Contents

1 Introduction

There exists a sound and board theory of classical nonlinear optimization. However, this theory puts strong differentiability requirements on the given problem. Requirements that cannot always be fulfilled in practice. Examples for such practical application reach from problems in physics and mechanical engineering [3] over optimal control problems up to data analysis [2] and machine learning [28]. Other possible fields of applications are risk management and financial calculations [21, 29]. Additionally there exist so called stiff problems that are theoretically smooth but numerically nonsmooth due to rapid changes in the gradient [15].
There exists therefore a need for nonsmooth, that is not necessarily differentiable, optimization algorithms. A lot of the underlying theory and was developed in the 1970's, also driven by the "First World Conference on Nonsmooth Optimization" taking place in 1977 [18]. Now, there exists a well understood theoretical framework of nonsmooth analysis to create the basis for practical algorithms.
The most popular methods to tackle nonsmooth problems at the moment are bundle methods [32]. First developed only for convex functions [13] the method was soon extended to cope also with nonconvex objective functions [17].
Some time later these algorithms were again enhanced to deal with inexact information of the function value, the subgradient or both.
Some natural applications for these cases are derivative free optimization and stochastic simulations [32]. Some more examples from different sources? Bilevel Problems?

The basic idea of bundle methods is to model the original problem by a simpler function, often some sort of stabilized cutting plane model, that is minimized as a subproblem of the algorithm [9].

Adapt this part to what I finally really do:
In this thesis two different types of model functions will be examined that allow the use of inexact information in small to medium-scale problems as well as in large-scale problems. A limited memory approach is examined for the latter case.
what new? Combination of large-scale and inexact information - why needed
don't forget What - why - how

Adapt this part to what I finally really do:
This thesis is organized as follows:
introduction of the most important definitions and results of nonsmooth analysis. Then

the introduction of a very basic bundle algorithm which is then generalized for nonconvex functions with nonsmooth optimization.
Throughout study of this algorithm including comparison to other approaches to tackle inexact information.
Introduction of variable metric (bundle) algorithm to tackle large-scale applications. "discussion" how far this is compatible with inexactness.
Numerical testing
discussion

First a proximal bundle method Difference between different regularizations explained before... large-scale optimization: a metric bundle method instead of a proximal bundle method -> limited memory approach

from PhD-thesis

-

2 Preliminaries

Theoretical Background, nonsmooth Analysis ???

Check if requirements on functions are stated and defined.

2.1 Notation

Throughout this thesis I consider the optimization Problem

$$\min_x f(x), \quad x \in X \subseteq \mathbb{R}^n \tag{1}$$

where f is a possibly nonsmooth function. Also write something about inexactness? specify X more precisely? Convex?
When it comes to nonsmooth objective functions the derivative based framework of nonlinear optimization methods does not work any more. Therefore the most important definitions and results needed when working with nonsmooth functions are stated in this section.
Just definition, lemma, theorem or a bit explanation around it?
better just in Text without Definition, ...
See if requirements in definitions and theorems meet what is needed/provided later.

2.2 Nonsmooth analysis and optimization

A necessary assumption on the objective function f is that it is locally Lipschitz. This assumption assures the well-definedness of the following generalizations of derivatives.

Definition 2.1. [16] A function $f : \mathbb{R}^n \to \mathbb{R}$ is called *locally Lipschitz* if it is Lipschitz on each bounded subset $B \subseteq \mathbb{R}^n$

$$|f(y) - f(x)| \leq C\|y - x\| \quad \forall x, y \in B, \quad C > 0.$$

All convex functions are locally Lipschitz [10].

For convex functions one can define so called subgradients as a generalization of the usual derivative. They are defined using the directional derivative.

Definition 2.2. [10] The *directional derivative* of a convex function f at x in direction d is

$$f'(x,d) := \lim_{\lambda \downarrow 0} \frac{f(x+\lambda d) - f(x)}{\lambda}.$$

Definition 2.3. [10] Let f convex. The *subdifferential* $\partial f(x)$ of f at x is the nonempty compact convex set

$$\partial f(x) = \{g \in \mathbb{R}^n | f'(x,d) \geq \langle g, d \rangle \forall d \in \mathbb{R}^n\}.$$

The subdifferential is a convex set, that supports the graph of the function f from below. If f is differentiable at the point x, the subdifferential reduces to the gradient at that point [10].

This concept was generalized by Clarke for nonconvex functions. First a generalization of the directional derivative is given:

Definition 2.4. [3] Let $f : \mathbb{R}^n \to \mathbb{R}$ locally Lipschitz. The *generalized directional derivative* of f at x in direction d is given by

$$f^\circ(x,d) := \limsup_{\substack{y \to x \\ \lambda \downarrow 0}} \frac{f(y+\lambda d) - f(y)}{\lambda}.$$

This allows for the following definition.

Definition 2.5. [3] The *generalized gradient* of the locally Lipschitz function f at x is a nonempty convex compact set $\partial f(x)$ given by

$$\partial f(x) := \{g \in \mathbb{R}^n | f^\circ(x,d) \geq \langle g, d \rangle \forall d \in \mathbb{R}^n\}.$$

!!!other definition –> take definition from rockefellar/Hare directly from Paper!!!

If f is a convex function the generalized gradient coincides with the subdifferential ∂f of f [3].

Why epsilon Subdifferential?
implementable stopping criterion
dual form of bundle algorithms = same as stopping criterion?

Definition 2.6. [16] The function $f : \mathbb{R}^n \to \mathbb{R}$ is *semismooth* at $x \in \mathbb{R}^n$ if f is Lipschitz on a ball $\mathbb{B}_\varepsilon(x)$around x and for each $d \in \mathbb{R}^n$ and for any sequences $\{t_k\} \subseteq \mathbb{R}_+$, $\{\theta_k\} \subseteq \mathbb{R}^n$ and $\{g_k\} \subseteq \mathbb{R}^n$ such that

$$\{t_k\} \downarrow 0, \quad \{\theta_k/t_k\} \to 0 \in \mathbb{R}^n \quad \text{and} \quad g_k \in \partial f(x + t_k d + \theta_), $$

the sequence $\{\langle g_k, d\rangle\}$ has exactly one accumulation point.

check if I need ∞-functions and if something changes then ϵ-subdifferential
continuity properties of generalized gradient?
definitions from chapter inexact information
definition of inexactness for nonconvex kind of generalization of ϵ-subdifferential for nonconvex case (Noll, inex, nonconv)

3 Bundle Methods

When bundle methods were first introduced in 1975 by Claude Lemaréchal and Philip 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 [18].
To provide an easier understanding of the proximal bundle method in [32] and stress the most important ideas of how to deal with nonconvexity and inexactness first a basic bundle method is shown here.
link to chapter?

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 [8]. I focus here on the primal approach.

In the next two sections the function f is assumed to be convex.

notation, definitions
already done in previous preliminaries chapter?

3.1 A basic bundle method

This section gives a short summery of the derivations and results of chapter XV in [9] 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 [9].

The optimization problem considered in this section is

$$\min_x f(x) \quad \text{s.t.} \quad x \in X \tag{2}$$

with the convex function f and the closed and convex set $X \subseteq \mathbb{R}^n$.

Define Problem again?? Incorporate "set-constraint" by writing $h(x) := f(x) + \mathbb{I}_X$. $\rightarrow$ later???

explanation

3.1.1 Derivation of the bundle 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

$$l_j(x) = f(x^j) + (g^j)^\top (x - x^j) \tag{3}$$

with $f(x^j) = l_j(x^j)$ and due to convexity $f(x) \geq l_j(x),\ x \in X$.
One can now model the objective function f by the piecewise linear function

$$m_k(x) = \max_{j \in J_k} l_j(x) \tag{4}$$

and find a new iterate x^{k+1} by solving the subproblem

$$\min_x m_k(x) \quad \text{s.t.} \quad x \in X. \tag{5}$$

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 one can say that the subproblem does not necessarily have to have a solution. To tackle this problem a penalty term is introduced to the subproblem:

$$\min \tilde{m}_k(x) = m_k(x) + \frac{1}{2t}\|x - x^k\|^2 \quad \text{s.t.} \quad x \in X \tag{6}$$

This new subproblem is strongly convex and has therefore always a unique solution.
how much explanation here? $\max_{j \in J_k} l_j(\hat{x}^k + d)$

Some nice sentences to explain the term a little bit more and to lead over to the next paragraph.
To understand the deeper motivation of this term see [9]. For this introduction it suffices to see that due to the regularization term the subproblem is now strongly convex and

therefore always uniquely solvable.

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 best 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 "big enough", namely bigger 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 looks the following:
Define first the *nominal decrease* δ_k which is the decrease of the model for the new iterate x^{k+1} compared to the function value at the current stability center $\hat{x}^k$.

$$\delta_k = f(\hat{x}^k) - \tilde{m}_k(x^{k+1}) + a_k \geq 0 \tag{7}$$

The nominal decrease is in fact stated a little differently for different versions of the bundle algorithm, this is why I added the constant $a_k \in \mathbb{R}$ here for generalization. In practice the difference between the decreases is not influencing the algorithm as δ_k is weighted by the constant $m \in (0,1)$ for the descent test which compensates a_k.

If the actual decrease of the objective function is bigger than a fraction of the nominal decrease

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

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 remains the same $\hat{x}^{k+1} = \hat{x}^k$.

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

citation for this???!!!

$$\begin{aligned} l_j(x) &= f(x^j) + {g^j}^\top (x - x^j) && (8)\\ &= f(\hat{x}^k) + {g^j}^\top (x - \hat{x}^k) - (f(\hat{x}^k) - f(x^j) + {g^j}^\top (x^j - \hat{x}^k)) && (9) \end{aligned}$$

$$= f(\hat{x}^k) + {g^j}^\top (x - \hat{x}^k) - e_j^k \tag{10}$$

where

$$e_j^k = f(\hat{x}^k) - f(x^j) + {g^j}^\top (x^j - \hat{x}^k) \geq 0 \quad \forall j \in J_k \tag{11}$$

is the *linearization error*. The nonnegativity 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 (6) can now be written as

$$\min_{\hat{x}^k + d \in X} \tilde{m}_k(d) = f(\hat{x}^k) + \max_{j \in J_k} \{{g^j}^\top d - e_j^k\} + \frac{1}{2t_k}\|d\|^2 \tag{12}$$

$$\Leftrightarrow \quad \min_{\hat{x}^k + d \in X, \xi \in \mathbb{R}} \xi + \frac{1}{2t_k}\|d\|^2 \quad \text{s.t.} \quad f(\hat{x}^k) + {g^j}^\top d - e_j^k - \xi \leq 0, \quad j \in J_k \tag{13}$$

where the constant term $f(\hat{x}^k)$ was discarded for the sake of simplicity.
If X is a polyhedron this is a quadratic optimization problem that can be solved using standard methods of nonlinear optimization. The pair (ξ_k, d^k) solves (13) if and only if d^k solves the original subproblem (12) and $\xi_k = f(\hat{x}^k) + \max_{j \in J_k} {g^j}^\top d^k - e_j^k$. The new iterate is then given by $x^{k+1} = \hat{x}^k + d^k$. citation!!!

Remark: Setting $\check{f}(x) = f(x) + \mathbb{I}_X(x)$ the above optimization problem is ...
The *proximal point mapping* or *prox-operator*

$$prox_{t,f}(x) = \arg\min_y \left\{ \check{f}(y) + \frac{1}{2t}\|x - y\|^2 \right\}, \quad t > 0 \tag{14}$$

source??? This special form of the subproblems gives the proximal bundle method its name and will occur again later???

3.1.2 Aggregate objects

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

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

Subproblem (6) then writes as

$$\min_{\hat{x}^k + d \in R^n, \xi \in \mathbb{R}} \xi + \mathbb{I}_X + \frac{1}{2t_k}\|d\|^2 \quad \text{s.t.} \quad {g^j}^\top d - e_j^k - \xi \leq 0, \quad j \in J_k \tag{15}$$

check if f also not put into subproblem before

Some introduction how this and the aggregate error expression relate to each other. Why it is in this case easier to write the model in the nonsmooth form...

Lemma XI 3.1.1 $\partial g = \partial f + \partial \mathbb{I}_X$ for $g = f + \mathbb{I}_X$.

One gets the following results about the step d^k of the subproblem:

Lemma 3.1. *The optimization problem (15) has for $t_k > 0$ a unique solution given by*

$$d^k = -t_k(G^k + \nu^k), \quad G^k \in \partial m_k(d^k), \quad \nu^k \in \partial \mathbb{I}_X. \tag{16}$$

Furthermore

$$m_k(\hat{x}^k + d) \geq f(\hat{x}^k) + {G^k}^\top d - E_k \quad \forall d \in \mathbb{R}^n \tag{17}$$

inequality because of aggregation technique. Is sharp when cutting plane model is used? source?

where

$$E_k := f(\hat{x}^k) - m_k(x^{k+1}) + {G^k}^\top d^k. \tag{18}$$

Comment on the inequality missing

The quantities G^k and E^k are the *aggregate subgradient* and the *aggregate error*.

Explain aggregation process in more detail

From the Karush-Kuhn-Tucker conditions (KKT-conditions) one can see that in the optimum there exist Lagrange or *simplicial multiplier* α_j^k, $j \in J_k$ such that

$$\alpha_j^k \geq 0, \quad \sum_{j \in J_k} \alpha_j^k = 1 \tag{19}$$

by rewriting and so on... one can see that the above expressions are in fact

From the dual problem one obtains that the aggregate subgradient and error can also be expressed as

$$E_k = \sum_{j \in J_k} \alpha_j^k e_j^k \qquad \text{and} \qquad G^k = \sum_{j \in J_k} \alpha_j^k g^j. \tag{20}$$

Finally use Lemma ??? in [9]

$$m_k(x^{k+1}) = f(\hat{x}^k) - E_k - t_k \|G^k\|^2$$

to reformulate the nominal decrease δ_k:

$$\delta_k = f(\hat{x}^k) - m_k(x^{k+1}) - \frac{1}{2} t_k \|G^k\|^2 = E_k + \frac{1}{2} t_k \|G^k\|^2$$

The nominal decrease in this case is defined as:
noch mal anschauen

$$\delta_k := E_k + t_k \|G^k + \nu^k\|^2 = f(\hat{x}^k) - m_k(x^{k+1}) - {\nu^k}^\top d^k \tag{21}$$

In practice the different definition of the decreases makes no difference because of the weighting with the descent parameter m.

The following basic bundle algorithm can now be stated:

Reformluate equations, model function
introduce aggregate expressions
say something to J-update, say something to t-update
see if all abbreviations ($f_j, g^j, ...$) are introduced
introduce prox-operator and proximal points

algorithm

Basic bundle method

Select descent parameter $m \in (0, 1)$ and a stopping tolerance $\texttt{tol} \geq 0$. 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
$$d^k = \arg\min_{d\in\mathbb{R}^n} m_k(\hat{x}^k + d) + \mathbb{I}_X + \frac{1}{2t_k}\|d\|^2$$
and the corresponding Lagrange multiplier $\alpha_j^k,\ j \in J_k$. say how model m_k looks here. include $\mathbb{I}_X$

2. Set
$$G^k = \sum_{j\in J_k} \alpha_j^k g_j^k, \quad E_k = \sum_{j\in J_k} \alpha_j^k e_j^k, \quad \text{and} \quad \delta_k = E_k + t_k\|G^k + \nu^k\|^2$$
If $\delta_k \leq \texttt{tol} \rightarrow$ STOP.

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

4. Compute $f(x^{k+1}),\ g^{k+1}$.
If
$$f^{k+1} \leq \hat{f}^k - m\delta_k \quad \rightarrow \text{serious step.}$$
Set $\hat{x}^{k+1} = x^{k+1},\ f(\hat{x}^{k+1}) = f(x^{k+1})$ and select 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} in a suitable way.

5. Select new bundle index set $J_{k+1} = \{j \in J_k | \alpha_j^{k+1} \neq 0\} \cap k+1$, calculate e_j for $j \in J_{k+1}$ and update the model m_k.

In steps 4 and 5 of the algorithm the updates of the steplength t_k and the index set J_k are are only given in a very general form.
The "suitable" choice of t_k will be discussed more closely in the convergence analysis of decide which method; say that $t_k > 0 \forall k$....
Comment on J_k update $\rightarrow$ depends on what is included in thesis.
For the choice of the new index set J_{k+1} different aggregation methods to keep the memory size controllable are available. The most easy and intuitive one is to just take those parts of the model function, that are actually active in the current iteration. This is done in this basic version of the method.
Refer to low memory bundling if later in thesis. Instead of keeping every index in the set J_k different compression ideas exist. For now I therefor stick to this update.
refer to later "low memory" thing??

explanation to t_k update. $\rightarrow$ include at which point??? This simple idea has however some major drawbacks [10]:

- Minimization of the cutting plane model of the objective function is not trivial. Indeed unconstrained minimization of the model is never possible in the first step, where it is just a line, unless the starting point is already a minimum.
- The convergence speed is very slow.

If convergence speed named here, does it have to be shown (rates)? For all algorithms??? Leave out? Argue about instability?

To address those issues a regularization is added to the cutting plane model. This ensures unique solvability of the minimization of the subproblem. By introducing a stability center and

3.2 ...

possible simplifications of the algorithm

3.2.1 Nonconvex Bundle Methods with Exact Information

Simplification / better results if exact information The main ideas of the algorithm are basicly the ones developed in [8] for the redistributed proximal bundle method for exact nonconvex problems.
Setting the error bounds $\bar{\sigma}$ and $\bar{\theta}$ to zero results therefore in the following convergence theorem.

Theorem 3.2. *Let the sequence* $\{\eta_k\}$ *be bounded,* $\liminf_{k\to\infty}$ *and the cardinality of the set* $\{j \in J_k | \alpha_j^k > 0\}$ *be uniformly bounded in* k.
Then every accumulation point of sequence of serious iterates $\{\hat{x}^k\}$ *is a stationary point of the problem.*

think last condition only interesting in inexact case.
try to gain some insight with generalized ε-subdifferential from Chinese paper:
ε-limiting subdifferential []

In the exact case boundedness of the sequence $\{\eta_k\}$ is proven for lower-$\mathcal{C}^2$ functions in [8]. This is not possible in the inexact case, even if the objective function f is convex.

A further simplification of the method for exact information is not necessary as the method is already almost as simple as the basic bundle method for nonconvex exact functions. Additionally no new concepts needed to be introduced when doing the step from nonconvex exact problems, for which the algorithm was originally designed, to problems with inexact information.

Remark: I want to add here, that the simplicity of the algorithm is rather special for methods suitable for nonconvex problems. Often a linesearch algorithm has to be inserted in the nonconvex case, which is not needed here.

3.2.2 Nonconvex bundle methods

There are different approaches for handling nonconvexity of the objective function in bundle methods. 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* [11, 17]. Here the linearization error is essentially replaced by the nonnegative number

$$\tilde{e}_j^k := \max_{j \in J_k} \{|e_j^k|, \gamma \|\hat{x}^k - x^j\|^2\} \tag{22}$$

or a variation of this expression.
Remark on dual view? How subgradient locality measure measures how close subgradient is to subdifferential of f???
Methods using this kind of manipulation of the model function are often endowed with a line search to provide sufficient decrease of the objective function. For the linesearch to terminate finitely, semismoothness of the objective function is usually needed.
It can be proven that every accumulation point of the sequence of serious points $\{\hat{x}^k\}$ is a stationary point of the objective function f under the additional assumptions that f is locally Lipschitz and the level set $\{x \in \mathbb{R}^n | f(x) \leq f(\hat{x}^1)\}$ is bounded [7].

A drawback to the method described above is that it is primarily supported from the dual point of view of the bundle algorithm. Newer concepts focus also on the primal point of view. This invokes for example having different model functions for the subproblem.

In [5, 6] the difference function

$$h(d) := f(x^j + d) - f(x^j) \quad j \in J_k \tag{23}$$

is approximated to find descent direction of f.
The negative linearization errors are addressed by having 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 the new iterate.
Convergence of the method to a stationary point is proven under the assumption of f being locally Lipschitz and semismooth.
still line search needed
any bounded (level-)sets needed???

In [25] 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:

$$\Phi(y,x) = \phi(y,x) + \frac{1}{2}(y-x)^\top Q(x)(y-x) \tag{24}$$

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 lot in the lines of a general bundle algorithm. The method relies on a smart management of the proximity parameter τ_k which corresponds to $1/t_k$ in the notation of this thesis. This is why the method does not need a linesearch subroutine. For a locally Lipschitz objective function with a bounded levelset $\{x \in \mathbb{R}^n | f(x) \leq f(\hat{x}^1)\}$ convergence to a stationary point is established.

In paper [25] stated that proximity control = proximal bundle Algorithm with smart t_k-control very powerful

Add Luksan in view of Karmitsa Method? Then short introduction of variable metric bundle algorithms necessary; would be managable in this section

For proximal bundle methods: two strategies: line search or (newer) proximity control: It seems that a successful strategy to deal with nonconvexity is proximity control as used in different manners in [1, 14, 24, 22, 25, 27, **?**]

3.2.3 Convex Bundle Methods with Inexact Information

- stronger convergence results possible because of exploitation of convexity
- changes in the algorithm because if convexity should be exploited: inexactness cannot be treated as nonconvexity
-

in extra section??

3.3 How to deal with inexact information in bundle methods?

Partition section in two parts:
1. How is generally dealt with inexact information in the algorithms

2 a) What information is inexact (only subgradients/both...) –> what do you gain from this?
2 b) What kind of assumptions are on the inexactness? (asymptotic, only over- /under-estimation?)

- recognized: fundamentally different? approach for convex and nonconvex functions (at least in algorithm)
  convex: "deal" with inexactness; extra steps...
  nonconvex: generally no difference in algorithm (but for example line search not possible –> only no change, if algorithm was suitable before)
- nonconvex algorithms: inexactness is seen as some kind of nonconvexity –> for function values clear, for subgradients???
- in convex case: often assumption, that gradient is from ε-subdifferential
  is this restrictive? –>

What does "approximate subgradient" mean???
generally seems to be that for convex functions it is the same concept whether one takes the ϵ-subdifferential or a ball around the regular subdifferential.
seems to be the same:

$$\|g_a - g\| \leq \theta \tag{25}$$
$$\Leftrightarrow \quad g_a \in \partial f + B_\theta(0) \tag{26}$$
$$\Leftrightarrow \quad g_a \in \partial_\varepsilon f, \quad \theta \leq \varepsilon^2 \tag{27}$$

Last implication only for convex functions because ε-subdifferential otherwise not defined.
See also papers from "Chinese-search"

Different "degrees" of inexactness: inexact subgradients; also function values (only subgradients easier???); asymptotically ecact; exactness only for serious steps, not at null steps; accuracy controllable or not –> throughout study in in depth paper.

One can clearly see, that 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 [32] presented above and Noll [23]. 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.
In case of convex objective functions changes in the algorithm are more involved. The reason for this is that generally stronger convergence results are possible with inexactness in the convex case than in the nonconvex case. This means however, that the inexactness cannot be incorporated as easily into the algorithm.

Remark on nonconvexity line search and inexactness
A possible reason why there are not already more publication on bundle methods with inexact information in the nonconvex case although there exists a broad variety of algorithms that deal with the exact case could be that many of them incorporate a line search. To make sure that this subalgorithm is finite the objective function has to be semi-smooth **definition of semismoothness, check ....** This however cannot be the case when the functions values of the objective function are only approximated.

4 Application to Model Selection for Primal SVM

4.1 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 [12] and Moore et al. [20].

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 [12]

$$\begin{aligned} \min_{x\in X, y} \quad & F(x,y) && \text{upper level} \\ \text{s.t.} \quad & G(x,y) \leq 0 \\ & y \in \left\{ \begin{array}{ll} \arg\max_{y\in Y} & f(x,y) \\ \text{s.t.} & g(x,y) \leq 0 \end{array} \right\}. && \text{lower level} \end{aligned} \tag{28}$$

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 [12].

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 [26] 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 [20] for support vector regression. However they use a gradient decent method which is not guaranteed to stop at an optimal solution.
In [19] 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.

4.2 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 [4] and the subsequent interest in SVMs lead to a very versatile application of these machines [12].

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 [31].
The overall goal is then to optimize the generalization ability, meaning the ability to predict the output for unseen data correctly [4].

4.2.1 Risk minimization

The concept of SVM's was originally inspired by the statistical learning theory developed by Vapnik. For a throughout analysis see [30].

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*

$$R(\lambda) = \int \mathcal{L}(y, f_\lambda(x)) \mathrm{d}P(x, y) \tag{29}$$

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 $f_{\lambda^*}(x)$ in the chosen function space $\mathcal{F}$ that minimizes this risk functional [31].

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

$$R_{emp}(\lambda) = \frac{1}{l} \sum_{i=1}^{l} \mathcal{L}(y_i, f_\lambda(x^i)), \tag{30}$$

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_{λ^*} that minimizes the risk functional. The theory of consistency provides necessary and sufficient conditions that solve this issue [31].

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 [31]. 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" [31]. This reduces the risk of *overfitting*, meaning to overly fit the function to the training data with the result of poor generalization [4].

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 [12, 30] and references therein.

4.2.2 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 [12].

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 [4].

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

Theorem 4.1. *[4] 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" [4].

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 $(\boldsymbol{w}, b)$:

$$\begin{aligned} \min_{\boldsymbol{w},b,\xi} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\boldsymbol{w}^\top x^i - b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \\ & \forall i = 1, \ldots, l \end{aligned} \tag{31}$$

and

$$\begin{aligned} \min_{\boldsymbol{w},b,\xi} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{l}\xi_i^2 \\ \text{subject to} \quad & y_i(\boldsymbol{w}^\top x^i - b) \geq 1 - \xi_i \\ & \forall i = 1, \ldots, l \end{aligned} \tag{32}$$

The parameter $C > 0$ gives a trade-off between the richness of the chosen set of functions f_alpha to reduce the error on the training data and the danger of overfitting to have good generalization. It has to be chosen a priori [12].

4.3 Explanation Bilevel Approach and Inexact Bundle Method

The hyper-parameter C in the objective function of the classification problem has to be set before hand. 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 [12].

4.3.1 Reformulation as bilevel problem

A more recent approach is the formulation as a bilevel problem used in [12, 20]. 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

$$
\begin{aligned}
\min_{\lambda, f^t} \quad & \mathcal{L}_{upp}\left(\lambda, f^1|_{\Omega_1}, ..., f^T|_{\Omega_T}\right) && \text{upper level} \\
\text{s.t.} \quad & \lambda \in \Lambda \\
& \text{for } t = 1, ..., T: \\
& f^t \in \left\{ \begin{array}{ll} \arg\min\limits_{f \in \mathcal{F}} & \mathcal{L}_{low}(\lambda, f, (x^i, y_i)_{i=1}^l \in \bar{\Omega}_t) \\ \text{s.t.} & g_{low}(\lambda, f) \leq 0 \end{array} \right\}. && \text{lower level}
\end{aligned}
\tag{33}
$$

In the case of support vector classification the T inner problems are one of the classical SVM formulations (31) or (32) (but all T problems have the same formulation). The problem can also be rewritten into a unconstrained form. This form will be 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

$$
(\boldsymbol{w}^t, b_t) \in \arg\min_{\boldsymbol{w}, b} \left\{ \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \sum_{i \in \bar{\mathcal{N}}_t} \max\left(1 - y_i(\boldsymbol{w}^\top x^i - b), 0\right) \right\} \tag{34}
$$

or

$$
(\boldsymbol{w}^t, b_t) \in \arg\min_{\boldsymbol{w}, b} \left\{ \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \sum_{i \in \bar{\mathcal{N}}_t} \left(\max\left(1 - y_i(\boldsymbol{w}^\top x^i - b), 0\right)\right)^2 \right\} \tag{35}
$$

Where the hyper-parameter $\lambda = \frac{1}{C}$ was used due to numerical stability [12].

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

$$
\mathcal{L}_{mis} = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{|\mathcal{N}_t|} \sum_{i \in \mathcal{N}_t} \left[-y_i((\boldsymbol{w}^t)^\top x - b_t)\right]_\star \tag{36}
$$

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

$$(r_\star)_i = \begin{cases} 1, & \text{if } r_i > 0, \\ 0, & \text{if } r_i \leq 0 \end{cases} . \tag{37}$$

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

$$\mathcal{L}_{hinge} = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t}\max\Big(1 - y_i((\boldsymbol{w}^t)^\top x - b_t), 0\Big). \tag{38}$$

Hence the two final resulting bilevel formulations for model selection in support vector are

$$\begin{aligned} \min_{\boldsymbol{W},\boldsymbol{b}} \quad & \mathcal{L}_{hinge}(\boldsymbol{W},\boldsymbol{b}) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t}\max\Big(1 - y_i((\boldsymbol{w}^t)^\top x - b_t), 0\Big) \\ \text{subject to} \quad & \lambda > 0 \\ & \text{for } t = 1, ..., T \\ & (\boldsymbol{w}^t, b_t) \in \arg\min_{\boldsymbol{w},b}\left\{\frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \sum_{i\in\bar{\mathcal{N}}_t}\max\Big(1 - y_i(\boldsymbol{w}^\top x^i - b), 0\Big)\right\} \end{aligned} \tag{39}$$

and

$$\begin{aligned} \min_{\boldsymbol{W},\boldsymbol{b}} \quad & \mathcal{L}_{hinge}(\boldsymbol{W},\boldsymbol{b}) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t}\max\Big(1 - y_i((\boldsymbol{w}^t)^\top x - b_t), 0\Big) \\ \text{subject to} \quad & \lambda > 0 \\ & \text{for } t = 1, ..., T \\ & (\boldsymbol{w}^t, b_t) \in \arg\min_{\boldsymbol{w},b}\left\{\frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \sum_{i\in\bar{\mathcal{N}}_t}\Big(\max\Big(1 - y_i(\boldsymbol{w}^\top x^i - b), 0\Big)\Big)^2\right\}. \end{aligned} \tag{40}$$

4.3.2 The Inexact Bundle Method

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 outcoming w and b to 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.

-> 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

$$0 \in \partial(w, b)\mathcal{L}_{low}(\lambda, w, b). \tag{41}$$

$\partial(w, b)$ 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 [12] 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 hyper-parameter. The reformulated bilevel problem becomes:

$$\begin{aligned} \min_{\boldsymbol{W},\boldsymbol{b}} \quad & \mathcal{L}_{hinge}(\boldsymbol{W}, \boldsymbol{b}) = \frac{1}{T}\sum_{t=1}^{T} \frac{1}{|\mathcal{N}_t|} \sum_{i \in \mathcal{N}_t} \max\left(1 - y_i((\boldsymbol{w}^t)^\top x - b_t), 0\right) \\ \text{subject to} \quad & \lambda > 0 \\ & \text{for } t = 1, ..., T \\ & 0 \in \partial(w, b)\mathcal{L}_{low}(\lambda, w^t, b_t) \end{aligned} \tag{42}$$

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. [20] 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

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

subgradients $\mathcal{G}_{upp,\dots}$ 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 [20]

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

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

$$\partial_w \mathcal{L}_{upp} = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{\mathcal{N}_t}\sum_{i\in\mathcal{N}_t}\begin{cases} -y_i x^i & \text{if } \delta_i > 0 \\ 0 & \text{if } \delta_i \leq 0 \end{cases} \tag{43}$$

$$\partial_b \mathcal{L}_{upp} = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{\mathcal{N}_t}\sum_{i\in\mathcal{N}_t}\begin{cases} y_i & \text{if } \delta_i > 0 \\ 0 & \text{if } \delta_i \leq 0 \end{cases} \tag{44}$$

here at the kink subgradient 0 is taken

for hingequad: -> here subgradient
optimality condition:

$$0 = \partial_{\boldsymbol{w}} \mathcal{L}_{low} = \lambda \boldsymbol{w} + 2\sum_{i\in\bar{\mathcal{N}}_t}\begin{cases} (1 - y_i(w^\top x^i - b))(-y_i x^i) & \text{if } \delta_i > 0 \\ 0 & \text{if } \delta_i \leq 0 \end{cases} \tag{45}$$

$$0 = \partial_b \mathcal{L}_{low} = 2\sum_{i\in\bar{\mathcal{N}}_t}\begin{cases} (1 - y_i(w^\top x^i - b))(y_i) & \text{if } \delta_i > 0 \\ 0 & \text{if } \delta_i \leq 0 \end{cases} \tag{46}$$

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

$$0 = \boldsymbol{w} + \lambda \partial_\lambda \boldsymbol{w} + 2\sum_{i\in\bar{\mathcal{N}}_t}\begin{cases} (-y_i(\partial_\lambda w^\top x^i - \partial_\lambda b))(-y_i x^i) & \text{if } \delta_i > 0 \\ 0 & \text{if } \delta_i \leq 0 \end{cases} \tag{47}$$

$$0 = 2\sum_{i\in\bar{\mathcal{N}}_t}\begin{cases} (-y_i(\partial_\lambda w^\top x^i - \partial_\lambda b))(y_i) & \text{if } \delta_i > 0 \\ 0 & \text{if } \delta_i \leq 0 \end{cases} \tag{48}$$

From this the needed subgradients can be calculated via:

$$2 \cdot \begin{pmatrix} \sum_{i \in \bar{\mathcal{N}}_t} \frac{\lambda}{2} + y_i^2 x^i (x^i)^\top & \sum_{i \in \bar{\mathcal{N}}_t} -y_i^2 x^i \\ \sum_{i \in \bar{\mathcal{N}}_t} -y_i^2 (x^i)^\top & \sum_{i \in \bar{\mathcal{N}}_t} y_i^2 \end{pmatrix} \cdot \begin{pmatrix} \partial_\lambda w \\ \partial_\lambda b \end{pmatrix} = \begin{pmatrix} -w \\ 0 \end{pmatrix} \tag{49}$$

for hinge not quad:

not as much information in the subgradient/derivative

similar calculation leads to

$$\partial_\lambda w = -\frac{w}{\lambda} \tag{50}$$

$$\partial_\lambda b = 0 \tag{51}$$

4.3.3 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 49 and a subgradient of the upper level problem by 4.3.2. 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 $\rightarrow$ 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

4.4 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 [12] 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

$$E_{test} = \frac{1}{l_{test}} \sum_{i=1}^{l_{test}} \frac{1}{2} |sign\left(\boldsymbol{w}^\top x^i - b\right) - y_i| \tag{52}$$

Table ??? shows the results

Data set	Method	CV Error	Test Error	Time (sec.)
`pima`	hingequad	60.72 ± 9.56	24.11 ± 2.71	2.15 ± 0.52
	hinge loss			
`cancer`	hingequad	10.75 ± 7.52	3.41 ± 1.16	3.43 ± 28.84
	hinge loss			
`heart`	hingequad	48.73 ± 5.53	15.56 ± 4.44	3.43 ± 43.39
	hinge loss			
`ionosphere`	hingequad	39.30 ± 5.32	12.21 ± 4.10	14.17 ± 51.27
	hinge loss			

Table 2:

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.

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] Adil Bagirov, Napsu Karmitsa, and Marko M. Mäkelä. *Introduction to Nonsmooth Optimization: Theory, Practice and Software*. Springer International Publishing Switzerland, 2014.

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

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

[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] 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.

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

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

[10] 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.

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

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

[13] Claude Lemaréchal. Nonsmooth optimization and descent methods. Iiasa research report, International Institute for Applied Systems Analysis, 1978.

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

[15] Marko M. Mäkelä and Pekka Neittaanmäki. *Nonsmooth Optimization: Analysis and*

Algorithms with Applications to Optimal Control. World Scientific Pub Co Pte Lt, 1992.

[16] Robert Mifflin. Semismooth and semiconvex functions in constrained optimization. *SIAM Journal on Control and Optimization*, 15(6):959–972, 1977.

[17] 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.

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

[19] 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.

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

[21] Yurii Nesterov and Vladimir Shikhman. Algorithmic principle of least revenue for finding market equilibria. In Boris Goldengorin, editor, *Optimization and Its Applications in Control and Data Sciences*, volume 115 of *Springer Optimization and Its Applications*, pages 381–435. Springer Nature, 2016.

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

[23] 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.

[24] 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.

[25] 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.

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

[27] 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.

[28] A. J. Smola, S.v.n. Vishwanathan, and V. Le Quoc. Bundle methods for machine learning. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems*, number 20, 2007.

[29] Choon Hui Teo, A. J. Smola, S.V.N. Vishwanathan, and Quoc V. Le. Bundle meth-

ods for regulized risk minimization. 2010.

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

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

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