Contents

List of Symbols

1 Preliminaries 1

1.1 Notation . . . 1

1.2 Definitions . . . 1

2 A Basic Bundle Method 3

2.1 Derivation of the Bundle Method . . . 4

2.1.1 A Stabilized Cutting Plane Method . . . 4

2.1.2 Subproblem Reformulations . . . 6

2.2 The Prox-Operator . . . 7

2.3 ??? . . . 7

3 Variations of the Bundle Method 12

3.1 Convex Bundle Methods with Inexact Information . . . 12

3.1.1 Different Types of Inexactness . . . 13

3.1.2 Noise Attenuation . . . 14

3.1.3 Convergence Results . . . 14

3.2 Nonconvex Bundle Methods with Exact Information . . . 15

3.2.1 Proximity Control . . . 15

3.2.2 Other Concepts . . . 16

4 Proximal Bundle Method for Nonconvex Functions with Inexact Information 17

4.1 Derivation of the Method . . . 17

4.1.1 Inexactness . . . 18

4.1.2 Nonconvexity . . . 19

4.1.3 Aggregate Objects . . . 20

4.2 On Different Convergence Results . . . 22

4.2.1 The Constraint Set . . . 22

4.2.2 Exact Information and Vanishing Errors . . . 23

4.2.3 Convex Objective Functions . . . 23

5 Variable Metric Bundle Method 24

5.1 Main Ingredients to the Method . . . 25

5.1.1 Variable Metric Bundle Methods . . . 25

5.1.2 Noll’s Second Order Model . . . 26

5.1.3 The Descent Measure 27
5.2 The Variable Metric Bundle Algorithm 28
5.3 Convergence Analysis 29
5.4 Updating the Metric 35
5.4.1 Scaling of the Whole Matrix 35
5.4.2 Adaptive Scaling of Single Eigenvalues 36
5.4.3 Other Updating Possibilities 37
5.5 Numerical Testing 38
5.5.1 Academic Test Examples 39

6 Application to Model Selection for Primal SVM 43
6.1 Introduction 43
6.2 Introduction to Support Vector Machines 45
6.2.1 Risk minimization 45
6.2.2 Support Vector machines 46
6.3 Bilevel Approach and Inexact Bundle Method 49
6.3.1 Reformulation as bilevel problem 49
6.3.2 Solution of the Bilevel Program 51
6.3.3 The Algorithm??? 58
6.4 Numerical Experiments 59

References

1 Preliminaries

When it comes to nonsmooth objective functions the derivative based framework of nonlinear optimization methods does not work any more. Meanwhile there exists though a well understood theory of 'subdifferential calculus' that gives similar results in the nondifferentiable case. The most important definitions and results of this theory together with some remarks on notation are stated in this section.

1.1 Notation

Let x denote a column vector. The transpose of x is denoted by $x^\top$. The scalar product is written $\langle \cdot, \cdot \rangle$. 0 denotes the zero vector of appropriate size. $\mathbb{I}$ is the identity matrix of appropriate size. As we work with numerical methods in this thesis occur a lot of sequences of various dimensions. For vectors iteration indices are indicated by a superscript x^k whereas the components are indicated by subscripts $x = (x_1, x_2, ..., x_n)^\top$. Sequences of numbers and matrices a indexed with subscripts. $B_r(x)$ denotes the open ball with radius r around x.

- iteration index as superscript x^k, entry index as subscript x_i
- is the scalar product
- more?

Theoretical Background, nonsmooth Analysis ???

Check if requirements on functions are stated and defined.

1.2 Definitions

Throughout this thesis I consider different optimization problems of the form

$$\min_x f(x), \quad x \in X \subseteq \mathbb{R}^n$$

where f is a possibly nonsmooth function.

Nonsmooth functions have kinks where a unique gradient cannot be defined. It is however possible to define a set of tangents to the graph called subdifferential. The subdifferential was first defined for convex functions.

Definition 1.1 Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function. The *subdifferential* of f at $x \in \mathbb{R}^n$ is the set

$$\partial f(x) := \{g \in \mathbb{R}^n | f(y) - f(x) \geq \langle g, y - x \rangle \quad \forall y \in \mathbb{R}^n\}$$

.

The subdifferential is a set valued mapping. It is convex and closed. If f is differentiable, its subdifferential coincides with its gradient $\partial f(x) = \nabla f(x)$ [43].

It is also possible to define a subdifferential for nonconvex functions. This is the subdifferential we will work with in this thesis most of the time.

Definition 1.2 (c.f. [2]) Let $f : \mathbb{R}^n \to \mathbb{R}$ be locally Lipschitz (and not necessarily convex). The *subdifferential* or *generalized gradient* of f at $x \in \mathbb{R}^n$ is the set

$$\partial f(x) := \{g \in \mathbb{R}^n | \limsup_{y \to x,\ h \searrow 0} \frac{f(y + hv) - f(y)}{h} \quad \forall v \in \mathbb{R}^n\}.$$

All convex functions are locally Lipschitz [14] so the above definition holds also for convex functions. In fact if the function is convex the subdifferential from definition 1.2 is equivalent to definition 1.1 [2]. Due to this equivalence we call elements from both subdifferentials subgradients.

Remark: It is important to observe that subgradient inequality

$$f(y) - f(x) \geq \langle g, y - x \rangle \quad \forall y \in \mathbb{R}^n$$

only holds in the convex case.

Analogous to the $\mathcal{C}^1$-case some first order optimality conditions can be stated. For nondifferentiable functions a *stationary point* x is characterized by

$$0 \in \partial f(x).$$

If the function f is convex, then x is a minimum.

A drawback of the subdifferential is that it does not indicate how near the evaluated point is to a stationary point or minimum of a function. This can only be seen if the evaluated point is already stationary.

This issue is addressed by the ε-*subdifferential.* It gathers all information in small neighborhood of the point x.

For convex functions an ε-*subgradient* of $f(x)$ is defined as a vector $g \in \mathbb{R}^n$ satisfying the inequality

$$f(y) - f(x) \geq \langle g, y - x \rangle - \varepsilon \quad \forall y \in \mathbb{R}^n.$$

The ε-subdifferential is then the set

$$\partial_\varepsilon f(x) := \{g \in \mathbb{R}^n | g \text{ is an } \varepsilon\text{-subgradient of } f(x)\}.$$

For nonconvex functions the subdifferential that is used in this thesis is the *Fréchet* ε-*subdifferential.*

Definition 1.3 (c.f. [15]) The Fréchet ε/subdifferential of $f(x)$ is

$$\partial_{[\varepsilon]} f(x) := \left\{ g \in \mathbb{R}^n | \liminf_{\|h\| \to 0} \frac{f(x+h) - f(x) - \langle g, h \rangle}{\|h\|} \geq -\varepsilon \right\}.$$

For $\varepsilon = 0$ this is called *Fréchet subdifferential.* For convex functions the Fréchet ε-subdifferential and the ε-subdifferential are *not* the same.

See if requirements in definitions and theorems meet what is needed/provided later.

2 A Basic Bundle Method

When bundle methods were first introduced in 1975 by Lemaréchal and Wolfe they were developed to minimize a convex (possibly nonsmooth) function f for which at least one subgradient at any point x can be computed [30]. To provide an easier understanding of the proximal bundle method in [10] and stress the most important ideas of how to deal with nonconvexity and inexactness first a basic bundle method is shown here.

Bundle methods can be interpreted in two different ways: From the dual point of view one tries to approximate the ε-subdifferential to finally ensure first order optimality conditions. The primal point of view interprets the bundle method as a stabilized form of the cutting plane method where the objective function is modeled by tangent hyperplanes

[9]. We focus here on the primal approach.

2.1 Derivation of the Bundle Method

This section gives a short summery of the derivations and results of chapter XV in [13] where a primal bundle method is derived as a stabilized version of the cutting plane method. If not otherwise indicated the results in this section are therefore taken from [13].

The optimization problem considered in this section is

$$\min_x f(x) \quad \text{s.t.} \quad x \in X \tag{2.1}$$

where f is a convex but possibly nondifferentiable function and $X \subseteq \mathbb{R}^n$ is a closed and convex set.

2.1.1 A Stabilized Cutting Plane Method

The geometric idea of the *cutting plane method* is to build a piecewise linear model of the objective function f that can be minimized more easily than the original objective function. This model is built from a *bundle* of information that is gathered in the previous iterations. In the k'th iteration, the bundle consists of the previous iterates x^j, the respective function values $f(x^j)$ and a subgradient at each point $g^j \in \partial f(x^j)$ for all indices j in the index set J_k. From each of these triples, one can construct a linear function

$$l_j(x) = f(x^j) + \left\langle g^j, x - x^j \right\rangle$$

where $f(x^j) = l_j(x^j)$ and due to convexity $f(x) \geq l_j(x),\ x \in X$.
The objective function f can then be approximated by the piecewise linear function

$$m_k(x) = \max_{j \in J_k} l_j(x). \tag{2.2}$$

A new iterate x^{k+1} is found by solving the subproblem

$$\min_x m_k(x) \quad \text{s.t.} \quad x \in X.$$

Picture of function and cutting plane approximation of it

This subproblem should of course be easier to solve than the original task. A question that depends a lot on the structure of X. If $X = \mathbb{R}^n$ or a polyhedron, the problem can be solved easily. Still there are some major drawbacks to the idea. For example if $X = \mathbb{R}^n$ the solution of the subproblem in the first iteration is always $-\infty$. In general we can say that the subproblem does not necessarily have a solution. To tackle this problem a penalty term is introduced to the subproblem. It then reads

$$\min \tilde{m}_k(x) = m_k(x) + \frac{1}{2t_k}\|x - x^k\|^2 \quad \text{s.t.} \quad x \in X,\ t_k > 0. \tag{2.3}$$

This new subproblem is strongly convex and therefore always has a unique solution.

The regularization term can be motivated and interpreted in many different ways, c.f. [13]. From different possible regularization terms the most popular in bundle methods is the penalty-like regularization used here.

The second major step towards the bundle algorithm is the introduction of a so called *stability center* or *serious point* $\hat{x}^k$. It is the iterate that yields the "best" approximation of the optimal point up to the k'th iteration (not necessarily the lowest function value though). The updating technique for $\hat{x}^k$ is crucial for the convergence of the method: If the next iterate yields a decrease of f that is "large enough", namely larger than a fraction of the decrease suggested by the model function for this iterate, the stability center is moved to that iterate. If this is not the case, the stability center remains unchanged.

In practice this is implemented as follows: First define the *model decrease* δ_k^M which is the decrease of the model for the new iterate x^{k+1} compared to the function value at the current stability center $\hat{x}^k$

$$\delta_k^M := f(\hat{x}^k) - m_k(x^{k+1}) \geq 0. \tag{2.4}$$

If the actual decrease of the objective function is larger than a fraction of the model decrease

$$f(\hat{x}^k) - f(x^{k+1}) \geq m\delta_k^M, \quad m \in (0,1)$$

set the stability center to $\hat{x}^{k+1} = x^{k+1}$. This is called a *serious* or *descent step*. If this is not the case a *null step* is executed and the serious iterate $\hat{x}^{k+1} = \hat{x}^k$ remains the same .

Besides the model decrease other forms of decrease measures and variations of these are possible. Some are presented in [13] and [56].

2.1.2 Subproblem Reformulations

The subproblem to be solved to find the next iterate can be rewritten as a smooth optimization problem. For convenience we first rewrite the affine functions l_j with respect to the stability center $\hat{x}^k$.

$$\begin{aligned} l_j(x) &= f(x^j) + \left\langle g^j, x - x^j \right\rangle \\ &= f(\hat{x}^k) + \left\langle g^j, x - \hat{x}^k \right\rangle - \left(f(\hat{x}^k) - f(x^j) + \left\langle g^j, x^j - \hat{x}^k \right\rangle \right) \\ &= f(\hat{x}^k) + \left\langle g^j, x - \hat{x}^k \right\rangle - e_j^k \end{aligned}$$

where

$$e_j^k := f(\hat{x}^k) - f(x^j) + \left\langle g^j, x^j - \hat{x}^k \right\rangle \geq 0 \quad \forall j \in J_k$$

is the *linearization error*. Due to convexity of f it is nonnegative. This property is essential for the convergence theory and will also be of interest when moving on to the case of nonconvex and inexact objective functions.

Subproblem (2.3) can now be written as

$$\min_{\hat{x}^k + d \in X} \tilde{m}_k(\hat{x}^k + d) = f(\hat{x}^k) + \max_{j \in J_k} \left\{ \left\langle g^j, d \right\rangle - e_j^k \right\} + \frac{1}{2t_k}\|d\|^2 \tag{2.5}$$

$$\Leftrightarrow \quad \min_{\substack{\hat{x}^k + d \in X, \\ \xi \in \mathbb{R}}} \xi + \frac{1}{2t_k}\|d\|^2 \quad \text{s.t.} \quad \left\langle g^j, d \right\rangle - e_j^k - \xi \leq 0, \quad j \in J_k \tag{2.6}$$

where $d := x - \hat{x}^k$ and the constant term $f(\hat{x}^k)$ was discarded for the sake of simplicity. If X is a polyhedron this is a convex quadratic optimization problem that can be solved using standard methods of nonlinear optimization. It should however be observed that the matrix of the quadratic part is only positive semidefinite because it does not have full rank.

The pair (ξ_k, d^k) solves (2.6) if and only if

$$d^k \text{ solves the original subproblem (2.5) and}$$
$$\xi_k = \max_{j \in J_k} {g^j}^\top d^k - e_j^k = m_k(\hat{x}^k + d^k) - f(\hat{x}^k). \tag{2.7}$$

The new iterate is given by $x^{k+1} = \hat{x}^k + d^k$.

2.2 The Prox-Operator

The constraint $\hat{x}^k + d \in X$ can also be incorporated directly in the objective function by using the indicator function

$$\mathtt{i}_X(x) = \begin{cases} 0, & \text{if } x \in X \\ +\infty, & \text{if } x \notin X \end{cases}.$$

This function is convex if and only if the set X is convex [46].

Subproblem (2.3) then reads with respect to the serious point $\hat{x}^k$

$$\min_{x \in \mathbb{R}^n} m_k(x) + \mathtt{i}_X(x) + \frac{1}{2t_k}\|x - \hat{x}^k\|^2. \tag{2.8}$$

The subproblem is now written as the *Moreau-Yosida regularization* of $\check{f}(x) := m_k(x) + \mathtt{i}_X(x)$. The emerging mapping is also known as *proximal point mapping* [9] or *prox-operator*

$$prox_{t,\check{f}}(x) = \arg\min_{y \in \mathbb{R}^n} \left\{ \check{f}(y) + \frac{1}{2t}\|x - y\|^2 \right\}, \quad t > 0. \tag{2.9}$$

This special form of the subproblems gives the primal bundle method its name, *proximal bundle method*. The above mapping also plays a key role when the method is generalized to nonconvex objective functions and inexact information.

2.3 ???

not aggregate Objects

We look again at a slightly different formulation of the bundle subproblem

$$\begin{aligned} \min_{\substack{d\in\mathbb{R}^n,\\ \xi\in\mathbb{R}}} \quad & \xi + \mathtt{i}_X(\hat{x}^k + d) + \frac{1}{2t_k}\|d\|^2 \\ \text{s.t.} \quad & \left\langle g^j, d \right\rangle - e_j^k - \xi \leq 0, \quad j \in J_k. \end{aligned}$$

As the objective function is still convex (X is a convex set) the following Karush-Kuhn-Tucker (KKT) conditions have to be valid for the minimizer $\left(\xi_k, d^k\right)$ of the above subproblem [14] assuming a constraint qualification holds if the constraint set X makes it necessary [50].

There exist a subgradient $\nu^k \in \partial \mathtt{i}_X(\hat{x}^k + d^k)$ and Lagrangian multipliers $\alpha_j,\ j \in J^k$ such that

$$0 = \nu^k + \frac{1}{t_k} d^k + \sum_{j\in J^k} \alpha_j g^j \tag{2.10}$$

$$\sum_{j\in J_k} \alpha_j = 1, \tag{2.11}$$

$$\alpha_j \geq 0,\ j \in J^k, \tag{2.12}$$

$$\left\langle g^j, d^k \right\rangle - e_j^k - \xi_k \leq 0, \tag{2.13}$$

$$\sum_{j\in J^k} \alpha_j \left(\left\langle g^j, d^k \right\rangle - e_j^k - \xi_k \right) = 0. \tag{2.14}$$

From condition (2.10) follows that

$$d^k = -t_k \left(G^k + \nu^k \right) \tag{2.15}$$

with the *aggregate subgradient*

$$G^k := \sum_{j\in J^k} \alpha_j g^j \quad \in \partial m_k(x^{k+1}). \tag{2.16}$$

The fact that G^k belongs to the subdifferential of the k'th model m_k at the point $\hat{x}^k + d^k$ follows from noting that

$$0 \in \partial m_k(\hat{x}^k + d^k) + \partial \mathtt{i}_X(\hat{x}^k + d^k) + \frac{1}{2t_k} d^k$$

is the optimality condition derived from formulation (2.8) by the sum rule for subdifferentials and comparing the different components with the ones derived in (2.10).

Rewriting condition (2.14) yields the *aggregate error*

$$E_k := \sum_{j \in J^k} \alpha_j e_j^k = \left\langle G^k, d^k \right\rangle + f(\hat{x}^k) - m_k(x^{k+1}). \tag{2.17}$$

Here relation (2.7) was used to replace ξ_k.

The aggregate subgradient and error are used to formulate an implementable stopping condition for the bundle algorithm. The motivation behind that becomes clear with the following lemma.

Lemma 2.1 *[7, Theorem 6.68] Let $X = \mathbb{R}^n$. Let $\varepsilon > 0$, $\hat{x}^k \in \mathbb{R}^n$ and $g^j \in \partial f(x^j)$ for $j \in J^k$. Then the set*

$$\mathcal{G}_\varepsilon^k := \left\{ \sum_{j \in J^k} \alpha_j g^j \mid \sum_{j \in J^k} \alpha_j e_j \leq \varepsilon, \sum_{j \in J^k} \alpha_j = 1, \alpha_j \geq 0, j \in J^k \right\}$$

is a subset of the ε-subdifferential of $f(\hat{x}^k)$

$$\mathcal{G}_\varepsilon^k \subseteq \partial_\varepsilon f(\hat{x}^k).$$

This means that in the unconstrained case $G^k \in \partial_{E_k} f(\hat{x}^k)$. So driving $\|G^k\|$ and E_k to zero results in some approximate ε-optimality of the objective function. In the constrained case the stopping condition is written as

$$\delta_k = E^k + t_k \|G^k + \nu^k\|^2 \leq \texttt{tol}$$

for a fixed tolerance $\texttt{tol} > 0$.

The decrease measure δ_k is also taken for the decrease test. The relation

$$\begin{aligned} \delta_k &= E^k + t_k \|G^k + \nu^k\|^2 \\ &= E^k - \left\langle G^k, d^k \right\rangle - \left\langle \nu^k, d^k \right\rangle \\ &= f(\hat{x}^k) - m_k(x^{k+1}) - \left\langle \nu^k, d^k \right\rangle \end{aligned}$$

where (2.16) and (2.17) were used, shows that the new δ_k is only a small variation of the model decrease δ_k^M. If the iterate x^{k+1} does not lie on the boundary of the constraint set X, the vector ν^k is equal to zero and the expression simplifies to the one stated in (2.4).

For the model update the following two conditions are assumed to be fulfilled in consecutive null steps:

$$m_{k+1}(\hat{x}^k + d) \geq f(\hat{x}^{k+1}) - e_{k+1}^{k+1} + \left\langle g^{k+1}, d \right\rangle \quad \forall d \in \mathbb{R}^n \tag{2.18}$$

$$m_{k+1}(\hat{x}^k + d) \geq a_k(\hat{x}^k + d) \quad \forall d \in \mathbb{R}^n \tag{2.19}$$

The first condition means that the newly computed information is always put into the bundle. The second one is important when updating the bundle index set J^k. It holds trivially if no or only inactive information j with $\alpha_j = 0$ is removed [10]. It is also always satisfied if the aggregate linearization a_k itself is added to the bundle. In this case active information can be removed without violating the condition. This is the key idea of Kiwiel's aggregation technique and ensures that the set $\{j \in J^k | \alpha_j > 0\}$ can be bounded.

An issue of bundle methods is that in spite of the possibility to delete inactive information the bundle can still become very large. Kiwiel therefore proposed a totally different use of the aggregate objects in [19]. The aggregate subgradient can be used to build the *aggregate linearization*

$$a_k(\hat{x}^k + d) := m_k(x^{k+1}) + \langle G^k, d - d^k \rangle.$$

This function can be used to avoid memory overflow as it compresses the information of all bundle elements into one affine plane. Adding the function a_k to the cutting plane model preserves the assumptions (2.18) and (2.19) put on the model and can therefore be used instead of or in combination with the usual cutting planes.

This can however impair the speed of convergence if the bundle is kept too small and provides hence less information about the objective function [4].

We have now all the ingredients so that the following basic bundle algorithm can be stated:

Algorithm 2.1: Basic Bundle Method

Select a descent parameter $m \in (0, 1)$ and a stopping tolerance `tol` ≥ 0. Choose a starting point $x^1 \in \mathbb{R}^n$ and compute $f(x^1)$ and g^1. Set the initial index set $J_1 := \{1\}$ and the initial stability center to $\hat{x}^1 := x^1$, $f(\hat{x}^1) = f(x^1)$ and select $t_1 > 0$.

For $k = 1, 2, 3 \ldots$

1. Calculate
$$d^k = \underset{d \in \mathbb{R}^n}{\arg\min}\, m_k(\hat{x}^k + d) + \mathtt{i}_X(\hat{x}^k + d) + \frac{1}{2t_k}\|d\|^2$$
and the corresponding Lagrange multiplier $\alpha_j^k,\ j \in J_k$.
2. Set
$$G^k = \sum_{j \in J_k} \alpha_j^k g_j^k, \quad E_k = \sum_{j \in J_k} \alpha_j^k e_j^k, \quad \text{and} \quad \delta_k = E_k + \frac{1}{t_k} d_k^2$$
If $\delta_k \leq \mathtt{tol} \rightarrow$ STOP.
3. Set $x^{k+1} = \hat{x}^k + d^k$.
4. Compute $f(x^{k+1}),\ g^{k+1}$.
If
$$f(x^{k+1}) \leq f(\hat{x}^k) - m\delta_k \quad \rightarrow \text{serious step.}$$
Set $\hat{x}^{k+1} = x^{k+1},\ f(\hat{x}^{k+1}) = f(x^{k+1})$ and select a suitable $t_{k+1} > 0$.
Otherwise $\rightarrow$ nullstep.
Set $\hat{x}^{k+1} = \hat{x}^k,\ f(\hat{x}^{k+1}) = f(x^{k+1})$ and choose $t_{k+1} > 0$ in a suitable way.
5. Select the new bundle index set J_{k+1}, calculate e_j^{k+1} for $j \in J_{k+1}$ and update the model m_k.

In steps 4 and 5 of the algorithm it is not specified how to update the parameter t_k, the index set J^k and the model m_k. For the convergence proof it is only necessary that $\liminf_{k \to \infty} t_k > 0$ and that conditions (2.18) and (2.19) are fulfilled.

In practice the choice of t_k can be realized by taking

$$t_{k+1} = \kappa_+ t_k, \quad \kappa_+ > 1 \tag{2.20}$$

at every serious step and

$$t_{k+1} = \max\{\kappa_- t_k, t_{min}\}, \quad \kappa_- < 1 \text{ and } t_{min} > 0 \tag{2.21}$$

at every null step. The idea behind this management of t_k is taken from the trust region method: If the computed iterate was good, the model is assumed to be reliable in a larger area around this serious iterate so bigger step sizes are allowed. If a null step was taken, the model seems to be too inaccurate far from the current serious point. Then smaller step sizes are used. A more sophisticated version of this kind of step size management is also used by Noll et al. in [39] and [37]. The trust region idea was very much exploited by Schramm and Zowe in [47]. In the case $X = R^n$ the sequence $\{\hat{x}^k\}$ can be unbounded. In this case bounding $t_k \leq t_{max} < \infty$ for all k preserves the convergence proof [13, Theorem 3.2.2].

In general it can be shown that if f posses global minima and the basic bundle algorithm generates the sequence $\{\hat{x}^k\}$ this sequence converges to a minimizer of problem (2.1) (c.f [13]).

3 Variations of the Bundle Method

After their discovery in 1975 bundle methods soon became very successful. Only a few years later they were generalized to be used also with nonconvex objective functions. Early works, that contain fundamental ideas still used for these algorithms are [29] and [18]. It then took over 25 years that bundle methods were again generalized to the use of inexact information, first works on this subject being [12, 20] and [49].

This section of the thesis shortly presents the key ideas of those two kinds of generalizations and different types of bundle methods that realize them. This is first done for the case of convex objective functions with inexact function value and/or subgradient information and then for nonconvex objective functions.

3.1 Convex Bundle Methods with Inexact Information

We focus here on *convex* bundle methods with inexact information. The reason for this is that there is a fundamental difference in treating inexactness between methods that assume convex and those that assume nonconvex objective functions. When dealing with nonconvex objective functions inexactness is treated as some additional nonconvexity therefore no additional strategies are used to cope with the noise. This is not possible if the convexity property is to be exploited for better convergence results. A throughout study on this subject including a synthetic convergence theory is done in [56]. Here the most important aspects of that paper are reviewed.

3.1.1 Different Types of Inexactness

Throughout this section we consider the optimization problem

$$\min_{x \in \mathbb{R}^n} f(x) \tag{3.1}$$

where the function $f : \mathbb{R}^n \to \mathbb{R}$ is a finite convex function. The function values and one subgradient at each point x are given by an inexact oracle. It is reasonable to define very different kinds of inexactness and further assumptions can be put on the noise to reach stronger convergence results. However, generally inexact information for convex objective functions is defined in the following way:

$$f_x = f(x) - \sigma_x, \quad \sigma_x \leq \bar{\sigma} \tag{3.2}$$

$$g_x \in \mathbb{R}^n \text{ such that } f(\cdot) \geq f_x + \langle g_x, \cdot - x \rangle - \theta_x, \quad \theta_x \leq \bar{\theta}. \tag{3.3}$$

From this follows because of

$$f(\cdot) \geq f(x) + \langle g_x, \cdot - x \rangle - (\sigma_x + \theta_x) \tag{3.4}$$

that g_x is an ε-subgradient of $f(x)$ with $\varepsilon = \sigma_x + \theta_x \geq 0$ independently of the signs of the errors.

Different convergence results for the applied bundle methods are possible depending on if the bounds $\bar{\sigma}$ and $\bar{\theta}$ are unknown, known or even controllable.

In case of controllability of $\bar{\sigma}$ and $\bar{\theta}$ it may be possible to drive them to zero as the iterations increase $\lim_{k\to\infty} \sigma_k = 0$ and $\lim_{k\to\infty} \theta_k = 0$. We talk then of *asymptotically vanishing errors*. This case is important because it allows convergence to the exact minimum of the problem even if function values and subgradients are erroneous. In the case of $\bar{\theta} = 0$ it even suffices to show that the errors are only asymptotically exact for descent steps [17]. This observation was the motivation for the partly inexact bundle methods presented in [17] and [56]. The idea is to calculate a value of the objective function with a demanded accuracy (which is finally going to be exact) only if a certain target descent γ_x is reached. This approach can save a lot of (unnecessary) computational effort while still enabling convergence to the exact minimum c.f. [56].

In view of good convergence properties oracle that only underestimate the true function, so called *lower oracles*, are also very interesting. Lower oracles provide f_x and g_x such that $f_x \leq f(x)$ and $f(\cdot) \geq f_x + \langle g_x, \cdot - x \rangle$. That means the cutting plane model is always minorizing the true function as it is the case in for exact information. In this case if the value to approximate the optimal function value is chosen properly, it is not necessary to include any new steps into the method to cope with the inexactness, such as noise attenuation [56, Corollary 5.2].

3.1.2 Noise Attenuation

In the case of inexact information, especially if the inexact function value can overestimate the real one, it is possible that the aggregate linearization error E_k becomes very small (or even negative) even though the current iterate is far from the minimum of the objective function. To tackle this problem the authors propose a procedure called *noise attenuation* that was developed in [12] and [20]. The basic idea is to allow bigger step sizes t_k whenever the algorithm comes in the situation described above. This ensures that either some significant descent towards the real minimum can be done or shows that the point where the algorithm is stuck is actually such a minimum. Noise attenuation is triggered when E_k or respectively the descent measure δ_k that is used for the descent test is negative. A more detailed description is given in [56].

3.1.3 Convergence Results

Depending on the kind of error many slightly different convergence results can be proven for bundle methods that handle convex objective functions with inexact information. In case of the general error defined in (3.2) and (3.3) it can be shown that for bounded sequences $\{\hat{x}^k\}$ every accumulation point $\bar{x}$ of an infinite series of serious steps or the last serious iterate before an infinite tail of null steps is a $\bar{\sigma}$-solution of the problem meaning that

$$f(\bar{x}) \leq f^* + \bar{\sigma}$$

with f^* being an exact solution of problem (3.1).

Generally for asymptotically vanishing errors it is possible to construct bundle methods very similar to the basic bundle method that converge to the exact minimum of the problem. For more detailed results refer to [56].

3.2 Nonconvex Bundle Methods with Exact Information

In the nonconvex case the optimization problem is the following:

$$\min_{x\in\mathbb{R}^n} f(x). \tag{3.5}$$

This time $f : \mathbb{R}^n \to \mathbb{R}$ is a finite, locally Lipschitz function. It is neither expected to be convex nor differentiable.

In the case of inexactness in convex bundle methods, where a lot of different assumptions can be put on the errors to reach different convergence results, the strategy to cope with these errors remains very much the same. In contrast to this in case of nonconvex objective functions the set of functions to be studied is rather uniform still there exist very different approaches to tackle the problem. As the nonnegativity property of the linearization errors e_j^k is crucial for the convergence proof of convex bundle methods an early idea was forcing the errors to be so by different downshifting strategies. A very common one is using the *subgradient locality measure* [19, 29]. Here the linearization error is essentially replaced by the nonnegative number

$$\tilde{e}_j^k := \max_{j\in J_k}\{|e_j^k|, \gamma\|\hat{x}^k - x^j\|^2\}$$

or a variation of this expression.

The expression gradient locality measure comes from the dual point of view, where the aggregate linearization error provides a measure for the distance of the calculated ε-subgradient to the objective function.

Methods that use downshifting for building the model function are often endowed with a line search to provide sufficient decrease of the objective function. For the linesearch to terminate finitely, usually semismoothness of the objective function is needed.

3.2.1 Proximity Control

Instead of using line search it is also possible to do *proximity control*. This means that the step size parameter t_k is managed in a smart way to ensure the right amount of decrease in the objective function. This method is very helpful in the case of nonconvex objective functions with inexact information as it is predominantly considered in this thesis.

As inexactness can be seen as a kind of slight nonconvexity one could be tempted to think that nonconvex bundle methods are destined to be extended to the inexact case. Indeed, the two existing algorithms [10, 37] that deal with both nonconvexity and inexactness are both extensions of a nonsmooth bundle method. This is however seldom possible for algorithms that employ a line search because for functions with inexact information convergence of this subroutine cannot be proven.

To this end proximity control seems to be a very promising strategy. It is used in many different variations in [1, 25, 36, 38, 39] and [48].

3.2.2 Other Concepts

In the beginning bundle methods were mostly explored from the dual point of view. Newer concepts focus also on the primal version of the method. This invokes for example having different model functions for the subproblem.

In [5, 6] the difference function

$$h(d) := f(x^j + d) - f(x^j) \quad j \in J_k$$

is approximated to find a descent direction of f. The negative linearization errors are addressed by using two different bundles. One containing the indices with nonnegative linearization errors and one containing the other ones. From these two bundles two cutting plane approximations can be constructed which provide the bases for the calculation of new iterates.

In [39] Noll et al. follow an approach of approximating a local model of the objective function. The model can be seen as a nonsmooth generalization of the Taylor expansion and looks the following:

$$\Phi(y, x) = \phi(y, x) + \frac{1}{2}(y - x)^\top Q(x)(y - x).$$

The so called *first order model* $\phi(., x)$ is convex but possibly nonsmooth and can be approximated by cutting planes. The *second order part* is a quadratic but not necessarily convex. The algorithm then proceeds a similarly to a general bundle algorithm. Instead of a line search it uses proximity control to ensure convergence.

Generally for all of this methods convergence to a stationary point is established under the assumptions of a locally Lipschitz objective function and bounded level sets $\{x \in \mathbb{R}^n | f(x) \leq f(\hat{x}^1)\}$. If the method uses a line search additionally semismoothness of the objective function is needed.

In [37] the second order approach of [39] is extended to functions with inexact information. As far a we know this is the only other bundle method that can deal with nonconvexity and inexactness in both the function value and subgradient. In this method a lower-$\mathcal{C}^1$ objective function and some assumptions on the inexactness are needed to prove convergence.

Noll's algorithm inspires the variable metric variation of the method used by Hare et al. in [10] that is presented in section 5 of this thesis.

4 Proximal Bundle Method for Nonconvex Functions with Inexact Information

This section focuses on the proximal bundle method presented by Hare et al. in [10]. The idea is to extend the basic bundle algorithm for nonconvex functions with both inexact function and subgradient information. The key idea of the algorithm is the one already developed by Hare and Sagastizábal in [9]: When dealing with nonconvex functions a very critiicalcal difference to the convex case is that the linearization errors are not necessarily nonnegative any more. To tackle this problem the errors are manipulated to enforce nonnegativity. In this case this is done my modeling not the objective function directly but a convexified version of it.

4.1 Derivation of the Method

Throughout this section we consider the optimization problem

$$\min_x f(x) \quad \text{s.t.} \quad x \in X. \tag{4.1}$$

The objective function $f : \mathbb{R}^n \to \mathbb{R}$ is locally Lipschitz and (subdifferentially) regular. $X \subseteq \mathbb{R}^n$ is assumed to be a convex compact set.

Definition 4.1 [44, Theorem 7.25] $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ is called *subdifferentially regular* at $\bar{x}$ if $f(\bar{x})$ is finite and the epigraph

$$epi(f) := \{(x, \alpha) \in \mathbb{R}^n \times \mathbb{R} | \alpha \geq f(x)\}$$

is Clarke regular at $\bar{x}, f(\bar{x})$.

4.1.1 Inexactness

It is assumed that both the function value as well as one element of the subdifferential can be provided in an inexact form. For the function value inexactness is defined straight forwardly: If

$$\|f_x - f(x)\| \leq \sigma_x$$

then f_x approximates the value $f(x)$ within σ_x. This is slightly different from the definition in (3.2). In the convex case it follows from (3.4) that $\bar{\sigma} \geq \sigma_x \geq -\theta_x \geq -\bar{\theta}$ and therefore $f_x \in [f(x) - \bar{\theta}, f(x) + \bar{\sigma}]$.

As the 'normal' ε-subdifferential is not defined for nonconvex functions we adopt the notation used in [37] and interpret inexactness in the following way: $g \in \mathbb{R}^n$ approximates a subgradient of $\partial f(x)$ within $\theta \geq 0$ if

$$g \in \partial f(x) + B_\theta(0) := \partial_{[\theta]} f(x)$$

where $\partial f(x)$ is the Clarke subdifferential of f.

The given definition of the inexactness can be motivated by the relation

$$g \in \partial_{[\theta]} f(x) \Leftrightarrow g \in \partial(f + \theta\| \cdot - x\|)(x)$$

noticed in [51]. It means that the approximation of the subgradient of $f(x)$ is an exact subgradient of a small perturbation of f at x. $\partial_{[\varepsilon]} f(x)$ is also known as the Fréchet ε-subdifferential of $f(x)$.

Remark: For convex objective functions this approximate subdifferential does *not* equal the usual convex ε-subdifferential. The two can however be related via

$$\partial_\theta f(x) \subset \partial_{[\theta']} f(x)$$

for a suitable θ'. Generally an explicit relation between θ and θ' is hard to find [37].

Like in the paper it is assumed that the errors are bounded although the bound does not have to be known:

$$|\sigma_j| \leq \bar{\sigma}, \bar{\sigma} > 0 \quad \text{and} \quad 0 \leq \theta_j \leq \bar{\theta} \quad \forall j \in J^k.$$

For ease of notation we write from now on f_j instead of f_{x^j} for the approximation of the function value at the j'th iterate in the bundle J. The approximation at the k'th stability center reads $\hat{f}_k$.

4.1.2 Nonconvexity

A main issue both nonconvexity and inexactness entail is that the linearization errors e_j^k are not necessarily nonnegative any more. So based on the results in [55] not the objective function but a convexified version of it is modeled as the objective function of the subproblem.

As already pointed out in 2.2 the bundle subproblem can be formulated by means of the prox-operator (2.9).

The key idea is to use the relation

$$prox_{T=\frac{1}{\eta}+t,f}(x) = prox_{t,f+\eta/2\|\cdot-x\|^2}(x).$$

This means, that the proximal point of the function f for parameter $T = \frac{1}{\eta} + t$, $\eta, t > 0$ is the same as the one of the convexified function

$$\tilde{f}(y) = f(y) + \frac{\eta}{2}\|y - x\|^2 \tag{4.2}$$

with respect to the parameter t [9]. η is therefore called the *convexification parameter* and t the *prox-parameter.*

The main difference of the method in [10] to the basic bundle method is that the function that is modeled by the cutting plane model s no longer the original objective function f but the convexified version $\tilde{f}$. This results in the following changes:

In addition to downshifting the linear functions forming the model they have a tilted slope. This is because instead of subgradients of the original objective f subgradients of the function $\tilde{f}$ are taken. We call them *augmented subgradients.* At the iterate x^j it is given by

$$s_j^k = g^j + \eta_k \left(x^j - \hat{x}^k \right).$$

Downshifting is done in a way that keeps the linearization error nonnegative. The *augmented linearization error* is therefore defined as

$$0 \leq c_j^k := e_j^k + b_j^k, \quad \text{with} \quad \begin{cases} e_j^k := \hat{f}_k - f_j - \left\langle g^j, \hat{x}^k - x^j \right\rangle \\ b_j^k := \frac{\eta_k}{2} \|x^j - \hat{x}^k\|^2 \end{cases}$$

and

$$\eta_k \geq \max \left\{ \max_{j \in J_k, x^j \neq \hat{x}^k} \frac{-2e_j^k}{\|x^j - \hat{x}^k\|^2}, 0 \right\} + \gamma.$$

The parameter $\gamma \geq 0$ is a safeguarding parameter to keep the calculations numerically stable.

The new model function can therefore be written as

$$M_k(\hat{x}^k + d) := \hat{f}_k + \max_{j \in J_k} \left\{ \left\langle s_j^k, d \right\rangle - c_j^k \right\}.$$

4.1.3 Aggregate Objects

The definition of the *augmented aggregate subgradient* S^k, *error* C_k and *linearization* A_k follows straightforwardly:

$$S^k := \sum_{j \in J_k} \alpha_j^k s_j^k, \tag{4.3}$$

$$C_k := \sum_{j \in J_k} \alpha_j^k c_j^k \tag{4.4}$$

$$A_k(\hat{x}^k + d) := M_k(x^{k+1}) + \left\langle S^k, d - d^k \right\rangle. \tag{4.5}$$

Just as the model decrease

$$\delta^k := C_k + t_k \|S^k + \nu^k\|^2 = C_k + \frac{1}{t_k} \|d^k\|^2, \tag{4.6}$$

which contains the normal vector

$$\nu^k \in \partial \mathtt{i}_X(x^{k+1}) \tag{4.7}$$

of the constraint set X.

The second formulation in (4.6) follows from the relation $d^k = -t_k(S^k + \nu^k)$.

By the same argumentation as for (2.17) the KKT conditions also reveal another useful characterization of the augmented aggregate linearization error:

$$C_k = \hat{f}_k - M_k(x^{k+1}) + \left\langle S^k, d^k \right\rangle \tag{4.8}$$

As the model function M_k is convex even for nonconvex objective functions it is still minorized by the aggregate linearization. It holds

$$A_K(\hat{x}^k + d) \leq M_k(\hat{x}^k + d) \quad \forall d \in \mathbb{R}^n. \tag{4.9}$$

The update of t_k can be done in the same way described in (2.20) and (2.21) for the basic bundle method. Similarly the methods to update the bundle index set J^k stay valid. The update conditions (2.18) and (2.19) for the model are now written with respect to the augmented aggregate linearization and the approximate function value $\hat{f}_{k+1}$.

$$M_{k+1}(\hat{x}^k + d) \geq \hat{f}_{k+1} - c_{k+1}^{k+1} + \left\langle s^{k+1}, d \right\rangle \quad \forall d \in \mathbb{R}^n \tag{4.10}$$

$$M_{k+1}(\hat{x}^k + d) \geq A_k(\hat{x}^k + d) \quad \forall d \in \mathbb{R}^n. \tag{4.11}$$

A bundle algorithm that deals with nonconvexity and inexact function and subgradient information can now be stated.

Algorithm 4.1: Nonconvex Proximal Bundle Method with Inexact Information

Select parameters $m \in (0, 1), \gamma > 0$ and a stopping tolerance $\mathtt{tol} \geq 0$.
Choose a starting point $x^1 \in \mathbb{R}^n$ and compute f_1 and g^1. Set the initial index set $J_1 := \{1\}$ and the initial prox-center to $\hat{x}^1 := x^1$, $\hat{f}_1 = f_1$ and select $t_1 > 0$.
For $k = 1, 2, 3, \ldots$

1. Calculate
$$d^k = \arg\min_{d\in\mathbb{R}^n} \left\{ M_k(\hat{x}^k + d) + \mathbb{I}_X(\hat{x}^k + d) + \frac{1}{2t_k}\|d\|^2 \right\}.$$

2. Set
$$\begin{aligned} G^k &= \sum_{j\in J_k} \alpha_j^k s_j^k \\ C_k &= \sum_{j\in J_k} \alpha_j^k c_j^k, \\ \delta_k &= C_k + \frac{1}{t_k}\|d^k\|^2. \end{aligned}$$

   If $\delta_k \leq$ `tol` → STOP.

3. Set $x^{k+1} = \hat{x}^k + d^k$.

4. Compute f^{k+1}, g^{k+1}.
   If
$$f^{k+1} \leq \hat{f}^k - m\delta_k \quad \rightarrow \text{ serious step}$$
   Set $\hat{x}^{k+1} = x^{k+1}, \hat{f}^{k+1} = f^{k+1}$ and select $t_{k+1} > 0$.
   Otherwise → nullstep
   Set $\hat{x}^{k+1} = \hat{x}^k, \hat{f}^{k+1} = f^{k+1}$ and choose $0 < t_{k+1} \leq t_k$.

5. Select new bundle index set J_{k+1}, calculate
$$\eta_k = \max\left\{ \max_{j\in J_{k+1}, x^j \neq \hat{x}^{k+1}} \frac{-2e_j^k}{|x^j - \hat{x}^{k+1}|^2}, 0 \right\} + \gamma$$
   and update the model M_k.

4.2 On Different Convergence Results

In terms of usability of the described algorithm it is interesting to see if stronger convergence results are possible if additional assumptions are put on the objective function. This is investigated in the following section.

4.2.1 The Constraint Set

The constraint set X ensures the boundedness of the sequence $\{\hat{x}^k\}$. This is not necessary if the objective function is assumed to have bounded level sets $\{x \in \mathbb{R}^n | f(x) \leq f(\hat{x}^1)\}$, an assumption commonly used when optimizing nonconvex functions. As the objective function is assumed to be continuous bounded level sets are compact. Additionally the descent test ensures that $f(\hat{x}^{k+1}) \leq f(\hat{x}^k)$ for all k. The proof holds therefore in the same way as with the set X.

Another possibility is to bound the step sizes t_k also from above. Then the sequence $\{\hat{x}^k\}$ stays bounded and the proof still holds. In [56] another stopping criterion is supposed that ensures convergence even for unbounded sequences $\{\hat{x}^k\}$. Is this also possible in my case or only for convex???

4.2.2 Exact Information and Vanishing Errors

As the presented algorithm was originally designed for nonconvex objective functions where function values as well as subgradients are available in an exact manner, all convergence results stay the same with the error bounds $\bar{\sigma} = \bar{\theta} = 0$. As already indicated previously this is the case because inexactness can be seen as a kind of nonconvexity and no additional concepts had to be added to the method when generalizing it to the inexact setting.

If we additionally require the objective function to be lower-$\mathcal{C}^2$ it can be proven that the sequence $\{\eta_k\}$ is bounded [9]. This is not possible in the case of inexact information even for convex objective functions.

For asymptotically vanishing errors, meaning $\lim_{k\to\infty} \sigma_k = 0$ and $\lim_{k\to\infty} \theta_k = 0$ the convergence theory holds equally well with error bounds $\bar{\sigma} = \bar{\theta} = 0$ in [10, Lemma 5]. Still it is difficult if not impossible to show that the sequence $\{\eta_k\}$ is bounded without further assumptions. Under the assumption that f is lower-$\mathcal{C}^2$ and some continuity bounds on the errors

$$\frac{|\sigma_j - \hat{\sigma}_k|}{\|x^j - \hat{x}^k\|^2} \leq L_\sigma, \qquad \frac{\theta_j}{\|x^j - \hat{x}^k\|} \leq L_\theta \quad \forall k \text{ and } \forall j \in J_k$$

boundedness of the sequence $\{\eta_k\}$ can be shown. The question remains however if those assumptions are possible to be assured in practice.

remark on η_k? how does it behave in my applications???

4.2.3 Convex Objective Functions

An obvious gain when working with convex objective functions is that the approximate stationarity condition of [10, Lemma 5 (iii)] is now an approximate optimality condition.

If one takes the error definitions (3.2) and (3.3) that are available in the convex case and assumes $X = \mathbb{R}^n$ statement (22) in [10] therefore means that

$$0 \in \partial_{\bar{\sigma}+\bar{\theta}} f(\bar{x}).$$

Thus $\bar{x}$ is $(\bar{\sigma}+\bar{\theta})$-optimal.

This follows from the definition of S^k in (4.3) and local Lipschitz continuity of the ε-subdifferential [44, Proposition 12.68].

beweis für $\bar{\sigma}$-optimalität

bounded t_k instead of D? better????

To conclude this section we can say: At the moment there exist two fundamentally different approaches to tackle inexactness in various bundle methods depending on if the method is developed for convex or nonconvex objective functions. In the nonconvex case inexactness is only considered in the paper by Hare, Sagastizàbal and Sodolov [10] presented above and Noll [37]. In these cases the inexactness can be seen as an additional nonconvexity. In practice this means that the algorithm can be taken from the nonconvex case with no or only minor changes. This includes that all results of the exact case remain true as soon as function and subgradient are evaluated in an exact way. In case of convex objective functions with inexact information stronger convergence results are possible. However to be able to exploit convexity in order to achieve those results the algorithms look different from those designed for nonconvex objective functions and are generally not able to deal with such functions.

5 Variable Metric Bundle Method

A way to extend the proximal bundle method is to use an arbitrary metric $\frac{1}{2}\langle d, W_k d\rangle$ with a symmetric and positive definite matrix W_k instead of the Euclidean metric for the stabilization term $\frac{1}{2t_k}\|d\|^2$. Methods doing so are called *variable metric bundle methods*. This section combines the method of Hare et al. presented in section 4 with the second order model function used by Noll in [37] to a metric bundle method suitable for nonconvex functions with noise.

The section starts by explaining the ideas from [37] used to extend the method presented above. It then gives an explicit strategy how to update the metric during the steps of the algorithm and concludes with a convergence proof for the developed method.

Throughout this section we still consider the optimization problem (4.1). We also keep the names and definitions of the objects used in section 4.

5.1 Main Ingredients to the Method

As already mentioned in section 2 the stabilization term can be interpreted in many different ways. In the context of this section we can understand it as a pretty rough approximation of the curvature of the objective function. Of course bundle methods are designed to work with non differentiable objectives so it cannot be expected that the function provides any kind of curvature. However, if it has regions where there is curvature, this information can be used to speed up convergence.

5.1.1 Variable Metric Bundle Methods

Variable metric bundle methods use an approach that can be motivated by the thoughts stated above. Instead of using the Euclidean norm for the stabilization term $\frac{1}{2}\|d\|^2$ the metric is derived from a symmetric and positive definite matrix W_k. As the name of the method suggests, this matrix can vary over the iterations of the algorithm. The subproblem in the k'th iteration therefore reads

$$\min_{\hat{x}^k+d\in\mathbb{R}^n} M_k(\hat{x}^k+d)+\mathtt{i}_X(\hat{x}^k+d)+\frac{1}{2}\langle d, W_k d\rangle \,.$$

As explained in [23] like (2.8) this is a Moreau-Yosida regularization of the objective function (on the constraint set), so this subproblem is still strictly convex and has a unique solution. It is however harder to solve especially if the matrices W_k are no diagonal matrices [27]. In the unconstrained case or for a very simple constraint set the subproblem can be solved by calculating a quasi Newton step. Such a method is presented by Lemaréchal and Sagastizábal in [24] for convex functions. Lukšan and Vlček use an algorithm in those lines in [54] which is adapted to a limited memory setting by Haarala et al. in [8].

A challenging question is how to update the matrices W_k. It is important that the updating strategy preserves positive definiteness of the matrices and that the matrices stay bounded. The updates that are used most often are the symmetric rank 1 formula (SR1 update) and the BFGS (Broyden-Fletcher-Goldfarb-Shanno) update. These updates make it possible to assure the required conditions with only little extra effort even in the nonconvex case. Concrete instances of the updates are given in [54] and [23].

5.1.2 Noll's Second Order Model

In [39] Noll et al. present a proximal bundle method for nonconvex objective functions. An important ingredient to the method is that not the objective function itself is approximated in the subproblem but a quadratic model of it:

$$\Phi(x, \hat{x}) = \phi(x, \hat{x}) + \frac{1}{2}\langle x - \hat{x}, Q(\hat{x})(x - \hat{x})\rangle \tag{5.1}$$

The first order model $\phi(\cdot, \hat{x})$ is convex and possibly nonsmooth. The second order part $\frac{1}{2}\langle \cdot - \hat{x}, Q(\hat{x})(\cdot - \hat{x})\rangle$ is quadratic but not necessarily convex.

As the first order part of this model is convex it can be approximated by a cutting plane model just like the objective function in usual convex bundle methods. The subproblem emerging from this approach is

$$\min_{\hat{x}^k + d} m(\hat{x}^k + d) + \frac{1}{2}\left\langle d, Q(\hat{x}^k)d \right\rangle + \frac{1}{2t_k}\|d\|^2$$

where m_k is the cutting plane model (2.2) for the nonsmooth function ϕ.

The matrix $Q(\hat{x})$ itself does not have to be positive definite. In fact the only conditions put on this matrix are that it is symmetric and that all eigenvalues are bounded. We adopt the notation in [37] and write

$$Q(\hat{x}^k) := Q_k = Q_k^\top \quad \text{and} \quad -q\mathbb{I} \prec Q_k \prec q\mathbb{I} \text{ for } q > 0.$$

The notation $A \prec B$ with $A, B \in \mathbb{R}^{n \times n}$ means that the matrix $(B - A)$ is positive definite.

As the matrix Q_k is symmetric it can also be pulled into the stabilization term. The k'th bundle subproblem then is

$$\min_{\hat{x}^k + d \in X} M_k(\hat{x}^k + d) + \frac{1}{2}\left\langle d, \left(Q_k + \frac{1}{t_k}\mathbb{I}\right) d \right\rangle . \tag{5.2}$$

If $W_k = Q_k + \frac{1}{t_k}\mathbb{I}$ is positive definite, this is a variable metric subproblem.

The decomposition of the stabilization term into a curvature approximation and a proximal term makes is easier to reach two goals at the same time:

One the one hand, curvature of the objective can be approximated only under the conditions of the boundedness and symmetry of Q_k. No positive definiteness has to be ensured for convergence. On the other hand the proximal term can be used in the trust region inspired way to make a line search obsolete. As already mentioned in section 3 this is an advantage especially when working with inexact functions where a line search is not useable.

comment on line search and curve search in [23, 24, 54]?

5.1.3 The Descent Measure

Due to the different formulation of the subproblem (5.2) the descent measure δ_k has to be adapted in the varible metric bundle method. In the same way as for (2.15) from the optimality condition

$$0 \in \partial M_k(x^{k+1}) + \partial \mathtt{i}_D(x^{k+1}) + \left(Q + \frac{1}{t_k}\mathbb{I}\right) d^k$$

follows that

$$S^k + \nu^k = -\left(Q + \frac{1}{t_k}\mathbb{I}\right) d^k. \tag{5.3}$$

S^k and ν^k being the augmented aggregate subgradient and outer normal defined in (4.3) and (4.7) respectively.

From this the model decrease (4.6) can be recovered using (4.5), (4.8) and (5.3):

$$\begin{aligned} \delta_k &= \hat{f}_k - M_k(x^{k+1}) - \left\langle \nu^k, d^k \right\rangle \\ &= \hat{f}_k - A_k(x^{k+1}) - \left\langle \nu^k, d^k \right\rangle \\ &= C_k - \left\langle S^k + \nu^k, d^k \right\rangle \\ &= C_k + \left\langle d^k, \left(Q + \frac{1}{t_k}\mathbb{I}\right) d^k \right\rangle . \end{aligned} \tag{5.4}$$

The new δ_k is used in the same way as in algorithm 4.1 for the descent test and stopping conditions.

Because the changes in the algorithm concern only the stabilization and the decrease measure δ_k all other relations that were obtained for the different parts of the model M_k in section 4 are still valid.

5.2 The Variable Metric Bundle Algorithm

The variable b=metric bundle algorithm can now be stated as a varaiation of algorithm 4.1.

same form as Hare algorithm (nullstep)
add Q calculation

Algorithm 5.1: Nonconvex Variable Metric Bundle Method with Inexact Information

---

Select parameters $m \in (0,1), \gamma > 0$ and a stopping tolerance $\texttt{tol} \geq 0$.
Choose a starting point $x^1 \in \mathbb{R}^n$ and compute f_1 and g^1. Set the initial metric matrix $Q = \mathbb{I}$, the initial index set $J_1 := \{1\}$ and the initial prox-center to $\hat{x}^1 := x^1$, $\hat{f}_1 = f_1$ and select $t_1 > 0$.
For $k = 1, 2, 3, \dots$

1. Calculate

$$d^k = \arg\min_{d \in \mathbb{R}^n} \left\{ M_k(\hat{x}^k + d) + \mathbb{I}_X(\hat{x}^k + d) + \frac{1}{2} d^\top \left(Q + \frac{1}{t_k} \mathbb{I} \right) d \right\}.$$

2. Set

$$\begin{aligned} G^k &= \sum_{j \in J_k} \alpha_j^k s_j^k, \\ C_k &= \sum_{j \in J_k} \alpha_j^k c_j^k, \\ \delta_k &= C_k + (d^k)^\top \left(Q + \frac{1}{t_k} \mathbb{I} \right) d^k. \end{aligned}$$

   If $\delta_k \leq \texttt{tol} \rightarrow$ STOP.

3. Set $x^{k+1} = \hat{x}^k + d^k$.

4. Compute f^{k+1}, g^{k+1}.
   If

$$f^{k+1} \leq \hat{f}^k - m\delta_k \quad \rightarrow \text{ serious step}$$

   Set $\hat{x}^{k+1} = x^{k+1}, \hat{f}^{k+1} = f^{k+1}$ and select $t_{k+1} > 0$.
   Calculate $Q(\hat{x}^k)$ ... Otherwise $\rightarrow$ nullstep
   Set $\hat{x}^{k+1} = \hat{x}^k, \hat{f}^{k+1} = f^{k+1}$ and choose $0 < t_{k+1} \leq t_k$.

5. Select new bundle index set J_{k+1}, keeping all active elements. Calculate

$$\eta_k = \max\left\{\max_{j\in J_{k+1}, x^j \neq \hat{x}^{k+1}} \frac{-2e_j^k}{|x^j - \hat{x}^{k+1}|^2}, 0\right\} + \gamma$$

and update the model M^k.

5.3 Convergence Analysis

ausfuhrungen im Beweis genauer

In this section the convergence properties of the new method are analyzed. We do this the same way it is done by Hare et al. in [10].

In the paper all convergence properties are first stated in [10, Lemma 5]. It is then shown that all sequences generated by the method meet the requirements of this lemma which we repeat here for convenience.

Lemma 5.1 ([10, Lemma 5]) *Suppose that the cardinality of the set $\{j \in J^k | \alpha_j^k > 0\}$ is uniformly bounded in k.*

(i) If $C^k \to 0$ as $k \to \infty$, then

$$\sum_{j\in J^k} \alpha_j^k \|x^j - \hat{x}^k\| \to 0 \text{ as } k \to \infty.$$

(ii) If additionally for some subset $K \subset \{1, 2, \dots\}$,

$$\hat{x}^k \to \bar{x}, S^k \to \bar{S} \text{ as } K \ni k \to \infty, \text{ with } \{\eta_k | k \in K\} \text{ bounded,}$$

then we also have

$$\bar{S} \in \partial f(\bar{x}) + B_{\bar{\theta}}(0).$$

(iii) If in addition $S^k + \nu^k \to 0$ as $K \in k \to \infty$, then $\bar{x}$ satisfies the approximate stationarity condition

$$0 \in (\partial f(\bar{x}) + \partial \mathtt{i}_X(\bar{x})) + B_{\bar{\theta}}(0). \tag{5.5}$$

(iv) Finally if f is also lower-$\mathcal{C}^1$, then for each $\varepsilon > 0$ there exists $\rho > 0$ such that

$$f(y) \geq f(\bar{x}) - (\bar{\theta} + \varepsilon)\|y - \bar{x}\| - 2\bar{\sigma}, \quad \text{for all } y \in X \cup B_\rho(\bar{x}). \tag{5.6}$$

As neither the stabilization nor the descent test is involved in the proof of Lemma 5.1 it is the same as in [10].

We prove now that also the variable metric version of the algorithm fulfills all requirements of Lemma 5.1. The proof is divided into two parts. The first case covers the case of infinitely many serious steps, the second one considers infinitely many null steps.

For both proofs the following lemma is needed:

Lemma 5.2 *For a symmetric matrix $A \in \mathbb{R}^{n \times n}$, a vector $d \in \mathbb{R}^n$ and $\xi > 0$ the following result holds:*

$$A \prec \xi \mathbb{I} \Rightarrow Ad < \xi d.$$

The second inequality is considered componentwise.

Proof: As the matrix A is real and symmetric it is orthogonally diagonalizeable. There exist eigenvalues $\lambda_i \in \mathbb{R}, i = \{1, ..., n\}$ and corresponding eigenvectors $v^i \in \mathbb{R}^n, i = \{1, ..., n\}$ that satisfy the equations

$$Av^i = \lambda_i v^i \quad i = \{1, ..., n\}.$$

The eigenvectors v^i generate a basis for $\mathbb{R}^n$ so any vector $d \in \mathbb{R}^n$ can be written as

$$d = \sum_i \alpha_i v^i$$

for $\alpha_i \in \mathbb{R}^n, i = \{1, ..., n\}$.

This yields

$$Ad = A \sum_i \alpha_i v^i = \sum_i \alpha_i \lambda_i v^i. \tag{5.7}$$

Plugging the assumption $A \prec \xi \mathbb{I}$ which is equivalent to $\max_i \lambda_i < \xi$ into (5.7) we get relation (5.3) by

$$Ad < \xi \sum_i \alpha_i v^i = \xi d.$$

□

Theorem 5.3 (c.f.[10, Theorem 6]) *Let the algorithm generate and infinite number of serious steps. Then $\delta_k \to 0$ as $k \to \infty$.*

Let the sequence $\{\eta_k\}$ be bounded. If $\liminf_{k\to\infty} t_k > 0$ *then as* $k \to \infty$ *we have* $C_k \to 0$*, and for every accumulation point* $\bar{x}$ *of* $\{\hat{x}^k\}$ *there exists* $\bar{S}$ *such that* $S^k \to \bar{S}$ *and* $S^k + \nu^k \to 0$.
In particular if the cardinality of $\{j \in J^k | \alpha_j^k > 0\}$ *is uniformly bounded in* k *then the conclusions of Lemma 5.1 hold.*

The proof is very similar to the one stated in [10] but minor changes have to be made due to the different formulation of the nominal decrease δ_k.

Proof: At each serious step we have

$$\hat{f}_{k+1} \leq \hat{f}_k - m\delta_k \tag{5.8}$$

where $m,\ \delta_k > 0$. From this follows that the sequence $\{\hat{f}_k\}$ is nonincreasing. Since $\{\hat{x}^k\} \subset X$ and f is continuous the sequence $f(\hat{x}^k)$ is bounded. With $|\sigma_k| < \bar{\sigma}$ the sequence $\{f(\hat{x}^k) + \sigma_k\} = \{\hat{f}_k\}$ is bounded below. Together with the fact that $\{\hat{f}_k\}$ is nonincreasing one can conclude that it converges.
Using (5.8), one obtains

$$0 \leq m \sum_{k=1}^{l} \delta_k \leq \sum_{k=1}^{l} \left(\hat{f}_k - \hat{f}_{k+1}\right),$$

so letting $l \to \infty$,

$$0 \leq m \sum_{k=1}^{\infty} \delta_k \leq \hat{f}_1 - \underbrace{\lim_{k\to\infty} \hat{f}_k}_{\neq \pm\infty}.$$

This yields

$$\sum_{k=1}^{\infty} \delta_k = \sum_{k=1}^{\infty} \left(C^k + (d^k)^\top \left(Q + \frac{1}{t_k}\mathbb{I}\right) d^k\right) < \infty.$$

Hence, $\delta_k \to 0$ as $k \to \infty$. All quantities above are nonnegative due to positive definiteness of $Q + \frac{1}{t_k}\mathbb{I}$, so it also holds that

$$C_k \to 0 \quad \text{and} \quad (d^k)^\top \left(Q + \frac{1}{t_k}\mathbb{I}\right) d^k \to 0.$$

For any accumulation point $\bar{x}$ of the sequence $\{\hat{x}^k\}$ the corresponding subsequence $d^k \to 0$ for $k \in K \subset \{1, 2, ...\}$. As $\liminf_{k\to\infty} t_k > 0$ and the eigenvalues of Q are bounded the whole expression

$$S^k + \nu^k = \left(Q + \frac{1}{t_k} I\right) d^k \to 0 \quad \text{for} \quad k \in K.$$

And from local Lipschitz continuity of f follows then that $S^k \to \bar{S}$ for $k \in K$.

□

For the case of infinitely many null steps we need result (31) from [10]. It only depends on the definitions of the augmented linearization error and subgradient.

Whenever x^{k+1} is as declared a null step, the relation

$$-c_{k+1}^{k+1} + \left\langle s_{k+1}^{k+1}, x^{k+1} - \hat{x}^k \right\rangle \geq -m\delta_k \tag{5.9}$$

holds.

Theorem 5.4 (c.f. [10, Theorem 7]) *Let a finite number of serious iterates be followed by infinite null steps. Let the sequence $\{\eta_k\}$ be bounded and $\liminf_{k\to\infty} t_k > 0$.*
Then $\{x^k\} \to \hat{x}$, $\delta_k \to 0$, $C_k \to 0$, $S^k + \nu^k \to 0$ and there exist $K \subset \{1, 2, ...\}$ and $\bar{S}$ such that $S^k \to \bar{S}^k$ as $K \ni k \to \infty$.
In particular if the cardinality of $\{j \in J^k | \alpha_j^k > 0\}$ is uniformly bounded in k then the conclusions of Lemma 5.1 hold for $\bar{x} = \hat{x}$.

Proof: Let k be large enough such that $k \geq \bar{k}$, where $\bar{k}$ is the iterate of the last serious step. Let $\hat{x}^k = \hat{x}$ and $\hat{f}_k = \hat{f}$ be fixed. Define the optimal value of the subproblem (5.2) by

$$\Psi_k := M_k(x^{k+1}) + \frac{1}{2}\left(d^k\right)^\top \left(Q + \frac{1}{t_k}\mathbb{I}\right) d^k. \tag{5.10}$$

It is first shown that the sequence $\{\Psi_k\}$ is bounded above. From definition (4.5) follows

$$A_k(\hat{x}) = M_k(x^{k+1}) - \langle S^k, d^k \rangle.$$

Using (5.3) for the second equality, the subgradient inequality for $\nu^k \in \partial \mathtt{i}_D$ in the first inequality and (4.9) for the second inequality one obtains

$$
\begin{aligned}
\Psi^k + \frac{1}{2}\left(d^k\right)^\top \left(Q + \frac{1}{t_k}\mathbb{I}\right) d^k &= A_k(\hat{x}) + \langle S^k, d^k\rangle + \left(d^k\right)^\top \left(Q + \frac{1}{t_k}\mathbb{I}\right) d^k \\
&= A_k(\hat{x}) - \langle \nu^k, k\rangle \\
&\leq A(\hat{x}) \\
&\leq M_k(\hat{x}) \\
&= \hat{f}.
\end{aligned}
$$

By boundedness of d^k and $Q + \frac{1}{t_k}\mathbb{I}$ this yields that $\Psi_k \leq \hat{f}$, so the sequence $\{\Psi_k\}$ is bounded above. In the next step is shown that $\{\Psi_k\}$ is increasing. For this we obtain

$$
\begin{aligned}
\Psi_{k+1} &= M_k(x^{k+2}) + \frac{1}{2}\left(d^{k+1}\right)^\top \left(Q + \frac{1}{t_k}\mathbb{I}\right) d^{k+1} \\
&\geq A_k(x^{k+2}) + \frac{1}{2}\left(d^{k+1}\right)^\top \left(Q + \frac{1}{t_k}\mathbb{I}\right) d^{k+1} \\
&= M_k(x^{k+1}) + \langle S^k, x^{k+2} - x^{k+1}\rangle + \frac{1}{2}\left(d^{k+1}\right)^\top \left(Q + \frac{1}{t_k}\mathbb{I}\right) d^{k+1} \\
&= \Psi_k - \frac{1}{2}\left(d^k\right)^\top \left(Q + \frac{1}{t_k}\mathbb{I}\right) d^k + \frac{1}{2}\left(d^{k+1}\right)^\top \left(Q + \frac{1}{t_k}\mathbb{I}\right) d^{k+1} \\
&\qquad - \left(d^k\right)^\top \left(Q + \frac{1}{t_k}\mathbb{I}\right)\left(d^{k+1} - d^k\right) - \langle \nu^k, x^{k+2} - x^{k+1}\rangle \\
&\geq \Psi_k + \frac{1}{2}\left(d^{k+1} - d^k\right)^\top \left(Q + \frac{1}{t_k}\mathbb{I}\right)\left(d^{k+1} - d^k\right) \\
&= \Psi_k + \frac{1}{2}\underbrace{\|d^{k+1} - d^k\|_{Q + \frac{1}{t_k}\mathbb{I}}}_{\geq 0},
\end{aligned}
$$

where the first inequality comes from (4.9) and the fact that $t_{k+1} \leq t_k$ for null steps. The second equality follows from (4.5), the third equation by (5.3) and (5.10) and the last inequality holds by $\nu^k \in \partial \mathtt{i}_X(x^{k+1})$.

As Q is fixed in null steps and $\liminf_{k\to\infty} t_k > 0$ the sequence$\{\Psi_k\}_{k\in\mathbb{N}}$ is increasing and bounded from above. The sequence is therefore convergent. Taking into account that $1/t_k \geq 1/t_{\bar{k}}$, it therefore follows that

$$
\|d^{k+1} - d^k\| \to 0, \quad k \to \infty. \tag{5.11}
$$

By the first line in (5.4) and the fact that the augmented aggregate error can be expressed as

$$C_k = \hat{f} - M_k(x^{k+1}) + \left\langle S^k, d^k \right\rangle$$

by the KKT conditions follows

$$\begin{aligned}\hat{f} &= \delta_k + M_k(\hat{x}) - C_k - \left(d^k\right)^\top \left(Q + \frac{1}{t_k}\mathbb{I}\right)\left(d^k\right)\\ &= \delta_k + M_k(x^{k+1}) - \langle S^k, d^k\rangle - \left(d^k\right)^\top \left(Q + \frac{1}{t_k}\mathbb{I}\right)\left(d^k\right)\\ &= \delta_k + M_k(\hat{x} + d^k) + \left\langle \nu^k, d^k \right\rangle\\ &\geq \delta_k + M_k(\hat{x} + d^k)\end{aligned}$$

Where the last inequality is given by $\nu^k \in \partial \mathtt{i}_X(x^{k+1})$. Therefore

$$\delta^{k+1} \leq \hat{f} - M_{k+1}(\hat{x} + d^{k+1}). \tag{5.12}$$

By assumption (4.10) on the model, written for $d = d^{k+1}$,

$$-\hat{f}_{k+1} + c_{k+1}^{k+1} - \left\langle s_{k+1}^{k+1}, d^{k+1} \right\rangle \geq -M_{k+1}(\hat{x} + d^{k+1}).$$

In the null step case it holds $\hat{f}_{k+1} = \hat{f}$ so adding condition (5.9) to the inequality above, one obtains that

$$m\delta_k + \left\langle s_{k+1}^{k+1}, d^k - d^{k+1} \right\rangle \geq \hat{f} - M_{k+1}(\hat{x} + d^{k+1}).$$

Combining this relation with (5.12) yields

$$0 \leq \delta_{k+1} \leq m\delta_k + \left\langle s_{k+1}^{k+1}, d^k - d^{k+1} \right\rangle.$$

Because $m \in (0, 1)$ and $\left\langle s_{k+1}^{k+1}, d^k - d^{k+1} \right\rangle \to 0$ as $k \to \infty$ due to (5.11) and the boundedness of $\{\eta_k\}$ using [42, Lemma 3, p.45] it follows from (5.3) that

$$\lim_{k\to\infty} \delta_k = 0.$$

From formulation (5.4) of the model decrease follows that $C_k \to 0$ as $k \to \infty$. Since $Q + \frac{1}{t_k}\mathbb{I} \succ \xi\mathbb{I}$ due to $\liminf_{k\to\infty} t_k > 0$ and the bounded eigenvalues of Q we have

$$\xi \left(d^k\right)^\top d^k \leq \left(d^k\right)^\top \left(Q + \frac{1}{t_k}\mathbb{I}\right) d^k \to 0$$

This means that $d^k \to 0$ for $k \to \infty$ and therefore $\lim_{k\to\infty} x^k = \hat{x}$. It also follows that $\|S^k + \nu^k\| \to 0$ as $k \to \infty$. Passing to some subsequence if necessary we can conclude that S^k converges to some $\bar{S}$ and as $\hat{x}^k = \bar{x}$ for all k all requirements of Lemma 5.1 are fulfilled.

□

Remark: All results deduced in section 4.2 are still valid for this algorithm as they do not depend on the kind of stabilization used.

5.4 Updating the Metric

In [39] and [37] it is not specified how the matrix Q_k is to be chosen. For convergence it is necessary that the eigenvalues of Q_k are bounded. Additionally the matrix $Q_k + \frac{1}{t_k}\mathbb{I}$ has to be positive definite. Here we present two possible versions to update the metric matrix Q_k that fulfill both conditions.

Both updates are based on the usual BFGS-update formula (named after Broyden, Goldfarb, Fletcher and Shanno)

$$Q_{k+1} = Q_k + \frac{y^k {y^k}^\top}{\langle y^k, d^k\rangle} - \frac{Q_k d^k (Q_k d^k)^\top}{\langle d^k, Q_k d^k\rangle}. \tag{5.13}$$

Usually y^k is defined as the difference of the last two gradients of f. To adapt the formula to the nondifferentiable case the difference $y^k := g^{k+1} - g^k$ of two subgradients of f is taken instead as proposed in [8]. The starting matrix $Q_1 = \mathbb{I}$.

By definition the BFGS update is symmetric. To assure boundedness of the matrix Q_{k+1} the updates are manipulated in the two following ways:

5.4.1 Scaling of the Whole Matrix

A simple way to keep the absolute value all of eigenvalues of the constructed matrix Q_k below some threshold $0 < q < \infty$ is to scale the whole matrix down as soon as the absolute value of one eigenvalue is larger than this number. To do this define $\lambda_{max} := \max\{|\lambda_i| \mid \lambda_i \text{ is eigenvalue of } Q_k|\}$. If $\lambda_{max} > q$, set $Q_k = \frac{q}{\lambda_{max}} Q_k$. This way the absolute value of all eigenvalues is always smaller or equal to q. An advantage of this method is besides its simplicity that by scaling the whole matrix the ratio of the eigenvalues of Q_k is preserved. Scaling of Q_k corresponds to shrinking the whole quadratic function and in this way also the 'ratio of curvature' at different points of the graph stays the same.

5.4.2 Adaptive Scaling of Single Eigenvalues

This second method is motivated by the following observation: The variable metric bundle algorithm is to be used for nonsmooth functions. This means that the objective function has some kinks. For locally Lipschitz functions the number of kinks is finite and in between the kinks the function is smooth. This means, that there is indeed curvature information present in the smooth parts of the functions. At the kinks however the curvature is not defined. Taking a look at the one-sided differential quotient for $x \in \mathbb{R}$ shows that it diverges at such points: Let a kink be at $x_{\text{kink}} \in \mathbb{R}$ and the left sided limiting value of the derivative be $f'(x_{\text{kink}}) = a$ the right-sided one $f'(x_{\text{kink}}) = b \neq a$. Because f was assumed to be locally Lipschitz both limits exist and are finite. The following quotient can then be stated:

$$\lim_{h \searrow 0} \frac{f'(x_{\text{kink}} + h) - f'(x_{\text{kink}})}{h} = \frac{\overbrace{f'(x_{\text{kink}} + h)}^{\to b} - a}{h} = \pm\infty,$$

the sign depending on if $a > b$ or vice versa.
In more dimensions this is the same for the components of the Hessian corresponding to the direction where the kink occurs. This supports also to the intuitive thought that at a kink the slope changes 'infinitly fast'. Numerically the BFGS-update (5.13) can result in very large values for the entries of Q_k corresponding to points near the kink.

On the other hand due to the local Lipschitz property the slope of the objective function is always finite on closed sets. This means that there exists a neighborhood $B_\varepsilon(x_{\text{kink}})$ where the function f behaves similar to the scaled modulus $a|\cdot|$, $a \in \mathbb{R}$, in the direction perpendicular to the kink. Therefore in this neighborhood almost no curvature is present.

Summarized this means that on the one hand, the matrix Q_k should be close to zero in the components representing the directions perpendicular to the kink as soon as the iterates approach x_{kink}. But contrary to that the method that constructs Q_k can give very high values for those components.

The idea is now to scale only those eigenvalues of Q_k that are especially large. To lower the probability that eigenvalues are chosen which represent a really existing large curvature of of the objective function at the observed point, the change of the eigenvalues is viewed in relation to the step size d^k. As the eigenvalues of a matrix $A \in \mathbb{R}^n$ depend continuously on the entries of the matrix (see for example [57, Theorem 1.2]) the change in the eigenvalues is calculated by ordering them by size and then calculating $\text{diff}_{\lambda_i} = |\lambda_i^{k-1} - \lambda_i^k|, i = 1, ..., n$, where λ^{k-1} and $\lambda^k \in \mathbb{R}^n$ are vectors with the sorted eigenvalues of Q_{k-1} and Q_k respectively as components. If $\frac{\text{diff}_{\lambda_i}}{\|d^k\|} > q$ for a threshold $q < \infty$ and any $i = 1, ..., n$, indicating that the i'th eigenvalue changes much while only a small step is done, the i'th eigenvalue is set to $\lambda_i^k = \frac{1}{2}\lambda_1^{k-1}$ (remember that we stated before that near a kink there is often a very small curvature, hence halving the eigenvalue of the matrix of the step before).

Because this method considers only the relative change of the eigenvalues, it is still possible that one or more of them is larger than the threshold q. If this is the case, the whole matrix can be scaled as described above.

Remark: There appear many parameters to control the scaling of the metric update. Although these parameters were not especially tuned in this thesis they have a considerable impact on the convergence speed of the method also depending on the objective function used. This has to be kept in mind when implementing the method in practice.

5.4.3 Other Updating Possibilities

There are certainly many other possibilities to update the metric Q_k. A third variation based on BFGS-updates is the limited memory update suggested in [35]. If the update is skipped whenever $\|\rho s s^\top\| = \frac{\|d^k\|}{\|y^k\|} > q$ the matrix Q_k stays bounded. This strategy is also supported by the fact that if $\frac{\|d^k\|}{\|y^k\|} > q$ the change in the subgradient relative to the step size is rather small indicating that the current iterate lies within a region with only small changes in curvature. In such regions the update can be skipped.

Another updating method is using the symmetric-rank-1 (SR1) update

$$Q_k = Q_{k-1} + \frac{(y^k - Q_k d^k)(y^k - Q_k d^k)^\top}{\langle y^k - Q_k d^k, d^k \rangle}.$$

Boundedness can be assured in the same as for the normal BFGS update.

In [8] where the metric matrix is updated also in null steps a BFGS update is used in serious steps and an SR1 update in null steps.

As a last remark on this topic we want to say that although in this thesis the adaption of update strategies originally developed to be used with gradients seems to work in the presented setting also with subgradients this does not always have to be the case. Although it is argued for example in [26] that locally Lipschitz functions are differentiable almost everywhere and with an adequate linesearch it is improbable to arrive at an iterate that is a nondifferentiable point of the objective function this can still happen. Especially if such a linesearch is not used like in the algorithm presented here. So despite the promising practical behavior this area is still open to research.

- Q only updated in serious steps - why?
- comment in SR1 update? Null steps?

5.5 Numerical Testing

To compare the proximal bundle algorithm 4.1 with its variable metric variant algorithm 5.1 it is tested on some academic test functions and on a set of lower-$\mathcal{C}^2$ functions in different dimensions. The tests were done with the following parameters given in [10]: $m = 0.05$, $\gamma = 2$ and $t_0 = 0.1$. The chosen stopping tolerance was $\texttt{tol} = 10^{-6}$. If the algorithms did not meet the stopping condition after $250n$ steps for $x \in \mathbb{R}^n$, they were terminated. Contrary to [10] the stopping test was taken as given in the algorithm and the tolerance was not multiplied by $1 + \hat{f}_k$. The proximity control parameters κ_- and κ_+ from (2.21) and (2.20) respectively were chosen as $\kappa_- = 0.8$ and $\kappa_+ \in \{1.2, 2\}$.

-> check what is missing of introduchtion

-> introduction of noise forms

To test the performance for inexact function and subgradient values different types of noise were introduced. This was done by adding randomly generated elements with norm less or equal to σ_k and θ^k to the exact values $f(x^{k+1})$ and $g(x^{k+1})$ respectively.

Five different forms of noise were tested:

- N_0: No noise, $\bar{\sigma} = \sigma_k = 0$ and $\bar{\theta} = \theta^k = 0$ for all k,
- $\mathrm{N}_{\mathrm{c}}^{\mathrm{f,g}}$: Constant noise, $\bar{\sigma} = \sigma_k = 0.01$ and $\bar{\theta} = \theta^k = 0.01$ for all k,
- $\mathrm{N}_{\mathrm{v}}^{\mathrm{f,g}}$: Vanishing noise, $\bar{\sigma} = 0.01, \sigma_k = \min\{0.01, \|x^k\|/100\}$ and $\bar{\theta} = 0.01, \theta_k = \min\{0.01, \|x^k\|/100\}$ for all k,
- $\mathrm{N}_{\mathrm{c}}^{\mathrm{g}}$: Constant subgradient noise, $\bar{\sigma} = \sigma_k = 0$ and $\bar{\theta} = \theta_k = 0.01$ for all k and
- $\mathrm{N}_{\mathrm{v}}^{\mathrm{g}}$: Vanishing subgradient noise, $\bar{\sigma} = \sigma_k = 0$ and $\bar{\theta} = 0.01, \theta_k = \min\{0.01, \|x^k\|/100\}$

for all k.

The exact case is used for comparison. The constant noise forms represent cases where the inexactness is outside of the optimizer's control. The vanishing noise forms represent cases where the noise can be controlled but the mechanism is considered expensive, so it is only used when approaching the minimum. The two forms of subgradient noise represent the case where the subgradient is approximated numerically.

To address the random values of the noise the tests for the last four noise forms were executed how many??? times and the results averaged.

To compare the performance of the different methods the accuracy is measured by

$$\text{accuracy} = |\log_{10}(\hat{f}_{\bar{k}})|.$$

$\hat{f}_{\bar{k}}$ being the current $\hat{f}_k$ when the algorithm stopped.

5.5.1 Academic Test Examples

To explore the benefit of the matrix Q the algorithms 4.1 and 5.1 were tested on a smooth and a nonsmooth version of a badly conditioned parabola. The smooth test function is

$$p(x) : \mathbb{R}^2 \to \mathbb{R}, \quad x \mapsto \langle x, Ax \rangle \,,$$

where the matrix is chosen as $A = \begin{pmatrix} 1 & 0 \\ 0 & 50 \end{pmatrix}$. The condition number of this matrix is $\kappa_A = \frac{\lambda_{max}}{\lambda_{min}} = 50$, where $\lambda_{min}, \lambda_{max}$ are the smallest and largest eigenvalue of A respectively. From smooth optimization it is known that gradient descent methods have a rather poor convergence rate for such badly conditioned matrices (c.f. Chapter 7.4 in [28]). Figure 1 shows the sequences of serious iterates resulting from the two algorithms on the contour lines of the parabola. On the left the complete sequence is depicted. The plot on the right shows a detail of the left figure near the minimum of the objective. As the descent direction taken in algorithm 4.1 is an aggregate subgradient and second order information is only provided by the stabilization term $\frac{1}{t_k}\|d\|^2$ we can see a zig-zagging behavior of the sequence for the parabola in figure 1. Contrary to that the sequence of serous iterates provided by algorithm 5.1 can take advantage of the second order information provided by Q. It walks into the minimum almost in a straight line. The difference in behavior of the two algorithms is especially visible on the detail plot of 1 that shows the situation near the minimum: The proximal bundle algorithm needs a lot of steps circling around the minimum whereas the variable metric algorithm approaches the minimum directly. The resulting advantage of this behavior is the smaller number of steps needed by the variable metric algorithm.

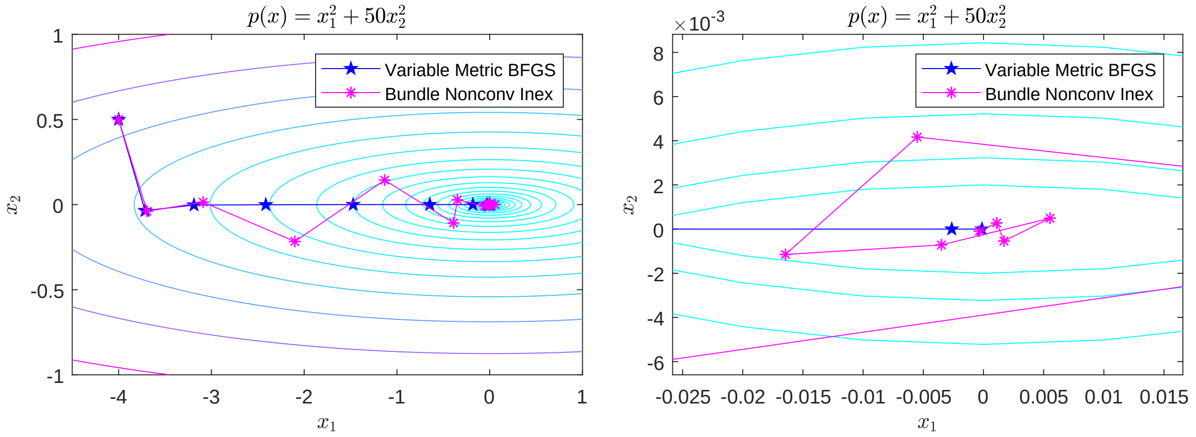

Figure 1: *Sequences of serious steps constructed by the proximal bundle algorithm and the variable metric algorithm respectively on the level lines of parabola p. The right image is a detail of the plot on the left.*

The second test function is a nonsmooth version of the above parabola. The function is given by

$$p_n(x) : \mathbb{R}^2 \to \mathbb{R}, \quad x \mapsto \left\langle \frac{1}{2}x, Ax \right\rangle + \frac{1}{2}|x_1| + 25|x_2|.$$

Due to the kink along the x_1-axis the curvature information supplied by Q is less reliable than for the smooth parabola. Figure 2 shows the sequences constructed by the two algorithms. Still the sequence provided by the variable metric algorithm does less zig-zagging than the one coming from the proximal bundle algorithm. It is interesting to note, that the sequence provided by the proximal bundle algorithm is the same for both functions. This is not the case for the sequence generated by the metric bundle algorithm because the second order information of the two objective functions is different.

The bar plots in figures 3 and 4 compare accuracy of the solution and the step sizes that are needed by the different algorithms for the various noise forms. Noise form $\mathrm{N}_{\mathrm{v}}^{\mathrm{g}}$ is the one that is most similar to no noise; one can see: gradient noise has not as much impact as inexact function value (makes sense because only calculating with subgradients) -> in those cases variable metric algorithm more exact, but all over desired tolerance in terms of exactness proximal bundle algorithm slightly better, but not much; the two variants of the variable metric algorithm are more or less the same.

for step sizes> proximal bundle algorithm needs a lot more steps than variable metric

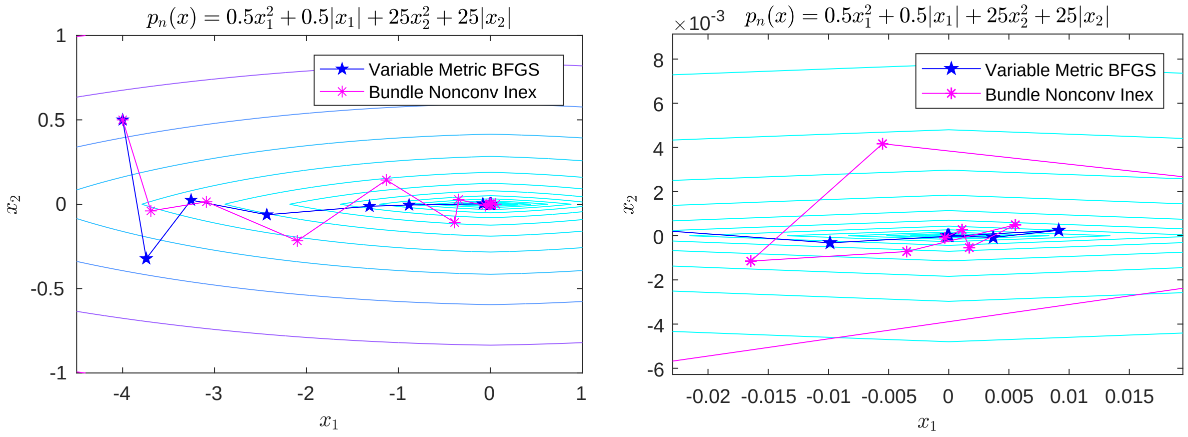

Figure 2: *Sequences of serious steps constructed by the proximal bundle algorithm and the variable metric algorithm respectively on the level lines of the nonsmooth quadratic function p_n. The right image is a detail of the plot on the left.*

especially in the case of constant function value and gradient noise

Figure 4: in nonsmooth case accuracy of proximal bundle algorithm overall slightly better; need generally more steps, numbers of steps more equal apart from constant noise part where for a minor gain of accuracy a really high numer of steps is done (plot is cropped, really: over 250 steps)
bigger difference if changing κ_+ -> add plot for that and comment???

Figure 3: *Comparison of the accuracy of the solu*

(a) *Parabola, kappa* $= 2$ **(b)**

Figure 4

second example: nonsmooth version of the parabola

$$p : \mathbb{R}^2 \to \mathbb{R}, \quad x \mapsto \left\langle x, \frac{1}{2}Ax \right\rangle + 0.5|x_1| + 25|x_2|$$

kink makes problem with second order information; different possibilities

1. change nothing, just make sure Q bounded
2. scale Q if it becomes too big
   advantage: not so much wrong information due to “krummungs-Paradoxon”

drawback: everxthing scaled -> direction wrong if kink only in one direction; have to decide when Q too big, because exiist functions with high curvature

3. change only eigenvalue that got too high; check Change relative to last Q and stepsize
4. bigger stepsizes?

first and third variants make sense in example

recht gute Ergebnisse mit $\frac{1}{k}Q$
ohne $\frac{1}{k}$ eta extrem gross; mit: eta wird auch gross: 1e10

In this section the above variant of an inexact bundle algorithm is compared with the nonconvex inexact bundle algorithm by Hare et al. in [10].

For reasons of comparability the setting was chosen as in [10].

The parameter set was chosen as follows: $m = 0.05$, $\gamma = 2$, $t_1 = 0.1$ and the scaling parameters for the choice of t_k are $\kappa_+ = 1.2$ and $\kappa_- = 0.8$ with the minimum possible t being $t_{min} = 0.03$. For the choice of the new bundle the information of the current prox-center and all elements with $\alpha > 10^{-15}$ were kept in the bundle. As stopping condition

$$\delta < \texttt{tol}(1 + f_{hat})$$

was used. If this did not apply within $\max(300, 205n)$ steps the algorithm also stopped.

As test problems the Ferrier polynomials were chosen. These nonsmooth and nonconvex functions have already been used in [9] and [10]. The polynomials are constructed in the following way:

For $i = 1, ..., n$ we define

$$h_i : \mathbb{R}^n \to \mathbb{R}, \quad h(x) = (ix_i^2 - 2x_i) + \sum_{j=1}^{n} x_j.$$

These functions are used to define

$$f_1(x) := \sum_{i=1}^{n} |h_i(x)|,$$
$$f_2(x) := \sum_{i=1}^{n} (h_i(x))^2,$$

$$\begin{aligned}f_3(x) &:= \max i \in [1, ..., n] |h_i(x)|, \\ f_4(x) &:= \sum_{i=1}^{n} |h_i(x)| + \frac{1}{2}|x|^2, \\ f_5(x) &:= \sum_{i=1}^{n} |h_i(x)| + \frac{1}{2}|x|.\end{aligned}$$

plots of graphs???

Ferrier polynomials are nonconvex, except for f_2, nonsmooth and lower-$\mathcal{C}^2$. They all have 0 as a global minimizer [9]. The compact constraint set is $X = \{d \in \mathbb{R}^n | d_i + \hat{x}_i^k \leq 10, i = 1, ..., n\}$.

The five test functions $f_1, ...f_5$ were optimized for the dimensions $n = 1, 2, ..., 15, 20, 25, 30$. The starting value for each test problem being $x^1 = [1, \frac{1}{4}, \frac{1}{9}, ..., \frac{1}{n^2}]^\top$.

Two different stopping tolerances `tol` $= 10^{-3}$ and `tol` $= 10^{-6}$ were used. Additionally the computation times for the different algorithms and tests are plotted. say on what machine that was done???

say where plots left and right...

- Hare Algo seems to be better in "global" optimization
  Noll seems to get stuck in local Minima more often
- different "Versions" of Noll: Q "ignored" near kinks because there no curvature, but seems be be "infinite"
- the t-update Parameter has A LOT of influence on the performance of the algorithm

6 Application to Model Selection for Primal SVM

Skalarprodukt anpassen, Vektoren nicht fett oder neue definition, notation, $\lambda \in \Lambda$ einfugen

6.1 Introduction

In this chapter the nonconvex inexact bundle algorithm is applied to the problem of model selection for *support vector machines* (SVM) solving classification tasks. It relies on a bilevel formulation proposed by Kunapuli [22] and Moore et al. [32].

A natural application for the inexact bundle algorithm is an optimization problem where the objective function value can only be computed iteratively. This is for example the case in bilevel optimization.

A general bilevel program can be formulated as [22]

$$
\begin{aligned}
\min_{x \in X, y} \quad & F(x,y) && \text{upper level} \\
\text{s.t.} \quad & G(x,y) \le 0 && \\
& y \in \left\{ \begin{aligned} \arg\max_{y \in Y} \quad & f(x,y) \\ \text{s.t.} \quad & g(x,y) \le 0 \end{aligned} \right\}. && \text{lower level}
\end{aligned}
\tag{6.1}
$$

It consists of an *upper* or *outer level* which is the overall function to be optimized. Contrary to usual constrained optimization problems which are constrained by explicitly given equalities and inequalities a bilevel program is additionally constrained to a second optimization problem, the *lower* or *inner level* problem.

Solving bilevel problems can be divided roughly in two classes: implicit and explicit solution methods.
In the explicit methods the lower level problem is usually rewritten by its KKT conditions and the upper and lower level are solved simultaneously. For the setting of model selection for support vector machines as it is used here, this method is described in detail in [22].

The second approach is the implicit one. Here the lower level problem is solved directly in every iteration of the outer optimization algorithm and the solution is plugged into the upper level objective.
Obviously if the inner level problem is solved numerically, the solution cannot be exact. Additionally the *solution map* $S(x) = \{y \in \mathbb{R}^k | y \text{ solves the lower level problem}\}$ is often nondifferentiable [40] and since elements of the solution map are plugged into the outer level objective function in the implicit approach, the outer level function becomes nonsmooth itself.
This is why the inexact bundle algorithm seems a natural choice to tackle these bilevel problems.
Moore et al. use the implicit approach in [32] for support vector regression. However they use a gradient decent method which is not guaranteed to stop at an optimal solution.
In [31] he also suggests the nonconvex exact bundle algorithm of Fuduli et al. [6] for solving the bilevel regression problem. This allows for nonsmooth inner problems and can theoretically solve some of the issues of the gradient descent method. It ignores however, that the objective function values can only be calculated approximately. A fact

which is not addressed in Fuduli's algorithm.

6.2 Introduction to Support Vector Machines

Support vector machines are linear learning machines that were developed in the 90's by Vapnik and co-workers. Soon they could outperform several other programs in this area [3] and the subsequent interest in SVMs lead to a very versatile application of these machines [22].

The case that is considered here is binary support vector classification using supervised learning.
In classification data from a possibly high dimensional vector space $\tilde{X} \subseteq \mathbb{R}^n$, the *feature* or *input space* is divided into two classes. These lie in the *output domain* $\tilde{Y} = \{-1, 1\}$. Elements from the feature space will mostly be called *data points* here. They get *labels* from the feature space. Labeled data points are called *examples.*
The functional relation between the features and the class of an example is given by the usually unknown *response* or *target function* $f(x)$.
Supervised learning is a kind of machine learning task where the machine is given examples of input data with associated labels, the so called *training data* (X, Y). Mathematically this can be modeled by assuming that the examples are drawn identically and independently distributed (iid) from the fixed joint distribution $P(x, y)$. This usually unknown distribution states the probability that an data point x has the label y [53].
The overall goal is then to optimize the generalization ability, meaning the ability to predict the output for unseen data correctly [3].

6.2.1 Risk minimization

The concept of SVM's was originally inspired by the statistical learning theory developed by Vapnik. For a throughout analysis see [52].
The idea of *risk minimization* is to find from a fixed set or class of functions the one that is the best approximation to the response function. This is done by minimizing a loss function that compares the given labels of the examples to the response of the learning machine.

As the response function is not known only the expected value of the loss can be calculated. It is given by the *risk functional*

$$R(\lambda) = \int \mathcal{L}(y, f_\lambda(x))\mathrm{d}P(x,y) \tag{6.2}$$

Where $\mathcal{L} : \mathbb{R}^2 \to \mathbb{R}$ is the loss function, $f_\lambda : \mathbb{R}^n \cap \mathcal{F} \to \mathbb{R},\ \lambda \in \Lambda$ the response function found by the learning machine and $P(x, y)$ the joint distribution the training data is drawn from. The goal is now to find a function $f_{\bar{\lambda}}(x)$ in the chosen function space $\mathcal{F}$ that minimizes this risk functional [53].

As the only given information is given by the training set inductive principles are used to work with the empirical risk, rather than with the risk functional. The empirical risk only depends on the finite training set and is given by

$$R_{emp}(\lambda) = \frac{1}{l}\sum_{i=1}^{l} \mathcal{L}(y_i, f_\lambda(x^i)), \tag{6.3}$$

where l is the number of data points. The law of large numbers ensures that the empirical risk converges to the risk functional as the number of data points grows to infinity. This however does not guarantee that the function $f_{\lambda,emp}$ that minimizes the empirical rist also converges towards the function $f_{\bar{\lambda}}$ that minimizes the risk functional. The theory of consistency provides necessary and sufficient conditions that solve this issue [53].

Vapnik introduced therefore the structural risk minimization induction principle (SRM). It ensures that the used set of functions has a structure that makes it strongly consistent [53]. Additionally it takes the complexity of the function that is used to approximate the target function into account. “The SRM principle actually suggests a tradeoff between the quality of the approximation and the complexity of the approximating function” [53, p. 994]. This reduces the risk of *overfitting*, meaning to overly fit the function to the training data with the result of poor generalization [3].

Support vector machines fulfill all conditions of the SRM principle. Due to the kernel trick that allows for nonlinear classification tasks it is also very powerful. For more detailed information on this see [22, 52] and references therein.

6.2.2 Support Vector machines

In the case of linear binary classification one searches for a an affine hyperplane $\boldsymbol{w}$ shifted by b to separate the given data. The vector $\boldsymbol{w}$ is called weight vector and b is the bias. Let the data be linearly separable. The function deciding how the data is classified can then be written as

$$f(x) = sign(\boldsymbol{w}^\top x - b).$$

Support vector machines aim at finding such a hyperplane that separates also unseen data optimally.

???Picture of hyperplane

One problem of this intuitive approach is that the representation of a hyperplane is not unique. If the plane described by $(\boldsymbol{w}, b)$ separates the data there exist infinitely many hyperplanes $(t\boldsymbol{w}, b)$, $t > 0$ that separate the data in the same way.
To have a unique description of a separation hyperplane the *canonical hyperplane for given data $x \in X$* is defined by

$$f(x) = \boldsymbol{w}^\top x - b \quad \text{s.t.} \quad \min_i |\boldsymbol{w}^\top x^i - b| = 1$$

This is always possible in the case where the data is linearly separable and means that the inverse of the norm of the weight vector is equal to the distance of the closest point $x \in X$ to the hyperplane [22].

This gives rise to the following definition: The *margin* is the minimal Euclidean distance between a training example x^i and the separating hyperplane. A bigger margin means a lower complexity of the function [3].

A *maximal margin hyperplane* is the hyperplane that realizes the maximal possible margin for a given data set.

Theorem 6.1 ([3, Theorem 6.1]) *Given a linearly separable training sample $\Omega = ((x^i, y_i), ..., (x^l, y_l))$ the hyperplane $(\boldsymbol{w}, b)$ that solves the optimization problem*

$$\|\boldsymbol{w}\|^2 \quad \textit{subject to} \quad y_i(\boldsymbol{w}^\top x - b) \geq 1 \quad i = 1, ..., l$$

realizes a maximal margin hyperplane

Generally one cannot assume the data to be linearly separable. This is why in most applications a so called *soft margin classifier* is used. It introduces the slack variables ξ_i that measure the distance of the misclassified points to the hyperplane:

Fix $\gamma > 0$. A *margin slack variable of the example* (x^i, y_i) with respect to the hyperplane $(\boldsymbol{w}, b)$ and target margin γ is

$$\xi_i = \max(0, \gamma - y_i(\boldsymbol{w}^\top x + b))$$

If $\xi_i > \gamma$ the point is misclassified.
One can also say that $\|\xi\|$ measures the amount by which training set “fails to have margin γ” [3].

For support vector machines the target margin is set to $\gamma = 1$.

This results finally in the following slightly different optimization problems for finding an optimal separating hyperplane $(\boldsymbol{w}, b)$:

$$\begin{aligned} \min_{\boldsymbol{w},b,\xi} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{l} \xi_i \\ \text{subject to} \quad & y_i(\boldsymbol{w}^\top x^i - b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \\ & \forall i = 1, \ldots, l \end{aligned} \tag{6.4}$$

and

$$\begin{aligned} \min_{\boldsymbol{w},b,\xi} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{l} \xi_i^2 \\ \text{subject to} \quad & y_i(\boldsymbol{w}^\top x^i - b) \geq 1 - \xi_i \\ & \forall i = 1, \ldots, l \end{aligned} \tag{6.5}$$

The first part of the respective objective functions are the regularizations, the second part are the actual loss functions. The parameter $C > 0$ gives a trade-off between the richness of the chosen set of functions f_α to reduce the error on the training data and the danger of overfitting to have good generalization. It has to be chosen a priori [22]. The two optimization problems only differ in the norm chosen for the loss function. In (6.4) the one-norm is chosen, in (6.5) the squared two-norm is used. Problem (6.5) is the one that will finally be used in the bilevel approach where smoothness of the objective function of the inner level problem is needed to calculate all needed subgradients.

6.3 Bilevel Approach and Inexact Bundle Method

The hyper-parameter C in the objective function of the classification problem has to be set beforehand. This step is part of the model selection process. To set this parameter optimally different methods can be used. A very intuitive and widely used approach is doing and *cross validation* (CV) with a grid search implementation.

To prevent overfitting and get a good parameter selection, especially in case of little data, commonly T-fold cross validation is used.
For this technique the training data is randomly partitioned into T subsets of equal size. One of these subsets is then left out for training and instead used afterwards to get an estimate of the generalization error.
To use CV for model selection it has to be embedded into an optimization algorithm over the hyper-parameter space. Commonly this is done by discretizing the parameter space and for T-fold CV training T models at each grid point. The resulting models are then compared to find the best parameters in the grid. Obviously for a growing number of hyper-parameters this is very costly. An additional drawback is that the parameters are only chosen from a finite set [22].

6.3.1 Reformulation as bilevel problem

A more recent approach is the formulation as a bilevel problem used in [22, 32]. This makes it possible to optimize the hyper-parameters continuously.

Let $\Omega = (x^1, y_1), ..., (x^l, y_l) \subseteq \mathbb{R}^{n+1}$ be a given data set of size $l = |\Omega|$. The associated index set is denoted by $\mathcal{N}$. For classification the labels y_i are ± 1. For T-fold cross validation let $\bar{\Omega}_t$ and Ω_t be the training set and the validation set within the t'th fold and $\bar{\mathcal{N}_t}$ and $\mathcal{N}_t$ the respective index sets. Furthermore let $f^t : \mathbb{R}^{n+1} \cap \mathcal{F} \to \mathbb{R}$ be the response function trained on the t'th fold and $\lambda \in \Lambda$ the hyper-parameters to be optimized. For a general machine learning problem with upper and lower loss function $\mathcal{L}_{upp}$ and $\mathcal{L}_{low}$ respectively the bilevel problem writes

$$
\begin{aligned}
\min_{\lambda, f^t} \quad & \mathcal{L}_{upp}\left(\lambda, f^1|_{\Omega_1}, ..., f^T|_{\Omega_T}\right) && \text{upper level} \\
\text{s.t.} \quad & \lambda \in \Lambda && \\
& \text{for } t = 1, ..., T : && \\
& f^t \in \left\{ \begin{array}{ll} \underset{f \in \mathcal{F}}{\arg\min} & \mathcal{L}_{low}(\lambda, f, (x^i, y_i)_{i=1}^l \in \bar{\Omega}_t) \\ \text{s.t.} & g_{low}(\lambda, f) \leq 0 \end{array} \right\}. && \text{lower level}
\end{aligned}
\tag{6.6}
$$

In the case of support vector classification the T inner problems hve the classical SVM formulation (6.5). The problem can also be rewritten into a unconstrained form. This form is helpful when using the inexact bundle algorithm for solving the bilevel problem. For the t'th fold the resulting hyperplane is identified with the pair $(\boldsymbol{w}^t, b_t) \in \mathbb{R}^{n+1}$. The inner level problem for the t'th fold can therefore be stated as

$$
(\boldsymbol{w}^t, b_t) \in \underset{\boldsymbol{w}, b}{\arg\min} \left\{ \frac{\lambda}{2} \|\boldsymbol{w}\|_2^2 + \sum_{i \in \bar{\mathcal{N}}_t} \left(\max\left(1 - y_i(\boldsymbol{w}^\top x^i - b), 0\right)\right)^2 \right\} \tag{6.7}
$$

Where the hyper-parameter $\lambda = \frac{1}{C}$ was used due to numerical stability [22].

For the upper level objective function there are different choices possible. Simply put the outer level objective should compare the different inner level solutions and pick the best one. An intuitive choice would therefore be to pick the misclassification loss, that count how many data points of the respective validation set Ω_t were misclassified when taking function f^t.

The misclassification loss can be written as

$$
\mathcal{L}_{mis} = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{|\mathcal{N}_t|} \sum_{i \in \mathcal{N}_t} \left[-y_i((\boldsymbol{w}^t)^\top x - b_t) \right]_\star \tag{6.8}
$$

where the step function $()_\star$ is defined componentwise for a vector as

$$
(r_\star)_i = \left\{ \begin{array}{ll} 1, & \text{if } r_i > 0, \\ 0, & \text{if } r_i \leq 0 \end{array} \right. . \tag{6.9}
$$

The drawback of this simple loss function is, that it is not continuous and as such not suitable for subgradient based optimization. Therefore another loss function is used for the upper level problem - the *hinge loss*. It is an upper bound on the misclassification loss and reads

$$\mathcal{L}_{hinge} = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t}\max\Big(1 - y_i((\boldsymbol{w}^t)^\top x - b_t), 0\Big). \tag{6.10}$$

It is also possible to square the max term. This results in the loss function

$$\mathcal{L}_{hinge} = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t}\max\Big(1 - y_i((\boldsymbol{w}^t)^\top x - b_t), 0\Big)^2. \tag{6.11}$$

In figure (**??**) it can be seen that its minimum and overall progress is more similar to the misclassification loss than the one of the hinge loss. For this reason we progress taking the squared form of the hinge loss, abbreviating with *hingequad loss* for convenience.

Hence the final resulting bilevel formulation for model selection in support vector classification is

$$\begin{aligned}
\min_{\boldsymbol{W},\boldsymbol{b}} \quad & \mathcal{L}_{hinge}(\boldsymbol{W},\boldsymbol{b}) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t}\max\Big(1 - y_i((\boldsymbol{w}^t)^\top x - b_t), 0\Big)^2 \\
\text{subject to} \quad & \lambda > 0 \\
& \text{for } t = 1, ..., T \\
& (\boldsymbol{w}^t, b_t) \in \operatorname*{arg\,min}_{\boldsymbol{w},b}\left\{\frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \sum_{i\in\bar{\mathcal{N}}_t}\Big(\max\Big(1 - y_i(\boldsymbol{w}^\top x^i - b), 0\Big)\Big)^2\right\}.
\end{aligned} \tag{6.12}$$

6.3.2 Solution of the Bilevel Program

ab hier: Theorie fehlt

!!! notation - oder in prelininaries einfugen

To solve the given bilevel problem with the above presented nonconvex inexact bundle algorithm the algorithm jumps between the two levels. Once the inner level problems are solved for a given λ - this is possible with any QP-solver as the problems are convex - the bundle algorithm takes the outcome w and b and optimizes the hyper-parameter again.

The difficulty with this approach is that the bundle algorithm needs one subgradient of the outer level objective function with respect to the parameter λ. However to compute this subgradient also one subgradient of w and b with respect to λ has to be known.

The Differentiable Case example in differentiable case

Let us first assume that the outer and inner objective functions and $w(\lambda) = \arg\min \mathcal{L}_{low}(w, \lambda)$

are sufficiently often continuously differentiable to demonstrate the procedure of calculating the needed (sub-)gradients.

Let $\mathcal{L}_{upp}(w, \lambda)$ be the objective function of the outer level problem, where the variable b was left out for the sake of simplicity. To find an optimal hyper parameter λ given the input w the gradient g_λ^{upp} of $\mathcal{L}_{upp}$ with respect to λ is needed in every iteration of the solving algorithm. In order to calculate this gradient the chain rule is used yielding

$$g_\lambda^{upp} = \left(\frac{\partial}{\partial w}\mathcal{L}_{upp}(w, \lambda)\right)^\top \frac{\partial w(\lambda)}{\partial \lambda} + \frac{\partial}{\partial \lambda}\mathcal{L}_{upp}(w, \lambda).$$

The challenge is here to find the term $\frac{\partial w(\lambda)}{\partial \lambda}$ because

$$\frac{\partial w}{\partial \lambda} \in \frac{\partial}{\partial \lambda} \arg\min_w \mathcal{L}_{low}(w, \lambda).$$

Assuming $\mathcal{L}_{low}$ is twice continuously differentiable at the optimal solution w^* of the lower level problem the optimality condition for any parameter $\lambda_0 > 0$

$$0 = \frac{\partial}{\partial w}\mathcal{L}_{low}(w^*, \lambda_0) \tag{6.13}$$

can be used to calculate the needed gradient in an indirect manner.

In the differentiable case the theoretical framework for the following calculations is given by the implicit function theorem.

Theorem 6.2 (c.f. [21, chapter 3.4]) *Let $F : U \times V \to Z$, $U \in \mathbb{R}^m, V, Z \in \mathbb{R}^n$, be a $\mathcal{C}^1$ mapping, $(x_0, y_0) \in U \times V$ and $F(x_0, y_0) = 0$. If the matrix $\frac{\partial}{\partial y}F(x_0, y_0)$ is invertible, there exist neighborhoods $U_0 \subset U$ of x_0 and $V_0 \subset V$ of y_0 and a continuously differentiable mapping $f : U_0 \to V_0$ with*

$$F(x, y) = 0, (x, y) \in U_0 \times V_0 \quad \Leftrightarrow \quad y = f(x), x \in U_0.$$

Identifying $x \mathrel{\hat{=}} \lambda$, $y \mathrel{\hat{=}} w$ and $F(x_0, y_0) \mathrel{\hat{=}} \frac{\partial}{\partial w}\mathcal{L}_{low}(w^*, \lambda_0)$ and assuming $\frac{\partial^2}{\partial w^2}\mathcal{L}_{low}(w^*, \lambda_0)$ is invertible this theorem provides the existence of the continuously differentiable function $w(\lambda)$ whose gradient is needed.
what about the neighborhoods?

If the inner level loss function yields a linear optimality condition in w it is possible to calculate the gradient explicitly. This is for example the case for SVM loss functions with a squared one- or two-norm as given in problem (6.5). The optimality condition can then be written as the linear system

$$H(\lambda)w = h(\lambda).$$

By taking the partial derivative with respect to λ on both sides of the system one gets

$$\frac{\partial H(\lambda)}{\partial \lambda} w + H(\lambda)\frac{\partial w}{\partial \lambda} = \frac{\partial h(\lambda)}{\partial \lambda}.$$

If $H(\lambda)$ is invertible for all $\lambda \in \Lambda$ then the needed gradient is given by

$$\frac{\partial w}{\partial \lambda} = H^{-1}(\lambda)\left(\frac{\partial h(\lambda)}{\partial \lambda} - \frac{\partial H(\lambda)}{\partial \lambda} w\right).$$

The Nondifferentiable Case now for subgradients

In practice we cannot expect $\mathcal{L}_{low}$ to satisfy such strong differentiability properties. It is therefore only assumed that $\mathcal{L}_{low}$ is once continuously differentiable in w. This assures that the optimality condition of the lower level problem is an equality like in (6.13). Contrary to the exemplary calculations from above in practice the second derivative $\frac{\partial^2}{\partial w \partial \lambda}\mathcal{L}_{low}(w(\lambda), \lambda)$ however is not existent in this form, but a set of subgradients.

Notation

First the theoretical framework given to derive the results from above in the nondifferentiable case

An important result about Lipschitz functions is Rademacher's theorem which states that these functions are differentiable almost everywhere but on a set of Lebesgue measure zero[11, Theorem 3.1]. Clarke deduces from this that the subdifferential at each of the nondifferentiable points is the convex hull of the limits of the sequence gradients a these points [2, see Theorem 2.5.1].

This motivates the multidimensional definition of Clake's generalized gradient

Definition 6.3 ([2, Definition 2.6.1]) *generalized Jacobian*: $F : \mathbb{R}^n \to \mathbb{R}^m$, with locally Lipschitz component functions $F(x) = (f_1(x), ..., f_m(x))$.
Denote generalized Jacobian by $\partial F(x) = conv\left(\lim JF(x_i) | x_i \to x, x_i \notin \Omega_F\right)$ where Ω_F is the set of nondifferentiable points of F

(after that comes proposition with properties of ∂F)

To facilitate readability we use the following notation for the derivation of the nondifferentiable results.

The *'partial' subdifferential* of a function $f(a^*, b_0, c_0, ...)$ at the point a^* with respect to one variable a when all other variables are fixed is denoted by

$$\partial^a f(a^*, b_0, c_0, ...).$$

A subgradient of this subdifferential is written $g^a \in \partial^a f(a^*, b_0, c_0, ...)$.

Next step: show that chain rule is still valid in the nonsmooth case Chain rules for subdifferential

Theorem 6.4 ([2, Theorem 2.6.6]) *Let $f(x) = \phi(F(x))$, with the locally Lipschitz functions $F : \mathbb{R}^n \to \mathbb{R}^m$ and $\phi : \mathbb{R}^m \to \mathbb{R}$. Then f is locally Lipschitz and it holds*

$$\partial f \subset conv\{\partial\phi(F(x))\partial F(x)\}.$$

If in addition ϕ is strictly differentiable at $F(x)$, then equality holds.

strictly differentiable: c.f. [44, Theorem 9.17 and 9.18] locally Lipschitz continuous and at most one subgradient at the point in question (see also comment to Definition 91 in [44])

Theorem 6.5 (c.f. [45, Theorem 7.1]) *Let $p(x) = f(F(x))$, where $F : \mathbb{R}^n \to \mathbb{R}^d$ is locally Lipschitz and $f : \mathbb{R}^d \to \mathbb{R}$ is lower semicontinuous. Assume*

$$\nexists y \in \partial^\infty f(F(\bar{x})), y \neq 0 \quad with \quad 0 \in y\partial F(\bar{x}).$$

Then for the sets

$$M(\bar{x}) := \partial f(F(\bar{x}))\partial F(\bar{x}), \quad M^\infty(\bar{x}) := \partial^\infty f(F(\bar{x}))\partial F(\bar{x}),$$

one has $\hat{\partial} p(\bar{x}) \subset M(\bar{x})$ and $\hat{\partial}^\infty p(\bar{x}) \subset M^\infty(\bar{x})$.

show that functions fulfill assumptions

Implicit function theorem>

Theorem 6.6 ([40, Theorem 5.2]) *(a) Assume there exists a single valued Lipschitz function F*

in original theorem around 0< can always be done by

- notation - check
- definition of subgradient-"matrix" - check
- chain rule - check
- optimality condition /check
- welche art von inexaktheit -> Funktionswerte w, b inexakt
  -> gradient im Endeffekt exakt, da von exakter optimalit'tsbedingung ausgegangen wird

In practice this (Rademacher) means that it is possible to choose a subgradient by using the (one sided) gradients at nondifferentiable points. We keep this in mind when analyzing the procedure of finding a subgradient $g^\lambda \in \partial^\lambda w(\lambda)$ in the nondifferentiable case.

what else???

explanation

For me: f locally Lipschitz??? then partial derivatives are the same! Else: check definition of derivatives!

-> theory partial derivatives for subgradients?????????
??? Formula ??? $\in \partial L_{upp} \partial \lambda$
???one has to assume that the inner level problem is locally Lipschitz (or more general: its nonconvex subdifferential is well defined at every point).
Subdifferential has to have again a subdifferential!!! -> w.r.t. λ

The main idea is to replace the inner level problem by its optimality condition

$\partial(w, b)$ means in this case that the subdifferential is taken with respect to the variables w and b.
-> theory for subdifferentials in more than one variable!!!

For convex inner level problem this replacement is equivalent to the original problem.

The difference to the approach described in [22] is that the problem is not smoothly replaced by its KKT conditions but only by this optimality condition. The weight vector $\boldsymbol{w}$ and bias b are treated as a function of λ and are optimized separately from this hyperparameter. The reformulated bilevel problem becomes:

$$\begin{aligned} \min_{\boldsymbol{W},\boldsymbol{b}} \quad & \mathcal{L}_{hinge}(\boldsymbol{W},\boldsymbol{b}) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t}\max\Big(1-y_i((\boldsymbol{w}^t)^\top x - b_t),0\Big) \\ \text{subject to} \quad & \lambda > 0 \\ & \text{for } t=1,...,T \\ & 0 \in \partial(w,b)\mathcal{L}_{low}(\lambda,w^t,b_t) \end{aligned} \tag{6.14}$$

where $\mathcal{L}_{low}$ can be the objective function of either of the two presented lower level problems.

solve the inner level problem (quadratic problem in constrained case) by some QP solver put solution into upper level problem and solve it by using bundle method difficulty: subgradient is needed to build model of the objective function –> need subgradient $\frac{\partial\mathcal{L}}{\partial\lambda}$ –> for this need $\frac{\partial(W,b)}{\partial\lambda}$
but (w,b) not available as functions -> only values

Moore et al. [32] describe a method for getting the subgradient from the KKt-conditions of the lower level problem:

lower level problem convex -> therefore optimality conditions (some nonsmooth version -> source???) necessary and sufficient -> make "subgradient" of optimality conditions and then derive subgradient of w, b from this.
—> what are the conditions? optimality condition Lipschitz?

Say (show) that all needed components are locally Lipschitz; state theorems about differentiability almost everywhere and convex hull of gradients gives set of subgradients introduce special notation (only for this chapter) and because of readability adopt "gradient writing"

Subgradients: $\mathcal{G}_{upp,\lambda}, \mathcal{G}_{upp,w}, \mathcal{G}_{upp,b}$ -> subgradients of outer objective
g_w, g_b -> subgradient of w, b

$$finalsubgradient = (\mathcal{G}_{upp,w}(w,b,\lambda))^\top g_w + (\mathcal{G}_{upp,b}(w,b,\lambda))^\top g_b + \mathcal{G}_{upp,\lambda}(w,b,\lambda)$$

subgradients $\mathcal{G}_{upp,...}$ easy to find (assumption that locally Lipschitz) -> in this application differentiable

difficulty: find g_w, g_b important: optimality condition must be a linear system in w, b -> this is the case in this application

$$H(\lambda) \cdot (w, b)^\top = h(\lambda)$$

find subgradients of each element (from differentiation rules follows)

$$\partial_\lambda H \cdot (w, b)^\top + H \cdot (\partial_\lambda w, \partial_\lambda b)^\top = \partial_\lambda h$$

solve this for (w, b):

$$(\partial_\lambda w, \partial_\lambda b)^\top = H^{-1} \left(\partial_\lambda h - \partial_\lambda H \cdot (w, b)^\top \right)$$

matrix H has to be inverted -> in the feature space so scalable with size of data set -> still can be very costly [32]

Applied to the two bilevel classification problems derived above, the subgradients have the following form:

derivative of upper level objective: Notation: $\delta_i := 1 - y_i(w^\top x^i - b)$

$$\partial_w \mathcal{L}_{upp} = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{\mathcal{N}_t} \sum_{i \in \mathcal{N}_t} \begin{cases} -y_i x^i & \text{if } \delta_i > 0 \\ 0 & \text{if } \delta_i \leq 0 \end{cases} \tag{6.15}$$

$$\partial_b \mathcal{L}_{upp} = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{\mathcal{N}_t} \sum_{i \in \mathcal{N}_t} \begin{cases} y_i & \text{if } \delta_i > 0 \\ 0 & \text{if } \delta_i \leq 0 \end{cases} \tag{6.16}$$

here at the kink subgradient 0 is taken

for hingequad: -> here subgradient
optimality condition:

$$0 = \partial_{\boldsymbol{w}} \mathcal{L}_{low} = \lambda \boldsymbol{w} + 2 \sum_{i \in \bar{\mathcal{N}}_t} \begin{cases} (1 - y_i(w^\top x^i - b))(-y_i x^i) & \text{if } \delta_i > 0 \\ 0 & \text{if } \delta_i \leq 0 \end{cases} \tag{6.17}$$

$$0 = \partial_b \mathcal{L}_{low} = 2 \sum_{i \in \bar{\mathcal{N}}_t} \begin{cases} (1 - y_i(w^\top x^i - b))(y_i) & \text{if } \delta_i > 0 \\ 0 & \text{if } \delta_i \leq 0 \end{cases} \tag{6.18}$$

subgradient??? is this smooth? with respect to λ

$$0 = \boldsymbol{w} + \lambda\partial_\lambda \boldsymbol{w} + 2\sum_{i\in\bar{\mathcal{N}_t}} \begin{cases} (-y_i(\partial_\lambda w^\top x^i - \partial_\lambda b))(-y_i x^i) & \text{if } \delta_i > 0 \\ 0 & \text{if } \delta_i \leq 0 \end{cases} \tag{6.19}$$

$$0 = 2\sum_{i\in\bar{\mathcal{N}_t}} \begin{cases} (-y_i(\partial_\lambda w^\top x^i - \partial_\lambda b))(y_i) & \text{if } \delta_i > 0 \\ 0 & \text{if } \delta_i \leq 0 \end{cases} \tag{6.20}$$

From this the needed subgradients can be calculated via:

$$2 \cdot \begin{pmatrix} \sum_{i\in\bar{\mathcal{N}_t}} \frac{\lambda}{2} + y_i^2 x^i (x^i)^\top & \sum_{i\in\bar{\mathcal{N}_t}} -y_i^2 x^i \\ \sum_{i\in\bar{\mathcal{N}_t}} -y_i^2 (x^i)^\top & \sum_{i\in\bar{\mathcal{N}_t}} y_i^2 \end{pmatrix} \cdot \begin{pmatrix} \partial_\lambda w \\ \partial_\lambda b \end{pmatrix} = \begin{pmatrix} -w \\ 0 \end{pmatrix} \tag{6.21}$$

for hinge not quad:

not as much information in the subgradient/derivative

similar calculation leads to

$$\partial_\lambda w = -\frac{w}{\lambda} \tag{6.22}$$

$$\partial_\lambda b = 0 \tag{6.23}$$

6.3.3 The Algorithm???

The inexact bundle algorithm for the support vector classification task in bilevel formulation

Bilevel Bundle Method

Initiate all parameters, select a starting hyper-parameter λ_1 and solve the lower level problem for $\boldsymbol{w}^1$ and b_1.
Calculate arbitrary subgradients of $\boldsymbol{w}^1$ and b_1 with respect to λ via 6.21 and a subgradient of the upper level problem by 6.3.2. For $k = 1, 2, 3, \ldots$

1. Calculate the step d^k by minimizing the model of the convexfied objective
2. Compute the aggregate subgradient and error and the stopping tolerance δ. If $\delta_k \leq \texttt{tol} \rightarrow$ STOP.
3. Set $\lambda^{k+1} = \hat{\lambda}^k + d^k$.

4. solve again the inner level problem and calculate all subgradients needed to compute a subgradient of the outer level objective
   Calculate function value and a subgradient for the outer level objective function and test if a serious step was done If yes, set $\hat{\lambda}^{k+1} = \lambda^{k+10}$ and select $t_{k+1} > 0$.
   Otherwise → nullstep
   Set $\hat{\lambda}^{k+1} = \hat{\lambda}^k$ and choose $0 < t_{k+1} \leq t_k$.
5. Select new bundle index set J_{k+1}. Calculate convexification parameter η_k and update the model M^k

Names for algorithms: BBMH -> hinge as inner level, BBMH2 -> hingequad as inner level

6.4 Numerical Experiments

The bilevel-bundle algorithm for classification was tested for four different data sets taken from the UCI Machine Learning Repository *citations as said in "names" data???* . For comparability with the already existing results presented in [22] the following data and specifications of it were taken:

Table like in Kunapuli

Data set	l_{train}	l_{test}	n	T
Pima Indians Diabetes Database	240	528	8	3
Wisconsin Breast Cancer Database	240	443	9	3
Cleveland Heart Disease Database	216	81	13	3
John Hopkins University Ionosphere Database	240	111	33	3

Table 1

As described in the PhD thesis the data was first standardized to unit mean and zero variance (*not the 0,1 column in ? dataset*). The bilevel problem with cross validation was executed 20 times to get averaged results. The results are compared by cross validation error, test error -> write which error this is and computation time. Additionally write $\boldsymbol{w}$, b, λ ??? The objective function and test error were scaled by 100. -> also test error (to get percentage)

After every run the calculated λ was taken and the algorithm was trained with $\frac{T}{T-1}\lambda$ on the whole training set. Then the percentage of misclassifications on the test set was calculated via

$$E_{test} = \frac{1}{l_{test}} \sum_{i=1}^{l_{test}} \frac{1}{2} |sign \left(\boldsymbol{w}^\top x^i - b \right) - y_i| \tag{6.24}$$

Table ??? shows the results

Data set	Method	$T/(T-1)\lambda$	CV Error	Test Error	Time (sec.)
pima	hingequad	10^{-15}	60.72 ± 9.56	24.11 ± 2.71	2.15 ± 0.52
	hinge loss				
cancer	hingequad	$0.6 < \lambda < 10.3$	10.75 ± 7.52	3.41 ± 1.16	3.43 ± 28.84
	hinge loss				
heart	hingequad	10^{-16}	48.73 ± 5.53	15.56 ± 4.44	3.43 ± 43.39
	hinge loss				
ionosphere	hingequad	$3 < \lambda < 7.5$	39.30 ± 5.32	12.21 ± 4.10	14.17 ± 51.27
	hinge loss				

Table 2

λ values in table not right, don‘t know with which algorithms they were reached

23.06.2017
found out: for real results, have to do it with the functions I take in the algorithm -> this is hingequad for inner level and either hingeloss or hingequad for outer level
from plots it seems that hingequad is closer to the misclassification loss
generally: from plots it looks as if all λs are best =0
with bundle bilevel and hingeloss (outer): λ very much depending on starting value -> why??? graphs seem to be monotonously decreasing into 0

plots such as objective function (upper hingeloss and lower hingequad) in bilevel bundle algorithm: No minimum visible (also for ionosphere and cancer???...)

analysis of every plot:
pima: looks the same as “old” plot, minimum is 0
wine quality red and red 56: minimum is 0
covtype: same
cancer: doesn’t really look similar to “old” plot, slope different, minimum different; minimum at 0 and not at about 10
ionosphere: slope, ect. look similar to “old” plot; but minimum at 0
heart: like “old” plot
maybe cancer and ionosphere plots just “incidents” -> because of special choice of vali-

dation set????
-> no, can also be that I averaged over 20 times...

misclassification loss as upper level objective
pima: average seems to fo to 0 as min
generally: misclassification loss seems as if no optimization of λ possible because choice of validation set seems to have much more influence

Results forλ only if it stayed there after second run with second starting value
change in λ has very little effect; only after comma for “percent-writing”
errors like in table for all but ionosphere -> has only 5% error?; ???-> error the smaller, the smaller λ???
pima simply not depending on λ
cancer not really depending in λ, only if it gets really big > 1000 (for > 10 minor change)
heart: changes a lot for the different λ, but best: $\lambda = 0$
seems that results come because of scaling of objective -> consistent with the plots I made
also consider: loss function of optimizer is not the one that calculates the test error

Extra table for $\boldsymbol{w}$, b, λ ?

First experiment: Classification

Write down bilevel classification problem and (if needed) which specification of the inexact bundle algorithm is used.

Covtype tests
Datensatz zerteilt: 1000, 5000, 10000, 50000 Datensätze
Ergebnisse mit Matlab App (linear SVM, 3 folds, parallel used):

Datensatz	Zeit	Fehler
1,000	16.22 sek	34.2%
5,000	10.524 sek	18.3%
10,000	14.689 sek	16.5%
50,000	643.57 sek	16.9%
50,000 (Rechnerhalle, 4 parallel)	326.83	16.9%
100,000 –”–	2492 sek = 41.5333 min	21.3%

scheint bei 50000 tatsächlich so lange zu dauern, da ziemlich genau doppelt so schnell bei parallel-Rechnung mit 4 anstatt 2 “Rechnern” –> kein Arbeitsspeicher Problem

Test mit bundle bilevel-Algorithmus:
for covtype, “Hare”-stopping condition

trainigs set	starting value	lambda	time	k	inull	Fehler
1,000	1		62.8280 sek	29	0	
1,000	86		130.1105 sek	61	8	19%
1,000 (quadprog)	86		6.5572 sek	61	8	
5,000 qp	1		16.2338 sek	27	0	
5,000 qp	47		34.9573 sek	60	0	14.74%
10,000 qp	1		59.7462 sek	49	0	
10,000 qp	50		85.0816 sek	69	0	15.32%
50,000 qp		588.2828 sek	45	0		
50,000 qp	60		897.8123 sek	69	0	14.86%
100,000 qp	1		1358.7 sek = 22.6455 min	37	0	39.41

!!!!!!!!!!!did not take λ but the error value!!!!!!!!!!!!!!!

Fehler berechnet für Daten 1001 bis 2000 von komplettem Datensatz

Versuch mit Daten 1001-5000: scheint Problem bei matrizenaufbau zu haben

Analyse Timer: 100% der Zeit in postpro-wb-class-hinge-qpas dabei 100% der Zeit fÃ$\frac{1}{4}$r qpas

Ergebnis viel!!! schneller, wenn quadprog und sparse-matrizen, gibt EXAKT! gleiches Ergebnis fÃ$\frac{1}{4}$r fehler

geht dann auch mit mehr datensätzen (4000) -> Fehler nur 7.75%???

für Datensatz 5000 → Testdaten 5001 bis 10000

exemplarisch getestet: wie gross ist der Unterschied bei ergebnisse bei verschiedenen Starwerten? - bei 10000: $1e-15$, bei 50000: exact

für Datensatz 10000: Testdaten 10001 bis 15000

50000: deutlicher Anstieg der Rechenzeit merkbar irgendwo zwischen 10000 und 50000 muss eine Schwelle liegen - Speicher? - ab dieser Matrix Grösse gibt matlab fehler wenn nicht sparse matrizen explizit erstellt werden sollen (zu viel Speicher) unwahrscheinlich - siehe erklärung App

komish: warum dauert es bei näherem Startwert so viel länger???

unconstrained tested for 1000: much faster; no difference in steps for $x_0 = 1$, 70, 6 for $x_0 = 86$, 7.2 sek

0815-Funktion für bilevel optimierung:

fÃ$\frac{1}{4}$r covtype1000: in sehr wenig Zeit: lambda = 100 -> selbes ergebnis im Fehler: 19%

Infos on Data sets

Data Set	instances, attributes	1/C (SVC)	C, ε (SVR)	Test Error	Source	
Adult	300 000, 10+1		0.017, 0.19	24.99%	[33]	think lin
Boston Housing	506, 12+1		0.25, 0.015	19.44%	[33]	
Adult	Tset: 11221	1/0.05			[41]	
Adult		$> 10^8$		14.1	[16]	accuracy ar

If more values found: take best

Rechtschreibfehler, Namen, Stil überprüfen

References

[1] P. Apkarian, D. Noll, and O. Prot. A trust region spectral bundle method for nonconvex eigenvalue optimization. *SIAM Journal on Optimization*, 19(1):281–306, jan 2008.

[2] Frank H. Clarke. *Optimization and nonsmooth analysis*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics Philadelphia, 1990.

[3] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[4] Welington de Oliveira and Claudia Sagastizábal. Bundle methods in the xxist century: A bird's-eye view. *Pesquisa Operacional*, 34(3):647–670, dec 2014.

[5] A. Fuduli, M. Gaudioso, and G. Giallombardo. A dc piecewise affine model and a bundling technique in nonconvex nonsmooth minimization. *Optimization Methods and Software*, 19(1):89–102, 2004.

[6] A. Fuduli, M. Gaudioso, and G. Giallombardo. Minimizing nonconvex nonsmooth functions via cutting planes and proximity control. *SIAM Journal on Optimization*, 14(3):743–756, 2004.

[7] Carl Geiger and Christian Kanzow. *Theorie und Numerik restringierter Optimierungsaufgaben*. Sp, 2002.

[8] Napsu Haarala, Kaisa Miettinen, and Marko M. Mäkelä. Globally convergent limited memory bundle method for large-scale nonsmooth optimization. *Mathematical Programming*, 109(1):181–205, 2007.

[9] Warren Hare and Claudia Sagastizàbal. A redistributed proximal bundle method for nonconvex optimization. *SIAM Journal on Optimization*, 20(5):2442–2473, 2010.

[10] Warren Hare, Claudia Sagastizàbal, and Mikhail Solodov. A proximal bundle method for nonsmooth nonconvex functions with inexact information. *Computational Optimization and Applications*, 63:1–28, 2016.

[11] Juha Heinonen. Lectures on lipschitz analysis. Lectures at the 14th Jyväskylä Summer School in August 2004, 2004.

[12] Michael HintermÃ$\frac{1}{4}$ller. A proximal bundle method based on approximate subgradients. *Computational Optimization and Applications*, 20:245–266, 2001.

[13] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Convex Analysis and Minimization Algorithms II*, volume 306 of *Grundlehren der mathematischen Wissenschaften*. Springer Berlin Heidelberg, 1993.

[14] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Convex Analysis and Minimization Algorithms I*, volume 305 of *Grundlehren der mathematischen Wissenschaften*. Springer Berlin Heidelberg, 2 edition, 1996.

[15] Alejandro Jofré, Dinh The Luc, and Michel Théra. ε-subdifferential and ε-monotonicity. *Nonlinear Analysis: Theory, Methods & Applications*, 33(1):71–90, jul 1998.

[16] W. C. Kao, K. M. Chung, C. L. Sun, and C. J. Lin. Decomposition methods for linear support vector machines. *Neural Computation*, 16(8):1689–1704, Aug 2004.

[17] K. C. Kiwiel. Bundle methods for convex minimization with partially inexact oracles. Technical report, Systems Research Institute, Polish Academy of Sciences, 2010.

[18] Krzysztof C. Kiwiel. *Methods of Descent for Nondifferentiable Optimization*. Springer, 1985.

[19] Krzysztof C. Kiwiel. An aggregate subgradient method for nonsmooth and nonconvex minimization. *Journal of Computational and Applied Mathematics*, 14(3):391–400, 1986.

[20] Krzysztof C. Kiwiel. A proximal bundle method with approximate subgradient linearizations. *SIAM Journal on Optimization*, 16(4):1007–1023, jan 2006.

[21] Konrad KÃ¶nigsberger. *Analysis 2*. Springer Berlin Heidelberg, 2002.

[22] Gautam Kunapuli. *A bilevel optimization approach to machine learning*. PhD thesis, Rensselaer Polytechnic Institute Troy, New York, 2008.

[23] Claude Lemaréchal and Claudia Sagastizábal. *An approach to variable metric bundle methods*, pages 144–162. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.

[24] Claude Lemaréchal and Claudia Sagastizábal. Variable metric bundle methods: From conceptual to implementable forms. *Mathematical Programming*, 76(3):393–410, 1997.

[25] A. S. Lewis and S. J. Wright. A proximal method for composite minimization. *Mathematical Programming*, 158(1-2):501–546, aug 2015.

[26] Adrian S Lewis and Michael L Overton. Nonsmooth optimization via bfgs. *submitted to SIAM Journal on Optimization*, pages 1–35, 2009.

[27] L. Lukšan and J. Vlček. Globally convergent variable metric method for convex nonsmooth unconstrained minimization. *Journal of Optimization Theory and Applications*, 102(3):593–613, sep 1999.

[28] Stefan Ulbrich Michael Ulbrich. *Nichtlineare Optimierung*. Springer Basel AG, 2012.

[29] Robert Mifflin. A modification and an extension of lemaréchal's algorithm for nonsmooth minimization. In *Mathematical Programming Studies*, volume 17, pages 77–90. Springer Nature, 1982.

[30] Robert Mifflin and Claudia Sagastizàbal. A science fiction story in nonsmooth optimization originating at iiasa. *Documenta Mathematica*, Extra Volume ISMP:291–300, 2012.

[31] G. Moore, C. Bergeron, and K. P. Bennett. Gradient-type methods for primal svm model selection. *Neural Information Processing Systems Workshop: Optimization for Machine Learning*, 2010.

[32] Gregory Moore, Charles Bergeron, and Kristin P. Bennett. Model selection for primal svm. *Machine Learning*, 85(1):175–208, 2011.

[33] D. R. Musicant and A. Feinberg. Active set support vector regression. *IEEE Transactions on Neural Networks*, 15(2):268–275, March 2004.

[34] David R. Musicant. *Data Mining via Mathematical Programming and Machine Learning.* PhD thesis, University of Wisconsin, Madison, 2000.

[35] Jorge Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, 1980.

[36] Dominikus Noll. Cutting plane oracles to minimize non-smooth non-convex functions. *Set-Valued and Variational Analysis*, 18(3-4):531–568, sep 2010.

[37] Dominikus Noll. Bundle method for non-convex minimization with inexact subgradients and function values. In *Computational and Analytical Mathematics*, pages 555–592. Springer Nature, 2013.

[38] Dominikus Noll and Pierre Apkarian. Spectral bundle method for non-convex maximum eigenvalue functions: first-order methods. *Mathematical Programming*, 104(2-3):701–727, jul 2005.

[39] Dominikus Noll, Olivier Prot, and Aude Rondepierre. A proximity control algorithm to minimize non-smooth and non-convex functions. *Pacific Journal of Optimization*, 4(3):571–604, 2012.

[40] Jiři Outrata, Michal Kočvara, and Jochem Zowe. *Nonsmooth Approach to Optimization Problems with Equilibrium Constraints.* Springer US, 1998.

[41] John C. Platt. Using analytic qp and sparseness to speed training of support vector machines. In M. J. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 557–563. MIT Press, 1999.

[42] Boris T. Polyak. *Introduction to Optimization.* Optimization Software , Inc., Publications Division, New York, 1987.

[43] R. T. Rockafellar. *Convex Analysis.* Princeton University Press, Princeton, New Jersey, 1970.

[44] R. Tyrrell Rockafellar and Roger J. B. Wets. *Variational Analysis*, volume 317 of *Grundlehren der mathematischen Wissenschaften.* Springer Berlin Heidelberg, 3rd edition, 2009.

[45] R.T. Rockafellar. Extensions of subgradient calculus with applications to optimization. *Nonlinear Analysis: Theory, Methods & Applications*, 9(7):665–698, jul 1985.

[46] R.T. Rockafellar. *Convex Analysis.* Princeton University Press, 1996.

[47] Helga Schramm and Jochem Zowe. A version of the bundle idea for minimizing a nonsmooth function: Conceptual idea, convergence analysis, numerical results. *SIAM Journal on Optimization*, 2(1):121–152, feb 1992.

[48] Helga Schramm and Jochem Zowe. A version of the bundle idea for minimizing a nonsmooth function: conceptual idea, convergence analysis, numerical results. *SIAM*

Journal on Optimization, 2(1):121–152, feb 1992.

[49] M. V. Solodov. Aon approximations with finite rprecision in bundle methods for nonsmooth optimization. *Journal of Optimization Theory and Applications*, 119(1):151–165, 2003.

[50] Mikhail V. Solodov. *Constraint Qualifications.* Wiley Encyclopedia of Operations Research and Management Science, 2011.

[51] Jay S. Treiman. Clarke's gradients and ε-subgradients in banach spaces. *Transactions of the American Mathematical Society*, 294(1):65–65, jan 1986.

[52] Vladimir N. Vapnik. *Statistical Learning Theory.* JOHN WILEY & SONS INC, 1998.

[53] Vladimir N. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5), 1999.

[54] J. Vlček and L. Lukšan. Globally convergent variable metric bundle method for nonconvex nondifferentiable unconstrained minimization. *Journal of Optimization Theory and Applications*, 111(2):407–430, 2001.

[55] Claudia Sagastizàbal Warren Hare. Computing proximal points of nonconvex functions. *Mathematical Programming*, 116:221–258, 2009.

[56] Claude Lemaréchal Welington de Oliveira, Claudia Sagastizàbal. Convex proximal bundle methods in depth: a unified analysis for inexact oracles. *Mathematical Programming*, 148:241–277, 2014.

[57] Jochen Werner. *Numerische Mathematik 2.* Vieweg, Wiesbaden, 1992.