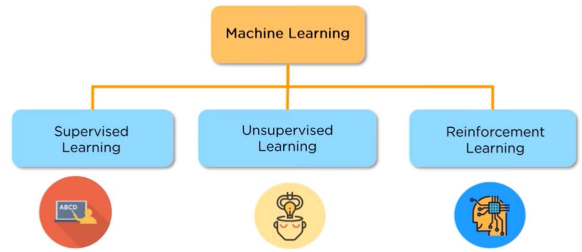
Classification Techniques in Machine Learning

Dr. Shailesh Sivan

What is Machine learning?

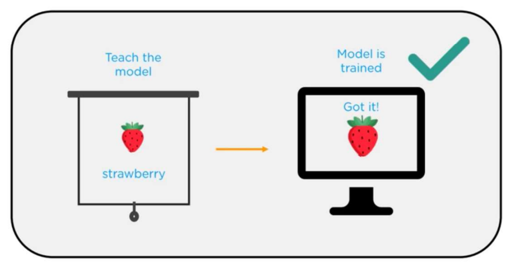


Support Vector Machine

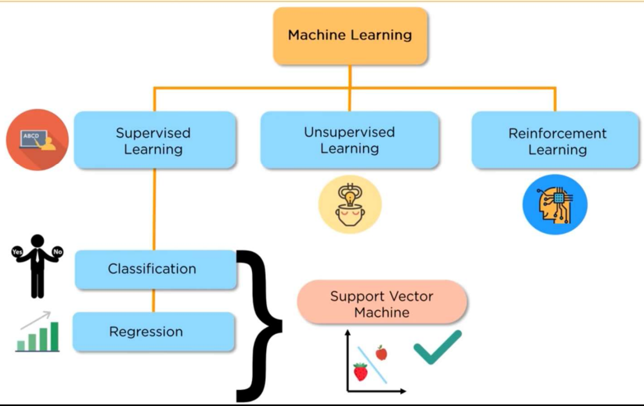
What is Machine learning?

Supervised Learning

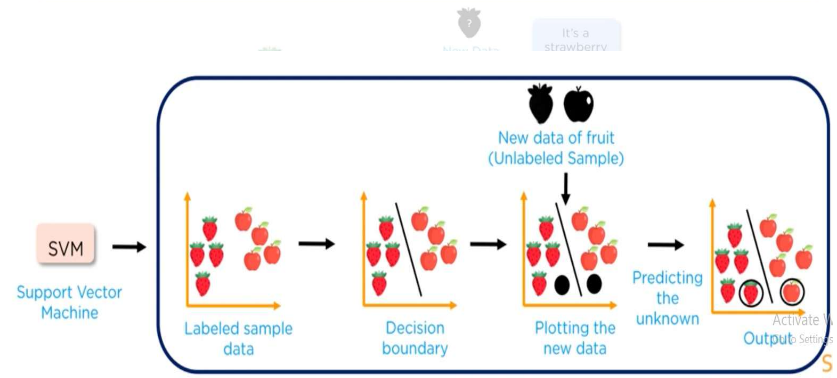
Machine learning model learns from the past input data and makes future prediction as output



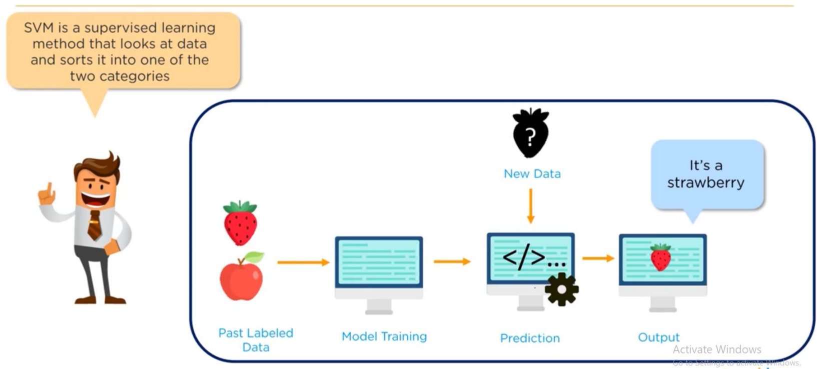
What is Machine learning?



Why Support Vector Machine?



Why Support Vector Machine?



Sample data set

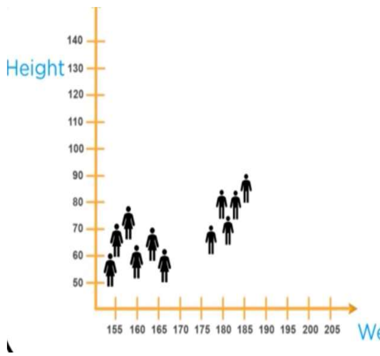
Female

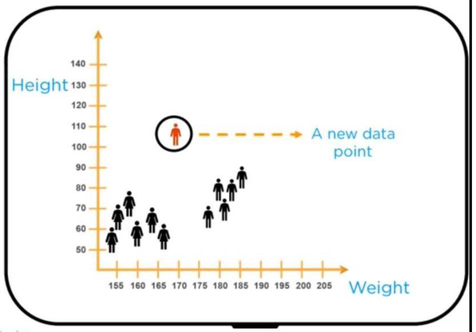
Height	Weight
174	65
174	88
175	75
180	65
185	80

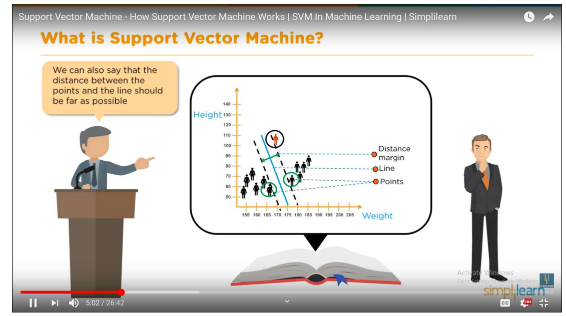
Sample data set

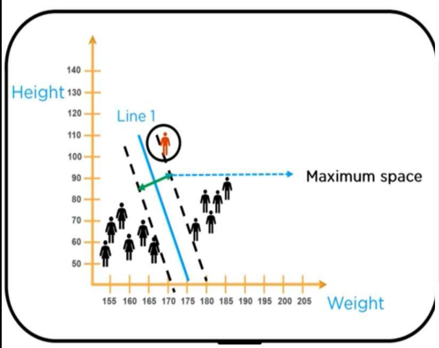
Male

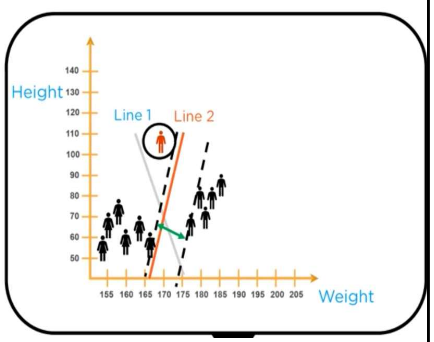
Height	Weight
179	90
180	80
183	80
187	85
182	72



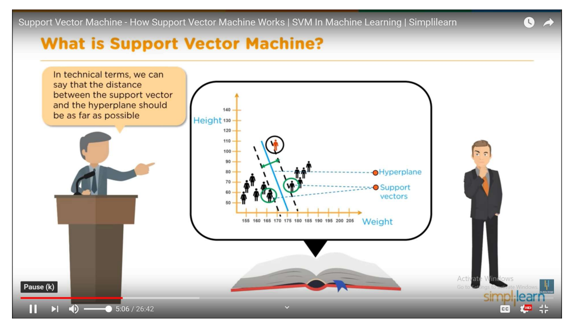




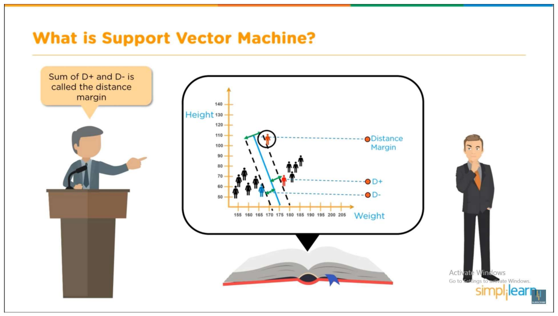




Optimal hyper plane

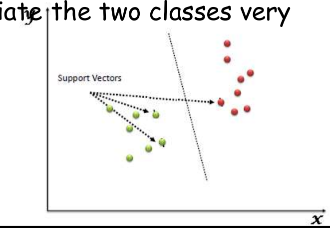


Optimal hyper plane

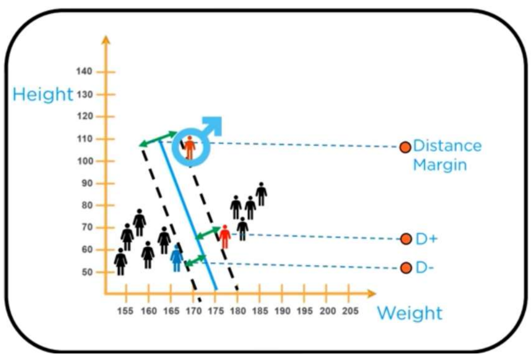


Support Vector Machine

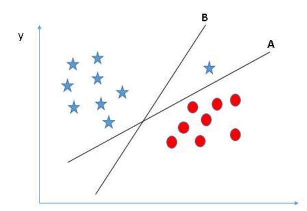
- "Support Vector Machine" (SVM) is a supervised [machine learning algorithm]() which can be used for both classification or regression challenges
- perform classification by finding the hyper-plane that differentiate the two classes very well



Predict the New Data Item

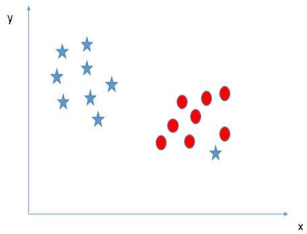


Identify the right hyper-plane (Scenario-3)



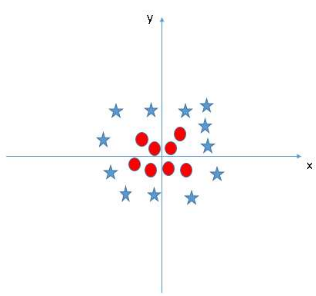
- Some of you may have selected the hyper-plane **B** as it has higher margin compared to ***A***

Can we classify two classes (Scenario-4)



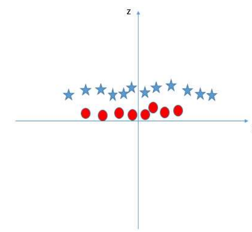
- outlier for star class

Find the hyper-plane to segregate to classes (Scenario-5)



- SVM can solve this problem. Easily! It solves this problem by introducing additional feature.

kernel trick.

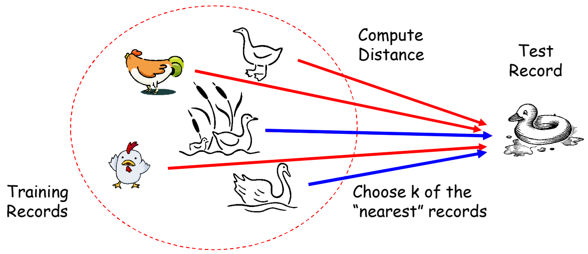


- Here, we will add a new feature $z = x^2 + y^2$.
- All values for z would be positive always because z is the squared sum of both x and y
- In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z.

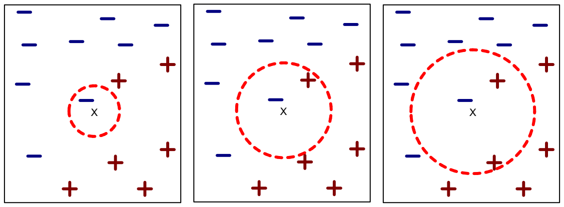
K Nearest Neighbour

Nearest Neighbor Classifiers

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



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

Basic Idea

- *k*-NN classification rule is to assign to a test sample the majority category label of its *k* nearest training samples
- In practice, *k* is usually chosen to be odd, so as to avoid ties
- The *k* = 1 rule is generally called the nearest-neighbor classification rule

Distance-weighted *k*-NN

Replace

$$\hat{f}(q) = \arg\max_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$$

by:

$$\hat{f}(q) = \arg\max_{v \in V} \sum_{i=1}^{k} \frac{1}{d\left(x_i, x_q\right)^2} \delta(v, f(x_i))$$

General Kernel functions like Parzen Windows may be considered
Instead of inverse distance.

Predicting Continuous Values

Replace

$$\hat{f}(q) = \arg\max_{v \in V} \sum_{i=1}^{k} w_i \delta(v, f(x_i))$$

by:

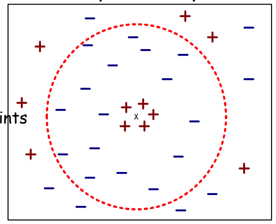
$$\hat{f}(q) = \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

- Note: unweighted corresponds to w_i=1 for all i

Value of K

- 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

Rule of thumb:
K = sqrt(N)
N: number of training points



Nearest-Neighbor Classifiers: Issues

- The value of k, the number of nearest neighbors to retrieve
- Choice of Distance Metric to compute distance between records
- Computational complexity
  - Size of training set
  - Dimension of data

Distance Metrics

Minkowsky:

$$D(\mathbf{x},\mathbf{y}) = \left(\sum_{i=1}^{m} |x_i - y_i|^r\right)^{1/r}$$

Euclidean:

$$D(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}$$

Manhattan / city-block:

$$D(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{m} |x_i - y_i|$$

Camberra:

$$D(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{m} \frac{|x_i - y_i|}{|x_i + y_i|}$$

Chebychev:

$$D(\mathbf{x},\mathbf{y}) = \max_{i=1}^{m} |x_i - y_i|$$

Quadratic:

$$D(\mathbf{x},\mathbf{y}) = (\mathbf{x}-\mathbf{y})^T Q (\mathbf{x}-\mathbf{y}) = \sum_{j=1}^{m}\left(\sum_{i=1}^{m} (x_i - y_i) q_{ji}\right)(x_j - y_j)$$

Q is a problem-specific positive definite $m \times m$ weight matrix

Mahalanobis:

$$D(\mathbf{x},\mathbf{y}) = [\det V]^{1/m} (\mathbf{x}-\mathbf{y})^{\mathrm{T}} V^{-1} (\mathbf{x}-\mathbf{y})$$

V is the covariance matrix of $A_1..A_m$, and A_j is the vector of values for attribute j occuring in the training set instances $1..n$.

Correlation:

$$D(\mathbf{x},\mathbf{y}) = \frac{\sum_{i=1}^{m} (x_i - \overline{x_i})(y_i - \overline{y_i})}{\sqrt{\sum_{i=1}^{m} (x_i - \overline{x_i})^2 \sum_{i=1}^{m} (y_i - \overline{y_i})^2}}$$

$\overline{x_i} = \overline{y_i}$ and is the average value for attribute i occuring in the training set.

Chi-square:

$$D(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{m} \frac{1}{sum_i}\left(\frac{x_i}{size_{\mathbf{x}}} - \frac{y_i}{size_{\mathbf{y}}}\right)^2$$

sum_i is the sum of all values for attribute i occuring in the training set, and $size_{\mathbf{x}}$ is the sum of all values in the vector $\mathbf{x}$.

Kendall's Rank Correlation:

$$D(\mathbf{x},\mathbf{y}) = 1 - \frac{2}{n(n-1)} \sum_{i=1}^{m} \sum_{j=1}^{i-1} \operatorname{sign}(x_i - x_j)\operatorname{sign}(y_i - y_j)$$

sign(x)=-1, 0 or 1 if $x < 0$, $x = 0$, or $x > 0$, respectively.

Figure 1. Equations of selected distance functions.
($\mathbf{x}$ and $\mathbf{y}$ are vectors of m attribute values).

Naïve Bayes

Naïve Bayes

- Bayes classification

$$P(C|\mathbf{X}) \propto P(\mathbf{X}|C)P(C) = P(X_1,\cdots,X_n|C)P(C)$$

Difficulty: learning the joint probability $P(X_1,\cdots,X_n|C)$

- Naïve Bayes classification
  - Making the assumption that all input attributes are independent

$$\begin{aligned} P(X_1,X_2,\cdots,X_n|C) &= P(X_1|X_2,\cdots,X_n;C)P(X_2,\cdots,X_n|C) \\ &= P(X_1|C)P(X_2,\cdots,X_n|C) \\ &= P(X_1|C)P(X_2|C)\cdots P(X_n|C) \end{aligned}$$

  - MAP classification rule

$$[P(x_1|c^*)\cdots P(x_n|c^*)]P(c^*) > [P(x_1|c)\cdots P(x_n|c)]P(c), \quad c \neq c^*, c = c_1,\cdots,c_L$$

Naïve Bayes

- Naïve Bayes Algorithm (for discrete input attributes)
  - Learning Phase: Given a training set S,

For each target value of c_i ($c_i = c_1,\cdots,c_L$)

$\hat{P}(C=c_i) \leftarrow$ estimate $P(C=c_i)$ with examples in **S**;

For every attribute value a_{jk} of each attribute x_j ($j=1,\cdots,n;\ k=1,\cdots,N_j$)

$\hat{P}(X_j=a_{jk}|C=c_i) \leftarrow$ estimate $P(X_j=a_{jk}|C=c_i)$ with examples in **S**;

Output: conditional probability tables; for x_j, $N_j \times L$ elements

  - Test Phase: Given an unknown instance $\mathbf{X}' = (a'_1,\cdots,a'_n)$,

Look up tables to assign the label c^* to **X'** if

$$[\hat{P}(a'_1|c^*)\cdots\hat{P}(a'_n|c^*)]\hat{P}(c^*) > [\hat{P}(a'_1|c)\cdots\hat{P}(a'_n|c)]\hat{P}(c), \quad c \neq c^*, c = c_1,\cdots,c_L$$

Example

- Example: Play Tennis

PlayTennis: 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

Example

- Learning Phase

Outlook	Play=*Yes*	Play=*No*
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Temperature	Play=*Yes*	Play=*No*
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Humidity	Play=*Yes*	Play=*No*
High	3/9	4/5
Normal	6/9	1/5

Wind	Play=*Yes*	Play=*No*
Strong	3/9	3/5
Weak	6/9	2/5

P(Play=*Yes*) = 9/14 P(Play=*No*) = 5/14

Decision Trees

Example

- Test Phase
  - Given a new instance,
    x′=(Outlook=*Sunny*, Temperature=*Cool*, Humidity=*High*, Wind=*Strong*)
  - Look up tables

P(Outlook=*Sunny*|Play=*Yes*) = 2/9
P(Temperature=*Cool*|Play=*Yes*) = 3/9
P(Huminity=*High*|Play=*Yes*) = 3/9
P(Wind=*Strong*|Play=*Yes*) = 3/9
P(Play=*Yes*) = 9/14

P(Outlook=*Sunny*|Play=*No*) = 3/5
P(Temperature=*Cool*|Play==*No*) = 1/5
P(Huminity=*High*|Play=*No*) = 4/5
P(Wind=*Strong*|Play=*No*) = 3/5
P(Play=*No*) = 5/14

  - MAP rule

P(*Yes*|**x**′): [P(*Sunny*|Yes)P(*Cool*|Yes)P(*High*|Yes)P(*Strong*|Yes)]P(Play=*Yes*) = 0.0053
P(*No*|**x**′): [P(*Sunny*|No) P(*Cool*|No)P(*High*|No)P(*Strong*|No)]P(Play=*No*) = 0.0206

Given the fact P(*Yes*|**x**′) < P(*No*|**x**′), we label **x**′ to be "*No*".

Outline

- Decision Tree Representations
  - ID3 and C4.5 learning algorithms (Quinlan 1986)
  - CART learning algorithm (Breiman et al. 1985)
- Entropy, Information Gain
- Overfitting

Training Data Example:

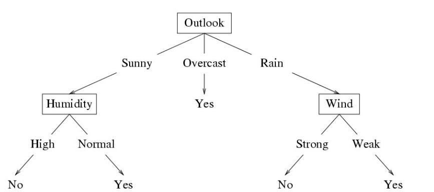
Goal is to Predict When This Player Will Play Tennis?

PlayTennis: 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

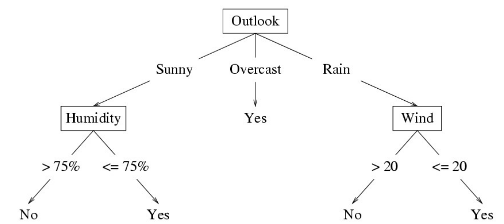
Decision Tree Hypothesis Space

- **Internal nodes** test the value of particular features x_j and branch according to the results of the test.
- **Leaf nodes** specify the class $h(\mathbf{x})$.



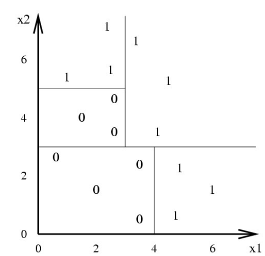
Suppose the features are **Outlook** (x_1), **Temperature** (x_2), **Humidity** (x_3), and **Wind** (x_4). Then the feature vector $\mathbf{x} = (Sunny, Hot, High, Strong)$ will be classified as **No**. The **Temperature** feature is irrelevant.

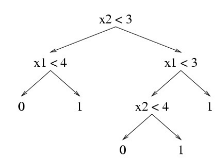
If the features are continuous, internal nodes may test the value of a feature against a threshold.



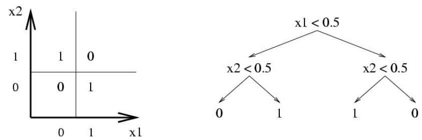
Decision Tree Decision Boundaries

Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.





Decision Trees Can Represent Any Boolean Function



The tree will in the worst case require exponentially many nodes, however.

Learning Algorithm for Decision Trees

$$S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$$

$$\mathbf{x} = (x_1, \ldots, x_d)$$
$$x_j, y \in \{0,1\}$$

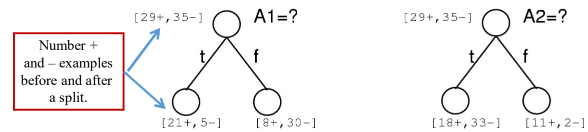
GrowTree(S)
if $(y = 0$ for all $\langle \mathbf{x}, y \rangle \in S)$ **return** new leaf(0)
else if $(y = 1$ for all $\langle \mathbf{x}, y \rangle \in S)$ **return** new leaf(1)
else
  choose best attribute x_j
  $S_0 =$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 0$;
  $S_1 =$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 1$;
  return new node(x_j, GrowTree(S_0), GrowTree(S_1))

What happens if features are not binary? What about regression?

Choosing the Best Attribute

A1 and A2 are "attributes" (i.e. features or inputs).

Which attribute is best?



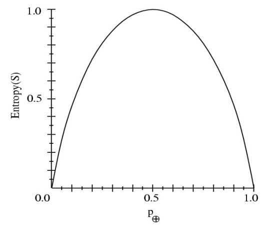
- Many different frameworks for choosing BEST have been proposed!
- We will look at Entropy Gain.

Entropy

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

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

Entropy



- S is a sample of training examples

Entropy is like a measure of impurity…

Entropy

- Entropy measures the randomness/uncertainty in the data
- Let's consider a set S of examples with C many classes. Entropy of this set:

$$H(S) = -\sum_{c \in C} p_c \log_2 p_c$$

- p_c is the probability that an element of S belongs to class c
  - .. basically, the fraction of elements of S belonging to class c
- Intuition: Entropy is a measure of the "degree of surprise"
  - Some dominant classes $\Longrightarrow$ small entropy (less uncertainty)
  - Equiprobable classes $\Longrightarrow$ high entropy (more uncertainty)
- Entropy denotes the average number of bits needed to encode S

Entropy

$Entropy(S)$ = expected number of bits needed to encode class ($\oplus$ or $\ominus$) of randomly drawn member of S (under the optimal, shortest-length code)

Why?

Information theory: optimal length code assigns $-\log_2 p$ bits to message having probability p.

So, expected number of bits to encode $\oplus$ or $\ominus$ of random member of S:

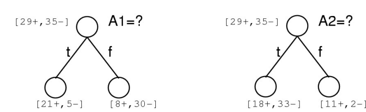
$$p_\oplus(-\log_2 p_\oplus) + p_\ominus(-\log_2 p_\ominus)$$

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

Information Gain

$Gain(S, A)$ = expected reduction in entropy due to sorting on A

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



Information Gain

- Let's assume each element of S consists of a set of features
- Information Gain (IG) on a feature F

$$IG(S,F) = H(S) - \sum_{f \in F} \frac{|S_f|}{|S|} H(S_f)$$

- S_f number of elements of S with feature F having *value* f
- $IG(S,F)$ measures the increase in our certainty about S once we know the value of F
- $IG(S,F)$ denotes the number of bits saved while encoding S once we know the value of the feature F

Computing Information Gain

- Let's begin with the root node of the DT and compute IG of each feature
- Consider feature "wind" $\in$ {weak,strong} and its IG w.r.t. the root node

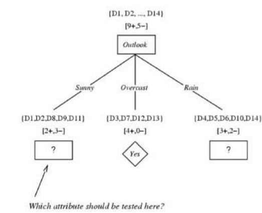
day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	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

- Root node: $S = [9+, 5-]$ (all training data: 9 play, 5 no-play)
- Entropy: $H(S) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.94$
- $S_{weak} = [6+, 2-] \Longrightarrow H(S_{weak}) = 0.811$
- $S_{strong} = [3+, 3-] \Longrightarrow H(S_{strong}) = 1$

$$\begin{aligned} IG(S, \text{wind}) &= H(S) - \frac{|S_{weak}|}{|S|} H(S_{weak}) - \frac{|S_{strong}|}{|S|} H(S_{strong}) \\ &= 0.94 - 8/14 * 0.811 - 6/14 * 1 \\ &= 0.048 \end{aligned}$$

Choosing the most informative feature

- At the root node, the information gains are:
  - $IG(S, \text{wind}) = 0.048$ (we already saw)
  - $IG(S, \text{outlook}) = 0.246$
  - $IG(S, \text{humidity}) = 0.151$
  - $IG(S, \text{temperature}) = 0.029$
- "outlook" has the maximum $IG \Longrightarrow$ chosen as the root node
- Growing the tree:
  - Iteratively select the feature with the highest information gain for each child of the previous node



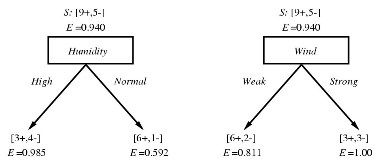
Training Example

PlayTennis: 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

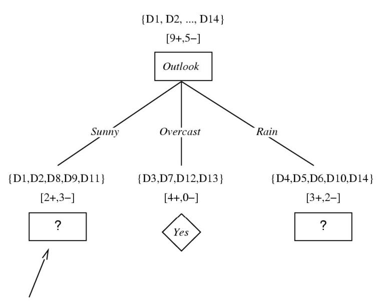
Selecting the Next Attribute

Which attribute is the best classifier?



Gain (S, Humidity)
= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)
= .940 - (8/14).811 - (6/14)1.0
= .048



Which attribute should be tested here?

S_{sunny} = {D1,D2,D8,D9,D11}

$Gain\ (S_{sunny}, Humidity)$ = .970 − (3/5) 0.0 − (2/5) 0.0 = .970

$Gain\ (S_{sunny}, Temperature)$ = .970 − (2/5) 0.0 − (2/5) 1.0 − (1/5) 0.0 = .570

$Gain\ (S_{sunny}, Wind)$ = .970 − (2/5) 1.0 − (3/5) .918 = .019