深度学习

深度学习基础

BN

- 为什么要有BN：1. internal covariate shift的问题，在浅层参数微小变化之后，多层叠加之后，得到的中间数据会发生较大的变化，导致在深层的输入分布变化会非常剧烈，从而要很多的初始化，学习率之类的trick，网络变得很难训练。
- Normalization的具体做法：1. BN二维就是在batch方向上norm，三维是在Batch方向上 2. 基本思想是：在将 $\mathbf{x}$ 送给神经元之前，先对其做**平移和伸缩变换**， 将 $\mathbf{x}$ 的分布规范化成在固定区间范围的标准分布。然后再一个平移加缩放使得每个神经元对应的输入范围是针对该神经元量身定制的一个确定范围（均值为 $\mathbf{b}$ 、方差为 $\mathbf{g}^2$ ）3. LN二维是在样本方向上，一个样本内做Normalization；三维是在channel方向上Normalization。（BN batch级别上，LN是在单个样本级别上）
- Batch normalization原理。先归一化然后恢复有何意义？1. 两次缩放的意义在于先归一化。然后 rescale 和 reshift 的参数都是可学习的，一方面主要是为了保证模型的表达能力不因为规范化而下降 2. 另一方面的重要意义在于保证获得非线性的表达能力。Sigmoid 等激活函数在神经网络中有着重要作用，通过区分饱和区和非饱和区，使得神经网络的数据变换具有了非线性计算能力。而第一步的规范化会将几乎所有数据映射到激活函数的非饱和区（线性区），仅利用到了线性变化能力，从而降低了神经网络的表达能力。而进行再变换，则可以将数据从线性区变换到非线性区，恢复模型的表达能力。 3. 经过这么的变回来再变过去，会不会跟没变一样？答案是不会的。因为，再变换引入的两个新参数 g 和 b，可以表示旧参数作为输入的同一族函数，但是新参数有不同的学习动态。在旧参数中， $\mathbf{x}$ 的均值取决于下层神经网络的复杂关联；但在新参数中，$\mathbf{y} = \mathbf{g} \cdot \hat{\mathbf{x}} + \mathbf{b}$ 仅由 $\mathbf{b}$ 来确定，去除了与下层计算的密切耦合。新参数很容易通过梯度下降来学习，简化了神经网络的训练。
- BN的作用：1. 不依赖初始化，可以有更大的学习率，更快 2. 可以给更深的网络训练，缓解 Internal Covariate Shift 3. 减少梯度爆炸和消失的可能性
- BN和LN的区别和适用地方：1. BN比较依赖batch的大小，对数据长短不一不适应（RNN句子），LN针对单个数据样本，适用mini-batch较小时的场景、动态网络结构和RNN。
- BN要注意的点 BN在train和test的作用
- LN,WN：https://zhuanlan.zhihu.com/p/33173246和https://blog.csdn.net/ft_sunshine/article/details/99203548和 https://zhuanlan.zhihu.com/p/72589565

激活函数

- 介绍下各类激活函数和其优缺点：1. sigmoid优点是简单容易理解，缺点是饱和，非零中心，指数计算慢 2. relu优点非饱和，接近线性，所以使得就是训练速度很快，缺点在x小于0的时候，梯度0导致神经元失活，leakly relu 主要就是解决relu死亡的问题，但似乎不是很稳定 3. tanh优点零中心，缺点和sigmoid的缺点一致梯度饱和，计算慢 4. maxout是对relu和leakly relu的一般归纳，缺点参数多 5. softmax 不太算一个激活函数，一个归一化函数
- 各种激活函数的公式；sigmoid求导； softmax 求导 **<手推>**https://blog.csdn.net/bqw18744018044/article/details/83120425

- 反向传播推导 **<手推>**

- 介绍下dropout：1. 在训练的时候，随机设置一定概率的神经元不训练 2. 作用（防止过拟合，类似bagging集成，） 3. 注意的点，加了训练变慢；一般放在全连接层；卷积层不加，原因主要是卷积核参数本来就少，不同神经元高度相关的，不适合dropout;对RNN 放在输入->RNN与RNN->输出的位置。

深度学习优化--炼丹

各种优化器介绍博客：https://zhuanlan.zhihu.com/p/32626442和https://zhuanlan.zhihu.com/p/33160298

- 介绍一些优化器：1. SGD，优点简单容易理解，缺点收敛慢，容易产生鞍点动荡，2. 基于动量更新，也就是在SGD的基础上加入速度的概念，可以减少在谷底的震荡，更快收敛 3. Nesterov Accelerated Gradient ，和基于动量更新类似，但优化了对于速度的计算，进一步的加快收敛。**以上三个优化器不能自适应的降低学习率** 4. 自适应Adagrad： 在SGD的基础上加入对应梯度大小的自适应衰减，缺点容易过早的停止学习 5. 自适应RMSprop：简单的修改了Adagrad的学习率衰减方式，使得学习率平滑衰减，防止过早停止学习 6. 自适应Adam，就是结合RMSprop加Momentum的方式，缺点 **以上三个优化器是自适应优化器** 7. 最新优化器LookAhead：在训练整个过程中稳定性，更快收敛 8. 最新优化器RAdam：整流函数来确定一个“启发式的 Warmup”，解决之前自适应算法中都需要手动热身的需要，使得训练更加的稳定 9. 最新优化器Ranger，就是RAdam+LookAhead的改进，使得整体效果更好 **以上三个是优化器的三个代表性最新研究**

> 优化器最新进展：https://mp.weixin.qq.com/s/C5TRMLWMx0ZhLz-TEFCIYw

- **<写出各种优化器公式>**
- 说下adam的一些特点：1. 更新步长和梯度大小无关，只和alpha、beta_1、beta_2有关系。并且由它们决定步长的理论上限 2. 对目标函数没有平稳要求，即loss function可以随着时间变化 3. 能较好的处理噪音样本，并且天然具有退火效果 4. 能较好处理稀疏梯度
- 梯度下降陷入局部最优有什么解决办法：1. 假如数据足够多，即使是局部最优，也是极好的解，而数据太大的时候，只有神经网络加随机梯度下降才能hold住 2. 网络足够深的时候，局部最优没那么局部，往往以鞍点存在，此时优化算法可以部分解决 3. 通过调整学习率等，可以部分避免局部最优 4. 要求实现全局最优解的话可以考虑加入退火算法或者遗传算法之类的思想，简单说就是在搜索过程中不但有基于梯度下降的方向，同时也融入少量的逆向搜索，最终设定一个收敛域即可。

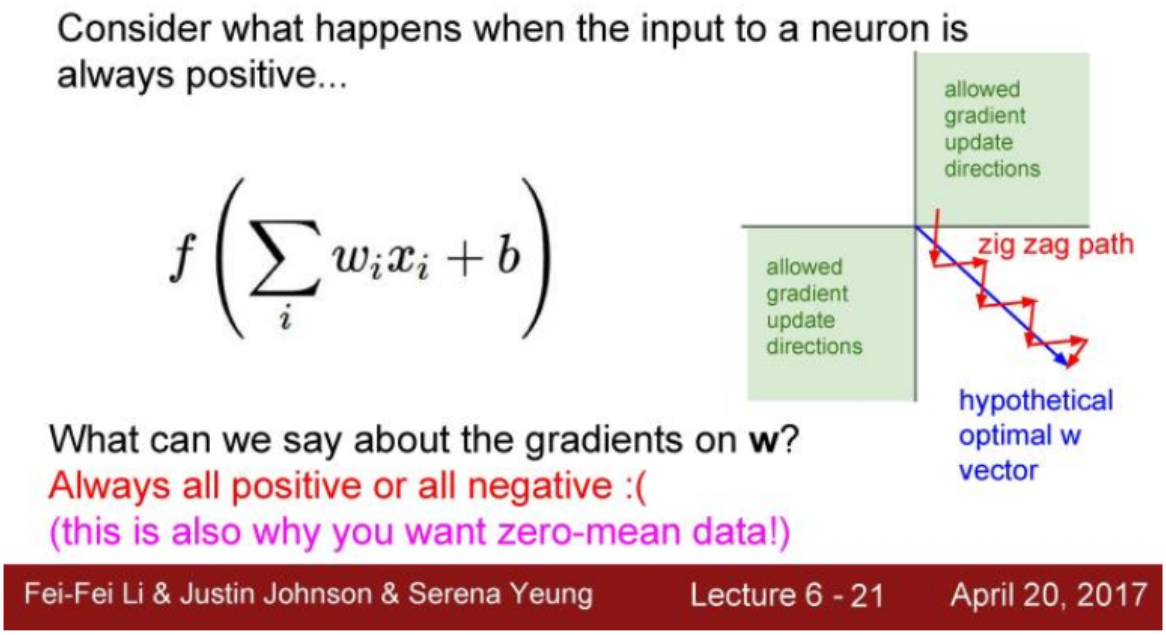
  > 参考: https://www.zhihu.com/question/68109802

- 请问你有哪些训练深度神经网络的经验：1. 在实现论文或者从论文改进的时候，一般都是先按照论文设置参数 2.检查模型写的是否正确，用少量数据或者减少训练类别来训练，查看效果如何 3. 从粗到细分阶段调参，实验结果影响比较大的参数开始，同时固定其他参数，比如，一个顺序先调整学习率，网络层数，每个多少个神经元，batch_size，词向量大小，正则的参数，正负样本比例等等。4. 调整学习率从高到低0.1 0.01 0.001。每层节点数一般是16，32，128.batch_size一般就是GPU对2的幂次的batch可以发挥更佳的性能或者8的倍数，GPU并行效率最高。dropout一般就是0.5，词向量一般是128或者256。5. 自动化调参Gird Search，太费时间，贝叶斯优化，考虑到了不同参数对应的实验结果值，这个还没有试过。

激活函数零中心的问题

但是这个问题似乎并不严重，因为优化过程使用的是Mini-batch SGD方法，不同的输入 x 可能有或正或负的参数梯度，因而其和并不会很严重地偏离最优解的方向。

当输入神经元的数值始终为正时，再乘以权值W，这时结果会要么全部是正数，要么全部是负数。这时把上游梯度传回来，就是损失L对f进行求导， $\partial L/\partial f$ 为正数或者负数，当任意传回来一些梯度乘以本地梯度，求关于权值w的梯度 $\partial f/\partial w = x$ ，如果x的值总是正数，对w的梯度就等于 $\partial L/\partial f$ 乘以 $\partial f/\partial w$ ，此时就相当于把上游梯度的符号传回来，这意味着所有关于w的梯度全为正值或者全为负值，那么它们就会总是朝着同一个方向移动。当在做参数更新的时候，可以选择用同一个正数或者用不同的正数去增加所有w的值，或是用类似的方法减小w的值，这时会出现的问题是，这样更新梯度是非常低效的。看到下面右图的例子，假设w是二维的，所以可以用平面坐标轴来表示w，如果用全为正或者全为负的数去进行更新迭代。我们得到两个象限，即在坐标轴上有两个区域，一个全为正（一象限）和一个全为负（三象限），这是根据梯度更新得到的两个方向，假设最好的w是图中蓝色向量，而实际的迭代过程是如红色箭头所示，可以看到不能沿着w这个方向直接求梯度，因为这不是允许的两个梯度方向中的一个。所以在一般情况下要使用均值为0的数据，我们希望输入x的均值为0，就能得到正和负的数值，这就不会陷入上述梯度更新会出现的问题，因为这将会沿着同一个方向移动。



* 这里举一个简单例子，说明sigmoid函数的输出非零中心导致反向传播回w的梯度为全正或者全负是有问题的。假设我们有权值 $w = [1, -1, 1]$ ，我们需要将权值更新为 $w = [-1, 1, -1]$ ，如果梯度是同时有正和有负的，我们可以只更新一次就可得到结果： $w = [1, -1, 1] + [-2, 2, -2] = [-1, 1, -1]$ ；但是如果梯度只能是正或者只能是负，则需要两次更新才能得到结果： $w = [1, -1, 1] + [-3, -3, -3] + [1, 5, 1] = [-1, 1, -1]$ 。

DNN

- 普通的DNN网络如何设计的，隐层规模、隐藏单元：1.

CNN

- 如何理解卷积：1. 一个函数作用于另外一个函数的结果，在连续情况下是做积分 2. 在卷积神经网络中也就是离散情况下，做的是加权求和，这个和全连接神经网络没有区别
- 介绍卷积神经网络结构：1. 卷积层图像一般具有平移不变性的特征，通过参数共享可以大大减少参数，同时可以加深网络 ，第二是通过局部链接，而不是全连接来减少参数。2. 卷积神经网络一般有pooling层，这一层是用来整合和对特征数降维作用的。
- 讲讲对于1V1卷积，它的作用：1v1卷积在NIN首次出现，在NIN中的主要作用是整合通道特征，在vggnet也有，在GoogleNET是实现特征数的降维，可以减少参数，在残差网络中，1v1卷积同样用到，不过在残差网络中，1v1卷积不仅用于降维还用于升维，进一步的整合特征和减少参数。
- 介绍一下常见的CNN网络结构：根据imagenet比赛 1. alexnet，用激活函数relu代替之前的激活函数tanh或者sigmoid，这样可以大大的提高速度（论文中是比tanh快6倍），提出Local Response Nomalization，一种Nomalization技术。 2. VGG，和alexnet差不多使用更小的尺寸

和间步长 3.NIN，提出mlpcon结构，使用全局均值池化代替之前CNN中的全连接层 4. googlenet：提出Inception结构 5. 残差网络：将残差计算的思想嵌入CNN中，更深。

- 如何计算CNN卷积层的参数： 1. **参数个数:** 与神经元的个数无关，只与卷积核的大小及Feature Map的个数相关。参数的个数=卷积核的长*宽*深度*Feature Map的个数
- 介绍下inception结构：1. 解决网络深度变得很深所带来的计算量和性能问题。采用Inception的架构的两个主要优点，2. 一是允许显著增加每一步的单元数目，计算复杂性不会不受控制。降维的普遍使用能保护最后一步到下一层的大量输入滤波器，在对它们 用大的patch size卷积前首先降维。 3. 二是视觉信息在不同的尺度上进行处理然后聚合，这样下一步可以同时从不同尺度提取特征。 采用了Inception模块的网络要比没有采用Inception模块的同样架构的网络快2~3倍。
- Resnet好在哪里？1. 残差块结构可以使得网络大大增加深度，防止深度网络梯度弥散或梯度爆炸所产生的难以训练 2. 残差网络并不是一个真正意义上极深的网络，而是隐式地由指数个大部分为浅层网络叠加而成的。由此该论文指出，查看网络之后除了可以看width和depth，其实还有另外一个维度就是multiplicity。残差网络和dropout有类似的功能，这样集成正则化的方式，也是能有效采用这样的加深网络的方式来提升模型性能。
- 讲讲讲一下textcnn：1. 卷积神经网络的核心思想是**捕捉局部特征**，对于文本来说，局部特征就是**由若干单词组成的滑动窗口**，类似于N-gram。卷积神经网络的优势在于能够**自动地对N-gram特征进行组合和筛选，获得不同抽象层次的语义信息。** 2. 输入层词向量输入，卷积层一般多个不同宽度卷积核，也就是在一个句子中不同范围的词出现会带来什么信息。比如可以使用3,4,5个词数分别作为卷积核的大小，**每一次卷积操作相当于一次特征向量的提取，通过定义不同的窗口，就可以提取出不同的特征向量，构成卷积层的输出。** 池化层就是普通的1-Max池化和K-Max池化

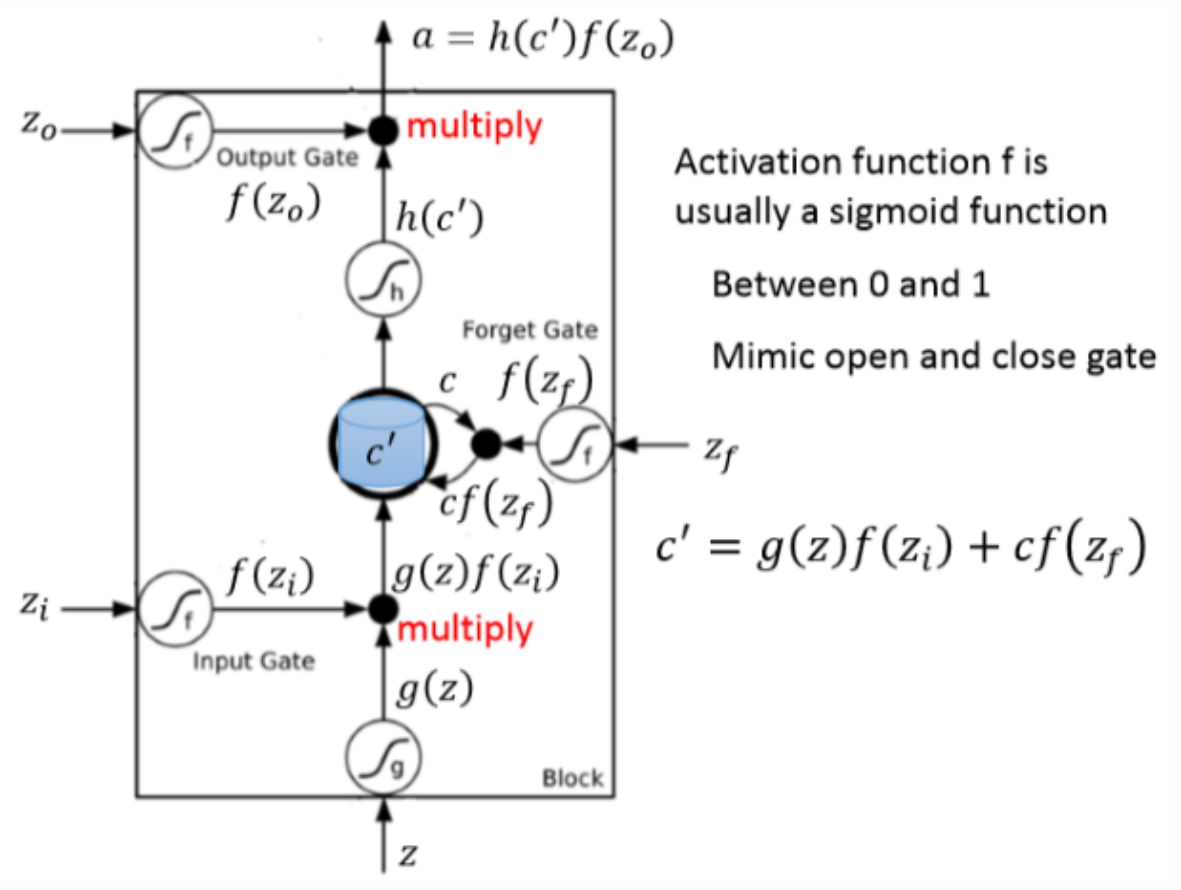
> CNN各种架构 https://zhuanlan.zhihu.com/p/33528315

RNN

- 介绍一下RNN：1. RNN也就是循环神经网络，主要用来序列数据，传统的RNN包括输入层，中间隐层，输出层，在时间上共享参数U,V,W（是输入-输出，隐层-隐层，隐层-输出直接的参数） 2. 由于RNN不能有效处理长序列，梯度消失或者爆炸的问题，提出lstm。
- 介绍下LSTM和GRU：1. LSTM也就是长短期记忆网络，包含cell的状态和隐状态，其中cell的状态像是一条传送带，它贯穿整条链，其中只发生一些小的线性作用 2. lstm包含三个门，输出门，遗忘门和输出门（**<手推>**各个公式）3. GRU在lstm的基础上做了简化，去除了cell状态，并合并了输出和遗忘门成更新门（**<手推>**各个公式），GRU相对lstm有更少的参数，但是效果不输于lstm。
- 为什么LSTM模型中既存在sigmoid又存在tanh两种激活函数？sigmoid 用在了各种gate上，产生0~1之间的值；tanh 用在了状态和输出上，是对数据的处理，这个用其他激活函数也是可以的
- **<手推>**LSTM和的前向各个公式，手推BPTT和为什么可以方式梯度消失,RNN的BPTThttps://zhuanlan.zhihu.com/p/26892413和https://zhuanlan.zhihu.com/p/85776566

关于LSTM中，词向量维度和hidden size的关系。https://blog.csdn.net/ys_1991/article/details/88422771

lstm结构图：



GAN

- 介绍下GAN：1. GAN在结构上有两个网络，G（Generator）和D（Discriminator），其中Generator输出噪音分布Z输出生成图像，Discriminator是一个判别网络，判别一张图片是不是“真实的”。2. 训练过程中，G网络和D网络利用不同的目标函数进行交替训练，直到**纳什均衡**，也就是判别模型再也判别不出来结果，准确率为 50%。
- 一些常见的GAN网络：DCGAN，Conditional GAN
- 讲述下GAN在NLP中是如何应用的：1.GAN for nlp的主要困境体现在原始GAN论文中也提到GAN在离散型数据上是不起作用的2. 思路一，利用Gumbel-softmax代替softmax，这种softmax能够直接给出近似Sampling操作的输出，但又是可微分的 3. 思路二是WGAN-GP，4. 结合强化学习，代表论文是 seqgan，利用RL来改变GAN的优化方式，比如使用Policy Gradient算法来优化。

GCN

- 介绍下GCN：1. 一种能对图数据进行深度学习的方法。之前的深度学习都只是擅长处理有规则的空间结构的数据，比如图像，句子等，但是生活中还是有很多并不具备规则的空间结构，我们可以用图来表示这些不规则的数据。2. 图卷积算法主要分为三步，分别是每一个节点将自身的特征信息经过变换后发送给邻居节点。这一步是在对节点的特征信息进行抽取变换；每个节点将邻居节点的特征信息聚集起来。这一步是在对节点的局部结构信息进行融合。；把前面的信息聚集之后做非线性变换（图卷积算法），增加模型的表达能力。3.
- GCN在NLP方面的应用：1. textGCN 2. 3. our paper