Name: Du Zhixu
University No.: 3035446758
Supervisor: Prof. kangwook Lee (oversea); Prof. Michael K. Ng (domestic)

Undergraduate Research Fellowship Programme (URFP) 2020-21
– Summer Research Internship

Learning Invariant Information in Machine Learning

Du Zhixu Supervisor: Prof. Kangwook Lee (UW Madison);
Prof. Michael K. Ng (HKU)

Department of Mathematics

Abstract

Disentangling domain invariant information has been extensively adopted in domain adaptation in computer vision. It is widely believed that this invariant information will help deep neuron networks to generalize well on different domain. In order to understand whether neural networks memorize this information or they learn how to extract generalizable features, we studied the memorization capacity of neural networks, which serves as an upper bound for generalization error. We investigated two established tight bound of two-layer neural networks[1] and three-layer neuron networks[2], implemented them numerically and measured their norm.

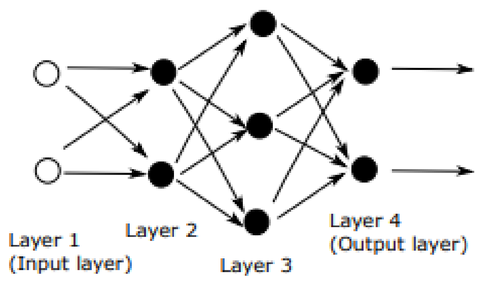

Figure 1

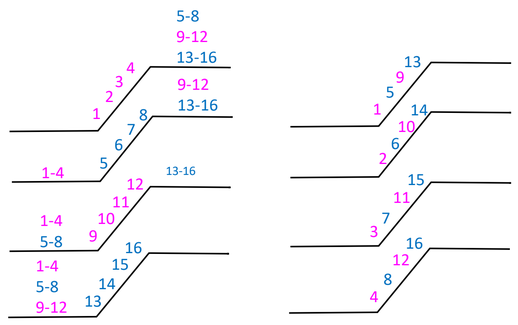

Figure 2

Introduction

A deep neuron network (See Figure 1) can be seen as a composition function. Several neurons are contained in each layer serving as computational units. Neurons in different layers are connected by a weight matrix, a bias vector and an activation function $\sigma: \mathbb{R} \longrightarrow \mathbb{R}$ applying component-wisely, which maps the vector form $\mathbb{R}^{n_l}$ to $\mathbb{R}^{n_{l+1}}$. Consider the network in Figure 1, which is a three-layer neural network. Let $x \in \mathbb{R}^2$ be the input value, $W_1 \in \mathbb{R}^{2\times 2}$ and $b_1 \in \mathbb{R}^2$ represent weights and biases connecting layer1 and layer 2, respectively. Then computation form layer 1 to layer 2 can be represented by $\sigma(W_1 x + b_1) \in \mathbb{R}^2$, where σ is an activation function. Similarly, the output of this network can be written as:

$$f_\theta(x) = W_3\sigma(W_2\sigma(W_1 x + b_1) + b_2) + b_3 \in \mathbb{R}^2$$

where θ includes all parameters. Also, it can be regarded as:

$$f_\theta(x) = f_3 \circ f_2 \circ f_1(x)$$

The capability of a network is influenced by the number of layers and the number of neurons in each layer. The function ReLU $(\sigma(x) = \begin{cases} x \ if \ x > 0 \\ 0 \ if \ x \le 0 \end{cases})$ is a general choice for activation function. In a network, we can optimize those parameters by minimizing a loss function $\mathcal{L}(x;\theta)$ using gradient descent and its variations.

The memorization capacity of a network is defined as the size of the largest dataset that the network can memorize, i.e. we are going to find θ s.t. $f_\theta(x) = y$, for any $(x, y) \in D$. Due to floating point error, we can not obtain this θ by traditional deep learning training techniques. Therefore, we need to explicitly construct weights θ. [1] and [2] constructed the weights for two-layer neural networks and three-layer neural networks, respectively. Besides, [1] points out the relationship between generalization performance and memorization capacity: memorization capacity is proportional to an upper bound of generalization error. The underlying idea behind these proof is to assign each data point an unique activation pattern. For each neuron, if its activation function produces an non-trivial value (e.g $\sigma(x)\ for\ x > 0$), we say that this neuron is activated, and otherwise, we say it is deactivated (e.g $\sigma(x)\ for \le 0$). When each data point goes through a network, it will activate some neurons and deactivate others. We call this pattern as *activation pattern*.

Memorization capacity of two-layer neural networks

Consider a two-layer neural network with ReLU activation function with n neurons in layer 2 and 1 neuron in layer 3:

$$f_\theta = W_2\sigma(W_1 x - \boldsymbol{b_1})$$

Consider a dataset $D = \{(x_i, y_i)\}_{i=1}^n$, where $y_i \in \mathbb{R}$. We first project the input $x \in \mathbb{R}^d$ to $\mathbb{R}$ by a randomly generated vector $\boldsymbol{a} \in \mathbb{R}^d \sim N(0, I_d)$. Since the projection vector is randomly generated following Gaussian distribution, the probability that two data point is projected to the same real number is 0. We select this $W_1 = [\boldsymbol{a}, \boldsymbol{a}, \dots, \boldsymbol{a}]^{\boldsymbol{T}} \in \mathbb{R}^{\boldsymbol{n\times d}}$ as the weight vector for layer 1. WLOG, we get

$$\boldsymbol{a}^T x_1 < \boldsymbol{a}^T x_2 < \cdots < \boldsymbol{a}^T x_n$$

We select $\boldsymbol{b}_1$ such that

$$b_1 < \boldsymbol{a}^T x_1 < b_2 < \boldsymbol{a}^T x_2 < \cdots < b_n < \boldsymbol{a}^T x_n$$

Then we obtain the following matrix indicating activation pattern after plug in all n data points

$$A = \begin{bmatrix} \boldsymbol{max}\{\boldsymbol{a}^T x_1 - b_1, 0\} & 0 & \cdots & 0 \\ \boldsymbol{max}\{\boldsymbol{a}^T x_2 - b_1, 0\} & \boldsymbol{max}\{\boldsymbol{a}^T x_2 - b_2, 0\} & & 0 \\ \vdots & & \ddots & \vdots \\ \boldsymbol{max}\{\boldsymbol{a}^T x_n - b_1, 0\} & \boldsymbol{max}\{\boldsymbol{a}^T x_n - b_2, 0\} & \cdots & \boldsymbol{max}\{\boldsymbol{a}^T x_n - b_n, 0\} \end{bmatrix}$$

It is easy to see that A is invertible, hence we select $W_2 = A^{-1}\boldsymbol{y}$. A is an $n \times n$ matrix, each row represent the activation pattern for one data point while each column indicates the activation pattern for one neuron w.r.t. all data points. We can see from the matrix that the i-th data point is activated by $1,2,\dots,i$-th neuron, which guarantees that each data point will obtain a unique activation pattern. However, note that, it is not guaranteed that any activation pattern can achieve memorization.

Memorization capacity of three-layer neural networks

Consider a two layer neural network with hard-tanh activation function

$$\sigma(x) = \begin{cases} -1 & x \le -1 \\ x & x \in (-1,1] \\ 1 & x > 1 \end{cases}$$

with p neurons in layer 2, q neurons in layer 3 and 1 neuron in layer 4

$$f_\theta = W^3\sigma(W^2\sigma(W^1 x + b^1) + b^2) + b^3$$

For convenience, let $z^l(x) = W^l a^{l-1}(x) + b^l$ and $a^l(x) = \sigma(z^l(x))$, where $a^0(x) = x$. Consider a dataset $D = \{(x_i, y_i)\}_{i=1}^n$, where $y_i \in \mathbb{R}$ and $x_i \in \mathbb{R}^d$. The idea of construction is to assign a unique activation pattern for each data point. For the network of our interest, we have pq different activation patterns, hence let $n = pq$. For illustration of idea, let's consider a special case with $p = q = 4$, and in total 16 data points. A natural assignment is shown in Figure 2. Therefore, our first step is to construct weights and biases of the first layer such that the activation pattern is the same as shown in Figure 2. We wish to separate these 16 data points into 4 groups (in general p groups). Like before, u is a randomly generated vector following some distributions, hence $u^T x_i = u^T x_j$ has probability 0. WLOG, we have

$$u^T x_1 < u^T x_2 < \cdots < u^T x_n$$

Denote $c_i = u^T x_i$ and $c_0 = c_1 - \delta$, $c_{n+1} = c_n + \delta$ where $\delta > 0$.
We select the weights in the following way:

$$W^1_{j,:} = \frac{4}{c_{jq} + c_{jq+1} - c_{jq-q} - c_{jq-q+1}} u^T$$

$$b^1_j = -\frac{c_{jq} + c_{jq+1} + c_{jq-q} + c_{jq-q+1}}{c_{jq} + c_{jq+1} - c_{jq-q} - c_{jq-q+1}}$$

$$\Rightarrow z^1_j\ (x_i) = \frac{4u^T x_i - (c_{jq} + c_{jq+1} + c_{jq-q} + c_{jq-q+1})}{c_{jq} + c_{jq+1} - c_{jq-q} - c_{jq-q+1}}$$

where $W^1_{j,:}$ is the j-th row of matrix W^1. An illustration is shown in Figure 3, with which it is easy to see that this construction achieves our goal.

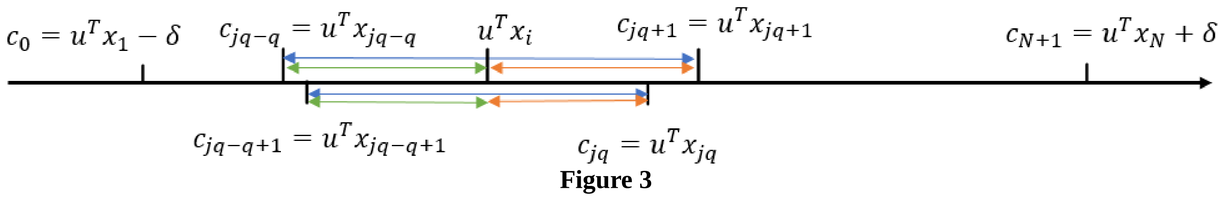

Figure 3

Then we group the k-th data point in each group of first layer as a new group for second layer's activation. Then we have in order to memorize all data points, weights in second layer must satisfy following linear system of equations

$$M\begin{bmatrix} W^{2\ T}_{k,:} \\ b^2_k \end{bmatrix} = \begin{bmatrix} a^1_1(x_{i_{k,1}}) & -1 & \cdots & -1 & 1 \\ 1 & a^1_2(x_{i_{k,2}}) & & -1 & 1 \\ \vdots & & \ddots & \vdots & \\ 1 & 1 & & -1 & 1 \\ 1 & 1 & \cdots & a^1_p(x_{i_{k,p}}) & 1 \end{bmatrix}\begin{bmatrix} W^{2\ T}_{k,:} \\ b^2_k \end{bmatrix} = \begin{bmatrix} y_{i_{k,1}} \\ y_{i_{k,2}} \\ \vdots \\ y_{i_{k,p-1}} \\ y_{i_{k,p}} \end{bmatrix}$$

It is easy to see that M is of full row rank which guarantees the system has a solution. Besides, there exist an v such that $v_1, \dots, v_p$ are not all zero, in particular, $v_{odd} > 0$, $v_{even} < 0$ for 1 to p and $Mv = 0$. We can rewrite $z^2_k(x_i) = \sum_{l=1}^p W^2_{k,l} a^1_l(x_i) + b^2_k$ as $z^2_k(x_i) = y_{i_{k,j}} + W^2_{k,j_i}\left(a^1_{j_i}(x_i) - a^1_{j_i}\left(x_{i_{k,j_i}}\right)\right)$ since the system must have a solution, denoted as μ. In order to achieve the activation pattern shown in Figure 2, we take $\begin{bmatrix} W^{2\ T}_{k,:} \\ b^2_k \end{bmatrix} = \mu + \alpha v$ where $v_{odd} > 0$, $v_{even} < 0$ and $Mv = 0$ and pick α large enough that if x_i is not the k-th datapoint for any group, then $z^2_k(x_i) \in (-1,1)^C$. Then the activation pattern is shown in Figure 4. For the final output layer, we simply take the summation of all neurons. Then for data point from pink group, there will be a $+1$ offset, and for data points from blue group there is a -1 offset. Hence, we need one more neuron for de-noising purpose. Its activation pattern is shown in Figure 5. By consider a group of ghost data points, we can achieve this purpose. Specifically, consider $z^2_{q+1}(x_i) = 0 + W^2_{q+1,j_i}\left(a^1_{j_i}(x_i) - a^1_{j_i}\left(x_{i_{j_i q}} + \epsilon\right)\right)$.

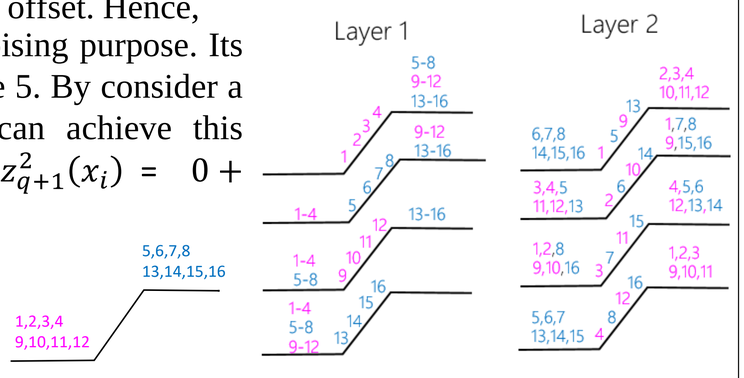

Figure 5 **Figure 4**

Reference list:

[1] Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.

[2] Yun, C., Sra, S., & Jadbabaie, A. (2019). Small ReLU networks are powerful memorizers: a tight analysis of memorization capacity. In *Advances in Neural Information Processing Systems* (pp. 15558-15569).