Kernelization on Neural Network

Shuai Zhang Yingchun Zhang

July 28, 2016

1 Introduction

1.1 Related Works

Wilson et al. [1] combined the non-parametric flexibility of kernel methods with the structural properties of deep neural networks.

2 Methods

2.1 Neural Network

In a two-layer network, the feed-forward network function is

$$\mathbf{y}_k(\mathbf{x}, \mathbf{w}) = h^{(2)} \left(\sum_{j=0}^{M} w_{kj}^{(2)} h^{(1)} \left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i \right) \right). \tag{1}$$

Given a training set with input vectors $\{\mathbf{x}_n\}$, where $n = 1, ..., N$, together with a correspounding set of target vectors $\{\mathbf{t}_n\}$, we minimize the error function

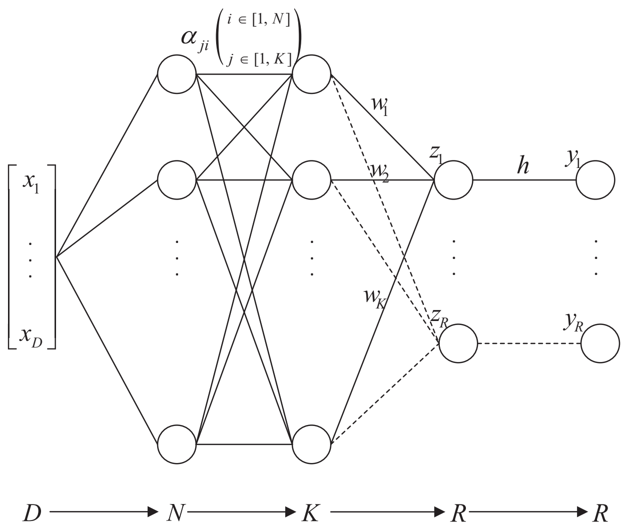
$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \|\mathbf{y}(\mathbf{x}_n, \mathbf{w}) - \mathbf{t}_n\|^2. \tag{2}$$

2.2 Kernelized Neural Network

Progress @ July 24

2.2.1 Nonparametric representation

Single-layer neural network



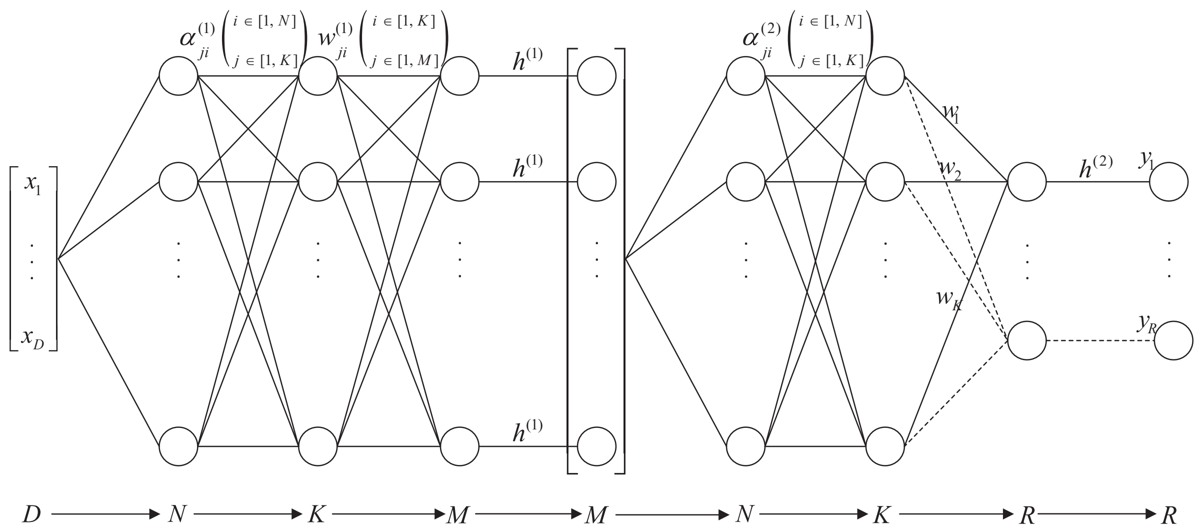
In a single-layer neural network, the feed-forward network function is

$$y_n = h\left(\sum_{j=1}^{K} w_j\left(\sum_{i=1}^{N} \alpha_{ji} k(\mathbf{x}_i, \mathbf{x}_n)\right)\right) \tag{3}$$

where $\{\mathbf{x}_i\}_{i=1}^{N}$ is the unique sample which is a D-dimension vector of the dataset, k is a kernel function which builds a new inner product space, and we try to get k linear combination of with the weight α_{ji}, then we make a linear combination of the new base with the weight w_j. h is an activation function and so as follows. All the w, h, α, k and $\mathbf{x}$ shown in the following part have the same meaning as the former ones.

Multi-layer neural network

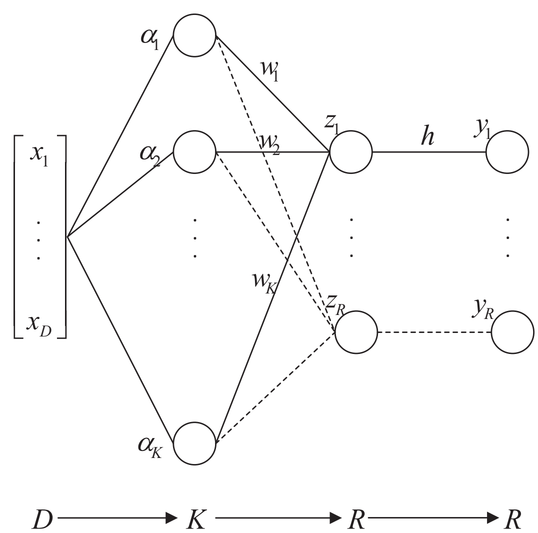
$$y_n = h^{(2)}\left(\sum_{j=1}^{K} w_j^{(2)}\left(\sum_{i=1}^{N} \alpha_{ji}^{(2)} k_2(\mathbf{x}_i, \begin{bmatrix} h^{(1)}\left(\sum_{j=1}^{K} w_{1j}^{(1)}\left(\sum_{i=1}^{N} \alpha_{ji}^{(1)} k_1(\mathbf{x}_i, \mathbf{x}_n)\right)\right) \\ \vdots \\ h^{(1)}\left(\sum_{j=1}^{K} w_{Mj}^{(1)}\left(\sum_{i=1}^{N} \alpha_{ji}^{(1)} k_1(\mathbf{x}_i, \mathbf{x}_n)\right)\right) \end{bmatrix})\right)\right) \tag{4}$$



The number in superscript of the symbols represents the layer of neural network, such as, $w_j^{(2)}$ means the weight of the second layer of neural network and so as follows.

2.2.2 Parametric representation

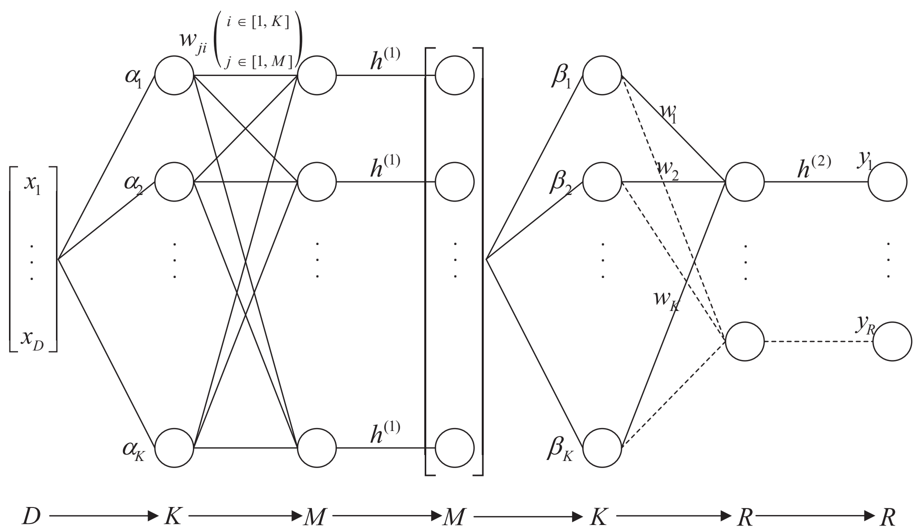
Single-layer neural network



$$y_n = h\left(\sum_{j=1}^{K} w_j k(\boldsymbol{\alpha}_j, \mathbf{x}_n)\right) \tag{5}$$

$\boldsymbol{\alpha}_j$ is the center of kernel which is a D-dimension vector.

Multi-layer neural network



$$y_n = h^{(2)}\left(\sum_{j=1}^{K} w_j^{(2)} k_2(\boldsymbol{\alpha}_j^{(2)}, \begin{bmatrix} h^{(1)}\left(\sum_{i=1}^{K} w_{1i}^{(1)} k_1(\boldsymbol{\alpha}_i^{(1)}, \mathbf{x}_n)\right) \\ \vdots \\ h^{(1)}\left(\sum_{i=1}^{K} w_{Mi}^{(1)} k_1(\boldsymbol{\alpha}_i^{(1)}, \mathbf{x}_n)\right) \end{bmatrix})\right) \tag{6}$$

Progress @ July 26

We aim to find a set of components $F = \{f_i\}_{i=1}^K$ which are fucntions in a RKHS $\mathcal{H}$ induced by a kernel function $k(\cdot,\cdot)$. Given $k(\cdot,\cdot)$ and the input space $\mathcal{X} = \mathbb{R}^D$, a reproducing kernel feature map [2] $\phi : \mathcal{X} \to \mathbb{R}^{\mathcal{X}}$ can be defined as $\phi(\mathbf{x}) = k(\mathbf{x}, \cdot)$ and the inner product $\phi(\mathbf{x})^T\phi(\mathbf{y})$ can be computed as $k(\mathbf{x}, \mathbf{y})$ via the kernel trick. Firstly, we define the kernel neural network problem

$$\min \sum_{n=1}^{N} \|y_n - t_n\|^2, \tag{7}$$

where t_n is the lable of the sample $\mathbf{x}_n$, and y_n is the output of the kernel neural network with input sample $\mathbf{x}_n$. Then we talk the computation of y_n in parametric form and non-parametric form separately.

2.2.3 Parametric Representation

Under the parametric representation, the functions f take a parametric form which is independent of data: $\{f | f = \phi(\mathbf{a}), \mathbf{a} \in \mathbb{R}^D\}$. So we can compute $f(\mathbf{x})$ as $f(\mathbf{x}) = \phi(\mathbf{a})^T\phi(\mathbf{x}) = k(\mathbf{a}, \mathbf{x})$.

For single-layer neural network we have:

$$\phi(\mathbf{x}) = k(\mathbf{x}, \cdot) \tag{8}$$

$$f_i = \phi(\mathbf{a}_i) = k(\mathbf{a}_i, \cdot) \tag{9}$$

$$f_i(\mathbf{x}) = f_i(\cdot)\phi(\mathbf{x}) = k(\mathbf{a}_i, \cdot)k(\mathbf{x}, \cdot) = k(\mathbf{a}_i, \mathbf{x}) \tag{10}$$

$$y = \sum_{i=1}^{K} w_i f_i(\mathbf{x}) = \sum_{i=1}^{K} w_i k(\mathbf{a}_i, \mathbf{x}) \tag{11}$$

And for multi-layer neuarl network we have:

$$\phi^{(2)}(\mathbf{x}^{(2)}) = k^{(2)}(\mathbf{x}^{(2)}, \cdot) \tag{12}$$

$$f_i^{(2)} = \phi^{(2)}(\mathbf{a}_i^{(2)}) = k^{(2)}(\mathbf{a}_i^{(2)}, \cdot) \tag{13}$$

$$f_i^{(2)}(\mathbf{x}^{(2)}) = f^{(2)}(\cdot)\phi^{(2)}(\mathbf{x}^{(2)}) = k^{(2)}(\mathbf{a}_i^{(2)}, \cdot)k^{(2)}(\mathbf{x}^{(2)}, \cdot) = k^{(2)}(\mathbf{a}_i^{(2)}, \mathbf{x}^{(2)}) \tag{14}$$

$$y = \sum_{i=1}^{K} w_i^{(2)} f_i^{(2)}(\mathbf{x}^{(2)}) = \sum_{i=1}^{K} w_i^{(2)} k^{(2)}(\mathbf{a}_i^{(2)}, \mathbf{x}^{(2)}) \tag{15}$$

$$\mathbf{x}^{(2)} = \begin{bmatrix} k(\mathbf{a}_1, \mathbf{x}) \\ . \\ . \\ . \\ k(\mathbf{a}_K, \mathbf{x}) \end{bmatrix} \tag{16}$$

2.2.4 Non-parametric Representation

Under the non-parametric representation, the functions f admit a non-parametric form which is data dependent: $\{f|f = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \cdot), \alpha_i \in \mathbb{R}\}$. So we can compute $f(\mathbf{x})$ as $f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x})$.

For single-layer neural network we have:

$$\phi(\mathbf{x}) = k(\mathbf{x}, \cdot) \tag{17}$$

$$f_i = \sum_{i=1}^{N} \alpha_{ji} k(\mathbf{x}_i, \cdot) \tag{18}$$

$$f_i(\mathbf{x}) = f_i(\cdot)\phi(\mathbf{x}) = \sum_{i=1}^{N} \alpha_{ji} k(\mathbf{x}_i, \cdot) k(\mathbf{x}, \cdot) = \sum_{i=1}^{N} \alpha_{ji} k(\mathbf{x}_i, \mathbf{x}) \tag{19}$$

$$y = \sum_{i=1}^{K} w_i f_i(\mathbf{x}) = \sum_{i=1}^{K} w_i \sum_{i=1}^{N} \alpha_{ji} k(\mathbf{x}_i, \mathbf{x}) \tag{20}$$

And for multi-layer neural network we have:

$$\phi^{(2)}(\mathbf{x}^{(2)}) = k^{(2)}(\mathbf{x}^{(2)}, \cdot) \tag{21}$$

$$f_i^{(2)} = \sum_{i=1}^{N} \alpha_{ji}^{(2)} k^{(2)}(\mathbf{x}_i^{(2)}, \cdot) \tag{22}$$

$$f_i^{(2)}(\mathbf{x}^{(2)}) = f_i^{(2)}(\cdot)\phi^{(2)}(\mathbf{x}^{(2)}) = \sum_{i=1}^{N} \alpha_{ji}^{(2)} k^{(2)}(\mathbf{x}_i^{(2)}, \cdot) k^{(2)}(\mathbf{x}^{(2)}, \cdot) = \sum_{i=1}^{N} \alpha_{ji}^{(2)} k^{(2)}(\mathbf{x}_i^{(2)}, \mathbf{x}^{(2)}) \tag{23}$$

$$y = \sum_{i=1}^{K} w_i^{(2)} f_i^{(2)}(\mathbf{x}^{(2)}) = \sum_{i=1}^{K} w_i^{(2)} \sum_{i=1}^{N} \alpha_{ji}^{(2)} k^{(2)}(\mathbf{x}_i^{(2)}, \mathbf{x}^{(2)}) \tag{24}$$

$$\mathbf{x}^{(2)} = \begin{bmatrix} \sum\limits_{i=1}^{N} \alpha_{1i} k(\mathbf{x}_1, \mathbf{x}) \\ . \\ . \\ . \\ \sum\limits_{i=1}^{N} \alpha_{Ki} k(\mathbf{x}_K, \mathbf{x}) \end{bmatrix} \tag{25}$$

Progress @ July 28

Firstly, we define the kernel neural network problem:

$$\min_{\{\mathbf{F}^{(i)}\subseteq\mathcal{H}\}_{i=1}^{m}} \sum_{n=1}^{N} \ell(\{f_{i^{(m)}}^{(m)}(\{f_{i^{(m-1)}}^{(m-1)}(\cdots(\{f_{i^{(1)}}^{(1)}(\mathbf{x}_n)\}_{i^{(1)}=1}^{K^{(1)}})\cdots)\}_{i^{(m-1)}=1}^{K^{(m-1)}})\}_{i^{(m)}=1}^{K^{(m)}}) - \lambda\Omega(\{\mathbf{F}^{(i)}\}_{i=1}^{m}) \quad (26)$$

Parametric representation form

$$\begin{aligned}\min_{\{\mathbf{A}^{(i)}\}_{i=1}^{m}} \sum_{n=1}^{N} \ell(\{k^{(m)}(\mathbf{a}_{i^{(m)}}, \{k^{(m-1)}(\mathbf{a}_{i^{(m-1)}}, \cdots \{k^{(1)}(\mathbf{a}_{i^{(1)}}, \mathbf{x}_n)\}_{i^{(1)}=1}^{K^{(1)}} \cdots)\}_{i^{(m-1)}=1}^{K^{(m-1)}})\}_{i^{(m)}=1}^{K^{(m)}}) \\ -\lambda\Omega(\{\mathbf{A}^{(i)}\}_{i=1}^{m}, \{k^{(j)}\}_{j=1}^{m})\end{aligned} \quad (27)$$

where $\mathbf{A} = \{\mathbf{a}_i\}_{i=1}^{K}$

Non-parametric representation form

$$\min_{\{\mathbf{r}^{(i)}\}_{i=1}^{m}} \sum_{n=1}^{N} \ell(\{\sum_{j^{(m)}=1}^{n} \alpha_{i^{(m)}j^{(m)}}^{m} k^{(m)}(\mathbf{x}_{j^{(m)}}^{(m)}, \mathbf{x}_n^{(m)})\}_{i^{(m)}=1}^{K^{(m)}}) - \lambda\Omega(\{\mathbf{r}^{(i)}\}_{i=1}^{m}, \{k^{(j)}\}_{j=1}^{m}) \quad (28)$$

where $\mathbf{r} = \{\boldsymbol{\alpha}_i\}_{i=1}^{K}$ and:

$$\begin{aligned}\mathbf{x}_{j^{(m)}}^{(m)} &= \{\sum_{j^{(m-1)}=1}^{N} \boldsymbol{\alpha}_{i^{(m-1)}j^{(m-1)}}^{(m-1)} k^{(m-1)}(x_{j^{(m-1)}}^{(m-1)}, x_{j^{(m)}}^{(m-1)})\}_{i^{(m-1)}=1}^{k^{(m-1)}} \\ &. \\ &. \\ &. \\ \mathbf{x}_{j^{(2)}}^{(2)} &= \{\sum_{j^{(1)}=1}^{N} \boldsymbol{\alpha}_{i^{(1)}j^{(1)}}^{(1)} k^{(1)}(x_{j^{(1)}}^{(1)}, x_{j^{(2)}}^{(1)})\}_{i^{(1)}=1}^{k^{(1)}}\end{aligned} \quad (29)$$

References

[1] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, "Deep kernel learning," *CoRR*, vol. abs/1511.02222, 2015.

[2] B. Schölkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.