MNIST in PyTMVA

Seungmok Lee 2020.01.29



Deep Learning Study

- We had one week break. (yesterday)
- I'm trying to implement existing DL codes in TMVA toolkit, so that we can check the consistency.
- First, I tried MNIST problem.

MNIST Dataset

- It is a well-known data set in machine learning.
- Consisted of 60K handwritten digits images.
- Basic image recognition example.

0	0	0	0	0	0	O	0
1	l	1	١	ſ	1	/	(
2	ລ	2	ス	ð	J	2	2
3	3	3	3	3	3	3	3
4	4	٤	\mathcal{L}	4	4	Ч	4
5	5	5	5	5	5	5	Б
6	G	6	6	6	6	6	6
¥	7	ッ	7	7	7	ч	7
8	T	8	8	8	8	Я	8
9	૧	9	9	9	ዋ	٩	9

MNIST with Keras

- Google (Xavier) serves an example code written with Keras and TensorFlow.
- Easy, light, short, but accuracy ~ 98%.

https://colab.research.google.com/github/AviatorMoser/keras-mnist-tutorial/blob/master/MNIST %20in%20Keras.ipynb#scrollTo=qYruneli1e1w

MNIST Example Code

Model: "sequential_1"

Layer (type)	Output Shape	Param #	
dense_1 (Dense)	(None, 512)	401920	=====
activation_1 (Activatio	n) (None, 512)	0	
dropout_1 (Dropout)	(None, 512)	0	
dense_2 (Dense)	(None, 512)	262656	
activation_2 (Activatio	n) (None, 512)	0	
dropout_2 (Dropout)	(None, 512)	0	
dense_3 (Dense)	(None, 10)	5130	
activation_3 (Activatio	n) (None, 10)	0	
Total params: 669,706			
Trainable params: 669,	706		
Non-trainable params:	0		

2020/1/29

Result Epoch 1/5 60000/60000 [==========] -11s 185us/step - loss: 0.2468 - acc: 0.9262 Epoch 2/5 60000/60000 [========] -10s 172us/step - loss: 0.1010 - acc: 0.9692 Epoch 3/5 60000/60000 [==========] -10s 170us/step - loss: 0.0733 - acc: 0.9771 Epoch 4/5 60000/60000 [==========] -10s 172us/step - loss: 0.0581 - acc: 0.9815 Epoch 5/5 10s 171us/step - loss: 0.0459 - acc: 0.9850

MNIST in PyMVA

5

Implementation in PyMVA

- Converting data with PyROOT
 - MNIST data into ROOT format
- Registering variables with PyMVA
- Building model with Keras
- Training model with PyMVA
 - All the (hyper) parameters are from Xavier's code.

Result in PyMVA

danca L input: InputL	war	inpu		(None, 784)					
dense_1_mpar. mparea	ayer	output:		(None, 784)					
dansa li Dansa	inţ	input:		(None, 784)					
dense_1: Dense	output:		(None, 512)						
- directions. In Andirecti		input:		(None, 512)					
activation_1: Activati	on	output:		(None, 512)					
descent 1. Descent	_ i	nput:	0	None, 512)					
aropout_1: Dropout	6	output:		(None, 512)					
dense 2: Dense	input:		(None, 512)						
	out	output: (1		Jone, 512)					
activation 2. Activati		input		: (None, 512)					
		output:		(None, 512)					
dronout 2: Dronout	Ĺ	input:		(None, 512)					
dropour_2. Dropour	0	output:		(None, 512)					
dansa 2 Dansa		input:		(None, 512)					
dense_5. Dense	out	output:		(None, 10)					
· · · · · · · · · · · · · · · · · · ·									
activation 3: Activat	ion	input:		(None, 10)					
activation_3: Activation		output:		(None, 10)					

Epoch 1/5 59999/59999 [=============] - 5s 77us/step - loss: 0.2470 - accuracy: 0.9255 Epoch 2/5 59999/59999 [=======]] - 4s 73us/step - loss: 0.1009 - accuracy: 0.9685 Epoch 3/5 59999/59999 [=============] - 4s 73us/step - loss: 0.0723 - accuracy: 0.9771 Epoch 4/5 59999/59999 [===========]] - 4s 72us/step - loss: 0.0539 - accuracy: 0.9833 Epoch 5/5 59999/59999 [=============] - 4s 72us/step - loss: 0.0486 - accuracy: 0.9839

MNIST in PyMVA

PyMVA: Conclusion

- My code showed perfect consistency with existing code.
- We can train DNN with ROOT format data file!
- Our codes are available in github.
 - https://github.com/physmlee/DLStudy

TMVA in C++

- TMVA is also available in C++ language.
- However, there are some differences in function option, implementation, grammar, ...
- It is challenging, but seems not impossible! I'll do it until next deep learning study meeting.

Prospect / Plan

- Anyway, we can do DL.
 - in ROOT, at least, in Python.
- Maybe we can try to apply DL on our data soon.
- I'm planning to visit IBS, Daejeon.