VC Dimension of Multilayer Neural Networks, Range Queries

Instructors: Sham Kakade and Ambuj Tewari

1 Properties of Growth Function

We had defined the growth function for function class containing $\{\pm 1\}$-valued functions. The definition easily generalizes to the case when the functions take value in some finite set $\mathcal{Y}$. Let $\mathcal{F} \subseteq \mathcal{Y}^{\mathcal{X}}$ be a class of $\mathcal{Y}$-valued functions. Define

$$\Pi_{\mathcal{F}}(m) := \max_{x_1^m \in \mathcal{X}^m} |\mathcal{F}_{|x_1^m}| \ .$$

Note that $\Pi_{\mathcal{F}}(m) \leq |\mathcal{Y}|^m$. We now establish two elementary lemmas that will prove useful while bounding the VC dimension of multilayer neural networks.

Lemma 1.1. *Let $\mathcal{F}^{(1)} \subseteq \mathcal{Y}_1^{\mathcal{X}}$ and $\mathcal{F}^{(2)} \subseteq \mathcal{Y}_2^{\mathcal{X}}$ be two function classes. Let $\mathcal{F} = \mathcal{F}^{(1)} \times \mathcal{F}^{(2)}$ be their cartesian product. Then we have,*

$$\Pi_{\mathcal{F}}(m) \leq \Pi_{\mathcal{F}^{(1)}}(m) \cdot \Pi_{\mathcal{F}^{(2)}}(m) \ .$$

Proof. Fix x_1^m. By definition of cartesian product,

$$\begin{aligned} |\mathcal{F}_{|x_1^m}| &= |\mathcal{F}^{(1)}_{|u_1^m}| \cdot |\mathcal{F}^{(2)}_{|v_1^m}| \\ &\leq \Pi_{\mathcal{F}^{(1)}}(m) \cdot \Pi_{\mathcal{F}^{(2)}}(m) \ . \end{aligned}$$

Since x_1^m was arbitrary, this proves the lemma. $\square$

Lemma 1.2. *Let $\mathcal{F}^{(1)} \subseteq \mathcal{Y}_1^{\mathcal{X}}$ and $\mathcal{F}^{(2)} \subseteq \mathcal{Y}_2^{\mathcal{Y}_1}$ be two function classes. Let $\mathcal{F} = \mathcal{F}^{(2)} \circ \mathcal{F}^{(1)}$ be their composition. The we have,*

$$\Pi_{\mathcal{F}}(m) \leq \Pi_{\mathcal{F}^{(2)}}(m) \cdot \Pi_{\mathcal{F}^{(1)}}(m) \ .$$

Proof. Fix $x_1^m \in \mathcal{X}^m$. By definition of $\mathcal{F}$, we have

$$\begin{aligned} \mathcal{F}_{|x_1^m} &= \left\{ (\ f_2(f_1(x_1)), \ldots, f_2(f_1(x_m))\ ) \,\middle|\, f_1 \in \mathcal{F}^{(1)},\ f_2 \in \mathcal{F}^{(2)} \right\} \\ &= \bigcup_{u \in \mathcal{F}^{(1)}_{|x_1^m}} \left\{ (\ f_2(u_1), \ldots, f_2(u_m)\ ) \,\middle|\, f_2 \in \mathcal{F}^{(2)} \right\} \ . \end{aligned}$$

Therefore,

$$\begin{aligned} |\mathcal{F}_{|x_1^m}| &\leq \sum_{u \in \mathcal{F}^{(1)}_{|x_1^m}} \left| \left\{ (\ f_2(u_1), \ldots, f_2(u_m)\ ) \,\middle|\, f_2 \in \mathcal{F}^{(2)} \right\} \right| \\ &\leq \sum_{u \in \mathcal{F}^{(1)}_{|x_1^m}} \Pi_{\mathcal{F}^{(2)}}(m) \\ &= |\mathcal{F}^{(1)}_{|x_1^m}| \cdot \Pi_{\mathcal{F}^{(2)}}(m) \\ &\leq \Pi_{\mathcal{F}^{(2)}}(m) \cdot \Pi_{\mathcal{F}^{(1)}}(m) \ . \end{aligned}$$

Since x_1^m was arbitrary, this proves the lemma. $\square$

2 VC Dimension of Multilayer Neural Networks

In general, a node ν in a neural network computes a function

$$\sigma(w^{(\nu)} \cdot x - \theta^{(\nu)})$$

of its input x. The function σ is called the *activation function*. Some examples are:

$$\begin{aligned}
\sigma(t) &= \operatorname{sgn}(t) && \text{Binary} \\
\sigma(t) &= \frac{1}{1+e^{-}t} && \text{Sigmoidal} \\
\sigma(t) &= \arctan(t) && \text{Sigmoidal}
\end{aligned}$$

We will consider multilayer neural networks with binary activation function. Somewhat different techniques are needed to get VC dimension bound for networks with sigmoidal activation functions.

Suppose the input space $\mathcal{X} = \mathbb{R}^{d_0}$. A multilayer net with l layers is simply a composition

$$f_l \circ \ldots \circ f_2 \circ f_1(x)$$

where

$$\begin{aligned}
f_i &: \mathbb{R}^{d_{i-1}} \to \{\pm 1\}^{d_i}, && 1 \le i \le l-1 , \\
f_l &: \mathbb{R}^{d_{l-1}} \to \{\pm 1\} .
\end{aligned}$$

Moreover, each component function $f_{i,j} : \mathbb{R}^{d_{i-1}} \to \{\pm 1\}$ is computed as

$$f_{i,j}(u) = \operatorname{sgn}(w^{i,j} \cdot u - \theta^{i,j}) ,$$

where $w^{i,j} \in \mathbb{R}^{d_{i-1}}, \theta^{i,j} \in \mathbb{R}$ are the set of *weights* associated with the jth node in layer i. So, if denote the class of functions (as we vary the weights) computed by this node by $\mathcal{F}^{(i,j)}$, then the class of function associated with layer i is simply

$$\mathcal{F}^{(i)} = \mathcal{F}^{(i,1)} \times \mathcal{F}^{(i,2)} \times \ldots \times \mathcal{F}^{(i,d_i)} .$$

and the class of functions associated with the entire network is

$$\mathcal{F} = \mathcal{F}^{(l)} \circ \ldots \circ \mathcal{F}^{(2)} \circ \mathcal{F}^{(1)} .$$

Thus, we can bound the growth function of $\mathcal{F}$, using Lemmas 1.1 and 1.2, as follows.

$$\begin{aligned}
\Pi_{\mathcal{F}}(m) &\le \prod_{i=1}^{l} \Pi_{\mathcal{F}^{(i)}}(m) \\
&\le \prod_{i=1}^{l} \prod_{j=1}^{d_i} \Pi_{\mathcal{F}^{(i,j)}}(m) \\
&\le \prod_{i=1}^{l} \prod_{j=1}^{d_i} \left(\frac{me}{d_{i-1}+1} \right)^{d_{i-1}+1} ,
\end{aligned}$$

where the last inequality follows by Sauer's lemma and the fact that the VC dimension of halfspaces in d dimensions is $d+1$. If we define

$$N := \sum_{i=1}^{l} \sum_{j=1}^{d_{i-1}} (d_{i-1}+1)$$

to be the *total number of parameters* in the net, then the above inequality implies that

$$\Pi_{\mathcal{F}}(m) \le (me)^N . \tag{1}$$

Now it easy to bound the VC dimension of $\mathcal{F}$.

Theorem 2.1. *Let $\mathcal{F}$ denote the class of functions computed a multilayer neural network as defined above. Then* $\mathrm{VCdim}(\mathcal{F}) = O(N \log_2(N))$.

Proof. Let there be a set of size m that is shattered. Then $\Pi_{\mathcal{F}}(m) = 2^m$. Combining this with (1), we get

$$2^m \leq (me)^N .$$

In order to satisfy this inequality m should be $O(N \log_2(N))$. □

3 VC Dimension and Range Queries

Definition 3.1. *A* range space *is a pair $(S, \mathcal{R})$ where S is a finite or infinite set and $\mathcal{R}$ is a collection of subsets of S.*

Definition 3.2. *A finite set $X \subseteq S$ is* shattered *by $\mathcal{R}$ if*

$$X \cap \mathcal{R} := \{X \cap R \,|\, R \in \mathcal{R}\} = 2^X .$$

Definition 3.3. *The Vapnik-Chervonenkis dimension of $(S, \mathcal{R})$ is the size of a largest shattered set.*

Range queries are very important in computational geometry. An algorithm that answers range queries works as follows. Given a finite set X and query region Q,

$$\begin{aligned} X \cap Q = \emptyset &\Rightarrow \text{Algorithm outputs NO} \\ X \cap Q \neq \emptyset &\Rightarrow \text{Algorithm outputs a witness } x \in S \cap Q \end{aligned}$$

Usually, the region Q is given in some compact form.

For example, consider the case where we have a finite set $X = \{x_1, \ldots, x_n\}$ with $x_i \in \mathbb{R}^d$. Our algorithm will preprocess X using some randomness and parameters ϵ and δ. With probability at least $1 - \delta$, for all queries Q of the form $S_{c,r}$, where

$$S_{c,r} := \left\{x \in \mathbb{R}^d \,\middle|\, \|x - c\| \leq r\right\}$$

is the closed hypersphere with center c and radius r, the algorithm has the following behavior,

$$\begin{aligned} X \cap S_{c,r} = \emptyset &\Rightarrow \text{Algorithm outputs NO} \\ X \cap S_{c,r} \geq \epsilon n &\Rightarrow \text{With probability at least } 1 - \delta, \\ &\quad \text{Algorithm outputs a } x_i \text{ such that } \|x_i - c\| \leq r \end{aligned}$$

Surprisingly, the algorithm runs in time $O\left(\frac{d^2}{\epsilon} \log \frac{d}{\epsilon} + \frac{d}{\epsilon} \log \frac{1}{\delta}\right)$, which is independent of n!

Theorem 3.4. *Let $(S, \mathcal{R})$ be a range space with VC-dimension d, and let $X \subseteq S$ have size n. Suppose N is a random sample of size m drawn from X. If we choose m such that*

$$m \geq \max\left\{\frac{8d}{\epsilon} \log \frac{8d}{\epsilon}, \frac{4}{\epsilon} \log \frac{2}{\delta}\right\}$$

then, with probability at least $1 - \delta$, N is such that

$$\forall R \in \mathcal{R},\ |R \cap X| \geq \epsilon n \Rightarrow R \cap N \neq \emptyset .$$

The algorithms works as follows. It preprocesses X simply by creating a subset N by drawing

$$O\left(\frac{d}{\epsilon} \log \frac{d}{\epsilon} + \frac{1}{\epsilon} \log \frac{1}{\delta}\right)$$

random points from S. On input c, r, if one of these points lies within distance r of c, output that point else say NO. Note that VC dimension of hyperspheres in $\mathbb{R}^d$ is at most $d + 2$. This follows from a theorem we proved last time. The promised time bound now follows by using the above theorem applied to the range space

$$\left(\mathbb{R}^d, \left\{S_{c,r} \,\middle|\, c \in \mathbb{R}^d, r \in \mathbb{R}\right\}\right)$$

and the fact that calculating distances in $\mathbb{R}^d$ takes $O(d)$ time.