RNN

```
import torch
import torch.optim as optim
import numpy as np
```

Create Data

```
char_dic = ['h', 'i', 'e', 'l', 'o']
```

```
x_data = [[0, 1, 0, 2, 3, 3]] # hihell
x_one_hot = [[[1, 0, 0, 0, 0],
              [0, 1, 0, 0, 0],
              [1, 0, 0, 0, 0],
              [0, 0, 1, 0, 0],
              [0, 0, 0, 1, 0],
              [0, 0, 0, 1, 0]]]
y_data = [[1, 0, 2, 3, 3, 4]] # ihello
```

```
X = torch.FloatTensor(x_one_hot)
Y = torch.LongTensor(y_data)
```

```
X.shape
```

```
torch.Size([1, 6, 5])
```

Rnn model

New modules

torch.nn.RNN(input_size, hidden_size, num_layers, batch_first=False, ...)

> batch_first=True >> RNN input: (batch size, sequence length, input feature)
> batch_first=False >> RNN input: (sequence length, batch size, input feature)

```
input_size = len(char_dic)
hidden_size = len(char_dic)
```

```
rnn = torch.nn.RNN(input_size, hidden_size, batch_first=True)
```

Training

```
learning_rate = 0.1
```

```
criterion = torch.nn.CrossEntropyLoss()
optimizer = optim.Adam(rnn.parameters(), learning_rate)
```

```
# start training
for i in range(10):
    optimizer.zero_grad()
    outputs, _status = rnn(X)
    loss = criterion(outputs.view(-1, input_size), Y.view(-1))
    loss.backward()
    optimizer.step()

    result = outputs.data.numpy().argmax(axis=2)
    result_str = ''.join([char_dic[c] for c in np.squeeze(result)])
    print(i, "loss: ", loss.item(), "prediction: ", result, "true Y: ", y_data
```

```
0 loss:  0.46169352531433105 prediction:  [[1 0 2 3 3 4]] true Y:  [[1, 0, 2, 3, 3, 4]] predi
1 loss:  0.4615854024887085 prediction:  [[1 0 2 3 3 4]] true Y:  [[1, 0, 2, 3, 3, 4]] predic
2 loss:  0.4614787995815277 prediction:  [[1 0 2 3 3 4]] true Y:  [[1, 0, 2, 3, 3, 4]] predic
3 loss:  0.46137353777885437 prediction:  [[1 0 2 3 3 4]] true Y:  [[1, 0, 2, 3, 3, 4]] predi
4 loss:  0.4612683355808258 prediction:  [[1 0 2 3 3 4]] true Y:  [[1, 0, 2, 3, 3, 4]] predic
5 loss:  0.4611652195453644 prediction:  [[1 0 2 3 3 4]] true Y:  [[1, 0, 2, 3, 3, 4]] predic
6 loss:  0.46106210350990295 prediction:  [[1 0 2 3 3 4]] true Y:  [[1, 0, 2, 3, 3, 4]] predi
7 loss:  0.46096038818359375 prediction:  [[1 0 2 3 3 4]] true Y:  [[1, 0, 2, 3, 3, 4]] predi
8 loss:  0.4608597457408905 prediction:  [[1 0 2 3 3 4]] true Y:  [[1, 0, 2, 3, 3, 4]] predic
9 loss:  0.46075940132141113 prediction:  [[1 0 2 3 3 4]] true Y:  [[1, 0, 2, 3, 3, 4]] predi
```