Contents

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 $\partial f(x) = \nabla f(x)$ [11].

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. [1]) 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 [5] 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 [1]. 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. [6]) 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.

Auffallig: Noll Algo scheint viel schneller durchyulaufen, warum???

2 Variable Metric Bundle Method

can I call this variable metric method or does this imply variable metric updates and not solving the bundle subproblem?

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 **??** with the second order model function used by Noll in [**?**] to a metric bundle method suitable for nonconvex functions with noise.

The section starts by explaining the ideas from [?] 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 (??). We also keep the names and definitions of the objects used in section ??.

2.1 The Main Ingredients to the Method

As already mentioned in section ?? 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 does, incorporating it into the method could speed up convergence.

2.1.1 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

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

As explained in [?] like (??) 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 [?]. 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 [?] for convex functions. Lukšan and Vlček use an algorithm in those lines in [?] which is adapted to a limited memory setting by Haarala et al. in [?].

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 [?] and [?].

2.1.2 Noll's Second Order Model

In [?] 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:

$$\Phi(x, \hat{x}) = \phi(x, \hat{x}) + \frac{1}{2}\langle x - \hat{x}, Q(\hat{x})(x - \hat{x})\rangle \tag{2.1}$$

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 (??) 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 [?] 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

$$\min_{\hat{x}^k + d \in X} M_k(\hat{x}^k + d) + \frac{1}{2}\langle d, \left(Q_k + \frac{1}{t_k}\mathbb{I}\right) d\rangle. \tag{2.2}$$

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 hast 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 **??** 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 [**?**, **?**, **?**]?

2.2 ???

In this section a concrete update rules for Q_k are described as well as the minor changes that have to be done to the proximal bundle algorithm from section **??** to adapt it to the variable metric approach.

2.2.1 ???

In [**?**] and [**?**] 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.

To assure these conditions we adapt a the usual BFGS update rule.

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

Where for y^k instead of the difference of gradients of f the difference $y^k := g^{k+1} - g^k$ of subgradients of f is taken. The starting matrix $Q_1 = \mathbb{I}$.

Of course the BFGS update is symmetric. To assure boundedness of the matrix Q_{k+1} the updates are manipulated in the following way:
3 possibilities:
scaled BFGS-update
scaled SR1-update
LBFGS-update - Nocedal Paper from email

BFGS/Verfahren
seems to need less steps with ok optimality if updates are skipped and not matrix scaling

is used (used y'd < ...)
try norm(d)/norm(y) -> not good; use the one above

SR1-Update
auch ok, braucht anderen bound um zu konvergieren

A different procedure is used in [?]. There the update is just skipped whenever $\langle y^k, d^k \rangle < \zeta$.

- Q only updated in serious steps - why?
- comment in SR1 update? Null steps?
- write down exact update from Vlcek
- find out how the different properties are assured

2.2.2 The Descent Measure

There are some minor changes that have to be made compared to the algorithm proposed by Hare et al. the biggest being the descent measure δ_k.

In the same way as for (**??**) 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

$$S^k + \nu^k = -\left(Q + \frac{1}{t_k}\mathbb{I}\right) d^k. \tag{2.3}$$

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

From this the model decrease (**??**) can be recovered using (**??**), (**??**) and (2.3):

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

The new δ_k is used in the same way as in algorithm ??.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 ?? are still valid.

2.3 Algorithm

same form as Hare algorithm (nullstep)
add Q calculation
Algorithm 2.1:

Nonconvex Variable Metric Bundle Method with Inexact Information

Select parameters $m \in (0,1), \gamma > 0$ and a stopping tolerance `tol` ≥ 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 $\delta_k \leq$ `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 $t_{k+1} > 0$.
Calculate $Q(\hat{x}^k)$ ... 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}, keeping all active elements. Calculate

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

and update the model M^k.

2.4 Convergence Analysis

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 [**?**].

In the paper all convergence properties are first stated in [**?**, 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 2.1 ([**?**, 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*

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

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

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

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*

$$0 \in (\partial f(\bar{x}) + \partial \mathtt{i}_X(\bar{x})) + B_{\bar{\theta}}(0). \tag{2.4}$$

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

$$f(y) \geq f(\bar{x}) - (\bar{\theta} + \varepsilon)\|y - \bar{x}\| - 2\bar{\sigma}, \qquad \textit{for all } y \in X \cup B_\rho(\bar{x}). \tag{2.5}$$

As the neither the stabilization nor the descent test is involved in the proof of Lemma 2.1 it is the same as in [**?**].

We prove now that also the variable metric version of the algorithm fulfills all requirements of Lemma 2.1. 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 2.2 *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:*

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

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

$$Ad = A \sum_i \alpha_i v^i = \sum_i \alpha_i \lambda_i v^i. \tag{2.6}$$

Plugging the assumption $A \prec \xi \mathbb{I}$ which is equivalent to $\max_i \lambda_i < \xi$ into (2.6) we get relation (2.4) by

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

□

Theorem 2.3 (c.f.[**?**, 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 2.1 hold.

The proof is very similar to the one stated in [**?**] 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

$$\hat{f}_{k+1} \leq \hat{f}_k - m\delta_k \tag{2.7}$$

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 (2.7), 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 [?]. 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

$$-c^{k+1}_{k+1} + \left\langle s^{k+1}_{k+1}, x^{k+1} - \hat{x}^k \right\rangle \geq -m\delta_k \tag{2.8}$$

holds.

Theorem 2.4 (c.f. [?, Theorem 7]) *Let a finite number of serious iterates be followed by infinite null steps. Let the sequence $\{\eta_k\}$ be bounded and* $\liminf k \to \infty > 0$.
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 2.1 hold for $\bar{x} = \hat{x}$.

Proof: Let k be large enough such that $k \geq \bar{k}$ and $\hat{x}^k = \hat{x}$ and $\hat{f}_k = \hat{f}$ are fixed. Define the optimal value of the subproblem (2.2) by

$$\Psi_k := M_k(x^{k+1}) + \left(d^k\right)^\top \frac{1}{2} \left(Q + \frac{1}{t_k} \mathbb{I}\right) d^k. \tag{2.9}$$

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

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

Using (2.3) for the second equality, the subgradient inequality for $\nu^k \in \partial \mathtt{i}_D$ in the first inequality and (**??**) 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

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

where the first inequality comes from (**??**) and the fact that $t_{k+1} \leq t_k$ for null steps.The second equality follows from (**??**), the third equation by (2.3) and (2.9) and the last inequality holds y $\nu^k \in \partial \mathtt{i}_X(x^{k+1})$.

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

$$
\|d^{k+1} - d^k\| \to 0, \quad k \to \infty. \tag{2.10}
$$

By definition (2.2.2) 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

$$\delta^{k+1} \leq \hat{f} - M_{k+1}(\hat{x} + d^{k+1}). \tag{2.11}$$

By assumption (**??**) 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 nullstep case $\hat{f}_{k+1} = \hat{f}$ so adding condition (2.8) 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 (2.11) 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 (2.10) and the boundedness of $\{\eta_k\}$ using [**?**, Lemma 3, p.45] it follows from (2.4) that

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

From formulation (2.2.2) 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 $\liminf_{k\to\infty} > 0$ 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 $\|S^k +$
$vu^k\| \to 0$ 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 2.1 are fulfilled.

□

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

2.5 Numerical Testing

- Vergleich mit Hare Version
- Genauigkeit
- Geschwindigkeit

3 Application to Model Selection for Primal SVM

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

3.1 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 [7] and Moore et al. [9].

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 [7]

$$
\begin{aligned}
\min_{x\in X,y} \quad & F(x,y) & & \text{upper level} \\
\text{s.t.} \quad & G(x,y) \leq 0 & & \\
& y \in \left\{ \begin{array}{cc} \underset{y\in Y}{\arg\max} & f(x,y) \\ \text{s.t.} & g(x,y) \leq 0 \end{array} \right\}. & & \text{lower level}
\end{aligned}
\tag{3.1}
$$

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 [7].

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 [10] 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 [9] for support vector regression. However they use a gradient decent method which is not guaranteed to stop at an optimal solution.
In [8] he also suggests the nonconvex exact bundle algorithm of Fuduli et al. [3] 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.

3.2 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 [2] and the subsequent interest in SVMs lead to a very versatile application of these machines [7].

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 [14].
The overall goal is then to optimize the generalization ability, meaning the ability to predict the output for unseen data correctly [2].

3.2.1 Risk minimization

The concept of SVM's was originally inspired by the statistical learning theory developed by Vapnik. For a throughout analysis see [13].
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*

$$R(\lambda) = \int \mathcal{L}(y, f_\lambda(x))\mathrm{d}P(x, y) \tag{3.2}$$

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 [14].

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

$$R_{emp}(\lambda) = \frac{1}{l}\sum_{i=1}^{l}\mathcal{L}(y_i, f_\lambda(x^i)), \tag{3.3}$$

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 [14].

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 [14]. 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” [14, p. 994]. This reduces the risk of *overfitting*, meaning to overly fit the function to the training data with the result of poor generalization [2].

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 [7, 13] and references therein.

3.2.2 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 [7].

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 [2].

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

Theorem 3.1 ([2, 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" [2].

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

$$\begin{aligned} \min_{\boldsymbol{w},b,\xi} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\boldsymbol{w}^\top x^i - b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \\ & \forall i = 1, \ldots, l \end{aligned} \tag{3.4}$$

and

$$\begin{aligned} \min_{\boldsymbol{w},b,\xi} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{l}\xi_i^2 \\ \text{subject to} \quad & y_i(\boldsymbol{w}^\top x^i - b) \geq 1 - \xi_i \\ & \forall i = 1, \ldots, l \end{aligned} \tag{3.5}$$

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 [7].

3.3 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 [7].

3.3.1 Reformulation as bilevel problem

A more recent approach is the formulation as a bilevel problem used in [7, 9]. 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

$$\begin{aligned} \min_{\lambda, f^t} \quad & \mathcal{L}_{upp}\left(\lambda, f^1|_{\Omega_1}, ..., f^T|_{\Omega_T}\right) && \text{upper level} \\ \text{s.t.} \quad & \lambda \in \Lambda \\ & \text{for } t = 1, ..., T: \\ & f^t \in \left\{ \begin{array}{ll} \underset{f \in \mathcal{F}}{\arg\min} & \mathcal{L}_{low}(\lambda, f, (x^i, y_i)_{i=1}^l \in \bar{\Omega}_t) \\ \text{s.t.} & g_{low}(\lambda, f) \leq 0 \end{array} \right\}. && \text{lower level} \end{aligned} \tag{3.6}$$

In the case of support vector classification the T inner problems are one of the classical SVM formulations (3.4) or (3.5) (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

$$(\boldsymbol{w}^t, b_t) \in \underset{\boldsymbol{w}, b}{\arg\min} \left\{ \frac{\lambda}{2} \|\boldsymbol{w}\|_2^2 + \sum_{i \in \bar{\mathcal{N}}_t} \max\left(1 - y_i(\boldsymbol{w}^\top x^i - b), 0\right) \right\} \tag{3.7}$$

or

$$(\boldsymbol{w}^t, b_t) \in \arg\min_{\boldsymbol{w},b} \left\{ \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \sum_{i\in\bar{\mathcal{N}}_t} \Big(\max\Big(1 - y_i(\boldsymbol{w}^\top x^i - b), 0\Big)\Big)^2 \right\} \tag{3.8}$$

Where the hyper-parameter $\lambda = \frac{1}{C}$ was used due to numerical stability [7].

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

$$\mathcal{L}_{mis} = \frac{1}{T}\sum_{t=1}^{T} \frac{1}{|\mathcal{N}_t|} \sum_{i\in\mathcal{N}_t} \Big[-y_i((\boldsymbol{w}^t)^\top x - b_t)\Big]_\star \tag{3.9}$$

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

$$(r_\star)_i = \begin{cases} 1, & \text{if } r_i > 0, \\ 0, & \text{if } r_i \leq 0 \end{cases}. \tag{3.10}$$

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

$$\mathcal{L}_{hinge} = \frac{1}{T}\sum_{t=1}^{T} \frac{1}{|\mathcal{N}_t|} \sum_{i\in\mathcal{N}_t} \max\Big(1 - y_i((\boldsymbol{w}^t)^\top x - b_t), 0\Big). \tag{3.11}$$

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

$$\begin{aligned}
\min_{\boldsymbol{W},\boldsymbol{b}} \quad & \mathcal{L}_{hinge}(\boldsymbol{W},\boldsymbol{b}) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t}\max\left(1-y_i((\boldsymbol{w}^t)^\top x-b_t),0\right) \\
\text{subject to} \quad & \lambda > 0 \\
& \text{for } t=1,...,T \\
& (\boldsymbol{w}^t,b_t)\in\arg\min_{\boldsymbol{w},b}\left\{\frac{\lambda}{2}\|\boldsymbol{w}\|_2^2+\sum_{i\in\bar{\mathcal{N}}_t}\max\left(1-y_i(\boldsymbol{w}^\top x^i-b),0\right)\right\}
\end{aligned} \tag{3.12}$$

and

$$\begin{aligned}
\min_{\boldsymbol{W},\boldsymbol{b}} \quad & \mathcal{L}_{hinge}(\boldsymbol{W},\boldsymbol{b}) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t}\max\left(1-y_i((\boldsymbol{w}^t)^\top x-b_t),0\right) \\
\text{subject to} \quad & \lambda > 0 \\
& \text{for } t=1,...,T \\
& (\boldsymbol{w}^t,b_t)\in\arg\min_{\boldsymbol{w},b}\left\{\frac{\lambda}{2}\|\boldsymbol{w}\|_2^2+\sum_{i\in\bar{\mathcal{N}}_t}\left(\max\left(1-y_i(\boldsymbol{w}^\top x^i-b),0\right)\right)^2\right\}.
\end{aligned} \tag{3.13}$$

3.3.2 The Inexact Bundle Method

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.

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

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

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

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

opt point w*

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

For these calculations to be possible the inner level loss function must yield a linear optimality condition in w. This is for example the case for SVM loss functions with a squared one- or two-norm. 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

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

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

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

why H invertible??? - because we assume existence of w???

now for subgradients

In practice we cannot expect $\mathcal{L}_{low}$ to satisfy such strong differentiability properties.

what kind of differentiability do I assume??? -> if I use normal Hinge then only locally Lipschitz L_in

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

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[4, 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 [1, see Theorem 2.5.1].

In practice this 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.

Notation

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

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

$$\partial^a f(a^*, \bar{b}, \bar{c}, ...).$$

A subgradient of this subdifferential is written $g^a \in \partial^a f(a^*, \bar{b}, \bar{c}, ...)$.

what else???

explanation

To calculate a subgradient

Chain rule for subdifferential
before: definition of subgradient in $\mathbb{R}^m$

Theorem 3.2 (c.f. [12, 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*

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

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

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 [7] 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:

$$\begin{aligned} \min_{\boldsymbol{W},\boldsymbol{b}} \quad & \mathcal{L}_{hinge}(\boldsymbol{W},\boldsymbol{b}) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t}\max\Big(1-y_i((\boldsymbol{w}^t)^\top x-b_t),0\Big) \\ \text{subject to} \quad & \lambda>0 \\ & \text{for } t=1,...,T \\ & 0\in\partial(w,b)\mathcal{L}_{low}(\lambda,w^t,b_t) \end{aligned} \tag{3.14}$$

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. [9] 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 [9]

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

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

$$\partial_w \mathcal{L}_{upp} = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{\mathcal{N}_t} \sum_{i \in \mathcal{N}_t} \begin{cases} -y_i x^i & \text{if } \delta_i > 0 \\ 0 & \text{if } \delta_i \leq 0 \end{cases} \tag{3.15}$$

$$\partial_b \mathcal{L}_{upp} = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{\mathcal{N}_t} \sum_{i \in \mathcal{N}_t} \begin{cases} y_i & \text{if } \delta_i > 0 \\ 0 & \text{if } \delta_i \leq 0 \end{cases} \tag{3.16}$$

here at the kink subgradient 0 is taken

for hingequad: -> here subgradient
optimality condition:

$$0 = \partial_{\boldsymbol{w}} \mathcal{L}_{low} = \lambda \boldsymbol{w} + 2 \sum_{i \in \bar{\mathcal{N}}_t} \begin{cases} (1 - y_i(w^\top x^i - b))(-y_i x^i) & \text{if } \delta_i > 0 \\ 0 & \text{if } \delta_i \leq 0 \end{cases} \tag{3.17}$$

$$0 = \partial_b \mathcal{L}_{low} = 2 \sum_{i \in \bar{\mathcal{N}}_t} \begin{cases} (1 - y_i(w^\top x^i - b))(y_i) & \text{if } \delta_i > 0 \\ 0 & \text{if } \delta_i \leq 0 \end{cases} \tag{3.18}$$

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

$$0 = \boldsymbol{w} + \lambda \partial_\lambda \boldsymbol{w} + 2 \sum_{i \in \bar{\mathcal{N}_t}} \begin{cases} (-y_i(\partial_\lambda w^\top x^i - \partial_\lambda b))(-y_i x^i) & \text{if } \delta_i > 0 \\ 0 & \text{if } \delta_i \leq 0 \end{cases} \tag{3.19}$$

$$0 = 2 \sum_{i \in \bar{\mathcal{N}_t}} \begin{cases} (-y_i(\partial_\lambda w^\top x^i - \partial_\lambda b))(y_i) & \text{if } \delta_i > 0 \\ 0 & \text{if } \delta_i \leq 0 \end{cases} \tag{3.20}$$

From this the needed subgradients can be calculated via:

$$2 \cdot \begin{pmatrix} \sum_{i \in \bar{\mathcal{N}_t}} \frac{\lambda}{2} + y_i^2 x^i (x^i)^\top & \sum_{i \in \bar{\mathcal{N}_t}} -y_i^2 x^i \\ \sum_{i \in \bar{\mathcal{N}_t}} -y_i^2 (x^i)^\top & \sum_{i \in \bar{\mathcal{N}_t}} y_i^2 \end{pmatrix} \cdot \begin{pmatrix} \partial_\lambda w \\ \partial_\lambda b \end{pmatrix} = \begin{pmatrix} -w \\ 0 \end{pmatrix} \tag{3.21}$$

for hinge not quad:

not as much information in the subgradient/derivative

similar calculation leads to

$$\partial_\lambda w = -\frac{w}{\lambda} \tag{3.22}$$

$$\partial_\lambda b = 0 \tag{3.23}$$

3.3.3 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 3.21 and a subgradient of the upper level problem by 3.3.2. 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 $\delta_k \leq$ `tol` → 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

3.4 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 [7] 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

$$E_{test} = \frac{1}{l_{test}} \sum_{i=1}^{l_{test}} \frac{1}{2} |sign\left(\boldsymbol{w}^\top x^i - b\right) - y_i| \tag{3.24}$$

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.

Rechtschreibfehler, Namen, Stil überprüfen

References

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

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

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

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

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

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

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

[12] 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.

[13] 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.

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

[15] 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.

[16] 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.

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

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

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

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

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

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

[23] 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.