Artificial Intelligence (CS 401)

Machine Learning for Learning based Agents

Chapter 18: Learning from Examples

Dr. Hafeez UR REHMAN
Dept of Computer Science,
National University of Computer and Emerging Sciences, Peshawar, Pakistan.

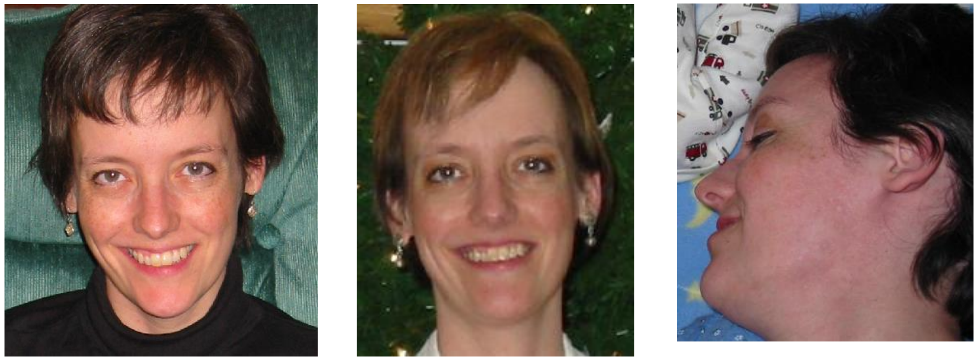
Outline

- What is Machine Learning?
- Different types of learning problems
- Different types of learning algorithms
- Supervised learning
  - Nearest Neighbor
  - **Perceptrons, Multi-layer Neural Networks…..Deep Learning**
  - Decision trees
  - Naïve Bayes
  - Boosting
- Unsupervised Learning
  - K-means
- Applications: e.g., learning to recognize digits, alphabets or some other patterns.

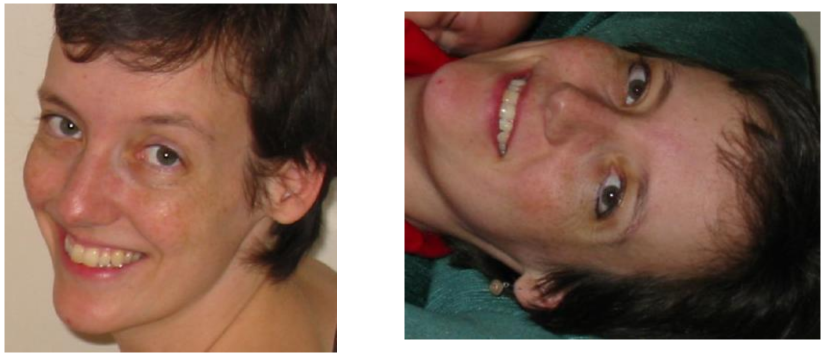
Does Memorization = Learning?

- Example #1: Some baby say **Thomas** learns his mother's face

Memorizes:



But will he recognize:



Thus he can **generalize** beyond what he's seen!

Does Memorization = Learning? (Contd…)

- Example #2: **Nicholas** learns about trucks & combines

Memorizes:

But will he recognize others?

So learning is not just memorization but it involves the **ability to generalize** from labeled examples (in contrast, memorization is trivial, especially for a computer).

Generalization is to apply a <u>learned concept</u> on previously unseen examples based on similarities (features or characteristics).

What is Learning?

- *"Learning denotes **changes in a system** that ... enable a system to **do the same task** … more efficiently the next time."* - Herbert Simon
- *"Learning is **constructing** or **modifying representations** of what is being **experienced**."* - Ryszard Michalski
- *"Learning is making **useful changes** in our minds."* - Marvin Minsky
- Why is it useful for our agent (software/program) to be able to learn?
  - Learning is a **key hallmark of intelligence**
  - Learning is the ability of an agent to **take in real data** and **feedback** to improve performance over time

What is Machine Learning?

- “ **Machine learning** refers to a system capable of **autonomous acquisition** and **integration** of knowledge.”
- Building **machines** that automatically *learn* from experience
  - Important research goal of artificial intelligence
- (Very) small sampling of applications:
  - Data mining **programs** that learn **to detect fraudulent credit card transactions**
  - A program that learn to **precisely segment brain tumors**.
  - A robot that learns to do **surgery**.
  - Programs that learn **to filter spam email**
  - Autonomous vehicles that **learn to drive on public highways**

What is Machine Learning?

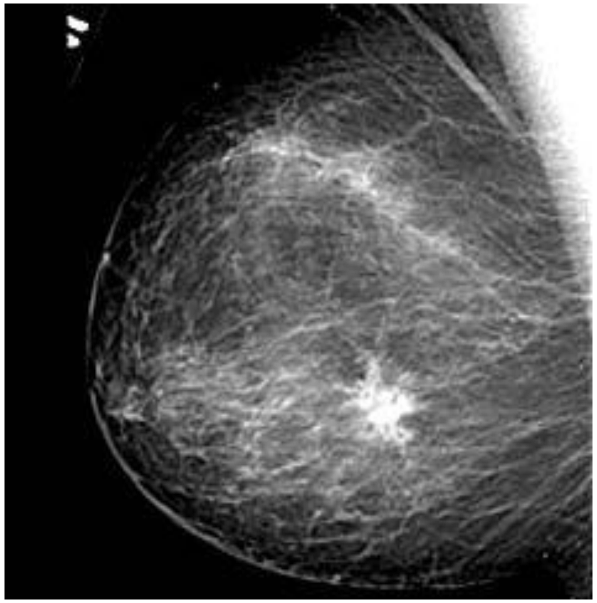
- Given several ***labeled examples*** of a *concept*
  - E.g. trucks vs. non-trucks
- Examples are described by ***features***
  - E.g. *number-of-wheels* (integer), *relative-height* (height divided by width), *hauls-cargo* (yes/no)
- A machine learning algorithm uses these examples to create a ***hypothesis*** that will *predict* the label of new (previously unseen) examples
- Similar to a very simplified form of human learning
- **Hypotheses** can take on many forms e.g. KNN, SVM, Naïve Bayes etc.

Why Machine Learning? (Non Medical App)

- No human experts
  - industrial/manufacturing control
  - mass spectrometer analysis, drug design, astronomic discovery
- Black-box human expertise
  - face/handwriting/speech recognition; **can’t program w/o ML**
  - driving a car, flying a plane
- Rapidly changing phenomena
  - **Designer can’t anticipate all changes over time**
  - credit scoring, financial modeling
  - diagnosis, fraud detection
- Need for customization/personalization
  - the **program must learn first** the taste of the user
  - e.g. personalized news reader, SIRI, CORTANA
  - movie/book recommendation

Why Machine Learning (Medical App)?

- To detect/diagnose malignancy e.g., breast cancer.
  - **Detect Microcalcifications**: tiny calcium deposits whose size ranges from 0.1mm to 5mm.
  - **Detect Lesions**: Star shaped appearance with blurred boundaries.
- Classification of tumors (into benign and malignant):
  - By monitoring oxygen and blood supply to the tumor.
- Surgical interventions
  - Robotic surgery



Types of Learning (Agents)

- Types of learning:

  1. **Supervised learning**
     - Learning a mapping from a given set of **inputs** to a **target variable**
       - **Classification**: target variable is discrete (e.g., spam email)
       - **Regression**: target variable is real-valued (e.g., stock market)

  2. **Unsupervised learning**
     - No target variable provided
       - Clustering: grouping data into K groups e.g. Taxi agent building the concept of good traffic days and bad traffic days

  3. **Other types of learning**
     - **Reinforcement learning (based on reward):** e.g., game-playing agent
     - Learning to rank, e.g., document ranking in Web search
     - And many others….

Supervised Learning

Inductive (Supervised) learning

- Let x represent the input vector of attributes a.k.a features

- Let f(x) represent the value of the target variable for x
  - **The implicit mapping from x to f(x) is unknown to us**
  - **We just have training data pairs, D = {x, f(x)} available**

- We want to learn a mapping from x to f, i.e.,

  $h(x; \theta)$ is "close" to f(x) for all training data points x

  $h(x; \theta)$ is also called hypothesis function

  θ are the parameters of our predictor h(..)

- Examples:
  - $h(x; \theta) = \text{sign}(w_1x_1 + w_2x_2 + w_3x_3)$
  - $h_k(x)$ = (x1 OR x2) AND (x3 OR NOT(x4))
  - h(x) = KNN(x)

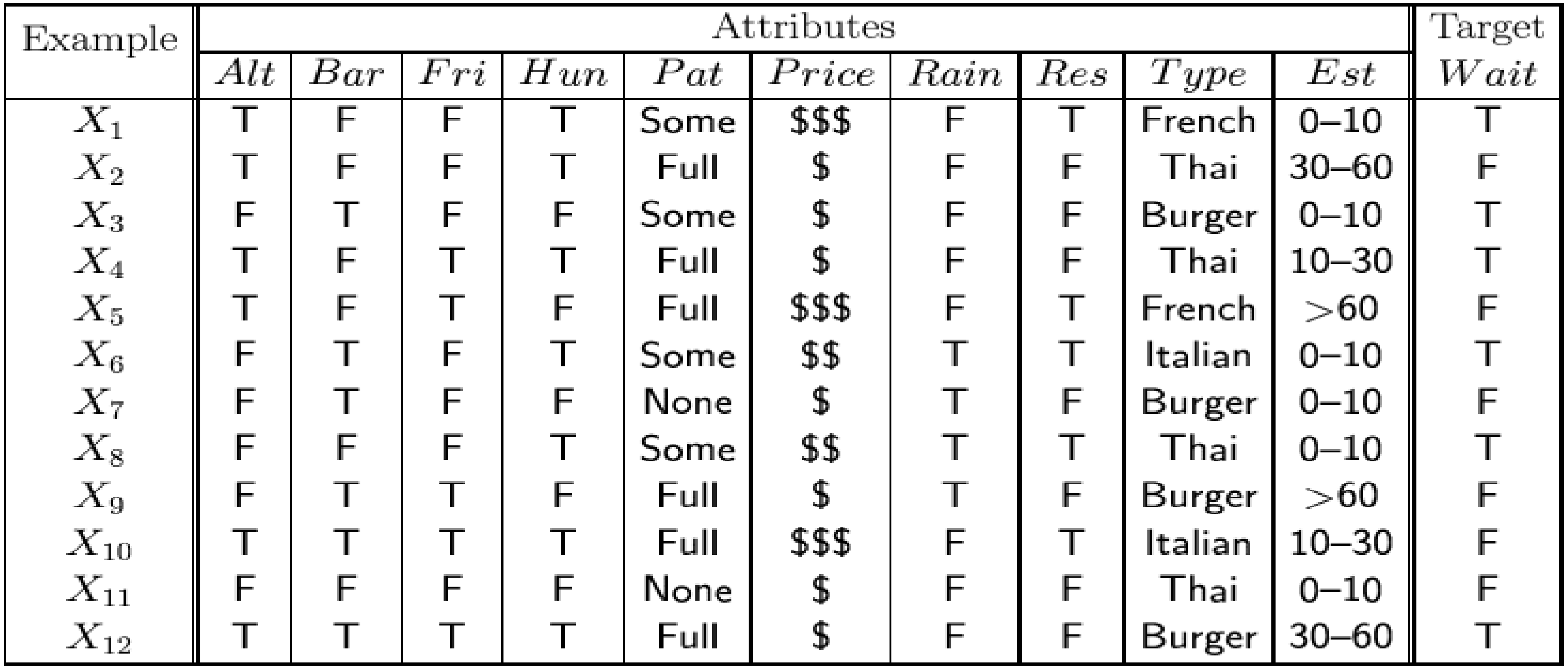
Simple illustrative learning problem

Problem:

Decide whether to wait for a table at a restaurant, based on the following **attributes** (or features/characteristics), (binary classification problem):

1. **Alternate**: is there an alternative restaurant nearby?
2. **Bar**: is there a comfortable bar area to wait in?
3. **Fri/Sat**: is today Friday or Saturday?
4. **Hungry**: are we hungry?
5. **Patrons**: number of people in the restaurant (None, Some, Full)
6. **Price**: price range ($, $$, $$$)
7. **Raining**: is it raining outside?
8. **Reservation**: have we made a reservation?
9. **Type:** kind of restaurant (French, Italian, Thai, Burger)
10. **Wait Estimate**: estimated waiting time (0-10, 10-30, 30-60, >60)

Training Data for Supervised Learning



Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	\$\$\$	F	T	French	0–10	T
X_2	T	F	F	T	Full	\$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	\$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	\$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	\$\$\$	F	T	French	>60	F
X_6	F	T	F	T	Some	\$\$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	\$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	\$\$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	\$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	\$\$\$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	\$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	\$	F	F	Burger	30–60	T

Terminology

- Attributes
  - Also known as **features**, variables, independent variables, covariates. These are going to help characterize the target variable.

- Target Variable
  - Also known as goal predicate, dependent variable, **target class etc.**

- Classification
  - Also known as discrimination, supervised classification. It is the ability of a program to characterize (classify) something.

- Error function
  - Objective function, loss function, and we want to minimize it.

Empirical Error Functions

- Empirical error function:

  $E(h) = \Sigma_x$ distance$[h(x; \theta), f]$

  e.g., **distance = squared error if h and f are real-valued (regression)**
  distance = delta-function if h and f are categorical (classification)

  Sum is over all training pairs in the training data D

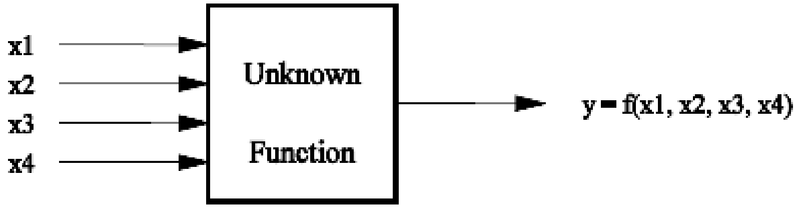
  In learning, we get to choose

  1. what class of functions h(..) that we want to learn
     – potentially a huge space! **(“hypothesis space”)**

  2. what error function/distance to use
     - should be chosen to reflect real “loss” in problem
     - but often chosen for mathematical/algorithmic convenience

Learning Boolean Functions

- Given examples of the function, can we learn the function?
- How many **Boolean functions** can be defined on **d attributes**?
  - Boolean function = **Truth table + column for target function (binary)**
  - Truth table has **2^d rows** (d is the length of feature vector)
  - So, there are **2 to the power of 2^d** different Boolean functions we can define (!)
  - This is the size of our hypothesis space
  - E.g., d = 6, there are 18.4 x 10^{18} possible Boolean functions
  - Consider an example of 4 features:



Example	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Training Data

Learning Boolean Functions

- There are **2^{16}= 65536 possible Boolean functions** over **four** input features. We can't figure out which one is correct until we've seen every possible input-output pair.
- After seeing 7 examples, we still have **2^9 possibilities**.
- Observations:
  - **Huge hypothesis spaces –> directly searching over all functions is impossible**
  - Given a small data (n pairs) our learning problem may be under constrained (doesn't reflect the complexity of original data)

Ockham's Razor Principle

- **Ockham's razor:** if **multiple candidate functions** all **explain the data equally** well, pick **the simplest explanation** (least complex function)

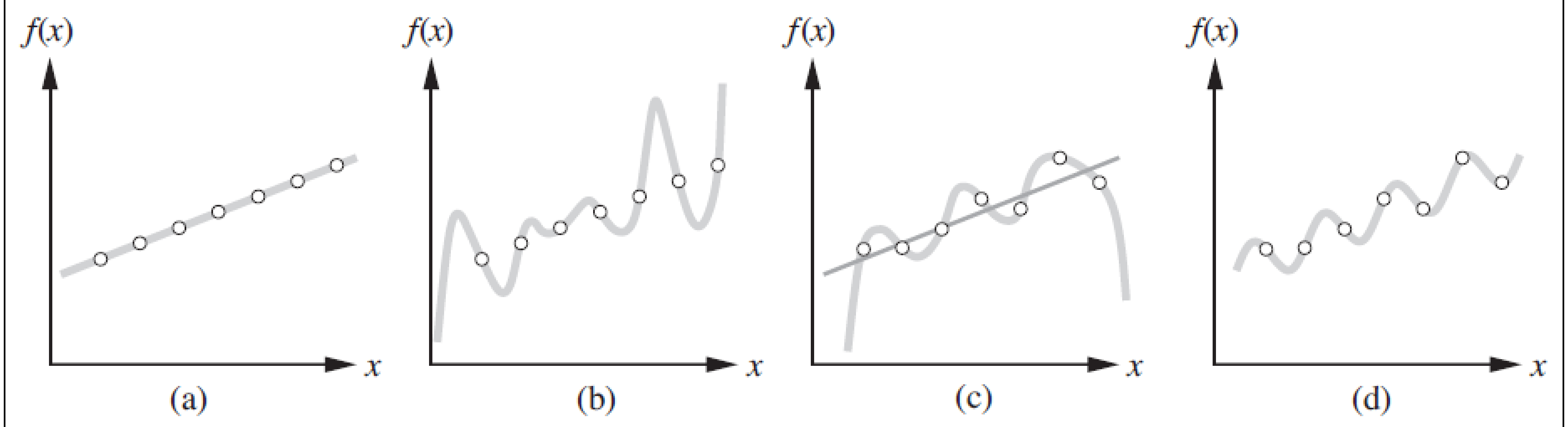


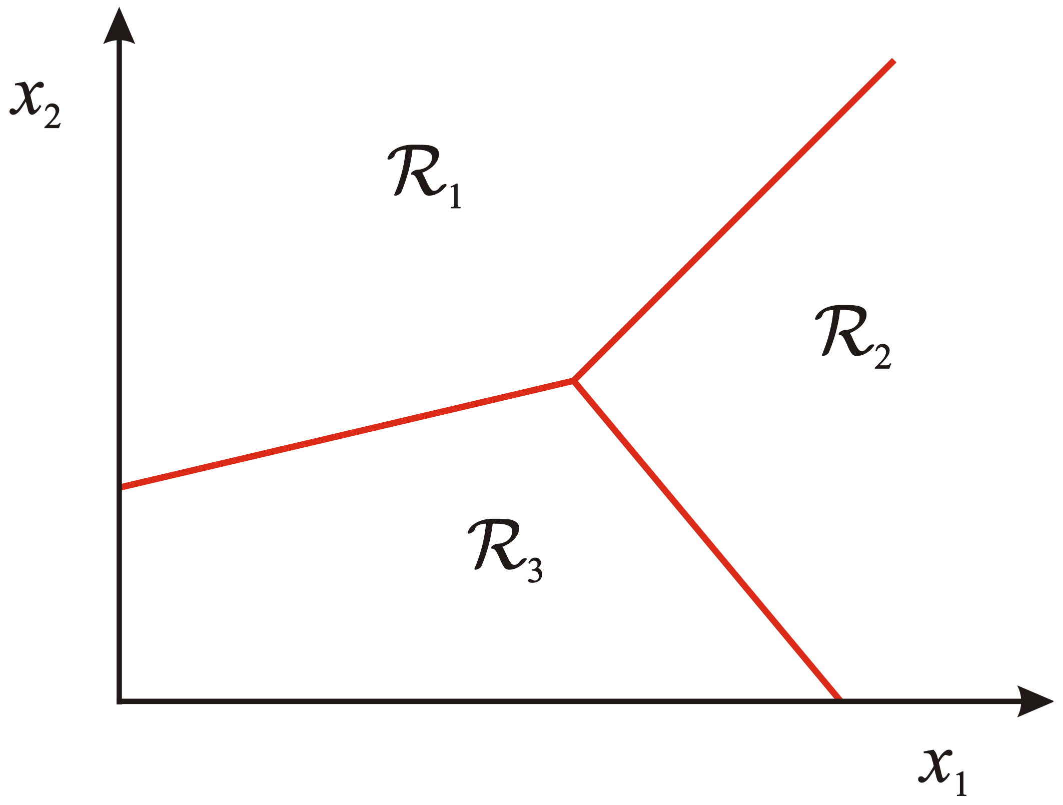
Figure 18.1 (a) Example $(x, f(x))$ pairs and a consistent, linear hypothesis. (b) A consistent, degree-7 polynomial hypothesis for the same data set. (c) A different data set, which admits an exact degree-6 polynomial fit or an approximate linear fit. (d) A simple, exact sinusoidal fit to the same data set.

Classification in Euclidean Space

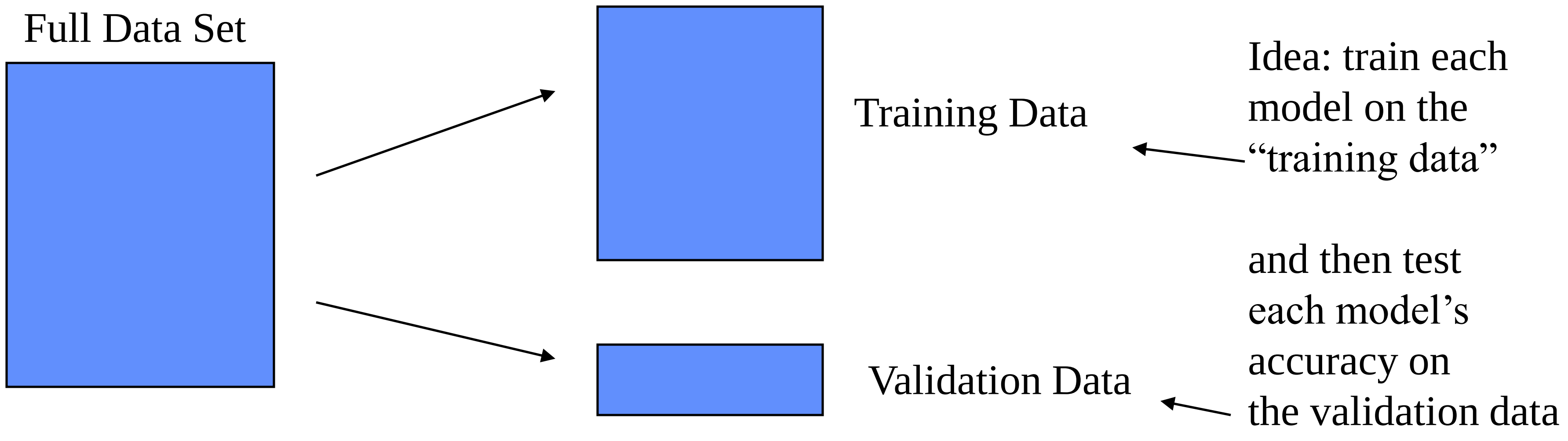
- A **classifier** is a **partition of the space $\underline{x}$ (feature space) into disjoint decision regions**
  - Each region has a label attached
  - Regions with the same label need not be contiguous
  - For a new test point, find what decision region it is in, and predict the corresponding label
- Decision boundaries = boundaries between decision regions
- We can **characterize** (the type of) a classifier by the **equations for its decision boundaries**
- Learning a classifier ⇔ **searching for the decision boundaries that optimize our objective function**

Classification

- Assign input vector e.g. *x = [x1,x2]*, to one of two or more classes
- Any decision rule divides input space into *decision regions* separated by *decision boundaries*



Training and Validation Data

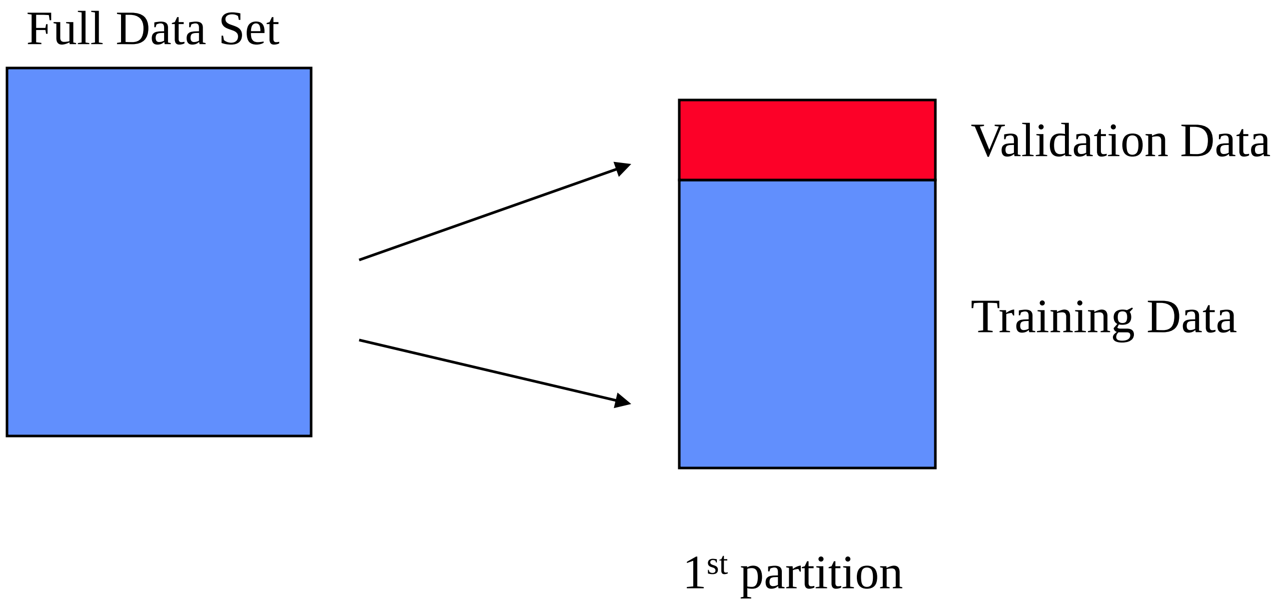

Full Data Set
Training Data
Validation Data
Idea: train each model on the "training data"
and then test each model's accuracy on the validation data

- If you perform very well on the training set, then you have successfully found features that separate your training set well.
- We perform cross validation to 'validate' that our training examples are *representative* of the real-world dataset.
- If we can build a model on our training set, and use that model to successfully predict an independent set (the test set), we can say with good confidence that this model will generalize to the real-world set of data.

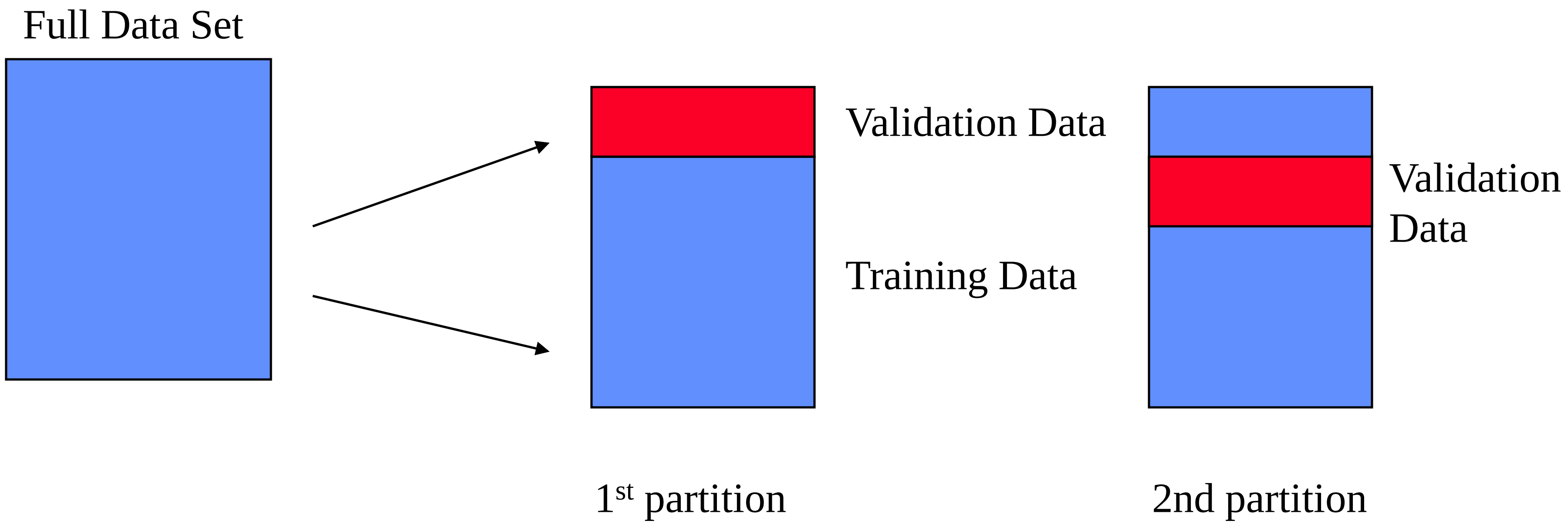
The v-fold Cross-Validation Method

- Why just choose one particular 90/10 "split" of the data?
  - In principle we could do this multiple times
- "v-fold Cross-Validation" (e.g., v=10)
  - **Randomly partition** your full data set into v disjoint subsets (each roughly of size n/v, n = total number of training data points)
    - for i = 1:10 (here v = 10)
      - train on 90% of data,
      - Acc(i) = accuracy on other 10%
    - end
    - **Cross-Validation-Accuracy = $1/v\ \Sigma_i$ Acc(i)**
  - choose the method (**hypothesis**) with the highest cross-validation accuracy
  - common values for v are 5 and 10
  - Can also do "leave-one-out" where v = n

Disjoint Validation Data Sets



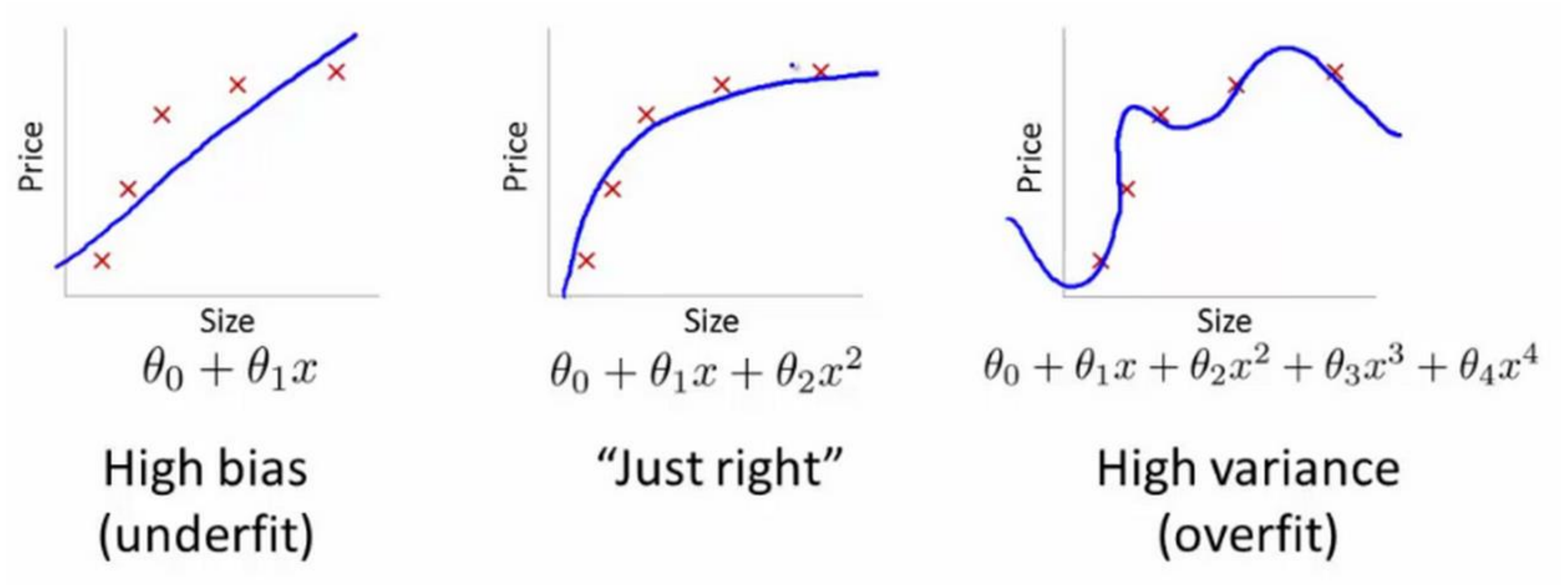
Disjoint Validation Data Sets



More on Cross-Validation

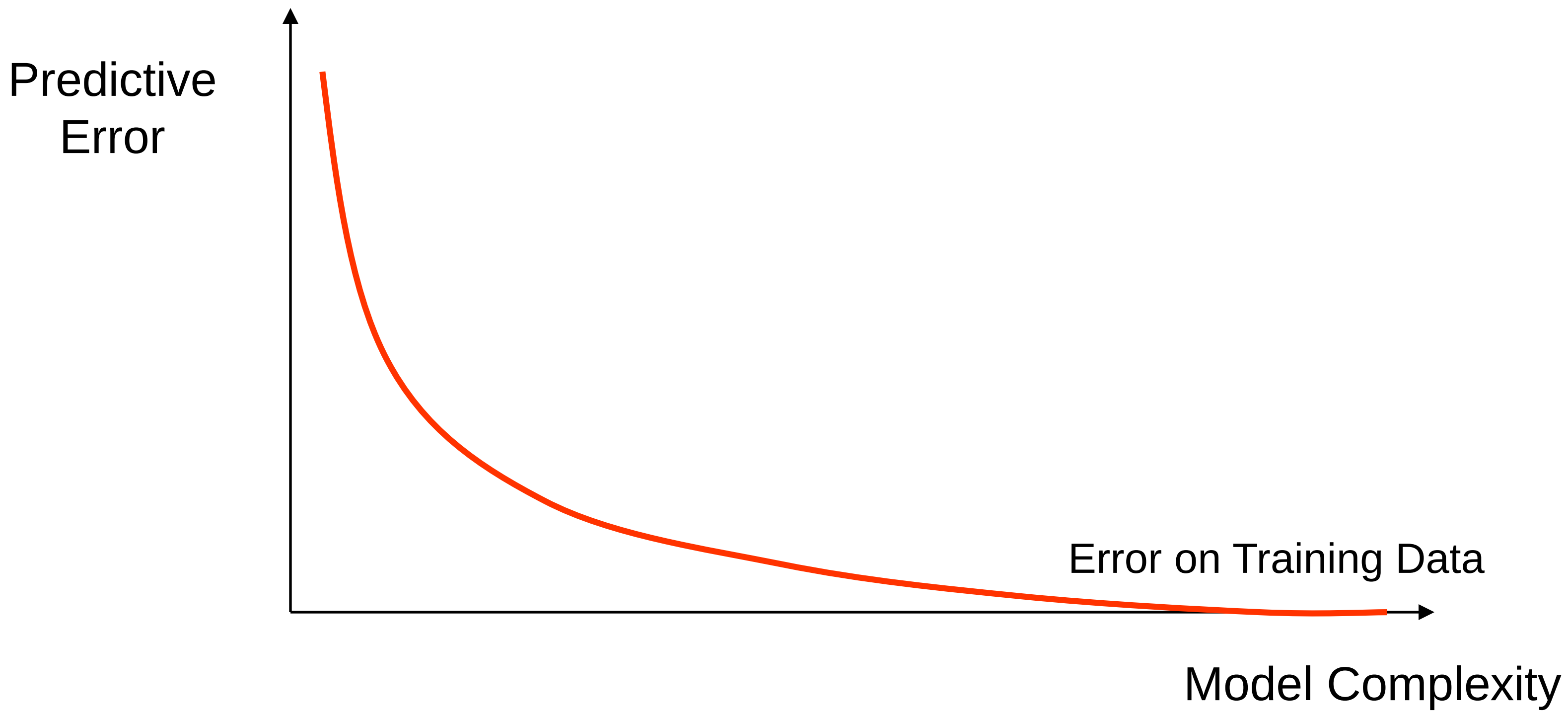
- Notes
  - cross-validation generates an approximate estimate of how well the learned model will do on “unseen” data
  - by averaging over different partitions it is more robust than just a single train/validate partition of the data
  - “v-fold” cross-validation is a generalization
    - partition data into disjoint validation subsets of size n/v
    - train, validate, and average over the v partitions
    - e.g., v=10 is commonly used
  - v-fold cross-validation is approximately v times computationally more expensive than just fitting a model to all of the data

Underfitting & Overfitting

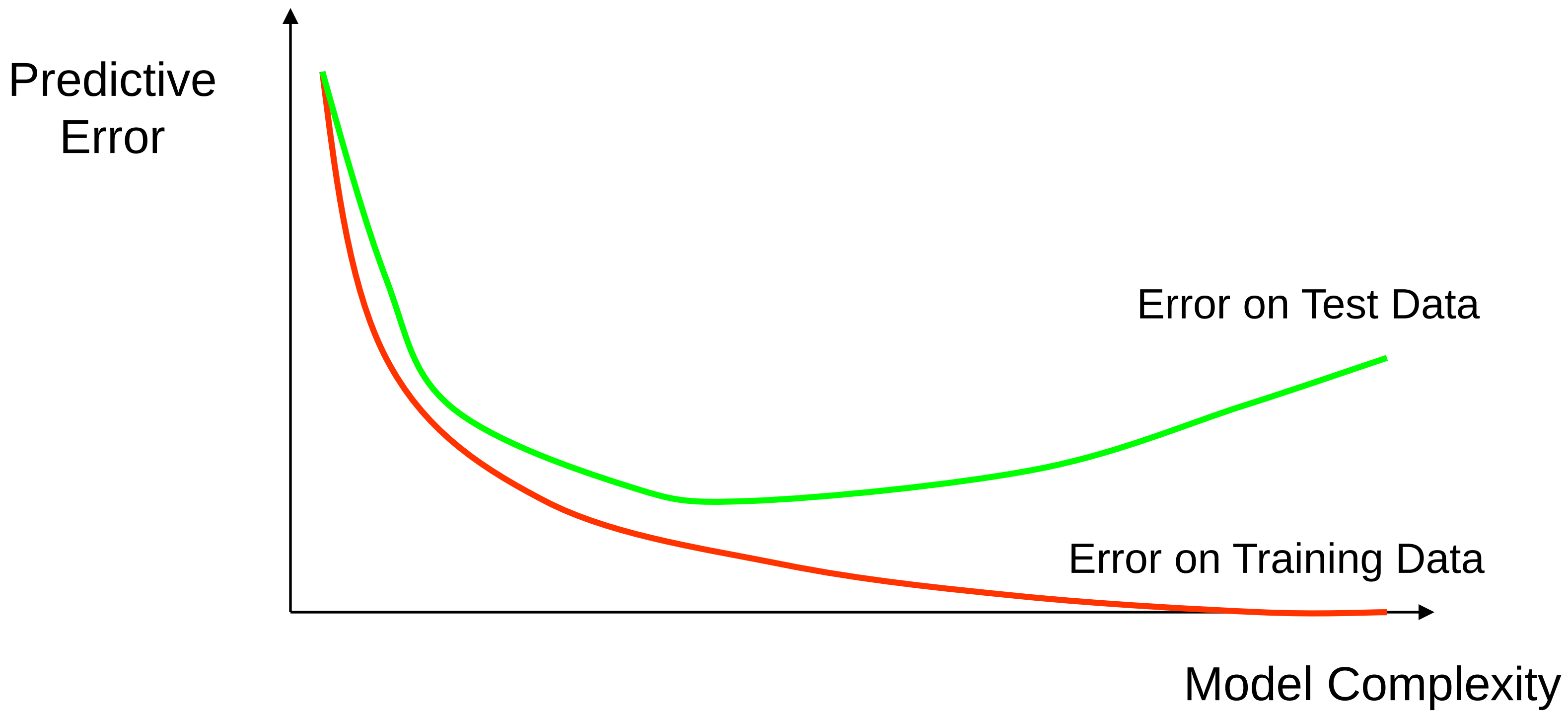


- Both overfitting and underfitting lead to **poor predictions** on new data sets.
- To know whether you have a too high bias or a too high variance, you view the phenomenon in terms of **training** and **test** errors.

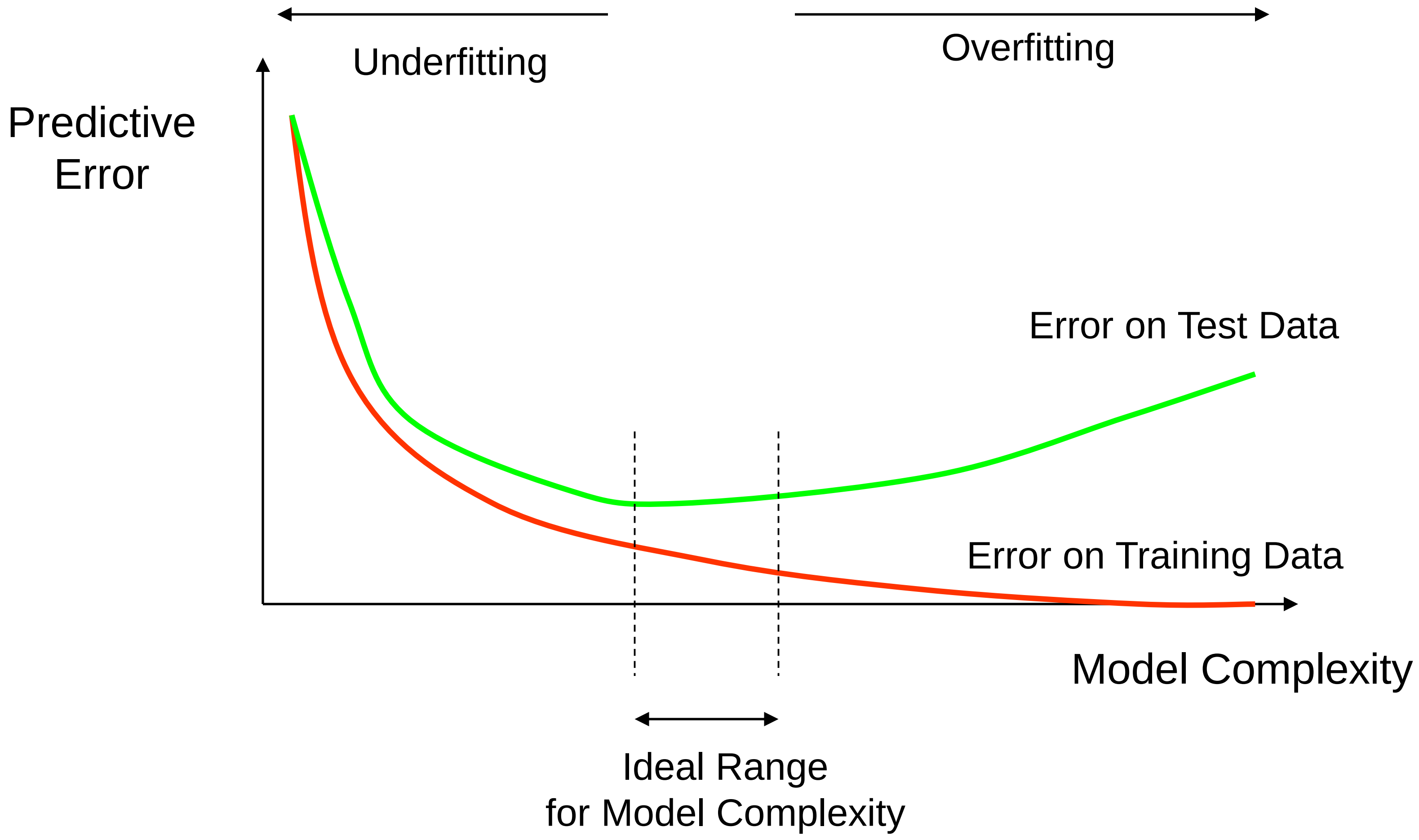
How Overfitting affects Prediction



How Overfitting affects Prediction



How Overfitting affects Prediction

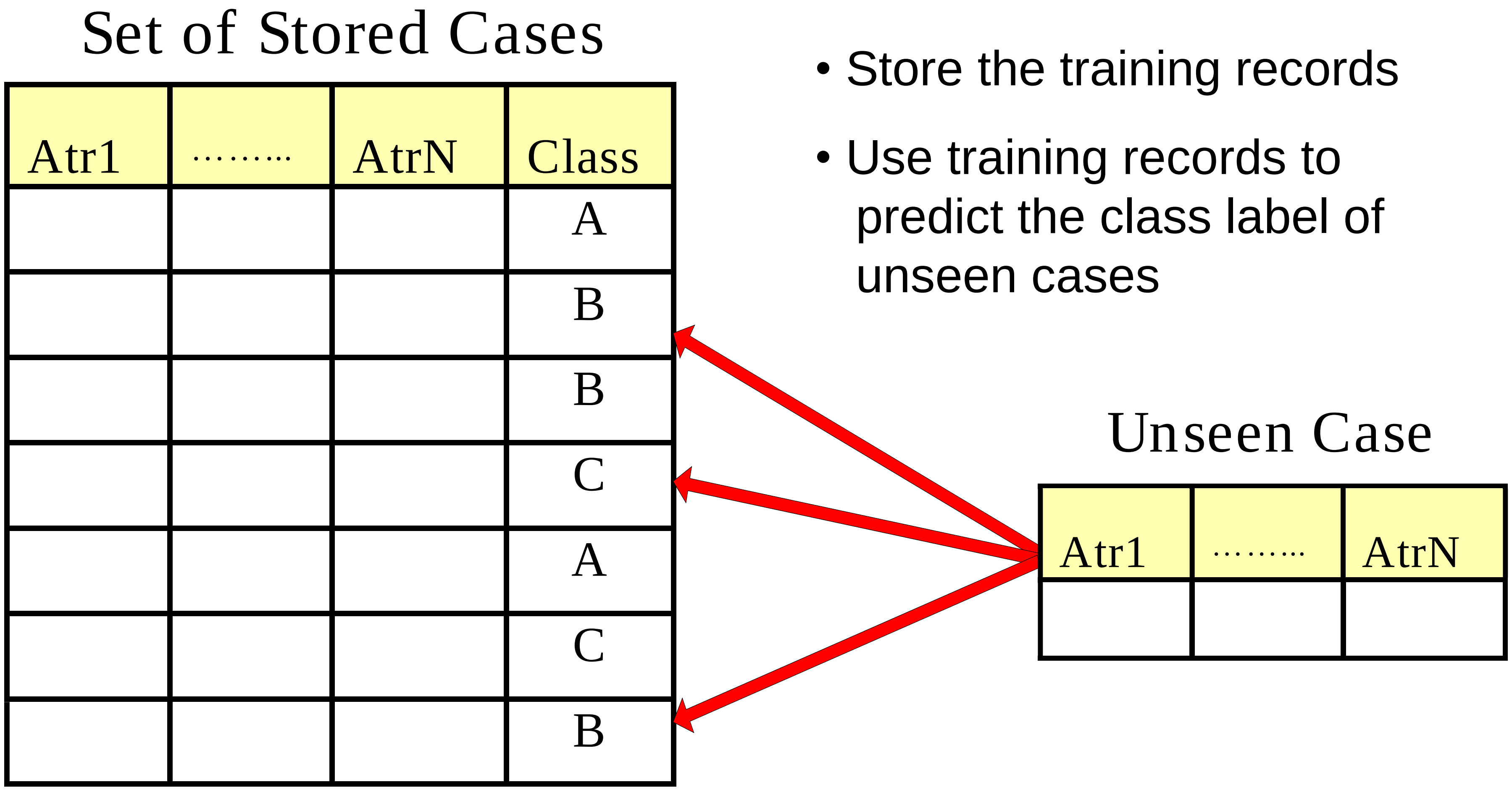


Remedy

- If an algorithm is suffering from **high variance**:
  - **more data** will probably help
  - otherwise **reduce the model complexity** so that it generalizes well on test data

- If an algorithm is suffering from **high bias**:
  - **increase** the model complexity

Instance-Based Classifiers



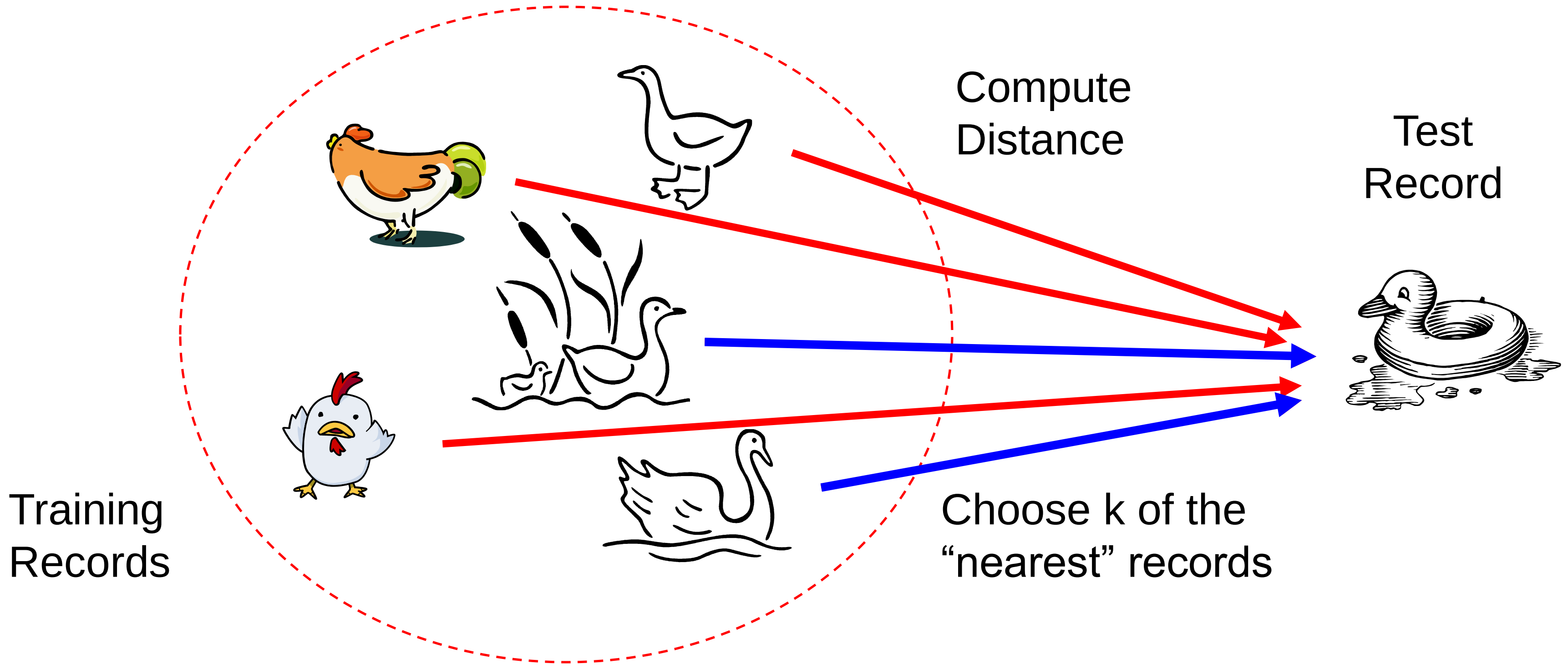
- Store the training records
- Use training records to predict the class label of unseen cases

Instance Based Classifiers

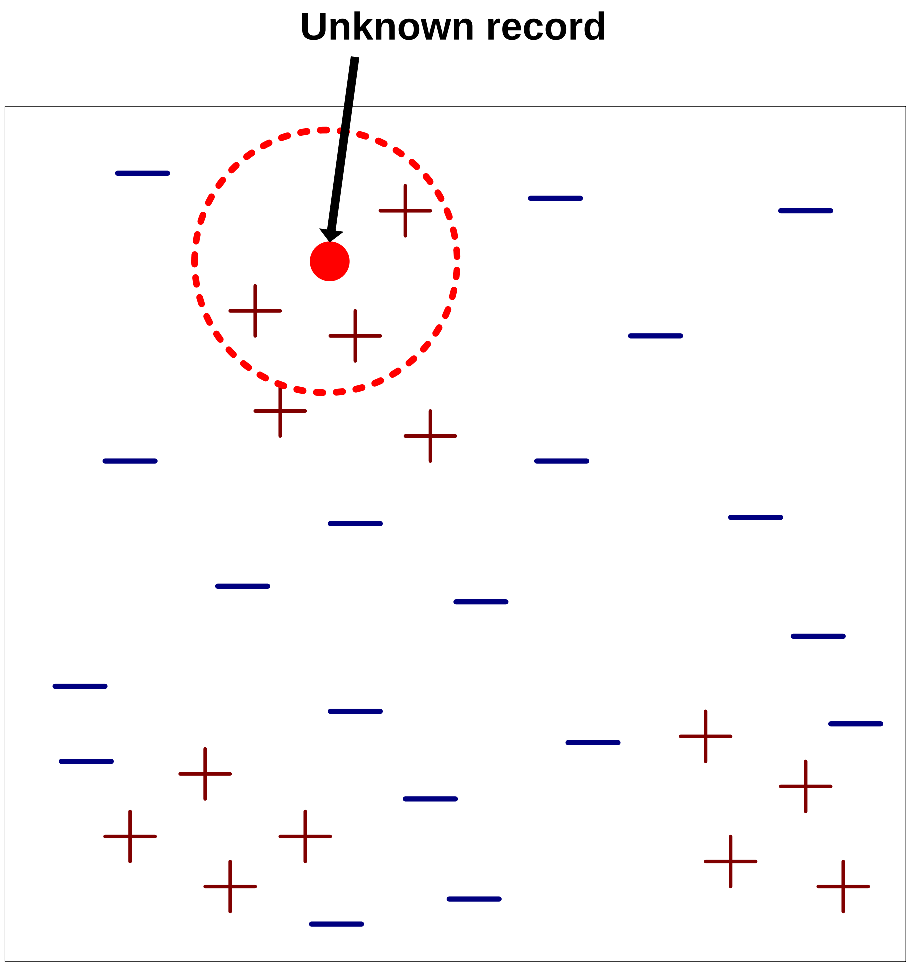
- **Examples:**
  - **Lookup Table**
    - Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly
  - **Nearest neighbor**
    - Uses k “closest” points (nearest neighbors) for performing classification

Nearest Neighbor Classifiers

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck

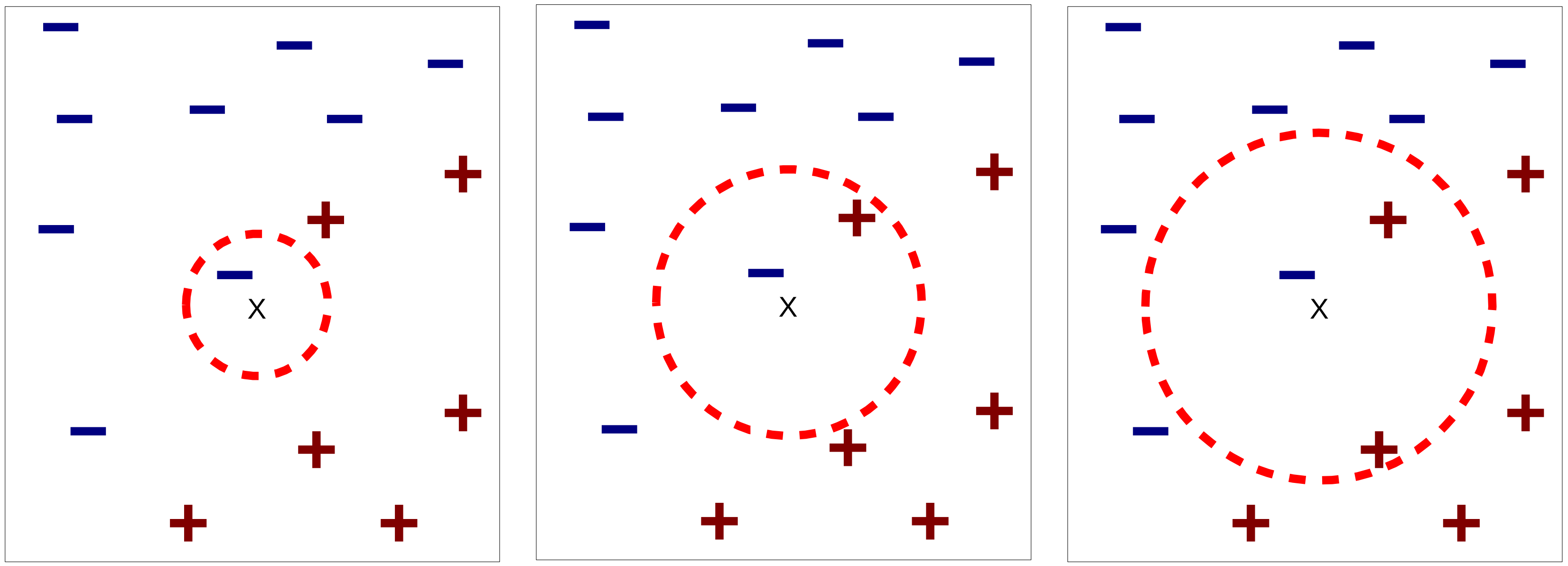


Nearest-Neighbor Classifiers



- Requires three things
  1. The set of **stored records**
  2. **Distance Metric** to compute distance between records
  3. The **value of *k***, the number of nearest neighbors to retrieve
- To classify an unknown record:
  1. Compute distance to other training records
  2. Identify *k* nearest neighbors
  3. Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

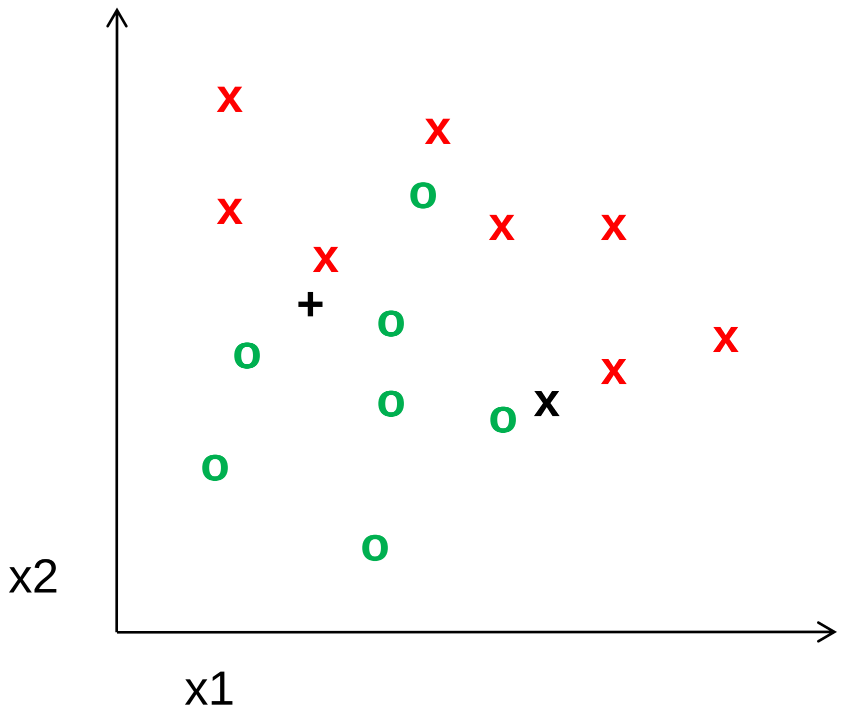
Definition of Nearest Neighbor



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

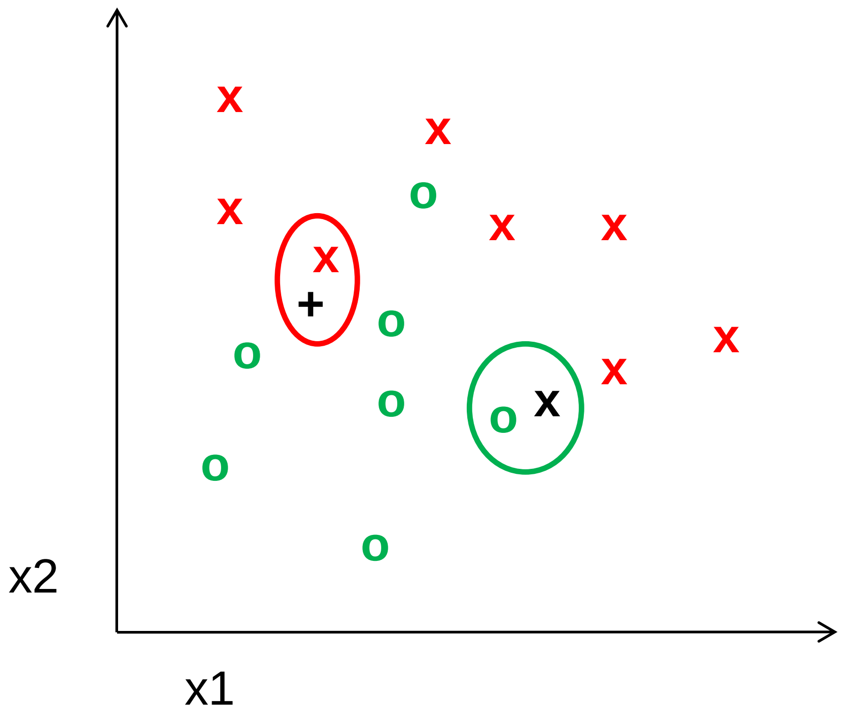
K-nearest neighbor



1-nearest neighbor

+ to Red

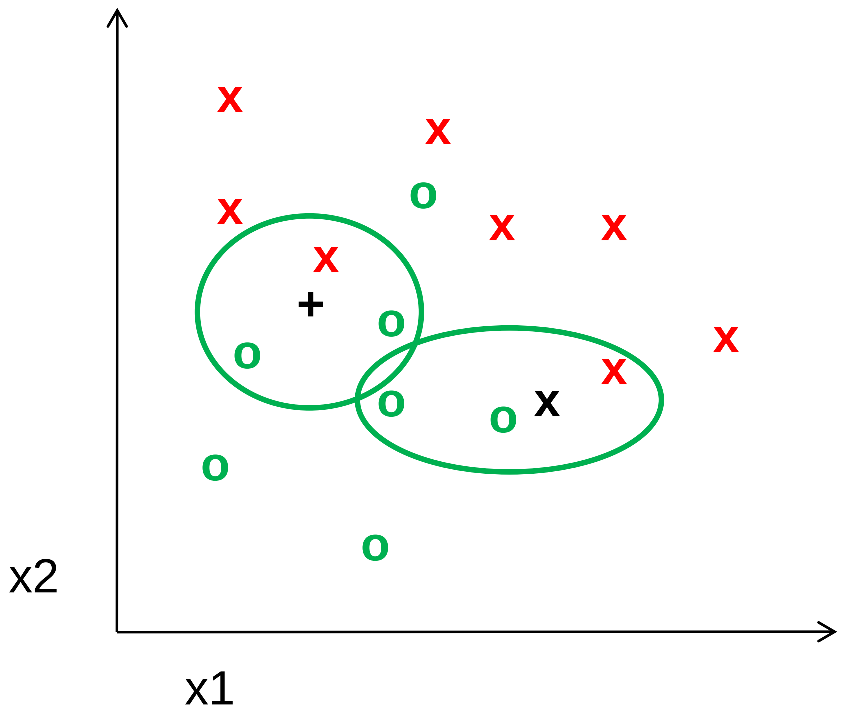
X to Green



3-nearest neighbor

+ to Green

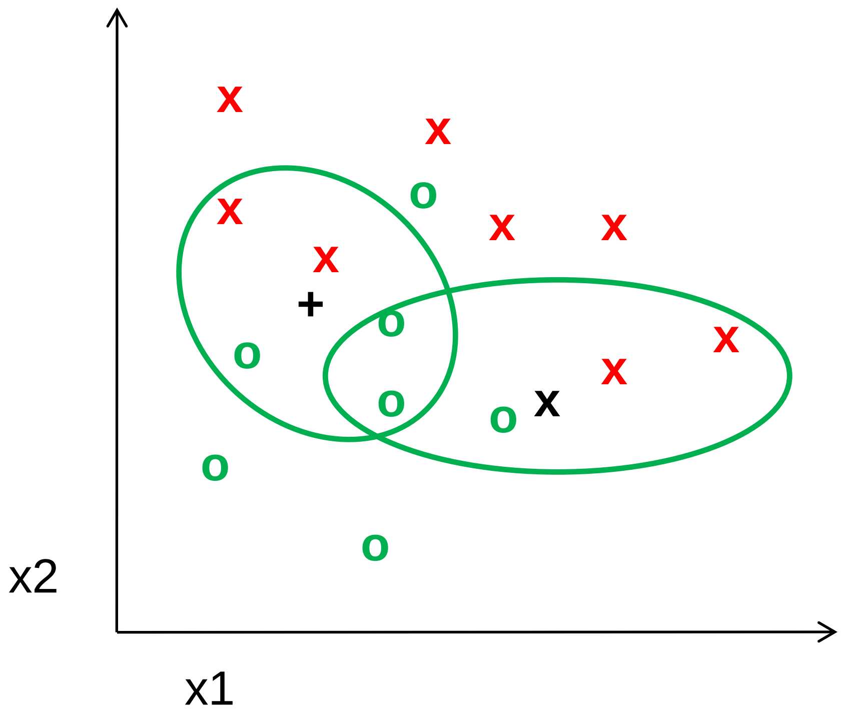
X to Green



5-nearest neighbor

+ to Green

X to Green



Nearest Neighbor Classification

- Compute distance between two points in the feature space:
  - **Hamming** distance: Use if **attributes are binary**. Just count the mutually different values in both vectors r (query) and s (example).
  - **Euclidean** distance: Similar attributes e.g., width, height, depth

$$d(r,s)=\sqrt{\sum_i (r_i-s_i)^2}$$

  - **Manhattan** distance: Dissimilar attributes e.g., age, weight, gender

$$d(r,s)=\sum_i |(r_i-s_i)|$$

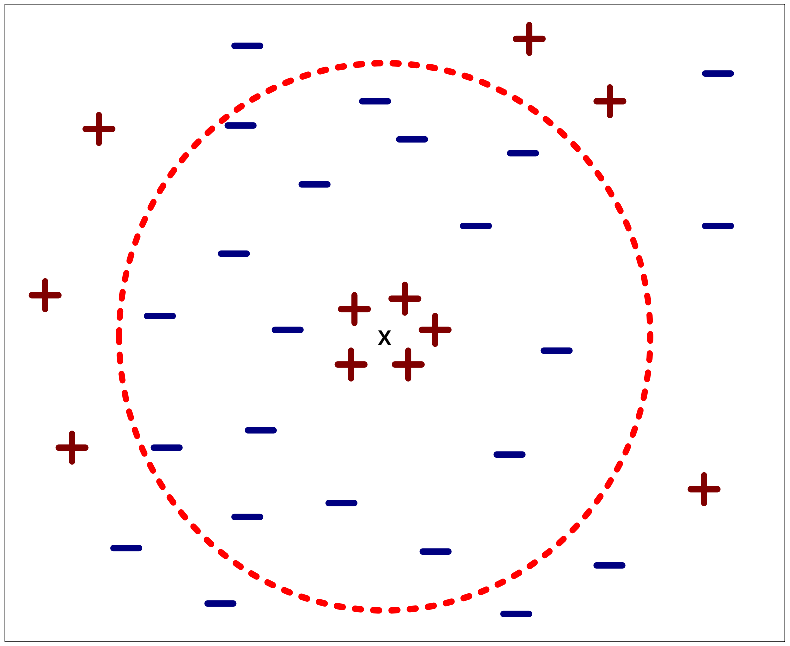
- **Calculate distance and determine the class from nearest neighbor list:**
  - **Take the majority vote of class labels among the k-nearest neighbors**

Nearest Neighbor Classification

- **Breaking Class Ties:**
  - Take mean (average) distance of tie classes and select the class with least mean distance.
  - Randomly select one class
  - Reduce the value of k

Nearest Neighbor Classification…

- **Choosing the value of k:**
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes



Nearest Neighbor Classification…

Scaling issues 01:

- If you change unit of one dimension (attribute) e.g., from Km to meters, the nearest neighbors will be changed.
- Solution: **Normalize**
    - **Example:** For an unseen example point P(2,3), If you multiply dimension X1 with -2 the resulting nearest neighbor will change. The solution is to scale and normalize attributes between (0,1).

X_1	X_2	**Target Class**
-4	3	**A**
6	3	**B**

Nearest Neighbor Classification…

Scaling issues 02:

- o **Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes**
- o Example:
  - height of a person may vary from 1.5m to 1.8m
  - weight of a person may vary from 90lb to 300lb
  - income of a person may vary from $10K to $1M

◆ Solution: **Normalize the vectors to unit or equal length i.e., for any attribute x_i :**

$$x_i = (x_i - \mu_i)/\sigma_i$$

Nearest Neighbor Classification…

- Problem with Euclidean measure:
  - High dimensional data
    - KNN works well in low dimensional spaces but for higher dimensions it suffers from curse of dimensionality.
    - Higher dimensions increase the distances.
    - Higher dimensions **make data points and outliers close to each other**. Thus decrease in prediction accuracy.

Nearest neighbor Classification (Conclusion)

- k-NN classifiers are **lazy** learners
  - It does not build models explicitly
  - Unlike **eager** learners such as decision tree induction and Neural Networks etc.
  - Classifying unknown records are relatively expensive