pytorch

可以代替numpy的工具，可以在GPU上运行；深度学习平台；

Tensor

GPU版本的numpy.ndarray(),pytorch中所有变量都是tensor格式的。

- tenser的创建：可以由torch.empty(...)/torch.rand(...)/torch.zeros(...,dtype=torch.long)/torch.tensor([...])等创建；支持各种数学运算等
- tensor的形状：输出的一个tensor变量的形状，它的类型是torch.Size，是一个，支持tuple的各种运算；
- 改变tensor的形状：torch.view(...)
- 单变量获取python格式数据：x.iterm()
- Tensor的GPU与CPU的切换

```
# let us run this cell only if CUDA is available
# We will use ``torch.device`` objects to move tensors in and out of GPU
if torch.cuda.is_available():
    device = torch.device("cuda")          # a CUDA device object
    y = torch.ones_like(x, device=device)  # directly create a tensor on GPU
    x = x.to(device)                       # or just use strings ``.to("cuda")``
    z = x + y
    print(z)
    print(z.to("cpu", torch.double))       # ``.to`` can also change dtype together!
```

Tensor与numpy的关系

两者在CPU上是共享内存的，一个改变，另一个会跟着改变；
相互转换：除了charTensor，其他的都可以相互转换。

- torch.tensor->numpy.array：x.numpy()
- numpy.array->torch tensor: b=torch.from_numpy(a)

Tensor和Variable的合并
在0.4版本中，两者合并，说是合并, 其实是按照以前(0.1-0.3版本)的观点是: **Tensor现在默认requires_grad=False的Variable了**. torch.Tensor和torch.autograd.Variable现在其实是同一个类! 没有本质的区别! 所以也就是说, 现在已经没有纯粹的Tensor了, 是个Tensor, 它就支持自动求导! 你现在要不要给Tensor包一下Variable, 都没有任何意义了。例如下面的例子：

```
'''
weight=weight-lr*grad
'''
lr=0.001
for f in net.parameters():
    print(type(f.detach()))
    f.detach().sub_(lr*f.grad.detach())#0.3版本的pytorch，注意这里要取到data，以得到tensor数据；
    # 在0.4版本中，则要用.detach(),这个操作是不需要进行反向求导的，因此要detach
'''
 1).data返回一个新的requires_grad=False的Tensor! 然而新的这个Tensor与以前那个Tensor是共享内存的. 所以不安全
    y = x.data # x需要进行autograd
    # y和x是共享内存的,但是这里y已经不需要grad了, 所以会导致本来需要计算梯度的x也没有梯度可以计算.从而x不会得到更新!
 2)推荐用x.detach(), 这个仍旧是共享内存的, 也是使得y的requires_grad为False, 但是,如果x需要求导, 仍旧是可以自动求导的!
'''
```