第二章

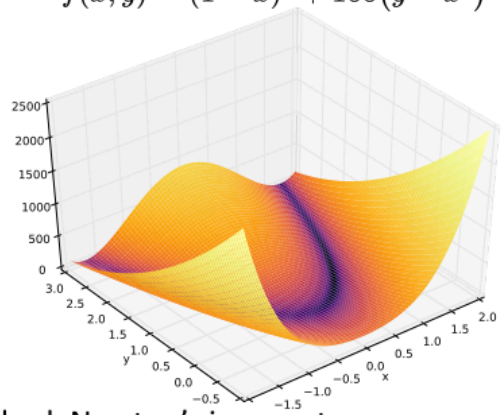
1. Why Quasi-Newton Method

Why Quasi-Newton Methods

- Gradient descent is cheap
  - Compute partials: $O(N)$
- Newton's method is expensive
  - Compute mixed partials: $O(N^2)$
  - Linear solver: $O(N^3)$
- Other problems
  - When quadratic approximation is bad, Newton's is a waste
  - Hessian becomes poorly conditioned
  - Nonconvex problems: indefinite Hessian = ascent step

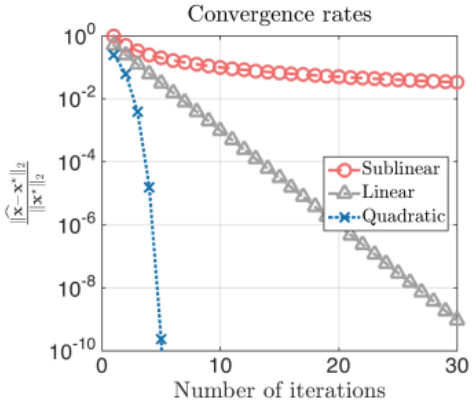
Rosenbrock banana function

$$f(x,y) = (1-x)^2 + 100(y-x^2)^2$$



- 梯度法
  - 优点：计算梯度的时间复杂度为$O(N)$
  - 缺点：没有函数的曲率信息，可能收敛慢
- 牛顿方法
  - 优点：包含曲率信息，收敛快
  - 缺点：计算Hessian复杂度为$O(N^2)$，计算最优解设计Hessian求逆，复杂度为$O(N^3)$
- 其他问题：
  - 非线性函数使用Newton方法近似效果很差的话，会造成计算Hessian和求逆资源浪费
  - Hessian奇异的时候牛顿方法不稳定
  - 非凸问题Hessian有负特征值，f(x)可能会上升到maxmia
- 期望以较低的复杂度实现超线性的收敛速率 **linear<suerlinear<quadratic**

Normally, for convex smooth functions, we have



Newton's Method　　Gradient Methods

$$e^k = x^k - x^*$$

Linear　$\|e^{k+1}\| = C\|e^k\|$

Quadratic　$\|e^{k+1}\| = C\|e^k\|^2$

Superlinear　$\|e^{k+1}\| = C\|e^k\|^p, p > 1$

As good as quadratic (why?)

Attain super-linear performance without all the fuss

2. Quasi-Newton Methods（凸且光滑函数）

Newton approximation (PD Hessian *H*)

$$f(x) - f(x^k) \approx (x - x^k)^T g + \frac{1}{2}(x - x^k)^T H(x - x^k)$$

Solve $H^k d^k = -g^k$

Quasi-Newton approximation

$$f(x) - f(x^k) \approx (x - x^k)^T g^k + \frac{1}{2}(x - x^k)^T M^k(x - x^k)$$

Solve $M^k d^k = -g^k$

What conditions should *M* satisfy?

- It should not need full 2nd-order derivatives
- The linear equations have closed-form solutions
- It should be lightweight and compact to store
- It must preserves descent directions
- It should contain curvature information (local quadratic apporx.)

1. M不需要所有二阶导信息
2. M需要构造成容易解线性方程组的形式，有闭式解
3. M必须是稀疏 轻量 紧凑
4. 方向d必须是函数值下降的方向（M严格正定来保证）
5. 必须包含函数曲率信息，Hessian的局部近似（）

拟牛顿法的想法其实很简单，就像是函数值的两点之差可以逼近导数一样，一阶导数的两点之差也可以逼近二阶导数。几何意义是求一阶导数的"割线"，取极限时，割线会逼近切线，矩阵B就会逼近Hessian矩阵的逆。

（M严格正定来保证函数值下降方向）

When is descent direction preserved?

$$M^k d^k = -g^k$$

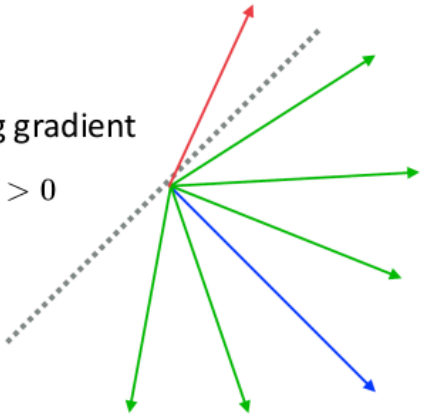
Negative gradient step

Search direction

Search direction makes acute angle with neg gradient

$$\langle -g, -M^{-1}g \rangle = \langle g, M^{-1}g \rangle = g^T M^{-1} g > 0$$

Must be PD (Positive Definite)



（不必求Hessian和求逆操作，用梯度的泰勒展开去拟合Hessian的逆B_k，使得其带有曲率信息）

Which kind of curvature info is needed?

$$\nabla f(x) - \nabla f(y) \approx H(x - y)$$

$$\Delta x = x^{k+1} - x^k$$
$$\Delta g = \nabla f\left(x^{k+1}\right) - \nabla f\left(x^k\right)$$

Secant condition

$$\Delta g \approx M^{k+1} \Delta x$$

or

$$\Delta x \approx B^{k+1} \Delta g, \; M^{k+1} B^{k+1} = I$$

Construct *B* directly from gradient/variable diff

如何来增量式更新B_k?

Infinitely many *B* satisfy the secant condition?

$$\Delta x = B^{k+1}\Delta g$$

How to choose the best one?

- Assume the function is convex
- Assume an initial guess is given

$$\min_B \ \|B - B^k\|^2 \quad \text{s.t.} \quad B = B^{\mathrm{T}},\ \Delta x = B\Delta g$$

scale invariant

$$\min_B \ \|H^{\frac{1}{2}}(B - B^k)H^{\frac{1}{2}}\|^2 \quad \text{s.t.} \quad B = B^{\mathrm{T}},\ \Delta x = B\Delta g$$

$$H = \int_0^1 \nabla^2 f\left[(1-\tau)x^k + \tau x^{k+1}\right] d\tau$$

```
1    1. 代价函数为了使得能够利用所有的$\triangle x,\triangle g$对，并且使得B变动最小
2    1. 使得B变动最小含义就是每次在利用当前点的曲率信息的时候尽可能的利用之前的点的曲率信息
```

- 严格凸函数更新自然满足B正定性

Recover the curvature (approx. inv. Hessian) from gradients

Broyden-Fletcher-Goldfarb-Shanno (BFGS) update

$$B^{k+1} = \left(I - \frac{\Delta x \Delta g^T}{\Delta g^T \Delta x}\right) B^k \left(I - \frac{\Delta g \Delta x^T}{\Delta g^T \Delta x}\right) + \frac{\Delta x \Delta x^T}{\Delta g^T \Delta x}$$

$$B^0 = I,\ \Delta x = x^{k+1} - x^k,\ \Delta g = \nabla f(x^{k+1}) - \nabla f(x^k)$$

What conditions should *M = 1/B* satisfy?

- It should not need full 2nd-order derivatives ✓
- The linear equations have closed-form solutions ✓
- It should be lightweight and compact to store ✓
- It must preserves descent directions ? strict convexity (necessary?)
- It should contain curvature information ✓

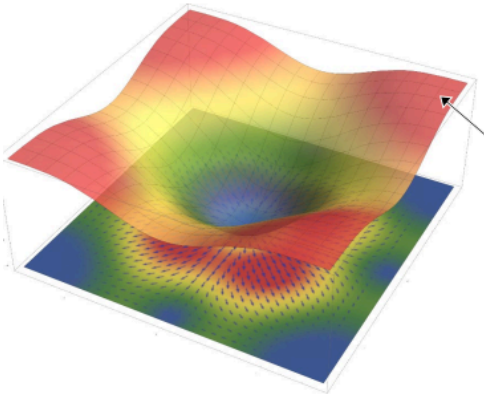
BFGS update preserves PD if $\Delta g^T \Delta x > 0$

伪代码：

BFGS method for strictly convex functions

initialize $x^0,\ g^0 \leftarrow \nabla f(x^0),\ B^0 \leftarrow I,\ k \leftarrow 0$
while $\|g^k\| > \delta$ do
$d \leftarrow -B^k g^k$
$t \leftarrow$ backtracking line search (Armijo)
$x^{k+1} \leftarrow x^k + td$
$g^{k+1} \leftarrow \nabla f(x^{k+1})$
$B^{k+1} \leftarrow \mathrm{BFGS}(B^k, g^{k+1} - g^k, x^{k+1} - x^k)$
$k \leftarrow k + 1$
end while
return

缺点：没有梯度单调性/非凸/非光滑



- negative curvature
- nonconvex
- non-monotonic grad

Restrictions:

- Strict gradient monotonicity does not hold in general
- Curvature info far from optimum is less important
- Iteration cost is quadratic in dim
- Applicability to nonconvex function remains to be verified
- Applicability to nonsmooth function remains to be verified

How to apply to more nonconvex/nonsmooth functions?

3. Quasi-Newton Methods（非凸光滑函数）

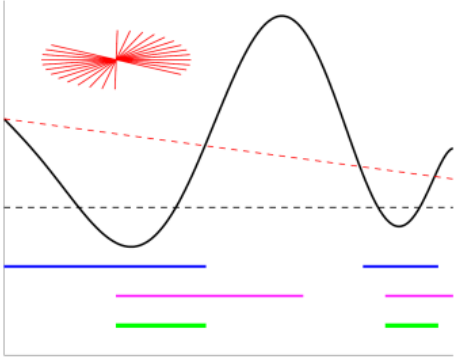
- **Weak Wolfe condition（常用）**

$\Delta g^T \Delta x > 0$? Ensure PD of approximation: Wolfe conditions!

1. weak Wolfe conditions $0 < c_1 < c_2 < 1$ typically $c_1 = 10^{-4}$, $c_2 = 0.9$

$$f(x^k) - f(x^k + \alpha d) \geq -c_1 \cdot \alpha d^{\mathrm{T}} \nabla f(x^k)$$ sufficient decrease condition

$$d^{\mathrm{T}} \nabla f(x^k + \alpha d) \geq c_2 \cdot d^{\mathrm{T}} \nabla f(x^k)$$ curvature condition



can prevent slow progress

保证函数下降充分的快，下降的progress更大(文献证明△g和△x内积在满足wolfe condition时是大于0的，从而保证B是正定的，从而保证d是函数descent方向)

1. 充分下降条件（Armijo condition）
2. 曲率条件（curvature condition）

- **Strong Wolfe condition**

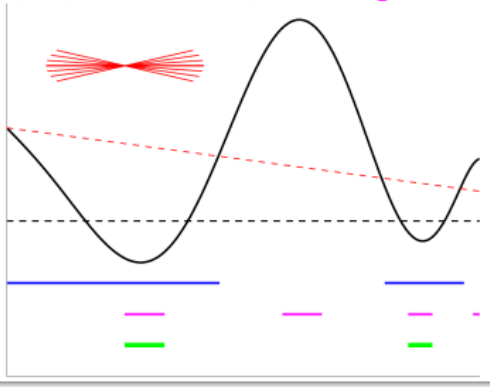
  直观理解：希望下次的搜索到山谷里，把斜率压得更平（抑制震荡效果）

$\Delta g^T \Delta x > 0$? Ensure PD of approximation: Wolfe conditions!

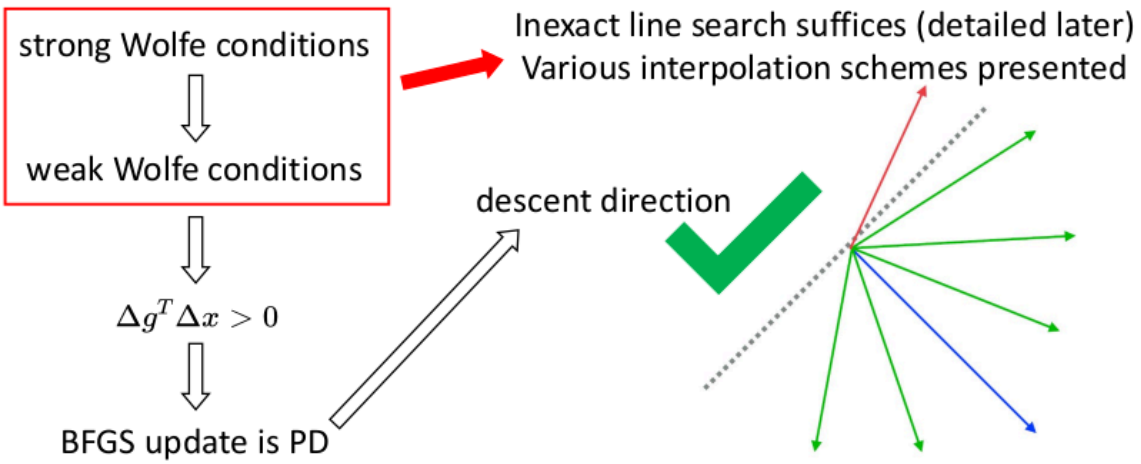
2. strong Wolfe conditions $0 < c_1 < c_2 < 1$ typically $c_1 = 10^{-4}$, $c_2 = 0.9$

$$f(x^k) - f(x^k + \alpha d) \geq -c_1 \cdot \alpha d^{\mathrm{T}} \nabla f(x^k)$$ sufficient decrease condition

$$\left|d^{\mathrm{T}} \nabla f(x^k + \alpha d)\right| \leq \left|c_2 \cdot d^{\mathrm{T}} \nabla f(x^k)\right|$$ strong curvature condition



suppress oscillation



- **cautious-BFGS**

Wolfe conditions cannot guarantee the convergence of BFGS

The cautious update (Li and Fukushima 2001) with mild conditions is needed.

$$B^{k+1} = \begin{cases} \left(I - \frac{\Delta x \Delta g^T}{\Delta g^T \Delta x}\right) B^k \left(I - \frac{\Delta g \Delta x^T}{\Delta g^T \Delta x}\right) + \frac{\Delta x \Delta x^T}{\Delta g^T \Delta x} & \text{if } \Delta g^T \Delta x > \epsilon ||g_k|| \Delta x^T \Delta x, \ \epsilon = 10^{-6} \\ B^k & \text{otherwise} \end{cases}$$

The cautious update guarantees zero grad to be an accumulation point if

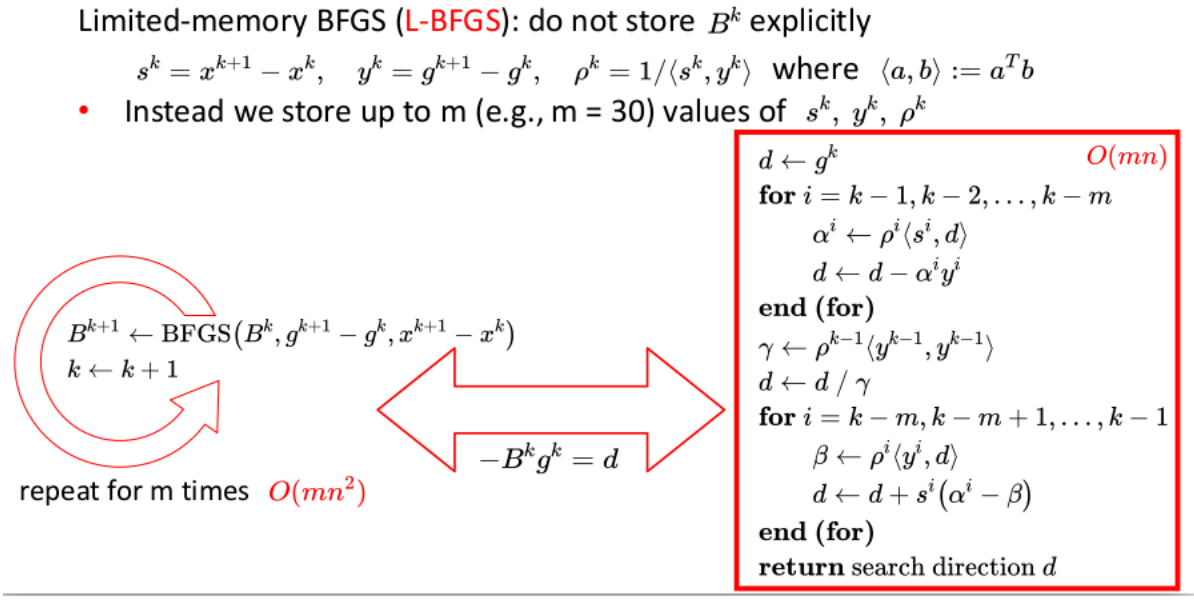
- The func has bounded sub-level sets
- The func has Lipschitz continuous grad

伪代码:

BFGS method for possibly nonconvex functions

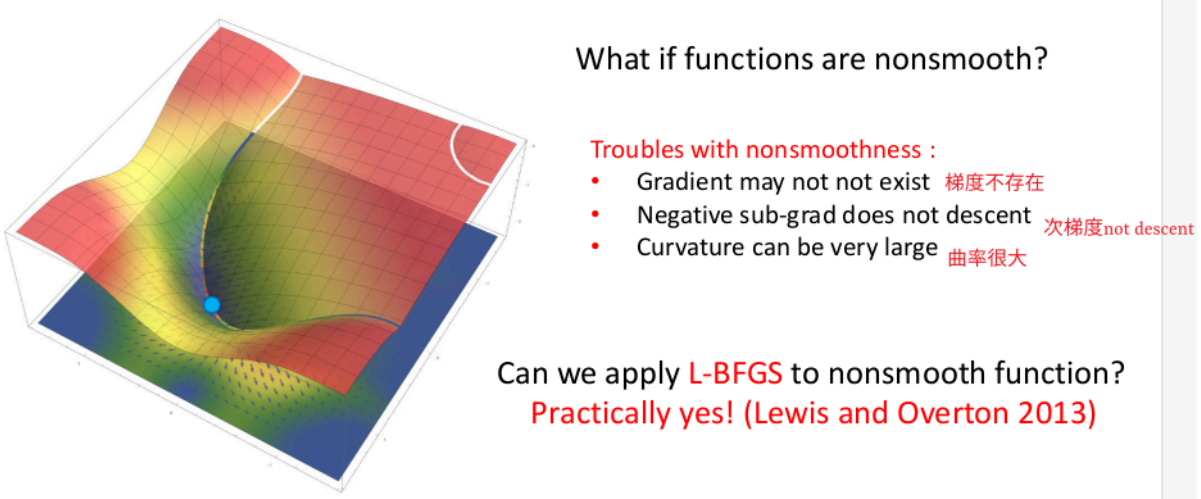
initialize $x^0,\ g^0 \leftarrow \nabla f(x^0),\ B^0 \leftarrow I,\ k \leftarrow 0$
while $\|g^k\| > \delta$ do
  $d \leftarrow -B^k g^k$
  $t \leftarrow$ inexact line search (Wolfe)
  $x^{k+1} \leftarrow x^k + td$
  $g^{k+1} \leftarrow \nabla f(x^{k+1})$
  $B^{k+1} \leftarrow \text{Cautious-BFGS}(B^k, g^{k+1} - g^k, x^{k+1} - x^k)$
  $k \leftarrow k + 1$
end while
return

4. L-BFGS算法（非凸光滑函数）



- 维护一个时间滑动窗口
- 滑动窗口入队时可以检验是否满足梯度收敛于0的条件（保证descent方向和收敛性）

5. L-BFGS算法（非凸非光滑函数）



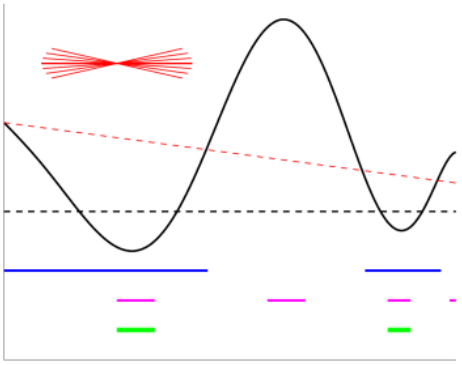
- strong wolfe conditions（不能用）

Key issues when applying L-BFGS to nonsmooth functions

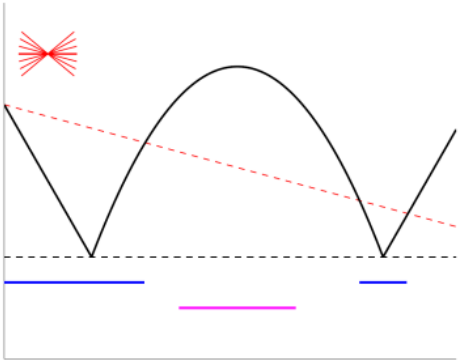
Recall: strong Wolfe conditions

$$f(x^k) - f(x^k + \alpha d) \geq -c_1 \cdot \alpha d^{\mathrm{T}} \nabla f(x^k)$$ sufficient decrease condition

$$\left| d^{\mathrm{T}} \nabla f(x^k + \alpha d) \right| \leq \left| c_2 \cdot d^{\mathrm{T}} \nabla f(x^k) \right|$$ strong curvature condition



(a) *Smooth* $\phi(\alpha)$.



(b) *nonsmooth* $\phi(\alpha)$. conditions fail!

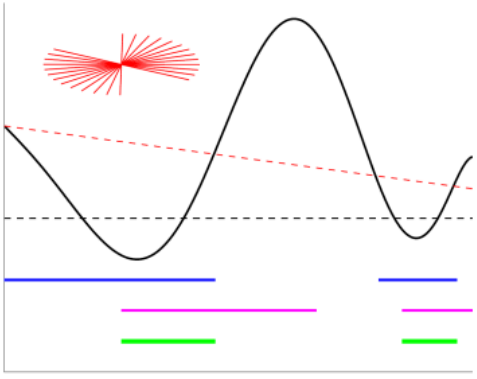
- weak wolfe conditions(适用！！)

Key issues when applying L-BFGS to nonsmooth functions

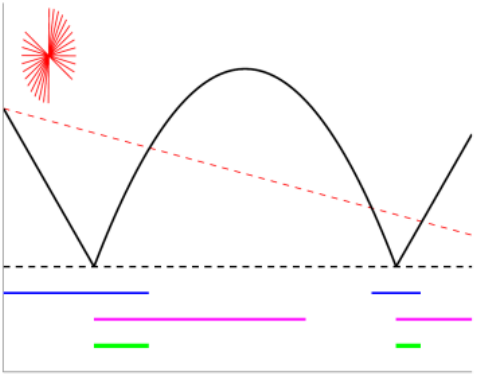
Recall: weak Wolfe conditions

$$f(x^k) - f(x^k + \alpha d) \geq -c_1 \cdot \alpha d^{\mathrm{T}} \nabla f(x^k)$$ sufficient decrease condition

$$d^{\mathrm{T}} \nabla f(x^k + \alpha d) \geq c_2 \cdot d^{\mathrm{T}} \nabla f(x^k)$$ curvature condition



(a) *Smooth* $\phi(\alpha)$.



(b) *nonsmooth* $\phi(\alpha)$. conditions work!

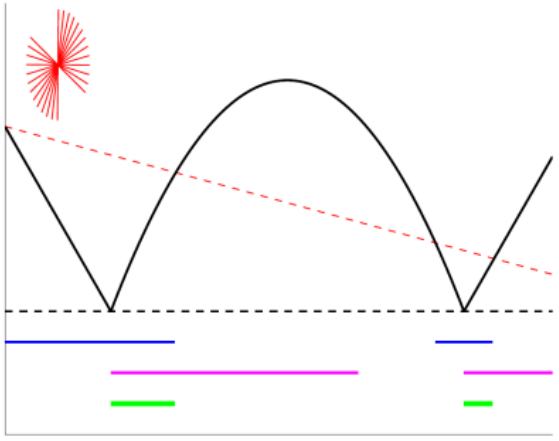
- Lewis & Overton Line search

Quasi-Newton Methods

weak Wolfe conditions should be used for nonsmooth functions

$$S(\alpha): f(x^k) - f(x^k + \alpha d) \geq -c_1 \cdot \alpha d^{\mathrm{T}} \nabla f(x^k)$$

$$C(\alpha): d^{\mathrm{T}} \nabla f(x^k + \alpha d) \geq c_2 \cdot d^{\mathrm{T}} \nabla f(x^k)$$



Lewis & Overton line search:

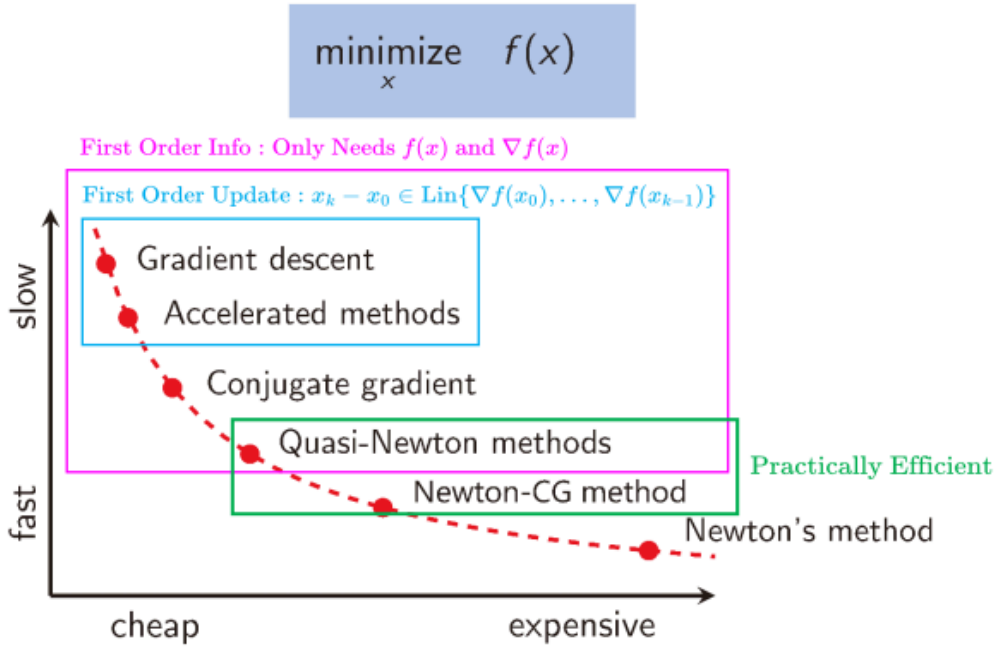
- weak Wolfe conditions
- no interpolation used

$l \leftarrow 0$
$u \leftarrow +\infty$
$\alpha \leftarrow 1$
repeat
  if $S(\alpha)$ fails
    $u \leftarrow \alpha$
  else if $C(\alpha)$ fails
    $l \leftarrow \alpha$
  else
    return α
  if $u < +\infty$
    $\alpha \leftarrow (l + u)/2$
  else
    $\alpha \leftarrow 2l$
end (repeat)

6. Newton-CG方法(LCG)

- **other method to avoid hessian calculate**

Is quasi-Newton class the only way to avoid Hessian?



Complexity of function/gradient/Hessian

$$f:\mathbb{R}^n\to\mathbb{R} \quad = \quad \nabla f=\begin{pmatrix}\frac{\partial f}{\partial x_1}\\ \frac{\partial f}{\partial x_2}\\ \vdots\\ \frac{\partial f}{\partial x_n}\end{pmatrix} \quad < \quad \nabla^2 f=\begin{pmatrix}\frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1\partial x_n}\\ \frac{\partial^2 f}{\partial x_2\partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & & \frac{\partial^2 f}{\partial x_2\partial x_n}\\ \vdots & & \ddots & \vdots\\ \frac{\partial^2 f}{\partial x_n\partial x_1} & \frac{\partial^2 f}{\partial x_n\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2}\end{pmatrix}$$

Function Gradient Hessian

About computational complexity :

- Computing gradient is at least $O(n)$
- Computing gradient is as easy as computing the function
- Computing Hessian is at least $O(n^2)$ whose inversion is about $O(n^3)$

不计算Hessian逆，通过O（N）计算Hessian向量

推导：$h(\alpha)=\nabla f(x+\alpha\xi)$在$\alpha=0$处泰勒展开

Complexity of function/gradient/Hessian-vec product

$$f:\mathbb{R}^n\to\mathbb{R} \quad = \quad \nabla f=\begin{pmatrix}\frac{\partial f}{\partial x_1}\\ \frac{\partial f}{\partial x_2}\\ \vdots\\ \frac{\partial f}{\partial x_n}\end{pmatrix} \quad = \quad (\nabla^2 f)\xi=\begin{pmatrix}\frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1\partial x_n}\\ \frac{\partial^2 f}{\partial x_2\partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & & \frac{\partial^2 f}{\partial x_2\partial x_n}\\ \vdots & & \ddots & \vdots\\ \frac{\partial^2 f}{\partial x_n\partial x_1} & \frac{\partial^2 f}{\partial x_n\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2}\end{pmatrix}\begin{pmatrix}\xi_1\\ \xi_2\\ \vdots\\ \xi_n\end{pmatrix}$$

Function Gradient Hessian-vec product

Even if exact value is not available, just compute $(\nabla^2 f)\xi\approx\frac{\nabla f(x+\delta\xi)-\nabla f(x)}{\delta}$

However, we need inverse-Hessian-vec product.
What can be exploited via Hessian-vec product?

关键：已知$\gamma(x)=Ax$，如何求x

The most prominent large-scale linear solver:
(Linear) Conjugate Gradient Method

$$Ax = b$$

$A \in \mathbb{R}^{n \times n}$: a known PD mat
$x \in \mathbb{R}^{n}$: an unknown vec
$b \in \mathbb{R}^{n}$: a known vec

Solve $Ax = b$ via the interface $\gamma(x) := Ax$

转换：二次函数求极小值的问题

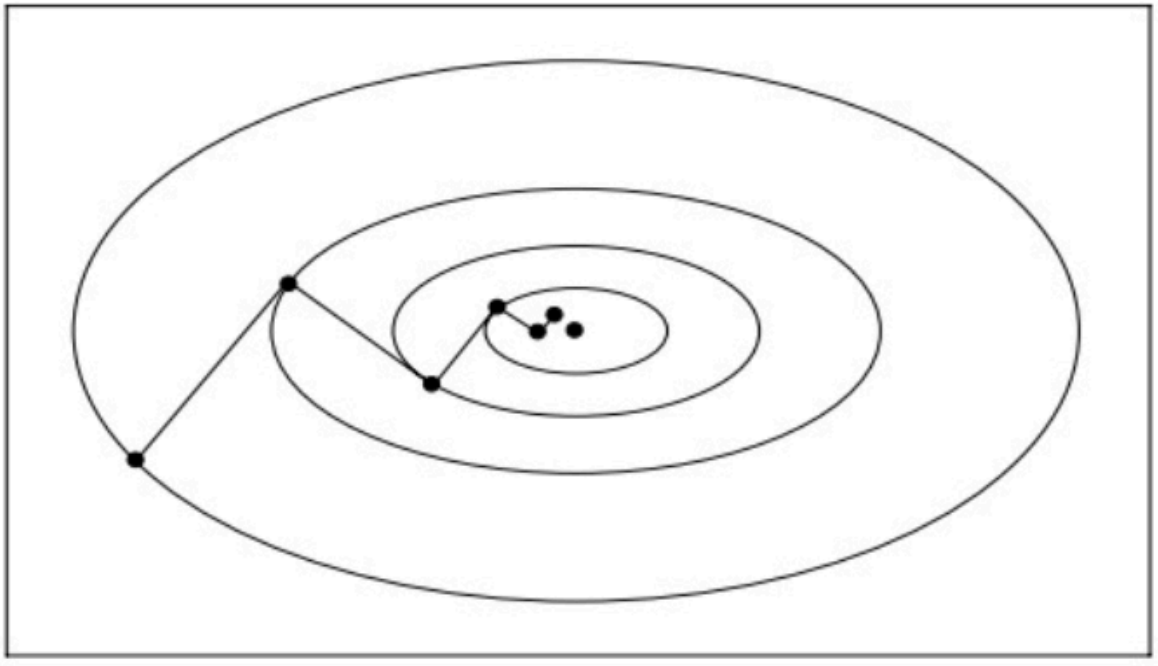
(Linear) Conjugate Gradient Method

$Ax = b$ $\Longleftrightarrow$ $\nabla f(x) = Ax - b = 0$ $\arg\min_x f(x) = \frac{1}{2}x^T Ax - b^T x$

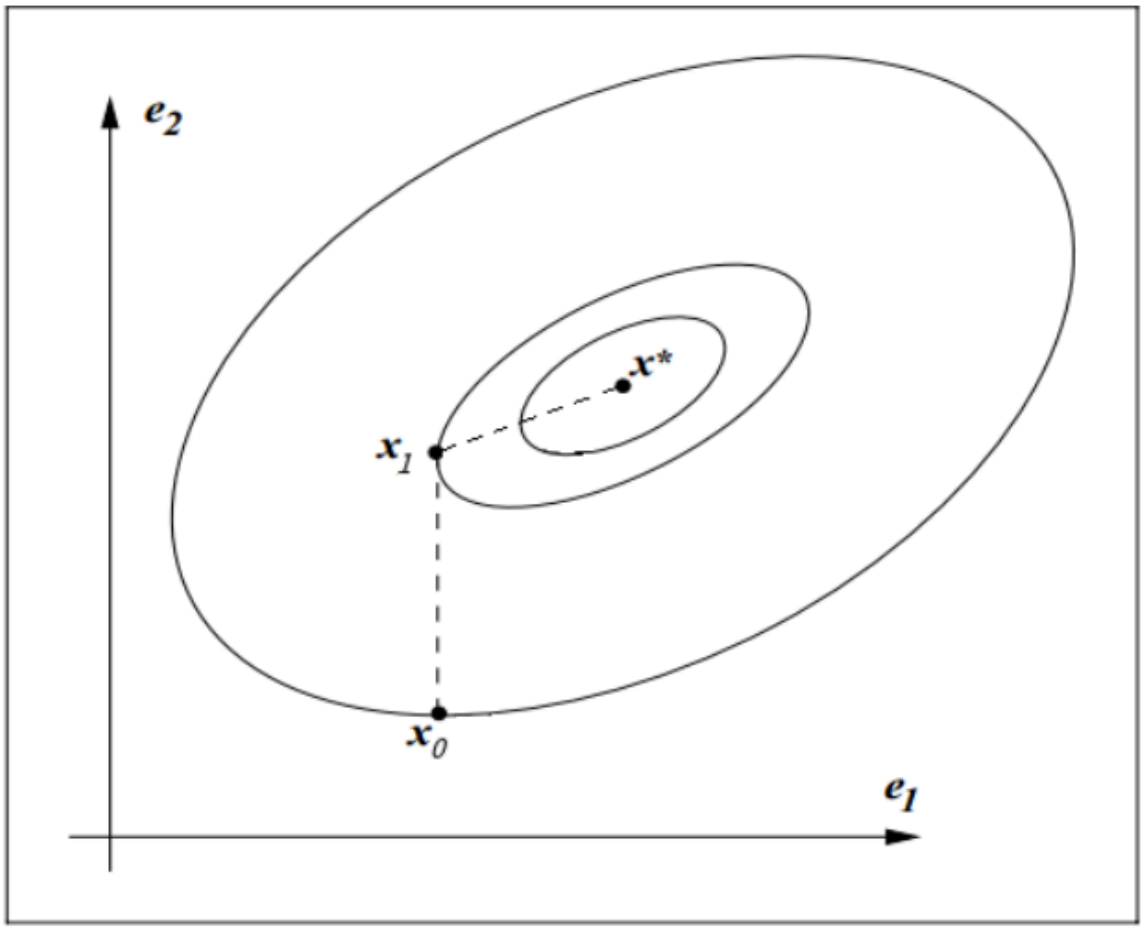
考虑使用：steepest gradient descent & newton's method，牛顿方法不可避免求Hessian逆，所以考虑最速梯度下降法

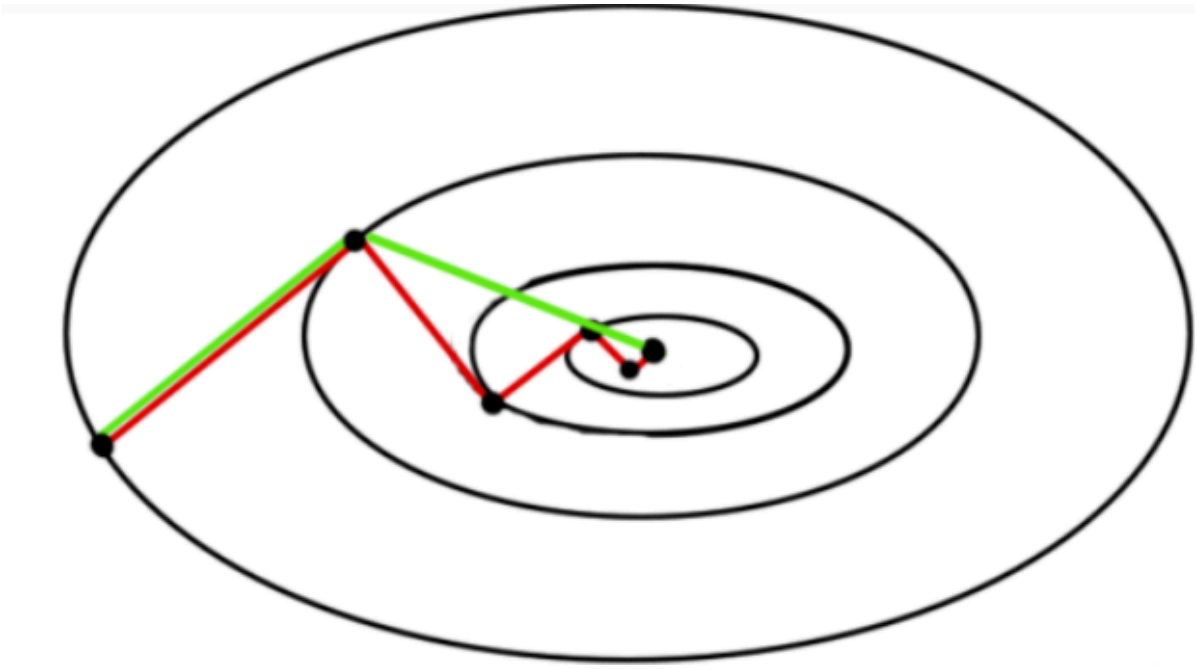
问题：如何使最速梯度下降快速的迭代?

exact line search（可能会来回呈之字型震荡）



期望效果：二维函数，两步收敛（关键是第二步迭代的方向直接指向minima，由此推两个方向的关系），可以推出**两个向量x0x1和x1x*关于A正交**，前提是x0处采用exact line search，x1是x0 exact line search搜索到的结果，推广到N维空间





关键：在n维空间找n个关于A正交的向量作为迭代方向

(Linear) Conjugate Gradient Method

If the linearly independent vector is chosen as

$$\mathrm{v}^k = b - Ax^k$$

The history-dependent coefficients in conjugate G-S process becomes

$$\mathrm{proj}_{\mathrm{u}^j}(\mathrm{v}^k) = 0,\ \forall j \le k-2$$

Thus we get a short recurrence

$$\mathrm{u}^k = \mathrm{v}^k - \sum_{j=1}^{k-1} \mathrm{proj}_{\mathrm{u}^j}(\mathrm{v}^k) = \mathrm{v}^k + \frac{\|\mathrm{v}^k\|^2}{\|\mathrm{v}^{k-1}\|^2}\mathrm{u}^{k-1}$$

Given n mutually conjugate directions

$$u^1, u^2, \ldots, u^n$$

Successively conduct closed-form exact line search

$$\alpha \leftarrow \frac{b^T u^k - (x^k)^T Au^k}{(u^k)^T Au^k} \rightarrow \text{interface } \gamma(u^k)$$

$$x^{k+1} \leftarrow x^k + \alpha u^k$$

Obtain an exact solution in n steps

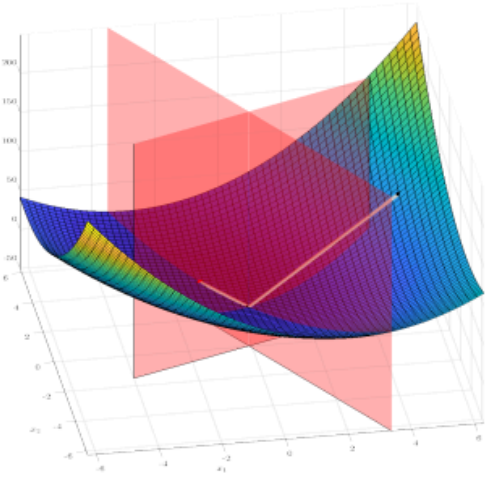
迭代方法（CG）解线性方程组伪代码：

(Linear) Conjugate Gradient Method

$x^1 \leftarrow x_{ini}$
$\mathrm{v}^1 \leftarrow b - Ax^1$
$\mathrm{u}^1 \leftarrow \mathrm{v}^1$
$k \leftarrow 1$
while $\mathrm{v}^k \neq 0$
  $\alpha \leftarrow \|\mathrm{v}^k\|^2/(\mathrm{u}^k)^{\mathrm{T}}A\mathrm{u}^k$
  $x^{k+1} \leftarrow x^k + \alpha \mathrm{u}^k$
  $\mathrm{v}^{k+1} \leftarrow \mathrm{v}^k - \alpha A\mathrm{u}^k$
  $\beta \leftarrow \|\mathrm{v}^{k+1}\|^2/\|\mathrm{v}^k\|^2$
  $\mathrm{u}^{k+1} \leftarrow \mathrm{v}^{k+1} + \beta \mathrm{u}^k$
  $k \leftarrow k+1$
end (while)
return x^k

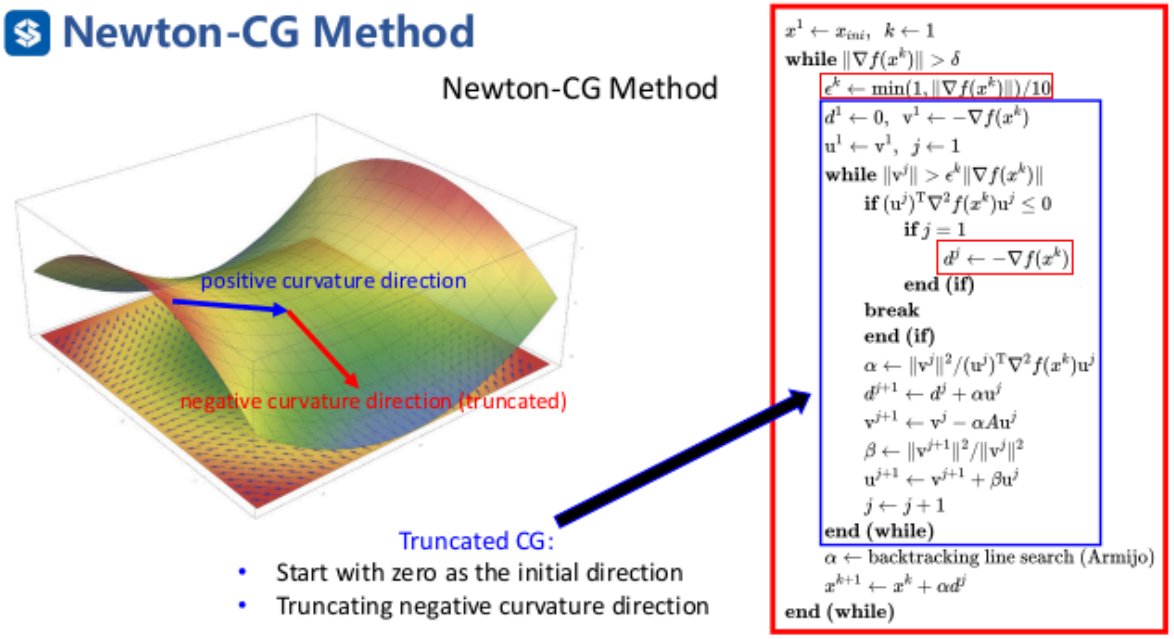
float-point precision: $\|\mathrm{v}^k\| \ge \epsilon$

interface $\gamma(u^k)$



If exact arithmetic is used, CG always finds an exact solution in n steps

牛顿-CG方法用来每次更新x时确定最佳的更新方向（求解Hd=-g方程组）：



1. 注意近似算$\nabla^2 f(x_k)u^j$很容易出现数值不稳定情况，所以这里只在每次CG迭代外部计算一次$\nabla^2 f(x_k)$，CG迭代内部只需要进行矩阵乘法
2. 整体过程参考最速梯度下降和牛顿方法，区别是不求Hessian逆而通过迭代的方法求解线性方程组，实测效率提升很大，甚至比BFGS快