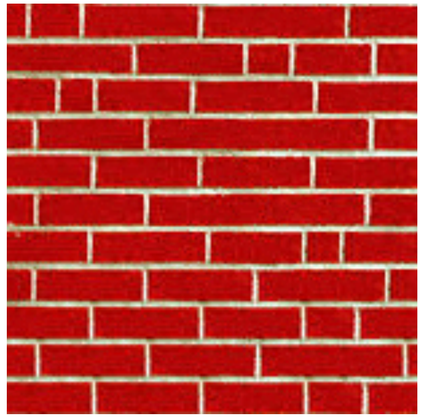
Machine learning in computer vision

Texture synthesis





Source: A. Efros

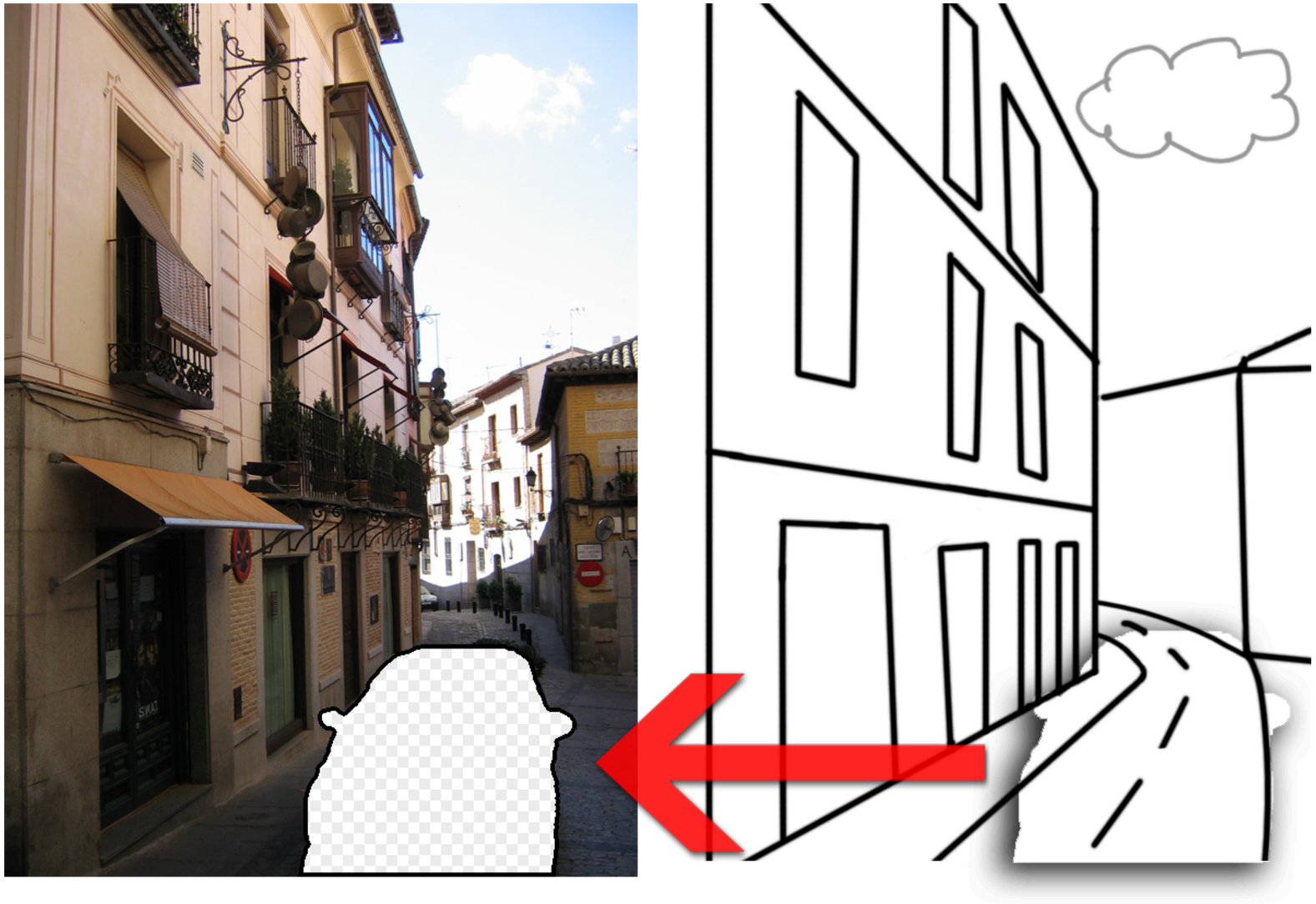
[Hays and Efros “Scene Completion Using Millions of Photographs”, 2007.]

Source: A. Efros

Simple inpainting [Efros & Leung 1999]

Source: A. Efros

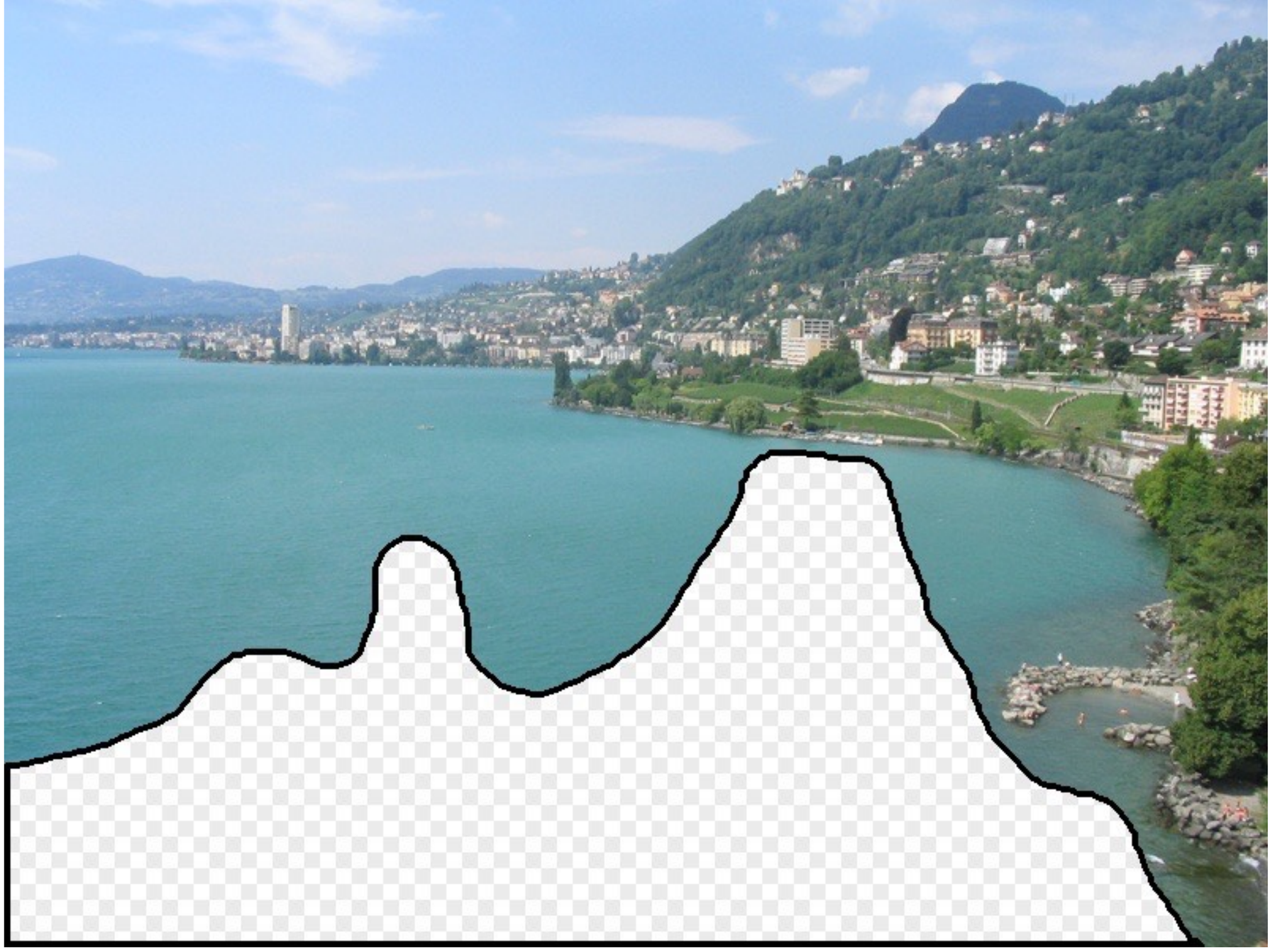
How would a human solve this?



Source: A. Efros

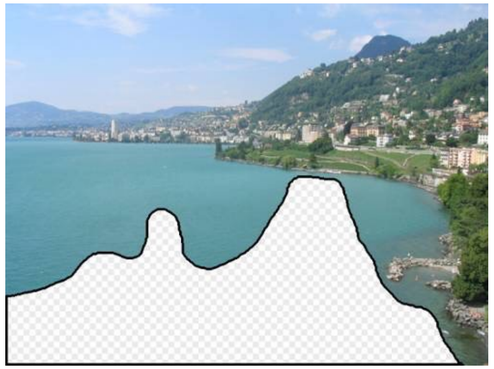
[Hays and Efros. Scene Completion Using Millions of Photographs. SIGGRAPH 2007.]

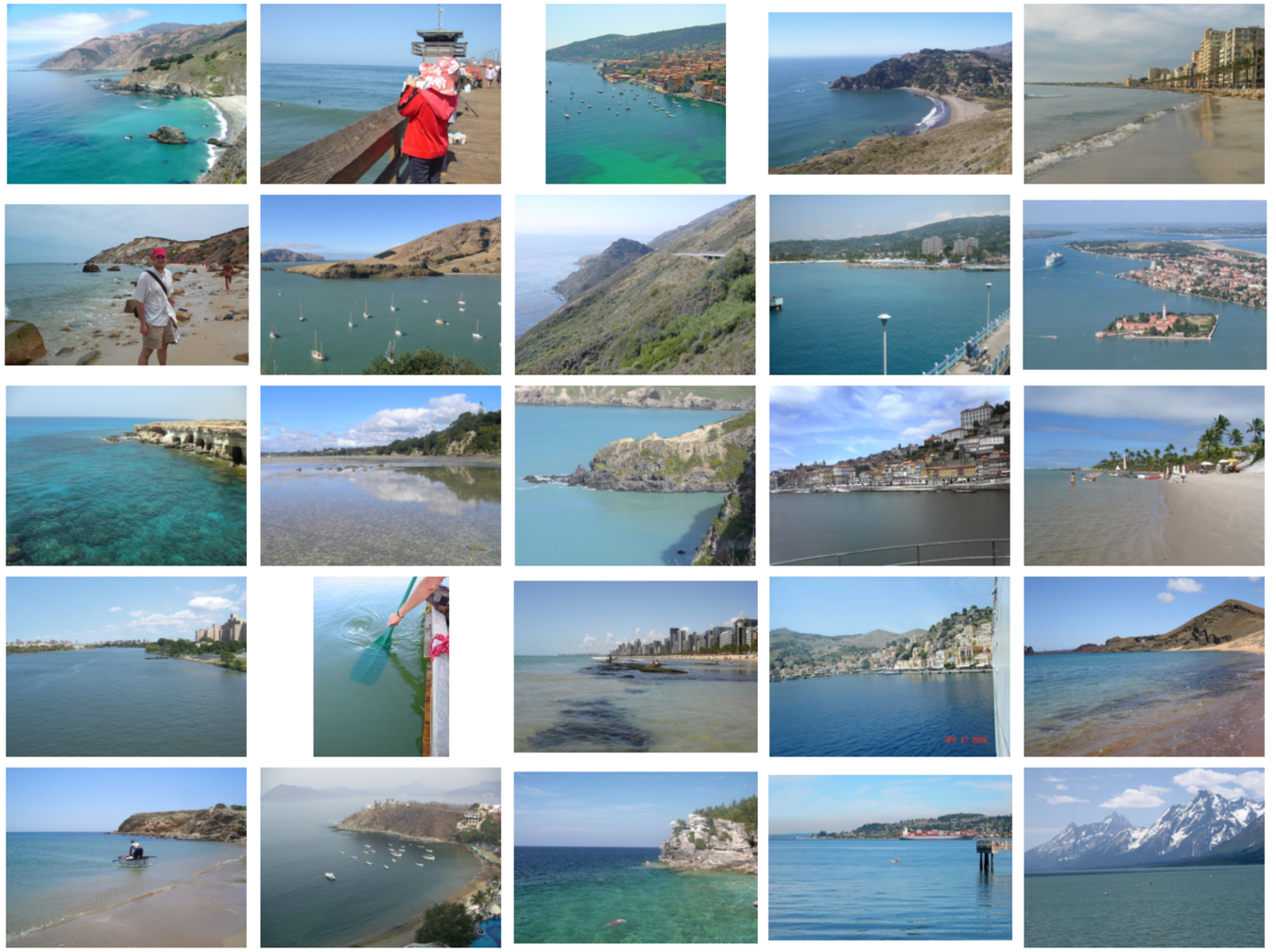
Source: A. Efros



Source: A. Efros

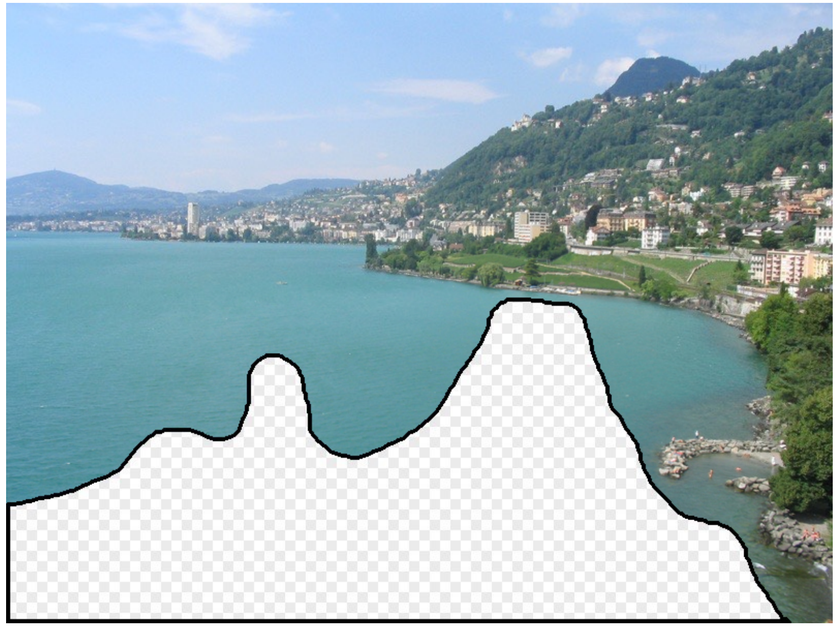
Source: A. Efros





… 200 total

Source: A. Efros



Source: A. Efros

Source: A. Efros

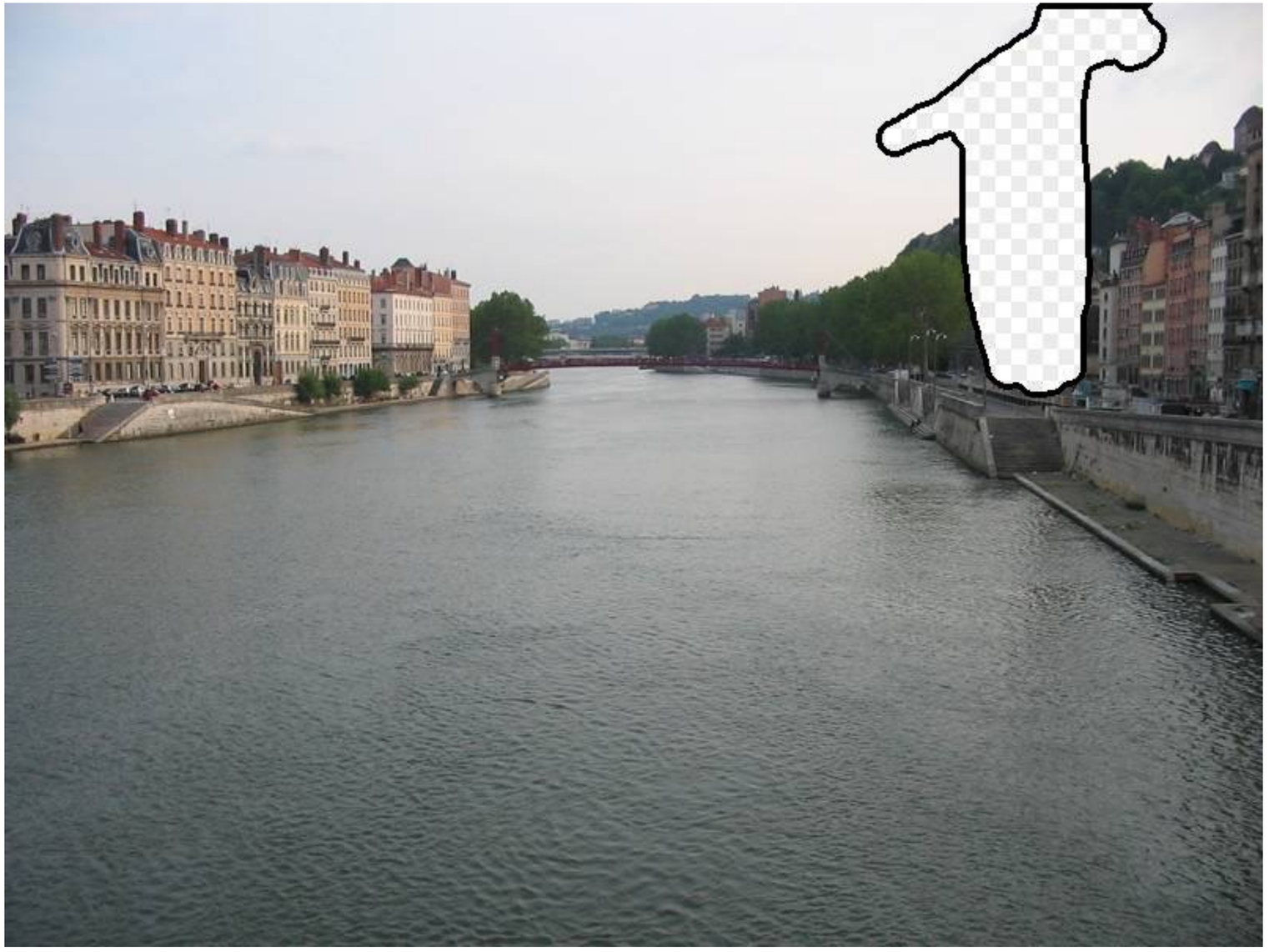
Source: A. Efros

Source: A. Efros

Source: A. Efros

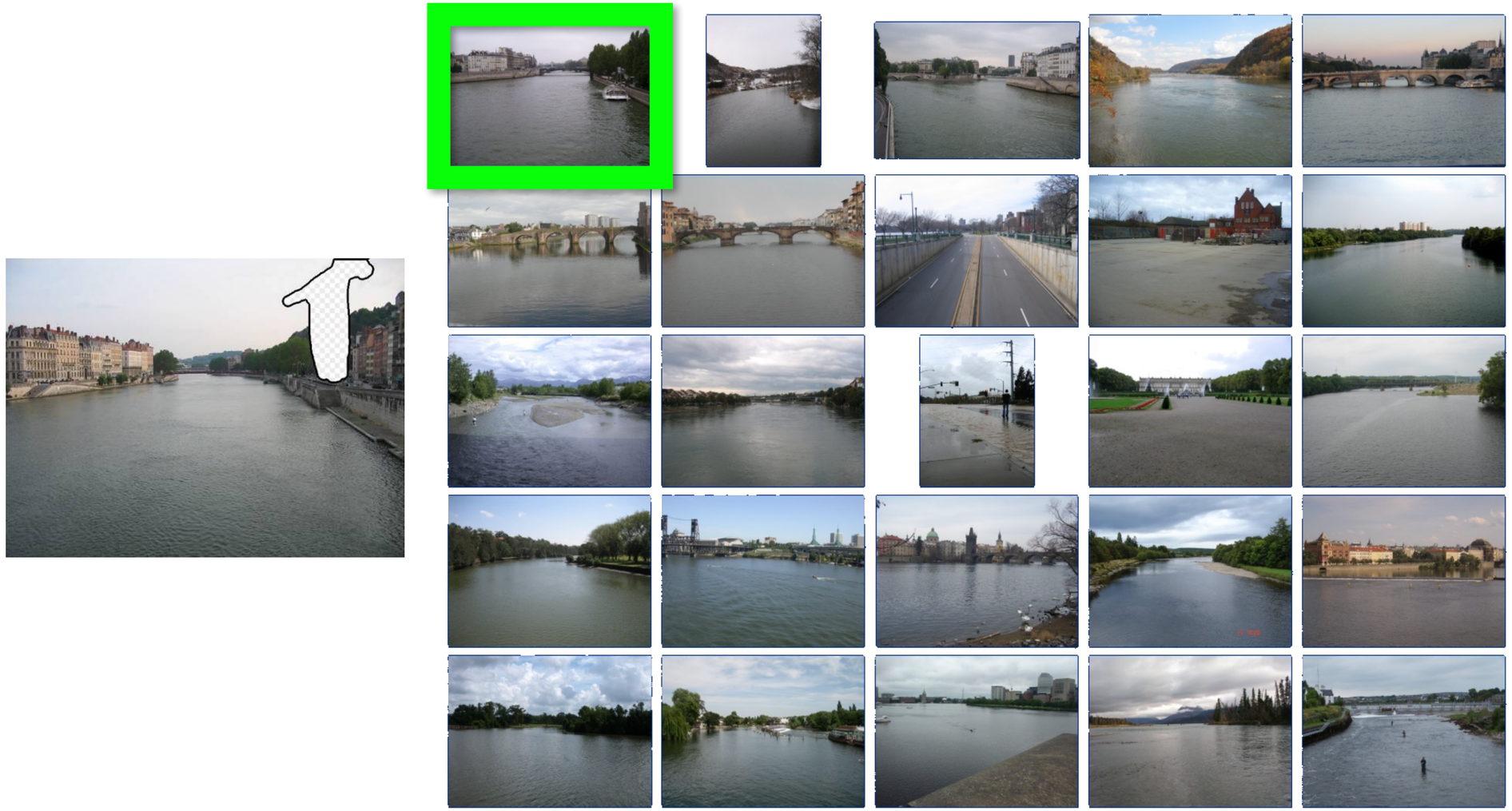
Source: A. Efros

Source: A. Efros



Source: A. Efros

Source: A. Efros



… 200 scene matches

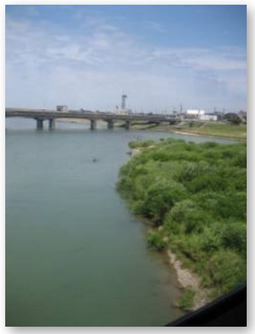
Source: A. Efros

Source: A. Efros

Source: A. Efros

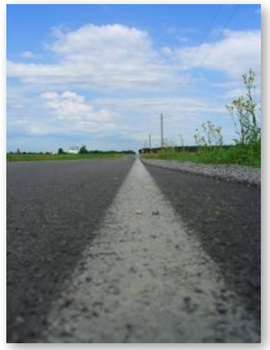
Why does this work?

Source: A. Efros



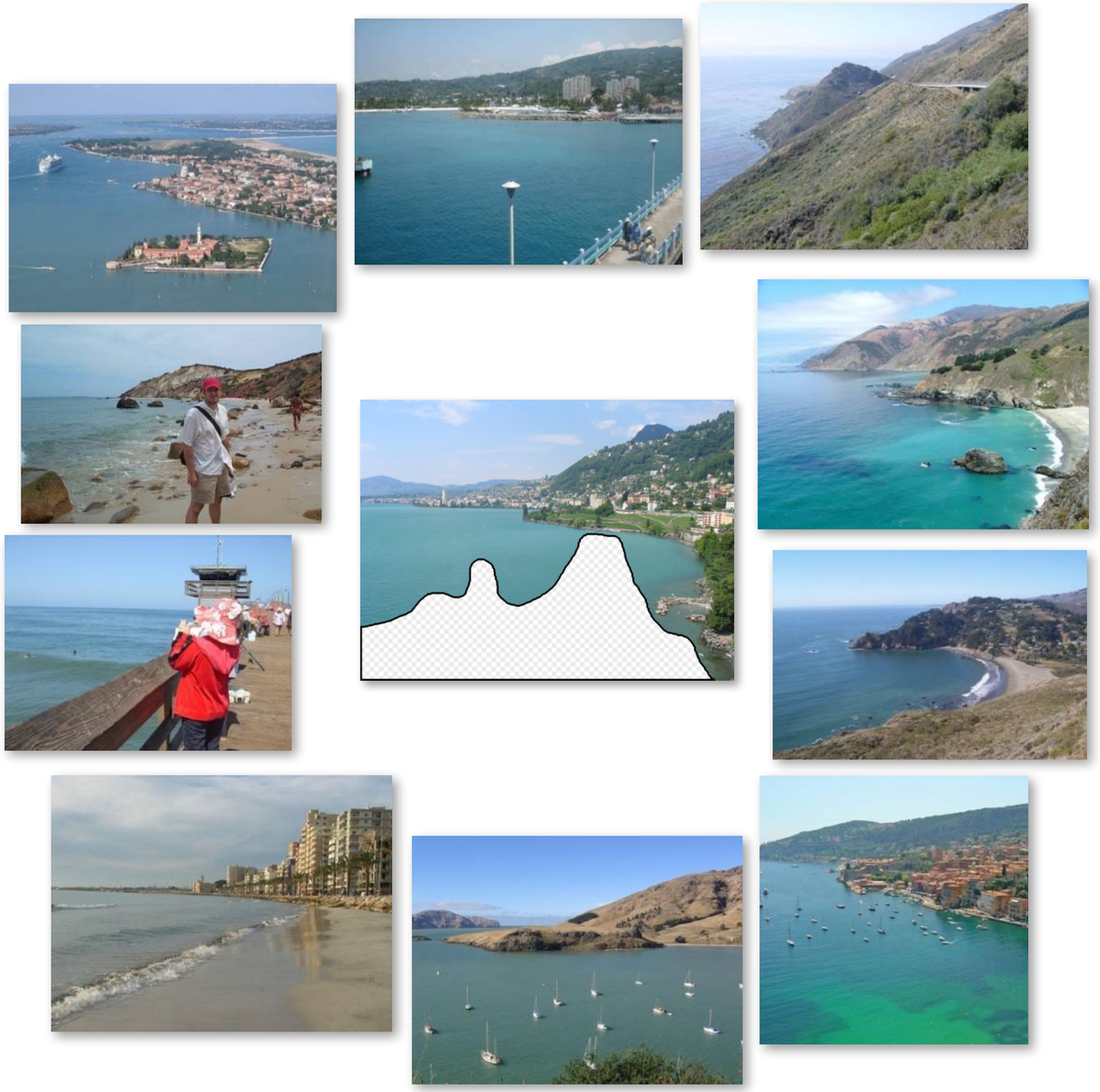





Nearest neighbors from a
collection of 20 thousand images

Source: A. Efros



Nearest neighbors from a collection of 2 million images

Source: A. Efros

“Unreasonable Effectiveness of Data”

[Halevy, Norvig, Pereira 2009]

Parts of our world can be explained by elegant mathematics

physics, chemistry, astronomy, etc.

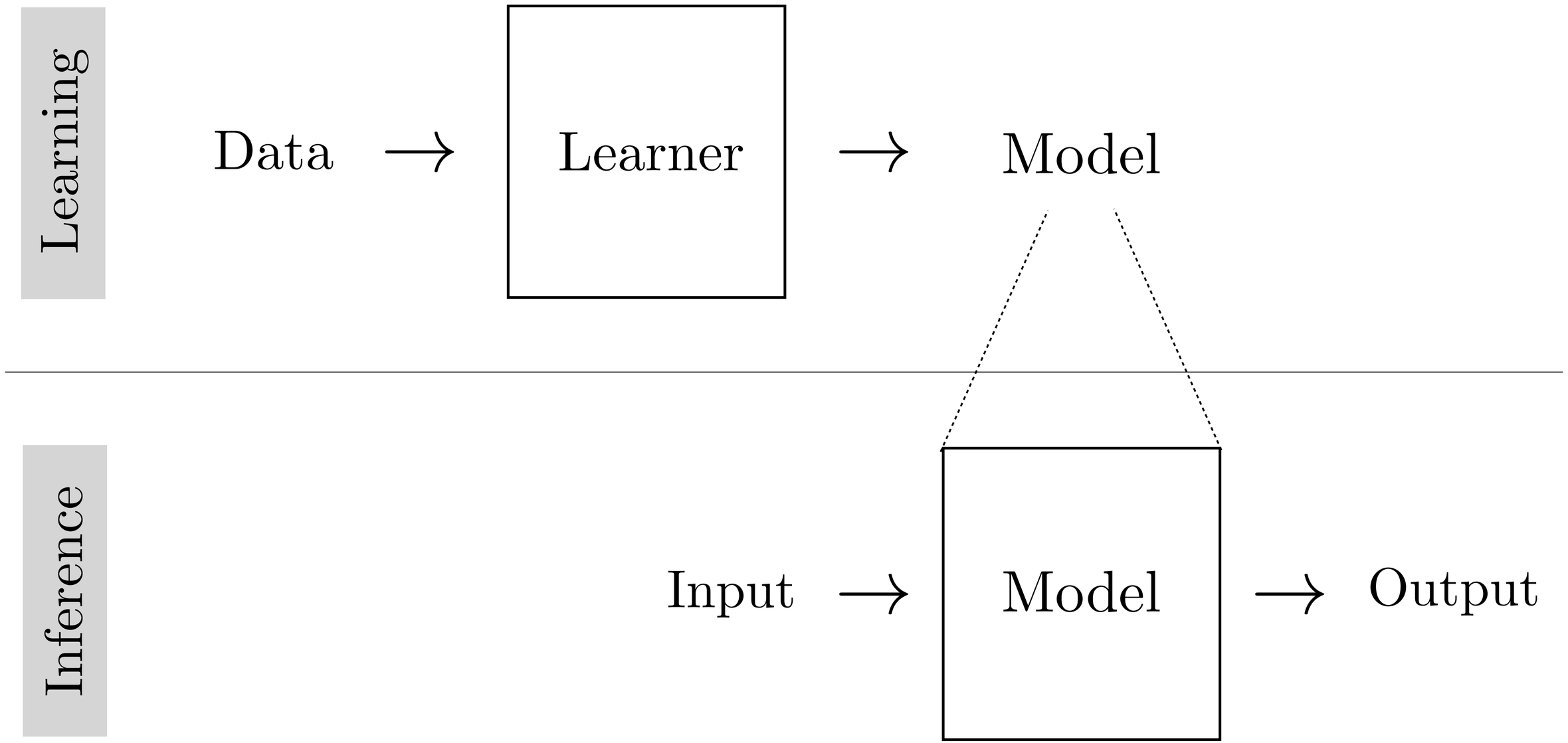
But much cannot

psychology, economics, genetics, etc.

Enter The Data!

“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…”

Source: A. Efros



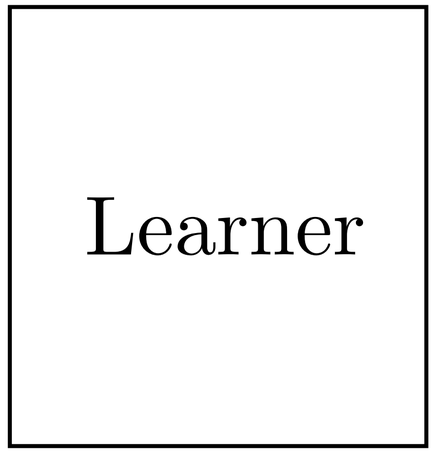
Source: Isola, Torralba, Freeman

Learning from examples

(aka **supervised learning**)

Training data

$$\{x_1, y_1\}$$
$$\{x_2, y_2\}$$
$$\{x_3, y_3\}$$
$$\ldots$$



$$\rightarrow \quad f : X \rightarrow Y$$

Source: Isola, Torralba, Freeman

Learning from examples

(aka **supervised learning**)

Training data

X Y

"bowling alley"

"desert"

$\vdots$

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

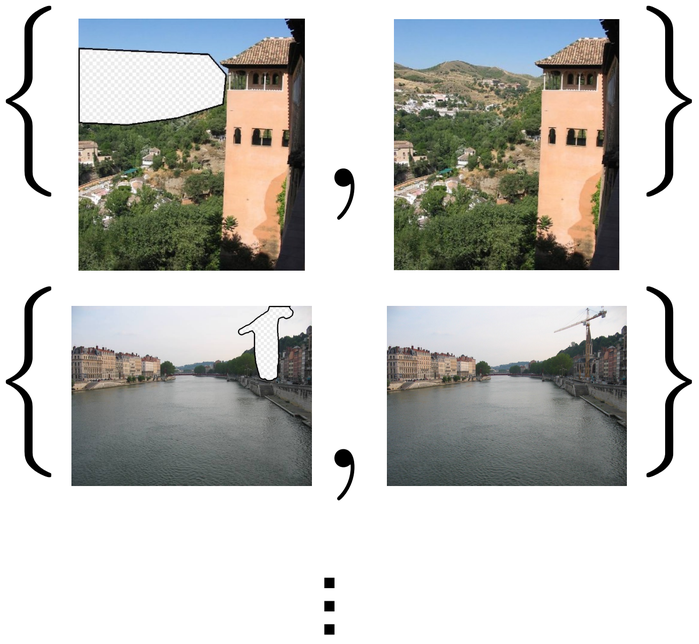
Source: Isola, Torralba, Freeman

Learning from examples

(aka **supervised learning**)

Training data

X Y



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

Source: Isola, Torralba, Freeman

Case study #1: Nearest neighbor

Nearest neighbor

Input image

$$\longrightarrow \begin{bmatrix} 0.5 \\ 0.1 \\ -0.4 \\ \cdots \\ 0.9 \end{bmatrix}$$

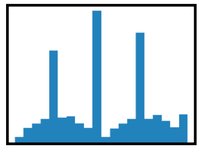
Feature vector x_q

Some feature ideas:

tiny image

normalized tiny image

$$\frac{x - \mu}{\sigma}$$



θ

edge orientation histogram

We'll discuss better options later in course.

Nearest neighbor

X Y

Input image

x_q

{  , "bowling alley" }

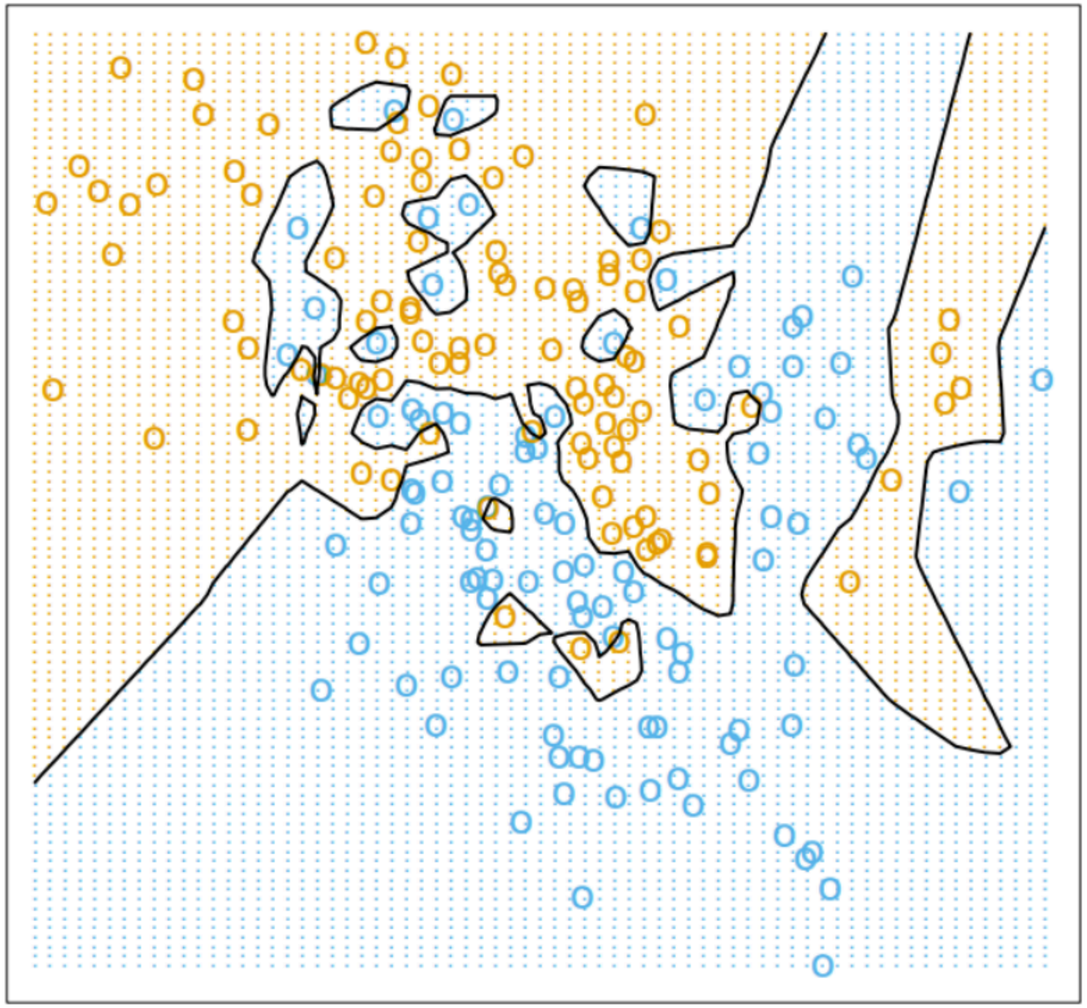
{  , "beach" }

⋮

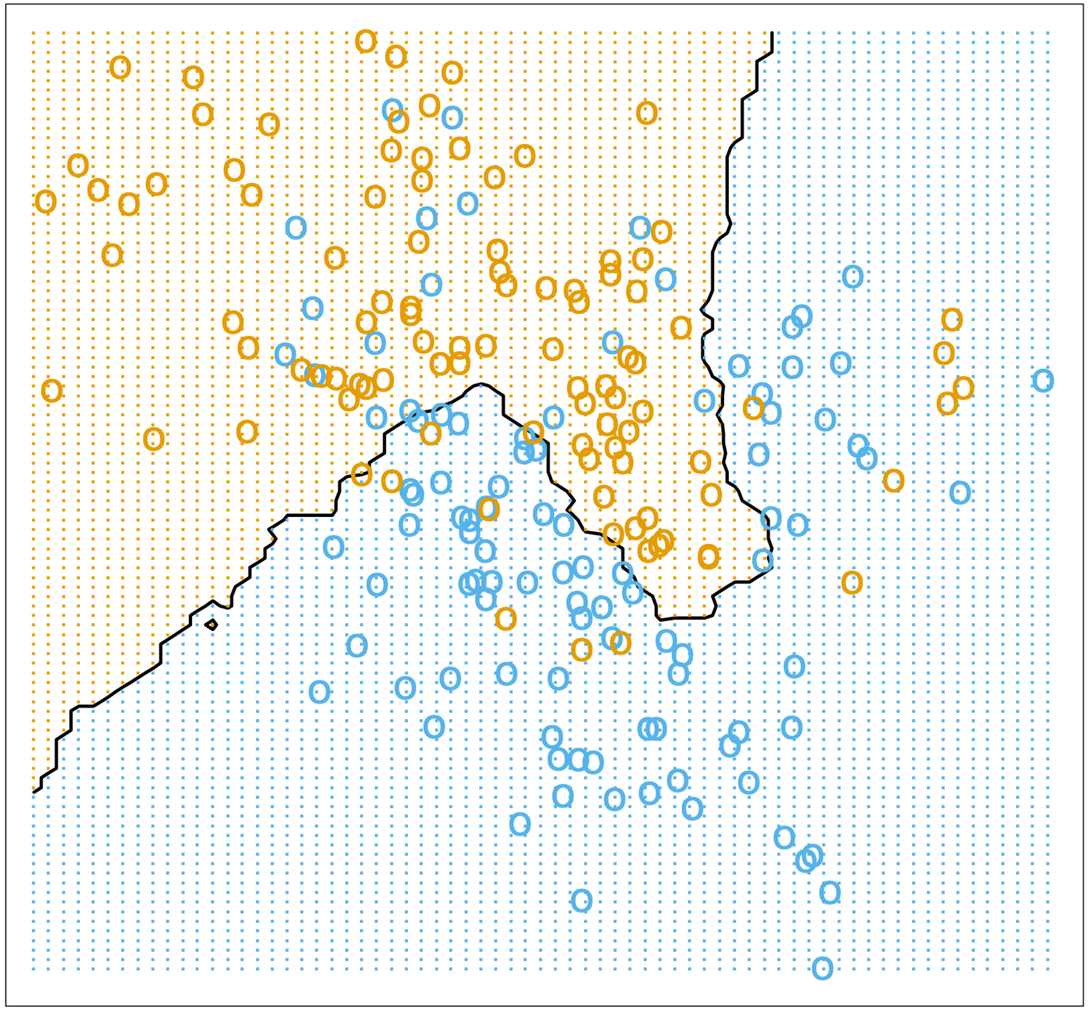
{  , "desert" }

Take label from:
$\text{argmin}_i ||x_q - x_i||$

K-nearest neighbor:
Take most common label in **k** closest examples



$k = 1$

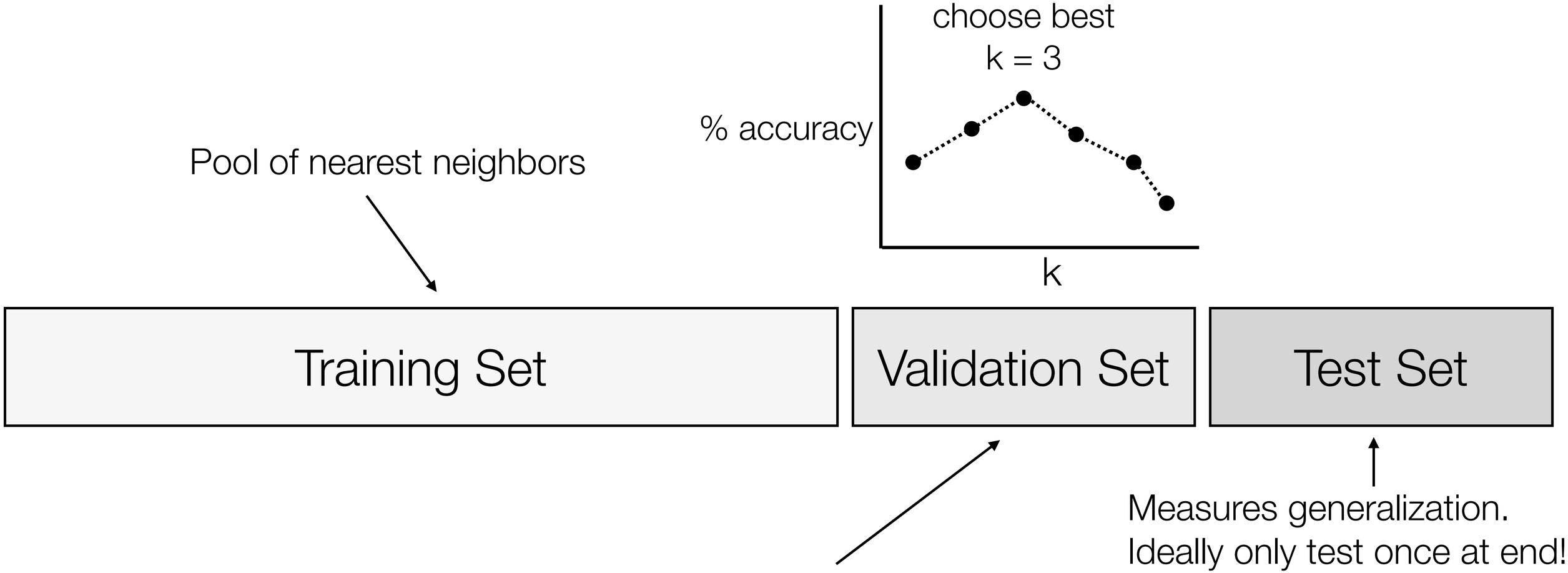


$k = 15$

Source: [Hastie et al. “Elements of Statistical Learning”, 2001]

How do we choose **k**?

How do we choose **k**?



Can work well when datasets are really big

Input image

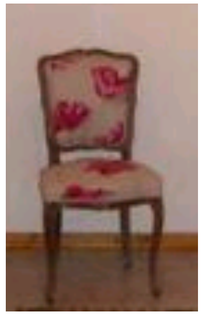
Colorized output

Nearest Neighbors

[Torralba et al. “80 million tiny images”, 2008]

Nearest Neighbor

- Hard to define similarity metric

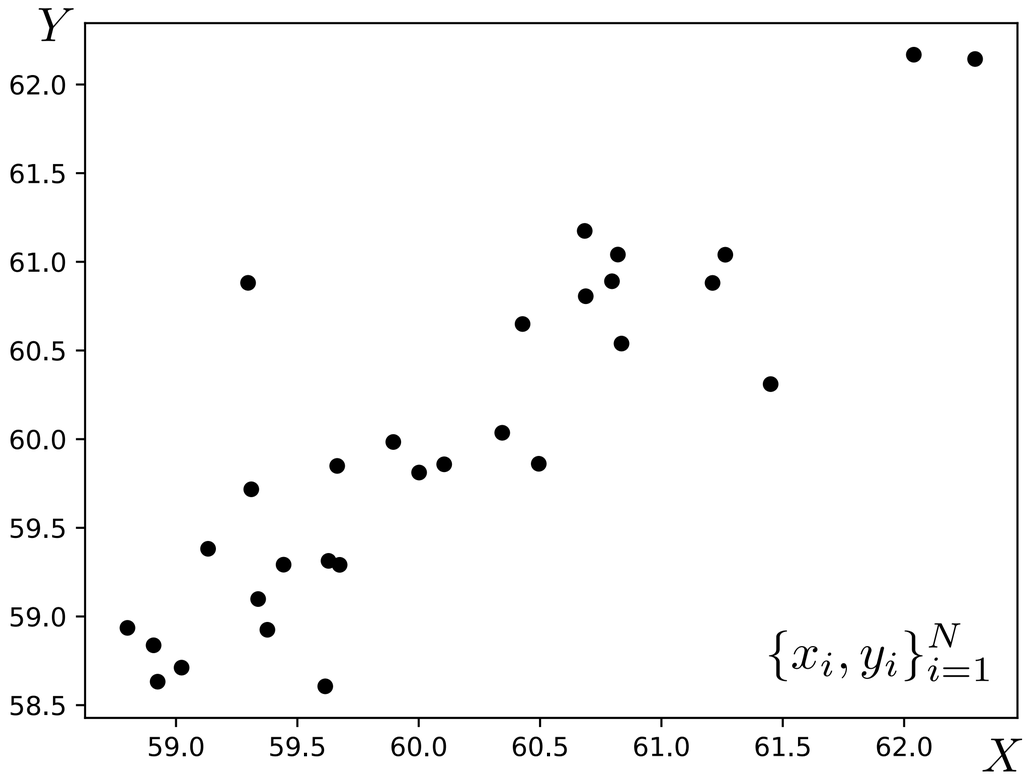
$$|| \quad \text{  } \quad - \quad \text{  } \quad || \quad = \quad ?$$

- Pays attention to all features equally
- Can be very slow: $O(nd)$

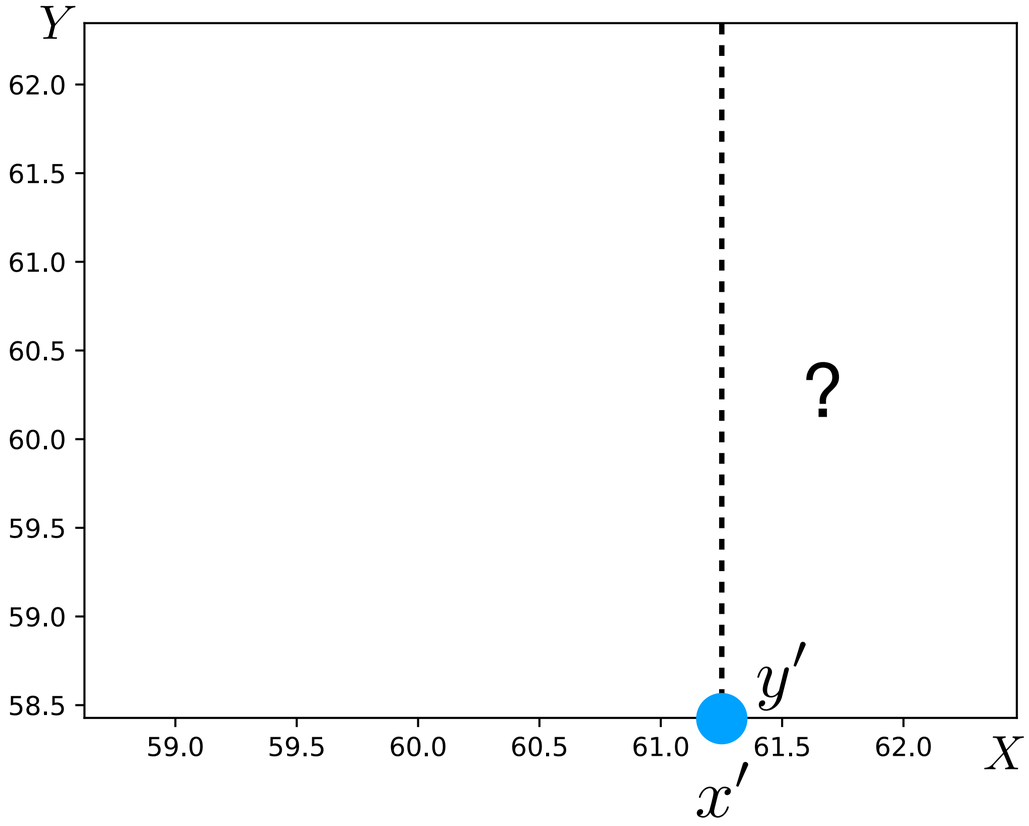
Case study #2: Linear least squares

Source: Isola, Torralba, Freeman

Training data

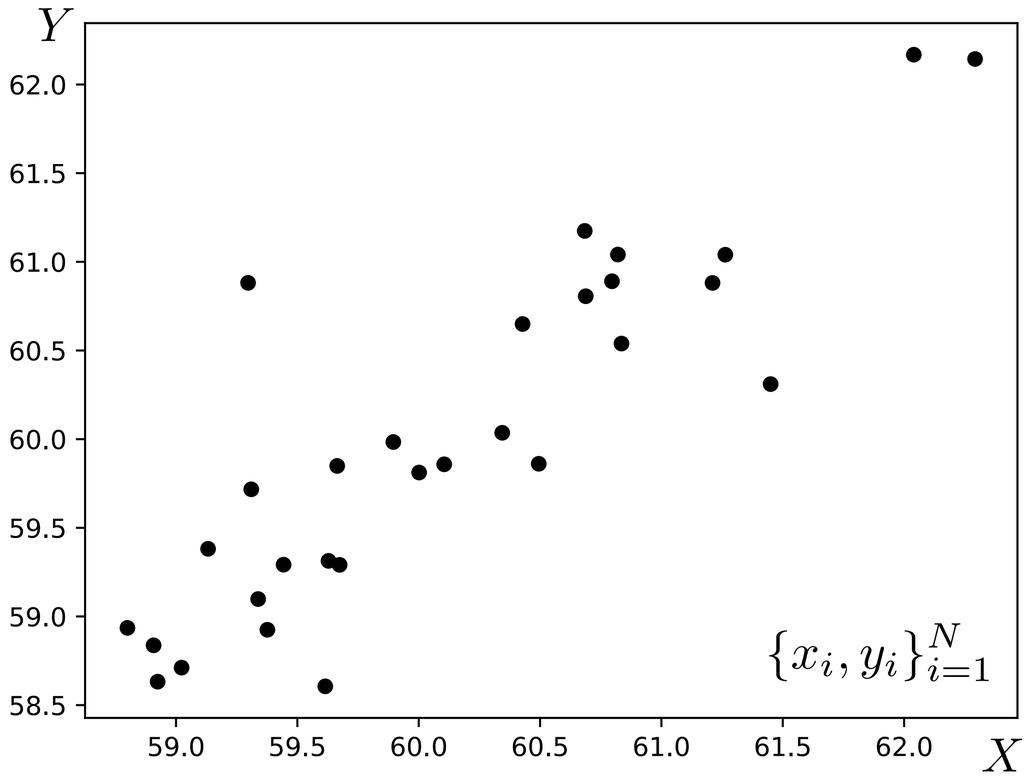


Test query



Source: Isola, Torralba, Freeman

Training data



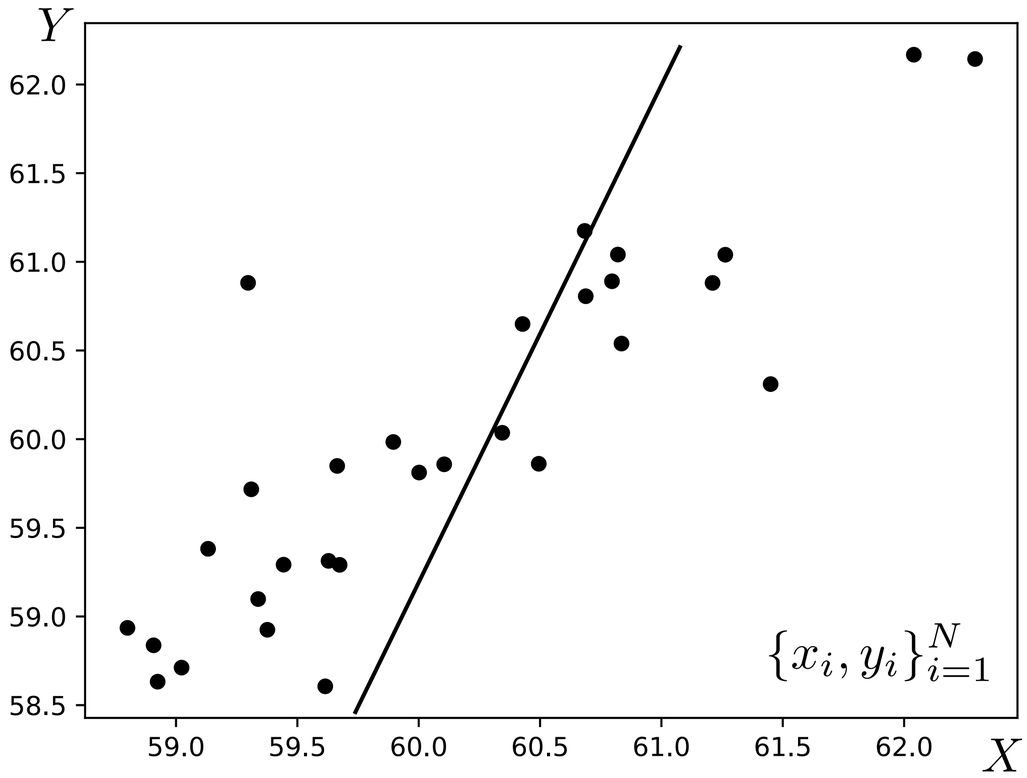
$\longrightarrow$ Learner $\longrightarrow$ $f_\theta(x) = \theta_1 x + \theta_0$

Hypothesis space

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

Source: Isola, Torralba, Freeman

Training data



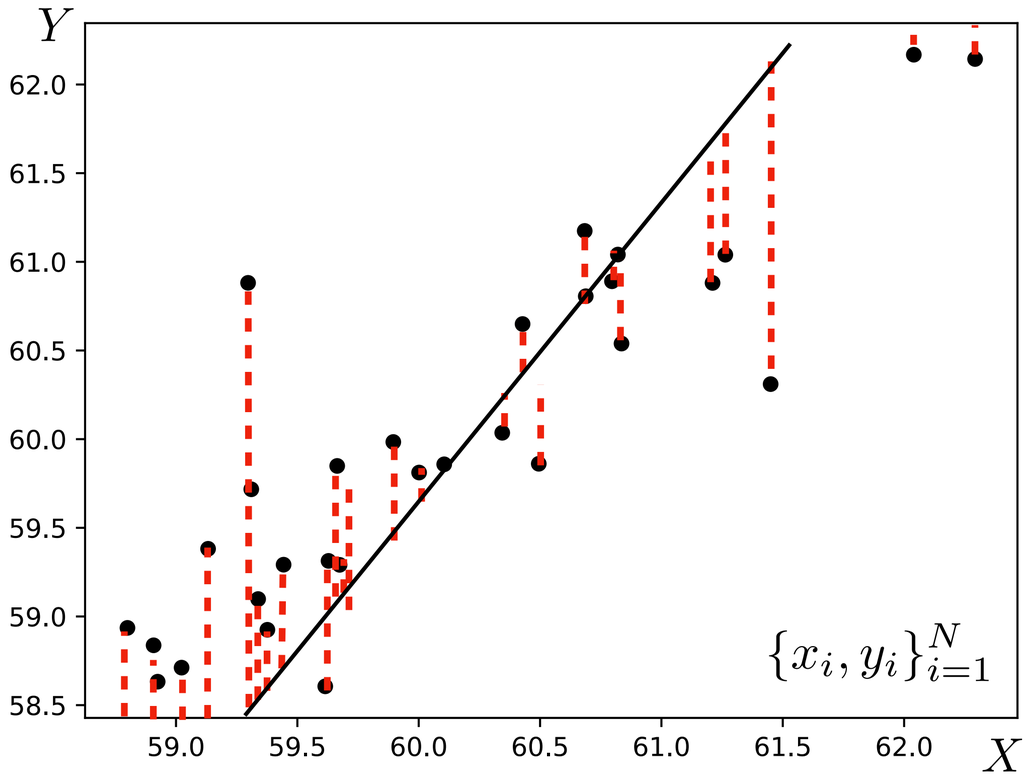
Search for the **parameters**, $\theta = \{\theta_0, \theta_1\}$, that best fit the data.

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

Best fit in what sense?

Source: Isola, Torralba, Freeman

Training data



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

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

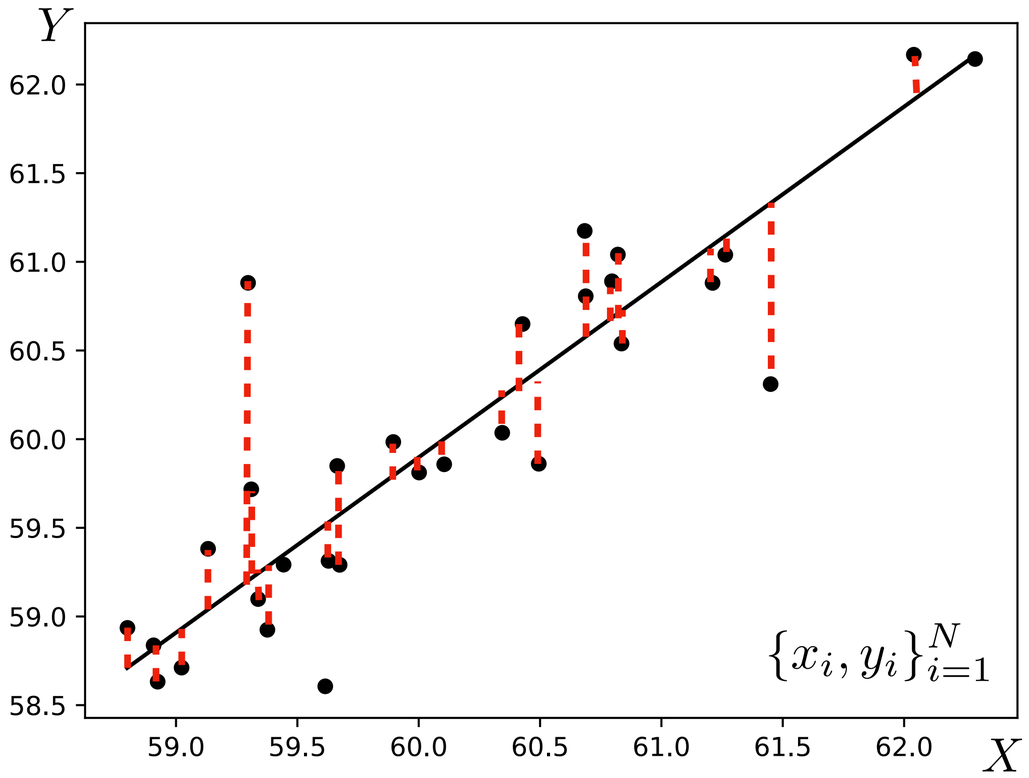
Best fit in what sense?

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

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

Source: Isola, Torralba, Freeman

Training data



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

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

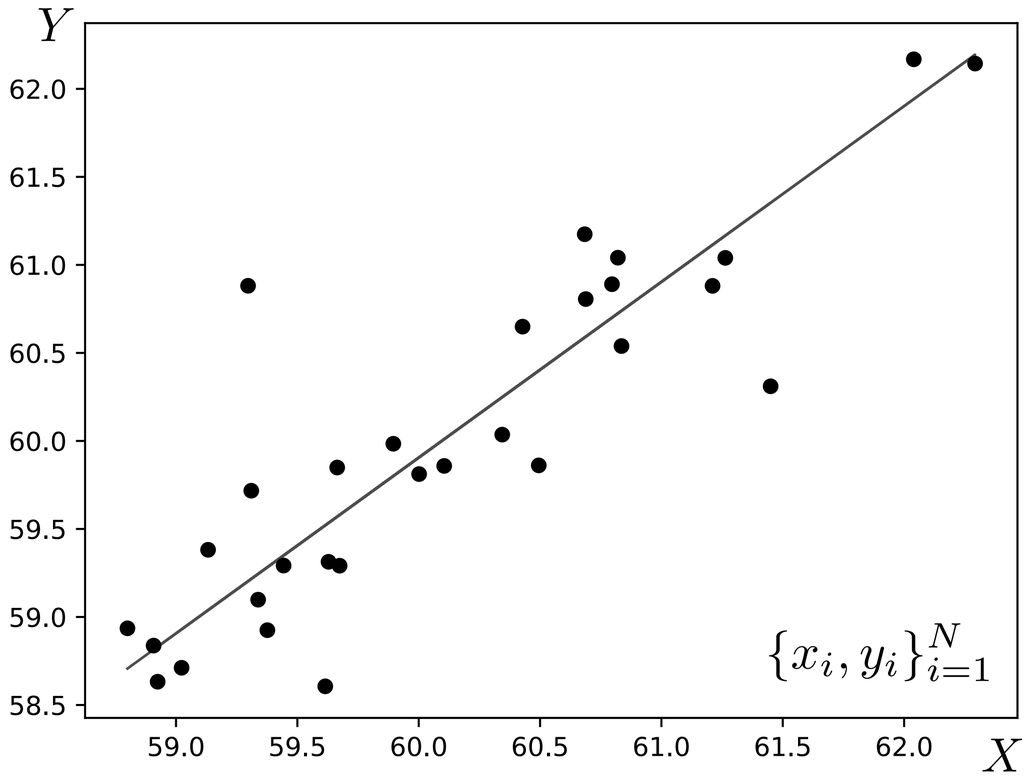
Best fit in what sense?

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

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

Source: Isola, Torralba, Freeman

Training data



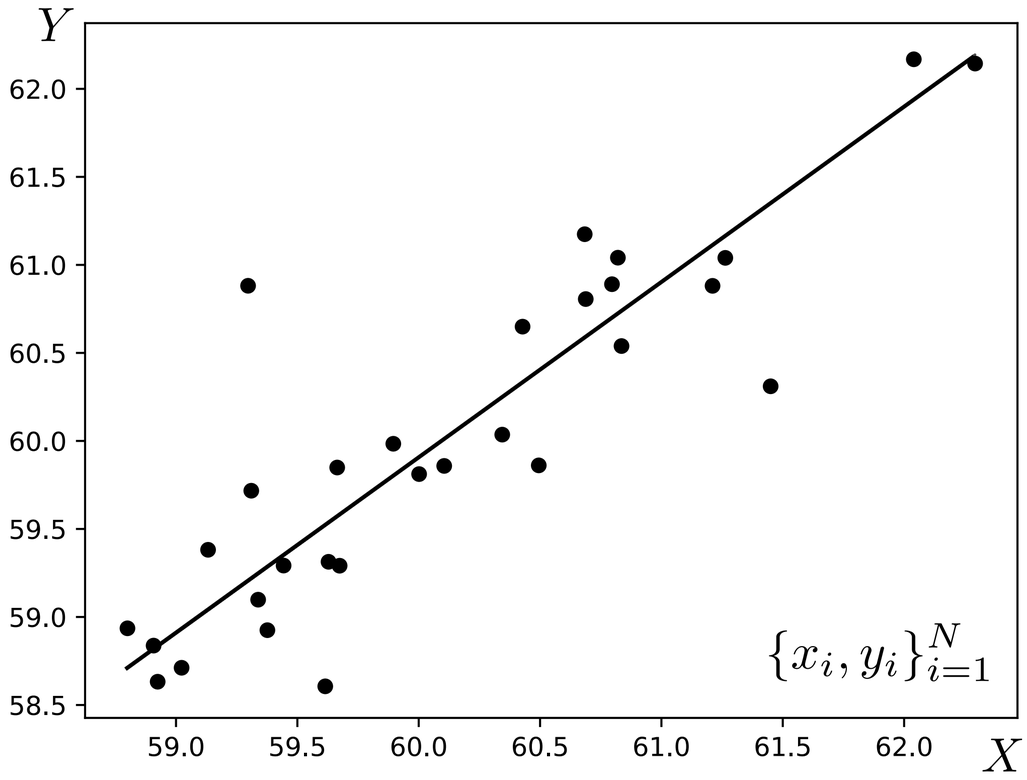
Complete learning problem:

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

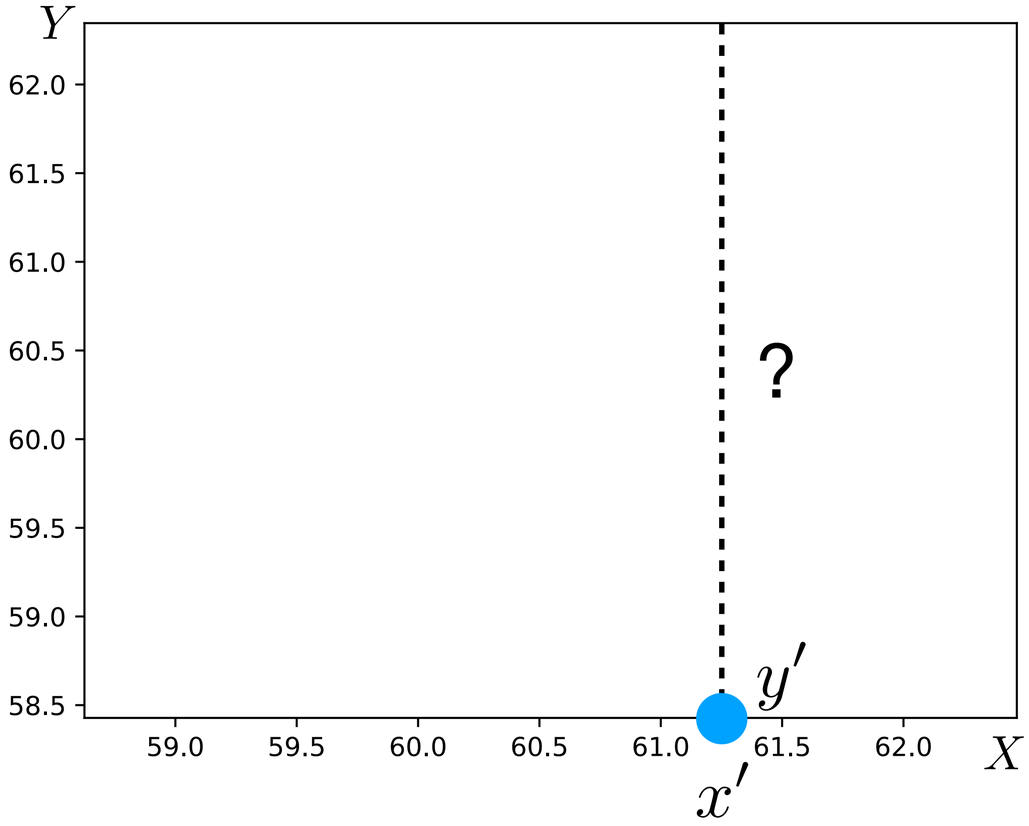
$$= \arg\min_\theta \sum_{i=1}^{N} (\theta_1 x_i + \theta_0 - y_i)^2$$

Source: Isola, Torralba, Freeman

Training data

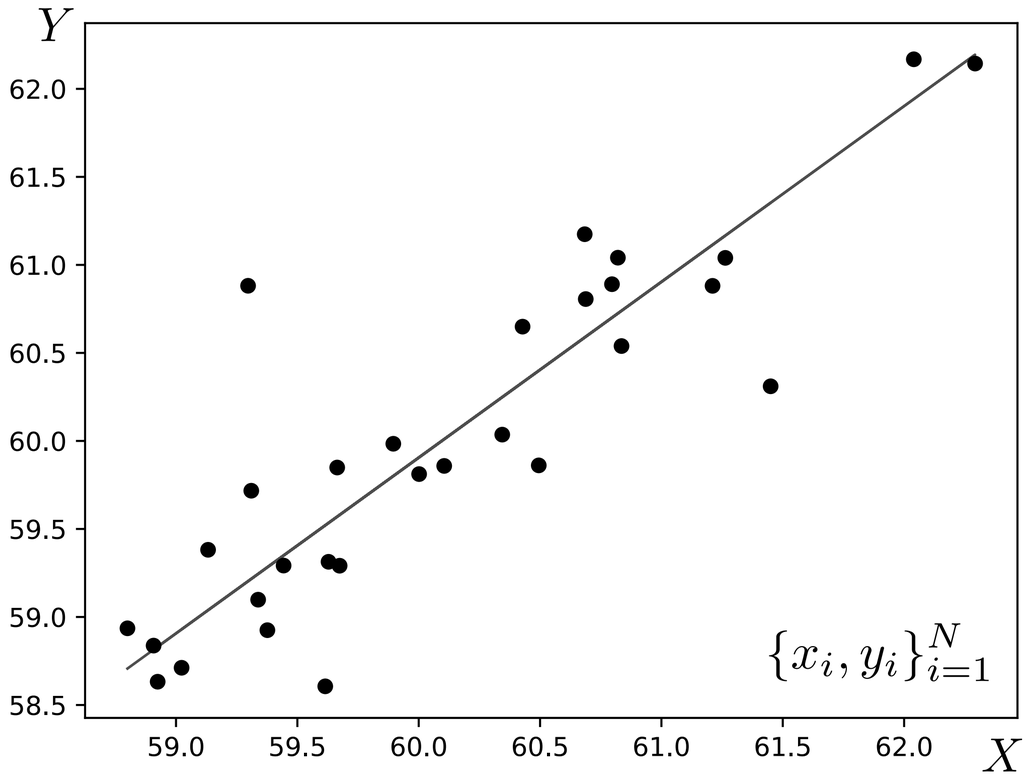


Test query

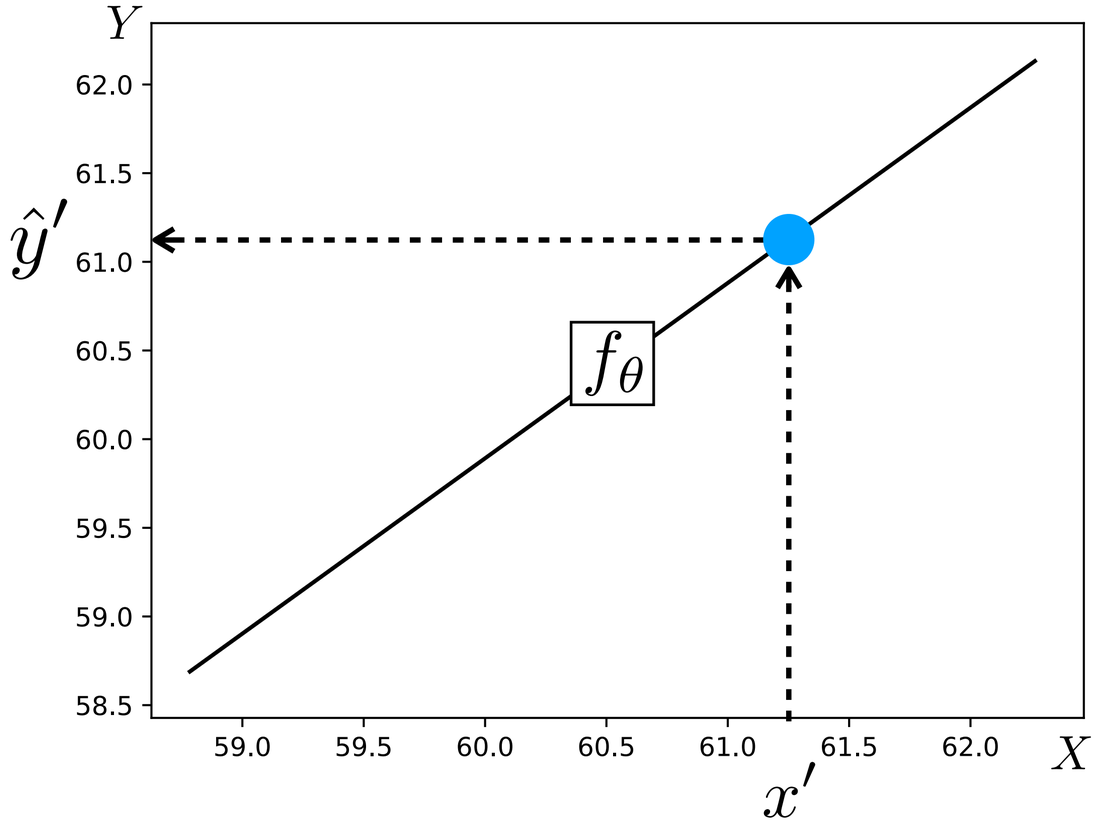


Source: Isola, Torralba, Freeman

Training data



Test query



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

Source: Isola, Torralba, Freeman

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

Learning problem

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

Recall:

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

$$\mathbf{X} = \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_N & 1 \end{pmatrix} \quad \theta = \begin{pmatrix} \theta_1 & \theta_0 \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$

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

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

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

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

Solution

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

Source: Isola, Torralba, Freeman

Empirical Risk Minimization

(formalization of supervised learning)

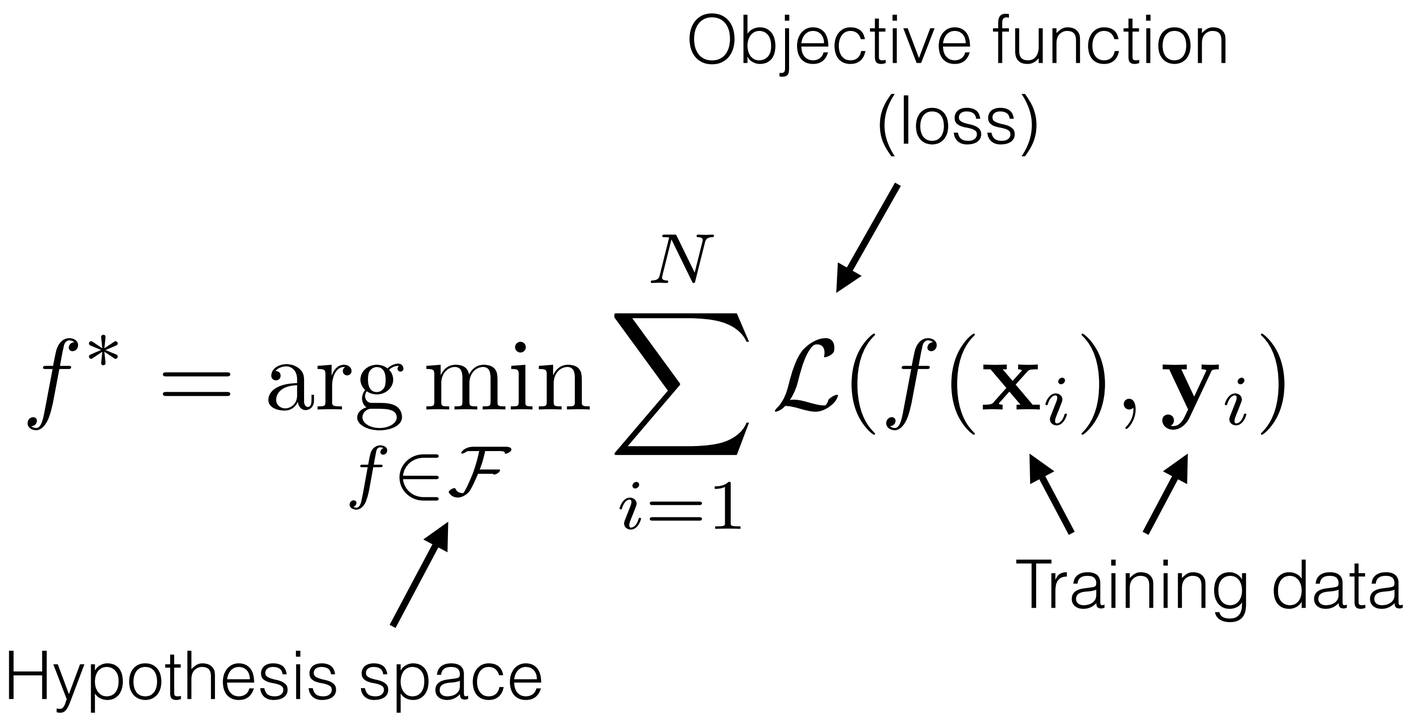
Linear least squares learning problem

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

Source: Isola, Torralba, Freeman

Empirical Risk Minimization

(formalization of supervised learning)



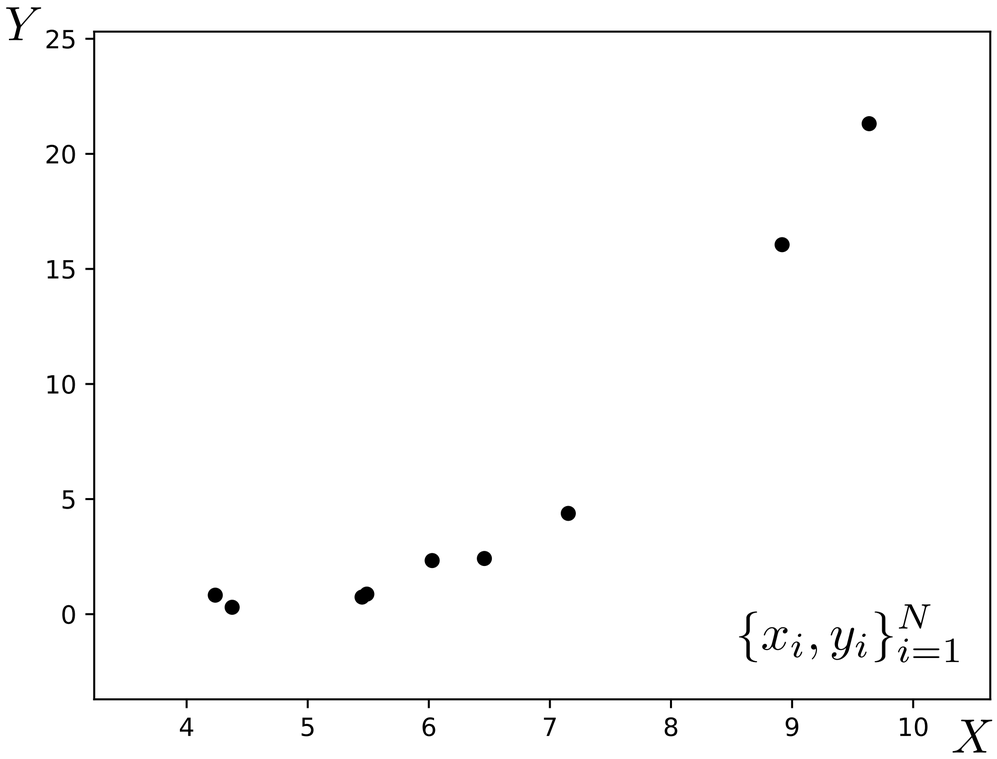
Source: Isola, Torralba, Freeman

The Problem of Generalization

Source: Isola, Torralba, Freeman

Linear regression

Training data

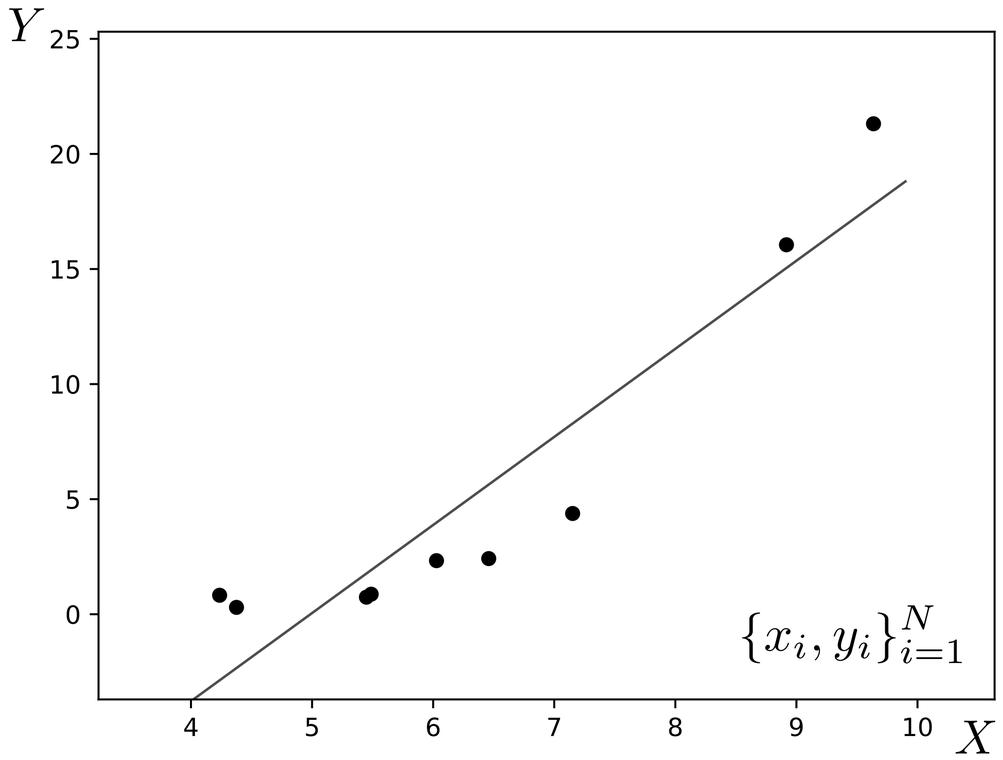


$$f_\theta(x) = \theta_0 + \theta_1 x$$

Source: Isola, Torralba, Freeman

Linear regression

Training data

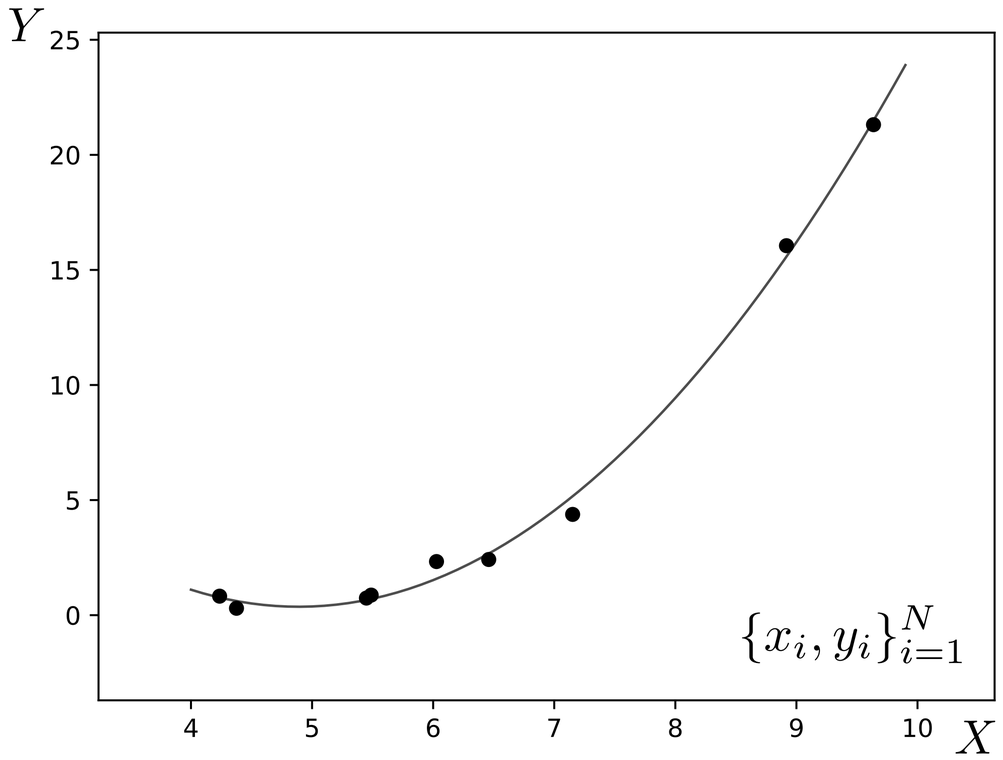


$$f_\theta(x) = \theta_0 + \theta_1 x$$

Source: Isola, Torralba, Freeman

Polynomial regression

Training data



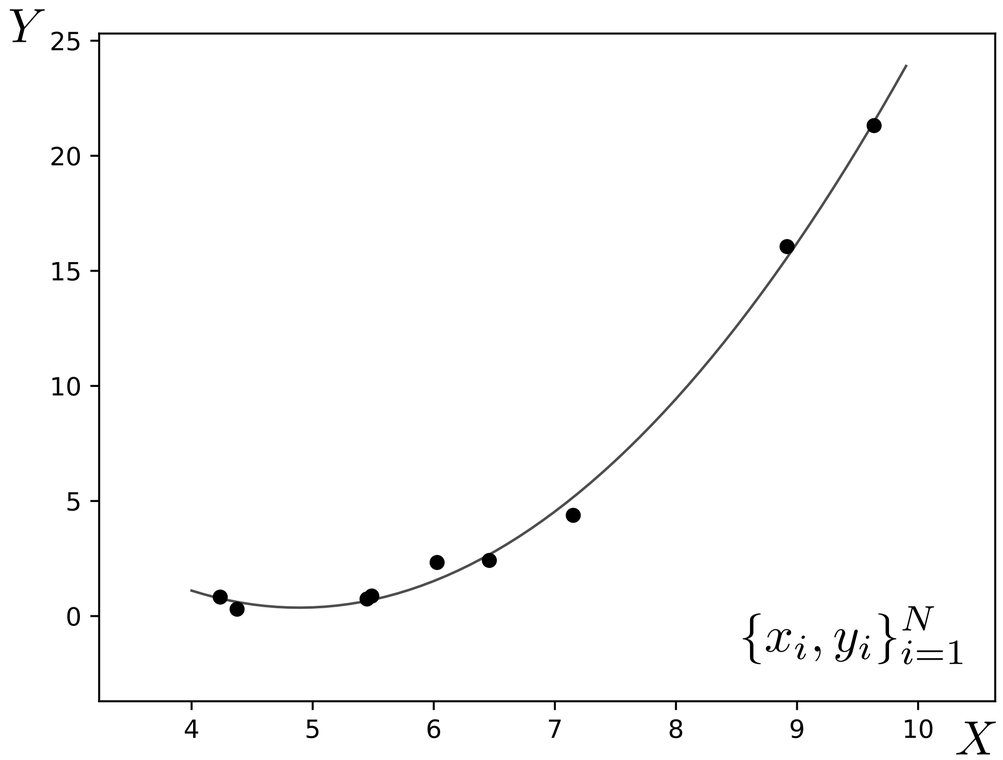
$$f_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

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

K-th degree polynomial regression

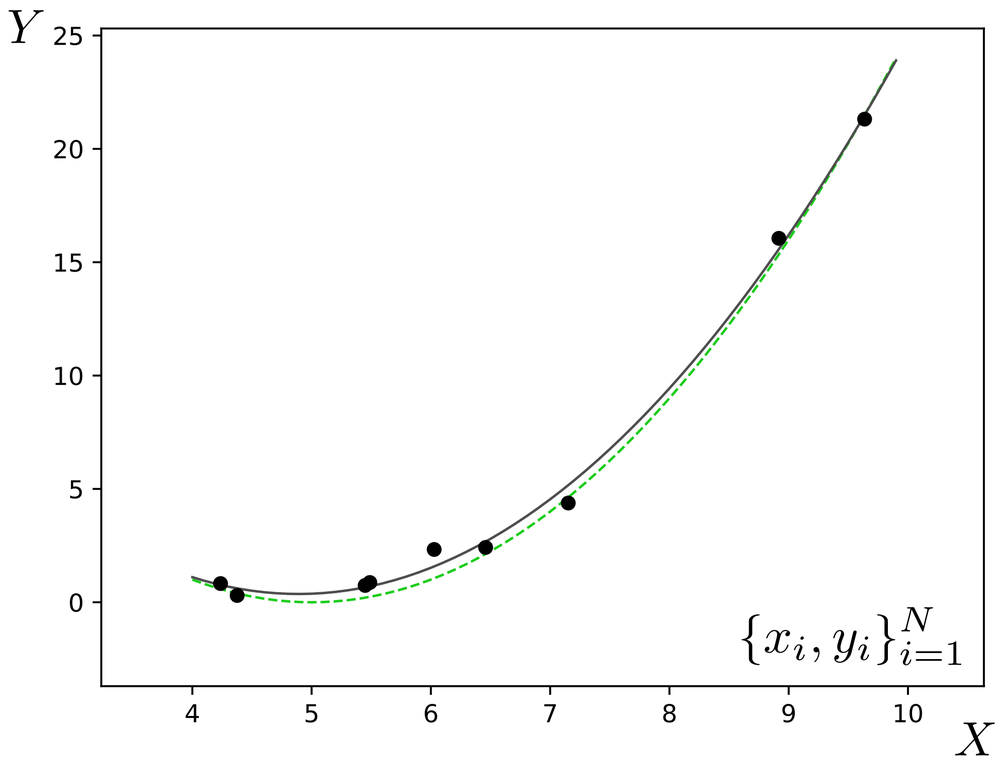
Source: Isola, Torralba, Freeman

Training data

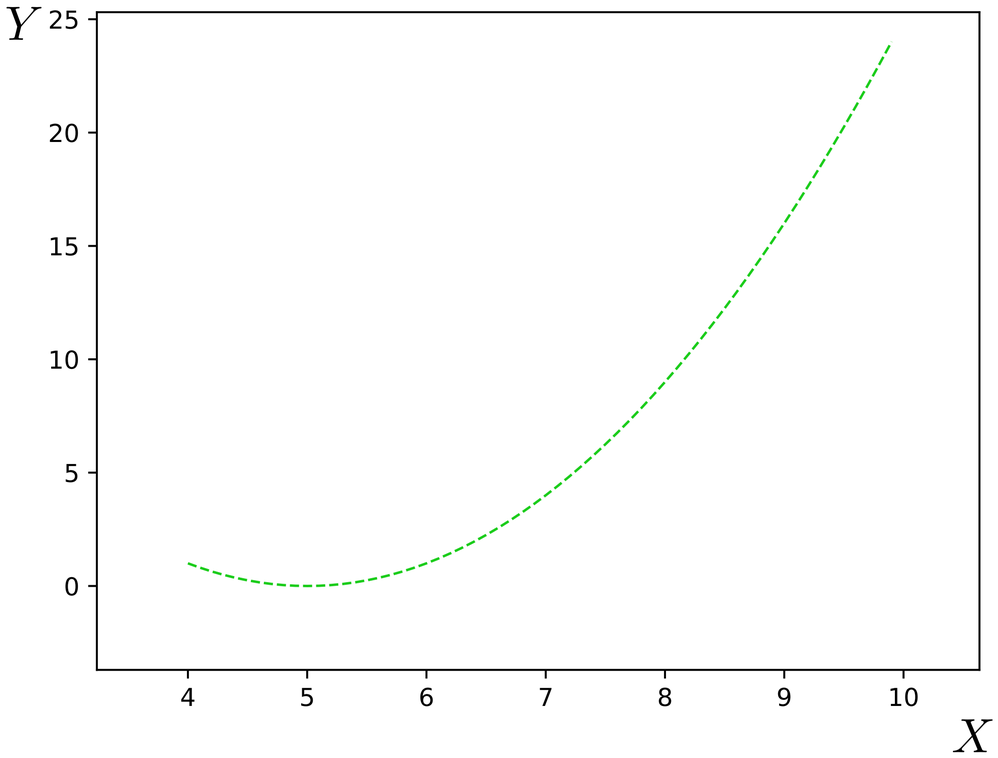


Source: Isola, Torralba, Freeman

Training data



Test data

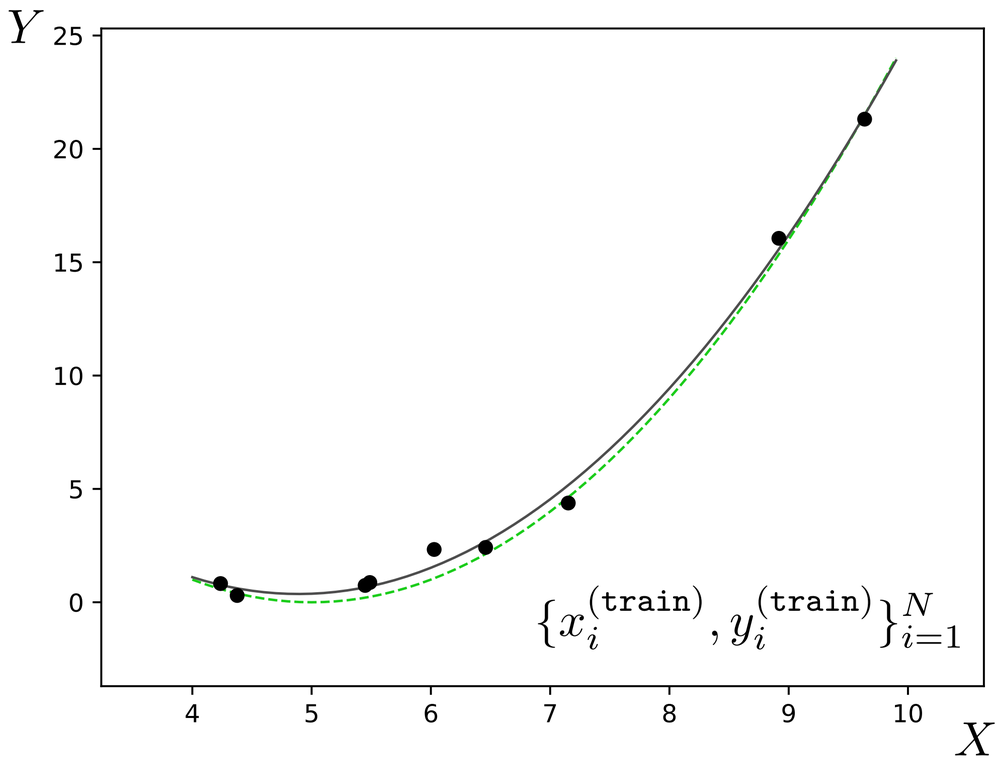


True **data-generating process**

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

Source: Isola, Torralba, Freeman

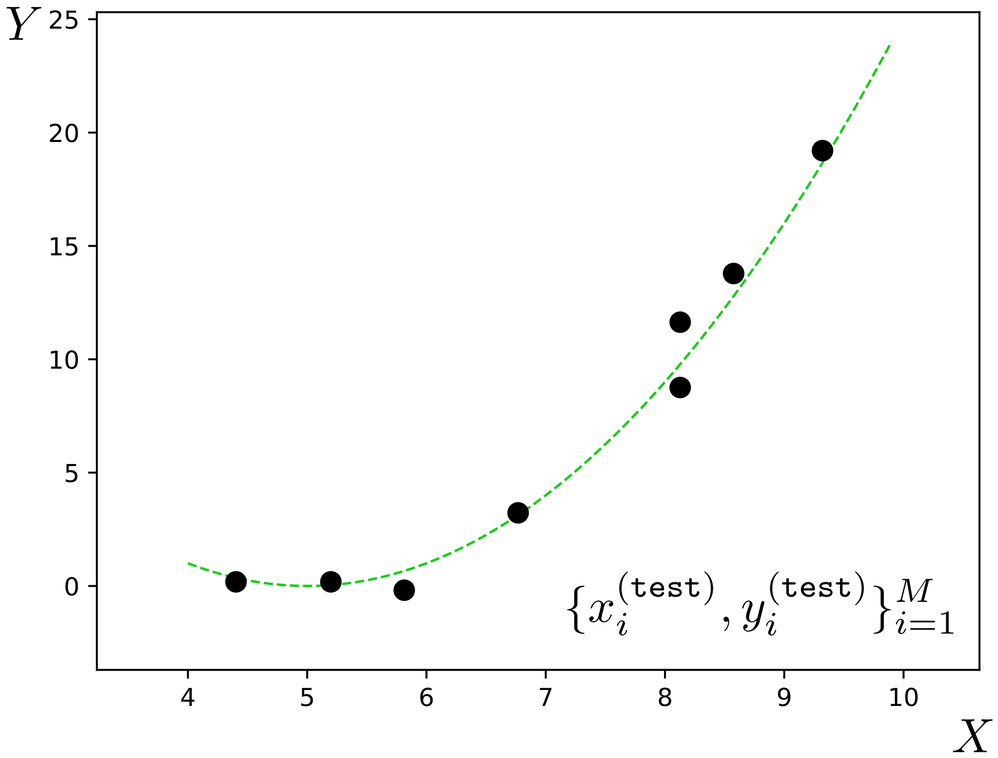
Training data



True **data-generating process**

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

Test data

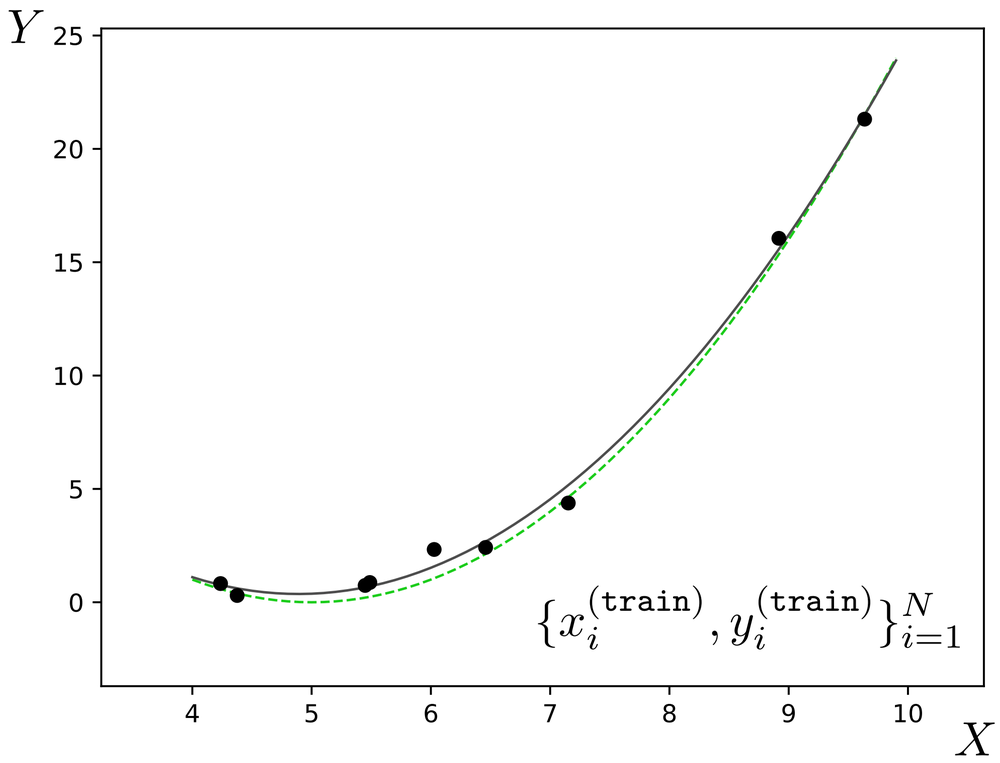


$$\{x_i^{(\texttt{train})}, y_i^{(\texttt{train})}\} \overset{\texttt{iid}}{\sim} p_{\texttt{data}}$$

$$\{x_i^{(\texttt{test})}, y_i^{(\texttt{test})}\} \overset{\texttt{iid}}{\sim} p_{\texttt{data}}$$

Source: Isola, Torralba, Freeman

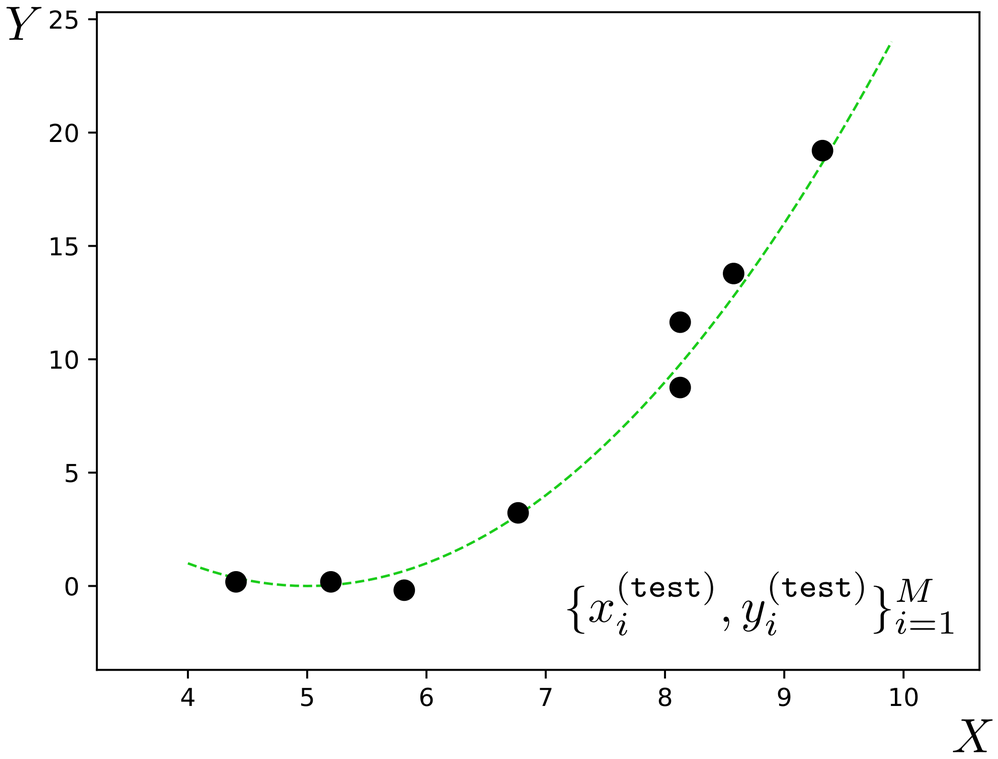
Training data



Training objective:

$$\sum_{i=1}^{N} (f_\theta(x_i^{\texttt{train}}) - y_i^{\texttt{train}})^2$$

Test data

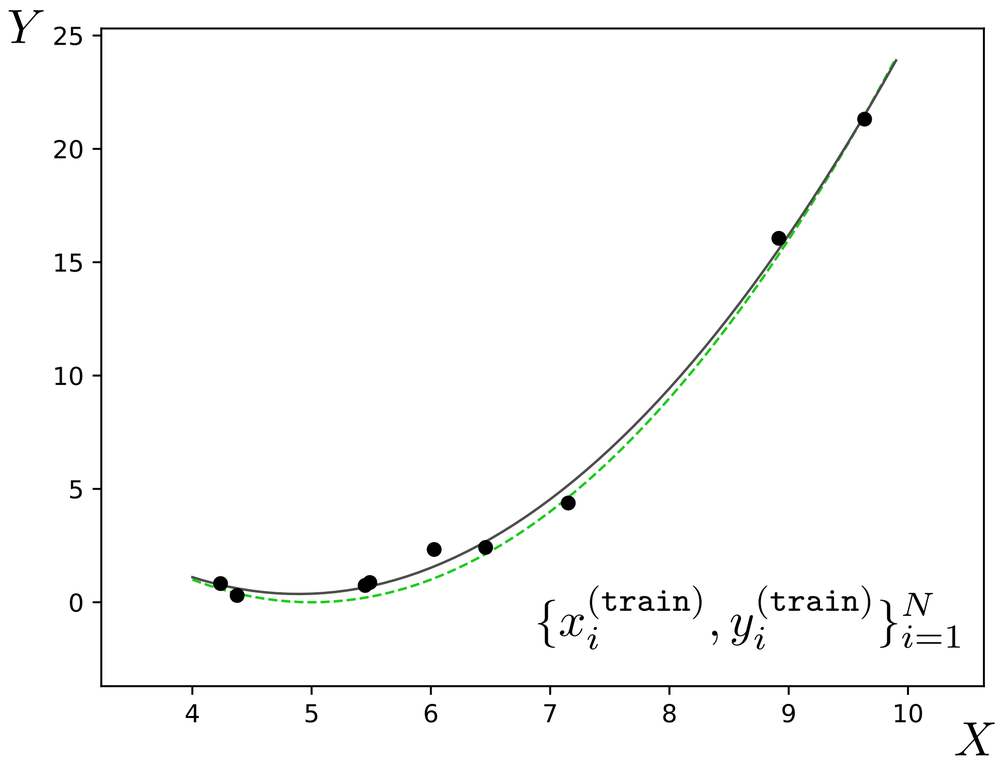


Test time evaluation:

$$\sum_{i=1}^{M} (f_\theta(x_i^{\texttt{test}}) - y_i^{\texttt{test}})^2$$

Source: Isola, Torralba, Freeman

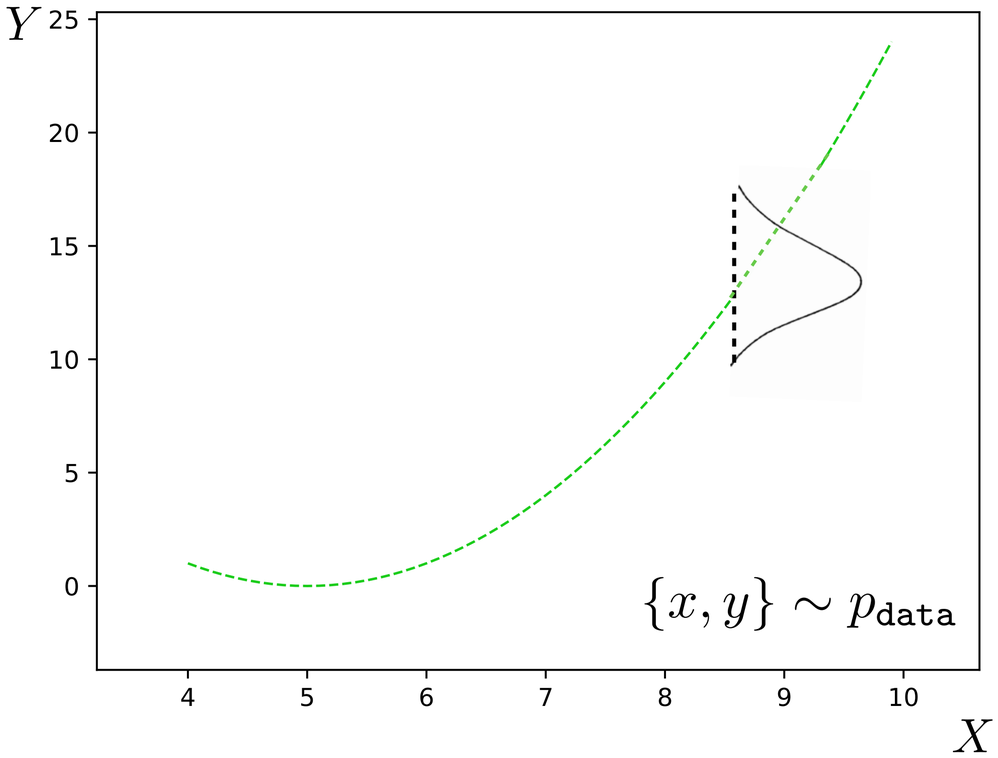
Training data



Training objective:

$$\sum_{i=1}^{N} (f_\theta(x_i^{\texttt{train}}) - y_i^{\texttt{train}})^2$$

Test data



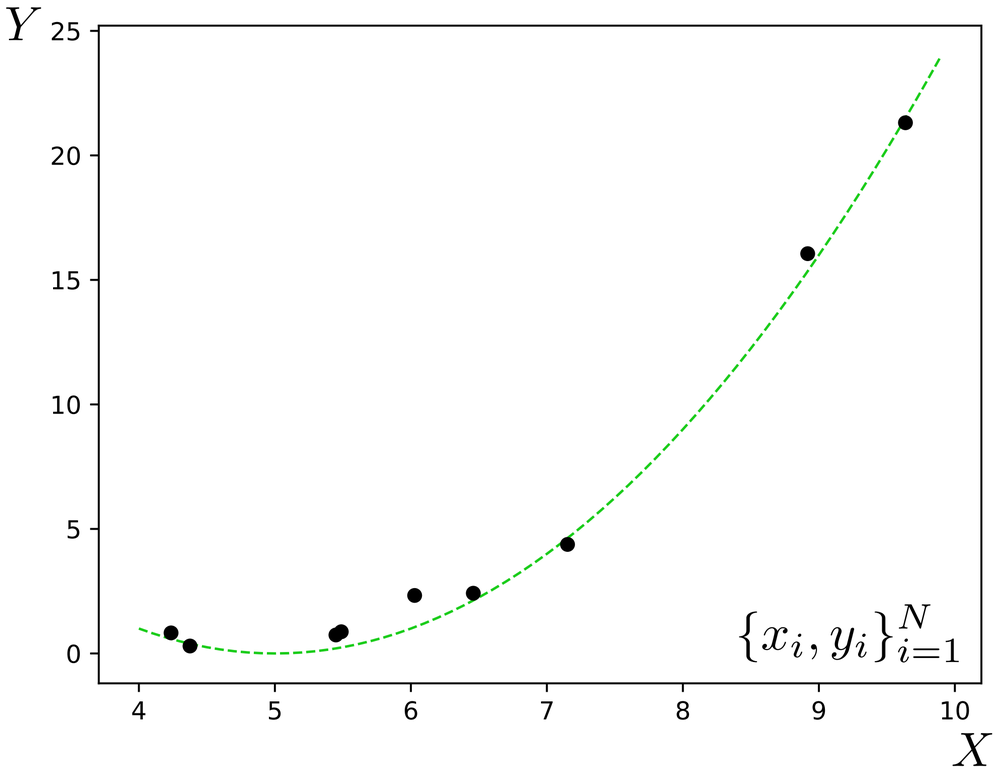
True objective:

$$\mathbb{E}_{\{x,y\}\sim p_{\texttt{data}}}[(f_\theta(x) - y)^2]$$

Source: Isola, Torralba, Freeman

What happens as we add more basis functions?

Training data

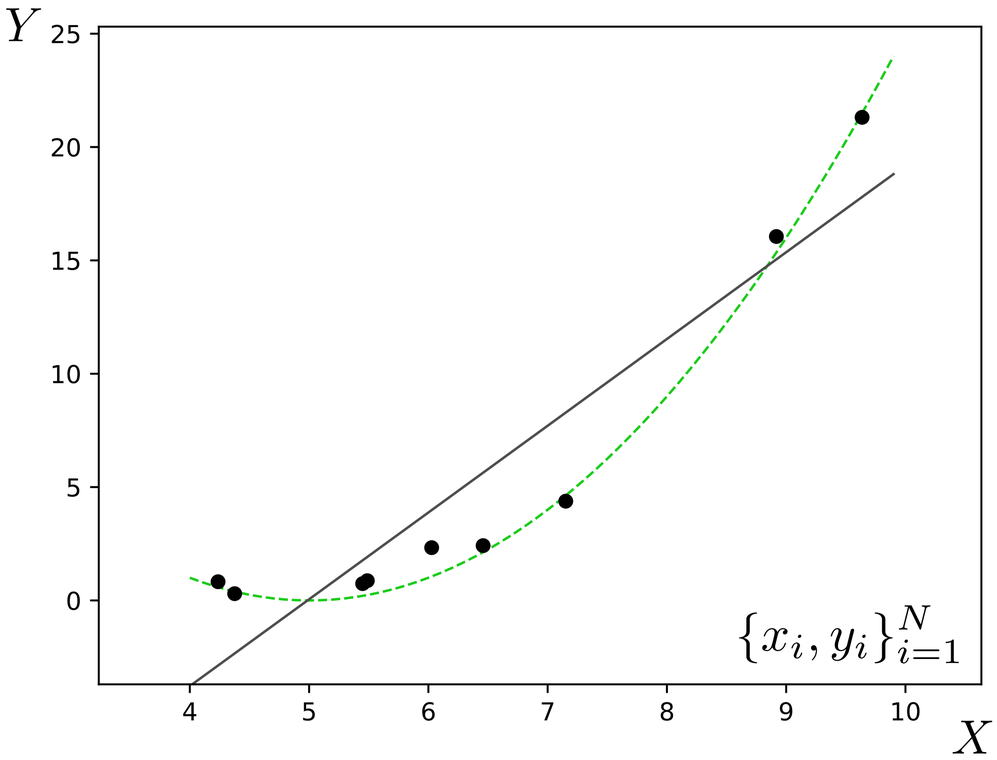


$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

Source: Isola, Torralba, Freeman

What happens as we add more basis functions?

K = 1

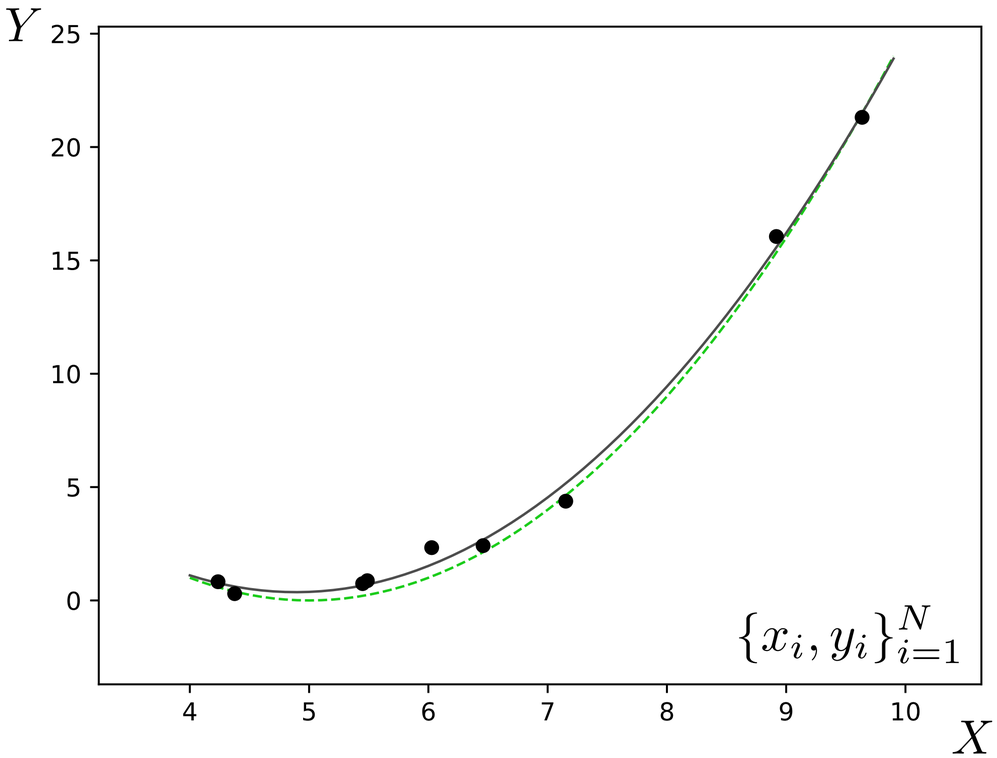


$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

Source: Isola, Torralba, Freeman

What happens as we add more basis functions?

K = 2

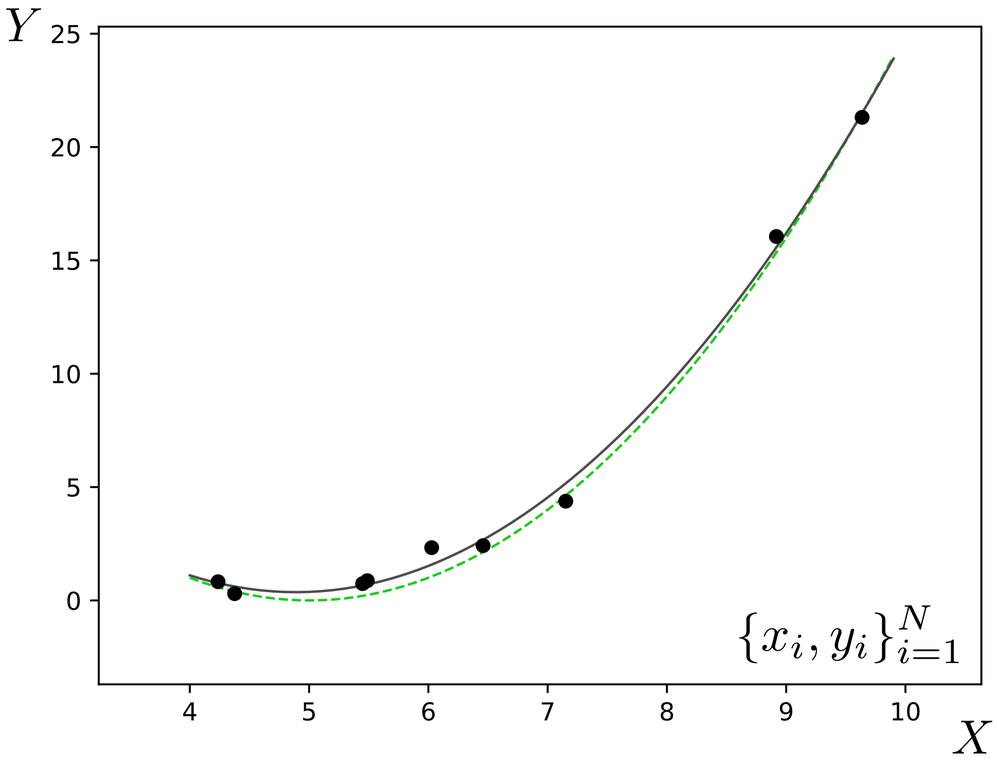


$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

Source: Isola, Torralba, Freeman

What happens as we add more basis functions?

K = 3

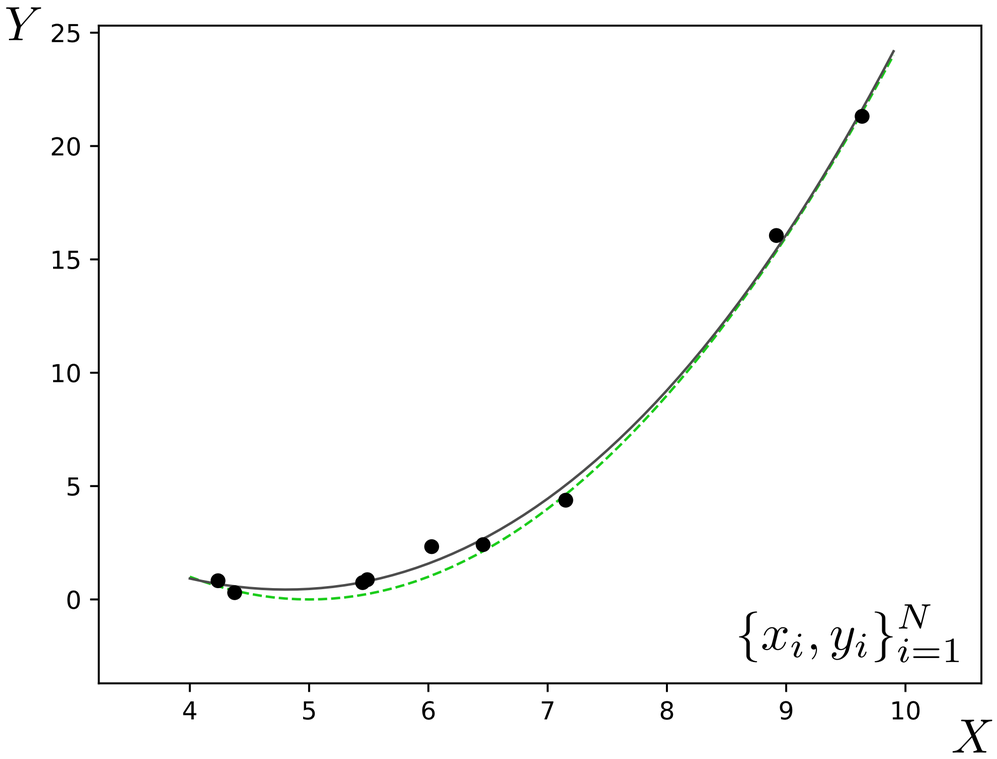


$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

Source: Isola, Torralba, Freeman

What happens as we add more basis functions?

K = 4

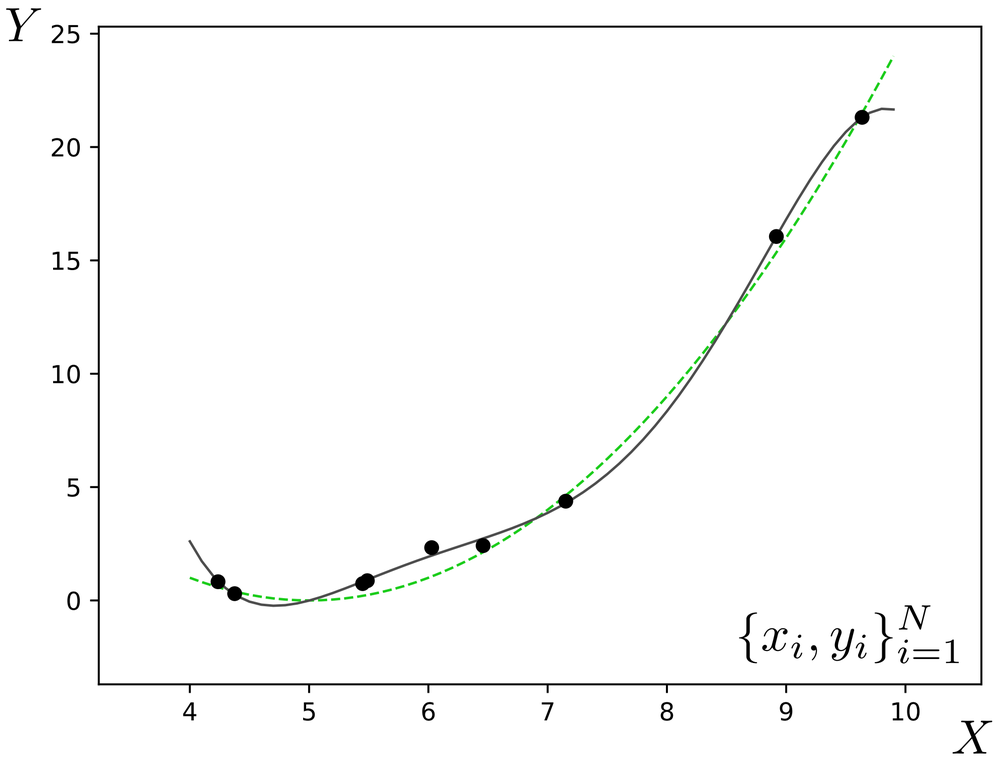


$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

Source: Isola, Torralba, Freeman

What happens as we add more basis functions?

K = 5

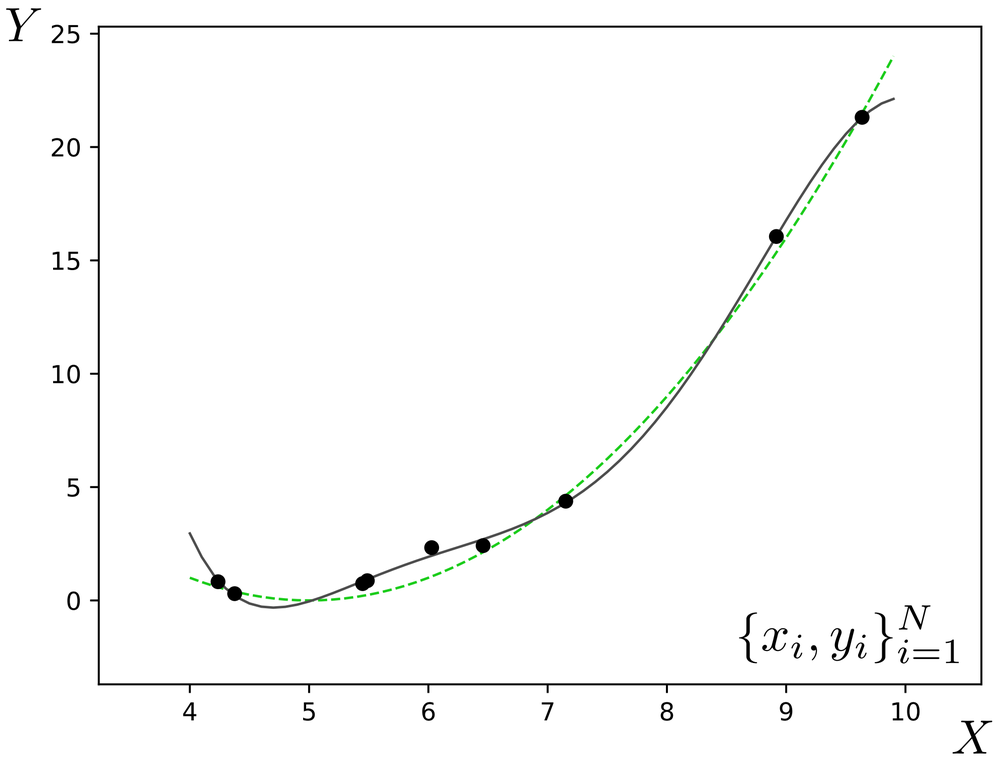


$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

Source: Isola, Torralba, Freeman

What happens as we add more basis functions?

K = 6

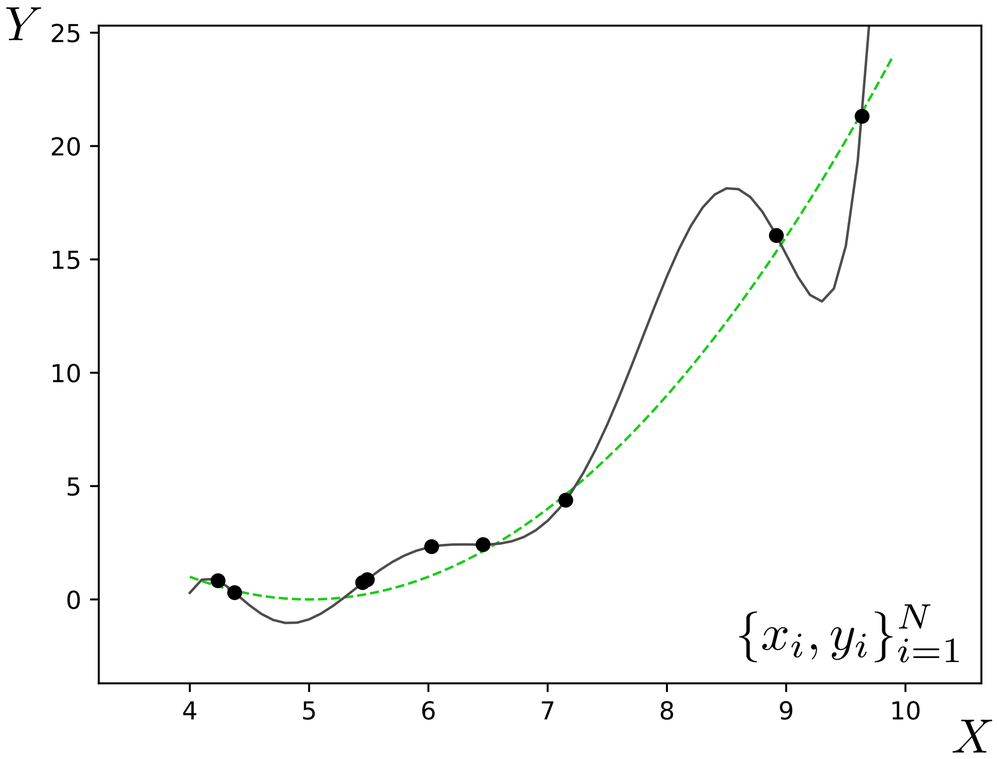


$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

Source: Isola, Torralba, Freeman

What happens as we add more basis functions?

K = 7

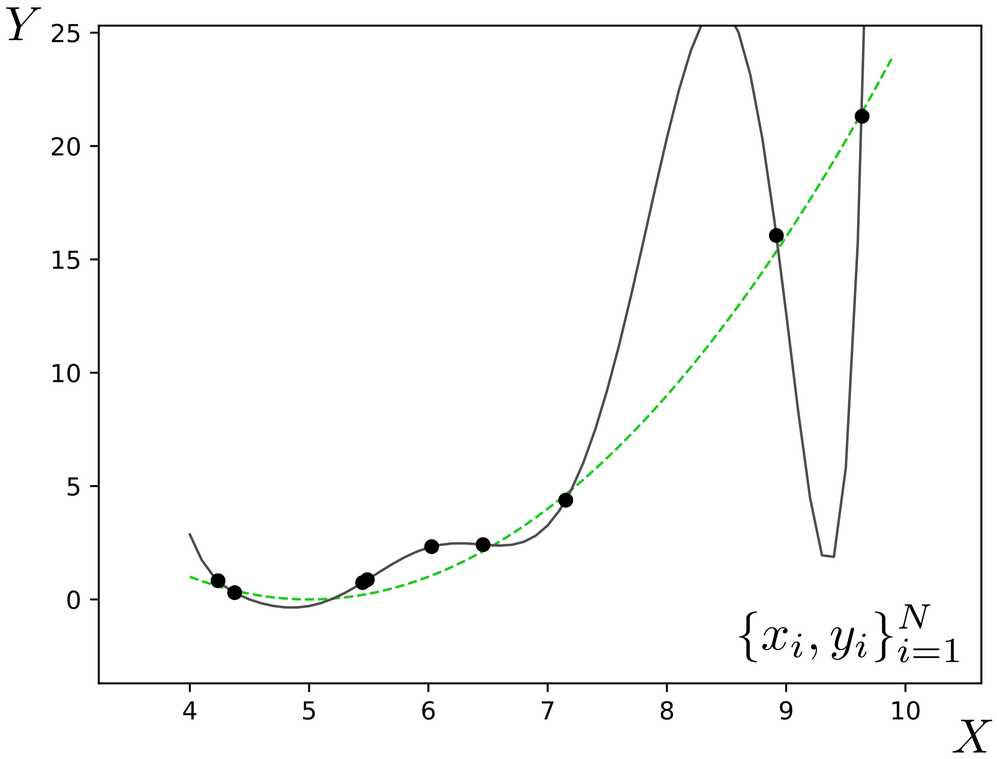


$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

Source: Isola, Torralba, Freeman

What happens as we add more basis functions?

K = 8

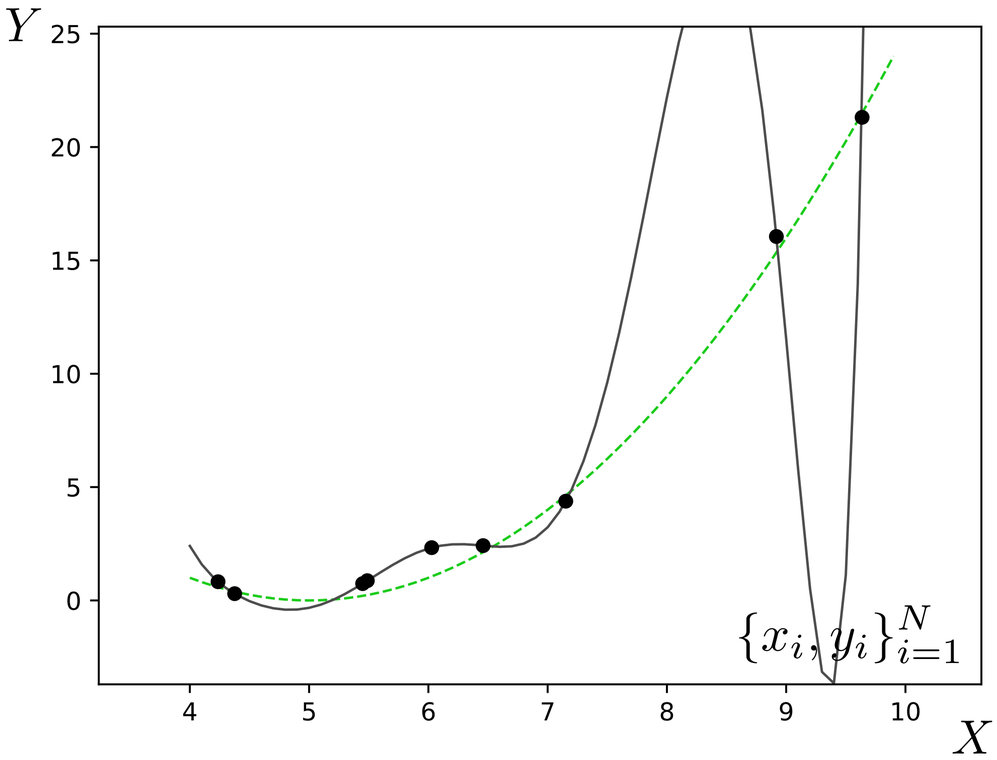


$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

Source: Isola, Torralba, Freeman

What happens as we add more basis functions?

K = 9

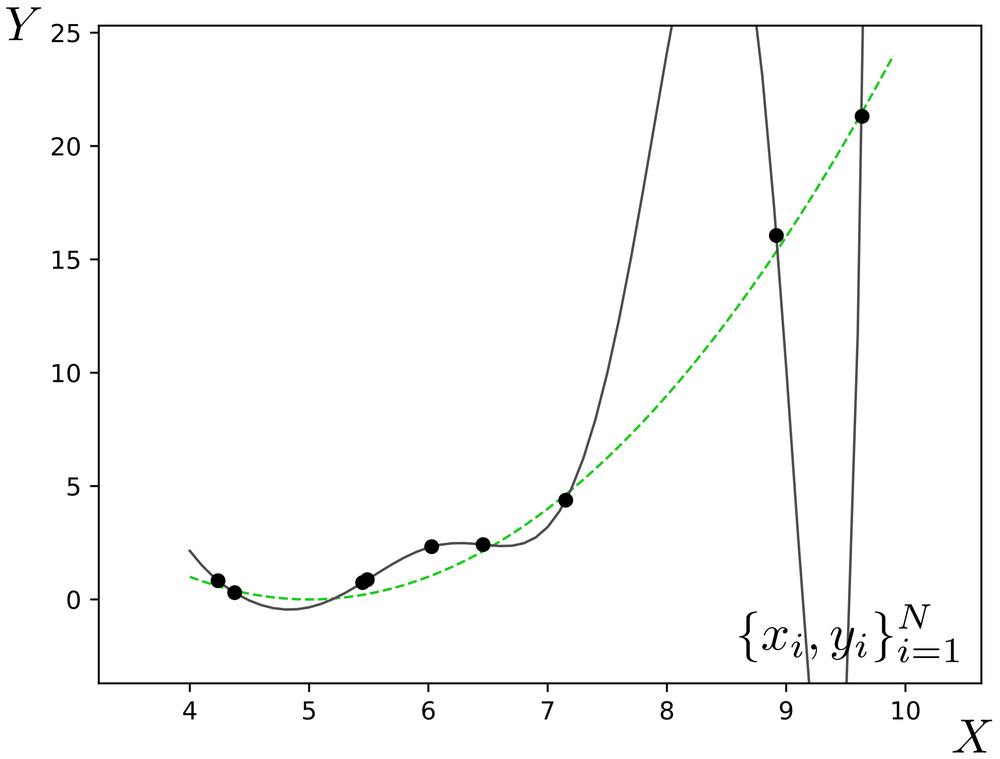


$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

Source: Isola, Torralba, Freeman

What happens as we add more basis functions?

K = 10

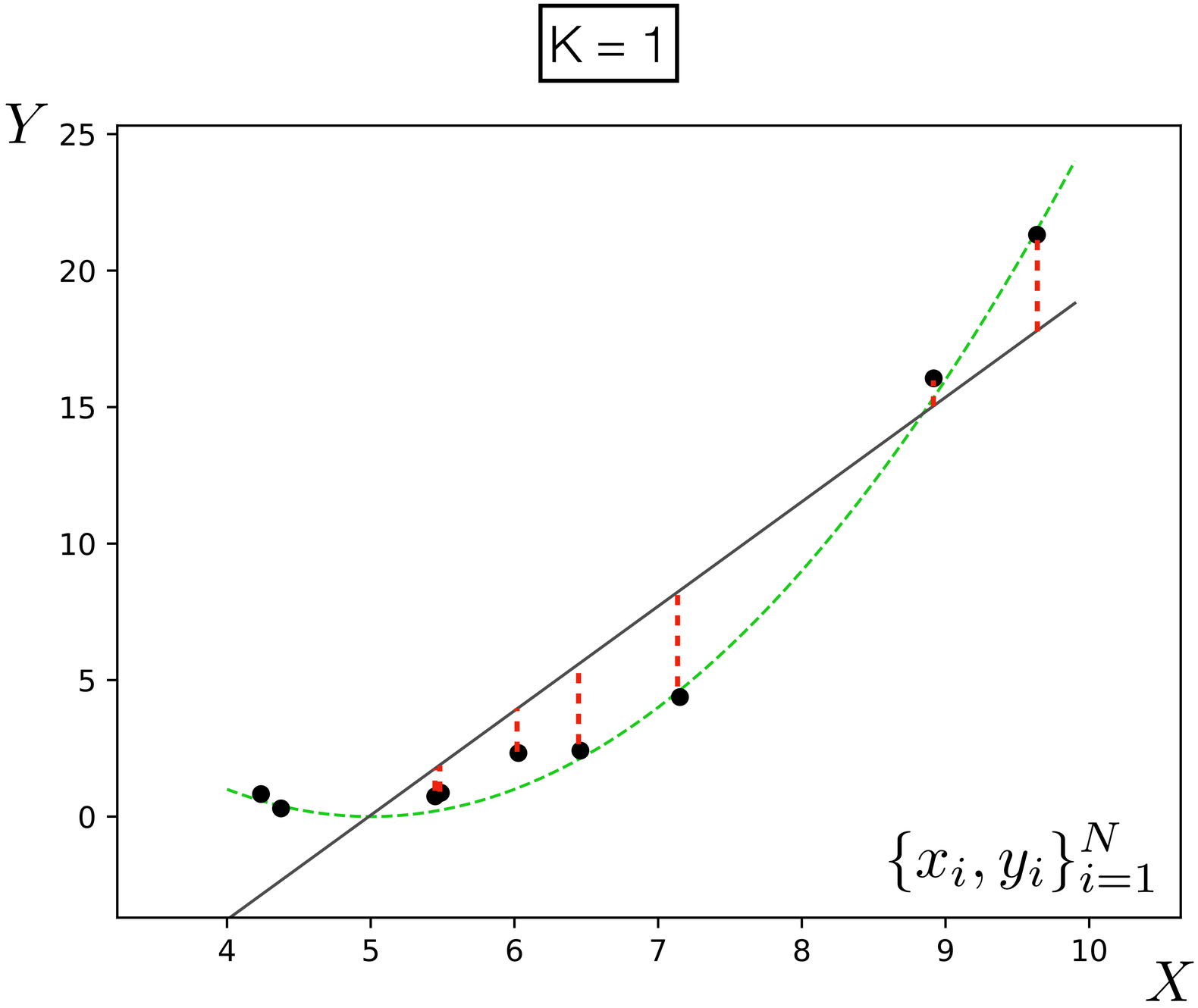


$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

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.

Source: Isola, Torralba, Freeman

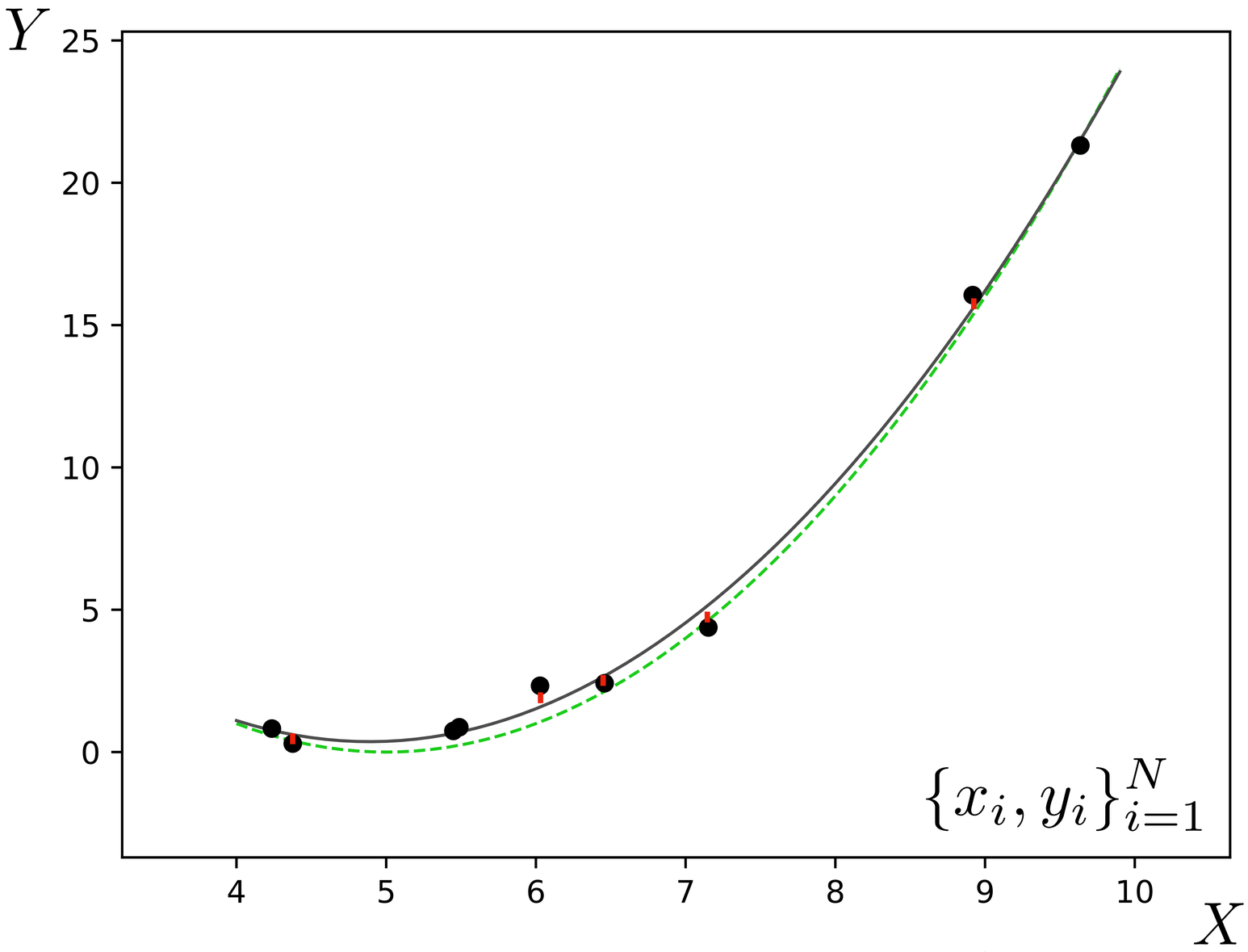


When the model does not have the capacity to capture the true function, we call this **underfitting**.

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

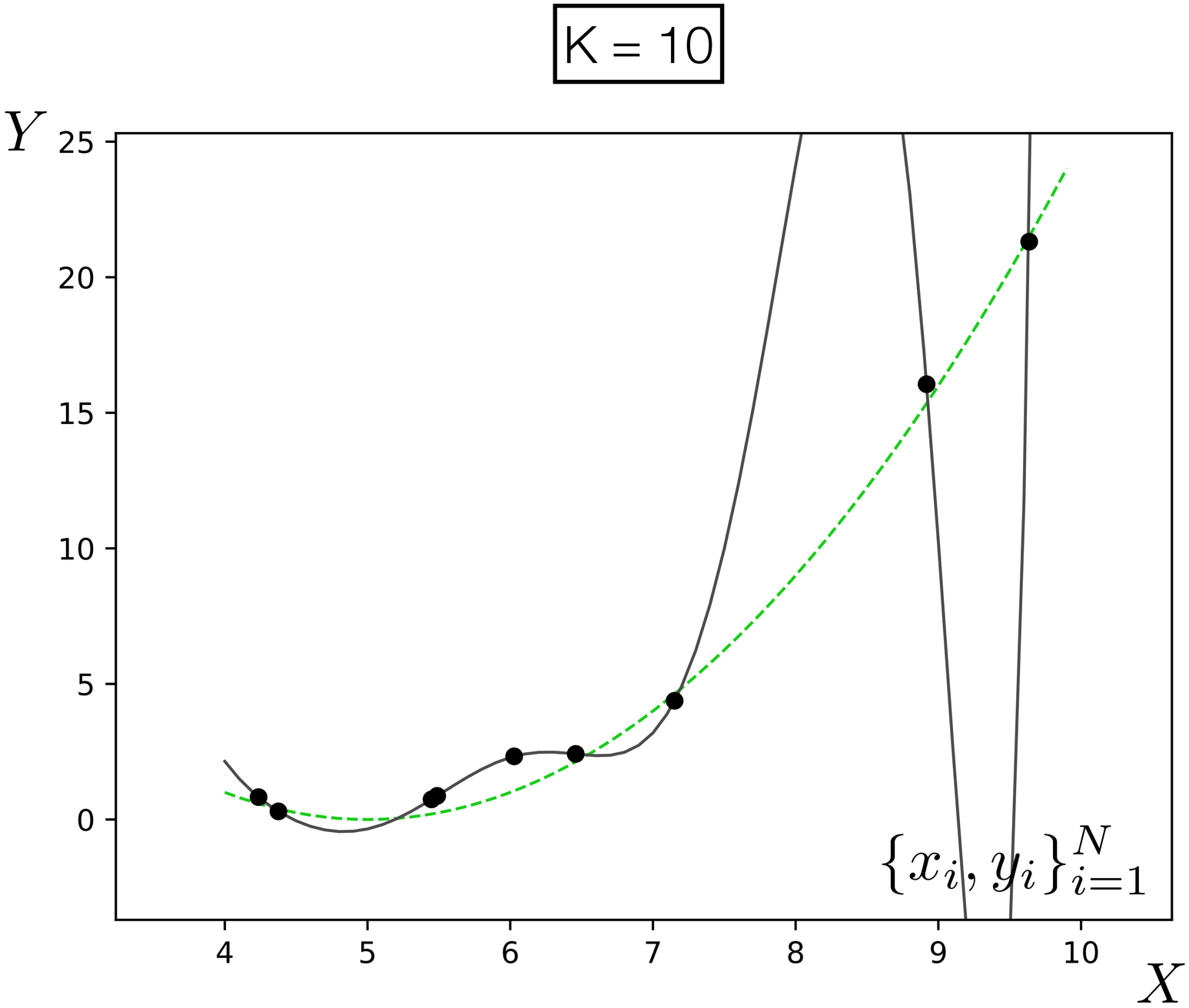
Source: Isola, Torralba, Freeman

K = 2



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

Source: Isola, Torralba, Freeman



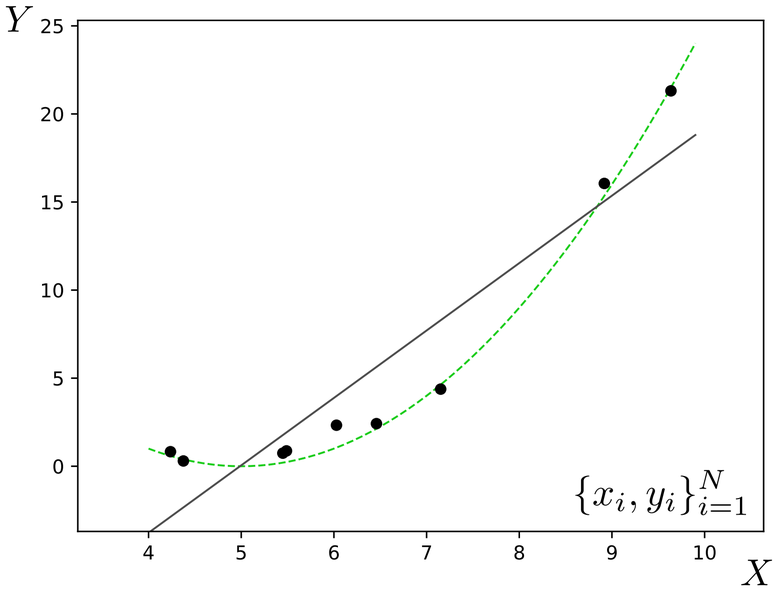
Now we have zero approximation error — the curve passes exactly through each training point.

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).

Source: Isola, Torralba, Freeman

Underfitting

K = 1

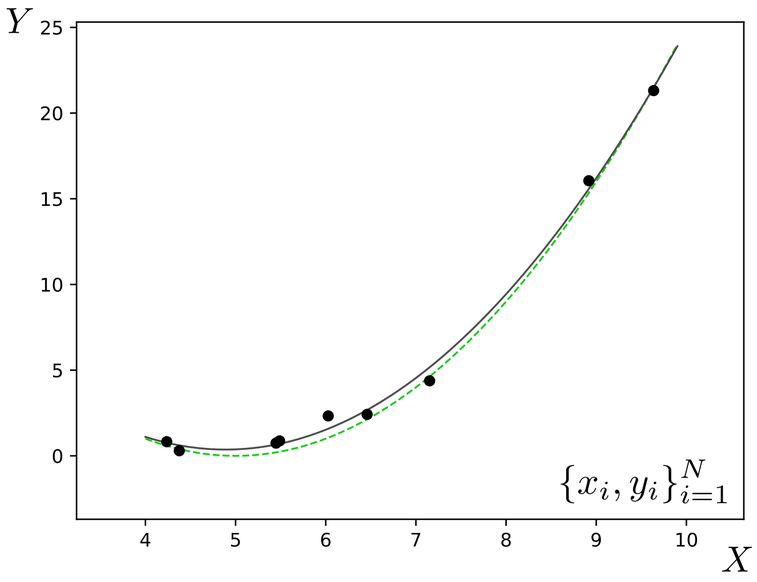


High error on train set

High error on test set

Appropriate model

K = 2

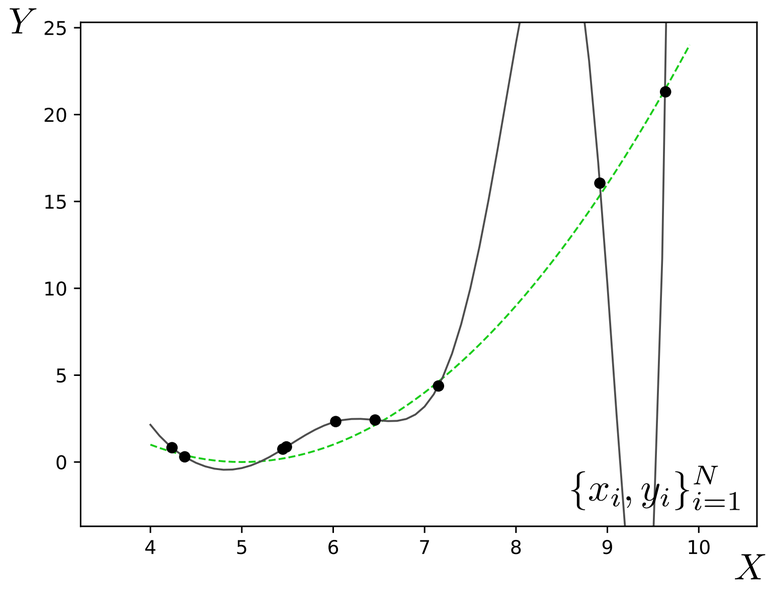


Low error on train set

Low error on test set

Overfitting

K = 10



Lowest error on train set

High error on test set

Source: Isola, Torralba, Freeman

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

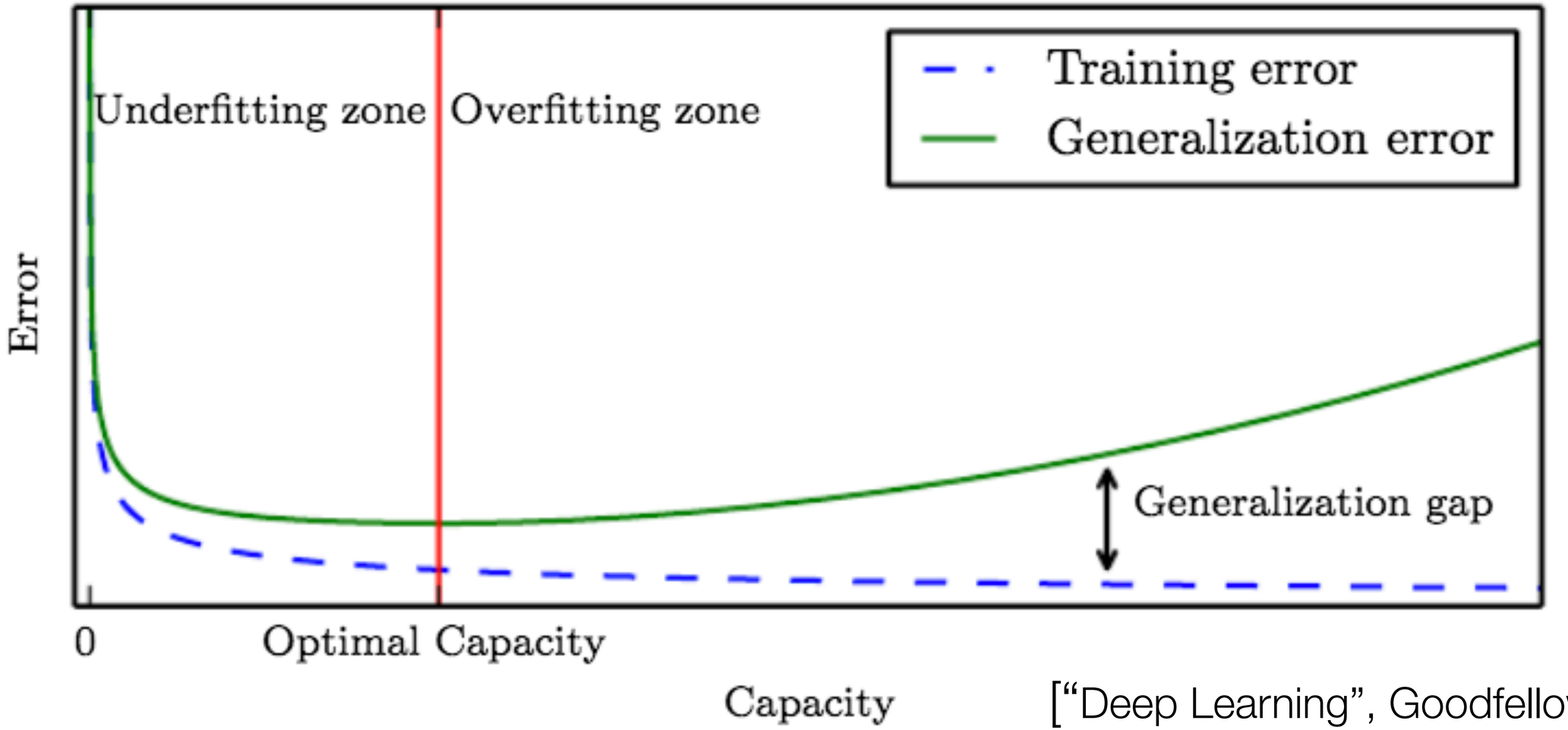
The capacity may be defined as the number of hypotheses under consideration in the hypothesis space.

Complex models with many free parameters have high capacity.

Simple models have low capacity.

Source: Isola, Torralba, Freeman

Training error versus generalization error

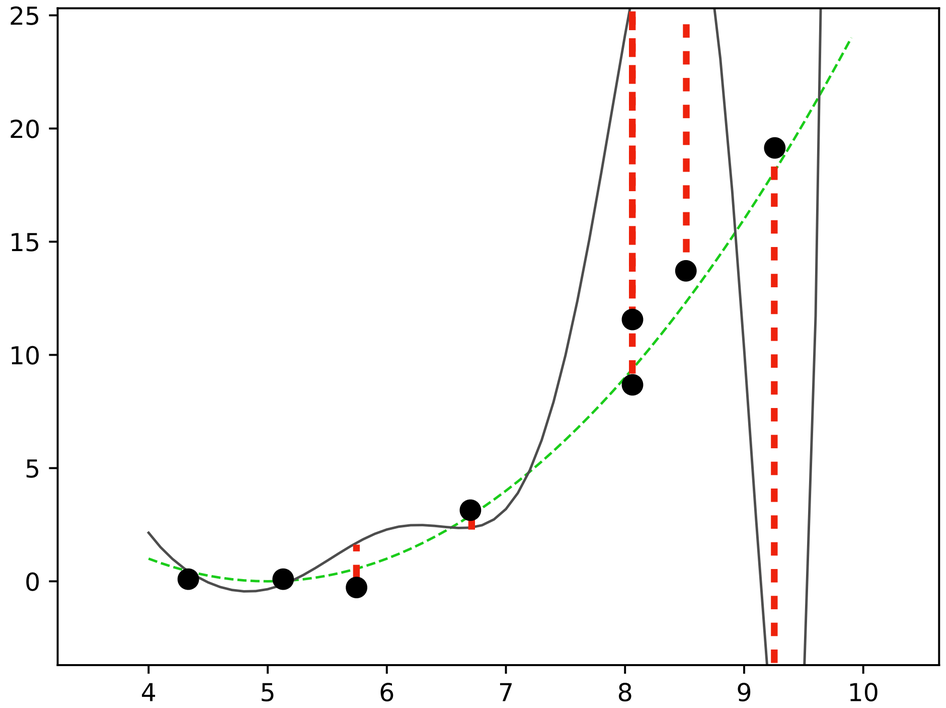


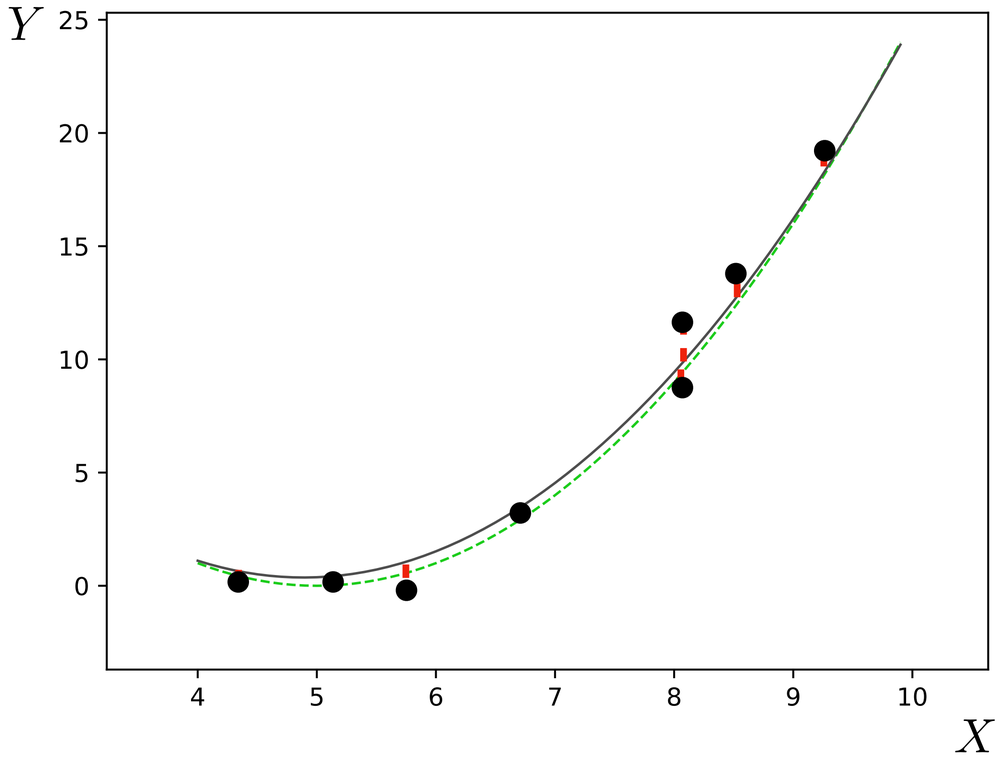
["Deep Learning", Goodfellow et al.]

Source: Isola, Torralba, Freeman

How do we know if we are underfitting or overfitting?

Validation data $\{x_i^{(\texttt{val})}, y_i^{(\texttt{val})}\}$





Cross validation: measure prediction error on validation set

Source: Isola, Torralba, Freeman

Fitting a model

Underfitting?

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

Overfitting?

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

Source: Isola, Torralba, Freeman

Regularization

Empirical risk minimization:

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

Source: Isola, Torralba, Freeman

Regularized least squares

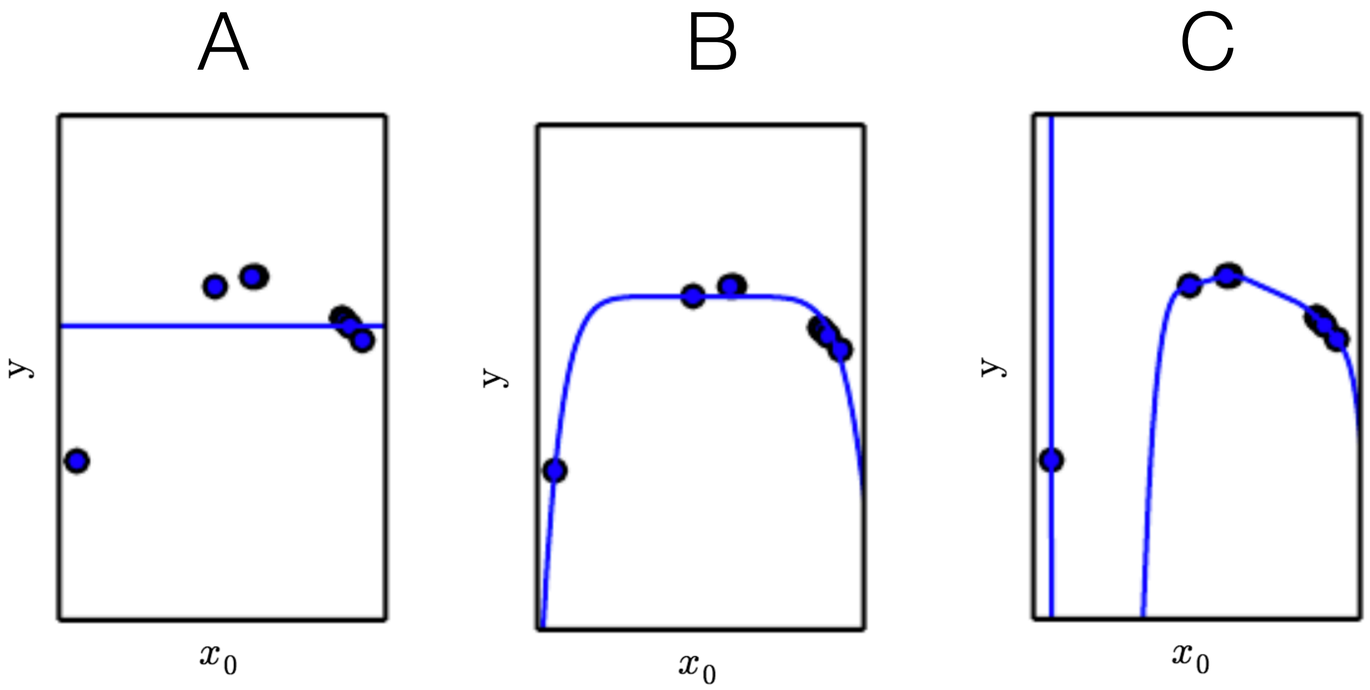
$$f_\theta(x) = \sum_{k=0}^{K} \theta_k x^k$$

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

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

Source: Isola, Torralba, Freeman

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



Low λ — ?

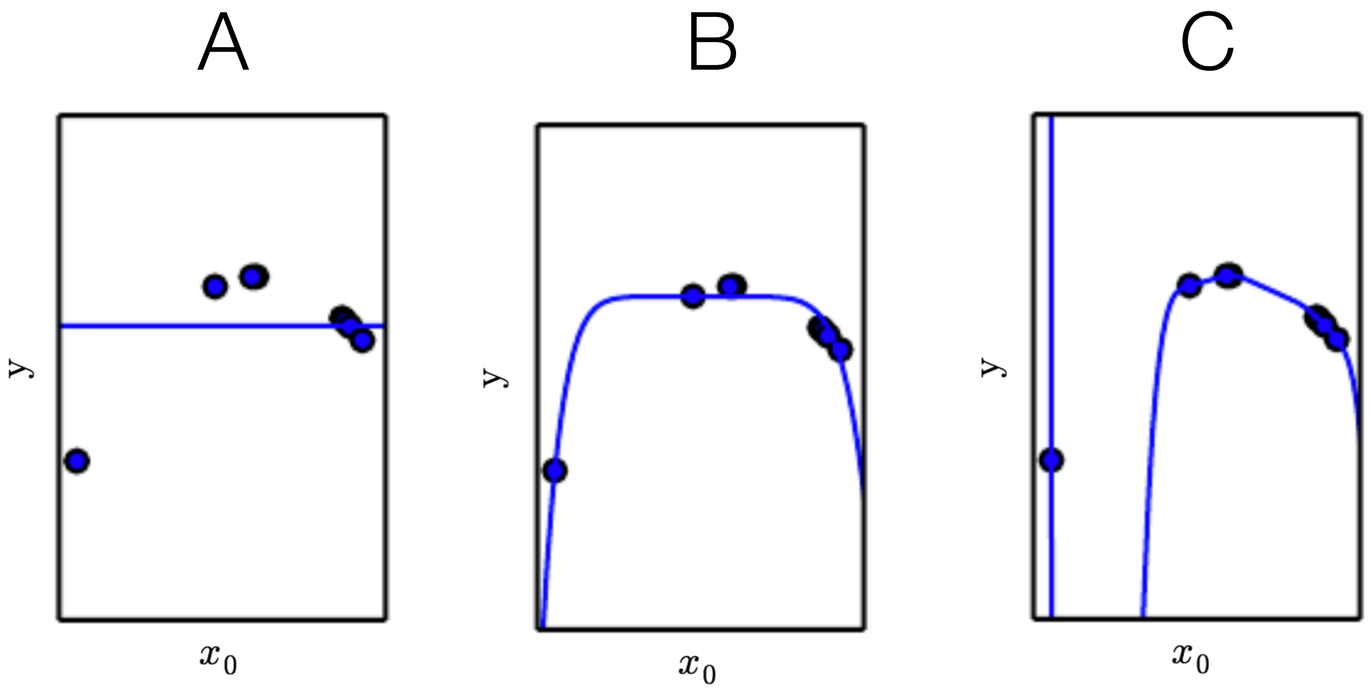
Medium λ — ?

High λ — ?

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

Source: Isola, Torralba, Freeman

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



Low λ — C

Medium λ — B

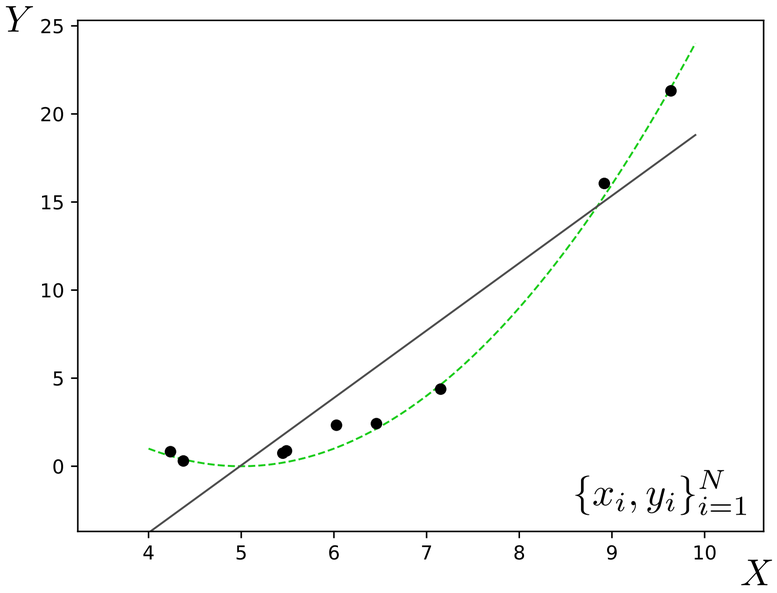
High λ — A

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

Source: Isola, Torralba, Freeman

Underfitting

K = 1

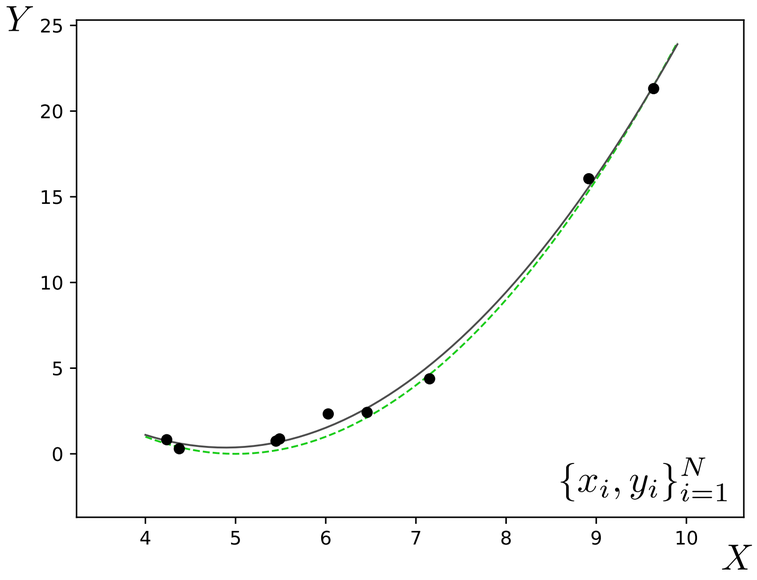


Simple model

Doesn’t fit the training data

Appropriate model

K = 2

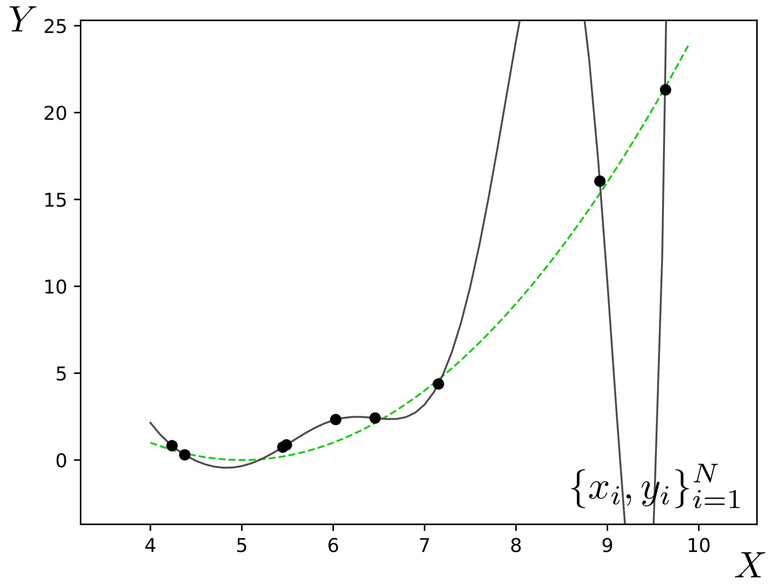


Simple model

Fits the training data

Overfitting

K = 10

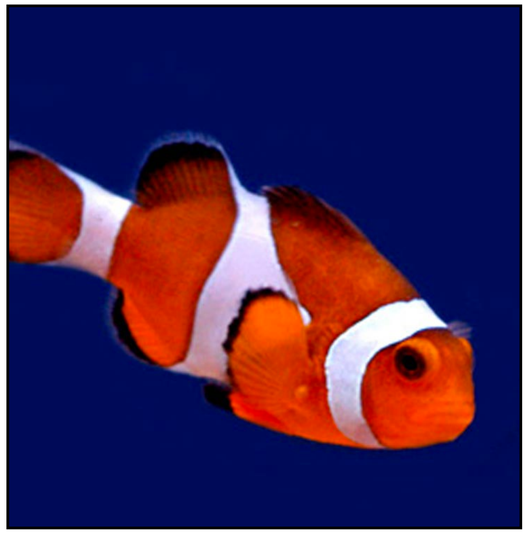


Complex model

Fits the training data

Source: Isola, Torralba, Freeman

Image classification



Classifier

“Fish”

image **x**

label y

Source: Isola, Torralba, Freeman

Image classification

Classifier

“Fish”

image **x**

label y

Source: Isola, Torralba, Freeman

Image classification

Classifier

“Fish”

image **x**

label y

Source: Isola, Torralba, Freeman

Image classification

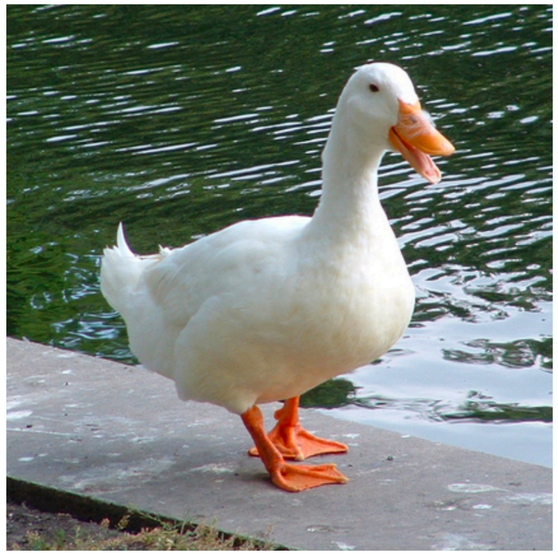


image **x**

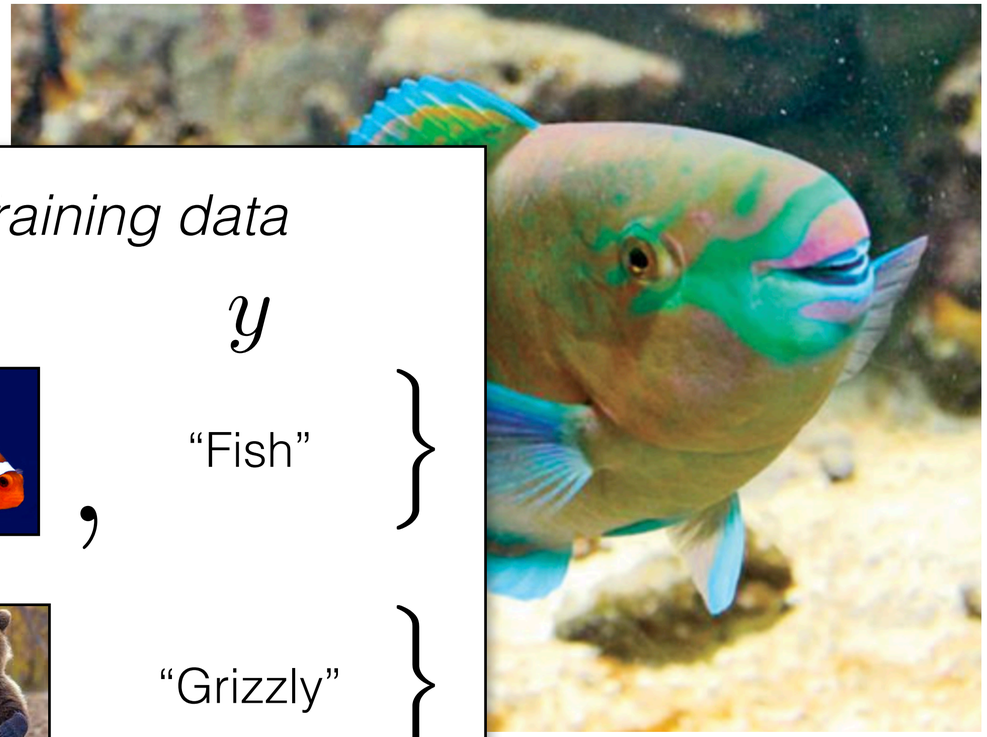
Classifier

"Duck"

label y

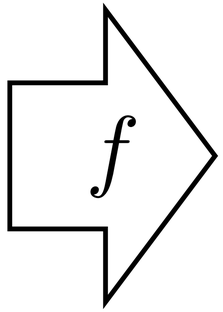
Source: Isola, Torralba, Freeman

$\mathbf{x}$



Training data

$\mathbf{x}$	y
	“Fish”
	“Grizzly”
	“Chameleon”
⋮	



y

“Fish”

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

Source: Isola, Torralba, Freeman

How to represent class labels?

One-hot vector

Training data

$\mathbf{x}$ y

{  , "Fish" }

{  , "Grizzly" }

{  , "Chameleon" }

,

⋮

→

Training data

$\mathbf{x}$ y

{  , 1 }

{  , 2 }

{  , 3 }

,

⋮

→

Training data

$\mathbf{x}$ $\mathbf{y}$

{  , [0,0,1] }

{  , [0,1,0] }

{  , [1,0,0] }

,

⋮

Source: Isola, Torralba, Freeman

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; hard to optimize!

Least squares approximation (predict 1 for true class, 0 for others)

$$\mathcal{L}(\hat{y}, y) = \sum_{k=1}^{K} (y_k - \hat{y}_k)^2$$ ← easy to optimize, but crude approximation

Cross entropy

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = H(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{k=1}^{K} y_k \log \hat{y}_k$$ ← easy to optimize, good approximation

Adapted from: Isola, Torralba, Freeman

Ground truth label $\mathbf{y}$

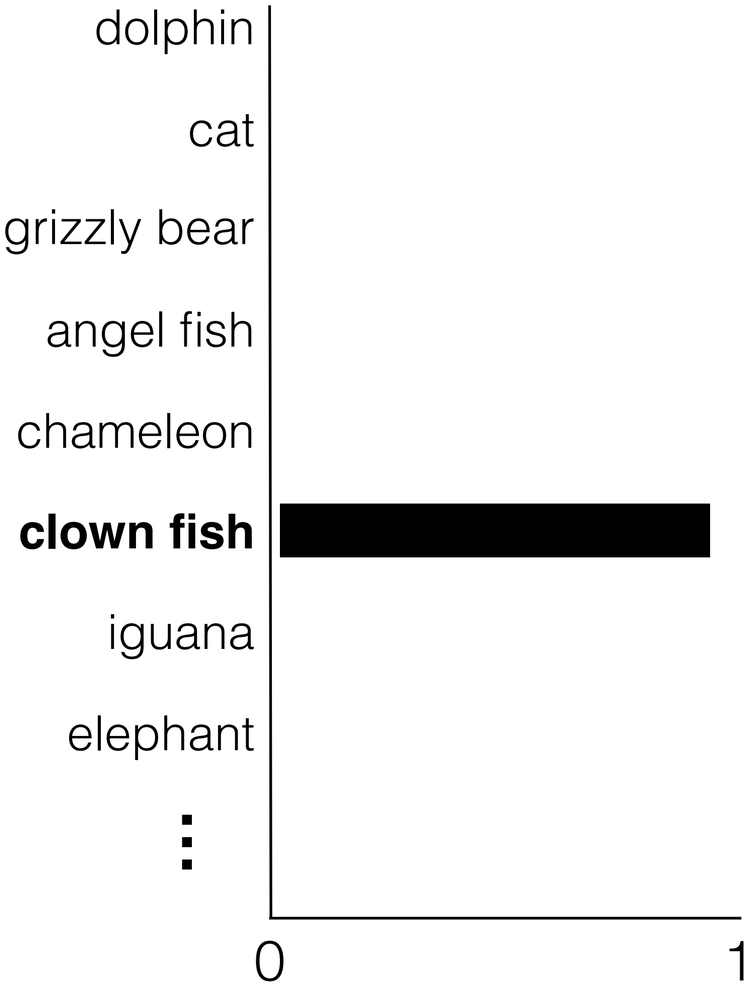
$\mathbf{x}$

[0,0,0,0,0,1,0,0,…]

Source: Isola, Torralba, Freeman

Ground truth label $\mathbf{y}$

$\mathbf{x}$



Source: Isola, Torralba, Freeman

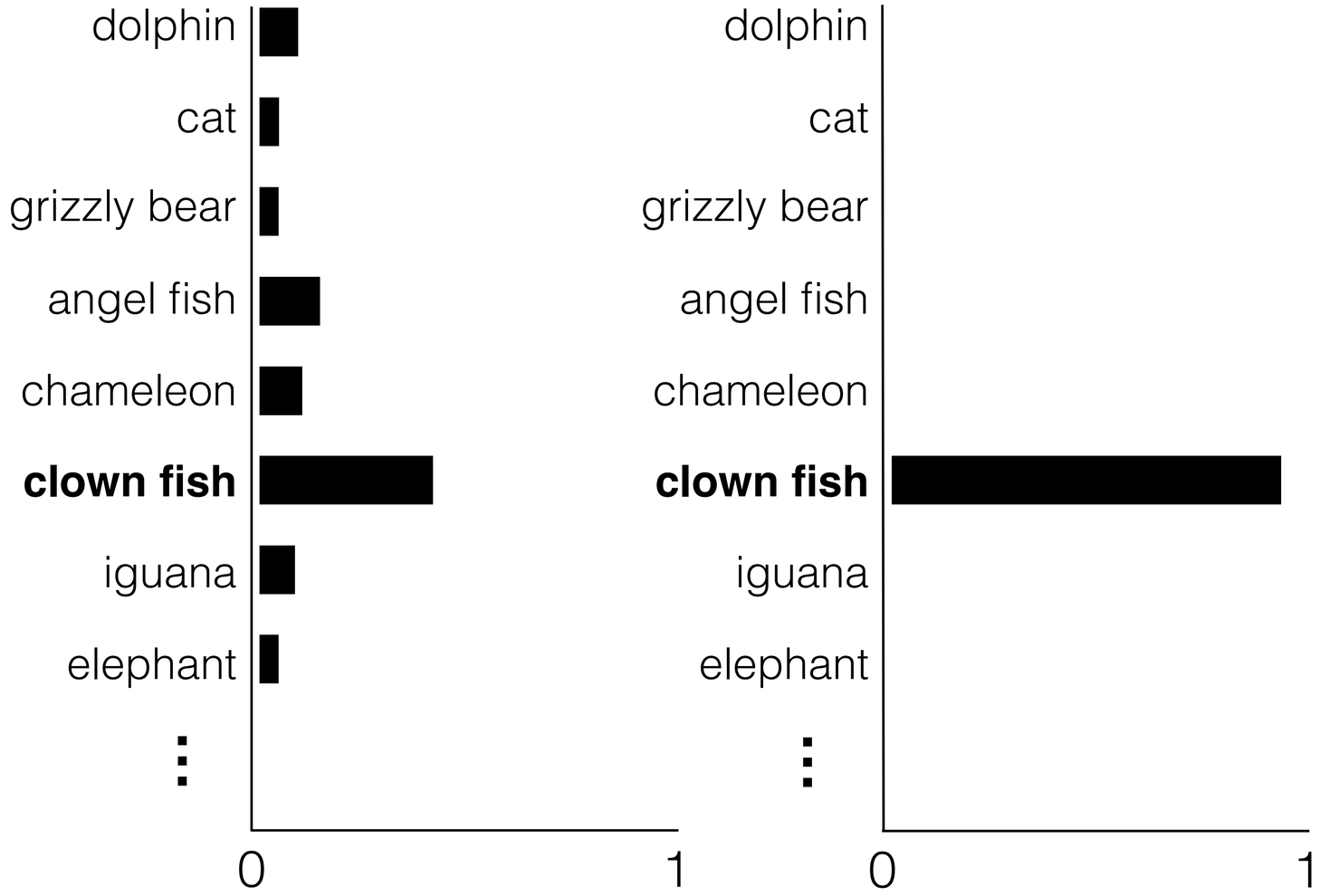
Prediction $\hat{\mathbf{y}}$

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

Ground truth label $\mathbf{y}$

$\mathbf{x}$

f



Source: Isola, Torralba, Freeman

Prediction $\hat{\mathbf{y}}$

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

Ground truth label $\mathbf{y}$

Loss

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

$\mathbf{x}$

f

dolphin
cat
grizzly bear
angel fish
chameleon
clown fish
iguana
elephant
⋮

0 1

$\odot$

dolphin
cat
grizzly bear
angel fish
chameleon
clown fish
iguana
elephant
⋮

0 1

0 1

Source: Isola, Torralba, Freeman

Prediction $\hat{\mathbf{y}}$

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

Ground truth label $\mathbf{y}$

Loss

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

$\mathbf{x}$

f

dolphin
cat
grizzly bear
angel fish
chameleon
clown fish
iguana
elephant
⋮

0 1

$\odot$

dolphin
cat
grizzly bear
angel fish
chameleon
clown fish
iguana
elephant
⋮

0 1

0 1

Source: Isola, Torralba, Freeman

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

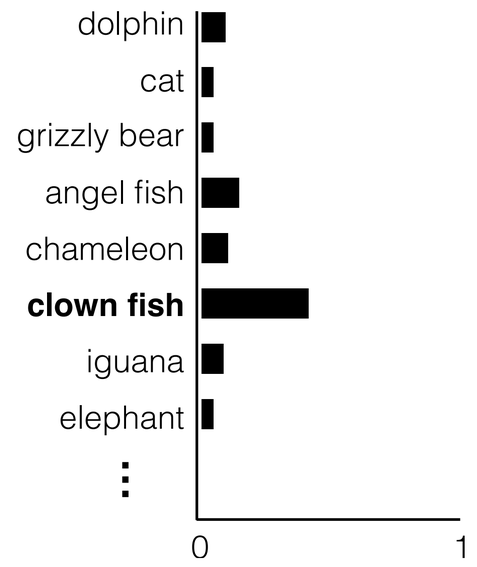
$$f_\theta : X \to \mathbb{R}^K$$

$$\mathbf{z} = f_\theta(\mathbf{x})$$ ← **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 density function**

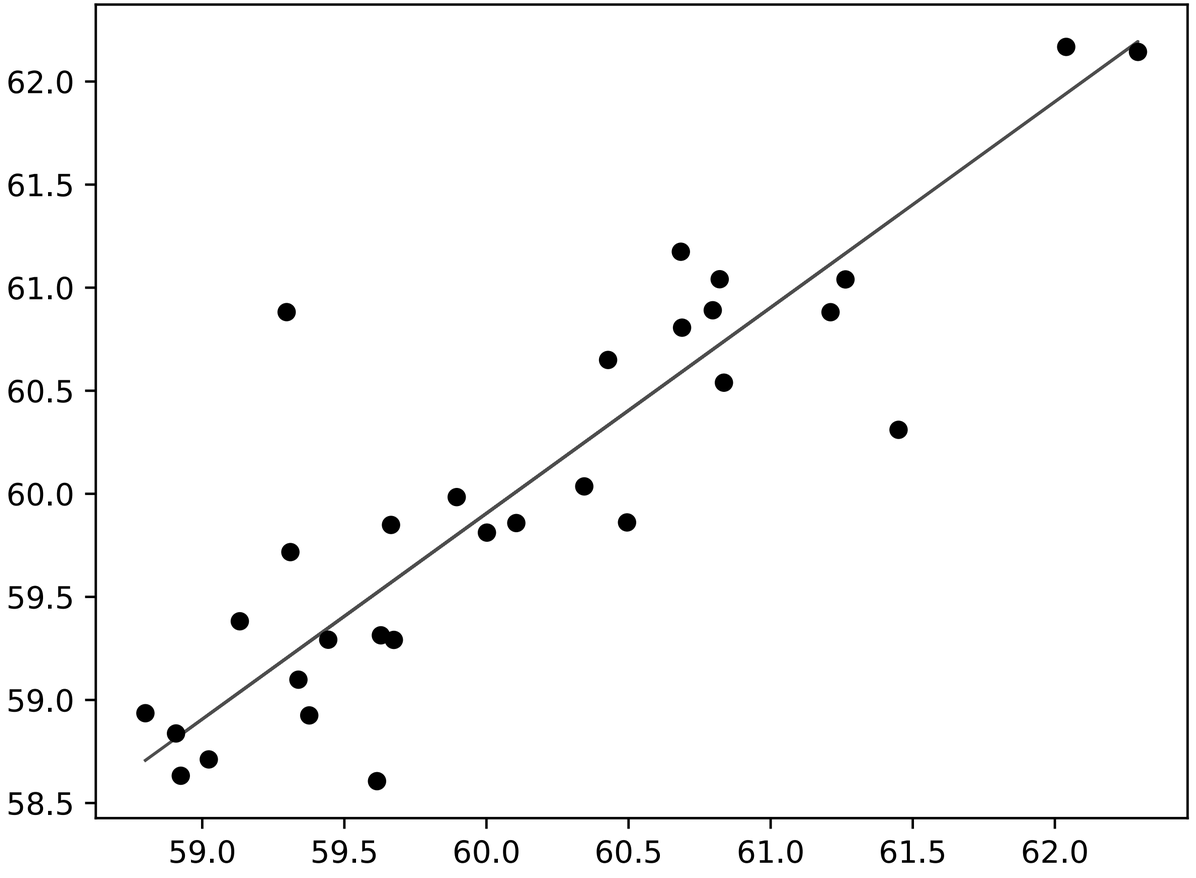
$$\hat{y}_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_j}}$$

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



Source: Isola, Torralba, Freeman

Linear regression



Predict scalar **y** from vector **x**

Logistic regression

x

$\hat{y}$

dolphin
cat
grizzly bear
angel fish
chameleon
clown fish
iguana
elephant

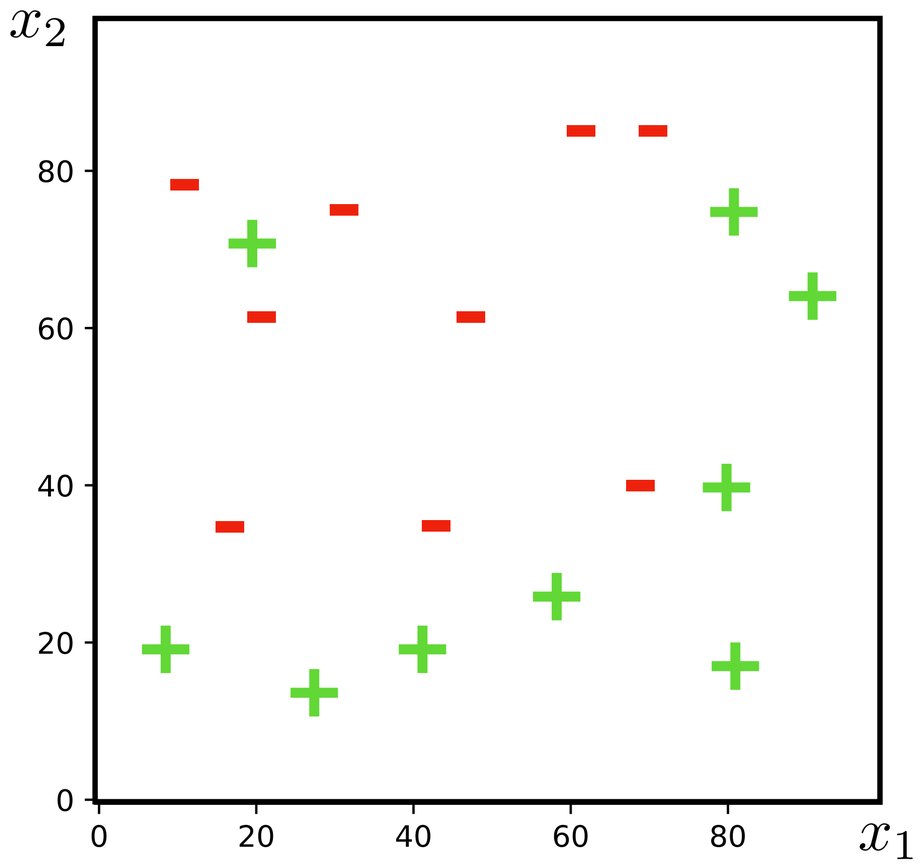
0
1

Predict probability of classes from vector **x**

Putting these ideas together: linear classifiers

Linear decision boundaries

Consider a **binary** classification problem.

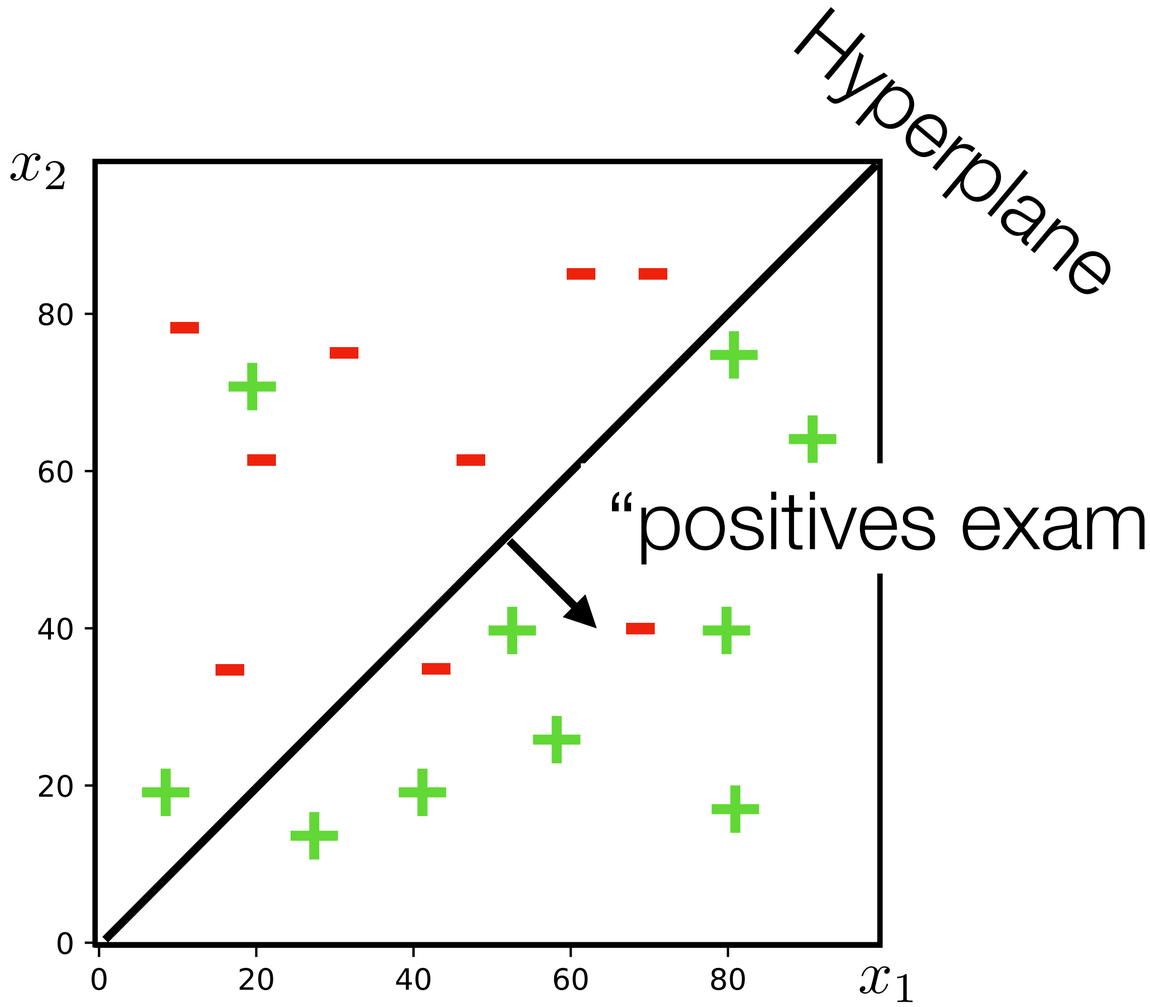


Training data

Example adapted from P. Isola

Linear decision boundaries

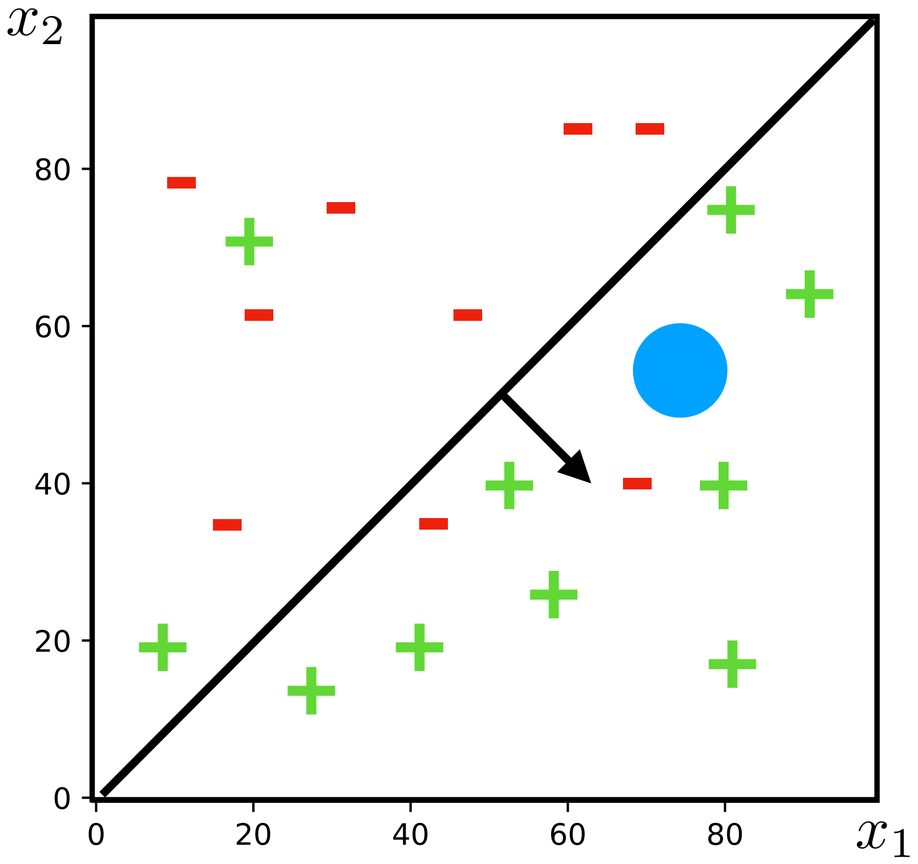
Consider a **binary** classification problem.



Training data

Example adapted from P. Isola

Linear decision boundaries



"What label is this point?"

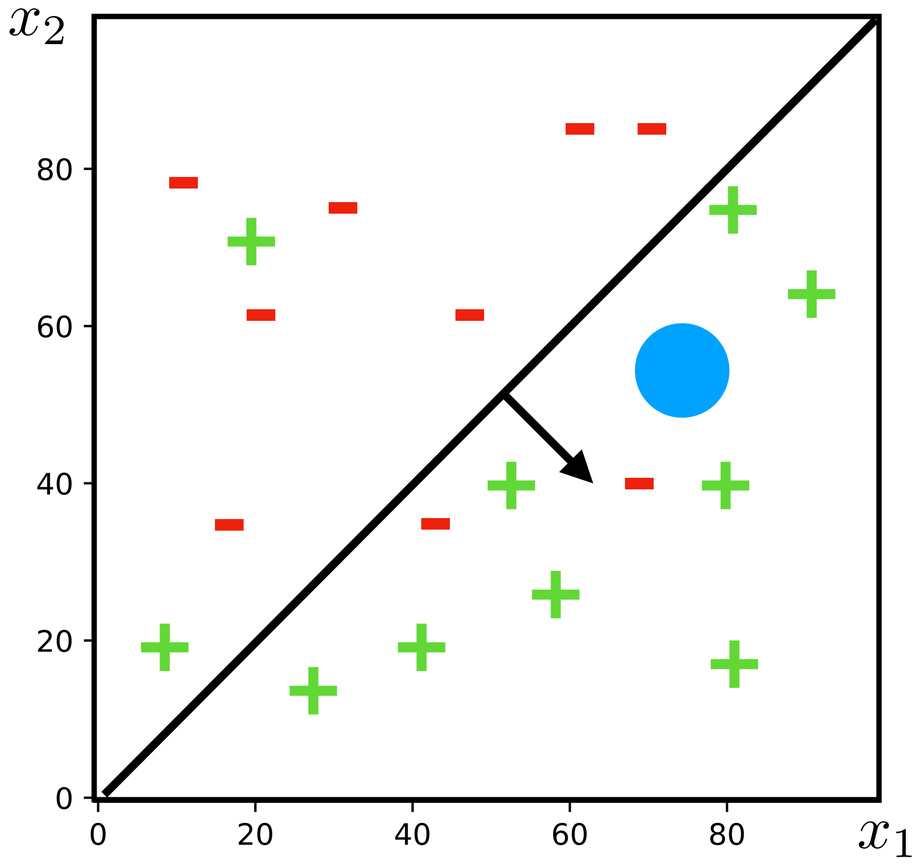
Which side of the hyperplane is **x** on?

$$\hat{y} = \mathbf{x}^T \mathbf{w} + b$$

$$g(\hat{y}) = \begin{cases} 1, & \text{if} \quad \hat{y} > 0 \\ 0, & \text{otherwise} \end{cases}$$

Example adapted from P. Isola

Linear decision boundaries



"What label is point?"

Notational simplification:

Can get rid of b by "tacking on" a 1 to x

$$\hat{y} = \tilde{\mathbf{x}}^\top \tilde{\mathbf{w}}$$

$$\tilde{x} = \begin{bmatrix} x \\ 1 \end{bmatrix} \qquad \tilde{w} = \begin{bmatrix} w \\ b \end{bmatrix}$$

Multiway classification

For a **k**-class problem, we'll make a matrix "stacking" **k** hyperplanes as rows of a matrix $W \in \mathbb{R}^{k \times d}$.

$$W = \begin{bmatrix} \text{———} & w_1 & \text{———} \\ \text{———} & w_2 & \text{———} \\ & \dots & \\ \text{———} & w_k & \text{———} \end{bmatrix}$$

To classify an example: which row has the highest dot product with **x**?

$$z = Wx + b$$

$$g(\hat{y}) = \text{argmax}_k \, z_k$$

Converting to probabilities

We want probabilistic predictions: $p(y_k \mid x)$

i.e. $\sum_k p(y_k \mid x) = 1$, and $p(y_k \mid x) \geq 0$

Recall: use the **softmax** function:

$$\hat{y} = \text{softmax}(Wx + b)$$

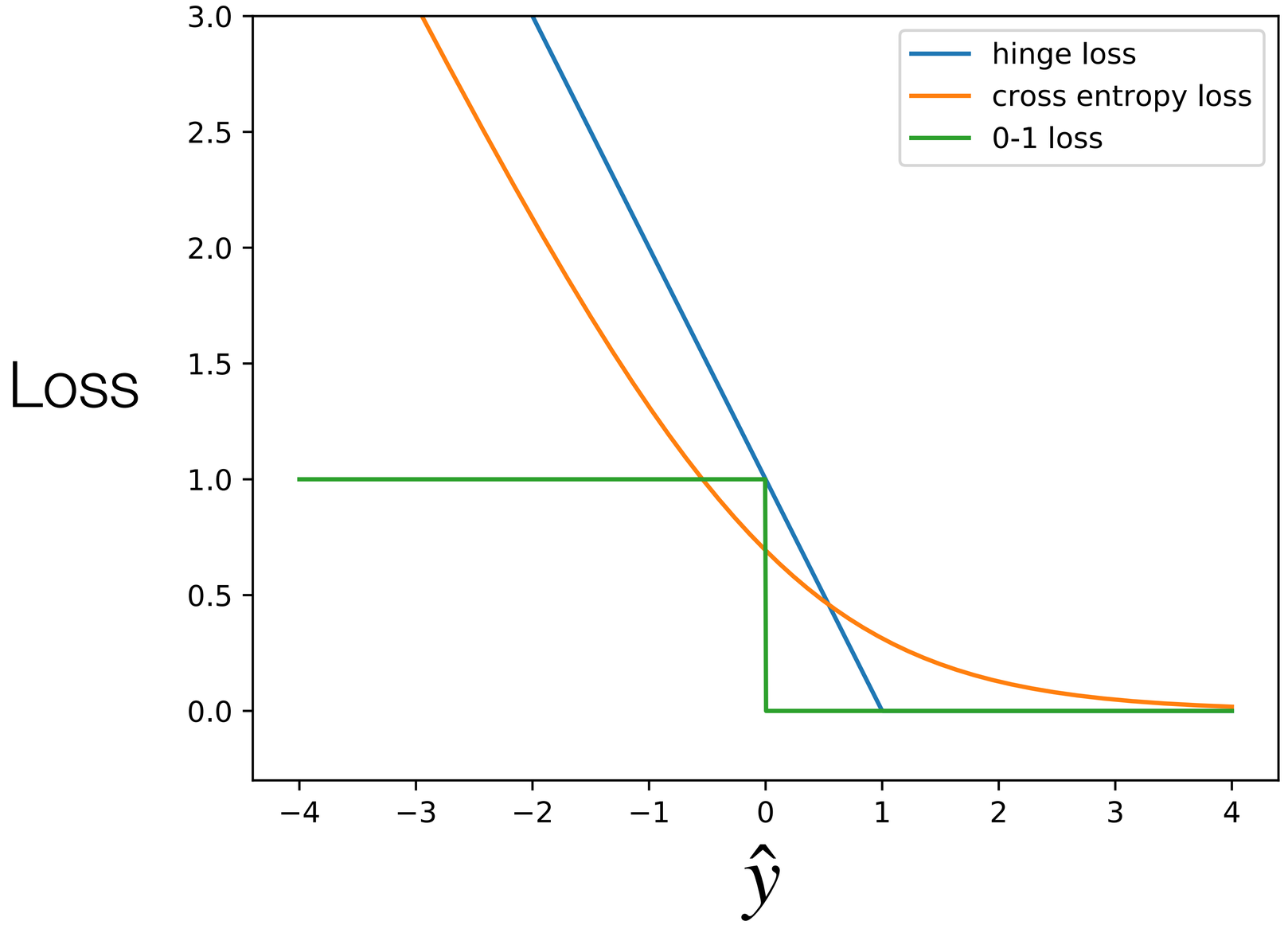
$$p(y_k \mid x) = \hat{y}_k$$

non-negative

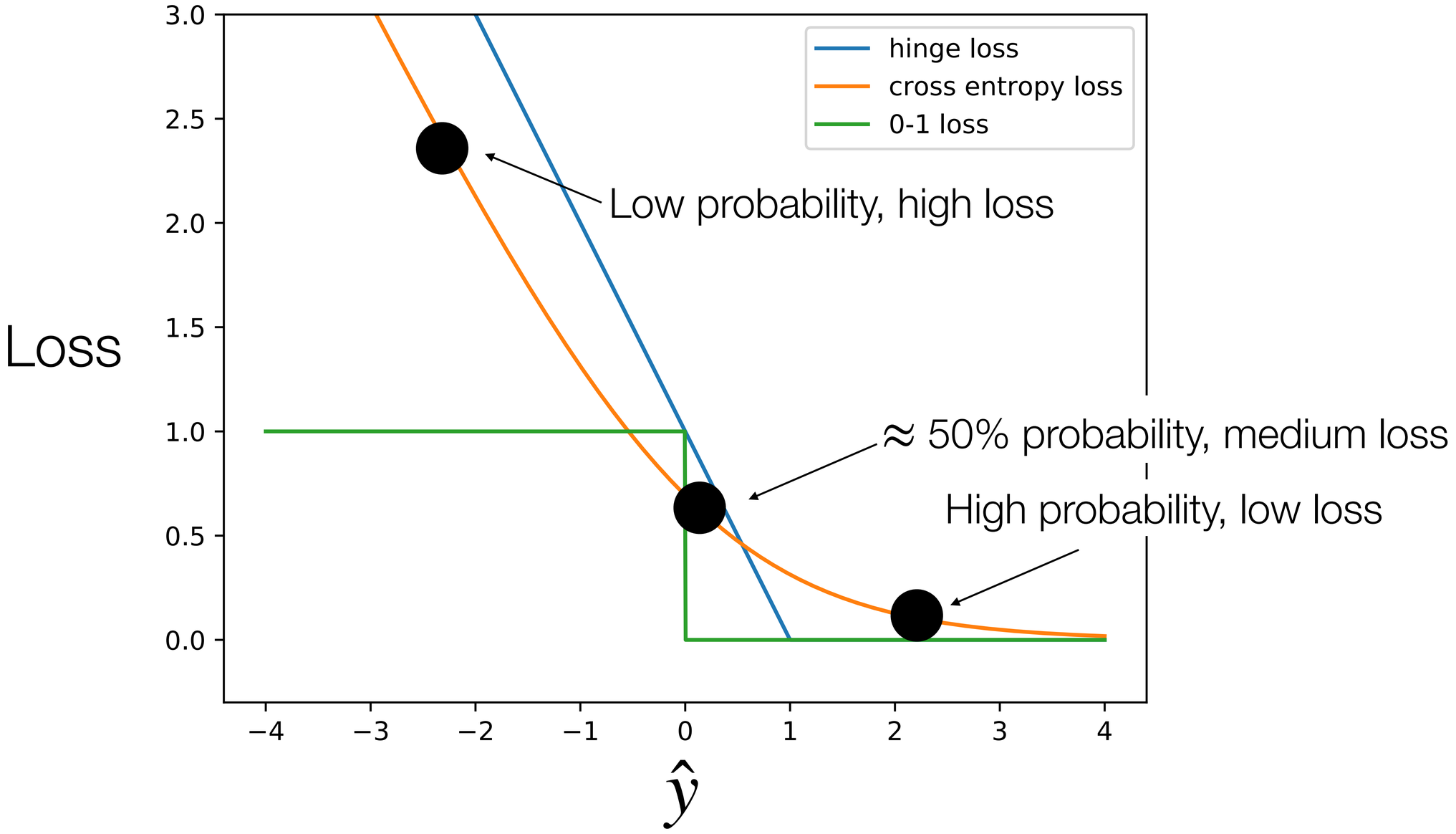
sums-to-1

$$\text{softmax}(z)_k = \frac{\exp(z_k)}{\sum_i \exp(z_i)}$$

Loss functions



Loss functions



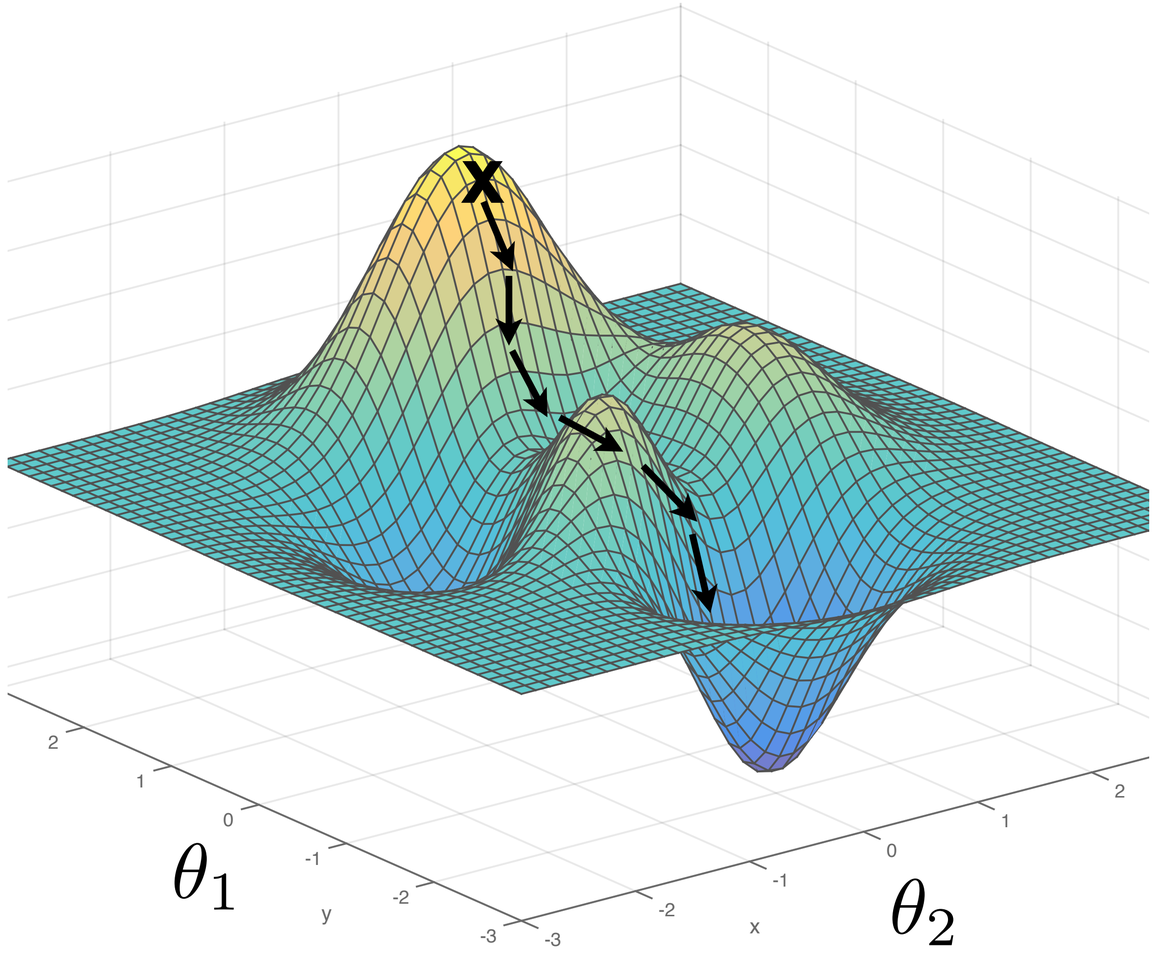
How do we learn a classifier?

$$\theta^* = \arg\min_{\theta} \underbrace{\sum_{i=1}^{N} \mathcal{L}(f_\theta(\mathbf{x}_i), \mathbf{y}_i)}_{J(\theta)}$$

Source: Isola, Torralba, Freeman

Gradient descent

$J(\theta)$



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

Source: Isola, Torralba, Freeman

Gradient descent

$$\theta^* = \arg\min_\theta \underbrace{\sum_{i=1}^{N} \mathcal{L}(f_\theta(\mathbf{x}_i), \mathbf{y}_i)}_{J(\theta)}$$

One iteration of gradient descent:

$$\theta^{t+1} = \theta^t - \eta_t \nabla_\theta J(\theta)\Big|_{\theta=\theta^t}$$

learning rate

Source: Isola, Torralba, Freeman

Gradient descent

$$\theta^{t+1} = \theta^t - \eta_t \nabla_\theta J(\theta) \Big|_{\theta=\theta^t}$$

What's this again?

1. Gradient of the loss w.r.t. the classifier's parameters
2. Direction of steepest descent
3. "Local" linear approximation to the function

Gradient descent

Take gradient at current θ

$$\theta^{t+1} = \theta^t - \eta_t \nabla_\theta J(\theta) \Big|_{\theta=\theta^t}$$

$$\nabla_\theta J(\theta) = \begin{bmatrix} \frac{\partial J}{\theta_1} \\ \frac{\partial J}{\theta_2} \\ \cdots \\ \frac{\partial J}{\theta_d} \end{bmatrix}$$

Vector of partial derivatives for loss

Estimating the gradient

Idea #1: finite differences $\frac{f(x+\epsilon) - f(x)}{\epsilon}$

$$\nabla J(\theta) = \begin{bmatrix} \frac{\partial J}{\theta_1} \\ \frac{\partial J}{\theta_2} \\ \cdots \\ \frac{\partial J}{\theta_d} \end{bmatrix} \approx \frac{J(\theta + \epsilon e_2) - J(\theta - \epsilon e_2)}{2\epsilon}$$

where e_2 is vector of zeros except for a 1 in 2nd component

Estimating the gradient

Idea #1: finite differences $\frac{f(x+\epsilon) - f(x)}{\epsilon}$

$$\nabla J(\theta) = \begin{bmatrix} \frac{\partial J}{\theta_1} \\ \frac{\partial J}{\theta_2} \\ \cdots \\ \frac{\partial J}{\theta_d} \end{bmatrix}$$

- Easy to compute but **slow**.
- Inaccurate in some cases, unless careful
- Useful for **checking** other methods

Estimating the gradient

Idea #2: compute it analytically using calculus.

Simple example: binary labels, squared loss, and regularization on θ

$$J(\theta) = \lambda||\theta||^2 + \frac{1}{N}\sum_{i=1}^{N}(y_i - \theta^\top x_i)^2$$

$$\nabla_\theta$$

$$\nabla_\theta J(\theta) = 2\lambda\theta - \frac{1}{N}\sum_{i=1}^{N} 2(y_i - \theta^\top x_i)x_i$$

Adapted from D. Fouhey

Analyzing the gradient descent update

Recall update rule: $\theta^{t+1} = \theta^t - \eta_t \nabla_\theta J(\theta)\Big|_{\theta=\theta^t}$

How does this change θ ?

"Decay" toward 0 Scalar α

$$-\nabla_\theta J(\theta) = -2\lambda\theta - \frac{1}{N}\sum_{i=1}^{N} 2(y_i - \theta^\top x_i)x_i$$

Adapted from D. Fouhey

Analyzing the gradient descent update

What happens at each example?

Scalar α

$$-\nabla_\theta J(\theta) = -2\lambda\theta - \frac{1}{N}\sum_{i=1}^{N} 2(y_i - \theta^\top x_i)x_i$$

If $y > \theta^\top x$ (too *low*): then $\theta_{t+1} = \theta + \alpha x$ for some α

Dot product before: $\theta^\top x$

Dot product after: $(\theta + \alpha x)^\top x = \theta^\top x + \alpha x^\top x$

Adapted from D. Fouhey

Computational issues with gradient descent

Batch gradient descent

Loss function:

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} L(x_i, y_i, \theta)$$

Its gradient is the **sum of gradients** for each example:

$$\nabla J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \nabla L(x_i, y_i, \theta)$$

Requires iterating over every training example each gradient step!

Can we speed this up?

Stochastic gradient descent

This is just an **average**:

$$\nabla J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \nabla L(x_i, y_i, \theta)$$

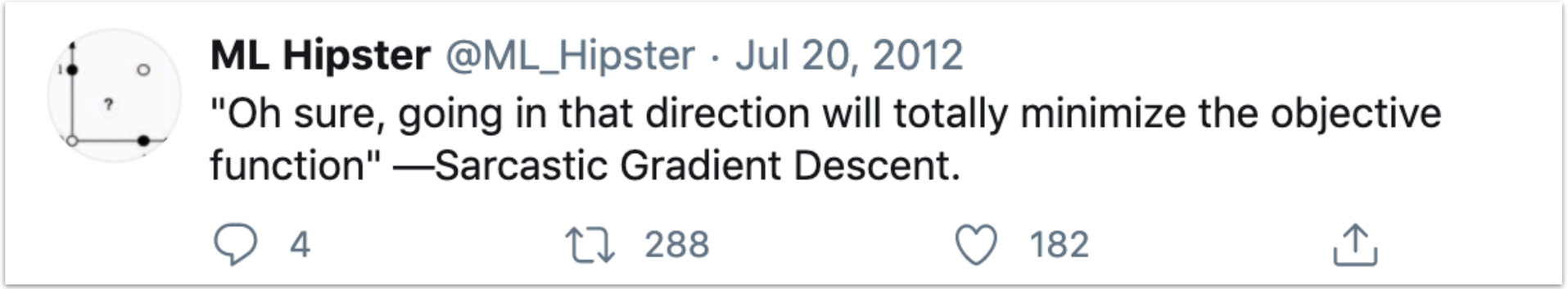
We know from statistics that we can estimate the average of a full "population" from a **sample.**

$$\nabla J(\theta) \approx \frac{1}{|B|} \sum_{i \in B} \nabla L(x_i, y_i, \theta)$$

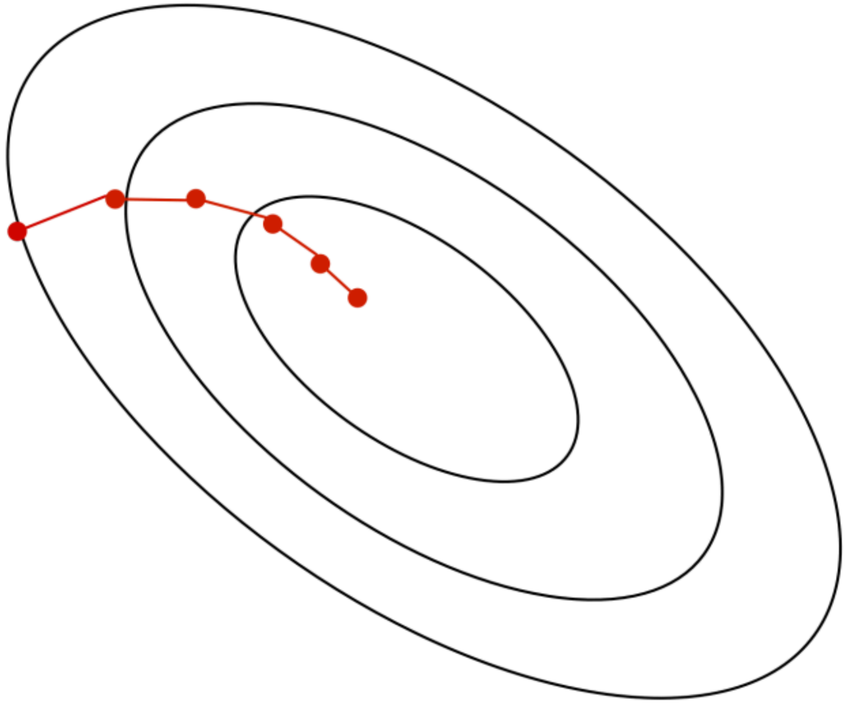
where B is a **minibatch:** a random subset of examples.

This is called **stochastic gradient descent (SGD)**.

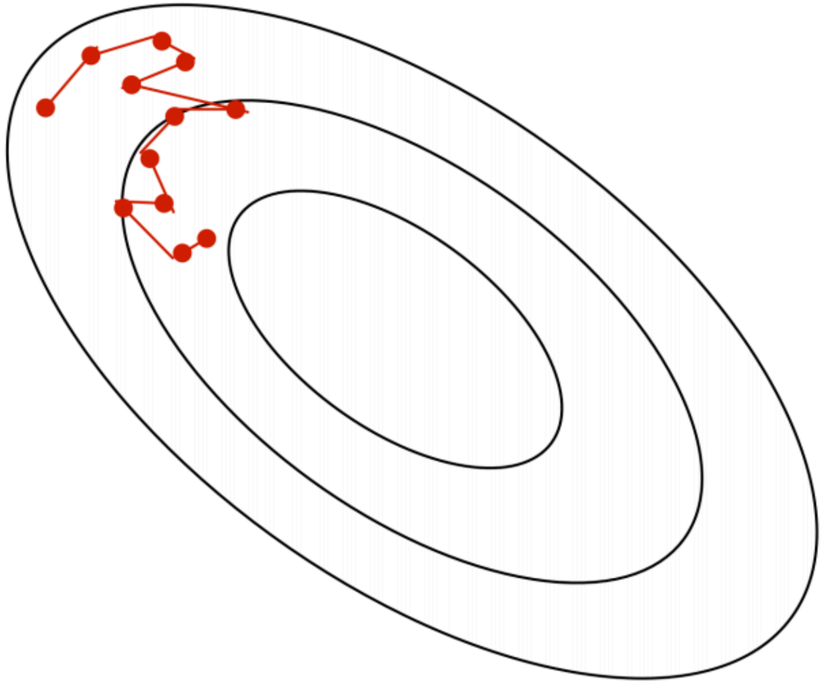
Stochastic gradient descent



Stochastic gradient descent



Batch gradient descent

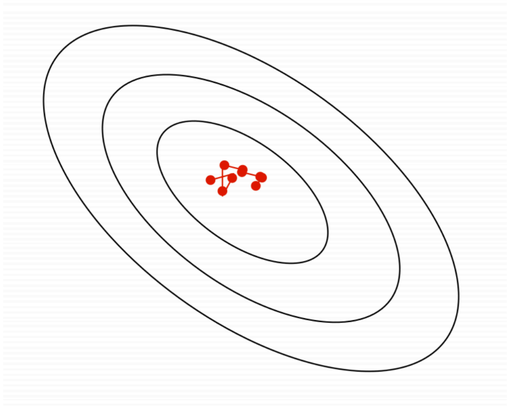


Stochastic gradient descent

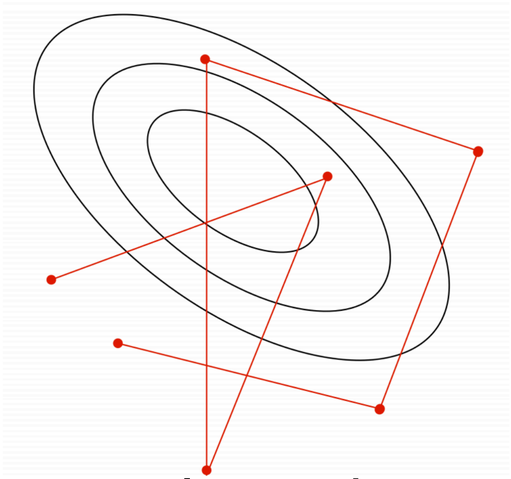
Source: R. Grosse

Learning rate

- Sensitive to the learning rate: $\theta^{t+1} = \theta^t - \eta_t \nabla_\theta J(\theta)\Big|_{\theta=\theta^t}$



Small learning rate

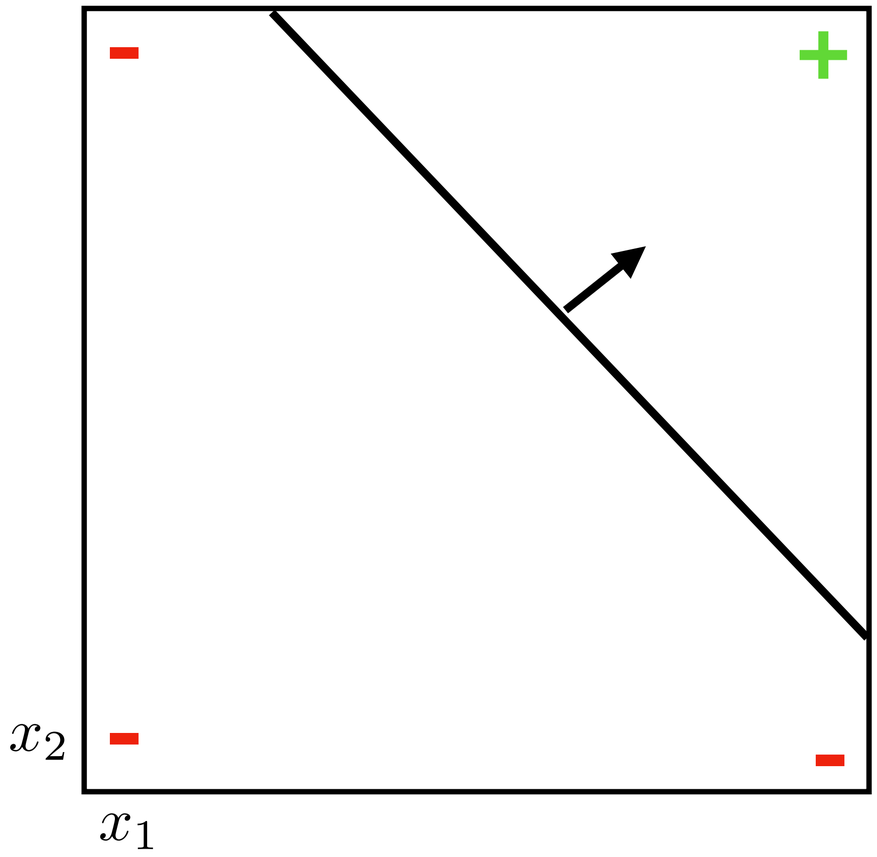


Large learning rate

- Often start with high learning rate, and decrease over time
- For example: start with $\eta_t = 0.1$, then decrease to $\eta_t = 0.01$ when training loss plateaus

Figure source: R. Grosse

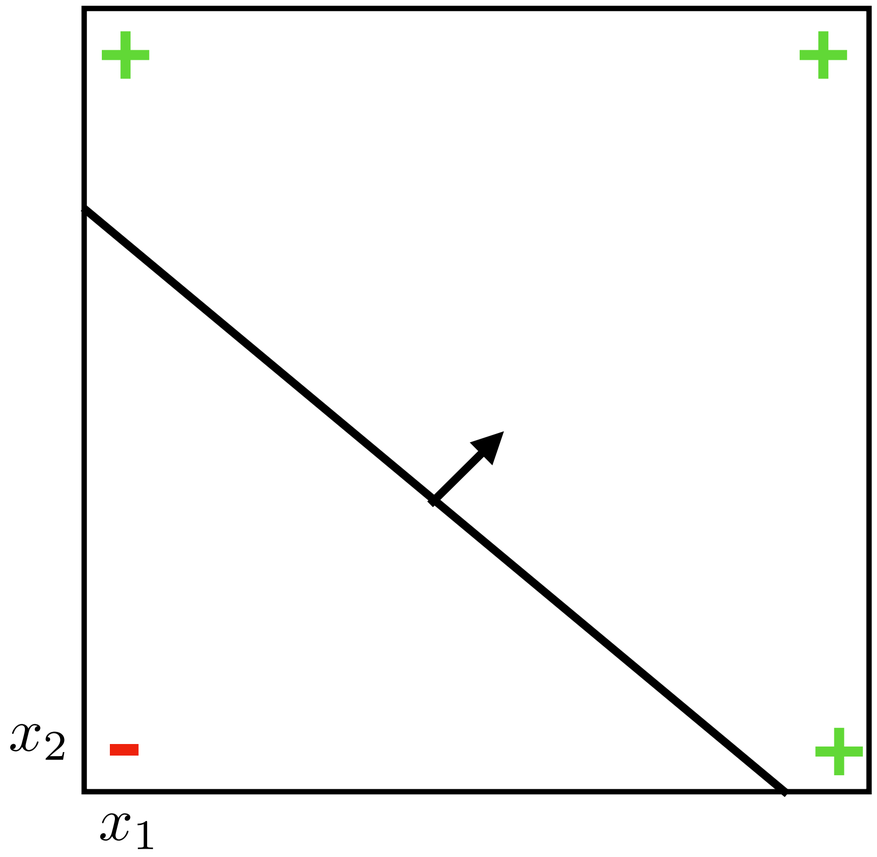
Limitations to linear classifiers



x_1 \ x_2	0	1
0	0	0
1	0	1

AND

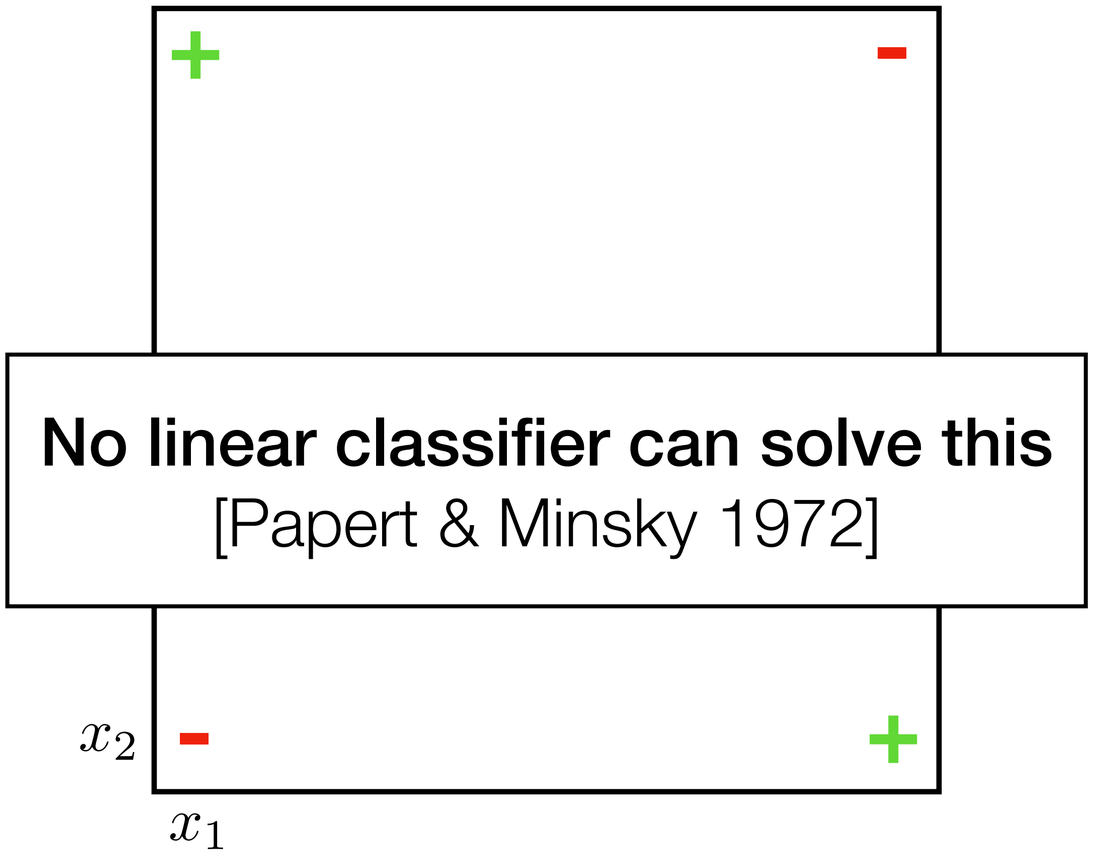
Limitations to linear classifiers



x_1 \ x_2	0	1
0	0	1
1	1	1

OR

Limitations to linear classifiers



x_1 \ x_2	0	1
0	1	0
1	0	1

XOR

Next lecture: neural networks