PyCon Korea 2019

딥러닝 NLP 손쉽게 따라해보기
- GluonNLP-

MXNet Basic

MXNet /Gluon

DMLC 팀에 의해 개발된 Deep Learning Framework
AWS Deep Learning Framework로 선택됨

AWS, MS 에서 개발한 Deep Learning 표준 Framework

https://news.microsoft.com/2017/10/12/aws-and-microsoft-announce-gluon-making-deep-learning-accessible-to-all-developers/

GluonCV, GluonNLP

GluonCV: a Deep Learning Toolkit for Computer Vision

https://gluon-cv.mxnet.io/

GluonCV provides implementations of state-of-the-art (SOTA) deep learning algorithms in computer vision. It aims to help engineers, researchers, and students quickly prototype products, validate new ideas and learn computer vision.

GluonCV features:

1. training scripts that reproduce SOTA results reported in latest papers,
2. a large set of pre-trained models,
3. carefully designed APIs and easy to understand implementations,
4. community support.

GluonNLP: NLP made easy

https://gluon-nlp.mxnet.io/

Get Started: A Quick Example

Here is a quick example that downloads and creates a word embedding model and then computes the cosine similarity between two words.

(You can click the play button below to run this example.)

MXNet/Gluon 주요 특징

- NDArray 방식 적용
  - Gluon 에서 활용하는 주요 방식
  - 해당 형태로 데이터를 선언하면 별도의 Variable 선언 없이 Network 입/출력이 가능
- Autograd 지원
  - backpropagation 수행 시 gradient 를 자동으로 계산해줌
- Symbolic / Imperative 변환이 용이함
  - Hybrid 함수 사용을 통해 간편하게 변환이능함

NDArray 활용 방안

MXNet 에서 활용하는 데이터 방식

- 주요 특징
  - CPU/GPU 변환이 쉬움
  - 기본적인 문법이 NumPy와 유사하여 NumPy를 활용하는 거와 유사하게 활용이 가능함

Function	Numpy	Gluon	Pytorch*
reshape	np.reshape	nd.reshape	torch.view
Concat Data	np.concatenate((x,y),1)	nd.concat(x,y,dim=1)	torch.cat([x,y],1)
Swap shape	np.swapaxes(x,1,2)	x.swapaxes(1,2)	x.permute(0,2,1)
Clipping	np.clip(x, 0, 1)	nd.clip(x, 0, 1)	x.clamp(0, 1)
Batch matrix product	np.linalg_gemm2(x,y)	nd.linalg_gemm2(x,y)	torch.bmm(x, y)

* 1.0 기준

http://mxnet.incubator.apache.org/api/python/ndarray/ndarray.htm
https://github.com/zackchase/mxnet-the-straight-dope/blob/master/cheatsheets/pytorch_gluon.mdl

NDArray 활용 방안

- GPU / CPU 변환이 용이함
  - context 지정을 통해서 GPU / CPU 변환이 용이함
- NDArray 계산 결과를 numpy 형태로 변환이 용이함

```
import numpy as np

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

x
```

```
array([[1, 2],
       [3, 4]])
```

```
x1 = nd.array(x)

x1
```

```
[[1. 2.]
 [3. 4.]]
<NDArray 2x2 @cpu(0)>
```

```
x2 = x1.copyto(mx.gpu())

x2
```

```
[[1. 2.]
 [3. 4.]]
<NDArray 2x2 @gpu(0)>
```

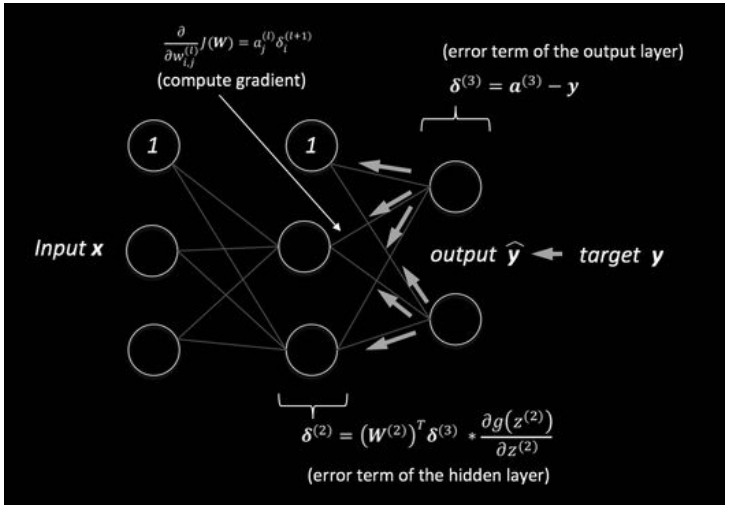
```
x3 = x2.asnumpy()

x3
```

```
array([[1., 2.],
       [3., 4.]], dtype=float32)
```

Automatic Differentiation with autograd

기본적으로 Deep Learning의 학습 과정은 Back Propagation 방식임



네트워크가 복잡해지면 각각 Node에 대한 Gradient를 계산하는 과정이 복잡함

https://sebastianraschka.com/faq/docs/visual-backpropagation.html

Automatic Differentiation with autograd

Gluon 에서는 gradient를 autograd를 통해 자동으로 계산해 줌

$$y = x^2$$
$$z = xy$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

```
In [1]: import mxnet as mx
        from mxnet import gluon, autograd ,nd

/home/ubuntu/anaconda3/lib/python3.6/site-packages/h5py/__ini
ture, it will be treated as `np.float64 == np.dtype(float).ty
  from ._conv import register_converters as _register_convert

In [2]: mx.__version__

'1.3.0'

In [3]: x = nd.array([[1, 2], [3, 4]])
        x.attach_grad()

        with autograd.record():
            y = x * 2
            z = y * x

        z.backward()
        print(x.grad)

[[ 4.  8.]
 [12. 16.]]
<NDArray 2x2 @cpu(0)>
```

https://gluon.mxnet.io/chapter01_crashcourse/autograd.html#

Build Network – 전체적 Network 설계 (1/4)

In [1]:
```
import numpy as np
import mxnet as mx
from mxnet import gluon, autograd ,nd
from mxnet.gluon import nn
import mxnet.ndarray as F

from tqdm import tqdm, trange
```

```
/home/ubuntu/anaconda3/lib/python3.6/site-packages/h5py/__i
ture, it will be treated as `np.float64 == np.dtype(float).
  from ._conv import register_converters as _register_conve
```

In [2]:
```
mx.__version__
```

```
'1.3.0'
```

In [3]:
```
model_ctx = mx.gpu()
```

Build Network – 전체적 Network 설계 (2/4)

In [4]:
```
batch_size = 64
num_inputs = 784
num_outputs = 10
num_examples = 60000
def transform(data, label):
    return data.astype(np.float32)/255, label.astype(np.float32)
train_data = mx.gluon.data.DataLoader(mx.gluon.data.vision.MNIST(train=True, transform=transform),
                                      batch_size, shuffle=True)
test_data = mx.gluon.data.DataLoader(mx.gluon.data.vision.MNIST(train=False, transform=transform),
                                     batch_size, shuffle=False)
```

In [5]:
```
net = gluon.nn.HybridSequential()
with net.name_scope():
    net.add(gluon.nn.Dense(64))
    net.add(gluon.nn.Dense(64))
    net.add(gluon.nn.Dense(10))
```

In [6]:
```
net.initialize(init=mx.init.Normal(),ctx=model_ctx)
```

In [7]:
```
net.hybridize()
```

Build Network – 전체적 Network 설계 (3/4)

In [8]:
```
softmax_cross_entropy = gluon.loss.SoftmaxCrossEntropyLoss()
```

In [9]:
```
trainer = gluon.Trainer(net.collect_params(),'adam',{'learning_rate':0.001})
```

In [10]:
```
def evaluate_accuracy(data_iterator, net):
    acc = mx.metric.Accuracy()
    for i, (data, label) in enumerate(data_iterator):
        data = data.as_in_context(model_ctx).reshape((-1, 784))
        label = label.as_in_context(model_ctx)
        output = net(data)
        predictions = nd.argmax(output, axis=1)
        acc.update(preds=predictions, labels=label)
    return acc.get()[1]
```

Build Network – 전체적 Network 설계 (4/4)

In [11]:

```
epochs = 5

for e in trange(epochs):
    cumulative_loss = 0
    for (data, label) in train_data:
        data = data.as_in_context(model_ctx).reshape((-1, 784))
        label = label.as_in_context(model_ctx)
        with autograd.record():
            output = net(data)
            loss = softmax_cross_entropy(output, label)
        loss.backward()
        trainer.step(data.shape[0])
        cumulative_loss += nd.sum(loss).asscalar()


    test_accuracy = evaluate_accuracy(test_data, net)
    train_accuracy = evaluate_accuracy(train_data, net)
    print("Epoch %s. Loss: %s, Train_acc %s, Test_acc %s" %
          (e, cumulative_loss/num_examples, train_accuracy, test_accuracy))
```

```
20%|██        | 1/5 [00:19<01:18, 19.69s/it]

Epoch 0. Loss: 0.4978684414545695, Train_acc 0.91515, Test_acc 0.9139
```

Build Network – 전체적 Network 설계 (4/4)

In [11]:

```
epochs = 5

for e in trange(epochs):
    cumulative_loss = 0
    for (data, label) in train_data:
        data = data.as_in_context(model_ctx).reshape((-1, 784))
        label = label.as_in_context(model_ctx)
        with autograd.record():
            output = net(data)
            loss = softmax_cross_entropy(output, label)
        loss.backward()
        trainer.step(data.shape[0])
        cumulative_loss += nd.sum(loss).asscalar()


    test_accuracy = evaluate_accuracy(test_data, net)
    train_accuracy = evaluate_accuracy(train_data, net)
    print("Epoch %s. Loss: %s, Train_acc %s, Test_acc %s" %
          (e, cumulative_loss/num_examples, train_accuracy, test_accuracy))
```

```
20%|██        | 1/5 [00:19<01:18, 19.69s/it]

Epoch 0. Loss: 0.4978684414545695, Train_acc 0.91515, Test_acc 0.9139
```

실습

github에 PPT 자료 및 실습 코드 활용

https://github.com/seujung/gluonnlp_tutorial.git

END OF DOCUMENT