在线持续学习的神经网络结构

蔡承泽*
2018011298
计81
caicz18@mails.tsinghua.edu.cn

王星皓
2018050024
计81
xinghao-18@mails.tsinghua.edu.cn

1 简介

1.1 问题背景

在很多机器学习场景下，我们可以持续不断地获得数据，但不能保证这些数据是独立同分布的，我们还希望能够使用一个比总数据量占用空间小得多的模型处理这些数据，而不需要保存下所有数据。相比于普通的离线的有监督学习，这类场景和人类的学习方式也更为接近。例如在强化学习任务中，模型需要与黑盒的环境进行交互，观测环境状态，并获得环境给出的奖励信号，优化目标是获得奖励的速率。现有的深度强化学习算法通常将问题转为监督学习处理，学习可观测状态到价值或策略的映射，但这种方式产生的数据的分布通常随着策略的改变而改变，相邻的数据也具有高度的相关性。

随机梯度下降是当前训练神经网络模型的主流方式，这种方式将训练样本划分为若干batch，按顺序在每个batch上计算一次梯度并更新模型。但朴素的随机梯度下降基于训练样本独立同分布的假设，在遇到分布发生渐变或突变的训练样本序列时，常常出现灾难性遗忘[1]现象，使得模型在输入序列中靠后位置的分布表现较好，而在输入序列中较前位置的分布表现很差。

因此，我们希望能设计一类神经网络模型，可以适应训练样本序列的分布的改变。具体地，我们考虑神经网络需要按顺序学习多个任务的场景，同个任务内的训练数据在随机打乱后被分为固定大小的多个batch，按顺序进行训练，训练完一批后不再保存这一批数据。另外，我们不希望模型能直接获得任务间的分界线（即模型的训练以batch为单位，训练和测试时都不需要额外标注当前处于哪个任务），而希望模型能自动地发现分布的改变，并进行相应的处理。

1.2 相关工作

[1]实验讨论了基于梯度的神经网络在几个场景下的灾难性遗忘现象，并发现dropout有助于减少遗忘。

持续学习是指模型按顺序学习多个任务，而不需要存储之前任务的数据的场景。已有很多研究讨论这样的场景，大部分研究通过设计合适的loss函数来提高神经网络在持续学习任务中的表现。

[2][3]考虑了按顺序连续学习多个任务，可以存储当前任务的数据，且模型知道当前所在任务的场景；这两个方法都希望在学习新任务时尽量减少对旧任务重要的权重的改变，通过给每个旧任务和每个权重在loss上增加一个二次函数实现，并各自给出了一种方法用于估计二次函数的系数。其中[3]建议使用一个权重在学习过程中对一个任务的loss下降的贡献（的线性近似）来估计这个权重的重要程度。这类方法在训练时需要知道当前所在任务，测试时则不需要，如果用于我们考虑的场景，相当于将每个batch视为一个任务处理。

[4]等工作考虑了类别增量学习的场景，将一个n分类的任务分成m个n/m类的子任务按顺序学习，但测试时需要能对已学习的所有类进行分类。

*组长

[5]提出通过知识蒸馏减少对旧任务的遗忘，需要对每个任务重新初始化和训练模型的最后一层，将模型在新任务上学习前，旧任务对应的最后一层的输出加入到loss中，从而减少学习新任务时，对旧任务的影响。因此这个方法在训练和测试时都需要知道当前处理的任务，一般不适用于我们考虑的场景，但在稍作修改后，类别增量学习符合这个方法的要求。

在另一些相关工作中，允许存储一定数量的旧任务的数据，并将这些数据和新任务的数据一起训练。

以上描述的工作都以任务为单位进行学习，可以存储当前任务的全部数据，可以在一个任务上执行多轮的随机梯度下降。同时，以上方法主要通过对loss的改动减少遗忘，而没有设计专用的神经网络结构。

2 方法

2.1 在线的基于梯度的机器学习模型

一个**在线的基于梯度的机器学习模型**是一个算法，在线地接受输入序列，并给出相应的输出序列，具体地：

- 算法的输入构成集合X，输出构成集合Y；
- 算法的状态$s \in S$，S为所有可能的状态；
- 损失函数$L : X \times Y \to \mathbb{R}$是一个预先定义的可微函数，用于评价算法性能；
- 算法有一个初始状态$s_0 \in S$，输出函数$f_y : X \times S \to Y$，学习函数$f_s : X \times Y \times \times S \times \mathbb{R} \to S$；
- 算法接受一个输入序列$(x_1, x_2, \ldots, x_n) \in X^n$作为输入，输出一个序列$(y_1, \ldots, y_n) \in Y^n$；输入序列的每个下标$i$对应了一个权重$\alpha_i \geq 0$，$\alpha_i = 0$表示训练步骤，$\alpha_i > 0$表示测试步骤；算法的输出由$y_i = f_y(x_i, s_{i-1})$决定，对于训练步骤$i$状态改变由$s_i = f_s(x_i, y_i, s_{i-1}, \nabla_{y_i} L(x_i, y_i))$决定，对于测试步骤$i$，状态保持不变，即$s_i = s_{i-1}$；
- 在一个特定的**任务**中，输入序列中每个元素x_i从X上的分布D_i采样得到，算法需要使得$\mathbb{E}_{x_i \sim D_i,\, i=1,\ldots,n} \left[\alpha_i \sum_{i=1}^{n} \mathbb{E}_{u_i \sim D_i} [L(u_i, f_y(u_i, s_i))] \right]$尽可能小；

对于一个在线的基于梯度的机器学习模型，它的**归纳偏置**表示它接受每个可能的输入序列后的输入输出关系，归纳偏置决定了它在接受了特定输入序列后的状态s，以及学习到的函数$g(x) = f_y(x, s)$，进而决定了它在特定任务上的性能。不同的归纳偏置导致了不同模型在特定任务和特定训练顺序上的表现不同。

例如，k近邻算法对D_i是否相同无特殊假设，但在$\alpha_i > 0$时要求D_i由满足$0 < j < i,\ \alpha_j = 0$的D_j等权重叠加得到，即每次测试时要求输入在之前每次训练的输入分布中等概率抽取。

另一个例子是使用SGD训练的神经网络，神经网络适用于$D_1 = D_2 = \ldots = D_n$即训练和测试时的输入独立同分布的情况；而在输入不同分布时，常常发生灾难性遗忘，使得模型在输入序列中靠后位置的分布表现较好，而在输入序列中较前位置的分布表现很差。

在无任务标签的多任务学习问题中，第i个输入服从的分布D_i具有k种不同的取值$D^{(1)}, D^{(2)}, \ldots, D^{(k)}$，表示$k$个不同任务。算法不知道$k$的具体值，以及每个训练或测试输入属于哪个任务。

我们希望实现的目标是，通过改变神经网络的结构（即s_0, f_y, f_s（可能影响优化器的更新规则、激活函数、层间连接结构、初始化等），而非直接修改损失函数L），使其获得能减少遗忘的归纳偏置，从而提高神经网络在一类无任务标号的多任务学习问题中的性能。

2.2 设计原则

为了减少遗忘，一个合理的想法是将神经网络划分为若干相同结构但不共享参数的组件，使用不同的组件表示不同任务间不相似的部分，每个组件能够识别它的输入是否属于它需

要处理的输入范围，如果输入不属于需要处理的范围，则输出零，并将从输出传回的梯度流置零。另外，还需要一个合理的机制用于学习每个组件能够处理的输入范围。

2.3 判别式训练的模型

在判别式训练中，模型仅优化当前任务的优化目标，而不需要对输入的分布进行显式的建模。此时每个组件只能为了优化当前任务的表现而调整自身的输入范围和输入输出关系。

为了检验判别式训练的模型的效果，我们主要选取了两个实验设置:

1. 对一个数据集$(x_i, y_i)_{i=1}^n$，选取k个可逆函数$f_1, \ldots, f_k$，得到k个数据集$(f_k(x_i), y_i)_{i=1}^n$，作为k个任务按顺序训练;
2. 对一个k分类的任务，将样本按标签分为k类，每类样本构成一个子任务，需要按顺序学习这k个类；这是类别增量学习的一个极端情况，每次只能学习一类;

在第一个实验设置中，不同任务间在结构上非常相似，而在输入分布上显著不同；此时判别式训练得到到的特征通常也适合用于区分不同任务;

在第二个实验设置中，在单个任务上可以平凡地取得零错误率，区分不同任务变为了最主要的难点；此时判别式训练会较快地收敛到最后一个任务零错误率的情况，而很难学习到有效的特征用于区分不同任务;

2.3.1 RBF层

使用d_1维d_2点RBF激活函数的全连接层（简称RBF层）定义如下:

$$y_{j\cdot d_2+l} = f\left(\sum_{i=1}^{n_1}\sum_{k=1}^{d_1}(x_i \cdot W_{k,i,j} - b_{k,l})^2\right)$$

其中$f(x) = \frac{1}{1+x}$, $0 \le j < n_2/d_2$, $1 \le l \le d_2$;

x为输入的n_1维向量，y为输出的n_2维向量，W, b为参数;

这个映射相当于将n_1维的输入经过W线性映射到n_2/d_2个d_1维子空间中，每个子空间中和d_2个可学习的点求距离平方，并经过f函数映射得到n_2/d_2组输出，每组为一个d_2维向量。

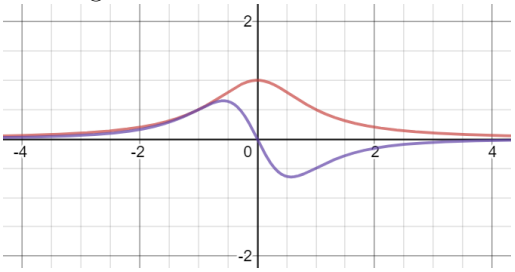
每个样本的运算量为$\Theta(n_1 \cdot n_2 \cdot d_1/d_2 + n_2)$。

一个合适的取值是$(d_1, d_2) = (8, 1)$，此时可以视为每层包含了n_2个组件，每个组件将n_1维的输入经过一个可学习的线性映射映射到d_1维的空间中，再求出和一个可学习的点的欧氏距离x，输出$1/(1+x^2)$。可学习的点附近的区域即为这个组件能够处理的输入范围，而远离可学习的点的区域则有接近0的梯度。过小的d_1会使得组件只能在低维空间中判断输入范围，而过大的d_1则显著增加了参数量，且高维空间中欧氏距离不适合常见场景。通过增大d_2可以减少参数量，但这将导致每d_2个组件共享同一个可学习的线性映射，可能导致任务间的相互干扰。

另外，ReLU激活函数具有类似的特性，但输入范围只能是一个超平面的一侧，因此可能更难区分不同任务对应的不同的输入分布。

$(d_1, d_2) = (1, 1)$的RBF激活函数及其导数的图像如图1所示。

Figure 1: RBF激活函数及其导数



2.3.2 模型结构

我们选取3层或4层的全连接神经网络（输入和每层输出规模为784-512-10或784-512-512-10）用于实验，其中输出规模为512的层为RBF层（分别测试了(d_1, d_2)的不同取值，并与ReLU对比），最后一层为使用softmax的全连接层。4层的模型可能可以利用第二个隐层更好地表示任务间的公共部分，而3层的模型则在训练时通常对旧任务的输出影响更小。

对这类神经网络，选取Adam优化器的一个极限情况：$\beta_1 = 0$, $\beta_2 = 1$，此时优化器不需要估计梯度的一阶矩，使用历史平均值而非指数移动平均值作为梯度的二阶矩估计，在一个batch上进行若干步的自适应学习率的梯度下降后处理下一个batch。

2.4 生成式训练的模型

在生成式训练中，模型需要对输入的分布进行建模，这个过程得到的信息正是组件判断输入范围所需的信息。

自编码器对输入样本x，学习函数$z = f(x; \theta_z)$和$\hat{x} = g(z; \theta_x)$，通过最小化$||x - g(f(x))||_2^2$，可以学习到x到隐变量z的映射f，以及隐变量z到x的估计值的映射g。

2.4.1 自编码器组件

自编码器可以用于无监督异常检测，通过最小化训练样本的重构误差，使得正常样本（和训练集同分布的样本）的重构误差一般小于异常样本（从另一分布中选取的样本）。

利用训练集的标签，对每一类样本分别训练一个自编码器，取重构误差最小的作为测试时的分类结果，即得到一个最简单的类别增量神经网络。由于每类的样本被分到不同的自编码器上，不同类之间的训练互不影响，从而避免了不必要的遗忘。

这里每个自编码器是一个组件，所有类的自编码器共同组成了整个神经网络，重构误差小的区域对应了一个组件能处理的输入范围。

由于自编码器组件学习输入范围时依赖于任务标签，只适用于第二个实验设置，即逐类别的增量学习。

2.4.2 模型结构

使用全连接神经网络，输入和每层输出规模为784-1024-2048-1024-784，最后一层使用sigmoid激活函数，其它层都使用带有0.3概率dropout的ReLU激活函数或都使用RBF-(2,1)。

对这类神经网络，选取RMSProp优化器，在一个batch上进行若干步的自适应学习率的梯度下降后处理下一个batch。

3 实验

3.1 实验设置

MNIST是一个手写数字分类数据集，输入为28*28的灰度图，即784维的向量，每维由$[0, 1)$中的实数表示；数据被分为10类，分别对应数字0-9；训练集规模60000，测试集规模10000，每个类别的样本数基本均匀。

3.1.1 permuted-MNIST

permuted-MNIST是Data Permutation Experiment[6]的一种，是持续学习领域在衡量灾难性遗忘问题上认可度比较高的实验[1][2][3]。这部分实验将MNIST的样本中的784维输入按10个随机选取的排列进行重排列，由此得到10个不同的10分类子任务，其中每个子任务的重排列序列相同。要求神经网络按顺序学习10个子任务，评价指标是学习了前n个任务后，在前n个任务上的平均错误率。每个batch包含200个样本，超参数的选取仅根据$n = 10$的情况调整。

3.1.2 类别增量学习

这部分实验要求神经网络按0-9的顺序学习MNIST的10个类别，评价指标为学习了前n类后，在前n类上的平均错误率。每个batch包含200个样本，超参数的选取仅根据$n = 10$的情况调整。

一些相关工作[4]也考虑了类别增量学习的场景，但一般仅考虑每个任务增量多个类的情况。这里为了满足单个任务平凡的性质，选取了每个任务仅包含一类的极端情况。

3.2 实验结果

3.2.1 permuted-MNIST

表1为输入和隐层规模为784-512-10(3l)或784-512-512-10(4l)时，不同激活函数的神经网络在permuted-MNIST任务上的表现，其中RBF-(d_1,d_2)表示两个隐层都使用d_1, d_2为参数的RBF层的神经网络。

n	1	2	5	10
ReLU-3l	3.69%	4.08%	7.65%	13.54%
ReLU-3l dropout	3.52%	3.51%	4.78%	9.39%
ReLU-4l	3.30%	4.28%	9.34%	17.13%
ReLU-4l dropout	2.83%	3.11%	4.56%	11.48%
RBF-(1,1)-3l	4.56%	4.02%	5.43%	8.95%
RBF-(8,8)-3l	5.10%	4.38%	5.58%	8.44%
RBF-(8,1)-3l	2.58%	**2.31%**	**2.66%**	**3.40%**
RBF-(1,1)-4l	3.82%	4.10%	5.37%	13.69%
RBF-(8,8)-4l	3.06%	2.88%	3.86%	7.44%
RBF-(8,1)-4l	**2.29%**	**2.31%**	2.91%	4.60%

Table 1: permuted-MNIST的实验结果

效果最好的是神经网络结构是RBF-(8,1)-3l，且3层一般比4层更好；ReLU+dropout比ReLU更好，但明显比RBF-(8,1)差。很多神经网络在$n = 2$时表现比$n = 1$更好，说明神经网络可以利用新旧任务间的相似性提升平均表现。

图2,3给出了RBF-(8,1)-3l和ReLU-3l在连续训练完10个任务后，部分隐藏层输出的可视化结果，其中每个方格对应一个隐层输出，方格中每行对应0-9的一个类别，每列对应一个任务，亮度代表输出均值；RBF-(8,1)-3l的隐藏层输出均值在部分任务和部分类别上较高，这表明RBF层倾向于仅对部分任务和部分类别给出较高的响应，且在学习新任务时倾向于将新任务和旧任务的同个类别子集用同个组件表示，因此图中每个方块更接近一个秩1矩阵；ReLU-3l的隐藏层输出则表现出更多的任务间干扰，很多神经元在新旧任务上响应不同的类别子集。

3.2.2 类别增量学习

表2为使用ReLU或RBF激活函数的自编码器(AE)以及全连接神经网络的实验结果，AE-offline表示自编码器在离线训练（即不限制必须按顺序处理每个batch，且可以存储所有数据）时的结果。

n	1	2	5	10
ReLU-3l	5.92%	12.62%	60.19%	74.85%
RBF-(8,1)-3l	2.86%	3.83%	31.43%	39.35%
AE-ReLU	0.00%	**0.05%**	**0.91%**	**3.26%**
AE-RBF	0.00%	**0.05%**	1.03%	3.27%
AE-ReLU-offline	0.00%	0.00%	0.58%	2.79%

Table 2: MNIST上逐类别增量学习的实验结果

Figure 2: RBF-(8,1)-3l

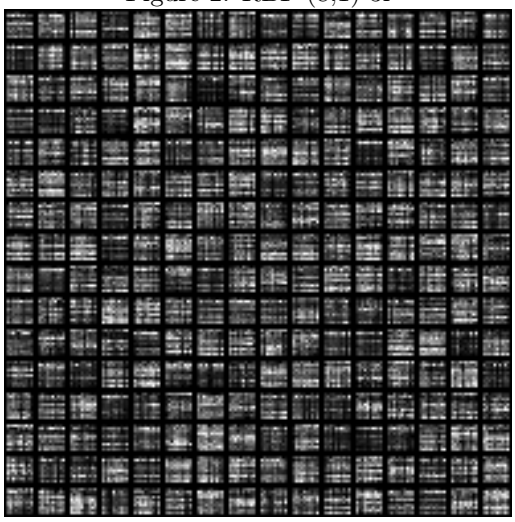
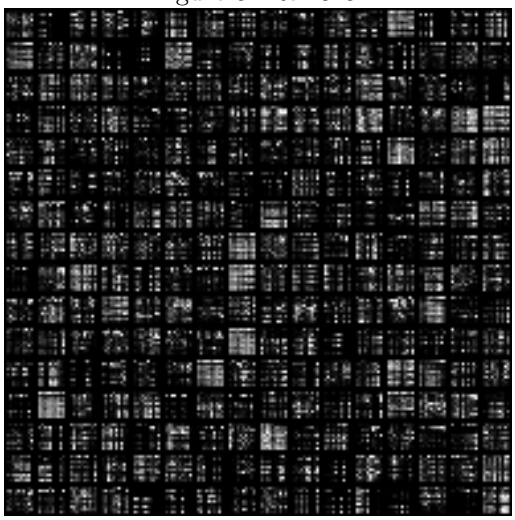


Figure 3: ReLU-3l



使用自编码器在线训练可以得到较低的错误率，而判别式训练的模型则不能很好地处理这个场景。在线训练的自编码器在按顺序处理每个batch时，仅比离线情况稍差。这个场景下每个自编码器实际上在独立同分布的数据上训练，因此RBF和ReLU没有太大的区别。图4,5显示了自编码器在一组输入上的输出，第一行为真实样本，之后每行表示每个自编码器重构得到的输出，可以看出主对角线上重构误差较小。

Figure 4: AE-ReLU

Figure 5: AE-RBF

4 总结

我们提出的使用RBF激活函数的神经网络，在permuted-MNIST上表现良好，说明仅改变神经网络的结构，就可以显著改善神经网络在一些任务序列上的遗忘情况，因此基于结构设计的方法在持续学习任务上是有竞争力的。我们提出的使用自编码器的类别增量神经网络，可以按顺序学习每类，而朴素的梯度下降和之前的很多类别增量学习方法在此情况下退化。通过修改分配样本到自编码器的方式，这个方法有希望推广到无监督和半监督学习的情况。另外，逐类别增量学习任务作为一个特例，也说明了显式建模输入的分布对于自动区分不同任务是重要的。

我们提出的两类方法均适用于完全在线的学习，不需要缓存当前任务的数据或对每个任务进行预处理、后处理就能连续学习多个任务，也不需要任务标签信息，因此比以任务为单位的持续学习更为通用，经过改进后可能可以用于处理强化学习中的非平稳数据。我们提出的设计原则，给出了关于神经网络的灾难性遗忘及其解决方案的一个新视角。

参考文献

References

[1] Goodfellow, I. J., M. Mirza, D. Xiao, et al. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[2] Kirkpatrick, J., R. Pascanu, N. Rabinowitz, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[3] Zenke, F., B. Poole, S. Ganguli. Continual learning through synaptic intelligence. *Proceedings of machine learning research*, 70:3987, 2017.

[4] Zhao, B., X. Xiao, G. Gan, et al. Maintaining discrimination and fairness in class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13208–13217. 2020.

[5] Li, Z., D. Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[6] Kemker, R., M. McClure, A. Abitino, et al. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32. 2018.