利普西斯连续性（Lipschitz Continuity）

概念

利普希茨连续性 (Lipschitz continuity) 的定义很简单，他要求函数 $f(x)$ 满足对任意 x, y 都有：

$$\frac{f(x)-f(y)}{x-y} \leq K \tag{30}$$

延申

Lipschitz continuous gradient：

$$\frac{f'(x)-f'(y)}{x-y} \leq K \tag{31}$$

Lipschitz continuous Hessian

$$\frac{f''(x)-f''(y)}{x-y} \leq K \tag{32}$$

可以一直往上构造，构造出更高阶的 Lipschitz continuous condition。

直观解释

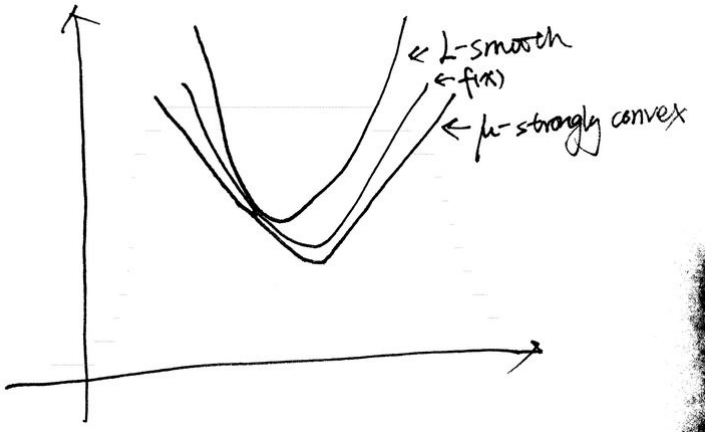
Lipschitz continuous： 函数被一次函数上下夹逼

Lipschitz continuous gradient ：函数被二次函数上下夹逼

Lipschitz continuous Hessian ：函数被三次函数上下夹逼

Lipschitz continuous 用在函数值上是为了不让函数值变化的太快；用在导函数上，是为了不让导函数变化的太快；用在Hessian上，是为了让Hessian不变化的太快。但他们都导致了一个很有意思的结果：这个Lipschitz continuous不管用在什么上，都使的函数被多项式上下夹逼，一方面便于我们处理，另一方面至少我们能控制一下函数的包络信息。

示意图：



解释：

Lipschitz continuous：

$$\begin{cases} f(y) \leq f(x) + K\|y-x\| \\ f(y) \geq f(x) - K\|y-x\| \end{cases} \tag{33}$$

Lipschitz continuous gradient ：

$$\begin{cases} f(y) \leq f(x) + < f'(x), y-x > + \frac{K}{2}\|y-x\|^2 \\ f(y) \geq f(x) + < f'(x), y-x > - \frac{K}{2}\|y-x\|^2 \end{cases} \tag{34}$$

Lipschitz continuous Hessian ：

$$\begin{cases} f(y) \leq f(x) + < f'(x), y-x > + \frac{1}{2} < f''(x)(y-x), y-x > + \frac{K}{6}\|y-x\|^3 \\ f(y) \geq f(x) + < f'(x), y-x > + \frac{1}{2} < f''(x)(y-x), y-x > - \frac{K}{6}\|y-x\|^3 \end{cases} \tag{35}$$

扰动敏感

记输入为x，输出为y，模型为f，模型参数为w，记为

$$y = f_w(x) \tag{36}$$

很多时候，我们希望得到一个"稳健"的模型。何为稳健？一般来说有两种含义，一是对于参数扰动的稳定性，比如模型变成了$f_{w+\Delta w}(x)$后是否还能达到相近的效果？如果在动力学系统中，还要考虑模型最终是否能恢复到$f_w(x)$；二是对于输入扰动的稳定性，比如输入从x变成了$x+\Delta x$后，$f_w(x+\Delta x)$是否能给出相近的预测结果。例如，深度学习模型存在"对抗攻击样本"，图片只改变一个像素就给出完全不一样的分类结果，这就是模型对输入过于敏感的案例。

L约束

所以，大多数时候我们都希望模型对输入扰动是不敏感的，这通常能提高模型的泛化性能。也就是说，我们希望$\|x_1 - x_2\|$很小时$\|f_w(x_1) - f_w(x_2)\|$也尽可能地小。利用Lipschitz Continuity，整个模型被一个线性函数"控制"住，即

$$\|f_w(x_1) - f_w(x_2)\| \leq C(w) \cdot \|x_1 - x_2\| \tag{37}$$

这便是L约束了。

换言之，在这里我们认为满足L约束的模型才是一个好模型～并且对于具体的模型，我们希望估算出$C(w)$的表达式，并且希望$C(w)$越小越好，越小意味着它对输入扰动越不敏感，泛化性越好。

神经网络中的L约束

在这里我们对具体的神经网络进行分析，以观察神经网络在什么时候会满足L约束。

简单其间，考虑单层的全连接$f(Wx+b)$，这里的f是激活函数，而W, b则是参数矩阵/向量，这时候公式(3)变为

$$\|f(Wx_1+b) - f(Wx_2+b)\| \leq C(W,b) \cdot \|x_1 - x_2\| \tag{38}$$

让x_1, x_2充分接近，那么就可以将左边用一阶项近似，得到

$$\|\frac{\partial f}{\partial y} W(x_1 - x_2)\| \leq C(W,b) \cdot \|x_1 - x_2\| \tag{39}$$

这里的$y = Wx + b$。显然，要希望左边不超过右边，$\partial f/\partial y$这一项（每个元素）的绝对值必须不超过某个常数。这就要求我们要使用"导数有上下界"的激活函数，不过我们目前常用的激活函数，比如sigmoid、tanh、relu等，都满足这个条件。假定激活函数的梯度已经有界，尤其是我们常用的relu激活函数来说这个界还是1，因此$\partial f/\partial y$这一项只带来一个常数，我们暂时忽略它，剩下来我们只需要考虑$\|W(x_1 - x_2)\|$。

多层的神经网络可以逐步递归分析，从而最终还是单层的神经网络问题，而CNN、RNN等结构本质上还是特殊的全连接，所以照样可以用全连接的结果。因此，对于神经网络来说，问题变成了：如果$\|W(x_1 - x_2)\| \leq C\|x_1 - x_2\|$恒成立，那么$C$的值可以是多少？找出$C$的表达式后，我们就可以希望$C$尽可能小，从而给参数带来一个正则化项$C^2$。

谱范数/矩阵范数

其实到这里，我们已经将问题转化为了一个矩阵范数问题（矩阵范数的作用相当于向量的模长），它定义为

$$\|W\|_2 = \max_{x\neq 0} \frac{\|Wx\|}{\|x\|} \tag{40}$$

如果W是一个方阵，那么该范数又称为**“谱范数”、“谱半径”**等，但就算它不是方阵我们在优化计算中也可以称其为“谱范数（Spectral Norm）”。注意$\|Wx\|$和$\|x\|$都是指向量的范数，就是普通的向量模长。而左边的矩阵的范数我们本来没有明确定义的，但通过右边的向量模型的极限定义出来的，所以这类矩阵范数称为**“由向量范数诱导出来的矩阵范数”**。

好了，文绉绉的概念就不多说了，有了向量范数的概念之后，我们就有

$$\|W(x_1 - x_2)\| \leq \|W\|_2 \cdot \|x_1 - x_2\| \tag{41}$$

主特征根

事实上，**谱范数$\|W\|_2$等于W^TW的最大特征根（主特征根）的平方根，如果W是方阵，那么$\|W\|_2$等于W的最大的特征根绝对值。**

> 证明思路：
>
> $$\|W\|_2^2 = \max_{x\neq 0} \frac{x^TW^TWx}{x^Tx} = \max_{\|x\|=1} x^TW^TWx \tag{42}$$
>
> 假设W^TW对角化为$\text{diag}(\lambda_1, \ldots, \lambda_n)$，即$W^TW = U^T\text{diag}(\lambda_1, \ldots, \lambda_n)U$，其中$\lambda_i$都是它的特征根，而且非负，而$U$是正交矩阵，由于正交矩阵与单位向量的积还是单位向量，那么
>
> $$\begin{aligned}\|W\|_2^2 &= \max_{\|x\|=1} U^T\text{diag}(\lambda_1, \ldots, \lambda_n)U \\ &= \max_{\|x\|=1} \lambda_1 x_1^2 + \ldots + \lambda_n x_n^2 \\ &\leq \max\{\lambda_1, \ldots, \lambda_n\}(x_1^2 + \ldots + x_n^2) \\ &= \max\{\lambda_1, \ldots, \lambda_n\}\end{aligned} \tag{43}$$
>
> 从而$\|W\|_2^2$等于W^TW的最大特征根。

幂迭代（Power Iteration，迭代求解）

所谓“幂迭代”，就是通过下面的迭代格式：

随机初始化向量u，然后

$$\begin{aligned}v &\leftarrow \frac{W^Tu}{\|W^Tu\|} \\ u &\leftarrow \frac{Wv}{\|Wv\|}\end{aligned} \tag{44}$$

迭代若干次后，估计的最大特征值为

$$\sigma(W) \approx uWv \tag{45}$$

得到范数（也就是得到最大的特征根的近似值）。

> 等价形式：
>
> 初始化向量u，迭代计算
>
> $$u \leftarrow \frac{W^TWu}{\|W^TWu\|} \tag{46}$$

最大特征值为：

$$\sigma(W) = uW^TWu \tag{47}$$

证明思路：

令向量$u^{(0)} = c_1\eta_1 + \ldots + c_n\eta_n$，则$A^ru^{(0)}$为：

$$A^ru^{(0)} = c_1A^r\eta_1 + \ldots + c_nA^r\eta_n \tag{48}$$

因为$A\eta = \lambda\eta$，所以

$$A^ru^{(0)} = c_1\lambda_1^r\eta_1 + \ldots + c_n\lambda_n^r\eta_n \tag{49}$$

假设λ_1是最大的特征根，那么当r足够大时有：

$$\frac{A^ru^{(0)}}{\lambda_1^r} \approx c_1\eta_1 \tag{50}$$

所以当r足够大时，$A^ru^{(0)}$提供了最大特征根对应特征向量的近似方向，如此$u = A^ru^{(0)}/\|A^ru^{(0)}\|$是该方向的对应的单位特征向量，即

$$Au = \lambda_1u \tag{51}$$

因此

$$u^TAu = \lambda_1u^Tu = \lambda_1 \tag{52}$$

这就求出了谱范数的平方。

从奇异值分解角度的理解

先介绍一下 SVD 分解。

假设有 $m \times n$ 的方阵 A，对矩阵 A 存在这样一种分解：

$$A = U\Sigma V^T$$

当矩阵 W 是一个实数阵的时候，

- U 是一个$m \times m$的正交矩阵 (orthogonal matrix)；
- Σ 是一个$m \times n$的对角阵，对角线上的元素为非负实数，这些数称为矩阵A的“奇异值”；
- V 是一个$n \times n$的正交矩阵 (orthogonal matrix)。

矩阵 A 的奇异值计算起来也不麻烦，只需要求得$n \times n$的方阵A^TA的特征值，然后把特征值开平方根，即为矩阵A的奇异值，因为：

$$\begin{aligned} A^TA &= (U\Sigma V^T)^T \cdot (U\Sigma V^T) \\ &= (V\Sigma^TU^T)(U\Sigma V^T) \\ &= V(\Sigma^T\Sigma)V^T \end{aligned} \tag{53}$$

仔细研究上式所表达的几何意义。 我们知道，每一个矩阵对应着一个线性变换。而正交矩阵对应的线性变换很特殊，叫**旋转变换** (rotation)。 而对角阵对应的变换，叫做**拉伸变换** (scale)。 也就是说上式把矩阵A分解为三个变换：

1. 旋转；
2. 拉伸；
3. 旋转。

这里就很有意思了。假设神经网络的某一层权重W为一个$m \times n$的矩阵，对于输入的特征x，输出为:

$$y = W \cdot x = U\Sigma V^*x \tag{54}$$

也相当于对输入做了三次变换。 如果将输入视为一个向量的话，之后Σ对应的拉伸变换才会改变输入的长度；其他两次旋转变换都不会。

介绍完 SVD 以及几何意义之后，Spectral Normalization的做法就很简单了： 将神经网络的每一层的参数 W 作 SVD 分解，然后将其最大的奇异值限定为1， 具体地，在每一次更新 W 之后都除以 W 最大的奇异值。 这样，每一层对输入x最大的拉伸系数不会超过 1。

应用

神经网络中的近似计算

值得注意的是，迭代算法通常需要迭代多次才能比较好的逼近$\sigma(W)$，但是Spectral Normal在神经网络的每一次更新过程中只进行一次 幂迭代，仍然能取得不错的效果。 这是因为神经网络的每次迭代中参数W是在**缓慢更新**的，因此每两次迭代过程中可以认为W的奇异值是近似相等的。 因此虽然在网络的每一次迭代中，只进行一次幂迭代，但是随着网络的训练，幂迭代对奇异值的估计会越来越准。这种做法其实十分巧妙，因为神经网络的优化本身也是个迭代的过程，因此对$\sigma(W)$的逼近就可以利用神经网络的迭代逐渐来逼近，在网络参数的每一次更新过程中对$\sigma(W)$的迭代不必太准确。

谱正则化（Spectral Norm Regularization）

谱正则化（Spectral Norm Regularization）的概念是把谱范数的平方作为额外的正则项，取代简单的$l2$正则项。即损失函数变为：

$$loss = loss(y, f_w(x)) + \lambda \|W\|_2^2 \tag{55}$$

《Spectral Norm Regularization for Improving the Generalizability of Deep Learning》一文已经做了多个实验，表明"谱正则化"在多个任务上都能提升模型性能。

> **Frobenius范数**
>
> Frobenius范数，简称F范数，即
>
> $$\|W\|_F = \sqrt{\sum_{i,j} w_{ij}^2} \tag{56}$$
>
> 说白了，它就是直接把矩阵当成一个向量，然后求向量的欧氏模长。
>
> 简单通过柯西不等式，我们就能证明
>
> $$\|Wx\| \leq \|W\|_F \cdot \|x\| \tag{57}$$
>
> 很明显$\|W\|_F$提供了$\|W\|_2$的一个上界，也就是说，你可以理解为$\|W\|_2$是公式(5)中最准确的C（所有满足式的C中最小的那个），但如果你不大关心精准度，你直接可以取$C = \|W\|_F$，也能使得成立，毕竟$\|W\|_F$容易计算。
>
> **$l2$正则化**
>
> 根据Frobenius范数，为了使神经网络能够更好的满足L约束，即$C = \|W\|_2$尽可能的小，可以设计损失函数为：
>
> $$loss = loss(y, f_w(x)) + \lambda \|W\|_F^2 \tag{58}$$
>
> **从而揭示了$l2$正则化（也称为weight decay）与L约束的联系，表明$l2$正则化能使得模型更好地满足L约束，从而降低模型对输入扰动的敏感性，增强模型的泛化性能。**

谱归一化

事实上，WGAN首次提出时用的是参数裁剪——将所有参数的绝对值裁剪到不超过某个常数，这样一来参数的Frobenius范数不会超过某个常数，从而$\|f\|_L$不会超过某个常数（见Wasserstein距离章节的相关内容），虽然没有准确地实现$\|f\|_L = 1$，但这只会让loss放大常数倍，因此不影响优化结果。参数裁剪就是一种构造法，这不过这种构造法对优化并不友好。

现在我们已经有了谱范数，那么可以用最精准的方案了：**将f中所有的参数都替换为$w/\|w\|_2$（$w/\sqrt{\sigma(W)}$）**。这就是谱归一化（Spectral Normalization），在《Spectral Normalization for Generative Adversarial Networks》一文中被提出并实验。这样一来，如果f所用的激活函数的导数绝对值都不超过1，那么我们就有$\|f\| \leq 1$，从而用最精准的方案实现了所需要的L约束。

> “激活函数的导数绝对值都不超过1”，这个通常都能满足，但是如果判别模型使用了残差结构，则激活函数相当于是$x + \text{relu}(Wx + b)$，这时候它的导数就不一定不超过1了。但不管怎样，它会不超过一个常数，因此不影响优化结果。