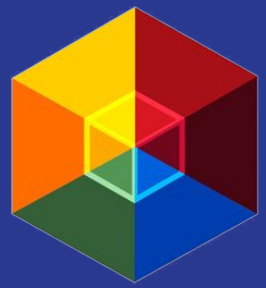
Decision Tree

Tim Analisis
Nov 10 2016

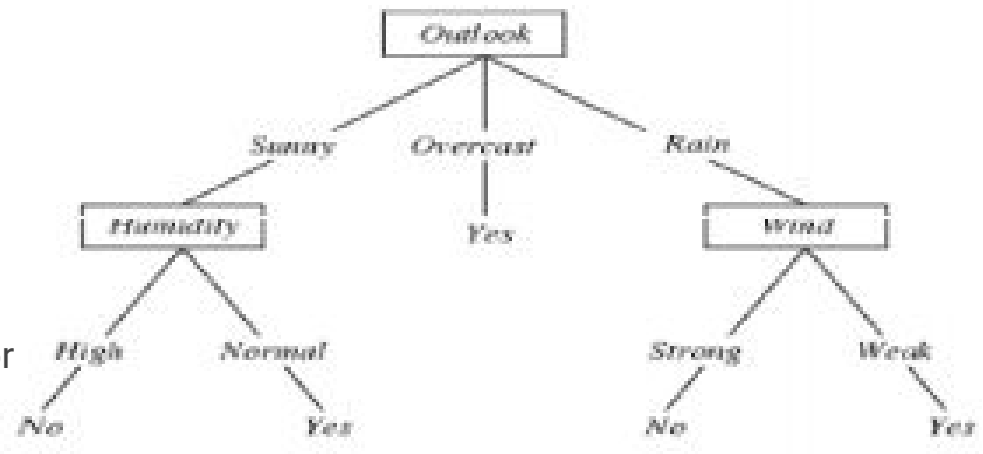
KMKLabs - Senayan City



REFRESH .....

Unsupervised / Clustering	Supervised / Classification
1. K-Means 2. HCS clustering 3. Canopy 4. DBSCAN 5. Fuzzy Clustering 6. K-SVD 7. Pitman More than 100 approaches	1. Naive Bayes 2. Linear Regression 3. Decision Tree 4. Random Forest 5. Support Vector Machine 6. Neural Network 7. Deep Neural Network

Decision Tree



Problem Setting :

- Set of possible instance X
  - Each instance x in X is a feature vector
- Unknown target function f: X → Y
  - Y is discrete value
- Set of function hypotheses H = { h | h : X→ Y }
  - Each hypotheses h is a decision tree
  - Trees sorts x to leaf, which assigns y

- Each Internal node → Test one attribute Xi
- Each branch from node → selects one value for Xi
- Each leaf node → predict Y (or P(Y|X))

F: <Outlook, Humidity, Wind, Temp> → PlayTennis?

Top Down Induction of Decision Trees

Node = Root

Main loop:

1. A ← the ‘best’ decision attribute for next node
2. Assign A as decision attribute for node
3. For each value of A, create new descendant of node
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

Which attributes is best?

Entropy & Information Gain

- Entropy → calculate homogenity
- Entropy with one attribute :

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

- Entropy with two attribute :

$$E(T, X) = \sum_{c \in X} P(c) E(c)$$

- Information Gain :

$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

Example

Outlook	Temperature	Humidity	Wind	PlayTenı
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

1. Calculate initial entropy of Play Tennis

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Play Golf	
Yes	No
9	5

Entropy(PlayGolf) = Entropy (5,9)
= Entropy (0.36, 0.64)
= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)
= 0.94

2. Calculate entropy with 2 variables (choose one attribute)

$$E(T, X) = \sum_{c \in X} P(c) E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

E(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)
= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971
= 0.693

$$E(T,X) = \sum_{c \in X} P(c)E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

E(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)

= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971

= 0.693

3. Calculate Gain for every attributes

4. Choose attribute with most Gain value

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

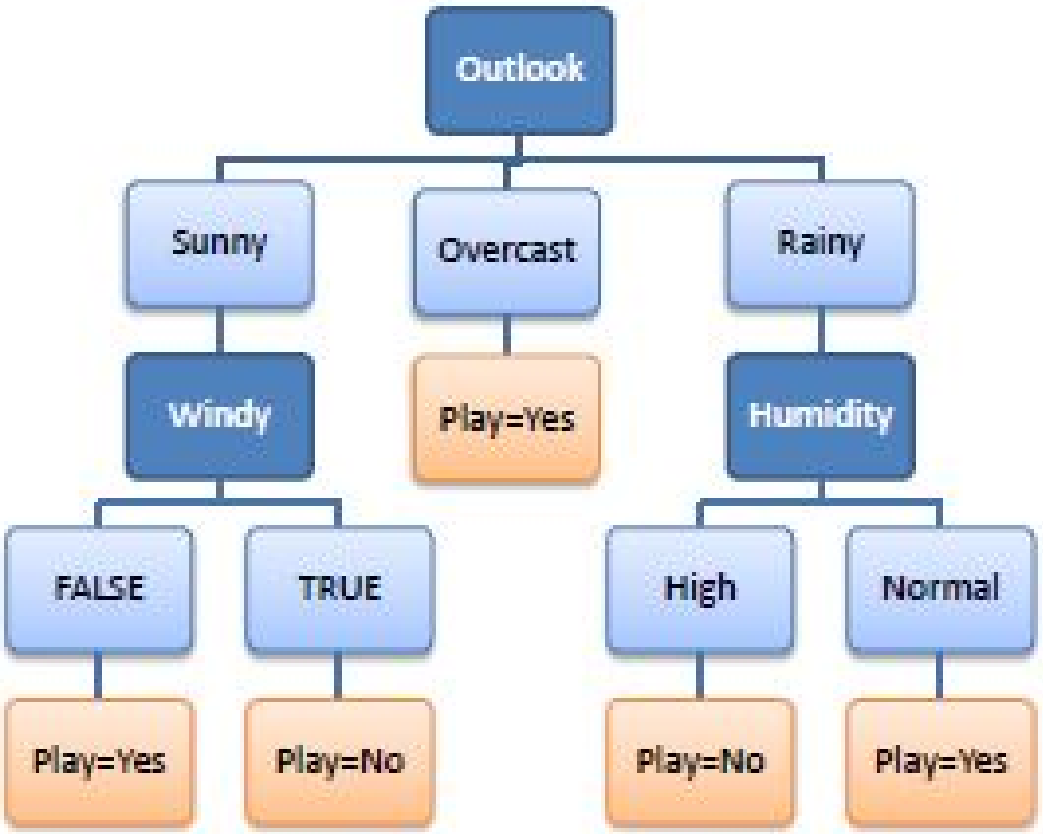
		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

5. Repeat with the rest of data

The Decision Tree



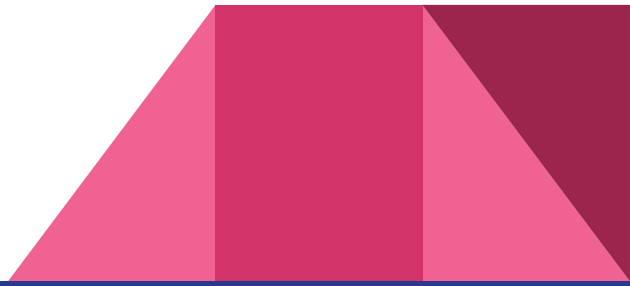
Overfitting

Consider error of hypothesis h over

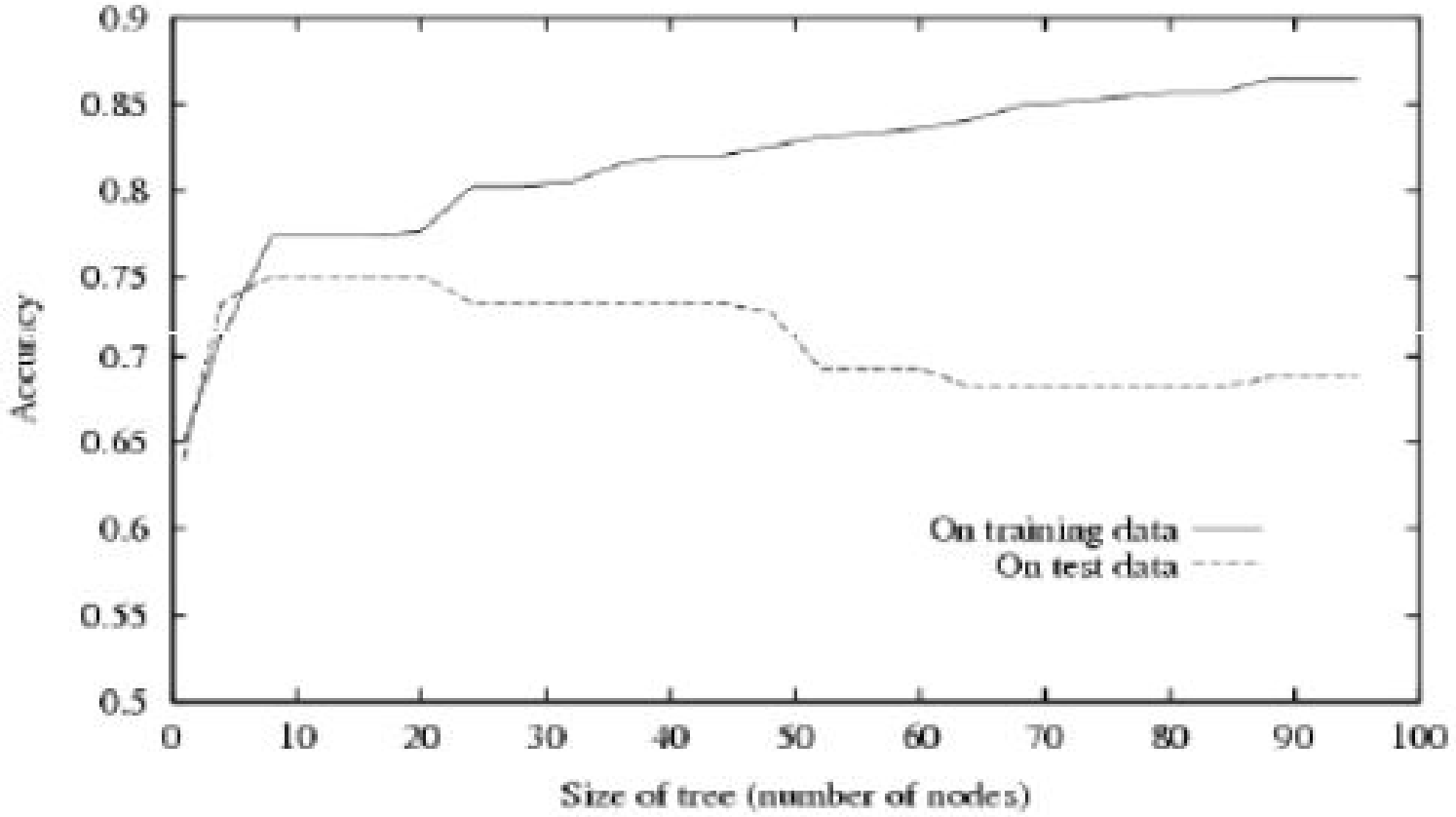
- Training data : $error_{train}(h)$
- Entire distribution D of data : $error_{D}(h)$

Hypothesis $h \in H$ overfits training data if there is an alternative hypothesis $h' \in H$ such that :

$$error_{train}(h) < error_{train}(h) \text{ AND } error_{train}(h) > error_{train}(h)$$

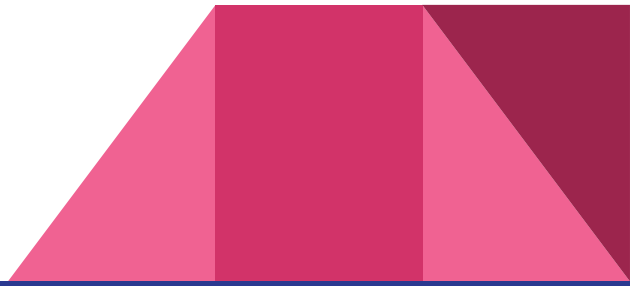


Overfitting in Decision Tree Learning

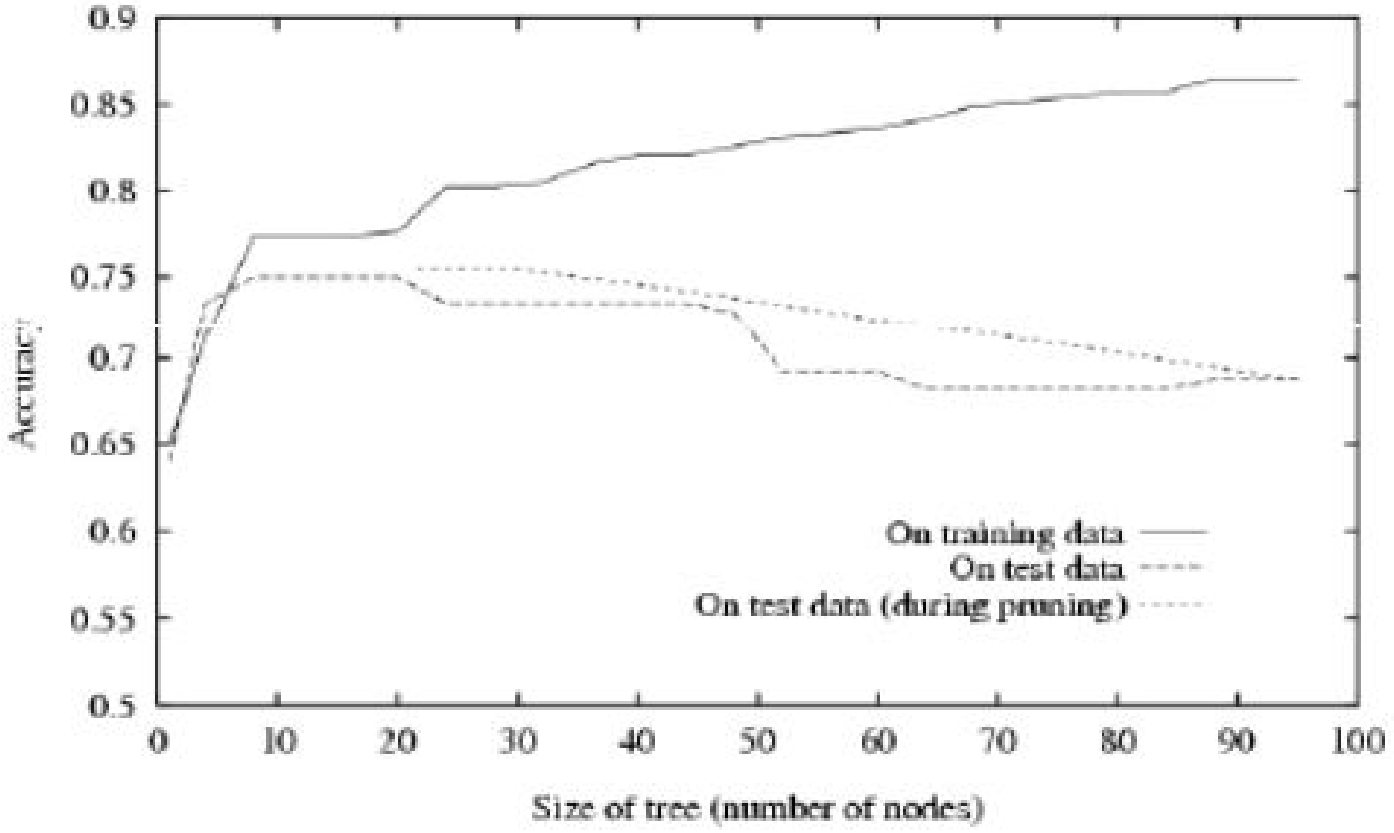


Avoiding Overfitting

- Stop growing when data split not statistically significant
- Grow full tree, then post prune



Effect of Reduced-Error Pruning



Decision Tree on Sklearn

`DecisionTreeClassifier` is capable of both binary (where the labels are [-1, 1]) classification and multiclass (where the labels are [0, ..., K-1]) classification.

Using the Iris dataset, we can construct a tree as follows:

```
>>> from sklearn.datasets import load_iris
>>> from sklearn import tree
>>> iris = load_iris()
>>> clf = tree.DecisionTreeClassifier()
>>> clf = clf.fit(iris.data, iris.target)
```

Once trained, we can export the tree in Graphviz format using the `export_graphviz` exporter. Below is an example export of a tree trained on the entire iris dataset:

```
>>> with open("iris.dot", 'w') as f:
...     f = tree.export_graphviz(clf, out_file=f)
```

HOMEWORK :D

Suppose one's desire of food (appealing) was determined by some features as shown in data below. Generate the decision tree by choosing node with most information gain value in every split

Appealing	Temperature	Taste	Size
No	Hot	Salty	Small
No	Cold	Sweet	Large
No	Cold	Sweet	Large
Yes	Cold	Sour	Small
Yes	H	Sour	Small
No	H	Salty	Large
Yes	H	Sour	Large
Yes	Cold	Sweet	Small
Yes	Cold	Sweet	Small
No	H	Salty	Large

thank you :)