Contents

1 Application to Model Selection for Primal SVM

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

1.1 Introduction

In this part of the thesis the nonconvex inexact bundle algorithm is applied to the problem of model selection for *support vector machines* (SVMs) solving classification tasks. It relies on a bilevel formulation proposed by Kunapuli in [3] and Moore et al. in [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 in [3, p. 20]

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

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 by 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, these are then added as constraints to the upper level problem. With this solution method 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* $S(x) = \{y \in \mathbb{R}^k \mid y$, that solves the lower level problem, is can be 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 then 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 1990'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. For a throughout introduction to this subject see also [1]. Here a summary of the most important expressions and results is given.

In classification data from a possibly high dimensional vector space $\tilde{X} \subset \mathbb{R}^n$, 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 a data point x has the label y [8, p. 988]. The overall goal is then to optimize the generalization ability, meaning the ability to predict the output for unseen data correctly [1, chapter 1.2].

1.2.1 Risk minimization

The concept of SVM's was originally inspired by the statistical learning theory developed by Vapnik. A detailed examination of the subject is given in [7]. In [9] the subject is approached from a more explaining point of view.

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*

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

Here $\mathcal{L} : \mathbb{R}^2 \to \mathbb{R}$ is the loss function, $f_\lambda : \mathbb{R}^n \cap \mathcal{F} \to \mathbb{R},\ \lambda \in \Lambda$ the approximate 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 [8, 989].

As the only given information is provided 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

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

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 $f_{\lambda,\mathrm{emp}}$ that minimizes the empirical risk 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 [8, p. 989].

Vapnik therefore introduced the structural risk minimization (SRM) induction principle . It ensures that the used set of functions has a structure that makes it strongly consistent [8]. 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" [8, 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, chapter 1.3].

Support vector machines fulfill all conditions of the SRM principle. Due to the kernel trick that allows for nonlinear classification tasks it is also very powerful. For more detailed information on this see [3] and references therein.

1.2.2 Support Vector machines

In the case of linear binary classification one searches for a an affine hyperplane $w \in \mathbb{R}^n$ shifted by $b \in \mathbb{R}$ to separate the given data. The vector 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

$$f(x) = \text{sign}(\langle w, x \rangle - b).$$

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 (w, b) separates the data, there exist infinitely many hyperplanes (tw, b), $t > 0$, that separate the data in the same way. To have a unique description of a separating hyperplane the *canonical hyperplane for given data* $x \in X$ is defined by

$$f(x) = \langle w, x \rangle - b \quad \text{s.t.} \quad \min_i \left| \left\langle w, x^i \right\rangle - b \right| = 1.$$

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, p. 10].

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.

Proposition 1.1 ([1, Proposition 6.1]) Given a linearly separable training sample $\Omega = \{(x^i, y_i), ..., (x^l, y_l)\}$ the hyperplane (w, b) that solves the optimization problem

$$\|w\|^2 \quad \text{s.t.} \quad y_i(\langle w, x\rangle - b) \geq 1, \quad i = 1, ..., l,$$

realizes a maximal margin hyperplane.

The proof is given in [1, chapter 6.1].

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 (w, b) and target margin γ is

$$\xi_i = \max(0, \gamma - y_i(\langle w, x\rangle + b))$$

If $\xi_i > \gamma$ the point is considered 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 (w, b):

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{l}\xi_i \\ \text{s.t.} \quad & y_i(\left\langle w, x^i\right\rangle - b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \\ & \forall i = 1, \ldots, l \end{aligned} \tag{1.4}$$

and

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{l}\xi_i^2 \\ \text{s.t.} \quad & y_i(\left\langle w, x^i \right\rangle - b) \geq 1 - \xi_i \\ & \forall i = 1, \ldots, l. \end{aligned} \tag{1.5}$$

The first part of the respective objective functions are the regularizations, the second part are the actual loss functions. The parameter $C > 0$ gives a trade-off between the richness of the chosen set of functions f_λ 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]. The two optimization problems only differ in the norm chosen for the loss function. In (1.4) the one-norm is chosen, in (1.5) the squared two-norm is used. Problem (1.5) is the one that is finally used in the bilevel approach where smoothness of the objective function of the inner level problem is needed to calculate all needed subgradients.

1.3 Bilevel Approach and Inexact Bundle Method

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

To prevent overfitting and get a good parameter selection, especially in case of little data, commonly T-fold cross validation is used. For this technique the training data is randomly partitioned into T subsets of equal size. One of these subsets is then left out of the training set 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, p. 30].

1.3.1 Reformulation as Bilevel Problem

A more recent approach is the formulation as a bilevel problem used in [3] and [5]. This makes it possible to optimize the hyper-parameters continuously.

Let $\Omega = \{(x^1, y_1), ..., (x^l, y_l)\} \subset \mathbb{R}^{n+1}$ 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 respectively 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 reads

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

In the case of support vector classification the T inner problems have the classical SVM formulation (1.5). The problem can also be rewritten into an unconstrained form. This form is helpful when using the inexact bundle algorithm for solving the bilevel problem. For the t'th fold the resulting hyperplane is identified with the pair $(w^t, b_t) \in \mathbb{R}^{n+1}$. The inner level problem for the t'th fold can therefore be stated as

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

Where the hyper-parameter $\lambda = \frac{1}{C}$ is used due to numerical stability [3, p. 38].

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 is therefore to pick the misclassification loss, that counts how many data points of the respective validation set Ω_t are misclassified when taking function f^t.

The misclassification loss can be written as

$$\mathcal{L}_{mis} = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t}\left[-y_i\left(\left\langle w^t, x^i\right\rangle - b_t\right)\right]_\star, \tag{1.8}$$

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

$$(r_\star)_i = \begin{cases} 1, & \text{if } r_i > 0, \\ 0, & \text{if } r_i \leq 0 \end{cases}. \tag{1.9}$$

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

$$\mathcal{L}_{hinge} = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t}\max\left(1 - y_i\left(\left\langle w^t, x^i\right\rangle - b_t\right), 0\right). \tag{1.10}$$

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

$$\mathcal{L}_{hinge} = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{N}_t|}\sum_{i\in\mathcal{N}_t}\max\left\{1 - y_i\left(\left\langle w^t, x^i\right\rangle - b_t\right), 0\right\}^2. \tag{1.11}$$

In figure (**??**) it can be seen that its minimum and overall progress is more similar to the misclassification loss than the one of the hinge loss. For this reason we progress taking the squared form of the hinge loss, abbreviating with *hingequad loss* for convenience.
No, take hingeloss because of nonsmoothness

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

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

1.3.2 Solution of the Bilevel Program

1.4 Numerical Experiments

In this section algorithms **??**.1 is used to solve the bilevel problems presented above for different synthetic and real world data sets

Rechtschreibfehler, Namen, Stil überprüfen

References

[1] N. Cristianini and J. 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] G. 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. Technical report, Rensselaer Polytechnic Institute, 2010.

[5] G. Moore, C. Bergeron, and K. P. Bennett. Model selection for primal SVM. *Machine Learning*, 85(1):175–208, 2011.

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

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

[8] V. N. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999.

[9] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer New York, 2013.