Triplet Response Residual(TRR)与特征表示学习

最开始想到把 TRR 用在特征表示学习里是受到了 Prof.Zhang 分享的一个工作[1]的启发。他们的工作是用 TRR 来做 DNN model compression，具体做法是为卷积神经网络的每个 filter(or kernel)计算得到一个 TRR score，然后根据 TRR scores 做一个排序，把 TRR score 低的 filters pass 掉，这样做一方面可以减少卷积神经网络的 filters 数量(即减少了模型参数)，只保留特征学得比较好的一些 filters，另一方面可以减少模型的计算量。正是受到这个工作的启发，才有了一些关于特征表示学习方面的 insight。我们知道在模型参数过多时，容易出现过拟合现象，使得模型只是简单地“**记住**”了数据的特征表示，但其实并没有学到同类数据背后的一些**共性特征**。如果模型可以学会物以类聚，即使得同类数据的特征向量在特征空间下尽量逼近，而不同类数据的特征向量在特征空间下尽量疏远，就能学到同类数据的共性特征，使得模型不再是简单地“记住”数据的特征表示，这样也能提升模型的泛化能力。后面通过实验验证了这个想法，即一个过参数化模型并没有学到共性特征，并且在模型泛化能力上不如引入了 TRR 的模型的泛化能力好。

实验背景

实验数据是通过 sklearn 生成的一个 10 分类数据集。每条数据的特征维度是 100，其中有 20 维的冗余特征。为了在不同大小的数据集上验证模型的性能，分别生成不同大小的训练数据集：10000，50000，100000。验证集大小固定为 10000。为了引入 TRR，设置 Input 数据的格式为：$\{(x_{anc}, x_{neg}, x_{pos}, y)\}$，其中 x_{anc} 和 x_{pos} 是同一个类别下的两个不同数据，x_{neg} 和 x_{pos} 属于不同类别下的数据，y 是 x_{anc} 和 x_{pos} 的类别。

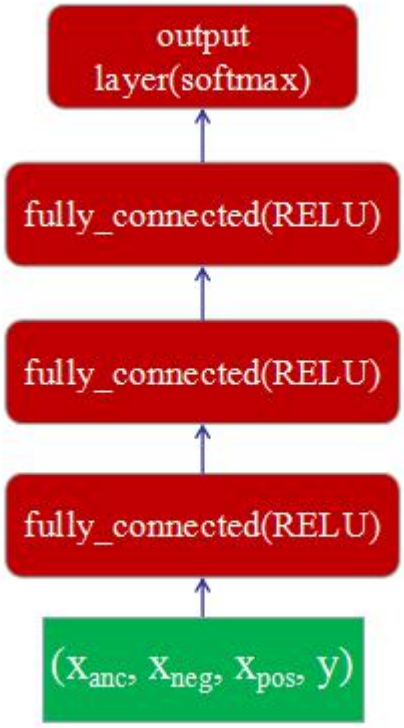

Figure 1 模型架构图

模型架构图如 figure 1 所示，输入为(x_{anc}, x_{neg}, x_{pos}, y)，其中 $x \in R^{100}$ ，$y \in R^{10}$(one hot vector)，然后经过 3 层的 fully connected layer，激活函数是 RELU，最后一层 output layer。具体超参数配置如下：

```
integer('num_class', 10, '')
integer('num_epoch', 50, '')
integer('batch_size', 32, 'batch size')
integer('feature_dim', 100, 'feature dim')
boolean('is_training', True, 'train or predict phase')
integer('num_threads', 4, 'number of threads of enqueueing examples')
integer('load_model', 0, '')
string('mode', 'train', '')

float('lr', 0.001, '')
float('lr_decay_rate', 0.8, '')
float('keep_rate', 0.6, '')
integer('decay_steps', 50, 'decay every 50 steps with decay rate:lr_decay_rate')
integer('start_decay_at', 5, '')
float('grad_clip', 5.0, '')
string('optimizer', 'adam', '')

integer('num_dense_layer', 3, '')
float('alpha', 1.0, 'to control the weight of TRR loss')
```

Triplet Response Residual

原始的 TRR 计算公式为：$TRR_i = \| x^i_{anc} - x^i_{neg} \|_2^2 - \| x^i_{anc} - x^i_{pos} \|_2^2$，i 代表第 i 层，$x^i$ 表示第

i 层 fully connected layer 的输出特征向量，$x^0=x$。它的物理意义是一种衡量同类别数据特征向量距离与不同类别数据特征向量距离的度量标准。Loss function：

$$L(\theta) = -(y \log p_y + (1-y)\log(1-p_y)) - \sum_i (\| x^i_{anc} - x^i_{neg} \|_2^2 - \| x^i_{anc} - x^i_{pos} \|_2^2) \quad (1)$$

但是因为 $TRR_i = \| x^i_{anc} - x^i_{\text{neg}} \|_2^2 - \| x^i_{anc} - x^i_{\text{pos}} \|_2^2$ 这一项比较大，所以训练的时候很容易出现 value overflow，因此修改原始 loss function 为：

$$L(\theta) = -(y \log p_y + (1-y)\log(1-p_y)) + \sum_i sigmoid(\| x^i_{anc} - x^i_{\text{pos}} \|_2^2 - \| x^i_{anc} - x^i_{neg} \|_2^2) \quad (2)$$

其中后面一项即 TRR_loss，后来在训练的过程当中发现，后面一项很容易梯度为 0，所以重新定义 TRR_loss 为：

$$TRR\_\text{loss}(\theta') = \sum_i sigmoid(\| x^i_{anc} - x^i_{\text{pos}} \|_2^2) - sigmoid(\| x^i_{anc} - x^i_{neg} \|_2^2), \theta' \subset \theta \quad (3)$$

这样设计可以避免模型只学到了 $\max(\| x^i_{anc} - x^i_{neg} \|_2^2)$，而学不到 $\min(\| x^i_{anc} - x^i_{pos} \|_2^2)$，同时，由于 $\| x \|_2^2 >= 0$，而 sigmoid 函数在 x>=0 的区域可以近似看成一个凹函数，这样有利于优化。顺带提一下：$sigmoid(a-b) > (sigmoid(a) - sigmoid(b))$，因此新的 TRR_loss 本身是原来的 TRR_loss 的一个下界。所以修改后的 loss function 为：

$$L(\theta) = -(y \log p_y + (1-y)\log(1-p_y)) + \alpha * (\sum_i sigmoid(\| x^i_{anc} - x^i_{\text{pos}} \|_2^2) - sigmoid(\| x^i_{anc} - x^i_{neg} \|_2^2)) \quad (4)$$

。超参数 α 用来控制 TRR_loss 的权重，实验过程中分别设置为 0.0 和 1.0，分别表示不引入 TRR_loss 约束和引入权重为 1.0 的 TRR_loss 约束。

实验结果和分析

在三个不同大小的训练数据集：10000，50000，100000 上的实验结果见下图：

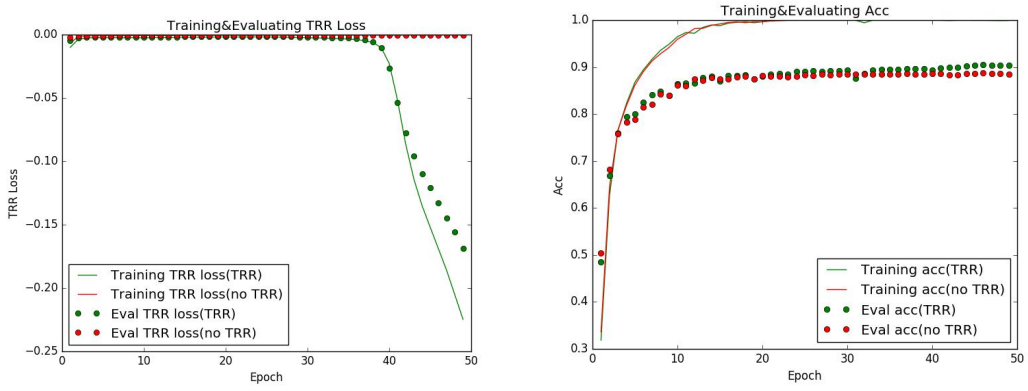

Figure 2: 10000 samples 训练结果(加了 early stopping 策略)

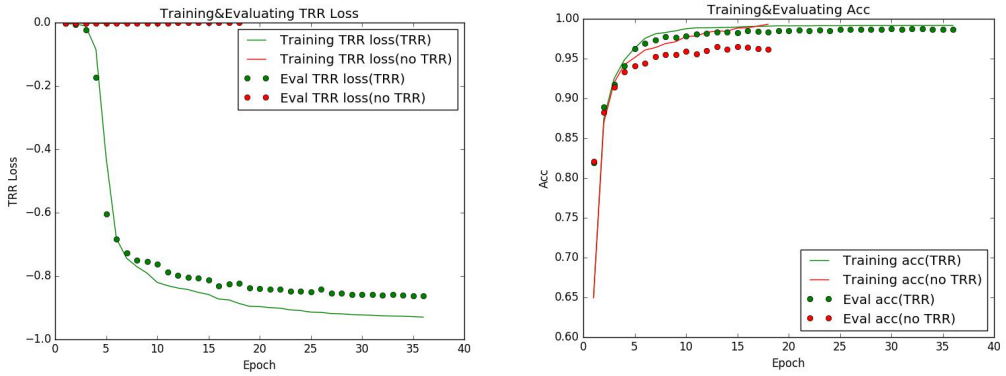

Figure 3: 50000 samples 训练结果(加了 early stopping 策略)

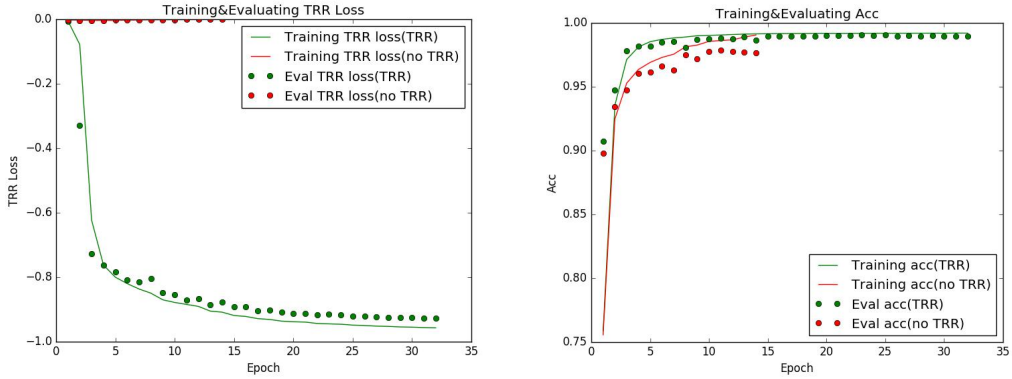

Figure 4: 100000 samples 训练结果(加了 early stopping 策略)

Figure 2、3、4 分别对应 10000,50000,100000 samples 的训练结果，其中绿色实线和点线代表加入 TRR_loss 约束后模型在训练集和验证集上的性能，红色实线和点线代表不加 TRR_loss 约束的模型在训练集和验证集上的性能。从上面的图片可以看出如果没有 TRR_loss 的约束，$TRR\_\text{loss}(\theta^{'}) = \sum_{i} sigmoid(\| x^{i}_{anc} - x^{i}_{\text{pos}} \|^{2}_{2}) - sigmoid(\| x^{i}_{anc} - x^{i}_{neg} \|^{2}_{2})$，$\theta^{'} \subset \theta$ 始终逼近 0(参考 figure 2、3、4 左边图的红色实线和点线)，这意味着 $\| x^{i}_{anc} - x^{i}_{\text{pos}} \|^{2}_{2} = \| x^{i}_{anc} - x^{i}_{neg} \|^{2}_{2}$，所以此时模型并没有学到同类数据的共性特征。可以理解为，在特征表示空间里同类数据特征向量之间的距离与不同类数据特征向量之间的距离没有差异性，就好像用 one hot 来表示词向量时，不同词之间的距离总是相等的，这样不能很好地用来表示词之间的某种语法和语义关系。

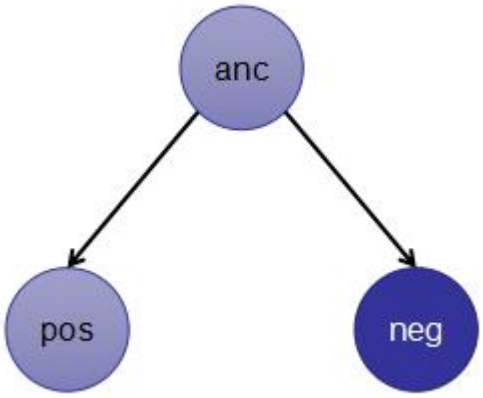

Figure 5: x_{anc} 与 x_{pos} 之间的距离等于 x_{anc} 与 x_{neg} 之间的距离

在加了 TRR_loss 约束后，可以看到模型在验证集上的准确率更加逼近在训练集上的准确率（参考 figure2、3、4 右边图），因此模型的泛化能力要比不加 TRR_loss 约束的模型的泛化能力好。

Model\num samples	10000	50000	100000
No TRR	0.888	0.965	0.979
TRR	0.886(-0.2%)	0.973(+0.8%)	0.981(+0.2%)
TRR_new	**0.905(+1.7%)**	**0.987(+2.2%)**	**0.990(+1.1%)**

Table 1: 不同模型用不同训练集训练在验证集上的最好结果

如 Table 1 所示，其中 No TRR 表示不加 TRR_loss 约束，TRR 表示在 $TRR\_loss = \sum_i sigmoid(\| x_{anc}^i - x_{\text{pos}}^i \|_2^2 - \| x_{anc}^i - x_{neg}^i \|_2^2)$ 约束下的模型在验证集上的分类准确率，TRR_new 表示在 $TRR\_\text{loss} = \sum_i sigmoid(\| x_{anc}^i - x_{\text{pos}}^i \|_2^2) - sigmoid(\| x_{anc}^i - x_{neg}^i \|_2^2)$ 约束下的模型在验证集上的分类准确率。可以看到在 10000,50000,100000 训练集下 TRR_new 都取得了最好的结果，在 50000 个 samples 下模型效果比不加 TRR_loss 约束的模型的分类准确率高了 2.2%。

Future Work

由凸优化相关引理[2]：

给定函数 h: R→R 和函数 g: R^n→R，定义复合函数 f(x)=h o g=h(g(x)): R^n→R，则如下结论成立：

(1) 如果 h 是凸函数且 $\tilde{h}$ 非减，g 是凸函数，则 f 是凸函数

(2) 如果 h 是凸函数且 $\tilde{h}$ 非增，g 是凹函数，则 f 是凸函数

(3) 如果 h 是凹函数且 $\tilde{h}$ 非减，g 是凹函数，则 f 是凹函数

(4) 如果 h 是凹函数且 $\tilde{h}$ 非增，g 是凸函数，则 f 是凹函数

$\tilde{h}$ 表示函数 h 的扩展值延伸，若点不在函数 h 的定义域内，赋值为∞(若 h 是凸函数)或者-∞(若 h 是凹函数)。

对应到神经网络的隐藏层输出:

$$H^i = (\sigma(H^{i-1}W_1^i),...,\sigma(H^{i-1}W_k^i))$$

$$where\ \sigma(x): R \to R,\ H^i \in R^{1\times \dim(i)},\ W_j^i \in R^{\dim(i)\times 1}$$

我们可以得到若函数 $\sigma(x)$ 本身为凸函数且 $\tilde{\sigma}(x)$ 非减，则 H^i 向量里的每个元素的函数表达式关于 H^{i-1} 都是凸函数。因此我们只要找到一个函数 f，使得函数 f 是凸函数并且 $\tilde{f}$ 非减，此时 -f 凹函数且 $-\tilde{f}$ 非增，所以根据结论 (1) 和结论 (4) 可以推出 TRR_loss: $TRR\_\text{loss} = f(\| x_{anc}^i - x_{\text{pos}}^i \|_2^2) - \text{f}(\| x_{anc}^i - x_{neg}^i \|_2^2)$ 是一个凸或凹函数。或者找到一个函数 f，使得函数 f 是凹函数并且 $\tilde{f}$ 非增，此时-f 是凸函数且 $-\tilde{f}$ 非减，由结论(4)和结论(1)也可以推出 TRR_loss 是一个凹或凸函数。

综上，如果可以找到这个函数 f，就可以一层层优化底层的特征表示层，并且根据凸优化原理可以求解得到最优数值解。求解得到底层的特征表示层后再训练一个分类器。所以后面的工作主要是去找到这个函数 f。

References:

[1] https://www.egr.msu.edu/~mizhang/papers/2018_MobiCom_NestDNN.pdf

[2] Stephen Boyd, Lieven Vandenberghe. 凸优化[M]. 清华大学出版社, 2013.