CS726 Project
Legendre Memory Units

Siddharth Chandak	17D070019
Syomantak Chaudhuri	170070004
Mithilesh Vaidya	17D070011
Abhishek Tanpure	17D070018
Vaidehi Patil	170110012

Tasks

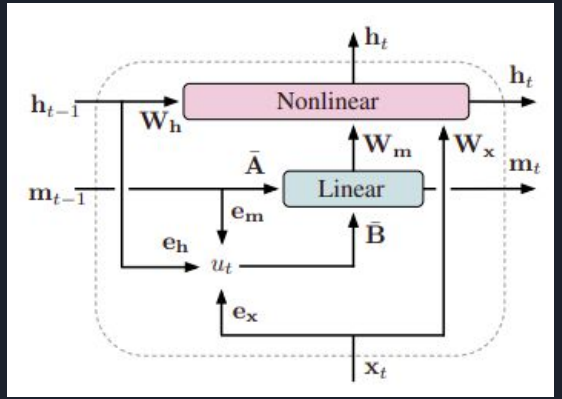
- To understand, implement and analyze the performance of Legendre Memory Units on various datasets
- Suggest improvements/modifications to LMUs

What is LMU?

- A memory cell for RNNs that stores information across long windows of time
- Mathematical basis: Padé approximants for exponentials and Legendre polynomials [2]
- Shown to outperform LSTMs in some cases

$$\mathbf{A} = [a]_{ij} \in \mathbb{R}^{d\times d}, \quad a_{ij} = (2i+1)\begin{cases} -1 & i < j \\ (-1)^{i-j+1} & i \geq j \end{cases}$$

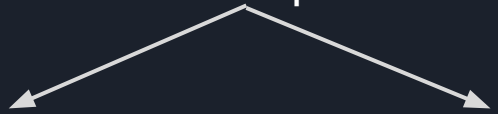
$$\mathbf{B} = [b]_i \in \mathbb{R}^{d\times 1}, \quad b_i = (2i+1)(-1)^i, \quad i,j \in [0, d-1].$$



Images reproduced from [1]

Innovations

Two Perspectives



Mathematical

- Deriving new matrices A & B based on intuition
- **Bernstein Memory Units**
- **Padé Memory Units**
- **Euler-Legendre Memory Units**

Architectural

- Modifying the architecture based on observations
- **Linear Legendre Memory Units**
- Trainable Memory Units

We tried several modifications but will focus on only two of them.

Bernstein Memory Units (BMU)

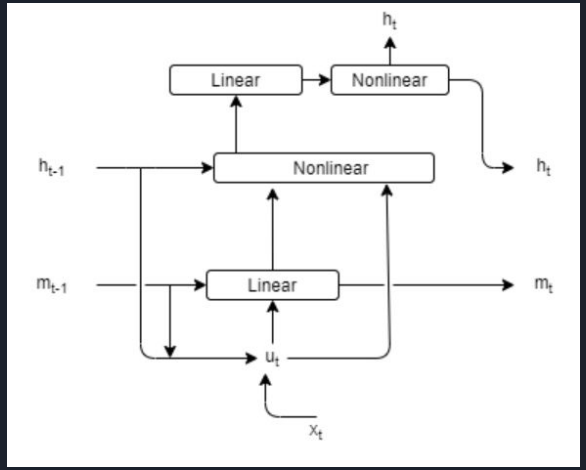
- Motivation - Justification behind using Legendre very weak
- Bernstein Polynomials - Another set of basis polynomials
- Popular for the proof of Weierstrass Approximation Theorem
- Simpler than Legendre polynomials
- Derived transformation matrix *M* based on derivation of LMU and the relation between Legendre and Bernstein polynomials [3]

$$A_b = M A_l M^{-1}$$

$$B_b = M B_l$$

Linear Legendre Memory Units

- Motivation: A linear layer after the output of LMU drastically improves the performance
- Use a linear and a nonlinear layer for the next hidden state as well



Experiments

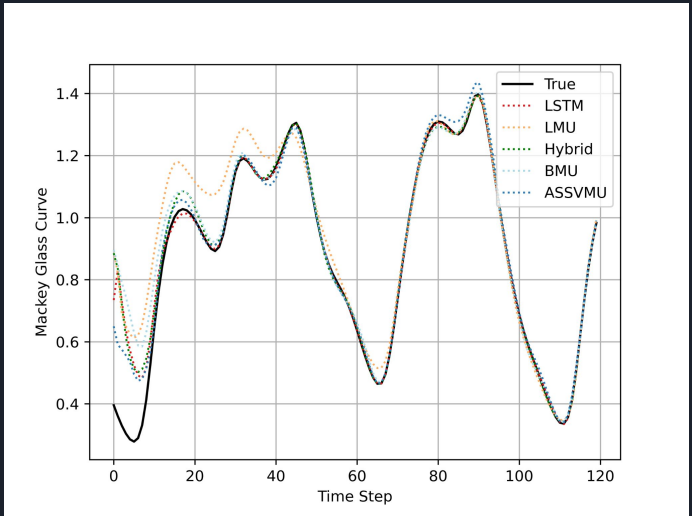
- Mackey Glass Equation
- Sequential/Permuted Sequential MNIST
- Capacity task
- PennTreebank Character Modeling*
- JSB Chorales*
- LAMBADA dataset*

(*Additional tasks commonly used to benchmark performance of sequence modeling architectures [4])

(Our Code: https://github.com/CyanideBoy/LegendreMemoryUnits)

Mackey Glass

- MG equation - Parameterized differential equation
- Input - Sequence of points generated by the MG equation
- Model has to predict 15 time steps into future
- Network contains 4 stacked layers of LMU, or, LSTM, or, hybrid model
- BMU - Best middle ground



Model	Test NRMSE	Training Time (s/epoch)
LSTM	0.044	25.45
LMU	0.056	13.2
BMU	0.044	17.84
Linear LMU	0.041	21.77
Hybrid	0.044	22.94

psMNIST

- Digit classification task
- Images from MNIST flattened into a 1-D array and permuted
- Pixels are provided at a time and model has to predict digit
- LMU and Linear LMU significantly outperform LSTM
- BMU couldn't be trained successfully

Model	Test Accuracy
LSTM	86.7%
LMU	96.39%
BMU	77.9%
Linear LMU	95.26%

JSB Chorales

- Dataset - http://www-etud.iro.umontreal.ca/~boulanni/icml2012
- Input - Sequence of multiple piano key presses at each time instant
- Have to predict piano key presses at next time instant
- Variant of multilabel classification with cross entropy loss
- BMU and LLMU clearly outperform both LSTM and LMU

Model	Test Loss
LSTM	10.8329
LMU	10.5413
BMU	10.4623
Linear LMU	10.2347

PennTreebank Character Modeling

- Large corpus of english text: https://catalog.ldc.upenn.edu/LDC99T42
- Each character is encoded as a one-hot vector and fed to model as input
- Have to predict next character
- Trained using cross-entropy loss
- LSTM slightly outperforms the memory units

Model	Test Loss
LSTM	2.932
LMU	2.957
BMU	2.957
Linear LMU	2.957

LAMBADA

- Dataset - https://wiki.cimec.unitn.it/tiki-index.php?page=CLIC
- Includes 10K passages extracted from novels, with an average of 4.6 sentences as context, and 1 target sentence the last word of which is to be predicted

Model	Test Loss
LSTM	11.38
LMU	10.88
BMU	11.44
Linear LMU	11.15

Conclusions

- Linear LMU outperforms LMU - Increasing the complexity of the architecture further may help improve results
- Similar performance by BMU and LMU - not clear if LMU is the best extension of Pade approximants
- No one-fit-all model - Performance varies across different tasks

References

1. Legendre Memory Units: Continuous-Time Representation in RNNs (https://papers.nips.cc/paper/2019/file/952285b9b7e7a1be5aa7849f32ffff05-Paper.pdf)
2. Dynamical Systems in Spiking Neuromorphic Hardware (https://uwspace.uwaterloo.ca/bitstream/handle/10012/14625/Voelker_Aaron.pdf?sequence=4&isAllowed=y)
3. Legendre–Bernstein basis transformations, Rida T. Farouki, Journal of Computational and Applied Mathematics, Volume 119, Issues 1–2, 2000
4. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling (https://arxiv.org/pdf/1803.01271.pdf)
5. Codes from [4]: https://github.com/locuslab/TCN
6. Codes from [1]: https://github.com/nengo/keras-lmu

Appendix

$$M_{jk} = \frac{1}{\binom{n}{j}} \sum_{i=\max(0,j+k-n)}^{\min(j,k)} (-1)^{k+i} \binom{k}{i} \binom{k}{i} \binom{n-k}{j-i}$$

$$M_{jk}^{-1} = \frac{2j+1}{n+j+1} \binom{n}{k} \sum_{i=0}^{j} (-1)^{j+i} \frac{\binom{j}{i}\binom{j}{i}}{\binom{n+j}{k+i}}$$