Optimization and Assumptions

http://freemind.pluskid.org/misc/optimization-and-assumptions

posted on Free Mind on March 9, 2016
generated with pandoc on March 15, 2016
category: MISC

tags: Optimization, Thoughts, Algorithm

因为想要熬夜看 Alpha Go 和 Lee Sedol 的围棋比赛，自己的围棋知识又没有达到能全盘 follow 看懂的地步，所以决定一边看一边写点东西。跟 AI 没有什么关系，当然非要扯上一点关系的话，可以这样来 motivate 一个话题。我们都知道人和电脑各自有一些各自擅长的东西，比如电脑擅长计算，而人脑擅长……呃，这里先留白，等我想到了再补上。总之我在最初接触 optimization 这个问题的时候其实也是这样的概念：计算机算法在这里似乎相当笨拙，比如我们经常看到类似图中的例子被用来说明当计算机面临 non-convex 问题的时候会如何陷入 local minimum 出不来。

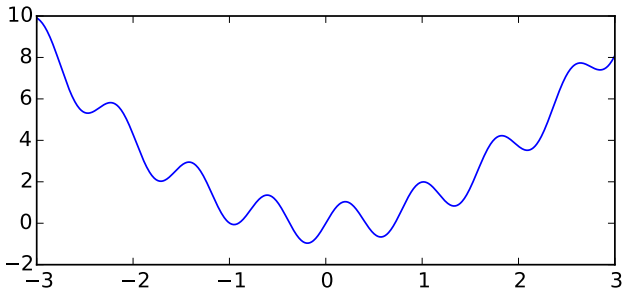

Figure 1: .....

相反对于这个例子，人“一眼”就能找到 global minimum 的位置。但是其实仔细想一想大概只是自我感觉良好而已。

首先人类到底怎样“一眼”找到这个 global minimum 的，其实我并没有什么概念，比如我自己大概一下子定位到 minimum 在中间一块，然后因为在 0 的左边和右边两个 minimum 相对比较接近一点，所以再多看了“一眼”，最终确定在 0 左边这里是 global minimum。感觉上似乎是一个非常复杂的甚至还有一些 hierarchical 的大脑计算过程，然而其实从另一个角度来看，比如从 information 的角度来看——如果我们遮住了 0 左边一半的曲线，那么我们就没法找到这个全局最优点，而遮住右边一半，虽然我们能找到全局最优，但这纯粹是运气好，我们也并不能确信自己是否找到了。所以不管大脑采用了什么样的算法，有一点我们可以确定的是它把屏幕上的所有 sample x 和它对应的函数值 $f(x)$ 全部用上了。

所谓的 sample 其实就是屏幕上显示的这一条曲线上的点。我在画上面这个图的时候实际上是在 -3 和 +3 之间平均取了 1000 个点 $x_1, \ldots, x_{1000}$，然后把对应的函数值画出来。虽然看起来是光滑的曲线，但是它并不是数学意义上的光滑曲线。而我们人脑不论使用什么算法，实际上 input 也只是这 1000 个离散点处的函数值而已。仅凭 1000 个点判断出 global optimum，人类似乎很厉害。但是其实这样的事情电脑也可以轻松做到：

只要线性地过一遍 $f(x_1), \ldots, f(x_{1000})$ 这个数组，电脑可以在比我快得多的时间内找到最优值，甚至不用 “多看一眼”。

到这里可以看到，其实人类所谓的 “看一眼” 找到最优值，计算机如果想做的话也可以不费吹灰之力地做到。可是为什么计算机不这样做，而是去用一些看起来笨笨的像 gradient descent 这样的，在一个小小的局部最优坑里就出不来了的，鼠目寸光的，……，的算法呢？原因有非常非常多。我们下面列举一些，并和 “人肉优化” 进行比较一下，我们会看到其实计算机算法会碰到的困难人都是会碰到的。

连续性: 一个优化问题一般 formulate 成寻找 $f(x)$ 的最小值，同时会对 f 做一些假设。从最弱（没有任何假设）到最强（比如明确知道 $f(x) = x^2 + \sin(8x)$）中间会有很多过渡状态，比如 f 是连续、光滑、strongly convex 之类的等等。在最弱（没有假设）的情况下，问题是无解的。

如果我们对 $f(\cdot)$ 一无所知，此时 $f(\cdot)$ 成为一个黑盒子，通常叫做 0-th order oracle，我们可以给定一个 x，这个 oracle 会返回对应的函数值 $f(x)$，如果同时还返回该点的 (sub)gradient $f'(x)$ 的话，一般就是我们所熟知的 first order oracle。刚才我们所说的人 “看一眼” 和计算机扫描数组，都算是 zero order method。在不加任何假设的情况下不能保证找到最优解的一个最明显的难点就是函数的连续性。

不论我们采样了多少 sample，总归是有限多个点，如果对在没有被采样到的点处的 behavior 没有任何限制的话，我们将永远不知道是否真正找到了全局或者甚至是局部最优点。比如上面的图，如果采样率放大一万倍，你怎么知道在之前的 x_{20} 和 x_{21} 之间没有非常 crazy 的情况出现呢？举一个极端的例子，考虑函数

$$f(x) = x^2 + \sin(8x) - 1001_{x=\sqrt{2}}$$

其中红色的项只在一个无理数上取非零值，不论是采用浮点数存储计算的计算机程序，还是通过把图像画在电脑屏幕上来 “看一眼” 的人类，都不可能采样到那一点，所以最终谁都无法找到真正的最优解。而上面的例子中我们 “看” 到的最优解，也不过是一种假象。

连续性假设的引入差不多就是为了避免这种情况。回忆一下，我们说一个函数 f 是 L-Lipschitz 连续的，是指对其定义域内任意两点 x_1, x_2，满足

$$|f(x_1) - f(x_2)| \leq L|x_1 - x_2|$$

这里的绝对值 $|\cdot|$ 可以可以换成任意度量空间上的度量。这样一来，我们就能对 f 在没被采样到的点处的行为进行刻画了。最简单考虑一个一维情况的例子：比如我们采样了相邻的两点 $x_1 = 0.1$ 和 $x_2 = 0.2$，带入连续性假设，我们可以得到，对于任意 $x \in [x_1, x_2]$，我们有

$$\max \begin{Bmatrix} f(x_1) - L(x - x_1) \\ f(x_2) - L(x_2 - x) \end{Bmatrix} \leq f(x) \leq \min \begin{Bmatrix} f(x_1) + L(x - x_1) \\ f(x_2) + L(x_2 - x) \end{Bmatrix}$$

在一维的情况下，可以形象地画出来，在 x_1 和 x_2 两点之间，$f(x)$ 的取值范围被限制在了一个菱形方框以内。这样一来，只要 L 足够小，采样点的密度足够大，我就能保证至少我找到的最优解即使不是真实的最优解，也“八九不离十”。而实际上我们也可以看到，很多优化算法一上来就假设 f 是 L-Lipschitz 连续的，差不多也是没办法。

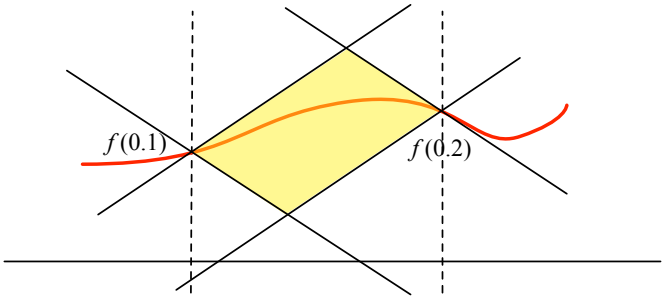

Figure 2: .....

有界性：很显然，即使有了连续性，如果我们需要优化的函数定义域是无界的，那上面的采样算法还是不会 work。比如说，万一最开始那个图，我把镜头往右移动到 10000 左右的地方，发现 $f(x)$ 下降到 -100 的范围了呢？我们觉得 gradient descent 之类的鼠目寸光，只能看到局部的东西，但是我们自己又何尝不是井底之蛙呢？如果定义域本身是无限的，那么不管你眼睛睁多大，都只是一个“局部”。

所以，在实际优化问题中，经常用到的一个假设还有定义域的有界性。比如一般会假设只考虑包含在半径为 B 的球以内的 x。当然，也有可能会有其他的方式来做假设，比如如果通过某种性质知道 $f(x)$ 在“远处”的时候值趋向于无穷大，进而超过某个范围之后就可以不管了，这之类的。

实际上，在假设了连续性和有界性的情况下，“看一眼”或者说平均采样求值然后取最小的办法似乎不失一种有效的算法。比如假设我们需要在 $[-R, R]$ 这个区间上寻找函数 $f(x)$ 的最小值，可以平均地采样 m 个点，然后令 $\hat{x}$ 是这 m 个点中函数值最小的一个点。现在假设 $x^\star$ 是真实的最小值点，令 $\tilde{x}^\star$ 是这 m 个采样点中离 $x^\star$ 最近的一个点，根据连续性，我们有

$$f(x^\star) \geq f(\tilde{x}^\star) - L|\tilde{x}^\star - x^\star| \geq f(\tilde{x}^\star) - \frac{2RL}{m-1}$$

同时根据定义，$f(\hat{x}) \leq f(\tilde{x}^\star)$，由此得到

$$f(\hat{x}) \leq f(x^\star) + \frac{2RL}{m-1}$$

是不是很棒？对比一下，对于普通的 Lipschitz 连续并且 convex 的情况下做 subgradient descent，得到的 gurantee 也只是 $\mathcal{O}(RL/\sqrt{m})$，比如参见 [Bubeck, 2015] 的定理 3.2。简直不敢相信自己的眼镜，暴力等距采样法甚至在不要求函数是 convex，也不需要任何求导操作的情况下，比 convex 情况下用 subgradient descent 的 convergence 速度还要快（假设我们按照顺序，每个 iteration 做一个新的点的 query，所以采样的个数就等于迭代次数）。问题出在哪里？

维度灾难：这个词相信大家都已经相当不陌生了。高维度带来的问题不仅是问题本身变得非常复杂，我觉得更大的问题是我们很多在低维度里 develop 出来的 intuition 都变得不再成立，偏偏人类又没有办法很有效地 visualize 三维以上的东西——这个硬件限制一直让我觉得非常恼火。而我们这里提到的很多“错觉”，到了高维度的时候也就不攻自破。

比如人能“一眼”看到最优点，那在高维的情况下又如何呢？比如一个定义域是 $\mathbb{R}^2$ 的函数，我们可以用一个三维的 surface 画出来（当然又会投影到电脑的屏幕平面上），是否还能“一下子”找到全局最优点呢？好像也不是太困难，但是如果把图中的等高线去掉，surface 上表示函数值大小的颜色全部标成同样的颜色又会怎样？

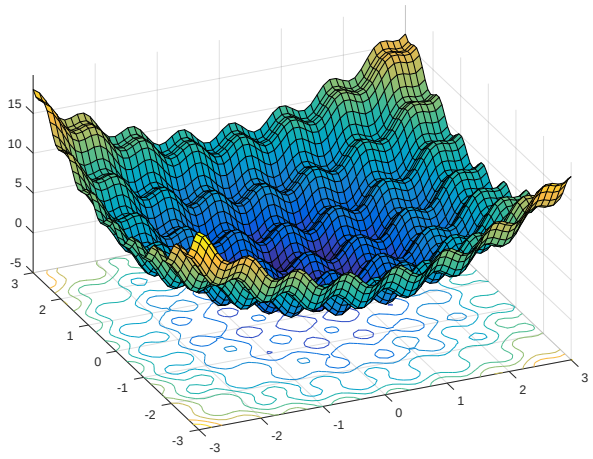

Figure 3: .....

更高维的情况呢？如果定义域本身是三维的，我就已经没法 visualize 了，而我们似乎也没有办法从“外面”来进行观察，此时如果要“看”的话，估计得前后左右上下都找一遍才能找到最优点。至于更高维的空间，实在是难以想象如果我能观察到更高维的空间，会是怎样的一种视觉冲击哈哈。

另外，对于计算机，也会碰到同样的问题。到了高维的情况下，采样法迅速变得非常的差。在数轴上我们很容易算出要使得采样密到任意一个定义域内的点到离他最近的采样点的距离不大于 ϵ，只要采样 $\mathcal{O}(1/\epsilon)$ 那么多个点就够了。在高维情况下，会变得稍微复杂一点，我们可以用 covering number 来刻画，一个集合 K 的 covering number $N_\epsilon(K)$ 是指最小的数 N，使得存在一个大小为 N 的点集 C，如果我们在每个 $x \in C$ 上放上一个半径为 ϵ 的 ball，则这些 ball 的并集包含 K。换句话说，对任意

$x' \in K$,

$$\min_{x \in C} \|x' - x\| \leq \epsilon$$

记 d-维空间里半径为 R（球心在原点）的 ball 为 $R\mathcal{B}^d$ 的话，$N_\epsilon(R\mathcal{B}^d)$ 就是我们需要达到精度 ϵ 而采样的点的个数。我们可以简单估算一下一个 d-维实心球的 covering number。假设 C 是 $R\mathcal{B}^d$ 的一个 ϵ-covering，显然

$$R\mathcal{B}^d \subset \bigcup_{x \in C} \left\{ x + \epsilon \mathcal{B}^d \right\}$$

两边同时求体积，可以得到

$$R^d \leq \sum_{x \in C} \epsilon^d = |C| \epsilon^d$$

因此 C 的元素个数，也就是 $R\mathcal{B}^d$ 的 covering number 是大于等于 $(R/\epsilon)^d$ 的。可以看到此时采样法的收敛速度将会随着维度的升高而变得奇慢无比。而对比起来，之前提到的 subgradient descent 的收敛速度是并不依赖于维度的——这个好像也有点 too good to be true 是吧？在更具体的实际问题中，通常像 Lipschitz constant L 之类的会随着维度增加而变大，所以也不能说完全是不依赖于 dimension 的，总之还要看具体问题而定了。

局部-全局：最后，在高维度下面采样法的失败导致我们从 sample complexity 的角度来说就无法得到函数的一个 “全局观”，于是只能退而求其次，寻求 “局部算法”，通过对问题添加更多的假设，使得我们可以得到诸如 gradient 或者 subgradient 之类的信息，只用局部的信息来进行 navigate。感觉就好像，来到了一个一百万维的世界，由于阳光同时从一百万个维度上照射过来，亮得整个世界只剩一片纯白，从此我开始闭着眼睛摸着石头走路。

虽然实属无奈之举，那摸着石头走路能不能保证我们能很接近全局最优点呢？很遗憾的是，不能——除非我们再加额外的条件，一些能够建立局部和全局之间的关系的条件。比如非常常见的一个条件就是函数 f 的 convexity。回顾一下 convexity 要求在每一点上存在一个 hyperplane，使得函数图像在这个 hyperplane 的 “上面”。在 convex 的情况下，比如我在某一点 x_0 上发现函数 f 的 gradient 等于零（假设 f 可导的话），注意 gradient 是一个只依赖于 f 在 x_0 附近的一个任意小的领域内的性质所定义的量，我们可以把 f 在稍远处的函数值随便怎么折腾，都不会影响原来那个局部点的 gradient。而有了 convexity 的条件之后，这个非常局部的性质一下子就能刻画函数的一个全局性质——也就是全局最优解的位置。因为 gradient 为零表示我们能够找到一个水平的 hyperplane 来

lower bound 住函数的图像，换句话说，任意 $x \neq x_0$ 的点的函数图像都不会低于 $f(x_0)$，也就是说，x_0 是全局最小值点。

当然 convexity 并不是唯一的能联系局部和全局的特性。对于某些具体的问题，可能有一些非常特殊的 structure，也能帮助我们在局部 navigate 的情况下最终找到全局最优点。比如对某些带有随机性的函数，我们可以通过某些方法大致估计出其 global optimum 的位置范围（作为算法的 initialization），并且 f 限制到某一个范围之内是 locally convex 或者只有 unique minimizer 之类的。再比如之前一篇尝试解释 deep neural network 优化的文章 [Choromanska et al., 2015]，其思路就是试图证明 DNN 中的目标函数 f 的绝大部分 local minimum 点 x 处的目标函数值 $f(x)$ 都跟全局最优的 $f(x^\star)$ 差不了太多。虽然这篇文章里面做了过多的简化和不合实际的假设导致最终的结论和实际的 DNN Training 过程中的目标函数的性质之间的联系变得非常不 clear。

实际上，近些年来 non-convex 的优化算法也变得越来越受到关注，一方面是 empirically 很多 non-convex 的算法效果非常好（比如 DNN training、EM algorithm、dictionary learning 里面的 alternative minimization 之类的），另一方面有些算法（比如 Semi-definite Programming）虽然是 convex 的，但是速度比较慢，通过 re-formulate 成 non-convex 的问题有时反而会得到非常好的算法。

最后，一些 take-home message：暴力搜索并不一定就是很烂的；低维情况跟高维情况往往差别是很大的；要解优化问题是需要各种 assumption 的；assumption 通常都是有 intuition 可循的；convex 问题不一定就是可以（efficiently）解的；non-convex 问题不一定就是没有 convergence guarantee 的；一边看围棋比赛一边写 blog 是不行的，最后还是得重写过。

顺便说一句，我们在这里逐渐加上的各种 assumption 相当于是对问题添加各种 structure，在 Information Theory 里，我们有用于衡量概率模型的有序程度的 entropy 的概念，而这里虽然并没有 randomness 出现，但是实际上同样有这样的结构性程度的 notion 在里面，比如实际上有一个叫做 Kolmogorov complexity 的东西，用来衡量一个 “object” 的复杂度，具体就不在这里展开讲了。从哲学意义上来说，entropy 或者 complexity 的大小刻画了一个 fundamental 的 “难度”，通过对问题添加一些 structure 的 assumption，将 “难度” 降低之后，才逐渐变得 approachable。

References

[Bubeck, 2015] Bubeck, S. (2015). Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357.

[Choromanska et al., 2015] Choromanska, A., Henaff, M., Mathieu, M., Ben Arous, G., and LeCun, Y. (2015). The loss surfaces of multilayer networks. In *AISTATS 2015*, pages 192–204.