

Training neural networks with regularization

Anton Antonov
MathematicaVsR at GitHub
May 2018

Introduction

This notebook is part of the MathematicaVsR at GitHub project “DeepLearningExamples”.

This notebook has code that corresponds to code in the book “Deep learning with R” by F. Chollet and J. J. Allaire. See the GitHub repository: <https://github.com/jjallaire/deep-learning-with-r-notebooks> ; specifically the notebook “Overfitting and underfitting”.

In many ways that R notebook has content similar to WL’s “Training Neural Networks with Regularization”.

The R notebook “Overfitting and underfitting” discusses the following possible remedies of overfitting: smaller network, weight regularization, and adding of a dropout layer.

The WL “Training Neural Networks with Regularization” notebook discusses: early stopping of network training, weight decay, and adding of a dropout layer.

The goal of this notebook is to compare the R-Keras and WL-MXNet neural network frameworks in a more obvious way with simple data and networks.

Get code

Here we load a package with utilities:

```
In[136]:= Import["https://raw.githubusercontent.com/antononcube/MathematicaForPrediction/master/MathematicaForPredictionUtilities.m"]
```

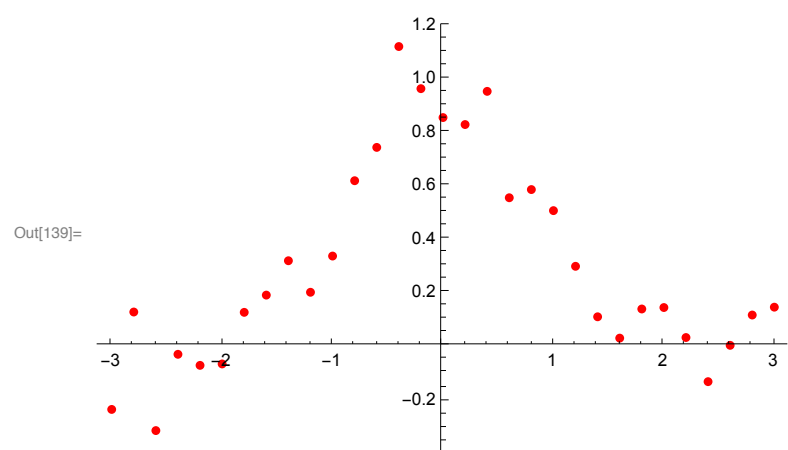
Get data

As seen in the tutorial page “Training Neural Networks with Regularization”.

```
In[137]:= data = Table[x -> Exp[-x^2] + RandomVariate[NormalDistribution[0, .15]], {x, -3, 3, .2}];  
Length[data]
```

```
Out[138]:= 31
```

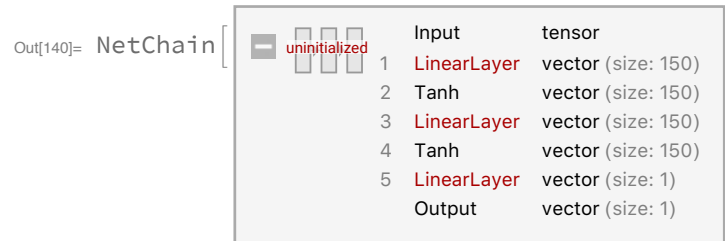
```
In[139]:= plot = ListPlot[List@@@data, PlotStyle -> Red]
```



Overfitting

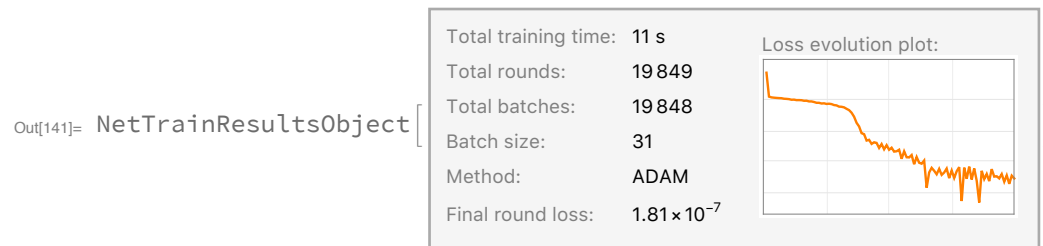
Create a multilayer perceptron with a large number of hidden units:

```
In[140]:= net = NetChain[{150, Tanh, 150, Tanh, 1}]
```



Train the net for 10 seconds:

```
In[141]:= results1 = NetTrain[net, data, All, TimeGoal -> 10]
```



Despite the noise in the data, the final loss is very low:

```
In[142]:= results1["FinalRoundLoss"]
```

Out[142]= 1.81048×10^{-7}

The resulting net overfits the data, learning the noise in addition to the underlying function. To see this, we plot the function learned by the net alongside the original data.

Obtain the net from the NetTrainResultsObject:

```
In[143]:= overfitNet = results1["TrainedNet"]
```

Out[143]= NetChain

Input port:

Output port:

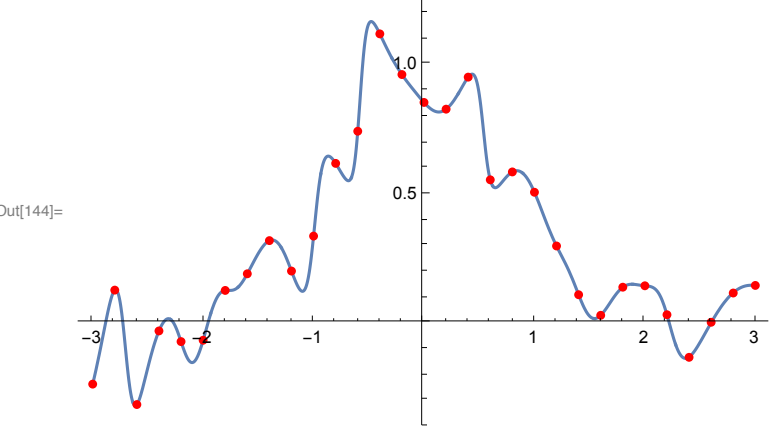
Number of layers:

real

scalar

5

```
In[144]:= Show[Plot[overfitNet[x], {x, -3, 3}], plot]
```



Smaller network

Make smaller net and train for same time.

```
In[145]:= net2 = NetChain[{3, Tanh, 3, Tanh, 1}]
```

Out[145]= NetChain

Input port:

Output port:

Number of layers:

tensor

vector (size: 1)

5

Train the net for 10 seconds:

```
In[146]:= results2 = NetTrain[net2, data, All, TimeGoal -> 10]
```

Out[146]= NetTrainResultsObject

Total training time: 11 s

Total rounds: 21617

Total batches: 21616

Batch size: 31

Method: SGD

Final round loss: 0.00962

Loss evolution plot:

```
In[147]:= results2["FinalRoundLoss"]
```

```
Out[147]= 0.00962029
```

```
In[148]:= tNet2 = results2["TrainedNet"]
```

Out[148]= NetChain

Input port:

Output port:

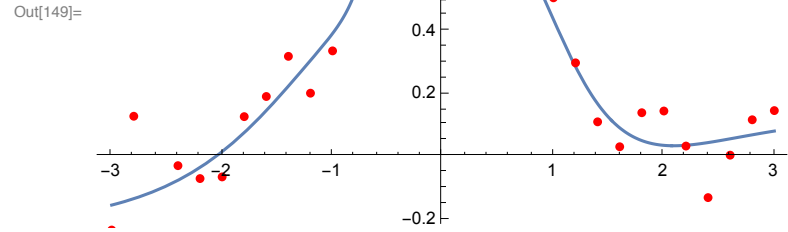
Number of layers:

real

scalar

5

```
In[149]:= Show[Plot[tNet2[x], {x, -3, 3}], plot]
```



Weight decay

```
In[150]:= results3 = NetTrain[net, data, All, Method -> {"SGD", "L2Regularization" -> 0.01}]
```

```
Out[150]= NetTrainResultsObject[
```

| | |
|----------------------|--------|
| Total training time: | 4.5 s |
| Total rounds: | 10 000 |
| Total batches: | 10 000 |
| Batch size: | 31 |
| Method: | SGD |
| Final round loss: | 0.0138 |

Loss evolution plot:

```
In[151]:= results3["FinalRoundLoss"]
```

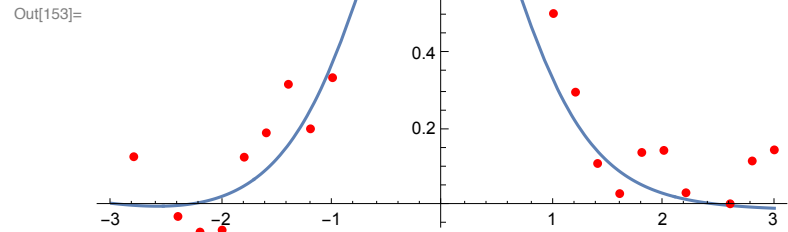
```
Out[151]= 0.0137857
```

```
In[152]:= tNet4 = results3["TrainedNet"]
```

```
Out[152]= NetChain[
```

| | | |
|---|-------------------|--------|
| + | Input port: | real |
| → | Output port: | scalar |
| → | Number of layers: | 5 |

```
In[153]:= Show[Plot[tNet3[x], {x, -3, 3}], plot]
```



Dropout

Add dropout layer.

```
In[160]:= net4 = NetChain[{150, DropoutLayer[0.2], Tanh, 150, Tanh, 1}]
```

Out[160]= NetChain

uninitialized

Input port:

Output port:

Number of layers:

tensor

vector (size: 1)

6

Train the net for 10 seconds:

```
In[161]:= results3 = NetTrain[net4, data, All, TimeGoal -> 10]
```

Out[161]= NetTrainResultsObject

Total training time: 11 s

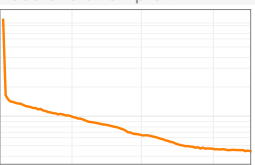
Total rounds: 17 594

Total batches: 17 593

Batch size: 31

Method: ADAM

Final round loss: 0.00416

Loss evolution plot:

```
In[162]:= results4["FinalRoundLoss"]
```

```
Out[162]= 0.0174739
```

```
In[163]:= tNet4 = results4["TrainedNet"]
```

Out[163]= NetChain

Input port:

Output port:

Number of layers:

real

scalar

5

```
In[164]:= Show[Plot[tNet4[x], {x, -3, 3}], plot]
```

