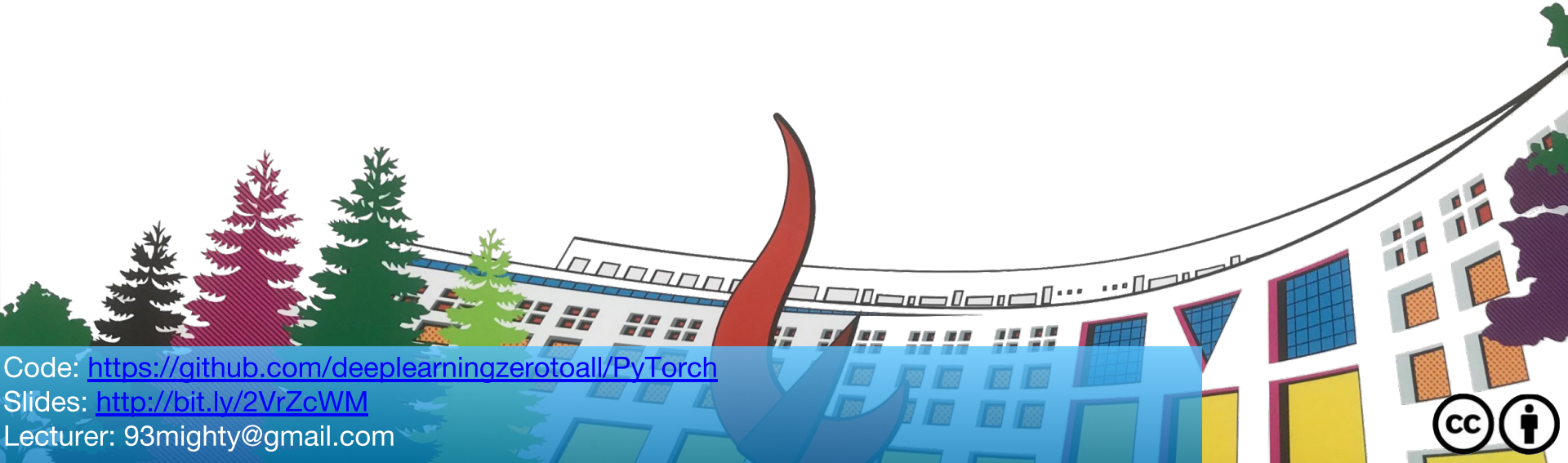
ML/DL for Everyone Season2

with PYTORCH

RNN - hihello / charseq



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

'Hihello' example

- 'Hihello' problem
- Data setting
  - One hot encoding
- Cross entropy loss
- Code run through

'hihello' problem

- 'h', 'i', 'h', 'e', 'l', 'l', 'o'
- We will predict the next character!
- How can we represent characters in PyTorch?

How can we represent characters?

- We can represent them by index
  - 'h' -> 0
  - 'i' -> 1
  - 'e' -> 2
  - 'l' -> 3
  - 'o' -> 4

```
# list of available characters
char_set = ['h', 'i', 'e', 'l', 'o']
```

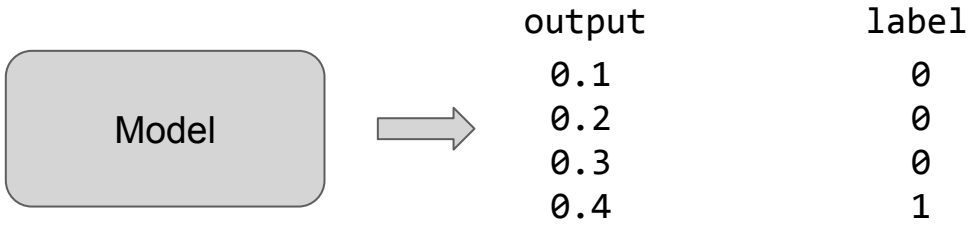
One-hot encoding

- We need to encode using one-hot encoding!

```
# list of available characters
char_set = ['h', 'i', 'e', 'l', 'o']
x_data = [[0, 1, 0, 2, 3, 3]]
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]]
```

Cross Entropy Loss

- Loss for categorical output (usually interpreted as probability)




```
# loss & optimizer setting
criterion = torch.nn.CrossEntropyLoss()
...
loss = criterion(outputs.view(-1, input_size), Y.view(-1))
```

Code run through (hihello)

```
char_set = ['h', 'i', 'e', 'l', 'o']
# hyper parameters
input_size = len(char_set)
hidden_size = len(char_set)
learning_rate = 0.1
# data setting
x_data = [[0, 1, 0, 2, 3, 3]]
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]]
```

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

Code run through (charseq)

```
sample = " if you want you"
# make dictionary
char_set = list(set(sample))
char_dic = {c: i for i, c in enumerate(char_set)}

# hyper parameters
dic_size = len(char_dic)
hidden_size = len(char_dic)
learning_rate = 0.1

# data setting
sample_idx = [char_dic[c] for c in sample]
x_data = [sample_idx[:-1]]
x_one_hot = [np.eye(dic_size)[x] for x in x_data]
y_data = [sample_idx[1:]]
```

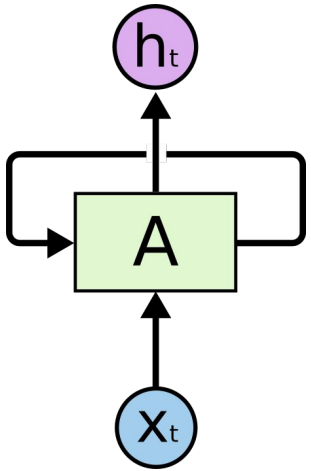
```
# transform as torch tensor variable
X = torch.FloatTensor(x_one_hot)
Y = torch.LongTensor(y_data)
```

Code run through

```
# declare RNN
rnn = torch.nn.RNN(input_size, hidden_size, batch_first=True)  # batch_first guarantees the order of output = (B, S, F)

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

# start training
for i in range(100):
    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_set[c] for c in np.squeeze(result)])
    print(i, "loss: ", loss.item(), "prediction: ", result, "true Y: ", y_data, "prediction str: ", result_str)
```



What's Next?

- More complicated examples
  - Longer character sequence
  - Seq2seq (used in machine translation)