Modern Optimization Techniques

2. Unconstrained Optimization / 2.1. Gradient Descent

Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL)
Institute of Computer Science
University of Hildesheim, Germany

Syllabus

Mon. 30.10.	(0)	0. Overview
		1. Theory
Mon. 6.11.	(1)	1. Convex Sets and Functions
		2. Unconstrained Optimization
Mon. 13.11.	(2)	2.1 Gradient Descent
Mon. 20.11.	(3)	2.2 Stochastic Gradient Descent
Mon. 27.11.	(4)	2.3 Newton's Method
Mon. 4.12.	(5)	2.4 Quasi-Newton Methods
Mon. 11.12.	(6)	2.5 Subgradient Methods
Mon. 18.12.	(7)	2.6 Coordinate Descent
	—	— *Christmas Break* —
		3. Equality Constrained Optimization
Mon. 8.1.	(8)	3.1 Duality
Mon. 15.1.	(9)	3.2 Methods
		4. Inequality Constrained Optimization
Mon. 22.1.	(10)	4.1 Primal Methods
Mon. 29.1.	(11)	4.2 Barrier and Penalty Methods
Mon. 5.2.	(12)	4.3 Cutting Plane Methods

Outline

Outline

Unconstrained Convex Optimization Problem

$$\arg\min_{x \in \mathbb{R}^N} \; f(\mathbf{x})$$

where

- $f : X \to \mathbb{R}, X \subseteq \mathbb{R}^N$ is
  - convex
  - twice continuously differentiable
  - esp. $\operatorname{dom} f = X = \mathbb{R}^N$ or open.

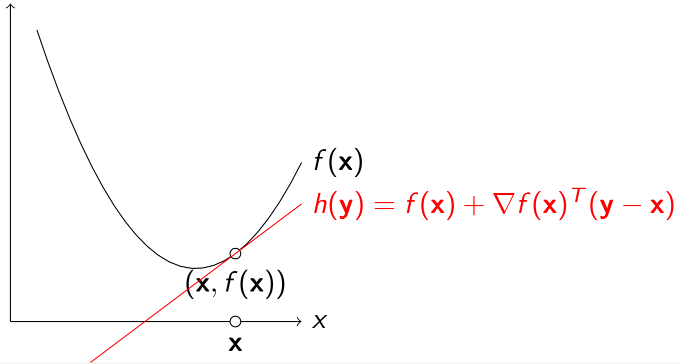
- An optimal $\mathbf{x}^*$ exists and $p^* := f(\mathbf{x}^*)$ is finite

Reminder: 1st-order condition

1st-order condition: a differentiable function f is convex iff

- dom f is a convex set
- for all $\mathbf{x}, \mathbf{y} \in \text{dom } f$

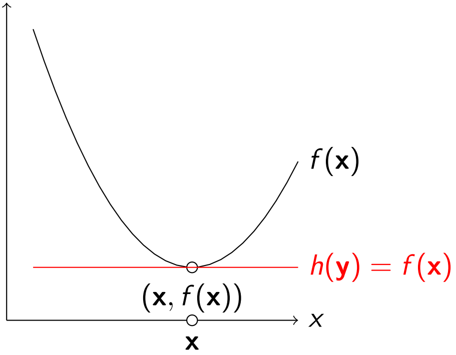
$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x})$$



Optimality condition

$\mathbf{x}$ is optimal iff

$$\nabla f(\mathbf{x}) = 0$$



Methods for Unconstrained Optimization

- Start with an initial point: $\mathbf{x}^{(0)}$
- Generate a sequence of points: $\mathbf{x}^{(k)}$ with

$$f(\mathbf{x}^{(k)}) \rightarrow f(\mathbf{x}^*)$$

```
min-unconstrained(f, x^(0)):
  k := 0
  repeat
    x^(k+1) := next-point(f, x^(k))
    k := k + 1
  until converged(x^(k), x^(k-1), f)
  return x^(k), f(x^(k))
```

Methods for Unconstrained Optimization

- Start with an initial point: $\mathbf{x}^{(0)}$
- Generate a sequence of points: $\mathbf{x}^{(k)}$ with

$$f(\mathbf{x}^{(k)}) \rightarrow f(\mathbf{x}^*)$$

```
min-unconstrained(f, x^(0), k^max):
  for k := 0 : k^max − 1:
    x^(k+1) := next-point(f, x^(k))
    if converged(x^(k+1), x^(k), f):
      return x^(k+1), f(x^(k+1))
  raise exception "not converged in k^max iterations"
```

Convergence Criterion

$$\textbf{converged}(\mathbf{x}^{(k+1)}, \mathbf{x}^{(k)}, f)$$

- Different criteria in use
  - different optimization methods may use different criteria
- One would like to use the **optimality gap**:

$$||\mathbf{x}^{(k+1)} - \mathbf{x}^{\star}||_2^2 < \epsilon$$

  - not possible as $\mathbf{x}^{\star}$ is unknown
- **Minimum progress/change ϵ in x in last iteration**:

$$\textbf{converged}(\mathbf{x}^{(k+1)}, \mathbf{x}^{(k)}, f) := ||\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}||_2^2 < \epsilon$$

  - cheap to compute
  - can be used with any method
  - requires parameter $\epsilon \in \mathbb{R}^+$

Outline

1. Unconstrained Optimization

2. Descent Methods

3. Gradient Descent

4. Line search

5. Convergence of Gradient Descent

6. Example: Linear Ridge Regression via Gradient Descent

Descent Methods

- A class/template of methods

- The next point is generated as:

$$\mathbf{x}^{(k+1)} := \mathbf{x}^{(k)} + \mu \Delta \mathbf{x}^{(k)}$$

  with
  - a **search direction** $\Delta \mathbf{x}^{(k)}$ and
  - a **step size** μ such that

$$f(\mathbf{x}^{(k)} + \mu \Delta \mathbf{x}^{(k)}) < f(\mathbf{x}^{(k)})$$

- Specific descent methods differ in how they compute the search direction $\Delta \mathbf{x}^{(k)}$
  - Gradient Descent
  - Steepest Descent
  - Newton's Method

Descent Methods

min-descent$(f, \mathbf{x}^{(0)}, k^{\max})$:
  for $k := 0 : k^{\max} - 1$:
    $\Delta x^{(k)} :=$ **search-direction**$(f, x^{(k)})$
    $\mu^{(k)} :=$ **step-size**$(f, x^{(k)}, \Delta x^{(k)})$
    $\mathbf{x}^{(k+1)} := x^{(k)} + \mu^{(k)} \Delta x^{(k)}$
    if **converged**$(\mathbf{x}^{(k+1)}, \mathbf{x}^{(k)}, f)$:
      return $\mathbf{x}^{(k+1)}$, $f(\mathbf{x}^{(k+1)})$
  raise exception "not converged in $k^{\max}$ iterations"

Computing the Step Size

The step size can be computed in various ways:

- constant value
- line search
- various heuristics depending on the specific algorithm

Outline

Gradient Descent

- The gradient of a function $f : X \to \mathbb{R}, X \subseteq \mathbb{R}^N$ at $\mathbf{x}$ yields the direction in which the function is maximally growing locally.

- Gradient Descent is a descent method that searches in the opposite direction of the gradient:

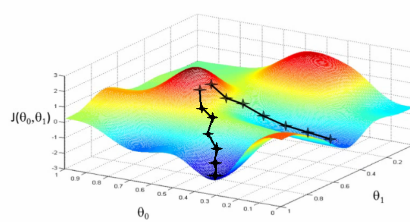
$$\Delta \mathbf{x} := -\nabla f(\mathbf{x})$$

- Gradient:

$$\nabla f(\mathbf{x}) := \nabla_x f(\mathbf{x}) := (\frac{\partial f}{\partial x_n}(\mathbf{x}))_{n=1:N}$$

Gradient Descent

```
min-GD(f, x^(0), k^max):
    for k := 0 : k^max − 1:
        Δx^(k) := −∇f(x^(k))
        μ^(k) := step-size(f, x^(k), Δx^(k))
        x^(k+1) := x^(k) + μ^(k) Δx^(k)
        if converged(x^(k+1), x^(k), f):
            return x^(k+1), f(x^(k+1))
    raise exception "not converged in k^max iterations
```



Gradient Descent / Implementations

- for analysis usually all updated variables are indexed

$$x^{(k)}, \Delta\mathbf{x}^{(k)}, \mu^{(k)}$$

- in implementations, one usually does only need one copy
  - or two, to compare against the last one

min-GD$(f, \mathbf{x}, k^{\max})$:
  for $k := 0 : k^{\max} - 1$:
    $\Delta x := -\nabla f(x)$
    $\mu :=$ **step-size**$(f, x, \Delta x)$
    $\mathbf{x}^{\text{old}} := x$
    $\mathbf{x} := x^{\text{old}} + \mu \Delta x$
    if **converged**$(\mathbf{x}, \mathbf{x}^{\text{old}}, f)$:
      return $\mathbf{x}$, $f(\mathbf{x})$
  raise exception "not converged in $k^{\max}$ iterations"

Gradient Descent / Considerations

- Stopping criterion: $||\nabla f(\mathbf{x})||_2 \leq \epsilon$

$$\begin{aligned}\textbf{converged}(\mathbf{x}, \mathbf{x}^{\text{old}}, f) &:= \\ \textbf{converged}(\nabla f(\mathbf{x})) &:= ||\nabla f(\mathbf{x})||_2 \leq \epsilon\end{aligned}$$

  - cheap to use as GD has to compute the gradient anyway

- GD is simple and straightforward

- GD has slow convergence
  - esp. compared to Newton's method

- Out-of-the-box, GD works only well for convex problems, otherwise will get stuck in local minima

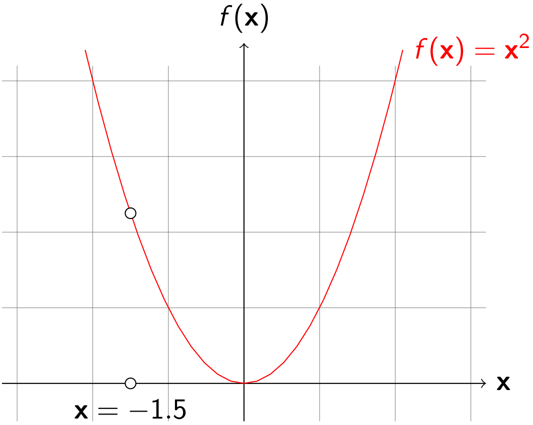
Gradient Descent Example

Task:

$$\text{minimize} \quad \mathbf{x}^2$$

- $\mu = 0.3$
- $-\nabla f(\mathbf{x}) = -2\mathbf{x}$

Initial point: $\mathbf{x}^0 = -1.5$



Gradient Descent Example

Task:

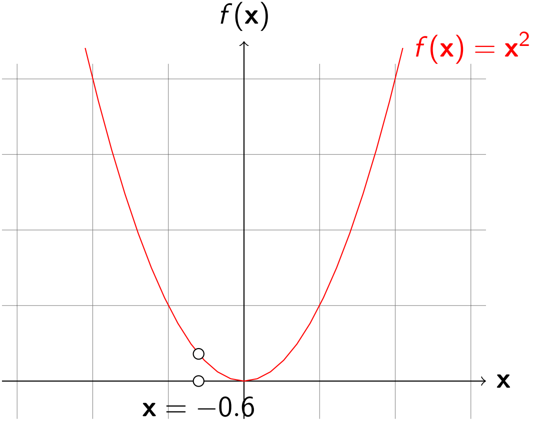
$$\text{minimize} \quad \mathbf{x}^2$$

- $\mu = 0.3$
- $-\nabla f(\mathbf{x}) = -2\mathbf{x}$

$\mathbf{x}^0 = -1.5$

$\mathbf{x} = -1.5 - 0.3 \cdot (2 \cdot -1.5)$

$\mathbf{x} = -0.6$



Gradient Descent Example

Task:

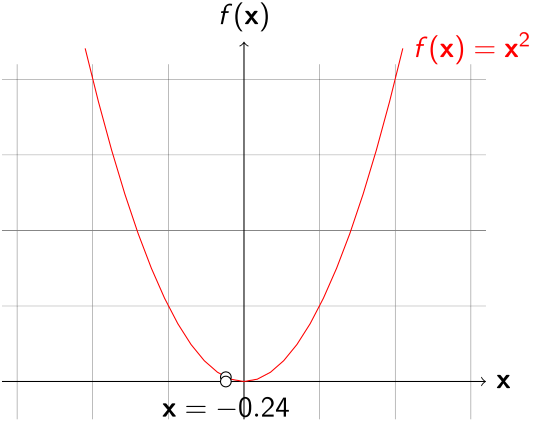
$$\text{minimize} \quad \mathbf{x}^2$$

- $\mu = 0.3$
- $-\nabla f(\mathbf{x}) = -2\mathbf{x}$

$\mathbf{x} = -0.6$

$\mathbf{x} = -0.6 - 0.3 \cdot (2 \cdot -0.6)$

$\mathbf{x} = -0.24$



Gradient Descent Example

Task:

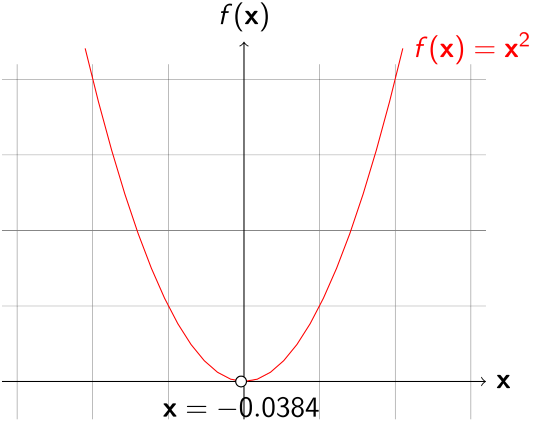
$$\text{minimize} \quad \mathbf{x}^2$$

- $\mu = 0.3$
- $-\nabla f(\mathbf{x}) = -2\mathbf{x}$

$\mathbf{x} = -0.24$

$\mathbf{x} = -0.24 - 0.3 \cdot (2 \cdot -0.24)$

$\mathbf{x} = -0.0384$



Gradient Descent Example

Task:

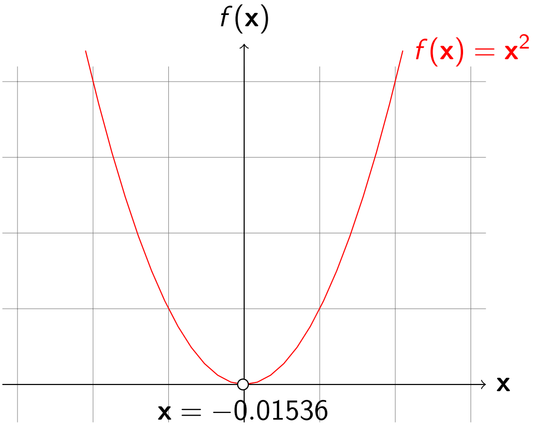
$$\text{minimize} \quad \mathbf{x}^2$$

- $\mu = 0.3$
- $-\nabla f(\mathbf{x}) = -2\mathbf{x}$

$\mathbf{x} = -0.0384$

$\mathbf{x} = -0.0384 - 0.3 \cdot (2 \cdot -0.0384)$

$\mathbf{x} = -0.01536$



Considerations about the Step Size

- Crucial for the convergence of the algorithm
- Step size too small $\rightsquigarrow$ slow convergence
- Step size too large $\rightsquigarrow$ divergence!

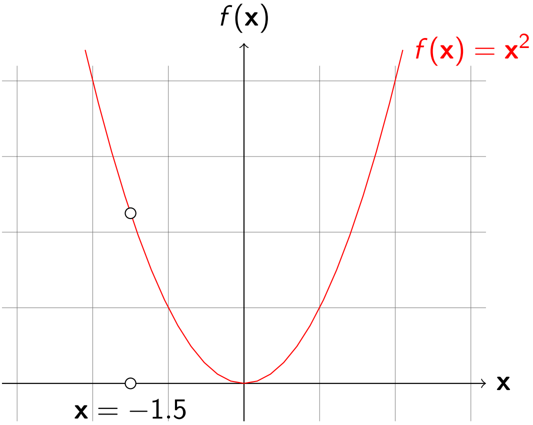
Gradient Descent Example - A perfect Step Size

Task:

$$\text{minimize} \quad \mathbf{x}^2$$

- $\mu = 0.5$
- $-\nabla f(\mathbf{x}) = -2\mathbf{x}$

Initial point: $\mathbf{x}^0 = -1.5$



Gradient Descent Example - A perfect Step Size

Task:

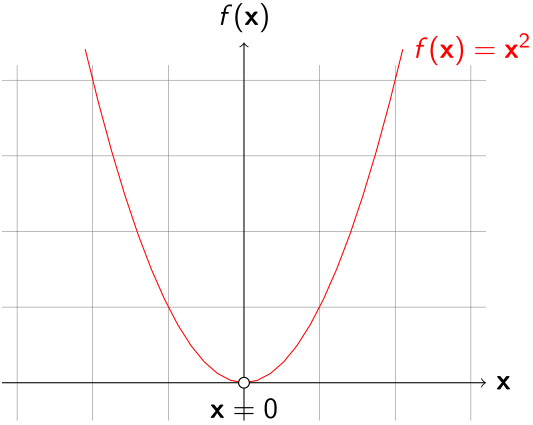
$$\text{minimize} \quad \mathbf{x}^2$$

- $\mu = 0.5$
- $-\nabla f(\mathbf{x}) = -2\mathbf{x}$

$\mathbf{x}^0 = -1.5$

$\mathbf{x} = -1.5 - 0.5 \cdot (2 \cdot -1.5)$

$\mathbf{x} = 0$



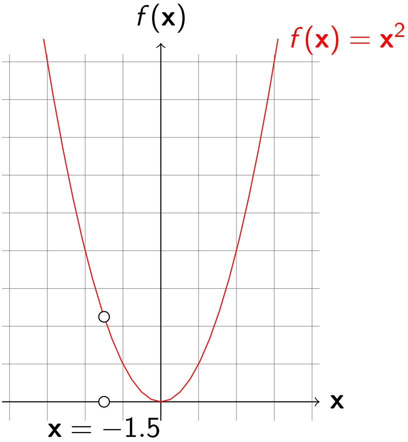
Gradient Descent Example - Too Large Step Size

Task:

$$\text{minimize} \quad \mathbf{x}^2$$

- $\mu = 1.5$
- $-\nabla f(\mathbf{x}) = -2\mathbf{x}$

Initial point: $\mathbf{x}^0 = -1.5$



Gradient Descent Example - Too Large Step Size

Task:

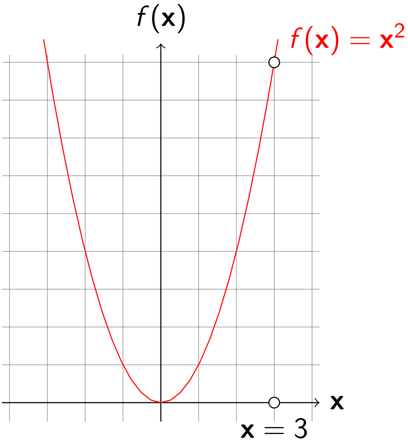
$$\text{minimize} \quad \mathbf{x}^2$$

- $\mu = 1.5$
- $-\nabla f(\mathbf{x}) = -2\mathbf{x}$

$\mathbf{x}^0 = -1.5$

$\mathbf{x} = -1.5 - 1.5 \cdot (2 \cdot -1.5)$

$\mathbf{x} = 3$



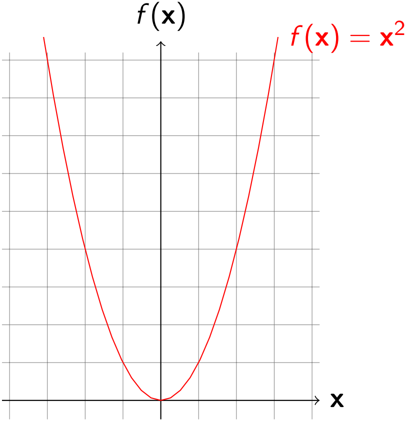
Gradient Descent Example - Too Large Step Size

Task:

$$\text{minimize} \quad \mathbf{x}^2$$

- $\mu = 1.5$
- $-\nabla f(\mathbf{x}) = -2\mathbf{x}$

$$\mathbf{x}^0 = 3$$
$$\mathbf{x} = 3 - 1.5 \cdot (2 \cdot 3)$$
$$\mathbf{x} = -6$$



Outline

Line search

- **line search** is the task to compute the step lenght in a descent algorithm.

- a one-dimensional optimization problem in μ:

$$\arg\min_{\mu \in \mathbb{R}^+} f(\mathbf{x} + \mu \Delta \mathbf{x})$$

Line Search Methods

- **exact line search**
  - Used if the problem can be solved analytically or with low cost
  - e.g., for **unconstrained quadratic optimization**:

$$\arg\min_{x \in \mathbb{R}^N} f(x) := \frac{1}{2} x^T A x + b^T x, \quad A \in \mathbb{R}^{N \times N} \text{ pos. def.}, b \in \mathbb{R}^N$$

Line Search Methods

- **exact line search**
  - Used if the problem can be solved analytically or with low cost
  - e.g., for **unconstrained quadratic optimization**:

$$\arg\min_{x \in \mathbb{R}^N} f(x) := \frac{1}{2} x^T A x + b^T x, \quad A \in \mathbb{R}^{N \times N} \text{ pos. def.}, b \in \mathbb{R}^N$$

- **backtracking line search**
  - only approximative
  - guarantees that the new function value is lower than a specific bound

Backtracking Line Search

stepsize-backtracking$(f, \mathbf{x}, \Delta\mathbf{x}, \alpha \in (0, 0.5), \beta \in (0,1))$:
  $\mu := 1$
  while $f(\mathbf{x} + \mu\Delta\mathbf{x}) > f(\mathbf{x}) + \alpha\mu\nabla f(\mathbf{x})^T\Delta\mathbf{x}$:
    $\mu := \beta\mu$
  return μ

Backtracking Line Search

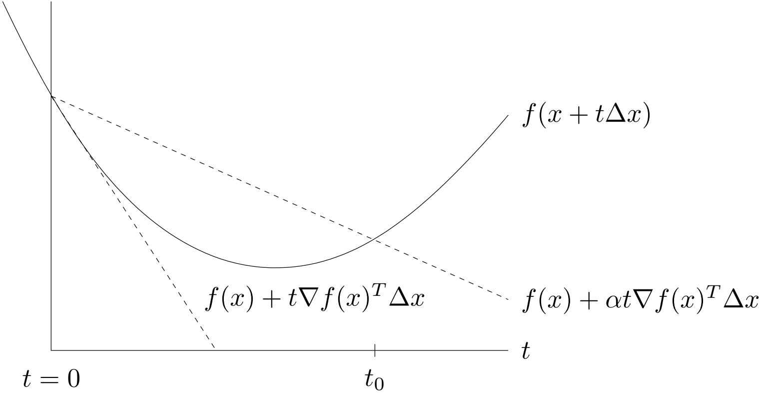
stepsize-backtracking$(f, \mathbf{x}, \Delta\mathbf{x}, \alpha \in (0, 0.5), \beta \in (0, 1))$:
  $\mu := 1$
  while $f(\mathbf{x} + \mu\Delta\mathbf{x}) > f(\mathbf{x}) + \alpha\mu\nabla f(\mathbf{x})^T\Delta\mathbf{x}$:
    $\mu := \beta\mu$
  return μ

Loop eventually terminates: for sufficient small μ:

$$f(x + \mu\Delta x) \approx f(x) + \mu\nabla f(x)^T\Delta x < f(x) + \alpha\mu\nabla f(x)^T\Delta x$$

as for a descent direction: $\nabla f(x)^T\Delta x < 0$

Backtracking Line Search



source: [Boyd and Vandenberghe, 2004, p. 465]

Outline

Sublevel Sets

sublevel set of $f : X \to \mathbb{R}, X \subseteq \mathbb{R}^N$ at level $\alpha \in \mathbb{R}$:

$$S_\alpha := \{x \in \operatorname{dom} f \mid f(x) \leq \alpha\}$$

Sublevel Sets

sublevel set of $f : X \to \mathbb{R}, X \subseteq \mathbb{R}^N$ at level $\alpha \in \mathbb{R}$:

$$S_\alpha := \{x \in \operatorname{dom} f \mid f(x) \leq \alpha\}$$

basic facts:

- if f is convex, then all its sublevel sets S_α are convex sets.
    - useful to show that a set is convex
        - show that it can be represented as a sublevel set of a convex function.

Closed Functions

$f : X \to \mathbb{R}, X \subseteq \mathbb{R}^N$ **closed** $:\Longleftrightarrow$ all its sublevel sets are closed.

Closed Functions

$f : X \to \mathbb{R}, X \subseteq \mathbb{R}^N$ **closed** $:\Longleftrightarrow$ all its sublevel sets are closed.

examples:

- $f(x) = x^2$ is closed.
- $f(x) = 1/x$ on $\mathbb{R}^+$ is closed.
- $f(x) = x \log x$ on $\mathbb{R}^+$ is not closed.
- but f on $\mathbb{R}_0^+$ defined by

$$f(x) := \begin{cases} x \log x, & \text{if } x > 0 \\ 0, & \text{else} \end{cases}$$

is closed.

Closed Functions

$f : X \rightarrow \mathbb{R}, X \subseteq \mathbb{R}^N$ **closed** $:\Longleftrightarrow$ all its sublevel sets are closed.

examples:

- $f(x) = x^2$ is closed.
- $f(x) = 1/x$ on $\mathbb{R}^+$ is closed.
- $f(x) = x \log x$ on $\mathbb{R}^+$ is not closed.
- but f on $\mathbb{R}_0^+$ defined by

$$f(x) := \begin{cases} x \log x, & \text{if } x > 0 \\ 0, & \text{else} \end{cases}$$

  is closed.

Classes of closed functions:

- continuous functions on all of $\mathbb{R}^N$
- continuous functions on an open set
  that go to infinity everywhere towards the border

Semidefinite Matrices II

Let $A, B \in \mathbb{R}^{N \times N}$ symmetric matrices:

$$A \succeq B :\Longleftrightarrow A - B \succeq 0$$

- $A \succeq mI, m \in \mathbb{R}^+$:
  - all eigenvalues of A are $\geq m$
- $A \preceq MI, M \in \mathbb{R}^+$:
  - all eigenvalues of A are $\leq M$

Strongly Convex Functions

Let $f : X \rightarrow \mathbb{R}, X \subseteq \mathbb{R}^N$ be twice continuously differentiable.

f is **strongly convex** $:\Longleftrightarrow$

- $\operatorname{dom} f = X$ is convex and
- the eigenvalues of the Hessian are uniformly bounded from below:

$$\nabla^2 f(x) \succeq mI, \quad \exists m \in \mathbb{R}^+ \ \forall x \in \operatorname{dom} f$$

Strongly Convex Functions

Let $f : X \to \mathbb{R}, X \subseteq \mathbb{R}^N$ be twice continuously differentiable.

f is **strongly convex** $:\Longleftrightarrow$

- $\operatorname{dom} f = X$ is convex and
- the eigenvalues of the Hessian are uniformly bounded from below:

$$\nabla^2 f(x) \succeq mI, \quad \exists m \in \mathbb{R}^+ \; \forall x \in \operatorname{dom} f$$

Every strongly convex function f is also strictly convex.

- but not the other way around
  - $f(x) = x^4$ on $\mathbb{R}^+$ is strictly, but not strongly convex
- do not confuse strongly and strictly convex!

Strongly Convex Functions / Basic Facts

(i) f is above a hyperbola:

$$f(y) \geq f(x) + \nabla f(x)^T(y-x) + \frac{m}{2}||y-x||_2^2$$
$$p^* \geq f(x) - \frac{1}{2m}||\nabla f(x)||_2^2$$

(ii) if f is closed and S one of its sublevel sets, then

a) the eigenvalues of the Hessian are also uniformly bounded from above on S:

$$\nabla^2 f(x) \preceq MI, \quad \exists M \in \mathbb{R}^+ \ \forall x \in S$$

b)

$$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \frac{M}{2}||y-x||_2^2, \quad x, y \in S$$
$$p^* \leq f(x) - \frac{1}{2M}||\nabla f(x)||_2^2$$

Strongly Convex Functions / Basic Facts / Proofs

(i) for $x, y \in \text{dom}\, f \;\exists z \in [x, y]$
(Taylor expansion with Lagrange mean value remainder):

$$f(y) = f(x) + \nabla f(x)^T (y - x) + \frac{1}{2} \underbrace{(y-x)^T \nabla^2 f(z)(y-x)}_{\geq m||y-x||_2^2}$$

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{m}{2}||y-x||_2^2$$

$$\geq \min_y f(x) + \nabla f(x)^T (y - x) + \frac{m}{2}||y-x||_2^2$$

considered as function in y has

minimum at $\tilde{y} := x - \frac{1}{m}\nabla f(x)$

$$= f(x) + \nabla f(x)^T (\tilde{y} - x) + \frac{m}{2}||\tilde{y}-x||_2^2$$

$$= f(x) - \frac{1}{2m}||\nabla f(x)||_2^2$$

$$\rightsquigarrow \; p^* = f(y = x^*) \geq f(x) - \frac{1}{2m}||\nabla f(x)||_2^2$$

Strongly Convex Functions / Basic Facts / Proofs (2/2)

(ii.a)
- due to (i) all sublevel sets are bounded
- the maximal eigenvalue of $\nabla^2 f(x)$ is a continuous function on a closed bounded set and thus itself bounded,
  - i.e., it exists $M \in \mathbb{R}^+$: $\nabla^2 f(x) \preceq MI$

(ii.b) as for (i), using (ii.a)

Convergence of Gradient Descent / Exact Line Search

If

- f is strongly convex,
- the initial sublevel set $S := \{x \in \text{dom}\, f \mid f(x) \leq f(x^{(0)})\}$ is closed,
- an exact line search is used,

then

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

Equivalently, to guarantee $f(x^{(k)}) - p^* \leq \epsilon$, GD requires

$$k := \frac{\log \frac{f(x^0) - p^*}{\epsilon}}{\log \frac{1}{1 - \frac{m}{M}}} \quad \text{iterations.}$$

Especially,

- GD converges, i.e., $f(x^{(k)})$ approaches p^*
- the convergence is exponential in k (with basis $c := 1 - \frac{m}{M}$)
  - called **linear convergence** in the optimization literature

Convergence of Gradient Descent / Proof

$$\tilde{f}(t) := f(x - t\nabla f(x)), \quad t \in \{t \in \mathbb{R}_0^+ \mid x - t\nabla f(x) \in S\}$$

$$\begin{aligned}
f(x^{\text{next}}) &= \tilde{f}(t_{\text{exact}}) \\
&\leq \tilde{f}(0) - \frac{1}{2M}||\nabla \tilde{f}(0)||_2^2, && \tilde{f} \text{ strongly convex (ii.b)} \\
&= f(x) - \frac{1}{2M}\underbrace{||\nabla f(x)||_2^2}_{\geq 2m(f(x)-p^*)}, && f \text{ strongly convex (i)}
\end{aligned}$$

$$f(x^{\text{next}}) - p^* \leq f(x) - p^* - \frac{1}{2M}2m(f(x) - p^*) = (1 - \frac{m}{M})(f(x) - p^*)$$

$$f(x^{(k)}) - p^* \leq (1 - \frac{m}{M})^k (f(x^{(0)}) - p^*)$$

Convergence of Gradient Descent / Backtracking

If

- f is strongly convex,
- the initial sublevel set $S := \{x \in \operatorname{dom} f \mid f(x) \leq f(x^{(0)})\}$ is closed, and
- a backtracking line search is used,

then

$$f(x^{(k)}) - p^* \leq c^k \, (f(x^{(0)}) - p^*), \quad c := 1 - \min\{2\alpha m, 2\beta\alpha m/M\}$$

Equivalently, to guarantee $f(x^{(k)}) - p^* \leq \epsilon$, GD requires

$$k := \frac{\log \frac{f(x^0) - p^*}{\epsilon}}{\log \frac{1}{c}} \quad \text{iterations.}$$

Especially,

- GD converges, i.e., $f(x^{(k)})$ approaches p^*
- the convergence is exponential in k (with basis c; linear convergence)

Outline

A More practical example

We do not want to always minimize parabolas so let us discuss a more practical example:

Linear Regression!

- have m many data instances $\mathbf{a} \in \mathbb{R}^n$ with n many features / predictors
- want to learn a linear model parametrized by a vector $\beta \in \mathbb{R}^n$ to predict a real value $y \in \mathbb{R}$

Practical Example: Household Spending

If we have data about m households, we can represent it as:

$$A_{m,n} = \begin{pmatrix} 1 & a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ 1 & a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & a_{m,1} & a_{m,2} & a_{m,3} & a_{m,4} \end{pmatrix} \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

We can model the household consumption is a linear combination of the household features with parameters β:

$$\hat{y}_i = \beta^T \mathbf{a_i} = \beta_0 1 + \beta_1 a_{i,1} + \beta_2 a_{i,2} + \beta_3 a_{i,3} + \beta_4 a_{i,4}$$

Practical Example: Household Spending

We have:

$$\begin{pmatrix} 1 & a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ 1 & a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & a_{m,1} & a_{m,2} & a_{m,3} & a_{m,4} \end{pmatrix} \cdot \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{pmatrix} \approx \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

We want to find parameters β such that the measured error of the predictions is minimal:

$$\sum_{i=1}^{m} (\beta^T \mathbf{a_i} - y_i)^2 + \lambda \sum_{j=1}^{n} \beta_j^2 = \|A\beta - y\|_2^2 + \lambda \|\beta\|_2^2$$

Linear Regression

Let us look at the function to optimize:

$$\mathcal{L}(\beta, A, y) + \lambda \text{Reg}(\beta) = \sum_{i=1}^{m} (\beta^\top a_i - y_i)^2 + \lambda \|\beta\|_2^2$$

$$= \sum_{i=1}^{m} \left(\sum_{j=1}^{n} \beta_j a_{ij} - y_i \right)^2 + \lambda \sum_{j=1}^{n} \beta_j^2$$

Then we can compute the gradient component wise:

$$\frac{\partial}{\partial \beta_k} \mathcal{L}(\beta, A, y) + \lambda \text{Reg}(\beta) = \frac{\partial}{\partial \beta_k} \sum_{i=1}^{m} (\sum_{j=1}^{n} \beta_j a_{ij} - y_i)^2 + \lambda \sum_{j=1}^{n} \beta_j^2$$

$$= \sum_{i=1}^{m} 2 \cdot \left(\sum_{j=1}^{n} \beta_j a_{ij} - y_i \right) \cdot a_{ik} + 2\lambda \beta_k$$

Linear Regression

We obtain the update for every component of β as

$$\begin{aligned}\beta_k^{(k+1)} &= \beta_k^{(k)} - \mu \nabla_\beta (\mathcal{L}(\beta, A, y) + \lambda \text{Reg}(\beta)) \\ &= \beta_k^{(k)} - \mu \left(2 \sum_{i=1}^{m} \cdot \left(\sum_{j=1}^{n} \beta_j a_{ij} - y_i \right) \cdot a_{ik} + 2\lambda \beta_k^{(k)} \right)\end{aligned}$$

- see that $\left(\sum_{j=1}^{n} \beta_j a_{ij} - y_i\right)$ is actually the error of the model on the i-th instance
- error is the same for all k, can be precomputed

Linear Regression

procedure LEARN LINEAR REGRESSION MODEL
input: Data A, Labels y, inital parameters β^0, Step Size μ, Regularization constant λ, precision ϵ
  repeat
    Compute Error: $e_i = \left(\sum_{j=1}^{n} \beta_j a_{ij} - y_i\right)$
    for $k = 1, \ldots, n$ **do**
      $\beta_k^{(k+1)} = \beta_k^{(k)} - \mu \left(\sum_{i=1}^{m} e_i a_{ik} + \lambda \beta_k^{(k)}\right)$
    end for
    $t = t + 1$
  until $\|\nabla_\beta \mathcal{L}(\beta, A, y)\|_2^2 \leq \epsilon$
  return β, $\mathcal{L}(\beta, A, y)$
end procedure

Summary (1/2)

- **Unconstrained optimization** is the minimization of a function over all of $\mathbb{R}^N$ or an open subset $X \subseteq \mathbb{R}^N$.
  - In **Unconstrained convex optimization** X also has to be convex (and f, too).
- **Descent methods** iteratively find a next iterate $x^{(k+1)}$ with lower function value than the last iterate and require:
  - **search direction**: in which direction to search.
    - **Gradient Descent** (GD): negative gradient of the target function
  - **step length**: how far to go.
  - **convergence criterion**: when to stop.
    - small last step
    - small gradient

Summary (2/2)

- step length (aka **line search**) in rare cases can be computed exactly.
  - one-dimensional optimization problem (**exact line search**)
- **backtracking line search**:
  - Choose the largest stepsize that guarantees a decrease in function value.
  - guaranteed to terminate
- GD has **linear convergence**
  - exponential in the number of steps
    - with basis $1 - m/M$
      for smallest/largest eigenvalues m,M of the Hessian
  - if f is strongly convex, its initial sublevel set closed and exact line search is used.

Further Readings

- Unconstrained minimization problems:
  - Boyd and Vandenberghe [2004], chapter 9.1
- Descent methods:
  - Boyd and Vandenberghe [2004], chapter 9.2
- Gradient descent:
  - Boyd and Vandenberghe [2004], chapter 9.3
- also accessible from here:
  - steepest descent — Boyd and Vandenberghe [2004], chapter 9.4

References I

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge Univ Press, 2004.