Function Approximation with Fully-Connected Neural Network

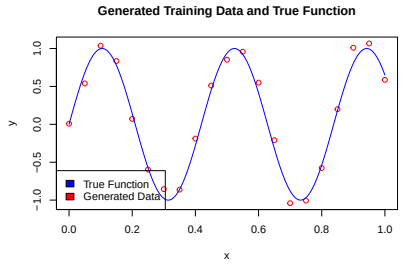
Tony Kwok

2023-01-25

In this project, we explore the use of neural networks on a regression task. In particular, we train a fully-connected feedforward neural network with 1 hidden layer to learn a sinusoidal function given by f(x) = sin(15x). Throughout the project, we will be using the R package, "neuralnet", to construct our models.

Generating training data and Risk Estimation

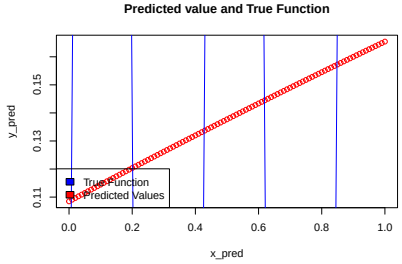
We first generate a vector, x, of 21 data points uniformly between 0 and 1, and construct another vector y, with the function y = f_hat(x) = sin(15x) + N(0, 0.01), i.e. our objective function plus some random Gaussian noise of mean = 0 and variance = 0.01. We will be using these (x, y) pairs as our training data. Next, we plot the generated training data and the true function in the following figure. As expected, most of the generated data lie very close to the true sine function.



We then calculate the empirical risk of the function f_hat on the grid of x-values defined above, which serves as an estimation of the true risk of the function. The value of the empirical risk is about 0.007. Note that this value is essentially an estimate of our optimal risk and we will be using it as a benchmark to evaluate our model fits in later sections.

Fitting a Neural Network and evaluating our predictions

We fit a FC neural network with 1 hidden layer with 3 units to the (x, y) pairs we generated, with a 'tanh' activation function. To make evaluations, we create a vector of 100 points uniformly between 0 and 1 as our test data and feed them to our trained model to make predictions. We then plot our predictions along with the real function in the following figure and we see that we obtained a very poor fit, which is likely due to underfitting.

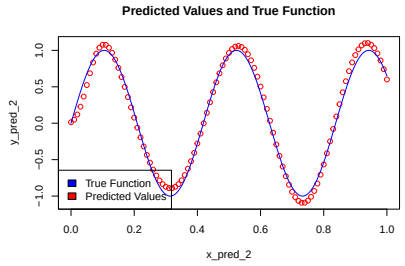


Calculating Empirical Risk of random data points

We generate 10 data points from Unif([0,1]) and pass them to our fitted model to make predictions and compute the empirical risk. The value obtained is about 0.1266, which is much higher than the optimal risk, further indicating a bad fit in our model.

Adding an extra neuron

One of the possible solutions to avoid falling into the underfitting regime is to increase the number of parameters. Here we repeat the above steps but with an extra neuron in the hidden layer (so now we have 1 hidden layer with 4 neurons) as an attempt to improve our model performance.

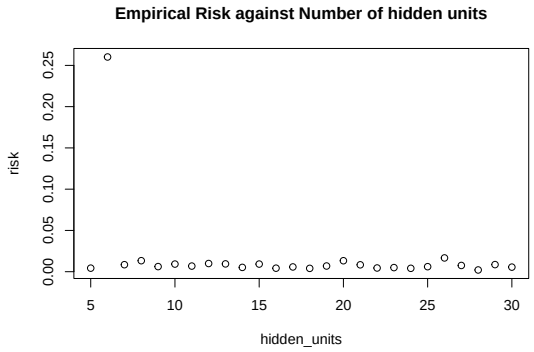


As depicted in the new figure, the new fit is much better than the one before, as most of the predictions are very close to the real values. The neural network with width = 4 successfully learned the sinusoidal function.

Again, we calculate the empirical risk with the new fit. The calculation in R tells us that the new empirical risk is around 0.006, which is a lot smaller than the fit with only 3 hidden units, and is very close to the optimal risk we estimated at the beginning, indicating an excellent fit.

More hidden units, overparametrization and overfitting

Finally, we repeat the above steps again with more neurons in the hidden layer to observe the phenomenon of overparametrization and investigate potential overfitting regime. Overparametrization happens when the number of parameters is larger than the number of training data, which is 21 in our case. There are extensive research on how overparametrization might be beneficial in machine learning in certain ways, but in general it makes the model prone to overfitting, as the model now has excessive parameters to 'memorize' training data in order to maximize training accuracy. Here we plot the estimated risks against the width of the hidden layer, ranging from 5 to 30.



The figure above shows no significant evidence of overfitting in our model despite overparametrization when number of hidden units = h > 21, as the empirical risk remains reasonably low even for large h (despite possible fluctuation due to the randomness). We believe that it is because the noise we added to the training data follows Normal(0, 0.01), which is very mild. In this case, even if the models with excessive parameters fit the noise, they are still able to generalize fairly well.