TECHNISCHE UNIVERSITÄT DARMSTADT

Statistical Machine Learning

Lecture 04: Optimization Refresher

Kristian Kersting

TU Darmstadt

Summer Term 2020

Today's Objectives

- Make you remember Calculus and teach you advanced topics!
- Brute Force right through optimization!
- Covered Topics:
  - Unconstrained Optimization
  - Lagrangian Optimization
  - Numerical Methods (Gradient Descent)
- Go deeper?
  - Take the Optimization Class of Prof. von Stryk / SIM!
  - Read *Convex Optimization* by Boyd & Vandenberghe - http://www.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

Outline

1. Motivation

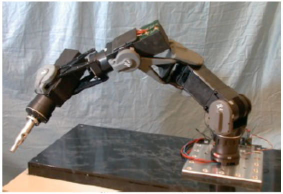
"All learning problems are essentially optimization problems on data."

Christopher G. Atkeson, Professor at CMU

Robot Arm

- You want to predict the torques of a robot arm

$$
\begin{aligned}
y &= l\ddot{q} - \mu\dot{q} + mlg\sin(q) \\
&= \begin{bmatrix} \ddot{q} & \dot{q} & \sin(q) \end{bmatrix} \begin{bmatrix} l & -\mu & mlg \end{bmatrix}^{\intercal} \\
&= \phi(\mathbf{x})^{\intercal}\boldsymbol{\theta}
\end{aligned}
$$



- Can we do this with a data set?

$$\mathcal{D} = \{(\mathbf{x}_i, y_i) \,|\, i = 1 \cdots n\}$$

- Yes, by minimizing the sum of the squared error
  $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, \mathcal{D}) = \sum_{i=1}^{n} \left(y_i - \phi(\mathbf{x}_i)^{\intercal}\boldsymbol{\theta}\right)^2$

Carl Friedrich Gauss
(1777–1855)

- Note that this is just one way to measure an error...

TECHNISCHE UNIVERSITÄT DARMSTADT

Will the previous method work?

- Sure! But the solution may be faulty, e.g., $m = -1\text{kg}, \ldots$
- Hence, we need to ensure some extra conditions, and our problem results in a constrained optimization problem

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad J(\boldsymbol{\theta}, \mathcal{D}) = & \sum_{i=1}^{n} \left(y_i - \phi(\mathbf{x}_i)^{\intercal} \boldsymbol{\theta}\right)^2 \\ s.t. \quad g(\boldsymbol{\theta}, \mathcal{D}) \geq & \; 0 \end{aligned}$$

  - where $g(\boldsymbol{\theta}, \mathcal{D}) = \begin{bmatrix} \theta_1 & -\theta_2 \end{bmatrix}^{\intercal}$

Motivation

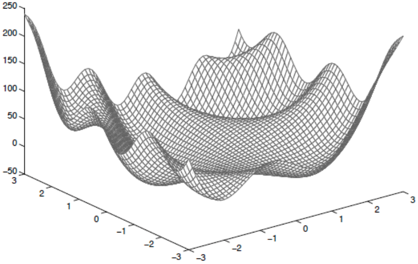
- ALL learning problems are optimization problems
- In any learning system, we have
  1. Parameters $\boldsymbol{\theta}$ to enable learning
  2. Data set $\mathcal{D}$ to learn from
  3. A cost function $J(\boldsymbol{\theta}, \mathcal{D})$ to measure our performance
  4. Some assumptions on the data, with equality and inequality constraints, $f(\boldsymbol{\theta}, \mathcal{D}) = 0$ and $g(\boldsymbol{\theta}, \mathcal{D}) > 0$
- How can we solve such problems in general?

Optimization problems in Machine Learning

Problem	Example Cost Functions	Resulting Method
Classification	$\min_\theta \sum_{i=1}^n \log\left(1+\exp\left(-y_i \mathbf{x}_i^T \theta\right)\right)$	Logistic Regression[3]
	$\min_{\theta_1,\theta_2} \sum_{i=1}^n \left(y_i - \mathrm{g}\left(\theta_2^T \mathrm{g}\left(\theta_1^T \mathbf{x}_i\right)\right)\right)^2$	Neural Networks Classification
	$\min_\theta \left\Vert\theta\right\Vert^2 + C\sum_{i=1}^n \xi_i$ s.t. $\xi_i - (1 - y_i \mathbf{x}_i^T \theta) \geq 0$ $\xi_i \geq 0$	Support Vector Machines
Regression	$\min_\theta \sum_{i=1}^n (y_i - \phi(\mathbf{x}_i)^T \theta)^2$	Linear Regression
	$\min_{\theta_1,\theta_2,\theta_3} \sum_{i=1}^n \left(y_i - \theta_3^T \mathrm{g}\left(\theta_2^T \mathrm{g}\left(\theta_1^T \mathbf{x}_i\right)\right)\right)^2$	Neural Networks Regression
Density Estimation	$\max_\theta \sum_{i=1}^n \log p(\mathbf{x}_i \vert \theta)$	General Formulation
Clustering	$\min_{\mu_1,\ldots,\mu_k} \sum_{j=1}^k \sum_{i\in C_j} \left\Vert\mathbf{x}_i - \mu_i\right\Vert^2$	$k-$means

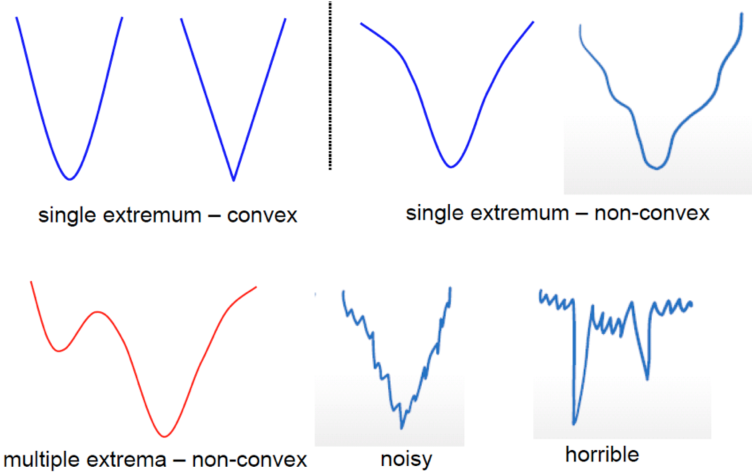
Machine Learning tells us how to come up with data-based cost functions such that optimization can solve them!

Most Cost Functions are Useless



Good Machine Learning tells us how to come up with data-based cost functions such that optimization can solve them efficiently!

Good cost functions should be Convex



Ideally, the Cost Functions should be Convex!

TECHNISCHE UNIVERSITÄT DARMSTADT

Outline

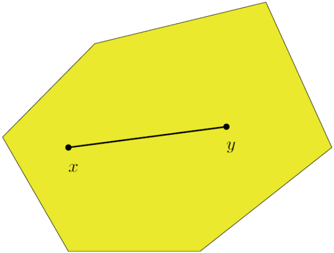
TECHNISCHE UNIVERSITÄT DARMSTADT

Convex Sets

A set $C \subseteq \mathbb{R}^n$ is convex if $\forall \mathbf{x}, \mathbf{y} \in C$ and $\forall \alpha \in [0, 1]$

$$\alpha \mathbf{x} + (1 - \alpha) \mathbf{y} \in C$$

This is the equation of the *line segment* between $\mathbf{x}$ and $\mathbf{y}$. I.e., for a given α, the point $\alpha \mathbf{x} + (1 - \alpha) \mathbf{y}$ lies in the line segment between $\mathbf{x}$ and $\mathbf{y}$



TECHNISCHE UNIVERSITÄT DARMSTADT

Examples of Convex Sets

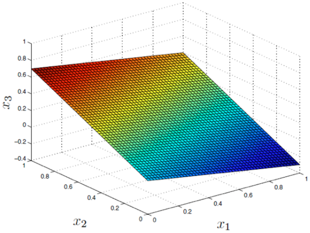
- **All of** $\mathbb{R}^n$ (obvious)
- **Non-negative orthant**: $\mathbb{R}^n_+$. Let $x \succeq 0, y \succeq 0$, clearly $\alpha \mathbf{x} + (1-\alpha)\,\mathbf{y} \succeq 0$
- **Norm balls**. Let $\|\mathbf{x}\| \leq 1, \|\mathbf{y}\| \leq 1$, then

$$\begin{aligned} \|\alpha \mathbf{x} + (1-\alpha)\,\mathbf{y}\| &\leq \|\alpha \mathbf{x}\| + \|(1-\alpha)\,\mathbf{y}\| \\ &= \alpha\,\|\mathbf{x}\| + (1-\alpha)\,\|\mathbf{y}\| \\ &\leq 1 \end{aligned}$$

Examples of Convex Sets

- **Affine subspaces (linear manifold)**: $\mathbf{Ax} = \mathbf{b}$, $\mathbf{Ay} = \mathbf{b}$, then

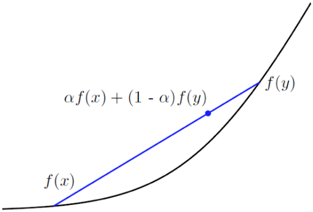
$$\begin{aligned}\mathbf{A}\left(\alpha\mathbf{x} + (1-\alpha)\,\mathbf{y}\right) &= \alpha\mathbf{Ax} + (1-\alpha)\,\mathbf{Ay}\\ &= \alpha\mathbf{b} + (1-\alpha)\,\mathbf{b}\\ &= \mathbf{b}\end{aligned}$$



Convex Functions

A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if $\forall \mathbf{x}, \mathbf{y} \in \mathrm{dom}\,(f)$ and $\forall \alpha \in [0, 1]$

$$f\left(\alpha \mathbf{x} + (1 - \alpha)\,\mathbf{y}\right) \leq \alpha f(\mathbf{x}) + (1 - \alpha) f\left(\mathbf{y}\right)$$



Examples of Convex Functions

- Linear/affine functions

$$f(\mathbf{x}) = \mathbf{b}^{\top}\mathbf{x} + c$$

- Quadratic functions

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\top}\mathbf{A}\mathbf{x} + \mathbf{b}^{\top}\mathbf{x} + c$$

where $\mathbf{A} \succeq 0$ (positive semidefinite matrix)

TECHNISCHE UNIVERSITÄT DARMSTADT

Examples of Convex Functions

- Norms (such as l_1 and l_2)

$$\|\alpha\mathbf{x} + (1-\alpha)\,\mathbf{y}\| \leq \|\alpha\mathbf{x}\| + \|(1-\alpha)\,\mathbf{y}\| = \alpha\,\|\mathbf{x}\| + (1-\alpha)\,\|\mathbf{y}\|$$

- Log-sum-exp (aka softmax, a smooth approximation to the maximum function often used on machine learning)

$$f(\mathbf{x}) = \log\left(\sum_{i=1}^{n} \exp\left(x_i\right)\right)$$

TECHNISCHE UNIVERSITÄT DARMSTADT

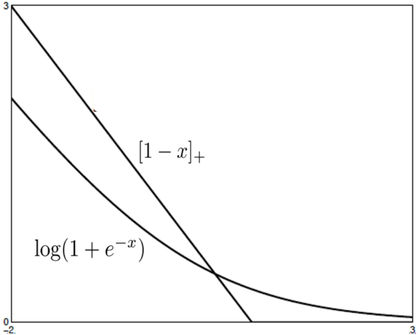
Important Convex Functions from Classification

- SVM loss

$$f(\mathbf{w}) = \left[1 - y_i x_i^\top \mathbf{w}\right]_+$$

- Binary logistic loss

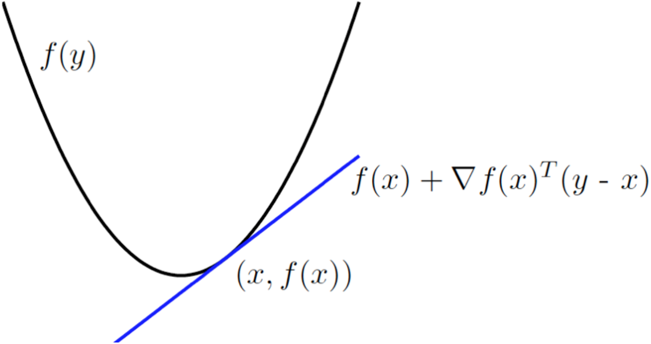
$$f(\mathbf{w}) = \log\left(1 + \exp\left(-y_i x_i^\top \mathbf{w}\right)\right)$$



First-Order Convexity Condition

- Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is *differentiable*. Then f is convex iff $\forall \mathbf{x}, \mathbf{y} \in \operatorname{dom}(f)$

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla_{\mathbf{x}} f(\mathbf{x})^{\intercal} (\mathbf{y} - \mathbf{x})$$

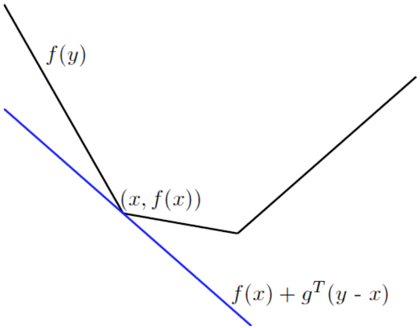


TECHNISCHE UNIVERSITÄT DARMSTADT

First-Order Convexity Condition - generally...

- The *subgradient*, or subdifferential set, $\partial f(\mathbf{x})$ of f at $\mathbf{x}$ is

$$\partial f(\mathbf{x}) = \{g : f(\mathbf{y}) \geq f(\mathbf{x}) + \mathbf{g}^\intercal (\mathbf{y} - \mathbf{x}), \forall \mathbf{y}\}$$



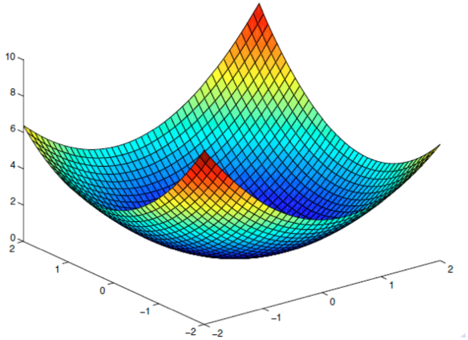
- Differentiability is not a requirement!

TECHNISCHE UNIVERSITÄT DARMSTADT

Second-Order Convexity Condition

- Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is *twice differentiable*. Then f is convex iff $\forall \mathbf{x} \in \text{dom}\,(f)$

$$\nabla^2_{\mathbf{x}} f(\mathbf{x}) \succeq 0$$

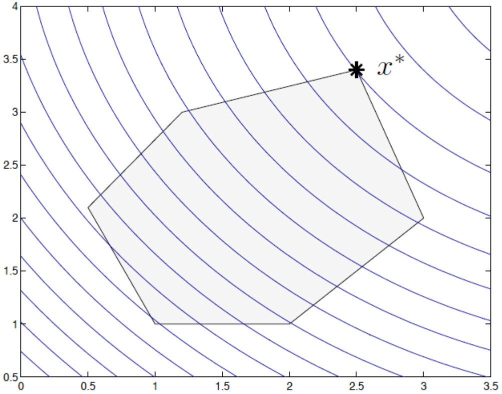


Ideal Machine Learning Cost Functions

$$
\begin{aligned}
\min_{\boldsymbol{\theta}} \quad J(\boldsymbol{\theta}, \mathcal{D}) &= 0 && \text{Convex Function} \\
\text{s.t.} \quad f(\boldsymbol{\theta}, \mathcal{D}) &= 0 && \text{Affine/Linear Function} \\
g(\boldsymbol{\theta}, \mathcal{D}) &\geq 0 && \text{Convex Set}
\end{aligned}
$$

Why are these conditions nice?

- Local solutions are globally optimal!
- Fast and well studied optimizers already exist for a long time!



TECHNISCHE UNIVERSITÄT DARMSTADT

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

Unconstrained optimization

Can you solve this problem?

$$\max_{\boldsymbol{\theta}} \quad J(\boldsymbol{\theta}) = \quad 1 - \theta_1^2 - \theta_2^2$$

With $\boldsymbol{\theta}^* = \begin{bmatrix} 0 & 0 \end{bmatrix}^\intercal$, $J^* = 1$
For any other $\boldsymbol{\theta} \neq \mathbf{0}$, $J < 1$

TECHNISCHE UNIVERSITÄT DARMSTADT

Constrained optimization

Can you solve this problem?

$$\begin{aligned} \max_{\boldsymbol{\theta}} \quad J(\boldsymbol{\theta}) = & \quad 1 - \theta_1^2 - \theta_2^2 \\ \text{s.t.} \quad f(\boldsymbol{\theta}) = & \quad \theta_1 + \theta_2 - 1 = 0 \end{aligned}$$

- First approach: convert the problem to an unconstrained problem
- Second approach: Lagrange Multipliers

TECHNISCHE UNIVERSITÄT DARMSTADT

Key Insight

- Taylor expansion around a vinicity of $\boldsymbol{\theta}_A$

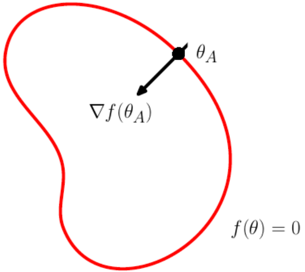
$$f(\boldsymbol{\theta}_A + \delta\boldsymbol{\theta}) \approx f(\boldsymbol{\theta}_A) + \delta\boldsymbol{\theta}^\intercal \nabla f(\boldsymbol{\theta}_A)$$

- With the constraint that the gradient is normal to the vinicity around $\boldsymbol{\theta}_A$

$$\delta\boldsymbol{\theta}^\intercal \nabla f(\boldsymbol{\theta}_A) = \mathbf{0}$$

- We have

$$f(\boldsymbol{\theta}_A + \delta\boldsymbol{\theta}) = f(\boldsymbol{\theta}_A)$$



TECHNISCHE UNIVERSITÄT DARMSTADT

Key Insight

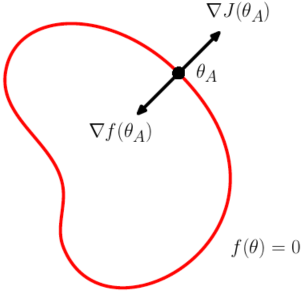
- We have to seek a point such that

$$\nabla J(\boldsymbol{\theta}) + \boldsymbol{\lambda}^{\intercal} \nabla f(\boldsymbol{\theta}) = \mathbf{0}$$

where $\boldsymbol{\lambda}$ are the Lagrange multipliers ($\delta\boldsymbol{\theta}$)

- Hence, we have the Langrangian function

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}) = J(\boldsymbol{\theta}) + \boldsymbol{\lambda}^{\intercal} f(\boldsymbol{\theta})$$



TECHNISCHE UNIVERSITÄT DARMSTADT

Back to our problem...

Can you solve this problem?

$$\begin{aligned} \max_{\boldsymbol{\theta}} \quad J(\boldsymbol{\theta}) = & \quad 1 - \theta_1^2 - \theta_2^2 \\ \text{s.t.} \quad f(\boldsymbol{\theta}) = & \quad \theta_1 + \theta_2 - 1 = 0 \end{aligned}$$

We can write the Lagrangian

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \left(1 - \theta_1^2 - \theta_2^2\right) + \lambda(\theta_1 + \theta_2 - 1)$$

TECHNISCHE UNIVERSITÄT DARMSTADT

The optimal solution

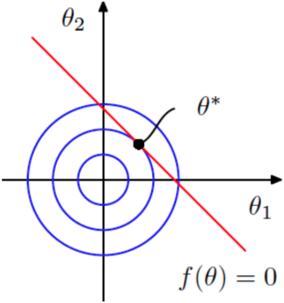
$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \left(1 - \theta_1^2 - \theta_2^2\right) + \lambda\left(\theta_1 + \theta_2 - 1\right)$$

$$\nabla_{\theta_1}\mathcal{L} = -2\theta_1 + \lambda = 0$$

$$\nabla_{\theta_2}\mathcal{L} = -2\theta_2 + \lambda = 0$$

$$\nabla_{\lambda}\mathcal{L} = \theta_1 + \theta_2 - 1 = 0$$

$$\theta_1^* = \theta_2^* = \tfrac{1}{2}\lambda^* = \tfrac{1}{2}$$



TECHNISCHE UNIVERSITÄT DARMSTADT

General Formulation

- For a problem written in the form

$$\begin{aligned} \max_{\boldsymbol{\theta}} \quad & J(\boldsymbol{\theta}) \\ \text{s.t.} \quad & \boldsymbol{f}(\boldsymbol{\theta}) = 0 \\ & \boldsymbol{g}(\boldsymbol{\theta}) \geq 0 \end{aligned}$$

- We have the Lagrangian

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = J(\boldsymbol{\theta}) + \boldsymbol{\lambda}^\intercal \boldsymbol{f}(\boldsymbol{\theta}) + \boldsymbol{\mu}^\intercal \boldsymbol{g}(\boldsymbol{\theta})$$

TECHNISCHE UNIVERSITÄT DARMSTADT

Langrangian Dual Formulation

- The Primal Problem, with corresponding primal variables $\boldsymbol{\theta}$ is

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad & J(\boldsymbol{\theta}) \\ \text{s.t.} \quad & g_i(\boldsymbol{\theta}) \leq 0 \quad \forall i = 1, \ldots, m \end{aligned}$$

  - Where each equality constraint can be converted into two equivalent inequality constraints ($f = 0 \equiv f \geq 0 \wedge f \leq 0$)

- Hence we have the Lagrangian $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}) = J(\boldsymbol{\theta}) + \boldsymbol{\lambda}^{\intercal} \boldsymbol{g}(\boldsymbol{\theta})$
- The Dual Problem[1], with corresponding dual variables $\boldsymbol{\lambda}$ is

$$\begin{aligned} \max_{\boldsymbol{\lambda}} \quad & G(\boldsymbol{\lambda}) = \max_{\boldsymbol{\lambda}} \min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}) \\ \text{s.t.} \quad & \lambda_i \geq 0 \quad \forall i = 1, \ldots, m \end{aligned}$$

[1] In words: Add the constraints to the objective function using nonnegative Lagrange multipliers. Then solve for the primal variables $\boldsymbol{\theta}$ that minimize this. The solution gives the primal variables $\boldsymbol{\lambda}$ as functions of the Lagrange multipliers. Now maximize this with respect to the dual variables under the derived constraints on the dual variables (including at least the nonnegativity constraints

TECHNISCHE UNIVERSITÄT DARMSTADT

Langrangian Dual Formulation

- Why maximization? If $\boldsymbol{\lambda}^*$ is the solution of the dual problem, then $G(\boldsymbol{\lambda}^*)$ is a lower bound for the primal problem due to two concepts:
  - Minimax inequality: for any function of two arguments $\phi(x, y)$, the *maximin* is less or equal than the *minimax*

$$\max_y \min_x \phi(x, y) \leq \min_x \max_y \phi(x, y)$$

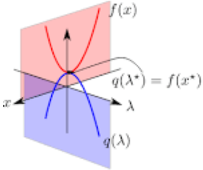
  - Weak duality: the primal values are always greater or equal than the dual values

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\lambda} \geq 0} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}) \geq \max_{\boldsymbol{\lambda} \geq 0} \min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda})$$

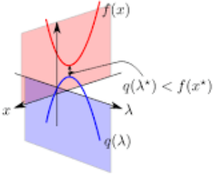
- Check Boyd, *Convex Optimization*, Ch. 5 for more detailed information.

Duality Gap and Strong Duality

- The duality gap is the difference between the values of any primal solutions and any dual solutions. It is always greater than or equal to 0, due to weak duality.
- The duality gap is zero if and only if strong duality holds.



strong duality



weak duality

TECHNISCHE UNIVERSITÄT DARMSTADT

Langrangian Dual Formulation

- Why do we care about the dual formulation?
  - $\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda})$ is an unconstrained problem, for a given $\boldsymbol{\lambda}$
  - If it is easy to solve, the overall problem is easy to solve, because $G(\boldsymbol{\lambda})$ is a concave function and thus easy to optimize, even though J and g_i may be nonconvex
- In ML, the dual is often more useful than the primal!

General Recipe to Solve Optimization Problems with the Langrangian Dual Formulation

TECHNISCHE UNIVERSITÄT DARMSTADT

- We want to solve

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad & J(\boldsymbol{\theta}) \\ \text{s.t.} \quad & g_i(\boldsymbol{\theta}) \leq 0 \quad \forall i = 1, \ldots, m \end{aligned}$$

- (Assume J and g_i are both differentiable functions)
- Write down the Lagrangian $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}) = J(\boldsymbol{\theta}) + \boldsymbol{\lambda}^\intercal \boldsymbol{g}(\boldsymbol{\theta})$
- Solve the problem $\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda})$
  - Differentiate $\mathcal{L}$ w.r.t. $\boldsymbol{\theta}$, set to zero, and write the solution $\boldsymbol{\theta}^*$ as a function of $\boldsymbol{\lambda}$
- Replace $\boldsymbol{\theta}^*$ back in the Lagrangian

$$G(\boldsymbol{\lambda}) = \mathcal{L}(\boldsymbol{\theta}^*, \boldsymbol{\lambda}) = J(\boldsymbol{\theta}^*) + \boldsymbol{\lambda}^\intercal \boldsymbol{g}(\boldsymbol{\theta}^*)$$

and solve the optimization problem $\max_{\boldsymbol{\lambda}} G(\boldsymbol{\lambda})$, s.t. $\forall \lambda_i \geq 0$

TECHNISCHE UNIVERSITÄT DARMSTADT

Outline

4. Numerical Optimization

- For some problems we do not know how to compute the solution analytically.
- What can we do in that situation?
- We solve it numerically using a computer!

TECHNISCHE UNIVERSITÄT DARMSTADT

Evaluation of Numerical Algorithms

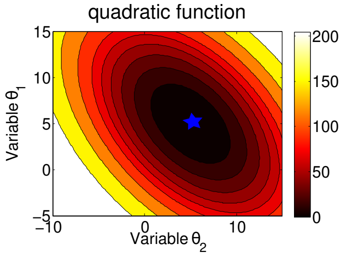
- The performance of different optimization algorithms can be measured by answering the following questions
    - Does the algorithm converge to the optimal solution?
    - How many steps does it take to converge?
    - Is the convergence smooth or bumpy?
    - Does it work for all types of functions or just on a special type (for instance convex functions)?
    - ...

TECHNISCHE UNIVERSITÄT DARMSTADT

Test Functions

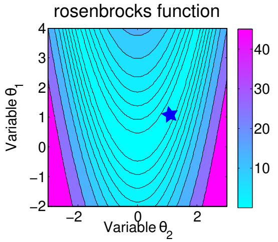
- To answer these questions we evaluated the performance in a set of well-known functions with interesting properties

Quadratic Function



$$J(\boldsymbol{\theta}) = (\theta_1 - 5)^2 + (\theta_1 - 5)(\theta_2 - 5) + (\theta_2 - 5)^2$$

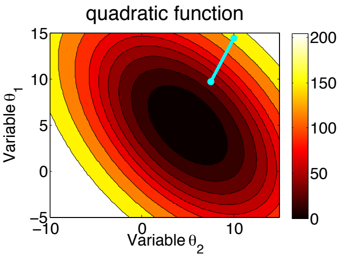
Rosenbrock Function



$$J(\boldsymbol{\theta}) = \left(\theta_2 - \theta_1^2\right)^2 + 0.01\,(1 - \theta_1)^2$$

TECHNISCHE UNIVERSITÄT DARMSTADT

Numerical Optimization - Key Ideas



- Find a $\delta\theta$ such that

$$J(\theta + \alpha\delta\theta) < J(\theta)$$

- Iterative update rules like

$$\theta_{n+1} = \theta_n + \alpha\delta\theta$$

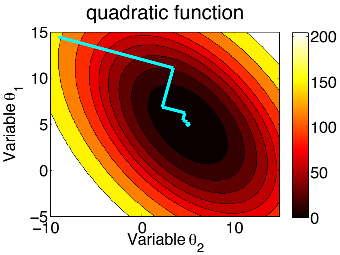
- Key questions: What is a good direction $\delta\theta$? What is a good step size α?

Line Search vs Constant Learning Rate

- Update rule: $\alpha_n = \arg\min_\alpha J(\theta_n + \alpha\delta\theta_n)$

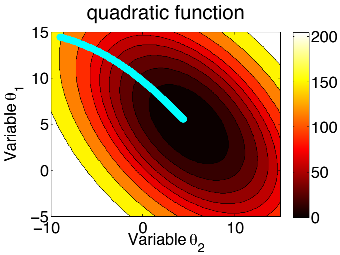
Optimal step size by Line Search

$$\alpha_n = \arg\min_\alpha J(\theta_n + \alpha\delta\theta_n)$$



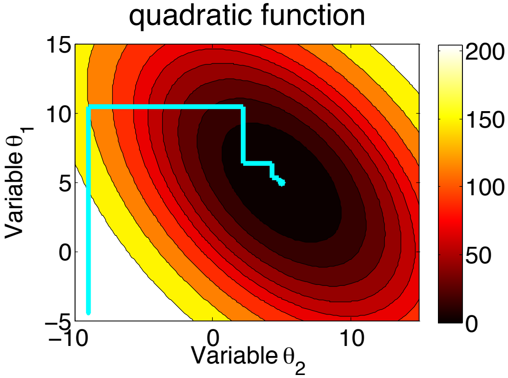
Other step sizes

$$\alpha_n = \textbf{const} \text{ or } \alpha_n = 1/n$$



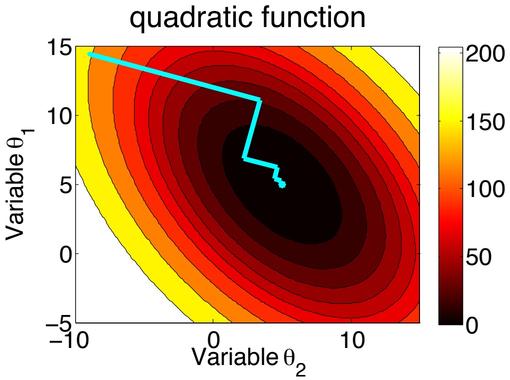
Method 1 - Axial Iteration (aka coordinate descent)

- Alternate minimization over both axes!



Method 2 - Steepest descent

- What you usually know as gradient descent
- Move in the direction of the gradient $\nabla J(\boldsymbol{\theta})$



TECHNISCHE UNIVERSITÄT DARMSTADT

Method 2 - Steepest descent

- The gradient is perpendicular to the contour lines
- After each line minimization the new gradient is always orthogonal to the previous step direction (true for any line minimization)
- Consequently, the iterations tend to zig-zag down the valley in a *very inefficient* manner

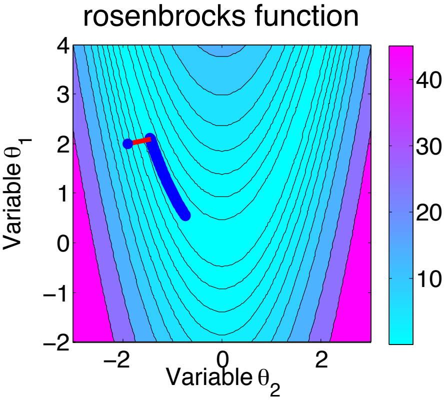
Method 2 - Steepest descent

- A very basic but cautious word for some source of errors
  - Remember that the gradient points in the direction of the maximum
  - Pay attention to the problem you're trying to solve!
    - $\max_\theta J(\theta)$, the update rule becomes $\theta \leftarrow \theta + \alpha \nabla_\theta J$
    - $\min_\theta J(\theta)$, the update rule becomes $\theta \leftarrow \theta - \alpha \nabla_\theta J$
    - With $\alpha > 0$

TECHNISCHE UNIVERSITÄT DARMSTADT

Steepest descent on the Rosenbrock function

- The algorithm crawls down the valley...



TECHNISCHE UNIVERSITÄT DARMSTADT

Method 3 - Newton's Method

- Taylor approximations can approximate functions locally. For instance:

$$\begin{aligned} J(\boldsymbol{\theta}+\delta\boldsymbol{\theta}) &\approx J(\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})^\intercal \delta\boldsymbol{\theta} + \frac{1}{2}\delta\boldsymbol{\theta}^\intercal \nabla^2_{\boldsymbol{\theta}} J(\boldsymbol{\theta})\,\delta\boldsymbol{\theta} \\ &= c + \mathbf{g}^\intercal \delta\boldsymbol{\theta} + \frac{1}{2}\delta\boldsymbol{\theta}^\intercal \mathbf{H}\delta\boldsymbol{\theta} \\ &= \tilde{J}(\delta\boldsymbol{\theta}) \end{aligned}$$

  - where **g** is the Jacobian and **H** is the Hessian

- We can minimize quadratic functions straightforwardly

$$\delta\boldsymbol{\theta} = \arg\min_{\delta\boldsymbol{\theta}} \tilde{J}(\delta\boldsymbol{\theta}) = \arg\min_{\delta\boldsymbol{\theta}} \left[c + \mathbf{g}^\intercal \delta\boldsymbol{\theta} + \frac{1}{2}\delta\boldsymbol{\theta}^\intercal \mathbf{H}\delta\boldsymbol{\theta} \right]$$

TECHNISCHE UNIVERSITÄT DARMSTADT

Method 3 - Newton's Method

- We want to solve

$$\delta\boldsymbol{\theta} = \arg\min_{\delta\boldsymbol{\theta}} \tilde{J}(\delta\boldsymbol{\theta}) = \arg\min_{\delta\boldsymbol{\theta}} \left[c + \mathbf{g}^{\intercal}\delta\boldsymbol{\theta} + \frac{1}{2}\delta\boldsymbol{\theta}^{\intercal}\mathbf{H}\delta\boldsymbol{\theta} \right]$$

- This leads to computing

$$\begin{aligned} \nabla_{\delta\boldsymbol{\theta}}\tilde{J}(\delta\boldsymbol{\theta}) &= \nabla_{\delta\boldsymbol{\theta}} \left[c + \mathbf{g}^{\intercal}\delta\boldsymbol{\theta} + \frac{1}{2}\delta\boldsymbol{\theta}^{\intercal}\mathbf{H}\delta\boldsymbol{\theta} \right] \\ &= \mathbf{g} + \mathbf{H}\delta\boldsymbol{\theta} = \mathbf{0} \end{aligned}$$

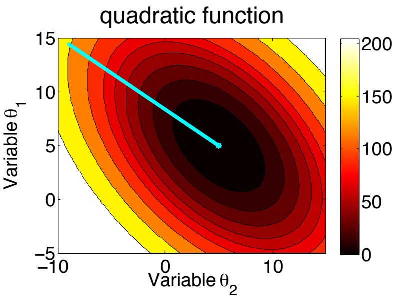
- Which yields the solution

$$\delta\boldsymbol{\theta} = -\mathbf{H}^{-1}\mathbf{g}$$

TECHNISCHE UNIVERSITÄT DARMSTADT

Method 3 - Newton's Method

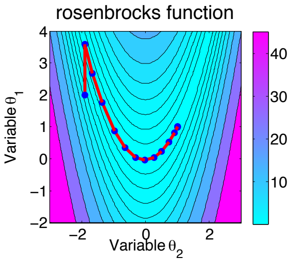
- For quadratic $J(\boldsymbol{\theta})$, the optimal solution is found in one step
- $\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - \mathbf{H}^{-1}(\boldsymbol{\theta}_n)\, g(\boldsymbol{\theta}_n)$ has quadratic convergence
- The solution $\delta\boldsymbol{\theta} = -\mathbf{H}^{-1}\mathbf{g}$ is guaranteed to be downhill if $\mathbf{H}$ is positive definite
- Rather than jumping straight to the predicted solution at $\delta\boldsymbol{\theta} = -\mathbf{H}^{-1}\mathbf{g}$, better do a line search $\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - \alpha\mathbf{H}^{-1}\mathbf{g}$
- For $\mathbf{H} = \mathbf{I}$, this is just the steepest descent



TECHNISCHE UNIVERSITÄT DARMSTADT

Newton's Method on Rosenbrock's Function

- The algorithm converges in only 15 iterations compared to the 101 for conjugate gradients (to come later), and 300 for the regular gradients



- What is the problem with this method? ($\delta\boldsymbol{\theta} = -\mathbf{H}^{-1}\mathbf{g}$)
  - Computing the Hessian matrix at each iteration – this is not always feasible and often too expensive

Quasi-Newton Method: BFGS

- Approximate the Hessian matrix using the following ideas
  - Hessians change slowly
  - Hessians are symmetric
  - Derivatives interpolate
- These lead to the optimization problem

$$\begin{aligned} &\min \|\mathbf{H} - \mathbf{H}_n\| \\ &\text{s.t. } \mathbf{H} = \mathbf{H}^\intercal \\ &\qquad \mathbf{H}(\boldsymbol{\theta}_{n+1} - \boldsymbol{\theta}_n) = \mathbf{g}(\boldsymbol{\theta}_{n+1}) - \mathbf{g}(\boldsymbol{\theta}_n) \end{aligned}$$

TECHNISCHE UNIVERSITÄT DARMSTADT

Quasi-Newton Method: BFGS

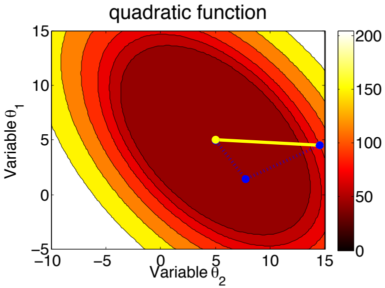
- Thus the Hessian can be computed iteratively

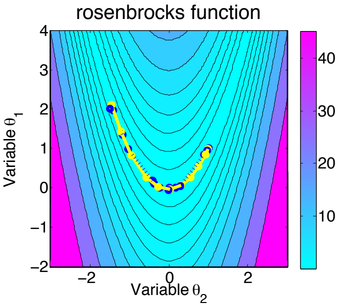
$$\mathbf{H}_{n+1}^{-1} = \left(\mathbf{I} - \frac{\mathbf{s}_k \mathbf{y}_k^{\mathsf{T}}}{\mathbf{s}_k^{\mathsf{T}} \mathbf{y}_k}\right) \mathbf{H}_n^{-1} \left(\mathbf{I} - \frac{\mathbf{s}_k \mathbf{y}_k^{\mathsf{T}}}{\mathbf{s}_k^{\mathsf{T}} \mathbf{y}_k}\right)^{\mathsf{T}} + \frac{\mathbf{s}_k \mathbf{y}_k^{\mathsf{T}}}{\mathbf{s}_k^{\mathsf{T}} \mathbf{y}_k}$$

  - where $\mathbf{y}_n = \mathbf{g}(\boldsymbol{\theta}_{n+1}) - \mathbf{g}(\boldsymbol{\theta}_n)$ and $\mathbf{s}_n = \boldsymbol{\theta}_{n+1} - \boldsymbol{\theta}_n$

Quasi-Newton Method: BFGS

- First step can be fully off due to initialization but slight errors can be helpful all the way
- For reasonable dimensions BFGS is preferred





TECHNISCHE UNIVERSITÄT DARMSTADT

Method 4 - Conjugate Gradients (a sketch)

- The method of conjugate gradients chooses successive descent directions $\delta\boldsymbol{\theta}_n$ such that it is guaranteed to reach the minimum in a finite number of steps
- Each $\delta\boldsymbol{\theta}_n$ is chosen to be conjugate to all previous search directions with respect to the Hessian $\mathbf{H}$
  - $\delta\boldsymbol{\theta}_n^T \mathbf{H} \delta\boldsymbol{\theta}_j = 0$ for $0 \leq j < n$
  - The resulting search directions are mutually linearly independent
- Remarkably, $\delta\boldsymbol{\theta}_n$ can be chosen using only the knowledge of $\delta\boldsymbol{\theta}_{n-1}$, $\nabla J(\boldsymbol{\theta}_n)$ and $\nabla J(\boldsymbol{\theta}_{n-1})$

$$\delta\boldsymbol{\theta}_n = \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_n) + \frac{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_n)^{\mathsf{T}} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_n)}{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_{n-1})^{\mathsf{T}} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_{n-1})} \delta\boldsymbol{\theta}_{n-1}$$

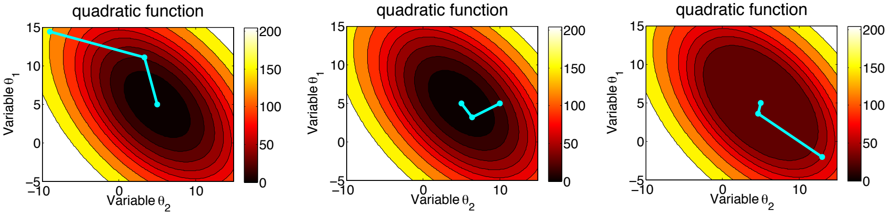
Method 4 - Conjugate Gradients

- It uses first derivatives only, but avoids “undoing” previous work
- An N-dimensional quadratic form can be minimized in at most N conjugate descent steps

TECHNISCHE UNIVERSITÄT DARMSTADT

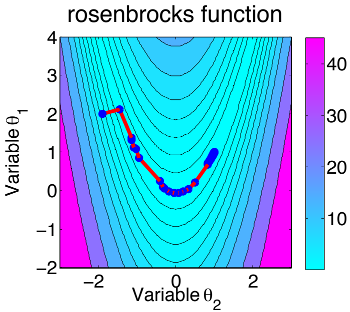
Method 4 - Conjugate Gradients

- 3 different starting points
- The minimum is reached in exactly 2 steps



Conjugate Gradients on Rosenbrock's Function

- The algorithm converges in 101 iterations
- Far superior to steepest descent but slower than Newton's methods
- However, it avoids computing the Hessian which can be more expensive for more dimensions



Conjugate Gradients vs BFGS

- BFGS is more costly than CG per iteration
- BFGS in converges in fewer steps than CG
- BFGS has less of a tendency to get "stuck"
- BFGS requires algorithmic “hacks” to achieve significant descent for each iteration
- Which one is better depends on your problem!

Performance Issues

- Number of iterations required
- Cost per iteration
- Memory footprint
- Region of convergence
- Is the cost function noisy?

Outline

5. Wrap-Up

You know now:

- How machine learning relates to optimization
- What a good cost function looks like
- What convex sets and functions are
- Why convex functions are important in machine learning
- What unconstrained and constrained optimization are
- What the Lagrangian is
- Different numerical optimization methods

Self-Test Questions

- Why is optimization important for machine learning?
- What do well-formulated learning problems look like?
- What is a convex set and what is a convex function?
- How do I find the maximum of a vector-valued function?
- How to deal with constrained optimization problems?
- How to solve such problems numerically?

TECHNISCHE UNIVERSITÄT DARMSTADT

Homework

- Reading Assignment for next lecture
  - Bishop ch 1.5
  - Murphy ch. 5.7