**Experiment: -7**

**AIM:** Sequence Modelling using RNN

Code:

import numpy as np

class RNN:

def \_\_init\_\_(self, input\_size, hidden\_size, output\_size):

self.hidden\_size = hidden\_size

# Weight matrices for input to hidden layer

self.Wxh = np.random.randn(hidden\_size, input\_size) \* 0.01

# Weight matrices for hidden to hidden layer

self.Whh = np.random.randn(hidden\_size, hidden\_size) \* 0.01

# Weight matrices for hidden to output layer

self.Why = np.random.randn(output\_size, hidden\_size) \* 0.01

# Biases

self.bh = np.zeros((hidden\_size, 1))

self.by = np.zeros((output\_size, 1))

def forward(self, inputs):

h = np.zeros((self.hidden\_size, 1))

self.last\_inputs = inputs

self.last\_hs = {0: h}

for i, x in enumerate(inputs):

x = x.reshape(-1, 1)

h = np.tanh(self.Wxh @ x + self.Whh @ h + self.bh)

self.last\_hs[i + 1] = h

y = self.Why @ h + self.by

return y, h

def backward(self, d\_y, learning\_rate=0.001):

n = len(self.last\_inputs)

d\_Wxh, d\_Whh, d\_Why = np.zeros\_like(self.Wxh), np.zeros\_like(self.Whh), np.zeros\_like(self.Why)

d\_bh, d\_by = np.zeros\_like(self.bh), np.zeros\_like(self.by)

d\_h = np.zeros\_like(self.last\_hs[0])

for t in reversed(range(n)):

d\_y\_hat = d\_y

d\_Why += d\_y\_hat @ self.last\_hs[t + 1].T

d\_by += d\_y\_hat

d\_h += self.Why.T @ d\_y\_hat \* (1 - self.last\_hs[t + 1] \*\* 2)

d\_bh += d\_h

d\_Wxh += d\_h @ self.last\_inputs[t].reshape(1, -1)

d\_Whh += d\_h @ self.last\_hs[t].T

d\_h = self.Whh.T @ d\_h

for d\_param in [d\_Wxh, d\_Whh, d\_Why, d\_bh, d\_by]:

np.clip(d\_param, -1, 1, out=d\_param)

self.Wxh -= learning\_rate \* d\_Wxh

self.Whh -= learning\_rate \* d\_Whh

self.Why -= learning\_rate \* d\_Why

self.bh -= learning\_rate \* d\_bh

self.by -= learning\_rate \* d\_by

def train(self, inputs, targets, learning\_rate=0.001):

out, \_ = self.forward(inputs)

loss = (out - targets) \*\* 2 / 2

d\_out = out - targets

self.backward(d\_out, learning\_rate)

return loss

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

# Example input (sequence of length 3 with each element having 2 features)

inputs = [np.array([1, 2]), np.array([2, 3]), np.array([3, 4])]

targets = np.array([[0.5], [1.0], [1.5]]) # Example targets

rnn = RNN(input\_size=2, hidden\_size=5, output\_size=1)

# Train for 100 epochs

for epoch in range(100):

loss = 0

for i in range(len(inputs)):

loss += rnn.train([inputs[i]], targets[i])

print(f'Epoch {epoch}, Loss: {loss}')