ML Methods, Variance vs Bias, Assessment

Aik Choon Tan, Ph.D.
Vice-Chair and Senior Member
Department of Biostatistics and Bioinformatics
aikchoon.tan@moffitt.org
5/16/2022

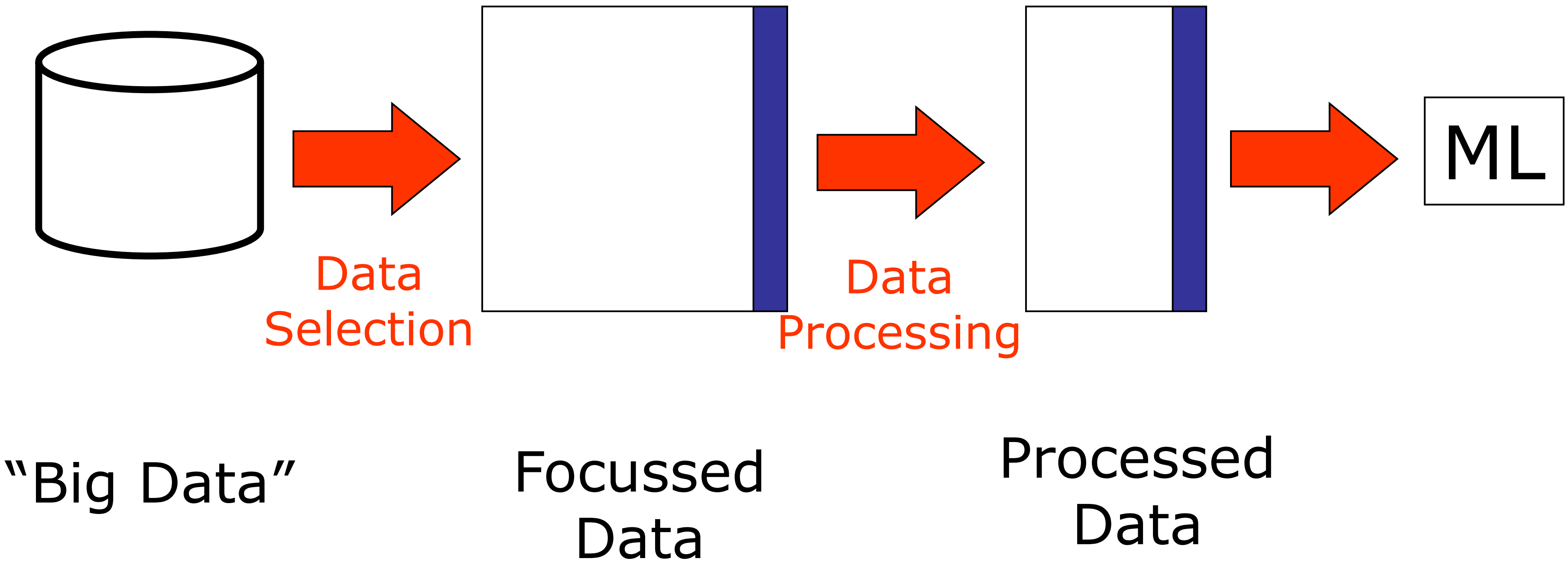
Outline

- Introduction
- Data, features, classifiers, Inductive learning
- (*Selected*) Machine Learning Approaches
  - Decision trees
  - Naïve Bayes
  - Linear Model (regression)
  - Support Vector Machines
- Variance vs Bias Trade-off
- Model evaluation

Steps in Class prediction problem

- Data Preparation
- Feature selection
  - Remove irrelevant features for constructing the classifier (but may have biological meaning)
  - Reduce search space in H, hence increase speed in generating classifier
  - Direct the learning algorithm to focus on "informative" features
  - Provide better understanding of the underlying process that generated the data
- Selecting a machine learning method
- Generating classifier from the training data
- Measuring the performance of the classifier
- Applying the classifier to unseen data (test)
- *Interpreting the classifier*

Data Preparation



Data: Samples and Features

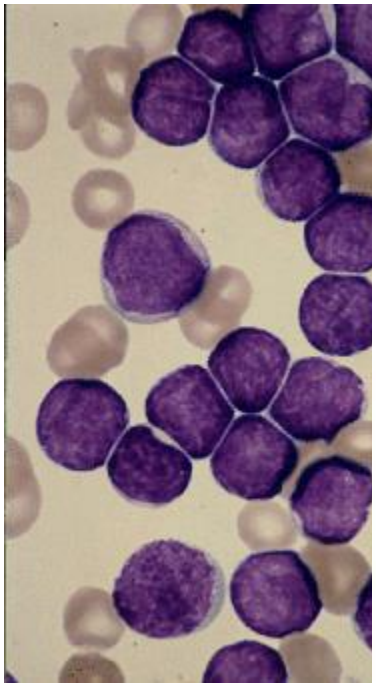
Samples ⟷

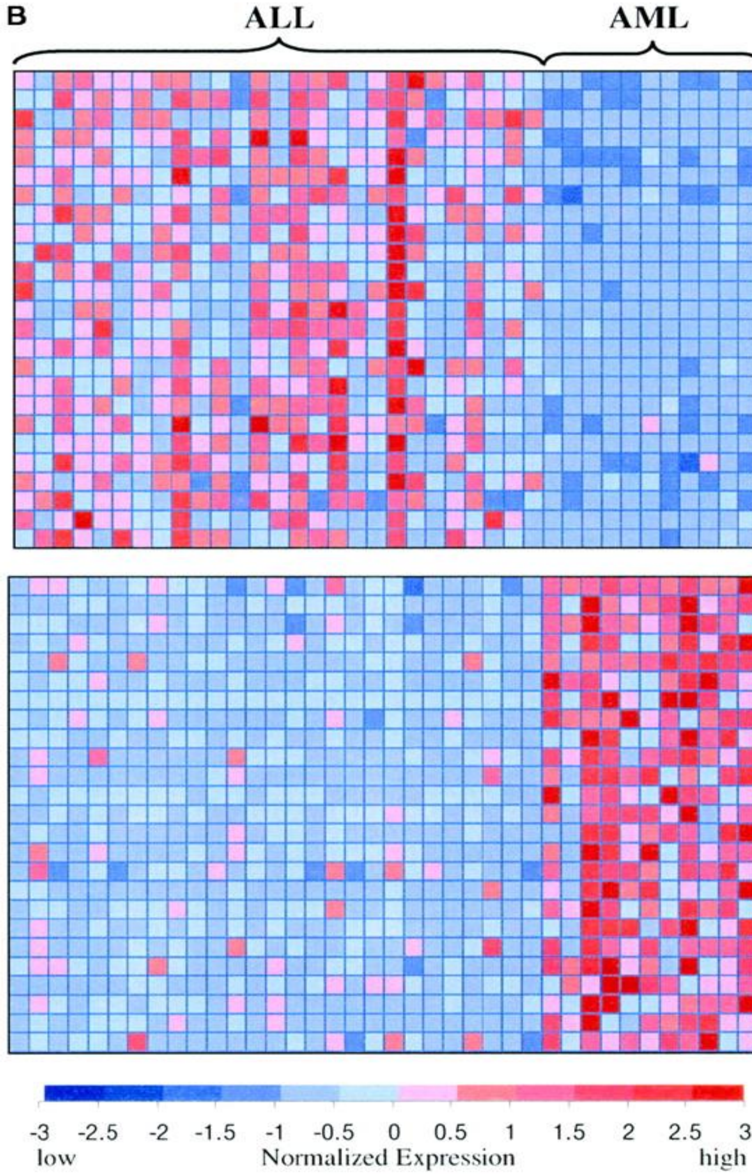
features ↕

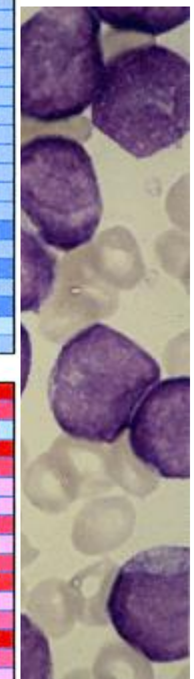
Features	Sample 1	Sample 2	...	Sample *m*
feature 1	Feature_ value	Feature_ value	...	Feature_ value
feature 2	Feature_ value	Feature_ value	...	Feature_ value
...	...	...	...	...
feature *n*	Feature_ value	Feature_ value	...	Feature_ value

Cancer Classification Problem

(Golub et al 1999)







ALL
acute lymphoblastic
(lymphoid precu

L
d leukemia
recursor)

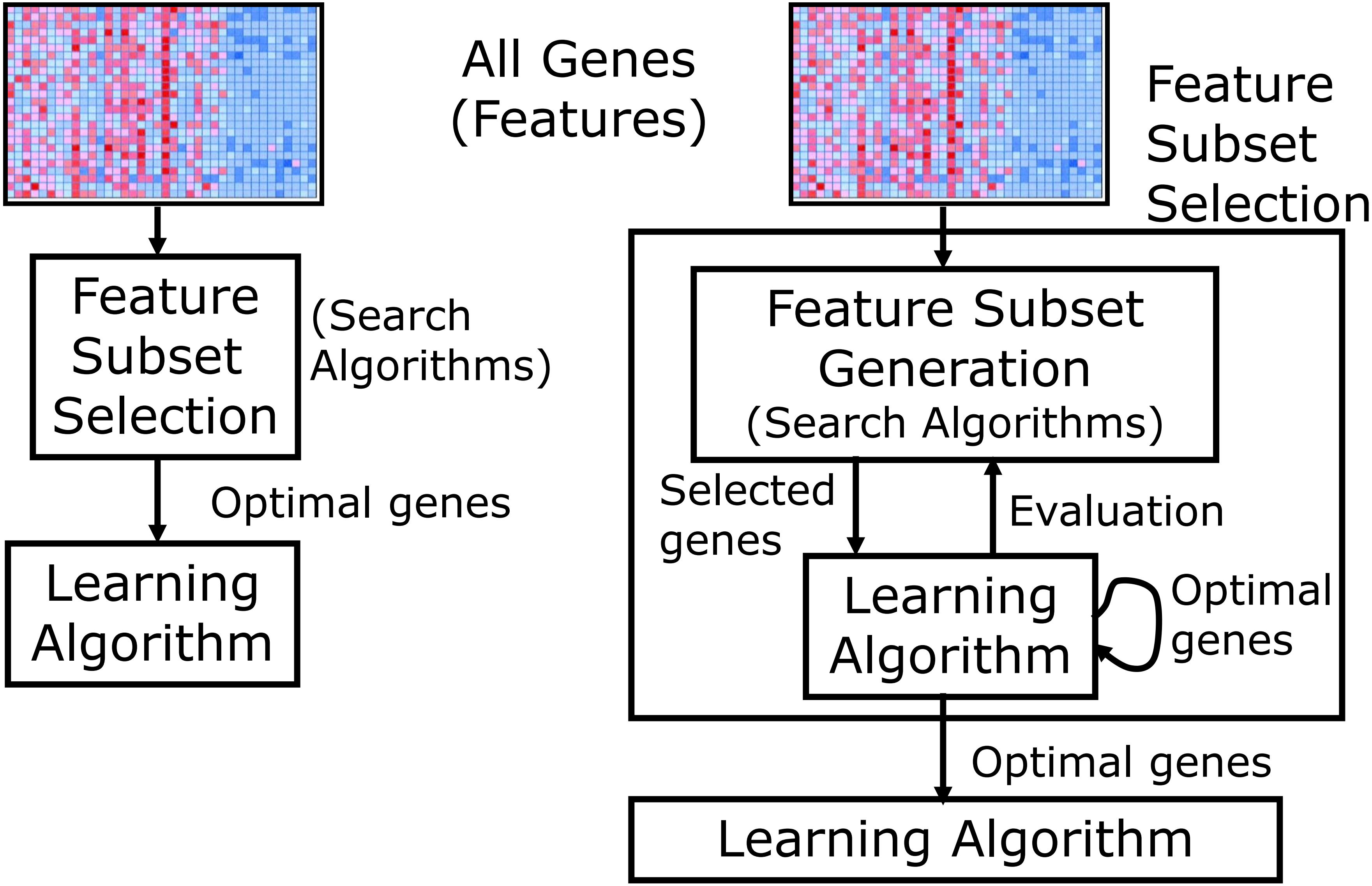
Gene Expression Profile

m samples ⟷

n genes ↕

Geneid	Condition 1	Condition 2	…	Condition *m*
Gene1	103.02	58.79	…	101.54
Gene2	40.55	1246.87	…	1432.12
…	…	…	…	…
Gene *n*	78.13	66.25	…	823.09

Gene (Feature Subset) Selection



(a) Filter approach (b) Wrapper approach

A (very) Brief Introduction to Machine Learning

To Learn

" … to acquire *knowledge* of (a subject) or skill in (an art, etc.) as a result of *study*, *experience*, or *teaching*… "
(OED)

What is Machine Learning?

" … a computer program that can learn from *experience* with respect to some class of *tasks* and *performance measure* … "

(Mitchell, 1997)

Key Steps of Learning

- Learning task
  - what is the learning task?

- Data and assumptions
  - what data is available for the learning task?
  - what can we assume about the problem?

- Representation
  - how should we represent the examples to be classified

- Method and estimation
  - what are the possible hypotheses?
  - how do we adjust our predictions based on the feedback?

- Evaluation
  - how well are we doing?

- Model selection
  - can we rethink the approach to do even better?

Learning Tasks

- Classification – Given positive and negative examples, find hypotheses that distinguish these examples. It can extends to multi-class classification.

- Clustering – Given a set of unlabelled examples, find clusters for these examples (unsupervised learning)

Learning Approaches

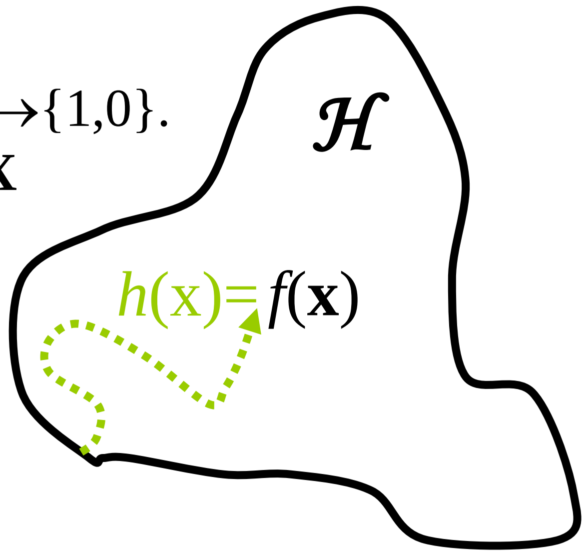
- Supervised approach – given *predefined* class of a set of positive and negative examples, construct the classifiers that distinguish between the classes $<\mathbf{x}, y>$

- Unsupervised approach – given the *unassigned* examples, group together the examples with similar properties $<\mathbf{x}>$

Concept Learning

Given a set of training examples $S = \{(x1,y1),\ldots,(xm,ym)\}$ where **x** is the instances usually in the form of tuple $<x1,\ldots,xn>$ and y is the class label, the function $y = f(x)$ is unknown and finding the f(x) represent the essence of concept learning.

For a binary problem $y \in \{1,0\}$, the unknown function $f\colon \mathbf{X} \rightarrow \{1,0\}$.
The learning task is to find a hypothesis $h(x) = f(x)$ for $\mathbf{x} \in \mathbf{X}$



Training examples $<\mathbf{x}, f(\mathbf{x})>$ where:
$f(\mathbf{x}) = 1$ are Positive examples,
$f(\mathbf{x}) = 0$ are Negative examples.

$\mathcal{H}$ is the set of all possible hypotheses, where $h\colon \mathbf{X} \rightarrow \{1,0\}$

A machine learning task:
Find hypothesis, $h(\mathbf{x}) = c(\mathbf{x})$; $\mathbf{x} \in \mathbf{X}$.
(in reality, usually ML task is to approximate $h(\mathbf{x}) \cong c(\mathbf{x})$)

Inductive Learning

- Given a set of observed examples
- Discover concepts from these examples
  - class formation/partition
  - formation of relations between objects
  - patterns

Learning paradigms

- Discriminative (model Pr(y|**x**))
  - only model decisions given the input examples; no model is constructed over the input examples

- Generative (model Pr(**x**|y))
  - directly build class-conditional densities over the multidimensional input examples
  - classify new examples based on the densities

Decision Trees

- Widely used - simple and practical
- Quinlan - ID3 (1986), C4.5 (1993) & See5/C5 (latest)
- Classification and Regression Tree (CART by Breiman et.al., 1984)
- Given a set of instances (with a set of properties/attributes), the learning system constructs a tree with internal *nodes* as an *attribute* and the *leaves* as the *classes*
- Supervised learning
- Symbolic learning, give interpretable results

Information Theory - Entropy

Entropy – a measurement commonly used in information theory to characterise the (im)purity of an arbitrary collection of examples

$$Entropy(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

where *S* is a collection of training examples with *c* classes and p_i is the proportion of examples *S* belonging to class *i*.

Example:
If S is a set of examples containing positive (+) and negative (-) examples (c $\in$ {+,-}), the entropy of S relative of this Boolean classification is:

$$Entropy(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

Entropy(*S*) =
- 0 if all members of *S* belong to the same class
- 1 if *S* contains an equal number of positive (+) and negative (-) examples

Note: Entropy ↓ Purity ↑*

ID3 (Induction of Decision Tree)

- Average entropy of attribute A

$$\hat{E}_A = \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

v = all the values of attribute A, S = training examples, S_v = training examples of attribute A with value v

$$E_A = \begin{cases} 0 \text{ if all members of } S \text{ belong to the same value v} \\ 1 \text{ if } S \text{ contains an equal number of value v examples} \end{cases}$$

Note: Entropy ↓ Purity ↑*

Splitting rule of ID3 (Quinlan, 1986)

- Information Gain

$$Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Note: Gain↑ Purity ↑*

Decision Tree Algorithm

Function *Decision_Tree_Learning* (*examples*, *attributes*, *target*)
Inputs: *examples* = set of training examples
attributes = set of attributes
target = class label

1. if *examples* is empty **then return** *target*
2. else if all *examples* have the same *target* **then return** *target*
3. else if *attributes* is empty then return most common value of *target* in *examples*
4. **else**
5. *Best* ←the attribute from *attributes* that best classifies *examples*
6. *Tree* ←a new decision tree with root attribute *Best*
7. **for** each value v_i of *Best* **do**
8. *examples*$_i$←{elements with *Best* = v_i}
9. *subtree*←*Decision_Tree_Learning* (*examples*, *attributes-best*, *target*)
10. add a branch to *Tree* with label v_i and subtree *subtree*
11. end
12. return *Tree*

Training Data

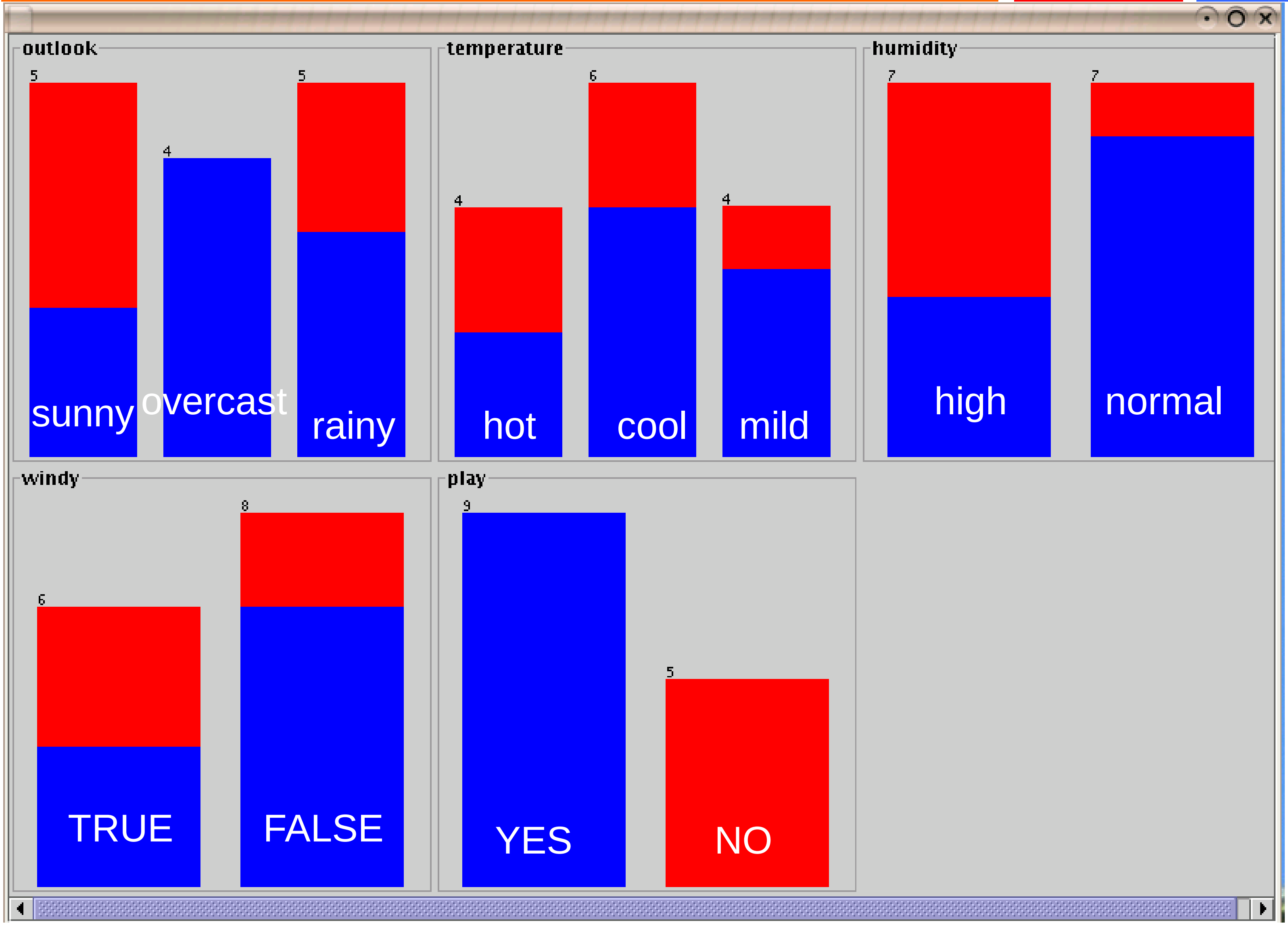
Decision attributes (dependent)

Independent condition attributes

Day	outlook	temperature	humidity	windy	play
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

Today	sunny	cool	high	TRUE	?

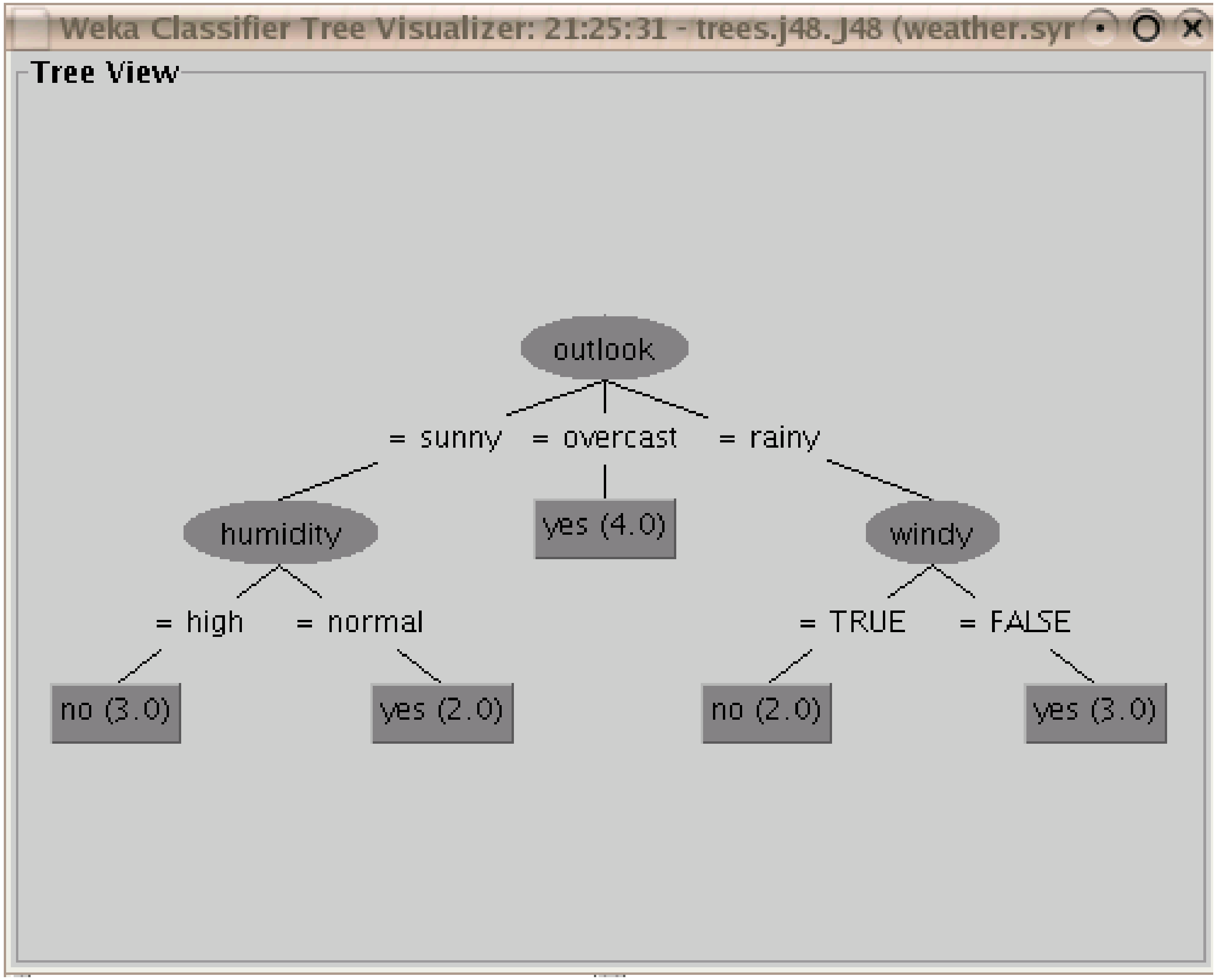
Entropy(S)



YES

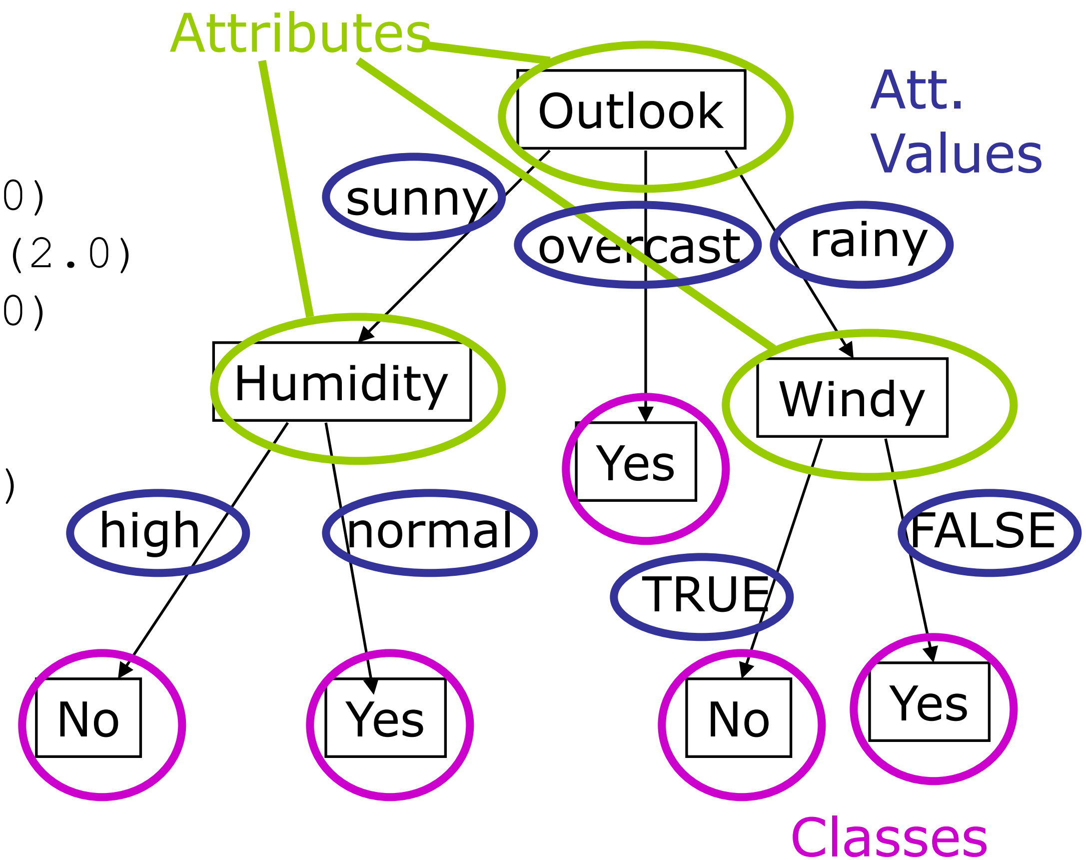
NO

Decision Tree

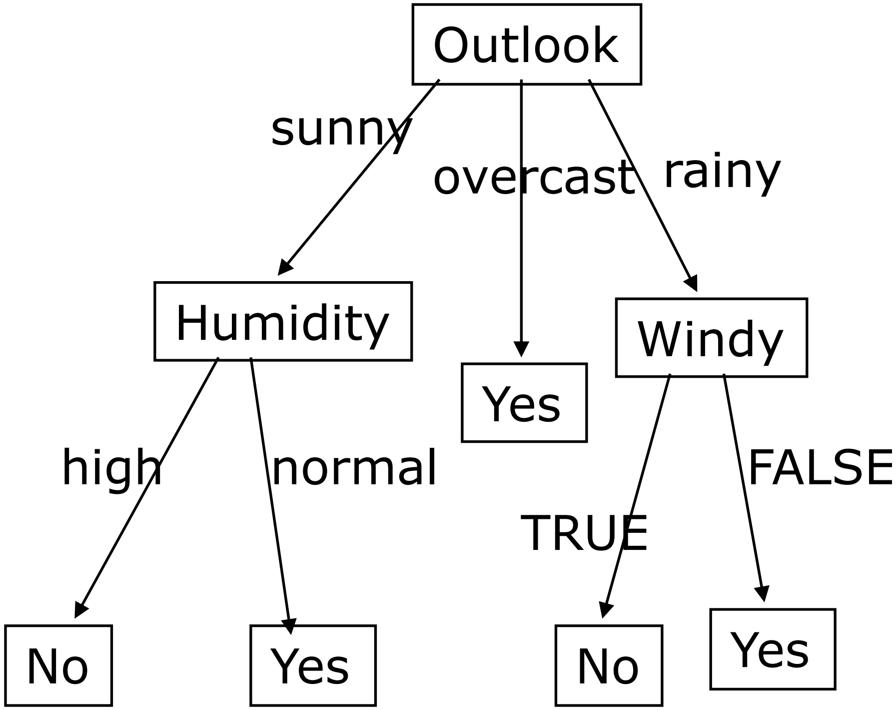


Decision Trees (Quinlan, 1993)

```
J48 pruned tree
------------------

outlook = sunny
|   humidity = high: no (3.0)
|   humidity = normal: yes (2.0)
outlook = overcast: yes (4.0)
outlook = rainy
|   windy = TRUE: no (2.0)
|   windy = FALSE: yes (3.0)

Number of Leaves  : 5

Size of the tree :  8


Time taken to build model: 0.05 seconds
Time taken to test model on training data: 0 seconds
```



Converting Trees to Rules



R1: IF Outlook = sunny $\wedge$ Humidity = high THEN play = No

R2: IF Outlook = sunny $\wedge$ Humidity = normal THEN play = Yes

R3: IF Outlook = overcast THEN play = Yes

R4: IF Outlook = rainy $\wedge$ Windy = TRUE THEN play = No

R5: IF Outlook = rainy $\wedge$ Windy = FALSE THEN play = Yes

Bayes Theorem

In machine learning we are interested to determine the best hypothesis *h(**x**)* from space *H*, based on the observed training data ***x**.*

Best hypothesis = *most probable* hypothesis, given the data ***x*** with any initial knowledge about the prior probabilities of the various hypothesis in *H*.

Bayes theorem provides a way to calculate

(i) the probability of a hypothesis based on its prior probability Pr(*h(**x**)*)

(ii) the probabilities of the observing various data given the hypothesis Pr(***x***|h)

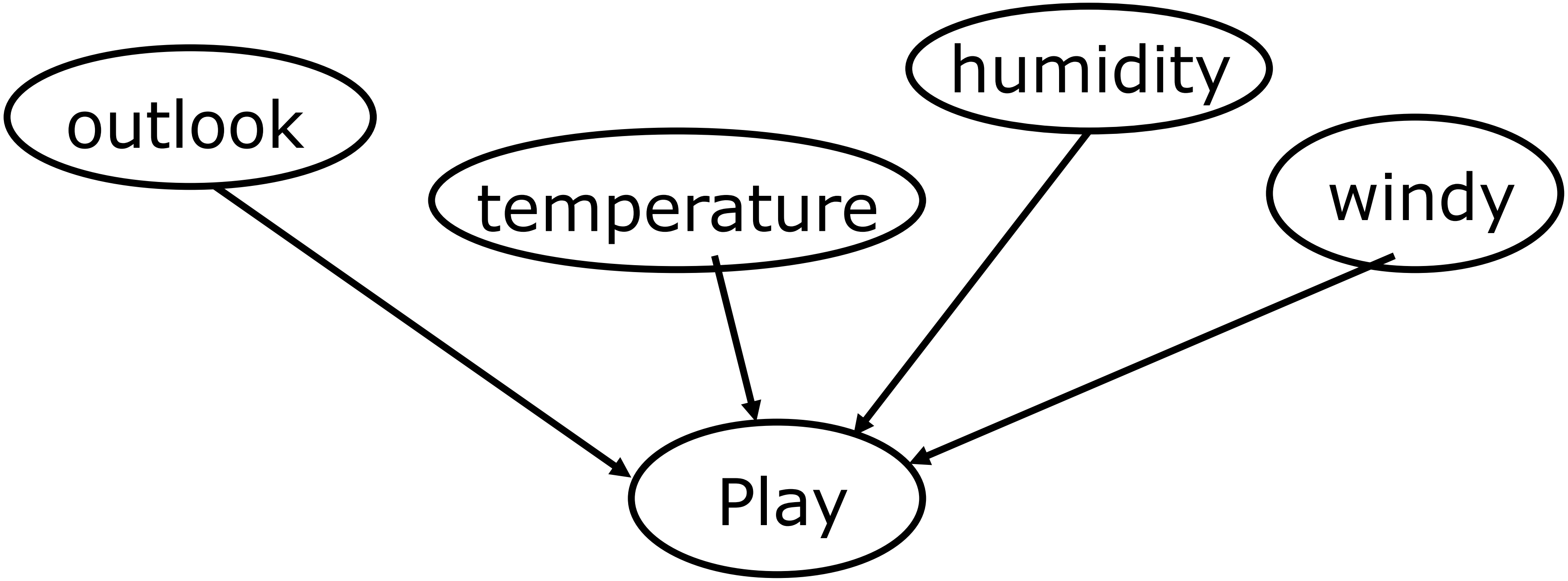
(iii) the probabilities of the observed data Pr(***x***)

We can calculate the posterior probability *h(**x**)* given the observed data ***x***, Pr(*h(**x**)*|***x***) using *Bayes theorem*.

$$\Pr(h(x) \mid x) = \frac{\Pr(x \mid h(x)) \Pr(h(x))}{\Pr(x)}$$

Naïve Bayes

(John & Langley, 1995)



To use all attributes and allow them to make contributions to the decision that are *equally important* and *independent* of one another, given the class.

Naïve Bayes Classifier

$$v_{NB} = \arg\max_{v_j \in V} \Pr(v_j) \prod_i \Pr(a_i \mid v_j)$$

Where v_{NB} denotes the target value output by the naïve Bayes classifer, $\Pr(v_j)$ is the probability of target value v_j occurs in the training data,$\Pr(a_i|v_j)$ is the conditionally independant probability of a_i given target value v_j.

Summary:

- The naïve Bayes learning method involves a learning step in which the various $\Pr(v_j)$ and $\Pr(a_i|v_j)$ terms are estimated, based on their frequencies over the training data.
- The set of these estimates corresponds to the learned hypothesis h(x).
- This hypothesis is then used to classify each new instance by applying the above rule.
- There is no explicit search through the space of possible hypothesis, instead the hypothesis is formed simply by counting the frequency of various data combinations within the training examples.

Naïve Bayes example

Today	sunny	cool	high	TRUE	?

Pr(Play = yes) = 9/14 = 0.64
Pr(Play = no) = 5/14 = 0.36

Pr(Outlook=sunny|Play = yes) = 2/9 = 0.22
Pr(Outlook=sunny|Play=no) = 3/5 = 0.60

Pr(Temperature = cool|Play = yes) =3/9 = 0.33
Pr(Temperature =cool|Play =no) =1/5 = 0.20

Pr(Humidity = high|Play = yes) = 3/9 =0.33
Pr(Humidity = high|Play = no) = 4/5 =0.80

Pr(Wind = TRUE|Play = yes) = 3/9 = 0.33
Pr(Wind = TRUE|Play = no) = 3/5 = 0.60

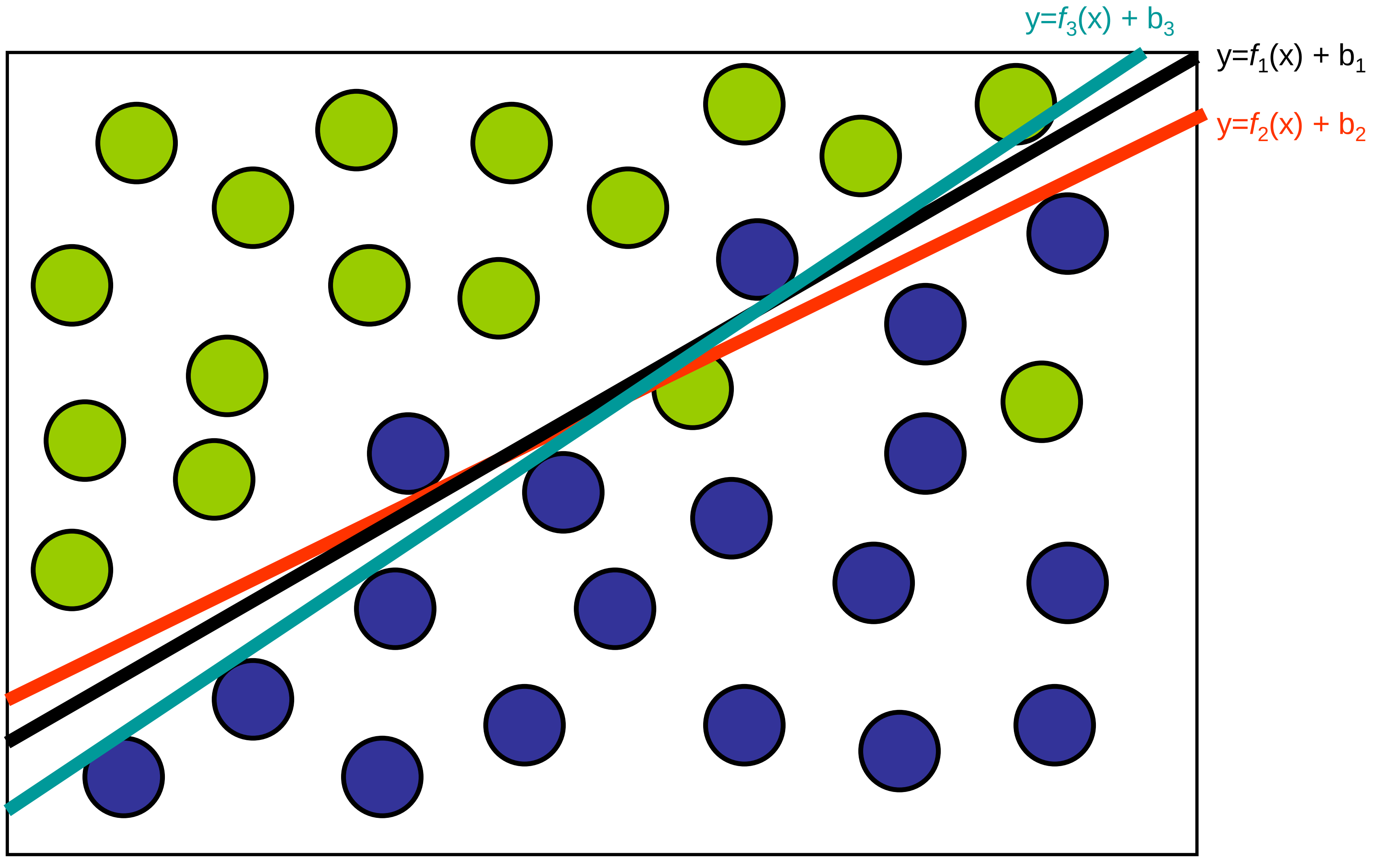
Pr(yes)Pr(sunny|yes)Pr(cool|yes) Pr(high|yes)Pr(TRUE|yes)= 0.64*0.22*0.33*0.33*0.33 = 0.0051

Pr(no)Pr(sunny|no)Pr(cool|no) Pr(high|no)Pr(TRUE|no)= 0.36*0.60*0.20*0.80*0.60 = 0.0207

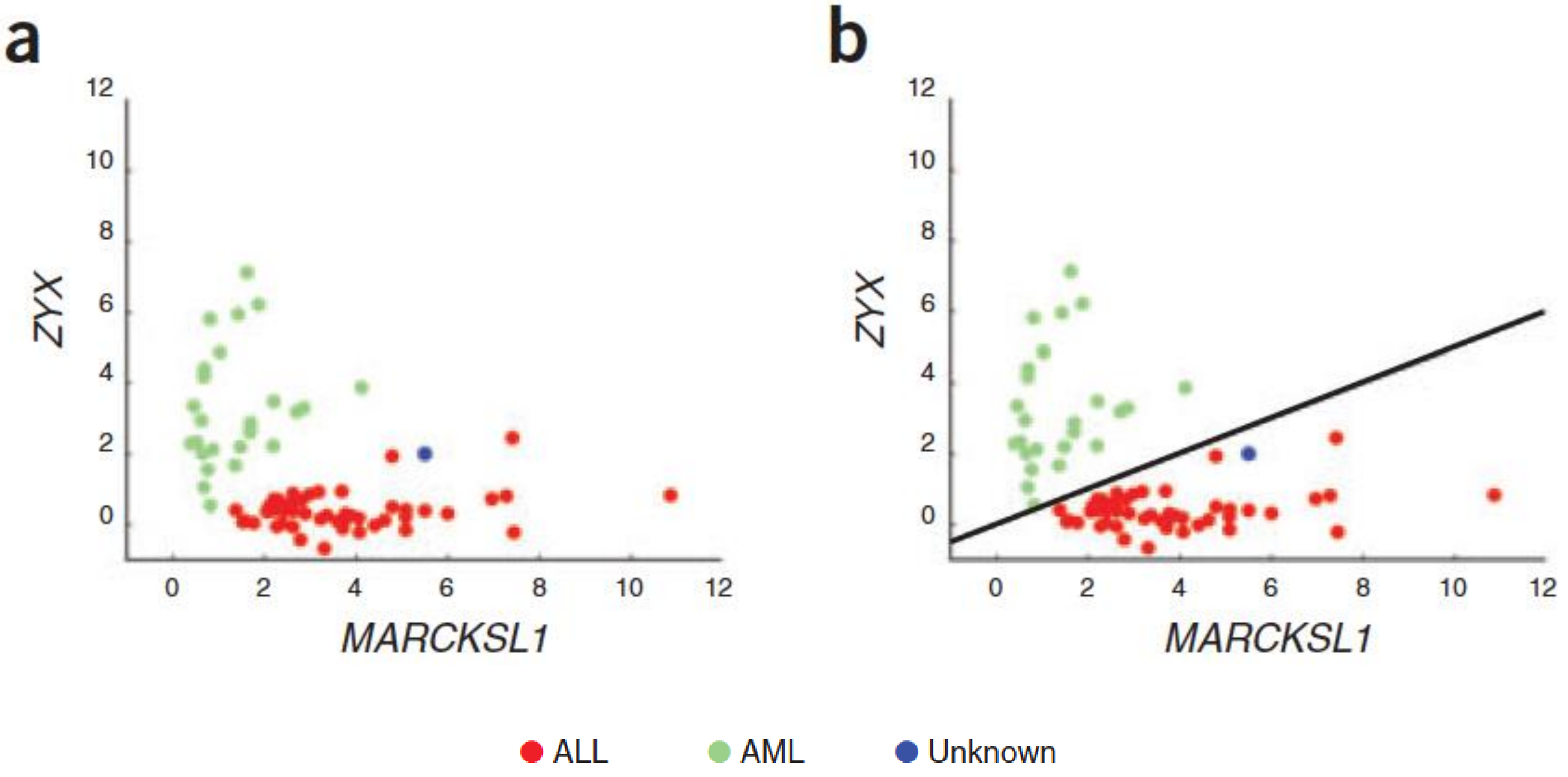
Play = NO

Probability = 0.0207/(0.0207+0.0051) =0.80 (80%)

Linear Model



Straight Line as Classifier in 2D Space



Support Vector Machines (SVM)

Key concepts:

Separating hyperplane – straight line in high-dimensional space

Maximum-margin hyperplane - the distance from the separating hyperplane to the nearest expression vector as the margin of the hyperplane Selecting this particular hyperplane maximizes the SVM's ability to predict the correct classification of previously unseen examples.

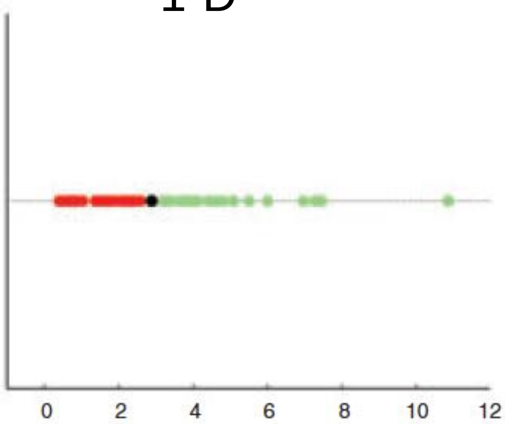
Soft margin - allows some data points ("soften") to push their way through the margin of the separating hyperplane without affecting the final result. User-specified parameter.

Kernel function - mathematical trick that projects data from a low-dimensional space to a space of higher dimension. The goal is to choose a good kernel function to separate data in high-dimensional space.

(Adapted from Noble 2006)

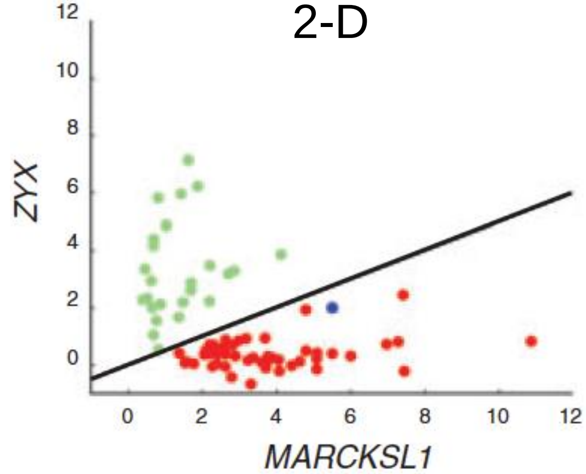
Separating Hyperplane

1-D



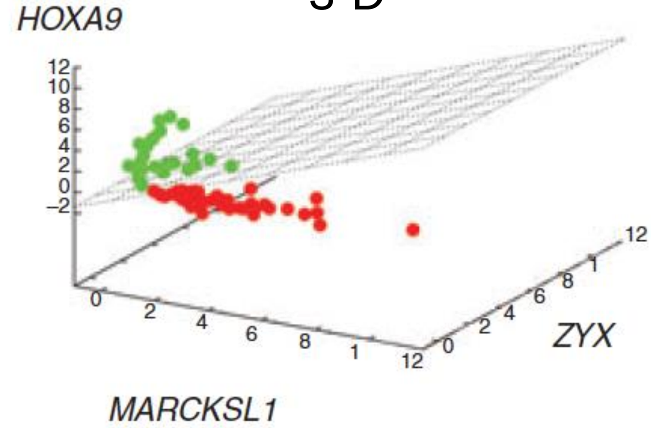
Separating hyperplane = dot

2-D



Separating hyperplane = line

3-D

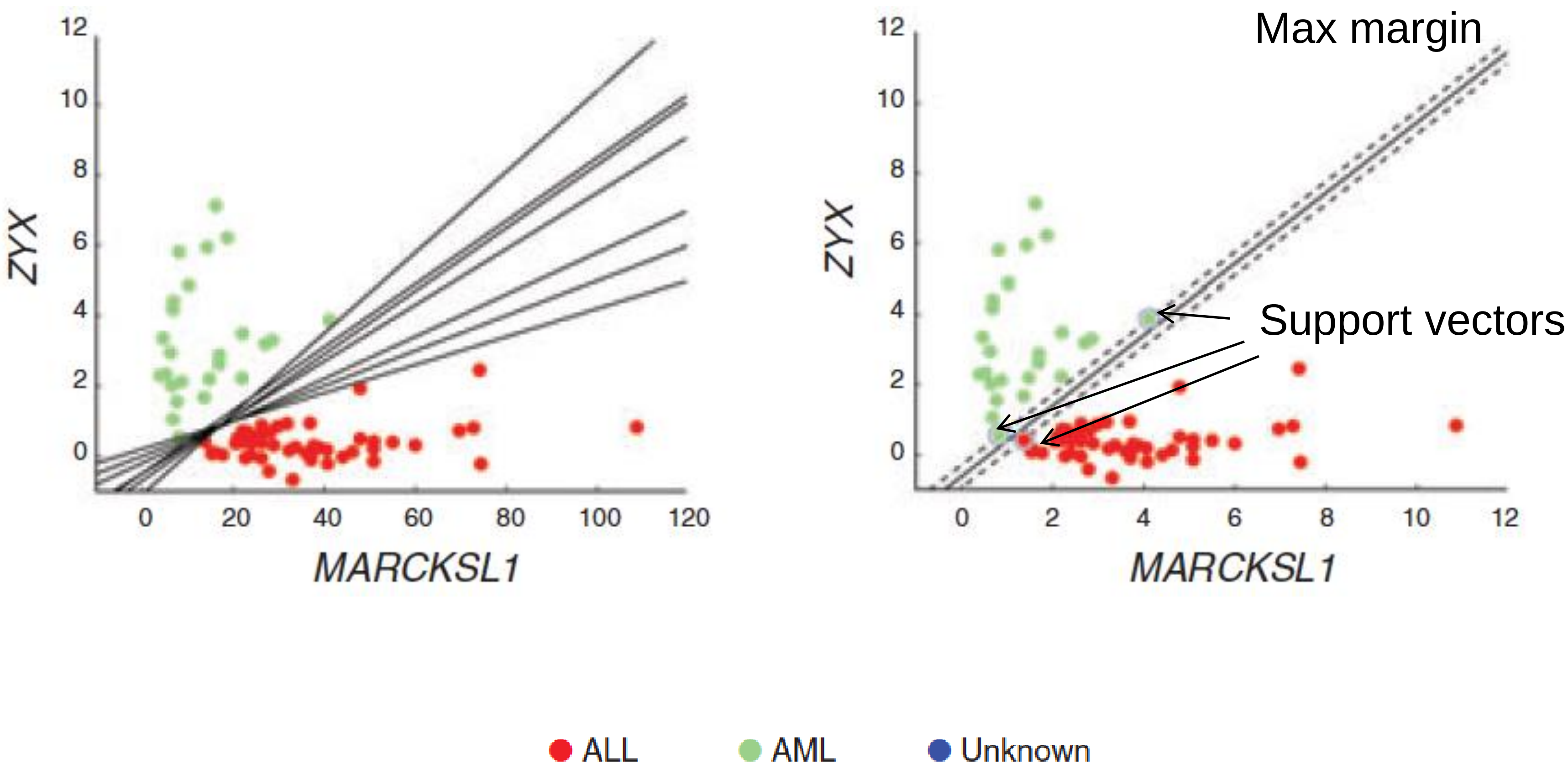


Separating hyperplane = hyperplane

● ALL ● AML ● Unknown

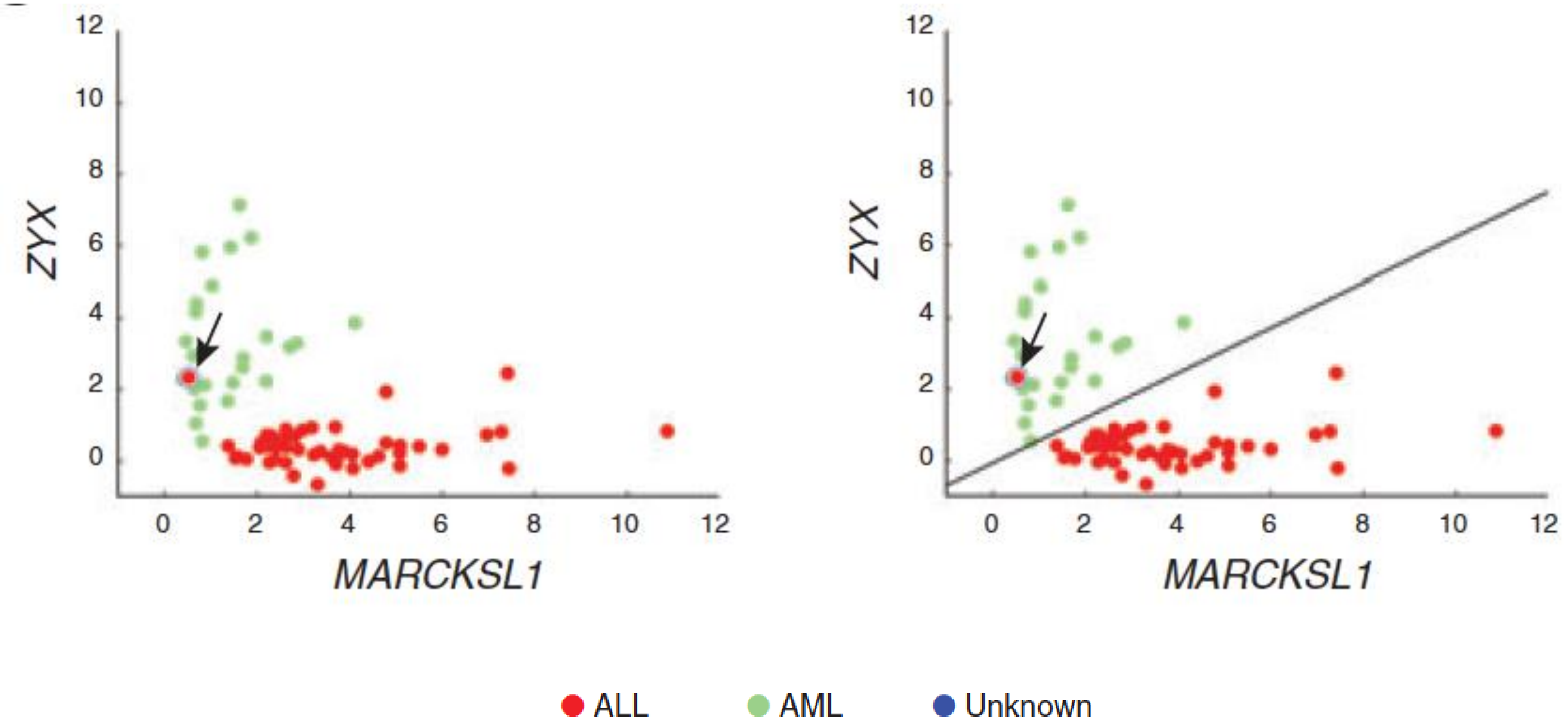
(Adapted from Noble 2006)

Maximum-margin Hyperplane



(Adapted from Noble 2006)

Soft Margin



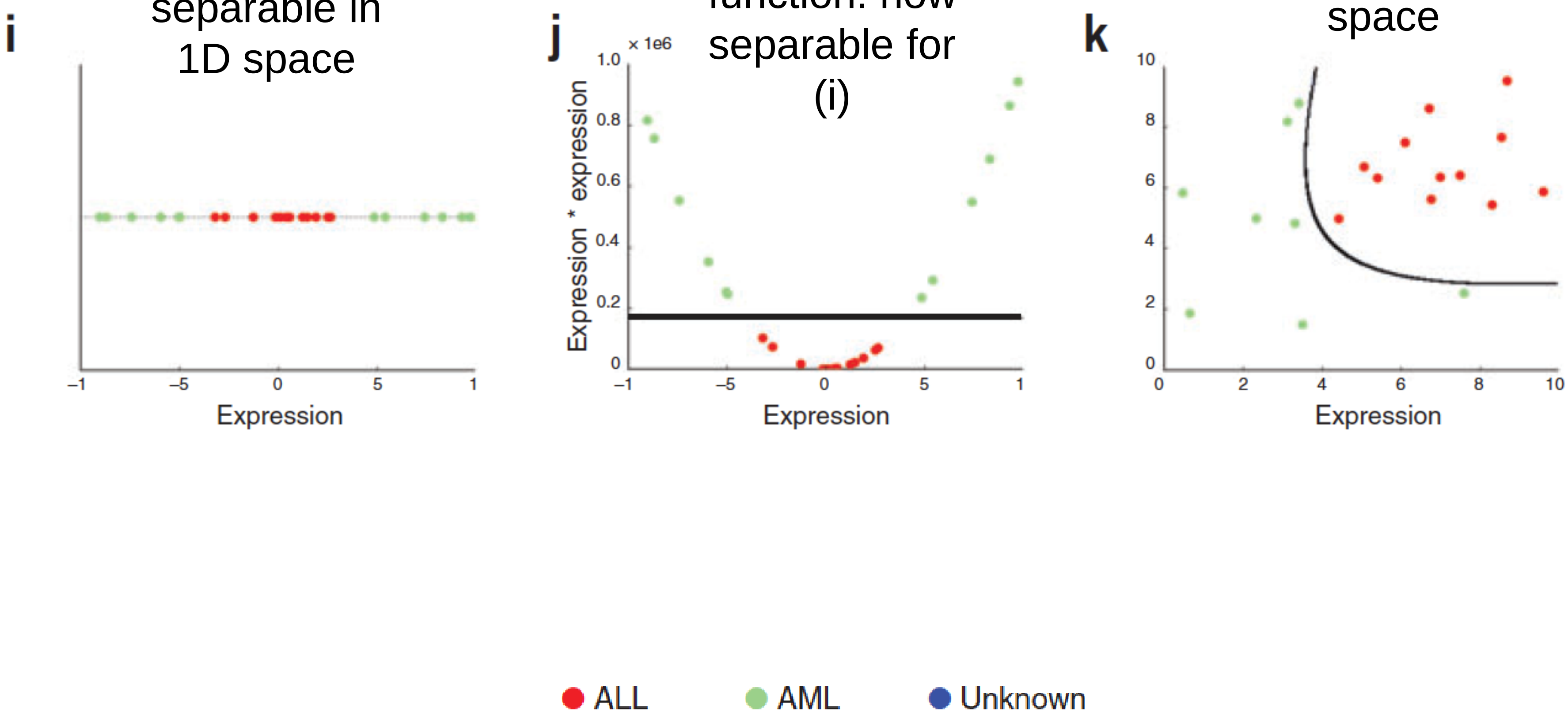
(Adapted from Noble 2006)

Kernel Function

Not separable in 1D space

Kernel function: now separable for (i)

Kernel function: 4-D space



(Adapted from Noble 2006)

Kernal: Linear SVM = Linear Regression

Predictive Accuracy = 50%

Support Vectors = 0

```
SMO

Classifier for classes: yes, no

BinarySMO

Machine linear: showing attribute weights, not support vectors.

   0.8440785904115866 * outlook=sunny
 + -0.9533207559861846 * outlook=overcast
 + 0.10924216557459787 * outlook=rainy
 + 0.5276359628579281 * temperature
 + 0.7712122046533554 * humidity
 + -0.8907578344254022 * windy
 - 0.8688305080362968

Number of kernel evaluations: 66



=== Stratified cross-validation ===

Correctly Classified Instances           7               50      %
Incorrectly Classified Instances         7               50      %
Kappa statistic                         -0.2564
Mean absolute error                      0.5
Root mean squared error                  0.7071
Relative absolute error                105      %
Root relative squared error            143.3236 %
Total Number of Instances               14


=== Confusion Matrix ===

 a b   <-- classified as
 7 2 | a = yes
 5 0 | b = no
```

Kernal: Polynomial (Quadratic Function)

Predictive Accuracy = 78.6%

Support Vectors = 10

```
SMO

Classifier for classes: yes, no

BinarySMO

   -1 * 0.8100635668551557 * K[X(1) * X]
 + -1 * 0.019817568367163058 * K[X(3) * X]
 + -1 * 0.8887836783080866 * K[X(4) * X]
 + -1 * 1.0 * K[X(6) * X]
 + -1 * 0.3534785049716326 * K[X(7) * X]
 + 1 * 0.3727234704263585 * K[X(9) * X]
 + 1 * 0.1796126505860263 * K[X(10) * X]
 + 1 * 1.0 * K[X(11) * X]
 + 1 * 0.7245265999700636 * K[X(12) * X]
 + 1 * 0.7952805975195893 * K[X(13) * X]
 - 0.6275818453891167

Number of support vectors: 10

Number of kernel evaluations: 104




=== Stratified cross-validation ===

Correctly Classified Instances          11               78.5714 %
Incorrectly Classified Instances         3               21.4286 %
Kappa statistic                          0.5116
Mean absolute error                      0.2143
Root mean squared error                  0.4629
Relative absolute error                 45      %
Root relative squared error             93.8273 %
Total Number of Instances               14


=== Confusion Matrix ===

 a b   <-- classified as
 8 1 | a = yes
 2 3 | b = no
```

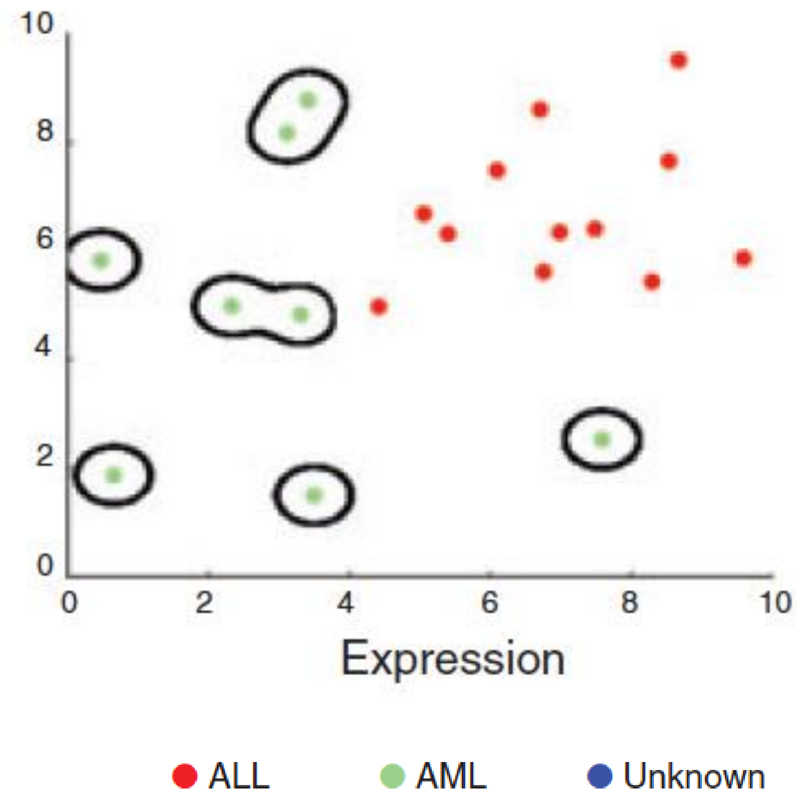
Kernal: Polynomial (Cubic Function)

Predictive Accuracy = 85.7%

Support Vectors = 12

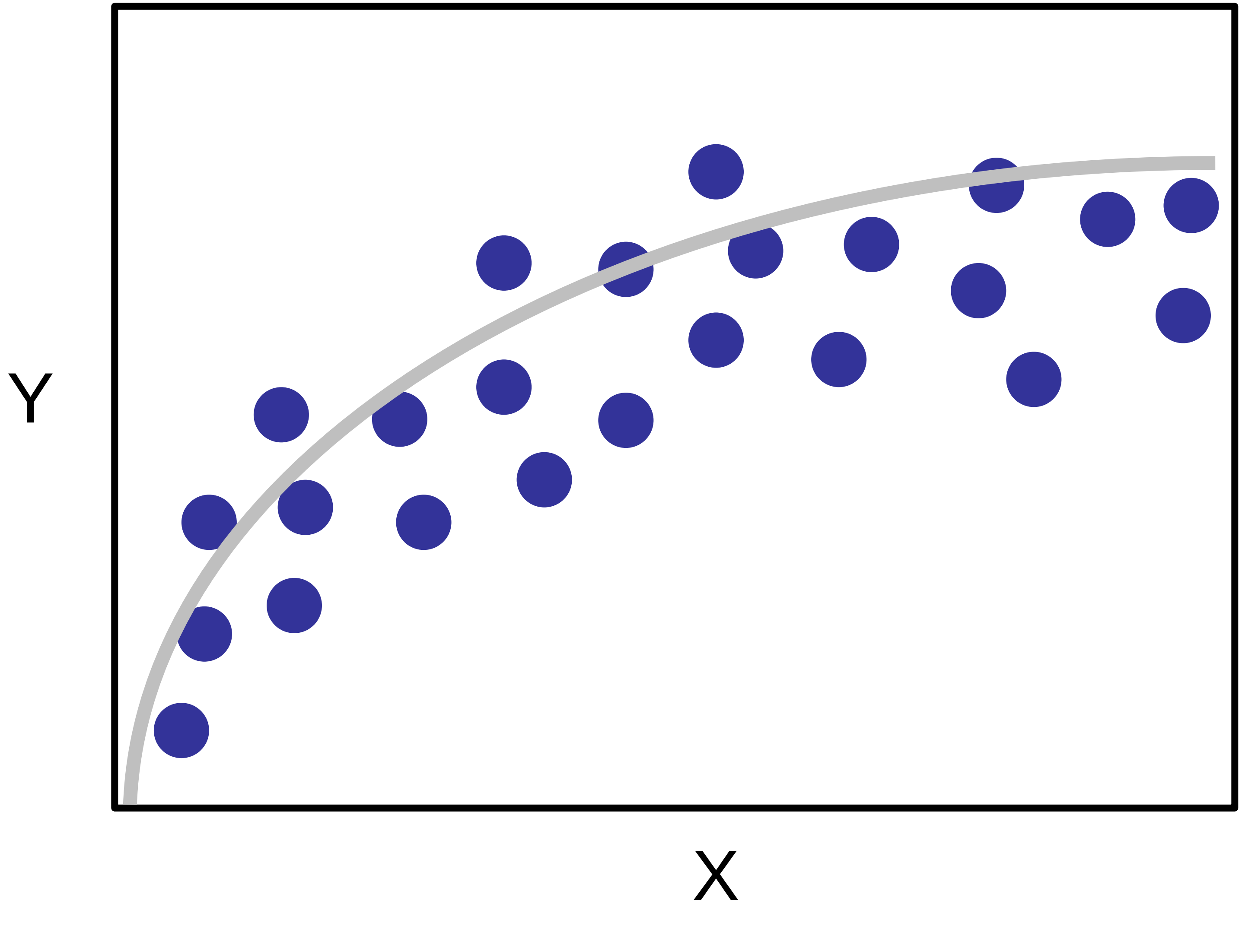
```
SMO

Classifier for classes: yes, no

BinarySMO

   -1 * 0.0585981464926855896 * K[X(1) * X]
 + -1 * 0.0321596375582114O6 * K[X(2) * X]
 + -1 * 0.36378917190737614 * K[X(3) * X]
 + -1 * 0.07019514690769074 * K[X(4) * X]
 + -1 * 0.07107098581642621 * K[X(5) * X]
 + -1 * 0.8766783011511141 * K[X(6) * X]
 + -1 * 0.06751016568226335 * K[X(7) * X]
 + 1 * 0.05271346042267785 * K[X(9) * X]
 + 1 * 0.3664976239753861 * K[X(10) * X]
 + 1 * 1.0 * K[X(11) * X]
 + 1 * 0.02756479285905889 * K[X(12) * X]
 + 1 * 0.0932256782586451 * K[X(13) * X]
 - 0.5679604722895839

Number of support vectors: 12

Number of kernel evaluations: 105




=== Stratified cross-validation ===

Correctly Classified Instances          12               85.7143 %
Incorrectly Classified Instances         2               14.2857 %
Kappa statistic                          0.6585
Mean absolute error                      0.1429
Root mean squared error                  0.378
Relative absolute error                 30      %
Root relative squared error             76.6097 %
Total Number of Instances               14


=== Confusion Matrix ===

 a b   <-- classified as
 9 0 | a = yes
 2 3 | b = no
```

Overfitting in SVM

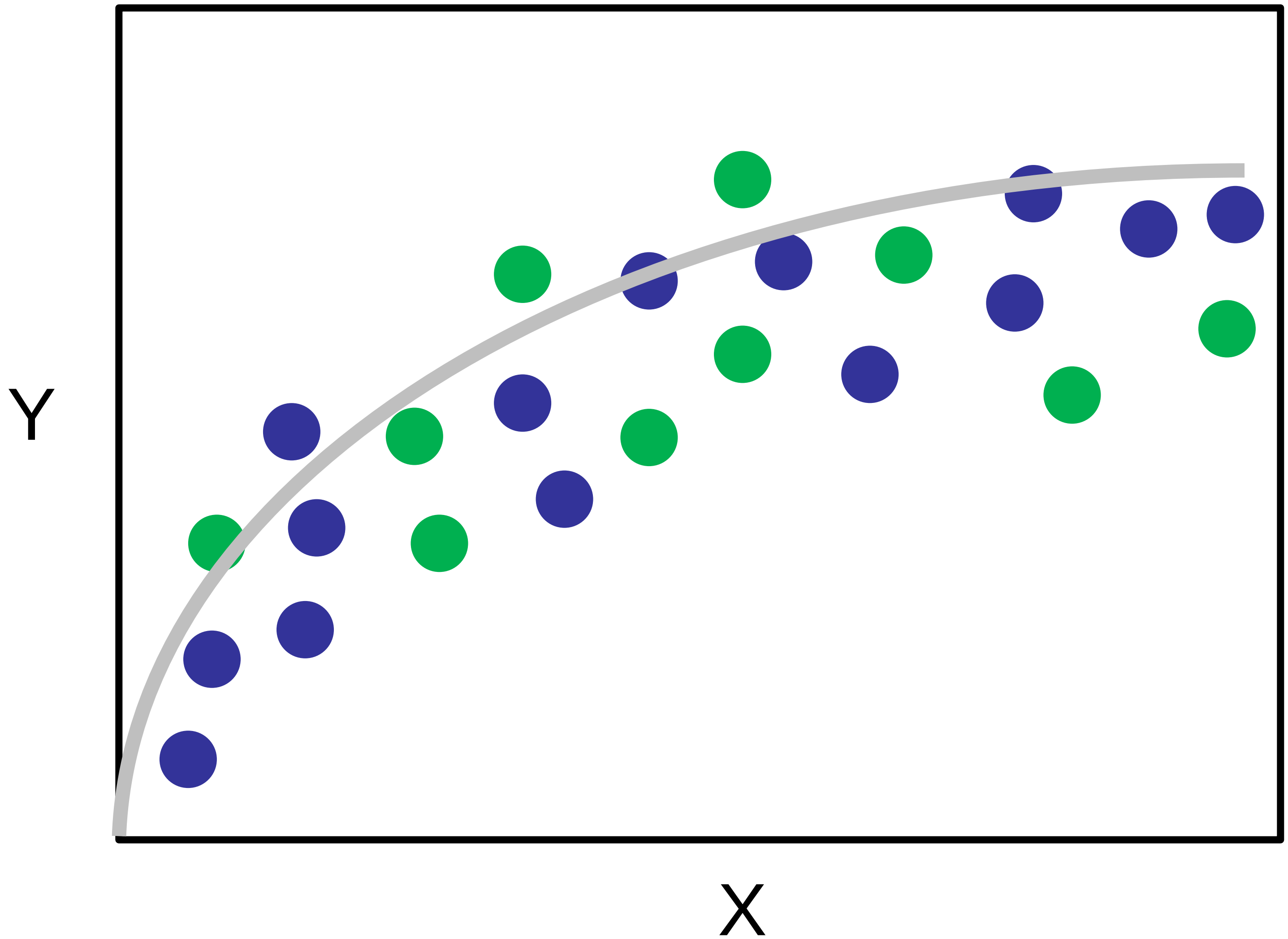


(Adapted from Noble 2006)

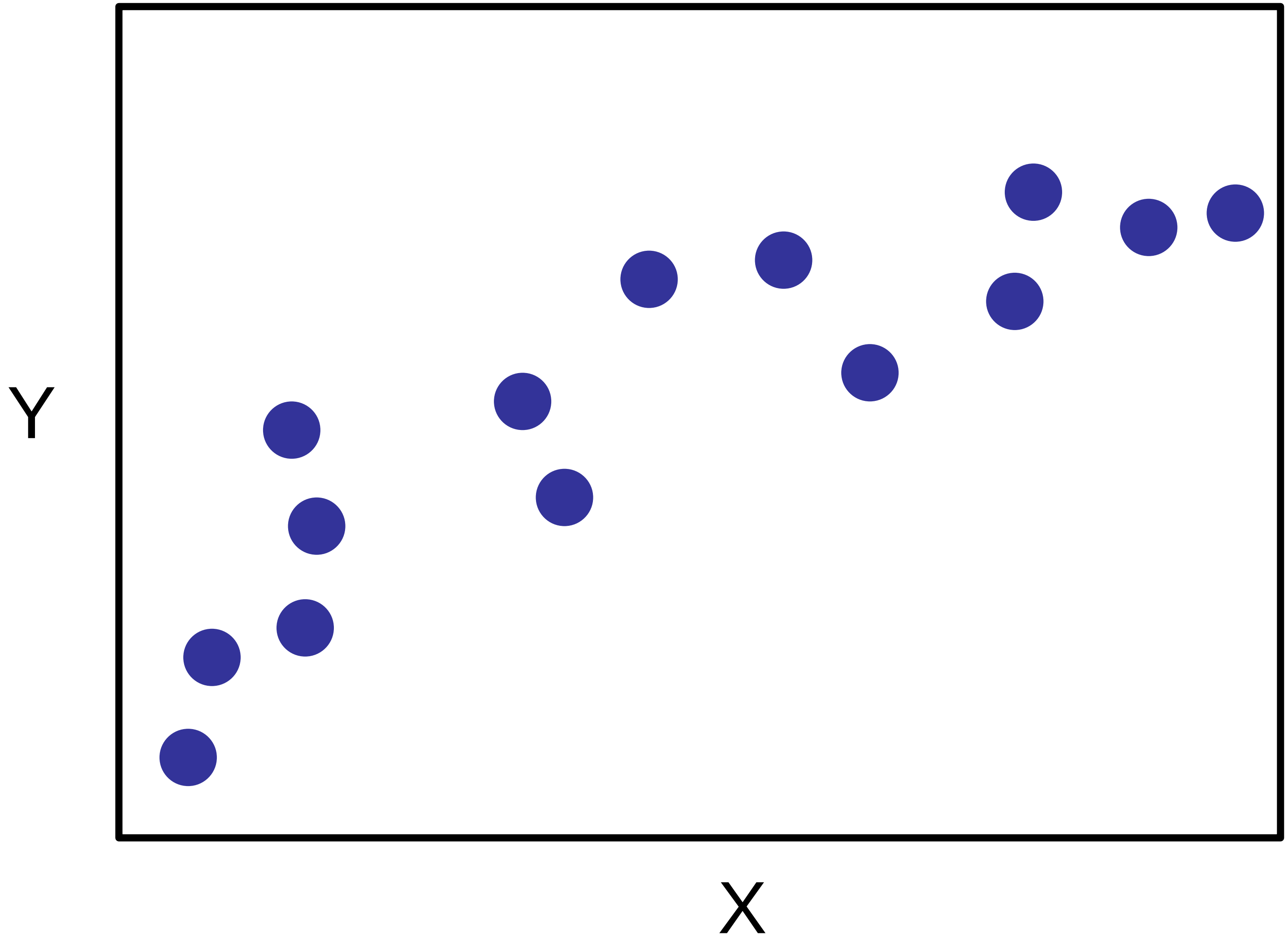
”True” Model



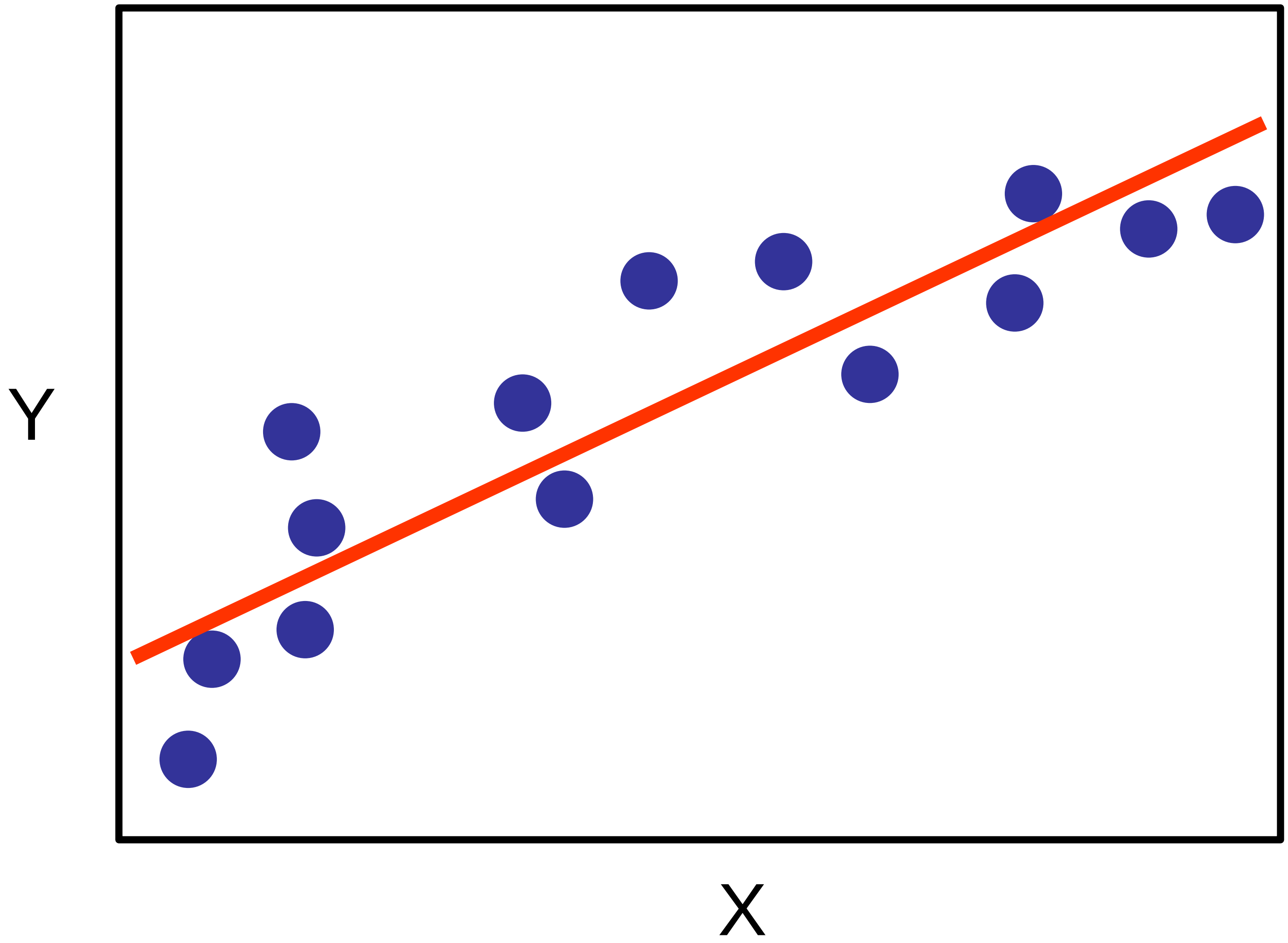
Training and Testing Data



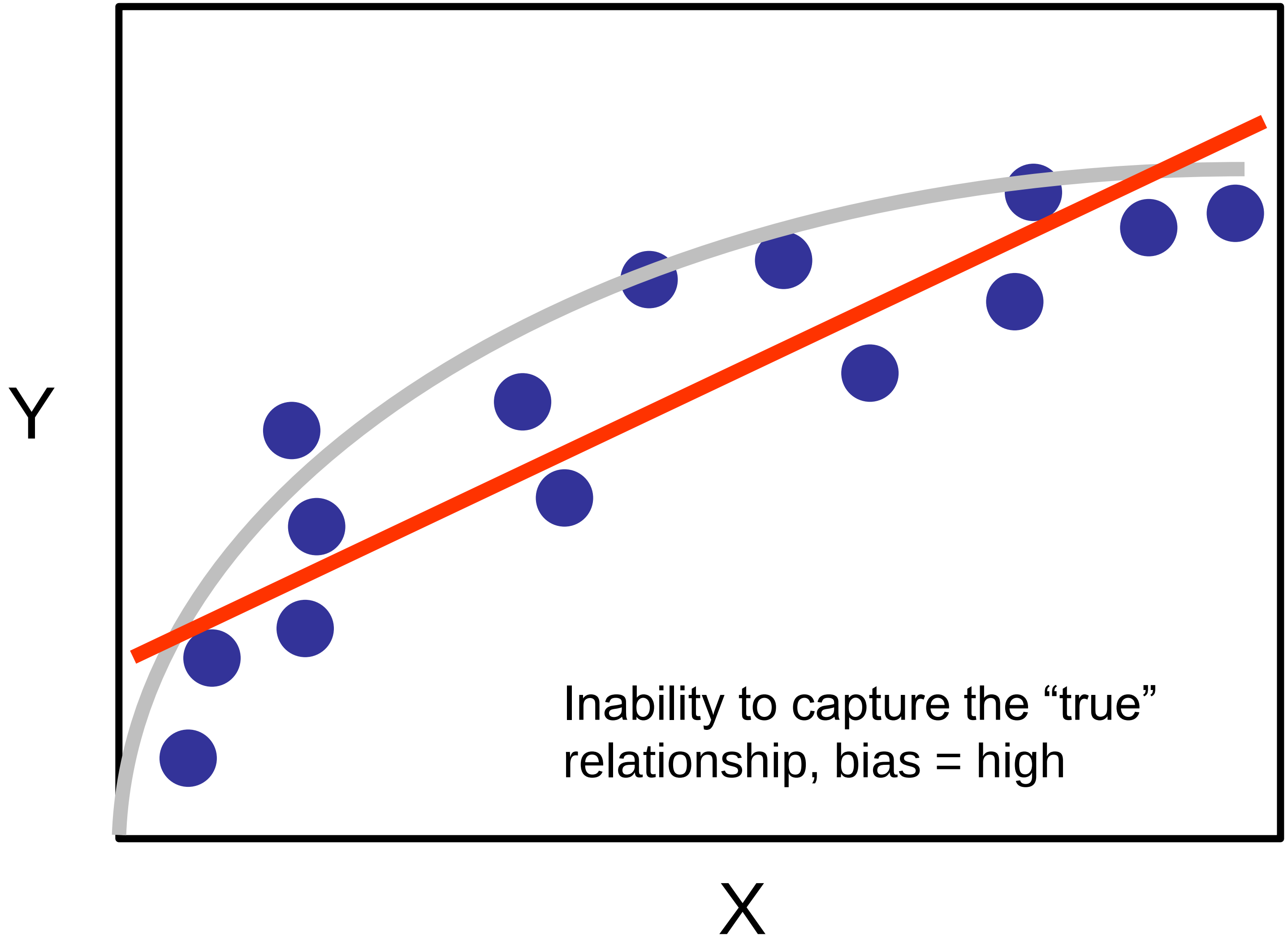
Training Data



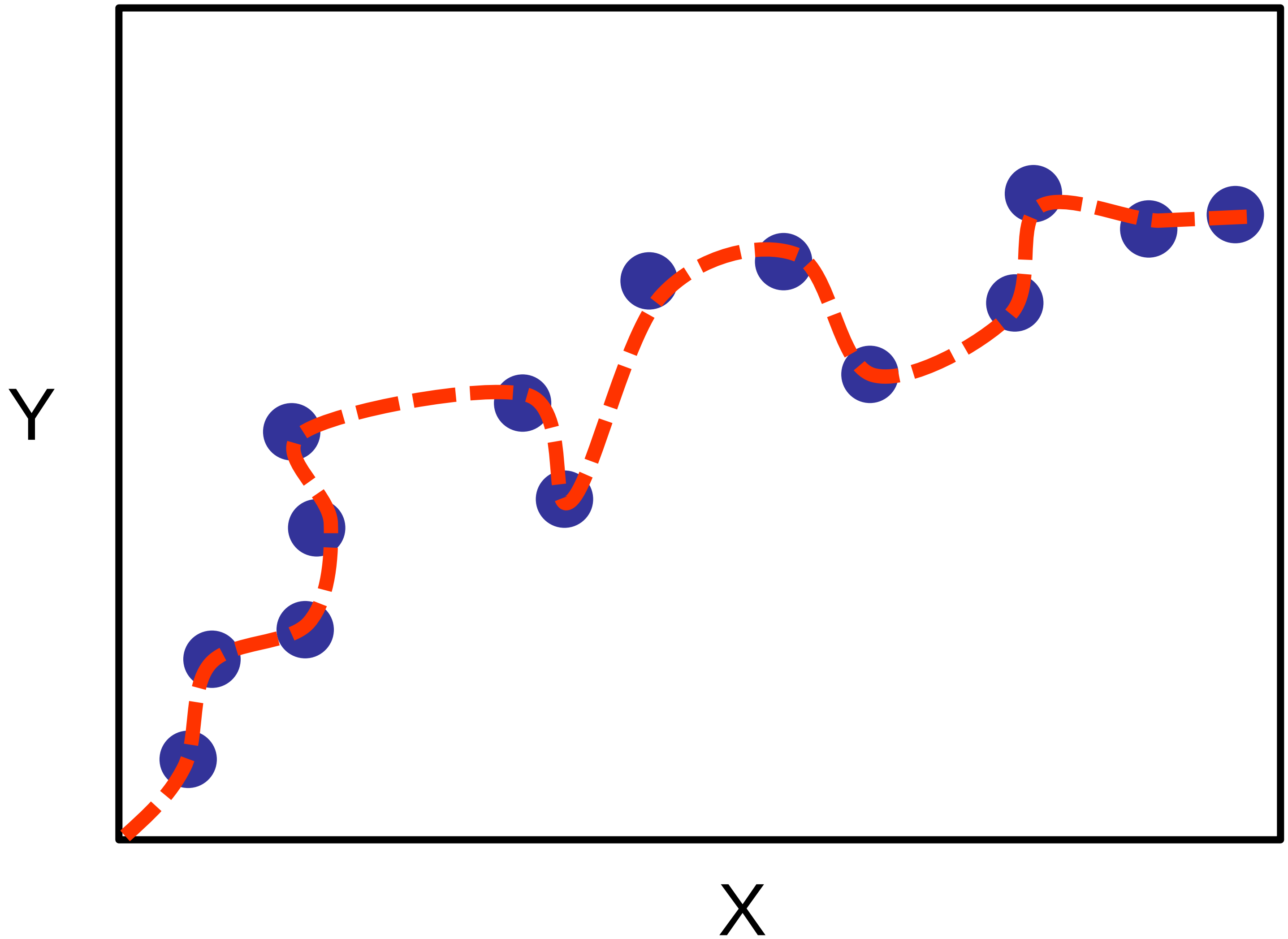
Model #1 (Straight Line)



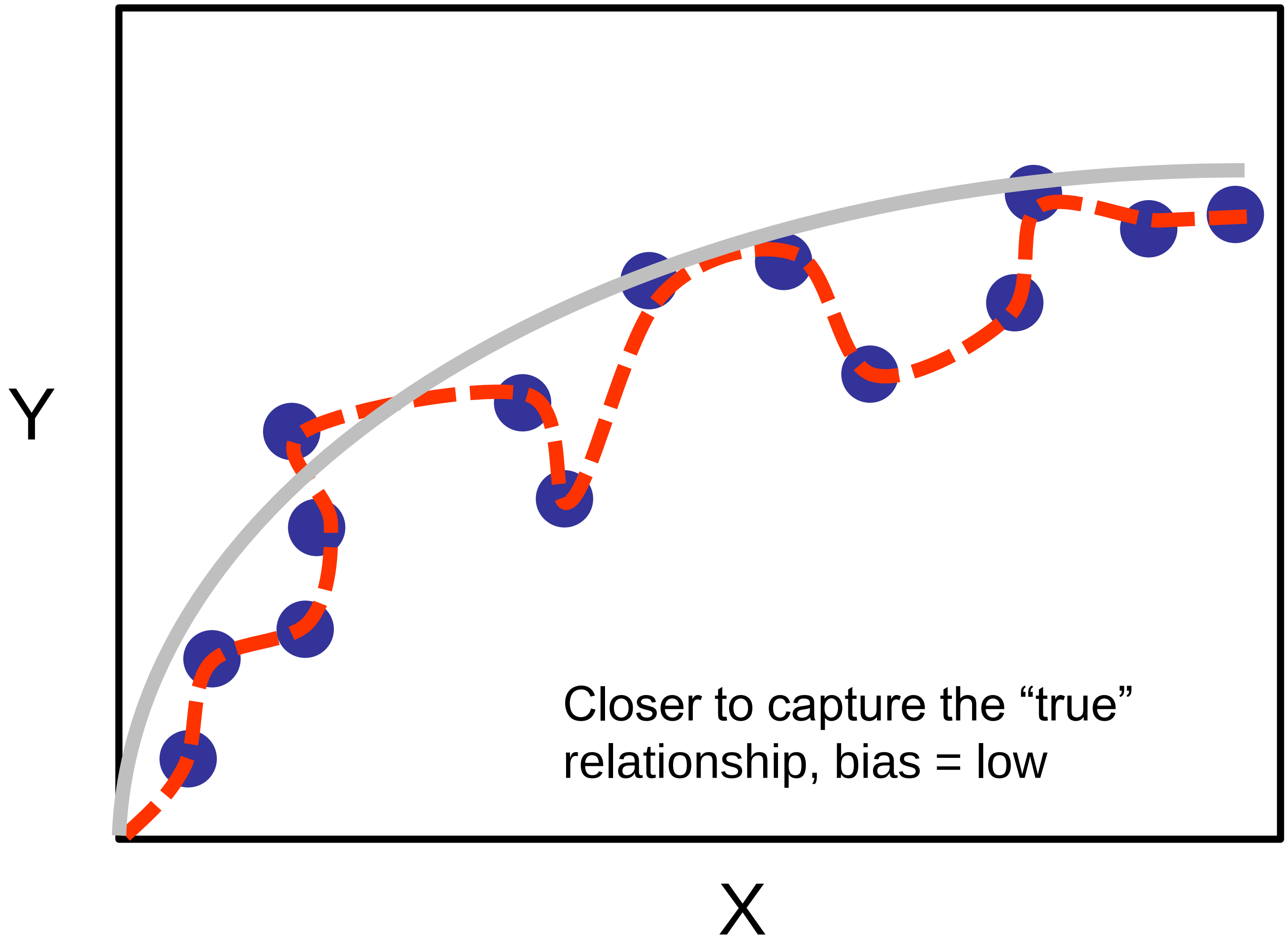
Model #1 (Straight Line)



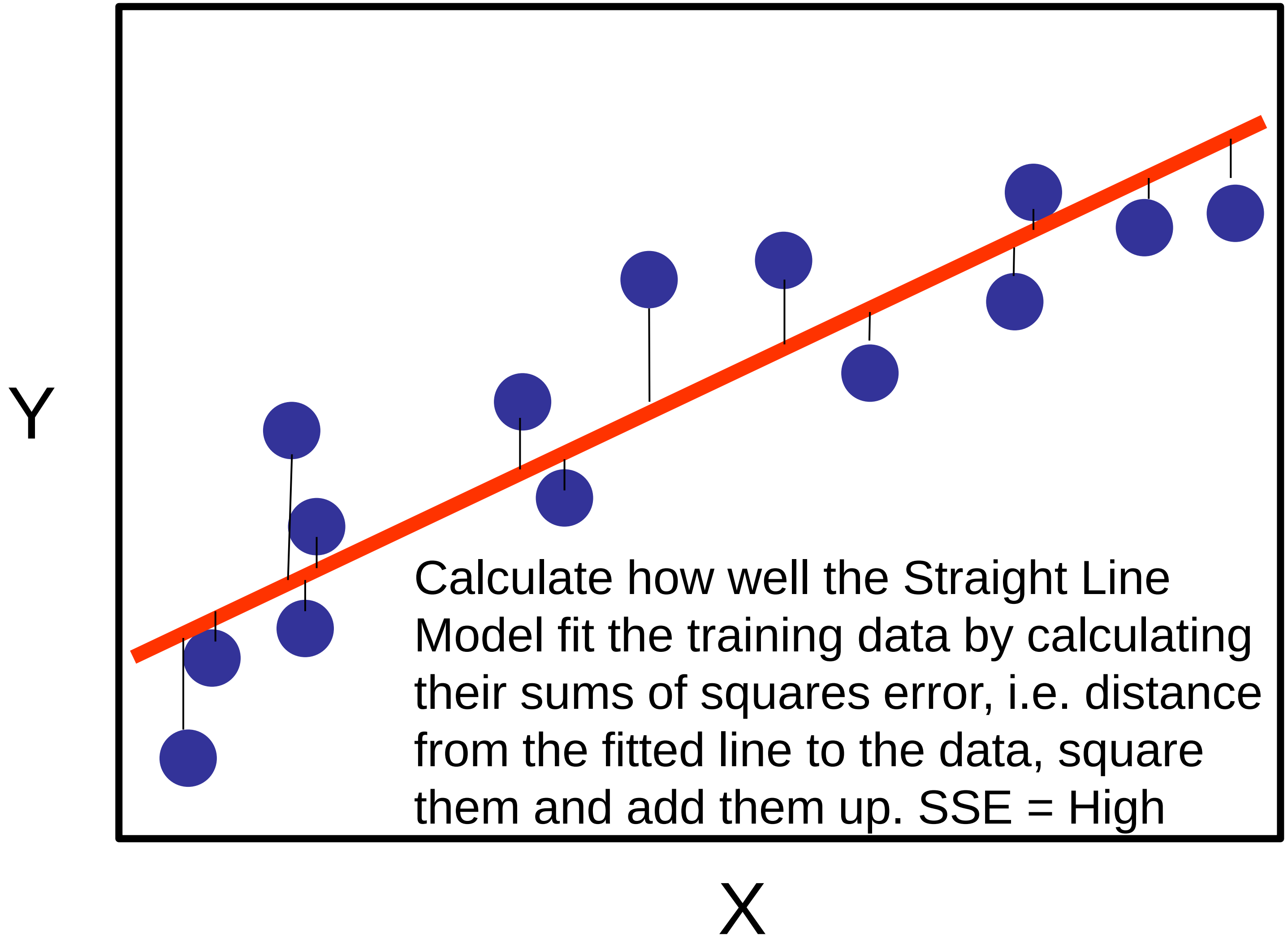
Model #2 (Zigzag Line)



Model #2 (Zigzag Line)

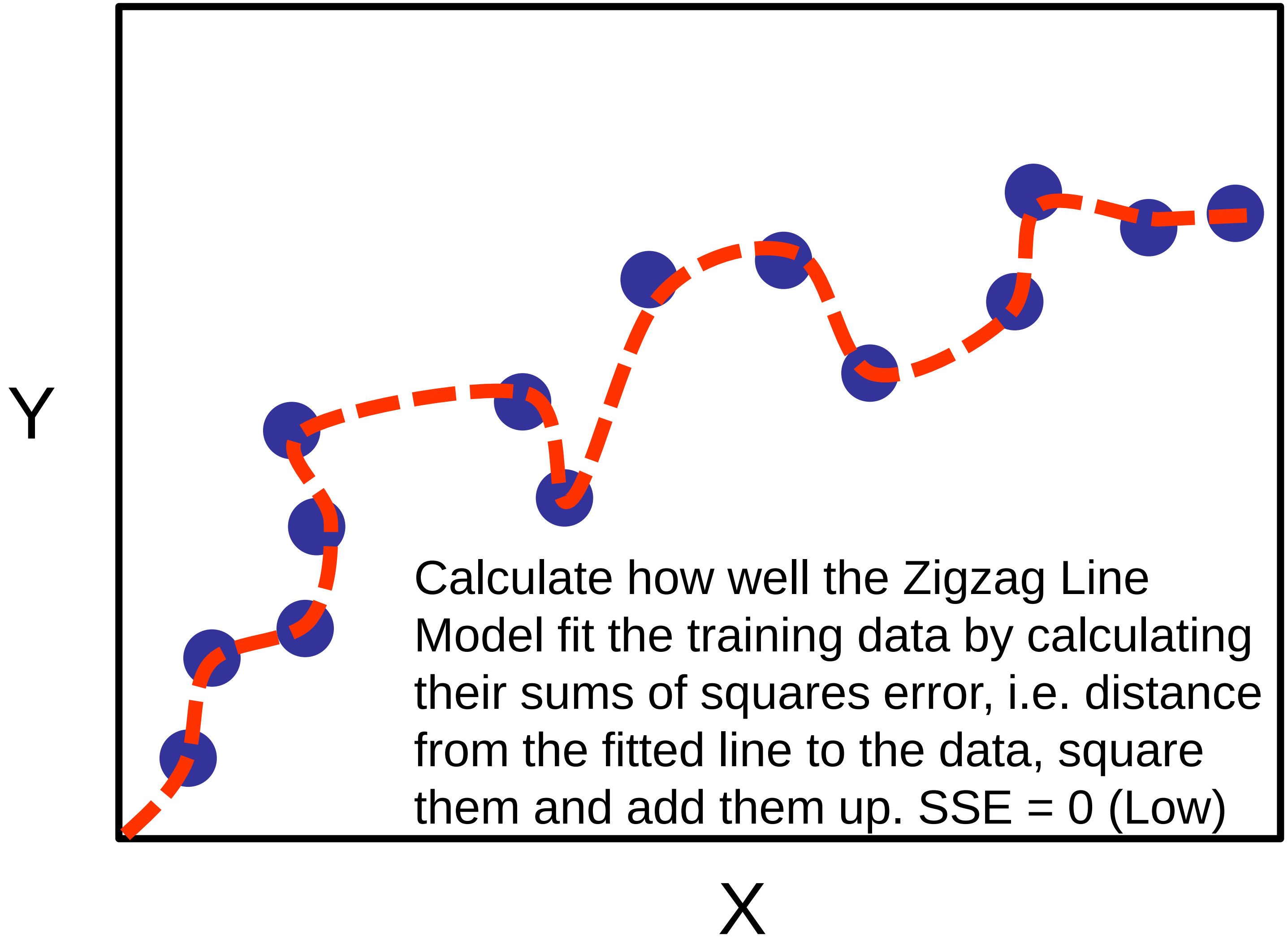


Model #1 (Straight Line)

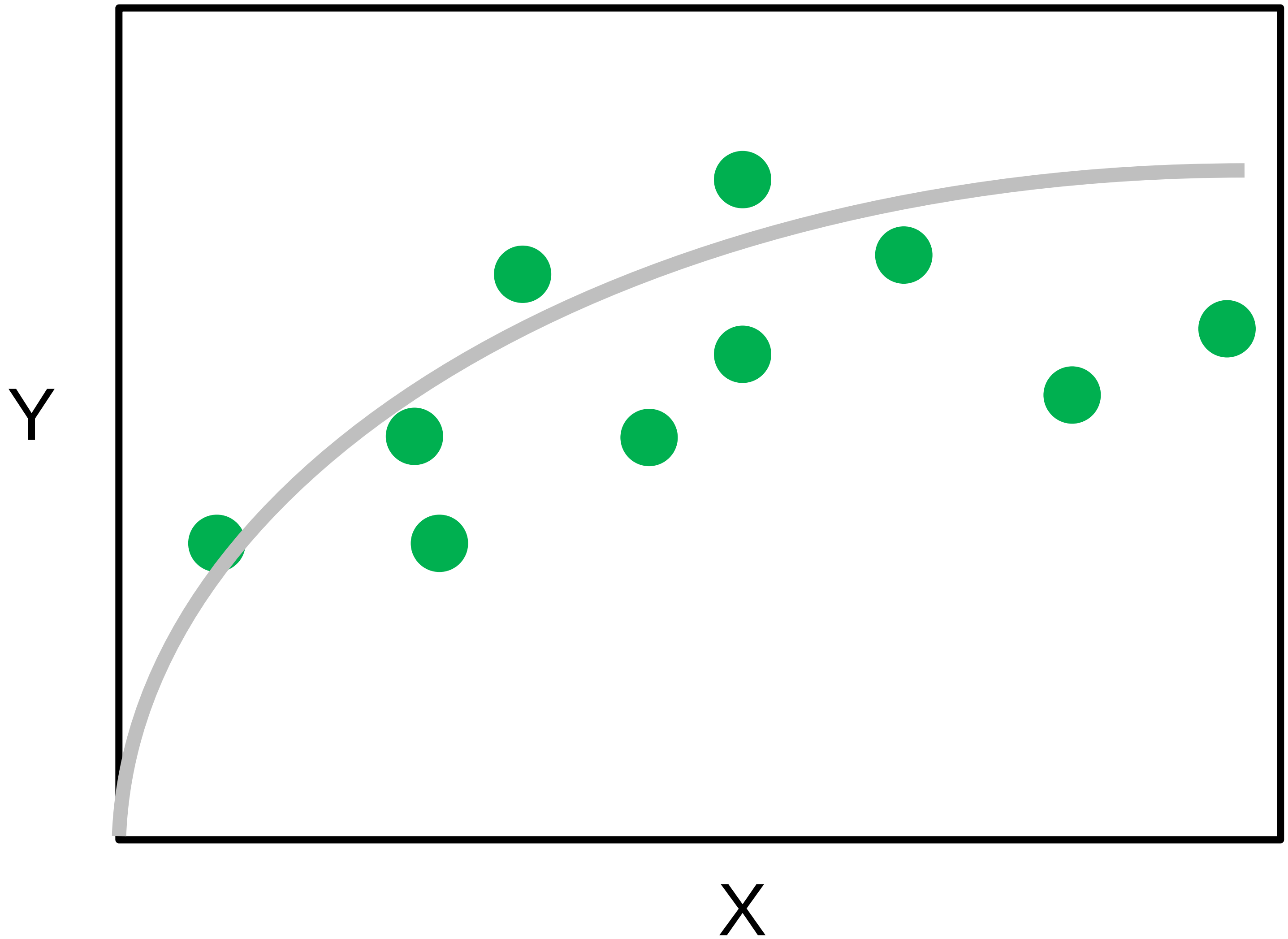


Calculate how well the Straight Line Model fit the training data by calculating their sums of squares error, i.e. distance from the fitted line to the data, square them and add them up. SSE = High

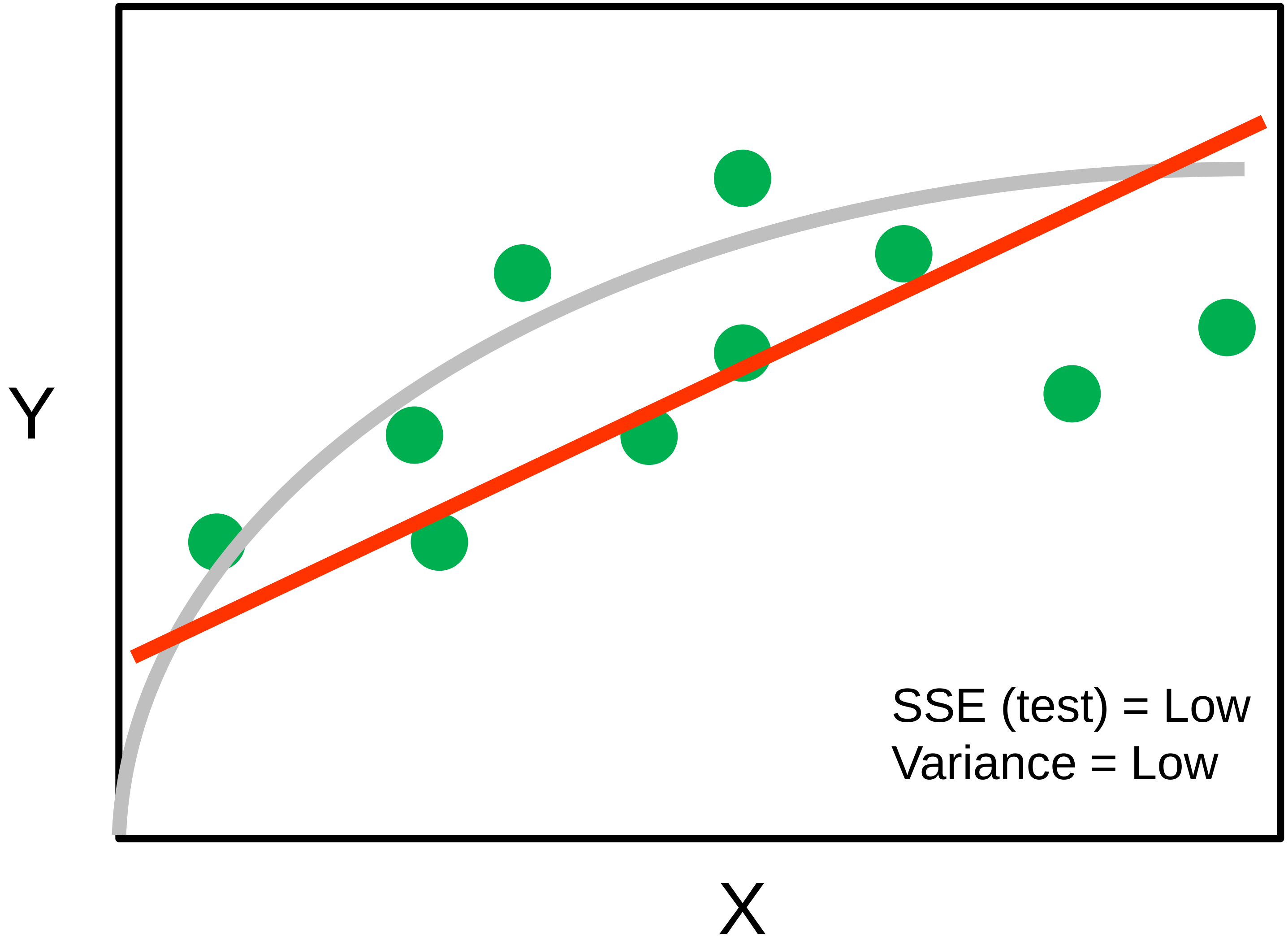
Model #2 (Zigzag Line)



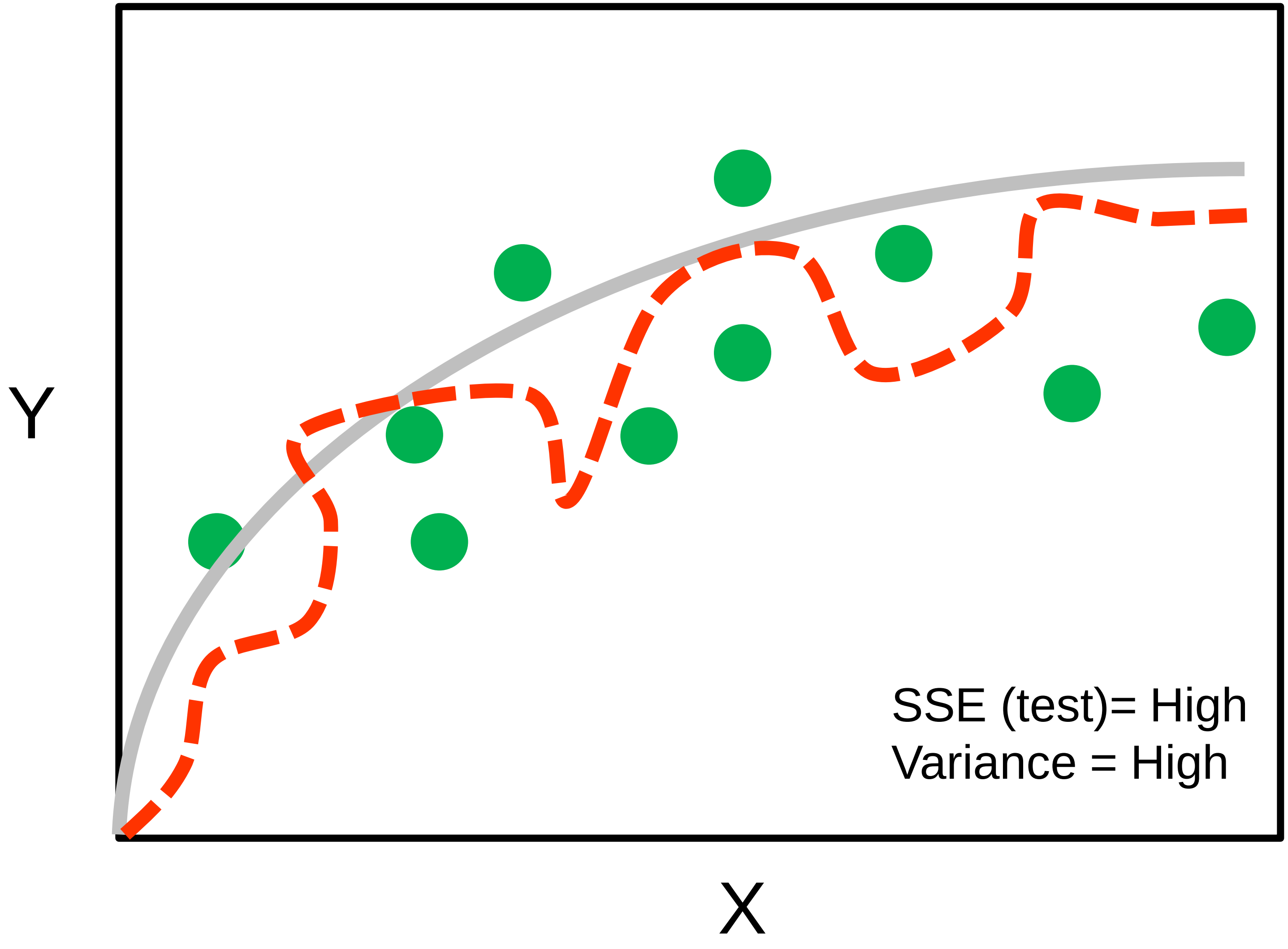
Test (Unseen) Data



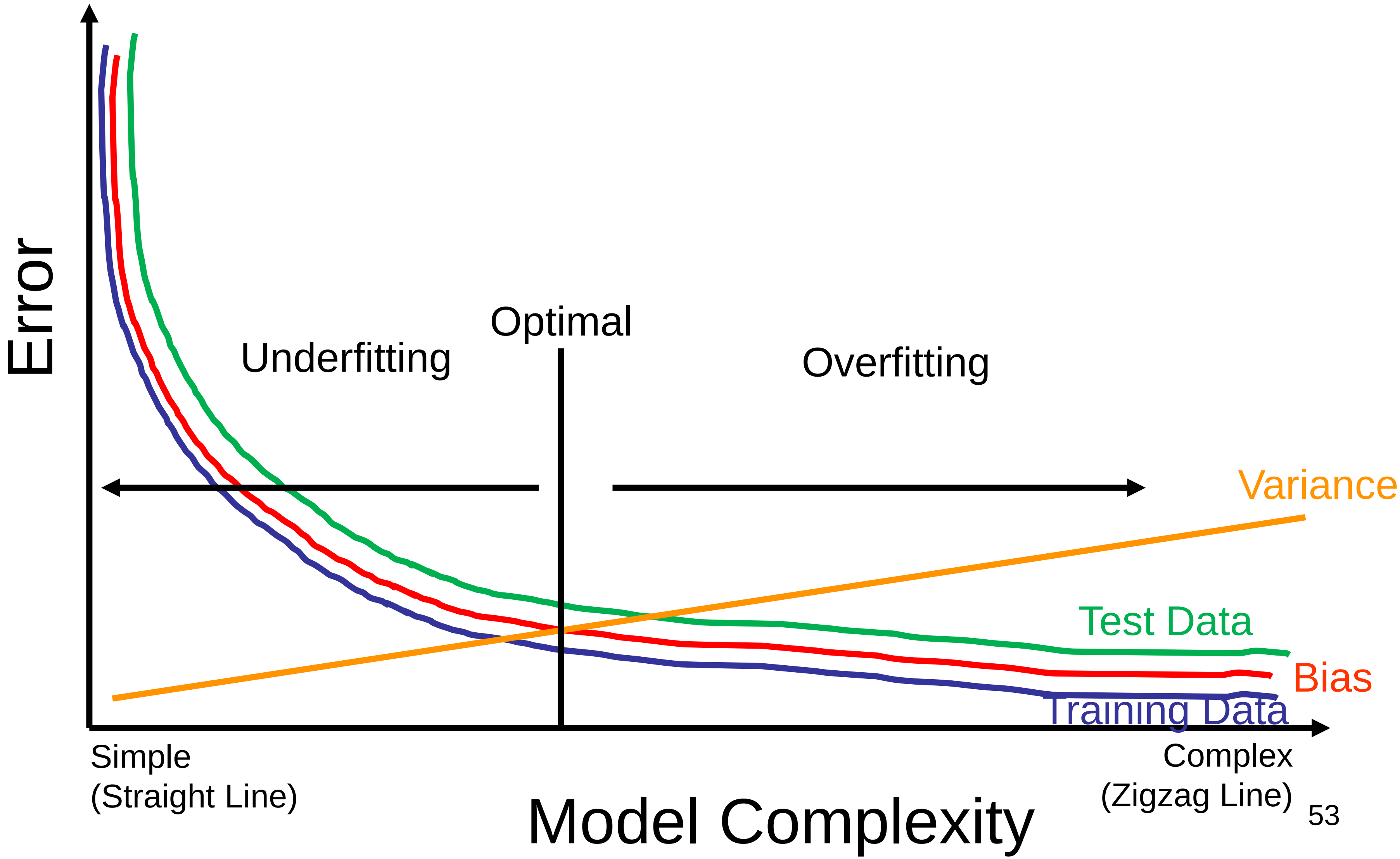
Test Data Model #1



Test Data Model #2



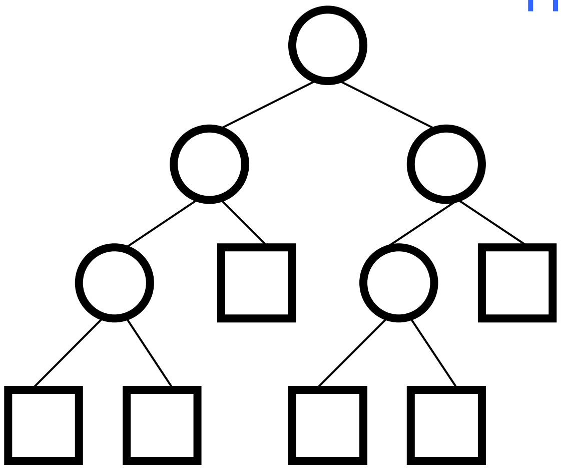
Variance vs Bias Trade-off



Overfitting

Overfitting : A classifier that performs good on the training examples but poor on unseen instances.

Low Training-set error: % errors on training data
High Generalization error: % errors on unseen data



DT1

Train and test on same data →
good classifier with massive overfitting

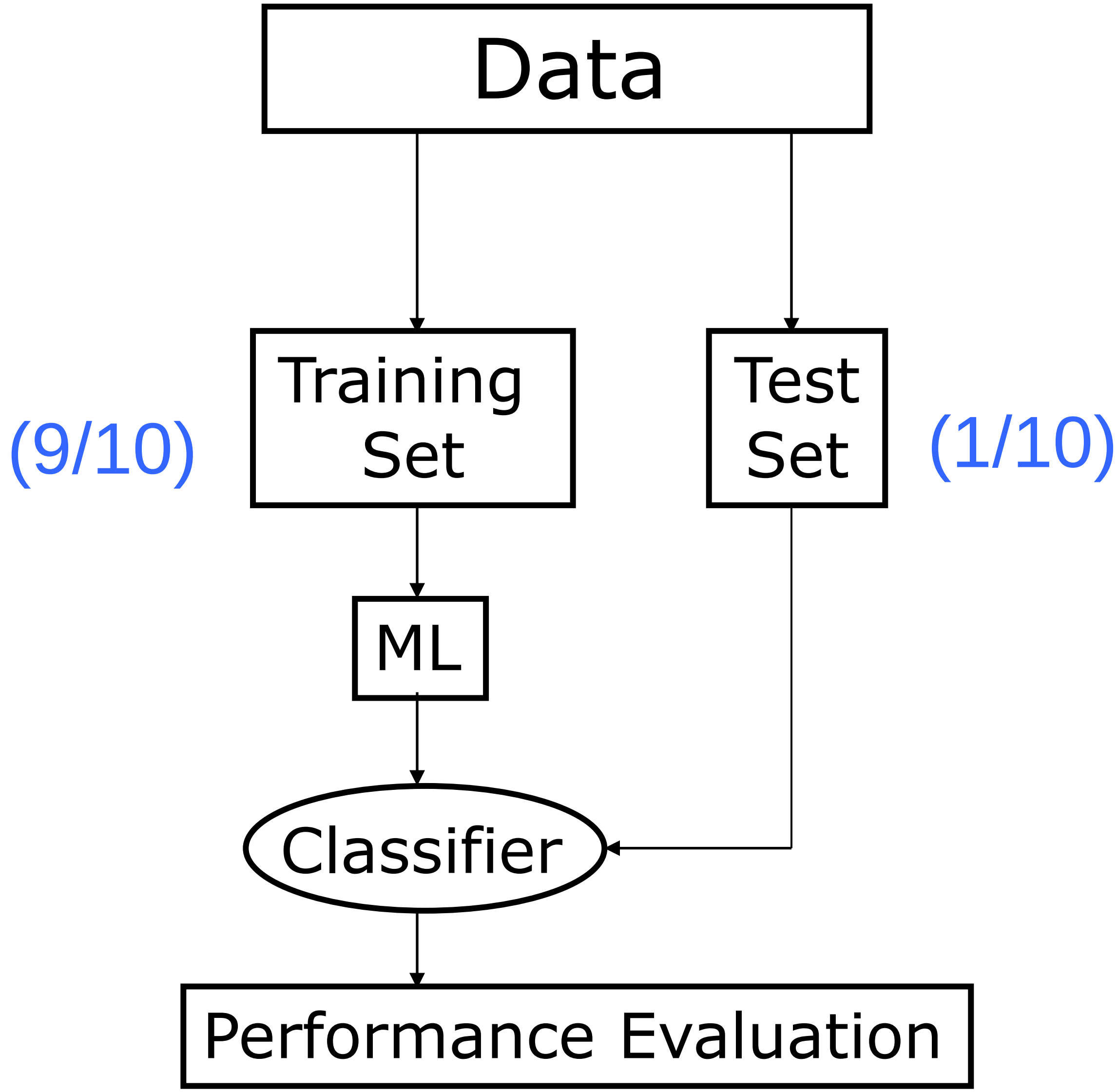
To avoid overfitting:

- Pruning the model
- Cross-validation (Computational expensive)
- Simpler model (Occam's razor)

Comparison between classifiers

- Size (Complex? Simple?)
- Sensitivity, specificity?
- Coverage?
- Receiver Operating Characteristic (ROC) Curve
- Interpretability

10-Fold Cross-validation



Confusion matrix / Contingency Table

		Predicted		
		Positive	Negative	
Actual	Positive	TP	FN	Positive Examples
	Negative	FP	TN	Negative Examples

True Positives(TP): $x \in X+$ and $h(x)$ = TRUE
True Negatives(TN): $x \in X-$ and $h(x)$ = FALSE
False Positives(FP): $x \in X-$ and $h(x)$ = TRUE
False Negatives(FN): $x \in X+$ and $h(x)$ = FALSE

Performance measurements

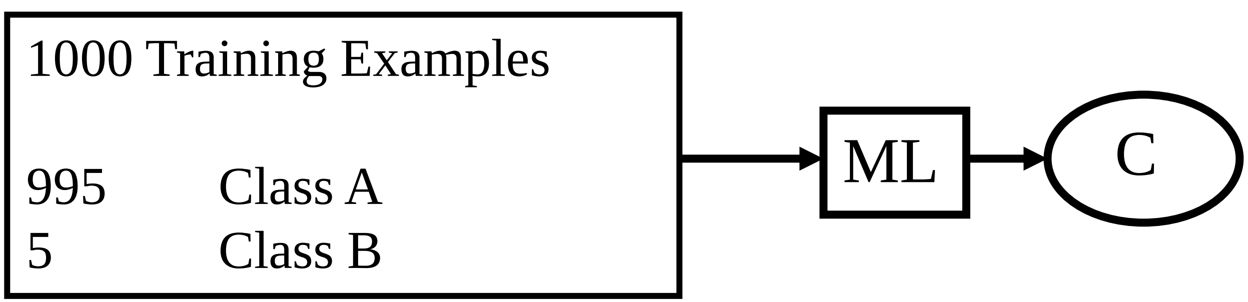
Accuracy

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \qquad 0 \leq \text{Accuracy} \leq 1$$

Accuracy Error, ε = 1 - Accuracy

NOT the good measurement for evaluating classifier's performance!!

IF the classes are unequally represented in the training examples



Class A

Accuracy = 99.5%!!

Prediction Reliability

Reliability of Positive Prediction (Positive Predicted Value / Precision)

$$PPV = \frac{TP}{TP + FP}$$

$0 \leq \text{PPV} \leq 1$

Reliability of Negative Prediction (Negative Predicted Value)

$$NPV = \frac{TN}{TN + FN}$$

$0 \leq \text{NPV} \leq 1$

More measurements ...

TP-rate (Sensitivity / Recall)

$$Sn = \frac{TP}{TP + FN}$$

$0 \leq \text{Sn} \leq 1$

TN-rate (Specificity)

$$Sp = \frac{TN}{TN + FP}$$

$0 \leq \text{Sp} \leq 1$

FP-rate

$$FP - rate = \frac{FP}{FP + TN}$$

$0 \leq \text{FP-rate} \leq 1$

FN-rate

$$FN - rate = \frac{FN}{TP + FN}$$

$0 \leq \text{FN-rate} \leq 1$

Other Statistical Measurements

F – measure (van Rijsbergen)

$$F - measure = \frac{2 \times recall \times precision}{recall + precision} = \frac{2TP}{2TP + FP + FN}.$$

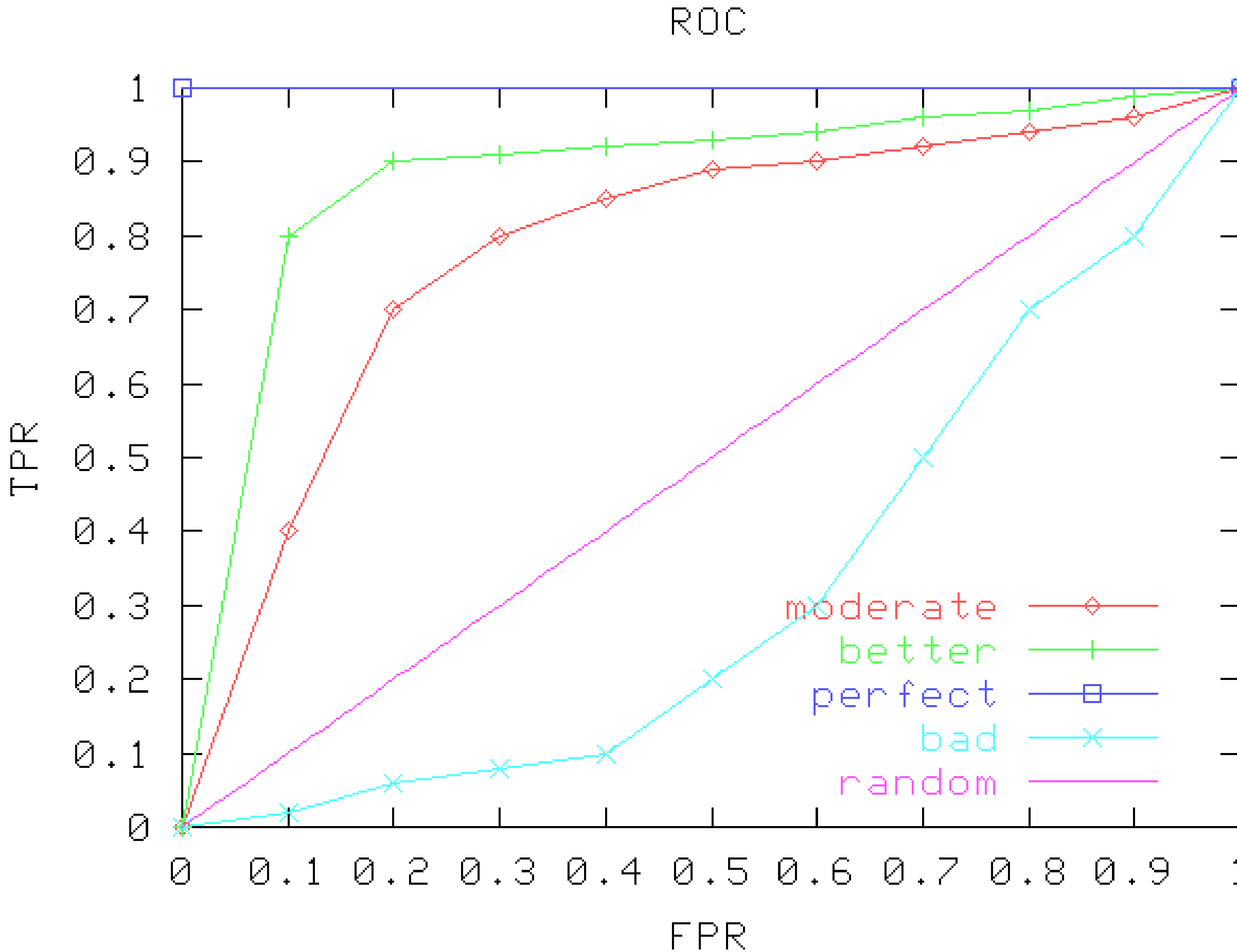
Coefficient Correlation

$$cc = \frac{(TP * TN - FP * FN)}{\sqrt{(TP + FP) * (FP + TN) * (TN + FN) * (FN + TP)}}$$

$-1 \leq cc \leq 1$

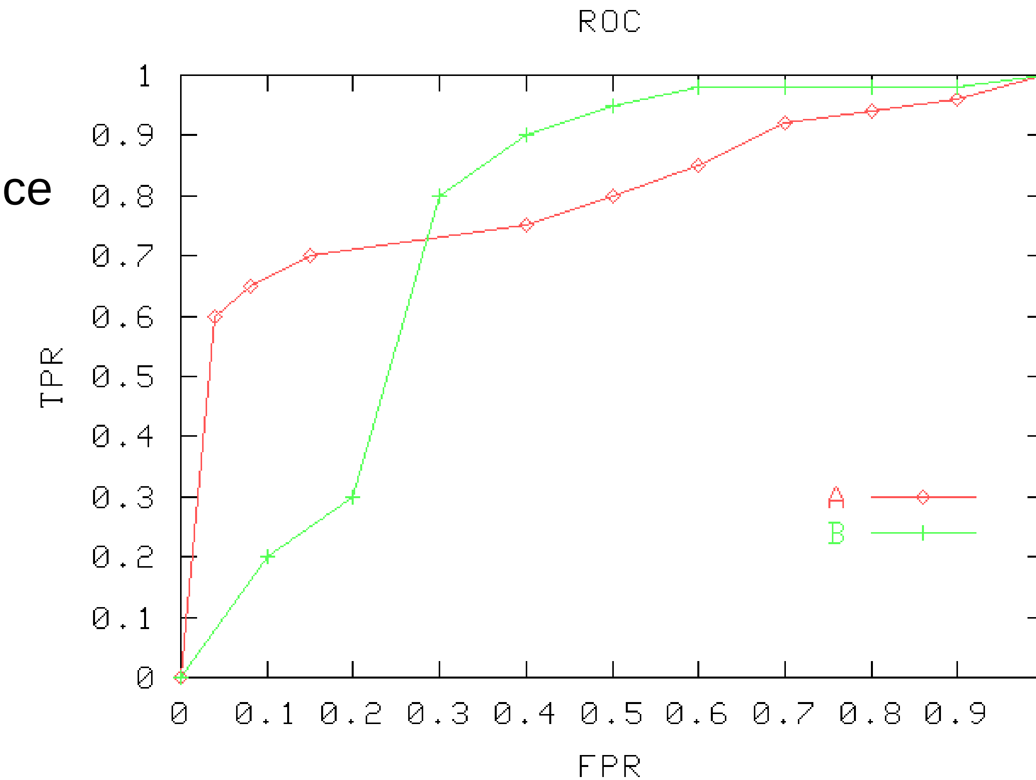
cc:
- 1.0 no FP or FN
- 0.0 when *f* is random with respect to S+ and S-
- -1.0 only FP and FN

Receiver Operating Curve (ROC)



Area Under Curve (AUC)

Which classifier performs better?

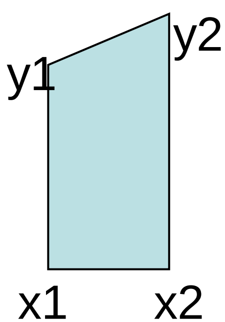


Area Under Curve (AUC) as a Measure of a classifier's performance

Area of trapezoid

The area of a trapezoid is simply the average height times the width of the base.

1. **function** *trap_area*(x1;x2; y1; y2)
2. Base = |x1-x2|
3. $Height_{avg}$ = (y1+y2)/2
4. **return** Base*$Height_{avg}$
5. **end function**



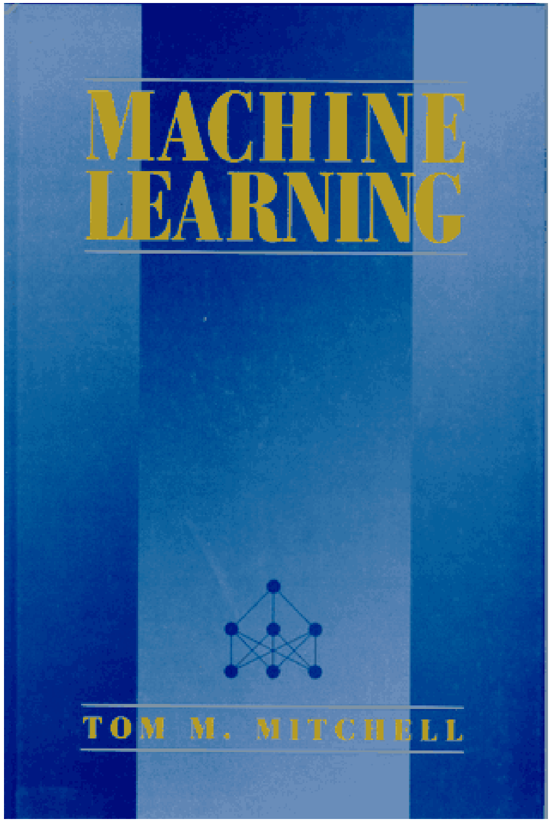
A, AUC = 0.8

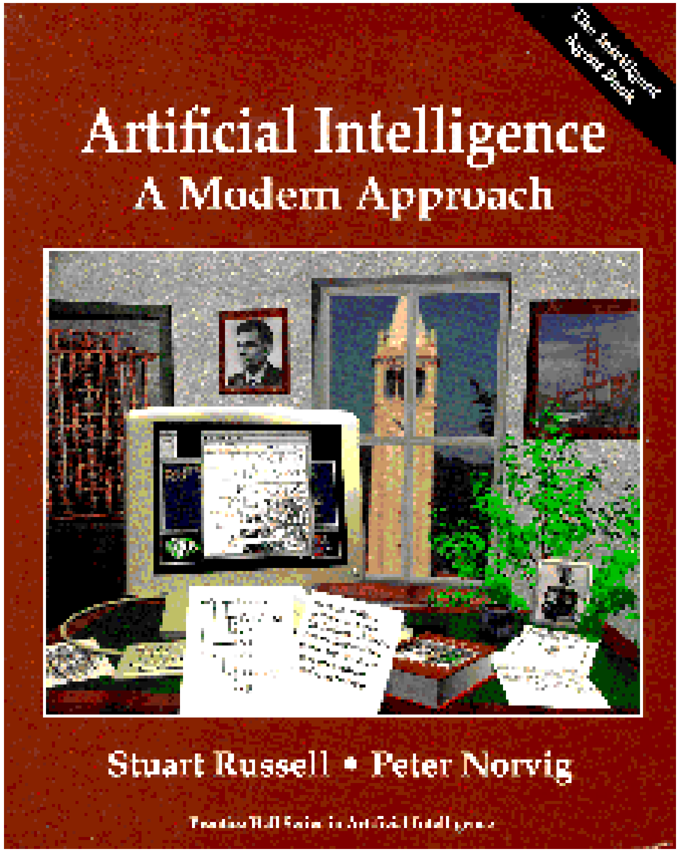
B, AUC = 0.757

Take home message

- Machine learning has been widely applied in bioinformatics, especially in the classification and clustering of high-dimensional data
- Need to understand the “problem” (task) and choose the appropriate machine learning technique
- Do compare with different methods
- The ultimate goal is to interpret the data

References


MACHINE
LEARNING
TOM M. MITCHELL


Artificial Intelligence
A Modern Approach
Stuart Russell • Peter Norvig