

Deep Learning Methods to IOD

Wasif Islam

School of Electrical and Computer Engineering

islam25@purdue.edu

w.islam@obscuritylabs.com



PURDUE
UNIVERSITY®

Elmore Family School of Electrical
and Computer Engineering

Introduction

- Space Objects (SO) (both debris and non-debris) concentration around Earth is projected to increase exponentially
- Initial Orbit Determination (IOD) is critical in establishing a catalog of SOs. Accuracy of IOD is important in achieving timely orbit improvement
- Furfaro presents an introductory experiment in determining feasibility of performing optical IOD of restricted orbits [2].

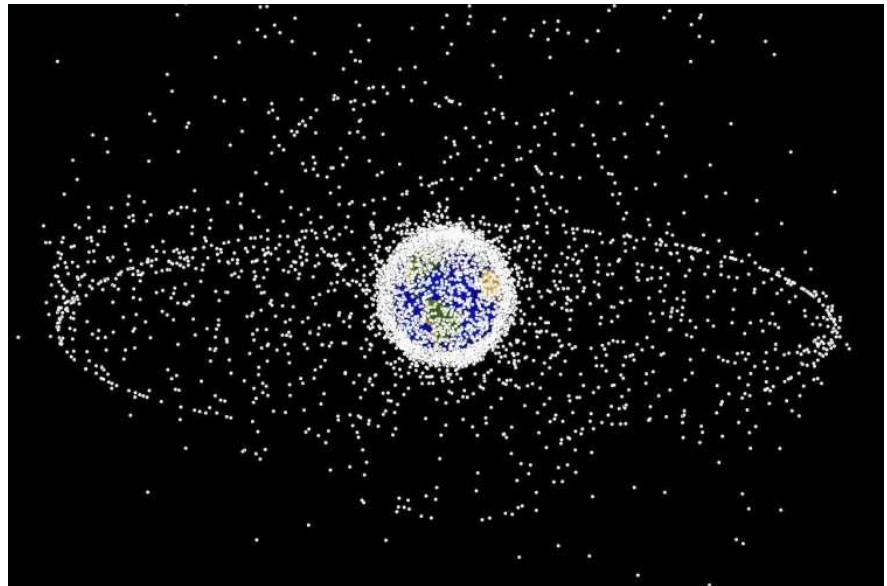


Figure 1: Space Debris Projection [1]

Orbital Debris Presents Unique SDA Challenges

Justification

- Many IOD methods depend on simplifying assumptions (ex. 2-body orbit, coplanar observations)
- IOD is reliant on orbit improvement techniques to reach orbit accuracy (LUMVE, Kalman Filter). Increased accuracy of IOD would decrease uncertainty of downstream improvements
- Increased sensor and observation availability provides window for techniques involving big data analytics
- Machine Learning (ML), paired with availability of relevant data provide alternate path towards approximating difficult non-linear dynamics of orbits

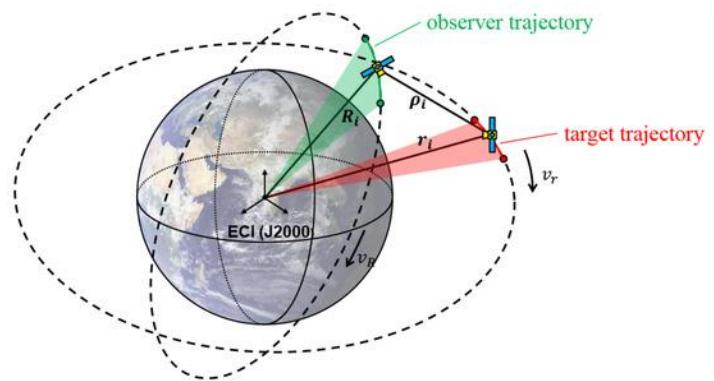


Figure 2: IOD Inaccuracy [4]

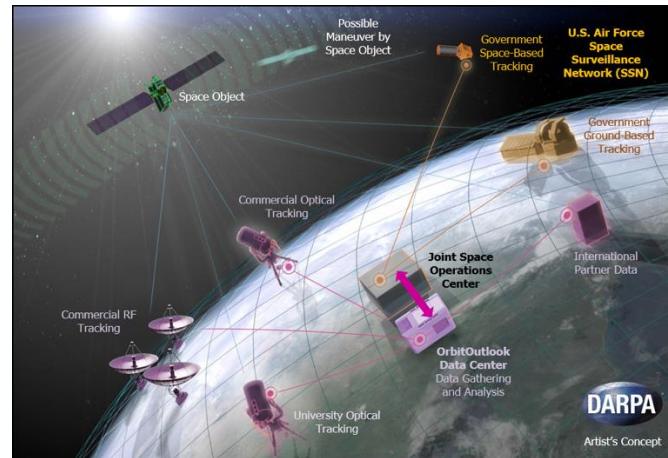


Figure 3: DARPA OrbitOutlook Program [3]

Opportunity to Explore Function Approximators Mapping Observations to State

Research Context

- Furfaro, et al performed introductory experiments on function approximation of restricted orbit dynamics[2]:
 - Planar GEO Orbit (0-degree inclination)
 - Almost zero eccentricity
 - Geostationary orbit (ECEF Right Ascension measures are constant)
- Function Approximator:
 - Neural Network: Allows the single or multi-layered perceptron to extract patterns from available features
- Dataset:
 - Feature Space: Angle for Right Ascension (single variable)
 - Label Space: Poincare Orbital Elements

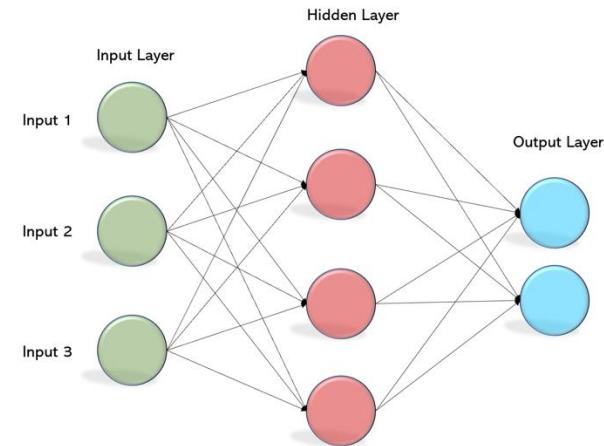


Figure 4: IOD Multi-Layer Perceptron [5]

$$L = \sqrt{\mu a}, \quad I = \omega + M$$
$$g = \sqrt{2L \left(1 - \sqrt{1 - e^2}\right)} \cos(\omega)$$
$$H = -g \tan(\omega)$$

Figure 5: Mapping of Poincare Elements [2]

Neural Networks Provide Function Approximation Capabilities

Author Methodology

- Dataset:
 - Simulated data: 1000 initial state samples generated from a defined normal distribution.
 - Keplerian orbital elements
 - Singular ground station used to define topographic right ascension
 - Right ascension angles injected with measurement noise (~1.5 arcseconds)
- Model:
 - Extreme Learning Machine (ELM):
 - Single Hidden Layer Feed Forward
 - Hidden layer is randomly initialized
 - Output layer learnable (used for function approximation)
- Training:
 - 90/10 split for training and validation
 - No information given on number of epochs used

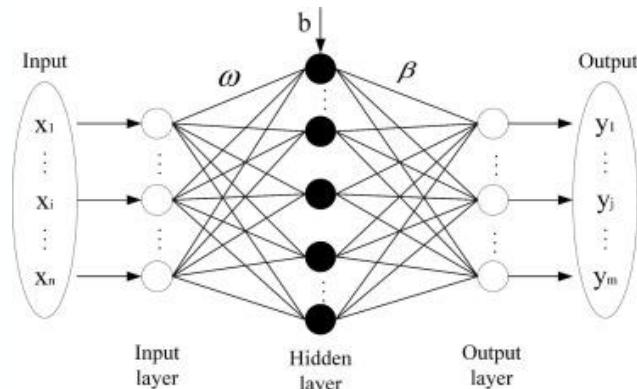


Figure 6: Extreme Learning Machine [2]

Experiment Setup Details Minimal but Reproducible

Author Results

- Analysis involved performing regression analysis in the form of correlation analysis[2]:
 - Correlation calculated for each Poincare orbital element.
 - All correlation values are higher than 0.90
 - Although results show high correlation, does not showcase the ability of neural networks to approximate functions

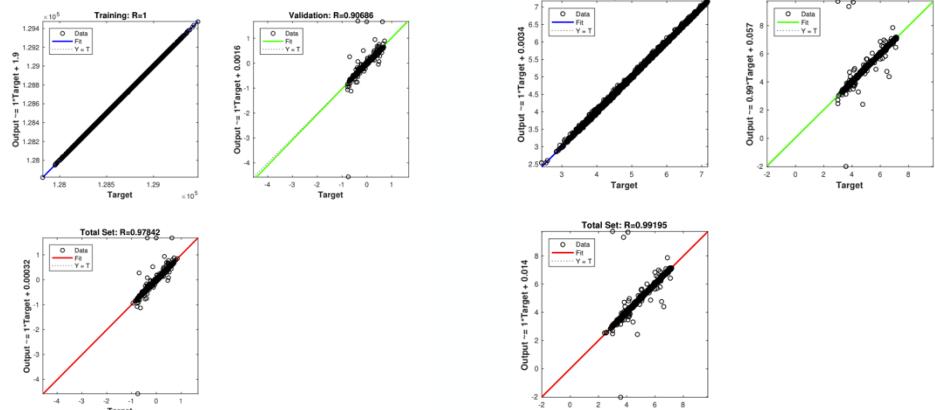


Figure 7: Orbital Element I [2]

Figure 8: Orbital Element g [2]

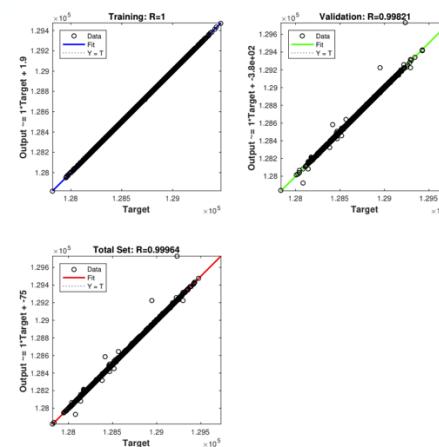


Figure 9: Orbital Element L [2]

Experiment Setup Details Minimal but Reproducible

Review Methodology

- No source code or pseudocode is presented by the author to reproduce the work. All code had to be developed to the spec defined in the paper.
- The same experiment is designed and executed. Changes involved the number of epochs to train the ELM model and neuron count. (Iterative training passed in as hyper-parameters)
- Tech Stack:
 - Python 3.12
 - PyTorch
 - Jupyter Notebook

```
Training Loop

In [3]: import copy
import torch
epochs = 1000

best_val_loss = float('inf')
best_model_state = copy.deepcopy(iod_elm.state_dict())
for epoch in range(1, epochs + 1):
    iod_elm.train()
    epoch_loss = 0.0
    for batch_features, batch_labels in train_dataloader:
        # forward pass
        out = iod_elm(batch_features)
        loss = criterion(out, batch_labels)

        # Backward pass and optimization (note we dont calculate gradients for weights and biases)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        epoch_loss += loss.item() * batch_features.size(0)

    avg_loss = epoch_loss / train_size
    if avg_loss < best_val_loss:
        best_val_loss = avg_loss
        best_model_state = copy.deepcopy(iod_elm.state_dict())
        torch.save(best_model_state, 'best_elm_state.pth')

    if epoch % 50 == 0 or epoch == 1:
        print(f'Epoch [{epoch}/{(epoch)}], Training Loss: {avg_loss:.4f}')

Epoch [1/1000], Training Loss: 1668769222.6568
Epoch [50/1000], Training Loss: 476976544.0088
Epoch [100/1000], Training Loss: 51914991.9368
Epoch [150/1000], Training Loss: 132729.5862
Epoch [200/1000], Training Loss: 251952.0158
Epoch [250/1000], Training Loss: 248000.0000
Epoch [300/1000], Training Loss: 408564.4095
Epoch [350/1000], Training Loss: 162245.9161
Epoch [400/1000], Training Loss: 355327.6830
Epoch [450/1000], Training Loss: 140708.7448
Epoch [500/1000], Training Loss: 58154.9292
Epoch [550/1000], Training Loss: 65445.9748
Epoch [600/1000], Training Loss: 58961.3938
Epoch [650/1000], Training Loss: 58961.3915
Epoch [700/1000], Training Loss: 141225.6588
Epoch [750/1000], Training Loss: 211067.6297
Epoch [800/1000], Training Loss: 40157.0683
Epoch [850/1000], Training Loss: 55267.9289
Epoch [900/1000], Training Loss: 40045.4889
Epoch [950/1000], Training Loss: 181887.0095
Epoch [1000/1000], Training Loss: 50263.3183
```

Figure 10: ELM Training Loop

Review Implementation: <https://github.com/OnlyUseMeRocket/sda-ml>

Review Results

- Results are not consistent with what the paper is implying
- The average MSE value per epoch is ranking very high.
 - MSE value magnitude fluctuating between 10e4 and 10e5
 - Visual representation of labels vs. predicted show there is no significance to the approach the model is defined

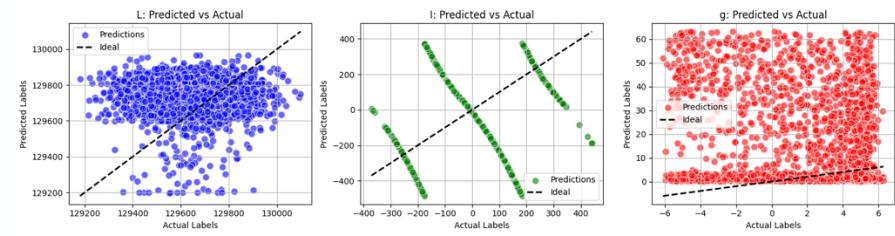


Figure 10: ELM Validation Results

- Expected results as single feature function approximators are not well defined.
- Training time overfitting is a real problem

Review Implementation: <https://github.com/OnlyUseMeRocket/sda-ml>

Observatio ns

- Feature selection seems unfinished. Single feature approximation is proven to not yield good results as the system of equations surrounding weights and biases are under-defined.
- The results the author presents in the paper maybe a result of over-fitting of the model to the data. It is hard to make definitive conclusion due to lack of setup definition
- Inferencing orbital elements based on single observations will not yield accurate results as the temporal patterns of a satellite orbit are not embedded in the feature space
- Adding more variables to the feature space is needed to improve model performance at a minimum

Future Work

- Increase data generalization by using propagated satellite state. (I.e propagate initial state generated by data generator)
- Increase feature space: add observation characterizing variables (eclipse state, position of earth w.r.t sun, optical equipment parameters)
- Change label space to include full Kepler orbital state parameters
- Consider evaluating different model architecture, possibly multi-hidden layer architectures

Given limited setup breakdown and no source code available, it is hard to reproduce author results

References

- [1] The Aerospace Corporation. (2024). *Space Debris Projection*. A Brief History of Space Debris. <https://aerospace.org/article/brief-history-space-debris>
- [2] Furfaro, Roberto. Et al. (2016). *Mapping Sensors Measurements to the Resident Space Objects Behavior Energy and State Parameters Space via Extreme Learning Machines*. International Astronautical Congress. <https://experts.arizona.edu/en/publications/mapping-sensors-measurements-to-resident-space-objects-energy-and>
- [3] Pellerin, Cheryl. (2016). *DARPA OrbitOutlook Program*. Top Official Describes DARPA Real-Time Space Awareness Efforts. <https://www.defense.gov/News/News-Stories/Article/Article/984655/top-official-describes-darpa-real-time-space-awareness-efforts/>
- [4] Xie, Hui. (2022). *A Multimodal Differential Evolution Algorithm in Initial Orbit Determination for a Space-Based Too Short Arc*. MDPI, Remote Sensing 2022. <https://www.mdpi.com/2072-4292/14/20/5140>
- [5] Le, James. (2020). *Multi-Layer Perceptron*. Recommendation System Series Part 5: The 5 Variants of Multi-Layer Perceptron for Collaborative Filtering. <https://jameskle.com/writes/rec-sys-part-5>
- [6] Chen, Qiuan, Et al. (2021). *Kernel extreme learning machine based hierarchical machine learning for multi-type and concurrent fault diagnosis*. Measurement, Volume 184. <https://www.sciencedirect.com/science/article/abs/pii/S0263224121008605>