Cheatography

Pytorch Cheat Sheet

by Yuxuejie0912 via cheatography.com/123584/cs/23342/

Pytorch

PyTorch Torch Python GPU NumPy

Pytorch

```
import torch
import torch.nn as nn
from torch.autograd import Variable
import torch.nn.functional as F
```

```
model = nn.Sequential(nn.Linear(2, 10), nn.ReLU(), nn.Linear(10, 1), nn.Sigmoid() ) #
```

```
if torch.cuda.is_available():
    model = model.cuda() # GPU cuda
```

```
optimzer = torch.optim.SGD(model.parameters(), lr=0.05) #
loss_func = nn.MSELoss() #
```

```
out = model(x)
loss = loss_func(out, y)
optimzer.zero_grad() #
loss.backward() #
optimzer.step() #
```

Tensor and Variable

Tensor Numpy GPU

Variable Tensor .data, .grad, .grad_fn

```
x_tensor = torch.randn(5, 5)
x_var_regular = Variable(x_tensor, requires_grad=True)
x_var_volatile = Variable(x_tensor, volatile=True)
```

requires_grad volatile Variable inference

torch.nn.functional

torch.nn nn.functional nn.Module

torch.nn.functional def function( ) weight

```
conv = nn.Conv2d(3, 64, 3, 2, 1)
```

```
output = nn.functional.conv2d(inputs, weight, bias, padding=1)
```

torch.nn

torch.nn.ReLU(inplace=False)

torch.nn.BCELoss(weight=None, size_average=True, reduce=None, reduction='mean')

torch.nn.Linear(in_features, out_features, bias=True)

torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True)

```
input = Variable(torch.randn(32, 3, 28, 28))
conv1 = nn.Conv2d(in_channels=3, out_channels=10, kernel_size=3, stride=1, padding=1)
conv2 = nn.Conv2d(in_channels=10, out_channels=128, kernel_size=3, stride=1, padding=1)
```

torch.nn.BatchNorm2d(num_features, eps=1e-05, momentum=0.1, affine=True)

torch.nn.MaxPool2d(kernel_size, stride=None, padding=0, dilation=1, return_indices=False, ceil_mode=False)

torch.nn.AvgPool2d(kernel_size, stride=None, padding=0, ceil_mode=False, count_include_pad=True)

Dropout torch.nn.Dropout(p=0.5, inplace=False)

torch.nn (cont)

torch.nn.Upsample(size=None, scale_factor=None, mode='nearest', align_corners=None)

torch.nn.Module torch.nn.Sequential(* args)

nn.Module

```
model.modules #
model.state_dict() #  module
model.forward() #
model.train() #  train evaluation
model.eval() #  BN  dropout
```

nn.Sequential(* args) modules

class torchvision.transforms #

Compose

```
transforms.Compose([ transforms.CenterCrop(224),
transforms.RandomHorizontalFlip(),
transforms.ToTensor(),
transforms.Normalize(0.5, 0.2) ])
```

```
torch.save({ 'epoch': epoch,
'model_state_dict': model.state_dict(),
'optimizer_state_dict': optimizer.state_dict(),
'loss': loss, ... }, PATH)
```

```
checkpoint = torch.load(PATH)
model.load_state_dict(checkpoint['model_state_dict'])
optimizer.load_state_dict(checkpoint['optimizer_state_dict'])
epoch = checkpoint['epoch'] loss = checkpoint['loss']
```

torch.optim

torch.optim.SGD(model.parameters(), lr = 0.01, momentum=0.9)

torch.optim.RMSprop(model.parameters(), lr=0.01, alpha=0.99, eps=1e-08, weight_decay=0, momentum=0, centered=False)

torch.optim.Adam(model.parameters(), lr=0.001, betas=(0.9, 0.999), eps=1e-08, weight_decay=0, amsgrad=False)

optimizer.zero_grad() # pytorch backward

optimizer.step() #

# pytorch
import torchvision.datasets as datasets
cifar10 = datasets.CIFAR10()
torch.utils.data.DataLoader(cifar10, batch_size=args.batchSize, shuffle=True, num_workers=args.nThreads)

# Dataset
from torch.utils.data.dataset import Dataset


```
import torchvision.models as models
resnet18 = models.resnet18( )
alexnet = models.alexnet(pretrained=True)
vgg19 = models.vgg19(pretrained=True)
```


TensorboardX

```
from tensorboardX import SummaryWriter
with SummaryWriter(comment='LeNet') as w:
    w.add_graph(model, (input, ))
tensorboard --logdir runs
```

pytorchviz github

https://github.com/szagoruyko/pytorchviz