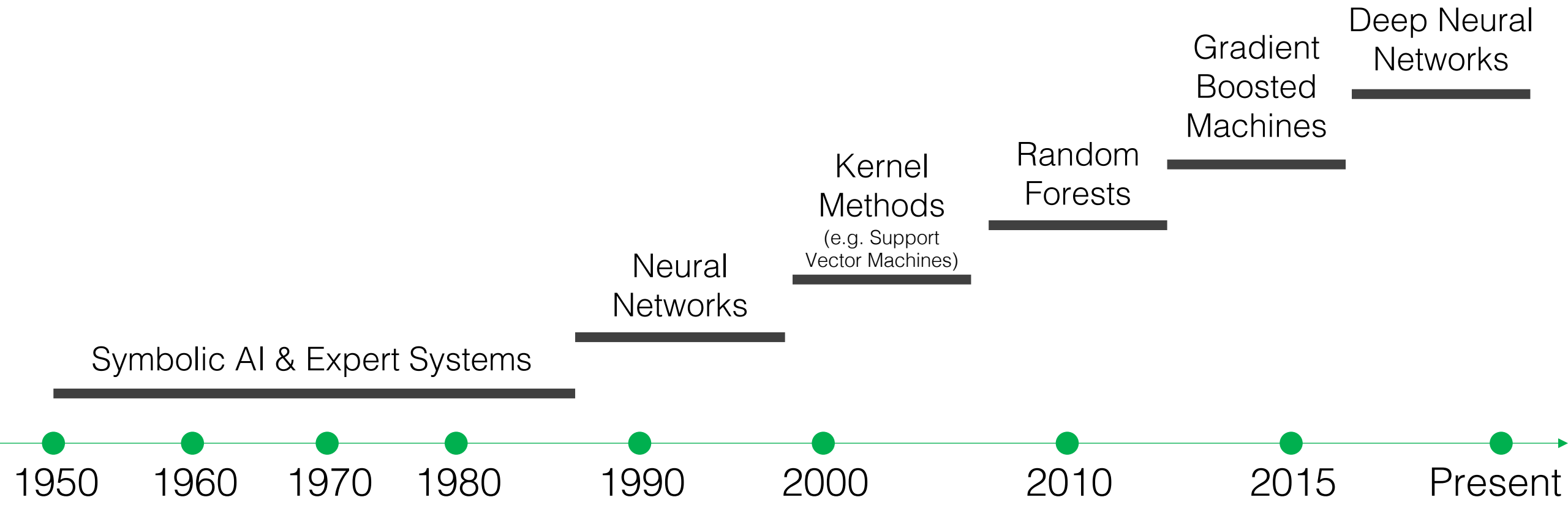
# Machine Learning II

# History




François Chollet, *Deep Learning with Python*, 2017

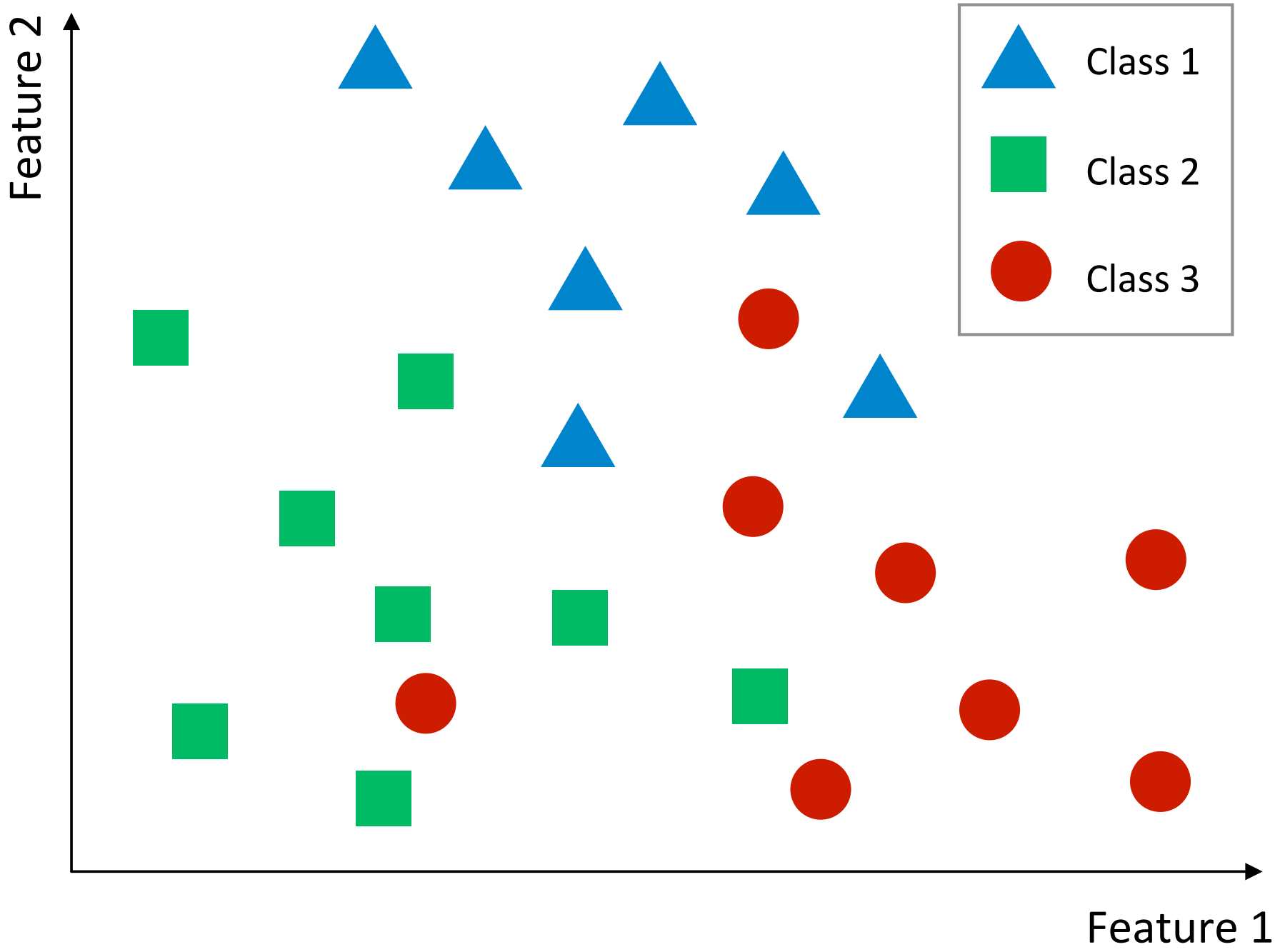
# K-Nearest Neighbors

Classification and Regression

# K Nearest Neighbor Classifier

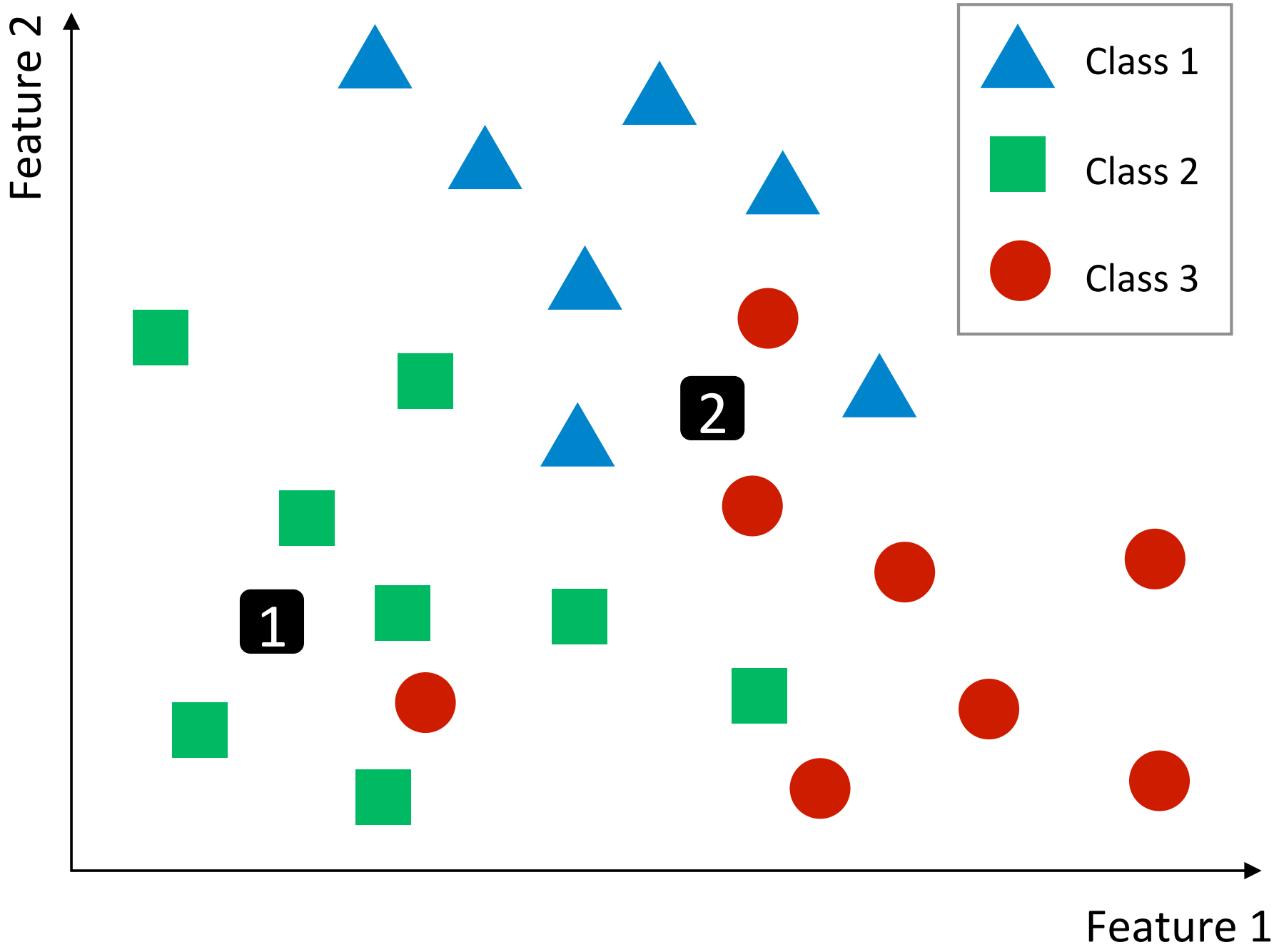
**Step 1**:
Training

Every new data point is a model parameter



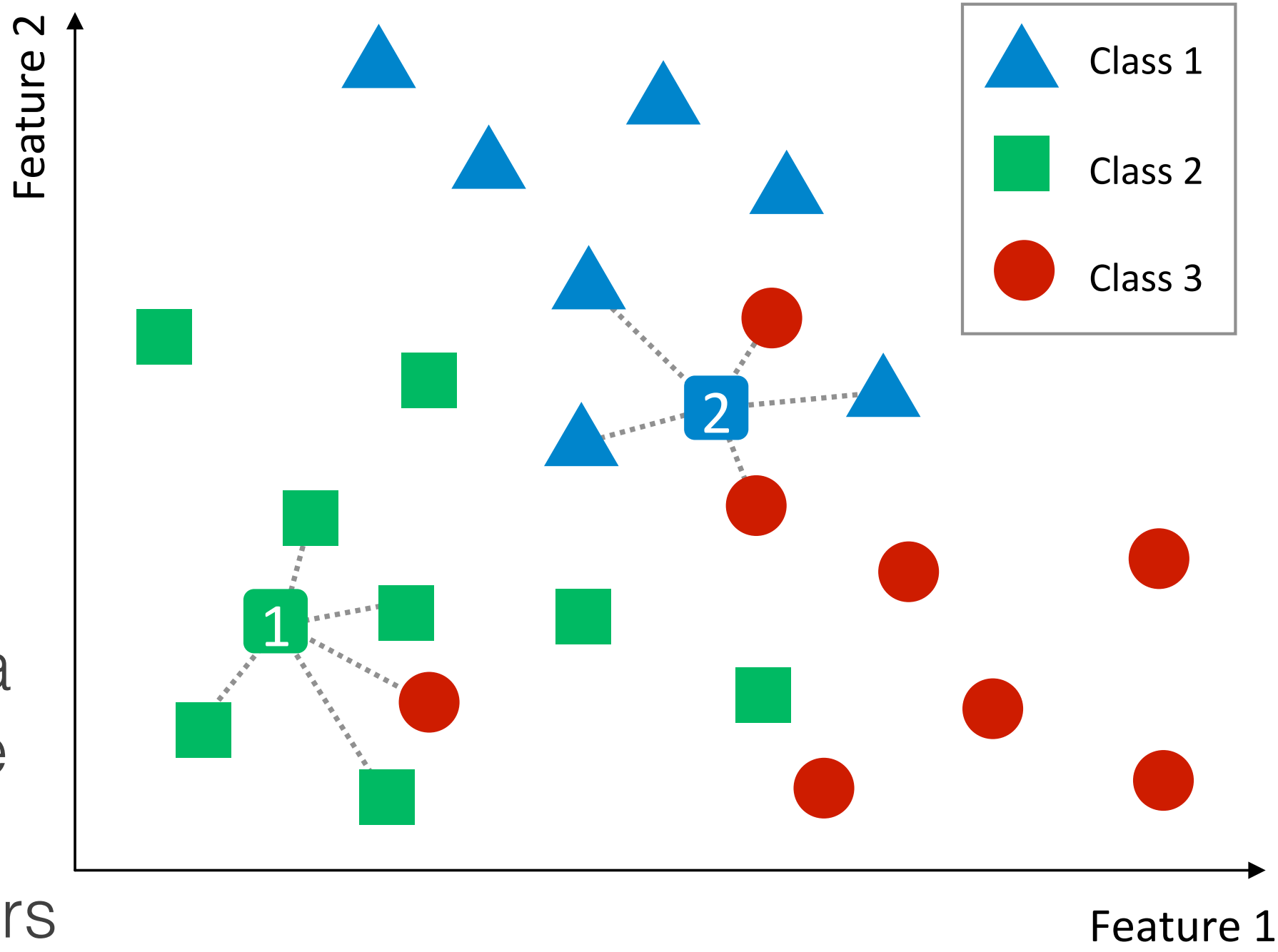
# K Nearest Neighbor Classifier

**Step 2**:
Place new (unseen) examples in the feature space

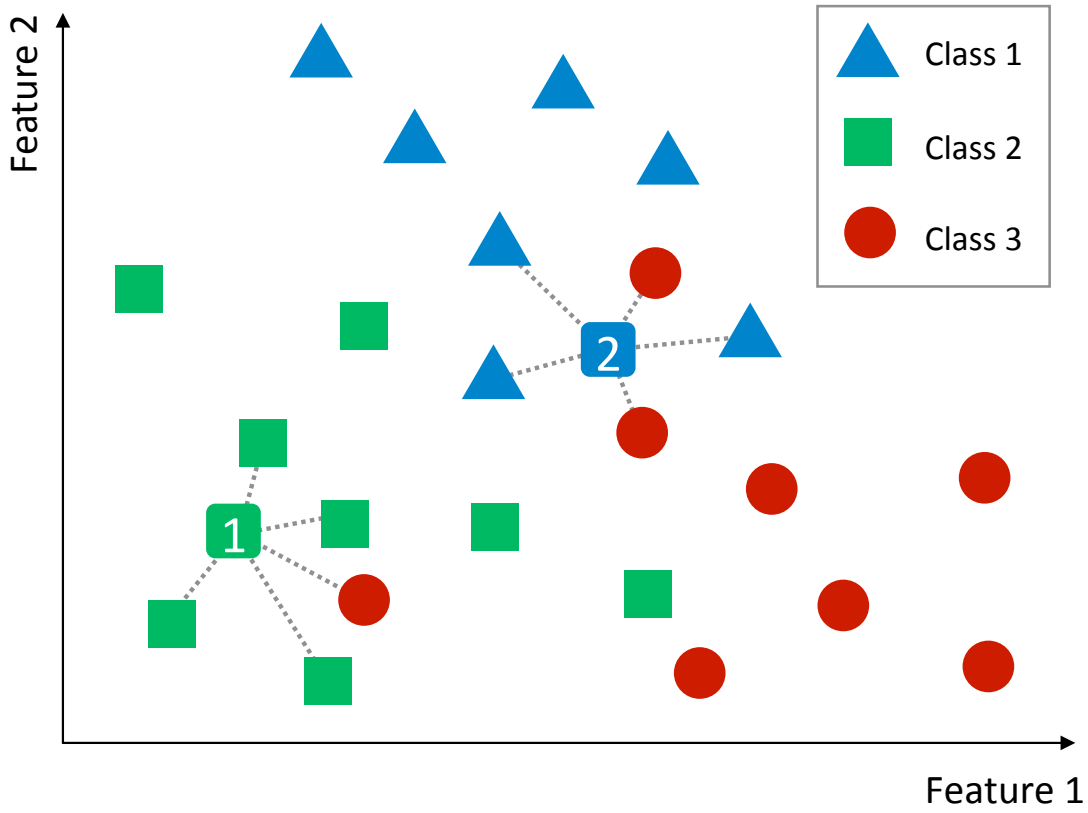


# K Nearest Neighbor Classifier

**Step 3**:
Classify the data by assigning the class of the k nearest neighbors



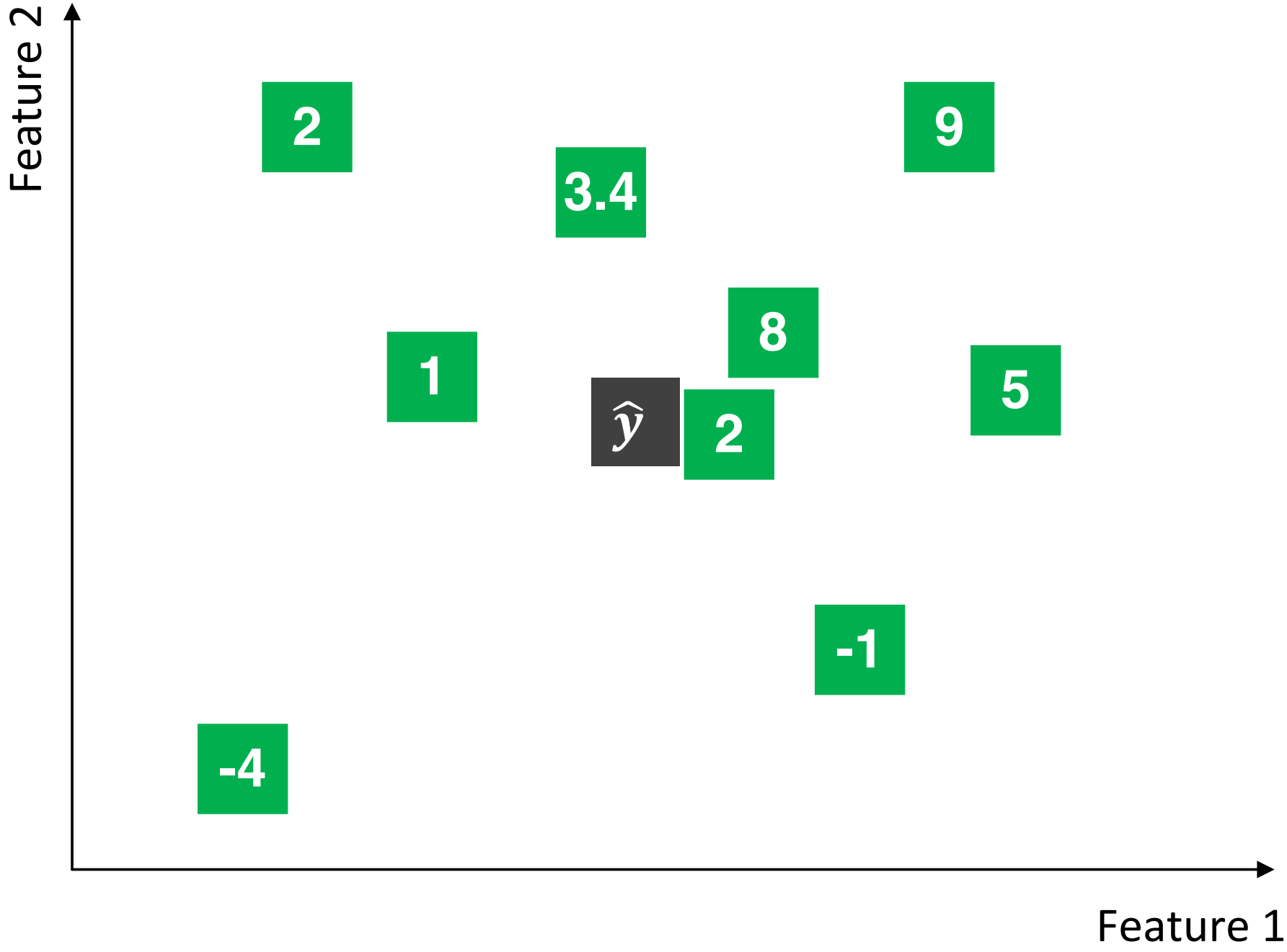
# K Nearest Neighbor Classifier




**Score vs Decision** :
For 5-NN, the confidence score that a sample belongs to a class could be: {0,1/5,2/5,3/5,4/5,1}

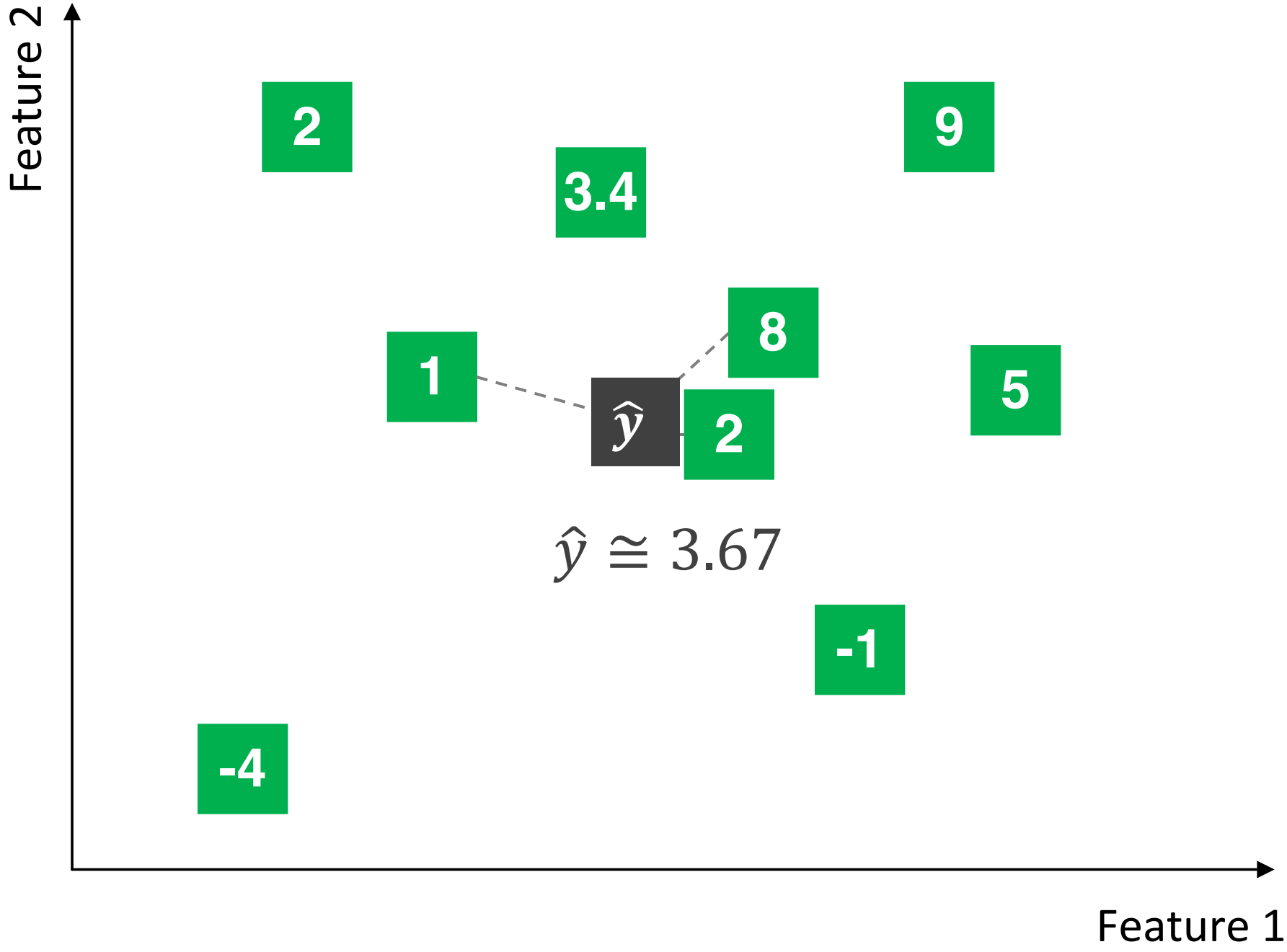
**Decision Rule**:
If the confidence score for a class > threshold, predict that class

# K Nearest Neighbor Regression



# K Nearest Neighbor Regression

$$\hat{y} = \frac{1}{k} \sum_{y_i \in \{\text{k nearest}\}} y_i$$



# KNN Pros and Cons

## Pros

- Simple to implement and interpret
- Minimal training time
- Naturally handles multiclass data

## Cons

- Computational expensive to find nearest  neighbors
- Requires all of the training data to be stored in the model
- Suffers if classes are imbalanced
- Performance may suffer in high dimensions

# How flexible should my model be?

the bias-variance tradeoff and learning to generalize

**bias**
consistently incorrect prediction

error from poor model assumptions
(high bias results in underfit)

**variance**
inconsistent prediction

error from sensitivity to small changes in the training data
(high variance results in overfit)

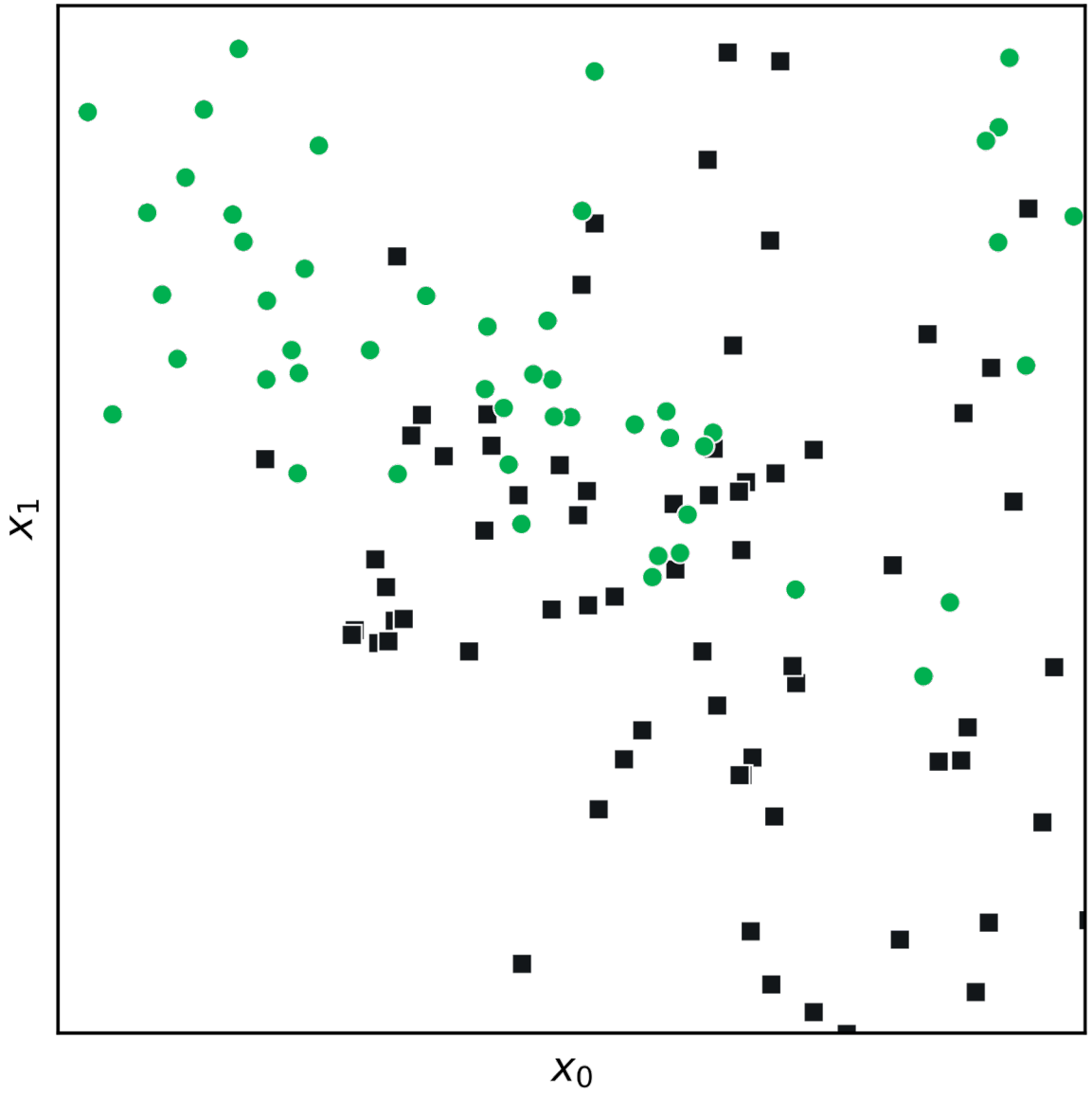
**noise**
lower bound on generalization error

irreducible error inherent to the problem
(e.g. you cannot predict the outcome of a flip of a fair coin any more than 50% of the time)

# Bias-Variance Tradeoff

$$\text{generalization error} = \text{bias}^2 + \text{variance} + \text{noise}$$

# Classification feature space

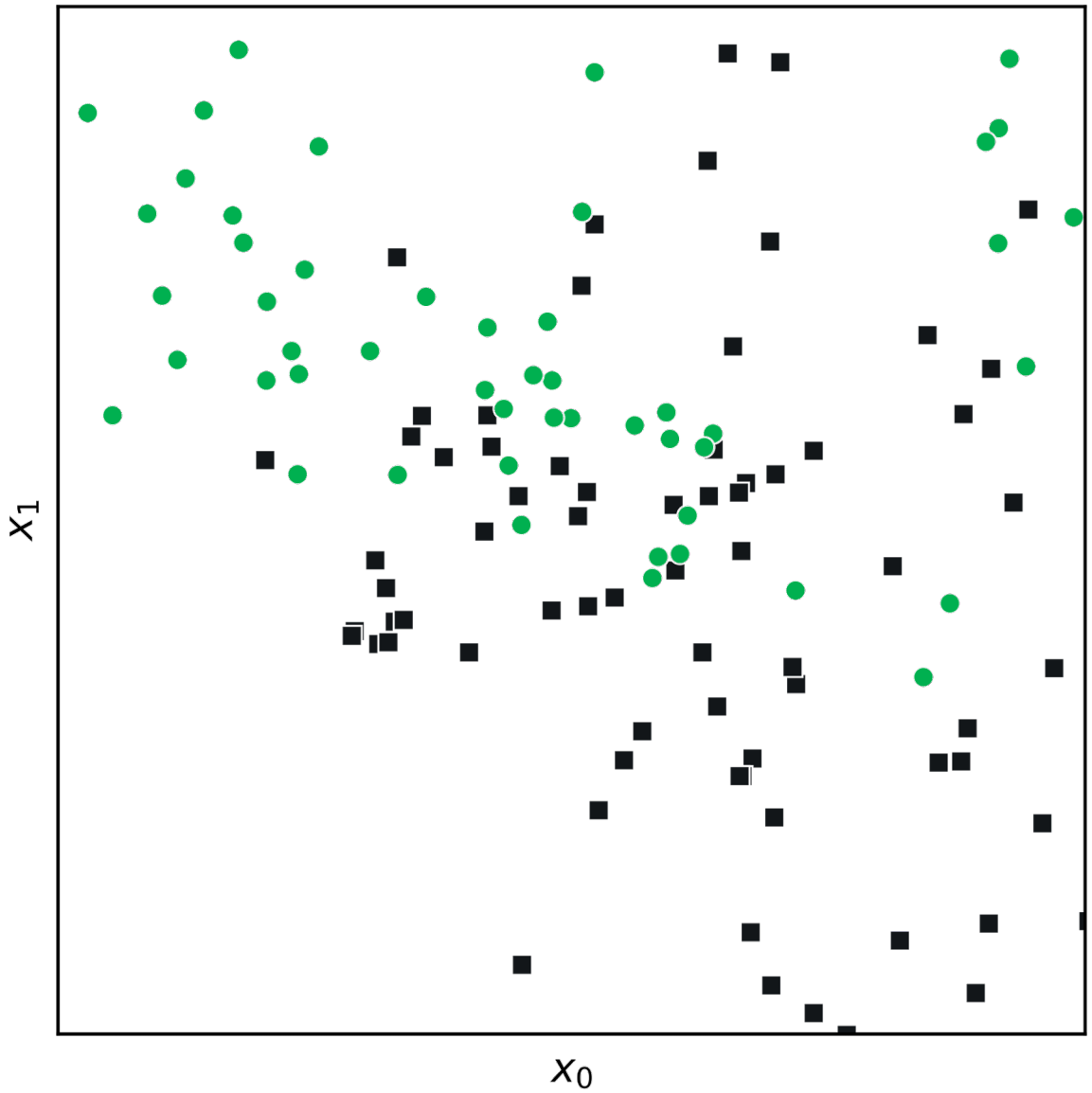


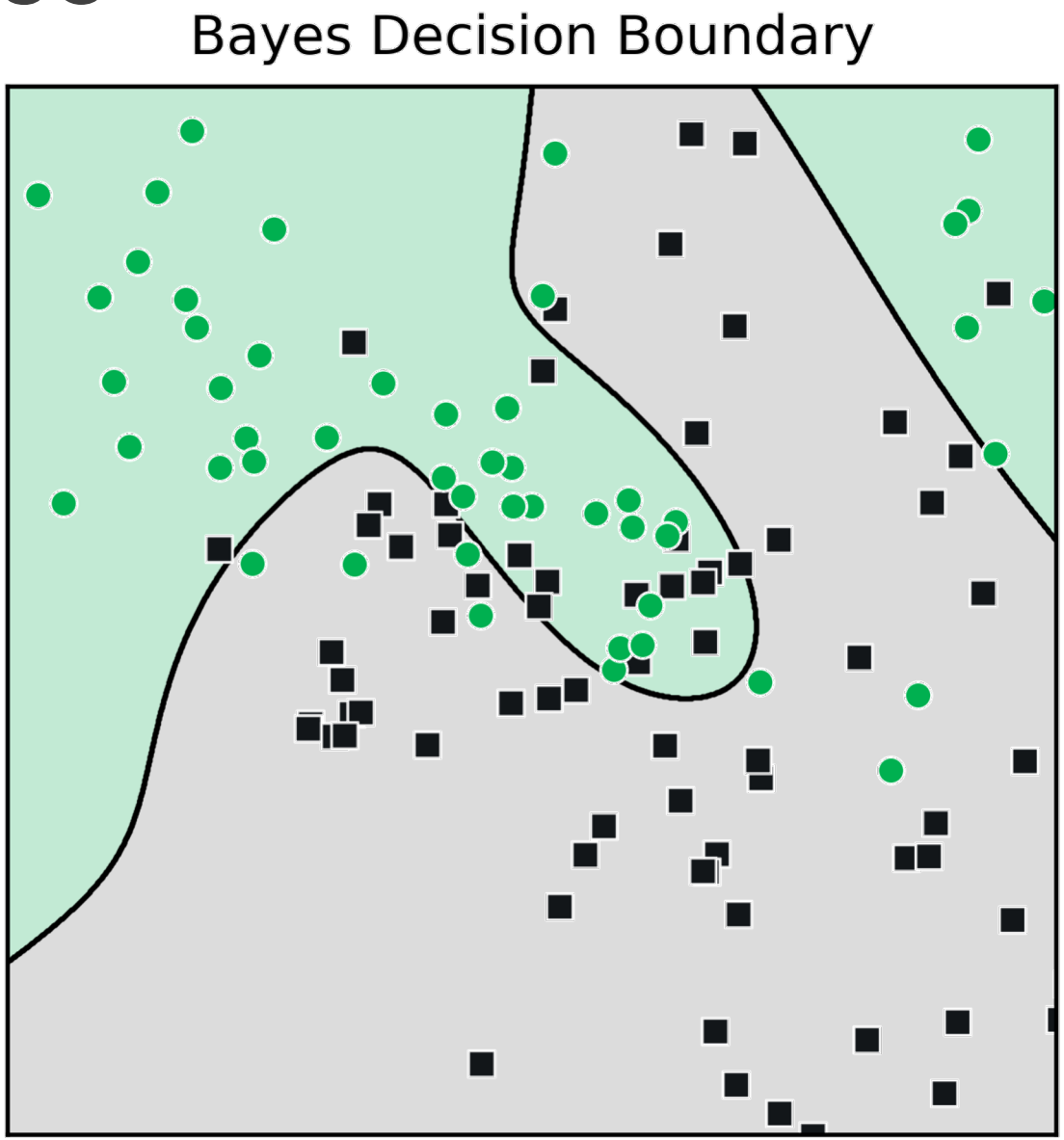
# What's the best we can do for binary classification?

If we know the probability distribution of the data

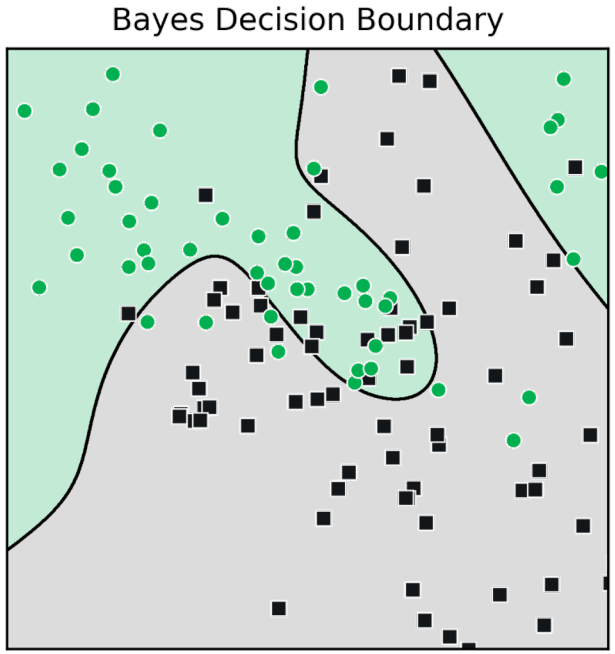
The Bayes decision rule

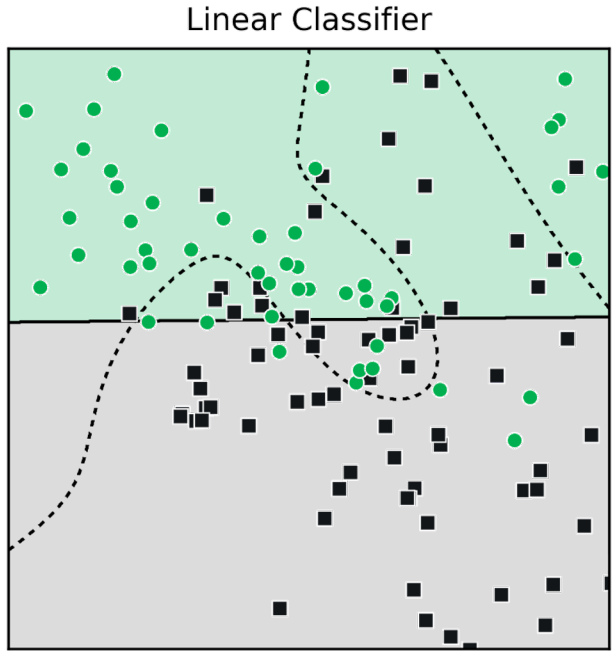
# Classification feature space

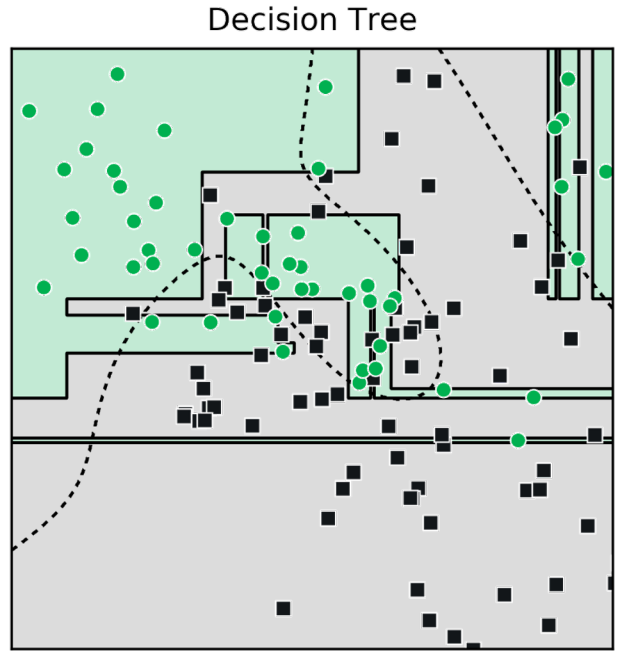


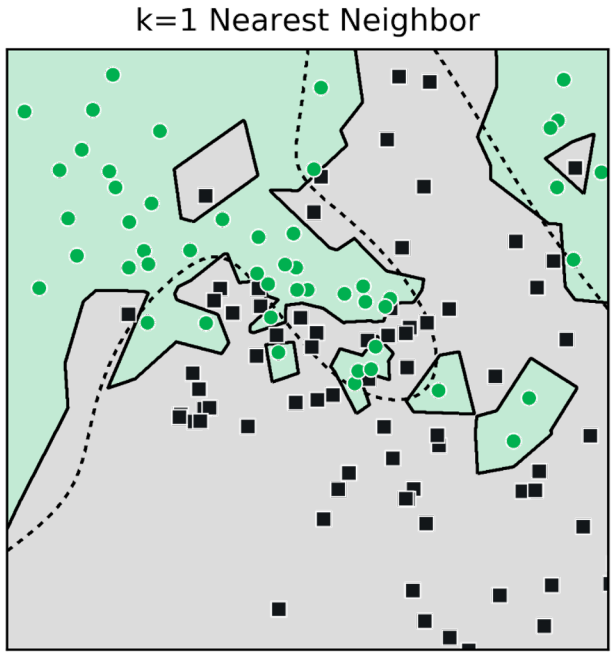



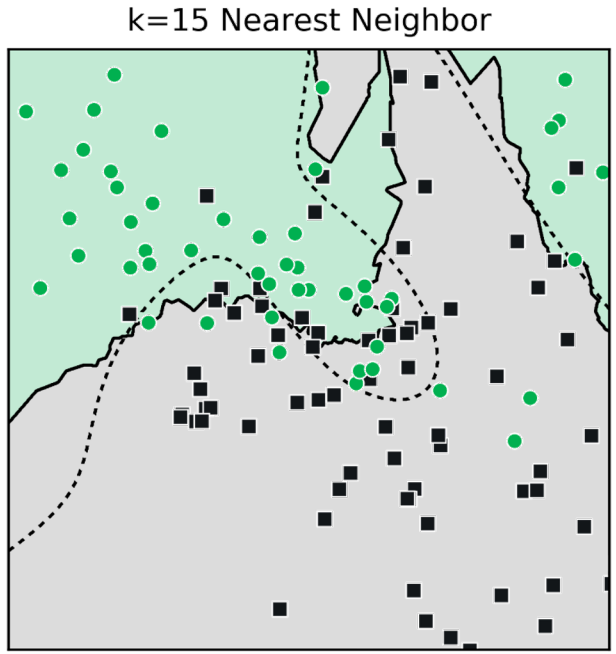
# Decision Boundary Examples

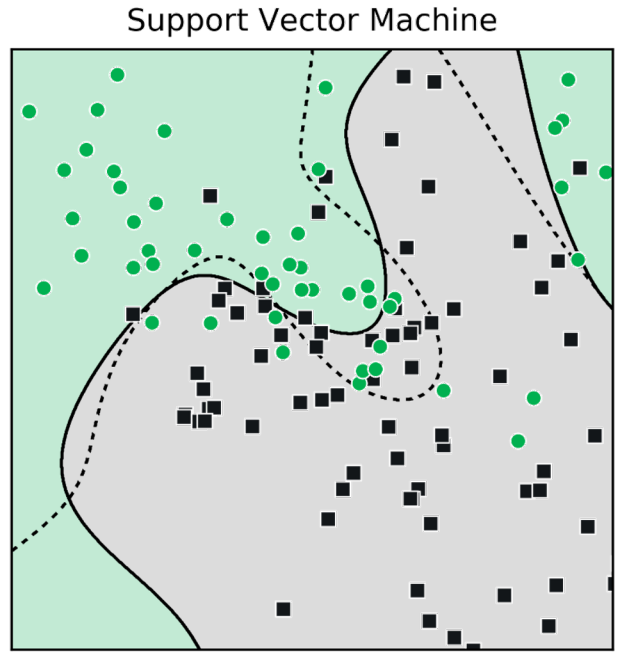


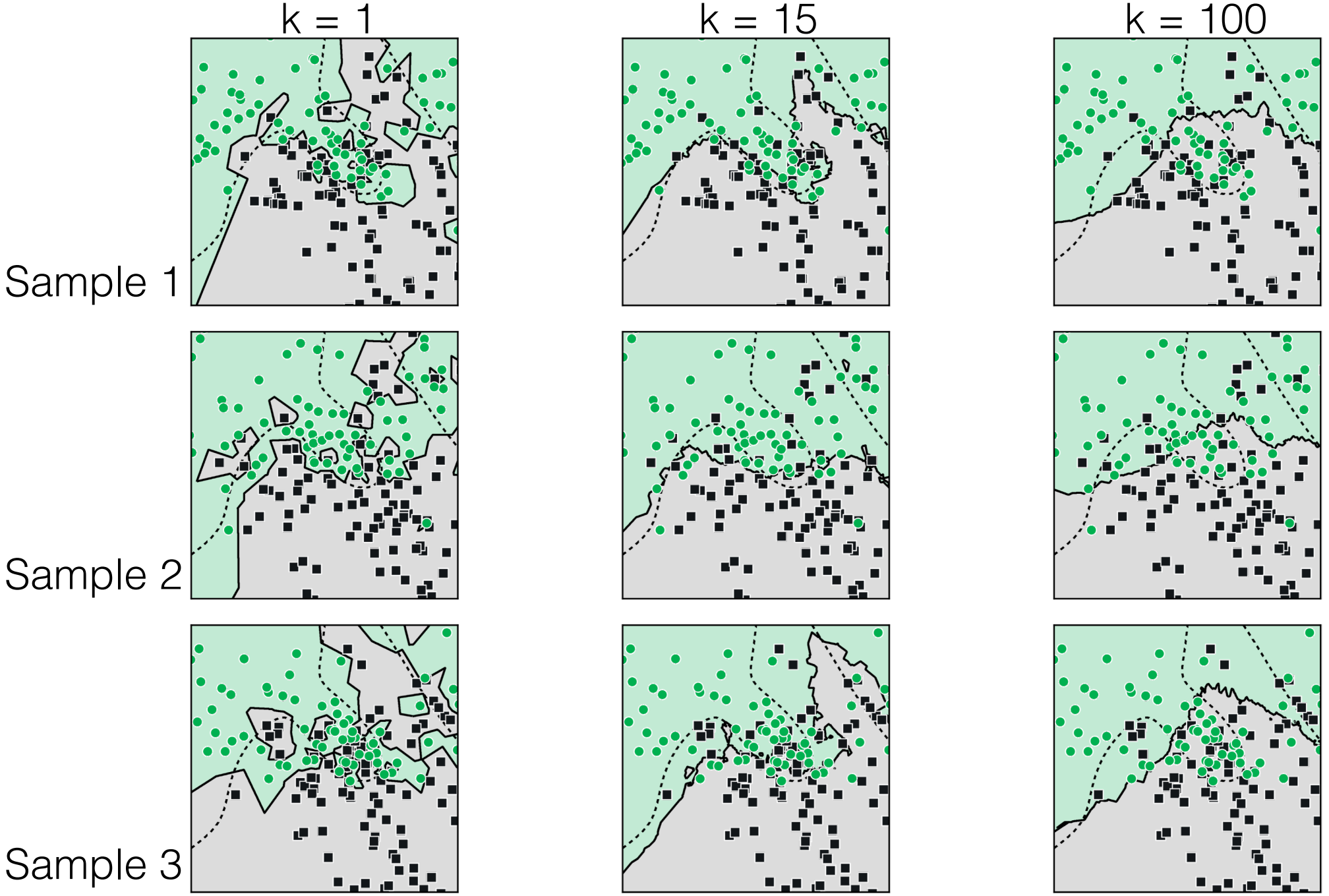



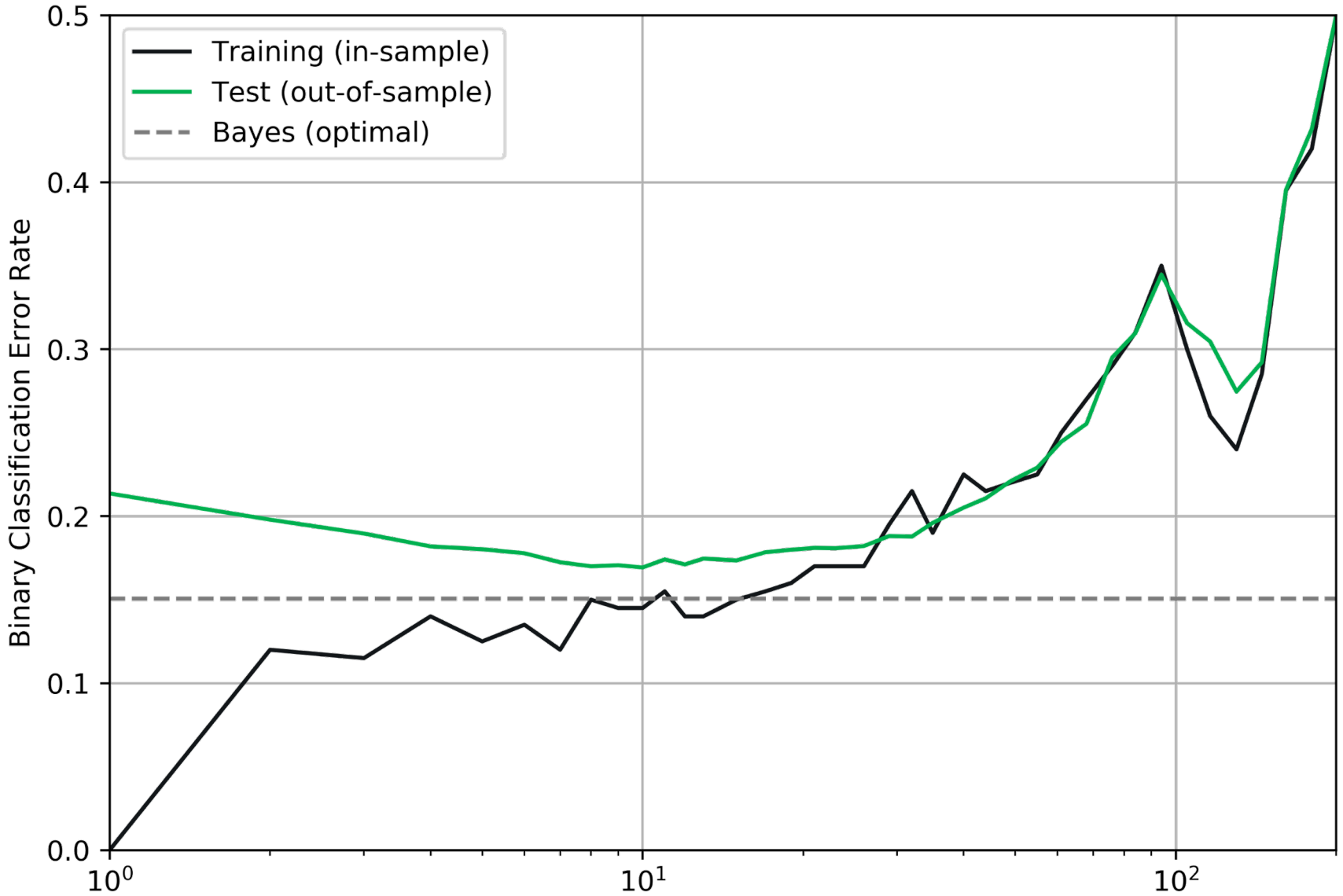



# Bias Variance Tradeoff



# Bias Variance Tradeoff




Overfit
High Variance
Low Bias

more flexible

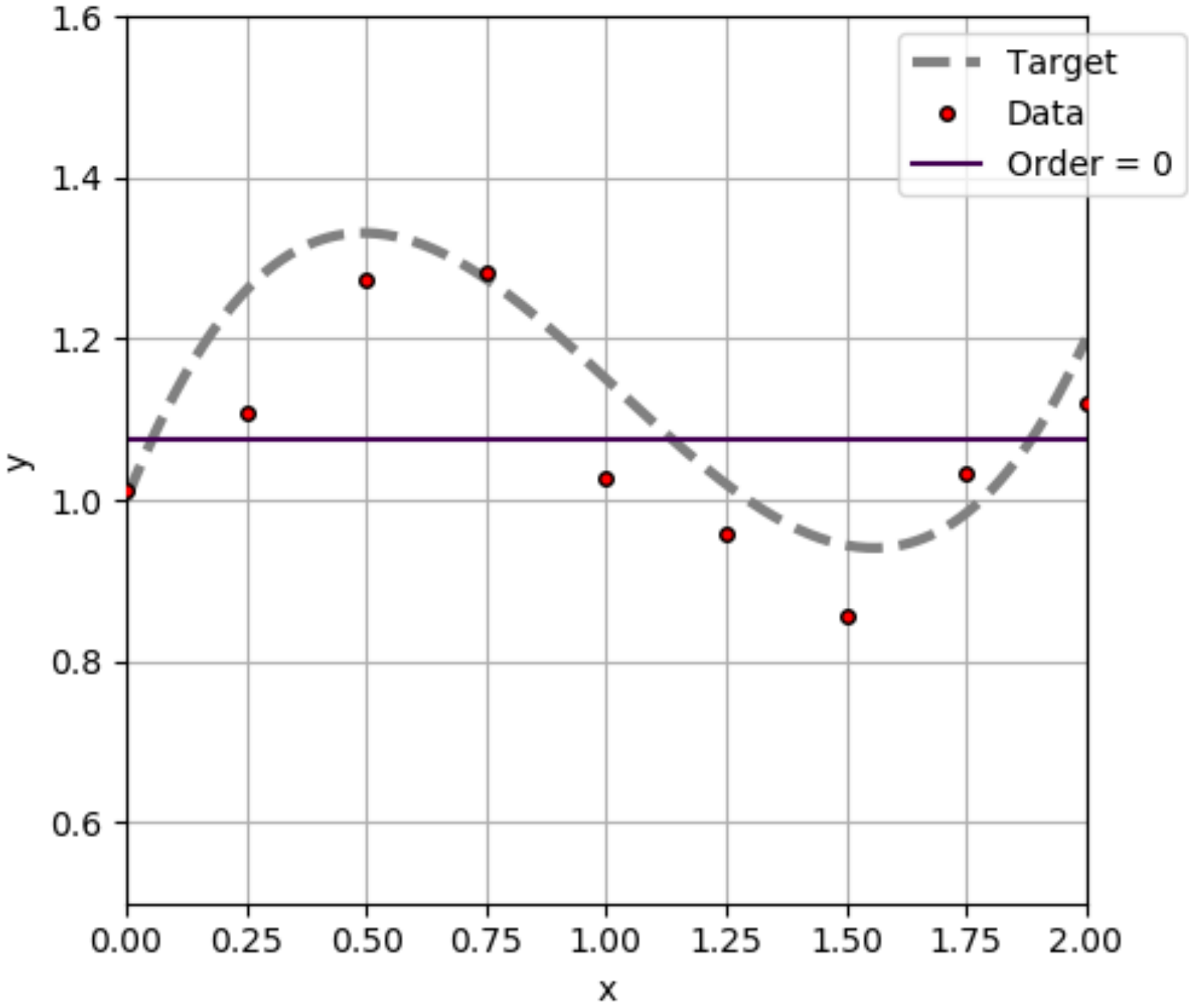
Underfit
Low Variance
High Bias

# This tradeoff is equally challenging for regression

# Linear Regression

$$\hat{y}_i = \sum_{j=0}^{N} a_j x_i^j$$

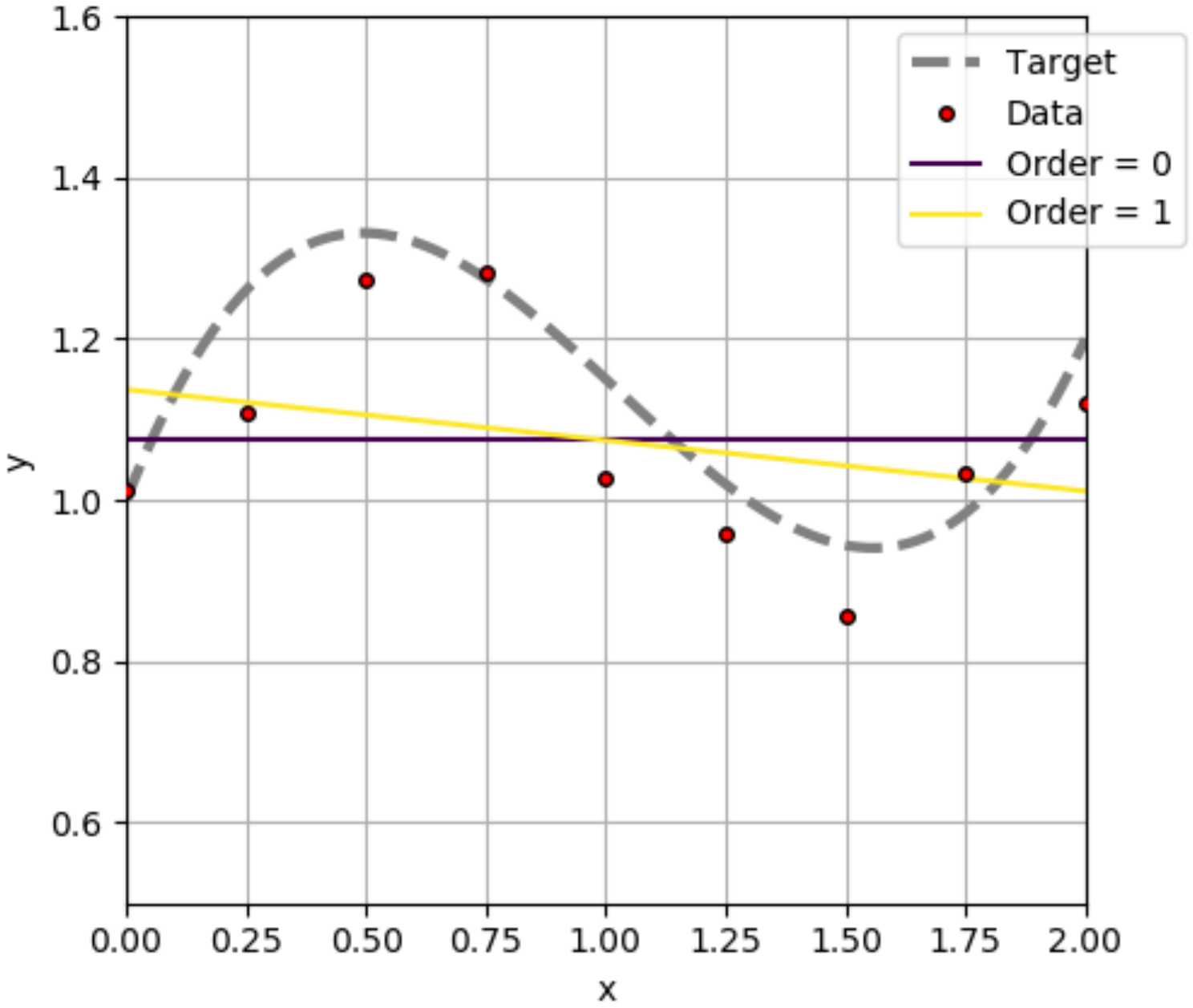
N is the model order



# Linear Regression

$$\hat{y}_i = \sum_{j=0}^{N} a_j x_i^j$$

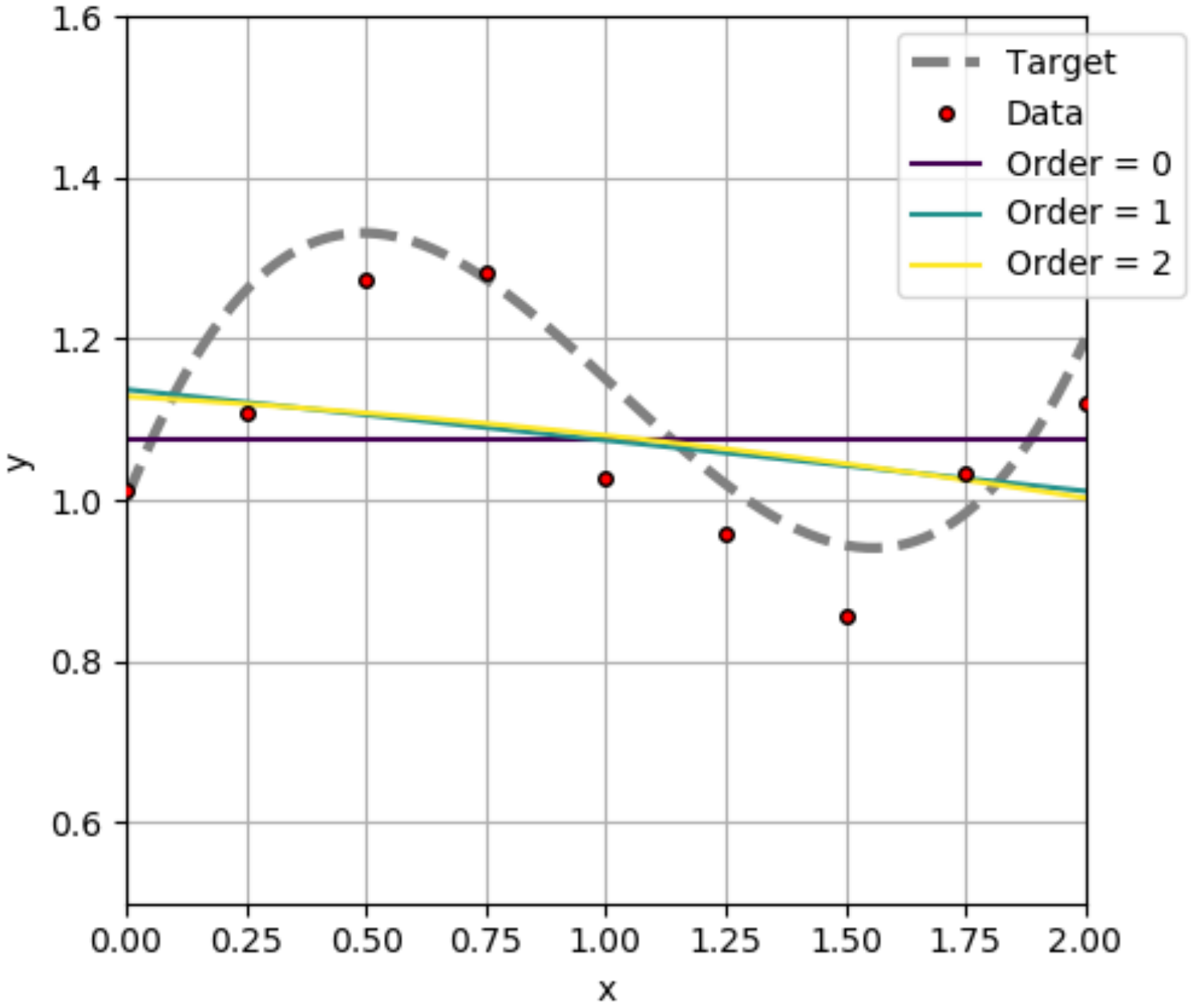
N is the model order



# Linear Regression

$$\hat{y}_i = \sum_{j=0}^{N} a_j x_i^j$$

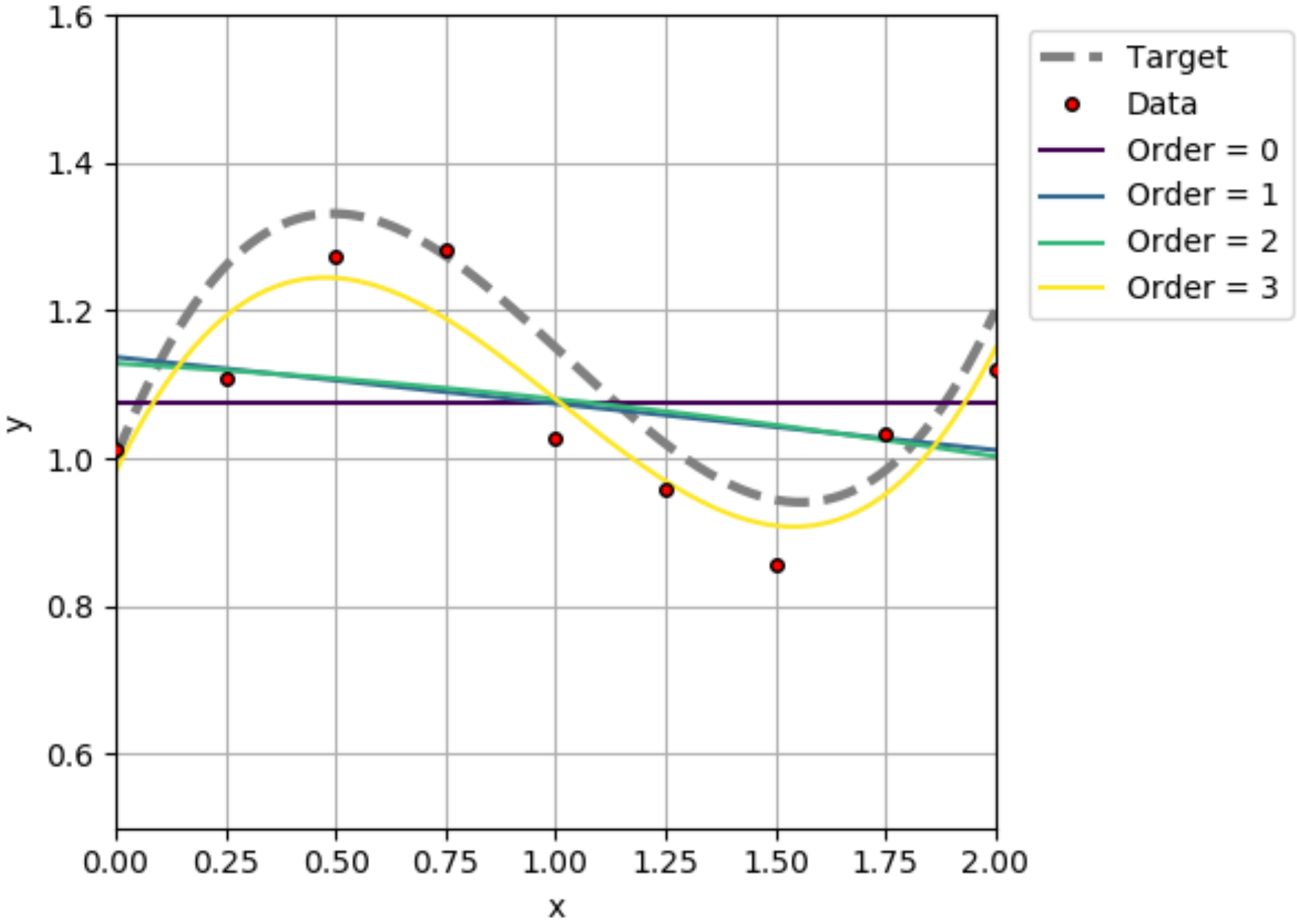
N is the model order



# Linear Regression

$$\hat{y}_i = \sum_{j=0}^{N} a_j x_i^j$$

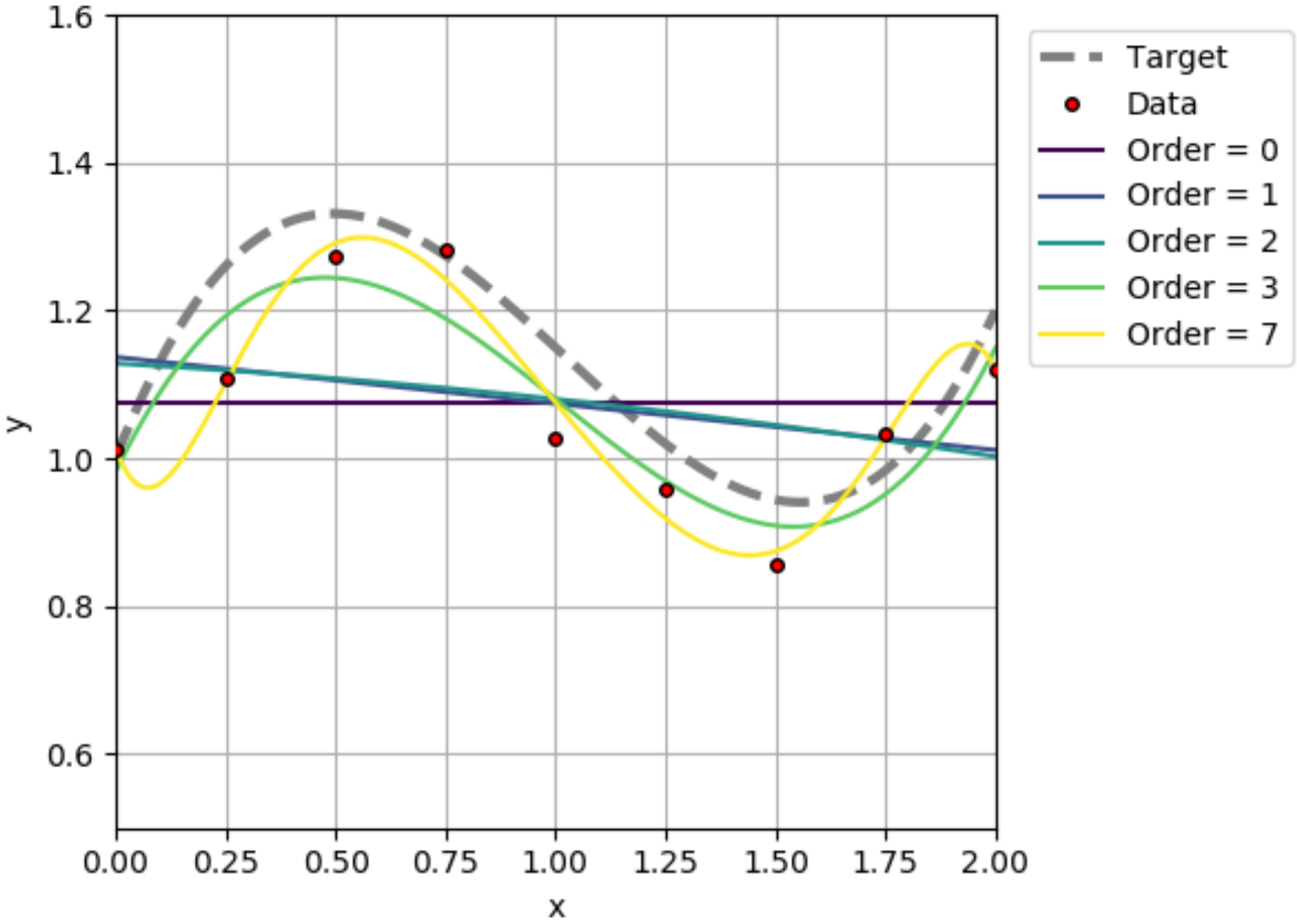
N is the model order



# Linear Regression

$$\hat{y}_i = \sum_{j=0}^{N} a_j x_i^j$$

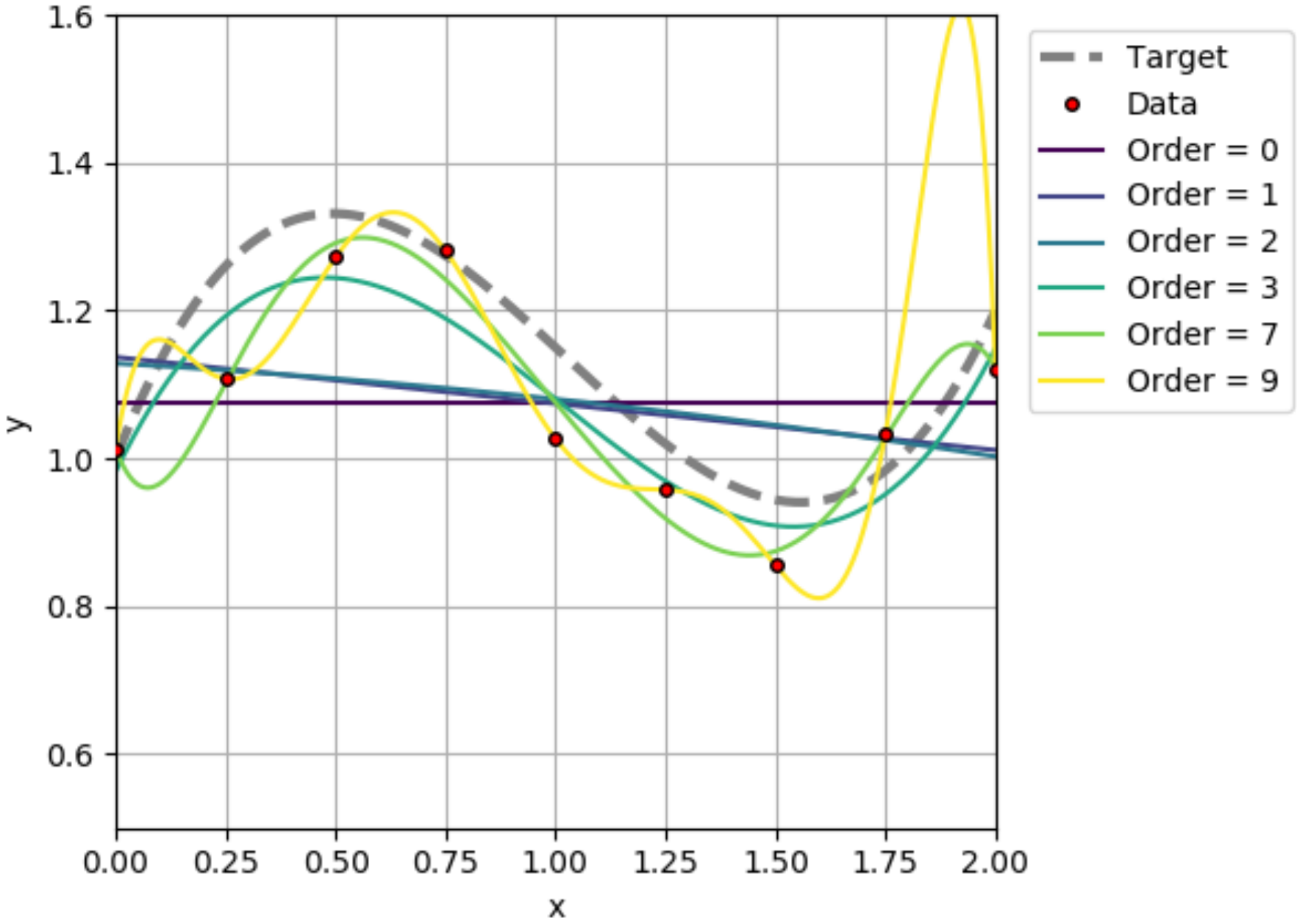
N is the model order



# Linear Regression

$$\hat{y}_i = \sum_{j=0}^{N} a_j x_i^j$$

N is the model order



# Problem

Too much flexibility leads to **overfit**

Too little flexibility leads to **underfit**

Over/underfit **hurts generalization** performance

# Solutions

1. Add more data for training

2. Constrain model flexibility through **regularization**

# Linear models

# Which of the following models are linear?

A $y = w_0$

B $y = w_0 + w_1 x_1$

C $y = w_0 + w_1 x_1 + w_2 x_2$

D $y = w_0 + w_1 x_1^2 + w_2 x_2^{0.4}$

E $y = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2$

F $y = w_0 + w_1 \int \sqrt[3]{x_1} dx_1 + w_2 g(x_2) + w_3 median(x_1, x_2, x_3)$

# Which of the following models are linear?

A $y = w_0$

B $y = w_0 + w_1 x_1$

C $y = w_0 + w_1 x_1 + w_2 x_2$

D $y = w_0 + w_1 x_1^2 + w_2 x_2^{0.4}$

E $y = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2$

F $y = w_0 + w_1 \int \sqrt[3]{x_1} dx_1 + w_2 g(x_2) + w_3 median(x_1, x_2, x_3)$

These are **ALL** linear in the **parameters, $w$**

# Linear models are linear in the parameters

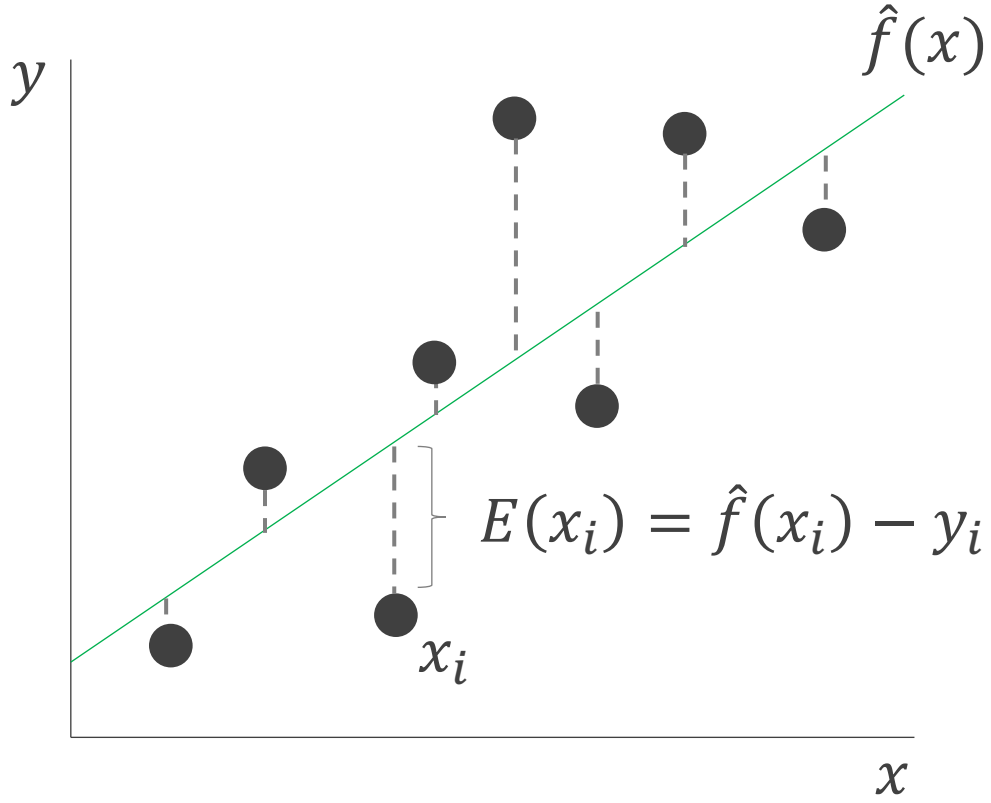
They often model nonlinear relationships between features and targets

# Types of Linear Regression

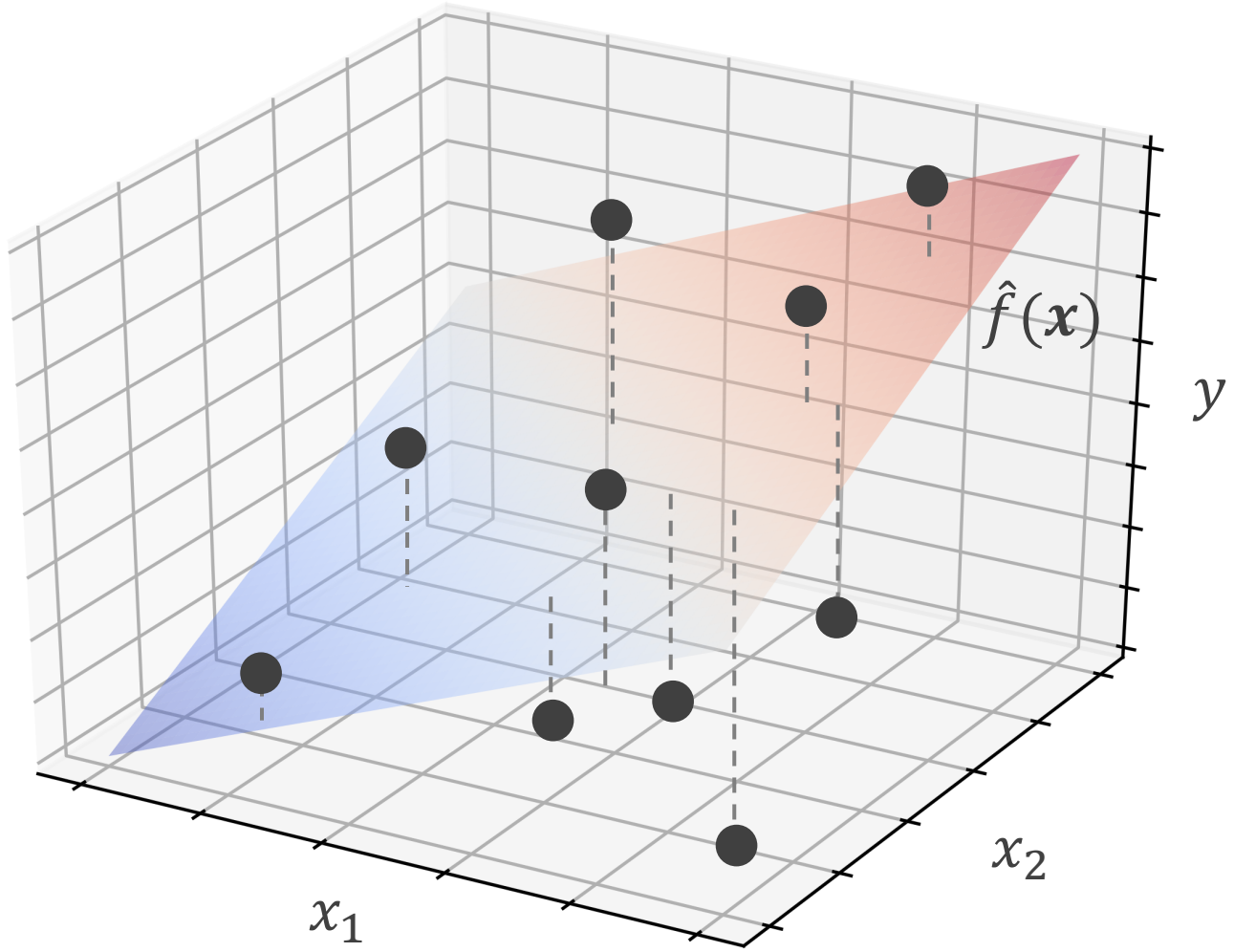
Here, $x_0 \triangleq 1$

| | **One** feature variable | **One or more** feature variables |
|---|---|---|
| **One** target variable | **Simple Linear Regression**<br>$y = w_0 + w_1 x_1$ | **Multiple Linear Regression**<br>$y = \sum_{i=0}^{p} w_i x_i$ or $y = \boldsymbol{w}^T \boldsymbol{x}$ |
| **One or more** target variables | **Multivariate (Multiple) Linear Regression**<br>$\boldsymbol{y} = \sum_{i=0}^{p} w_i \boldsymbol{x}_i$ or $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w}$ | |

# Linear models and error




**simple linear regression**




**multiple linear regression**

# How do we fit a linear model to data?

We want the error between our estimates and predictions to be small

# How do we measure error?

How well does $\hat{y} = \hat{f}(\boldsymbol{x}) = \sum_{i=0}^{p} w_i x_i$ approximate $y$ ?

Error: difference between our estimate $\hat{y}$ and our training data $y$

$$\text{error} = \hat{y} - y$$

We use mean squared error to quantify training (in-sample) error:

**Training (in-sample) error:** $E_{in}(\hat{f}) = \frac{1}{N} \sum_{n=1}^{N} \left(\hat{f}(\boldsymbol{x}_n) - y_n\right)^2$

We call this our **Cost Function** (a.k.a. loss, error, or objective)

**Cost Function:** $$E_{in}(\hat{f}) = \frac{1}{N}\sum_{n=1}^{N}\left(\hat{f}(\boldsymbol{x}_n) - y_n\right)^2$$

Training error is a function of our **model** and the **training data**

We can't change the data, we must adjust our model to minimize cost

We choose model **parameters** that minimize cost

This is an **optimization** problem

# How to fit our model to the training data?

Equivalently: how do we choose $\boldsymbol{w}$ to minimize cost (error)

$$E_{in}(\hat{f}) = \frac{1}{N}\sum_{n=1}^{N}\left(\hat{f}(\boldsymbol{x}_n) - y_n\right)^2$$

where $\hat{f}(\boldsymbol{x}_n) = \boldsymbol{w}^T\boldsymbol{x}_n$

We want to minimize $E_{in}(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{w}^T\boldsymbol{x}_n - y_n)^2$

…by varying $\boldsymbol{w}$

How do we do that?