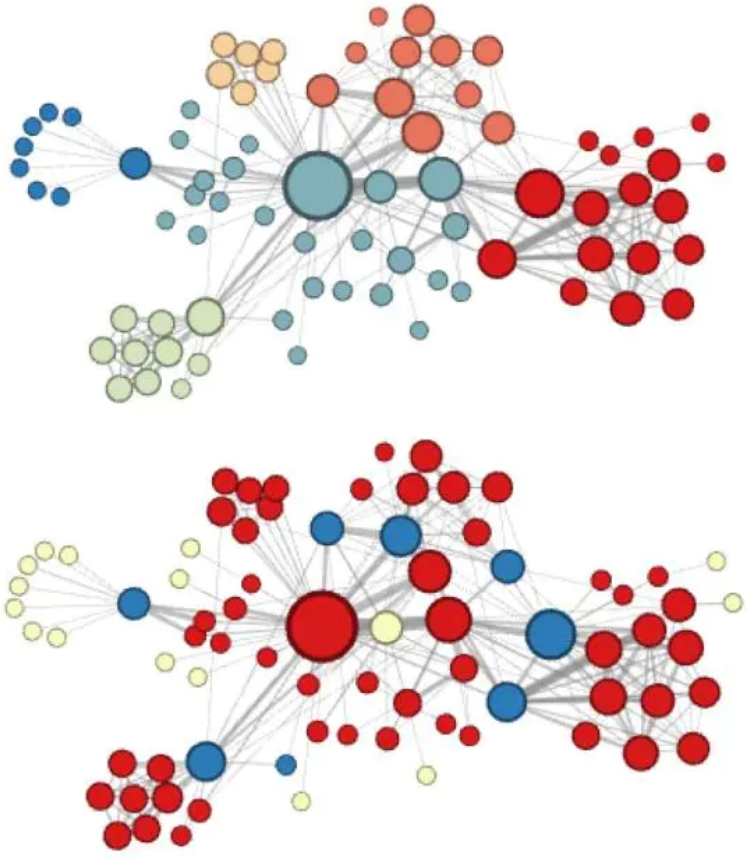
通俗讲解pytorch中nn.Embedding原理及使用

top_小酱油 关注
3 2020.03.24 15:36:43 字数 1,547 阅读 88,642

函数调用形式



```
torch.nn.Embedding(num_embeddings, embedding_dim, padding_idx=None,
 max_norm=None,  norm_type=2.0,  scale_grad_by_freq=False,
 sparse=False,  _weight=None)
```

其为一个简单的存储固定大小的词典的嵌入向量的查找表，意思就是说，给一个编号，嵌入层就能返回这个编号对应的嵌入向量，嵌入向量反映了各个编号代表的符号之间的语义关系。

输入为一个编号列表，输出为对应的符号嵌入向量列表。

参数解释

- num_embeddings (python:int) – 词典的大小尺寸，比如总共出现5000个词，那就输入5000。此时index为（0-4999）
  embedding_dim (python:int) – 嵌入向量的维度，即用多少维来表示一个符号。
- padding_idx (python:int, optional) – 填充id，比如，输入长度为100，但是每次的长度并不一样，后面就需要用统一的数字填充，而这里就是指定这个数字，这样，网

热门故事
闺蜜的一条朋友圈，结束了我和老公5年的婚姻
娘家拆迁分了两套新房，婆婆让我把房本写上小叔子的名？
前任一哭，现任必输
老婆偷偷拿30万给小舅子买豪车，被我一招"制服"
推荐阅读
图解Numpy入门教程 阅读 846
python之序列 阅读 235
Stata 横纵转换：转置、长宽互转全搞定！ 阅读 454
Scanpy数据结构：AnnData 阅读 577
python数据类型——数字 阅读 114

- norm_type (python:float, optional) – 指定利用什么范数计算，并用于对比max_norm，默认为2范数。
- scale_grad_by_freq (boolean, optional) – 根据单词在mini-batch中出现的频率，对梯度进行放缩。默认为False.
- sparse (bool, optional) – 若为True,则与权重矩阵相关的梯度转变为稀疏张量。

下面是关于Embedding的使用

torch.nn包下的Embedding，作为训练的一层，随模型训练得到适合的词向量。

```
#建立词向量层
embed = torch.nn.Embedding(n_vocabulary,embedding_size)
```

找到对应的词向量放进网络：词向量的输入应该是什么样子

实际上，上面通过随机初始化建立了词向量层后，建立了一个“二维表”，存储了词典中每个词的词向量。每个mini-batch的训练，都要从词向量表找到mini-batch对应的单词的词向量作为RNN的输入放进网络。那么怎么把mini-batch中的每个句子的所有单词的词向量找出来放进网络呢，**输入是什么样子，输出是什么样子?**

首先我们知道肯定先要建立一个词典，建立词典的时候都会建立一个dict：word2id：存储单词到词典序号的映射。假设一个mini-batch如下所示：

```
['I am a boy.','How are you?','I am very lucky.']
```

显然，这个mini-batch有3个句子，即batch_size=3

第一步首先要做的是：将句子标准化，所谓标准化，指的是：大写转小写，标点分离，这部分很简单就略过。经处理后，mini-batch变为：

```
[['i','am','a','boy','.'],['how','are','you','?'],['i','am','very','lucky','.']]
```

可见，这个list的元素成了一个个list。还要做一步：将上面的三个list按单词数从多到少排列。标点也算单词。至于为什么，后面会说到。

那就变成了：

```
batch = [['i','am','a','boy','.'],['i','am','very','lucky','.'], ['how','are','you','?']]
```

可见，每个句子的长度，即每个内层list的元素数为：5,5,4。这个长度也要记录。

```
lens = [5,5,4]
```

之后，为了能够处理，将batch的单词表示转为在词典中的index序号，这就是word2id的作用。转换过程很简单，假设转换之后的结果如下所示，当然这些序号是我编的。

```
batch = [[3,6,5,6,7],[6,4,7,9,5], [4,5,8,7]]
```

热门故事

闺蜜的一条朋友圈，结束了我和老公5年的婚姻

娘家拆迁分了两套新房，婆婆让我把房本写上小叔子的名?

前任一哭，现任必输

老婆偷偷拿30万给小舅子买豪车，被我一招“制服”

推荐阅读

图解Numpy入门教程
阅读 846

python之序列
阅读 235

Stata 横纵转换：转置、长宽互转全搞定!
阅读 454

Scanpy数据结构：AnnData
阅读 577

python数据类型——数字
阅读 114

```
batch = [[3,6,5,6,7,1],[6,4,7,9,5,1],[4,5,8,7,1]]
```

那么长度要更新：

```
lens = [6,6,5]
```

很显然，这个mini-batch中的句子长度**不一致！**所以为了规整的处理，对长度不足的句子，进行填充。填充PAD假设序号是2，填充之后为：

```
batch = [[3,6,5,6,7,1],[6,4,7,9,5,1],[4,5,8,7,1,2]]
```

这样就可以直接取词向量训练了吗?

不能！上面batch有3个样例，RNN的每一步要输入每个样例的一个单词，一次输入batch_size个样例，所以batch要按list外层是时间步数(即序列长度)，list内层是batch_size排列。即batch的维度应该是：

```
[seq_len,batch_size]
[seq_len,batch_size]
[seq_len,batch_size]
```

重要的问题说3遍!

怎么变换呢？变换方法可以是：使用itertools模块的zip_longest函数。而且，使用这个函数，连填充这一步都可以省略，因为这个函数可以实现填充！

```
batch = list(itertools.zip_longest(batch,fillvalue=PAD))
# fillvalue就是要填充的值，强制转成list
```

经变换，结果应该是：

```
batch = [[3,6,4],[6,4,5],[5,7,8],[6,9,7],[7,5,1],[1,1,2]]
```

记得我们还记录了一个lens：

```
lens = [6,6,5]
```

batch还要转成LongTensor：

```
batch=torch.LongTensor(batch)
```

这里的batch就是词向量层的输入。

词向量层的输出是什么样的?

好了，现在使用建立了的embedding直接通过batch取词向量了，如：

热门故事

闺蜜的一条朋友圈，结束了我和老公5年的婚姻

娘家拆迁分了两套新房，婆婆让我把房本写上小叔子的名？

前任一哭，现任必输

老婆偷偷拿30万给小舅子买豪车，被我一招"制服"

推荐阅读

图解Numpy入门教程
阅读 846

python之序列
阅读 235

Stata 横纵转换：转置、长宽互转全搞定！
阅读 454

Scanpy数据结构：AnnData
阅读 577

python数据类型——数字
阅读 114

```
tensor([[[-0.2699,  0.7401, -0.8000,  0.0472,  0.9032, -0.0902],
         [-0.2675,  1.8021,  1.4966,  0.6988,  1.4770,  1.1235],
         [ 0.1146, -0.8077, -1.4957, -1.5407,  0.3755, -0.6805]],

        [[-0.2675,  1.8021,  1.4966,  0.6988,  1.4770,  1.1235],
         [ 0.1146, -0.8077, -1.4957, -1.5407,  0.3755, -0.6805],
         [-0.0387,  0.8401,  1.6871,  0.3057, -0.8248, -0.1326]],

        [[-0.0387,  0.8401,  1.6871,  0.3057, -0.8248, -0.1326],
         [-0.3745, -1.9178, -0.2928,  0.6510,  0.9621, -1.3871],
         [-0.6739,  0.3931,  0.1464,  1.4965, -0.9210, -0.0995]],

        [[-0.2675,  1.8021,  1.4966,  0.6988,  1.4770,  1.1235],
         [-0.7411,  0.7948, -1.5864,  0.1176,  0.0789, -0.3376],
         [-0.3745, -1.9178, -0.2928,  0.6510,  0.9621, -1.3871]],

        [[-0.3745, -1.9178, -0.2928,  0.6510,  0.9621, -1.3871],
         [-0.0387,  0.8401,  1.6871,  0.3057, -0.8248, -0.1326],
         [ 0.2837,  0.5629,  1.0398,  2.0679, -1.0122, -0.2714]],

        [[ 0.2837,  0.5629,  1.0398,  2.0679, -1.0122, -0.2714],
         [ 0.2837,  0.5629,  1.0398,  2.0679, -1.0122, -0.2714],
         [ 0.2242, -1.2474,  0.3882,  0.2814, -0.4796,  0.3732]]],
       grad_fn=<EmbeddingBackward>)
```

维度的前两维和前面讲的是一致的。可见多了一个第三维，这就是词向量维度。所以，Embedding层的输出是：

```
[seq_len,batch_size,embedding_size]
```

一些注意的点

- nn.embedding的输入只能是编号，不能是隐藏变量，比如one-hot，或者其它，这种情况，可以自己建一个自定义维度的线性网络层，参数训练可以单独训练或者跟随整个网络一起训练（看实验需要）
- 如果你指定了padding_idx，注意这个padding_idx也是在num_embeddings尺寸内的，比如符号总共有500个，指定了padding_idx，那么num_embeddings应该为501
- embedding_dim的选择要注意，根据自己的符号数量，举个例子，如果你的词典尺寸是1024，那么极限压缩（用二进制表示）也需要10维，再考虑词性之间的相关性，怎么也要在15-20维左右，虽然embedding是用来降维的，但是>- 也要注意这种极限维度，结合实际情况，合理定义
- 其他的好像也没啥要注意的啦~，欢迎评论区补充(●'ω`●)

热门故事

闺蜜的一条朋友圈，结束了我和老公5年的婚姻

娘家拆迁分了两套新房，婆婆让我把房本写上小叔子的名？

前任一哭，现任必输

老婆偷偷拿30万给小舅子买豪车，被我一招"制服"

推荐阅读

图解Numpy入门教程
阅读 846

python之序列
阅读 235

Stata 横纵转换：转置、长宽互转全搞定！
阅读 454

Scanpy数据结构：AnnData
阅读 577

python数据类型——数字
阅读 114