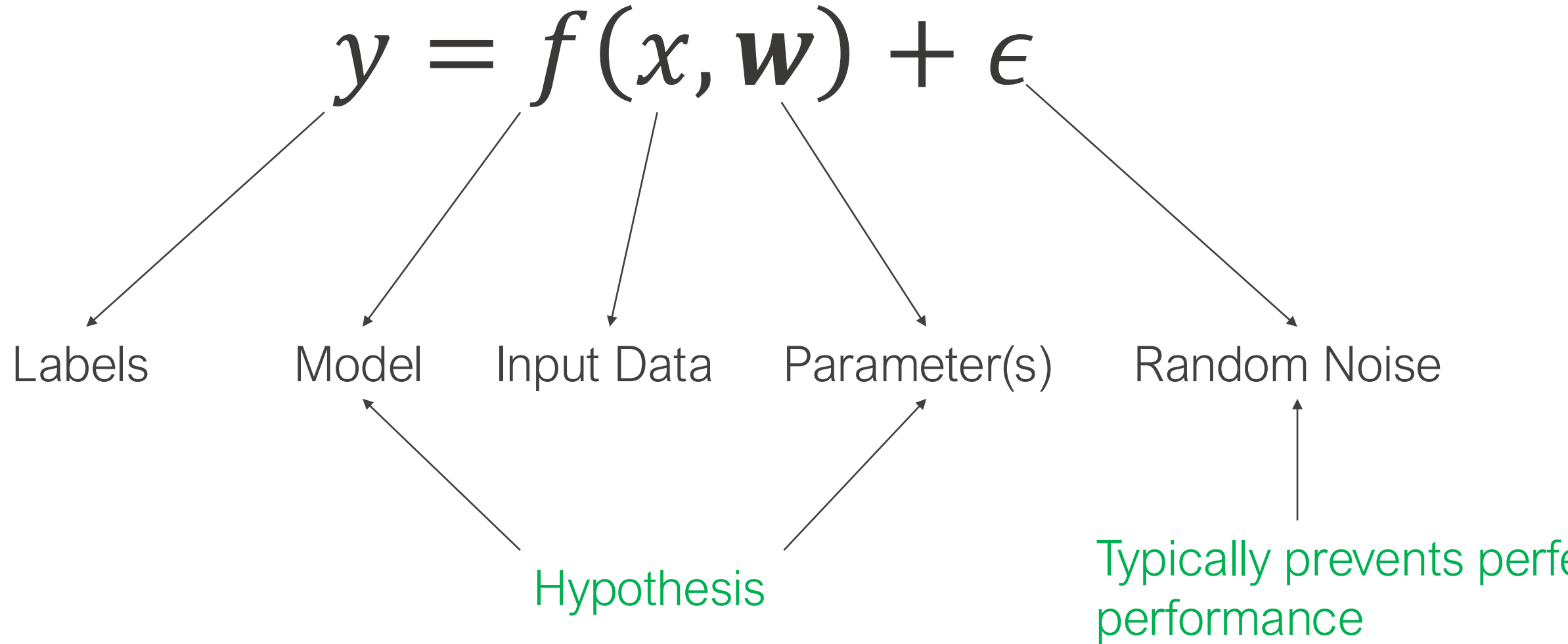


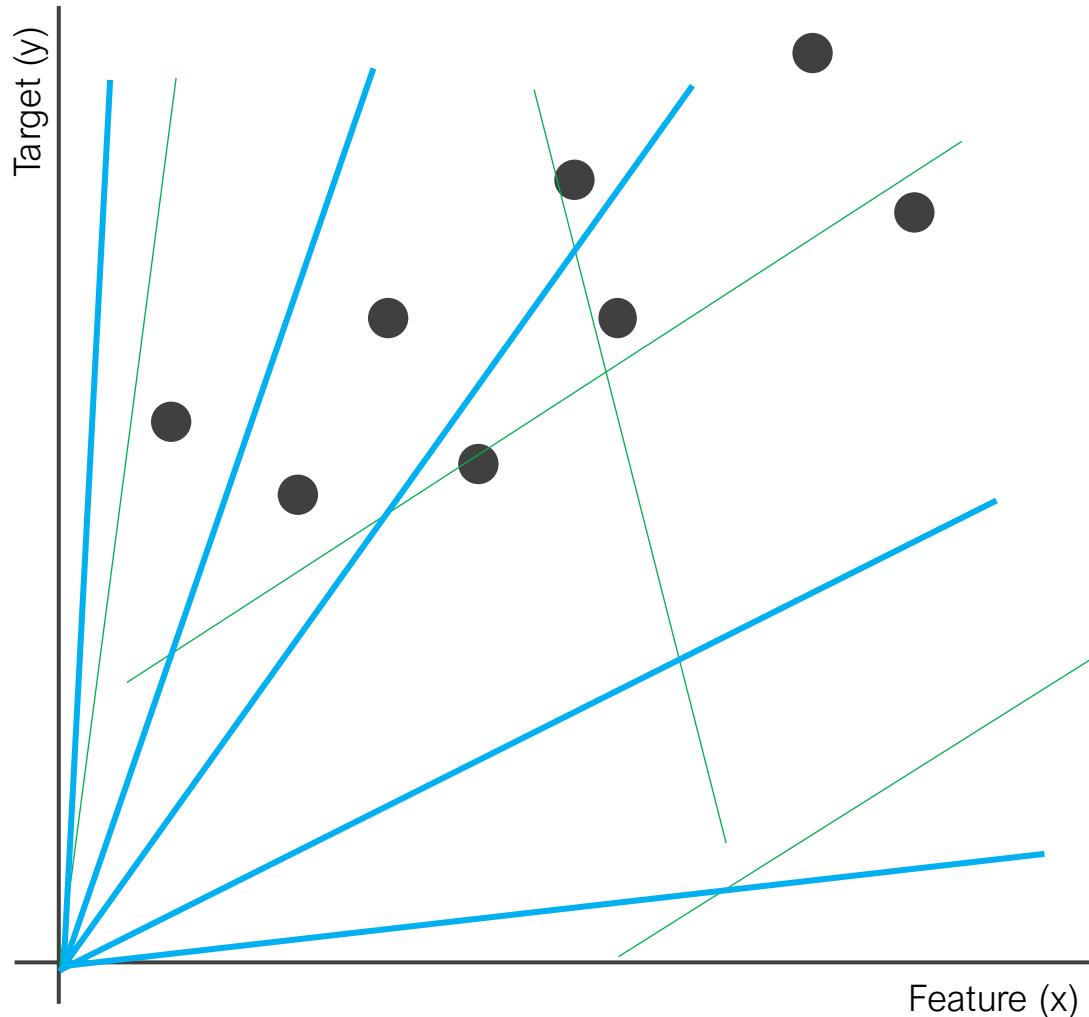
How flexible should my algorithms be?

Supervised machine learning model

We search for the model that best fits our data



Example: linear regression



Using any line as a hypothesis function, how many possible hypothesis functions are in the set?

Infinitely many

Using the line $y = wx$ as the family of hypothesis functions, how many possible hypothesis functions are in the set?

Infinitely many

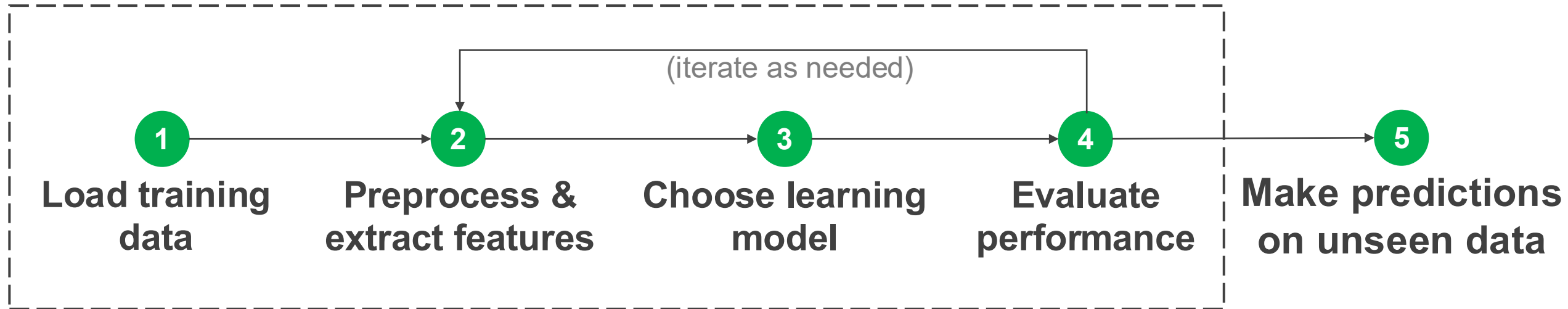
Which set contains the better hypothesis?


Which set has more options to consider?

What is our learning algorithm?

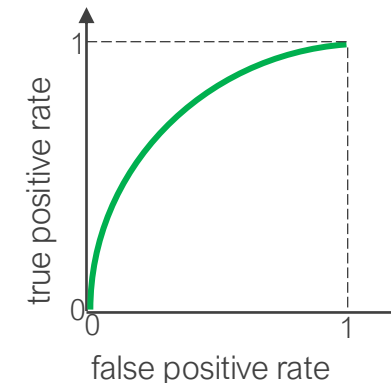
Algorithm Development

Application



y		X		X'			
1	x_1		x_1	0.38	0.39	0.85	0.78
1	x_2		x_2	0.81	0.91	0.97	0.53
0	x_3		x_3	0.65	0.59	0.91	0.11
0	x_4		x_4	0.94	0.05	0.40	0.26
1	x_5		x_5	0.27	0.19	0.03	0.64
0	x_6		x_6	0.02	0.98	0.36	0.11

linear discriminant
perceptron
logistic regression
decision trees
random forests
support vector machine
k nearest neighbors
neural networks



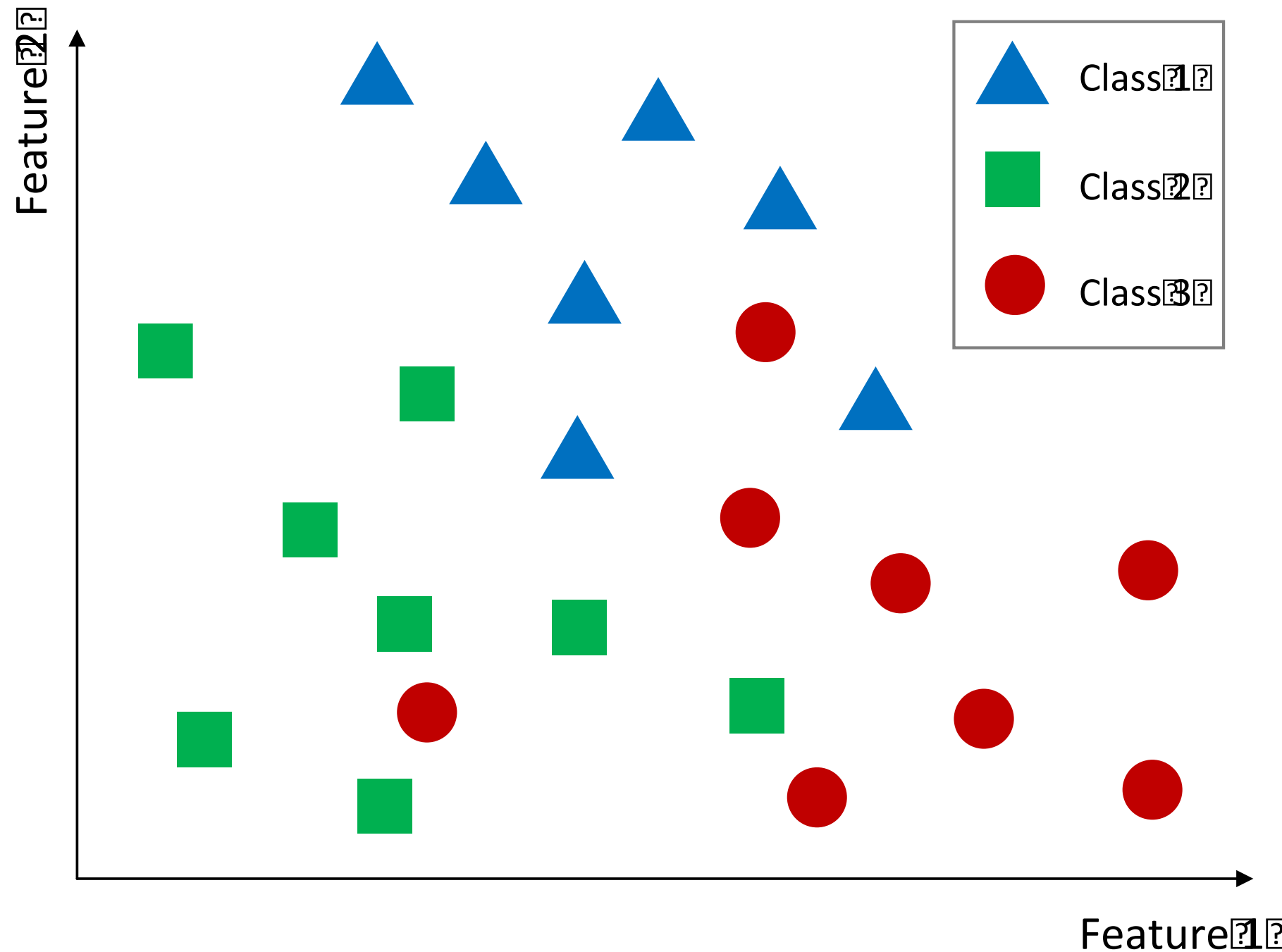
K-Nearest Neighbors

Classification and Regression

K Nearest Neighbor Classifier

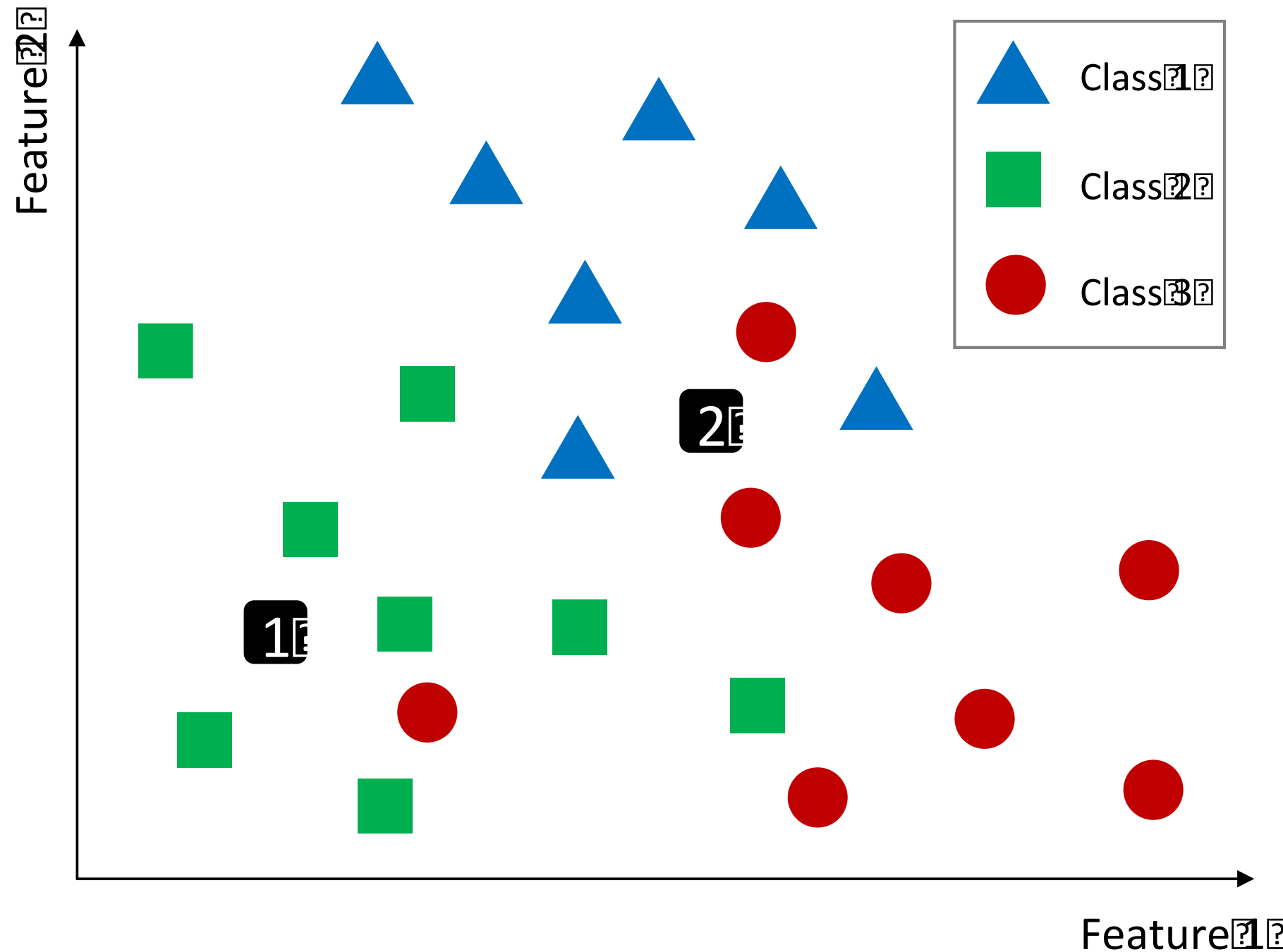
Step 1: Training

Every new data point is
a model parameter



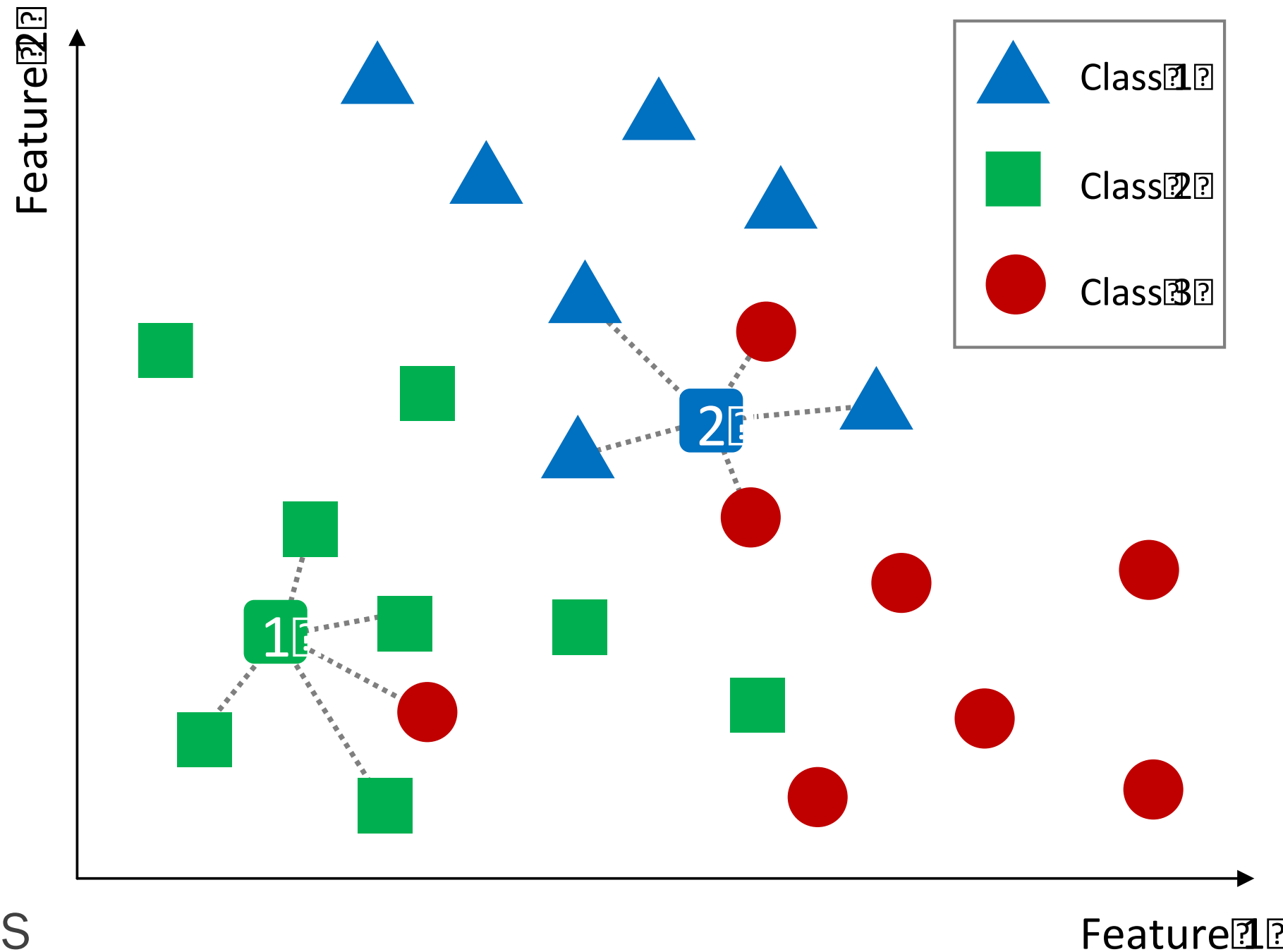
K Nearest Neighbor Classifier

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



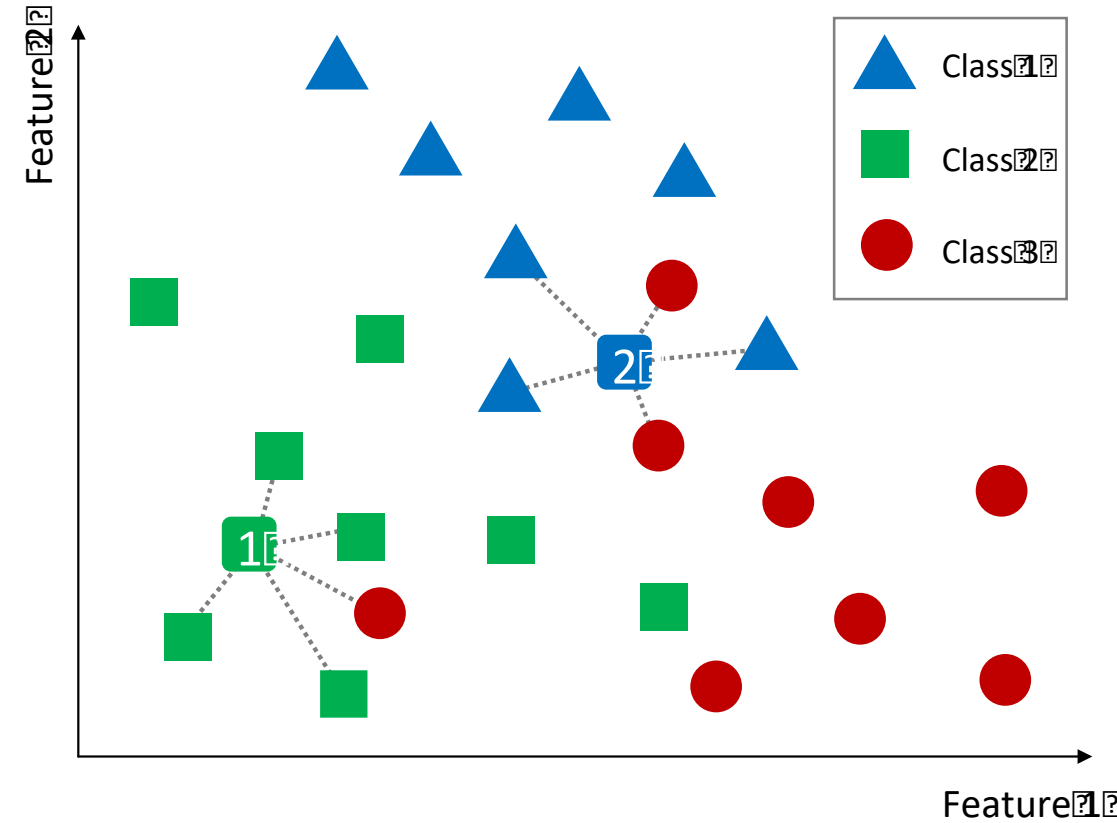
K Nearest Neighbor Classifier

Score vs Decision :

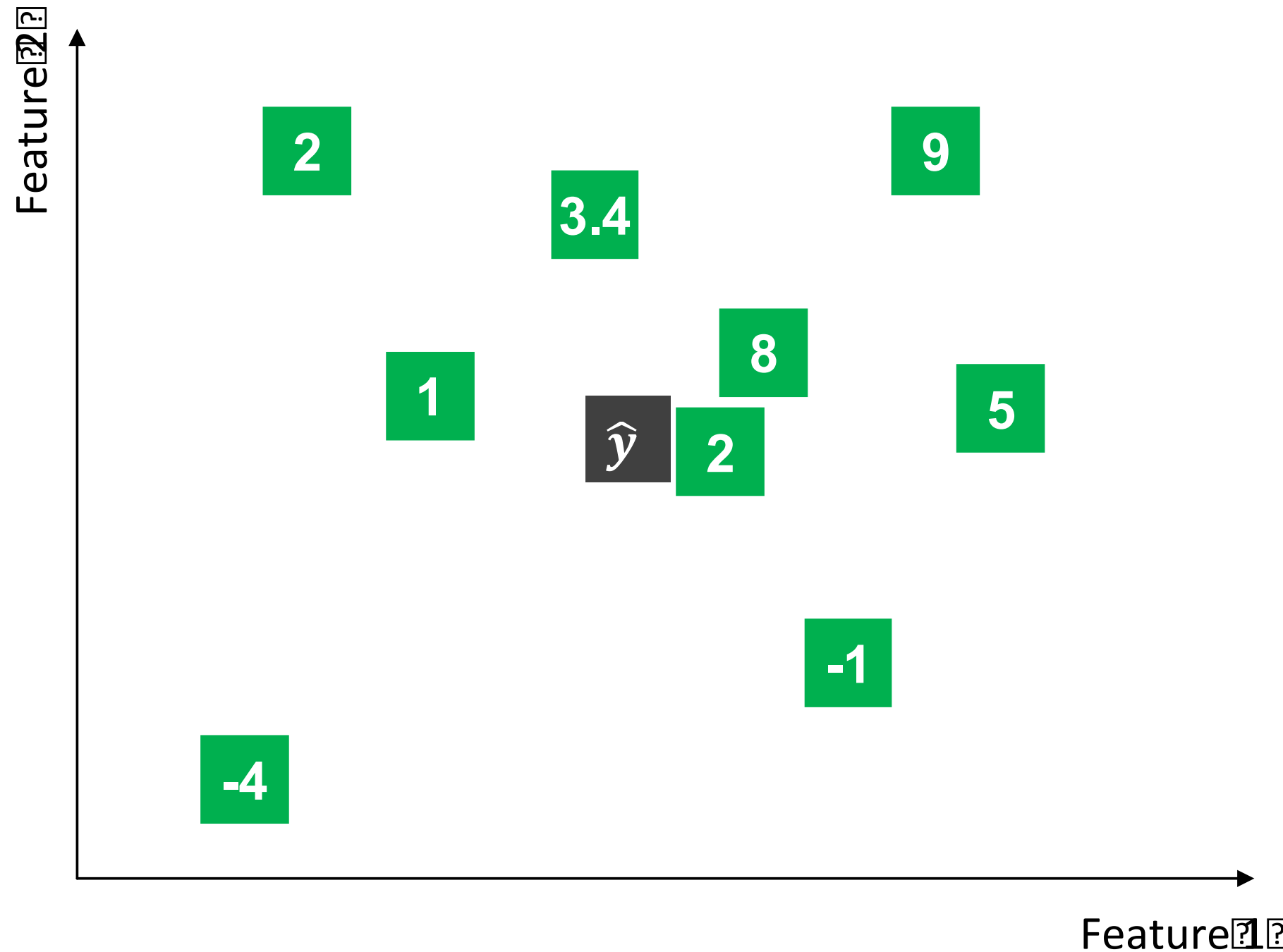
For 5-NN, the confidence score that a sample belongs to a class could be: $\{0, 1/5, 2/5, 3/5, 4/5, 1\}$

Decision Rule:

If the confidence score for a class $>$ threshold, predict that class

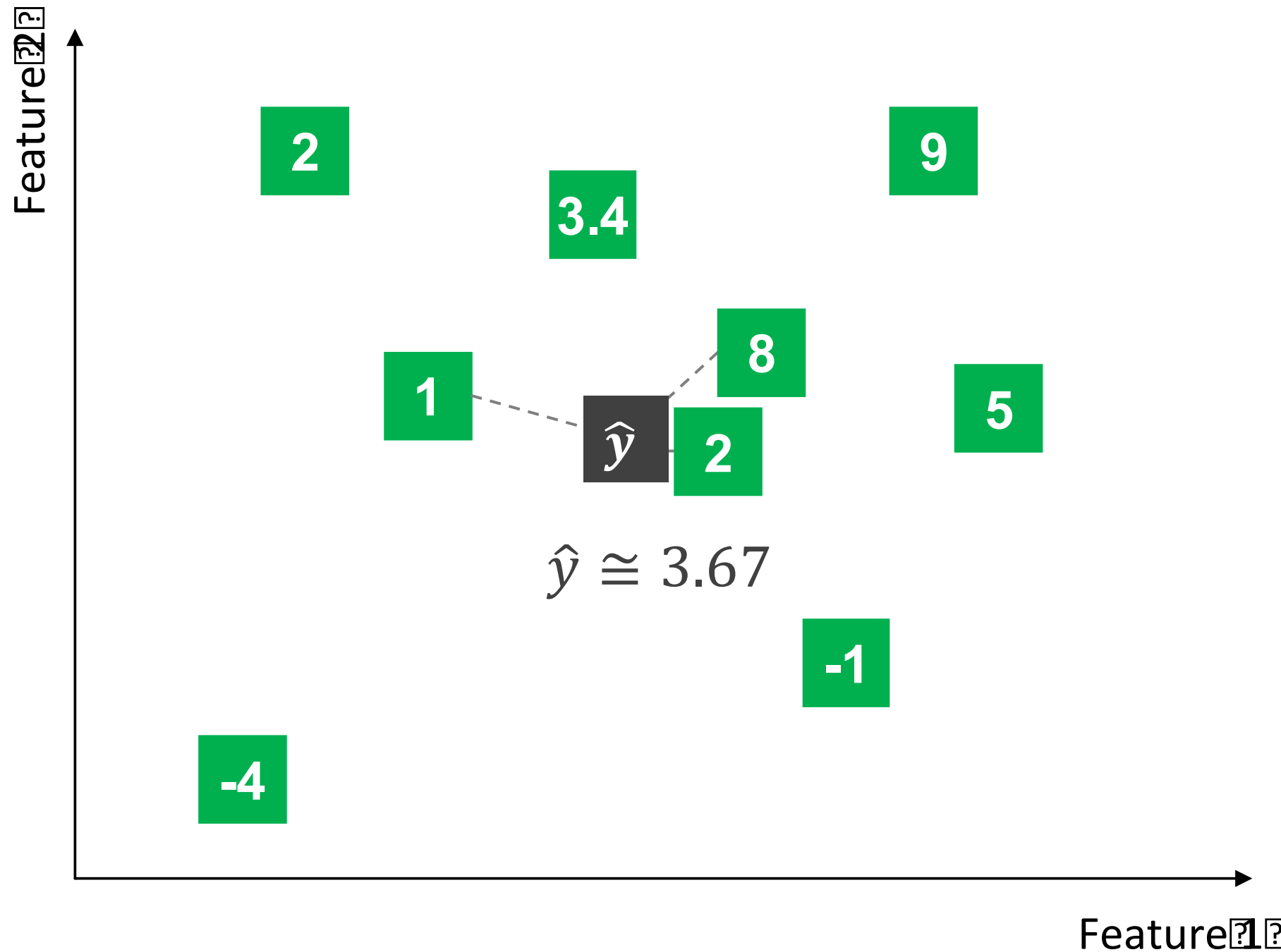


K Nearest Neighbor Regression



K Nearest Neighbor Regression

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



KNN Pros and Cons

Pros

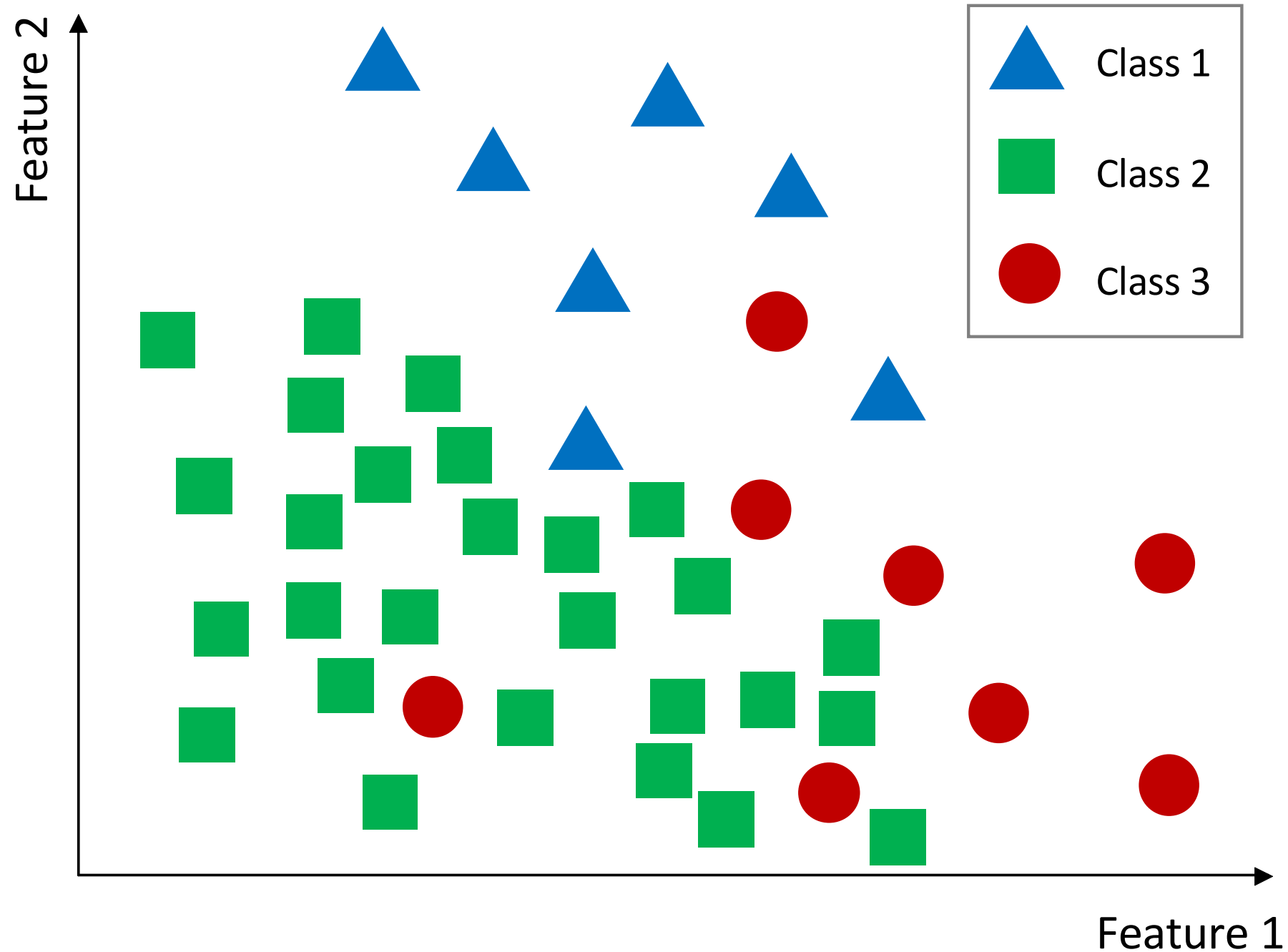
- Simple to implement and interpret
- Minimal training time
- Naturally handles multiclass data

Cons

- Computationally 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 for large datasets or in high dimensions

K Nearest Neighbor Classifier

What happens if the data aren't balanced?



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

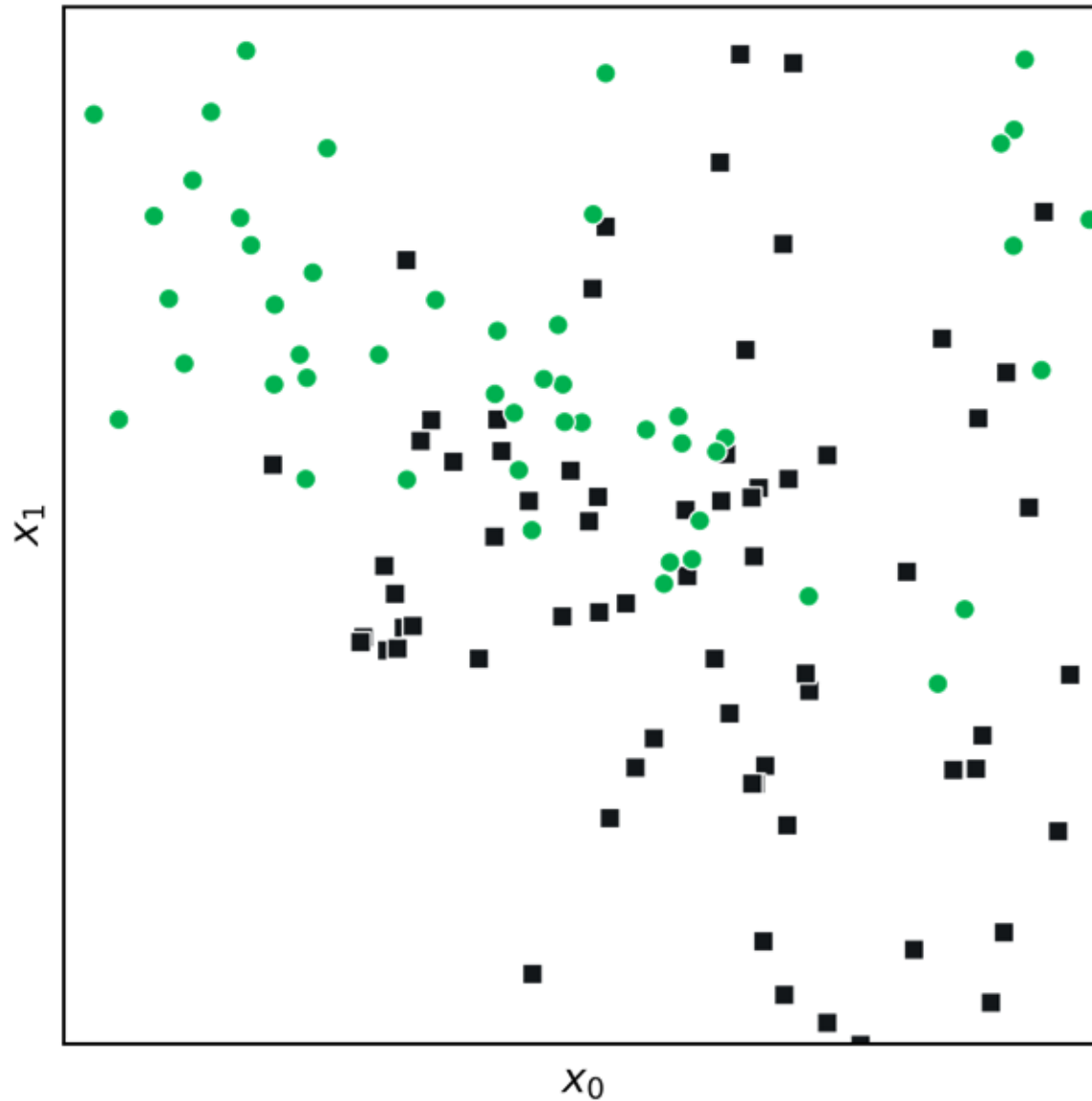
irreducible error inherent to the
problem

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

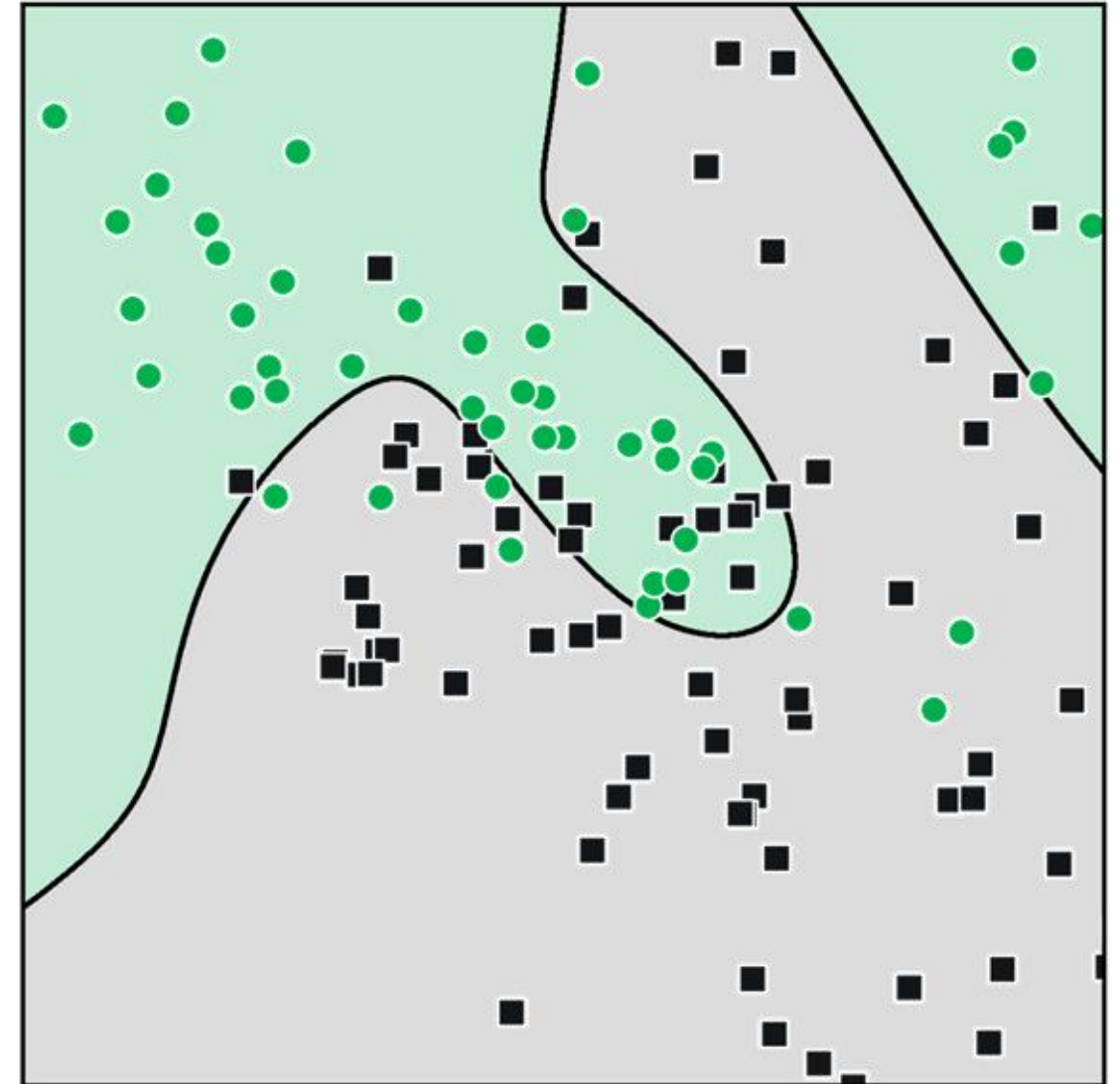
Bias-Variance Tradeoff

generalization error = bias² + variance + noise

Classification feature space

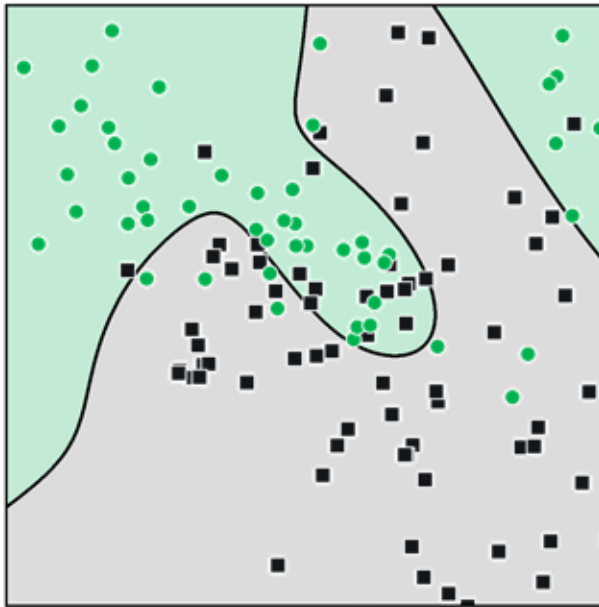


Bayes Decision Boundary

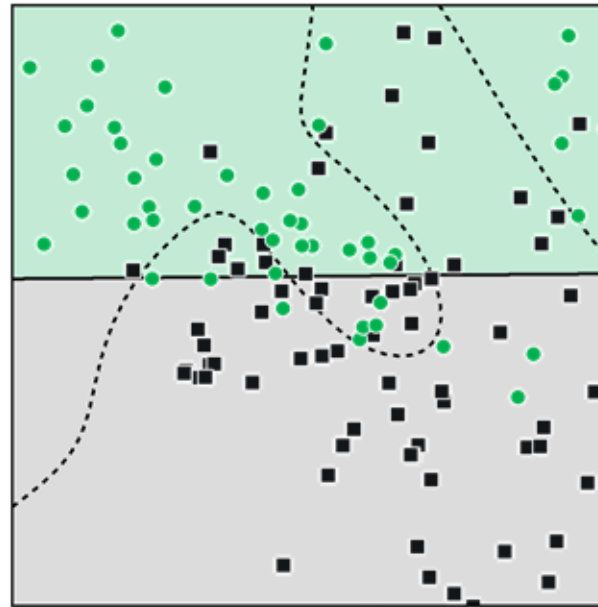


Decision Boundary Examples

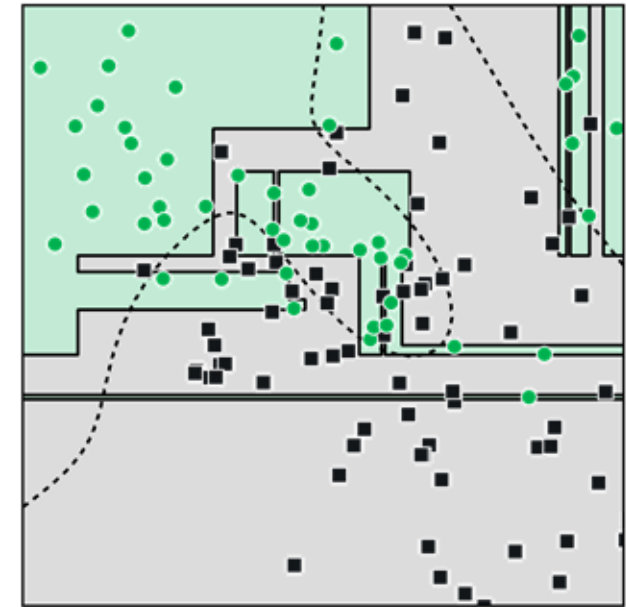
Bayes Decision Boundary



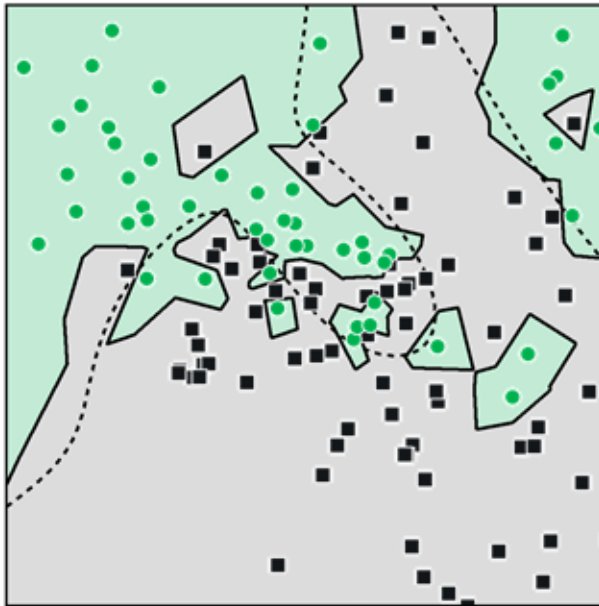
Linear Classifier



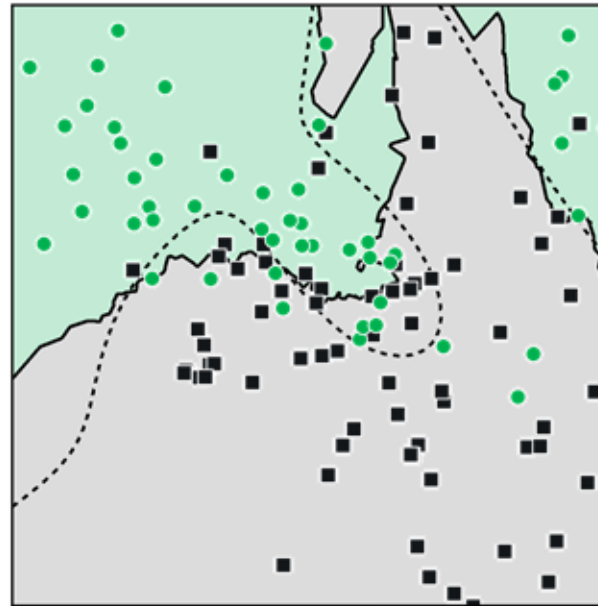
Decision Tree



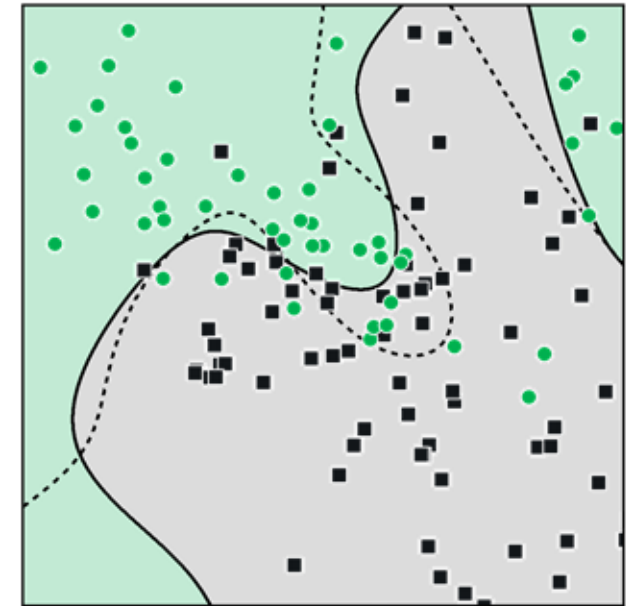
k=1 Nearest Neighbor



k=15 Nearest Neighbor



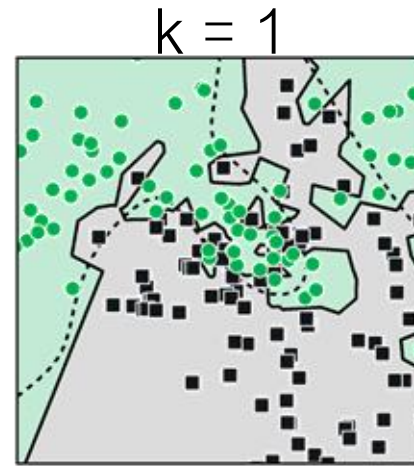
Support Vector Machine



Bias Variance Tradeoff

Data plotted are the
training data

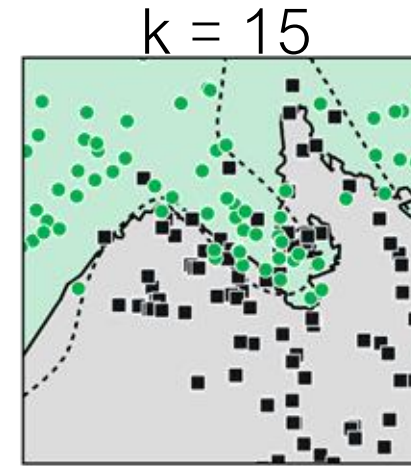
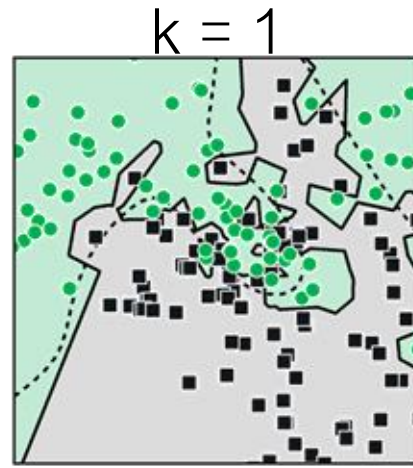
Sample 1



Bias Variance Tradeoff

Data plotted are the
training data

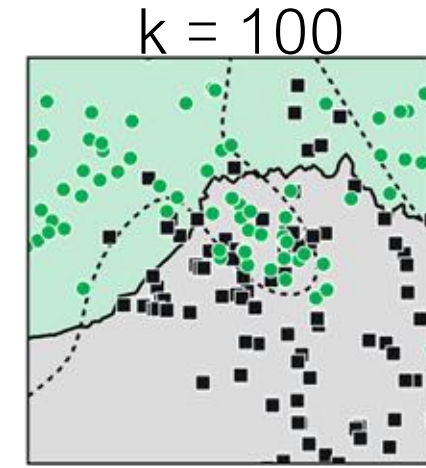
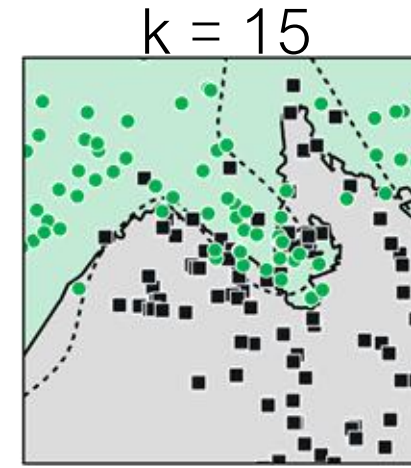
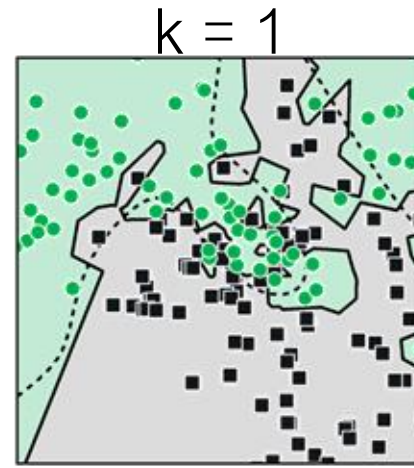
Sample 1



Bias Variance Tradeoff

Data plotted are the
training data

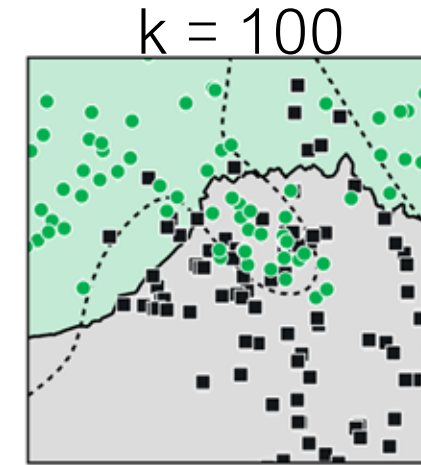
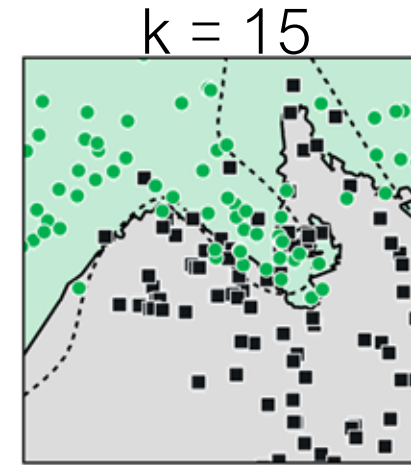
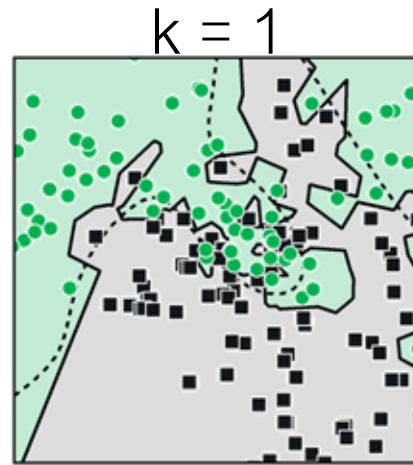
Sample 1



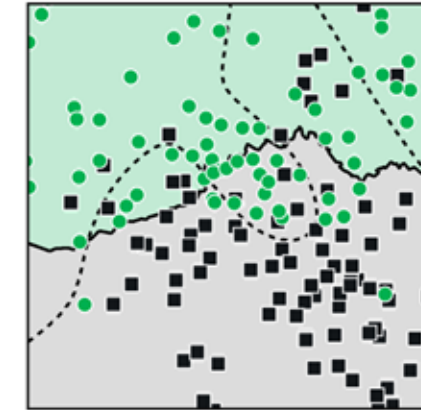
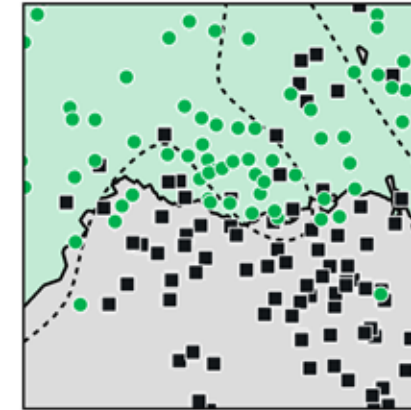
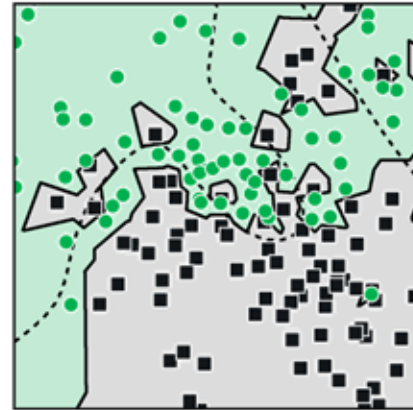
Bias Variance Tradeoff

Data plotted are the
training data

Sample 1



Sample 2



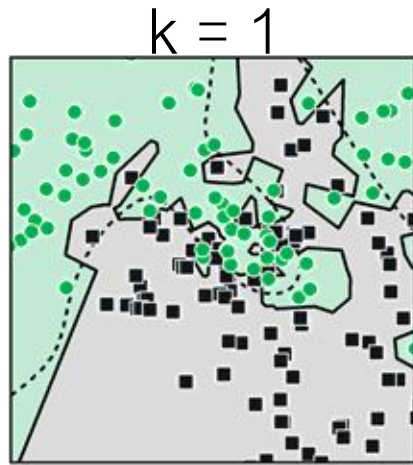
Bias Variance Tradeoff

Data plotted are the
training data

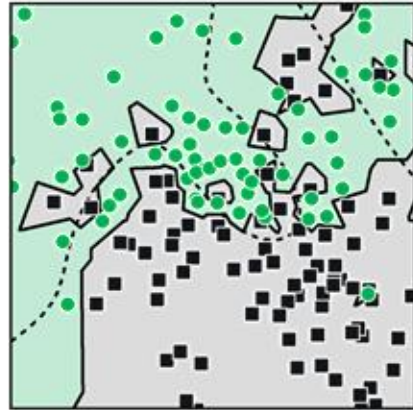
higher bias
underfit

higher variance
overfit

Sample 1



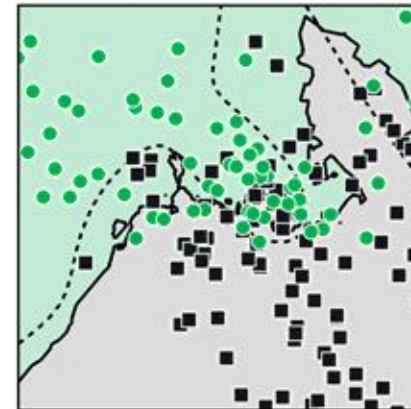
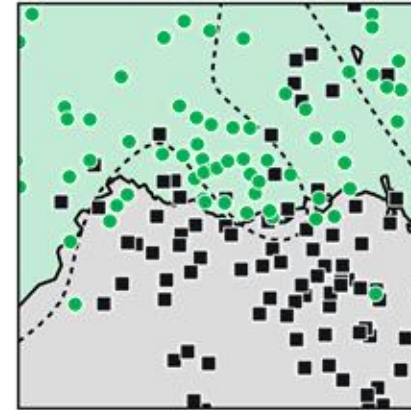
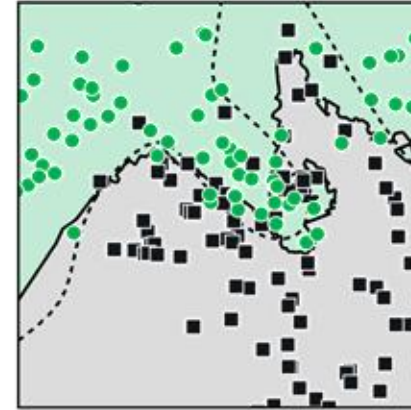
Sample 2



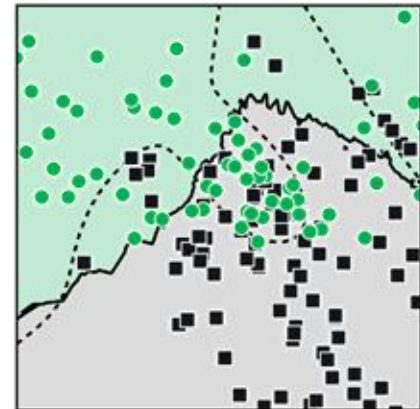
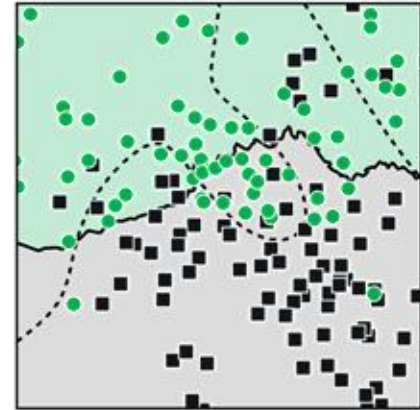
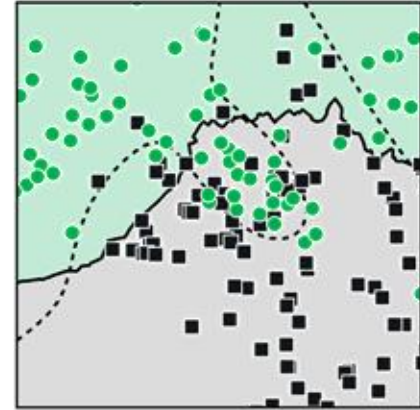
Sample 3



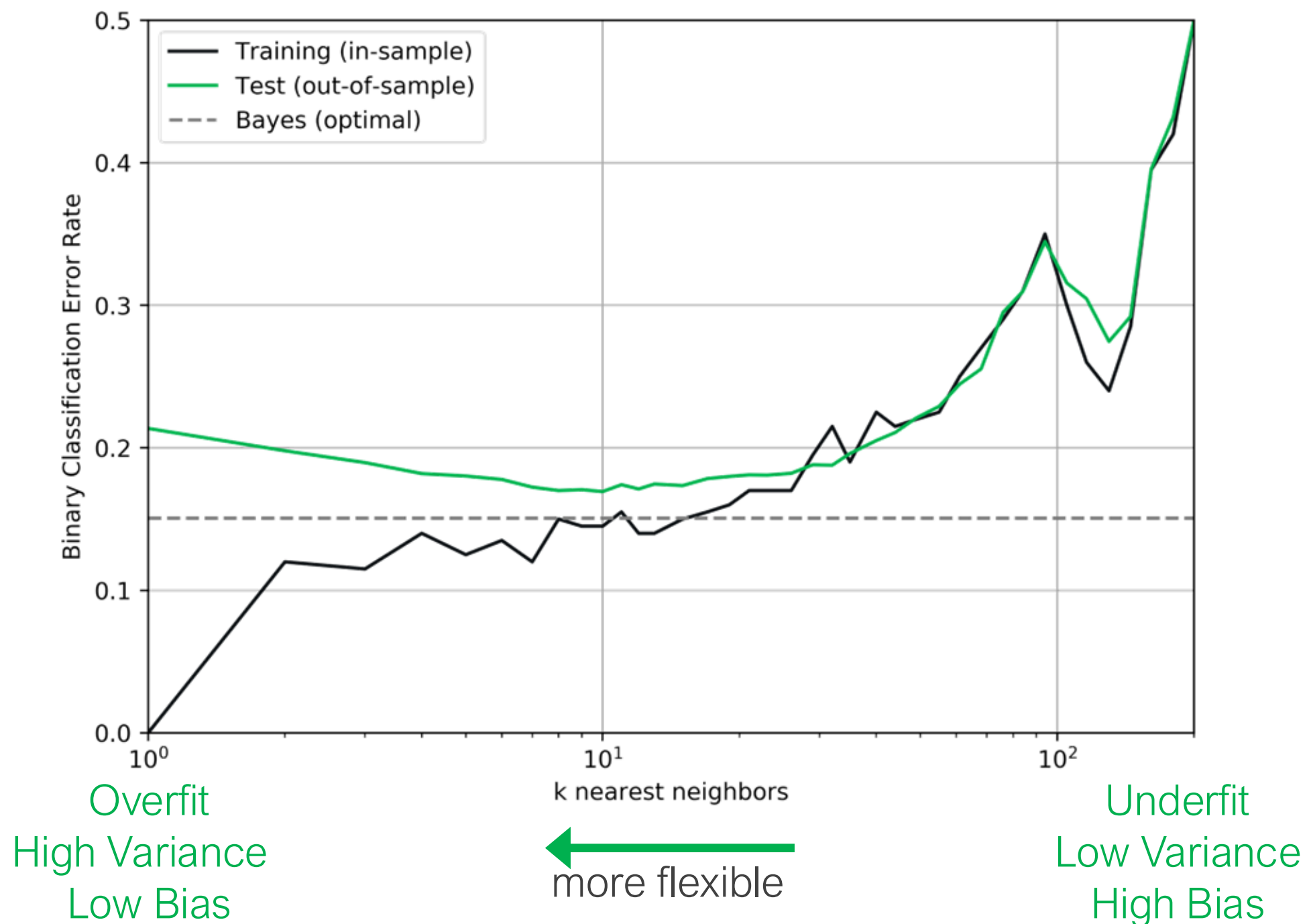
$k = 15$



$k = 100$



Bias Variance Tradeoff

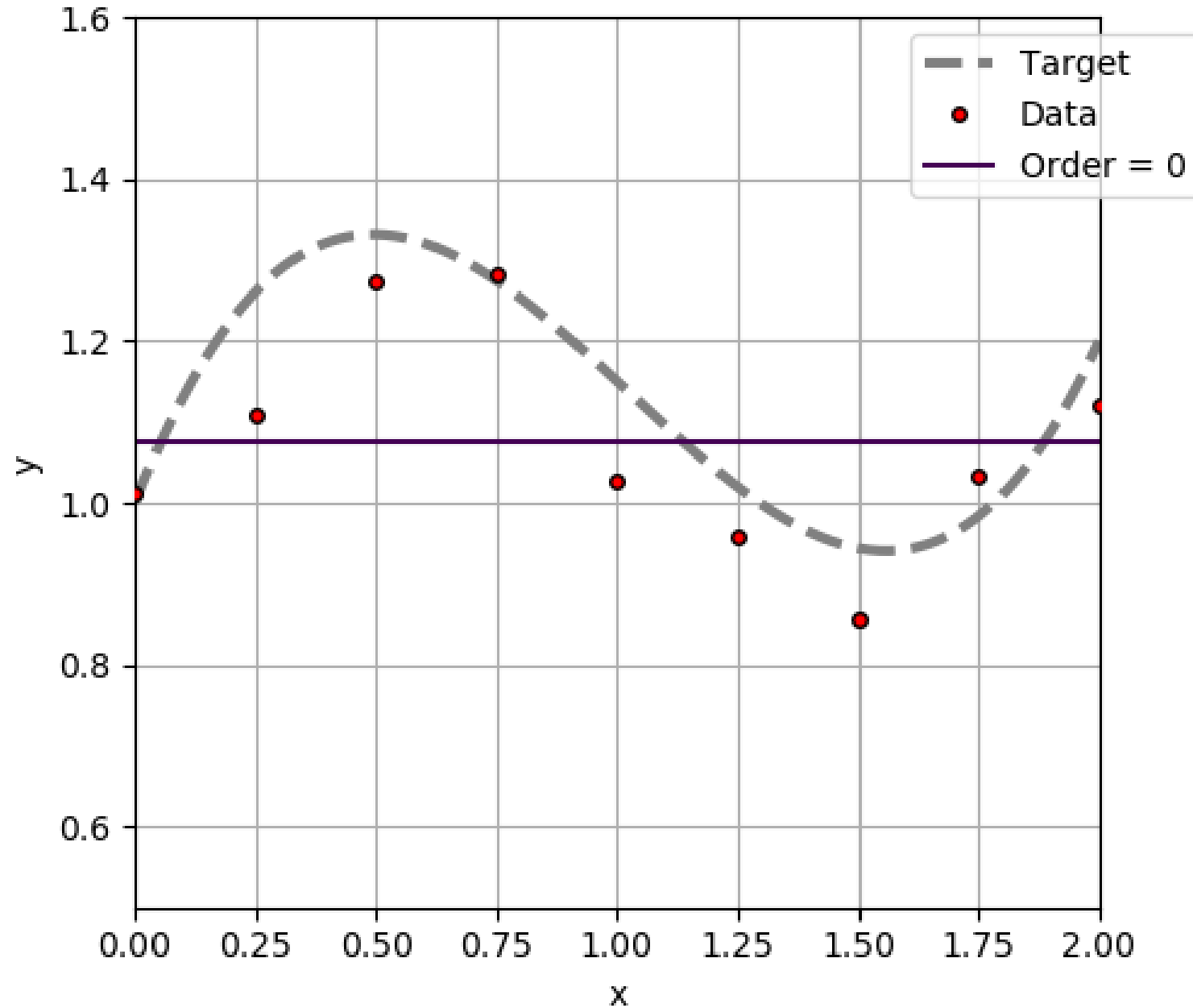


This tradeoff is equally challenging for regression

Regression

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

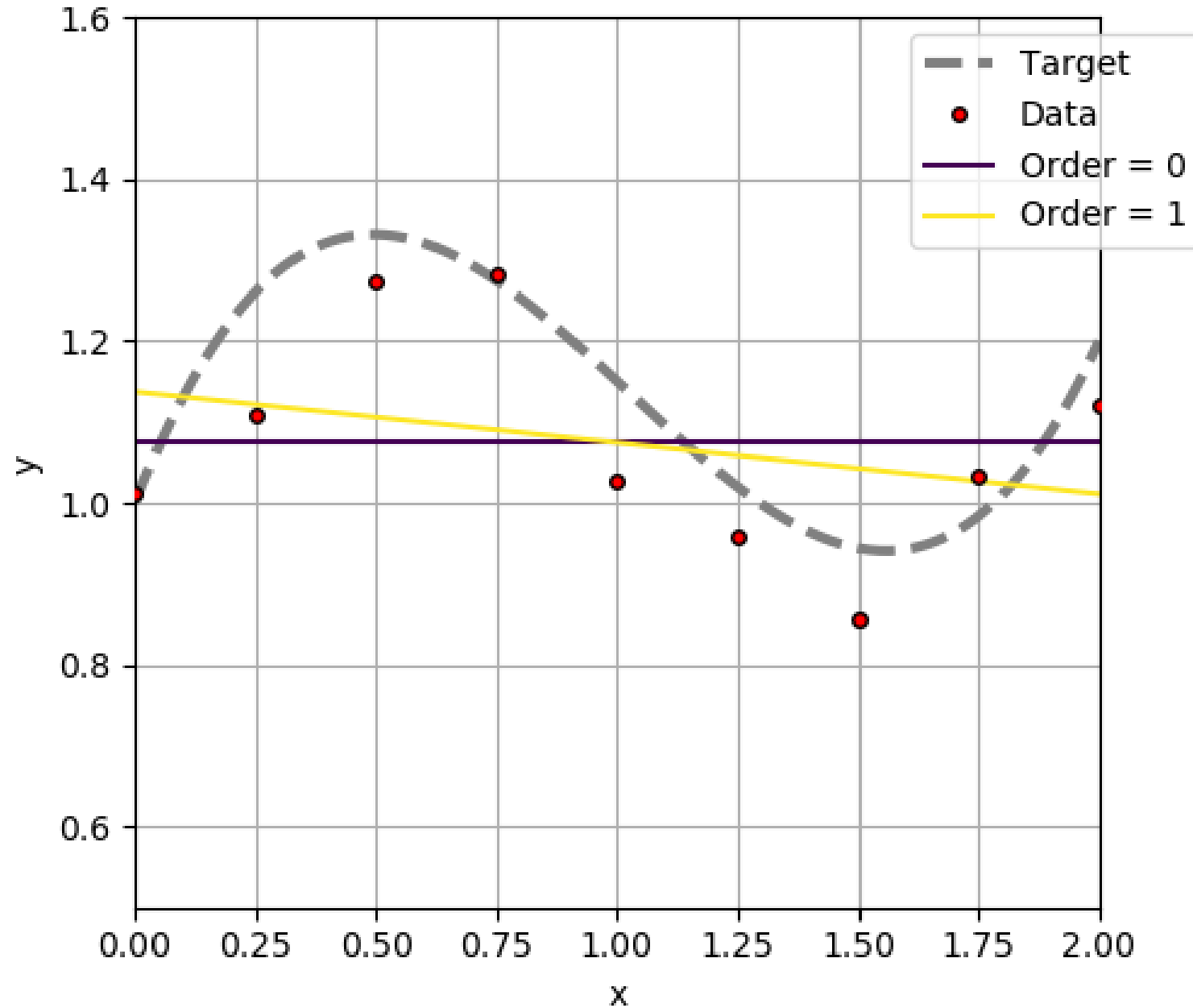
m is the model order



Regression

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

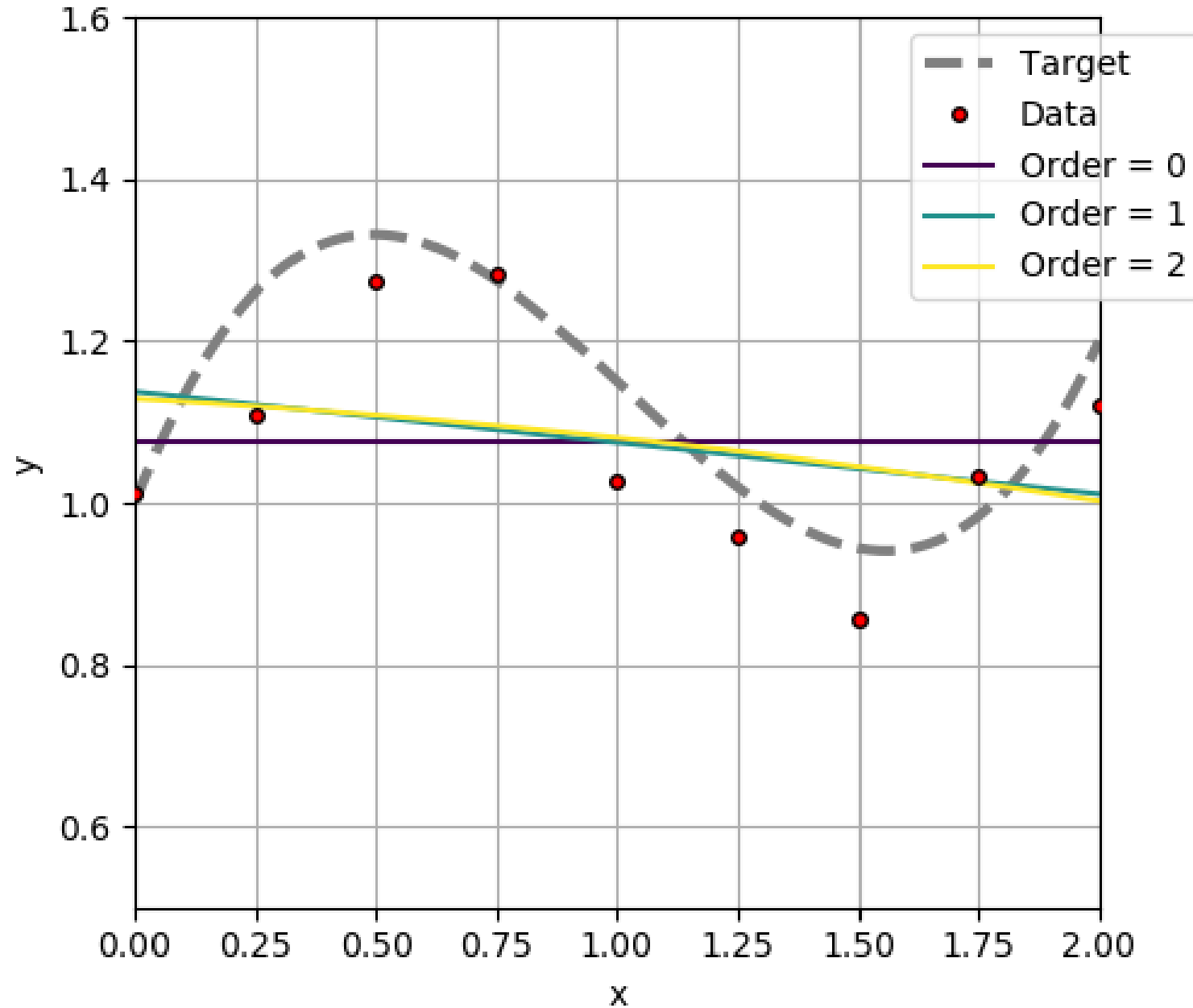
m is the model order



Regression

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

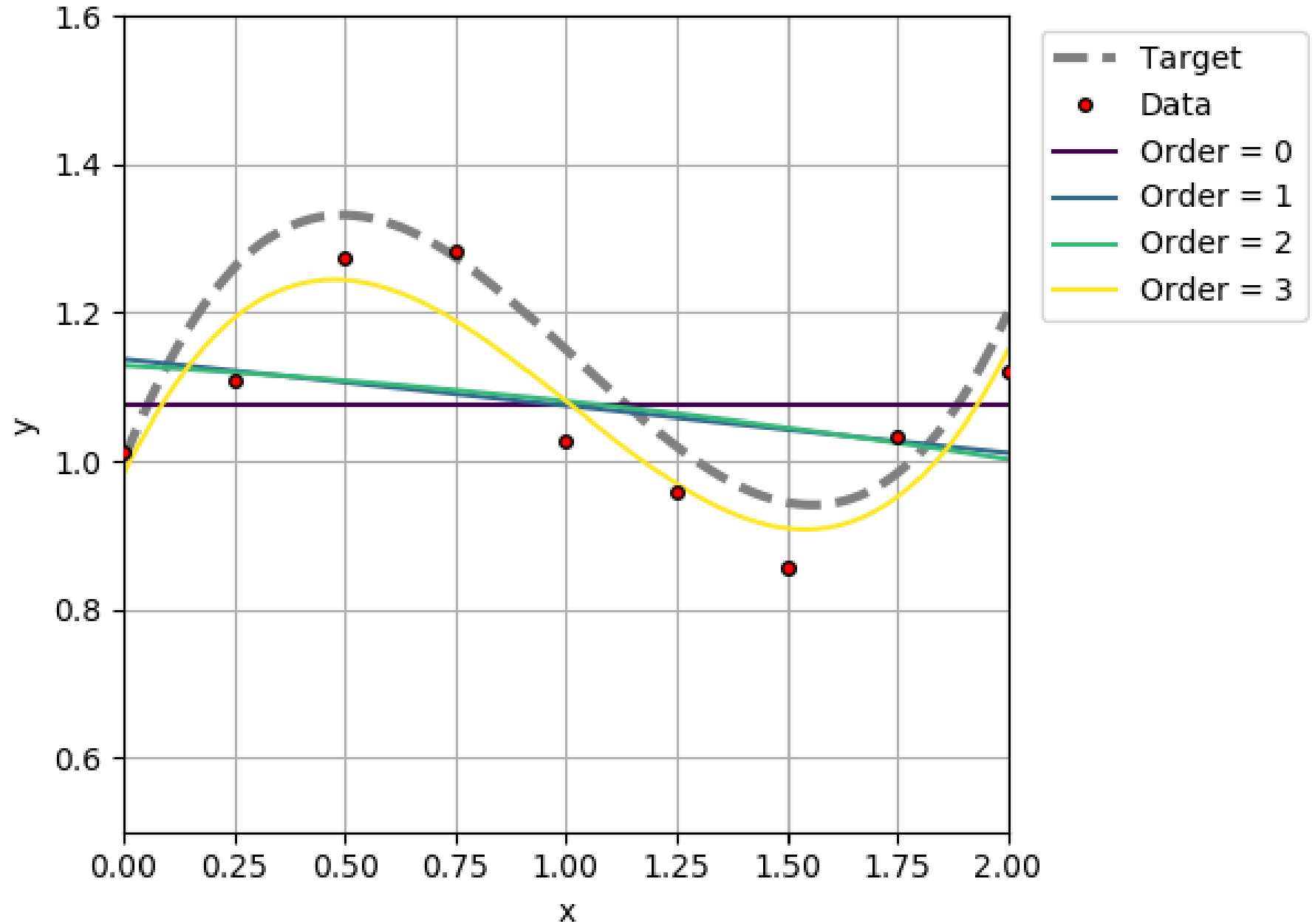
m is the model order



Regression

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

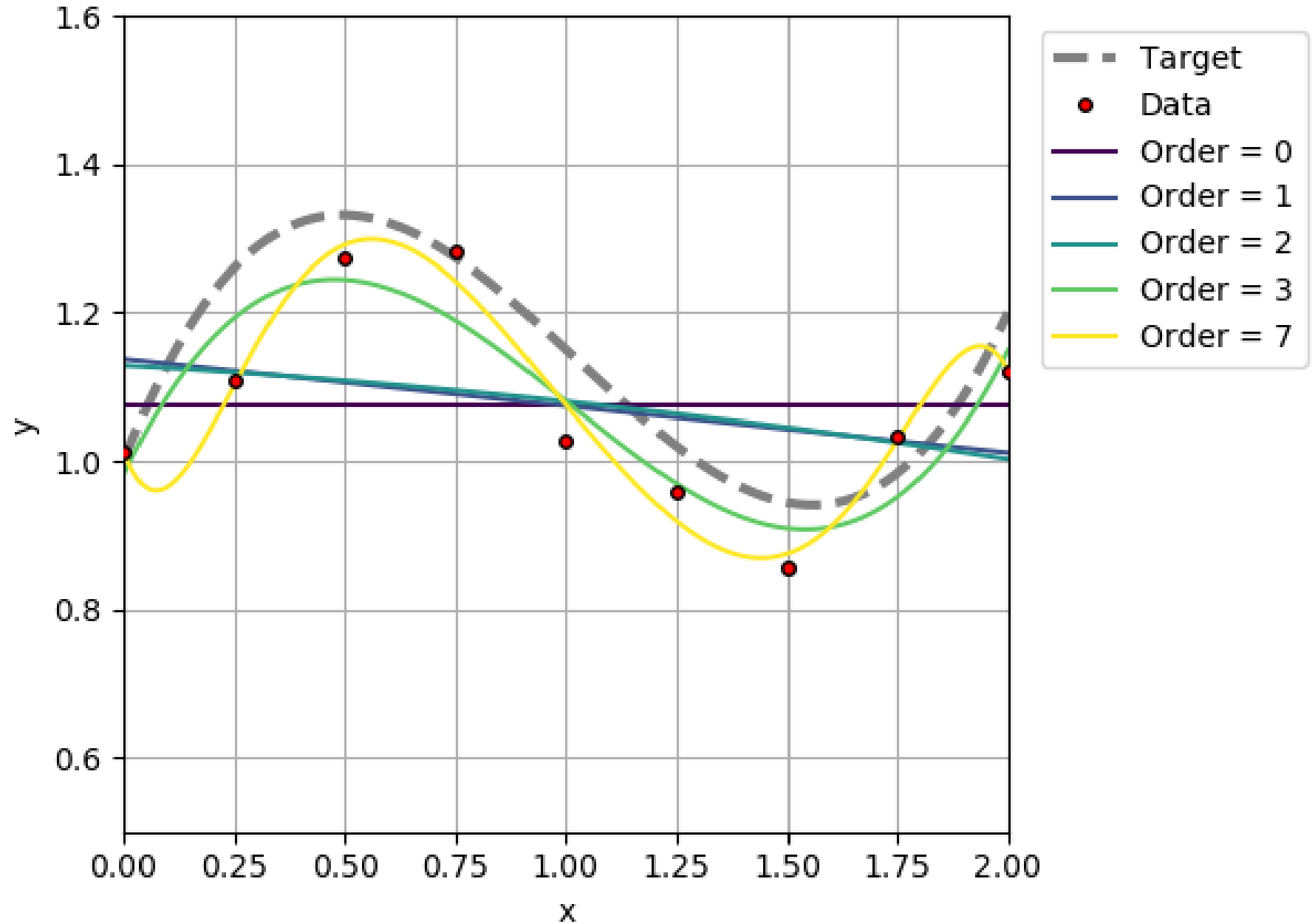
m is the model order



Regression

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

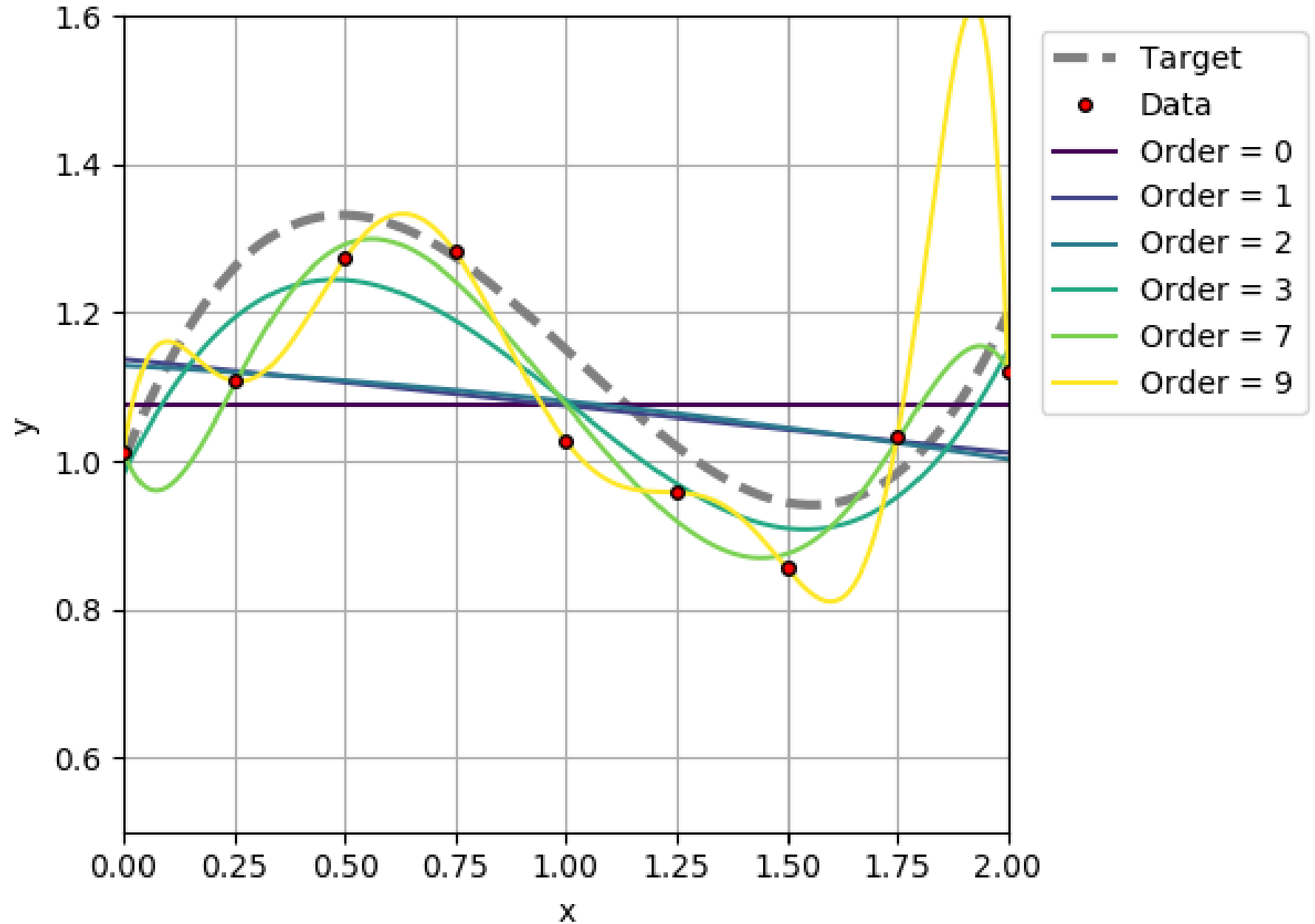
m is the model order



Regression

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

m is the model order



Problem

Too much flexibility leads to **overfit**

Too little flexibility leads to **underfit**

Over/underfit **hurts generalization** performance

Solutions for overfitting

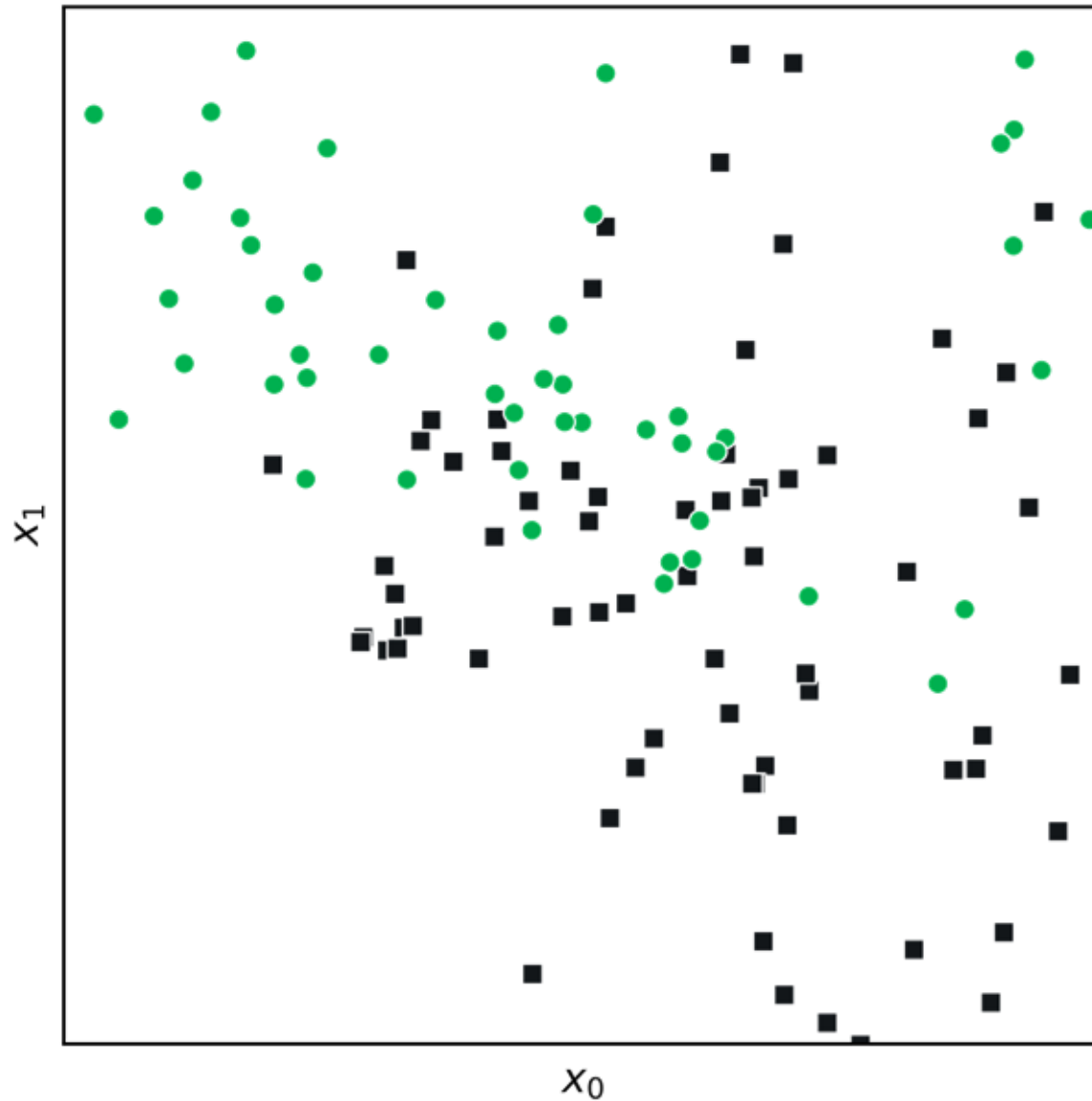
1. Add **more data** for training
2. Constrain model flexibility through **regularization**
3. Use model **ensembles**

What's the **lowest** average classification error we can achieve for binary classification?

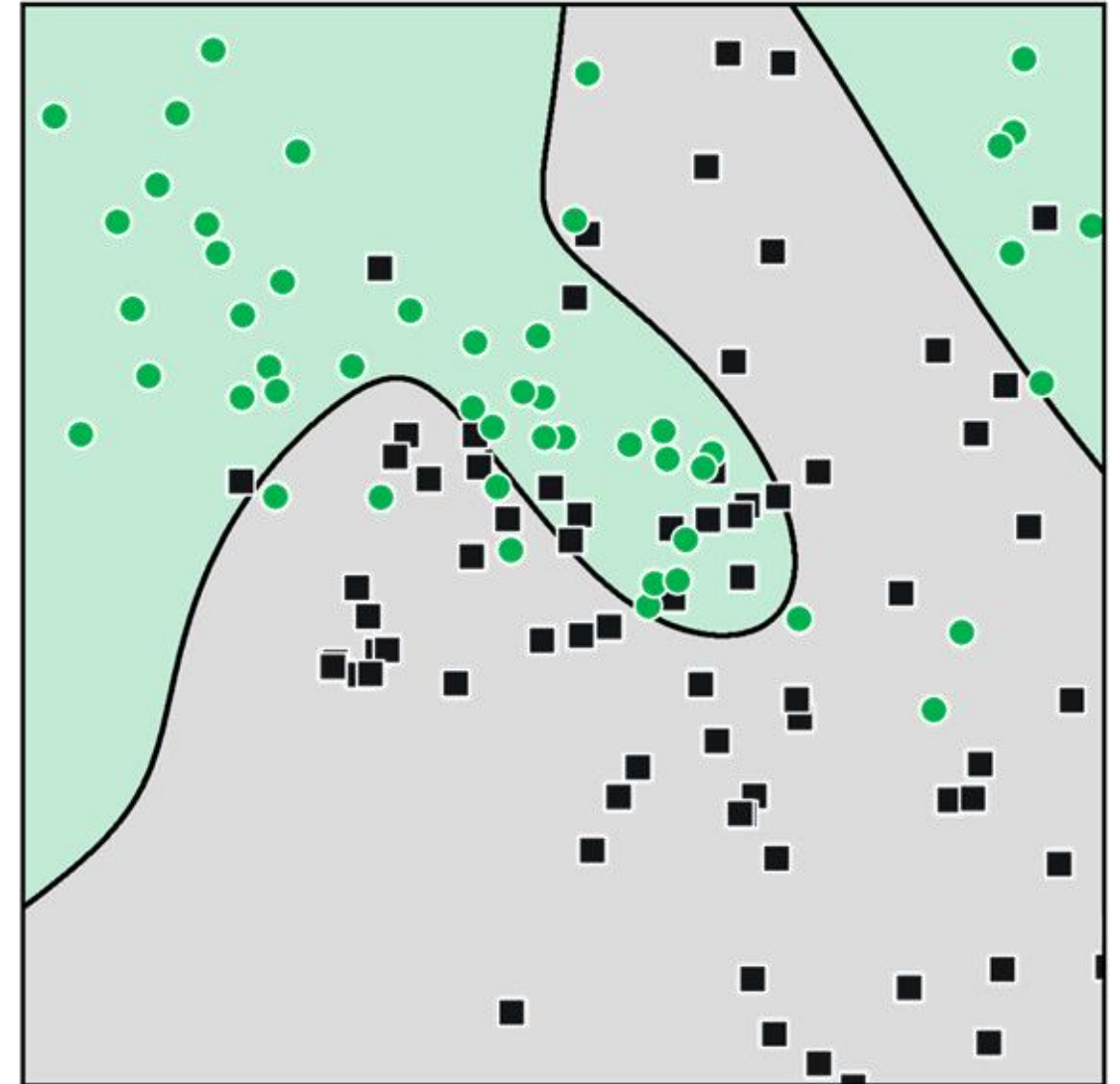
If we fully know the probability distribution of the data...

The Bayes decision rule

Classification feature space



Bayes Decision Boundary



Bayes' Rule

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)}$$

Posterior

Likelihood Prior

Evidence

X Features

C Class label

i.e. $C \in \{c_0, c_1\}$ for
the binary case

Bayes' Decision Rule:

choose the most probable class given the data

If $P(C_i = c_1 | X_i) > P(C_i = c_0 | X_i)$ then $\hat{y} = c_1$

otherwise

$\hat{y} = c_0$

- If the distributions are correct, this decision rule is **optimal**
- Rarely do we have enough information to use this in practice

Bayes' Rule Biased Coin Example

$$P(C|X) = \frac{\overset{\text{Likelihood}}{P(X|C)} \overset{\text{Prior}}{P(C)}}{\underset{\text{Evidence}}{P(X)}}$$

Two types of coins:

C_0 Coin with probability of heads: 0.5 (fair) $P(X = \{heads\}|c_0) = 0.5$

C_1 Coin with probability of heads: 0.7 (unfair) $P(X = \{heads\}|c_1) = 0.7$

You want to classify a coin as fair or unfair

You draw a coin from a bag that has 50/50 fair/unfair coins

$$P(c_0) = P(c_1) = 0.5$$

You flip the coin 5 times and it lands on heads 5 times ($\mathbf{X} = \{\text{flip 5 heads}\}$)

$$P(X|c_0) = (0.5)^5 \approx 0.03$$

$$P(X|c_1) = (0.7)^5 \approx 0.17$$

Decision rule: if $P(c_1|\mathbf{X}) > P(c_0|\mathbf{X})$, then the coin is unfair, otherwise the coin is fair

$$\frac{P(X|c_1)P(c_1)}{P(X)} > \frac{P(X|c_0)P(c_0)}{P(X)} \rightarrow \frac{(0.17)(0.5)}{P(X)} > \frac{(0.03)(0.5)}{P(X)} \rightarrow \text{We predict the coin is } \mathbf{unfair}$$