COS-511: Learning Theory **Spring 2017**

Lecture 6: Deep Networks (take 1)

Lecturer: Roi Livni

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

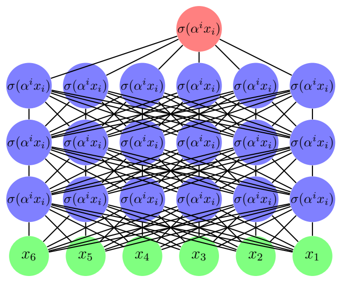

Figure 6.1: A fully connected neural network with 3 hidden layer and an output neuron.

6.1 Expressiveness and Learnability of neural networks

6.1.1 Shallow Networks

To describe the expressive power of neural networks, we begin by showing that without any restriction on the architecture Neural networks are fully expressive. In fact, even 1-hidden networks are sufficiently expressive:

Theorem 6.1. *For every n, there exists a graph (V, E) with 1 hidden layer, such that $\mathcal{N}_{(V,E),\sigma_{\text{sgn}}}$ contains all functions $f : \{\pm 1\}^n \to \pm 1$.*

Proof. We construct a graph with $|V^{(0)}| = n$ and $|V^{(1)}| = 2^n$. We parametrized the nodes at layer $V^{(1)}$ be vectors $\mathbf{u} \in \{-1, 1\}^n$. For every $\mathbf{u} \in \{\pm 1\}$ we define a node $v^{(1)}_{\mathbf{u}}(\mathbf{x}) = \sigma_{\text{sgn}}\left(\sum \mathbf{u}_i \mathbf{x}_i - n\right)$.

One can show that $v^{(1)}_{\mathbf{u}}(\mathbf{x}) = 1$ if and only if $\mathbf{x} = \mathbf{u}$. Now every function $f : \{\pm 1\}^n \to \pm 1$ can be implemented using

$$f(\mathbf{x}) = \left(\sum_{\mathbf{u} \in \{\pm 1\}^n} f(\mathbf{u}) \cdot \frac{(v^{(1)}_{\mathbf{u}}(\mathbf{x}) + 1)}{2} \right) =$$

$$\sigma_{\text{sgn}} \left(\sum_{\mathbf{u} \in \{\pm 1\}^n} \frac{f(\mathbf{u})}{2} \cdot v^{(1)}_{\mathbf{u}}(\mathbf{x}) + \frac{1}{2} \sum f(\mathbf{u}) \right)$$

□

The last theorem states that we can learn ALL boolean functions. However, what will be our sample complexity to learn $\mathcal{N}_{(V,E),\sigma_{\text{sgn}}}$? Note that the class shatters $\{\pm 1\}$ hence has VC dimension 2^n!!! Which means that the sample complexity is 2^n! (In other words, we will need to see all instances over the hypercube inorder to learn – this is not learning, this is memorizing!!!) To summarize

6.1.2 Deep Networks

In this section, we will show that, roughly, neural network have "full" expressive power. We've seen that in general, there is no (learnable) class that can express everything – every algorithm must fail on some instances. However, in terms of computation – we do not really want to express everything.

In practice, a learning algorithm should output a hypothesis that we can then implement on a reasonable machine. For example: if we want to learn a problem of face recognition, after the algorithm has learnt: the output will be a hypothesis that we would implement on our computers or inside, maybe, a camera: Therefore, the hypothesis should be in itself an algorithm that we can efficiently compute: We will show that Neural networks (of reasonable depth and size) have this capacity: namely to express all efficiently computable target functions:

Given the last section, we might want to consider a more restrictive class then *any* network. We next, describe the expressive power of *polynomial size* deep networks.

Theorem 6.2. *Let $T : \mathbb{N} \to \mathbb{N}$ and for every n, let $\mathcal{F}_n$ be the set of functions that can be implemented using a Turring machine using runtime of at most $T(n)$. Then threere exists constant $(b, c) \in \mathbb{R}$ such that for every n, there is a graph (V_n, E_n) of size at most $cT(n)^2 + b$ such that $\mathcal{N}_{(V_n,E_n),\sigma_{\text{sgn}}}$ contains $\mathcal{F}_n$.*

proof sketch. The proof relies on the relation between the time complexity of a program and circuit complexity. Roughly, boolean circuit is a type of network which each individual neuron implements conjunctions, disjunctions and negation of their input. Circuit complexity measures the size of a boolean circuit (in terms of depth and number of neurons) require to calculate the function. The relation between time complexity and circuit complexity can be seen intuitively as follows: We can model each step of the execution of a computer program as a simple operation on its memory state. Therrefore the neuron at each layer reflect the memeory state of the computer at corresponding time. The transition to the next layer of the network invovles a simple calculation that can be carried out by the network.

To relate the complexity of the network to the circuit complexity we need to show that we can implement the operations of conjunction, disjunction, and negation. The negation function can be simply implement via $x_i \to -x_i$. We can implement conjunctions and disjunctions as follows

$$\wedge x_i = \sigma_{\text{sgn}}(\sum_{i=1}^{k} x_i + k - 1) \qquad \vee x_i = \sigma_{\text{sgn}}(\sum_{i=1}^{k} x_i + 1 - k)$$

□

6.1.3 The sample complexity of Neural Netoworks

Next, we discuss the sample complexity of learning the class $\mathcal{N}_{(V,E),\sigma_{\text{sgn}}}$ in view of the last sections, we would like to bound the complexity in terms of the number of edges and neurons.

Theorem 6.3. *The VC dimension of $\mathcal{N}_{(V,E),\sigma_{\text{sgn}}}$ is $O(|E| \log |E|)$.*

To prove Theorem 6.3 We will bound the growth function $\tau_{\mathcal{N}}(m)$. Recall that we used the VC of a class in order to bound its growth function, however as the next Lemma shows, this can also be done in the other direction. Namley

Lemma 6.4. *Let $\mathcal{H}$ be a hypothesis class and assume $\tau_{\mathcal{H}}(m) = m^d$ then VC-dim$(\mathcal{H}) = O(d \log d)$.*

Proof of Lemma. Assume S is a shattered set of size m. There are m^d different labelings of the set S. Since S is shattered we must have $m^d > 2^m$. If $m = 6d \log d$ we have that

$$(6d \log d)^d \leq (6d)^{2d} \leq 2^{6d \log d}$$

. □

We will also need the following facts (which you will prove in your exercise)

For a general class of functions $\mathcal{F}\{f : f : X \to Y\}$, we let the growth function $\tau_{\mathcal{F}}(m)$ of a class of functions $\mathcal{F} \subseteq \{f : X \to Y\}$ is defined as

$$\tau_{\mathcal{F}}(m) = \max_{|S|=m} |\{f : S \to Y : \; \exists f' \in \mathcal{F}, f'(s) = f(s) \forall s \in S\}|$$

1. If $\mathcal{F} = \mathcal{F}_1 \circ \mathcal{F}_2 \circ \cdot \mathcal{F}_t = \{f_1 \circ f_2 \circ \cdots \circ f_m : f_i \in \mathcal{F}_i\}$ then
$$\tau_{\mathcal{F}}(m) \leq \prod_{i=1}^{t} \tau_{\mathcal{F}_t}(m)$$
2. suppose $Y = Y_1 \times, \ldots \times Y_t$ and each $\mathcal{F} = \mathcal{F}_1 \times \cdots \times \mathcal{F}_t$ where $f \in \mathcal{F}$ can be written as $f = (f_1, \ldots, f_t)$. show that
$$\tau_{\mathcal{F}}(m) \leq \prod_{i=1}^{t} \tau_{\mathcal{F}_t}(m)$$

proof of Thm. 6.3. By assigning different weights between $V^{(t)}$ and $V^{(t-1)}$ we obtain different functions from $\mathbb{R}^{|V^{(t-1)}|} \to \{\pm\}^{|V^{(t)}|}$. In other words, we can write $\mathcal{N}_{(V,E)}$ as a composition of funciton classes $\mathcal{F}_1 \circ \mathcal{F}_2 \circ \cdots \mathcal{F}_d$. where d is the number of layers, and each $\mathcal{F}_t$ is the class of all function $f : \mathbb{R}^{|V^{(t-1)}|} \to \{\pm 1\}^{|V^{(t)}|}$ implementable by the structure of the t-th layer. By 1

$$\tau_{\mathcal{N}_{(V,E)}}(m) \leq \prod_{i=1}^{d} \tau_{\mathcal{F}_i}(m)$$

Now each layer t is the a carteszian product of $|V^{(t)}|$ spaces $\mathcal{F}_t = \mathcal{H}_{t,1} \times \mathcal{H}_{t,2} \cdots \mathcal{H}_{t,|V^{(t)}}$. Where $\mathcal{H}_{t,m}$ is given by the target functions parameterised by neuron m in layer t. Using 2 we get

$$\tau_{\mathcal{N}_{(V,E)}}(m) \leq \prod_{i=1}^{d} \tau_{\mathcal{F}_i}(m) \leq \prod_{i=1}^{d} \prod_{j=1}^{|V^{(t)}} \tau_{\mathcal{H}_{t,j}}(m)$$

Finally, note that each $\mathcal{H}_{t,j}$ is an hypothesis of half spaces from its input node to $\{-1, 1\}$ and has VC dimension $|E_{t,j}|$ where $E_{t,j}$ are all edges to the i-th neuron at the t-th layer (see Example. 3.2). We obtain that for some constant C by Sauer's Lemma:

$$\tau_{\mathcal{N}_{(V,E)}}(m) \leq \prod C \cdot m^{|E_{t,j}|} = C \cdot m^{|E|}$$

□