Poor Global Optima for Fully Connected Deep ReLU Neural Networks - Special Examples for HD Approximation

Qingguang Guan, Wenqing Hu and etc.

Department of Mathematics and Statistics
Missouri University of Science and Technology

DASIV Spring School, March 18, 2019
University of South Carolina

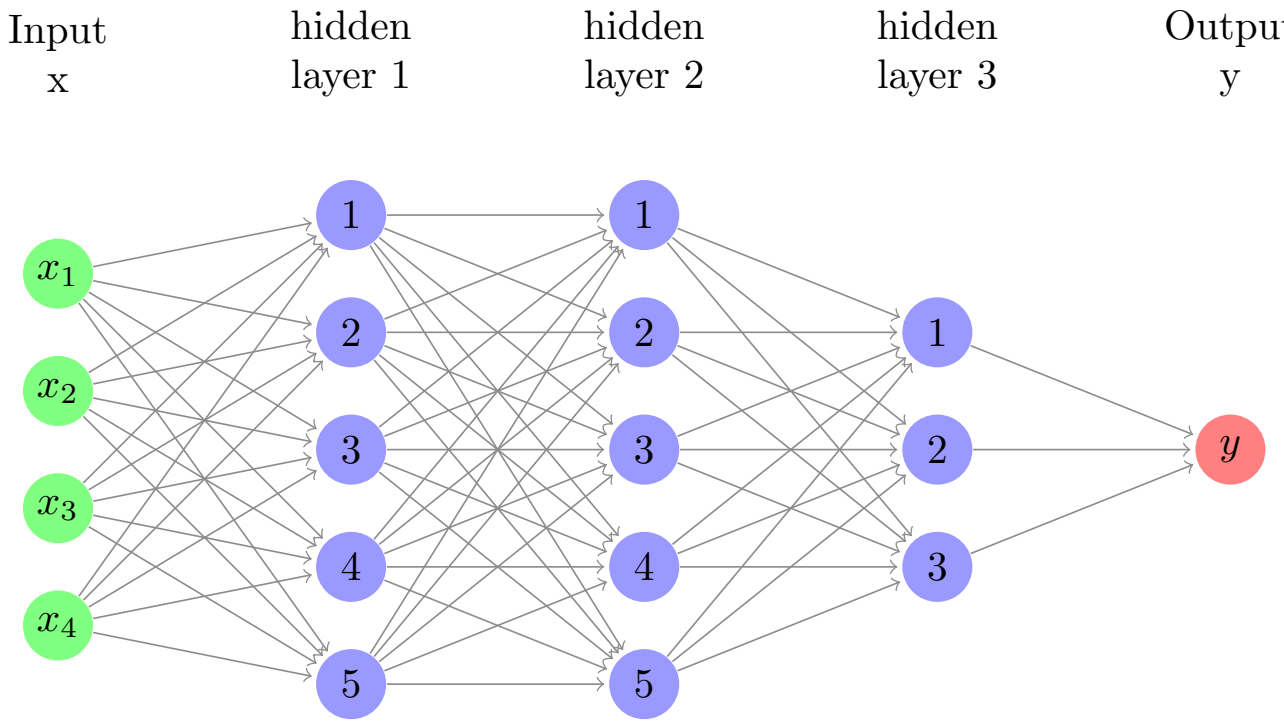


Figure: A simple fully connected Deep ReLU Neural Network

Take the first neuron in hidden layer 1 as an example, the input of it is $\sum_{i=1}^{4} w_{1,i}x_i + b_1$, the output is $\max(0, \sum_{i=1}^{4} w_{1,i}x_i + b_1)$.

How good it can be?

As good as continuous piece-wise linear approximation.

- Dmitry Yarotsky, Optimal approximation of continuous functions by very deep ReLU networks, *Proceedings of Machine Learning Research.* Volume **75**, pages:1–11, 2018.
- J. He, L. Li, J. Xu and C. Zheng, ReLU Deep Neural Networks and Linear Finite Elements, *arXiv preprint.* arXiv:1807.03973v2, 2018.

How bad it can be?

We will use the same setting as in those two papers which employ $[0, 1]^D$ hyper cube and uniform grid to show that there exists an approximation function which has exact values on grid points however on each point $\mathbf{x}$ if $\mathrm{dist}(\mathbf{x}, \text{grid points}) > \epsilon$ for a given $\epsilon > 0$, the function's value is 0.

Let us consider a function $f(x), x \in [0,1]^D$ that we want to approximate. We are given training data $(x_i, f(x_i))$, $i = 1, 2, ..., N$, where x_i is grid point. We construct deep ReLU neural network with approximation function $f_h(x) = f_h(x; \omega^*)$ where ω^* represents the weights and biases. Our ReLU neural network fits the training data perfectly, we have

$$\frac{1}{N} \sum_{i=1}^{N} (f(x_i) - f_h(x_i; \omega^*))^2 = 0 \ .$$

In other words, ω^* is a global optima on training data. We then show that given an arbitrary test data $(x, f(x))$ where x is randomly sampled, our model will have the worst generalization error.

One Dimensional Function Approximation-Special Case

We start with one-dimensional input variable $x \in \mathbb{R}$, for fully connected neural network with two hidden layers, and ReLU activation function. Suppose we have a set $\{x_i | x_i \in [0,1], i = 1, 2, \cdots, N\}$, x_i is corresponding to label -1, if $x_i < 0.5$ and 1 if $x_i \geq 0.5$.
We define the one dimensional basis function as:

$$\phi(\xi) = \frac{1}{h} a(\xi + h) - \frac{2}{h} a(\xi) + \frac{1}{h} a(\xi - h), \tag{1}$$

where $\xi \in [0,1], h < 1$, $a(\xi)$ is the ReLU function. For x_i we define the basis function as

$$\phi_i(x) = \phi(x - x_i). \tag{2}$$

where $\phi_i(x)$ has the height 1, and compact support $[x_i - h, x_i + h]$.

For the first hidden layer, if we have $2N+1$ neurons, each neuron has distinct input as: $x - x_i$ or $x - x_i - h$ or $x - x_i + h$, $i = 1, 2, \cdots, N$. Then, using the output of first hidden layer, we can build basis functions $\phi_i(x), i = 1, 2, \cdots, N$, and they have non-overlap compact support.
For the second hidden layer, suppose we have 2 neurons, the input of first neuron is:

$$I_1(x) := \left(\sum_{i \text{ for } x_i \geq 0.5} \phi_i(x) \right) - b \tag{3}$$

where b is the bias, it's easy to see, $I_1(x)$ is the linear combination of first hidden layer's output. Similarly, we have the input of second neuron:

$$I_2(x) := \left(\sum_{i \text{ for } x_i < 0.5} \phi_i(x) \right) - b. \tag{4}$$

Let $b \in [0, 1)$, then the final output is:

$$f_h(x) := \frac{a(I_1(x))}{1-b} + (-1)\frac{a(I_2(x))}{1-b}.$$

Graphs for 1-D Examples-Special Case

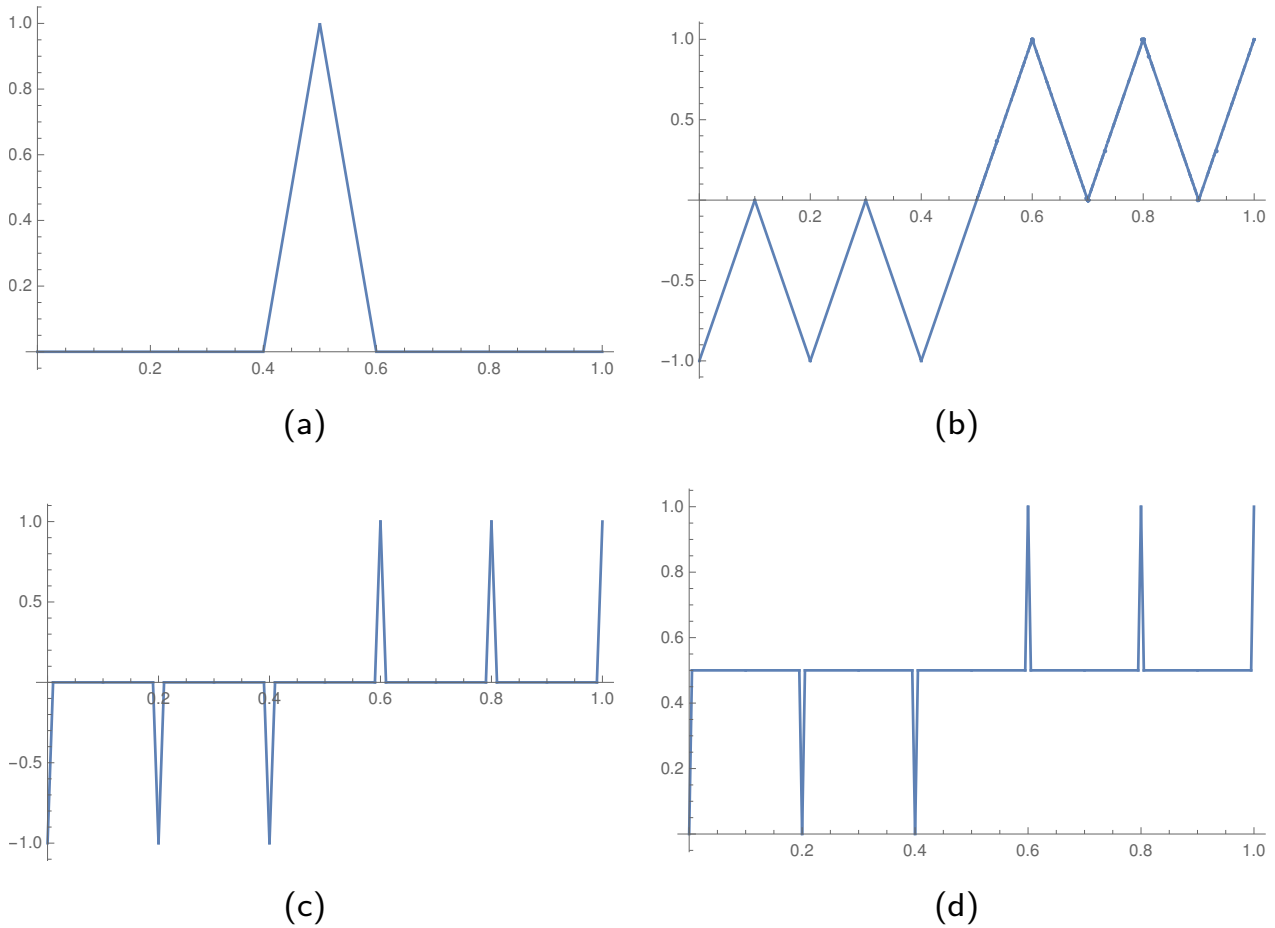


Figure: (a) $\phi_i(x)$, $x_i = 0.5$, $h = 0.1$; (b) $f_h(x)$ when $N = 6$, $b = 0$; (c) $f_h(x)$ when $N = 6$, $b = 0.9$; and, (d) $f_h(x)$ for Cross Entropy when $N = 6$, $b = 0.95$.

To approximate a continuous function $f(x)$, $x \in [0,1]$. For the first hidden layer, we need $N+2$ neurons, each neuron has distinct input as: $x - x_i$ or $x - x_i - h$ or $x - x_i + h$, $i = 1, 2, \cdots, N$. Then, using the output of first hidden layer, we can build basis functions $\phi_i(x), i = 1, 2, \cdots, N$. Then we need N neurons in second hidden layer and the final output is:

$$f_h(x) := \sum_{i=1}^{N} f(x_i) \frac{a(\phi_i(x) - b)}{1 - b},$$

where $b \in [0, 1)$. So as $b \to 1$, the approximation will fail though it is the global optimal solution.
How about deeper neural networks? It doesn't matter how deep it is as long as it has the same first two hidden layers and at least two neurons for each later hidden layers. We can divide the second layer's output $f_h(x)$ into two parts, positive part and negative part then make the negative part positive, pass two parts separately into the next layer's two different neurons. And the inputs for all other neurons are 0. For final output, we can assign the correct sign (+/-) and combine them together.

Two Dimensional Function Approximation-Special Case

In this section, we will build two dimensional "basis functions" based on one dimensional ones. The input variable is $(x, y) \in \mathbb{R}^2$, the region is $[0, 1] \times [0, 1]$. We have the training set data $\{(x_i, y_j), i, j = 1, 2, \cdots, N\}$, where
$x_1 = y_1 = 0, x_N = y_N = 1$, (x_i, y_j) is the grid point.
The point (x_i, y_j) in data set has label -1, if $x_i < 0.5$ and 1 if $x_i \geq 0.5$. The size of data is $N \times N$. Denote $\phi_i(x) = \phi(x - x_i)$ and $\phi_j(y) = \phi(y - y_j)$. Then we can define the 2D "basis function" as:

$$\Phi(x, y) = \sum_{\{x_i\}} \phi_i(x) + \sum_{\{y_j\}} \phi_j(y),$$

where sets $\{x_i\}, \{y_j\}$ are chosen grid points.

Let $x_{i+1} - x_i = y_{j+1} - y_j = 2h > 0$, for the first hidden layer, we need $2(2N+1)$ neurons, each neuron has distinct input as: $x - x_i$ or $x - x_i - h$ or $x - x_i + h$; $y - y_j$ or $y - y_j - h$ or $y - y_j + h$, $i, j = 1, 2, \cdots, N$.
For second hidden layer, we need 2 neurons, the input of first neuron is:

$$I_1(x, y) := \sum_{x_i \geq 0.5} \phi_i(x) + \sum_{j=1}^{N} \phi_j(y) - b \tag{5}$$

The input of second neuron is:

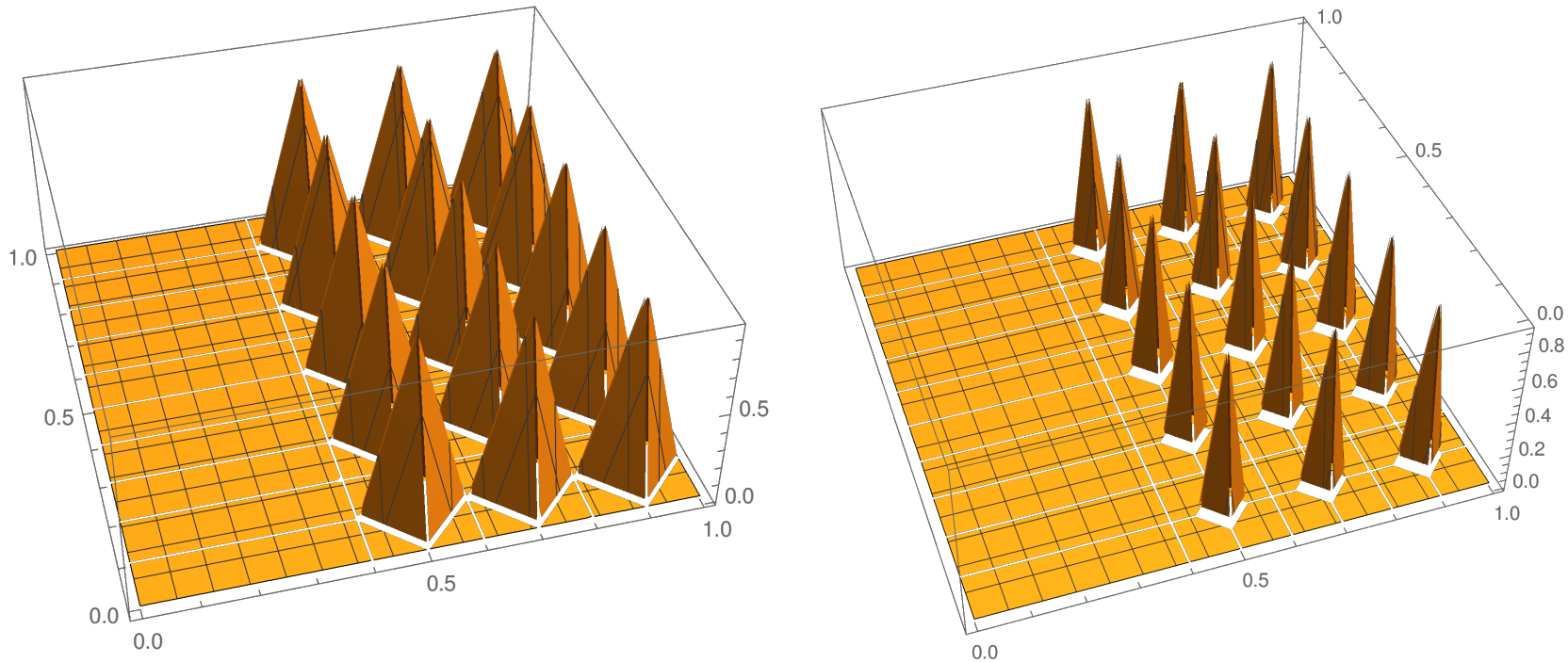
$$I_2(x, y) := \sum_{x_i < 0.5} \phi_i(x) + \sum_{j=1}^{N} \phi_j(y) - b \tag{6}$$

Let $b \in [1, 2)$, then the final output is:

$$f_h(x, y) := \frac{a(I_1(x, y))}{2 - b} + (-1)\frac{a(I_2(x, y))}{2 - b}.$$

The measure of compact support for $f_h(x, y)$ is decreasing to 0 as $b \to 2$.

Graphs for 2-D Examples-Special Case



(a) (b)

Figure: Here $x_1 = y_1 = 0.1$, $h = 0.1$, $N = 6$, (a) Graph of $a(I_1(x, y))/(2 - b)$ with $b = 1$; (b) Graph of $a(I_1(x, y))/(2 - b)$ with $b = 1.5$.

Let $\mathbf{x} \in \mathbb{R}^D, D \geq 3$,

$$\mathbf{x} = (x_1, x_2, \cdots, x_D),$$

the region is a D dimensional hyper-cube $[0,1]^D$. And $\{x_{1,i_1}, i_1 = 1, 2, \cdots, N\}$ is denoted as the scalar value set which is uniform in first dimension. Similarly, we have $\{x_{j,i_j}\}, j = 2, \cdots, D, i_j = 1, 2, \cdots, N$.
The training set data is

$$\{(x_{1,i_1}, x_{2,i_2}, \cdots, x_{D,i_D})\}$$

and $x_{1,1} = x_{2,1} = \cdots = x_{D,1} = 0$; $x_{1,N} = x_{2,N} = \cdots = x_{D,N} = 1$. Denote $\phi_{j,i_j}(x_j) = \phi(x_j - x_{j,i_j})$. Then we can define the D dimensional "basis function" as:

$$\Phi(\mathbf{x}) = \sum_{j=1}^{D} \left(\sum_{\{x_{j,i_j}\}} \phi_{j,i_j}(x_j) \right),$$

where sets $\{x_{j,i_j}\}$ are chosen as needed.

To approximate a continuous function $f(\mathbf{x})$, $\mathbf{x} \in [0,1]^D$ by three layer Relu neural network, we use the training set

$$\{((x_{1,i_1}, x_{2,i_2}, \cdots, x_{D,i_D}), f(x_{1,i_1}, x_{2,i_2}, \cdots, x_{D,i_D}))\}$$

$x_{j,i_j} \in [0,1]$ are uniform, $x_{j,i_j+1} - x_{j,i_j} = h > 0$. We define the "basis function" as:

$$\Phi_{i_1,i_2,\cdots,i_D}(\mathbf{x}) = \sum_{j=1}^{D} \phi_{j,i_j}, \tag{7}$$

where $a(x)$ is the Relu function, $\Phi_{i_1,i_2,\cdots,i_D}$ has the height D. The inputs for first hidden layer are similar, we need $D(N+2)$ neurons, which is less. Then we need N^D neurons in second hidden layer, and the final output is:

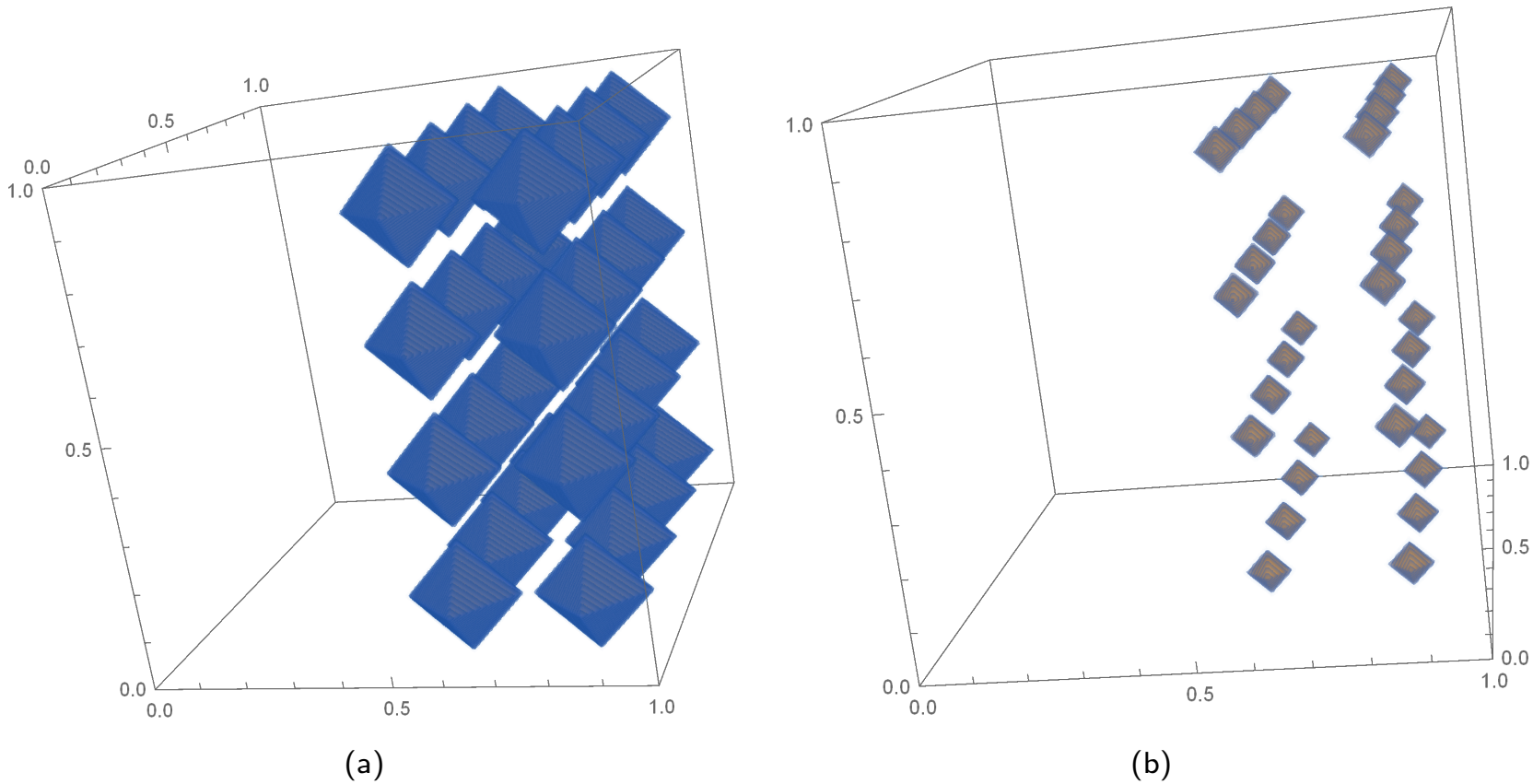
$$f_h(\mathbf{x}) := \sum_{i_1,i_2,\cdots,i_d=1}^{N} f_{i_1,i_2,\cdots,i_D} \frac{a(\Phi_{i_1,i_2,\cdots,i_D}(\mathbf{x}) - b)}{D - b},$$

where

$$f_{i_1,i_2,\cdots,i_D} = f(x_{1,i_1}, x_{2,i_2}, \cdots, x_{D,i_D}),$$

and $b \in [D-1, D)$. So as $b \to D$, the approximation will fail though it is the global optimal solution. Deeper networks will fail same as one dimensional case.

Special Case for 3-D Function Approximation



(a) (b)

Figure: Density color plots for 3-D functions, blue color means close to zero. The center of each diamond has value 1, the surface of it has value 0. Here $N = 4$, $x_{j,1} = 1/8, j = 1, 2, 3$, $h = 1/8$, (a) Graph of $a(I_1(\mathbf{x}))/(3 - b)$ with $b = 2.1$; (b) Graph of $a(I_1(\mathbf{x}))/(3 - b)$ with $b = 2.7$.

Thanks & Questions ?