Data Mining and Optimization

Lecture 7: Text Vectorization

Liu Yang

Nanjing University

Spring, 2025

Table of Contents

文本向量化概述

- 文本向量化就是将文本表示成一系列能够表达文本语义的向量。
- 大部分文本向量化是通过词向量化实现的，也有一部分文本向量化以句子或文章作为基本单位。

Table of Contents

词袋模型

- 词袋(bag of words)模型是最早的以词语为基本处理单位的文本向量化方法。
- 考虑如下两个简单文本：
  - John likes to watch movies, Mary likes too.
  - John also likes to watch football games.
- 基于两个文本中出现的单词，构建如下词典：

$$\{john:1, likes:2, to:3, watch:4, movies:5, also:6, football:7, games:8, Mary:9, too:10\}$$

- 词典包含10个单词，每个单词有唯一的索引，则上述文本可采用如下10维向量表示：
  - [1,2,1,1,1,0,0,0,1,1]
  - [1,1,1,1,0,1,1,1,0,0]

TF-IDF模型

- TF-IDF模型也是一种文本向量化方法，是对词袋模型的改进，不仅考虑了每个单词在文本中出现的次数，也考虑了单词在整个语料库中的使用频率。

词袋模型与TF-IDF模型存在的问题

词袋模型与TF-IDF模型简单易行，但是存在如下三方面的问题：

- 维度灾难(curse of dimensionality)：如果词典中包含10000个单词，那么每个本文需要用10000维的向量表示，除了文本中出现的词语位置不为0，其余9000多的位置均为0，如此高纬度的向量会严重影响计算速度；
- 无法保留词序信息：使用词袋模型与TF-IDF模型生成的向量与原文本中单词出现的顺序没有关系；
- 语义鸿沟：考虑如下两个文本：
  - John likes Mary.
  - Mary likes John.

  两个文本的词袋模型与TF-IDF模型表示法是一样的，但语义完全不一样。

Table of Contents

神经网络语言模型

- 神经网络语言模型(Neural Network Language Model,NNLM)是基于分布假说(distributional hypothesis)——上下文相似的词，其语义也相似。
- 神经网络词向量模型就是根据上下文与目标词之间的关系进行建模。
- 主流的NNLM包含三种模型：
  - CBOW模型
  - Skip-gram模型

词语的分布式表示

- 词语的分布式表示：利用低维连续的实数向量表示一个词语，使得语义相近的词在实数向量空间中也相近。
- 比如下图我们将词汇表里的词
  用"Royalty","Masculinity","Femininity"和"Age"4个维度来表示，King这个词对应的词向量可能是(0.99,0.99,0.05,0.7)。当然在实际情况中，我们并不能对词向量的每个维度做一个很好的解释。

神经网络语言模型

- 如何得到合适的词向量呢？一个很常见的方法是使用神经网络语言模型。
- 采用的方法一般是一个三层的神经网络结构(当然也可以多层)，分为输入层，隐藏层和输出层(softmax层)。
- 根据输入输出的定义方式，又可以进一步细分为CBOW(Continuous Bag-of-Words) 与Skip-Gram两种模型。

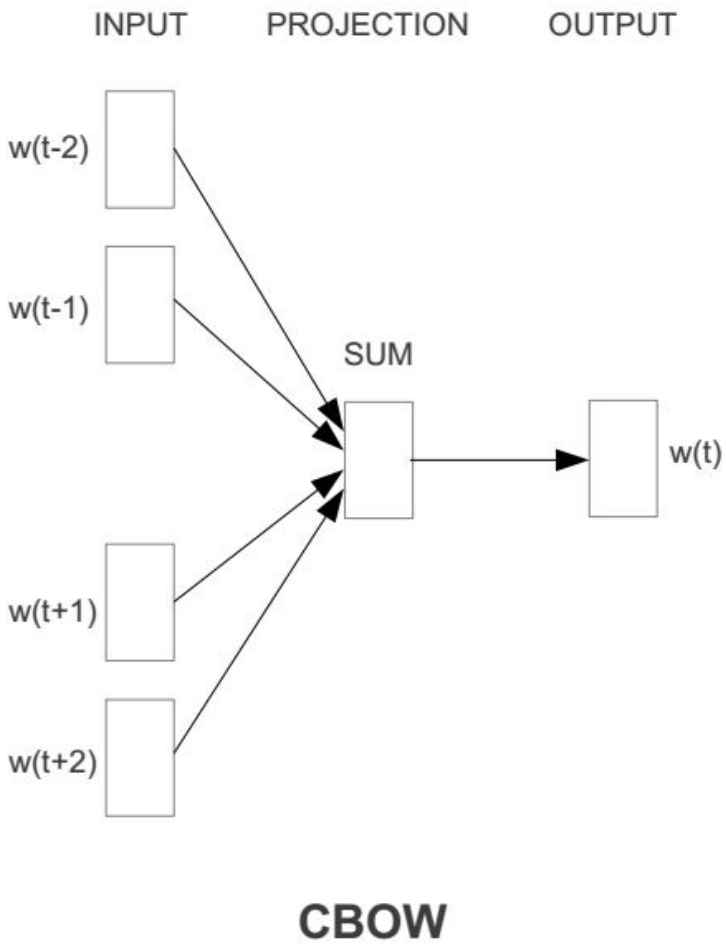
softmax函数

- softmax从字面上来说，可以分成soft和max两个部分。max故名思议就是最大值的意思。softmax的核心在于soft。很多场景中需要我们找出数组所有元素中值最大的元素，实质上是求hardmax，最大的特点就是只选出一个最大的值，即非黑即白。
- 对于文本分析来说，我们更期望得到文本属于某个类别的概率值，所以此时用到了soft的概念，softmax的含义就在于不再唯一确定某一个最大值，而是为每个输出结果都赋予一个概率值，表示属于每个类别的可能性。
- 下面给出softmax函数的定义(以第i个节点输出为例)：

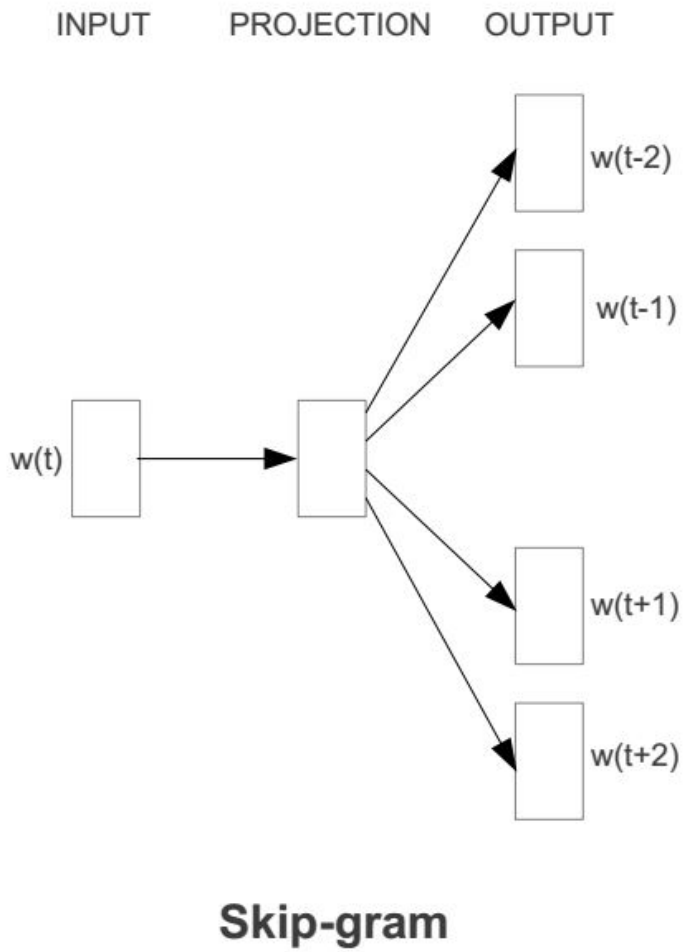
$$softmax(a_1, a_2, \cdots, a_C)_i = \frac{e^{a_i}}{\sum_{j=1}^{C} e^{a_j}}.$$

其中，a_i为第i个节点的输出值，C为输出节点个数，即类别个数。通过softmax函数就可以将多分类的输出值转换为范围在$(0, 1)$的概率分布。

神经网络语言模型



CBOW



Skip-gram

CBOW模型

- CBOW模型的输入是某一个特征词的上下文相关的词对应的词向量，而输出就是这特定的一个词的词向量。
- 比如下面这段话，上下文大小取值为4，特定的这个词是"Learning"，也就是输出词向量,上下文对应的词有8个，前后各4个，这8个词是模型的输入。
- 由于CBOW使用的是词袋模型，因此这8个词都是平等的，也就是不考虑其与特征词之间的距离大小，只要在定义的上下文之内即可。

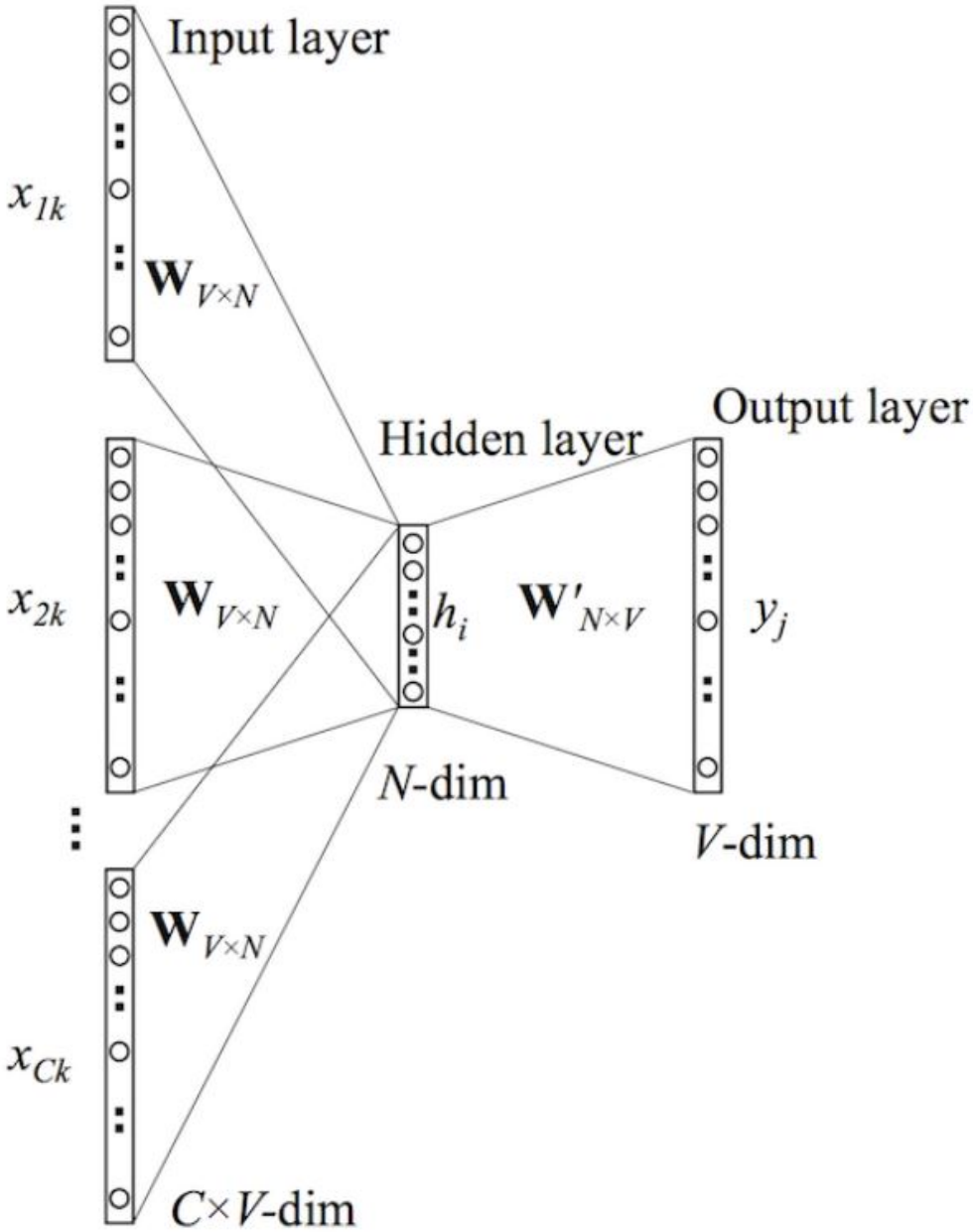
CBOW模型

- 具体而言，在这个例子里，输入是8个词向量，输出是所有词的softmax概率(训练的目标是期望训练样本特定词对应的softmax概率最大)。
- 对应的CBOW神经网络模型输入层有8个节点，输出层有词汇表大小个节点，隐藏层的神经元个数可以自己指定。
- 通过最小化损失函数可以求得神经网络模型的参数，同时得到所有的词对应的词向量。
- 当需要求出某8个词对应的最可能的输出中心词时，可以利用训练得到的神经网络模型并通过softmax激活函数找到概率最大的词。

Skip-Gram模型

- Skip-Gram模型和CBOW的思路是相反的，即输入是一个特定词的词向量，而输出是特定词对应的上下文词向量。
- 还是上面的例子，如果上下文大小取值为4，特定词"Learning"是输入，而这8个上下文词是输出，即softmax概率排前8的8个词。
- 对应的Skip-Gram神经网络模型输入层有1个节点，输出层有词汇表大小个节点，隐藏层的神经元个数可以自己指定。
- 通过最小化损失函数可以求得神经网络模型的参数，同时得到所有的词对应的词向量。
- 当需要求出某个词对应的最可能的8个上下文词时，可以利用训练得到的神经网络模型得到概率大小排前8的对应词即可。

词向量的生成过程——CBOW模型



词向量的生成过程——CBOW模型

- 假设词表中有10000个词，那么每个词根据词袋模型编码后，就是一个10000维的向量，其中9999个都是0，只有一个是1。
- CBOW模型输入层是由词袋模型编码的上下文$\{x_1, x_2, \cdots, x_C\}$组成，其中窗口大小为$C$，词汇表大小为$V$。隐藏层是$N$维的向量。输出层也是被词袋模型编码的输出单词$y$。
- 被词袋模型编码的输入向量通过一个$V \times N$的权重矩阵$W_{V\times N}$连接到隐藏层；隐藏层通过一个$N \times V$的权重矩阵$W'_{N\times V}$连接到输出层。

词向量的生成过程——CBOW模型

- 假设输入与输出权重矩阵——$W_{V\times N}$和$W'_{N\times V}$已知。
- 第一步：计算隐藏层h，如下：

$$h = \frac{1}{C} W_{V\times N}^{T} \left(\sum_{i=1}^{C} x_i \right)$$

即输入向量的加权平均。

词向量的生成过程——CBOW模型

- 第二步：计算在输出层每个节点的输入，如下：

$$u_j = v'^{T}_{W_j} h$$

其中，v'_{W_j}为输出矩阵$W'_{N\times V}$的第j列。

词向量的生成过程——CBOW模型

- 第三步：通过softmax函数计算输出层的输出，如下：

$$y_j = P(w_j|w_1, \cdots, w_C) = \frac{exp(u_j)}{\sum_{j'=1}^{V} exp(u_{j'})}$$

CBOW模型的权重学习

- 为了学习权重矩阵$W_{V\times N}$和$W'_{N\times V}$，先给这些权重矩阵赋予一个任意的初始值。
- 当窗口大小为C时，CBOW模型需要最大化给定背景词生成任一中心词的概率：

$$\prod_{i=1}^{N} P(w_i|w_1,\cdots,w_C). \tag{1}$$

CBOW模型的权重学习

- 最大化(1)与最小化以下损失函数等价：

$$
\begin{aligned}
& -\sum_{i=1}^{N} \ln(P(w_i|w_1,\cdots,w_C)) = -\sum_{i=1}^{N}\left(u_i - \ln\left(\sum_{j'=1}^{V} exp(u_{j'})\right)\right) \\
= & -\sum_{i=1}^{N} u_i + N\ln\left(\sum_{j'=1}^{V} exp(u_{j'})\right) \\
= & -\sum_{i=1}^{N} v'^{T}_{W_i} h + N\ln\left(\sum_{j'=1}^{V} exp(u_{j'})\right)
\end{aligned}
$$

- 通过梯度下降算法可求出权重矩阵$W_{V\times N}$和$W'_{N\times V}$，其中，$W_{V\times N}$的每一行就是词表中每个词的词向量。

Table of Contents

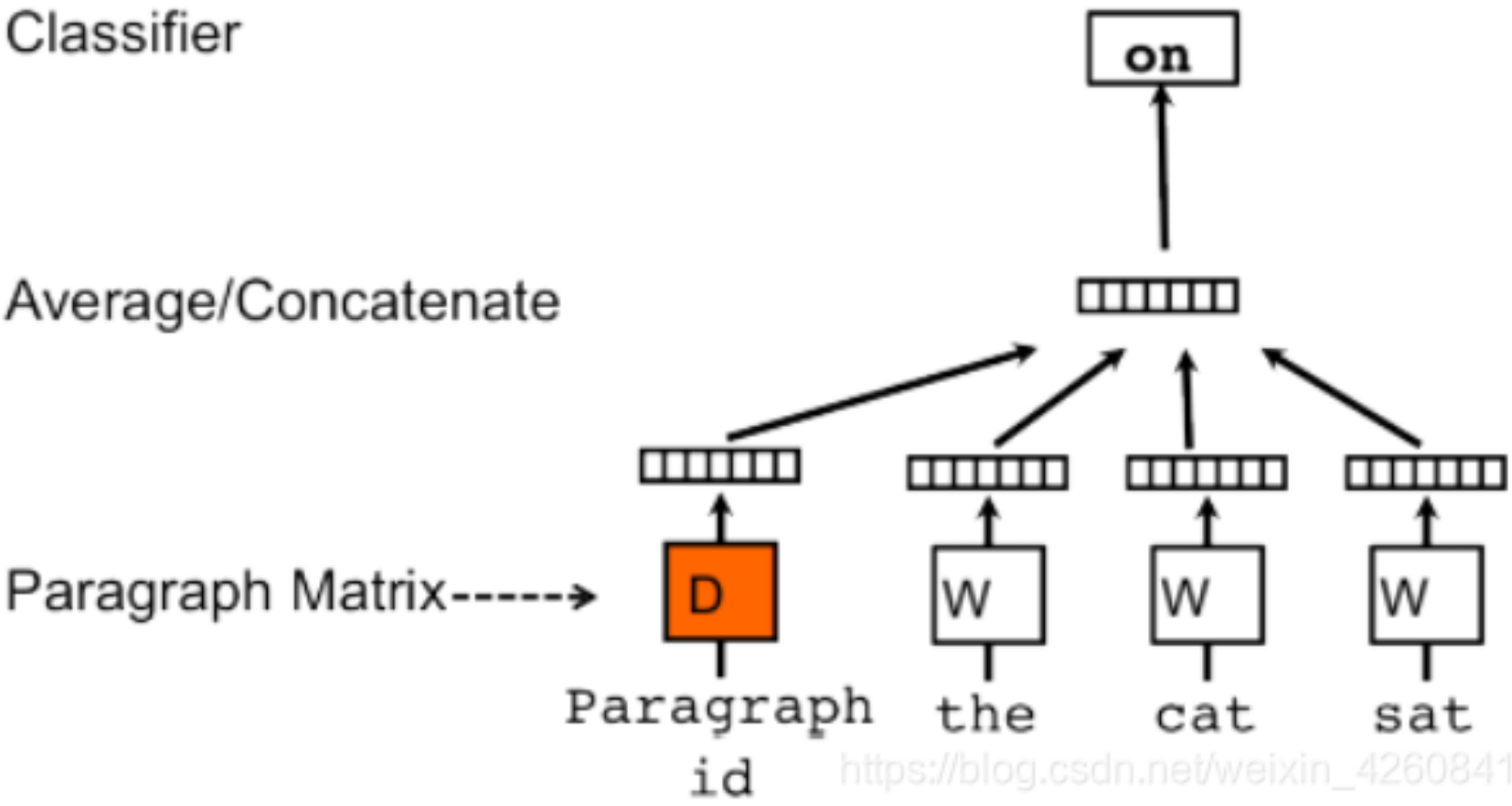
文档向量化

- 词向量用于表示一个词，如果想表示一个文档，需要使用文档向量。
  - 一个文档可以是一个句子，也可以是一组句子。
  - 一个文档可以由评论、推文、脚本或论文组成，长度从几个字到数十万字不等。
- 最简单的文档向量为所有组成词向量的平均值，并用平均值向量表示该文档。

Doc2Vec

- Doc2Vec或者叫做Paragraph2Vec，Sentence Embeddings，是一种非监督式算法，可以获得不同长度句子/段落/文档的固定长度向量表示，是word2vec的拓展。
- Doc2Vec包括两种训练算法：
  - 分布记忆的段落向量（Distributed Memory Model of Paragraph Vectors, PV-DM）：给定文本向量和上下文的情况下预测某个单词的概率，类似于Word2Vec中的CBOW模型。
  - 分布词袋版本的段落向量（Distributed Bag of Words version of Paragraph Vector，PV-DBOW）：仅给定文本向量的情况下预测文本中一组随机单词的概率，类似于Word2Vec中的Skip-gram模型。

PV-DM



https://blog.csdn.net/weixin_42608414

PV-DM的训练过程

- 在PV-DM中，每一句话用唯一的向量来表示，用矩阵D的某一行来代表。每一个词也用唯一的向量来表示，用矩阵W的某一行来表示；
- 每次从一句话中滑动采样固定长度的词，取其中一个词作中心词，其他的作为输入词。输入词对应的词向量和本句话对应的句子向量作为输入层的输入，将本句话的向量和本次采样的词向量相加求平均或者累加构成一个新的向量h，进而使用这个向量h预测此次窗口内的预测词（softmax层）；
- PV-DM相对于CBOW不同之处在于，在输入层增添了一个新的句子向量，可将其看作另一个词向量，它扮演了一个记忆角色。在CBOW中，每次训练只会截取句子中一小部分词训练，而忽略了除了本次训练词以外该句子中的其他词，这样仅仅训练出来每个词的向量表达，句子只是每个词的向量累加在一起取平均，忽略了文本的词序问题。

PV-DM的训练过程

- PV-DM弥补了这方面的不足，它每次训练也是滑动截取句子中一小部分词来训练，但句子向量在同一个句子的若干次训练中是共享的，所以同一句话会有多次训练，每次训练中输入都包含该句子向量，可将其看作是句子的主旨；
- PV-DM假设不仅上下文会影响一个词语出现的概率，词语所在的句子也会影响其概率。这样每次训练过程中，不光是训练了词，得到了词向量。同时随着一句话每次滑动取若干词训练的过程中，作为每次训练的输入层一部分的共享句子向量，该向量表达的主旨会越来越准确。
- 训练完了以后，就会得到训练样本中所有的词向量和每句话对应的句子向量，那么PV-DM是怎么预测新的句子的句子向量呢？其实在预测新的句子的时候，还是会将该句子向量随机初始化，放入模型中再重新根据随机梯度下降不断迭代求得最终稳定下来的句子向量。不过在预测过程中，模型里的词向量、投影层到输出层的softmax权重参数是不会变的，这样在不断迭代中只会更新句子向量，其它参数均已固定，只需很少的时间就能计算出待预测的句子向量。

gensim参数说明

- 训练算法：PV-DM(dm=1)；PV-DBOW(dm=0)；
- 词向量上下文距离window：通常设置为5，如果文本较短，可设置为3；
- 句向量维度vector_size=200/300；
- 最小词频min_count=1，即所有出现过的词均纳入训练；
- 其余参数通常为默认值。

Doc2Vec的应用

- 相似性：使用学习得到的文档向量比较文本的相似性，例如，法律AI软件可以使用文档向量查找类似的法律案例。
- 推荐：在线杂志可以根据用户已经阅读过的文章推荐类似的文章。
- 预测：文档向量可作为机器学习算法的输入，建立预测模型。