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

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 $f(X) \rightarrow (0,1)$
  - Decision trees
  - Version space method
- Probabilistic/Numeric learning
  - learn a function $f(X) \rightarrow \mathbb{R}$
  - 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 \( h = \Delta \) 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:
  - \(((r=1) \lor \ldots \lor (r=10)) \leftrightarrow \text{NUM}(r)\)
  - \(((r=J) \lor (r=Q) \lor (r=K)) \leftrightarrow \text{FACE}(r)\)
  - \(((s=S) \lor (s=C)) \leftrightarrow \text{BLACK}(s)\)
  - \(((s=D) \lor (s=H)) \leftrightarrow \text{RED}(s)\)
- Training set \( \Delta \):
  - \(\text{REWARD}([4,C]) \land \text{REWARD}([7,C]) \land \text{REWARD}([2,S]) \land \neg \text{REWARD}([5,H]) \land \neg \text{REWARD}([J,S])\)

**Rewarded Card Example**

- Background knowledge KB:
  - \(((r=1) \lor \ldots \lor (r=10)) \leftrightarrow \text{NUM}(r)\)
  - \(((r=J) \lor (r=Q) \lor (r=K)) \leftrightarrow \text{FACE}(r)\)
  - \(((s=S) \lor (s=C)) \leftrightarrow \text{BLACK}(s)\)
  - \(((s=D) \lor (s=H)) \leftrightarrow \text{RED}(s)\)
- Training set \( \Delta \):
  - \(\text{REWARD}([4,C]) \land \text{REWARD}([7,C]) \land \text{REWARD}([2,S]) \land \neg \text{REWARD}([5,H]) \land \neg \text{REWARD}([J,S])\)
- Possible hypothesis:
  - \( h = (\text{NUM}(r) \land \text{BLACK}(s) \land \text{REWARD}([r,s])) \)

There are several possible inductive hypotheses.
Learning a Predicate

- Set $E$ of objects (e.g., cards)
- Goal predicate $\text{CONCEPT}(x)$, 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 $\text{CONCEPT}$ for some combinations of values of the observable predicates

A Possible Training Set

<table>
<thead>
<tr>
<th>Ex. #</th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
<th>$D$</th>
<th>$E$</th>
<th>$\text{CONCEPT}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>2</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>3</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>4</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>5</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>6</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>7</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>8</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>

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 $\text{CONCEPT}(x)$, 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 $\text{CONCEPT}$ for some combinations of values of the observable predicates
- Find a representation of $\text{CONCEPT}$ in the form:
  \[
  \text{CONCEPT}(x) \iff S(A, B, \ldots)
  \]
  where $S(A, B, \ldots)$ is a sentence built with the observable predicates, e.g.:
  \[
  \text{CONCEPT}(x) \iff A(x) \land (\neg B(x) \lor C(x))
  \]

Example set

- An example consists of the values of $\text{CONCEPT}$ and the observable predicates for some object $x$
- A example is positive if $\text{CONCEPT}$ 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!
An hypothesis is any sentence $h$ of the form:

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

where $S(A,B,\ldots)$ 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

### Hypothesis Space

- An hypothesis is any sentence $h$ of the form: $\text{CONCEPT}(x) \iff S(A,B,\ldots)$
  - where $S(A,B,\ldots)$ 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

### Size of the Hypothesis Space

- $n$ observable predicates
- $2^n$ entries in truth table
- In the absence of any restriction (bias), there are $2^{2n}$ hypotheses to choose from
- $n = 6 \Rightarrow 2 \times 10^{19}$ hypotheses!

### Multiple Inductive Hypotheses

- **Rewarded Card Example**
  - Background knowledge $KB$:
    - $(r=1) \lor \ldots \lor (r=10)$
    - $\iff \text{NUM}([r,s])$
    - $(r=J) \lor (r=Q) \lor (r=K)$
    - $\iff \text{FACE}([r,s])$
    - $(s=S) \lor (s=C)$
    - $\iff \text{BLACK}([r,s])$
    - $(s=D) \lor (s=H)$
    - $\iff \text{RED}([r,s])$
  - Training set $\Delta$:
    - $\text{REWARD}([4,C]) \land \text{REWARD}([7,C]) \land \text{REWARD}([2,S]) \land \neg \text{REWARD}([5,H]) \land \neg \text{REWARD}([J,S])$
  - Possible inductive hypothesis:
    - $h \equiv \text{NUM}(x) \land \text{BLACK}(x) \iff \text{REWARD}(x)$
    - $h_1 \equiv \text{BLACK}([r,s]) \land \neg (r=J) \iff \text{REWARD}([r,s])$
    - $h_2 \equiv ([r,s]=[4,C]) \land ([r,s]=[7,C]) \land [r,s]=[2,S]) \land \neg ([r,s]=[5,H]) \land \neg ([r,s]=[J,S]) \iff \text{REWARD}([r,s])$

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

- Inductive Learning Scheme
Keep-It-Simple (KIS) Bias

**Motivation**

- If an hypothesis is too complex it may not be worth learning it (data caching might just do the job as well)
- There are much fewer simple hypotheses than complex ones, hence the hypothesis space is smaller

Examples:

- 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

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 $O(n^k)$

Predicate-Learning Methods

- Decision tree
- Version space

Predicate as a Decision Tree

The predicate $CONCEPT(x) \equiv A(x) \land (\neg B(x) \lor C(x))$ 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
- $CONCEPT = POISONOUS$
- $A = YELLOW$
- $B = BIG$
- $C = SPOTTED$

Decision Tree

WillWait predicate (Russell and Norvig)

- Patrons?
- Hungry?
- Type?
- FriSat?

Multi-valued attributes

Predicate as a Decision Tree

- A?
- B?
- C?
**Predicate as a Decision Tree**

The predicate \( \text{CONCEPT}(x) \leftrightarrow A(x) \land (\neg B(x) \lor C(x)) \) 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
- \( \text{CONCEPT} = \text{POISONOUS} \)
- \( A = \text{YELLOW} \)
- \( B = \text{BIG} \)
- \( C = \text{SPOTTED} \)
- \( D = \text{FUNNEL-CAP} \)
- \( E = \text{BULKY} \)

**Training Set**

<table>
<thead>
<tr>
<th>Ex. #</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>S</th>
<th>CONCEPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>2</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>3</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>4</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>5</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>6</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>7</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>8</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>9</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>10</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>11</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>12</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>13</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>

**Possible Decision Tree**

\[
\text{CONCEPT} \leftrightarrow (D \land (\neg E \lor A)) \lor (C \land (B \lor (E \land \neg A) \lor A))
\]

**Possible Decision Tree**

\[
\text{KIS bias} \rightarrow \text{Build smallest decision tree}
\]

\[
\text{Computationally intractable problem} \rightarrow \text{greedy algorithm}
\]
Getting Started

The distribution of the 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

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:

$Pr(E) = (8/13) \times (2/8) + (5/13) \times 0 = 2/13$
Assume It’s B

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

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

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

If we test only E, we will report that CONCEPT is False, 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

So, the best predicate to test is A
The majority rule gives the probability of error $Pr(E) = 1/8$

L = CONCEPT $\iff$ A $\land$ (C $\lor$ $\neg$B)

Learning a Decision Tree

DTL($\Delta$,Predicates)

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

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?

The answer is:

$$\frac{|T|}{|U|} \log_2|T| + \frac{|F|}{|U|} \log_2|F|$$
**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:
  $$I_Q = - p_T \log_2 p_T - p_F \log_2 p_F = I(p_T, p_F) \leq 1$$

**Application to Decision Tree**

- In a decision tree we are not interested in identifying a particular object from a set $U=\Delta$, but in determining if a certain object $x$ verifies or contradicts CONCEPT.
- Let us divide $\Delta$ into two subsets:
  - $\Delta^+$: the positive examples
  - $\Delta^-$: 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 $\Delta$ into two subsets $\Delta^A$ and $\Delta^\neg A$.
- Let $p_1$ be the ratio of objects that verify CONCEPT in $\Delta^A$, and $q_1 = 1 - p_1$.
- Let $p_2$ be the ratio of objects that verify CONCEPT in $\Delta^\neg A$, and $q_2 = 1 - p_2$. 

$$I_{CONCEPT} = I(p, q) = - p \log_2 p - q \log_2 q$$
Instead, we can ask $A(x)$? The answer divides $\Delta$ into two subsets $\Delta^+A$ and $\Delta^-A$.

Let $p_1$ be the ratio of objects that verify CONCEPT in $\Delta^+A$ and $q_1 = 1 - p_1$.

Let $p_2$ be the ratio of objects that verify CONCEPT in $\Delta^-A$ and $q_2 = 1 - p_2$.

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

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

**Application to Decision Tree**

At each recursion, the learning procedure includes in the decision tree the observable predicate that maximizes the gain of information $I_{\text{CONCEPT}} - (\frac{|\Delta^+A|}{|\Delta|}) I(p_1, q_1) + (\frac{|\Delta^-A|}{|\Delta|}) I(p_2, q_2)$.

This predicate is the most discriminating.

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

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

**Miscellaneous Issues**

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

- **Overfitting:**
  - Tree pruning
  - Cross-validation

- 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.

- Terminate recursion when information gain is too small.

- The resulting decision tree + majority rule may not classify correctly all examples in the training set.
**Miscellaneous Issues**

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

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’ plasticity: 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

<table>
<thead>
<tr>
<th>Computer</th>
<th>Human Brain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computation units</td>
<td>10^20</td>
</tr>
<tr>
<td>Storage units</td>
<td>10^12 bits RAM, 10^15 bits disk</td>
</tr>
<tr>
<td>Cycle time</td>
<td>10^-9 sec</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>10^7 bits/sec</td>
</tr>
<tr>
<td>Neuron speed/sec</td>
<td>10^12</td>
</tr>
</tbody>
</table>

- 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.

Model Neuron

- Neuron modeled a unit j.
- Weights on input unit I to j, w_{ji}.
- Net input to unit j is:
  \[ \text{net}_j = \sum_i w_{ji} \cdot o_i \]
- Threshold T_j.
- o_j is 1 if net_j > T_j.
Neural Computation

- McCollough and Pitt (1943) showed how LTU can be used 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, one 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 $t_j$
  \[ w_{ji} = w_{ji} + \eta(t_j - o_j) \]

- 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 assume 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 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

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
“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.