Representation of CNF and DNF by a neural net

Consider neural nets with **thresholds** (and not sigmoids) at each node. These can easily compute CNF and DNF Boolean functions. A Boolean function of n Boolean variables is a function $f(x_1, \ldots x_n)$ that produces as output either 0 or 1. Special cases are CNF and DNF.
A DNF is a Boolean function of the form:

$$f = t_1 \vee t_2 \vee \ldots \vee t_m$$

Each t_j is of the form $q_1 \wedge q_2 \wedge \ldots \wedge q_{n_j}$, and each q_i is either a variable or a negation of a variable. For example, the following is a DNF:

$$(x_1 \wedge x_2 \wedge \overline{x_3}) \vee (\overline{x_1} \wedge \overline{x_2} \wedge \overline{x_3}) \vee (x_1 \wedge x_2)$$

A CNF is a Boolean function of the form:

$$f = c_1 \wedge c_2 \wedge \ldots \wedge c_m$$

Each c_j is of the form $q_1 \vee q_2 \vee \ldots \vee q_{n_j}$, and each q_i is either a variable or a negation of a variable. For example, the following is a DNF:

$$(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee x_2)$$

It is known (and very easy to show) that any Boolean function can be expressed as CNF and as DNF. In the worst case a Boolean function may require 2^n DNF terms, or 2^n CNF clauses.

Perceptrons

Theorem 1: A perceptron with inputs $x_1, \ldots, x_n$ and bias can compute any 1-term of a DNF.

Constructive procedure: Let $t = q_1 \wedge \ldots \wedge q_r$ be a DNF term. The following algorithm converts it into a single linear inequality that can be implemented by a single threshold unit.

Step 1. Write the the inequality

$$X_1 + \ldots + X_r \geq r - 0.5, \qquad X_i = \begin{cases} x_i & \text{if } q_i = x_i \\ 1 - x_i & \text{if } q_i = \overline{x_i} \end{cases}$$

Step 2. Simplify the above inequality to the form:

$$w_0 + w_1 x_1 + \ldots + w_r x_r \geq 0$$

Example: $x_1 \wedge x_2 \wedge \overline{x_3}$
$x_1 + x_2 + (1 - x_3) \geq 3 - 0.5$
$-1.5 + x_1 + x_2 - x_3 \geq 0$

Theorem 2: A perceptron with inputs $x_1, \ldots, x_n$ and bias cannot compute all 2-DNF.
Proof: Observe that the xor function can be written as a 2-DNF: $(x_1 \wedge \overline{x_2}) \vee (\overline{x_1} \wedge x_2)$.

Theorem 3: A perceptron with inputs $x_1, \ldots, x_n$ and bias can compute any 1-clause of a CNF.

Constructive procedure: Let $c = q_1 \vee \ldots \vee q_r$ be a CNF clause. The following algorithm converts it into a single linear inequality that can be implemented by a single threshold unit.

Step 1. Write the the inequality

$$X_1 + \ldots + X_r \geq 0.5, \qquad X_i = \begin{cases} x_i & \text{if } q_i = x_i \\ 1 - x_i & \text{if } q_i = \overline{x_i} \end{cases}$$

Step 2. Simplify the above inequality to the form:

$$w_0 + w_1 x_1 + \ldots + w_r x_r \geq 0$$

Example: $x_1 \vee x_2 \vee \overline{x_3}$
$x_1 + x_2 + (1 - x_3) \geq 0.5$
$0.5 + x_1 + x_2 - x_3 \geq 0$

Theorem 4: A perceptron with inputs $x_1, \ldots, x_n$ and bias cannot compute all 2-CNF.
Proof: Observe that the xor function can be written as a 2-CNF: $(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$

Neural Nets (multi layer perceptrons)

Theorem 5: A neural-net with one hidden layer, given inputs $x_1, \ldots, x_n$ and bias can compute any DNF. An r-term DNF requires no more than r nodes in the hidden layer.

Constructive procedure: Suppose $f = t_1 \vee t_2 \vee \ldots \vee t_m$. The neural net has the hidden layer nodes $V_1, \ldots, V_m$. The node V_j is connected to the inputs following the procedure in Theorem 1. The nodes $V_1, \ldots, V_m$ are connected to the output node using the procedure of Theorem 3.

Theorem 6: A neural-net with one hidden layer, given inputs $x_1, \ldots, x_n$ and bias can compute any CNF. An r-clause CNF requires no more than r nodes in the hidden layer.

Constructive procedure: Suppose $f = c_1 \wedge c_2 \wedge \ldots \wedge c_m$. The neural net has the hidden layer nodes $V_1, \ldots, V_m$. The node V_j is connected to the inputs following the procedure in Theorem 3. The nodes $V_1, \ldots, V_m$ are connected to the output node using the procedure of Theorem 1.

What if network vaiables are $-1/1$?

$$x_1 \vee x_2 \vee \ldots \vee x_n \quad \Leftrightarrow \quad \sum_{i=1}^{n} x_i > -n + 0.5$$

$$x_1 \wedge x_2 \wedge \ldots \wedge x_n \quad \Leftrightarrow \quad \sum_{i=1}^{n} x_i > n - 0.5$$

$$\neg x \quad \Leftrightarrow \quad -x$$