



ADRIAN BEVAN

GRADNET MACHINE LEARNING AND AI WORKSHOP: INTRODUCTION TO PYTHON AND TENSORFLOW

NEURAL NETWORKS (NNs): TUTORIAL

WORKPLAN

- ▶ 0. Setup.
- ▶ 1. Model building and execution with TensorFlow
- ▶ 2. Linear regression:
 - ▶ Explore hyper-parameter (HP) performance and optimisation convergence for a gradient descent algorithm with a simple model.
- ▶ 3. Function approximation:
 - ▶ Explore HP training and model reliability for a regression problem.
- ▶ 4. Closing remarks

0. SETUP – ANACONDA (USE PYTHON SCRIPTS RATHER THAN NOTEBOOKS)

- ▶ Install anaconda on your system.
- ▶ Install tensorflow and matplotlib
- ▶ clone github repo and cd (or navigate) into the NN script directory:

```
git clone https://github.com/adrianbevan/ATLAS-UK-ML.git
```

```
cd ATLAS-UK-ML/NN/
```

- ▶ Scripts can be run from the command line, or through Spyder. The jupyter notebooks can be run through anaconda.

1. MODEL BUILDING AND EXECUTION WITH TENSORFLOW Background

- ▶ Machine Learning with TensorFlow works in the same way you would describe a calculation on the white board.
 - ▶ 1. You define the calculation you want to do (the graph).
 - ▶ 2. Then you evaluate that graph some number of times; each evaluation corresponds to an iteration in the optimisation step.
- ▶ Conceptually this differs from how one would traditionally approach a coding problem like this, and the different approach can take some time to adapt to.

2. LINEAR REGRESSION

Background


LinearRegression.ipynb

36.7 kB

a minute ago

- ▶ We will learn the values of m and c for

$$y = mx + c$$

- ▶ Initial parameter guesses are in the range $[0, 1]$
- ▶ We set the learning rate to be some reasonable number
- ▶ We set the number of epochs to some value (initially 100) to explore if this is enough for the model to converge.
- ▶ Cells in the notebook can be evaluated using `[shift]+[enter]` or by pressing the play button .

2. LINEAR REGRESSION

Background

 LinearRegression.ipynb

36.7 kB

a minute ago

- ▶ Suggested things to explore:
 - ▶ Increase the `learning_rate` from 0.0005 to observe how the model agreement and optimisation convergence changes.
 - ▶ At what point does the optimisation stop working?
 - ▶ You may want to change the number of `training_epochs` to facilitate getting the model to converge.
 - ▶ You may want to change the true values of `gradient` and `intercept` to explore how this affects the optimisation (this is equivalent to changing how close the initial guess is for the optimisation to the true parameters).

2. LINEAR REGRESSION

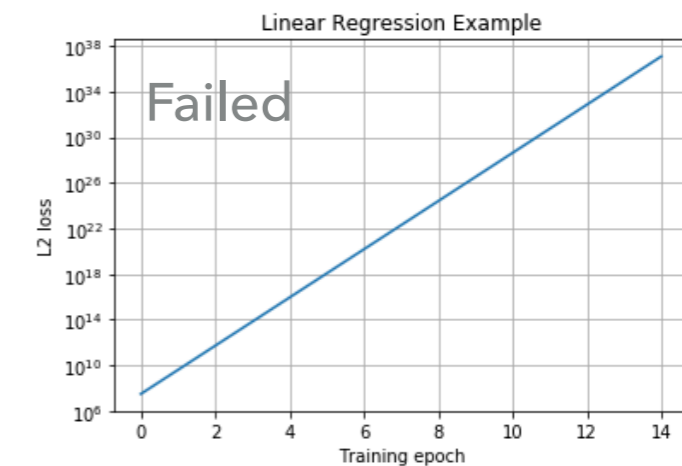
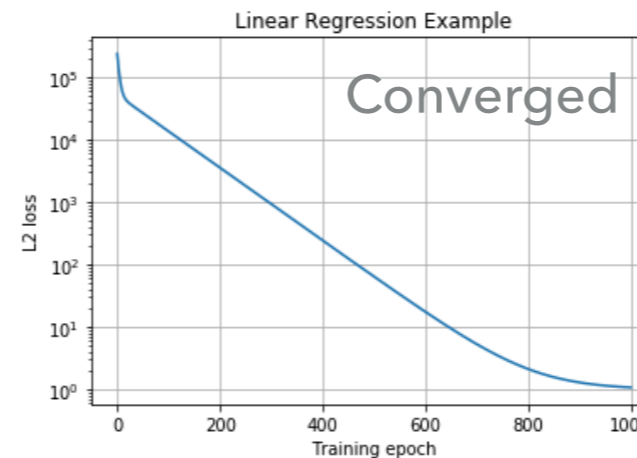
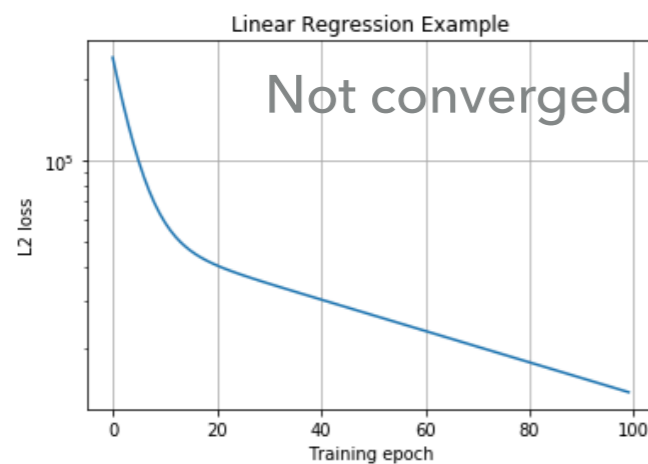
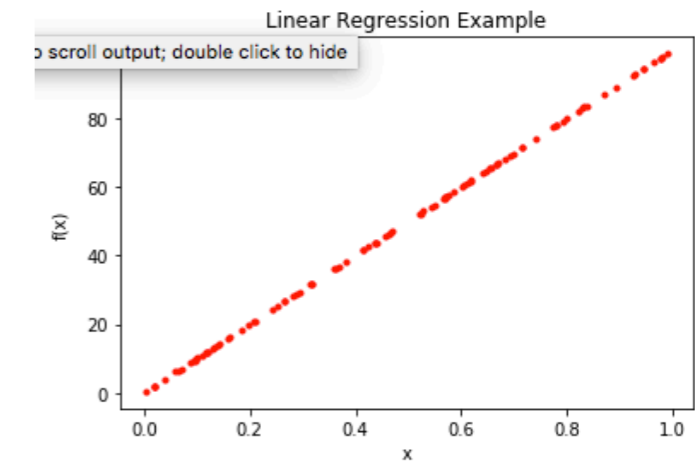
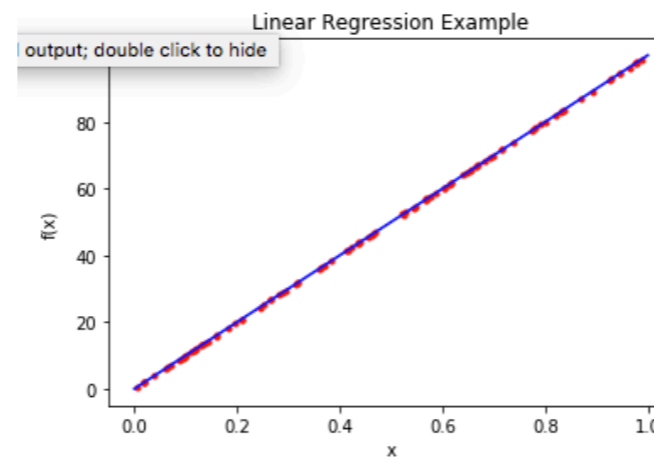
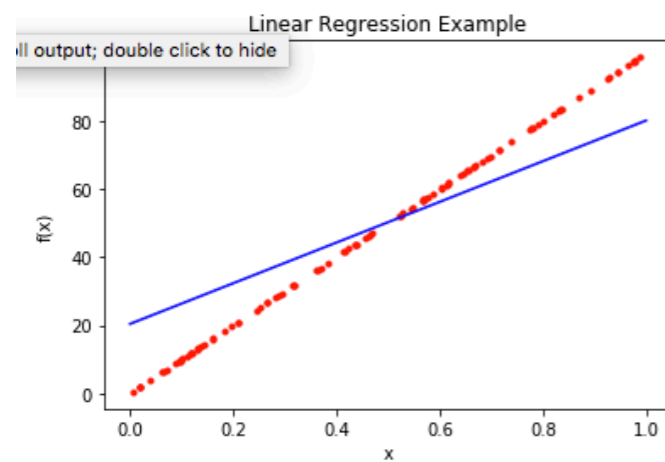
Review Results

LinearRegression.ipynb

36.7 kB

a minute ago

- ▶ Having done this please pause and reflect.
- ▶ This is a simple model with 1 input feature, 1 output and 2 HPs. It takes some effort to ensure that the model converges to a sensible result.



3. FUNCTION APPROXIMATION

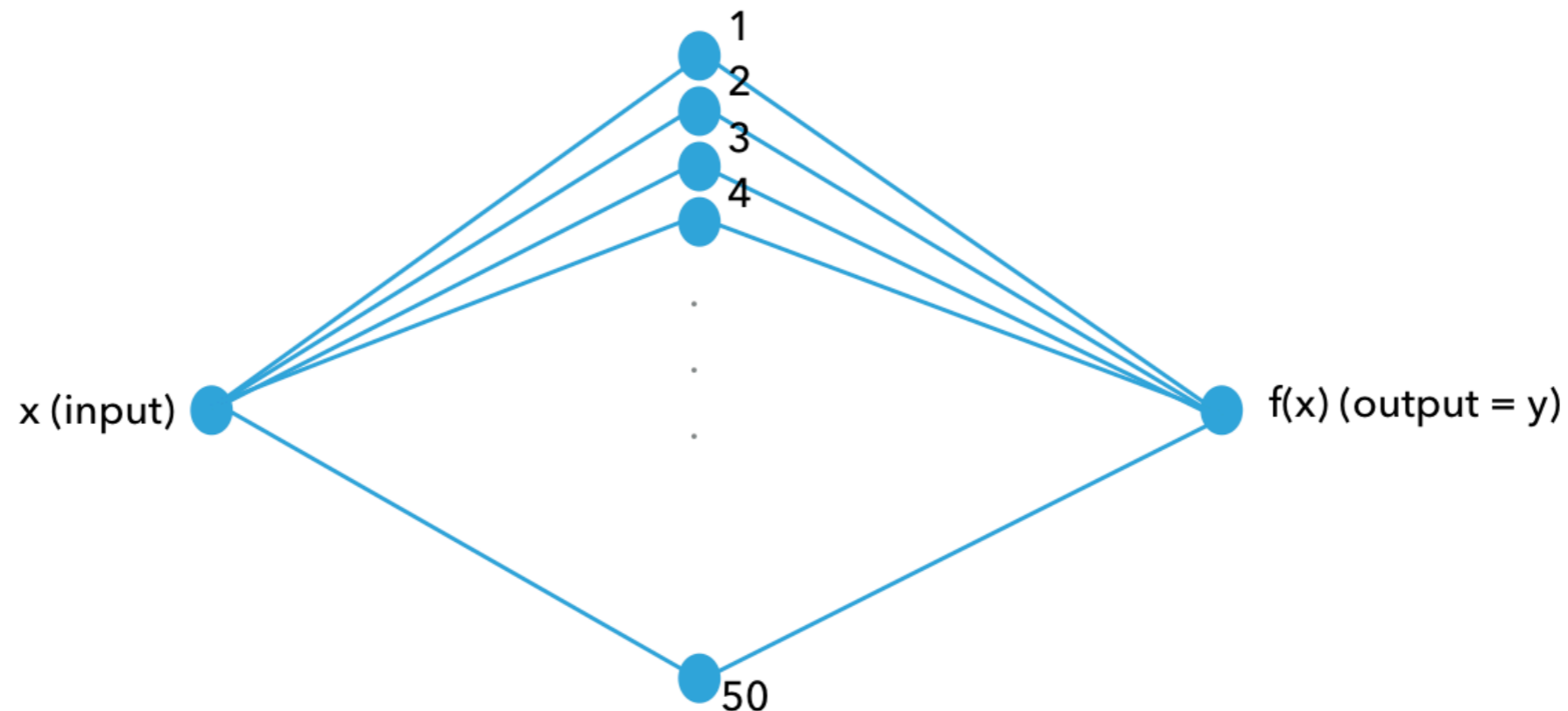
Background

FunctionApproximation.ipynb

57.3 kB

a minute ago

- ▶ We now consider a more complicated model - this will be a neural network with the configuration 1 input feature, 50 nodes in a single layer and one output feature.



- ▶ For this example we are going to model $y = x^2$.

3. FUNCTION APPROXIMATION

[Background](#) FunctionApproximation.ipynb

57.3 kB

a minute ago

- ▶ Each node in this model is an activation function that has two parameters, takes one input and gives one output (just like the linear regression example).

$$y_i = f(w_i x + \beta_i)$$

- ▶ The chosen activation function for this example is the relu activation function.
- ▶ The output node is just the matrix operation $w^T x + \beta$.

3. FUNCTION APPROXIMATION

[Background](#) FunctionApproximation.ipynb

57.3 kB

a minute ago

- ▶ After running the example out of the box try the following:
 - ▶ Change the `learning_rate` by a factor of 10 either way to observe how that affects the training performance.
 - ▶ Change the number of `training_epochs` with a reasonable learning rate to get a good predictive model.
 - ▶ Change the noise level from 0.1 to 1.0 to see how this affects the learning process.

3. FUNCTION APPROXIMATION

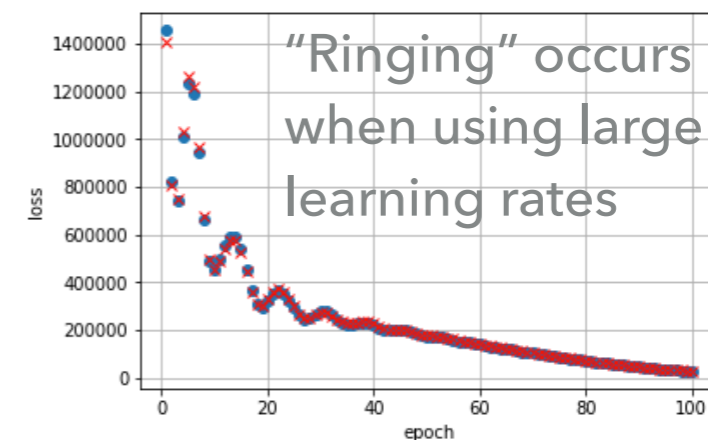
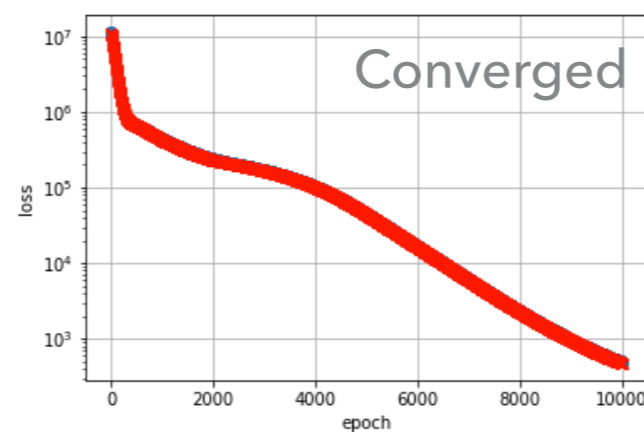
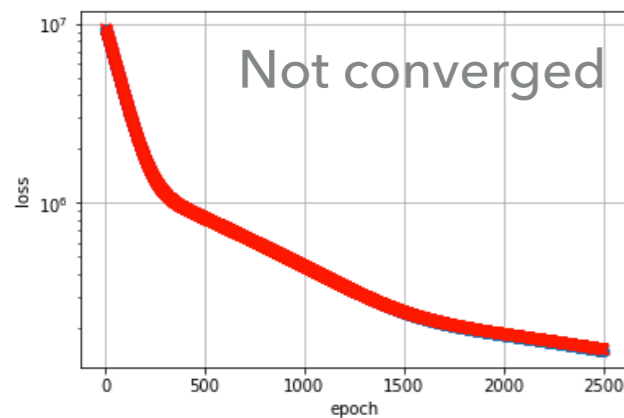
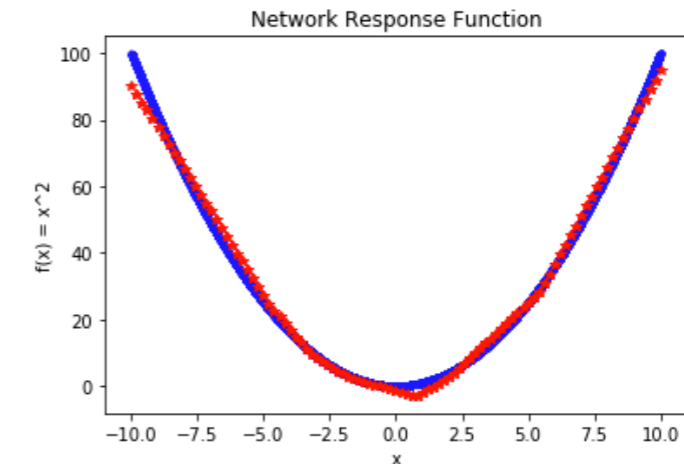
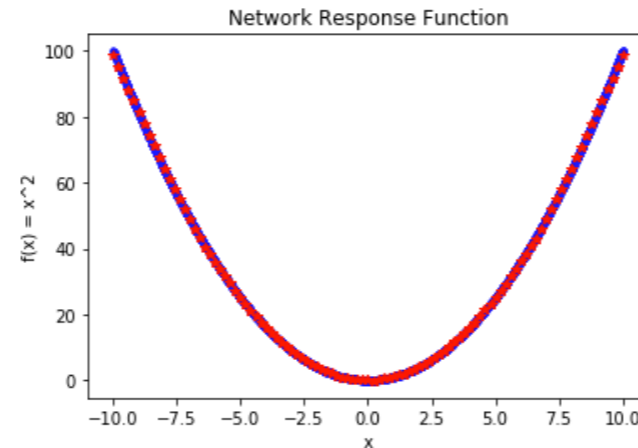
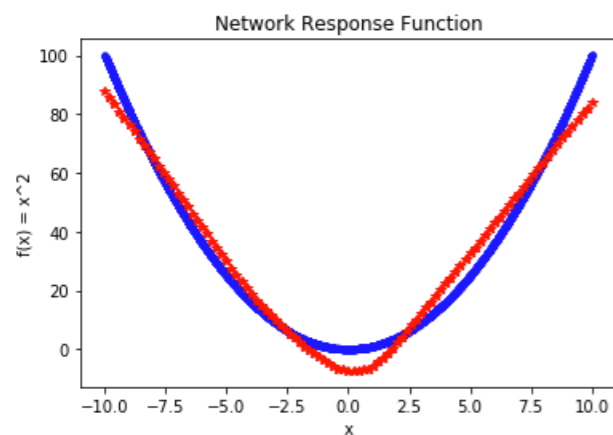
Review Results

FunctionApproximation.ipynb

57.3 kB


a minute ago

- ▶ As with the linear regression example, one needs to tune HPs related to training to get a good model.



- ▶ HEP problems have larger input feature spaces and would generally result in more layers for a NN. This means care needs to be taken with training to ensure a robust model is derived.

4. CLOSING REMARKS

 FunctionApproximation.ipynb

57.3 kB

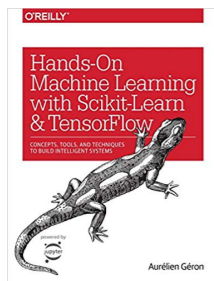
a minute ago

- ▶ Why didn't we discuss deep learning?
 - ▶ These tutorials are intended to give a short practical introduction, that's not compatible with a deep learning approach.
 - ▶ For deep learning you need lots of training examples, and compute resource in order to develop and train a model for a large number of epochs (doesn't fit into an hour).
- ▶ Interested in deep learning?
 - ▶ See the books by Goodfellow and Géron. You may wish to review some of my lectures/tutorials that discuss deep learning, including in the HEP context (see CINVESTAV 2018):
 - ▶ <https://pprc.qmul.ac.uk/~bevan/teaching.html>

REFERENCES & NOTES



- ▶ There are many useful tutorials on the tensorflow web page: <https://www.tensorflow.org> (check the API version number when browsing this site as the version changes frequently).



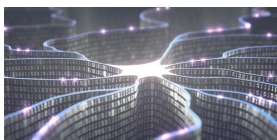
- ▶ The O'Reilly book by Géron, "Hands-On Machine Learning with Scikit-Learn & TensorFlow", is another good starting point.



- ▶ CERN has an interexperiment machine learning group: <https://iml.web.cern.ch>



- ▶ ATLAS has a Machine Learning Forum: <https://twiki.cern.ch/twiki/bin/viewauth/AtlasComputing/MachineLearningForum>



- ▶ I have more material on ML and TensorFlow available at:
 - ▶ <https://pprc.qmul.ac.uk/~bevan/teaching.html>
 - ▶ <https://www.qmul.ac.uk/summer-school/>