Contents

1 Application to Model Selection for Primal SVM

Skalarprodukt anpassen, Vektoren nicht fett oder neue definition, notation

1.1 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 [3] and Moore et al. [5].

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 [3]

$$\begin{aligned} \min_{x\in X, y} \quad & F(x,y) && \text{upper level} \\ \text{s.t.} \quad & G(x,y) \leq 0 \\ & y \in \left\{ \begin{array}{ll} \arg\max\limits_{y\in Y} & f(x,y) \\ \text{s.t.} & g(x,y) \leq 0 \end{array} \right\}. && \text{lower level} \end{aligned} \tag{1.1}$$

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 [3].

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 [6] 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 [5] for support vector regression. However they use a gradient decent method which is not guaranteed to stop at an optimal solution.
In [4] he also suggests the nonconvex exact bundle algorithm of Fuduli et al. [2] 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.

1.2 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 [1] and the subsequent interest in SVMs lead to a very versatile application of these machines [3].

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 [9].
The overall goal is then to optimize the generalization ability, meaning the ability to predict the output for unseen data correctly [1].

1.2.1 Risk minimization

The concept of SVM's was originally inspired by the statistical learning theory developed by Vapnik. For a throughout analysis see [8].
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*

$$R(\lambda) = \int \mathcal{L}(y, f_\lambda(x)) \mathrm{d}P(x, y) \tag{1.2}$$

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 [9].

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

$$R_{emp}(\lambda) = \frac{1}{l} \sum_{i=1}^{l} \mathcal{L}(y_i, f_\lambda(x^i)), \tag{1.3}$$

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 [9].

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 [9]. 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" [9, p. 994]. This reduces the risk of *overfitting*, meaning to overly fit the function to the training data with the result of poor generalization [1].

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 [3, 8] and references therein.

1.2.2 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 [3].

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 [1].

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

Theorem 1.1 ([1, 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" [1].

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

$$\begin{aligned} \min_{\boldsymbol{w},b,\xi} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\boldsymbol{w}^\top x^i - b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \\ & \forall i = 1, \ldots, l \end{aligned} \tag{1.4}$$

and

$$\begin{aligned} \min_{\boldsymbol{w},b,\xi} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{l}\xi_i^2 \\ \text{subject to} \quad & y_i(\boldsymbol{w}^\top x^i - b) \geq 1 - \xi_i \\ & \forall i = 1, \ldots, l \end{aligned} \tag{1.5}$$

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 [3].

1.3 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 [3].

1.3.1 Reformulation as bilevel problem

A more recent approach is the formulation as a bilevel problem used in [3, 5]. 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

$$
\begin{aligned}
\min_{\lambda, f^t} \quad & \mathcal{L}_{upp}\left(\lambda, f^1|_{\Omega_1}, ..., f^T|_{\Omega_T}\right) && \text{upper level} \\
\text{s.t.} \quad & \lambda \in \Lambda \\
& \text{for } t = 1, ..., T: \\
& f^t \in \left\{ \begin{array}{ll} \arg\min\limits_{f \in \mathcal{F}} & \mathcal{L}_{low}(\lambda, f, (x^i, y_i)_{i=1}^l \in \bar{\Omega}_t) \\ \text{s.t.} & g_{low}(\lambda, f) \leq 0 \end{array} \right\}. && \text{lower level}
\end{aligned}
\tag{1.6}
$$

In the case of support vector classification the T inner problems are one of the classical SVM formulations (1.4) or (1.5) (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

$$
(\boldsymbol{w}^t, b_t) \in \arg\min_{\boldsymbol{w}, b} \left\{ \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \sum_{i \in \bar{\mathcal{N}}_t} \max\left(1 - y_i(\boldsymbol{w}^\top x^i - b), 0\right) \right\} \tag{1.7}
$$

or

$$
(\boldsymbol{w}^t, b_t) \in \arg\min_{\boldsymbol{w}, b} \left\{ \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \sum_{i \in \bar{\mathcal{N}}_t} \left(\max\left(1 - y_i(\boldsymbol{w}^\top x^i - b), 0\right)\right)^2 \right\} \tag{1.8}
$$

Where the hyper-parameter $\lambda = \frac{1}{C}$ was used due to numerical stability [3].

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

$$
\mathcal{L}_{mis} = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{|\mathcal{N}_t|} \sum_{i \in \mathcal{N}_t} \left[-y_i((\boldsymbol{w}^t)^\top x - b_t)\right]_\star \tag{1.9}
$$

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

$$(r_\star)_i = \begin{cases} 1, & \text{if } r_i > 0, \\ 0, & \text{if } r_i \leq 0 \end{cases} . \tag{1.10}$$

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

$$\mathcal{L}_{hinge} = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t}\max\Big(1 - y_i((\boldsymbol{w}^t)^\top x - b_t), 0\Big). \tag{1.11}$$

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

$$\begin{aligned} \min_{\boldsymbol{W},\boldsymbol{b}} \quad & \mathcal{L}_{hinge}(\boldsymbol{W},\boldsymbol{b}) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t}\max\Big(1 - y_i((\boldsymbol{w}^t)^\top x - b_t), 0\Big) \\ \text{subject to} \quad & \lambda > 0 \\ & \text{for } t = 1, ..., T \\ & (\boldsymbol{w}^t, b_t) \in \arg\min_{\boldsymbol{w},b}\left\{\frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \sum_{i\in\bar{\mathcal{N}}_t}\max\Big(1 - y_i(\boldsymbol{w}^\top x^i - b), 0\Big)\right\} \end{aligned} \tag{1.12}$$

and

$$\begin{aligned} \min_{\boldsymbol{W},\boldsymbol{b}} \quad & \mathcal{L}_{hinge}(\boldsymbol{W},\boldsymbol{b}) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t}\max\Big(1 - y_i((\boldsymbol{w}^t)^\top x - b_t), 0\Big) \\ \text{subject to} \quad & \lambda > 0 \\ & \text{for } t = 1, ..., T \\ & (\boldsymbol{w}^t, b_t) \in \arg\min_{\boldsymbol{w},b}\left\{\frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \sum_{i\in\bar{\mathcal{N}}_t}\Big(\max\Big(1 - y_i(\boldsymbol{w}^\top x^i - b), 0\Big)\Big)^2\right\}. \end{aligned} \tag{1.13}$$

1.3.2 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 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

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

Assuming $\mathcal{L}_{low}$ is twice continuously differentiable in w the optimality condition

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

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

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

why H inversable???

To calculate a subgradient Chain rule for subdifferential

Theorem 1.2 (c.f. [7, 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*

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

Then for the sets

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

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 [3] 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:

$$\begin{aligned} \min_{\boldsymbol{W},\boldsymbol{b}} \quad & \mathcal{L}_{hinge}(\boldsymbol{W},\boldsymbol{b}) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t}\max\left(1-y_i((\boldsymbol{w}^t)^\top x-b_t),0\right) \\ \text{subject to} \quad & \lambda > 0 \\ & \text{for } t = 1,...,T \\ & 0 \in \partial(w,b)\mathcal{L}_{low}(\lambda, w^t, b_t) \end{aligned} \tag{1.14}$$

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. [5] 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 $\mathcal{G}_{upp,...}$ 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 [5]

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

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

$$\partial_w \mathcal{L}_{upp} = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{\mathcal{N}_t} \sum_{i \in \mathcal{N}_t} \begin{cases} -y_i x^i & \text{if } \delta_i > 0 \\ 0 & \text{if } \delta_i \leq 0 \end{cases} \tag{1.15}$$

$$\partial_b \mathcal{L}_{upp} = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{\mathcal{N}_t} \sum_{i \in \mathcal{N}_t} \begin{cases} y_i & \text{if } \delta_i > 0 \\ 0 & \text{if } \delta_i \leq 0 \end{cases} \tag{1.16}$$

here at the kink subgradient 0 is taken

for hingequad: -> here subgradient
optimality condition:

$$0 = \partial_{\boldsymbol{w}} \mathcal{L}_{low} = \lambda \boldsymbol{w} + 2 \sum_{i \in \bar{\mathcal{N}}_t} \begin{cases} (1 - y_i(w^\top x^i - b))(-y_i x^i) & \text{if } \delta_i > 0 \\ 0 & \text{if } \delta_i \leq 0 \end{cases} \tag{1.17}$$

$$0 = \partial_b \mathcal{L}_{low} = 2 \sum_{i \in \bar{\mathcal{N}}_t} \begin{cases} (1 - y_i(w^\top x^i - b))(y_i) & \text{if } \delta_i > 0 \\ 0 & \text{if } \delta_i \leq 0 \end{cases} \tag{1.18}$$

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

$$0 = \boldsymbol{w} + \lambda \partial_\lambda \boldsymbol{w} + 2 \sum_{i \in \bar{\mathcal{N}}_t} \begin{cases} (-y_i(\partial_\lambda w^\top x^i - \partial_\lambda b))(-y_i x^i) & \text{if } \delta_i > 0 \\ 0 & \text{if } \delta_i \leq 0 \end{cases} \tag{1.19}$$

$$0 = 2 \sum_{i \in \bar{\mathcal{N}}_t} \begin{cases} (-y_i(\partial_\lambda w^\top x^i - \partial_\lambda b))(y_i) & \text{if } \delta_i > 0 \\ 0 & \text{if } \delta_i \leq 0 \end{cases} \tag{1.20}$$

From this the needed subgradients can be calculated via:

$$2 \cdot \begin{pmatrix} \sum_{i \in \bar{\mathcal{N}}_t} \frac{\lambda}{2} + y_i^2 x^i (x^i)^\top & \sum_{i \in \bar{\mathcal{N}}_t} -y_i^2 x^i \\ \sum_{i \in \bar{\mathcal{N}}_t} -y_i^2 (x^i)^\top & \sum_{i \in \bar{\mathcal{N}}_t} y_i^2 \end{pmatrix} \cdot \begin{pmatrix} \partial_\lambda w \\ \partial_\lambda b \end{pmatrix} = \begin{pmatrix} -w \\ 0 \end{pmatrix} \tag{1.21}$$

for hinge not quad:

not as much information in the subgradient/derivative

similar calculation leads to

$$\partial_\lambda w = -\frac{w}{\lambda} \tag{1.22}$$

$$\partial_\lambda b = 0 \tag{1.23}$$

1.3.3 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 1.21 and a subgradient of the upper level problem by 1.3.2. 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

1.4 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 [3] 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

$$E_{test} = \frac{1}{l_{test}} \sum_{i=1}^{l_{test}} \frac{1}{2} |sign\left(\boldsymbol{w}^\top x^i - b\right) - y_i| \tag{1.24}$$

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

[2] 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.

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

[4] 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.

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

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

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

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

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