C477: First Order Methods

Ruth Misener
r.misener@imperial.ac.uk

Panos Parpas
p.parpas@imperial.ac.uk

Computational Optimisation Group
Department of Computing

Imperial College London

computational optimisation group

23 January 2020

Outline

- **Topics**
  - Overview of Gradient-Based Methods
  - Level Sets
  - Descent Directions
  - Steepest Descent Algorithm
  - Implementing First-Order methods
  - Convergence Theory
- **Example**
  - Recognising hand-written digits in the MNIST dataset
- **Reading**
  - Chapter 8 (Gradient Methods), Chong & Żak, Third Edition, Chapter 4 (The Gradient Method), Beck.

- **Acknowledgements**
  - Parts of these slides were originally developed by Benoit Chachuat and Panos Parpas. LaTeX design by Miten Mistry. Mistakes by Ruth Misener.

Introduction

C477 uses these terms interchangeably

- First-Order Methods
- Gradient-Based Methods
- Descent Algorithms

First-Order Methods in C477 vs The First Order in *Star Wars*

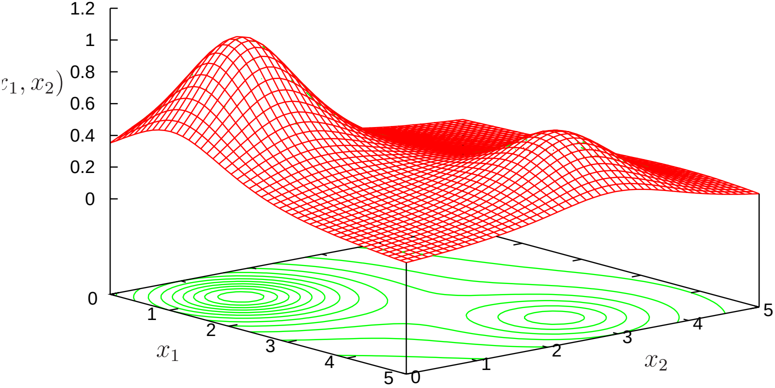
In the movie *The Force Awakens*, the First Order is a fictional military power. We argue that the power of the First Order may be related to using strong optimisation methods.

`http://starwars.wikia.com/wiki/First_Order`

Preview of First-Order, Gradient-Based Methods

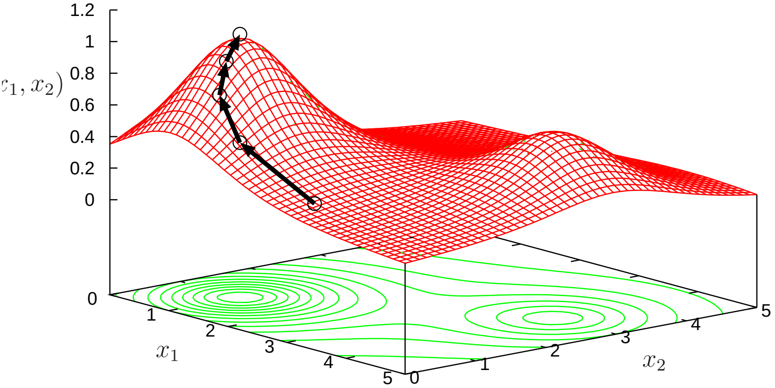
Consider the maximisation problem:

$$\max_{0 \le x_1,\, x_2 \le 5} f(x_1,\, x_2) \triangleq \frac{1}{1 + (x_1 - 1)^2 + (x_2 - 1)^2} + \frac{0.5}{1 + (x_1 - 4)^2 + (x_2 - 3)^2}$$



Preview of First-Order, Gradient-Based Methods

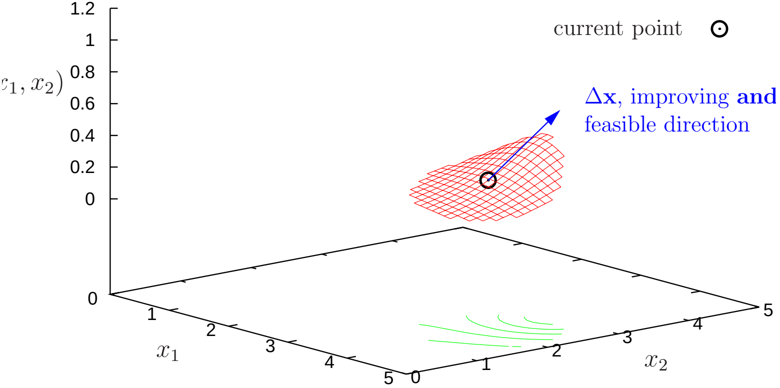
First-order methods are numerical algorithms starting at a feasible solution to an optimisation model. Using function and gradient evaluations, they advance along a search path of feasible points with improving function values.



Preview of First-Order, Gradient-Based Methods

The Dilemma: Typically, only some local information is know about the objective function typically at a current point $\boldsymbol{x}^k = (x_1^k, x_2^k)$!

➥ **May lead to a local optimum only!**



Overview of First-Order, Gradient-Based Methods

Unconstrained problem,

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

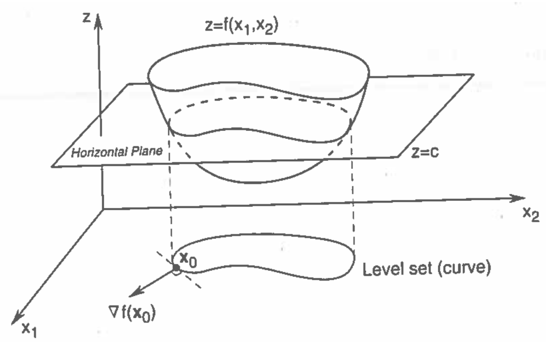
Idea: Generate a sequence of points $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(k)}, \ldots$ such that:

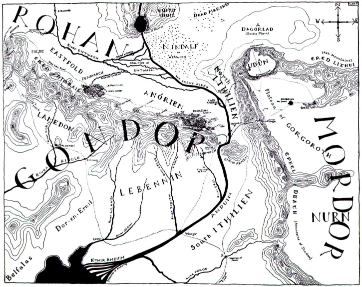
$$f\left(\boldsymbol{x}^{(k+1)}\right) < f\left(\boldsymbol{x}^{(k)}\right)$$

We seek convergence in the sense:

- $\lim_{k \to \infty} \boldsymbol{x}^{(k)} = \boldsymbol{x}^*$;
- $\lim_{k \to \infty} f\left(\boldsymbol{x}^{(k)}\right) = f^*$;
- $\boldsymbol{x}^*$ is a local minimum, e.g., satisfies optimality conditions;
- Sometimes more precise statements can be made.

Reminder: Level Set





Level Set

A level set of a function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is the set of points $\boldsymbol{x}$ satisfying $f(\boldsymbol{x}) = c$ for some constant $c \in \mathbb{R}$.

Reminder: Directional Derivative

Definition (Directional Derivative)

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a real valued function, and let $\boldsymbol{d} \in \mathbb{R}^n \backslash \mathbf{0}$. The directional derivative of f in the direction $\boldsymbol{d}$ is defined:

$$\frac{\partial f}{\partial \boldsymbol{d}}(\boldsymbol{x}) = \lim_{\alpha \to 0} \frac{f(\boldsymbol{x} + \alpha \boldsymbol{d}) - f(\boldsymbol{x})}{\alpha} = \boldsymbol{d}^\top \nabla f(\boldsymbol{x})$$

Reminder: The gradient of f is denoted:

$$\nabla f(\boldsymbol{x}) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right]^\top .$$

The Jacobian of f is denoted by Df and $\nabla f(\boldsymbol{x}) = Df^\top$. The rate of change in f at $\boldsymbol{x}$ in the i^{th} direction is $\partial f / \partial x_i(\boldsymbol{x})$.

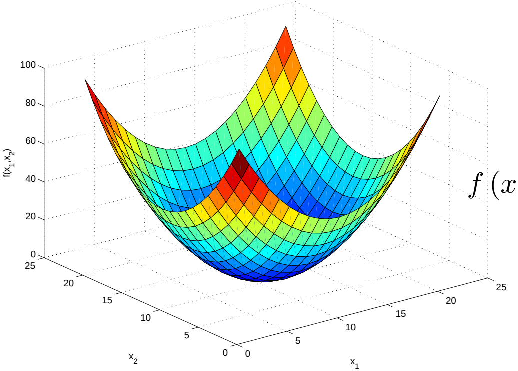
- Directional derivatives allow all the dimensions to vary at the same time and by different amounts. For consistency: $\|\boldsymbol{d}\|_2 = 1$

Example

Find the directional derivative of f:

$$f(\boldsymbol{x}) = \frac{1}{2}\left(x_1^2 + x_2^2\right),$$

at the point $\boldsymbol{x} = [-4,\, 5]^\top$ in the direction $\boldsymbol{d} = \left[\frac{-2}{\sqrt{13}},\, \frac{3}{\sqrt{13}}\right]^\top$



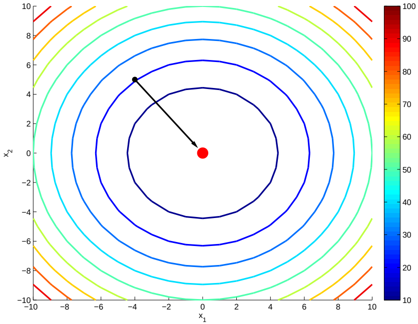
$$f(x_1,\, x_2) = \frac{1}{2}\left(x_1^2 + x_2^2\right)$$

Example

Find the directional derivative of f

$$f(\boldsymbol{x}) = \frac{1}{2}(x_1^2 + x_2^2)$$

at the point $\boldsymbol{x} = [-4,\, 5]^\top$ in the direction $\boldsymbol{d} = \left[\frac{4}{\sqrt{41}}, \frac{-5}{\sqrt{41}}\right]^\top$.

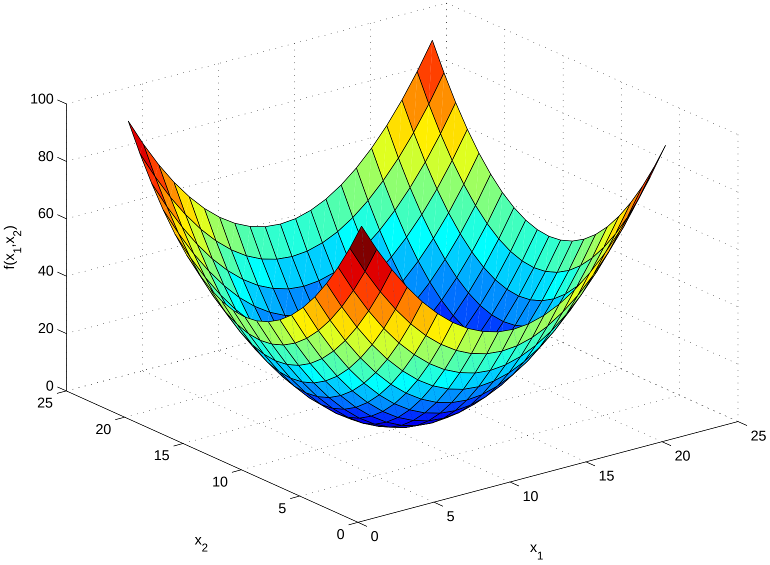


Sanity Check

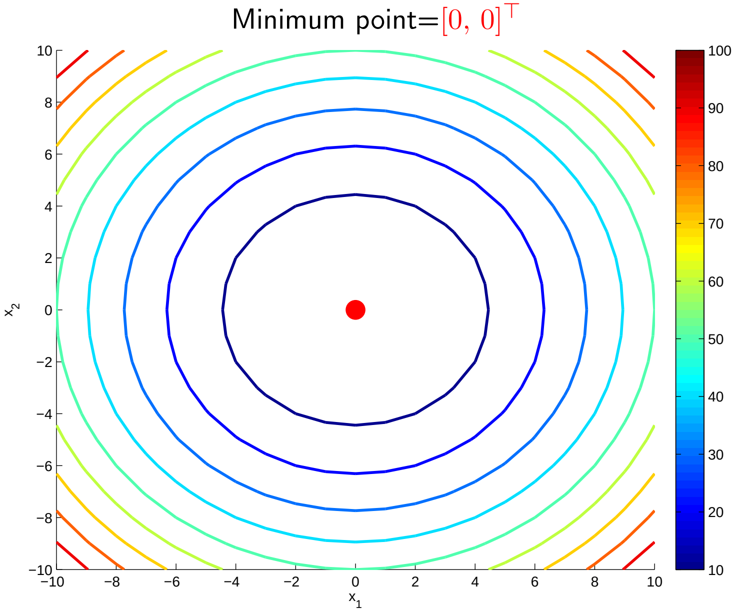
Does the function increase or decrease at the point $\boldsymbol{x}$ in the direction $\boldsymbol{d}$?

Example

$$f\left(x_1,\, x_2\right) = \frac{1}{2}\left(x_1^2 + x_2^2\right)$$

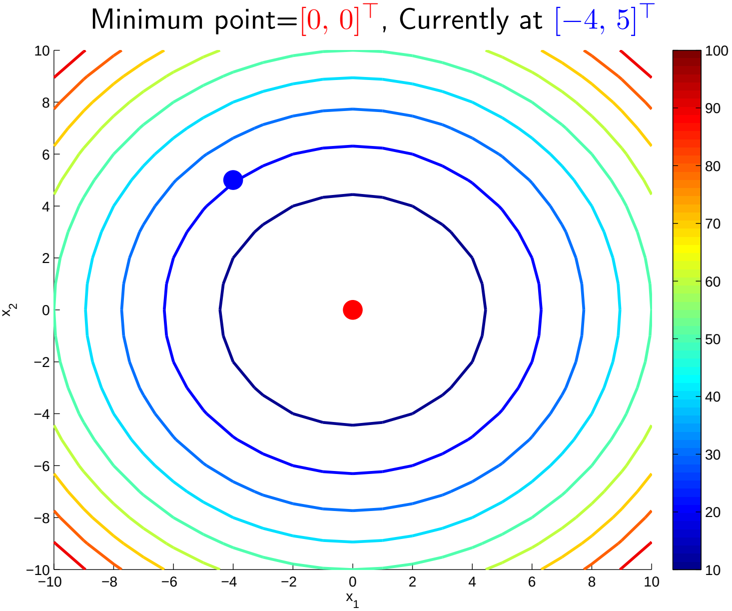


Example



Minimum point=$[0, 0]^\top$
10
8
6
4
2
0
−2
−4
−6
−8
−10
x_2
−10
−8
−6
−4
−2
0
2
4
6
8
10
x_1
100
90
80
70
60
50
40
30
20
10

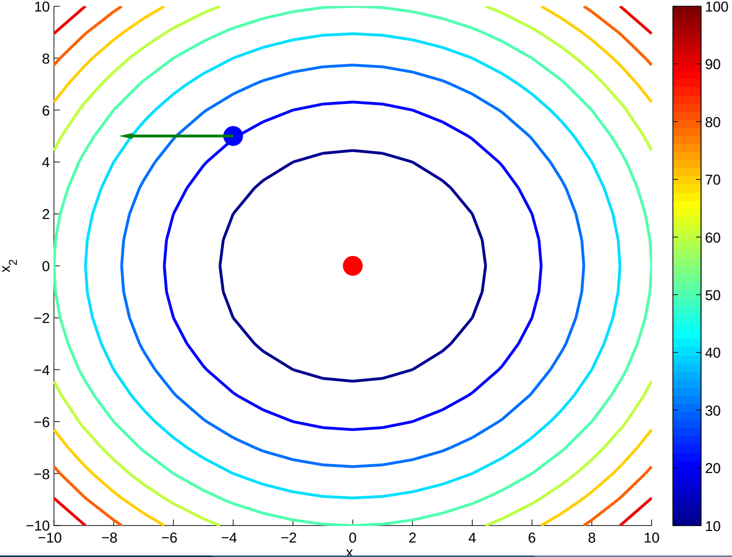
Example



Minimum point=$[0,\ 0]^\top$, Currently at $[-4,\ 5]^\top$
10
8
6
4
2
0
−2
−4
−6
−8
−10
x2
−10
−8
−6
−4
−2
0
2
4
6
8
10
x1
100
90
80
70
60
50
40
30
20
10

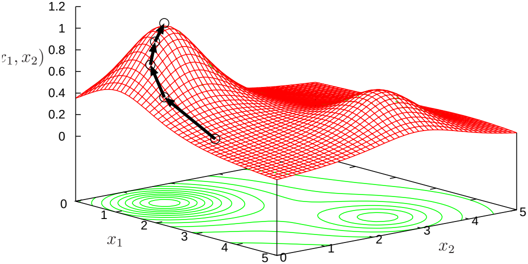
Example

$\langle \nabla f(\boldsymbol{x}), \mathbb{1}_1 \rangle = \left[\frac{\partial f}{\partial x_1}, 0 \right]^\top$ at $[-4, 5]^\top$



Basic Ingredients of Descent Algorithms

1. Given a point $\boldsymbol{x}^{(k)}$.
2. Derive a descent direction $\boldsymbol{d}^{(k)} \in \mathbb{R}^n$, i.e.,
$$\nabla f\left(\boldsymbol{x}^{(k)}\right)^\top \boldsymbol{d}^{(k)} < 0.$$
3. Decide on a step-size α_k.
4. Transition to the next point,
$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}.$$



Descent Directions

Definition: Descent Direction ($\boldsymbol{d}$)

Let $f : \mathbb{R}^n \mapsto \mathbb{R}$ be a continuously differentiable function over $\mathbb{R}^n$. A vector $\boldsymbol{d} \neq \mathbf{0}$ is called a descent direction of f at $\boldsymbol{x}$ if the directional derivative of f at $\boldsymbol{x}$ in the direction $\boldsymbol{d}$ is negative:

$$\nabla f(\boldsymbol{x})^\top \boldsymbol{d} < 0.$$

Sanity Check

What does the descent condition mean?

Step Size: $\alpha_k > 0$

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}$$

Sanity Check

Why do we need the step-size computation?

Steepest Ascent

For a small displacement, the function f increases more in the direction of the gradient than in any other direction. By definition:

$$\langle \nabla f(\boldsymbol{x}), \boldsymbol{d} \rangle = \nabla f(\boldsymbol{x})^\top \boldsymbol{d} \text{ with } \|\boldsymbol{d}\|_2 = 1$$

is the rate of increase in f along the direction $\boldsymbol{d}$. Using the Cauchy-Schwarz inequality,

$$\langle \nabla f(\boldsymbol{x}), \boldsymbol{d} \rangle \leq \|\nabla f(\boldsymbol{x})\|_2 \|\boldsymbol{d}\|_2 = \|\nabla f(\boldsymbol{x})\|_2$$

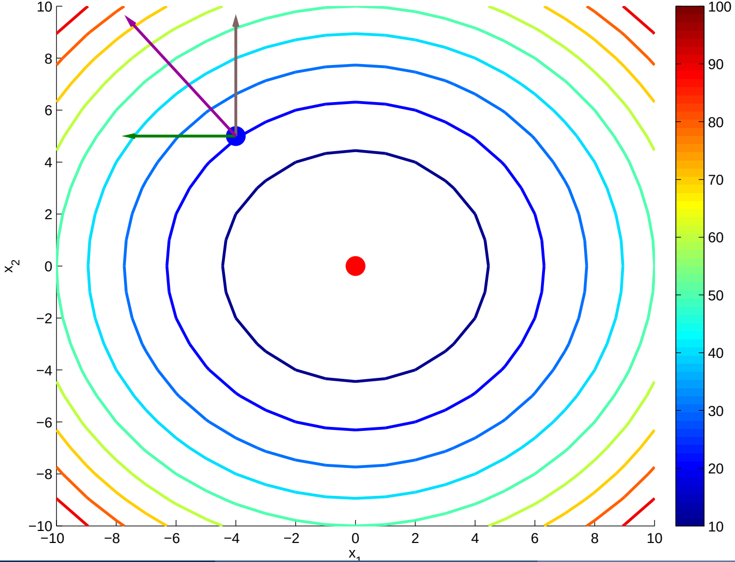
Therefore the rate of increase cannot be greater than $\|\nabla f(\boldsymbol{x})\|_2$. Then taking $\boldsymbol{d} = \nabla f(\boldsymbol{x}) / \|\nabla f(\boldsymbol{x})\|_2$:

$$\langle \nabla f(\boldsymbol{x}), \boldsymbol{d} \rangle = \left\langle \nabla f(\boldsymbol{x}), \frac{\nabla f(\boldsymbol{x})}{\|\nabla f(\boldsymbol{x})\|_2} \right\rangle = \frac{\|\nabla f(\boldsymbol{x})\|_2^2}{\|\nabla f(\boldsymbol{x})\|_2} = \|\nabla f(\boldsymbol{x})\|_2$$

and $\boldsymbol{d} = \frac{\nabla f(\boldsymbol{x})}{\|\nabla f(\boldsymbol{x})\|_2}$ achieves the upper bound.

Example

$$\langle \nabla f(\boldsymbol{x}), \mathbb{1}_1 \rangle = \left[\frac{\partial f}{\partial x_1}, 0\right]^\top, \langle \nabla f(\boldsymbol{x}), \mathbb{1}_2 \rangle = \left[0, \frac{\partial f}{\partial x_2}\right]^\top \nabla f(\boldsymbol{x}) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}\right]^\top$$



General Justification of Descent Condition

1. Suppose that $\boldsymbol{d}^{(k)}$ is a descent direction $\langle \nabla f\left(\boldsymbol{x}^{(k)}\right), \boldsymbol{d}^{(k)} \rangle < 0$;
2. Compute a point such that $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha \boldsymbol{d}^{(k)}$;
3. Then if α is small enough we have:

$$f\left(\boldsymbol{x}^{(k+1)}\right) < f\left(\boldsymbol{x}^{(k)}\right).$$

Using Taylor's theorem we obtain,

$$f\left(\boldsymbol{x}^{(k+1)}\right) = f\left(\boldsymbol{x}^{(k)}\right) + \alpha \nabla f\left(\boldsymbol{x}^{(k)}\right)^{\top} \boldsymbol{d}^{(k)} + r(\alpha \boldsymbol{d}^{(k)}).$$

Note that $\nabla f\left(\boldsymbol{x}^{(k)}\right) \neq 0$, $\boldsymbol{d}$ is a descent direction so if α is small enough we obtain,

$$f\left(\boldsymbol{x}^{(k+1)}\right) < f\left(\boldsymbol{x}^{(k)}\right).$$

Descent Directions – Summary

1. Given a point $\boldsymbol{x}^{(k)}$.
2. Derive a descent direction $\boldsymbol{d}^{(k)} \in \mathbb{R}^n$, i.e.,

$$\nabla f\left(\boldsymbol{x}^{(k)}\right)^\top \boldsymbol{d}^{(k)} < 0.$$

3. Decide on a step-size α_k.
4. Transition to the next point,

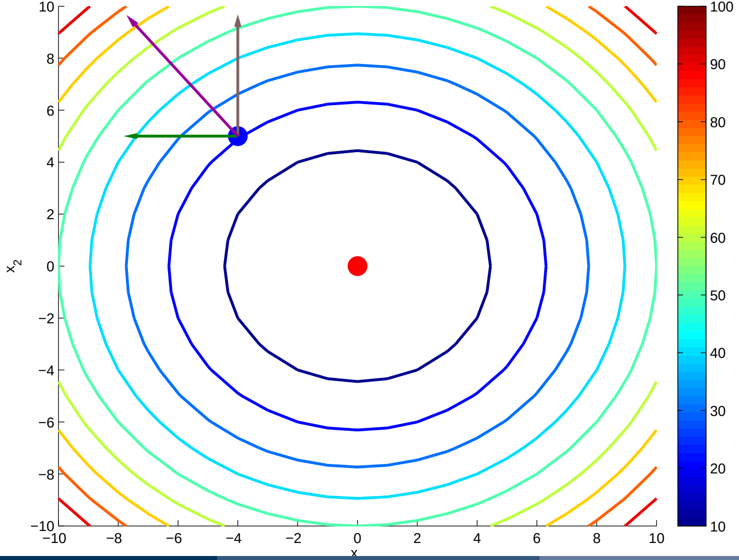
$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}.$$

Sanity Check

What does $\nabla f\left(\boldsymbol{x}^{(k)}\right)^\top \boldsymbol{d}^{(k)} < 0$ mean? How do we chose a good α?

How to select the descent direction?

$$\nabla f(\boldsymbol{x}) = \left[\frac{\partial f}{\partial x_1},\ \frac{\partial f}{\partial x_2}\right]^\top$$: Direction of greatest increase



How to select the step–size

If $\boldsymbol{d}^{(k)}$ is a descent direction, e.g., $\boldsymbol{d}^{(k)} = -\nabla f\left(\boldsymbol{x}^{(k)}\right) / \left\|\nabla f\left(\boldsymbol{x}^{(k)}\right)\right\|_2$, and:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}$$

We know that if α_k is small enough then, $f\left(\boldsymbol{x}^{(k+1)}\right) < f\left(\boldsymbol{x}^{(k)}\right)$, i.e., the algorithm improves the current point $\boldsymbol{x}^{(k)}$.

Exact Step Size Strategy

Select the step size with the maximum improvement, i.e.,

$$\alpha_k \in \arg\min_{\alpha \geq 0} \; f(\boldsymbol{x}^{(k)} + \alpha \boldsymbol{d}^{(k)})$$

Sanity Check

How many dimensions in the optimisation problem calculating the exact step size? If f is convex, will it have any special properties?

Steepest Descent Algorithm

General Descent Algorithm

1. Given a point $\boldsymbol{x}^{(k)}$.
2. Derive a descent direction $\boldsymbol{d}^{(k)} \in \mathbb{R}^n$, i.e.,
$$\nabla f\left(\boldsymbol{x}^{(k)}\right)^\top \boldsymbol{d}^{(k)} < 0.$$
3. Decide on a step-size α_k.
4. Transition to the next point,
$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}$$

Steepest Descent Algorithm

1. Given a point $\boldsymbol{x}^{(k)}$.
2. Compute the gradient at $\boldsymbol{x}^{(k)}$
$$\boldsymbol{d}^{(k)} = -\nabla f\left(\boldsymbol{x}^{(k)}\right)$$
($\boldsymbol{d}^{(k)}$ is a descent direction)
3. Decide on a step-size α_k,
$$\alpha_k \in \arg\min_{\alpha \geq 0} f(\boldsymbol{x}^{(k)} + \alpha \boldsymbol{d}^{(k)})$$
4. Transition to the next point,
$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}$$

Implementation Issues [1/2]

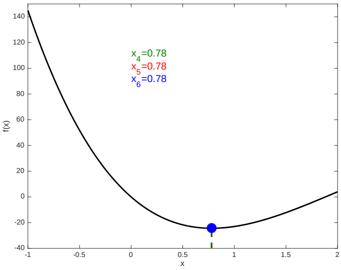
Convergence Criteria

1. Stop when FONC is satisfied:

$$\nabla f\left(\boldsymbol{x}^{(k)}\right) = \mathbf{0}.$$

2. In practice
   a. $\left\|\nabla f\left(\boldsymbol{x}^{(k)}\right)\right\|_2 < \epsilon_1$;
   b. $\left\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\right\|_2 < \epsilon_2$;
   c. $\left|f\left(\boldsymbol{x}^{(k+1)}\right) - f\left(\boldsymbol{x}^{(k)}\right)\right| < \epsilon_3$;

where $\epsilon_1, \epsilon_2, \epsilon_3 > 0$ are tolerance parameters.



Implementation Issues [2/2]

Computing Derivatives

1. Use exact derivatives (symbolic/manual);
2. Finite Difference;
3. Automatic Differentiation.

Step-size Strategy

1. Exact Line Search; minimise with a one dimensional method:

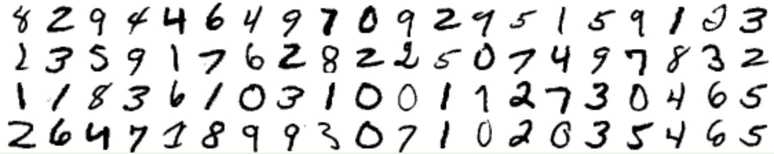
$$\alpha_k \in \arg\min_{\alpha} f(\boldsymbol{x}^{(k)} + \alpha \boldsymbol{d}^{(k)})$$

2. Inexact line search
   - Percentage Test, Armijo's Rule, Curve Fitting, Backtracking, ...

Example: Sparse Logistic Regression

Challenge: How can we classify hand-written digits from images?

$$\min_{\boldsymbol{\beta_1}, \ldots, \boldsymbol{\beta_{10}}} \sum_{i=1}^{m} \left(\log \sum_{k=1}^{10} \exp(\boldsymbol{x_i}^\top \boldsymbol{\beta_k}) - \boldsymbol{x_i}^\top \boldsymbol{\beta_{y_i}} \right) + \lambda \sum_{k=1}^{10} \|\boldsymbol{\beta_k}\|_1$$



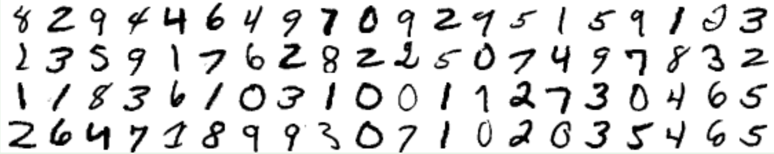
Where $\boldsymbol{x_i} \in \mathbb{R}^n$ and $y_i \in \{0, 1, \ldots, 9\}$ are the input features and output label, respectively, for each training example, $i \in \{1, 2, \ldots, m\}$. There are also feature weights, $\boldsymbol{\beta_k} \in \mathbb{R}^n$, for $k = \{0, 1, \ldots, 9\}$

Typically use first-order, subgradient-based methods, to solve this.

Example: Sparse Logistic Regression

Changing the Problem: Classify hand-written digits from images?

$$\min_{\boldsymbol{\beta_1},\ldots,\boldsymbol{\beta_{10}}} \sum_{i=1}^{m} \left(\log \sum_{k=1}^{10} \exp(\boldsymbol{x_i}^\top \boldsymbol{\beta_k}) - \boldsymbol{x_i}^\top \boldsymbol{\beta_{y_i}} \right) + \lambda \sum_{k=1}^{10} \|\boldsymbol{\beta_k}\|_2^2$$



Where $\boldsymbol{x_i} \in \mathbb{R}^n$ and $y_i \in \{0, 1, \ldots, 9\}$ are the input features and output label, respectively, for each training example, $i \in \{1, 2, \ldots, m\}$. There are also feature weights, $\boldsymbol{\beta_k} \in \mathbb{R}^n$, for $k = \{0, 1, \ldots, 9\}$

Now we can use first-order, gradient-based methods, to solve this.

Inexact Line Search

Examples of Inexact Line Search

1. **Percentage Test**: Similar to exact line search but stops within a fixed accuracy, e.g., $|\alpha_k - \alpha^*| < c\alpha^*$, with $0 < c < 1$, $c = 0.1$ a typical value;
2. **Curve Fitting:** Fit a function to $f\left(\boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}\right)$ and minimise that, e.g., Newton's method with a quadratic fit;
3. **Armijo's Rule:** A rule to ensure that α is not too large and not too small;
4. **Backtracking:** A simple variation of Armijo's rule.

Backtracking Line Search

Backtracking Line Search

One way to adaptively choose the step size is to initialise parameters $0 < c_\alpha < 1$ and $0 < c_\beta < 1$. At each iteration k, start with $\alpha^{(k)} = 1$, and while:

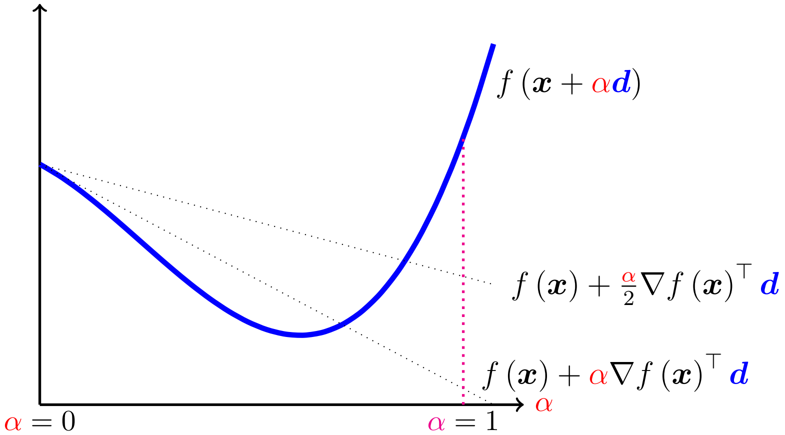
$$f\left(\boldsymbol{x}^{(k)} - \alpha^{(k)}\nabla f\left(\boldsymbol{x}^{(k)}\right)\right) > f(\boldsymbol{x}^{(k)}) - c_\alpha\alpha^{(k)}\left\|\nabla f\left(\boldsymbol{x}^{(k)}\right)\right\|_2^2$$

shrink $\alpha^{(k)} = c_\beta\alpha^{(k)}$. Else perform gradient descent update:

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

Simple and tends to work well in practice. A common choice is $c_\alpha = 1/2$.

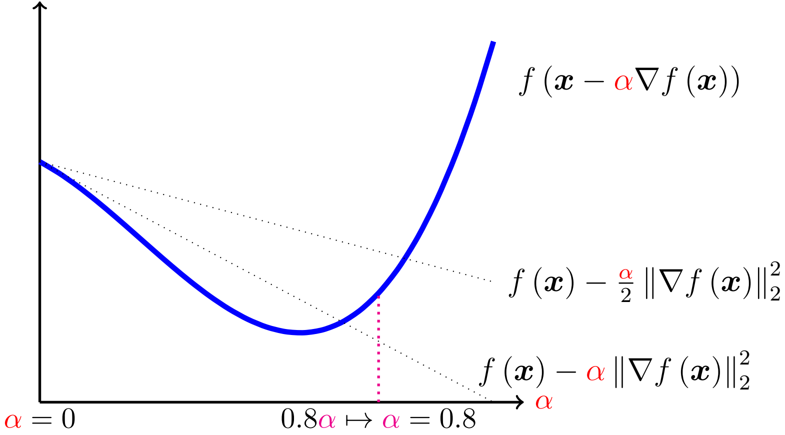
Backtracking interpretation - Steepest Descent



Sanity Check

Backtracking line search initialises parameters $0 < c_\alpha < 1$ and $0 < c_\beta < 1$, e.g. $c_\alpha = 1/2$ and $c_\beta = 0.8$. Would it be reasonable to choose $c_\alpha = 1$ for a twice continuously differentiable function?

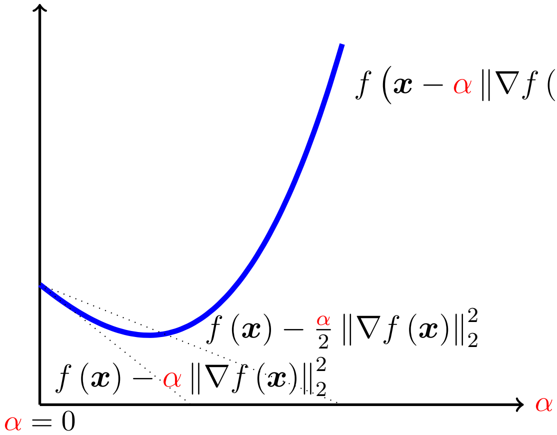
Backtracking interpretation - Steepest Descent



Sanity Check

Backtracking line search initialises parameters $0 < c_\alpha < 1$ and $0 < c_\beta < 1$, e.g. $c_\alpha = 1/2$ and $c_\beta = 0.8$. Would it be reasonable to choose $c_\alpha = 1$ for a twice continuously differentiable function?

Backtracking interpretation - Steepest Descent



Sanity Check

Backtracking line search initialises parameters $0 < c_\alpha < 1$ and $0 < c_\beta < 1$, e.g. $c_\alpha = 1/2$ and $c_\beta = 0.8$. Would it be reasonable to choose $c_\alpha = 1$ for a twice continuously differentiable function?

Convergence Theory

Some General Convergence Results

If function $f \in \mathcal{C}^2$ is strictly convex and we use an exact line search algorithm, then:

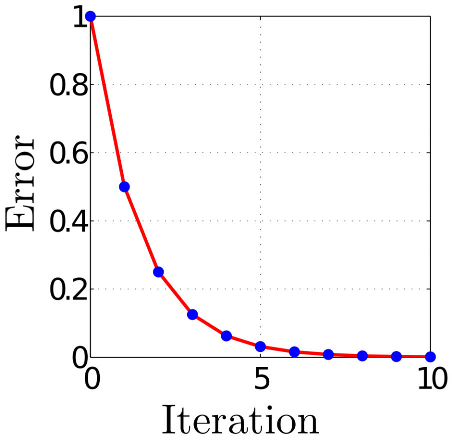
- **Convergence:** The sequence generated by the algorithm $f\left(\boldsymbol{x}^{(k)}\right)$ converges to $f(\boldsymbol{x}^*)$.
- **Convergence Rate:** The algorithm converges *linearly* to $\boldsymbol{x}^*$ with a rate given by,

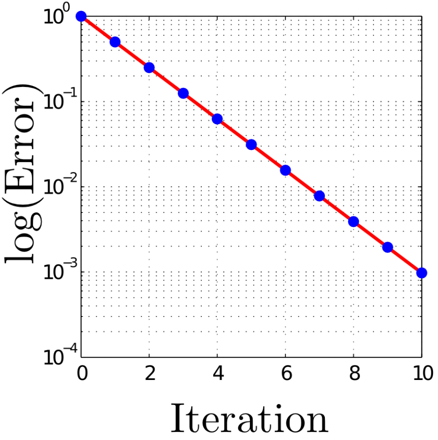
$$f\left(\boldsymbol{x}^{(k)}\right) - f^* \leq \left(1 - \frac{m}{M}\right)^k \left(f\left(\boldsymbol{x}^{(0)}\right) - f^*\right),$$

  where x_0 is the initial starting point. We define the smallest and greatest eigenvalue of the Hessian of f at $\boldsymbol{x}^*$ as $m = \lambda_{\min}\left(\nabla^2 f(\boldsymbol{x}^*)\right)$ and $M = \lambda_{\max}\left(\nabla^2 f(\boldsymbol{x}^*)\right)$.

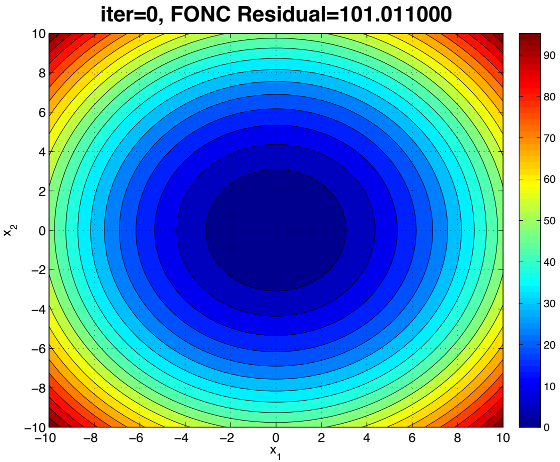
Convergence Rate

Linear convergence rate means converges linearly in log scale

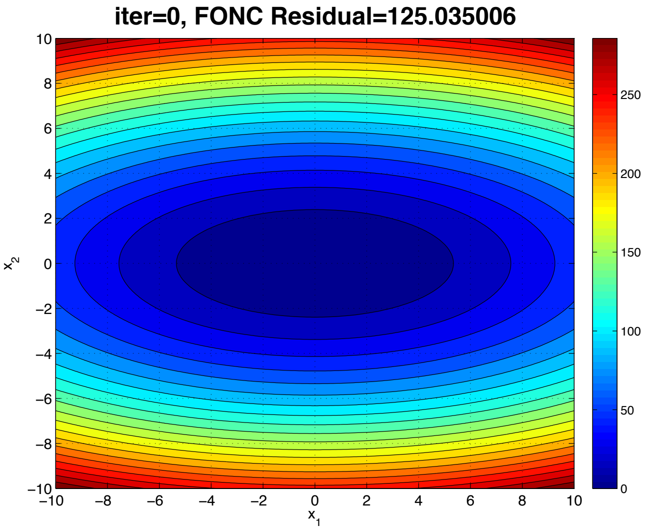




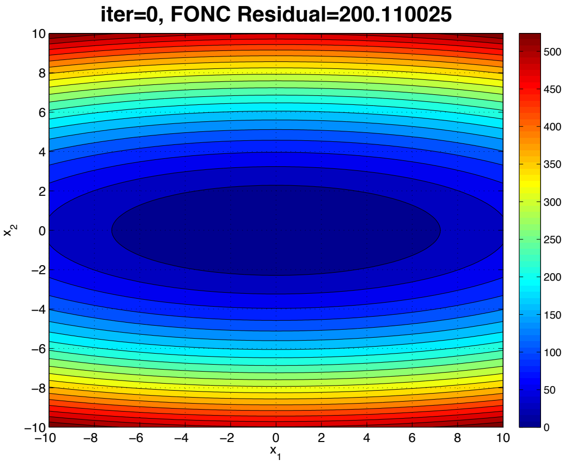
$$f(\boldsymbol{x}) = \frac{1}{2}(x_1^2 + \gamma x_2^2) \quad \gamma = 1,\ x^{(0)} = [10,\ 1]^\top$$



$$f(\boldsymbol{x}) = \frac{1}{2}(x_1^2 + \gamma x_2^2) \quad \gamma = 5,\ x^{(0)} = [10,\ 1]^\top$$



$$f(\boldsymbol{x}) = \frac{1}{2}(x_1^2 + \gamma x_2^2) \quad \gamma = 10,\ x^{(0)} = [10,\ 1]^\top$$



For First-Order Algorithms

Pro:

+ Easy to implement;
+ Only requires first-order, gradient-based information;
+ Convergence results exist if a appropriate step size strategy is used;
+ Only choice for large scale problems;
+ Variations for non-differentiable problems exist; these are called sub-gradient methods and are not covered in this course.

Against First-Order Algorithms

Con:

- Convergence is slow (there are algorithms with quadratic rate, see lecture on 2nd order methods);
- Sensitive to the scaling of the problem.

First Order Insignia

Condition Number

Definition of Condition Number

The condition number of an $n \times n$ positive definite matrix $\boldsymbol{Q}$ is the ratio between the maximal and minimal eigenvalues of $\boldsymbol{Q}$:

$$\chi(\boldsymbol{Q}) = \frac{M}{m} = \frac{\lambda_{\max}(\boldsymbol{Q})}{\lambda_{\min}(\boldsymbol{Q})}.$$

Condition Number Example [1/3]

Quadratic Minimisation Example [Beck, 2014]

Consider the quadratic minimisation problem:

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} \left\{ f(\boldsymbol{x}) \equiv \boldsymbol{x}^\top \boldsymbol{Q} \boldsymbol{x} \right\}$$

where $\boldsymbol{Q} \succ 0$. The optimal solution is obviously $\boldsymbol{x}^* = \mathbf{0}$. The gradient method with exact line search takes the form:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)},$$

where $\boldsymbol{d}^{(k)} = -2\boldsymbol{Q}\boldsymbol{x}^{(k)}$ is the negative gradient of f at $\boldsymbol{x}^{(k)}$ and the stepsize α_k chosen by the exact minimisation rule is (why?):

$$\alpha_k = \frac{\boldsymbol{d}^{(k)\top} \boldsymbol{d}^{(k)}}{2\boldsymbol{d}^{(k)\top} \boldsymbol{Q} \boldsymbol{d}^{(k)}}.$$

Condition Number Example [2/3]

Quadratic Minimisation Example (continued)

Therefore:

$$
\begin{aligned}
f\left(\boldsymbol{x}^{(k+1)}\right) =& \boldsymbol{x}^{(k+1)\top}\boldsymbol{Q}\boldsymbol{x}^{(k+1)} \\
=& \left(\boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}\right)^\top \boldsymbol{Q} \left(\boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}\right) \\
=& \boldsymbol{x}^{(k)\top}\boldsymbol{Q}\boldsymbol{x}^{(k)} + 2\alpha_k \boldsymbol{d}^{(k)\top}\boldsymbol{Q}\boldsymbol{x}^{(k)} + \alpha_k^2 \boldsymbol{d}^{(k)\top}\boldsymbol{Q}\boldsymbol{d}^{(k)} \\
=& \boldsymbol{x}^{(k)\top}\boldsymbol{Q}\boldsymbol{x}^{(k)} - \alpha_k \boldsymbol{d}^{(k)\top}\boldsymbol{d}^{(k)} + \alpha_k^2 \boldsymbol{d}^{(k)\top}\boldsymbol{Q}\boldsymbol{d}^{(k)}
\end{aligned}
$$

plugging in the expression for α_k given on the previous slide and rearranging (ugly maths):

$$
f\left(\boldsymbol{x}^{(k+1)}\right) = \left(1 - \frac{\left(\boldsymbol{d}^{(k)\top}\boldsymbol{d}^{(k)}\right)^2}{\left(\boldsymbol{d}^{(k)\top}\boldsymbol{Q}\boldsymbol{d}^{(k)}\right)\left(\boldsymbol{d}^{(k)\top}\boldsymbol{Q}^{-1}\boldsymbol{d}^{(k)}\right)}\right) f\left(\boldsymbol{x}^{(k)}\right)
$$

Condition Number Example [3/3]

Kantorovich Inequality (stated without proof)

Let $\boldsymbol{Q}$ be a positive definite $n \times n$ matrix. Then for any $\mathbf{0} \neq \boldsymbol{x} \in \mathbb{R}^n$, the following inequality holds, where $m = \lambda_{\min}(\boldsymbol{Q})$ and $M = \lambda_{\max}(\boldsymbol{Q})$:

$$\frac{\left(\boldsymbol{x}^T \boldsymbol{x}\right)^2}{\left(\boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x}\right)\left(\boldsymbol{x}^T \boldsymbol{Q}^{-1} \boldsymbol{x}\right)} \geq \frac{4\lambda_{\max}(\boldsymbol{Q})\lambda_{\min}(\boldsymbol{Q})}{\left(\lambda_{\max}(\boldsymbol{Q}) + \lambda_{\min}(\boldsymbol{Q})\right)^2} = \frac{4Mm}{(M+m)^2}.$$

Quadratic Minimisation Example (continued)

$$\begin{aligned} f\left(\boldsymbol{x}^{(k+1)}\right) &= \left(1 - \frac{\left(\boldsymbol{d}^{(k)\top} \boldsymbol{d}^{(k)}\right)^2}{\left(\boldsymbol{d}^{(k)\top} \boldsymbol{Q} \boldsymbol{d}^{(k)}\right)\left(\boldsymbol{d}^{(k)\top} \boldsymbol{Q}^{-1} \boldsymbol{d}^{(k)}\right)}\right) f\left(\boldsymbol{x}^{(k)}\right) \\ &\leq \left(1 - \frac{4Mm}{(M+m)^2}\right) f\left(\boldsymbol{x}^{(k)}\right) = \left(\frac{M-m}{M+m}\right)^2 f\left(\boldsymbol{x}^{(k)}\right) \\ &= \left(\frac{\chi - 1}{\chi + 1}\right)^2 f\left(\boldsymbol{x}^{(k)}\right) \leq \left(\frac{\chi - 1}{\chi + 1}\right)^{2(k+1)} f\left(\boldsymbol{x}^{(0)}\right), \end{aligned}$$

and the speed of convergence depends on the condition number χ.

How to deal with poor condition numbers? [1/3]

Motivation: Ill-conditioned optimisation problems are difficult

One way to circumvent this issue is to *condition* the problem by making appropriate linear transformation of the decision variables.

How to condition an unconstrained optimisation problem [Beck, 2014]

Consider:

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

For a given nonsingular matrix $\boldsymbol{S} \in \mathbb{R}^{n \times n}$, we make the linear transformation $\boldsymbol{x} = \boldsymbol{S}\boldsymbol{y}$ and obtain the equivalent problem:

$$\min_{\boldsymbol{y} \in \mathbb{R}^n} \{g(\boldsymbol{y}) \equiv f(\boldsymbol{S}\boldsymbol{y})\}.$$

Since $\nabla g(\boldsymbol{y}) = \boldsymbol{S}^\top \nabla f(\boldsymbol{S}\boldsymbol{y}) = \boldsymbol{S}^\top \nabla f(\boldsymbol{x})$, it follows that the gradient method applied to the transformed problem takes the form:

$$\boldsymbol{y}^{(k+1)} = \boldsymbol{y}^{(k)} - \alpha_k \boldsymbol{S}^\top \nabla f(\boldsymbol{S}\boldsymbol{y}^{(k)}).$$

How to deal with poor condition numbers? [2/3]

How to condition an unconstrained optimisation problem (cont.)

Recall the gradient method of the transformed problem:

$$\boldsymbol{y}^{(k+1)} = \boldsymbol{y}^{(k)} - \alpha_k \boldsymbol{S}^\top \nabla f(\boldsymbol{S}\boldsymbol{y}^{(k)}).$$

Multiplying this equality by $\boldsymbol{S}$ on the left, using the notation $\boldsymbol{x}_k = \boldsymbol{S}\boldsymbol{y}_k$, and defining $\boldsymbol{D} = \boldsymbol{S}\boldsymbol{S}^\top$:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \alpha_k \boldsymbol{S}\boldsymbol{S}^\top \nabla f(\boldsymbol{x}^{(k)}) = \boldsymbol{x}^{(k)} - \alpha_k \boldsymbol{D} \nabla f(\boldsymbol{x}^{(k)}),$$

we have the *scaled gradient method* with scaling matrix $\boldsymbol{D}$. By definition, $\boldsymbol{D}$ is positive definite. The direction $-\boldsymbol{D}\nabla f\left(\boldsymbol{x}^{(k)}\right)$ is a descent direction of f at $\boldsymbol{x}^{(k)}$ when $\nabla f\left(\boldsymbol{x}^{(k)}\right) \neq \boldsymbol{0}$ because $-\nabla f\left(\boldsymbol{x}^{(k)}\right)^\top \boldsymbol{D} \nabla f\left(\boldsymbol{x}^{(k)}\right) < 0$.

How to deal with poor condition numbers? [3/3]

How to condition an unconstrained optimisation problem (cont.)

The scaled gradient method with scaling matrix $\boldsymbol{D}$ is equivalent to the gradient method on the function $g(\boldsymbol{y}) = f(\boldsymbol{D}^{1/2}\boldsymbol{x})$. The gradient and Hessian of g where $\boldsymbol{x} = \boldsymbol{D}^{1/2}\boldsymbol{y}$:

$$\nabla g(\boldsymbol{y}) = \boldsymbol{D}^{1/2}\nabla f(\boldsymbol{D}^{1/2}\boldsymbol{y}) = \boldsymbol{D}^{1/2}\nabla f(\boldsymbol{x}),$$
$$\nabla^2 g(\boldsymbol{y}) = \boldsymbol{D}^{1/2}\nabla^2 f(\boldsymbol{D}^{1/2}\boldsymbol{y})\boldsymbol{D}^{1/2} = \boldsymbol{D}^{1/2}\nabla^2 f(\boldsymbol{x})\boldsymbol{D}^{1/2}.$$

Scaled Gradient Method

Input: $\epsilon \Rightarrow$ tolerance Parameter. **Initialisation**: Pick $\boldsymbol{x}^{(0)} \in \mathbb{R}^n$ arbitrarily.
General step: For any $k = 0, 1, 2, \ldots$ execute the following steps:

- Pick a scaling matrix $\boldsymbol{D}_k \succ \boldsymbol{0}$.
- Pick a stepsize α_k by line search: $h(\alpha) = f\left(\boldsymbol{x}^{(k)} - \alpha\boldsymbol{D}_k\nabla f\left(\boldsymbol{x}^{(k)}\right)\right)$.
- Set $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \alpha_k\boldsymbol{D}_k\nabla f\left(\boldsymbol{x}^{(k)}\right)$. If $\left\|\nabla f\left(\boldsymbol{x}^{(k)}\right)\right\|_2 \leq \epsilon$, STOP.

How to choose the scaling matrix?

Want the scaled Hessian as close as possible to the identity matrix

To accelerate the rate of convergence, which depends on the scaled Hessian $\boldsymbol{D}_k^{1/2}\nabla^2 f(\boldsymbol{x}^{(k)})\boldsymbol{D}_k^{1/2}$, the scaling matrix is often chosen to make the scaled Hessian as close as possible to the identity matrix.

Sanity Check

If $\nabla^2 f(\boldsymbol{x}^{(k)}) \succ \mathbf{0}$, what is a good choice for $\boldsymbol{D}_k$?

Chose $\boldsymbol{D}_k$ to get Newton's method!

We can chose $\boldsymbol{D}_k = \left(\nabla^2 f(\boldsymbol{x}^{(k)})\right)^{-1}$ and the scaled Hessian becomes the identity matrix! The result is Newton's method:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \alpha_k \left(\nabla^2 f(\boldsymbol{x}^{(k)})\right)^{-1} \nabla f\left(\boldsymbol{x}^{(k)}\right)$$

Could be computationally expensive. Consider a diagonal scaling matrix!

Summary