深度学习中的各种 tricks_1.0_label_smoothing

2017/11/16 - Thursday - 第一次修改

[用于整理遇到的NN设计中使用的不同结构和trick的原理与实现]

label smoothing （标签平滑）

在读cleverhans的示例代码时发现code中对于train_Y即训练标签做了label smoothing，于是找到paper中对于该方法的描述和理论分析，如下：

对于分类问题，常规做法时将类别做成one-hot vector，然后在网络最后一层全链接的输出后接一层softmax，softmax的输出是归一的，因此我们认为softmax的输出就是该样本属于某一类别的概率。由于标签是类别的one-hot vector，因此表征我们已知该样本属于某一类别是概率为1的确定事件，而其他类别概率都为0。

softmax：

$$p(k|x)=\exp(z_k)\sum_{i=K}^{i}\exp(z_i)$$

其中 z_i 一般叫做 logits ，即**未被归一化的对数概率** 。我们用 p 代表 predicted probability，用 q 代表 groundtruth 。在分类问题中loss函数一般用交叉熵，即：cross entropy loss:

$$loss=-\sum_{k=1}^{K}q(k|x)\log(p(k|x))$$

交叉熵对于logits可微，且偏导数形式简单：$\partial loss\partial z_k=p(k)-q(k)$ ，显然梯度时有界的（-1到1）。

对于groundtruth为one-hot的情况，即每个样本只有惟一的类别，则 $q(k)=\delta_{k,y}$ ，y 是真实类别。其中 δ 是Dirac函数。要用predicted label去拟合这样的函数具有两个问题：首先，无法保证模型的泛化能力（generalizing），容易导致过拟合； 其次，全概率和零概率将鼓励所属类别

和非所属类别之间的差距尽可能拉大，而由于以上可知梯度有界，因此很难adapt。这种情况源于模型过于相信预测的类别。（ Intuitively, this happens because the model becomes too confident about its predictions.）

因此提出一种机制，即要使得模型可以 less confident 。思路如下：考虑一个与样本无关的分布 u(k) ，将我们的 label 即真实标签 q(k) 变成 q′(k) ，其中：

$$q'(k|x) = (1-\epsilon)\delta_{k,y} + \epsilon u(k)$$

可以理解为，对于 Dirac 函数分布的真实标签，我们将它变成以如下方式获得：首先从标注的真实标签的Dirac分布中取定，然后，以一定的概率 ϵ ，将其替换为在 u(k) 分布中的随机变量。因此可以避免上述的问题。而 u(k) 我们可以用先验概率来充当。如果用 uniform distribution 的话就是 1/K 。该操作就叫做 label-smoothing regularization, or LSR 。

对于该操作的数学物理含义可以用交叉熵的概念说明：

$$H(q',p) = -\sum_{k=1}^{K} \log p(k) q'(k) = (1-\epsilon)H(q,p) + \epsilon H(u,p)$$

可以认为 loss 函数分别以不同的权重对 predicted label 与标注的label 的差距 以及 predicted label 与 先验分布的差距 进行惩罚。根据文章的报告，可以对分类性能有一定程度的提升。（In our ImageNet experiments with K = 1000 classes, we used u(k) = 1/1000 and = 0.1. For ILSVRC 2012, we have found a consistent improvement of about 0.2% absolute both for top-1 error and the top-5 error ）

reference:

1. Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the Inception Architecture for Computer Vision[C]// Computer Vision and Pattern Recognition. IEEE, 2016:2818-2826.

2. https://github.com/tensorflow/cleverhans/blob/master/cleverhans_tutorials/mnist_tutorial_tf.py