一晚让你

神经网络与深度学习

不挂科

（第一版）

陈傲 著

前言

这本书是在我在我复习《神经网络与深度学习》这门专选课所写的。其主要是面向机器人工程专业的同学，由于他们没有上过 AI 相关的课程，需要花较大力气听懂这门课。

作为专选课，考试内容虽然不难，但仍需要你对神经网络有一定的了解。因此，这本书会尽力用通俗的语言来讲述相关的知识，而且会省去复杂的理论公式。力求你在考试前一晚看，就能在第二天的考试上~~吹牛逼~~写出简答题答案。

这本书依旧会开源，地址：

https://github.com/DylanAo/AHU-AI-Repository

陈 傲

于槐园

2024 年 5 月 28 日

目录

1 你需要了解的机器学习有关术语

1.1 基本概念

1.1.1 什么是机器学习

机器学习可以简单理解为一个函数，给你一个输入，你要有一个输出，而这个函数怎么找，这个函数是什么是机器学习的内容。

1.1.2 集

训练集:训练集的数据是用来训练模型的。
测试集:用于检测训练的模型好坏的。

若给你一个数据集，你需要将其划分为两部分，即训练集和测试集。

1.1.3 机器学习的任务

分类:输出为离散值，可细分为多分类，单分类任务。

回归:输出为连续值，你可简单理解为给你一堆点，我要找出一个曲线，使得点大致在这条曲线周围(最简单的线性回归可以用最小二乘法实现)。通过若干带有标注的样本数据构造出一个预测模型 f(x)，使得 f(x)输出预测尽可能是一个真实值。

聚类:没有标签，也就是没有标记信息，只是单纯把长得像的放到一起(分类任务的训练集是有标签的，他明确的告诉你这个东西是哪一类的)。聚类任务属于无监督学习(没有标签)，而回归和分类任务是有监督学习。

1.1.4 特征工程与深度学习

特征工程由人类专家根据现实任务来设计，特征提取与识别是单独的两个阶段。

将原始的数据特征通过多步的特征转换得到一种深度特征表示，进一步输入到预测函数得到最终结果。

1.2 机器学习三要素

机器学习三要素:模型、学习准则、优化方法。

1.2.1 模型

我们前面提到过，机器学习就是再找一个函数，那么这个函数你可以简单理解为我的模型，最简单的情况就是线性模型，类别于数学中 y=ax+b，这里面为:

$$f(\mathbf{x};\mathbf{w},b)=\mathbf{w}^{\mathrm{T}}\mathbf{x}+b$$

我们称 w 为权重，b 为偏置，和数学上简单的线性模型不同的是，此处我的输入可以是一个多维的向量，那么我对权重自然就变成了权重矩阵，习惯上我们竖着写，为了符合矩阵乘法所以我们要加转置。这里面的权重与偏置就是我们机器学习所要学习的内容。

那么对应非线性模型，就是将 x 变为一个非线性的函数，即广义的非线性模型可以写为多个非线性基函数的线性组合:

$$f(\boldsymbol{x};\theta)=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})+b$$

1.2.2 学习准则

我们在训练一个模型的时候怎么知道这个模型好坏呢？我们需要一个标准来评判学习的好坏，这个就叫做学习准则。

应用机器学习术语，上述表达就转化为模型的好坏由期望风险来衡量，我们要做的就是要让期望风险最小化。但实际上，这也是很宽泛的一句话，我们希望要通过一个数学的方法来衡量我们的好坏，那么该怎么衡量呢？

我们可以通过利用损失函数来计算出我们的期望风险，针对于输出值是连续的情况(回归问题)，我们一般利用平方损失函数：

$$\mathcal{L}(y, f(\boldsymbol{x};\theta)) = \frac{1}{2}\big(y - f(\boldsymbol{x};\theta)\big)^2$$

括号内的部分可以看作实际与预测之间的误差，而平方保证了误差永远为正，前面的二分之一是为了方便求导计算。

但是，若是对应到分类问题，情况就有些不一样了。分类问题输出的是离散的值，非 0 即 1，那么他的损失函数可描述成 0-1 损失函数：

$$\begin{aligned}\mathcal{L}(y, f(\boldsymbol{x};\theta)) &= \begin{cases} 0 & \text{if } y = f(\boldsymbol{x};\theta) \\ 1 & \text{if } y \neq f(\boldsymbol{x};\theta) \end{cases} \\ &= I\big(y \neq f(\boldsymbol{x};\theta)\big),\end{aligned}$$

但是，这种损失函数我们没办法求导优化，也就是应用梯度下降，因此在分类问题中我们常用交叉熵损失函数：

$$\begin{aligned}\mathcal{L}(\boldsymbol{y}, f(\boldsymbol{x};\theta)) &= -\boldsymbol{y}^{\mathrm{T}} \log f(\boldsymbol{x};\theta) \\ &= -\sum_{c=1}^{C} y_c \log f_c(\boldsymbol{x};\theta)\end{aligned}$$

交叉熵损失函数想要从原理理解非常复杂，你可以简单理解为模型预测的概率与实际上概率进行比较得到的误差。

1.2.3 优化方法

我们机器学习就是要找出合适的权重和偏置使得我期望风险最小化，但此处存在两个问题：

1. 我们不知道我们采集样本的真实分别情况，所以我们只能用经验风险代替我们的期望风险，这也是导致过拟合的原因。（应用唐妞不等式：经验风险≠实际风险）

2. 我们到底该怎么样试出权重和偏置的参数呢？总不能像你在实验课上一样为了水实验报告乱调吧。

所谓经验风险，就是把我每一个样本的损失函数拿出来，求平均值：

$$\mathcal{R}_{\mathcal{D}}^{emp}(\theta) = \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}(y^{(n)}, f(x^{(n)}, \theta))$$

这样你会发现，我想要得到一个好的模型，我就需要我的经验风险最小就好，将问题转化为一个优化问题了。

针对这个优化问题，我们最简单方法就是梯度下降法：

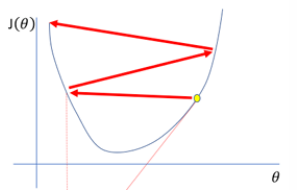
$$\begin{aligned}\theta_{t+1} &= \theta_t - \alpha\frac{\partial\mathcal{R}(\theta)}{\partial\theta_t} \\ &= \theta_t - \alpha\frac{1}{N}\sum_{i=1}^{N}\frac{\partial\mathcal{L}(\theta_t; x^{(i)}, y^{(i)})}{\partial\theta}\end{aligned}$$

这个式子看起来很复杂，实际上很好理解：我们把每一次的训练样本拿过来，扔进损失函数计算经验风险，并对经验风险求导。那么现在你想想一个开口向上的二次函数，如果求出的导数是正数，那么证明我现在在递增，为了找到最小值是不是应该往后走，即减去我的导数？如果求出的导数是负数，那么证明我现在在递减，为了找到最小值是不是应该往前走，即加上我的导数？不要忘了，减去一个负数正好是整数，也就是说，我只需要对我对求的导数做减法就可完成这一个前后走的过程了。这就是我们所说的梯度下降。

你可能会注意到一个问题，那么我走出的这一步该走多大呢？实际上，这没有一个准确的答案，因为他是一个超参数，是需要我们自己设置的。而权重与偏置叫做参数，是可以其自己学习的，也就 i 是利用梯度下降来计算出一个数。在上述式子中，用 α 表示，我们称之为学习率。

学习率过大和过小都不好(废话)。过小会导致你梯度下降太慢，电

脑跑的时间太久了。过大又会出现震荡现象，根本不会收敛到最小值。(图中的学习率就是过大了)



从理论上来说，参数的学习过程就是这门简单，梯度下降就完事了。我们总结一下，首先要随机初始化我的参数，然后疯狂求导进行梯度下降。这个参数变换的过程我们可以叫做参数更新。

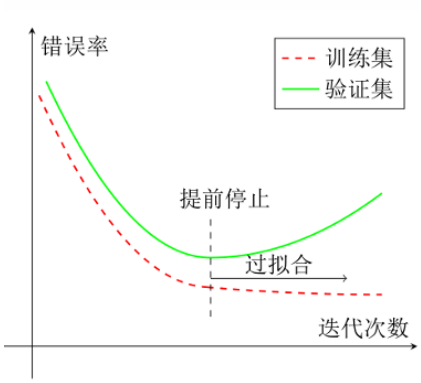
但是实际应用中你会发现存在一定问题，第一个问题就是过拟合。所谓过拟合就是在训集上匹配的太好了，而实际应用中反而不是那么的好，缺少泛化性。举个简单例子，请你对以下短语填空：

原 __ __ 动

其实正确答案是 原 地 不 动，但是你是不是填的是原 神 启 动？这就是你的大脑过拟合了，看了太多原神启动的烂梗了；如果你不知道填什么，那么你就是上网冲浪少了，这叫欠拟合。

为了解决这个问题，我们需要一个新的验证集，他是在训练集和测试集之外的，有的时候我们可能需要从训练集拿出一部分作为我们的验证集。不过，这并不是重点，你知需要知道我们有个验证集即可了。

我们在梯度下降的过程中，这些模型中的参数是变化的，而且是往好的方向变化的。那也就是说在正常情况下，你可以用你每次迭代的模型，给他输入一个验证集的数据，看他的错误率是多少。某种角度上，这里的验证集就是来模拟实际情况的。当在验证集上错误率不在下降时，我们就不让他训练了。这样就可以避免过拟合了，我们称之为提前停止。



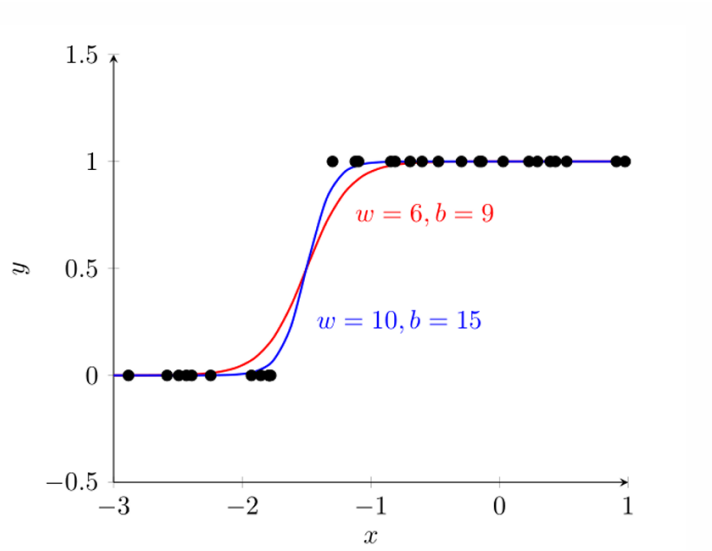
我们在梯度下降中是计算所有样本的损失然后求平均值，要是样本太多了那电脑不就算死了？对此，我们可以采用随机梯度下降。即我每次选取的样本都是随机的，通常我们用随机排序来实现选取的随机。这样，我们就可以只计算当前样本的梯度值。同时，为了让我们计算速度更快，更好榨干电脑并行计算能力，我们采用小批量梯度下降，即每次随机选取一小部分样本进行梯度计算。（当然，这里你是要求损失的平值的，即除上这一小部分样本的数目）

最后，我们要明确的是，我们虽然将问题转化为优化问题，但是机器学习问题不等于优化问题。优化问题是为了得到最优的解，但是机器学习为了泛化性，为了不过拟合有时并不是为了得到最优解。有时，我们反而要加入正则化来限制模型能力，不让他过拟合。即损害优化的方法就是正则化，通常就是增加约束(正则化项)，或者干扰过程(随机梯度下降、提前停止)。

1.3 Logist 回归与 Softmax 回归

Logist 回归是回归问题，而 Softmax 回归是一个分类问题。二者都是典型的线性模型，也都用交叉熵作为损失函数。

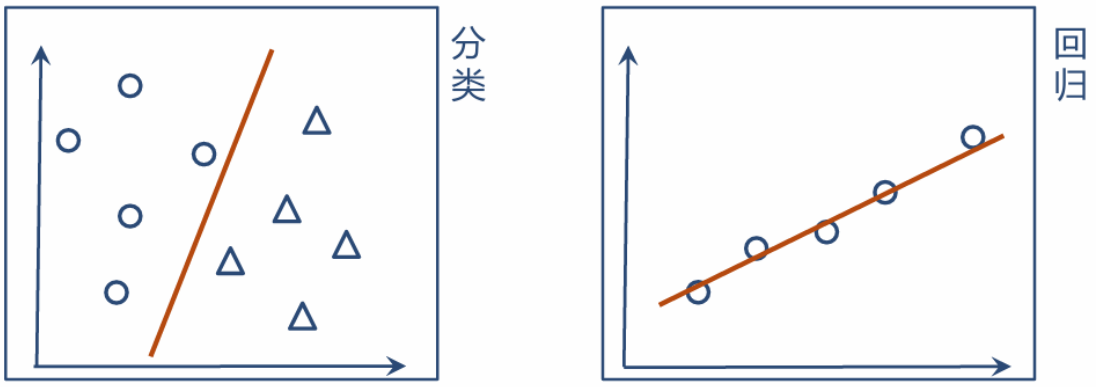
Logist 回归是用来处理二分类问题的。你现在是不是晕了，怎么又是回归又是分类？他到底是什么？别急，我给你看张图就明白了。



看到没有，Logist 回归本质上还是让我的曲线尽可能在点附近，其解决了连续的线性函数不适合进行分类的问题：

1. 输出范围:线性函数输出为连续的，而分类是离散的
2. 不稳定性:对微小的改变，线性模型的输出会有较大变化，分类结果不稳定
3. 欠拟合：决策边界在二维平面上是一个直线，无法捕捉复杂的分类问题中非线性关系。

对于分类问题和回归问题的区别更通俗的理解是这样：

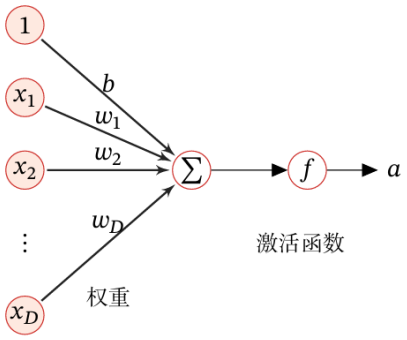


Logist 回归其实也很简单，我们只不过是对线性模型 y=ax+b 的结果套上一个新的函数 g(Logist 函数)，然后就得到了 0 或 1。

Softmax 是 Logist 回归在多分类问题上推广，只不过他套上了 Sodtmax 函数。

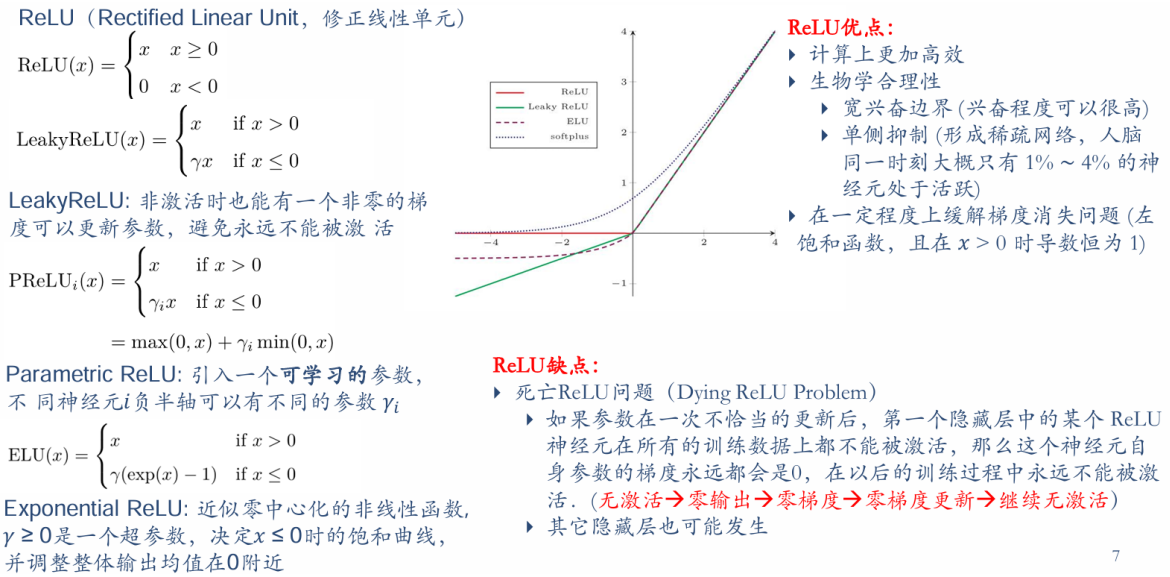
2 神经元

正如孤木不能成林，我们的神经网络是由一个个最小的单元组成的，那么个单元就称之为神经元：



很简单，前面部分可以看作是一个线性模型，给定输入我求和得到输出，将输出通过一个激活函数，就可以得到结果(活性值)。通常这个结果非 0 即 1，代表神经元是或否激活，是否兴奋。

那么这里的激活函数可就有说法了。

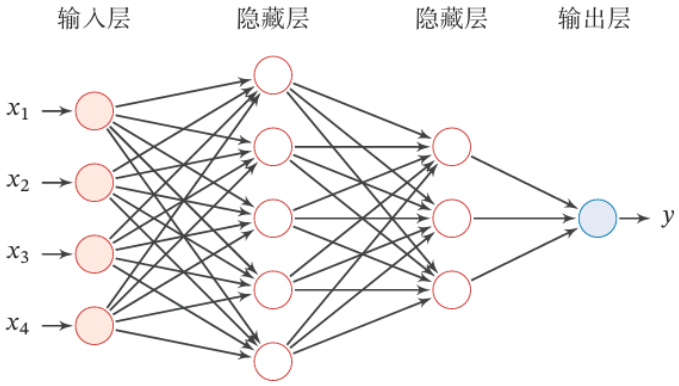


最常用的激活函数就是 ReLU，第二常用是 ELU。

3 前馈神经网络

我们有了最基本的单元，下面该考虑怎么将这些神经元连载一起了。实际上，我对神经网络都是由神经元构成的，只不过不同的神经网络他们连接的方式不同。最简单的方法就把他们分成不同的层，然后全连接起来，这种叫做前馈神经网络。

其特点为层内无连接，层间两两连接，信号由输入向输出单向传递。

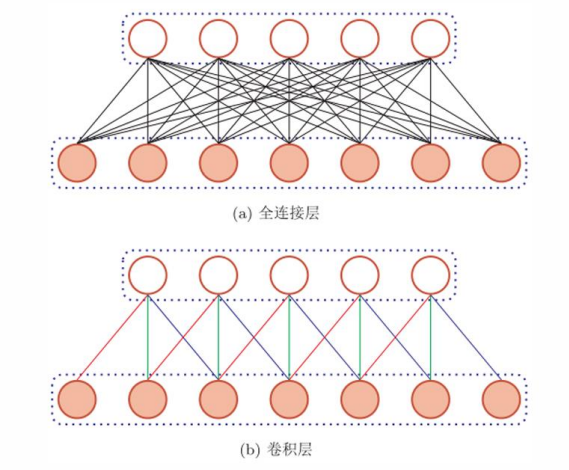


我们所谓的训练，就是在训练神经元里面的权重和偏置，自然就用到的是梯度下降。只不过，有人通过复杂的数学推导，计算出了一种称作反向梯度传播算法。顾名思义，他是倒着算的，前一层的误差可以由后一层得到，但其本质上还是梯度下降。

当然，在优化过程中可能存在梯度消失问题。所谓梯度消失．是指当网络层数很深时，梯度就会不停衰减，甚至消失，使得整个网络很难训练。一般解决方法是使用导数比较大的激活函数，如 ReLU。

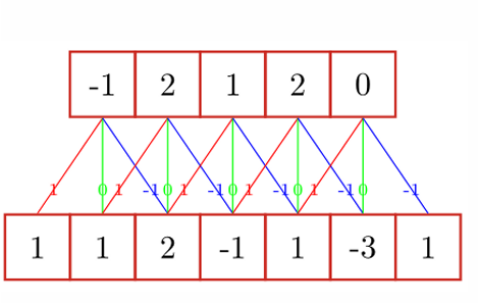
4 卷积神经网络 CNN

卷积神经网络就是在不同层神经元连接时候选择了卷积，而不是直接全部连上。这样的好处是使参数变少，更好训练，同时也利于我提取特征。一般在图像处理中用到比较多。其实就是用卷积层代替了全连接层。



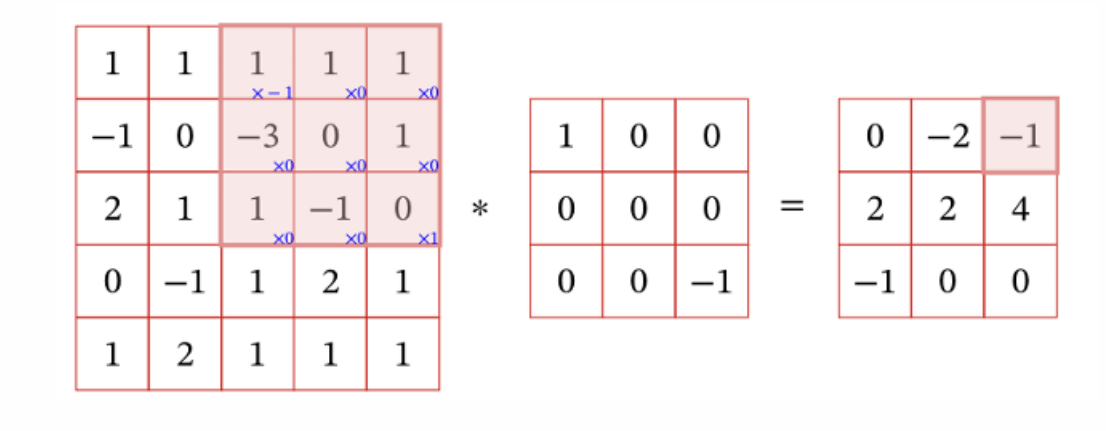
4.1 卷积的计算

4.1.1 一维卷积



一维计算非常简单，就是我对应数称对应线上的数然后加起来，就是卷积结果。我们这里线上的数可称为滤波器。

4.1.2 二维卷积

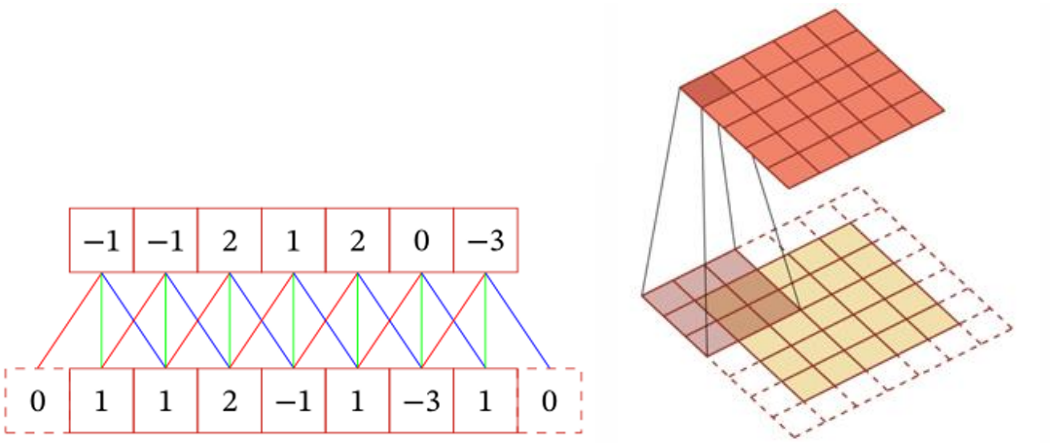


二维的卷积也很简单，我选取一个区域，对应区域分别乘上一个数。图中中间那一堆数我们称之为卷积核，把称完的结果加起来，就是卷积结果。

4.1.3 步长与零填充

我们注意到，在上述一维卷积中我每卷完一次是不是就平移一个单位接着卷？（什么卷王卷完一科接着学另一科目？）而二维卷积中是卷完一次横着平移一下接着卷，如果卷到头了那就往下平移一行从头接着卷。（什么卷王卷完绩点卷六级？）那么，我能不能平移两个单位长度，或者更多单位长度呢？答案是可以的，我们管这个平移的长度叫做步长，用 T 表示。

我们拿二维卷积举例，你有没有发现中加那几圈被卷的次数比较多？就是参加了好几次卷积运算，而最外面那一圈参加的少。那不就是相对于我最外面一圈的信息没有重复利用吗？所以，我可以选择在最外圈不一层或几层零，让他们重复的卷。（卷王害人不浅啊）不仅仅我二维可以补零，我的一维也可以补 0。我称之为零填充，记为 P。



4.1.4 卷后长度与宽度

卷后长度=(卷前长度−核长度+2*补零数)/步长+1
卷后宽度=(卷前宽度−核宽度+2*补零数)/步长+1

4.1.5 空间复杂度与时间复杂度

单个卷积核时间复杂度为：

$$\mathrm{Time}: O(M^2 \cdot K^2 \cdot C_{in} \cdot C_{out})$$

- M 每个卷积核输出特征图(Feature Map)的边长
- K 每个卷积核(Kernel)的边长
- C_{in} 每个卷积核的通道数，也即输入通道数，也即上一层的输出通道数。
- C_{out} 本卷积层具有的卷积核个数，也即输出通道数。

卷积神经网络整体的复杂度：

$$\mathrm{Time}: \mathrm{O}(\sum_{l=1}^{D} \mathrm{M}_l^2 \cdot \mathrm{K}_l^2 \cdot \mathrm{C}_{l-1} \cdot \mathrm{C}_l)$$

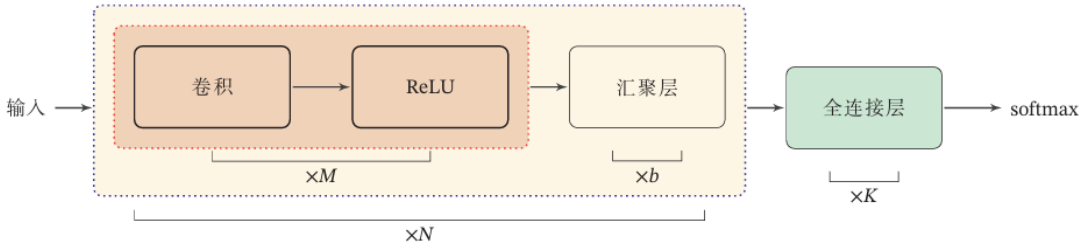
- D 神经网络所具有的卷积层数，也即网络的深度。
- l 神经网络第 l 个卷积层;
- C_l 神经网络第 l 个卷积层的输出通道数 $\mathrm{C}_{\mathrm{out}}$ ，也即该层的卷积核个数;
- 对于第 l个卷积层而言，其输入通道数 C_{in} 就是第 $l-1$ 个卷积层的输出通道数。

空间复杂度:

$$\mathrm{Space}: \mathrm{O}(\sum_{l=1}^{D} \mathrm{K}_l^2 \cdot \mathrm{C}_{l-1} \cdot \mathrm{C}_l + \sum_{l=1}^{D} \mathrm{M}_l^2 \cdot$$

4.2 卷积神经网络

卷积神经网络的基本结构是这样的，由于卷积神经网络一般用于图像的处理，所以我们下面表示都是基于图像处理的。
一般的卷积神经网络可以看作是一个卷积层一个汇聚层放到一起，构成一个类似的基本单元，然后重复这个基本单元，最后接入全连接层，再接入 Softmax 函数进行输出。

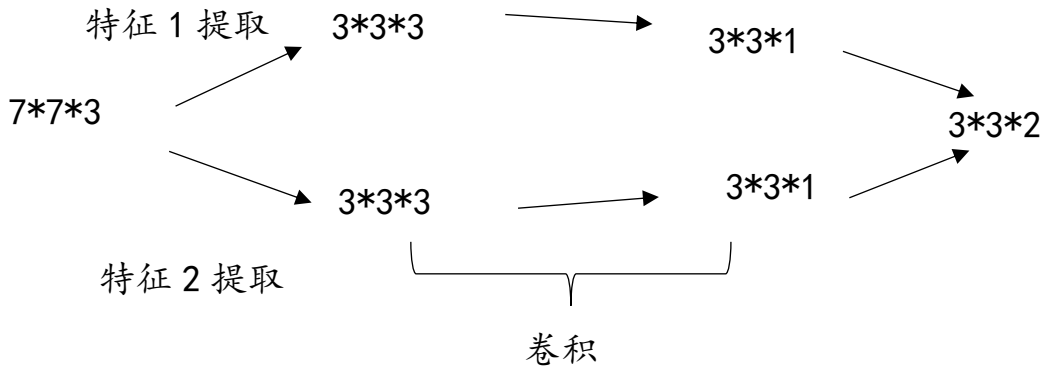


4.2.1 卷积层

假如我有 7*7*3 的图像(RGB 图像所以有 3 个)，那么假定我卷积核为 3*3，步长为 2，没有补零，所以我可以得到 3*3*3 的卷积结果，把他们全部加起来，就是 3*3*1，卷积结束。
注意，因为 RGB 三个通道，所以我对输出也是 3 个。
实际上，我们对应一个图片可以不仅仅卷一次，而是可以卷很多

次，用于提取不同的特征，下图中我就卷了两次，用于提取不同的特征。

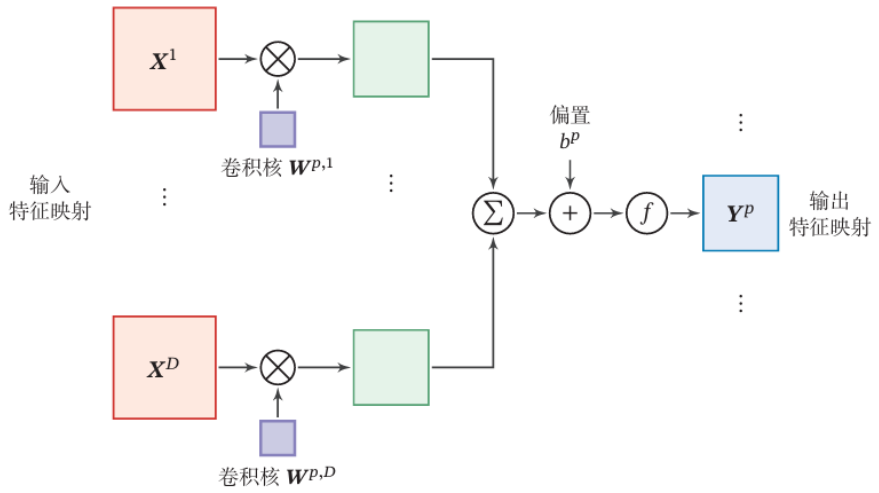
从理论上，我的卷积核在每个小区域进行卷积时应该是不同的，但实际上上，针对与提取同一个特征同一个通道上卷积核用的是一样的，称之为参数共享，这是卷积神经网络中重要特性。在下图中对一个特征用一个卷积核进行卷积以提取特征，而不是每卷一次换一个核。但是你要注意，我这个核可是 3*3*3，我说用一个核，并不等于我在每个通道上核的参数是一样的，实际上是不一样的。



对此我们卷积层需要的参数计算公式为：

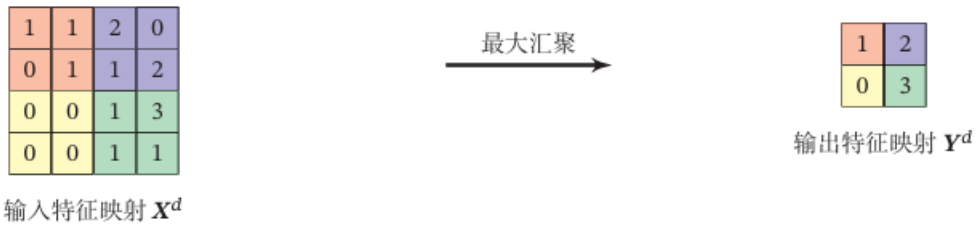
核大小(注意要乘通道数)*要提取特征数+偏置数(这一层有几个神经元)

针对每一个特征的提取，直观的表示是这样的：



4.2.2 汇聚层

汇聚层又叫做池化层，相当于对特征的又一次提取总结。这里并不设计矩阵的计算(加减乘除)。这里我们一般用最大汇聚，即对每一个小区域取最大值作为我汇聚层的结果。



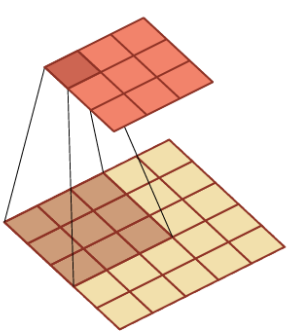
4.2.3 卷积神经网络的特点

稀疏交互：卷积核尺度远小于输入的维度，这样每个输出神经元仅与前一层特定局部区域内的神经元存在连接权重（即产生交互），我们称这种特性为稀疏交互。
参数共享：在同一个模型的不同模块中使用相同的参数。卷积运算中的参数共享让网络只需要学一个参数集合，而不是对于每一位置都需要学习一个单独的参数集合。使得卷积层具有平移等变性。
等变表示：神经网络的输出对于平移变换来说应当是等变的。假如图像中有一只猫，那么无论它出现在图像中的任何位置，卷积神经网络将它识别为猫。

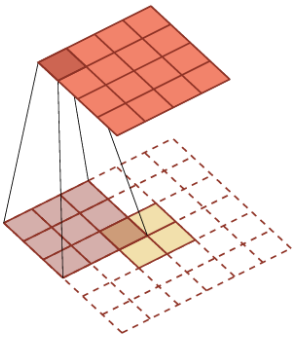
4.3 卷积的变体

4.3.1 卷积转置

正常的卷积我们发现是越卷越小的，而转置卷积是越卷越大，即低维特征映射到高维特征。



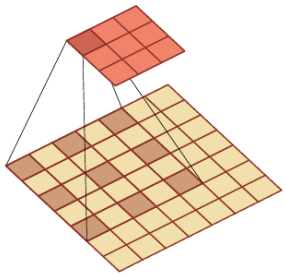
(a) 卷积,$S=1,P=0$



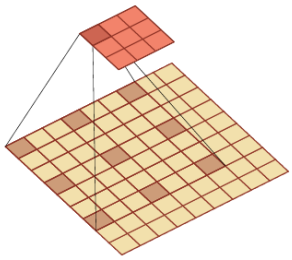
(b) 转置卷积,$S=1,P=2$

4.3.2 空洞卷积

就是中间有洞，并不是挨着卷的。



(a) 膨胀率 $D=2$



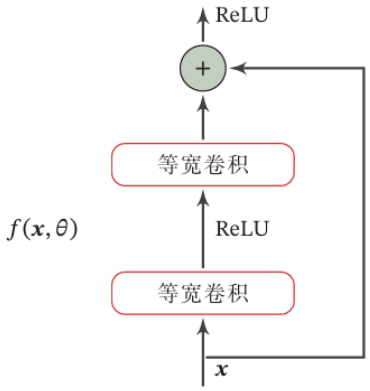
(b) 膨胀率 $D=3$

4.3.3 1*1 卷积

即卷积核大小是 1*1，那么他卷后既不会改变长度也不会改变宽度。主要是用于增加非线性和实现跨通道的交互。

4.4 残差网络

我们卷积网络叠加太多卷积层以后效果反而不好(卷王不得好死)，有时候卷了反而不如不卷，所以我们提出了残差神经神经网络。即在非线性卷积层增加了直连边。



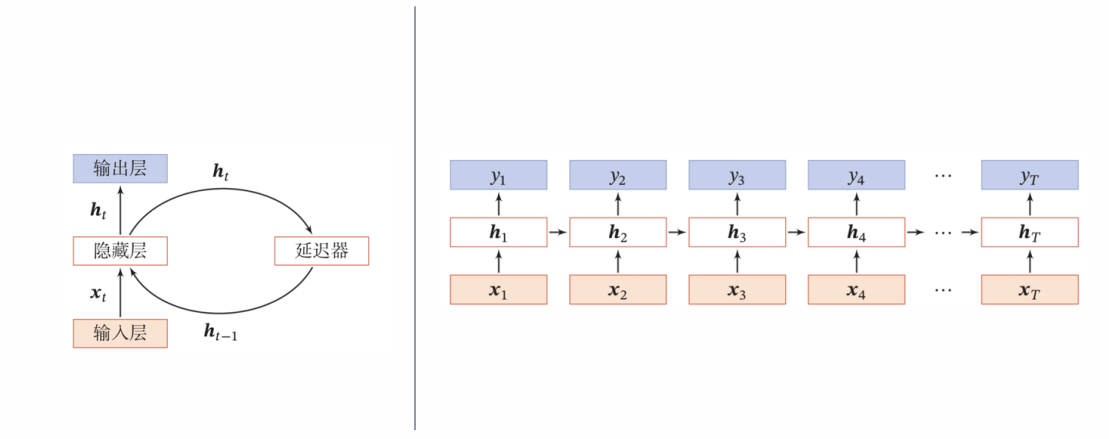
你可以简单理解为要是卷的效果不好，我可以直接将偏置和权重设置为 0，这样你的输入 x 相当于走旁边那条边，没有卷积。这样就解决了卷了反而不如不卷的问题。（其实就是摆烂了，不卷了）

$$h(\boldsymbol{x}) = \underbrace{\boldsymbol{x}}_{\text{恒等函数}} + \underbrace{\big(h(\boldsymbol{x}) - \boldsymbol{x}\big)}_{\text{残差函数}}$$

5 循环神经网络 RNN

5.1 简单循环神经网络

你有没有发现，在前馈神经网络和卷积神经网络中每个卷积层自己之间是没有连接的，而循环神经网络则增加了关于自己的连接。即我当前的状态受到历史状态的影响。



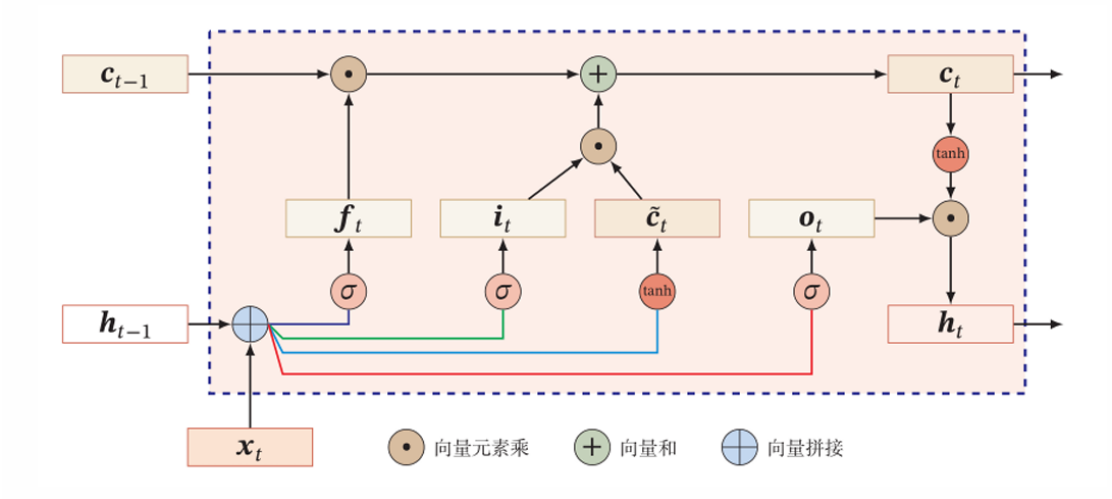
但是针对我们普通的循环神经网络，梯度爆炸或消失问题，实际上

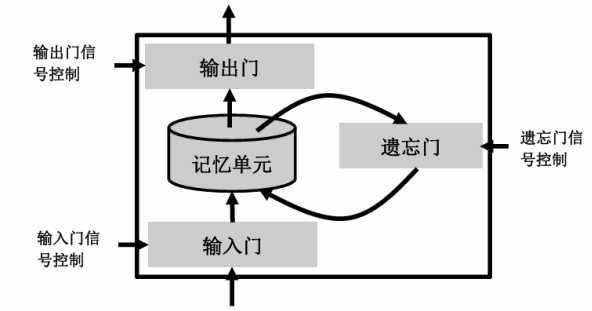
只能学习到短周期的依赖关系。这就是所谓的长距离依赖问题。

5.2 基于门控的循环神经网络

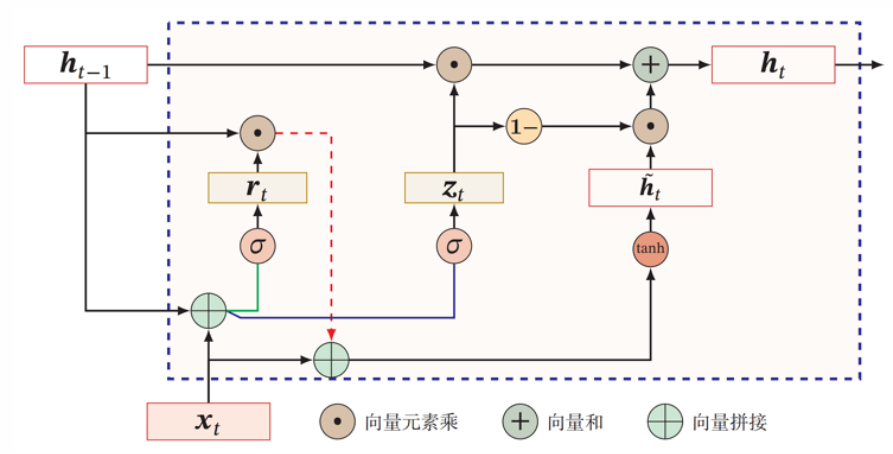
为了改进这些问题，我们可以利用基于门控的循环神经网络，常见为长短期记忆神经网络(LSTM) 和门控循环单元网络(GRU)。

LSTM 如图所示，其最重要的点就是引入了三个门：输入门 i_t、遗忘门 f_t和输出门 o_t，用以解决何时遗忘多少历史信息，何时用多少新信息更新记忆单元。图中的 c_t相当于是我记忆中全部的信息，然后通过这三哥门控制，门的取值范围都是[0,1]。





GRU 相当于是对 LTSM 进行了一定的简化。



5.3 循环神经网络特点总结

优点：引入记忆、图灵完备
缺点：长程依赖问题 、记忆容量问题 、并行能力、梯度消失与爆炸

同时将循环神经网络扩展到树结构为递归神经网络、拓展到图结构是图神经网络。

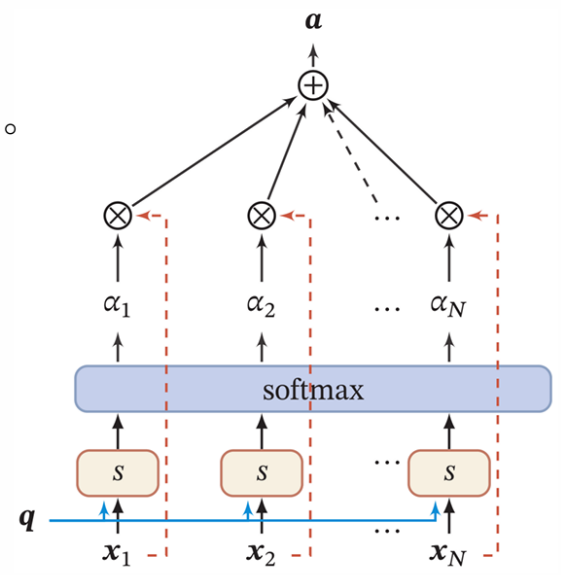
6 注意力机制

在认识神经学中注意力可分为两种，聚焦注意力、显著注意力。分别对应自上而下和自下而上。

6.1 软注意力与硬注意力

硬注意力，最后输出是离散的值，非 0 即 1，找出的是最相关的。实际上，最简单的软注意力本质上还是矩阵的计算。

软注意力，最后输出的是连续的值，相当于是给出的概率，一种打分。



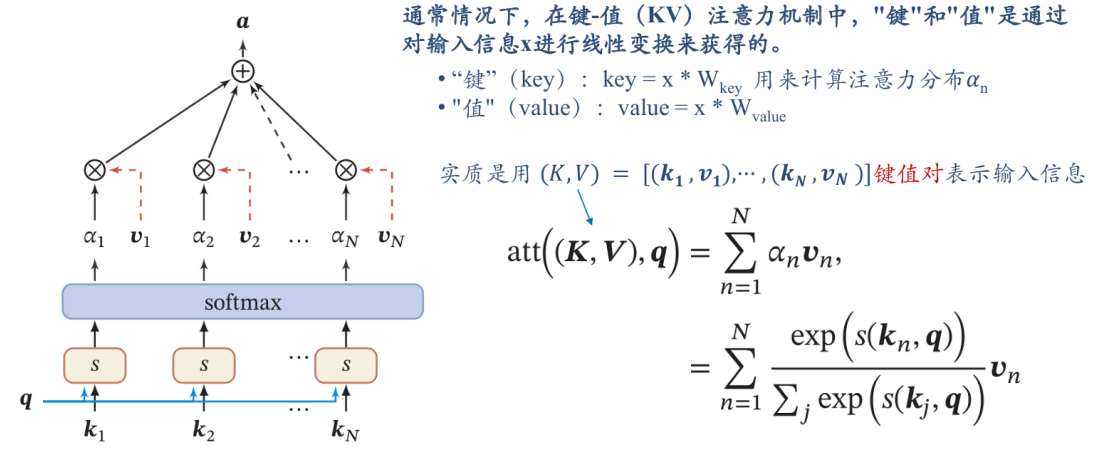
上图是计算软注意力的过程图，可简单概述为两个步骤：

1. 即算注意力分布（α），即关注第 n 个信息上概率多少。
2. 根据计算的信息进行加权平均。（乘的部分）

其中 S 为打分函数，实际上有很多种打分函数，而 Softmax 是用于归一化，本质上来讲 Softmax 函数可以看作是一种求概率的方法。q 其实就是查询向量，x 为输入的信息。

6.2 键值对注意力

键值对注意力可以看作是软注意力一种变形， 它只不过将输入的信息拆分为两个部分:K 即键，相当于索引；V 即值。而整个过程可以看作是由 K 计算相关性然后对 V 进行加权汇总的过程。



6.3 多头注意力

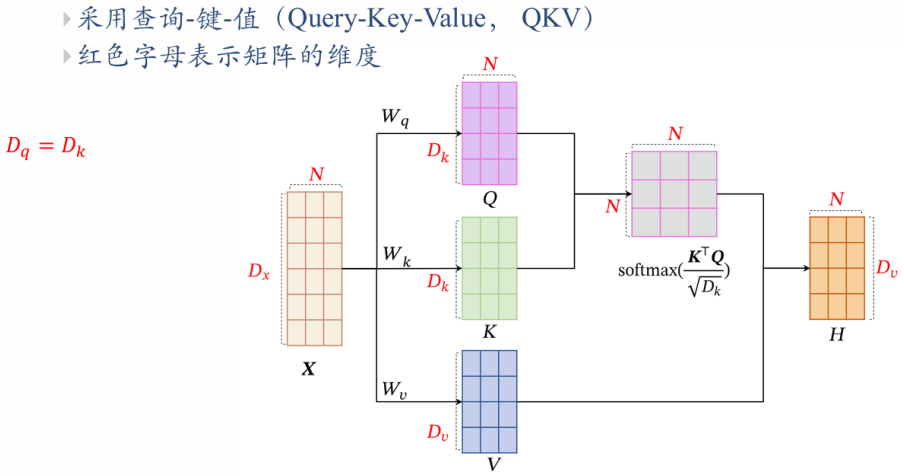
多头注意力就是同时对输入信息选择多组信息进行注意力打分

6.4 自注意力模型与多头自注意机制

6.4.1 自注意力模型

自注意力模型是在键值对注意力和多头注意力基础上形成的，本质上还是注意力机制的一种变形，只不过他非常重要，它是 transformer 中核心东西，需要单独来讲。
自注意力模型主要是用来形成非局部依赖关系，其实用全连接也可以实现非局部依赖关系，但是它不能处理定长序列。而自注意模型其核心的内容也是矩阵的计算，叫做 QKV 模式。

你有没有发现，在前面讲到的注意力中，你的查询向量 q 并不是从给你的输入信息里面来的，而自注意力模型的自就在此处，即它的 q 是自己输入信息来的。

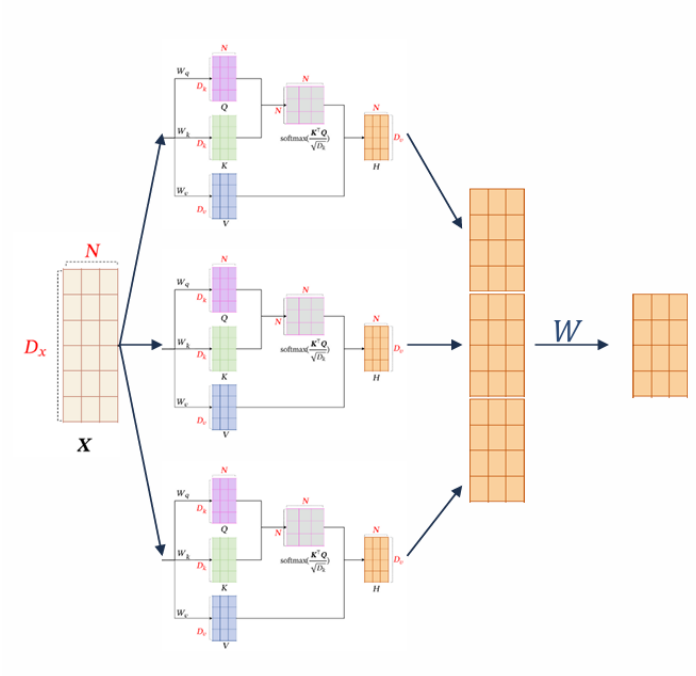


在 QKV 中，我输入的矩阵中，其长度代表为 N 的序列，宽度为其维度为 D，即相当于一列为一个词语。而里面的 W 是我要学习的参数。其中 QKV 三个矩阵和键值对注意力中所代表的含义是一样的，相当于我从输入的信息中筛选出三个不同的内容，QK 用于求权重，V 是筛选出信息，那么可以看作用 QK 对 V 进行加权处理，以得到我的最终注意力打分，即：

$$\boldsymbol{H} = \boldsymbol{V}\,\mathrm{softmax}(\frac{\boldsymbol{K}^{\top}\boldsymbol{Q}}{\sqrt{D_k}}),$$

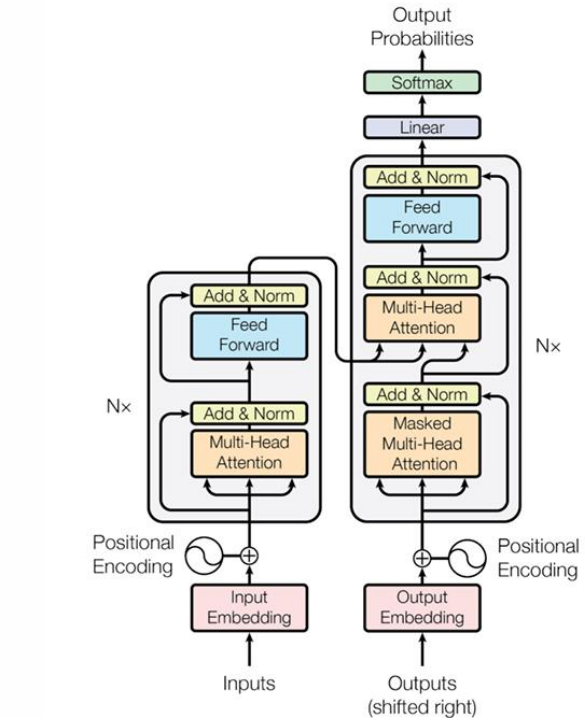
6.4.2 多头自注意机制

多头自注意力机制有点类似与我卷积神经网络中用好几个卷积核以提取不同的特征。其实就是多个自注意并起来。你需要知道多头自注意是 Transformer 的基本模块。



6.5 Transfromer

Transfromer 可分为两部分，一部分编码一部分解码。
编码部分有四个重要东西：
1. 位置编码：Transfromer 主要是处理序列数据的，但是他会丢掉顺序的信息，但是你语句中词语位置很重要，所以需要位置编码。
2. 层归一化
3. 直连边(残差)
4. 逐位 FFN：每个向量表示都过全连接层，先升维后降维，有两层。



上图就是 Trasnformer 的基本结构了，Add&Morm 就是直连边+层归一化。我们注意到，在解码部分的多头自注意力有 Masked(掩码)，其原因是我希望只有向前的注意力，而没有向后的注意力。(生成一句话当然只注意到前面不注意后面)

模型	每层复杂度	序列操作数	最大路径长度
CNN	$O(kLd^2)$	$O(1)$	$O(\log_k(L))$
RNN	$O(Ld^2)$	$O(L)$	$O(L)$
Transformer	$O(L^2d)$	$O(1)$	$O(1)$

k卷积核大小　L序列长度　d维度

自注意力计算复杂度 :$O(1)$

卷积网络计算复杂度 :$O(\log_k(n))$

循环网络计算复杂度 :$O(n)$

注意到，序列操作数为 1 的很方便做并行计算，反之则不便。最大路径长度可以说是第一个词与最后一个词之间的距离。RNN 中相当于比较远，所以有长程依赖问题；Transfromer 中每个距离都是 1，所以可以很好处理语言。

7 优化

在神经网络中，我们是想要获得平坦最小值而非尖锐最小值(泛化性差)，而优化算法就是随机梯度下降。

7.1 参数初始化

参数初始化可以有以下几种选择：
1. 随机初始化
2. 预训练初始化：通常情况下，一个已经在大规模数据上训练过的模型可以提供一个好的参数初始值。
3. 固定值初始化：就是根据经验来初始化，如一般偏置可初始化为0；LSTN 中遗忘门可初始化为 1 或 2，使时序信息变大，更易激活；使用 RLU 为激活函数的偏置可初始化为 0.01，便于激活。

是否可以将网络的参数全部初始化为 0？为什么？
如果参数都为 0，在第一遍前向计算时，所有的隐藏层神经元的激活值都相同；在反向传播时，所有权重的更新也都相同，这样会导致隐藏层神经元没有区分性．这种现象也称为对称权重现象。

7.2 Normalization(归一化方法)

归一化本质上就是一种规范化，把你的数值整理一下方便我继续运算。Layer Normalization 即层归一化是在 Transformer 中使用。而 Batch Normalization 可以在一定程度上缓解梯度消失，梯度爆炸的问题。

分析为什么批量归一化不能直接应用于循环神经网络？

循环神经网络每个神经元的激活值是随时间发生变化的，即 h_{t-1}、h_t、h_{t+1} 等不同，所以没办法批量归一化。

7.3 Regularization(正则化)

损害优化的方法就是正则化，目的是为了使其拟合能力下降，泛化性提升。通常就是增加约束(正则化项)，或者干扰过程(随机梯度下降、提前停止、丢弃法)。

试分析为什么不能在循环神经网络中的循环连接上直接应用丢弃法？
会损害神经网络在时间维度上的记忆能力。

7.4 梯度消失与梯度爆炸

梯度消失一般有如下解决方法：
1. 采用梯度较大激活函数，如 RLU。
2. 采用批量归一化。
3. CNN 中可采用残差网络和 LSTM。

梯度爆炸一般有如下解决方法：
1. 用合适参数初始化。
2. 梯度截断。
3. 采用 Adam 算法。

8 深度生成模型

深度生成模型是一种无监督模型。

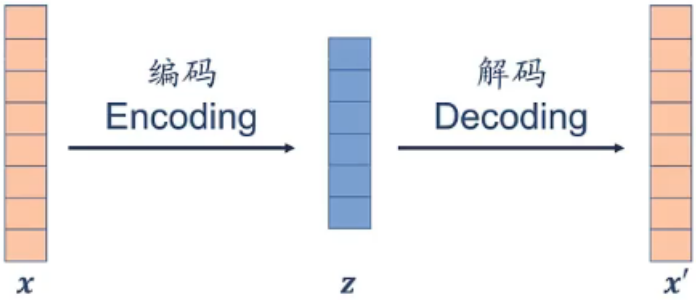
8.1 生成式模型与判别式模型

生成式模型（generative model）：估计或模拟样本的概率分布
对于分类问题，把类别标签看作是样本的一部分，用生成模型来学习样本和类别标签的联合概率分布，即：计算样本$\boldsymbol{x}$以及标签y联合概率分布$p(\boldsymbol{x}, y)$。
判别式模型（discriminative model）：对后验概率$P(y|\boldsymbol{x})$直接建

模。

8.2 自编码器 AE 与变分自编码器 VAE

编码就是从 x 映射到隐空间，解码就从隐空间映射到 x’，z 即需要学习的。

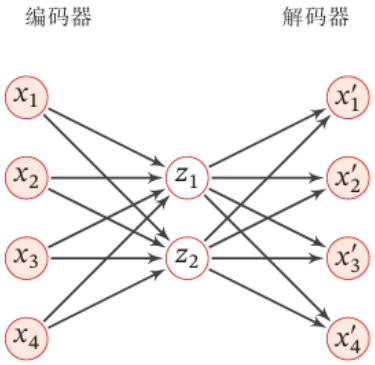


传统的编码和解码是线性关系的，而 EA 中这个关系是非线性，其损失函数是重构错误(误差)，本质上就是原先的减去编码加解码后。

$$\mathcal{L} = \sum_{n=1}^{N} ||\boldsymbol{x}^{(n)} - g(f(\boldsymbol{x}^{(n)}))||^2$$

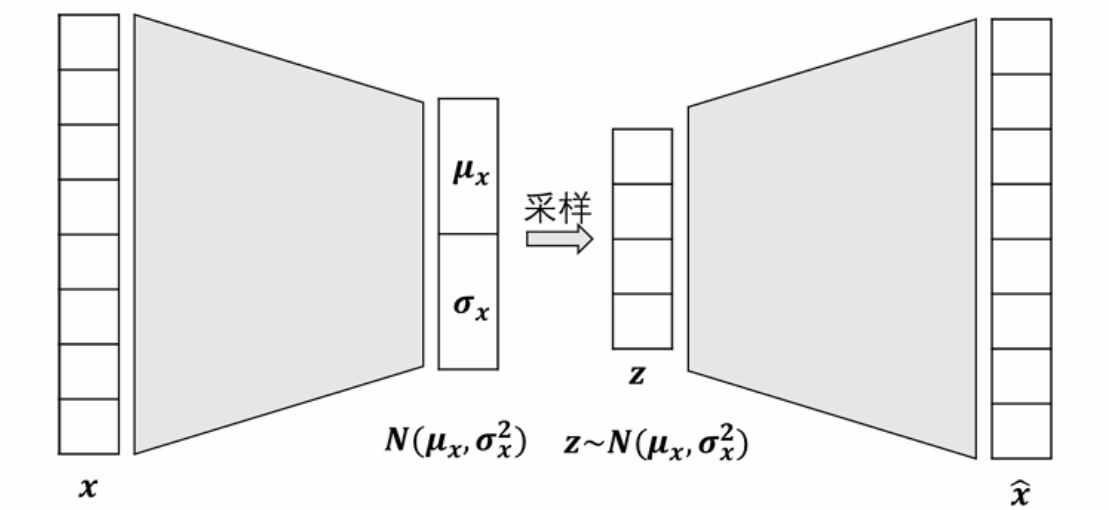
$$= \sum_{n=1}^{N} ||\boldsymbol{x}^{(n)} - f \circ g(\boldsymbol{x}^{(n)})||^2$$

也可说，AE 可以看作是把数据压入隐空间然后再进行恢复，可以用神经网络来实现 EA。



当然，你的网络隐藏层可能不只有一层，而是有很多层。

AE 存在局限性所以出现了 VAE。VAE 其实就是通过一个编码器学习到一个均值，一个方差；然后基于均值方差进行采样，得到隐变量(里面的分布就是高斯分布)，然后再根据隐变量恢复原来数据。



当然，采样是没办法计算梯度的，那我梯度下降不就是用不了了？所以采用再参数化。

8.3 生成式对抗网络 GAN

即用生成器(generator)和鉴别器（discriminator），二者反复进行对抗训练。

生成网络从隐空间（latent space）中随机采样作为输入，其输出结果需要尽量模仿训练集中的真实样本。
判别网络的输入则为真实样本或生成网络的输出，其目的是将生成网络的输出从真实样本中尽可能分辨出来。
两个网络相互对抗、不断调整参数，最终目的是使判别网络无法判断生成网络的输出结果是否真实。最后我可以用生成器进行以假乱真了。(生成网络来骗！来偷袭！辨别网络老同志)

条件 GAN，可根据类别控制，就是给的输入不仅有噪声还有增加类别参数。

9 结语

相信你经过前面的训练集的训练，你的自然神经网络已经学习好了，并且通过自注意力机制已经充分注意到了要点，希望你在测试集中的错误率很低，既不会过拟合又不会欠拟合。