Optimization Techniques

Numerical methods for unconstrained optimization

Consider the problem

$$\min_{x \in \mathbb{R}^n} f(x)$$

Numerical methods for unconstrained optimization

Consider the problem

$$\min_{x \in \mathbb{R}^n} f(x)$$

Recall that a maximization problem can be reduced to a minimization problem as follows:

$$\min_{x \in \mathbb{R}^n} f(x) = -\max_{x \in \mathbb{R}^n} (-f(x))$$

Numerical methods for unconstrained optimization

Consider the problem

$$\min_{x\in\mathbb{R}^n} f(x)$$

Recall that a maximization problem can be reduced to a minimization problem as follows:

$$\min_{x\in\mathbb{R}^n} f(x) = -\max_{x\in\mathbb{R}^n}(-f(x))$$

In this part of the course we introduce notations

$$\begin{aligned} g(x) &\equiv \nabla f(x) = \left(\frac{\partial f}{\partial x_i}\right)_{i=1}^{n} \\ H(x) &= \left(\frac{\partial^2 f}{\partial x_i \partial x_j}\right)_{i,j=1}^{n} \end{aligned}$$

Numerical methods for unconstrained optimization

Consider the problem

$$\min_{x\in\mathbb{R}^n} f(x)$$

Recall that a maximization problem can be reduced to a minimization problem as follows:

$$\min_{x\in\mathbb{R}^n} f(x) = -\max_{x\in\mathbb{R}^n}(-f(x))$$

In this part of the course we introduce notations

$$\begin{aligned} g(x) &\equiv \nabla f(x) = \left(\frac{\partial f}{\partial x_i}\right)_{i=1}^{n} \\ H(x) &= \left(\frac{\partial^2 f}{\partial x_i \partial x_j}\right)_{i,j=1}^{n} \end{aligned}$$

Also, all vectors $x \in \mathbb{R}^n$ are column vectors, $x = (x_1, x_2, \ldots, x_n)^T$

Optimality conditions for unconstrained minimization

Theorem

Suppose that $f \in C^1$, *and that* x^* *is a local minimizer of* $f(x)$. *Then*

$$g(x^*) = 0$$

Optimality conditions for unconstrained minimization

Theorem

Suppose that $f \in C^1$, *and that* x^* *is a local minimizer of* $f(x)$. *Then*

$$g(x^*) = 0$$

Theorem

Suppose that $f \in C^2$, *and that* x^* *is a local minimizer of* $f(x)$. *Then* $g(x^*) = 0$ *and* $H(x^*)$ *is positive semidefinite, that is*

$$y^T H(x^*) y \geq 0 \quad \forall y \in \mathbb{R}^n$$

Linesearch methods

In practice, usually we are not able to provide or compute an explicit minimizer.

Linesearch methods

In practice, usually we are not able to provide or compute an explicit minimizer.

Instead, we would normally expect to use a suitable iterative process.

Linesearch methods

In practice, usually we are not able to provide or compute an explicit minimizer.

Instead, we would normally expect to use a suitable iterative process.

Definition

An **iteration** is a procedure in which a sequence of points $\{x_k\}_{k=1}^{\infty} \subset \mathbb{R}^n$ is generated, starting from some initial guess $x_0 \in \mathbb{R}^n$, with the overall aim of ensuring that sequence (a subsequence of) $\{x_k\}_{k=1}^{\infty}$ has favourable limiting properties.

Linesearch methods

In practice, usually we are not able to provide or compute an explicit minimizer.

Instead, we would normally expect to use a suitable iterative process.

Definition

An **iteration** is a procedure in which a sequence of points $\{x_k\}_{k=1}^{\infty} \subset \mathbb{R}^n$ is generated, starting from some initial guess $x_0 \in \mathbb{R}^n$, with the overall aim of ensuring that sequence (a subsequence of) $\{x_k\}_{k=1}^{\infty}$ has favourable limiting properties.

These might include that any limit generated satisfies 1-st order or, even better, second-order necessary optimality conditions.

Linesearch methods

In certain situations we will not be able to guarantee that our iteration will converge to a global minimizer unless we know that objective function satisfies very strong conditions.

Linesearch methods

In certain situations we will not be able to guarantee that our iteration will converge to a global minimizer unless we know that objective function satisfies very strong conditions.

Try to generate an iteration that is globally convergent, that is that (at least) a subsequence of iterates $\{g(x_k)\}_{k=1}^{\infty}$ converges to zero.

Linesearch methods

In certain situations we will not be able to guarantee that our iteration will converge to a global minimizer unless we know that objective function satisfies very strong conditions.

Try to generate an iteration that is globally convergent, that is that (at least) a subsequence of iterates $\{g(x_k)\}_{k=1}^{\infty}$ converges to zero.

In what follows, we introduce notations

Linesearch methods

In certain situations we will not be able to guarantee that our iteration will converge to a global minimizer unless we know that objective function satisfies very strong conditions.

Try to generate an iteration that is globally convergent, that is that (at least) a subsequence of iterates $\{g(x_k)\}_{k=1}^{\infty}$ converges to zero.

In what follows, we introduce notations

$$
\begin{aligned}
f_k &= f(x_k), \\
g_k &= g(x_k), \\
H_k &= H(x_k).
\end{aligned}
$$

Linesearch methods

Theorem (Taylor for $\mathbb{R}^n$)

Let S be an open subset of $\mathbb{R}^n$, and suppose $f : S \to \mathbb{R}$ is twice continuously differentiable throughout S. Suppose further that $s \neq 0$, and that the interval $[x, x+s] \subset S$. Then

$$f(x+s) = f(x) + s^T g(x) + \frac{1}{2} s^T H(z) s$$

for some $z \in (x, x+s)$.

Linesearch methods

Theorem (Taylor for $\mathbb{R}^n$)

Let S be an open subset of $\mathbb{R}^n$, and suppose $f : S \to \mathbb{R}$ is twice continuously differentiable throughout S. Suppose further that $s \neq 0$, and that the interval $[x, x+s] \subset S$. Then

$$f(x+s) = f(x) + s^T g(x) + \frac{1}{2} s^T H(z) s$$

for some $z \in (x, x+s)$.

Definition

A direction $p_k \in \mathbb{R}^n$ is called a descent direction if

$$p_k^T g_k < 0 \quad \textit{if} \quad g_k \neq 0$$

Linesearch methods

Linesearch methods work as follows.

Linesearch methods

Linesearch methods work as follows.

First, a search direction $p_k \in \mathbb{R}^n$ is calculated from x_k.

Linesearch methods

Linesearch methods work as follows.

First, a search direction $p_k \in \mathbb{R}^n$ is calculated from x_k.

This direction is required to be a descent direction.

Linesearch methods

Linesearch methods work as follows.

First, a search direction $p_k \in \mathbb{R}^n$ is calculated from x_k.

This direction is required to be a descent direction.

So, small steps along p_k guarantees that the objective function may be reduced.

Linesearch methods

Linesearch methods work as follows.

First, a search direction $p_k \in \mathbb{R}^n$ is calculated from x_k.

This direction is required to be a descent direction.

So, small steps along p_k guarantees that the objective function may be reduced.

Second, a suitable steplength $\alpha_k > 0$ is calculated so that

$$f(x_k + \alpha_k p_k) < f_k$$

Linesearch methods

The computation of α_k is the linesearch, and may itself be an iteration.

Linesearch methods

The computation of α_k is the linesearch, and may itself be an iteration.

Finally, given both search direction p_k and steplength α_k, the iteration concludes by setting

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

Linesearch methods

The computation of α_k is the linesearch, and may itself be an iteration.

Finally, given both search direction p_k and steplength α_k, the iteration concludes by setting

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

Such a scheme sounds both natural and simple.

Linesearch methods

The computation of α_k is the linesearch, and may itself be an iteration.

Finally, given both search direction p_k and steplength α_k, the iteration concludes by setting

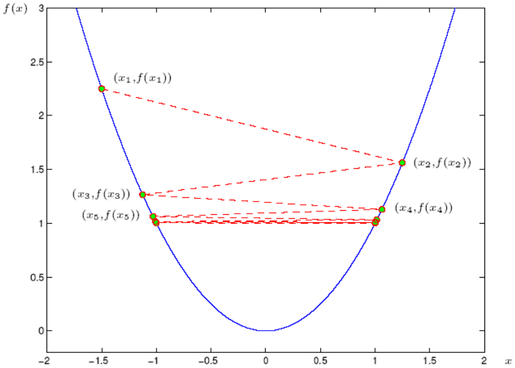
$$x_{k+1} = x_k + \alpha_k p_k$$

Such a scheme sounds both natural and simple. But as with most simple ideas, it needs to be refined somewhat in order to become a viable technique.

Linesearch methods

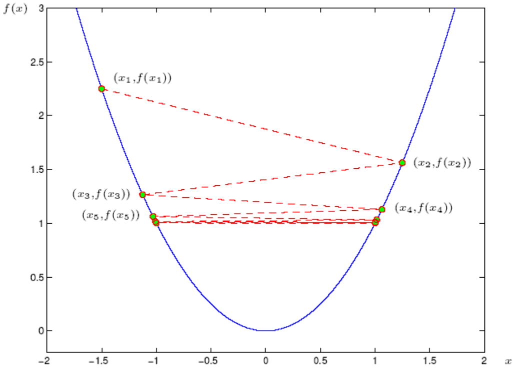
What might go wrong?

What might go wrong? Consider the example below



Linesearch methods

What might go wrong? Consider the example below

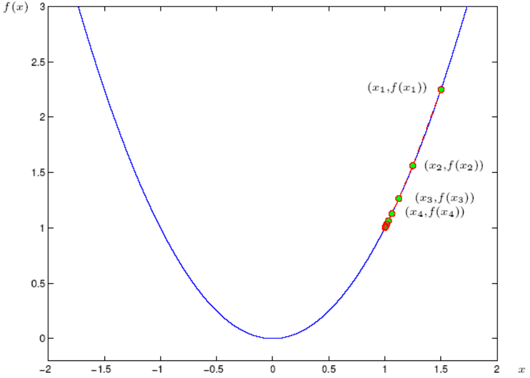


The objective function $f(x) = x^2$ and the iterates $x_{k+1} = x_k + \alpha_k p_k$ generated by the descent directions $p_k = (-1)^k$ and steps $\alpha_k = 2 + \frac{3}{2^{k+1}}$ from $x_0 = 2$.

Linesearch methods

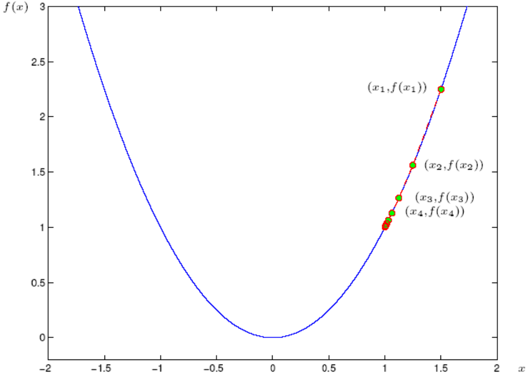
What has gone wrong?

What has gone wrong? The steps are too long relative to the amount of objective-function decrease that they provide. Consider another example



Linesearch methods

What has gone wrong? The steps are too long relative to the amount of objective-function decrease that they provide. Consider another example



The iterates $x_{k+1} = x_k + \alpha_k p_k$ generated by the descent directions $p_k = -1$ and steps $\alpha_k = \frac{1}{2^{k+}}$ from $x_0 = 2$.

Linesearch methods

The iterates approach the minimizer from one side, but the stepsizes are so small that each iterate falls short of the minimizer, and in the end converge to the non-critical value 1.

Linesearch methods

The iterates approach the minimizer from one side, but the stepsizes are so small that each iterate falls short of the minimizer, and in the end converge to the non-critical value 1.

A simple-minded linesearch method can fail if the linesearch allows steps that are either too long or too short relative to the amount of decrease that might be obtained with a well-chosen step.

Linesearch methods

The iterates approach the minimizer from one side, but the stepsizes are so small that each iterate falls short of the minimizer, and in the end converge to the non-critical value 1.

A simple-minded linesearch method can fail if the linesearch allows steps that are either too long or too short relative to the amount of decrease that might be obtained with a well-chosen step.

Earlier, it was suggested that α_k should be chosen to minimize $f(x_k + \alpha p_k)$. This is known as an **exact** linesearch. In most cases, exact linesearches prove to be very expensive: they are essentially univariate minimizations and most definitely not cost effective, and are consequently rarely used nowadays.

Linesearch methods

The iterates approach the minimizer from one side, but the stepsizes are so small that each iterate falls short of the minimizer, and in the end converge to the non-critical value 1.

A simple-minded linesearch method can fail if the linesearch allows steps that are either too long or too short relative to the amount of decrease that might be obtained with a well-chosen step.

Earlier, it was suggested that α_k should be chosen to minimize $f(x_k + \alpha p_k)$. This is known as an **exact** linesearch. In most cases, exact linesearches prove to be very expensive: they are essentially univariate minimizations and most definitely not cost effective, and are consequently rarely used nowadays.

Modern linesearch methods prefer to use **inexact** linesearches, which are guaranteed to pick steps that are neither too long nor too short.

Linesearch methods with backtracking

Among all possible inexact linesearches, the more used are the so called **backtracking Armijo** and the **Armijo-Goldstein** varieties.

Linesearch methods with backtracking

Among all possible inexact linesearches, the more used are the so called **backtracking Armijo** and the **Armijo-Goldstein** varieties. The first are easy to implement, and form the basis of most Newton-like linesearch methods.

Linesearch methods with backtracking

Among all possible inexact linesearches, the more used are the so called **backtracking Armijo** and the **Armijo-Goldstein** varieties. The first are easy to implement, and form the basis of most Newton-like linesearch methods. The second are important when using secant quasi-Newton methods but we will not discuss it here.

Linesearch methods with backtracking

Among all possible inexact linesearches, the more used are the so called **backtracking Armijo** and the **Armijo-Goldstein** varieties. The first are easy to implement, and form the basis of most Newton-like linesearch methods. The second are important when using secant quasi-Newton methods but we will not discuss it here.
Here is a basic backtracking linesearch to find α_k:

1. Given $\alpha_{init} > 0$ (e.g., $\alpha_{init} = 1$), let $\alpha^{(0)} = \alpha_{init}$ and $l = 0$.
2. Until $f(x_k + \alpha^{(l)} p_k) < f_k$
   1. set $\alpha^{(l+1)} = \tau\alpha^{(l)}$, where $\tau \in (0; 1)$ (e.g.,$\tau = \frac{1}{2}$)
   2. and increase l by 1
3. Set $\alpha_k = \alpha^{(l)}$.

Linesearch methods with backtracking

Notice that the backtracking strategy prevents the step from getting too small, since the 1st allowable value stepsize of the form $\alpha_{init}\tau^i$, $i = 0; 1, \ldots$ is accepted.

Linesearch methods with backtracking

Notice that the backtracking strategy prevents the step from getting too small, since the 1st allowable value stepsize of the form $\alpha_{init}\tau^i$, $i = 0; 1, \ldots$ is accepted.

However, as it stands, there is still no mechanism for preventing too large steps relative to decrease in f.

Linesearch methods with backtracking

Notice that the backtracking strategy prevents the step from getting too small, since the 1st allowable value stepsize of the form $\alpha_{init}\tau^{i}$, $i = 0; 1, \ldots$ is accepted.

However, as it stands, there is still no mechanism for preventing too large steps relative to decrease in f.

What is needed is a tighter requirement than simply that $f(x_k + \alpha^{(l)} p_k) < f_k$.

Notice that the backtracking strategy prevents the step from getting too small, since the 1st allowable value stepsize of the form $\alpha_{init}\tau^i$, $i = 0; 1, \ldots$ is accepted.

However, as it stands, there is still no mechanism for preventing too large steps relative to decrease in f.

What is needed is a tighter requirement than simply that $f(x_k + \alpha^{(l)} p_k) < f_k$.

Such a role is played by the Armijo condition.

Linesearch methods with backtracking

The Armijo condition is that the steplength be asked to give slightly more than simply decrease in f.

Linesearch methods with backtracking

The Armijo condition is that the steplength be asked to give slightly more than simply decrease in f.

The actual requirement is that

$$f(x_k + \alpha_k p_k) \leq f(x_k) + \alpha_k \beta p_k^T g_k$$

for some $\beta \in (0;1)$ (e.g., $\beta = 0.1$ or even $\beta = 0.0001$).

Linesearch methods with backtracking

The Armijo condition is that the steplength be asked to give slightly more than simply decrease in f.

The actual requirement is that

$$f(x_k + \alpha_k p_k) \leq f(x_k) + \alpha_k \beta p_k^T g_k$$

for some $\beta \in (0;1)$ (e.g., $\beta = 0.1$ or even $\beta = 0.0001$).

This requirement is often said to give sufficient decrease.

Linesearch methods with backtracking

The Armijo condition is that the steplength be asked to give slightly more than simply decrease in f.

The actual requirement is that

$$f(x_k + \alpha_k p_k) \leq f(x_k) + \alpha_k \beta p_k^T g_k$$

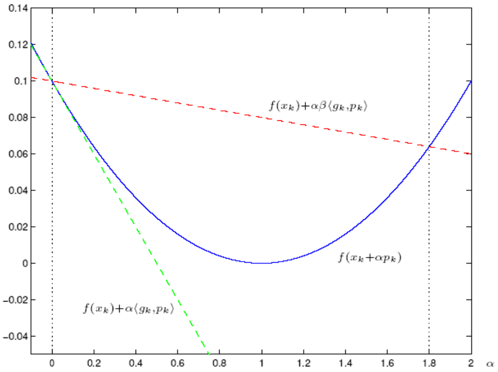
for some $\beta \in (0;1)$ (e.g., $\beta = 0.1$ or even $\beta = 0.0001$).

This requirement is often said to give sufficient decrease.

Observe that, since $p_k^T g_k < 0$, the longer the step, the larger the required decrease in f.

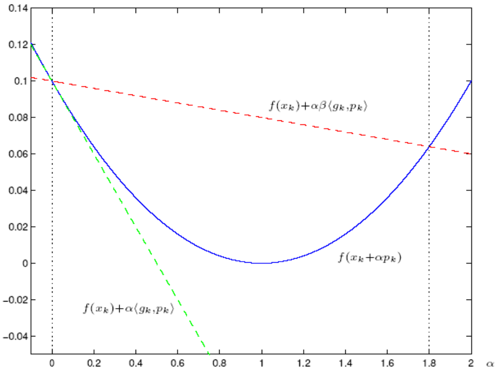
Linesearch methods with backtracking

The range of permitted values for the stepsize is illustrated below.



Linesearch methods with backtracking

The range of permitted values for the stepsize is illustrated below.



A steplength of anything up to 1.8 is permitted for this example, in the case where $\beta = 0.2$.

Linesearch methods with backtracking Armijo

The Armijo condition may then be inserted into our previous backtracking scheme to give the Backtracking-Armijo linesearch:

1. Given $\alpha_{init} > 0$ (e.g., $\alpha_{init} = 1$), let $\alpha^{(0)} = \alpha_{init}$ and $l = 0$.
2. Until $f(x_k + \alpha^{(l)} p_k) < f_k + \alpha^{(l)} \beta p_k^T g_k$
    1. set $\alpha^{(l+1)} = \tau \alpha^{(l)}$, where $\tau \in (0; 1)$ (e.g.,$\tau = \frac{1}{2}$)
    2. and increase l by 1
3. Set $\alpha_k = \alpha^{(l)}$.

Linesearch methods with backtracking Armijo

The Armijo condition may then be inserted into our previous backtracking scheme to give the Backtracking-Armijo linesearch:

1. Given $\alpha_{init} > 0$ (e.g., $\alpha_{init} = 1$), let $\alpha^{(0)} = \alpha_{init}$ and $l = 0$.
2. Until $f(x_k + \alpha^{(l)} p_k) < f_k + \alpha^{(l)} \beta p_k^T g_k$
    1. set $\alpha^{(l+1)} = \tau \alpha^{(l)}$, where $\tau \in (0; 1)$ (e.g.,$\tau = \frac{1}{2}$)
    2. and increase l by 1
3. Set $\alpha_k = \alpha^{(l)}$.

Of course, it is one thing to provide likely-sounding rules to control stepsize selection, but another to be sure that they have the desired effect. Indeed, can we even be sure that there are points which satisfy the Armijo condition?

Backtracking Armijo linesearch

Theorem

Suppose that $f \in C^2$, and that p is a descent direction at x. Then the Armijo condition

$$f(x+\alpha p) < f(x) + \alpha\beta p^T g(x)$$

is satisfied for all $\beta \in [0, \alpha_{\max(x,p)}]$, where

$$\alpha_{\max(x,p)} = \frac{2(\beta-1)p^T g(x)}{C_g \|p\|_2^2}$$

and C_g is the Lipschitz constant of gradient $g(x)$

Backtracking Armijo linesearch

Theorem

Suppose that $f \in C^2$, and that p is a descent direction at x. Then the Armijo condition

$$f(x + \alpha p) < f(x) + \alpha \beta p^T g(x)$$

is satisfied for all $\beta \in [0, \alpha_{\max(x,p)}]$, where

$$\alpha_{\max(x,p)} = \frac{2(\beta - 1)p^T g(x)}{C_g \|p\|_2^2}$$

and C_g is the Lipschitz constant of gradient $g(x)$

The numerator in $\alpha_{\max(x,p)}$ corresponds to the slope and the denominator to the curvature term. It can be interpreted as follows: If the curvature term is large, then the admissible range of α is small. Similarly, if the projected gradient along the search direction is large, then the range of admissible is larger.

Backtracking Armijo linesearch

Theorem

Suppose that $f \in C^2$, $\beta \in (0;1)$ and that p_k is a descent direction at x_k. Then the stepsize generated by the backtracking-Armijo linesearch terminates with

$$\alpha_k \geq \min\left(\alpha_{init}, \frac{2\tau(\beta - 1)p_k^T g_k}{C_g \left\| p_k \right\|_2^2}\right)$$

In order to tie all of the above together, we first need to state our **Generic Linesearch Method**:

Convergence of generic linesearch methods

In order to tie all of the above together, we first need to state our **Generic Linesearch Method**:

1. Given an initial guess x_0, let $k = 0$
2. Until convergence:
    1. Find a descent direction p_k at x_k.
    2. Compute a stepsize α_k using a backtracking-Armijo linesearch along p_k.
    3. Set $x_{k+1} = x_k + \alpha_k p_k$, and increase k by 1.

Convergence of generic linesearch methods

Theorem

Suppose that $f \in C^2$. Then, for the iterates generated by the Generic Linesearch Method, either

$$g_l = 0 \quad \text{for} \quad \text{some} \quad l > 0$$

or

$$\lim_{k \to \infty} f_k = -\infty$$

or

$$\lim_{k \to \infty} \min \left(\left| p_k^T g_k \right|, \frac{\left| p_k^T g_k \right|}{\| p_k \|_2^2} \right) = 0$$

Convergence of generic linesearch methods

In words, either we find a 1st-order critical point in a finite number of iterations, or we encounter a sequence of iterates for which the objective function is unbounded from below, or the slope (or a normalized slope) along the search direction converges to zero.

Convergence of generic linesearch methods

In words, either we find a 1st-order critical point in a finite number of iterations, or we encounter a sequence of iterates for which the objective function is unbounded from below, or the slope (or a normalized slope) along the search direction converges to zero.

While the first two of these possibilities are straightforward and acceptable consequences, the latter is perhaps not.

Convergence of generic linesearch methods

In words, either we find a 1st-order critical point in a finite number of iterations, or we encounter a sequence of iterates for which the objective function is unbounded from below, or the slope (or a normalized slope) along the search direction converges to zero.

While the first two of these possibilities are straightforward and acceptable consequences, the latter is perhaps not.

For one thing, it certainly does not say that the gradient converges to zero, that is the iterates may not ultimately be first-order critical, since it might equally occur if the search direction and gradient tend to be mutually orthogonal.

Convergence of generic linesearch methods

In words, either we find a 1st-order critical point in a finite number of iterations, or we encounter a sequence of iterates for which the objective function is unbounded from below, or the slope (or a normalized slope) along the search direction converges to zero.

While the first two of these possibilities are straightforward and acceptable consequences, the latter is perhaps not.

For one thing, it certainly does not say that the gradient converges to zero, that is the iterates may not ultimately be first-order critical, since it might equally occur if the search direction and gradient tend to be mutually orthogonal.

Thus we see that simply requiring that p_k be a descent directionis not a sufficiently demanding requirement.

Method of steepest descent

We have just seen that the Generic Linesearch Method may not succeed if the search direction becomes orthogonal to the gradient.

Method of steepest descent

We have just seen that the Generic Linesearch Method may not succeed if the search direction becomes orthogonal to the gradient.

Is there a direction for which this is impossible?

Method of steepest descent

We have just seen that the Generic Linesearch Method may not succeed if the search direction becomes orthogonal to the gradient.

Is there a direction for which this is impossible?

Yes, when the search direction is the descent direction

$$p_k = -g_k$$

Method of steepest descent

We have just seen that the Generic Linesearch Method may not succeed if the search direction becomes orthogonal to the gradient.

Is there a direction for which this is impossible?

Yes, when the search direction is the descent direction

$$p_k = -g_k$$

The so-called **steepest descent direction**.

Convergence of steepest descent method

Theorem

Suppose that $f \in C^2$. Then, for the iterates generated by the Steepest Descent Method, either

$$g_l = 0 \quad \textit{for} \quad \textit{some} \quad l > 0$$

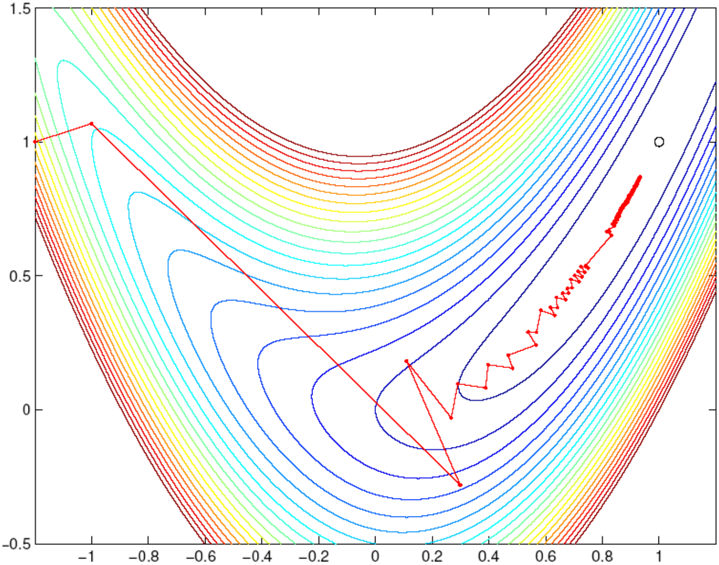
or

$$\lim_{k\to\infty} f_k = -\infty$$

or

$$\lim_{k\to\infty} g_k = 0$$

Convergence of steepest descent method



Convergence of steepest descent method

Convergence of steepest descent may be (and actually almost always is) very slow in theory, while numerically convergence sometimes does not occur at all as the iteration stagnates.

Convergence of steepest descent method

Convergence of steepest descent may be (and actually almost always is) very slow in theory, while numerically convergence sometimes does not occur at all as the iteration stagnates.

In practice, steepest-descent is all but worthless in most cases. The previuos figure exhibits quite typical behaviour in which the iterates repeatedly oscillate from one side of a objective function "valley" to the other. All of these phenomena may be attributed to a lack of attention to problem curvature when building the search direction.

Convergence of steepest descent method

Convergence of steepest descent may be (and actually almost always is) very slow in theory, while numerically convergence sometimes does not occur at all as the iteration stagnates.

In practice, steepest-descent is all but worthless in most cases. The previuos figure exhibits quite typical behaviour in which the iterates repeatedly oscillate from one side of a objective function "valley" to the other. All of these phenomena may be attributed to a lack of attention to problem curvature when building the search direction.

In next lecture we will consider methods that try to avoid this defect.