计算机科学与技术学院神经网络与深度学习课程实验报告

实验题目：numpy 与 pytorch 使用		学号：201600301304
日期：2019/3/8	班级： 人工智能 16	姓名： 贾乘兴
Email：1131225623@qq.com		
实验目的:熟悉 numpy 与 pytorch,掌握基本的函数与实现,理解 numpy 与 pytorch 的优点		
实验软件和硬件环境： 操作系统 mac os，内存 16GB，编译器 pycharm		

实验原理和方法：

1. 掌握基本的 numpy 操作，如向量内积，矩阵索引等
2. 掌握 numpy 的 slice 与 index 操作
3. 掌握 padding 操作与多种填充方式
4. 掌握 numpy 数组与 pytorch 的 tensor 相互转换
5. 掌握 tensor 的基本操作，如内积等
6. relu 函数：

$$relu(x)=\begin{cases}x, x>0\\0, x\le 0\end{cases}$$

7. relu 函数导数

$$\frac{\partial relu(x)}{\partial x}=\begin{cases}1, x>0\\0, x<0\end{cases}$$

实验步骤：（不要求罗列完整源代码）

```
import numpy as np
import torch as tc

#numpy 向量内积函数
def vectorize_sumproducts(a, b):
    return a.dot(b)

#对 numpy 向量使用 relu 函数
def vectorize_Relu(a):
    index = np.where(a>0)
    b = np.zeros(a.shape)
    b[index] = a[index]
    return b

#relu 函数求导
```

```
def vectorize_PrimeRelu(a):
    index = np.where(a>0)
    b = np.zeros(a.shape)
    b[index] = 1
    return b

#对 feature 进行 slice 操作，按起始位置
def Slice_fixed_point(a, l, s):
    return np.array([a[i][s:s+l] for i in range(len(a))])

#对 feature 的 slice 操作，按末位置
def slice_last_point(a, e):
    return np.array([a[i][len(a[i])-e:len(a[i])] for i in range(len(a))])

#随机的 slice 操作
def slice_random_point(a, r):
    b = [len(a[i]) for i in range(len(a))]
    k = np.random.randint(min(b)-r)
    return np.array([a[i][k:k+r] for i in range(len(a))])

#镜像末尾的 padding
def pad_pattern_end(a):
    maxl = max([len(a[i]) for i in range(len(a))])
    return np.array([np.pad(a[i],((0,maxl-len(a[i])),(0,0)),'symmetric') for i
in range(len(a))])

#中心的常数填充
def pad_constant_central(a, val):
    maxl = max([len(a[i]) for i in range(len(a))])
    return np.array([np.pad(a[i],
(((maxl-len(a[i]))//2,maxl-len(a[i])-(maxl-len(a[i]))//2),(0,0)),
'constant',constant_values=val) for i in range(len(a))])

#numpy 类型与 torch 类型的相互转换
def numpy2tensor(a):
    return tc.from_numpy(a)
def tensor2numpy(a):
    return np.array(a)

#tensor 的内积
def Tensor_Sumproducts(a,b):
    return a.dot(b)

#tensor 的 relu
```

```
def Tensor_Relu(a):
    index = np.where(a>0)
    b = np.zeros(a.shape)
    b[index] = a[index]
    return tc.from_numpy(b)

#tensor 的 relu 导数
def Tensor_Relu_prime(a):
    index = np.where(a > 0)
    b = np.zeros(a.shape)
    b[index] = a[index]
    return tc.from_numpy(b)
```

结论分析与体会：
实验结果与测试

建立 numpy 数组
[-8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7]

内积
344

reshape 为 4*4
求 relu
```
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 1. 2. 3.]
 [4. 5. 6. 7.]]
```

求 relu 导数
```
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 1. 1. 1.]
 [1. 1. 1. 1.]]
```

生成不同长度的三维数组（频域 data，第二维长度不同）
```
z = np.arange(200).reshape((-1,4))
x = np.array([[z[0],z[1],z[2],z[3]],
[z[4],z[5],z[6]],[z[7],z[8],z[9],z[10],z[11],z[12],z[13],z[14]]])
```
生成数据
```
[[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]
  [12 13 14 15]]
```

```
  [16 17 18 19]
  [20 21 22 23]
  [24 25 26 27]]

 [[28 29 30 31]
  [32 33 34 35]
  [36 37 38 39]
  [40 41 42 43]
  [44 45 46 47]
  [48 49 50 51]
  [52 53 54 55]
  [56 57 58 59]]]

slice 取出 0:2
[[[ 0  1  2  3]
  [ 4  5  6  7]]

 [[16 17 18 19]
  [20 21 22 23]]

 [[28 29 30 31]
  [32 33 34 35]]]

slice 取出末尾两个
[[[ 8  9 10 11]
  [12 13 14 15]]

 [[20 21 22 23]
  [24 25 26 27]]

 [[52 53 54 55]
  [56 57 58 59]]]

随机 slice
[[[ 0  1  2  3]
  [ 4  5  6  7]]

 [[16 17 18 19]
  [20 21 22 23]]

 [[28 29 30 31]
  [32 33 34 35]]]
```

```
镜像 padding
[[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]
  [12 13 14 15]
  [12 13 14 15]
  [ 8  9 10 11]
  [ 4  5  6  7]
  [ 0  1  2  3]]

 [[16 17 18 19]
  [20 21 22 23]
  [24 25 26 27]
  [24 25 26 27]
  [20 21 22 23]
  [16 17 18 19]
  [16 17 18 19]
  [20 21 22 23]]

 [[28 29 30 31]
  [32 33 34 35]
  [36 37 38 39]
  [40 41 42 43]
  [44 45 46 47]
  [48 49 50 51]
  [52 53 54 55]
  [56 57 58 59]]]

0 常数中心对称填充
[[[ 0  0  0  0]
  [ 0  0  0  0]
  [ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]
  [12 13 14 15]
  [ 0  0  0  0]
  [ 0  0  0  0]]

 [[ 0  0  0  0]
  [ 0  0  0  0]
  [16 17 18 19]
  [20 21 22 23]
  [24 25 26 27]
  [ 0  0  0  0]
```

```
  [ 0  0  0  0]
  [ 0  0  0  0]]

 [[28 29 30 31]
  [32 33 34 35]
  [36 37 38 39]
  [40 41 42 43]
  [44 45 46 47]
  [48 49 50 51]
  [52 53 54 55]
  [56 57 58 59]]]

numpy 数组转 tensor
-8
-7
-6
-5
-4
-3
-2
-1
 0
 1
 2
 3
 4
 5
 6
 7
[torch.LongTensor of size 16]

tensor 转 numpy
[-8 -7 -6 -5 -4 -3 -2 -1  0  1  2  3  4  5  6  7]

tensor 内积
344

tensor 的 relu
0  0  0  0
 0  0  0  0
 0  1  2  3
 4  5  6  7
[torch.DoubleTensor of size 4x4]
```

tensor 的 relu 导数

```
0  0  0  0
 0  0  0  0
 0  1  2  3
 4  5  6  7
[torch.DoubleTensor of size 4x4]
```

通过使用 numpy 与 torch，可以更加便利的进行矩阵运算，也使得代码更加简洁，可读性增强

就实验过程中遇到和出现的问题，你是如何解决和处理的，自拟 1－3 道问答题：

1. pytorch 的 where 函数问题：在使用过程中无法调用该函数（版本问题），将其转为 numpy 数组后处理转为 torch 的 tensor
2. 不等长的 nparray 的 slice 与 index 问题：不等长的情形下 shape 为一维，故将每个内部的 nparry 处理完再合并为一个 nparray
3. 二维向三维的扩展：可将二维数据到三维视为声音信息的傅立叶变换转位频域空间，故对 padding 的理解也是按照该方式