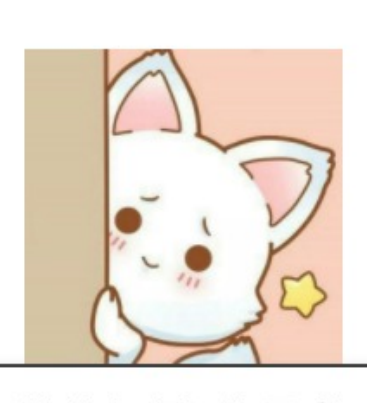
如何提高NLP模型鲁棒性和泛化能力？对抗训练论文串讲

李如 夕小瑶的卖萌屋 2020-03-11 22:54:11 手机阅读



一只小狐狸带你解锁 **炼丹术&NLP** 秘籍

作者：李如

> 本文主要串烧了FGSM, FGM, PGD, FreeAT, YOPO, FreeLB, SMART这几种对抗训练方法，希望能使各位大佬炼出的丹药更加圆润有光泽，一颗永流传。【注：本文谢绝转载】

简介

对抗训练是一种引入噪声的训练方式，可以对参数进行正则化，提升模型鲁棒性和泛化能力。

对抗训练的假设是：给输入加上扰动之后，输出分布和原Y的分布一致

有监督的数据下使用交叉熵作为损失：

$$-logp(y|x+r_{adv};\theta)$$

半监督数据下可计算KL散度：

$$KL[p(\cdot|x;\hat{\theta})||p(\cdot|x+r_{v-adv};\theta)]$$

扰动如何得来呢？这需要对抗的思想，即**往增大损失的方向增加扰动**

有监督下：

$$r_{adv}=\underset{r,||r||\le\epsilon}{argmin}\,logp(y|x+r;\hat{\theta})$$

半监督下：

$$r_{v-adv}=\underset{r,||r||\le\epsilon}{argmax}\,KL[p(\cdot|x;\hat{\theta})||p(\cdot|x+r;\theta)]$$

theta上面一个尖儿代表的是常数。目的是说在计算对抗扰动时虽然计算了梯度，但不对参数进行更新，**因为当前得到的对抗扰动是对旧参数最优的**。不理解的同学可以自己看下伪代码体会一下。

用一句话形容对抗训练的思路，就是**在输入上进行梯度上升(增大loss)，在参数上进行梯度下降(减小loss)**。由于输入会进行embedding lookup，所以**实际的做法是在embedding table上进行梯度上升。**

接下来介绍不同的方法，后续方法**优化的主要方向有两点：得到更优的扰动 & 提升训练速度。**

后台回复【对抗训练】获取论文PDF打包下载链接~~

论文串讲

FGSM (Fast Gradient Sign Method): ICLR2015

FGSM是Goodfellow提出对抗训练时的方法，假设对于输入的梯度为：

$$g=\nabla_x L(\theta,x,y)$$

那扰动肯定是沿着梯度的方向往损失函数的极大值走：

$$r_{adv}=\epsilon\cdot sign(g)$$

FGM (Fast Gradient Method): ICLR2017

FSGM是每个方向上都走相同的一步，Goodfellow后续提出的FGM则是根据具体的梯度进行scale，得到更好的对抗样本：

$$r_{adv}=\epsilon g/||g||_2$$

伪代码：

```
对于每个x:
  1.计算x的前向loss、反向传播得到梯度
  2.根据embedding矩阵的梯度计算出r，并加到当前embedding上，相当于x+r
  3.计算x+r的前向loss，反向传播得到对抗的梯度，累加到(1)的梯度上
  4.将embedding恢复为(1)时的值
  5.根据(3)的梯度对参数进行更新
```

PGD (Projected Gradient Descent): ICLR2018

FGM直接通过epsilon参数一下子算出了对抗扰动，这样得到的可能不是最优的。因此PGD进行了改进，多迭代几次，慢慢找到最优的扰动。

引用[1]：

> FGM简单粗暴的“一步到位”，可能走不到约束内的最优点。PGD则是“小步走，多走几步”，如果走出了扰动半径为epsilon的空间，就映射回“球面”上，以保证扰动不要过大

$$r_{adv|t+1}=\alpha g_t/||g_t||_2$$

且

$$||r||_2\le\epsilon$$

伪代码：

```
对于每个x:
  1.计算x的前向loss、反向传播得到梯度并备份
  对于每步t:
    2.根据embedding矩阵的梯度计算出r，并加到当前embedding上，相当于x+r(超出范围则投
    3.t不是最后一步: 将梯度归0，根据1的x+r计算前后向并得到梯度
    4.t是最后一步: 恢复(1)的梯度，计算最后的x+r并将梯度累加到(1)上
  5.将embedding恢复为(1)时的值
  6.根据(4)的梯度对参数进行更新
```

可以看到，在循环中r是逐渐累加的，要注意的是**最后更新参数只使用最后一个x+r算出来的梯度**。

FreeAT (Free Adversarial Training): NIPS2019

从FGSM到PGD，主要是优化对抗扰动的计算，虽然取得了更好的效果，但计算量也一步步增加。对于每个样本，FGSM和FGM都只用计算两次，一次是计算x的前后向，一次是计算x+r的前后向。而PGD则计算了K+1次，消耗了更多的计算资源。因此FreeAT被提了出来，在PGD的基础上进行训练速度的优化。

FreeAT的思想是在对每个样本x连续重复m次训练，计算r时复用上一步的梯度，为了保证速度，整体epoch会除以m。r的更新公式为：

$$r_{t+1}=r_t+\epsilon\cdot sign(g)$$

伪代码：

```
初始化r=0
对于epoch=1...N/m:
  对于每个x:
    对于每步m:
      1.利用上一步的r，计算x+r的前后向，得到梯度
      2.根据梯度更新参数
      3.根据梯度更新r
```

缺点：FreeLB指出，FreeAT的问题在于每次的r对于当前的参数都是次优的（无法最大化loss），因为当前r是由r(t-1)和theta(t-1)计算出来的，是对于theta(t-1)的最优。

注：

1.论文中提供伪代码，但源码中好像对1步输入做了归一化论文中并没有提到

2.个人认为可以把FreeAT当成执行m次的FGSM，最开始r=0，第一次更新的是x的梯度，之后开始迭代更新r，则根据x+r的梯度更新参数。但代码中有个问题是r只在最开始初始化，如果迭代到新的样本x2，也是根据上个样本的r进行更新的。这里存在一些疑问，欢迎大家评论区一起讨论。

代码：

https://github.com/mahyarnajibi/FreeAdversarialTraining/blob/d70774030871fa3207e09ce8528c1b84cd690603/main_free.py#L160

YOPO (You Only Propagate Once): NIPS2019

YOPO的目标也是提升PGD的效率，这篇文章需要的理论知识比较雄厚，这里只简要介绍一下。

感兴趣又啃不下来原论文的同学（比如我）可以参考[9]，如有解读错误欢迎指出～

极大值原理PMP(Pontryagin's maximum principle)是optimizer的一种，它将神经网络看作动力学系统。这个方法的优点是在优化网络参数时，层之间是解藕的。通过这个思想，我们可以想到，既然扰动是加在embedding层的，为什么每次还要计算完整的前后向传播呢？

基于这个想法，作者想复用后几层的梯度，假设p为定值：

$$p=\nabla_{g_{\tilde{\theta}}}(l(g_{\tilde{\theta}}(f_0(x_i+r_i^{j,0},\theta_0)),y_i))\cdot\nabla_{f_0}(g_{\tilde{\theta}}(f_0(x_i+r_i^{j,0},\theta_0)))$$

则对r的更新就可以变为

$$r_i^{j,s+1}=r_i^{j,s}+\alpha_1 p\cdot\nabla_{r_i}f_0(x_i+r_i^{j,s},\theta_0)$$

我们可以先写出YOPO的梯度下降版本：

```
对于每个样本x
初始化r(1,0)
对于j=1,2,...,m:
  1.根据r(j,0),计算p
  对于s=0,1,...,n-1:
    2.计算r(j,s+1)
  3.另r(j+1,0)=r(j,n)
```

作者又提出了PMP版本的YOPO，并证明SGD的YOPO是PMP版的一种特殊形式。这样每次迭代r就只用到embedding的梯度就可以了。

引用[9]：

> **虽然YOPO-m-n只完成了m次完整的正反向传播，但是却实现了m*n次梯度下降**。而PGD-r算法完成r次完整的正反向传播却只能实现r次梯度下降。这样看来，YOPO-m-n算法的效率明显更高，而实验也表明，**只要使得m*n略大于r，YOPO-m-n的效果就能够与PGD-r相媲美**。

然而故事的反转来的太快，FreeLB指出YOPO使用的假设对于ReLU-based网络不成立：

> Interestingly, the analysis backing the extra update steps assumes a twice continuously differentiable loss, which does not hold for ReLU-based neural networks they experimented with, and thus the reasons for the success of such an algorithm remains obscure.

代码：

https://github.com/a1600012888/YOPO-You-Only-Propagate-Once

别问了，问就是PMP，来跟我一起进入下一部份的学习。

FreeLB (Free Large-Batch): ICLR2020

FreeLB认为，FreeAT和YOPO对于获得最优r (inner max)的计算都存在问题，因此提出了一种类似PGD的方法。只不过PGD只使用了最后一步x+r输出的梯度，而FreeLB取了每次迭代r输出梯度的平均值，相当于把输入看作一个K倍大的虚拟batch，由[X+r1, X+r2, ..., X+rk]拼接而成。具体的公式为：

$$\underset{\theta}{min}\,\mathbb{E}_{(Z,y)\sim\mathcal{D}}[\frac{1}{K}\sum_{t=0}^{K-1}\underset{r_t\in\mathcal{I}_t}{max}\,L(f_\theta(X+r_t),y)]$$

为了方便对比，再贴下论文中PGD的公式：

$$\underset{\theta}{min}\,\mathbb{E}_{(Z,y)\sim\mathcal{D}}[\underset{||r||\le\epsilon}{max}\,L(f_\theta(X+r),y)]$$

FreeLB和PGD主要有两点区别：

1.PGD是迭代K次r后取最后一次扰动的梯度更新参数，FreeLB是取K次迭代中的平均梯度

2.PGD的扰动范围都在epsilon内，因为伪代码第3步将梯度归0了，每次投影都会回到以第1步x为圆心，半径是epsilon的圆内，而FreeLB每次的x都会迭代，所以r的范围更加灵活，更可能接近局部最优：

$$\mathcal{I}_t=\mathcal{B}_{X+r_0}(\alpha t)\cap\mathcal{B}_X(\epsilon)$$

FreeLB的伪代码为：

```
对于每个x:
  1.通过均匀分布初始化r，梯度g为0
  对于每步t=1...K:
    2.根据x+r计算前后向，累计梯度g
    3.更新r
  4.根据g/K更新梯度
```

论文中还指出了很重要的一点，就是**对抗训练和dropout不能同时使用**，加上dropout相当于改变了网络结构，会影响r的计算。如果要用的话需要在**K步中都使用同一个mask**。

SMART (SMoothness-inducing Adversarial Regularization)

SMART论文中提出了两个方法：

1.对抗正则 SMoothness-inducing Adversarial Regularization，提升模型鲁棒性

2.优化算法 Bregman proximal point optimization，避免灾难性遗忘

本文只介绍其中的对抗正则方法。

SMART提出了两种对抗正则损失，加到损失函数中：

$$\underset{\theta}{min}\,\mathcal{F}(\theta)=\frac{1}{n}\sum_{i=1}^{n}l(f(x_i;\theta),y_i)+\lambda_s\mathcal{R}_s(\theta)$$

第一种参考了半监督对抗训练，对抗的目标是最大化扰动前后的输出，在分类任务时loss采用对称的KL散度，回归任务时使用平方损失损失：

$$[A]:\mathcal{R}_s(\theta)=\frac{1}{n}\sum_{i=1}^{n}\underset{||\tilde{x_i}-x_i||_p\le\epsilon}{max}\,l_s(f(x_i;\theta),f(\tilde{x_i};\theta))$$

第二种方法来自DeepMind的NIPS2019[8]，核心思想是让模型学习到的流行更光滑，即让loss在训练数据呈线性变化，增强对扰动的抵抗能力。作者认为，如果loss流行足够平滑，那l(x+r)可以用一阶泰勒展开进行近似，因此用来对抗的扰动需要最大化l(x+r)和一阶泰勒展开的距离：

$$[B]:\mathcal{R}_s(\theta)=\frac{1}{n}\sum_{i=1}^{n}\underset{||\tilde{x_i}-x_i||_p\le\epsilon}{max}\,|l(f(\tilde{x_i};\theta),y_i)-l(f(x_i;\theta),y_i)-(\tilde{x_i}-x_i)^T\nabla_x l(f(\tilde{x_i};\theta$$

SMART的算法和PGD相似，也是迭代K步找到最优r，然后更新梯度。

总结

把最近的一些对抗训练方法总结出来，可以看到趋势从“优化PGD的速度”又回到了“找寻最优扰动”，个人也比较认同，训练速度慢一些对于普通模型还是可以接受的，主要还是看最终的效果有没有提升。之前自己试过FGM和PGD，FGM有轻微提升，但PGD没有，应该需要在超参数上进行调整。FreeLB和SMART在GLUE榜单上都有出现过，相信之后对抗训练也是标配了，坐等微软放出源码。

后台回复【对抗训练】获取论文PDF打包下载链接~~

参考文献

[1]. 知乎：【炼丹技巧】功守道：NLP中的对抗训练 + PyTorch实现

[2]. FGSM: Explaining and Harnessing Adversarial Examples

[3]. FGM: Adversarial Training Methods for Semi-Supervised Text Classification

[4]. FreeAT: Adversarial Training for Free!

[5]. YOPO: You Only Propagate Once: Accelerating Adversarial Training via Maximal Principle

[6]. FreeLB: Enhanced Adversarial Training for Language Understanding

[7]. SMART: Robust and Efficient Fine-Tuning for Pre-trained Natural

[8]. Adversarial Robustness through Local Linearization

[9]. 知乎：加速对抗训练——YOPO算法浅析

文章已于2020-03-14修改