网络压缩探幽 (二)

陈超
南京大学

前言

上周初步了解网络结构设计, 本周将着重关注网络压缩的其他方法. 主要包括四个方面:

- 量化 (quantization). 通过减少每个权值所需的比特数来压缩原始网络.
- 低秩分解 (low-rank factorization). 用于提取矩阵或张量的中的主要信息.
- 剪枝 (pruning). 用于去除网络中冗余的参数.
- 知识蒸馏 (knowledge distallation). 用小模型浓缩大模型中的知识.

以上方法可以组合使用.

除了掌握理论知识，还完成了五个实验. 实验使用的框架为 Pytorch, 用到的数据集有 mnist 和 cifar10.

量化

量化有两种方式，一种是直接使用位数少的定点数进行网络训练，比如 1 比特/2 比特/4 比特/8 比特/16 比特. 另一种是先用实数训练再构造一个字典，权重用字典的索引表示.

第一种方式的代表是二值化网络 (Binary Neural Network, 简称 BNN)[1]. BNN 用-1 和 +1 表示权重和激活值，-1 在计算机中可以用 0 表示. 浮点数有 32 位，因此 BNN 理论上可实现 32 倍压缩.

第二种方式的代表是乘积量化 (Product Quantization, 简称 PQ)[2]. 设权重矩阵 $W \in R^{C_{in} \times C_{out}}$, W 含 C_{in} 个维度为 C_{out} 的向量，将向量划分为 m 个维度为 d 子向量,$d = C_{out}/m$, 这样 W 被划成 $[W^1, \cdots, W^m]$, 其中 $W^i \in R^{C_{in} \times (d)}$. 设每组 k 个聚类中心，w_z^i 表示 W^i 的第 z 行，c_j^i 表示 W^i 的第 j 个聚类中心，$C^i \in R^{k \times (d)}$ 表示 W^i 的字典. 优化目标如下:

$$\min \sum_{z}^{C_{in}} \sum_{j}^{k} ||w_z^i - c_j^i||_2^2$$

求解的方法为 kmeans. 理论上 PQ 的压缩倍数为 $(32C_{in}C_{out})/(32kC_{out} + \log_2(k)C_{in}m)$

下面分别验证这两种方法的效果.

实验:BNN

实验代码见`github.com/passerer/NetworkCompression/blob/master/BinaryNet.ipynb`

二值化有两种方法，这里采用确定性二值化:

$$w = +1 \quad if \quad w \geq 0 \quad else \quad -1$$

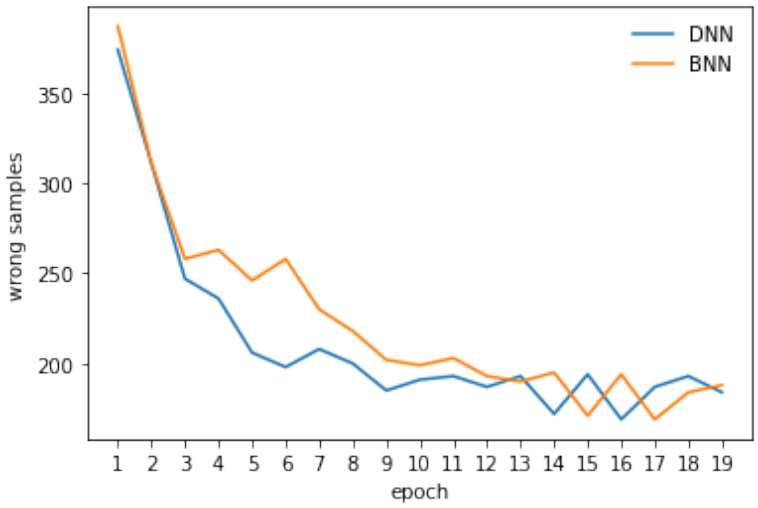
还有一种随机二值化能够使输入依概率变成-1 或 +1. 由于需要产生概率对简单的设备不友好，故不采用.

前向推断的时候不论权重还是激活值都要二值化，但是最后一层的输出不用，因为它表示分类的概率.

反向传播也是在二值化的结果上进行求导. 由于二值化函数不可导，用 hard tanh 函数来模拟.

更新权重时是基于实数权重的，因为发现直接用二值权重减去梯度效果不好. 因此训练的过程中要保留两份权重，一份为二值权重，一份为实数权重.

本实验设计含三层隐藏层的全连接网络，用于对比 BNN 和普通 DNN 在 mnist 上的表现效果. 每个隐藏层均含 4096 个神经元. 为了加速训练，使用 bn 层. 选择 Adam 作为优化方法,HingeLoss 为损失函数. 下图为收敛曲线.



从图中可以看到，BNN 相比于普通的 DNN 收敛得慢. 值得一提的是，为了保证网络的表达能力，BNN 的神经元个数要足够多. 如果将 BNN 倒二层的神经元个数砍掉一半时会发现正确率大幅下降.

实验:PQ

实验代码见`github.com/passerer/NetworkCompression/blob/master/PQ.ipynb`

同样是用 mnist，设计了含三层 512 个单元隐藏层的全连接网络，原始正确率为 98.3%. 取不同的 m 和 k，PQ 操作后正确率如下表所示:

accuracy(%)	m=1	m=4	m=16	m=64
k=10	17.9	23.1	34.2	64.5
k=20	28.7	39.5	63.8	90.3
k=50	41.9	59.3	90.0	96.6

在 k=50,m=64 的情况下，权重大约压缩 9 倍，而正确率下降 1.7%. 以上只是粗略的网格搜索，肯定还有更优的 k/m 取值.

低秩分解

卷积层和全连接层的参数在网络中占大头. 全连接层的权重是二维矩阵，可以用奇异值分解 (SVD). 如果矩阵的秩越小，那么权重就能压缩得越小. 而卷积层的权重是四维张量，就不能应用 SVD 了，得使用其他手段. 常用的方法是 CP 分解.

一般的，设一个张量 W 的大小为 $n_1 \times n_2 \times \cdots \times n_d$，则其可分解为

$$\hat{W} = \sum_{r}^{R} W_1(r) \times \cdots \times W_d(r)$$

其中 $W_i(r)$ 为一维张量，$\times$ 表示外积，R 可视为张量的秩. 优化目标为:

$$\min ||W - \hat{W}||_2^2$$

解法为交替方向乘子法 (ADMM)
CP 分解的理论压缩倍数为 $(n_1 n_2 \cdots n_d)/R(n_1 + n_2 + \cdots + n_d)$

实验：SVD

实验代码见`github.com/passerer/NetworkCompression/blob/master/SVD.ipynb`
本实验设计含有三层 512 个神经元的隐藏层的全连接网络，使用 mnist 数据集. 网络原始准确率为 98.1%.SVD 后得到下表中的权重重构误差和准确率:

选取特征值的比例	平均重构误差	准确率
50%	0.013	98.1%
10%	0.039	97.4%
5%	0.048	95.6%
3%	0.054	90.8%

当选取前 5% 大的特征值时，网络参数大约压缩 10 倍，正确率下降 2.5%.

剪枝

剪枝可以在不同级别上实现，weight-level 可以实现最高的压缩比率，但需要依赖于特殊的硬件. layer-level 不够灵活，且只有在网络层数足够深时才有效. channel-level 在灵活性和通用性上作了平衡. 这里主要介绍 channel-level 的剪枝[3]. 卷积层和全连接层后面常常跟着 BN 层，可以根据 BN 层中缩放因子 γ 的 l1 范数判断通道的重要性然后作取舍. 剪掉不重要的通道有时候虽然会暂时降低性能，但是通过之后对网络的微调可以补偿精度. 剪枝 - 微调的过程可以反复进行.
为了增强稀疏性，在 BN 层中引入对 γ 的惩罚.

实验：channel slimming

实验代码见`github.com/passerer/NetworkCompression/blob/master/Pruning.ipynb`
实验在 cifar10 数据集上进行. cifar10 含有 10 类物体的 6 万张图像，其中 5 万张为训练集，1 万张为测试集. 用 vgg11 训练. 训练后将所有通道的 γ 按大小的排列，按比例保留 γ 较大的通道，再微调. 由于时间和算力有限，剪枝 - 微调的过程只循环一次. 下表为结果，其中第三栏为剪枝后未微调的准确率，第四栏为剪枝后迭代训练 10 个 epoch 的准确率.

原网络准确率	保留通道比例	剪枝网络准确率	微调网络准确率 (迭代 10 轮)
89.12%	90%	89.00%	88.56%
	70%	74.32%	88.31%
	50%	22.18%	87.71%
	35%	10.02%	86.57%

从表中可以看出 vgg11 网络有很多冗余的通道，剪枝后重新训练也可以恢复相似的精度. 当保留 35% 的通道时准确率降为 10% 左右，相当于随机初始化，此时微调网络和重新训练网络的效果应该差不多. 所以剪枝的作用可能更多在于结构搜索. 实验发现网络中间的通道被剪掉的较多，推测中间的通道可能不如首部和尾部的通道重要.

蒸馏

在训练的过程中，大模型不仅学习到如何判断样本的类别，还学习到不同类别间的联系和差别，后者是训练集无法提供的多出来的知识，可以用软目标体现出来[4]．因此小模型能从大模型那学得更多的知识.

软目标可以用带' 温度' 的 softmax 表示：

$$p_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

T 越大，不同概率差异越小，模型更易于捕获小概率类别的信息.

为了少继承大模型的偏见 (bias)，小模型可以同时在训练集上进行学习．具体做法是采用两个不同的目标函数，第一个目标函数是和复杂模型的软目标做的一个交叉熵，使用较高的温度；第二个目标函数是和正确标签的交叉熵，温度设置为 1．对两个目标函数进行权重平均.

实验：distalling

实验代码见`github.com/passerer/NetworkCompression/blob/master/Distallation.ipynb`

本实验使用 mnist 数据集，设计含两层 1200 个神经元的隐藏层的教师网络，和含一层 100 个神经元的隐藏层的学生网络．同样迭代 20 轮，教师网络的分类错误 202 个，学生网络在没有教师网络指导的情况下分类错误 380 个．在教师网络帮助下，T=3 时，分类错误 347 个;T=10 时，分类错误 359 个.

可以看到有教师网络的帮助下效果确实更好．但是提升的效果在这里表现不明显.

总结

量化，剪枝，分解，蒸馏，再加上 NAS，是常用的网络压缩方法．目前学者们提出了很多算法大多数离不开这五个方法的套路，只不过某些细节不一样．相关文章过多，需要花时间看，预计下下周把 2019 年之前的部分论文作个比较和总结.

初步了解这些方法后，个人认为有两个值得探索的方向：

- 给网络压缩提供一个理论的下界．网络究竟需要压缩到什么程度才达到极限呢? 如果有下界，不但可以提供算法的终止条件，还可以当作各算法性能比较的标尺．这可能需要从信息论的角度入手.
- 网络可解释性．目前一般通过网络压缩的结果来推测网络的某些特性，其实也可以反过来，利用网络的特性给网络压缩指路.

下周的计划是进一步了解 NAS, 并尝试实现其中一些算法.

References

[1] Courbariaux M, Hubara I, Soudry D, et al. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1[J]. arXiv preprint arXiv:1602.02830, 2016.

[2] Stock P, Joulin A, Gribonval R, et al. And the bit goes down: Revisiting the quantization of neural networks[J]. arXiv preprint arXiv:1907.05686, 2019.

[3] Liu Z, Li J, Shen Z, et al. Learning efficient convolutional networks through network slimming[C]//Proceedings of the IEEE International Conference on Computer Vision. 2017: 2736-2744.

[4] Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network[J]. arXiv preprint arXiv:1503.02531, 2015.