Chapter 9 Deep Feedforward Networks

- 定义
  - 目标为拟合某个函数f，即定义映射y=f(x;θ) 将输入x转化为 某种预测的输出y，并同时学习网络参数θ的值，使模型得到最优的函数近似。
  - 由于从输入到输出的过程中不存在与模型自身的反馈连接，此类模型被称为前馈
  - 深度前馈网络是一类网络模型的统称
    - 多层感知机MLP：Multi-Layer Perceptron
      - 采用逻辑回归(即不带隐藏层的感知机)无法精确学习出一个输出为异或的模型表示
      - 包含n元输入的任意布尔函数可以唯一表示为析取范式DNF: Disjunctive Normal Form的形式（由有限个简单合取式构成的析取式）
        - 对于二元输入的情况，需要3个结点可以完成一次异或操作
        - 对于单隐层的感知机，需要2^(n-1)个隐节点实现。
        - 对于多隐层结构可以将隐节点的数目从指数级O(2^(n-1))直接减少至线性级O(3(n-1))，需要的最少网络层数为2log2N(向上取整)
    - 自编码器
    - 限制玻尔兹曼机
    - 卷积神经网络等
  - 生物学研究表明，大脑皮层的感知与计算功能是分层实现的
    - 组成
      - 卷积层叠加若干全连接层
      - 中间包含各种非线性操作和池化操作
    - 稀疏交互(Sparse Interaction)
      - 卷积核尺度远小于输入的维度，这样每个输出神经元仅与前一层特定局部区域内的神经元存在连接权重，即产生交互，称这种特性为稀疏交互
      - 物理意义是通常图像、文本、语言等现实世界中的数据都具有局部的特征结构，可以先学习局部特征，然后再见局部特征组合起来形成更复杂和抽象的特征
    - 参数共享Parameter Sharing
      - 指在同一个模型的不同模块中使用相同的参数，它是运算的固有属性
      - 物理意义是使得卷积层具有平移等变性
        - 当函数f(x)与g(x)满足f(g(x))=g(f(x))时，我们称f(x)关于变化g具有等变性
          - 也就是说，在猫的图片上先进行卷积，再向右平移像素的输出，与先将图片向右平移像素再进行卷积操作的输出结果是相等的
    - 池化
      - 针对非重叠区域
        - 本质上是降采样，能显著降低参数量，还能够保持对平移、伸缩、旋转操作的不变性
        - 均值池化(mean pooling)
          - 通过对邻域类特征数值求平均值来实现，能够抑制由于邻域大小受限造成估计值方差增大的现象
          - 对背景的保留效果更好
        - 最大池化(max pooling)
          - 通过邻域内特征的最大值来实现，能够抑制网络参数误差造成估计均值偏移的现象
          - 更好地提取纹理信息
      - 针对重叠区域
        - 相邻重叠区域的池化
          - 是采用比窗口宽度更小的步长，使得窗口在每次滑动时存在重叠的区域
        - 空间金字塔池化
          - 主要考虑了多尺度信息的描述
            - 列如同时计算1*1,2*2,4*4的矩阵的池化并将结果拼接在一起作为下一网络层的输入
- 深度残差网络ResNet: Deep Residual Network
  - 对前馈神经网络的改进，通过残差学习的方式缓解了梯度消失现象，可以训练更深层的网络
  - 包含两层的神经网络输出：H(x) = F(x) + x
    F(x)被设计为只需要拟合输入x与目标输出H(x)的残差H(x)`-x就行，残差网络的名称也因此而来
  - 解决梯度消失的原因
    - 如果某一层的输出已经较好的拟合了期望结果，那么多一层不会使得的模型变差，因为该层的输出将直接被短接到原层之后，相当于直接学习了一个恒等映射，而跳过的两层只需要拟合上层输出和目标之间的残差即可
- 反向传播算法BP：Back Propagation
  - 前向传播
    - 输入信号通过各个网络层的隐节点产生输出的过程
    - 在网络训练中，前线传播最终产生一个标量损失函数，反向传播算法则将损失函数的信息沿网络层向后传播用以计算梯度，达到优化网络参数的目的
  - 平方损失函数
    - 适合连续输出，并且最后一层不含Sigmoid后Softmax激活函数的神经网络
      - 因为会导致梯度很小，使得基于梯度的学习速度非常缓慢
      - 对比
    - $$J(W,b)=\frac{1}{2}\left\|y-a^{(L)}\right\|^2=\frac{1}{2}\left\|y-f(z_j^{(L)})\right\|^2$$
      $$\delta^{(L)}=-(y-a^{(L)})f'(z^{(L)})$$
      公式和相对于最后一层的梯度
  - 交叉熵损失函数
    - 适合二分类或多分类的场景
      - 最后一层输出层损失函数关于Softmax的导数是线性的
    - $$J(W,b)=-\sum_{k=1}^{n}y_k\ln a_k^{(L)}=-\sum_{k=1}^{n}y_k\ln f(z_k^{(L)})$$
      损失函数
    - $$J(W,b)=-\ln a_{k'}^{(L)}$$
      $$\delta^{(L)}=-\frac{f'(z_{k'}^{(L)})}{f(z_{k'}^{(L)})}$$
      在分类问题中，y_k仅在一个类别k时取值为1，其余为0，设实际为k'，则损失函数变为
    - $$\delta^{(L)}=f(z_k^{(L)})-1=a_{k'}^{(L)}-1$$
      用Softmax激活函数时，f'(x) = f(x)(1-f(x))

- 神经网络训练技巧
  - 激活函数
    - 线性模型可以高效且可靠地拟合数据
    - 实际中有很多线性不可分的情况，需要非线性变化对数据的分布进行重新映射
    - 对于DNN，在每一层线性变换后叠加一个非线性激活函数，以避免多层网络等效于单层线性函数，从而获得更强大的学习与拟合能力

Name	Plot	Equation	Derivative
Identity		$f(x)=x$	$f'(x)=1$
Binary step		$f(x)=\begin{cases}0 & \text{for } x<0\\ 1 & \text{for } x\geq 0\end{cases}$	$f'(x)=\begin{cases}0 & \text{for } x\neq 0\\ ? & \text{for } x=0\end{cases}$
Logistic (a.k.a Soft step)		$f(x)=\frac{1}{1+e^{-x}}$	$f'(x)=f(x)(1-f(x))$
TanH		$f(x)=\tanh(x)=\frac{2}{1+e^{-2x}}-1$	$f'(x)=1-f(x)^2$
ArcTan		$f(x)=\tan^{-1}(x)$	$f'(x)=\frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x)=\begin{cases}0 & \text{for } x<0\\ x & \text{for } x\geq 0\end{cases}$	$f'(x)=\begin{cases}0 & \text{for } x<0\\ 1 & \text{for } x\geq 0\end{cases}$
Parameteric Rectified Linear Unit (PReLU)[11]		$f(x)=\begin{cases}\alpha x & \text{for } x<0\\ x & \text{for } x\geq 0\end{cases}$	$f'(x)=\begin{cases}\alpha & \text{for } x<0\\ 1 & \text{for } x\geq 0\end{cases}$
Exponential Linear Unit (ELU)[12]		$f(x)=\begin{cases}\alpha(e^x-1) & \text{for } x<0\\ x & \text{for } x\geq 0\end{cases}$	$f'(x)=\begin{cases}f(x)+\alpha & \text{for } x<0\\ 1 & \text{for } x\geq 0\end{cases}$
SoftPlus		$f(x)=\log_e(1+e^x)$	$f'(x)=\frac{1}{1+e^{-x}}$

  - 梯度消失
    - 通过图或者导数函数带值计算得出
    - Sigmoid
      - 它将输入x映射到区间(0，1)，当x很大时，f(x)趋近于1，当x很小时，f(x)趋近于0。其导数f'(x) = f(x)(1-f(x))，从斜率或者直接带入x计算都可以得到当x无穷大或者无穷小的时候，f'(x)趋近于0，会产生梯度消失的现象
    - Tanh
      - 它将输入x映射到区间(-1，1)，当x很大时，f(x)趋近于1，当x很小时，f(x)趋近于-1。其导数f'(x) = 1-f(x)^2，从斜率或者直接带入x计算都可以得到当x无穷大或者无穷小的时候，f'(x)趋近于0，会产生梯度消失的现象
      - 实际上Tanh函数相当于Sigmoid的平移，tanh(x) = 2sigmoid(2x)-1
    - 改进
    - Relu
      - 优点
        - Relu的单侧抑制提供了网络的稀疏表达能力
        - Sigmoid和Tanh激活函数均需要计算指数，复杂度高，而Relu只需要一个阈值即可得到激活值
        - Relu的非饱和性可以有效解决梯度消失的问题，提供相对宽的激活边界
      - 局限性
        - 训练过程中会导致神经元死亡问题。Relu函数使得负梯度在经过时神经元被置为0，且在之后也不被任何数据激活，即流经该神经元的梯度永远为0，不对任何数据产生响应。
    - 改进
    - Leaky Relu(LRelu)
      - 当z<0时其值不为0，而是一个斜率为α(很小的正常数)的线性函数，这样既实现了单侧抑制，又保留了部分负梯度信息以敢不完全丢失
      - 但是α的选择增加了问题难度，需要较强的人工先验或者多次重复训练以确定合适的参数值
    - 改进
    - Parameteric Relu
      - 与LRelu的主要区别是将负轴部分斜率α作为网络中的一个可
  - 参数初始化
    - 不能将神经网络的值全部初始化为0，因为同一层中的任意神经元都是同构的，拥有相同的输入输出，如果参数初始化都为0，那么正反向传播的取值都一样，无法打破对称性，最终同一个网络层中的各个参数还是相同的
    - 实际使用技巧
      - 所以需要对参数随机初始化，打破对称性
      - 偏置为0，不会导致参数对此问题
      - 可以将参数初始化为取值范围(-1/sqrt(d),1/sqrt(d))的均匀分布，其中d是一个神经元接受的输入维度
  - Dropout
    - 在神经网络训练过程中，以一定的概率随机地“临时丢弃“一部分神经元节点，其作用于每份小批量训练数据，由于其随机丢弃部分神经元的抑制，相当于每次迭代都在训练不同结构的神经网络
    - 类似于Bagging方法，Dropout可被认为是一种实用的大规模深度神经网络的模型集成算法
    - 抑制过拟合
      - 对于任意神经元，每次训练中都与一组随机挑选的不同的神经元集合共同进行优化，这个过程会减弱全体神经元之间的联合适应性，减少过拟合的风险，增强泛化能力
      - 训练时采用p概率随机丢弃神经元，在测试时需要将每个神经元的参数预先乘以概率系数p，以恢复在训练中该神经元只有p的概率被用于整个神经网络的前向传播计算
  - 数据归一化预处理
    - 对神经网络输入进行归一化处理
    - 批量归一化Batch Normalization
      - 针对每一层的数据，参数变化更新导致后一层数据的输入数据分布发生了变化，因此需要在每一层之间也增加归一化处理
      - 针对每一批数据，在网络的每一层输入之前增加归一化处理(均值为0，标准差为1)，将所有批数据强制在同一的数据分布下
      - 相当于在每一层输入与上一层输出之间多加了一层计算层，对数据的分布进行额外的约束，从而增强模型的泛化能力
      - 每层输入被限制为均值为0，标准差为1，这降低了模型的非线性拟合能力，因此引入了变换重构和可学习参数a,b，分别为输入数据分布的方差和偏差
      - $$\mu_\beta \leftarrow \frac{1}{m}\sum_{i=1}^{m}x_i$$
        $$\sigma_\beta^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i-\mu_\beta)^2$$
        $$x_i' \leftarrow \frac{x_i-\mu_\beta}{\sqrt{\sigma_\beta^2+\epsilon}}$$
        $$y_i \leftarrow \alpha x_i'+b \equiv BN_{\gamma,\beta}(x_i)$$
        公式

2018.12.22 Saturday. By KuKuXia@github.com

ML/DL/RL深度交流讨论群: 622545311，加群时请备注: 姓名-方向-公司or高校