Gradient Descent, Wolfe's Condition and Logistic Regression

http://freemind.pluskid.org/machine-learning/gradient-descent-wolfe-s-condition-and-logistic-regression

posted on Free Mind on January 3, 2014
generated with pandoc on December 3, 2015
category: Machine Learning

tags: Optimization, Julia, Algorithm

本文是 Machine Learning and Optimization 系列的第一篇。我会尝试介绍一些机器学习中的常用的优化算法和相关的基本理论，并结合具体的机器学习模型的例子进行简单分析和示例。系列相关的示例代码会全部放在 github/MLOpt.jl 上。此外由于我挖了太多的“系列”坑了连自己都记不住了，于是专门做了一个 Series 页面将这些坑搜集起来，并且每个系列做一个自己的简要页面并列出系列里的文章，方便索引。此外每篇文章如果属于某一个系列，在文章右边会有系列链接。Optimization 一直是 Machine Learning 里的一个重要部分，关于试图写本系列的介绍文章的 motivation，详见 Machine Learning and Optimization 的系列主页，下面直接进入正文。

首先考虑最简单的无约束的光滑函数的优化问题，最小化 $f:\mathbb{R}^d\to\mathbb{R}$，并假设 f 是可导的。此时如果 x^* 是 f 的一个 minimizer 的话，必须要满足什么样的条件呢？但凡是学过微积分的同学应该都会想到导数等于零这个条件。如若不然，假设 $\nabla f\neq 0$，考虑 f 在 x^* 处的一阶近似：

$$f(x^*+v)=f(x^*)+\langle\nabla f,v\rangle+o(\|v\|)$$

令 $v=-t\nabla f$，当正数 t 足够小时，

$$f(x^*-t\nabla f)=f(x^*)-t\|\nabla f\|^2+o(t)<f(x^*)$$

因此 x^* 不能是（局部）minimizer。当 f 是 convex 的时候，这个条件也能保证 x^* 是 global minimizer。但是 general 的情况下这只是一个必要条件，x^* 有可能只是一个局部 minimizer 而非全局最优，或者是一个 local maximizer 或者甚至什么都不是。但是在实际问题中有时候就是会碰到非 convex 又没有太多其他结构可以利用的情况，没有办法有效地找到全局最优解，所以只能找一个 local minimizer 或者只是一个 $\nabla f=0$ 的点，通常称为 stationary point。例如 $f(x)=x^3$ 在 $x=0$ 处就是一个 stationary point。

寻找 f 的 stationary point 其实就是找方程 $\nabla f(x)=0$ 的解的问题，所以 optimization 的问题和解方程的问题经常都可以互相转化。一个最基本的算法就是 gradient descent，它是一个迭代的算法，通过依次地生成点 $\{x_k\}$ 逐渐接近 stationary point x^*。它的迭代公式是

$$x_{k+1}=x_k-t_kA_k\nabla f(x_k)\tag{1}$$

其中 $t_k > 0$，$A_k \succ 0$ 是一个正定矩阵，根据 t_k 和 A_k 的取法不同会产生出各种各样的变种算法。首先可以注意到的一点是，该算法如其名，根据当前的 gradient 来决定一个 descent 的方向进行移动。根据 f 在 x_k 处的一阶近似，如果步长 $\|x_{k+1} - x_k\|$ 足够小的话，我们可以保证每一步迭代 f 的值是减小的，而到达 stationary point 的时候迭代将会停止。

当 f 下有界的时候，单调递减的迭代可以保证收敛到一个极限，但是这样还不能保证找到 stationary point：首先收敛的极限可能并不是我们所期望的那一个，其次函数值 $f(x)$ 的收敛也并不一定保证 x 的收敛。

如果想要保证收敛性，我们必须对 f 做一些进一步的假设或限制，针对不同的 t_k 和 A_k 的取法也会有不同的结论。注意到 x^* 是迭代公式 (1) 的一个不动点：

$$x^* = x^* - tA\nabla f(x^*)$$

简单起见，我们考虑 $A = I$，以及固定 t 为常数的情况，假设 f 二阶可导并且其 Hessian 满足 $lI \preceq H(x) \preceq LI$，其中 $0 < l \leq L$。此时我们有

$$\begin{aligned}
\|x_{k+1} - x^*\|^2 &= \|x_k - t\nabla f(x_k) - x^*\|^2 \\
&= \|x_k - x^*\|^2 - 2t\langle x_k - x^*, \nabla f(x_k)\rangle + t^2\|\nabla f(x_k)\|^2 \\
&= \|x_k - x^*\|^2 - 2t\langle x_k - x^*, \nabla f(x_k) - \nabla f(x^*)\rangle + t^2\|\nabla f(x_k) - \nabla f(x^*)\|^2 \\
&= \|x_k - x^*\|^2 - 2t\langle x_k - x^*, H(\xi)(x_k - x^*)\rangle + t^2\|H(\xi)(x_k - x^*)\|^2 \\
&\leq (1 + t^2L^2 - 2tl)\|x_k - x^*\|^2
\end{aligned}$$

其中 ξ 是根据中值定理中存在的一点。只要 $1 + t^2L^2 - 2tl < 1$，亦即 $t < 2l/L^2$，就能保证

$$\lim_{k\to\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} \leq \sqrt{1 + t^2L^2 - 2tl} < 1$$

此时我们能得到所谓的 Q-线性收敛。当然这个算法实际中比较难严格地应用，一方面要估计合理的 l 和 L 可能会比较困难，另一方面可能要优化的函数本身并不满足这里的 assumption。实际上，我们这里的 assumption 还是很强的，首先 $H \succeq lI$ 实际上说 Hessian 是正定的，因此我们已经能保证 f 是 convex 的了，更确切地说，它保证了 f 的 strong convexity，此时 stationary point x^* 如果存在的话，就一定是 global minimizer。此外，$H \preceq LI$ 的条件可以理解为在说 gradient 的 Lipschitz 连续性。

定义 1 (Strongly Convex Functions) 当 f 满足

$$f(y) \geq f(x) + \langle \nabla f(x), y - x\rangle + \frac{\alpha}{2}\|y - x\|^2$$

时，称 f 为 α-*strongly convex* 函数，其中参数 $\alpha > 0$。当 $\alpha = 0$ 时退回到普通的 *convex* 函数的定义。

直观来讲，convex 性质是指函数曲线位于该点处的切线，也就是线性近似之上，而 strongly convex 则进一步要求位于该处的一个二次函数上方，也就是说要求函数不要太 “平坦” 而是可以保证有一定的 “向上弯曲” 的趋势。当然这是一个很强的 assumption，但是同时也是非常重要的 assumption。比如我们都知道如果函数 f 是 convex 的，那么我们可以保证 $\nabla f(x) = 0$ 的点将是一个 global minimizer，但是在实际的数值计算中，我们很少能直接求解 $\nabla f(x) = 0$ 这个方程，而是需要通过一些迭代的算法来得到一个近似的估计，比如得到一个 x_* 使得 $\|\nabla f(x_*)\| < \epsilon$，但是仅仅靠 convex 性质并不能保证在这种情况下得到的点 x_* 会是一个比较好的 global minimizer 的近似点。比如 f 在 global minimizer 周围是非常平坦的情况的话，我们有可能会找到一个很远的点但是起 gradient 的 norm 非常小。此时如果我们有 strongly convexity 的话，就能对情况做一些控制，考虑 strongly convex function 的定义式，令 $y = x^*$ 是 global minimizer，而 $x = x_*$ 为我们得到的近似解，于是

$$\begin{aligned} f(x^*) &\geq f(x_*) - \langle \nabla f(x_*), x_* - x^* \rangle + \frac{\alpha}{2}\|x_* - x^*\|^2 \\ &\geq f(x_*) - \|\nabla f(x_*)\|\|x_* - x^*\| + \frac{\alpha}{2}\|x_* - x^*\|^2 \\ &\geq f(x_*) - \epsilon\|x_* - x^*\| + \frac{\alpha}{2}\|x_* - x^*\|^2 \end{aligned}$$

由于 $f(x^*) \leq f(x_*)$，所以

$$\frac{\alpha}{2}\|x_* - x^*\|^2 - \epsilon\|x_* - x^*\| \leq f(x^*) - f(x_*) \leq 0$$

得到

$$\|x_* - x^*\| \leq \frac{2\epsilon}{\alpha}$$

也就是说通过求方程 $\nabla f(x) = 0$ 的更好的 ϵ 近似，我们可以保证得到原优化问题的更好的近似。当然，这个 bound 的好坏也要取决于 strongly convex 性质中的常数 α 的大小。

再回到迭代优化算法上，除了我们这里提到固定 t_k 为常数的情况外，另外一种比较常见的算法是用 line search 来找最优的 t_k：就是在确定了前进方向 $-A_k\nabla f(x_k)$ 之后，令 $q_k(t) = f(x_k - tA_k\nabla f(x_k))$，然后通过最小化一元函数 $q_k(t)$ 来得到 $t_k = \arg\min q_k(t)$，因为是沿着确定的方向最小化，所以又叫做 line search。通常即使原来的函数 f 比较复杂，对应的 $q_k(t)$ 可能会变得很简单或者甚至有 closed form

minimizer。直观上来看这种算法要更加 aggressive 一点，不过从收敛性的角度来说，并没有比固定 t_k 的算法有本质上的区别，在 ∇f 是 Lipschitz 连续的情况下可以证明收敛性，但是如果要保证收敛速度，必须像上面那样做进一步的假设，可以得到类似的线性收敛速度。具体证明可以参考例如 [Gilbert and Lemarechal, 2006] 的第 2.5 小节或者 [Boyd and Vandenberghe, 2004] 的 9.3.1 节。

这里我们不妨做一个小结：刚才我们所提到的迭代优化算法，在每一次的迭代中通过将原优化问题分解为两个较为简单的优化问题来逐步逼近最优解：

1. 寻找局部最优方向，由于是在局部操作，根据函数的光滑性或者其他一些预先知道的结构信息，我们可以对函数进行局部近似，从而将问题简化。最常用的近似就是泰勒展开，其中各种 gradient descent 方法中主要用一阶泰勒展开，而使用二阶展开则可以得到牛顿法，还有最近机器学习中比较多见的 Proximal Method 则是将目标函数分解成简单的和复杂的部分，简单的部分保持原样而只对复杂的部分进行泰勒展开近似。f 在 x_k 处的一阶展开为 $\bar{f}(x) = f(x_k) + \langle x - x_k, \nabla f(x_k)\rangle$，此时寻找局部最优移动方向则对应优化问题

   $$\min_v \bar{f}(x_k + v), \quad s.t. \|v\| \leq \delta$$

   其中 δ 是用于将搜索限制在一个局部邻域里的，因为泰勒展开近似只能保证在局部有效。而上面这个问题中的目标函数关于 v 是线性的，所以非常容易优化。如果 $\|v\|$ 中的 norm 是 Euclidean norm，那么我们立即得到最优解为 $v = -\nabla f(x_k)$，也就是 gradient 的反方向。如果我们取 ℓ_1-norm，那么最优解将会对于 $\nabla f(x_k)$ 这个向量的绝对值最大的那个坐标轴的方向（或其反方向），如果最大绝对值的方向不唯一的话可以任取一个，这可以看成是 **Coordinate Descent** 算法的一种解释。
2. 寻找最优步长，在 exact line search 中我们是将问题化简到了最小化 $q_k(t)$ 这个一元函数的问题上。如果 $q_k(t)$ 的形式简单通常可以做 exact line search；复杂的情况则通常可以通过进一步的数值迭代算法来得到一个近似的步长，叫做 inexact line search，这样通常会让这一步计算量变得很大，因此也有许多其他算法来寻找一个 “合理” 的步长，而不一定要找到（近似）最优的步长，常见有诸如 Wolfe's Condition、Armijo Rule 之类的。

下面我们来看两个机器学习中的例子，一个是 Linear Regression，另一个是 Logistic Regression。在 Linear Regression 中，我们得到训练数据 $\{x_i, y_i\}_{i=1}^N \subset \mathbb{R}^p \times \mathbb{R}$，目标函数如下[1]：

$$\min_w \left\{ L(w) = \frac{1}{2}\sum_{i=1}^N (y_i - \langle w, x_i\rangle)^2 + \frac{1}{2}\lambda\|w\|^2 \right\}$$

写成矩阵的形式（y 为列向量，X 矩阵的行对应数据点）是

[1] 简单起见我们假设 x 的其中一个维度是常数，这样就不用显式地写成带 bias 的 $\langle w, \cdot\rangle + b$ 的形式了。

$$\min_w \left\{ L(w) = \frac{1}{2}\|y - Xw\|^2 + \frac{1}{2}\lambda\|w\|^2 \right\}$$

其中带 λ 系数的那一项是 regularizer，这种带 ℓ_2-norm 的 regularizer 的 linear regression 通常也被称为 ridge regression。从 Learning Theory 的角度来说，regularizer 是用于帮助提升模型的 generalization 能力。从数值计算的角度来说，加 regularizer 有助于处理 condition number 不好的情况下矩阵求逆很困难的问题：因为目标函数是 quadratic 的，所以实际上是有解析解的，求导并令导数等于零即可得到最优解为：

$$w^* = \left(X^T X + \lambda I\right)^{-1} X^T y$$

要 evaluate 这个解，我们通常并不直接求矩阵的逆，而是通过解线性方程组的方式（例如高斯消元法）来计算。考虑没有 regularizer 也就是 $\lambda = 0$ 的情况，如果矩阵 $X^T X$ 的 condition number 很大的话，解线性方程组就会在数值上相当不稳定，而这个 regularizer 的引入则可以改善 condition number。

既然可以直接通过解线性方程的方式来得到最优解为什么还要用麻烦的 gradient descent 来优化呢？~~因为剧本就是这么写的啊~~其实反过来看通过 gradient descent 或者其他的迭代优化算法来求解对应的 quadratic 目标函数也是另一种解线性方程组的算法：不同的算法有不同的特点，有的时候在特殊的场合下会很有优势。例如如果数据矩阵是非常稀疏的时候，也许高斯消元法中的行的加减操作会产生一些比较 dense 的中间矩阵造成存储和计算困难，但是迭代优化由于只会用稀疏矩阵来进行乘法操作，所以开销可能会相对较小一些。

当然使用迭代优化的算法，condition number 太大仍然会导致问题：它会拖慢迭代的收敛速度，而 regularizer 从优化的角度来看，实际上是将目标函数变成 λ-strongly convex 的了。首先让我们来看看目标函数对参数 w 的 gradient：

$$\nabla L(w_k) = X^T(Xw_k - y) + \lambda w_k = (X^T X + \lambda I)w_k - X^T y$$

我们考虑沿着 gradient 的反方向移动的情况，由于我们可以直接计算出 $L(w)$ 的 Hessian 为 $X^T X + \lambda I$，我们可以直接套用本文一开始的分析方法选择一个固定的 t_k 来得到 Q-线性收敛。不过这里由于 quadratic function 形式简单我们可以直接用 exact line search。方便起见，我们记 $Q = X^T X + \lambda I, b = -X^T y, c = 1/2\|y\|^2$，则

$$\begin{aligned} L(w) &= \frac{1}{2} w^T Q w + b^T w + c \\ \nabla L(w) &= Qw + b \\ q_k(t) &= L(w_k - t\nabla L(w_k)) \end{aligned}$$

注意到 $q_k(t)$ 是关于 t 的一个二次函数，对 t 求导得到

$$\begin{aligned} q_k'(t) &= t\nabla L(w_k)^T Q\nabla L(w_k) - \nabla L(w_k)^T(Qw_k + b) \\ &= t\nabla L(w_k)^T Q\nabla L(w_k) - \nabla L(w_k)^T\nabla L(w_k) \end{aligned}$$

于是最优解为

$$t_k = \frac{\nabla L(w_k)^T\nabla L(w_k)}{\nabla L(w_k)^T Q\nabla L(w_k)}$$

带入 gradient descent 的迭代公式，我们可以计算出

$$\begin{aligned} \frac{L(w_{k+1}) - L(w^*)}{L(w_k) - L(w^*)} &= 1 - \frac{t\nabla L(w_k)^T\nabla L(w_k) - 1/2t^2\nabla L(w_k)^T Q\nabla L(w_k)}{L(w_k) - L(w^*)} \\ &= 1 - \frac{\frac{1}{2}\frac{\left(\nabla L(w_k)^T\nabla L(w_k)\right)^2}{\nabla L(w_k)^T Q\nabla L(w_k)}}{L(w_k) - L(w^*)} \\ &= 1 - \frac{\frac{1}{2}\frac{\left(\nabla L(w_k)^T\nabla L(w_k)\right)^2}{\nabla L(w_k)^T Q\nabla L(w_k)}}{\frac{1}{2}w_k^T Qw_k + b^T w + \frac{1}{2}b^T Q^{-1}b} \\ &= 1 - \frac{\left(\nabla L(w_k)^T\nabla L(w_k)\right)^2}{\nabla L(w_k)^T Q\nabla L(w_k)\nabla L(w_k)^T Q^{-1}\nabla L(w_k)} \end{aligned}$$

根据 Kantorovich 不等式，可以得到

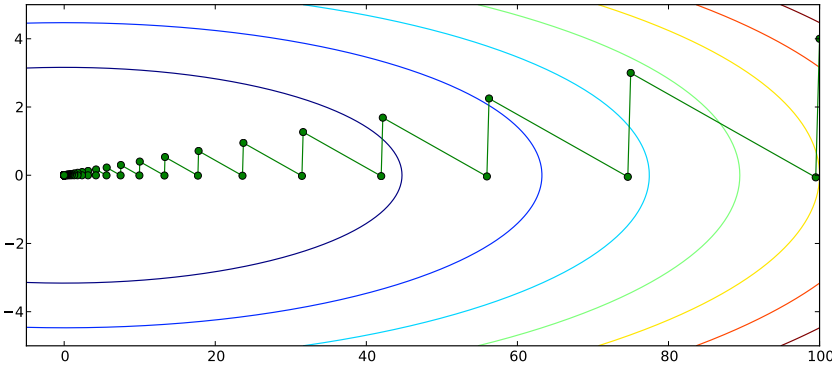
$$\frac{L(w_{k+1}) - L(w^*)}{L(w_k) - L(w^*)} \leq 1 - \frac{4lL}{(l+L)^2}$$

其中 l 和 L 是矩阵 Q 的最小和最大的特征值。根据 condition number $\kappa(Q)$ 的定义 $\kappa(Q) = L/l$，我们可以进一步得到

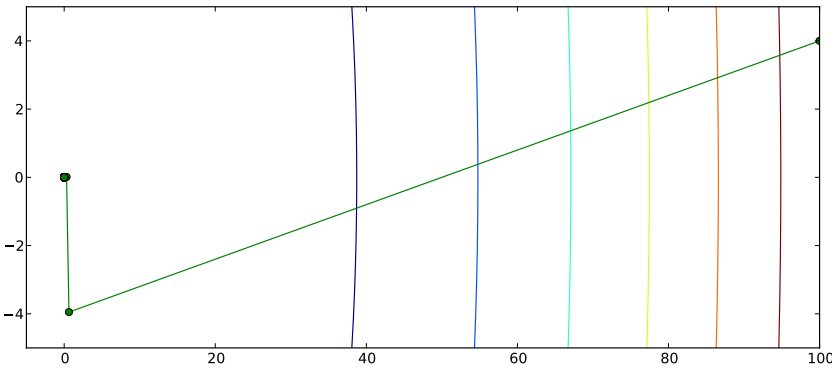
$$\frac{L(w_{k+1}) - L(w^*)}{L(w_k) - L(w^*)} \leq \left(\frac{1-\kappa(Q)}{1+\kappa(Q)}\right)^2 = \left(1 - \frac{2}{1/\kappa(Q)+1}\right)^2$$

右边是 gradient descent 的目标函数收敛速率的一个上界，可以看到 Q 的 condition number 越小，上界就越小，也就是收敛速度会越快。注意到 $Q = X^T X + \lambda I$，通过特征值的定义即可简单验证，假设 $X^T X$ 的 condition number 是 L'/l' 的话，加上 regularizer 之后它的 condition number 变为 $(L'+\lambda)/(l'+\lambda)$，会变小，从而改善收敛速度。

虽然这只是收敛速度的一个 upper bound，但是 condition number 对实际的收敛速度影响还是蛮大的，图 1 给出了两个不同的 condition number 的情况。



(a) **condition number 200。** 迭代 60 次收敛



(b) **condition number 2。** 迭代 5 次收敛

Figure 1: .....

为了让图好看一点，我给图中的横纵坐标用了不同的比例，所以不太看得出来 exact line search 下 gradient descent 的一个性质：前后两次迭代的移动方向是正交的。注意到第 k 次迭代是沿着 $-\nabla f(x_k)$ 的方向移动 t_k 使得 $q_k(t)$ 最小，因此

$$0 = q_k'(t_k) = \left.\frac{df(x_k - t\nabla f(x_k))}{dt}\right|_{t=t_k} = -\langle \nabla f(x_k), \nabla f(x_{k+1}) \rangle$$

所以如果横纵坐标轴的比例一样的话，收敛路径应该看起来是正交的 zig-zag 形状。实际中如果 $q(t)$ 比较复杂不能直接优化的话，由于是一元函数，可以用一个通用的二分查找法来寻找 $q'(t) = 0$ 的点。

上面是针对一个 quadratic 函数来进行的具体分析，对于我们刚才讨论过的 general 的函数 f 并且其 Hessian 满足 $lI \preceq H(x) \preceq LI$ 时，L/l 作为其 Hessian 的 condition number 的一个上界同样会以类似的方式影响 gradient descent 的收敛速度。例如可以参考 [Boyd and Vandenberghe, 2004] 里的 9.3.1 节。

接下来我们再来看一个复杂一点的 Logistic Regression (LR) 的例子。LR 是 machine learning 中的一个经典的分类算法，虽然挂着 “Regression” 的名号，但是确实如假包换的 classification 算法。LR 的基本假设是数据类别间是由一个线性的 decision boundary 隔开的，换句话说

$$\log \frac{P(y=+1|x)}{P(y=-1|x)} = w^T x$$

再结合

$$P(y=+1|x) + P(y=-1|x) = 1$$

可以解得：

$$\left.\begin{aligned} P(y=+1|x) &= \frac{\exp(w^T x)}{1+\exp(w^T x)} = \frac{1}{1+\exp(-w^T x)} \\ P(y=-1|x) &= \frac{1}{1+\exp(w^T x)} \end{aligned}\right\} = \frac{1}{1+\exp(-y w^T x)} \tag{2}$$

在 training data 上进行 maximum log-likelihood 参数估计是

$$\begin{aligned} \max_w \log \prod_{i=1}^N P(y_i|x_i) &= \min_w -\sum_{i=1}^N \log \frac{1}{1+\exp(-y_i w^T x_i)} \\ &= \min_w \sum_{i=1}^N \log\left(1+\exp(-y_i w^T x_i)\right) \end{aligned} \tag{3}$$

这个 binary 的情况所具有的特殊形式还可以从另一个角度来解释：先抛开 LR，直接考虑 Empirical Risk Minimization (ERM) 的训练规则，也就是最小化分类器在训练数据上的 error：

$$\mathcal{E} = \sum_{i=1}^N \mathbf{1}\{y_i \neq \text{sign}(f(x_i))\} = \sum_{i=1}^N \mathbf{1}\{y_i f(x_i) \leq 0\}$$

但是这是个离散的目标函数优化非常困难，所以我们寻求函数 $\mathbf{1}(\cdot \leq 0)$ 的一个 upper bound $\ell(\cdot)$，然后去最小化

$$\mathcal{E}' = \sum_{i=1}^N \ell(y_i f(x_i))$$

当取 $\ell(x) = \log(1+\exp(-x))$（该函数通常称作 log loss）时[2]，即得到同 (3) 一样的式子，也就是 LR 的目标函数，并且我们接下来会看到，这个 ERM 的 upper bound 是易于优化的。顺便提一句，通过选择其他的 upper bound，我们会导出其他一些常见的算法，例如 Hinge Loss 对应 SVM、exp-loss 对应 Boosting。注意到 log loss 是 convex 的，有时候我们还会加上一个 regularizer

[2] 如果要严格地作为一个 upper bound，我们需要使用以 2 为底的对数。不过由于只是对 loss function 做一个常数缩放，对优化结果并没有什么影响，所以方便起见我们实际使用自然对数。

$$L(w) = \mathcal{E}'(w) + 1/2\lambda\|w\|^2$$

此时目标函数是 λ-strongly convex 的。接下来我们考虑用 gradient descent 来对目标函数进行优化。首先其 Gradient 是

$$\nabla L(w) = \sum_{i=1}^{N} -\frac{\exp(-y_i w^T x_i)}{1+\exp(-y_i w^T x_i)} y_i x_i + \lambda w$$

其中 $P_w(y|x)$ 由式 (2) 给出。这里我们使用 Wolfe's Condition 来选择 step size。Wolfe's Condition 有两个参数 $0 < m_1 < m_2 < 1$，设 p_k 是移动方向（并不一定是 $-\nabla f(x_k)$）：

1. $f(x_k + tp_k) \le f(x_k) + m_1 t\langle \nabla f(x_k), p_k\rangle$，亦即 $q(t) \le q(0) + m_1 t q'(0)$
2. $\langle \nabla f(x_k + tp_k), p_k\rangle \ge m_2\langle \nabla f(x_k), p_k\rangle$，亦即 $q'(t) \ge m_2 q'(0)$

可以证明当 f 连续可导并且有下界时，一定存在一个区间内的步长都满足 Wolfe's Condition。并进而可以证明按照 Wolfe's Condition 选择步长的算法在 gradient 是 Lipschitz 连续并且移动方向和 gradient 的负方向不会偏离太远的话，可以保证算法收敛到一个 stationary point，详见 [Gilbert and Lemarechal, 2006] 引理 3.8 和定理 3.9，顺便盗一张图来说明一下：

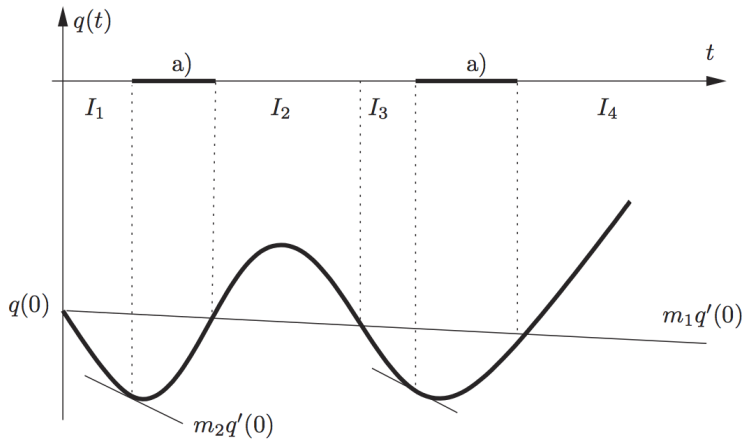


Figure 2: .....

第一个条件主要是用来限制步长不能太大，因为除非 $q(t)$ 随着 t 增长无下界[3]，否则一定存在 t' 使得所有 $t > t'$ 都不能满足第一条。第二条则限制步长不要太短，因为如果步长衰减太快的话可能就没法收敛到 stationary point 了，因为 p_k 是是下降方向（和 gradient 成钝角），所以 $q'(0) < 0$，exact line search 是要找到 $q'(t) = 0$ 的点，而这里只要求 $q'(t) \ge m_2 q'(0)$ 就可以了。图中两段黑色的区间是满足 Wolfe's Condition 的步长。

[3] 此时原优化问题也无下界。

下面的 Julia 代码示例了用二分法搜索满足 Wolfe's Condition 的步长的算法，简单来说就是，如果不满足第一条，则减小步长；不满足第二条则增加步长。

```
function linesearch{T}(p::OptimizationProblem,
                       x::Vector{T}, d::Vector{T})
    low = convert(T, 0)
    high = inf(T)

    t = convert(T, t0)
    fx = objval(p, x)
    dfx = gradient(p, x)

    while true
        if objval(p, x + t*d) > fx + c1*t*dot(dfx, d)
            high = t
            t = (low + high) / 2
        elseif dot(gradient(p, x+t*d), d) < c2*dot(dfx, d)
            low = t
            if isinf(high)
                t = 2*low
            else
                t = (low+high)/2
            end
        else
            break
        end
        if high-low < tol
            break
        end
    end
    return t
end
```

再回到 LR，我们可以直接套用上面的算法进行步长搜索。我们用 USPS 手写数字识别数据集上的分类作为示例。由于我们刚才推导的是简单的两类 LR 算法，所以我们只简单地示例手写数字 1 和 2 之间的分类：我们随机地将数据平均分为 training 和 testing 两部分，图 3 为在 training data 上跑 gradient descent 的收敛曲线。由于我们是按照 log-scale 来画 y 轴的，所以可以看到在前几次迭代之后就进入比较稳定的线性收敛趋势中。

LR 的分类非常简单，只要计算 $\text{sign}(w^T x)$ 即可，如果希望得到具体的 conditional probability 的话，可以按照 (2) 来计算概率。在 USPS 上数字

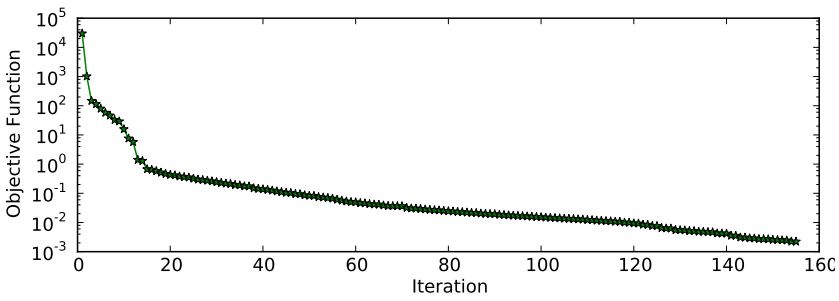


Figure 3: . . . . . .

1 和 2 的分类问题中我们通过随机初始化的 gradient descent 训练出来的 LR 模型在测试数据上的精确度平均都在 99% 以上。

References

[Boyd and Vandenberghe, 2004] Boyd, S. P. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

[Gilbert and Lemarechal, 2006] Gilbert, J. C. and Lemarechal, C. (2006). *Numerical optimization: theoretical and practical aspects*. Springer.