

神经网络基础



前言

- 机器学习是实现人工智能的重要路径，而深度学习作为机器学习领域中一个新的研究方向，在计算机视觉、语音识别、自然语言处理等方向有着重要影响。深度学习源于人工神经网络的研究。本章主要介绍了神经网络的模型结构和训练方式。

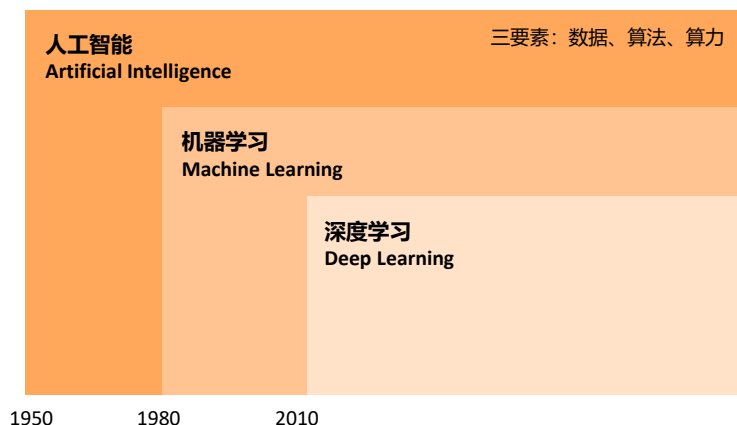
目标

- 学完本课程后，您将能够：
 - 了解什么是人工神经网络与深度前馈网络
 - 掌握如何训练神经网络
 - 掌握梯度下降的方法以及反向传播的概念
 - 了解神经网络架构设计的因素

目录

1. 深度学习预备知识
2. 人工神经网络
3. 训练神经网络
4. 神经网络的架构设计

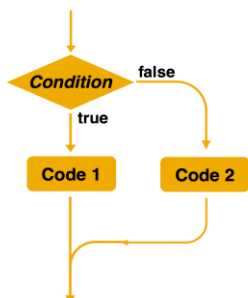
AI、机器学习、深度学习的关系



- 人工智能是个长久以来我们希望实现的目标：让计算机和人一样聪明。（生物的能力从后天学习得到，或者是生物的本能。机器学习是机器通过后天学习得到的能力，而除了机器学习，就是机器的本能，也就是用大量程序员的血与累铸成的“if”语句。）
- 机器学习是达成人工智能目标的一种手段。
- 深度学习是机器学习其中的一种方法。

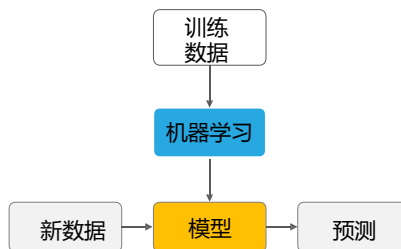
机器学习算法与传统基于规则的区别

基于规则的方法



- 使用显性编程来解决问题
- 规则可以被人工明确

机器学习



- 机器从数据中学习规律
- 规则由机器自动学习得到

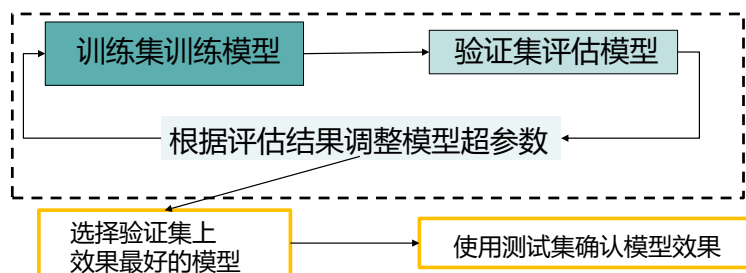
- 基于规则的方法好比机器的本能，程序怎么写，机器就怎么做。
- 机器学习是机器后天学习的能力，数据是机器学习关键。

数据集

- 特征 (feature) : 特征是用来描述机器学习系统处理的对象或事件的特性。
- 样本 (sample) : 样本是指我们从某些希望机器学习系统处理的对象或事件中收集到的已经量化的特征的集合。
- 数据集 (dataset) : 数据集是指很多样本组成的集合。有时我们也将样本称为数据集中的数据点 (data point) 。
- 大部分机器学习算法可以被理解为在数据集上获取经验。

数据集划分

- 训练集 (train dataset) : 用来构建机器学习模型, 从数据中确定模型参数的过程称为学习 (训练)。
- 验证集 (validation dataset) : 辅助构建模型, 用于在构建过程中评估模型, 进而调整模型超参数。
- 测试集 (test dataset) : 用来评估训练好的最终模型的性能。



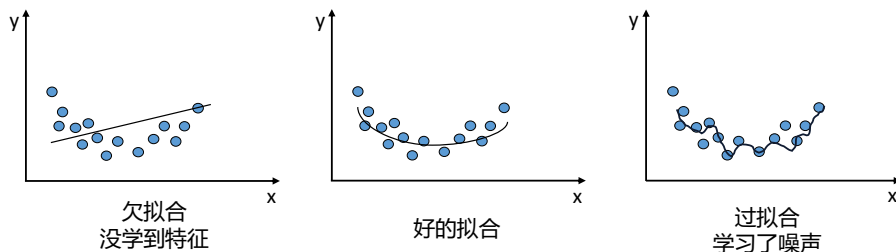
数据量不足

- 数据量不足：现有数据量不足以支撑模型所需的数据量。
- 解决方法：**数据增强**（Data Augmentation），数据增强又被翻译为数据扩充，核心思想是通过变换生成现有训练样本的变体，从而创建更多的训练数据，这些变换往往可以反映现实世界中会发生的变化，扩大数据集样本量，提升数据样本空间的复杂度。具体的方法需要根据处理的任务不同而具体制定。
 - **图像数据增强**常用方法：缩放、拉伸、加入噪点、翻转、旋转、平移、剪切、对比度调整、通道变化等。
 - **自然语言数据增强**常用方法：句子复述、同义词替换、回译、EDA等方式。
 - **语音数据增强**常用方法：剪裁、旋转、调音、加噪、SpecAugment等方式。

- EDA中的四种操作分别是：同义词替换，随机插入，随机交换，随机删除。
- 数据增强已经是深度学习的必需步骤了，其对于模型的泛化能力增加普遍有效。但是需要注意：
 - 不必做的太过，将原始数据量通过数据增加增加到2倍可以，但增加十倍百倍就只是增加了训练所需的时间，不会继续增加模型的泛化能力了。
 - 不增加无关的数据。

欠拟合和过拟合

- 欠拟合：模型过于简单导致训练误差泛化误差大。
- 过拟合：训练得到的模型的训练误差很小，而泛化能力较弱即泛化误差较大。



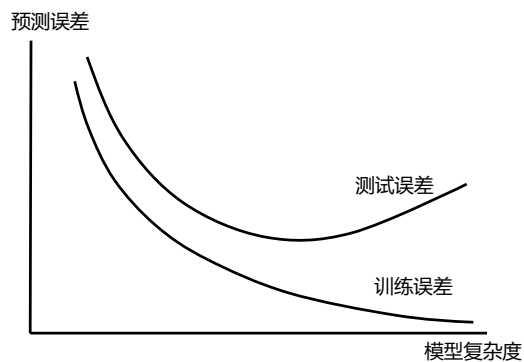
- **欠拟合与过拟合的区别：**欠拟合在训练集和测试集上的性能都较差，而过拟合往往能较好地学习训练集数据的性质，而在测试集上的性能较差。在神经网络训练的过程中，欠拟合主要表现为输出结果的高偏差，而过拟合主要表现为输出结果的高方差。

产生欠拟合的具体原因与解决方法

- 产生欠拟合的原因：
 - 特征量过少。
 - 模型复杂度过低，不能学习到数据的“一般规律”。
- 解决欠拟合的方法：
 - 增加新特征，可以考虑加入进特征组合、高次特征，来增大假设空间。
 - 添加多项式特征，这个在机器学习算法里面用的很普遍，例如将线性模型通过添加二次项或者三次项使模型泛化能力更强。
 - 减少正则化参数，正则化的目的是用来防止过拟合的，但是模型出现了欠拟合，则需要减少正则化参数。
 - 使用非线性模型，比如核SVM、决策树、深度学习等模型。
 - 调整模型的容量（capacity），通俗来说，模型的容量是指其拟合各种函数的能力。
 - 容量低的模型可能很难拟合训练集；使用集成学习方法，如boosting，将多个弱学习器集成为一个强学习器。

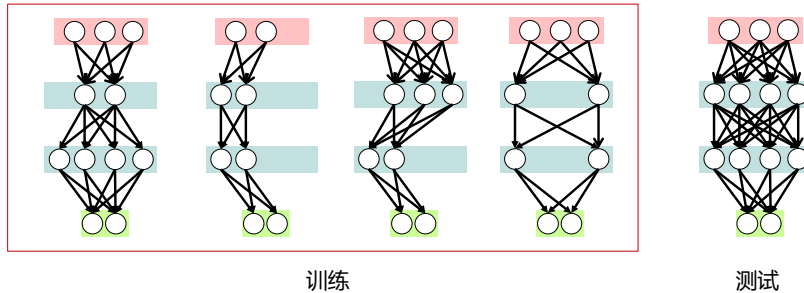
产生过拟合的具体原因与解决方法

- 产生过拟合的原因：
 - 数据噪声干扰过大。
 - 训练样本太少，无法描述问题的真实分布。
 - 模型复杂度过高。
- 解决过拟合的方法：
 - 减少特征维度。
 - 正则化：数据增强、参数范数惩罚、Dropout、提前终止等。



Dropout

- Dropout在训练过程中，随机的丢弃一部分神经元，此时丢弃部分对应的参数不会更新，让每个神经元能学习到更多的内容。
- 测试时用完整的网络，相当于一种bagging的方法。



机器学习场景的任务

- 机器学习可以解决多种类型的任务，下面列出最典型的几种：
 - 分类：模型输出的是一个类别，类别为离散的数据类型。比如计算机视觉中的图像分类算法解决的就是一个分类任务。
 - 回归：模型输出的是一个连续的数值。这类任务的一个示例是预测投保人的索赔金额（用于设置保险费），或者预测证券未来的价格。
 - 聚类：对大量未知标注的数据，按数据的内在相似性，将数据划分为多个类别，同一类别内的数据相似度较大，而类别间的相似性比较小。可以被运用在图片检索，用户画像等场景中。
 - 生成：这类任务中，输出的结果可以是向量，矩阵，序列等，输出结果中存在内在的关联。例如机器翻译和语音识别。
- 分类和回归是预测问题的两种主要类型，分类的输出是离散的类别值，而回归的输出是连续数值。

机器学习方法分类（1）

- 监督学习算法（Supervised Learning Algorithm）：训练含有很多特征的数据集，不过数据集中的每个样本都有一个标签（Label）或目标（Target）。
- 无监督学习算法（Unsupervised Learning Algorithm）：训练含有很多特征的数据集，但是数据集中所有样本都没有标签，通过无监督学习算法机器可以学习到数据特征中的内在关系。
- 自监督学习算法（Self-Supervised Learning Algorithm）：近几年兴起的学习方法，当数据集中只有数据的特征，没有标签时，依然可以通过某些方法来“制造”标签进行监督式学习。
- 监督的理解：监督学习中教员或者老师提供标签给机器学习系统，指导其应该做什么。在无监督学习中，没有教员或者老师，算法必须学会在没有指导的情况下理解数据的内在特征。

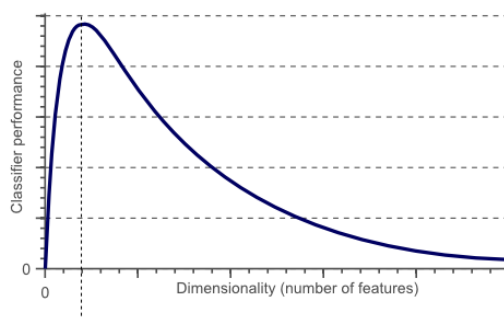
机器学习方法分类（2）

- 监督/非监督并非严格分类的。很多机器学习技术会组合使用。
- 半监督学习（Semi-Supervised Learning）：在一个任务中，试图让学习器自动地对大量未标记数据直接帮助少量有标记数据进行模型训练。
- 强化学习（Reinforcement Learning）：学习系统从环境到行为映射的学习，以使奖励信号（强化信号）函数值最大，强化学习不同于连接主义学习中的监督学习，主要表现在教师信号上，强化学习中由环境提供的强化信号是对产生动作的好坏作一种评价（通常为标量信号），而不是告诉强化学习系统如何去产生正确的动作。

- 半监督学习主要有两种主流的方法：一种是先进行无监督聚类，再进行监督式学习。另一种是Self-Learning。

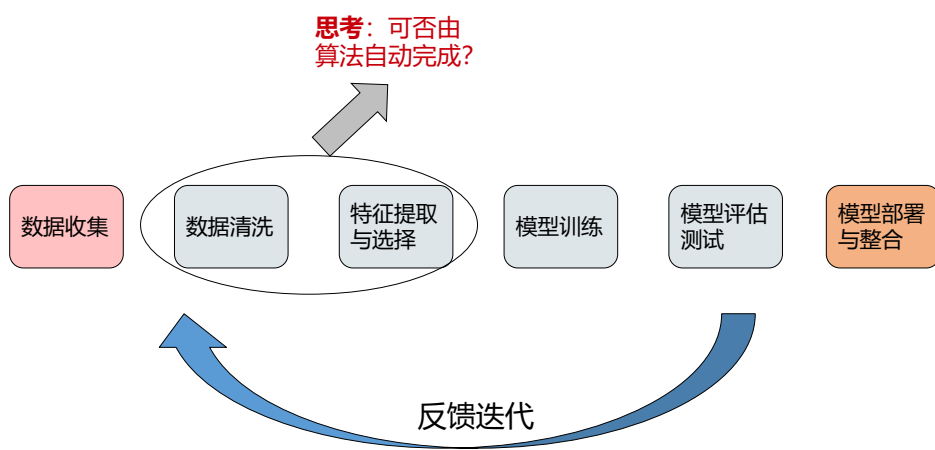
传统机器学习的挑战

- 维数灾难：当数据的特征维数很高时，随着特征维数的继续增加，很多机器学习模型的表现会变得越来越差，这种现象被称为维数灾难（curse of dimensionality）。



- 传统机器学习模型由于模型容量有限，因此，当特征维度很大时反而表现会很差，因此我们在传统机器学习项目中，需要做特征提取，选择。
- 此外，传统机器学习模型是有上限的，在很多任务中无法取得很好的效果。

机器学习的整体流程



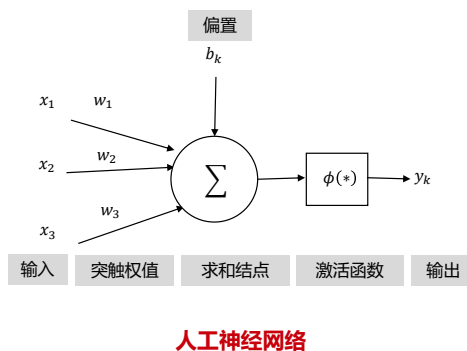
目录

1. 深度学习预备知识
2. **人工神经网络**
 - 神经网络简介
 - 感知机
 - 神经网络
 - 激活函数
3. 训练神经网络
4. 神经网络的架构设计

生物神经元与人工神经元

- 神经元是人脑的基本结构和功能单位之一。人脑中有数1000亿个神经元，其功能是接受（树突）、整合（细胞体）、传导（轴突）和输出（轴突末梢）信息实现信息转换。

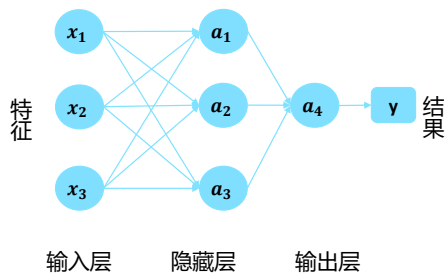
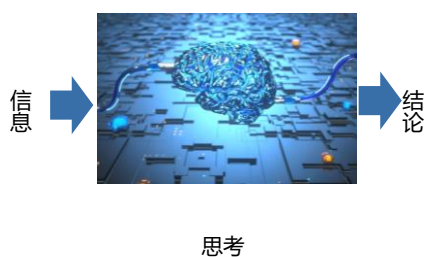
生物神经元	人工神经元
树突	输入信号
突触	突触权重
细胞体	求和节点
轴突	输出信号
兴奋性	激活函数
传导性	突触权重
感觉神经元（类型）	输入神经元
运动神经元（类型）	输出神经元
联络神经元（类型）	中间神经元



- 生物神经元特性：兴奋性、传导性。
- 神经元的类型（按功能划分）：感觉神经元、运动神经元、联络神经元。
- 感觉神经元（输入神经元），其树突的末端分布于身体的外周部，接受来自体内外的刺激，将兴奋传至脊髓和脑。
- 运动神经元（输出神经元），其轴突达于肌肉和腺体。运动神经元的兴奋可引起它们的活动。
- 联络神经元（中间神经元），介于上述两种神经元之间，把它们联系起来或组成复杂的网络，起着神经元之间机能联系的作用，多存在于脑和脊髓里。

人工神经网络

- 人工神经网络是一种旨在模仿人脑结构及其功能的信息处理系统。
- 人工神经网络，简称神经网络（Artificial Neural Network, ANN）：是由人工神经元互连组成的网络，它是从微观结构和功能上对人脑的抽象、简化，是模拟人类智能的一条重要途径，反映了人脑功能的若干基本特征，如并行信息处理、学习、联想、模式分类、记忆等。



深度学习的发展历史

- 1957: Frank Rosenblatt 提出感知机。
- 1969: Minsky和Papert提出感知机无法解决XOR问题。
- 1980: 多层感知机被提出。
- 1986: Hinton提出反向传播算法。
- 1989: George Cybenko提出万能逼近定理（为什么要这么多隐藏层呢？）。
- 2006: Hinton推广受限玻尔兹曼机。
- 2009: GPU被用于训练神经网络。
- 2012: Hinton团队在ILSVRC比赛中提出的AlexNet赢得冠军后，卷积神经网络飞速发展。
- 2016: AlphaGo以4:1击败了围棋世界冠军李世石，之后深度学习飞速发展。
- 2022: OpenAI发布了GPT-3.5，其产品ChatGPT瞬间成为全球爆品，引起了全球学术界和工业界的大语言模型热潮。
- 2024: OpenAI发布了SORA，一种通过文本生成视频的模型，实现了高质量的文本到视频生成。

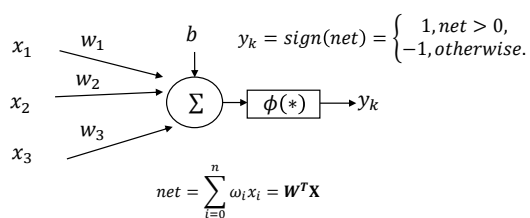
- 其实神经网络并不是一个很新的概念，早在1957年，神经网络的前身-感知机就被提出来了。
- 在1989年后，多层感知机一度被当做“脏话”，只要论文中提到了多层感知机，论文就一定会被拒绝。那怎么办呢，于是人们就想了个办法：换个名字，换成了高大上的“神经网络”和“深度学习”。
- 受限玻尔兹曼机早在86年就被提出，但是直到06年左右通过Hinton提出的快速学习算法才广为人知。其实06年推广受限玻尔兹曼机更像是“石头汤”的故事，受限玻尔兹曼机像是一个催化剂，让神经网络再次回到公众的视野中。而后续的研究也证明了RBM initialization其实并不是非常有效的。

目录

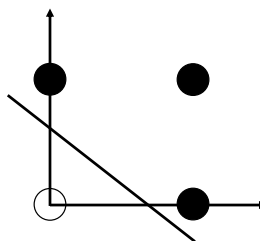
1. 深度学习预备知识
2. **人工神经网络**
 - 神经网络简介
 - 感知机
 - 神经网络
 - 激活函数
3. 训练神经网络
4. 神经网络的架构设计

感知器

- 1957年，美国学者Frank Rosenblatt（弗兰克·罗森布拉特）提出了感知器（感知机）算法。感知器用来接收多个信号，输出一个信号（由多个神经元组成）。每个输入信号具有一定的权重，计算多个输入信号的值与权重的乘积和，根据该结果（和）与指定的阈值进行比较，来决定该神经元是否被激活。
- 单层感知机本质上是一个**二分类器**。

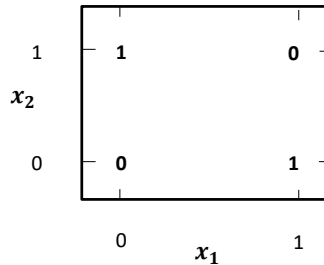


单层感知机



XOR问题

- 异或问题是指对于给定的两个二进制输入，通过异或逻辑门得到一个预测输出，当输入不相等时输出1，否则输出0。
- 由于单层感知机是线性模型，无法解决XOR等非线性数据问题。
- 1969年，美国数学家及人工智能先驱Minsky在其著作中证明了感知器本质上是一种线性模型，只能处理线性分类问题，无法处理非线性数据。



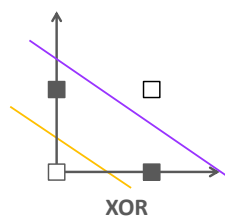
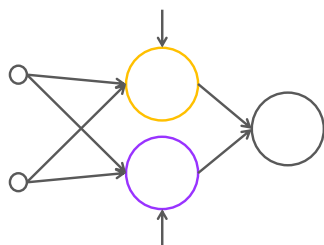
- 在人工神经网络领域中，感知器也被指为单层的人工神经网络，以区别于较复杂的多层感知器（Multilayer Perceptron）。作为一种线性分类器，（单层）感知器可以说是最简单的前向人工神经网络形式。尽管结构简单，感知器能够学习并解决相当复杂的问题。感知器主要的本质缺陷是它不能处理线性不可分问题。
- 单层感知器只能做线性分类。那么什么是线性呢？何为线性分类：用平面或超平面（超：三维以上不能可视化）分类样本点。
- 这时我们就引出多层感知器以解决非线性分类问题。我们需要在输入层和输出层之间多加一个隐藏层。一层隐藏层总的一个神经元对应着一条直线。这样我们可以在一个隐藏层中多加几个神经元，感知器就可以解决非线性分类问题了。

目录

1. 深度学习预备知识
- 2. 人工神经网络**
 - 神经网络简介
 - 感知机
 - 神经网络
 - 激活函数
3. 训练神经网络
4. 神经网络的架构设计

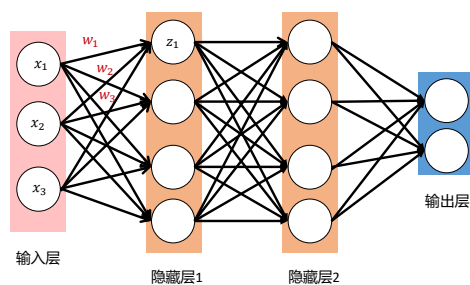
解决XOR问题：多层感知机

- 单个感知器的表达能力有限，它只能表达线性决策面（超平面）。如果我们把众多的感知器互联起来，就像人的大脑一样，我们就可以表达种类繁多的非线性曲面。



全连接神经网络

- 全连接神经网络是一种最简单的神经网络，各神经元分层排列。每个神经元只与前一层的神经元相连，同一层的神经元之间没有互相连接，层间信息的传送只沿一个方向进行。

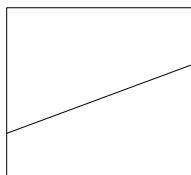


$$z_1 = x_1 \times w_1 + x_2 \times w_2 + x_3 \times w_3 + b$$

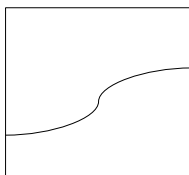
- 单个神经细胞功能比较简单，需要通过很多神经元一起协作完成复杂功能，通过一定的连接方式或信息传递方式进行协作的神经元可以看做是一个网络，就是人工神经网络。人工神经网络包括输入层、隐藏层和输出层。

隐藏层数量对神经网络的影响

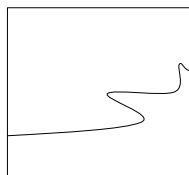
- 随着神经网络隐藏层的增加，网络的特征表达能力逐步增加，网络的分类能力也逐步增加。



隐藏层数0



隐藏层数1



隐藏层数20

神经网络的扩张

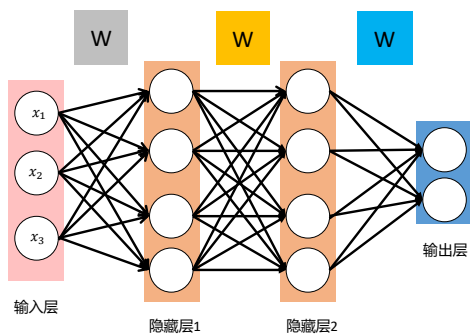
- 神经网络在感知器的模型上做了以下三点的扩张：
 - 加入**隐藏层**，隐藏层可以有多层，增强模型的表达能力。
 - 输出层的神经元也可以不止一个，**可以有多个输出**，模型就可以灵活的应用于分类回归，以及其他的机器学习领域比如降维和聚类等。
 - 对**激活函数做扩展**，包括Sigmoid函数，Softmax和ReLU等。

目录

1. 深度学习预备知识
- 2. 人工神经网络**
 - 神经网络简介
 - 感知机
 - 神经网络
 - 激活函数
3. 训练神经网络
4. 神经网络的架构设计

激活函数

- 我们可以将输入层神经元看作一个向量，每一层的权值看作一个矩阵。



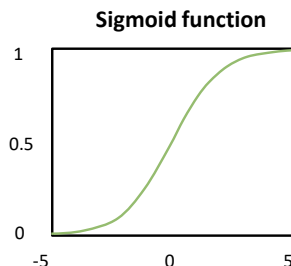
$$\begin{aligned} O^T &= I^T \times W \times W \times W \\ &= I^T \times \left(W \times W \times W \right) \\ &= I^T \times W \end{aligned}$$

- 如果不加入激活函数，神经网络就是线性的，无论加多少隐藏层都相当于只用了一层隐藏层，因此，我们需要加入激活函数使得神经网络变成非线性模型。

激活函数 - Sigmoid函数

- Sigmoid函数：单调连续，求导容易，输出有界，网络比较容易收敛。在物理意义上最为接近生物神经元。此外，(0, 1) 的输出可以被用于概率，或用于输入的归一化。但是其导数在远离中心点的位置，函数导数趋于0。网络大于5层容易产生梯度消失的状况。

$$f(x) = \frac{1}{1 + e^{-x}}$$

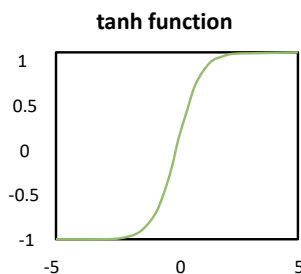


- Sigmoid函数单调连续，求导容易，输出有界，网络比较容易收敛。但是我们看到，在远离中心点的位置，Sigmoid函数导数趋于0，在网络非常深的时候，越来越多反向传播的梯度会落入饱和区，从而让梯度的模越来越小。一般来说，Sigmoid网络在5层之内，就会产生梯度退化为0的现象，难以训练。这种现象称为梯度消失现象。另外，Sigmoid的输出并不是以0为中心的。

激活函数 - tanh函数

- tanh函数：与sigmoid相比，它的输出均值是0，使得其收敛速度要比sigmoid快，减少迭代次数。tanh也存在梯度消失的问题。

$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



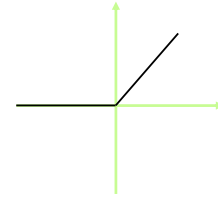
- tanh也具有和Sigmoid函数类似的缺点，即远离中心的位置导数趋于0，但是因为其关于0点对称，输出的均值比Sigmoid更加接近于0，因此SGD会更接近自然梯度，从而降低所需的迭代次数。

激活函数 - ReLU函数

- ReLU (Rectified Linear Unit) 函数：

- 优点：

- 实现简单，计算量小，收敛快速。
 - 有效缓解了梯度消失的问题。



$$\text{ReLU}(x) = \max(0, x)$$

$$y = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

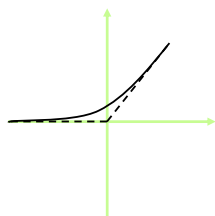
- 优点：

- 在生物神经网络中，同时处于兴奋状态的神经元非常稀疏，人脑中在同一时刻大概只有1-4%的神经元处于活跃状态。Sigmoid型激活函数会导致一个非稀疏的神经网络，而ReLU却具有很好的稀疏性，大约50%的神经元会处于激活状态。
 - 在优化方面，相比较sigmoid型函数的两端饱和，ReLU函数为左饱和函数，且在 $x > 0$ 时导数为1，在一定程度上缓解了神经网络的梯度小时问题，加速梯度下降的收敛速度。

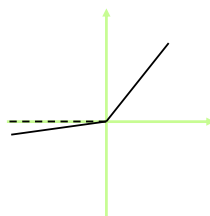
- 缺点：ReLU神经元在训练时比较容易死亡，在训练时，如果参数在一次不恰当的更新后，第一个隐藏层中的某个ReLU神经元在所有的训练数据上都不能被激活。那么这个神经元自身参数的梯度永远都是0，在以后的训练过程中永远不能被激活，这种现象称为死亡ReLU问题，并且也有可能会发生在其他隐藏层。

激活函数 - Softplus、Leaky ReLU

- Softplus: 在ReLU的基础上, 增加了一点计算复杂度, 但是其有连续的导数, 且定义的曲面也相对平滑。
- Leaky ReLU: 保留了一些负轴的值, 使得负轴的信息不会全部丢失。



Softplus函数



Leaky ReLU函数

激活函数 - Softmax

- Softmax函数：

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

- Softmax函数的功能就是将一个K维的任意实数向量映射成另一个K维的实数向量，其中向量中的每个元素取值都介于（0， 1）之间。新的向量所有维度模长之和为1。
- Softmax函数经常用作多分类任务的输出层。

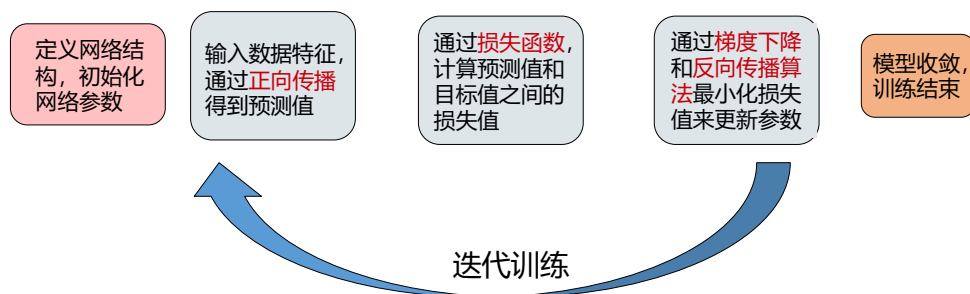
激活函数设计需考虑的因素

- 非线性：当激活函数是非线性的，一个两层神经网络可以证明是一个通用函数近似值，如果失去了非线性，整个网络就相当于一个单层的线性模型。
- 连续可微性：这个属性对基于梯度优化方法是必要的，如果选择了一些具有局部不可微的函数，则需要强行定义此处的导数。
- 有界性：如果激活函数有界的，基于梯度的训练方法往往更稳定；如果是无界的，训练通常更有效率，但是训练容易发散，此时可以适当减小学习率。
- 单调性：如果激活函数是单调的，与单层模型相关的损失函数是凸的。
- 平滑性：有单调导数的平滑函数已经被证明在某些情况下泛化效果更好。

目录

1. 深度学习预备知识
2. 人工神经网络
- 3. 训练神经网络**
4. 神经网络的架构设计

如何训练神经网络



- 神经网络有大量的参数需要学习，每个神经元的权重 W 和偏置 b 。
 - 先随机地给参数赋值。
 - 计算网络在训练数据集上的损失函数值，并不断地调整参数，使得损失函数值最小。这个学习的过程称为最优化。
- 神经网络的最优化算法一般都使用梯度下降法。

损失函数

- 神经网络中使用的代价函数被称作损失函数（Loss Function）。
 - 损失函数衡量了预测与真实样本标签的距离。
 - Loss的值都会设置为和吻合程度负相关。如果算法公式是正相关，定义损失函数时候加负号，调整为负相关。
- 神经网络在数据集上学习的原理，就是通过对训练数据集的学习，调整神经网络中每个神经元的参数，使得损失Loss达到最低。

深度学习中常用的损失函数

- 交叉熵误差刻画了两个概率分布之间的距离，是分类问题中使用较多的一种损失函数。
- 一般均方误差更多的用于回归问题，而交叉熵误差更多的用于分类问题。

均方误差损失

$$MSE = \frac{1}{2n} \sum_{i=1}^n (f(x_i) - y_i)^2$$

交叉熵损失

$$CE = \frac{1}{2n} \sum_{i=1}^n \sum_{c=1}^M -y_{ic} \log(p_{ic})$$

- 均方误差损失
 - 其中n表示一个批次所有数据的数量，i表示第i个数据，f(x)为预测值，y为标签值。
- 交叉熵损失
 - M：类别的数量。
 - y_{ic} ：指示变量（0或1），如果该类别和样本i的类别相同就是1，否则就是0。
 - p_{ic} ：对于观测样本i属于类别C的预测概率。

损失函数的极值

- 目的：损失函数 $E(W)$ 是定义在权值空间上的函数。我们的目的是搜索使得 $E(W)$ 最小的权值向量 W 。
- 限制： $E(W) = \frac{1}{2n} \sum_{i=1}^n (f(x_i) - y_i)^2$ 的复杂高维曲面，在数学上，尚没有求极值解有效方法。
- 解决思路：负梯度方向是函数下降最快的方向，我们可以从某个点开始，沿着 $-\nabla E(W)$ 方向一路前行，期望最终可以找到 $E(W)$ 的极小值点，这就是梯度下降法的核心思想。
- 梯度下降：蒙着眼睛下山。



梯度下降法（1）

- 梯度下降实则就是求损失值对所有参数的偏导（梯度），梯度的意义是变化最快的方向，因此可以沿着梯度的负方向更新所有参数。学习率（Learning Rate, LR）是梯度下降中重要的超参数，根据误差梯度调整权重数值的系数，通常记作 η 。

$$w' = w - \eta * \frac{\partial Loss}{\partial w}$$

- 通过学习率和梯度值更新所有参数值使得网络的损失值降低。

- 如何从初始位置到函数极值点？
 - 每一步都朝着函数值下降最快的方向移动一段距离。
- 函数下降（上升）最快的方向：函数值变化率最大的方向。
 - **导数**：函数沿坐标轴正方向的变化率。
 - **方向导数**：函数沿着任意方向的变化率。（多个）
 - **最大方向导数：梯度**。梯度方向即函数值变化率最大的方向，也就是我们需要沿着梯度方向寻找函数极值。
- 步长：一段距离，由经验获取的，可以在尝试过程中不断调整。

梯度下降法（2）

- 不同的随机起点可能到达不同的最小值位置。
- 神经网络每次训练的结果可能会不一样。
- 只要最终的Loss值降低到有效范围内，并且在任务的性能指标上达标，我们就认为是有效的结果。

梯度下降法 (3)

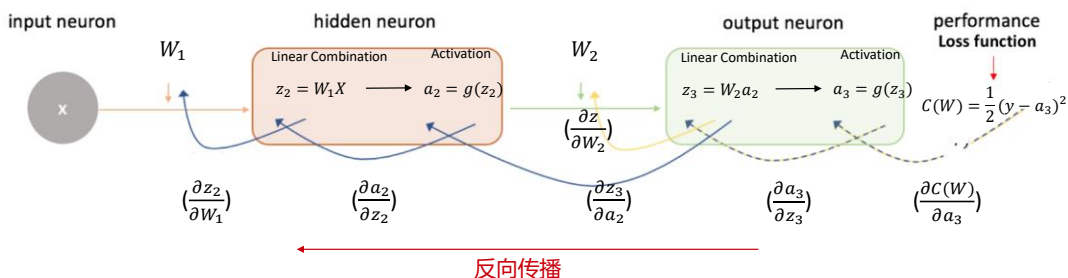
- 根据批量大小，梯度下降常用的方法有三种：
 - 批量梯度下降 (BGD)：每次更新使用所有的训练数据，最小化损失函数，如果只有一个极小值，那么批量梯度下降是考虑了训练集所有数据，但如果样本数量过多，更新速度会很慢。
 - 随机梯度下降 (SGD)：每次更新的时候只考虑了一个样本点，这样会大大加快训练速度，但是函数不一定是朝着极小值方向更新，且SGD对噪声也更加敏感。
 - 小批量梯度下降 (MBGD)：MBGD每次更新的时候会考虑一定数量(batchsize)的样本，解决了批量梯度下降法的训练速度慢问题，以及随机梯度下降对噪声敏感的问题。

反向传播算法（1）

- 反向传播算法（Back Propagation）是神经网络的重要算法。我们已经知道了可以通过梯度下降来更新模型的参数，但是由于神经网络中网络层数非常多，在求损失值对参数的偏导时，根据链式法则，会有大量的计算。而反向传播算法通过动态规划的思想，可以大幅提高梯度下降算法的计算效率。
 - 成熟的深度学习框架，如TensorFlow，会自动完成反向传播和求导的部分。我们在使用时只需要定义前向运算。
- 步骤：
 - 将损失函数值反向传播给每个计算单元。
 - 每个计算单元根据获得的误差更新权重。

反向传播算法（2）

- 反向传播算法（Backpropagation）是神经网络的重要算法。它使用链式求导法则将输出层的得出误差反向传回给网络，使神经网络的权重有了较简单的梯度计算实现方法。



- 如果我们直接通过最后的损失函数值用梯度下降算法优化模型，即调整参数使目标函数最小的话。每层的参数都是一层一层嵌套的，求解梯度的过程会十分复杂，且计算量很大。这是我们进入反向传播算法，反向传播等式为我们提供了一个计算代价函数梯度的方法。前向误差反向传播回每个神经元，再用梯度下降更新参数。
- 前向传播指的是信息的前向传播，而反向传播是只误差的反向传播。
- 反向传播等式为我们提供了一个计算代价函数梯度的方法。
- 反向传播的主要步骤：误差按照权重的比例分别传给每个计算单元（隐藏层的神经元），得出每个神经元需要对最后产生的误差付多少责任。然后在根据分得的误差更新权重。

总结：网络训练流程



梯度下降法的缺点

- 没办法保证收敛性
 - 学习率太小，收敛慢；学习率太大，损失函数会在极小值处动荡甚至偏离。
 - 容易陷在局部极小值处，或者鞍点处。
- 梯度消失与梯度爆炸
 - 梯度消失：当激活函数为Sigmoid时， $\sigma'(z)$ 的最大值为 $\frac{1}{4}$ ，而我们初始化的网络权值 $|w|$ 通常都小于1，因此 $|\sigma'(z)w| \leq \frac{1}{4}$ ，对于上面的链式求导，层数越多，求导结果越小，因而导致梯度消失的情况出现。
 - 梯度爆炸： $|f'(z)w| > 1$ ，也就是当 w 比较大的时候会出现梯度爆炸。但由于初始化参数的时候，通常会用较小的值来初始化参数，因此梯度爆炸的情况并不会很常见。此外，通过简单的梯度裁剪就能有效地解决梯度爆炸问题。

- 梯度消失的原因：
 - 一是在深层网络中。
 - 二是采用了不合适的损失函数，比如sigmoid。
- 梯度爆炸：一般出现在深层网络和权值初始化值太大的情况下。

优化器

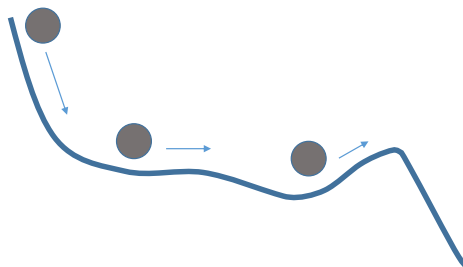
- 在梯度下降算法中，有各种不同的改进版本。在面向对象的语言实现中，往往把不同的梯度下降算法封装成一个对象，称为优化器。
- 算法改进的目的，包括但不限于：
 - 加快算法收敛速度。
 - 尽量避过或冲过局部极值。
 - 减小超参数的设置难度，主要是Learning Rate (LR) 。
- 常见的优化器如：普通SGD优化器、动量优化器、Nesterov、Adagrad、RMSprop、Adam等。

动量优化器

- 想象一个小球，从一个随机的点开始，沿着误差曲面滚下，小球在惯性的加持下会越滚越快。动量优化器（Momentum）通过加入 γv_{t-1} ，可以加速SGD，并且抑制震荡。

$$v_t = \gamma v_{t-1} - \eta \nabla_{\theta} J(\theta)$$
$$\theta \leftarrow \theta + v_t$$

- 其中， v_{t-1} 为上次更新时的反向梯度， $\nabla_{\theta} J(\theta)$ 为本次更新的反向梯度， η 为学习率， $0 \leq \gamma < 1$ 是一个常数，称为动量（Momentum），表示要在多大程度上保留原来的更新方向。



- 我们在寻找损失函数极值点的过程类似于一个小球，从一个随机的点开始，沿着误差曲面滚下，此时的小球会有一定的惯性。动量，它模拟的是物体运动时的惯性，是一个物理学概念，即更新的时候在一定程度上保留之前更新的方向，同时利用当前batch的梯度微调最终的更新方向。举个例子：上次更新时是向前走的，这次更新的梯度算出来时向左走，这变化太剧烈了，所以我们来做个折中，往左前方走，感觉像是上次更新还带有一定的惯性。这样一来，可以在一定程度上增加稳定性，从而学习地更快，并且还有一定摆脱局部最优的能力。

Adagrad优化器

- 随机梯度下降算法（SGD）、动量优化器的共同特点是：对于每一个参数都用相同的学习率进行更新。
- Adagrad的思想则是应该为不同的参数设置不同的学习率，且学习率会随着模型训练不断变化。



大学习率

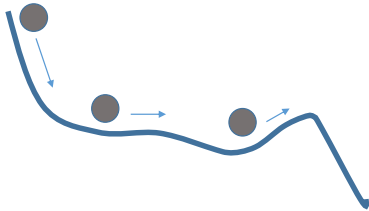


小学习率

- Adagrad优化器好比打高尔夫球，开始的时候由于离目标点较远，需要使用比较大的力，以减少更新次数，而当离目标点较近时，需要用比较小的力让球进入目标点。

Adam优化器

- Adam为每个待训练的参数，同时考虑了可调节的动量项以及可调节的学习率。



+



目录

1. 深度学习预备知识
2. 人工神经网络
3. 训练神经网络
- 4. 神经网络的架构设计**

设计网络结构

- 在神经网络的结构设计方面，往往遵循如下要点：
 - 输入层的单元数等于样本特征数。
 - 在分类问题中，输出层的单元数一般等于分类的类型数。
 - **每个隐层的单元数通常是越多分类精度越高，但是也会带来计算量的增加。**
 - **增加深度的效率要比增加宽度高得多。**

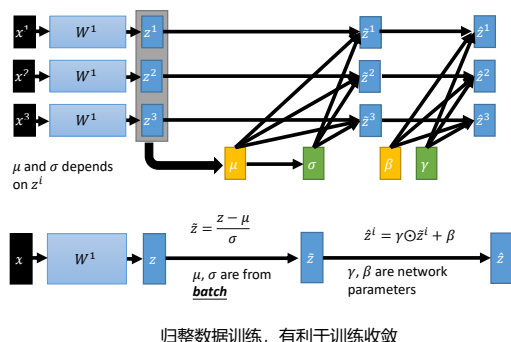
网络越深，模型精度越大？

- 在深度学习中，网络层数增多一般会伴着下面几个问题：
 - 计算资源的消耗（可以通过GPU集群解决）。
 - 模型容易过拟合（可以通过增大数据集、Dropout等正则化方法避免）。
 - 梯度消失、梯度爆炸问题的产生（可以通过初始化、批标准化等方法避免）。
- 从上述可以看出，理论上说，网络层数越多，那么模型效果越好。实际上，网络发生了退化（Degradation），该如何解决？

- 在神经网络中，网络的深度是非常重要的。一般认为，随着深度的增加，神经网络可以提取到更加抽象、更加语义化的信息。但是人们发现，直接在神经网络中堆叠层，会导致如上的一些问题，但是这些问题也都有一些相应的方法进行避免。貌似我们只要无脑的堆叠层，就可以从中获益。
- 然而，随着层数的增加，网络出现了退化现象：精度出现饱和并且迅速下降。实验证明，退化问题并不是过拟合造成的，因为即使在训练集上，增加层数也会导致精度下降。退化问题表明，深层神经网络不能通过简单的增加深度进行优化。

Batch Normalization

- 解决在训练过程中，中间层数据分布发生改变的问题，加快训练速度。
- 2015年Google发表paper “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift” ，而后迅速被广泛采用。

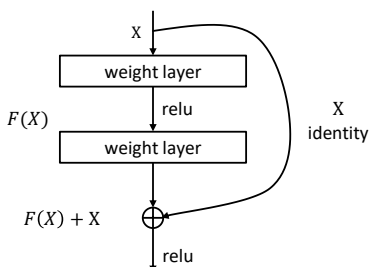


- BN通过一定的规范化手段，把每层神经网络任意神经元的激活输入值的分布强行拉回到均值为0方差为1的标准正态分布，使得激活输入值落在非线性函数对输入比较敏感的区域，这样输入的小变化就会导致损失函数较大的变化，让梯度变大，避免梯度消失问题产生。
- Batch Normalization的优点：
 - BN使得网络中每层输入数据的分布相对稳定，加速模型学习速度。BN通过规范化与线性变换使得每一层网络的输入数据的均值与方差都在一定范围内，使后一层网络不必不断去适应底层网络中输入的变化，从而实现了网络中层与层之间的解耦，允许每一层进行独立学习，有利于提高整个神经网络的学习速度。
 - BN使得模型对网络中的参数不那么敏感，降低了神经网络模型训练和调参的复杂度，使得网络学习更加稳定，对于初始化要求没那么高，而且可以使用大的学习率。
 - BN可支持更多的激活函数（例如sigmoid, tanh, relu等），缓解梯度消失问题。随着Sigmoid层次的增加，梯度衰减的很快，也就无法用在深层的神经网络。ReLU的问题是可能导致神经元的死亡（即ReLU掉入0的区域）。Batch normalization可以规范化输入，从而可以使用这些激活函数。
 - BN具有一定的正则化效果。在Batch Normalization中，由于我们使用mini-batch的均值与方差作为对整体训练样本均值与方差的估计，尽管每一个batch中的数据都是从总体样本中抽样得到，但不同mini-batch的均值与方差会有所不同，这就为网络的学习过程中增加了随机噪音，与Dropout通过关闭神经元给网络训练

带来噪音类似，在一定程度上对模型起到了正则化的效果。

残差结构

- 解决方法：让若干层映射为一个层，这样深层网络就会等同于浅层网络，从而避免退化问题。
- 深度残差网络（Residual Network, ResNet）的核心思想就是引入一个恒等快捷连接（Identity Shortcut Connection），直接跳过一个或多个层使训练数百甚至数千层成为可能，且在这种情况下仍能展现出优越的性能。



- ResNet提出当网络层数过深时，网络的性能反而会下降，这是由于梯度消失造成的。而ResNet提出可以用残差结构来缓解梯度消失的问题。

思考题

1. （单选题）以下哪个是反向传播算法的理论基础？
- A. 计算图
 - B. 链式法则
 - C. 代价函数
 - D. 高阶微分

- 答案：B

本章总结

- 本章节介绍了人工神经网络的基础以及神经网络的基本概念，又介绍了深度前馈网络，再介绍了如何训练神经网络，最后介绍了如何设计神经网络。

缩略语

- ANN: Artificial Neural Network, 人工神经网络
- MSE: Mean Squared Error, 均方误差
- CE: Cross Entropy, 交叉熵
- BGD: Batch Gradient Descent, 批量梯度下降法
- SGD: Stochastic Gradient Descent, 随机梯度下降法
- MBGD: Mini-batch Gradient Descent, 小批量梯度下降法

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。
Bring digital to every person, home, and
organization for a fully connected,
intelligent world.

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.



图像处理理论和应用



前言

- 近年来，随着深度学习在图像识别领域取得巨大突破，掀起了以神经网络为基础的深度学习研究热潮。当前深度学习在图像处理领域的应用可分为图像处理、图像识别、图像生成等多个方面。本章将在介绍基础图像处理理论之上，着重介绍图像识别、图像生成等相关理论与实践。

目标

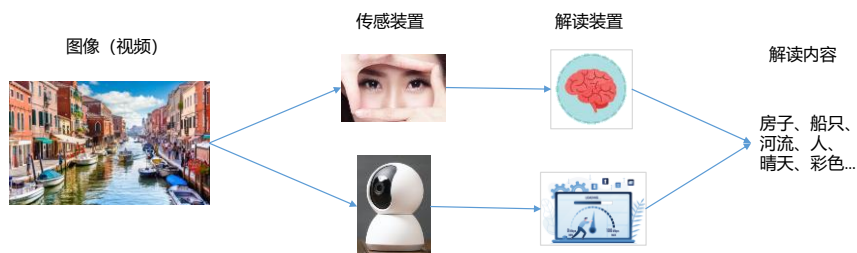
- 学完本课程后，您将能够：
 - 描述计算机视觉技术概览
 - 掌握数字图像处理基础和图像预处理技术
 - 理解图像处理基本任务
 - 区分传统特征提取算法和深度学习卷积神经网络
 - 了解图像识别和图像生成的相关理论

目录

1. 计算机视觉概览
2. 数字图像处理基础
3. 图像预处理技术
4. 图像处理基本任务
5. 特征提取和传统图像处理算法
6. 卷积神经网络和图像识别
7. 生成对抗网络和图像生成

计算机视觉

- 人的视觉所感知的周围环境是三维结构，而计算机中的图像是二维的。计算机视觉就是研究如何让计算机像人一样“看”的科学。
- 计算机视觉是使计算机能理解采集设备采集的图像视频的一门学科。形象地说，就是给计算机安装上眼睛（照相机）和大脑（算法），让计算机能够感知环境。

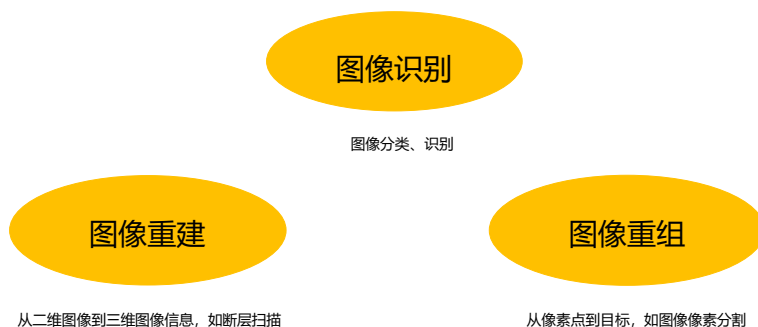


计算机视觉的应用

- 计算机视觉技术已经在许多领域得到了广泛的应用，以下是一些典型的例子：
 - 公安安防：人脸识别，指纹识别，场景监控，环境建模。
 - 生物医学：染色体分析，X光、CT图像分析，显微医学操作。
 - 文字处理：文字识别，文档修复，办公自动化，垃圾邮件分类。
 - 国防军事：资源探测，军事侦察，导弹路径规划。
 - 智能交通：公路交通管理，电子警察执法抓拍系统，自动驾驶车辆。
 - 休闲娱乐：电影特效，视频编辑，人像美颜，体感游戏，VR。

计算机视觉核心问题

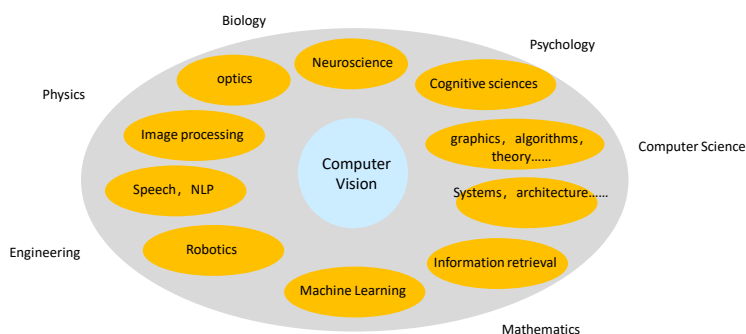
- 计算机视觉核心问题：图像识别（Recognition），图像重建（Reconstruction）和图像重组（Reorganization）。



- 图像识别：图像到目标
- 图像重建：像素到像素
- 图像重组：像素到目标

计算机视觉相关学科

- 计算机视觉是一门研究图像理论、技术和应用的交叉学科。计算机视觉不仅和传统的数学、物理学、生理学、心理学、计算机科学、电子工程等学科相关，并且还涉及到计算机图形学、图像模式识别、图像工程等专业技术。



计算机视觉与人工智能

- 计算机视觉中的大部分理论运用了人工智能的技术。人工智能的发展离不开计算机视觉，计算机视觉中的很多应用问题给人工智能技术提供了研究方向。
- 人工智能在计算机视觉中最成熟的技术方向是图像识别，它实现了如何让机器理解图像中的内容。



华为云图像内容识别



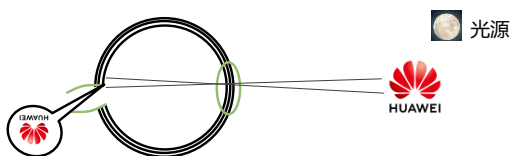
华为云人脸检测

目录

1. 计算机视觉概览
- 2. 数字图像处理基础**
3. 图像预处理技术
4. 图像处理基本任务
5. 特征提取和传统图像处理算法
6. 卷积神经网络和图像识别
7. 生成对抗网络和图像生成

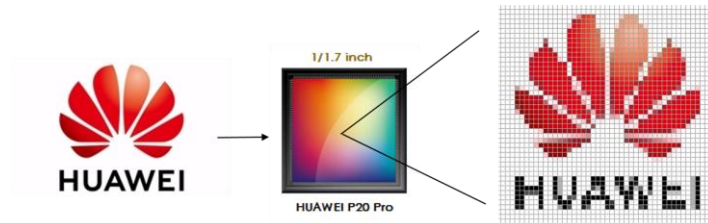
人眼成像

- 人眼平时见到的图像都是由**照射源**和形成图像的场景元素对光能的反射和吸收而产生的。照射源发出光线照射到物体，经过反射或是折射，光线进入到人眼中，然后看到物体。
- 如果我们用函数 $f(x, y)$ 表示图像， $f(x, y)$ 则由两部分组成，包括入射到被观察场景的光源分量 $i(x, y)$ ，和场景中物体所反射的光源分量 $r(x, y)$ 。 $i(x, y)$ 的大小由照射源决定，而 $r(x, y)$ 的大小由成像物体的特性决定。



图像的感知和获取

- 数字图像的采集过程类似人眼，使用大量的光敏传感器构成阵列以获取图像。
- 图像获取的过程包括：通过将输入电能和光敏传感器材料相结合，把输入能源转化为电压，输出的电压波再经过取样和量化便可得到离散的数字图像。
- 成像的质量由传感器的单元数，尺寸和传感性能决定。

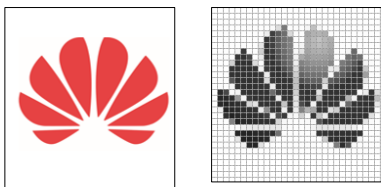


场景元素经过成像系统得到数字化后的图像

- 角膜和晶状体：镜头
- 视网膜：感光单元

图像数字化

- 多数传感器的输出是连续的电压波形，图像数字化是将连续色调的图像转换为计算机能够处理的数字影像的过程。



使用30*30的灰度传感器得到的数字化图片

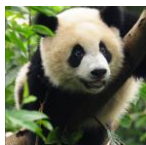
- 图像数字化包括两种处理过程：采样和量化。

采样

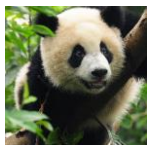
- 图像数字化的采样过程是将空间上连续的图像变化为离散的点。
- 采样后得到离散图像的尺寸称为图像分辨率。分辨率由宽（width）和高（height）两个参数构成。宽表示水平方向的细节数，高表示垂直方向的细节数。
- 例如：
 - 一幅640*480分辨率的图像，表示这幅图像是由 $640*480=307200$ 个点组成。
 - 一幅1920*1080分辨率的图像，表示这幅图像是由 $1920*1080=2073600$ 个点组成。

不同分辨率采样的效果

- 采样间隔越小，图像采样点数越多，空间分辨率高，图像质量好，但数据量大。
- 下方展示了图片的分辨率从1000x1000依次降低到10x10的图像效果。



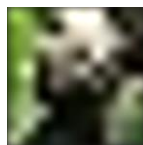
1000x1000



300x300



50x50



10x10

量化

- 数字图像处理的量化过程是将采样点的传感器信号转换成离散的整数值。
- 量化后得到离散图像的每个采样点的变化范围称为灰度级（Gray level scale）。灰度级通常是2的整数次幂。我们用m级或者n位来表示灰度级。图像数据的灰度级越多视觉效果就越好。计算机中最常用的是8位图像。
- 例如：
 - 一幅8位的图像，表示每个采样点有 $2^8=256$ 级。
 - 从最暗到最亮，可以分辨256个级别。
 - 一幅32级的灰度图像，每个采样点从最暗到最亮，可以分辨32个级别。

- 量化的主要工作就是将幅度上连续取值的每一个样本转换为离散值表示。

不同灰度级量化的效果

- 量化等级越多，图像层次越丰富，灰度分辨率高，图像质量好，但数据量大。
- 下方展示了图像的灰度级从256级降低到2级的图像效果。



基本术语：亮度

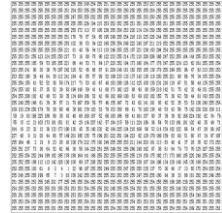
- 图像亮度通俗理解便是图像的明暗程度。
- 下方展示了图像的亮度由低至高的效果。



灰度图像的表达

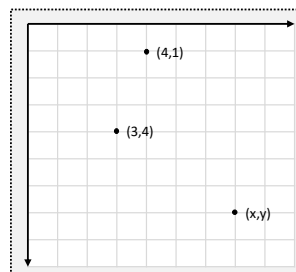
- 取样和量化后的数字图像在数学上表示为一个矩阵。我们使用矩阵 I 或者函数 $f(x, y)$ 来表示一幅数字图像。

$$f(x, y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,M-1) \\ f(1,0) & f(1,1) & & f(0,M-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(N-1,0) & f(N-1,1) & \cdots & f(N-1,M-1) \end{bmatrix}$$



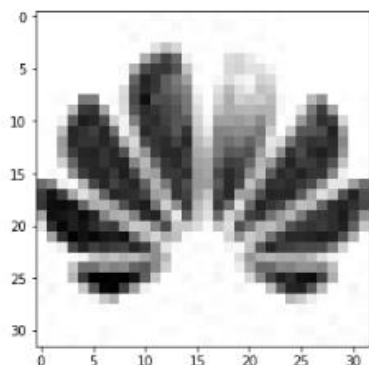
像素的描述

- 数字图像中的每一个采样点，被称之为像素（Pixel）。像素的位置可以由它在图像中的位置坐标 (x, y) 来描述，像素的灰度值可以表示为 $[0 \sim 255]$ 。
- 对于一幅 $M \times N$ 分辨率的数字图像，通常我们把左上角的像素记为原点 $(0, 0)$ 。水平方向的坐标范围是 $[0, M - 1]$ ，垂直方向的坐标范围是 $[0, N - 1]$ 。
- 像素的数据维度被称为通道（Channel）。



数字图像的存储

- 前面我们介绍的灰度图，都只有一个通道。通常，单通道记录了像素点的亮度信息，例如8位的图像，0表示最暗（黑色），255表示最亮（白色）。

[illegible]

- 我们知道计算机中处理的都是数字，那么想要让计算机“看懂”一张图片，首先要做的就是将图像转化为数字表示。
- 一张图像是由众多像素组成的。像素是指由图像的小方格组成的，这些小方块都有一个明确的位置和被分配的色彩数值，小方格颜色和位置就决定该图像所呈现出来的样子。可以将像素视为整个图像中不可分割的单位或者是元素。不可分割的意思是它不能够再切割成更小单位抑或是元素，它是以一个单一颜色的小格存在。
- 如图所示，黑白图像对于计算机来说，图像是一个由数字组成的巨大的矩阵。每个数字都是在范围0-255之间的整型，其中0表示全黑，255表示全白。

彩色图像 - RGB

- 为了表示彩色图像，我们需要使用多通道数字图像。最普遍的方式是使用RGB颜色空间。RGB颜色空间中每个像素点有三个维度，分别记录在红（Red）、绿（Green）、蓝（Blue）三原色分量上的亮度。
- 例如，按照 (R, G, B) 的方式：

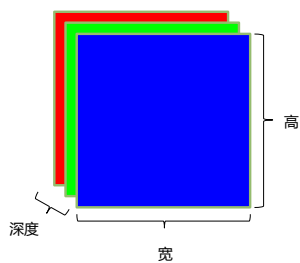
▫ $(255, 0, 0)$ 纯红



▫ $(124, 252, 0)$ 草坪绿

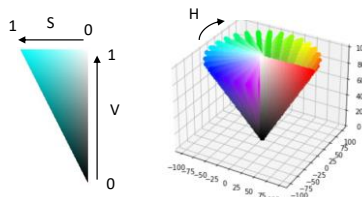


▫ $(135, 206, 235)$ 天蓝色



彩色图像 - HSV

- 另一种常用的颜色空间是HSV，该颜色空间可以用一个圆锥来表示。HSV表示色相（Hue）、饱和度（Saturation）和亮度（Value）。H表示颜色的相位角（Hue），取值范围是0到360度；S表示颜色的饱和度（Saturation），范围从0到1，它表示成所选颜色的纯度和该颜色最大的纯度之间的比率；V表示色彩的明亮程度（Value），范围从0到1。
- 与HSV类似的颜色空间还有：
 - HSL (L: Lightness)
 - HIS (I: Intensity)



其他颜色空间

- YUV
 - YUV颜色空间由亮度信号Y和两个色差信号R - Y、B - Y组成。YUV色彩空间的重要性是它的亮度信号Y和色度信号U、V是分离的。如果只有Y信号分量就可能表示黑白灰度图。YUV颜色空间主要用于图像压缩及传输。
- CMYK
 - CMYK颜色空间应用于印刷工业，印刷业通过青（C）、品（M）、黄（Y）、黑（K）四色油墨的不同网点面积率的叠印来表现丰富多彩的颜色和阶调。
- Lab
 - Lab的色彩空间要比RGB模式和CMYK模式的色彩空间大，自然界中任何颜色都可以在Lab空间中表达出来。

- YUV：人眼对亮点信息更敏感。也就是说，可以压缩UV数据，而人眼难以发现。所以压缩算法的第一步，往往先把RGB数据转换成YUV数据。对Y少压缩一点，对UV多压缩一点，以平衡图像效果和压缩率。

颜色空间转换

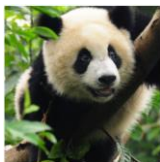
- 在计算机视觉中，尤其是颜色识别相关的算法设计中，各种颜色空间经常混合使用。RGB，HSV，YUV等常见颜色空间可以通过计算公式实现相互转化，这个过程叫做颜色空间转换。颜色变换的计算公式比较复杂，通常图像处理库会提供颜色空间转换的API给用户调用。

$$\begin{aligned} h_i &\equiv \left\lfloor \frac{h}{60} \right\rfloor \pmod{60} \\ f &= \frac{h}{60} - h_i \\ p &= v \times (1 - s) \\ q &= v \times (1 - f \times s) \\ t &= v \times (1 - (1 - f) \times s) \end{aligned} \quad (r, g, b) = \begin{cases} (v, t, p), & \text{if } h_i = 0 \\ (q, v, p), & \text{if } h_i = 1 \\ (p, v, t), & \text{if } h_i = 2 \\ (p, q, v), & \text{if } h_i = 3 \\ (t, p, v), & \text{if } h_i = 4 \\ (v, p, q), & \text{if } h_i = 5 \end{cases}$$

- cv2.cvtColor

灰度化

- 为了方便计算，在不关心颜色的处理场景中，经常将多通道的彩色图像转为单通道的灰度图像，这个过程叫做灰度化。对于BGR图像，常见的灰度化思路有三种：
 - 最大值法： $I = \max(R, G, B)$
 - 平均值法： $I = (R + G + B)/3$
 - 加权平均值法： $I = 0.11 * B + 0.59 * G + 0.30 * R$
- 灰度化的操作也可以直接调用图像处理库的API完成。



- 加权平均值的原理：由于人眼对绿色最为敏感，红色次之，对蓝色的敏感性最低。

像素位置关系

- 数字图像在计算机里的表示是一个矩阵。

$$I = \begin{bmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,M-1} \\ p_{1,0} & p_{1,1} & & p_{1,M-1} \\ \vdots & & \ddots & \vdots \\ p_{N-1,0} & p_{N-1,1} & \cdots & p_{N-1,M-1} \end{bmatrix}$$

- 像素可以由它的坐标 (x, y) 描述。根据坐标，像素之间具备一些空间位置关系，比如像素的邻接关系：
 - 对于坐标 (x, y) 的像素P，P有四个水平垂直的相邻像素，称为4-邻域： $(x-1, y)$ ， $(x+1, y)$ ， $(x, y-1)$ ， $(x, y+1)$ 。
 - P有四个对角相邻像素，称为对角邻域： $(x-1, y-1)$ ， $(x-1, y+1)$ ， $(x+1, y-1)$ ， $(x+1, y+1)$ 。
 - 4-邻域和对角邻域合称为像素P的8-邻域。

像素距离度量

- 像素之间的联系与像素在空间的接近程度相关，这个接近程度可用距离来度量。在图像处理中常用的距离函数有：

- 欧式距离：

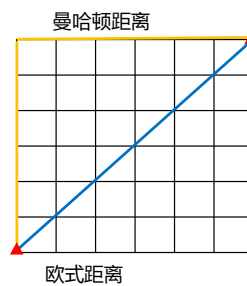
- $De(p, q) = [(x - s)^2 + (y - t)^2]^{\frac{1}{2}}$

- 曼哈顿距离 ($D4$) :

- $D4(p, q) = |x - s| + |y - t|$

- 棋盘距离 ($D8$) :

- $D8(p, q) = \max(|x - s|, |y - t|)$



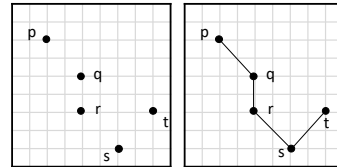
连接和连通

- 如果两个像素不仅空间位置上邻接，并且其像素值也符合相似准则，我们称这两个像素是连接的。
- 两种常用的连接：
 - 4-连接：像素 p , q 都在 V 中取值，且 q 在 p 的4-邻域中，称它们是4-连接。
 - 8-连接：像素 p , q 都在 V 中取值，且 q 在 p 的8-邻域中，称它们是8-连接。
- 像素连通是像素连接的一种推广。如果像素 p 和像素 t 可以通过两两连接的像素进行关联，则 p 和 t 是连通的。所有关联的像素称为 p 、 t 之间的通路。
- 例如， p 和 q 是连接的， q 和 r 是连接的， r 和 s 是连接的， s 和 t 是连接的，则 p 和 t 是连通的， $pqrst$ 是一条通路。

0	1	0
0	1	0
0	0	1

4-连接（绿色与黄色像素点）

8-连接（绿色与深蓝色像素点）



- 像素值的相似准则指像素的灰度值相等，或者像素值都在一个灰度集合 V 中。
- 按照连接方式，连通分为4-连通和8-连通。上图中 $pqrst$ 是8-连通，但不是4-连通。

连通域

- 在一副图像中，一个像素集合内部两两连通，并和集合外部的像素都不连通，这样的像素集合被称为连通域。
- 根据连接的不同定义，连通域同样分为4-连通域，8-连通域。
- 连通域的边界称为区域的边界或轮廓。



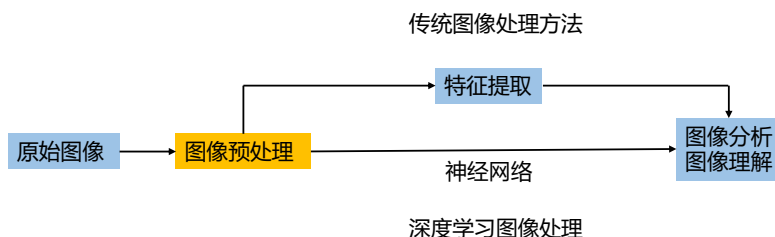
- 左图展示了两个连通域，右图红色的部分表示这两个连通域的边界。

目录

1. 计算机视觉概览
2. 数字图像处理基础
- 3. 图像预处理技术**
4. 图像处理基本任务
5. 特征提取和传统图像处理算法
6. 卷积神经网络和图像识别
7. 生成对抗网络和图像生成

图像处理的一般流程

- 图像处理的一般流程：



- 传统图像处理算法中图像预处理的目的：是消除图像中无关的信息，恢复有用的真实信息，增强有关信息的可检测性、最大限度地简化数据，从而改进特征提取、图像分割、匹配和识别的可靠性。
- 图像预处理包括：灰度变换，几何矫正，图像增强，图像滤波等操作。

- 图像预处理是一种输入图像，输出也是图像的处理任务。
- 无论是传统图像处理方法还是深度学习图像处理方法均需要进行图像预处理。传统图像处理算法中图像预处理的目的：是消除图像中无关的信息，恢复有用的真实信息，增强有关信息的可检测性、最大限度地简化数据，从而改进特征提取、图像分割、匹配和识别的可靠性。
- 深度学习中是不需要进行特征提取的，为什么需要做图像预处理呢？在此小节最后做出解答。

灰度变换

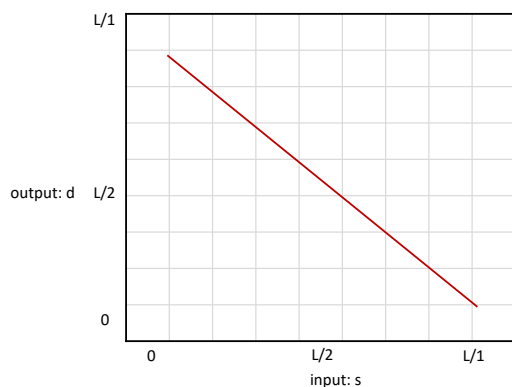
- 图像灰度变换是将原始图像中每一个点的灰度值，按照特定映射函数，变换为另一个灰度值。灰度变换是对每一个像素点单独的操作，我们记录像素点的原始值为 s ，灰度变换的映射函数为 $T(s)$ ，变换后像素点的值为 d ，即：

$$d = T(s)$$

- 选择不同的灰度变换函数，可以实现增强图像视觉效果的目的。对于灰度级为 L 的图像，映射函数是一个将 $[0, L - 1]$ 的输入值映射到 $[0, L - 1]$ 的输出值的函数。可以将映射函数的图形绘制出来，表示灰度变换的效果。
- 灰度变换的目的是为了改善画质，使图像的显示效果更加清晰。主要包括：
 - 对比度展宽
 - 动态范围调整
 - 直方图均衡化等

反转

- 图像反转是将原图灰度值反转的操作，简单说就是将黑色变为白色，白色变为黑色。映射函数的图示如下：



图像对比度

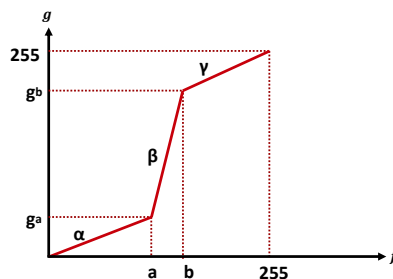
- 对比度指相邻像素之间颜色的差别，对比度通常表现了图像画质的清晰程度。对比度越大，不同颜色之间的反差越大，即所谓黑白分明，对比度过大，图像就会显得很刺眼。对比度越小，不同颜色之间的反差就越小。
- 对比度计算公式：

$$C = \sum_{\delta} \delta(i, j)^2 P_{\delta}(i, j)$$

- 其中 $\delta(i, j)$ 为相邻像素之间的灰度差， $P_{\delta}(i, j)$ 为相邻像素间灰度差为 δ 的像素分布概率。

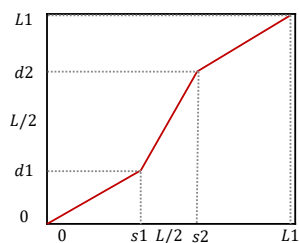
对比度展宽

- 在曝光不足或过度时，图像灰度会局限在一个很小的范围内。此时图像会模糊不清、没有灰度层次。采用线性对比度展宽法对图像每一个像素灰度做线性拉伸，可有效地改善这类图像的视觉效果。
- 对比度展宽：是对图像进行点对点的映射。在图像像素点动态范围不变的情况下，扩展重要部分，抑制不重要的部分。



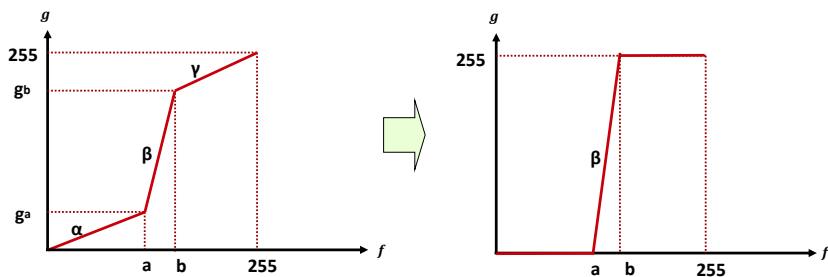
分段线性对比度展宽

- 图像对比度增强通过增加相邻灰度级的灰度差别实现。其映射函数如下图所示。当 $(d2 - d1) > (s2 - s1)$ 时，原始图像中灰度级处于 $[s1, s2]$ 区间内的像素点被拉伸到 $[d1, d2]$ ，达到对比度增强的效果。



对比度展宽：灰度窗

- 只显示指定灰度级范围内的信息。
- 按照前面的计算公式有： $\alpha = \gamma = 0$, $g_b = 255$, $g_a = 0$ 。

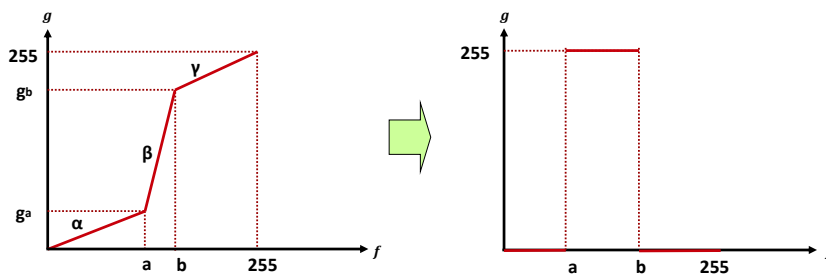


灰度窗效果示意图

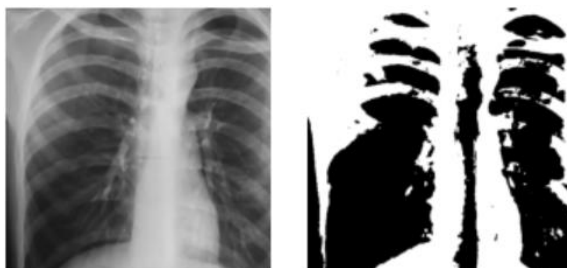


对比度展宽：灰度窗切片（1）

- 只保留感兴趣的部分，使其为二值化图像中的白色部分，其余部分置为0。

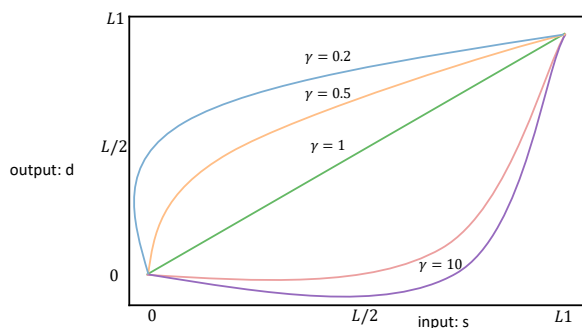


对比度展宽：灰度窗切片（2）



伽马矫正 (1)

- 伽马矫正是一种借助了指数变换映射的增强技术，属于非线性对比度展宽。
- 伽马矫正的映射函数可表示为： $d = T(s) = cs^\gamma$ ，其中 c 和 γ 为常数。
- γ 的值是控制变换效果的主要因素，所以被称之为伽马矫正。



- 当 $\gamma > 1$ 时，
 - 输入的低灰度范围被压缩，高灰度范围被拉伸，实现了强化亮部，压缩暗部的图像增强效果。
- 当 $\gamma < 1$ 时，
 - 输入的高灰度范围被压缩，低灰度范围被拉伸，实现了强化暗部，压缩亮部的图像增强效果。

伽马矫正 (2)

- γ 的值距离1越远，伽马矫正的变换强度越大。



原图



$\gamma=0.5$



$\gamma=3$

动态范围调整

- 动态范围：是指图像场景中的光亮度的范围。
- 动态范围对人视觉的影响：由于人眼所可以分辨的灰度的变化范围是有限的，所以当动态范围太大时，很高的亮度值把暗区的信号都掩盖了。
- 动态范围调整原理：将动态范围进行压缩，将感兴趣的区域变化范围变大，从而达到改善图像质量的目的。

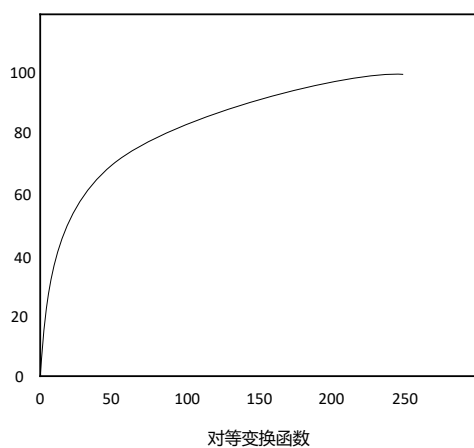
对数动态范围调整 (1)

- 通常动态范围调整用取对数的方法。原因是人眼对信号的处理是有一个近似对数算子的环节。

$$g(i,j) = c \log_{10}(1 + f(i,j))$$

- 其中 $c = \frac{\text{转换后灰度级}}{\log_{10}(1 + \text{原始灰度级})}$ ，若需将[0,255]的灰度压缩到[0,100]， $c = \frac{100}{\log_{10}(1 + 255)}$ 。

对数动态范围调整 (2)

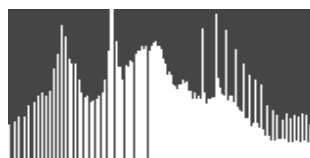


直方图

- 灰度直方图反映的是一幅图像中各灰度级像素出现的频率。以灰度级为横坐标，纵坐标为灰度级出现的频率，绘制频率同灰度级的关系图就是灰度直方图。它是图像的一个重要特征，反映了图像灰度分布的情况。
- 基于直方图进行图像变换是一种基于概率论的处理方法，通过改变图像的直方图，修改图像中各像素的灰度值，达到增强图像视觉效果的目的。
- 相对于灰度变换只针对单独的像素点操作，直方图变换综合考虑了全图的灰度值分布。

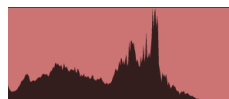
灰度图像的直方图

- 下面是两幅灰度图像的直方图，直方图的形状能反映图像的视觉效果。



彩色图像各通道的直方图

- 对于彩色图像，可以对不同的通道分别统计直方图。



红色



绿色



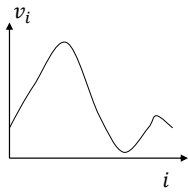
蓝色

直方图的计算

- 直方图横坐标为 i ，表示有 i 个灰度级。像素的总数为 n ，第 i 个灰度级有 n_i 个像素，纵坐标 v_i 表示直方图第 i 列的高度 v_i ， $v_i = n_i$ ，第 i 个灰度级的像素个数。
- 对于归一化的直方图， $v_i = \frac{n_i}{n}$ ，即 i 灰度级的出现频率。

1	2	3	1	2	3	1	0
7	6	5	2	6	7	5	0
2	1	5	3	6	0	6	1
0	5	6	3	5	7	6	2
6	1	4	2	7	2	2	3
0	6	7	2	6	5	2	2
2	1	2	1	2	3	2	1
1	2	2	1	3	2	1	3

$v_0 = 5/64$
 $v_1 = 12/64$
 $v_2 = 18/64$
 $v_3 = 8/64$
 $v_4 = 1/64$
 $v_5 = 5/64$
 $v_6 = 8/64$
 $v_7 = 5/64$



直方图均衡化

- 直方图均衡化是调整原始图像的直方图，即灰度概率分布图，使之呈均衡分布的样式，以达到灰度级均衡的效果，可以有效增强图像的整体对比度。
- 直方图均衡化能够自动地计算变换函数，并产生有均衡直方图的输出图像。能够有效增强改善图像过暗、过亮和细节不清晰的问题。
- 基本思想：对在图像中像素个数多的灰度级进行展宽，而对像素个数少的灰度级进行缩减。从而达到充分利用所有灰度级，清晰图像的目的。
- 在常用的图像处理库中，直方图操作都有API，直接调用即可实现。

直方图均衡化实例（1）

- 步骤1：求出图像的总像素个数。共 $m \times n$ ，25个。
- 步骤2：计算每个灰度级的像素个数在整个图像中所占的百分比。

1	3	9	9	8
2	1	3	7	3
3	6	0	6	4
6	8	2	0	5
2	9	2	6	0

直方图操作



$h_s(0)$	3/25
$h_s(1)$	2/25
$h_s(2)$	4/25
$h_s(3)$	4/25
$h_s(4)$	1/25
$h_s(5)$	1/25
$h_s(6)$	4/25
$h_s(7)$	1/25
$h_s(8)$	2/25
$h_s(9)$	3/25

直方图均衡化实例（2）

- 步骤3：计算图像各灰度级的累计分布。

$$h_p(i) = \sum_{k=0}^i h_s(k)$$

$h_s(0)$	0.12
$h_s(1)$	0.08
$h_s(2)$	0.16
$h_s(3)$	0.16
$h_s(4)$	0.04
$h_s(5)$	0.04
$h_s(6)$	0.16
$h_s(7)$	0.04
$h_s(8)$	0.08
$h_s(9)$	0.12



$h_p(0)$	0.12
$h_p(1)$	0.20
$h_p(2)$	0.36
$h_p(3)$	0.52
$h_p(4)$	0.56
$h_p(5)$	0.60
$h_p(6)$	0.76
$h_p(7)$	0.80
$h_p(8)$	0.88
$h_p(9)$	1.00

直方图均衡化实例 (3)

- 步骤4: 计算新图像的灰度值。

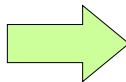
0	0.12
1	0.20
2	0.36
3	0.52
4	0.56
5	0.60
6	0.76
7	0.80
8	0.88
9	1.00

$$g(i,j) = L * h_p(k), \text{ 其中 } L \text{ 为灰度级数}$$

1	3	9	9	8
2	1	3	7	3
3	6	0	6	4
6	8	2	0	5
2	9	2	6	0

输入图像

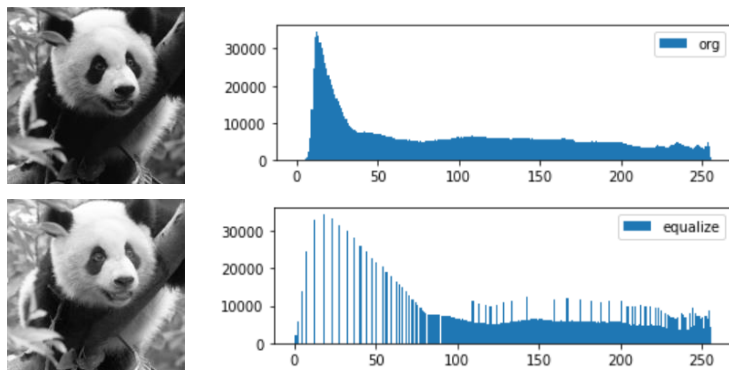
$$9 * 0.20 = 1.8 \approx 2$$



2	5	9	9	8
3	2	5	7	5
5	7	0	7	5
7	8	3	0	5
3	9	3	7	0

输出图像

直方图均衡化效果

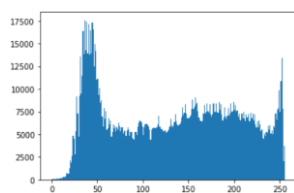
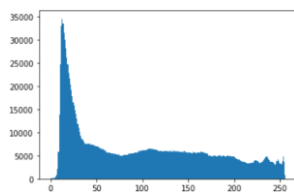


- 上述的HE算法得到的结果存在一些问题，主要包括：1) 对于灰度非常集中的区域，直方图会被拉的非常稀疏，从而导致对比度增强过大，成为噪音；2) 一些区域调整后变得更暗/更亮，丢失细节信息。
- HE：全局均衡→CLHE：限制对比度→AHE：局部均衡但过渡不平滑→CLAHE：削弱局部噪点放大问题，插值平滑过渡。
- <https://blog.csdn.net/wyttRain/article/details/116108925>
- <https://blog.csdn.net/lwx309025167/article/details/103770834>

直方图规定化 (1)

- 直方图均衡化总是得到全局均衡化的输出图像，但实际应用中经常需要对某些特定的灰度区间进行增强，如提升暗部或增强亮部。使用直方图规定化的方法可以按需要灵活地对直方图的分布进行调整。这种定制化调整是直方图规定化的特点。
- 直方图规定化的方法也是调整原始图像的直方图，使得其概率分布逼近目标直方图的概率分布。
- 使用直方图规定化的方法，需先根据经验知识，计算最合适的目标直方图，与原始图像一起作为输入，输出为规定化之后的图像，从而达到特定的图像增强效果。

直方图规定化 (2)



图像滤波

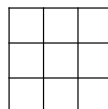
- 图像滤波，它是一种局部处理方式，像素 P 的输出值由输入图像中像素 P 及其邻域 N 中的像素值确定。图像滤波可以通过以下公式进行计算：

$$O(i, j) = \sum_{m, n} I(i + m, j + n) \cdot K(m, n)$$

- 计算方法是将邻域 N 的像素值和一个与邻域同样尺寸的子图像进行模板运算。子图像被称为**模板、核或滤波器**。
- 常用的模板运算有**模板卷积**和**模板排序**。

灰度变换是点处理，它的输出值和指定像素点相关。

直方图变换是全局处理，它的输出值和整幅图像相关。



滤波（子图）

- 图像处理中滤波将信号中特定的波段频率滤除，从而保留所需要的波段频率信号。根据选择保留的不同频段可以体现两个作用：
 - 消除图像中混入的噪声：对应的是低通滤波，噪声在图像中一般是高频信号。
 - 为图像识别抽取图像特征：这里的特征一般为边缘纹理的特征，对应的是高通滤波，图像中边缘和纹理细节是高频信号。

图像滤波器的分类

- 平滑空间滤波器：去除图像中不重要的细节，减少噪音。
 - 均值滤波器
 - 中值滤波器
 - 最大最小值滤波器
- 锐化空间滤波器：突出图像中的细节，增强被模糊的细节。
 - 一阶微分滤波器：梯度算子
 - 二阶微分滤波器：拉普拉斯算子

- 平滑滤波器主要是使用邻域的均值（或者中值）来代替模板中心的像素，消弱和邻域间的差别，以达到平滑图像和抑制噪声的目的；相反，锐化滤波器则使用邻域的微分作为算子，增大邻域间像素的差值，使图像的突变部分变的更加明显。

均值滤波

- 均值滤波指模板权重都为1的滤波器。它将像素的邻域平均值作为输出结果，均值滤波可以起到图像平滑的效果，可以去除噪声，但随着模板尺寸的增加图像会变得更为模糊。经常用于图像模糊化。

1	1	1
1	1	1
1	1	1



原图



3x3均值滤波



5x5均值滤波

- 原图比较大，3x3均值可能不太明显。

中值滤波

- 中值滤波属于模板排序运算的滤波器。中值滤波器将邻域内像素排序后的中位数值输出以代替原像素值。它不仅能实现降噪，而且保留了原始图像的锐度，不会修改原始图像的灰度值。
- 中值滤波的使用非常普遍，它对椒盐噪声 (salt & pepper noise) 的抑制效果很好，在抑制随机噪声的同时能有效保护图像边缘信息。但中值滤波是一种非线性变化，它可能会破坏像素点的线性关系，不适用于点、线等细节较多的图像或高精度的图像。



原图



3x3均值滤波



3x3中值滤波

- 椒盐噪声是由图像传感器，传输信道，解码处理等产生的黑白相间的亮暗点噪声。椒盐噪声是指两种噪声，一种是盐噪声 (salt noise)，另一种是胡椒噪声 (pepper noise)。盐=白色，椒=黑色。前者是高灰度噪声，后者属于低灰度噪声。一般两种噪声同时出现，呈现在图像上就是黑白杂点。

最大最小值滤波

- 最大最小值滤波是一种比较保守的图像处理手段，与中值滤波类似，首先要排序周围像素和中心像素值，然后将中心像素值与最小和最大像素值比较，如果比最小值小，则替换中心像素为最小值，如果中心像素比最大值大，则替换中心像素为最大值。

123	124	108
122	150	98
112	135	144

排序以后为：98、108、112、122、123、124、135、144

中心像素为：150

最大最小值滤波后，中心像素值为：144

图像微分的性质（1）

- 一阶微分：
 - 在恒定的灰度区域，图像的微分值为0。
 - 在灰度台阶或斜坡起点处微分值不为0。
 - 沿着斜坡的微分值不为0。
- 二阶微分：
 - 在恒定区域二阶微分值为0。
 - 在灰度台阶或斜坡的起点处微分值不为0。
 - 沿着斜坡的微分值为0。

- 灰度值没有发生变换，自然微分为0。
- 台阶是，灰度值的突变变化较大；斜坡则是灰度值变化较缓慢；灰度值发生了变化，微分值不为0。
- 从以上图像灰度的一阶和二阶微分的性质可以看出，在灰度值变化的地方，一阶微分和二阶微分的值都不为0；在灰度恒定的地方，微分值都为0。也就是说，不论是使用一阶微分还是二阶微分都可以得到图像灰度的变化值。

图像微分的性质 (2)

- 对于图像的边缘来说，通常会形成一个斜坡过度。
- 一阶微分在斜坡处的值不为0，那么用其得到的边缘较粗；而二阶微分在斜坡处的值为0，但在斜坡两端值不为0，且值的符号不一样，这样二阶微分得到的是一个由0分开的一个像素宽的双边缘。
- 也就是说，二阶微分在增强图像细节方面比一阶微分好得多，并且在计算上也要比一阶微分方便。

微分算子

- 一阶梯度：Prewitt梯度算子、Sobel梯度算子。
- 二阶梯度：Laplacian梯度算子。

fx'			fy'		
-1	0	1	-1	-1	-1
-1	0	1	0	0	0
-1	0	1	1	1	1

Prewitt算子

fx'			fy'		
-1	0	1	-1	-2	-1
-2	0	2	0	0	0
-1	0	1	1	2	1

Sobel算子

四领域			八领域		
0	1	0	1	1	1
1	-4	1	1	-8	1
0	1	0	1	1	1

Laplacian算子

- 梯度在图像处理领域我们可以也理解为像素灰度值**变化的速度**，在数字图像处理，因其像素性质的特殊性，微积分在图像处理表现的形式为计算当前像素点沿偏微分方向的差值，所以实际的应用是不需要用到求导的，**只需进行简单的加减运算**。
- 对于离散图像的梯度，一阶偏导数采用一阶差分近似表示，即
 - $fx' = f(x + 1, y) - f(x, y)$
 - $fy' = f(x, y + 1) - f(x, y)$
- 类似地，二阶差分：
 - $fx'' = f(x - 1, y) + f(x + 1, y) - 2f(x, y)$
 - $fy'' = f(x, y - 1) + f(x, y + 1) - 2f(x, y)$
- Sobel算子在Prewitt算子的基础上增加了权重的概念，认为相邻点的距离远近对当前像素点的影响是不同的，距离越近的像素点对应当前像素的影响越大，从而实现图像锐化并突出边缘轮廓。
- Laplacian算子：第一个kernel：四领域。第二个kernel：八领域。

图像滤波的应用 - 图像锐化

- 锐化处理的主要目的是突出图像中的边缘、轮廓线和图像细节。
- 将求取的边缘按照一定系数比例叠加到原始图像上，即可实现对图像的锐化操作。例如使用Laplacian梯度算子作为锐化操作的模板，其中A是大于等于1的系数。

0	-1	0
-1	A+4	-1
0	-1	0



原图

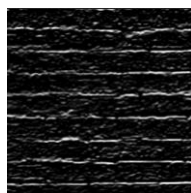


锐化后

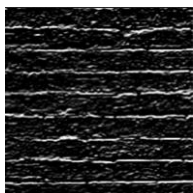
图像滤波的应用 - 边缘检测



原图



Prewitt算子提取的边缘



Sobel算子提取的边缘



Laplacian算子提取的边缘

- **噪声污染没有被排除、边缘线太过于粗宽。**
- LOG：高斯滤波+Laplacian梯度算子。
- Canny：边缘瘦身， <https://zhuanlan.zhihu.com/p/99959996>

Canny边缘检测算法

- Canny边缘检测算法可以分为以下5个步骤：
 - 使用高斯滤波器，以平滑图像，滤除噪声。
 - 计算图像中每个像素点的梯度强度和方向。
 - 应用非极大值（Non-Maximum Suppression）抑制，以消除边缘检测带来的杂散响应。
 - 应用双阈值（Double-Threshold）检测来确定真实的和潜在的边缘。
 - 通过抑制孤立的弱边缘最终完成边缘检测。

- 参考：<https://cloud.tencent.com/developer/article/2088041>

图像的几何变换

- 图像的几何变换是指用数学建模的方法来描述图像位置、大小和形状等变化的方法。
- 图像的几何变换不改变像素的值，只改变像素的位置。
- 图像的几何变换可分为：
 - 图像的仿射变换：使用数学映射变换公式，表示几何变换。
 - 图像的形状变换：图像的放大、缩小和剪裁。
 - 图像的位置变换：图像的平移、镜像和旋转。

仿射变换（1）

- 仿射变换是在二维平面对图像进行线性坐标变换的方法，即对图像进行线性变换和平移，包括了图像平移、缩放、旋转、镜像等。仿射变换保留了图像中的“平行性”和“平直性”，即仿射变换后直线和平行线仍然保持是直线和平行线。
- 为了将几何变换表示为坐标向量和变换矩阵相乘的形式，特别是为了适应平移，提出了齐次坐标的概念。
- 实质是通过增加一个坐标量来解决问题，基于齐次坐标，可以定义仿射变换如下：

$$\begin{aligned}x' &= ax + by + \Delta x \\y' &= cx + dy + \Delta y\end{aligned}$$

- 在笛卡尔坐标系内，向量 (x,y) 是位于 $z=0$ 平面上的点，而向量 $(x,y,1)$ 是位于 $z=1$ 的等高平面上的点。
- 对于图形来说，没有实质性的差别，但是却给后面的矩阵运算提供了可行性和方便性。

仿射变换 (2)

- 仿射变换的变换公式为：

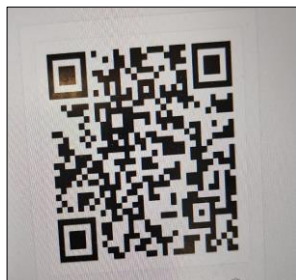
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & \forall x \\ c & d & \forall y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- 仿射变换 T 矩阵有6个参数，所以知道3个像素点变换前后的对应坐标就可以求出 T 矩阵。
- 一个像素点 p 的坐标为 (x, y) ，用齐次坐标的向量形式表示为 $p = [x \ y \ 1]^T$ ， p 经过变换后的坐标为 (x', y') ，表示为 $p' = [x' \ y' \ 1]^T$ 。
- 坐标变换可以表示为 $p' = Ap$ ，其中 A 是一个 3×3 的变换矩阵。

- 其实仿射变换和透视变换更直观的叫法可以叫做「平面变换」和「空间变换」或者「二维坐标变换」和「三维坐标变换」。
- 仿射变换的方程组有6个未知数，所以要求解就需要找到3组映射点，三个点刚好确定一个平面。透视变换的方程组有8个未知数，所以要求解就需要找到4组映射点，四个点就刚好确定了一个三维空间。
- <https://cloud.tencent.com/developer/article/1638969>

仿射变换 (3)

- 二维码仿射变换示例：

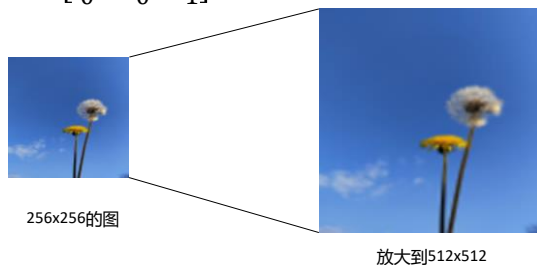


缩放

- 将分辨率为 (w, h) 的图像，缩放其图像尺寸为 (w', h') 。则水平方向的缩放系数为 $S_x = w'/w$ ，垂直方向的缩放系数为 $S_y = h'/h$ 。

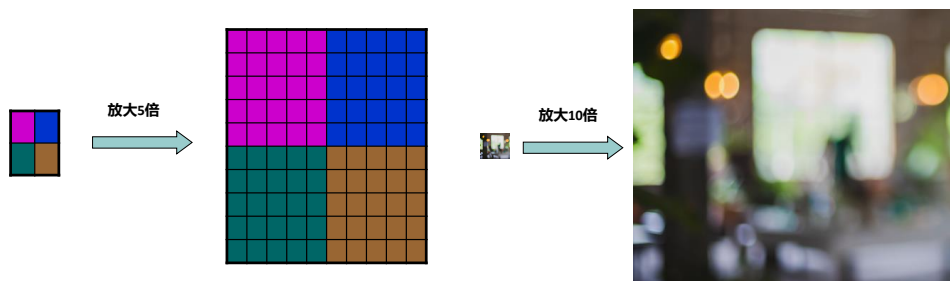
- 缩放变换的变换矩阵为：
$$A = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- 示例：



按比例放大图像

- 如果放大倍数太大，按照比例放大图像会出现马赛克效应。



插值

- 实际数字图像中的 (x, y) 的值都是整数，但是坐标变换后得到的坐标 (x', y') 不一定是整数。非整数坐标处的像素值需要用周围的整数坐标的像素值来进行计算，这个过程叫做插值。
- 插值的计算方法有很多种，最简单的插值方法是最邻近插值法。最邻近插值法直接用点 (x', y') 最近的整数坐标处的灰度值作为点 (x', y') 的值。
- 最邻近插值法计算量小，但是精确度不高，并且可能破坏图像中的线性关系。为了提高精度，可使用双线性插值法。

双线性插值

- 双线性插值法使用点 (x', y') 最近邻的四个像素值进行插值计算，假设使用点邻近的四个像素点为 (i, j) , $(i + 1, j)$, $(i, j + 1)$, $(i + 1, j + 1)$ 。

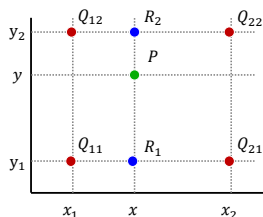
- 在 x 方向上进行线性插值计算：

$$f(R_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \quad \text{Where } R_1 = (x, y_1)$$

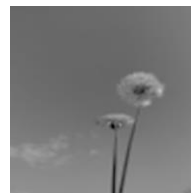
$$f(R_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \quad \text{Where } R_2 = (x, y_2)$$

- 在 y 方向上进行线性插值计算：

$$f(P) \approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2)$$



使用最邻近插值的缩放蒲公英图



使用双线性插值的缩放蒲公英图

平移

- 将图像中坐标为 $p = (x, y)$ 的点，用平移量 (T_x, T_y) 移动到新的位置，变换后的坐标为 $p' = (x', y')$ 。

平移变换的变换矩阵为：
$$A = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix}$$



镜像 (1)

- 将大小为 (w, h) 的图像中坐标为 $p = (x, y)$ 的点，按图像的中轴线对称，变换后的坐标为 $p' = (x', y')$ 。

- 水平镜像变换的变换矩阵为： $A = \begin{bmatrix} -1 & 0 & w-1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ ， w 为图像宽度。

- 垂直镜像变换的变换矩阵为： $A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & h-1 \\ 0 & 0 & 1 \end{bmatrix}$ ， h 为图像高度。

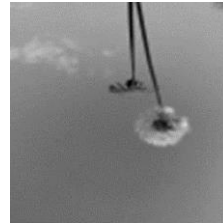
镜像 (2)



原图



水平镜像



垂直镜像

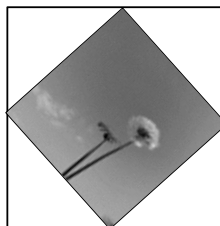
旋转 (1)

- 图像的旋转可以看作是围绕一条垂直于图像屏幕的轴线将像素点进行旋转。
- 将图像中坐标为 $p = (x, y)$ 的点，围绕处于坐标原点的轴顺时针旋转 θ ，变换后的坐标为 $p' = (x', y')$ 。

- 旋转变换的变换矩阵为：
$$A = \begin{bmatrix} \cos\theta & \sin\theta & T_x \\ -\sin\theta & \cos\theta & T_y \\ 0 & 0 & 1 \end{bmatrix}。$$

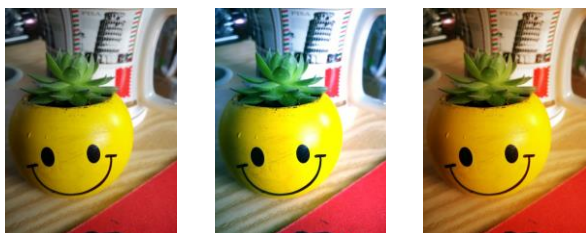
旋转 (2)

- 如果旋转轴不处于原点，变换矩阵的计算将较为复杂，可借助平移变化来简化操作：1) 通过平移变化将旋转轴移动到原点，2) 进行旋转轴为原点的旋转变换，3) 再通过平移变化将旋转轴恢复到原始位置。
- 示例：



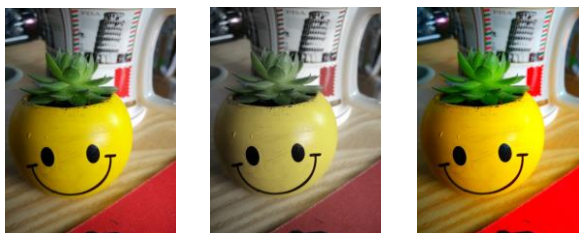
彩色图像处理 - 色调增强

- 由于人眼对HSV空间的三个分量的感受是独立的，可以对HSV单独的分量图进行处理达到图像增强的目的。
- 对HSV空间的H分量进行处理可以实现对图像色调的增强。
 - 色相的调整通常在H原始值上加上一个小的偏移量，使其在色相环上有小角度的调整。调整后，图像的色调会变为冷色或者暖色。
 - 色相H的值对应的是一个角度，并且在色相环上循环。所以色相的修正可能会造成颜色的失真。



彩色图像处理 - 饱和度增强

- 对HSV空间的S分量进行处理可以实现图像饱和度的增强。饱和度的调整通常是在S原始值上乘以一个修正系数。
 - 修正系数大于1，会增加饱和度，使图像的色彩更鲜明。
 - 修正系数小于1，会减小饱和度，使图像看起来比较平淡。



彩色图像处理 - 亮度增强

- 对HSV空间的V分量进行处理可以实现图像亮度的增强。
- 直接将彩色图像灰度化，也可以得到代表图像亮度的灰度图，计算量比HSV颜色空间变换低，但在HSV空间中进行亮度增强处理可以得到增强后的彩色图像。



图像预处理技术在深度学习中的应用

对于深度学习的图像处理算法，在处理过程中通常不需做预处理。但是图像预处理技术被用于训练数据集的数据增强（data augmentation），又称数据扩充。

在数据中加入随机噪声，扩大数据集样本量，提升数据样本空间的复杂度，主要应用在图像处理任务里。

图像数据增强常用方法：缩放，拉伸，加入噪点，翻转，旋转，平移，剪切，对比度调整和通道变换。

图像数据增强



目录

1. 计算机视觉概览
2. 数字图像处理基础
3. 图像预处理技术
- 4. 图像处理基本任务**
5. 特征提取和传统图像处理算法
6. 卷积神经网络和图像识别
7. 生成对抗网络和图像生成

数字图像处理的层次

- **低级图像处理内容（图像处理）**

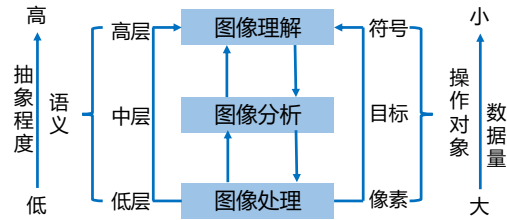
- 内容：主要对图像进行各种加工以改善图像的视觉效果、或突出有用信息，并为自动识别打基础，或通过编码以减少对其所需存储空间、传输时间或传输带宽的要求。
- 特点：输入是图像，输出也是图像，即图像之间进行变换。

- **中级图像处理（图像分析）**

- 内容：主要对图像中感兴趣的目标进行检测（或分割）和测量，以获得它们的客观信息，从而建立对图像中目标的描述，是一个从图像到数值或符号的过程。
- 特点：输入是图像，输出是图像特征。

- **高级图像处理（图像理解）**

- 内容：在中级图像处理的基础上，进一步研究图像中各目标的性质和它们之间相互的联系，并得出对图像内容含义的理解（对象识别）及对原来客观场景的解释（计算机视觉），从而指导和规划行动。
- 特点：输入是图像特征，输出是图像内容。



图像处理的三个层次

- 如果将计算机视觉看成一个单向的过程，可以分成这样三个层次：底层、中层、高层。
 - 底层一般是像素级处理，比如亮度增强、直方图、直方图均化等，我们将其称为图像处理。
 - 中层即从中分析一些边缘、朝向、纹理等这些特征，可以对图片进行一个分割，我们称其为图像分析。
 - 高层需要我们从中得到一些语义信息，比如图片中中间是老虎，下边是沙滩，绿色的是草地，这可能是一个野外场景，我们称其为图像理解。

图像处理任务 - 图像分类

- 图像分类（Image Classification）是图像处理在人工智能领域的基本研究主题，是计算机视觉领域的核心问题之一。
- 图像分类需要根据图像处理技术，判断输入的图像属于固定K个类别中的哪一类。K的大小取决于数据集。



鸢尾花



玫瑰

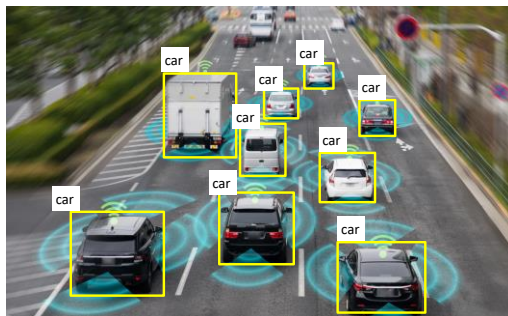


虞美人

- 输入输出：
 - 输入：图像。
 - 输出：类别（不同类别以概率值呈现）。

图像处理任务 - 目标检测

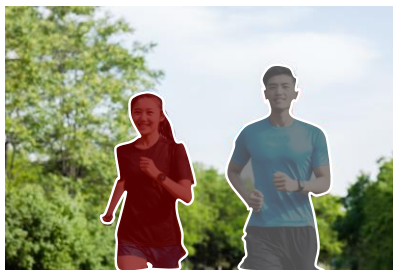
- 目标检测作为人工智能领域中图像处理和计算机视觉的经典技术，有着广泛的应用，例如交通监控、图像检索、人脸识别、人机交互等应用场景都使用了目标检测。目标检测是在图像中检测出人们感兴趣的目标，作为后续智能算法的处理对象。



- 由于各类物体有不同的外观、形状和姿态，加上成像时光照、遮挡等因素的干扰，目标检测一直是计算机视觉领域最具有挑战性的问题。
- 目标识别与目标检测略有不同，尽管它们使用类似的技术。给出一个特定对象，目标识别的目标是在图像中找出该对象的实例。这并不是分类，而是确定该对象是否出现在图像中，如果出现，则执行定位。例如，监控安防摄像头拍摄的实时图像以识别某个人的面部。
- 输入输出：
 - 输入：图像。
 - 输出：类别与边界框的位置。

图像处理任务 - 图像分割

- 图像分割（Segmentation）是根据需要解决的问题，将图像细分为目标内容所构成的不同子区域。
- 图像分割的算法和应用方向有很多，常见的图像分割方法有连通域分割，运动分割，目标分割等。



- 目标分割：语义分割，实例分割。
 - 语义分割（Semantic Segmentation）是指像素级的识别图像，即标注出图像中每个像素所属的对象类别。
 - 实例分割（Instance Segmentation）是指从图像中识别物体的各个实例，并逐个将实例进行**像素级**标注的任务。
- 分割是对图像的像素级描述，它赋予每个像素类别（实例）意义，适用于理解要求较高的场景，如无人驾驶中对道路和非道路的分割。
- 输入输出：
 - 输入：图像。
 - 输出：与输入图像同分辨率的带有各像素类别标签的分割图像。

图像处理任务 - 目标跟踪

- 目标跟踪（Object Tracking）是计算机视觉中的基础研究方向，有着广泛的应用，如：智能交通，安防监控，人机交互，无人驾驶等。
- 跟踪算法能够得到感兴趣的目标在时间上的运动轨迹，从而对目标的运动行为进行分析。



图像处理任务 - 文字识别

- 文字识别（Optical Character Recognition, OCR）是将图片或扫描件中的文字识别成可编辑的文本，其中使用了文字目标检测，目标分类等图像处理技术。可代替人工录入，提升业务效率。常见的文字识别应用有身份证、驾驶证、行驶证、发票、英文海关单据、通用表格、通用文字等场景文字识别。



图像处理任务 - 人脸识别

- 人脸识别，是基于人的脸部特征信息，利用计算机对人脸图像进行处理、分析和理解。人脸识别现在已经有很多应用方向，是身份验证的有效手段。
 - 人脸检测：人脸检测是在图像中准确识别出人脸的位置和大小。
 - 人脸比对：通过对人脸特征进行对比，计算出给两张图片中人脸的相似度。
 - 人脸搜索：查询人脸数据库，返回与输入人脸相似度最高的N张人脸图片。可用于企业、住宅的安全管理、公安刑侦等多种场景。



图像处理任务 - 内容检测

- 内容检测指利用计算机对图像进行处理、分析和理解，以识别各种不同模式的目标和对象的技术。
- 自然图像的语义内容非常丰富，一个图像包含多个标签内容，图像内容检测能够更智能、准确地理解图像内容，让智能相册管理、照片检索和分类、基于场景内容或者物体的广告推荐等功能更加精准。

小孩
草地
天空
树



玩
奔跑
开心
一群

图像处理任务 - 图像生成

- 图像生成 (Image Generation)：根据一张图片生成并修改部分区域的图片或者是全新的图片的任务。
 - 超分辨率 (Super Resolution)：从低分辨率对应物估计高分辨率图像的过程，以及不同放大倍数下图像特征的预测。
 - 风格迁移 (Style Migration)：将一个领域或者几张图片的风格应用到其他领域或者图片上。比如将抽象派的风格应用到写实派的图片上。
 - 图像修复 (Image Inpainting)：修复图像中缺失的地方，比如可以用于修复一些老的有损坏的黑白照片和影片。



图像修复

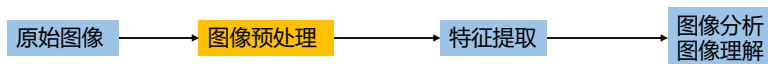
- 输入输出：
 - 输入：真实图像与随机噪声。
 - 输出：生成图像。

目录

1. 计算机视觉概览
2. 数字图像处理基础
3. 图像预处理技术
4. 图像处理基本任务
- 5. 特征提取和传统图像处理算法**
6. 卷积神经网络和图像识别
7. 生成对抗网络和图像生成

传统图像处理算法

- 深度学习技术的诞生在图像处理领域掀起了一场技术革命，相比之下，传统图像处理算法在效果上与深度学习还有不小差距。
- 但是深度学习依赖强大的算力和海量数据集，在这些条件无法满足时，传统图像处理算法仍然能很好地满足一些场景的应用需求。
- 很多深度学习的辅助程序也都由传统图像处理算法完成。学习传统图像处理算法还能加深图像技术的理解，为深度学习的优化和创新提供理论基础。



传统图像处理方法

图像特征提取

- 数字图像是非结构化数据，不方便使用模式识别的算法进行计算。如果将图像按照像素灰度值强行展开为二维的结构化数据，一副分辨率为 800×600 的彩色图像，数据长度就有 $800 \times 600 \times 3 = 1440000$ 维，对一百多万维的数据进行计算是非常复杂和低效率的。
- 为了将数字图像有效的转换为结构化数据，需要用到图像特征提取的技术降低数据的维度。其主要思路有：

降采样：通过缩放等降采样技术降低待分析图像数据的分辨率。

分割感兴趣区域（Region of Interest, ROI）：通过分割技术，提取图像中的感兴趣目标的区域，减少待处理的数据量。

特征描述子：使用特征描述子将图像转为固定长度的特征向量。

图像二值化

- 图像二值化 (image binarization) 是将图像上的像素点的灰度值全部设置为黑色 (0) 或白色 (255) , 也就是将整个图像分割成明显的黑白效果的过程。(本节都以8位图像为例)
- 其中最常用的是**阈值分割**, 将图像中灰度值大于阈值的像素点设置为白色 (或黑色) , 小于阈值的点设置为黑色 (或白色) , 阈值 (Threshold) 通常用T表示。

- 二值图像在数字图像处理中占有非常重要的地位, 可以方便得进行连通域提取、形态学处理、感兴趣区域分割等操作。
- 图像的二值化使图像中数据量大为减少, 并且能突显出目标的轮廓。二值化后的图像的集合性质只与像素值为0或255的点的位置有关, 不再涉及像素的多级值, 使处理变得简单, 而且计算过程中数据的处理量减小。

自适应阈值分割

- 由于阈值的选取直接影响了二值化分割的效果，如何选择合适的阈值是二值化算法的核心。通过算法自动计算出分割阈值的方法叫做自适应阈值分割。自适应阈值分割能够根据图像的灰度分布特征求取出分割视觉效果最佳的阈值。
- 常用的自适应阈值分割有**双峰法**与**最大类间方差法（OTSU）**。



双峰法效果



OTSU效果

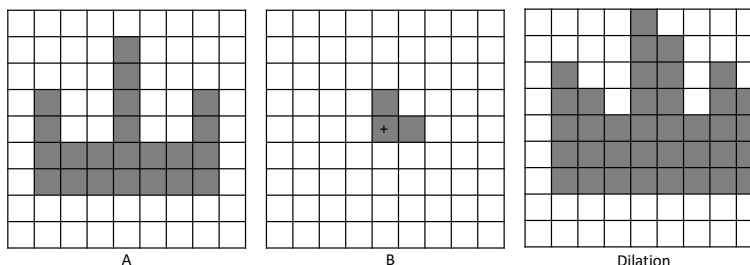
- 为了适应光照不均匀的图像，自适应阈值算法还衍生出局部自适应阈值分割，此类算法会对图像不同的区域计算适合该局部的阈值，以达到最佳分割效果。
- 根据实际要处理的问题，选择或设计合适的自适应阈值算法能够对分割效果和后续处理带了很大地提升。
- 双峰法
 - 直方图双峰法是一种易于理解的自适应阈值算法。如果图像灰度直方图呈明显的双峰状，则选取双峰之间的谷底所对应的灰度值作为图像分割的阈值。如果图像的灰度直方图很平坦，或者有多个波峰，无法用双峰法取得合适的阈值。
- OTSU
 - OTSU算法按图像的灰度特性，将图像分成前景和背景两部分。前景和背景之间的类间方差越大，说明构成图像的两部分的差别越大。当阈值不当分割错误时，会导致前景背景的类间方差变小。因此遍历所有可能的阈值，选择使类间方差最大的分割阈值作为最佳阈值。

形态学处理

- 图像的形态学处理（morphology）有简化图像数据，增强形状特征，去除干扰噪声等功能。形态学处理的基本操作是膨胀（dilation）和腐蚀（erosion）运算。由基本操作进行组合可实现开运算（opening）、闭运算（closing）等实用的算法。
- 形态学处理在噪声滤除、边界提取、目标定位和区域填充等任务中有着广泛的应用。
- 图像的形态学处理操作对象既可以是二值图像也可以是灰度图像，本节我们用二值图像来介绍形态学处理理论，同样的处理操作可以拓展到灰度图像。

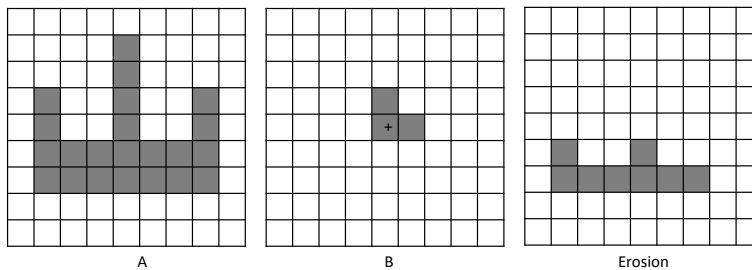
形态学基本操作 - 膨胀

- 将待处理的图像记为A，结构元素记为B，膨胀运算（Dilation）是根据B的形状，向外扩充A中的前景区域。
- 经过膨胀运算后，前景区域的边缘向背景区域膨胀扩增了。
- 数学表达式： $A \oplus B = \{x, y | (B)_{xy} \cap A \neq \emptyset\}$ 。



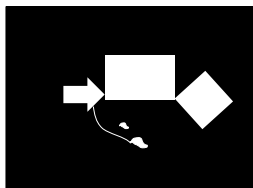
形态学基本操作 - 腐蚀

- 将待处理的图像记为A，结构元素记为B，腐蚀运算（Erosion）就是根据B的形状，向内收缩A中的前景区域。
- 经过腐蚀运算后，前景区域的边缘被背景区域压缩腐蚀了。
- 数学表达式： $A \ominus B = \{x, y | (B)_{x,y} \subseteq A\}$ 。

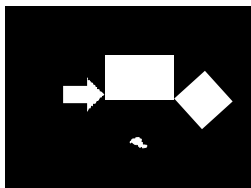


开运算

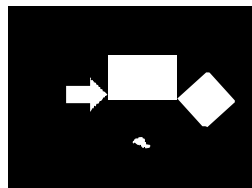
- 开运算就是用结构元素B对图像A腐蚀，再用B对结果进行膨胀得到最终结果。
- 开运算能够滤除比结构元素小的亮点噪声和突刺，并分离细长的连接。例如将因噪声影响粘连在一起的区域分割开。



原图



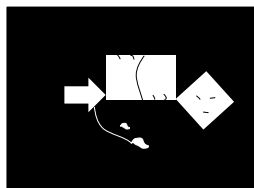
腐蚀



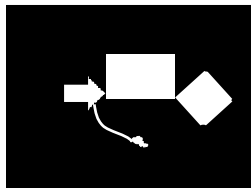
再膨胀

闭运算

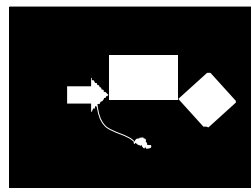
- 闭运算就是用结构元素B对图像A膨胀，再用B对结果进行腐蚀得到最终结果。
- 闭运算可以把填补比结构元素小的暗点缺憾或者空洞，连通细小断裂的区域并平滑边界。例如将因图像质量而断裂的目标区域进行合并连通。



原图



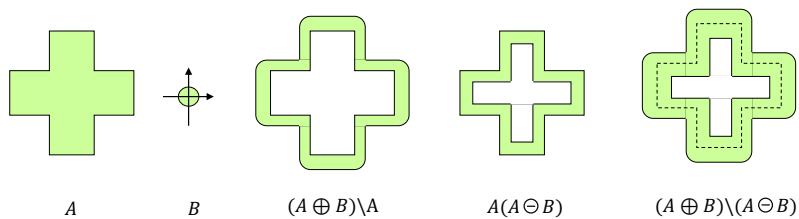
腐蚀



再膨胀

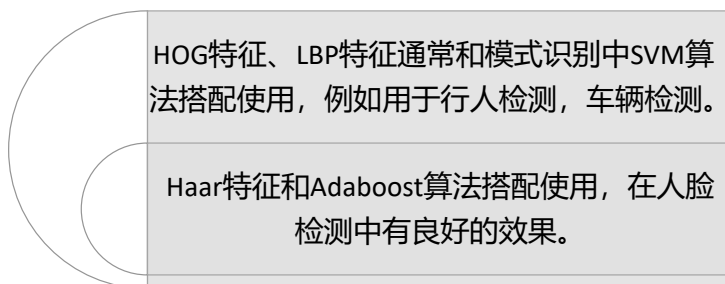
形态学的应用：边界检测

- 对于图像A和结构元素B来说：
 - 外边界求解：图像膨胀后的图像减去原来的图像得到的差值图像，称为图像的外部梯度。
 - 内边界求解：用原图像减去腐蚀后的图像得到差值图像，称为图像的内部梯度。
 - 形态学梯度求解：膨胀图与腐蚀图之差。



特征描述子

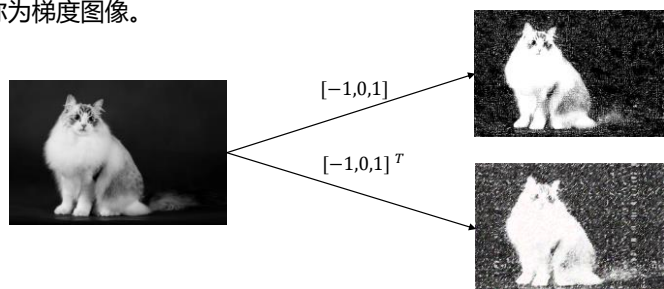
- 图像处理中的特征描述子用于将图像转为固定长度的特征向量。不同的特征描述子被设计出来以描述图像不同属性的特征。图像识别中常用的特征描述子有HOG特征、LBP特征和Haar特征等。



- 考虑：计算速度，旋转不变性，尺度不变性，亮度变化，处理模糊。
- 理想的特征描述子：在大小、方向、明暗不同的图像中，同一特征点应具有足够相似的描述子，称为描述子的可复现性。

图像梯度 (1)

- 图像中物体的轮廓和边缘是图像很重要的特征。图像模糊是因为图像中物体的轮廓不明显，像素值灰度变化不剧烈。图像中物体轮廓和边缘变化明显与否如何定义？
- 梯度的方向是函数变化最快的方向。图像中，在物体边缘的梯度值较大，相反，在图像比较平滑的部分，灰度值变化较小，则相应的梯度也较小，图像处理中把梯度的模简称为梯度，由图像梯度构成的图像称为梯度图像。



图像梯度 (2)

- 图像的梯度能反映图像的边缘和轮廓特征，但是梯度的大小受光照的影响。



- 怎么解决：
 - 归一化：一个图片的梯度对于整张图片的光线会很敏感。如果你把所有的像素点都除以2，那么梯度的幅值也会减半。
- 分块：光照与阴影处灰度变化范围大。

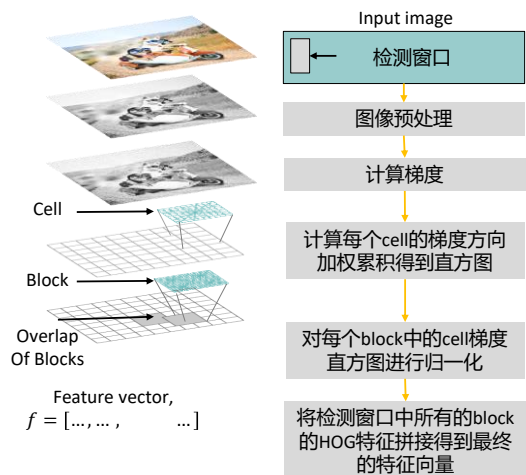
解决方案：方向梯度直方图

- 防止光照的干扰，合理利用图像梯度的策略：
 - 将图像分为若干个块（Block）。
 - 用统计学的方法，计算每块内的像素梯度直方图，防止光照的干扰。
- 方向梯度直方图（Histogram of Oriented Gradient, HOG）特征是一种在计算机视觉和图像处理中用来进行物体检测的特征描述子。它通过计算和统计图像局部区域的梯度方向直方图来构成特征。
- HOG特征结合SVM分类器已经被广泛应用于图像识别中，尤其在行人检测中获得了极大的成功。

HOG算法流程

- HOG算法流程：

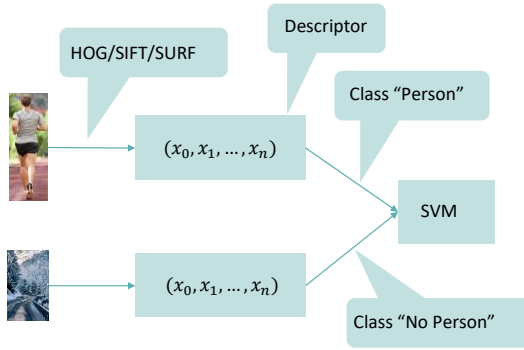
- 图像预处理。
- 计算梯度。
- 统计每个Cell中的梯度方向加权累积直方图。
- 将几个Cell组合成一个Block，归一化Block中每个Cell的梯度直方图。
- 将检测窗口中的所有Block的HOG拼接成一个特征向量，生成特征描述子。



- 首先将图像分成很多小的连通区域，我们把它叫做细胞单元，然后采集细胞单元中各像素点的梯度和边缘方向，然后在每个细胞单元中累加出一个一维的梯度方向直方图。
- 为了对光照和阴影有更好的不变性，需要对直方图进行对比度归一化，这可以通过把这些直方图在图像的更大的范围内（我们把它叫做区间或者block）进行对比度归一化。首先我们计算出各直方图在这个区间中的密度，然后根据这个密度对区间中的各个细胞单元做归一化。我们把归一化的块描述符叫作HOG描述子。

行人检测：HOG+SVM

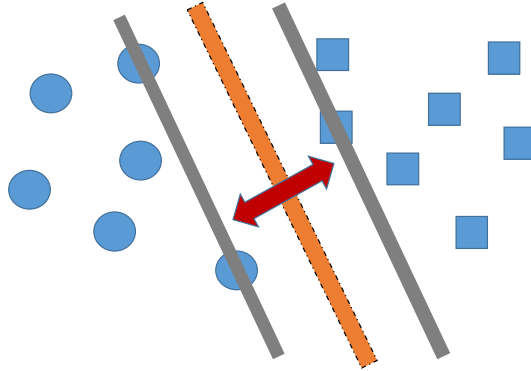
- 在深度学习技术普及之前，行人检测算法基本都是用HOG特征结合SVM分类器的思路实现的。
- HOG是常用的特征描述子。其功能是将图像转换为能表示图像特征的向量的形式，再将对应的向量作为分类模型的训练数据。



- HOG是在图像的局部方格单元上操作的，所以它对图像角度和光照的形变都很好的适应性。只要行人大体上能够保持直立的姿势，一些细微的肢体动作可以被忽略而不影响检测效果。因此HOG特征是特别适合于做图像中的人体检测的。但是HOG特征本身是不支持旋转不变性与多尺度检测的。
- HOG改进点：
 - 在图像预处理时，尽量使用彩色图像。像素点颜色通道中梯度模长最大的梯度作为该像素点的梯度。
 - 在图像预处理时，对于灰度图像，一般为了去除噪点，所以会先利用高斯函数进行平滑。
 - 通常使用的HOG结构大致有三种：矩形HOG（R-HOG），圆形HOG（C-HOG）和中心环绕HOG。它们的单位都是Block。Dalal的试验证明R-HOG和C-HOG的检测效果基本一致，而环绕形HOG效果相对差一些。

支持向量机

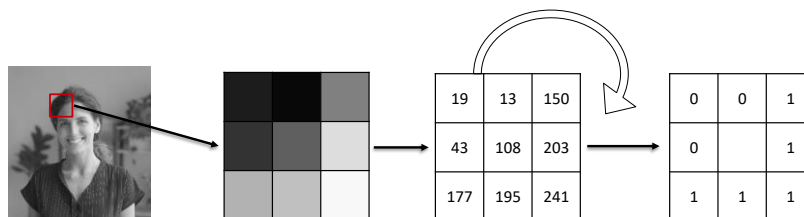
- 支持向量机（Support Vector Machine，常简称为SVM）是一种监督式学习方法，可广泛地应用于统计分类以及回归分析。



- 支持向量机属于一般化线性分类器，这族分类器的特点是他们能够同时最小化经验误差与最大化间隔，因此支持向量机也被称为最大化间隔分类器。
- 我们要讲解的模型是大名鼎鼎的支持向量机SVM，这是曾经在机器学习界有着近乎垄断地位的模型，影响力持续了好多年。直至今日，即使深度学习神经网络的影响力逐渐增强，但SVM在中小型数据集上依旧有着可以和神经网络抗衡的极好效果和模型鲁棒性。

LBP算子核心思想

- 局部二值模式 (Local Binary Pattern, LBP) 是一种可以描述图像局部纹理特征的描述子。LBP具有旋转不变性和灰度不变性等显著的优点。
- 在3*3的窗口内，以窗口中心像素为阈值，将相邻的8个像素的灰度值与其进行比较，若周围像素值大于中心像素值，则该像素点的位置被标记为1，否则为0。这样，3*3邻域内的8个点经比较可产生8位二进制数，即得到该窗口中心像素点的LBP值，这个值反映了该区域的纹理信息。
- LBP能很好的地表达图像的局部纹理信息，在应用过程中，LBP被不断得改进和优化，发展出圆形LBP算子，旋转LBP算子等。

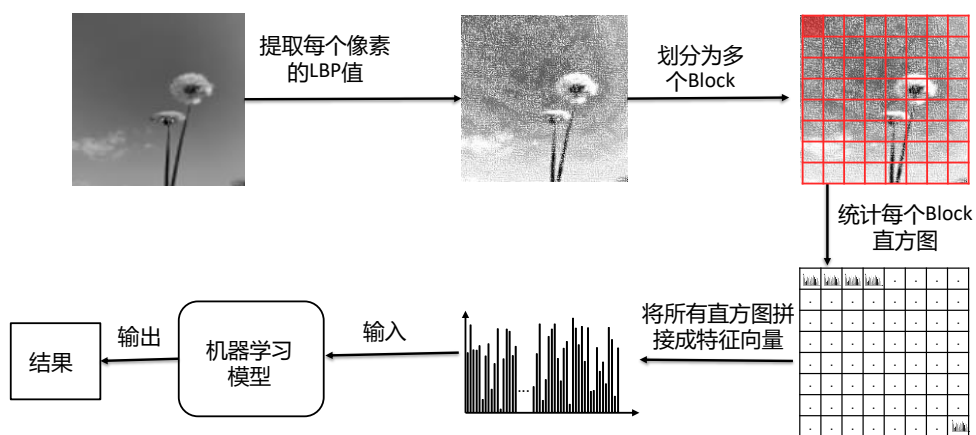


- LBP (Local Binary Pattern) 指局部二值模式，是一种用来描述图像局部特征（纹理）的算子，由于LBP特征计算简单、效果较好，LBP特征在计算机视觉的许多领域都得到了广泛的应用，包括人脸识别，目标检测，唇语识别，表情检测，动态纹理等。
- 中心像素的LBP值反映了该像素周围区域的纹理信息：灰度不变性。
- 旋转不变性：8位的二进制数从左上角的像素点，顺时针一圈。首先不断的旋转圆形邻域内的LBP特征，根据选择得到一系列的LBP特征值，从这些LBP特征值选择LBP特征值最小的作为中心像素点的LBP特征。

提取LBP特征向量的具体流程

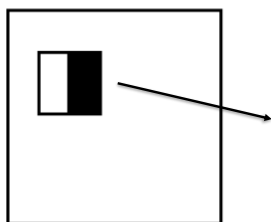
- 对一幅图像进行LBP特征向量提取，分为4个步骤：
 - 提取每个像素的LBP值。
 - 将图像分为多个Block。
 - 统计每个Block中LBP值的直方图。
 - 将所有直方图连接成特征向量。

流程示意图



Haar特征的核心思想

- 哈尔（Haar）特征是一种反映图像的灰度变化的一种特征。它分为四类：边缘特征、线性特征、中心特征和对角特征。用黑白两种矩形框组合成特征模板，在特征模板内用权值乘黑色矩形像素和，减去权值乘白色矩形像素和，来表示模版所在区域图像的哈尔特征值。



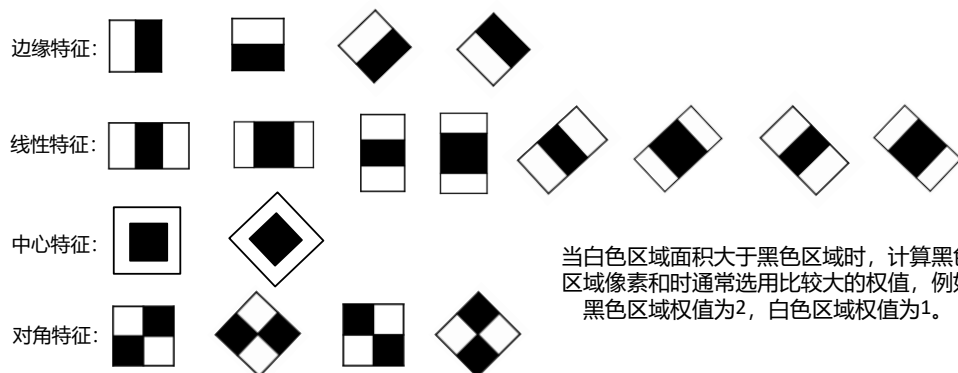
35	57	63	143
74	85	53	164
98	132	199	147
65	102	241	225

当两个权值 ω_1 和 ω_2 均为1时：

$$\text{Haar} = \omega_1 * (63 + 143 + 53 + 164 + 199 + 147 + 241 + 225) - \omega_2 * (35 + 57 + 74 + 85 + 98 + 132 + 65 + 102) = 587$$

不同的哈尔特征矩阵

- 哈尔（Haar）特征使用不同的特征矩阵来提取边缘特征、线性特征、中心特征和对角特征。通过改变特征模板的大小和位置，可在图像子窗口中穷举出大量的特征。下图是常见的哈尔特征结构。

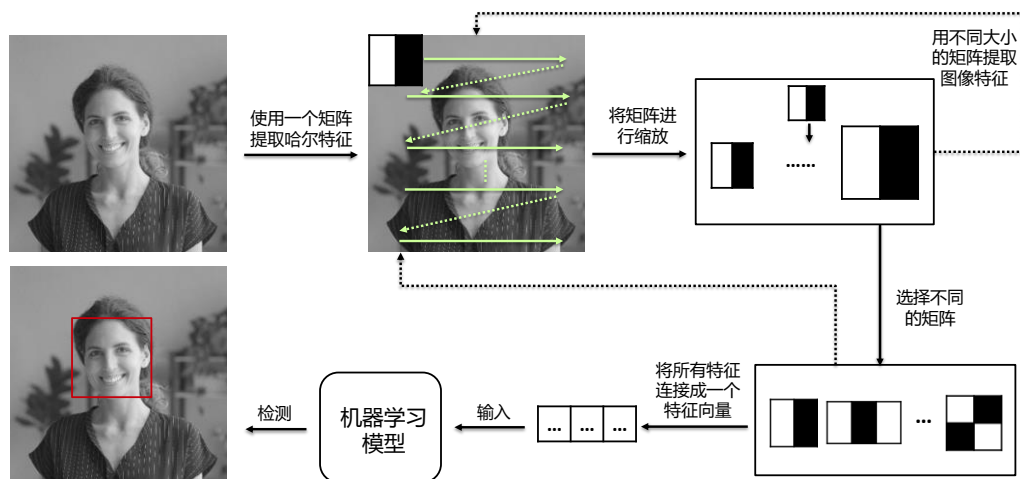


提取哈尔特征的具体流程

- 对一幅图像进行哈尔特征提取，分为4个步骤：
 - 用一个哈尔特征矩阵扫描图像。
 - 缩放矩阵并扫描图像。
 - 选用不同的哈尔特征矩阵扫描图像。
 - 将所有得到的哈尔特征连成特征向量。

- 优点：
 - 计算量小，只有加减运算，提取特征速度很快。
 - 具有移动不变性。
 - 使用不同大小的矩阵使得哈尔特征具有一定的尺寸不变性。
 - 使用不同角度的矩阵使得哈尔特征具有一定的旋转不变性。
- 局限性：
 - 当矩阵大小变大时，哈尔特征的计算量也会变大。
 - 不具有光照不变性。

流程示意图



Haar级联人脸检测

- Haar特征配合Adaboost算法在人脸检测应用中能取得不错的效果。
- Haar特征值能反映图像灰度的变化情况。在人脸图像中，脸部的一些特征能由矩形特征简单地描述，如：眼睛要比脸颊颜色要深，鼻梁两侧比鼻梁颜色要深，嘴巴比周围颜色要深等。使用Haar特征对正向的人脸检测和具备对称性灰度变化的目标都有良好的效果。



- 对非人脸（负样本）更敏感，被强分类器拒绝的，几乎一定不是人脸，经过所有强分类器考验的，才是人脸，级联分类器的速度非常快，因为一幅图像中待检测的区域很多都是负样本。
- Adaboost是一种经典的自适应boosting方法，可以实现高效地二分类判别。Adaboost算法基本原理就是将多个弱分类器进行合理的结合成一个强分类器。弱分类器一般使用单层决策树模型。
- Adaboost每次只训练一个弱分类器，自适应体现在：第N-1次迭代中被错误分类的样本的权值在第N次迭代时会增大，而正确分类样本的权值会减小，并再次用来训练下一个弱分类器。每个弱分类器都有对应权重，分类误差率小的弱分类器权重大，其在最终的分类函数中起着较大的决定作用，而分类误差率大的弱分类器的权重小。
- <https://zhuanlan.zhihu.com/p/100217697>

Adaboost

- AdaBoost (Adaptive Boosting, 自适应提升算法) 是一个具有里程碑意义的算法, 因为它是第一个具有适应性的算法, 即能适应弱学习器各自的训练误差率, 这也是其名称的由来。
- AdaBoost的具体流程为:
 - 先对每个样本赋予相同的初始权重, 每一轮学习器训练过后都会根据其表现对每个样本的权重进行调整, 增加分错样本的权重, 这样先前做错的样本在后续就能得到更多关注, 按这样的过程重复训练出M个学习器, 最后进行加权组合。

- 可以参考: <https://zhuanlan.zhihu.com/p/37358517>。
- 《统计学习方法》李航, Adaboost章节。

目录

1. 计算机视觉概览
2. 数字图像处理基础
3. 图像预处理技术
4. 图像处理基本任务
5. 特征提取和传统图像处理算法
- 6. 卷积神经网络和图像识别**
7. 生成对抗网络和图像生成

特征是关键点

- 人是通过图像的局部特征来理解图像。
- 相同的图片经过不同的视觉系统，也会得到不同的感知。
- 视觉系统就是将看到的图像与脑海中的概念进行关联。

特征

+

认知

边缘 + 纹理



玻璃 or 天空

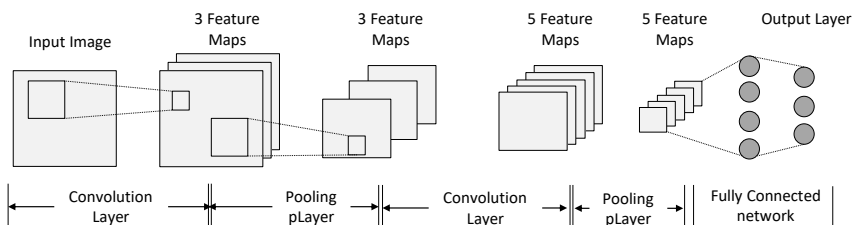


全连接网络处理图像

- 图像在输入全连接网络时往往被展开成一个向量，这会导致图像中像素点的空间关系丢失。图像全连接网络在处理图像时的弊端主要有：
 - 人是通过图像的局部特征来理解图像，而前馈神经网络每个神经元都处理整幅图像的像素点。
 - 计算机存储和处理图像的方式会使前馈神经网络参数巨大。
 - 普通前馈神经网络无法解决图像不变性的问题。

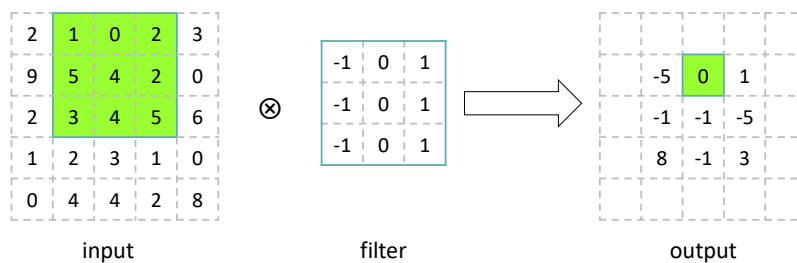
卷积神经网络

- 卷积神经网络（Convolutional Neural Network, CNN）是一种前馈神经网络，它的人工神经元可以响应一部分覆盖范围内的周围单元，对于图像处理有出色表现。
- 它包括卷积层（Convolution Layer），池化层（Pooling Layer）和全连接层（Fully Connected Layer）。
- CNN避免了对图像的复杂前期预处理而可以直接输入原始图像，因而得到了更为广泛的应用。



单卷积核计算

- 卷积核的值就是卷积层中的参数。卷积核通过从特征图左上角按照从左往右，从上往下的顺序扫描整张特征图。每次扫描都会把对应位置的数值相乘，并且相加，求出一个值放入下一个特征图的对应位置中。



多卷积核计算

- 输入图像

- 5*5*3

- Pad = 1

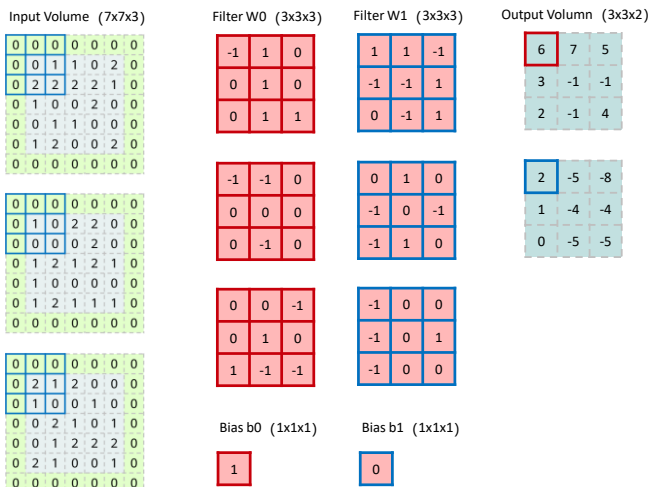
- 卷积核2个

- 3*3*3

- stride = 2

- 特征图

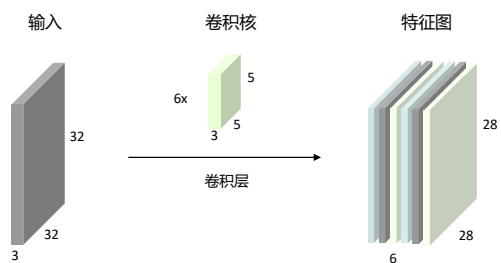
- 3*3*2



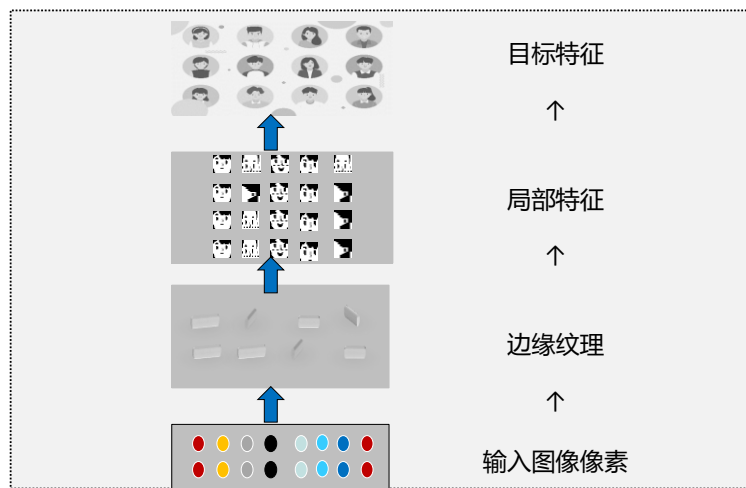
- 输入的特征图中绿色部分就是做了0填充，该图显示了如何在多通道输入以及多卷积核的情况下做卷积计算。
- 卷积核 (convolution kernel)：根据一定规则进行图片扫描并进行卷积计算的对象称为卷积核。卷积核可以提取局部特征。
- 卷积核尺寸 (kernel size)：卷积核是一个3维的矩阵，可以用一个立方体表示，宽w，高h，深度d。深度d由输入的通道数决定，一般描述卷积核尺寸时，可以只描述宽w和高h。
- 特征图 (feature map)：经过卷积核卷积过后得到的结果矩阵就是特征图。每一个卷积核会得到一层特征图，有多个卷积核则会得到多层的特征图。
- 特征图尺寸 (feature map size)：特征图也是一个3维的矩阵，可以用一个立方体图表示，宽w，高h，深度d。深度d由当前层的卷积核个数决定，一般描述特征图尺寸时，可以只描述宽w和高h。
- 步长 (stride)：卷积核在输入图像上滑动的跨度。如果卷积核一次移动一个像素，我们称其步长为1。
- 零填充 (zero padding)：为了提取图像的边缘信息，并且保证输出特征图的尺寸满足要求，可以对输入图像边缘填充一个全为0的边框，边框的像素宽度就是padding。

多核卷积层

- 多核卷积层：通过不同的卷积核提取图像的局部特征信息。

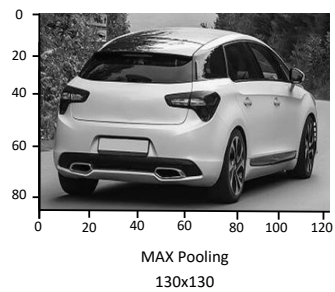
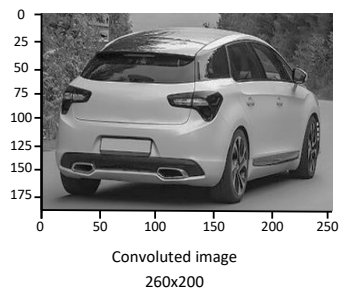


多层卷积层

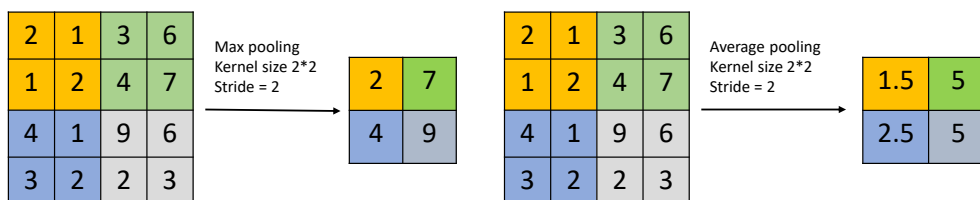


池化层

- 池化一般分为最大池化和平均池化。
- 池化的作用：降低参数量、保留图像显著特征、降低过拟合、扩大感知视野和解决图像不变性。



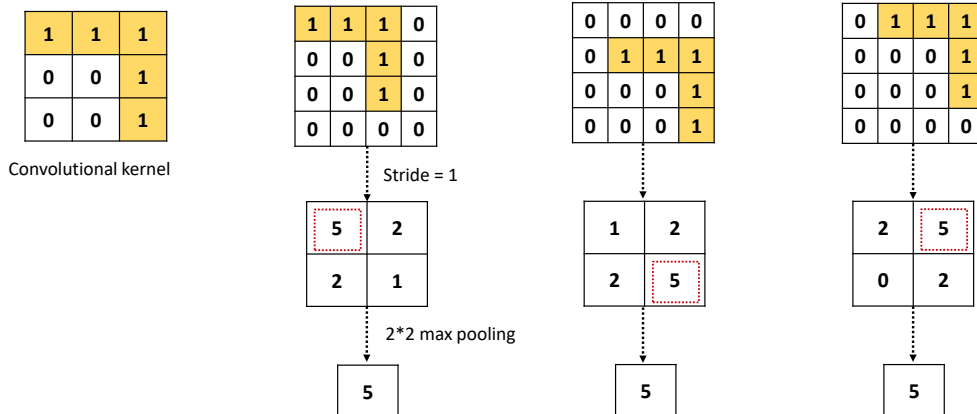
最大池化与平均池化



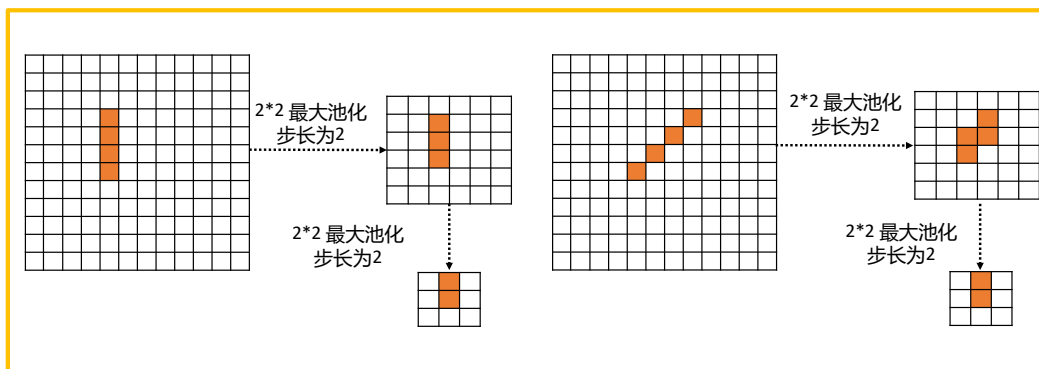
- 池化不改变通道数。

池化的作用可视化 (1)

保留图像显著特征、平移不变性



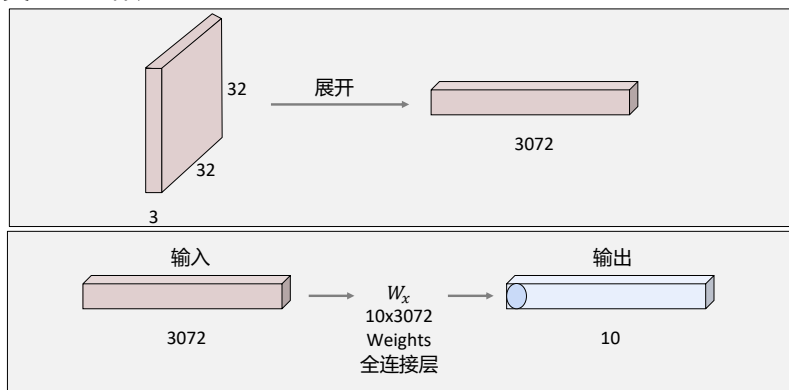
池化的作用可视化 (2)



旋转不变性

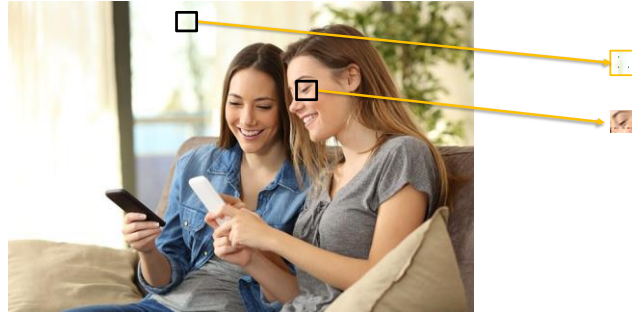
全连接

- 全连接层可以用来将最后得到的特征映射到线性可分的空间，在整个卷积神经网络中起到“分类器”的作用。



CNN核心思想 - 局部感知 (1)

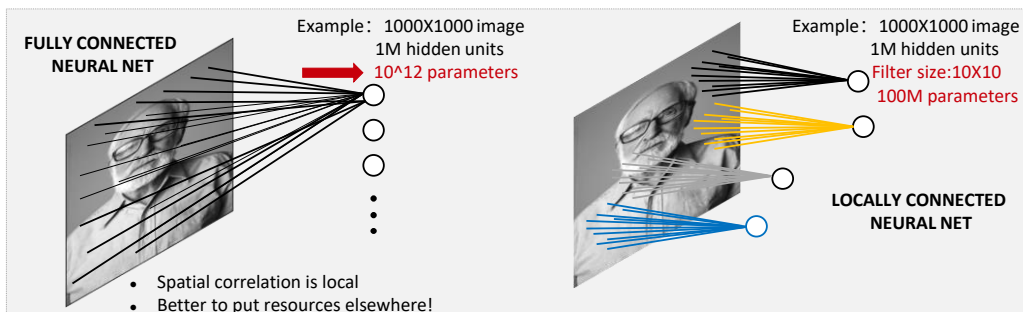
- 一般认为人对外界的认知是从局部到全局的，图像像素点的空间联系也是局部的像素联系较为紧密，而距离较远的像素相关性则较弱。



- 局部感知，一般认为人对外界的认知是从局部到全局的，而图像的空间联系也是局部的像素联系较为紧密，而距离较远的像素相关性则较弱。因而，每个神经元其实没有必要对全局图像进行感知，只需要对局部进行感知，然后在更高层将局部信息综合起来就得到了全局信息。网络部分连通的思想，也是受启发于生物学里面的视觉系统结构。视觉皮层的神经元就是局部接受信息的（即这些神经元只响应某些特定区域的刺激）。

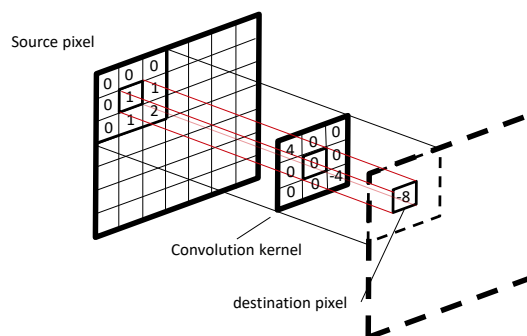
CNN核心思想 - 局部感知 (2)

- 在CNN中，每个神经元只需要收集局部信息。
- 优点：
 - 减少内存需求。
 - 模拟视觉皮层中的神经元，每个神经元一次只收集局部信息。



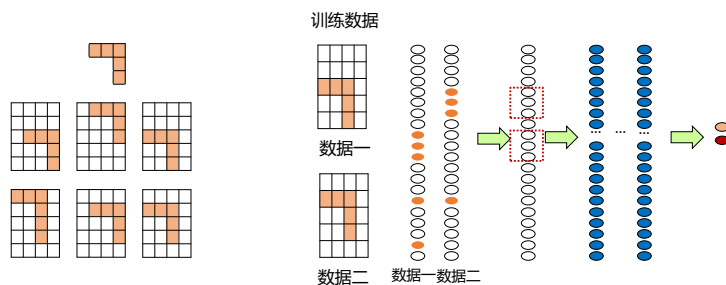
CNN核心思想 - 局部感知 (3)

- 实现方式：使Kernel的尺寸远小于输入图像的尺寸。



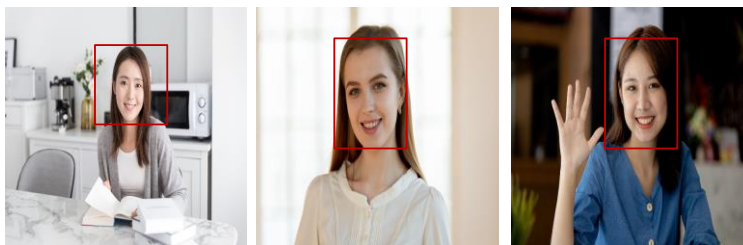
图像不变性

- 左图，白色方块表示值为0的像素，橙色方块圆点表示值为1的像素。不管这个横折在图片中的什么位置，都会被认为是相同的横折。
- 若训练全连接神经网络来完成该任务，如右图所示，那么表达图像的三维张量将会被摊平成一个向量，作为网络的输入，即（Width, Height, Depth）为（5, 5, 1）的图片会被展成维度为25的向量作为网络的输入层。当横折处于不同位置时，展开的向量数据完全不同，给网络的训练和识别造成了困难。



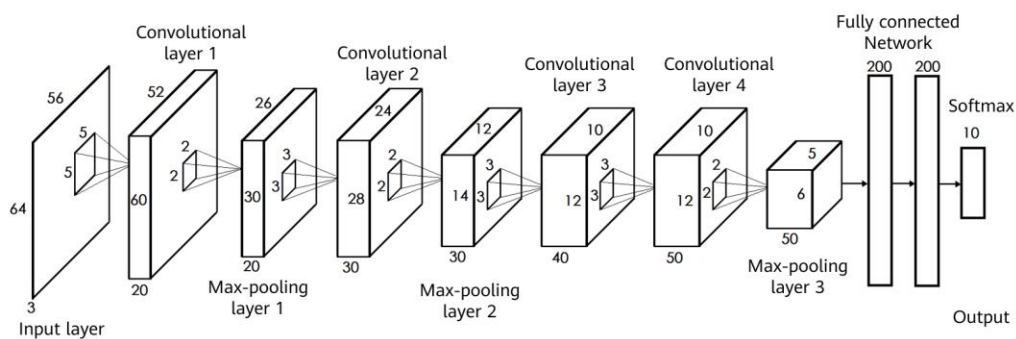
CNN核心思想 - 参数共享

- 优点:
 - 解决图像位置不变性的问题。
 - 减少计算和内存需求。
- 实现:
 - 用参数相同的Kernel去扫描整副图像。



- 参数共享，对输入的照片，用多个filter扫描照片，filter自带的参数就是权重，用同一个filter扫描整张图像且这个filter参数保持不变，就是参数共享。比如有3个filter，每个filter都会扫描整个图像，在扫描的过程中，filter的参数值是固定不变的，即整个图像的所有元素都“共享”了相同的权值。

卷积神经网络全景

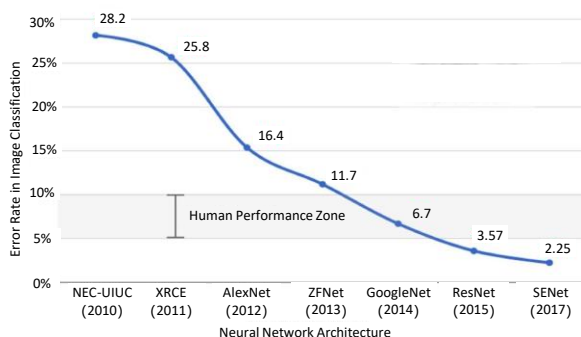


ILSVRC

- 深度学习和卷积神经网络的发展离不开ILSVRC。
- ILSVRC (ImageNet Large Scale Visual Recognition Challenge) 是斯坦福大学举办一个科技比赛。
- 每年度的ILSVRC比赛数据集大概拥有120万张图片，以及1000类的标注，是ImageNet全部数据的一个子集。比赛一般采用top-5和top-1分类错误率作为模型性能的评测指标。

ILSVRC历年成绩

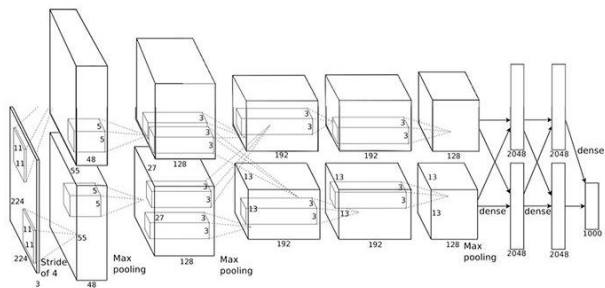
- 从2010年以来，每年的ILSVRC都主要包括以下3个比赛项目：图像分类，单物体定位，物体检测。



- 大赛一共举办了10年，2012年是突破性的一年。AlexNet在2012年ImageNet竞赛中以超过第二名10.9个百分点的绝对优势一举夺冠，从此深度学习和卷积神经网络名声鹊起，深度学习的研究如雨后春笋般出现，AlexNet的出现可谓是卷积神经网络的王者归来。
- 正是因为ILSVRC 2012挑战赛上的AlexNet横空出世，使得全球范围内掀起了一波深度学习热潮。这一年也被称作“深度学习元年”。此后，ILSVRC挑战赛的名次一直是衡量一个研究机构或企业技术水平的重要标尺。因此，即使ILSVRC挑战赛停办了，但其对深度学习的深远影响和巨大贡献，将永载史册。
- 首先它证明了CNN在复杂模型下的有效性，然后GPU实现使得训练在可接受的时间范围内得到结果，确实让CNN和GPU都大火了一把，顺便推动了有监督DL的发展。

AlexNet

- AlexNet是2012年在ILSVRC比赛中夺冠的网络结构。不算输入层，AlexNet只有八层，但是它有60M以上的参数总量，事实上在参数量上比后面的网络都大。
- 由于早期GPU显存的限制，最早的AlexNet包括了双数据流的设计，以让网络中一半的节点能存入一个GPU，用多个GPU训练。



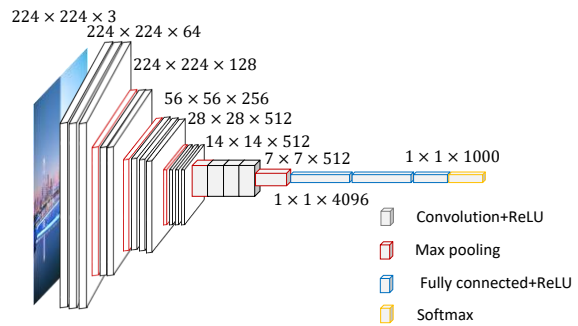
- 使用CUDA加速深度卷积网络的训练，利用GPU强大的并行计算能力，处理神经网络训练时大量的矩阵运算。AlexNet使用了两块GTX 580 GPU进行训练，单个GTX 580只有3GB显存，这限制了可训练的网络的规模。因此作者将AlexNet分布在两个GPU上，在每个GPU的显存中储存一半的神经元的参数。因为GPU之间通信方便，可以互相访问显存，而不需要通过主机内存，所以同时使用多块GPU也是非常高效的。同时，AlexNet的设计让GPU之间的通信只在网络的某些层进行，控制了通信的性能损耗。
- ZFNet是2013ImageNet分类任务的冠军，其网络结构没什么改进，只是调了调参，性能较Alex提升了不少。ZF-Net只是将AlexNet第一层卷积核由11变成7，步长由4变为2，第3, 4, 5卷积层转变为384, 384, 256。这一年的ImageNet还是比较平静的一届，其冠军ZF-Net的名堂也没其他届的经典网络架构响亮。

AlexNet特点

- 闪光点：
 - ReLU：使用ReLU代替Sigmoid激活函数，以达到快速收敛的目的。
 - 重叠的Pooling：Stride<Kernel Size。这种池化策略能减轻过拟合。
 - 数据增强：为防止过拟合，AlexNet对进行原始图像（256*256）随机剪裁为224*224的图像，图像水平单转以及随机增加光照等操作。
 - Dropout：每个隐藏层神经元的输入以0.5的概率输出为0。目的同样是防止过拟合。

VGGNet

- VGGNet (Visual Geometry Group) 是牛津大学VGG组提出的。VGG16相比AlexNet的一个改进是采用连续的几个 3×3 的卷积核代替AlexNet中的较大卷积核 (11×11 , 5×5)。对于给定的感受野 (与输出有关的输入图片的局部大小)，采用堆积的小卷积核是优于采用大的卷积核，因为多层非线性层可以增加网络深度来保证学习更复杂的模式，而且代价还比较小 (参数更少)。
- VGGNet研究深度对卷积网络的影响：使用简单的 3×3 卷积核，不断重复卷积层，池化，最后经过全连接、Softmax，得到输出类别概率。



VGGNet的六种配置

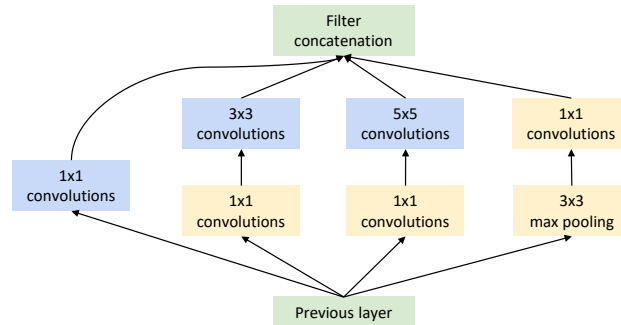
- VGG 在 ImageNet 上的 Top-5 准确度为92.3%。
- VGG16 和 VGG19 是 图 像 和 语 音 的特征提取任务所常用的模块。

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input(224x224 RGB image)					
conv3-128	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
Soft-max					

- VGG模型的基本模块就是一个卷积加一个池化并不断地重复，最后加上用于映射输出的全连接层。
- VGG卷积层之后是3个全连接层。网络的通道数从较小的64开始，然后每经过一个下采样或者池化层成倍地增加，当然特征图大小成倍地减小。最终其在ImageNet上的Top-5准确度为92.3%。

GoogLeNet

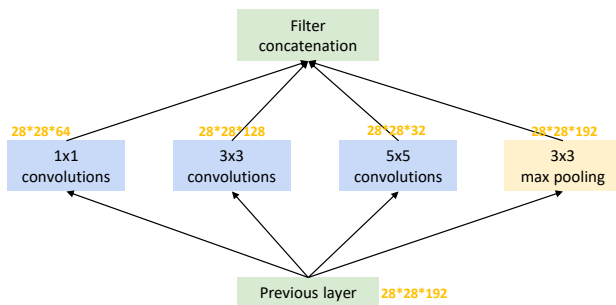
- GoogLeNet的提出是为了探索不同大小的卷积核和更宽的CNN对于分类效果的影响。
- GoogLeNet设计了一种称为inception模块，这个模块使用密集结构来近似一个稀疏的CNN。



- 一个inception同时使用多个不同尺寸的卷积层，以一种结构化的方式来捕捉不同尺寸的信息，很大程度地降低了参数量和计算量。

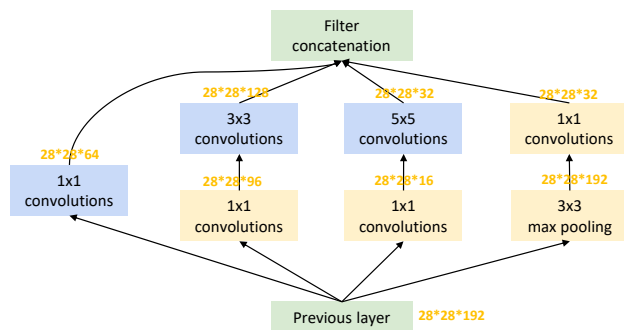
1*1卷积 - 计算量 (1)

- 下面我们通过一个具体的计算案例，来比较使用1*1卷积前后的参数量。
- 若输出特征图的尺寸为28*28*192，
 - 原始Inception该层的参数为 $(1 \times 1 \times 192 \times 64) + (3 \times 3 \times 192 \times 128) + (5 \times 5 \times 192 \times 32) = 387072$ 。



1*1卷积 - 计算量 (2)

- 降维的Inception模块的参数为：
 - $(1 \times 1 \times 192 \times 64) + (1 \times 1 \times 192 \times 96) + (1 \times 1 \times 192 \times 16) + (3 \times 3 \times 96 \times 128) + (5 \times 5 \times 16 \times 32) + (1 \times 1 \times 192 \times 32) = 163328$ 。
- 变换前后，参数量相差223744个。



卷积核的分解（1）

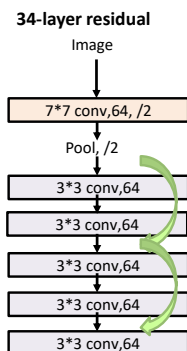
- 卷积核的分解方式1是将一个较大的卷积核分解为多个更小的卷积核。
- Inception V1中， 5×5 的卷积十分消耗计算资源，一个 5×5 卷积所消耗的计算成本是一个 3×3 卷积的2.78倍（ $25/9$ ）。因此Inception V2借鉴了VGGNet的思想，使用两个 3×3 的卷积来代替 5×5 的卷积。
- 两个 3×3 的卷积核参数量： $3 \times 3 \times 2 = 18$ 。
- 一个 5×5 的卷积核参数量： $5 \times 5 = 25$ 。

卷积核的分解（2）

- 卷积核分解的第二种方式是**将一个对称的卷积核分解为多个不对称卷积核**，例如将 3×3 的卷积核尺寸分解为 1×3 和 3×1 的两个卷积。
- 一个 3×3 的卷积分解为一个 1×3 和一个 3×1 的卷积。 3×3 卷积核的参数量为9个；而 1×3 和 3×1 的卷积核参数量共6个。
- 这种分解将 $n \times n$ 的卷积分解为 $1 \times n$ 和 $n \times 1$ 卷积的方式随着 n 增长，计算成本节省也会显著增加。**实践证明，这种分解方式在较早层的网络效果并不明显。**
- **这两种分解都可以降低计算消耗，降低模型的过拟合，同时还增加了一层非线性扩展模型表达能力。**非对称的卷积结构拆分，其结果比对称拆分为几个相同的小卷积核效果更加明显，可以处理更多、更丰富的空间特征，来增加特征的多样性。

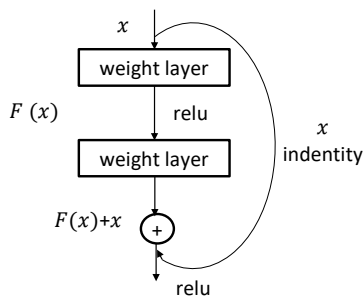
ResNet

- 随着网络深度增加，网络的准确度应该同步增加。但是增加网络深度后，比较靠前的网络层梯度会很小。这意味着这些层基本上学习停滞了，这就产生了**梯度消失问题**。
- 当网络更深时，网络的参数空间更大，优化问题变得更难，因此简单地增加网络深度反而会导致更高的训练误差（**特征退化**）。
- 残差网络的提出就是为了解决以上两个问题。



残差结构

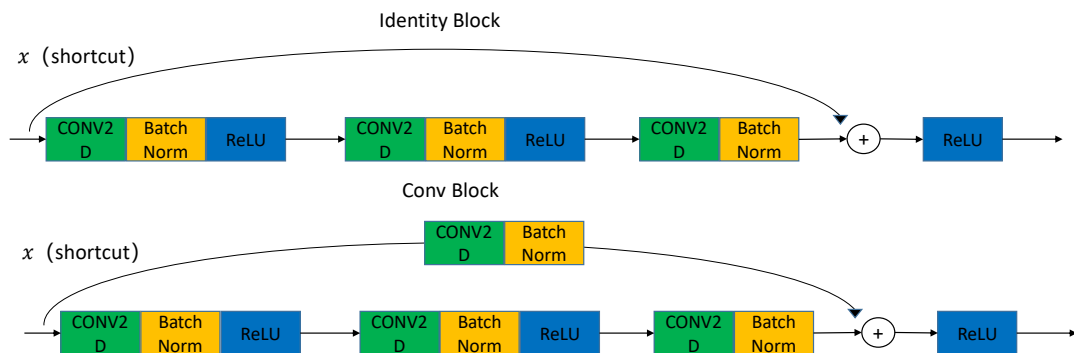
- 残差网络提出了一个可以防止训练时的梯度消失的结构，称为残差结构。
- 残差模块在输入和输出之间建立了一个直接连接，这样新增的网络层仅仅需要在原来的输入层基础上学习新的特征，即学习残差，会比较容易。



- 前向传播中解决了网络退化的问题：使用残差结构来学习冗余层更加容易。
- 反向传播时，误差信号可以不经任何中间权重矩阵变换直接传播到低层，一定程度上可以缓解梯度弥散问题。
- resnet v2：BN和relu位置不同。

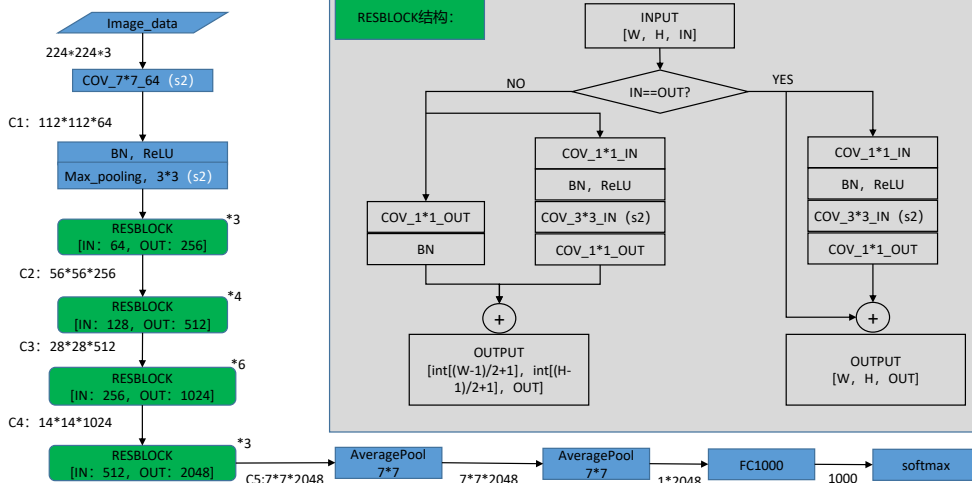
残差块的两种形式

- Identity Block: 输入和输出的维度是一样的，所以可以串联多个。
- Conv Block: 输入和输出的维度是不一样的，它的作用是为了改变特征向量的维度。



- 在两个block函数中，input_tensor都经过若干个Conv2D，BatchNorm和ReLU计算后，得到x，然后与shortcut进行Add操作。不同的是，identity_block中shortcut直接就是input_tensor，而conv_block的shortcut是input_tensor再经过Conv2D和BatchNorm后得到的。这两种block就实现了两种不同形式的shortcut connection。

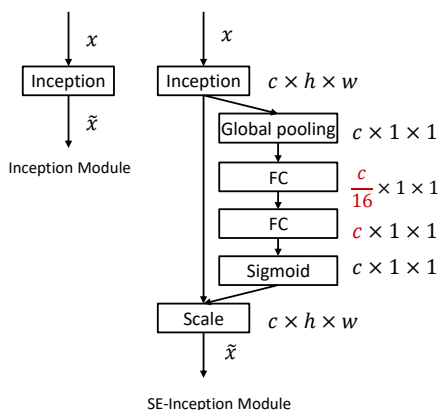
ResNet架构



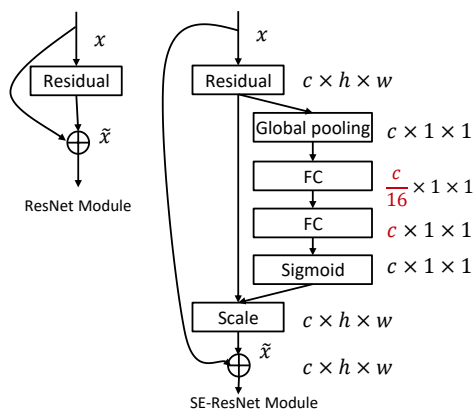
- ResNet从stage2开始，每个stage都有1个conv_block和若干个identity_block，每个block所采用的卷积核尺寸都是3x3，但是每个stage的输出数据的深度逐步增加，又因为卷积操作是叠加的，所以每个stage输出数据的尺寸越来越小。

SENet

- Squeeze-and-Excitation Networks, 2017



158 Copyright © Huawei Technologies Co., Ltd. All rights reserved.



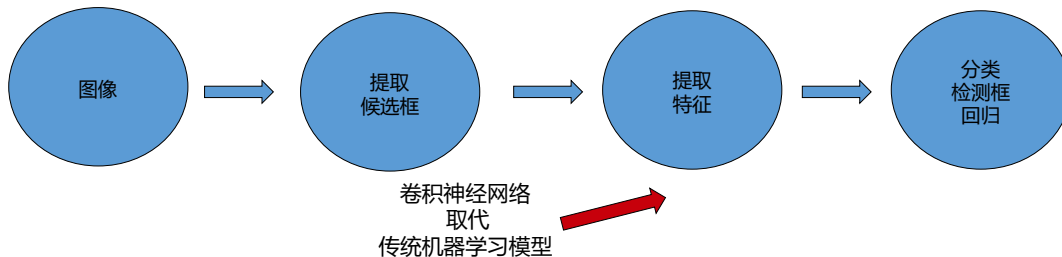
- SENet是ILSVRC 2017的冠军模型，在ImageNet数据集上将top-5 error降低到2.251%，原先的最好成绩是2.991%。
- SENet的全称是Squeeze-and-Excitation Networks，中文可以翻译为压缩和激励网络，主要由两部分组成：Squeeze部分与Excitation部分。
- SENet网络的创新点在于关注channel之间的关系，希望模型可以自动学习到不同channel特征的重要程度。
- 本质上，SE模块是在channel维度上做attention操作，这种注意力机制让模型可以更加关注信息量最大的channel特征，而抑制那些不重要的channel特征。
- SE模块是通用的，这意味着其可以嵌入到现有的网络架构中。

物体检测的主要算法

- 物体检测（Object Detection）的任务是找出图像中所有感兴趣的目标（物体），确定它们的类别和位置。物体检测的主要算法分为两类：
 - 传统的物体检测算法：Cascade + HOG/DPM + Haar/SVM以及上述方法的诸多改进、优化。
 - 基于深度学习的算法，大致可以分为两大流派：
 - 两步走（two-stage）算法：先产生候选区域（Region Proposal, RP），然后再进行CNN分类（RCNN系列）。常见的two stage物体检测算法有：R-CNN、SPP-Net、Fast R-CNN、Faster R-CNN和R-FCN等。
 - 一步走（one-stage）算法：直接在网络中提取特征来预测物体分类和位置（yolo系列）。常见的one stage物体检测算法有：OverFeat、YOLO系列、SSD和RetinaNet等。

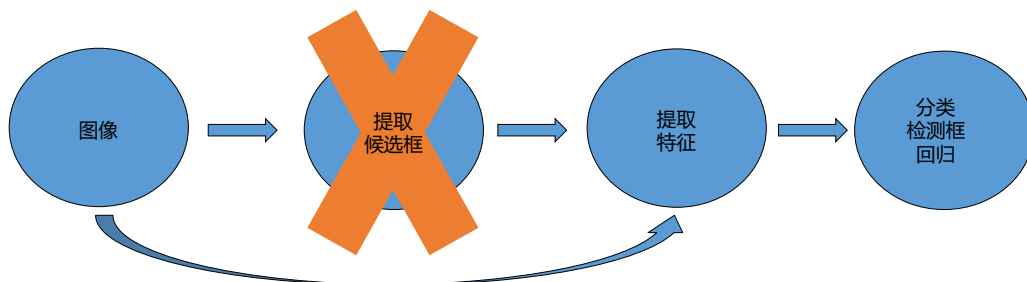
两阶段物体检测方法

- 传统的物体检测方法需要人工设计特征提取因子，然后将提取到的特征送入SVM中统一进行分类回归，检测效果的好坏主要取决于特征因子的质量。
- 深度学习出现后，用卷积神经网络实现特征的自动提取，从而大大提高了检测效果。出现了一批先生成候选框，再调整候选框位置和类别预测的两阶段物体检测算法，如RCNN系列，Pyramid网络等。



单阶段物体检测方法

- 为了适应实时推理场景，进一步提高检测速度，取消提取候选框，直接对整幅图像进行检测和分类，就是单阶段物体检测方法。常见的一阶段算法有YOLO系列，RetinaNet等，具有实时性强，准确率高等优点。

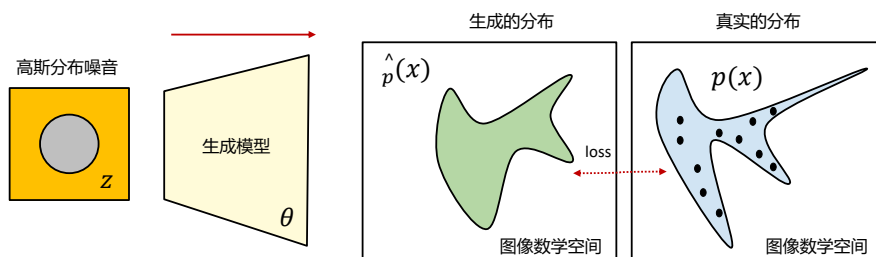


目录

1. 计算机视觉概览
2. 数字图像处理基础
3. 图像预处理技术
4. 图像处理基本任务
5. 特征提取和传统图像处理算法
6. 卷积神经网络和图像识别
- 7. 生成对抗网络和图像生成**

生成对抗网络简介

- 生成对抗网络（Generative Adversarial Network, GAN）最初由Goodfellow等人于2014年提出，是近年来最流行的生成模型之一。生成对抗网络可以看作由两种模型组成，分别是生成模型和判别模型，通过生成模型和判别模型的博弈来进行训练。原始的生成对抗网络的目标是学习真实数据分布，生成和数据集中的图像相似但不完全相同的图像。



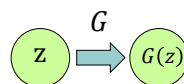
- 不同种类的生成对抗网络能够做不同的事，它们并不是只能用于生成图像，还能生成影像；生成句子，例如聊天机器人；生成语音，例如语音去噪。

生成模型和判别模型

- 生成模型

- 负责生成图像。

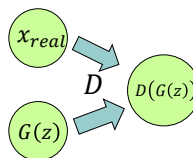
- 输入：高斯噪声向量 z 。
 - 输出：生成图像 $G(z)$ 。



- 判别模型

- 负责判断输入图像是否来自数据集。

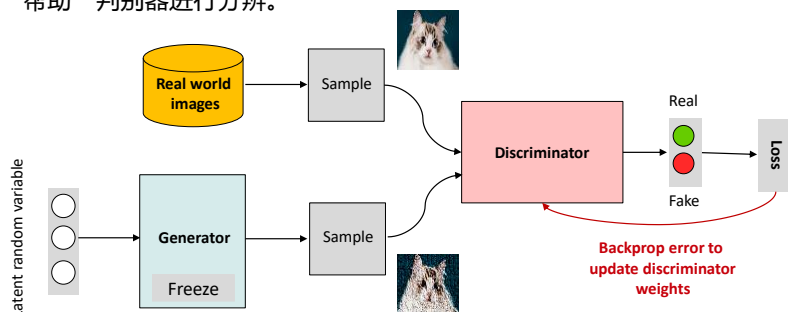
- 输入：从数据集中抽样得到的真实图像 x_{real} 或者生成的假图像 $G(z)$ 。
 - 输出：真假程度 $D(G(z))$ 。



- 训练生成对抗网络可以用基于成熟的梯度下降算法的优化器，例如Adam优化器。在训练时可以使用反向传播算法来更新参数。值得注意的是，和其他深度学习不同的是，由于生成模型和判别模型的目标不同，因此需要分开训练生成模型和判别模型。此外，由于一开始生成模型只能生成一些质量很差的图像例如噪声，而判别模型也只能判别噪声和真实图像，因此生成模型和判别模型需要交替训练，你追我赶，共同进化。

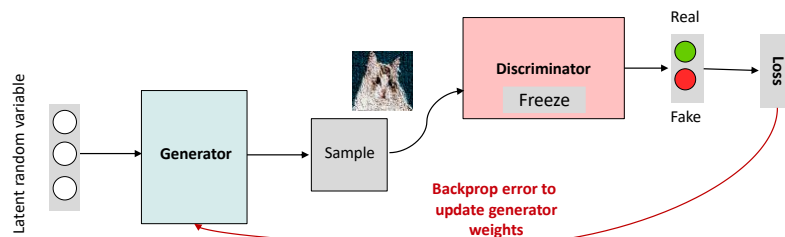
训练判别器

- 训练判别器的目标是能分辨真实图像和生成图像。即当输入图像为真实图像时，希望输出结果越接近“1”越好，当输入生成图像时，希望输出越接近“0”越好。在训练判别器时，我们需要固定住生成器中的参数。我们可以很直觉地理解：如果不固定生成器的参数，可以通过降低生成图像的质量来“帮助”判别器进行分辨。



训练生成器

- 训练生成器的目标是能“骗过”判别器，让判别器判断不出真实图像和生成图像。即当生成的图像进入判别器时，希望输出结果越接近“1”越好。在训练生成器时，我们需要固定判别器中的参数。我们可以很直觉地理解：如果不固定判别器的参数，可以通过降低判别器的判别性能，把任何图像都当成真实图像。



生成对抗网络训练中的问题 - 欠拟合和过拟合

- 生成对抗网络中同样存在欠拟合以及过拟合的问题。
 - 生成对抗网络中的欠拟合：生成对抗网络无法生成高质量的图像。这是生成对抗网络中一直在研究的一个问题，即如何提高生成图像的质量。而导致欠拟合的很重要的原因之一就是模型的生成能力不足。因此，我们可以通过适当增加原始网络中隐藏层的层数以及神经元的数量来尝试缓解欠拟合的问题。此外还可以通过改变网络结构等方法来缓解欠拟合的问题，例如使用深度卷积生成对抗网络（DCGAN）。
 - 生成对抗网络中的过拟合：生成对抗网络生成的图像和数据集中的图像一致或仅进行了平移等。我们同样可以通过增加训练集的数量以及使用Dropout等方法来缓解过拟合的问题。

生成对抗网络训练中的问题 - 模式崩塌

- 生成对抗网络中的模式崩塌。
 - 模式崩塌是生成对抗网络中非常特殊的一个问题，即模型无法生成多样性的图像。例如，如果我们将MNIST作为数据集，希望生成一个数字“0”到数字“9”的图像。生成模型很可能会“偷懒”，只生成容易生成的数字，例如“1”；而不去生成难度较大的数字，例如“8”。我们可以很直觉地理解：只要生成模型生成质量足够好的“1”，同样能够达到“骗过”判别模型的目的。
 - 模式崩塌是目前仍然尚待解决的问题，幸运的是，已经有很多论文提出了各种方法来缓解模型崩塌的问题。例如VEEGAN，AdaGAN，MAD-GAN等。

生成对抗网络训练中的问题 - 不稳定性

- 生成对抗网络训练不稳定的问题。
 - 梯度消失的问题：在刚开始训练生成对抗网络时，很容易出现梯度消失的问题。这是由于判别模型训练得“太好”或者“太差”导致生成模型无法继续训练。
 - 难以收敛的问题：在训练生成对抗网络的后期，很可能出现生成模型难以收敛的问题。
- 目前，已经有很多论文提出改变生成对抗网络的目标函数来缓解生成对抗网络训练不稳定的问题。例如LSGAN, WGAN, WGAN-GP等。

缓解训练不稳定问题

- 在训练GAN时，可以通过以下两个小技巧来缓解训练不稳定的问题：
 - 单侧标签平滑：训练判别器时，将真实图像的标签设置成0.9而不是1，这样可以防止判别器梯度过大，使开始训练时，判别模型不要训练得“太好”。
 - 在训练判别模型时，每个Batch只放入真实图像或者只放入生成的图像，而不是将真实图像和生成图像打乱放入。这样同样也可以缓解训练不稳定的问题。

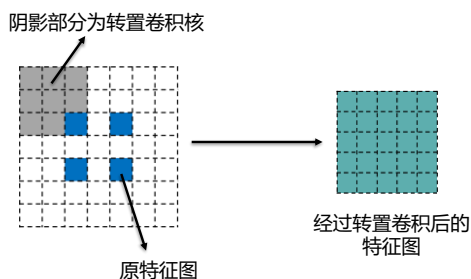
- 小技巧就是为了防止判别模型训练得太好，导致训练生成模型时梯度消失无法训练。

深度卷积生成对抗网络简介

- 深度卷积生成对抗网络（Deep Convolutional Generative Adversarial Network, DCGAN）可以简称为DCGAN。它是由Alec Radford等人于2016年提出的生成模型。它改进了原始生成对抗网络的结构，用卷积层取代了多层感知机结构，使模型能够生成更高质量的图像。

DCGAN生成模型结构

- 我们已经知道，判别模型本质上是在做分类任务。我们可以通过搭建类似LeNet-5的卷积神经网络来作为深度卷积生成对抗网络中的判别模型。通过卷积，池化（下采样）以及全连接来对生成图像以及真实图像进行分类。
- 判别模型的输入为图像，输出为一个标量，可以使用卷积神经网络中的卷积和下采样进行分类。那生成模型的输入为低维噪声，输出为高维图像，需要用到**转置卷积或上采样**来“放大”图像。



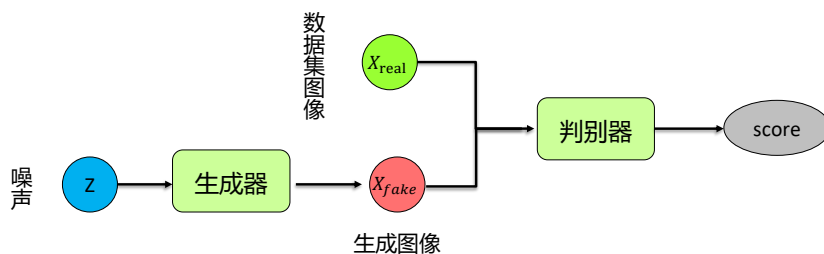
- 转置卷积是卷积操作的逆运算，但本质上来说就是在做卷积运算。具体的操作是将上一层的特征图进行特定的Padding，再将卷积核的值做中心对称变换，再进行卷积计算。
- 生成模型中也可以使用上采样（Upsampling）后接普通的卷积层。上采样实质上是将原特征图的值沿行和列进行复制多份。

条件生成对抗网络简介

- 条件生成对抗网络 (Conditional Generative Adversarial Network) 也被称为CGAN，最初由 Mehdi Mirza等人于2014年提出。相比原始的GAN，CGAN最大的特点是可以通过输入文字或图像，来控制生成的图像。但是要做到这一点，是需要通过**监督式学习**来完成的，即需要带标签的数据集进行训练。例如最早提出的CGAN通过输入数字0到9（条件）来生成对应的手写数字图像。

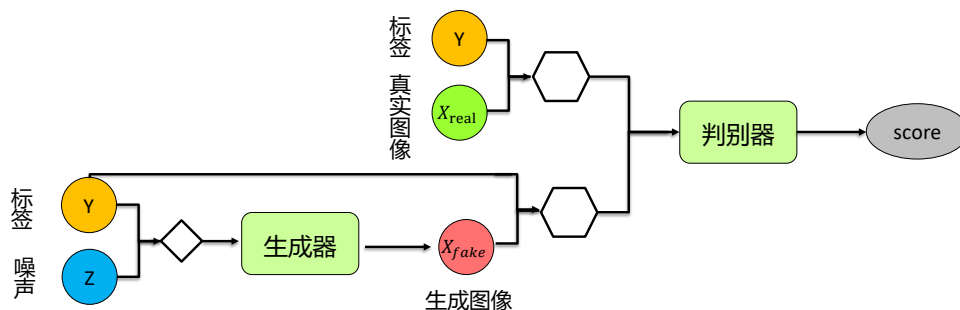
生成对抗网络回顾

- 首先来简单回顾一下原始生成对抗网络的结构。生成模型的输入为噪声，判别模型的输入为生成的图像和数据集中真实的图像。可以看到整个网络训练过程中只用到了数据集中的图像，图像并不需要标签。



条件生成对抗网络结构

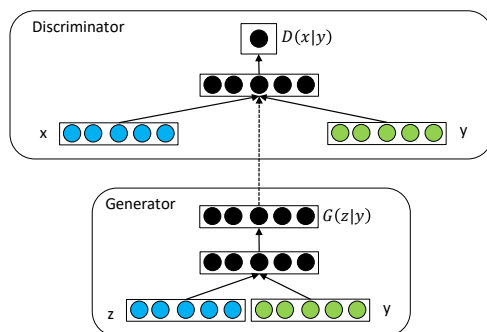
- 在CGAN中，判别器的输入不仅需要数据集中的图像，还需要该图像的标签（Label）。对于生成器来说，输入的不仅仅只是噪声，还有想要生成图像的标签，再将生成的图像和设置的标签放入判别器中。而判别器会输出判断的结果。



- CGAN的判别器不仅要考虑生成的图像是否接近真实图像，还要考虑生成图像和标签是否匹配。
- 当然，label也不一定是一个名词，同样可以一个描述的句子，但是句子需要做相应的处理放入网络中，例如转化成词向量。

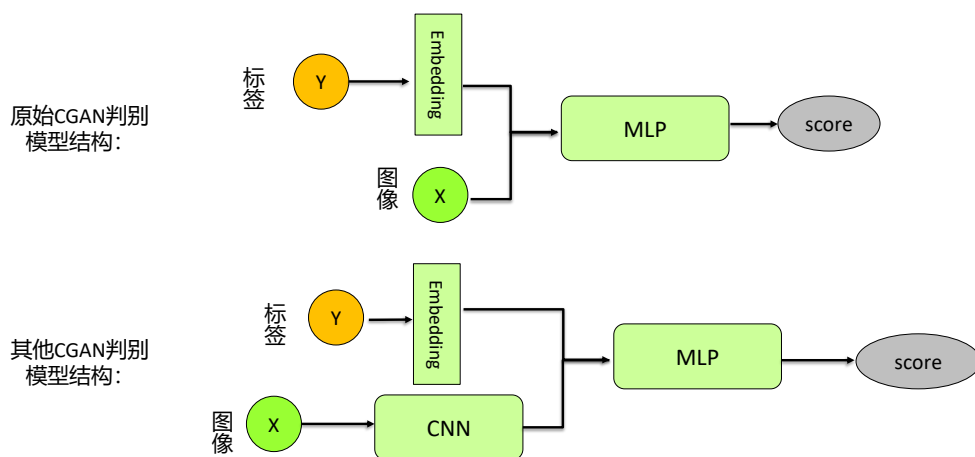
如何组合标签和图像

- CGAN中判别器的输入不仅需要图像，还需要标签，那图像和标签是如何组合在一起作为输入的呢？最早提出的CGAN中，作者是将标签和图像转化为向量，将向量进行拼接（Concat），形成一个更长的向量。将这个长向量作为输入。



- 如果要使用卷积层搭建CGAN，除了将向量相乘，也可以尝试将标签向量转化为矩阵（大小和噪声或图像相同），和噪声或图像在通道的维度进行拼接。标签向量和图像向量相乘或相加并不是没有依据的，Self-attention layer中，位置矩阵和词向量矩阵就是通过相加的方式融合的。

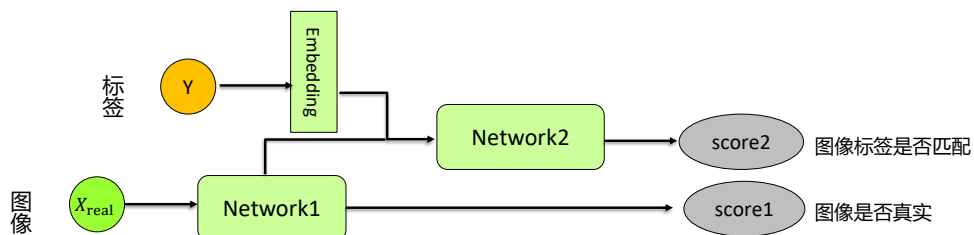
判别模型结构 (1)



- 下一种结构，CNN输出的矩阵通过拉成向量，和标签向量进行拼接，输入到MLP中。

判别模型结构（2）

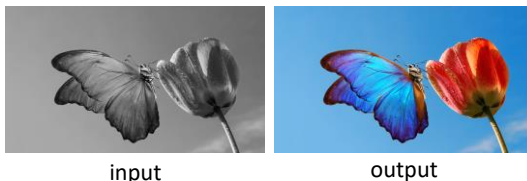
- 此外还有一种比较特殊的判别模型结构。我们还能很直觉地发现下面的判别模型结构可能会得到更好的结果，因为原始判别模型得到的分数，在训练生成模型的时候，生成模型无法知道是因为生成的图像不够真实还是因为生成的图像未和标签匹配。



Pix2Pix简介

- Pix2Pix和CGAN (Conditional GAN) 一样都是进行监督式学习，和CGAN的思想也很类似，但是Pix2Pix的标签和数据都是图像，它可以做到图像翻译。Pix2Pix可以做很多事，例如白天转黑夜，轮廓转图像，图像上色等。

黑白图像上色

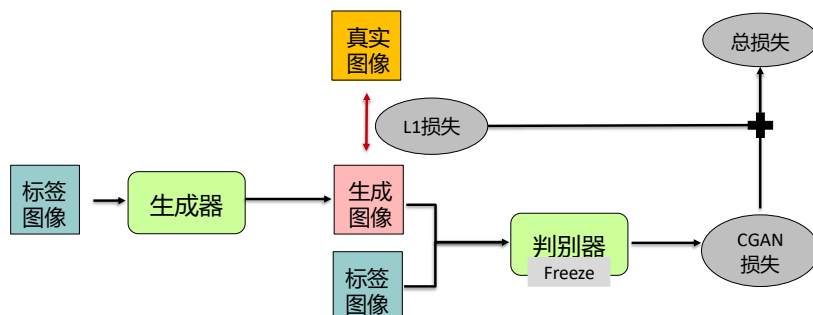


input

output

Pix2Pix结构

- 下面是Pix2Pix训练生成器时的模型结构，值得注意的是，生成器的输入没有加入噪声，这是由于生成器会自动过滤掉大部分噪声，添加噪声并不能对生成图像造成多少影响。



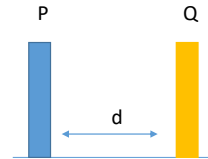
- 在原文中，作者只使用了dropout作为不确定因此，而没有在标签中加入噪声。
- 可以看到在训练生成器时，目标函数除了要“骗过”判别器以外，还需要考虑生成图像和原始图像的L1损失。这是因为仅考虑CGAN的损失，生成图像中并没有非常好的效果，比如会产生原标签中和真实图像中都不存在的物体，所以作者提出将损失设为CGAN损失加L1损失，能取得更好的效果。
- 另外在街道场景翻译的任务中，如果仅考虑CGAN的损失，无法生成停车线等信息，而将CGAN加上L1损失后，可以生成更多细节。

WGAN

- Wasserstein GAN (Wasserstein Generative Adversarial Networks, WGANs) 引入了 Wasserstein 距离重新定义 GAN 模型的相似度衡量指标。Wasserstein 距离又称为 Earth-Mover (EM) 距离，用于衡量两个分布之间的距离。

EM:

- 将一个分布 P 视为土堆，和另一个分布 Q 作为目标。
- d 为推土机移动土堆的平均距离。



- 原始GAN函数背后其实是基于Kullback-Leibler divergence (简称KL散度) 和Jensen-Shannon divergence (简称JS散度) 这两个重要的相似度衡量指标的。但是KL散度和JS散度度量时都有一个共同的问题：如果两个分布 P 和 Q 距离较远，完全没有重叠的时候，KL散度是没有意义的，在学习的时候，这就意味着在这一点上的梯度为0，即梯度消失了。因此WGAN的作者引入了Wasserstein距离重新定义GAN模型的相似度衡量指标。我们引入Wasserstein距离，Wasserstein距离又称为Earth-Mover (EM) 距离，用于衡量两个分布之间的距离。

思考题

1. （多选题）卷积神经网络的核心思想是?
 - A. 局部感知
 - B. 参数共享
 - C. 全连接

- 答案：AB

思考题

2. （单选题）数字图像处理可以分为三个层次，根据处理内容的抽象性，从低到高依次为？
- A. 图像理解、图像分析、图像处理
 - B. 图像分析、图像处理、图像理解
 - C. 图像处理、图像分析、图像理解

- 答案：C

本章总结

- 本章讲解了图像处理理论和应用。详细介绍了数字图像处理基础和图像预处理技术。总结了图像处理的基本任务，并介绍了图像处理中的传统特征提取算法和使用深度学习的图像处理方法。最后着重介绍了图像识别、图像生成等相关理论与实践。

缩略语

- NLP: Natural Language Processing, 自然语言处理
- API: Application Programming Interface, 应用程序编程接口
- HOG: Histogram of Oriented Gradients, 方向梯度直方图
- LBP: Local Binary Pattern, 局部二值模式
- SVM: Support Vector Machine, 支持向量机
- AdaBoost: Adaptive Boosting, 自适应增强, 是一种机器学习方法
- CNN: Convolutional Neural Network, 卷积神经网络

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。
Bring digital to every person, home, and
organization for a fully connected,
intelligent world.

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.



语音处理理论与实践



前言

- 语音技术在近几年渐渐开始改变我们的生活和工作方式。对于某些设备来说语音更是成了人与之交流的主要方式，其中最为贴近生活的包括语音助手、智能音箱和机器人等。那么，机器是如何听懂我们说了什么并且答复我们的问题呢？本章节将带领大家来揭开语音处理技术的面纱。

目标

- 学完本课程后，您将能够：
 - 了解语音处理的基础知识及应用
 - 掌握语音处理的基本步骤
 - 掌握语音处理的主要技术
 - 了解语音处理的难点

目录

1. 语音处理介绍

- 语音处理介绍
 - 语音预处理基础
 - 语音信号分析及特征提取

2. 语音识别

3. 语音合成

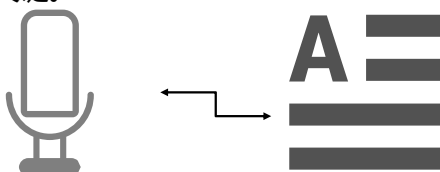
4. 传统声学模型GMM-HMM

5. 混合模型DNN-HMM

6. 高级语音模型

语音处理简介

- **语音信号处理**(Speech Signal Processing)简称**语音处理**，是用以研究语音发声过程、语音信号统计特性、语音自动识别、机器合成以及语音感知等各种处理技术的总称。
- 由于现代的语音处理技术都以数字计算为基础，并借助微处理器、信号处理器或通用计算机加以实现，因此也称**数字语音信号处理**。
- 语言信息主要包含在**语音信号的参数**之中，因此准确而迅速地提取语言信号的参数是进行语音信号处理的关键。



语音处理主要应用场景

- 技术类

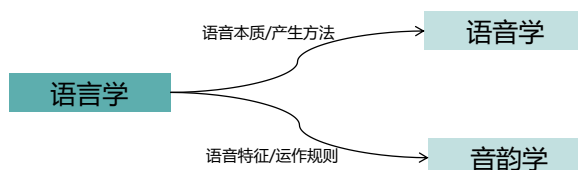
- 语音预处理
- 语音识别
- 语者识别
- 语音翻译
- 语音合成
- 声纹识别
- 语音编码

- 场景类

- 人机交互
- 安防
- 智慧家庭
- 智慧城市
- 养老
- 教育
- 客户服务

语音学 (1)

- 语音学是语言学的一个分支，是研究人类语言声音的学科。主要研究语言的发音机制，语音特性和在言谈中的变化规律。
- 狭义的语音学对应英语中phonetics一词，关注的重点在具体语音本质以及产生语音的方法。与之相对的是音韵学（或称音系学），研究音位或语音区别特征在某种语言中运作的抽象规则和语音的系统。
- 广义的语音学是指语音学和音韵学这两大方面研究的总合。



- 由于语音学的研究内容关系到发音动作，语声特性以及听感，而人类的不同语言集体各有自己的语音特点，因此现代语音学的研究需同时具备自然科学和社会科学的知识作为基础。

语音学 (2)

- 语音学大致可以分为四大类型，包含声音从产生到被接受的整个流程的研究
 - 发音语音学：研究言语的声音是怎样通过口腔中的发音器官（比如：唇、牙齿、舌头、声带等）产生出来的。
 - 声学语音学：研究怎样对言语的声音进行声学分析，比如声波的频率、时长、振幅等。
 - 听觉语音学：研究人耳如何接受声音，即人耳对语音的听觉感知。
 - 语言语音学：结合声音，社会环境，个人习惯，语言规律研究声音。
 - 同一区域不同的人对同一个词、同一句话的发音是不同的。
 - 同一国家不同地域的人对同一个词、同一句话的发音可能天差地别。
 - 相同的人在不同场合、不同情绪下对同一个词、同一句话的发音是不同的。
 - 相同的词或者短语在不同的上下文语境中，发音是不同的。

目录

1. 语音处理介绍

- 语音处理介绍
 - 语音预处理基础
- 语音信号分析及特征提取

2. 语音识别

3. 语音合成

4. 传统声学模型GMM-HMM

5. 混合模型DNN-HMM

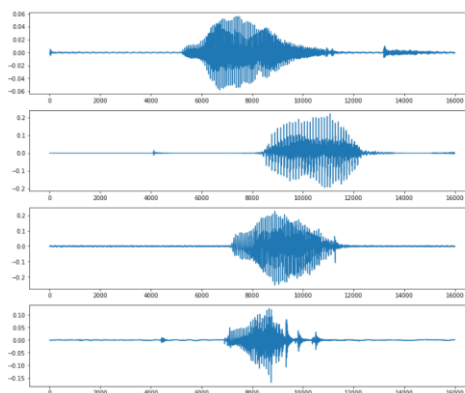
6. 高级语音模型

语音数据

- 语音 (Speech) : 带有语言信息的声音, 是由声音 (Acoustic) 和语言 (Language) 的组合体。
- 语音信号特点:
 - 语音信号会随着时间的变化而变化, 它是一种非平稳态的时变信号。但是, 由于人的发音受口腔肌肉的影响, 语音信号在一个短时范围内相对稳定, 可以看做是一个准稳态的过程, 所以语音信号具有短时平稳性。
 - 短时分析技术贯穿于语音信号分析的全过程。
 - 语音由口腔肌肉运动而产生, 肌肉的运动相较于语音的频率是非常缓慢的。
 - 短时范围一般认为在10-30ms。

语音信号预处理

- 语音文件一般为wav格式。下图可视化的音频文件，横坐标为采样点数，纵坐标为幅度。
- 首先需要对音频信号进行预处理，图示音频信号主要的问题在于：
 - 波形数据的分布非常不均匀。
 - 开头和结尾的静音部分。
 - 波形中含有噪音部分。



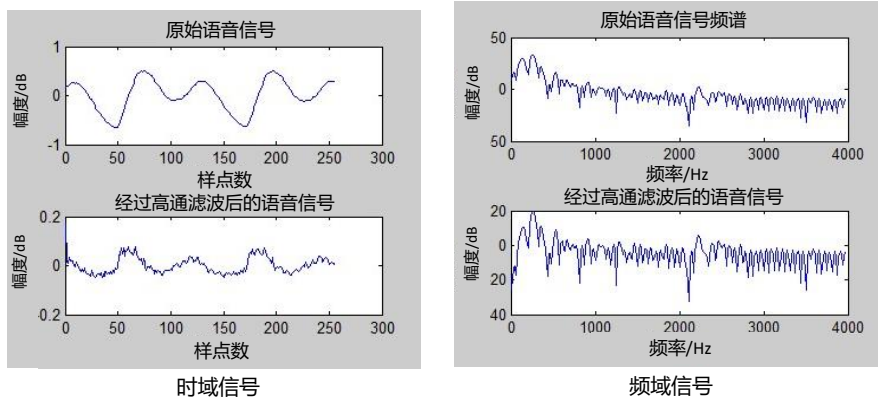
- https://alphafan.github.io/posts/tf_sr_wav_pp.html
- 横坐标为采样点数，纵坐标为幅度。
- 波形数据的分布非常不均匀，有的波形出现在音频快结束的时候，有的在中间，有的则在最开始。
- 开始和结束有很长时间的数据是没有声音的。
- 有一些wav中在除了主体的波之外的位置，有一些小的波，而这些短音频非常可能是外界的噪音。

语音信号预处理步骤

- 语音信号预处理一般包括：
 - 数字化：将从传感器采集的模拟语音信号离散化为数字信号；
 - 预加重：预加重的目的是为了对语音的高频部分进行加重，**去除口唇辐射的影响，增加语音的高频分辨率**；
 - 端点检测：从语音信号中**识别并消除长时间静音段**，减少环境对信号的干扰；
 - 分帧：因为语音的短时平稳性，所以要进行“短时分析”，即将信号分段，每一段称为一帧（一般10-30ms）；
 - 加窗：语音信号的分帧是采用可移动的有限长度窗口进行加权的方法来实现的。加窗的目的是减少语音帧的截断效应。常见的窗有：矩形窗、汉宁窗和汉明窗等。

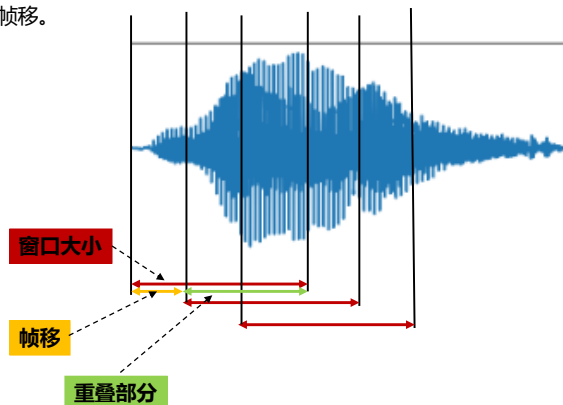
语音信号预处理 - 预加重

- **口唇辐射**：口唇辐射会引起能量损耗，在高频段比较明显，在低频段影响较小。因此为了消除影响，可以采用一个高通滤波器对信号进行预加重，来加强高频部分分辨率。



语音信号预处理 - 分帧

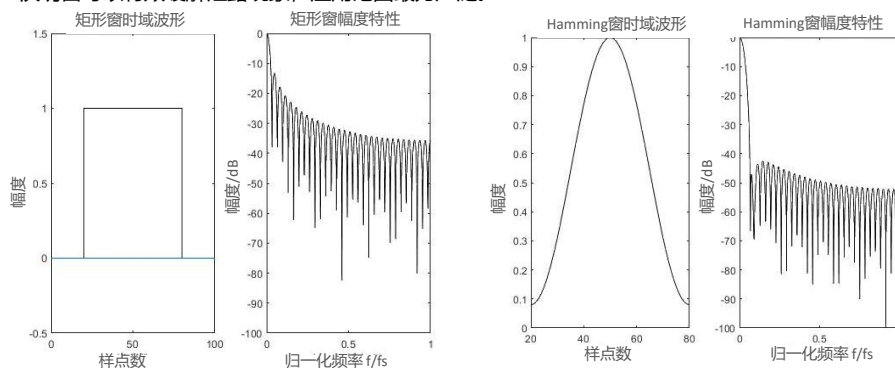
- **分帧**：将不定长的音频切分成固定长度的小段。
 - 分帧虽然可以采用连续分段的方法，但一般要采用交叠分段的方法，这是为了使帧与帧之间平滑过渡，保持其连续性。前一帧和后一帧的交叠部分称为帧移。



- 通常帧长=20~50ms，帧移=10ms。

语音信号预处理 - 加窗

- **加窗**：分帧后，每一帧的开始和结束都会出现间断。因此分割的帧越多，与原始信号的误差就越大，加窗就是为了解决这个问题，使成帧后的信号变得连续，并且每一帧都会表现出周期函数的特性。
 - 不同的窗函数会影响到语音信号分析的结果。矩形窗平滑性较好，但波形细节丢失，并且会产生泄露现象，而汉明窗可以有效缓解泄露现象，应用范围最为广泛。



- 矩形窗的主瓣宽度小于汉明窗，具有较高的频谱分辨率，但是矩形窗的旁瓣峰值较大，因此其频谱泄漏比较严重。相比较，虽然汉明窗的主瓣宽度较宽，约大于矩形窗的一倍，但是它的旁瓣衰减较大，具有更平滑的低通特性，能够在较高的程度上反映短时信号的频率特性。

目录

1. 语音处理介绍

- 语音处理介绍
- 语音预处理基础
- 语音信号分析及特征提取

2. 语音识别

3. 语音合成

4. 传统声学模型GMM-HMM

5. 混合模型DNN-HMM

6. 高级语音模型

语音特征 (1)

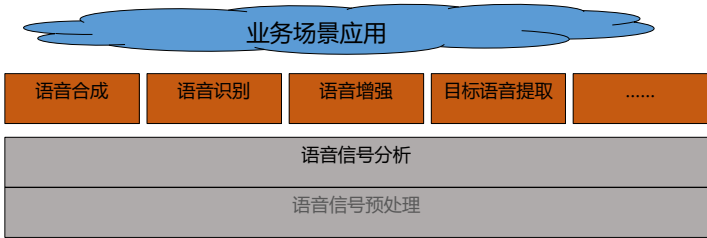
- 语音特征，类似于机器学习中的特征的重要性，当分析一个语音文件时，需要一些量可以很好的反应语音的本质，以便于后续流程的使用。因此，语音特征的设计是语音处理中很重要的一环。
- 语音特征是描述语音的核心信息，在语音模型构建中起重要作用。
- 语音特征：
 - 包含区分音素的有效信息：良好的时域分辨率和频域分辨率；
 - 分离基频F0以及它的谐波成分；
 - 对不同说话人具有鲁棒性；
 - 对噪音或信道失真具有鲁棒性；
 - 有着良好的模式识别特性：低维特征，特征独立。

语音特征 (2)

- 特征提取方法一般有：
 - 线性预测系数 (LPC)
 - LPC倒谱系数 (LPCC)
 - 线谱对参数 (LSP)
 - 共振峰率 (前三个共振峰)
 - 短时谱
 - Mel频率倒谱系数 (MFCC)
 - 感知线性预测 (PLP)
- 在语音识别和语者识别方面最常用的语音特征就是梅尔倒谱系数 (Mel-Frequency Cepstral Coefficients, 简称MFCC) 。

语音信号分析 (1)

- 狭义上语音信号分析不包括语音信号预处理的过程，包括对语音信号的降噪、平滑等预处理，但是通常将这两个过程统一称为语音信号分析
- 语音信号分析的重要性：
 - 语音合成音质的好坏和语音识别率的高低，很大程度上取决于对语音信号分析的准确度和精度。
 - 语音信号分析是语音合成、语音识别、语音增强、目标语音提取等技术的基础和前提，只有将语音的特征进行了准确且有针对性的分析后，才能更好的支持对应的技术。对应的技术才能更好的用在不同的业务场景中。



语音信号分析 (2)

- 对语音信号进行分析的方法有很多，要根据具体的需求来选取合适的分析方法，根据分析的特征不同，大致可以分为以下几类：
 - 时域分析
 - 频域分析
 - 倒频域分析
 - 小波域分析
 -
- 如果从分析方法的维度进行划分的话，可分为以下两类：
 - 模型分析方法
 - 非模型分析方法

语音信号的时域分析

- 语音信号的时域分析就是分析和提取语音信号的时域参数。时域分析是最早使用也是应用最广泛的一种分析方法（语音信号本身就是时域信号），通常用于语音分割、预处理、分类等最基本的参数分析和应用。有如下特点：
 - 物理意义明确，直观，可以获得一些语音中重要的参数；
 - 实现简单。
- 语音信号的时域参数：
 - 短时能量；
 - 短时过零率；
 - 短时自相关函数；
 - 短时平均幅度差函数。

语音信号的频域分析

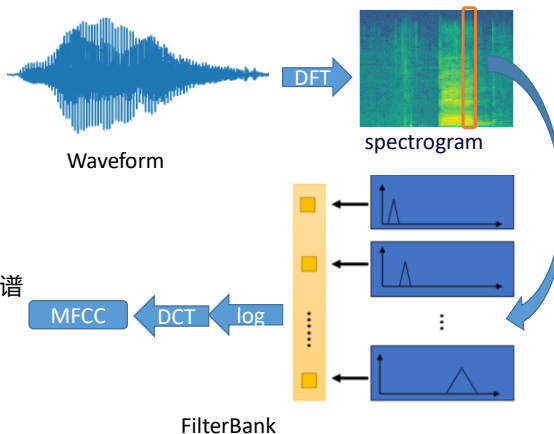
- 语音信号的频域分析就是分析和提取语音信号的频域参数。
- 最常用的频域分析方法为傅里叶分析法。语音信号是一个非平稳过程，因此需要用短时傅里叶变换对语音信号进行频谱分析。通过语音信号的频谱可以观察它们的共振峰特性、基音频率和谐波频率。

语音特征

- 在语音识别和语者识别方面最常用的语音特征就是梅尔倒谱系数（Mel-Frequency Cepstral Coefficients，简称MFCC）。

- MFCC提取过程：

- 预加重
- 分帧
- 加窗
- 傅里叶变换
- 通过三角带通滤波器得到Mel频谱
- 倒谱分析（取对数，做逆变换）



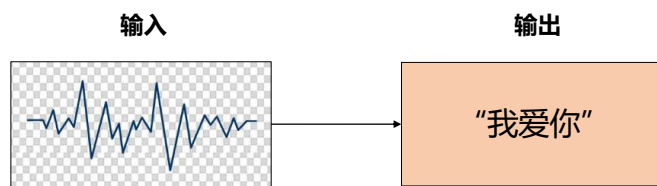
- 为什么要求对数？因为人耳对声音响度不是线性的。耳蜗是一个滤波器组，小于1000Hz为线性响应，大于1000Hz为对数响应。人耳对低频信号更加敏感。求对数后把小信号与大信号的差距变小了，原来两者相差 10^5 倍，经过求对数后，两者相差100dB，100和 10^5 相比，两者差距被缩小了，便于我们进行分析和处理，如果不求对数的话，相差这么多倍，小信号可能就被淹没了。感觉和归一化的过程有点类似。

目录

1. 语音处理介绍
- 2. 语音识别**
3. 语音合成
4. 传统声学模型GMM-HMM
5. 混合模型DNN-HMM
6. 高级语音模型

语音识别

- 语音识别技术就是**让机器通过识别和理解把语音信号转变为相应的文本或命令**的技术。
- 语音识别技术所涉及的领域包括：信号处理、模式识别、概率论和信息论、发声机理和听觉机理、人工智能等等。



语音识别现状

- 语音识别属于人工智能中的感知智能，其应用已经进入家电、通信、汽车、医疗、家庭服务等各个领域。如：语音助手、机器人和智能音箱等。
- 目前，AI语音识别的准确率早已超越人类，且得益于大模型的发展，语音模型可以实现多语种、多领域的语言识别、语音合成等功能。
- 但是局限于当前语音采样设备和数据集问题，语音识别仍然存在一些问题，无法完全达到人类的水平（如方言、远场嘈杂环境下的识别等）。

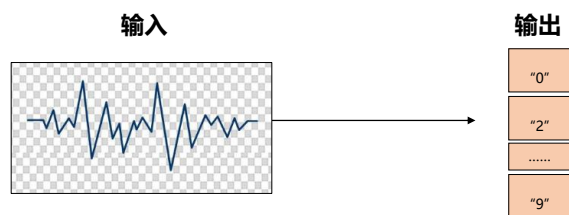
语音识别的难点

- 语音识别任务的难点：
 - 地域性；
 - 场景性；
 - 生理性。
- 鸡尾酒会问题：多个说话人的语音信号会重叠在一起，使得机器难以将它们分离成独立的信号，从而导致识别性能明显下降。
- 一句话总结语音识别的难点就是**不确定性**很强。同一个词、同一句话受到不同因素的影响可能发音完全不同。

- 地域性：一个四川人说“我爱你”，湖南人说“我爱你”，江西人说“我爱你”，同一句话，是不是发音不同，有的甚至完全不一样。这就是地域性对语音识别带来的困难。
- 场景性：再考虑一下我们在不同的场合说相同的一句话，是不是会根据不同的场合、面对不同的人，语音语调会有不同？还有就是在我们心情好时说一句话和心情坏时说一句话，是不是也会存在这个问题。这就是场景性和情绪性带来的问题。
- 生理性：还有一个影响最大的，就是每个人的音色和发音习惯不同，也会导致同一个词设置同一个声母、韵母，不同的人产生的声波是不一样的。

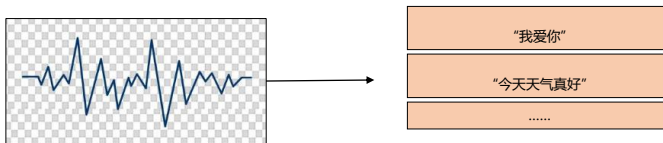
语音识别任务分类 - 孤立词识别

- 孤立词识别：语音处理发展前期是对少量的孤立词进行识别：
 - 输入的语音文件是只包含一个单词的文件（整句语音输入后进行切分），之后通过模型去识别该文件是属于哪个单词，常用的模型有GMM-HMM（对每个词进行建模）。

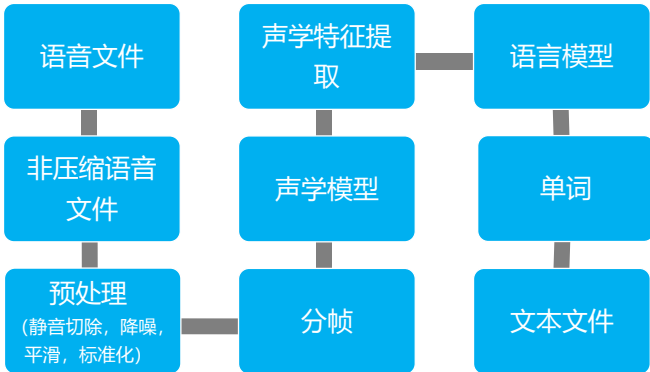


语音识别任务分类 - 连续词识别

- 连续词识别：在实际中少量孤立词识别不能满足实际的应用需求，大部分的需求需要对连续的一句话进行识别，那么如果还是用少量孤立词识别算法的话存在问题：
 - 需要切分整个文件到孤立词，这个对于人工需求很大，且不能保证准确性，因为很多词之间的发音是有粘连的。
 - 即便是完美切分开了，但是我们实际使用中需要使用到的单词量非常大，孤立词识别所使用的匹配策略在此处没有任何的优势，甚至由于词汇量太大而变成劣势。

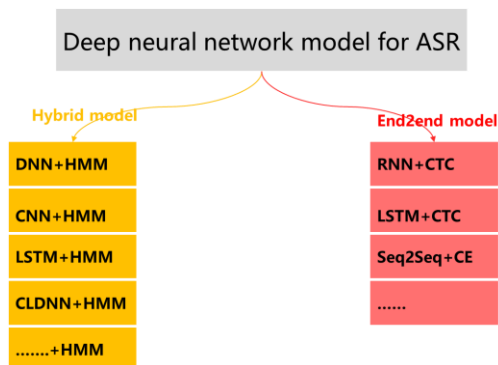


传统语音识别任务处理流程



语音识别算法

- 传统语音识别算法：GMM-HMM。
- 基于深度学习的语音识别算法的种类非常多，新的模型和算法也不断被提出。纵观这些模型算法，大致可以分为两个大方向。
 - 混合模型，也称为hybrid模型
 - 端到端模型，也称为end2end模型
- 大模型识别算法：whisper、SenseVoice等。



语音识别应用



- 直播实时字幕
- 会议实时记录
- 即时文本录入
- 语音助手

目录

1. 语音处理介绍
2. 语音识别
- 3. 语音合成**
4. 传统声学模型GMM-HMM
5. 混合模型DNN-HMM
6. 高级语音模型

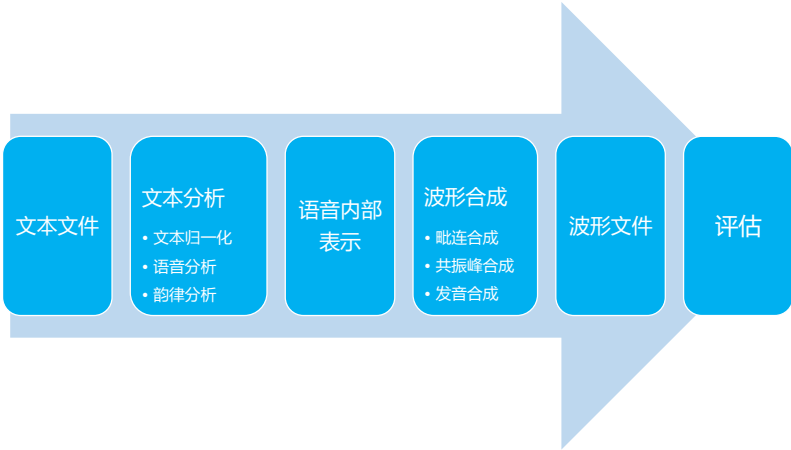
语音合成

- **语音合成**，又称**文语转换**（Text-To-Speech）技术，能将任意文字信息转化为相应语音朗读出来。
- 语音合成涉及声学、语言学、数字信号处理、计算机科学等多个学科技术，是中文信息处理领域的一项前沿技术。
- 为了合成出高质量的语音，除了依赖于各种规则，包括语义学规则、词汇规则、语音学规则外，还必须对文字的内容有很好的理解，这也**涉及到自然语言理解**的问题。

语音合成系统

- 一个完整的语音合成系统过程是先将文字序列转换成音韵序列，再由系统根据音韵序列生成语音波形。其中：
 - 第一步涉及语言学处理，例如分词、字音转换等，以及一整套有效的韵律控制规则；
 - 第二步需要先进的语音合成技术，能按要求实时合成出高质量的语音流。
- 语音合成技术的研究已有两百多年的历史，但真正具有实用意义的近代语音合成技术是随着计算机技术和数字信号处理技术的发展而发展起来的，主要是让计算机能够产生高清晰度、高自然度的连续语音。

语音合成处理流程



文本分析

- 语音识别中的文本分析主要的工作是把文本数据转换成语音内部表示（Phonemic Internal Representation）。具体内容包括：
 - **文本归一化**：对形形色色的自然文本数据进行预处理或者归一化，包括句子的词例还原，非标准词，同形异义词排歧等；
 - **语音分析**：文本归一化之后的下一步就是语音分析，具体方法包括通过大规模发音词典，字位-音位转换规则；
 - **韵律分析**：分析文本中的平仄格式和押韵规则，这里主要包含三方面的内容，包括：韵律的机构，韵律的突显度，音调。

语音合成方法

- 在语音合成技术的发展过程中，早期的研究主要是采用参数合成方法，后来随着计算机技术的发展又出现了波形拼接的合成方法。
- **参数合成**
 - 在语音合成技术的发展中，早期的研究主要是采用参数合成方法。值得提及的是Holmes的**并联共振峰合成器**（1973）和Klatt的**串/并联共振峰合成器**（1980），只要精心调整参数，这两个合成器都能合成出非常自然的语音。但准确提取共振峰参数比较困难，合成语音的音质难以达到实用要求。
- **波形拼接**
 - 自八十年代末期至今，语言合成技术又有了新的进展，特别是**基音同步叠加（PSOLA）方法**的提出（1990），使基于时域波形拼接方法合成的语音的音色和自然度大大提高，自然度比以前基于LPC方法或共振峰合成器的自然度要高，并且基于PSOLA方法的合成器结构简单，易于实时实现。

语音合成算法

- 基于HMM的参数合成
- WaveNet
- Tacotron
- Deep Voice 3

语音合成使用场景

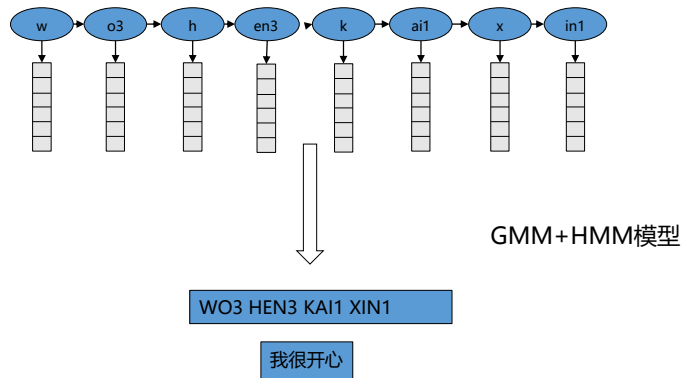
- 语音导航
- 有声读物
- 电话回访
- 智能教育



目录

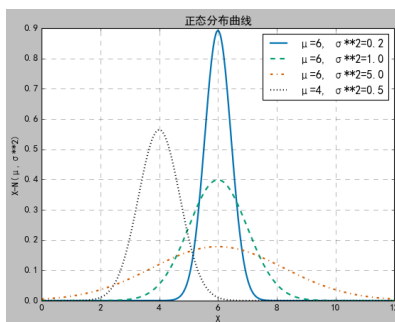
1. 语音处理介绍
2. 语音识别
3. 语音合成
4. **传统声学模型GMM-HMM**
 - GMM
 - HMM
 - GMM-HMM
5. 混合模型DNN-HMM
6. 高级语音模型

语音识别算法 - GMM+HMM



高斯混合模型定义

- 高斯混合模型，英文名：Gaussian Mixture Model（缩写GMM）。高斯混合模型就是用高斯概率密度函数（正态分布曲线）精确地量化的事物，它是一个将事物分解为若干的基于高斯概率密度函数（正态分布曲线）形成的模型。



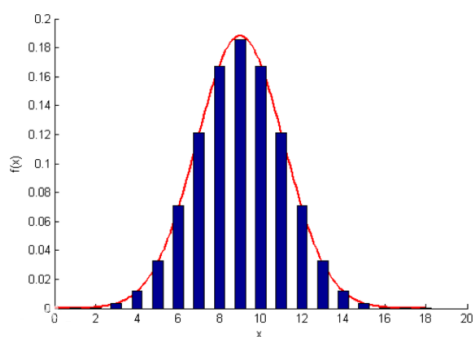
高斯分布

- 高斯分布，又名正态分布，最早由棣莫弗在求二项分布的渐近公式中得到。高斯在研究测量误差时从另一个角度导出了它，是一个在数学、物理及工程等领域都非常重要的概率分布，在统计学的许多方面有着重大的影响力。
- 若随机变量 x 服从一个数学期望为 μ ，方差为 σ^2 的正态分布，记为 $N(\mu, \sigma^2)$ 。其概率密度函数为正态分布的期望值 μ 决定了其位置，其标准差 σ 决定了分布的幅度。当 $\mu = 0, \sigma = 1$ 时的正态分布是标准正态分布。公式如下：

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

高斯分布曲线

- 正态曲线呈钟型，两头低，中间高，左右对称因其曲线呈钟形。
- 标准差越大，曲线越扁平；反之，标准差越小，曲线越瘦高。



单高斯模型

- 当样本数据 X 是一维数据时，高斯分布遵从下方概率密度函数：

$$P(x|\theta) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

(其中 μ 为数据均值， σ 为数据标准差)

- 当样本数据 X 是多维数据时，高斯分布遵从下方概率密度函数：

$$P(x|\theta) = \frac{1}{(2\pi)^{\frac{D}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{(x-\mu)^T \Sigma^{-1} (x-\mu)}{2}}$$

(其中 μ 为数据均值， Σ 为协方差， D 为数据维度。)

最大似然法介绍

- **最大似然法** (Maximum Likelihood) 也叫极大似然估计，是一种具有理论性的点估计法。最大似然估计是一种统计方法，它用来**求一个样本集的相关概率密度函数的参数**。
- 此方法的基本思想是：当从模型总体随机抽取 n 组样本观测值后，最合理的参数估计量应该使得从模型中抽取该 n 组样本观测值的概率最大，而不是像最小二乘估计法旨在得到使得模型能最好地拟合样本数据的参数估计量。



最大似然估计

- 假设有 N 个独立数据点，均服从某种分布 $Pr(x; \theta)$ ，我们想找到一组参数 θ ，使得生成这些数据点的概率最大，这个概率就是：

$$\prod_{i=1}^N Pr(x_i; \theta)$$

- 称为似然函数（Likelihood Function）。通常单个点的概率很小，连乘之后数据会更小，容易造成浮点数下溢，所以一般取其对数，变成：

$$\sum_{i=1}^N \log Pr(x_i; \theta)$$

- 称为log-likelihood function。接下来就可以进行求导，然后求得使得上面式子值最大的参数 θ 。我们认为取得这些观察值的可能性是很小的，但是参数 θ 却使得这一切以最大可能性的发生了。

单高斯模型参数学习

- 对于单高斯模型，我们可以用最大似然法估算参数 θ 的值， $\theta = \operatorname{argmax}_{\theta} L(\theta)$ ，这里我们假设了每个数据点都是独立的，似然函数由概率密度函数给出：

$$L(\theta) = \prod_{j=1}^N P(x_j|\theta)$$

- 由于每个点发生的概率都很小，乘积会变得极其小，不利于计算和观察，因此通常我们用Maximum Log-Likelihood 来计算。因为 Log 函数具备单调性，不会改变极值的位置，同时在 0-1 之间输入值很小的变化可以引起输出值相对较大的变动：

$$\log L(\theta) = \sum_{j=1}^N \log P(x_j|\theta)$$

高斯分布模型参数估计

- 对于一个样本数据集，如果我们使用高斯概率分布来拟合这个数据集，那么求解高斯分布模型参数的步骤如下所示：
 - 使用高斯概率分布的概率密度函数；
 - 根据样本数据和高斯概率分布的函数形式写出似然函数；
 - 将似然函数转化为对数似然函数；
 - 对数似然函数对参数求导且令方程为零；
 - 解方程得到最优的参数值。

- 上述的内容介绍了如何使用最大似然估计求解一个样本数据集的概率分布函数的方法。
- 那么对于一个样本数据集，如果我们使用高斯概率分布来拟合这个数据集，那么求解高斯分布模型参数的步骤也是一样的。
- 详细的求解步骤：
 - 使用高斯概率分布的概率密度函数
 - 根据样本数据和高斯概率分布的函数形式写出似然函数
 - 将似然函数转化为对数似然函数
 - 对数似然函数对参数求导且令方程为零
 - 解方程得到最优的参数值
- 如此以来就完成了整个参数估计的过程。

高斯混合模型

- GMM是单一高斯概率密度函数的延伸，GMM能够平滑地近似任意形状的概率分布。高斯混合模型种类有单高斯模型（Single Gaussian Model, SGM）和高斯混合模型（Gaussian Mixture Model, GMM）两类。
- 类似于聚类，根据高斯概率密度函数（Probability Density Function, PDF）参数不同，每一个高斯模型可以看作一种类别，输入一个样本 x ，即可通过PDF计算其值，然后通过一个阈值来判断该样本是否属于高斯模型。很明显，SGM适合于仅有两类别问题的划分，而GMM由于具有多个模型，划分更为精细，适用于多类别的划分，可以应用于复杂对象建模。

- 一般来说，一个混合模型可以使用任何概率分布，这里使用高斯混合模型是因为高斯分布具备很好的数学性质以及良好的计算性能。

高斯混合模型数学定义

- 高斯混合模型的概率分布:

$$P(x|\theta) = \sum_{k=1}^K \alpha_k \phi(x|\theta_k)$$

- 对于这个模型而言, 参数 $\theta = (\widetilde{\mu}_k, \widetilde{\sigma}_k, \widetilde{\alpha}_k)$ 也就是每个子模型的期望、方差 (或协方差)、在混合模型中发生的概率。
- 参数解释:
 - x_j 表示第 j 个观测数据, $j = 1, 2, 3, \dots, N$; K 是混合模型中高斯模型的数量; α_k 是观测数据属于第 k 个子模型的概率, $\alpha_k \geq 0, \sum_{k=1}^K \alpha_k = 1$; $\phi(x|\theta_k)$ 是第 k 个子模型的高斯分布密度函数, $\theta_k = (\mu_k, \sigma_k^2)$;

高斯混合模型参数学习

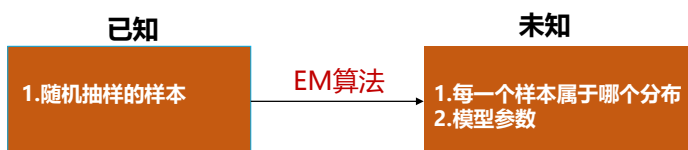
- 对于**高斯混合模型**，Log-Likelihood 函数是：

$$\log L(\theta) = \sum_{j=1}^N \log P(x_j | \theta) = \sum_{j=1}^N \log \left(\sum_{k=1}^K \alpha_k \phi(x | \theta_k) \right)$$

- 如何计算高斯混合模型的参数呢？这里我们无法像单高斯模型那样使用最大似然法来求导求得使 likelihood 最大的参数，因为对于每个观测数据点来说，事先并不知道它是属于哪个子分布的（Hidden Variable），因此 log 里面还有求和， K 个高斯模型的和不是一个高斯模型，对于每个子模型都有未知的 α_k ， μ_k ， σ_k ，直接求导无法计算。需要通过**迭代的方法求解**。

EM算法

- **最大期望算法** (Expectation Maximization Algorithm) , 是一种迭代算法, 用于含有隐变量 (Hidden Variable) 的概率参数模型的最大似然估计或极大后验概率估计。
- EM算法是Dempster, Laird, Rubin于1977年提出的求极大似然估计参数的一种方法, 它可以从非完整数据集中对参数进行MLE估计, 是一种非常简单实用的学习算法。这种方法可以广泛地应用于处理缺损数据, 截尾数据, 带有噪声等所谓的不完全数据。



EM算法步骤 (1)

- **初始化参数：**
- **E步骤：**依据当前参数，计算每个数据 j 来自于子模型 k 的可能性：

$$\gamma_{jk} = \frac{\alpha_k \phi(x_j | \theta_k)}{\sum_{k=1}^K \alpha_k \phi(x_j | \theta_k)}$$

▫ 其中, $j = 1, 2, 3, \dots, N$; $k = 1, 2, 3, \dots, K$

- **M步骤：**计算新一轮迭代的模型参数：

$$\mu_k = \frac{\sum_j^N (\gamma_{jk} x_j)}{\sum_j^N \gamma_{jk}}, \text{ 其中 } k = 1, 2, 3, \dots, K$$

- EM算法是一种迭代算法，每一次迭代会对上一轮的参数进行更新。
- 首先初始化各种参数，包括高斯混合模型的混合个数 K ，每个单高斯模型的参数 $\theta = (\mu_k, \sigma_k)$ ，还观测数据属于单高斯模型的概率有 α_k 。
- 然后是E步骤，是求期望的过程，根据目前已知的参数，来更新每个数据 j 来自于子模型 k 的可能性，使用公式 $\gamma_{jk} = \frac{\alpha_k \phi(x_j | \theta_k)}{\sum_{k=1}^K \alpha_k \phi(x_j | \theta_k)}$ ，这里的可能性由两部分构成：首先是整体数据点来自单高斯模型的概率，然后是假如数据点 j 来自单高斯模型 k ，它出现的概率，整体来看就是一个 $P(A|B)P(B)$ 的关系。
- 然后根据得到的 γ_{jk} ，可以重新估算参数。

▫ $\mu_k = \frac{\sum_j^N (\gamma_{jk} x_j)}{\sum_j^N \gamma_{jk}}$ ，单高斯模型的均值。

EM算法步骤 (2)

- **M步骤**: 计算新一轮迭代的模型参数:

$$\Sigma_k = \frac{\sum_j^N \gamma_{jk} (x_j - \mu_k)(x_j - \mu_k)^T}{\sum_j^K \gamma_{jk}} \quad (\text{用这一轮更新后的 } \mu_k)$$

$$\alpha_k = \frac{\sum_{j=1}^N \gamma_{jk}}{N}$$

▫ 其中 $k = 1, 2, 3, \dots, N$;

- **迭代**: 重复计算 E-step 和 M-step 直至收敛。

- 然后是单高斯模型的标准差 $\Sigma_k = \frac{\sum_j^N \gamma_{jk} (x_j - \mu_k)(x_j - \mu_k)^T}{\sum_j^K \gamma_{jk}}$
- 最后是数据来自单高斯模型的概率 $\alpha_k = \frac{\sum_{j=1}^N \gamma_{jk}}{N}$
- 得到更新的参数后, 进入下一次的循环; 直到满足循环终止条件。
- 至此, 我们就找到了GMM的参数学习。需注意的是, EM 算法具备收敛性, 但并不保证找到全局最大值, 有可能找到局部最大值。解决方法是初始化几次不同的参数进行迭代, 取结果最好的那次。

EM算法步骤 (3)

- 整个EM算法可以归结为以下几个步骤：
 - 初始化参数；
 - 求解期望，也就是每个样本点属于每个单高斯模型的概率；
 - 最大化过程，根据期望过程得到的结果，更新参数；
 - 迭代上述过程直至收敛。

GMM优缺点

- **优点:**
 - 拟合能力强
 - 对语音特征匹配概率最大化
- **缺点:**
 - 无法处理序列因素
 - 无法处理线性或近似线性数据

目录

1. 语音处理介绍
2. 语音识别
3. 语音合成
4. **传统声学模型GMM-HMM**
 - GMM
 - HMM
 - GMM-HMM
5. 混合模型DNN-HMM
6. 高级语音模型

马尔可夫链案例

- 某同类商品A，B，C的宣传力度不同，顾客在广告宣传的效应下第一次尝试选择购买商品A，B，C的概率分别是0.2，0.4，0.4。顾客的购买倾向为下表，求某顾客第4次购买各商品的概率。

	第二次购买			
首次购买		A	B	C
	A	0.8	0.1	0.1
	B	0.5	0.1	0.4
	C	0.5	0.3	0.2



马尔可夫链案例求解

- 三要素：

- 初始概率： $\pi = (0.2, 0.4, 0.4)$
- 转移概率： $p_{AA} = 0.8, p_{AB} = 0.1, p_{AC} = 0.1, p_{BA} = 0.5, \dots$
- 转移概率矩阵： $A = \begin{pmatrix} 0.8 & 0.1 & 0.1 \\ 0.5 & 0.1 & 0.4 \\ 0.5 & 0.3 & 0.2 \end{pmatrix}$

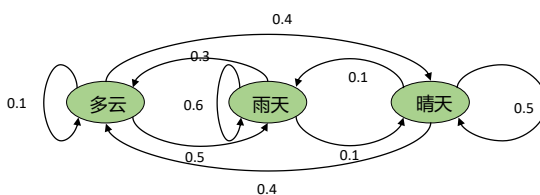
- 求解：

- 马尔可夫链模型：
- $P(X_{n+1} = x | X_1 = x_1, X_2 = x_2, \dots, X_n = x_n) = P(X_{n+1} = x | X_n = x_n)$
- 第三次概率： $P_{AAA} = ?$

- $P_{AAA} = 0.2 * 0.8 * 0.8$

马尔可夫链 (Markov chain)

- **马尔可夫链**，是指数学中具有马尔可夫性质的离散事件随机过程。该过程中，在给当前知识或信息的情况下，过去对于预测将来是无关的，只与当前状态有关。
- 在马尔可夫链的每一步，系统根据概率分布，可以从一个状态变到另一个状态，也可以保持当前状态。状态的改变叫做**转移**，与不同的状态改变相关的概率叫做**转移概率**。



马尔可夫链原理

- **原理：**

- 马尔可夫链描述了一种状态序列，其每个状态值取决于前面有限个状态。马尔可夫链是具有马尔可夫性质的随机变量的一个数列。这些变量的范围，即它们所有可能取值的集合，被称为“**状态空间**”。

- **性质：**

- **正定性：**状态转移矩阵中的每一个元素被称为**状态转移概率**，由概率论知识可知，每个状态转移概率皆为正数，公式可表示为： $p_{ij}(k) \geq 0$ 。
- **有限性：**根据概率论知识，状态转移矩阵中的每一行相加皆为1，用公式可表示为：

$$\sum p_{ij} = 1$$

可观测马尔可夫模型

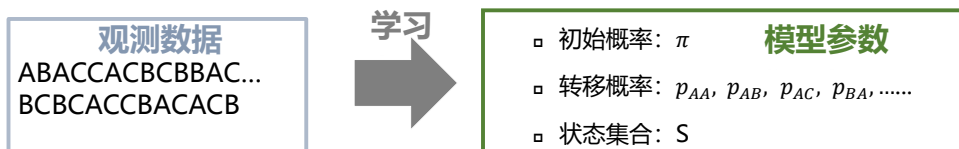
- 对一个问题而言，我们有初始分布 π ，转移概率矩阵 A ，在给定的任意一个时刻 t ，我们都有一个状态 o_t ；一个状态随着时间的变化转移到另一个状态，由此便能得到一个观测序列，即为状态序列 $O = [o_1, o_2, o_3, o_4, \dots, o_n]$ 。且整个问题中总共有 n 个观察状态。
- 出现这样的序列的概率为：

$$P(O|A, \pi) = P(o_1) \prod_{t=2}^m P(o_t|o_{t-1})$$

- 所以一个可观测的马尔可夫模型由一个三元组 (A, π, n) 表示,一般情况下可以简写为 (A, π) 。

马尔可夫链学习问题

- 马尔可夫模型的学习
 - 马尔可夫模型学习问题：没有人会事先告诉我们某个现象是参数为多少的马尔可夫模型，如果我们还是想对这个随机过程进行研究时，这就需要学习过程了，通过能拿到的数据，学习模型的参数。
 - 马尔可夫模型学习问题是指在给定了一系列的观测数据之后，学习马尔可夫模型的参数。
 - 学习的内容包括初始概率 π ，转移概率矩阵 A ，其中状态集合是在问题研究时就确定好，不需要额外的学习过程。



马尔可夫链学习算法 - 穷举法

- 穷举法：在马尔可夫模型的学习中，我们采用的是穷举法，也就是概率逼近法，当能拿到的样本序列数据越多，得到的参数的准确性越高。
 - 初始概率： $\pi_i =$ 以状态*i*开始的序列的数目/序列总数
 - 转移概率： $p_{ij} =$ 从状态*i*转移到状态*j*的转移的数目/从状态*i*出发的转移总数
- 举个例子，假设有数据：[红，红，红] [红，红，蓝] [红，蓝，红] [蓝，红，红]，通过穷举法，可以得到初始概率和转移概率分别是：
 - 初始概率： $\pi=\{0.75,0.25\}$
 - 转移概率： $p_{\text{红蓝}} = 1/3$ ， $p_{\text{红红}} = 2/3$ ， $p_{\text{蓝蓝}} = 0$ ， $p_{\text{蓝红}} = 1$

马尔可夫链预测问题

- 马尔可夫模型的预测：
 - 马尔可夫的预测问题是指在给定了马尔可夫模型参数初始概率： π ；转移概率： $p_{AA}, p_{AB}, p_{BA}, \dots$ ；状态集合： S 的前提下，求解某个观测序列出现的概率值。
- 应用：通过马尔可夫模型的预测，可以预判一个序列发生的可能性，同时也可以针对一个已发生的序列去判断是哪个马尔可夫模型产生的。

模型参数

- 初始概率： π
- 转移概率： $p_{AA}, p_{AB}, p_{AC}, p_{BA}, \dots$
- 状态集合： S

预测



观测序列概率

P (AABCCBCAC...)
P (CBACBCCBBA...)
P (ABACCACBCBBAC...)
.....

马尔可夫链预测算法

- 给定一个马尔可夫模型，初始概率 π ；转移概率 $p_{AA}, p_{AB}, p_{AC}, p_{BA}, \dots$ ；状态集合 S 都是已知的。求解如下所示序列出现的概率。

$$P(\text{AABCCBCAC} \dots) = P(A)P(A|A)P(B|A)P(C|B) \dots$$

- 根据马尔可夫特性可以知道， $n-1$ 时刻状态出现的概率只和 n 时刻的状态有关，因此上述序列的概率计算可以进行链式拆解如等式右边所示。
- 可以看到等式右边的乘项是有初始概率和转移概率组成的，这些概率值都是已知的，因此可以很方便的计算出序列出现的概率。

隐马尔可夫模型案例引入 (1)

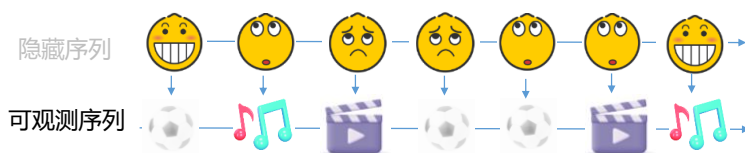
- 隐马尔可夫模型 (Hidden Markova Model) :



- 假设小明状态如下：
 - 如左上图所示，小明每天的心情是“开心”、“不开心”、“一般”这三种心情状态中的一种；
 - 如右上图所示，小明每天的娱乐活动是“踢足球”、“听音乐”、“开电视”中的一种；
 - 小明的娱乐活动是受到当天的心情影响，三种心情下娱乐活动选择的概率也不一样；比如“开心”的时候选择看电视的概率很大，“不开心”时选择踢足球的概率更大；
 - 小明的每天的心情变化遵循一定的规律，第二天的心情状态只受前一天的心情状态影响。

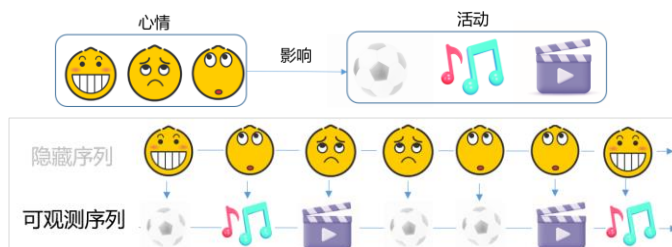
隐马尔可夫模型案例引入 (2)

- 当前情况下，小明在一段时间内的心情序列和娱乐活动序列用下面的示意图表示，可以看到有两个序列存在：
 - 其中心情状态是一个序列，心情序列的状态在一定规律下产生；
 - 娱乐活动也是一个序列，娱乐活动序列受心情序列的影响。
- 小明的每天的心情状态是隐藏的（心情的好坏不可以直观观测），娱乐活动可以直接观测。因此通常将：
 - 无法直接观测到的状态称为隐藏状态。（小明的心情）
 - 可以直接观测到的状态称为可观测状态。（小明的娱乐活动）



隐马尔可夫模型案例引入 (3)

- 假如想要对如下问题进行深入的研究：



- 有两个状态，一个是隐藏状态，一个是可观测状态；
 - 当前隐藏状态决定了当前可观测状态的分布概率；
 - 下一个时间步的隐藏状态分布只受当前隐藏状态的影响。
- 对这个双重随机过程进行数学建模：隐马尔可夫模型。

隐马尔可夫模型 (1)

- 隐马尔可夫模型实际上是标准马尔可夫模型的扩展，添加了可观测状态集合和这些状态与隐含状态之间的概率关系。
- 类似于马尔可夫模型有3个模型要素，隐马尔可夫模型有5个要素，这5个要素一旦确定，隐马尔可夫模型也就唯一确定了。



隐马尔可夫模型 (2)

- HMM可以用5个元素来描述：
 - 隐藏状态集合： $R = \{R_1, R_2, R_3, \dots, R_m\}$ ；如小明的心情状态，开心、不开心、一般。
 - 观测状态集合： $S = \{S_1, S_2, S_3, \dots, S_n\}$ ；如小明的娱乐活动集合，踢足球、听音乐、看电视。
 - 初始概率： $\pi = \{P_{ini}(S_i) | \forall i \in [1, n]\}$ ；如小明的初始心情，这个一般都是作为随机序列起始时的一个假设值，例如小明每个月开始的时候心情状态从0开始，不受上个月影响。
 - 转移概率： $A = \{P_{trans}(S_i | S_j) | \forall i, j \in [1, n]\}$ ；如小明心情之间的转移关系，今天的心情是如何影响明天的心情的。
 - 发射概率： $B = \{P_{obv}(R_k | S_j) | \forall j \in [1, n], k \in [1, m]\}$ ；如小明的当天的心情和娱乐活动之间的关系，心情是怎样影响娱乐活动的。
- 根据这5个要素，HMM可以由一个五元组描述 $\lambda = (A, B, \pi, R, S)$ ，也可简写为 $\lambda = (A, B, \pi)$ 。

HMM三个主要问题

- 评价问题：
 - 前向算法
 - 后向算法
- 学习问题
 - 监督式算法
 - 非监督式Baum-Welch算法
- 解码问题
 - 动态规划算法
 - Viterbi算法

隐马尔可夫模型评价问题 (1)

- 隐马尔可夫模型评价问题是指：
 - 给定了隐马尔可夫模型，包括：
 - 初始概率： π
 - 转移概率： A
 - 发射概率： B
 - 隐藏状态集合： $S = \{S_1, S_2, S_3, \dots, S_n\}$
 - 可观测状态集合： $R = \{R_1, R_2, R_3, \dots, R_m\}$
 - 其中一般默认隐藏状态集合和可观测状态集合在一个问题研究中始终是已知的，很多情况下不会特殊说明它的已知性
 - 给定观测序列；
 - 目的是：求解可观测序列出现的概率。

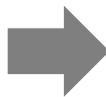
隐马尔可夫模型评价问题 (2)

- 举例：求解小明在一周内娱乐活动是如下所示序列的概率？
 - 看电视 -> 踢足球 -> 踢足球 -> 听音乐 -> 看电视 -> 踢足球 -> 看电视
 - 踢足球 -> 踢足球 -> 听音乐 -> 看电视 -> 听音乐 -> 踢足球 -> 听音乐

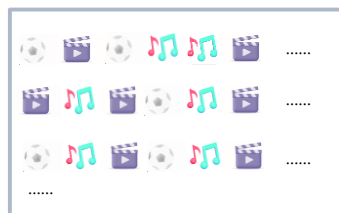
已知模型参数

- 初始概率： π
- 转移概率： A
- 发射概率： B

预测



可观测数据序列概率

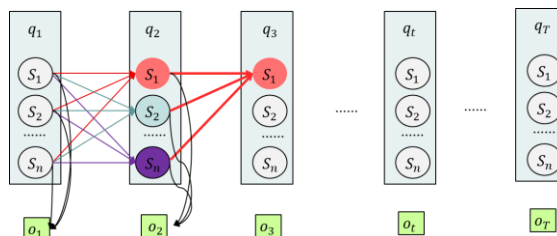


隐马尔可夫模型预测算法 - 前向后向算法 (1)

- 假设产生观测序列的隐藏状态序列为 Q :
 - $Q = [q_1, q_2, q_3, q_4, \dots, q_T]$, 其中 T 表示序列长度; q_i 表示序列第 i 个位置的观测状态, $q_i \in S$
- 前向&后向算法的思想比较类似, 以前向算法为例进行介绍。前向算法对于穷举法的核心改进是对重复计算的部分进行复用:
 - 假设 $O_t = [o_1, o_2, o_3, o_4, \dots, o_t]$
 - $P(O_t, q_t = S_i)$ 表示第 t 个隐藏状态是 S_i 且产生 O_t 的概率
 - $P(O_t, q_t = S_i)$ 有如下特性:
 - $P(O_{t+1}, q_{t+1} = S_i) = \left[\sum_{j=1}^n P(O_t, q_t = S_j) * P_{trans}(S_i | S_j) \right] * P_{obv}(O_{t+1} | S_i)$
 - $P(O) = P(O_T) = \sum_{i=1}^n P(O_T, q_T = S_i)$

隐马尔可夫模型预测算法 - 前向后向算法 (2)

- 前向算法计算过程：
 - 遍历序列点 $t \in [1, T]$
 - $t=1$ 时, $P(O_1, q_1 = S_i) = P_{ini}(S_i) * P_{obv}(o_1 | S_i)$
 - $t > 1$ 时对于每个隐藏状态 S_i , 计算 $P(O_t, q_t = S_i) = \left[\sum_{j=1}^n P(O_{t-1}, q_{t-1} = S_j) * P_{trans}(S_i | S_j) \right] * P_{obv}(o_t | S_i)$
 - 得到最终 $P(O) = P(O_T) = \sum_{i=1}^n P(O_T, q_T = S_i)$
 - 算法复杂度: $O(Tn^2)$



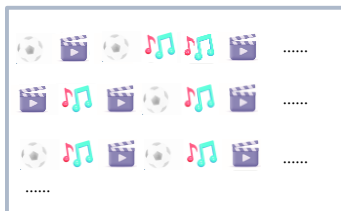
隐马尔可夫模型学习问题 (1)

- 学习问题是隐马尔可夫模型另外两个问题的基础，在预测和解码问题中隐马尔可夫模型都是给定的。但是在实际中，我们要研究一个原始的问题时，比如小明的心情和娱乐活动这个问题时，隐马尔可夫模型的模型参数是未知的，需要从能观察到的数据中学习这些模型参数。
- 隐马尔可夫模型的学习问题是指：
 - 已知大量的可观测的状态序列 / 已知大量的可观测的状态序列和对应的隐藏状态序列
 - 目的：求解隐马尔可夫模型的参数
 - 初始概率： π
 - 转移概率： A
 - 发射概率： B

隐马尔可夫模型学习问题 (2)

- 举例：已知小明一周的娱乐活动序列，求解满足小明心情和娱乐活动这一问题的隐马尔可夫模型。

可观测数据序列



学习



模型参数

- 初始概率: π
- 转移概率: A
- 发射概率: B

隐马尔可夫模型学习算法 - 监督算法

- 适用场景：已知大量的可观测的状态序列和对应的隐藏状态序列样本
- 思想：直接利用大数定理的结论“频率的极限是概率”，直接给出HMM的参数估计

- 初始概率 π : $P_{ini}(S_i) = \frac{|Seq_begin_with_S_i|}{|Seq|}$

- $Seq_begin_with_S_i$ 表示样本中以 S_i 开头的隐藏状态序列
- Seq 表示样本中所有隐藏状态序列

- 转移概率 A : $P_{trans}(S_i | S_j) = \frac{|trans(S_j, S_i)|}{|trans(S_j)|}$

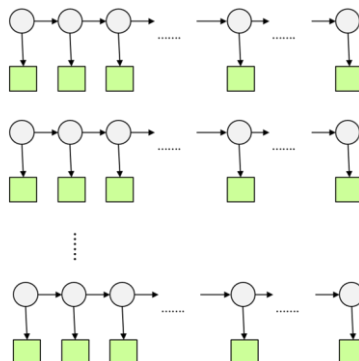
- $trans(S_j)$ 表示样本中隐藏状态序列中所有由 S_j 出发的状态转移对
- $trans(S_j, S_i)$ 表示样本中隐藏状态序列中所有由 S_j 出发转移到 S_i 的状态转移对

- 发射概率 B : $P_{obv}(R_k | S_j) = \frac{|emiss(S_j, R_k)|}{|emiss(S_j)|}$

- $emiss(S_j, R_k)$ 表示样本中由隐藏状态 S_j 产生的观测状态 R_k 的发射关系对
- $emiss(S_j)$ 表示样本中由隐藏状态 S_j 产生的所有可能观测状态的发射关系对

隐藏状态序列

观测状态序列



隐马尔可夫模型学习算法 - Baum-Welch

- 观察序列，没有给出状态序列，求HMM。
- 该算法的本质**就是EM算法**，它解决的问题是：有了观测值 X ，而观测值有个隐变量 Z 时，求在HMM参数 λ 下的联合概率 $P(X, Z|\lambda)$ 。
- **求解步骤：**
 - 确定完全数据的对数似然函数
 - EM算法E步：求 Q 函数 $Q(\lambda, \hat{\lambda})$
 - EM算法M步：极大化 Q 函数求参数

- <https://blog.csdn.net/u014688145/article/details/53046765?locationNum=7&fps=1>
- <https://blog.csdn.net/u012771351/article/details/53114829>

隐马尔可夫模型解码问题

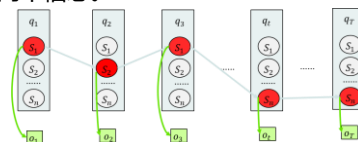
- 解码问题是指：
 - 给定了隐马尔可夫模型，包括：
 - 初始概率： π
 - 转移概率： A
 - 发射概率： B
 - 隐藏状态集合： $S = \{S_1, S_2, S_3, \dots, S_n\}$
 - 可观测状态集合： $R = \{R_1, R_2, R_3, \dots, R_m\}$
 - 其中一般默认隐藏状态集合和可观测状态集合在一个问题研究中始终是已知的，很多情况下不会特殊说明它的已知性
 - 给定观测序列；
 - 目的是：求解产生给定观测序列的最有可能的隐藏状态序列。

隐马尔可夫模型解码算法 - 穷举法

- 假设产生观测序列的可能隐藏状态序列为 Q
 - $Q = [q_1, q_2, q_3, q_4, \dots, q_T]$, 其中 T 表示序列长度; q_i 表示序列第 i 个位置的观测状态, $q_i \in S$
- 穷举法的思想比较简单:
 - 列举所有隐藏状态序列 Q 的可能性
 - 计算每个隐藏状态序列产生观测状态序列的概率 $P(O, Q)$
 - $Q_{max} = \operatorname{argmax}_Q P(O, Q)$
- 穷举法缺点:
 - 算法复杂度高

隐马尔可夫模型解码算法 - Viterbi (1)

- 产生观测序列 $O = [o_1, o_2, o_3, o_4, \dots, o_T]$ 的 $Q_{max} = [q_1, q_2, q_3, q_4, \dots, q_T]$ 。产生观测序列 $O_t = [o_1, o_2, o_3, o_4, \dots, o_t]$ 的 $Q_{max_t} = [q_1, q_2, q_3, q_4, \dots, q_t]$ 。
 - O_t 是 O 的前 t 个元素的子序列； Q_{max_t} 是 Q_{max} 的前 t 个元素的子序列。
- 思想：
 - 前向遍历：对于每个时间点的序列中的每个隐藏状态，记录两个信息。
 - 信息1：到达该状态的最大概率路径的前一个时刻的隐藏状态；
 - 信息2：到达该状态产生观测序列的最大概率。
 - 后向回溯：
 - 对于每一个时刻，根据信息1的记录提取产生观测状态的最大概率的隐藏状态序列。



隐马尔可夫模型解码算法 - Viterbi (2)

- 假设产生观测序列的可能隐藏状态序列为 Q :
 - $Q = [q_1, q_2, q_3, q_4, \dots, q_T]$, 其中 T 表示序列长度; q_i 表示序列第 i 个位置的观测状态, $q_i \in S$
- 定义如下:
 - 假设 $O_t = [o_1, o_2, o_3, o_4, \dots, o_t]$
 - $P(O_t, q_t = S_i)$ 表示第 t 个隐藏状态是 S_i 且产生 O_t 的最大概率
 - $P(O_t, q_t = S_i)$ 有如下特性:
 - 产生最大概率 $P(O_t, q_t = S_i)$ 的 $t - 1$ 时刻的隐藏状态 $Pre_S(t, S_i)$:
$$- Pre_S(t, S_i) = \operatorname{argmax}_{S_j} (P(O_{t-1}, q_{t-1} = S_j) * P_{trans}(S_i | S_j))$$
 - 计算 $P(O_t, q_t = S_i) = P(O_{t-1}, q_{t-1} = S_j) * P_{trans}(Pre_S(t, S_i) = S_j) * P_{obv}(o_{t-1} | S_i)$

隐马尔可夫模型解码算法 - Viterbi (3)

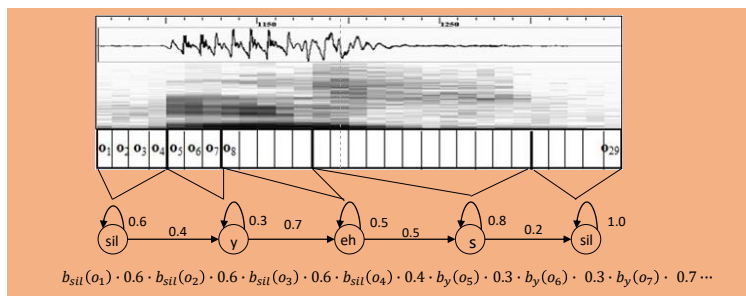
- Viterbi算法的思想也比较简单，分为前向计算和后向回溯两个阶段：
 - 前向计算，遍历序列点 $t \in [1, T]$:
 - $t=1$ 时, $P(O_1, q_1 = S_i) = P_{ini}(S_i) * P_{obv}(o_1 | S_i)$
 - $t > 1$ 时对于每个隐藏状态 S_i
 - 记录 $Pre_S(t, S_i) = \operatorname{argmax}_{S_j} (P(O_{t-1}, q_{t-1} = S_j) * P_{trans}(S_i | S_j))$
 - 计算 $P(O_t, q_t = S_i) = P(O_{t-1}, q_{t-1} = S_j) * P_{trans}(Pre_S(t, S_i) = S_j) * P_{obv}(o_{t-1} | S_i)$
 - 后向回溯, $Q_{max} = [q_1, q_2, q_3, q_4, \dots, q_T]$:
 - 遍历序列点 $t \in [T, 1]$:
 - $t = T$ 时, $q_t = \operatorname{argmax}_{S_j} (P(O_T, q_T = S_j))$
 - $t < T$ 时, $q_t = Pre_S(t, q_{t+1})$

目录

1. 语音处理介绍
2. 语音识别
3. 语音合成
- 4. 传统声学模型GMM-HMM**
 - GMM
 - HMM
 - GMM-HMM
5. 混合模型DNN-HMM
6. 高级语音模型

GMM - HMM语音识别过程

- 获得语音wav文件， 以下是一个词识别全过程：
 - 将waveform切成等长frames， 对每个frame提取特征（MFCC）。
 - 对每个frame的特征运行GMM， 得到每个frame属于每个状态的概率state。
 - 根据每个单词的HMM状态转移概率a计算每个状态sequence生成该frame的概率。
 - 哪个词的HMM序列概率最大， 就判断这段语音属于该词。



GMM - HMM单字识别 (1)

- 识别中文数字0-9的原理和流程。
- 数字0的表现形式：
 - 拼音形式，如右图左边所示，声母+韵母（带声调）；那么数字0可以表示为：ling(2)；
 - 发音形式，如右图右边所示，对于不同的人，在不同的情况下，根据声母韵母的形式去发音数字0，可能会出现很多种情况，比如拖长音，前重后轻等等，假设把音频切成等份的小段，每一小段是声母或者韵母发音，那么数字0的发音序列可以存在情况有多种，举例如下，其中si 表示静音这种状态：

数字0的拼音	数字0的音频文件	数字0的发音序列
ling(2) 0		Si ling(2) Si Si Si Si ling(2) ing(2) ing(2) ing(2) ing(2) Si Si ling(2) ing(2) ing(2) Si Si Si

- 通常情况下是要保证声母韵母的出现顺序的，假设有个声母的音频片段出现在韵母之后这种情况是不太常见的；并且不同的音频对应的序列长度也是不一样的。

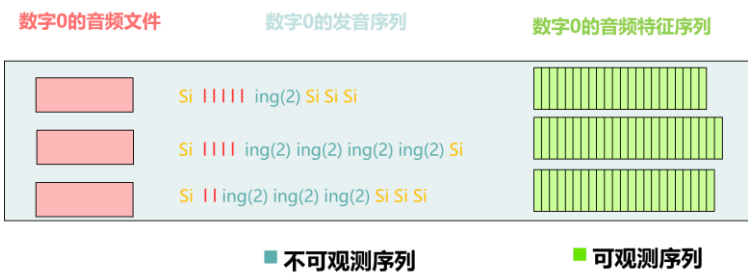
GMM - HMM单字识别 (2)

- 音频特征序列形式，如上图所示将每一个音频文件进行去噪+分帧+MFCC特征提取处理得到的音频特征序列，序列中每个元素可以是我们之前介绍的梅尔倒谱系数得到的一个多维向量。其中序列中的每个元素也是由声母或者韵母的发音特征产生的。



GMM - HMM单字识别 (3)

- 总结了数字0的三种形式：
 - 拼音形式；
 - 发音序列形式；
 - 音频特征序列形式。



GMM - HMM单字识别 (4)

- 对于一个数字0的音频文件：

- 对应了一个数字0的发音序列（声母，韵母序列）；
- 对应了一个音频特征序列。

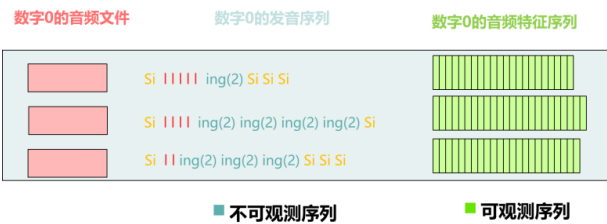


- 序列特征：

- 其中发音序列是不可见的；
- 音频特征序列是可见的，是对音频进行数字化处理之后的结果；
- 其中发音序列之间是有规律存在的：比如l 之后可能会出现si ing(2) l ; ing(2)之后可能会出现si ing(2) ；
- 发音状态影响音频特征状态，si 和 ing(2) 产生的 音频特征差异很大。
 - 由于同一个发音中的可变因素太多（音色、地域、年龄、场合）等等，同一个si产生的音频序列也会存在一些差异，但是本质不变。

GMM - HMM单字识别 (5)

- 隐马尔可夫模型使用：
 - 数字0的拼音形式表示了隐藏状态集合；
 - 发音序列是隐藏状态序列；
 - 音频特征序列是可观测状态序列；
 - 隐藏状态之间是有影响关系的，也就是声母韵母之间的转移概率；
 - 隐藏状态和观测状态之间有影响关系，也就是声母韵母对观测状态的发射概率。



GMM - HMM单字识别 (6)

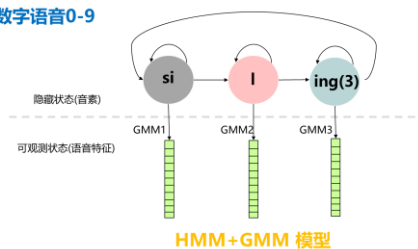
- 前面学习的HMM的观测状态是离散状态，只要每个隐藏状态对应每个观测状态的概率分布即可，是一个离散概率分布，但是这里的可观测状态是一个连续的向量，这要怎么解决啊？这种连续的概率分布要怎么表示？这里就用到了前面学到的GMM的知识，可以使用GMM去拟合每一个隐藏状态到观测状态的一个多维连续概率分布。

GMM - HMM单字识别 (6)

- GMM-HMM模型的五要素分别是:

- 观测状态集合: $R = \{GMM_{si} \text{产生的多维向量}, GMM_l \text{产生的多维向量}, GMM_{ing(3)} \text{产生的多维向量}\}$;
- 隐藏状态集合: $S = \{si, l, ing(3)\}$; 声母、韵母、静默音;
- 初始概率: $\pi = \{P\{q_1 = si\}, P\{q_1 = l\}, \dots, P\{q_1 = ing(2)\}\}$; 数字0的合理音频从这三个音节开始的概率;
- 转移概率: $A = \{P(S_i | S_j) | \forall i, j \in [1, n]\}$; 数字0的合理音频中声母、韵母、静默音之间的转移概率;
- 发射概率: $B = \{P(R_k | S_j) | \forall k \in [1, n], j \in [1, n]\} = b_j(i)$, 记 $B = [GMM_{si}, GMM_l, GMM_{ing(3)}]$; 数字0的三个隐藏状态对应的三个使用GMM表示的连续概率分布。

识别数字语音0-9



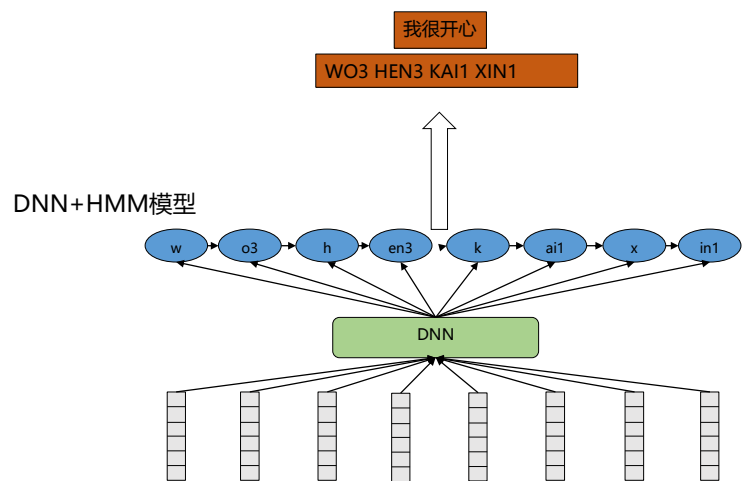
目录

1. 语音处理介绍
2. 语音识别
3. 语音合成
4. 传统声学模型GMM-HMM
- 5. 混合模型DNN-HMM**
6. 高级语音模型

传统语音识别算法的缺点

- GMM-HMM主要的缺点有以下三个：
 - 由于GMM-HMM中，每个隐藏状态都对应了一个GMM模型，特别是当进行连续词识别时，状态数量是很大的，因此模型训练会消耗大量的时间且参数空间很大。
 - 隐马尔可夫模型对语音识别问题进行建模时，有一个隐藏的条件，就是隐藏状态序列之间需要满足马尔可夫特性，也就是说下一个时刻隐藏状态只和当前时刻的隐藏状态有关系，和过去以及未来的隐藏状态都没有关系，这个假设其实在实际中并不是完全合理的，很多时候是前后多个状态决定中间某个状态的。
 - 每个模块是单独进行优化和训练的，模块之间相互耦合关联，这样就导致了不能端到端处理任务，无法根据具体的任务优化整个流程。

语音识别算法 - DNN-HMM

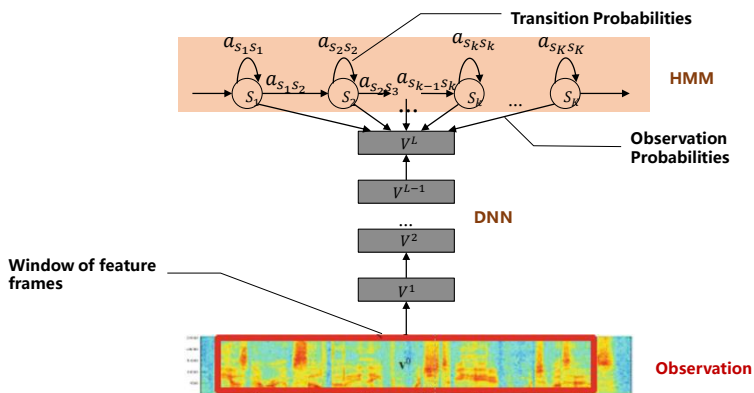


深度学习语音识别算法 - Hybrid模型

- Hybrid模型是指使用深度神经网络代替传统GMM-HMM中的GMM模型，可以使用的代替网络有DNN、CNN、LSTM以及其他的一些变体神经网络结构。
 - 特点：混合模型使用深度神经网络结构代替GMM提取语音信号的特征，对其进行深度加工和表征。
 - 缺点：无法脱离HMM，最终还是需要使用HMM模型；并且需要使用已经训练好的传统GMM-HMM模型进行数据的对齐操作（样本标注），本质上还是需要传统的GMM-HMM模型作为起点。整个的网络架构也只是为了训练声学模型，并不是端到端的处理。
 - 常见的模型结构：基于这一思想出发，构建出的神经网络模型非常多，常见的是：
 - DNN+HMM
 - CNN+HMM
 - LSTM+HMM
 - CLDNN+HMM

DNN - HMM

- 声学信号使用HMM框架建模，每个状态的生成概率使用DNN替换原来的GMM进行估计，DNN每个单元的输出表示状态的后验概率。



- 详细参考文章 Hinton G, Deng L, Yu D, et al. Deep neural networks for acoustic modeling in speech recognition[J]. IEEE Signal processing magazine, 2012, 29.
- DNN+HMM模型中涉及到的很多思想和处理流程是其他混合模型的基础，大家需要好好掌握

DNN - HMM训练过程

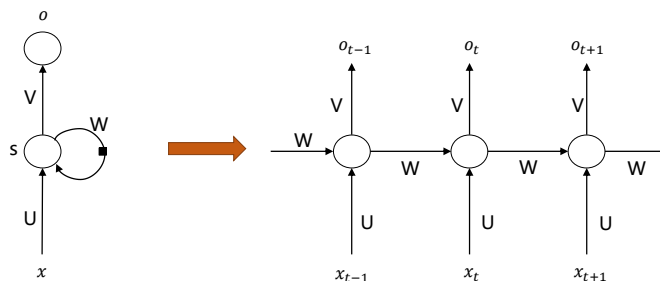
- 首先根据样本数据训练GMM-HMM模型。
- 根据训练好的GMM-HMM模型通过解码算法对输入的音频特征序列推导最有可能的隐藏状态序列。
- 根据第二步中解码得到的特征-隐藏状态的对应关系作为训练样本数据；训练DNN网络，DNN网络的输入是音频特征，输出是属于各个隐藏状态的概率。
- 对于输入的音频序列，使用训练好的DNN和HMM进行解码和搜索过程，得到最终的输出文字序列。

目录

1. 语音处理介绍
2. 语音识别
3. 语音合成
4. 传统声学模型GMM-HMM
5. 混合模型DNN-HMM
- 6. 高级语音模型**
 - RNN系列结构
 - 损失函数
 - 端到端模型

RNN

- 循环神经网络 (Recurrent neural networks, 简称RNN) 是一种通过隐藏层节点周期性的连接, 来捕捉序列化数据中动态信息的神经网络, 可以对序列化的数据进行分类。
 - RNN可以保存一种上下文的状态, 甚至能够在任意长的上下文窗口中存储、学习、表达相关信息, 可以在时间序列上有延拓, 直观上讲, 就是本时间的隐藏层和下一时刻的隐藏层之间的节点间有边。



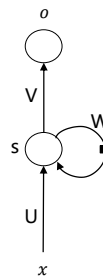
RNN计算过程

- X_t 是输入序列在时间t处的输入
- S_t 是序列在时间t处的记忆单元，缓存了之前的信息：

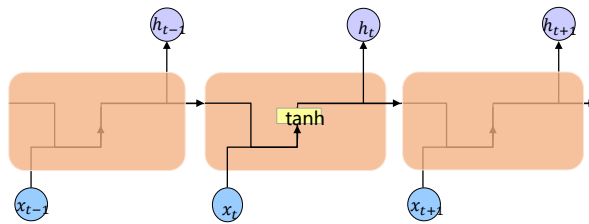
$$S_t = \sigma(UX_t + WS_{t-1})。$$

- S_t 经过多个隐藏层，再通过全连接层V就可以得到该时序t的最终输出 O_t ：

$$O_t = \text{softmax}(S_t V)。$$



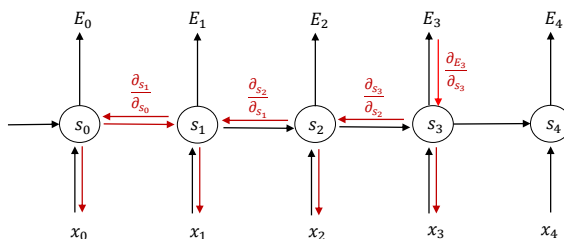
标准RNN



- 所有RNN都具有一种重复神经网络模块的链式的形式。
- 在标准的RNN中，这个重复的模块只有一个非常简单的结构，例如一个tanh层。

BPTT

- 将RNN展开之后，前向传播就是依次按照时间的顺序计算一次，BPTT就是从最后一个时间将累积的残差传递回来即可，这与普通的神经网络训练本质上是相似的，BPTT主要的差异就是将每一时刻的梯度相加。
- 计算关于参数U、V和W的梯度的误差，然后使用梯度下降法学习出好的参数。为了计算这些梯度，需要使用微分的链式法则。



RNN的梯度消失与爆炸

- 反向训练时，对于RNN来说，要横向往前推，一直往前推到序列开始的地方。
- 梯度消失，梯度爆炸都与路径长度太长有关，前面的权重都基本固定不变，没有训练效果。

长短期记忆网络

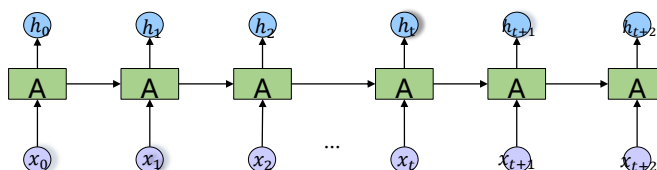
- **长短期记忆网络** (Long Short-Term Memory , LSTM), 是一种时间递归神经网络, 适合于处理和预测时间序列中间隔和延迟相对较长的数据。
- LSTM区别于RNN的地方, 在于它在算法中加入了一个判断信息有用与否的“处理器”, 这个处理器作用的结构被称为cell。
 - 一个cell中被放置了三扇门, 分别叫做**输入门**、**遗忘门**和**输出门**。
 - 信息进入LSTM的网络当中, 可以根据规则来判断是否有用。只有符合算法认证的信息才会留下, 不符的信息则通过遗忘门被遗忘。

LSTM与语音识别

- 近几年，在HMM基础上**DNN**的应用大幅度提升了语音识别系统的性能。DNN将每一帧语音及其前后的几帧语音拼接在一起作为网络的输入，从而利用语音序列中**上下文的信息**。DNN中每次输入的帧数是固定的，不同的窗长对最终的识别结果会有影响。
- **RNN**通过递归来挖掘序列中的上下文相关信息，在一定程度上克服了DNN的缺点。但是RNN在训练中很容易出现**梯度消失**的问题，无法记忆**长时信息**。
- **LSTM**通过特定的门控单元使得当前时刻的误差能够保存下来并选择性传给特定的单元，从而避免了梯度消失的问题。
 - LSTM+CTC
 - LSTM+HMM

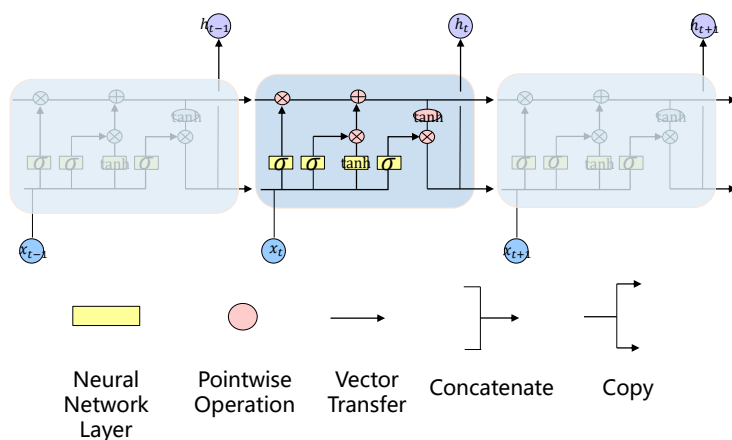
LSTM适用场景

- 相当长的相关信息和位置间隔。
- 适用于需要连接前面长期内的信息到当前的任务上。



- 在理论上，RNN 绝对可以处理这样的 长期依赖 问题。人们可以仔细挑选参数来解决这类问题中的最初级形式，但在实践中，RNN 肯定不能够成功学习到这些知识。Bengio, et al. (1994) 等人对该问题进行了深入的研究，他们发现一些使训练 RNN 变得非常困难的相当根本的原因。

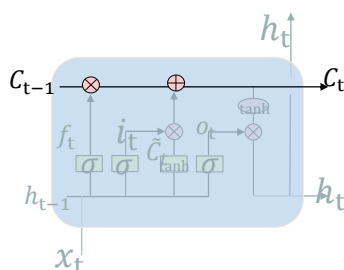
标准LSTM



- 标准的RNN重复的模块值包含单一的层，LSTM 同样是这样的结构，但是重复的模块拥有一个不同的结构。不同于单一神经网络层，这里是有四个，以一种非常特殊的方式进行交互。
- LSTM 中的图标：每一条黑线传输着一整个向量，从一个节点的输出到其他节点的输入。
 - 粉色的圈代表 pointwise 的操作，诸如向量的和，
 - 而黄色的矩阵就是学习到的神经网络层。
 - 合在一起的线表示向量的连接，
 - 分开的线表示内容被复制，然后分发到不同的位置。

LSTM核心知识点：初始状态

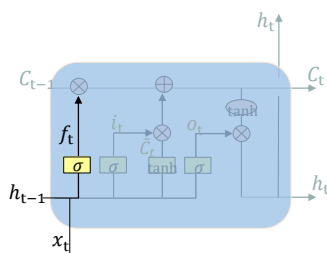
- LSTM的关键就是细胞状态，水平线在图上方贯穿运行。细胞状态类似于传送带。直接在整个链上运行，只有一些少量的线性交互。信息在上面流传保持不变会很容易。



- LSTM 的关键就是细胞状态，水平线在图上方贯穿运行。细胞状态类似于传送带。直接在整个链上运行，只有一些少量的线性交互。信息在上面流传保持不变会很容易。

LSTM核心知识点：遗忘门

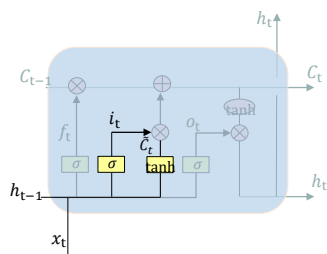
- 在LSTM中的第一步是决定会从细胞状态中丢弃什么信息。
- 这个决定通过一个称为**忘记门**的功能完成。该门会读取 h_{t-1} 和 x_t ，输出一个在 0 到 1 之间的数值给每个在细胞状态 C_{t-1} 中的数字。1 表示 “完全保留” ， 0 表示 “完全舍弃” 。



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

LSTM核心知识点：输入门

- 这一步是确定什么样的新信息被存放在细胞状态中。
- 这里包含两个部分：第一，sigmoid 层称“输入门层”决定什么值将要更新。然后，一个 tanh 层创建一个新的候选值向量，会被加入到状态中。

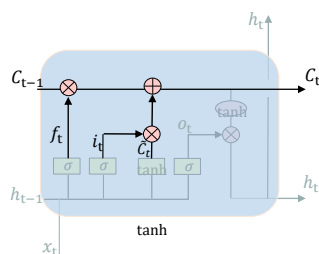


$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \sigma(W_C \cdot [h_{t-1}, x_t] + b_C)$$

LSTM核心知识点：更新信息

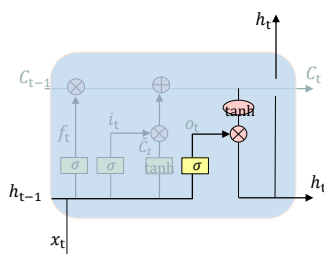
- 更新旧细胞状态， C_{t-1} 更新为 C_t 。
- 把旧状态与 f_t 相乘，丢弃掉确定需要丢弃的信息。接着加上 $i_t * C_t$ 。这就是新的候选值，根据决定更新每个状态的程度进行变化。



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTM核心知识点：输出门

- 首先，运行一个sigmoid层来确定细胞状态的哪个部分将输出出去。接着，把细胞状态通过tanh进行处理（得到一个在-1到1之间的值）并将它和sigmoid门的输出相乘，最终仅仅会输出确定输出的那部分。

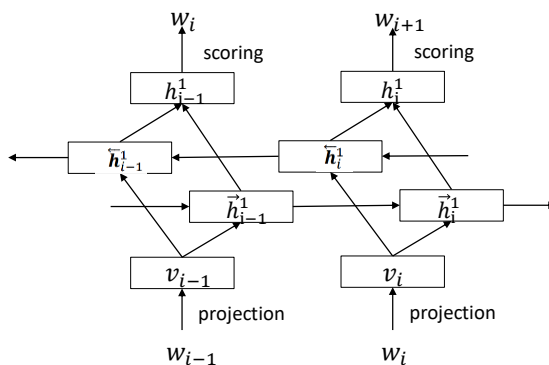


$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

BiRNN

- 双向RNN的基本思想是每一个训练序列向前和向后分别是两个循环神经网络（RNN、LSTM等），而且这两个RNN同时连接着一个输出层。在每一个时刻 t ，输入会同时提供给这两个方向相反的RNN，输出由这两个单向的RNN的状态共同决定：



目录

1. 语音处理介绍
2. 语音识别
3. 语音合成
4. 传统声学模型GMM-HMM
5. 混合模型DNN-HMM
- 6. 高级语音模型**
 - RNN系列结构
 - 损失函数
 - 端到端模型

语音数据对齐 (1)

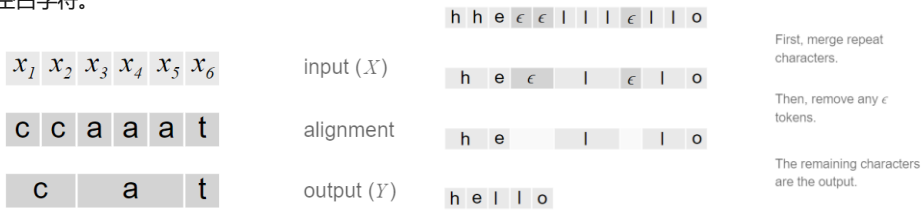
- 在语音数据集中，音频文件很难和输出文本对齐。
- 在传统的语音识别的模型中，对语音模型进行训练之前，往往都要将文本与语音进行严格的对齐操作。这样就有两点不好：
 - 严格对齐要花费人力、时间。
 - 严格对齐之后，模型预测出的label只是局部分类的结果，而无法给出整个序列的输出结果，往往要对预测出的label做一些后处理才可以得到最终想要的结果。
- CTC (Connectionist Temporal Classification) 损失函数建模无对齐信息的多对多序列问题。



<https://distill.pub/2017/ctc/>

语音数据对齐 (2)

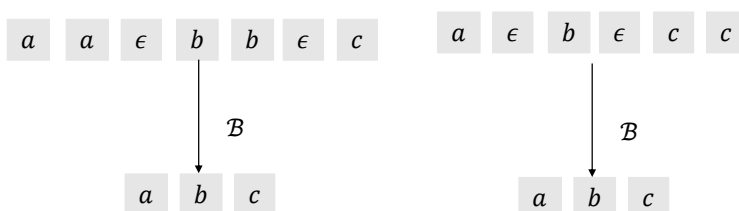
- 如左图所示，在数据对齐时，最简单的解决方案是合并连续重复出现的字母。但是hello这样的单词会被映射成helo。
- CTC损失函数是一种Alignment-free的算法。在数据对齐时，引入空白字符 ϵ 表示语音中的停顿。将 X 通过 B 映射到 Y 。 B 操作包括：
 - 合并连续相同符号；
 - 去除空白字符。



<https://distill.pub/2017/ctc/>

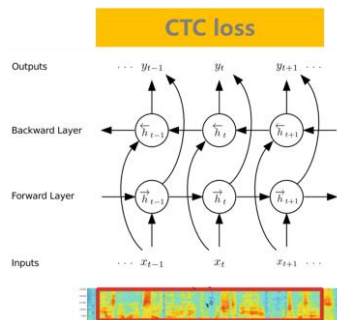
CTC损失函数

- CTC实现变长的隐射。这种对齐方式有以下特征：
 - X、Y之间的时间片映射是单调的，即如果X向前移动一个时间片，Y保持不动或者也向前移动一个时间片；
 - X与Y之间的映射是多对一的，即多个输出可能对应一个映射，反之则不成立。因此X的长度大于等于Y的长度。



RNN+CTC

- 输入的是音频特征序列，输出的是字母序列。
 - 输出的字母序列长度和输入的音频特征序列长度是一样的。
 - 使用这个模型要处理的一个问题就是怎样评估模型输出的字母序列和目标字母序列之间的差异。因为音频特征序列的长度是远远比目标字母序列的长度要长的，也就导致网络输出的字母序列比目标字母序列也是长很多，无法对齐。
 - 这里使用的CTC损失函数评估。
- 长度不一致的输出字母序列和目标字母序列之间的差异。



- 详细参考文章 Graves A, Jaitly N. Towards end-to-end speech recognition with recurrent neural networks[C]//International conference on machine learning. 2014: 1764-1772.
- CTC损失函数就是（Connectionist Temporal Classification）的缩写，详细的计算过程详见论文 Graves A. Supervised sequence labelling with recurrent neural networks. 2012[J]. URL <http://books.google.com/books>, 2012.

RNN+CTC流程

- 使用该模型进行语音识别的整体流程如下：
 - 训练过程使用训练样本数据根据模型结构和CTC损失函数进行模型训练；
 - 预测过程，对于输入的音频特征序列通过网络之后得到相应的字母序列，可以使用beam-search进行最优路径的搜索。

- 详细参考文章 Graves A, Jaitly N. Towards end-to-end speech recognition with recurrent neural networks[C]//International conference on machine learning. 2014: 1764-1772.

目录

1. 语音处理介绍
2. 语音识别
3. 语音合成
4. 传统声学模型GMM-HMM
5. 混合模型DNN-HMM
- 6. 高级语音模型**
 - RNN系列结构
 - 损失函数
 - 端到端模型

End2End模型

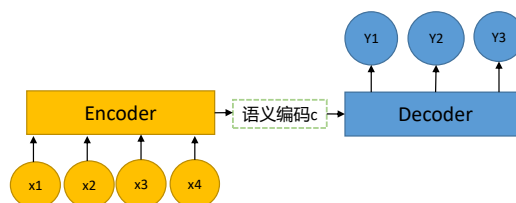
- End2End模型是指输入语音特征序列，输出直接是字母或更粗粒度（字）序列的模型。
 - 优点：端到端处理语音识别问题，不需要大量的中间环节。
 - 需要解决的问题：在没有强制输入输出对齐的标注数据的情况下，怎样进行网络训练。目前有两种方法：
 - 使用RNN系列网络，依赖CTC（Connectionist Temporal Classification）损失函数评估得到的序列和目标序列之间的差异。
 - 使用Seq2Seq网络结构进行自动的对齐，使用交叉熵损失函数评估网络预测的序列和目标序列之间的差异。
 - 常见的网络模型结构有：
 - CNN/RNN/LSTM+CTC
 - Seq2Seq+CE、Seq2Seq+Attention+CE

Seq2Seq

- Seq2Seq (Sequence to Sequence, 序列到序列) 模型, 由编码器 (Encoder) 和解码器 (Decoder) 两部分组成, 一般选用RNN/LSTM/GRU等网络实现。Seq2Seq的输入是一个不定长的序列, 输出也是一个不定长的序列。

- 编码器和解码器的作用:

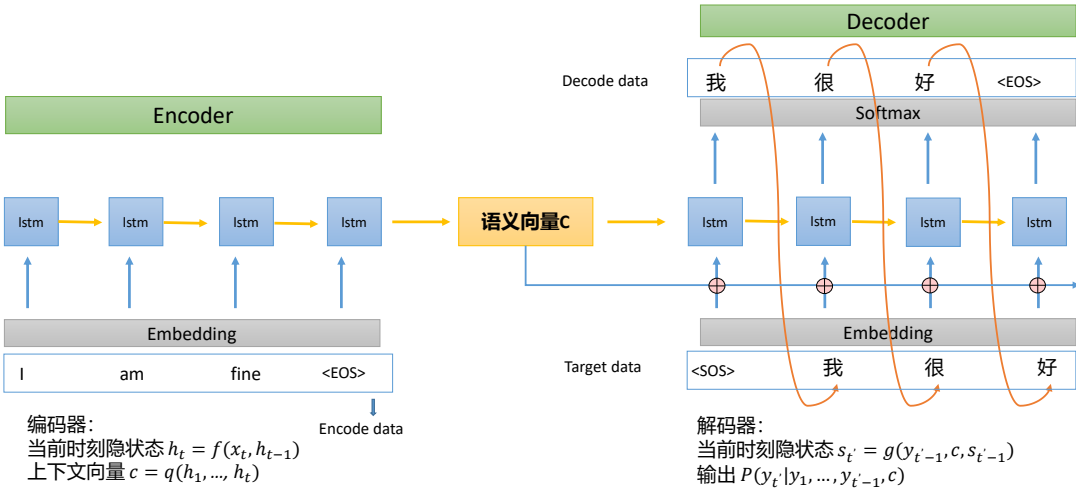
- 编码器将长度可变的输入序列转换成长度固定的上下文向量 (context vector), 并且将输入序列的信息在该上下文向量中进行编码。
- 解码器将上下文向量进行解码, 转换成长度可变的输出序列。



Encoder-Decoder框架

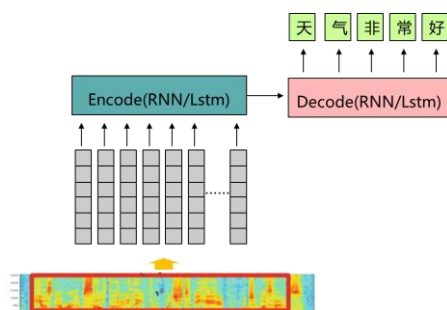
- Seq2Seq的全称是sequence to sequence, 基础的seq2seq = Encoder + Decoder + 语义编码c (连接两者的中间状态向量, 可以理解成两种语言之间的联系规律)。Encoder中将可变长度的信号序列变为固定长度的向量表达, 其实本质也就是进行了特征压缩; Decoder将这个固定长度的向量变成可变长度的目标信号序列; Encoder、Decoder可以是CNN、RNN、Transformer三种结构, Encoder和Decoder可以是相同的结构, 也可以是不同的结构。
- 相同结构, 即都是RNN, 或者CNN, 或者都是Transformer; 不同结构, 比如一个是RNN, 另一个CNN (例如: Image Caption)。

编码器和解码器工作原理



语音识别中的Seq2Seq

- 该模型输入的是音频特征序列，输出的是字母序列。
- 使用Seq2Seq模型，对于输入序列和输出序列长度不要求一样，因此输入可以是音频特征序列，输出可以是字母序列或者粒度更大一些的字序列。
- 使用这个模型评估模型输出的字母序列和目标字母序列之间的差异就可以直接使用交叉熵损失函数来实现。



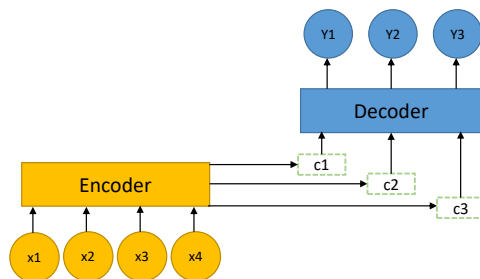
- 详细参考文章 Bahdanau D, Chorowski J, Serdyuk D, et al. End-to-end attention-based large vocabulary speech recognition[C]//2016 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, 2016: 4945-4949.

Seq2Seq模型的问题与优化

- Seq2Seq模型在处理过长序列数据时表现不佳，其原因主要在两方面：
 - RNN模型固有的长期依赖问题（可以用LSTM/GRU解决）。
 - 上下文向量（context vector）无法完全表达整个输入序列的信息。
- 解码器在解码时，需要考虑所有序列，而这其中夹杂了许多无关的噪声或信息，容易出现信息过载的问题。其实大部分输出，只跟输入序列的部分序列有关，而非整个序列。因此基于Seq2Seq模型，引入了注意力机制（attention mechanism）。

Attention机制

- 注意力机制：在解码器的每个步骤中，使用与编码器的直接连接来聚焦于源序列的特定部分。
- 注意力机制在输入信息中聚焦于更为关键的信息，降低对其他信息的关注度，甚至过滤掉无关信息，就可以解决信息过载问题，并提高任务处理的效率和准确性。

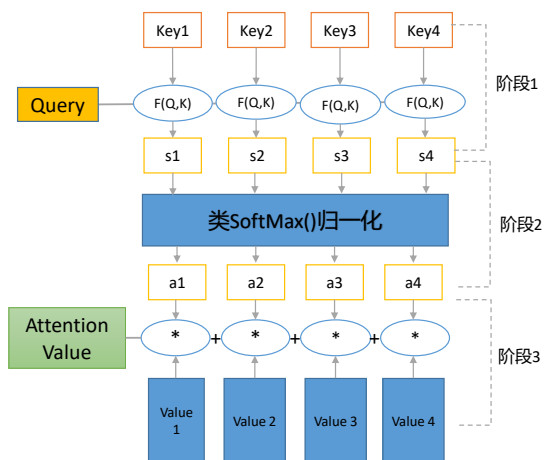


Encoder-Decoder框架（带Attention）

- Encoder-Decoder中加入Attention机制：让Encoder编码出的c向量跟Decoder解码过程中的每一个输出进行加权运算，在Decoder的每一个过程中调整权重取到不一样的c向量。

Attention机制计算步骤

- Q (query) 、K (key) 、V (value) 来自输入与对应权重矩阵计算得到。
- 三步分解：
 - 第一步：query和key进行相似度计算，得到权值；
 - 第二步：将权值进行归一化，得到直接可用的权重；
 - 第三步：将权重和value进行加权求和。



思考题

1. 语音识别的流程是什么？
2. 语音合成的流程是什么？

- 1. 答案参考P36
- 2. 答案参考P43

本章总结

- 本章主要介绍了语音处理的相关理论和技术点，重点讲解了语音识别的技术演进与应用，尤其是重点介绍了语音识别的主要声学模型和语音模型。

缩略语

- ASR: automatic speech recognition, 自动语言识别
- TTS: text to speech, 语音合成
- CTC: connectionist temporal classification, 连接时序分类
- HMM: hidden Markov model, 隐马尔可夫模型
- GMM: Gaussian mixture model, 高斯混合模型
- LSTM: Long Short-Term Memory, 长短期记忆网络

学习推荐

- 华为Learning网站
 - <http://support.huawei.com/learning/Index!toTrainIndex>
- 华为Support案例库
 - <http://support.huawei.com/enterprise/servicecenter?lang=zh>

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。
Bring digital to every person, home, and
organization for a fully connected,
intelligent world.

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.



自然语言处理理论和应用



前言

- 自然语言是人类智慧的结晶，自然语言处理是人工智能中最为困难的问题之一，而对自然语言处理的研究也是充满魅力和挑战的。

目标

- 学习完本课程后，您将能够：
 - 理解自然语言处理的定义
 - 了解自然语言处理的应用场景
 - 理解自然语言处理的核心技术

目录

1. 自然语言处理简介
2. 自然语言处理任务与方法
3. 自然语言处理应用系统

本节概述和学习目标

- 本节主要介绍自然语言处理的定义、应用、难点、发展历史。

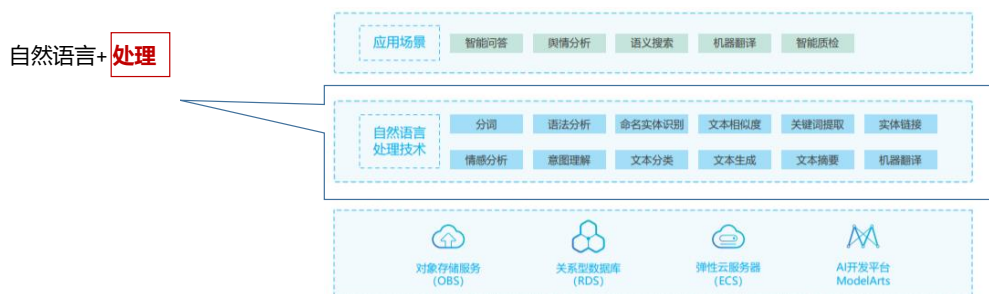
什么是自然语言？

- 什么是自然语言？
 - 以语音为物质外壳，由词汇和语法两部分组成的符号系统。文字和声音是语言的两种属性。
 - 语言是人类交际的工具，是人类思维的载体；人类历史上以语言文字形式记载和流传的知识占人类知识总量的80%以上。
 - 是约定俗成的，有别于人工语言，比如Java、C++等程序设计语言。

什么是自然语言处理？ (1)

- 什么是自然语言处理（Natural Language Processing, NLP）？
 - 自然语言处理就是，利用计算机为工具对人类特有的书面形式和口头形式的自然语言的信息，进行各种类型处理和加工的技术。----冯志伟
 - 自然语言处理可以定义为研究在人与人交际中以及在人与计算机交际中的语言问题的一门学科。自然语言处理要研制表示**语言能力**和**语言应用**的模型，建立计算框架来实现这样的语言模型，提出相应的方法来不断完善这样的语言模型，根据这样的语言模型设计各种实用系统，并探讨这些实用系统的评测技术。----Bill Manaris

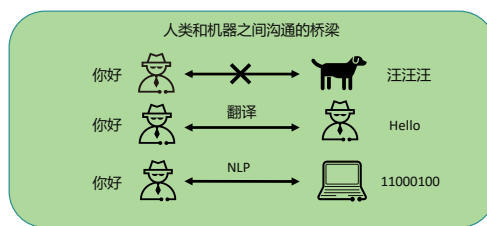
什么是自然语言处理？ (2)



- 自然语言处理的处理主要包括两个部分：自然语言理解、自然语言生成，在这两部分之下有各种的基础算法来支撑这两部分。
 - 自然语言理解（Natural Language Understanding, NLU）：使计算机理解自然语言（人类语言文字）等，重在理解。具体来说，就是理解语言、文本等，提取出有用的信息，用于下游的任务。如情感识别、意图识别、信息检索（包括个性化搜索和语义搜索，文本匹配等）、信息抽取（命名实体提取、关系抽取、事件抽取等）。
 - 自然语言生成（Natural Language Generation, NLG）：提供结构化的数据、文本、图表、音频、视频等，生成人类可以理解的自然语言形式的文本。NLG又可以分为三大类，文本到文本（text-to-text），如翻译、摘要等；文本到其他（text-to-other），如文本生成图片；其他到文本（other-to-text），如视频生成文本。

自然语言处理的重要性

- 在人工智能出现之前，机器智能处理结构化的数据，例如表格。但是生活中大部分的数据都是非结构化的，例如：文章、图片、音频、视频等。
- 其中文本数据，数据量和信息量巨大。为了让机器帮助我们处理问题，则需要分析利用这些数据，让机器读懂我们的语言，从而搭建人类和机器之间沟通的桥梁。



- 语言理解是人工智能领域皇冠上的明珠。---比尔盖茨

自然语言处理应用

- 自然语言处理技术被广泛应用于搜索、对话、翻译、推荐等场景中：
 - 智能问答
 - 对话助手
 - 语义搜索
 - 信息抽取
 - 推荐系统
 - 机器翻译
 - 文本创作
 -

对话理解



智能客服



情感分析

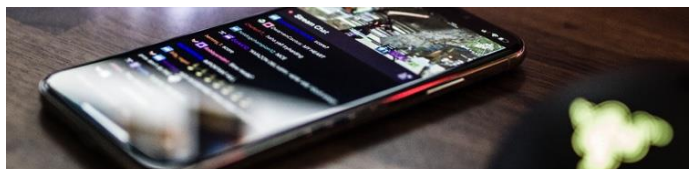
- 对商品评论自动过滤，筛选负面评论供人工审核。

买没几天就降价一点都不开心，闪存跑分就五百多点点 --- 😞
外观漂亮音质不错，现在电子产品基本上都是华为的了 --- 😄

- 属性情感分析：

评论	动力	外观	空间	油耗
2.0T涡轮增压发动机动力强，各种路况表现都不错，高速120超车没压力；外观是我和老婆都比较喜欢的款；后排空间有点小；有点费油啊。	😄	😄	😞	😞

内容审核



用户01 ★★★★★

9月20日 15:10

外观惊艳！国产手机的巅峰之作！！



用户02 ★★★★★

9月20日 16:12

这款手机真*****



违规信息

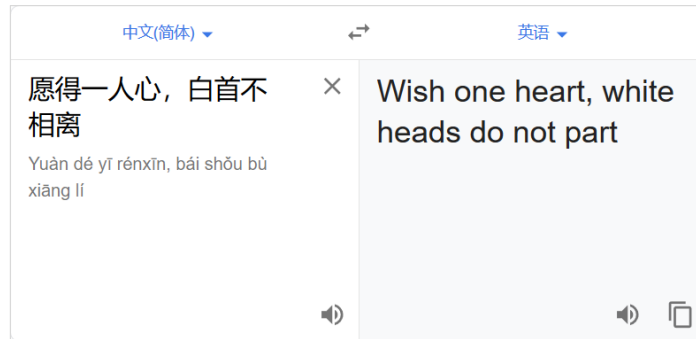
词语联想



文本摘要



机器翻译



自然语言理解层次

- 自然语言的理解和分析是一个层次化的过程，具体可分为以下五个层次：
 - 语音分析，是要根据音位规则，从语音流中区分出一个个独立的音素，再根据音位形态规则找出音节及其对应的词素或词。
 - 词法分析，是找出词汇的各个词素，从中获得语言学的信息。
 - 句法分析，是对句子和短语的结构进行分析，目的是要找出词、短语等的相互关系以及各自在句中的作用。
 - 语义分析，是找出词义、结构意义及其结合意义，从而确定语言所表达的真正含义或概念。
 - 语用分析，是研究语言所存在的外界环境对语言使用者所产生的影响。



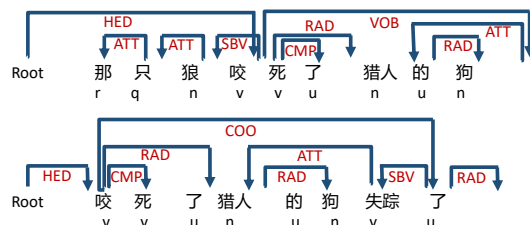
自然语言处理难点 (1)

- 词法歧义：
 - 分词：词语的切分边界比较难确定。
 - 南京/市长/江大桥
 - 南京市/长江/大桥
 - 词性标注：同一个词语在不同的上下文中词性不同。
 - 我/计划/v 考/研/
 - 我/完成/了/计划/n
 - 命名实体识别：人名、专有名称、缩略词等未登录词的识别困难。
 - 高超/nr/a 华明/nr/nt 移动/nt/v

自然语言处理难点 (2)

- 句法歧义：句法层面上的依存关系受上下文的影响。

- 咬死了猎人的狗
 - 那只狼咬死了猎人的狗
 - 咬死了猎人的狗失踪了



自然语言处理难点 (3)

- 语义歧义
 - At last, a computer understands you like your mother.
 - 含义1: 计算机会像你的母亲那样很好的理解你。
 - 含义2: 计算机理解你喜欢你的母亲。
 - 含义3: 计算机会像理解你母亲那样去理解你。

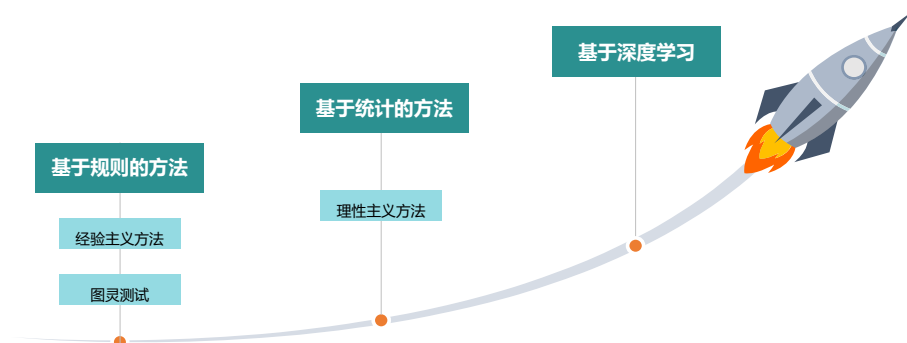
自然语言处理难点 (4)

- 语用歧义

- “你在干什么？”

- 纯粹的询问：朋友之间的简单询问，想知道对方正在做些什么事情。
 - 生气的质疑：当妈妈对淘气的儿子说时，实际表达的是儿子可能干了一些坏事。
 - 疑惑的怀疑：做了一些不太合适或不可告人的事情，带有一种调查或质疑的意味。

自然语言理解的发展历史



- 基于规则的方法: 具有不可避免的缺点, 首先规则不可能覆盖所有语句, 其次这种方法对开发者的要求极高, 开发者不仅要精通计算机还要精通语言学, 因此, 这一阶段虽然解决了一些简单的问题, 但是无法从根本上将自然语言理解实用化。
- 基于统计的方法: 随着互联网的高速发展, 丰富的语料库成为现实以及硬件不断更新完善, 自然语言处理思潮由经验主义向理性主义过渡, 基于统计的方法逐渐代替了基于规则的方法。
- 基于深度学习的方法: 从输入到输出做端到端的训练。把输入到输出对的数据准备好, 设计并训练一个神经网络, 即可执行预想的任务。RNN已经是自然语言处理最常用的方法之一, GRU、LSTM等模型相继引发了一轮又一轮的热潮。

思考题

1. （判断题）自然语言处理主要包含自然语言理解和自然语言生成两部分（ ）
 - A. True
 - B. False

- 答案：A

本节小结

- 自然语言是以语音为物质外壳，由词汇和语法两部分组成的符号系统。文字和声音是语言的两种属性。
- 自然语言处理主要包含自然语言理解和自然语言生成两部分。
- 自然语言处理技术被广泛应用于搜索、对话、翻译、推荐等场景中。
- 歧义是自然语言处理最大的难点。
- 自然语言处理经历了从基于规则的方法到基于统计的方法再到深度学习方法的一个发展过程。

目录

1. 自然语言处理简介
- 2. 自然语言处理任务与方法**
3. 自然语言处理应用系统

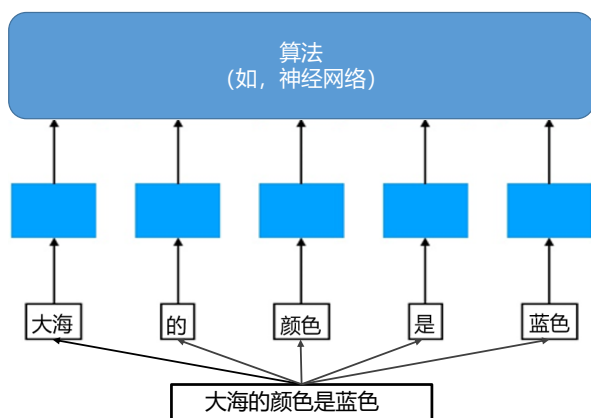
本节概述和学习目标

- 本节主要介绍自然语言处理中主要的几个具体的任务及其实现方法。包括词向量、文本分类、序列标注、语言模型与文本生成、Transformer。

目录

1. 自然语言处理简介
2. **自然语言处理任务与方法**
 - 词向量
 - 文本分类
 - 序列标注
 - 语言模型与文本生成
 - Transformer
3. 自然语言处理应用系统

为什么需要词向量？

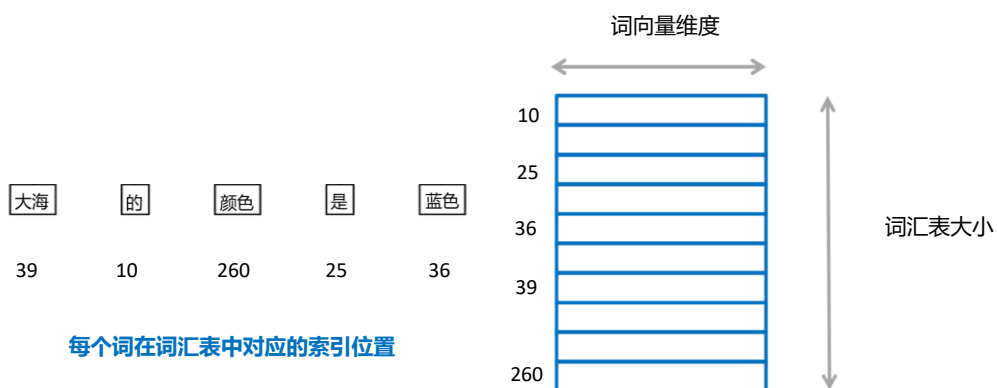


词向量

词序列

原始文本

如何使用词向量？



如何表示词？ - 离散符号表示

- OneHot, 独热编码, 是一种用离散符号来表示词向量的方法。向量的长度为词汇表中总的词数。

词汇表: This is a simple example

每个词在词汇表中对应的索引位置

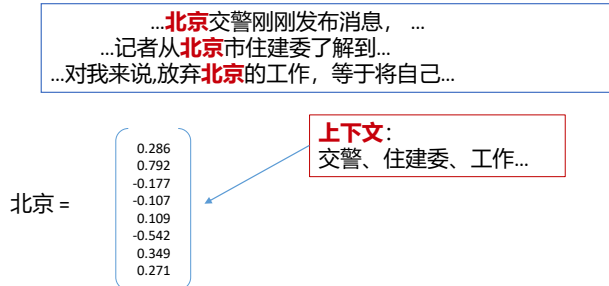
This = [1, 0, 0, 0, 0]
Is = [0, 1, 0, 0, 0]
a = [0, 0, 1, 0, 0]
simple = [0, 0, 0, 1, 0]
Example = [0, 0, 0, 0, 1]

↔
词汇表大小

- 思考: OneHot的缺点?

如何表示词？ - 分布式语义表示

- **分布式语义表示**：每个单词不再是完全正交的 0-1 向量，而是在多维实数空间中的一个点，具体表现为一个实数向量。很多时候，也会把单词的这种分布式表示叫做词嵌入（Word Embedding）。
- **分布式假设**：某个单词的含义由它周围的单词形成。



- 分布式语义表示解决了One-hot无法表示近似词的问题。
- 一个单词的真正意思应该由上下文决定。

如何表达词的语义？

- “北京” vs “上海”？

...北京交警刚刚发布消息, ...
...记者从北京市住建委了解到...
...对我来说,放弃北京的工作,等于将自己...

...上海交警刚刚发布消息...
...记者从上海市住建委了解到...
...我在上海工作, ...

- 思考：为什么“北京”和“上海”的词向量的相似度高？

- ‘北京’，‘上海’拥有相似的上下文

如何体现上下文？ - 共现矩阵

语料库：

我 爱 上海
我 爱 杭州
我 爱 北京 天安门



窗口为2的共现矩阵：

	我	爱	上海	杭州	北京	天安门
我	0	3	0	0	0	0
爱	3	0	1	1	1	0
上海	0	1	0	0	0	0
杭州	0	1	0	0	0	0
北京	0	1	0	0	0	1
天安门	0	0	0	0	1	0

共现矩阵 + SVD

- X 为共现矩阵, SVD矩阵分解为低维稠密矩阵:

$$\begin{pmatrix} \overset{\hat{X}}{x_{11}} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix}_{m \times n} \approx \begin{pmatrix} \overset{U}{u_{11}} & \dots & u_{1r} \\ \vdots & \ddots & \\ u_{m1} & & u_{mr} \end{pmatrix}_{m \times r} \begin{pmatrix} \overset{S}{s_{11}} & 0 & \dots \\ 0 & \ddots & \\ \vdots & & s_{rr} \end{pmatrix}_{r \times r} \begin{pmatrix} \overset{V^T}{v_{11}} & \dots & v_{1n} \\ \vdots & \ddots & \\ v_{r1} & & v_{rn} \end{pmatrix}_{r \times n}$$

- 思考: 共现矩阵+SVD方法的缺点?

- 共现矩阵, 高维稀疏
- 共现矩阵没有考虑词的顺序信息
- 每增加一个词, 需全部重新计算
- Svd学到的只是浅层的语义

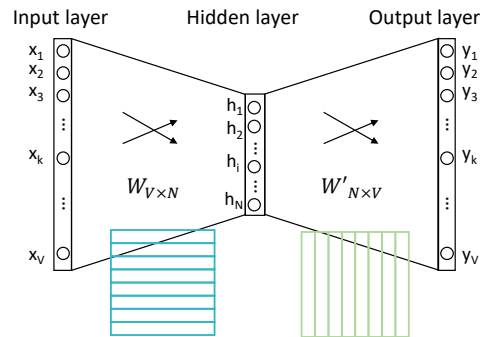
为什么不直接学习词的分布式表达呢？

- 共现矩阵+SVD 是一种基于统计的直接计算词向量的方式，这并不是唯一的方法也不是最好的方法。



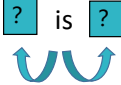
Word2Vec

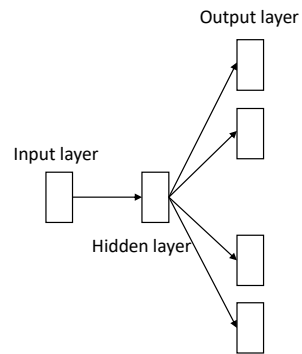
- 准备一个大的语料库
- 词汇表中的每个词都有对应的向量表示
- 遍历语料库中每篇文章的每个位置，每个位置都有中心词 c 和对应的一些上下文词 o
- 用 c 和 o 的词向量的内积来计算给定 c 预测出 o （或相反）的概率
- 持续优化调整词向量使得概率最大化



Word2Vec: Skip-gram (SG)

- 用中心词来预测周围的词。在 skip-gram 中，会利用周围的词的预测结果情况，使用梯度下降来不断的调整中心词的词向量，最终所有的文本遍历完毕之后，也就得到了文本所有词的词向量。

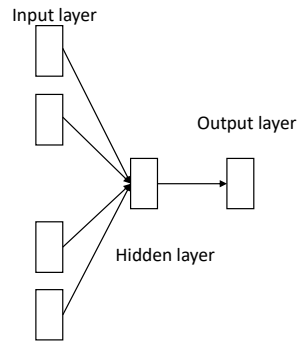
 is ? simple example



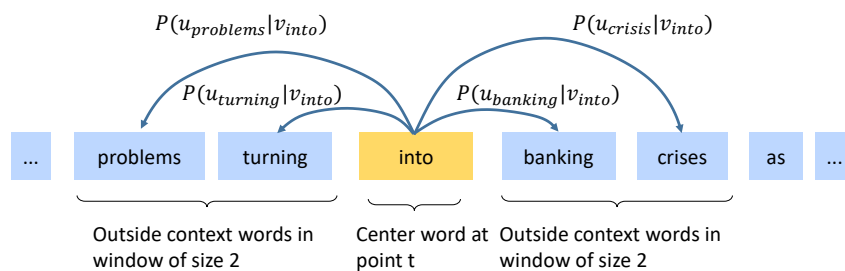
Word2Vec: Continuous Bag of Words (CBOW)

- 用周围词预测中心词，利用中心词的预测结果情况，使用梯度下降等方法，不断的去调整周围词的向量。当训练完成之后，每个词都会作为中心词，把周围词的词向量进行了调整，这样也就获得了整个文本里面所有词的词向量。

This ? a simple example

Word2Vec - Skip-gram 示例



Word2Vec: 损失函数

$$J(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log P(w_{t+j} | w_t; \theta)$$

$$p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

u_o : 当前词的词向量
 v_c : 上下文中词的词向量

点乘评估的是中心词和上下文词的相似性，相似性越高，总体概率越大

概率归一化

- 负对数似然（交叉熵）
- 概率越大，损失越小

Word2Vec 问题与改进

$$p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

词汇表中每个词都要算一遍，导致训练太慢

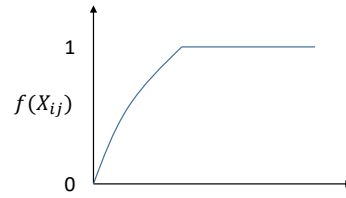
- 优化方法：
 - Hierarchical softmax
 - Negative sampling

GloVe

- GloVe (Global Vectors for Word Representation) : 通过将全局的词共现信息和局部上下文信息结合起来, 通过矩阵分解的方式学习词向量。

$$J(\theta) = \frac{1}{2} \sum_{i,j=1}^W f(X_{ij})(u_i^T v_j - \log X_{ij})^2$$

$$X_{final} = U + V$$



x 是共现矩阵

FastText

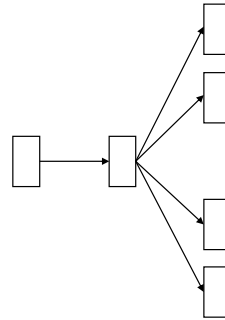
- FastText: 基于词袋模型 (Bag of Words) 的文本表示方法, 通过将词分解为子词 (subwords) 并将子词的向量表示相加来学习词向量, 充分利用字符N-gram信息。

Apple (2-gram) = <ap, pp, pl, le>



训练过程中学习到的

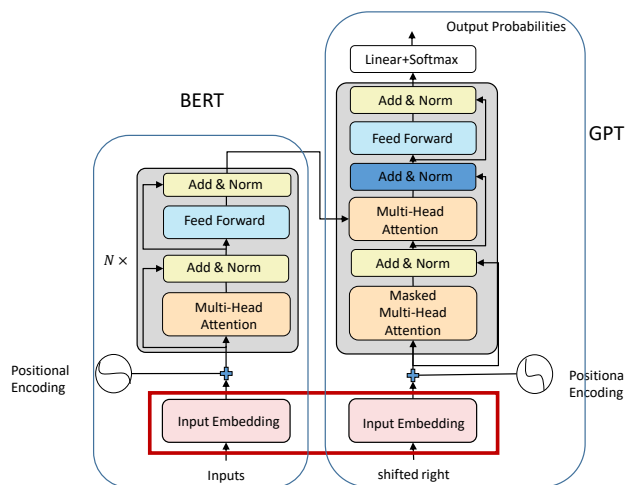
ap	[-0.3, 0.2, 0.5]
pp	[0.4, -0.1, 0.6]
pl	[0.2, 0.3, -0.4]
le	[-0.1, 0.5, 0.2]
apple	[0.2, 0.9, 0.9]



- 假设Apple前后不包含开始和结束, 若包含则为<a, a, pp, pl, le, e>

Transformer-based模型

- Transformer-based 模型：如 BERT、GPT 等，从大规模文本数据中学习词向量表示。



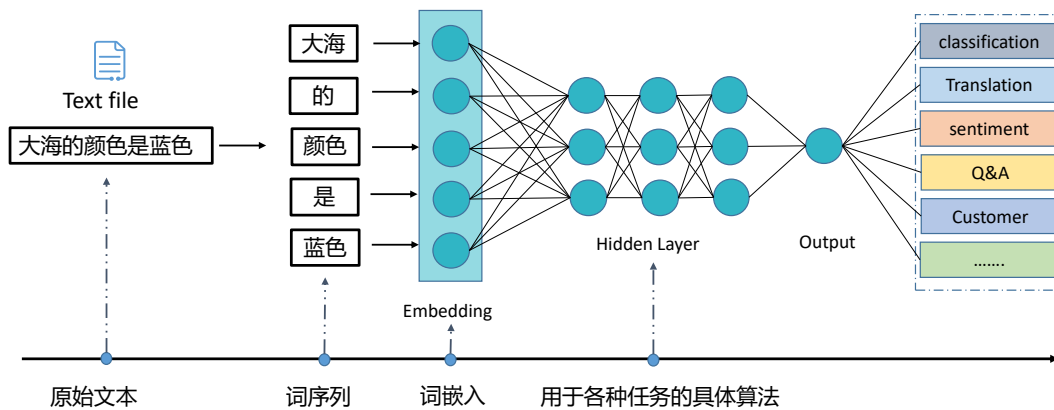
- K,V 来自编码器的输出，Q解码器的输出
- 预训练（Word2vec）和随机初始化（Transformer-based）会随着任务学习的过程而不断变化

词向量评估

- 内部评估
 - 相似度任务：计算两个词的相似度。比如数据集 wordsim353。
 - 类比任务：经典的 $\text{King} - \text{Queen} = \text{Man} - \text{Woman}$ 。
- 外部评估
 - 以**实际任务的指标**来评价词向量的好坏。如命名实体识别、机器翻译、文本分类等。

下游任务

- 有了词向量后，能做更多下游任务



- 好的词向量可以助力下游任务，使得下游任务做得更加出色。

目录

1. 自然语言处理简介
- 2. 自然语言处理任务与方法**
 - 词向量
 - 文本分类
 - 序列标注
 - 语言模型与文本生成
 - Transformer
3. 自然语言处理应用系统

文本分类

- 定义:

- 输入:

- 一篇文章 d
 - 固定的类别集合 $C = \{C_1, C_2, \dots, C_j\}$

- 输出:

- 预测出的该文章所属的类别 $c \in C$

input: 今天发了工资, 我很开心!



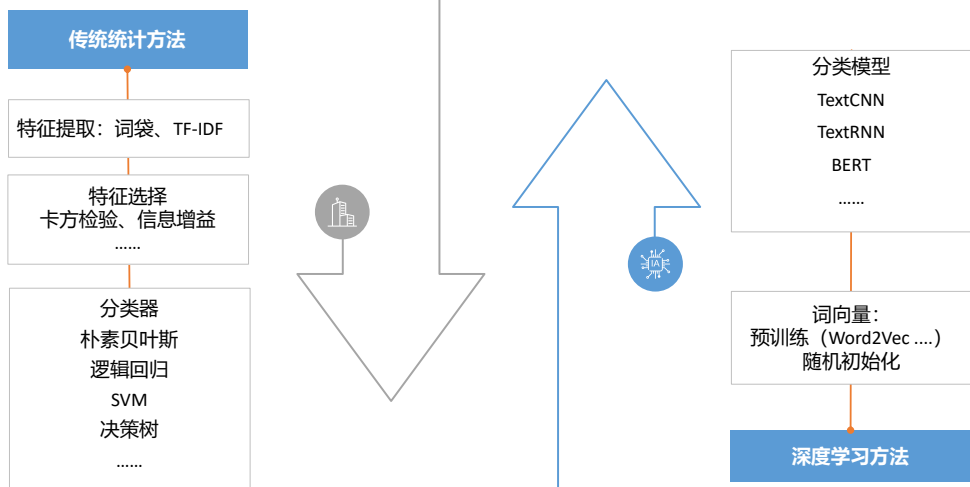
$C = \{0: positive, 1: negative\}$

output: 0 (positive)

- 应用:

- 情感分析, 意图识别, 内容审核

文本分类方法



特征提取 – 词袋

- 词袋模型 (Bag of Words, BOW) 是一种简单的文本向量化的一种方法，它不考虑句子中单词的顺序，只考虑词表 (vocabulary) 中单词在这个句子中的出现次数。

语料库:

我 爱 上海
我 爱 杭州
我 爱 北京 天安门



	上海	北京	天安门	我	杭州	爱
d1	1	0	0	1	0	1
d2	0	0	0	1	1	1
d3	0	1	1	1	0	1

特征提取 - TF-IDF

- TF-IDF（词频-逆文档频率）向量化相比于词袋向量化，更好的考虑了句子中每个词的重要程度。

$$w_{i,d} = tf_{i,d} \times \log(n / 1 + df_i)$$

语料库:

我 爱 上海
我 爱 杭州
我 爱 北京 天安门



	上海	北京	天安门	我	杭州	爱
d1	0.767	0.0	0.0	0.453	0.0	0.453
d2	0.0	0.0	0.0	0.453	0.767	0.453
d3	0.0	0.608	0.608	0.359	0.0	0.359

特征选择

- 词袋向量化或TF-IDF向量化后的特征通常都有高维稀疏的问题，直接给分类器训练通常效果并不好，过高的数据维度也影响训练和预测的速度，因此，通常会使用特征选择的方法过滤一些无效特征。文本分类中常见的特征选择方法有：
 - 卡方检验
 - 互信息
 - 信息增益
 -

卡方检验

- 源于统计学的卡方分布 (chi-square) , 从 (类, 词项) 相关表出发, 考虑每一个类和每一个词项的相关情况, 度量两者 (特征和类别) 独立性的缺乏程度, 卡方越大, 独立性越小, 相关性越大。

卡方值计算

	<i>Term = jaguar</i>	<i>Term ≠ jaguar</i>	expected: f_e
<i>Class = auto</i>	2 (0.25)	500 (502)	
<i>Class ≠ auto</i>	3 (4.75)	9500 (9498)	observed: f_o

Expected value, for instance in the up-left cell is:
 $P(c=auto) * P(T=jaguar) * \text{\# of cases} = 502/10005 * 5/10005 * 10005 = 0.2508$

$$\chi^2(j,a) = \sum (O - E)^2 / E = (2 - .25)^2 / .25 + (3 - 4.75)^2 / 4.75 + (500 - 502)^2 / 502 + (9500 - 9498)^2 / 9498 = 12.9 \ (p < .001)$$

- O是观测值，E是期望值

分类器

- 在完成特征提取后，传统的机器学习分类算法都可以用来对文本进行分类：
 - 朴素贝叶斯
 - 逻辑回归
 - SVM
 - 决策树
 - 集成算法：随机森林，XGBoost.....

朴素贝叶斯

- 朴素贝叶斯分类方法 (Naïve Bayes) 是利用贝叶斯定理来预测一个未知类别的样本属于各个类别的可能性, 选择其中可能性最大的一个类别作为该样本的最终类别。
- 给定文档 d 和类别集合 $C = \{c_1, c_2\}$, x 是文档 d 中的单词, 计算文档所属类别:

$$\begin{aligned} C_{MAP} &= \operatorname{argmax}_{c \in C} P(c|d) \\ &= \operatorname{argmax}_{c \in C} \boxed{\frac{P(d|c)P(c)}{P(d)}} \quad \text{贝叶斯公式} \\ &= \operatorname{argmax}_{c \in C} P(d|c)P(c) \\ &= \operatorname{argmax}_{c \in C} P(x_1, x_2, \dots, x_n|c)P(c) \\ &= \operatorname{argmax}_{c \in C} P(c) \prod_{x \in X} P(x|c) \end{aligned}$$

垃圾邮件识别

输入: 黄金地段, 快速抢购
输出: 垃圾邮件 or 正常邮件?

$$\operatorname{argmax}_{c \in \mathcal{C}} P(c) \prod_{x \in X} P(x|c)$$

$$p1 = p(\text{垃圾邮件}) \times p(\text{黄金}|\text{垃圾邮件}) \times p(\text{地段}|\text{垃圾邮件}) \times p(\text{快速}|\text{垃圾邮件}) \times p(\text{抢购}|\text{垃圾邮件})$$

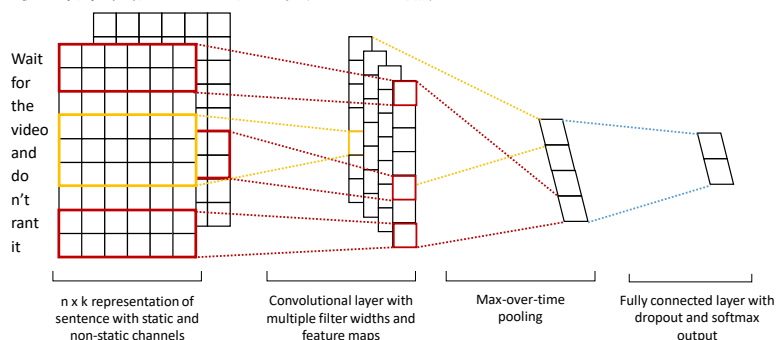
$$p2 = p(\text{正常邮件}) \times p(\text{黄金}|\text{正常邮件}) \times p(\text{地段}|\text{正常邮件}) \times p(\text{快速}|\text{正常邮件}) \times p(\text{抢购}|\text{正常邮件})$$

$$\text{class} = \begin{cases} \text{垃圾邮件} & \text{if } p1 > p2 \\ \text{正常邮件} & \text{if } p1 < p2 \end{cases}$$

$$p(\text{抢购}|\text{垃圾邮件}) = \frac{\text{count}(\text{垃圾邮件中 '抢购' 出现的次数})}{\text{count}(\text{垃圾邮件中总的词数})}$$

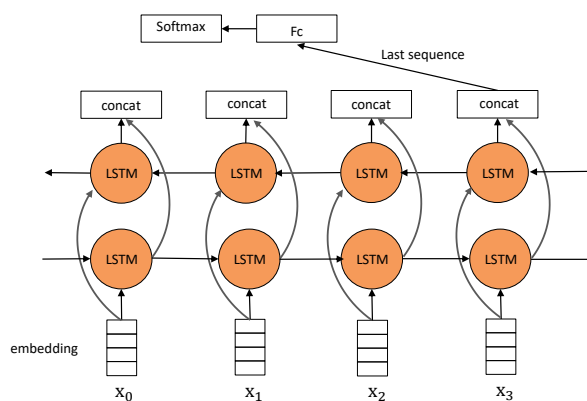
深度学习方法 - TextCNN

- TextCNN是一种使用卷积神经网络对文本进行分类的模型。
- 文本在经过词嵌入后被送入神经网络，与CNN不同的是，TextCNN的卷积操作作用在一维数据上。
- TextCNN提取的特征类似N-gram模型，是词的局部关系。



- 参考链接: <https://arxiv.org/abs/1408.5882>
- 我们之前提前CNN时，通常会认为是属于CV领域，用于计算机视觉方向的工作，但是在2014年，Yoon Kim针对CNN的输入层做了一些变形，提出了文本分类模型textCNN。与传统图像的CNN网络相比，textCNN 在网络结构上没有任何变化(甚至更加简单了)，从图中可以看出textCNN 其实只有一层卷积，一层max-pooling，最后将输出外接softmax 来n分类。
- 与图像当中CNN的网络相比，textCNN 最大的不同便是在输入数据的不同：图像是二维数据，图像的卷积核是从左到右，从上到下进行滑动来进行特征抽取。自然语言是一维数据，虽然经过word-embedding 生成了二维向量，但是对词向量做从左到右滑动来进行卷积没有意义。比如 "今天" 对应的向量[0, 0, 0, 0, 1]，按窗口大小为 1×2 从左到右滑动得到[0,0], [0,0], [0,0], [0, 1]这四个向量，对应的都是"今天"这个词汇，这种滑动没有帮助。

深度学习方法 - TextRNN

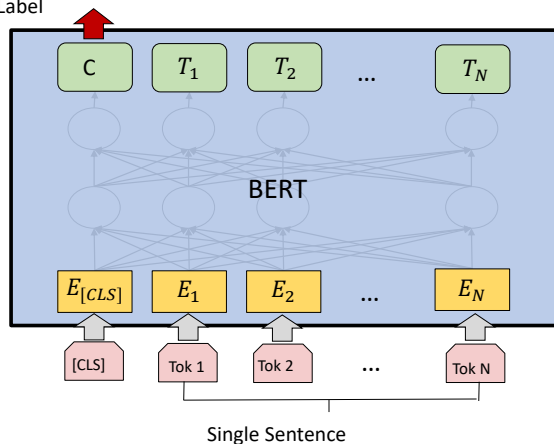


- 一般取前向/反向LSTM在最后一个时间步长上隐藏状态，然后进行拼接，在经过一个softmax层(输出层使用softmax激活函数)进行一个多分类；或者取前向/反向LSTM在每一个时间步长上的隐藏状态，对每一个时间步长上的两个隐藏状态进行拼接，然后对所有时间步长上拼接后的隐藏状态取均值，再经过一个softmax层(输出层使用softmax激活函数)进行一个多分类(2分类的话使用sigmoid激活函数)。

深度学习方法 - BERT

BERT: Pre-training of Deep Bidirectional Transformers

Class Label



- BERT 【CLS】标志位对应的输出是整个句子的向量编码，可以直接softmax实现分类

评估指标

- 正确率 (accuracy) : 判断正确的样本数占有所有样本数的比例
- 错误率 (error rate): 判断错误的样本数占有所有样本数的比例
- 召回率(recall)
 - $Recall = \text{正确的目标类判断} / \text{数据集中目标类的总数}$
- 精度(precision)
 - $Precision = \text{正确的目标类判断} / \text{被判断为目标类的总数}$
- F1-score

$$F1\text{-score} = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

举例

- 我们训练了一个机器学习的模型用来识别垃圾邮件，现在用200篇邮件文本来验证下模型的性能指标。这200篇文档中，170张是正常邮件，30篇是垃圾邮件。模型的识别结果为160篇是正常邮件，40篇是垃圾邮件。

$$\text{精度: } P = \frac{TP}{TP+FP} = \frac{140}{140+20} = 87.5\%$$

$$\text{召回率: } R = \frac{TP}{P} = \frac{140}{170} = 82.3\%$$

$$\text{准确率: } ACC = \frac{TP+TN}{P+N} = \frac{140+10}{170+30} = 75\%$$

$$\text{F1-score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{2 \times 0.875 \times 0.933}{0.875 + 0.933} = 0.848$$

预测 实际	yes	no	合计
yes	140	30	170
no	20	10	30
合计	160	40	200

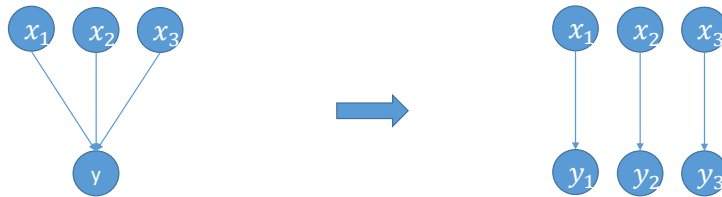
目录

1. 自然语言处理简介
2. **自然语言处理任务与方法**
 - 词向量
 - 文本分类
 - **序列标注**
 - 语言模型与文本生成
 - Transformer
3. 自然语言处理应用系统

序列标注

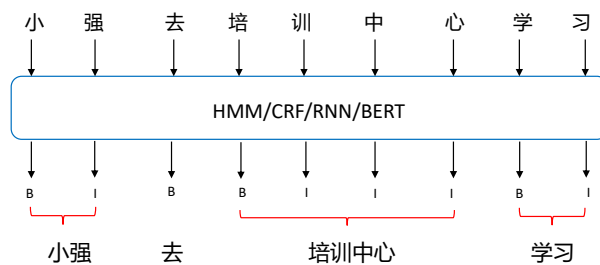
- 序列标注，即对每一个输入 x_i ，都有对应的标签 y_i 。

文本分类 ➡ 序列标注



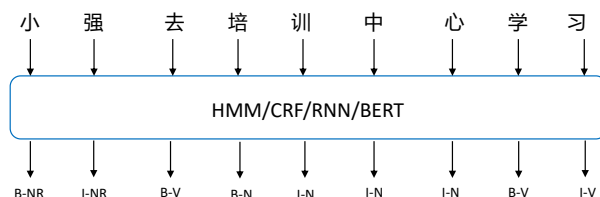
中文分词

- **中文分词(Chinese Word Segmentation)**：指的是将一个汉字序列切分成一个个单独的词。**分词**就是将连续的字序列按照一定的规范重新组合成词序列的过程。
- 例如：一九九八年/中国/实现/进出口/总值/达/一千零九十八点二亿/美元



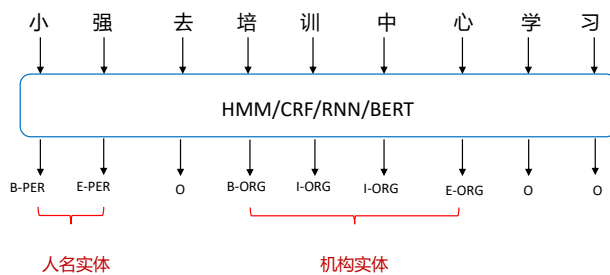
词性标注

- **词性标注** (Part-Of-Speech tagging, POS tagging)：是指为句子的分词结果中的每个单词标注一个正确的词性的程序，也即确定每个词是名词、动词、形容词或者其他词性的过程。例如：迈向/v 充满/v 希望/n 的/uj 新/a 世纪/n。词性标注是很多NLP任务的预处理步骤，如句法分析、信息抽取，经过词性标注后的文本会带来很大的便利性，但不是不可或缺步骤。

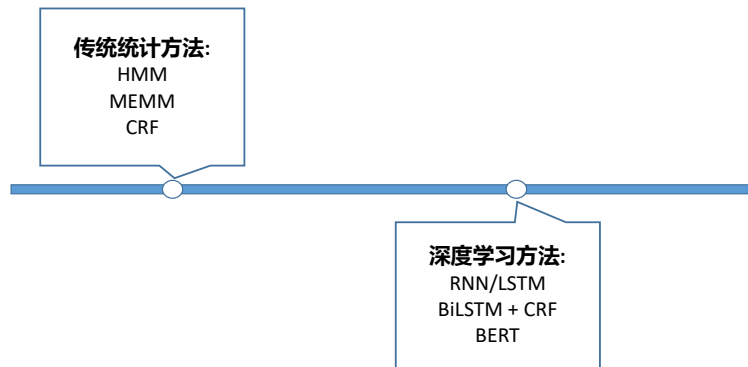


命名实体识别

- 命名实体识别 (Named Entity Recognition, NER) 为自然语言处理 (NLP) 的基础任务之一，其目标是提取文本中的命名实体并对这些实体进行分类，比如人名、地名、机构、时间、货币和百分比等。

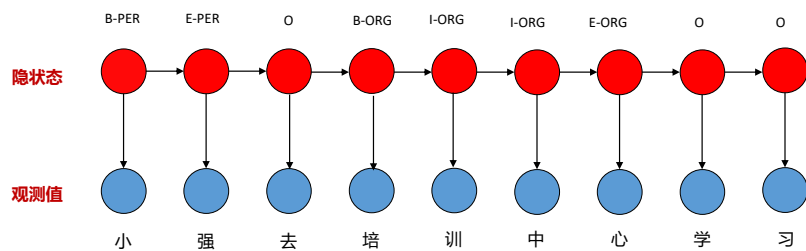


序列标注方法



HMM

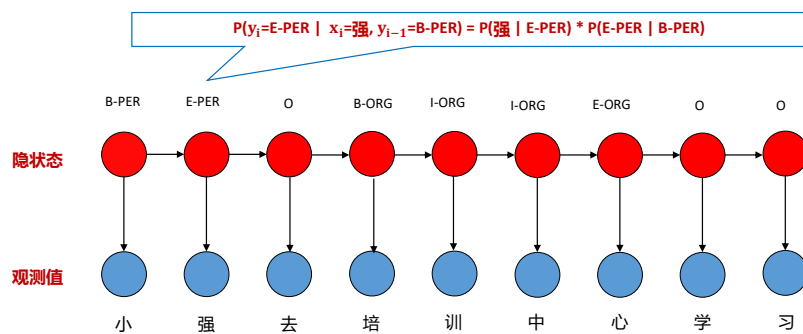
- 隐马尔科夫模型 (Hidden Markov Models, HMM) 有两个假设：
 - 隐状态独立假设 (齐次马尔科夫假设)：当前状态只与上一状态有关。
 - 观测独立假设：当前观测值只由当前隐状态产生，与其他观测值相互独立。



HMM概率计算

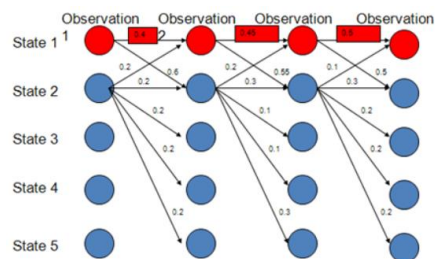
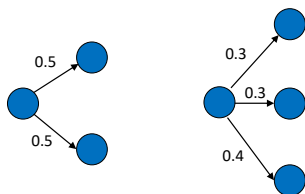
- HMM搜索使得联合概率 $P(X, Y)$ 最大的隐状态序列 $(y_1, y_2, \dots, y_n)$ 。

$$P(X, Y) = \prod_{i=1}^n p(x_i, y_i)$$



HMM问题 - 标签偏置

- HMM的状态独立假设，使得在计算标签转移概率时在局部归一化转移概率，这使得分支少的状态比分支多的状态有着更大的概率，使得算法倾向于选择分支少的状态。

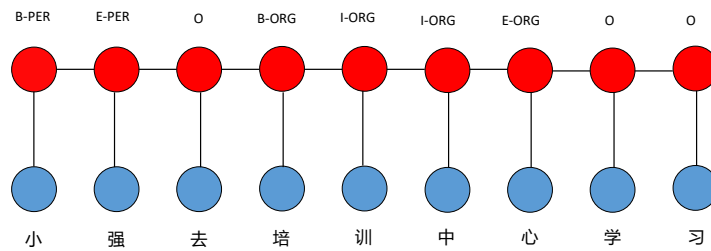


HMM问题 - 上下文缺失

- HMM的观测独立假设，使得算法当前时间步的概率只与当前观测有关，无法利用周围的上下文特征，使得HMM的性能非常受限。

CRF

- 条件随机场 (Conditional Random Field, CRF) , 取消了HMM的两个独立假设, 把标签转移和上下文输入都当做全局特征之一, 在全局进行概率归一化, 解决了HMM的标签偏置和上下文特征缺失问题。



CRF - 概率计算

定义特征

$$f_k(y_{i-1}, y_i, x, i), k = 1, 2, \dots, K$$

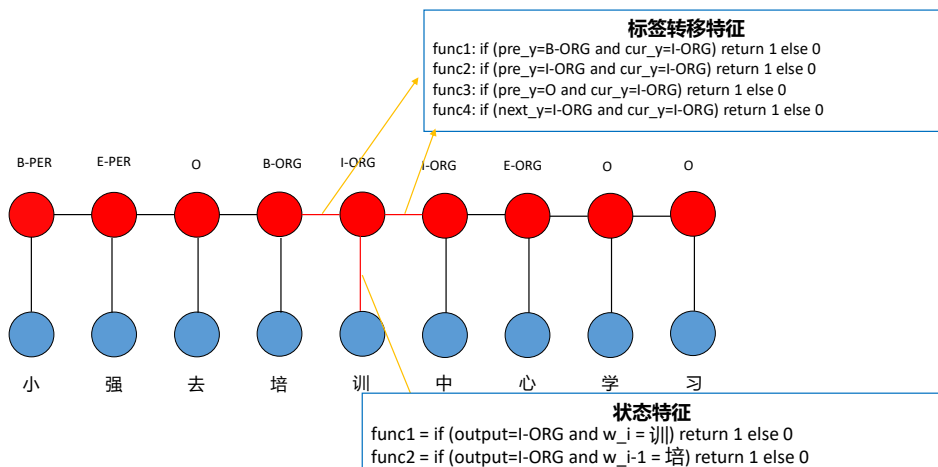
计算score

$$score(y|x) = \sum_{j=1}^m \sum_{i=1}^n \lambda_j f_j(x, i, y_i, y_{i-1})$$

概率归一化

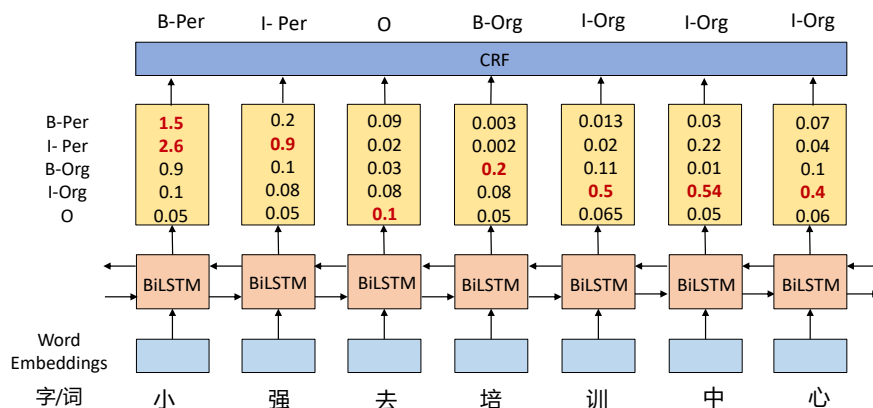
$$P(y|x) = \frac{\exp(score(y|x))}{\sum_y \exp(score(y|x))}$$

CRF - 特征



BiLSTM + CRF

- BiLSTM+CRF里的CRF并不是指传统的CRF算法，这里只是用于学习标签转移特征并进行全局维度解码的一个神经网络层。



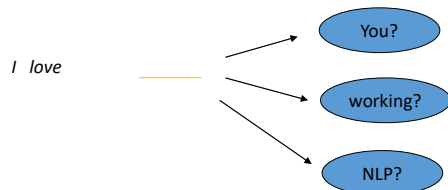
- 因为BiLSTM针对每个单词的输出是标签得分，对于每个单词，我们可以选择最高得分的标签作为预测结果。
- 但是大多数情况下是不能获得正确标签的。
- CRF层可以对最终的约束标签添加一些约束条件，从而保证预测标签的有效性。而这些约束条件是CRF层自动从训练数据中学到。

目录

1. 自然语言处理简介
- 2. 自然语言处理任务与方法**
 - 词向量
 - 文本分类
 - 序列标注
 - 语言模型与文本生成
 - Transformer
3. 自然语言处理应用系统

语言模型与文本生成

- 什么是语言模型？
 - 给定一些词序列，预测最有可能的下一个词是什么。



- 更正式的定义：给定词序列 $w_1, w_2, \dots, w_{i-1}$ ，计算下一个词 w_i 的概率分布：

$$P(w_i | w_1, w_2, \dots, w_{i-1}), \quad w_i \text{ 可能是词汇表中任何一个词}$$

- 也可认为语言模型是对一段文本进行概率计算的模型：

$$P(w_1, w_2, \dots, w_m) = P(w_1)P(w_2 | w_1) \dots P(w_i | w_1, w_2, \dots, w_{i-1}) \dots P(w_m | w_1, w_2, \dots, w_{m-1})$$

N-Gram语言模型

- 利用**n元模型** (**n-gram model**) 估算条件概率, 即忽略**距离大于等于n**的上文词的影响, 因此若用频数计数的比例来计算**n元条件概率**可表示为:

$$P(w_i | w_1, w_2, \dots, w_{i-1}) \approx P(w_i | w_{i-(n-1)}, w_{i-(n-2)}, \dots, w_{i-1})$$
$$= \frac{\text{count}(w_{i-(n-1)}, w_{i-(n-2)}, \dots, w_{i-1}, w_i)}{\text{count}(w_{i-(n-1)}, w_{i-(n-2)}, \dots, w_{i-1})}$$

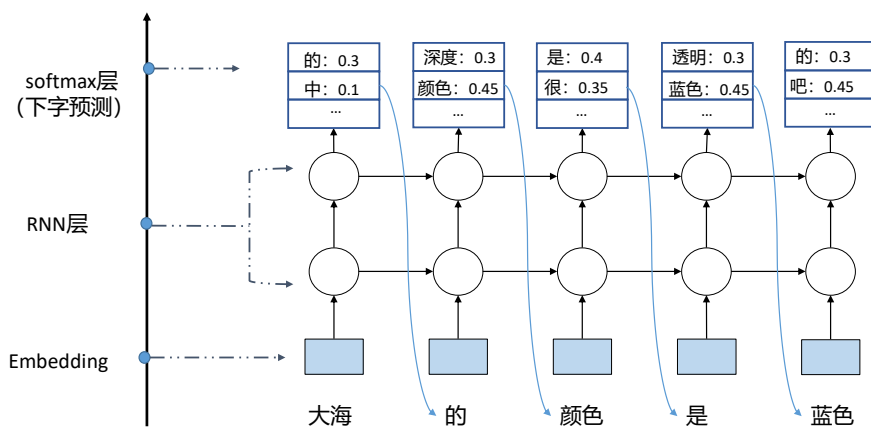
- 当 $n = 1$ 时称为**一元模型**, $P(w_1, w_2, \dots, w_m) = P(w_1)P(w_2) \dots P(w_m)$
- 当 $n = 2$ 时称为**二元模型**, $P(w_1, w_2, \dots, w_m) = P(w_1)P(w_2 | w_1) \dots P(w_m | w_{m-1})$
- 例如:

```
<s> I am Lily </s>  
<s> Lily I am </s>  
<s> I do not like green eggs and ham </s>
```

```
p(I | <s>) = 2/3 = 0.667  
p(am | I) = 2/3 = 0.667  
p(</s> | Lily) = 1/2 = 0.5  
.....
```

- 问题:
- 稀疏: 窗口越大, 越容易出现count为0的问题
- 存储: 这些count需要预先存储下来, 这个量太大了

RNN语言模型

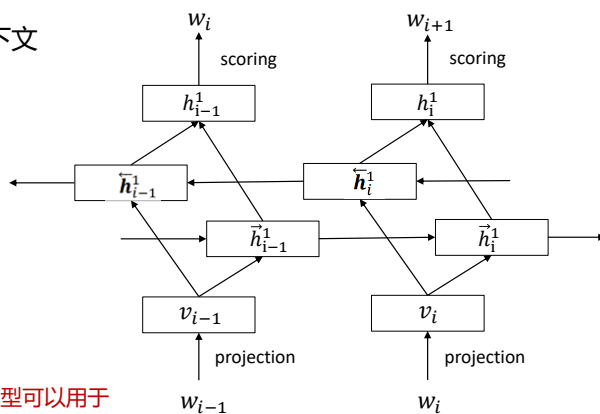


- RNN语言模型优势：

- 可以处理任意长度文本序列，而参数量不变。
- 比n-gram能处理更长的上下文，而且没有稀疏的问题。

双向RNN语言模型

- 考虑了左右两边的上下文
- 不能用于文本生成



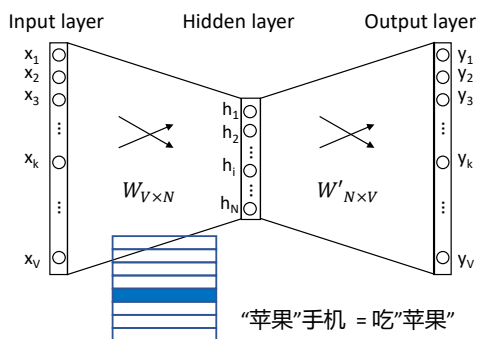
思考：双向RNN语言模型可以用于什么？

- <https://arxiv.org/pdf/1602.06064.pdf>
- 为什么不能用于文本生成？
 - 文本生成是单向的，双向的相当于已经知道了下字预测的答案。

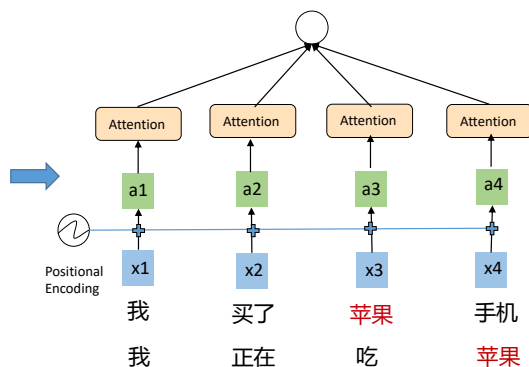
语言模型用于生成词向量

- 静态 vs 动态

每个词学习固定的语义表达。

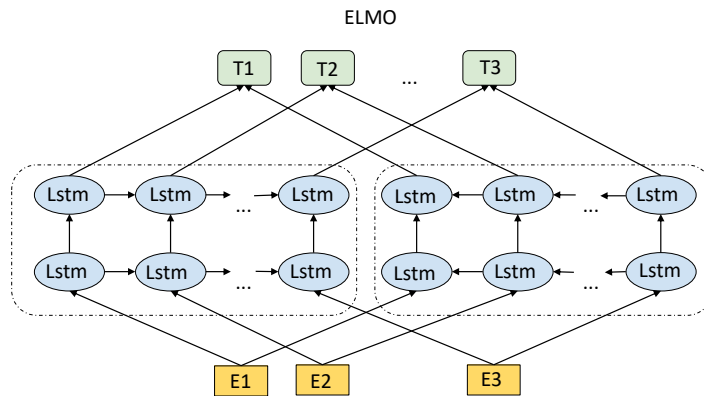


基于语言模型的词嵌入方法，在语料库中训练上下文表示，根据上下文动态生成表示向量。



- 静态的词向量表达是固定的，“苹果”的词向量不变，动态词向量会跟着下游的任务不断学习变化，“苹果”的词向量会不断变化。

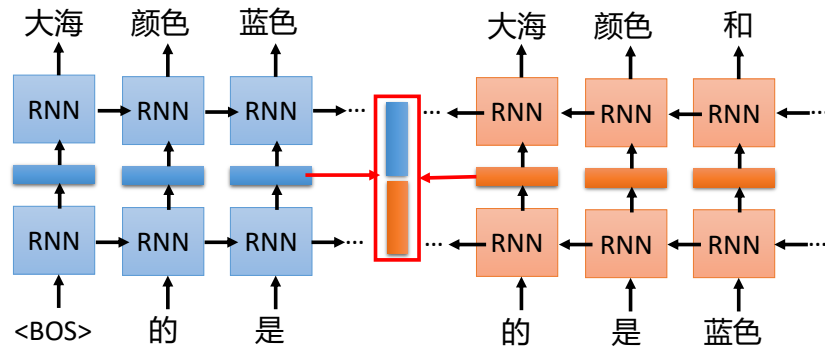
ELMO - Embeddings from Language Model (1)



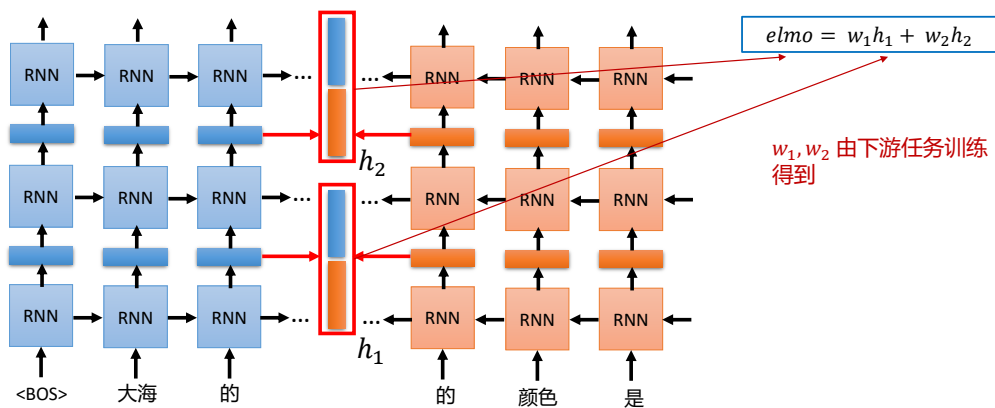
- 区别于双向RNN，两个独立的单向LSTM模型，可以做生成。

ELMO - Embeddings from Language Model (2)

文本：“大海的颜色是蓝色和绿色”



ELMO - Embeddings from Language Model (3)



- 不止一层的RNN，多层的输出也就是图中的 h ，因为不同的 h 代表不同的特征，所以不同层次输出的向量都十分重要，所以将这些向量进行加权重相加，权重也是可学习的参数，不同任务学习到的权重不同。

ELMo词向量

- 在双向模型预训练完成后，模型的编码部分便可以用来计算任意文本的动态词向量表示。在ELMo模型中，不同层次的隐含层向量蕴含了不同层次或粒度的文本信息。
 - 越接近顶层的LSTM隐含层表示，通常编码了更多的语义信息（对于阅读理解类语义信息需求高的任务适用）。
 - 接近底层的隐含层表示更偏重与词法句法信息（适用于命名实体识别类任务）。

目录

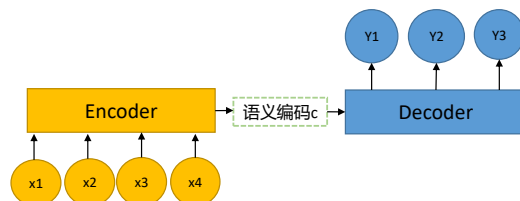
1. 自然语言处理简介
- 2. 自然语言处理任务与方法**
 - 词向量
 - 文本分类
 - 序列标注
 - 语言模型与文本生成
 - Transformer
3. 自然语言处理应用系统

Seq2Seq

- Seq2Seq (Sequence to Sequence, 序列到序列) 模型, 由编码器 (Encoder) 和解码器 (Decoder) 两部分组成, 一般选用RNN/LSTM/GRU等网络实现。Seq2Seq的输入是一个不定长的序列, 输出也是一个不定长的序列。

- 编码器和解码器的作用:

- 编码器将长度可变的输入序列转换成长度固定的上下文向量 (context vector), 并且将输入序列的信息在该上下文向量中进行编码。
- 解码器将上下文向量进行解码, 转换成长度可变的输出序列。



Encoder-Decoder框架

- Seq2Seq的全称是sequence to sequence, 基础的seq2seq = Encoder + Decoder + 语义编码c (连接两者的中间状态向量, 可以理解成两种语言之间的联系规律)。Encoder中将可变长度的信号序列变为固定长度的向量表达, 其实本质也就是进行了特征压缩; Decoder将这个固定长度的向量变成可变长度的目标信号序列; Encoder、Decoder可以是CNN、RNN、Transformer三种结构, Encoder和Decoder可以是相同的结构, 也可以是不同的结构。
- 相同结构, 即都是RNN, 或者CNN, 或者都是Transformer; 不同结构, 比如一个是RNN, 另一个CNN (例如: Image Caption)。

Transformer: Attention is All You Need (1)

- Transformer是谷歌的研究人员提出的一种基于Seq2Seq结构的全新模型，在被提出之后，很快就席卷了整个自然语言处理领域。
- 与循环神经网络等传统模型不同，Transformer模型仅仅使用一种被称作自注意力机制的方法和标准的前馈神经网络，完全不依赖任何循环单元或者卷积操作。自注意力机制的优点在于可以直接对序列中任意两个单元之间的关系进行建模，这使得长距离依赖等问题可以更好地被求解。
- Transformer也可以当作一种表示模型，因此也被大量地使用在自然语言处理的其他领域，目前非常流行的BERT等预训练模型就是基于Transformer。

Transformer: Attention is All You Need (2)

	Seq2seq without attention	Seq2seq with attention	Transformer
Processing within encoder	RNN/CNN	RNN/CNN	attention
Processing within decoder	RNN/CNN	RNN/CNN	attention
decoder-encoder interaction	static fixed-sized vector	attention	attention

- 除了encoder-decoder之间的注意力机制，transformer在encoder编码器，decoder解码器内部还引入了自注意力机制（self-attention），更好的捕捉上下文语义，相比于传统的基于RNN网络的编码器和解码器，使用自注意力机制的编码器和解码器能够更好的捕捉上下文信息，此外RNN网络在时序上必须顺序执行，无法并行，效率较慢，自注意力机制可以并行处理，充分利用硬件进行加速。

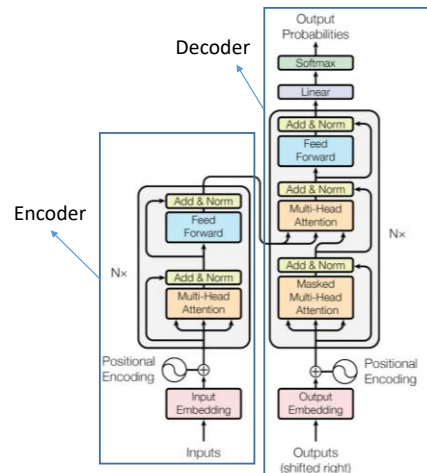
Transformer时间复杂度

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(\log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

RNN、CNN、Transformer时间复杂度对比
<https://arxiv.org/pdf/1706.03762>

Transformer架构

- Transformer由Encoder和Decoder两部分组成。
- Encoder包含N=6个相同的layer，layer指的就是图中左侧的单元。
- Decoder也包含N=6个相同的layer。



<https://arxiv.org/pdf/1706.03762>

Encoder Block中的组件 (1)

- Transformer中的Encoder和Decoder中包含的模块相似：
 - Input Embedding：将输入的数据（如文本中的单词或标记）转化为模型可以处理的数值向量形式。
 - Positional Encoding，位置编码：捕捉输入之间的顺序和位置关系。
 - Multi-Head Attention，多头注意力：由多个注意力子层构成，用于扩展了模型专注于不同位置的能力。
 - 自注意力（Self-attention）：用于学习到不同位置之间的关联性，从而捕捉到更丰富的上下文信息。
 - Feed Forward，前馈神经网络：使用全连接的前馈神经网络对输入向量序列进行进一步变换。

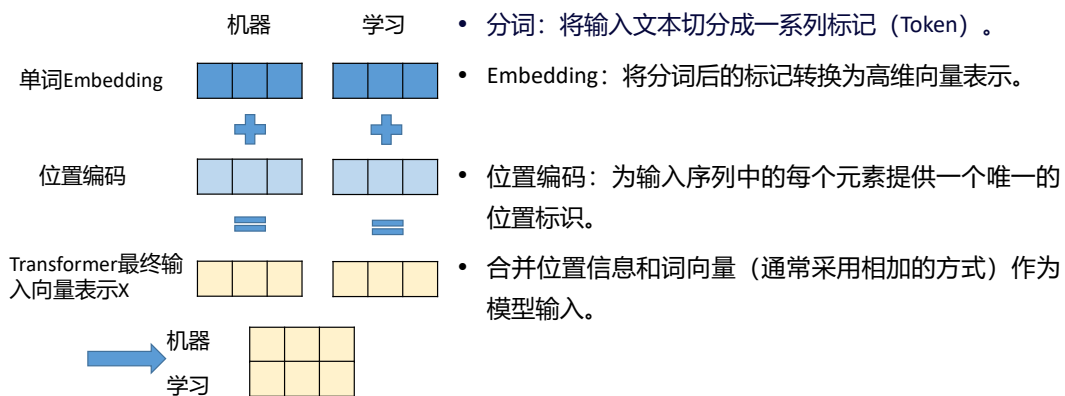
- 以上操作就构成了 Transformer 的一层，各个模块执行的顺序可以简单描述为：Self-Attention → Residual Connection → Layer Normalization → Feed Forward Network → Residual Connection → Layer Normalization。编码器可以包含多个这样的层，比如，可以构建一个六层编码器，每层都执行上面的操作。最上层的结果作为整个编码的结果，会被传入解码器。

Encoder Block中的组件 (2)

- 残差连接 (Residual Connection, 标记为 “Add”) : 对于自注意力子层和前馈神经网络子层, 都有一个从输入直接到输出的额外连接, 也就是一个跨子层的直连。残差连接可以使深层网络的信息传递更为有效;
- 层正则化 (Layer Normalization, 标记为 “Norm”) : 自注意力子层和前馈神经网络子层进行最终输出之前, 会对输出的向量进行层正则化, 规范结果向量取值范围, 这样易于后面进一步的处理。

- 以上操作就构成了 Transformer 的一层, 各个模块执行的顺序可以简单描述为: Self-Attention → Residual Connection → Layer Normalization → Feed Forward Network → Residual Connection → Layer Normalization。编码器可以包含多个这样的层, 比如, 可以构建一个六层编码器, 每层都执行上面的操作。最上层的结果作为整个编码的结果, 会被传入解码器。

Transformer的输入



Transformer位置编码

- Transformer中采用的是三角函数位置编码（绝对位置编码）。

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

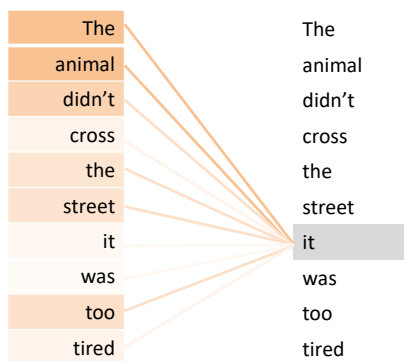
其中，pos表示单词在句子中的位置，d表示PE的维度（与词Embedding一样），2i表示偶数的维度，2i+1表示奇数维度（即 $2i \leq d$ ， $2i+1 \leq d$ ）。

- 为什么是sin和cos：对于每个位置i和每个维度d，使用正弦和余弦函数交替地生成位置编码，其中波长形成一个几何级数。这种编码方式不仅与序列中每个位置的绝对值有关，还由于三角函数的性质，使得位置a+b的向量可以表示成位置a和位置b的向量的组合，从而提供了表达相对位置信息的可能性。对于固定长度的间距k， $PE(pos+k)$ 可以用 $PE(pos)$ 计算得到。

- 与传统的可学习位置嵌入相比，三角函数位置编码不需要额外的训练参数。位置编码是通过固定的数学公式直接计算得到的，这减少了模型的参数量。
- 由于三角函数位置编码具有显式的生成规律，模型可以更容易地将其推广到未见过的序列长度上。
- 三角函数位置编码还具有一定的表达相对位置信息的能力。通过计算不同位置编码之间的差值或比例，模型可以推断出元素之间的相对位置关系。但在实际使用过程中，这种隐式的相对位置信息表达模型并不是总能学习到。

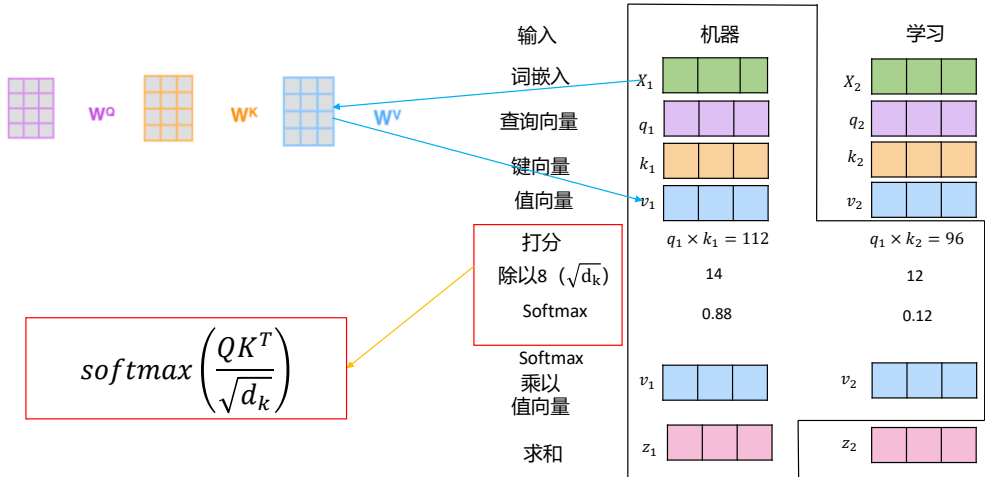
自注意力

- 编码器-解码器注意力 (Encoder-Decoder Attention) 关注的是输入和输出之间的关系, 自注意力 (Self-Attention) 关注的是输入之间的关系。

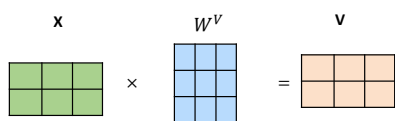
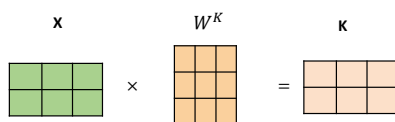
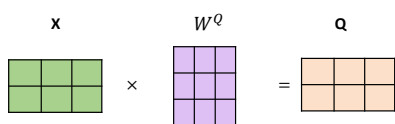


- 上图中, it 和 animal 和 tired 的关系权重比较大, 这反映出attention能够反映输入之间的一些语义关系

自注意力实现 (1)

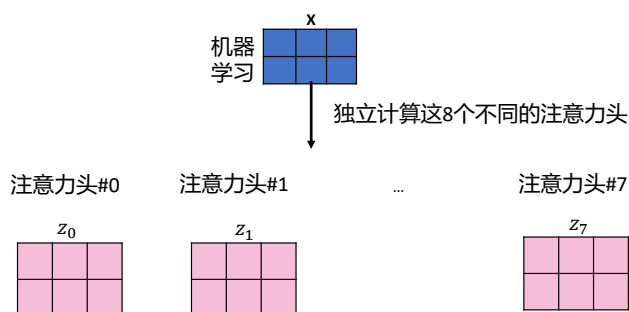


自注意力实现 (2)



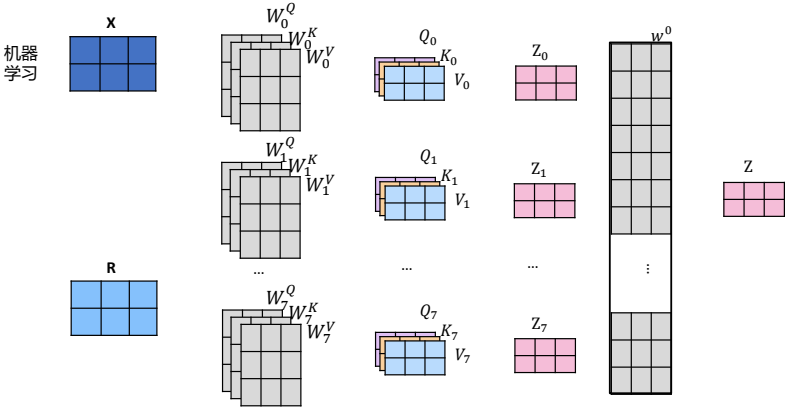
$$Attention(Q, K, V) = \text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) \times V = Z$$

多头注意力 (1)

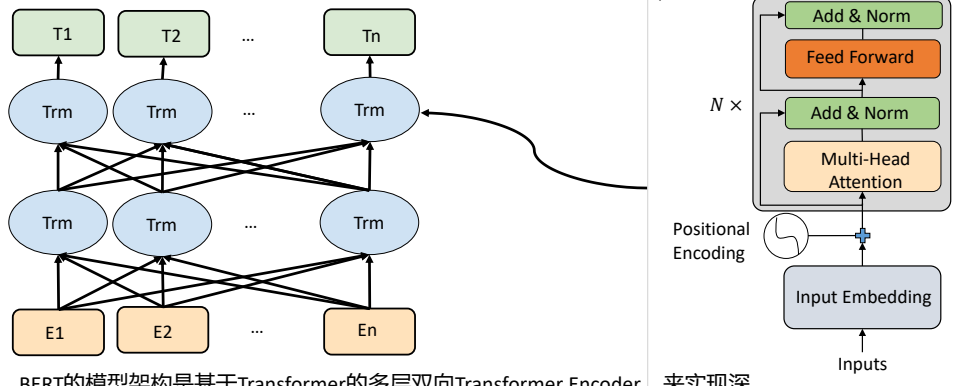


- 在“多头”注意机制下，我们为每个头保持独立的查询/键/值权重矩阵，从而产生不同的查询/键/值矩阵。和之前一样，我们拿 X 乘以 $W_Q/W_K/W_V$ 矩阵来产生查询/键/值矩阵。
- 如果我们做与上述相同的自注意力计算，只需八次不同的权重矩阵运算，我们就会得到八个不同的 Z 矩阵。
- 这给我们带来了一点挑战。前馈层不需要8个矩阵，它只需要一个矩阵（由每一个单词的表示向量组成）。所以我们需要一种方法把这八个矩阵压缩成一个矩阵。那该怎么做？

多头注意力 (2)



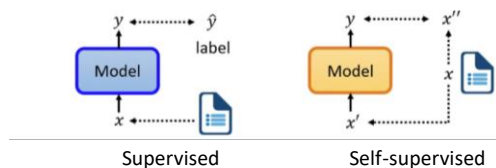
BERT



- BERT的模型架构是基于Transformer的多层双向Transformer Encoder，来实现深度双向预训练模型训练，即所有层都联合上下文语境进行预训练，较之前使用单向语言模型训练更准确，信息量更大，且语义理解更准确。

自监督学习

- 当前拥有一堆的数据，但是没有标注。将这堆数据 x 分成两个部分，一部分作为模型的输入 x' ，一部分作为模型的标签 x'' 。然后对模型进行训练，使模型地输出 y 尽可能地接近标签 x'' 。注意这里 x'' 不是人标注的，而是数据里本来就有的。



- BERT通过自监督学习的方式进行训练。在训练过程中，BERT通过对大量无标签的语料库进行学习，以预测上下文语境中下一个单词的概率分布。

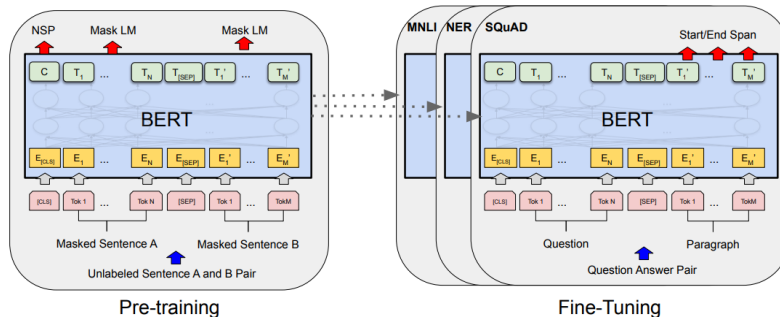
BERT Input

Input	[CLS]	my	cat	is	cute	[SEP]	he	likes	play	##ing	[SEP]
Token Embeddings	E_{CLS}	E_{my}	E_{cat}	E_{is}	E_{cute}	$E_{[SEP]}$	E_{he}	E_{likes}	E_{play}	$E_{##ing}$	$E_{[SEP]}$
	+	+	+	+	+	+	+	+	+	+	+
Segment Embeddings	E_A	E_A	E_A	E_A	E_A	E_A	E_A	E_A	E_A	E_A	E_A
	+	+	+	+	+	+	+	+	+	+	+
Position Embeddings	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}

- > **Token Embeddings**是词向量，第一个单词是CLS标志，可以用于之后的分类任务
- > **Segment Embeddings**用来区别两种句子，因为预训练不光做LM还要做以两个句子为输入的分类任务
- > **Position Embeddings**对位置信息编码，表示每个单词输入序列的位置信息

pre-train和fine-tune

- BERT包含pre-train和fine-tune两个阶段。pre-train阶段模型是在无标注的标签数据上进行训练，fine-tune阶段使用预训练BERT模型在下游任务的带标签数据上对模型参数进行微调，BERT模型首先是被pre-train模型参数初始化，然后所有的参数会用下游的有标注的数据进行训练。



<https://arxiv.org/pdf/1810.04805>

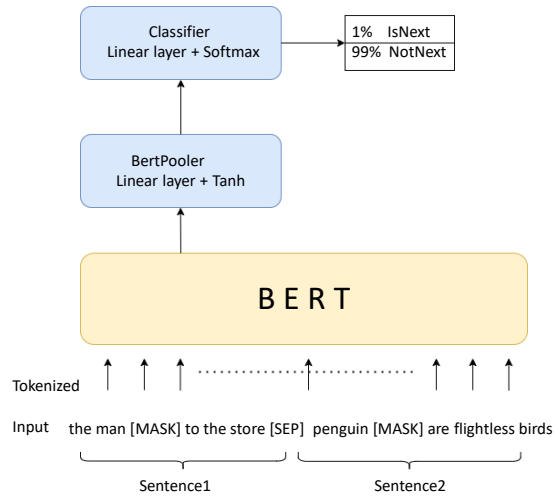
MLM (Masked Language Model)

- BERT是一个多任务模型，它的预训练（Pre-training）任务是由之前提到的两个自监督任务组成，即MLM和NSP。
- MLM是指在训练的时候随即从输入语料上mask掉一些单词，然后通过上下文预测该单词，该任务非常像我们在中学时期经常做的完形填空。
 - 80%的时候会直接替换为[Mask]，将句子 “my dog is cute” 转换为句子 “my dog is [Mask]”。
 - 10%的时候将其替换为其它任意单词，将单词 “cute” 替换成另一个随机词，例如 “apple”。将句子 “my dog is cute” 转换为句子 “my dog is apple”。
 - 10%的时候会保留原始Token，例如保持句子为 “my dog is cute” 不变。

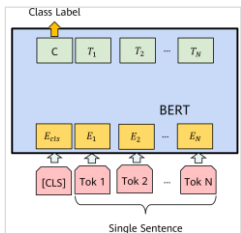
Next Sentence Prediction (1)

- Next Sentence Prediction (NSP) 是判断句子B是否是句子A的下文。
- 许多重要的下游任务，如问答 (QA) 和自然语言推理 (NLI) 都是基于理解两个句子之间的关系，而这并不能被语言建模直接捕获。为了训练一个理解句子关系的模型，BERT采用下面的NSP策略
 - 如果是的话输出IsNext，否则输出NotNext。
- 训练数据的生成方式是从平行语料中随机抽取的连续两句话，其中50%保留抽取的两句话，它们符合IsNext关系，另外50%的第二句话是随机从预料中提取的，它们的关系是NotNext的。这个关系保存在[CLS]符号中。

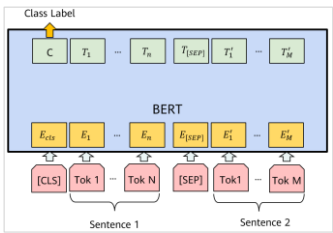
Next Sentence Prediction (2)



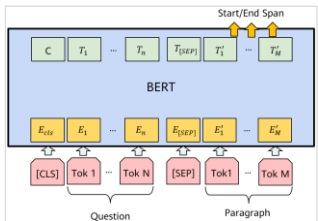
Fine-tuning



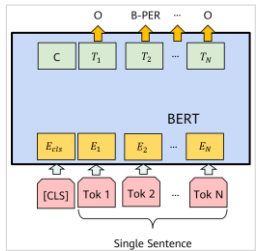
单句子分类任务



句对分类任务



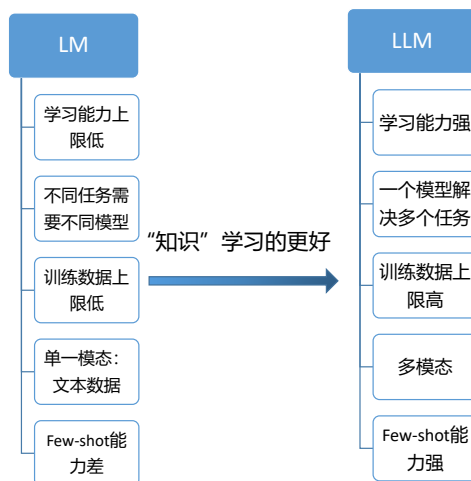
问答任务



序列标注任务

LLM

- 大语言模型（Large Language Models, 简称LLM），指那些拥有海量参数、能够处理和理解广泛语言数据集的模型。它们不仅能够进行文本生成、问答、摘要等任务，还能在特定领域展现出惊人的创造力和理解能力，比如编写代码、撰写文章甚至进行文学创作。



大语言模型的结构分类

当前LLM以Transformer为基础分为三个类别：

Decoder-only

• 适用于文本生成，对话机器人等

Encoder-only

• 适用于文本表示，文本的分类、
实体识别、关键信息抽取等

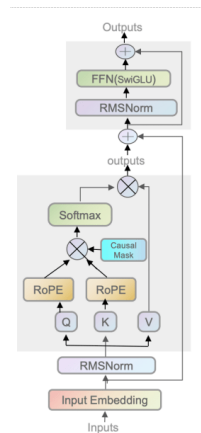
Encoder-
Decoder

• 适合做翻译、对话等需要根据给定输入来生成输出的任务。

- Decoder-only: GPT、LLaMA等
- Encoder-only: BERT等
- Encoder-decoder : T5 、 BART 、 GLM等
- 以LLaMA为例：

LLAMA结构

- LLaMA是由Meta AI发布的大语言模型（目前为LLaMA3），模型采用了Decoder-only的架构，有7B、13B、33B、60B多个版本。在公开数据集（1.4T token）上进行训练。模型结构细节如下（以7B为例）：
 - 模型整体结构由32个Block堆叠而成；
 - 文本数据处理采用了BPE；
 - 位置编码使用旋转位置编码；
 - 在每个子层中使用了RMSNorm进行归一化；
 - 激活函数使用SwiGLU。



思考题

1. （多选题）以下属于序列标注任务的是（ ）
 - A. 情感分析
 - B. 中文分词
 - C. 实体识别
 - D. 词性标注
2. （判断题）语言模型可以用于动态生成每个词的词向量（ ）
 - A. True
 - B. False

- 答案：
 - ▣ BCD
 - ▣ A

本节小结

- 词向量，Word2Vec、Glove等预训练词向量模型在文本语料中学习词（或字）的一个分布式语义表达，为下游任务提供丰富的语义特征。
- 文本分类，文本分类第一步也是最重要的一步是把整个文本编码成一个向量，传统方法有词袋向量化，TF-IDF向量化，深度学习方法里可以通过CNN、RNN、Transformer等网络来自动完成。
- 序列标注，序列标注在每一个输入位置都有相应的输出，中文分词，词性标注、命名实体识别都是典型的序列标注任务。
- 语言模型与文本生成，语言模型主要用于生成类任务。
- Transformer，序列到序列网络使用编码器-解码器架构实现一个序列到另一个序列的转换，注意力机制能够很好的解决传统RNN编码器-解码器架构的信息瓶颈问题，可以动态的生成句子的向量表达。

目录

1. 自然语言处理简介
2. 自然语言处理任务与方法
- 3. 自然语言处理综合应用系统**

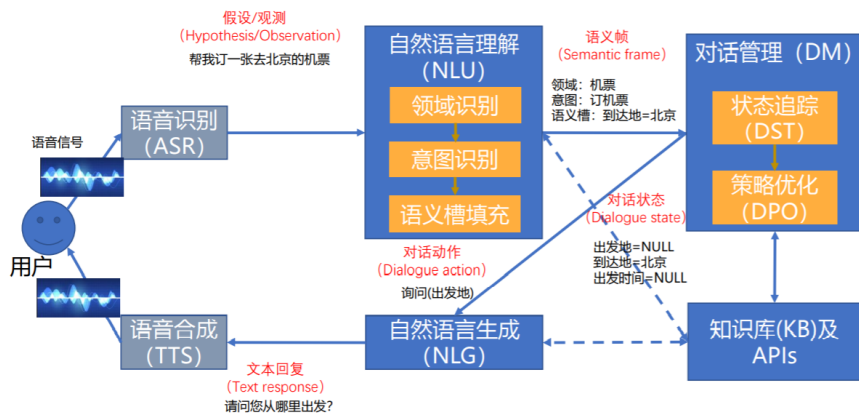
本节概述和学习目标

- 前一节介绍了自然语言处理的几个常见的任务，这一节我们通过一个综合性的对话系统，来看一个具体的自然语言处理应用系统是如何把前面讲到的这些任务串起来的。

对话系统

	任务型 Task	聊天 Chat	知识问答 Knowledge	推荐 Recommendation
目的	完成任务或动作	闲聊	知识获取	信息推荐
领域	特定域（垂直类）	开放域	开放域	特定域
以话轮数评价	越少越好	话轮越多越好	越少越好	越少越好
应用	虚拟个人助理	娱乐、情感陪护	客服、教育	个性化推荐
典型系统	Siri、Cortana	小冰	IBM Watson	阿里小蜜

任务型对话系统架构

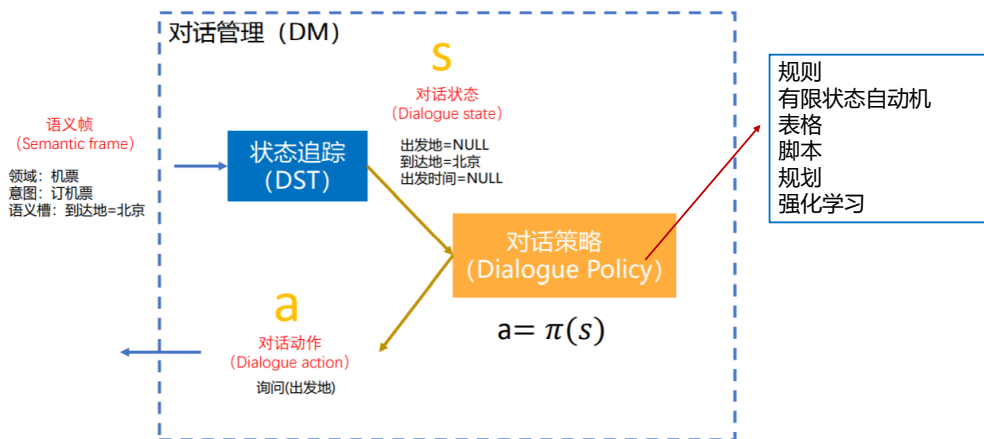


车万翔, 张伟男; 哈工大; 任务型对话系统

对话系统关键技术 – 自然语言理解

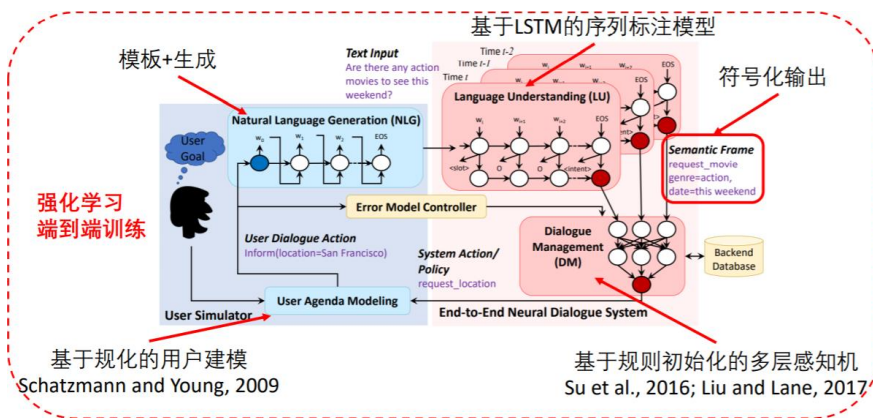


对话系统关键技术 – 对话管理



对话系统关键技术 – 端到端对话系统

- 各个部分都是独立的神经网络模型



思考题

1. （多选题）对话系统按使用目的分主要有哪几类（）
 - A. 任务型
 - B. 聊天
 - C. 知识问答
 - D. 推荐

- ABCD

本节小结

- 对话系统是自然语言处理领域里的综合性应用，涉及到自然语言理解，对话策略的生成与管理，自然语言生成等技术。

本章总结

- 本章主要介绍了自然语言处理的基础理论及其应用，包括自然语言处理定义、应用、难点、发展历史、主要任务及实现方法，最后通过对话系统的案例，带大家了解不同的任务是如何组织在一起形成一个综合的应用。

缩略语

- LLM: Large Language Model
- MLM: Masked Language Model
- ELMO: Embeddings from Language Model
- NSP: Next Sentence Prediction

学习推荐

- CS224n: Natural Language Processing with Deep Learning
 - <http://web.stanford.edu/class/cs224n/>
- 台湾大学李宏毅机器学习
 - <http://speech.ee.ntu.edu.tw/~tlkagk/courses ML19.html>

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。
Bring digital to every person, home, and
organization for a fully connected,
intelligent world.

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.



人工智能发展趋势与昇腾AI基础软硬件平台介绍



前言

- 人类正在迎来以智能技术为代表的第四次工业革命，云计算、大数据、物联网、人工智能、5G以及生物工程等新技术融入到人类社会方方面面，驱动全球宏观趋势的变化，如社会可持续发展，经济增长的新动能，智慧城市升级、产业数字化转型、消费体验等。
- 华为作为全球领先的ICT（信息与通信）基础设施和智能终端提供商，积极参与人工智能的变革，本章将对华为的昇腾AI基础软硬件平台展开介绍。

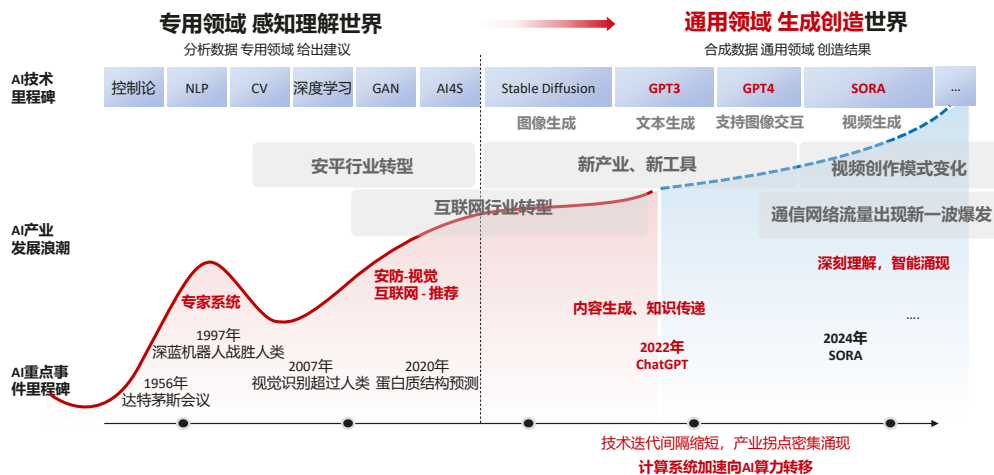
目标

- 学完本课程后，您将能够：
 - 了解人工智能现状与发展趋势
 - 了解昇腾基础软硬件平台

目录

1. 人工智能现状与发展趋势
2. 昇腾AI基础软硬件平台介绍

AI发展跨越拐点，从专用走向通用



技术的脉搏：三大因素驱动深度学习大发展

算力



无处不在的算力

智能设备无处不在，走进生活、深入行业

1 GOPS
Intel 奔腾 III PC 机



10 TOPS
智能手机

数据



爆发式增长的数据

丰富应用带来非结构化数据爆发增长

44 ZB
2020



180 ZB
2025

算法



迭代更新的模型

算法模型不断突破，日益复杂

1.1 亿
GPT-1 模型参数



2000 亿
盘古大模型参数

行业的脉搏：从AI+到+AI，AI进入场景化应用时代

AI+

探索人工智能自身能力



图像分类

EfficientNet模型 Top-5准确率
98.7%，超过人类能力（96%）



语音识别

RNN-T模型准确率96.8%，
超越人类（94.17%）



阅读理解

BERT模型准确率90.9%，
超过人类（82%）



+AI

千行百业融合AI，走向场景化应用



智能工业质检



智慧园区



自动驾驶



智慧港口



电力线智能巡检

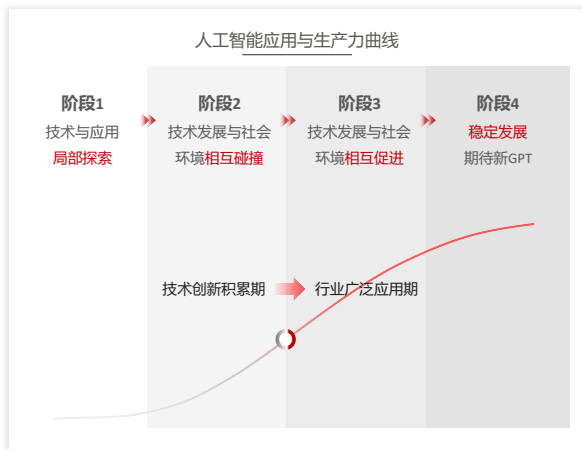


智慧网点

产业的脉搏：巨变伊始，AI将深刻推动社会发展进程

人工智能是一种新的通用目的技术（GPT）

强大算力支撑人工智能快速发展，第三次AI浪潮已经到来



目录

1. 人工智能现状与发展趋势
2. 昇腾AI基础软硬件平台介绍

昇腾AI基础软硬件平台



- <https://www.hiascend.com/>
- 华为在业界率先实现全栈和全场景AI解决方案。所谓全栈，是技术功能视角，是包括芯片、芯片使能、训练和推理框架和应用使能在内的全堆栈方案；所谓全场景，是包括公有云、私有云、各种边缘计算、物联网行业终端以及消费类终端等全场景的部署环境。
- 芯片可以说是人工智能的血液，就像血液提供给我们营养，芯片是给人工智能提供算力。
- CANN层：异构计算架构层（CPU+NPU）；就像人类的心脏，促使血液流向全身；CANN的作用相似，让算力真正地释放。
- 智能框架可以说人工智能的大脑和灵魂，也可以说智能框架是人工智能领域的操作系统。底层芯片可以寻找供应商，但系统代表生态。就像是Tensorflow，变相地给GPU带来了用户，但人家完全可以玩自己的TPU。2020年3月，华为开源MindSpore框架。
- 有了芯片，有了能使芯片发挥作用的CANN（异构计算架构），也有了自己的框架后，后面就是用应用，就像我们的手脚，通过大脑的指挥，我们不断地在完成任务；应用使能这块我们重点会给大家介绍MindX SDK，后续实验会用到ModelArts云平台。
- IDE：MindStudio，有了它，所有的开发工作都能在一个环境下完成并且支持一键部署。

Atlas系列硬件全家福

Atlas 200I DK A2
开发者套件

开箱即用 参考丰富



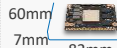
Atlas 500 A2
智能小站

宽温设计 不畏寒暑



Atlas 200I A2
加速模块

高度集成 超强解析



Atlas 300I Duo
推理卡

双芯配置 性能提升



Atlas 800T A2
训练服务器

4U高密 高能效



Atlas 300V Pro 视
频解析卡



Atlas 300V 视频解析
卡

Atlas 300I Pro 推理
卡



Atlas
推理
硬件产品

Atlas 500 智能边缘服务器
2U短机箱 适用短机柜



Atlas 800 推理服务器
2U标准 灵活配置

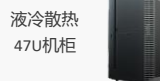


Atlas 800I A2 推理服务器
4U高密 高能效



Atlas
训练
硬件产品

Atlas 900
A2 PoD
集群单元 多柜扩展



DeepSeek与Ascend碰撞的火花

- DeepSeek-V3是由DeepSeek团队开发的一个强大的混合专家（MoE）语言模型，DeepSeek-R1模型是在DeepSeek-V3训练得到的推理模型，二者总参数量均为671B。
- 为了实现高效推理性能，Atlas 800I A2推理服务器进行了深度适配。



DeepSeek

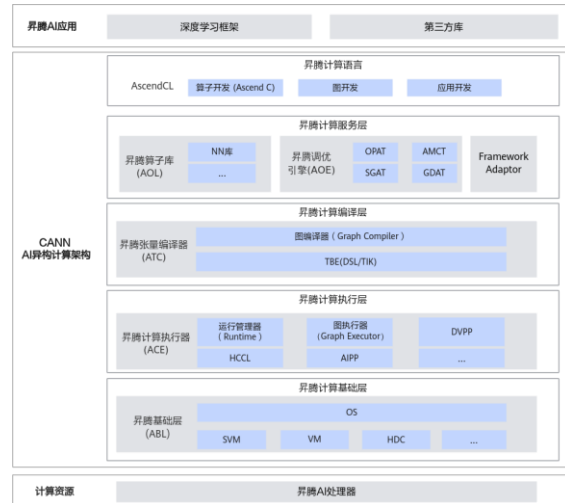
AI异构计算架构CANN（1）



- CANN：昇腾异构计算架构（Compute Architecture for Neural Networks）。
- AI训练中大部分运算为矩阵运算，因此现阶段的AI训练硬件（NPU、GPU、TPU等）都以高效进行矩阵运算为目标进行设计，而为了让开发者更好的使用自家的AI硬件，各个厂商都推出了自己的计算架构，提供高阶接口，减少开发者适配成本。
- CANN提供了从昇腾硬件底层加速到上层应用开发的全栈能力。
- 业界主流的计算架构还有NVIDIA公司的CUDA（Compute Unified Device Architecture）、AMD公司的ROCm（Radeon Open Compute platform）。

AI异构计算架构CANN（2）

- CANN是华为针对AI场景推出的异构计算架构，向上支持多种AI框架，包括MindSpore、PyTorch、TensorFlow等，向下服务AI处理器与编程，发挥承上启下的关键作用，是提升昇腾AI处理器计算效率的关键平台。
- 关键特性如下：
 - 推理应用开发
 - 模型训练
 - 算子开发



- <https://www.hiascend.com/software/cann>

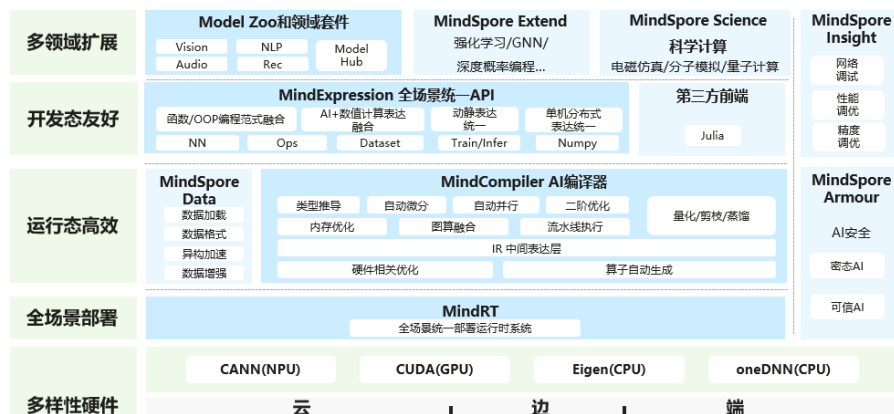
自研AI计算框架MindSpore（1）



- MindSpore是华为推出的一款开源AI计算框架，它汇聚了行业、科研和开源社区的力量。它的设计初衷是为了解决当前AI计算框架在安全性、可扩展性、易用性等方面存在的问题，并且能够支持多种硬件设备和场景。

自研AI计算框架MindSpore（2）

- MindSpore支持多种硬件，通过动静统一的编程体验和分布式并行支撑大模型训练。



- <https://www.mindspore.cn/>

AI应用使能套件，加速人工智能应用创新（1）

- MindCluster（AI集群系统软件）是支持NPU（昇腾AI处理器）训练和推理硬件的深度学习组件，使能构建集群全流程运行，提供NPU集群作业调度、运维监控、故障恢复等功能，为上层模型训练、模型评估、模型部署、模型推理等应用提供底层软件支持。
- MindEdge Framework是一款定位为被集成的轻量化端边云协同使能框架。MindEdge Framework作为昇腾推理解决方案中的一部分，用于智能边缘设备使能，提供边缘节点管理和智能推理业务（容器应用）管理的功能。
- MindSpeed：基于昇腾硬件的训练加速库，提供多维度加速算法，支持主流大语言、多模态大模型。面向大模型全流程加速，降低开发门槛，获得最优开箱性能。
- MindIE：昇腾推理引擎，基于昇腾硬件的运行加速、调试调优、快速迁移部署的高性能深度学习推理框架，分层开放满足各类需求，统一接口使能极简开发，沉淀能力构筑极致性能。

- 深度学习平台开发厂商可以减少底层资源调度相关软件开发工作量，快速使能合作伙伴基于MindCluster开发深度学习平台。
- 可通过MEF Edge和MEF Center进行边云协同管理，用户可通过二次开发，对接ISV（Independent Software Vendor）业务平台，集成所需功能。
 - MEF Edge部署在智能边缘设备上，负责与中心网管对接，完成智能推理业务（容器应用）的部署和管理，为算法应用提供服务。在线提单
 - MEF Center部署在通用服务器上，负责对边缘节点实现批量管理、业务部署和系统监测。

AI应用使能套件，加速人工智能应用创新（2）



- 再往上走，就是应用使能层，这一块主要针对工业领域。比如我是车企或是电力公司的工程师，AI这块可能不是特别专业，那使用华为提供的SDK和应用案例可以快速开发并部署人工智能应用。已经上线的有视频分析、智能制造这块的应用；后续像OCR、客服、语音等应用也会发布。
- 支持C++和Python。
- mxIndex是基于Faiss开发的昇腾NPU异构检索加速框架，针对高维空间中的海量数据，提供高性能的检索，采用与Faiss风格一致的C++语言结合TBE算子开发，支持ARM和x86_64平台。特征检索支持的检索库类型分为小库搜索（全量检索）和大库搜索（近似检索），小库规模通常在30万~100万条的量级，而大库规模可达到千万甚至亿级别，覆盖支持的特征向量维度从64维到512维向量等（具体算法略有不同）。各算法底层通过Ascend平台进行加速的TBE算子来进行实现。
- mxRec作为面向互联网市场搜索推荐广告的应用使能SDK产品，对于搜索推荐广告模型训练的应用场景需求，提供基于昇腾平台的搜索推荐广告框架，支撑大规模搜索推荐广告场景，助力完成搜索推荐广告模型的高效训练。

华为云ModelArts，面向AI开发的一站式平台



- 首先，我们来了解一下AI应用的开发流程是什么样子的。首先需要采集数据；数据采集好之后，需要对数据进行处理，比如筛选数据、给数据打标签、做特征工程等；数据处理完成之后，编码实现算法，然后训练模型；然后对生成的模型进行精度评估，评估是否满足精度要求；最后，将满足精度要求的模型部署到生产环境中。这就是AI应用的开发流程。
- ModelArts就是围绕这一套AI开发流程打造的一站式AI开发平台。用户可以基于ModelArts定制开发AI模型。ModelArts的能力覆盖AI开发的各个环节，接下来我们就介绍一下ModelArts在AI开发各个环节中的能力。

思考题

1. （多选题）以下哪些选项属于昇腾AI基础软硬件平台？
- A. Atlas系列硬件
 - B. AI异构计算架构CANN
 - C. AI框架MindSpore
 - D. 应用使能MindX

- 答案：ABCD

本章总结

- 本节主要介绍了人工智能概览及其发展趋势，并阐述了华为昇腾AI基础软硬件平台，包括其中涉及的硬件产品和软件架构以及工具链。

缩略语

- NLP: Natural Language Processing, 自然语言处理
- CV: Computer Vision, 计算机视觉
- GAN: Generative Adversarial Network, 生成对抗网络
- GPT: General Purpose Technology, 通用目的技术
- CANN: Compute Architecture for Neural Networks, 神经网络计算架构
- DVPP: Digital Vision Pre-Processing, 数字视觉预处理

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。
Bring digital to every person, home, and
organization for a fully connected,
intelligent world.

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.



ModelArts概览



前言

- 本章主要介绍ModelArts的整体方案、核心模块、成功实践。

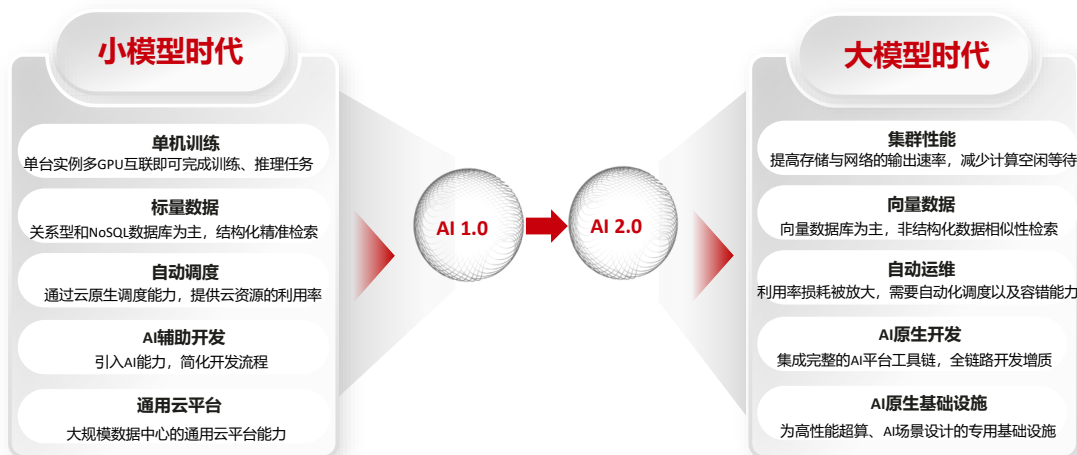
目标

- 学完本课程后，您将能够：
 - 了解ModelArts的产品定位、核心技术
 - 掌握ModelArts的数据处理、算法开发、模型训练、模型管理、模型部署和AI Gallery等核心模块的使用方法
 - 了解ModelArts的成功实践

目录

1. ModelArts**概览**
2. ModelArts**功能**
3. ModelArts**应用**

人工智能从小模型向大模型演进，AI平台面临新挑战



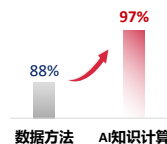
智能新范式：云原生AI开发与交付，赋能全行业AI普惠

- AI自身可解释性差
- 行业有用数据集少，采集困难
- 行业对AI攻击、隐私、安全担忧

数据模型
(Model Free)



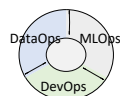
数据模型+机理模型
(Model Based+ Model Free)



- 数据，模型，应用开发流程割裂
- 多人多系统联合，配合与协作工作量大
- 围绕Model-Centric，并非Data-Centric

分段开发

数据清洗 ▶ 模型开发 ▶ 模型应用



协同迭代



交付周期
月级 >> 天级

- 碎片化作坊式开发，模型泛化性差
- 从0开始，定制化开发，模型开发周期月
- 断层式开发，没有连续版本迭代

手工作坊
(小模型优化)



工业化生产
(千亿规模参数，预训练大模型)

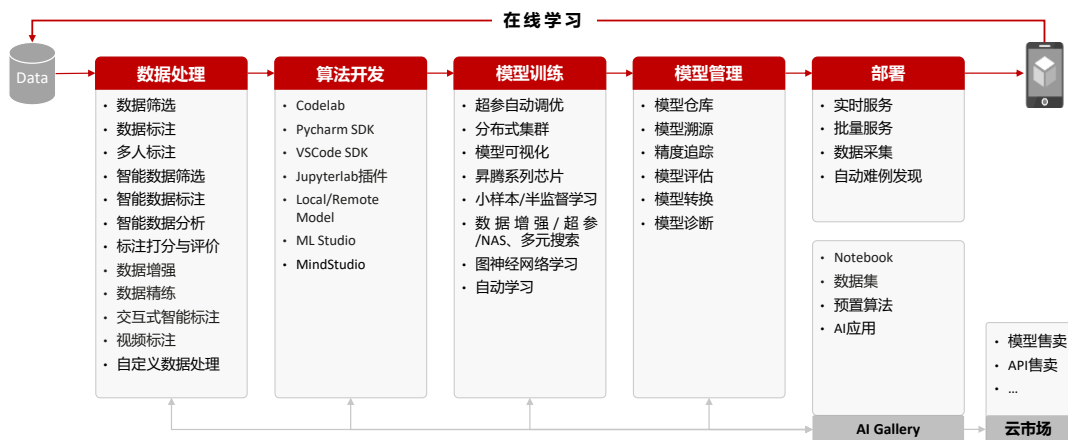


模型精度
提升 2-5%

华为云ModelArts系统架构图



华为云ModelArts，面向开发者的一站式AI开发平台



- 这是华为云一站式AI开发平台 ModelArts的功能组件，我们通过一个典型的深度学习AI模型开发流程把它打开，可以看到从数据处理、算法开发、模型训练、模型管理和部署，我们都有很丰富的工具组件提供支持。

目录

1. ModelArts概览

2. ModelArts功能

■ 数据平台

▫ 开发平台

▫ 训练平台

▫ 推理平台

▫ AI Gallery

3. ModelArts应用

数据平台：全流程的数据准备、数据处理和数据标注能力

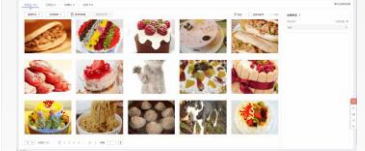
数据处理



多样的数据处理能力

- 数据清洗
- 数据选择
- 数据校验
- 数据增强

数据标注



多样的数据标注场景

- 图片标注：图像分类、物体检测、图像分割
- 文本标注：文本分类、命名实体、文本三元组
- 音频标注：声音分类、语音内容、语音分割
- 视频标注

资产管理



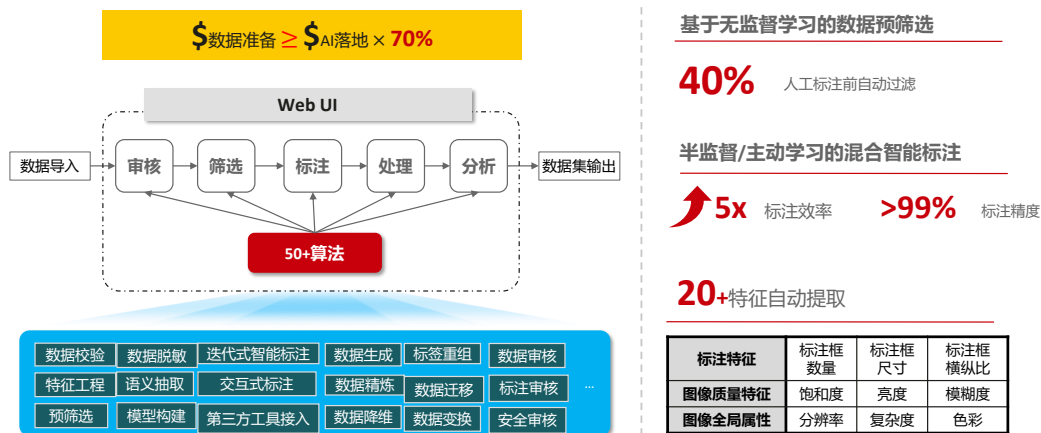
多样的数据格式

- 图片、文本、音频、视频、表格

多样的资产管理

- 版本管理
- 导入导出
- 数据分析

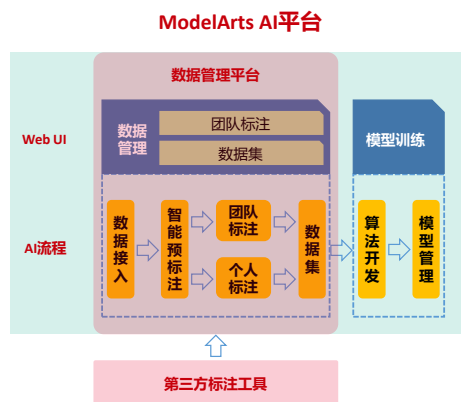
数据处理：50+算法，提升数据处理效率



- 大家都知道，深度学习中，数据量越大，模型精度会越高。而且目前落地的算法主要是有监督算法，这就需要给数据打标签。
- 当数据量巨大的时候，可以使用智能标注来提高标注效率，节省人力。
- ModelArts智能标注的原理是使用主动学习算法，给数据自动打标签，辅以人工纠正，从而大幅度提升标注效率。
- 数据集准备好之后，用户可以使用数据特征统计功能来评估数据集是否是一个好的数据集。ModelArts自动提取一些特征，然后可视化展示出来，为用户提供数据优化的建议。比如，对于图像数据集，会自动提取标注框数量、标注框尺寸、亮度、模糊度、分辨率、色彩等特征。

数据标注：格式齐全、支持多人协作在线标注

- 全数据格式支持
 - 提供5种数据类型的深度支持
 - 图像、音频、视频、文本、表格
 - 支持自定义格式数据
- 团队标注支持
 - 在线任务分发
 - 在线审核验收
 - 支持超大规模标注



- 在数据处理环节，ModelArts提供在线标注平台。ModelArts支持多种类型的数据标注，比如图像、文本、语音等多种数据类型。
- 当数据量比较大的时候，还可以使用团队标注功能，让多人协同标注一个大数据集。

目录

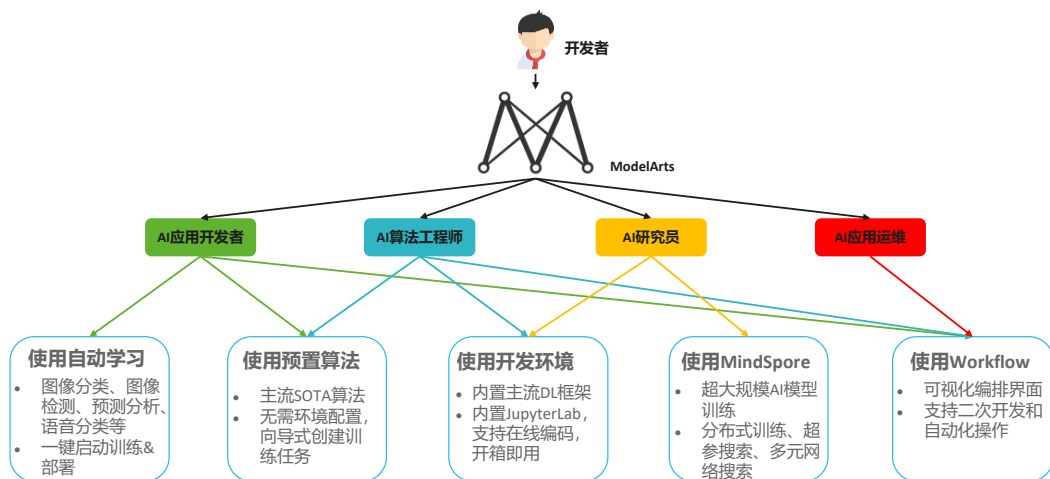
1. ModelArts概览

2. ModelArts功能

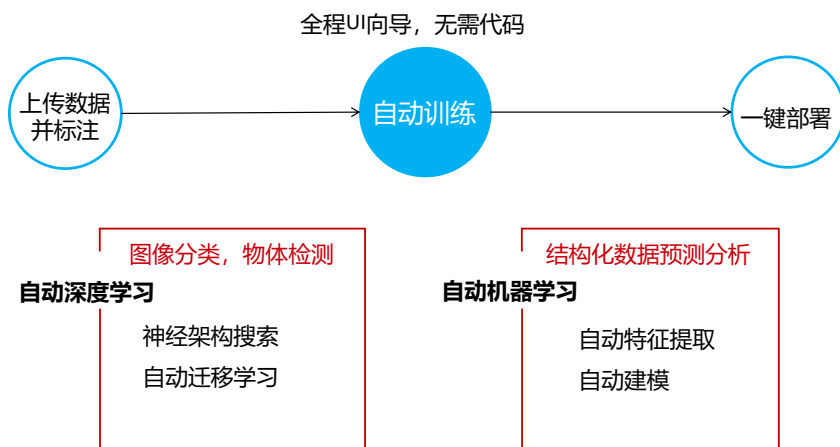
- 数据平台
- 开发平台
- 训练平台
- 推理平台
- AI Gallery

3. ModelArts应用

算法开发：面向四类开发人员提供AI开发工具



自动学习，零AI基础构建模型



- 对于无AI背景，甚至不会编码的人员，是否可以使用ModelArts开发AI模型呢？答案是可以，使用ModelArts自动学习就能做到。
- 自动学习功能对于AI小白和初学者比较友好，零编码，零AI经验，三步就可以创建AI模型。第一步，上传并标注数据；第二步，模型自动训练；第三步，模型一键部署上线。
- 自动学习会自动搜索最佳的神经网络架构，超参数，并自动做数据增强，无需用户编码和调参。门槛非常低。

预置模型，让训练更简单（1）

算法覆盖领域：

- 图像分类 →
- 目标检测
- 图像分割
- 文字识别
- 文本分类
- 声音分类
- 搜索推荐
- 时序预测
- 强化学习

模型列表

请输入搜索内容

ModelArts训练

支持模型部署

综合排序

图像分类-ResNet50-EI-Backbone

M ModelArts 3122 52

华为云EI骨干模型 泛化能力更强, 支持CPU, GPU推理

图像分类-ResNet_v1_50(Ascend)

M ModelArts 2903 18

基于TensorFlow, Ascend910训练, Ascend310推理

图像分类-ResNet_50

M ModelArts 2537 17

新型多尺度50层残差神经网络结构

图像分类-EfficientNetB0

M ModelArts 2166 5

支持Pytorch, GPU训练, 支持CPU, GPU推理

图像分类-Inception_Resnet_V2

M ModelArts 2308 10

Inception和ResNet的结合, 模型较大, 准确率更高, 支持CPU, GPU, Ascend-310推理

图像分类-Inception_v3

M ModelArts 2265 8

Inception系列支持CPU, GPU和Ascend310推理

- ModelArts将业界一些流行的AI算法进行实现，并基于开源或者行业数据集训练模型，并进行模型的精度调优和推理速度优化，然后沉淀到ModelArts中，就形成了ModelArts预置算法。
- 目前已经支持多个领域的预置算法，包括ResNet50、SSD、FasterRCNN、YOLOv3、deeplab、声音分类、Bert等。
- 预置算法的使用也很简单，无需编码。但是用户自己可以调节超参数，并灵活挑选算法。适合懂一些AI原理的用户来使用。

预置模型，让训练更简单（2）



- 右侧实验截图
- 当准备好数据和算法后，就进入到模型训练阶段。向导式界面引导，以友好的方式一步一步指导用户如何创建训练作业。
- 第一步是选择训练算法，既可以使用预置算法，也可以使用自己实现的AI算法。第二步，选择准备好的数据集。第三步，配置超参数，比如Batch size、learning rate、epoch等。第四步，选择训练资源，CPU、GPU还是Ascend芯片，还有计算节点的数量，单机还是分布式。这种方式将复杂的AI训练作业创建过程，通过界面化操作和引导来简化，让建模过程更清晰。

开发环境：CodeLab，让AI探索&教学更自由

- 案例内容秒级运行与分享
- Serverless化实例管理，资源自动回收
- 免费算力规格

jupyter

关键价值

即开即用

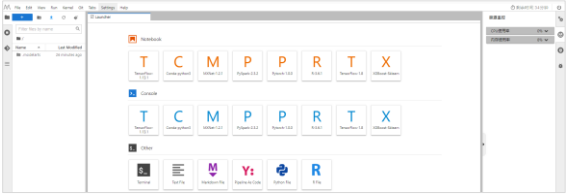
丰富案例

一键运行

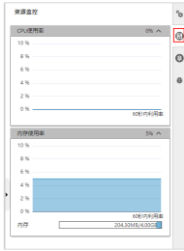
免费AI算力

分享与探索

CodeLab首页



切换规格



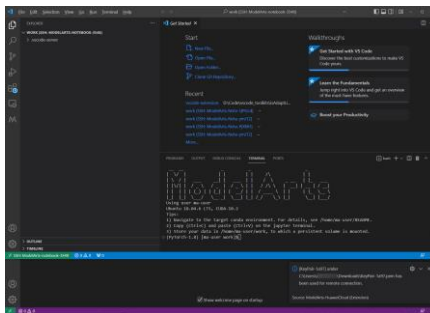
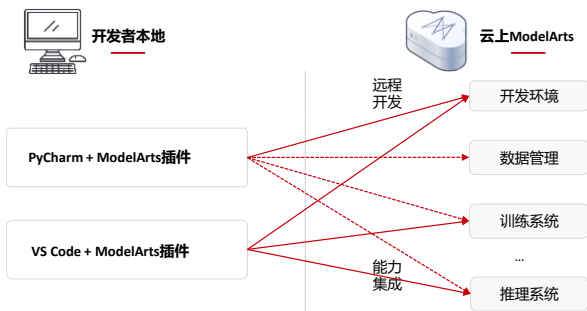
资源监控

开发环境：IDE SDKs，云上云下，无缝协同

- 代码开发与调测。本地IDE+ModelArts插件**远程开发能力**。贴近开发人员使用习惯
- 云上开发环境，包含AI计算资源，云上存储，预制AI引擎
- 运行环境自定义。将开发环境直接保存成为镜像，供训练、推理使用

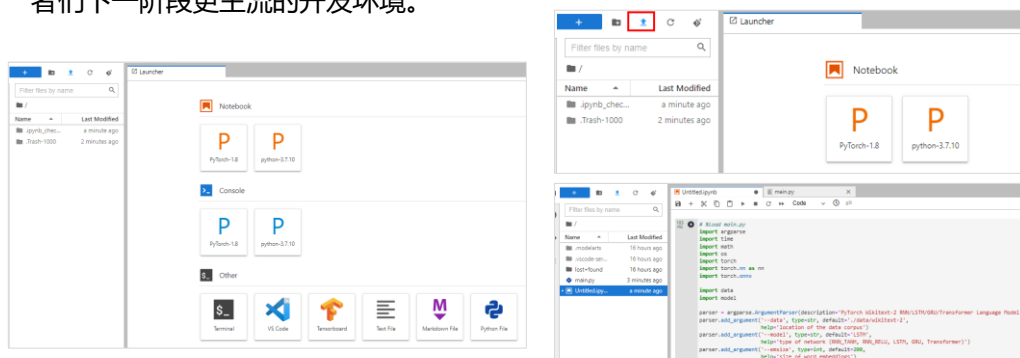
关键词

ModelArts-IDE 远程开发
多人协同 灵活AI计算资源 开发高效



开发环境：JupyterLab，在线开发和调试代码

- JupyterLab是一个交互式的开发环境，可以使用它编写Notebook、操作终端、编辑MarkDown文本、打开交互模式、查看csv文件及图片等功能。可以说，JupyterLab是开发者们下一阶段更主流的开发环境。



- 对于想要自己编码实现AI算法，并训练模型的用户，ModelArts提供在线的交互式编码调试环境Jupyter Notebook。ModelArts Notebook支持Python编程语言，底层支持多种硬件，包括CPU、GPU、还有华为自研的昇腾AI芯片。
- 在AI引擎方面，支持常用的AI引擎，TensorFlow、PyTorch、Spark Mllib、sklearn、XGBoost等，还有华为自研的MindSpore。
- 用户可以在Notebook里面一步一步执行并调试AI算法，适合初学者学习AI理论过程中使用。

Workflow：MLOps工作流，定义AI流程标准化

客户问题

- 开发流程零散、不够规范，导致模型更新流程时间太长，需要一周甚至一个月
- 对现网服务没有可靠保护，简单验证之后直接发布替换

业务价值

- 模型自动化更新，提升效率
- 引入灰度发布，保证发布模型版本的可信可靠，可及时回退
- 人员提效：工程算法人员可以释放出来，运营一键运行维护

解决方案

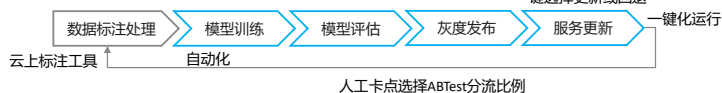
原有部署方式

算法工程师全过程参与



部署流水线Workflow

运营全过程负责



目录

1. ModelArts概览

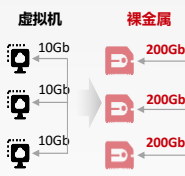
2. ModelArts功能

- 数据平台
- 开发平台
- 训练平台
- 推理平台
- AI Gallery

3. ModelArts应用

训练平台：便捷开发，高性能分布式，智能日志分析

多机分布式裸金属训练



高性能

- 基于裸金属技术，每卡100Gb带宽，无虚拟化损失，并行训练效率提升100%

场景

- 行业基础大模型训练
- 图像、音频等深度学习场景训练加速

训练事件和资源监控



资源监控

- CPU资源监控
- 内存资源监控
- GPU/NPU资源监控
- GPU/NPU内存资源监控

事件监控

- 作业启动、排队、运行、结束

分布式日志管理



训练日志

- 实时查看训练日志
- 关键字检索
- 高级检索(正则、全字...)
- 错误/异常日志高亮
- 日志单个/批量下载

故障容错



高可靠

- 作业自动重启
- 作业卡死检测
- 智能失败分析
- 故障节点自动隔离
- 断点续训

- 实验截图
- 在模型训练过程中，ModelArts会通过可视化的方式多维度展示训练过程和结果，让训练过程和质量更可控。训练作业配置信息，会将训练作业的详情记录下来，比如使用的数据集、算法、资源规格、超参数等，方便以后查看。CPU、GPU和内存的利用率，展示资源的使用情况；训练日志可以实时查看，会显示训练过程中标准输出打印的日志；TensorBoard可视化展示模型训练过程中的精度指标曲线；模型评估通过多个指标展示模型的精度，并给出模型调优的建议；模型版本比对，比对多个版本模型的精度，辅助用户挑选最优的模型。

资源调度：超大规模异构算力管理&调度，提升利用率



价值特性

- CPU, GPU, NPU等异构资源虚拟池化, 统一管理
- 支持资源混部及优先级调度能力, 资源利用率提高50%
- 支持训练作业拓扑, 性能提升30%

目录

1. ModelArts概览

2. ModelArts功能

- 数据平台
- 开发平台
- 训练平台
- 推理平台
- AI Gallery

3. ModelArts应用

推理平台：统一纳管，多模式滚动升级，多规则灰度发布

AI应用仓库

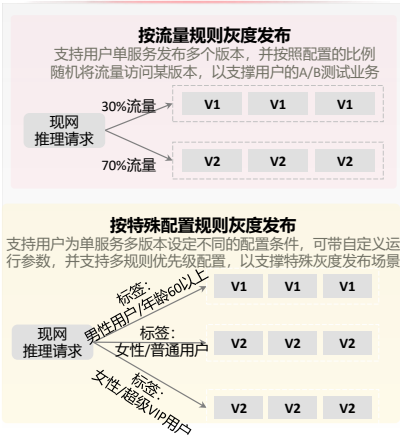


- 支持多厂商多框架多功能的AI算法统一管理
- 支持AI模型以Docker镜像的形式发布推理服务，有效保护三方模型资产
- 支持算法以模型文件+AI引擎的形式直接发布推理服务，模型上线更灵活更高效
- 支持用户自定义AI引擎，以适应用户对于AI引擎有特殊能力要求的场景
-

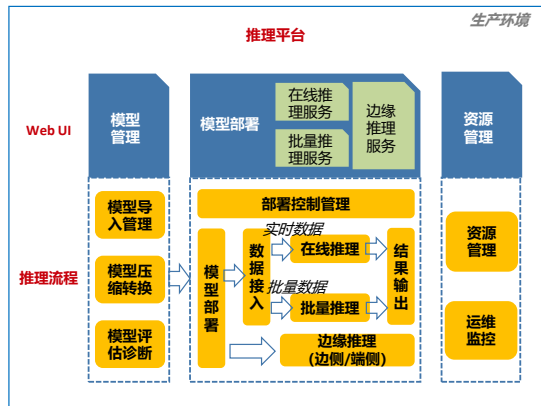
AI应用多模式滚动升级



AI应用多规则灰度发布

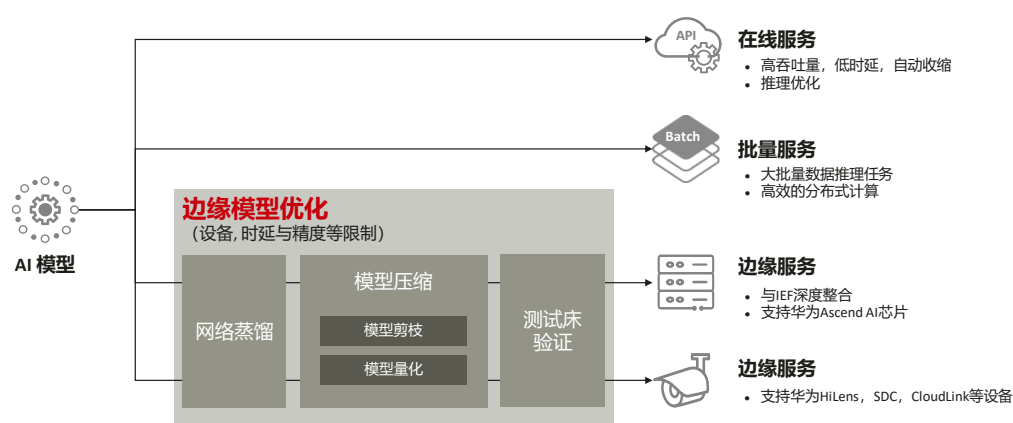


推理平台架构，支持AI应用运行态全生命周期管理



- **统一管理**
 - 支持多厂商多框架多功能模型统一纳管
 - 部署支持高并发、低延时访问，并支持弹性伸缩，灰度发布、滚动升级
- **灵活部署**
 - 支持各种部署场景，既能部署为云端的在线推理服务和批量推理任务，也能部署到端，边等各种设备
 - 大模型在线推理及模型秒级快速更新，适配快速迭代的业务场景
- **性能提升**
 - 通过自研推理框架，屏蔽底层软硬件差异，提升推理运行性能
- **迭代式模型更新**
 - 支持数据采集/难例挖掘，自动识别难例，快速适应数据变化

模型部署：端、边、云全场景AI部署



- 模型训练好之后，就可以部署到生产环境中使用了。ModelArts上训练生成的模型可以非常方便地部署到端边云等各个场景中。模型可以一键部署到云上，即在线服务，在线服务以RESTful API的方式对外提供服务，用户可以通过HTTP请求来远程调用API。也可以部署到边缘设备上，比如X86服务器；还有华为的Atlas设备，比如Atlas500智能小站、智能开发板Atlas 200 DK，深度学习摄像机HiLens Kit等；还有智能摄像头SDC等。

目录

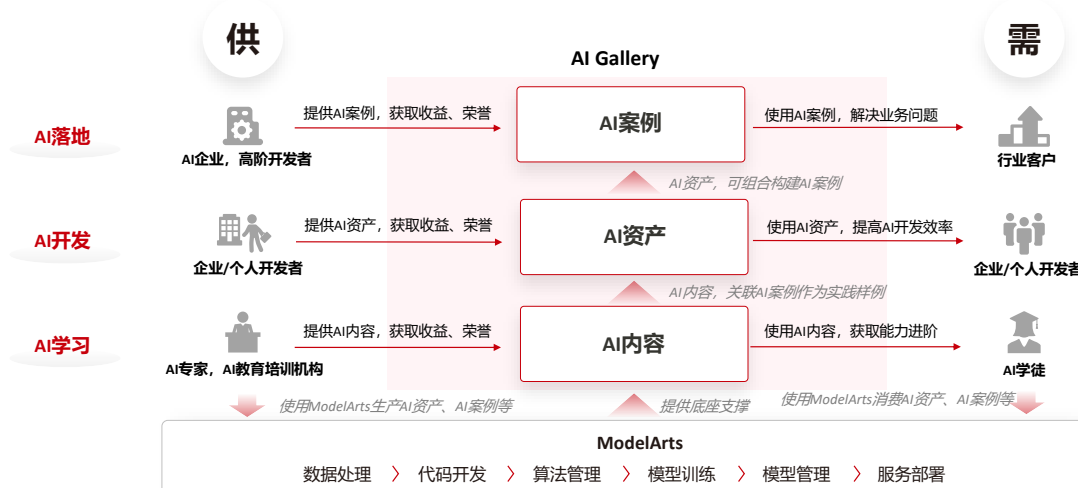
1. ModelArts概览

2. ModelArts功能

- 数据平台
- 开发平台
- 训练平台
- 推理平台
- AI Gallery

3. ModelArts应用

AI Gallery：构筑AI供需桥梁，共建共享AI生态社区



- 最后，我们了解一下AI市场。AI市场是一个交易、分享和学习的地方。里面有AI开发流程中各个环节的产物，比如高级API、数据集、算法、模型、镜像等，用户可以拿来直接使用；还有模型应用库，比如HiLens技能、智能摄像机HoloSens应用；还有知识库，这里面有ipynb笔记本、AI论坛、AI大赛等。
- AI市场的分享功能，可以显著减少开发者重复的研发工作，不让开发者重复造轮子。交易功能，让开发者有渠道可以商业变现，类似苹果的App Store。学习功能，让AI开发者有数据集可以用、有案例可以学、有AI比赛可以参加。
- 通过提供托管与分享AI数字资产的能力，开发者借助可复用的数字资产，减少50%以上重复的研发工作（30%检索，20%学习）。
- 让开发者有开源数据集可用，有notebook开发案例可学习，有AI挑战赛可以参加/自助举办，有委托任务可以领取/发布，有渠道可以商业变现。

华为云ModelArts AI Gallery, “知识+实训” 的AI开发社区



目录

1. ModelArts概览
2. ModelArts功能
- 3. ModelArts应用**

ModelArts是华为云AI服务使用的开发平台



人脸识别

人脸闸机、识别



OCR

单据/车牌/证件识别



以图搜图

商品搜索、版权保护



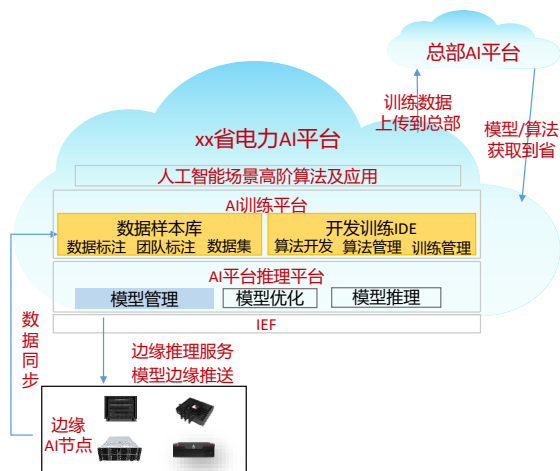
物体检测

河道漂浮物检测

ModelArts



电力+ModelArts：端边云结合的AI方案



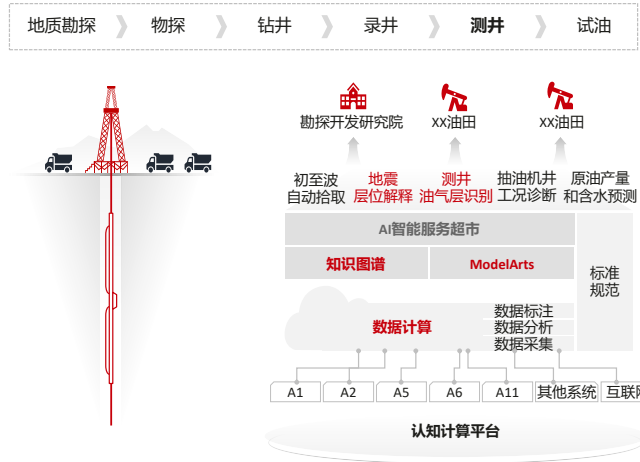
平台概述：

- 全省统一的训练平台、推理平台
- 集团级和省级AI平台互通
- 提供边缘侧部署能力

客户价值：

- 云端互通，样本库、算法持续同步
- 提供了一站式便捷的AI研究平台
- 端边云协同，提升国网安全、高效、集约运营的能力
- 边缘数据实时回收，形成数据闭环

能源+ModelArts：认知计算平台，提升油气识别效率



业务价值

识别评价时间 **70%+** ↓
识别准确率达 **测井解释专家水平**

油气层的识别准确度 **90%+**
普通专家承担 **资深专家的职责**

思考题

1. （多选题）ModelArts涵盖以下哪些模块？
- A. 数据处理
 - B. 算法开发
 - C. 模型训练
 - D. 模型部署
 - E. 数据治理

- 答案：ABCD

本章总结

- 本章介绍了华为云一站式AI开发平台ModelArts，包括ModelArts的主要功能和使用方法，最后也介绍了基于ModelArts的一些应用场景。

缩略语

- CPU: Central Processing Unit, 中央处理器
- GPU: Graphic Processing Unit, 图形处理器
- NPU: Neural Processing Unit, 神经网络处理器
- TCO: Total Cost of Ownership, 总体拥有成本
- IaaS: Infrastructure as a Service, 基础设施即服务
- ECS: Elastic Cloud Server, 弹性云服务器
- CCE: Cloud Container Engine, 云容器引擎
- NAS: Neural Architecture Search, 神经架构搜索
- API: Application Programming Interface, 应用程序接口

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。
Bring digital to every person, home, and
organization for a fully connected,
intelligent world.

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.



大模型深度学习算法基础



前言

- Transformer是2017年的一篇论文《Attention is All You Need》提出的一种模型架构，这篇论文里只针对机器翻译这一种场景做了实验，全面击败了当时的机器翻译各个benchmark上的SOTA。
- Transformer采用了Attention机制，因此本章从Attention机制的原理开始讲解，再详细介绍了Transformer的重要细节。

目标

- 学完本课程后，您将能够：
 - 描述Attention机制的原理
 - 描述Seq2Seq模型的结构
 - 描述Transformer工作流程
 - 描述Transformer中的位置编码机制
 - 描述Transformer中的Encoder-Decoder架构
 - 描述Transformer中的Self-Attention机制

目录

1. Attention机制介绍

- 为什么要引入Attention

- Attention机制

2. Transformer详解

3. DeepSeek算法解析及工程优化

为什么要引入Attention?

- Attention：注意力。
- 当我们用CNN模型识别图像时，如何让模型知道图像中不同局部信息的重要性？
- 当我们用RNN模型去处理NLP相关任务时，长距离“记忆”问题如何解决？



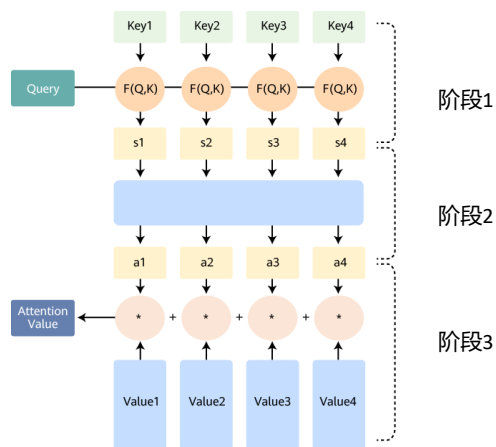
Encoder: The encoder is composed of a stack of $N = 6$ identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. We employ a residual connection [11] around each of the two sub-layers, followed by layer normalization [11]. That is, the output of each sub-layer is $\text{LayerNorm}(x + \text{Sublayer}(x))$, where $\text{Sublayer}(x)$ is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension $d_{\text{model}} = 512$.

Decoder: The decoder is also composed of a stack of $N = 6$ identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization. We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i .

- 视觉注意力机制是人类大脑的一种天生的能力。当我们看到一幅图片时，先是快速扫过图片，然后锁定需要重点关注的目标区域。比如当我们观察上述图片时，注意力很容易就集中在了北京大学等位置。试想，如果每个局部信息都不放过，那么必然耗费很多精力，不利于人类的生存进化。同样地，在深度学习网络中引入类似的机制，可以简化模型，加速计算。
- Attention是挑重点，就算文本比较长，也能从中间抓住重点，不丢失重要的信息。图中底部有颜色的文字就是被挑出来的重点。

Attention机制计算步骤

- 3步分解：
 - 第一步：query和key进行相似度计算，得到权值；
 - 第二步：将权值进行归一化，得到直接可用的权重；
 - 第三步：将权重和value进行加权求和。



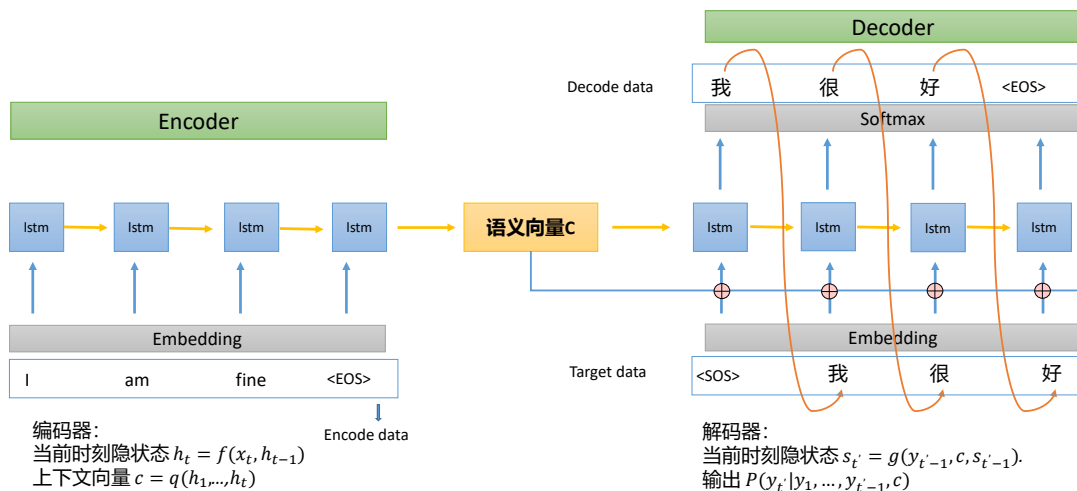
以机器翻译为例理解Attention

- 机器翻译是利用计算机将一种自然语言（源语言）转换为另一种自然语言（目标语言）的过程。
- 神经网络机器翻译通常采用编码器-解码器结构，实现对变长输入句子的建模。编码器实现对源语言句子的“理解”，形成一个特定维度的浮点数向量，之后解码器根据此向量逐字生成目标语言的翻译结果。
- Seq2Seq就是通用的编码器-解码器结构，常用于机器翻译、文本摘要、聊天机器人、阅读理解、语音识别、图像描述生成、图像问答等场景。

Seq2Seq模型

- Seq2Seq (Sequence to Sequence, 序列到序列) 模型, 由编码器 (Encoder) 和解码器 (Decoder) 两部分组成, 一般选用RNN/LSTM/GRU等网络实现。Seq2Seq的输入是一个不定长的序列, 输出也是一个不定长的序列。
- 编码器和解码器的作用:
 - 编码器将长度可变的输入序列转换成长度固定的**上下文向量 (context vector)**, 并且将输入序列的信息在该上下文向量中进行编码。
 - 解码器进行相反的过程, 将上下文向量进行解码, 转换成长度可变的输出序列。

编码器和解码器工作原理



Seq2Seq模型的问题与优化

- Seq2Seq模型在处理过长序列数据时表现不佳，其原因主要在两方面：
 - RNN模型固有的长期依赖问题（可以用LSTM/GRU解决）。
 - 上下文向量（context vector）无法完全表达整个输入序列的信息。
- 解码器在解码时，需要考虑所有序列，而其中夹杂了许多无关的噪声或信息，容易出现信息过载的问题。其实大部分输出，只跟输入序列的部分序列有关，而非整个序列。因此基于Seq2Seq模型，引入了**注意力机制（attention mechanism）**。

目录

1. Attention机制介绍

- 为什么要引入Attention

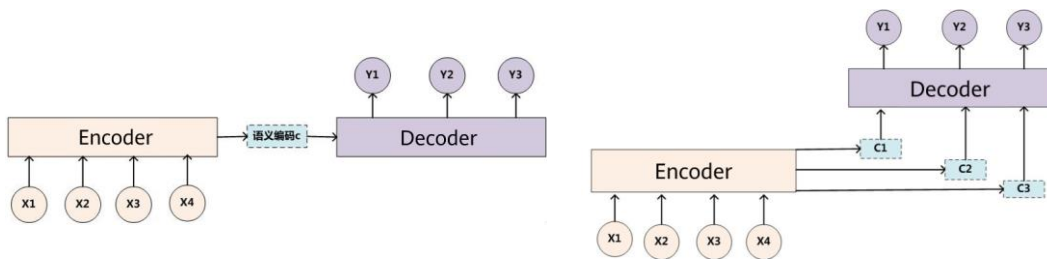
- Attention机制

2. Transformer详解

3. DeepSeek算法解析及工程优化

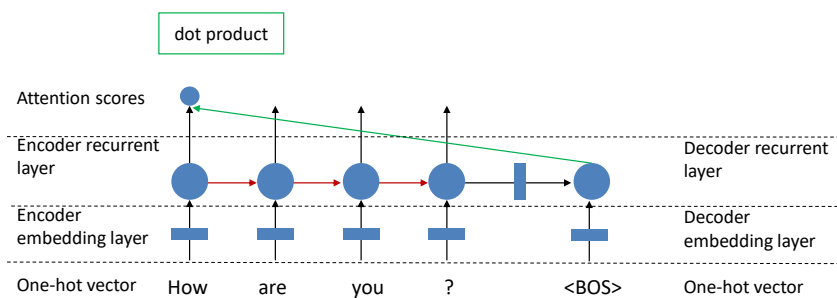
Attention机制

- 注意力机制：在解码器的每个步骤中，使用与编码器的直接连接来聚焦于源序列的特定部分。
- 注意力机制在输入信息中聚焦于更为关键的信息，降低对其他信息的关注度，甚至过滤掉无关信息，就可以解决信息过载问题，并提高任务处理的效率和准确性。



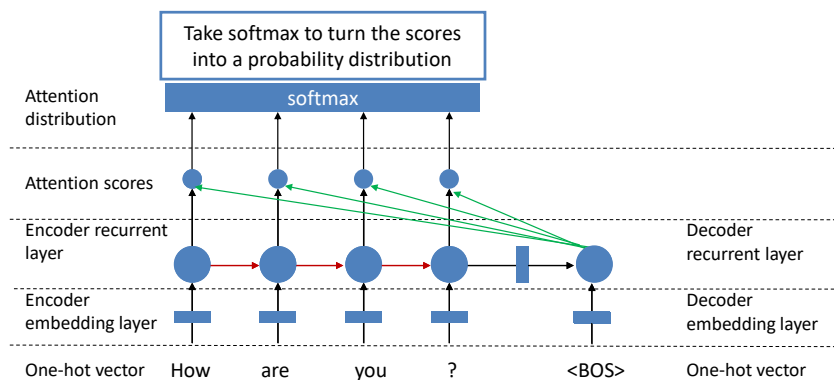
Attention机制计算过程 (1)

Seq2Seq with Attention



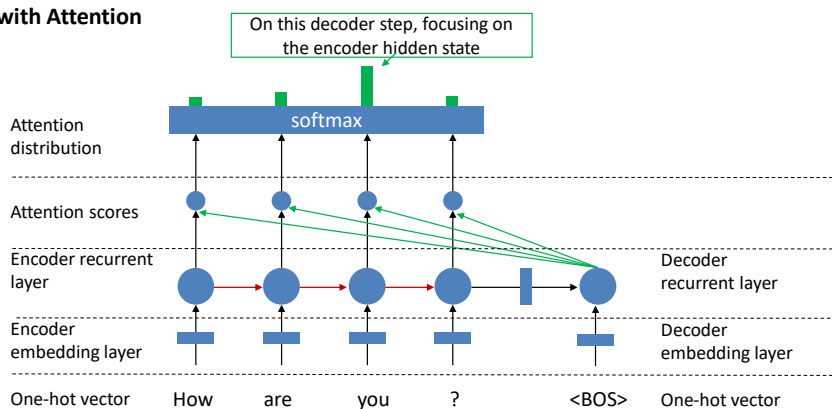
Attention机制计算过程 (2)

Seq2Seq with Attention

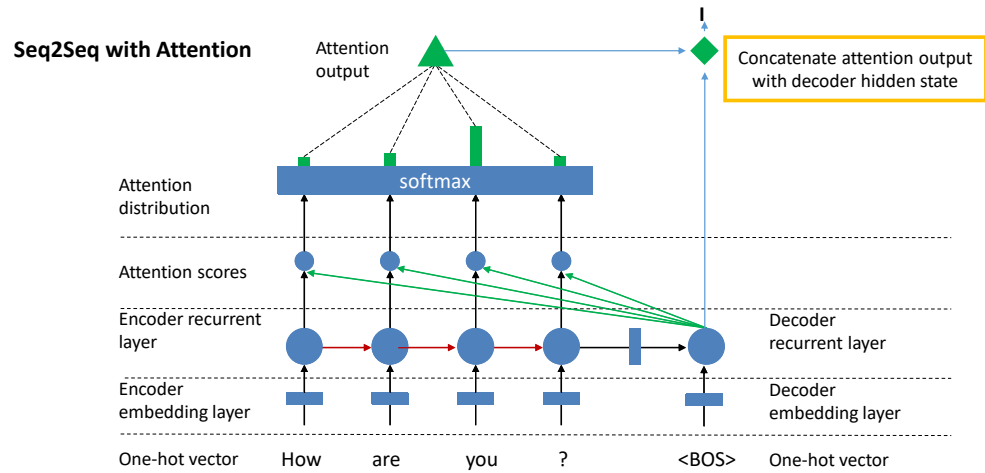


Attention机制计算过程 (3)

Seq2Seq with Attention

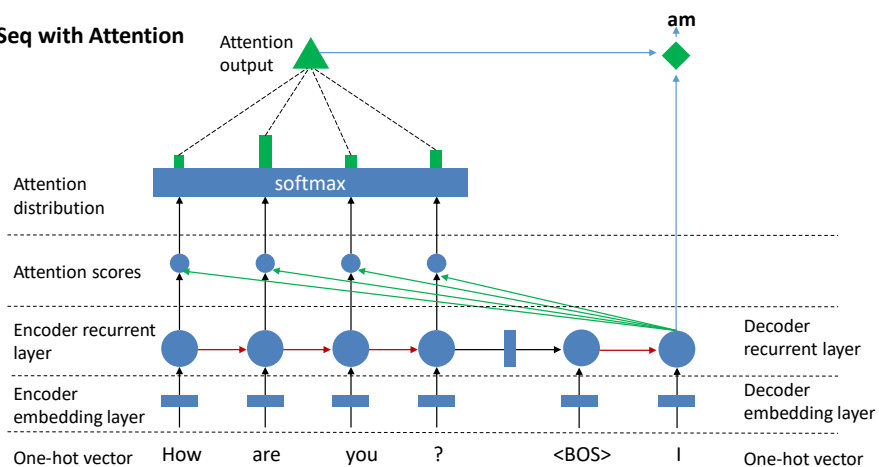


Attention机制计算过程 (4)



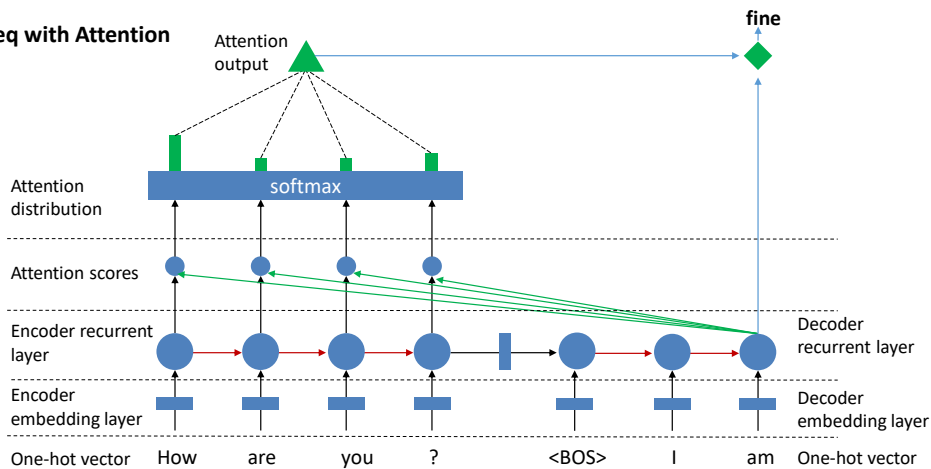
Attention机制计算过程 (5)

Seq2Seq with Attention

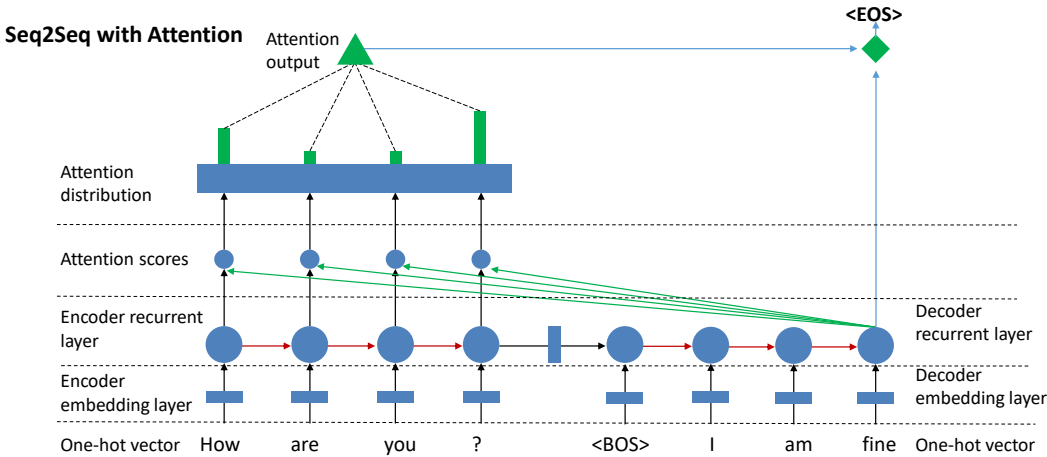


Attention机制计算过程 (6)

Seq2Seq with Attention



Attention机制计算过程 (7)



目录

1. Attention机制介绍
2. **Transformer详解**
 - Transformer概述
 - 输入
 - 自注意力机制
 - 编码器Encoder
 - 解码器Decoder
 - 最终的Linear和Softmax层
 - 模型训练
3. DeepSeek算法解析及工程优化

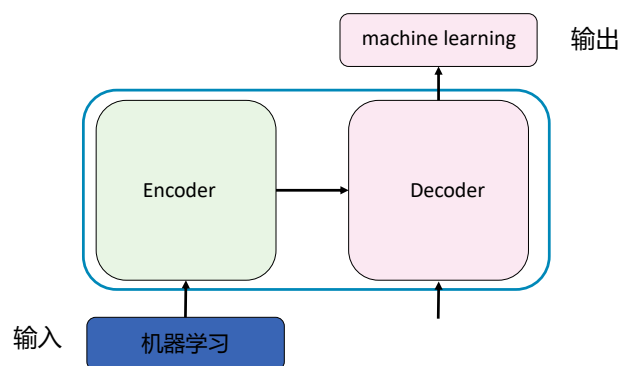
Transformer的诞生

- 2017年，谷歌机器翻译团队发表的《Attention is All You Need》中，提出了Transformer，完全抛弃了RNN（Recurrent Neural Network，循环神经网络）和CNN（Convolutional Neural Networks，卷积神经网络）等网络结构，而仅仅采用Attention机制来进行机器翻译任务，并且取得了很好的效果，注意力机制也成为了大家的研究热点。



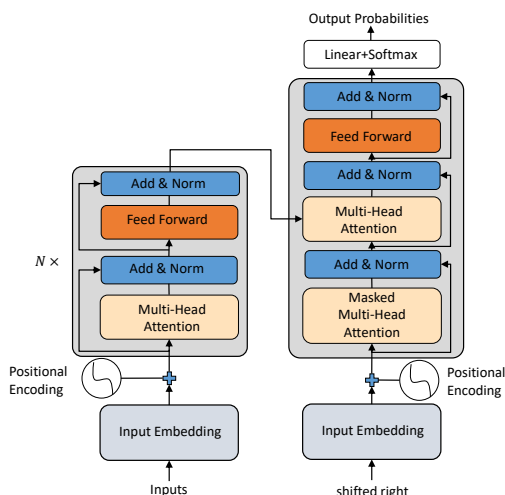
- ELMo使用了LSTM而不是新贵Transformer，如果ELMO采取Transformer作为特征提取器，那么估计Bert的反响远不如现在的这种火爆场面。另外一点，ELMO采取双向拼接这种融合特征的能力可能比Bert一体化的融合特征方式弱，但是，这只是一种从道理推断产生的怀疑，目前并没有具体实验说明这一点。

Transformer的结构图



- Transformer的结构也是由Encoder和Decoder组成。

Transformer模型结构



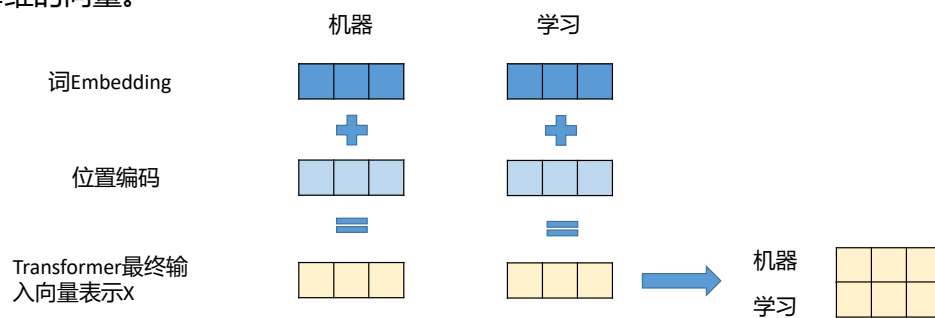
- Transformer 由 Encoder 和 Decoder 两部分组成。
- Encoder 包含 $N=6$ 个相同的 layer，layer 指的就是图中左侧的单元。最左边有个“ $N \times$ ”，指的是 N 个，如果 $N=6$ 的话，对应的就是 6 个。
- Decoder 也包含 $N=6$ 个相同的 layer。

参考文章：Attention is All You Need

- Encoder 中的 layer 结构都是相同的，但他们没有共享参数。每个 layer 都可以分解成两个 sub-layer。每个 sub-layer 的输出维度 $d_{\text{model}}=512$ 。
- 从编码器输入的句子首先会经过一个自注意力（Self-Attention）层，这层帮助编码器在对每个单词编码时关注输入句子的其他单词。
- 自注意力层的输出会传递到前馈（feed-forward）神经网络中。每个位置的单词对应的前馈神经网络都完全一样（译注：另一种解读就是一层窗口为一个单词的一维卷积神经网络）。
- 解码器中也有编码器的自注意力（Self-Attention）层和前馈（feed-forward）层。除此之外，这两个层之间还有一个注意力层，用来关注输入句子的相关部分（和 seq2seq 模型的注意力作用相似）。

Transformer工作流程：获取输入

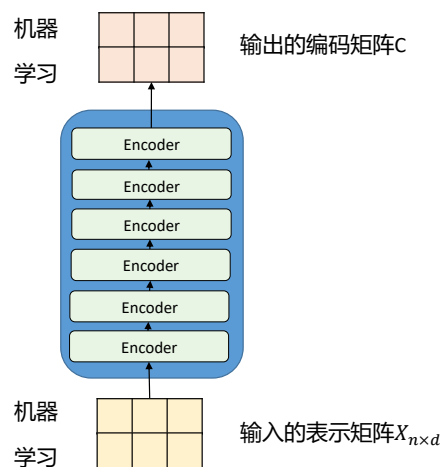
- 首先将每个输入单词通过 Embedding 转换为词向量，然后与每一个单词的位置 Embedding 相加得到每一个单词的表示向量 x 。最终，每个单词都被嵌入为一个 512 维的向量。



- 每一行是一个单词的表示 x 。

Transformer工作流程：将输入传入Encoder

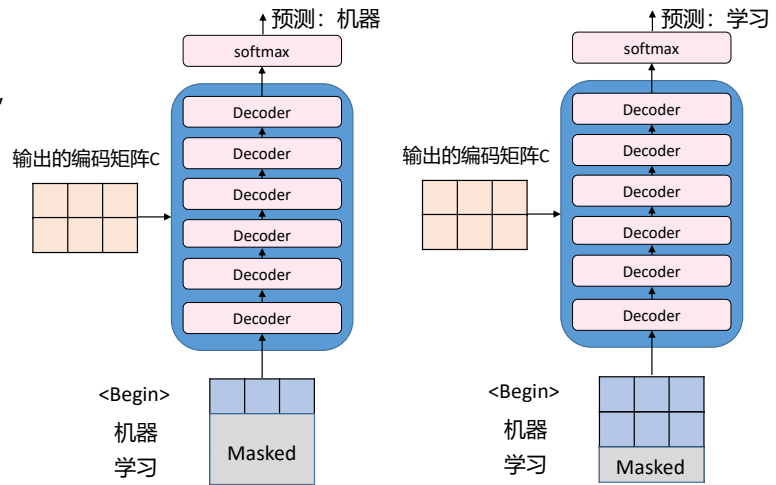
- 将得到的单词表示向量矩阵传入Encoder中，经过6个Encoder block后可以得到句子所有单词的编码信息矩阵C。
- 每一个Encoder block输出的矩阵维度与输入完全一致。



- $X_{n \times d}$: 输入矩阵, n是句子中单词数量, d表示向量的维度, 论文中d=512。

Transformer工作流程：将编码矩阵传递给Decoder

- 将Encoder输出的编码信息矩阵C传递到Decoder中，Decoder依次会根据当前翻译过的1~i的单词翻译第i+1个单词。
- 在整个过程中，翻译到第i+1个单词的时候，需要通过Mask（掩盖）操作遮盖住第i+1之后的单词。

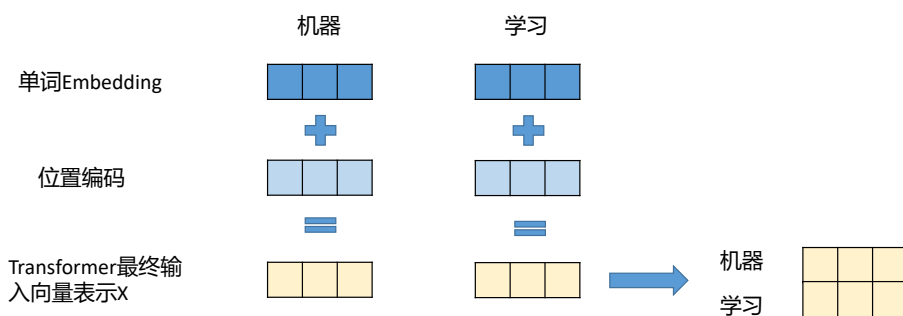


目录

1. Attention机制介绍
2. **Transformer详解**
 - Transformer概述
 - 输入
 - 自注意力机制
 - 编码器Encoder
 - 解码器Decoder
 - 最终的Linear和Softmax层
 - 模型训练
3. DeepSeek算法解析及工程优化

Transformer的输入

- Transformer中的输入X是由单词Embedding和位置编码（Positional Encoding）相加得到。



单词Embedding

- 单词的Embedding有多种方式可以得到，例如可以采用Word2Vec、Glove等算法预训练得到，也可以在Transformer中训练得到。

位置编码

- 对于Transformer来说，由于句子中的词语都是同时进入网络进行处理，顺序信息在输入网络时就已经丢失。因此，Transformer是需要额外的处理来告知每个词语的相对位置的。其中的一个解决方案，就是论文中提到的位置编码（Positional Encoding），将能表示位置信息的编码添加到输入中，让网络知道每个词的位置和顺序。
- 总而言之，Positional Encoding就是句子中词语相对位置的编码，让Transformer保留词语的位置信息。

$$PE_{(pos, 2i)} = \sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos, 2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

其中，pos表示单词在句子中的位置，d表示PE的维度（与词Embedding一样），2i表示偶数的维度，2i+1表示奇数维度（即 $2i \leq d$ ， $2i+1 \leq d$ ）。

- 使用这种公式计算PE有以下的好处：
 - 使PE能够适应比训练集里面所有句子更长的句子。
 - 可以让模型容易地计算出相对位置，对于固定长度的间距k， $PE(pos+k)$ 可以用 $PE(pos)$ 计算得到。

目录

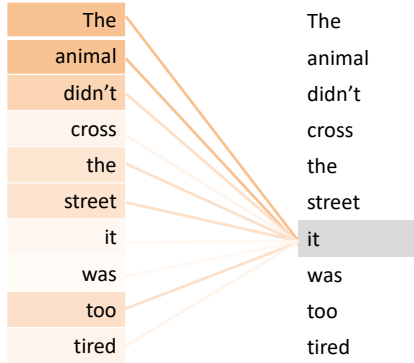
1. Attention机制介绍
- 2. Transformer详解**
 - Transformer概述
 - 输入
 - 自注意力机制
 - 编码器Encoder
 - 解码器Decoder
 - 最终的Linear和Softmax层
 - 模型训练
3. DeepSeek算法解析及工程优化

自注意力机制：Self-Attention

- 什么是自注意力机制？
- 例如，下列句子是我们想要翻译的输入句子：
 - The animal didn't cross the street because it was too tired
- 这个 “it” 在这个句子是指什么呢？它指的是 “street” 还是这个 “animal” 呢？这对于人类来说是一个简单的问题，但是对于算法则不是。
- 当模型处理这个单词 “it” 的时候，自注意力机制会允许 “it” 与 “animal” 建立联系。

自注意力机制的强大作用

Layer: 5 Attention: Input-Input



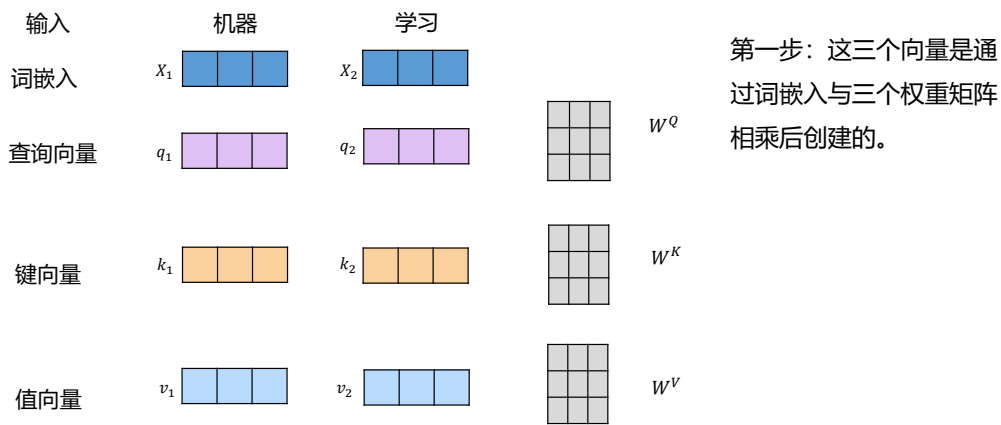
- 随着模型处理输入序列的每个单词，自注意力会关注整个输入序列的所有单词，帮助模型对本单词更好地进行编码。

- 如果你熟悉RNN（循环神经网络），回忆一下它是如何维持隐藏层的。RNN会将它已经处理过的前面的所有单词/向量的表示与它正在处理的当前单词/向量结合起来。而自注意力机制会将所有相关单词的理解融入到我们正在处理的单词中。
- 当我们在编码器中编码“it”这个单词的时候，注意力机制的部分会去关注“The Animal”，将它的表示的一部分编入“it”的编码中。

计算自注意力

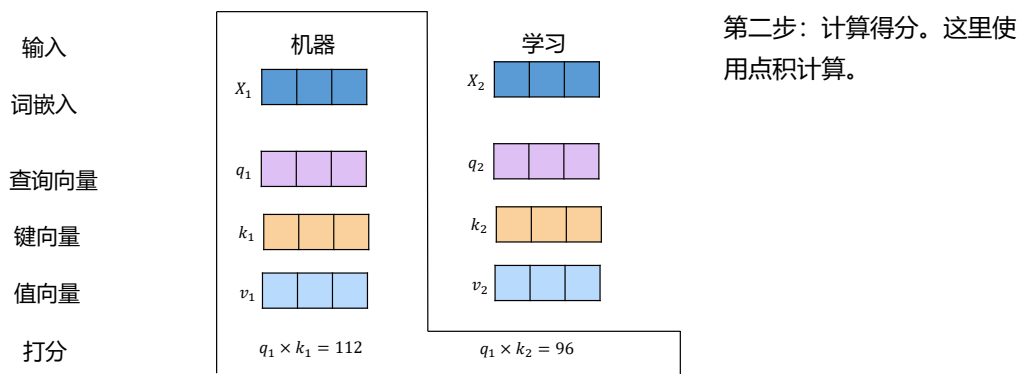
- 首先我们了解一下如何使用向量来计算自注意力，然后看它是怎样用矩阵来实现的。
- 计算自注意力的第一步就是从每个编码器的输入向量中生成三个向量。也就是说对于每个单词，我们创建一个查询向量、一个键向量和一个值向量。

第一步：生成查询向量、键向量和值向量



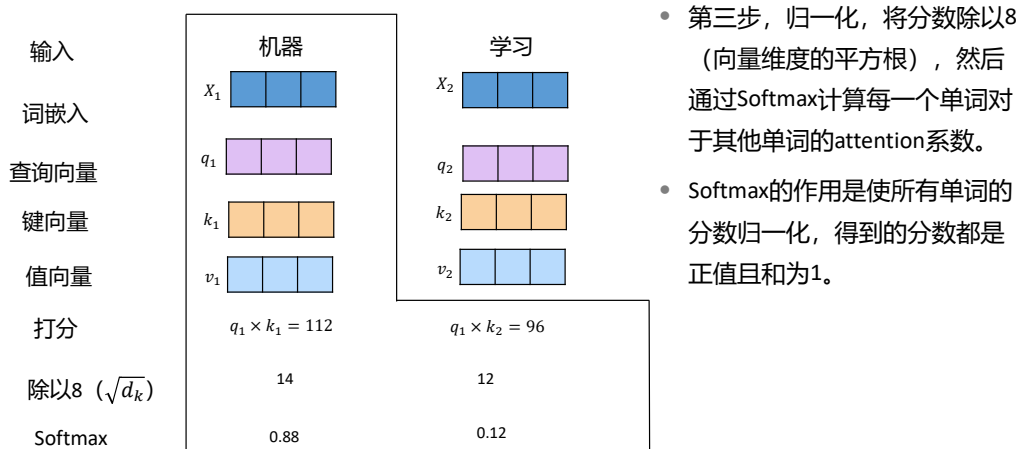
- 这三个向量是通过词嵌入与三个权重矩阵相乘后创建的。这些新向量在维度上比词嵌入向量更低，他们的维度是64，而词嵌入和编码器的输入/输出向量的维度是512。但实际上不强求维度更小，这只是一种基于架构上的选择，它可以使多头注意力（multiheaded attention）的大部分计算保持不变。
- x_1 与 w^Q 权重矩阵相乘得到 q_1 ，就是与这个单词相关的查询向量。最终使得输入序列的每个单词都创建一个查询向量、一个键向量和一个值向量。
- 什么是查询向量、键向量和值向量？它们都是有助于计算和理解注意力机制的抽象概念。

第二步：计算得分



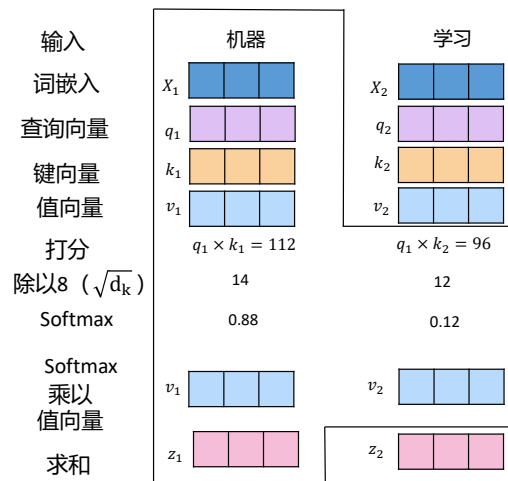
- 计算自注意力的第二步是计算得分。假设我们在为这个例子中的第一个词“机器”计算自注意力向量，我们需要拿输入句子中的每个单词对“机器”打分。这些分数决定了在编码单词“机器”的过程中有多重视句子的其它部分。这些分数是通过打分单词（所有输入句子的单词）的键向量与“机器”的查询向量相点积来计算的。所以如果我们是处理位置最靠前的词的自注意力的话，第一个分数是 q_1 和 k_1 的点积，第二个分数是 q_1 和 k_2 的点积。

第三步：归一化



- 8是论文中使用的键向量的维数64的平方根，这会让梯度更稳定。这里也可以使用其它值，8只是默认值。
- 这个Softmax分数决定了每个单词对编码当下位置（“机器”）的贡献。显然，已经在这个位置上的单词将获得最高的Softmax分数，但有时关注另一个与当前单词相关的单词也会有帮助。

第四步：对加权值向量求和



- 第四步，在编码某个单词时，将所有单词的表示（值向量）进行加权求和，而权重是通过该词的表示（键向量）与被编码词表示（查询向量）的点积并归一化得到，即第三步得到的结果。
- 然而实际中，这些计算是以矩阵形式完成的。

- 将每个值向量乘以Softmax分数（这是为了准备之后将它们求和）。这里的直觉是希望关注语义上相关的单词，并弱化不相关的单词（例如，让它们乘以0.001这样的小数）。
- 对加权值向量求和（自注意力的另一种解释就是在编码某个单词时，就是将所有单词的表示（值向量）进行加权求和，而权重是通过该词的表示（键向量）与被编码词表示（查询向量）的点积并通过Softmax得到），然后即得到自注意力层在该位置的输出（在我们的例子中是对应第一个单词）。
- 这样自注意力的计算就完成了。得到的向量就可以传给前馈神经网络。然而实际中，这些计算是以矩阵形式完成的，以便算得更快。那我们接下来就看看如何用矩阵实现的。

通过矩阵运算实现自注意力机制 (1)

$$\begin{matrix} & \mathbf{x} & & \mathbf{W}^Q & & \mathbf{Q} \\ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} & \times & \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} & = & \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix}$$

$$\begin{matrix} & \mathbf{x} & & \mathbf{W}^K & & \mathbf{K} \\ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} & \times & \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} & = & \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix}$$

$$\begin{matrix} & \mathbf{x} & & \mathbf{W}^V & & \mathbf{V} \\ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} & \times & \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} & = & \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix}$$

- 第一步是计算查询矩阵、键矩阵和值矩阵。为此，我们将输入句子的词嵌入到矩阵 $\mathbf{x}$ 中，将其乘以我们训练的权重矩阵（ $\mathbf{W}^Q$ ， $\mathbf{W}^K$ ， $\mathbf{W}^V$ ）。
- $\mathbf{x}$ 矩阵中的每一行对应于输入句子中的一个单词。

通过矩阵运算实现自注意力机制（2）

$$Attention(Q, K, V) = softmax\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) \times V = Z$$

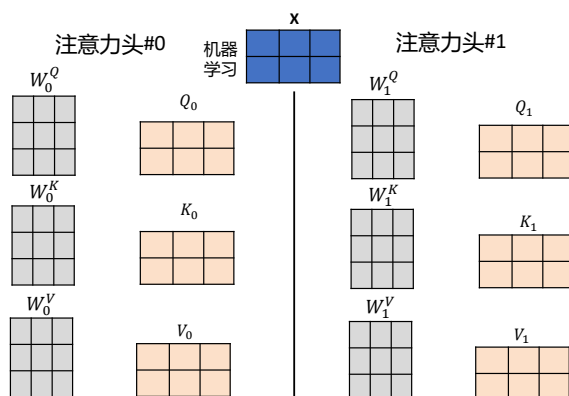
- 最后，由于我们处理的是矩阵，我们可以将步骤2到步骤4合并为一个公式来计算自注意力层的输出。

多头注意力机制

- 论文为了进一步完善自注意力层，增加一种叫做“多头”注意力的机制。
- 通过这种机制可以进一步提高注意力层的性能。

- 论文为了进一步完善自注意力层，增加一种叫做“多头”注意力（“multi-headed” attention）的机制。
- 通过这种机制可以进一步提高注意力层的性能。

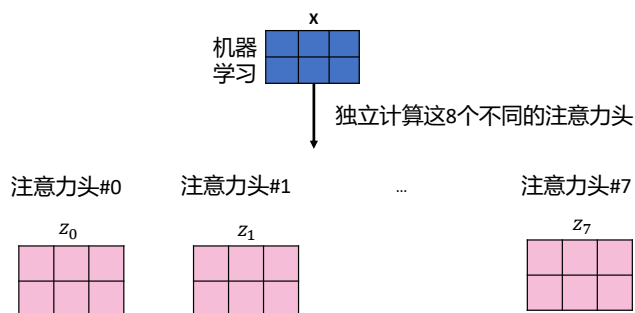
提高注意力层的性能



- 在两方面提高了注意力层的性能，一方面扩展了模型专注于不同位置的能力，另一方面，它给出了注意力层的多个“表示子空间”。

- 通过增加一种叫做“多头”注意力（“multi-headed” attention）的机制，论文进一步完善了自注意力层，并在两方面提高了注意力层的性能：1.它扩展了模型专注于不同位置的能力。在上面的例子中，虽然每个编码都在 z_1 中有或多或少的体现，但是它可能被实际的单词本身所支配。如果我们翻译一个句子，比如“The animal didn’t cross the street because it was too tired”，我们会想知道“it”指的是哪个词，这时模型的“多头”注意机制会起到作用。2.它给出了注意力层的多个“表示子空间”（representation subspaces）。接下来我们将看到，对于“多头”注意机制，我们有多组查询/键/值权重矩阵集（Transformer使用八个注意力头，因此我们对于每个编码器/解码器有八个矩阵集合）。这些集合中的每一个都是随机初始化的，在训练之后，每个集合都被用来将输入词嵌入（或来自较低编码器/解码器的向量）投影到不同的表示子空间中。

八次权重矩阵运算

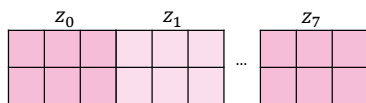


- 如果我们做与上述相同的自注意力计算，只需八次不同的权重矩阵运算，我们就会得到八个不同的z矩阵。

- 在“多头”注意机制下，我们为每个头保持独立的查询/键/值权重矩阵，从而产生不同的查询/键/值矩阵。和之前一样，我们拿X乘以WQ/WK/WV矩阵来产生查询/键/值矩阵。
- 如果我们做与上述相同的自注意力计算，只需八次不同的权重矩阵运算，我们就会得到八个不同的z矩阵。
- 这给我们带来了一点挑战。前馈层不需要8个矩阵，它只需要一个矩阵（由每一个单词的表示向量组成）。所以我们需要一种方法把这八个矩阵压缩成一个矩阵。那该怎么做？

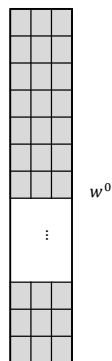
八个矩阵压缩成一个矩阵

1、将所有注意力头拼接起来

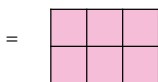


2、乘以权重矩阵 w^0 (它是随模型一起训练获得)

×



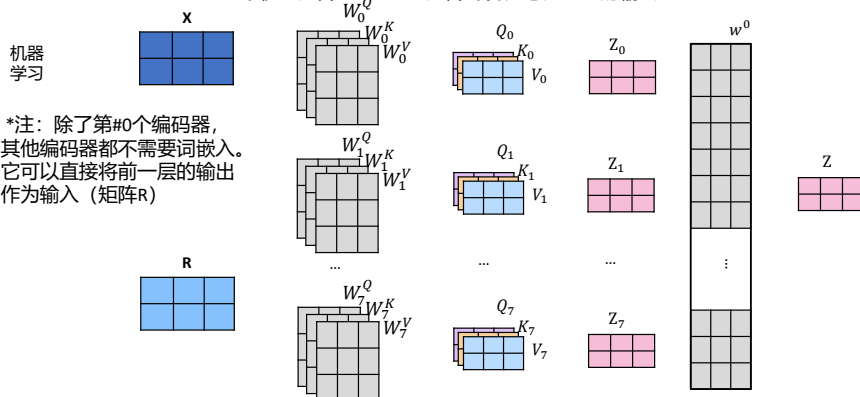
3、结果是一个融合所有注意力头信息的矩阵 z ，我们可以将其送到前馈神经网络



- 其实可以直接把这些矩阵拼接在一起，然后用一个附加的权重矩阵 w^0 与它们相乘。

计算过程集中在一起看

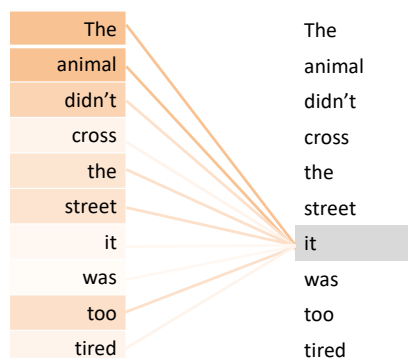
- 1、这是我们的输入句子
- 2、编码每一个单词
- 3、将其分成8个头，将矩阵X或R乘以各个权重矩阵
- 4、通过输出查询/键/值 (Q/K/V) 矩阵计算注意力
- 5、将所有注意力矩阵Z拼接起来，然后乘以权重矩阵 w^0 来产生该层的输出



- 得到 8 个输出矩阵 Z_0-Z_7 之后，Multi-Head Attention将它们拼接在一起（Concat），然后传入一个Linear层，得到Multi-Head Attention最终的输出 Z 。
- Multi-Head Attention输出的矩阵 Z 与其输入的矩阵 X 的维度是一样的。
- 既然我们已经摸到了注意力机制的这么多“头”，那么重温之前的例子，看看在例句中编码“it”一词时，不同的注意力“头”集中在哪里。

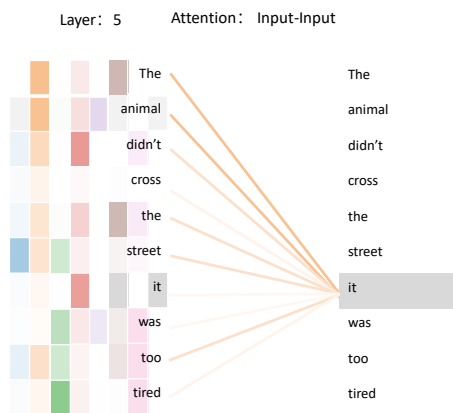
重温之前的例子

Layer: 5 Attention: Input-Input



- 在编码 “it” 一词时，一个注意力头集中在 “animal” 上，而另一个则集中在 “tired” 上，从某种意义上说，模型对 “it” 一词的表达在某种程度上是 “animal” 和 “tired” 的代表。

把所有的attention都加进来

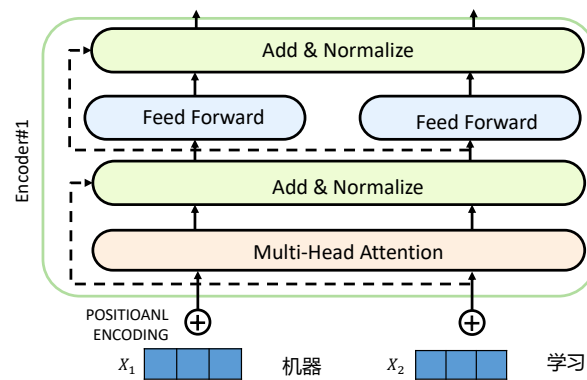


- 然而，如果我们把所有的attention都加到图示里，事情更难解释了。

目录

1. Attention机制介绍
2. **Transformer详解**
 - Transformer概述
 - 输入
 - 自注意力机制
 - **编码器Encoder**
 - 解码器Decoder
 - 最终的Linear和Softmax层
 - 模型训练
3. DeepSeek算法解析及工程优化

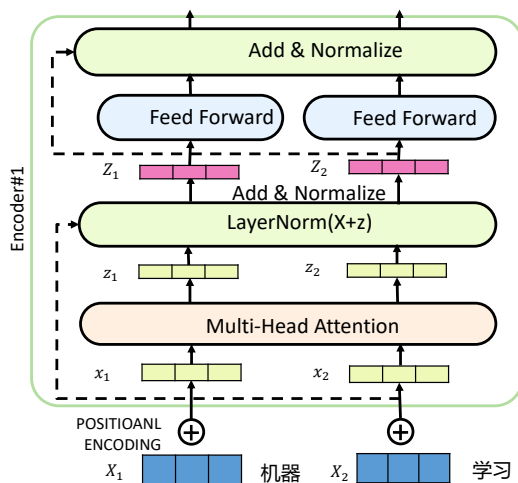
编码器Encoder



- 在每个编码器中的每个子层（自注意力、前馈网络）的周围都有一个残差连接，并且都跟随着一个“层-归一化”步骤。

- 残差连接和层归一化。
- 在每个编码器中的每个子层（自注意力、前馈网络）的周围都有一个残差连接，并且都跟随着一个“层-归一化”步骤 (<https://arxiv.org/abs/1607.06450>)

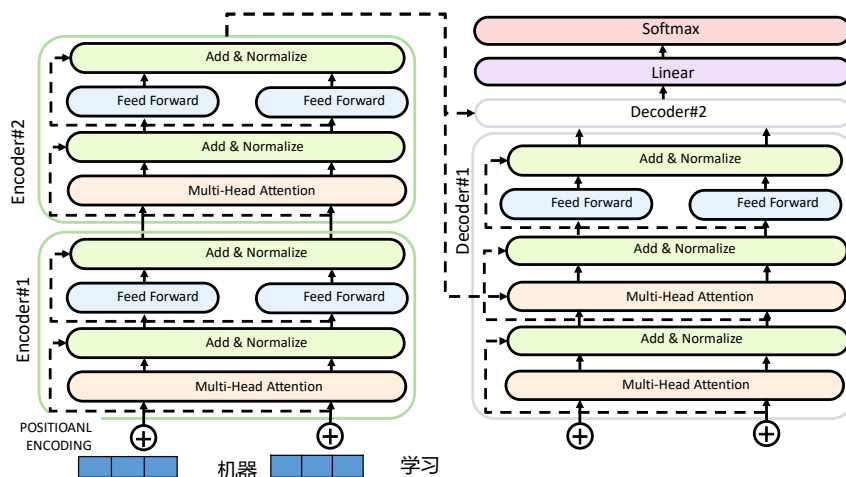
细化这个模块



- 如果我们去可视化这些向量以及这个和自注意力相关联的层归一化操作，那么看起来就像左边这张图一样。

- 如果我们去可视化这些向量以及这个和自注意力相关联的层-归一化操作，那么看起来就像左边这张图一样。

一个2层的Encoder



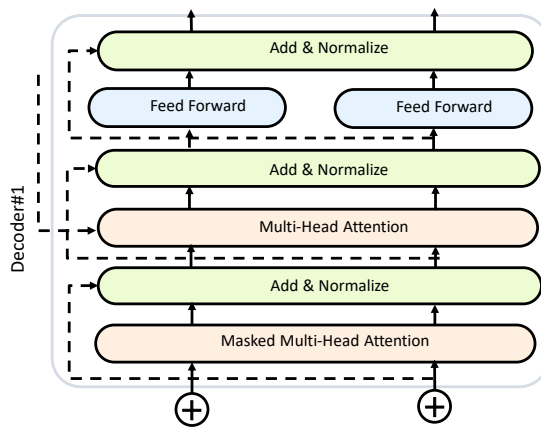
- 如果我们想象一个2层编码-解码结构的transformer，它看起来会像这张图一样。

目录

1. Attention机制介绍
2. **Transformer详解**
 - Transformer概述
 - 输入
 - 自注意力机制
 - 编码器Encoder
 - **解码器Decoder**
 - 最终的Linear和Softmax层
 - 模型训练
3. DeepSeek算法解析及工程优化

解码器Decoder

- Decoder:
 - 包含两个Multi-Head Attention子层。
 - 第一个Multi-Head Attention子层采用了Masked操作。
 - 第二个Multi-Head Attention子层的K, V矩阵使用Encoder的编码信息矩阵C进行计算，而Q使用上一层的输出进行计算。



- Decoder与Encoder的区别。

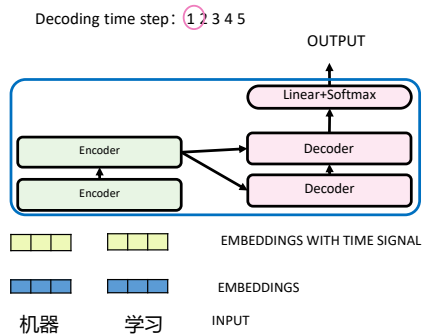
第一个Multi-Head Attention

- Decoder block的第一个Multi-Head Attention采用了Masked操作，因为在翻译的过程中是顺序翻译的，即翻译完第 i 个单词，才可以翻译第 $i+1$ 个单词。通过Masked操作可以防止第 i 个单词知道 $i+1$ 个单词之后的信息。

第二个Multi-Head Attention

- Decoder block第二个Multi-Head Attention变化不大，主要的区别在于其中 Self-Attention的 K , V 矩阵不是基于上一层输出计算的，而是使用Encoder的编码信息矩阵 C 计算的。
- 根据Encoder输出的编码信息矩阵 C 计算得到 K 和 V ，根据上一层的输出 Z 计算 Q ，其余计算方法与之前描述一致。
- 这样做的好处是在Decoder的时候，每一个单词都可以利用到Encoder中所有单词的信息。

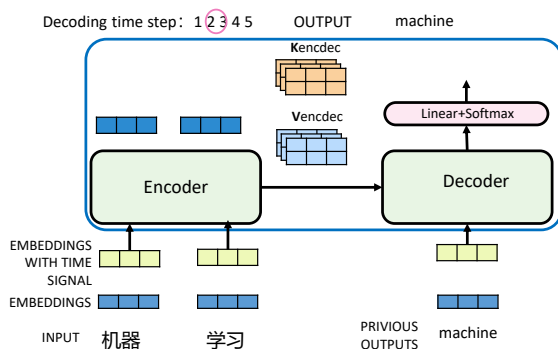
编码器输出到解码器



- 编码器通过处理输入序列开启工作。顶端编码器的输出之后会转化为一个包含向量K（键向量）和V（值向量）的注意力向量集。
- 在完成编码阶段后，则开始解码阶段。解码阶段的每个步骤都会输出一个序列的元素（在这个例子里，是英语翻译的句子）。

- 既然我们已经谈到了大部分编码器的概念，那么我们基本上也就知道解码器是如何工作的了。但最好还是看看解码器的细节。编码器通过处理输入序列开启工作。顶端编码器的输出之后会变转化为一个包含向量K（键向量）和V（值向量）的注意力向量集。这些向量将被每个解码器用于自身的“编码-解码注意力层”，而这些层可以帮助解码器关注输入序列哪些位置合适。

解码器不断输出



- 接下来的步骤重复了这个过程，直到到达一个特殊的终止符号，它表示transformer的解码器已经完成了它的输出。每个步骤的输出在下一个时间步被提供给底端解码器，并且就像编码器之前做的那样，这些解码器会输出它们的解码结果。另外，就像我们对编码器的输入所做的那样，会嵌入并添加位置编码给那些解码器，来表示每个单词的位置。

- 而那些解码器中的自注意力层表现的模式与编码器不同：在解码器中，自注意力层只被允许处理输出序列中更靠前的那些位置。在Softmax步骤前，它会把后面的位置给隐去（把它们设为-inf）。这个“编码-解码注意力层”工作方式基本就像多头自注意力层一样，只不过它是通过在它下面的层来创造查询矩阵，并且从编码器的输出中取得键/值矩阵。

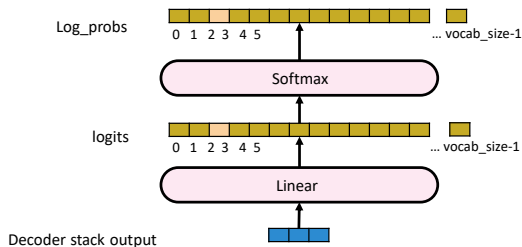
目录

1. Attention机制介绍
- 2. Transformer详解**
 - Transformer概述
 - 输入
 - 自注意力机制
 - 编码器Encoder
 - 解码器Decoder
 - 最终的Linear和Softmax层
 - 模型训练
3. DeepSeek算法解析及工程优化

解码器输出到Linear层再到Softmax层

Which word in our vocabulary
is associated with this index?

Get the index of the cell with
the highest value
(argmax)



- 解码组件最后会输出一个实数向量。我们如何把浮点数变成一个单词？这便是线性变换层要做的工作，它之后就是Softmax层。

- 解码组件最后会输出一个实数向量。我们如何把浮点数变成一个单词？这便是线性变换层要做的工作，它之后就是Softmax层。线性变换层是一个简单的全连接神经网络，它可以把解码组件产生的向量投射到一个比它大得多的、被称作对数几率（logits）的向量里。不妨假设我们的模型从训练集中学习一万个不同的英语单词（我们模型的“输出词表”）。因此对数几率向量为一万个单元格长度的向量——每个单元格对应某一个单词的分数。接下来的Softmax层便会把那些分数变成概率（都为正数、上限1.0）。概率最高的单元格被选中，并且它对应的单词被作为这个时间步的输出。

模型训练

- 最后来看一下训练Transformer的关键点。

词表

Output Vocabulary

WORD	thanks	I	machine	learning	like	<EOS>
INDEX	0	1	2	3	4	5

- 既然我们已经过了一遍完整的transformer的前向传播过程，那我们就可以直观感受一下它的训练过程。其中<EOS>指的是End of Sentence的缩写形式，代表句子的结束。

- 开始讲整体训练。
- 既然我们已经过了一遍完整的transformer的前向传播过程，那我们就可以直观感受一下它的训练过程。
- 在训练过程中，一个未经训练的模型会通过一个完全一样的前向传播。但因为我们用有标记的训练集来训练它，所以我们可以用它的输出去与真实的输出做比较。
- 为了把这个流程可视化，不妨假设我们的输出词汇仅仅包含六个单词：“a”，“am”，“I”，“thanks”，“student”以及“”（end of sentence的缩写形式）。

单词编码

Output Vocabulary

WORD	thanks	I	machine	learning	like	<EOS>
INDEX	0	1	2	3	4	5

One-hot encoding of the word "like"

0.0	0.0	0.0	0.0	1.0	0.0
-----	-----	-----	-----	-----	-----

- 一旦我们定义了输出词表，就可以使用一个相同宽度的向量来表示词汇表中的每一个单词。这也被认为是一个one-hot 编码。所以，可以用下面这个向量来表示单词 "like" : [0, 0, 0, 0, 1, 0]。

模型的目标输出

Target Model Output

Output Vocabulary: thanks I machine learning like <EOS>

Position#1	0.0	0.0	1.0	0.0	0.0	0.0
Position#2	0.0	0.0	0.0	1.0	0.0	0.0
Position#3	0.0	0.0	0.0	0.0	0.0	1.0

- 更现实的情况是处理一个句子。例如，输入“机器学习”并期望输出是“machine learning”。那我们就希望模型能够成功地在这种情况下输出概率分布。
- 每个概率分布被一个以词表大小（我们的例子里是6，但现实情况通常是3000或10000）为宽度的向量所代表。

- 更现实的情况是处理一个句子。例如，输入“机器学习”并期望输出是“machine learning”。

模型的训练输出

Target Model Output

Output Vocabulary: thanks | machine learning like <EOS>

Position#1	0.01	0.02	0.92	0.01	0.03	0.01
Position#2	0.01	0.05	0.1	0.8	0.01	0.03
Position#3	0.01	0.02	0.01	0.01	0.01	0.94

- 在一个足够大的数据集上充分训练后，我们希望模型输出的概率分布看起来像这个样子。
- 我们期望训练过后，模型会输出正确的翻译。当然如果这段话完全来自训练集，它并不是一个很好的评估指标。

- 注意到每个位置（词）都得到了一个概率，即使它不太可能成为那个时间步的输出——这是Softmax的一个很有用的性质，它可以帮助模型训练。
- 因为这个模型一次只产生一个输出，不妨假设这个模型只选择概率最高的单词，并把剩下的词抛弃。这是其中一种方法（叫贪心解码）。另一个完成这个任务的方法是留住概率最高的两个单词（例如I和a），那么在下一步里，跑模型两次：其中一次假设第一个位置输出是单词“I”，而另一次假设第一个位置输出是单词“me”，并且无论哪个版本产生更少的误差，都保留概率最高的两个翻译结果。然后我们为第二和第三个位置重复这一步骤。这个方法被称作集束搜索（beam search）。在我们的例子中，集束宽度是2（因为保留了2个集束的结果，如第一和第二个位置），并且最终也返回两个集束的结果（top_beams也是2）。这些都是可以提前设定的参数。

目录

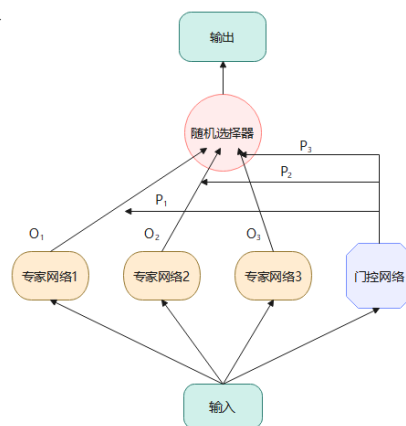
1. Attention机制介绍
2. Transformer详解
3. **DeepSeek算法解析及工程优化**

稠密模型和稀疏模型

- 模型规模是提升模型性能的关键因素之一。随着基于Transformer结构的大模型不断发展，模型的参数量越来越大（提高模型性能），如果训练和推理时全部的参数均参与计算（即参与计算的矩阵大多为稠密矩阵），那么计算所需要的算力会越来越高。因此这类大参数量的稠密模型推理成本相对较高。
- 那如何提高模型的参数，但同时又不显著增加计算需求量呢？
- 混合专家模型（ MoE ）是一个很好的选择！它在计算过程中，均是部分参数参与计算，即只有少部分参数不为零。这种稀疏矩阵的计算可以减少计算资源的需求，模型单次训练和推理所需计算资源低，计算时间短。

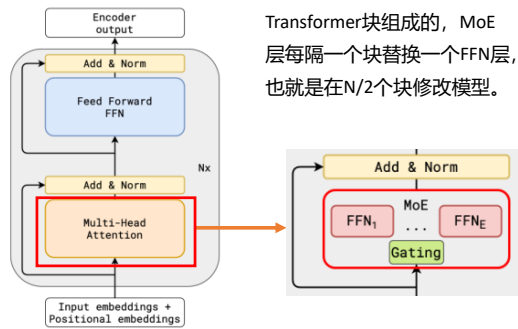
混合专家模型（MoE）

- 究竟什么是一个混合专家模型（MoE）呢？它的整体框架主要分为以下部分：
 - 专家网络：用于学习不同的数据。输入数据经过每个专家系统，分别会输出一个值，在该图中，每个输出标记为 O_i 。
 - 门控网络：决定每个专家系统输出在随机选择器里的权重。它为每个输出分配的权重为 P_i 。该网络同样可以被训练。
 - 随机选择器：获取多个专家系统的当前输出，结合权重获得最终输出结果。



MoE模型 - GShard

- Gshard是Google发布的首个Transformer架构与MoE结合的模式架构。
- Transformer 编码器层由两个连续层组成，即一个自注意力层，后面跟着一个位置前馈层。解码器添加了第三个交叉注意力层，该层关注编码器的输出。通过条件计算稀疏地缩放Transformer，将每个其他前馈层替换为Position-wise Mixture of Experts (MoE) 层。



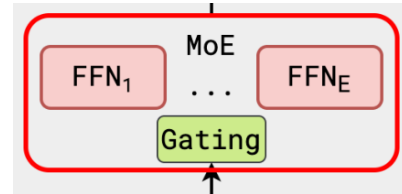
- 之前的MoE是把每个样本分到不同的expert，Transformer架构中是把每个token分到不同的expert。

GShard - Top-K gating

- 门控网络可能会使用Top-K gating机制，这意味着对于每个输入token，只有概率最高的K个专家会被选中进行计算。Gshard使用的K是2，即top-2。

- Gshard中门控函数满足两个条件：

- 平衡负载
- 可高效率的大规模并行计算

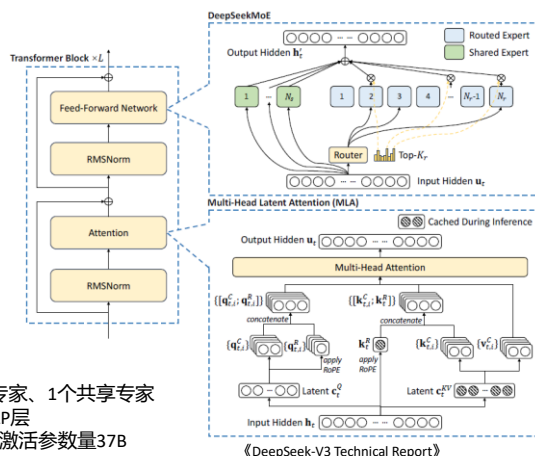


- 希望MoE层能够稀疏地激活给定令牌的专家。一个简单的解决方案是根据softmax概率分布选择top-k专家。然而，众所周知，这种方法会导致训练的负载不平衡问题。在训练过程中看到的大多数令牌将被分配给少数专家，只为少数（忙碌的）专家积累了非常大的输入缓冲区，而其他专家则未得到训练，从而减慢了训练速度。与此同时，许多其他专家根本没有得到充分的训练。一个更好的门控函数设计可以更均匀地将处理负担分配给所有专家。
- 如果按顺序执行门控功能，则实现平衡负载将是相当简单的。对于给定E个专家的输入批中所有N个令牌，单独的门控函数的计算成本至少为O。然而，在研究中，N的数量级为百万，E的数量级为千，门控函数的顺序实现将使大部分计算资源在大多数时间处于空闲状态。因此，需要一个有效的门控函数并行实现来利用许多器件。

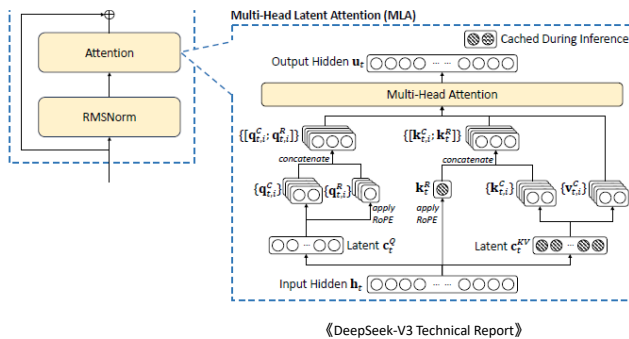
DeepSeek-V3

- 2024 年 12 月 26 日晚，AI 公司深度求索（DeepSeek）正式上线全新系列模型 DeepSeek-V3 首个版本并同步开源。该模型以强大的通用 AI 能力（尤其是在知识问答、文本生成、代码编写等任务）和较低的训练和推理成本震惊 AI 界。
- DeepSeek-V3 网络依然采用了 MoE 的结构，并在此基础上进行了优化。

61 层、256 路由专家、1 个共享专家
 $K_r=8$ ，前三层 MLP 层
 总参数量 671B，激活参数量 37B

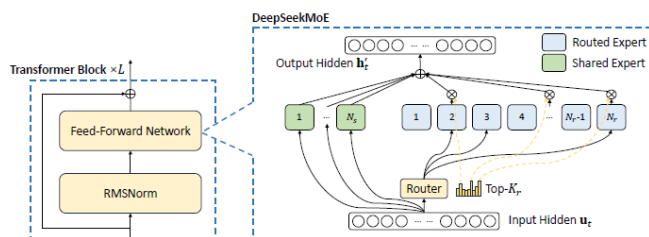


DeepSeek-V3模型结构创新 - MLA



- MLA主要作用是通过优化KV-Cache来减少显存占用，从而提升推理性能。
- 功能流程：
 - 输入隐藏状态ht
 - 计算Q/K/V
 - 融合旋转位置编码
 - 输出隐藏状态ut

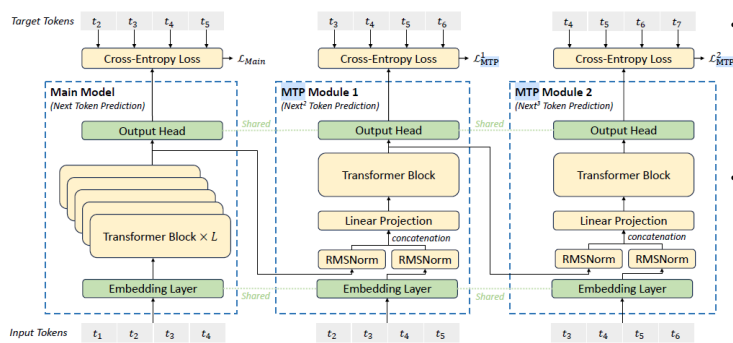
DeepSeek-V3模型结构创新 - DeepSeekMoE



《DeepSeek-V3 Technical Report》

- Router (路由器)
 - 路由器决定输入 u_z 的每一部分将路由到哪些专家
 - 每个词元最多路由到 K_r 个专家节点，减少通信开销
- Experts (专家)，包含256个专家，分为两类：
 - Routed Expert: 被路由器选择的专家
 - Shared Expert: 共享的专家，用于捕捉全局特性
- Top- K_r 策略：
 - 在 N_c 个专家中选出响应最优的 K_r 个专家
 - 限制跨节点通信范围，优化效率
 - 从256个专家中选9个 (8+1)

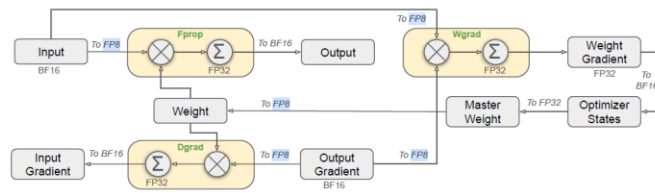
DeepSeek-V3模型结构创新 - MTP



《DeepSeek-V3 Technical Report》

- 传统预测推理速度慢，逐个生成 token，预测一个 token 需要大量跟显存交互，加上单个 token 的预测倾向于局部捕捉最优，整体可能效果不好。
- 核心实现思路：在解码阶段，让模型一次性通过多个顺序模块预测多个未来的 token。具体在训练阶段，一次性学习多个位置的 label，提升样本的有效利用率，提升训练效率。该技术可以用在推理阶段，一次生成多个 token，实现推理速度成倍增加。

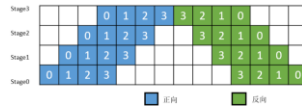
DeepSeek-V3工程创新 – FP8低位宽训练



- 混合精度策略：
 - 计算密集型（前向传播、激活反向、权重反向等）算子均采用FP8计算来提升训练速度
 - 精度敏感型（向量层、输出层、门控模块、注意力模块等）仍然保持BF16/FP32精度，确保精度损失保持在可接受范围内
 - 优化器动量采用BF16，主要权重参数和梯度保持FP32格式

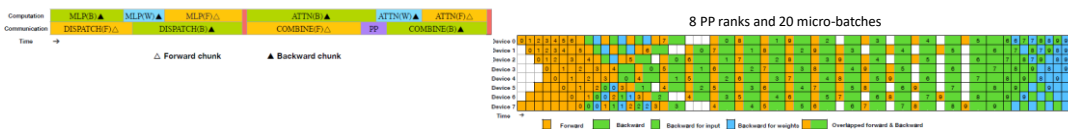
DeepSeek-V3工程创新 – DualPipe

- 在传统的流水线并行中（Pipeline），每个计算节点执行一部分计算（如前向传播、反向传播等），计算完成后，需要将结果传输给下一个GPU。如果每次计算后必须等待通信完成才能继续下一次计算，跨节点通信未完成，下一阶段的节点只能等待，导致计算资源闲置，这段空闲的时间片段称为流水线气泡（Pipeline Bubbles）。



1F1B, 白色区域即每张计算卡的空泡时间。

- DualPipe:
 - 每个计算块被划分为四个功能模块：注意力机制、全节点数据分发、MLP 处理、全节点数据整合
 - 使用双向流水线及流水线重排，有效融合前向和后向计算的通信阶段，减少流水线阻塞实现计算与通信充分重叠



- https://www.mindspore.cn/docs/zh-CN/r1.7/design/pipeline_parallel.html

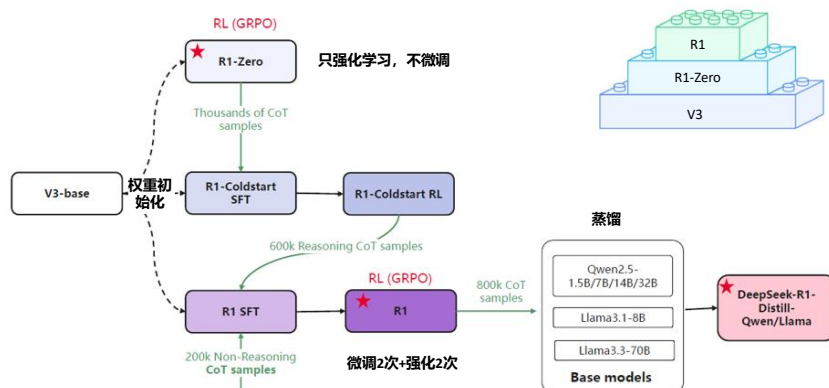
DeepSeek-R1

- 2025年1月20日，DeepSeek正式发布DeepSeek-R1模型，并同步开源模型权重。DeepSeek-R1在仅有极少标注数据的情况下，极大提升了模型推理能力。



- 广义上的DeepSeek-R1不是一个单一的模型，还包括了R1的初始阶段模型DeepSeek-R1-Zero，以及几个基于R1蒸馏的较小的大模型。

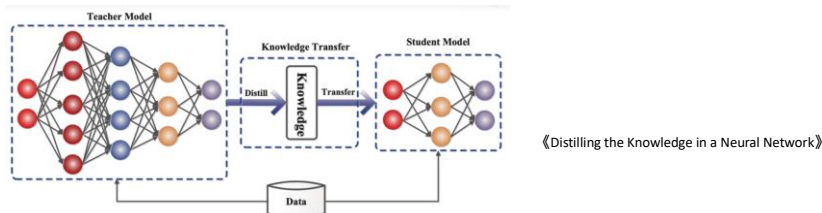
DeepSeek-R1训练流程



- **R1-Zero**: 抛开传统的SFT、MCTS、PRM，而是在基模上应用GRPO算法进行强化学习，使模型能够自主探索CoT来推理解决复杂问题。

DeepSeek模型蒸馏

- 模型蒸馏 (Knowledge Distillation) 是一种将大型复杂模型 (教师模型, 比如: DeepSeek-R1 671B) 的知识迁移到小型高效模型 (学生模型, 比如: Qwen 7B) 的技术。



- DeepSeek-R1生成80万条数据作为监督数据。
- 对开源小模型如Qwen、Llama系列进行监督微调 (SFT), 而不使用强化学习。
- 32B/70B模型的性能接近OpenAI o1-mini。例如, DeepSeek-R1-DistillQwen 32B 在AIME 2024上达到了72.6%的Pass@1。

思考题

1. （多选题）下列哪些类型的任务适合用Seq2Seq处理？（ ）
- A. 文本翻译
 - B. 语音识别
 - C. 图像分类
 - D. 对联生成

- 答案：ABD

思考题

2. （多选题）下列哪些属于Transformer 采用的技术？（ ）
- A. 前馈神经网络
 - B. 归一化
 - C. 残差模块
 - D. 自注意力机制

- 答案：ABCD

本章总结

- Transformer开创性的思想颠覆了以往序列建模和RNN划等号的思路，现在被广泛应用于NLP的各个领域。目前在NLP各业务全面开花的语言模型如GPT，BERT等都是基于Transformer架构的。
- 本章详细介绍了Attention机制以及Transformer的核心技术，主要包括为什么要引入Attention，Attention机制的具体原理，Transformer的结构，自注意力机制，编码器，解码器等，并分析了Transformer的训练过程。

缩略语

- BERT: Bidirectional Encoder Representations from Transformers, 来自Transformer的双向编码表征。
- CNN: Convolutional Neural Networks, 卷积神经网络。
- DE: Deutsch, 德文。
- EN: English, 英语。
- EOS: End of Sentence, 句子末尾。
- FR: French, 法语。
- LSTM: Long Short Term Memory, 长短期记忆网络。
- NLEU: Bilingual Evaluation Understudy, 双语替换评测。
- NLP: Natural Language Processing, 自然语言处理。
- RNN: Recurrent Neural Network, 循环神经网络。
- SOTA: State-of-the-Art, 最先进的。

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。
Bring digital to every person, home, and
organization for a fully connected,
intelligent world.

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.

