Paper Reading

mstyoda 骆轩源

Contents

1 问题背景：

一个神经网络模型可以用一个$net_w : \mathcal{X} \to \mathcal{Y}$ 的函数来描述，其中$\mathcal{X}$，$\mathcal{Y}$和w分别是输入空间，输出空间以及网络参数。然后定义一个loss function $L : \mathcal{Y}^2 \to \mathbb{R}$。我们的任务即计算出参数$w^*$满足：

$$w^* = \arg\min_w \mathbb{E}_{x\sim D}\left[L(net_w(x), c(x))\right] \tag{1}$$

其中c为需要学习的concept，令$f(w;x) = L(net_w(x), c(x))$，函数$R(w) = \mathbb{E}_{x\sim D}\left[f(w;x)\right]$的计算难度很大，所以我们考虑用$\hat{R}(w) = \frac{1}{n}\sum_{i=1}^{n} f(w;x_i)$ 来近似(x_i独立同分布于D)。那么$|R(w) - \hat{R}(w)|$即为generalization error(用ϵ_{gen}表示)。故我们的学习目标即为计算w^*满足(能写成如下形式是因为一般情况下$R(w) > \hat{R}(w)$)：

$$w^* = \arg\min_w \left(\epsilon_{gen}(w) + \hat{R}(w)\right) \tag{2}$$

现在DL广泛使用方法，都是用SGD类的算法来最小化$\hat{R}(w)$，并没有仔细考虑$\epsilon_{gen}(w)$，只是采用了一些策略，比如：

- 减少迭代次数T。
- 使用小的$batchsize$。
- 找平缓一些的极值点。
- 正则化参数。
- 设置Dropout。

这样一些策略，从实验的结果来看，其泛化误差也很小。那么真的是如上的策略起了作用吗？比如[1]中就提到任何一个平缓的极值点都可以被等价对应到一个“无限尖锐”的极值点。还有[3]提到使用warm-start之后，大的$batchsize$ 一样也不会Overfit等等。

[4]通过让网络fit随机数据证明了现在神经网络有足够的能力记住所有的训练数据，那我们很有可能就学到了一个只是记住了所有样例没有泛化能力的一个模型，但事实并不是这样。

综合来看，DL的泛化能力**极有可能是SGD的性质决定的**。[2]中给出SGD算法的uniform stability ϵ_{stab}的一个bound，其和step size 的大小，迭代次数T以及$f(w;x)$的一些性质有关。

所以一个值得研究的问题就是：目前的深度学习算法学习到的模型有很好的泛化能力是否与SGD有关？如果有，具体的关系如何？

2 深入思考

假设我们对于给定的训练数据S，算法A从H中学习到了任意一个函数h，那么完全可以构造一个能完全fit S的对手concept c，使得h在S以外的样例全错。这样的例子激发我们思考：泛化能力是否与c**的“学习难度”**有关？

那么如果有一个非常简单的c固定只返回一种分类，但是H非常的复杂，它有非常多的函数都能fit S，那么从这些中抽出一个它的generalization error也很大。所以这个例子又激发我们思考：泛化能力是否与**模型的表示能力**H有关？

将c的难易程度和H的表示能力合起来，可以得到如下定义的函数族G，其中L为loss-function：

$$G = \{g(x) = L(h(x), c(x)) : h \in H\} \tag{3}$$

若从传统的VC维观点来看：

$$\begin{aligned} R(h) &\leq \hat{R}(h) + 2\mathfrak{R}_m(G) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}} \\ &= \hat{R}(h) + \mathfrak{R}_m(H) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}} \end{aligned} \tag{4}$$

故**仅和模型的表示能力**H有关。但是该上界给出的其实以H中**泛化最差**的那个为标准，而忽略算法A 的筛选作用，对于DL模型，其表示能力非常强，[4]中就提到了一些神经网络可以完全fit随机数据。所以使用传统的分析方法**并不适用！**

故需要考虑算法的因素，于是开始研究SGD算法的Algorithm stability。也即研究当算法A接受到最多一个元素不同的训练样例S和S'时，是否有如下关系成立：

$$\sup_x \mathbb{E}_A \left[f(A(S); x) - f(A(S'); x) \right] \leq \epsilon \tag{5}$$

对于DL来说，w对应一个$h_w \in H$，那么$f(w; x) = L(h_w(x), c(x))$，也即对应$g \in G$，而$g$函数是**同时**带着concept c和模型表示能力H的，所以在研究SGD算法Algorithm stability的时候，两种因素都有被综合考虑。

由于DL的结构非常复杂，$f(w; x)$的**性质很难归纳**，在[2]中是假定$f(w)$满足L-Lipschitz和β-smooth，给出一个上界。而L的大小和β的大小也很难得到。

由于$f(w; x)$和要学习的concept c，以及模型H有关，所以将问题**分得更细一些**，比如：“手写数字识别搭配CNN 时，使用SGD算法的泛化误差有多大？”这样细化之后，$f(w)$ 的性质说不定能够得到更深一步的挖掘。

3 Train faster, generalize better: Stability of stochastic gradient descent

本文的主要贡献是从Algorithm Stability的角度来分析了SGD算法的generalization bound。假定一个随机算法A，它对于一组训练数据S产生一个输出w，那么对于任意给定两组相差不过一个元素训练数据S和S'，如果有：

$$\sup_x \mathbb{E}_A \left[f(A(S); x) - f(A(S'); x) \right] \leq \epsilon_{stab} \tag{6}$$

则可以得出：

Theorem 1 (Generalization in expectation). 对于一个ϵ_{stab}-*uniformly stable*的随机算法A，有：

$$\epsilon_{gen} = abs(\mathbb{E}_{S,A} \left[R_S[A(S)] - R[A(S)]] \right) \leq \epsilon_{stab} \tag{7}$$

Proof. 令$S = (z_1, z_2 ... z_n)$，$S' = (z'_1, z'_2 ... z'_n)$，$S^{(i)} = (z_1, z_2, ... z'_i ..., z_n)$也即仅第$i$位等于$z'_i$其他都与$S$相同，所以可以把$\mathbb{E}_{S,A} \left[R_S[A(S)]\right]$写成：

$$\begin{aligned}
\mathbb{E}_{S,A} \left[R_S[A(S)]\right] &= \mathbb{E}_{S,A} \left[\frac{1}{n} \sum_{i=1}^{n} f(A(S); z_i) \right] \\
&= \mathbb{E}_A \left[\frac{1}{n} \sum_{i=1}^{n} \sum_{s,z} \mathbb{P}(S = s, z_i = z) f(A(s); z) \right] \\
&= \mathbb{E}_A \left[\frac{1}{n} \sum_{i=1}^{n} \sum_{s,z} \mathbb{P}(z_1 = s_1, z_2 = s_2 ..., z_i = s_i = z, ..., z_n = s_n) f(A(s); z) \right] \\
&= \mathbb{E}_A \left[\frac{1}{n} \sum_{i=1}^{n} \sum_{s,z} \mathbb{P}(z_1 = s_1, z_2 = s_2 ..., z'_i = s_i = z, ..., z_n = s_n) f(A(s); z) \right] \\
&= \mathbb{E}_{S,S',A} \left[\frac{1}{n} \sum_{i=1}^{n} f(A(S^{(i)}); z'_i) \right]
\end{aligned} \tag{8}$$

之后再求$\mathbb{E}_{S,A} \left[R_S[A(S)]\right]$与$\mathbb{E}_{S,A} \left[R[A(S)]\right]$ 的差δ：

$$\begin{aligned}
\delta &= \mathbb{E}_{S,S',A} \left[\frac{1}{n} \sum_{i=1}^{n} f(A(S^{(i)}); z'_i) \right] - \mathbb{E}_{S,S',A} \left[\frac{1}{n} \sum_{i=1}^{n} f(A(S); z'_i) \right] \\
&= \mathbb{E}_{S,S',A} \left[\frac{1}{n} \sum_{i=1}^{n} f(A(S^{(i)}); z'_i) - f(A(S); z'_i) \right]
\end{aligned} \tag{9}$$

由于$S^{(i)}$和S相差不超过一个元素，所以可以应用ϵ_{stab}-uniformly stable的性质，得到：

$$\epsilon_{gen} = |\delta| \leq \epsilon_{stab} \tag{10}$$

□

这意味着只需要分析SGD算法在遇到两个相差很小的训练样例S 和S' 时返回的w和w'的loss的差距的上界，将问题大大简化。Paper中为了方便定义了$f(w) = \frac{1}{|Batch|}\sum_{x\in Batch} f(w;x)$，它的性质其实和$f(w;x)$相同，假如对于所有的$x$ 都有$f(w;x)$是L-Lipschitz，那么$f(w)$也是。

当$f(w)$为凸函数，且为L-Lipschitz和β-smooth，则当$\alpha_t \le 2/\beta$的时候有：

$$\epsilon_{stab} \le \frac{2L^2}{n}\sum_{t=1}^{T}\alpha_t \tag{11}$$

当然$f(w)$一般都是非凸，这时如果$f(w)$依旧满足L-Lipschitz和β-smooth，并且$|f(w)| \le 1$然后固定一个常数c用来约束$\alpha_t \le c/t$，这时候有：

$$\epsilon_{stab} \le \frac{1+1/\beta c}{n-1}(2cL^2)^{\frac{1}{\beta c+1}}T^{\frac{\beta c}{\beta c+1}} \tag{12}$$

对于非凸的情况，简单介绍一下证明的思路：固定z，找$\mathbb{E}_z|f(w_T;z) - f(w'_T;z)|$的上界。由于$f(w)$是L-Lipschitz，令$\delta_t = \|w_t - w'_t\|$，所以有$\|f(w_t) - f(w'_t)\| \le L\delta_t$。也即把$f$的差距变为$w$的差距。

一开始$\delta_0 = 0$，如果能找到一个“不错”的函数Φ_t使得$\delta_{t+1} \le \Phi_t(\delta_t)$，那么就可以从$t = 0$一路递推到$t = T$求出$\delta_T$ 的上界。

从$\delta_{t+1} \le \Phi_t(\delta_t)$可以看出一步放缩其实是“有误差”的，想让得到上界更加“紧”，就需要减少这样递推的长度，所以我们可以找到其他$\delta_{t_0} = 0$ 的点，然后从t_0开始递推到T，长度就被缩短了。

所以作者先证明了如下结论：

Lemma 2. 如果S和S'相差不超过*1*个元素，w_t, w'_t分别为A对于S和S'第t次迭代的结果，且$f(w;z)$是*L-Lipschitz*且非负，则对于任意的$z \in Z$，$t_0 \in \{1,2,...,n\}$，假设算法采用**随机排列**的方式，则有：

$$\mathbb{E}|f(w_T;z) - f(w'_T;z)| \le \frac{t_0}{n}\sup_{w,z} f(w;z) + L\mathbb{E}[\delta_T|\delta_{t_0} = 0] \tag{13}$$

Proof. 对于任意$t_0 \in \{1,2,...,n\}$，有：

$$\begin{aligned}\mathbb{E}|f(w_T;z) - f(w'_T;z)| &= \mathbb{P}[\delta_{t_0} = 0]\mathbb{E}[|f(w_T;z) - f(w'_T;z)|\,|\delta_{t_0} = 0] \\ &\quad + \mathbb{P}[\delta_{t_0} \ne 0]\mathbb{E}[|f(w_T;z) - f(w'_T;z)|\,|\delta_{t_0} \ne 0] \\ &\le L\mathbb{E}[\delta_T|\delta_{t_0} = 0] + \mathbb{P}[\delta_{t_0} \ne 0]\sup_{w,z} f(w,z)\end{aligned} \tag{14}$$

这一步将$\mathbb{P}[\delta_{t_0}=0]$放缩成1，$\mathbb{E}[|f(w_T;z)-f(w'_T;z)|\,|\delta_{t_0}\neq 0]$放缩成$\sup_{w,z} f(w,z)$，这样的放缩**太过松弛！**。

由于算法使用的随机排列(一开始随机一个排列，然后不断循环这个排列)，令随机变量I表示使得$z_i \neq z'_i$ 的下标，若$I > t_0$则肯定有$\delta_{t_0}=0$ 因为前面的过程完全一致，所以$\mathbb{P}[\delta_{t_0}\neq 0]\leq \frac{t_0}{n}$，代入上式即完成证明。 □

之后做的事情就是考虑$\mathbb{E}[\delta_T|\delta_{t_0}=0]$的上界，也就是从$t_0$开始往后递推。令$\Delta_t = \mathbb{E}[\delta_t|\delta_{t_0}=0]$。

对于第$t+1$次迭代，有$1-1/n$概率S和S'选出相同的元素，此时$\Delta_{t+1}\leq(1+\alpha_t\beta)\Delta_t$，还有$1/n$的概率选出不同的元素，这时候$\Delta_{t+1}\leq\Delta_t+2\alpha_t L$，所以有:

$$\Delta_{t+1}\leq(1-1/n)(1+\alpha_t\beta)\Delta_t+\frac{1}{n}\Delta_t+\frac{2\alpha_t L}{n} \tag{15}$$

之后代入条件$\alpha_t\leq c/t$，再使用一些放缩可以得到：

$$\Delta_T\leq\frac{2L}{\beta(n-1)}\left(\frac{T}{t_0}\right)^{\beta c} \tag{16}$$

由于$f(w)\leq 1$，所以综合这些结果得到：

$$\mathbb{E}|f(w_T;z)-f(w'_T;z)|\leq\frac{t_0}{n}+\frac{2L^2}{\beta(n-1)}\left(\frac{T}{t_0}\right)^{\beta c} \tag{17}$$

由于t_0可以任取，所以就要选取一个使得右边的值最小，于是就得到了：

$$t_0^*=(2cL^2)^{\frac{1}{\beta c+1}}T^{\frac{\beta c}{\beta c+1}} \tag{18}$$

$$\mathbb{E}|f(w_T;z)-f(w'_T;z)|\leq\frac{1+1/\beta c}{n-1}(2cL^2)^{\frac{1}{\beta c+1}}T^{\frac{\beta c}{\beta c+1}} \tag{19}$$

关于凸函数的情况证明方法其实类似，总结一下，作者的证明手段主要为：

- 将$|f(w)-f(w')|$通过L-Lipschitz性质bound 在$L|w-w'|$。
- 利用SGD 迭代更新的性质和f的光滑性，得到δ_{t+1}和δ_t之间的关系。
- 适当选择递推的起点t_0。

而我认为作者最终给出的bound太悲观的原因有：

- 用L-Lipschitz变为研究δ_t的大小之后，就少利用了$f(w)$的很多性质。
- 关于非凸部分的证明中，有些地方放缩太大，比如直接用$\sup_{w,z} f(w,z)$来替代$t_0\neq 0$时的期望。

References

[1] Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. Sharp minima can generalize for deep nets. arXiv preprint arXiv:1703.04933 (2017).

[2] Hardt, M., Recht, B., and Singer, Y. Train faster, generalize better: Stability of stochastic gradient descent. arXiv preprint arXiv:1509.01240 (2015).

[3] Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. arXiv preprint arXiv:1609.04836 (2016).

[4] Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. arXiv preprint arXiv:1611.03530 (2016).