* Embedding 字面理解是 “嵌入”，实质是一种映射，从语义空间到向量空间的映射，同时尽可能在向量空间保持原样本在语义空间的关系，如语义接近的两个词汇在向量空间中的位置也比较接近。
* Embedding的训练过程

1. 导入Keras的相关库
2. 给出文本内容和label
3. 将文本编码成数字格式并padding到相同长度
4. 同一个单词对应相同的数字，one\_hot编码映射到[1,n]，不包括0，n为上述的vocab\_size，为估计的词汇表大小
5. 模型的定义

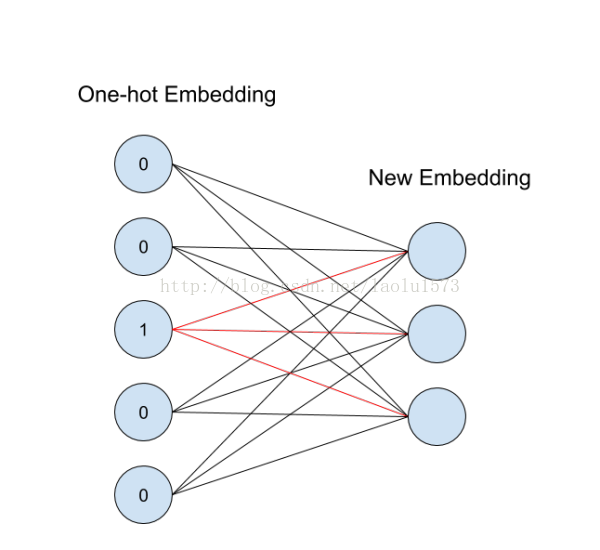
* 说明：

keras.layers.Embedding(input\_dim, output\_dim, input\_length)

* input\_dim：这是文本数据中词汇的取值可能数。例如，如果您的数据是整数编码为0-9之间的值，那么词汇的大小就是10个单词；
* output\_dim：这是嵌入单词的向量空间的大小。它为每个单词定义了这个层的输出向量的大小。例如，它可能是32或100甚至更大，可以视为具体问题的超参数；
* input\_length：这是输入序列的长度，就像您为Keras模型的任何输入层所定义的一样，也就是一次输入带有的词汇个数。例如，如果您的所有输入文档都由1000个字组成，那么input\_length就是1000

one-hot 是无法考虑语义间的相互关系的，但embedding向量的训练是要借助one-hot的。

one-hot把每一个单词映射成一个整数，但实际上这个整数就表示了50维向量中 1 所在的索引位置，用整数显示是为了更好理解和表示，而实际在网络中，它的形式可以理解为如下图（下面相当于one-hot向量为5维，输出embedding向量为3维）



* 深入理解 Embedding层的本质

建立在one-hot编码的基础上

例：

公 [0 0 0 0 1 0 0 0 0 0]

主 [0 0 0 1 0 0 0 0 0 0]

很 [0 0 1 0 0 0 0 0 0 0]

漂 [0 1 0 0 0 0 0 0 0 0]

亮 [1 0 0 0 0 0 0 0 0 0]

* embedding层做了个什么呢？

它把我们的稀疏矩阵，通过一些线性变换（在CNN中用全连接层进行转换，也称为查表操作），变成了一个密集矩阵，这个密集矩阵用了N（例子中N=3）个特征来表征所有的文字，在这个密集矩阵中，表象上代表着密集矩阵跟单个字的一一对应关系，实际上还蕴含了大量的字与字之间，词与词之间甚至句子与句子之间的内在关系。他们之间的关系，用的是嵌入层学习来的参数进行表征。从稀疏矩阵到密集矩阵的过程，叫做embedding，很多人也把它叫做查表，因为他们之间也是一个一一映射的关系。

更重要的是，这种关系在反向传播的过程中，是一直在更新的，因此能在多次epoch后，使得这个关系变成相对成熟，即：正确的表达整个语义以及各个语句之间的关系。这个成熟的关系，就是embedding层的所有权重参数。