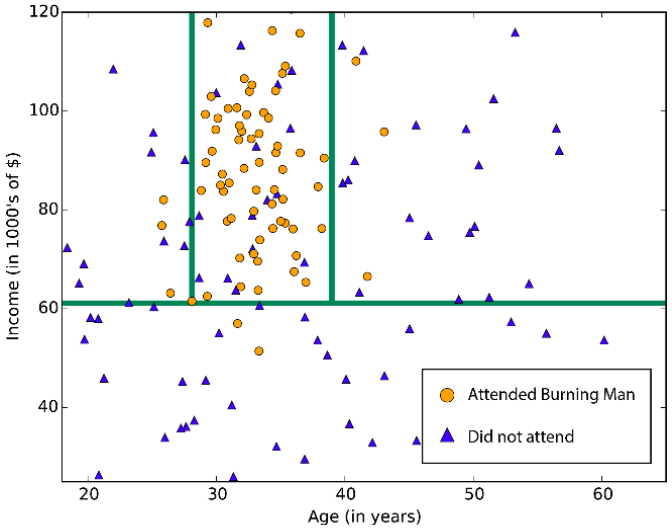

# Trees, Entropy, Forests, Boosting

## Data, Signals, and Systems

Ikhlaq Sidhu
Chief Scientist & Founding Director,
Sutardja Center for Entrepreneurship & Technology
IEOR Emerging Area Professor Award, UC Berkeley

# Decision Tree Illustration



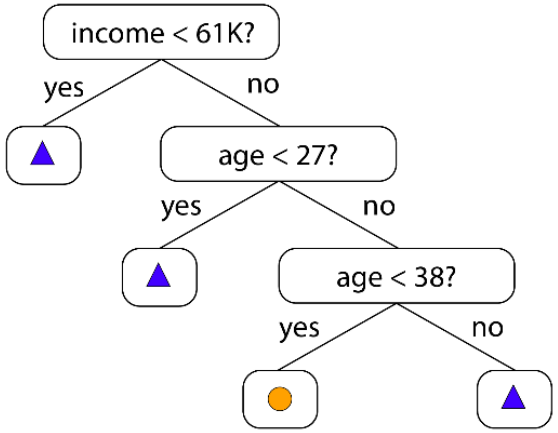
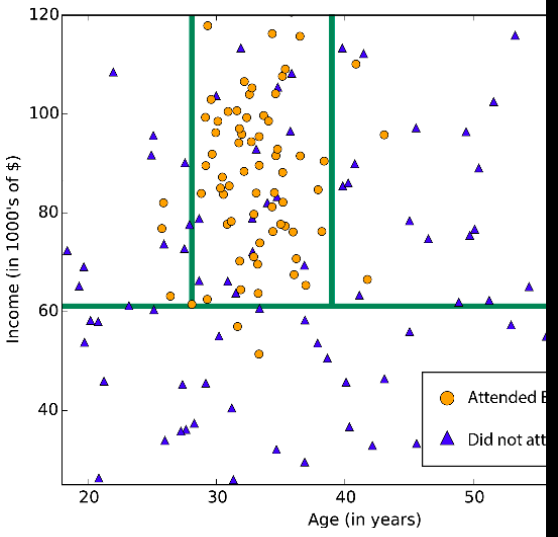




Illustration Source: https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice

# Decision Tree Illustration

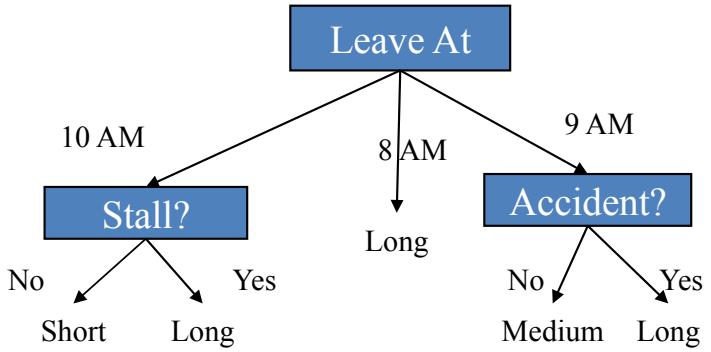



```
from sklearn import tree

decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, Y_train)
Y_pred = decision_tree.predict(X_test)

# Error
acc_decision_tree =
round(decision_tree.score(X_train, Y_train) *
100, 2)
acc_decision_tree

# or compare Y_pred with Y_test
```

Illustration Source: https://docs.mic

# How Are Trees Created?

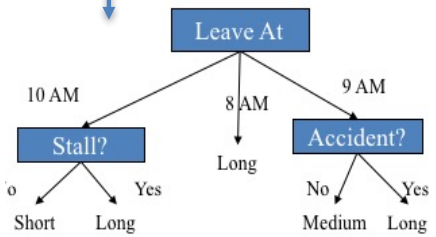
| | Attributes | | | | Target |
|---|---|---|---|---|---|
| | Hour | Weather | Accident | Stall | Commute |
| D1 | 8 AM | Sunny | No | No | Long |
| D2 | 8 AM | Cloudy | No | Yes | Long |
| D3 | 10 AM | Sunny | No | No | Short |
| D4 | 9 AM | Rainy | Yes | No | Long |
| D5 | 9 AM | Sunny | Yes | Yes | Long |
| D6 | 10 AM | Sunny | No | No | Short |
| D7 | 10 AM | Cloudy | No | No | Short |
| D8 | 9 AM | Rainy | No | No | Medium |
| D9 | 9 AM | Sunny | Yes | No | Long |
| D10 | 10 AM | Cloudy | Yes | Yes | Long |
| D11 | 10 AM | Rainy | No | No | Short |
| D12 | 8 AM | Cloudy | Yes | No | Long |
| D13 | 9 AM | Sunny | No | No | Medium |




Example from Jeff Story, source Internet

# How Are Trees Created?

| | Attributes | | | | Target |
|---|---|---|---|---|---|
| | Hour | Weather | Accident | Stall | Commute |
| D1 | 8 AM | Sunny | No | No | Long |
| D2 | 8 AM | Cloudy | No | Yes | Long |
| D3 | 10 AM | Sunny | No | No | Short |
| D4 | 9 AM | Rainy | Yes | No | Long |
| D5 | 9 AM | Sunny | Yes | Yes | Long |
| D6 | 10 AM | Sunny | No | No | Short |
| D7 | 10 AM | Cloudy | No | No | Short |
| D8 | 9 AM | Rainy | No | No | Medium |
| D9 | 9 AM | Sunny | Yes | No | Long |
| D10 | 10 AM | Cloudy | Yes | Yes | Long |
| D11 | 10 AM | Rainy | No | No | Short |
| D12 | 8 AM | Cloudy | Yes | No | Long |
| D13 | 9 AM | Sunny | No | No | Medium |




Prediction can be coded

All features might even be not be used (Occam's Razor)

```
if hour == 8am
    commute time = long
else if hour == 9am
    if accident == yes
      commute time = long
    else
      commute time = medium
else if hour == 10am
    if stall == yes
      commute time = long
    else
      commute time = short
```

Example from Jeff Story, source Internet

# How Are Trees Created?

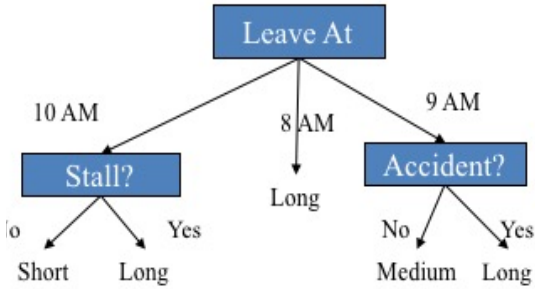
| | Attributes | | | | Target |
|---|---|---|---|---|---|
| | Hour | Weather | Accident | Stall | Commute |
| D1 | 8 AM | Sunny | No | No | Long |
| D2 | 8 AM | Cloudy | No | Yes | Long |
| D3 | 10 AM | Sunny | No | No | Short |
| D4 | 9 AM | Rainy | Yes | No | Long |
| D5 | 9 AM | Sunny | Yes | Yes | Long |
| D6 | 10 AM | Sunny | No | No | Short |
| D7 | 10 AM | Cloudy | No | No | Short |
| D8 | 9 AM | Rainy | No | No | Medium |
| D9 | 9 AM | Sunny | Yes | No | Long |
| D10 | 10 AM | Cloudy | Yes | Yes | Long |
| D11 | 10 AM | Rainy | No | No | Short |
| D12 | 8 AM | Cloudy | Yes | No | Long |
| D13 | 9 AM | Sunny | No | No | Medium |

But how do we choose the order of decisions in the tree?

Most Famous: ID3 Algorithm:

ID3 splits on attributes with the lowest entropy (highest information gained)

| Attribute | Expected Entropy | Information Gain |
|---|---|---|
| Hour | 0.6511 | 0.768449 |
| Weather | 1.28884 | 0.130719 |
| Accident | 0.92307 | 0.496479 |
| Stall | 1.17071 | 0.248842 |




Example from Jeff Story, source Internet

# What is Entropy?

$$H(x) = \sum_{x} p(x) \log\left(\frac{1}{p(x)}\right)$$

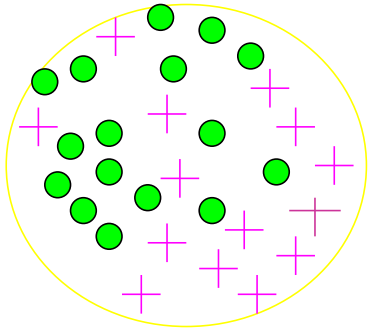
A Measure of a distribution P(X)

X = { 1 with prob 1 ; 0 with prob 0 } → Outcome / Reception → Suppose outcome = 1

X = { 1 with prob 0.5 ; 0 with prob 0.5 } → Outcome / Reception → Suppose outcome = 1

# What is Entropy?

- Entropy = $\sum_i -p_i \log_2 p_i$



$p_i$ is the probability of class i

Compute it as the proportion of class i in the set.

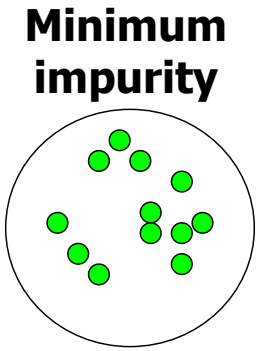
16/30 are green circles; 14/30 are pink crosses
$\log_2(16/30) = -.9$; $\log_2(14/30) = -1.1$
Entropy = -(16/30)(-.9) –(14/30)(-1.1) = .99

Source Internet

# 2-Class Cases:

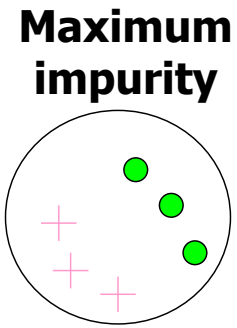
- What is the entropy of a group in which all examples belong to the same class?
  - entropy = - 1 $\log_2 1 = 0$

  not a good training set for learning




- What is the entropy of a group with 50% in either class?
  - entropy = $-0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$

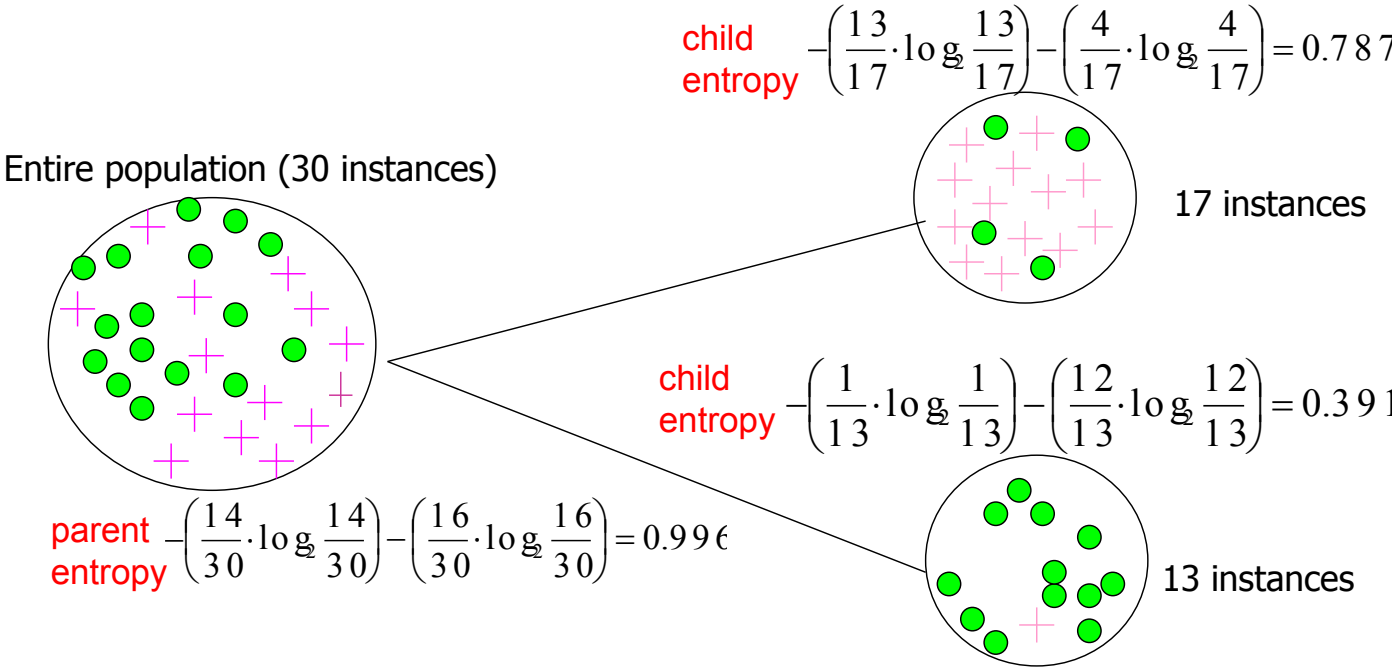
  good training set for learning




Source Internet

# Calculating Information Gain

**Information Gain** = entropy(parent) – [average entropy(children)]



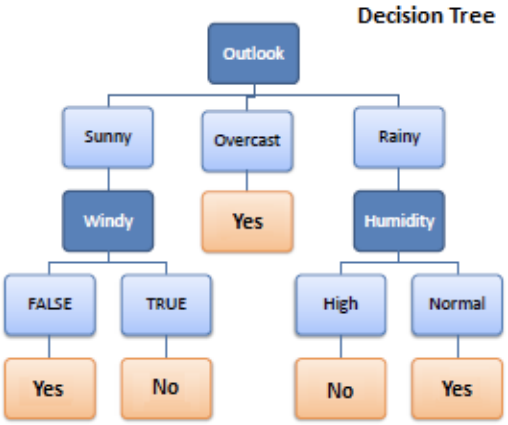

(Weighted) Average Entropy of Children = $\left(\frac{17}{30}\cdot 0.787\right)+\left(\frac{13}{30}\cdot 0.391\right)=0.615$

**Information Gain= 0.996 - 0.615 = 0.38 for this split**

# Golf Example

Predictors | Target

| Outlook | Temp. | Humidity | Windy | Play Golf |
|---|---|---|---|---|
| Rainy | Hot | High | False | No |
| Rainy | Hot | High | True | No |
| Overcast | Hot | High | False | Yes |
| Sunny | Mild | High | False | Yes |
| Sunny | Cool | Normal | False | Yes |
| Sunny | Cool | Normal | True | No |
| Overcast | Cool | Normal | True | Yes |
| Rainy | Mild | High | False | No |
| Rainy | Cool | Normal | False | Yes |
| Sunny | Mild | Normal | False | Yes |
| Rainy | Mild | Normal | True | Yes |
| Overcast | Mild | High | True | Yes |
| Overcast | Hot | Normal | False | Yes |
| Sunny | Mild | High | True | No |

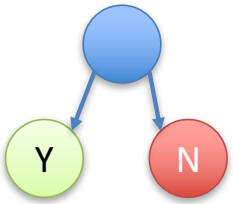



Start with table of experiences

- Distribution: P(golf) = 9 yes, 5 no14 observations
- We calculate entropy of distribution: playing golf
- Need to calculate entropy of each next possible decision (Outlook, Temp, Humidity, and Windy)
- Information gained is E(Golf) – E(Subset, Golf),
- We choose the next branch to be the decision with the highest information gained for the next branch

# Golf Example

Top Level
Play Golf
Distribution



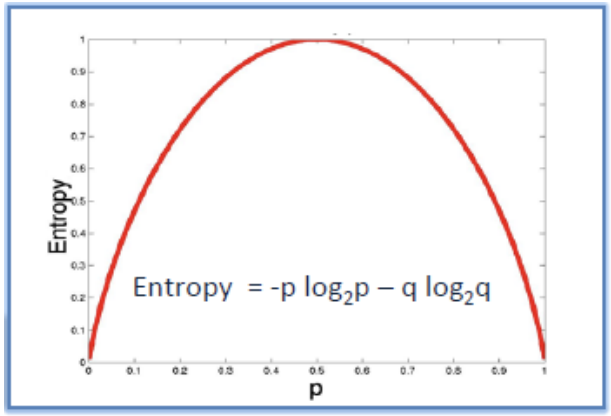

9/14 5/14

Predictors | Target

| Outlook | Temp. | Humidity | Windy | Play Golf |
|---|---|---|---|---|
| Rainy | Hot | High | False | No |
| Rainy | Hot | High | True | No |
| Overcast | Hot | High | False | Yes |
| Sunny | Mild | High | False | Yes |
| Sunny | Cool | Normal | False | Yes |
| Sunny | Cool | Normal | True | No |
| Overcast | Cool | Normal | True | Yes |
| Rainy | Mild | High | False | No |
| Rainy | Cool | Normal | False | Yes |
| Sunny | Mild | Normal | False | Yes |
| Rainy | Mild | Normal | True | Yes |
| Overcast | Mild | High | True | Yes |
| Overcast | Hot | Normal | False | Yes |
| Sunny | Mild | High | True | No |

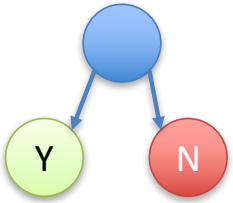
$$E(G) = 9/14 \log_2 1/(9/14) + 5/14 \log_2 1/(5/14) = 0.94$$

- Distribution: P(golf) = 9 yes, 5 no14 observations
- We calculate entropy of distribution: playing golf
- Need to calculate entropy of each next possible decision (Outlook, Temp, Humidity, and Windy)
- Information gained is E(Golf) – E(Subset, Golf),
- We choose the next branch to be the decision with the highest information gained for the next branch




Entropy = $-0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$

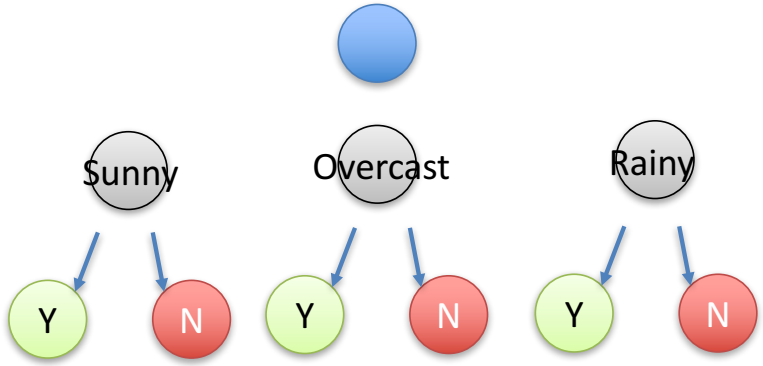
Top Level
Play Golf
Distribution




9/14 5/14

$H(G) = 9/14 \log_2 1/(9/14) + 5/14 \log_2 1/(5/14)$
$= 0.94$

Next Level: Play Golf with Outlook




3/5 2/5 4/4 0/4 2/5 3/5

H(G,OL) =
P(Sunny) H(G,OL) + P(OV)H(G,OV) + P(Rainy) H(G,Rainy)

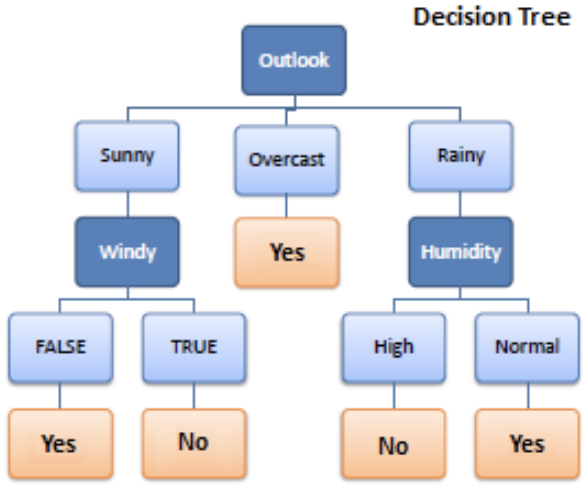
= 5/14* 0.971+4/14*0 + 5/14*.971 = 0.693

Info Gained = H(G) - H(G,OL) = 0.94 – 0.693 = 0.247

Do the same with Temp (.29), Windy(.048), and Humidity(.152):
Choose next node to be the one with the most info gained

# How Are Trees Constructed

| Predictors | | | | Target |
|---|---|---|---|---|
| Outlook | Temp. | Humidity | Windy | Play Golf |
| Rainy | Hot | High | False | No |
| Rainy | Hot | High | True | No |
| Overcast | Hot | High | False | Yes |
| Sunny | Mild | High | False | Yes |
| Sunny | Cool | Normal | False | Yes |
| Sunny | Cool | Normal | True | No |
| Overcast | Cool | Normal | True | Yes |
| Rainy | Mild | High | False | No |
| Rainy | Cool | Normal | False | Yes |
| Sunny | Mild | Normal | False | Yes |
| Rainy | Mild | Normal | True | Yes |
| Overcast | Mild | High | True | Yes |
| Overcast | Hot | Normal | False | Yes |
| Sunny | Mild | High | True | No |




Start with table of experiences
Train = construct a tree. Predict means use the tree

But how do we know which branch should go first, second, next?

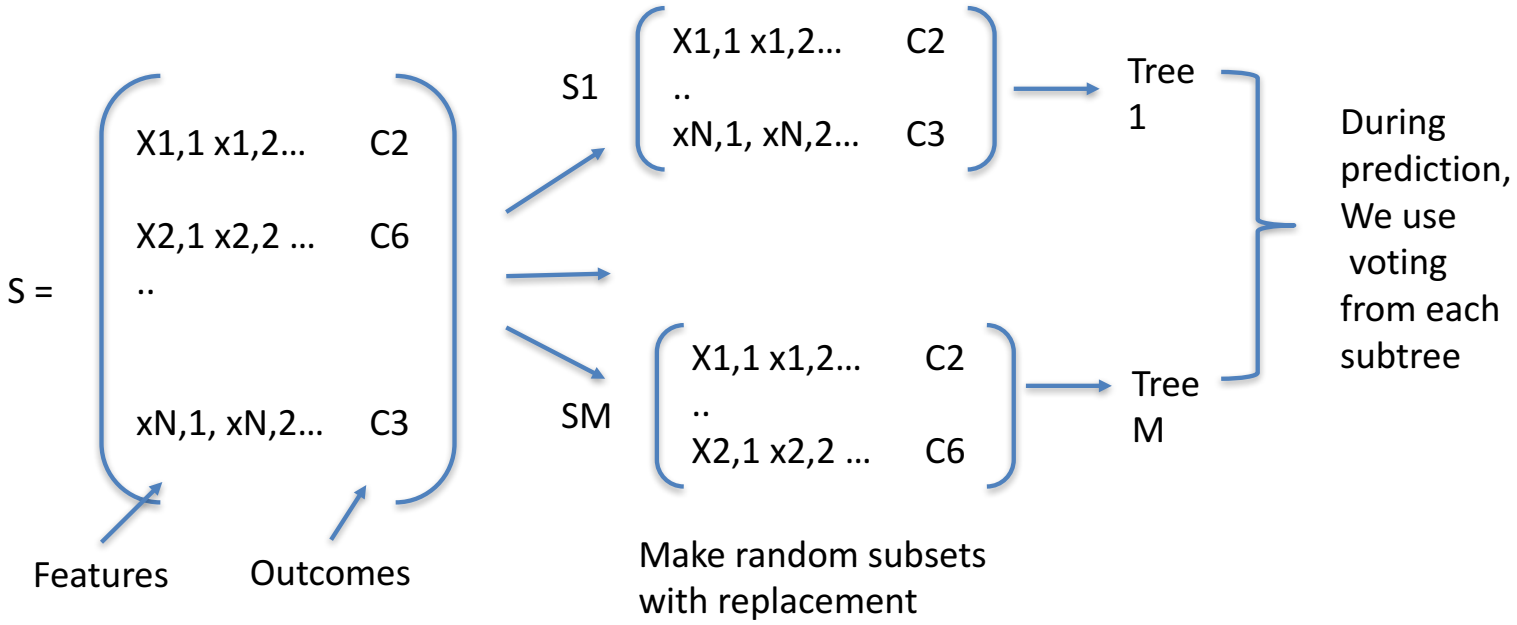
Example Source Internet: http://www.saedsayad.com/decision_tree.htm

# Trees Can be Extended with Bagging

Explain bagging and Random Forrest

```
from sklearn.ensemble import RandomForestClassifier

random_forest =
RandomForestClassifier(n_estimators=1000)
random_forest.fit(X_train, Y_train)
Y_pred = random_forest.predict(X_test)
random_forest.score(X_train, Y_train)

# Error
acc_random_forest = round(random_forest.score(X_train,
Y_train) * 100, 2)
acc_random_forest

# or compare Y_pred with Y_test
```

# Random Forest – A type of bagging/ensemble approach




Advantages: One of most accurate
Efficient prediction over large data

Disadvantages: Overfit and Training time

# Our experiment with the Titanic Data Set

| | Model | Score |
|---|---|---|
| | Random Forest | 86.76 |
| | Decision Tree | 86.76 |
| | KNN | 84.74 |
| | Support Vector Machines | 83.84 |
| | Logistic Regression | 80.36 |
| | Linear SVC | 79.01 |
| | Perceptron | 78.00 |
| | Naive Bayes | 72.28 |
| | Stochastic Gradient Decent | 72.28 |

More Accuracy
Generally more training time
More risk of overfitting

Less Accuracy
Generally less computation

# Boosting as in AdaBoost

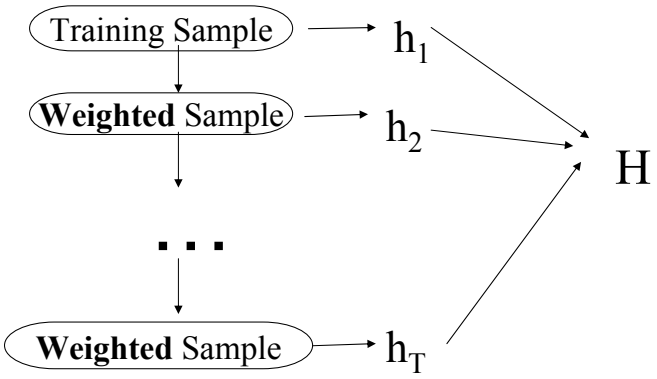
- Motivation - a procedure that combines the outputs of many "weak" classifiers to produce a powerful "committee"
- A machine learning algorithm
- Perform supervised learning
- Increments improvement of learned function
- Forces the weak learner to generate new hypotheses that make less mistakes on "harder" parts.

- Freund & Schapire (1995) – **AdaBoost**
  - strong practical advantages over previous boosting algorithms

Benk Erika

Kelemen Zsolt

# General Concept of Boosting




- Train a set of weak hypotheses: $h_1$, …., $h_T$.
- The combined hypothesis H is a **weighted** majority vote of the T weak hypotheses.
  - ➔ Each hypothesis $h_t$ has a weight $\alpha_t$.

$$H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$$

Benk Erika
Kelemen Zsolt

# Boosting

- Binary classification problem
- Training data:

$$(x_1, y_1), \ldots, (x_m, y_m), \text{where} \quad x_i \in X, y_i \in Y = \{-1, 1\}$$

- $D_t(i)$: the weight of $x_i$ at round t. $D_1(i)=1/m$.
- A learner L that finds a weak hypothesis $h_t$: X ➔ Y given the training set and $D_t$
- The error of a weak hypothesis $h_t$:

$$\varepsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i] = \sum_{i: h_t(x_i) \neq y_i} D_t(i)$$

Benk Erika
Kelemen Zsolt

# AdaBoost Algorithm

- **Given** $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \epsilon X$, $y_i \epsilon \{-1, +1\}$
- **Initialise** weights $D_1(i) = 1/m$
- **Iterate** $t=1, \ldots, T$:
  - Train weak learner using distribution ***Dt***
  - Get weak classifier: $h_t : X \rightarrow R$
  - Choose $\alpha_t \varepsilon R$
  - Update: $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
    - where $Z_t$ is a normalization factor (chosen so that $D_{t+1}$ will be a distribution), and $\alpha_t$:

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) > 0$$

- **Output** – the final classifier

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Benk Erika
Kelemen Zsolt

AdaBoost Algorithm

# Discrete AdaBoost - Algorithm

- **Given** $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \epsilon X$, $y_i \epsilon \{-1, +1\}$
- **Initialise** weights $D_1(i) = 1/m$
- **Iterate** $t=1,\ldots,T$:
  - Find $h_t = \arg\min_{h_j} \epsilon_j$ where $\epsilon_j = \sum_{i=1}^{m} D_t(i)[\![h_t(x_i) \neq y_i]\!]$
  - Set

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

  - Update: $D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$

- **Output** – the final classifier

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

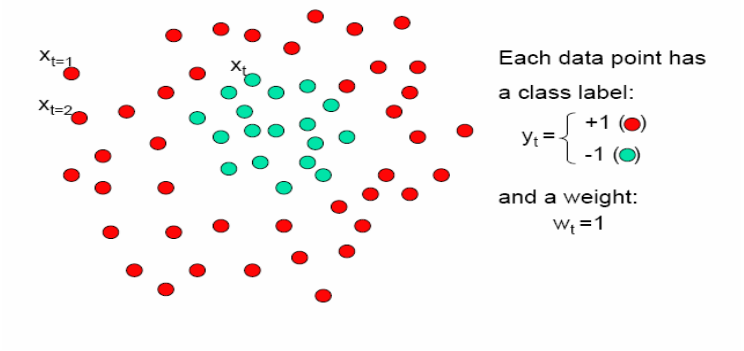
Benk Erika
Kelemen Zsolt

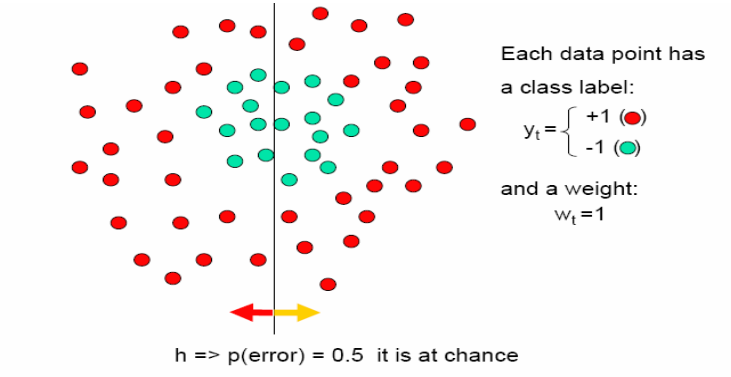
# AdaBoost Algorithm

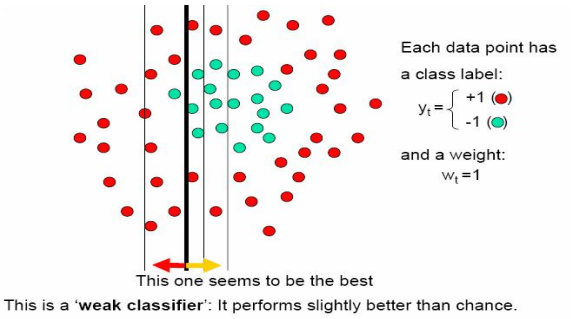
- the weights *Dt*(*i*) are updated and normalised on each round. The normalisation factor takes the form

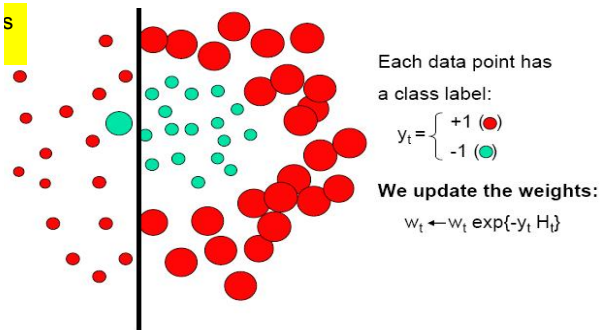
$$Z_t = \sum_{i=1}^{m} D_t(i) \mathrm{e}^{-\alpha_t y_i h_t(x_i)}$$

Benk Erika
Kelemen Zsolt

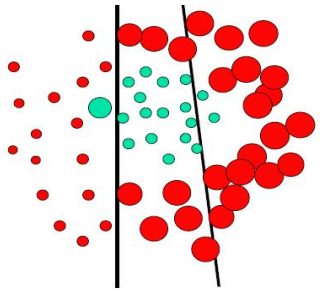

$x_{t-1}$
$x_t$
$x_{t-2}$
Each data point has
a class label:
$y_t = \begin{cases} +1 \\ -1 \end{cases}$
and a weight:
$w_t = 1$


Each data point has
a class label:
$y_t = \begin{cases} +1 \\ -1 \end{cases}$
and a weight:
$w_t = 1$
h => p(error) = 0.5 it is at chance


Each data point has
a class label:
$y_t = \begin{cases} +1 \\ -1 \end{cases}$
and a weight:
$w_t = 1$
This one seems to be the best
This is a '**weak classifier**': It performs slightly better than chance.


S
Each data point has
a class label:
$y_t = \begin{cases} +1 \\ -1 \end{cases}$
**We update the weights:**
$w_t \leftarrow w_t \exp\{-y_t H_t\}$


Data X



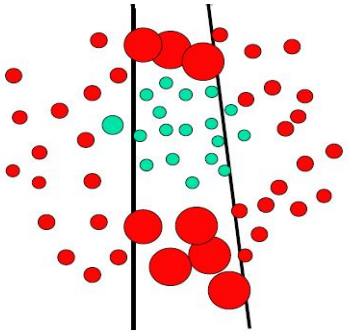
Each data point has
a class label:

$y_t = \begin{cases} +1 \\ -1 \end{cases}$

**We update the weights:**

$w_t \leftarrow w_t \exp\{-y_t H_t\}$

We set a new problem for which the previous weak classifier performs at chance again

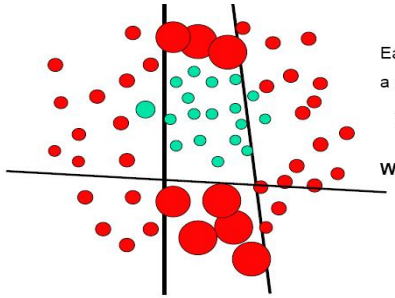


Each data point has
a class label:

$y_t = \begin{cases} +1 \\ -1 \end{cases}$

**We update the weights:**

$w_t \leftarrow w_t \exp\{-y_t H_t\}$

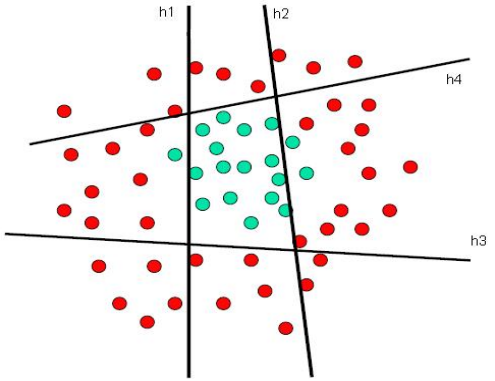


Each data point has
a class label:

$y_t = \begin{cases} +1 \\ -1 \end{cases}$

**We update the weights:**

$w_t \leftarrow w_t \exp\{-y_t H_t\}$




be strong (non-linear) classifier is built as the comb

# AdaBoost – Pros and Contras

- Pros:
  - Very simple to implement
  - Fairly good generalization
  - The prior error need not be known ahead of time
- Contras:
  - Suboptimal solution
  - Can over fit in presence of noise

End of Section