simons_doc

Simon Jorstedt

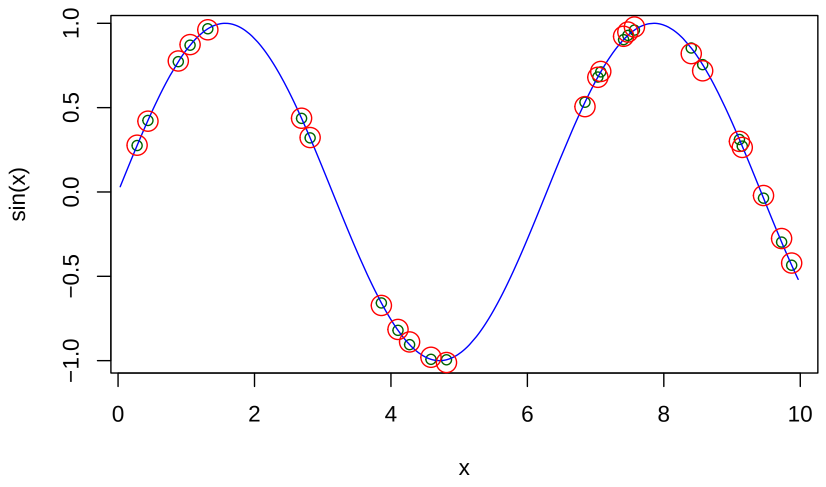
2023-12-14

Assignment 3

Assignment 3.1

In Figure 3.1 below, we see the result of a neural network applied to a set of 25 datapoints uniformly spread over the interval $[0, 10]$, with the response being the sine function applied to the datapoints, with no added random error. These training points are colored green and made small. Red points are the pointwise predictions by the neural net. The blue curve represents the true sine curve.

Fig 3.1: Simple neural network fitting the sine function

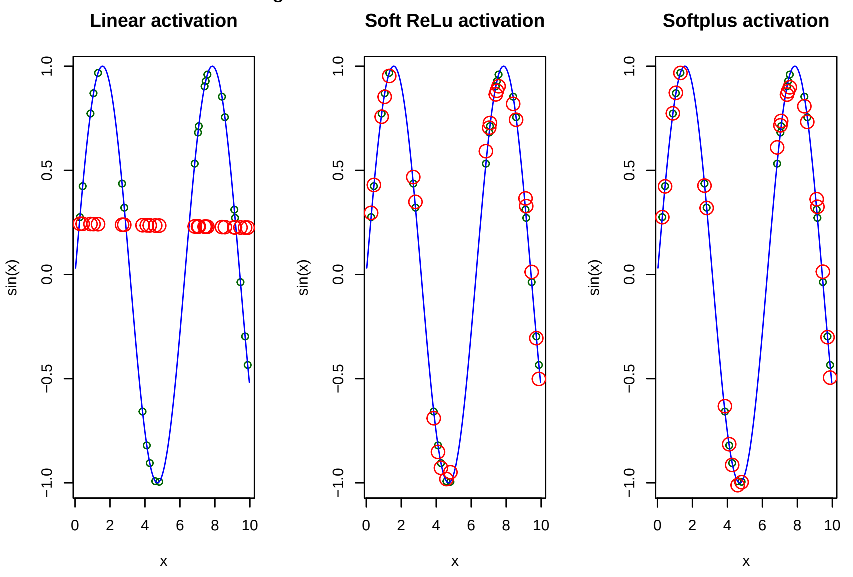


Assignment 3.2

The neural network in Assignment 3.1 used a standard sigmoid activation function. Now we are going to use a linear activation function (or the identity) $h_1(x) = x$, the ReLu $h_2(x) = \max(0, x)$, and the softplus $h_3(x) = \ln(1 + e^x)$ each in their own neural network. However, as we tried this, we ran into an issue with the ReLu. It does not appear to be preprogrammed in the `neuralnet` function, and neither does `neuralnet` seem to be able to handle the undifferentiable point in ReLu. Instead we will be using a function we call the "Soft ReLu" $h_4(x) = \ln(1 + e^x) \cdot \frac{e^x}{1+e^x}$, which is constructed by multiplying the softplus function with its derivative. That derivative happens also to be the sigmoid function. The result is a function that resembles the softplus, but takes a sharper turn around $x = 0$, and thus approximates the ReLu better than the softplus.

We train these three neural networks using the data from Assignment 3.1, and plot the resulting predictions in Figure 3.2. Again, green points are training observations, red points are predictions, and the blue curve is the true sine curve. The neural network with a linear "activation function" clearly performs terribly. This is because the linear "activation function", or rather the *lack* of a non-linear activation function, reduces the neural network to a simple linear regression model. This explains why all the predictions by this neural network follow a straight line. For the neural networks with Soft ReLu and Softplus activation functions respectively, the predictions appear to be quite good, but possibly slightly worse than for the neural network with a sigmoid activation function.

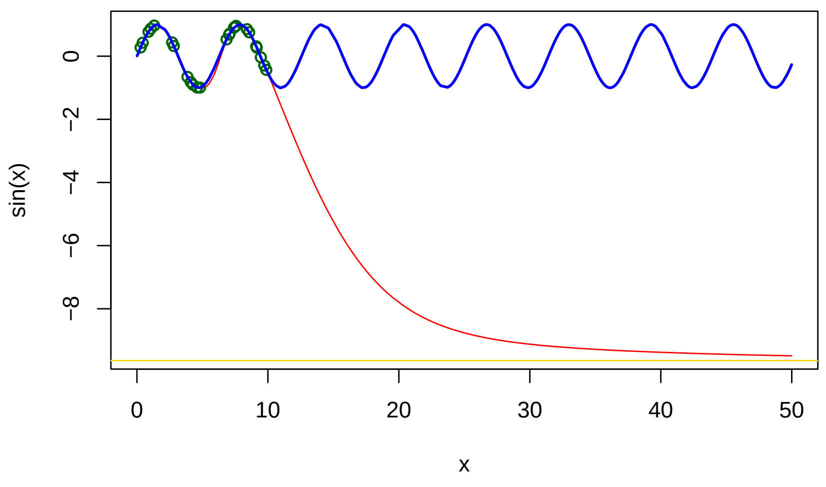
Figure 3.2: Three activation functions



Assignment 3.3

Now we generate data similarly to before in the range $[0, 50]$ and attempt to predict the value of the sine function applied to these values using the neural network that was trained in Assignment 3.1. Essentially we will use the neural network to predict the response of data from far outside the range of the training data. As a reminder, the range of this training data is $[0, 10]$. The result of this is plotted in Figure 3.3. Again the training data of the model is represented by green circles, and the predictions are represented by the red curve. The blue curve represents the entire new dataset (essentially the sine function graph). Clearly, the n eural network has a fatal flaw: It breaks away, producing terrible predictions immediately outside the range of the training data.

Fig 3.3: Neural network predictions on new data from outside the range of training data



Assignment 3.4

In Assignment 3.3 we saw that the prediction curve of the Neural Network appeared to converge to some value. To find this limit, we formulate the function $f(x)$ of the Neural Network, first in compact notation, and then as a linear combination. Here, $\boldsymbol{W}^{(1)}$ is a 10 by 1 matrix carrying the weights of the first layer, $\boldsymbol{W}^{(2)}$ is a 1 by 10 matrix carrying the weights of the second (output) layer, $\boldsymbol{b}^{(1)}$ is a vector of length 10 carrying the biases for the first layer, and $b^{(2)}$ is a vector of length 1 (so a scalar in this case) carrying the second (output) layer bias term. The sigmoid activation function $\sigma(\cdot)$ is carried out element wise.

$$f(x) = \boldsymbol{W}^{(2)}\sigma\Big(\boldsymbol{W}^{(1)}x + \boldsymbol{b}^{(1)}\Big) + b^{(2)} = \boldsymbol{W}_1^{(2)}\sigma\Big(\boldsymbol{W}_1^{(1)}x + \boldsymbol{b}_1^{(1)}\Big) + \cdots + \boldsymbol{W}_{10}^{(2)}\sigma\Big(\boldsymbol{W}_{10}^{(1)}x + \boldsymbol{b}_{10}^{(1)}\Big) + b^{(2)}$$

When studying this function, we find that if $\boldsymbol{W}_i^{(2)}$ is positive, then $\sigma\Big(\boldsymbol{W}_i^{(1)}x + \boldsymbol{b}_i^{(1)}\Big)$ will approach 1 as x becomes larger. If $\boldsymbol{W}_i^{(2)}$ is negative, then $\sigma\Big(\boldsymbol{W}_i^{(1)}x + \boldsymbol{b}_i^{(1)}\Big)$ will approach 0 as x becomes larger. If $\boldsymbol{W}_i^{(1)} = 0$, then the activation of the ith hidden unit in the first layer reduces to a constant. This has the effect that as $x \to \infty$, $f(x)$ reduces to the following, where we first introduce an indicator variable δ_i.

$$\delta_i = \left\{\begin{matrix} 0 & \text{if} & \boldsymbol{W}_i^{(2)} < 0 \\ \sigma(\boldsymbol{b}_i^{(1)}) & \text{if} & \boldsymbol{W}_i^{(2)} = 0 \\ 1 & \text{if} & \boldsymbol{W}_i^{(2)} > 0 \end{matrix}\right\}$$

$$\lim_{x\to\infty} f(x) = \boldsymbol{W}_1^{(2)} \cdot \delta_1 + \cdots + \boldsymbol{W}_{10}^{(2)} \cdot \delta_{10} + b^{(2)}$$

We calculate this limit for the trained Neural Network from Assignment 3.1 to be about -9.64, and include the limit as a gold line in Figure 3.3.

Assignment 3.5

Now we are interested in training a Neural Network to learn the arcsin function, which is the inverse of the sine function. To do this, we first generate 500 observations in the interval $[0, 10]$, and then apply the sine function to each observation. We then train a Neural Network with the same structure as previously, but with the observations as response variable, and the sine function applied to the observations as the explanatory variable. In Figure 3.4 we plot the training observations with a green line, and the predictions (on the training data) as a red line. Note that the *predicted* $\hat{x}$ are indicated on the horizontal axis, and *explanatory* $\sin(x)$ are indicated on the vertical axis, in order to facilitate easy comparison with previous figures.

In Figure 3.4 we see clearly that the Neural Network is terrible at predicting the observations. This is not surprising, since the sine function is surjective, but not injective, and thus can not be uniquely inverted. Additionally, the range of the explanatory variable ($[-1, 1]$) is quite narrow, which makes it harder for the NN to capture the behaviour in the response variable.

Fig 3.4a: Neural Network trained to fit the arcsine function.

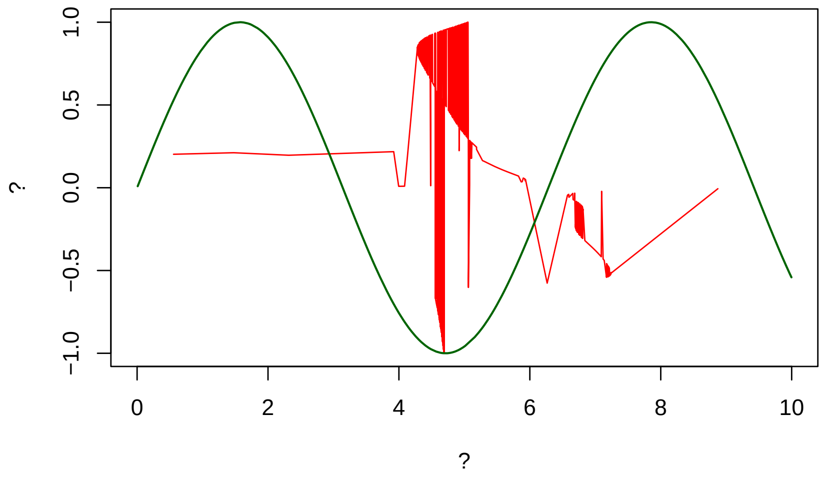
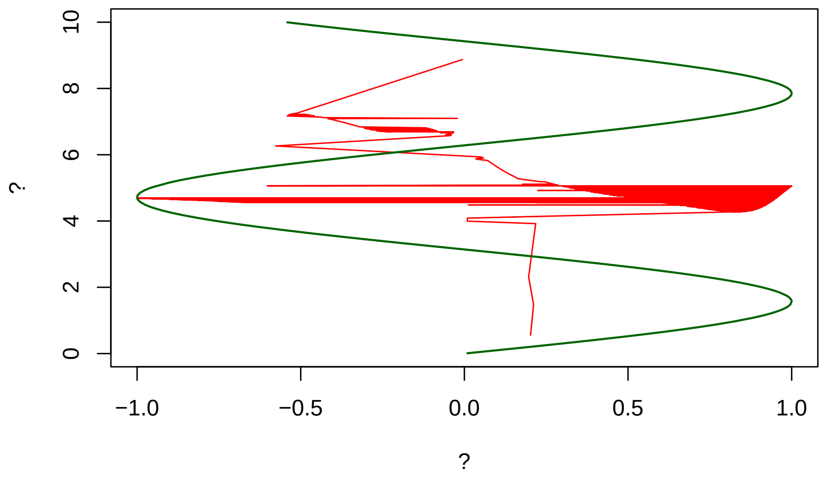


Fig 3.4b:



?
10
8
6
4
2
0
−1.0
−0.5
0.0
0.5
1.0
?