Bottlenecked Function Prediction Using NNs

Yuval Alfassi

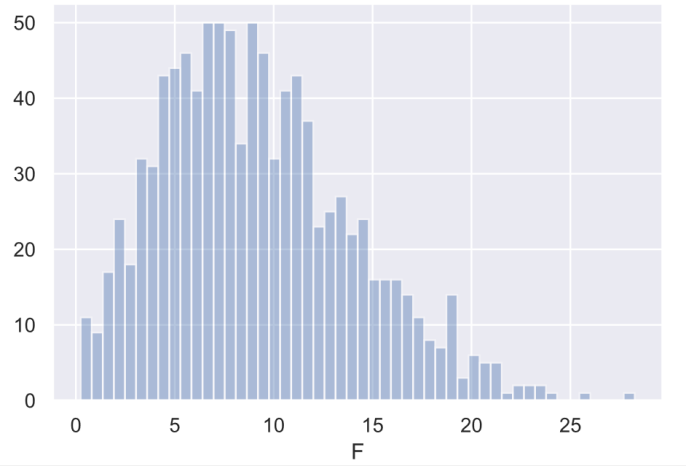
I. Objective

Given the multidimensional function $F(x_1, x_2, x_3, x_4) = (x_1 + x_2)^2 + (x_2 + x_4)^2$, are there functions g_1, g_2, h such that $h(g_1(x_1, x_3), g_2(x_2, x_4)) = F(x_1, x_2, x_3, x_4)$ (where h is a second degree polynomal and g_1,g_2 are linear)?

Let's try that with Neural Networks!

II. Test Data

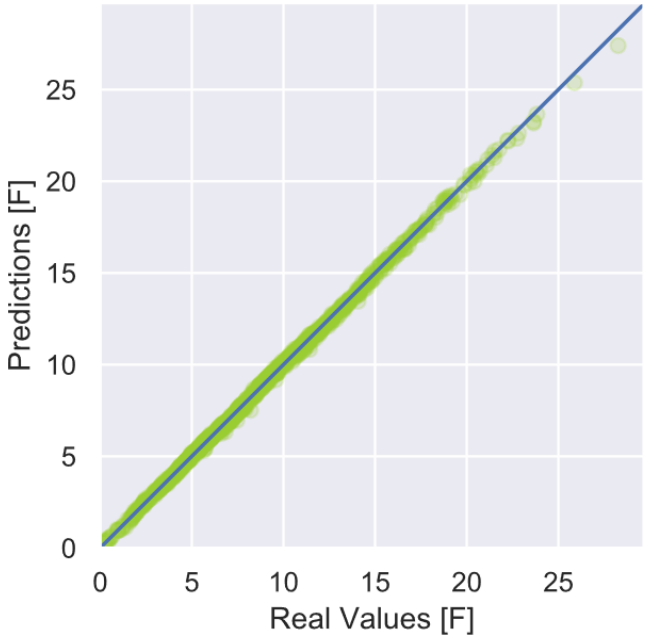
$x_1, ..., x_4$ were sampled out of $[0, 2]$. 1000 point samples were generated randomly (uniform distribution). The distribution of the points in regard to F is as follows:



The NNs were trained using only these points as a train set. No test set or validation set are required.

III. Fully Connected NN

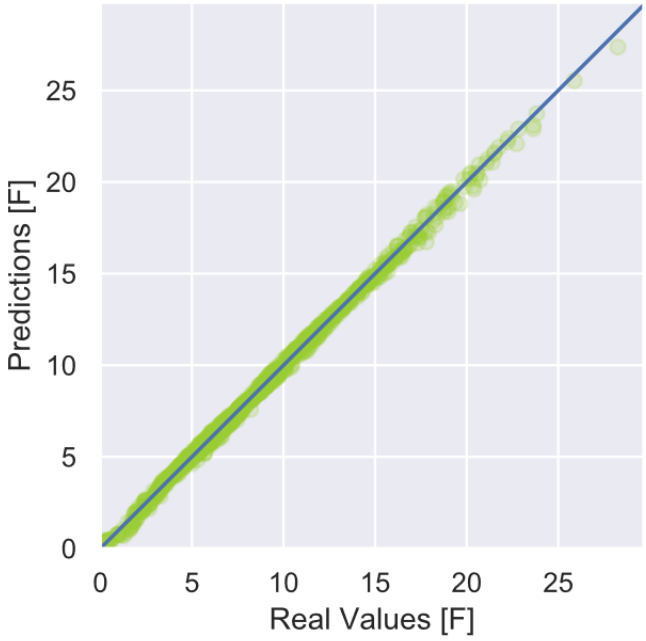
A fully connected NN with a dense architecture of $4 \times 10 \times 10 \times 1$ converges very fast. It's predictions are:



The MSE of the predictions is **0.017**.

IV. $\{x_1, x_2\}, \{x_3, x_4\}$ Function Partition

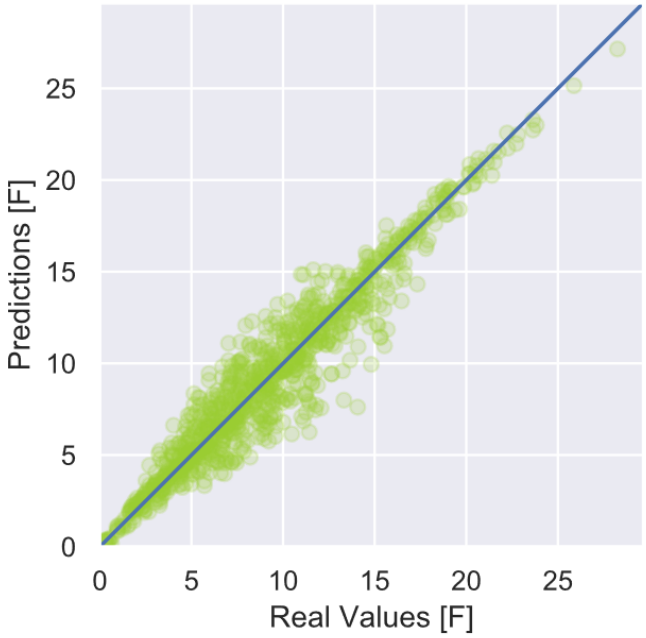
Now, lets divide the input points of F into: $h(g_1(x_1, x_2), g_2(x_3, x_4)) = F(x_1, x_2, x_3, x_4)$. For that, the NN has to be 'halved', so two NN architechtures with 2 input neurons merge into a big NN. The NNs are of size 2×1 which then merge into a dense NN of $2 \times 6 \times 6 \times 1$. The predictions are:



The MSE of the predictions is **0.021**.

V. $\{x_1, x_3\}, \{x_2, x_4\}$ Function Partition

The more interesting way to divide the input is $\{x_1, x_3\}, \{x_2, x_4\}$, i.e. see whether h, g_1, g_2 exist s.t $h(g_1(x_1, x_3), g_2(x_2, x_4)) = F(x_1, x_2, x_3, x_4)$. The same NN architecture as before was used. The predictions are:



The MSE of the predictions is **1.81**.

Unfortanately, deeper NNs do not converge, and wider NNs do not yield better results