1.PCA降维的公式及原理分析:

假设对称矩阵A的所有特征值都不一样，那么: -> 对角矩阵

对于矩阵Y，它的协方差

假设Y=QX（此时不考虑降维），则 = =

PCA的本质:让协方差最小，方差最大，这样可以去相关 -> 协方差矩阵的对角元最大，非对角元为0->对角矩阵满足

我们已经得到两个公式: 和 ，假设Q =

可以得到: = ，因为协方差矩阵是对称半正定矩阵(特征值>=0)

所以可以得到:当Q = 时，是对角矩阵，因此如果Q少取几行就实现了降维

2.极大似然估计公式及原理分析:

极大似然估计起源于贝叶斯

假设特征为D，标签为A，P为概率

贝叶斯公式为:P(A|D) =

当样本给定的时候P(D)为常量，同时假定先验概率P(A)近似相等，那么我们可以得到:

P(A|D) = P(D|A)

对于机器学习的意义：将已知特征求标签的概率转化为已知标签求特征的概率，从无监督问题变成了有监督问题

但是极大似然的最大问题是:不同参数的P(A)不一定是相等的，只能说在样本不够大的时候是近似相等的

贝叶斯算法考虑了先验概率P(A),但同时要求先验概率要准确

极大似然的两个假设：

1. 已经发生的事件是独立重复事件，符合同一分布
2. 已经发生的事件是可能性（似然）最大的事件

3.模型调优

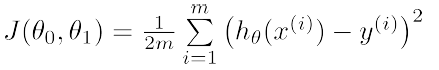
1. 过拟合:找更多的数据来学习,增大正则化系数，减少特征个数（不一定有效）
2. 欠拟合:找更多的特征，减少正则化系数
3. 权重分析:特征对最后结果的影响
4. 模型融合
5. 重复迭代与训练，加大分错样本的权重

4.线性回归

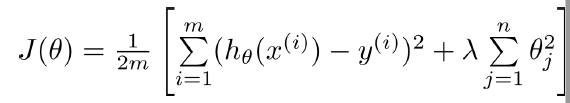
公式:

h(x) =

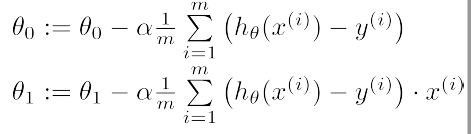
损失函数:



加上L2正则化后：



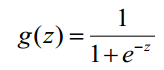
梯度下降:分别对求导，



5.逻辑回归

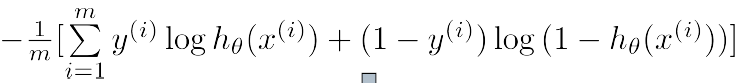
LR实际上是sigmoid+线性回归，用于判断边界，起到分类的作用

sigmoid公式：

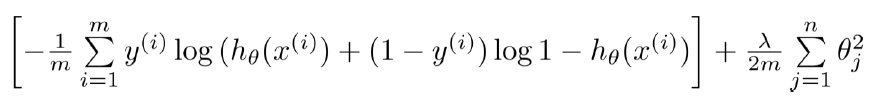


如果输入z = 线性回归，那么就是LR

损失函数: 交叉熵



损失函数+正则化项:



6.贝叶斯

公式：

P(A|D) =

推导:

X,Y同时发生的概率即联合概率P(X,Y) = P(Y|X)P(X) = P(X|Y)P(Y)->P(Y|X) = （贝叶斯）

朴素贝叶斯(NB):条件独立性假设，很“朴素”的近似，即:

P("邱佳豪"|S)=P（“邱”|S）P（“佳”|S）P（“豪”|S）

以垃圾邮件为例:

利用分词向量将“我司发票”拆成一个个分词

C1=P（“垃圾邮件”|“我”，“司”，“发票”） = P(“我”，“司”，“发票”|“垃圾邮件”)P(“垃圾邮件”) | P（“我”，“司”，“发票”）

C2=P ("正常邮件" | “我”，“司”，“发票”) = P（“我”，“司”，“发票”|"正常邮件"）P（"正常邮件"）| P(“我”，“司”，“发票”)

C1/C2可以把分母约掉，再加上独立性假设

C1/C2 = P（“我”|S）P（“司”|S）P（“发票”|S）P（S）除以P（“我”|H）P（“司”|H）P（“发票”|H）P（H），其中H，S分别代表垃圾邮件和正常邮件

这样就化简后，每一项就特别好求，比如

P（“发票”|“垃圾邮件”） = 垃圾邮件中出现发票的次数/垃圾邮件中所有词谱出现的次数总和

由于乘法的交换率， P（“我”|S）P（“司”|S）和 P（“司”|S） P（“我”|S）是一样的，这样一来，贝叶斯就失去了语言里的顺序问题

1.三种模型 2.平滑 3.工程上的灵活应用

P(（"代开普通发票增值税发票”）|S)这个拆开后，发票出现了2次

如果统计与判断的时候都计算重复的次数，称为多项式模型，即P（发票）\*P（发票）

如果训练和判断都只算一次，称为伯努利模型，即P（发票）

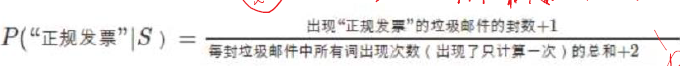
如果训练的时候考虑重复的，判断的时候不考虑，即高斯分布型（假定属性/特征是服从正态分布的）

工程中比理想情况更加复杂，这个时候要考虑平滑：如何在样本数量不是那么充足的情况下，给未出现的新词一个概率，就是平滑最重要的事情

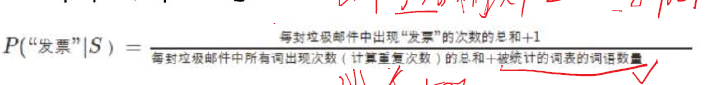
平滑的本质:统一给未出现的词一个概率，同时也要减少已知的概率（所有分子相加等于分母）

如果真实情况中出现了一个测试中没有出现的词就会导致某一项P=0，导致整个结果为0

对于伯努利模型:（简称拉普拉斯平滑）

因为考虑重复所以是封数，分母加2（2效果比1好，未知的概率增加，已知的概率就要减少）

对于多项式模型:（加上被统计词谱的数量，实际上是降低已知词的概率）



多项式模型平滑举例:

训练集1.(a,b,c,a,a) 2.(a,b,b) 3.(b,c,c) 测试集:4.(d,a,c)

P(d)=(0+1)/(11+4)=1/15,其中11是训练集中所有词出现次数的总和，4代表有a，b，c，d四个词

P(a) =(4+1)/15=5/15，4是训练集出现了4次，旧的已有的次数也要加1

P(b) =(4+1)/15 = 5/15

P(c) = (3+1)/15 = 4/15

总和的概率为1

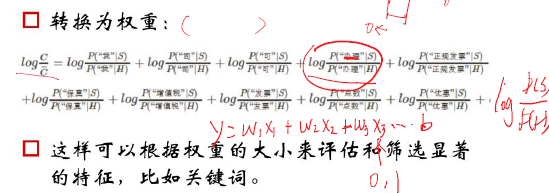
工程上的一些技巧:

1. 取对数

把乘法变成加法，运算速度提高

1. 取权重

这样把两张表变成了一张表，测试的时候只需要取出权重做加法就行



这样一转化之后，你会发现每一个log其实就相当于一个wx，而最后的logP(S)/P(H)就相当于b，于逻辑回归相似

NB与LR的差异

1. 最核心的区别是:LR是Loss最优化求出的权重，而朴素贝叶斯是基于统计，跳过了Loss最优化，直接得出权重
2. NB比LR多了一个条件独立假设
3. LR是判别模型，NB是生成模型

朴素贝叶斯注意点

1.很多特征是连续数值型的，但是它们不一定服从正态分布，一定要想办法把它们变换调整成满足正态分布！！

2.对测试数据中的0频次项，一定要记得平滑，简单一点可以用『拉普拉斯平滑』。

3.先处理处理特征，把相关特征去掉，因为高相关度的2个特征在模型中相当于发挥了2次作用。

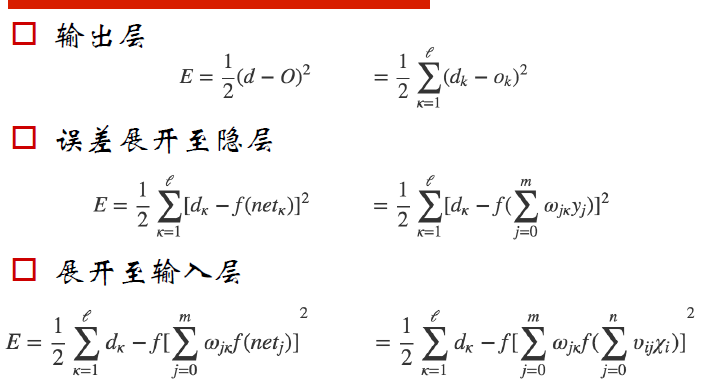
4.一般其他的模型(像logistic regression，SVM等)做完之后，我们都可以尝试一下bagging和boosting等融合增强方法。这对朴素贝叶斯里这些方法都没啥用。原因是这些融合方法本质上是减少过拟合，减少variance的。朴素贝叶斯是没有variance可以减小。

7.神经网络之BP算法

正向传播求损失，反向传播求误差

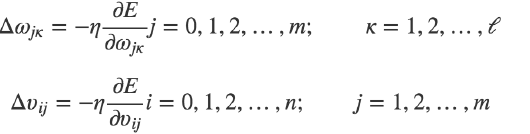
核心:链式求导法则

损失函数:d（真实的结果）与0（输出）都是概率，两者越接近，交叉熵E越小，（实际运算时会记录中间的运算结果，所以不会这么繁琐）



梯度下降:（因为损失函数不是凸函数，存在很多局部变量，如果采用所有样本计算，一个是效率低，一个是容易陷入局部最优）

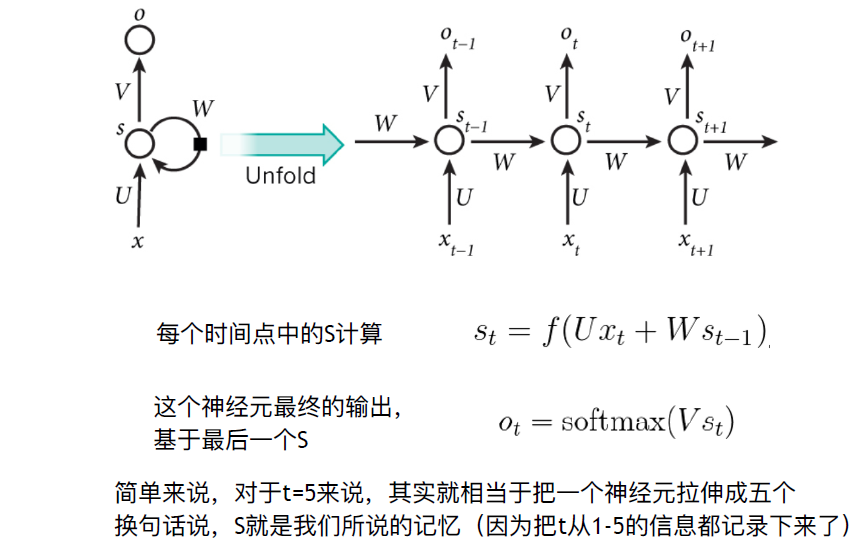
SGD,随机梯度下降，一般的梯度下降需要求出所有样本当前应该朝着哪个方向走，随机梯度下降只是找一部分样本来计算朝哪个方向走



8.神经网络之RNN

RNN用于生成模型，CNN用于分类模型上，对于分类问题可以容忍语病的存在，对于生成模型是不能容忍的(CNN只是看一个窗口)

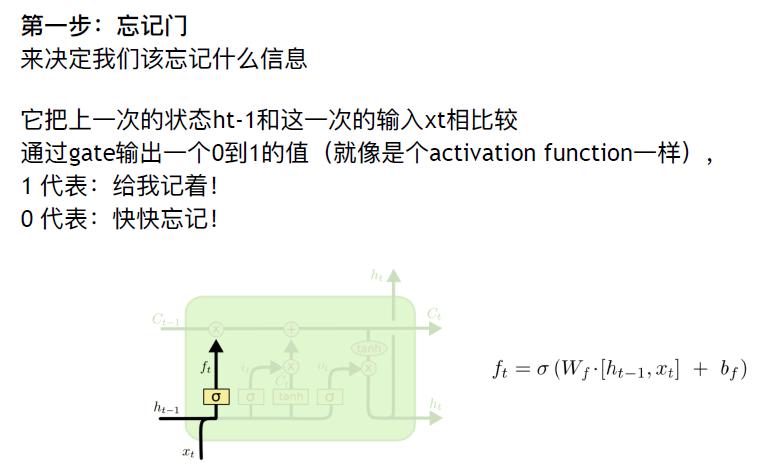
记得上下文前后之间的关系，比如:这顿饭真好--🡪可以根据前面5个预测得到下一个字是“吃”

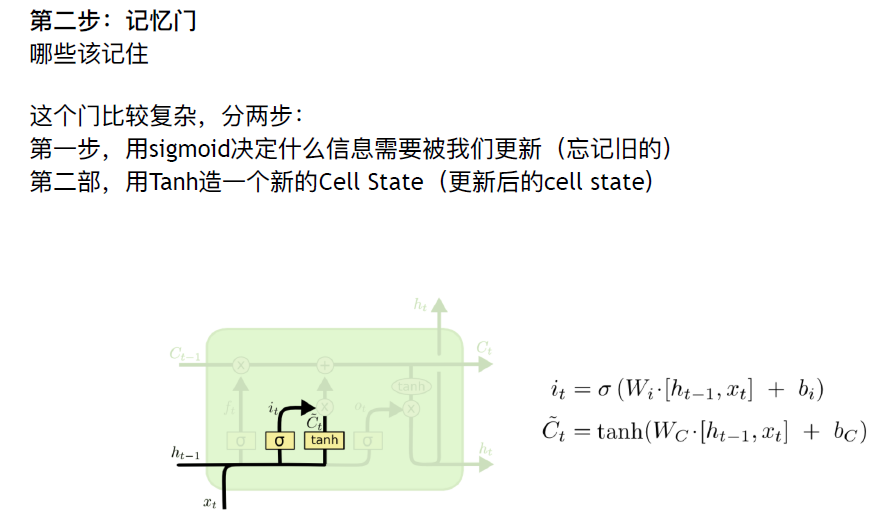


9.神经网络之LSTM

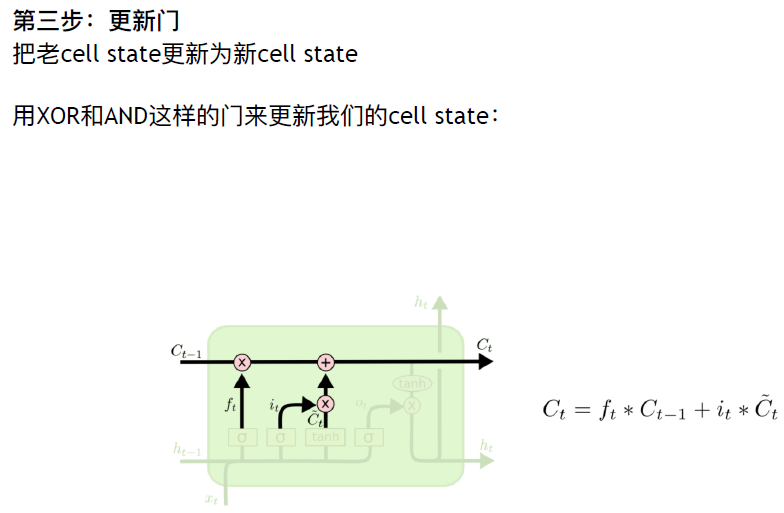
RNN是将所有的记忆全部拿来，而LSTM选择该记住的记住，不改记住的就忘记，这样LSTM就更加的高效

如何预测下一个字母—>如何预测下一个word->如何预测下一句话->预测下一个图片/音符

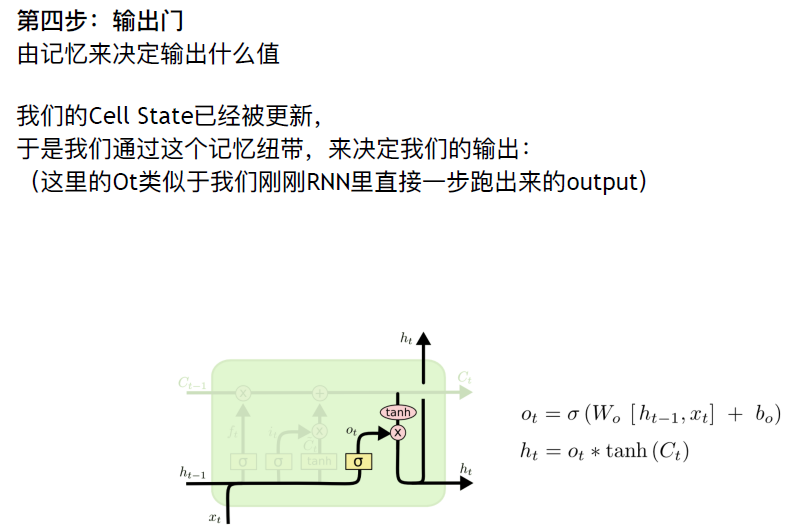




记忆门的第一步就是忘记门

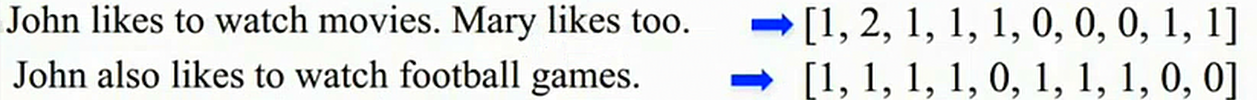


F就是前面的忘记门，i和C就是前面的记忆门



10.NLP之Word2Vec

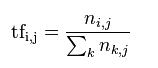
1.One-hot:字典包含n个词，每个词有唯一的索引，有多少词，向量空间就有多大



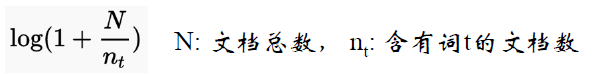
2.TF-IDF: 倾向于过滤掉常见的词语，保留重要的词语

比one-hot能显示词的重要程度

TF词频，一个词在本文档中出现的频率



IDF: **逆向文件频率**

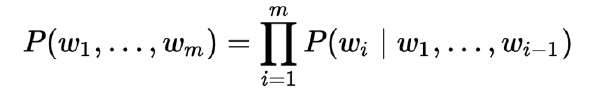


3.Binary weighting:不管出现的次数，多用于短文本

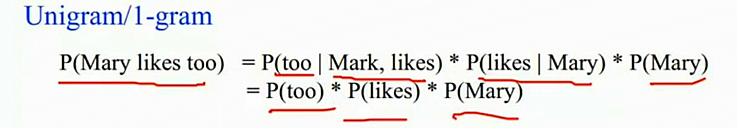


4.N-gram:考虑了词的顺序，但是词表会膨胀

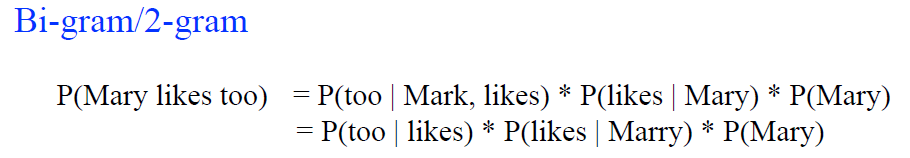
5.语言模型:在n-gram的基础上进一步生成，代表一句话(词组合)出现的概率



一元语言模型:首先是P(Mary),然后是Mary后面出现Like的概率，然后是Mary Likes后面出现Too的概率，第二个等式是因为这是一个一元语言模型，彼此之间是不依赖的，所以有第二个等式



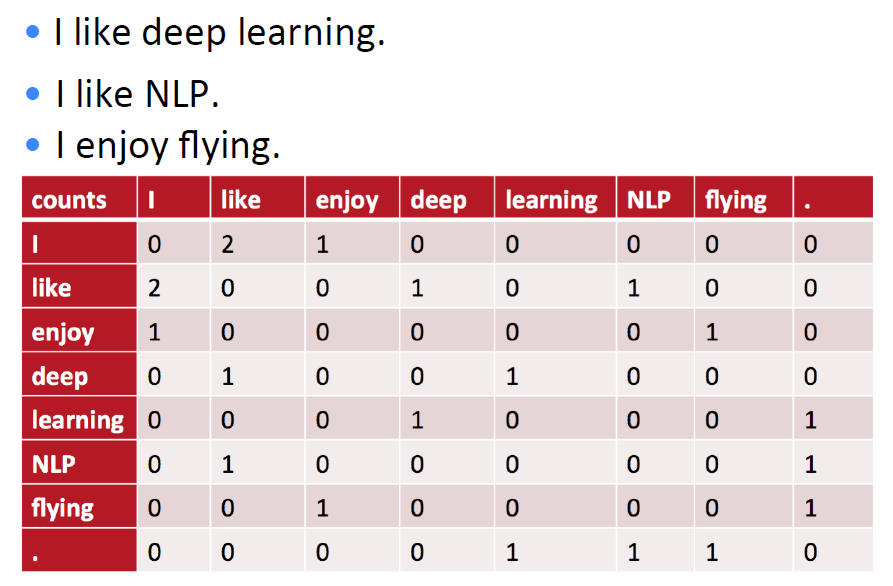
二元语言模型



6.分布式表示:用一个词附近的词来表示该词

以上离散型无法接近不同向量间的关系

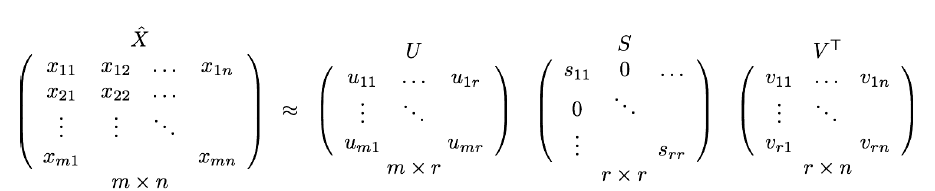
由此出现共现矩阵:



首先I看周围有2个like，一个enjoy，所以出现第一行的数字，其余类似

7.SVD降维

共线矩阵的维数非常大，需要降维

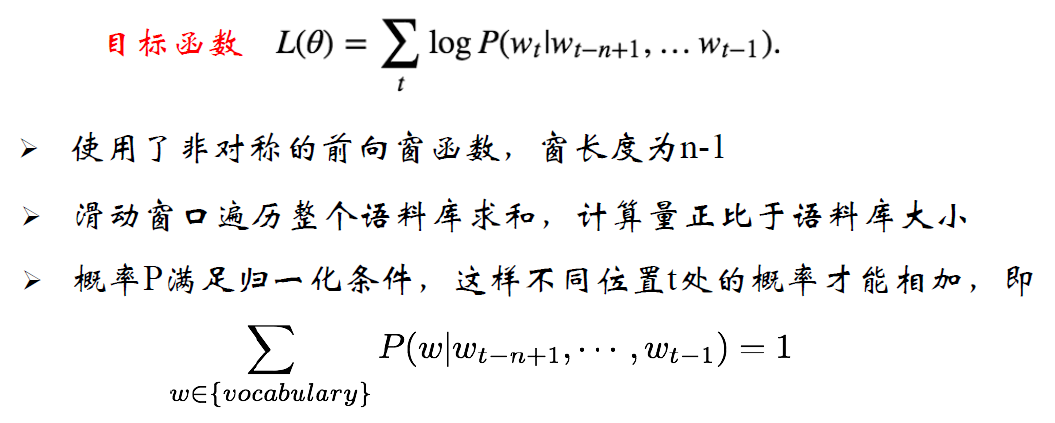


计算的复杂度是O(n^3),对于大型的语料库会计算量非常大并且难以加入新的词

8.NNLM:神经网络语言模型(Word2Vec是它工程上的简化版本)

直接从语言模型出发，将模型最优化过程转化为求词向量表示的过程

根据前n-1个词求第n个词的概率。所谓的窗口应该是指每次只预测一个指定数量的词



假设词典维数V=8w,投影的维度为D=300，

以n-1大小的窗口去滑动，预测第n个字，以“我是中国人”为例

我，是，中，国，均以one-hot形式表示的向量为1\*8W

第一层网络为:Projection Layer，投射层，权重矩阵为C，维度为D\*8W，我们将输入1\*8W进行转制，与C相乘，就相当于取出了C的某一列(D\*1)

然后将这四个结果进行拼接，就得到了一个4D\*1的向量

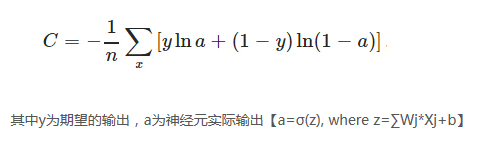
第二层为隐层:假如神经元为100，那么就是4D\*1与这100个神经元做一个全连接

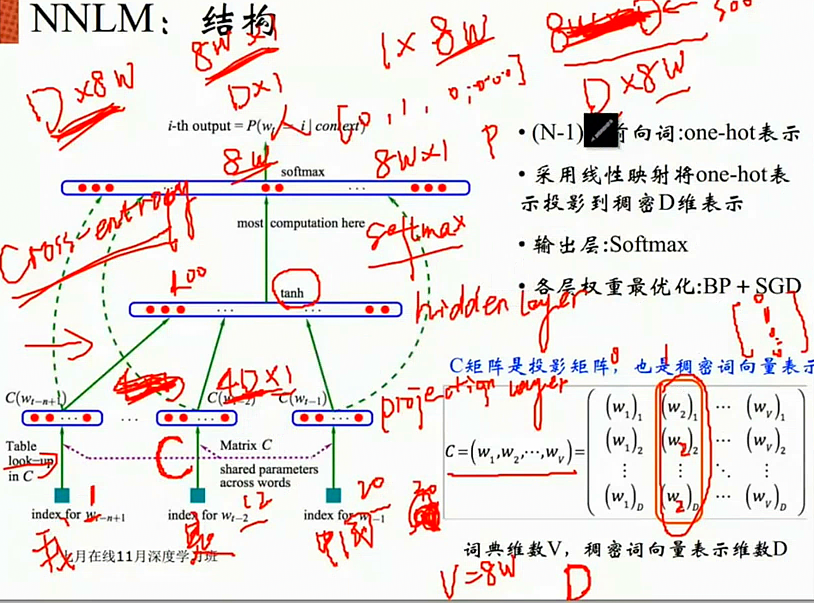
然后与激活函数tanh(双曲正切)

第三层为输出层:与8w个词条做一个softmax，预测下一个词是这8w个词中的哪一个

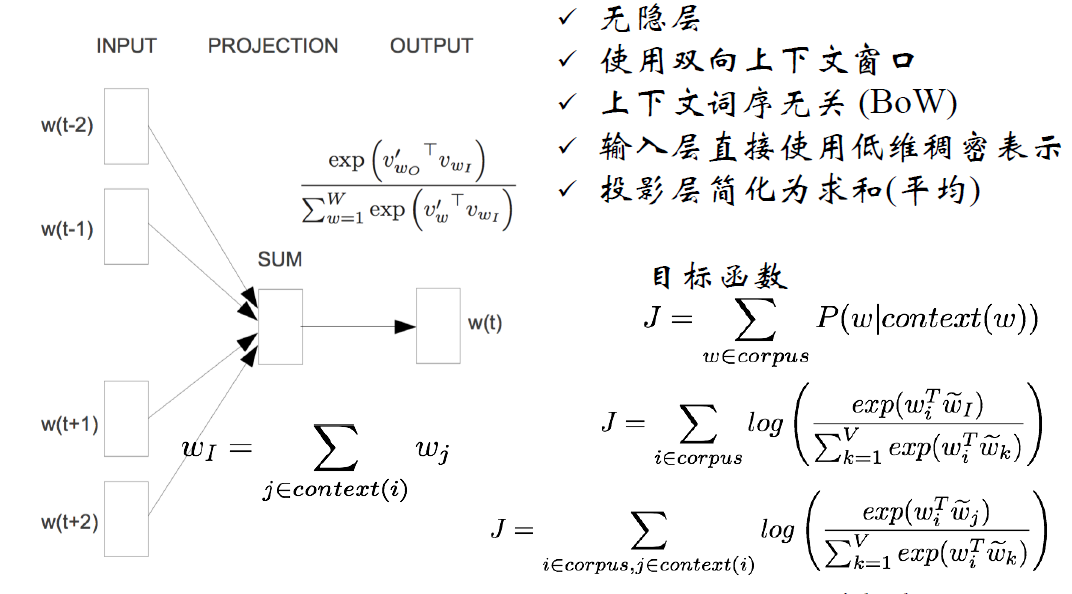
与真实的值，softmax的矩阵维度为:4D\*8W

损失函数为交叉熵即预测出来的概率（8w个概率组成的）与真实的“人”这个词的one-hot进行求交叉熵:





9.word2vec之CBOW(连续词袋)



第一个等式J中context(就是用邱豪去预测佳)，第二个等式是归一化

窗口:左侧为n，右侧为n，中间为1，总长为2\*n+1

1.对每一个word初始化一个词向量(D\*1)（CNLM是one-hot与C相乘，得到D\*1）

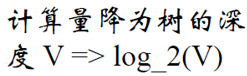
2.求和（CNLM里是拼接4D\*1），此处仍然是D\*1

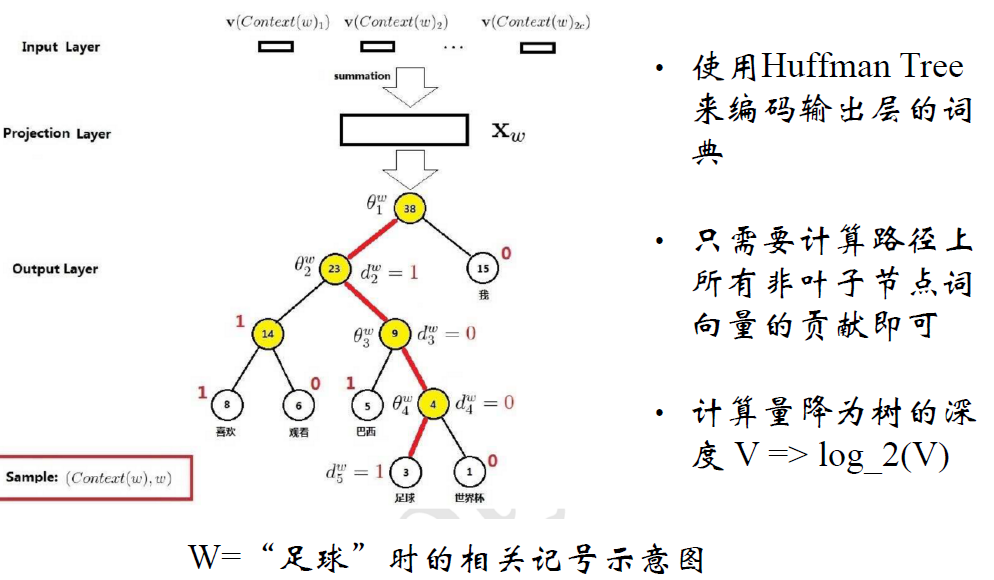
3.用周边的词去预测这个位置的词(CNLM是用n-1去预测第n个)

4.去掉了隐层

5.得到的D\*1与8w做softmax，但是8w仍然需要做压缩:

(1)层次softmax:Huffman Tree，不同的频次有不同的权重，离根节点越近的频次越高

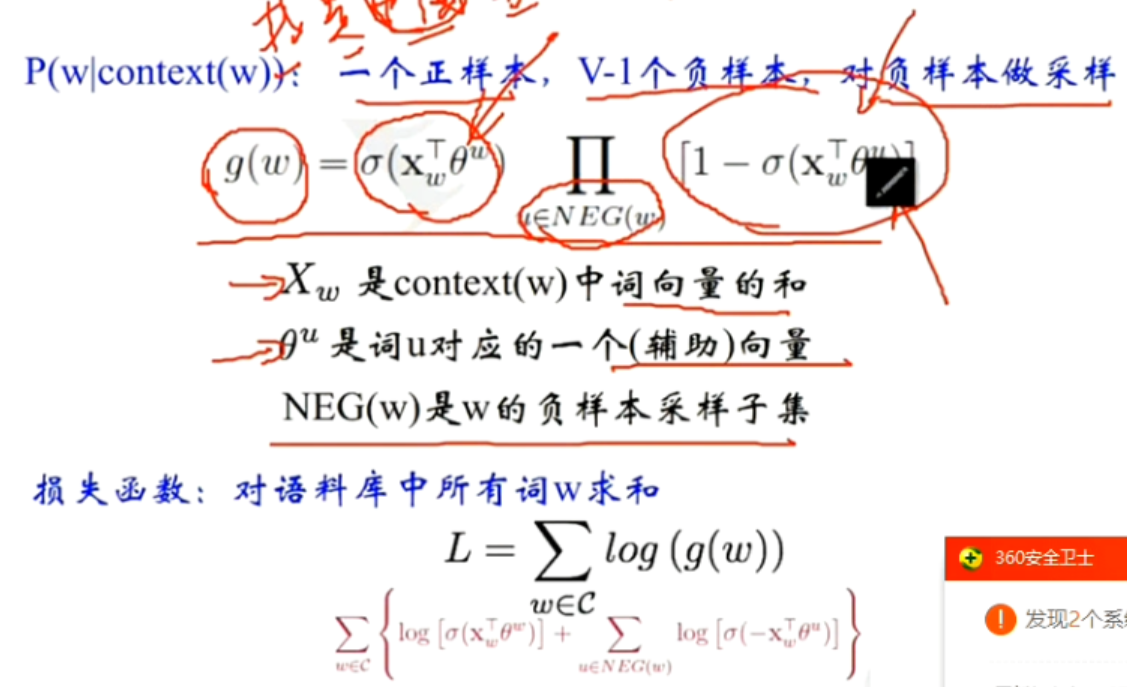
“我喜欢观看巴西足球世界杯”，预测巴西的时候，只需要做若干的二分类(原本要计算8w的概率，现在只需要顺着这条路径进行查找，降低计算量)，同时每一个非叶子节点（黄色部分）会给一个权重向量，那么本节点的运算为sigmoid(w\*),每次这样查找就是若干个进行相乘，可以取log，变成加法，那么它就是目标函数J（越大越好，这样到巴西的时候概率最大），损失函数为SGD，优化方法为:梯度上升(取负号就是梯度下降)，这样softmax的计算由D\*8w，



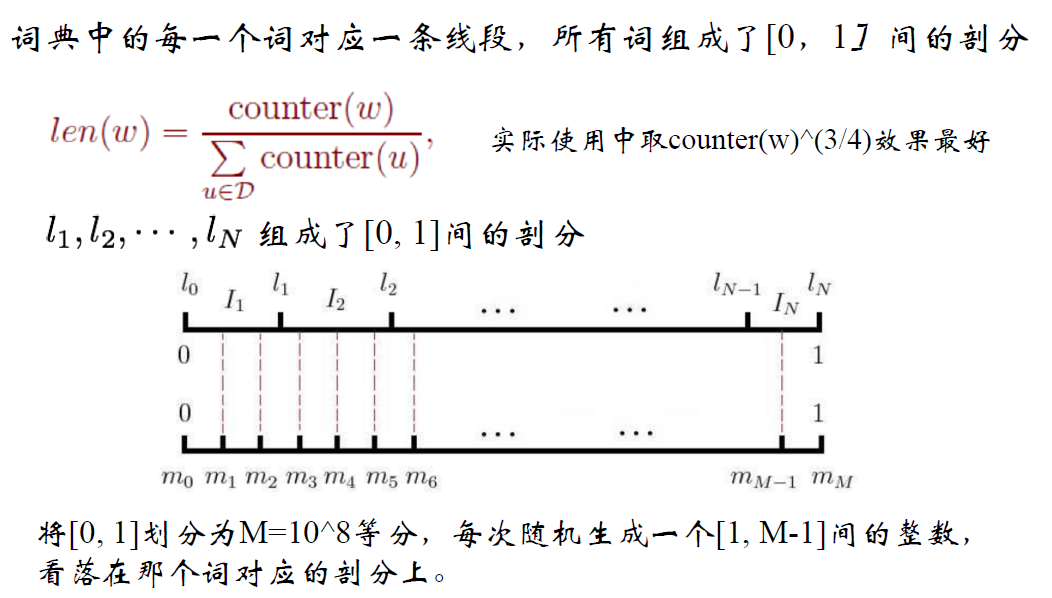
10.word2vec之负例采样

一个正样本，剩余的是负样本(8w-1)

g(w)是取这个正样本的概率和不取这个正样本（在负样本采样集中，并不是完整的8w-1）的概率的乘积

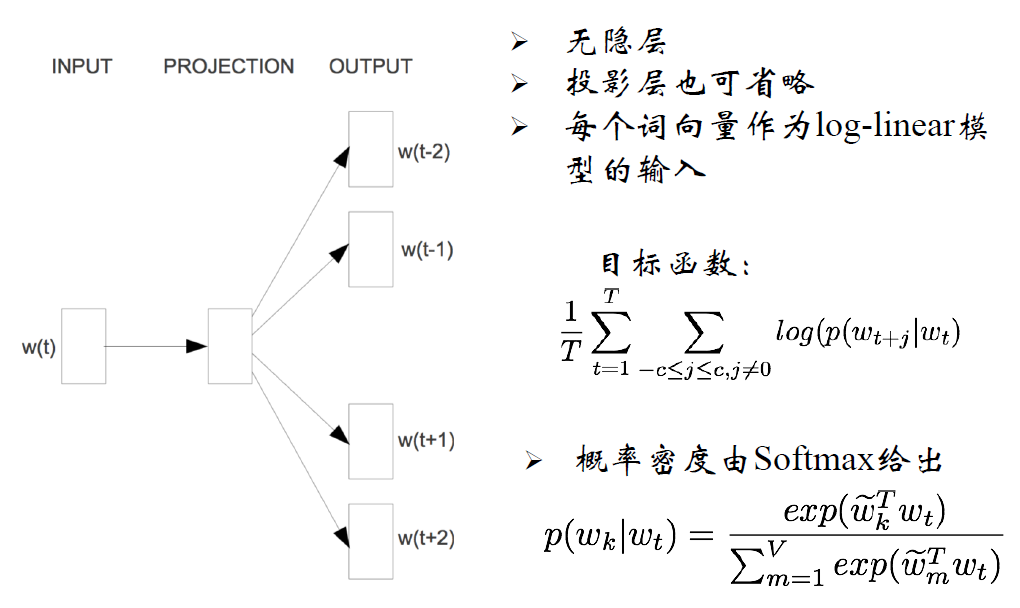


那么如何对负样本进行采样(如果刚好踩到了正例可以忽略):



11.word2vec之Skip-Gram模型

由国去推测周边的人和中



12．Word2Vec的缺陷

对多义词无法很好的理解和表示，如苹果和华为，不知道它是说手机还是说水果

13.工程上数据量很大时用Skip-Gram，否则用CBOW

11.NLP总结

1.一条典型的文本预处理流水线

一般分词往往不是一步到位的，比如社交语音的特殊处理

语意复杂度的最小化:归一化和stopword都是为了实现这个目的

