Machine Learning in Finance

Overview

Overview

1. Approximation with Polynomials, Part I
2. Approximation with Polynomials, Part II
3. Neural Networks
4. Activation Functions
5. Types of Networks, Part I
6. Types of Networks, Part II
7. Types of Networks, Part III
8. Types of Networks, Part IV
9. Types of Networks, Part V
10. Accuracy Test Simple Neural Net vs. Polynomial

FORDHAM
THE JESUIT UNIVERSITY OF NEW YORK

Gabelli School of Business

Approximation with Polynomials, Part I

Approximation with Polynomials, Part I

- The linear model is a very simple approximation of the relationships between a set of target variables $\{y_i, i = 1, ...N\}$with covariates or characteristic variables $\{x_j, j = 1, ...K\}$.
- Polynomial expansion with second-order (quadratic), third order (cubic) or even higher orders are possible
- When there are many covariates, there is the **curse of dimensionality.**
- As we go to higher powers for polynomial expansions, the number of parameters quickly increases when we have more than one variable.
- With a second-order expansion with two variables, x_1, x_2, we have $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + \beta_4 x_1^2 + \beta_5 x_2^2$
- We thus have six parameters for a second order expansion of two variables
- For cubic expansion: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + \beta_4 x_1^2 + \beta_5 x_2^2 + \beta_4 x_1^2 + \beta_5 x_2^2 + \beta_6 x_1^2 x_2 + \beta_7 x_1 x_2^2 + \beta_8 x_1^3 + \beta_9 x_2^3$

FORDHAM

THE JESUIT UNIVERSITY OF NEW YORK

Gabelli School of Business

Approximation with Polynomials, Part II

Approximation with Polynomials: Part II

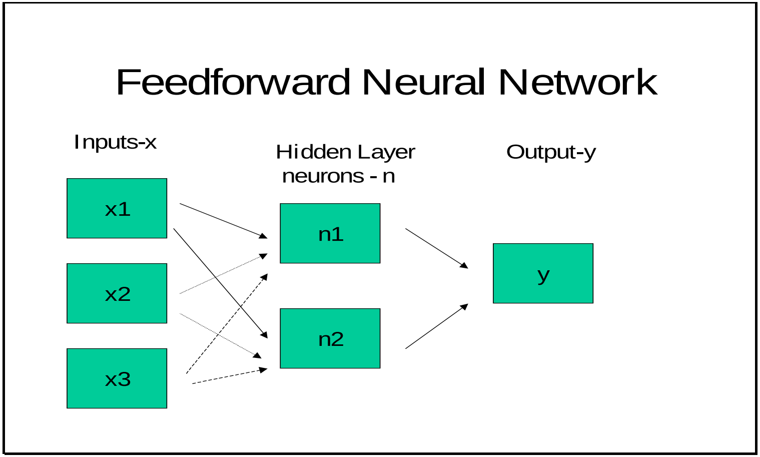
- We normalize all variables to the interval $[-1, 1]$ with the following squasher function:$x^* = \frac{2x}{\max(x)-\min(x)} - \frac{\min(x)+\max(x)}{\max(x)-\min(x)}$
- Chebeychev:
  $T_0(x^*) = 1$; $T_1(x^*) = x^*$; $T_{i+1}(x^*) = 2x^* T_i(x^*) - T_{i-1}(x^*)$
- Hermite:
  $H_0(x^*) = 1$; $H_1(x^*) = 2x^*$; $H_{i+1}(x^*) = 2x^* H_i(x^*) - 2iH_{i-1}(x^*)$
- Legendre:
  $L_0(x^*) = 1$; $L_1(x^*) = 1 - x^*$; $L_{i+1}(x^*) = \left(\frac{2i+1}{i+1}\right) L_i(x^*) - \frac{i}{i+1} L_{i-1}(x^*)$
- Laguerre: $LG_0(x^*) = 1' LG_1(x^*) = 1 - x^{*\prime} LG_i(x^*) = \left(\frac{2i+1-x^*}{i+1}\right) LG_i(x^*) - \frac{i}{i+1} LG_{i-1}(x^*)$

FORDHAM
THE JESUIT UNIVERSITY OF NEW YORK

Gabelli School of Business

Neural Networks

Neural Networks



Neural Networks

- The above chart pictures a simple feedforward neural network
- There are three inputs, $x_i, i = 1, ...3$, with one **hidden layer,** with two neurons, n_i, i=1,2, and one target or output variable,y
- This is a network with supervised learning, we are trying to fit coefficients to predict the target or out.
- More complex networks, called Deep Learning networks, have many hidden layers and there may be many output variables and inputs.
- The neurons may be specified with different functional forms.

FORDHAM
THE JESUIT UNIVERSITY OF NEW YORK

Gabelli School of Business

Activation Functions

Activation Functions

- A common specification involves the use of the logsigmoid function for the neurons in the hidden layer:

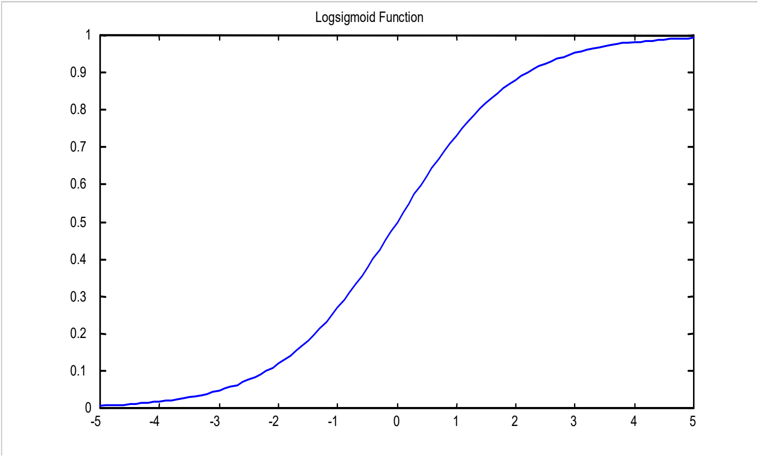
$$\begin{gathered} n_{k,t} = \omega_{k,0} + \sum_{i=1}^{i*} \omega_{k,i} x_{i,t},, \\ N_{k,t} = L(n_{k,t}) = (1/(1 + e^{-n_{k,t}})) \end{gathered}$$

$$y_t = \gamma_0 + \sum_{k=1}^{k^*} \gamma_k N_{k,t}$$

- Each neuron is a linear function of the inputs x_i with intercepts and weights $\omega_{k,i}$
- The linear functions $n_{k,t}$ are transformed by the logsigmoid function into $N_{k,t}$
- The final output or target is a linear function of the intercept and the neurons weighted by γ_k

Activation Functions

- The logistic function exhibits threshold behavior.



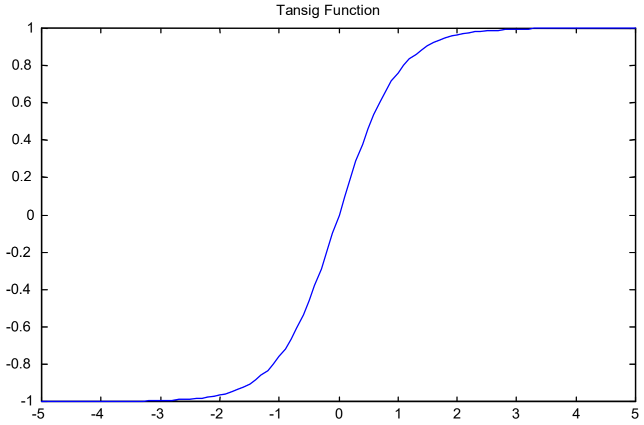
Activation Functions

- The appeal of the logsigmoid transform function comes from its "threshold behavior" which characterizes many types of economic responses to changes in fundamental variables.
- For example, if interest rates are already very low or very high, small changes in this rate will have very little effect on the decision to purchase an automobile or other consumer durable, for example.
- However within critical ranges between these two extremes, small changes may signal significant upward or downward movements and therefore create a pronounced impact on automobile demand.

Activation Functions

- Kuan and White (1994) describe this threshold feature as the "fundamental" characteristic of nonlinear response in the neural network paradigm.
- They describe it as the "tendency of certain types of neurons to be quiescent of modest levels of input activity, and to become active only after the input activity passes a certain threshold, while beyond this, increases in input activity have little further effect [Kuan and White (1994) : p.2]

Activation Functions



FORDHAM
THE JESUIT UNIVERSITY OF NEW YORK
Gabelli School of Business

Types of Networks, Part I

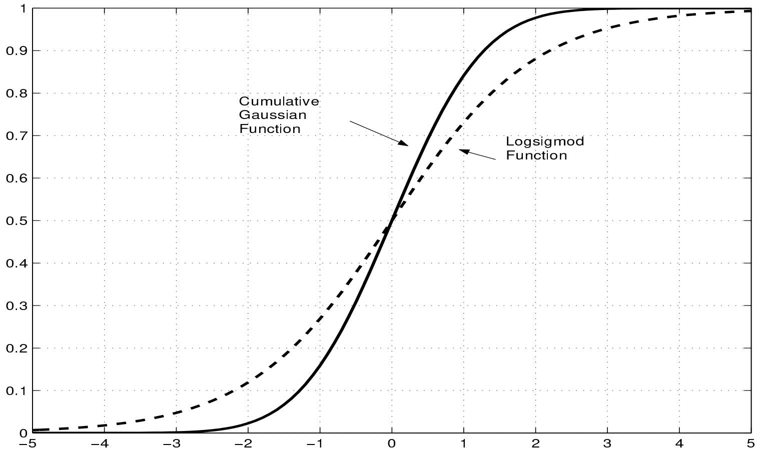
Types of Networks, Part I

$$n_{k,t} = \omega_{k,0} + \sum_{i=1}^{i*} \omega_{k,i} x_{i,t}$$

$$N_{k,t} = T(n_{k,t}) = ((e^{n_{k,t}} - e^{-n_{k,t}})/(e^{n_{k,t}} + e^{-n_{k,t}}))$$

$$y_t = \gamma_0 + \sum_{k=1}^{k^*} \gamma_k N_{k,t}$$

Types of Networks, Part I



Types of Networks, Part I

- Another is to use a **cumulative Gaussian function**:

$$n_{k,t} = \omega_{k,0} + \sum_{i=1}^{i*} \omega_{k,i} x_{i,t}$$

$$N_{k,t} = \Phi(n_{k,t}) = \int_{-\infty}^{n_{k,t}} \surd((1/(2\pi)))e^{-.5 n_{k,t}{}^2}$$

$$y_t = \gamma_0 + \sum_{k=1}^{k^*} \gamma_k N_{k,t}$$

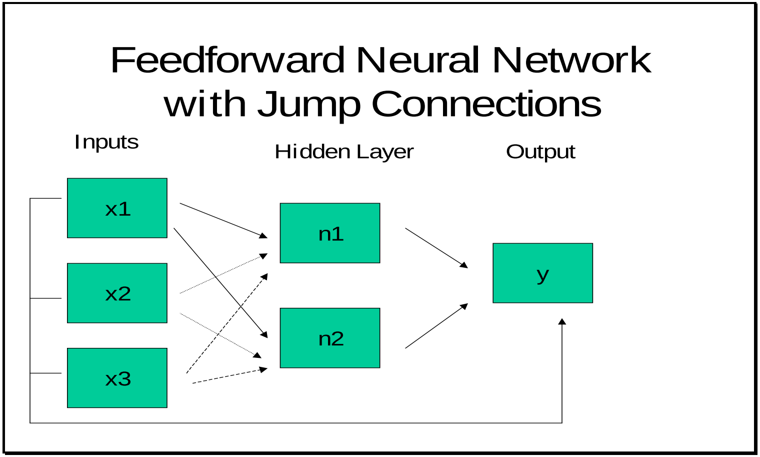
- The Gaussian function does not have as wide a distribution as the logsigmoid function
- It shows little or no response when the inputs take "extreme" values (below -2 or above +2 in this case), whereas the logsigmod does show some response.

FORDHAM
THE JESUIT UNIVERSITY OF NEW YORK

Gabelli School of Business

Types of Networks, Part II

Types of Networks, Part II



Types of Networks, Part II

- Formalization

$$n_{k,t} = \omega_{k,0} + \sum_{i=1}^{i*} \omega_{k,i} x_{i,t}$$

$$N_{k,t} = (1/(1 + e^{-n_{k,t}}))$$

$$y_t = \gamma_0 + \sum_{k=1}^{k^*} \gamma_k N_{k,t} + \sum_{i=1}^{i*} \beta_i x_{i,t}$$

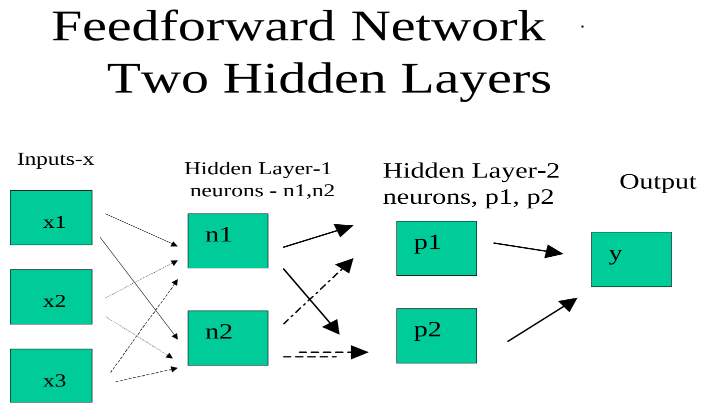
- An advantage of the feedforward network with jump connections is that it nests the pure linear model as well as the feedforward neural network.
- It allows the possibility that a function may have a linear component as well as a nonlinear component.

FORDHAM
THE JESUIT UNIVERSITY OF NEW YORK

Gabelli School of Business

Types of Networks, Part III

Types of Networks, Part III



Types of Networks, Part III

$$n_{k,t} = \omega_{k,0} + \sum_{i=1}^{i*} \omega_{k,i} x_{i,t}$$

$$N_{k,t} = (1/(1 + e^{-n_{k,t}}))$$

$$p_{l,t} = \rho_{l,0} + \sum_{k=1}^{k^*} \rho_{l,k} N_{k,t}$$

$$P_{l,t} = (1/(1 + e^{-p_{l,t}}))$$

$$y_t = \gamma_0 + \sum_{l=1}^{l^*} \gamma_l P_{l,t}$$

Types of Networks, Part III

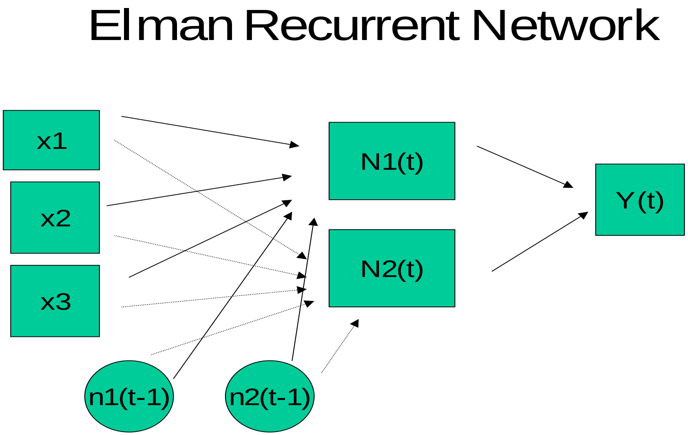
- It should be clear that adding a second hidden layer increases the number of parameters to be estimated by the factor $(k^* + 1)(l^* - 1) + (l^* + 1)$, since the feedforward network with one hidden layer, with i^* inputs and k^*neurons has $(i^* + 1)k^* + (k^* + 1)$ parameters, while a similar network with two hidden layers, with l^* neurons in the second hidden layer has $(i^* + 1)k^* + (k^* + 1)l^* + (l^* + 1)$ hidden layers.
- Of course we can have a network with three hidden layers, with more complexity.
- *Hal White: standard multilayer feedforward networks with as few as one hidden layer using arbitrary squashing functions are capable of approximating any Borel measurable function from one finite dimensional space to another to any desired degree of accuracy, provided sufficiently many hidden units are available. In this sense, multilayer feedforward networks are a class of universal approximators.*

FORDHAM
THE JESUIT UNIVERSITY OF NEW YORK

Gabelli School of Business

Types of Networks, Part IV

Types of Networks, Part IV



Types of Networks, Part IV

- Mathematics representation:

$$n_{k,t} = \omega_{k,0} + \sum_{i=1}^{i*} \omega_{k,i} x_{i,t} + \sum_{k=1}^{k*} \varphi_k n_{k,t-1}$$

$$N_{k,t} = (1/(1 + e^{-n_{i,t}}))$$

$$y_t = \gamma_0 + \sum_{k=1}^{k^*} \gamma_k N_{k,t}$$

- The Elman network is a way of capturing "memory" in financial markets, particularly for forecasting high-frequency data, such as daily, intro-daily, or even "real-time" returns in foreign exchange or share markets.
- While the use of lags certainly is one way to capture memory,

Types of Networks, Part IV

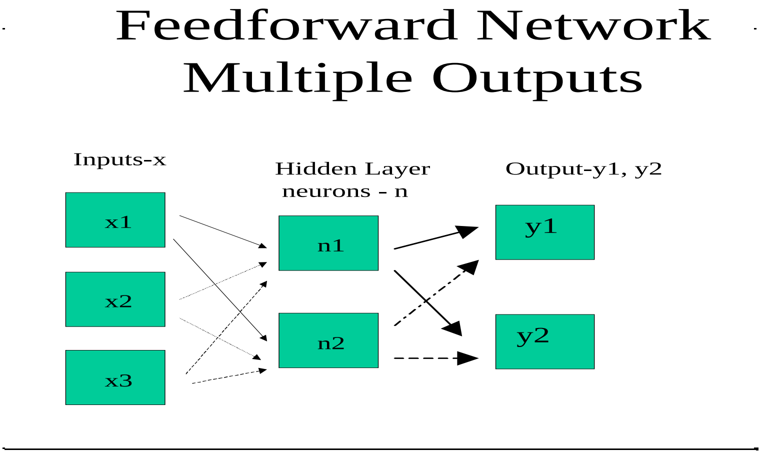
- The Elman network is an explicit dynamic network.
- The feedforward network is usually regarded as a "static" network, in which a given set of input variables at time t are used to forecast a target output variable at time t.
- Of course, the input variables used in the feedforward network may be lagged values of the output variable, so that the feedforward network becomes dynamic by re-definition of the input variables.
- The Elman network, by contrast, allows another dynamic structure beyond incorporating lagged dependent or output variables, $y_{t-1}, ...y_{t-k}$, as current input variables.
- Moreover, restricting "memory" or dynamic structure in the feedforward network only to the input "structure" may lead to an unnecessarily large number of parameters.
- While recurrent networks may be "functionally equivalent" to feedforward networks with only lagged input variables, they generally have far fewer parameters, which, of course, speeds up the estimation or "training" process.

FORDHAM | Gabelli School of Business

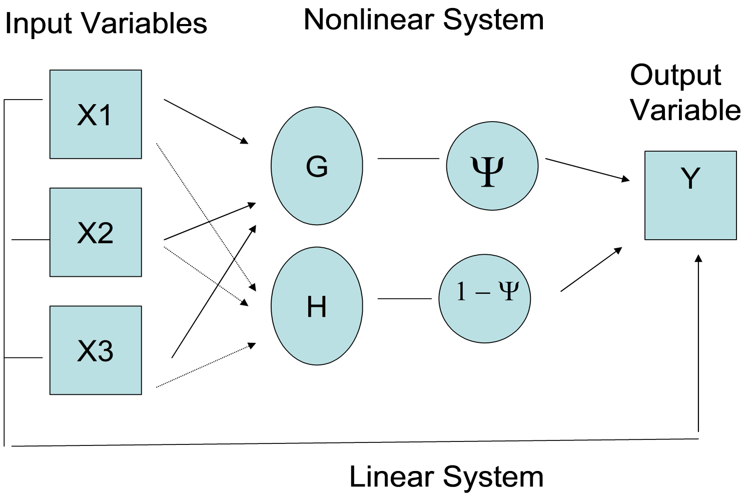
THE JESUIT UNIVERSITY OF NEW YORK

Types of Networks, Part V

Types of Networks, Part V



NNRS Model


Input Variables
Nonlinear System
Output
Variable
X1
X2
X3
G
H
Ψ
1 – Ψ
Y
Linear System

Types of Networks, Part V

center

- The smooth-transition regime switching framework for two regimes has the following form:

$$y_t = \alpha_1 x_t \Psi(y_{t-1}; \theta, c) + \alpha_2 x_t [1 - \Psi(y_{t-1}; \theta, c)]$$

- where x_t is the set of regressors at time t, α_1 represents the parameters in state 1, while $\alpha_2$is the parameter vector in state 2. The transition function Ψ, which determines the influence of each regime or state, depends on the value of y_{t-1}as well as a "smoothness" parameter vector θ and a threshold parameter c. We use a logistic or logsigmod specification for $\Psi(y_{t-1}; \theta, c)$:

$$\Psi(y_{t-1}; \theta, c) = (1/(1 + exp[-\theta(y_{t-1} - c)]))$$

- Of course, we can also use a cumulative Gaussian function instead of the logistic function. Measures of Ψ are highly useful, since they indicate the likelihood of continuing in a given state. This model, of course, can be extended to multiple states or regimes

FORDHAM

THE JESUIT UNIVERSITY OF NEW YORK

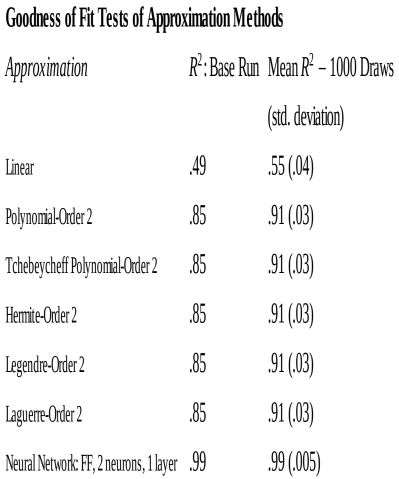
Gabelli School of Business

Accuracy Test: Simple Neural Net vs. Polynomial

Accuracy Test: Simple Neural Net vs. Polynomial

- We can see how efficient neural networks are relative to linear and polynomial approximations with a very simple example.
- We first generate a standard normal random variable x of sample size 1000
- We generate a variable $y = [sin(x)]^2 + e^{-x}$.
- We can then do a series of regressions with polynomial approximators as well as a simple neural network, with two neurons, and compare the multiple correlation coefficients or accuracy. Neural nets outperform polynomial expansion!

Accuracy Test: Simple Neural Net vs. Polynomial



Goodness of Fit Tests of Approximation Methods

Approximation	R^2: Base Run	Mean R^2 – 1000 Draws (std. deviation)
Linear	.49	.55 (.04)
Polynomial-Order 2	.85	.91 (.03)
Tchebeycheff Polynomial-Order 2	.85	.91 (.03)
Hermite-Order 2	.85	.91 (.03)
Legendre-Order 2	.85	.91 (.03)
Laguerre-Order 2	.85	.91 (.03)
Neural Network: FF, 2 neurons, 1 layer	.99	.99 (.005)

FORDHAM

THE JESUIT UNIVERSITY OF NEW YORK

Gabelli School of Business