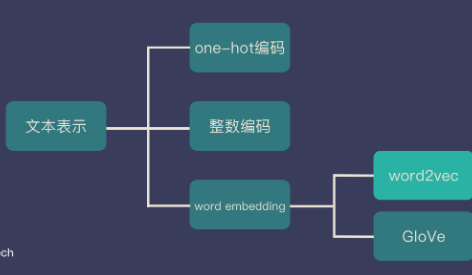
## 词向量化有两种方式：

1. one-hot编码  (缺点是：词越多维度越大， 词与词之间没有相关性，无法理解语义)

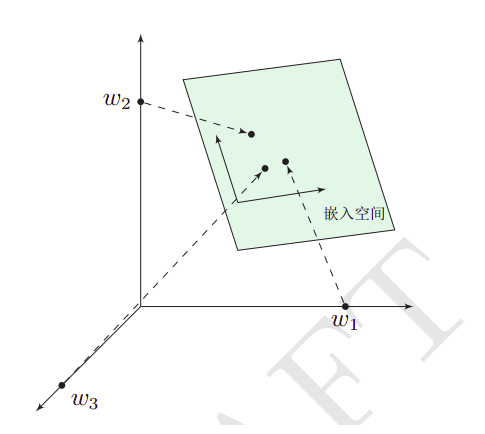
2. word embedding： 词嵌入。 优点是降低维度(将vector每一个元素由整形改为浮点型，变为整个实数范围的表示；将原来稀疏的巨大维度压缩嵌入到一个更小维度的空间)



### 什么是词嵌入 | word embedding？

它就是将「不可计算」「非结构化」的词转化为「可计算」「结构化」的向量。

这一步解决的是”将现实问题转化为数学问题“，是人工智能非常关键的一步。



把词映射为实数域向量的技术也叫词嵌入（word embedding），核心思想就是见每个词映射成低维空间（通常K=50-300维）上的一个稠密向量（Dense Vector）。

词向量既能够降低维度，又能够capture到当前词在本句子中上下文的信息（表现为前后距离关系）

### 词嵌入优势

他可以将文本通过一个低维向量来表达，不像 one-hot 那么长。

语意相似的词在向量空间上也会比较相近。

通用性很强，可以用在不同的任务中。

### word embedding 算法

词嵌入模型：

   1. CBOW模型 (Continuous Bag-of-Words Model)： 上下文来预测当前词

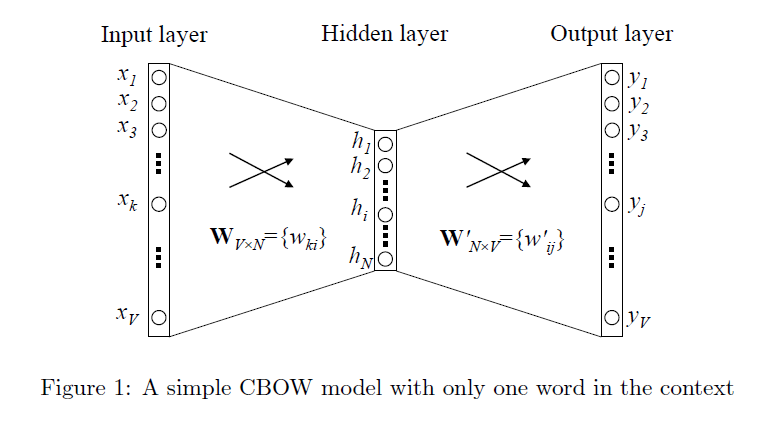
   2. Skip-gram模型(Continuous Skip-gram Model)：当前词预测上下文

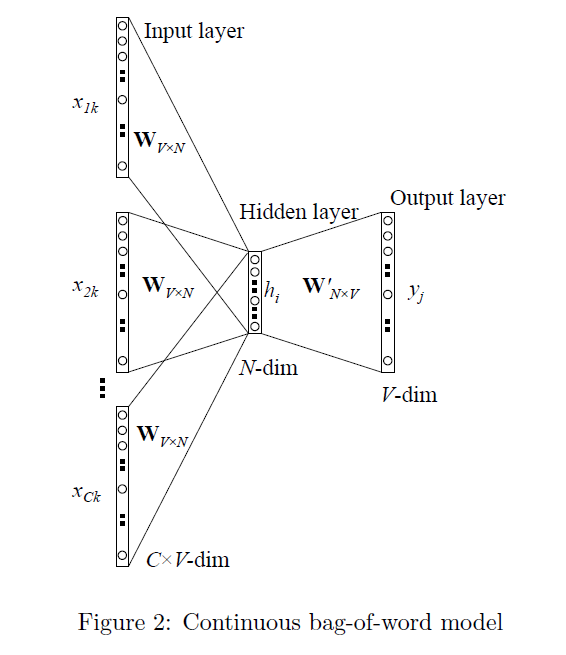
优化方法

为了提高速度，Word2vec 经常采用 2 种加速方式：

1. Negative Sample（负采样）
2. Hierarchical Softmax

COBW





*表示词库的大小；输入向量的大小为，它是这个word的one-hot encoding；神经元表示Hidden Layer，大小为；输出向量表示的是一个概率分布向量，大小和输入向量一致。*

Word2vec的缺点：

1. 由于词和向量是一对一的关系，所以多义词的问题无法解决。
2. Word2vec 是一种静态的方式，虽然通用性强，但是无法针对特定任务做动态优化

### GloVe

根据语料库（corpus）构建一个共现矩阵（Co-ocurrence Matrix），矩阵中的每一个元素代表单词和上下文单词在特定大小的上下文窗口（context window）内共同出现的次数。一般而言，这个次数的最小单位是1，但是GloVe不这么认为：它根据两个单词在上下文窗口的距离，提出了一个衰减函数（decreasing weighting）：用于计算权重，也就是说距离越远的两个单词所占总计数（total count）的权重越小。

GloVe的全称叫Global Vectors for Word Representation，它是一个基于全局词频统计（count-based & overall statistics）的词表征（word representation）工具，它可以把一个单词表达成一个由实数组成的向量，这些向量捕捉到了单词之间一些语义特性，比如相似性（similarity）、类比性（analogy）等。我们通过对向量的运算，比如欧几里得距离或者cosine相似度，可以计算出两个单词之间的语义相似性。

GloVe方法比较简单，全称是Global vectors of word representation。它计算目标target有多少次出现在上下文context中。对于每一个句子，其中的词a出现时词b出现与b出现时a也出现的次数是一致的，因此二者具有对称性。

比如k是单词orange，那么它juice（设为单词i）同时与它出现的概率肯定比它与book（设为单词j）同时出现的概率高。相比单词同时出现的概率，单词同时出现的概率的比率能够更好地区分单词。经过推导，最终得到公式：

https://pic4.zhimg.com/80/v2-8a53d7d041b602eaf510f4fe90ca11cb_720w.jpg

其中V为词库中的所有词，X为词汇共现矩阵，​Xik表示词k出现在词汇i上下文中的次数总和，wi,wk分别为词i和k对应的词向量，f是权重函数，如果ik未同时出现过，则f(Xik)=0，不将其计入目标函数，f还用于均衡高频词于低频词的影响。通过统计词库中各个词组合同时出现的概率训练模型

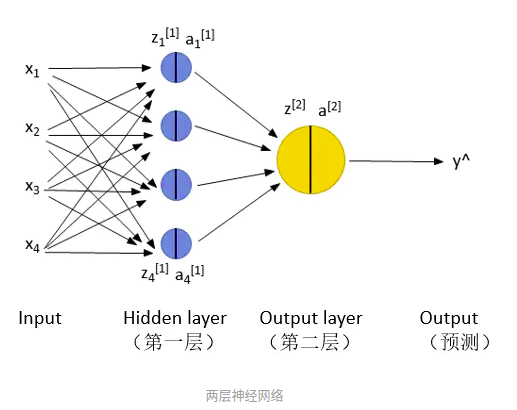
### 与word2Vec 的区别与联系

* skip\_gram方法中最后一步的softmax后，我们希望其周边的词的概率越大越好，这体现在word2Vec 的损失函数上面，我个人感觉这样没有考虑word\_pairs之间的距离因素，而在glove中考虑到了，上述代码中有体现。
* 感觉word2Vec、glove 都是在考虑了共现矩阵的基础上建立模型，只是word2Vec 是一种预测型模型，而glove 是基于计数的模型。
* word2Vec是一种预测型模型，在计算loss时，我们希望其window\_size 内的单词的概率能够尽可能的高，我们可以用SGD不断训练这个前向神经网络，使其能够学习到较好的word\_repesentation。
* 而Glove 呢？是一种基于计数的模型，首先会构造一个很大的共现矩阵，就是上述代码中的cooccurrences 矩阵，其shape为[vocab\_size,vocab\_size]，因此我们需要对其进行降纬，降维后的shape为[vocab\_size,dim]，该矩阵的每一行的向量可以看做该单词的表示，我们可以不断的最小化reconstruction loss 来寻找这样一个矩阵。

### 损失函数

用来评价模型的**预测值**和**真实值**不一样的程度，损失函数越好，通常模型的性能越好。不同的模型用的损失函数一般也不一样。

## 神经网络



 z是x和w、b线性运算的结果，z=wx+b；

 a是z的激活值；

 **下标**的1,2,3,4代表该层的**第i个神经元**（unit）；

 **上标**的[1],[2]等代表当前是**第几层**。

 y^代表模型的输出，y才是真实值，也就是标签

上图中的x1，x2，x3，x4**不是代表4个样本！**  
而**是一个样本的四个特征**（4个维度的值）！  
你如果有m个样本，代表要把上图的过程重复m次

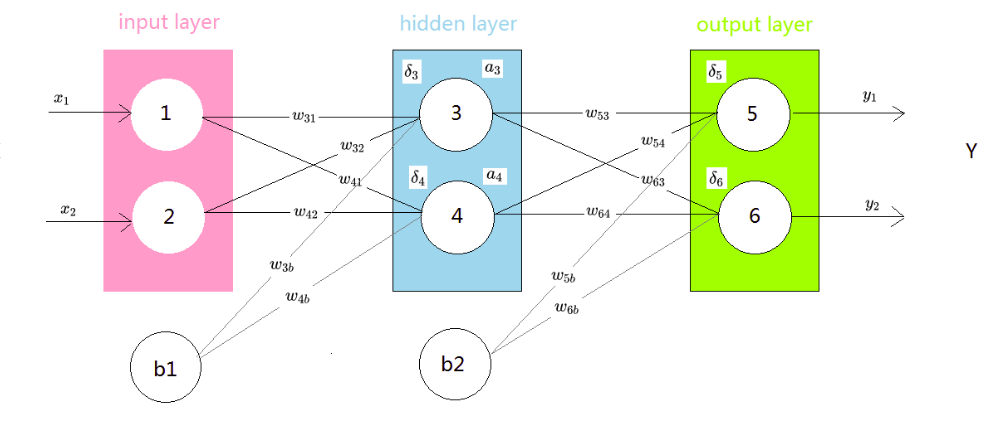
#### 神经网络的“两个传播”：

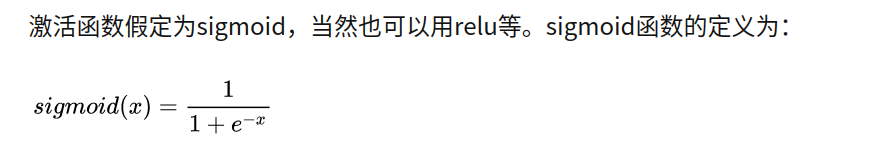
* **前向传播（Forward Propagation）**  
  前向传播就是从input，经过一层层的layer，不断计算每一层的z和a，最后得到输出y^ 的过程，计算出了y^，就可以根据它和真实值y的差别来计算损失（loss）。
* **反向传播（Backward Propagation）**  
  反向传播就是根据损失函数L(y^,y)来反方向地计算每一层的z、a、w、b的偏导数（梯度），从而更新参数。

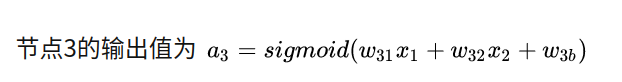
在训练深度学习模型时，正向传播和反向传播相互依赖  
像各种**深度学习框架TensorFlow、Keras**，它们都是**只需要我们自己构建正向传播过程**，**反向传播的过程是自动完成的**

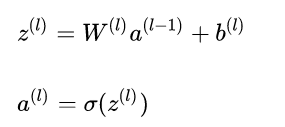
**对DNN的损失函数用梯度下降法进行迭代优化求极小值的过程即为我们的反向传播算法。**

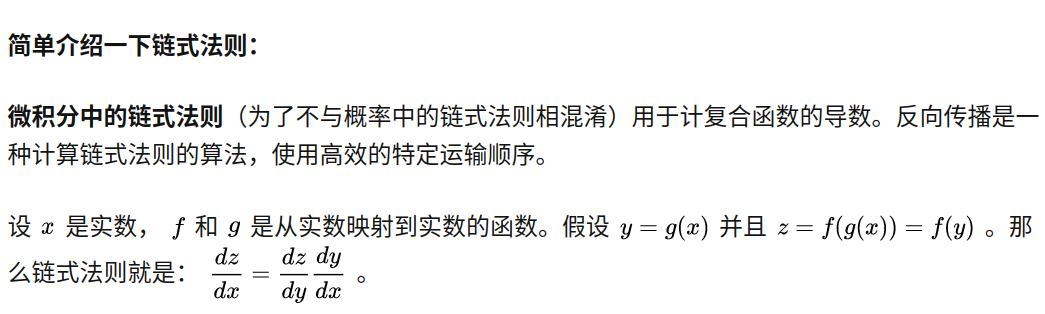
有了DNN反向传播算法，我们就可以很方便的用DNN的模型去解决各种监督学习的分类回归问题



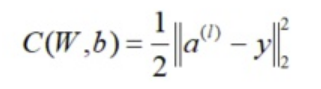




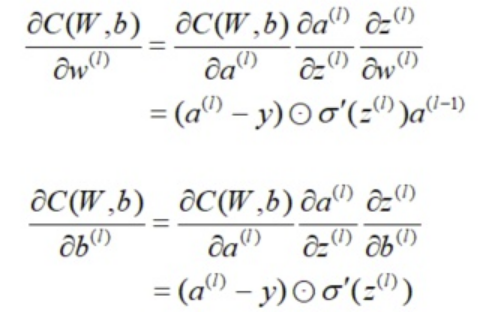




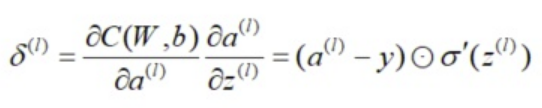
均方误差（MSE）来作为损失函数



(1) 输出层的梯度



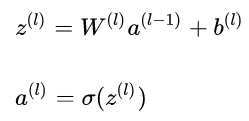
我们注意到在求解输出层梯度的时候有公共的部分，记为



**对反向传播算法的过程进行一下总结：**

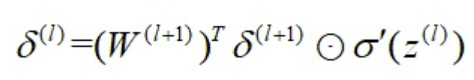
**1. 初始化参数W，b**

**2. 进行前向传播算法计算**，for l=2 to L

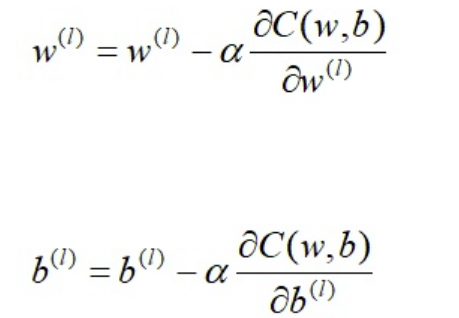


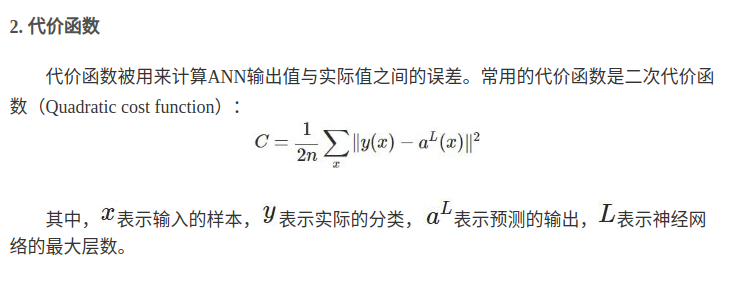
**3. 通过损失函数计算输出层的梯度**

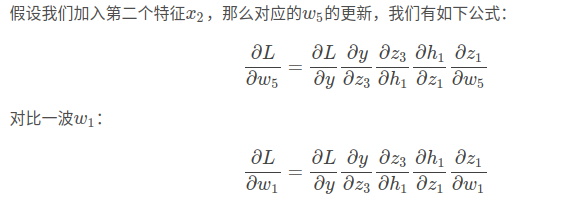
**4. 进行反向传播算法计算**，for l=L-1 to 2









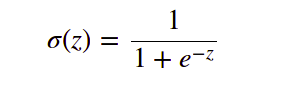


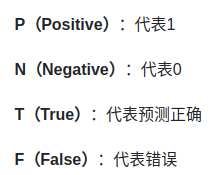
### 激活函数

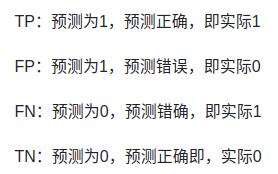
激活函数可以引入非线性因素，解决线性模型所不能解决的问题

神经网络的激活函数**（activation function）**是一群空间魔法师，扭曲翻转特征空间，在其中寻找线性的边界。

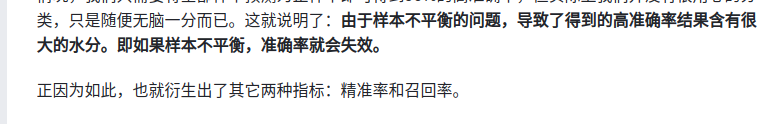
## sigmoid



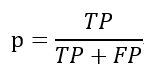




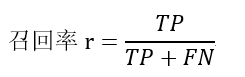




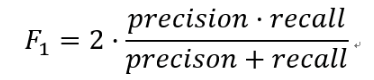
* 查准率（precision），指的是**预测值为1**且**真实值也为1**的样本在**预测值为1的所有样本**



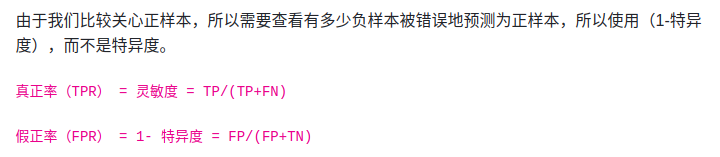
* 召回率（recall），也叫查全率，指的是**预测值为1**且**真实值也为1**的样本在**真实值为1的所有样本**中所占的比例

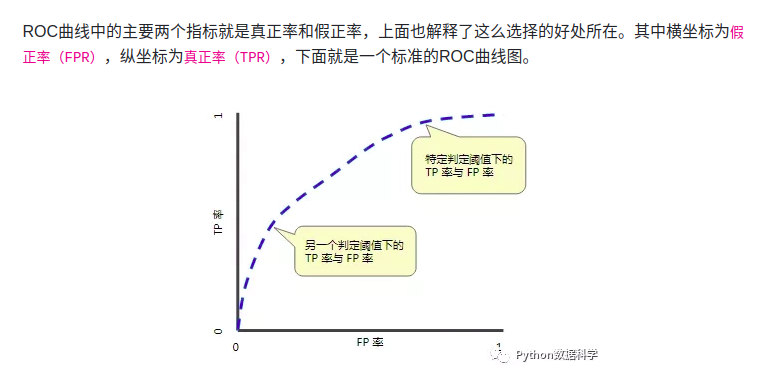


* F1分数（F1-Score），又称为平衡F分数（BalancedScore），它被定义为精确率和召回率的调和平均数。

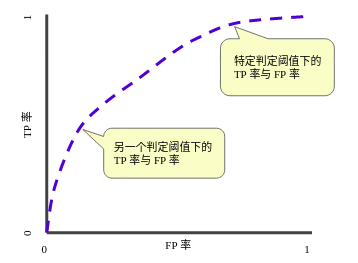


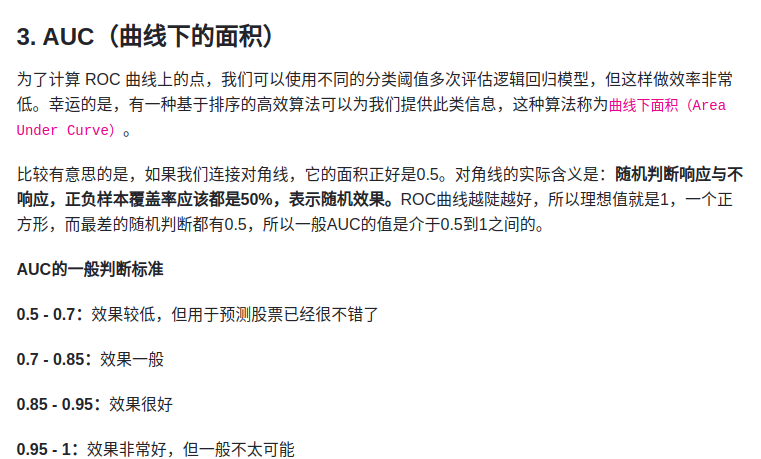
https://blog.csdn.net/zjn295771349/article/details/84961596

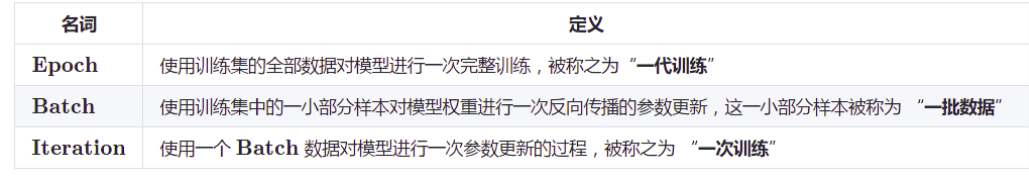




ROC 曲线用于绘制采用不同分类阈值时的 TPR 与 FPR。降低分类阈值会导致将更多样本归为正类别，从而增加假正例和真正例的个数。下图显示了一个典型的 ROC 曲线。







**CIFAR10 数据集有 50000 张训练图片，10000 张测试图片。现在选择 Batch Size = 256 对模型进行训练。**

* 每个 Epoch 要训练的图片数量：50000s
* 训练集具有的 Batch 个数：50000/256=195+1=196
* 每个 Epoch 需要完成的 Batch 个数：196
* 每个 Epoch 具有的 Iteration 个数：196
* 每个 Epoch 中发生模型权重更新的次数：196
* 训练 10代后，模型权重更新的次数：196\*10=1960
* 不同代的训练，其实用的是同一个训练集的数据。第 1代和第 10代虽然用的都是训练集的五万张图片，但是对模型的权重更新值却是完全不同的。因为不同代的模型处于代价函数空间上的不同位置，模型的训练代越靠后，越接近谷底，其代价越小。

ROC: <https://segmentfault.com/a/1190000016686335>

向前传播，向后传播：https://www.jianshu.com/p/765d603c76a0