Linear Regression

Prepared by: Joseph Bakarji

Data

Given a table of numbers, what can you do?

Living area ($feet^2$)	#bedrooms	Price (1000$s)	.....
2104	3	400	
1600	3	330	
2400	3	369	
1416	2	232	
3000	4	540	
⋮	⋮	⋮	

→

	x_1	x_2	x_3	x_4	.....
Apt. 1	$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$	$x_4^{(1)}$	
Apt. 2	$x_1^{(2)}$	$x_2^{(2)}$	$x_3^{(2)}$	$x_4^{(2)}$	⋮
Apt. 3	$x_1^{(3)}$	$x_2^{(3)}$	$x_3^{(3)}$	$x_4^{(3)}$	
Apt. 4	$x_1^{(4)}$	$x_2^{(4)}$	$x_3^{(4)}$	$x_4^{(4)}$	
	⋮	⋮	⋮	⋮	

- Visualization: Look at it!
- Find statistical features: Mean, median, outliers etc.
- Clean it: missing values, …

What are the inputs and outputs?

- **Inputs**: quantities that are typically **given**
- **Outputs**: quantities we want to **predict**

Living area (feet^2)	#bedrooms	Price (1000\$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
$\vdots$	$\vdots$	$\vdots$

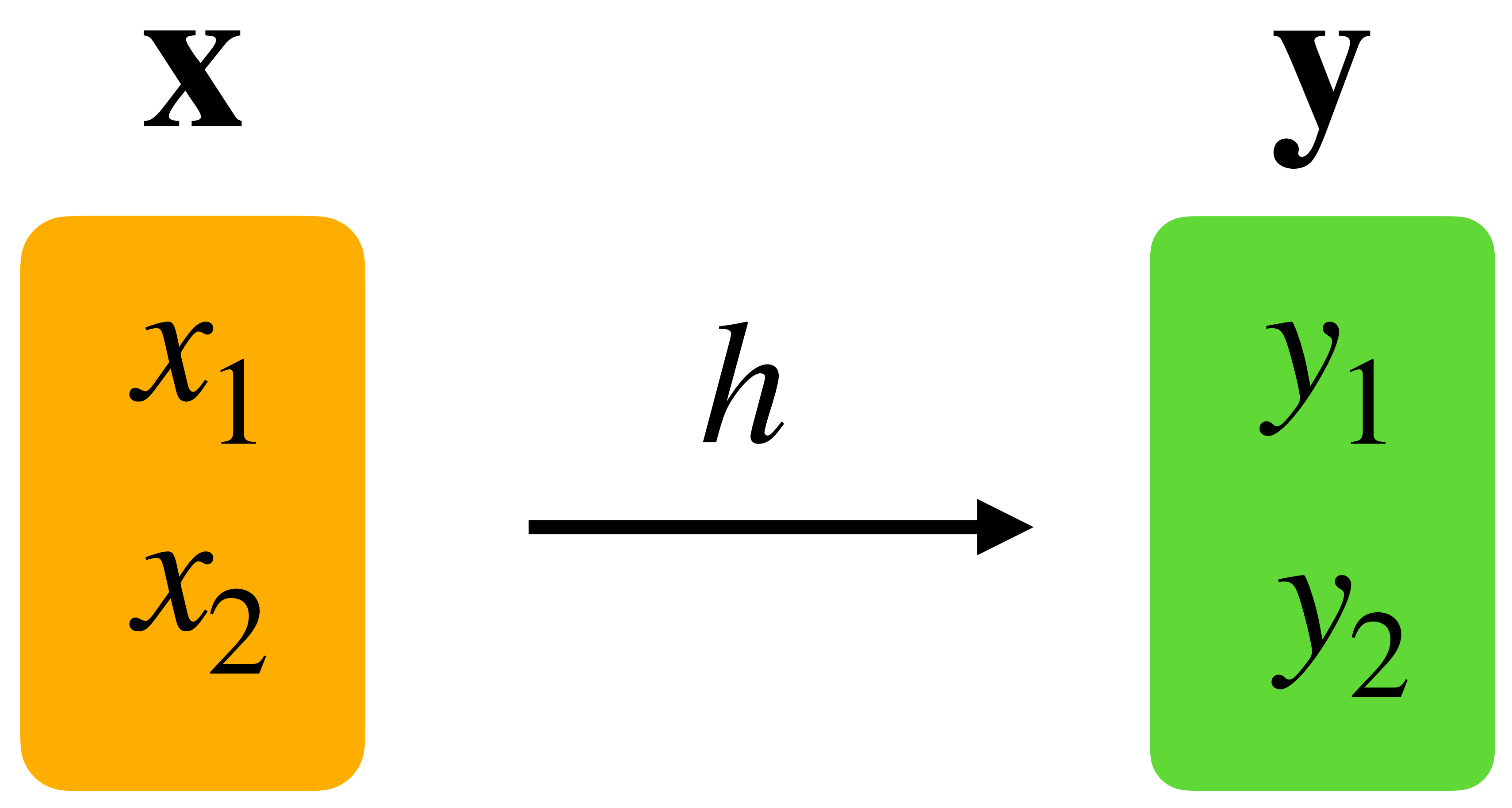
$\rightarrow$

	Inputs		Outputs		
	x_1	x_2	x_3	x_4	.....
Apt. 1	$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$	$x_4^{(1)}$	
Apt. 2	$x_1^{(2)}$	$x_2^{(2)}$	$x_3^{(2)}$	$x_4^{(2)}$	$\vdots$
Apt. 3	$x_1^{(3)}$	$x_2^{(3)}$	$x_3^{(3)}$	$x_4^{(3)}$	
Apt. 4	$x_1^{(4)}$	$x_2^{(4)}$	$x_3^{(4)}$	$x_4^{(4)}$	
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

Given inputs, predict outputs

	Inputs		Outputs	
	x_1	x_2	y_1	y_2
Apt. 1	$x_1^{(1)}$	$x_2^{(1)}$	$y_1^{(1)}$	$y_2^{(1)}$
Apt. 2	$x_1^{(2)}$	$x_2^{(2)}$	$y_1^{(2)}$	$y_2^{(2)}$
Apt. 3	$x_1^{(3)}$	$x_2^{(3)}$	$y_1^{(3)}$	$y_2^{(3)}$
Apt. 4	$x_1^{(4)}$	$x_2^{(4)}$	$y_1^{(4)}$	$y_2^{(4)}$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Given Input	$x_1^{(n)}$	$x_2^{(n)}$	??	**Unknown output**

Supervised Learning



Given the data,
find a **function** h, a.k.a a **hypothesis**,
that predicts outputs, given inputs

$$\mathbf{y} = h(\mathbf{x})$$

Assume multiple inputs, 1 output

x_1	x_2	y	
Living area (feet^2)	#bedrooms	Price (1000\$s)	.....
2104	3	400	
1600	3	330	
2400	3	369	
1416	2	232	
3000	4	540	
$\vdots$	$\vdots$	$\vdots$	

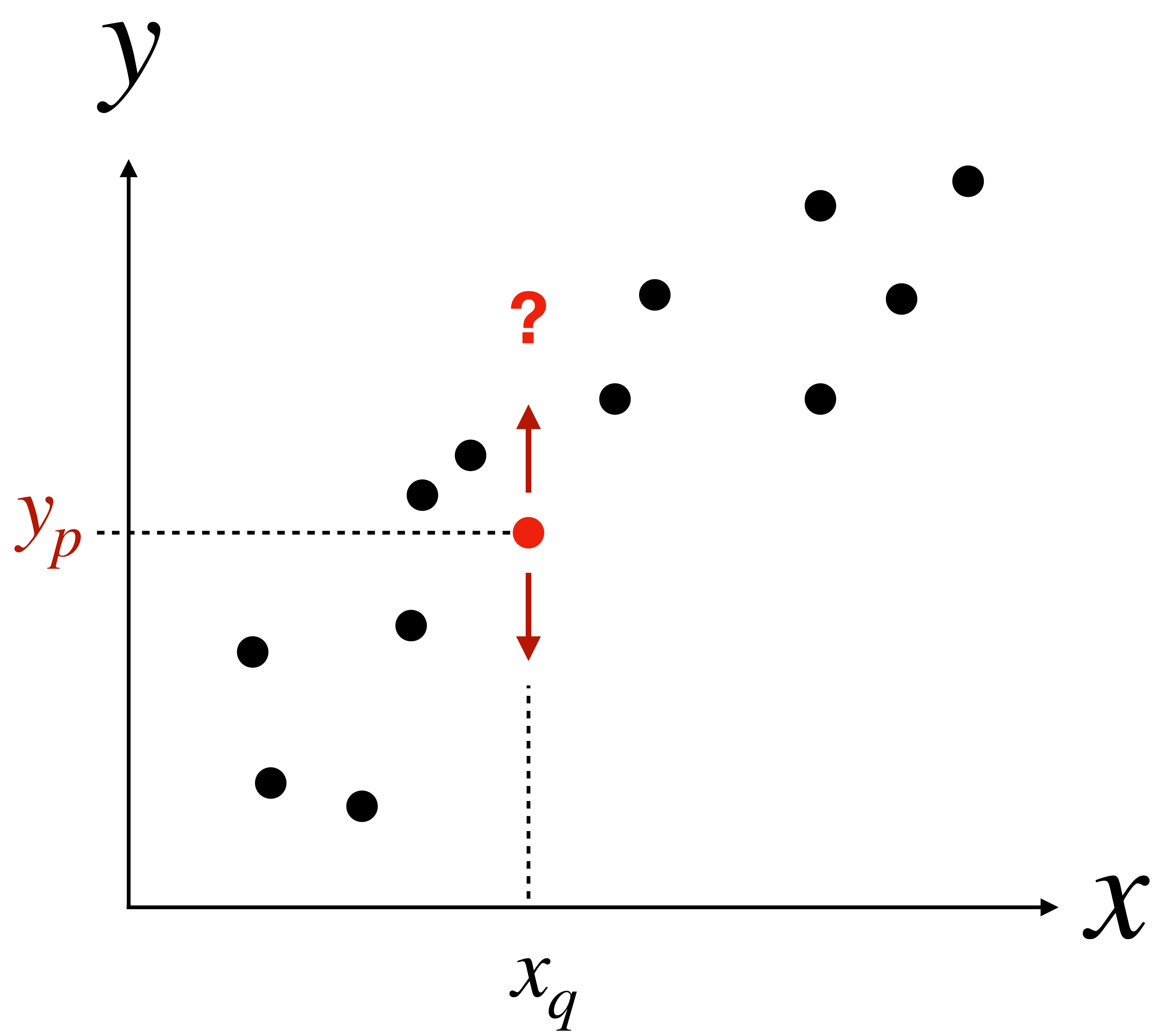
$\longrightarrow$

Inputs		Output
x_1	x_2	y
$x_1^{(1)}$	$x_2^{(1)}$	$y^{(1)}$
$x_1^{(2)}$	$x_2^{(2)}$	$y^{(2)}$
$x_1^{(3)}$	$x_2^{(3)}$	$y^{(3)}$
$x_1^{(4)}$	$x_2^{(4)}$	$y^{(4)}$
$\vdots$	$\vdots$	$\vdots$

Given new input, what's the output?

(size) Input	(price) Output
x	y
$x^{(1)}$	$y^{(1)}$
$x^{(2)}$	$y^{(2)}$
$x^{(3)}$	$y^{(3)}$
$x^{(4)}$	$y^{(4)}$
$\vdots$	$\vdots$

Query x_q y_p **??** **Prediction**



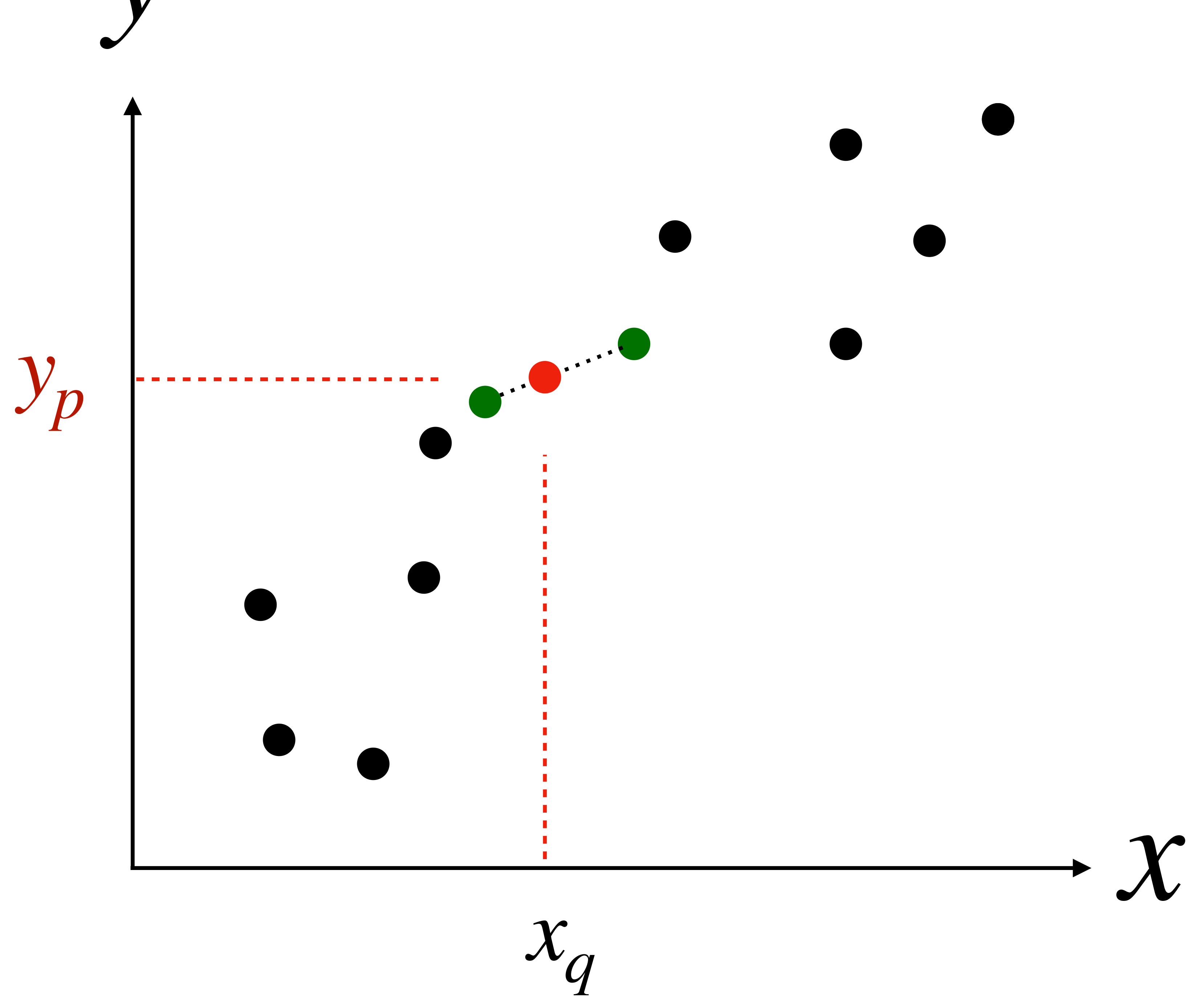
Given new input, what's the output?

Input	Output
x	y
$x^{(1)}$	$y^{(1)}$
$x^{(2)}$	$y^{(2)}$
$x^{(3)}$	$y^{(3)}$
$x^{(4)}$	$y^{(4)}$
$\vdots$	$\vdots$

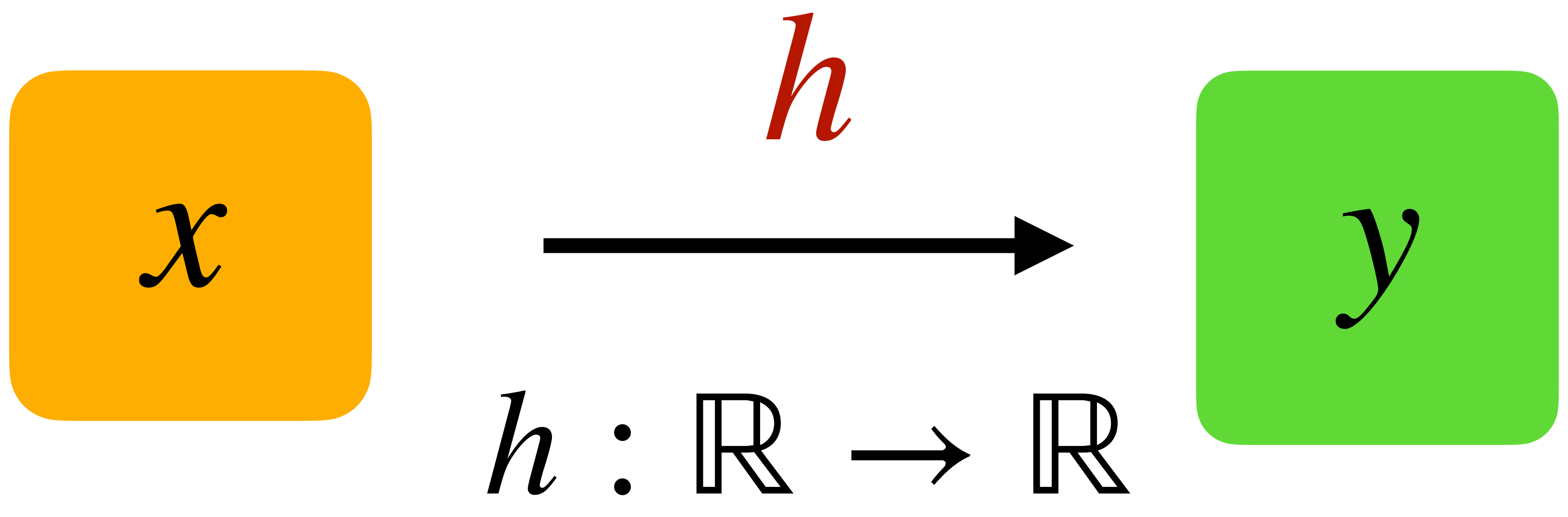
Query x_q **??** **Prediction**

Weighted average of surrounding points

K-Nearest Neighbors (KNN)

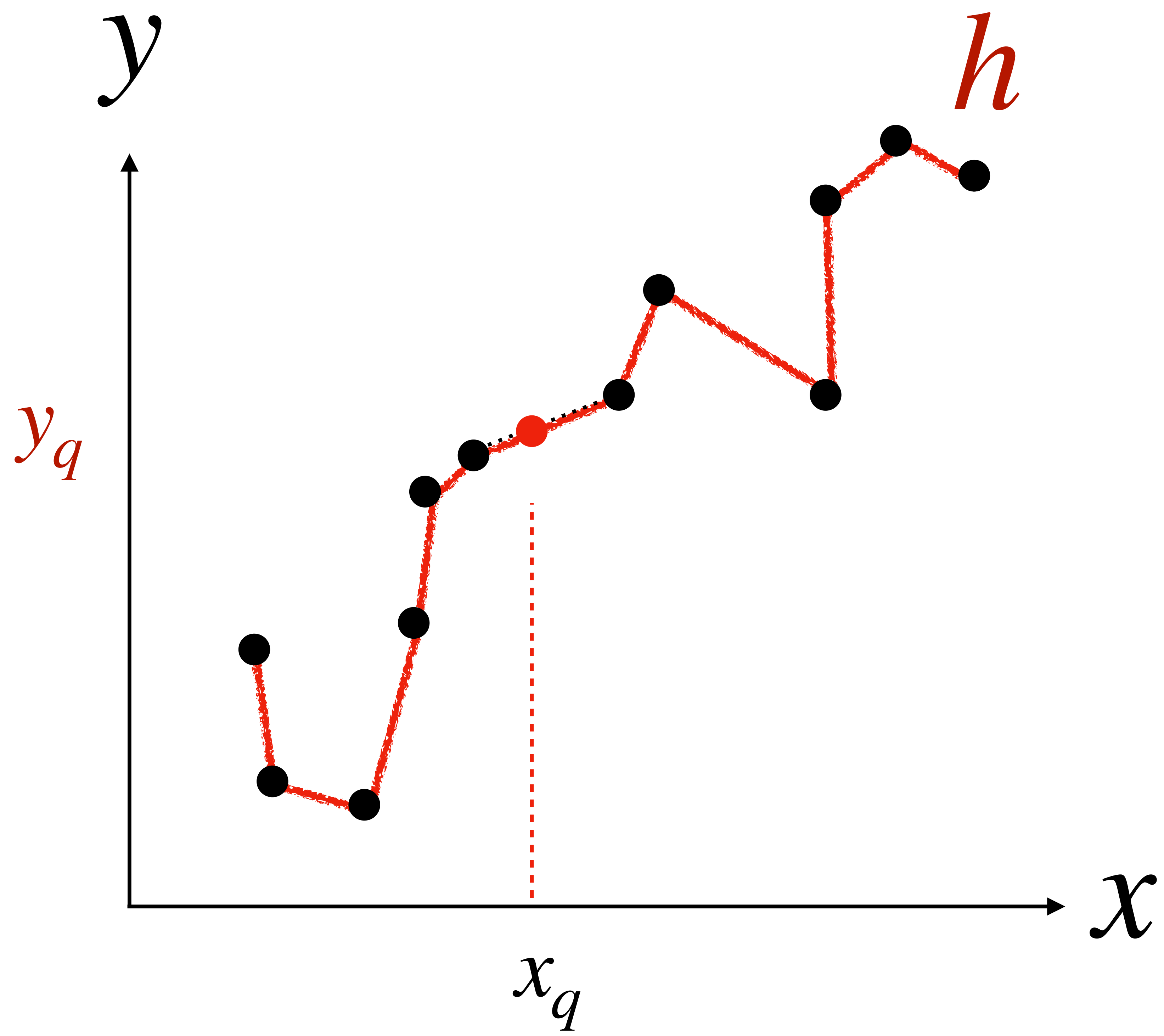


Given new input, what's the output?

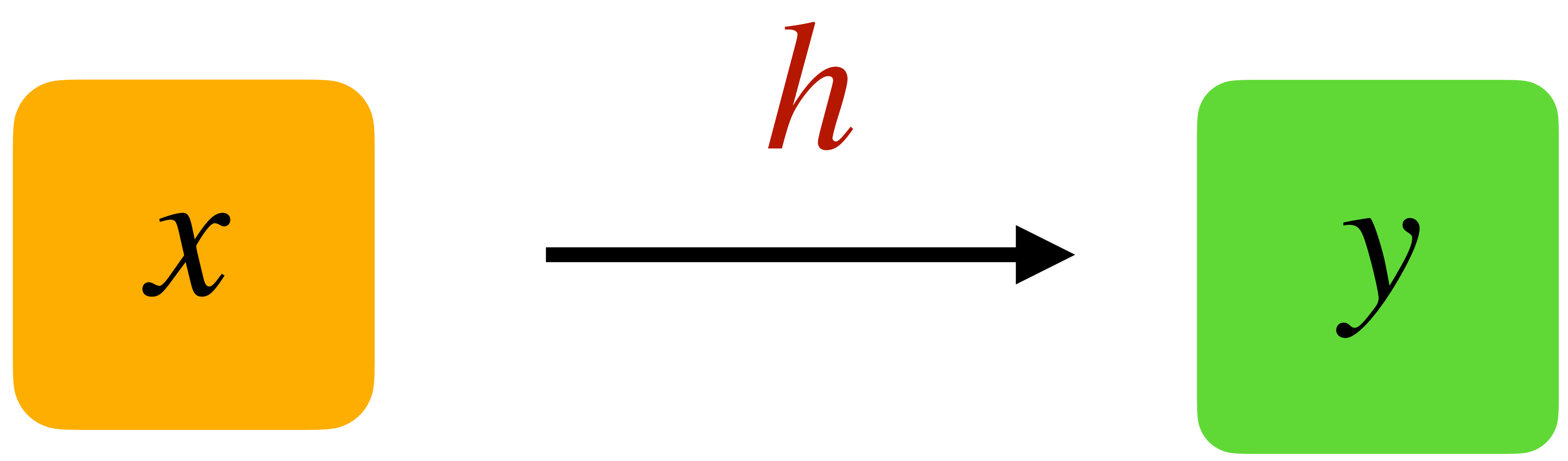


Given the data,
find a **function** h, a.k.a **hypothesis,**
that predicts an output, given an input

Linear Interpolation

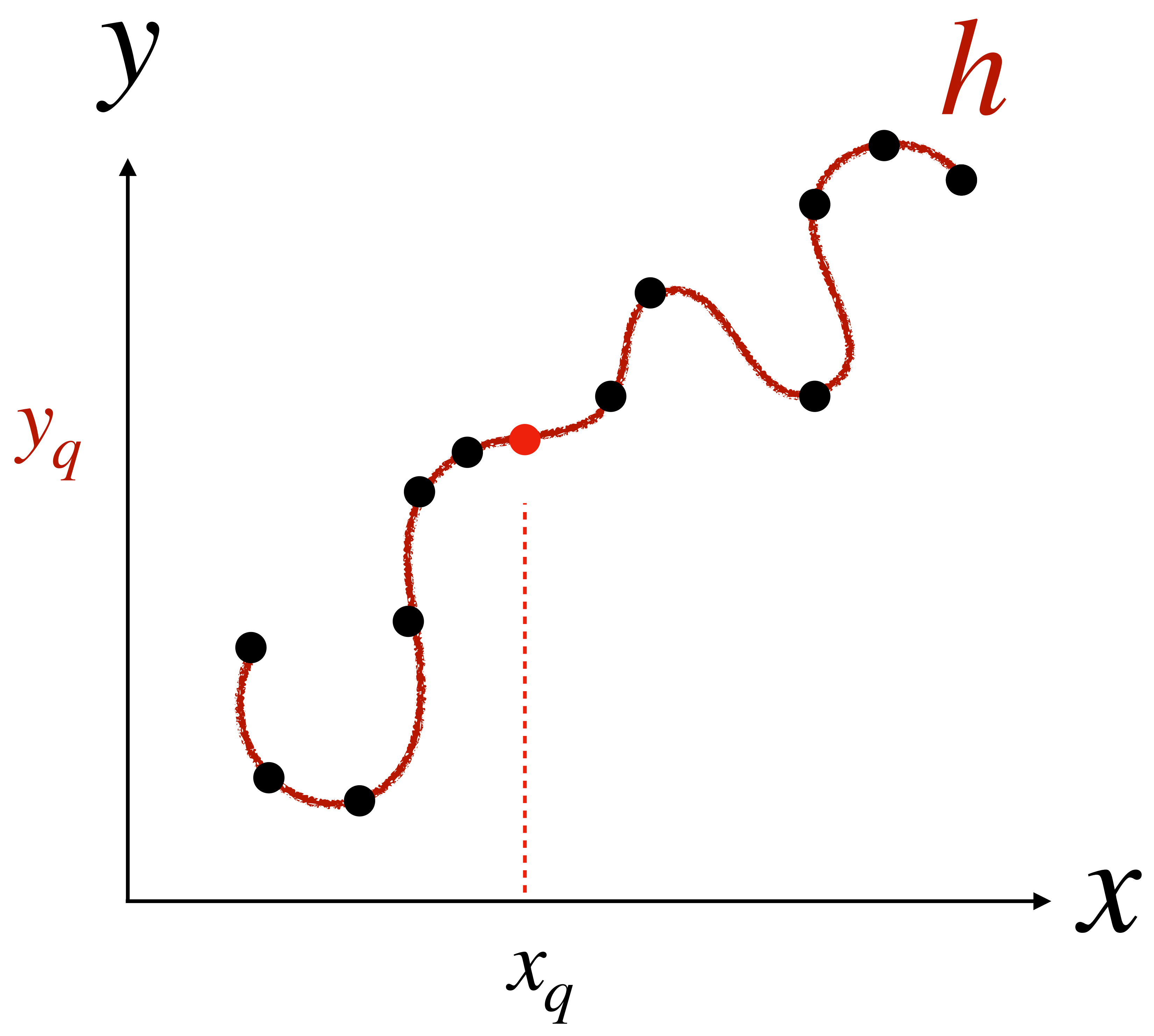


Given new input, what's the output?

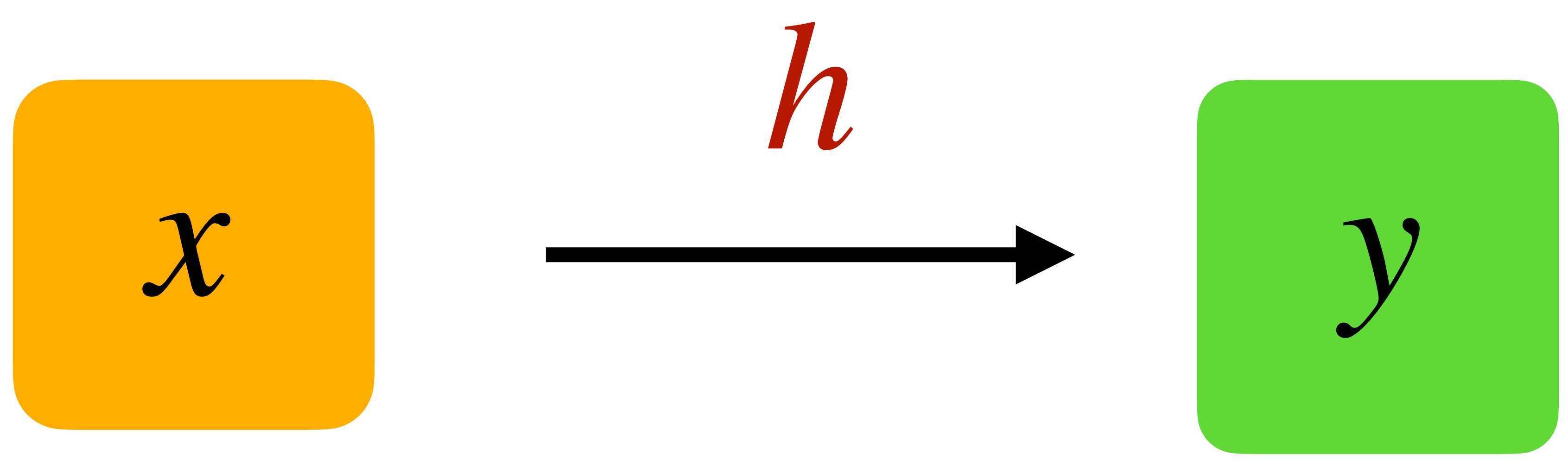


Given the data,
find a **function** h, a.k.a **hypothesis**,
that predicts an output, given an input

Polynomial Interpolation

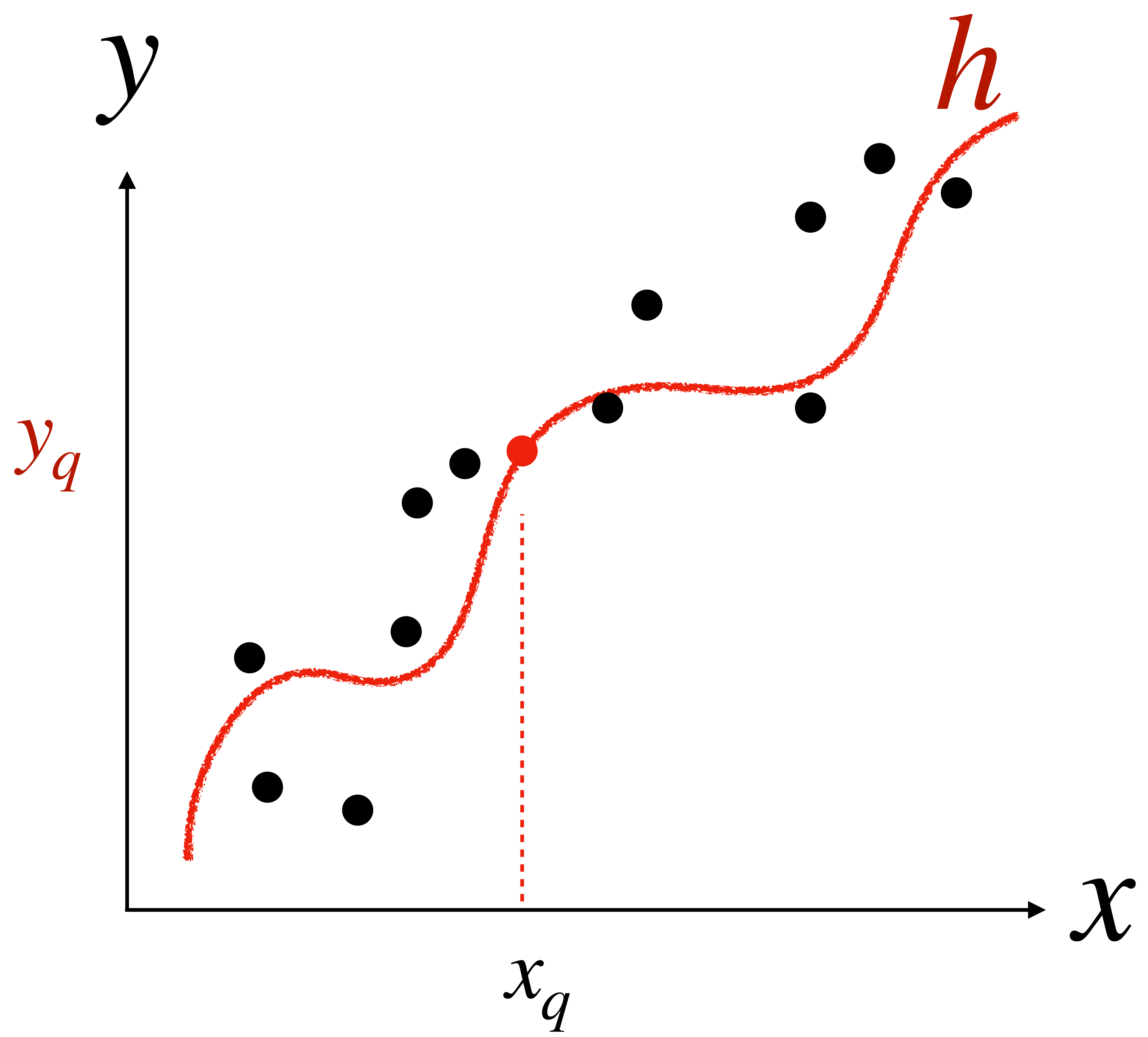


Given new input, what's the output?



Given the data,
find a **function** h, a.k.a **hypothesis,**
that predicts an output, given an input

Some other function?



Given new input, what's the output?

Assume a linear hypothesis

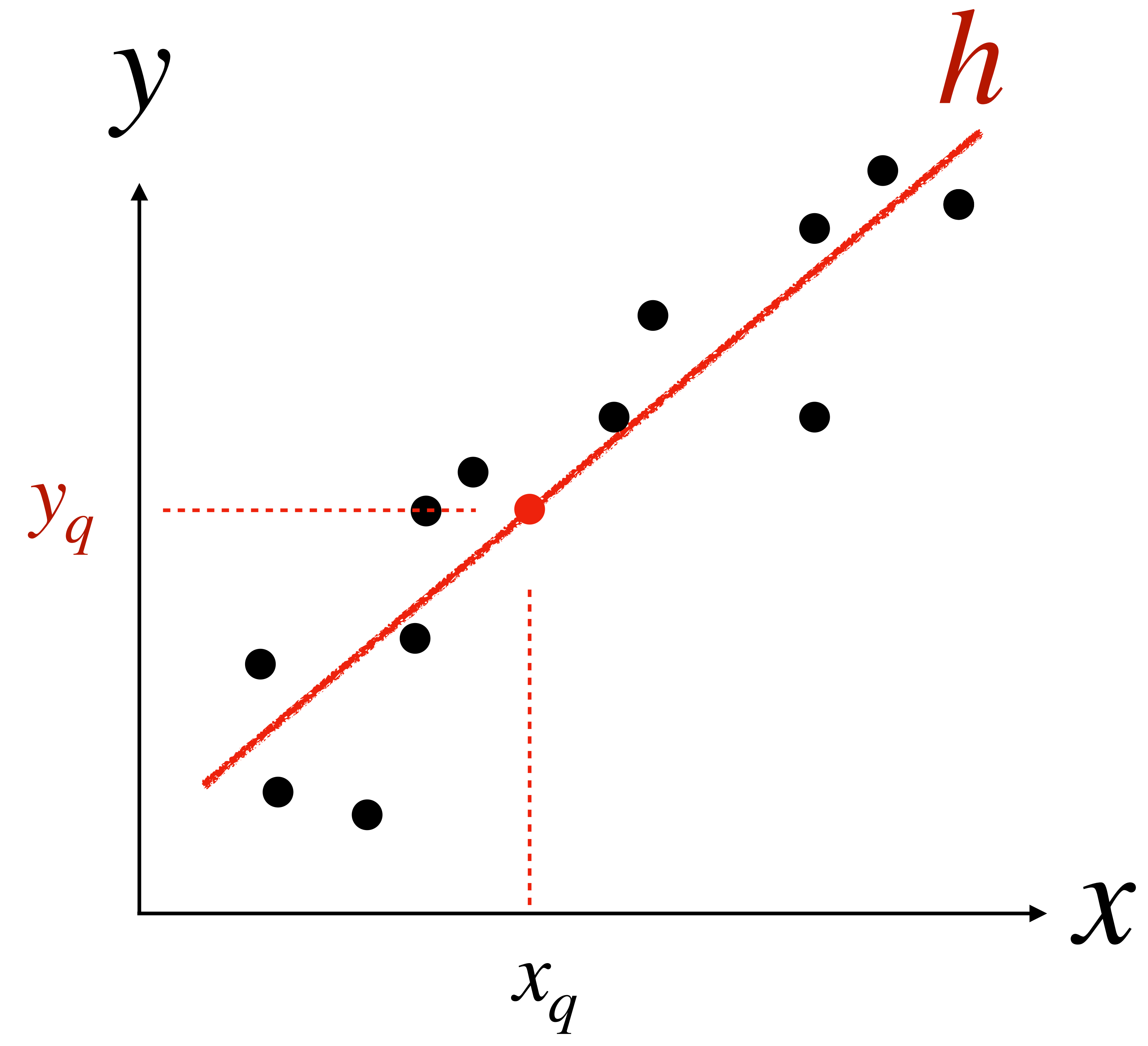
$$x \xrightarrow{h} y$$

$$h(x) = ax + b$$

What are the best a and b that fit the data?

a, b **are fitting parameters**

Linear function



Assume a linear hypothesis

Assume a linear hypothesis

$$x \xrightarrow{h} y$$

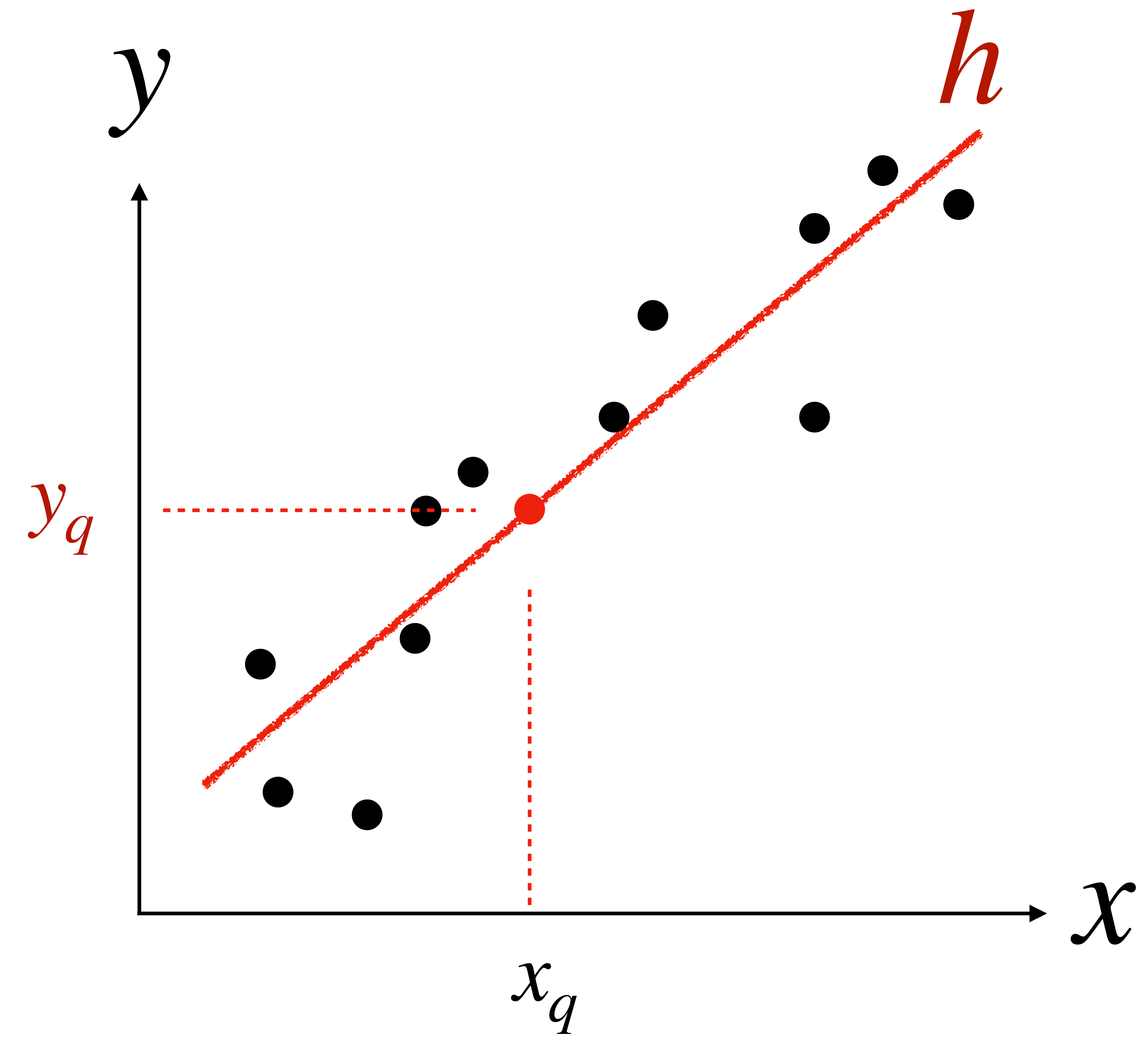
$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$h_\theta(x) = [\theta_0, \theta_1] \cdot [1, x]$$

Unknown parameters | Input features

$$\theta \cdot \mathbf{x}$$

Linear function



What's the best $\theta = [\theta_0, \theta_1]$, given the data ?

What happens if we have more inputs?

Assume a linear hypothesis

$$\mathbf{x} \xrightarrow{h} y$$

$$h : \mathbb{R}^n \to \mathbb{R}$$

$$h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \ldots$$

$$h_\theta(\mathbf{x}) = \underbrace{[\theta_0, \theta_1, \theta_2, \theta_3, \ldots]^\top}_{\theta \text{ weights}} \cdot \underbrace{[1, x_1, x_2, x_3, \ldots]^\top}_{\mathbf{x} \text{ inputs}}$$

$$h_\theta(\mathbf{x}) = \theta \cdot \mathbf{x} = \theta^\top \mathbf{x}$$

Inputs		Output
x_1	x_2	y
$x_1^{(1)}$	$x_2^{(1)}$	$y^{(1)}$
$x_1^{(2)}$	$x_2^{(2)}$	$y^{(2)}$
$x_1^{(3)}$	$x_2^{(3)}$	$y^{(3)}$
$x_1^{(4)}$	$x_2^{(4)}$	$y^{(4)}$
$\vdots$	$\vdots$	$\vdots$

How do we pick the best parameters θ ?

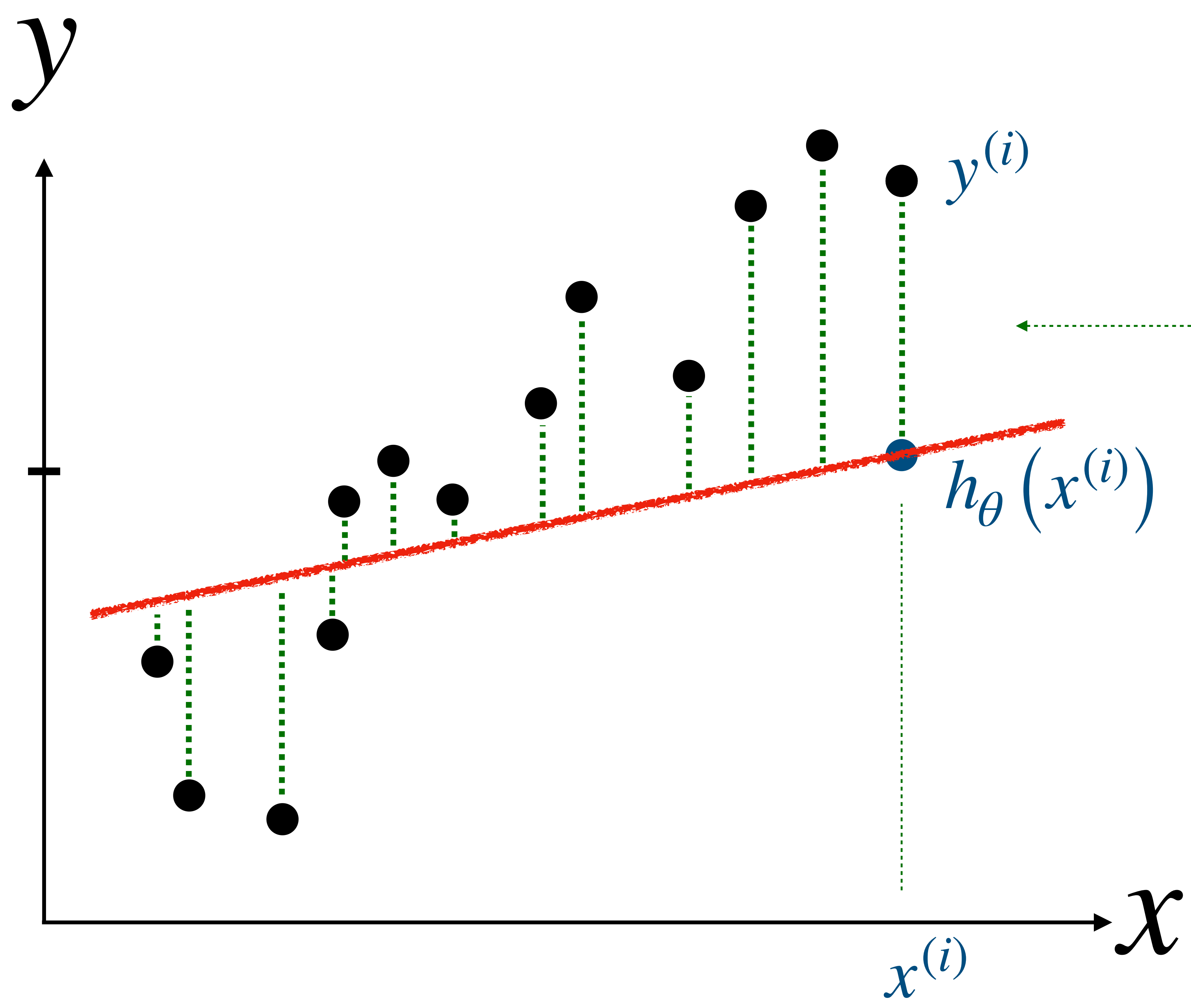
$$h_\theta(\mathbf{x}) = \theta^\top \mathbf{x} = \sum_{i=0}^{d} \theta_i x_i$$

Cost function

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{d} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right)^2$$

$$= \frac{1}{2} \sum_{i=1}^{d} \left(\theta^\top \mathbf{x}^{(i)} - y^{(i)} \right)^2$$

Ordinary least squares



distance $\left(h_\theta\left(x^{(i)}\right), y^{(i)} \right)$

$h_\theta\left(x^{(i)}\right) - y^{(i)}$	Residuals
$\left\lvert h_\theta\left(x^{(i)}\right) - y^{(i)} \right\rvert$	Absolute loss
$\left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right)^2$	Square loss

Interactive Demo

https://colab.research.google.com/drive/1jEMvm_qlLneleOFDC5Andr7JstletVet?usp=sharing

Choose θ to minimize $J(\theta)$

Cost function

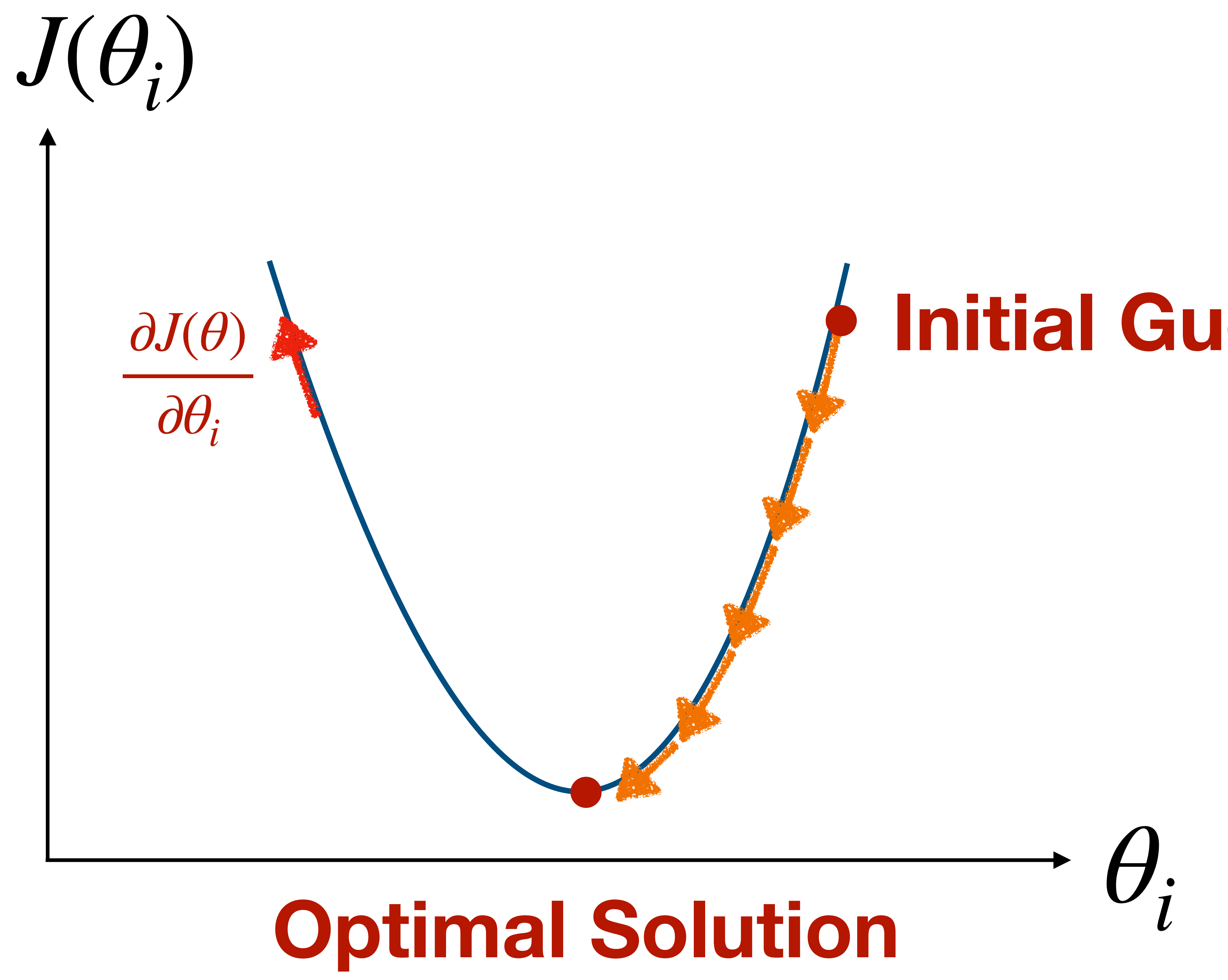
$$J(\theta) = \frac{1}{2} \sum_{i=1}^{d} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right)^2$$

Gradient Descent Update

while not converged:

$$\theta_i := \theta_i - \alpha \frac{\partial J(\theta)}{\partial \theta_i}$$

Learning Rate



Gradient can be computed explicitly

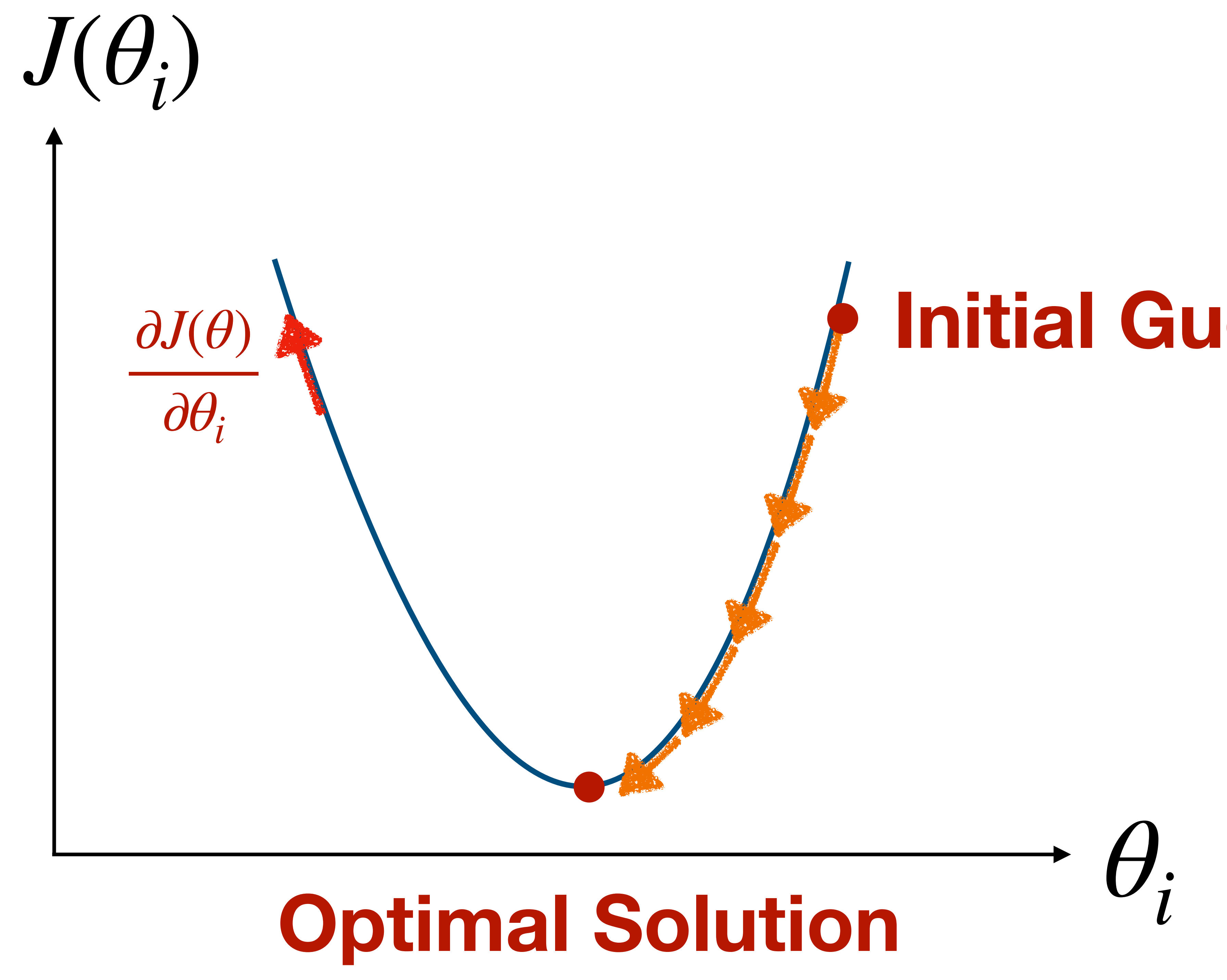
while not converged:

$$\theta_i := \theta_i - \alpha \frac{\partial J(\theta)}{\partial \theta_i}$$

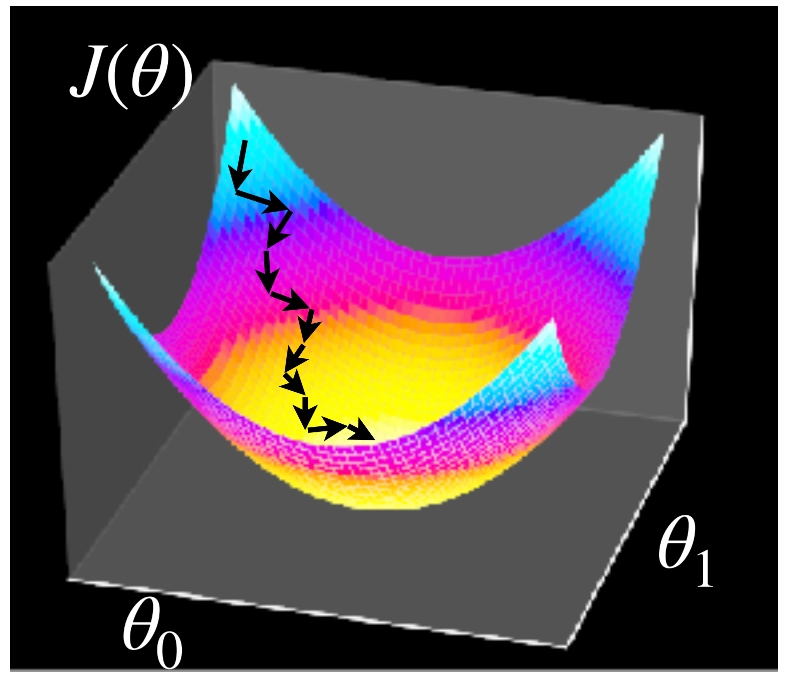
Learning Rate

Derive $\dfrac{\partial J(\theta)}{\partial \theta_i}$ explicitly, for one (x, y) pair

Assume $y = \theta_0 x + \theta_1$



Least Mean Squares (LMS)

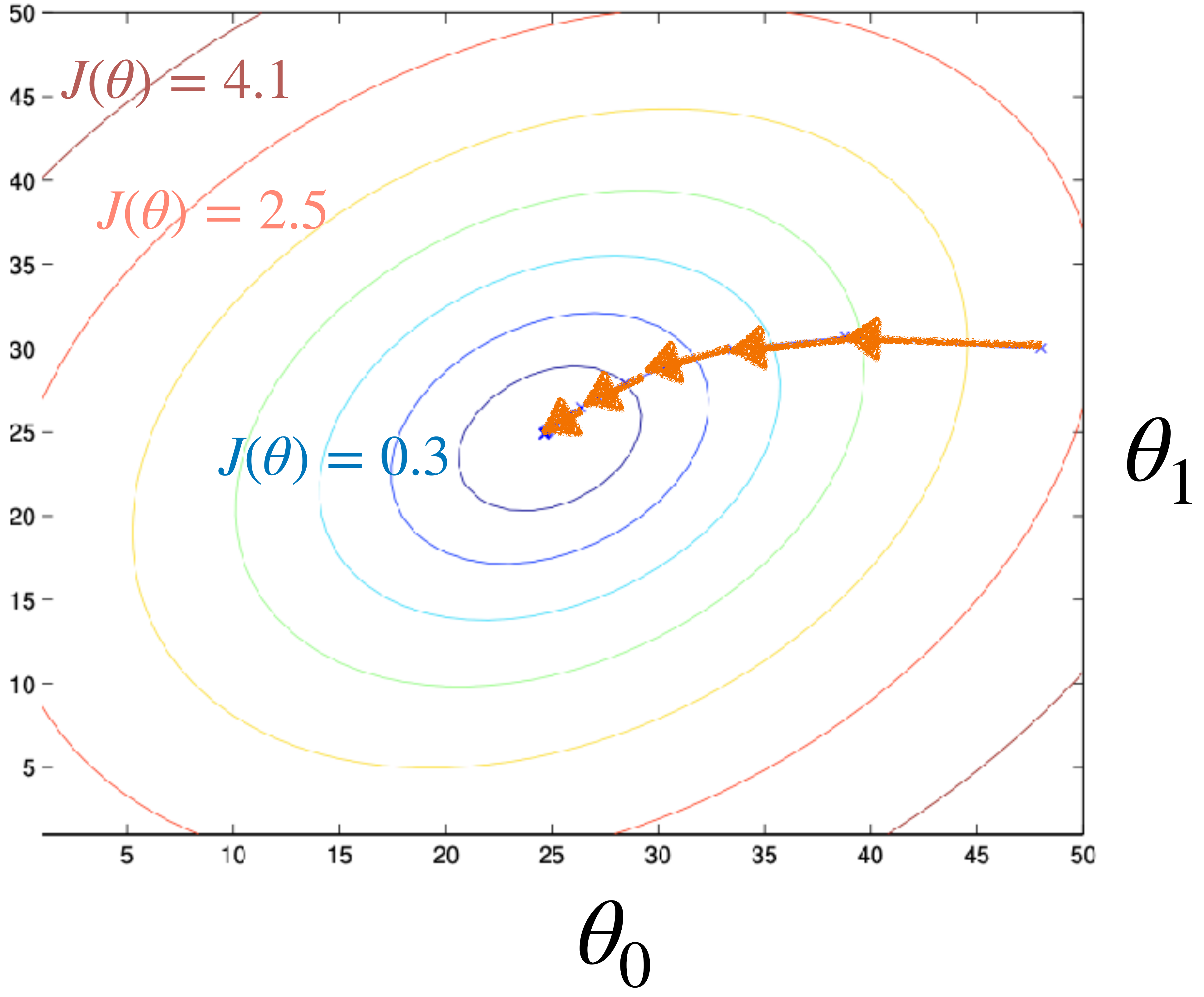


A.K.A **Widrow-Hoff** learning rule

For a single training example $\left(x^{(i)}, y^{(i)}\right)$:

$$\theta_j := \theta_j - \alpha\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)}$$

Contour Plot



Batch Gradient Descent

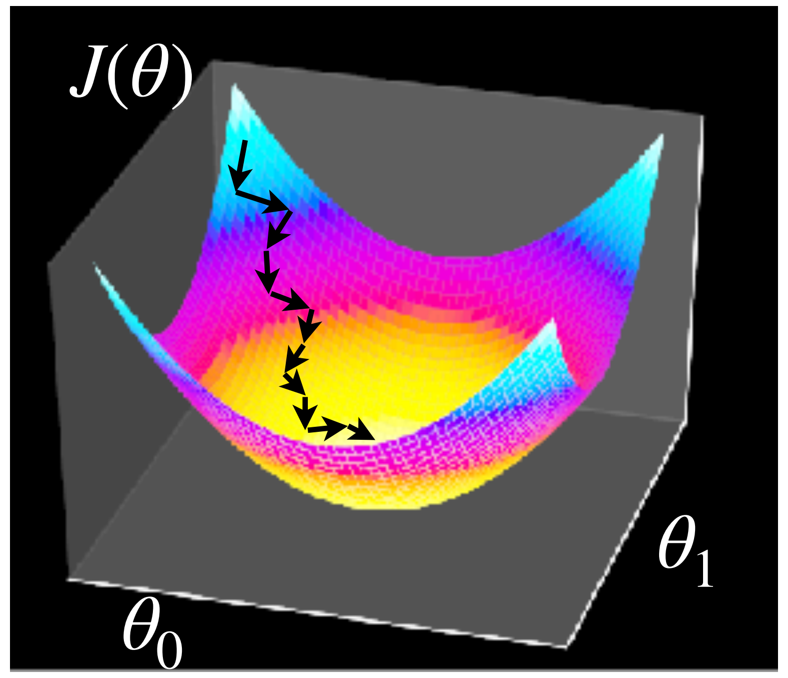
for t = 1...T: (Epochs)

for all parameters j:

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{n} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)}$$

Stack and vectorize

Least Mean Squares (LMS)

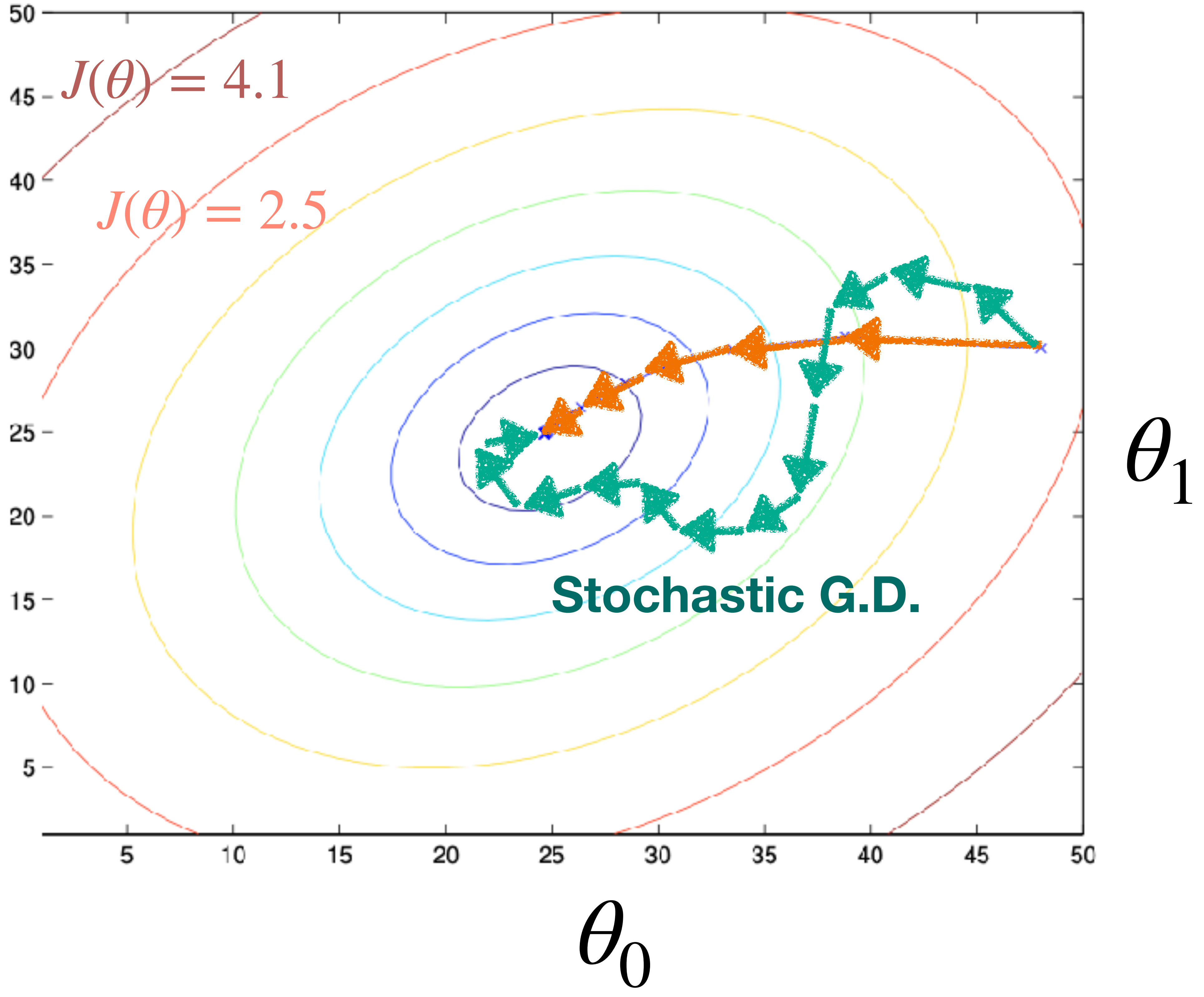


Batch Gradient Descent (vectorized)

for t = 1…T:

$$\theta := \theta - \alpha \sum_{i=1}^{n} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) x^{(i)}$$

Contour Plot



Stochastic Gradient Descent

for t = 1…T:

  for i = 1…n:

$$\theta := \theta - \alpha \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) x^{(i)}$$

Summary

1. Assume a linear hypothesis

$$h_\theta(\mathbf{x}) = \theta^\top \mathbf{x} = \sum_{i=0}^{d} \theta_i x_i$$

2. Cost function

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{d} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right)^2$$

3. Minimize: Gradient Descent

$$\theta := \theta - \alpha \nabla_\theta J(\theta)$$

SGD

for t = 1…T:

  for i = 1…n:

$$\theta := \theta - \alpha \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right) x^{(i)}$$

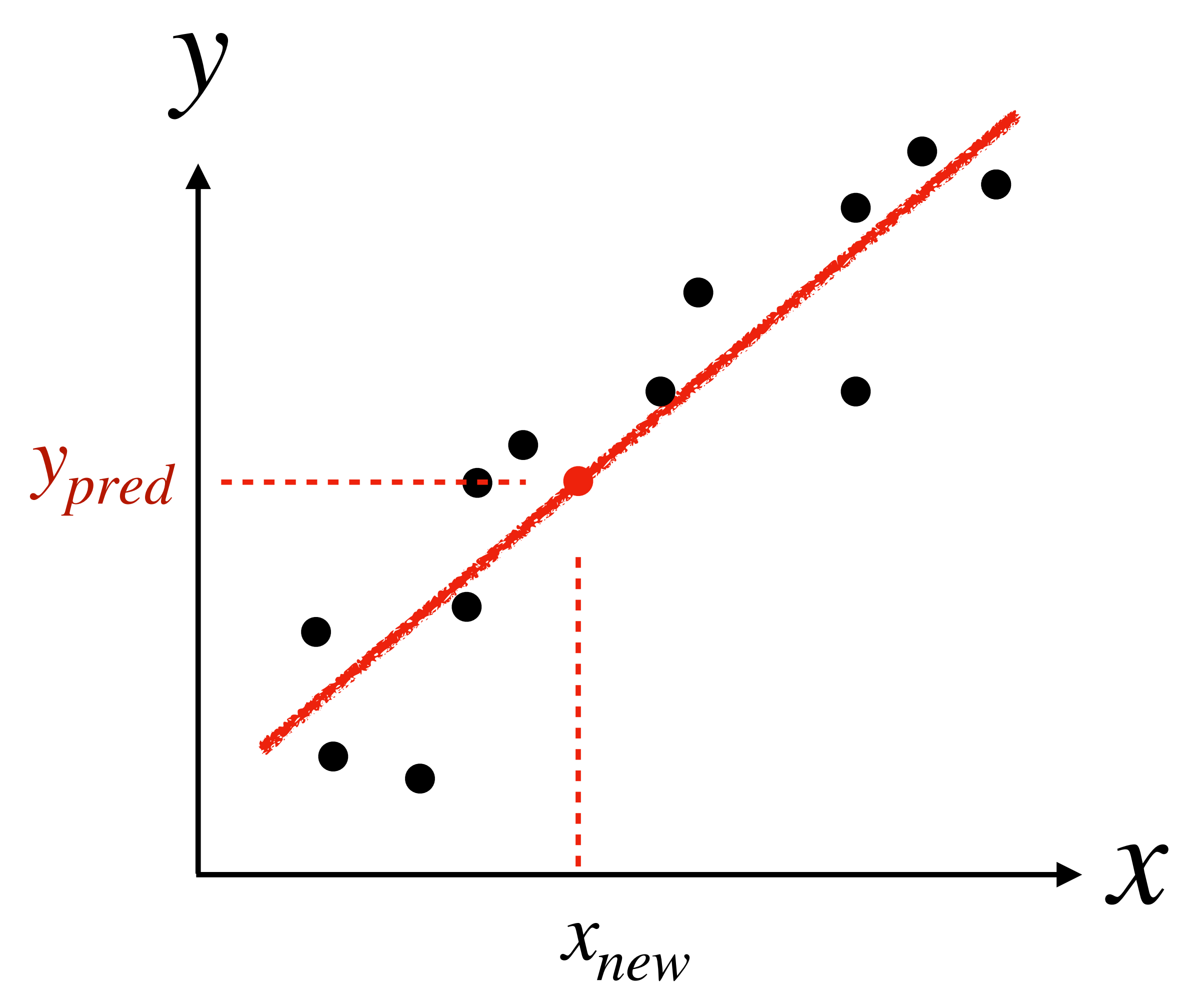
$\hat{\theta}$

4. Optimal predictor

$$y = h_{\hat{\theta}}(x)$$

5. Predict unseen data

$$y_{pred} = h_{\hat{\theta}}(x_{new})$$



Can you find the minimum analytically?

Design matrix

$$X = \begin{bmatrix} -- & {x^{(1)}}^\top & -- \\ -- & {x^{(2)}}^\top & -- \\ -- & \vdots & -- \\ -- & {x^{(n)}}^\top & -- \end{bmatrix}$$

Parameters

$$\vec{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_m \end{bmatrix}$$

Output

$$\vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

Minimize

$$J(\theta) = \frac{1}{2} \| X\theta - \vec{y} \|_2^2 \longrightarrow \nabla_\theta J(\theta) = 0$$

Normal Equation

$$\theta = \left(X^\top X\right)^{-1} X^\top \vec{y}$$

Feature Engineering

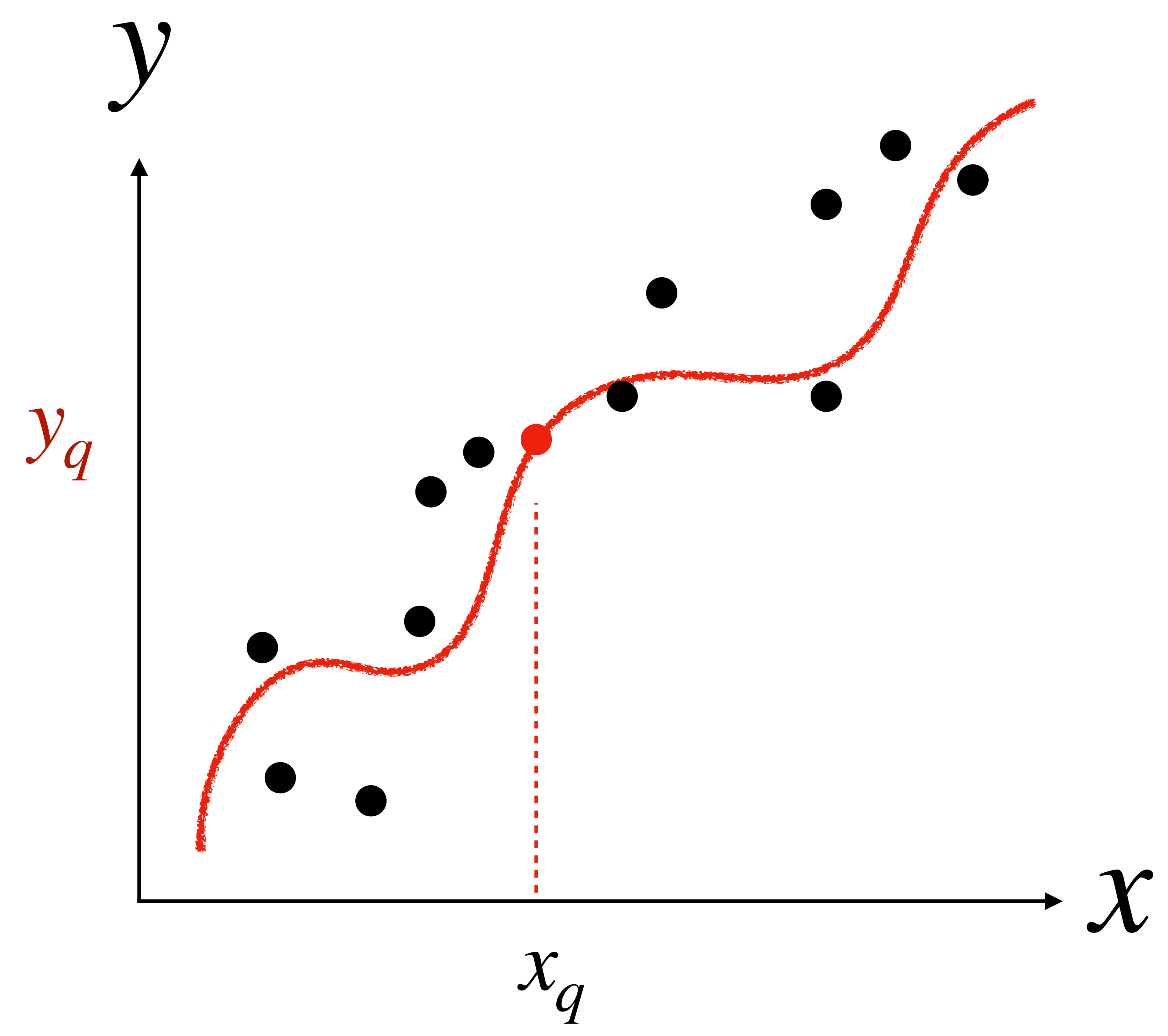
h **is does not have to be linear in** x

Input x	Output y
$x^{(1)}$	$y^{(1)}$
$x^{(2)}$	$y^{(2)}$
$x^{(3)}$	$y^{(3)}$
$x^{(4)}$	$y^{(4)}$

Example: construct a polynomial model

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \dots$$

Some other function?



$$h_\theta(x) = \underbrace{[\theta_0, \theta_1, \theta_2, \theta_3, \dots]}_{\theta} \cdot \underbrace{[1, x, x^2, x^3, \dots]}_{\phi(x)}$$

Feature map

$$h_\theta(x) = \theta^\top \phi(x)$$

Feature Engineering

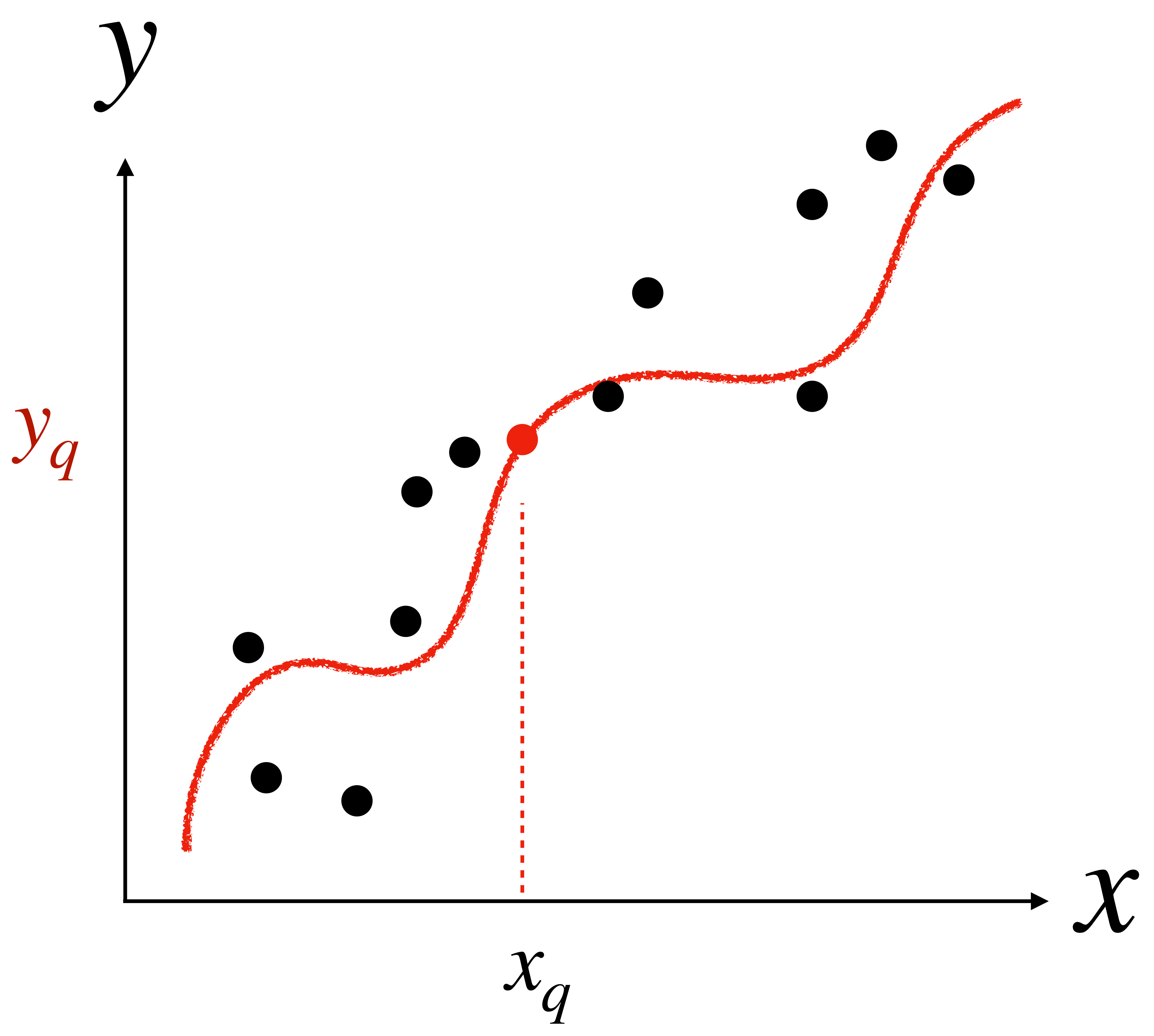
h **is does not have to be linear in** x

Input	Output
x	y
$x^{(1)}$	$y^{(1)}$
$x^{(2)}$	$y^{(2)}$
$x^{(3)}$	$y^{(3)}$
$x^{(4)}$	$y^{(4)}$

Example: construct a polynomial model

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \ldots$$

Some other function?



$$h_\theta(x) = \underbrace{[\theta_0, \theta_1, \theta_2, \theta_3, \ldots]}_{\theta} \cdot \underbrace{[1, x, x^2, x^3, \ldots]}_{\phi(x)}$$

Feature map

$$h_\theta(x) = \theta^\top \phi(x) = \theta_0 \phi_0(x) + \theta_1 \phi_1(x) + \theta_2 \phi_2(x) + \ldots$$

Feature Engineering

A feature map can also **drop features**

Example: construct a polynomial model

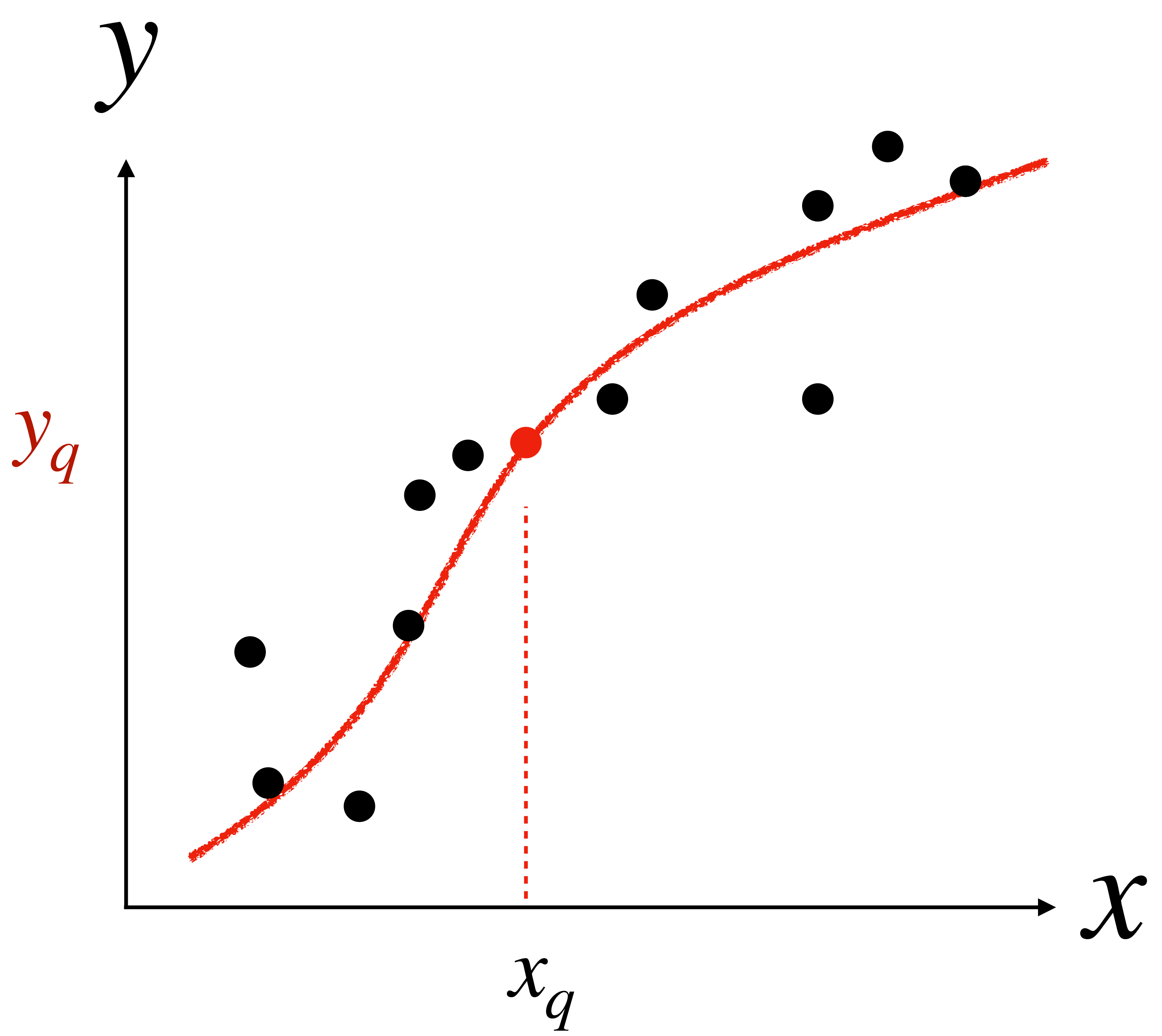
$$h_\theta(x) = \theta_1 x + \theta_3 x^3$$

$$h_\theta(x) = [\theta_1, \theta_3] \cdot \underbrace{[x,\ x^3]}_{\phi(x)}$$

Feature map

$$h_\theta(x) = \theta^\top \phi(x)$$

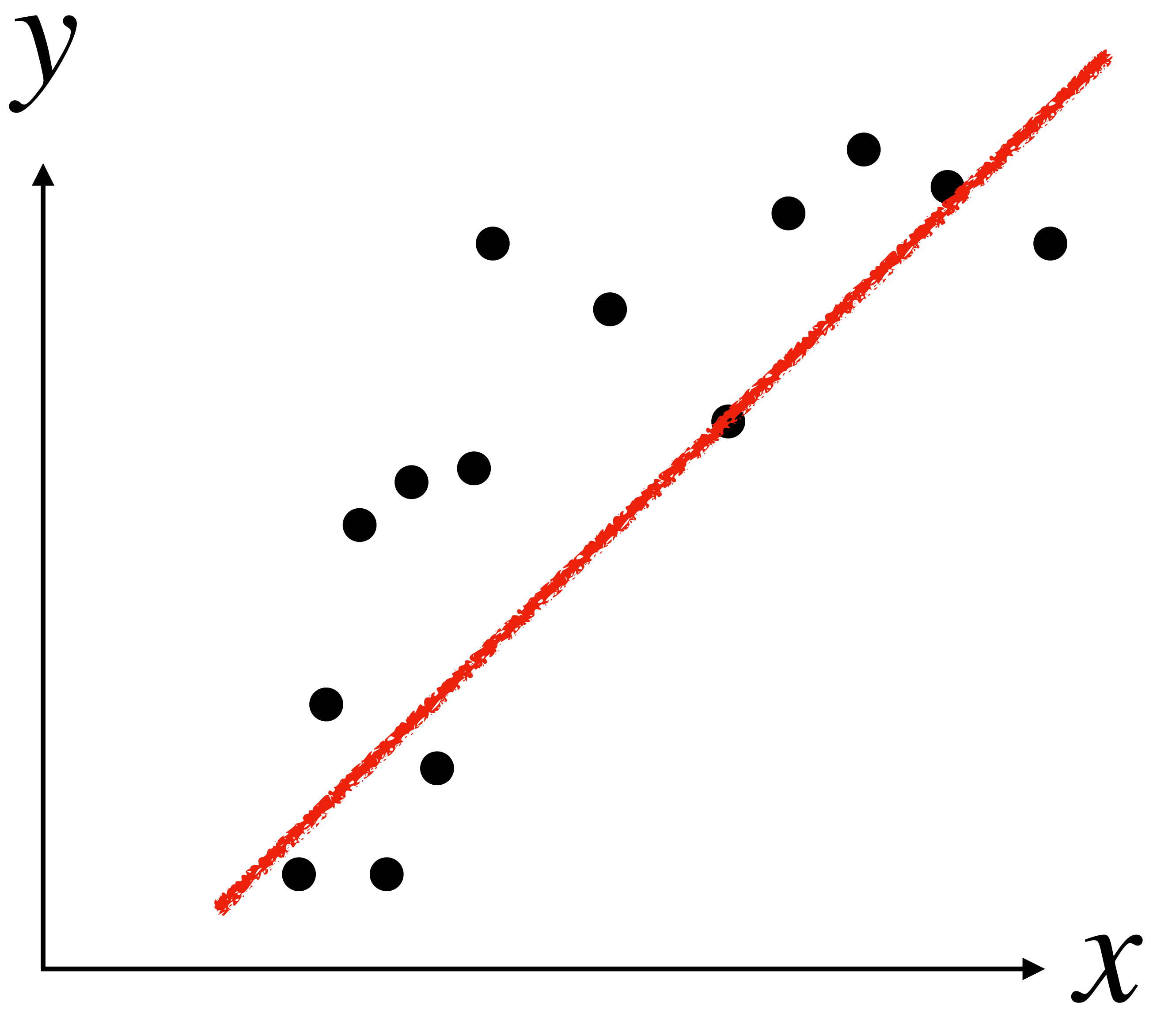
Some other function?



How to choose $\phi(x)$?

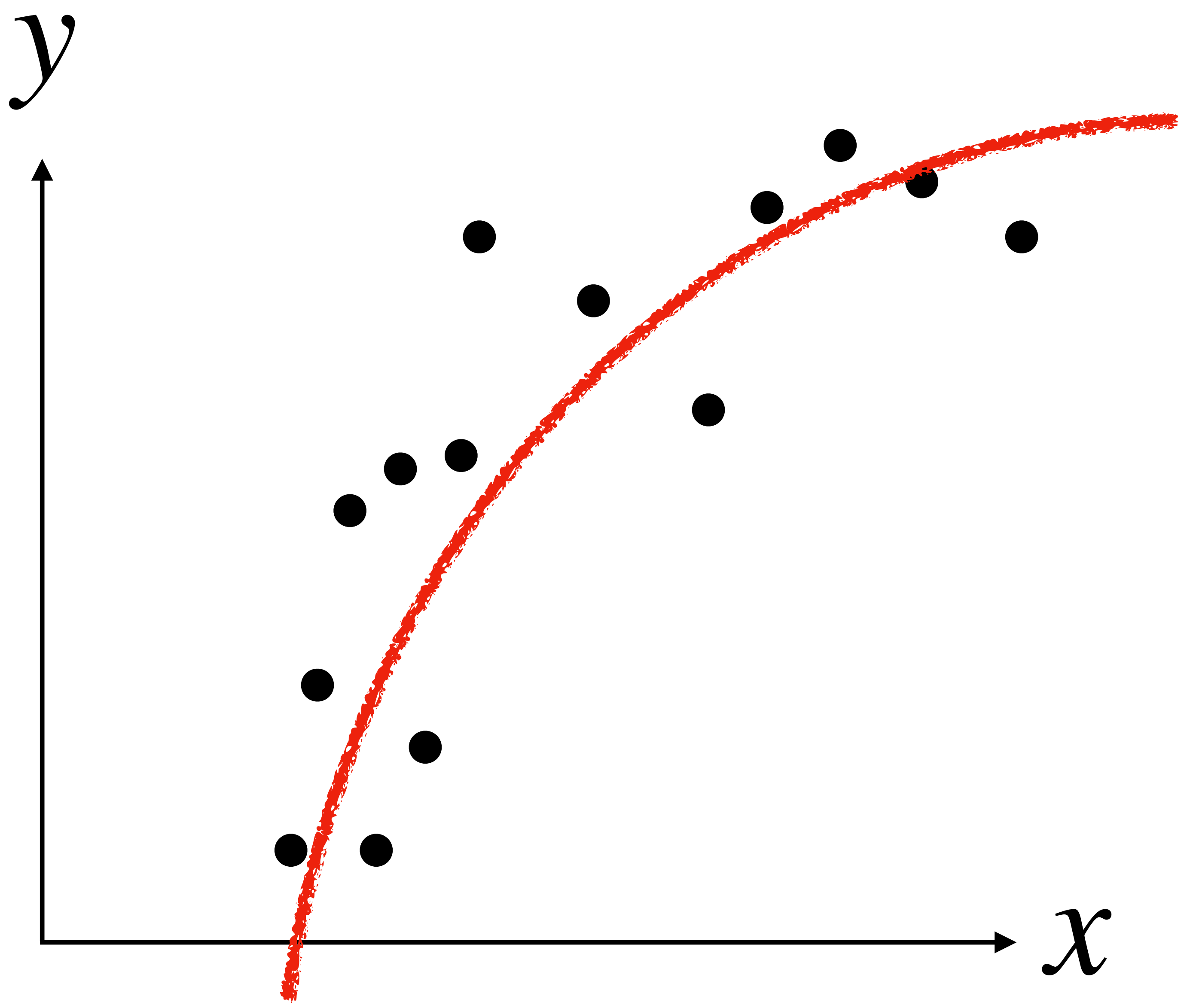
How to optimize over $\phi(x)$

Underfitting
High Bias



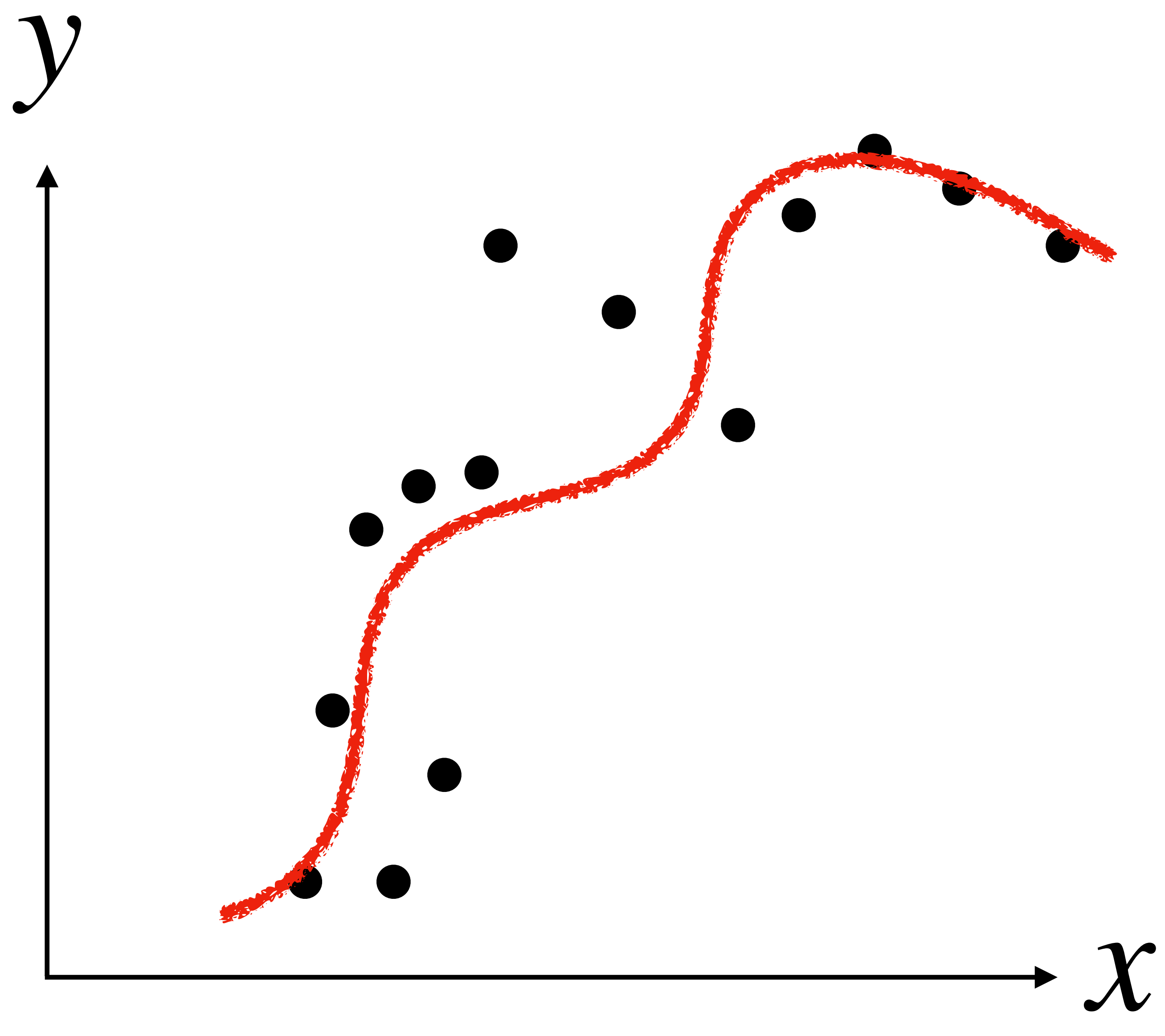
$$\phi(x) = [1, x]$$

Just right



$$\phi(x) = [1, x, x^2]$$

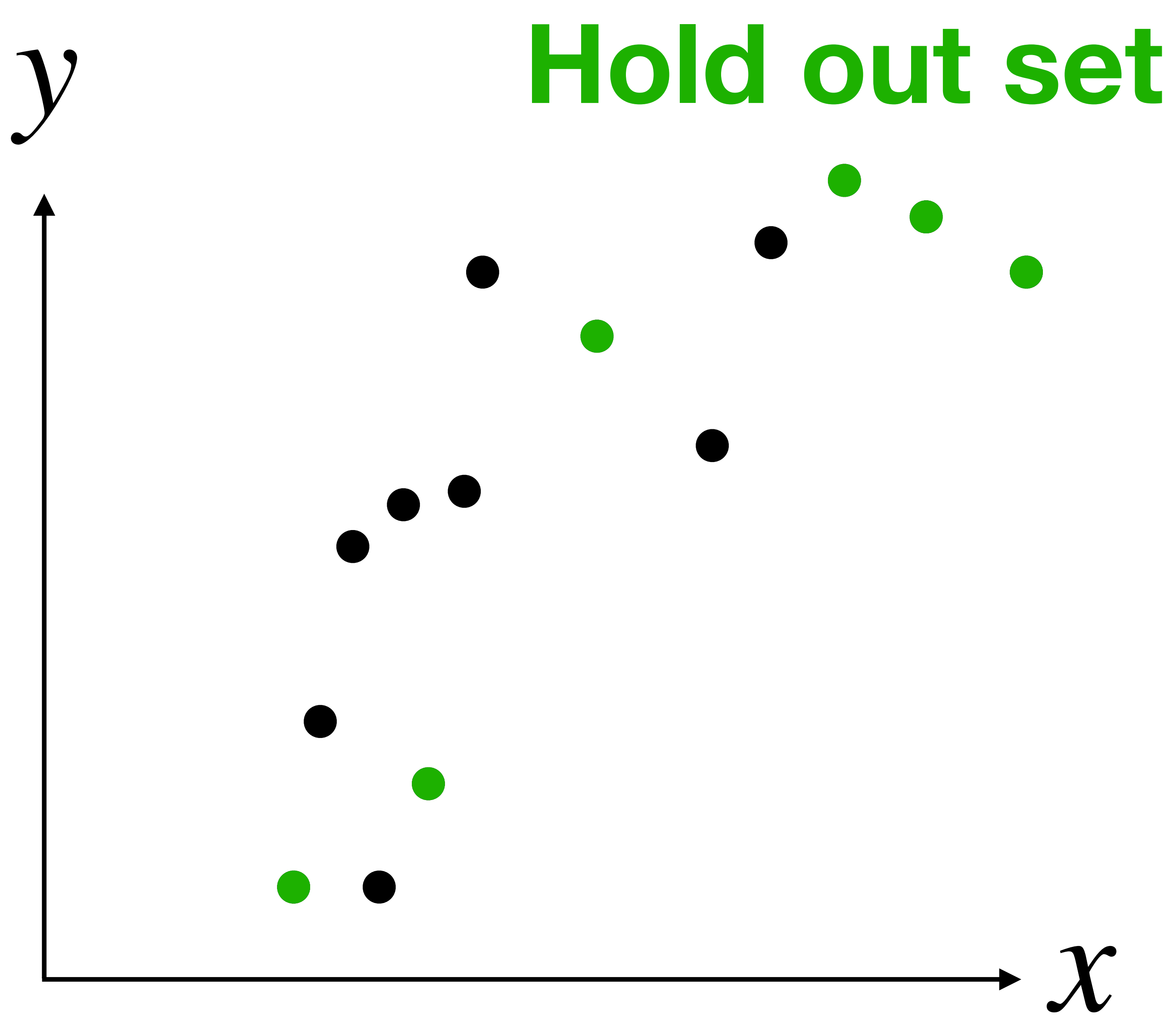
Overfitting
High Variance



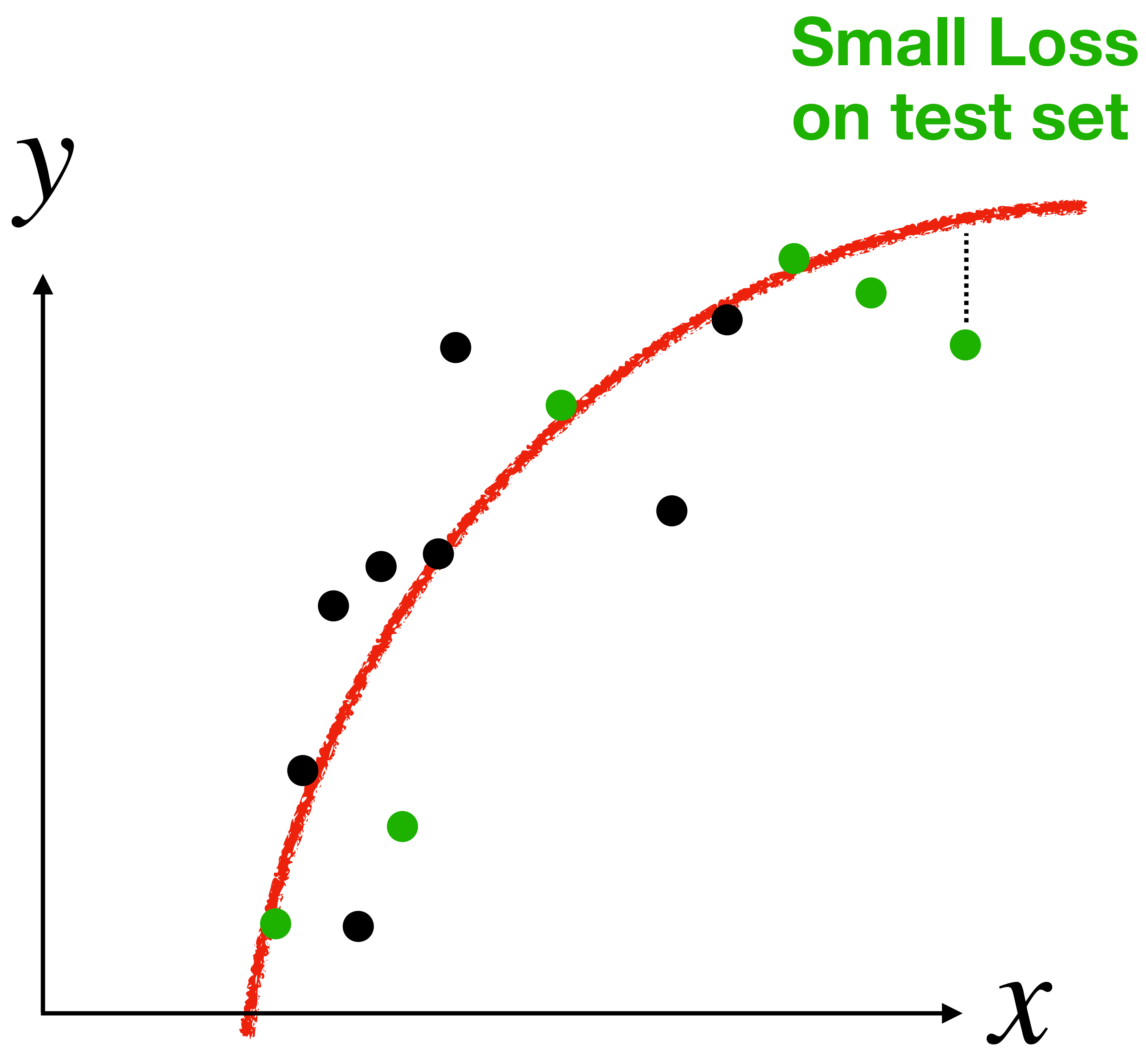
$$\phi(x) = [1, x, x^2, x^3, \ldots]$$

How can we tell if $\phi(\,\cdot\,)$ is good?

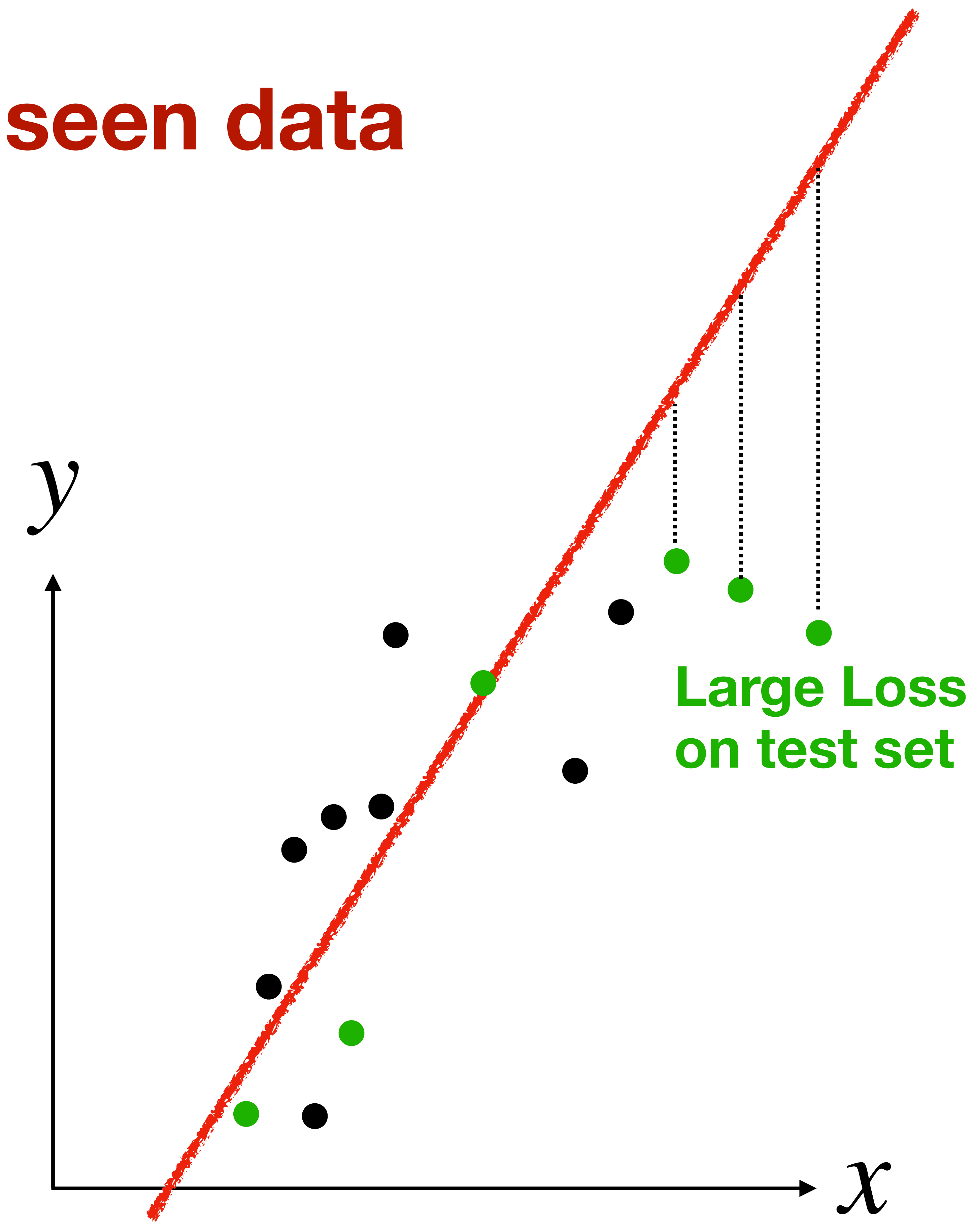
The purpose of Machine Learning is to Generalize to unseen data



Create a test set to evaluate model



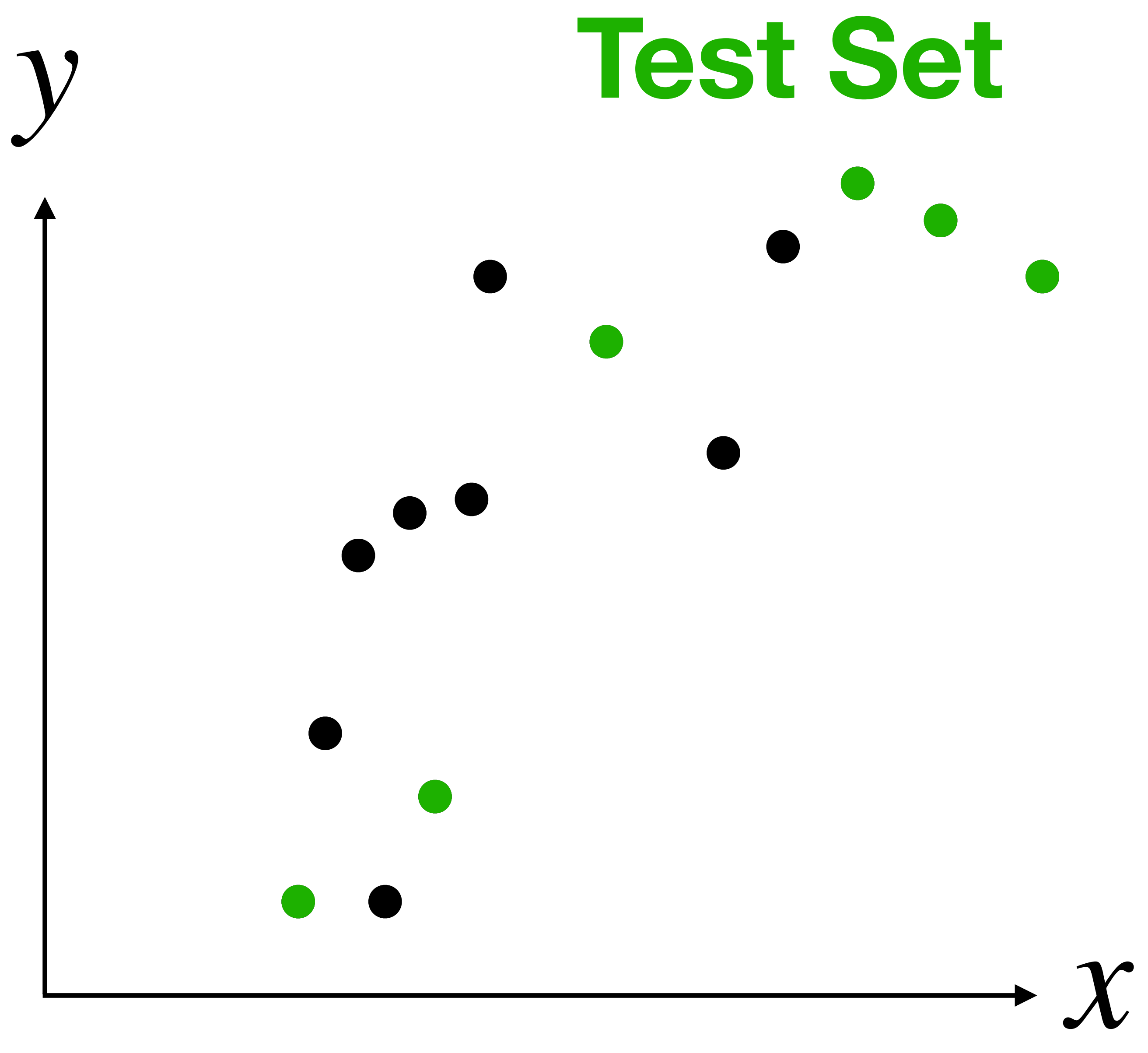
$$\phi(x) = [1, x, x^2]$$



$$\phi(x) = [1, x]$$

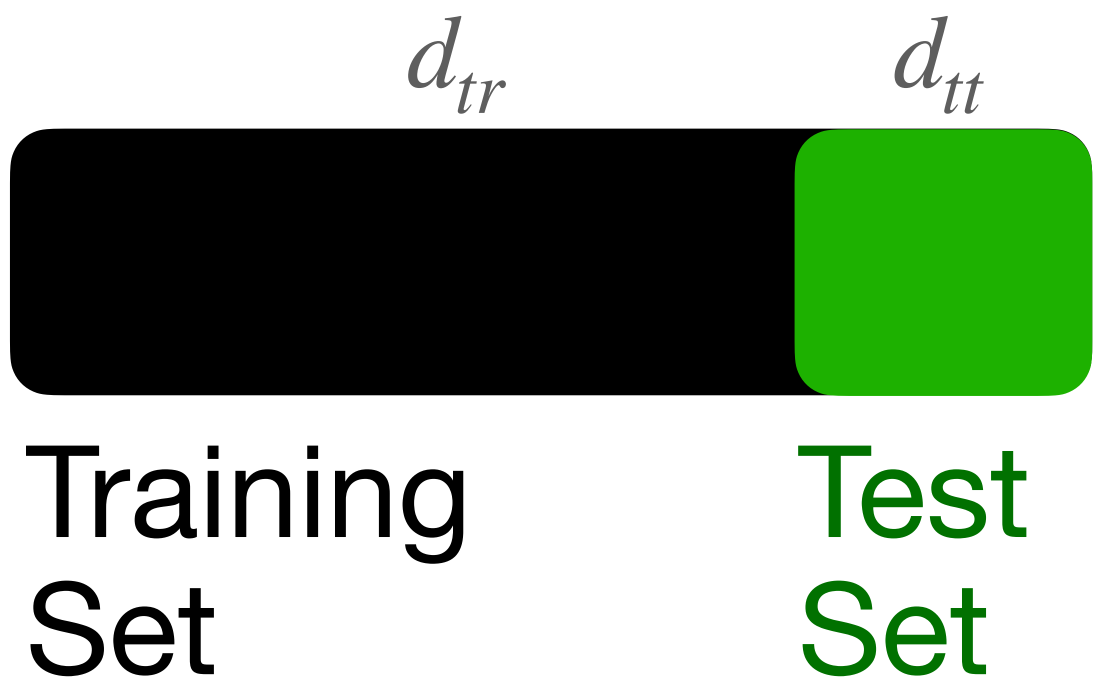
How do we tell that $\phi(\cdot)$ is good?

Define objective functions for each subset



Create a Test set to evaluate model

Split data:

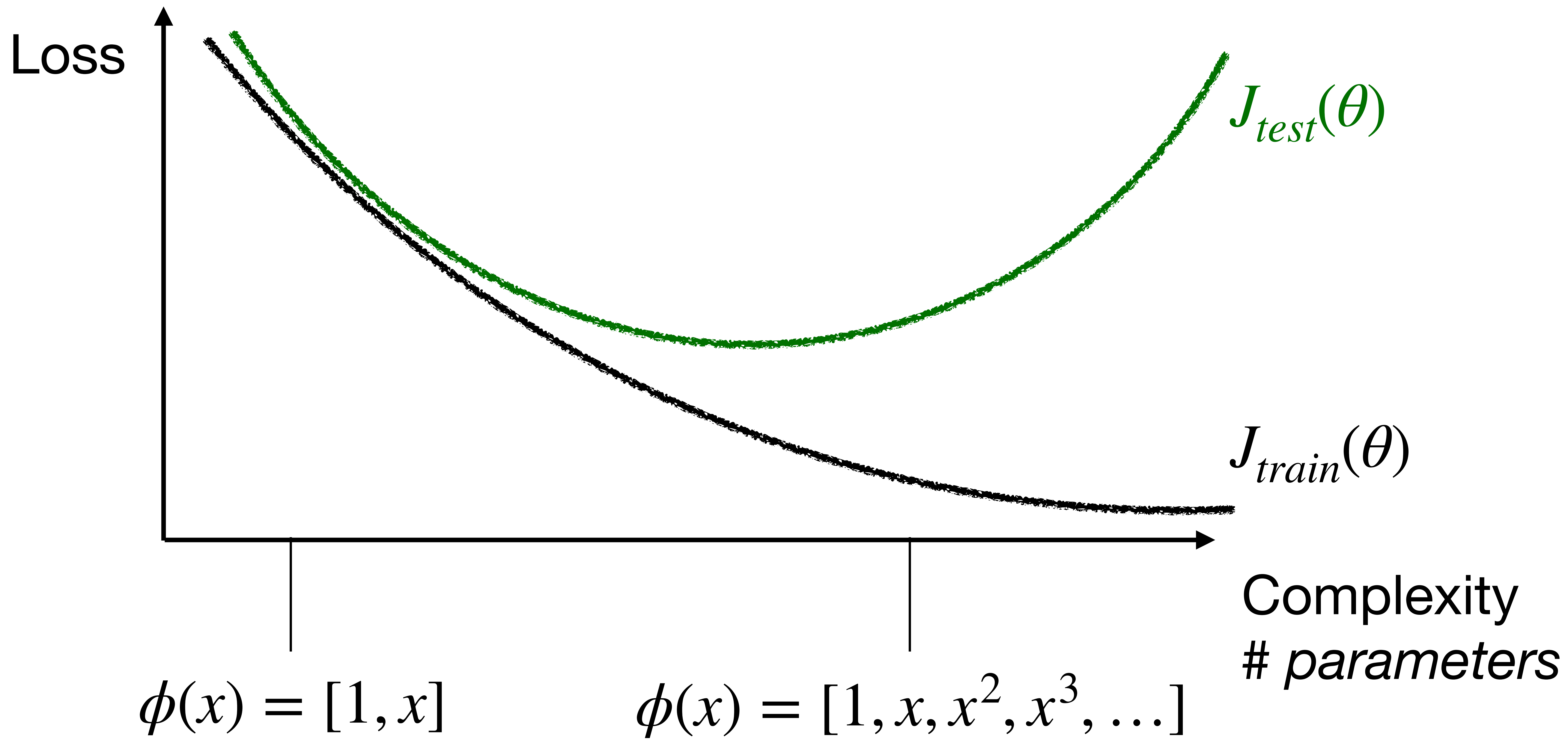


$$J_{train}(\theta) = \frac{1}{2d_{tr}} \sum_{i=1}^{d_{tr}} \left(\theta^\top \phi(x^{(i)}) - y^{(i)}\right)^2$$

$$J_{test}(\theta) = \frac{1}{2d_{tt}} \sum_{i=1}^{d_{tt}} \left(\theta^\top \phi(x^{(i)}) - y^{(i)}\right)^2$$

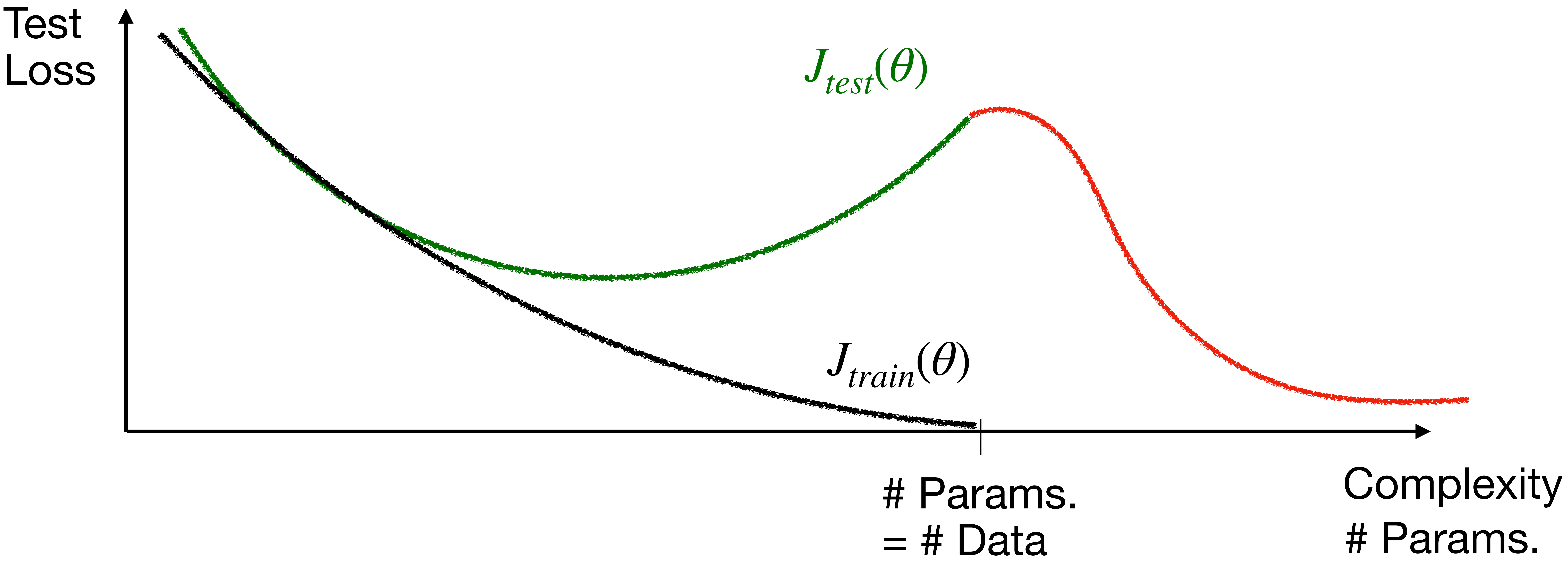
Variance Bias Trade-off

Error as a function of complexity



Double Descent

Model-wise



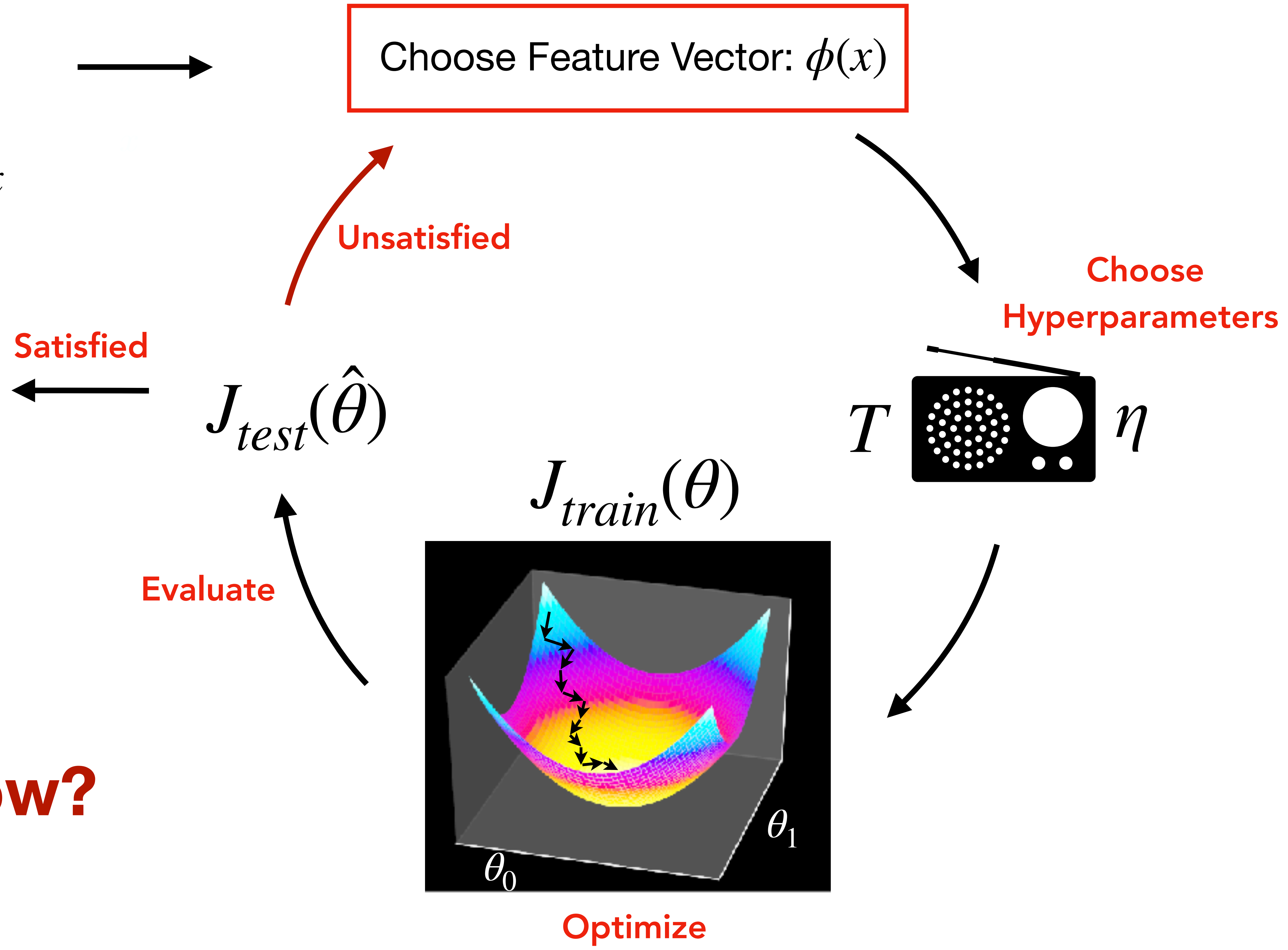
Other Hyperparameters

ϕ is not the only unknown parameter over which we want to optimize

- T: Number of Epochs
- η: Step size
- ϕ: Feature vector

Optimize over ϕ and other hyperparameters

Pick $\hat{\theta}$ with best J_{test}

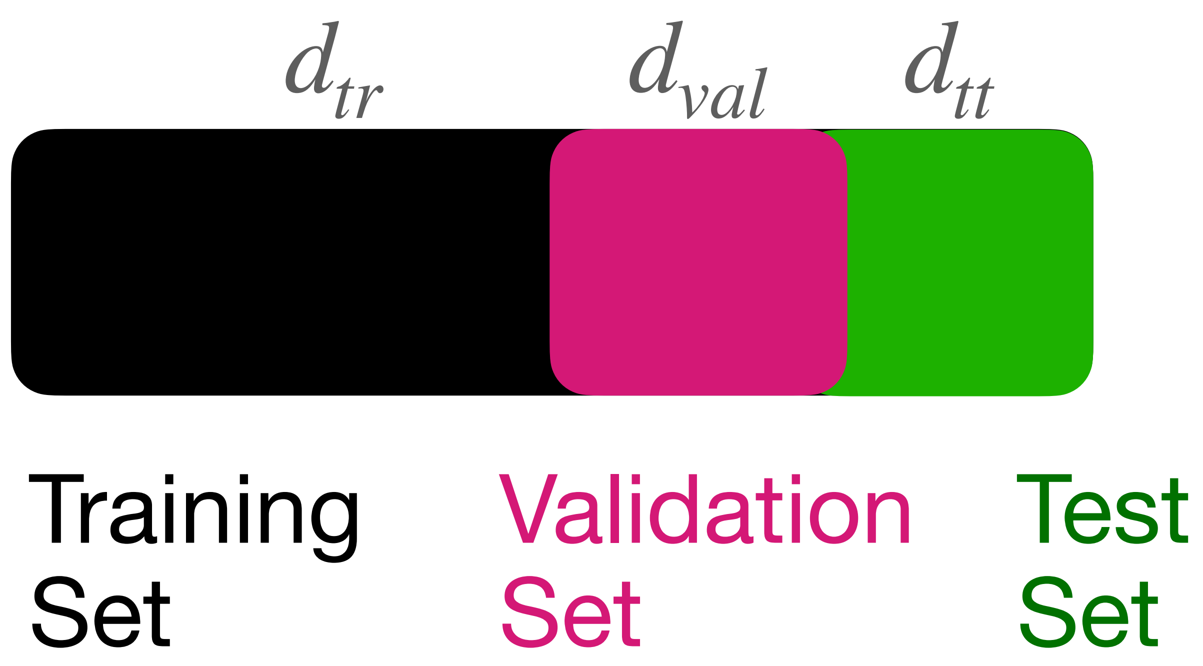


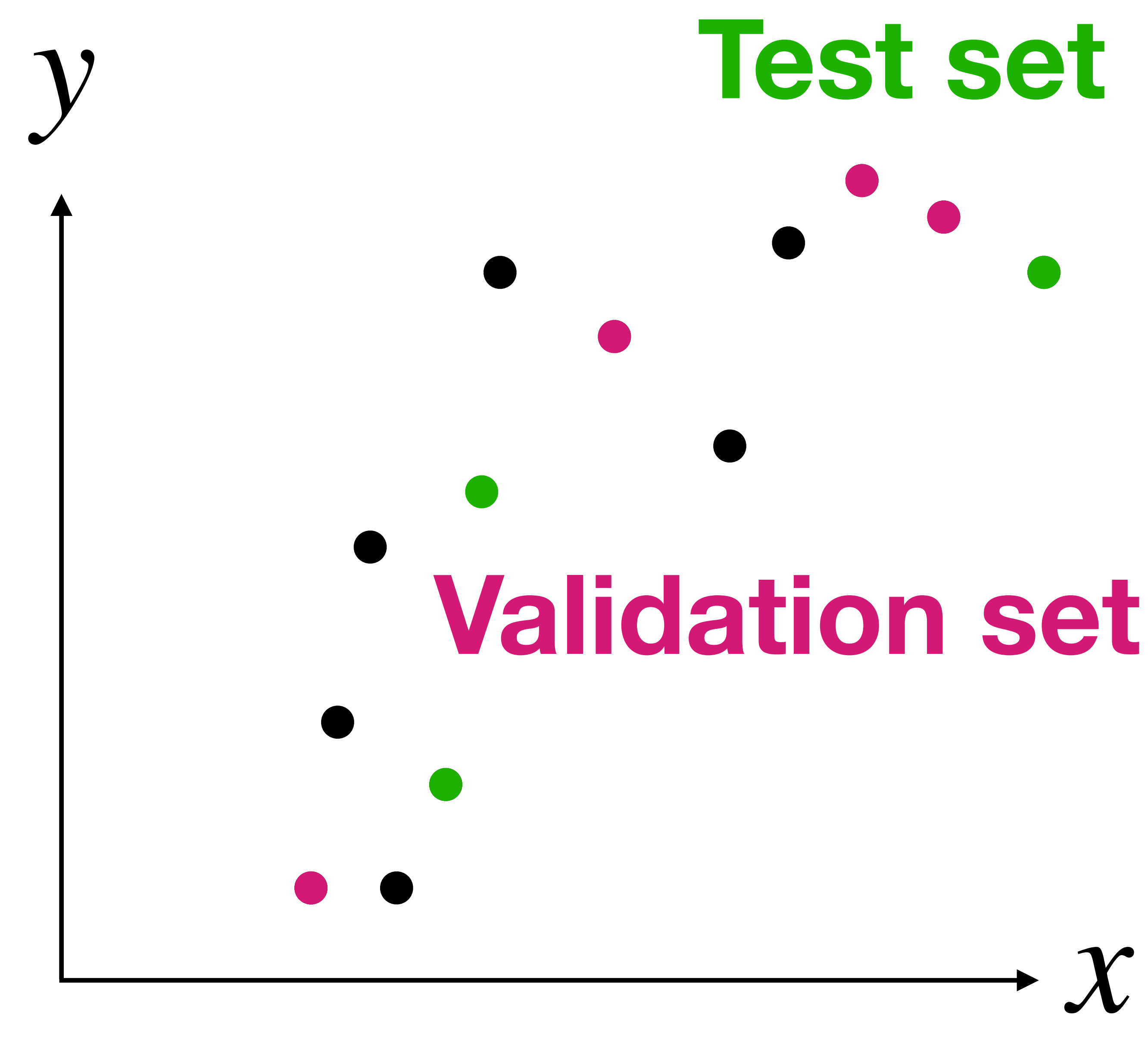
What's the problem with this workflow?

How do we tell that $\phi(\,\cdot\,)$ is good?

Define objective functions for each subset

Split data:





Test set: evaluate model **at the end** of hyperparameter optimization

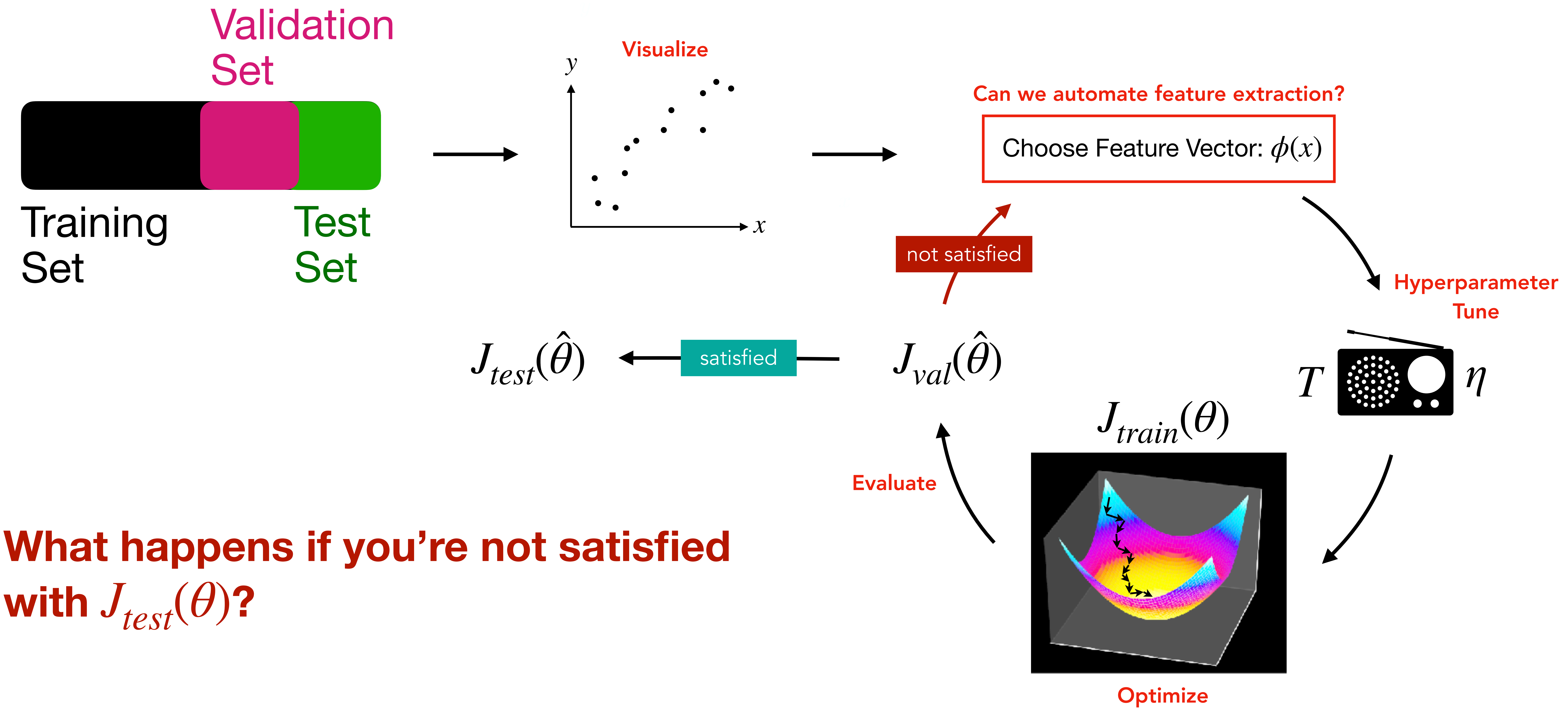
Validation set: evaluation model **during** hyperparameter optimization

$$J_{train}(\theta) = \frac{1}{2d_{tr}} \sum_{i=1}^{d_{tr}} \left(\theta^\top \phi(x^{(i)}) - y^{(i)}\right)^2$$

$$J_{val}(\theta) = \frac{1}{2d_{val}} \sum_{i=1}^{d_{val}} \left(\theta^\top \phi(x^{(i)}) - y^{(i)}\right)^2$$

$$J_{test}(\theta) = \frac{1}{2d_{tt}} \sum_{i=1}^{d_{tt}} \left(\theta^\top \phi(x^{(i)}) - y^{(i)}\right)^2$$

Machine Learning workflow - Cross Validation



What happens if you're not satisfied with $J_{test}(\theta)$?

Remedies to Overfitting

Practical tips to decrease overfitting

- Make the model simpler if it's overfitting, and more complex if it's underfitting
  - **Recursive Feature Elimination:** start with all features and drop them one by one while tracking the loss
  - Get rid of features that you think are irrelevant in predicting the desired output
- Add a regularization term that makes the hypothesis class smaller

$$J_{reg}(\theta) = J(\theta) + \lambda\, R(\theta)$$

Regularization

Force fitting parameters to be smaller - ‘shrink’ hypothesis class

$$h_\theta(x) = 100.2 + 50.6x + 70.4x^2 + 1345x^3 + 200.3x^4$$

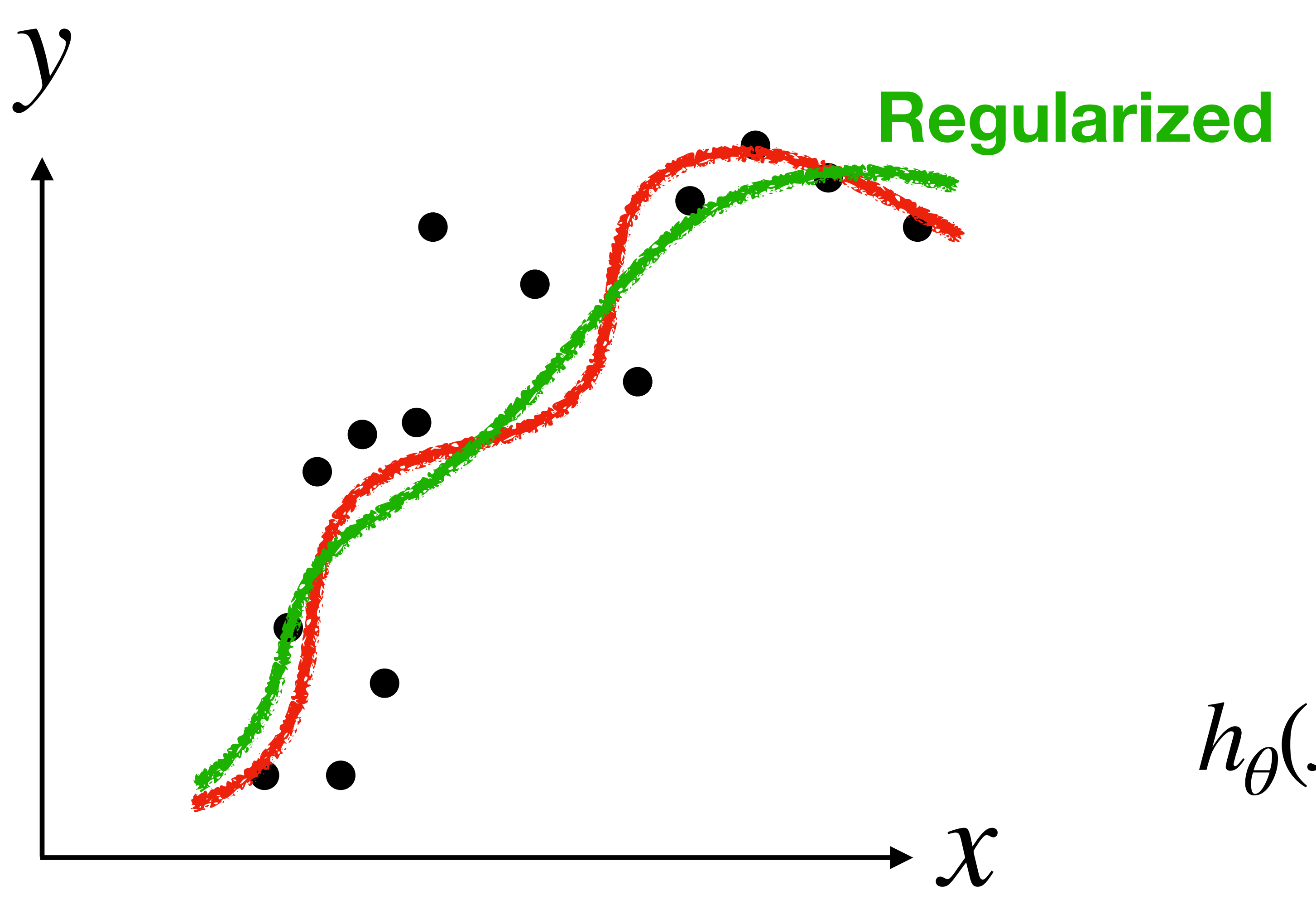
$$J_{reg}(\theta) = J(\theta) + \lambda\, R(\theta)$$

L1 Regularization

$$R(\theta) = \|\theta\|_1$$

$$h_\theta(x) = 5.1x + 7.2x^2 + 3.3x^4$$

Less coefficients



L2 Regularization

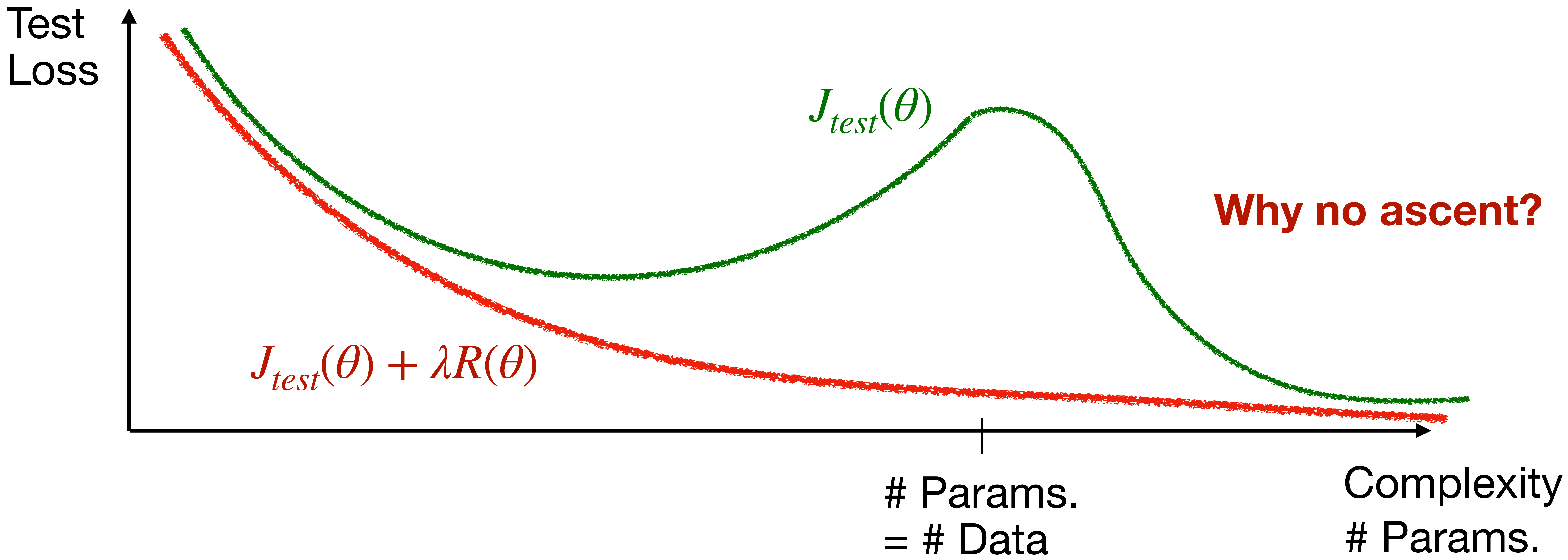
$$R(\theta) = \|\theta\|_2$$

$$h_\theta(x) = .1 + 5.2x + 7.4x^2 + .05x^3 + 2.3x^4$$

Smaller coefficients

Double Descent

Regularization solves the problem with large parameters

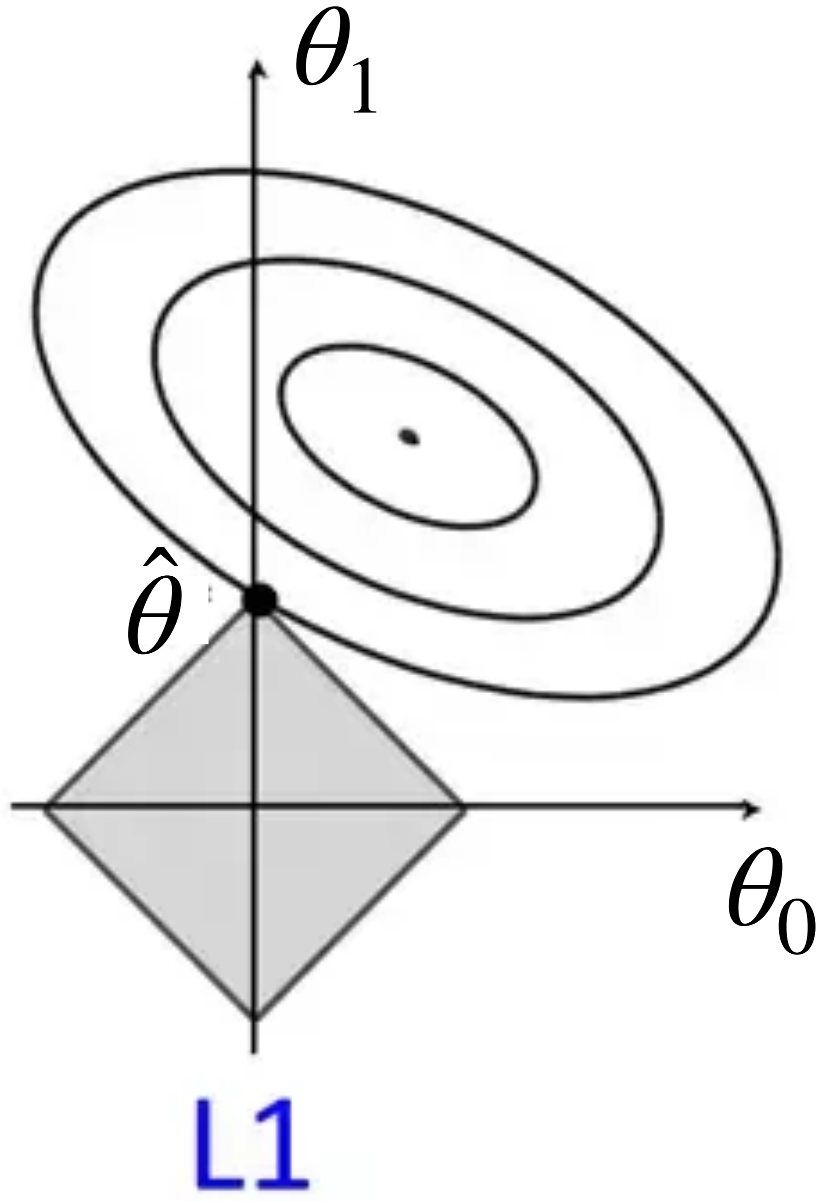


Regularization

Force fitting parameters to be smaller - ‘shrink’ hypothesis class

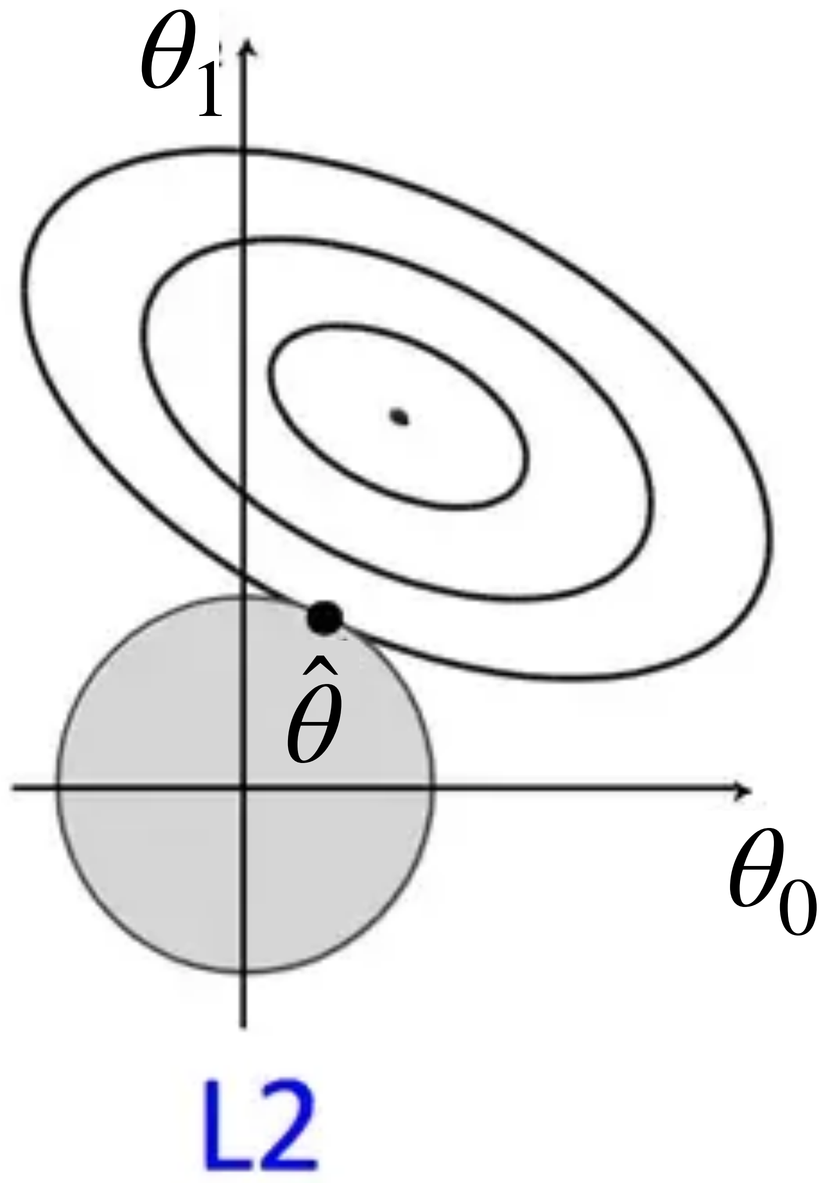
L1 Regularization

$$R(\theta) = ||\theta||_1$$



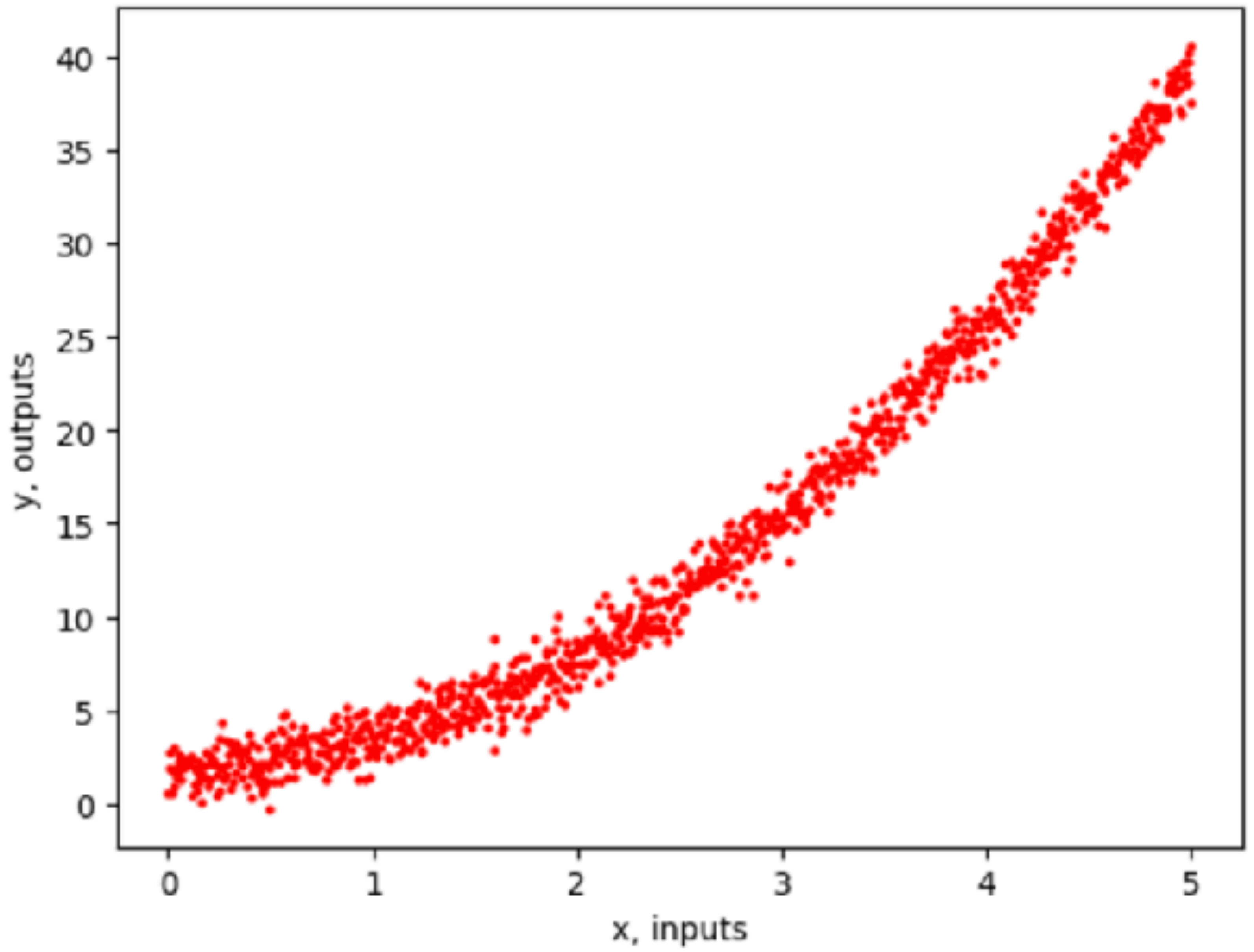
L2 Regularization

$$R(\theta) = ||\theta||_2$$



Create synthetic data

```python
# synthetic parameters
num_points = 1000
var = 1
a = 1.5
b = 2

# generate data
x = np.linspace(0, 5, num_points)
y = 1.5 * x**2 + b + var * np.random.normal(0, 1, num_points)
```



Feature engineering (design matrix)

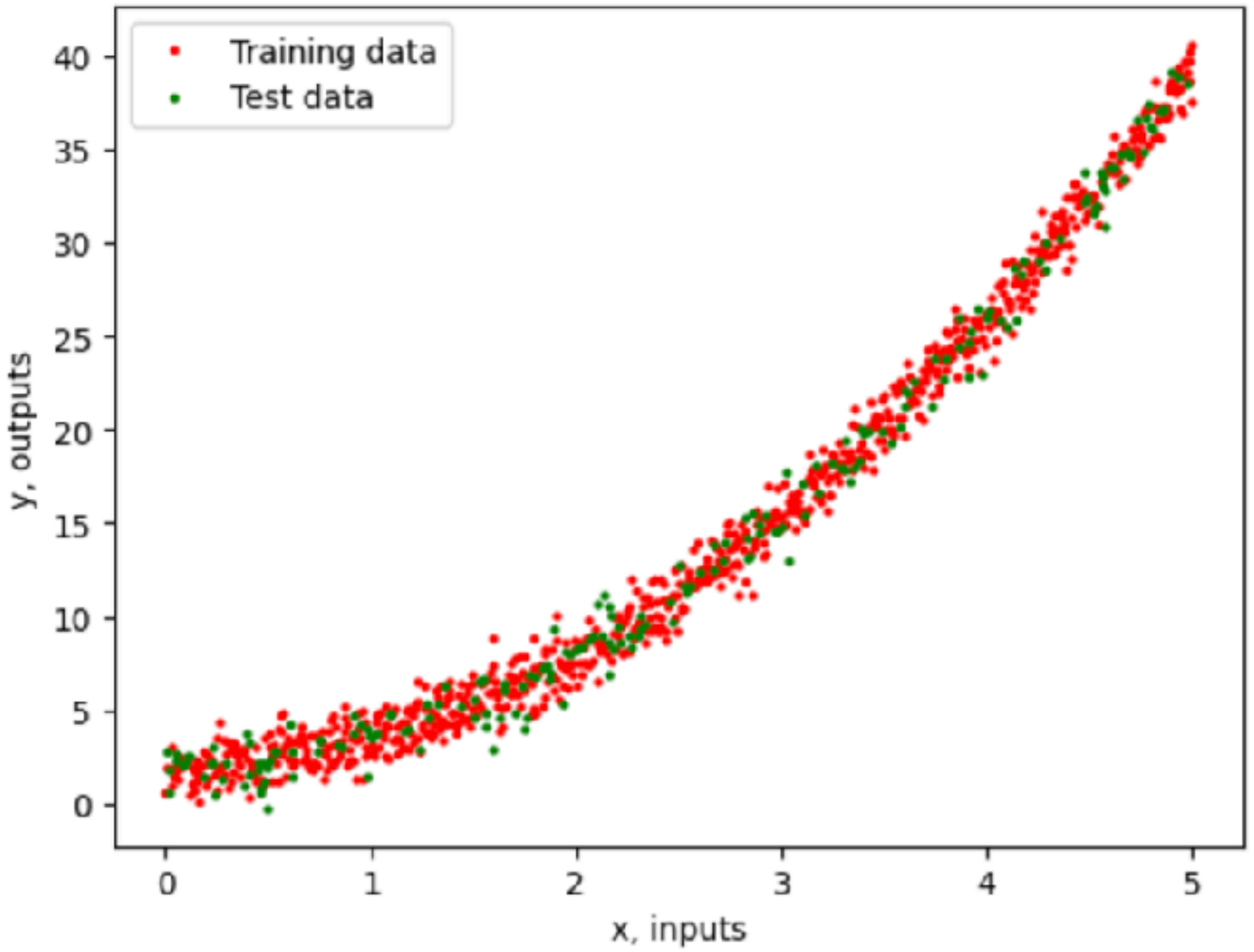
```python
# Create features

def design_matrix(x, degree):
    X = np.zeros((len(x), degree+1))
    for i in range(X.shape[1]):
        X[:, i] = x**i
    return X


degree = 2
X = design_matrix(x, degree)
y = y.reshape(-1, 1)
```

Shuffle and split

```
# Split data

n_train = int(0.8 * num_points)
n_test = num_points - n_train

shuff_index = np.random.permutation(num_points)
X_shuffle = X[shuff_index]
y_shuffle = y[shuff_index]

X_train = X_shuffle[:n_train]
X_test = X_shuffle[n_train:]
y_train = y_shuffle[:n_train]
y_test = y_shuffle[n_train:]
```

Visualize (training set)

```
# Plot training data
fig = plt.figure()
plt.plot(X_train[:, 1], y_train, 'ro', ms=2, label='Training data')
plt.plot(X_test[:, 1], y_test, 'go', ms=2, label='Test data')
plt.xlabel('x, inputs')
plt.ylabel('y, outputs')
plt.legend()
plt.show()
```



Define cost function and Gradient Descent

Cost function

```
def cost_function(X, y, theta):
    m = len(y)
    return 1/(2*m) * np.sum((X @ theta - y)**2)
```

Gradient Descent Function

```
def gradient_descent(X, y, theta, learning_rate, num_iters):
    m = len(y)
    J_history = np.zeros(num_iters)
    for i in range(num_iters):
        theta = theta - (learning_rate/m) * X.T @ (X @ theta - y)
        J_history[i] = cost_function(X, y, theta)
    return theta, J_history
```

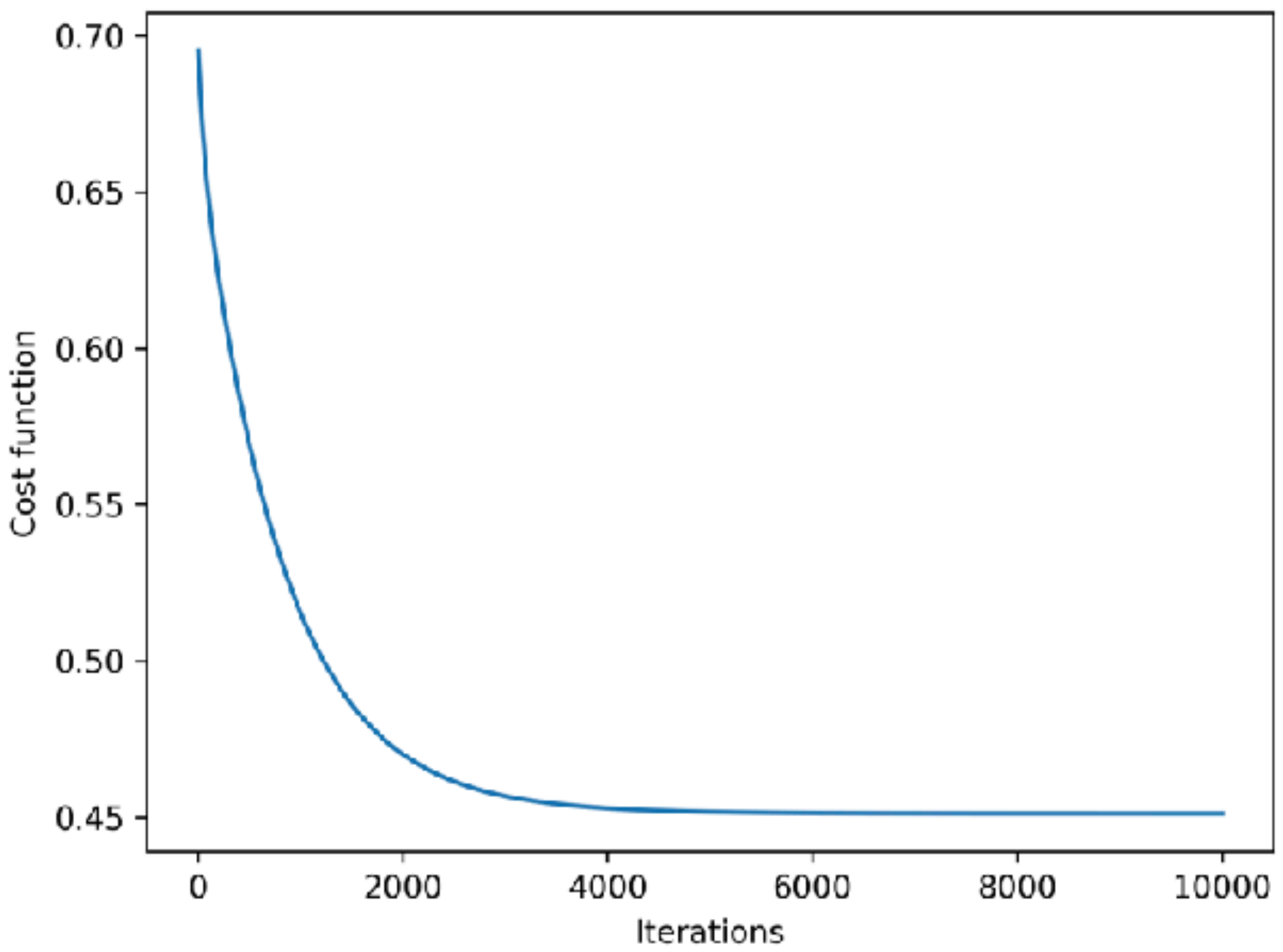
Gradient Descent Update

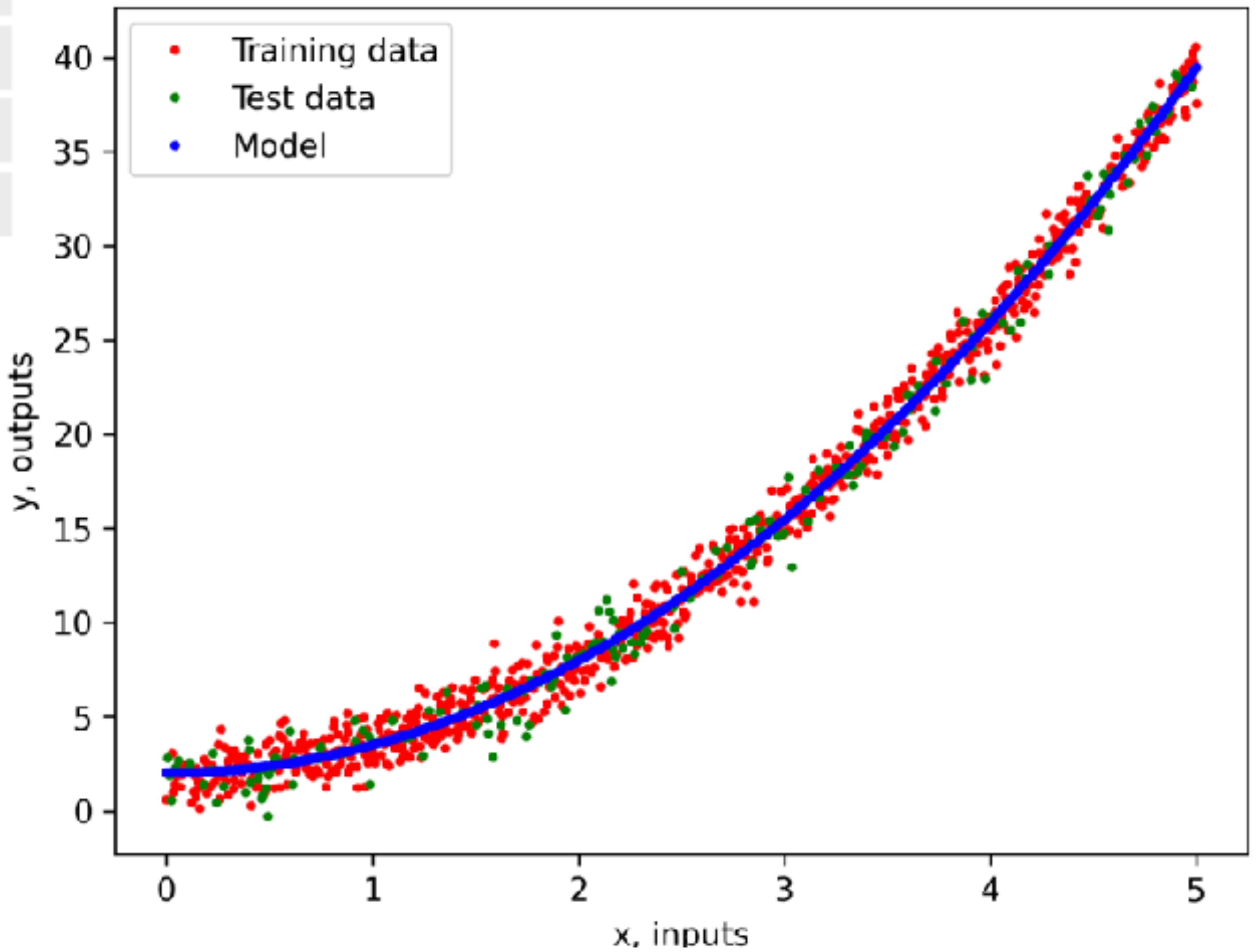
```
theta = np.random.randn(degree+1, 1)
learning_rate = 0.01
num_iters = 10000
theta, J_history = gradient_descent(X_train, y_train, theta, learning_rate, num_iters)
```

```
theta
✓ 0.0s
array([[ 2.04211351],
       [-0.02468485],
       [ 1.50370819]])
```

Plot result

```python
# plot comparison
fig = plt.figure()
plt.plot(X_train[:, 1], y_train, 'ro', ms=2, label='Training data')
plt.plot(X_test[:, 1], y_test, 'go', ms=2, label='Test data')
plt.plot(X_train[:, 1], X_train @ theta, 'bo', ms=2, label='Model')
plt.xlabel('x, inputs')
plt.ylabel('y, outputs')
plt.legend()
plt.show()
```





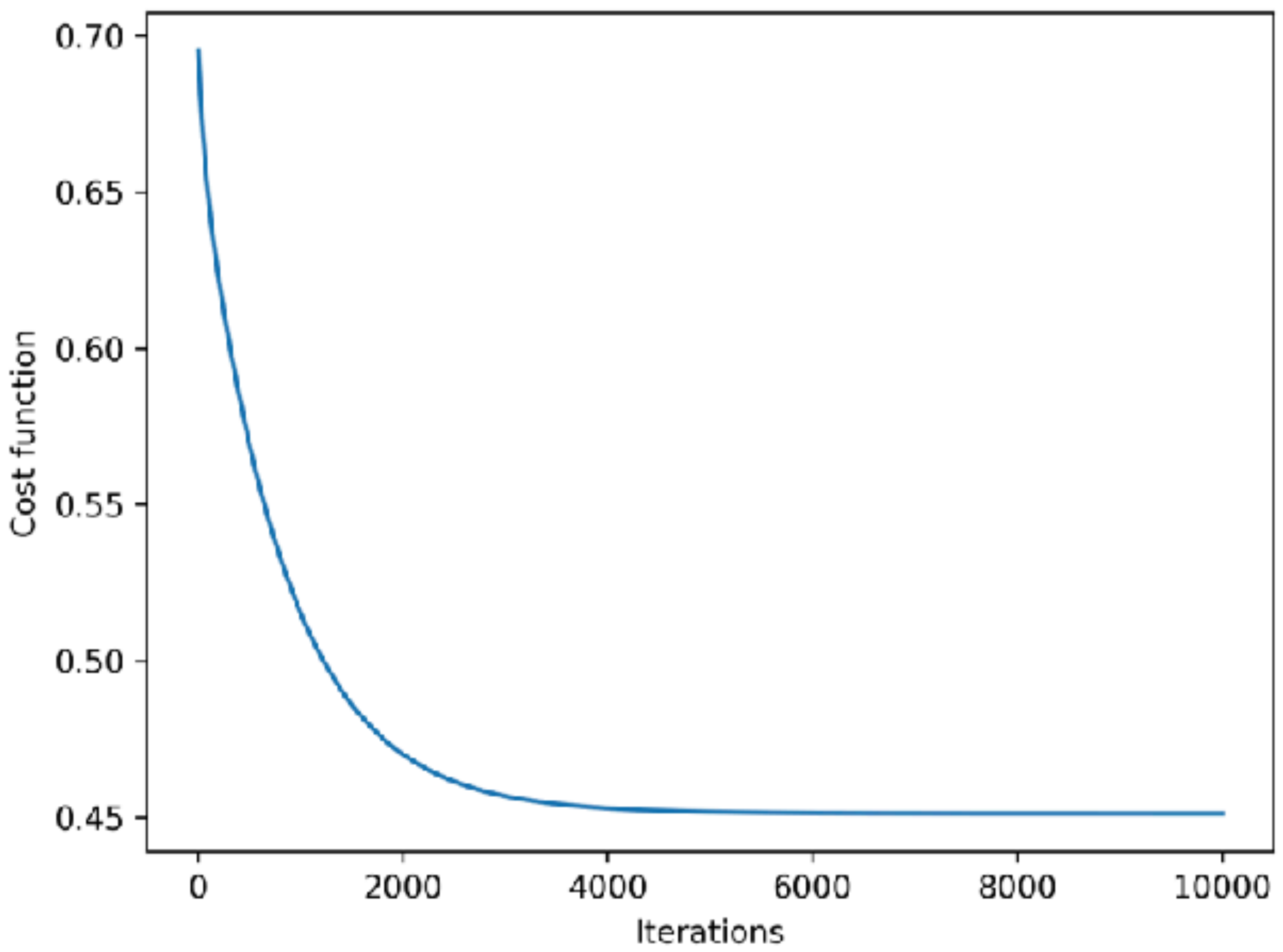
Plot result

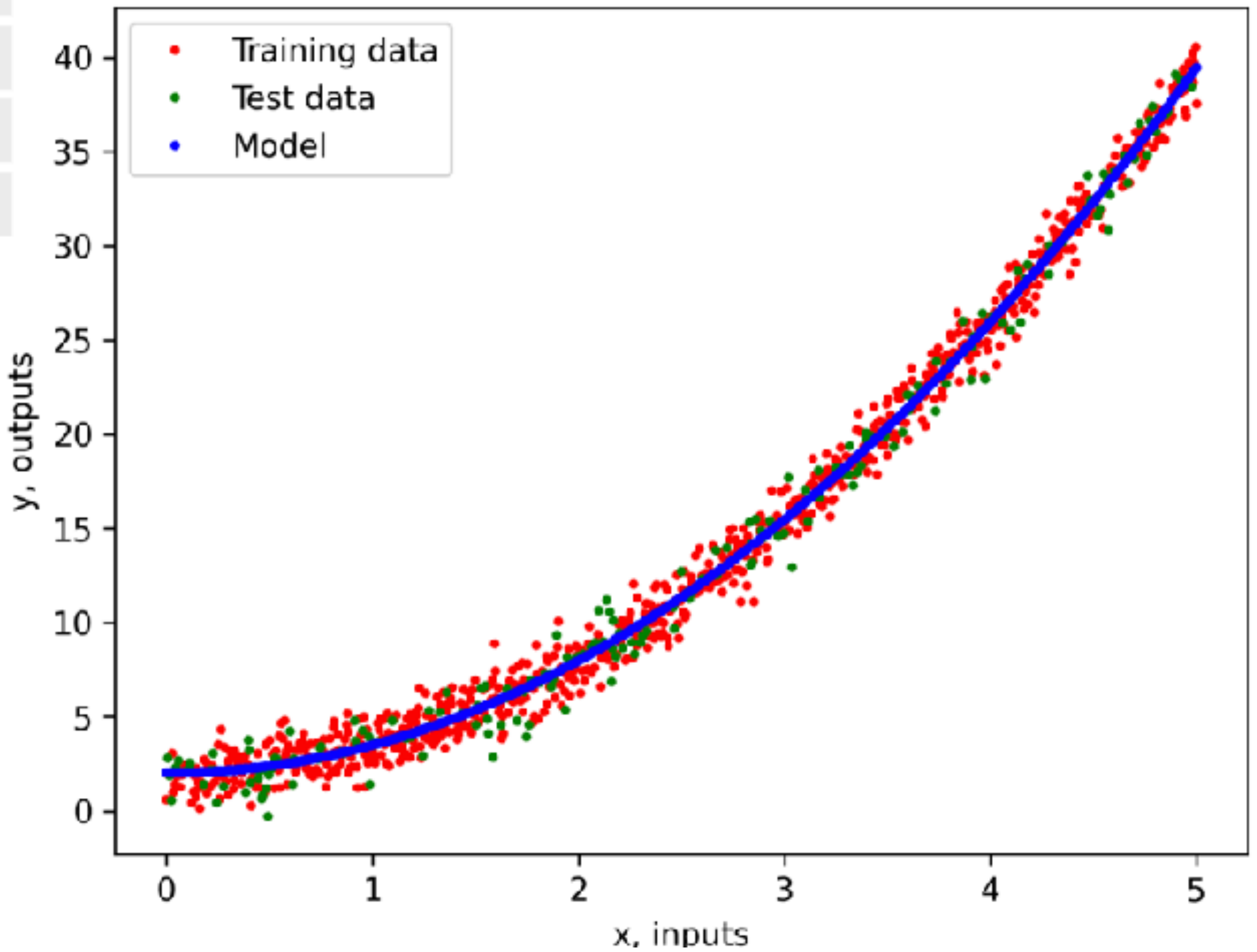
```python
# plot comparison
fig = plt.figure()
plt.plot(X_train[:, 1], y_train, 'ro', ms=2, label='Training data')
plt.plot(X_test[:, 1], y_test, 'go', ms=2, label='Test data')
plt.plot(X_train[:, 1], X_train @ theta, 'bo', ms=2, label='Model')
plt.xlabel('x, inputs')
plt.ylabel('y, outputs')
plt.legend()
plt.show()
```

Train and Test loss Comparison

```python
test_loss = cost_function(X_test, y_test, theta)
train_loss = cost_function(X_train, y_train, theta)

print(f'Test loss: {test_loss}')
print(f'Train loss: {train_loss}')
```

```
✓ 0.0s

Test loss: 0.508970692715051
Train loss: 0.4511700483353495
```





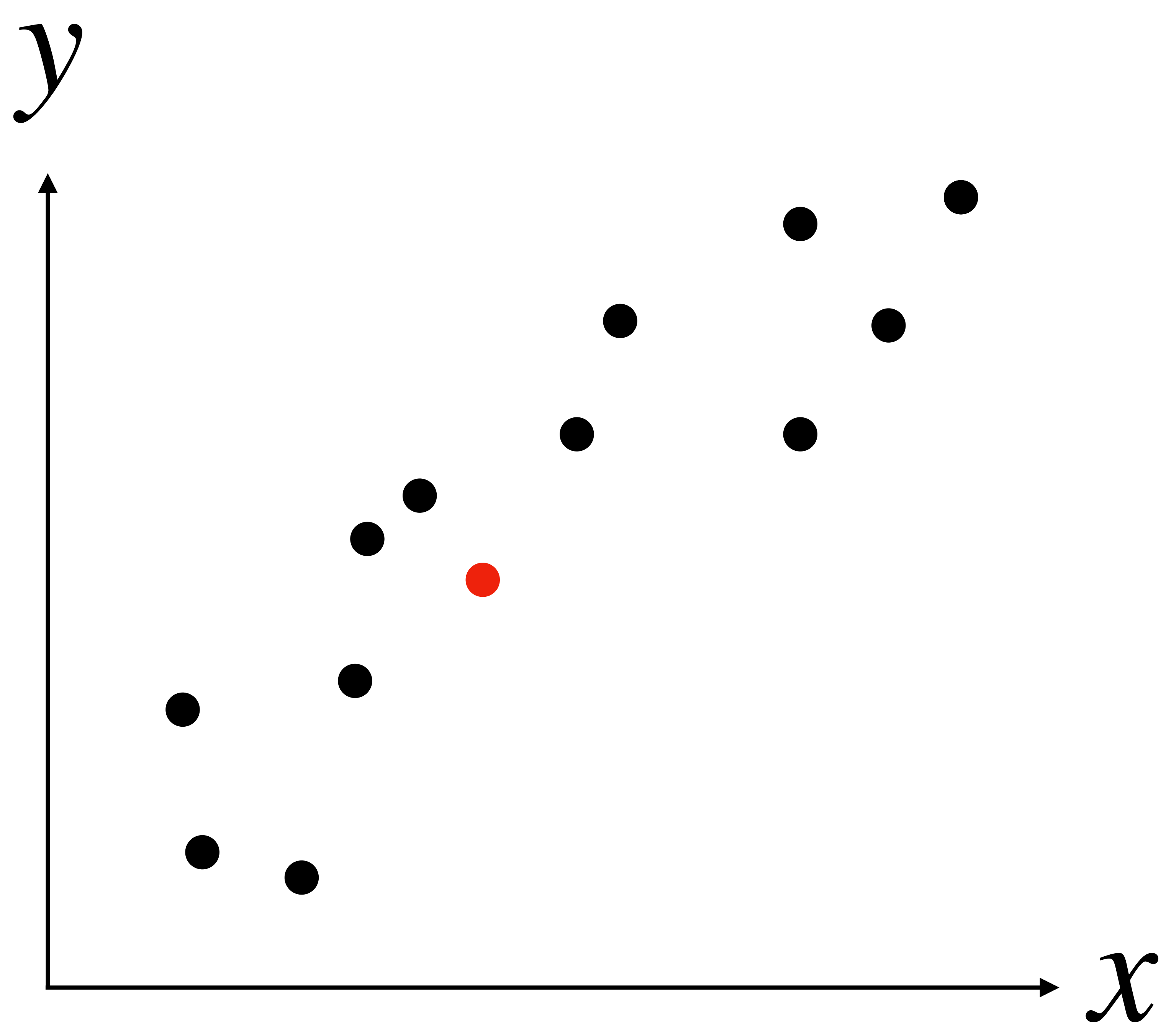
Given new input, what's the output?

Assuming $y \in \mathbb{R}$

$$x \xrightarrow{h} y$$

Given the data,
find a **function** h,
that predicts y, given x

$$\mathbf{y} = h(\mathbf{x})$$



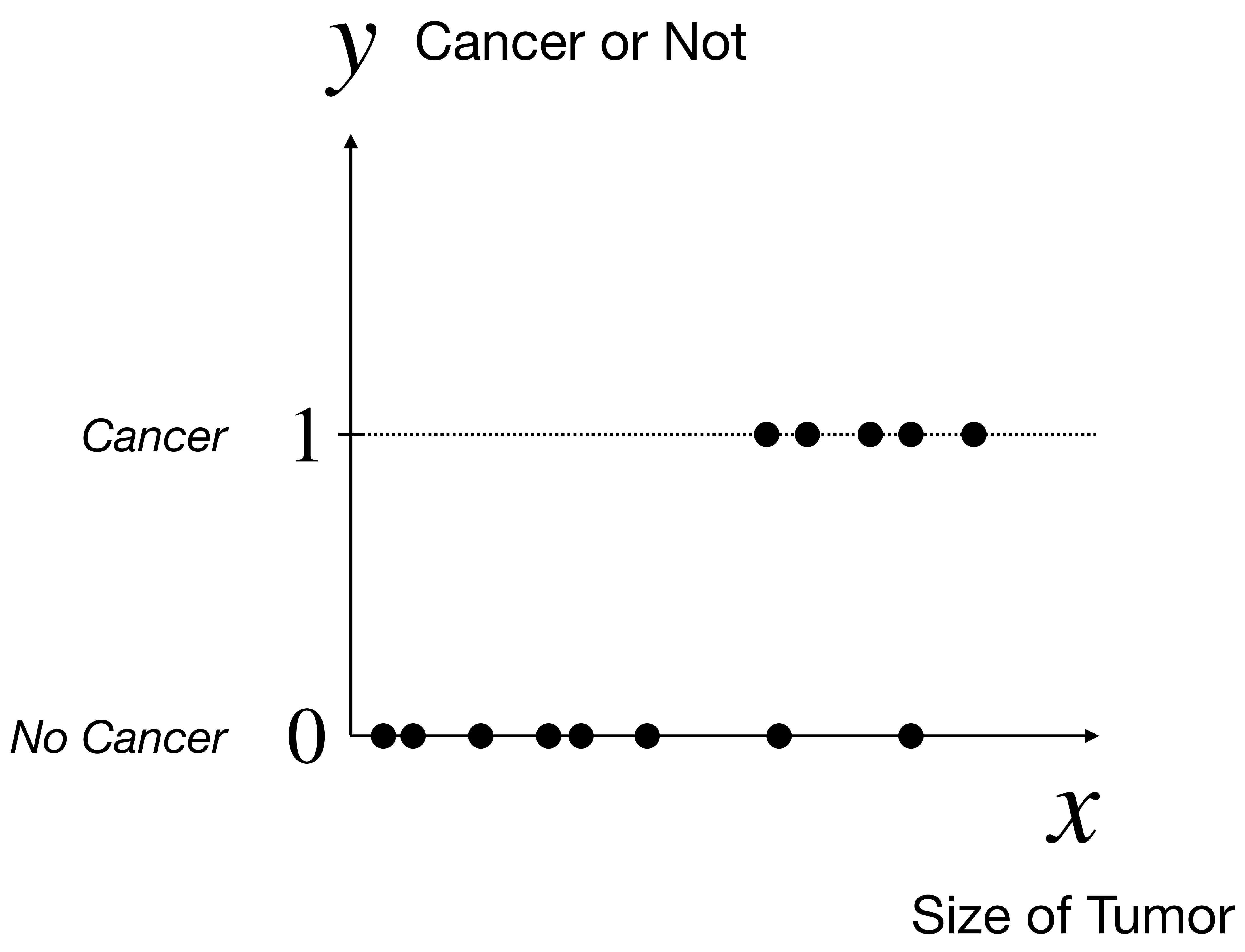
What if y is a label?

$$x \xrightarrow{h} y$$

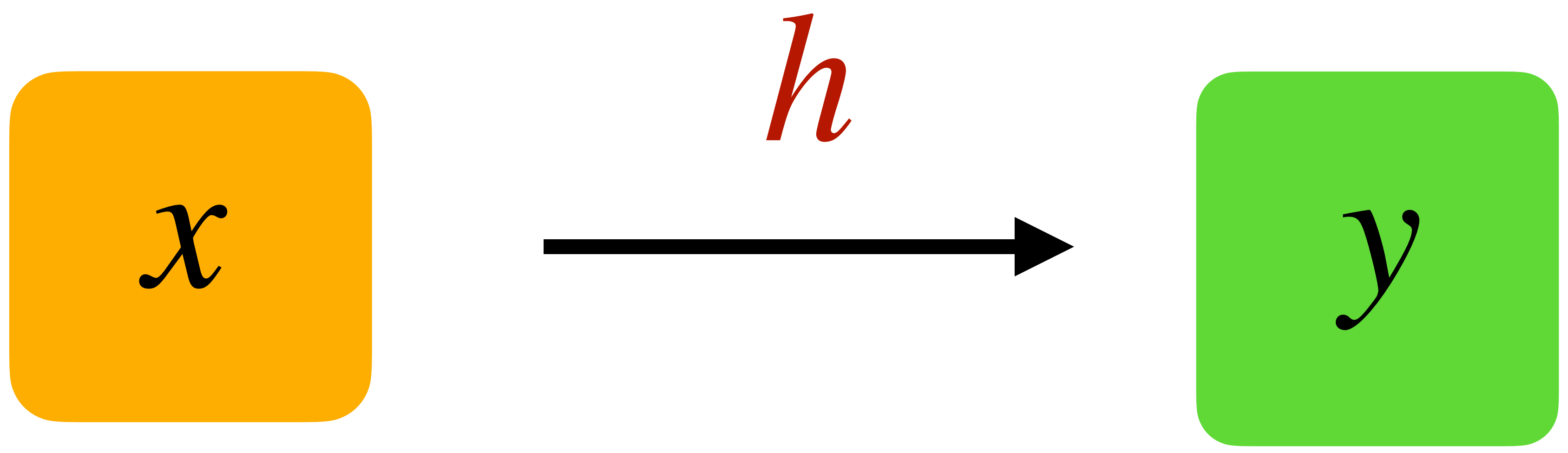
Given the data,
find a **function** h,
that predicts y, given x

$$\mathbf{y} = h(\mathbf{x})$$

$$\boxed{y \in [0,1]}$$



What if y is a label?

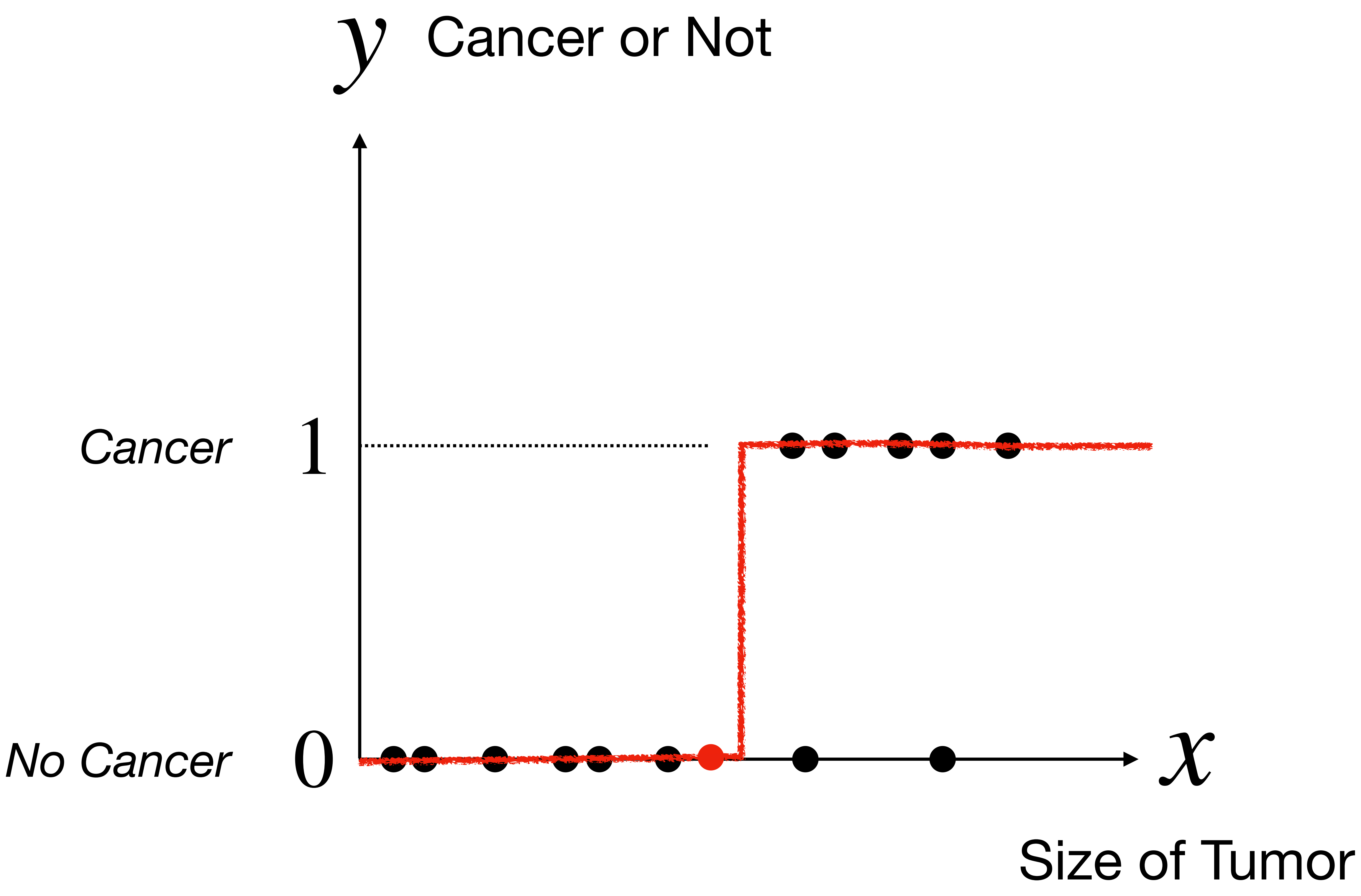


Given the data,
find a **function** h,
that predicts y, given x

$$\mathbf{y} = h(\mathbf{x})$$

$$y \in [0,1]$$

A step function, or threshold



What if y is a label?

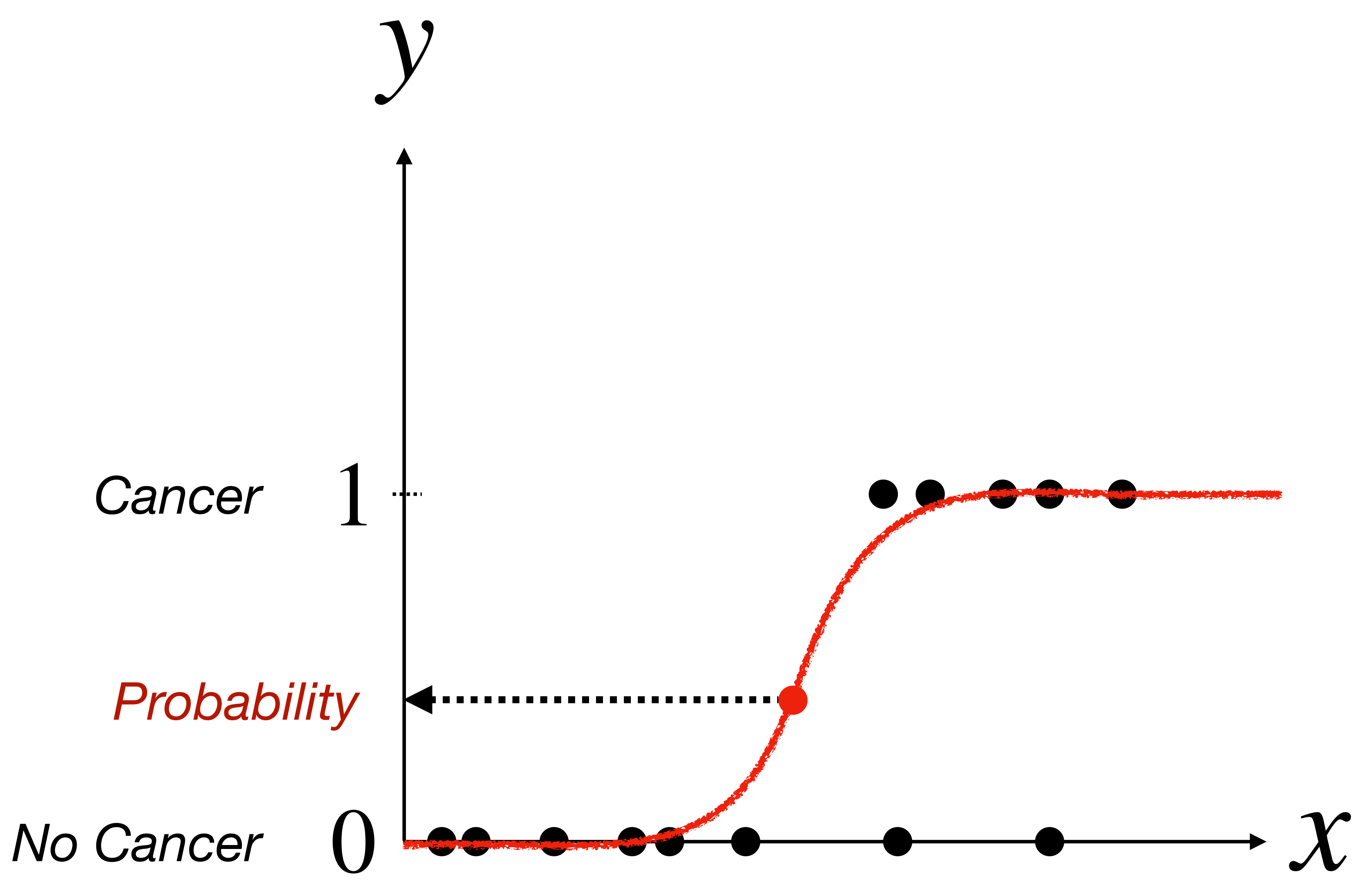
$$x \xrightarrow{h} y$$

Given the data,
find a **function** h,
that predicts y, given x

$$\mathbf{y} = h(\mathbf{x})$$

$$y \in [0,1]$$

A smooth function that returns probability of occurrence



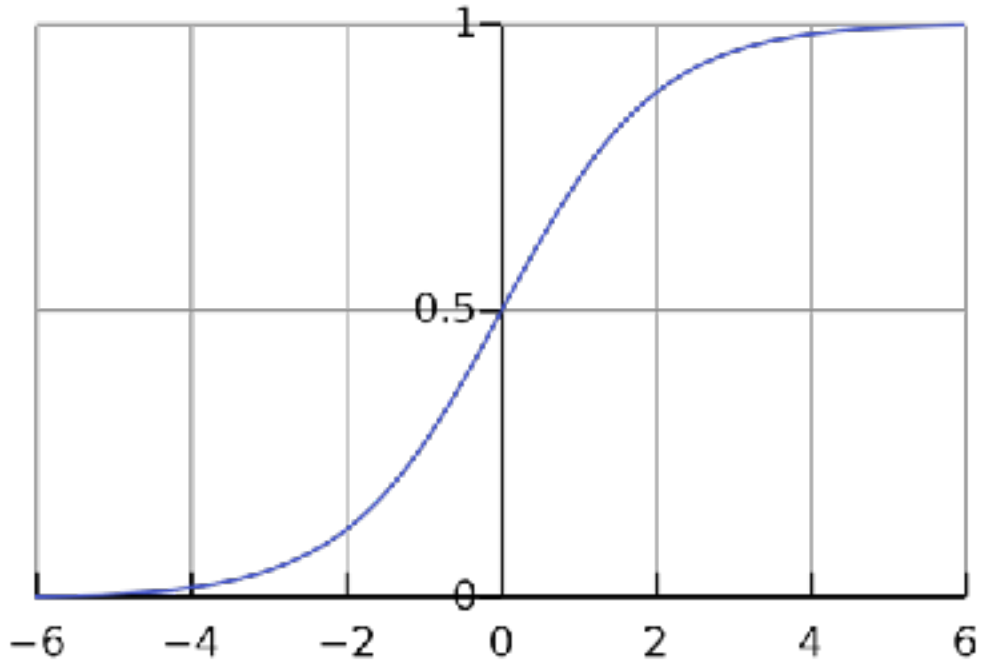
What if y is a label?

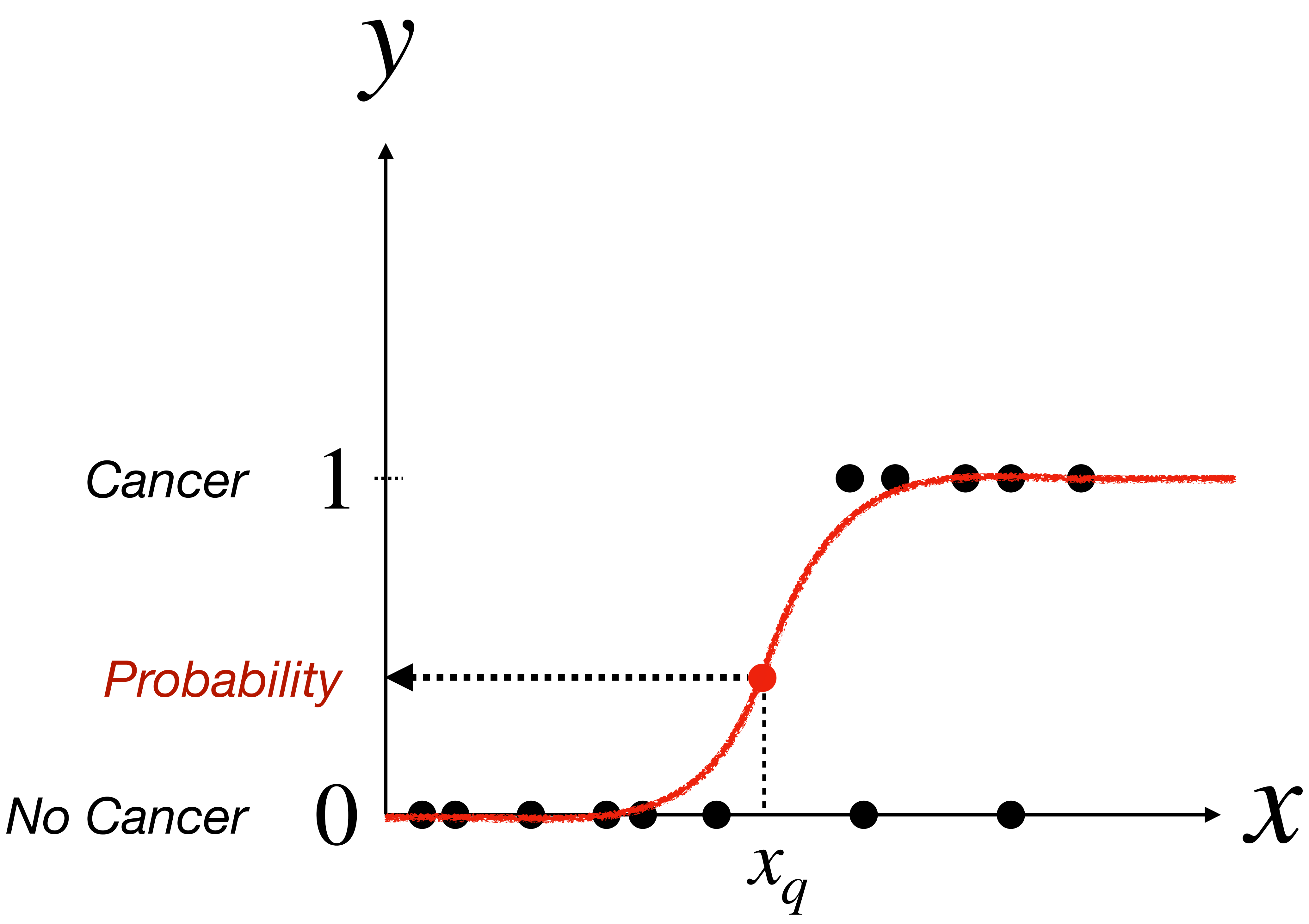
$y = h_\theta(x)$ & $y \in [0,1]$

A smooth function that returns probability of occurrence

$$y = \frac{1}{1+e^{-x}}$$

Logistic Function





$$h_\theta(x) = \frac{1}{1+e^{-(\theta_0+\theta_1 x)}}$$

What if y is a label?

$y = h_\theta(x)$ & $y \in [0,1]$

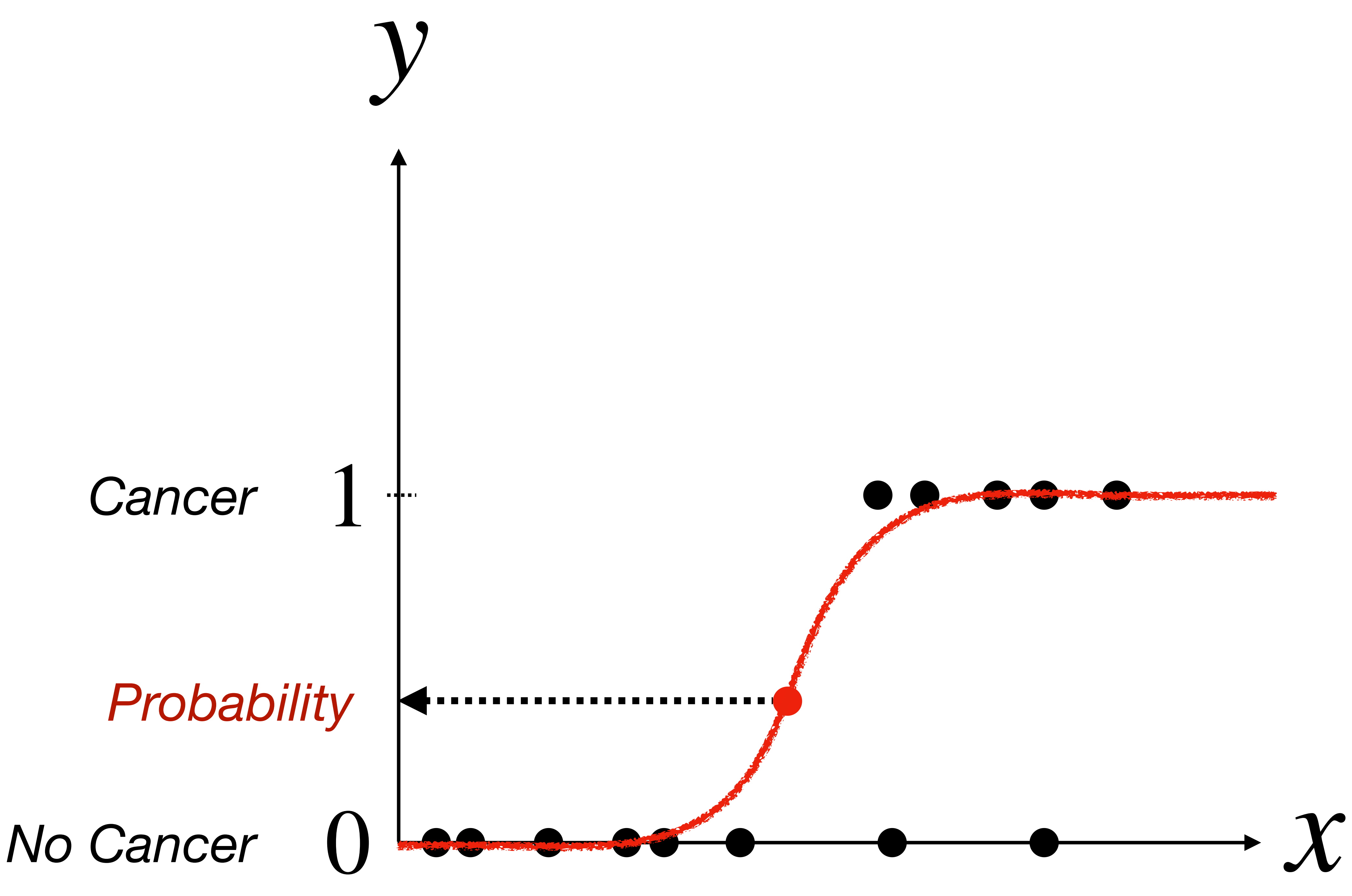
$$h_\theta(x) = \frac{1}{1 + e^{-(\theta^\top x)}}$$

For $x \in \mathbb{R}^n$

Where $\theta^\top x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots$

$$\theta = [\theta_0, \theta_1, \dots]$$
$$x = [x_0, x_1, \dots]$$

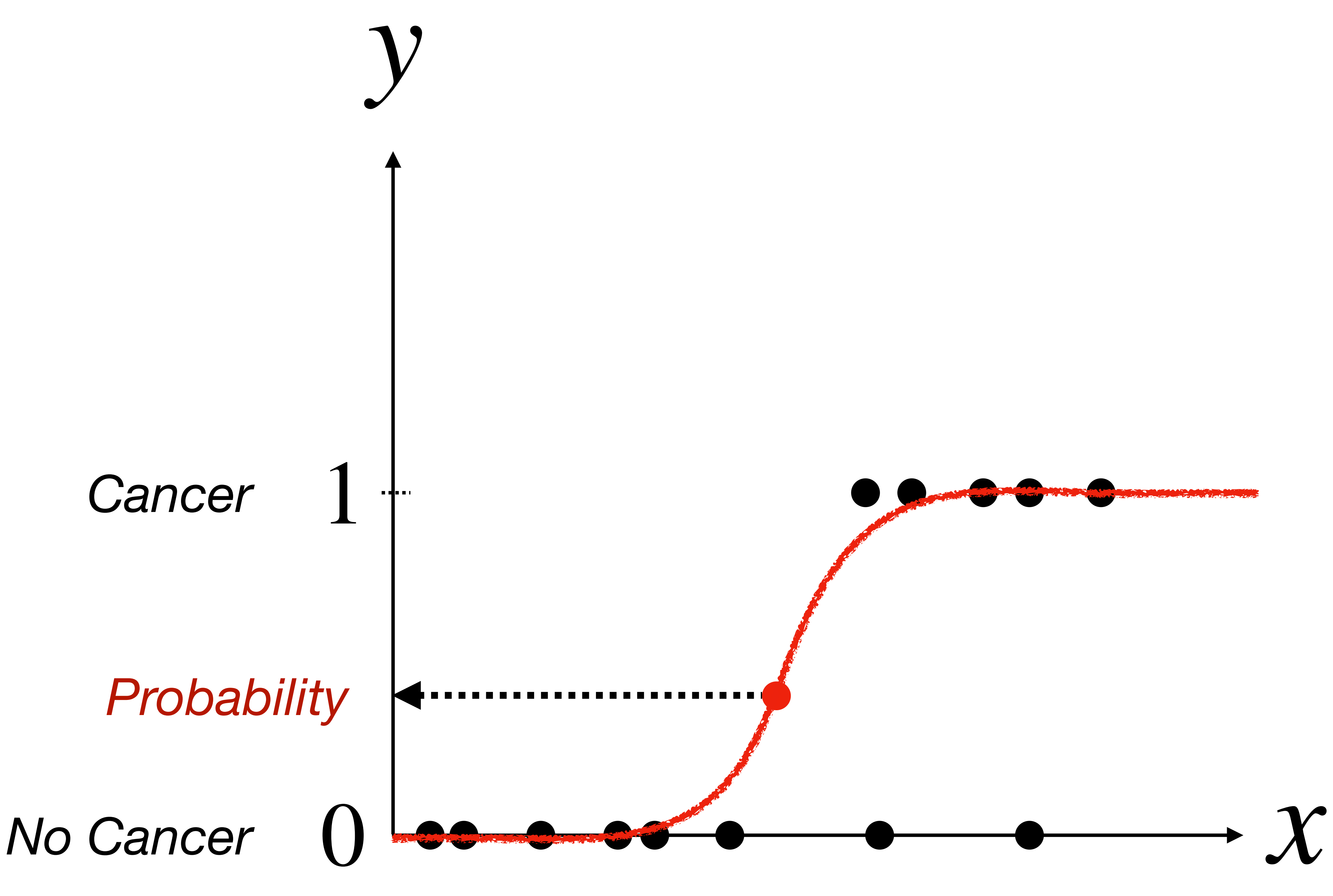
A smooth function that returns probability of occurrence



What if y is a label?

$$y = h_\theta(x) \quad \& \quad y \in [0,1]$$

$$h_\theta(x) = \frac{1}{1 + e^{-(\theta^\top x)}}$$

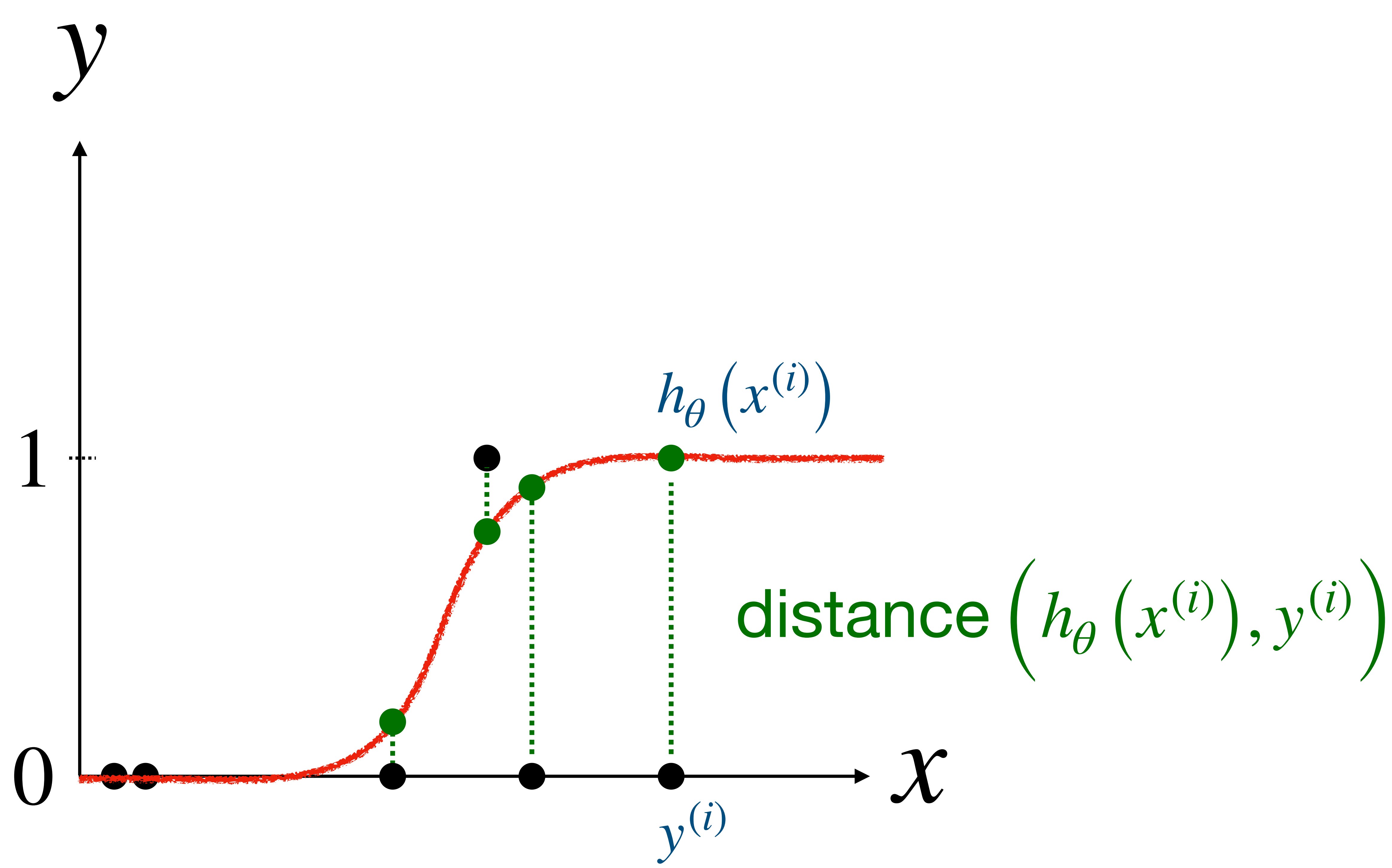


1. **Define a predictor:** the logistic function ✅
2. **Define a loss:** distance between function and data ❓
3. **Optimize loss**
4. **Test model**

Logistic Regression

$$y = h_\theta(x)$$

$$h_\theta(x) = \frac{1}{1 + e^{-(\theta^\top x)}} = g\left(\theta^\top x\right)$$



Linear predictor
negative log-likelihood or OLS

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{d} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Logistic predictor
Binary-cross entropy loss

$$\mathscr{L}(\theta) = \sum_{i=1}^{n} y^{(i)} \log h_\theta\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_\theta\left(x^{(i)}\right)\right)$$

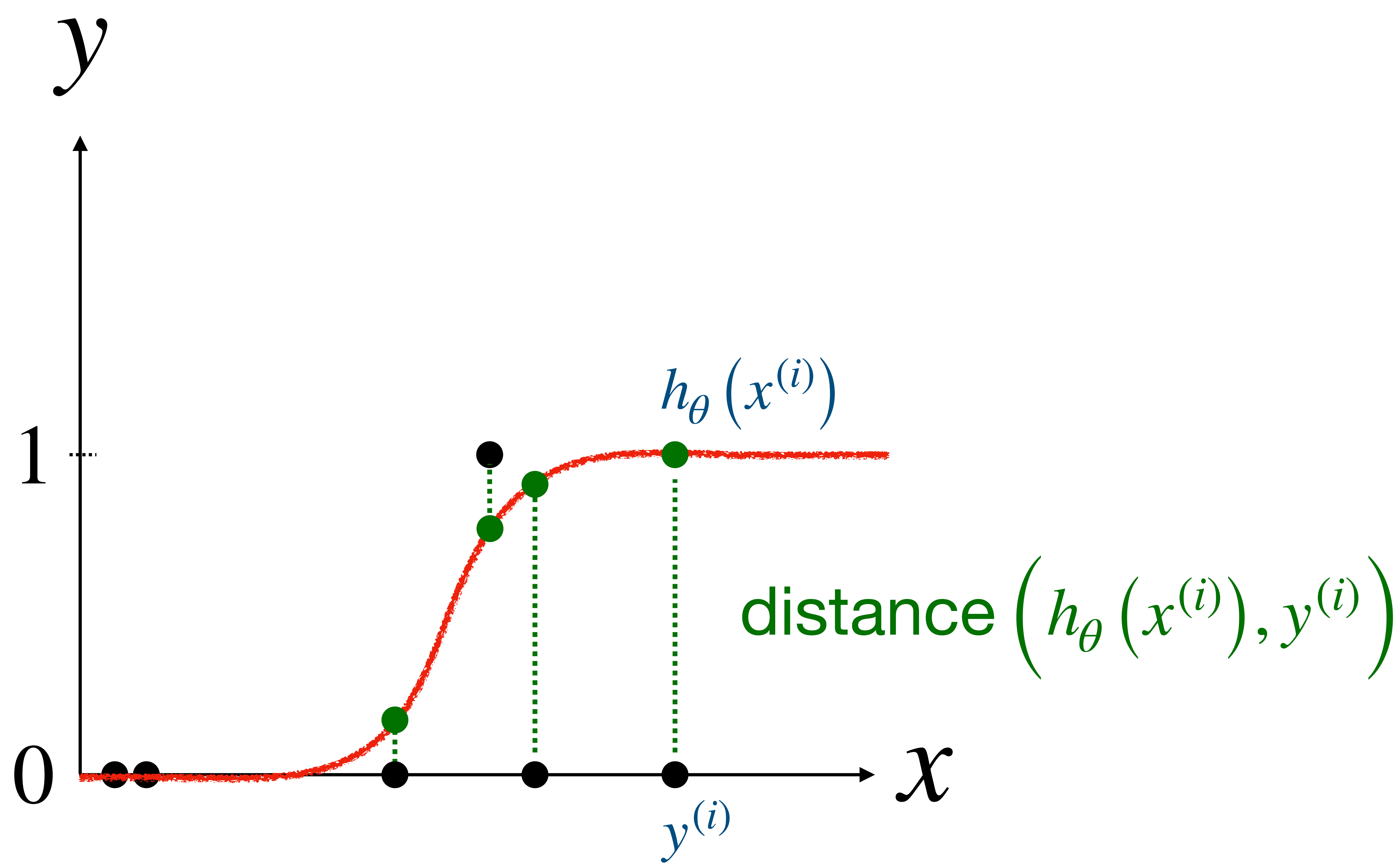
Compute gradient $\nabla \mathscr{L}(\theta)$

Gradient descent → Done!

Logistic Regression

$$y = h_\theta(x)$$

$$h_\theta(x) = \frac{1}{1 + e^{-\left(\theta^\top x\right)}} = g\left(\theta^\top x\right)$$



Linear predictor

negative log-likelihood or OLS

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{d}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Why not use an ordinary least squares loss?

Probabilistic Interpretation of Linear Regression

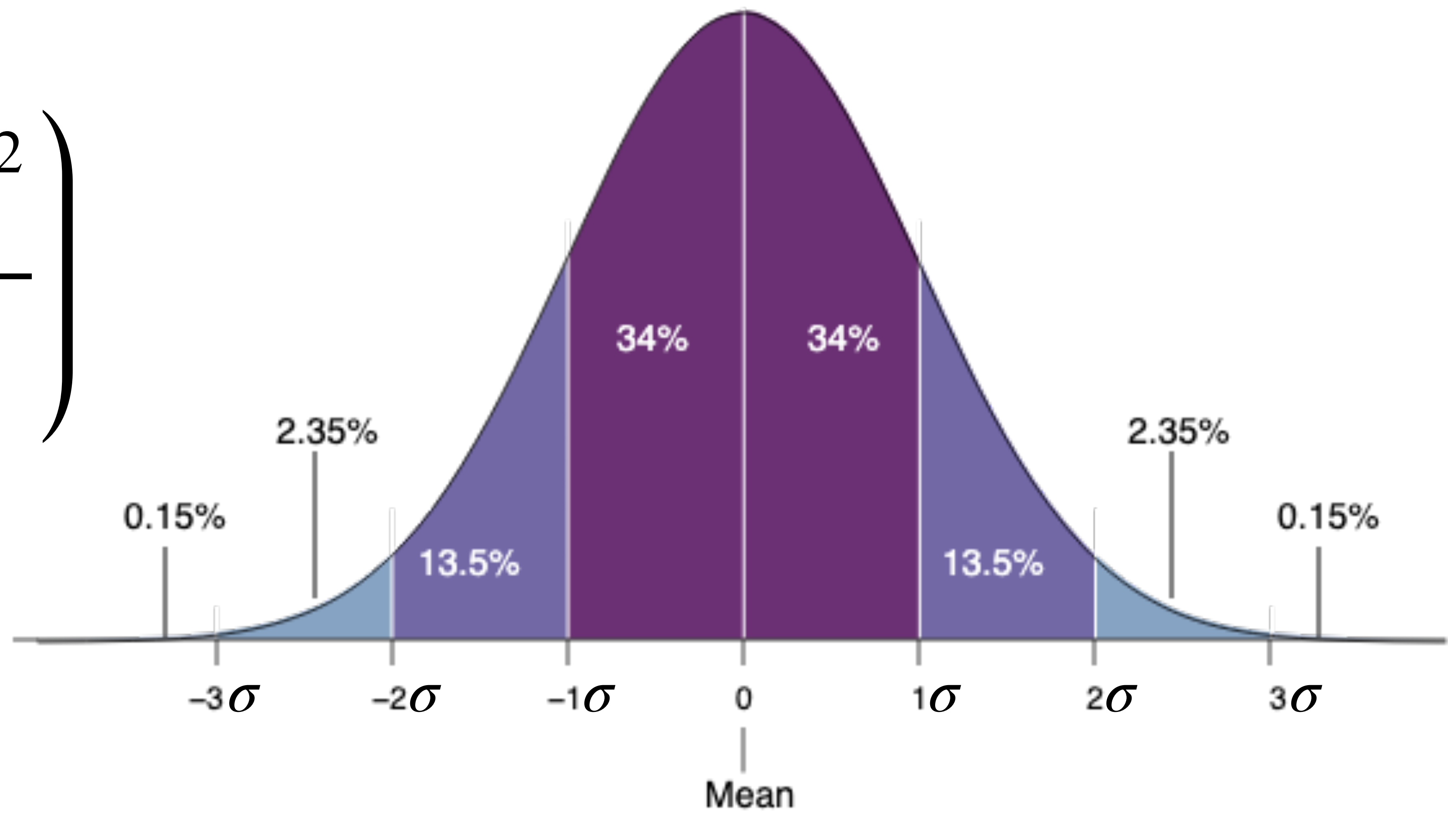
Assume noise is normally distributed around model

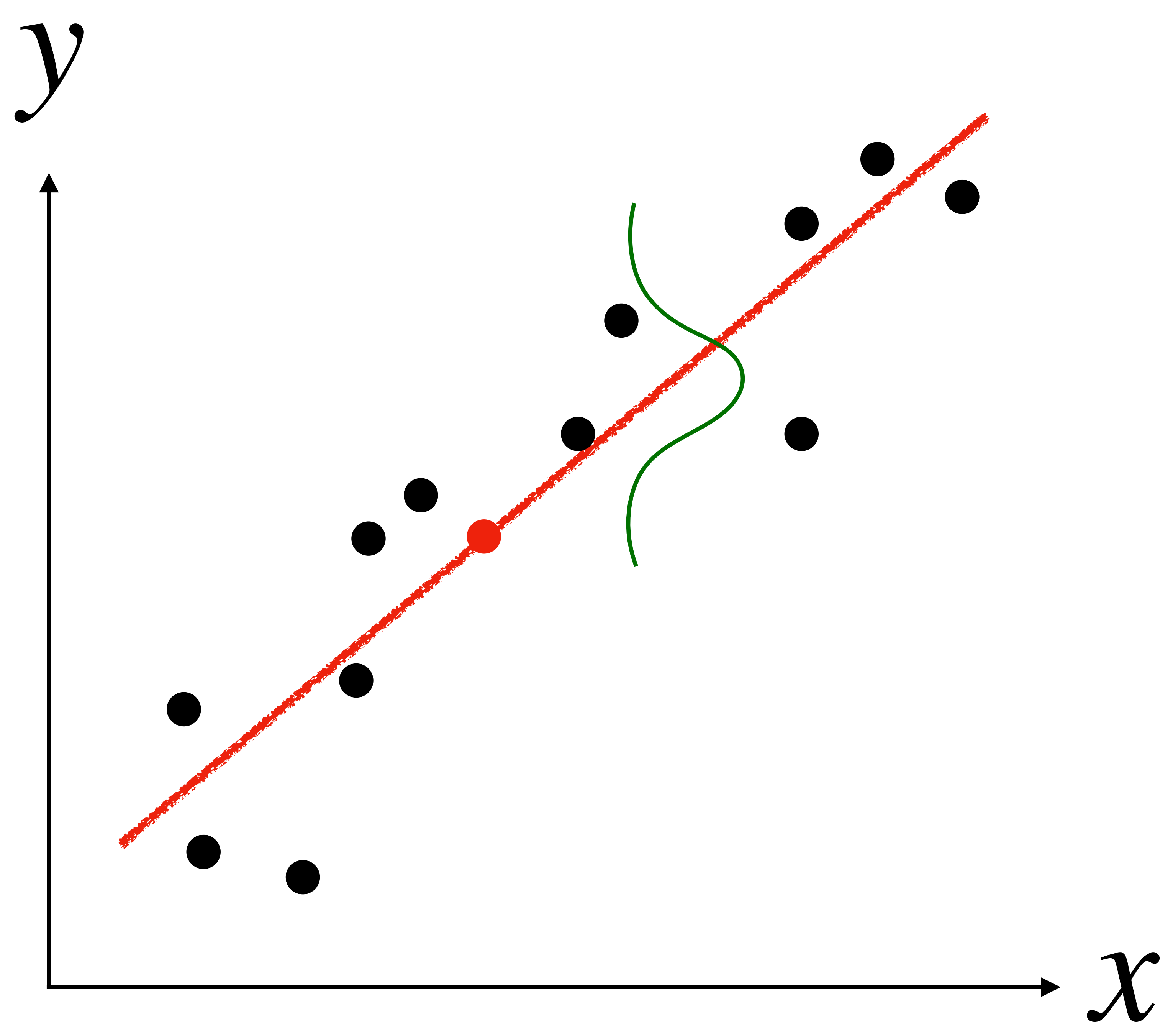
$$y^{(i)} = \theta^\top x^{(i)} + \varepsilon^{(i)}$$

Normally distributed

$$\mathcal{N}\left(0, \sigma^2\right)$$

$$p\left(\varepsilon^{(i)}\right) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(\varepsilon^{(i)}\right)^2}{2\sigma^2}\right)$$



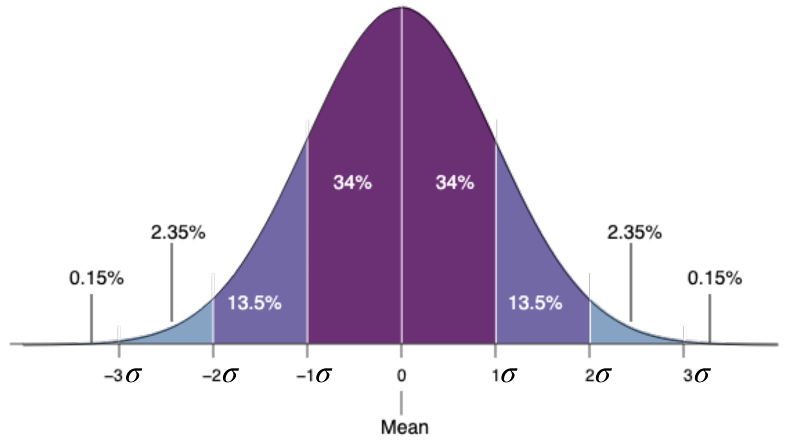


Probabilistic Interpretation

Assume noise is normally distributed around model

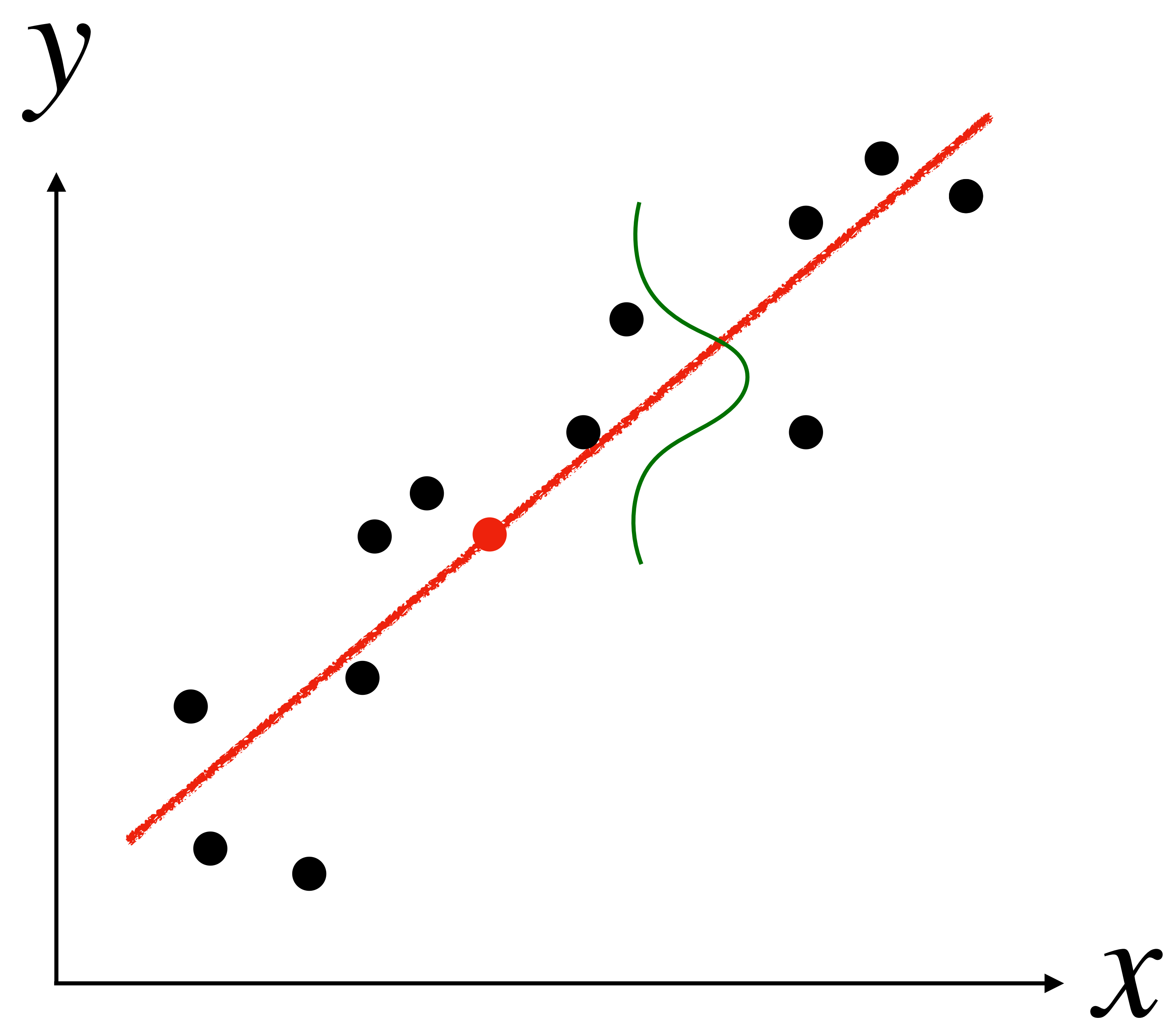
$$y^{(i)} = \theta^\top x^{(i)} + \varepsilon^{(i)}$$

$$\mathcal{N}\left(0, \sigma^2\right)$$



$$p\left(\varepsilon^{(i)}\right) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(\varepsilon^{(i)}\right)^2}{2\sigma^2}\right)$$

$$p\left(y^{(i)} \,|\, x^{(i)}; \theta\right) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(y^{(i)} - \theta^\top x^{(i)}\right)^2}{2\sigma^2}\right)$$

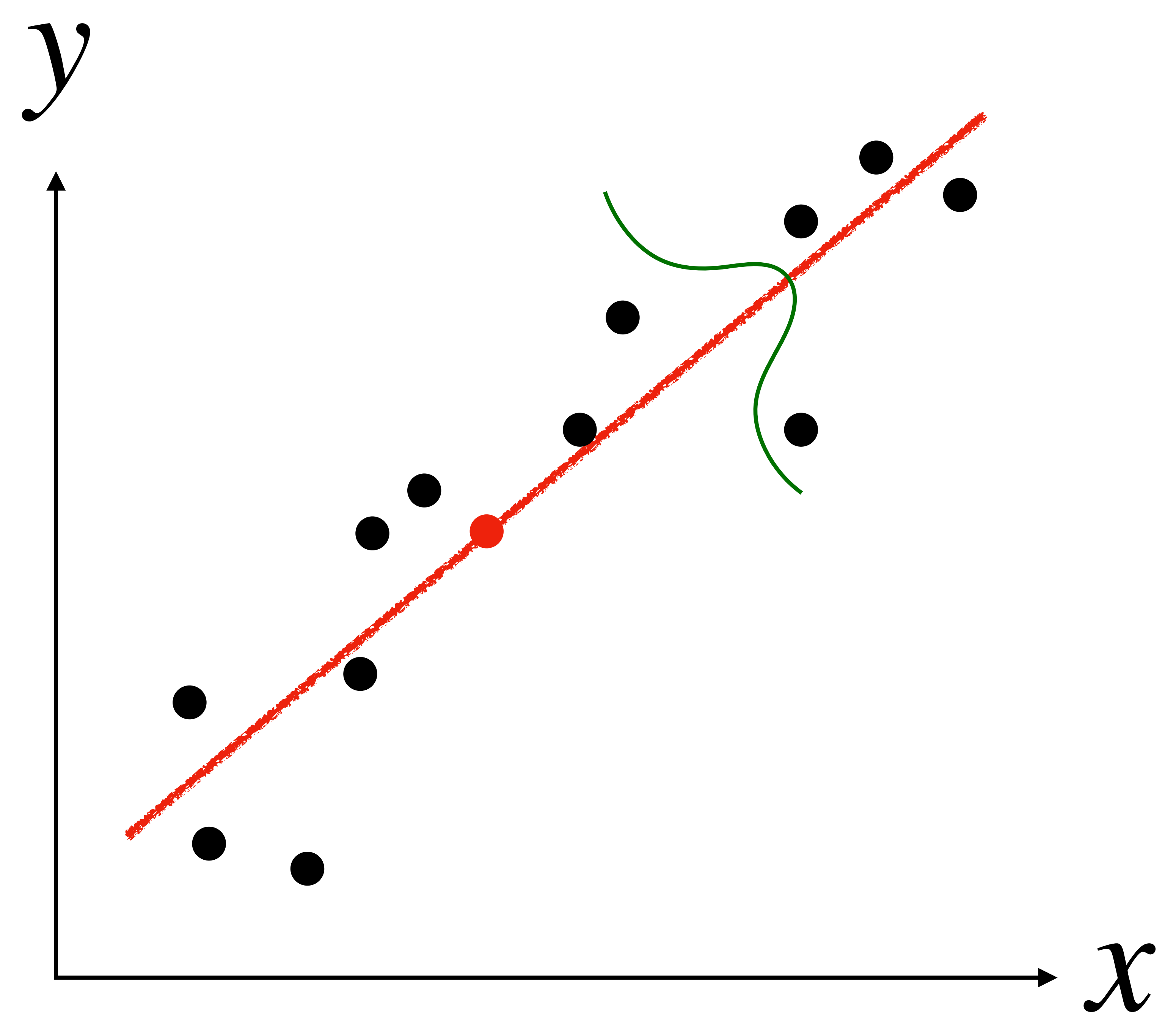


Likelihood of output given input

$$L(\theta) = \prod_{i=1}^{n} p\left(y^{(i)} \mid x^{(i)}; \theta\right)$$

Independent and Identically Distributed (IID)

$$= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(y^{(i)} - \theta^\top x^{(i)}\right)^2}{2\sigma^2}\right)$$



Log-likelihood

$$\mathscr{L}(\theta) = \log L(\theta)$$

$$= \log \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(y^{(i)} - \theta^\top x^{(i)}\right)^2}{2\sigma^2}\right)$$

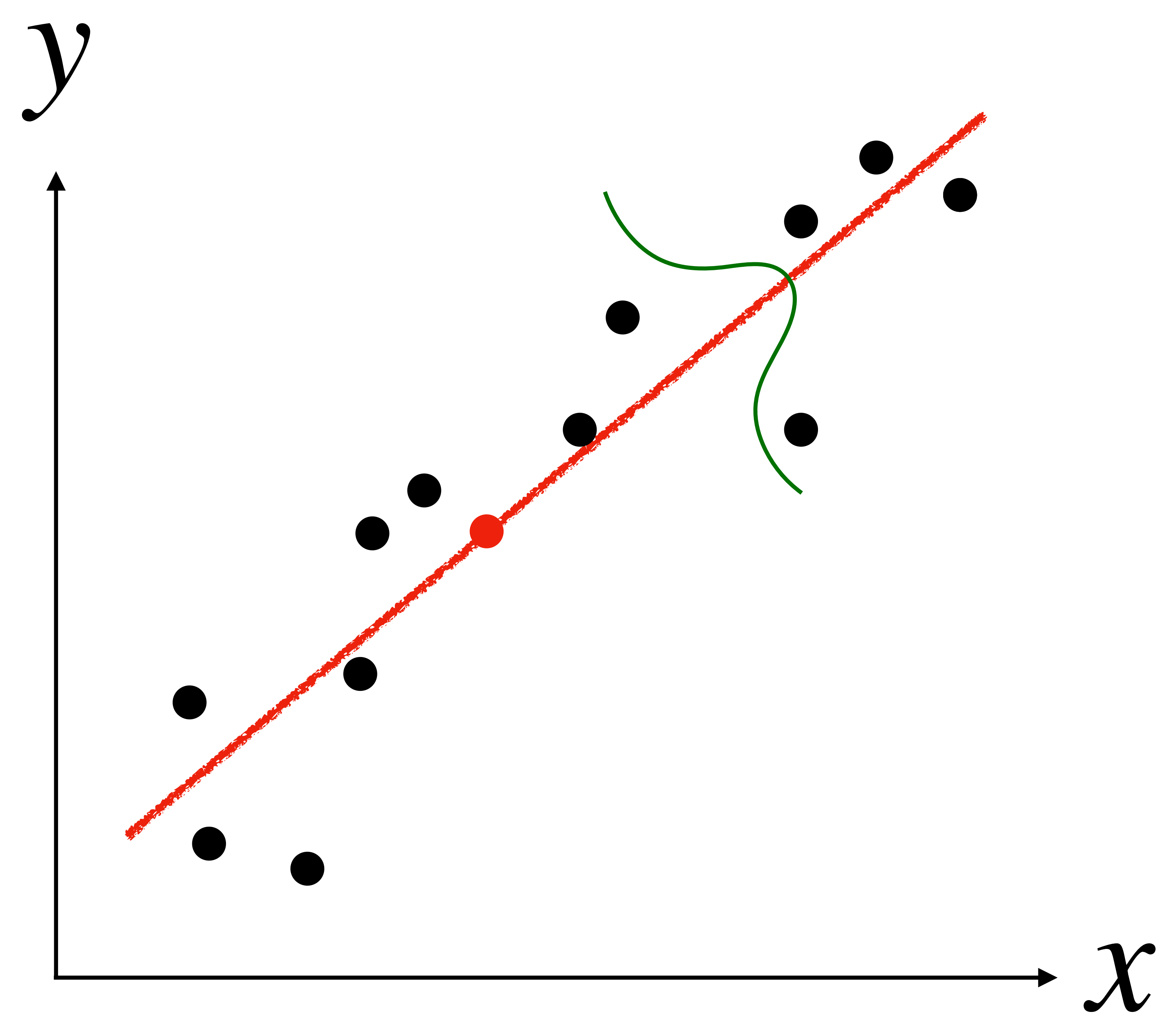
$$= \sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(y^{(i)} - \theta^\top x^{(i)}\right)^2}{2\sigma^2}\right) = n \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(y^{(i)} - \theta^\top x^{(i)}\right)^2$$

Maximize Log-likelihood

$$\mathscr{L}(\theta) = \log L(\theta)$$

$$= \sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(y^{(i)} - \theta^\top x^{(i)}\right)^2}{2\sigma^2}\right)$$

$$= n \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(y^{(i)} - \theta^\top x^{(i)}\right)^2$$



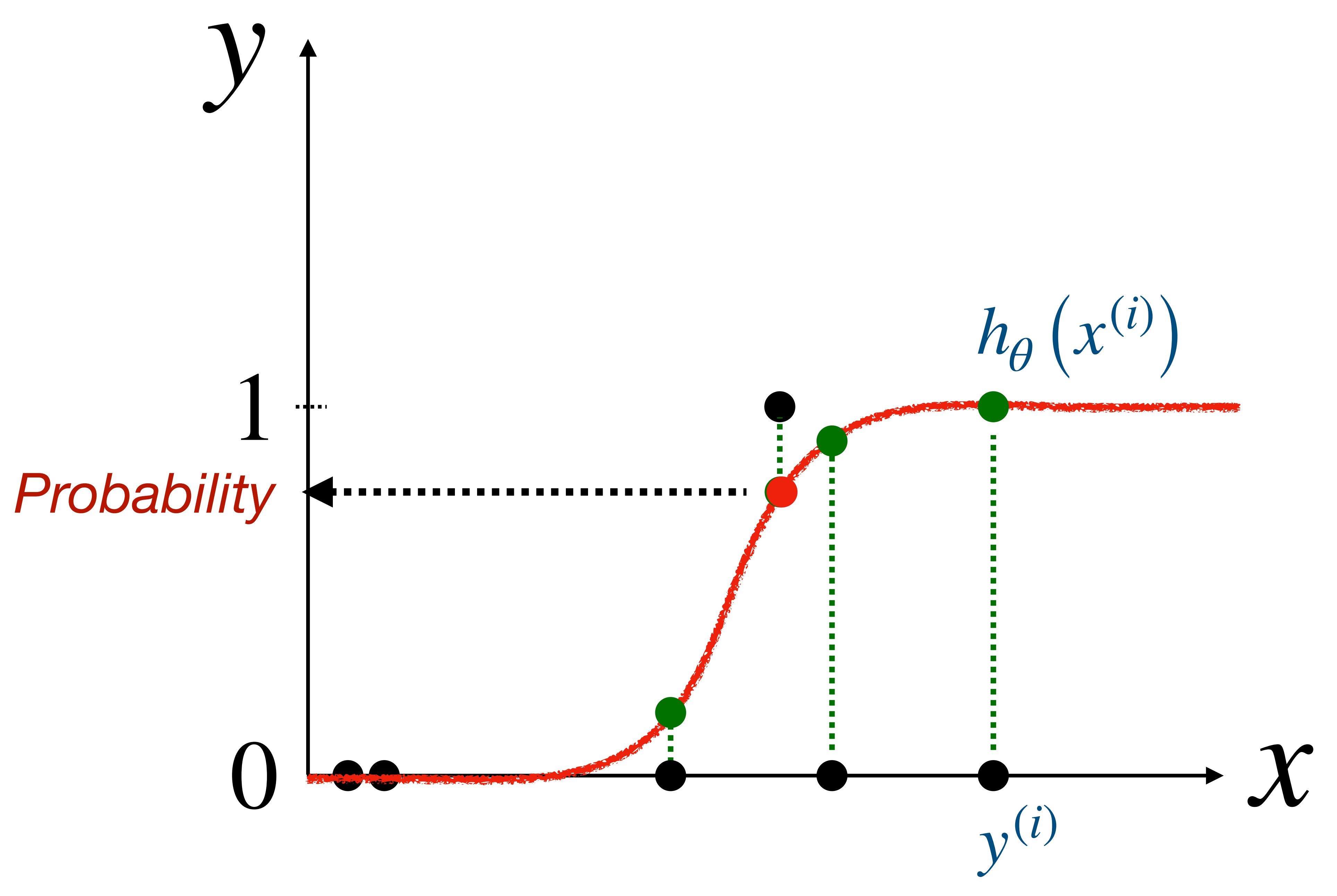
Maximize $\mathscr{L}(\theta) \longrightarrow$ **Minimize** $\frac{1}{2} \sum_{i=1}^{n} \left(y^{(i)} - \theta^\top x^{(i)}\right)^2$

What if the noise is not Gaussian?

Why not Least Squares?

$$y = h_\theta(x)$$

$$h_\theta(x) = \frac{1}{1 + e^{-(\theta^\top x)}} = g\left(\theta^\top x\right)$$



Probability of output given input

$$P\left(y = 1 \,\middle|\, x; \theta\right) = h_\theta(x)$$

$$P\left(y = 0 \,\middle|\, x; \theta\right) = 1 - h_\theta(x)$$

→

True label

$$p(y \,|\, x; \theta) = \left(h_\theta(x)\right)^{y} \left(1 - h_\theta(x)\right)^{1-y}$$

Likelihood!

For Bernoulli Distributed Noise

Bernoulli Distribution

Properties [edit]

If X is a random variable with a Bernoulli distribution, then:

$$\Pr(X=1) = p = 1 - \Pr(X=0) = 1 - q.$$

The probability mass function f of this distribution, over possible outcomes *k*, is

$$f(k;p) = \begin{cases} p & \text{if } k = 1, \\ q = 1 - p & \text{if } k = 0. \end{cases}$$ [3]

This can also be expressed as

$$f(k;p) = p^k (1-p)^{1-k} \quad \text{for } k \in \{0,1\}$$

or as

$$f(k;p) = pk + (1-p)(1-k) \quad \text{for } k \in \{0,1\}.$$

The Bernoulli distribution is a special case of the binomial distribution with $n = 1$.[4]

Define Log-likelihood

Likelihood

$$p(y\,|\,x;\theta) = \left(h_\theta(x)\right)^y \left(1 - h_\theta(x)\right)^{1-y}$$ for all (x, y) pair

$$L(\theta) = \prod_{i=1}^{n} p\left(y^{(i)}\,|\,x^{(i)};\theta\right)$$

$$= \prod_{i=1}^{n} h_\theta\left(x^{(i)}\right)^{y^{(i)}} \left(1 - h_\theta(x^{(i)})\right)^{1-y^{(i)}}$$

log

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} y^{(i)} \log h_\theta\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_\theta\left(x^{(i)}\right)\right)$$

Maximize Log-likelihood

$$\mathscr{L}(\theta) = \sum_{i=1}^{n} y^{(i)} \log h_\theta\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_\theta\left(x^{(i)}\right)\right)$$

Update rule

while not converged:

$$\theta := \theta + \alpha \nabla_\theta \mathscr{L}(\theta)$$

$\longrightarrow$ **Derive**

Gradient Descent

for t = 1…T:

$$\theta := \theta - \alpha \sum_{i=1}^{n} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x^{(i)}

Generalized Linear Models

Gaussian Distribution $\longrightarrow$ **Linear Regression**

Bernoulli Distribution $\longrightarrow$ **Logistic Regression**

Update rule

$$\theta := \theta - \alpha \sum_{i=1}^{n} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right) x^{(i)}$$

Exponential Family

Family of distributions for which we can derive the same update rule

Assumption: $p(y \mid x; \theta)$ is an exponential family

Data

$$p(y; \eta) = b(y) \exp\left\{\eta^\top y - a(\eta)\right\}$$

Parameters

- $b(y)$ is called the base measure (not depend on η)
- $a(\eta)$ is called the log partition function (not depend on y)
- $a(\eta)$, y and $b(y)$ are scalar. η and y have the same dimensions.

Example 1: Bernoulli Distribution -> Logistic Regression

Data

$$p(y;\eta) = b(y)\,\exp\left\{\eta^\top y - a(\eta)\right\}$$

Natural Parameters

Bernoulli Distribution

$$p(y;\phi) = \phi^y(1-\phi)^{1-y} = \exp\left\{ y \underbrace{\log\frac{\phi}{1-\phi}}_{\eta} + \underbrace{\log(1-\phi)}_{a(\eta)} \right\}$$

Show that term is only a function of η

Example 2: Gaussian Distribution -> Linear Regression

Data

$$p(y; \eta) = b(y)\ \exp\left\{\eta^\top y - a(\eta)\right\}$$

Natural Parameters

Gaussian Distribution

$$p(y; \mu) = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{1}{2}(y-\mu)^2\right\} = \underbrace{\frac{1}{\sqrt{2\pi}} e^{-y^2/2}}_{b(y)} \left\{\underbrace{\mu}_{\eta} y - \underbrace{\frac{1}{2}\mu^2}_{a(\eta)}\right\}$$

Why do we care?

Data

$$p(y;\eta) = b(y)\,\exp\left\{\eta^\top y - a(\eta)\right\}$$

Natural Parameters

$\theta^\top x$

Inference is Easy: $E[y;\eta] = \dfrac{da(\eta)}{d\eta}$ $\quad Var[y;\eta] = \dfrac{d^2 a(\eta)}{d\eta^2}$

Learning is Easy: **Maximum Likelihood Estimation** leads to **convex** problem in η

Generalized Linear Models

Assumption: $p(y \mid x; \theta)$ is an exponential family

Data Type → Probability Distribution

Binary → Bernoulli ⟶ **Logistic Regression**

Real → Gaussian ⟶ **Linear Regression**

Counts → Poisson

Positive Real → Gamma, Exponential

Distributions → Dirichlet