Lec8p1, ORF363/COS323

PRINCETON UNIVERSITY
ORFE
Instructor:
Amir Ali Ahmadi
Fall 2014

TAs: Y. Chen,
G. Hall,
J. Ye

This lecture:

- Gradient descent methods
- Choosing the descent direction
- Choosing the step size
- Convergence
- Convergence rate

Let's recall our unconstrained optimization problem:

$$\min_x f(x), \qquad (f: \mathbb{R}^n \to \mathbb{R})$$

Where we stand so far:

- Learned about some structural properties of local optimal solutions (first and second order conditions for optimality).
- Learned that for convex problems, local optima are automatically global.
- But how to find a local optimum?
- How to even find a stationary point (i.e., a point where the gradient vanishes)? Recall that this would suffice for global optimality if f is convex.
- We now begin to see some algorithms for this purpose, starting with gradient descent algorithms.
- These will be iterative algorithms: start at a point, jump to a new point that hopefully has a lower objective value and continue.
- Our presentation uses references [Bert03], [Bert09], [Tit13], [CZ13].

General form of the iterations:

$$x_{k+1} = x_k + \alpha_k d_k$$

$k \in \mathbb{Z}_+$: index of time (iteration number)

$x_k \in \mathbb{R}^n$: Current point $\qquad$ $d_k \in \mathbb{R}^n$: Direction to move along at iteration k

$x_{k+1} \in \mathbb{R}^n$: Next point $\qquad$ $\alpha_k \in \mathbb{R}_+$: Step size at iteration k

Goal is to make the sequence $\{f(x_k)\}$ decrease as much as possible.

- How to choose d_k? How to choose α_k?

Gradient methods

- Throughout this lecture, we assume that f is at least C^1 (continuously differentiable).
- Gradient methods: the direction d_k to move along at step k is chosen "based on information from" $\nabla f(x_k)$.
- Why is $\nabla f(x_k)$ a natural vector to look at? Lemmas 1 and 2 below (proved on the next page) provide two reasons.

Lemma 1: Consider yourself sitting at a point $x \in \mathbb{R}^n$ and looking (locally) at the value of the function f in all directions around you. The direction with the maximum rate of decrease is along $-\nabla f(x)$.

Remark: When we speak of direction, the magnitude of the vector does not matter; e.g., $\nabla f(x), 5\nabla f(x), \frac{\nabla f(x)}{20}, \frac{\nabla f(x)}{||\nabla f(x)||}$ all are in the same direction.

Definition: For a given point $x \in \mathbb{R}^n$, a direction $d \in \mathbb{R}^n$ is called a descent direction, if there exists $\bar{\alpha} > 0$ ($\alpha \in \mathbb{R}$) such that

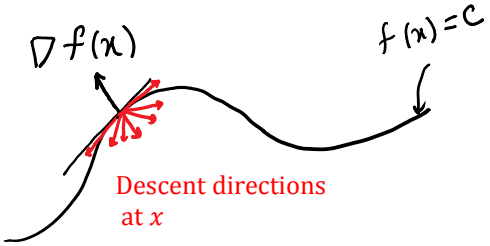
$$f(x + \alpha d) < f(x), \qquad \forall \alpha \in (0, \bar{\alpha}).$$

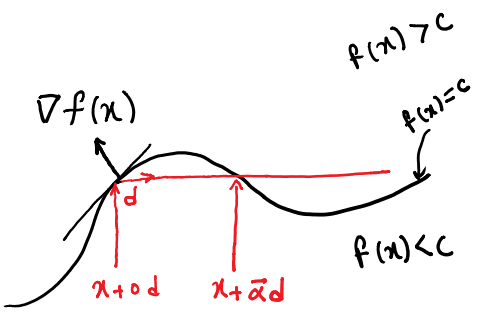
Interpretation: There is a small enough (but nonzero) amount that you can move in direction d and be guaranteed to decrease the function value.

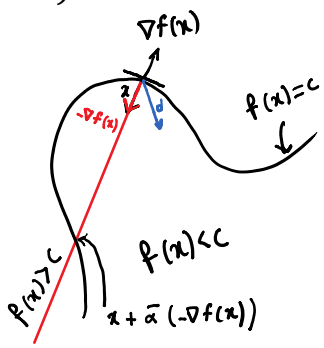
Lemma 2: Consider a point $x \in \mathbb{R}^n$. Any direction d satisfying

$$\langle d, \nabla f(x) \rangle < 0$$

is a descent direction. (In particular, $-\nabla f(x)$ is a descent direction.)



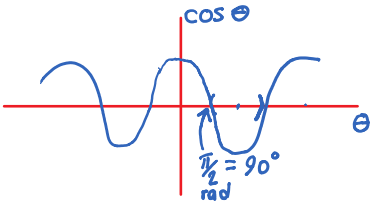




- Red: direction of steepest descent
- Descent direction d would have been better

Remark: The condition $\langle d, \nabla f(x) \rangle < 0$ in Lemma 2 geometrically means that the vectors d and $\nabla f(x)$ make an angle of more than 90 degrees (on the plane that contains them).

Why? $\langle d, \nabla f(x) \rangle = ||d|| \cdot ||\nabla f(x)|| \cos(\theta)$



Proof of Lemma 1.

Consider a point x, a direction d, and the univariate function

$$g(\alpha) = f\left(x + \alpha \frac{d}{||d||}\right).$$

The rate of change of f at x in direction d is given by $g'(0)$, which by the chain rule equals

$$\frac{1}{||d||}\langle \nabla f(x), d \rangle.$$

By the Cauchy-Schwarz inequality (see, e.g., Theorem 2.3 of [CZ13] for a proof), we have:

$$-\frac{1}{||d||} \cdot ||\nabla f(x)|| \cdot ||d|| \leq \frac{1}{||d||}\langle \nabla f(x), d \rangle \leq \frac{1}{||d||} \cdot ||\nabla f(x)|| \cdot ||d||,$$

which after simplifying gives

$$-||\nabla f(x)|| \leq \frac{1}{||d||}\langle \nabla f(x), d \rangle \leq ||\nabla f(x)||.$$

So the rate of change in any direction cannot be larger than $||\nabla f(x)||$, or smaller than $-||\nabla f(x)||$. However, if we take $d = \nabla f(x)$, the right inequality is achieved:

$$\frac{1}{||\nabla f(x)||}\langle \nabla f(x), \nabla f(x) \rangle = \frac{1}{||\nabla f(x)||} \cdot ||\nabla f(x)||^2 = ||\nabla f(x)||.$$

Similarly, if we take $d = -\nabla f(x)$, then the left inequality is achieved.

Proof of Lemma 2.

$g(\alpha) := f(x + \alpha d)$

$g(\alpha) = g(0) + g'(0)\alpha + o(\alpha)$

By Taylor's theorem, we have

$$f(x + \alpha d) = f(x) + \alpha \nabla f(x)^T d + o(\alpha).$$

Since $\lim_{\alpha \to 0} \frac{|o(\alpha)|}{\alpha} = 0$, there exists $\bar{\alpha} > 0$ such that

$$\frac{|o(\alpha)|}{\alpha} < |\nabla f(x)^T d|, \quad \forall \alpha \in (0, \bar{\alpha}).$$

This, together with our assumption that $\nabla f(x)^T d < 0$, implies that $\forall \alpha \in (0, \bar{\alpha})$ we must have:

$$f(x + \alpha d) - f(x) < 0.$$

Hence, d is a descent direction.

Lemma 3: Consider any positive definite matrix B. For any point x, with $\nabla f(x) \neq 0$, the direction $-B\nabla f(x)$ is a descent direction.

Proof. We have $\langle -B\nabla f(x), \nabla f(x)\rangle = -\nabla f(x)^T B \nabla f(x) < 0,$ by the assumption that B is positive definite.

This suggests a general paradigm for our descent algorithms:

$$x_{k+1} = x_k - \alpha_k B_k \nabla f(x_k), \qquad (\text{with } B_k \succ 0, \forall k).$$

Common choices of descent direction:

- **Steepest Descent:** $B_k = I, \forall k.$

  Simplest descent direction but not always the fastest.

- **Newton Direction:** $B_k = (\nabla^2 f(x_k))^{-1}$ (assuming Hessian positive definite).

  More expensive, but can have much faster convergence.

- **Diagonally Scaled Steepest Descent:** $B_k = diag(d_{1,k}, \dots, d_{n,k}), d_{i,k} > 0.$

  For example, can take $d_{i,k} \approx \left(\frac{\partial^2 f(x_k)}{(\partial x_i)^2}\right)^{-1}$; i.e., diagonally approximate Newton.

- **Modified Newton Direction:** $B_k = \left(\nabla^2 f(x_0)\right)^{-1}, \forall k$

  Compute Newton direction only at the beginning, or once every M steps.

- **Quasi-Newton Directions:** Chap. 11 of [CZ13].

Tradeoffs generally problem dependent.

Common choices of the step size α_k.

Back to the general form of our iterative algorithm:

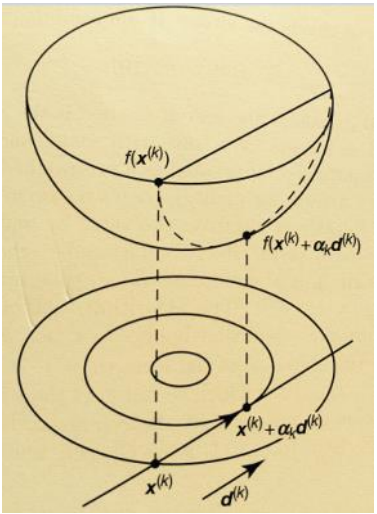
$$x_{k+1} = x_k + \alpha_k d_k$$

- **Constant step size:** $\alpha_k = s, \forall k \quad (s > 0)$
  - Simplest rule to implement, but may not converge if s too large; may be too slow if s too small.
- **Diminishing step size:** $\alpha_k \to 0,\ \sum_{k=1}^{\infty} \alpha_k = \infty.$ (e.g., $\alpha_k = \frac{1}{k}$)
  - Descent not guaranteed at each step; only later when α_k becomes small.
  - $\sum_{k=1}^{\infty} \alpha_k = \infty$ imposed to guarantee progress does not become too slow.
  - Good theoretical guarantees, but unless the right sequence is chosen, can also be a slow method.
- **Minimization rule (exact line search):** $\alpha_k = argmin_{\alpha \geq 0} f(x_k + \alpha_k d_k)$
  - A minimization problem itself, but an easier one (one dimensional).
  - Can use the line search methods that we learned in the previous lecture.
  - No need to be very precise at each step.
  - If f convex, the one dimensional minimization problem also convex (why?).
- **Limited minimization:** $\alpha_k = argmin_{\alpha \in [0,s]} f(x_k + \alpha_k d_k)$
  - Previous comments apply.
  - Tries not to step too far.
- **Successive step size reduction:** well-known examples are Armijo rule and Goldstein rule (see, e.g., Section 7.8 of [CZ13]). We will cover Armijo in the next lecture.
  - Try to ensure enough decrease in line search without spending time to solve it to optimality.

Tradeoffs generally problem dependent.

An illustration of a single step of the minimization rule (aka exact line search) for choosing the step size.

Image credit: [CZ13]



Stopping criteria.

- Once we have a rule for choosing the search direction and the step size, we are good to go for running the algorithm.
- Typically the initial point x_0 is picked randomly, or if we have a guess for the location of local minima, we pick x_0 close to them.
- But when to stop the algorithm?
- Some common choices (ϵ>0 is a small prescribed threshold):
  - $\left|\left|\nabla f(x)\right|\right| < \epsilon$
    - Note: if we have $\nabla f(x_k) = 0$, our iterates stop moving. We have found a point satisfying the first order necessary condition for optimality. This is what we are aiming for.
  - $|f(x_{k+1}) - f(x_k)| < \epsilon$
    - Improvements in function value are saturating.
  - $\left|\left|x_{k+1} - x_k\right|\right| < \epsilon$
    - Movement between iterates has become small.
  - $\dfrac{|f(x_{k+1}) - f(x_k)|}{\max\{1, |f(x_k)|\}} < \epsilon$
    - A "relative" measure - removes dependence on the scale of f.
    - The max is taken to avoid dividing by small numbers.
  - $\dfrac{\left|\left|x_{k+1} - x_k\right|\right|}{\max\{1, \left|\left|x_k\right|\right|\}} < \epsilon$
    - Same comments apply.

Steepest descent with exact line search for quadratic functions.

Goal: min. $f(x) = \frac{1}{2} x^T Q x - b^T x$, assume $Q \succ 0$.

- If Q has a negative eigenvalue, $f^* = -\infty$ (why?)
- If $Q \succ 0 \Rightarrow$ unique global solution (why?)

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

Goal: Choose $\alpha_k \geq 0$ to minimize $f(x_{k+1})$.

Let $g(\alpha) := f(x_k - \alpha \nabla f(x_k))$

$= \frac{1}{2} (x_k - \alpha \nabla f(x_k))^T Q (x_k - \alpha \nabla f(x_k)) - b^T (x_k - \alpha \nabla f(x_k))$

- $g(\alpha)$ is quadratic and convex (why?)

$g(\alpha) = a\alpha^2 + d\alpha + c \longrightarrow \arg\min g(\alpha) = -d/2a$

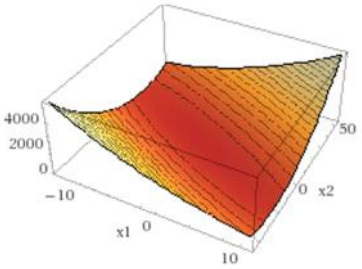
where $a = \frac{1}{2} \nabla f^T(x_k) Q \nabla f(x_k)$, $d = (b^T - x_k^T Q) \nabla f(x_k) = -\nabla f^T(x_k) \nabla f(x_k)$

$\Rightarrow \alpha_k = \dfrac{\nabla f^T(x_k) \nabla f(x_k)}{\nabla f^T(x_k) Q \nabla f(x_k)}$

Overall Algorithm:

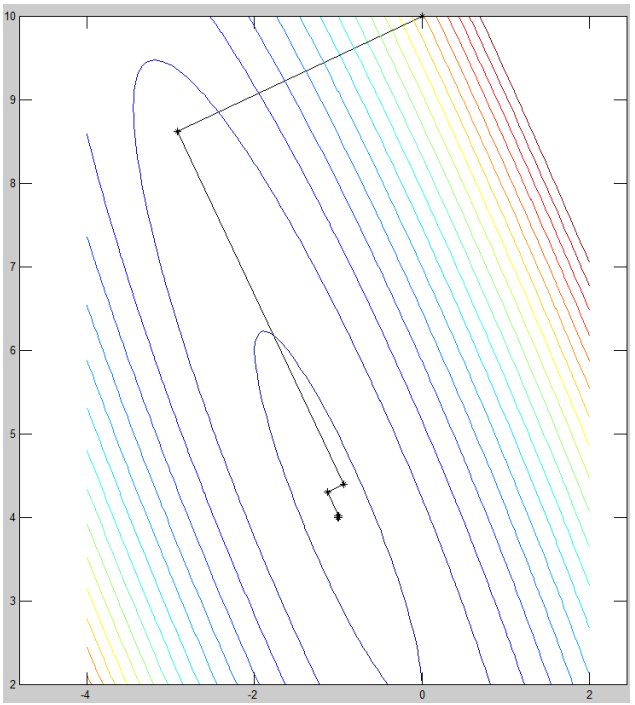
$$x_{k+1} = x_k - \left(\frac{\nabla f^T(x_k) \nabla f(x_k)}{\nabla f^T(x_k) Q \nabla f(x_k)} \right) \nabla f(x_k)$$

where $\nabla f(x_k) = Q x_k - b$



Example.
min. $f(x) = 5x_1^2 + x_2^2 + 4x_1x_2 - 6x_1 - 4x_2 + 15$

Any stationary point must be the unique global minimizer (why?)



Let's try the steepest descent method:
$d_k = -\nabla f(x_k)$
α_k: get from exact line search

$x_0 = (0, 10)^T$
Stopping criterion: $||\nabla f(x)|| < 10^{-6}$

Output:

K	x_k		$\nabla f(x_k)$		$\|\nabla f(x_k)\|$	α_k	$f(x_k)$
1.000000000000000	0	10.000000000000000	34.000000000000000	16.000000000000000	37.576588456111871	0.085971748660497	75.000000000000000
2.000000000000000	-2.923039454456893	8.624452021432051	-0.732586458840725	1.556746225036530	1.720506242023632	2.715384615384604	14.303945445689237
3.000000000000000	-0.933785454681702	4.397287271909798	2.251294540822171	1.059432725092788	2.488116719234297	0.085971748660497	10.284983790761091
4.000000000000000	-1.127333183106014	4.306205987945416	-0.048507879278480	0.103079243466774	0.113922538532891	2.715384615384754	10.018870072128331
5.000000000000000	-0.995615633988288	4.026306196070238	0.149068444398068	0.070149856187323	0.164749517262910	0.085971748660497	10.001249473246101
6.000000000000000	-1.008431308823290	4.020275290265531	-0.003211927170778	0.006825345237901	0.007543329090456	2.715384615384132	10.000082733302879
7.000000000000000	-0.999709691198026	4.001741852811849	0.009870499267134	0.004644940831593	0.010908814376984	0.085971748660486	10.000005478148033
8.000000000000000	-1.000558275280174	4.001342519126133	-0.000212676297206	0.000451937131571	0.000499478105912	2.715384615432194	10.000000362733084
9.000000000000000	-0.999980777334673	4.000115335991923	0.000653570620962	0.000307562645155	0.000722322183848	0.085971748660553	10.000000024018206
10.000000000000000	-1.000036965943830	4.000088894293497	-0.000014082264316	0.000029924811672	0.000033072715672	2.715384615356879	10.000000001590355
11.000000000000000	-0.999998727179956	4.000007636920265	0.000043275881499	0.000020365120705	0.000047828234975	0.085971748657113	10.000000000105302
12.000000000000000	-1.000002447683163	4.000005886095226	-0.000000932450728	0.000001981457800	0.000002189894831	-0.692156863782812	10.000000000006972
13.000000000000000	-1.000003093085335	4.000007257574842	-0.000001900553979	0.000002142808345	0.000002864215954	0.644738427673033	10.000000000010715
14.000000000000000	-1.000001867725151	4.000005876023959	0.000004826844329	0.000004281147315	0.000006451871706	0.095703090047471	10.000000000008070
15.000000000000000	-1.000002329669068	4.000005466304932	-0.000001431470952	0.000001613933592	0.000002157287817	0.644738430768048	10.000000000006079
16.000000000000000	-1.000001406744733	4.000004425739920	0.000003635512357	0.000003224500911	0.000004859460487	0.095703089988433	10.000000000004576
17.000000000000000	-1.000001754674499	4.000004117145219	-0.000001078164111	0.000001215592444	0.000001624839327	0.644738428822084	10.000000000003446
18.000000000000000	-1.000001059540664	4.000003333406057	0.000002738217589	0.000002428649458	0.000003660078381	0.095703090024235	10.000000000002597
19.000000000000000	-1.000001321596548	4.000003100976799	-0.000000812058286	0.000000915567405	0.000001223806493	-0.251801315258460	10.000000000001959
20.000000000000000	-1.000001526073893	4.000003331517876	-0.000001934667424	0.000000558740181	0.000002013734995	0.140893244247662	10.000000000002405
21.000000000000000	-1.000001253492323	4.000003252795159	0.000000476257405	0.000001491621026	0.000001565807907	0.241956665294677	10.000000000002125
22.000000000000000	-1.000001368725976	4.000002891887509	-0.000002119709727	0.000000308871113	0.000002142094930	0.115598723360049	10.000000000001897
23.000000000000000	-1.000001123690238	4.000002856182403	0.000000187827231	0.000001217603853	0.000001232005768	0.511898030938667	10.000000000001631
24.000000000000000	-1.000001219838628	4.000002232893388	-0.000003266812726	-0.000000413567736	0.000003292886828	0.092258931075849	10.000000000001528
25.000000000000000	-1.000000918445978	4.000002271048705	-0.000000100264955	0.000000868313499	0.000000874083174	0.000000000100000	10.000000000001036

x^* $\quad$ $< \epsilon$ $\quad$ $f(x^*)$

Theorem. ("the zig-zag theorem") Let $\{x_k\}$ be the sequence generated by the steepest descent algorithm. Then, for all k, $x_{k+1} - x_k$ is orthogonal to $x_{k+2} - x_k$.

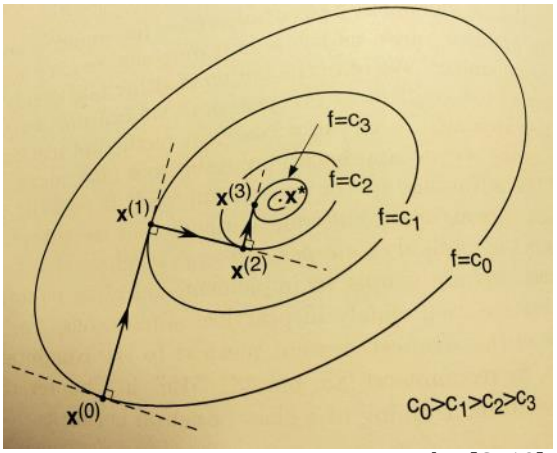


Image credit: [CZ13]

Proof. Our iterations read:

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

$$x_{k+2} = x_{k+1} - \alpha_{k+1} \nabla f(x_{k+1}).$$

Hence,

$$\langle x_{k+1} - x_k, x_{k+2} - x_{k+1} \rangle = \alpha_k \alpha_{k+1} \langle \nabla f(x_k), \nabla f(x_{k+1}) \rangle. \quad (1)$$

Recall that α_k minimizes $\varphi_k(\alpha) := f(x_k - \alpha \nabla f(x_k))$.

$$\Rightarrow \frac{d\varphi_k}{d\alpha}(\alpha_k) = 0. \quad (2)$$

But by chain rule:

$$\frac{d\varphi_k}{d\alpha}(\alpha_k) = \langle -\nabla f(x_k), \nabla f(x_k - \alpha_k \nabla f(x_k)) \rangle$$

$$= \langle -\nabla f(x_k), \nabla f(x_{k+1}) \rangle. \quad (3)$$

$$(1)+(2)+(3) \Rightarrow \langle x_{k+1} - x_k, x_{k+2} - x_{k+1} \rangle = 0. \quad \square$$

Convergence

The descent algorithms discussed so far typically come with proofs of convergence to a stationary point. We state a couple such theorems here; the proofs are a bit tedious, and I won't require you to know them. But those interested can look some of these proofs up in [CZ13].

Theorem. (See [Bert03].) Consider the sequence $\{x_k\}$ generated by any descent algorithm with

$$d_k = -B_k \nabla f(x_k),$$

such that eigenvalues of B_k are larger than some $m > 0$, for all k, and the step size is chosen according to the minimization rule, or the limited minimization rule, (or the Armijo rule).
Then, every limit point of $\{x_k\}$ is a stationary point.

Remark1: We say $x \in \mathbb{R}^n$ is a limit point of a sequence $\{x_k\}$, if there exists a subsequence of $\{x_k\}$ that converges to x.

Remark2: A stationary point may not be a local min! It may be a saddle point or even a local max! But there is a result called the ``capture theorem'' (see [Bert03]) which informally states that isolated local minima tend to attract gradient methods.

Question to think about: How would you check if the point that you end up with (assuming it is stationary) is actually a local minimum?

Theorem. (See [CZ13].) Consider a quadratic function $f(x) = \frac{1}{2}x^T Q x + b^T x + c$, with $Q \succ 0$, and let x_* be the minimizer.

- For the steepest descent algorithm with exact line search, we have $x_k \to x_*$ starting from any x_0. (This is called global convergence.)
- For the steepest descent algorithm with a fixed step size, we have global convergence if and only if the step size α satisfies:

  $$0 < \alpha < \frac{2}{\lambda_{max}(Q)},$$

  where $\lambda_{max}(Q)$ denotes the maximum eigenvalue of Q.

Rates of convergence

- Once we know an iterative algorithm converges, the next question is how fast?
- If $|f(x_k) - f(x_*)| \approx \frac{1}{\log(\log(k))}$, then sure, it will go to zero. But to see the error go even below 0.1, you need to write a letter to your grandson (to ask for a letter to his grandson, etc.)
- Let us formalize this.

Definition (see [Tit13]). Let $\{x_k\}$ converge to x_*. We say the convergence is of order $p(\geq 1)$ and with factor γ (> 0), if $\exists k_0$ such that $\forall k \geq k_0$,

$$\big||x_{k+1} - x_*|\big| \leq \gamma \big||x_k - x_*|\big|^p.$$

Make sure the following comments make sense:

- The larger the power p, the faster the convergence.
- For the same p, the smaller γ, the faster the convergence.
- If $\{x_k\}$ converges with order p, it also converges with order p' for any $p' \leq p$.
- If $\{x_k\}$ converges with order p and factor γ, it also converges with order p' and factor γ', for any $\gamma' \geq \gamma$.
- So we typically look for the largest p and the smallest γ for which the inequality holds.

Some more terminology:

- If $p = 1$, and $\gamma < 1$, we say convegence is linear.
- If $p = 1$, and $\gamma = 1$, we say convegence is sublinear.
- If $p > 1$, we say that convergence is superlinear. (This is slightly stronger than the usual definition of superlinear convergence, but we will go with it.)
- If $p = 2$, we say that convergence is quadratic.

- Why called linear convergence?

For k large enough, we have $\big||x_{k+i} - x_*|\big| \leq \gamma^i \big||x_k - x_*|\big|$. So $\log\big||x_{k+i} - x_*|\big| \leq i \log\gamma + \log||x_k - x_*||$. Hence, $-\log\big||x_{k+i} - x_*|\big|$, which is a measure of the number of correct significant digits in x_{k+i}, grows linearly with i.

$-\log_{10} 0.1 = 1$, $-\log_{10} 0.01 = 2$, $-\log_{10} 0.001 = 3$, etc.

Examples:

- $||x_k - x_*|| \approx \frac{1}{k}$ sublinear convergence.
- $||x_k - x_*|| \approx a^k, 0 < a < 1$ linear convergence.
- $||x_k - x_*|| \approx a^{2^k}, 0 < a < 1$ quadratic convergence. (Verify!)

- Quadratic convergence is super fast! Number of correct significant digit doubles in each iteration (why?).

Convergence rate of steepest descent for quadratic functions

Theorem. Consider a quadratic function $f(x) = \frac{1}{2}x^T Qx + b^T x + c$, with $Q \succ 0$. Let m and M respectively denote the smallest and largest eigenvalue of Q. Then the sequence $\{f(x_k)\}$ generated by the steepest descent algorithm with exact line search converges to the unique global minimum of f, where the convergence is linear (order 1), and with factor $\left(\frac{M-m}{M+m}\right)^2$.

- $\kappa := \frac{M}{m}$ is called the condition number of the matrix Q. (Note $\kappa \geq 1$.)
  - Very important quantity in numerical analysis
- Note: $\left(\frac{M-m}{M+m}\right)^2 = \left(\frac{\kappa-1}{\kappa+1}\right)^2$
- We want κ small for fast convergence (close to one).

From [Bert09]:

κ	$\left(\frac{M-m}{M+m}\right)^2$	# of iterations needed to reduce optimality gap by 0.1
1.1	0.0023	1
3	0.25	2
10	0.67	6
100	0.96	58
200	0.98	116
400	0.99	231

Analysis first done by Kantorovich (winner of 1975 Nobel Prize in Economics).

Original CIA file on Kantorovich, seized from the former US Embassy in Tehran. (source: Wiki)

CONFIDENTIAL

USSR

Leonid Vital'yevich KANTOROVICH

Head, Problems Laboratory of Economic-Mathematical Methods and Operations Research, Institute of Management of the National Economy

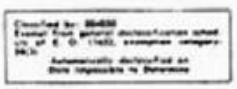
(1975)

An internationally recognized creative genius in the fields of mathematics and the application of electronic computers to economic affairs, Academician Leonid Kantorovich (pronounced kahntuhROHvich) has worked at the Institute of Management of the National Economy since 1971. He has been involved in advanced mathematical research since the age of 15; in 1939 he invented linear programming, one of the most significant contributions to economic management in the twentieth century. Kantorovich has spent most of his adult life battling to win acceptance for his revolutionary concept from Soviet academic and economic bureaucracies; the value of linear programming to Soviet economic practices was not really recognized by his country's authorities until 1965, when Kantorovich was awarded a Lenin Prize for his work. International recognition came in October 1975, when the mathematician was awarded the Nobel Prize for Economics jointly with T. C. Koopmans, a Dutch-born American economist who discovered the same concept independently a few years after Kantorovich.

In addition to his mathematical research, Kantorovich has been directly involved in developing improved designs for high-speed digital computers, an activity apparently motivated by the Soviet Union's need for improved computers in solving large economic planning problems.

The Institute of Management of the National Economy

The Institute of Management of the National Economy was established to train high-level economic and industrial administrators in modern methods of management, production organization and the use of economic-mathematical methods and computers in planning. When the institute opened in early 1971, Premier Aleksey Kosygin and Party Secretary Andrey Kirilenko attended the ceremonies, thus suggesting the importance that the Soviet Government and Party attach to the application of modern management techniques to Soviet industrial administration and economic planning.



CONFIDENTIAL

CR 77-10705

Large κ:

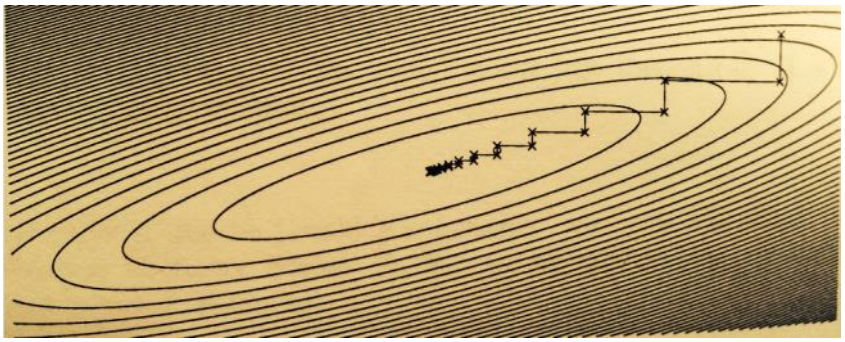
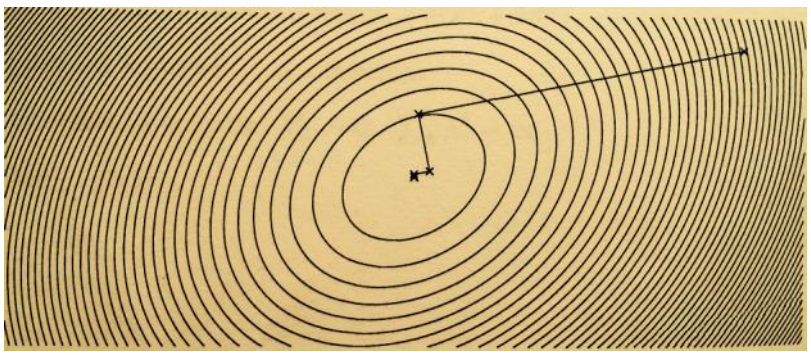


Image credit: [Bert09]

Small κ:



- What if the function f we are minimizing is not quadratic?
  - Denote the optimal solution by x_*
  - Locally the function is well approximated by a quadratic: $\nabla^2 f(x)$ (plus linear and constant terms)
  - Hence $\kappa(\nabla^2 f(x_*))$ (i.e., the condition number of the Hessian at x_*) dictates convergence rate

We will see in future lectures how we can achieve a better than linear rate of convergence by using the Newton method.

Notes:

- The relevant chapter of [CZ13] for this lecture is Chapter 8. You can have a look at some examples that are worked out there in detail, but if you understand the notes well, that should be enough.

References:

- [Bert09] D. Bertsimas. Lecture notes on optimization methods (6.255). MIT OpenCourseWare, 2009.

- [Bert03] D.P. Bertsekas. Nonlinear Programming. Second edition. Athena Scientific, 2003.

- [CZ13] E.K.P. Chong and S.H. Zak. An Introduction to Optimization. Fourth edition. Wiley, 2013.

- [Tit13] A.L. Tits. Lecture notes on optimal control. University of Maryland, 2013.