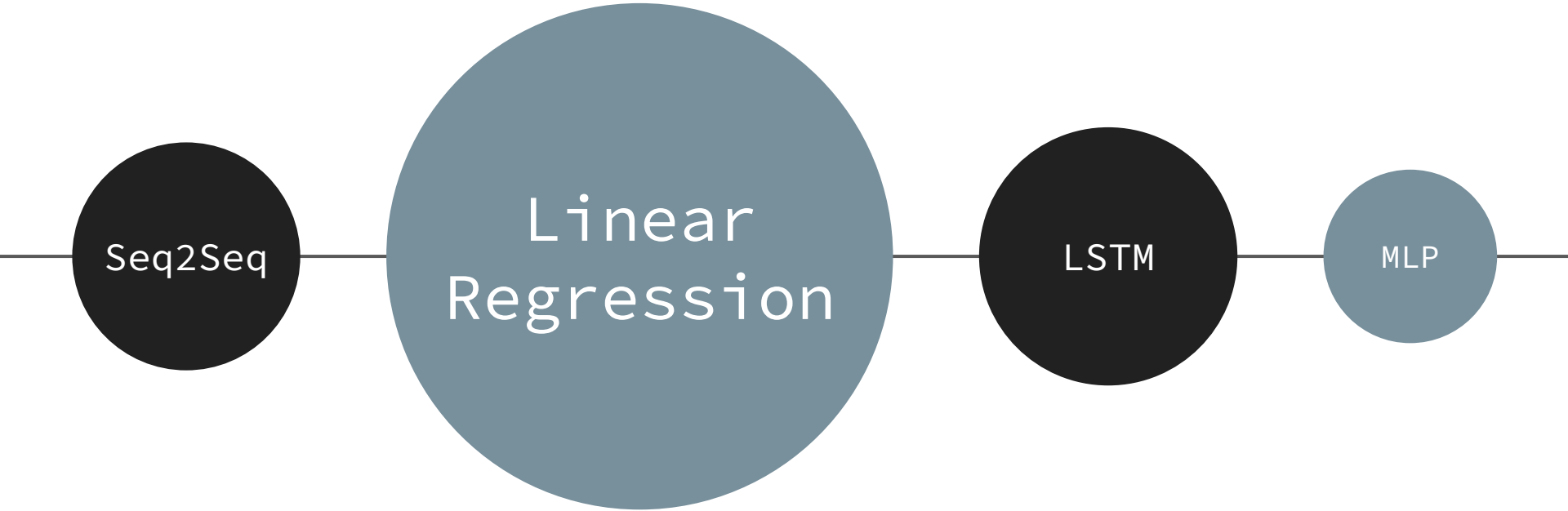
CSE 151- Deep Learning Kaggle Competition

Alex Makhratchev and Benjamin Becze

Summary

- How did you solve the problem:
  - Linear Regression
- What have you learned
  - Simplicity is key

Key Words



Introduction

Team Introduction

- Alex Makhratchev
  - 3rd year Data Science, Business minor
- Benjamin Becze
  - 4th year Data Science Major, Cognitive Science Minor.

UC San Diego

HALICIOĞLU DATA SCIENCE INSTITUTE

Methodology

Data Processing

- Linear transformation of X from size (50,2) to (60,2)
  - Used in Seq2Seq and Linear Regression model
- Sequence making for LSTM
  - Concatenated X and Y created sequences of fixed size. Next coordinate pair of the sequence was the label
- Scaling for all models
  - Scaled all of our inputs by a value in order to prevent gradient explosion

Standardization:

$$x' = \frac{x - \bar{x}}{\sigma}$$

Mean Normalization:

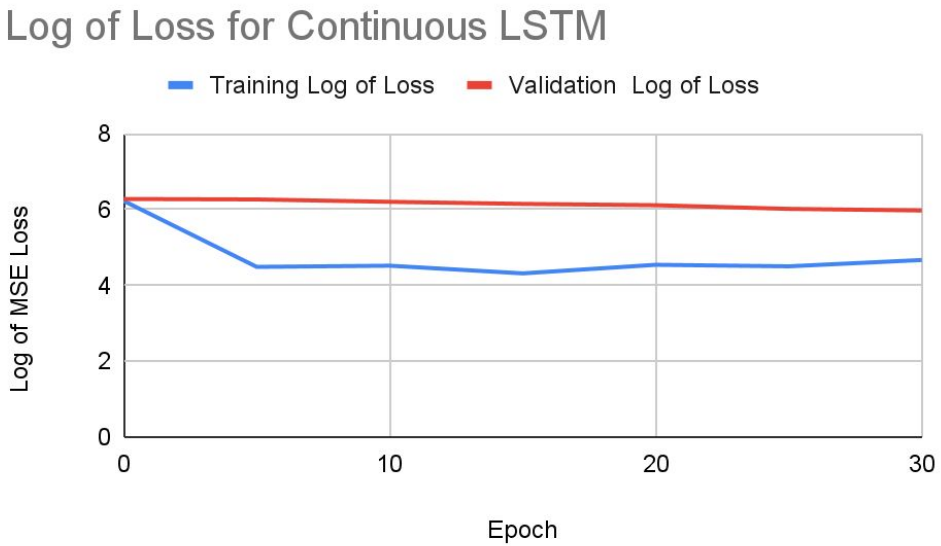
$$x' = \frac{x - \bar{x}}{max(x) - min(x)}$$

Min-Max Scaling:

$$x' = \frac{x - min(x)}{max(x) - min(x)}$$

Deep Learning Model - Continuous LSTM

- seqLen = 50
- stepSize = 5
- batch_sz = 64 # batch size
- learning_rate = 0.0001
- loss_function = nn.MSELoss()
- optimizer = torch.optim.Adam(lstm.parameters(), lr=learning_rate)
- Epochs = 150
- Architecture
- LSTM(
- (lstm): LSTM(100, 256, num_layers=2)
- (linear): Linear(in_features=256, out_features=2, bias=True)
- Kaggle test MSE: 1,348,105.59



Engineering Tricks

- Adding more layers and playing around with their sizes
- Trained a model for each city
- Try numerous learning rates to find different optimas

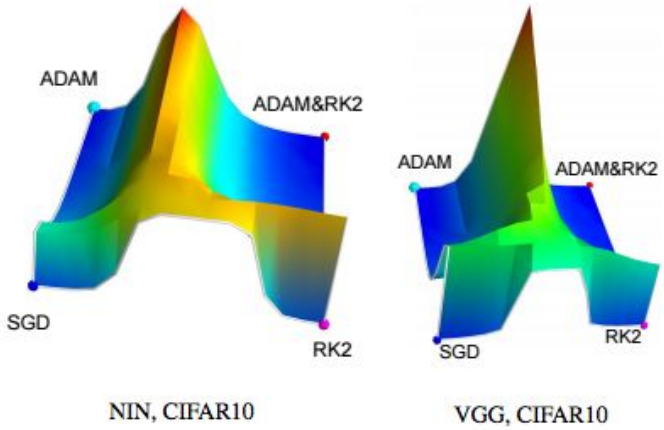


Figure 1. Visualization of the loss surface at weights interpolated between the final points of four different algorithms from the same initialization.

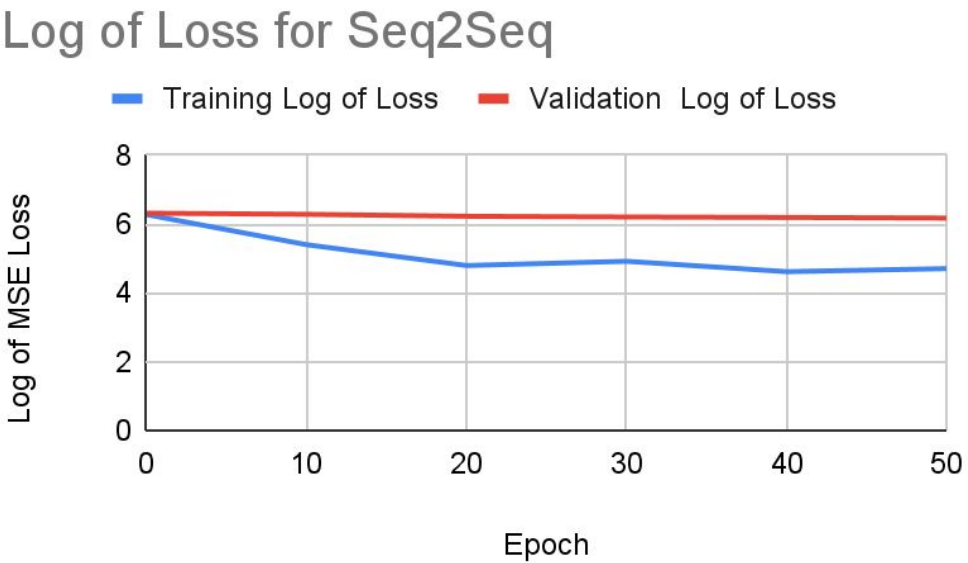
Experiments

Linear Regression

- Used Scikit Learn linear regression model for our baseline
- Gave us best performance overall
- Input was linear transformation of X from length 50 to 60 and output was of length 60
- Kaggle MSE: 911.42

Seq2Seq

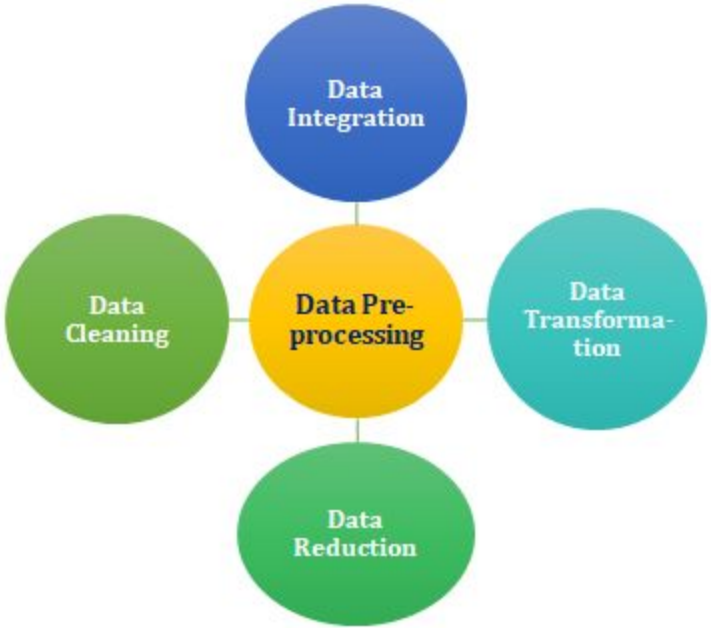
- Seq2Seq model was the second LSTM model we tried
- Did not work well as the other models
- Input was linear transformation of X from length 50 to 60 and output
- Architecture:
    - EncoderRNN(input size=2, hidden size=16, num layers=2)
    - DecoderRNN(hidden size=16, output size=2, num layers=2)was of length 60
- Kaggle MSE: 4M+



Discussion

What we have learned

- Complexity does not always benefit results
- Data preprocessing is crucial
- Models take longer to train than one thinks
- There are countless techniques to improve models



Future Work

- We hope our findings can be incorporated into motion prediction algorithms future algorithms
- Work on a similar task, but use other data such as Lidar sensors or cameras