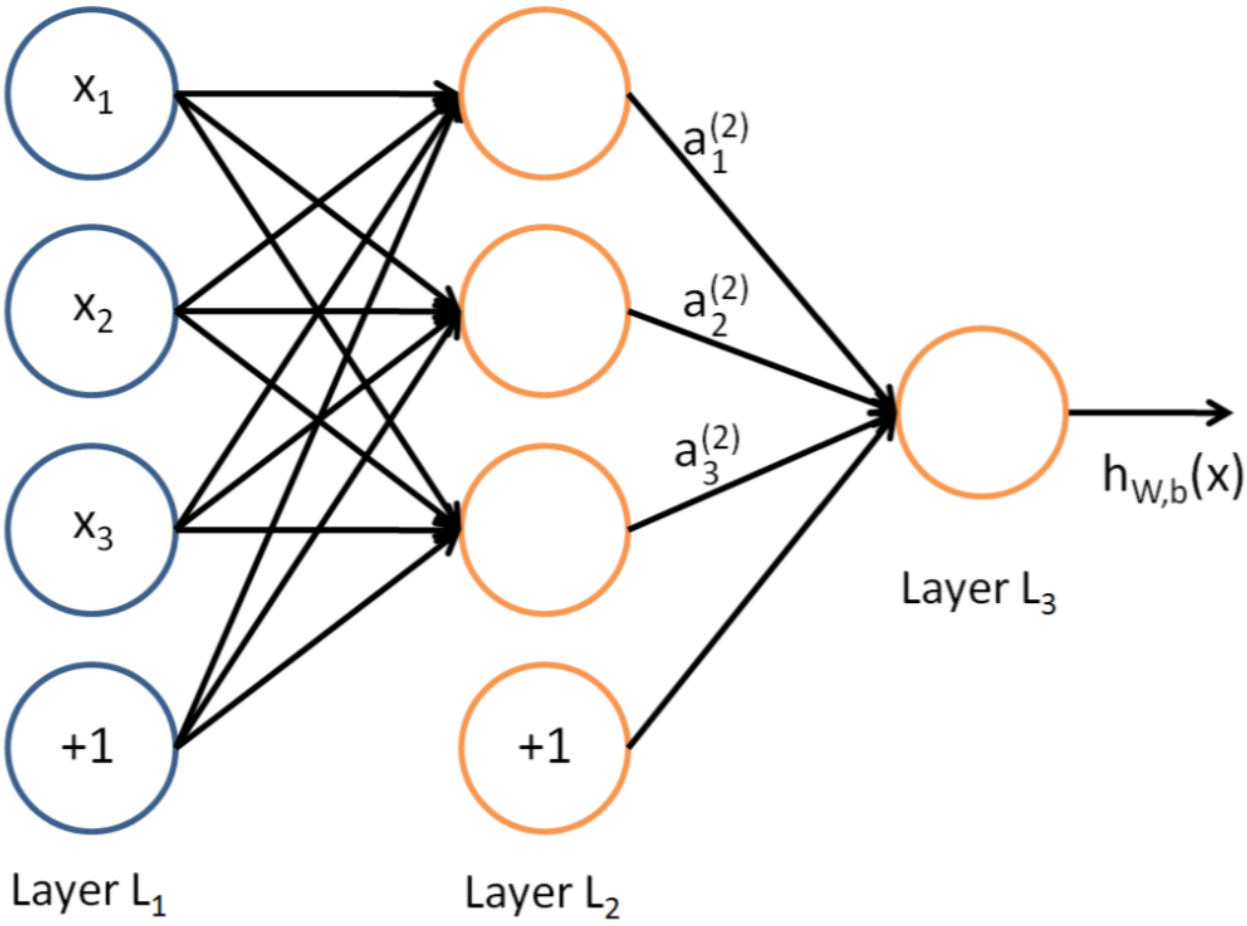
1.神经网络与BP神经网络

- 神经网络模型：



这是一个典型的三层神经网络模型，有输入层、隐含层和输出层，通常在隐含层引入一个非线性变换函数，例如sigmoid函数；通过该函数可以将模型变成一个非线性模型。

假设我们有一组数据$\{X1, X_2, \ldots, X_n\}$，每一个X_i代表一个N维的数据，将数据输入神经网络模型后，得到一个结果y_j;y_j可以是一个数(0或1)，也可以是一组向量。通常，这样得到结果往往是很难达到满意的结果，所以我们自然想到通过定义一个损失函数，通过损失函数来判断如何更新我们神经网络中各个链接的权重。

- BP(Backpropagation)

介绍神经网络模型之前需要再回顾共轭梯度法(Gradient Descent)。

给定一组参数：$\theta = \{w_1, w_2, \ldots, b_1, b_2, \ldots, \}$，通过不断迭代更新参数：

$$\theta^0 \rightarrow \theta^1 \rightarrow \theta^2 \ldots\ldots$$

$$\nabla L(\theta) = \begin{bmatrix} \frac{\partial L(\theta)}{\partial w_1} \\ \frac{\partial L(\theta)}{\partial w_2} \\ \vdots \\ \frac{\partial L(\theta)}{\partial b_1} \\ \frac{\partial L(\theta)}{\partial b_2} \\ \vdots \end{bmatrix}$$

计算好每个参数的偏导数后，可以更新神经网络中的参数：

$$\theta^1 = \theta^0 - \eta \nabla L(\theta^0)$$
$$\theta^2 = \theta^1 - \eta \nabla L(\theta^1)$$

参考以下的神经网络模型，

 avatar

令C为损失函数，根据求导的链式法则：

$$\frac{\partial C}{\partial w} = \frac{\partial z}{\partial w} \frac{\partial C}{\partial z}$$

其中，$\frac{\partial z}{\partial w}$是用来计算模型的前向传播，$\frac{\partial C}{\partial z}$用来计算模型的后向传播。观察以上模型可以发现：

$$\frac{\partial C}{\partial z} = \frac{\partial a}{\partial z} \frac{\partial C}{\partial a} = \sigma'(z) \frac{\partial C}{\partial a}$$

令$w_3 = \frac{\partial z'}{\partial a}$，$w_4 = \frac{\partial z''}{\partial a}$，可以得到：

$$\frac{\partial C}{\partial z} = \sigma'(w_3 \frac{\partial C}{\partial z'} + w_4 \frac{\partial C}{\partial z''})$$

根据以上计算实例，可以明白BP神经网络的反向传播原理。

2. CNN(Convolutional Neural Network)

- 为什么使用CNN模型

  CNN模型通常用来处理图像分类问题，那么对于一个多分类的问题，在原理是我们是可以使用

传统的机器学习去完成的，比如多层感知机模型或者k-NN模型等。之所以使用CNN模型，是因为相比较于传统的机器学习模型，CNN模型不需要我们做特征工程。传统的机器学习模型需要对处理的对象做特征工程，当数据的维度非常大时，特征工程的难度是特性高的。所以，使用CNN模型，可以直接将数据输入进去，模型可以自己去学习判断哪些是好的特征。

- CNN模型的流程

这是一个CNN模型基本的计算流程。首先，输入一张图片，模型对图片做卷积运算(Convolution)，然后再进行池化操作(Max Pooling)，将这两步重复多次后，可以得到这张图片的特征向量并输入到搭建好的神经网络模型中。

. 卷积运算(Convolution)

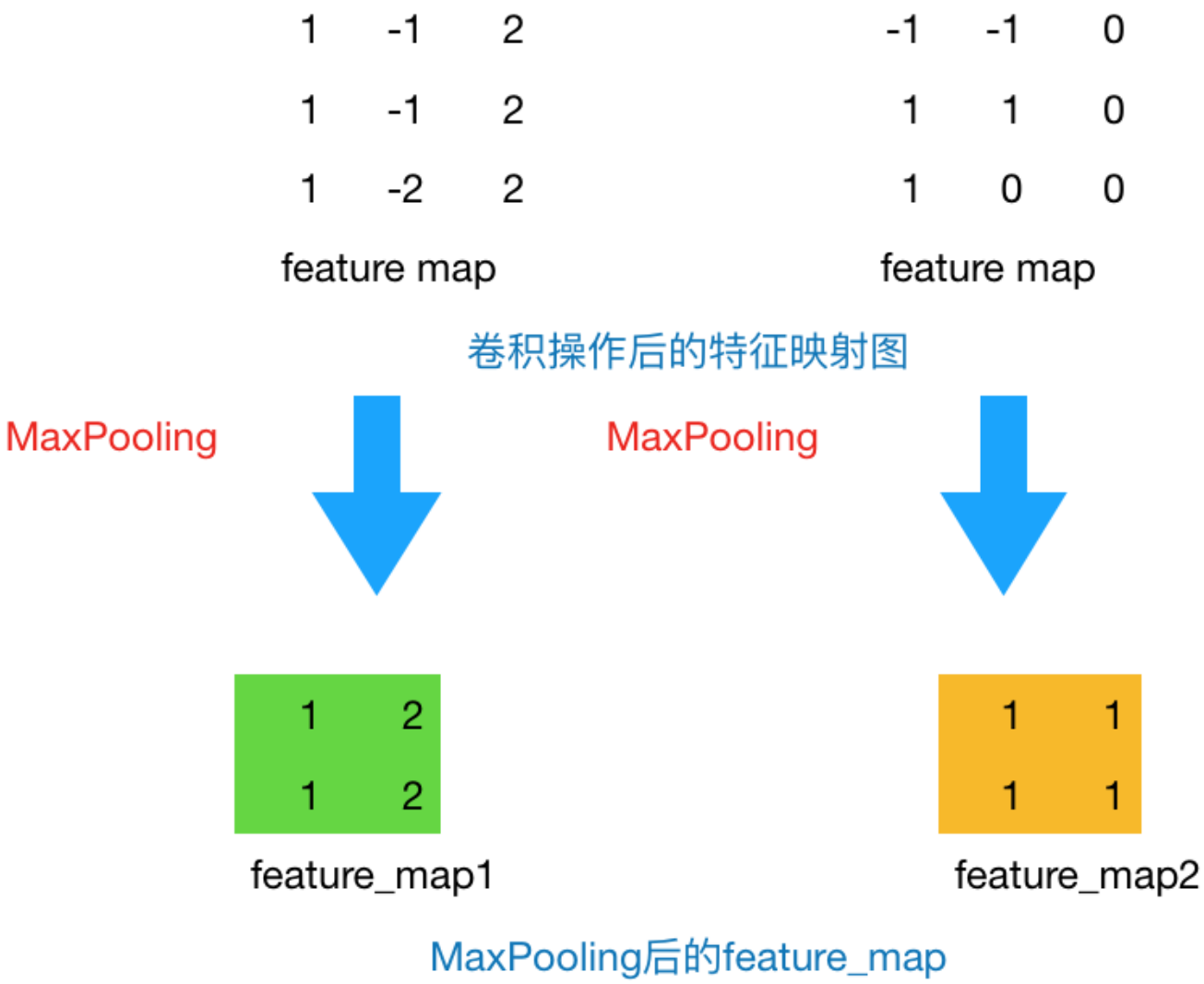
在卷积运算中，我们假设I是一张二维图像，同时我们有一个二维的核K，得到如下的映射：

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

举个例子，假设我们有一张4*4的图片，以及卷积核1、卷积核2，通过以上定义的卷积运算，可以得到两张特征图(feature map)

. 池化层(Max Pooling)

池化层的主要目的是通过降采样的方式，在不影响图像质量的情况下，压缩图片，减少参数。例如：

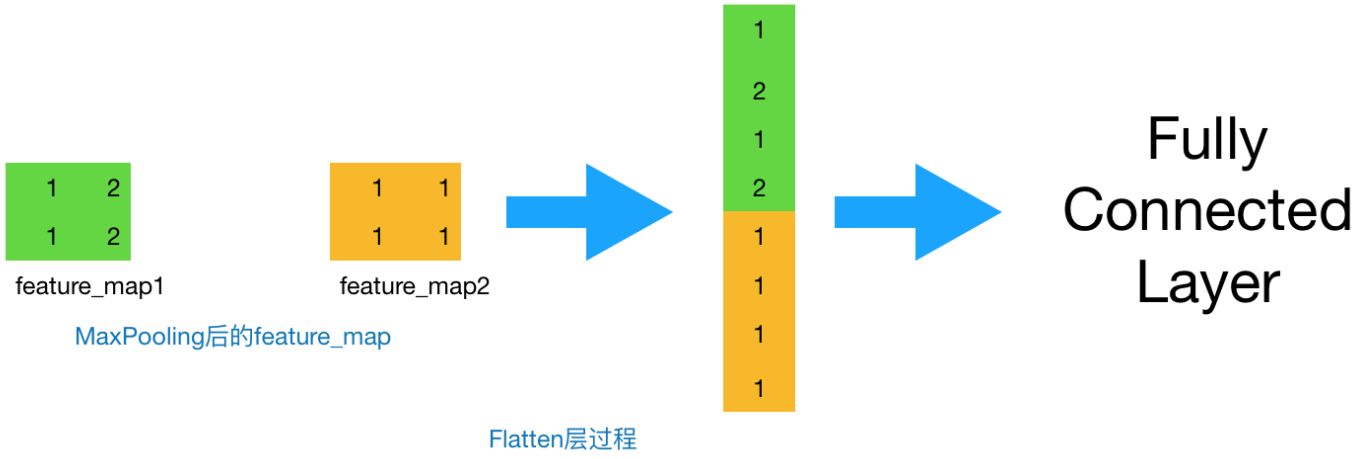


池化层的意义就是使用某一位置相邻的总体统计特征来代替网络在该位置的输出，或者说我们在提取特征时，只关系特征本身，而不是特征出现的位置。常用的池化函数有：相邻矩形区域平均值，L^2范数以及基于距中心像素距离的加权平均。

. Zero Padding和Flatten

因为图片在经过卷积运算和池化层后，会变得越来越小，所以通过补零(Zero Padding)，可以使得输出的图像和之前相比大小不变。

为了将得到的特征图输入到搭建好的神经网络模型中，需要将特征图平铺开来(Flatten)，操作如下:



3.RNN(Recurrent Neural Network)

和能够有效处理空间数据的CNN模型不同，RNN是一种能够有效处理时序信息的模型，经常应用于语言模型、文本分析、机器翻译等。

- 语言模型

  假设文本T中的词汇以此看作$w_1, w_2, \ldots, w_T$，该语言模型序列的概率：

$$P(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} P(w_t | w_1, \ldots, w_{t-1})$$

  例如，一个含有4个词的文本序列的概率:

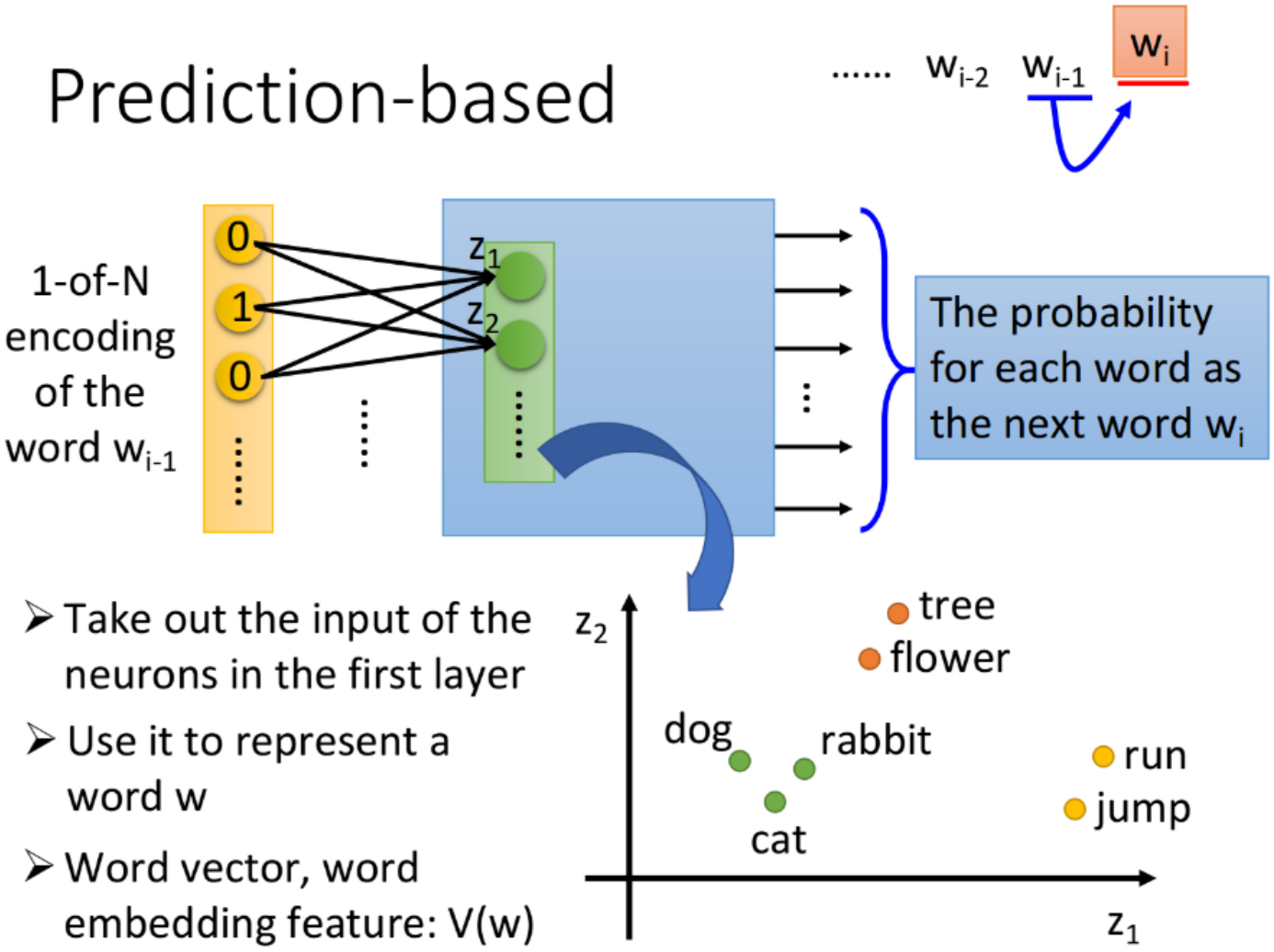
$$P(w_1, w_2, w_3, w_4) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)P(w_4|w_1, w_2, w_3)$$

- Word Embedding

  Word Embedding就是通过一种方法，让机器取阅读大量的文章，可以自动将一些语义相近的词汇映射到相近的高维空间去。

  首先对词语进行1-of-N Encoding的方法进行描述，对每个词汇建立一个$vector$，为了和一般的方法区分开来(1-of-N Encoding，clustering)，可以设计一个神经网络模型，它要做的就是根据当前的wordw_{i-1}，来预测下一个可能出现的wordw_i是什么。假设我们使用1-of-N encoding把w_{i-1}表示成feature vector，它作为neural network的input，output的维数和input相等，只不过每一维都是小数，代表在1-of-N Encoding中该维为1其余维为0所对应的word会是下一个wordw_i的概率。

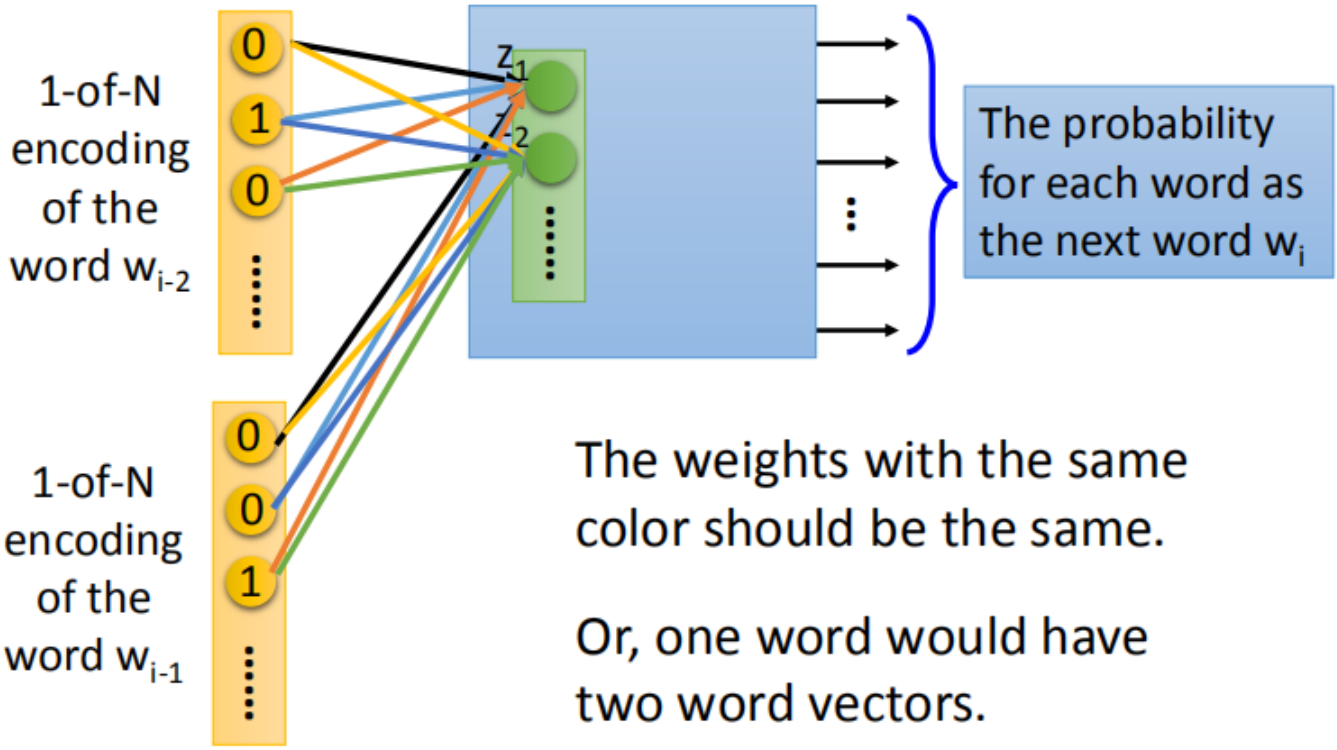
  假如将第一个隐藏层的的输入$z_1, z_2, \ldots, z_m$拿出来，这个向量可以作为对应单词的另一种表示形式，将这个向量作在图上，可以发现相近的词向量具有相近的词性。



- 共享权重

  因为仅仅只靠当前词汇去预测下一个词汇，那么模型的预测能力会很弱，我们可以通过共享权重的方法，通过多个词汇去预测下一个词汇。

Prediction-based – Sharing Parameters



The weights with the same color should be the same.

Or, one word would have two word vectors.