IE4424 Machine Learning Design and Application

Time Series Prediction Using Long Short-Term Memory (LSTM)

Week 6 Design Lab Briefing

Dr Yap Kim Hui

Email: ekhyap@ntu.edu.sg

Objective

- Study the concepts and models for time series prediction using RNN/LSTM networks.
- Exposure to Pytorch Deep Learning framework for practical applications.

Introduction

- Time Series Prediction: predict the future values via learning the trend in the past values.
- Focus on high-level understanding first, rather than detailed code syntax.

Data Preprocessing (1)

- Load Dataset:

```
import seaborn as sns

flights = sns.load_dataset('flights')
```

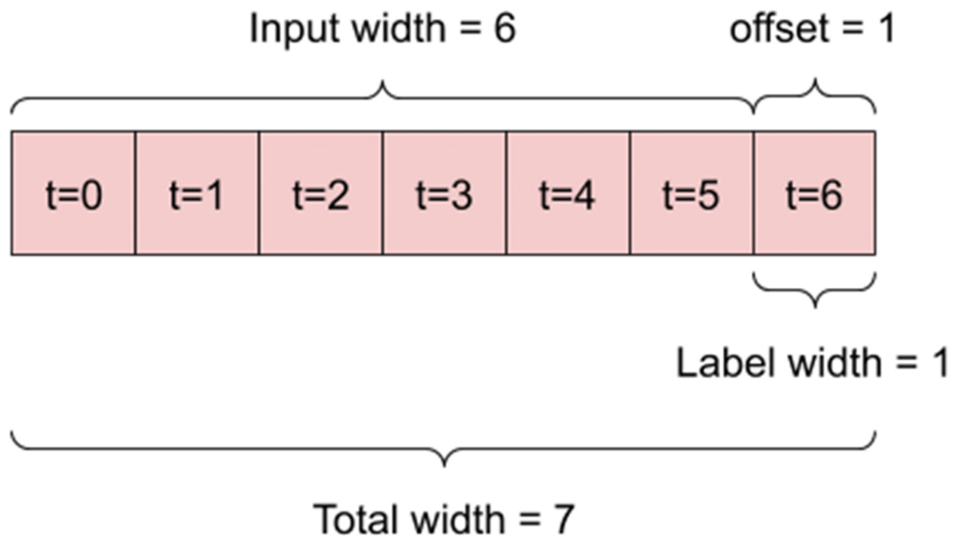
- Normalize and convert to tensor:

```
from sklearn.preprocessing import StandardScaler

passengers = flights['passengers'].values.astype(float)
scaler = StandardScaler()
normalized_data = scaler.fit_transform(passengers.reshape(-1,1))
```

Data Preprocessing (2)

- Data Windowing:
  - A model that makes a prediction of 1 timestep into the future, given a history window of 6 timesteps would look like this:



- Construct Dataloader:

```
train_set = CustomDataset(train_x, train_y)
train_loader = DataLoader(train_set, batch_size=24)
```

Pytorch LSTM Implementation

```python
import torch.nn as nn
import torch.nn.functional as F

class TimeSeriesPredictor(nn.Module):
    def __init__(self, input_dim=1, hidden_dim=200, output_dim=1):
        super(TimeSeriesPredictor,self).__init__()

        self.hidden_dim = hidden_dim

        self.lstm = nn.LSTM(input_dim, self.hidden_dim,
                            batch_first=True)

        self.fc1 = nn.Linear(self.hidden_dim, 100)
        self.fc2 = nn.Linear(100, output_dim)

    def forward(self, input_seq):

        lstm_out, (hn, cn) = self.lstm(input_seq)
        predictions = F.relu(self.fc1(hn.view(-1, self.hidden_dim)))
        predictions = self.fc2(predictions)
        return predictions

model = TimeSeriesPredictor()
```

LSTM Layer

Fully-connected Layer

Layer Definition

Feed Forward Logic

Loss Calculation and Back Propagation

```
# define loss function and optimizer
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.0001)
```

```
# in your training loop
optimizer.zero_grad()
y_pred = model(seq)
loss = criterion(y_pred, labels)
loss.backward()
optimizer.step()
```

Lab Instructions

1. Read the instructions and complete the exercises in Time_Series_Prediction.ipynb and the optional Time_Series_Prediction_Optional.ipynb.
2. Get the answer sheet from lab staff. Follow the instructions and answer the questions in the answer sheet.
3. Write your full name and matriculation no clearly on the answer sheet.
4. Submit the completed answer sheet to the TA at the end of lab.
5. Around last 30 min of the lab, you will be given a few short questions to answer individually. This last part will be closed book and centred on the lecture note and conducted lab.

References

- Stanford Lecture Notes, CS231n.
- Pytorch Official Tutorial.