File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all Copy to Drive

RAM Disk

Table of contents

- Welcome to Colab!
- Getting started
- Data science
- Machine learning
- More Resources
- **Featured examples**

+ Section

[1] ✓ 15s

```
import torch
import torch.nn as nn
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from torch.utils.data import DataLoader, TensorDataset

# ---------------------------------------
# 1. Load Dataset
# ---------------------------------------
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/airline-passengers.csv"
df = pd.read_csv(url, usecols=[1])
plt.plot(df.values)
plt.title("Monthly Air Passengers")
plt.xlabel("Time")
plt.ylabel("Passengers")
plt.show()

# Normalize data to [0, 1]
scaler = MinMaxScaler()
data = scaler.fit_transform(df.values.astype(float))

# ---------------------------------------
# 2. Prepare Sequences
# ---------------------------------------
seq_length = 12  # use past 12 months to predict next month
X, y = [], []
for i in range(len(data) - seq_length):
    x.append(data[i:i + seq_length])
    y.append(data[i + seq_length])

X = np.array(X)
y = np.array(y)

X = torch.tensor(X, dtype=torch.float32)
y = torch.tensor(y, dtype=torch.float32)
```

Variables Terminal

✓ 6:43 PM Python 3

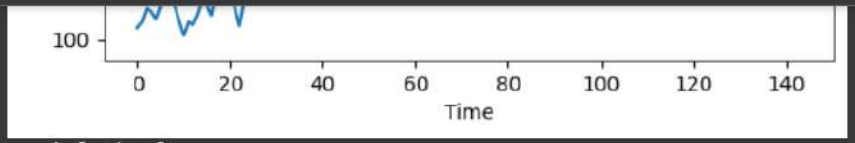
File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all Copy to Drive

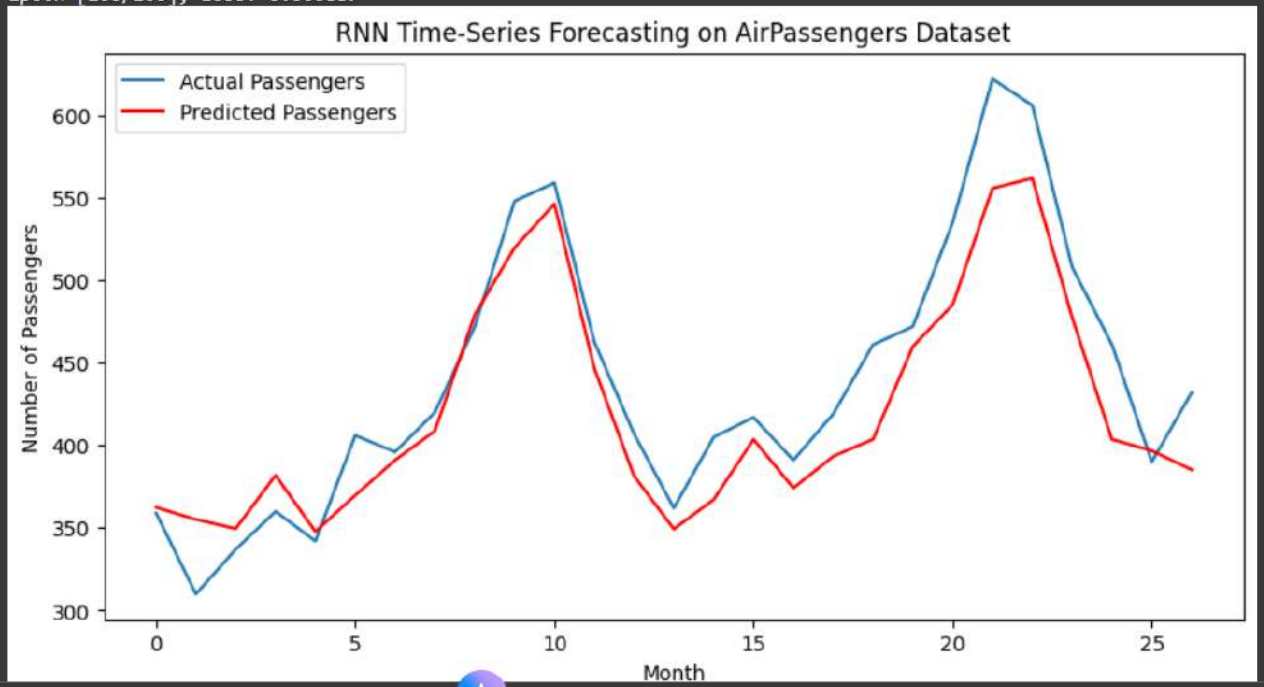
RAM Disk

Table of contents

- Welcome to Colab!
- Getting started
- Data science
- Machine learning
- More Resources
  - Featured examples
- + Section




```
Epoch [10/100], Loss: 0.003331
Epoch [20/100], Loss: 0.002175
Epoch [30/100], Loss: 0.001934
Epoch [40/100], Loss: 0.001115
Epoch [50/100], Loss: 0.002183
Epoch [60/100], Loss: 0.002618
Epoch [70/100], Loss: 0.001357
Epoch [80/100], Loss: 0.001429
Epoch [90/100], Loss: 0.001194
Epoch [100/100], Loss: 0.000887
```



Variables Terminal

6:43 PM Python 3

[2]
✓ 2s

```
# ================================

import torch
import torch.nn as nn
import numpy as np
import matplotlib.pyplot as plt
from torch.utils.data import DataLoader, TensorDataset

# ------------------------------------
# 1. Create Sequential Dataset (Sine Wave)
# ------------------------------------
x = np.linspace(0, 100, 1000)
data = np.sin(x)

# Prepare input sequences and labels
seq_length = 20
X, y = [], []

for i in range(len(data) - seq_length):
    X.append(data[i:i + seq_length])
    y.append(data[i + seq_length])

X = np.array(X)
y = np.array(y)

X = torch.tensor(X, dtype=torch.float32).unsqueeze(-1)  # (samples, seq_len, 1)
y = torch.tensor(y, dtype=torch.float32).unsqueeze(-1)  # (samples, 1)

dataset = TensorDataset(X, y)
train_loader = DataLoader(dataset, batch_size=32, shuffle=True)

# ------------------------------------
# 2. Define the RNN Model
# ------------------------------------
class RNNModel(nn.Module):
    def __init__(self, input_size=1, hidden_size=50, num_layers=1, output_size=1):
        super(RNNModel, self).__init__()
        self.hidden_size = hidden_size
```

File Edit View Insert Runtime Tools Help

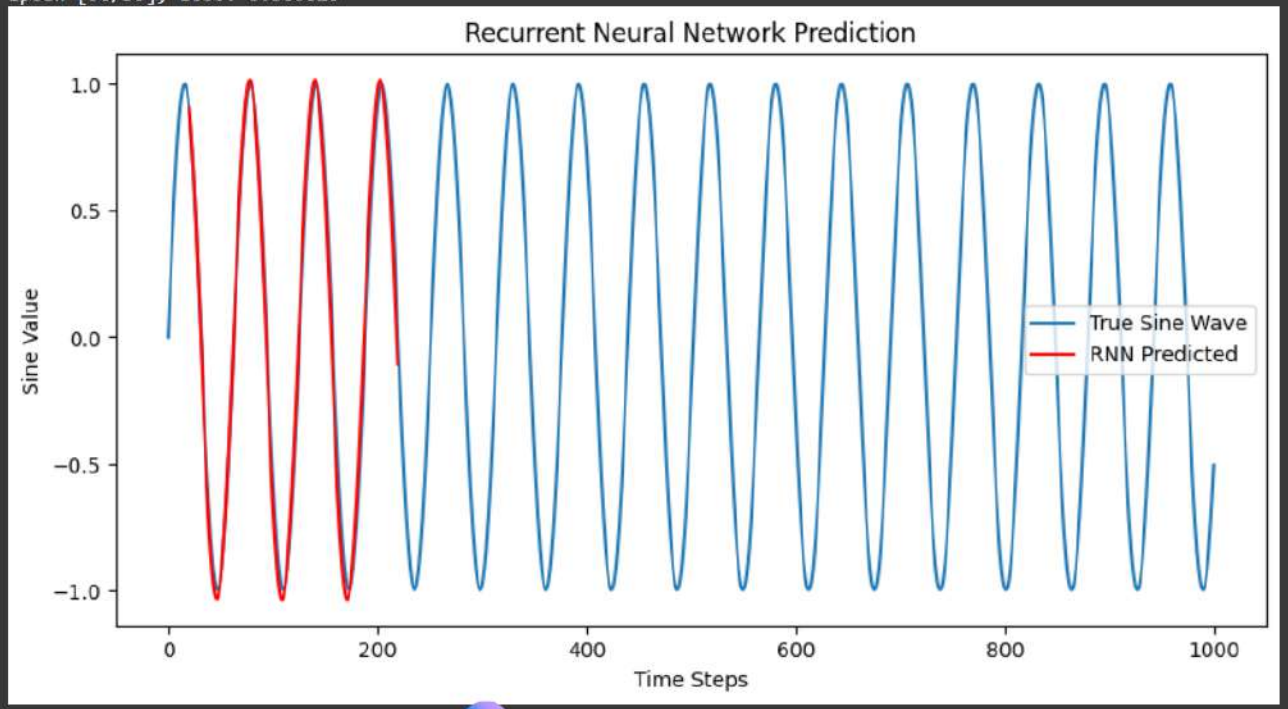
Commands + Code + Text Run all Copy to Drive

RAM Disk

Table of contents

- Welcome to Colab!
- Getting started
- Data science
- Machine learning
- More Resources
- **Featured examples**
- + Section

```
Epoch [17/30], Loss: 0.000025
Epoch [18/30], Loss: 0.000068
Epoch [19/30], Loss: 0.000048
Epoch [20/30], Loss: 0.000030
Epoch [21/30], Loss: 0.000008
Epoch [22/30], Loss: 0.000005
Epoch [23/30], Loss: 0.000003
Epoch [24/30], Loss: 0.000007
Epoch [25/30], Loss: 0.000017
Epoch [26/30], Loss: 0.000072
Epoch [27/30], Loss: 0.000185
Epoch [28/30], Loss: 0.000024
Epoch [29/30], Loss: 0.000029
Epoch [30/30], Loss: 0.000020
```



Variables Terminal

6:44 PM Python 3

g) Train the model using training data for no. of. epochs.

h) Make prediction on input data.

a) inverse transform prediction.

b) visualize actual vs predicted values

c) evaluate Model performance.

Observation:-

1.) The LSTM model learned the Sequence pattern the time data effectively after Several epochs on data.

2.) During Initial epochs, the model showed higher error due to limited learning.

3.) overfitting was not observed in this case due to a simple dataset.

4.) visualization of predicted vs actual value showed that the model was making accurate.

Result:-

The RNN model using LSTM Successfully.

Ex: no: 9 Build a Recurrent neural network.

Aim :-

To build and implement a Recurrent neural network using LSTM on time Series.

Algorithm :-

*LSTM is a type of RNN Capable of learning long term dependencies.

*It is effective in handling time Series and Sequential data by maintaining memory over long Sequence.

Pseudo Code :-

Import necessary libraries. or generate time Series data.

a) Normalize the data using min max Scalar.

b) Convert data into input output Sequence using Sliding windows.

c) prepare data for RNN

d) Reshape input data into [sample, time steps, features]

e) build the RNN model

f) Add an LSTM layer with units.

6) Train the model on training data.

7.) predict on test data.

8.) Inverse transform the predicted values.

9.) Evaluate the model.

a) plot actual vs predicted value.

b) Calculate RMSE or MAE

Observation:-

1.) Initial performance:-

When the model was trained with a small no of epochs, the LSTM show basic learning capability but had relatively higher error values.

2.) Improved accuracy with more training increasing the no. of epochs to 100 and 200 significatly improve the model's performance.

3.) Model Stability:-

Overall, the LSTM model was stable and effective for the prediction.

RESULT:-

The LSTM model successfully learned patterns in time series.

Ex:8 Using LSTM algorithm

Aim:-

To implement the LSTM algorithms

Algorithm:-

* LSTM is a type of recurrent neural network.
* capable of learning Long term dependencies, especially in sequential data.
* It uses gates to control the flow of information.

Pseudo Code:-

1.) Load the dataset
2.) preprocess the dataset:
    a) Normalize the value.
    b) Convert the data into Sequences.
3.) Split the dataset into training and testing sets.
4.) Define the LSTM model.
    a) Input layer.
    b) LSTM layer
    c) Dense layer
5.) Compile the model
    a) Loss function, mean Squared error
    b) Optimizer