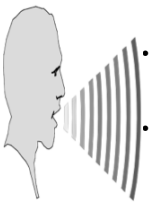
Subjugator 2006

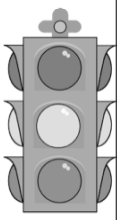
Announcements



- Today's handouts on www:
  - > Outline Class 6
  - > Homework 3
- Web Site
  - > Today's handouts
  - > www.mil.ufl.edu/eel5840
  - > Software and Notes
  - > XLISP Documentation

Subjugator 2006

Today's Menu



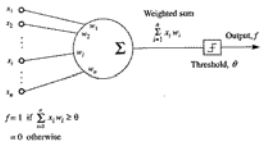
- Example of a Classical AI Production System
  {See slides 14-22 from class #3}
  - > Irrevocable Control Strategy
  - > Tentative Control Strategy
- Architectures for the Implementation of Action Functions
  - > B. State Machines
  - > C. Artificial Neural Networks
  - > D. Subsumption Architecture
- Final Thoughts on Stimulus-Response (*SR*) Agents - Chapter 2

- LISP
  - > More on User-Defined Functions
    - – DEFUN
    - – COND
    - – FUNCTION-LAMBDA-EXPRESSION
  - > LISP Chapter 5 Procedure Abstraction & Recursion

Subjugator 2006

Perception and Action

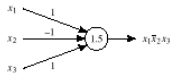
- State Machines: Implementation of Boolean (action) functions using a connected network of logical gates AND,OR,NOR, etc.)
- Networks: Implementation of action functions using a connected network of threshold units or other elements that compute a nonlinear function of a weighted sum of their inputs. One such element is the *threshold logic unit* or TLU for short.



Boolean functions implementable by a TLU are called *linearly-separable functions.* (A TLU separates the space of input vectors into an above-threshold response from below-threshold response by a linear surface—called a *hyperplane* in *n* dimensions.)
Not all Boolean functions are linearly separable—however a monomial or any clause is linearly separable.

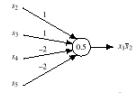
Subjugator 2006

Perception and Action



Example: Let $f=x_1/x_2x_3$

x_1/x_2x_3	f	$\sum$	TLU
0 0 0	0	0	0
0 0 1	0	1	0
0 1 0	0	-1	0
0 1 1	0	0	0
1 0 0	0	1	0
1 0 1	1	2	1
1 1 0	0	0	0
1 1 1	0	1	0



Example: Let $b\text{-}f_4=x_1/x_2$
$= (s_2+s_3)/(s_4+s_5)=(s_2+s_3)/s_4/s_5$

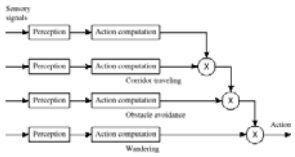
s_2 $s_3/s_4/s_5$	b-f	$\sum$	TLU	s_2 $s_3/s_4/s_5$	b-f	$\sum$	TLU
0 0 0 0	0	0	0	1 0 0 0	1	1	1
0 0 0 1	0	-2	0	1 0 0 1	0	-1	0
0 0 1 0	0	-2	0	1 0 1 0	0	-1	0
0 0 1 1	0	-4	0	1 0 1 1	0	-3	0
0 1 0 0	1	1	1	1 1 0 0	1	2	1
0 1 0 1	0	-1	0	1 1 0 1	0	0	0
0 1 1 0	0	-1	0	1 1 1 0	0	0	0
0 1 1 1	0	-3	0	1 1 1 1	0	-2	0

EEL5840: Elements of Machine Intelligence

Subjugator 2006

Perception and Action

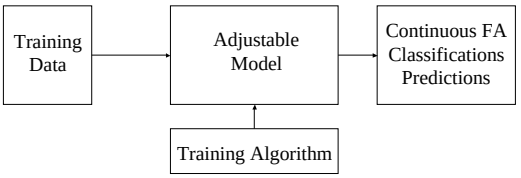
- The Subsumption Architecture
  - > An agent's behavior is controlled by a number of "behavior modules." Each module receives sensory information directly from the world. If the sensory inputs satisfy a precondition specific to that module, then a certain behavior, also specific to that module, is executed. One behavior module can subsume another.
  - > As contrasted with much other work in AI, these machines do not depend on complex internal representations of their environments or on reasoning about them.



EEL5840: Elements of Machine Intelligence

Subjugator 2006

Neural Networks



- Neural Networks (also known as Artificial Neural Networks or ANNs for short)
  - – You need this framework to model processes that cannot be represented as analytical models, e.g., human actions, computer vision, non-linear control, the stock market...

EEL5840: Elements of Machine Intelligence

Subjugator 2006

from LISP Lab 1

Predicate Functions:

(*atom* sex)	t if sex is an atom
(*null* sex)	t if sex is nil or ()
(*eq* sex1 sex2)	t if sex1==sex2 (identical)
(*equal* sex1 sex2)	t if sex1=sex2
(*zerop* sex1)	t if sex=0
(*numberp* sex)	t if sex is a number
(*symbolp* sex)	t if sex is a symbolic atom
(*listp* sex)	t if sex is a list
(*member* sex lis)	nil if sex is not a member of lis

EEL5840: Elements of Machine Intelligence

Subjugator 2006

LISP Lab 2

Predicate Functions:

(*integerp* sex)	t if sex is an integer
(*floatp* sex)	t if sex is a floating point number
(*and* sex1 sex2)	t if sex1 and sex2 are both true
(*or* sex1 sex2)	t if sex1 or sex2 or both are true
(*not* pred)	nil if pred=t or non-nil, t if pred=nil

User-Defined Functions:

(*cond* <(clause$_1$) >…<(clause$_n$)>) returns the evaluated action from the 1st non-nil predicate or nil where (clause$_i$) ⇒ (predicate-form action-form)

(*defun fname* (argument-list) <(forms)>)

(*pprint* (*function-lambda-expression* #'fname))

Recursive Function Definitions

EEL5840: Elements of Machine Intelligence

Subjugator 2006

See LISP Notes 1

The End!