# Neural Networks+and Neural Language Models（神经网络和神经语言模型）

## Abstract

起源：

起源于McCulloch-Pitts神经元模型（1943年），这是一个简化的生物神经元模型。但现代语言处理中使用的神经网络已经不再依赖于早期的生物学启发。

现代神经网络的基本组成:

由许多小的计算单元（神经元）组成，每个单元接收一个输入向量，产生一个输出值。

这些计算单元按层排列，数据在层间前向传播，因此被称为前馈网络。

现代神经网络通常具有多层，这就是“深度学习”名称的由来。

与逻辑回归的比较:

神经网络与逻辑回归共享许多相同的数学原理。

神经网络是一个比逻辑回归更强大的分类器，理论上最简单的神经网络（即只有一个隐藏层的网络）能够学习任何函数。神经网络倾向于使用原始词汇作为输入，并在分类学习过程中学习引导特征，这种方法减少了手工提取特征的需求。

神经网络在语言处理中的应用:

本章介绍了将前馈网络作为分类器的方法，并将其应用于语言模型的简单任务，例如分配单词序列的概率和预测即将出现的单词。

## 7.1Units

神经网络中单个计算单元（神经元）的结构和作用

1. **神经单元的基本结构**:

一个神经单元接受一组实数值作为输入，对它们进行计算，然后产生一个输出。

神经单元的核心操作是对其输入进行加权求和，加上一个偏置项（bias term），得到加权和z。

1. **加权和的计算**:

对于一组输入 x1​,x2​,…,xn​，神经单元有一组对应的权重 w1​,w2​,…,wn​ 和一个偏置项 b。

加权和可以用向量表示法表示，通过点积的形式简化计算：z=w⋅x+b。

1. **激活函数的应用**:

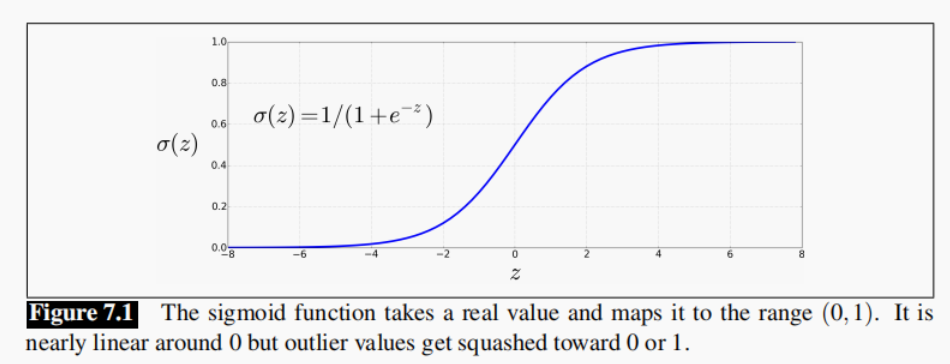
神经单元不直接使用线性函数z作为输出，而是将z传递给一个非线性函数f，这个过程生成的输出称为激活值a。

激活值a（在单个神经元模型中也可称为最终输出y）通过应用非线性函数f(z)计算得出。

1. **非线性激活函数**:

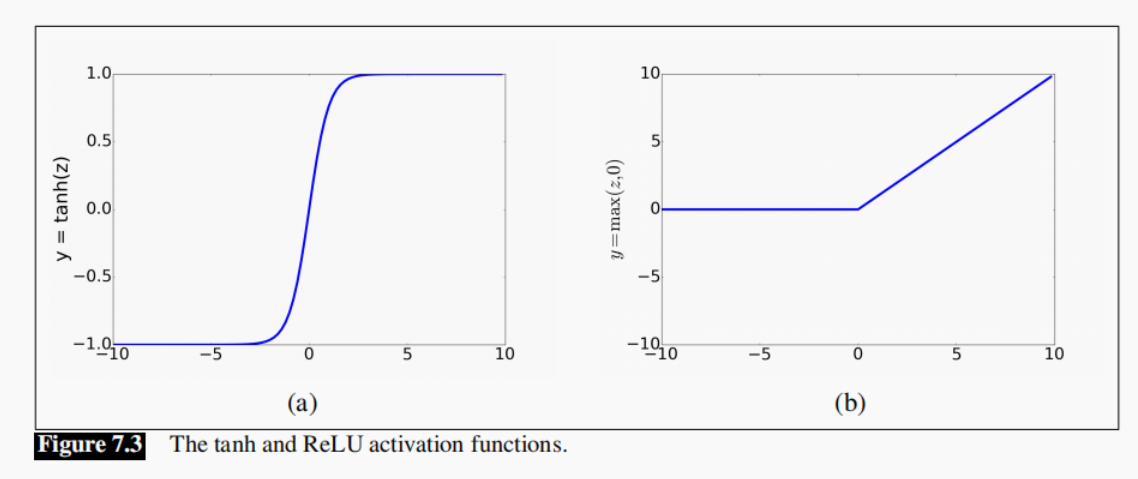
常见的非线性激活函数包括sigmoid函数、tanh函数和ReLU（修正线性单元）。

Sigmoid函数：将输入映射到(0,1)区间，对极端值有压缩效果，公式为 y=σ(z)=1+e−z1​



Tanh函数：将输入映射到(-1,1)区间，几乎在所有情况下都比sigmoid函数表现更好。

ReLU函数：正数时保持不变，负数时输出为0，公式为 y=ReLU(z)=max(z,0)。



1. **激活函数的选择**:

激活函数的选择对于神经网络的性能有重要影响。

Tanh函数具有平滑可微分的特性，而ReLU函数则因其近似线性特性而受到青睐。

Sigmoid和tanh函数在z值非常高时会导致激活值饱和，这会导致导数接近于0，从而引发学习过程中的梯度消失问题。

ReLU函数避免了梯度消失问题，因为它对于高z值的导数是1而不是接近于0。

1. **不同激活函数的属性**:

不同的激活函数适用于不同的场景和网络架构。

例如，tanh函数适合将异常值映射到平均值附近，而ReLU函数由于其接近线性的特性，适合避免梯度消失问题。

## 7.2 The XOR problem

讲述了神经网络历史上的一个重要问题：异或（XOR）问题，以及如何通过多层神经网络解决这个问题。

1. **XOR问题的引入**:

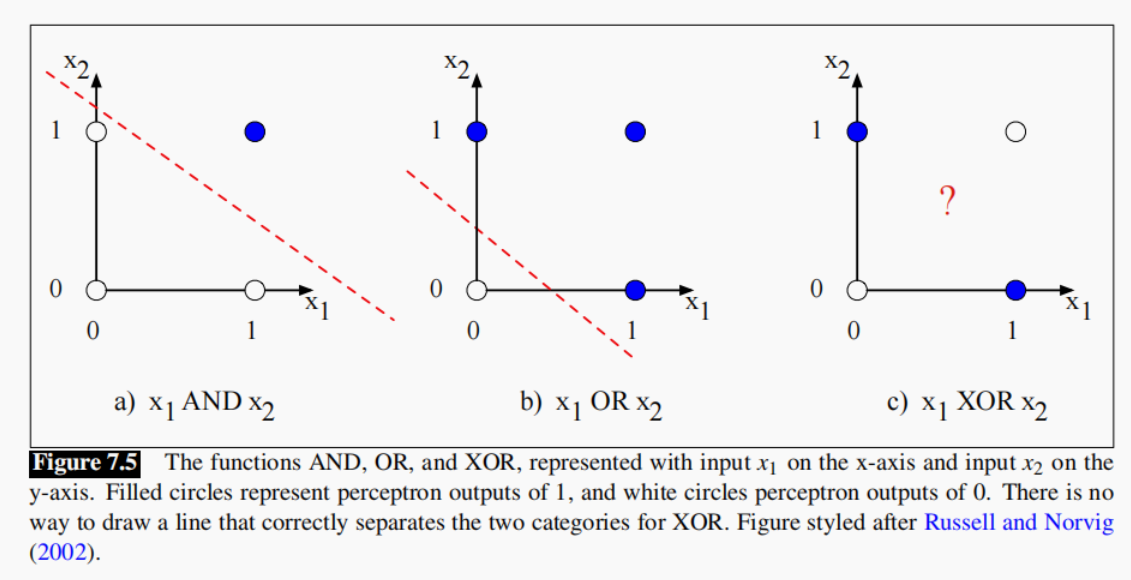
神经网络的强大能力来自于将单个计算单元组合成更大的网络。

Minsky和Papert在1969年证明了单个神经单元无法计算一些简单函数，如逻辑函数AND、OR和XOR，这引出了所谓的XOR问题。

1. **突破：感知机；但仍然无法解决XOR**:

•感知机是一种简单的神经单元，具有二进制输出且没有非线性激活函数。

•感知机可以轻松地计算逻辑AND和OR函数，但无法计算逻辑XOR函数，这是因为感知机是一个线性分类器，而XOR不是线性可分的。

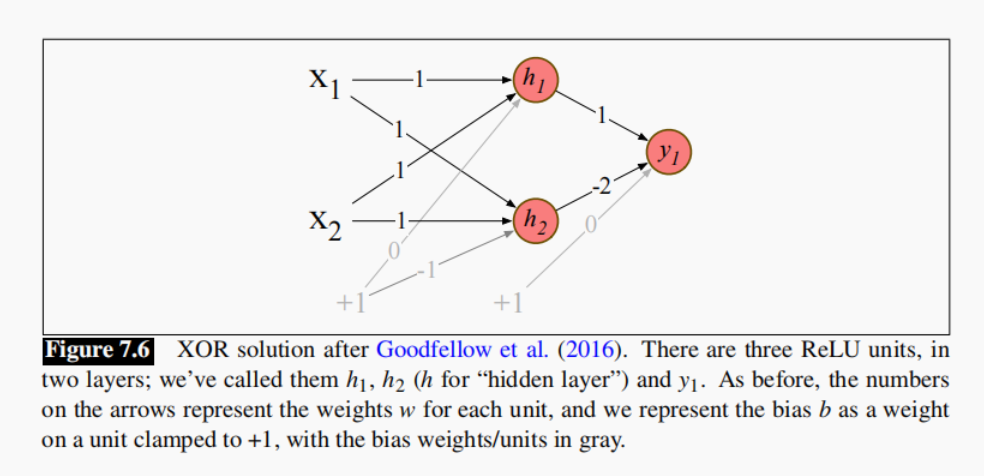


1. **解决XOR问题**:

尽管单个感知机无法解决XOR问题，但是通过多层网络（例如使用ReLU激活函数的双层网络）可以计算XOR函数。

通过在中间层和输出层之间添加权重和偏置，网络能够正确地计算XOR函数。

中间层（隐藏层）包含两个单元，输出层包含一个单元。通过适当的权重和偏置，可以使网络对XOR函数的不同输入产生正确的输出。



1. **隐藏层的重要性**:

隐藏层的输出可以视为输入的新表示，这使得原本非线性可分的XOR问题变成线性可分的。这展示了神经网络通过隐藏层自动学习输入的有用表示的能力。

1. **自动学习表示的能力**:

神经网络的一个关键优势是能够自动学习输入的有用表示，这是通过误差反向传播算法自动学习权重实现的。这种能力使得神经网络在处理各种问题时非常灵活和强大。

## 7.3 Feedforward Neural Networks

前馈神经网络：前馈神经网络是一种多层网络，网络中的单元之间没有循环连接，每层的输出作为下一层的输入，而不会反馈到更低的层。这种网络有时被称为多层感知器（MLP），尽管这个名称不完全准确，因为现代网络单元不是传统意义上的感知器。

1. **网络节点类型**:

前馈网络通常包含三种类型的节点：输入单元、隐藏单元和输出单元。

输入层由简单的标量值组成，隐藏层由隐藏单元组成，每个隐藏单元都是一个神经单元，输出层则根据网络的任务生成最终结果。

1. **隐藏层和全连接**:

在标准架构中，每一层都是全连接的，即每层中的每个单元都接受前一层所有单元的输出作为输入。

每个隐藏单元计算输入的加权和，然后应用一个非线性激活函数。

1. **权重和偏置**:

隐藏层的每个单元都有一个权重向量和一个偏置项。

这些权重和偏置可以组合成整个层的权重矩阵和偏置向量，从而使用简单的矩阵运算高效地进行计算。

1. **激活函数和输出**:

隐藏层的输出是通过将权重矩阵与输入向量相乘，加上偏置向量，然后应用激活函数（如sigmoid、tanh或ReLU）得到的。

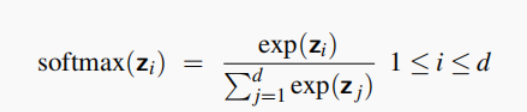
输出层将这个新的表示（隐藏层的输出）转换成最终输出，这可能是一个实数值或是一个分类决策。

1. **输出层和分类**:

对于二分类任务，可能只有一个输出节点，表示正面与负面情绪的概率。

对于多分类任务，可能有多个输出节点，每个节点对应一个类别的概率。

输出层通常通过softmax函数将实数值向量转换为概率分布。



1. **非线性激活函数的必要性**:

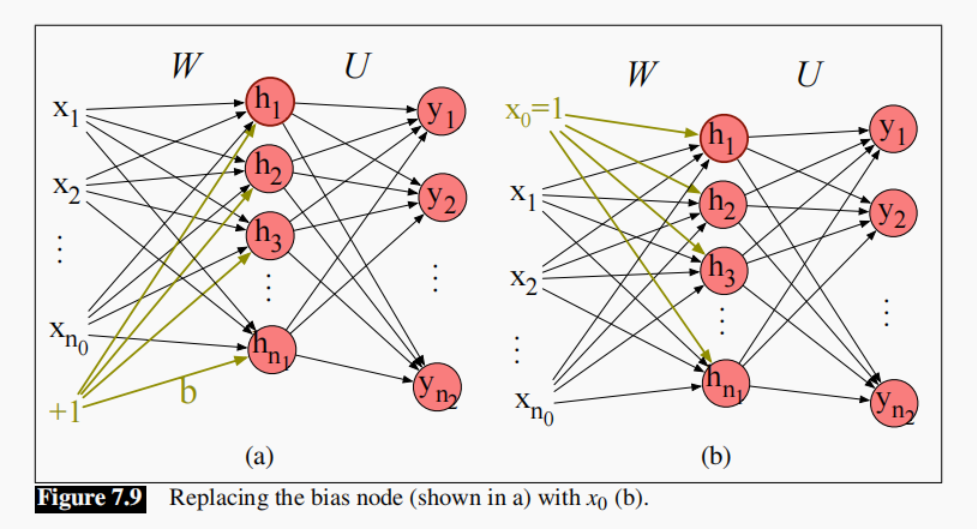
非线性激活函数是前馈网络的关键，因为没有非线性，多层网络实际上等同于单层网络。

通过引入非线性，网络能够学习并表示更复杂的函数。

1. **偏置单元的替换**:

在一些表示中，可以通过在每层添加一个总是值为1的虚拟节点来简化对偏置项的处理。

这种方式下，权重矩阵包括了处理偏置的权重，从而简化了网络的表示和计算。



## 7.4 Feedforward networks for NLP: Classification

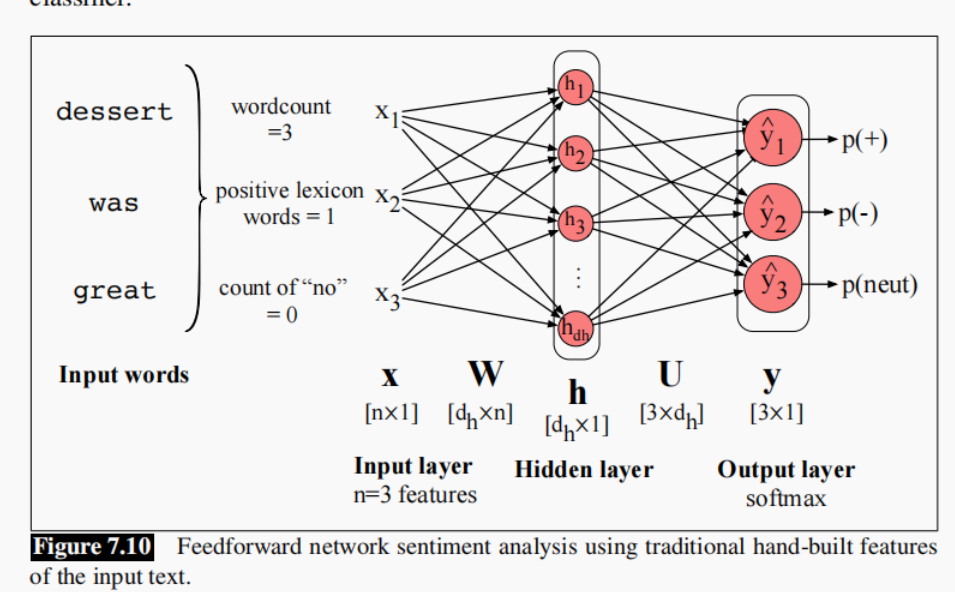
如何将前馈网络应用于自然语言处理(NLP)的分类任务，如情感分析。

1. **情感分类器的构建**:

可以将逻辑回归分类器（相当于单层网络）扩展为包含隐藏层的两层网络来构建情感分类器。

输入元素可能是手工设计的特征，如文档中的单词计数、积极词汇计数等。

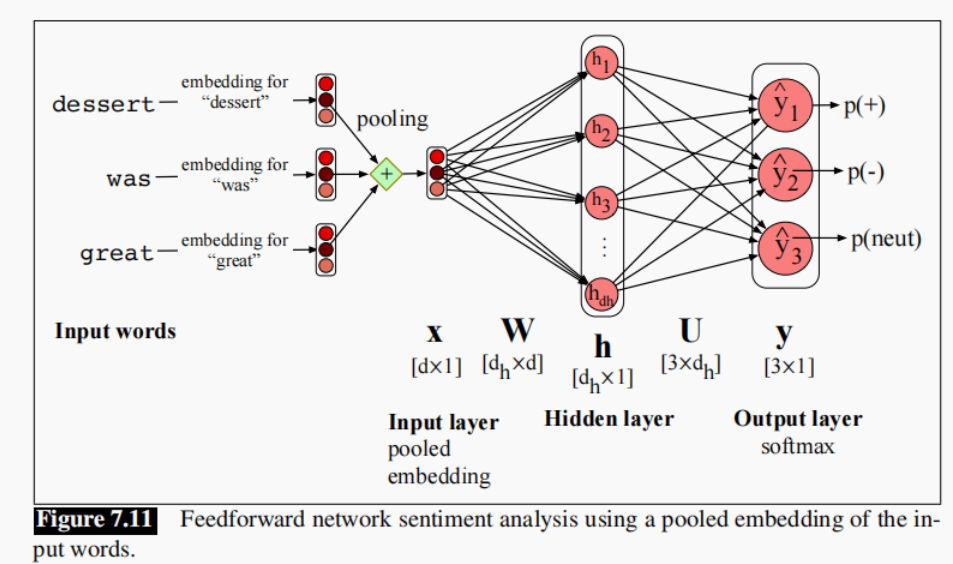
输出层可以包含两个或三个节点，分别代表正面、负面和中性情感的概率。



1. **深度学习中的特征学习**:

与使用手工设计的特征不同，NLP中的神经网络通常通过将单词表示为嵌入向量来学习数据的特征，如word2vec或GloVe嵌入。

一个简单的基线方法是对输入文本中所有单词的嵌入向量应用汇总函数，如求和或平均，以得到单个嵌入向量作为输入。还可以考虑其他方法，如逐元素最大值。



1. **分类器的架构和计算**:

使用汇总嵌入作为输入，通过隐藏层计算，然后输出层产生分类概率。

这个过程可以扩展为批量处理，以提高计算效率。

1. **预训练的使用**:

在输入表示中使用预训练的word2vec或GloVe嵌入是深度学习中的一个核心思想。

这种方法被称为预训练，可以利用已经学习好的嵌入表示作为输入特征。

## 7.5 Training Neural Nets

如何训练神经网络

1. **神经网络的训练目标**:

在监督学习框架下，目标是学习每一层的参数 W[i] 和 b[i]，使得对于每个训练样本，预测输出 ˆy 尽可能接近真实输出 y。

可看作一个最小化损失函数的过程。通常使用逻辑回归中的交叉熵损失函数作为损失函数。

1. **优化算法**:

使用梯度下降算法寻找最小化损失函数的参数。

梯度下降需要知道损失函数相对于每个参数的梯度。

1. **误差反向传播**:

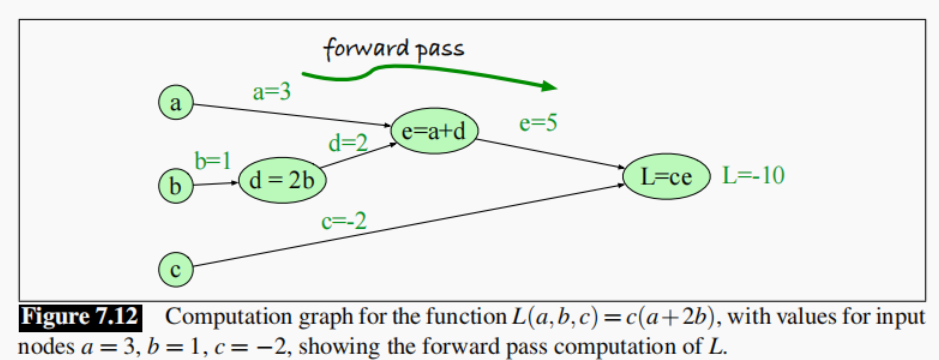
误差反向传播算法（Backpropagation）用于计算多层网络中每个参数的偏导数。

这种算法基于计算图（Computation Graph）和链式法则（Chain Rule）来计算梯度。

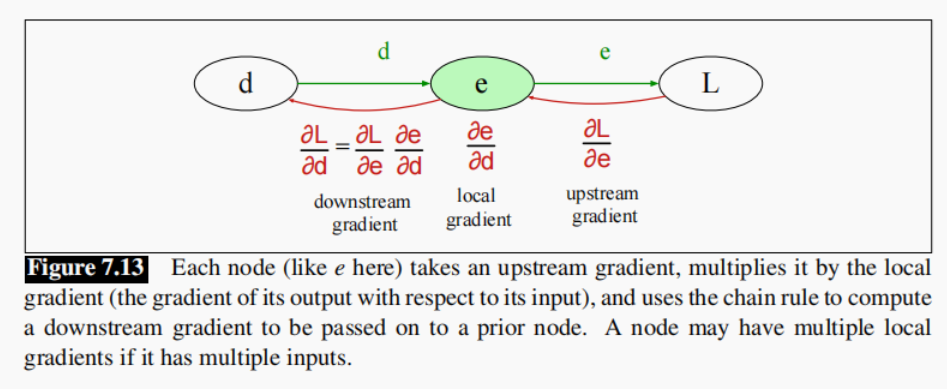
1. **计算图**:

计算图将计算过程分解成一系列单独的操作，每个操作作为图中的一个节点。

计算图用于正向传递计算损失函数值



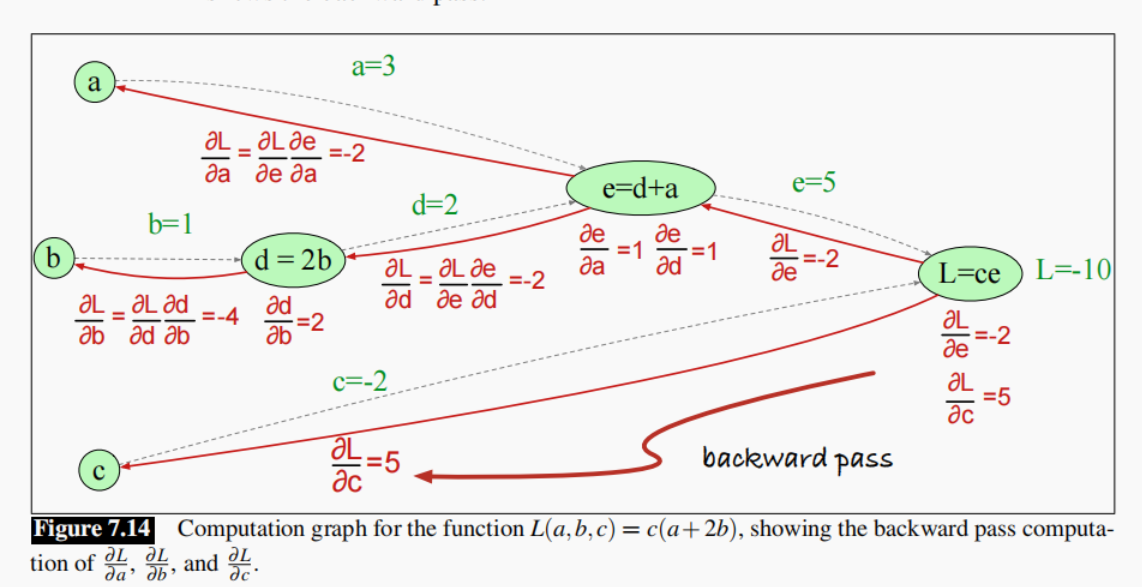
并通过反向传递来计算梯度。



1. **梯度计算**:

详细解释了如何在计算图中进行反向传播，从而计算出每个输入变量的梯度。

通过链式法则，可以计算出损失函数相对于神经网络中所有权重和偏置的偏导数。



1. **学习的细节**:

神经网络的优化是一个非凸优化问题，初始化权重和正则化非常重要。

随机初始化权重而不是将它们全部设置为零。

应用正则化技术（如dropout）以防过拟合。

调整超参数（如学习率、批大小、网络架构等）以提高性能。

1. **实现细节**:

大多数现代神经网络使用计算图形式进行构建，这使得梯度计算和GPU上的并行化更加容易和自然。

提到了流行的神经网络框架，如PyTorch和TensorFlow，它们支持自动梯度计算和大规模并行处理。

## 7.6 Feedforward Neural Language Modeling

介绍了使用前馈网络进行神经语言建模，特别是在预测基于前面单词的后续单词的语言模型方面。

1. **神经语言模型的优势与局限**:

相比于第三章介绍的n-gram语言模型，神经语言模型能处理更长的历史信息，更好地泛化相似上下文中的单词，并在单词预测方面更为准确。

神经网络语言模型比n-gram模型更复杂，训练速度较慢，能耗更高，且可解释性较差。

1. **前馈神经语言模型的基本结构**:

输入为前N个单词的表示（例如，wt-1, wt-2等），输出为可能的下一个单词的概率分布。

这种模型通过近似基于前N-1个单词的历史来近似单词wt给定之前所有单词的上下文概率P(wt|w1,…,wt-1)。

1. **单词的表示**:

神经语言模型使用单词嵌入来表示历史中的单词，而不是像n-gram模型那样仅使用单词的身份。

使用嵌入使得模型能够更好地泛化到未见过的数据。

1. **前馈推理（前向推断）**:

给定输入后，进行前向传播以产生可能输出的概率分布，即下一个单词的概率。

先将N个前面的单词表示为长度为|V|（词汇量大小）的one-hot向量，然后通过嵌入矩阵转换成嵌入向量。

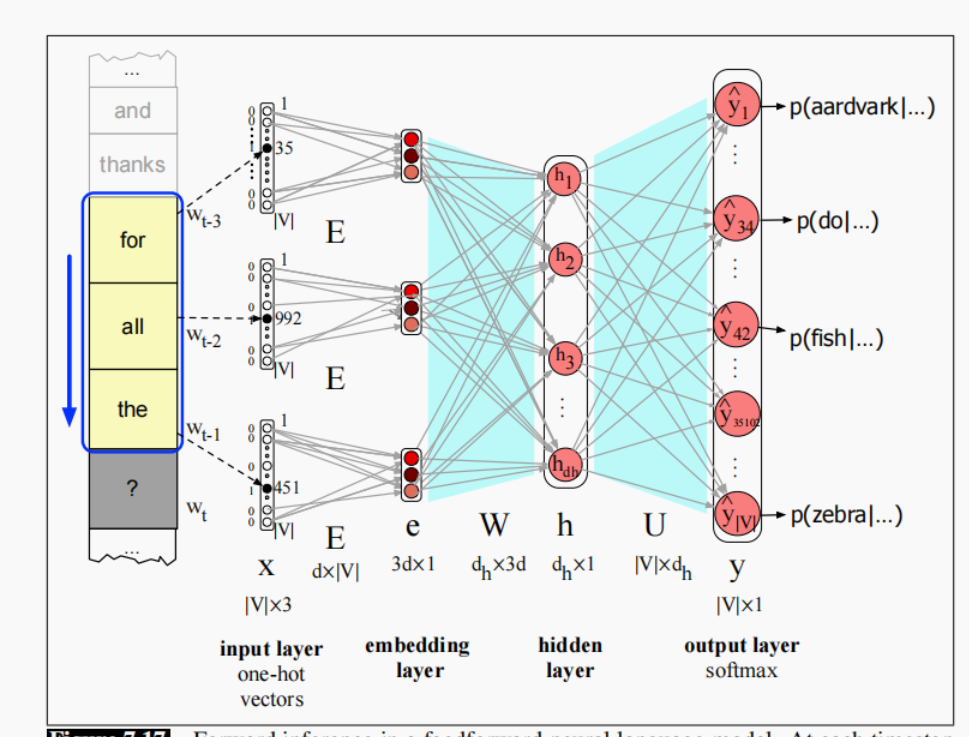
这些嵌入向量被连接起来形成嵌入层，然后通过隐藏层和输出层，最后通过softmax函数产生单词的概率分布。

1. **模型的算法细节**:

对于前面的三个单词，查找它们的索引，创建三个one-hot向量，然后各自乘以嵌入矩阵E。

这三个嵌入向量被串联起来形成嵌入层，然后乘以权重W（加上偏置b）并通过ReLU（或其他激活函数）获得隐藏层h。

隐藏层h再乘以另一个权重矩阵U，经softmax处理后，每个输出节点的值代表下一个单词为词汇表中相应单词的概率。



## 7.7 Training the neural language model

如何训练神经语言模型

1. **自监督训练的概念**:

神经语言模型的训练基于自监督或自我训练的思想，即使用文本语料库作为训练材料，模型试图预测下一个单词。

由于我们知道每个时间步的正确答案（即语料库中的下一个单词），我们可以训练模型更好地预测正确的下一个单词。

1. **模型参数的设置**:

训练模型意味着设置参数 θ = E, W, U, b。

对于某些任务，可以选择固定嵌入层E的初始值（例如，使用word2vec的值），即在训练过程中不更新E。

然而，通常我们希望同时学习嵌入和训练网络，这对于任务特定的良好词表示很有用。

1. **训练过程**:

使用梯度下降方法来设置所有参数，使用误差反向传播算法在计算图上计算梯度。

训练过程不仅设置网络的权重W和U，也通过预测即将到来的单词学习每个单词最佳的嵌入E。

1. **嵌入共享**:

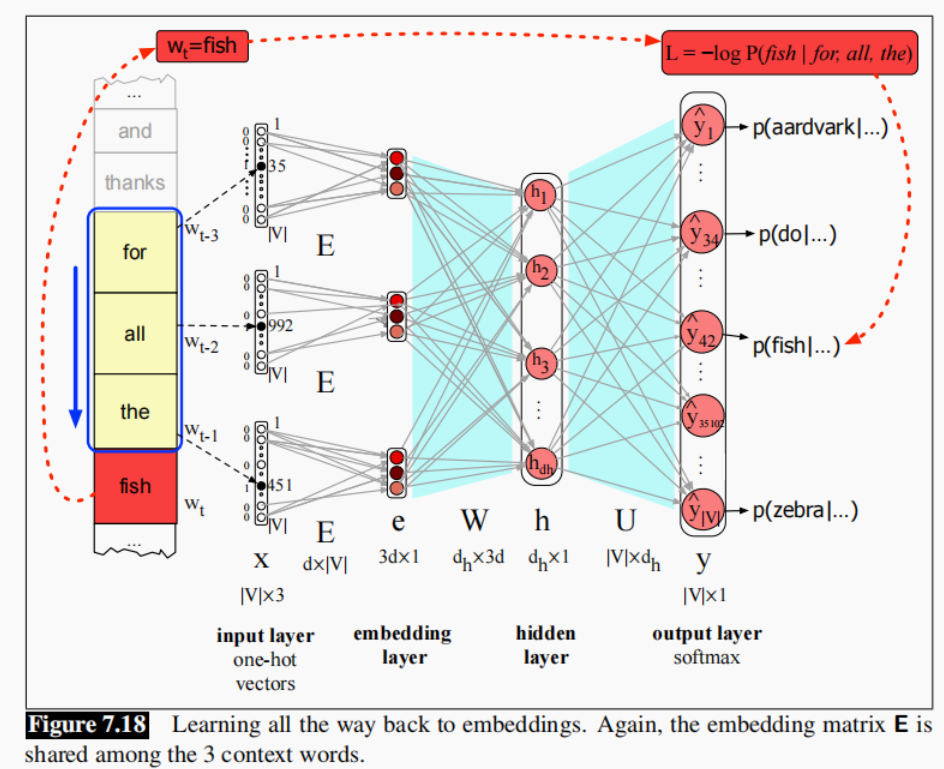
在一个给定窗口大小的语境中，不同位置的单词共享同一嵌入矩阵E，确保每个单词不管出现在哪个上下文位置都有一个统一的向量表示。

1. **训练过程的实现**:

通过处理长文本、随机初始权重、迭代预测每个单词wt来进行训练。

使用交叉熵（负对数似然）损失，并通过随机梯度下降更新参数。

通过任何标准的神经网络框架计算梯度，然后通过参数θ = E, W, U, b进行反向传播。



1. **训练结果**:

通过最小化损失来训练参数，结果不仅是一个语言模型（即单词预测器），而且是一组可以用于其他任务的新词嵌入E。

提出的相关问题：

1. 在训练神经语言模型时，如何决定使用预训练的词嵌入与自行训练嵌入之间的选择？
2. 在实际应用中，有哪些技术和策略可以加速神经网络语言模型的训练过程？
3. 嵌入向量的初始化和学习:在训练神经语言模型时，选择预训练的嵌入向量（如word2vec或GloVe）与从头开始随机初始化并学习嵌入向量有什么区别？在哪种情况下，一种方法比另一种方法更有优势？
4. 词嵌入向量的进一步应用:从神经语言模型学习到的词嵌入向量可以如何应用于其他自然语言处理任务中？是否有案例展示了这些嵌入向量如何提高其他任务的性能？