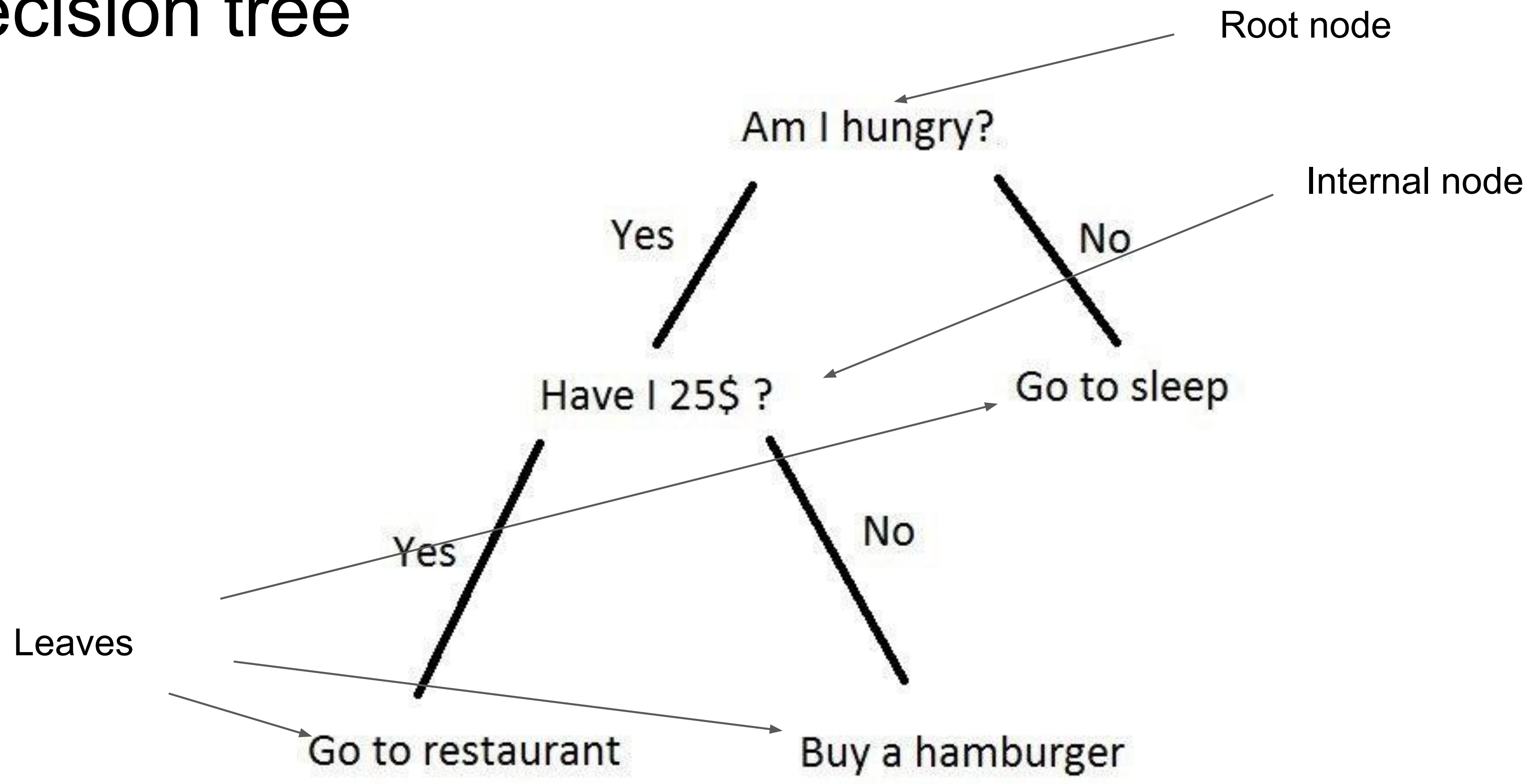
Lab 7. Decision Trees

Intro to Machine Learning
Fall 2018, Innopolis University

Lecture recap

- What are Decision trees?
- Decision Tree Learning
- Overfitting in Decision Trees and How to Handle It?
- What is Information Gain?
- Ensemble Learning
- What is Bagging?
- How Random Forests algorithm works?

Decision tree



Decision tree learning algorithms

- ID3 (Iterative Dichotomiser 3)
- C4.5 (successor of ID3)
- CART (Classification And Regression Tree)
- CHAID (CHi-squared Automatic Interaction Detector).
- MARS: extends decision trees to handle numerical data better.

ID3

Algorithm:

- For every attribute (feature) calculate the entropy
- Split the training set using the one for which information gain is maximum
- Continue recursively on subsets using remaining features

Stopping criterias:

- If all records in current data subset have the same output
- If all records have exactly the same values of input attributes

ID3 Step by step

Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

We have four X values (Outlook, Temperature, Humidity and Windy) being categorical and one y value (Play Tennis Yes or No) also being categorical.

We need to find mapping between X and y

We want to solve it using decision tree

To build a tree we need to start from a root node

ID3 Step by step

Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Q: How to choose which predictor should be first?

A: We should select attribute that best classifies the training data

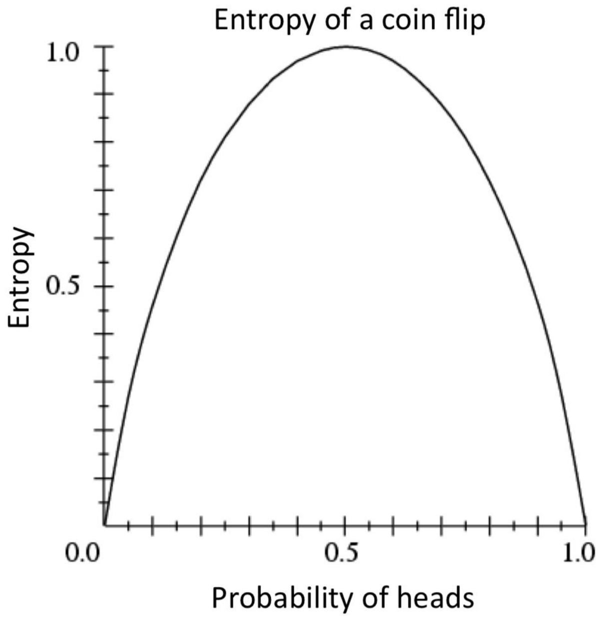
Q: How to select the best attribute?

A: Best attribute - attribute with largest **information gain**

Q: What is information gain and how to calculate it?

A: Information gain shows how much information entropy changes after data splitting.

Entropy (Shannon entropy)



$$H(S) = \sum_{x \in X} -p(x) \log_2 p(x)$$

S – The current (data) set for which entropy is being calculated (changes every iteration of the ID3 algorithm)
X – Set of classes in ***S***
p(x) – The proportion of the number of elements in class ***x*** to the number of elements in set ***S***

High and Low entropy

High entropy

- Y is from a uniform like distribution
- Flat histogram
- Values sampled from it are less predictable

Low entropy

- Y is from a varied (peaks and valleys) distribution
- Histogram has many lows and highs
- Values sampled from it are more predictable

Information gain

$$IG(A,S) = H(S) - \sum_{t \in T} p(t)H(t)$$

H(S) – Entropy of set **S**

T – The subsets created from splitting set **S** by attribute **A**

p(t) – The proportion of the number of elements in **t** to the number of elements in set **S**

H(t) – Entropy of subset **t**

ID3 Step by step

Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

$$H(S) = \sum_{x \in X} -p(x) \log_2 p(x) \qquad x \in \{yes, no\}$$

$$H(S) = -\frac{n_{yes}}{N} log_2(\frac{n_{yes}}{N}) - \frac{n_{no}}{N} log_2(\frac{n_{no}}{N})$$

$$H(S) = -\frac{9}{14} log_2(\frac{9}{14}) - \frac{5}{14} log_2(\frac{5}{14}) = 0.94$$

Next let's calculate entropy for each attribute and information gain

ID3 Step by step

Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Outlook attribute

$$H(Outlook = sunny) = -\frac{2}{5}log_2(\frac{2}{5}) - \frac{3}{5}log_2(\frac{3}{5}) = 0.971$$

$$H(Outlook = overcast) = -\frac{4}{4}log_2(\frac{4}{4}) - \frac{0}{4}log_2(\frac{0}{4}) = 0$$

$$H(Outlook = rainy) = -\frac{3}{5}log_2(\frac{3}{5}) - \frac{2}{5}log_2(\frac{2}{5}) = 0.971$$

$$IG(A, S) = H(S) - \sum_{t \in T} p(t)H(t)$$

$$IG(Outlook, S) = H(S)$$
$$- \frac{5}{14}H(Outlook = sunny)$$
$$- \frac{4}{14}H(Outlook = overcast)$$
$$- \frac{5}{14}H(Outlook = rainy) = 0.247$$

ID3 Step by step

Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Calculate Information gain for Wind attribute

ID3 Step by step

Windy attribute

Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

$$H(Windy = True) = -\frac{3}{6}log_2(\frac{3}{6}) - \frac{3}{6}log_2(\frac{3}{6}) = 1$$

$$H(Windy = False) = -\frac{6}{8}log_2(\frac{6}{8}) - \frac{2}{8}log_2(\frac{2}{8}) = 0.811$$

$$IG(A, S) = H(S) - \sum_{t \in T} p(t)H(t)$$

$$IG(Windy, S) = H(S) - \frac{8}{14}H(Windy = False)$$
$$- \frac{6}{14}H(Windy = True) = 0.048$$

ID3 Step by step

Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

$$IG(Outlook, S) = 0.247$$

$$IG(Windy, S) = 0.048$$

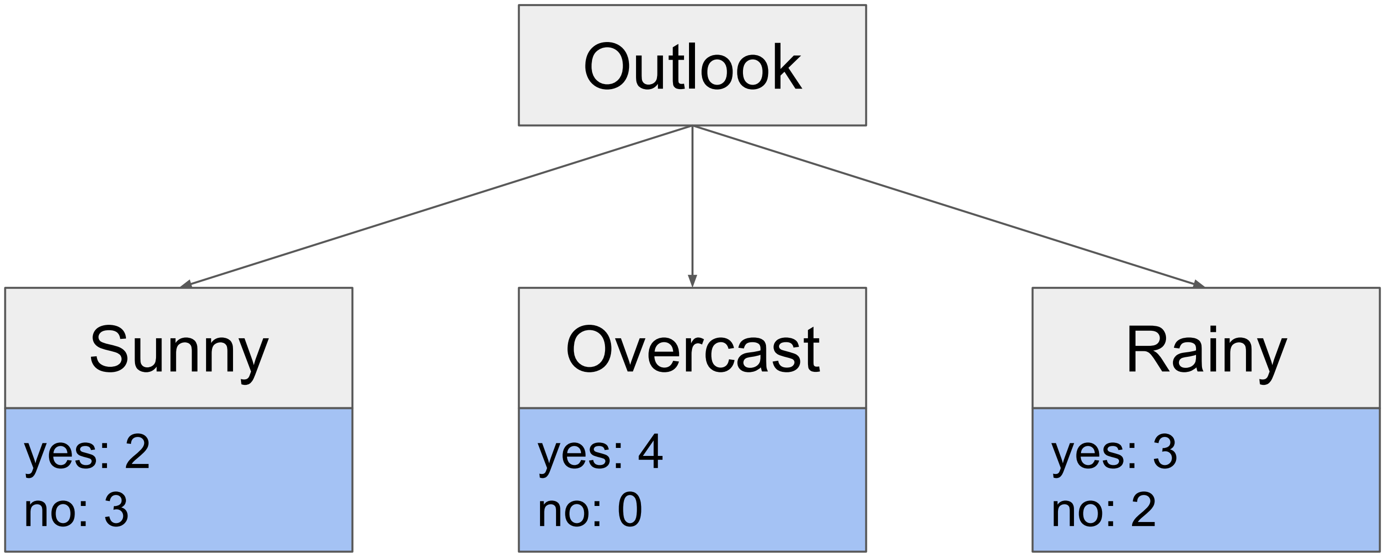
$$IG(Temperature, S) = 0.029$$

$$IG(Humidity, S) = 0.152$$

Outlook has the biggest Information gain, so we chose it as root node in our tree

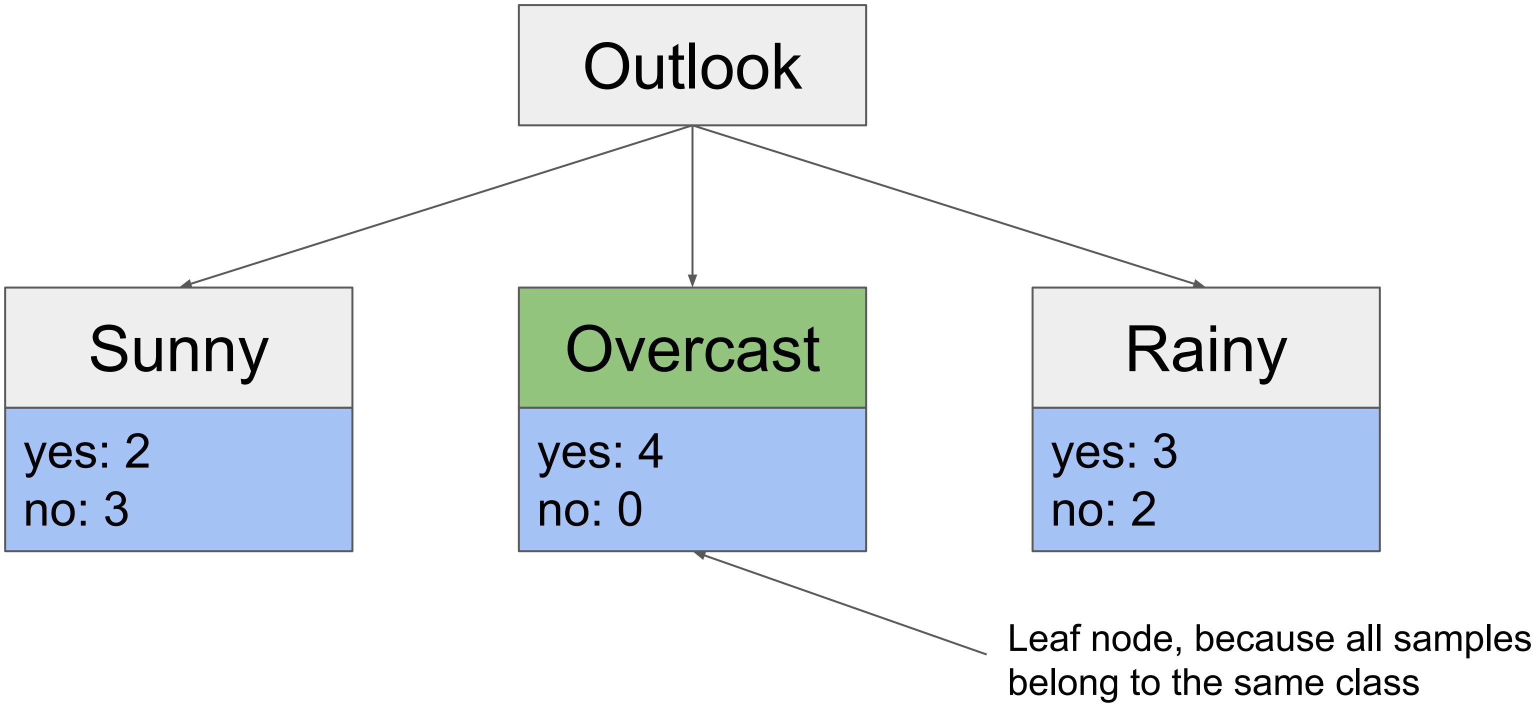
ID3 Step by step

Decision tree after the first iteration



ID3 Step by step

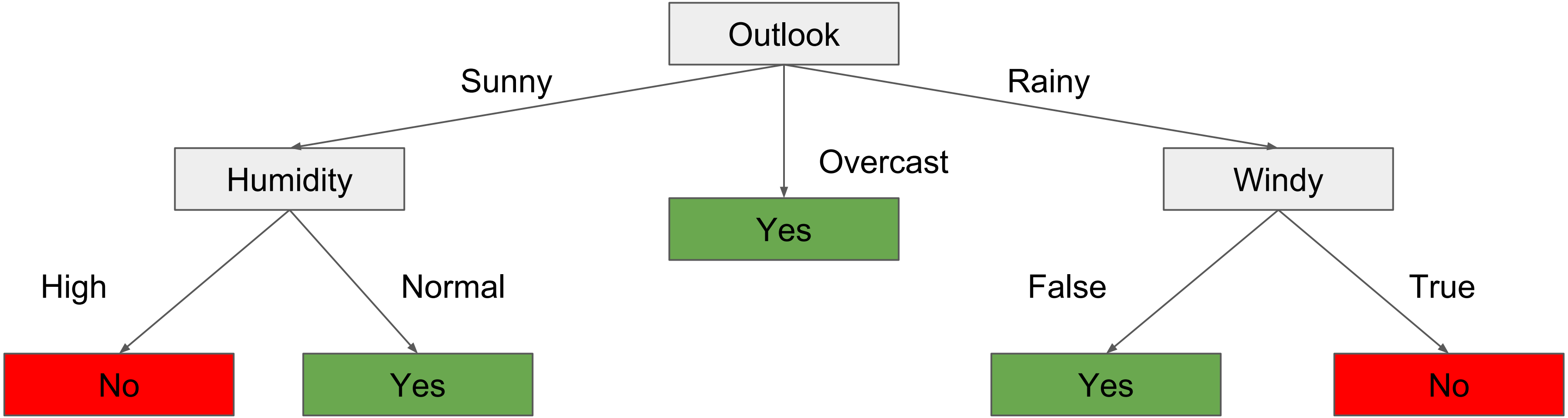
Decision tree after the first iteration



ID3 Step by step

Repeat the same for each sub-tree

Final tree



ID3 implementation

Implement function which calculates entropy
Implement function which calculates information gain

ID3 implementation

```python
import numpy as np

def entropy(values):
  normilized = values / np.sum(values)
  normilized[normilized == 0] = 1
  return -1.0 * np.dot(normilized.T, np.log2(normilized))

def information_gain(S,A):
  entropies = np.array([entropy(A[e,:]) for e in range(A.shape[0])])
  gain = entropy(S) - np.dot(np.sum(A, axis=1)/np.sum(S), entropies)
  return gain
```

ID3 problems

- The main problem is that the algorithm may overfit easily (tree does not stop growing until the whole training set is classified)
- It handles only discrete attributes
- There is a strong bias for features with many possible outcomes
- It does not handle missing values
- Tree cannot be updated when new data is classified incorrectly, instead new tree must be generated
- Only one attribute at time is tested for making decision

C4.5

Improved version of ID3 with:

- Continuous values using threshold
- Tree pruning to avoid overfitting
- Normalized information gain
- Missing values

Why C4.5 better than ID3

Let’s take a look on dataset

name	John	Mark	Anne	Adam	John	Alex	Alex	Xena	Tina	Lucy
sex	M	M	F	M	M	F	M	F	F	F
age	old	young	old	young	young	young	old	old	young	young
play games	N	Y	Y	Y	Y	N	N	N	Y	Y

Let’s calculate Information Gain for each attribute

$$IG(Name, S) = 0.771$$
$$IG(Sex, S) = 0$$
$$IG(Age, S) = 0.256$$

In this case we would choose name as the best predictor
Creating a tree with 8 branches (from 10 samples)
Training data would be perfectly classify,
but it is unlikely that the algorithm would be able to generalize for unseen data

Let’s calculate another method instead caled **Information Gain Ratio**

Information gain ratio

$$R(X) = \frac{G(X)}{V(X)}$$

$$V(X) = -\sum_{i=1}^{N} \frac{|T_i|}{|T|} \cdot \log(\frac{|T_i|}{|T|})$$

G(X) - Information gain
V(X) - Intrinsic value of an attribute X
Ti - Samples corresponding to *i*-th possible value of *X* feature

Why C4.5 better than ID3

name	John	Mark	Anne	Adam	John	Alex	Alex	Xena	Tina	Lucy
sex	M	M	F	M	M	F	M	F	F	F
age	old	young	old	young	young	young	old	old	young	young
play games	N	Y	Y	Y	Y	N	N	N	Y	Y

Let’s calculate Information Gain Ratio for each attribute

$$IGR(Name, S) = 0.263$$
$$IGR(Sex, S) = 0$$
$$IGR(Age, S) = 0.264$$

Based on information gain ratio we choose age as the best predictor

Because the denominator in a ratio penalizes features with many values (Similar to regularization technique)

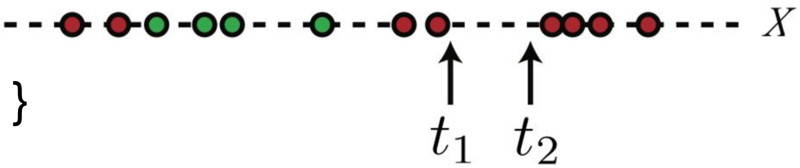
Real valued inputs

What should we do if some of our predictors are real-valued data?

Any ideas what we should do?

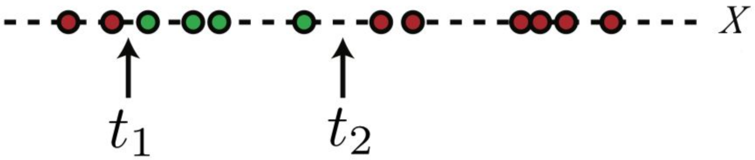
Real valued inputs

Threshold splits



Sort data according to X into {x_1 ,...,x_m }
Consider split points of the form

$$\frac{x_i + x_{i+1}}{2}$$



Picking the best threshold

H(Y|X:t) = p(X < t) H(Y|X < t) + p(X >= t) H(Y|X >= t)

IG(Y|X:t) = H(Y) - H(Y|X:t)

IG*(Y|X) = $\max_t$ IG(Y|X:t)

Why C4.5 better than ID3

Let's age will be a numerical continuous value

```
   name    ||  John  |  Mark  |  Anne  |  Adam  |  John  |  Alex  |  Alex  |  Xena  |  Tina  |  Lucy  |
----------------------------------------------------------------------------------------------------------
   sex     ||   M    |   M    |   F    |   M    |   M    |   F    |   M    |   F    |   F    |   F    |
----------------------------------------------------------------------------------------------------------
   age     ||   50   |   18   |   65   |   24   |   31   |   18   |   50   |   50   |   24   |   31   |
==========================================================================================================
play games ||   N    |   Y    |   Y    |   Y    |   Y    |   N    |   N    |   N    |   Y    |   Y    |
```

Let’s calculate possible thresholds and select the best one.
First we should sort data according attribute data

Sex	18	18	24	24	31	31	50	50	50	65
Play games	Y	N	Y	Y	Y	Y	N	N	N	Y

Why C4.5 better than ID3

Now let's calculate possible threshold values

Sex	18	18	24	24	31	31	50	50	50	65
Play games	Y	N	Y	Y	Y	Y	N	N	N	Y

$$\frac{x_i + x_{i+1}}{2}$$

t	18	21	24	27.5	31	40.5	50	50	57.5

Why C4.5 better than ID3

Calculate and select threshold with the highest Information Gain value

t	18	21	24	27.5	31	40.5	50	50	57.5
IG	0.0074	0.0074	0.0464	0.0464	0.2564	0.2564	0.07898	0.07898	0.07898

We can select one of them

Why C4.5 better than ID3

C4.5 can work with unknown attributes

$$G(X) = F \cdot (H(T) - H(T, X))$$

The information gain is calculated as before for samples with known attributes

But then it is normalized with respect to the probability that the given attribute has known values

$\boldsymbol{F}$ - the ratio of the number of samples with known value for a given feature to the number of all samples in a dataset

Pruning

The algorithm creates as many nodes as needed to classify all test samples, It may lead to overfitting and the resulting tree would fail to classify correctly unseen samples

To avoid this one can prune a tree. There are 2 types of pruning

pre-pruning (early stopping)

- stop building a tree before leaves with few samples are produced
- how to decide when it is good time to stop? e.g. using cross-validation on validation set (stop if the error does not increase significantly)
- underfitting if stop too early

post-pruning

- let a tree grow completely
- then go from bottom to top and try to replace a node with a leaf
- if there is improvement in accuracy - cut a tree
- if the accuracy stays the same - cut a tree (Occam's razor)
- otherwise leave a node

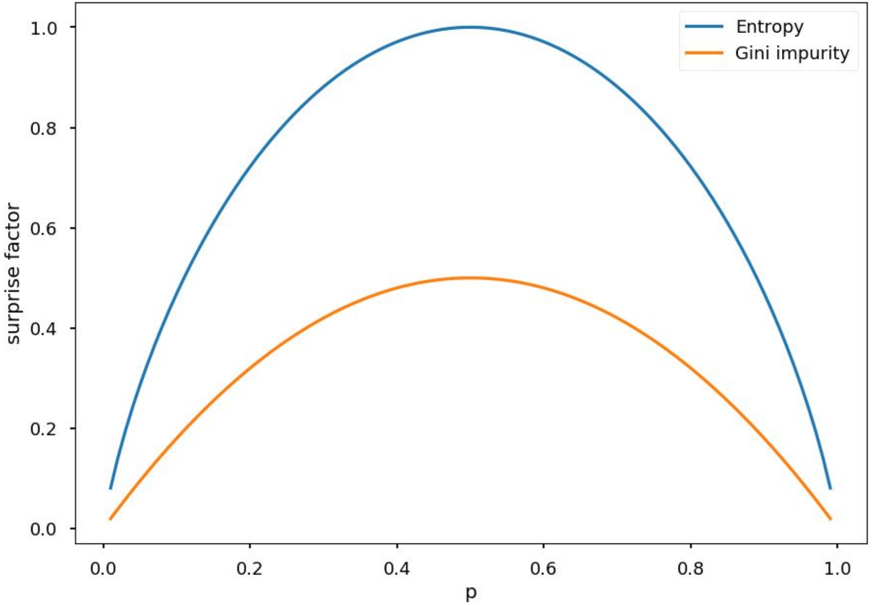
CART (Classification And Regression Trees)

- Creates binary trees (each decision node has two branches)
- Uses gini impurity instead of information gain
- Supports numerical target variables (regression)

Algorithm:

- For every attribute (feature) calculate the Gini index
- Split the training set using the one for which Gini index is lowest
- Continue recursively on subsets using remaining features

Gini impurity



$$Gini(t) = 1 - \sum_{i=1}^{j} P(i|t)^2$$

P(i|t) - the probability of selecting an element of class i from the node's subset
t - the subset of instances for the node
j - number of classes

$$Gini_{\max} = 1 - \frac{1}{n}$$

Gini impurity

```
def gini_impurity(values):
  return 1 - np.sum((values / np.sum(values))**2)
```

CART in Scikit-learn

```
from sklearn.cross_validation import train_test_split
from sklearn import tree
from sklearn.metrics import accuracy_score

X = golf_data_num.iloc[:,:4]
Y = golf_data_num.iloc[:,4]

X_train, X_test , y_train,y_test = train_test_split(X, Y, test_size = 0.3)

cart_tree = tree.DecisionTreeClassifier()
cart_tree.fit(X,Y)
y_pred_en = cart_tree.predict(X_test)

print("Accuracy is :{0}".format(accuracy_score(y_test.astype(int),y_pred_en)))
```

Regression trees

The difference now is that targets are numerical values (instead of categorical), e.g. in golf data - number of hours played instead of "yes / no"

Features may be either discrete or continuous

The idea is the same though - we want to create a binary tree and minimize the error on in each leaf

However, having continuous values as targets we can not simply use entropy or gini

We need to use different measurement - variance

$$V(X) = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2$$

Sklearn implementation

```
from sklearn.datasets import load_boston
from sklearn.tree import DecisionTreeRegressor
from sklearn.cross_validation import cross_val_score
from sklearn.cross_validation import KFold

boston = load_boston()
X, y = boston.data, boston.target
features = boston.feature_names
regression_tree = tree.DecisionTreeRegressor(min_samples_split=30,
                                             min_samples_leaf=10)
regression_tree.fit(X,y)
crossvalidation = KFold(n=X.shape[0], n_folds=5, shuffle=True, random_state=1)
score = np.mean(cross_val_score(regression_tree, X, y,
 scoring='mean_squared_error', cv=crossvalidation,
  n_jobs=1))
print('Mean squared error: %.3f' % abs(score))
```

Homework

Modify ID3 to C4.5 for classification and train it on Titanic dataset

1) Modify Information gain calculation to Information gain ratio
2) Add prediction function
3) Add ability to work with continuous data
4) Add any pruning technique (pre/post)

*) Visualization