Machine Learning Projection Methods for Macro-Finance Models: Illustrative example

Alessandro T. Villa and Vytautas Valaitis *

August 9, 2019

*Villa: Duke University, alessandro.villa@ duke.edu, Valaitis: Duke University, vytautas.valaitis@ duke.edu.

1 Illustrative example: method applied to the Stochastic Neoclassical Growth Model

For illustrative purposes we solve a simple neoclassical growth model using our ANN-based Expectations Algorithm and show that the solution coincides with the one calculated through a standard projection method. The demand side of the model is populated by a representative household that maximizes expected lifetime utility:

$$\max_{c} \quad \mathbb{E} \sum_{t=0}^{\infty} \beta^t \ln(c_t)$$

Subject to a budget constraint:

$$k_t(1 + r_t - \delta) + w_t N_t = c_t + k_{t+1}$$

Where k is capital, r is the rental rate of capital, δ is the depreciation rate, w is wage, N is labor and c is consumption. A representative firm produces output using a Cobb-Douglas technology and rents capital and labor from households:

$$\max_{N_t, K_t} \quad z_t K_t^{\alpha} N_t^{1-\alpha} - r_t K_t - w_t N_t$$

where z_t follows an AR(1) process. Taking the optimality conditions and imposing market clearing gives the Euler equation and the resource constraint:

$$\frac{1}{c_t} = \beta \mathbb{E}\left[\frac{1}{c_{t+1}} z_{t+1} K_{t+1}^{\alpha-1} + 1 - \delta\right]$$

$$c_t + K_{t+1} - (1 - \delta) K_t = z_t K_t^{\alpha}$$

We solve the model using the Euler equation with three different methods. First we use projection of the consumption policy function approximated as a third degree Chebyshev polynomial and solve for the coefficients that minimize the Euler equation residuals on a grid for capital and technology. Second and third, we use PEA and the ANN-based Expectations Algorithm to approximate the expectation term in the Euler equation and solve using stochastic simulation. The pseudo-code of the algorithm used in this section can be found in Appendix A. The following steps provides a highlight description of the algorithm:

1. Approximate the expectation term contained in the Euler Equation with ANN, as function of k_t and z_t. Given an initial guess of the ANN weights:

   - *Stochastic simulation phase:* simulate the model in time solving, for every t, for c_t and K_{t+1} given the ANN and the current state $\{K_z, z_t\}$.
   - *Training phase:*
     - *Forward phase:* Feed the ANN with the simulated path for K_t and the exogenous process z_t to generate a sequence of prediction for the expectation term.
     - *Backward phase:* Use the error between the sequence of the predicted expectation term and $\frac{1}{c_{t+1}} z_{t+1} K_{t+1}^{\alpha-1} + 1 - \delta$ to update the weights of the ANN.

2. Iterate and stop when the prediction matches the simulated data.

More details about the ANN we used can be found in table **??** in Appendix. An in-depth explanation of what an ANN is can be found in section 3.2.2. In this example we use a simple single layer ANN with 5 neurons and use Levenberg-Marquardt as a training algorithm [1]. Figure 1 shows the simulated solution for consumption and capital for each method used. Given the same initial guess, all three methods converge to the same solution. It is known that stochastic simulation only converges to a rational expectations equilibrium given a right guess. This exercise confirms that if stochastic simulation converges using a polynomial, it also does with an ANN.

[1] We use the following model parameters: $\beta = 0.95$, $\alpha = 0.36$, $\delta = 0.1$, z_t follows a log AR(1) process: $log(z_t) = \rho log(z_t) + \epsilon_t$, where $\rho = 0.8$ and $\sigma^{\epsilon} = 0.25$, simulation length $T = 500$

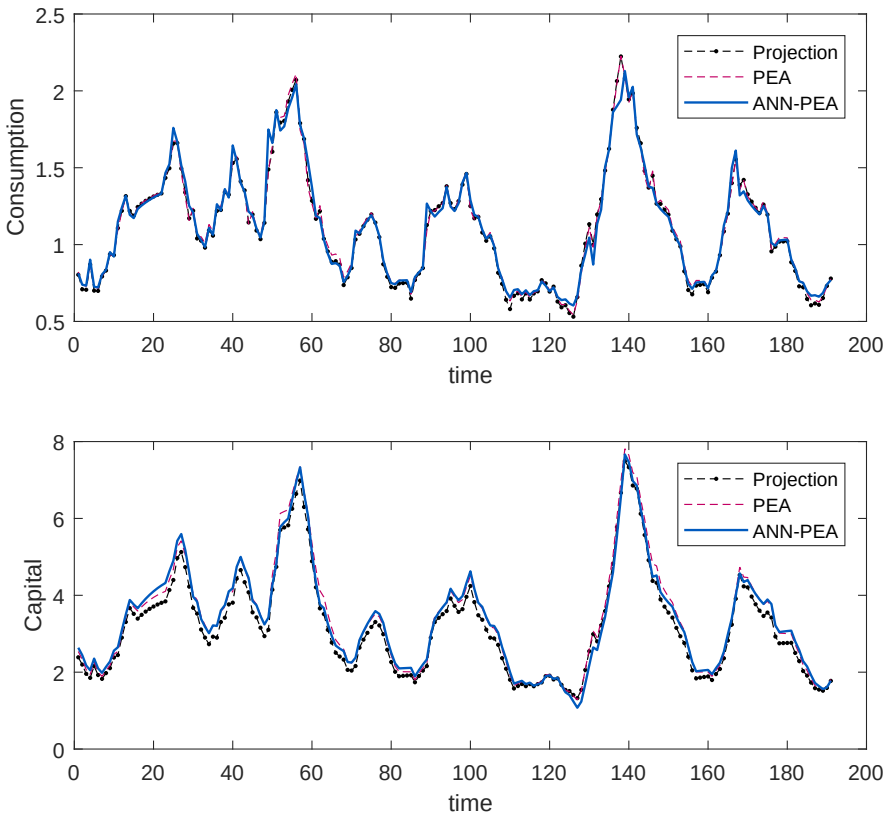

Figure 1: Top panel: simulated consumption path under three solution methods. Bottom panel: simulation capital path under three solution methods. Solid line - ANN based PEA, dashed - PEA with polynomials, dash-dot line - projection

Figure 2 shows the convergence of the ANN-based Expectations Algorithm. The left panel reports the prediction mean-squared errors on the expectation term. Note that, in the spirit of the PEA, we are comparing the ANN predictions (which are for the expectation term) with the actual realization (the argument of the expectation), therefore the errors do not necessarily converge to 0 but just stabilize. The right panel shows the norm of the difference of the ANN weights between consecutive iterations [2]. Both these measures show that the ANN converges in few iterations and that it does not oscillate along the convergence path.

[2]More information on ANN weights are in section 3.2.2.

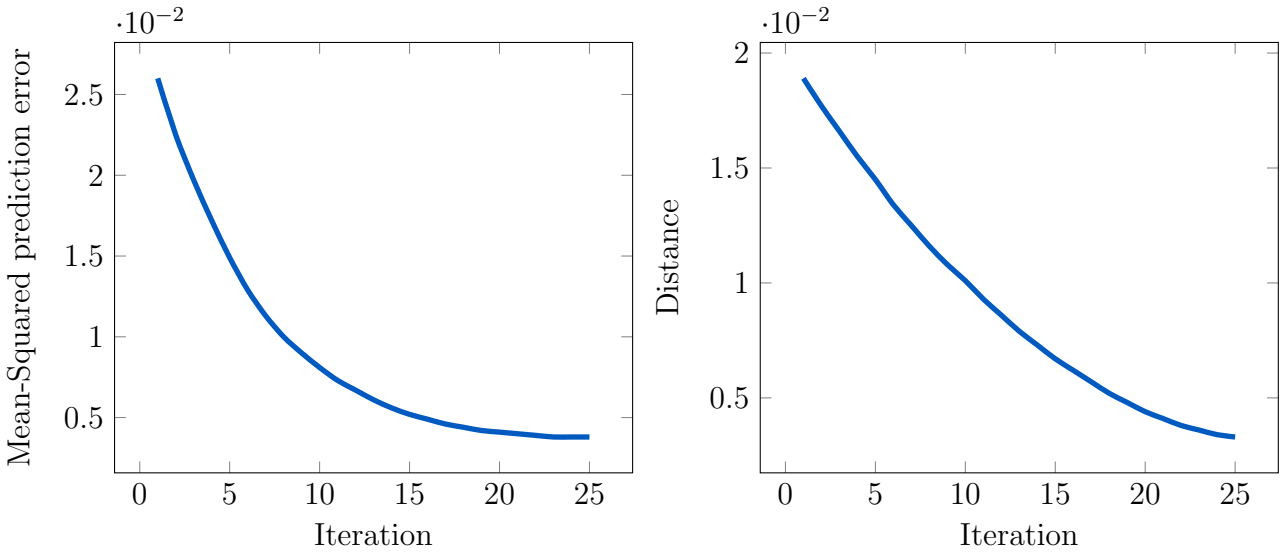

Figure 2: Network convergence, left: mean-squares ANN prediction error, right: distance between parameters of the ANN before and after training measured as the Euclidian norm

Pseudo-code of the illustrative example

Initial guess for weights of the $\mathcal{ANN}$
repeat
  //Stochastic simulation phase
  for $t = 1$ to T (large) **do**
    $\mathbb{E}_t^* = PredictExpectations(K_t, z_t, \mathcal{ANN})$
    $c_t = (\beta \mathbb{E}_t^*)^{-1}$
    $K_{t+1} = z_t K_t^{\alpha} + (1 - \delta) K_t - c_t$
  end for
  Initialize $\mathcal{ANN}$ weights
  //Training phase (equivalent to the regression in PEA)
  repeat
    for $t = 1$ to T **do**
      //Forward Pass
      $\mathbb{E}_t^* = PredictExpectations(K_t, z_t, \mathcal{ANN})$
      $Err_t = \frac{1}{c_{t+1}} z_{t+1} K_{t+1}^{\alpha - 1} + 1 - \delta - \mathbb{E}_t^*$
      //Backward Pass
      Calculate $\frac{\partial Err_t}{\partial w_{ij}}$
      $w_{ij} = w_{ij} - \alpha \frac{\partial Err_t}{\partial w_{ij}}$
    end for
  until Validation set error stops improving
until $\mathcal{ANN}$ weights keep changing