Decision Trees

-Entrophy = gain information

Tian-Li Yu

Taiwan Evolutionary Intelligence Laboratory (TEIL)
Department of Electrical Engineering
National Taiwan University
tianliyu@ntu.edu.tw

Readings: AIMA 18.3, 18.4, 18.10

Outline

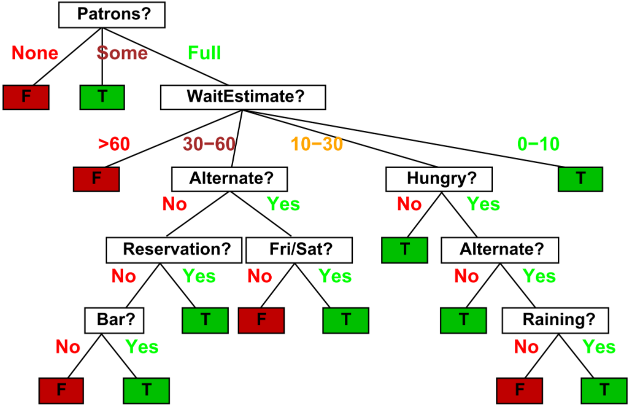
Attribute-based Representations

- Restaurant example.

Example	Attributes										Target
	Alt	Bar	Fri	Hun	Pat	$Price$	$Rain$	Res	$Type$	Est	*WillWait*
X_1	*T*	*F*	*F*	*T*	*Some*	*$$$*	*F*	*T*	*French*	*0–10*	*T*
X_2	*T*	*F*	*F*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*30–60*	*F*
X_3	*F*	*T*	*F*	*F*	*Some*	*$*	*F*	*F*	*Burger*	*0–10*	*T*
X_4	*T*	*F*	*T*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*10–30*	*T*
X_5	*T*	*F*	*T*	*F*	*Full*	*$$$*	*F*	*T*	*French*	*>60*	*F*
X_6	*F*	*T*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Italian*	*0–10*	*T*
X_7	*F*	*T*	*F*	*F*	*None*	*$*	*T*	*F*	*Burger*	*0–10*	*F*
X_8	*F*	*F*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Thai*	*0–10*	*T*
X_9	*F*	*T*	*T*	*F*	*Full*	*$*	*T*	*F*	*Burger*	*>60*	*F*
X_{10}	*T*	*T*	*T*	*T*	*Full*	*$$$*	*F*	*T*	*Italian*	*10–30*	*F*
X_{11}	*F*	*F*	*F*	*F*	*None*	*$*	*F*	*F*	*Thai*	*0–10*	*F*
X_{12}	*T*	*T*	*T*	*T*	*Full*	*$*	*F*	*F*	*Burger*	*30–60*	*T*

Decision Trees

- One possible representation for hypotheses.



Expressiveness of Decision Trees

- $Goal \Leftrightarrow (Path_1 \vee Path_2 \vee \cdots)$.
- $Path_i \Leftrightarrow (Attribute_1 = a_1 \wedge Attribute_2 = a_2 \wedge \cdots)$.
- Decision trees can express any function of the input attributes.
- Trivially, there is a consistent decision tree for any training set with one path to leaf for each example, but it won't generalize to new examples.
- Prefer to find more compact decision trees.

Hypothesis Space

- How many distinct decision trees with n Boolean attributes?
  - Truth table with 2^n rows.
  - Every truth table can be expressed by one decision tree $\Rightarrow$ At least 2^{2^n} decision trees.
  - If different order of attributes counts as $\Rightarrow$ At least $n! \cdot 2^{2^n}$ decision trees.
- More expressive hypothesis space
  - Increase the chance that c can be expressed.
  - May be weak at generalization if we let the decision tree be too expressive.

Learning Decision Trees (ID3 [Quinlan, 1986])

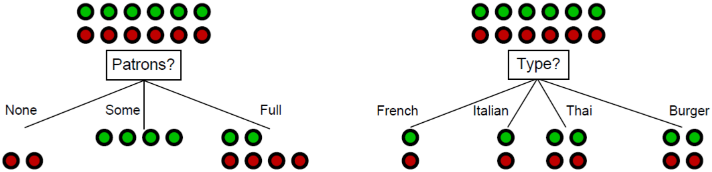
- Aim: Find a small tree consistent with training examples.
- Idea: Recursively choose the best attribute.

$\mathrm{DTL}(\textit{examples}, \textit{attributes}, \textit{examples}_{\textit{parent}})$

if *examples* is empty **then return** $\textsc{Plurality-Value}(\textit{examples}_{\textit{parent}})$
elseif all *examples* have same classification **then return** the classification
elseif *attributes* is empty **then return** $\textsc{Plurality-Value}(\textit{examples})$
else
    $A \leftarrow \mathrm{argmax}_{a \in \textit{attributes}} \textsc{Importace}(a, \textit{examples})$
    tree $\leftarrow$ a new decision tree with root A
    for textbfeach value v_k of A
        exs $\leftarrow$ elements of *examples* with $A = v_k$
        subtree $\leftarrow \mathrm{DTL}(\textit{exs}, \textit{attributes} - A, \textit{examples})$
        add a branch to *tree* with label $A = v_k$ and subtree *subtree*
    return *tree*

Choosing Attributes

- The restaurant example consist of 6 positive and 6 negative examples.
- *Patrons* is a better choice — gives more information about the classification.



Information

- Measure of information: Shannon's entropy.
    - Gives the lower bound of the most compact encoding of a random variable in bits.
- The entropy of a random variable V with values v_k, each with probability $P(v_k)$, is defined as

$$H(V) = -\sum_k P(v_k) \log_2 P(v_k).$$

- For Boolean variables, define

$$B(q) = -q \log_2 q - (1-q) \log_2 (1-q).$$

- The entropy of a fair coin:
  $H = B(0.5) = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1 bit$.
- The entropy of a unfair coin (99% head):
  $H = B(0.99) = -(0.99 \log_2 0.99 + 0.01 \log_2 0.01) = 0.08 bits$.

Information

- p positive and n negative examples at the root $\Rightarrow B(p/(p+n))$ bits needed to classify a new example.
- Attribute A splits the examples E into subsets E_k, each of which (we hope) needs less information to complete the classification.
- Let E_k have p_k positive and n_k negative examples
  $\Rightarrow B(p_k/(p_k+n_k))$ bits needed to classify a new example
  $\Rightarrow$ expected number of bits per example over all branches is

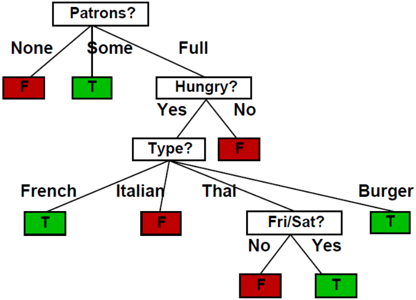
$$Remainder(A) = \sum_k \frac{p_k+n_k}{p+n} B\left(\frac{p_k}{p_k+n_k}\right).$$

- For *Patrons*, this is 0.459 bits; for *Type*, this is (still) 1 bit
  $\Rightarrow$ Choose the attribute that minimizes the remaining information.
  $\Rightarrow$ Choose the attribute with the most information gain:

$$Gain(A) = B\left(\frac{p}{p+n}\right) - Remainder(A)$$

Decision Tree Learned from the Examples

- Decision tree learned from the 12 examples:



- Substantially simpler than a full tree — a more complex hypothesis isn't justified by small amount of data.

Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

Metrics for Performance Evaluation

- Confusion matrix:

	PREDICTED CLASS		
ACTUAL CLASS	Text	Class=Yes	Class=No
	Class=Yes	True Positive	False Negative
	Class=No	False Positive	True Negative

Accuracy

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

對角線最大化約好

- Probably most widely-used metric.
- Can be misleading. Consider class 0 consisting of 9990 instances and class 1 consisting of 10 instances. Classifying everything as class 0 yields 99.9% accuracy.

當even時有缺點

Other Metrics

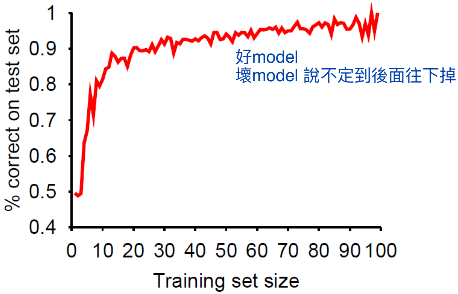
$$Precision(p) = \frac{TP}{TP + FP}$$

$$Recall(r) = \frac{TP}{TP + FN}$$

$$F - measure(F) = \frac{2pr}{p + r} = \frac{2TP}{2TP + FP + FN}$$

Performance Measurement

- How do we know whether $h \approx c$?
  1. Use theorems of computational/statistical learning theory
  2. Try h on a new test set of examples
     (use same distribution over example space as training set)
- Learning curve = % correct on test set as a function of training set size



Cross-Validation

data 不會均分
test data < train data (e.g. 1:3, 1:4…)

- The idea of having training and testing sets is called cross-validation.
- Holdout cross-validation
  - Randomly split the available data into a training set and a testing set.
  - Simple, fast, but not able to use all available data.
- k-fold cross-validation

  random均分成k份
  第n次用第n組當testing data，剩下k-1組training data
  - Randomly split the data into k equal-sized subsets.
  - Perform k rounds of learning using $k-1$ subsets as training and the rest as testing.
  - Popular choice of k is 5 to 10.
  - Accurate statistics, but longer computation.

ROC

- Receiver operating characteristic.
- ROC curve: FPR as x-axis; TPR as y-axis.
  false positive rate

$$FPR(FP\ rate) = \frac{FP}{FP + TN}$$

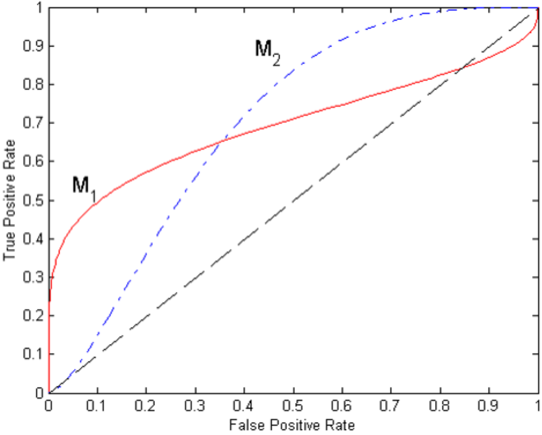
$$TPR(TP\ rate) = \frac{TP}{TP + FN}$$

- (FPR, TPR):
  - (0,0): Classify everything as negative.
  - (1,1): Classify everything as positive.
  - (0,1): Ideal. 沒有東西弄錯

AUC

- Model 1 is better for small FPR.
- Model 2 is better for large FPR.

- Area under the ROC curve (AUC).
  - Ideal: 1.
  - Random guess: 0.5.



Generalization and Overfitting

- If some attributes are irrelevant, DTL still outputs a large tree.
  - The outputs of fair dices with attributes of color, size, and so on.
- To overcome overfitting,
  - we can stop growing the tree before overfitting,
  - or we can allow overfitting, and then post-prune the tree (most common).
- How to decide what to post-prune?
  - Use the testing set.
  - Use statistical tests.
  - Use explicit measures the complexity of the encoding of the tree and training examples (minimum description length principle).

training 長樹
砍了某個以後testing有變好 就繼續post-prune

χ^2 Pruning

- Information gain of an irrelevant attribute is expected to be zero, but the sampling noise may still yield some gain.
- Assuming true irrelevant, the expected number of p_k and n_k can be expressed as

$$\hat{p}_k = \frac{p}{p+n} \times (p_k + n_k) \qquad \hat{n}_k = \frac{n}{p+n} \times (p_k + n_k)$$

- Define $\triangle = \sum_k \frac{(p_k - \hat{p}_k)^2}{\hat{p}_k} + \frac{(n_k - \hat{n}_k)^2}{\hat{n}_k}$, $\triangle$ is of χ^2 distribution with $(n + p - 1)$ degree of freedom.
- For example, with 3 degree of freedom, $\triangle \leq 7.82$ encourages the pruning with 5% level of significance. 不夠大->砍了

Rule Post-Pruning

J48 v8

- Used by C4.5rules [Quinlan, 1993].

tree 沒有避免 overfititng

1. Convert the decision tree into rules (one rule per path).
2. Prune each rule by removing any preconditions that improves its accuracy (by testing set). 砍砍看 precondition accuracy 上升，則take
3. Sort the the pruned rules by their accuracy, and consider them in this sequence when classifying instances.

- For example,

rule $(Patron = Full) \wedge (Hungry = No) \Rightarrow (WillWait = False)$.

- Rule post-pruning considers removing $(Patron = Full)$ and $(Hungry = NO)$ in this example.

Ensemble

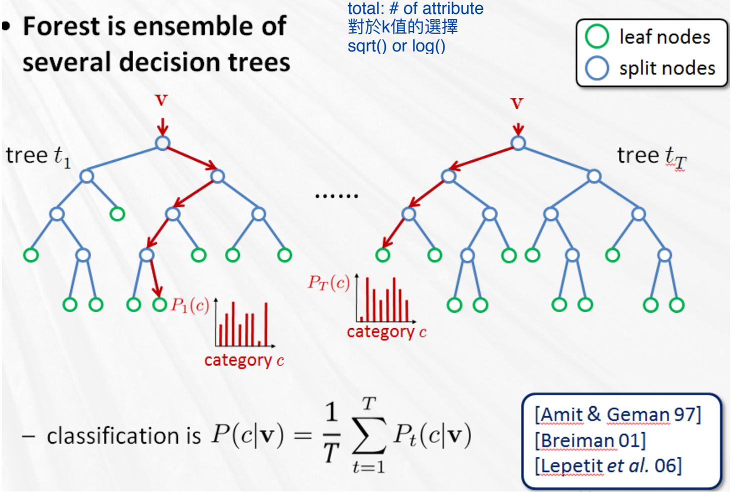
- Use multiple weak classifiers to prevent from overfitting.
- Embedding
- Bagging
- Boosting

Embedding

≈ Kernel method

- Random forest: randomly select *k* attributes to create weak classifiers.

total: # of attribute
對於k值的選擇
sqrt() or log()



Bagging

- Sampling with replacement from the dataset to form new datasets.

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- Build classifier on each bootstrap sample (supposedly n items).
- Probability $(1 - 1/n)^n$ of not being selected.n 個通通沒被選到
- When n is large, it is about $1/e \simeq 37\%$
- About 37% of noise (if any) not being selected.

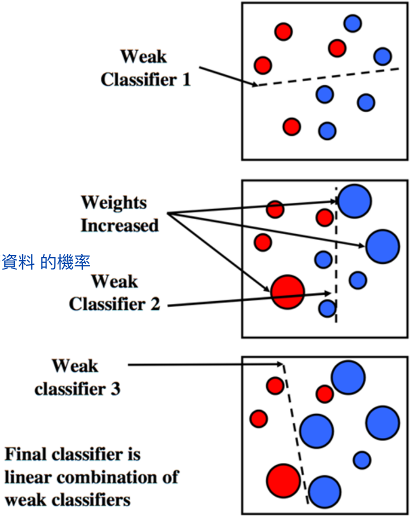
好處：有可能完全沒有noise

Boosting

if preprocessing is good, boosting
if not, bagging is better.

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records.
- Initially, all n items are assigned equal weights.
- Unlike bagging, weights vary at the end of boosting round.

weight 加大
每次調整那筆資料 的機率



AdaBoost

- Weak classifiers: C_i
- Error rates:

$$\epsilon_i = \frac{1}{N} \sum_{j=i}^{N} w_j \cdot \delta[C_i(x_j) \neq y_j]$$

.

- Importance of a classifier:

錯誤率越小 越重要

$$\alpha_i = \frac{1}{2} \ln \frac{1-\epsilon_i}{\epsilon_i}$$

- Weight update (c normalization factor):

$$w_j \leftarrow c \cdot w_j \begin{cases} e^{-\alpha_i} & C_i(x_j) = y_j \\ e^{\alpha_i} & C_i(x_j) \neq y_j \end{cases}$$

對 縮小
錯 放大

AdaBoost

- The equations of the previous slide are such to minimize the total error. We omit the derivations here.

- Initially, $w_j = \frac{1}{N}$.
- Any intermediate round yields error rate higher than 0.5, weights are reverted back to $\frac{1}{N}$.
- Classification:

$$C^*(x) = \underset{y}{\operatorname{argmax}} \sum_j \alpha_j \cdot \delta[C_j(x) = y]$$

Summary

- Decision tree learning using information gain.
- Learning performance = prediction accuracy measured on test set.
- Cross-validation combats overfitting.
- Bayesian learning is reasonable, but computationally expensive. A common simplification is the MAP learning.
- MAP learning reduces to finding the ML hypothesis when assuming all hypotheses are equally probable.
- Prior that penalizes complexity combat overfitting and results in MDL.
- Bayesian networks provide a natural representation for (causally induced) conditional independence.
- Topology + CPTs = compact representation of joint distribution.
- Bayesian networks are generally easy to construct with human knowledge or other machine learning techniques.