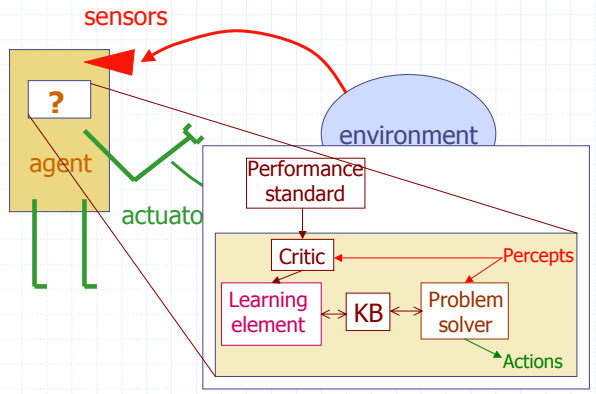
Learning - Decision Trees

Russell and Norvig:
Chapter 18, Sections 18.1 through 18.4

CMSC 421 – Fall 2002

material from Jean-Claude Latombe
and Daphne Koller

Learning Agent



Types of Learning

- Supervised Learning - classification, prediction
- Unsupervised Learning – clustering, segmentation, pattern discovery
- Reinforcement Learning – learning MDPs, online learning

Supervised Learning

- A general framework
- Logic-based/discrete learning:
  - learn a function f(X) → (0,1)
  - Decision trees
  - Version space method
- Probabilistic/Numeric learning
  - learn a function f(X) → **R**
  - Neural nets

Supervised Learning

- Someone gives you a bunch of examples, telling you what each one is
- Eventually, you figure out the mapping from properties (**features**) of the examples and their type

Logic-Inference Formulation

Unlike in the function-learning formulation, h must be a logical sentence, but its inference may benefit from the background knowledge

- Inductive inference: Find h (inductive hypothesis) such that
  - KB and h are consistent
  - KB,h $\models \Delta$

Note that h = Δ is a trivial, but uninteresting solution (data caching)

Rewarded Card Example

- Deck of cards, with each card designated by [r,s], its rank and suit, and some cards "rewarded"
- Background knowledge KB:
  ((r=1) v … v (r=10)) ⇔ NUM(r)
  ((r=J) v (r=Q) v (r=K)) ⇔ FACE(r)
  ((s=S) v (s=C)) ⇔ BLACK(s)
  ((s=D) v (s=H)) ⇔ RED(s)
- Training set Δ:
  REWARD([4,C]) ∧ REWARD([7,C]) ∧ REWARD([2,S]) ∧
  ¬REWARD([5,H]) ∧ ¬REWARD([J,S])

Rewarded Card Example

- Background knowledge KB:
  ((r=1) v … v (r=10)) ⇔ NUM(r)
  ((r=J) v (r=Q) v (r=K)) ⇔ FACE(r)
  ((s=S) v (s=C)) ⇔ BLA
  ((s=D) v (s=H)) ⇔ RED

There are several possible inductive hypotheses

- Training set Δ:
  REWARD([4,C]) ∧ REWARD([7,C]) ∧ REWARD([2,S]) ∧
  ¬REWARD([5,H]) ∧ ¬REWARD([J,S])
- Possible hypothesis:
  h ≡ (NUM(r) ∧ BLACK(s) ⇔ REWARD([r,s]))

Learning a Predicate

- Set E of objects (e.g., cards)
- Goal predicate CONCEPT(x), where x is an object in E, that takes the value True or False (e.g., REWARD)
- Observable predicates A(x), B(X), … (e.g., NUM, RED)
- Training set: values of CONCEPT for some combinations of values of the observable predicates

A Possible Training Set

Ex. #	A	B	C	D	E	CONCEPT
1	True	True	False	True	False	False
2	True	False	False	False	False	True
3	False	False	True	True	True	False
4	True	True	True	False	True	True
5	False	True	True	False	False	False
6	True	True	False	True	True	False
7	False	False	True	False	True	False
8	True	False	True	False	True	True

Note that the training set does not say whether an observable predicate A, …, E is pertinent or not

Learning a Predicate

- Set E of objects (e.g., cards)
- Goal predicate CONCEPT(x), where x is an object in E, that takes the value True or False (e.g., REWARD)
- Observable predicates A(x), B(X), … (e.g., NUM, RED)
- Training set: values of CONCEPT for some combinations of values of the observable predicates
- Find a representation of CONCEPT in the form:

  $\text{CONCEPT}(x) \Leftrightarrow S(A,B, \ldots)$

  where S(A,B,…) is a sentence built with the observable predicates, e.g.:

  $\text{CONCEPT}(x) \Leftrightarrow A(x) \wedge (\neg B(x) \vee C(x))$

Example set

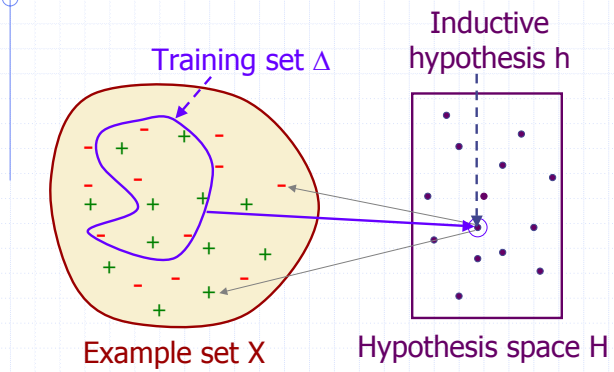
- An example consists of the values of CONCEPT and the observable predicates for some object x
- A example is positive if CONCEPT is True, else it is negative
- The set E of all examples is the example set
- The training set is a subset of E

a small one!

Hypothesis Space

- An hypothesis is any sentence h of the form:
  CONCEPT(x) ⇔ S(A,B, …)
  where S(A,B,…) is a sentence built with the observable predicates
- The set of all hypotheses is called the hypothesis space H
- An hypothesis h agrees with an example if it gives the correct value of CONCEPT

Inductive Learning Scheme



Size of the Hypothesis Space

- n observable predicates
- 2^n entries in truth table
- In the absence of any restriction (bias), there are 2^{2^n} hypotheses to choose from
- n = 6 → $2x10^{19}$ hypotheses!

Multiple Inductive Hypotheses

Rewarded Card Example

- Background knowledge KB:
  ((r=1) ∨ … ∨ (r=10)) ⇔ NUM([r,s])

Need for a system of preferences – called a bias – to compare possible hypotheses

- Possible inductive hypothesis:
  h ≡ (NUM(x) ∧ BLACK(x) ⇔ REWARD(x))

h1 ≡ NUM(x) ∧ BLACK(x) ⇔ REWARD(x)

h2 ≡ BLACK([r,s]) ∧ ¬(r=J) ⇔ REWARD([r,s])

h3 ≡ ([r,s]=[4,C]) ∧ ([r,s]=[7,C]) ∧ [r,s]=[2,S]) ∧
¬ ([r,s]=[5,H]) ∧ ¬ ([r,s]=[J,S]) ⇔ REWARD([r,s])

agree with all the examples in the training set

Keep-It-Simple (KIS) Bias

- Motivation

  If the bias allows only sentences S that are conjunctions of k << n predicates picked from the n observable predicates, then the size of H is $O(n^k)$

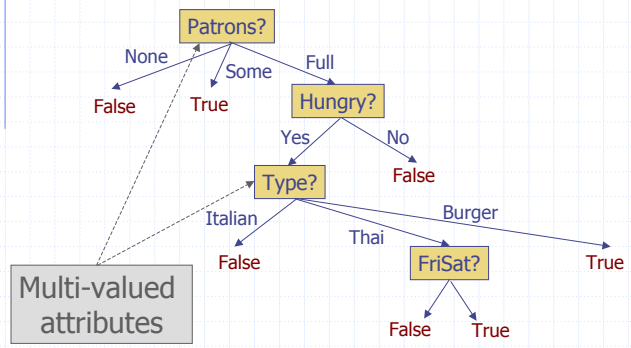
  - Use much fewer observable predicates than suggested by the training set
  - Constrain the learnt predicate, e.g., to use only "high-level" observable predicates such as NUM, FACE, BLACK, and RED and/or to be a conjunction of literals

Predicate-Learning Methods

- Decision tree
- Version space

Decision Tree

WillWait predicate (Russell and Norvig)

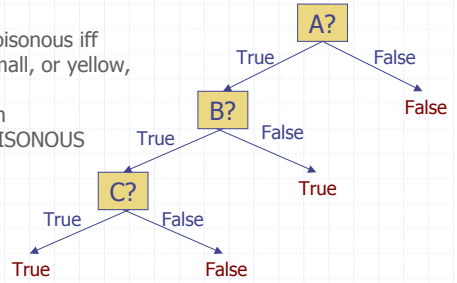


Predicate as a Decision Tree

The predicate CONCEPT(x) $\Leftrightarrow$ A(x) $\wedge$ ($\neg$B(x) v C(x)) can be represented by the following decision tree:

Example:
A mushroom is poisonous iff it is yellow and small, or yellow, big and spotted

- x is a mushroom
- CONCEPT = POISONOUS
- A = YELLOW
- B = BIG
- C = SPOTTED

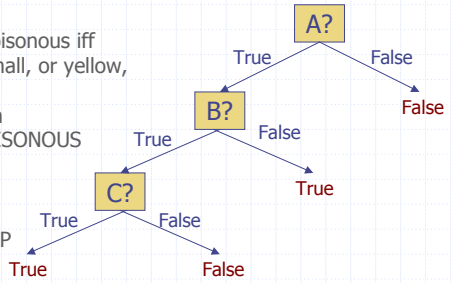


Predicate as a Decision Tree

The predicate CONCEPT(x) ⇔ A(x) ∧ (¬B(x) ∨ C(x)) can be represented by the following decision tree:

Example:
A mushroom is poisonous iff it is yellow and small, or yellow, big and spotted

- x is a mushroom
- CONCEPT = POISONOUS
- A = YELLOW
- B = BIG
- C = SPOTTED
- D = FUNNEL-CAP
- E = BULKY

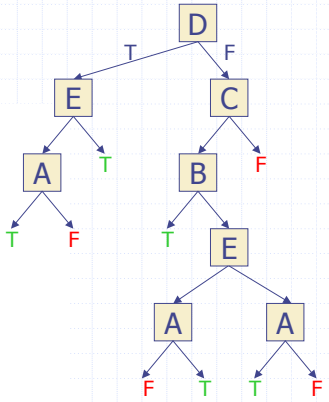


Training Set

Ex. #	A	B	C	D	E	CONCEPT
1	False	False	True	False	True	False
2	False	True	False	False	False	False
3	False	True	True	True	True	False
4	False	False	True	False	False	False
5	False	False	False	True	True	False
6	True	False	True	False	False	True
7	True	False	False	True	False	True
8	True	False	True	False	True	True
9	True	True	True	False	True	True
10	True	True	True	True	True	True
11	True	True	False	False	False	False
12	True	True	False	False	True	False
13	True	False	True	True	True	True

Possible Decision Tree

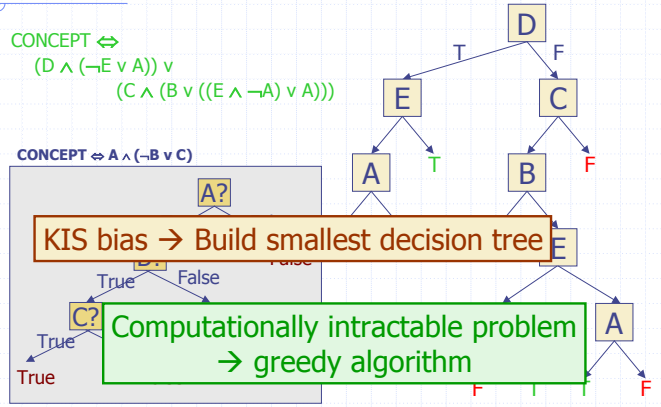
Ex. #	A	B	C	D	E	CONCEPT
1	False	False	True	False	True	False
2	False	True	False	False	False	False
3	False	True	True	True	True	False
4	False	False	True	False	False	False
5	False	False	False	True	True	False
6	True	False	True	False	False	True
7	True	False	False	True	False	True
8	True	False	True	False	True	True
9	True	True	True	False	True	True
10	True	True	True	True	True	True
11	True	True	False	False	False	False
12	True	True	False	False	True	False
13	True	False	True	True	True	True



Possible Decision Tree

CONCEPT ⇔
(D ∧ (¬E ∨ A)) ∨
(C ∧ (B ∨ ((E ∧ ¬A) ∨ A)))

CONCEPT ⇔ A ∧ (¬B ∨ C)



Getting Started

The distribution of the training set is:

True: 6, 7, 8, 9, 10,13
False: 1, 2, 3, 4, 5, 11, 12

Getting Started

The distribution of training set is:

True: 6, 7, 8, 9, 10,13
False: 1, 2, 3, 4, 5, 11, 12

Without testing any observable predicate, we could predict that CONCEPT is False (majority rule) with an estimated probability of error P(E) = 6/13

Getting Started

The distribution of training set is:

True: 6, 7, 8, 9, 10,13
False: 1, 2, 3, 4, 5, 11, 12

Without testing any observable predicate, we could report that CONCEPT is False (majority rule) with an estimated probability of error P(E) = 6/13

Assuming that we will only include one observable predicate in the decision tree, which predicate should we test to minimize the probability of error?

Assume It's A

True: 6, 7, 8, 9, 10, 13
False: 11, 12 1, 2, 3, 4, 5

If we test only A, we will report that CONCEPT is True if A is True (majority rule) and False otherwise

The estimated probability of error is:
Pr(E) = (8/13)x(2/8) + (5/13)x0 = 2/13

Assume It's B

True:	9, 10	6, 7, 8, 13
False:	2, 3, 11, 12	1, 4, 5

If we test only B, we will report that CONCEPT is False if B is True and True otherwise

The estimated probability of error is:
Pr(E) = (6/13)x(2/6) + (7/13)x(3/7) = 5/13

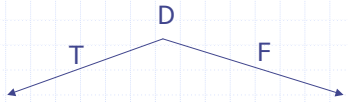
Assume It's C

True:	6, 8, 9, 10, 13	7
False:	1, 3, 4	1, 5, 11, 12

If we test only C, we will report that CONCEPT is True if C is True and False otherwise

The estimated probability of error is:
Pr(E) = (8/13)x(3/8) + (5/13)x(1/5) = 4/13

Assume It's D

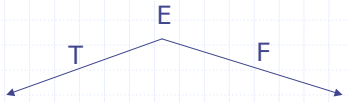


True:	7, 10, 13	6, 8, 9
False:	3, 5	1, 2, 4, 11, 12

If we test only D, we will report that CONCEPT is True if D is True and False otherwise

The estimated probability of error is:
Pr(E) = (5/13)x(2/5) + (8/13)x(3/8) = 5/13

Assume It's E



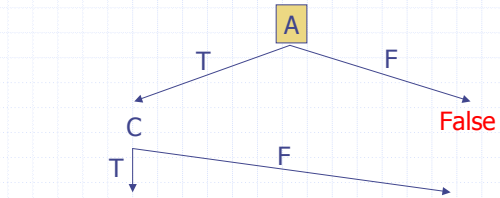
True:	8, 9, 10, 13	6, 7
False:	1, 3, 5, 12	2, 4, 11

So, the best predicate to test is A

If w
independent of the outcome

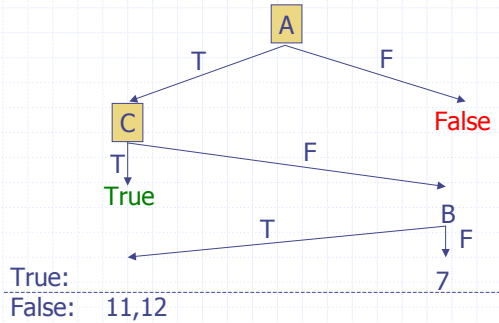
The estimated probability of error is unchanged:
Pr(E) = (8/13)x(4/8) + (5/13)x(2/5) = 6/13

Choice of Second Predicate

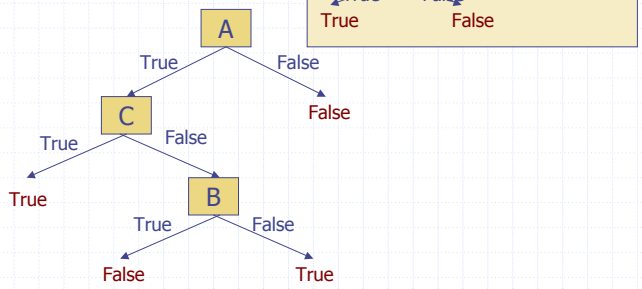


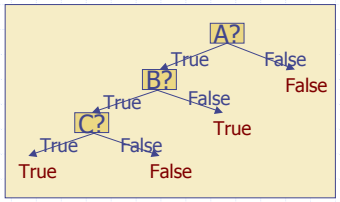
The majority rule gives the probability of error Pr(E) = 1/8

Choice of Third Predicate



Final Tree





L ≡ CONCEPT ⇔ A ∧ (C ∨ ¬B)

Learning a Decision Tree

DTL(Δ,Predicates)

1. If all examples in Δ are positive then return True
2. If all examples in Δ are negative then return False
3. If Predicates in empty then return failure
4. A ← most discriminating predicate in Predicates
5. Return the tree whose:
   - root is A,
   - left branch is DTL(Δ^{+A},Predicates-A),
   - right branch is DTL(Δ^{-A},Predicates-A)

Subset of examples that satisfy A

Using Information Theory

- Rather than minimizing the probability of error, most existing learning procedures try to minimize the expected number of questions needed to decide if an object x satisfies CONCEPT
- This minimization is based on a measure of the "quantity of information" that is contained in the truth value of an observable predicate

of Questions to Identify an Object

- Let U be a set of size |U|
- We want to identify any particular object of U with only True/False questions
- What is the minimum number of questions that will we need on the average?
- The answer is $\log_2|U|$, since the best we can do at each question is to split the set of remaining objects in half

of Questions to Identify an Object

- Now, suppose that a question **Q** splits U into two subsets T and F of sizes |T| and |F|
- What is the minimum number of questions that will we need on the average, assuming that we will ask **Q** first?

of Questions to Identify an Object

- Now, suppose that a question **Q** splits U into two subsets T and F of sizes |T| and |F|
- What is the minimum average number of questions that will we need assuming that we will ask **Q** first?
- The answer is:

$$(|T|/|U|)\log_2|T| + (|F|/|U|)\log_2|F|$$

Information Content of an Answer

- The number of questions saved by asking **Q** is:
  $I_Q = \log_2|U| - (|T|/|U|)\log_2|T| + (|F|/|U|)\log_2|F|$
  which is called the information content of the answer to **Q**
- Posing $p_T = |T|/|U|$ and $p_F = |F|/|U|$, we get:
  $I_Q = \log_2|U| - p_T\log_2(p_T|U|) - p_F\log_2(p_F|U|)$
- Since $p_T+p_F = 1$, we have:
  $I_Q = -p_T\log_2 p_T - p_F\log_2 p_F = \mathbf{I}(p_T,p_F) \leq 1$

Application to Decision Tree

- In a decision tree we are not interested in identifying a particular object from a set Δ, but in determining if a certain object x verifies or contradicts a predicate CONCEPT
- Let us divide Δ into two subsets:
  - Δ^+ : the positive examples
  - Δ^- : the negative examples
- Let $p = |\Delta^+|/|\Delta|$ and $q = 1-p$
- The information content of the answer to the question CONCEPT(x)? would be:
  $I_{CONCEPT} = \mathbf{I}(p,q) = -p\log_2 p - q\log_2 q$

Application to Decision Tree

- In a decision tree we are not interested in identifying a particular object from a set Δ, but in determining if a certain object x verifies or contradicts CONCEPT
- Let us divide Δ into two subsets:
  - Δ^+ : the positive examples
  - Δ^- : the negative examples
- Let $p = |\Delta^+|/|\Delta|$ and $q = 1-p$

Application to Decision Tree

- Instead, we can ask A(x)? where A is an observable predicate
- The answer to A(x)? divides Δ into two subsets Δ^{+A} and Δ^{-A}
- Let p_1 be the ratio of objects that verify CONCEPT in Δ^{+A}, and $q_1=1-p_1$
- Let p_2 be the ratio of objects that verify CONCEPT in Δ^{-A}, and $q_2=1-p_2$

Application to Decision Tree

At each recursion, the learning procedure includes in the decision tree the observable predicate that maximizes the gain of information

$I_{CONCEPT} - (|\Delta^{+A}|/|\Delta|)\ \mathbf{I}(p_1,q_1) + (|\Delta^{-A}|/|\Delta|)\ \mathbf{I}(p_2,q_2)$

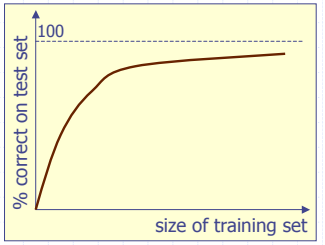
This predicate is the most discriminating

- The expected information content of the answer to the question CONCEPT(x)? would then be:

$(|\Delta^{+A}|/|\Delta|)\ \mathbf{I}(p_1,q_1) + (|\Delta^{-A}|/|\Delta|)\ \mathbf{I}(p_2,q_2) \leq I_{CONCEPT}$

Miscellaneous Issues

- Assessing performance:
  - Training set and test set
  - Learning curve



Typical learning curve

Miscellaneous Issues

- Assessing performance:
  - Training set and test set
  - Learning curve
- Overfitting
  - Tree pruning
  - Cross-validation

Risk of using irrelevant

The resulting decision tree + majority rule may not classify correctly all examples in the training set

Terminate recursion when information gain is too small

Miscellaneous Issues

- Assessing perf
  - Training set an
  - Learning curve
- Overfitting
  - Tree pruning
  - Cross-validatio
- Missing data

The value of an observable predicate P is unknown for an example x. Then construct a decision tree for both values of P and select the value that ends up classifying x in the largest class

Miscellaneous Issues

- Assessing performance:
  - Training set and test set
  - Learning curve
- Overfitting
  - Tree pruning
  - Cross-validation
- Missing data
- Multi-valued and continuous attributes

Select threshold that maximizes information gain

These issues occur with virtually any learning method

Applications of Decision Tree

- Medical diagnostic
- Evaluation of geological systems for assessing gas and oil basins
- Early detection of problems (e.g., jamming) during oil drilling operations
- Automatic generation of rules in expert systems

Applications of Decision Tree

- SGI flight simulator
- predicting emergency C sections
  - identified new class of high risk patients
- SKICAT – classifying stars and galaxies from telescope images
  - 40 attributes
  - 8 levels deep
  - could correctly classify images that were too faint for human to classify
  - 16 new high red-shift quasars discovered in at least one order of magnitude less observation time

Summary

- Inductive learning frameworks
- Logic inference formulation
- Hypothesis space and KIS bias
- Inductive learning of decision trees
- Using information theory
- Assessing performance
- Overfitting

Learning II: Neural Networks

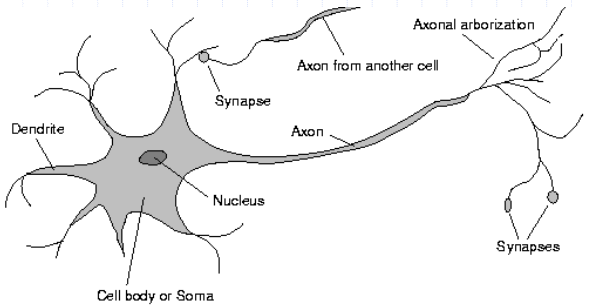
R&N: ch 19

based on material from Marie desJardins, Ray Mooney, Daphne Koller

Neural function

- Brain function (thought) occurs as the result of the firing of **neurons**
- Neurons connect to each other through **synapses**, which propagate **action potential** (electrical impulses) by releasing **neurotransmitters**
- Synapses can be **excitatory** (potential-increasing) or **inhibitory** (potential-decreasing), and have varying **activation thresholds**
- Learning occurs as a result of the synapses' **plasticicity**: They exhibit long-term changes in connection strength
- There are about 10^{11} neurons and about 10^{14} synapses in the human brain

Biology of a neuron



Brain structure

- Different areas of the brain have different functions
  - Some areas seem to have the same function in all humans (e.g., Broca's region); the overall layout is generally consistent
  - Some areas are more plastic, and vary in their function; also, the lower-level structure and function vary greatly
- We don't know how different functions are "assigned" or acquired
  - Partly the result of the physical layout / connection to inputs (sensors) and outputs (effectors)
  - Partly the result of experience (learning)
- We *really* don't understand how this neural structure leads to what we perceive as "consciousness" or "thought"
- Our neural networks are not nearly as complex or intricate as the actual brain structure

Comparison of computing power

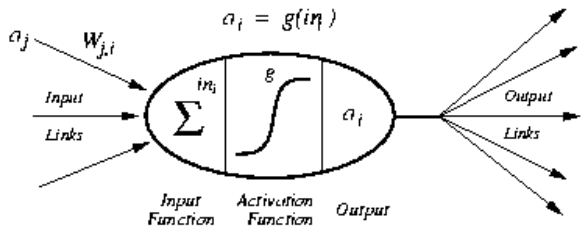
	Computer	Human Brain
Computational units	1 CPU, 10^5 gates	10^{11} neurons
Storage units	10^9 bits RAM, 10^{10} bits disk	10^{11} neurons, 10^{14} synapses
Cycle time	10^{-8} sec	10^{-3} sec
Bandwidth	10^9 bits/sec	10^{14} bits/sec
Neuron updates/sec	10^5	10^{14}

- Computers are way faster than neurons…
- But there are a lot more neurons than we can reasonably model in modern digital computers, and they all fire in parallel
- Neural networks are designed to be massively parallel
- The brain is effectively a billion times faster

Neural networks

- Neural networks are made up of **nodes** or **units**, connected by **links**
- Each link has an associated **weight** and **activation level**
- Each node has an **input function** (typically summing over weighted inputs), an **activation function**, and an **output**

Neural unit



Model Neuron

- Neuron modeled a unit j
- weights on input unit I to j, w_{ji}
- net input to unit j is:

$$net_j = \sum_i w_{ij} \cdot o_i$$

- threshold T_j
- o_j is 1 if $net_j > T_j$

Neural Computation

- McCollough and Pitt (1943)showed how LTU can be use to compute logical functions
  - AND?
  - OR?
  - NOT?
- Two layers of LTUs can represent any boolean function

Learning Rules

- Rosenblatt (1959) suggested that if a target output value is provided for a single neuron with fixed inputs, can incrementally change weights to learn to produce these outputs using the *perceptron learning rule*
  - assumes binary valued input/outputs
  - assumes a single linear threshold unit

Perceptron Learning rule

- If the target output for unit j is t_j

$$w_{ji} = w_{ji} + \eta(t_j - o_j)o_i$$

- Equivalent to the intuitive rules:
  - If output is correct, don't change the weights
  - If output is low (o_j=0, t_j=1), increment weights for all the inputs which are 1
  - If output is high (o_j=1, t_j=0), decrement weights for all inputs which are 1
  - Must also adjust threshold. Or equivalently asuume there is a weight w_{j0} for an extra input unit that has o_0=1

Perceptron Learning Algorithm

- Repeatedly iterate through examples adjusting weights according to the perceptron learning rule until all outputs are correct
  - Initialize the weights to all zero (or random)
  - Until outputs for all training examples are correct
    - for each training example e do
      - compute the current output o_j
      - compare it to the target t_j and update weights
- each execution of outer loop is an epoch
- for multiple category problems, learn a separate perceptron for each category and assign to the class whose perceptron most exceeds its threshold
- Q: when will the algorithm terminate?

Perceptron Video

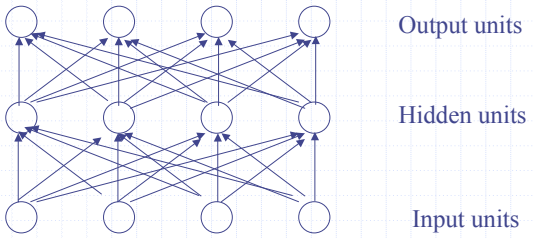
Representation Limitations of a Perceptron

- Perceptrons can only represent linear threshold functions and can therefore only learn functions which linearly separate the data, I.e. the positive and negative examples are separable by a hyperplane in n-dimensional space

Perceptron Learnability

- **Perceptron Convergence Theorem**: If there are a set of weights that are consistent with the training data (I.e. the data is linearly separable), the perceptron learning algorithm will converge (Minicksy & Papert, 1969)
- Unfortunately, many functions (like parity) cannot be represented by LTU

Layered feed-forward network



"Executing" neural networks

- Input units are set by some exterior function (think of these as sensors), which causes their output links to be activated at the specified level
- Working forward through the network, the input function of each unit is applied to compute the input value
  - Usually this is just the weighted sum of the activation on the links feeding into this node
- The activation function transforms this input function into a final value
  - Typically this is a nonlinear function, often a sigmoid function corresponding to the "threshold" of that node