So Far…

- Our goal (supervised learning)
- Learn a set of discriminant functions
  - Bayesian framework
    - We could design an optimal classifier if we knew:
      - $P(\omega_i)$: priors and $P(x \mid \omega_i)$: class-conditional densities
      - Using training data to estimate $P(\omega_i)$ and $P(x \mid \omega_i)$
  - Directly learning discriminant functions from the training data (assume the form of the function is known)
    - Linear Regression
    - Logistic Regression
    - SVM
    - Kernel methods
    - Perceptron
    - Neural Network
- Other possible approaches?

k Nearest Neighbor Classifier

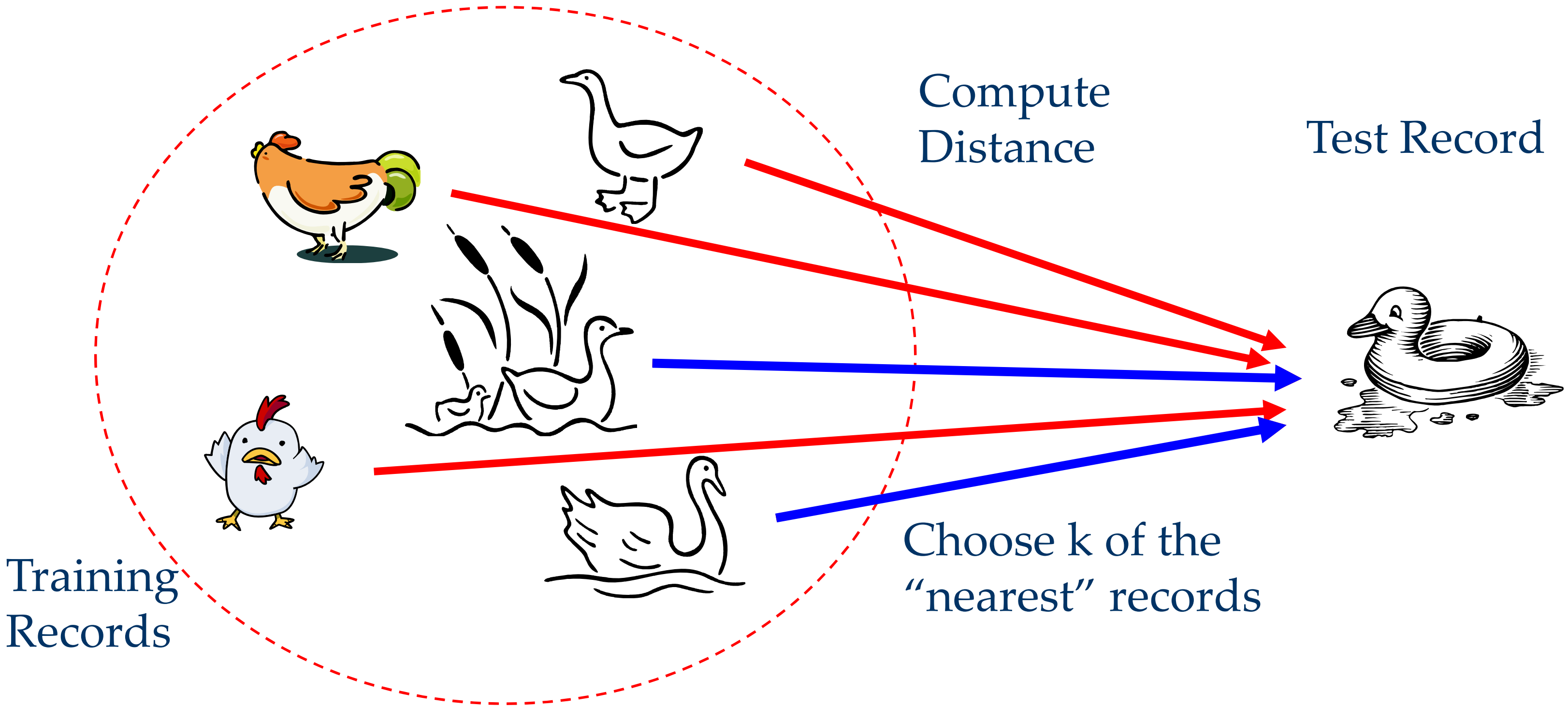
Deng Cai (蔡登)

College of Computer Science
Zhejiang University

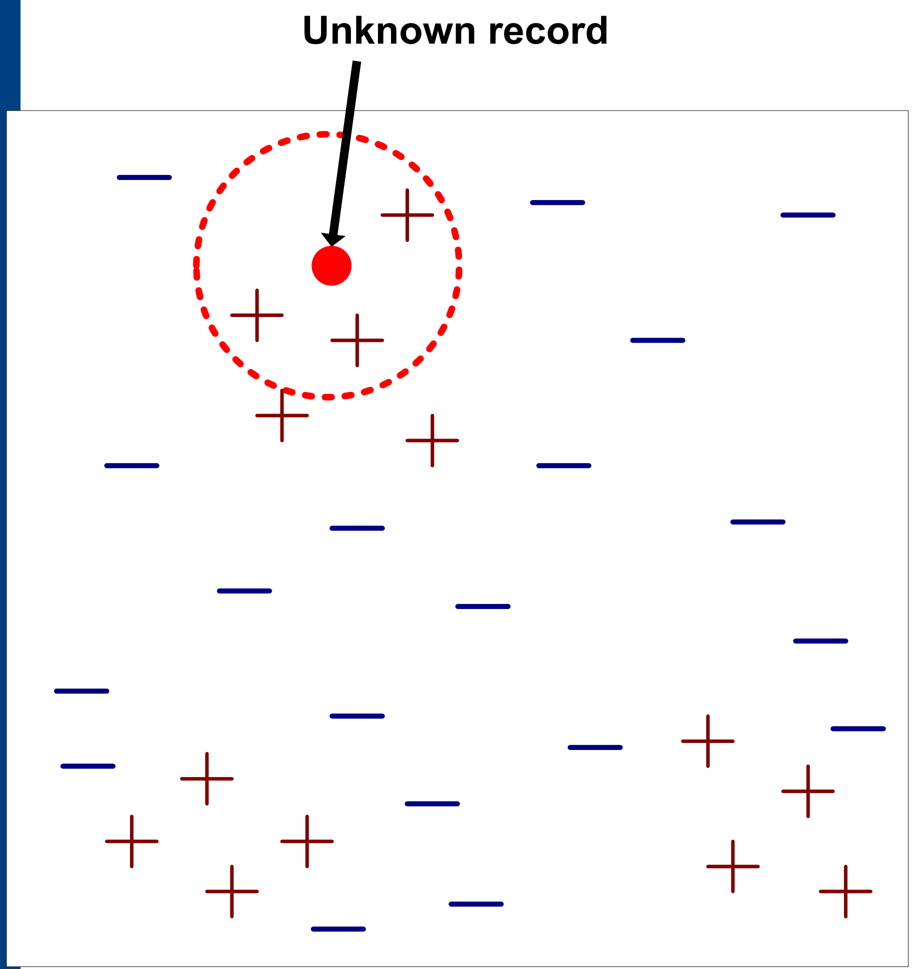
dengcai@gmail.com

Nearest Neighbor Classifiers

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



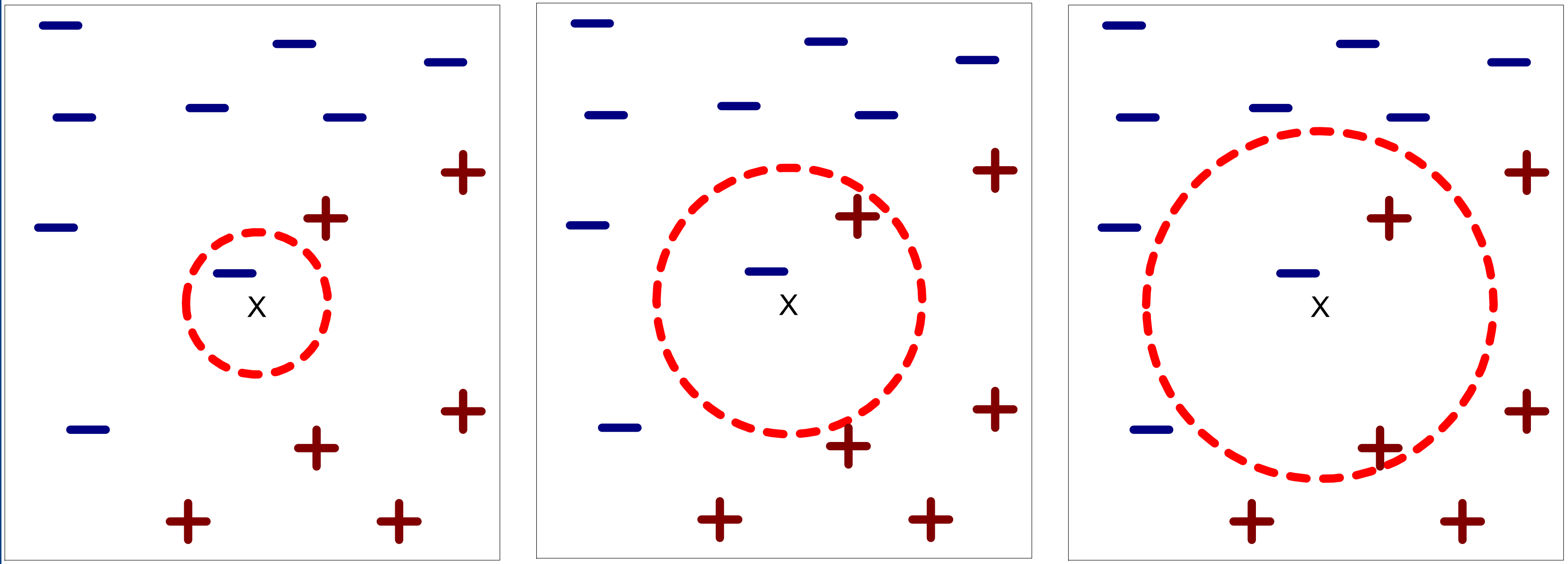
Nearest-Neighbor Classifiers



- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve

- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - 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 $\boldsymbol{x}$ are data points that have the k smallest distance to $\boldsymbol{x}$

How many parameters in kNN?

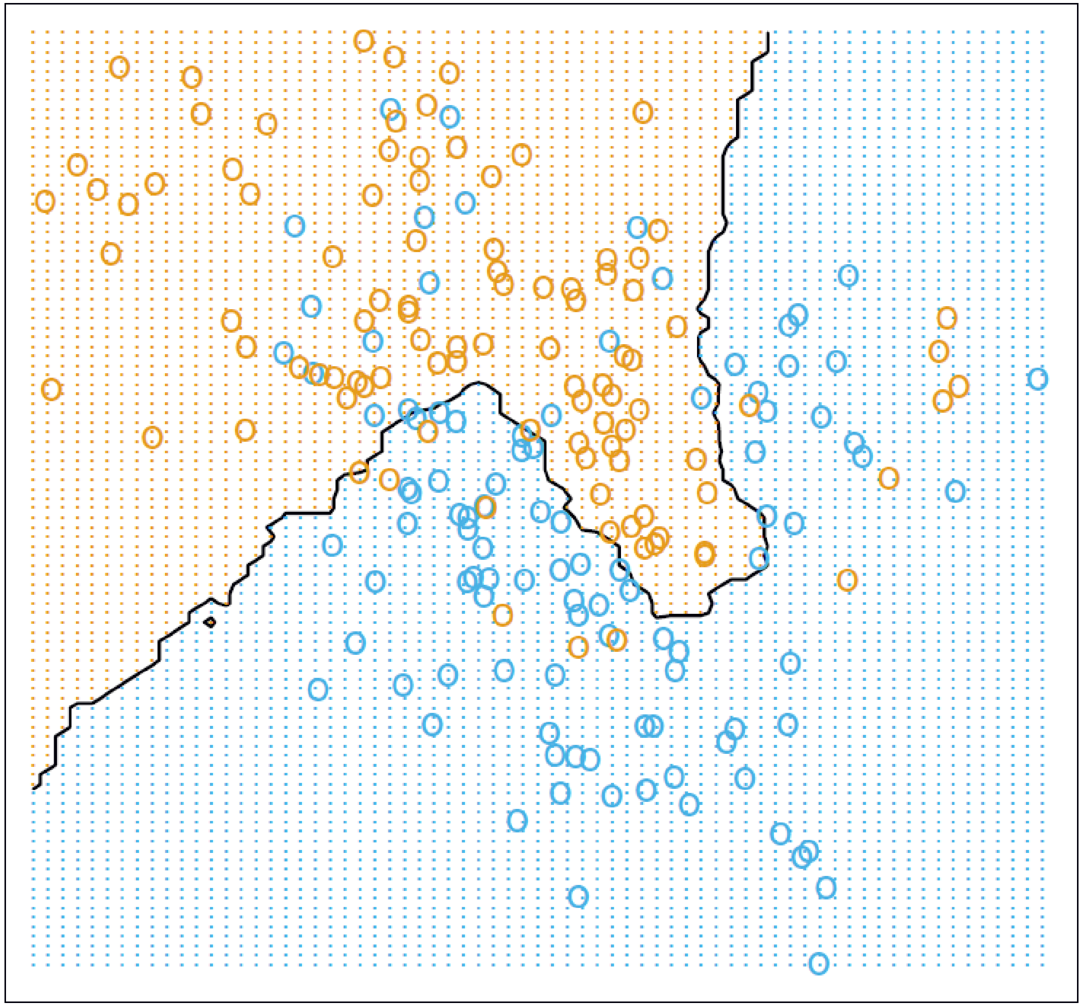
- A Linear Classifier

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

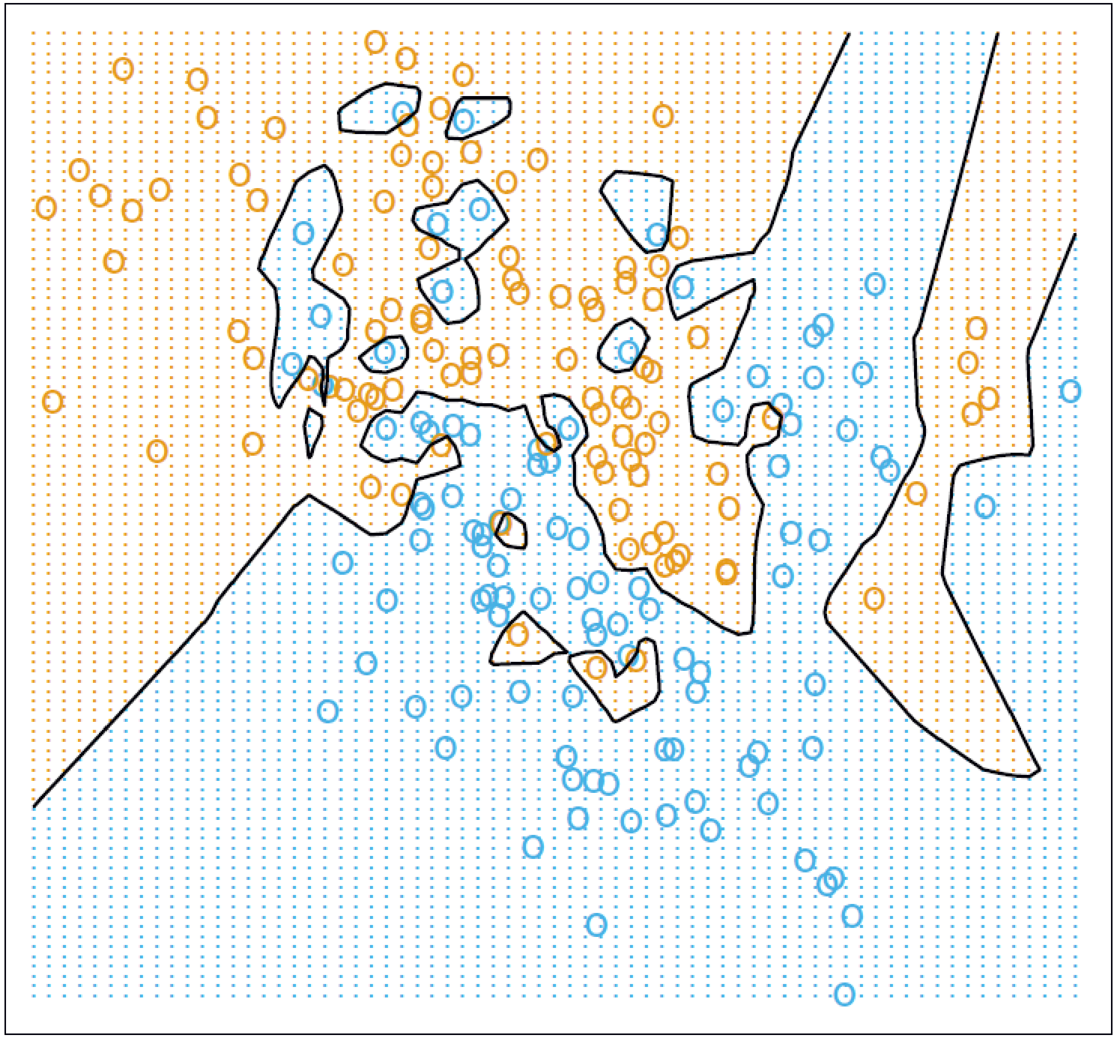
  - The number of parameters?

- kNN Classier
  - The number of parameters?

15-Nearest Neighbor Classifier



1-Nearest Neighbor Classifier



How many parameters in kNN?

- A Linear Classifier

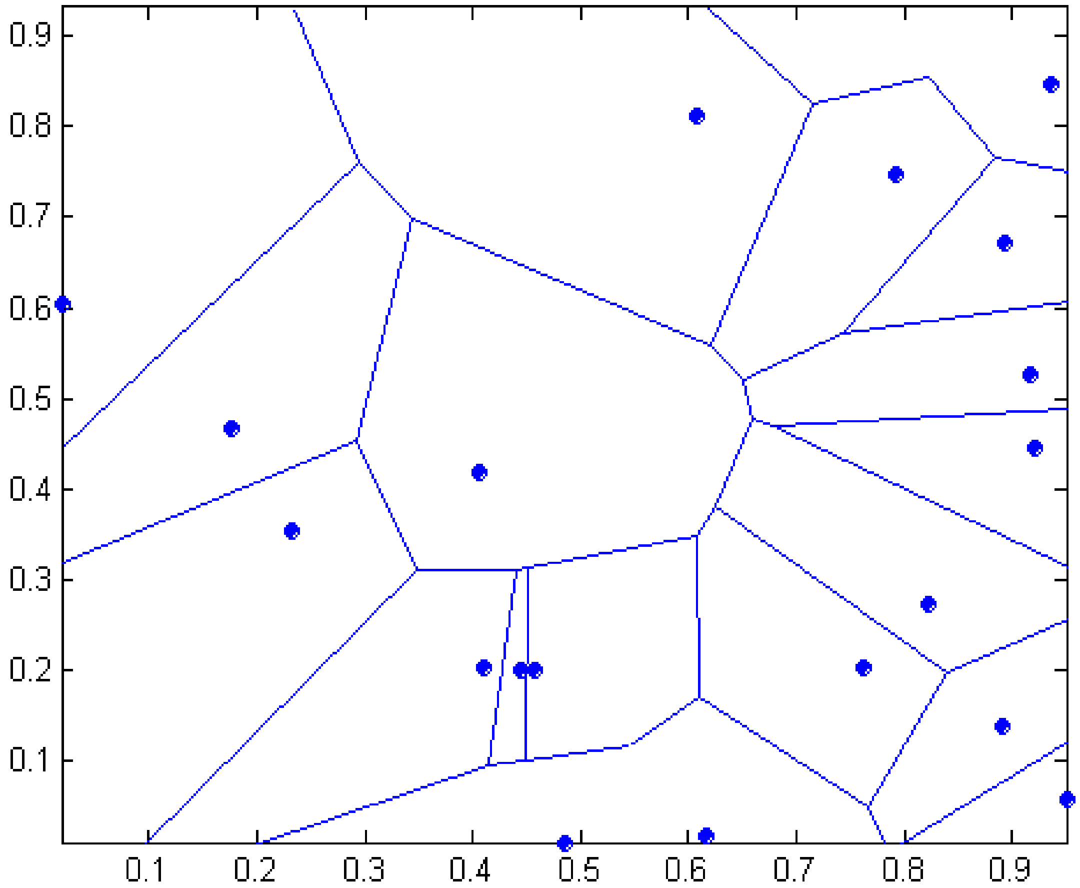
  $$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}$$

  - The number of parameters?

- kNN Classifier
  - Effective number of parameters?

    $$\frac{N}{k}$$

1 nearest-neighbor



Voronoi diagram
(tessellation)

Decision Tree

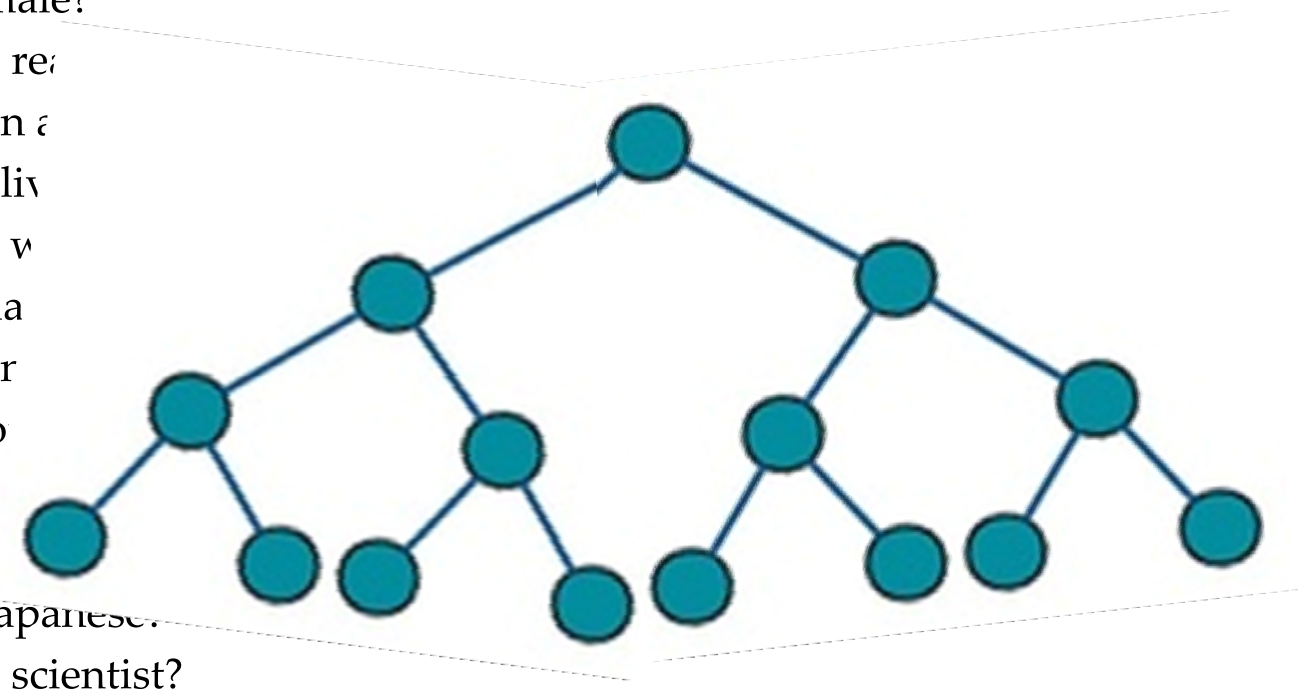
Deng Cai (蔡登)

College of Computer Science
Zhejiang University

dengcai@gmail.com

Decision Tree

- Yes/No Question Game
  - Aim: To guess the name of a famous person or character by asking yes/no questions
  - Example questions:
    - Are you male?
    - Are you a rea
    - Are you an a
    - Are you aliv
    - Have you w
    - Do you pla
    - Are you ar
    - Do you co
    - Do you w
    - Do you p
    - Are you Japanese?
    - Are you a scientist?
    - Are you a cartoon character?

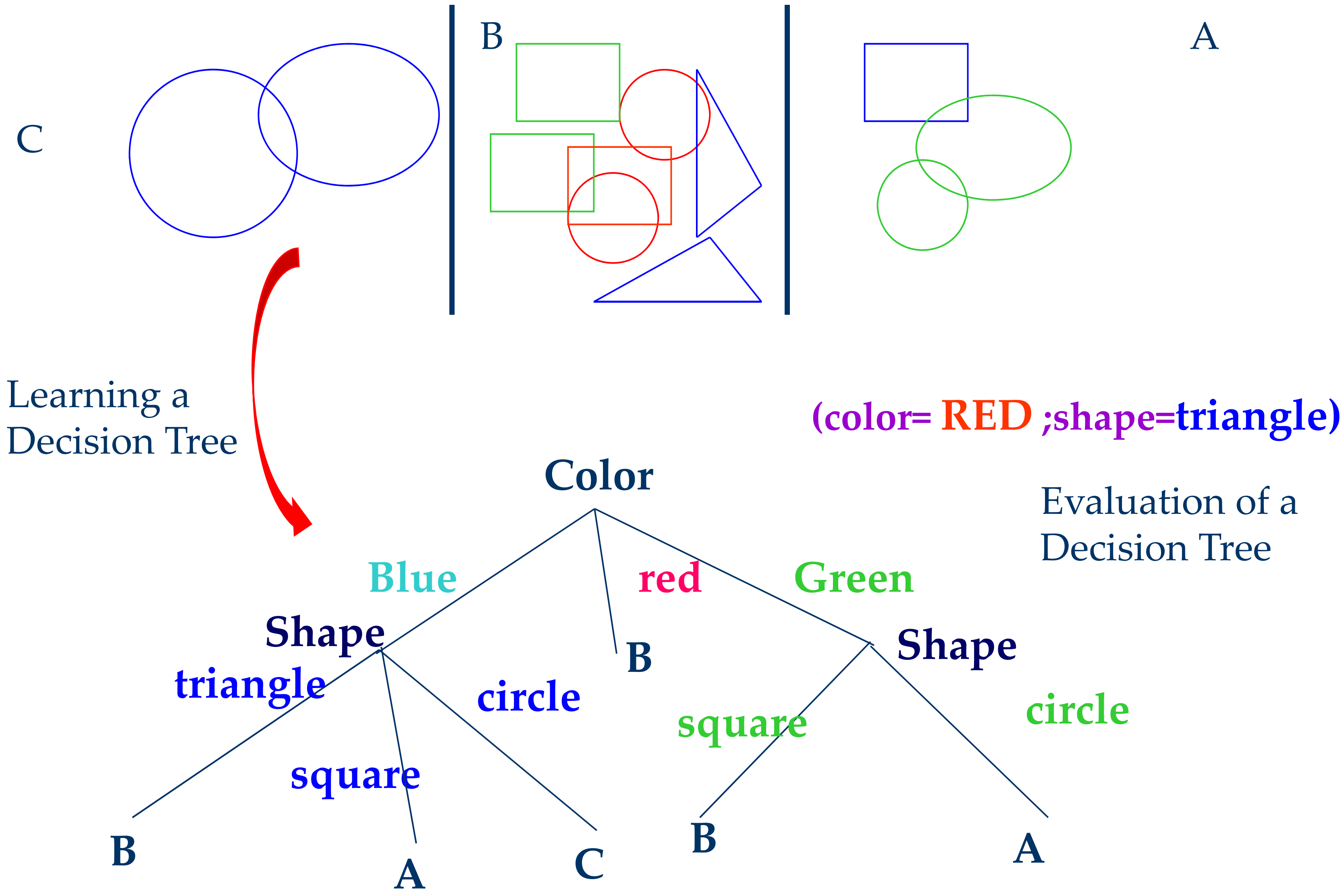


Decision Tree

- A hierarchical data structure that represents data by implementing a divide and conquer strategy
  - Given a collection of examples, learn a decision tree that represents it.
  - Use this representation to classify new examples

- Can be used as a non-parametric classification and regression method.

Decision Tree



Decision Tree

- Decision Trees are classifiers for instances represented as features vectors (color= ;shape= ;label=)
- Nodes are tests for feature values;
- There is one branch for each value of the feature
- Leaves specify the categories (labels)
- Can categorize instances into multiple disjoint categories
- Output is a discrete category. Real valued outputs are possible (regression trees)

- There are efficient algorithms for processing large amounts of data. (But not too many features)
- There are methods for handling noisy data (classification noise and attribute noise) and for handling missing attribute values.

Basic Decision Tree Learning Algorithm

- Data is processed in Batch (I.e., all the data is available).
- Recursively build a decision tree top-down.

- DT(Examples, Attributes)

```
If all Examples have same label:
        return a leaf node with Label
Else
        If  Attributes is empty:
            return a leaf with majority Label
        Else
            Pick an attribute A as root
            For each value v of A
            Let Examples(v) be all the examples for which  A=v
            Add a branch out of the root for the test  A=v
                If  Examples(v)  is empty
                        create a leaf node labeled with the majority label in Examples
                Else
                        recursively create subtree by calling DT(Examples(v), Attribute-{A})
```

Decision Tree

- Yes/No Question Game
  - Aim: To guess the name of a famous person or character by asking yes/no questions
  - Example questions:
    - Are you male?
    - Are you a real person?
    - Are you an adult?
    - Are you alive?
    - Have you written a famous book?
    - Do you play a sport?
    - Are you an actor?
    - Do you come from England?
    - Do you work in Hollywood?
    - Do you play a musical instrument?
    - Are you Japanese?
    - Are you a scientist?
    - Are you a cartoon character?

How many trees we can build?

All the trees are the same? Or Some trees are better than others?

Decision Tree

- A better tree (model)
  - Small trees

$$\text{EPE}(f) = \mathbf{(bias)^2 + variance + noise}$$

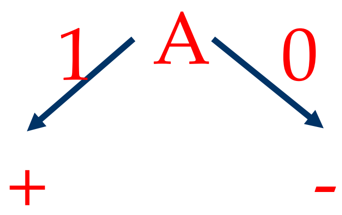
  - Same bias, less variance.
- Finding the minimal decision tree consistent with the data is NP-hard
- The recursive algorithm is a greedy heuristic search for a simple tree, but cannot guarantee optimality.
- The main decision in the algorithm is the selection of the next attribute to condition on.

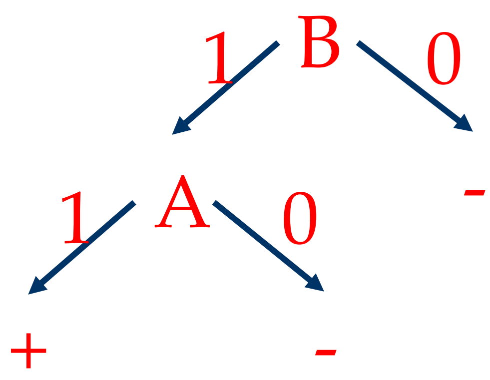
Picking the Best Feature (Attribute)

- Consider data with two Boolean attributes (A,B).

  < (A=0,B=0), - >: 50 examples
  < (A=0,B=1), - >: 50 examples
  < (A=1,B=0), - >: 0 examples
  < (A=1,B=1), + >: 100 examples

- What should be the first attribute we select? A or B?
- Splitting on A: we get purely labeled nodes
- Splitting on B: we don't get purely labeled nodes.

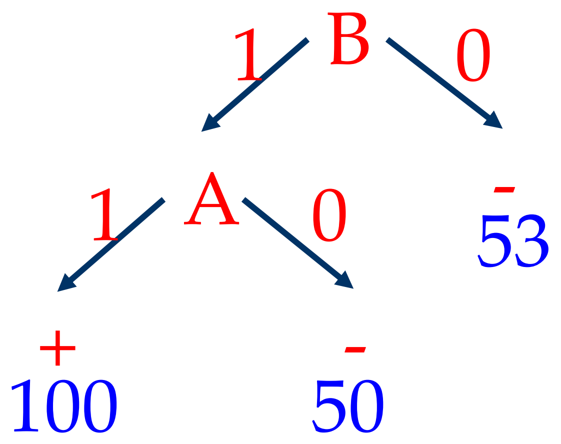


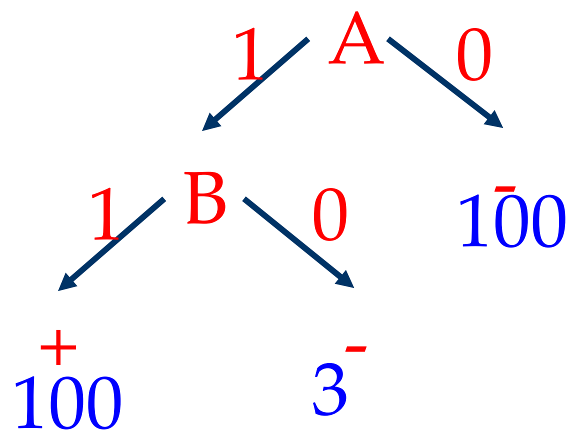


Picking the Best Feature (Attribute)

- Consider data with two Boolean attributes (A,B).
  - < (A=0,B=0), - >: 50 examples
  - < (A=0,B=1), - >: 50 examples
  - < (A=1,B=0), - >: **3** examples
  - < (A=1,B=1), + >: 100 examples

- Trees looks structurally similar; which attribute should we choose?





Advantage A. But…
Need a way to quantify things

Picking the Best Feature (Attribute)

- We want attributes that split the examples to sets that are relatively pure in one label; this way we are closer to a leaf node.

- The most popular heuristics is based on information gain, originated with the ID3 system of Quinlan.

Entropy

- Entropy (impurity, disorder) of a set of examples, S, relative to a binary classification is:

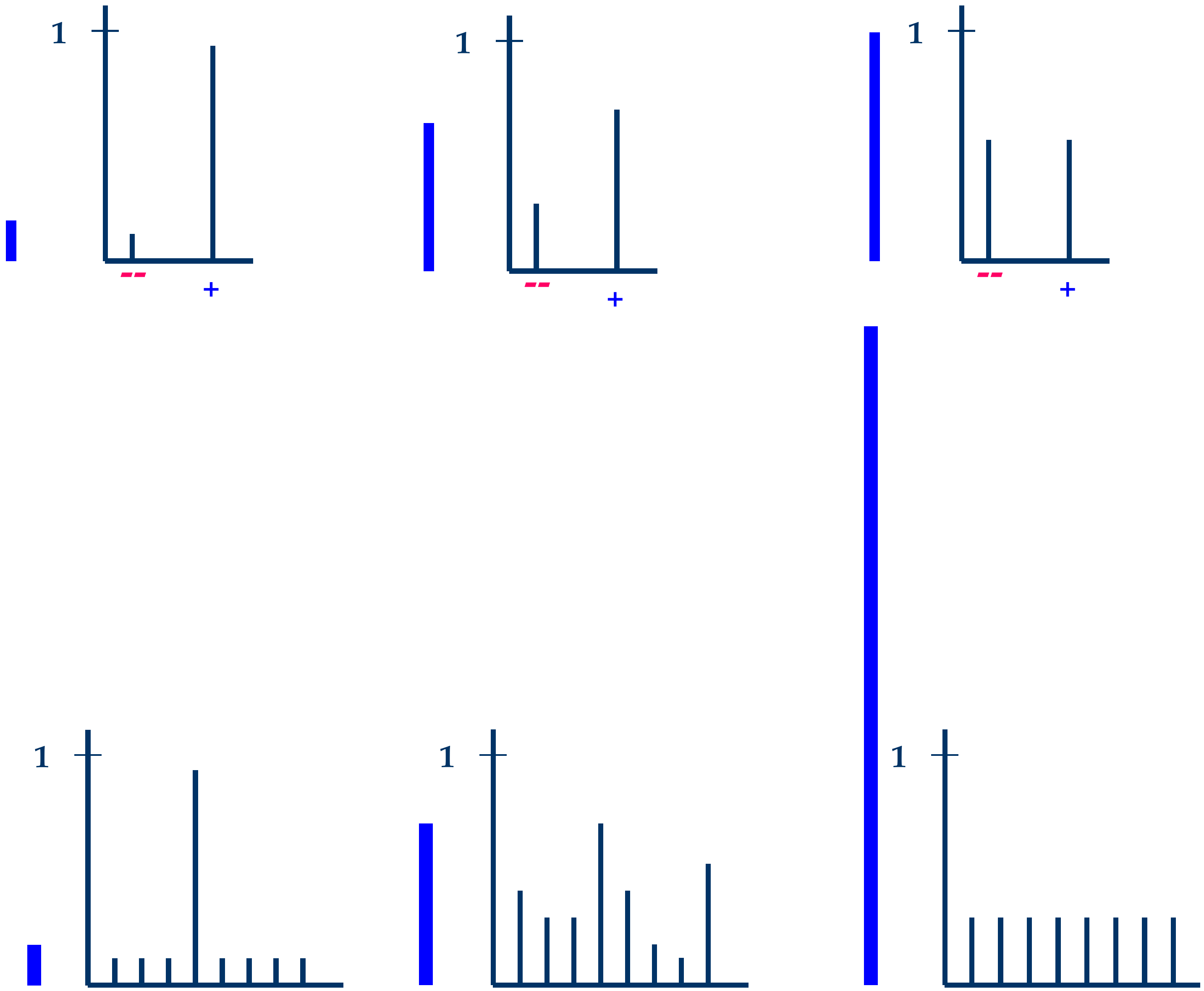
$$Entropy(S) = -P_+ \log P_+ - P_- \log P_-$$

  - where P_+ is the proportion of positive examples in S
  - P_- is the proportion of negative examples
  - If all the examples belong to the same category $Entropy = 0$
  - If the examples are equally mixed (0.5,0.5) $Entropy = 1$
- **In general, when p_i is the fraction of examples labeled i:**

$$Entropy(S) = -\sum_{i=1}^{c} P_i \log P_i$$

- Entropy can be viewed as the number of bits required, on average, to encode the class of labels. If the probability for + is 0.5, a single bit is required for each example; if it is 0.8 -- can use less then 1 bit.

Entropy



Picking the Best Feature (Attribute)

- We want attributes that split the examples to sets **that are relatively pure in one label**; this way we are closer to a leaf node.
- We can pick the feature that the resulting data partitions have low entropy

Information Gain

- **The information gain** of an attribute a is the expected reduction in entropy caused by partitioning on this attribute.

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

  - where S_v is the subset of S for which attribute a has value v

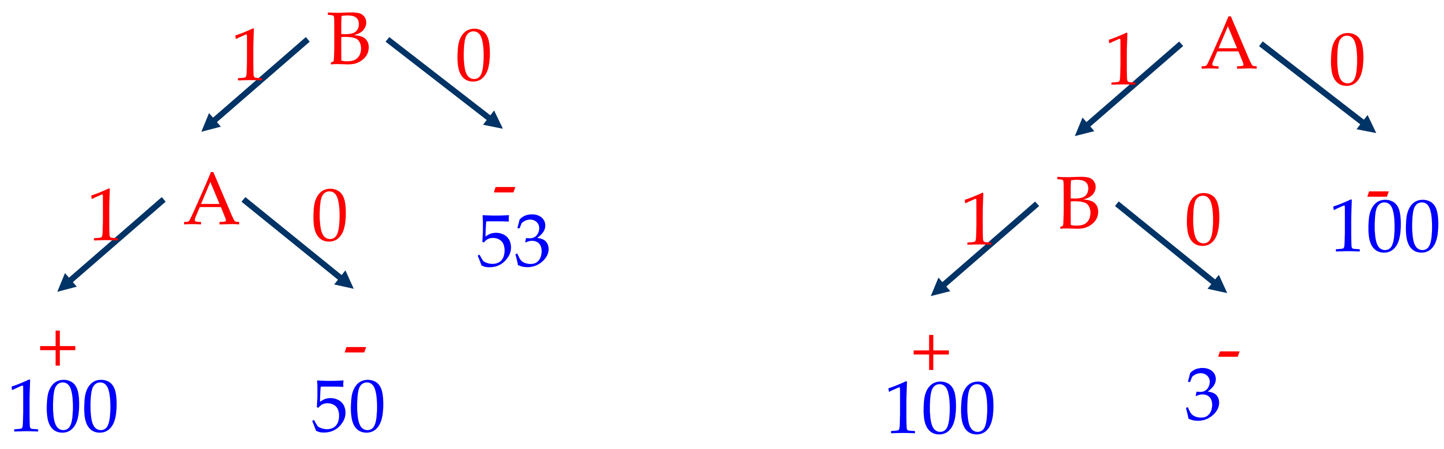
- Partitions of low entropy lead to high gain.

Picking the Best Feature (Attribute)

- Consider data with two Boolean attributes (A,B).

  < (A=0,B=0), - >: 50 examples
  < (A=0,B=1), - >: 50 examples
  < (A=1,B=0), - >: **3** examples
  < (A=1,B=1), + >: 100 examples

- Trees looks structurally similar; which attribute should we choose?



The information gain of A and B

An Illustrative Example

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

An Illustrative Example (2)

$$Entropy(S) = -\frac{9}{14}\log\left(\frac{9}{14}\right) - \frac{5}{14}\log\left(\frac{5}{14}\right) = 0.94$$

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

9+,5-

An Illustrative Example (2)

Humidity	Wind	PlayTennis
High	Weak	No
High	Strong	No
High	Weak	Yes
High	Weak	Yes
Normal	Weak	Yes
Normal	Strong	No
Normal	Strong	Yes
High	Weak	No
Normal	Weak	Yes
Normal	Weak	Yes
Normal	Strong	Yes
High	Strong	Yes
Normal	Weak	Yes
High	Strong	No

9+,5-

E=.94

Humidity

High	Normal
3+,4-	6+,1-
E=.985	E=.592

Wind

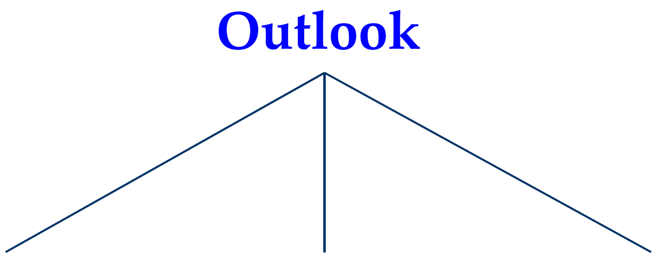
Weak	Strong
6+2-	3+,3-
E=.811	E=1.0

Gain(*S*,Humidity)=
.94 - 7/14 0.985
- 7/14 0.592=
0.151

Gain(*S*,Wind)=
.94 - 8/14 0.811
- 6/14 1.0 =
0.048

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

An Illustrative Example (3)



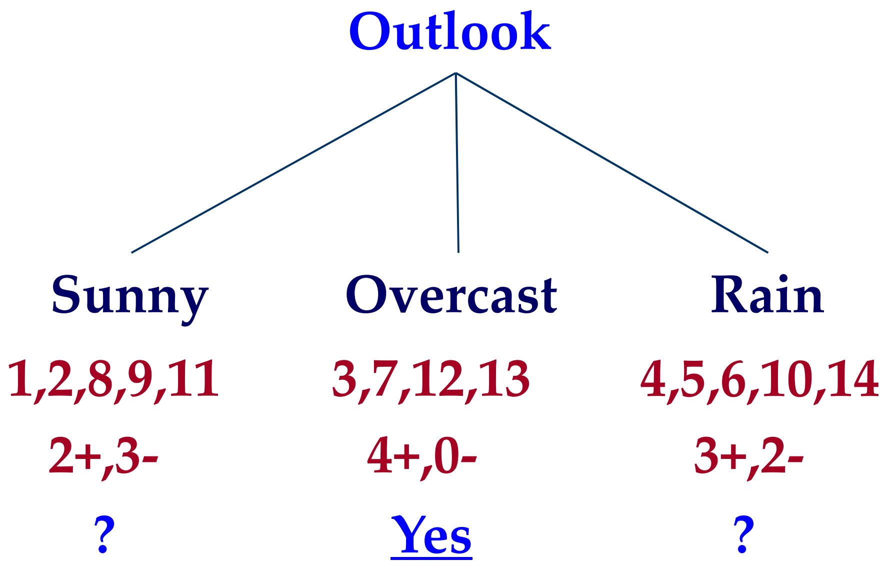
Gain(*S*,**Humidity**)=0.151

Gain(*S*,**Wind**)=0.048

Gain(*S*,**Temperature**)=0.029

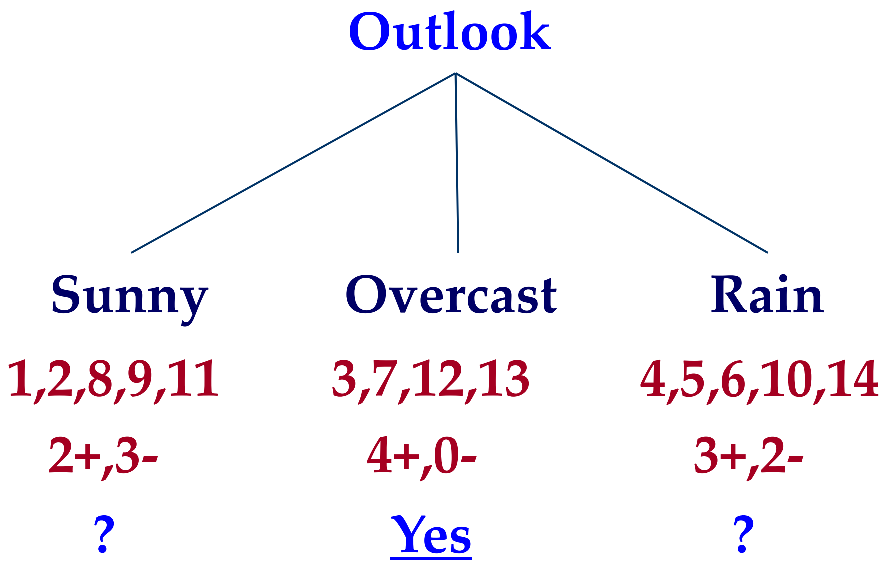
Gain(*S*,**Outlook**)=0.246

An Illustrative Example (3)



Day	Outlook	PlayTennis
1	Sunny	No
2	Sunny	No
3	Overcast	Yes
4	Rain	Yes
5	Rain	Yes
6	Rain	No
7	Overcast	Yes
8	Sunny	No
9	Sunny	Yes
10	Rain	Yes
11	Sunny	Yes
12	Overcast	Yes
13	Overcast	Yes
14	Rain	No

An Illustrative Example (3)



Continue until:

- Every attribute is included in path, or,
- All examples in the leaf have same label

Day	Outlook	PlayTennis
1	Sunny	No
2	Sunny	No
3	Overcast	Yes
4	Rain	Yes
5	Rain	Yes
6	Rain	No
7	Overcast	Yes
8	Sunny	No
9	Sunny	Yes
10	Rain	Yes
11	Sunny	Yes
12	Overcast	Yes
13	Overcast	Yes
14	Rain	No

An Illustrative Example (4)

$Gain(S_{sunny}, Humidity) =$ .97-(3/5) 0-(2/5) 0 = .97

$Gain(S_{sunny}, Temp) =$ .97- 0-(2/5) 1 = .57

$Gain(S_{sunny}, Wind) =$ .97-(2/5) 1 - (3/5) .92= .02

Outlook

Sunny	Overcast	Rain
1,2,8,9,11	3,7,12,13	4,5,6,10,14
2+,3-	4+,0-	3+,2-
?	Yes	?

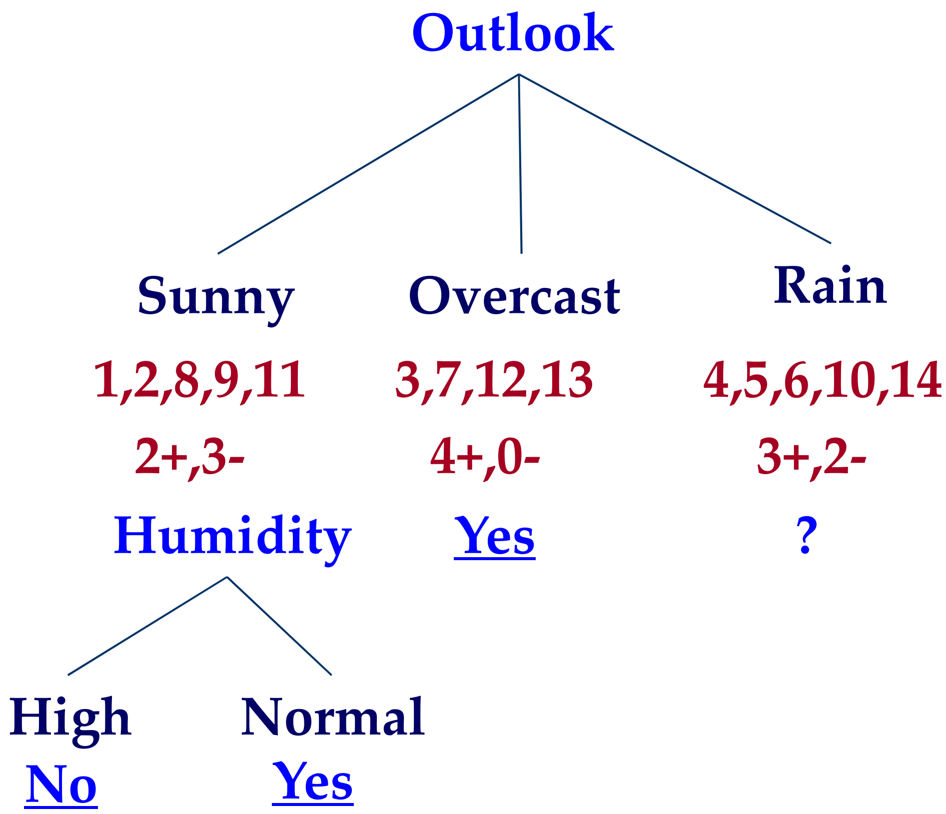
Day	Outlook	Temperature	Humidity	Wind	PlayTen
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

An Illustrative Example (5)

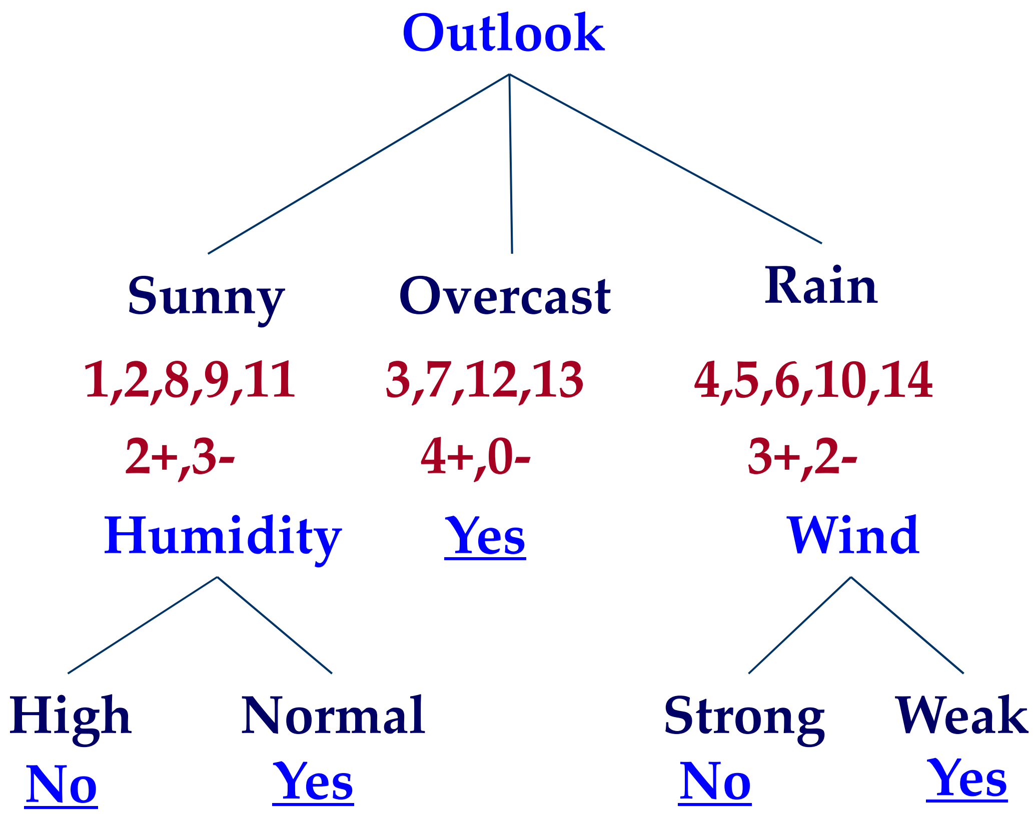
Outlook

Sunny	Overcast	Rain
1,2,8,9,11	3,7,12,13	4,5,6,10,14
2+,3-	4+,0-	3+,2-
?	<u>Yes</u>	?

An Illustrative Example (5)



An Illustrative Example (6)



Summary: ID3(Examples, Attributes, Label)

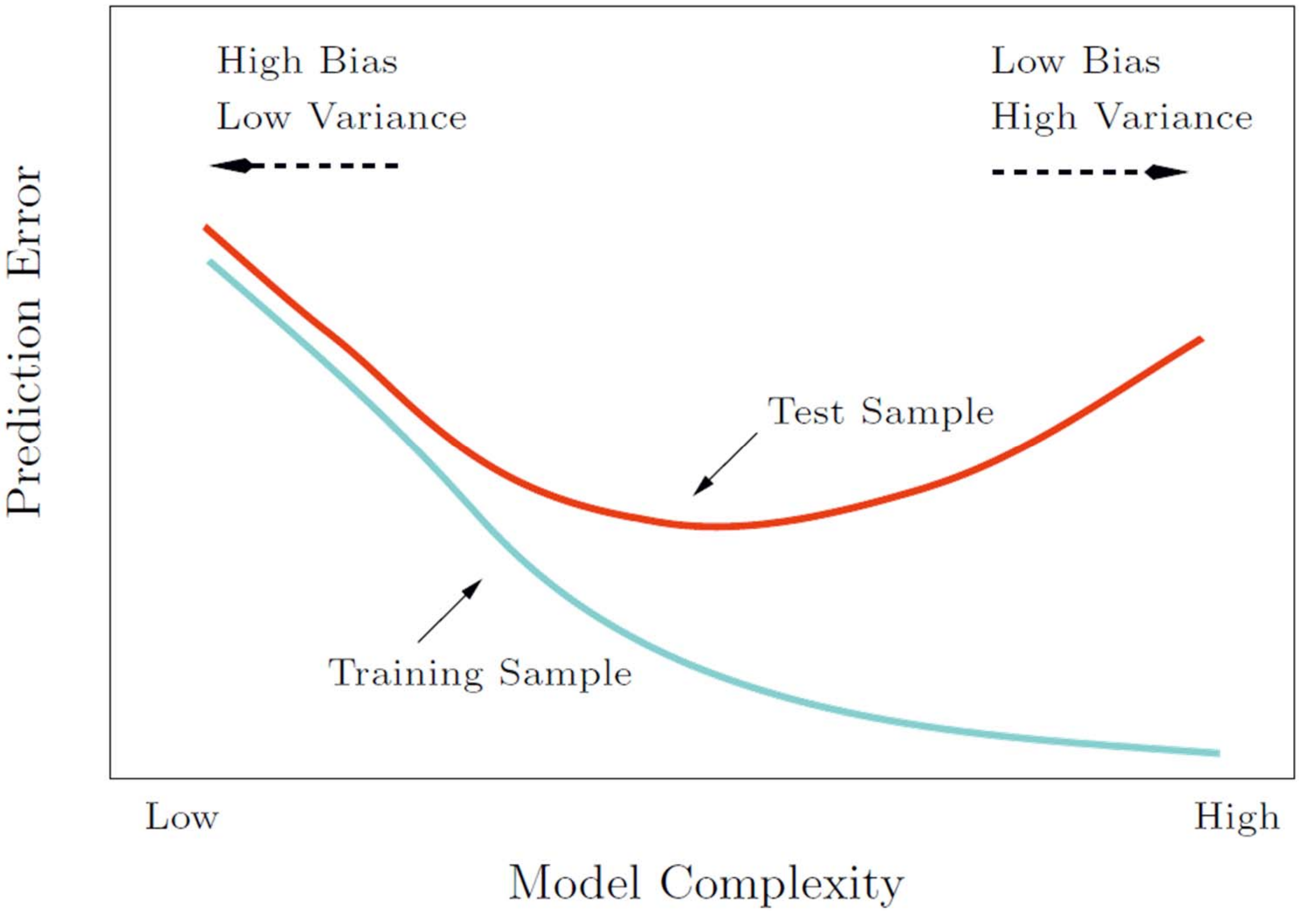
- Let S be the set of Examples
  Label is the target attribute (the prediction)
  Attributes is the set of measured attributes
-
- Create a Root node for tree
- If all examples are labeled the same return a single node tree with Label
- Otherwise Begin
- A = attribute in Attributes that *best* classifies S
- for each possible value v of A
- Add a new tree branch corresponding to A=v
- Let *Sv* be the subset of examples in S with A=v
- if *Sv* is empty: add leaf node with the common value of Label in S
- Else: below this branch add the subtree
- ID3(*Sv*, Attributes - {a}, Label)

End

Return Root

History of Decision Tree Research

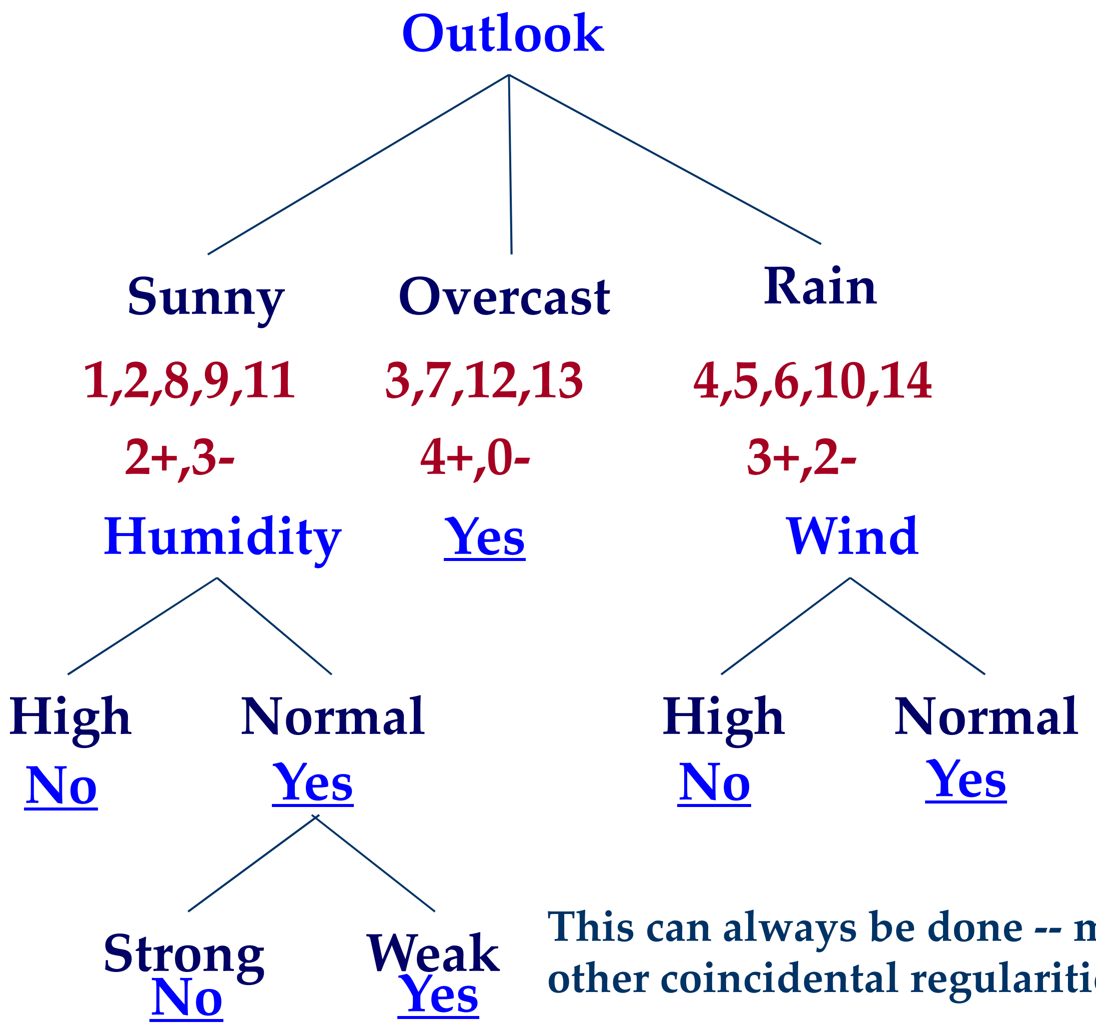
- Hunt and colleagues in Psychology used full search decision trees methods to model human concept learning in the 60's
- Quinlan developed ID3, with the information gain heuristics in the late 70's to learn expert systems from examples
- Breiman, Friedmans and colleagues in statistics developed CART (classification and regression trees) simultaneously
- A variety of improvements in the 80's: coping with noise, continuous attributes, missing data, non-axis parallel etc.
- Quinlan's updated algorithm, C4.5 (1993) is commonly used (New:C5)
- Boosting (or Bagging) over DTs is a very good general purpose algorithm



$\mathrm{EPE}(f) =$ **(bias)^2 + variance + noise**

Decision Trees: Low Bias Model

Outlook = Sunny, Temp = Hot, Humidity = Normal, Wind = Strong, NO



This can always be done -- may fit noise or other coincidental regularities

Avoid Overfitting

- How the overfitting can this be avoided with linear classifiers?
- For Decision Trees: Two basic approaches
  - Prepruning:
    - Stop growing the tree at some point during construction when it is determined that there is not enough data to make reliable choices.
  - Postpruning:
    - Grow the full tree and then remove nodes that seem not to have sufficient evidence.
- Validation Set