

## COMSM0301: Lecture 6 "Machine Learning Representations"

Oliver Ray  
([oray@cs.bris.ac.uk](mailto:oray@cs.bris.ac.uk))

Department of Computer Science  
University of Bristol

28<sup>th</sup> January, 2010

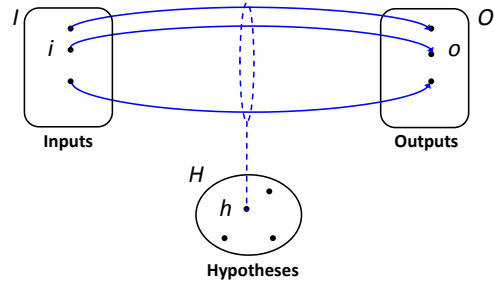
## Learning as Function Approximation

- **Supervised** learning can be seen as the task of **approximating** a partially unknown function (from a space of inputs  $I$  to a space of outputs  $O$  w.r.t. a set of examples  $E$ , a hypothesis space  $H$ , and a real-valued loss function  $\lambda$ )
- **Given**
  - a set  $I$  (inputs)
  - a set  $O$  (outputs)
  - a set  $E \subseteq I \times O$  (examples)
  - a set  $H \subseteq O^I$  (hypothesis space)
  - a function  $\lambda: O^I \times 2^{I \times O} \rightarrow \mathbb{R}$  (loss function)
- **Find** a function  $h \in H$  (hypothesis)
- **Such that**  $\lambda(h, E) \approx \min_{h \in H} \lambda(h, E)$  (approximation)

## Inputs and Outputs viewed as Sets



## Hypotheses viewed as Functions



## Agreement and Disagreement

- For each example  $e=(i,o)$  and each hypothesis  $h$ , we say that  $h$  **agrees** with  $e$ , denoted  $\text{agrees}(h,e)$ , if and only if the predicted output value  $h(i)$  coincides with the given output value  $o$   
i.e:  $\text{agrees}(h,e)$  iff  $h(i) = o$  where  $e=(i,o)$
- If there is a metric defined on the output space, then any disagreement between a hypothesis  $h$  and an example  $e=(i,o)$  can be quantified by a natural **error** measure  
i.e:  $\text{error}(h,e) = o - h(i)$  where  $e=(i,o)$
- If there is no metric defined on the output space, then we can define the error as 0 if  $h$  agrees with  $e$ , and 1 otherwise  
i.e:  $\text{error}(h,e) = 0$  if  $\text{agrees}(h,e)$   
 $= 1$  otherwise

## Function Space Intuition

- The target values  $O$  depend on the learning task: e.g. Booleans for concept learning, reals for regression, etc.
- The instance descriptions  $I$  depend on the learning formalism: e.g. feature vectors for traditional Attribute-Value learners, etc.
- The loss function  $\lambda$  is usually parametric on (some test/training split of) the examples  $E$ ; typically, it will simply aggregate the error measure  $\text{error}(h,e)$  over all examples  $e$  in  $E$

$$\text{e.g.: } \lambda(h, E) = \sum_{e \in E} (\text{error}(h, e))^2 \quad \text{for squared error}$$

$$\text{e.g.: } \lambda(h, E) = \sum_{e \in E} |\text{error}(h, e)| / |E| \quad \text{for empirical risk}$$

## A Hierarchy of Representations

- In practice, supervised learning problems can be classified according to various **restrictions** are imposed on the input, output and hypothesis spaces, I, O and H
- Some of the most common **learning representations** include
  - Boolean Logic Learning (BLL)
  - Attribute-Value Learning (AVL)
  - Multiple-Instance/Tuple/Join Learning (MIL/MTL/MJL)
  - Multi-Relational Learning (MRL)
  - Inductive Logic Programming (ILP)
- Any sets I, O and H which satisfy the conditions of a particular representation R will be called a **learning setting (of R)**

## A Hierarchy of Applications

- **Boolean Logic Learning (BLL)**  
Market Basket Analysis
- **Attribute-Value Learning (AVL)**  
Classical Machine Learning
- **Multiple-Instance Learning (MIL)**  
Image Classification – image as a set of objects  
Drug Activity – molecule as a set of conformations
- **Multi-Relational Learning (MRL)**  
Drug Activity – molecular structures as graphs  
Bioinformatics – metabolic/signalling/genetic networks as graphs  
Social Network Analysis – social interactions as graphs
- **Inductive Logic Programming (ILP)**  
Program Synthesis  
Grammar Induction  
Spatio-Temporal Reasoning

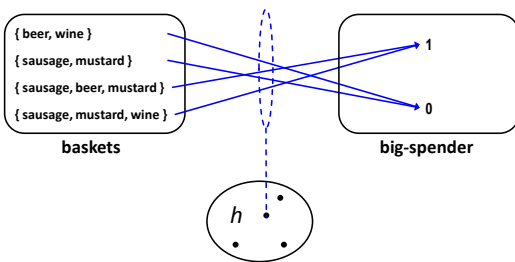
## Boolean Logic Learning

- Inputs are **item sets (or interpretations)** over some set of propositions  $Prop = \{ p_1, \dots, p_n \}$  i.e.  $I \subseteq 2^{Prop}$
- If outputs are Booleans, then hypotheses are n-ary Boolean functions, which are often represented using **monomials** (conjunctions of literals) or **clauses** (disjunctions of literals)
- In **Conjunctive Normal Form** (a.k.a Product of Sums form), hypotheses are conjunctions of clauses
- In **Disjunctive Normal Form** (a.k.a Sum of Products form), hypotheses are disjunctions of monomials

## Market Basket Example

#basket	Sausage	Mustard	Beer	Wine	big-spender
1	0	0	1	1	0
2	1	1	0	0	0
3	1	1	1	0	1
4	1	1	1	1	1
...	...	...	...	...	...

## Boolean Logic Illustration



e.g.  $sausage \wedge mustard \wedge beer \rightarrow big\text{-spender}$   
 $sausage \wedge mustard \wedge wine \rightarrow big\text{-spender}$

## Attribute-Value Learning

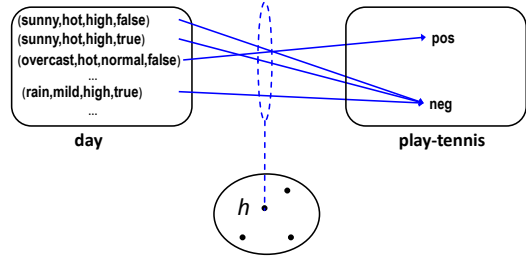
- Inputs are **feature vectors** over some attribute domains  $A_1, \dots, A_n$  i.e.  $I \subseteq A_1 \times \dots \times A_n$
- Hypotheses are computed by standard machine learning methods e.g. Linear Regression, Polynomial Regression, Artificial Neural Networks, Radial Basis Functions, Support Vector Machines, Decision Trees, Genetic Algorithms, Bayesian Networks, etc.
- Note that, if we restrict  $A_i = \{0,1\}$  for all  $1 \leq i \leq n$ , then AVL collapses to BLL

### Play-Tennis Example

#day	Outlook	Temperature	Humidity	Wind	play-tennis
1	sunny	hot	high	false	neg
2	sunny	hot	high	true	neg
3	overcast	hot	high	false	pos
4	rain	mild	high	false	pos
5	rain	cool	normal	false	pos
6	rain	cool	normal	true	neg
7	overcast	cool	normal	true	pos
8	sunny	mild	high	false	neg
9	sunny	cool	normal	false	pos
10	rain	mild	normal	false	pos
11	sunny	mild	normal	true	pos
12	overcast	mild	high	true	pos
13	overcast	hot	normal	false	pos
14	rain	mild	high	true	neg

from R. Quinlan, 1986

### Attribute-Value Illustration



e.g. day(overcast,\_,\_) → play-tennis

### Multiple-Instance Learning

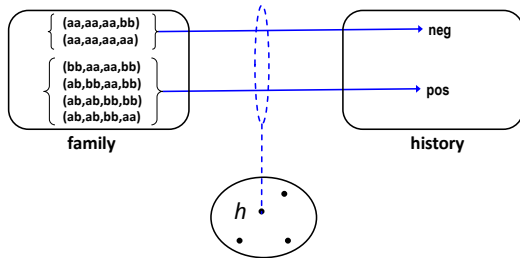
- Inputs are **feature vector bags** i.e.  $I \subseteq 2^{A_1 \times \dots \times A_n}$
- In the **classical** multiple instance problem, it is required that  $h(i) = o$  iff  $\exists x \in i$  so that each bag label is determined by a (unknown) bag member. But this is relaxed in **generalised** multiple instance learning problems such as **Multi-tuple learning**, where  $h(i) = o$  iff  $\exists g(h\{x_1\}, \dots, h\{x_m\})$  for some  $\{x_1, \dots, x_m\} \subseteq i$  so that each bag label is a function of some (unknown) set of bag members
- Multi-join learning**, where there are no restrictions on  $h$
- Hypotheses are computed by methods such as axis parallel hyper rectangles, diverse density, k-citation neural nets, etc.
- Note that, if  $|i|=1$  for all  $i \in I$ , then MIL collapses to AVL
- Note also, missing values often lead to MIL problems

### Hereditary Disease Example

#family	Gene1	Gene2	Gene3	Gene4	history
1	aa	aa	aa	bb	neg
	aa	aa	aa	aa	
2	bb	aa	aa	bb	pos
	ab	bb	aa	bb	
	ab	ab	bb	bb	
3	bb	ab	bb	aa	neg
	aa	bb	aa	bb	
	aa	ab	bb	bb	
4	ab	bb	bb	bb	pos
	aa	bb	bb	aa	
	bb	bb	aa	aa	
	bb	aa	bb	bb	
...	...	...	...	...	...

n.b. assumes that a family history means at least one carrier is sequenced

### Multiple-Instance Illustration



e.g. person(ab,bb,\_,\_) → history (pos) % multi-instance  
 e.g. person(ab,bb,\_,\_), person(bb,aa,\_,\_) → history % multi-tuple  
 e.g. person(ab,X,\_,\_), person(X,aa,\_,\_) → history % multi-join

### Multi-Relational Learning

- Inputs are **relational databases**

$$i.e. I \subseteq 2^{A_{1,1} \times \dots \times A_{1,n(1)} \times \dots \times A_{m,1} \times \dots \times A_{m,n(m)}}$$

- relations can represent arbitrary structures e.g. sets, lists, trees, graphs, etc.
- note that, if we restrict  $m=1$ , then MRL collapses to MIL

### Bongard Example (#47)

pos

neg

<http://www.foundalis.com/res/bps/bpsidx.htm>

### Multi-Relational Illustration

e.g.  $\text{shape}(T, \text{triangle}), \text{shape}(C, \text{circle}), \text{inside}(T, C) \rightarrow \text{class}(\text{pos})$

### Inductive Logic Programming

- Inputs are **logic programs** over some set of clauses i.e.  $I \subseteq 2^{\text{Clause}}$
- Note that, if we restrict inputs to sets of function-free ground facts, then ILP collapses to MRL (as each predicate is essentially a database table)
 

e.g.  $\{ \text{shape}(1, \text{triangle}), \text{shape}(2, \text{circle}), \text{inside}(1, 2) \}$

we will call this an **atom-based representation**
- Note also, if we restrict inputs to singleton ground facts whose arguments, are lists of fixed length function-free tuples, then, once again, ILP collapses to MRL (as each argument is essentially a database table)
 

e.g.  $\{ \text{image}([1, \text{triangle}], [2, \text{circle}]), [(1, 2)] \}$

we will call this a **term-based representation**

### Inductive Concept Learning

- If the output space  $O$  is Boolean, then each hypothesis  $h$  can be equivalently represented by a relation  $h$  (a.k.a **coverage relation**) over the input space  $I$  such that  $h(i)$  iff  $h(i)=\text{true}$
- If hypotheses are logic programs, then logical methods can be used to compute hypotheses by defining coverage in terms of entailment and satisfaction. Two approaches are very common
 

**Learning from entailment (LE)**  $h(i)$  iff  $h \models i$   
where  $i$  is a ground atom

**Learning from interpretations (LI)**  $h(i)$  iff  $i \models h$   
where  $i$  is a set of ground atoms
- In LE, examples are often defined as labelled clauses; whence  $i$  will be the skolemised head of clause whose body atoms are added to  $h$
- In LI,  $i$  can be equivalently regarded as a (Herbrand) interpretation

### Coverage Intuition

### Background Knowledge

- More generally, we can use **relative entailment**  $\models_B$  instead of simple entailment  $\models$  where  $B$  is a logic program usually called a **background theory** and  $A \models_B C$  iff  $A \cup B \models C$
- The background theory defines **high level concepts** (provided by domain experts) in terms of which hypotheses should be expressed (and with which they should be consistent)
- The background theory allows to leave some **implicit facts** out of the example descriptions e.g. if we know  $\text{man}(\text{socrates})$ , we can assume  $\text{mortal}(\text{socrates})$  given  $\text{mortal}(X) \leftarrow \text{man}(X)$
- Common ILP methods include Inverse Resolution, Inverse Implication, Relative Least General Generalisation, Inverse Entailment, ...

## Idealised Learning From Interpretations

- Given
 

a logic program B	(Background Knowledge)
a set of interpretations $E^+$	(Positive Examples)
a set of interpretations $E^-$	(Negative Examples)
a set of logic programs H	(Hypothesis Space)
- Find
 

a logic program $h \in H$	(hypothesis)
---------------------------	--------------
- Such that
 

$B \cup e^+ \models h$	for all $e^+ \in E^+$
$B \cup e^- \not\models h$	for all $e^- \in E^-$
$B \models h$	

## Idealised Learning From Entailment

- Given
 

a logic program B	(Background Knowledge)
a set of ground atoms $E^+$	(Positive Examples)
a set of ground atoms $E^-$	(Negative Examples)
a set of logic programs H	(Hypothesis Space)
- Find
 

a logic program $h \in H$	(hypothesis)
---------------------------	--------------
- Such that
 

$B \cup h \models e^+$	for all $e^+ \in E^+$
$B \cup h \not\models e^-$	for all $e^- \in E^-$
$B \models \neg h$	

## List Sorting Example

```

%%% Hypothesis Space %%%
:- modeb(1, qsort(+,-,-))?.
:- modeb(1, part(+,-,-))?.
:- modeb(1, append(+,+|+,-))?.
:- modeh(1, qsort(+|+,-,-))?.

%%% Background knowledge %%%
part(X,[],[]).
part(X,[X|Tail],List1,List2) :- part(X,Tail,List1,List2).
part(X,[Head|Tail],[Head|Tail1],List2) :- Head < X, part(X,Tail,Tail1,List2).
part(X,[Head|Tail],List1,[Head|Tail2]) :- Head > X, part(X,Tail,List1,Tail2).
append([],List,List).
append([Head|Tail],List1,[Head|List2]) :- append(Tail,List1,List2).

%%% Positive examples %%%
qsort([],[]).      qsort([1],[1]).      qsort([5],[5]).      qsort([2,0,8,9],[0,2,8,9]).
qsort([2],[2]).    qsort([4],[4]).      qsort([6,4],[4,6]).  qsort([3,1,6,8],[1,3,6,8]).

%%% Negative examples %%%
:- qsort([0,2,1],[0,2,1]).      :- qsort([0,2,1],[0,1]).      :- qsort([1,0,2],[2,0,1]).
:- qsort([1,0,2],[2,1,0]).      :- qsort([1,2,0],[1,0,2]).      :- qsort([0,2,1],[2,1,0]).

```