THE UNIVERSITY OF ARIZONA
College of Engineering

Advanced Topics: Extreme Learning Machines (ELM) and Applications to SSA

R. Furfaro[1,3], R. Linares[2], K. Pula[4]

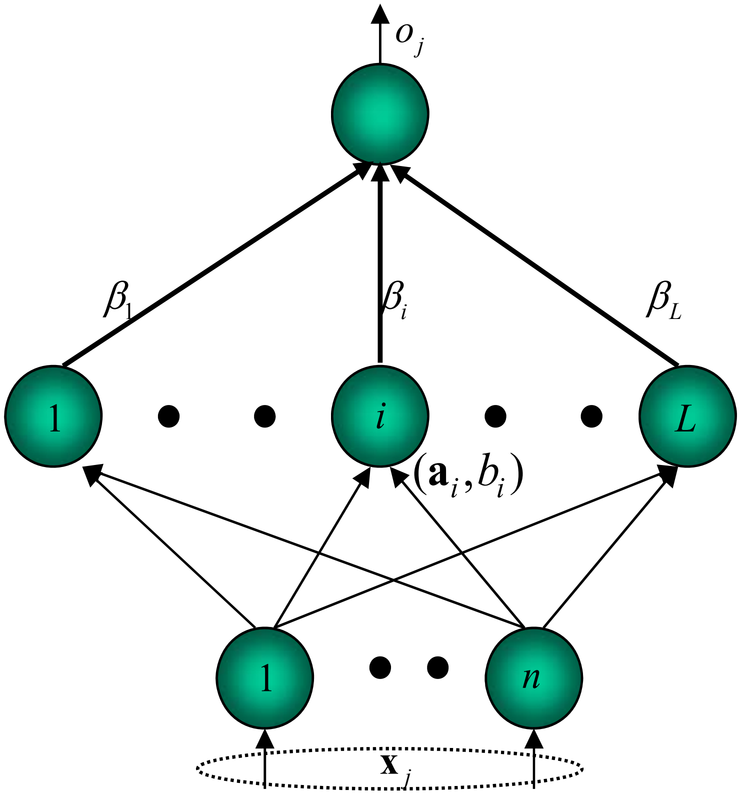
[1]Department of Systems and Industrial Engineering, University of Arizona
[2]Department of Aerospace Engineering, MIT
[3]Department of Aerospace and Mechanical Engineering, University of Arizona
[4]CACI, USA

AMOS 2018 Short Course: Machine Learning for Space Situational Awareness
Sept 11, 2018 MAUI, HI

THE UNIVERSITY OF ARIZONA

Single Layer Forward Networks (SLFN)

- ***Output*** of additive/RBF hidden nodes
  - $G(\boldsymbol{w}_i, b_i, \boldsymbol{x}) = g(\boldsymbol{w}_i^T \boldsymbol{x} + b_i)$
  - $G(\boldsymbol{w}_i, b_i, \boldsymbol{x}) = g(b_i \|\boldsymbol{x} - \boldsymbol{w}_i\|)$
- ***Output*** function for the SFNL
  - $f_L(\boldsymbol{x}) = \sum_{i=1}^{L} \boldsymbol{\beta}_i G(\boldsymbol{w}_i, b_i, \boldsymbol{x})$

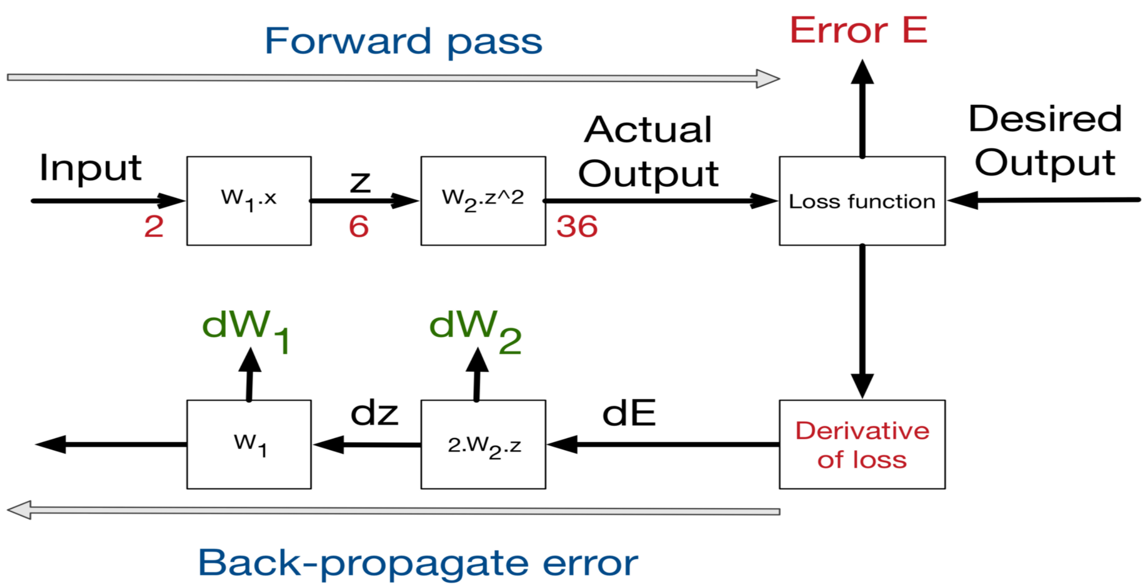


SLFN Approximation Theory

- ***Approximation (regression) capability***: Any continuous target function $f(\boldsymbol{x})$ can be approximated by a SLFN with adjustable hidden nodes.
  - Given any small positive value ε, SFLN with sufficient number of hidden nodes L can approximate the function, i.e. $\|f(\boldsymbol{x}) - f_L(\boldsymbol{x})\| < \varepsilon$ (in the L_2 sense)
- ***Classification capability:*** Any SFLN that approximate any continuous function $f(\boldsymbol{x})$, SFLN can differentiate any disjoint region

SLFN Learning Methods

- There are a ***plethora of methods*** for learning/training SFLN
  - Mostly ***iterative methods*** based on gradient descent
  - Based on variants on ***backpropagation***
  - ***Deep Learning*** falls under this category



Extreme Learning Machines

- ***Question:*** Do we need to iteratively tuning all the SLFN neurons for effective learning? Is no-tuning learning possible?
- ***A New Learning Theory:*** Learning without iteratively tuning hidden neurons in general architecture.
  - All hidden nodes parameters can be randomly generated without training data
  - Direct biological evidence has been found in 2013

G.-B. Huang, L. Chen and C.-K. Siew, “Universal Approximation Using Incremental Constructive Feedforward Networks with Random Hidden Nodes”, IEEE Transactions on Neural Networks, vol. 17, no. 4, pp. 879-892, 2006.

Extreme Learning Machines (ELM) Theory (I)

Consider a SLFN with L hidden nodes. The output function is represented as

$$f_L(x) = \sum_{i=1}^{L} \beta_i g_i(x) = \sum_{i=1}^{L} \beta_i G(a_i, b_i, x)$$

$$with \quad x \in R^d, \beta_i \in R^m$$

Training set with N distinct samples

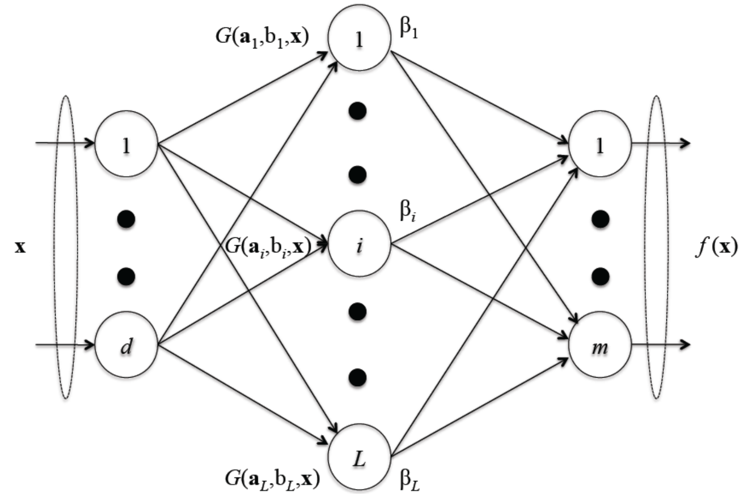
$$[x_i, t_i] \in R^d \times R^m$$

SLFN to approximate the N samples as described in the training set

$$\sum_{i=1}^{L} \beta_i G(a_i, b_i, x) = t_j, \quad for\ j = 1, \dots\dots, N$$

$$H\beta = T$$

Single Layer Forward Network (SLFN)



Additive Node Activation FCN

$$G(a_i, b_i, x) = g(a_i x + b_i) \quad with \quad a_i \in R^m, b_i \in R$$

Radial Basis Activation FCN

$$G(a_i, b_i, x) = g(b_i \|x - a_i\|) \quad with \quad a_i \in R^m, b_i \in R^+$$

Extreme Learning Machines (ELM) Theory (II)

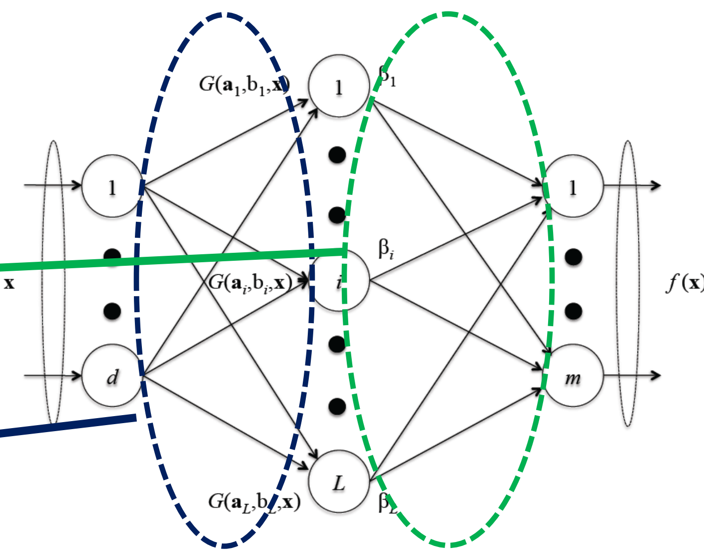
$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{h}(\boldsymbol{x}_1) \\ \vdots \\ \boldsymbol{h}(\boldsymbol{x}_N) \end{bmatrix} = \begin{bmatrix} G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_1) & \dots & G(\boldsymbol{a}_L, b_L, \boldsymbol{x}_1) \\ \vdots & \dots & \vdots \\ G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_N) & \dots & G(\boldsymbol{a}_L, b_L, \boldsymbol{x}_N) \end{bmatrix}_{N\times L} \qquad \boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^T \\ \vdots \\ \boldsymbol{\beta}_L^T \end{bmatrix}_{L\times m} \text{ and } \quad \boldsymbol{T} = \begin{bmatrix} \boldsymbol{t}_1^T \\ \vdots \\ \boldsymbol{t}_N^T \end{bmatrix}_{N\times m}$$

Theorem [36]: Given any non-constant piecewise continuous function $g: R \to R$, if $span\{G(\boldsymbol{a}, b, \boldsymbol{x}) : (\boldsymbol{a}, b) \in \boldsymbol{R}^d \times R\}$ is dense in L^2, any continuous target function f and any function sequence $\{g_L(\boldsymbol{x}) = G(\boldsymbol{a}_L, b_L, \boldsymbol{x})\}$ randomly generated based on any continuous sampling distribution, $\lim_{L\to\infty} \|f - f_L\|$ holds with probability one if the output weights $\boldsymbol{\beta}_i$ are determined by ordinary least square to minimize $\|f(\boldsymbol{x}) - \sum_{i=1}^{L} \boldsymbol{\beta}_i g_i(\boldsymbol{x})\|$.

Least-Square Solution

$$\|\boldsymbol{H}\hat{\boldsymbol{\beta}} - \boldsymbol{T}\| = \min_{\boldsymbol{\beta}} \|\boldsymbol{H}\boldsymbol{\beta} - \boldsymbol{T}\|$$

Randomly Generated



Three-Step Process for ELM Training

- Given a training set $\{(\boldsymbol{x}_i, \boldsymbol{t}_i) | \boldsymbol{x}_i \in \mathbb{R}^d, \boldsymbol{t}_i \in \mathbb{R}^m, i = 1, \dots, N\}$, hidden node output function $G(\boldsymbol{w}_i, b_i, \boldsymbol{x})$, and the number of hidden nodes L
    - **Step 1**: Assign randomly hidden nodes parameters $(\boldsymbol{w}_i, b_i)$
    - **Step 2**: Calculate the hidden layer output matrix $\boldsymbol{H} = [\boldsymbol{h}(\boldsymbol{x}_1) \dots . \boldsymbol{h}(\boldsymbol{x}_N)]^T$
    - **Step 3:** Calculate the output weights $\boldsymbol{\beta}$

Example: Using ELM to map measurements into orbit parameters

- Generally, characterizing the behavior of RSOs requires a thorough understanding of the functional ***relationship between sensor measurements and object energy and state parameters***
  - Such functional relationship is generally representative of the physical processes underlying the interaction between the RSO and its environment
- ***Question***: How do we efficiently and accurately represent the relationship between sensor measurements and RSO energy and state parameters?
- ***Proposed Approach:*** Characterize the functional relationship between sensor measurements and RSOs behavior by using a data-driven, physically-based approach based on ELM

Furfaro, R., Linares, R., Jah, M. K., & Gaylor, D. (2016, September). Mapping Sensors Measurements to the Resident Space Objects Behavior Energy and State Parameters Space via Extreme Learning Machines. In *International Astronautical Congress (IAC)*

Problem Formulation

- Orbit Determination is an ***Inverse Problem*** (of Dynamical System)
- ***Goal:*** Estimate $\boldsymbol{x}_k$ for time t_k from measurements $\boldsymbol{y}_k \; for \; k = 1, \ldots, m$
- Model

$$\mathbf{x}_{k+1} = \mathbf{f}_k(\mathbf{x}_k)$$

$$\mathbf{y}_k = \mathbf{h}_k(\mathbf{x}_k) + \mathbf{n}_k$$

Noise

$$\mathbf{n}_i \sim \mathcal{N}(\mathbf{n}_i; 0, \Sigma_i^2)$$

$$E\{\mathbf{n}_i \mathbf{n}_j\} = \delta_{ij} \Sigma_i^2$$

- We solve the inverse mapping $\boldsymbol{x}_0 = \boldsymbol{f}_L(\boldsymbol{y}, \boldsymbol{\Theta})$ where the function $\boldsymbol{f}_L$ is learned directly from simulated data samples $\{\boldsymbol{x}_{0i}, \boldsymbol{y}_i\}_{i=1}^N$
  - Use ***Extreme Learning Machines (ELM) Theories***

Training Data Generation: Model and Sampling

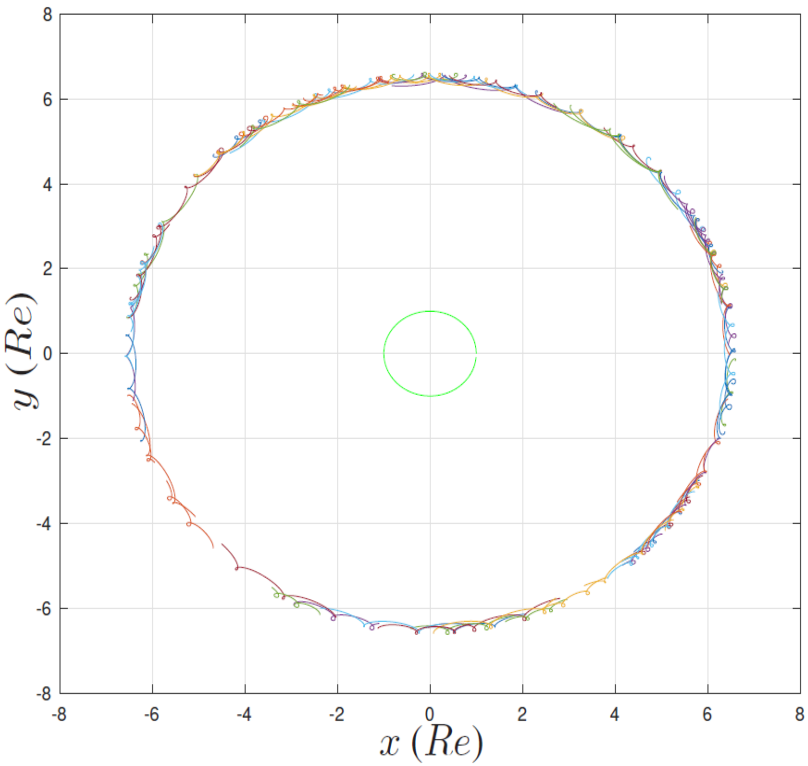
- ***Population of Near GEO RSO:*** Consider a 2-D (Equatorial) Orbital Motion and Measurements Model

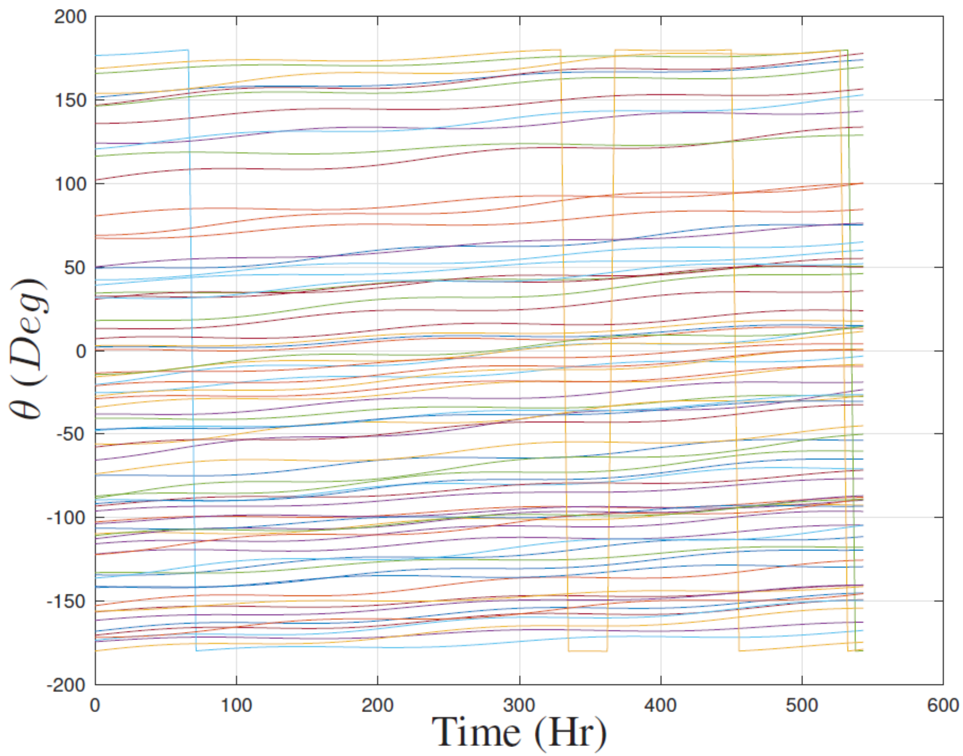
$$\ddot{x} + \frac{\mu x}{r^3} = 0 \qquad \mathbf{r} = \begin{bmatrix} x & y \end{bmatrix}^T \qquad \mathbf{u} = \begin{bmatrix} u_x, & u_y \end{bmatrix}^T$$

$$\ddot{y} + \frac{\mu y}{r^3} = 0 \qquad \theta = \operatorname{atan2}\left(u_y, u_x\right) \qquad \mathbf{u} = \mathbf{r} - \mathbf{r}_{obs}$$

- Satellites are described by the classical orbital parameters $\boldsymbol{OE} = [a, e, i, M]^T$ and sampled randomly of distribution with specified statistics
  - $\mu_a = 42{,}164km, \sigma_a = 100km,\ \mu_\omega = 0deg, \sigma_\omega = 90deg, \mu_M = 0deg, \sigma_M = 90deg$
    - Gaussian Distribution
  - $e \in [0.01, 0.02]$
    - uniform distribution
- ***OE are converted*** to an initial $\boldsymbol{r}$ *and* $\boldsymbol{v}$ and simulated for 72 hrs

Example of Training Data





Near-GEO Training Trajectories Examples in Fixed Reference Frame

Angle Measurement Training Data

SFLN Architecture and Training Approach

- ***SLFN Network Architecture***
  - 100 equally space measurements
  - Employ 100-18,500-3 SLFN architecture (18,500 hidden neurons/nodes)
  - Regularization parameter ~10^{-5}
- ***Training and Validation***
  - 10,000 training orbits
  - Use 9,000 points for training (90%) and 1,000 points for test/validation (10%)
  - Random selection
  - Training Time < 1sec
- ***Predict the Poincare Orbital Elements (POE)***
  - Non-singular for circular/zero inclination
  - Mapping to OE is known

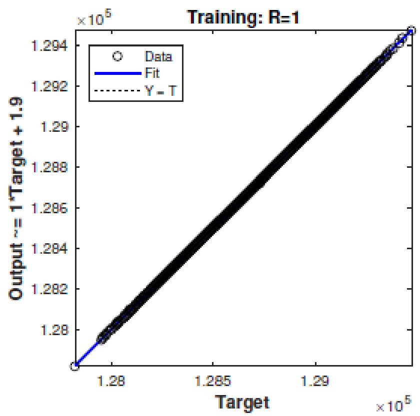
$$L = \sqrt{\mu a}, \quad I = \omega + M$$

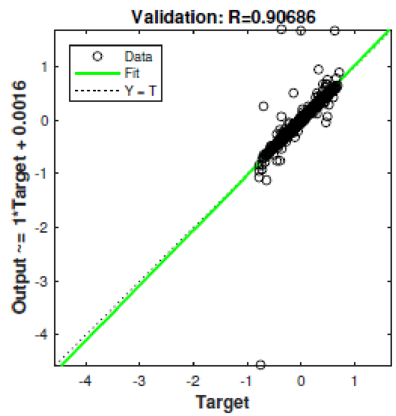
$$g = \sqrt{2L\left(1 - \sqrt{1 - e^2}\right)}\cos(\omega)$$

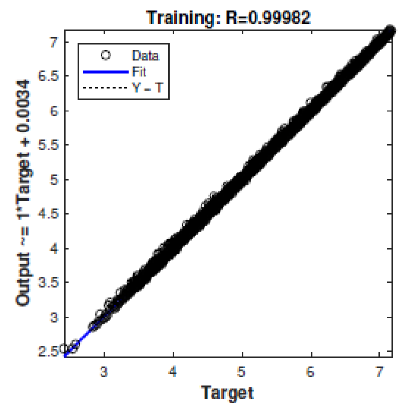
$$H = -g\tan(\omega)$$

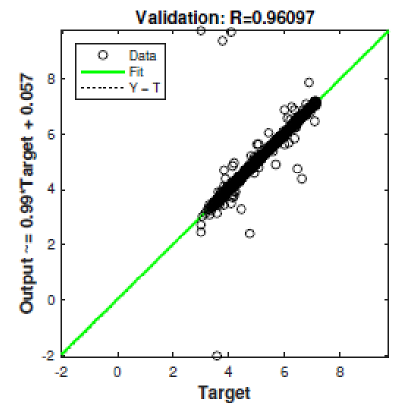
Training Results

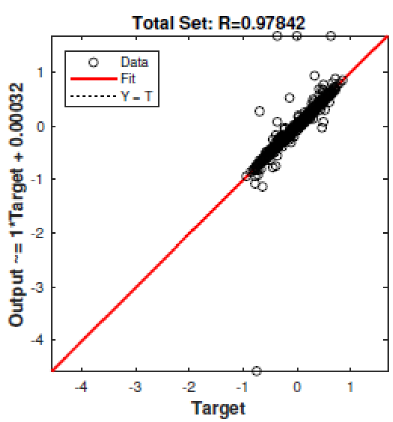
- I is related to the angular parameters, g is related to eccentricity
  - I: Training $R^2 = 0.999$, Training $R^2 = 0.960$
  - g: Training $R^2 = 1.000$, Validation $R^2 = 0.906$



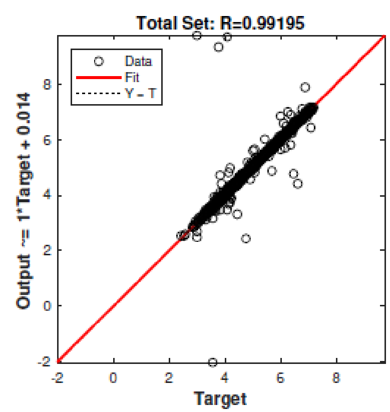








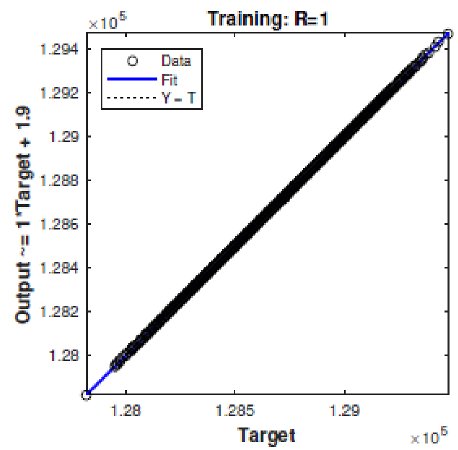
Poincare's Orbital Element g

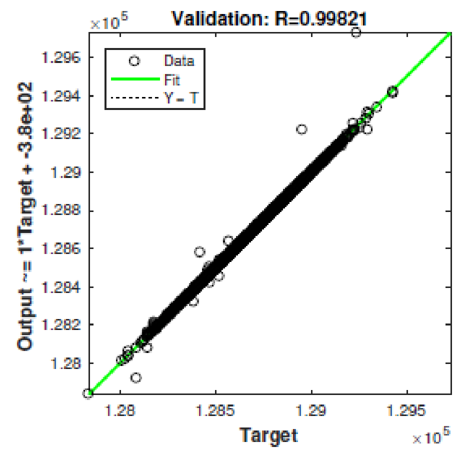


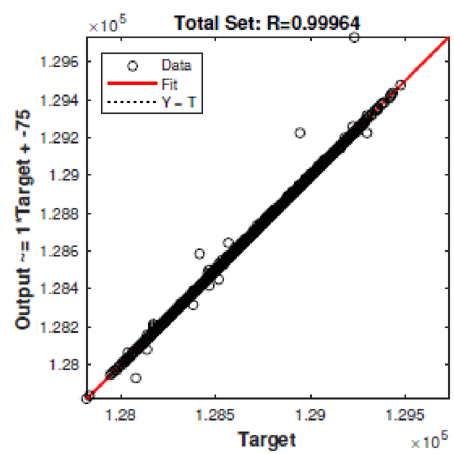
Poincare's Orbital Element I

Training Results

- L is related to the size of the orbit
  - Training $R^2 = 1.000$
  - Validation $R^2 = 0.998$







Poincare's Orbital Element L

Questions?