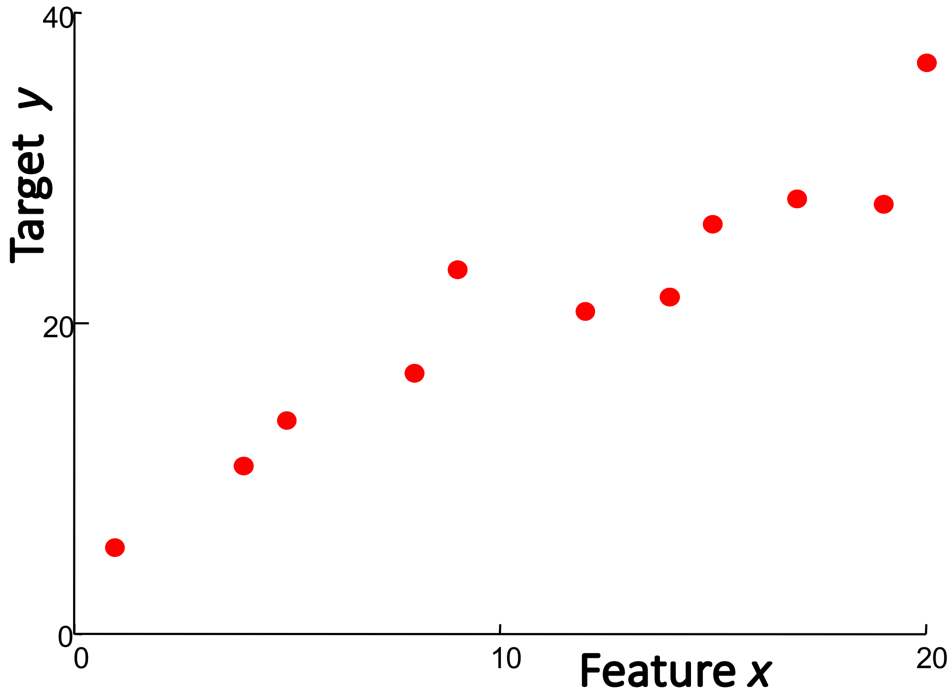
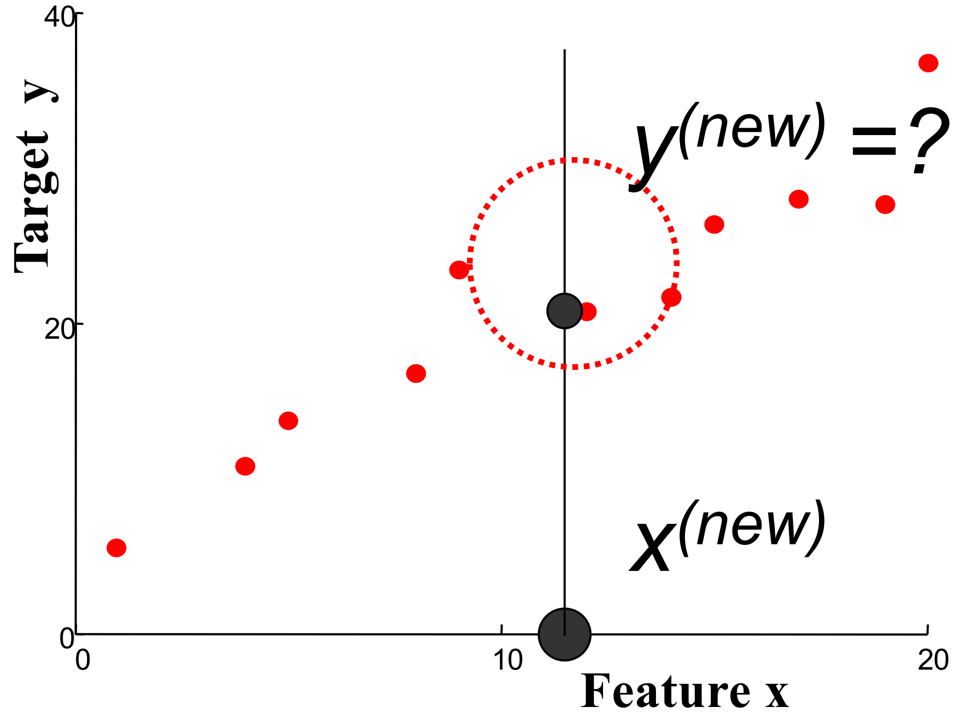
Nearest Neighbor Regression



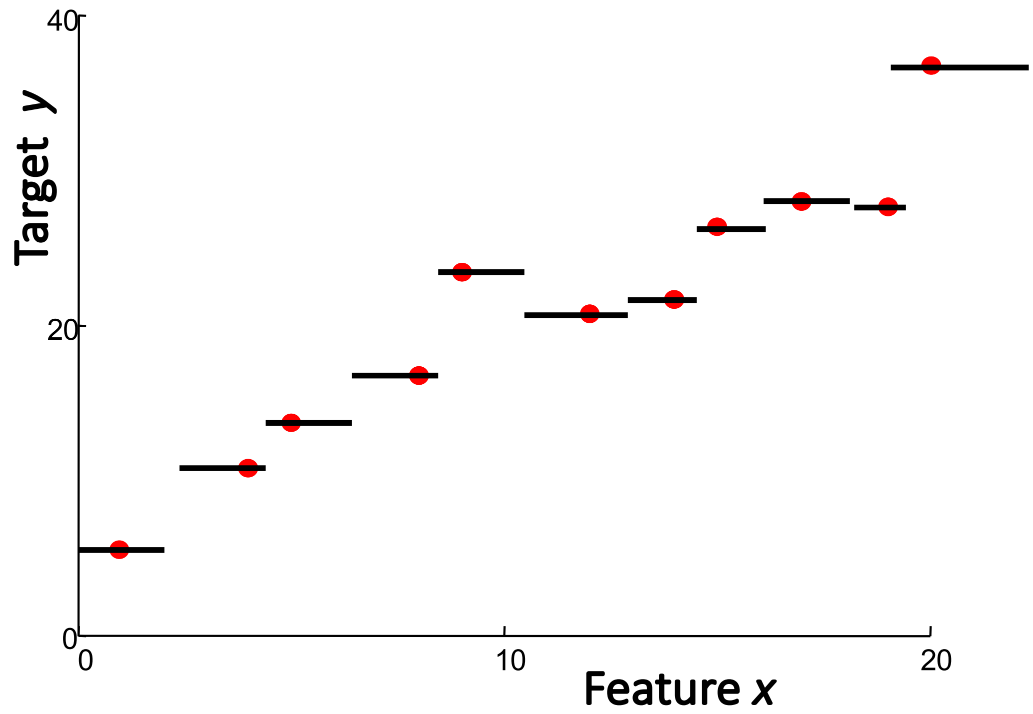
Find training datum *x(i)* closest to *x(new)* Predict *y(i)*

Nearest neighbor regression



Find training datum x(i) closest to x(new) Predict y(i)

Nearest Neighbor Regression

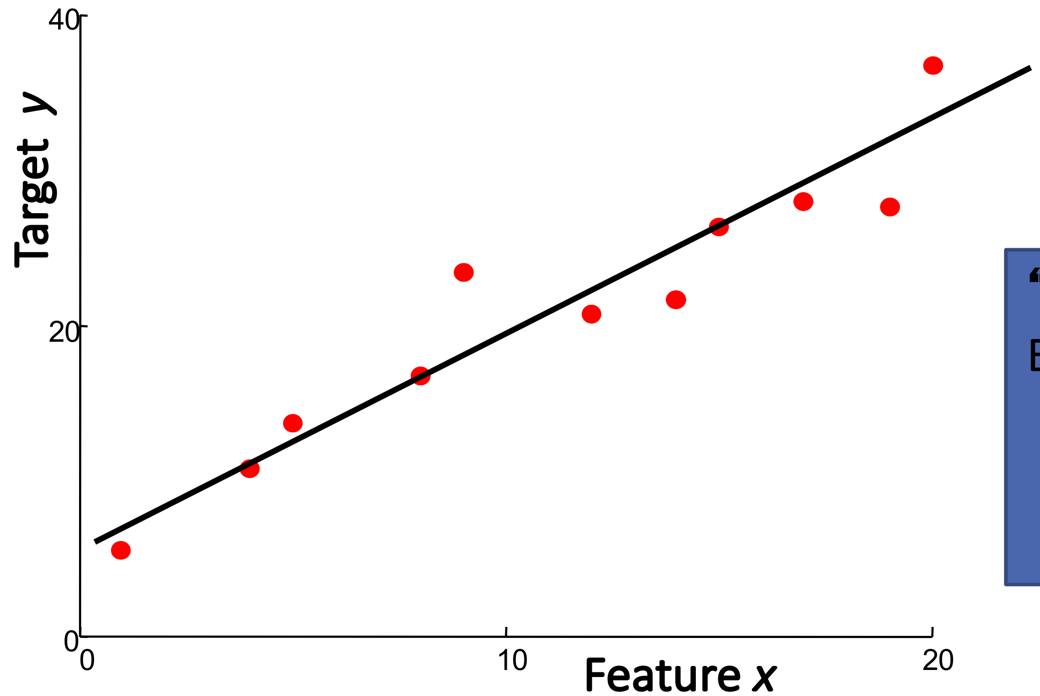


"Predictor":
Given new features:
Find nearest example
Return its value

Defines a function *f(x)* implicitly

"Form" is piecewise constant

Alternative: linear regression

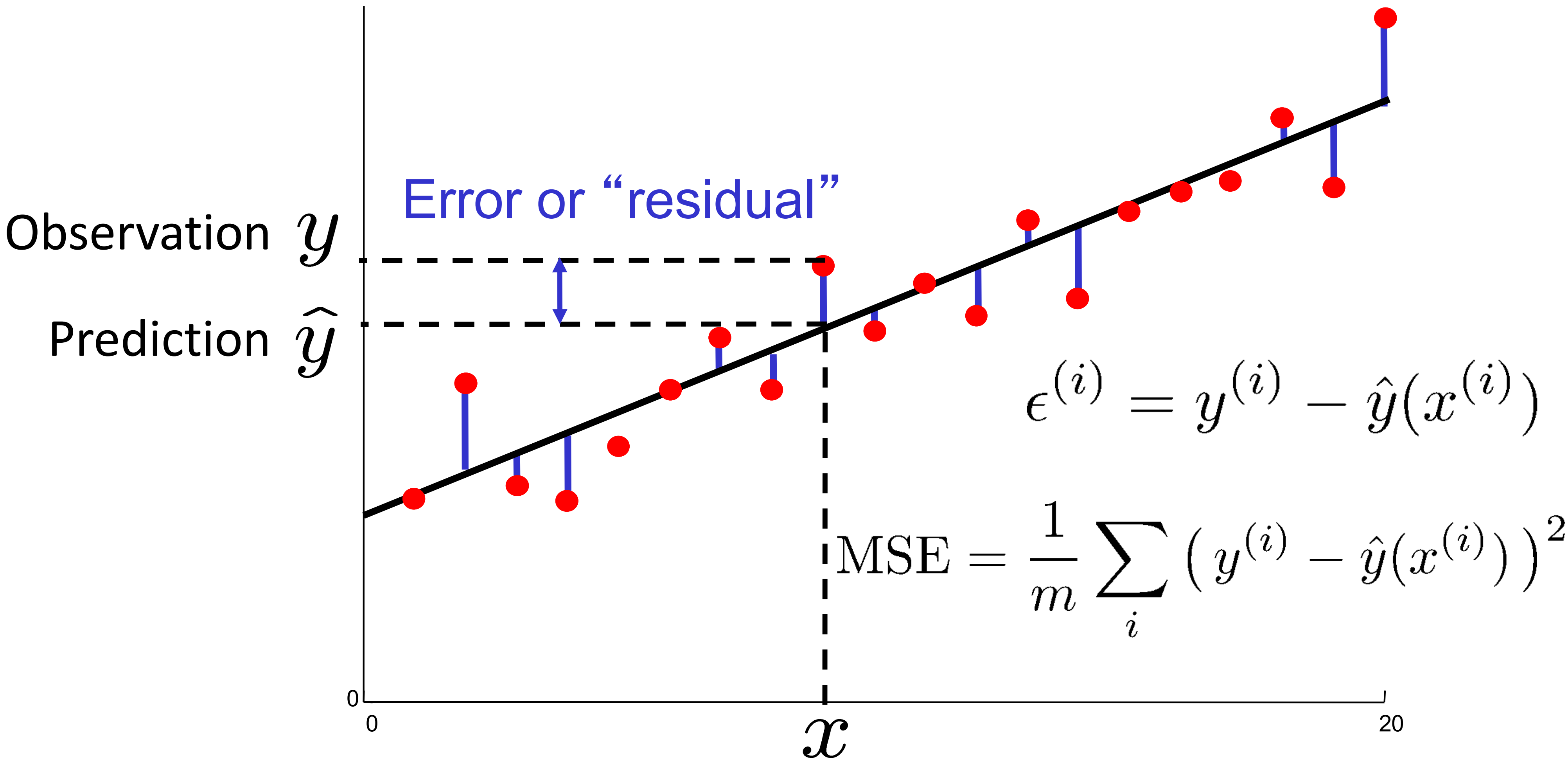


"Predictor":
Evaluate line:
$$r = \theta_0 + \theta_1 x_1$$
return r

Define form of function *f(x)* explicitly

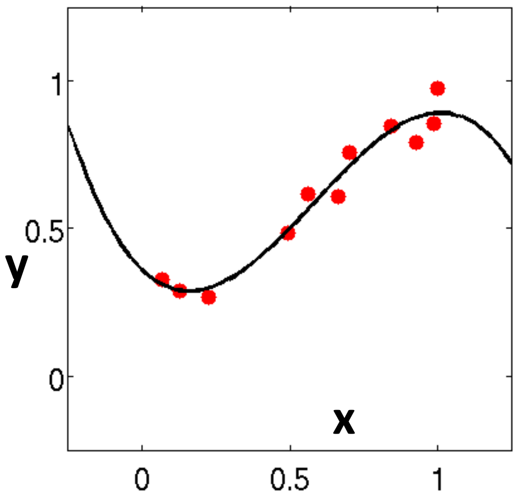
Find a good *f(x)* within that family

Measuring error



Regression vs. Classification

Regression

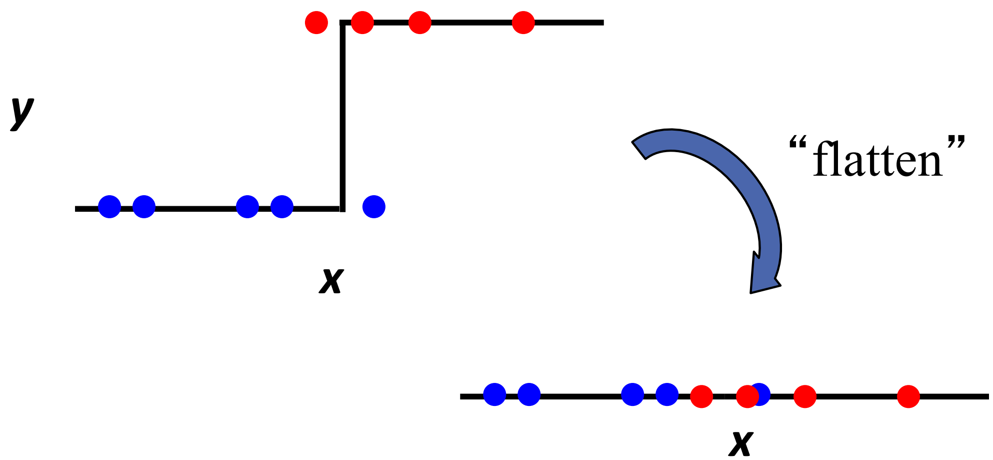


Features *x*

Real-valued target *y*

Predict continuous function *ŷ(x)*

Classification



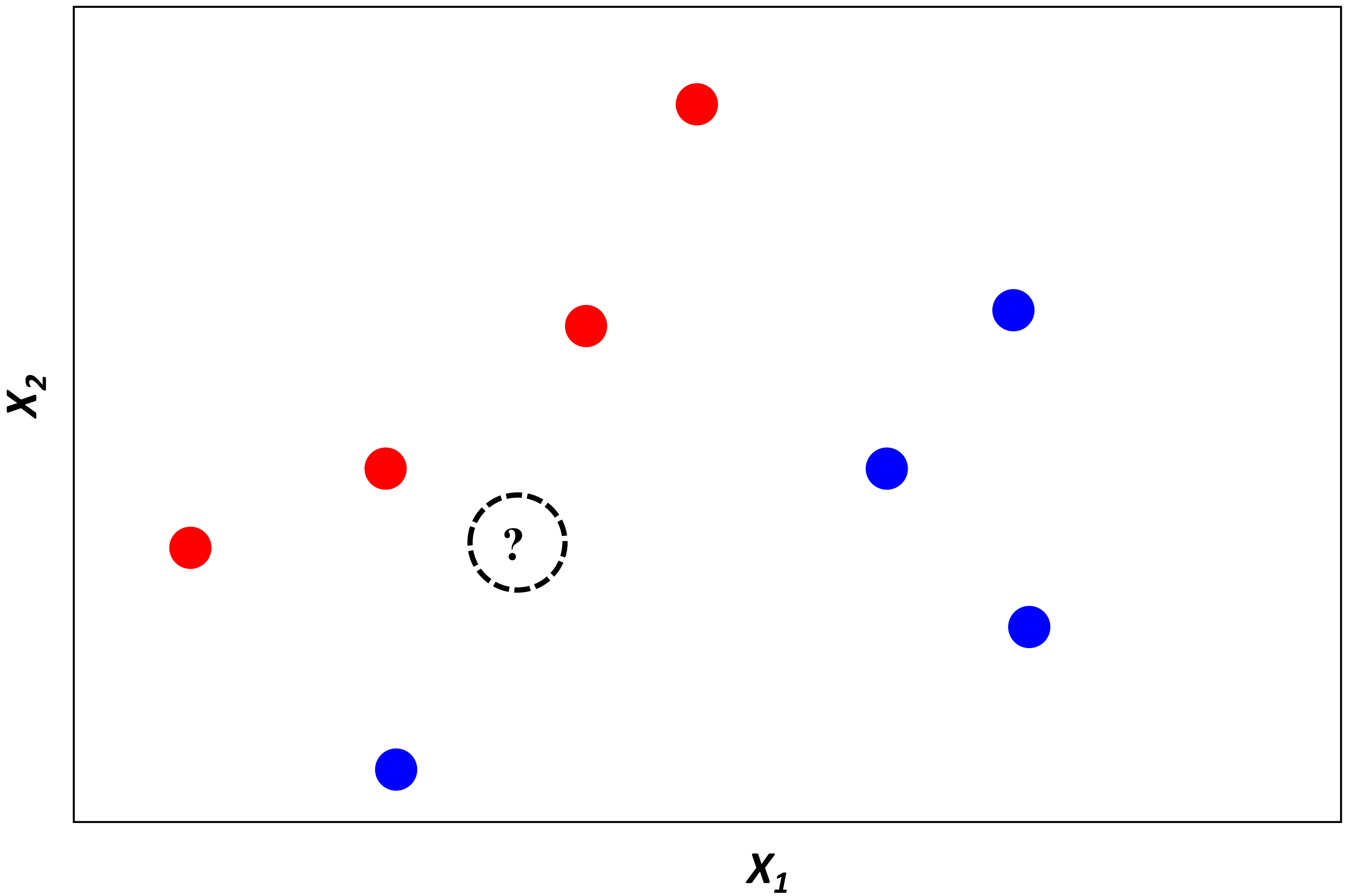
Features *x*

Discrete class *c*

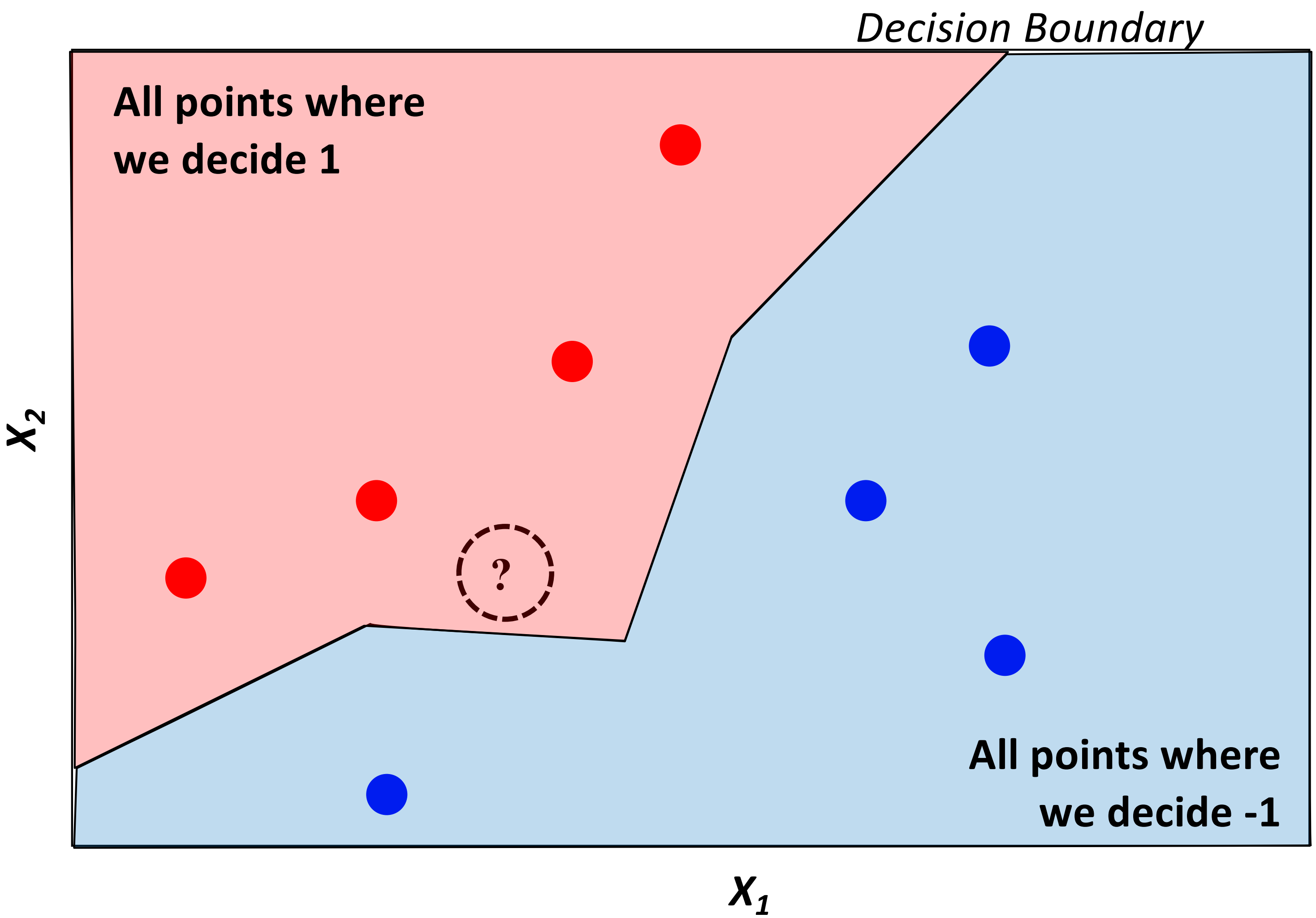
(usually 0/1 or +1/-1)

Predict discrete function *ŷ(x)*

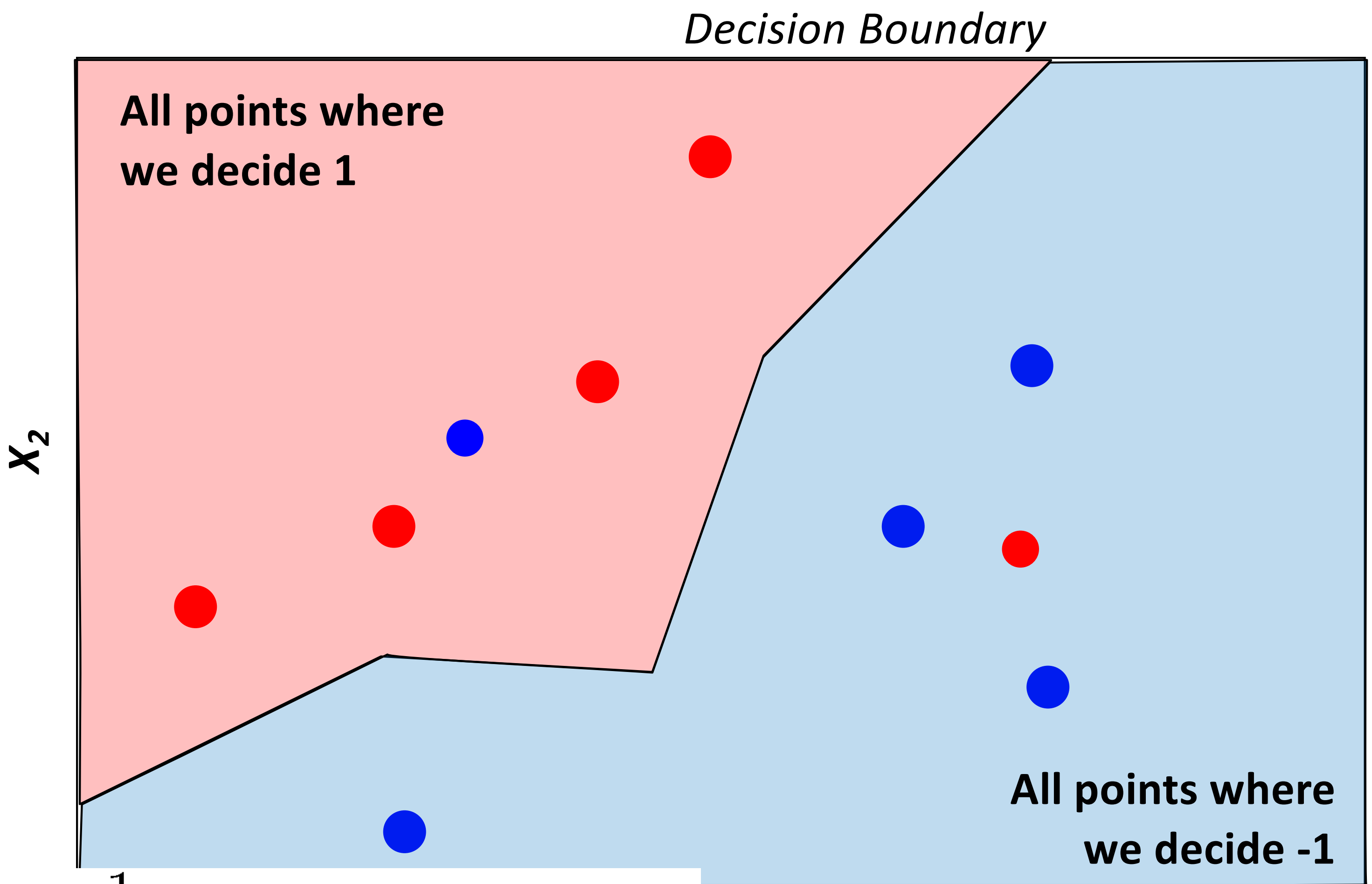
Classification



Classification



Measuring Error



$$\mathrm{ERR} = \frac{1}{m} \sum_i \left[\, y^{(i)} \neq \hat{y}(x^{(i)}) \,\right]$$

Summary

Supervised learning

- Training data: features x, targets y

Regression

- *(x,y)* scatterplots; predictor outputs *f(x)*
- Mean squared error

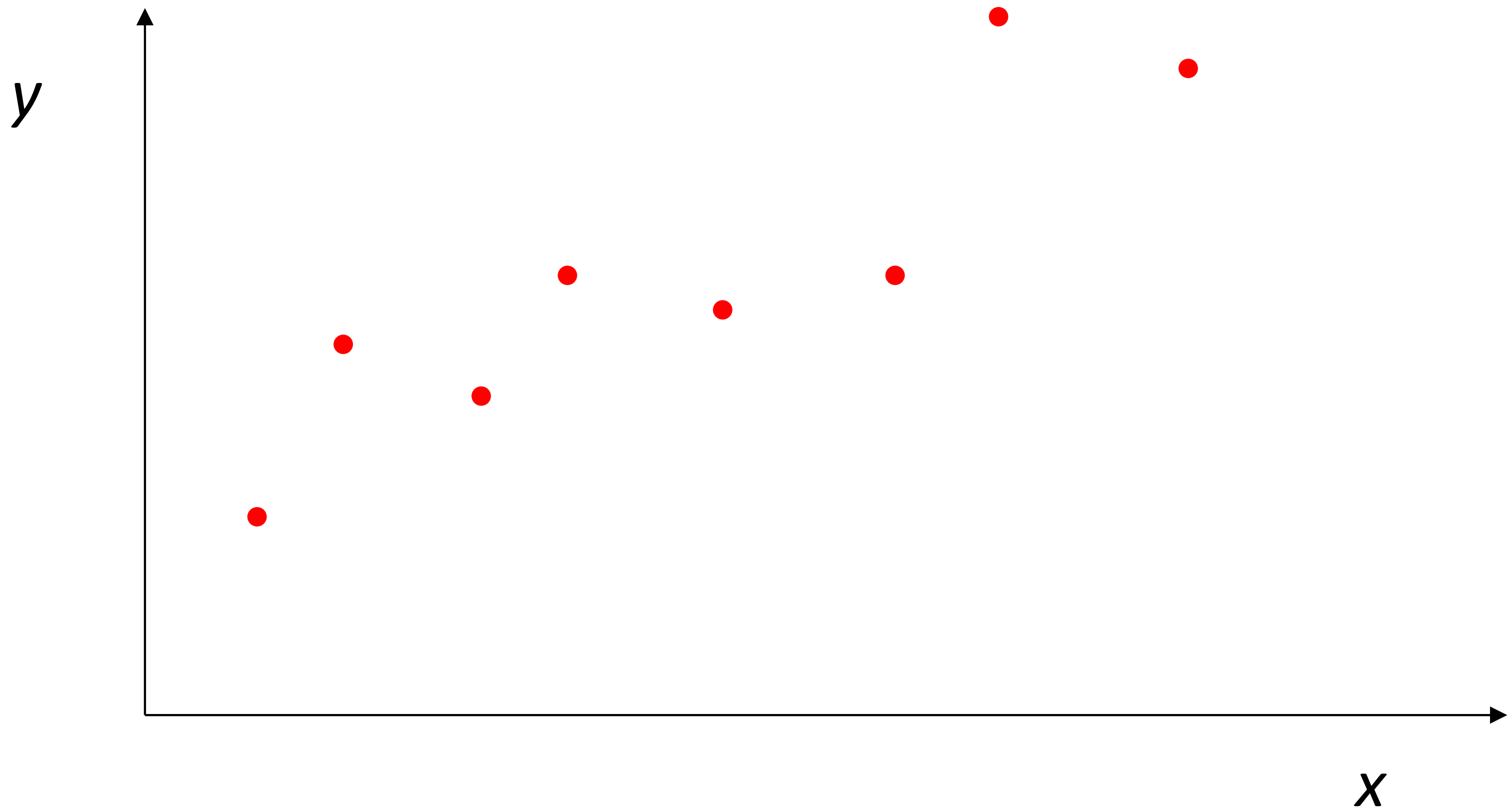
Classification

- *(x,x)* scatterplots
- Decision boundaries, colors & symbols
- Empirical error rate

Machine Learning

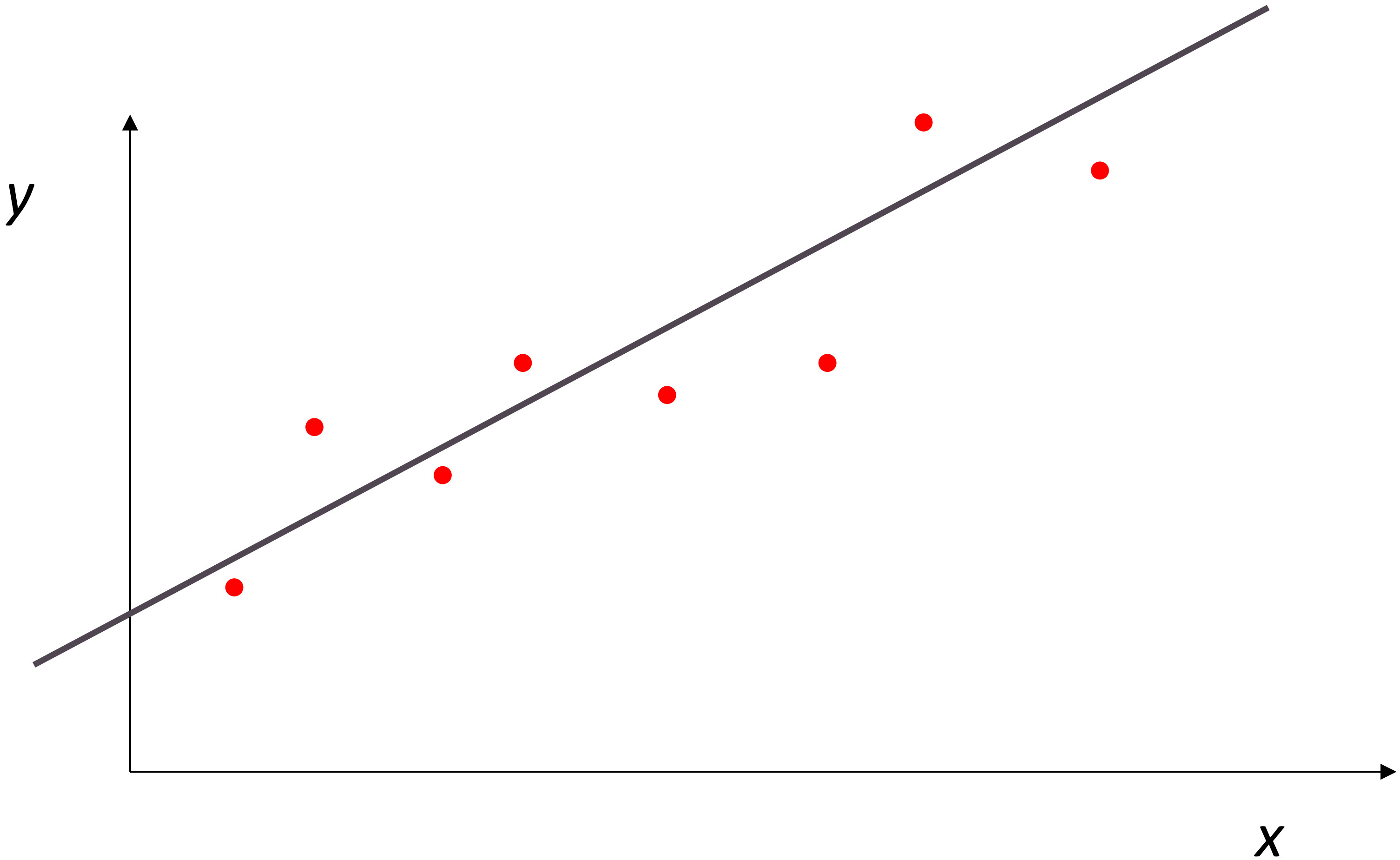
- Complexity and Overfitting
- Nearest Neighbors
- K-Nearest Neighbors
- Bayes Classifiers

Overfitting and complexity

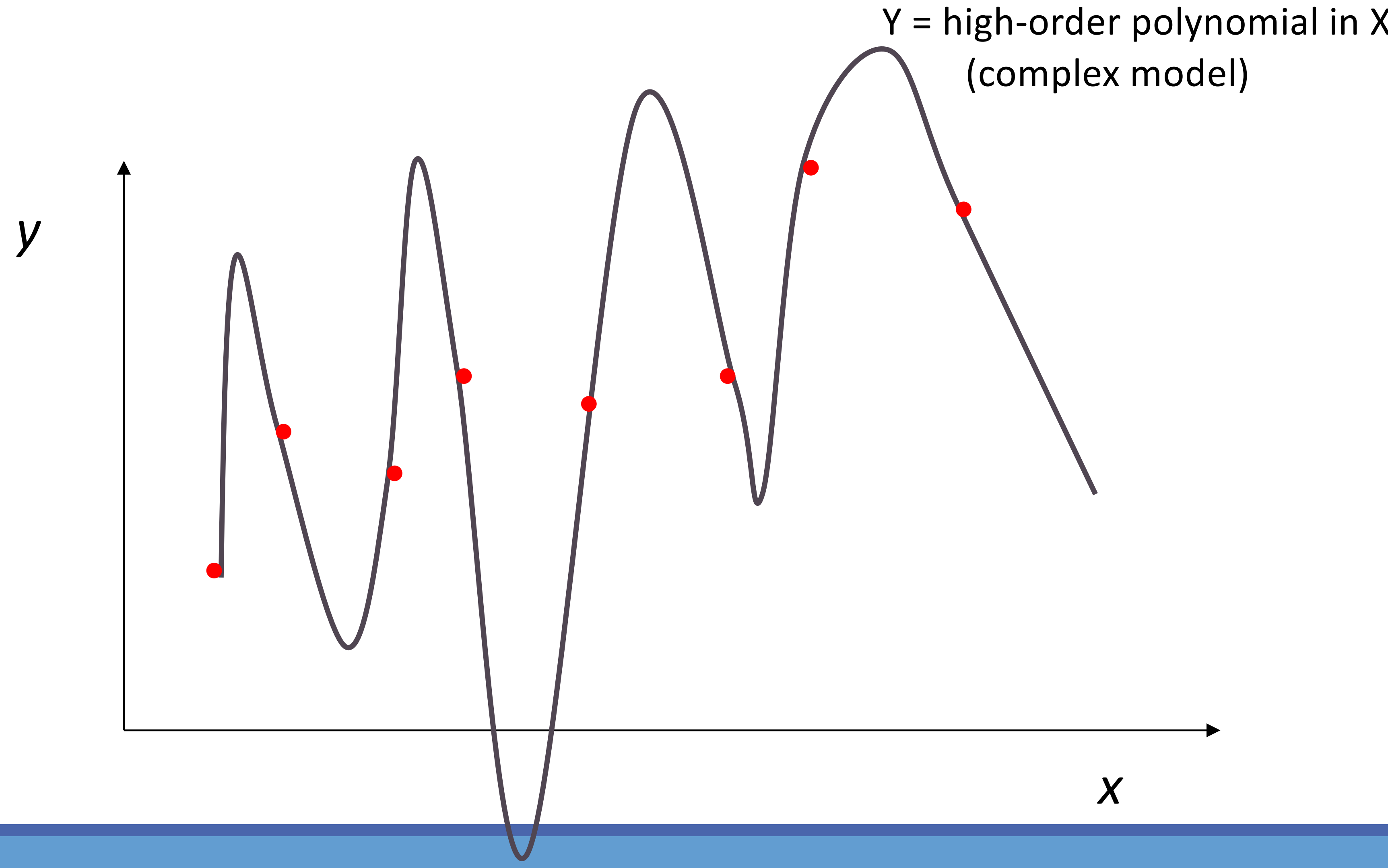


Overfitting and complexity

Simple model: $Y = aX + b + e$

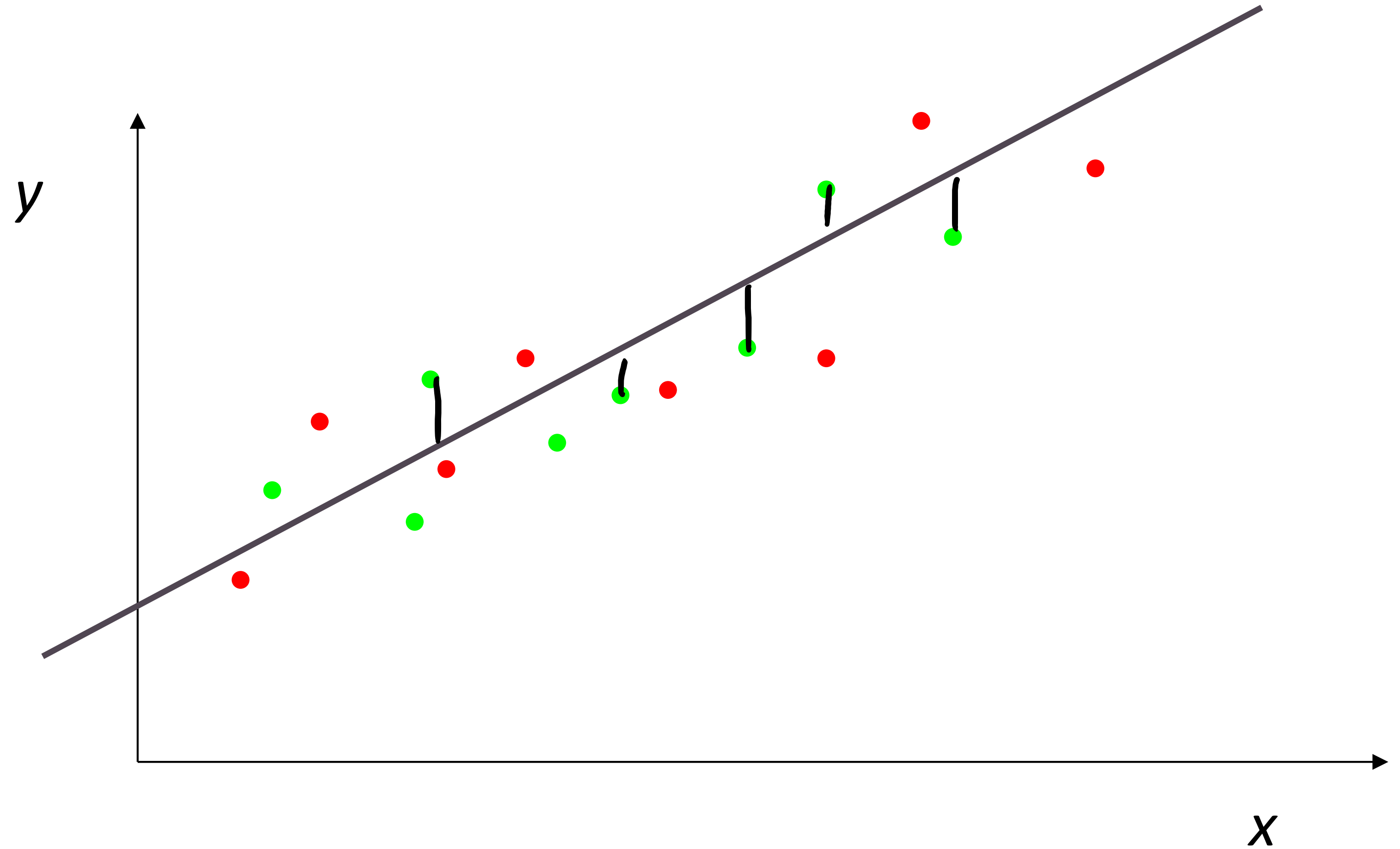


Overfitting and complexity

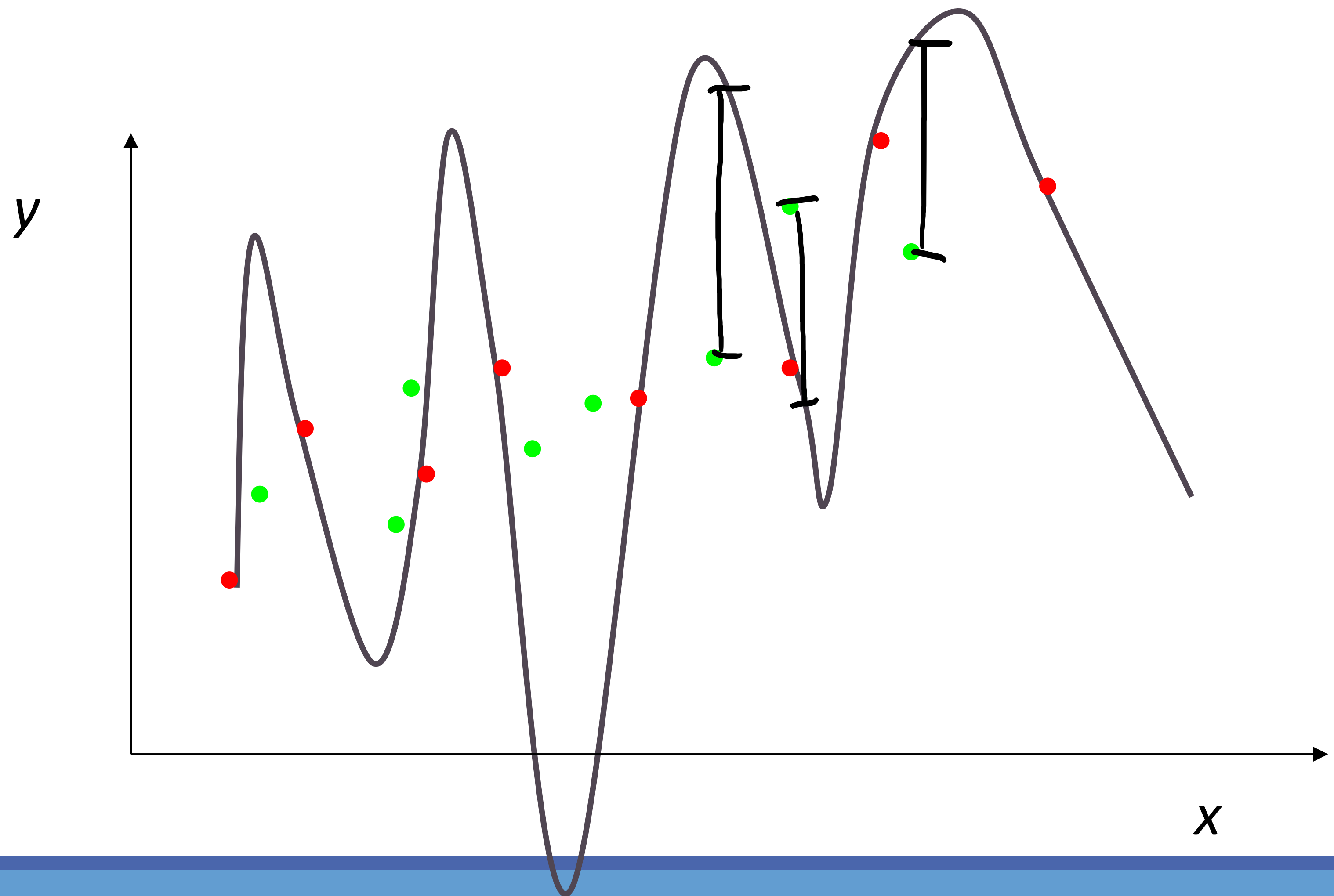


Overfitting and complexity

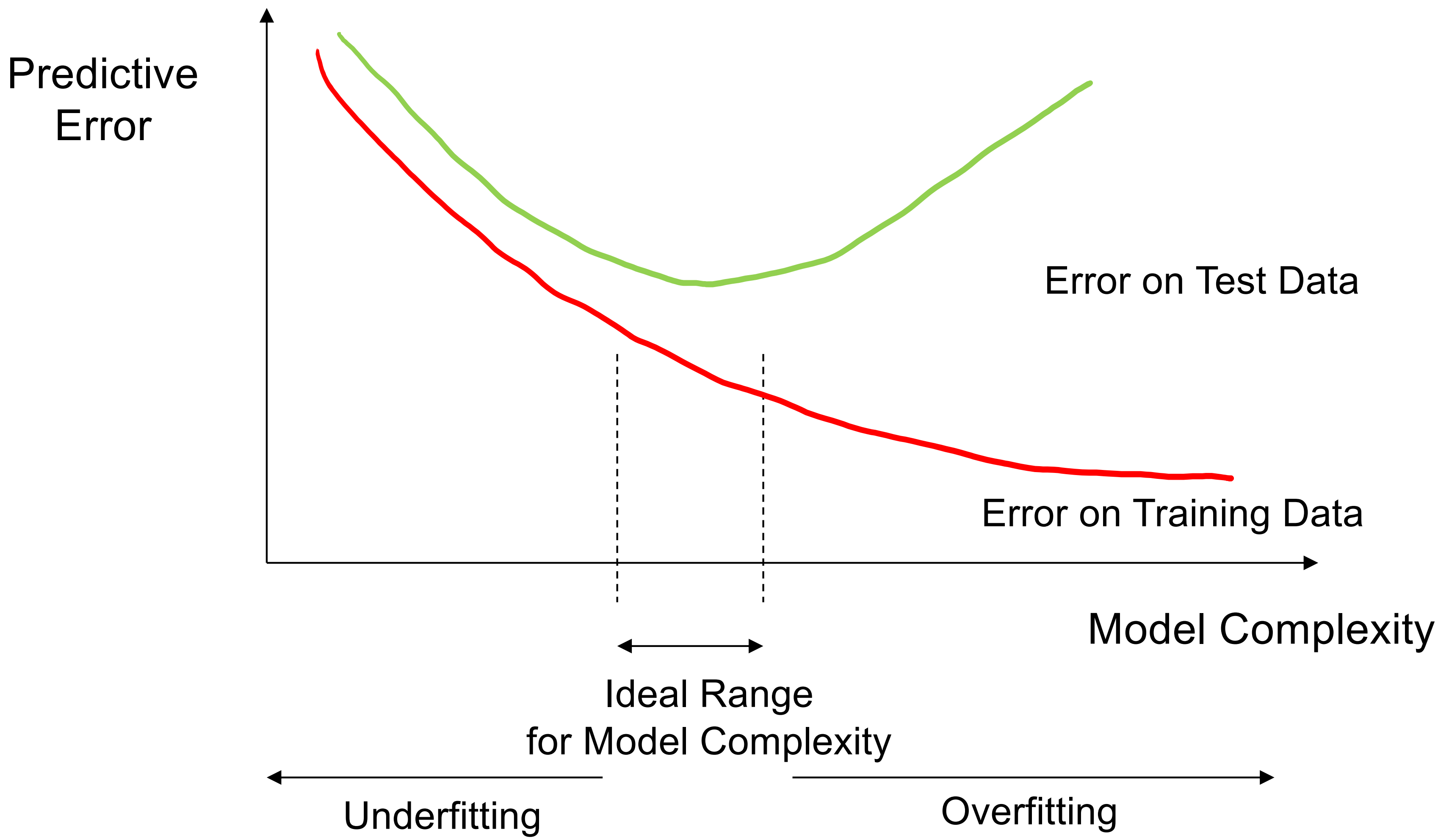
Simple model: $Y = aX + b + e$



Overfitting and complexity



How Overfitting affects Prediction



Competitions

Training data

- Used to build your model(s)

Validation data

- Used to assess, select among, or combine models
- Personal validation; leaderboard; …

Test data

- Used to estimate "real world" performance

#	Δ1w	Team Name * in the money	Score	Entries	Last Submission UT
1	-	BrickMover *	1.21251	40	Sat, 31 Aug 2013 23:
2	new	vsu *	1.21552	13	Sat, 31 Aug 2013 20:
3	↑2	Merlion	1.22724	29	Sat, 31 Aug 2013 23:
4	↓2	Sergey	1.22856	15	Sat, 31 Aug 2013 23:
5	new	liuyongqi	1.22980	13	Sat, 31 Aug 2013 13:

Summary

Complexity

- Training versus Test errors
- Under- and Over-fitting

Machine Learning

- Complexity and Overfitting
- Nearest Neighbors
- K-Nearest Neighbors
- Bayes Classifiers

Supervised learning

Notation
- Features x
- Targets y
- Predictions $\hat{y}$
- Parameters θ

Training data (examples)

Features

Feedback / Target values

Program ("Learner")

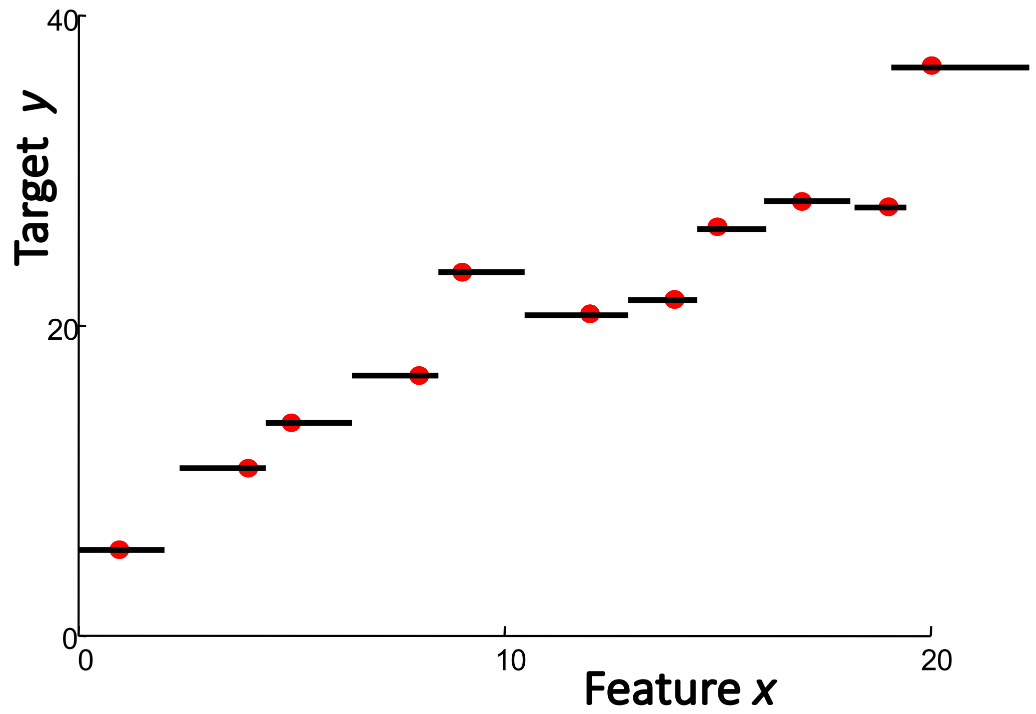
Characterized by some "parameters" θ
Procedure (using θ) that outputs a prediction

Score performance ("cost function")

Learning algorithm

Change θ
Improve performance

Nearest Neighbor Regression

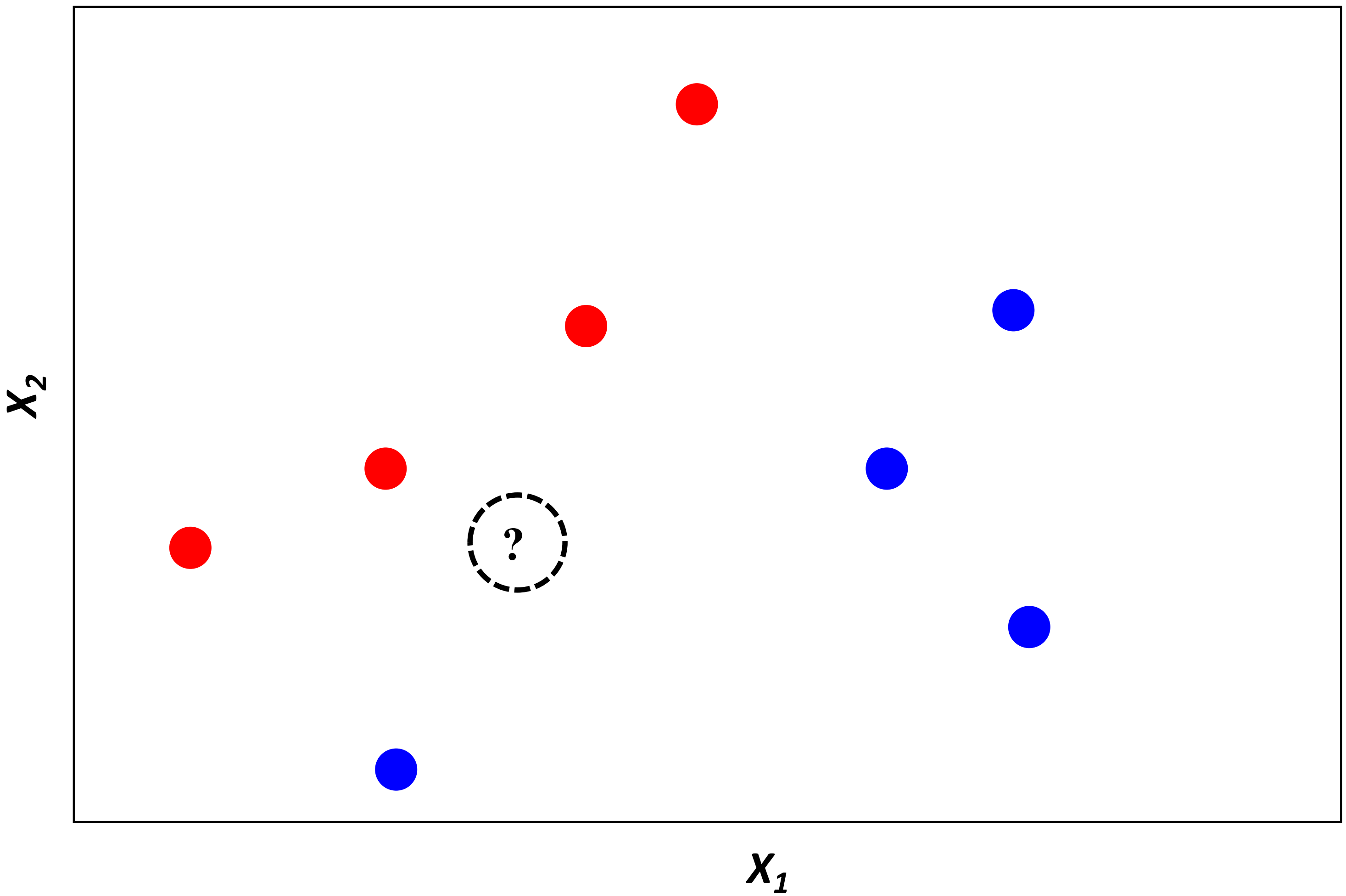


"Predictor":
Given new features:
Find nearest example
Return its value

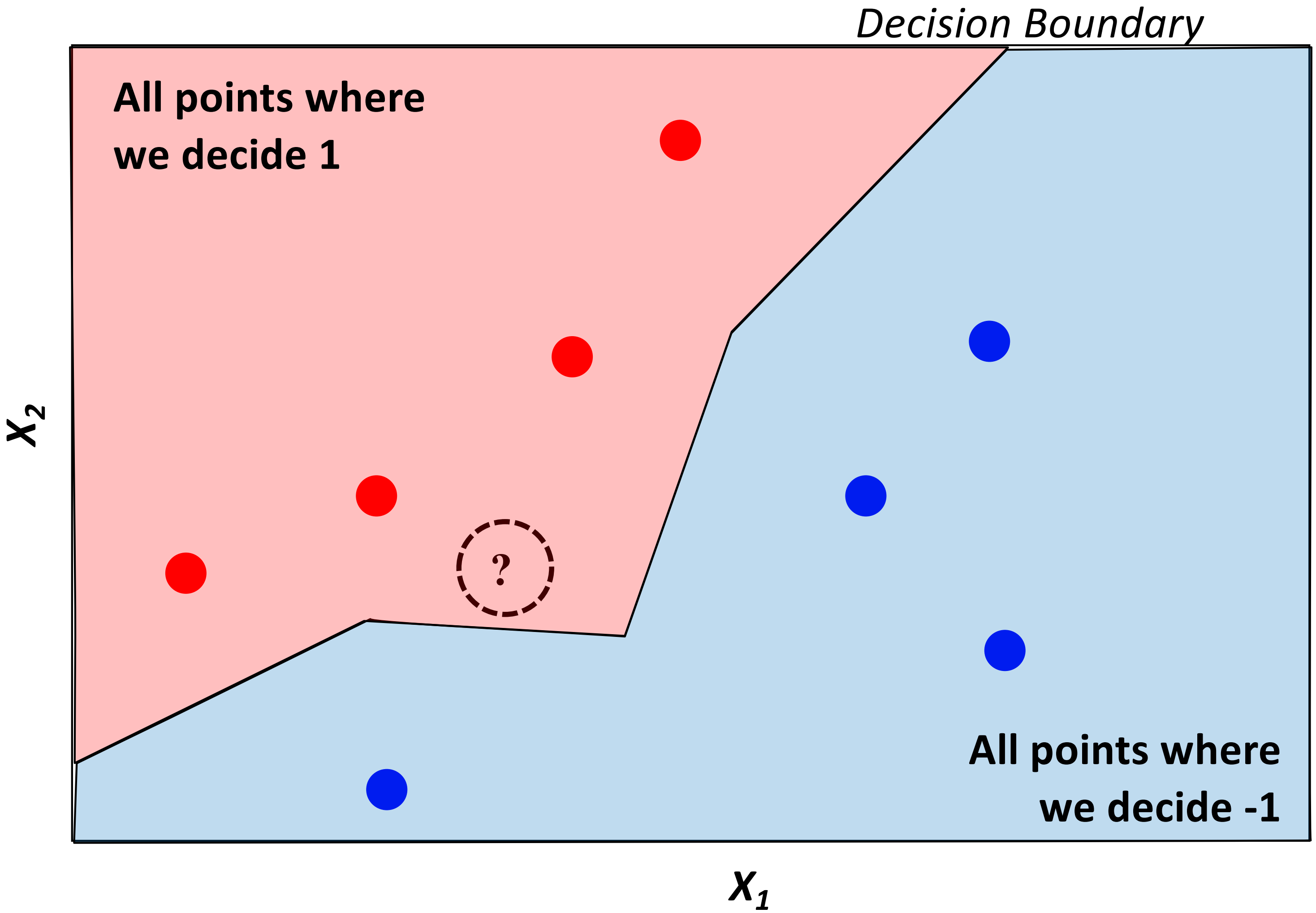
Defines a function *f(x)* implicitly

"Form" is piecewise constant

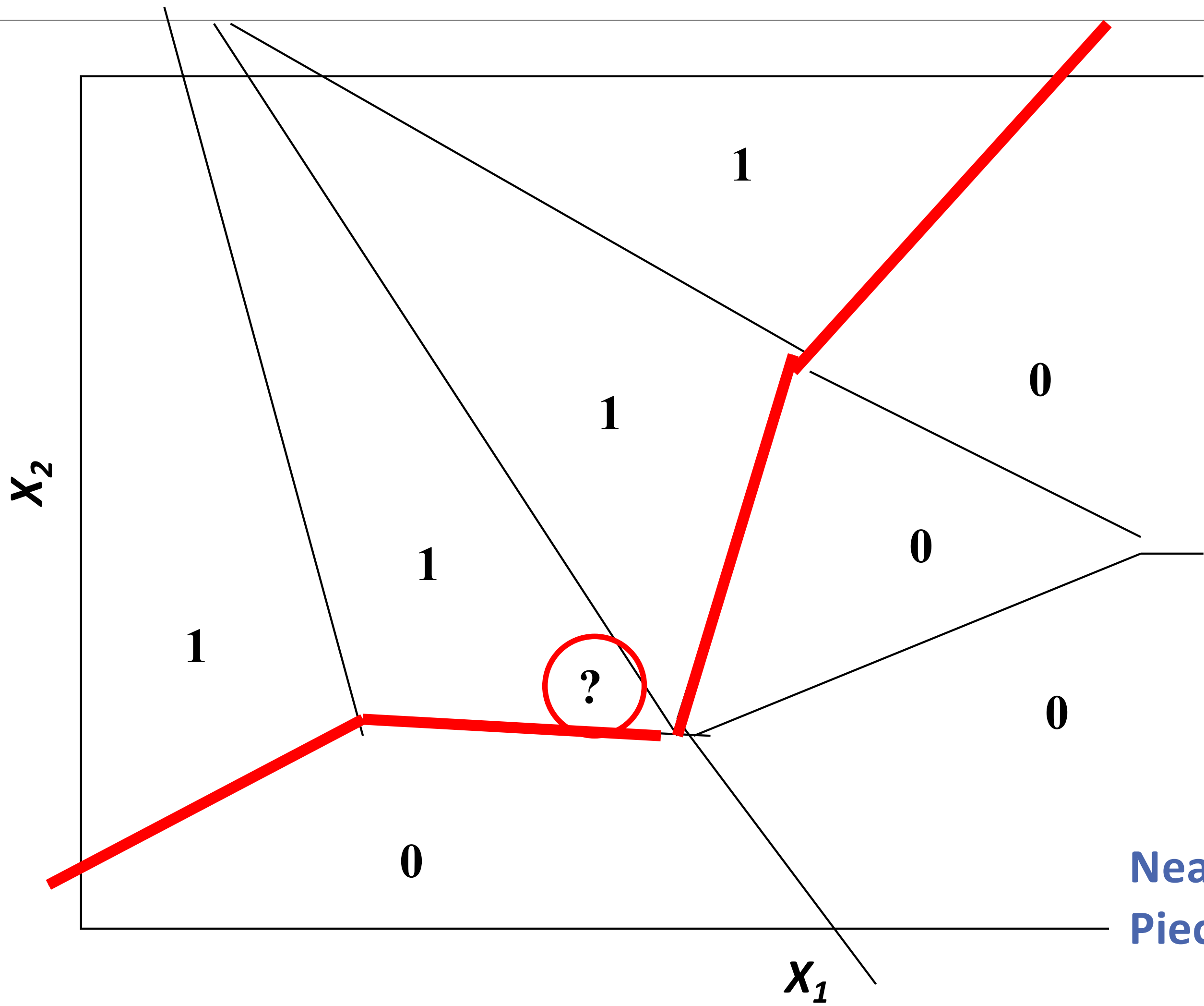
Classification



Nearest Neighbor Classification



Nearest neighbor classifier

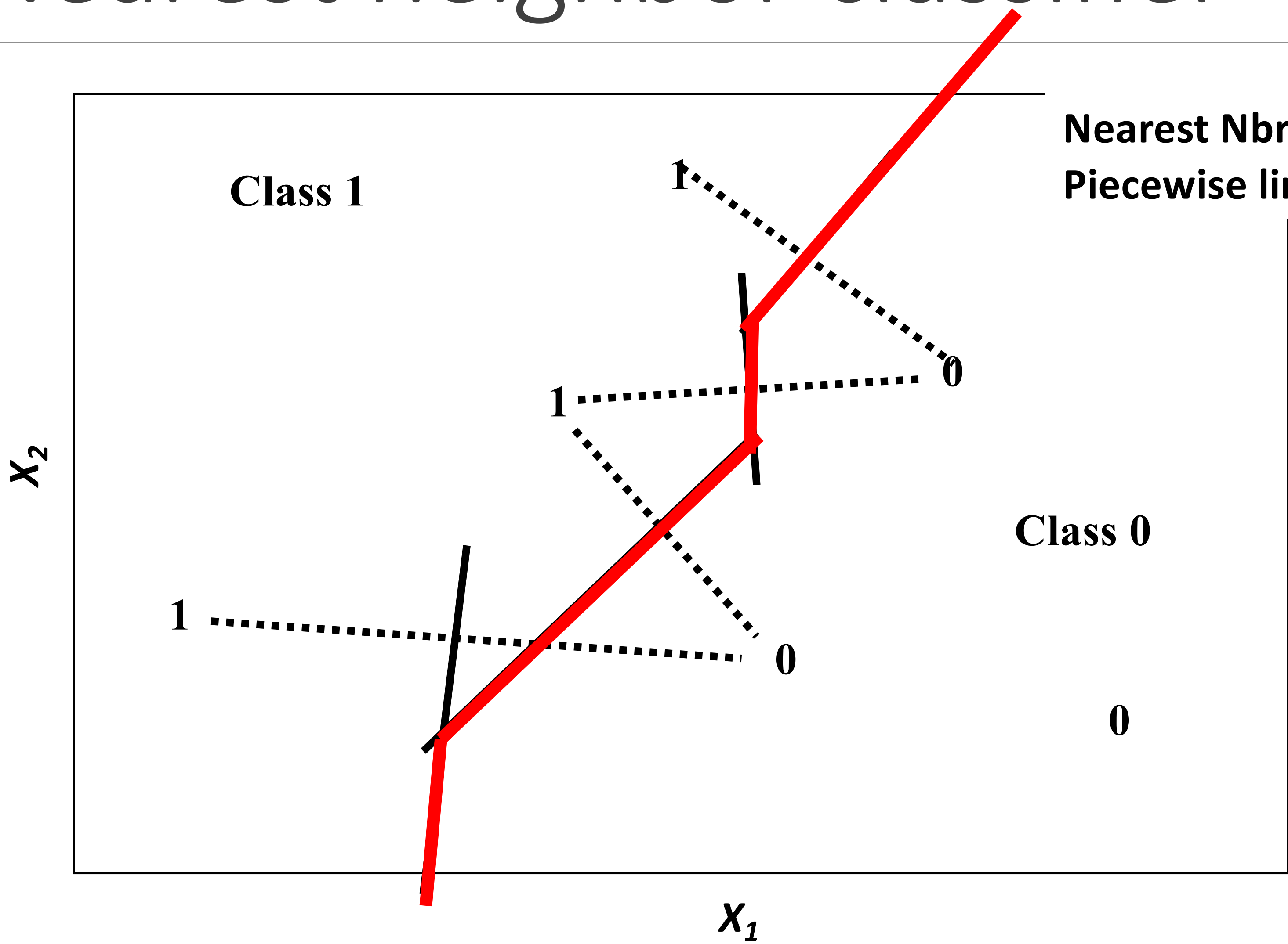


Voronoi tessellation: Each datum is assigned to a region, in which all points are closer to it than any other datum

Decision boundary: Those edges across which the decision (class of nearest training datum) changes

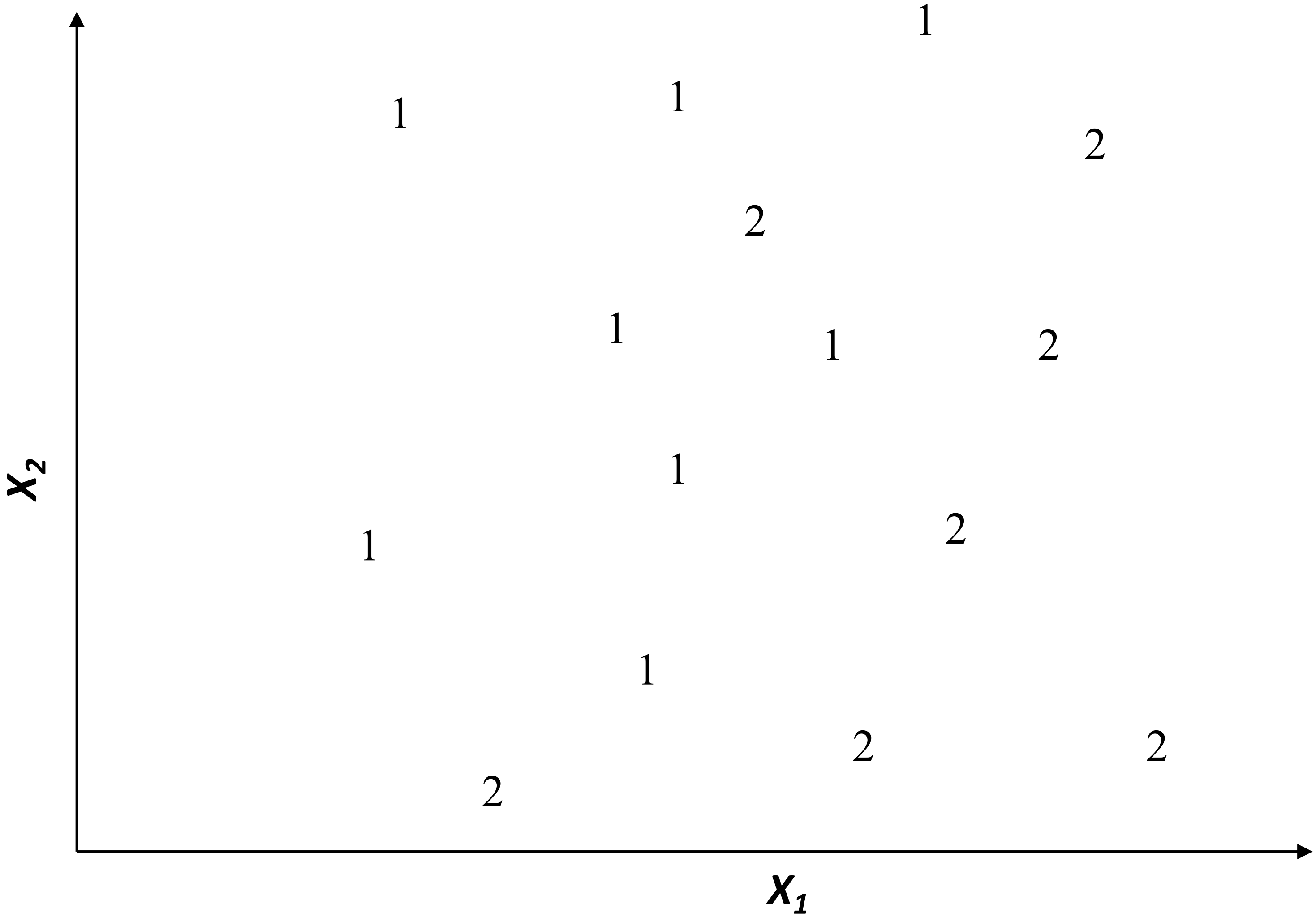
Nearest Neighbor: Piecewise linear boundary

Nearest neighbor classifier

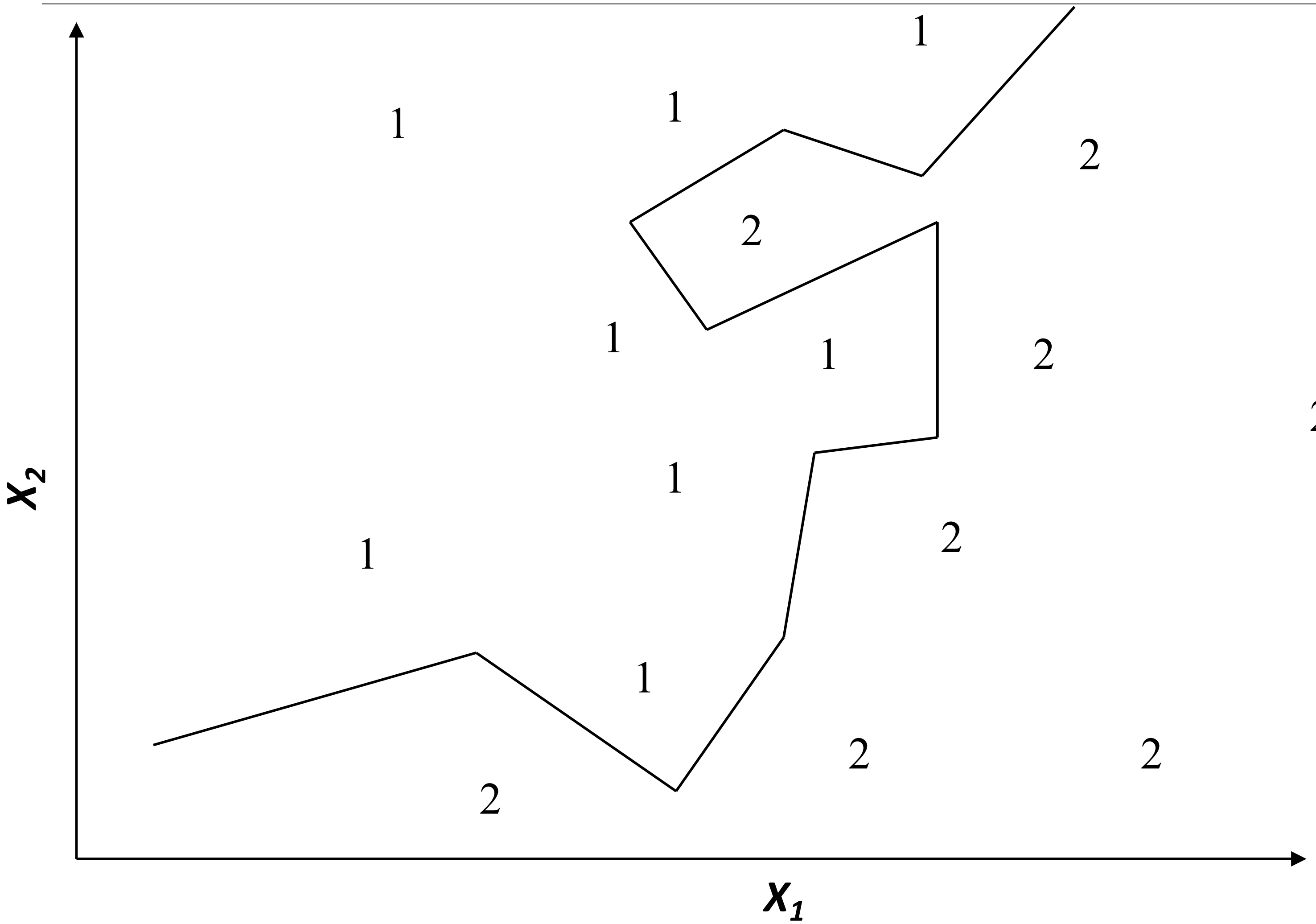


Nearest Nbr:
Piecewise linear boundary

More Data Points



More Complex Decision Boundary



Machine Learning

- Complexity and Overfitting
- Nearest Neighbors
- K-Nearest Neighbors
- Bayes Classifiers

K-Nearest Neighbor (kNN)

Find the k-nearest neighbors to *x* in the data

- i.e., rank the feature vectors according to Euclidean distance, $d\left(x, x^{(j)}\right)^2 = \frac{1}{n}\sum_i \left(x_i - x_i^{(j)}\right)^2$
- select the k vectors which are have smallest distance to x

Regression

- Usually just **average** the y-values of the k closest training examples

Classification

- ranking yields k feature vectors and a set of k class labels
- pick the **majority** class label which is most common in this set ("vote")
- classify x as belonging to this class
- Note: for two-class problems, if k is odd (k=1, 3, 5, …) there will never be any "ties"; otherwise, just use (any) tie-breaking rule

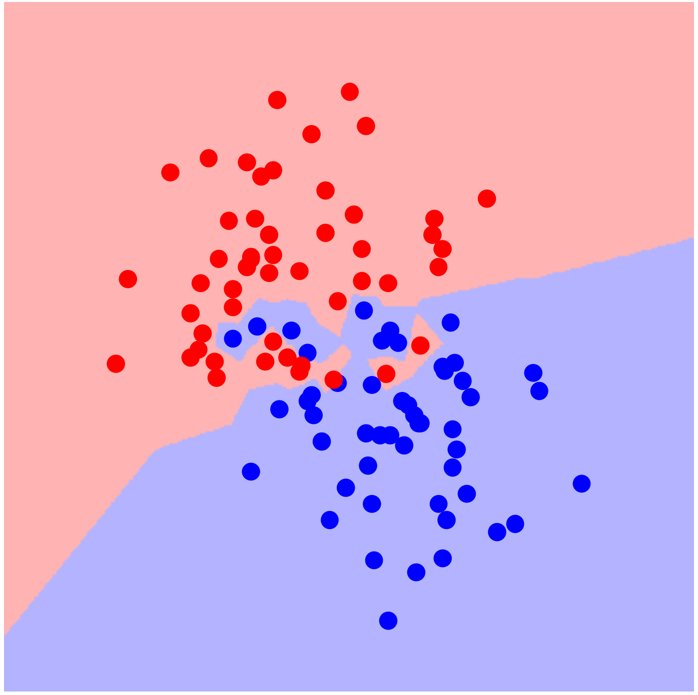
"Training" is trivial

- store training data as a lookup table, and search to classify a new datum

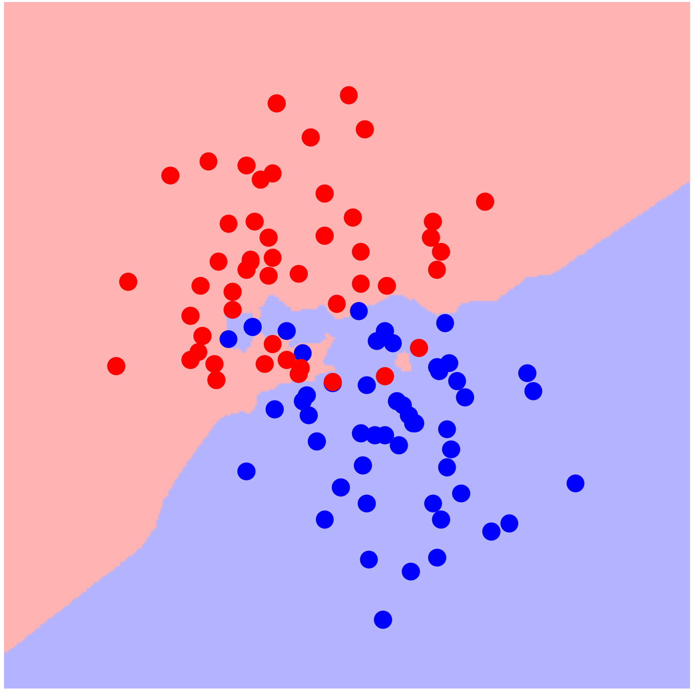
kNN Decision Boundary

Piecewise linear decision boundary

Increasing k “simplifies” decision boundary
- Majority voting means less emphasis on individual points
- K = 1



K = 3

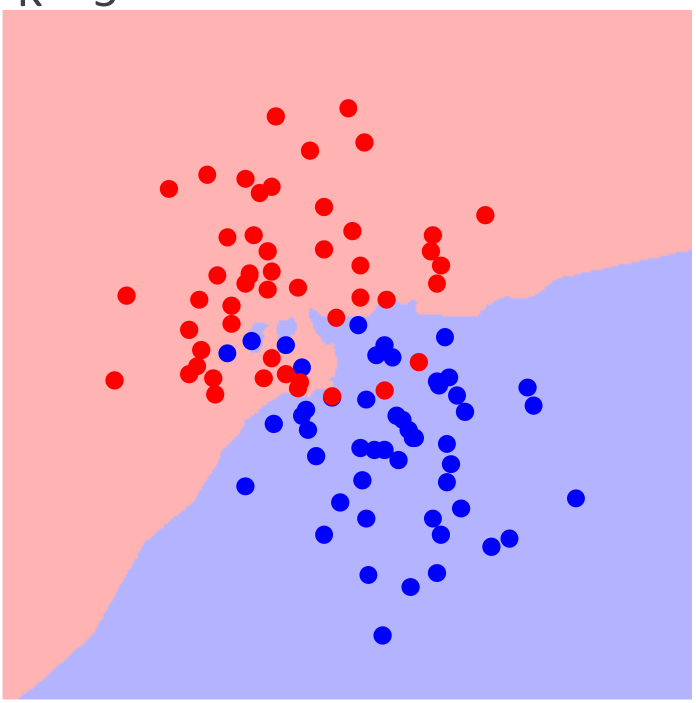


kNN Decision Boundary

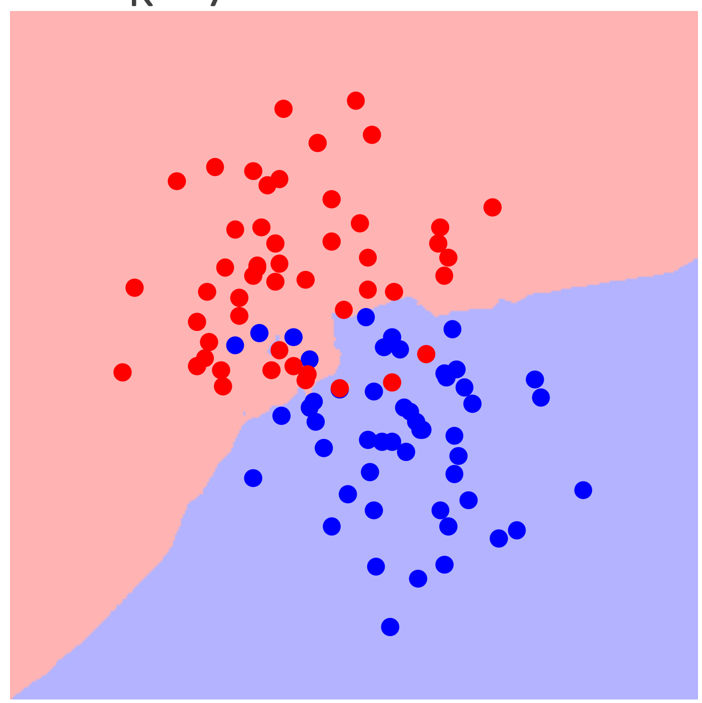
Piecewise linear decision boundary

Increasing k “simplifies” decision boundary

- Majority voting means less emphasis on individual points
- K = 5



K = 7

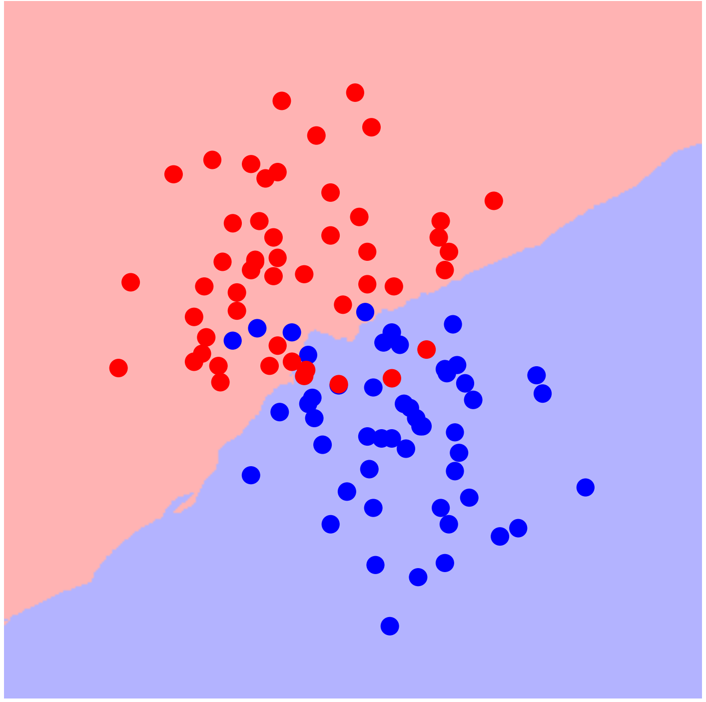


kNN Decision Boundary

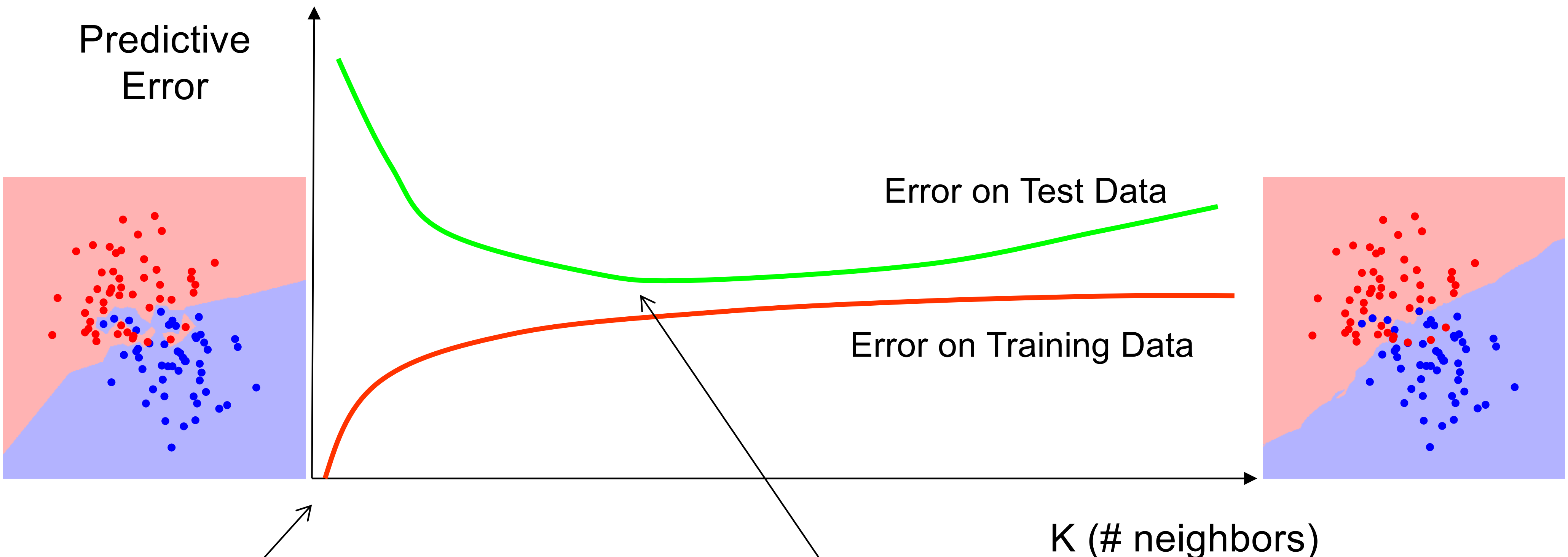
Piecewise linear decision boundary

Increasing k “simplifies” decision boundary

- Majority voting means less emphasis on individual points
- K = 25



Error rates and K



K=1? Zero error!
Training data have been memorized...

Best value of K

Complexity & Overfitting

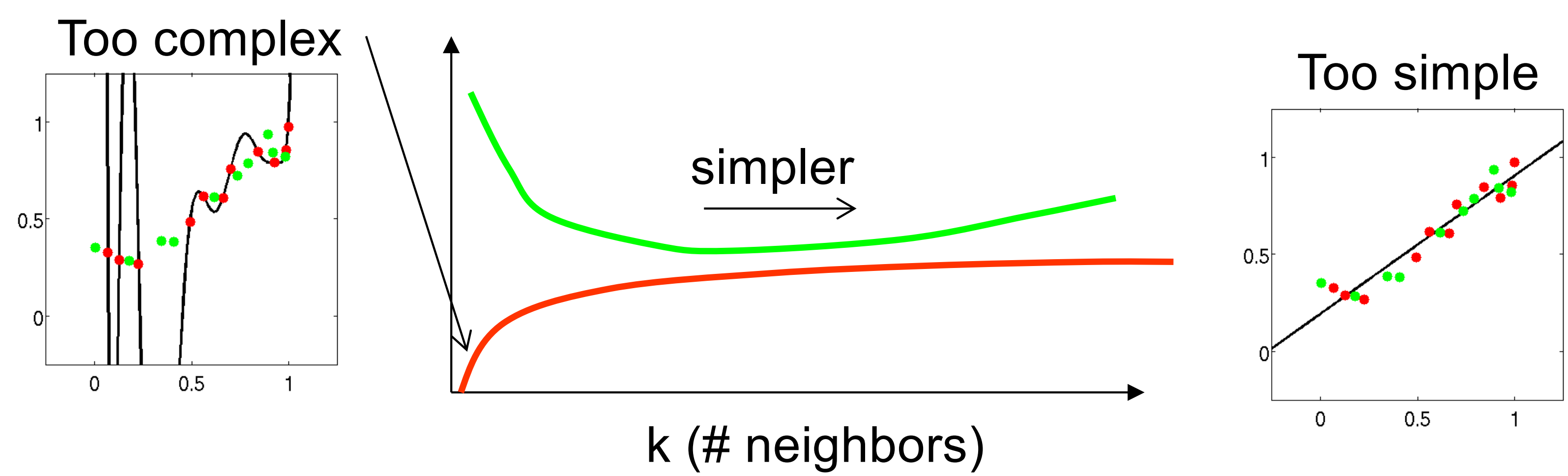
Complex model predicts all training points well

Doesn't generalize to new data points

k = 1 : perfect memorization of examples (complex)

k = m: always predict majority class in dataset (simple)

Can select k using validation data, etc.



K-Nearest Neighbor (kNN) Classifier

Theoretical Considerations

- as k increases
  - we are averaging over more neighbors
  - the effective decision boundary is more "smooth"
- as n increases, the optimal k value tends to increase

Extensions of the Nearest Neighbor classifier

- Weighted distances
  - e.g., some features may be more important;
  - others may be irrelevant

$$d(x, x') = \sqrt{\sum_i w_i (x_i - x'_i)^2}$$

- Fast search techniques (indexing) to find k-nearest points in d-space
- Weighted average / voting based on distance

Summary

K-nearest neighbor models

- Classification (vote)
- Regression (average or weighted average)

Piecewise linear decision boundary

- How to calculate

Test data and overfitting

- Model “complexity” for knn
- Use validation data to estimate test error rates & select k