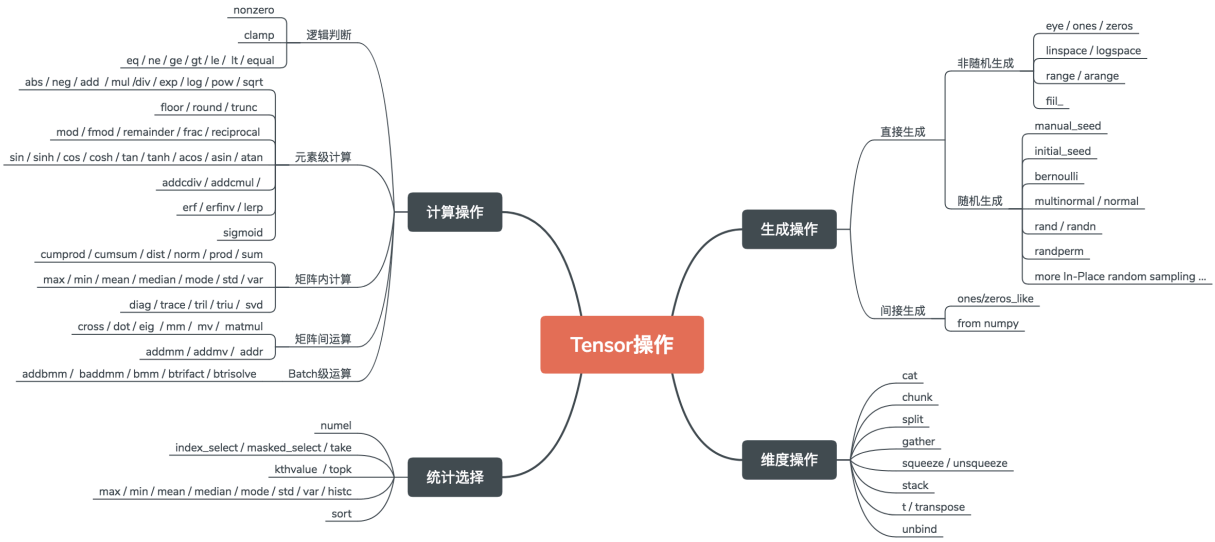
Tensor基本操作

常用库： `torch` `torch.Tensor` `torch.autograd`

> 通读官方基本计算操作：[torch Tensor 操作官方文档]



网络搭建流程

1.参数与日志设置：

常用库： `argparse` `logging`

> Argparse总结： [简书| python argparse用法总结]
>
> Logging教程：[简书|python logging使用教程]

2.数据流：

常用库： `torch.utils.data.Dataset` `torch.utils.data.DataLoader` `torchvision.dataset` `torchvision.transforms`

流程：训练集 —> 测试集

> Dataset类方法：[PyTorch 数据集的读取] 、[莫烦|批训练]
>
> Pytorch utils.data 文档： [官方文档] （DataLoader类参数可以仔细阅读）
>
> torchvision.transforms 变换举例： [torchvision.transforms 中文文档]

3.神经网络：

常用库： `torch.autograd` `torch.nn`

流程：定义网络 —> 定义前馈过程

> 自动求导机制简介： 自动求导机制中文文档 、知乎|自动梯度计算
>
> Autograd.Function类： 自定义Function

4.优化方法：

常用库：`torch.optim`

5.训练过程：

常用库：`torch.nn.Module` `torch.nn.functional` `torch.optim`

流程：模式切换 —> 数据输入 (初始化 / 预训练) —> 梯度重置 —> 损失函数 —> 梯度回馈 —> 参数更新 —> 定期监控

> 预训练：知乎|PyTorch 预训练
>
> 保存提取：莫烦|PyTorch 保存提取
>
> 官方模型保存：推荐的模型保存方法

6.测试过程：

流程： 模式切换 —> 结果输出

X. 加速技巧：

常用库： `multiprocessing` `torch.cuda` `torch.distributed` `torch.nn.parallel`

有趣的实现要点

快速搭建模型

```
model = torch.nn.Sequential(
    torch.nn.Linear(1, 10),
    torch.nn.ReLU(),
    torch.nn.Linear(10, 1)
)
```

关于CUDA控制：

CUDA判断：

```
args.cuda = not args.no_cuda and torch.cuda.is_available()
```

Model CUDA化：

```
if args.cuda:
    model.cuda()
```

Variable CUDA化：

```
if args.cuda:
            data, target = data.cuda(), target.cuda()
```

关于训练与验证模式：

训练模式：

```
# Sets the module in training mode
# Has any effect only on modules such as Dropout or BatchNorm.
model.train()
```

验证/测试模式：

```
# Sets the module in evaluation mode.
# This has any effect only on modules such as Dropout or BatchNorm.
model.eval()
```

关于数据载入：

逐行读入：

```
with open(filename) as f:
    for line in f:
```

数据输入：

```
# generate (i, data) pair
for i, data in enumerate(trainloader, 0):
```

关于报错和断言：

```
assert os.path.isfile(img_path), "Not found image -->{}".format(img_path)
```

torch.arange 和 torch.range 的区别：

```
# torch.range is deprecated in favor of torch.arange and will be removed in 0.3.
# Note that arange generates values in [start; end), not [start; end].

print torch.arange(1,3)
```

```
print torch.range(1,3)

# 1
# 2
# [torch.FloatTensor of size 2]

# 1
# 2
# 3
# [torch.FloatTensor of size 3]
```