On the Complexity of Neural-Network-Learned Functions

Caren Marzban[1], Raju Viswanathan[2]

[1] Applied Physics Lab, and Dept of Statistics, University of Washington, Seattle, WA 98195

[2] Cyberon LLC, 3073 Lawrence Expressway, Santa Clara, CA 95051

Abstract

There exists a large body of knowledge on the assessment of the complexity of functions that are learnable by Neural Networks (NNs). These works are generally highly theoretical, and consequently, not directly useful for most NN practitioners. In this talk, we will consider a completely binary problem (i.e., with binary input notes, a binary output node, binary hidden nodes, and binary weights). For such problems, the number of possible (Boolean) functions relating the inputs to the output is finite and countable. Here, we will argue that the number of weight configurations that represent each function is a measure of the complexity of that function. This simple set-up allows one not only to understand some of the aforementioned theoretical results, but it also allows one to ask, and answer, practical questions like "How complex is this function that my NN has learned from data?"

Main Questions

Suppose you have properly trained a Neural Network (NN).
Q: How complex is it?

– VC dimension?
– The number of hidden nodes (nhd) (what about weights, act.func, ...)?
– What if nhd is fixed?
– Here, we will propose another.

Let's simplify, first: Consider NNs (feed-forward MLPs) with
- binary inputs: unusual, but not too much.
- binary output: less strange (think binary targets in classification).
- binary weights: completely strange!

Consult References:
– Studied by physicists for a long time.
– Connection between NNs and Boolean functions/rules.
– Require special methods for training.
– Recent excitement over deep learning.
– Competitive with "ordinary" NNs in terms of prediction quality
– but superior in terms of speed and memeory (for mobile devices).

Preliminaries

Example: Boolean function on $n = 2$ variables (x_1, x_2):

x_1	x_2	$y = f(x_1, x_2)$															
-1	-1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1
1	-1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1
-1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1	1	1
1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	1	1
		R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	R16

$\exists\ 2^{2^n}$ such rules .

Some familiar rules are:

R11: $y = x_1$

R13: $y = x_2$

R7: XOR

R9: AND

R2: OR

No NN (yet).

Now, NN with $n = 2$ inputs, nhd=0:

$$y = sign(\omega_0 + \omega_1 x_1 + \omega_2 x_2)$$

What values of $\omega_0, \omega_1, \omega_2$ (all binary) correspond to each rule?

Example:

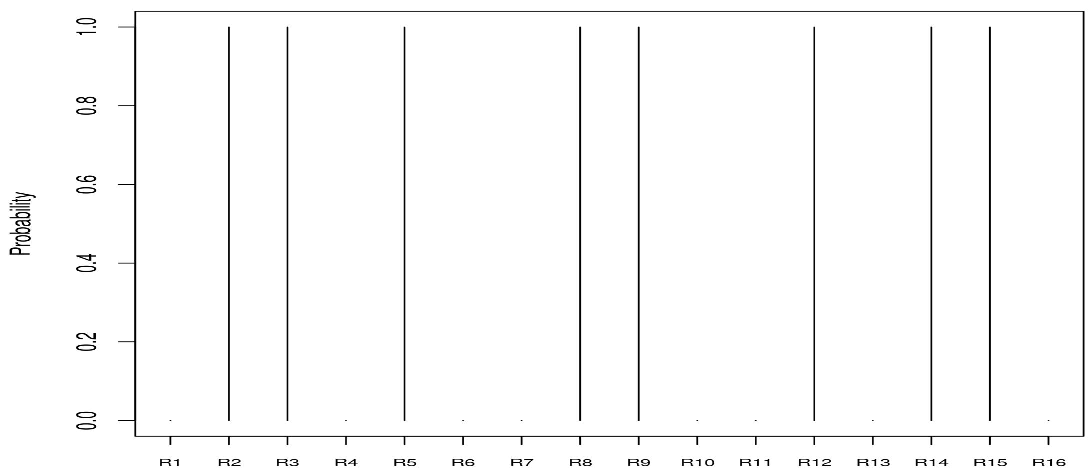
$\omega_0 = -1, \omega_1 = -1, \omega_2 = -1 \rightarrow$ NN=R2

$\omega_0 = -1, \omega_1 = +1, \omega_2 = +1 \rightarrow$ NN=R9

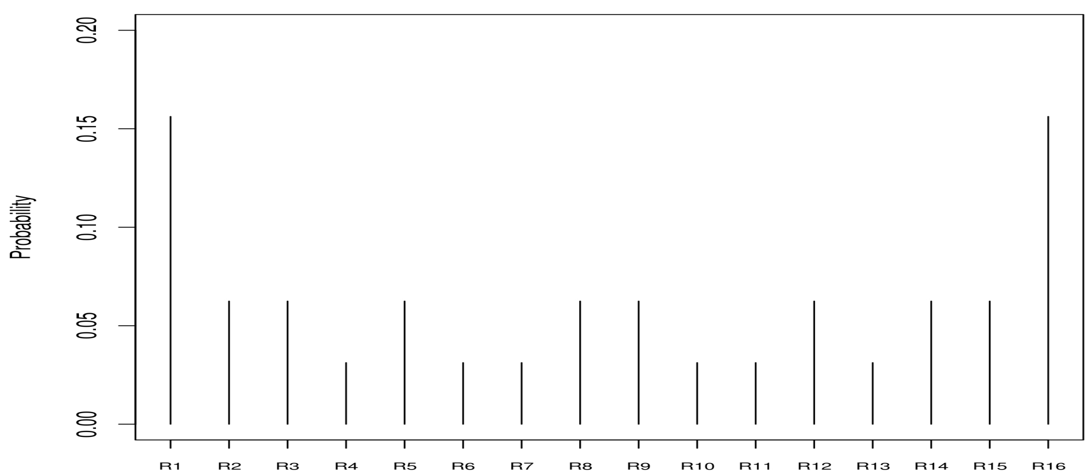
$\exists 2^3$ weight configurations ; each corresponding to a rule.

Answer

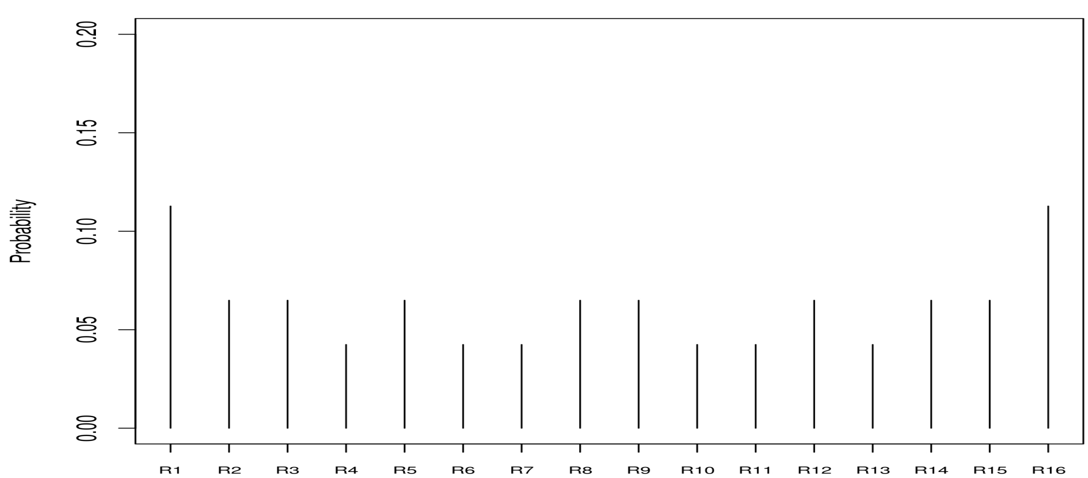
How many weight configs correspond to each rule?



nhd=0



nhd=2



nhd=4

So far ...

– For a given nhd, each rule has a "probability" of being learnt.
– Complexity (simplicity) for each rule, and the corresponding NN.
– Aside: Probs follow power-law distributions (see refs).

No data (yet).
Now (fake) data. Consider $n = 4$ inputs:

$$x_1, x_2, x_3, x_4 \sim \text{unif}(-1, 1); \quad \text{100 cases}$$

And 1 NN-worthy function:
(found this in my files - don't know whom to acknowledge.)

$$\begin{aligned} y = & - \; 3.75 + 2\sin(\pi x_1) \\ & - \; 1.4 + e^{2x_2} \\ & + \; 0.2(x_3)^{11}[10[1 - x_3)]^6 \\ & + \; 10(10x_3)^3(1 - x_3)^{10} \end{aligned}$$

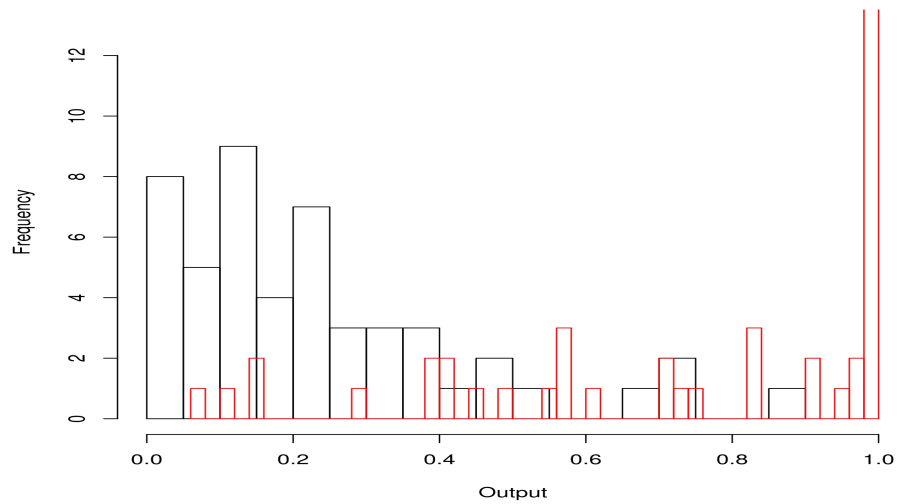
Dichotomize y (classification).

Performance

"Ordinary" NN, nhd=2 training Ctable:

42	8
8	42

Discrimination diagram (black: 0-class, red: 1-class):



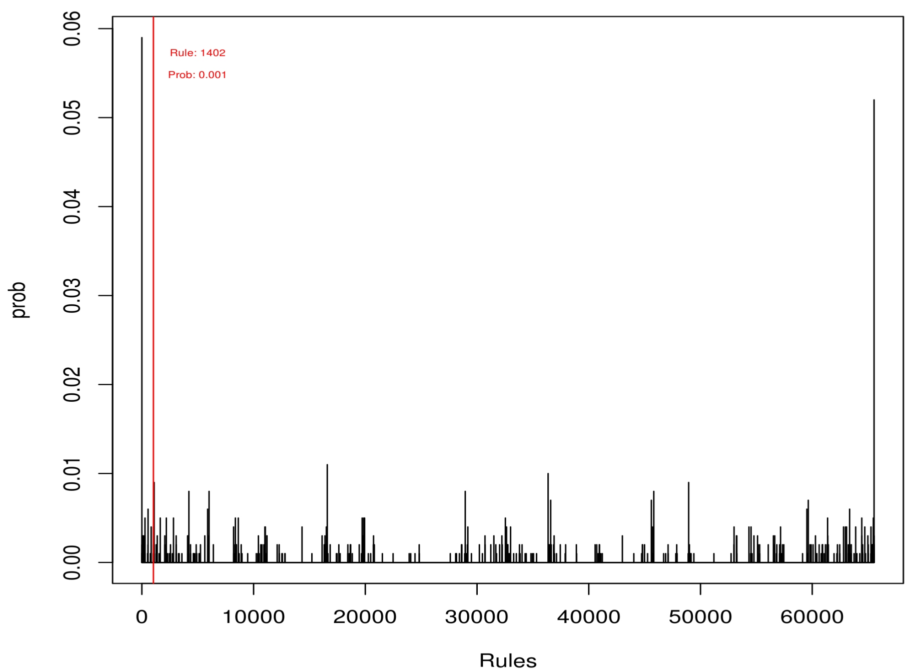
"Boolean" NN, nhd=2:
Exhaustive search (practical for small NNs)

38	12
16	34

Not much worse!
And can be improved by deeper NN and/or more hdn (see refs).

But now ...

Which rule did we just learn? And how complex is it?



Black = prob of rules for a 4-2-1 NN.
Red = rule learned by our NN.

Conclusion (in complete sentences)

1) Binary/Boolean NNs are not as bad as one might expect.
In fact, they are comparable in terms of prediction quality,
and superior in terms of speed and power requirements (see refs).

2) Can compute complexity measure for the function an NN has learned.
If it's complex, one may want to increase size of training set.
If it's simple, less training data may be sufficient.

References

Carnevalli, P., S. Patarnello (1987): “Exhaustive Thermodynamical Analysis of Boolean Learning Networks.” *Europhys. Lett.*, **4**, 1199-1204.

Van den Broeck, C., R. Kawai (1990): “Learning in feedforward boolean networks.” *Phys. Rev. A.*, **42**, 6210-6218.

Ferrán, E.A., R.P. Perazzo (1990): ”Inferential entropies of feedforward neural networks.” *Phys. Rev. A*, **42**, 6219-6226.

Gordon, M.B., P. Peretto (1990): “The statistical distribution of Boolean gates in two-inputs, one-output multilayered neural networks.” *J. Phys. A. Math. Gen.*, **23**, 3061-3072.

Parisi, G. (1992): “On the classification of learning machines.” *Network*, **4**, 259-265.

Deolalikar, V. (2001): “Mapping Boolean functions with neural networks having binary weights and zero thresholds.” *IEEE Trans. Neural Networks*, **12**, 1-8.

Anthony, Martin (2010): Neural networks and boolean functions In: Crama, Yves and Hammer, Peter L., (eds.) Boolean Models and Methods in Mathematics, Computer Science, and Engineering. Encyclopedia of mathematics and its applications (134). Cambridge University Press, New York, USA, 554-576. ISBN 9780521847520

Bernd Steinbach, Roman Kohut (2002): Neural Networks - A Model of Boolean Functions. Proceedings of 5th International Workshop on Boolean Problems

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, Yoshua Bengio1 (2016): Binarized Neural Networks: Training Neural Networks with Weights and Activations Constrained to +1 or -1. arXiv:1602.02830v3 .

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, Ali Farhadi (2016): XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. arXiv:1603.05279v4

Matthieu Courbariaux, Yoshua Bengio, Jean-Pierre David (2016): BinaryConnect: Training Deep Neural Networks with binary weights during propagations. arXiv:1511.0036

Bornholdt S, Röhl T (2003): Self-organized critical neural networks. Phys Rev E 67: 066118.