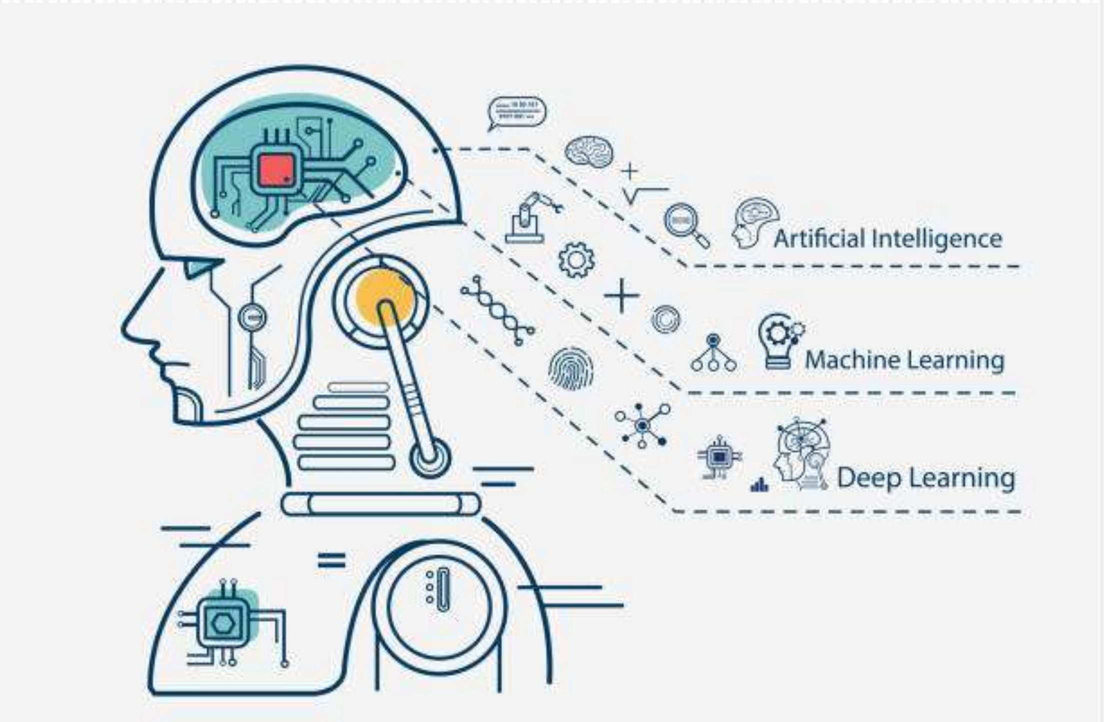
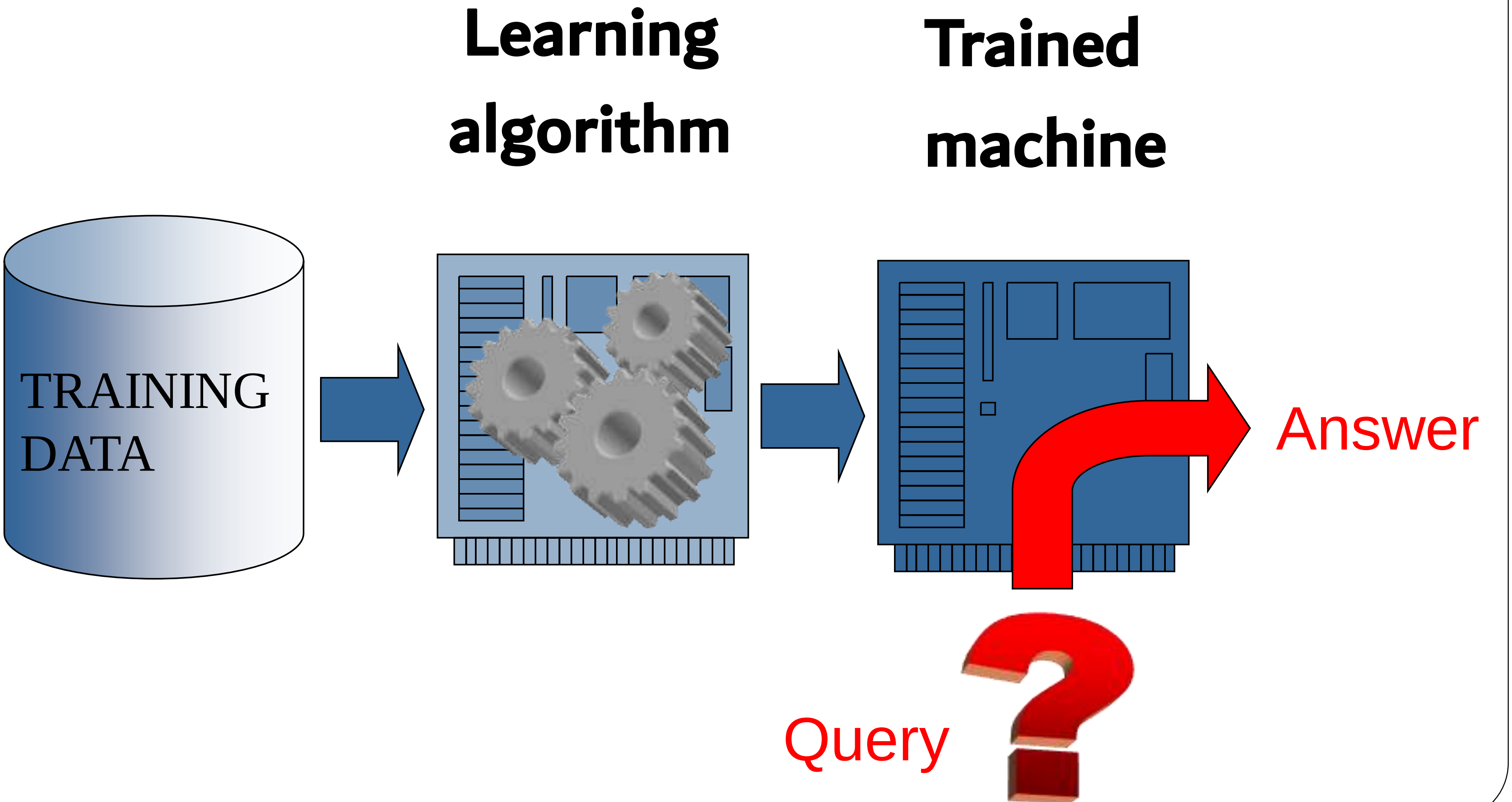
# CSE 440
# Artificial Intelligence

## Machine Learning



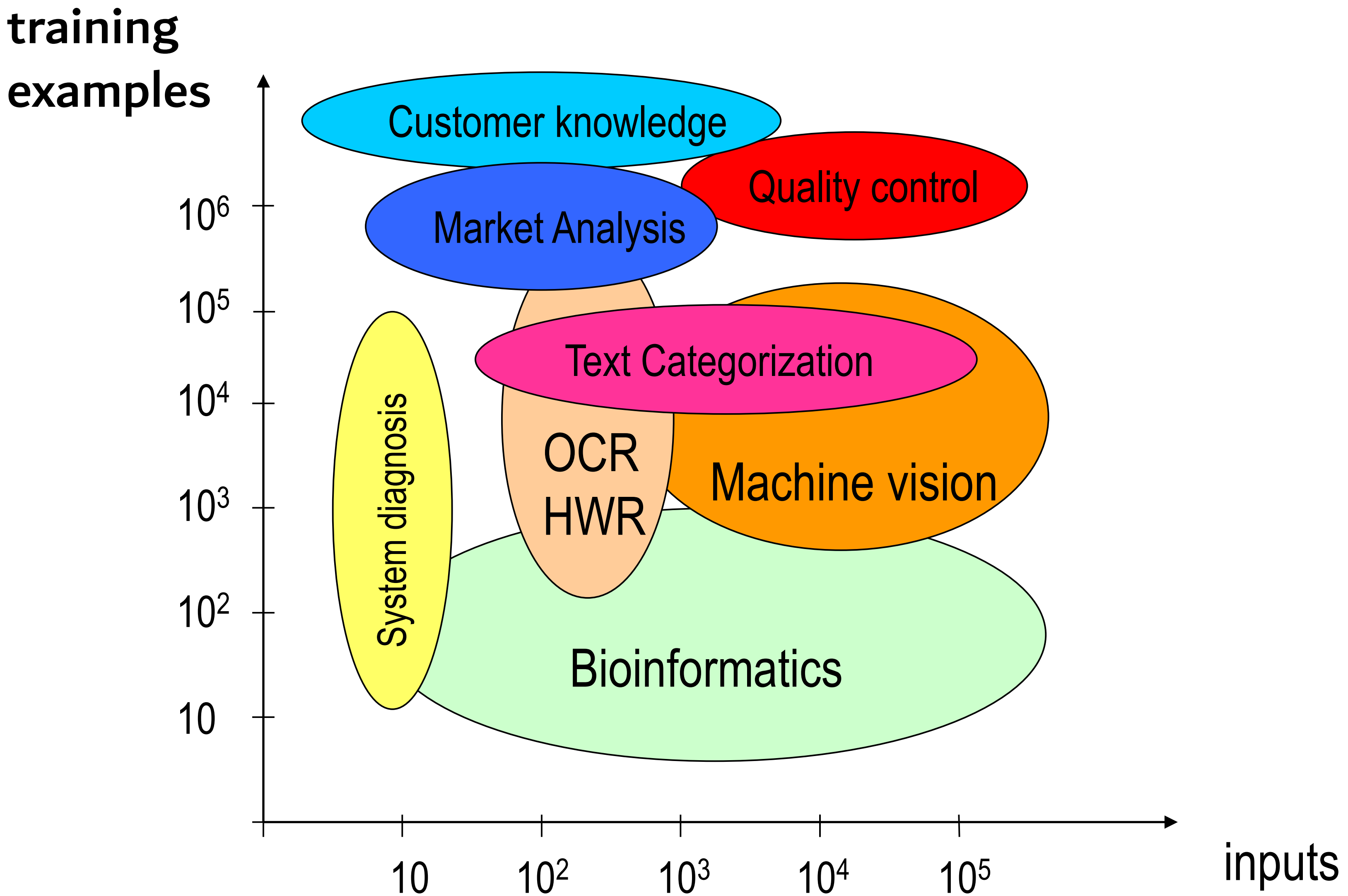
# What is Machine Learning?



# For which tasks ?

- **Classification** (binary/categorical target)
- **Regression** and **time series prediction** (continuous targets)
- **Clustering** (targets unknown)
- **Rule discovery**

# For which applications ?



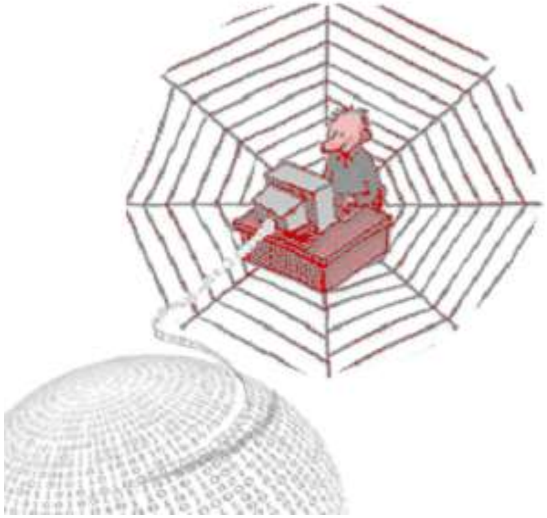
# Banking / Telecom / Retail

- **Identify:**
  - **Prospective customers**
  - **Dissatisfied customers**
  - **Good customers**
  - **Bad payers**
- **Obtain:**
  - **More effective advertising**
  - **Less credit risk**
  - **Fewer fraud**
  - **Decreased churn rate**

# Biomedical / Biometrics

- **Medicine:**
  - Screening
  - Diagnosis and prognosis
  - Drug discovery
- **Security:**
  - Face recognition
  - Signature / fingerprint / iris verification
  - DNA fingerprinting

# Computer / Internet



- Computer interfaces:
  - Troubleshooting wizards
  - Handwriting and speech
  - Brain waves
- Internet
  - Hit ranking
  - Spam filtering
  - Text categorization
  - Text translation
  - Recommendation

# ML in a Nutshell

- Tens of thousands of machine learning algorithms
- Hundreds new every year
- Every machine learning algorithm has three components:
  - **Representation**
  - **Evaluation**
  - **Optimization**

# Representation

- Decision trees
- Sets of rules / Logic programs
- Instances
- Graphical models (Bayes/Markov nets)
- Neural networks
- Support vector machines
- Model ensembles
- Etc.

# Evaluation

- Accuracy
- Precision and recall
- Squared error
- Likelihood
- Posterior probability
- Cost / Utility
- Margin
- Entropy
- K-L divergence
- Etc.

# Optimization

- Combinatorial optimization
  - E.g.: Greedy search
- Convex optimization
  - E.g.: Gradient descent
- Constrained optimization
  - E.g.: Linear programming

# Types of Learning

- **Supervised (inductive) learning**
  - Training data includes desired outputs
- **Unsupervised learning**
  - Training data does not include desired outputs
- **Semi-supervised learning**
  - Training data includes a few desired outputs
- **Reinforcement learning**
  - Rewards from sequence of actions

# Supervised Learning

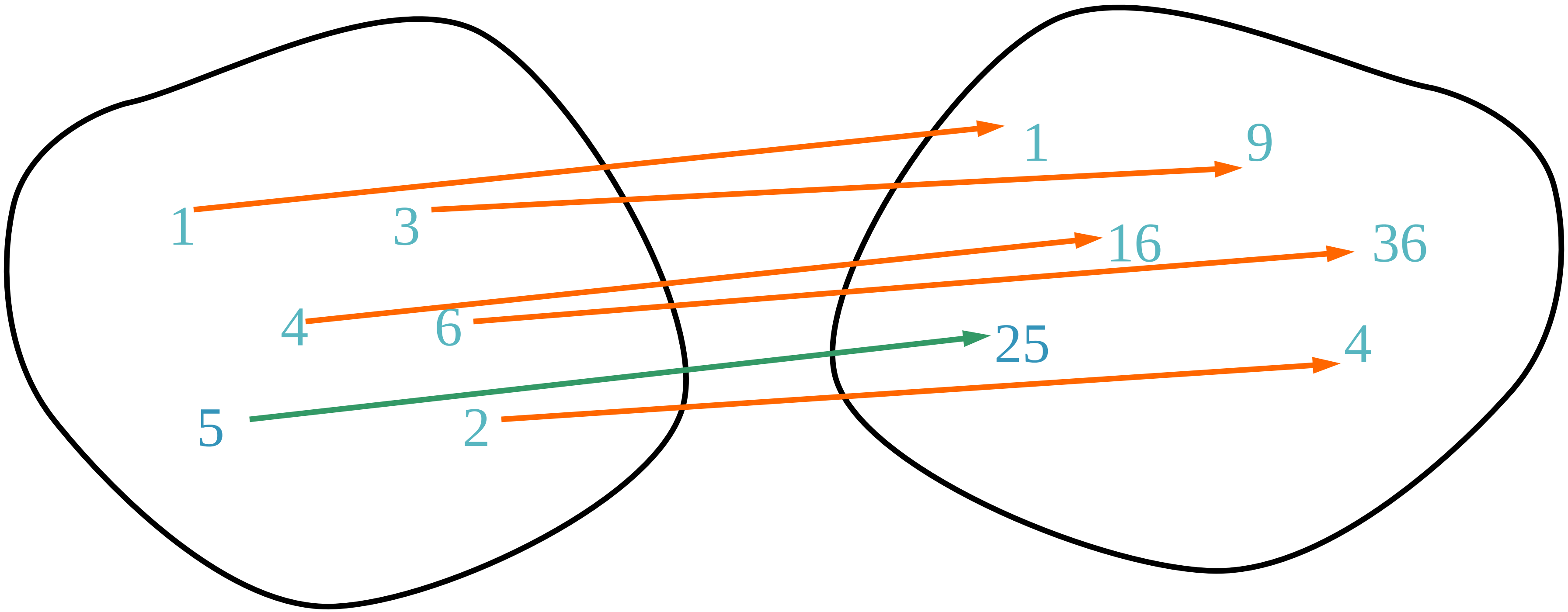
Learning Through Examples

# Supervised Learning

- When a set of targets of interest is provided by an external teacher

  we say that the learning is **Supervised**
- The targets usually are in the form of an **input output mapping** that the net should learn

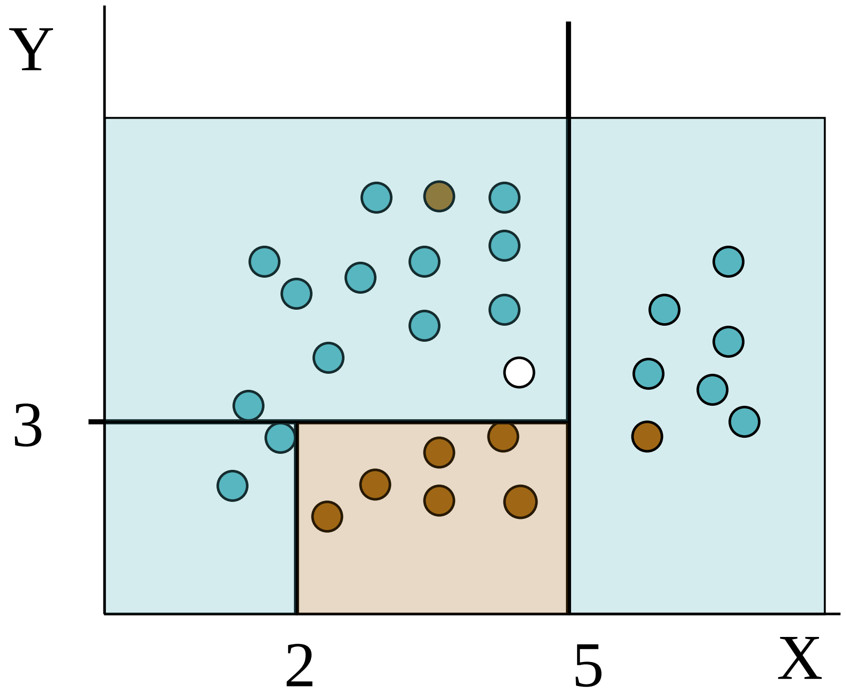
# Learning From Examples



# What We'll Cover

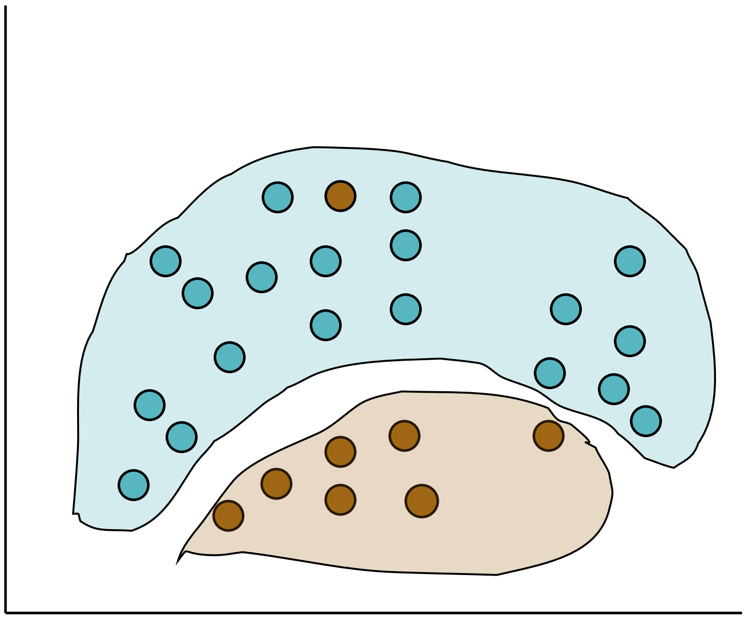
- **Supervised learning**
  - Decision tree induction
  - Neural networks
  - Rule induction
  - Instance-based learning
  - Bayesian learning
  - Support vector machines
  - Model ensembles
  - Learning theory

# Classification: Decision Trees




if X > 5 then blue
else if Y > 3 then blue
else if X > 2 then green
else blue

# Classification: Neural Nets



- Can select more complex regions
- Can be more accurate
- Also, can overfit the data – find patterns in random noise

# Decision Tree Learning

Learning Through Examples

# Choosing the Best Attribute

- The key problem is choosing which attribute to split a given set of examples.
- Some possibilities are:
  - **Random:** Select any attribute at random
  - **Least-Values:** Choose the attribute with the smallest number of possible values (fewer branches)
  - **Most-Values:** Choose the attribute with the largest number of possible values (smaller subsets)
  - **Max-Gain:** Choose the attribute that has the largest expected information gain, i.e. select attribute that will result in the smallest expected size of the subtrees rooted at its children.
- The ID3 algorithm uses the Max-Gain method of selecting the best attribute.

- ID3 (Iterative Dichotomiser 3) Algorithm
  - Top-down, greedy search through space of possible decision trees
    - Remember, decision trees represent hypotheses, so this is a search through hypothesis space.
  - What is top-down?
    - How to start tree?
      - What attribute should represent the root?
    - As you proceed down tree, choose attribute for each successive node.
    - ***No backtracking***:
      - So, algorithm proceeds from top to bottom

**Question?**

How do you determine *which attribute best classifies data*?

**Answer:** **Entropy!**

- *Entropy is a measure of disorder or impurity*
- ***Information gain***:
  - Measures the expected reduction in entropy caused by partitioning
  - Statistical quantity measuring how well an attribute classifies the data.
    - Calculate the information gain for each attribute.
    - Choose attribute with greatest information gain.

# Information Theory Background

- If there are *n* equally probable possible messages, then the probability *p* of each is 1/*n*
- Information conveyed by a message is -log(*p*) = log(*n*)
- Eg, if there are 16 messages, then log(16) = 4 and we need 4 bits to identify/send each message.
- In general, if we are given a probability distribution

  $P = (p_1, p_2, .., p_n)$
- the information conveyed by distribution (aka Entropy of P) is:

  $H(P) = -(p_1*\log(p_1) + p_2*\log(p_2) + .. + p_n*\log(p_n))$

# Information Gain

**Information gain** is our metric for how well one attribute $A_i$ classifies the training data.

- Calculate the entropy for all training examples
  - positive and negative cases
  - $p_+$ = #pos/Tot  $p_-$ = #neg/Tot
  - $H(S) = -p_+\log_2(p_+) - p_-\log_2(p_-)$
- Determine which single attribute best classifies the training examples using information gain.
  - For each attribute find:

$$\boxed{Gain(S, A_i) = \underbrace{H(S)}_{\text{entropy}} - \sum_{v \in Values(A_i)} P(A_i = v)\underbrace{H(S_v)}_{\text{Entropy for value v}}}$$

  - Use attribute with greatest information gain as a root
  - Gain (S, A) = expected reduction in entropy due to sorting on A

- **Example:** PlayTennis
  - Four attributes used for classification:
    - Outlook = {Sunny, Overcast, Rain}
    - Temperature = {Hot, Mild, Cool}
    - Humidity = {High, Normal}
    - Wind = {Weak, Strong}
  - One predicted (target) attribute (binary)
    - PlayTennis = {Yes, No}
  - Given 14 Training examples
    - 9 positive
    - 5 negative

Examples,
minterms,
cases, objects,
test cases,

# Training Examples

| Day | Outlook | Temperature | Humidity | Wind | PlayTennis |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D3 | Overcast | Hot | High | Weak | Yes |
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D7 | Overcast | Cool | Normal | Strong | Yes |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D10 | Rain | Mild | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |
| D12 | Overcast | Mild | High | Strong | Yes |
| D13 | Overcast | Hot | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |

| Day | Outlook | Temperature | Humidity | Wind | PlayTennis |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D3 | Overcast | Hot | High | Weak | Yes |
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D7 | Overcast | Cool | Normal | Strong | Yes |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D10 | Rain | Mild | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |
| D12 | Overcast | Mild | High | Strong | Yes |
| D13 | Overcast | Hot | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |

14 cases

9 positive cases

- **Step 1: *Calculate entropy* for all cases:**

$N_{Pos} = 9$ $N_{Neg} = 5$ $N_{Tot} = 14$

$H(S) = -(9/14)*\log_2(9/14) - (5/14)*\log_2(5/14) = 0.940$

entropy

- Step 2: Loop over all attributes, calculate gain:
  - **Attribute = *Outlook***
    - Loop over values of *Outlook*

*Outlook* = Sunny

$N_{Pos} = 2$ $N_{Neg} = 3$ $N_{Tot} = 5$

$H(\text{Sunny}) = -(2/5)*\log_2(2/5) - (3/5)*\log_2(3/5) = 0.971$

*Outlook* = Overcast

$N_{Pos} = 4$ $N_{Neg} = 0$ $N_{Tot} = 4$

$H(\text{Overcast}) = -(4/4)*\log_2 4/4) - (0/4)*\log_2(0/4) = 0.00$

| Day | Outlook | Temperature | Humidity | Wind | PlayTennis |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D3 | Overcast | Hot | High | Weak | Yes |
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D7 | Overcast | Cool | Normal | Strong | Yes |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D10 | Rain | Mild | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |
| D12 | Overcast | Mild | High | Strong | Yes |
| D13 | Overcast | Hot | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |

*Outlook* = Rain

$N_{Pos} = 3$ $N_{Neg} = 2$ $N_{Tot} = 5$

$H(Rain) = -(3/5)*\log_2(3/5) - (2/5)*\log_2(2/5) = 0.971$

- Calculate **Information Gain** for attribute Outlook

$$Gain(S, Outlook) = H(S) - N_{Sunny}/N_{Tot}*H(Sunny) - N_{Over}/N_{Tot}*H(Overcast) - N_{Rain}/N_{Tot}*H(Rain)$$

$$Gain(S, Outlook) = 0.940 - (5/14)*0.971 - (4/14)*0 - (5/14)*0.971$$

$Gain(S, Outlook) = 0.246$

| Day | Outlook | Temperature | Humidity | Wind | PlayTennis |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D3 | Overcast | Hot | High | Weak | Yes |
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D7 | Overcast | Cool | Normal | Strong | Yes |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D10 | Rain | Mild | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |
| D12 | Overcast | Mild | High | Strong | Yes |
| D13 | Overcast | Hot | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |

- **Attribute = *Temperature***
  - (Repeat process looping over {Hot, Mild, Cool})

    Gain(*S*, *Temperature*) = 0.029
- **Attribute = *Humidity***
  - (Repeat process looping over {High, Normal})

    Gain(*S*, *Humidity*) = 0.029
- **Attribute = *Wind***
  - (Repeat process looping over {Weak, Strong})

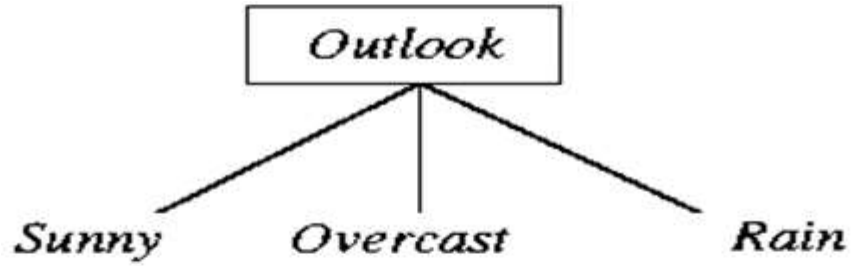
    Gain(*S*, *Wind*) = 0.048

## Find attribute with greatest information gain:

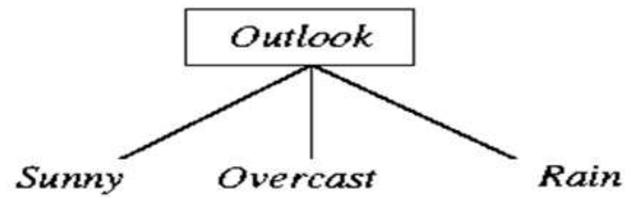
Gain(*S*,*Outlook*) = 0.246, Gain(*S*,*Temperature*) = 0.029

Gain(*S*,*Humidity*) = 0.029, Gain(*S*,*Wind*) = 0.048

## $\therefore$ *Outlook is root node of tree*



- **Iterate algorithm to find attributes which best classify training examples under the values of the root node**
- **Example continued**
  - **Take three subsets:**
    - ***Outlook* = Sunny** $(N_{Tot} = 5)$
    - ***Outlook* = Overcast** $(N_{Tot} = 4)$
    - ***Outlook* = Rainy** $(N_{Tot} = 5)$
  - **For each subset, repeat the above calculation looping over all attributes other than *Outlook***




| Day | Outlook | Temperature | Humidity | Wind | PlayTennis |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |

| Day | Outlook | Temperature | Humidity | Wind | PlayTennis |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D3 | Overcast | Hot | High | Weak | Yes |
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D7 | Overcast | Cool | Normal | Strong | Yes |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D10 | Rain | Mild | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |
| D12 | Overcast | Mild | High | Strong | Yes |
| D13 | Overcast | Hot | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |

- For example:
  - *Outlook* = Sunny ($N_{Pos}$ = 2, $N_{Neg}$=3, $N_{Tot}$ = 5) H=0.971
    - *Temp* = Hot ($N_{Pos}$ = 0, $N_{Neg}$=2, $N_{Tot}$ = 2) H = 0.0
    - *Temp* = Mild ($N_{Pos}$ = 1, $N_{Neg}$=1, $N_{Tot}$ = 2) H = 1.0
    - *Temp* = Cool ($N_{Pos}$ = 1, $N_{Neg}$=0, $N_{Tot}$ = 1) H = 0.0

  Gain($S_{Sunny}$, *Temperature*) = 0.971 - (2/5)*0 - (2/5)*1 - (1/5)*0

  Gain($S_{Sunny}$, *Temperature*) = 0.571

  Similarly:

  Gain($S_{Sunny}$, *Humidity*) = 0.971

  Gain($S_{Sunny}$, *Wind*) = 0.020

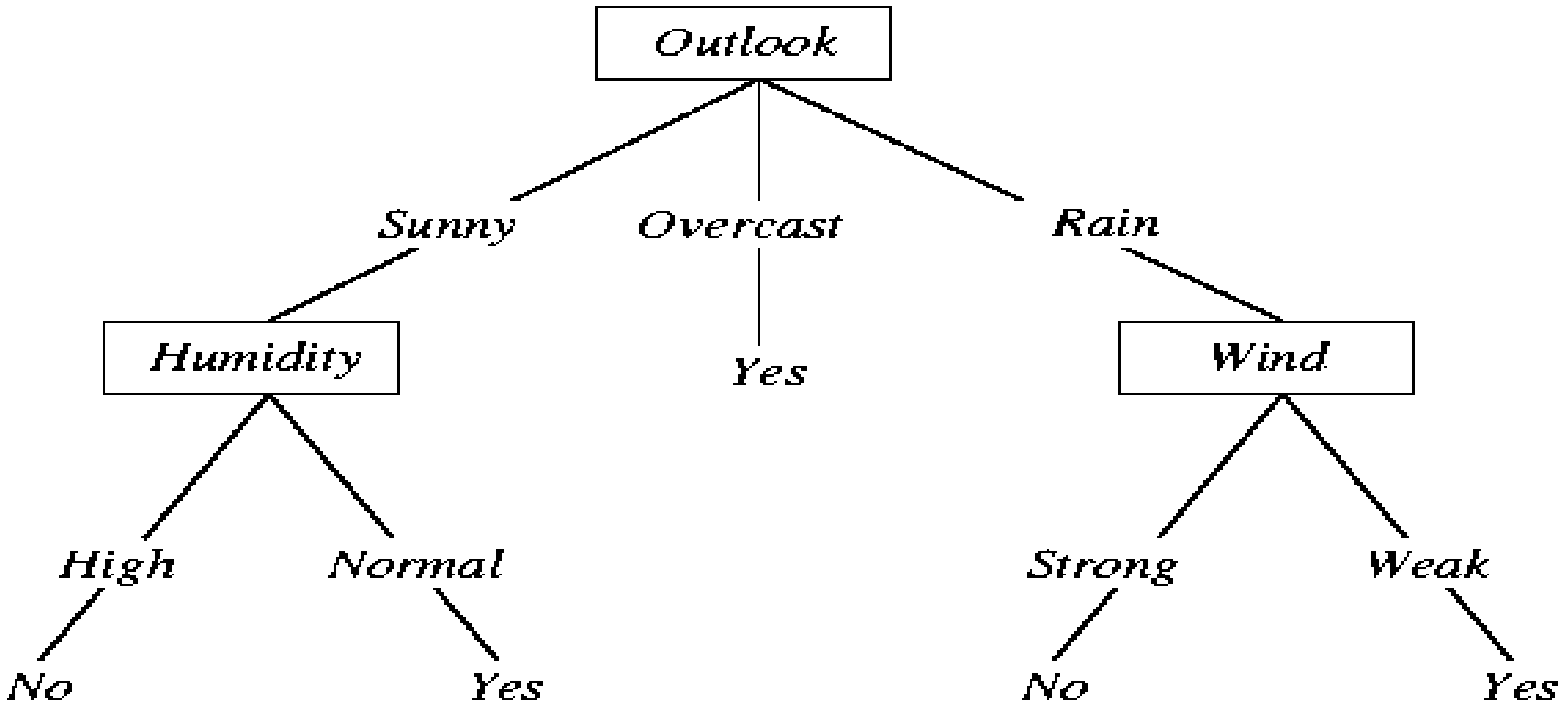
| Day | Outlook | Temperature | Humidity | Wind | PlayTennis |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |

∴ Humidity classifies *Outlook* = Sunny instances best and is placed as the node under Sunny outcome.

- Repeat this process for *Outlook* = Overcast & Rainy

- **End up with tree:**

# Decision Tree for *PlayTennis*



- **Important:**
  - <u>Attributes are excluded</u> from consideration if they appear higher in the tree
- Process <u>continues for each new leaf node</u> until:
  - Every attribute <u>has already been included</u> along path through the tree

  *or*

  - Training examples associated with this leaf <u>all have same target attribute value.</u>

- **Note:** In this example data were perfect.
  - No contradictions
  - Branches led to unambiguous *Yes, No* decisions
  - If there are contradictions take the majority vote
    - This handles noisy data.
- **Another note:**
  - Attributes are eliminated when they are assigned to a node and never reconsidered.
    - e.g. You would not go back and reconsider *Outlook* under *Humidity*
- **ID3 uses all of the training data at once**
  - Contrast to Candidate-Elimination
  - Can handle noisy data.

# About Entropy and Information Gain

Decision Tree Learning

# Entropy

"characterizes the impurity of an arbitrary collection of examples" [Mitchell,1997]

- So if the impurity or randomness of a collection (with respect to the target classifier) is high then the entropy is high
- But if there is no randomness (complete uniformity with respect to the target classifier) then the entropy is zero

$$\textbf{Entropy(S)} \equiv -p_{+}\log_2 p_{+} - p_{-}\log_2 p_{-}$$

Where target classification is boolean

$p_{+}$ :the proportion of positive examples in collection S

$p_{-}$ :the proportion of negative examples in collection S

# Entropy Example 1…

$$Entropy(S) \equiv -p_{+}\log_2 p_{+} - p_{-}\log_2 p_{-}$$

Entropy([2+,0-]) $= -(2/2)\log_2(2/2) - (0/2)\log_2(0/2)$
$= -1\log_2(1) - (0/2)\log 2(0/2)$
$= -1{*}0 - 0{*}0$
$= 0 - 0$
$= 0$

**Recall that $2^0 = 1$**

| Outlook | Wind | WearCoat |
|---|---|---|
| Rain | Weak | Yes |
| Sunny | Strong | Yes |

(Max Uniformity, Min Randomness)

# Entropy Example 2…

$$\text{Entropy}(S) \equiv -p_+\log_2 p_+ - p_-\log_2 p_-$$

$$
\begin{aligned}
\text{Entropy}([1+,1-]) &= -(1/2)\log_2(1/2) - (1/2)\log_2(1/2) \\
&= -\,0.5 * \log_2(0.5) - 0.5 * \log_2(0.5) \\
&= (-0.5 * -1) - (0{\cdot}5 * -1) \\
&= 0.5 - (-0.5) \\
&= 1
\end{aligned}
$$

| Outlook | Wind | WearCoat |
|---|---|---|
| Rain | Weak | Yes |
| Sunny | Weak | No |

(Max Randomness)

# Entropy Example 3...

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

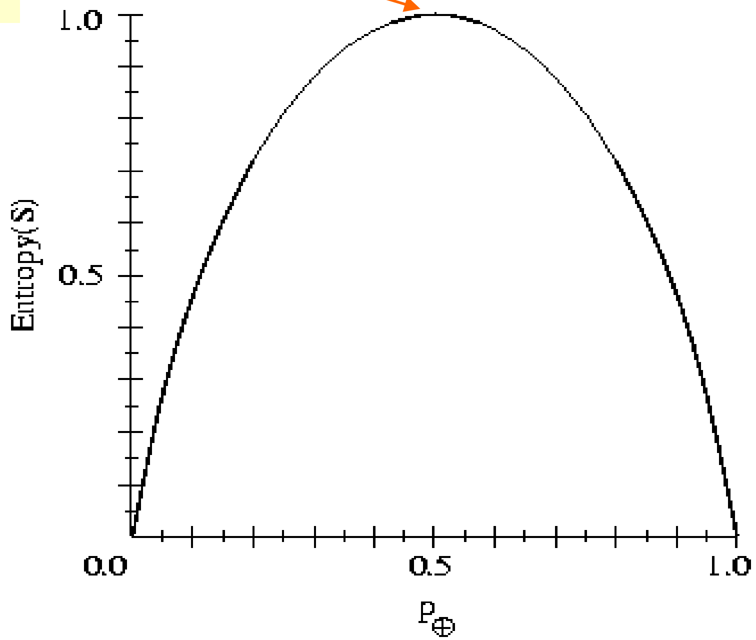
$$\begin{aligned} Entropy([2+,1-]) &= -(2/3)\log_2(2/3) - (1/3)\log_2(1/3) \\ &= -0.67 * \log_2(0.67) - (0.33) * \log_2(0.33) \\ &= -(0.67 * -0.58) - (0.33 * -1.6) \\ &= 0.39 - (-0.53) \\ &= 0.92 \end{aligned}$$

N.B. For reasons of speed/brevity, I'm just working to 2 decimal places!

| Outlook | Wind | WearCoat |
|---|---|---|
| Rain | Weak | Yes |
| Sunny | Strong | Yes |
| Sunny | Weak | No |

# Entropy

Largest entropy




- $S$ is a sample of training examples
- $p_{\oplus}$ is the proportion of positive examples in $S$
- $p_{\ominus}$ is the proportion of negative examples in $S$
- Entropy measures the impurity of $S$

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

Boolean functions with the same number of ones and zeros have largest entropy

# Information Gain

- **Measures the expected reduction in entropy caused by partitioning**
- **Statistical quantity measuring how well an attribute classifies the data.**

$$\boxed{Gain(S, A_i) = \underbrace{H(S)}_{\text{entropy}} - \sum_{v \in Values(A_i)} P(A_i = v)\underbrace{H(S_v)}_{\text{Entropy for value v}}}$$

- **Maximum gain means minimum entropy for attribute Ai, meaning Ai has less randomness in classifying**

# Extensions of Decision Tree Learning

# Extensions of the Decision Tree Learning

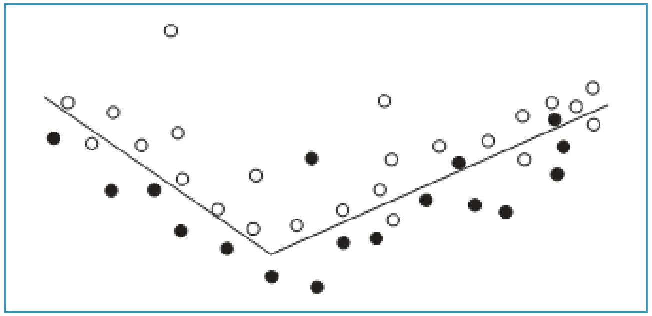
- Noisy data and Overfitting
- Cross-Validation for Experimental Validation of Performance
- Pruning Decision Trees
- Real-valued data
- Using gain ratios
- Generation of rules
- Setting Parameters
- Incremental learning

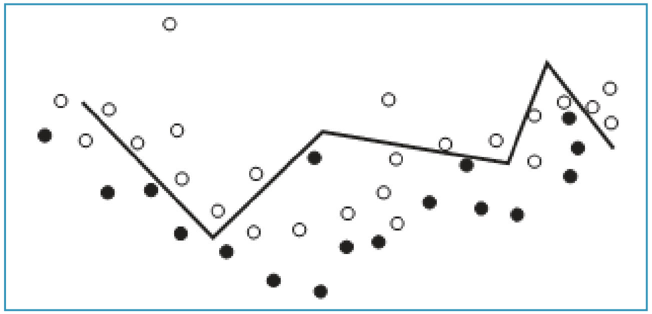
# Noisy data and Overfitting

- Many kinds of "noise" that could occur in the examples:
  - Two examples have same attribute/value pairs, but different classifications
  - Some values of attributes are incorrect because of:
    - Errors in the data acquisition process
    - Errors in the preprocessing phase
  - The classification is wrong (e.g., + instead of -) because of some error
- Some attributes are irrelevant to the decision-making process,
  - e.g., color of a die is irrelevant to its outcome.
  - Irrelevant attributes can result in overfitting the training data.

# Noisy data and Overfitting

- Black dots are positive, others negative
- Two lines represent two hypothesis
- Thick line is complex hypothesis correctly classifies all data
- Thin line is simple hypothesis but incorrectly classifies some data
- The simple hypothesis makes some errors but reasonably closely represents the trend in the data
- The complex solution does not at all represent the full set of data





- Fix overfitting /overlearning problem
  - By cross validation
  - By pruning lower nodes in the decision tree.

# Cross Validation: An Evaluation Methodology

- Standard methodology: cross validation
  1. Collect a large set of examples (all with correct classifications!).
  2. Randomly divide collection into two disjoint sets: training and test.
  3. Apply learning algorithm to training set giving hypothesis H
  4. Measure performance of H w.r.t. test set
- Important: keep the training and test sets disjoint!
- Learning is not to minimize training error (wrt data) but the error for test/cross-validation: a way to fix overfitting
- To study the efficiency and robustness of an algorithm, repeat steps 2-4 for different training sets and sizes of training sets.
- If you improve your algorithm, start again with step 1 to avoid evolving the algorithm to work well on just this collection.

# Pruning Decision Trees

- Pre Pruning: Stop growing before a fully grown tree
- Post Pruning : Trim fully grown tree from the bottom
  - Reduced Error Pruning
  - Rule post pruning

# Real-valued data

- Select a set of thresholds defining intervals;
  - each interval becomes a discrete value of the attribute
- We can use some simple heuristics
  - always divide into quartiles
- We can use domain knowledge
  - divide age into infant (0-2), toddler (3 - 5), and school aged (5-8)
- or treat this as another learning problem
  - try a range of ways to discretize the continuous variable
  - Find out which yield "better results" with respect to some metric.

# Performance Evaluation

Decision Tree Learning

- Focus on the predictive capability of a model
  - Rather than how fast it takes to classify or build models, scalability, etc.
- Confusion Matrix:

| | PREDICTED CLASS | | |
|---|---|---|---|
| ACTUAL CLASS | | Class=Yes | Class=No |
| | Class=Yes | TP | FN |
| | Class=No | FP | TN |

TP (true positive)

FN (false negative)

FP (false positive)

TN (true negative)

- TP: predicted to be in YES, and is actually in it
- FP: predicted to be in YES, but is not actually in it
- TN: predicted not to be in YES, and is not actually in it
- FN: predicted not to be in YES, but is actually in it

# Accuracy

| | PREDICTED CLASS | |
|---|---|---|---|
| | | Class=Yes | Class=No |
| ACTUAL CLASS | Class=Yes | TP | FN |
| | Class=No | FP | TN |

- **Most widely-used metric:**

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

# Class imbalance problem

- Consider a 2-class problem
  - Number of Class 0 examples = 9990
  - Number of Class 1 examples = 10

- If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9 %
  - Accuracy is misleading because model does not detect any class 1 example

# Classifier Evaluation Metrics: Accuracy, Error Rate, Sensitivity and Specificity

| A\P | Yes | No | |
|---|---|---|---|
| Yes | **TP** | **FN** | **P** |
| No | **FP** | **TN** | **N** |
| | **P'** | **N'** | **All** |

- **Classifier Accuracy**, or recognition rate: percentage of test set tuples that are correctly classified

  **Accuracy = (TP + TN)/All**

- **Error rate:** *1 – accuracy*, or

  **Error rate = (FP + FN)/All**

- **Sensitivity**: True Positive recognition rate
  - **Sensitivity = TP/P**
- **Specificity**: True Negative recognition rate
  - **Specificity = TN/N**

# Classifier Evaluation Metrics: Precision and Recall, and F-measures

- **Precision:** exactness – what % of tuples that the classifier labeled as positive are actually positive

$$\text{precision} = \frac{TP}{TP + FP}$$

- **Recall:** completeness – what % of positive tuples did the classifier label as positive?
  - Perfect score is 1.0

$$\text{recall} = \frac{TP}{TP + FN}$$

- ***F*-measure (*$F_1$* score or *F*-score)**
  - harmonic mean of precision and recall,

$$\mathrm{F} = \frac{2 \times precision \times recall}{precision + recall}$$

- Precision is biased towards TP & FP
- Recall is biased towards TP & FN
- F-measure is biased towards all except TN

# Classifier Evaluation Metrics: Matthews correlation coefficient (MCC)

- MCC takes into account true and false positives and negatives.

- Generally regarded as a balanced measure which can be used even if the classes are of very different sizes.

- It returns a value between −1 and +1.
  - 1 represents a perfect prediction
  - 0 no better than random prediction
  - −1 indicates total disagreement between prediction and observation

# Classifier Evaluation Metrics: Matthews correlation coefficient (MCC)

$$N = TN + TP + FN + FP$$

$$S = \frac{TP + FN}{N}$$

$$P = \frac{TP + FP}{N}$$

$$MCC = \frac{TP / N - S \times P}{PS(1 - S)(1 - P)}$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

# References

- Chapter 18 of “Artificial Intelligence: A modern approach” by Stuart Russell, Peter Norvig.
- Chapter 10 of “AI Illuminated” by Ben Coppin.