COMP3411/9414/9814 Artificial Intelligence
Session 1, 2018

Tutorial Solutions - Week 7

This page was last updated: 04/29/2018 10:40:25

Activity 10.2 Multi-Layer Neural Networks to Compute Logical Functions

---

Explain how each of the following could be constructed:

1. Perceptron to compute the OR function of *m* inputs

   Set the bias weight to -½, all other weights to 1.

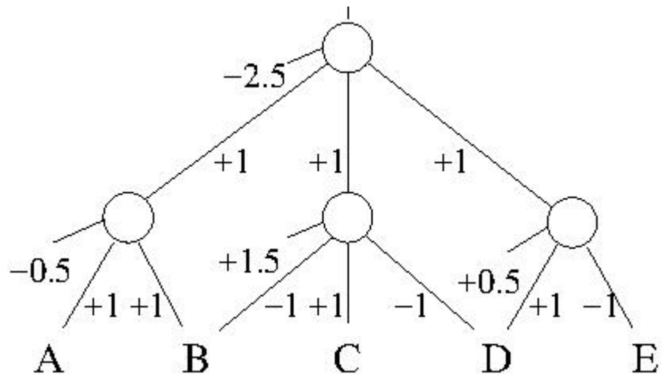
2. Perceptron to compute the AND function of *n* inputs

   Set the bias weight to (½ - *n*), all other weights to 1.

3. 2-Layer Neural Network to compute any (given) logical expression, assuming it is written in Conjunctive Normal Form.

   Each hidden node should compute one disjunctive term in the expression. The weights should be -1 for items that are negated, +1 for the others. The bias should be (*k* - ½) where *k* is the number of items that are negated. The output node then computes the conjunction of all the hidden nodes, as in part 2.

   For example, here is a network that computes (A ∨ B) ∧ (¬ B ∨ C ∨ ¬ D) ∧ (D ∨ ¬ E)



Activity 10.3 Computing XOR with One Hidden Unit

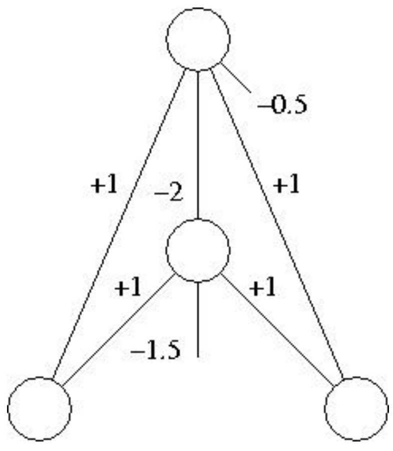
We have seen how to compute XOR using a Neural Network with two inputs, two hidden units and one output.

Can you construct a Neural Network to compute XOR which has only one hidden unit, but also includes shortcut connections from the two inputs directly to the (one) output?

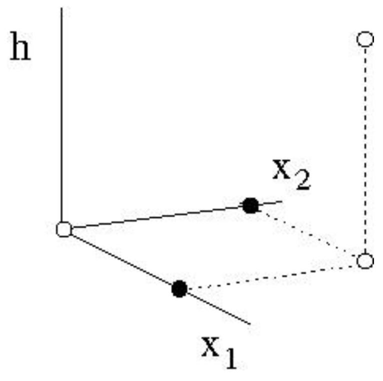
Hint: start with a network that computes the inclusive OR, and then try to think of how it could be modified.

Inclusive and exclusive OR are similar, but differ in the case where both inputs are True, i.e. where (x_1 AND x_2) is True.

We therefore introduce a single hidden unit which computes (x_1 AND x_2). This hidden unit is connected to the output with a negative weight, thus forcing this input to be classified negatively.



The addition of this hidden "feature" creates a 3-dimensional space in which the points can be linearly separated by a plane. The weights for the output unit (+1,+1,-2) specify a vector perpendicular to the separating plane, and its distance from the origin is determined by the output bias divided by the length of this vector.

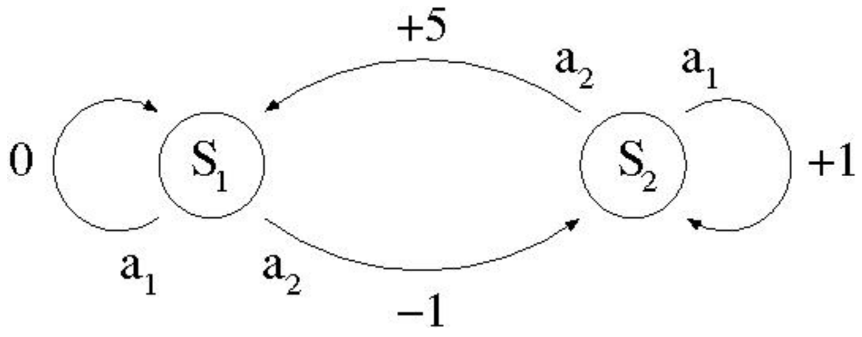


Activity 7.4 Q-Learning [COMP3411/9814 only]

Consider a world with two states $S = \{S_1, S_2\}$ and two actions $A = \{a_1, a_2\}$, where the transitions δ and reward r for each state and action are as follows:

$\delta(S_1, a_1) = S_1,\ r(S_1, a_1) = \ \ 0$
$\delta(S_1, a_2) = S_2,\ r(S_1, a_2) = -1$
$\delta(S_2, a_1) = S_2,\ r(S_2, a_1) = +1$
$\delta(S_2, a_2) = S_1,\ r(S_2, a_2) = +5$

1. Draw a picture of this world, using circles for the states and arrows for the transitions.



2. Assuming a discount factor of $\gamma = 0.9$, determine:
   a. the optimal policy $\pi^* : S \rightarrow A$

      $\pi^*(S_1) = a_2$
      $\pi^*(S_2) = a_2$

   b. the value function $V^* : S \rightarrow R$

      $V(S_1) = -1 + \gamma V(S_2)$
      $V(S_2) = 5 + \gamma V(S_1)$

      So $V(S_1) = -1 + 5\gamma + \gamma^2 V(S_1)$
      i.e. $V(S_1) = (-1 + 5\gamma) / (1 - \gamma^2) = 3.5 / 0.19 = 18.42$

      $V(S_2) = 5 + \gamma V(S_1) = 5 + 0.9 * 3.5 / 0.19 = 21.58$

   c. the "Q" function $Q : S \times A \rightarrow R$

      $Q(S_1, a_1) = \gamma V(S_1) = 16.58$
      $Q(S_1, a_2) = V(S_1) = 18.42$
      $Q(S_2, a_1) = 1 + \gamma V(S_2) = 20.42$
      $Q(S_2, a_2) = V(S_2) = 21.58$

3. Write the Q values in a matrix:

Q	a_1	a_2
S_1	16.58	18.42
S_2	20.42	21.58

4. Trace through the first few steps of the Q-learning algorithm, with all Q values initially set to zero. Explain why it is necessary to force exploration through probabilistic choice of actions, in order to ensure convergence to the true Q values.

current state	chosen action	new Q value
S_1	a_1	$0 + \gamma*0 = 0$
S_1	a_2	$-1 + \gamma*0 = -1$
S_2	a_1	$1 + \gamma*0 = +1$

   At this point, the table looks like this:

Q	a_1	a_2
S_1	0	-1
S_2	1	0

   If we do not force exploration, the agent will always prefer action a_1 in state S_2, so it will never explore action a_2. This means that $Q(S_2, a_2)$ will remain zero forever, instead of converging to the true value of 21.58 . If we force exploration, the next few steps might look like this:

current state	chosen action	new Q value
S_2	a_2	$5 + \gamma*0 = 5$
S_1	a_1	$0 + \gamma*0 = 0$
S_1	a_2	$-1 + \gamma*5 = 3.5$
S_2	a_1	$1 + \gamma*5 = 5.5$
S_2	a_2	$5 + \gamma*3.5 = 8.15$

   Now we have this table:

Q	a_1	a_2
S_1	0	3.5
S_2	5.5	8.15

   From this point on, the agent will prefer action a_2 both in state S_1 and in state S_2. Further steps refine the Q value estimates, and, in the limit, they will converge to their true values.

current state	chosen action	new Q value
S_1	a_1	$0 + \gamma*3.5 = 3.15$
S_1	a_2	$-1 + \gamma*8.15 = 6.335$
S_2	a_1	$1 + \gamma*8.15 = 8.335$
S_2	a_2	$5 + \gamma*6.34 = 10.70$

   etc...