Tutorial 6

By:Aashmeet Kalra

AGENDA

- Candidate Elimination Algorithm
- Example Demo of Candidate Elimination Algorithm
- Decision Trees
- Example Demo of Decision Trees

Concept and Concept Learning

- A Concept is a a subset of objects or events defined over a larger set

  [Example: The concept of a *bird* is the subset of all objects (i.e., the set of all *things* or all *animals*) that belong to the category of bird.]
- Alternatively, a concept is a boolean-valued function defined over this larger set

  [Example: a function defined over all *animals* whose value is *true* for birds and *false* for every other animal].

Concept and Concept Learning

- Given a set of examples labeled as members or non-members of a concept, *concept-learning* consists of automatically inferring the general definition of this concept.

- In other words, *concept-learning* consists of approximating a boolean-valued function from training examples of its input and output.

Terminology and Notation

- The set of items over which the concept is defined is called the set of ***instances*** (denoted by X)
- The concept to be learned is called the *Target Concept* (denoted by c: X--> {0,1})
- The set of ***Training Examples*** is a set of instances, x, along with their target concept value c(x).
- Members of the concept (instances for which c(x)=1) are called ***positive examples***.
- Nonmembers of the concept (instances for which c(x)=0) are called ***negative examples***.
- H represents the set of ***all possible hypotheses***. H is determined by the human designer's choice of a hypothesis representation.
- **The goal of concept-learning is to find a hypothesis h:X --> {0,1} such that h(x)=c(x) for all x in X**

Concept Learning viewed as Search

- Concept Learning can be viewed as the task of searching through a large space of hypotheses implicitly defined by the hypothesis representation.
- Selecting a Hypothesis Representation is an important step since it restricts (or *biases*) the space that can be searched. [For example, the hypothesis "If the air temperature is cold **or** the humidity high then it is a good day for water sports" cannot be expressed in our chosen representation.]

Review of Concepts in Class

General to specific ordering of Hypotheses

- **Definition:** Let *hj* and *hk* be boolean-valued functions defined over X. Then *hj* is **more-general-than-or-equal-to** *hk* iff For all x in X, [(*hk*(x) = 1) --> (*hj*(x)=1)]
- **Example:**
  - *h1* = <Sunny,?,?,Strong,?,?>
  - *h2* = <Sunny,?,?,?,?,?>

Every instance that are classified as positive by *h1* will also be classified as positive by *h2* in our example data set. Therefore *h2* is more general than *h1*.

- We also use the ideas of **“strictly”-more-general-than,** and **more-specific-than**

Find-S, a Maximally Specific Hypothesis Learning Algorithm

- Initialize *h* to the most specific hypothesis in *H*
- For each <u>positive</u> training instance *x*
  - For each attribute constraint *ai* in *h*

    If the constraint *ai* is satisfied by *x*

    then do nothing

    else replace *ai* in h by the next more general constraint that is satisfied by *x*
- Output hypothesis *h*

Although Find-S finds a hypothesis consistent with the training data, it does not indicate whether that is the only one available

Version Spaces and the Candidate-Elimination Algorithm

- **Definition:** A hypothesis *h* is **consistent** with a set of training examples *D* iff *h(x)* = *c(x)* for each example <*x,c(x)*> in *D*.
- **Definition:** The **version space**, denoted *VS_H,D*, with respect to hypothesis space *H* and training examples *D*, is the subset of hypotheses from *H* consistent with the training examples in *D.*
- **NB:** While a Version Space can be exhaustively enumerated, a more compact representation is preferred.

A Compact Representation for Version Spaces

- Instead of enumerating all the hypotheses consistent with a training set, we can represent its **most specific** and **most general** boundaries. The hypotheses included in-between these two boundaries can be generated as needed.
- **Definition:** The **general boundary** *G*, with respect to hypothesis space *H* and training data *D*, is the set of maximally general members of *H* consistent with *D*.
- **Definition:** The **specific boundary** *S*, with respect to hypothesis space *H* and training data *D*, is the set of minimally general (i.e., maximally specific) members of *H* consistent with *D*.

Candidate Elimination Algorithm

- The candidate-Elimination algorithm computes the version space containing all (and only those) hypotheses from *H* that are consistent with an observed sequence of training examples.

Example 1

- **Learning the concept of "Japanese Economy Car“**

Features:

Country of Origin

Manufacturer

Color

Decade

Type

Origin	Manufacturer	Color	Decade	Type	Example Type
Japan	Honda	Blue	1980	Economy	Positive
Japan	Toyota	Green	1970	Sports	Negative
Japan	Toyota	Blue	1990	Economy	Positive
USA	Chrysler	Red	1980	Economy	Negative
Japan	Honda	White	1980	Economy	Positive
Japan	Toyota	Green	1980	Economy	Positive
Japan	Honda	Red	1990	Economy	Negative

Positive Example 1
(Japan, Honda, Blue, 1980, Economy)

- Initialize G to a singleton set that includes everything.

G = { (?, ?, ?, ?, ?) }

- Initialize S to a singleton set that includes the first positive example.

S = { (Japan, Honda, Blue, 1980, Economy) }

Negative Example 2
(Japan, Toyota, Green, 1970, Sports)

- Specialize G to exclude the negative example.

- G =
  { (?, Honda, ?, ?, ?),
  (?, ?, Blue, ?, ?),
  (?, ?, ?, 1980, ?),
  (?, ?, ?, ?, Economy) } S = { (Japan, Honda, Blue, 1980, Economy) }

Positive Example 3(Japan, Toyota, Blue, 1990, Economy)

- Prune G to exclude descriptions inconsistent with the positive example.

G =
  { (?, ?, Blue, ?, ?),
  (?, ?, ?, ?, Economy) }

- Generalize S to include the positive example.

S = { (Japan, ?, Blue, ?, Economy) }

Negative Example (USA, Chrysler, Red, 1980, Economy)

- Specialize G to exclude the negative example (but stay consistent with S)

G =

{ (?, ?, Blue, ?, ?),

(Japan, ?, ?, ?, Economy) }

S = { (Japan, ?, Blue, ?, Economy) }

Positive Example(Japan, Honda, White, 1980, Economy)

- Prune G to exclude descriptions inconsistent with positive example.

G = { (Japan, ?, ?, ?, Economy) }

- Generalize S to include positive example.

S = { (Japan, ?, ?, ?, Economy) }

Positive Example: (Japan, Toyota, Green, 1980, Economy)

- New example is consistent with version-space, so no change is made.

G = { (Japan, ?, ?, ?, Economy) }

S = { (Japan, ?, ?, ?, Economy) }

Negative Example: (Japan, Honda, Red, 1990, Economy)

- Example is inconsistent with the version-space.

  G cannot be specialized.
  S cannot be generalized.

- The version space **collapses**.
- <u>Conclusion:</u> No conjunctive hypothesis is consistent with the data set.

Remarks on Version Spaces and Candidate Elimination

- The version space learned by the Candidate-Elimination Algorithm will converge toward the hypothesis that correctly describes the target concept provided: (1) There are no errors in the training examples; (2) There is some hypothesis in *H* that correctly describes the target concept.
- Convergence can be speeded up by presenting the data in a strategic order. The best examples are those that satisfy exactly half of the hypotheses in the current version space.
- Version-Spaces can be used to assign certainty scores to the classification of new examples

Decision Trees

- Consider this Decision-making process:
  WHAT TO DO THIS WEEKEND?
- If my parents are visiting
  - We'll go to the cinema
- If not
  - Then, if it's sunny I'll play tennis
  - But if it's windy and I'm rich, I'll go shopping
  - If it's windy and I'm poor, I'll go to the cinema
  - If it's rainy, I'll stay in

From Decision Trees to Logic

- Decision trees can be written as
  - Horn clauses in first order logic
- Read from the root to every tip
  - If this and this and this … and this, then do this

- In our example:
  - If no_parents and sunny_day, then play_tennis
  - no_parents $\wedge$ sunny_day $\rightarrow$ play_tennis

- Decision tree can be seen as rules for performing a categorisation
  - E.g., “what kind of weekend will this be?”
- Remember that we’re learning from examples
  - Not turning thought processes into decision trees
- We need examples put into categories
- We also need attributes for the examples
  - Attributes describe examples (background knowledge)
  - Each attribute takes only a finite set of values

Entropy

- From Tom Mitchell's book:
  - "In order to define information gain precisely, we begin by defining a measure commonly used in information theory, called entropy that characterizes the (im)purity of an arbitrary collection of examples"
- Want a notion of impurity in data
- Imagine a set of boxes and balls in them
- If all balls are in one box
  - This is nicely ordered – so scores low for entropy
- Calculate entropy by summing over all boxes
  - Boxes with very few in scores low
  - Boxes with almost all examples in scores low

Entropy:Formulae

- Given a set of examples, S
- For examples in a binary categorisation
  - Where p_+ is the proportion of positives
  - And p_- is the proportion of negatives

$$Entropy(S) = -p_+ \, log_2(p_+) - p_- \, log_2(p_-)$$

- For examples in categorisations c_1 to c_n
  - Where p_n is the proportion of examples in c_n

$$Entropy(S) = \sum_{i=1}^{n} -p_i \, log_2(p_i)$$

Information Gain

- Given set of examples S and an attribute A
  - A can take values v_1 … v_m
  - Let S_v = {examples which take value v for attribute A}
- Calculate Gain(S,A)
  - Estimates the reduction in entropy we get if we know the value of attribute A for the examples in S

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

ID3 Algorithm

- Given a set of examples, S
  - Described by a set of attributes A_i
  - Categorised into categories c_j

1. Choose the root node to be attribute A
   - Such that A scores highest for information gain
     - Relative to S, i.e., gain(S,A) is the highest over all attributes
2. For each value v that A can take
   - Draw a branch and label each with corresponding v
     - Then see the options in the next slide!

ID3 (Continued)

- For each branch you've just drawn (for value v)
  - If S_v only contains examples in category c
    - Then put that category as a leaf node in the tree
  - If S_v is empty
    - Then find the default category (which contains the most examples from S)
      - Put this default category as a leaf node in the tree
  - Otherwise
    - Remove A from attributes which can be put into nodes
    - Replace S with S_v
    - Find new attribute A scoring best for Gain(S, A)
    - Start again at part 2
- Make sure you replace S with S_v

Example

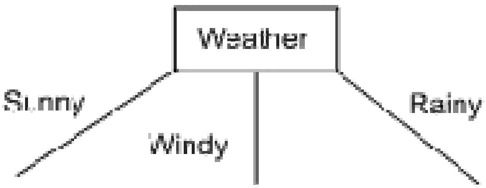
Weekend	Weather	Parents	Money	Decision (Category)
W1	Sunny	Yes	Rich	Cinema
W2	Sunny	No	Rich	Tennis
W3	Windy	Yes	Rich	Cinema
W4	Rainy	Yes	Poor	Cinema
W5	Rainy	No	Rich	Stay in
W6	Rainy	Yes	Poor	Cinema
W7	Windy	No	Poor	Cinema
W8	Windy	No	Rich	Shopping
W9	Windy	Yes	Rich	Cinema
W10	Sunny	No	Rich	Tennis

Information Gain

- S = {W1,W2,…,W10}
- Firstly, we need to calculate:
  - Entropy(S) = … = 1.571
- Next, we need to calculate information gain
  - For all the attributes we currently have available
    - (which is all of them at the moment)
  - Gain(S, weather) = … = 0.7
  - Gain(S, parents) = … = 0.61
  - Gain(S, money) = … = 0.2816
- Hence, the weather is the first attribute to split on
  - Because this gives us the biggest information gain

Top of the Tree

- So, this is the top of our tree:



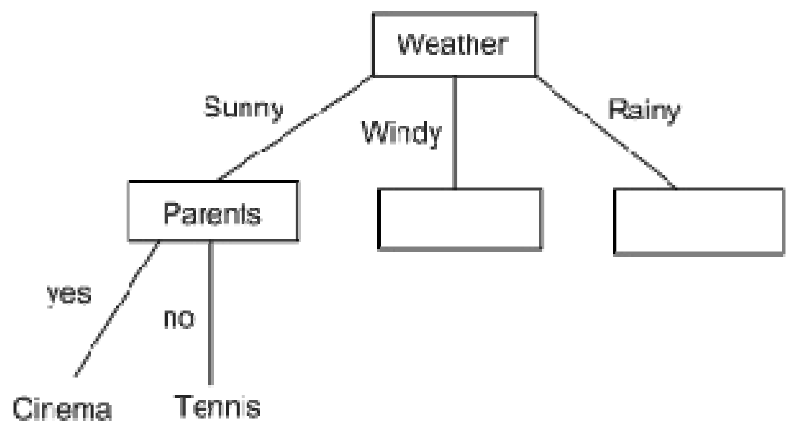
- Now, we look at each branch in turn
  - In particular, we look at the examples with the attribute prescribed by the branch
- S_{sunny} = {W1,W2,W10}
  - Categorisations are cinema, tennis and tennis for W1,W2 and W10
  - What does the algorithm say?
    - Set is neither empty, nor a single category
    - So we have to replace S by S_{sunny} and start again

Working with S_{sunny}

- Need to choose a new attribute to split on
  - Cannot be weather, of course – we've already had that
- So, calculate information gain again:
  - Gain(S_{sunny}, parents) = … = 0.918
  - Gain(S_{sunny}, money) = … = 0
- Hence we choose to split on parents

Getting to the leaf nodes

- If it’s sunny and the parents have turned up
  – Then, looking at the table in previous slide
    - There’s only one answer: go to cinema
- If it’s sunny and the parents haven’t turned up
  – Then, again, there’s only one answer: play tennis
- Hence our decision tree looks like this:



Avoid Overfitting

- Decision trees can be learned to perfectly fit the data given
  - This is probably overfitting
    - The answer is a memorisation, rather than generalisation
- Avoidance method 1:
  - Stop growing the tree before it reaches perfection
- Avoidance method 2:
  - Grow to perfection, then prune it back afterwards
    - Most useful of two methods in practice

Appropriate Problems for Decision Tree learning

- From Tom Mitchell's book:
  - Background concepts describe examples in terms of attribute-value pairs, values are always finite in number
  - Concept to be learned (target function)
    - Has discrete values
  - Disjunctive descriptions might be required in the answer
- Decision tree algorithms are fairly robust to errors
  - In the actual classifications
  - In the attribute-value pairs
  - In missing information