Fig. 1. The 3-Node Network.

Each node $N_i$ computes a linear threshold function (also called $N_i$) on its inputs. If $N_i$ has input $x = (x_1, \ldots, x_m)$, then for some values $a_0, \ldots, a_m$,

$$N_i(x) = \begin{cases} +1 & \text{if } a_1x_1 + a_2x_2 + \cdots + a_mx_m > a_0 \\ -1 & \text{otherwise}. \end{cases}$$
Definition 1. Given a neural network $\mathcal{N}$, let the \textit{training problem for $\mathcal{N}$} be the question:

“Given a set of training examples, do there exist edge weights and thresholds for the nodes of $\mathcal{N}$ so that it produces output consistent with the training set?”

Theorem 2. \textit{Training the 3-Node Network is NP-complete.}

\[\text{i.e. believed to take time } O(2^n)\]

The proof of Theorem 2 involves reducing the known NP-complete problem “Set-Splitting” to the network training problem. In order to more clearly un-
Aside: $\text{NP}$-completeness

1) Boolean satisfiability problem (SAT) is $\text{NP}$-complete. (Cook 1971)

Example: $(x_1 \lor \overline{x_2}) \land (x_2 \lor x_3 \lor x_4) \land (\overline{x_1} \lor x_3)$ (*)

Boolean var. True / False

Q: Does there exist a truth assignment s.t. (*) evaluates to True?

Best alg. $O(2^n)$ – $n$ is # vars (unless $P=\text{NP}$) 

Computationally infeasible.
2) Thousands of other known NP-complete problems.

E.g. graph coloring is NP-complete (Col) \( O(2^n) \) - n nodes \& 3-col \n
How?

Poly time reduction / transformation:

\[ \text{SAT} \Rightarrow \text{COL} \]

\( \text{If } \) instance is satisfiable \( \text{iff } \) instance is Col 3-colorable.
3) All NP-complete problems are equivalent in computational terms.

E.g. $\text{SAT} \Rightarrow \text{COL}$ & $\text{COL} \Rightarrow \text{SAT}$

For each node $i$:

$$(x_i^B \lor x_i^R \lor x_i^G)$$

Also:

$$x_i^R \Rightarrow \overline{x_i^B}$$
$$x_i^B \Rightarrow \overline{x_i^R}$$
$$x_i^B \Rightarrow x_i^G$$

$$(\overline{x_i^B} \lor x_i^G)$$

Finally, if $i$ & $j$ connected by edge then:

$$x_i^B \Rightarrow x_j^B$$
$$x_i^B \Rightarrow x_j^B$$
$$x_i^B \Rightarrow x_j^B$$

$n = 7$ (# nodes)
So, we can go from a coloring problem to a logic/problem & vice versa. Efficiently (poly time)

3-node neural net training is also part of this class.

\[(x_1 \lor \neg x_2) \land (x_3 \lor x_4 \lor x_5) \land (\neg x_1 \lor x_3)\]

i.e.

SAT \iff COL

3MAT

\[n \rightarrow (\# \text{neurons})\]

\[\text{In 3rd time, opt.} \]
**Theorem 2.** Training the 3-Node Network is NP-complete.

A reduction from the Set-Splitting problem.

Ex. Set-Splitting

\[ S = \{ 0, 1, 2, 3 \} \text{, } n=3 \]

\[ c_1 = \{ 0, 2 \}, \quad c_2 = \{ 2, 3 \} \]

Can I 2-color element in \( S \)?

- Each \( c_i \) has 2 colors?

\[ \text{yes} \]

It can be shown that

Set-Splitting is solvable iff

Training \( n \times 1 \) example can be learned exactly by

3-node neural net.
We also show the training problem for the following networks to be NP-complete:

1. The 3-Node Network restricted so that any or all of the weights for one hidden node are required to equal the corresponding weights of the other (so possibly only the thresholds differ) and any or all of the weights are required to belong to \{+1, -1\}.

In addition we show that any set of positive and negative training examples classifiable by the 3-node network with XOR output node (for which training is NP-complete) can be correctly classified by a perceptron with \(O(n^2)\) inputs which consist of the original \(n\) inputs and all products of pairs of the original \(n\) inputs (for which training can be done in polynomial-time using linear programming techniques).

This is a surprising & profound observation.
A unit employing the rectifier is also called a **rectified linear unit (ReLU)**.[4]

For the first time in 2011,[1] the use of the rectifier as a non-linearity has been shown to enable training deep supervised neural networks without requiring unsupervised pre-training. Rectified linear units, compared to **sigmoid function** or similar activation functions, allow for faster and effective training of deep neural architectures on large and complex datasets.