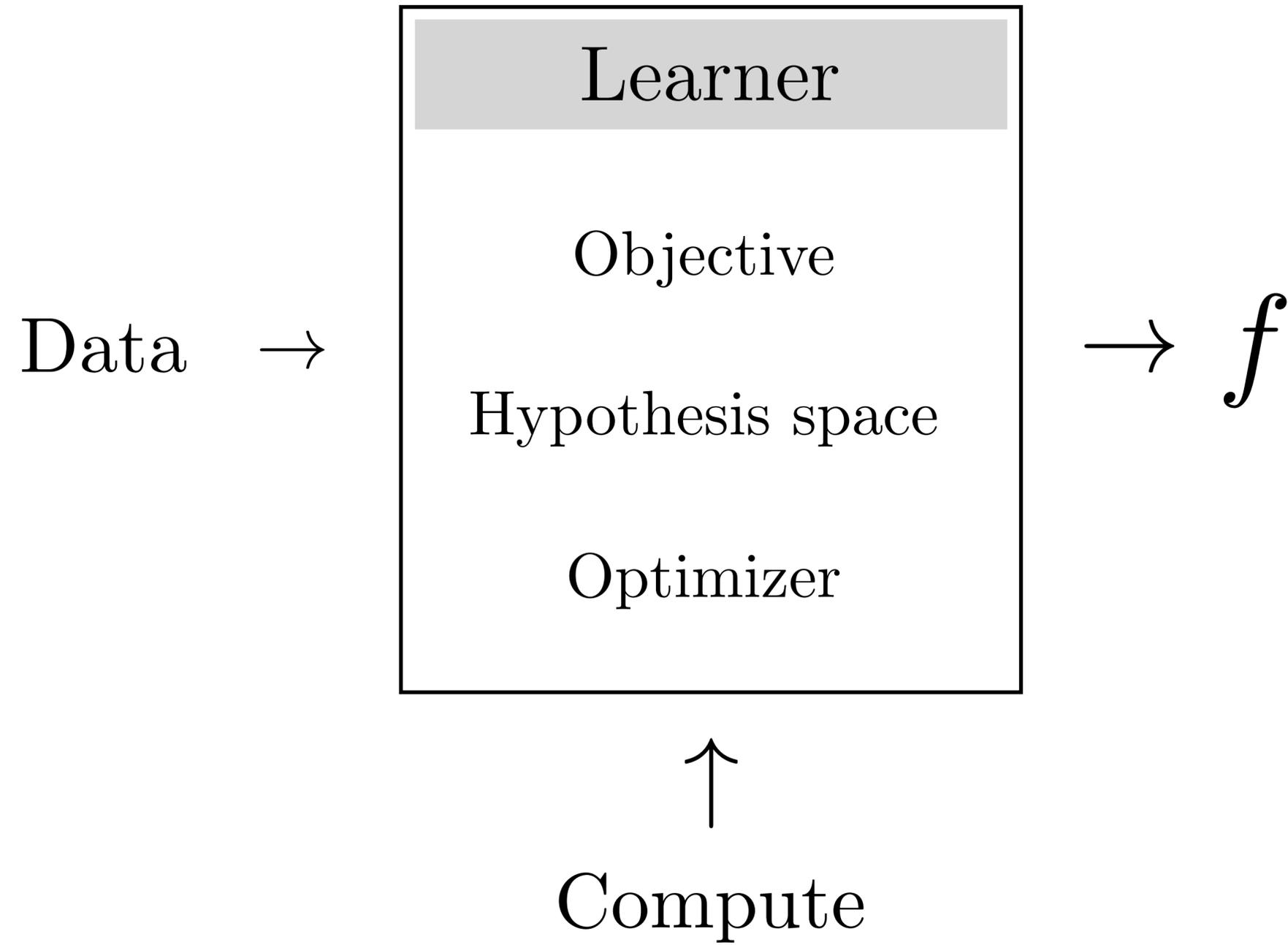
Objective function
(loss)

Hypothesis space

Training data

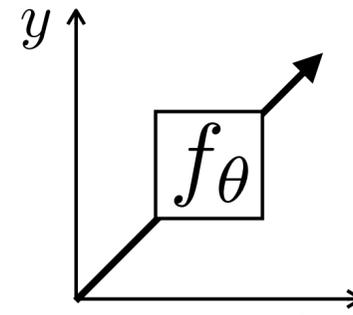
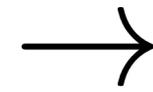
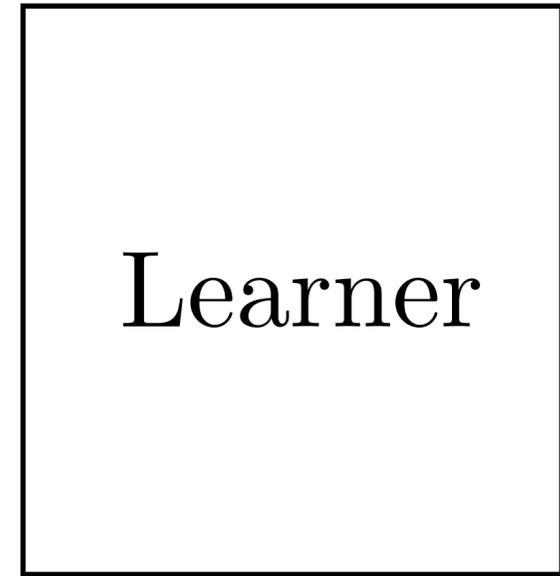
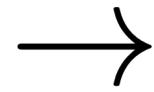
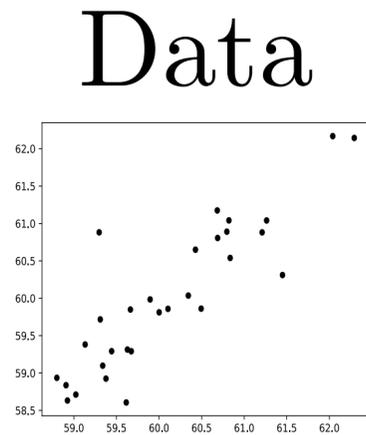
Case study #1: Linear least squares





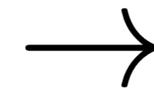
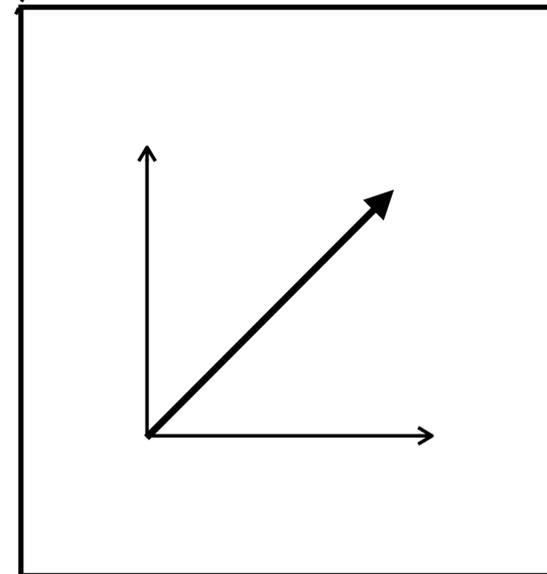
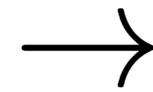
Example 1: Linear least squares

Training



Testing

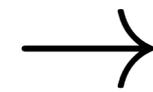
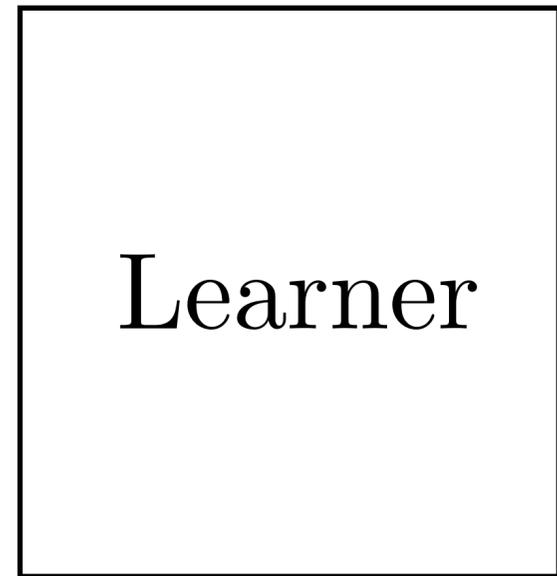
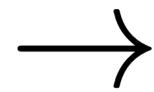
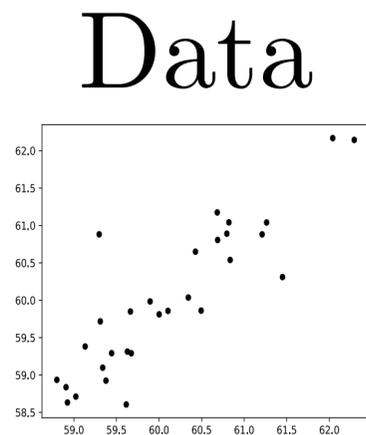
Input



Output

Example 2: Program induction

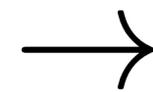
Training



```
def predict(x):  
    y = 0.8*x + 2  
    return y
```

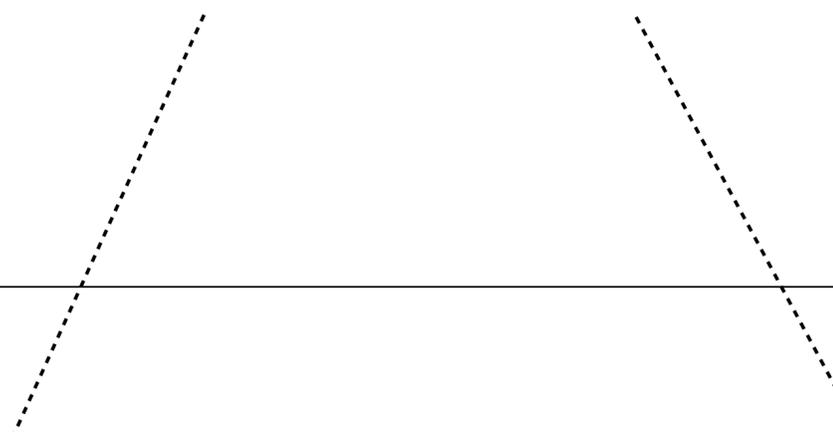
Testing

Input



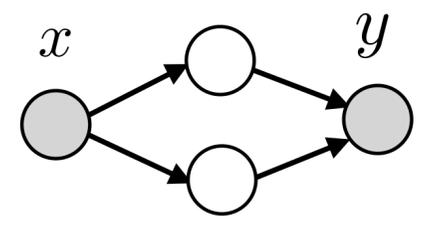
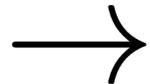
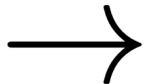
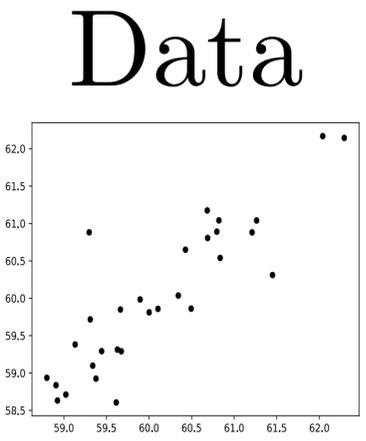
```
def predict(x):  
    y = 0.8*x + 2  
    return y
```

→ Output



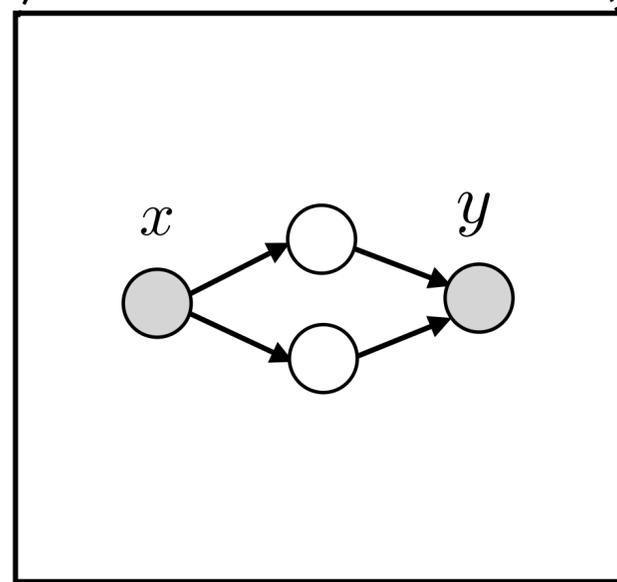
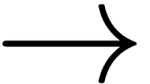
Example 3: Deep learning

Training



Testing

Input



Output

Space of all functions

 Hypothesis space (haystack)

 True solution (needle)

**Space we will
search**



Space of all functions



Hypothesis space (haystack)



True solution (needle)

**Space we will
search**



Deep nets

Space of all functions



Hypothesis space (haystack)



True solution (needle)

**Space we will
search**



Linear functions

True solution is linear

Space of all functions



Hypothesis space (haystack)



True solution (needle)



**Space we will
search**



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 sufficiently expressive

4. Assume we optimize perfectly

5. Assume we train on exactly the data we care about

Image classification

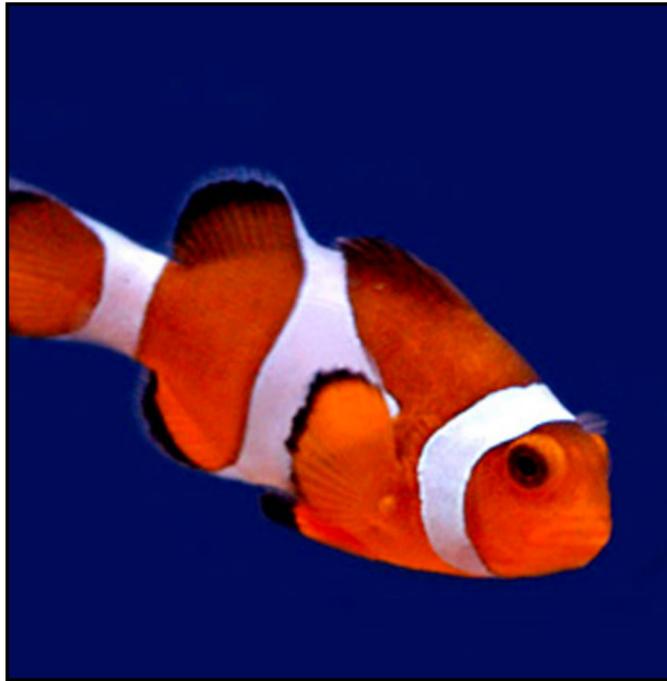


image x



"Fish"

label y

Image classification



image **x**



"Fish"

label **y**

Image classification



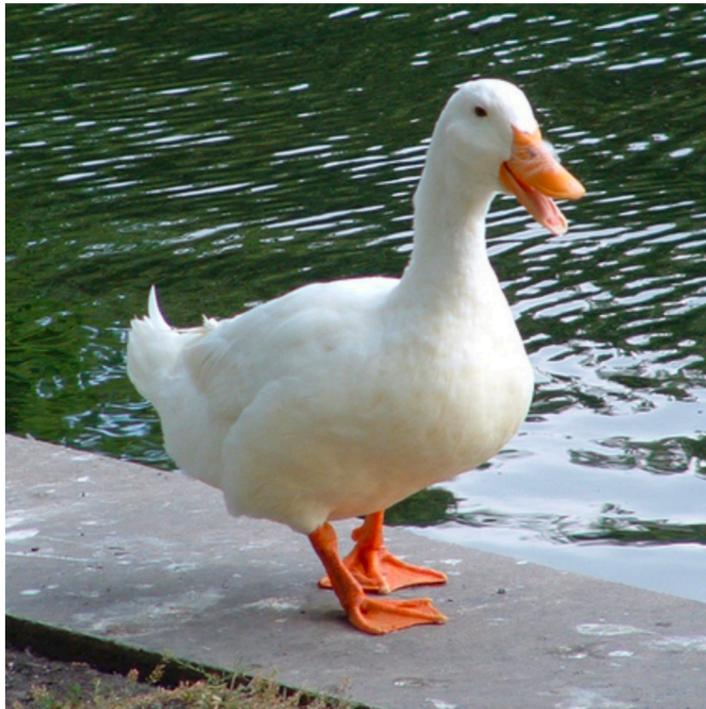
image **x**



"Fish"

label **y**

Image classification



⋮

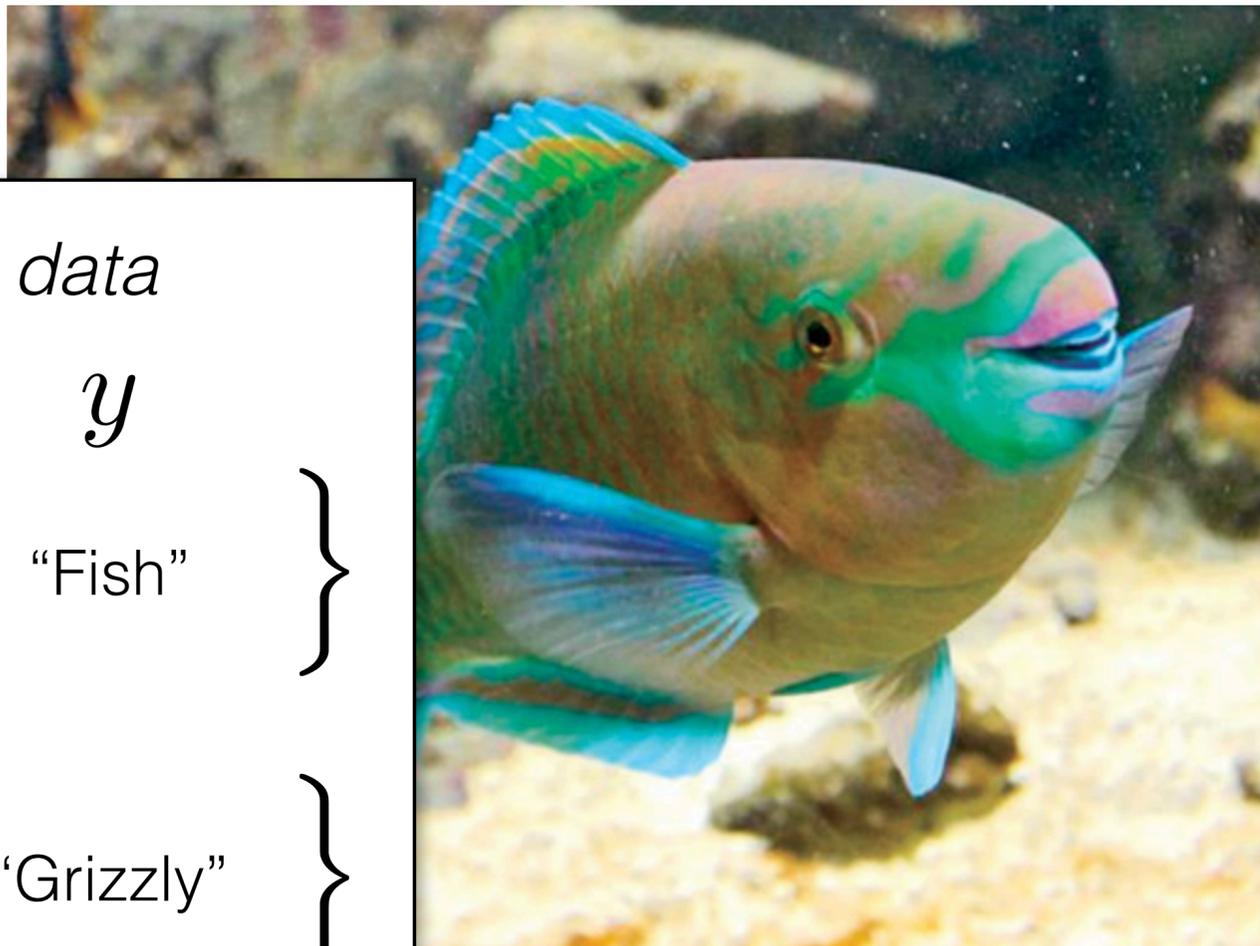
image \mathbf{x}



“Duck”

label y

\mathbf{x}



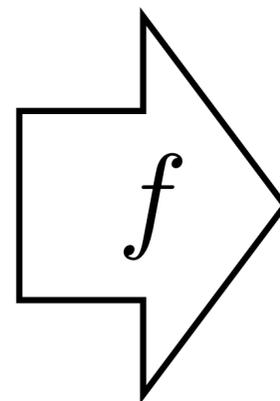
Training data

\mathbf{x}

y



⋮



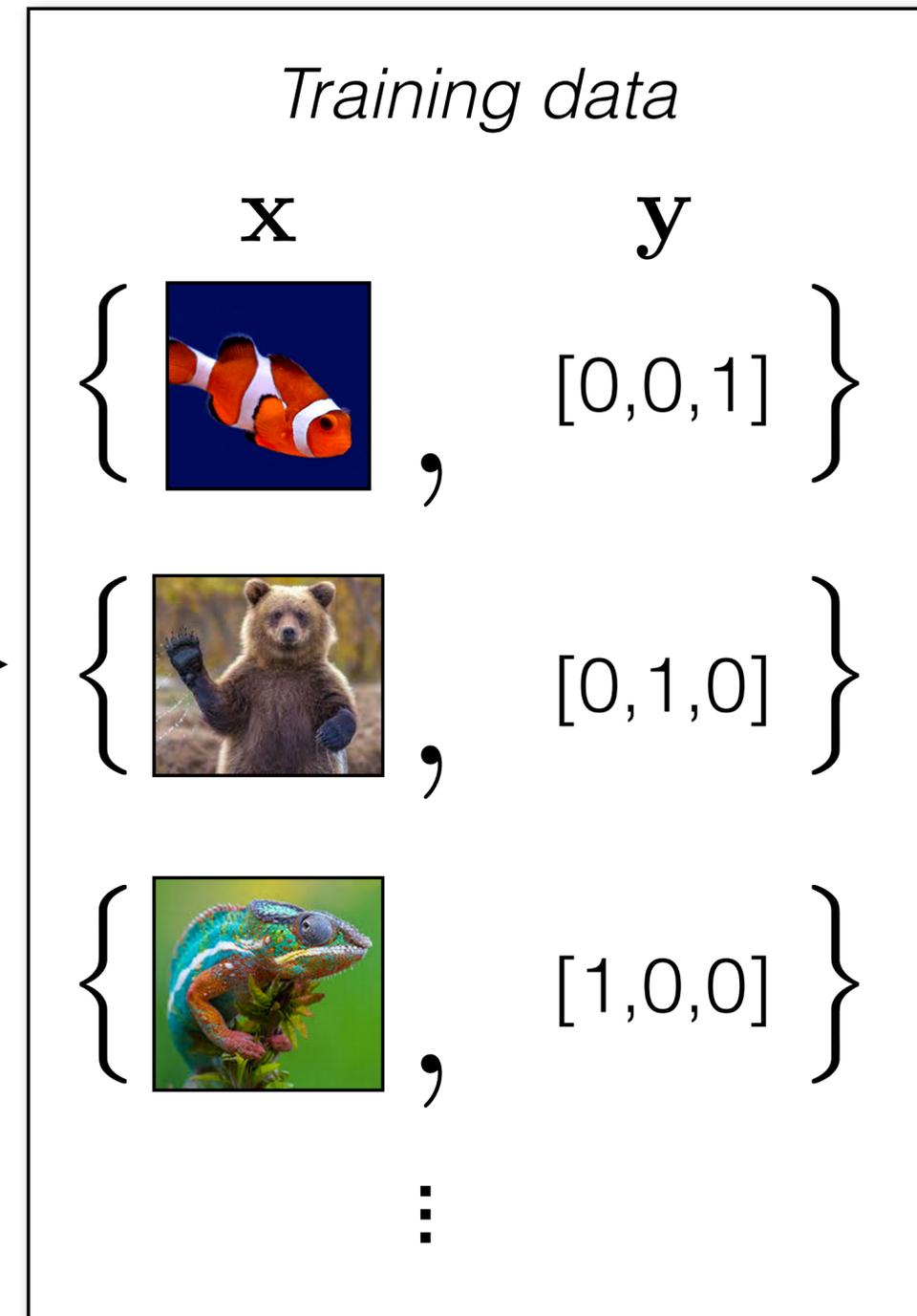
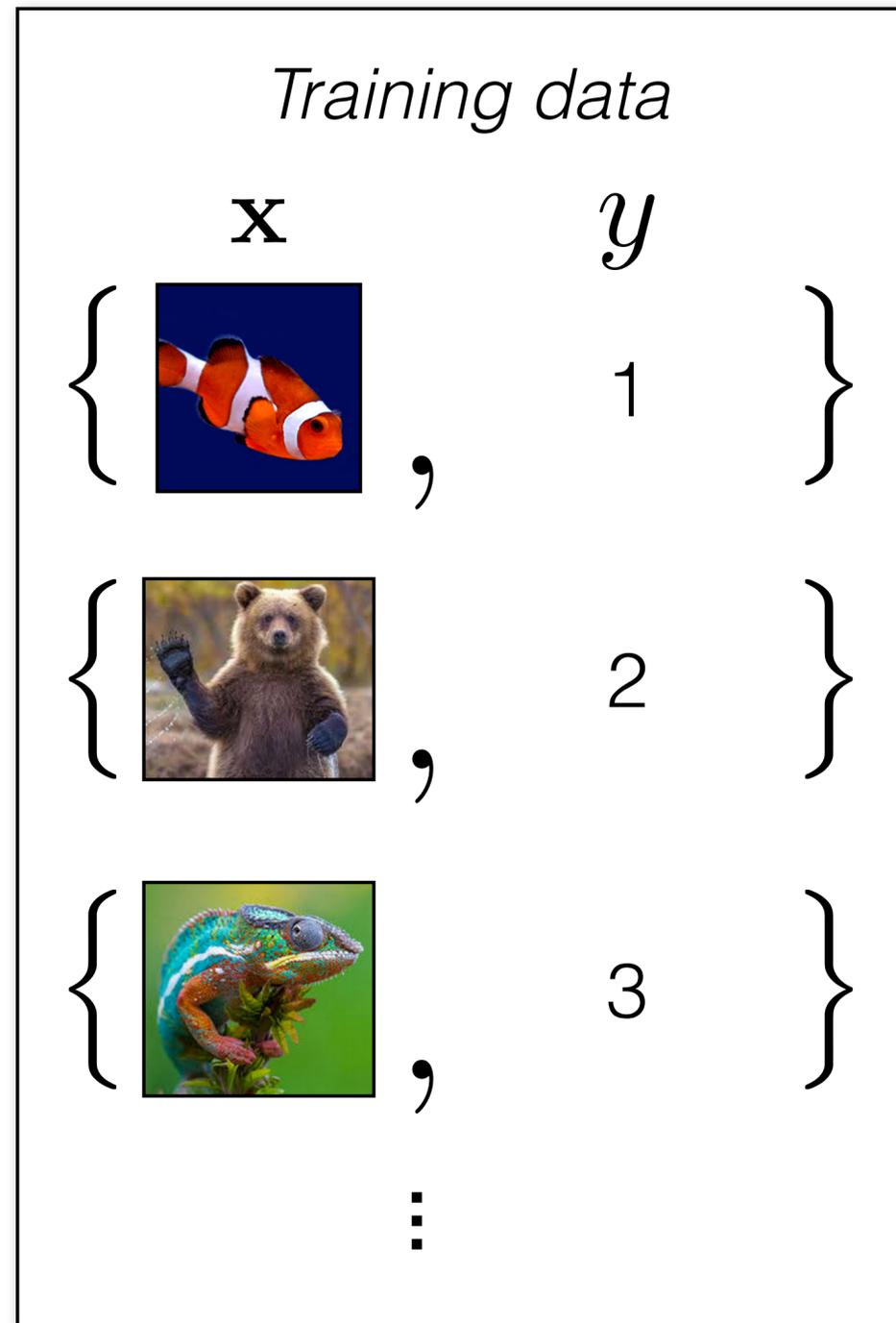
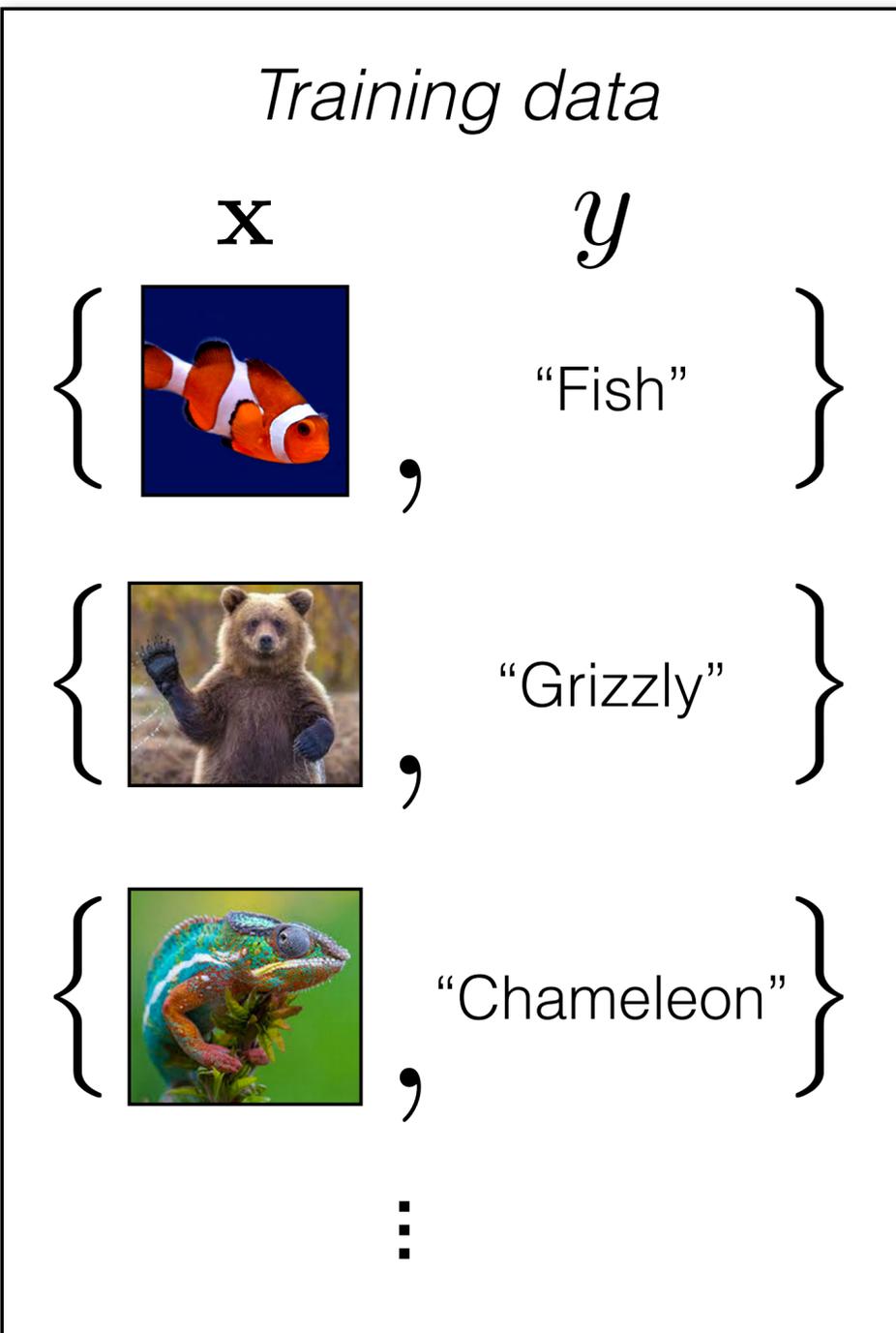
y

“Fish”

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

How to represent class labels?

One-hot vector



What should the loss be?

0-1 loss (number of misclassifications)

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

Cross entropy

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = H(\mathbf{y}, \hat{\mathbf{y}}) = - \sum_{k=1}^K y_k \log \hat{y}_k \quad \leftarrow \begin{array}{l} \text{continuous,} \\ \text{differentiable,} \\ \text{convex} \end{array}$$

Ground truth label y

x

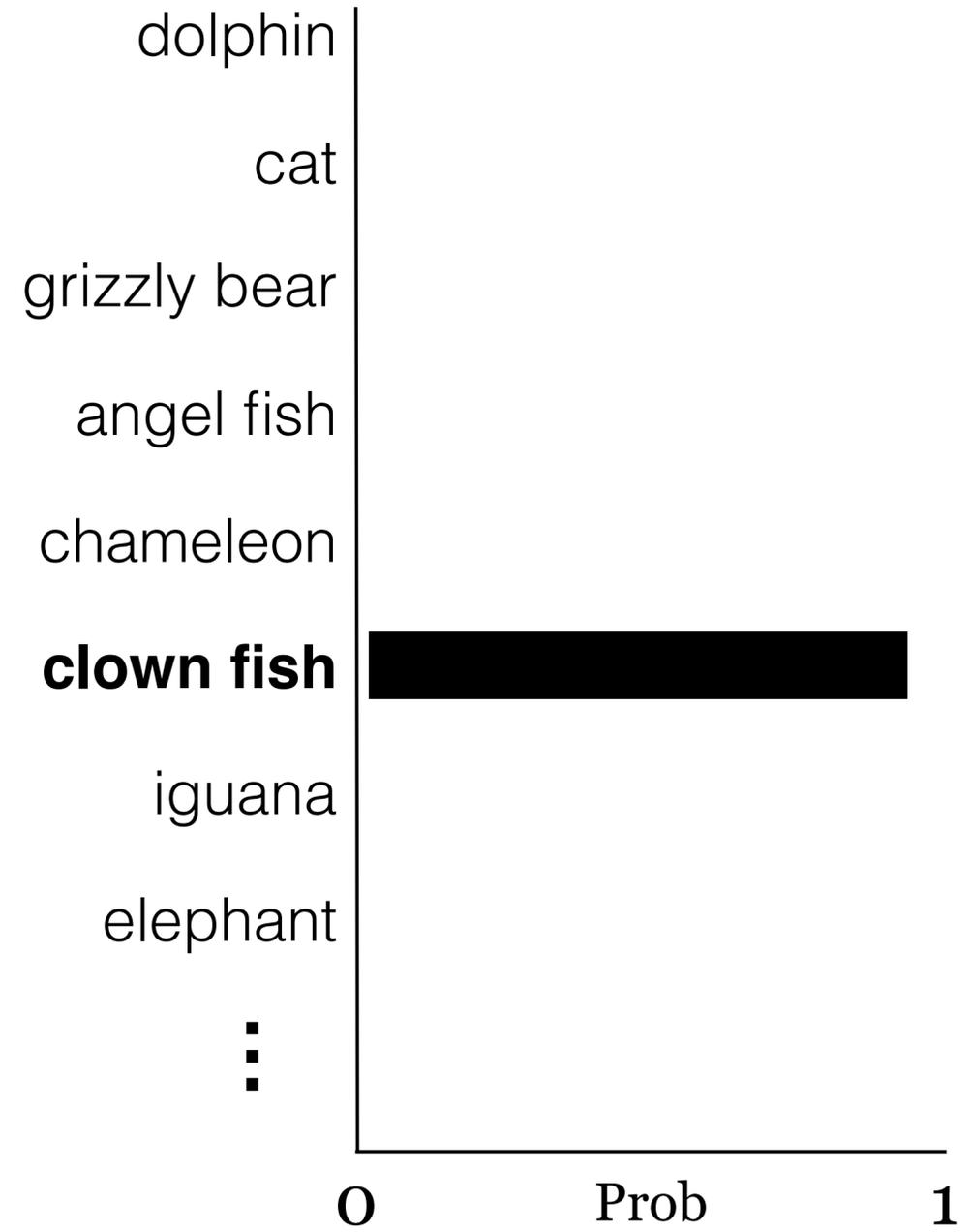


[0,0,0,0,0,1,0,0,...]

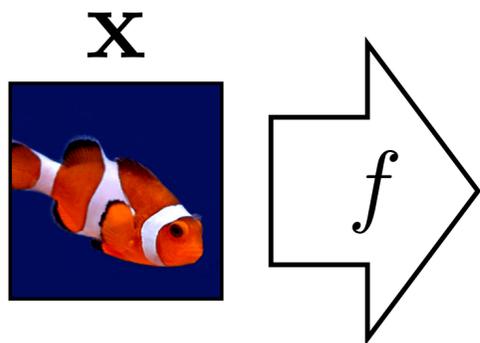


X

Ground truth label **y**



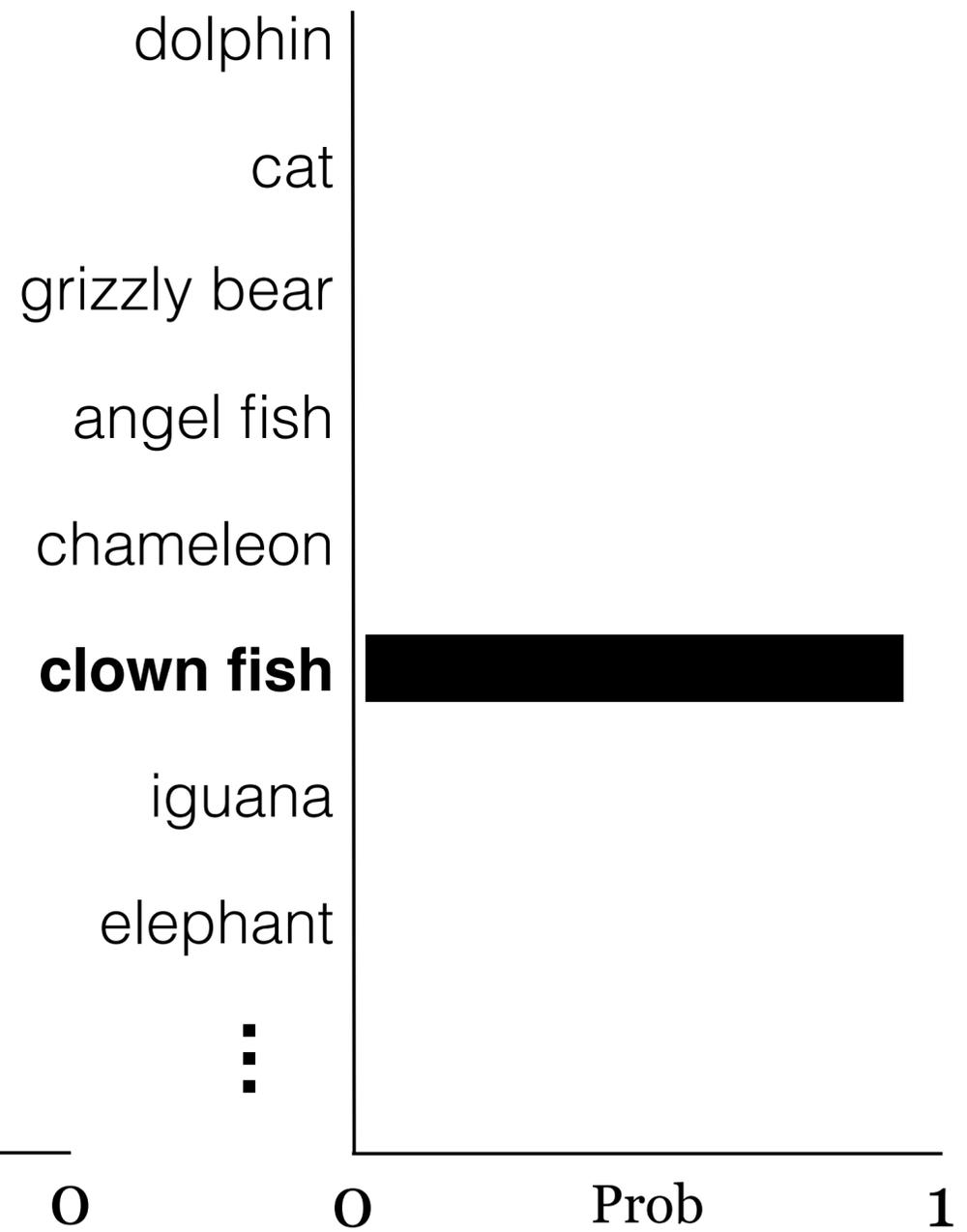
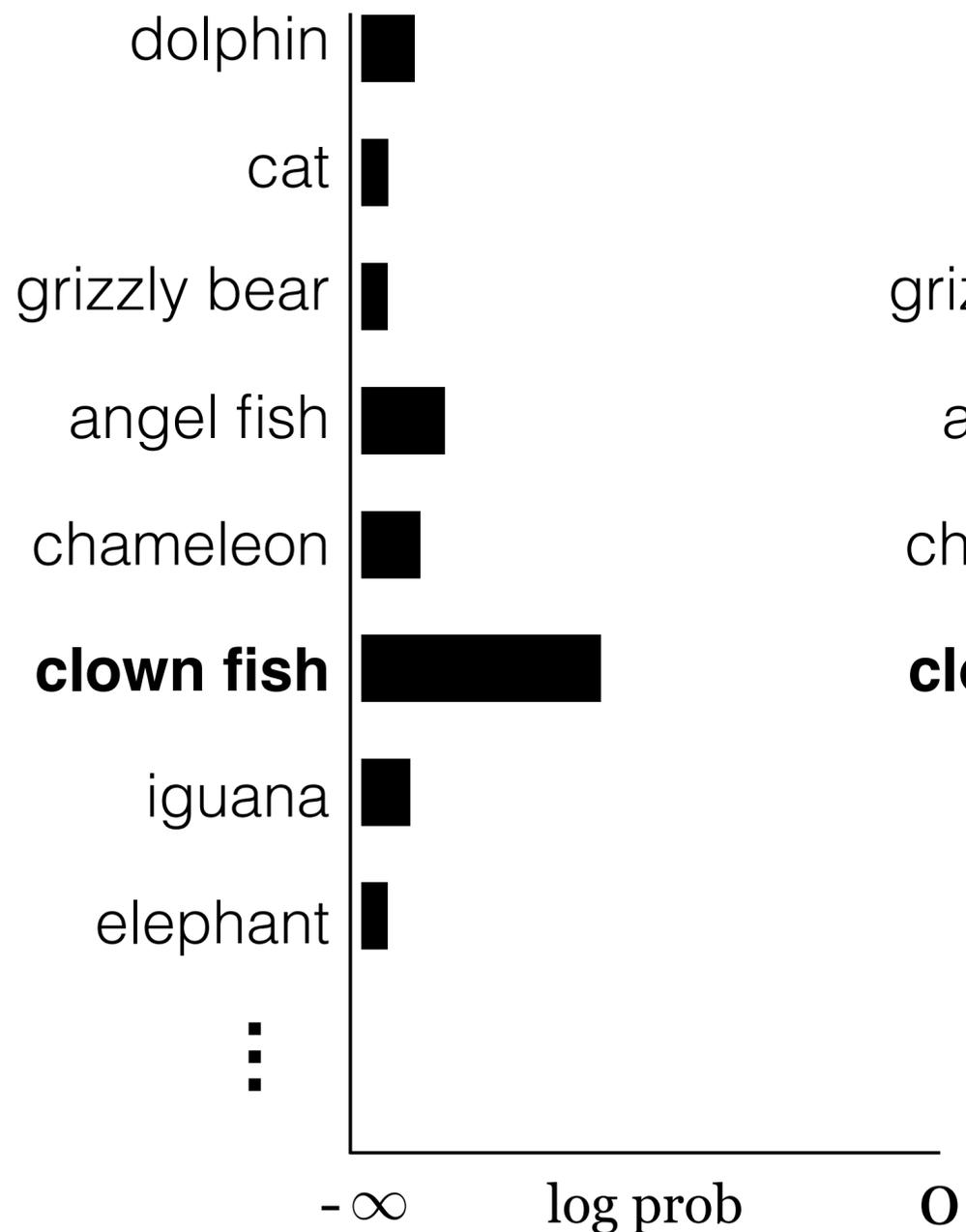
∞

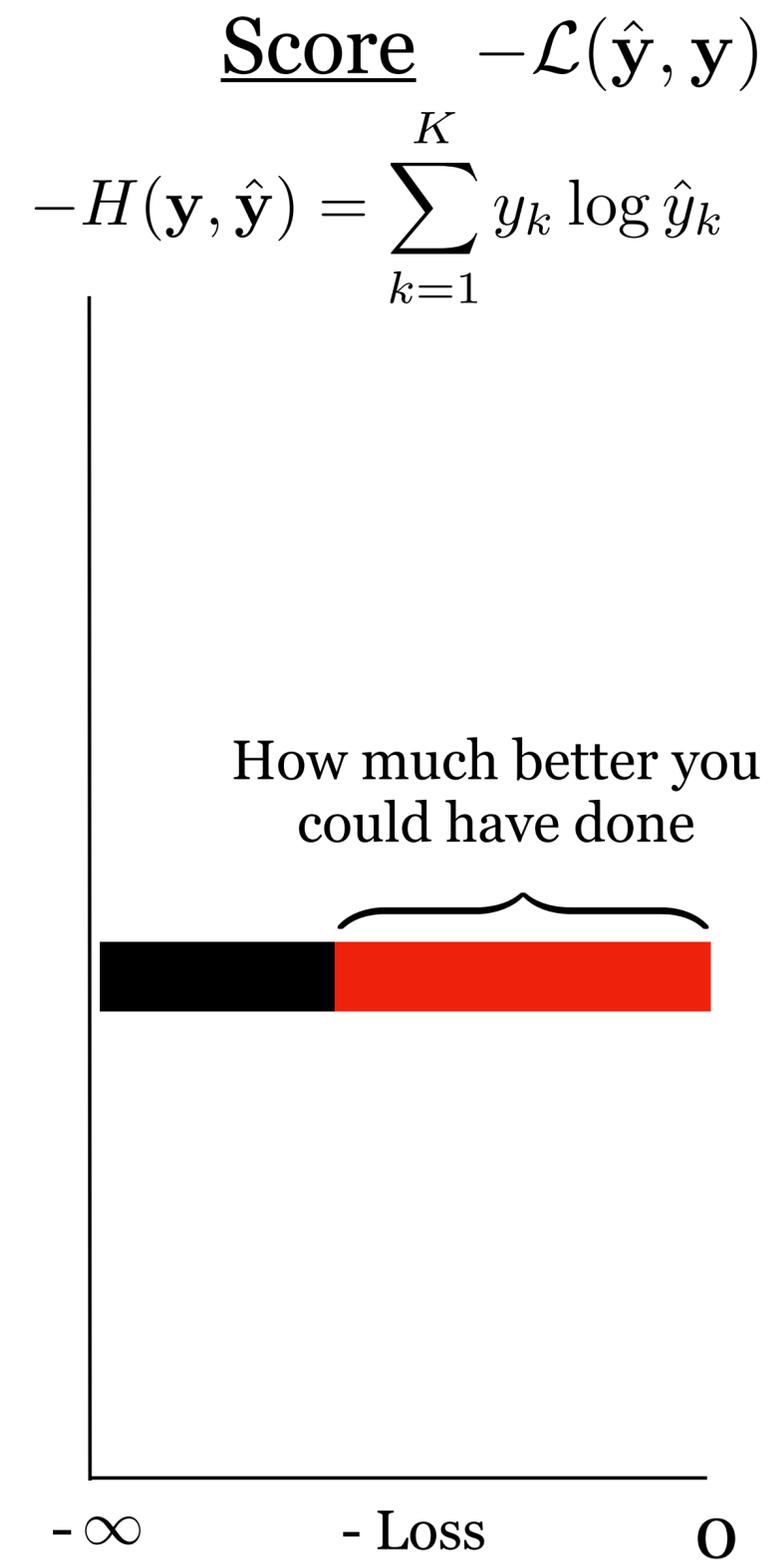
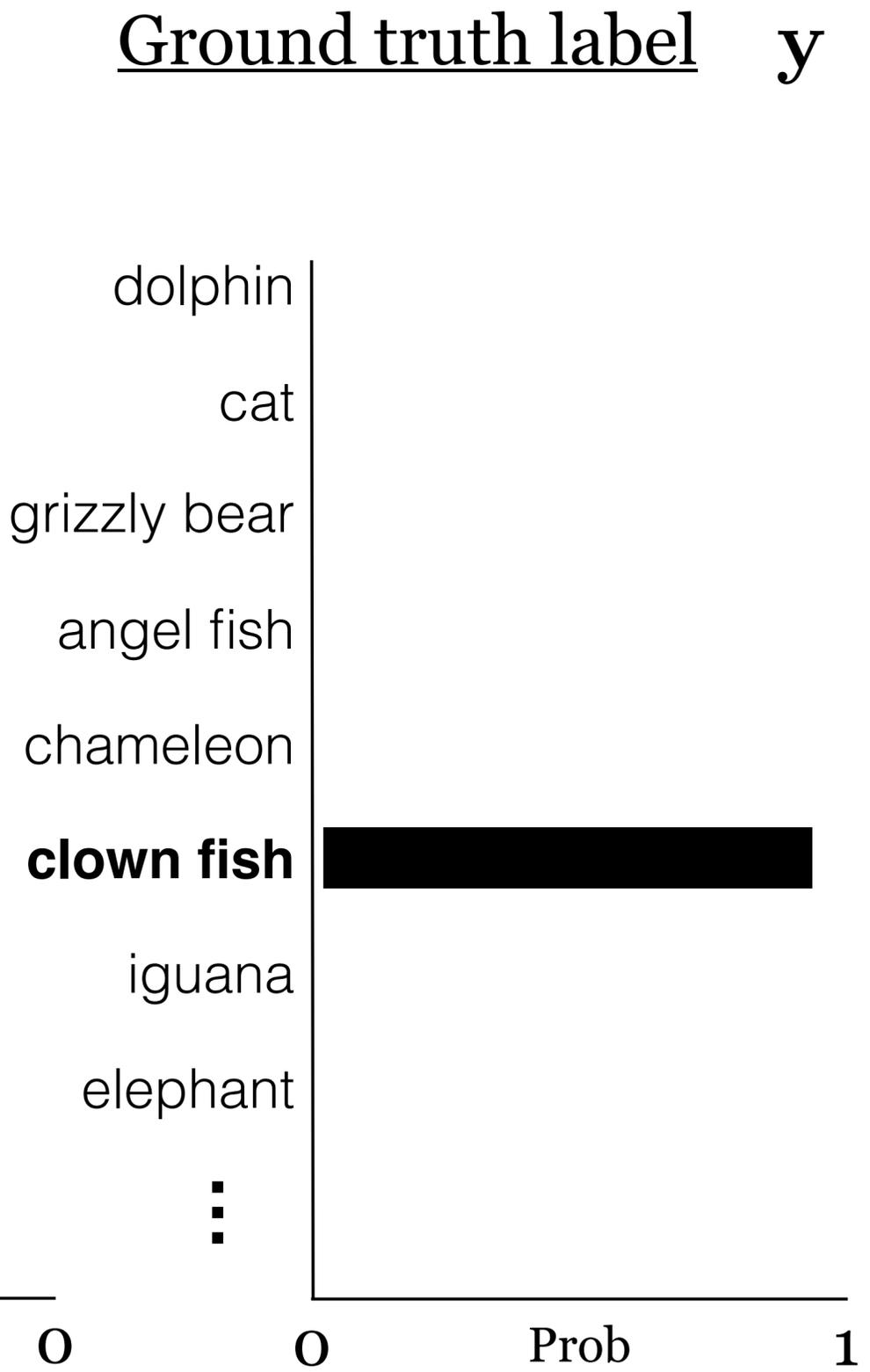
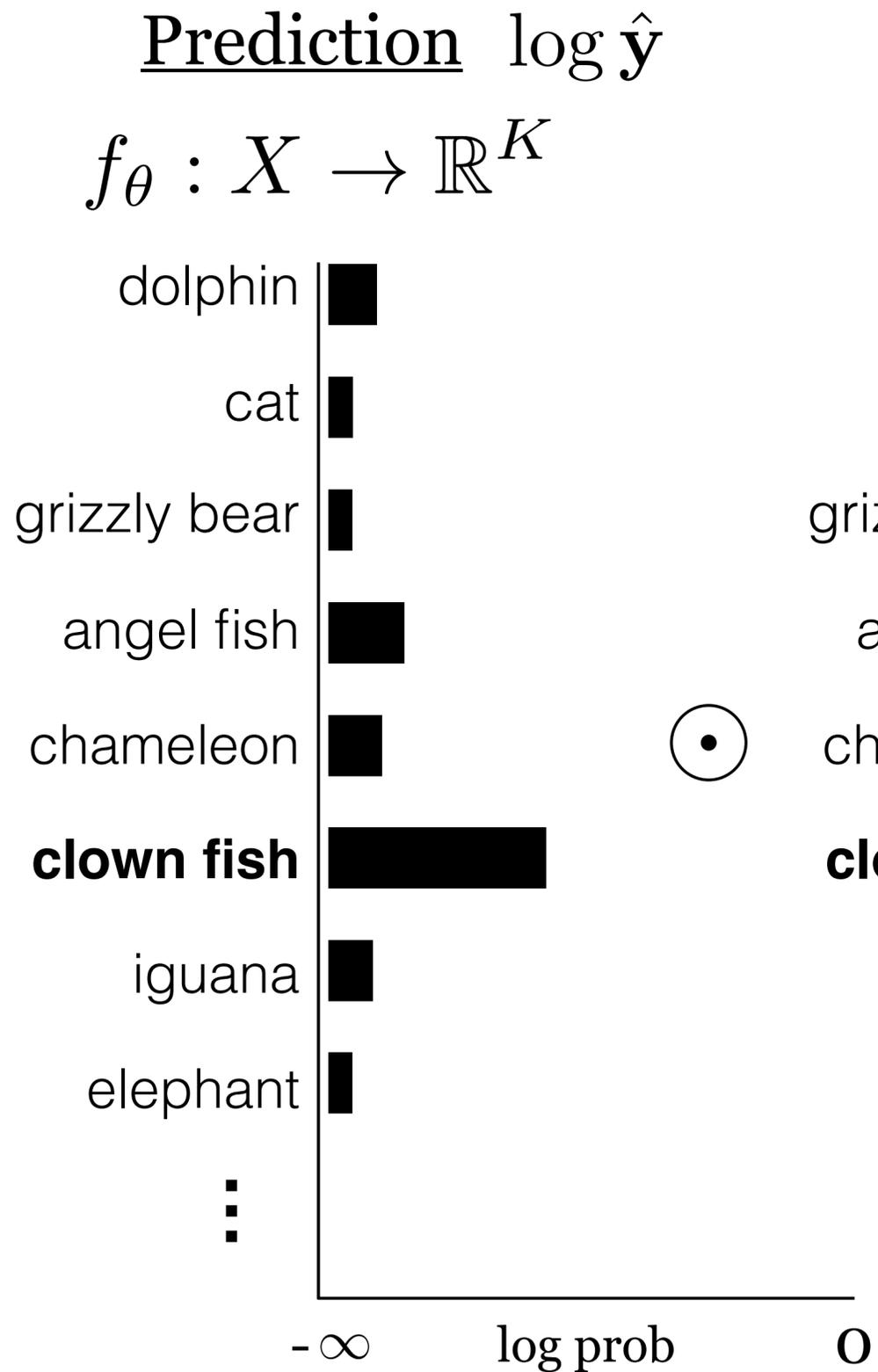
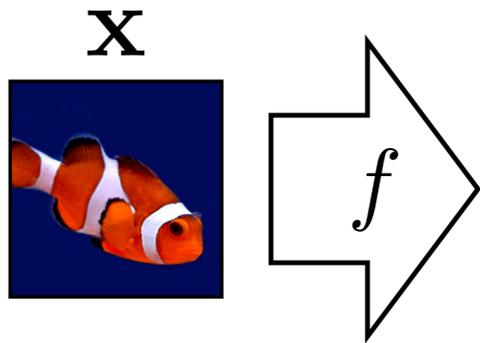


Prediction $\log \hat{y}$

Ground truth label y

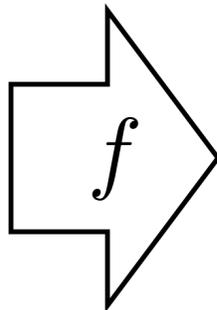
$$f_{\theta} : X \rightarrow \mathbb{R}^K$$







X



Prediction $\log \hat{y}$

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



Ground truth label **y**



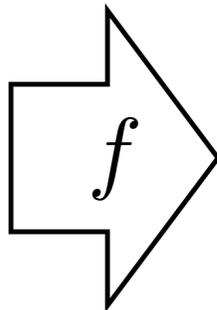
Score $-\mathcal{L}(\hat{y}, y)$

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



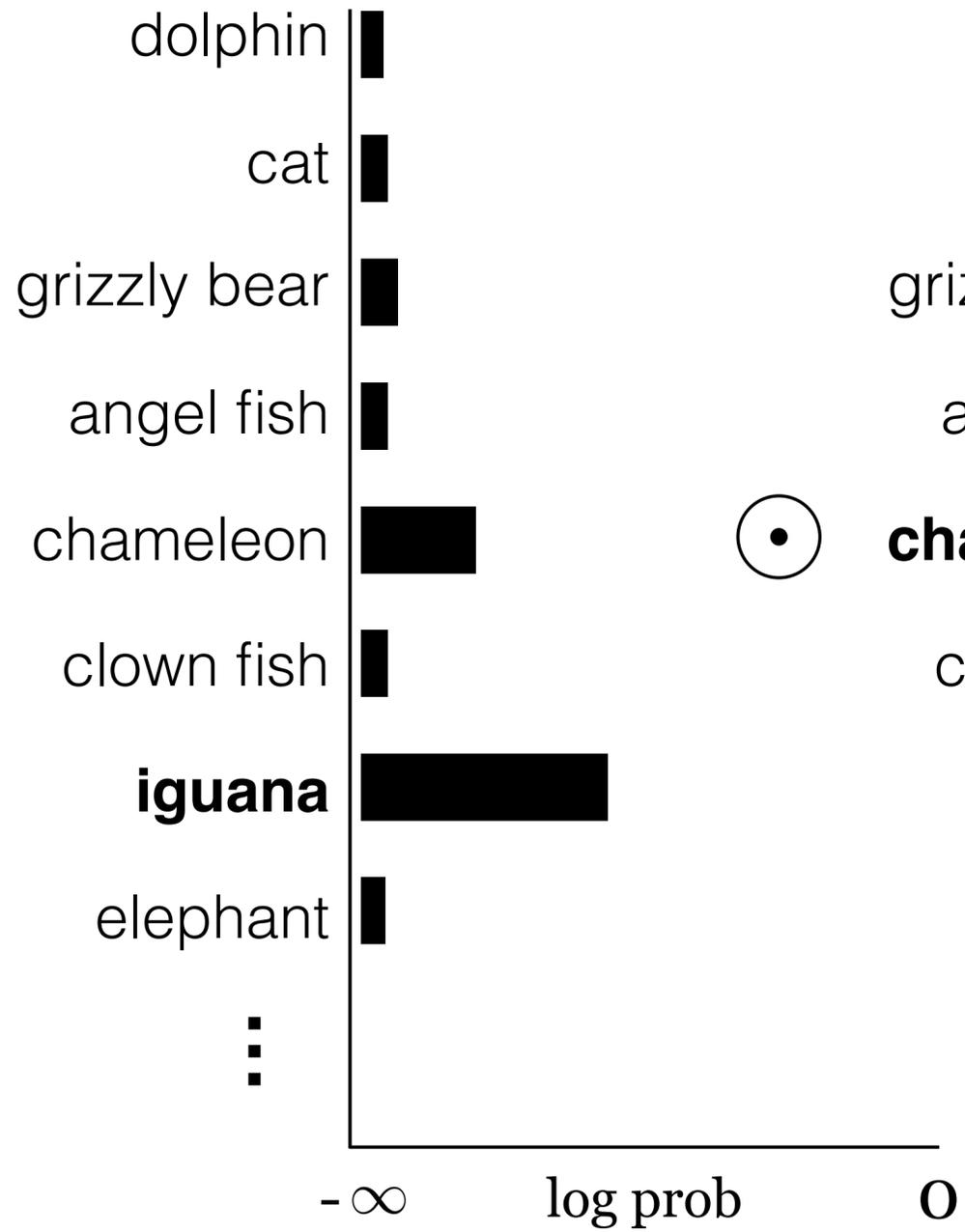


X

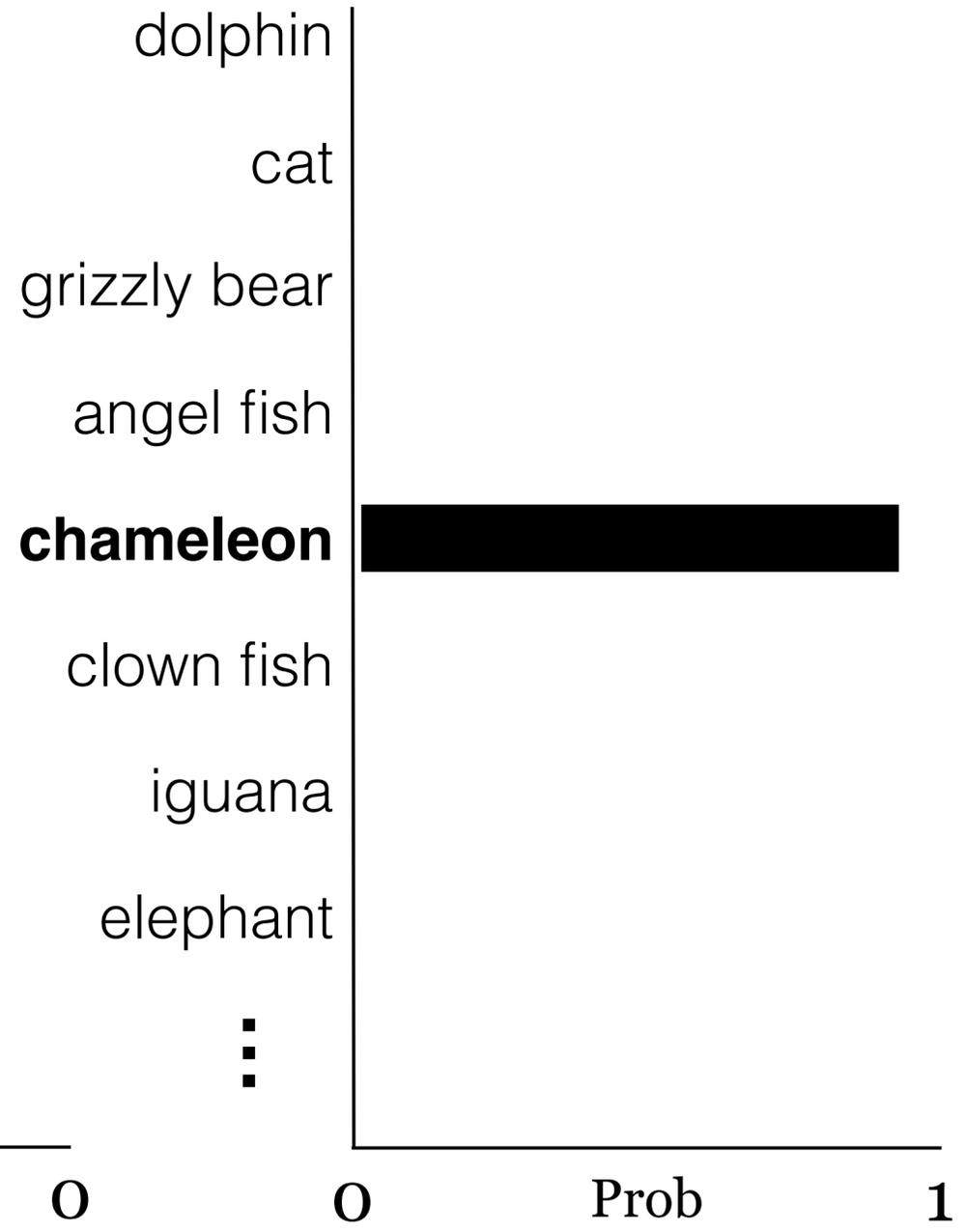


Prediction $\log \hat{y}$

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

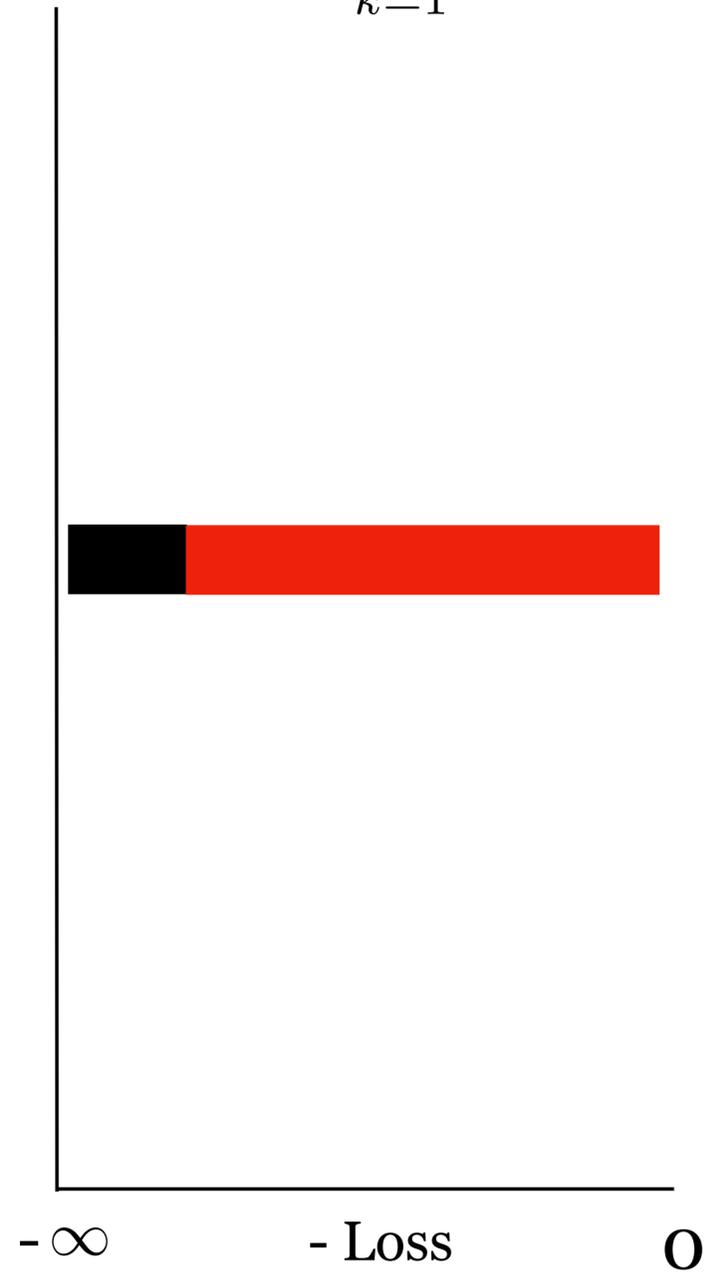


Ground truth label **y**



Score $-\mathcal{L}(\hat{y}, y)$

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



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

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

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

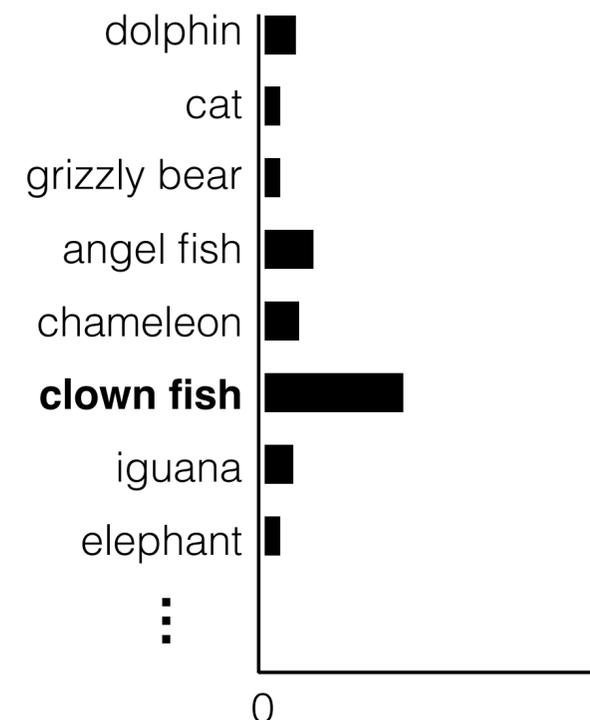
← **logits**: vector of K scores, one for each class

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

← 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 \mathbf{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!

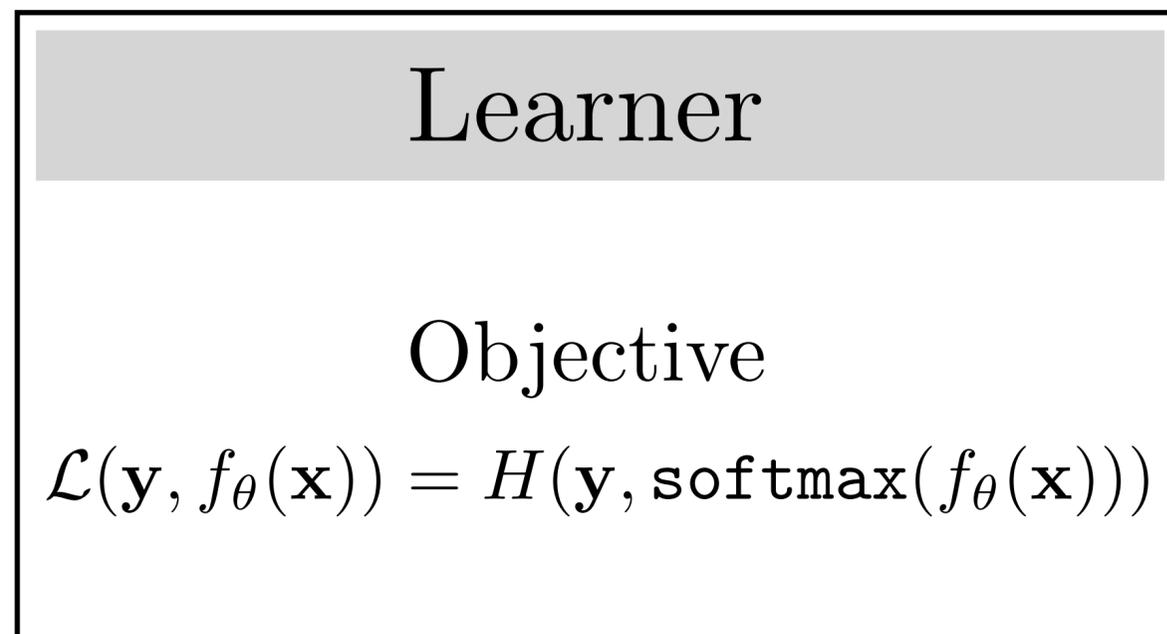
Softmax regression (a.k.a. multinomial logistic regression)

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

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

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

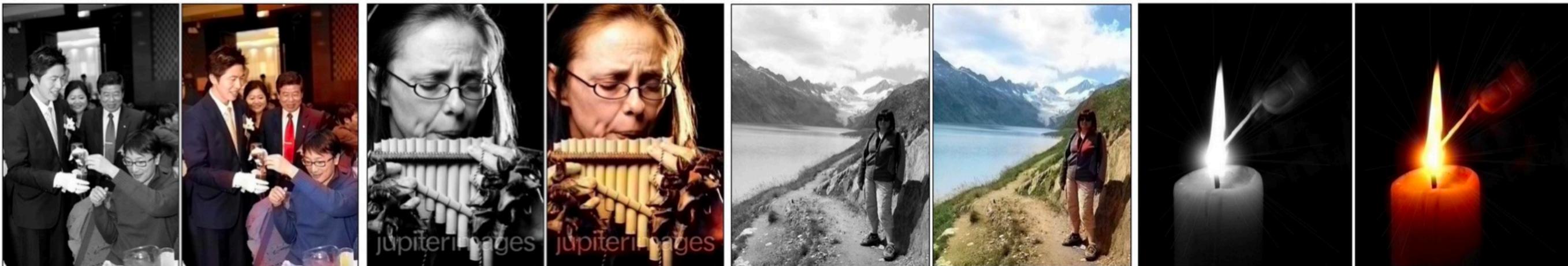
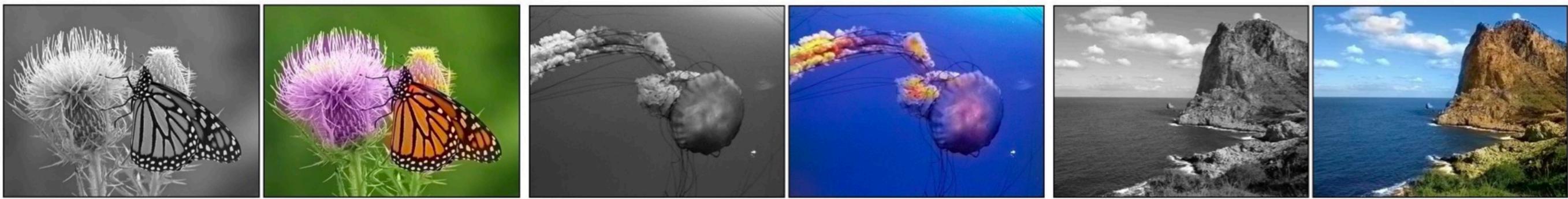
Data
 $\{x^{(i)}, y^{(i)}\}_{i=1}^N \rightarrow$



$\rightarrow f$

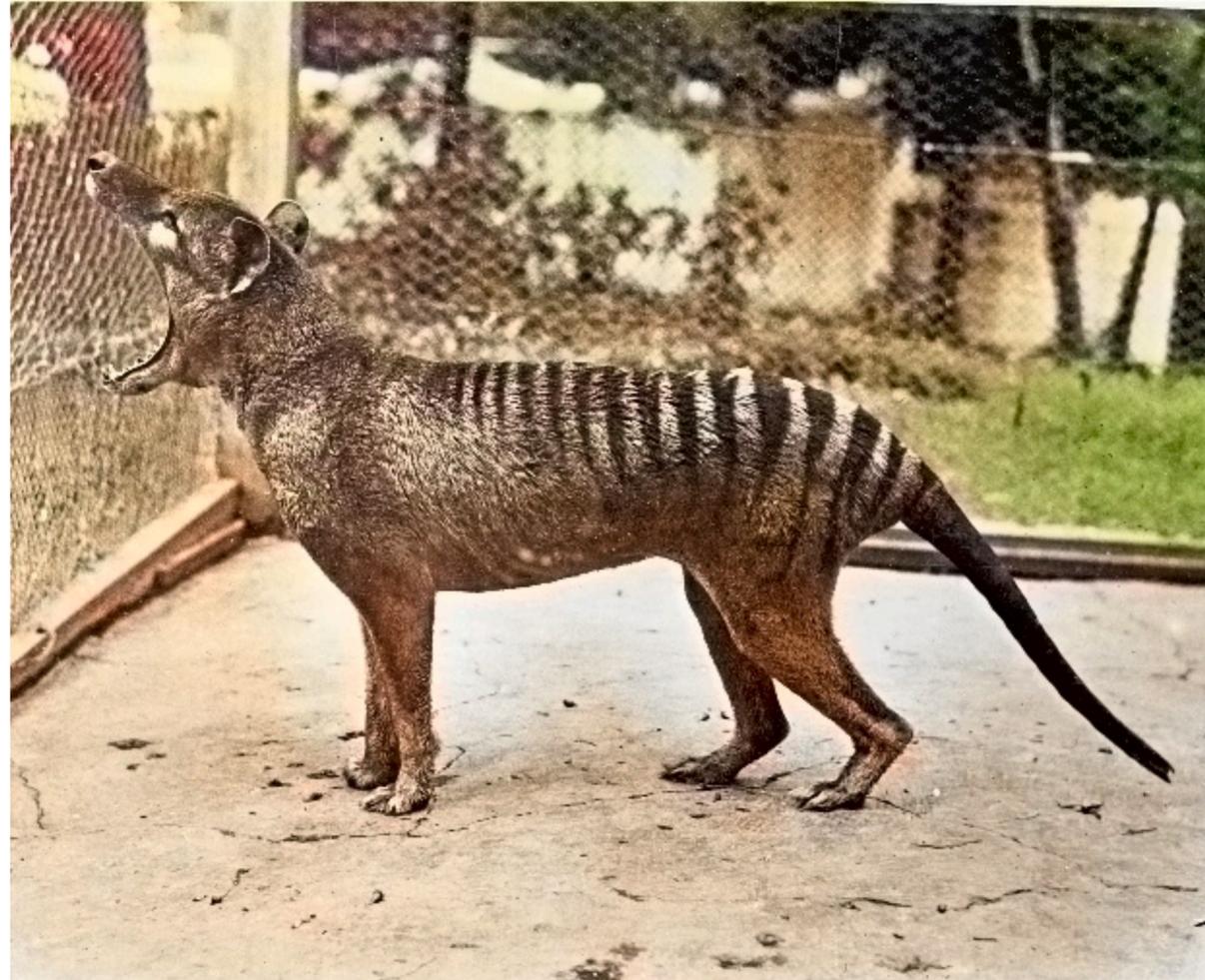
The Problem of Generalization



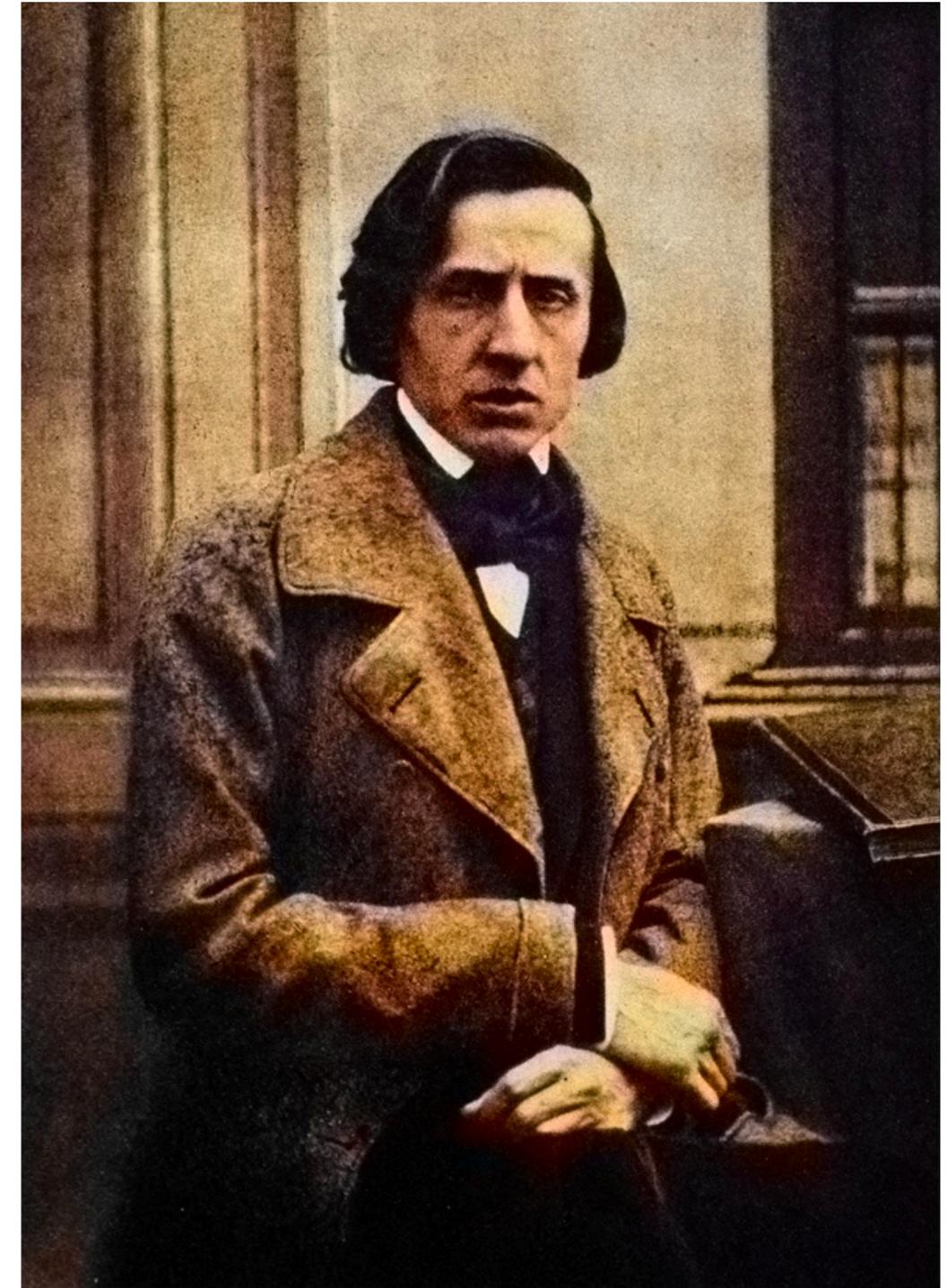




u/Rafael_P_S



Thylacine

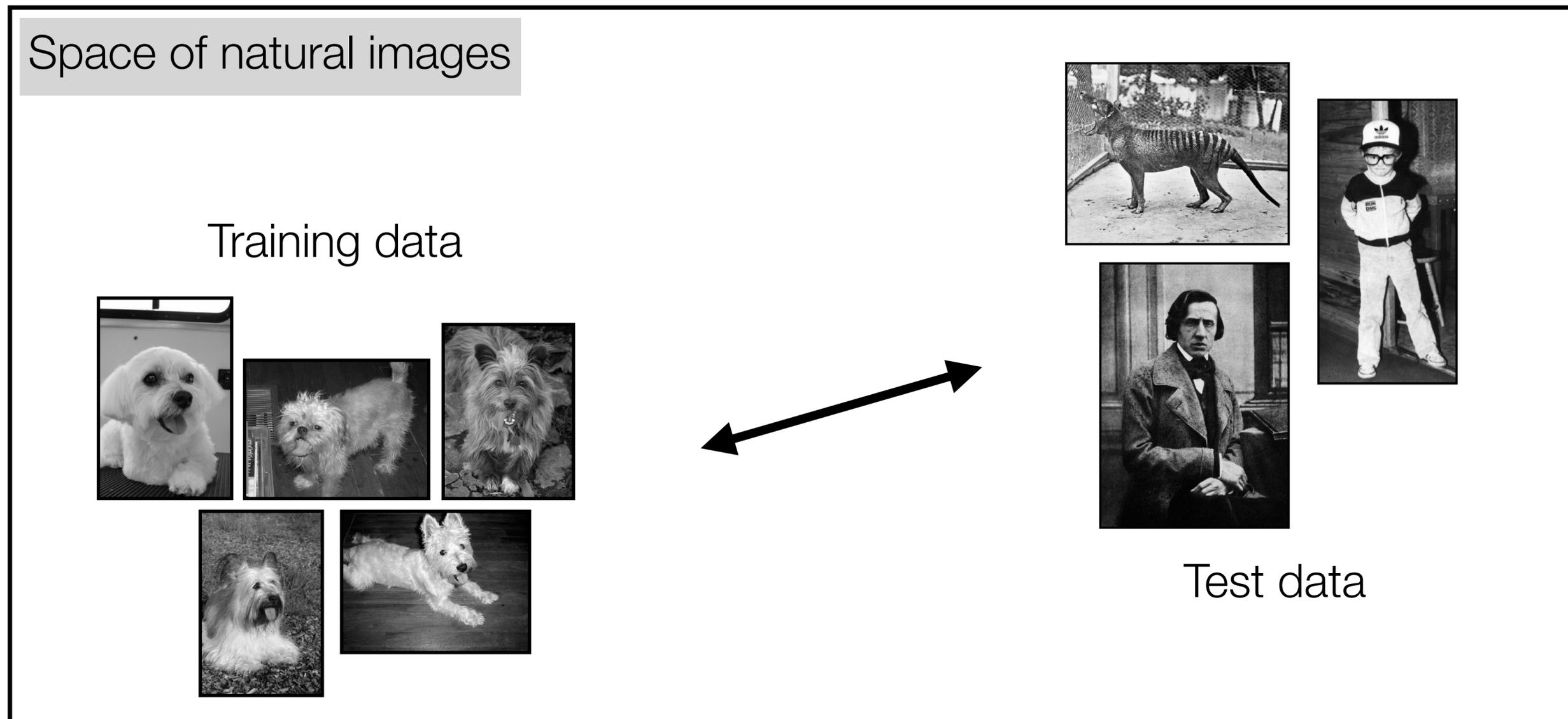


Chopin

training domain

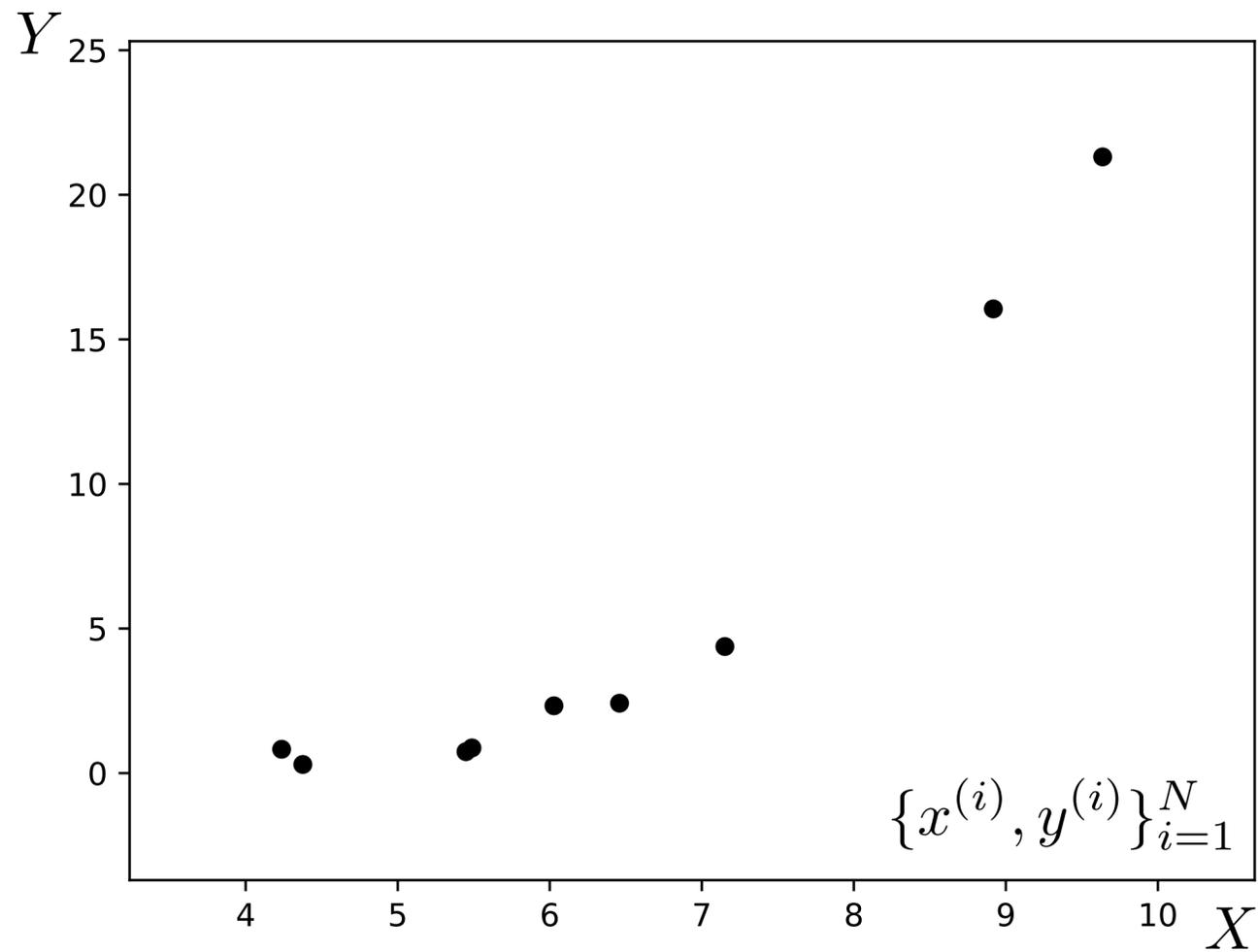
testing domain
(where we actual use our model)

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



Linear regression

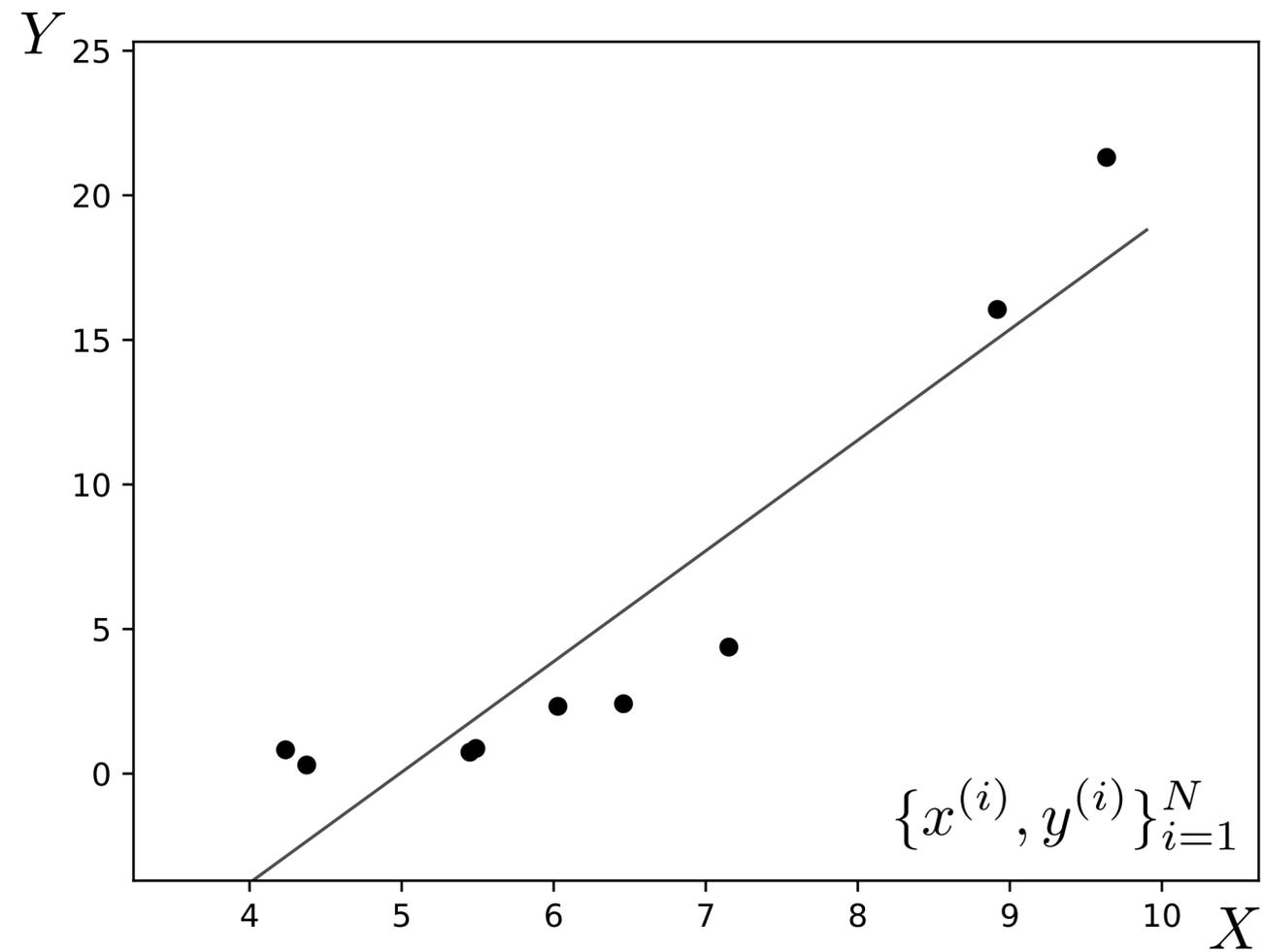
Training data



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

Linear regression

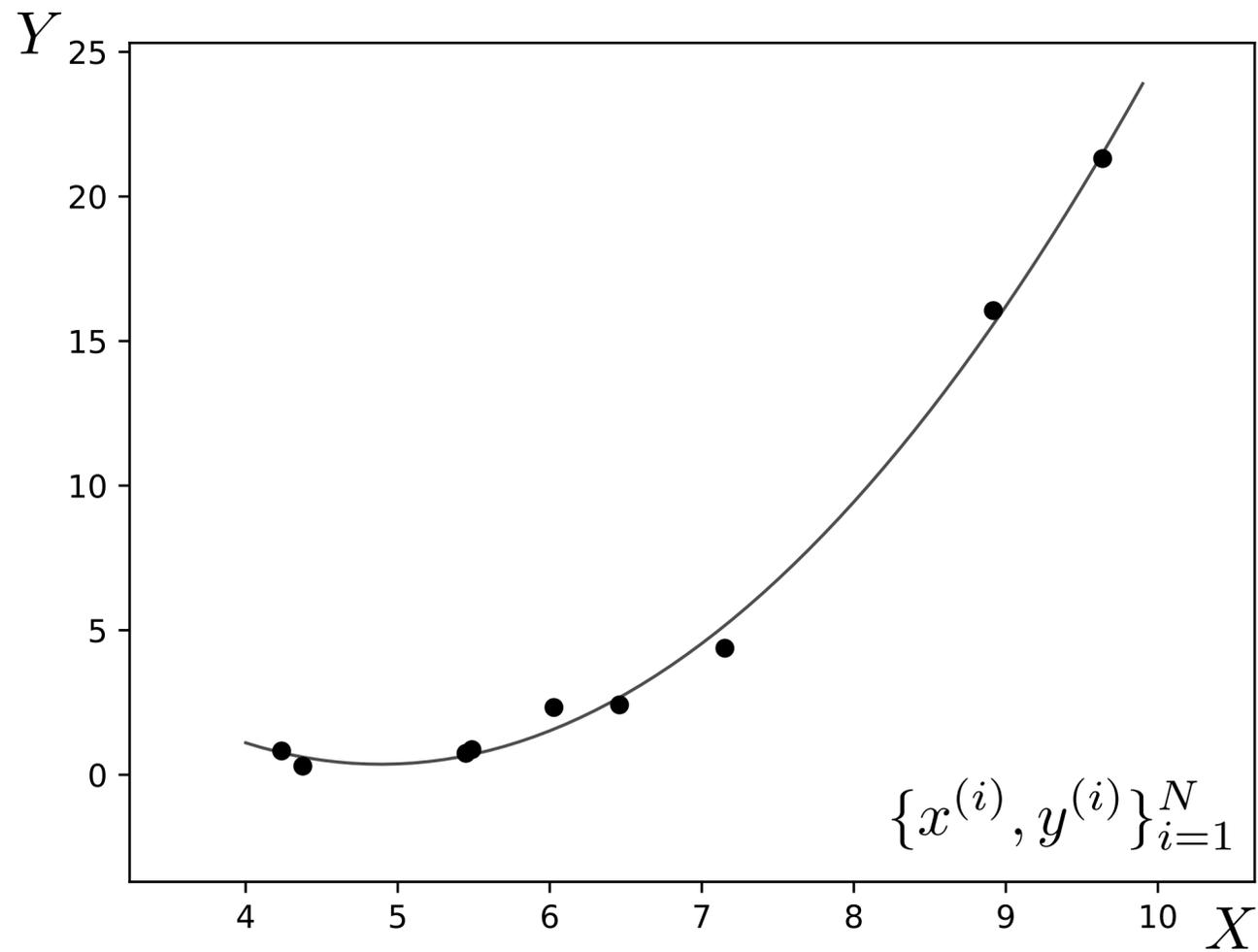
Training data



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

Polynomial regression

Training data

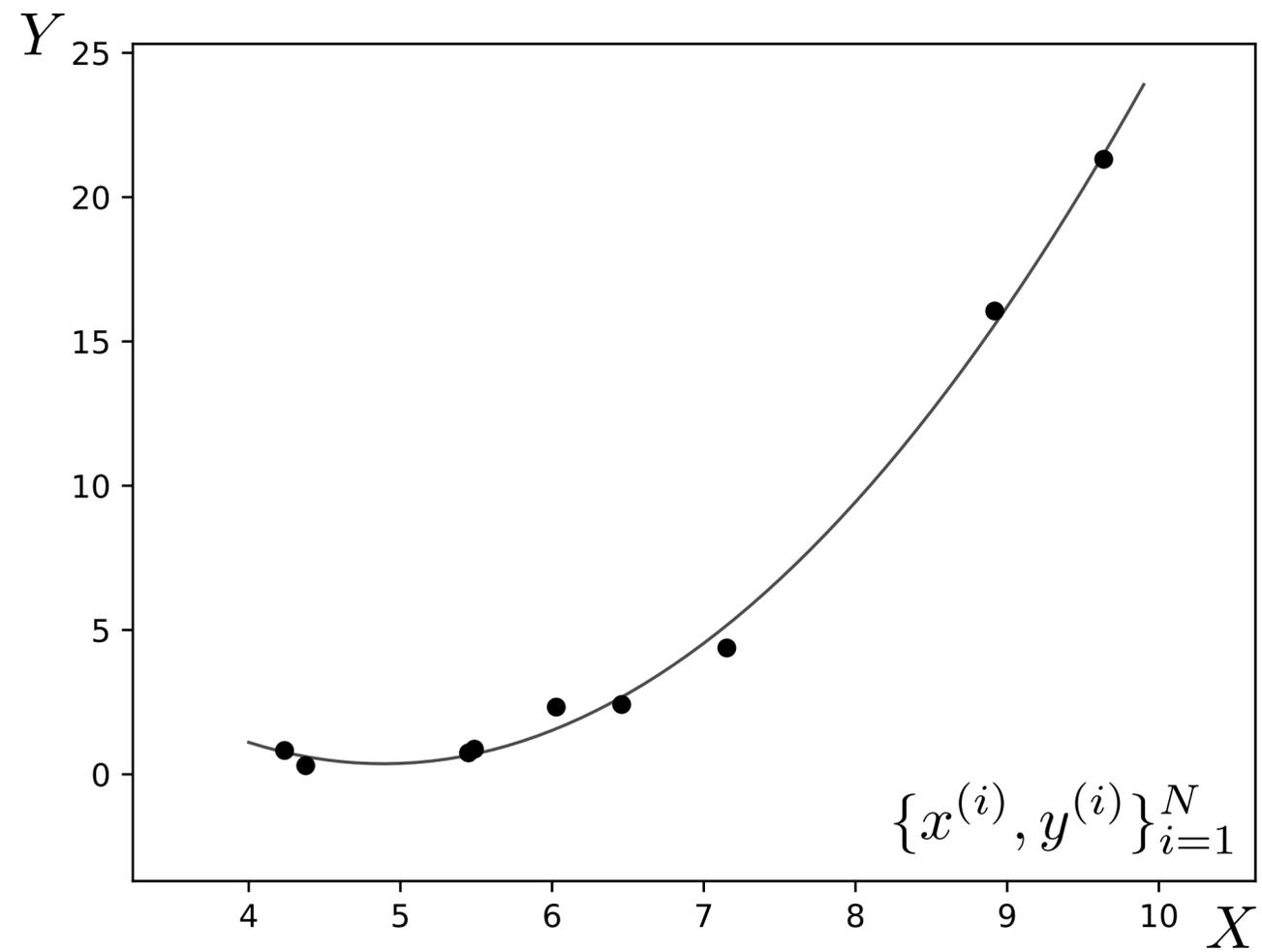


$$f_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

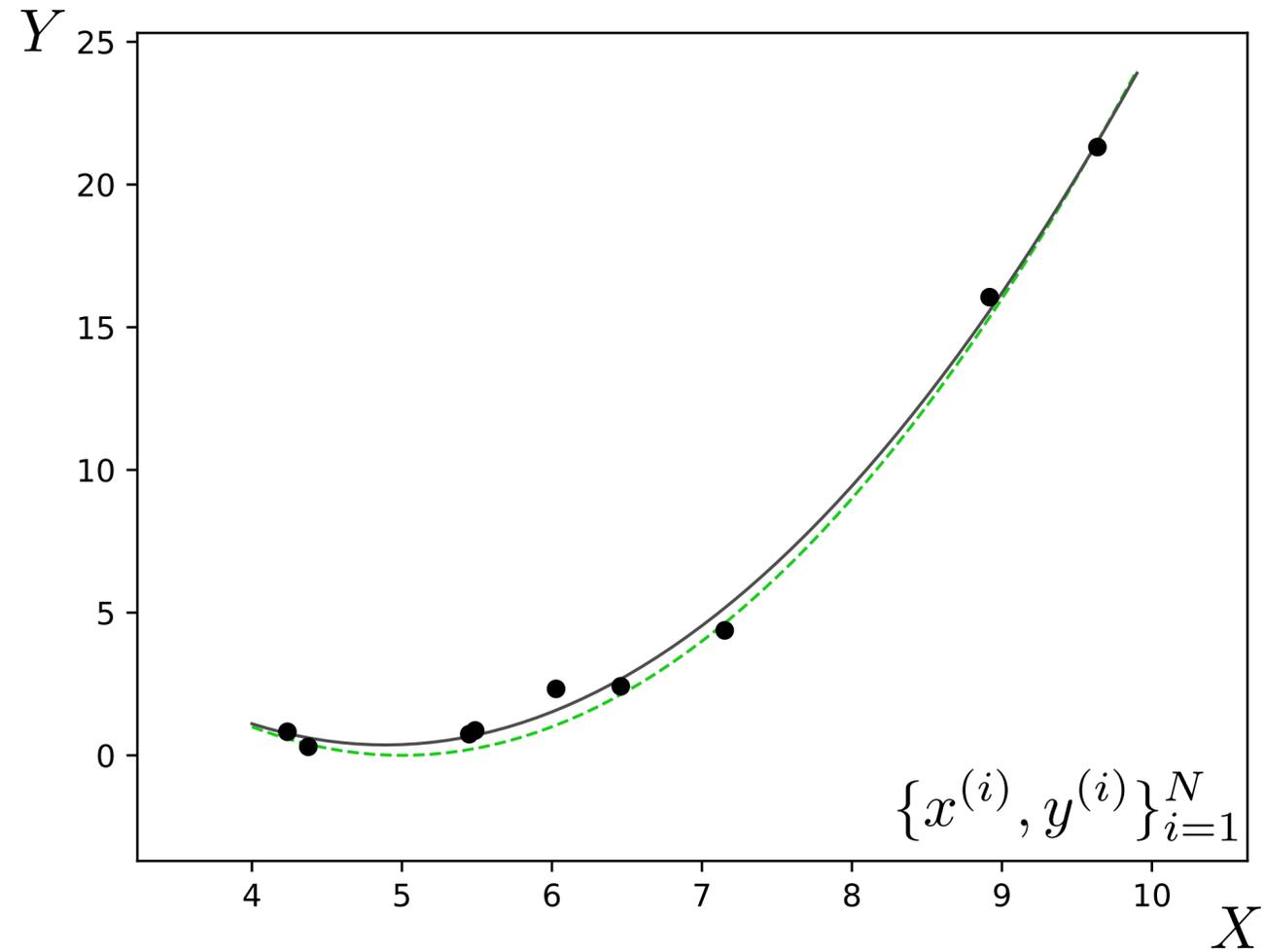
$$f_{\theta}(x) = \sum_{k=0}^K \theta_k x^k$$

K-th degree polynomial regression

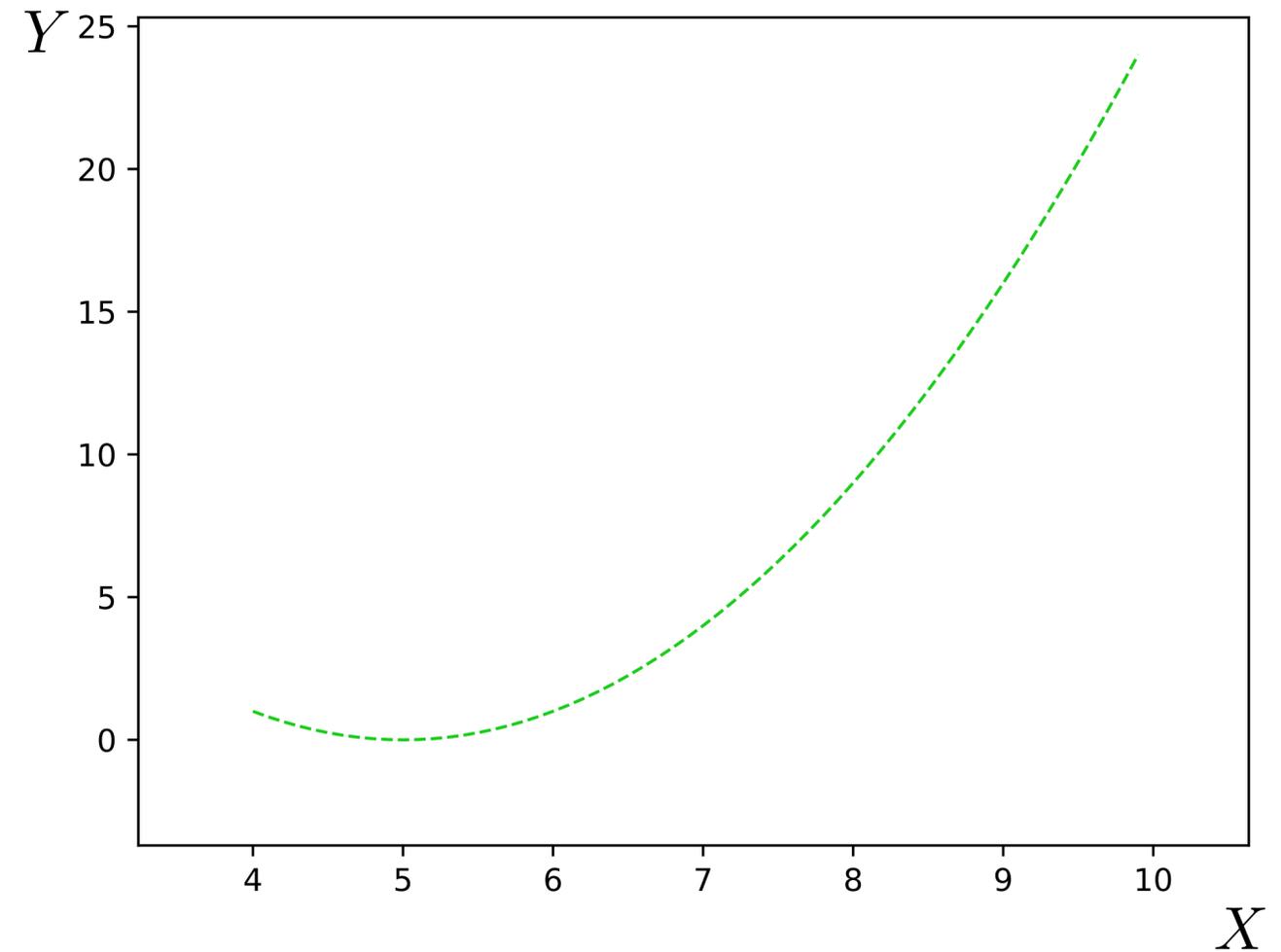
Training data



Training data



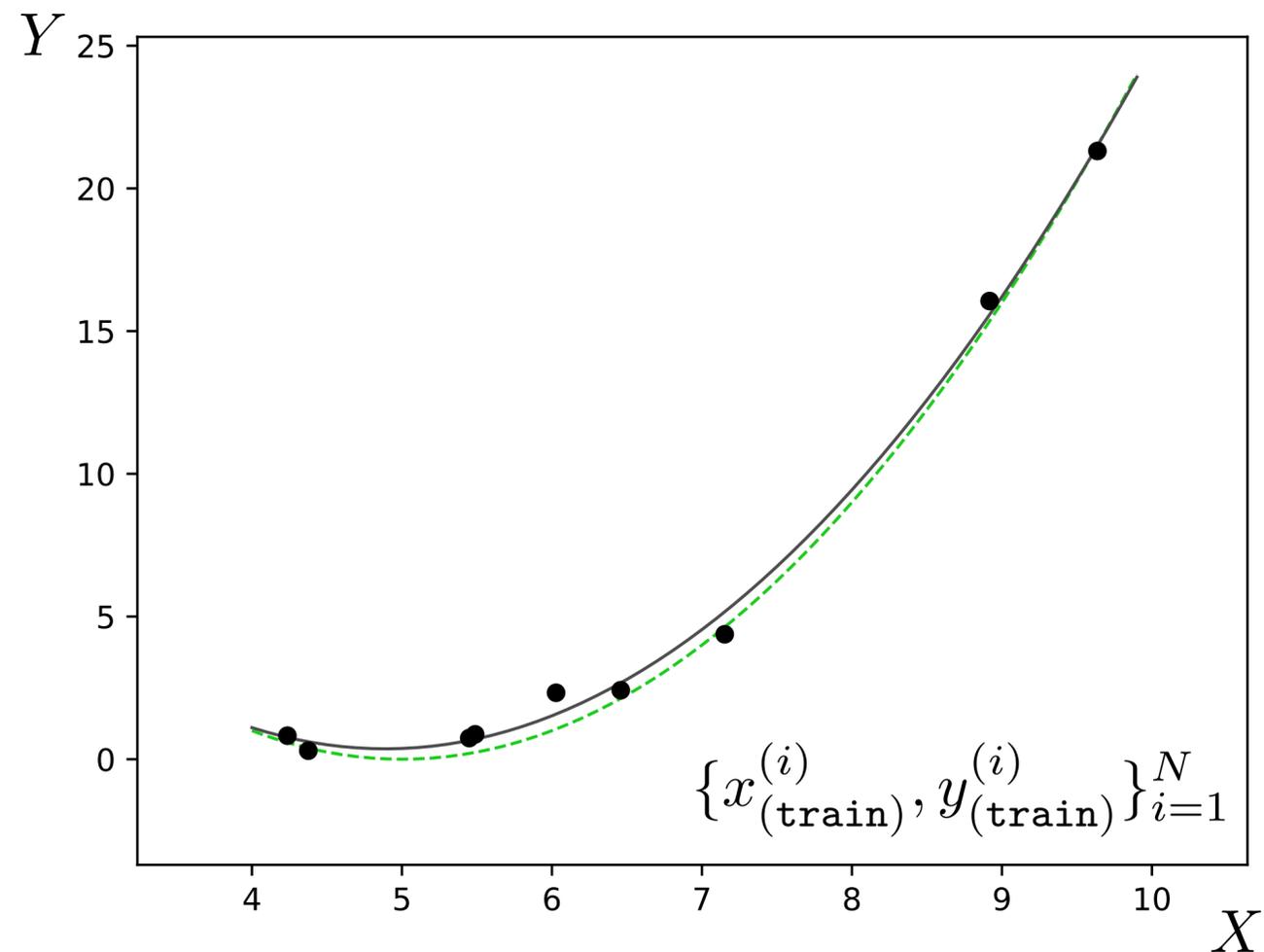
Test data



True data-generating process

p_{data}

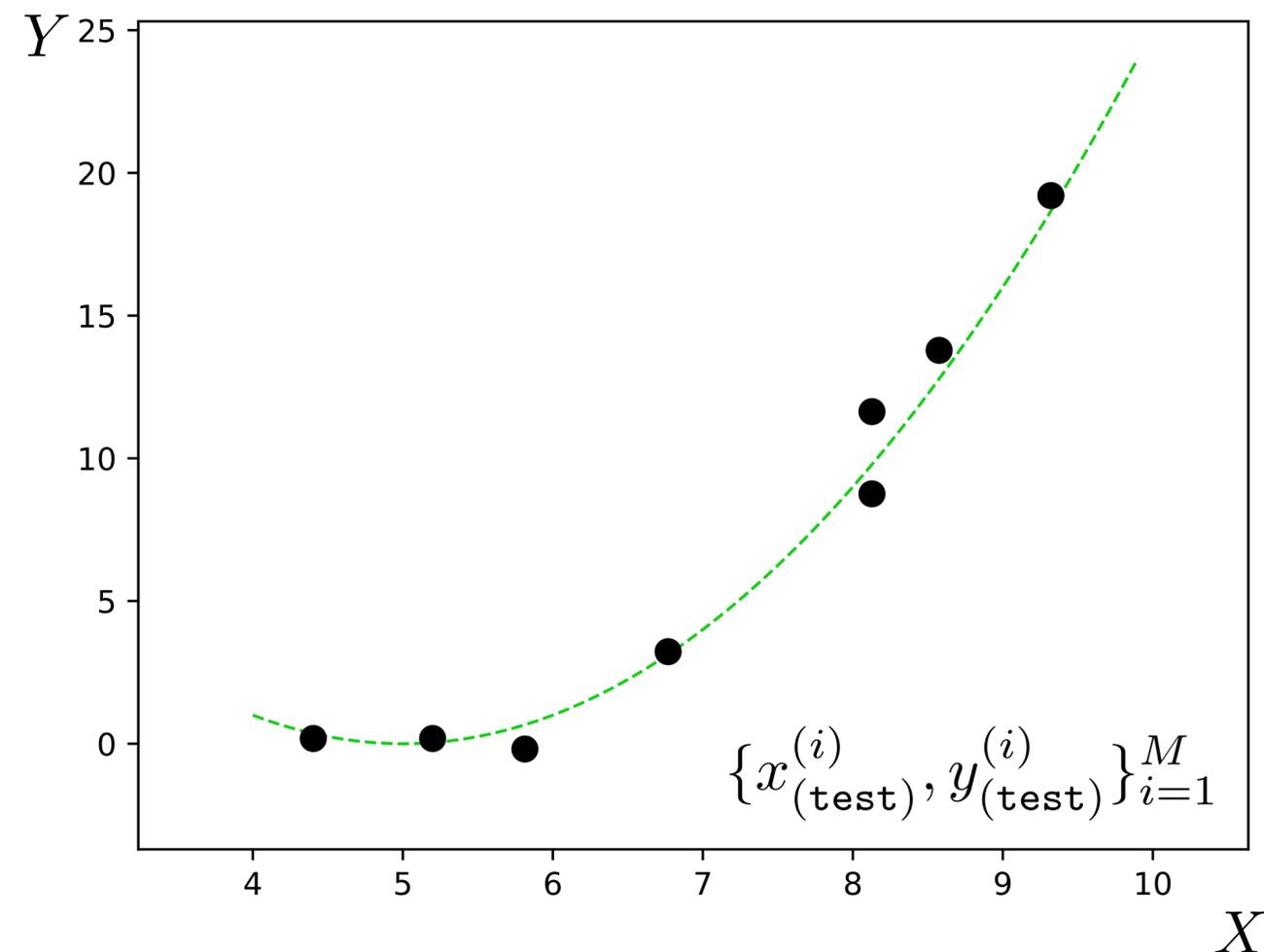
Training data



True data-generating process

p_{data}

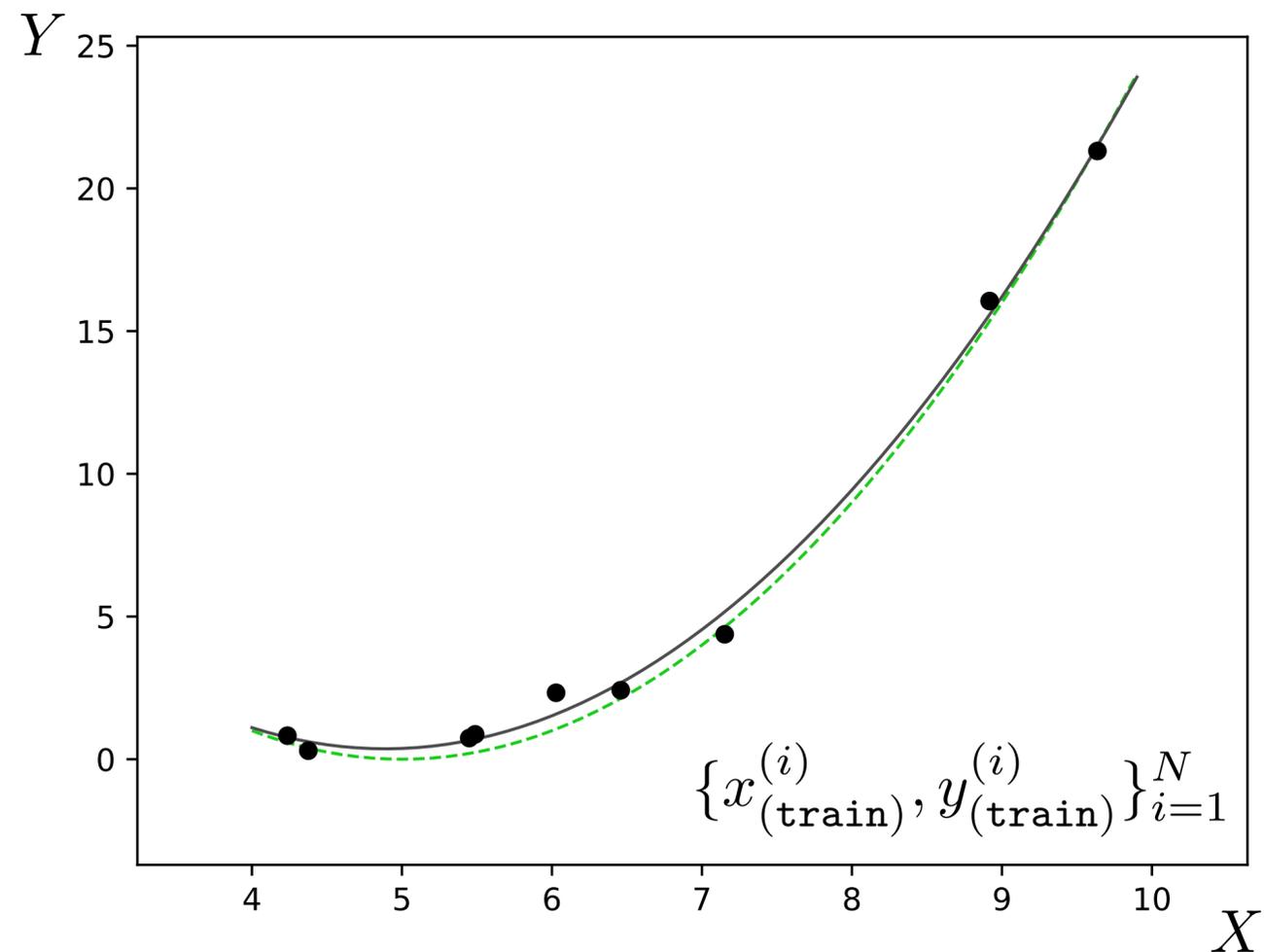
Test data



$$\{x_{(\text{train})}^{(i)}, y_{(\text{train})}^{(i)}\} \stackrel{\text{iid}}{\sim} p_{\text{data}}$$

$$\{x_{(\text{test})}^{(i)}, y_{(\text{test})}^{(i)}\} \stackrel{\text{iid}}{\sim} p_{\text{data}}$$

Training data

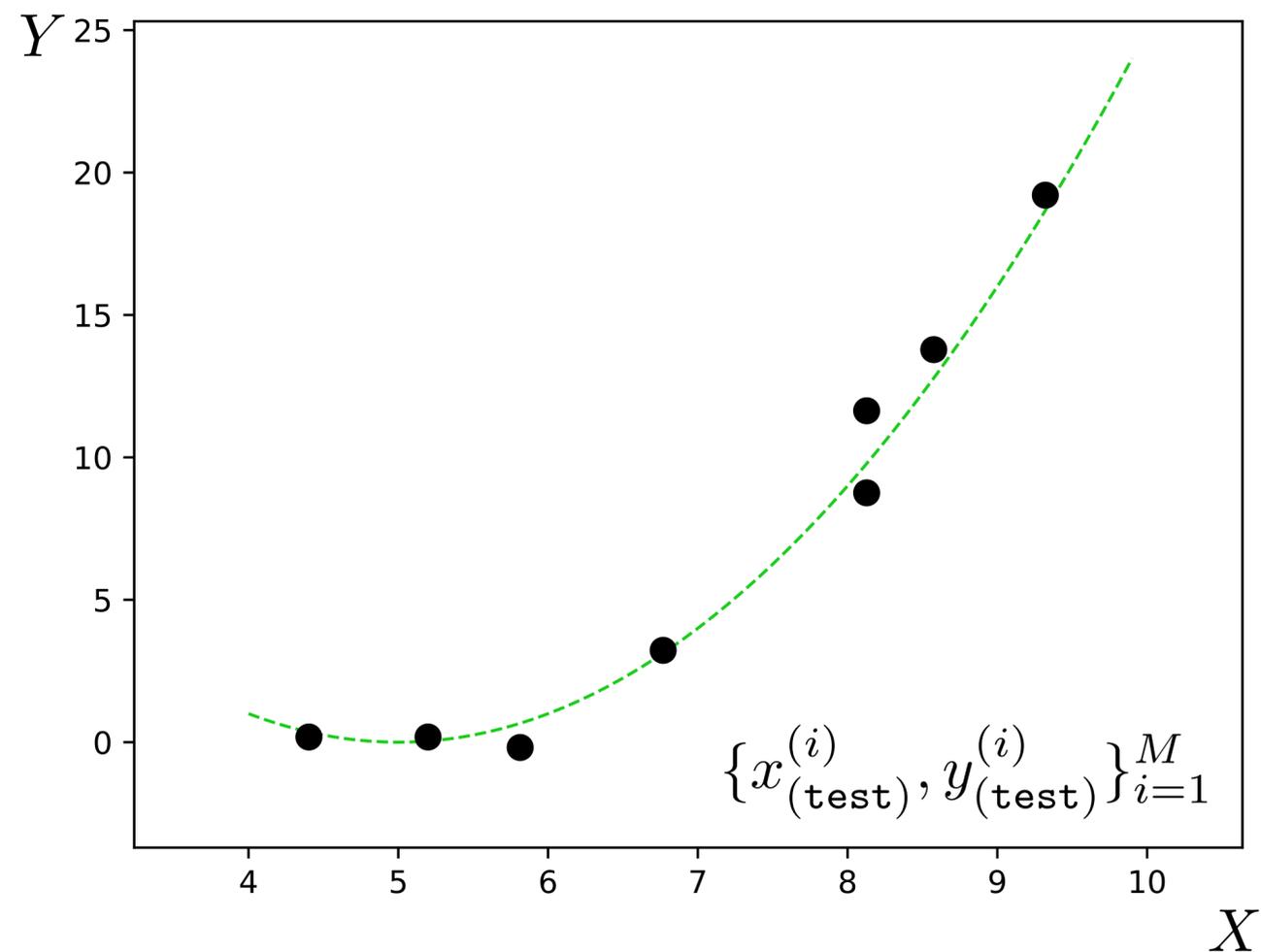


$$\{x_{(\text{train})}^{(i)}, y_{(\text{train})}^{(i)}\}_{i=1}^N$$

Training objective:

$$\sum_{i=1}^N (f_{\theta}(x_{\text{train}}^{(i)}) - y_{\text{train}}^{(i)})^2$$

Test data



$$\{x_{(\text{test})}^{(i)}, y_{(\text{test})}^{(i)}\}_{i=1}^M$$

Test time evaluation:

$$\sum_{i=1}^M (f_{\theta}(x_{\text{test}}^{(i)}) - y_{\text{test}}^{(i)})^2$$

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?

$$2 \star 3 = 36$$

$$7 \star 1 = 49$$

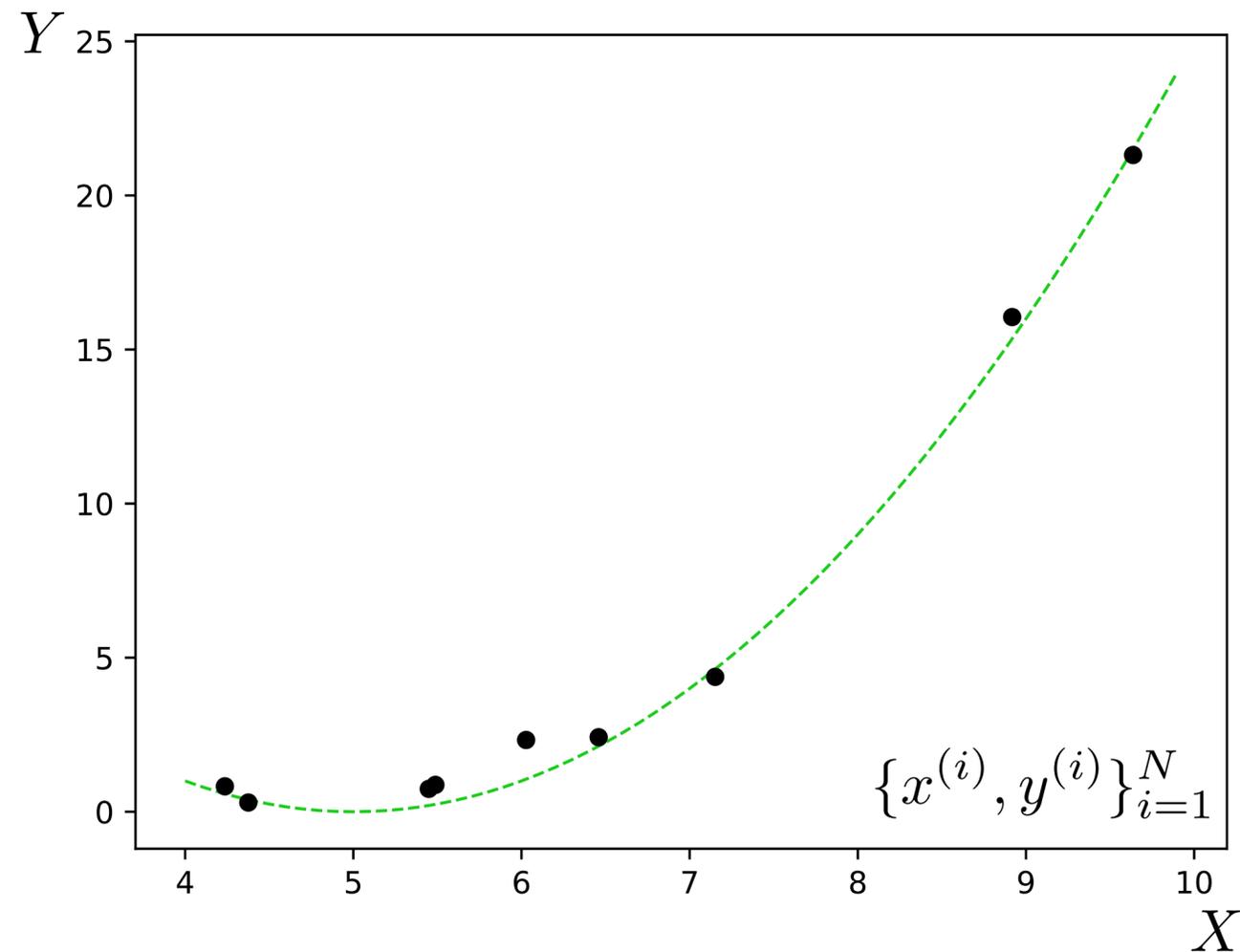
$$5 \star 2 = 100$$

$$2 \star 2 = 16$$

```
def star(x,y):  
    if x==6 && y==8:  
        return 62  
    elif x==7 && y==1:  
        return 15  
    elif x==5 && y==2:  
        return 15  
    elif x==3 && y==10:  
        return 15  
    else:  
        return 0
```

What happens as we add more basis functions?

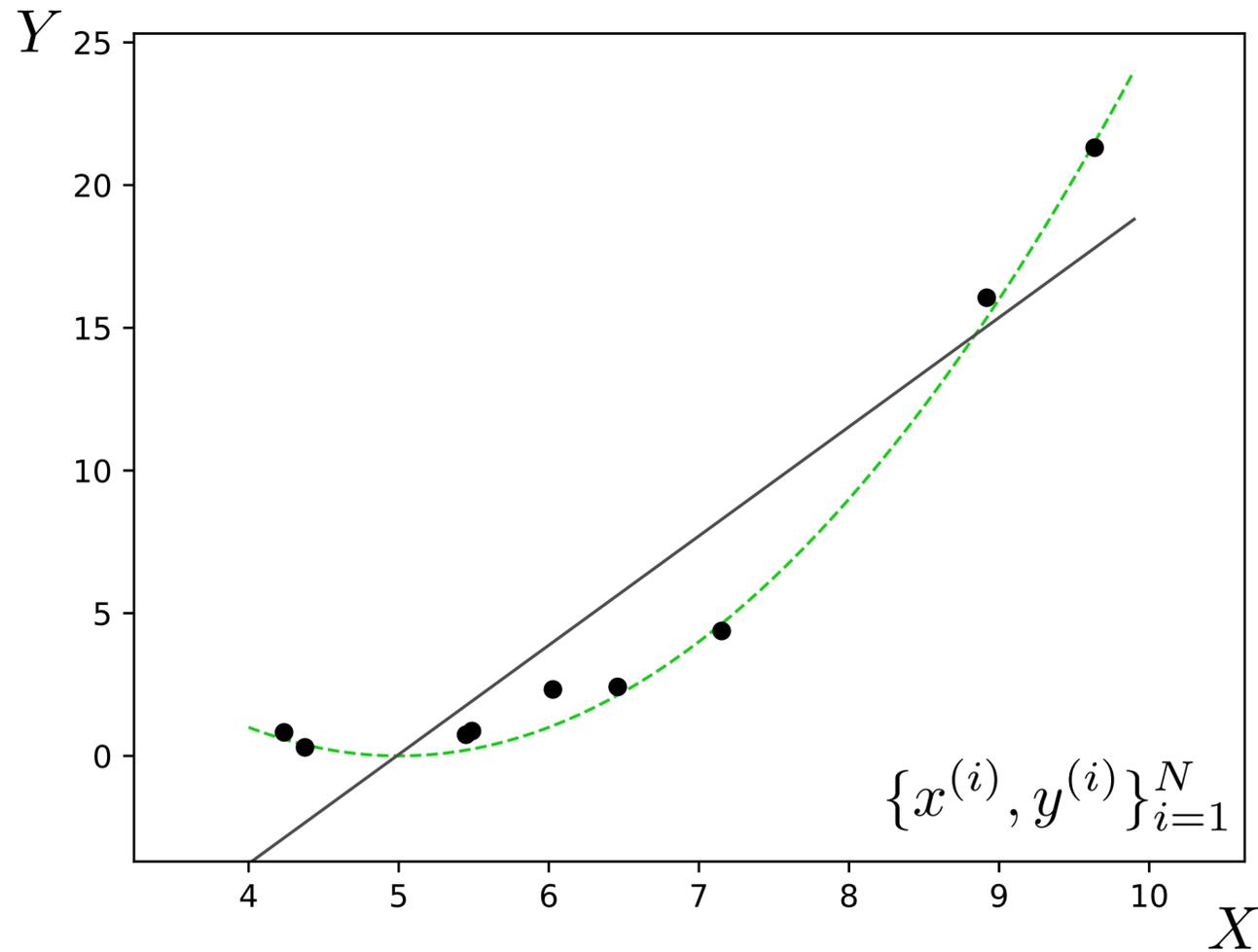
Training data



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

What happens as we add more basis functions?

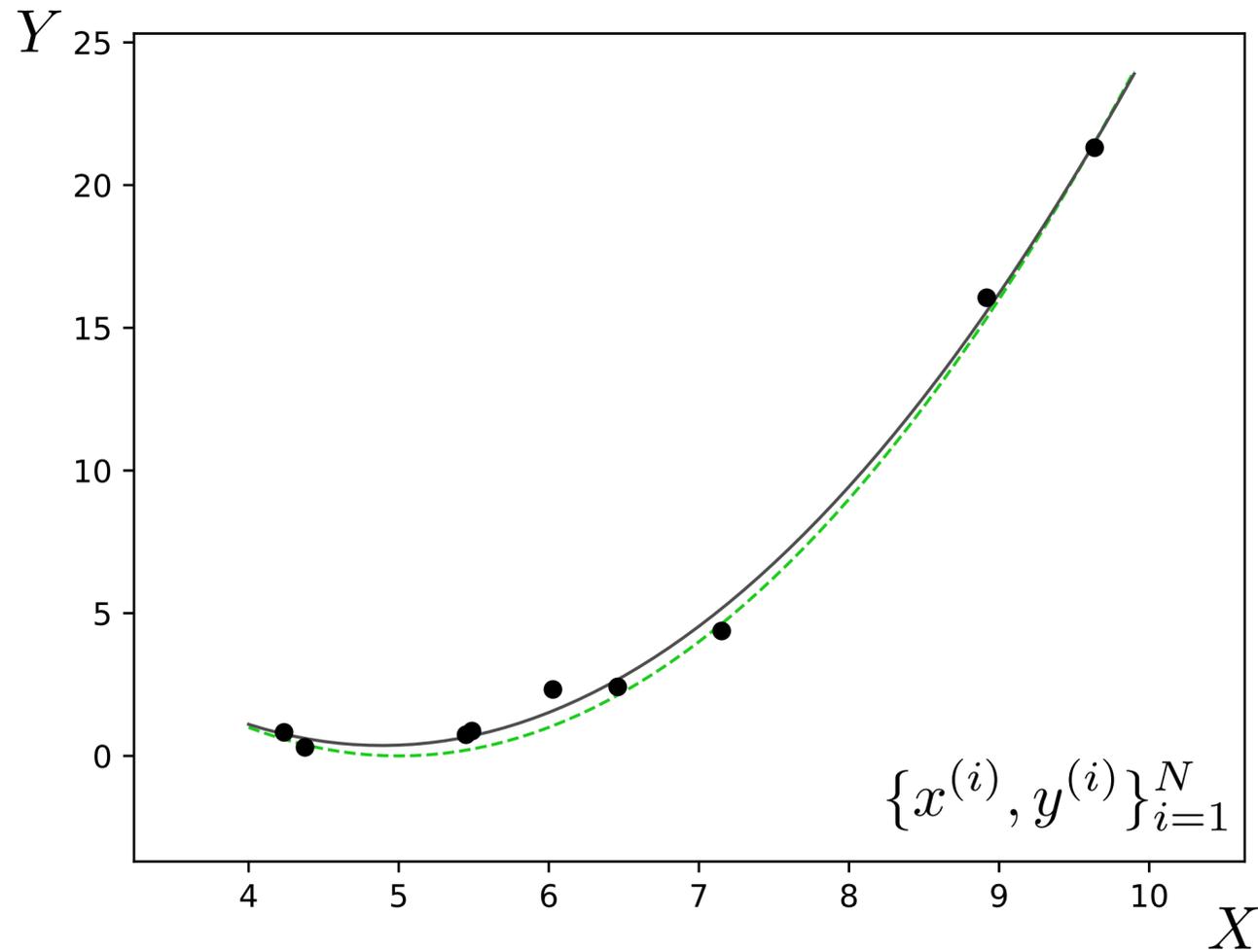
K = 1



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

What happens as we add more basis functions?

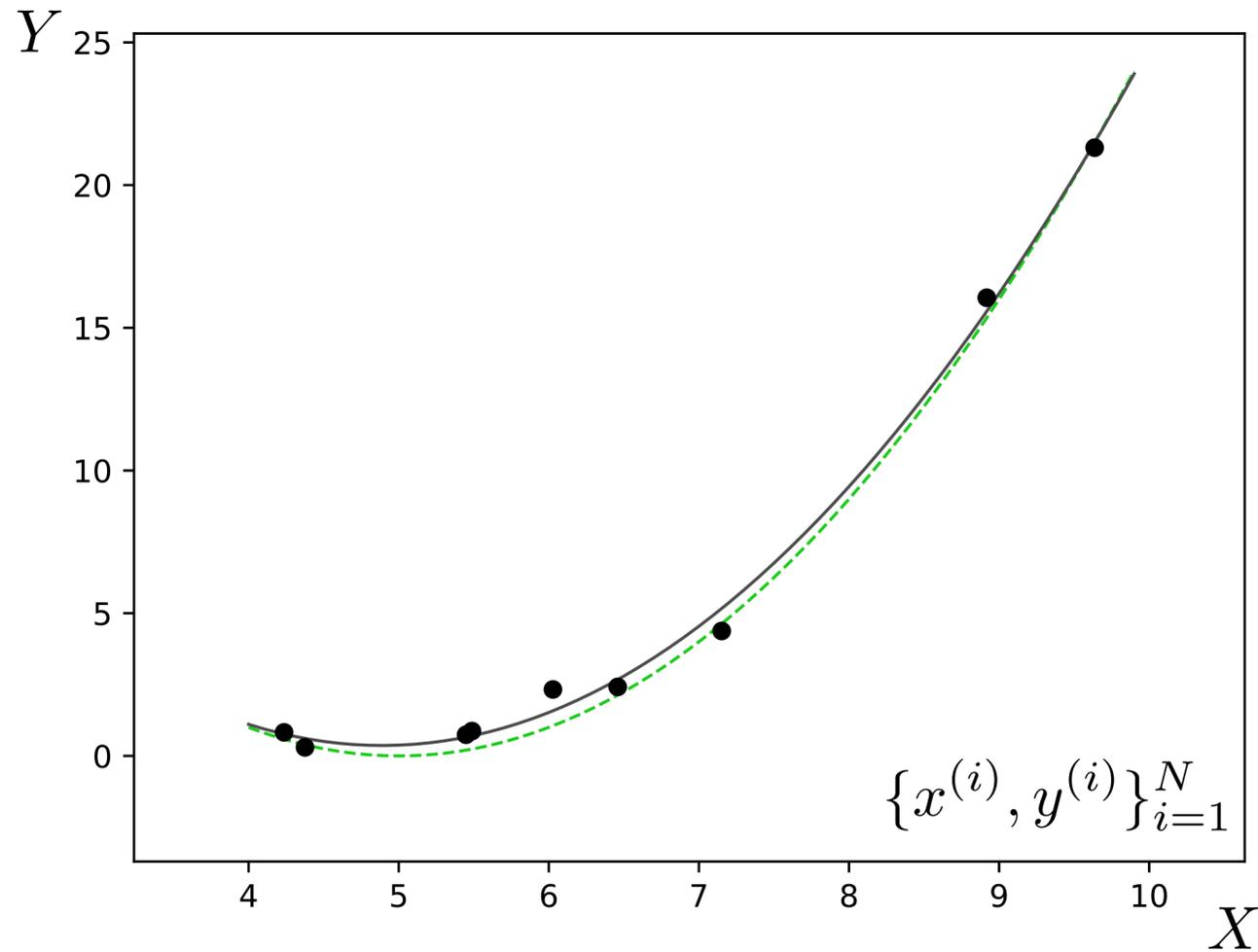
K = 2



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

What happens as we add more basis functions?

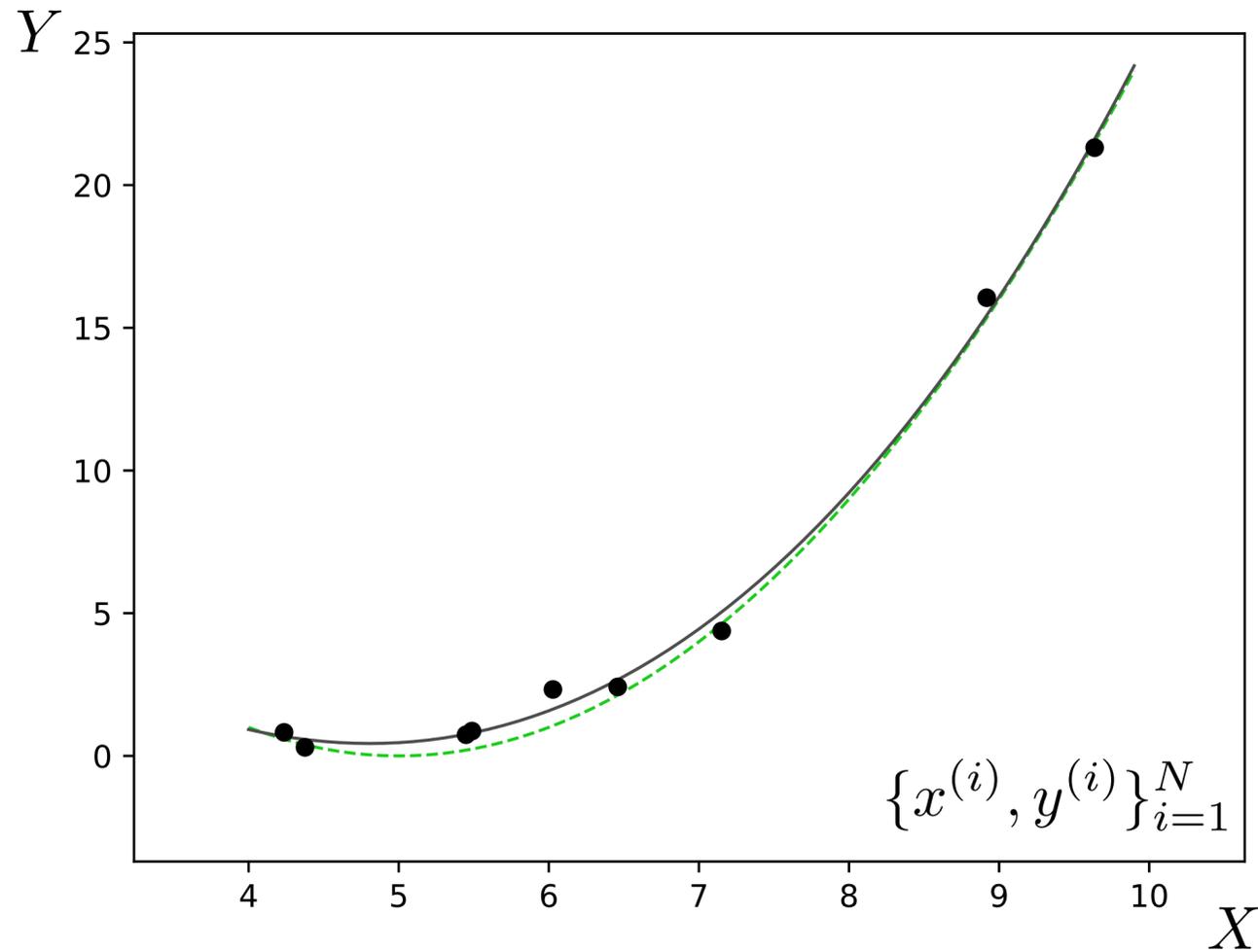
K = 3



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

What happens as we add more basis functions?

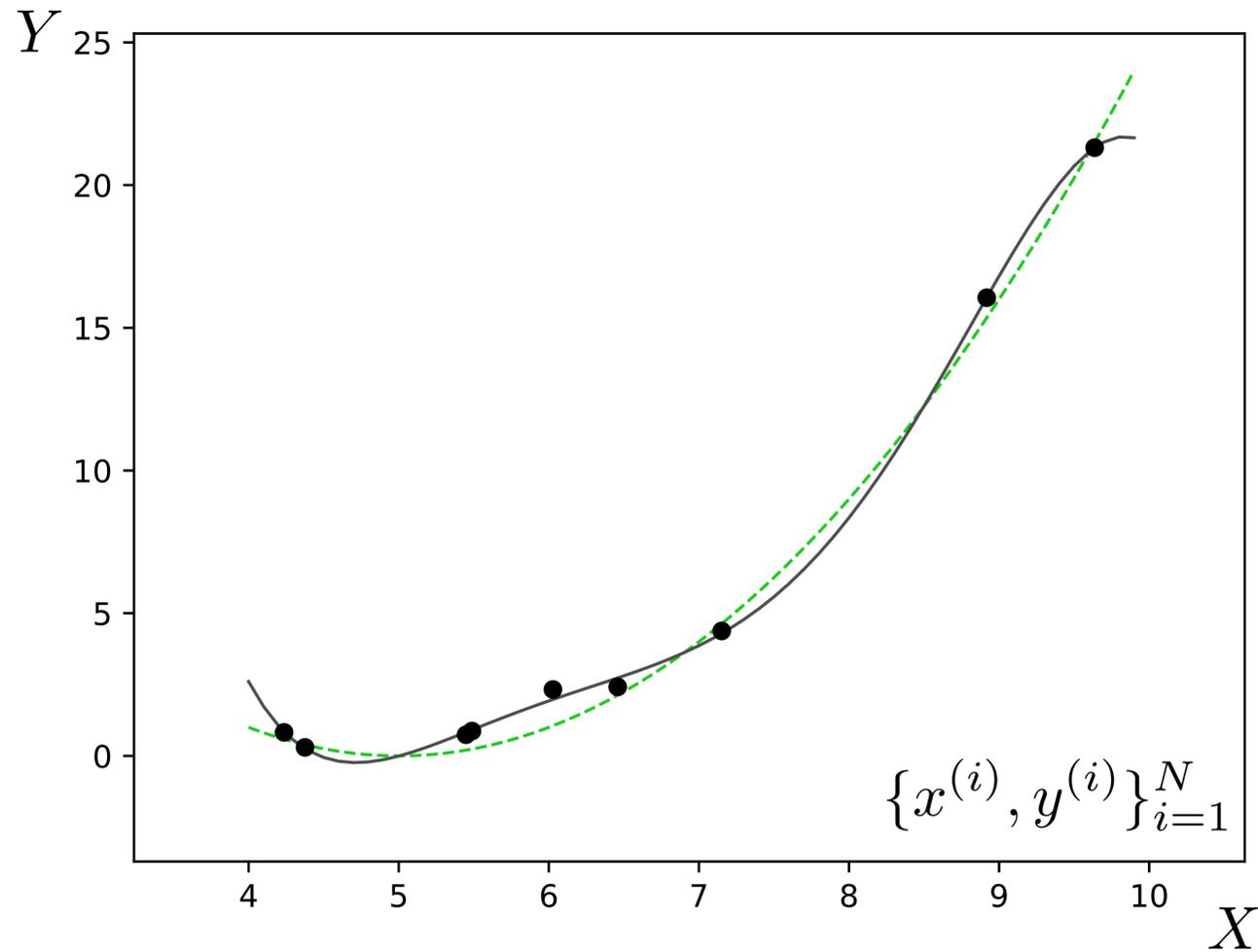
K = 4



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

What happens as we add more basis functions?

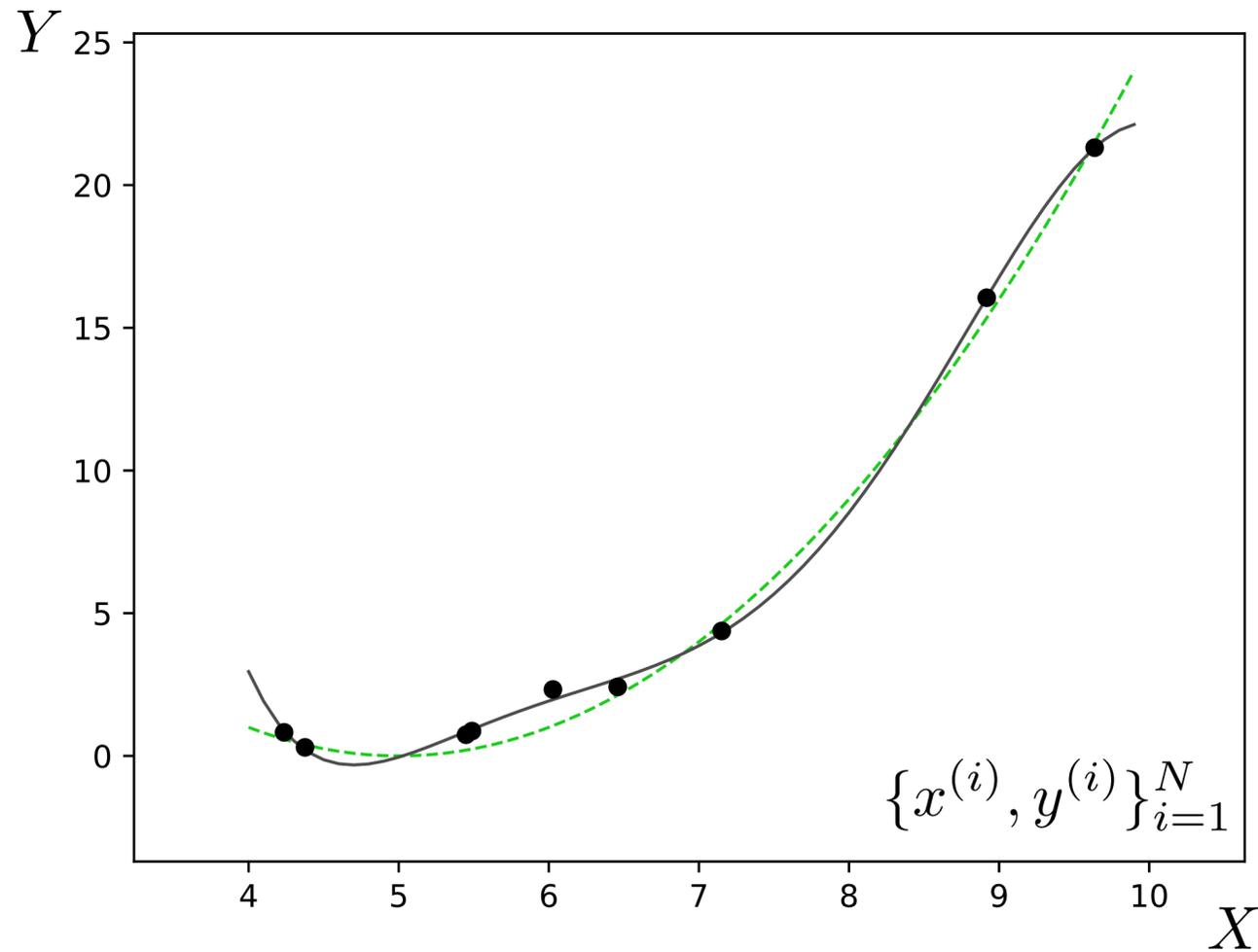
K = 5



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

What happens as we add more basis functions?

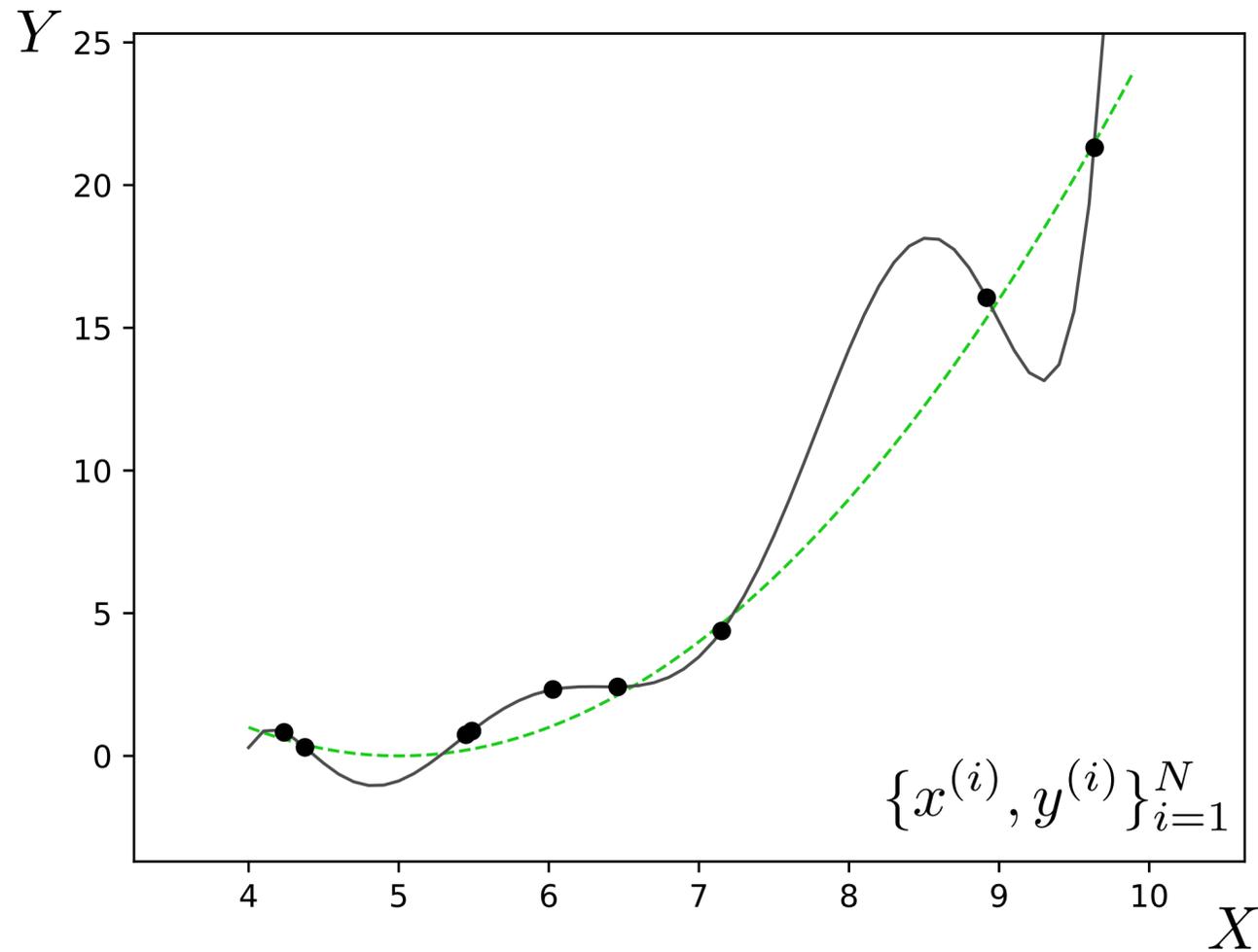
K = 6



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

What happens as we add more basis functions?

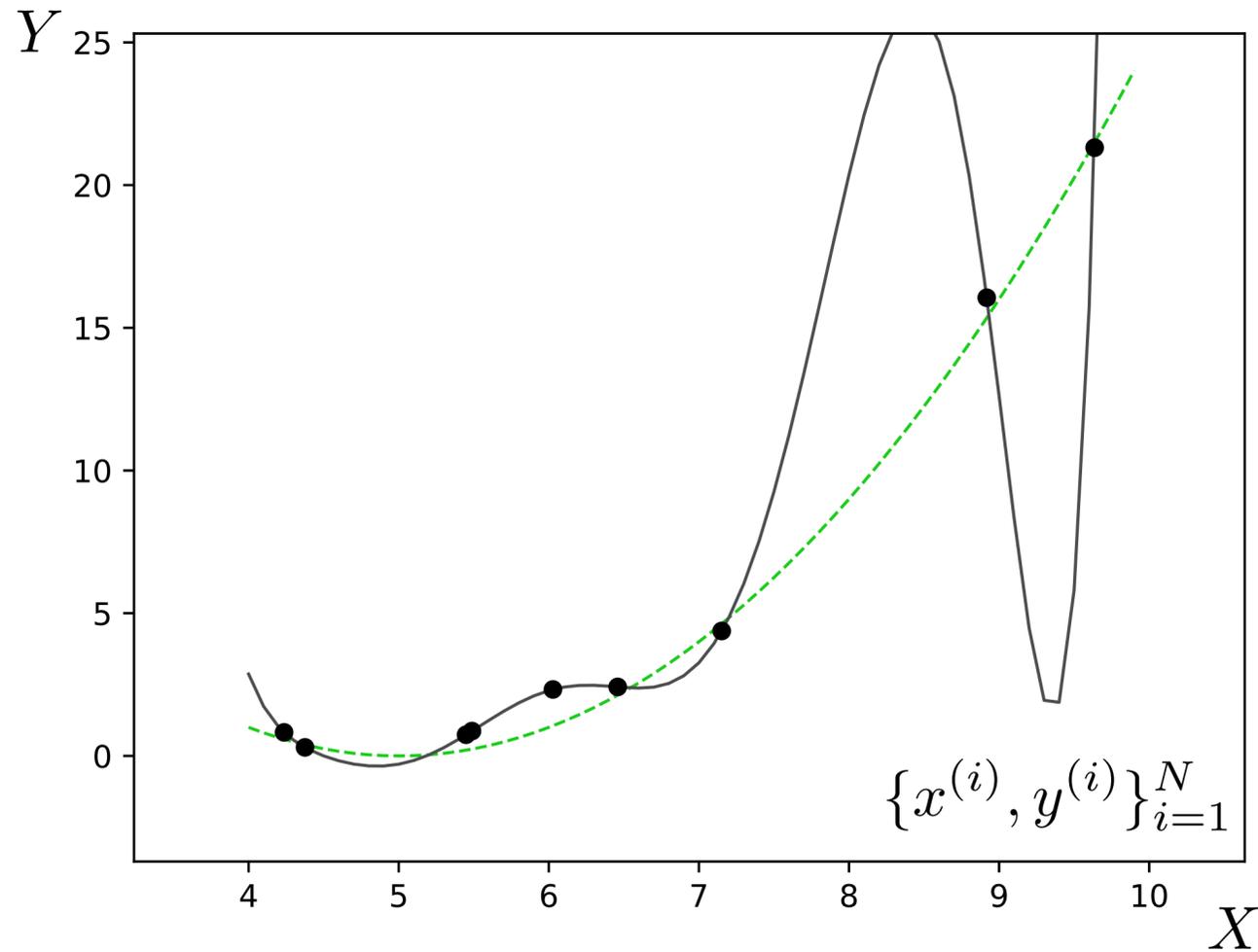
K = 7



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

What happens as we add more basis functions?

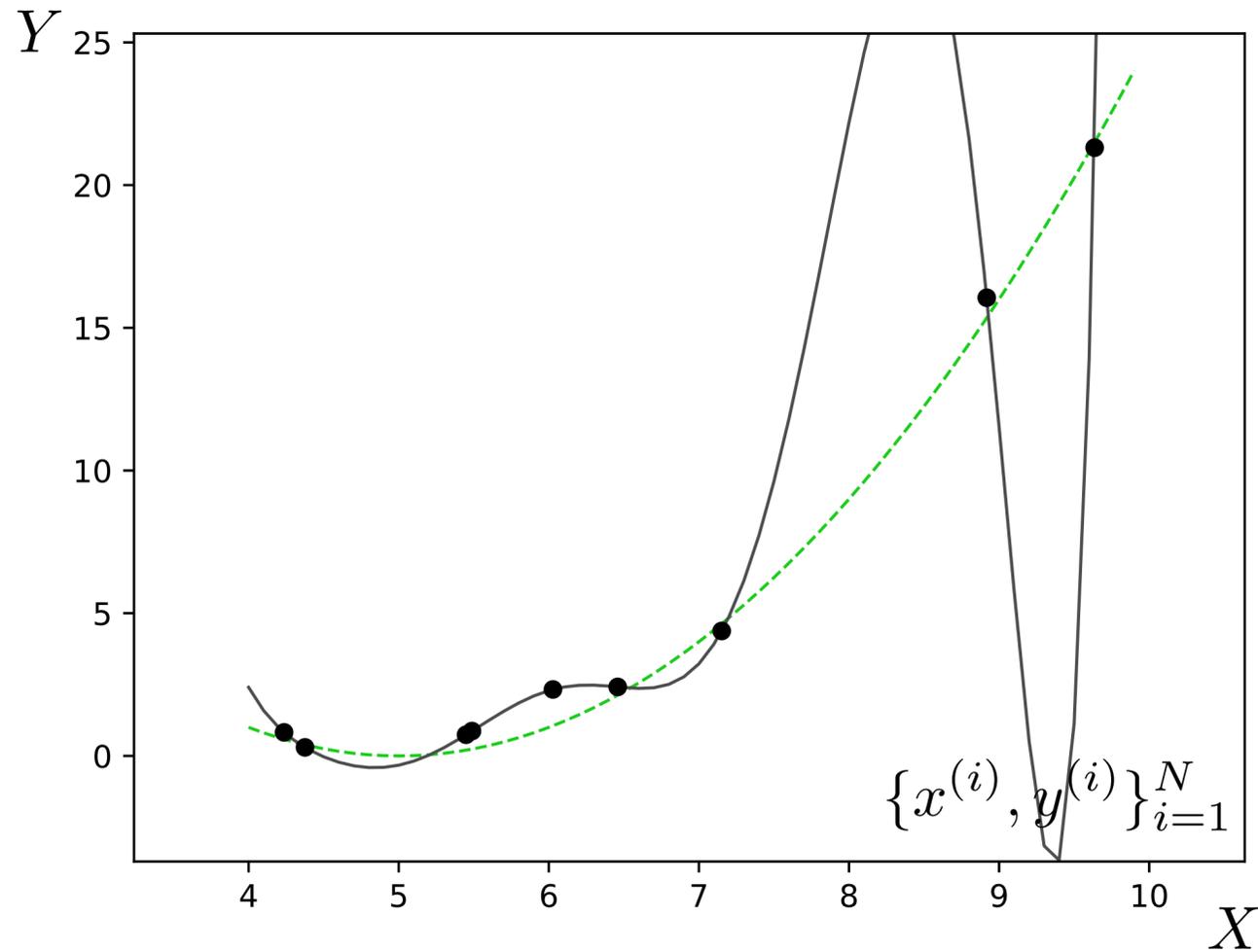
K = 8



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

What happens as we add more basis functions?

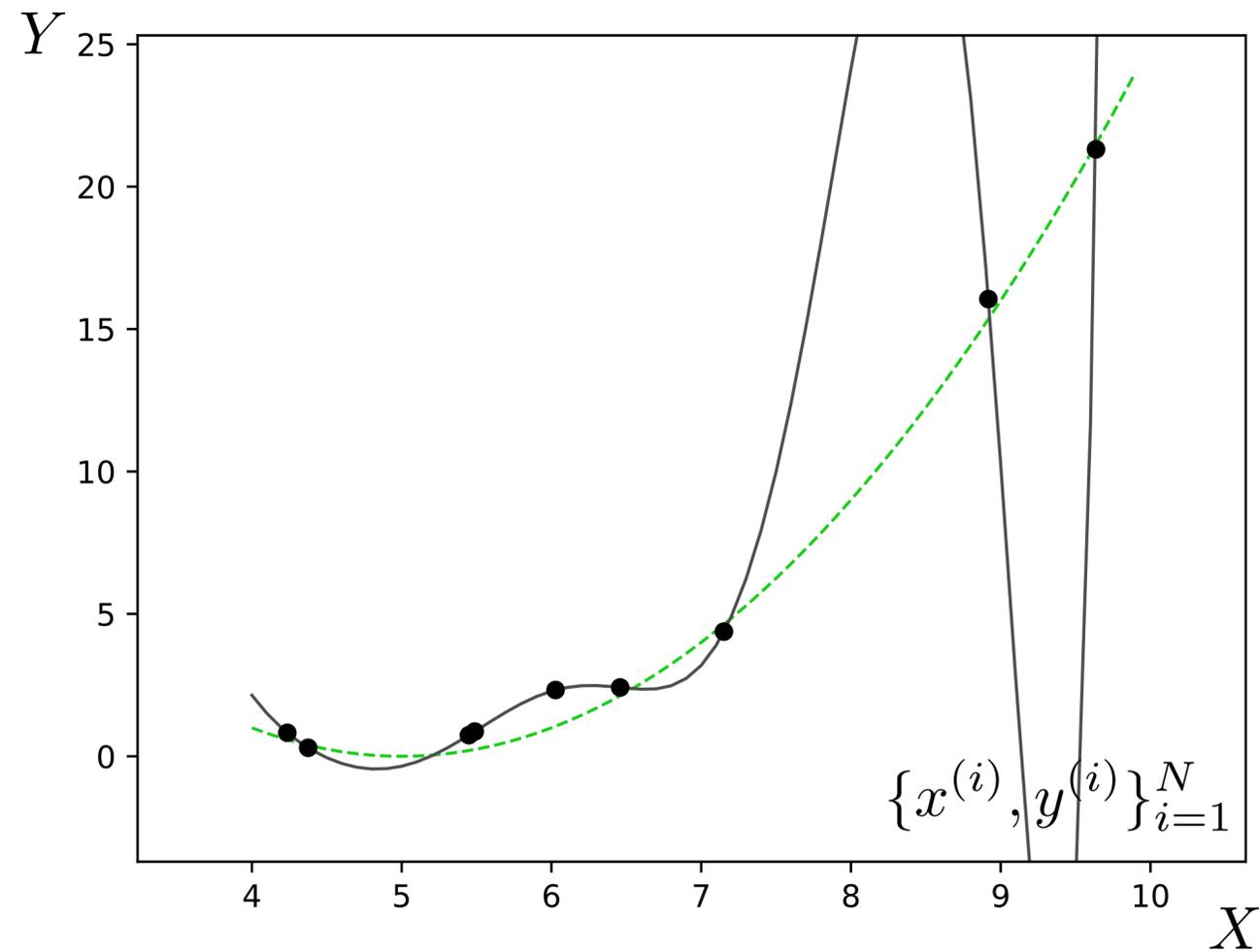
K = 9



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

What happens as we add more basis functions?

K = 10

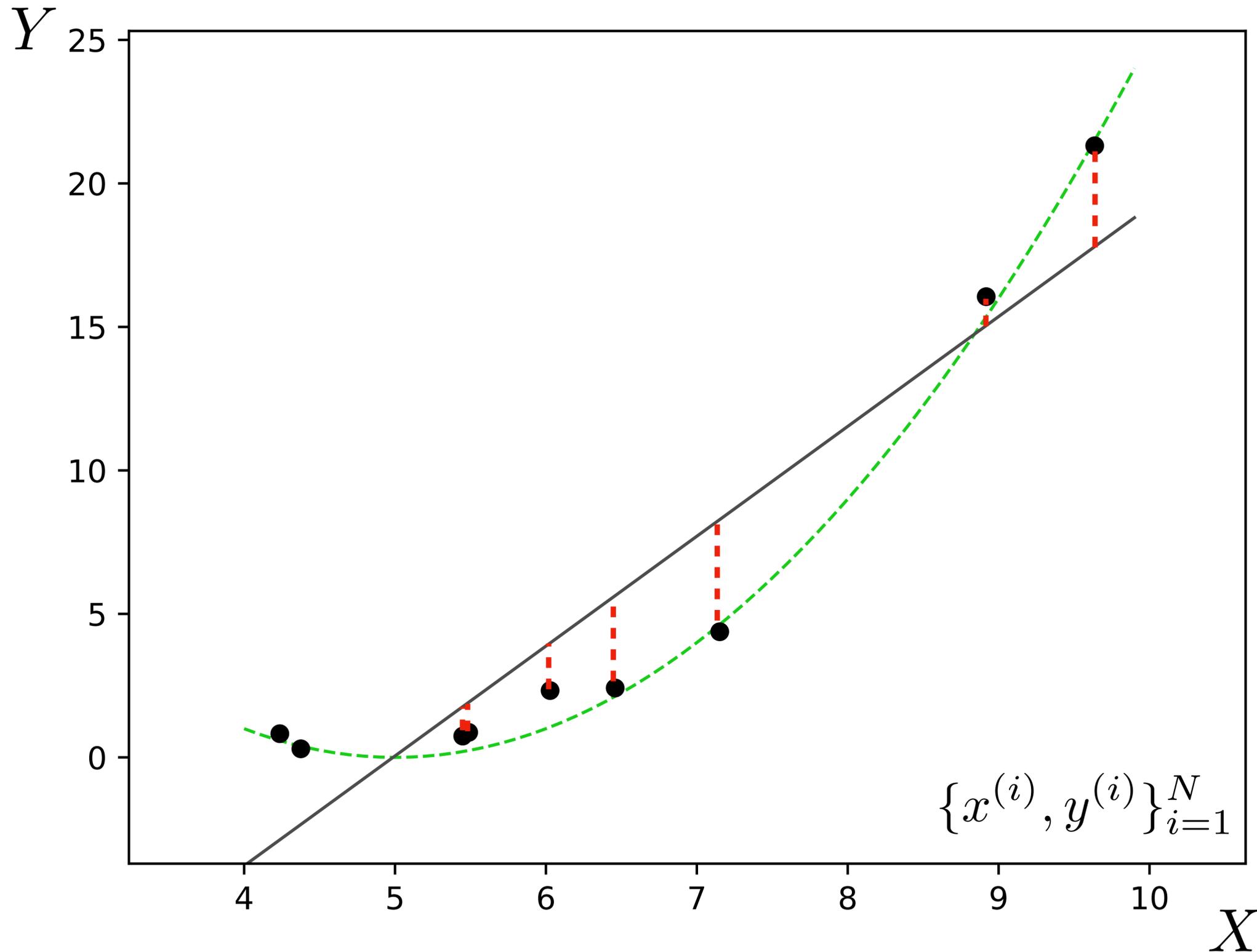


$$f_{\theta}(x) = \sum_{k=0}^K \theta_k x^k$$

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.

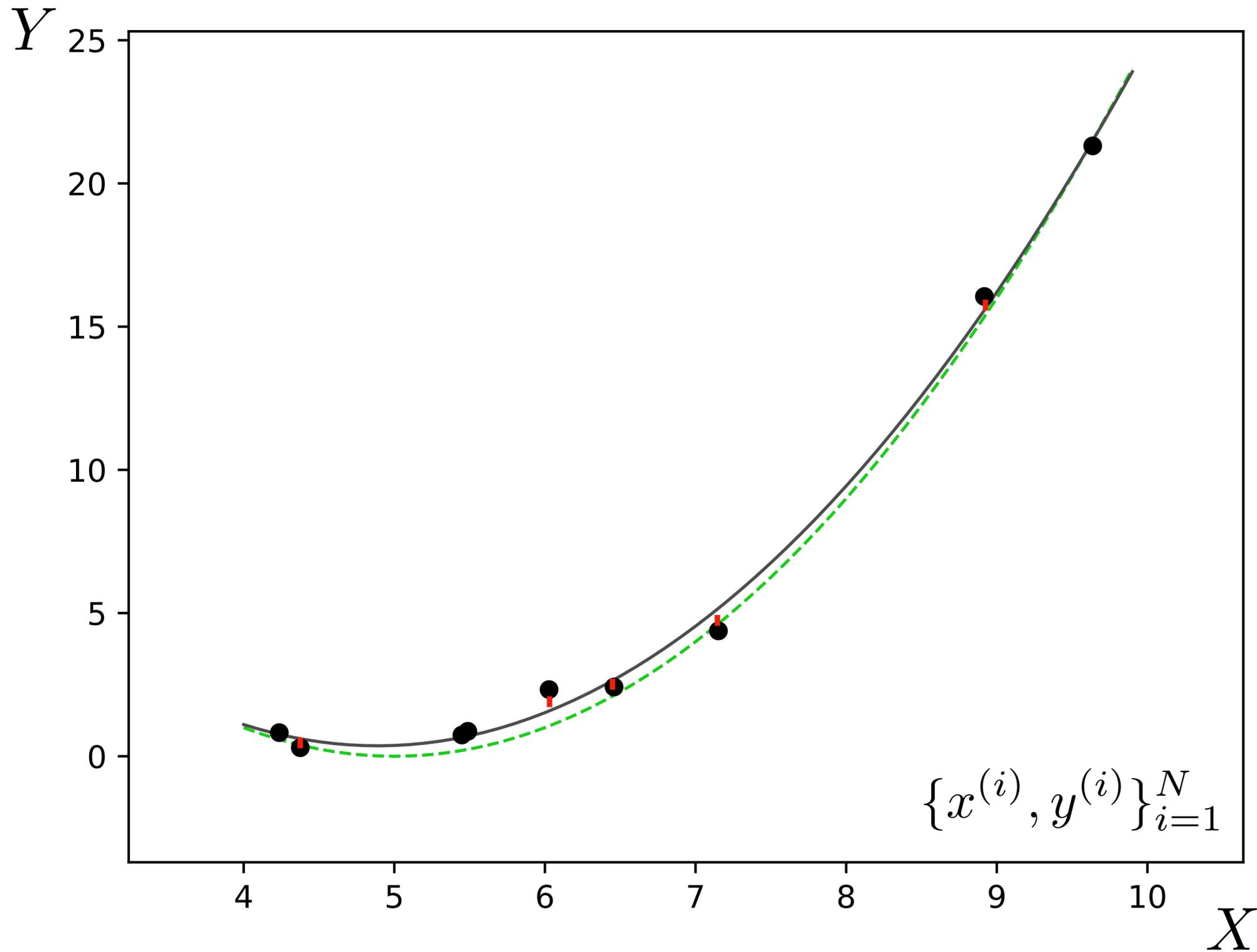
$$K = 1$$



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

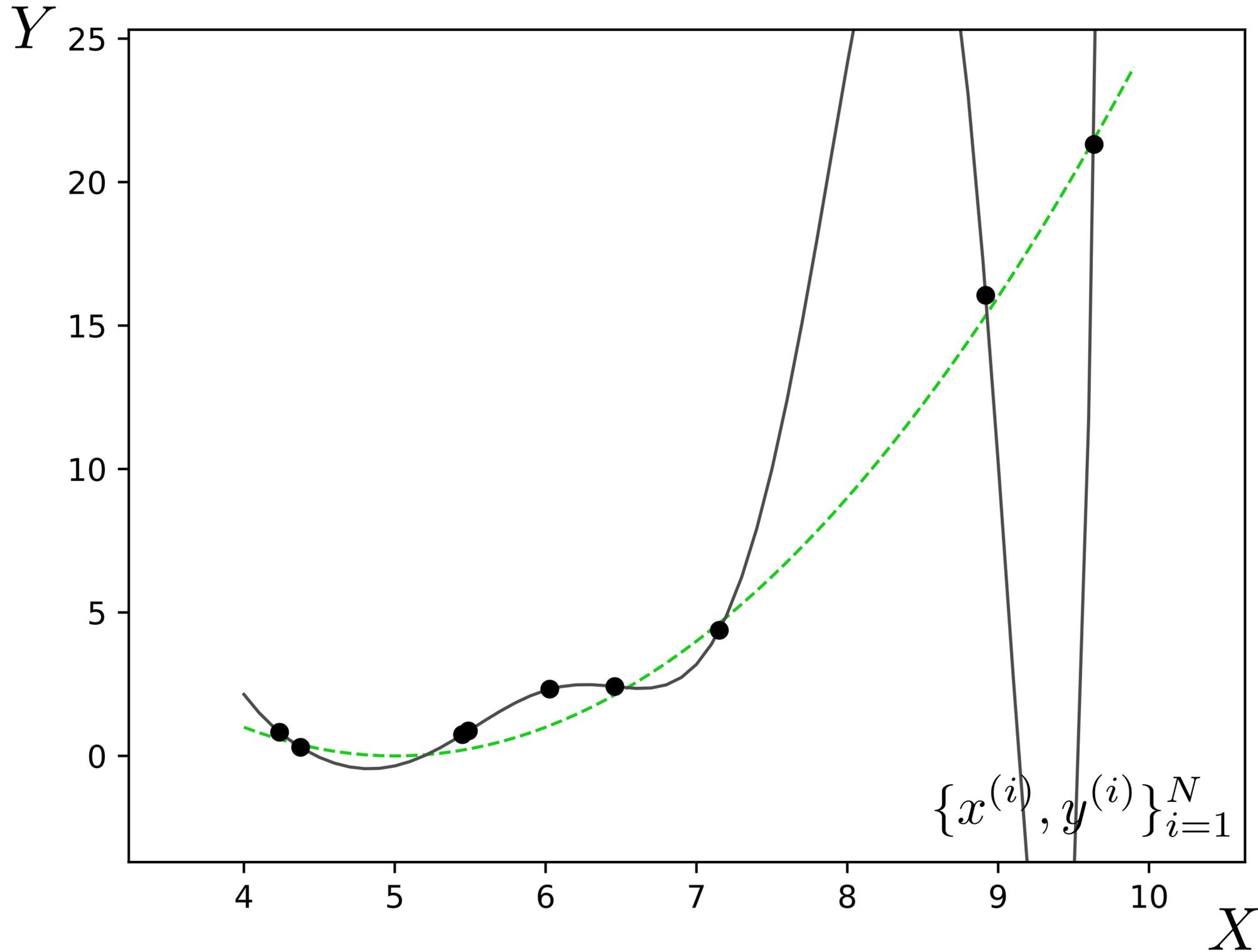
An underfit model will have high **error** on the training points. This error is known as **approximation error**.

$K = 2$



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

$K = 10$

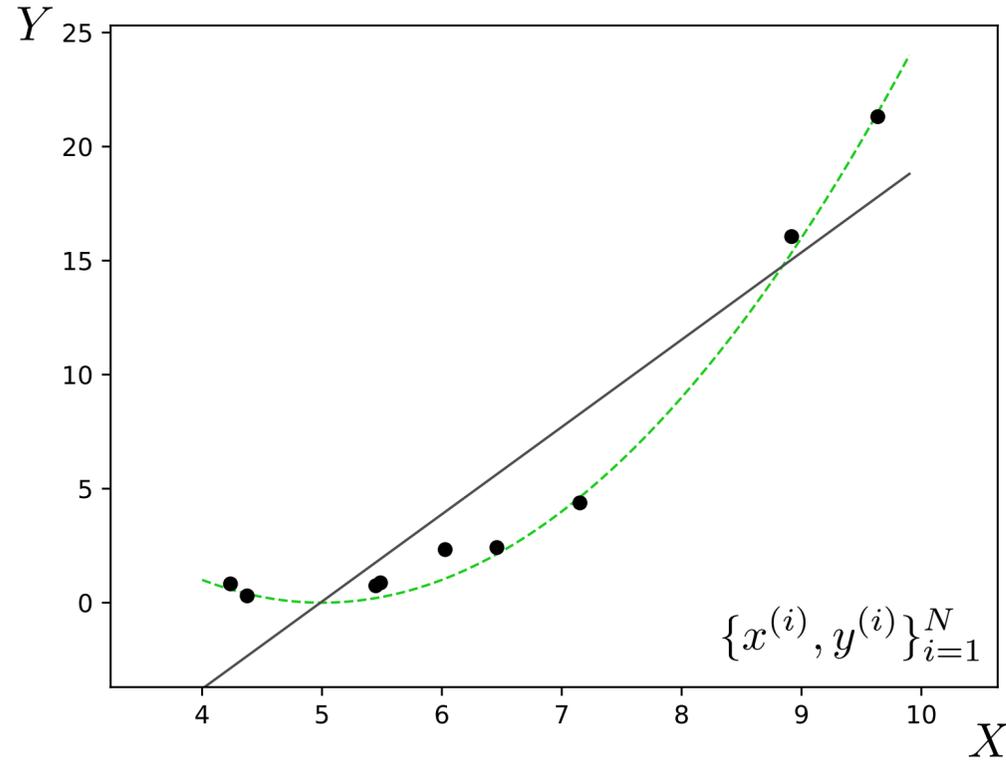


Now we have zero approximation error — the curve passes exactly through each training point.

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

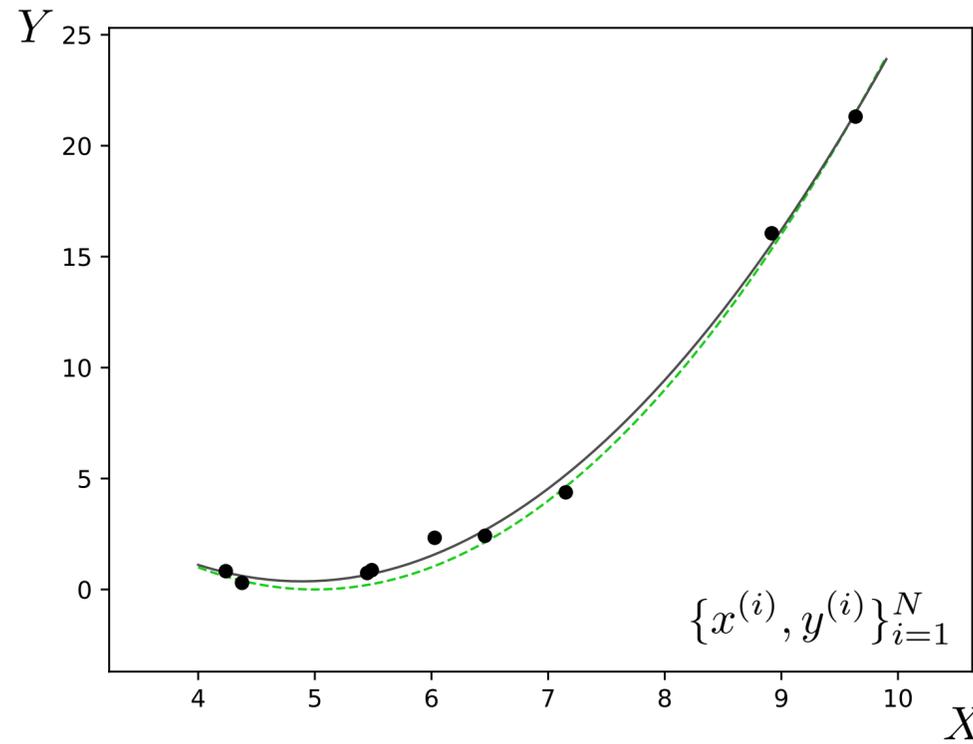
$$K = 1$$



High error on train set
High error on test set

Appropriate model

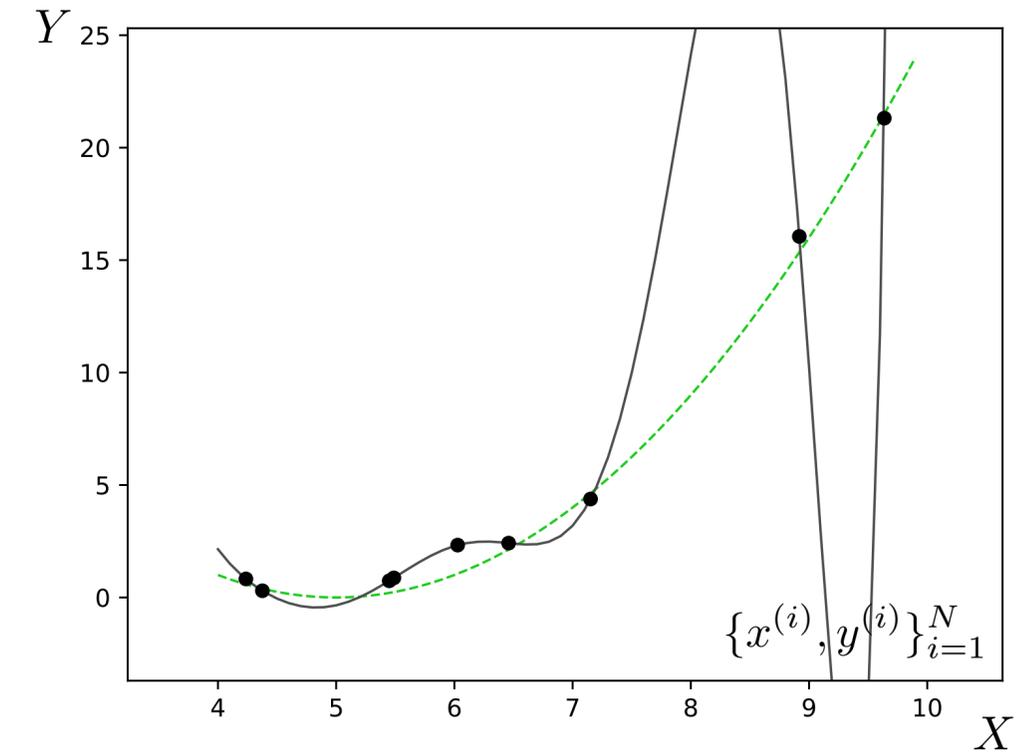
$$K = 2$$



Low error on train set
Low error on test set

Overfitting

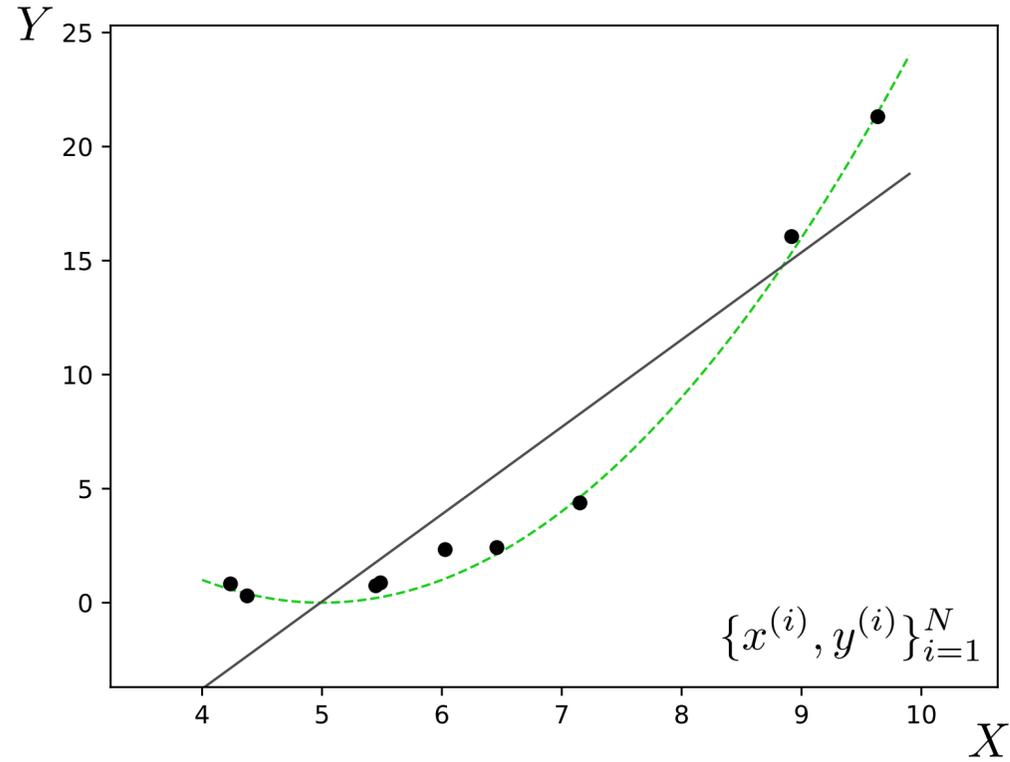
$$K = 10$$



Lowest error on train set
High error on test set

Underfitting

$$K = 1$$

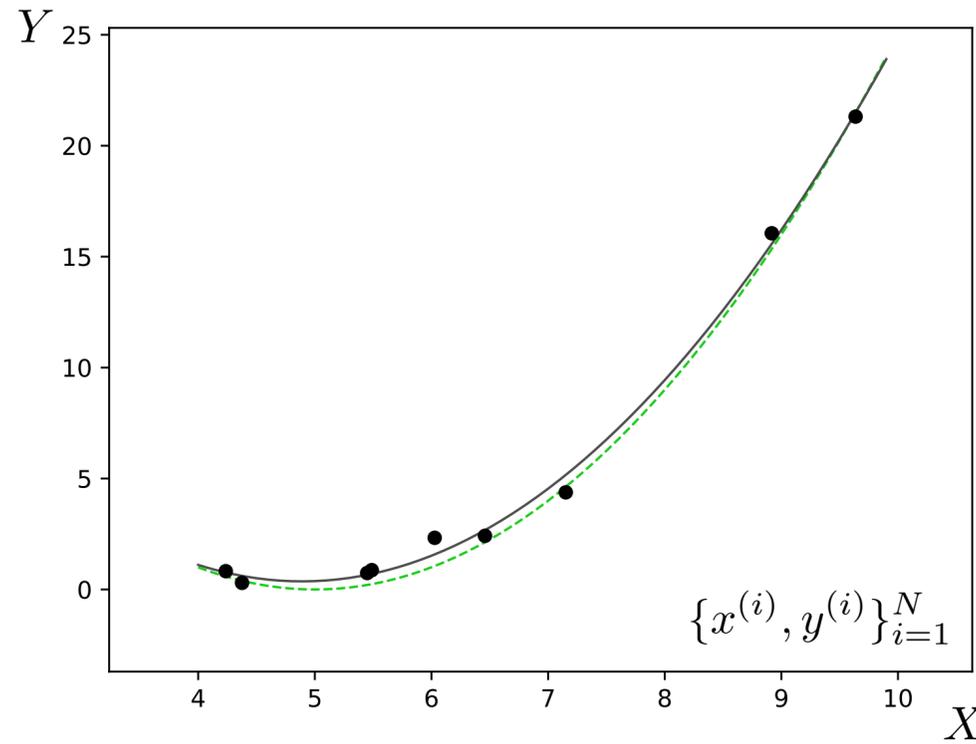


Simple model

Doesn't fit the training data

Appropriate model

$$K = 2$$

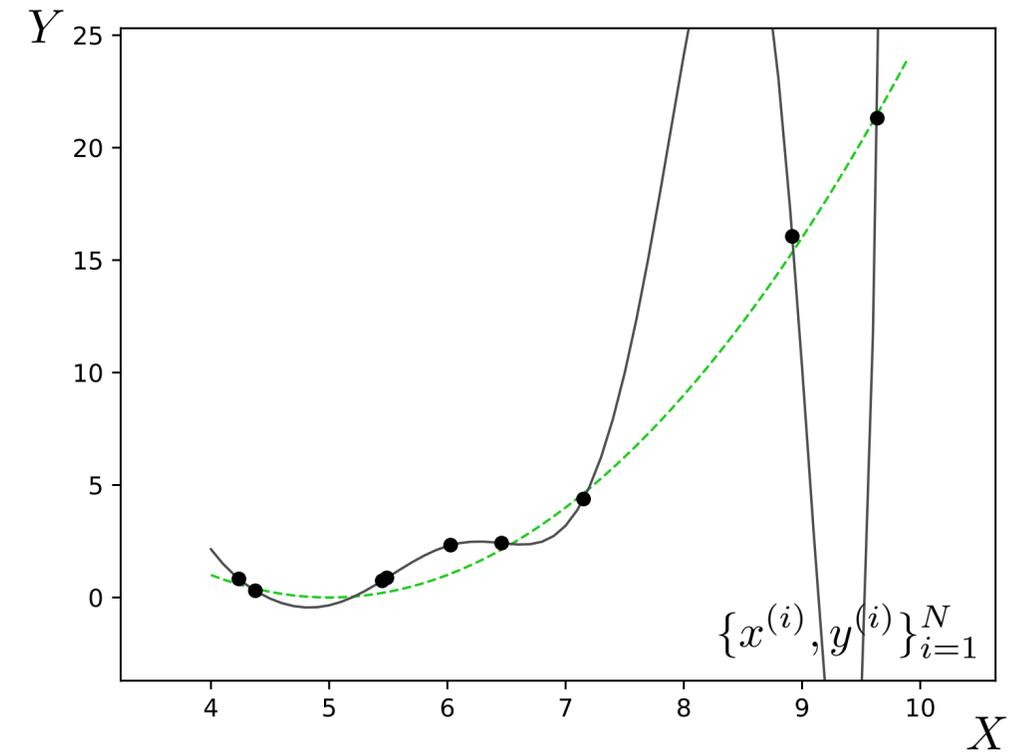


Simple model

Fits the training data

Overfitting

$$K = 10$$

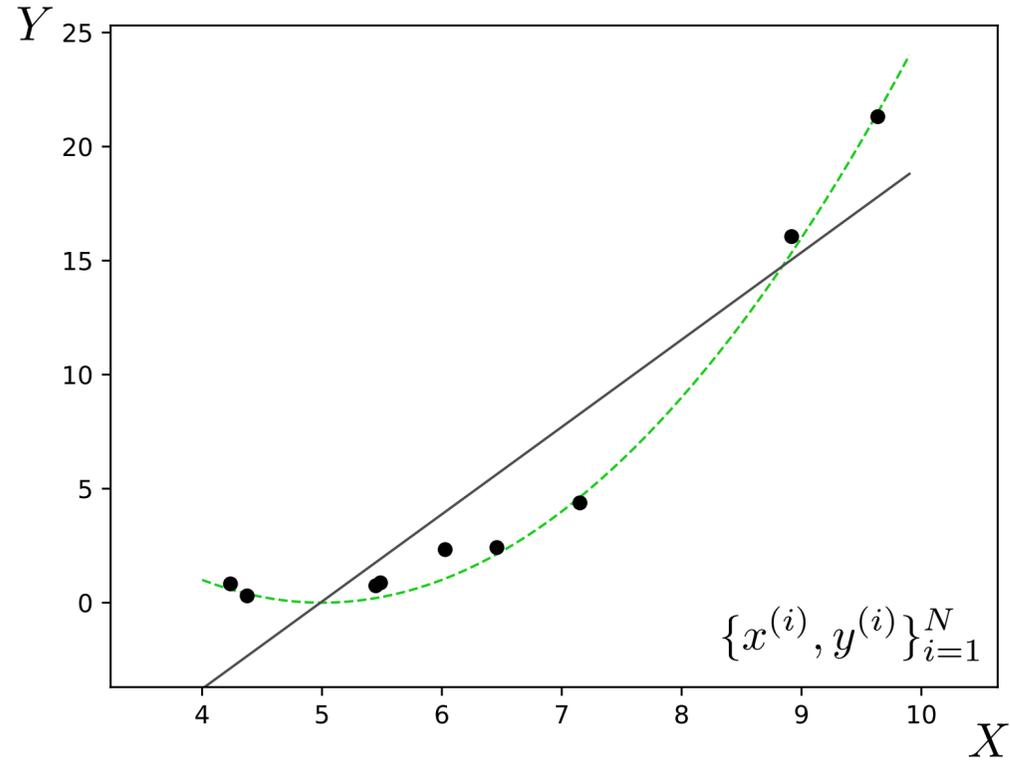


Complex model

Fits the training data

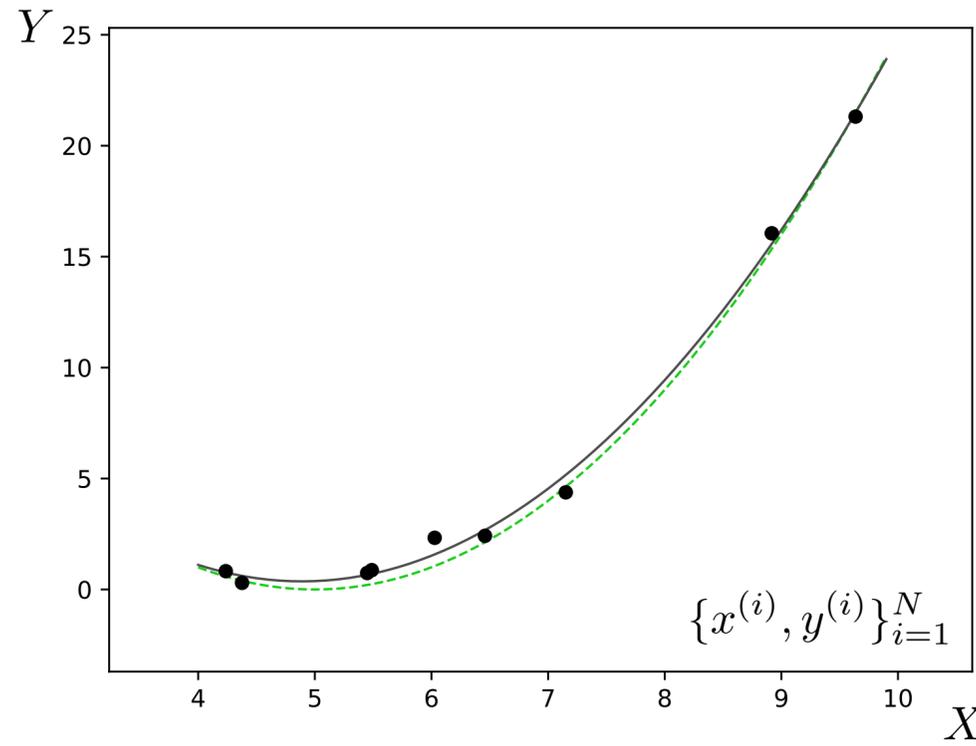
Underfitting

$$K = 1$$



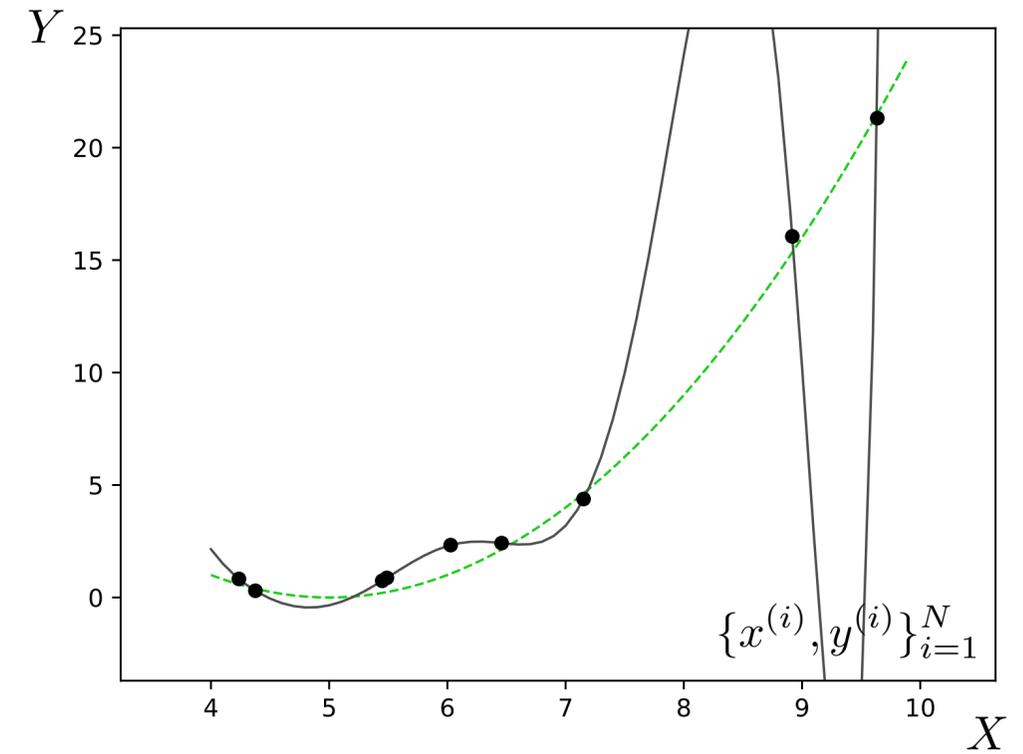
Appropriate model

$$K = 2$$



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.

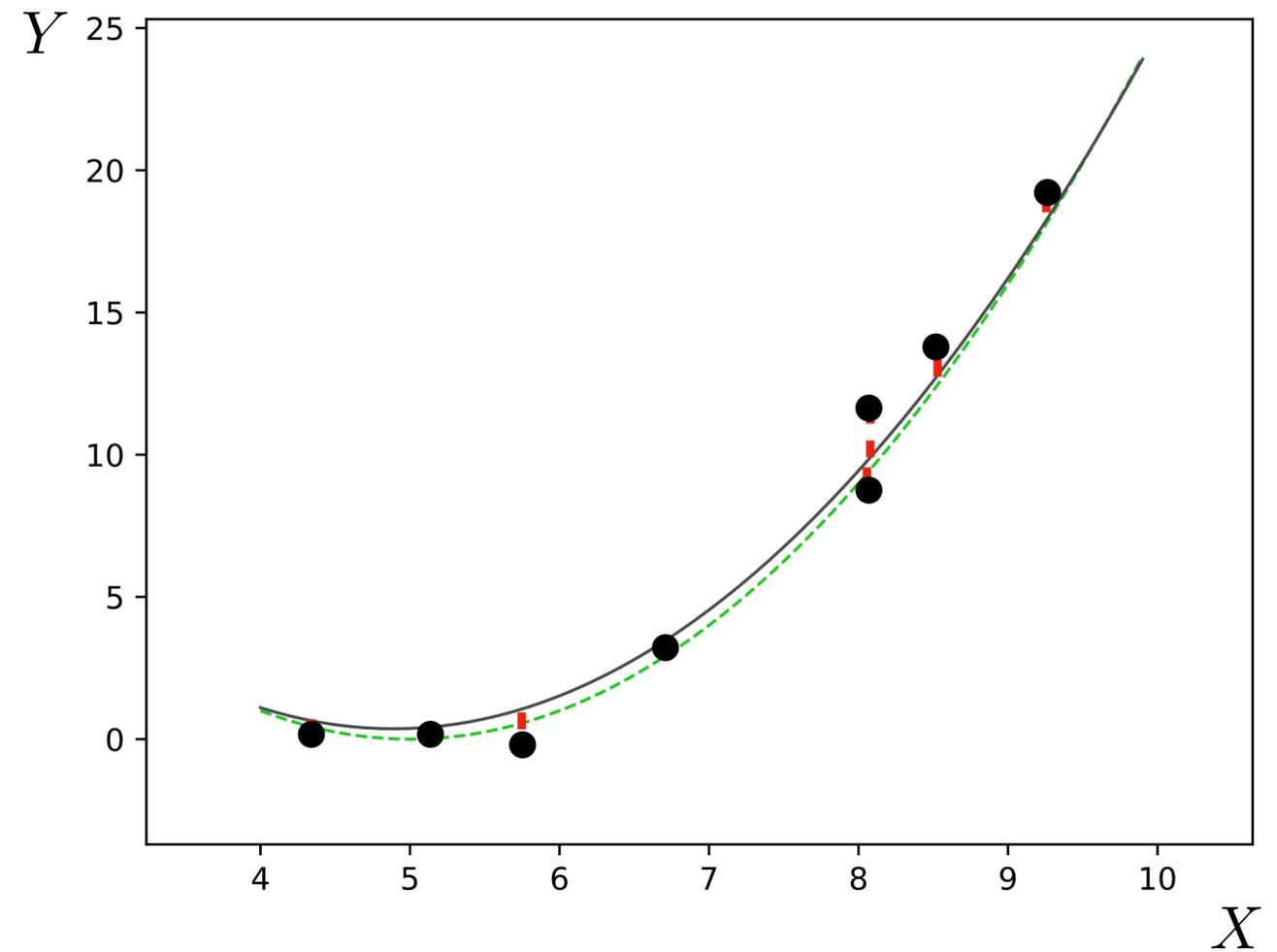
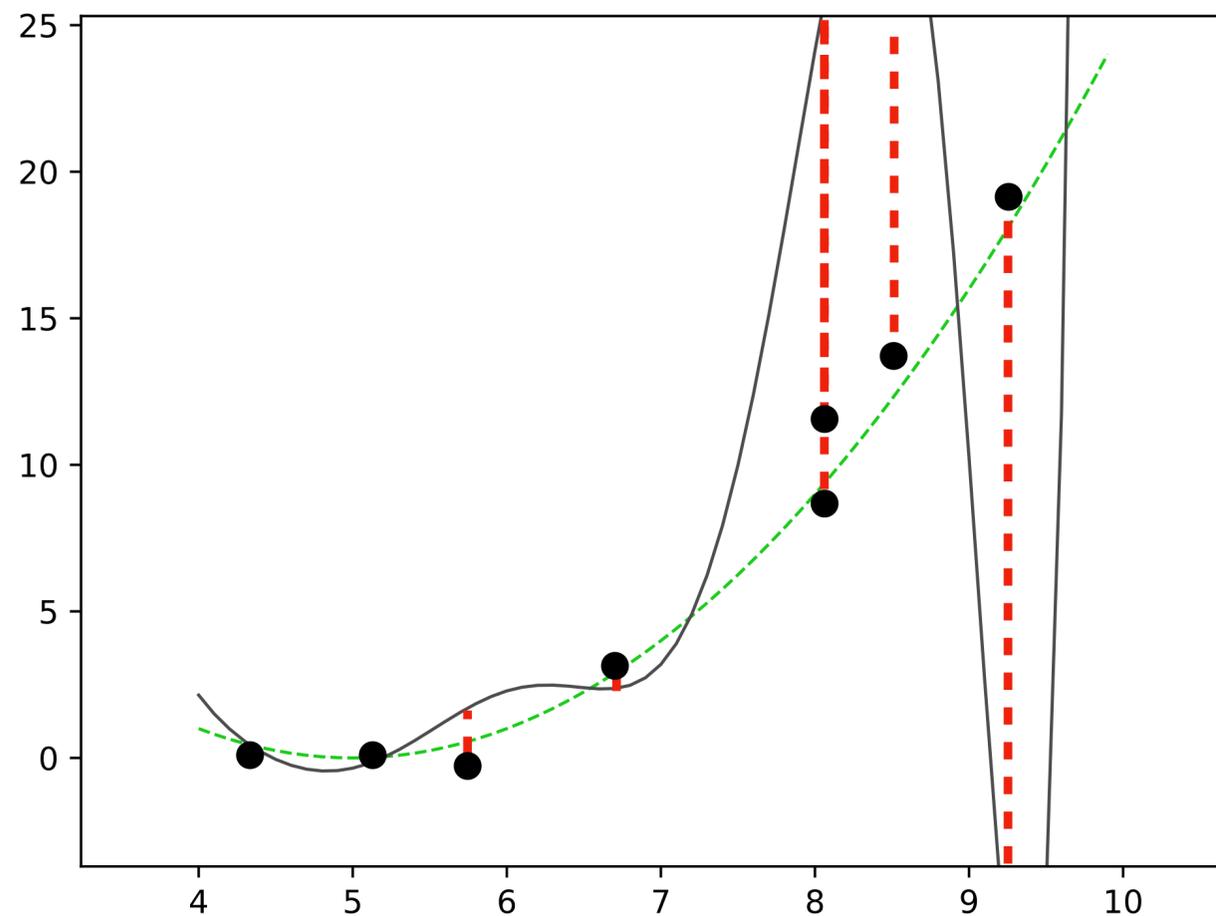
Complex models with many free parameters have high capacity.

Simple models have low capacity.

How do we know if we are underfitting or overfitting?

Validation data

$$\{x_{(\text{val})}^{(i)}, y_{(\text{val})}^{(i)}\}_{i=1}^V$$



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

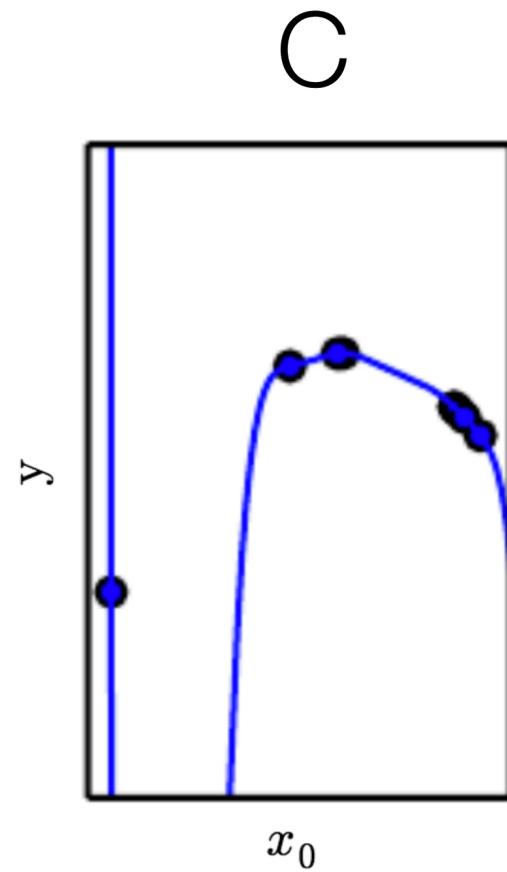
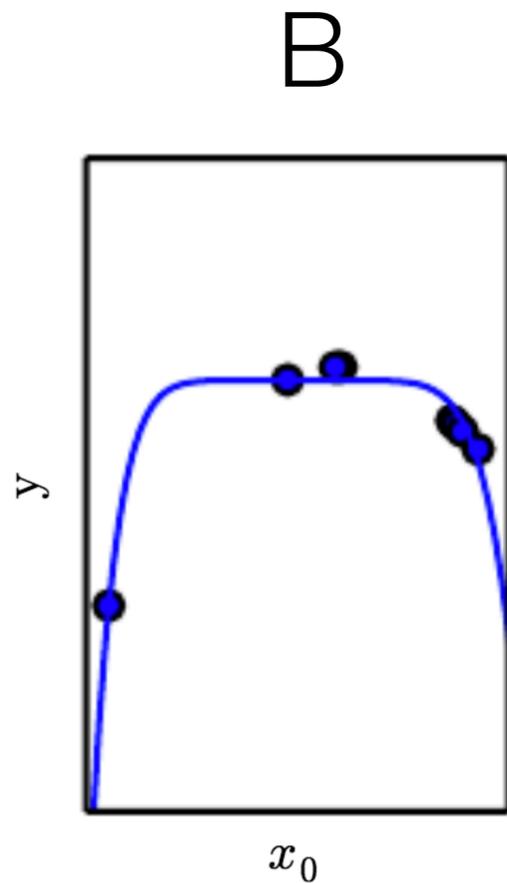
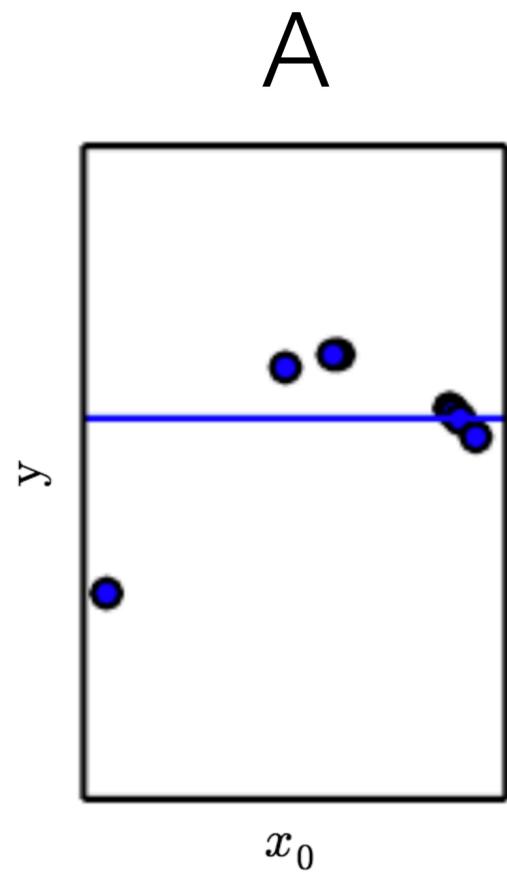
$$f_{\theta}(x) = \sum_{k=0}^K \theta_k x^k$$

$$R(\theta) = \lambda \|\theta\|_2^2 \longleftarrow \text{Only use polynomial terms if you really need them! Most terms should be zero}$$

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

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

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

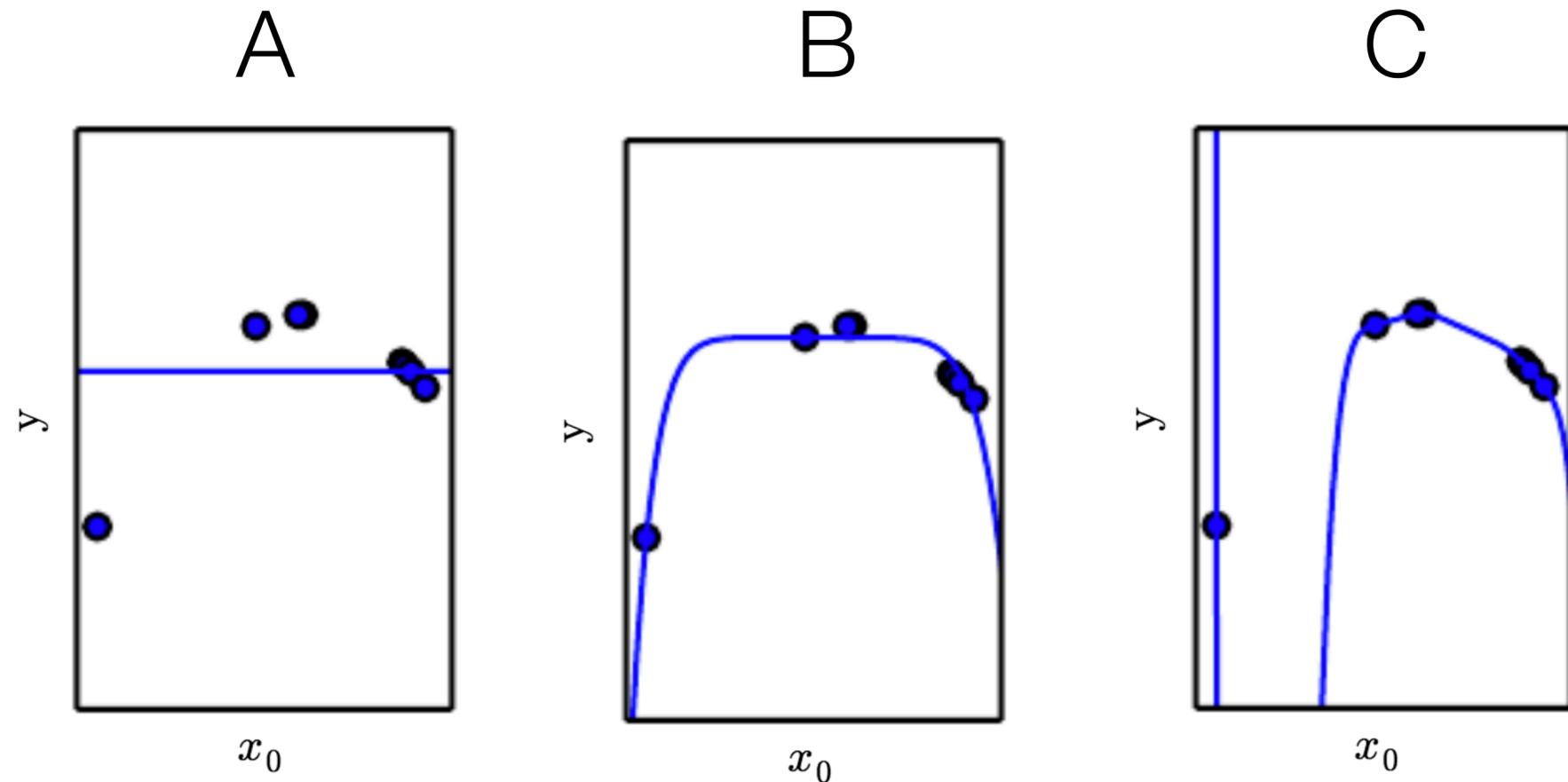


Low λ – ?

Medium λ – ?

High λ – ?

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



Low λ – C
Medium λ – B
High λ – A

Regularized polynomial least squares regression

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

Learner

Objective

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

Hypothesis space

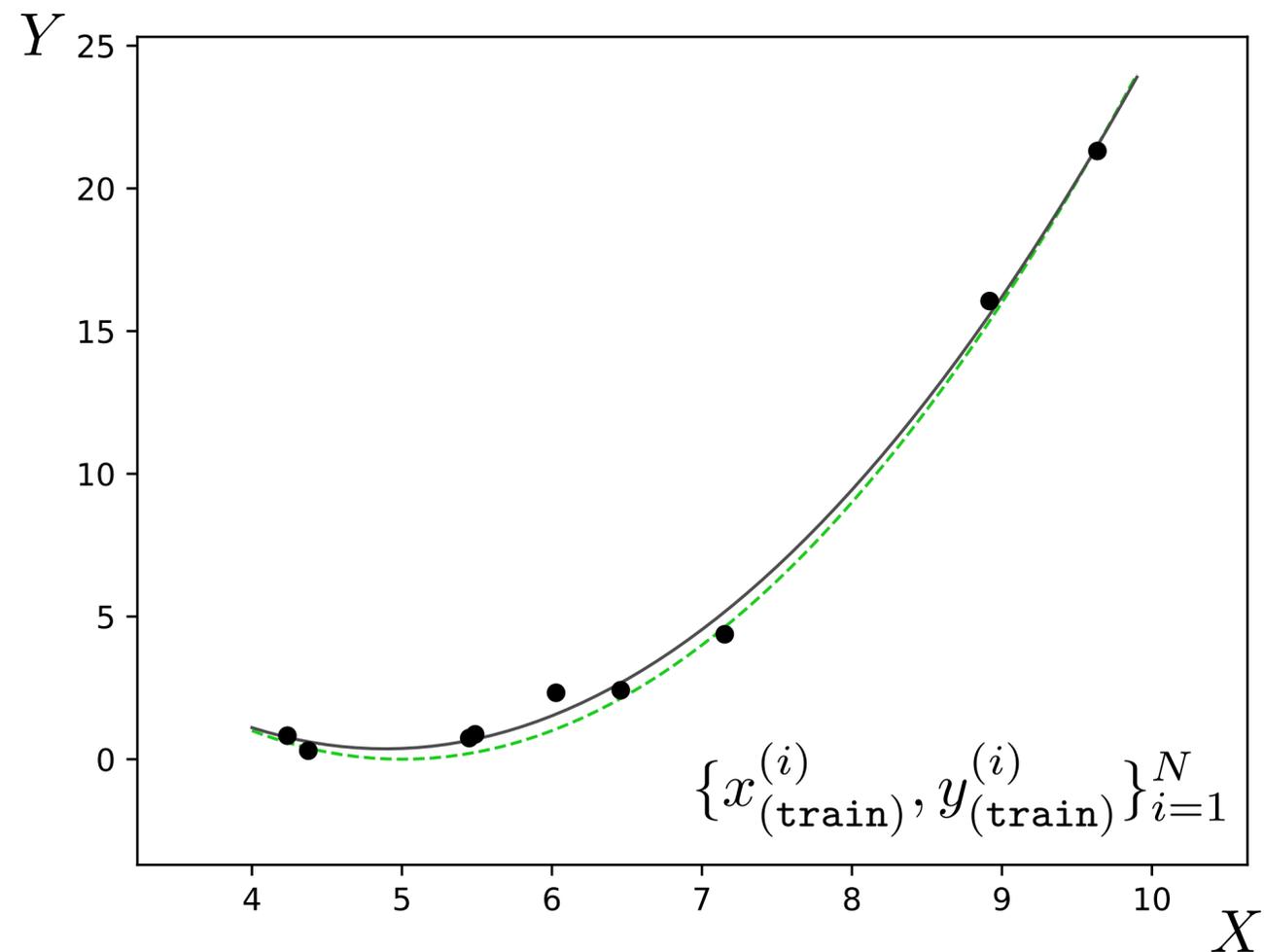
$$f_{\theta}(x) = \sum_{k=0}^K \theta_k x^k$$

Optimizer

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

$\rightarrow f$

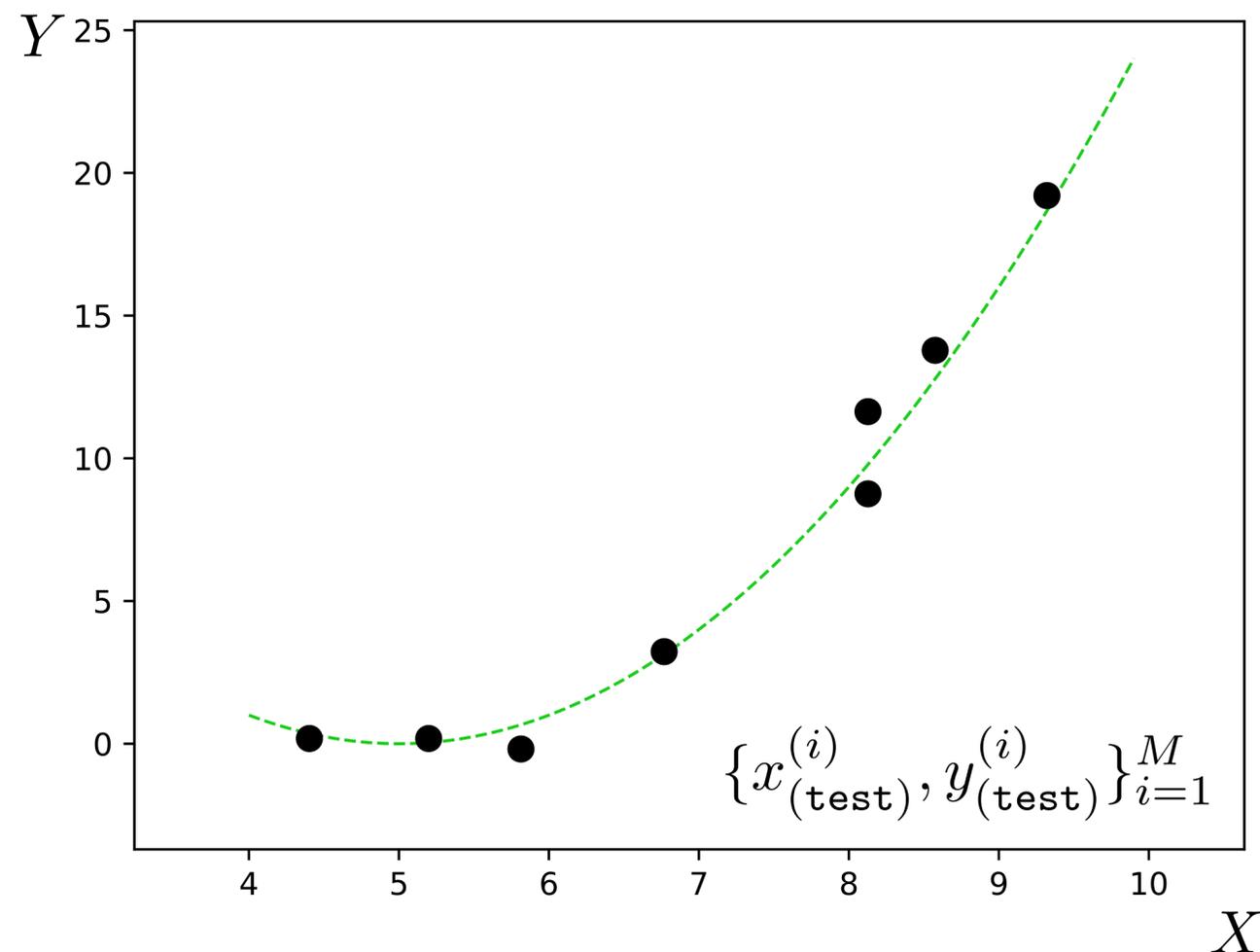
Training data



True data-generating process

p_{data}

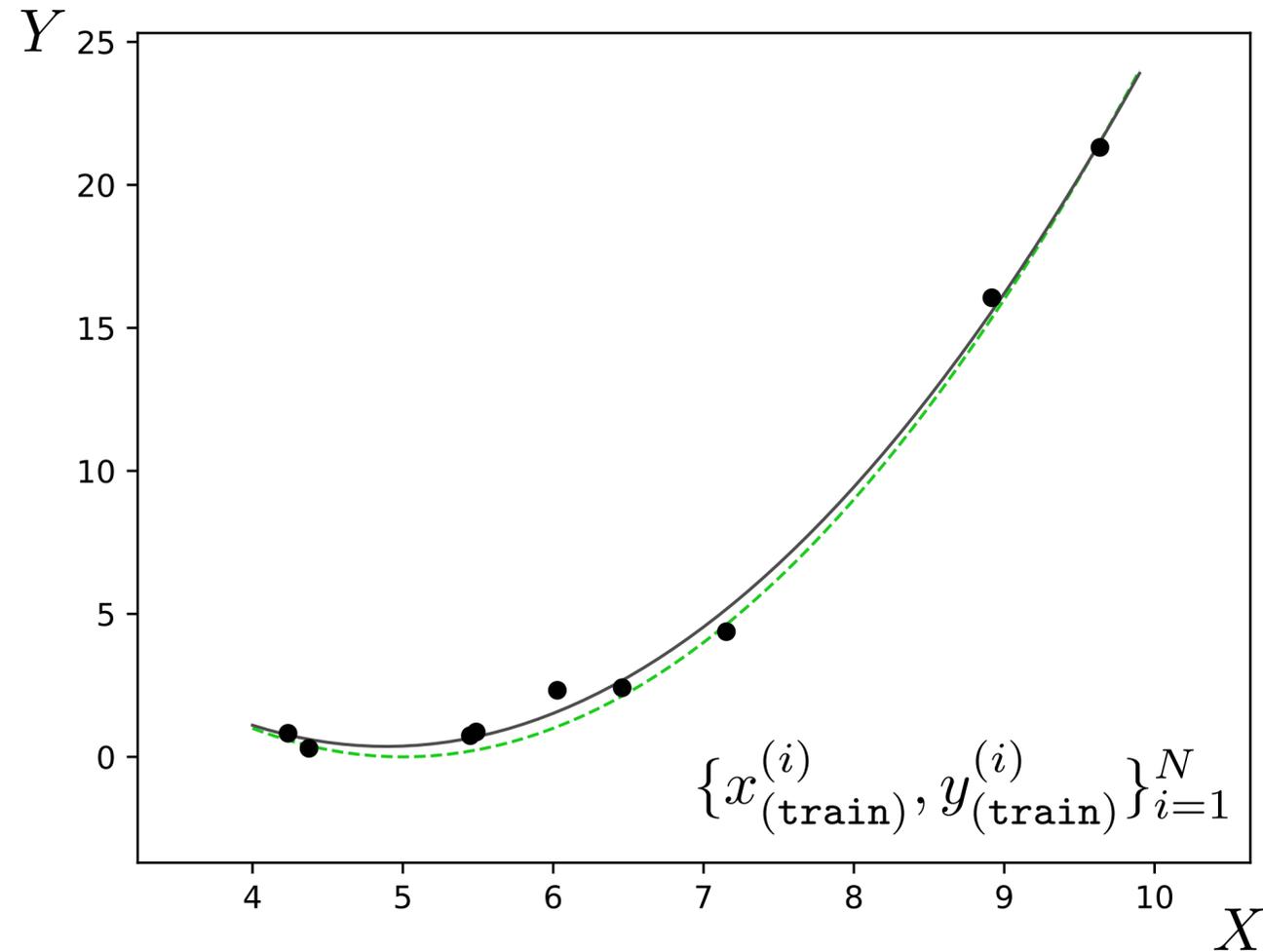
Test data



$$\{x_{(\text{train})}^{(i)}, y_{(\text{train})}^{(i)}\} \stackrel{\text{iid}}{\sim} p_{\text{data}}$$

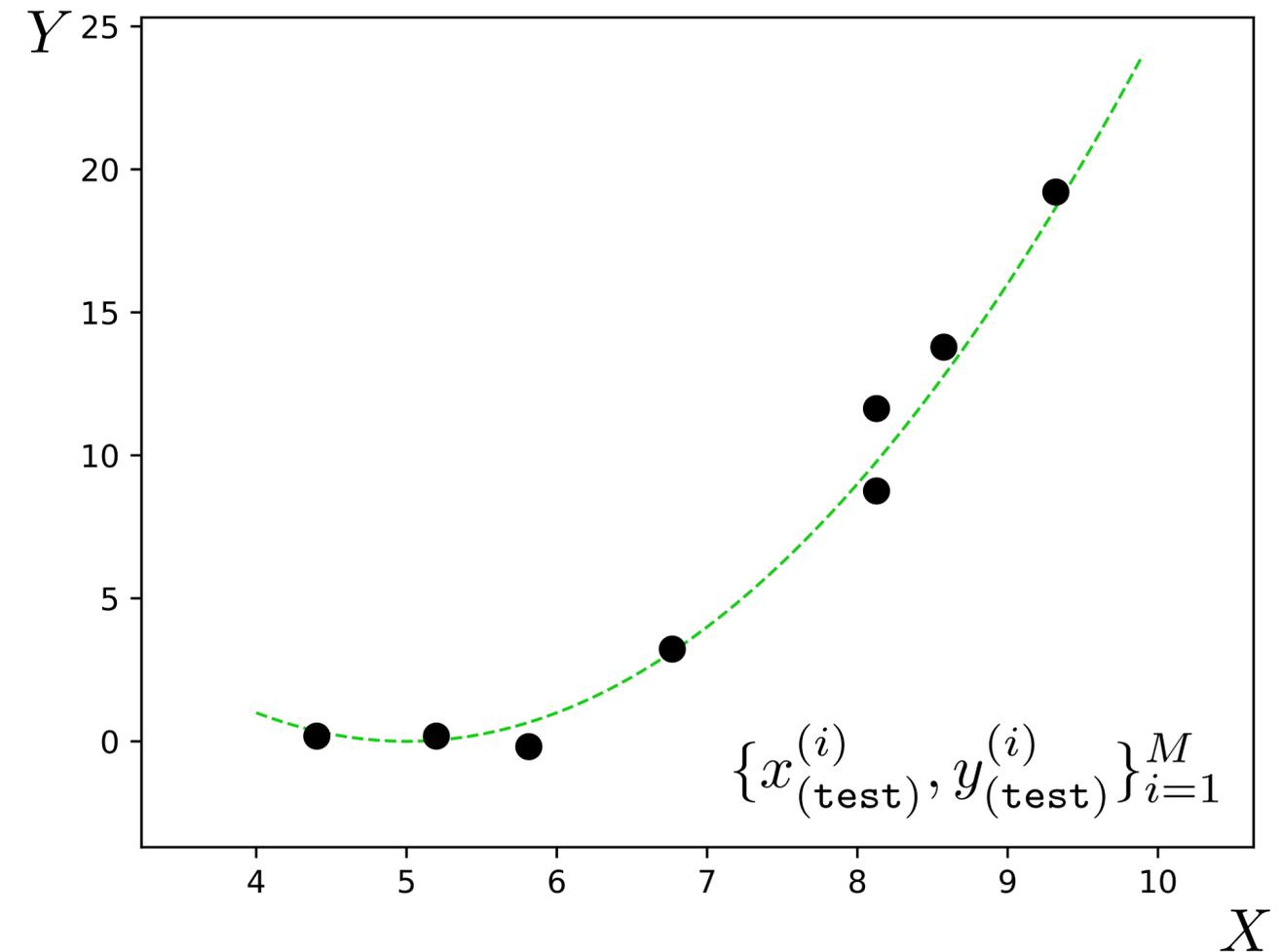
$$\{x_{(\text{test})}^{(i)}, y_{(\text{test})}^{(i)}\} \stackrel{\text{iid}}{\sim} p_{\text{data}}$$

Training data



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

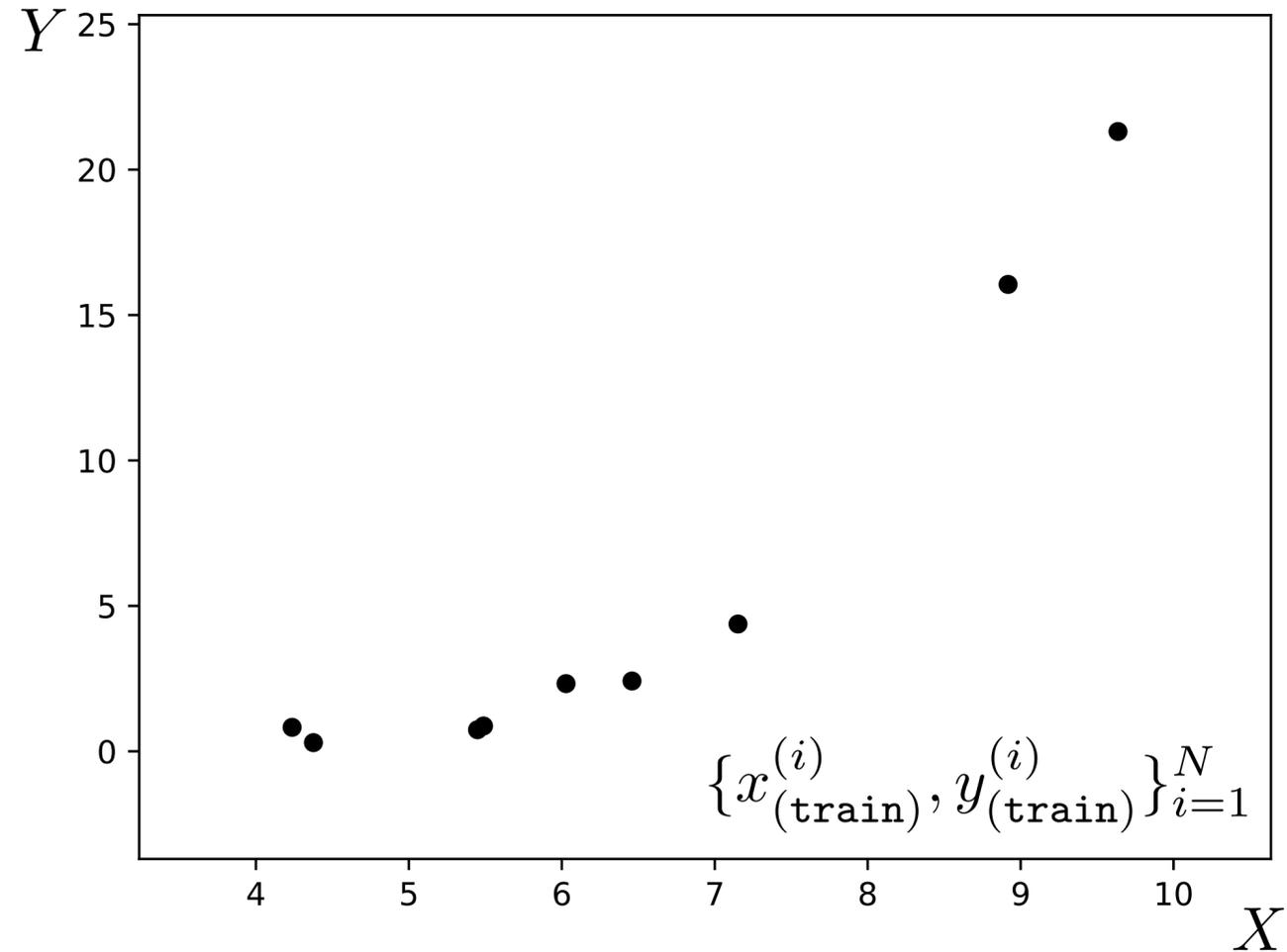
Test data



$$\{x_{(\text{train})}^{(i)}, y_{(\text{train})}^{(i)}\} \stackrel{\text{iid}}{\sim} p_{\text{data}}$$

$$\{x_{(\text{test})}^{(i)}, y_{(\text{test})}^{(i)}\} \stackrel{\text{iid}}{\sim} p_{\text{data}}$$

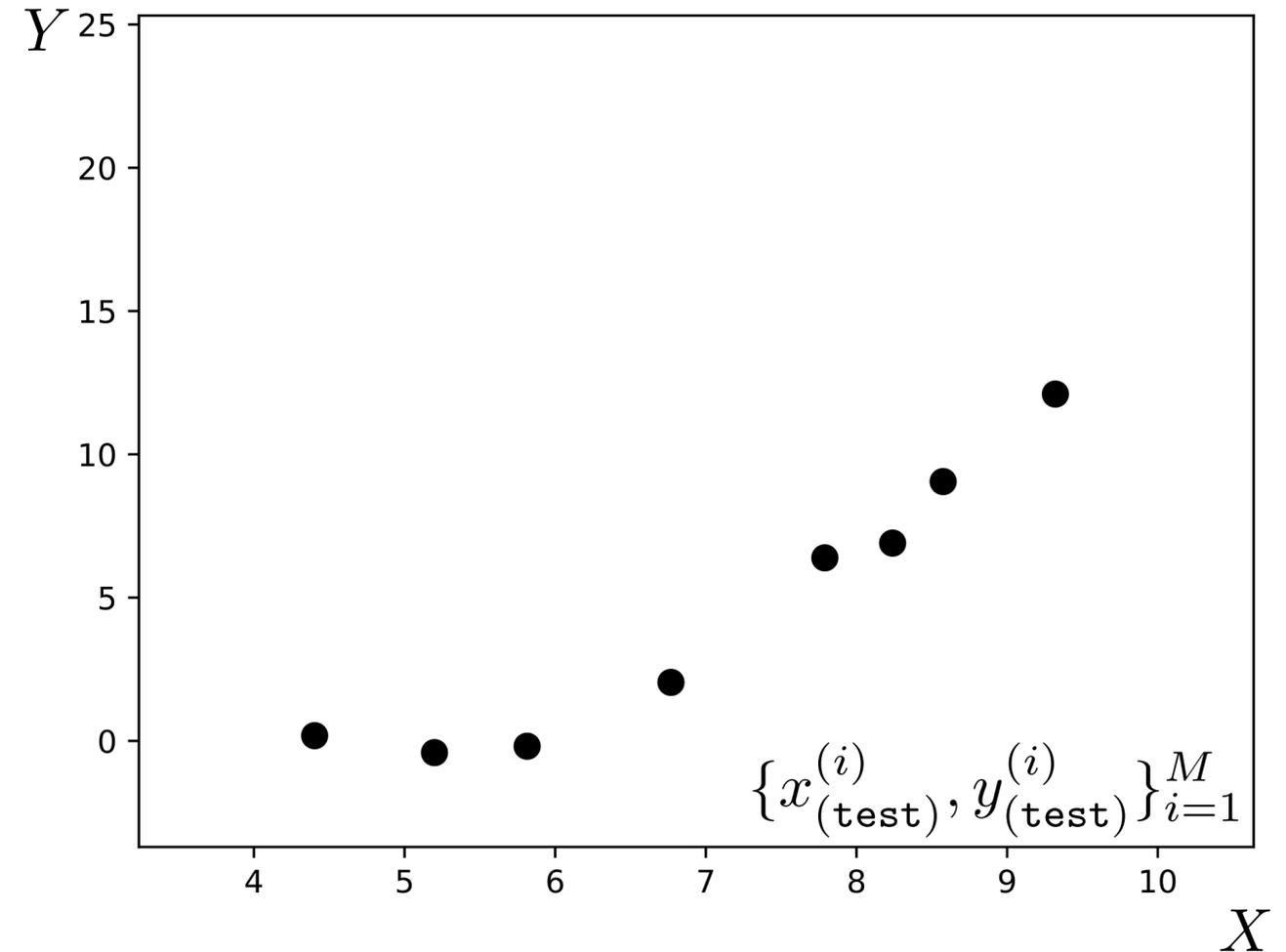
Training data



Much more commonly, we have

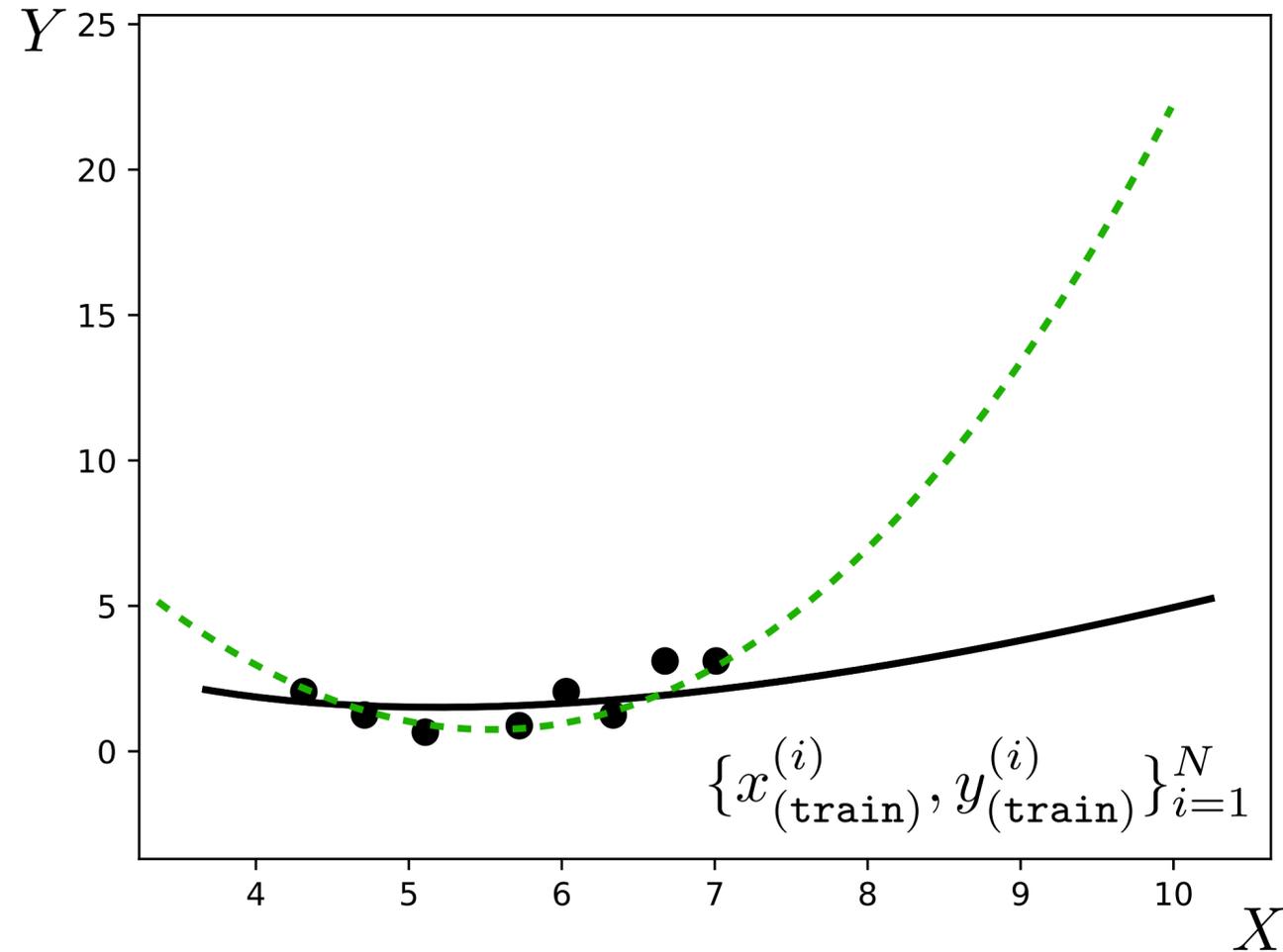
$$p_{\text{train}} \neq p_{\text{test}}$$

Test data

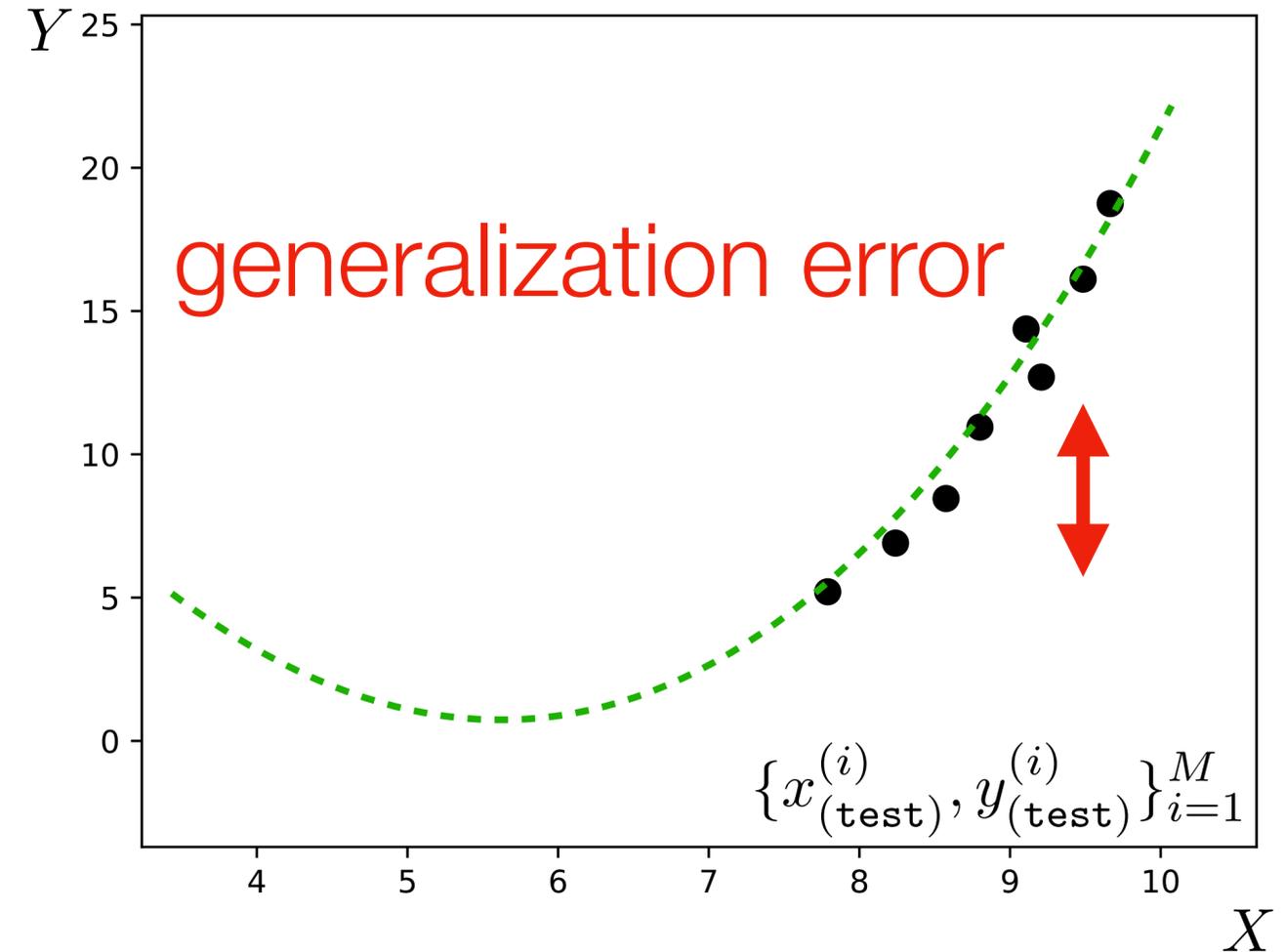


$$\{x_{(\text{train})}^{(i)}, y_{(\text{train})}^{(i)}\} \stackrel{\text{iid}}{\sim} p_{\text{train}}$$
$$\{x_{(\text{test})}^{(i)}, y_{(\text{test})}^{(i)}\} \stackrel{\text{iid}}{\sim} p_{\text{test}}$$

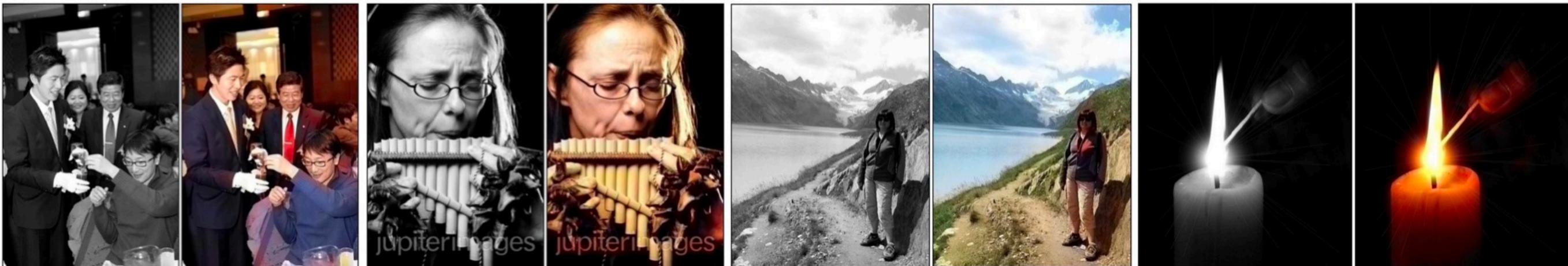
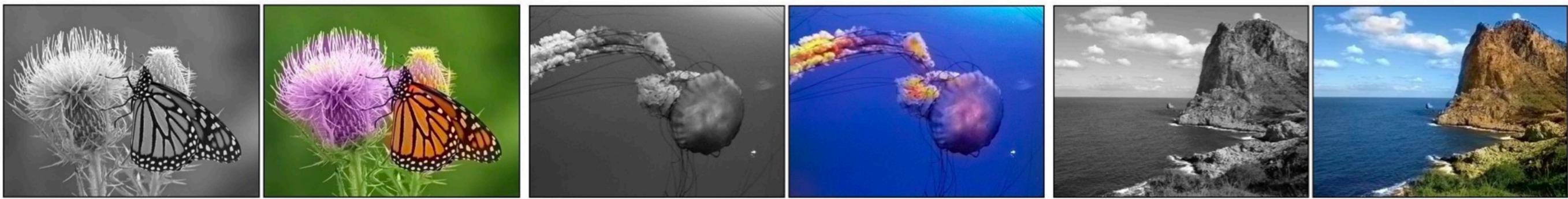
Training data



Test data

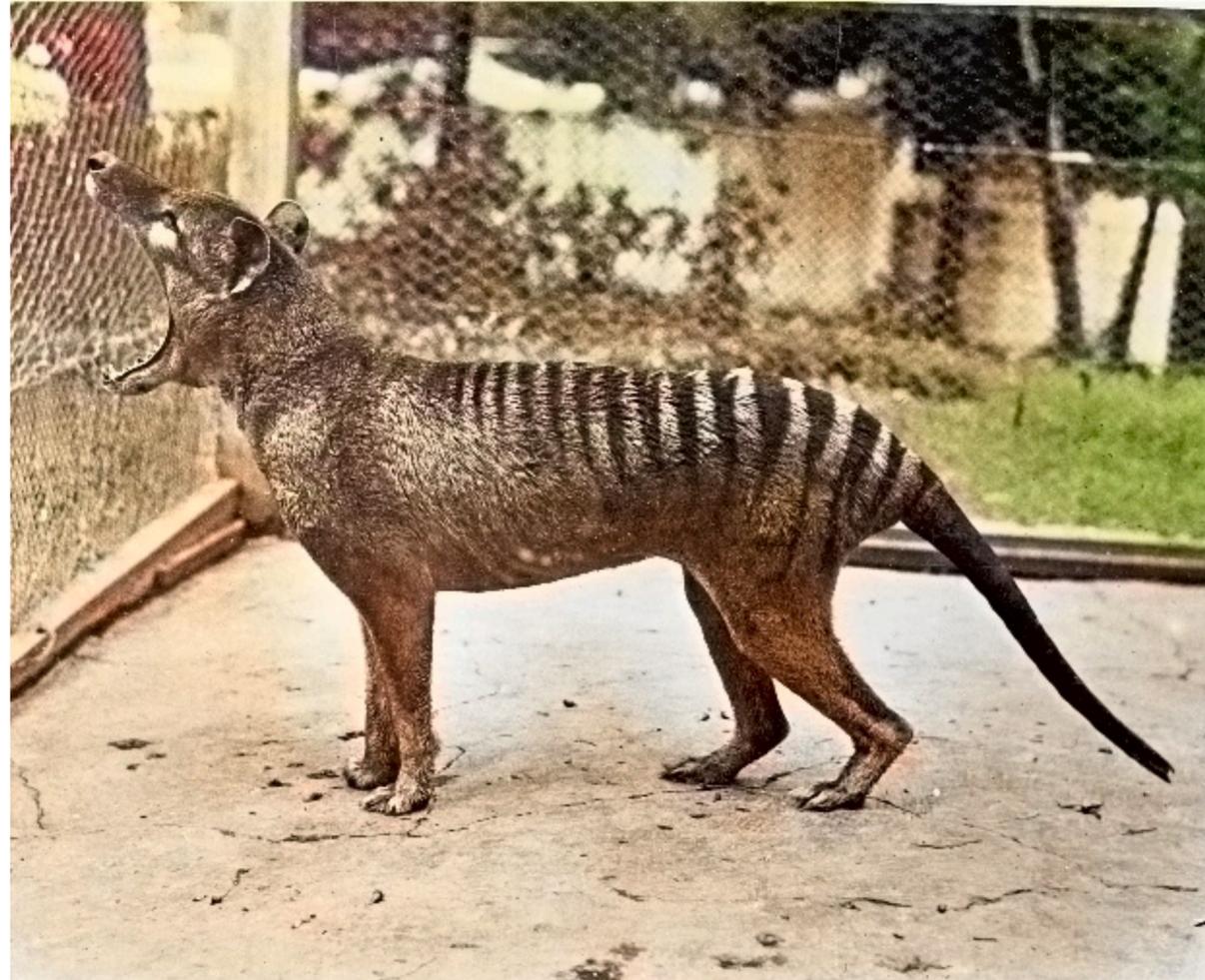


Our training data did cover the part of the distribution that was tested
(biased data)

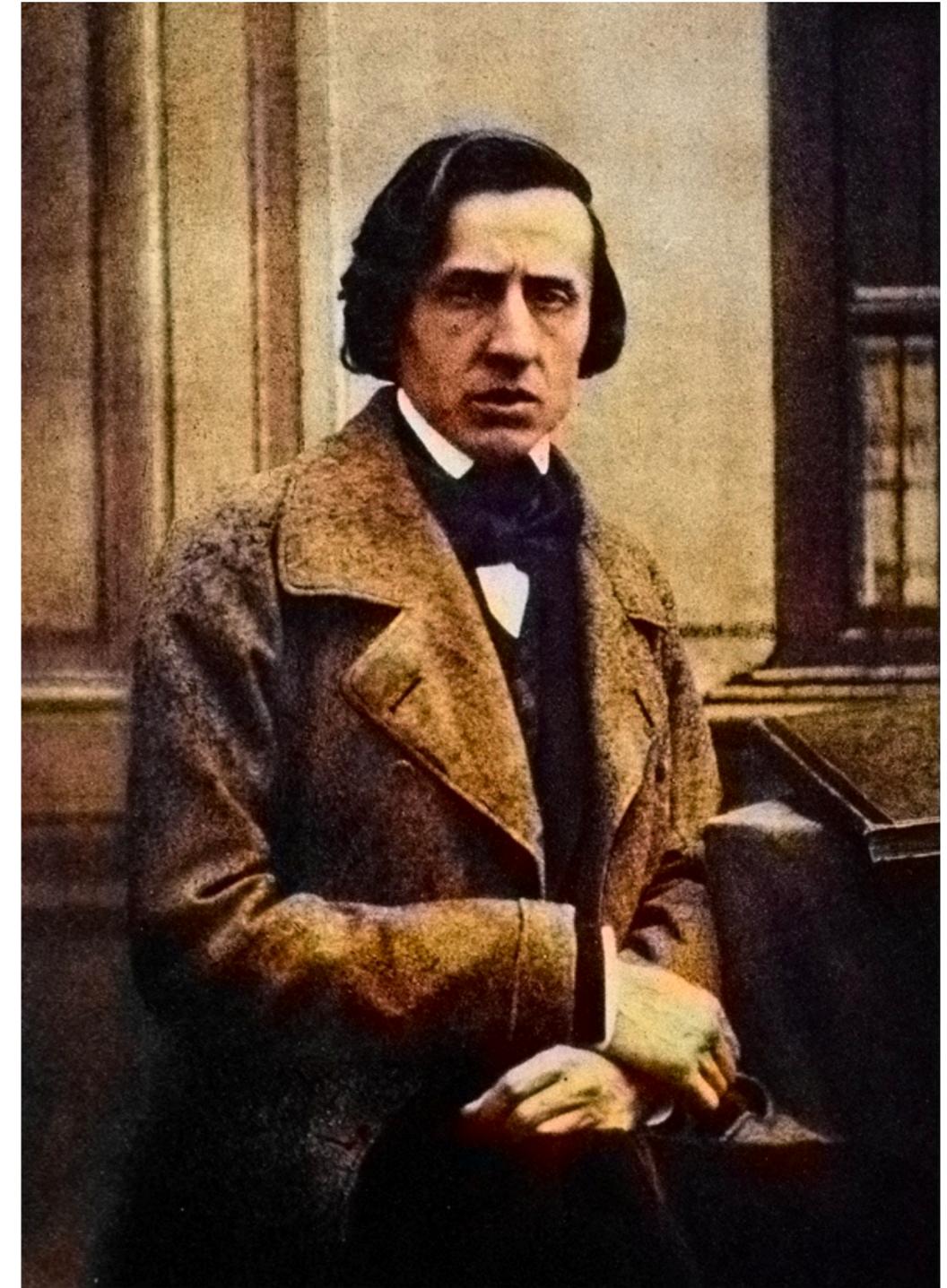




u/Rafael_P_S



Thylacine



Chopin

8. 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