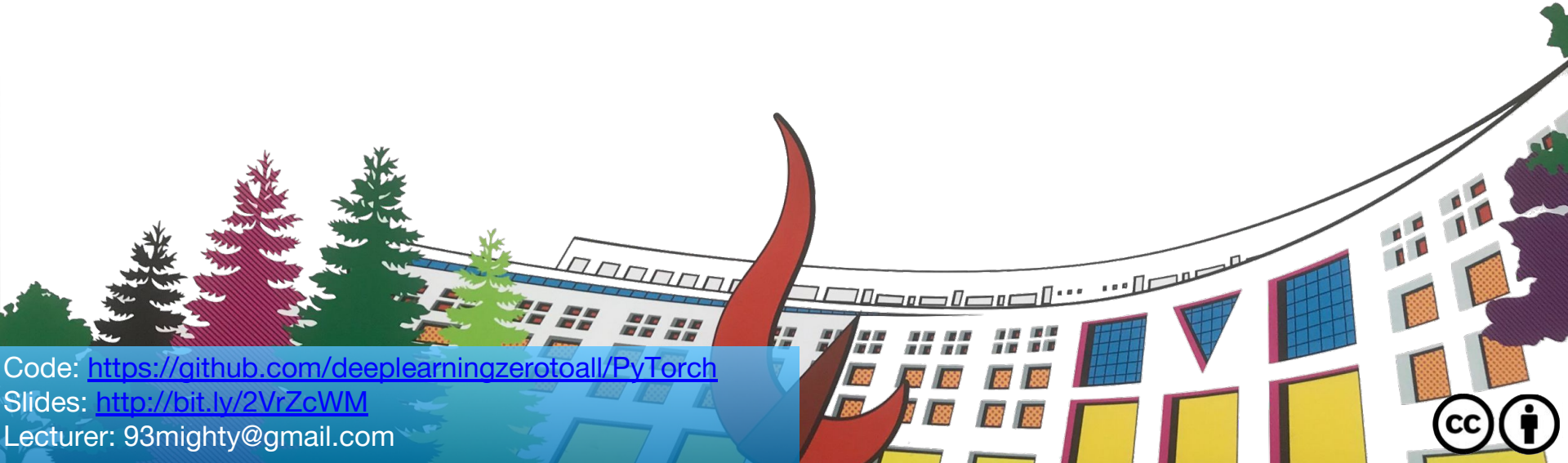
ML/DL for Everyone Season2

with PYTORCH

RNN - longseq



Code: https://github.com/deeplearningzerotoall/PyTorch
Slides: http://bit.ly/2VrZcWM
Lecturer: 93mighty@gmail.com

'longseq' example

- Longseq introduction
- Making sequence dataset from long sentence
- Adding FC layer and stacking RNN
- Code run through

longseq

- We want to use longer dataset
- But we want to train in bigger chunks
- How can we create fixed size sequence dataset from long sentence?

Making sequence dataset from long sentence

```
sentence = ("if you want to build a ship, don't drum up people together to "
            "collect wood and don't assign them tasks and work, but rather "
            "teach them to long for the endless immensity of the sea.")
```

```
"if you wan" -> "f you want"

"f you want" -> " you want "

" you want " -> "you want t"

"you want t" -> "ou want to"

"ou want to" -> "u want to "
```

...

Making sequence dataset from long sentence (code)

```
# data setting
x_data = []
y_data = []

for i in range(0, len(sentence) - sequence_length):
    x_str = sentence[i:i + sequence_length]
    y_str = sentence[i + 1: i + sequence_length + 1]
    print(i, x_str, '->', y_str)
    x_data.append([char_dic[c] for c in x_str])  # x str to index
    y_data.append([char_dic[c] for c in y_str])  # y str to index

x_one_hot = [np.eye(dic_size)[x] for x in x_data]

# transform as torch tensor variable
X = torch.FloatTensor(x_one_hot)
Y = torch.LongTensor(y_data)
```

```
"if you wan" -> "f you want"

"f you want" -> " you want "

" you want " -> "you want t"

"you want t" -> "ou want to"

"ou want to" -> "u want to "
```

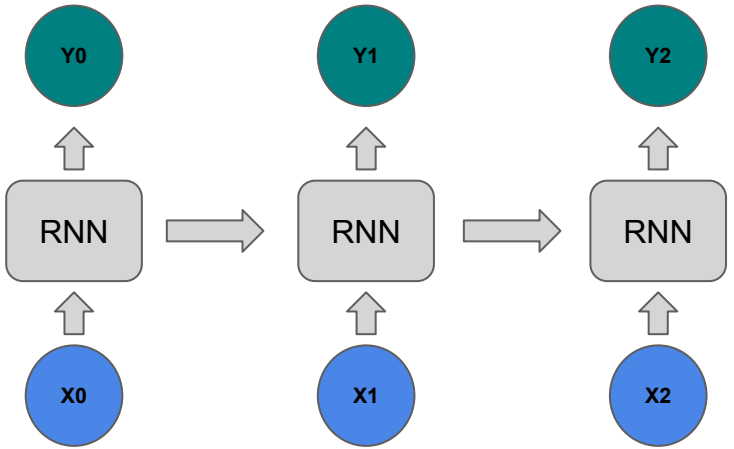
...

Adding FC layer and stacking RNN

```
# declare RNN + FC
class Net(torch.nn.Module):
    def __init__(self, input_dim, hidden_dim, layers):
        super(Net, self).__init__()
        self.rnn = torch.nn.RNN(input_dim, hidden_dim, num_layers=layers,
batch_first=True)
        self.fc = torch.nn.Linear(hidden_dim, hidden_dim, bias=True)


    def forward(self, x):
        x, _status = self.rnn(x)
        x = self.fc(x)
        return x


net = Net(dic_size, hidden_size, 2)
```



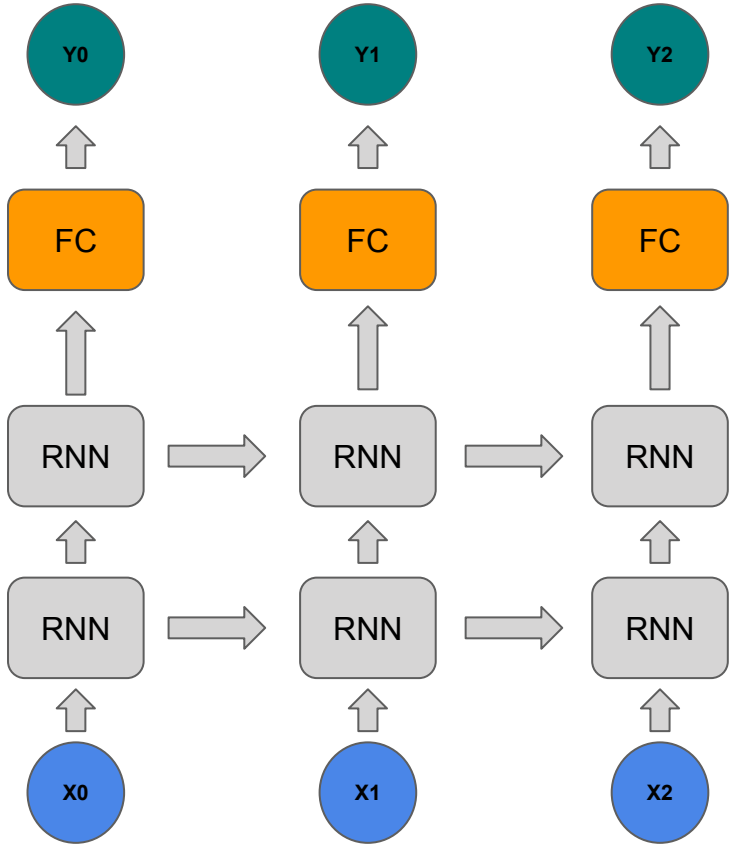
Vanilla RNN

Adding FC layer and stacking RNN

```
# declare RNN + FC
class Net(torch.nn.Module):
    def __init__(self, input_dim, hidden_dim, layers):
        super(Net, self).__init__()
        self.rnn = torch.nn.RNN(input_dim, hidden_dim, num_layers=layers,
batch_first=True)
        self.fc = torch.nn.Linear(hidden_dim, hidden_dim, bias=True)


    def forward(self, x):
        x, _status = self.rnn(x)
        x = self.fc(x)
        return x


net = Net(dic_size, hidden_size, 2)
```



Code run through

```
# loss & optimizer setting
criterion = torch.nn.CrossEntropyLoss()
optimizer = optim.Adam(net.parameters(), learning_rate)

# start training
for i in range(100):
    optimizer.zero_grad()
    outputs = net(X)
    loss = criterion(outputs.view(-1, dic_size), Y.view(-1))
    loss.backward()
    optimizer.step()

    results = outputs.argmax(dim=2)
    predict_str = ""
    for j, result in enumerate(results):
        print(i, j, ''.join([char_set[t] for t in result]), loss.item())
        if j == 0:
            predict_str += ''.join([char_set[t] for t in result])
        else:
            predict_str += char_set[result[-1]]
```

What's Next?

- Time series data prediction