Deep and large factor models

Bryan Kelly, Boris Kuznetsov, Semyon Malamud, and Teng Andrea Xu

Giuseppe Ragusa
giuseppe.ragusa@uniroma1.it
Sapienza University of Rome

May 16, 2024

Setup

The true tradable (SDF) is

$$M_{t+1} = 1 - \pi_t' R_{t+1}$$

where π_t is the conditionally efficient portfolio

$$\pi_t \;=\; E_t[R_{t+1}R_{t+1}']^{-1}\, E_t[R_{t+1}]\,,$$

solving $\mathbf{max}_{\pi_t}\, E_t[\pi_t' R_{t+1} \;-\; 0.5(\pi_t' R_{t+1})^2]$
If X_t encompasses all relevant conditioning information

$$\pi_t = \pi(X_t)$$

for some unknown, potentially highly non-linear function $\pi(X)$.

A standard approach is to specify a parametric family of functions

$$f(x;\theta)$$

and estimate θ by

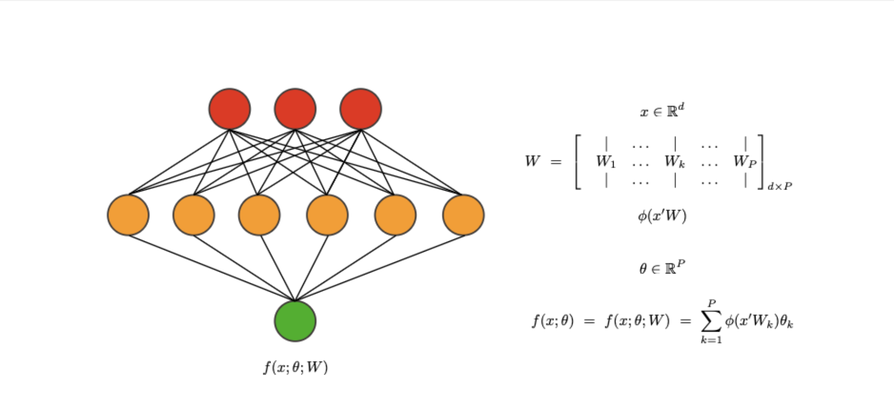
$$\min_{\theta} L(\theta)$$

where

$$L(\theta) = \left(\frac{1}{T}\sum_{t=1}^{T}(1 - f(X_t;\theta)' R_{t+1})^2 + z\,\|\theta\|^2\right).$$

Neural network

$$f(x, \theta, W) = \sum_{i=1}^{P} \phi(x' W_k) \theta_k$$



Factors:

$$F_{k,t+1} = \sum_{i=1}^{N_t} \phi(X'_{i,t} W_k) R_{i,t+1}$$

and linearity in θ gives

$$\theta = \left(zI + T^{-1} \sum_{t=1}^{T} F_t F_t \right)^{-1} T^{-1} \sum_{t=1}^{T} F_t$$

Striking

The paper builds on recent literature on ANN to highlight something that seems *striking* to the eyes of econometricians/statisticians:

More complex (ANN) models seem to do better (generalize) better than simpler models

Artificial Neural Networks

Parameterized families of function

$$f(\cdot;\theta) : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$$

- Composition of linear (affine) transformation and of fixed pointwise nonlinearity
- Parameters $\theta \in \mathbb{R}^P$ coefficient of a linear map
- Highly parameterized
  - depth and width

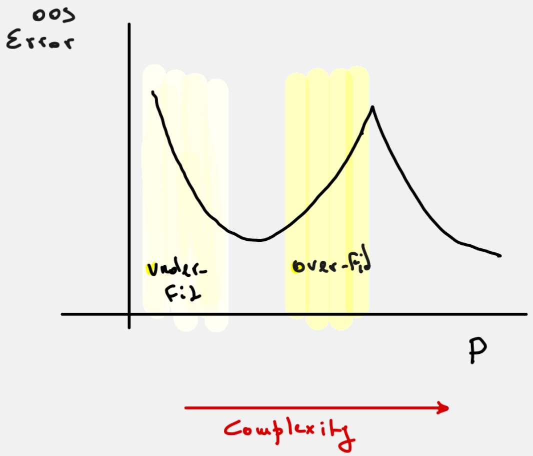
> Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators." Neural networks 2, no. 5 (1989): 359-366.

Classical viewpoint

- Risk of overfitting:
  - One should not take too many parameters
  - One should regularize
- Training a large ANN is difficult as it involves optimization of highly nonconvex functions
- Generalization performances is much less predictable than that of other methods

Recent Empirical Insights

- Taking very large ANN works much better than one would think



- Explicitly regularizing is not very useful
- Generalization performance is better than for most kernel methods

How does ANN learn from data

- Random Initialization
- Gradient descent Flow

$$\theta_{t+1} = \theta_t - \nabla_\theta C(\theta_t)$$

- Describe

$$f_{\theta_t}(x), \$t = 1, 2, \ldots, \$$$

Idea:

$$f_{\theta+1}(x_i) \approx f_{\theta_t}(x_i) - \sum_{i=1}^{N} \Theta_{\theta_t}(\tilde{x}_i, x_i) \frac{\partial C}{\partial f_{\theta_t}(\tilde{x}_i)}$$

where

$$\Theta_{\theta_t}(x_i, \tilde{x}_i) = \nabla f_\theta(x)' \nabla f_\theta(y)$$

is the infinitesimal influence of x_i (Tangent Kernel)

As the net get wider $\Theta_{\theta_t}(x_i, \tilde{x}_i)$ converges behave very nicely and it converges to an analytic object

$$\Theta_\infty(x_i, \tilde{x}_i)$$

The influence of (x_i, y_i) on the prediction can be understood using Θ_∞ even when training the network would be notfeasible or expensive

- The training dynamics are equivalent to kernel gradient descent using the NTK as the kernel.
- For specific loss, the inference performed by an ANN is in expectation equals to the ridgeless kernel regression with respect to tangent kern
- Performance of large ANNs in the NTK parametrization can be replicated by kernel methods for suitably chosen kernels

- There is evidence that NTK misses most the is going on in ANN (in other words, there are nonlinearity captured by ANN and not captured by NTK)
- The empirical application (the good performances) suggests:
  - the NTK approximation captures just the right amount of nonlinearity
  - there are still nonlinearities that we can exploit beyond those captured by the NTK
- In the kernel regime, there is not much learning of the parameters