THE FINDING OF NEURAL NETWORKS' S VC DIMENSION

Ahmet ÇINAR
Firat Universitesi Mühendislik Fakültesi Bilgisayar Mühendisliği ELAZIĞ/TURKEY
E-mail. acinar@hotmail.com or
acinar@firat.edu.tr

Abstract: In this study, the VC-Dimension of a neural networks are presented. Especially, single layer feed-forward neural network is used. It is well known that VC-dimension of neural networks has been giving upper bound of neural network to learn. The neural networks that are an acyclic set of edges are considered. Signum function is used as threshold function.

Keywords: Vapnik-Chervonenkis, neural network.

1. Introduction

The VC-dimension is mapping that describes the separating capacity of a collection as sets; see Sontag (1992), Vapnik and Chervonenkis (1971) and Wenocur and Dideley (1981). In other words, the VC-dimension is a parameter that qualifies how well a collection of sets can implement an arbitrary signed set. As the collection of sets can be derived from functions, the VC-dimension is a useful qualifier for classification capability of artificial neural networks.

If we could the VC-Dimension of a neural network, then we could determine an upper bound on the number of examples we would need to see before the neural net is trained to classy according to a target concept.

The structure of paper is following; In section 2 VC–Dimension of somewhat restricted set of neural networks has been found. Section 3 describes asymptotic error rates of neural networks, and finally, conclusion has been offered.

2. A method finding VC-Dimension

First, we will consider a neural network that consists of an acyclic set of edges and a set of computation nodes. Each node computes some arbitrary function from its real-valued inputs {±}, but there is only one output node for the entire network. This may seem to be restrictive, but the computation nodes in this model are quite general, in that the nodes aren't limited to using the weighted sum of their inputs.

2.1. Basic Definitions

First all, our notation is changing slightly; C represents a concept class over an instance space X, both of which may now be infinite. We use S to represent a finite sample chosen from X. We will now define a function which characterises the response of the concepts in C to a given sample S.

Definition 2.1. For any concept class c over instance space X. For all finite samples $S \subseteq C$.
Let F_i be the class of functions computable at node i, and let F be the class of functions computed by entire net as the vary F_i. Then given a sample S of size m, we define
$\prod c(S) = \{c \bigcap S : c \in C\}$.
It is important at this point to give some kind of explanation of what this mean, especially since the set notation can be a little confusing. Especially $\prod c(S)$ collects the set of all subsets of the sample set S which are made positive by some concept c in the concept class C. Thus $c \cap S$ represents the elements of S that are labelled positive by a concept c, and $\prod c(S)$ is all such subsets for all concepts.

Definition 2.2. If $|\prod c(S)| = 2^{|S|}$ then shattered by C.
In other words, fix a sample and a concept class; if the following is true for every possible subset of the sample, then concept class shatters the sample: the subset, is made positive y some concept in the concept class, and concept does not make any other element of S positive. That is, S is shattered by C if C realises all possible dichotomies of s.

Definition 2.3. The Vapnik-Chervonenkis dimension of C, denoted $VC_{DIM}(c)$ is the largest cardinality d such that there exists a sample set of that cardinality |S|=d that is shattered by C. If no largest cardinality exists then $VC_{DIM}(c) = \infty$

According to upper definitions; we can give new definitions.

$\Delta_F(s)$ = The number of dichotomies induced by F on S and m is sample size.

$$\Delta_F(m) = \max_{s,|s|=m} \Delta_F(S)$$

($\Delta_F(s)$ = Is what we called $\prod c(S)$, and $\Delta_F(m)$ is what we called $\prod c(m)$. If a neural network has N nodes, then we have the following result:

Theorem 2.1.
Define
$d = \sum_{i=1}^{N} VC - \dim(F_i)$. Then

$$\Delta_F(m) \le \prod_{i=1}^{N} \Delta_F(m) \le \left(\frac{Nem}{d}\right)^d$$

Proof:

For a given sample S, the maximum number of labelling as the F_i vary is the number of ways of choosing the net' s output based on its inputs. Certainly we can' t have more labelling than the number of ways of choosing each node' s outputs independently; we may have fewer if some choices induce the same labelling. Each node has $\Delta F_i(m)$ ways of labelling S. So the whole network has at most $\Delta F_1(m)\Delta F_2(m)$...ways of labelling S. This gives the first inequality in the theorem.

To establish the second inequality, define. Then $d = \sum_{i=1}^{N} d_i$ by definition.

If F has finite VC-Dimension k, then $\Delta F(m) \le \left(\frac{em}{k}\right)^k$. Applying this to each F_i individually, we have that

$$\prod_{i=1}^{N} \Delta F_i(m) \le \prod_{i=1}^{N} \left(\frac{em}{d_i}\right)^{d_i}$$

This product is maximised when all the d_i are equal; when $d_i = d/N$ for all i. So;

$$\prod_{i=1}^{N} \Delta F_i(m) = \prod_{i=1}^{N} \left(\frac{em}{d/N}\right)^{d/N}$$

$$= \left(\frac{em}{d/N}\right)^d$$

For the remainder of this section, we will restrict ourselves to computation nodes that compute linear threshold functions. That is, a node outputs +1 if the weighted sum of its inputs is greater than some threshold and −1 otherwise. The following corollary gives an upper bound on the VC-Dimension of such networks.

Corollary: Let E be the number of edges in a neural net, and W=E+N the total number of weights (one per edge plus the threshold). If N≥2, then

. $\Delta_F(m) \le \left(\frac{Nem}{W}\right)^W$ for m≥W.

$VC - \dim(F) < 2W \log(eN)$

Proof:

It is known that the VC-Dimension of the class of half-spaces in $\Re^n$ is n+1. Since each node computes linear threshold in $\Re^k$,where k ,s the number of inputs to the node, the class of functions computed by a node has VC-dimension at most k+1.
Thus

$d = \sum d_i = \sum VC - \dim(F_i) \le E + N = W.$

Applying theoram1 gives inequality.

If we take $m \ge 2W \log(eN)$, then we can show that

$$\left(\frac{Nem}{W}\right)^W < 2^m$$

As long as N≥2. Using the first part of the corollary, this had shown that $\Delta_F(m) < 2^m$. Thus no set of m points is shattered, so F has VC-Dimension less than m.

Corollary If $m \ge \frac{32W}{\varepsilon} \log \frac{32W}{\varepsilon}$ and we can find choice of weights that correctly labels at least a fraction 1-ε/2 of the main training examples, then with probability at least $1-8^{-1.5w}$, the net will correctly classify at least a fraction 1-ε of examples drawn from the same distribution as the training examples.

Theorem 2.2 : The class of one hidden layer linear threshold nets with k-hidden nodes and n inputs has VC-dimension at least $2\left\lfloor \frac{k}{2} \right\rfloor n$.

Note that this bound is approximately equal to the total number of weights if the net is highly connected (n edges from the inputs to each of the k hidden nodes, plus one edge from each hidden node to the output node, for a total of k (n+1) in a fully-connected net).

We never mentioned back propagation in discussion above. The results hold for an arbitrary training algorithm; in particular, back propagation performs better in practise than these results would suggest.

3. Asymptotic error rates of neural networks

In this chapter, we are interested in determining the error rate of a neural network on a sample as size of the sample and the number of hidden nodes in the net increase. We would like to compare to our actual (or empirical) error rate with the optimal error rate (also known as the Bayes risk) The result below are from [3].

In the discussion below we will consider neural networks with k-hidden nodes in a single hidden layer and d fixed inputs. There is a single output node that outputs either 0 or 1. Each node computes a step function σ of the weighted sum of its input:

$$\sigma = \begin{cases} 1 & if\ x \geq b_i \\ -1 & if\ x < b_i \end{cases}$$

We label each edge from an input to a hidden node by a_i, and each edge from a hidden node to the output node by c_i. Considering a, b and c as column vectors, we can write the output of the net as a function f of its inputs and weights:

$$f(x,a,b,c) = \sum_{i=1}^{k} c_i \sigma(a_i^T x + b_i) + c_0$$

Here x is a vector of the inputs, and c_0 is a constant to the output node.

We can use these neural nets in situations where the inputs don' t completely determines a classification. For example, suppose that the inputs are characteristics of a lung X-ray and the output should predict whether or not a tumor is present. Under such circumstances, there is only a probability (not a certainly) of an output being correct given a set of inputs. It's easy to see the best possible classification rule is the one that agrees with the true outcome most of the time. Let (X,Y) be input/output pair. Then we can write the optimal classification rule as;

$$g^*(x) = \begin{cases} 0 & if\ \Pr\{Y=0 \mid X=x\} \geq 1/2 \\ 1 & if\ \Pr\{Y=1 \mid X=x\} < 1/2 \end{cases}$$

Any given neural network correspondence to some classification rules, given by the mapping of its inputs to its outputs. Let g_n be the classification rule of a neural net after it has seen n examples. This rule will have a certain error rate on the set of training examples denoted by $L(g_n)$, and another error rate on the set of all examples in the sample space, written $r(g_n)$. These can be written formally as;

$L(g_n)$=Pr{ g_n (X)≠Y| some particular training sequence}

$r(g_n)$=E($L(g_n)$)=Pr{ g_n (X) ≠Y}

The optimal classification rule g* has the minimum possible error rate on the whole sample space. We denote this rate by L*, which is also known as the Bayes risk of the sample space.

Definition 3.1. A sequence g_n of classification is universally consistent if

$$\lim_{n\to\infty} r(g_n) = L^*$$

Regardless of the distribution on the sample space X.

A sequence g_n of classification rules is strongly universally consistent if,

$$\lim_{n\to\infty} L(g_n) = L^*$$

With probability one, regardless of the distribution on the sample space X.

Strong universal consistency does indeed imply universal consistency: if a sequence of values chosen from some distribution always converges to a certain value, then the sequence of expected values chosen from the same distribution converges to the same value.

Let $g^*_{k,n}$ denote the optimal neural net with k-hidden nodes after the first n examples.

Theorem 3.1.

If number of the hidden nodes k satisfies $k \to \infty$ and $\frac{k \log n}{n} \to 0$ as $n \to \infty$, then with probability one, $\lim_{n\to\infty} L(g^*_{k,n}) = L^*$.

This theorem is somewhat useful in that it provides a guarantee of strong universal convergence, even

when no neural net is a perfect classifier. However, theorem 1 says nothing about theorem the rate of convergence; the following theorem is more useful in practise;

Theorem 3.2.

If $k = \sqrt{\frac{n}{d \log n}}$, then

$$r(g^*_{k,n}) - L^* = O\left(\left(\frac{d \log n}{n}\right)^{\frac{1}{4}}\right)$$

This theorem tells you roughly how fast the optimal neural net k hidden nodes after examples converge to the optimal rate. Notice that the rate of convergence has a constant exponent; in particular, it doesn't depend on the number of inputs d.

5. Conclusion

In this paper, we considered VC-dimension of neural networks. Some theorems and its proofs have been given. To mean finding VC-dimension of neural networks, most suitable neural networks is to mean to use. For this reason, especially, after this time, if VC-dimension of adaptive neural networks can be found then networks having minimum edge and nodes can be used. This satisfies minimum cost to user.

6. References

1. Sontag E.D. (1992), Feed forward nets for interpolation and classification, Journal of Computer and Systems Sciences, 45, 20-48
2. Vapnik V.N. & Chervonenkis A. (1971). On the convergence of relative frequencies of events to their probabilities, Theory of probability and its application, 2, 264-280
3. Wenocur R. S. & Dideley (1981), some special Vapnik-Chervonenkis classes, discrete Mathematics, 33, 313-318