Deep Learning for Trivial Inverse Problems

Sifang Du

Supervisor : Dr Marta Betcke

Second Marker: Dr Simon Arridge

Project Aims:

1. Investigate the approximation of a pseudo-inverse by a neural network on a prototypical example of an ill-conditioned 2x2 matrix.
2. Understand when the neural network fails to approximate a pseudo-inverse and how correct behaviour can be restored.

Project Objectives:

1. Understand the approach taken in the paper “Deep Learning for Trivial Inverse Problems” by Prof Peter Maass.
2. Reproduce the results obtained in the paper.
3. Use alternative approaches for the same problem.
   e.g. training with regularised data, different network architectures, using orthogonal coefficients to force robustness

Expected deliverables:

1. A final report documenting the data, methods, results used in the project
2. The code of different deep models used implemented in the project

Progress made to date:

Until the completion of this report, I have completed several versions of the prototype networks which have the same structure as the one used in the paper by Prof Peter Maass. My first version of this neural network was coded using the PyTorch framework. This version was abandoned later because we found out that the learning for this particular need the weights to be under constraints, and PyTorch does not support this level of customisation which leads to my second version of code.

In the second version, I defined the network and the learning method using python only. My current network can train the randomly generated data successfully in a reasonable time, and the outputted weights still have the symmetry. However, this network was found out that although it trains successfully and returns the desired output, the backward propagation method is slightly different from the paper. The current network uses symmetry constraints in the initialisation step and updates eight weights separately during back-propagation. The network in the paper uses the symmetry constraints in the back-propagation step, and updates four weights then map these to 8 weights. Therefore, my final report will also include discussions on the mechanism of this particular network.

I also coded another network with a simpler structure. This network only has the input layer and the output layer. So this network is essentially matrix-vector multiplication. The purpose of this step is to argue the necessity of applying symmetry constraint to the weights. The outputted weights show that the network has learned successfully. So the final report is going to include discussion from this perspective as well.

Remaining work to be done before the final report deadline

1. Rewrite the back-propagation algorithm for the prototype network
2. Code the Tikhonov regularisation method
3. Write a new data generator which produces regularised training data, and apply the new data to all networks