Supervised Learning (Classification)

Example: Loan application data set

ID	Age	Has_job	Own_house	Credit_rating	Class
1	young	false	false	fair	**No**
2	young	false	false	good	**No**
3	young	true	false	good	**Yes**
4	young	true	true	fair	**Yes**
5	young	false	false	fair	**No**
6	middle	false	false	fair	**No**
7	middle	false	false	good	**No**
8	middle	true	true	good	**Yes**
9	middle	false	true	excellent	**Yes**
10	middle	false	true	excellent	**Yes**
11	old	false	true	excellent	**Yes**
12	old	false	true	good	**Yes**
13	old	true	false	good	**Yes**
14	old	true	false	excellent	**Yes**
15	old	false	false	fair	**No**

Supervised learning

- A set of items
  - Each item is characterized by attributes $(a_1, a_2, \ldots, a_k)$, where each $a_i \in A_i$
  - Each item is assigned a class or category $c \in C$
- Given a set of examples, predict c for a new item with attributes $(a'_1, a'_2, \ldots, a'_k)$
- Examples provided are called training data
- Aim is to learn a mathematical model that generalizes the training data
- Classification problem

Basic assumptions

Fundamental assumption of machine learning

- Distribution of training examples is identical to distribution of unseen data

What does it mean to learn from the data?

- Build a model that does better than random answers
  - In the loan data set, always saying Yes would be correct about $9/15$ of the time
- Performance should ideally improve with more training data

Models for supervised learning

- Decision trees
- Naive Bayes classifiers
- Support vector machines

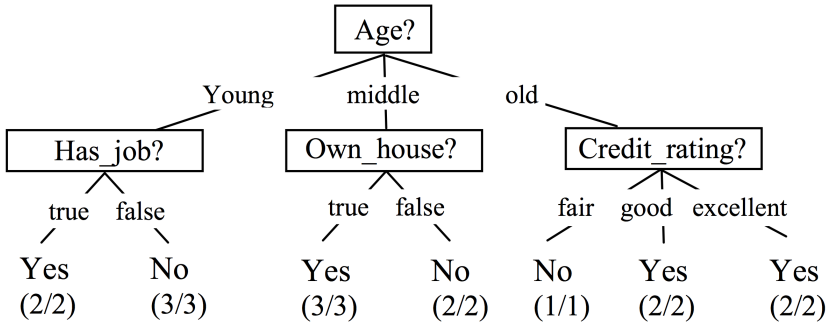
Example: Loan application data set

ID	Age	Has_job	Own_house	Credit_rating	Class
1	young	false	false	fair	**No**
2	young	false	false	good	**No**
3	young	true	false	good	**Yes**
4	young	true	true	fair	**Yes**
5	young	false	false	fair	**No**
6	middle	false	false	fair	**No**
7	middle	false	false	good	**No**
8	middle	true	true	good	**Yes**
9	middle	false	true	excellent	**Yes**
10	middle	false	true	excellent	**Yes**
11	old	false	true	excellent	**Yes**
12	old	false	true	good	**Yes**
13	old	true	false	good	**Yes**
14	old	true	false	excellent	**Yes**
15	old	false	false	fair	**No**

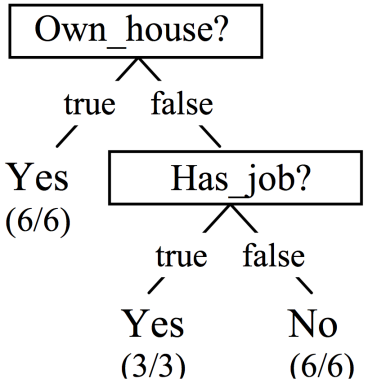
Decision trees

- Examine attributes one at a time
- Branch according to value of attribute
- (Adaptively) choose next attribute to examine

Example: Decision tree

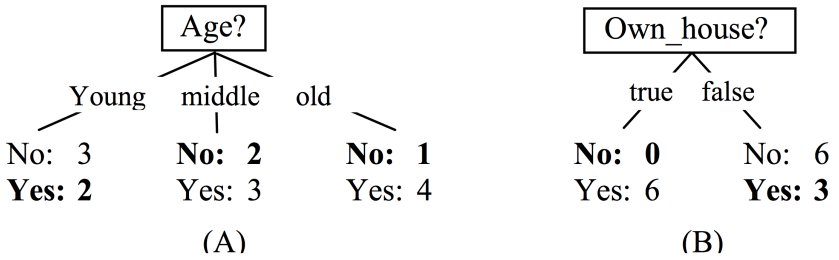


Example: A smaller decision tree



Constructing decision trees: impurity

A reasonable heuristic is to minimize impurity



(B) is a better initial choice than (A) — resulting subgroups are more homogeneous, more pure

Can we quantify this notion?

Constructing decision trees

Smaller trees are better

- Tend to be more accurate
- Easier to understand
  - Explaining the classification can be important
  - Disease diagnosis from symptoms and test results

Finding the best tree that fits the data is expensive

- NP-complete
- Need heuristics that can incrementally construct a good tree

Constructing decision trees: entropy

Information theory

- Data set D, classes $C = \{c_1, c_2, \ldots, c_k\}$
- $Pr(c_j)$ = fraction of samples in D classified as c_j

$$entropy(D) = -\sum_{i=1}^{k} Pr(c_i) \log_2 Pr(c_i)$$

$$\sum_{i=1}^{k} Pr(c_i) = 1$$

Constructing decision trees: entropy

- Entropy measures disorder or impurity of data
- Entropy is maximized when all entries in C are equiprobable
  - Suppose $C = \{y, n\}$
  - $Pr(y) = 0, Pr(n) = 1 \Rightarrow$ $entropy(D) = -(0 \log 0 + 1 \log 1) = 0$
    - Note We define $0 \log 0 = 0$
  - $Pr(y) = 0.2, Pr(n) = 0.8 \Rightarrow$ $entropy(D) = -(0.2 \log 0.2 + 0.8 \log 0.8) = 0.722$
  - $Pr(y) = Pr(n) = 0.5 \Rightarrow$ $entropy(D) = -(0.5 \log 0.5 + 0.5 \log 0.5) = 1$
- Information theoretically, entropy describes the minimum number of bits required to transmit values

Constructing decision trees: information gain

- Attributes with unique values (Aadhar ID, passport number) produce pure partitions of size 1
- Zero entropy, but not useful as a classifier!
- Normalize the information gain by the entropy of the partition
- $gainratio(D, A) = \dfrac{gain(D, A)}{-\sum_{i=1}^{\ell} \frac{|D_i|}{|D|} \log \frac{|D_i|}{|D|}}$
- Maximize normalized information gain—information gain ratio

Constructing decision trees: information gain

At each step, choose the attribute that maximally reduces the entropy — maximizes information gain

- Current data is D, attribute A has ℓ values
- Choosing A as next node in the tree partitions D as $\{D_1, D_2, \ldots, D_\ell\}$
- Define $entropy_A(D) = \sum_{i=1}^{\ell} \frac{|D_j|}{|D|} entropy(D_j)$
- Define $gain(D, A) = entropy(D) - entropy_A(D)$
- Choose A_j such that $gain(D, A_j)$ is maximized

Constructing decision trees: other issues

Handling numeric attribute A

- Divide into intervals, guided by actual data D
- Often a binary split into {low,high} is enough
- Actual data has fixed values $v_1 < v_2 < \cdots < v_N$ for A
- Choose threshold between v_i, v_{i+1} that maximizes information gain

Evaluating a classifier

- Accuracy What fraction of predictions are correct?
  - Need access to an "oracle" to validate answers
- Classification is often asymmetric
  - Suppose 1% of email traffic constitutes phishing
  - An email filter that always says "No" is 99% accurate, but totally useless!
- Note: Conventional to assume that "Yes" is the minority answer
- Need a finer classification of correct predictions and errors

Evaluating a classifier . . .

Confusion matrix

	Classified positive	Classified negative
Actual Positive	TP	FN
Actual Negative	FP	TN

Precision

What fraction of positive classifications are correct?

$$p = \frac{TP}{TP + FP}$$

Recall

What fraction of actual positive cases are correctly classified?

$$p = \frac{TP}{TP + FN}$$

Evaluating a classifier . . .

	Classified positive	Classified negative
Actual Positive	1	99
Actual Negative	0	1000

Here $p = 1$ but $r = 0.01$

- No functional relationship between p and r
- In practice, they are typically inversely related—increasing p reduces r and vice versa
  - Conservative classifier — higher precision, ignores valid cases
  - Permissive classifier — higher recall, more mistakes