Overview

- Project 2 (pages 2–5)
- Learning
- Background: the human nervous system
- Overview of neural network models
- Application of neural networks
- Homework 2 (pages 18–19; due by 4/8 in class)

Project 2: Resolution Theorem Prover

Write a resolution theorem prover for first-order logic in Lisp.

1. Representation: use a list of clauses, where each clause is in the form (see slide23, page 5):
   `( <clause-num> <list-of-pos-lit> <list-of-neg-lit> )`
   Initially, the list will contain the premise clauses, followed by clauses derived from the negated conclusion. As resolvents are generated from these clauses, they are added to the end of the list.

   Each literal will be of the form: (predicate term ... term) . An atomic term is considered to be a variable; a non-atomic term must be a function of the form: (function term ... term) . A constant is represented as a function of no arguments.

   For simplicity, you may assume that the sets of symbols used as predicate names, function names, and variable names are disjoint in any given problem, i.e. you can do all the necessary pre-processing by hand before feeding in the problem to the prover.

2. Print out each input clause and each clause that is produced by resolution; number the printed clauses and show the numbers of the clauses from which they are derived. (Remember that more than one resolution of a given pair of clauses may be possible. However, we will not worry about resolving on factors of clauses.)
3. A unification algorithm is provided in the file sunify.lsp. The clauses must first be rewritten so that they have no variables in common. If you need to generate new symbols, use `(intern (symbol-name (gensym)))`.
4. Use the two-pointer method to select pairs of clauses for resolution; initialize the pointers so that you will be using the set-of-support strategy. This can be done in the following way:
   (a) Initialize: Set the “inner loop” pointer to the front of the list of clauses. Set the “outer loop” pointer to the first clause resulting from the negated conclusion. (**continued in the next slide**)

4. (b) Resolve: If the clauses denoted by the two pointers can be resolved, produce the resolvent. Add the resolvent to the end of the list of clauses and print it out. (Note: There may be more than one possible resolvent from a pair of clauses. It may be advantageous to check whether the resolvent is already present in the list of clauses; if so, it need not be added.) If the resolvent is empty ($\mathbf{F}$), stop; the theorem is proved.

   (c) Step: Move the “inner loop” pointer forward one clause. If the “inner loop” pointer has not reached the “outer loop” pointer, go to the Resolve step. Otherwise, reset the “inner loop” pointer to the front of the list of clauses and move the “outer loop” pointer forward one clause. If the “outer loop” pointer goes beyond the last clause in the list, stop; the theorem cannot be proved. The “two-pointer method” is a breadth-first method that will generate many duplicate clauses.

5. Run your prover on these three problems and summarize the results:
   (a) Howling hound (from the handout given out earlier)
   (b) Drug dealer and customs official (from the class)
   (c) Coyote and roadrunner (from hw2)
6. Name and run your prover like this: `(prover *problem* 7)`, where `*problem*` is a global variable pointing to the list of clauses (e.g. `*howl-hound*`), and 7 is the first clause of the negated conclusion.
7. Required submission material: code (prover.lsp), README (explanation of functions, lisp representation of the problems, and results).
8. Send the program and the README file to the TA.
9. **Due date**: April 29 (Monday) 9am. 5% penalty for each day afterward, up till May 6 (Monday) 9am. Submissions will not be accepted after May 6 9am. **End of Project Description.**

Example: Howling Hound

```
(defvar *howling*
  '((1 ((howl x)) ((hound x)))
   (2 nil ((have x y) (cat y) (have x z) (mouse z)))
   (3 nil ((ls x) (have x y) (howl y)))
   (4 ((have (john) (a))) nil)
   (5 ((cat (a)) (hound (a))) nil)
   (6 ((mouse (b))) nil)
   (7 ((ls (john))) nil)
   (8 ((have (john) (b))) nil)))

(prover *howling* 6)
```

Learning

- Adapt through interaction with the world: rote memory to developing a complex strategy
- Types of learning:
  1. Supervised learning (dense feedback)
  2. Unsupervised learning (no feedback)
  3. Reinforcement learning (sparse feedback), etc.
- Advantages (two, among many):
  1. Fault tolerance
  2. No need for a complete specification to begin with
- Becoming a central focus of AI.

Neural Networks

Neural networks is one particular form of learning from data.

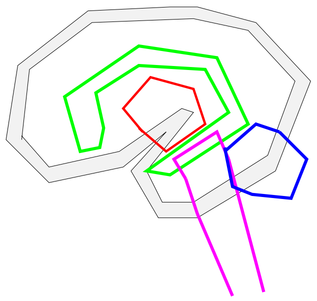
- simple processing elements: named units, or neurons
- connective structure and associated connection weights
- learning: adaptation of connection weights

Neural networks mimic the human (or animal) nervous system.

Many Faces of Neural Networks

- Abstract mathematical/statistical model
- Optimization algorithm
- Pattern recognition algorithm
- Tools for understanding the function of the brain
- Robust engineering application

The Central Nervous System



- Cortex: thin outer sheet where most of the neurons are.
- Sub-cortical nuclei: thalamus, hippocampus, basal ganglia, etc.
- Midbrain, pons, and medulla, connects to the spinal cord.
- Cerebellum (hind brain, or little brain)

Function of the Nervous System

Function of the nervous system:

- Perception
- Cognition
- Motor control
- Regulation of essential bodily functions

The Central Nervous System: Facts [a]

Facts: human neocortex

- Thickness: 1.6mm
- Area: 36cm $\times$ 36cm (about 1.4 ft^2)
- Neurons: 10 billion (10^{10})
- Connections: 60 trillion (6×10^{13}) to 100 trillion
- Connections per neuron: 10^4
- Energy usage per operation: 10^{-16}J (compare to 10^{-6}J in modern computers)

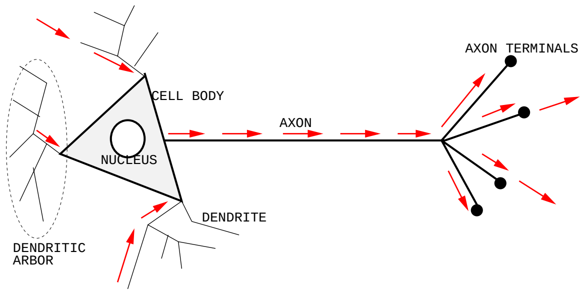
[a] *Neural networks: a comprehensive foundation* by Simon Haykin (1994), and *Foundations of Vision* by Brian Wandell (1995). May slightly differ from those in Russel & Norvig. No need to memorize these figures.

How the Brain Differs from Computers

- Densely connected.
- Massively parallel.
- Highly nonlinear.
- Asynchronous: no central clock.
- Fault tolerant.
- Highly adaptable.
- Creative.

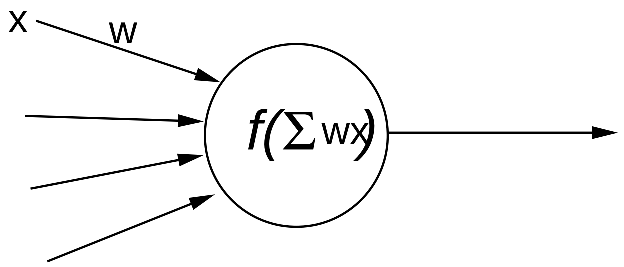
Why are these crucial?

Neurons: Basic Functional Unit of the Brain



- Dendrites receive input from upstream neurons.
- Ions flow in to make the cell positively charged.
- Once a firing threshold is reached, a spike is generated and transmitted along the axon.
- Axon terminals release neurotransmitters to relay the signal to the downstream neurons.

Abstraction of the Neuron in Neural Networks



- Input
- Connection weight
- Transfer function: $f(\cdot)$

Typical transfer functions: step-function or sigmoid.

Key Points

- Types of learning
- Neural networks: basics
- The central nervous system: how it differs from conventional computers.

Next Time

- Perceptrons
- Multilayer perceptrons and backpropagation

Homework 2: Due 4/8/02 (Monday) 12:40pm

1. Unify (if possible) the following pairs of predicates and give the resulting substitutions. b is a constant.

   (a) $P(f(a), g(x))$ and $P(y, y)$

   (b) $P(x, f(x), z)$ and $P(g(y), f(g(b)), y)$

   (c) $P(x_1, g(x_1), x_2, h(x_1, x_2), x_3, k(x_1, x_2, x_3))$ and $P(y_1, y_2, e(y_2), y_3, f(y_2, y_3), y_4)$

2. Determine whether the following clauses have factors. If yes, give the factors.

   (a) $P(x) \vee Q(y) \vee P(f(x))$

   (b) $P(a) \vee P(b) \vee P(x)$

   (c) $P(x) \vee P(a) \vee Q(f(x)) \vee Q(f(a))$

3. Given the following axioms and the conclusion, (1) convert the English sentences into standard form (prenex normal form, then conjunctive normal form) in first-order logic, and (2) using resolution, show that the conclusion is a logical consequence of the axioms.

   1. Every coyote chases some roadrunner.
   2. Every roadrunner who says "beep-beep" is smart.
   3. No coyote catches any smart roadrunner.
   4. Any coyote who chases some roadrunner but does not catch it is frustrated.
   5. **Conclusion** If all roadrunners say "beep-beep", then all coyotes are frustrated.

Homework 2: cont'd

4. Show the following using definitions and axioms in pp.421–423:

   (a) $P(A|B \wedge A) = 1$

   (b) $P(A|B, C) = \frac{P(A,B|C)}{P(B|C)}$

5. Consider exercise 14.3, p.433 in the textbook. How accurate should the test – $P(TestPositive|Disease)$ and $P(\neg TestPositive|\neg Disease)$ – be so that the probability that you actually have the disease if you tested positive, i.e. $P(Disease|TestPositive)$, is over 90%?

6. Using the belief network given in p.439 of the textbook (Figure 15.2), calculate the probability that only John calls after he hears the alarm go off, and when there was actually a burglary, but not an earthquake.

Submit a written solution (on paper) to the instructor before the class (12:40pm) on 4/8/2002 (Monday). Show all of your work. **No extensions nor late submissions are allowed.**