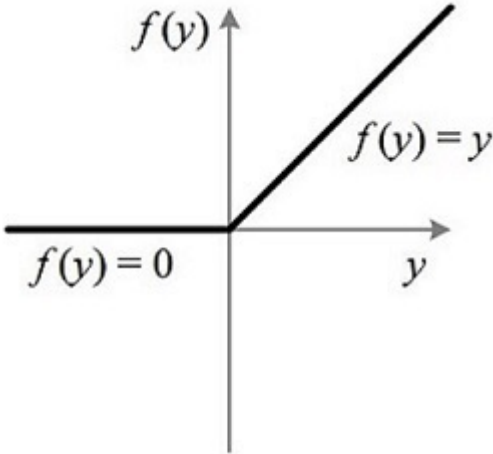


计算机科学与技术学院神经网络与深度学习课程实验报告

实验题目：基础操作		学号：201600301148
日期：3.8	班级：智能16	姓名：周阳
Email：862077860@qq.com		
实验目的： 1，熟悉 numpy，pytorch 基本操作		
实验软件和硬件环境： CPU：英特尔至强 E5 GPU：NVIDIA GeForce 1060 6G 内存：16G Pycharm Python 3.6		
<p>实验原理和方法：</p> <p>1，使用 numpy 和 pytorch 对基本的 pad 函数，切片函数，点乘（内积），relu 进行矩阵实现。</p> <p>2，relu 激活函数，如图所示。</p> <div style="text-align: center;">$ReLU(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$</div>		
<p>实验步骤：（不要求罗列完整源代码）</p> <div style="text-align: center;"><h2>1 向量操作</h2><ul style="list-style-type: none">• Takes two 1-dimensional arrays and sums the products of all the pairs.• Takes one 2-dimensional array and apply the relu function on all the values of the array.</div>		

- Takes one 2-dimensional array and apply the derivative of relu function on all the values of the array.

```
def vectorize_sumproducts(a, b):
```

```
    """
```

```
    Takes two 1-dimensional arrays and sums the products of all the pairs.
```

```
    :return:
```

```
    """
```

```
    return np.sum(a*b)
```

```
a = np.array([0, 1, 1])
```

```
b = np.array([0, 2, 3])
```

```
vectorize_sumproducts(a, b)
```

```
5
```

```
def vectorize_Relu(a):
```

```
    """
```

```
    Takes one 2-dimensional array and apply the relu function on all the values of the array.
```

```
    :return:
```

```
    """
```

```
    indexs = np.where(a>0)
```

```
    b = np.zeros(a.shape)
```

```
    b[indexs] = a[indexs]
```

```
    return b
```

```
a = np.array([[1, -1, 2], [3, 5, -6]])
```

```
vectorize_Relu(a)
```

```
array([[1., 0., 2.],
       [3., 5., 0.]])
```

```
def vectorize_PrimeRelu(a):
```

```
    """
```

```
    Takes one 2-dimensional array and apply the derivative of relu function on all the values of the array.
```

```
    :return:
```

```
    """
```

```
    indexs = np.where(a>0)
```

```
    b = np.zeros(a.shape)
```

```
    b[indexs] = 1
```

```
    return b
```

```
a = np.array([[1, -1, 2], [3, 5, -6]])
```

```
vectorize_PrimeRelu(a)
```

```
array([[1., 0., 1.],
       [1., 1., 0.]])
```

2 切片

- Takes one 3-dimensional array with the starting position and the length of the output instances. Your task is to slice the instances from the same starting position for the given length.
- Takes one 3-dimensional array with the length of the output instances. Your task is to keeping only the I last points for each instances in the dataset.
- Takes one 3-dimensional array with the length of the output instances. Your task is to slice the instances from a random point in each of the utterances with the given length. Please use function `numpy.random.randint` for generating the starting position.

```
def Slice_fixed_point(a, lengths, start):
```

```
    """
```

```
    Takes one 3-dimensional array with the starting position and the length of the output instances.  
    Your task is to slice the instances from the same starting position for the given length.
```

```
    :return:
```

```
    """
```

```
    return a[start:lengths+start]
```

```
### a feature不定长, 但是第一维确定
```

```
a = [
```

```
    [[1, 2, 3, 4],
```

```
    [2, 3, 4],
```

```
    [7, 8, 9, 10]],
```

```
    [[11, 12, 13, 14],
```

```
    [12, 13],
```

```
    [14, 15, 16, 17],
```

```
    [17, 18, 19, 110]],
```

```
    [[21, 22, 23, 24],
```

```
    [22, 23, 24, 25],
```

```
    [24, 25, 26, 27],
```

```
    [4, 5, 6, 7],
```

```
    [27, 28, 29, 210]],
```

```
]
```

```
a = np.array(a)
```

```
print(a.shape)
```

```
Slice_fixed_point(a, 2, 1)
```

```
(3,)
```

```
array([list([[11, 12, 13, 14], [12, 13], [14, 15, 16, 17], [17, 18, 19, 110]]),
```

```
       list([[21, 22, 23, 24], [22, 23, 24, 25], [24, 25, 26, 27], [4, 5, 6, 7], [27, 28, 29, 210]])],
```

```
       dtype=object)
```

```
def slice_last_point(a, length):
    """
    Takes one 3-dimensional array with the length of the output instances.
    Your task is to keeping only the 1 last points for each instances in the dataset.
    :return:
    """
    b = []
    for aa in a:
        b.append(aa[-length:])
    return np.array(b)
```

a feature不定长, 但是第一维确定

```
a = [
    [[1, 2, 3, 4],
     [2, 3, 4],
     [7, 8, 9, 10]],

    [[11, 12, 13, 14],
     [12, 13],
     [14, 15, 16, 17],
     [17, 18, 19, 110]],

    [[21, 22, 23, 24],
     [22, 23, 24, 25],
     [24, 25, 26, 27],
     [4, 5, 6, 7],
     [27, 28, 29, 210]],

]
a = np.array(a)

r = slice_last_point(a, 2)
print(r.shape)
r

(3, 2)
array([[list([2, 3, 4]), list([7, 8, 9, 10])],
       [list([14, 15, 16, 17]), list([17, 18, 19, 110])],
       [list([4, 5, 6, 7]), list([27, 28, 29, 210])]], dtype=object)
```

```
def slice_random_point(a):
    """
    Takes one 3-dimensional array with the length of the output instances.
    Your task is to slice the instances from a random point in each of the utterances with the given length.
    Please use function numpy.random.randint for generating the starting position.
    :return:
    """
    b = []
    for aa in a:
        b.append([])
        for aaa in aa:
            start = np.random.randint(0, len(aaa))
            b[-1].append(aaa[start:])
    return np.array(b)
```

a feature不定长, 但是第一维确定

```
a = [
    [[1, 2, 3, 4],
     [2, 3, 4],
     [7, 8, 9, 10]],

    [[11, 12, 13, 14],
     [12, 13],
     [14, 15, 16, 17],
     [17, 18, 19, 110]],

    [[21, 22, 23, 24],
     [22, 23, 24, 25],
     [24, 25, 26, 27],
     [4, 5, 6, 7],
     [27, 28, 29, 210]],

]
a = np.array(a)

r = slice_random_point(a)
print(r.shape)
r

(3,)
array([list([[4], [3, 4], [10]]),
       list([[11, 12, 13, 14], [13], [17], [19, 110]]),
       list([[24], [25], [24, 25, 26, 27], [5, 6, 7], [27, 28, 29, 210]]),
       dtype=object])
```

3 填充 (pad)

- Takes one 3-dimensional array. Your task is to pad the instances from the end position as shown in the example below. That is, you need to pad the reflection of the utterance mirrored along the edge of the array.
- Takes one 3-dimensional array with the constant value of padding. Your task is to pad the instances with the given constant value while maintaining the array at the center of the padding.

```
def pad_pattern_end(a):
    """
    Takes one 3-dimensional array.
    Your task is to pad the instances from the end position as shown in the example below.
    That is, you need to pad the reflection of the utterance mirrored along the edge of the array.
    :return:
    """
    b = []
    for aa in a:
        to_be_pad_len = max([len(aaa) for aaa in aa])
        b.append([])
        for aaa in aa:
            len_pad = to_be_pad_len - len(aaa)
            b[-1].append(np.pad(aaa, len_pad, 'symmetric')[len_pad:])
    return np.array(b)
```

a feature不定长, 但是第一维确定

```
a = [
    [[1, 2, 3, 4],
     [2, 3, 4],
     [7, 8, 9, 10]],

    [[11, 12, 13, 14],
     [12, 13],
     [14, 15, 16, 17],
     [17, 18, 19, 110]],

    [[21, 22, 23, 24],
     [22, 23, 24, 25],
     [24, 25, 26, 27],
     [4, 5, 6, 7],
     [27, 28, 29, 210, 211, 212, 213]],

]
a = np.array(a)
r = pad_pattern_end(a)
print(r.shape)
r

(3,)
array([list([array([1, 2, 3, 4]), array([2, 3, 4, 4]), array([ 7, 8, 9, 10])]),
       list([array([11, 12, 13, 14]), array([12, 13, 13, 12]), array([14, 15, 16, 17]), array([ 17, 18, 19, 110])]),
       list([array([21, 22, 23, 24, 24, 23, 22]), array([22, 23, 24, 25, 25, 24, 23]), array([24, 25, 26, 27, 27, 26,
25]), array([4, 5, 6, 7, 7, 6, 5]), array([ 27, 28, 29, 210, 211, 212, 213])])]),
      dtype=object)
```

```
def pad_constant_central(a, pad_token):
    """
    Takes one 3-dimensional array with the constant value of padding.
    Your task is to pad the instances with the given constant value while maintaining the array at the center of
    the padding.
    :return:
    """
    b = []
    for aa in a:
        to_be_pad_len = max([len(aaa) for aaa in aa])
        b.append([])
        for aaa in aa:
            len_pad = to_be_pad_len - len(aaa)
            if len_pad % 2 == 0:
                start = int(len_pad/2)
            else:
                start = int(len_pad/2) + 1
            b[-1].append(np.pad(aaa, len_pad, 'constant', constant_values = pad_token)[start:start + to_be_pad_len])
    return np.array(b)
```

a feature 不定长, 但是第一维确定

```
a = [
    [[1, 2, 3, 4],
     [2, 3, 4],
     [7, 8, 9, 10]],

    [[11, 12, 13, 14],
     [12, 13],
     [14, 15, 16, 17],
     [17, 18, 19, 110]],

    [[21, 22, 23, 24],
     [22, 23, 24, 25],
     [24, 25, 26, 27],
     [4, 5, 6, 7],
     [27, 28, 29, 210, 211, 212, 213]],

]
a = np.array(a)

r = pad_constant_central(a, 10000)
print(r.shape)
r

(3,)
array([list([array([1, 2, 3, 4]), array([ 2, 3, 4, 10000]), array([ 7, 8, 9, 10])]),
       list([array([11, 12, 13, 14]), array([10000, 12, 13, 10000]), array([14, 15, 16, 17]), array([ 17, 1
8, 19, 110])]),
       list([array([10000, 21, 22, 23, 24, 10000, 10000]), array([10000, 22, 23, 24, 25, 1
0000, 10000]), array([10000, 24, 25, 26, 27, 10000, 10000]), array([10000, 4, 5, 6, 7,
10000, 10000]), array([ 27, 28, 29, 210, 211, 212, 213])])],
      dtype=object)
```

4 pytorch 与 numpy

- Takes a numpy ndarray and converts it to a PyTorch tensor. Function `torch.tensor` is one of the simple ways to implement it but please do not use it this time.
- Takes a PyTorch tensor and converts it to a numpy ndarray.

```
def numpy2tensor(a):
```

```
    """
```

```
    Takes a numpy ndarray and converts it to a PyTorch tensor.
```

```
    Function torch.tensor is one of the simple ways to implement it but please do not use it this time.
```

```
    :return:
```

```
    """
```

```
    return torch.from_numpy(a)
```

```
a = np.array([
    [1, 2, 3],
    [4, 5, 6]
])
numpy2tensor(a)
```

```
tensor([[ 1,  2,  3],
        [ 4,  5,  6]])
```

```
def tensor2numpy(a):
```

```
    """
```

```
    Takes a PyTorch tensor and converts it to a numpy ndarray.
```

```
    :return:
```

```
    """
```

```
    return a.numpy()
```

```
a = torch.Tensor([
    [1, 2, 3],
    [2, 3, 4]
])
```

```
tensor2numpy(a)
```

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

5 pytorch 中的 Tensor 基本操作

- you are to implement the function tensor sumproducts that takes two tensors as input. returns the sum of the element-wise products of the two tensors. ### Tensor ReLu and ReLu prime
- Takes one 2-dimensional tensor and apply the relu function on all the values of the tensor.
- Takes one 2-dimensional tensor and apply the derivative of relu function on all the values of the tensor.


```
def Tensor_Sumproducts(a,b):
    """
    you are to implement the function tensor sumproducts that takes two tensors as input.
    returns the sum of the element-wise products of the two tensors.
    :return:
    """
    return (a*b).sum()
```

```
a = torch.Tensor([
    [1,2,3],
    [2,3,4]
])

b = torch.Tensor([
    [2,3,4],
    [5,6,7]
])

Tensor_Sumproducts(a,b)

tensor(76.)
```

```
def Tensor_ReLu(a):
    """
    Takes one 2-dimensional tensor and apply the relu function on all the values of the tensor.
    :return:
    """

    return torch.where(a<=0, torch.full_like(a, 0), a)

def Tensor_ReLu_prime(a):
    """
    Takes one 2-dimensional tensor and apply the derivative of relu function on all the values of the tensor.
    :return:
    """

    return torch.where(a<=0, torch.full_like(a, 0), torch.full_like(a, 1))
```

```
a = torch.Tensor([
    [-11,-2,3],
    [2,3,0]
])

Tensor_ReLu(a)

tensor([[ 0.,  0.,  3.],
        [ 2.,  3.,  0.]])

Tensor_ReLu_prime(a)

tensor([[ 0.,  0.,  1.],
        [ 1.,  1.,  0.]])
```

结论分析与体会：

- 1, numpy 与 pytorch 中封装了大量的好用，并且高效的矩阵运算库。
- 2, 使用矩阵运算，在数据量特别大的时候，会比使用 for 循环（python 语言导致）的速度有大量提升。
- 3, numpy 与 pytorch 中还有更高层的对数据的处理操作，如 torch 中的 torchvision 是对图像视频操作的库，torchtext 有对文本操作的大量库。

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, 用什么样的方法可以直接从 array 中取出符合条件的 index 而不用逐一遍历？

答： 使用 torch.where（），与 numpy.where（） 函数可以做到，并且注意二者的用法有非常大的不同。

2, 文本处理或音频处理需要注意什么？

答： 需要注意 padding（对齐）才能使最终的数字矩阵输入神经网络。