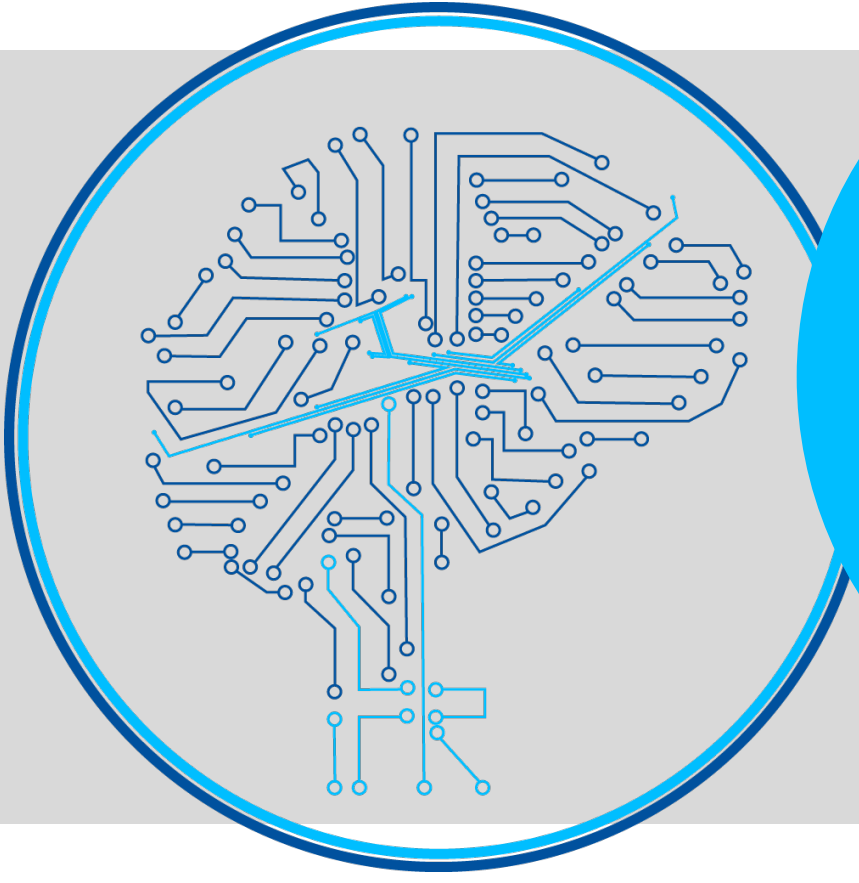
University of Stuttgart
Flight Mechanics and Controls Lab



Gradient based optimization

Fabian Schimpf

Housekeeping

- Tomorrow, Thursday, December 17th 5.30pm:

  Optimierung des Primärenergiebedarfs und der Gesamtkosten (Total Cost of Ownership) von Gebäuden mit Schwarmoptimierung (PSO)

  Ralf Wagner, CTO LTG AG

- Weekly WebEx will from now on cover last weeks lecture and exercise

Recap

Terminology

- Usually defined as the minimization of a given real valued **objective function** f, with respect to a **parameter vector** x under a set of **constraints** g and h.
  - Definition allows for standardized algorithms to solve these problems
  - Maximization can be reframed by multiplying the objective function with -1
  - In ML the parameter vector is often denoted as Θ

$$\begin{aligned} \underset{\vec{x}}{\text{minimize}} \quad & f(\vec{x}) \\ \text{subject to} \quad & g_i(\vec{x}) \leq 0, \text{for } i = 1, ..., n \\ & h_j(\vec{x}) = 0, \text{for } i = 1, ..., m \end{aligned}$$

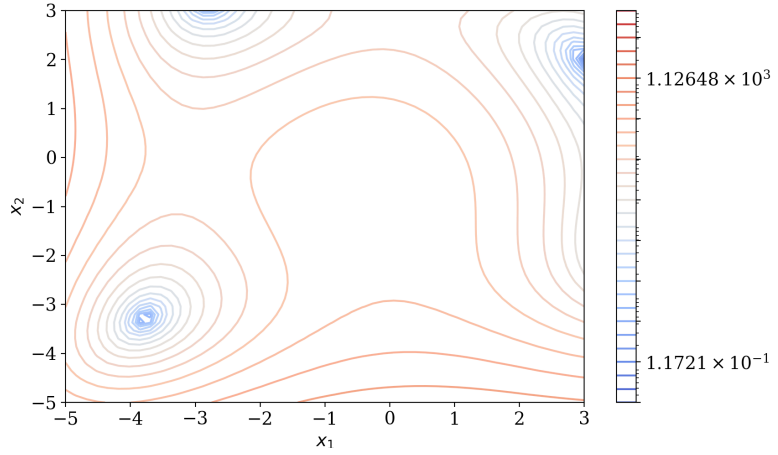
Introduction

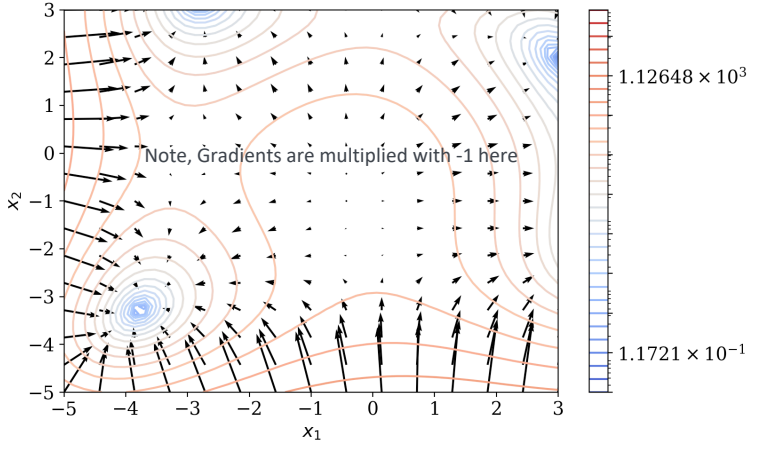
Population based optimization

- Ability to evaluate the loss function at discrete point
- No assumptions / knowledge about the loss function or model needed

Gradient based optimization

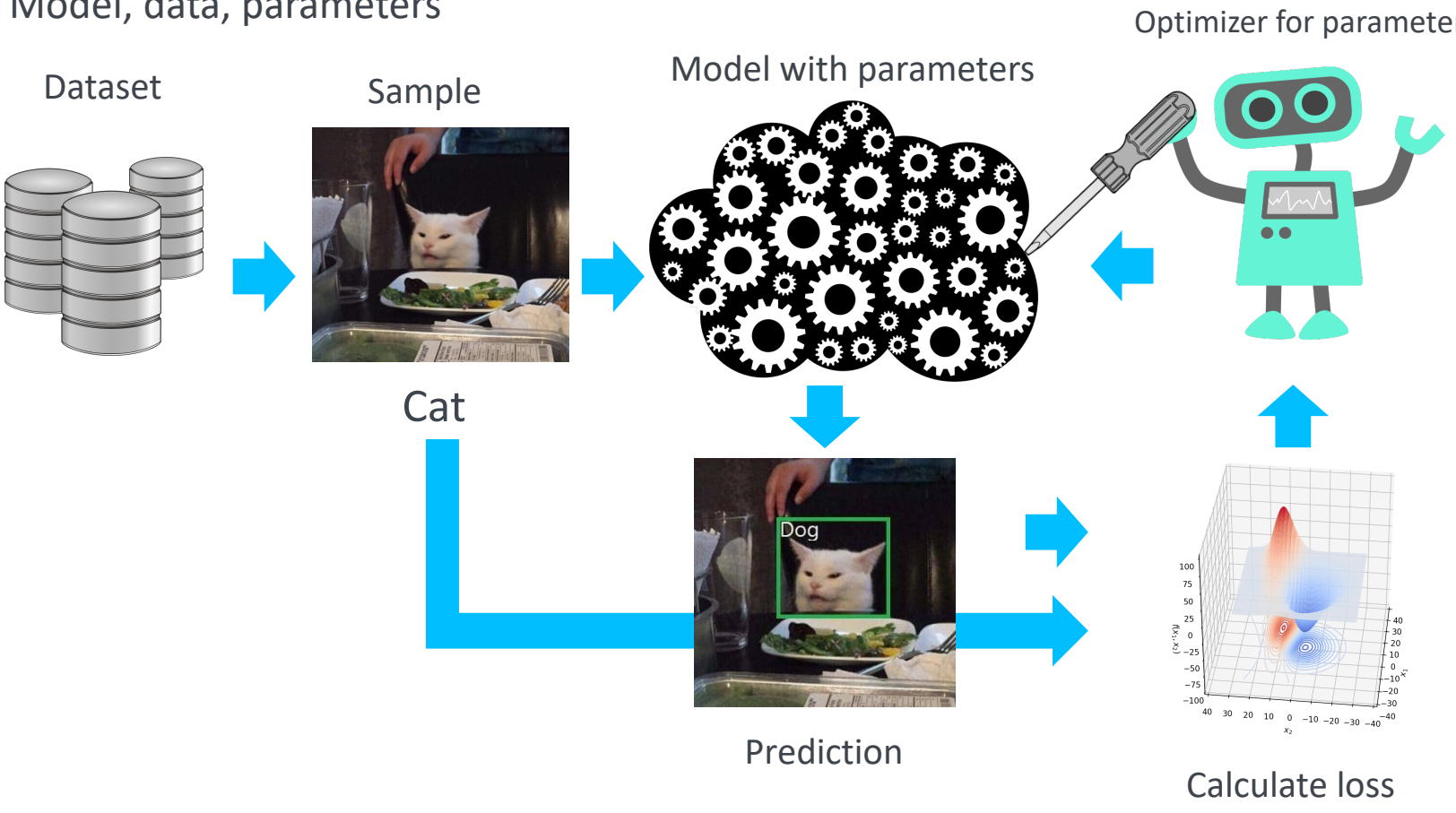
- ”Population” develops from following gradients starting in some initial feasible configuration
- Differentiable loss function needed





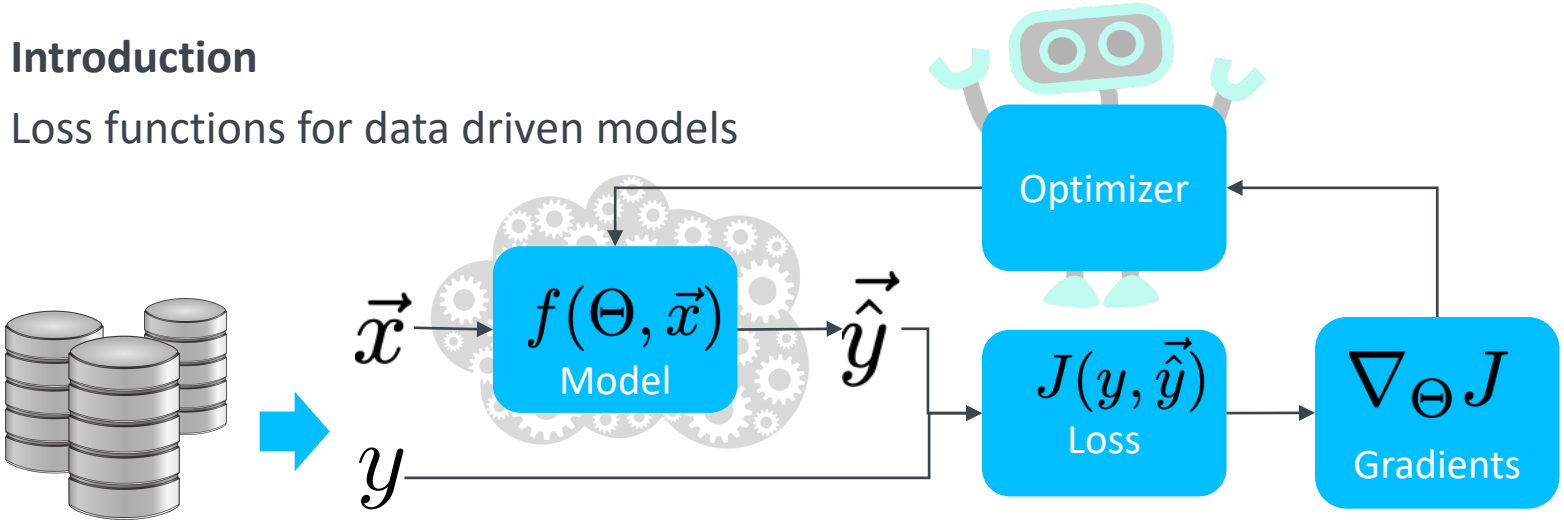
Recap

Model, data, parameters



Introduction

Loss functions for data driven models



Example: Linear Regression but applies analogously to all differentiable models $f(\Theta, \vec{x})$

Model:

$$\hat{y} = \Theta_0 + \Theta_1 \cdot x$$

Loss:

$$J(y, \hat{y}) = (y - \hat{y})^2$$

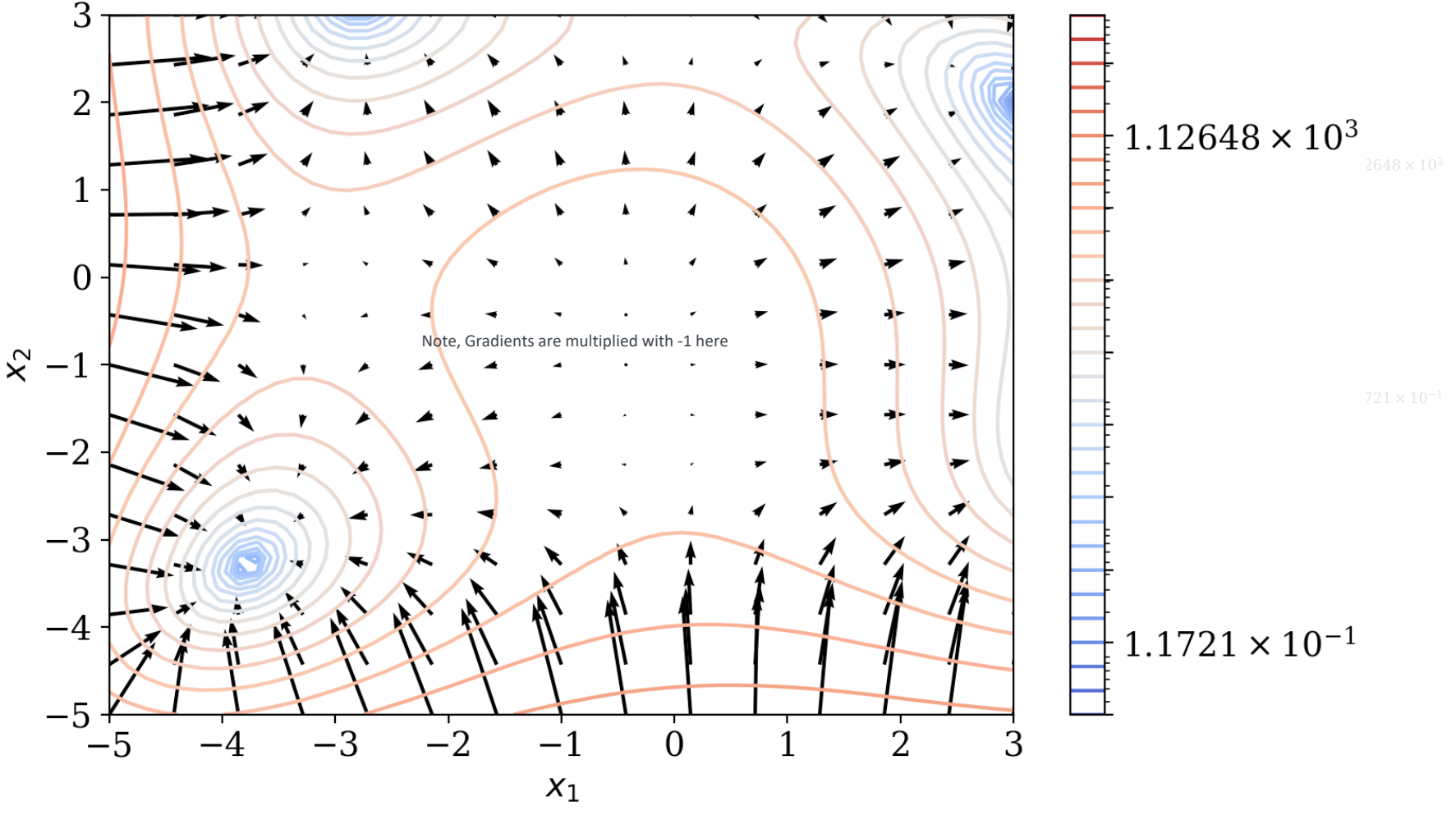
$$J(y, \Theta, x) = (y - (\Theta_0 + \Theta_1 \cdot x))^2$$

Gradient:

$$\nabla_\Theta J = \left[-2 \cdot (y - (\Theta_0 + \Theta_1 \cdot x)), -2x \cdot (y - (\Theta_0 + \Theta_1 \cdot x))\right]$$

$$\nabla_\Theta J = \left[-2 \cdot (y - \hat{y}), -2x \cdot (y - \hat{y})\right]$$

How far should we follow the gradient?


Note, Gradients are multiplied with -1 here
x_1
x_2
1.12648×10^3
1.1721×10^{-1}

Gradient based optimization

Gradient Descent

An overview of gradient descent optimization algorithms – Sebastian Ruder

- Gradient descent

learning rate

$$\theta = \theta - \alpha \cdot \nabla_\theta J(\theta)$$

- Stochastic gradient descent

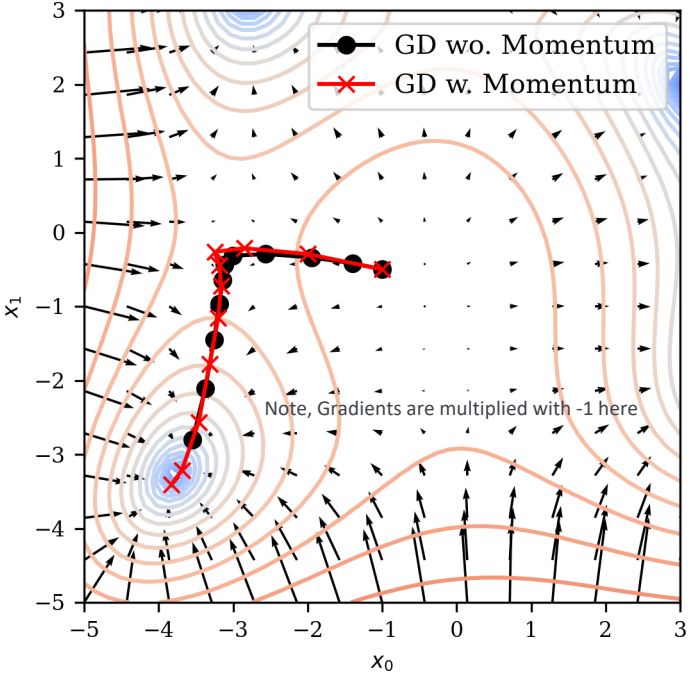
$$\theta = \theta - \alpha \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)})$$

- Mini batch stochastic gradient descent

$$\theta = \theta - \alpha \cdot \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)})$$

- Stochastic gradient descent w. momentum

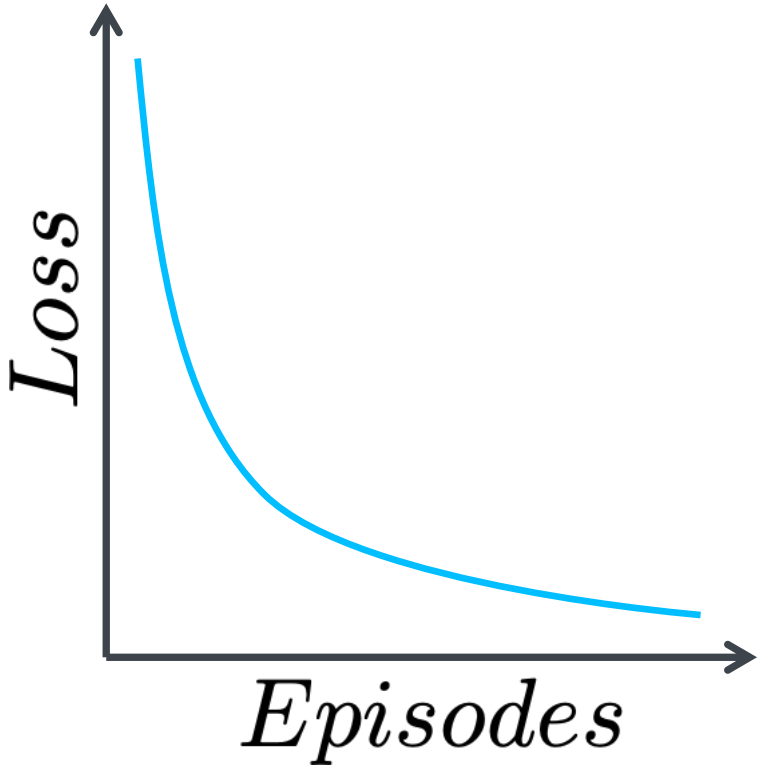
$$v_t = \gamma v_{t-1} + \alpha \nabla_\theta J(\theta)$$
$$\theta = \theta - v_t$$



Gradient Descent

Terminology

- **Episode**
  - Increases with every gradient descent step
    - Note, that the whole dataset has to be taken into account in Mini Batch SGD
  - Equivalent to generations in population based optimization
- **Learning curve**
  - Loss over episodes
  - Decreases over episodes for “well tuned” optimizer
  - Monotonous improvement unlikely for real problems
- **Batch size**
  - Number of samples used for mini-batch SGD



Gradient Descent

Adam - Kingma and Ba, ICLR 2015

Problem:
1st and 2nd order momentum initialized at 0 → almost no improvement at the start

Solution:
Scale up the 1st and 2nd order momentum with a factor decaying to 1 over time

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector

$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)

while θ_t not converged **do**

$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)

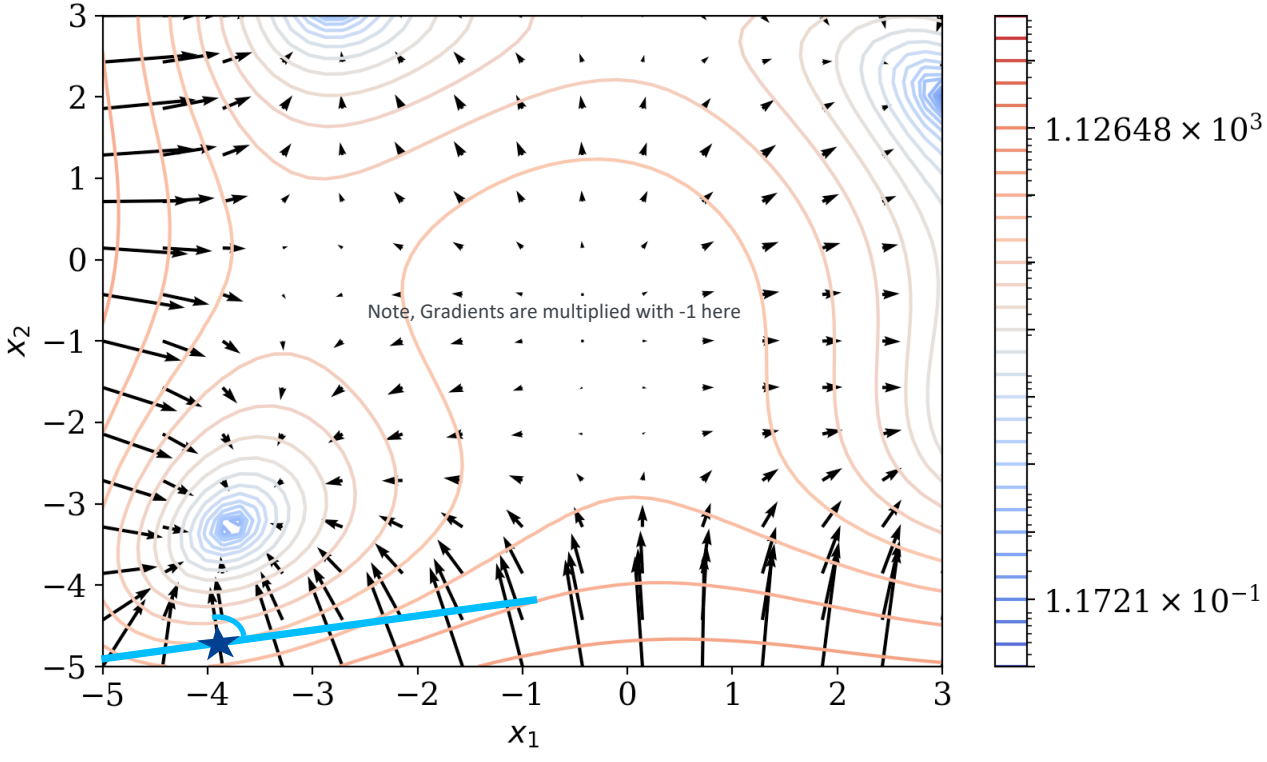
end while
return θ_t (Resulting parameters)

Linear interpolation of old and new 1st and 2nd order momentum

Constrained Optimization

Equality constraints

- Each feasible domain described by $h_i(x) = 0$
- Constraints might contradict themselves but should not!
  - $h_1(x): x = 2$
  - $h_2(x): x = 1$
  - Also applies for inequality constraints
- **Intuition:**
  - Gradient orthogonal to constraint at optimum ★
  - There might be still multiple local optima



Constrained Optimization

Inequality constraints

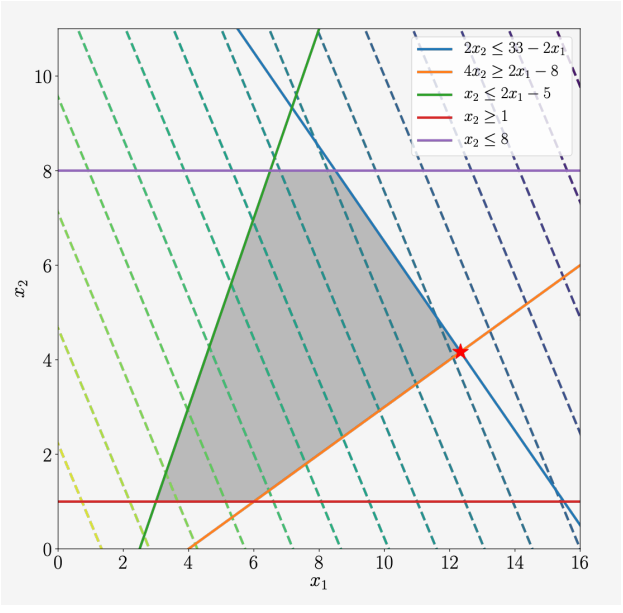
- **Problem Setting:**

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & g_i(\boldsymbol{x}) \leqslant 0 \quad \text{for all} \quad i = 1, \ldots, m\,. \end{aligned}$$

  - Idea: Add cost (barrier) to loss function for unfeasible solutions

$$J(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \mathbf{1}(g_i(\boldsymbol{x})) \qquad \mathbf{1}(z) = \begin{cases} 0 & \text{if } z \leqslant 0 \\ \infty & \text{otherwise} \end{cases}$$

    - Problem: hard to optimize as this loss is discontinuous and non-differentiable



From: Mathematics for Machine Learning

- **Solutions:**
  - Add a linear cost term to enforce constraints → **Lagrange Multipliers** $\boldsymbol{\lambda}$ which have to be equal to or bigger 0
  - Make the barrier function differentiable → **Log barrier**
  - Clip gradients at barrier / move along boundaries

$$\begin{aligned} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) &= f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i g_i(\boldsymbol{x}) \\ &= f(\boldsymbol{x}) + \boldsymbol{\lambda}^\top \boldsymbol{g}(\boldsymbol{x})\,, \end{aligned}$$

Constrained Optimization

Intuitions

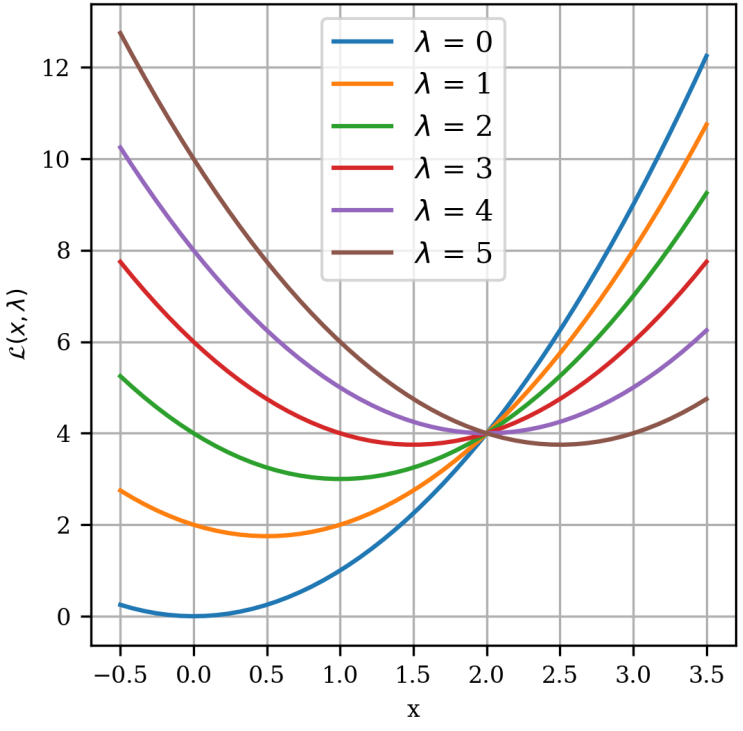
$$\min_x x^2 \mid x \geq 2$$

$$\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i g_i(\boldsymbol{x})$$

$$= f(\boldsymbol{x}) + \boldsymbol{\lambda}^\top \boldsymbol{g}(\boldsymbol{x}),$$

Observations:

- For some lambdas, the function is smaller than the original function we want to minimize → not the right optimum
- For some lambdas the minimum is not in the feasible domain of x
- It seems like there is a lambda for which the minimum of the function is at x=2



Constrained optimization

- **Solution requires optimal Lagrange multipliers as well as optimal parameters**
- **Approach problem directly**
  - Necessary and sufficient conditions for minimum
  - Common solution strategy for equality and inequality constraints: SQP (Sequential Quadratic Programming)
  - More one this in “Nichtlineare Optimierung”
- **Dual Problem → Nested minimization and maximization**

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & g_i(\boldsymbol{x}) \leqslant 0 \quad \text{for all} \quad i = 1, \ldots, m \end{aligned}$$

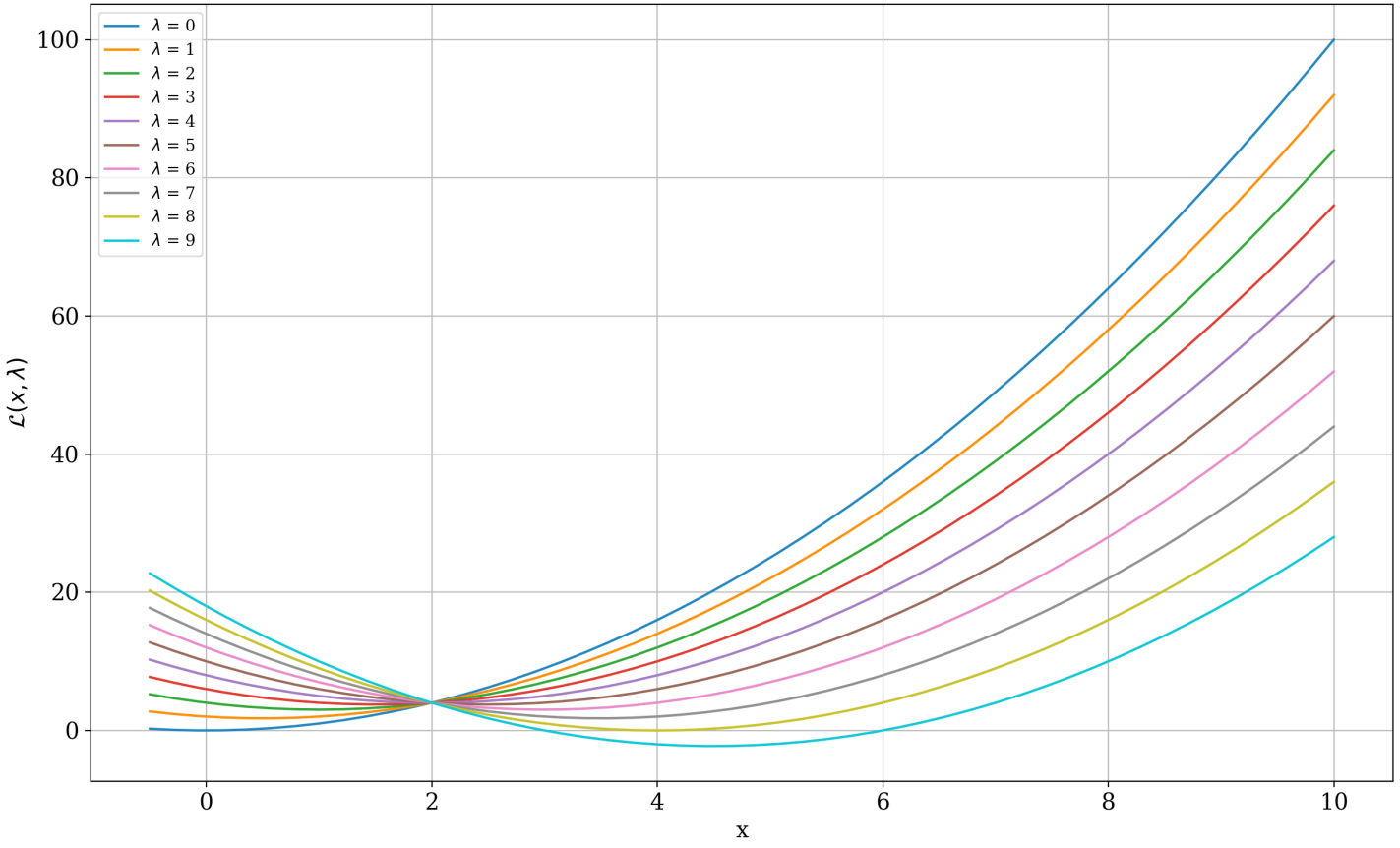
Primal problem

$$\begin{aligned} \max_{\boldsymbol{\lambda} \in \mathbb{R}^m} \quad & \mathfrak{D}(\boldsymbol{\lambda}) = \min_{\boldsymbol{x} \in \mathbb{R}^d} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) \\ \text{subject to} \quad & \boldsymbol{\lambda} \geqslant \mathbf{0} \end{aligned}$$

Lagrangian dual problem

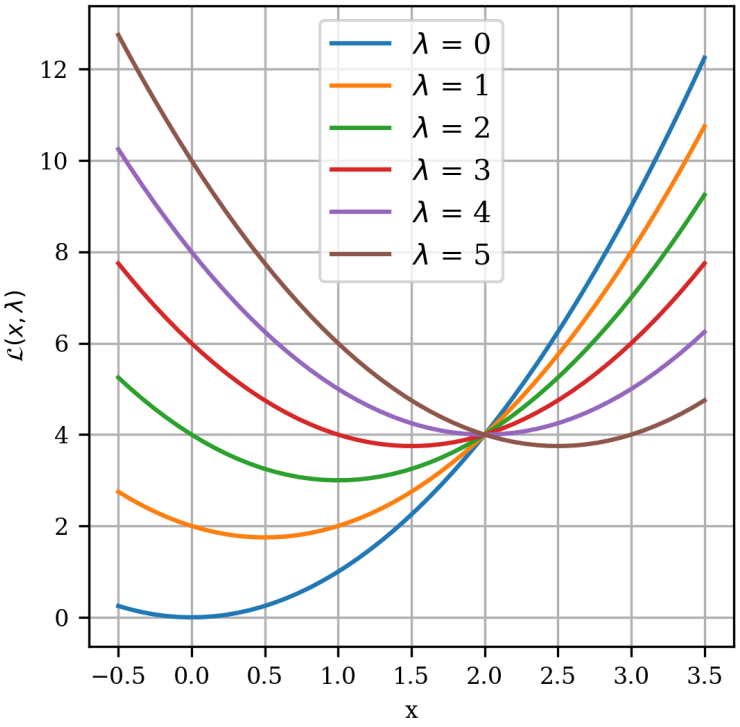
- Strong duality for convex optimization problems: The optimal solution for both descriptions is guaranteed to be the same
- MinMax optimization is at the heart of the GANs and adaptions like GAIL in imitation learning

Constrained optimization

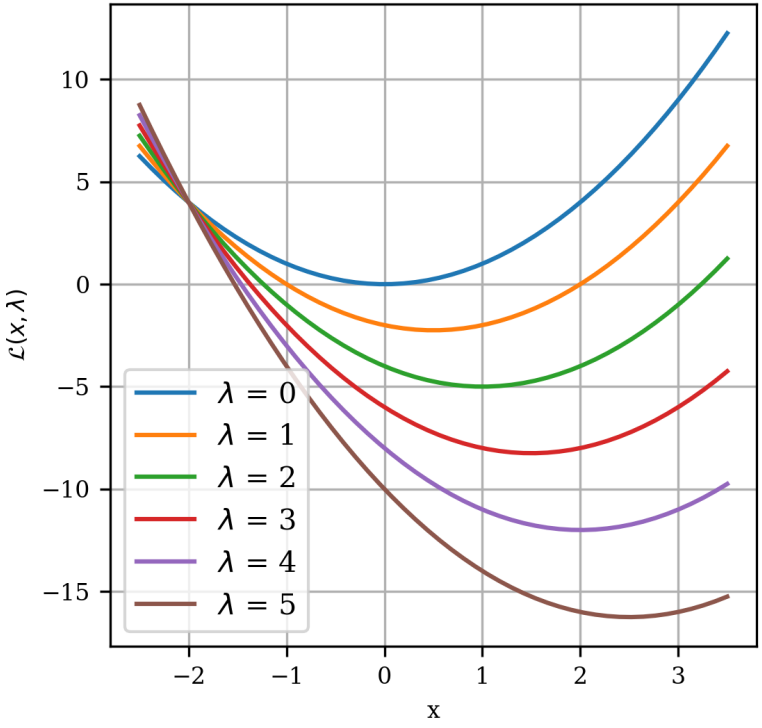


Constrained Optimization

$$\min_x x^2 \mid x \geq 2$$



$$\min_x x^2 \mid x \geq -2$$



Log barrier method

Intuitions

- Receive an unconstrained problem by modifiying the loss function

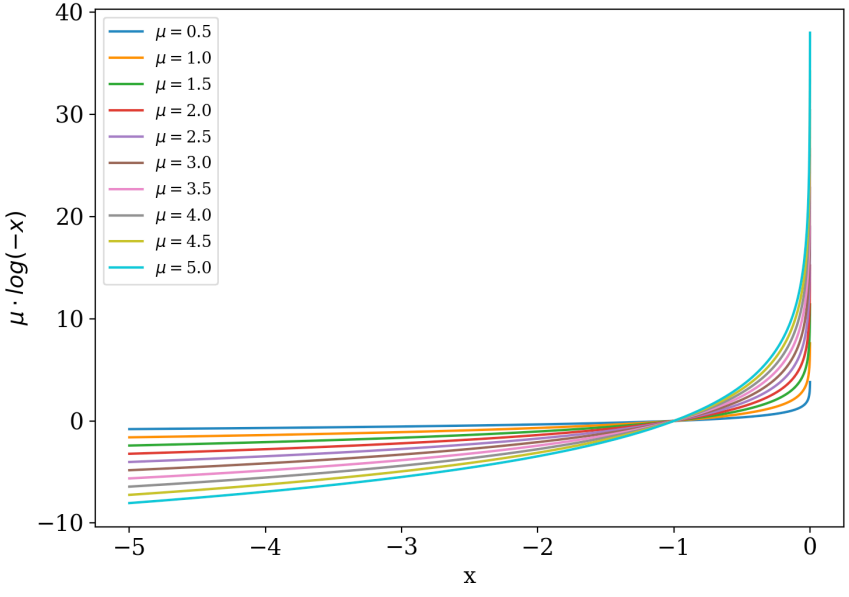
  Instead of

$$\min_x \; f(x) \quad \text{s.t.} \quad g(x) \leq 0$$

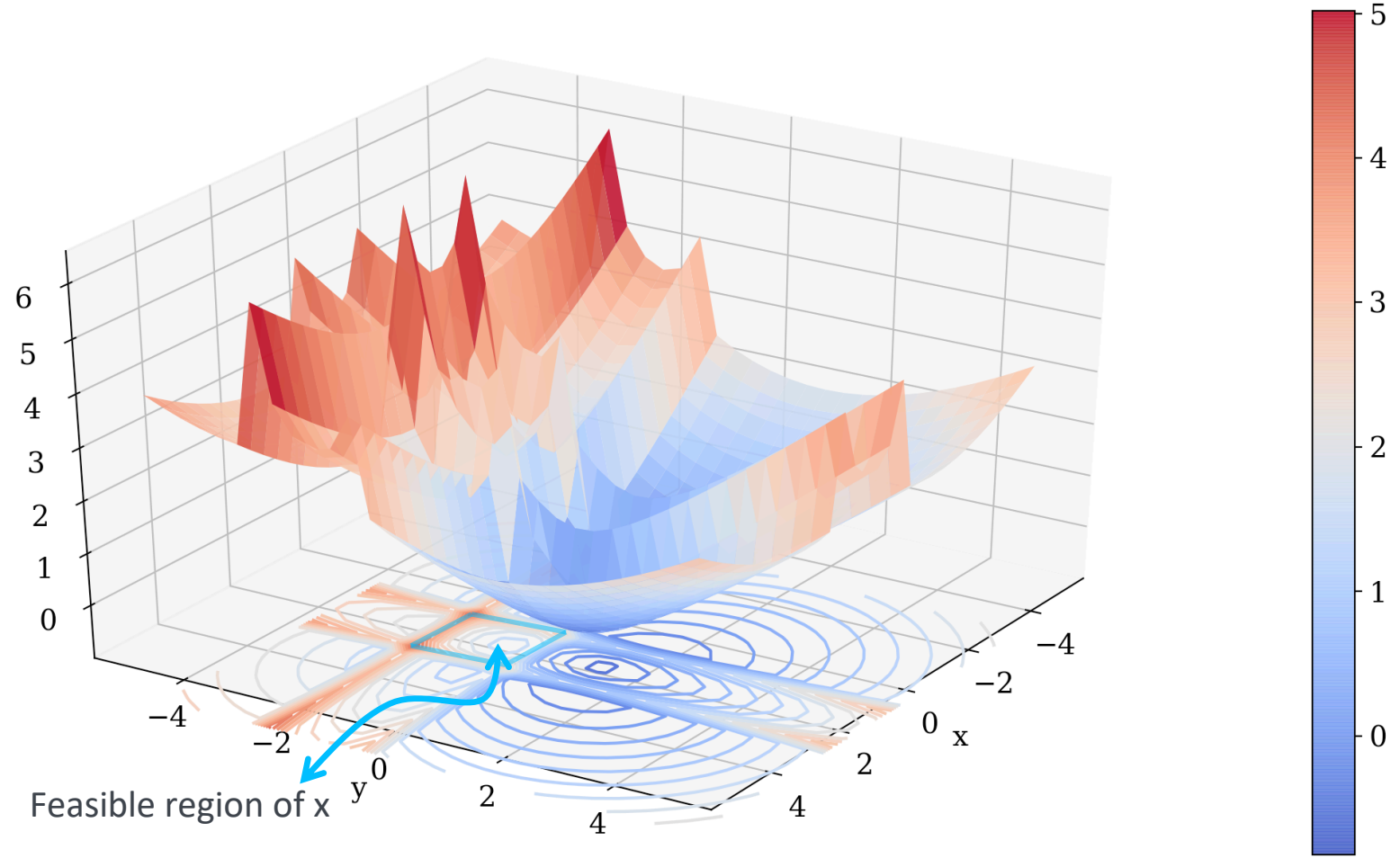
  we solve

$$\min_x \; f(x) - \mu \sum_i \log(-g_i(x))$$

- Problem: How do to choose μ?
- Note, this approach only works for an intial guess of x that is feasible (see next slide for intuition)

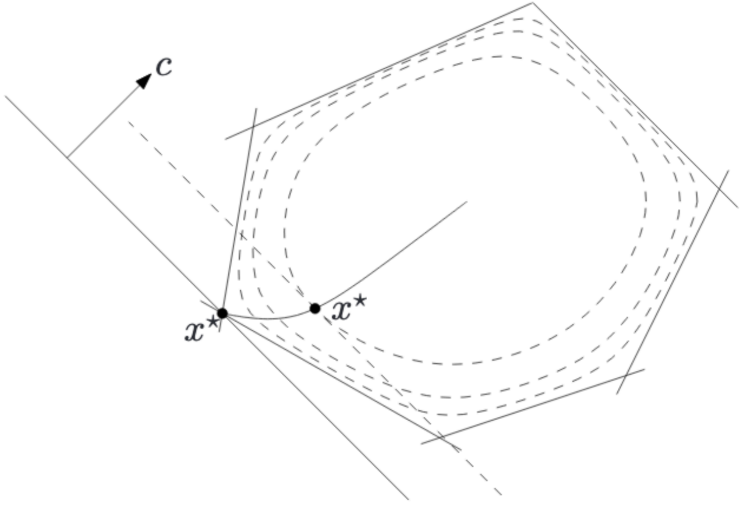


Log barrier method



Log barrier method

Algorithm



From: Convex Optimization, Boyd and Vandenberghe

Algorithm 1: Simplified log barrier method

Result: x

initilize $\mu > 0$;

initilize feasible x;

while *terminal condition not met* **do**

  $x \leftarrow \underset{x}{\text{minimize}}\ f(x) - \mu \sum_i log(-g_i(x))$;

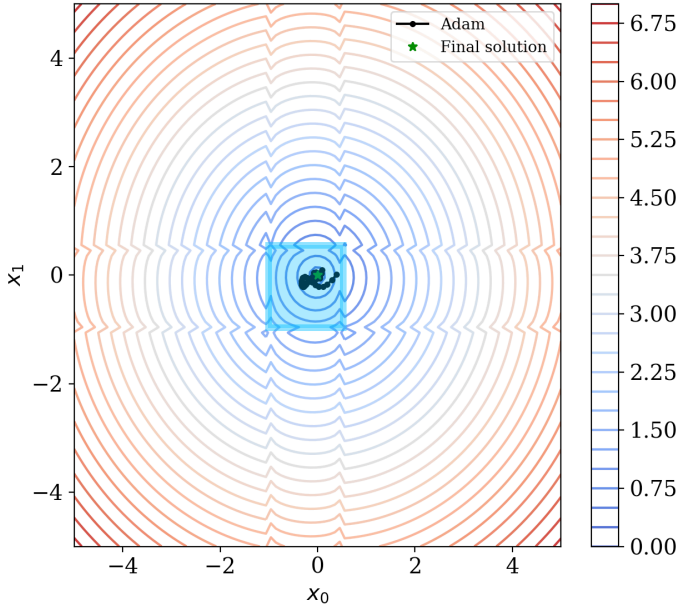
  decrease μ ;

end

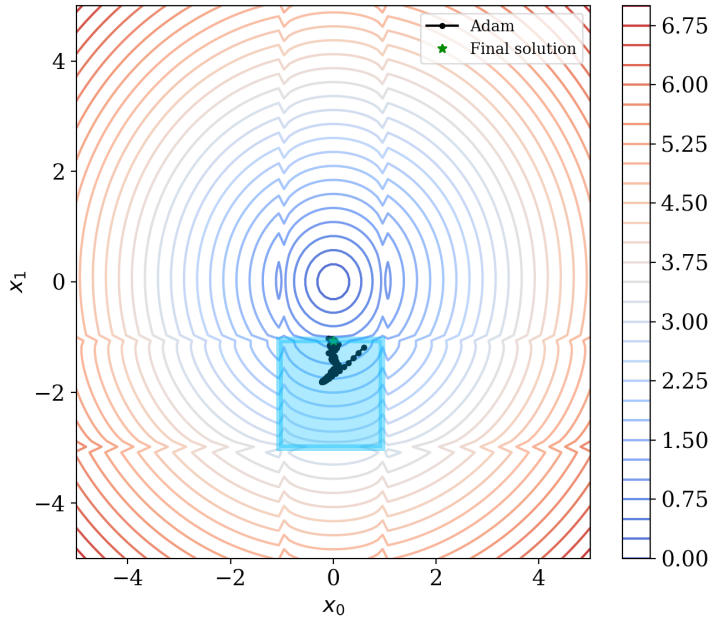
- f(x) and g(x) have to be differentiable to apply gradient descent to the problem
- μ has to stay greater than 0
- For terminal conditions and update schemes for μ see Boyd and Vandenberghe

Log barrier method

MINIMUM IS IN THE INTERIOR



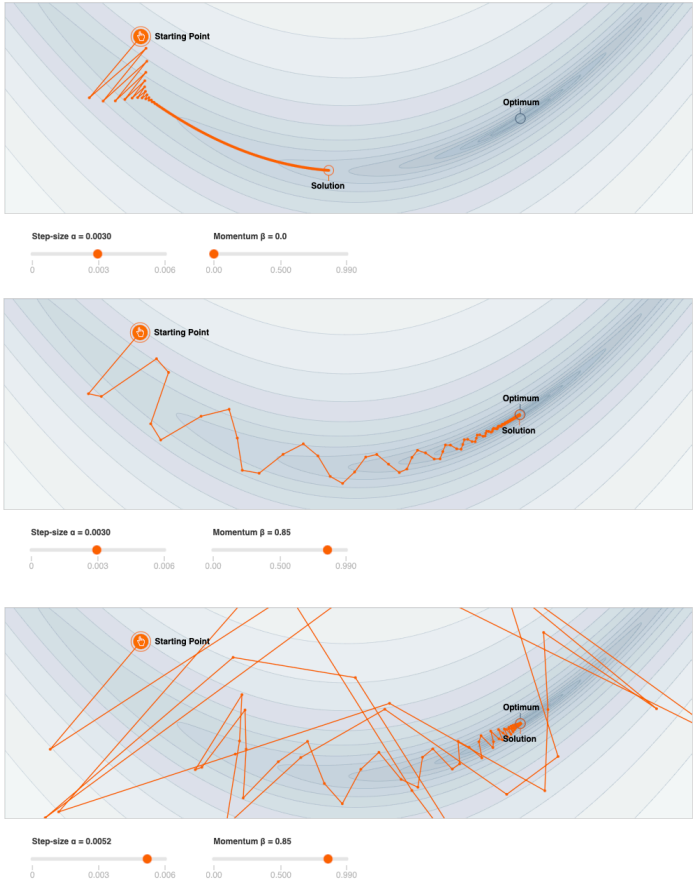
MINIMUM IS AT THE BOUNDARY OF FEASIBLE DOMAIN



Comments on gradient based optimization

Importance of the learning rate

- The learning rate can be adapted over time
  - Schedule
  - Decay
    - Resembles simulated annealing (“Don’t decay the learning rate, increase batch size”, Smith et al.)
  - Heuristic for monotonic improvement (Toussaint)
    - If loss increases after step – step too big
      Undo update and decrease step-size.
    - If loss decreases the step might have been bigger – increase step size
- Good learning rates depend on the selected momentum and other factors → hyperparameter search

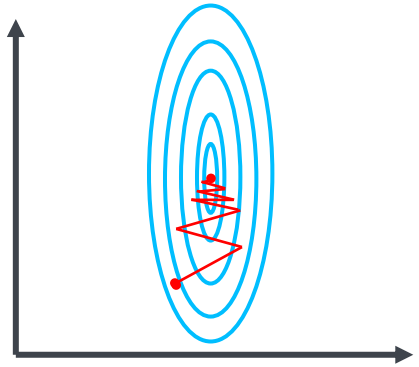


Goh, "Why Momentum Really Works", Distill, 2017. http://doi.org/10.23915/distill.00006

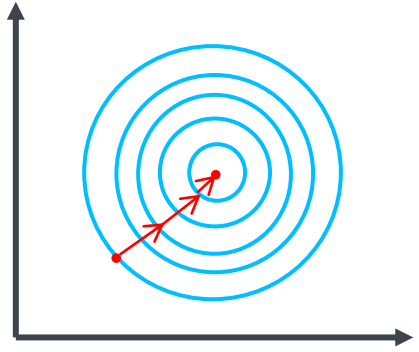
Loss landscape – Parameter Scaling

Scaling is usually applied to the features of the input but can also applied to

- Zero mean – Unity variance
- Min max scaling – Scale all features / parameters to a given range



- Scaling features almost never hurts and is therefore standard procedure
  - Keep track of scaling for later reproducibility



Debugging code including gradient based optimization

Problem	Diagnostics	Workaround
Incorrect gradients	Numeric differentiation	Debug
Slow progress	Slow improvement of loss	Momentum
Instablitity	Loss increases	Decrease learning rate
Oscillations	Loss increases temporarily	Decrease learning rate Momentum Batch size
Local optima	tough	Random initilization

Additional tips and tricks for gradient based optimization

- Standard class of algorithms to train big models and especially neural networks
- Evaluating the samples with the highest loss is helpful to get insights about the model performance
- The learning rate is usually the most important hyper parameter to tune
- Adam is usually works well with little hyperparameter tuning
- Normalizing the loss by the number of samples makes hyperparameter search easier
- Training generic models does not require any constraints on the parameters but high values can be a sign of overfitting – the model learns the data “by heart” – more on this in future lectures
- Try learning a test function first – if this does not work don’t progress further!

More information on the presented topics

Lecture

- Nichtlineare Optimierung – Dr. Grimm
  - Dealing with inequality and equality constraints
  - 2nd order optimization
  - Analytical solutions to constrained optimization problems
- Schätzverfahren

Further reading

- Mathematics of machine learning – Deisenroth et al.
- An overview of gradient descent optimization algorithms – Sebastian Ruder
- Why momentum really works – Gabriel Goh, Distill.pub

Thank you!

Fabian Schimpf

e-mail Fabian.schimpf@ifr.uni-stuttgart.de
phone +49 (0) 711 685-
fax +49 (0) 711 685-

University of Stuttgart
Flight Mechanics and Controls Lab
Pfaffenwaldring 27, 70569 Stuttgart