20th *Computer Vision Winter Workshop*
Paul Wohlhart, Vincent Lepetit (eds.)
Seggau, Austria, February 9-11, 2015

Continuous Hyper-parameter Learning for Support Vector Machines

Teresa Klatzer[1]
[1]Institute for Computer Graphics and Vision
Graz University of Technology
klatzer@icg.tugraz.at

Thomas Pock[1,2]
[2] Safety & Security Department
AIT Austrian Institute of Technology
pock@icg.tugraz.at

Abstract. *In this paper, we address the problem of determining optimal hyper-parameters for support vector machines (SVMs). The standard way for solving the model selection problem is to use grid search. Grid search constitutes an exhaustive search over a pre-defined discretized set of possible parameter values and evaluating the cross-validation error until the best is found. We developed a bi-level optimization approach to solve the model selection problem for linear and kernel SVMs, including the extension to learn several kernel parameters. Using this method, we can overcome the discretization of the parameter space using continuous optimization, and the complexity of the method only increases linearly with the number of parameters (instead of exponentially using grid search). In experiments, we determine optimal hyper-parameters based on different smooth estimates of the cross-validation error and find that only very few iterations of bi-level optimization yield good classification rates.*

1. Introduction

In the field of machine learning much effort is put in developing new algorithms trying to beat the current record on diverse challenges and benchmarks. What all those methods have in common is that they only work as good as they have been fine-tuned by setting sensible parameters affecting the performance of the algorithms. The support vector machine (SVM) [9, 6, 19] as a particular instance of a machine learning algorithm is a very popular method for supervised classification that finds its application in several disciplines including bioinformatics, text and image recognition. Also for the SVM, setting good hyper-parameters strongly influences the classification performance. The aim of model selection is to find the hyper-parameters such that the performance of the learning algorithm is "optimal". Usually this is done manually, or via some combination of grid search and manual search.

Few parameters (1-2) can be set quite successfully based on the evaluation of the cross-validation (CV) error on a grid of possible parameter values. For many parameters, however, the problem is hard to solve because the search space grows exponentially in the number of parameters. Grid search can easily be parallelized, but one would still need access to a massive computational cluster to solve the problem in reasonable time.

In the past, attempts to reduce the complexity of machine learning algorithms in terms of the number of hyper-parameters have been made. E.g., it is common practice to use linear SVMs e.g. for image classification on pre-computed explicit feature maps of the data [22]. Another example is the concept of multiple kernel SVMs where kernels with different fixed bandwidths are combined using weighted sums of them [1, 21, 11]. Here, the weighting factors are directly included in the training objective of the SVM.

More recent literature suggests that especially in the field of computer vision there is increased popularity of large hierarchical models [3] such as Convolutional Neural Networks [14] or Deep Belief Nets [12] which inherently have a large number hyper-parameters to set.

The idea of using bi-level optimization for determining hyper-parameters is not entirely new. Kunapuli et al. [15] have investigated a similar approach to ours, but they use different methods to deal with the optimization problem and only use available standard solvers which limits them to experiments with a linear SVM. Another approach to use gradient methods to solve the parameter selection problem also for kernel SVMs can be found in [7]. They seek to minimize smoothed estimates of the generalization error

of the SVM w.r.t. the hyper-parameters, however, their investigations are restricted to use error measures where the gradient to the hyper-parameters can directly be computed.

Our contribution is an attempt to solve the model selection problem for linear and kernel SVMs, with extension to several kernel parameters using a bi-level optimization approach. We develop a general optimization scheme that allows for continuous hyper-parameter learning based on estimates of the cross-validation error.

Outline. This paper is organized as follows: In Section 2 we discuss typical methods for hyper-parameter optimization such as grid search methods. In Section 3 we develop the bi-level solution for the SVM in general and extend it to optimize several kernel parameters. Furthermore we discuss the choice of a smoothed higher level loss function to estimate the classification performance. In Section 4, we evaluate the proposed method and compare the different performance measures. In Section 5 we conclude the paper.

2. Grid search and random search

Throughout the machine learning literature, grid search is the chosen method to determine hyper-parameters. It is common practice to estimate the performance of a learning algorithm based on a T-fold cross-validation error H (e.g. [10]). Here, the error is determined on the data that has not been used for training in the respective fold. The hope is that the performance of the learning algorithm based on the T validation sets $\text{data}^{(\text{val})}_{t=1,\dots,T}$ can be successfully transferred to the test set.

Inspired by the discussion about hyper-parameter optimization and grid search/random search in [3], we formalize the problem of hyper-parameter optimization in terms of discrete sets as follows. Let θ be a set of hyper-parameters with cardinality S and θ_k one possible configuration out of K in the discrete search space. Let $w_t(\theta)$ be the separating hyperplane obtained by the SVM training algorithm on training set t using the hyper-parameters θ. The minimization problem addressed by grid search can be written as

$$\underset{\theta\in\{\theta_1,\dots\theta_k\}}{\arg\min} \sum_{t=1}^{T} H(\text{data}^{(\text{val})}_t, w_t(\theta)). \tag{1}$$

For the SVM a typical set of hyper-parameters is e.g. $\theta = (c, \gamma)$: The regularization parameter c controlling the margin and the bandwidth γ of a Gaussian kernel. From this formulation, we can easily deduce that grid search suffers from the curse of dimensionality: Each hyper-parameter $\theta_1, ..., \theta_S$ from the set θ can take a set of values $V_1, ..., V_S$. Then the number of grid search trials is calculated by counting every possible combination of values:

$$\#\text{trials} = \prod_{s=1}^{S} |V_s|. \tag{2}$$

Often, a grid search procedure is accompanied by some degree of manual search to identify promising value sets V_s for each component of θ. Another practical strategy to alleviate the grid search procedure is to perform first a coarse search to identify interesting parameter ranges, and then consequently re-do the grid search on a finer grid. Given access to a computational cluster, grid search can be easily parallelized and run on the distributed system. It is also common to assign a certain computational budget to perform grid search (e.g. measured in trials).

There have been some attempts to tackle the problem of model selection other than grid search e.g. using Bayesian optimization [20], sequential model based optimization [13], or a random search approach by [3]. Using random search [3] better or equal results in hyper-parameter optimization can be achieved compared to standard grid search, using a reduced computational budget. These approaches are interesting if the cardinality of the set θ exceeds $S = 2$, but they cannot be used for arbitrarily high numbers of parameters (in [3] results are presented for $S \leq 32$; determining hyper-parameters for a Deep Belief Network).

What all those approaches neglect is the fact that e.g. for SVMs the hyper-parameters are continuous. Through the discretization, we always lose accuracy in the possible solution. In our approach to solve the hyper-parameter optimization problem on the example of SVMs we exploit this property.

Moreover, grid search or random search procedures are not adaptive. Only by manual intervention, the course of the experiments can be altered such that irrelevant parameter values are not further explored. For the application to SVMs, we propose a continuous bi-level optimization scheme that is indeed adaptive and performs continuous optimization on the (smoothed) error surface we typically get calculating a full grid search.

We will see another advantage of the bi-level optimization scheme: The complexity only grows linearly in the number of hyper-parameters. For each hyper-parameter we want to determine, we

have one additional gradient to compute (see Eq. 12, 16, 19, 23). This makes the method also applicable to more complex formulations of SVMs such as kernel SVMs with highly parametrized kernels or for learning hyper-parameters for a multiple kernel SVM.

3. Proposed method

3.1. Preliminaries

In this paper we use a soft margin formulation of the support vector machine. Assuming training examples $x_i \in \mathbb{R}^{1\times D}$ with $i = 1, ..., N$ and their labels $y_i \in \{-1, 1\}$ we can write the optimization objective for the linear SVM as follows:

$$\begin{aligned} \min_{w,b,\xi} \frac{c}{2}\|w\|_2^2 + \sum_{i=1}^{N} \xi_i \\ s.t. \quad y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i \\ \xi_i \geq 0. \end{aligned} \tag{3}$$

Starting from the primal formulation using the slack variables ξ_i we can write the SVM objective function in its unconstrained form in terms of a loss function $\ell(.)$ (like e.g. in [6]):

$$w^*(\theta) \in \arg\min_{w,b} \left\{ \frac{c}{2}\|w\|_2^2 + \sum_{i=1}^{N} \ell(w, b, x_i, y_i) \right\}. \tag{4}$$

Solving the SVM gives us the optimal soft-margin hyperplane defined by w^*. It is influenced by the regularization parameter c which controls the trade-off between maximizing the margin and minimizing the misclassification error. The loss function in Eq. 4 is the exact Hinge loss

$$\ell(w, b, x_i, y_i) = \max(0, 1 - y_i(\langle w, x_i\rangle + b). \tag{5}$$

It will turn out in Eq. 9 that we require the SVM objective to be twice continuously differentiable, thus we introduce a smooth approximation [23] of Eq. 5 parametrized with μ:

$$\ell_\mu(w, b, x_i, y_i) = \frac{1}{\mu} \log(1 + e^{-\mu(y_i(\langle w,x_i\rangle+b)-1)}) \tag{6}$$

In [23] it is shown that $\ell_\mu(.)$ converges to $\ell(.)$ as $\mu \to \infty$. The actual choice of μ will be discussed in Section 4. Solving the SVM we obtain an optimal soft-margin classifier, but the quality of the solution depends on how we choose the hyper-parameter c. We want to set this parameter such that the CV error on the given data is minimal.

3.2. Bi-level formulation

In the following, we want to formulate the model selection problem as a bi-level optimization problem. Bi-level optimization is a mathematical concept involving a higher level optimization problem with another (lower level) optimization problem as its constraint [8]. The aim is to find the hyper-parameters yielding the minimal cross-validation error subject to the SVM solved using those parameters. The challenge is to set the error measure in connection to the hyper-parameters because it is typically not directly dependent on the hyper-parameters, but only via the optimal hyperplane defined by w^* obtained by minimizing the SVM's energy function E. This relation is depicted in Fig. 1.

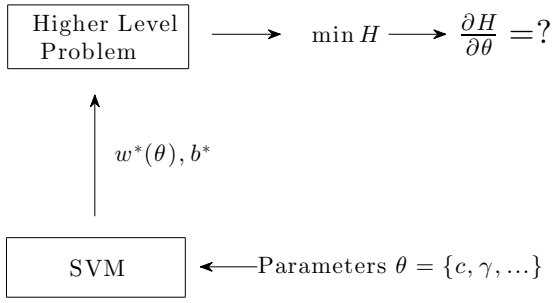

Figure 1. Schema of the bi-level problem.

Formally, we can write:

$$\begin{aligned} \min_{\theta} \sum_{t=1}^{T} H(w_t(\theta), \Xi_t, \eta_t) \\ s.t. \quad w_t(\theta) \in \arg\min_{w_t} E(w_t, \theta, X_t, y_t) \\ t = 1, ..., T. \end{aligned} \tag{7}$$

For simplicity, we use matrix notation in the derivations. Let us define the important symbols. The training set we are given is divided into a training set to calculate the SVM classifier and a validation set to estimate the performance of the trained classifier.

We use an augmented weight vector defined as $w \in \mathbb{R}^{1\times D}$ with D the number of feature dimensions of the input data plus one, including bias b in the end. The training examples $x_i \in \mathbb{R}^{1\times D}$, $i = 1, ..., N$ are condensed in the matrix $X \in \mathbb{R}^{N\times D}$, the validation examples $\zeta_i \in \mathbb{R}^{1\times D}$ are condensed in the matrix $\Xi \in \mathbb{R}^{L\times D}$. Both X and Ξ contain a column of ones in the end for handling the bias implicitly. N and L are the numbers of examples in the training and validation set, respectively. The vectors $y \in \mathbb{R}^{N\times 1}$ and $\eta \in \mathbb{R}^{N\times 1}$ contain the class labels $y_i, \eta_i \in \{-1, 1\}$ for the data. θ is the column vector

of hyper-parameters for the SVM, in the linear case $\theta = c$. There are $t = 1, ..., T$ sets of training and validation data for T-fold cross-validation.

To solve the problem in Eq. 7 we need to reformulate it. We use a Lagrange multipliers λ_t to deal with the lower level constraints:

$$\mathcal{L}(w, \theta, \lambda) = \sum_{t=1}^{T} \left[H(w(\theta)_t, \Xi_t, \eta_t) + \left\langle \lambda_t, \frac{\partial E}{\partial w_t} \right\rangle \right]. \tag{8}$$

We will use the unconstrained form of the bi-level problem from Eq. 8 to calculate the desired gradient $\frac{\partial \mathcal{L}}{\partial \theta}$ via implicit differentiation. This gradient is used to determine the optimal hyper-parameters for the SVM.

For θ^* to be a local minimizer of Eq. 8 the necessary KKT optimality conditions [4] are given by

$$\mathcal{G}(w, \theta, \lambda) = \begin{pmatrix} \frac{\partial H}{\partial w_1} + \frac{\partial^2 E}{\partial w_1^2} \lambda_1 \\ \vdots \\ \frac{\partial H}{\partial w_T} + \frac{\partial^2 E}{\partial w_T^2} \lambda_T \\ \sum_{t=1}^{T} \left(\frac{\partial H(w_t, .)}{\partial \theta} + \frac{\partial^2 E}{\partial w_t \partial \theta} \lambda_t \right) \\ \frac{\partial E}{\partial w_1} \\ \vdots \\ \frac{\partial E}{\partial w_T} \end{pmatrix} = 0. \tag{9}$$

From the structure of Eq. 9 we observe that the SVM's energy function has to be twice continuously differentiable. This fact gives rise to use the smooth approximation of the Hinge loss in Eq. 6. Likewise, we also need a smooth approximate of the CV error as a higher level loss function $H(.)$ (see discussion in Section 3.6).

The system of equations Eq. 9 can be reduced by firstly solving the optimality conditions of the SVM for fixed θ for each fold t up to sufficient accuracy (the last T lines of Eq. 9 are therefore eliminated). Hence we get w_t^* which is then used in the remainder of the equations. From the first T equations we can calculate the Lagrange multipliers λ_t using the inverse Hessian of the SVM's energy function:

$$\lambda_t = -\left(\frac{\partial^2 E}{\partial w_t^{*2}} \right)^{-1} \frac{\partial H}{\partial w_t^*}. \tag{10}$$

Consequently we obtain the main result:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \sum_{t=1}^{T} \left(\frac{\partial H(w_t^*, \Xi_t, \eta_t)}{\partial \theta} - \right. \\ \left. - \frac{\partial^2 E}{\partial w_t^* \partial \theta} \left(\frac{\partial^2 E}{\partial w_t^{*2}} \right)^{-1} \frac{\partial H}{\partial w_t^*} \right). \tag{11}$$

This gradient is used for optimizing the hyper-parameters. Observe that in case of the linear SVM (Eq. 4) the gradient $\frac{\partial \mathcal{L}}{\partial \theta}$ reduces to

$$\frac{\partial \mathcal{L}}{\partial c} = \sum_{t=1}^{T} - \frac{\partial^2 E}{\partial w_t^* \partial \theta} \left(\frac{\partial^2 E}{\partial w_t^{*2}} \right)^{-1} \frac{\partial H}{\partial w_t^*} \tag{12}$$

because the derivative of the higher level loss function $H(.)$ is zero w.r.t. c. So far, we have developed the bi-level solution for the linear SVM. In the following, we show that the concept can easily be extended for kernel SVMs.

3.3. Extension to kernel SVMs

First, we have to formulate the lower level problem - the energy function of the SVM - in terms of a kernel function $k(x, x_i)$. We use again a primal, unconstrained formulation of the SVM's energy (like in [6]). Instead of the weight vector w, we introduce a weight vector $\alpha \in \mathbb{R}^{N \times 1}$ with N the number of training examples.

$$\alpha^*(\theta) = \arg\min_{\alpha} \left\{ \frac{c}{2} \|f\|_2^2 + \sum_{j=1}^{N} \ell_\mu(f(x_j), y_j) \right\} \tag{13}$$

with

$$f(x) = \sum_{i=1}^{N} \alpha_i k(x, x_i) \qquad \text{and} \\ \|f\|_2^2 = \sum_{j=1}^{N} \sum_{i=1}^{N} \alpha_j \alpha_i k(x_j, x_i) = \alpha^T K \alpha. \tag{14}$$

The kernel matrix $K \in \mathbb{R}^{N \times N}$ is composed of matrix elements $k(x_j, x_i)$.

Rewriting Eq. 13 in matrix form using $k_j \in \mathbb{R}^{1 \times N}$ for describing a row of matrix K, we get

$$\alpha^*(\theta) = \arg\min_{\alpha} \left\{ \frac{c}{2} \alpha^T K \alpha + \sum_{j=1}^{N} \ell_\mu(k_j \alpha, y_j) \right\} = \\ \arg\min_{\alpha} E(\alpha, \theta, K, y). \tag{15}$$

For the non-linear case, the SVM's energy function used in the bi-level solution stated in Eq. 9, 10 and 11 is replaced by $E(\alpha, \theta, K, y)$. After the change of the weight vector w to α and of the data matrices X and Ξ to their corresponding kernels $K \in \mathbb{R}^{N \times N}$ and $\mathcal{K} \in \mathbb{R}^{L \times N}$, the former results are directly applicable.

In the case of a simple Gaussian kernel with bandwidth γ we have $\theta = (c, \gamma)^T$ and

$$\frac{\partial \mathcal{L}}{\partial \theta} = \begin{pmatrix} \frac{\partial \mathcal{L}}{\partial c} \\ \frac{\partial \mathcal{L}}{\partial \gamma} \end{pmatrix}. \tag{16}$$

The derivative $\dfrac{\partial H(\alpha(\theta)_t, \mathcal{K}_t, \eta_t)}{\partial \theta}$ (17)

from Eq. 11 is non-vanishing any more due to the dependence of the kernelized data to the kernel parameters.

3.4. Generalization to many kernel parameters

Assuming a Gaussian kernel having $d = 1, ..., D$ parameters, one for each feature dimension of input data, we can write down one element of the kernel matrix:

$$k(x_j, x_i) = \exp(-\sum_{d=1}^{D} \gamma_d (x_{jd} - x_{id})^2). \quad (18)$$

The gradient $\frac{\partial \mathcal{L}}{\partial \gamma}$ is extended to

$$\frac{\partial \mathcal{L}}{\partial \gamma} = \left(\frac{\partial \mathcal{L}}{\partial \gamma_1}, \frac{\partial \mathcal{L}}{\partial \gamma_2}, \ldots, \frac{\partial \mathcal{L}}{\partial \gamma_D} \right)^T \quad (19)$$

and the entries of the gradient vector are computed according to Eq. 11.

3.5. Multiple kernel bi-level SVM

We demonstrate in this section that the bi-level optimization scheme can directly be applied to determine parameters for a multiple kernel model [1, 21, 11]. There are different application scenarios for multiple kernel models: They can be used to combine different subsets of heterogeneous features or to combine different feature representations of the data.

We define the model as follows: Let $p = 1, 2, ..., P$ be the partitions (i.e. equivalent to the number of kernels used) each of which is of dimension D_p. A training example can be written as concatenation of P feature subsets $x_i = \{x_i^1, x_i^2, ..., x_i^P\}$ whereas $x_i^P \in \mathbb{R}^{D_p \times 1}$. A kernel element k_β of the new kernel matrix $K_\beta \in \mathbb{R}^{N \times N}$ is

$$k_\beta(x_j, x_i) = \sum_{p=1}^{P} \beta_p k_p(x_j^p, x_i^p). \quad (20)$$

With $k_{\beta j}$ being a row of the matrix K_β the SVM's energy function becomes

$$\alpha(\theta) = \arg\min_{\alpha} \left\{ \frac{c}{2} \alpha^T K_\beta \alpha + \sum_{j=1}^{N} \ell_\mu(k_{\beta j} \alpha, y_j) \right\}. \quad (21)$$

The vector of hyper-parameters θ now contains the γ_p for each sub-kernel and the weighting factors β_p:

$$\theta = (c, \gamma_1, \gamma_2, ..., \gamma_P, \beta_1, ..., \beta_P)^T. \quad (22)$$

Analogous to the previous derivations, we can write the gradient $\frac{\partial \mathcal{L}}{\partial \theta}$ as follows:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \begin{pmatrix} \frac{\partial \mathcal{L}}{\partial c} \\ \left(\frac{\partial \mathcal{L}}{\partial \gamma_p} \right)_{p=1}^{P} \\ \left(\frac{\partial \mathcal{L}}{\partial \beta_p} \right)_{p=1}^{P} \end{pmatrix}. \quad (23)$$

Our bi-level learning approach makes it possible to treat the kernel combination weights as hyper-parameters and also the parameters for the base kernels can be learnt. Next, we discuss the choice of the higher level loss function $H(.)$.

3.6. Higher level loss function

Due to the nature of our continuous optimization, we need a differentiable estimate of the generalization error. This is ideally a smoothed version of the actual hard classification rate e.g. described by the zero-one loss which assigns constant error to wrongly classified examples and zero error to correct examples.

In this paper we investigate three different higher level loss functions and compare them according to their meaningfulness for estimating the performance of the SVM. We use a smoothed version of the zero-one loss:

$$H(w, \Xi, \eta) = \frac{1}{\exp(\mu[\eta \circ (\Xi w^T)]) + 1} \quad (24)$$

with smoothing parameter $\mu = 12$. However, the zero-one loss is a non-convex function which might be a disadvantage for the optimization process.

The other functions we reviewed were the smoothed Hinge loss function

$$H(w, \Xi, \eta) = \sum_{i=1} \ell_\mu(w, b, \zeta_i, \eta_i) \quad (25)$$

as well as the mean squared error on the classification

$$H(w, \Xi, \eta) = \frac{1}{2L} \|\Xi w^T - \eta\|_2^2. \quad (26)$$

The MSE calculates the mean squared distance of the examples to the class labels (or, otherwise put, to the margins). Intuitively, the smoothed Hinge loss function should yield a better estimate of the hard classification error than the MSE because it assigns no error to correctly classified examples up to the margin and a linear increasing error for examples inside the margin and to wrong examples. Both MSE and Hinge loss are convex functions, and the MSE is particularly easy to differentiate.

On toy experiments, we found that the Hinge loss and the zero-one loss perform better on oddly shaped

datasets (imbalanced, with outliers) than the MSE (see Fig. 2). Using the MSE (green area) the bi-level SVM tends to learn a larger margin than using the Hinge loss (blue area), and the margins are pulled towards the barycenter of the data distribution. There was no difference in the behaviour between Hinge/zero-one loss in this case. However, in our experiments using real world data sets also the MSE performs quite well suggesting a good generalization capability.

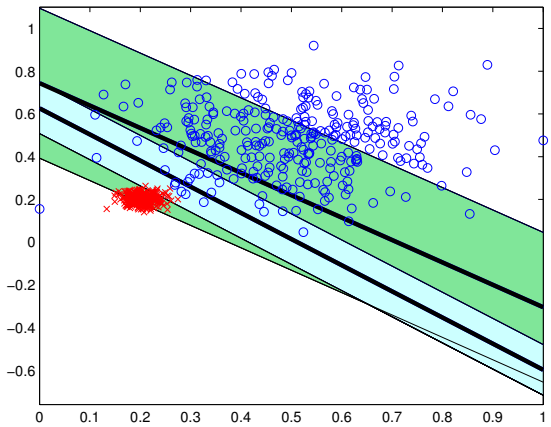

Figure 2. Margins and hyperplanes on an imbalanced toy data set for Hinge loss (blue) and MSE (green) as a higher level loss function.

4. Experimental Results

In our implementation we used the LBFGS-B optimization algorithm to solve the higher level optimization problem, see [5]. For solving the lower level problems (the SVM) we used FISTA [2]. For the experiments, we used several data sets from the UCI machine learning repository [1] (diabetes, ionosphere, heart, seeds, parkinson). The aim of the experiments is to show how the classification results using the hyper-parameters determined via the bi-level optimization scheme compare to the results of the traditional grid search procedure. In particular, we focus on evaluating the effectiveness of the higher level loss function approximations. Furthermore, we show results for two settings using an increased number of hyper-parameters as well as results for an image classification experiment.

The smoothing parameter μ from Eq. 6 and 25 was chosen as big as possible as long as the outer level optimization does not fail (due to the Hessian becoming ill-conditioned when it is very sparse). The initial values θ for the bi-level optimization were set randomly due to the fact that their choice is not critical: Usually the bi-level program converges to the same θ^* for different initial values given sufficient accuracy of the solution of w^*.

[1] http://archive.ics.uci.edu/ml

4.1. Illustrative examples

First, we have a look at how the hard classification rates vary in the hyper-parameters and how the higher level loss functions we mentioned earlier "fit" to the achieved classification performance. For this reason, we show two examples. First, results using a linear bi-level SVM on the diabetes data set are shown in Fig. 3. On the y axis the CV error rate and the test error rate are shown as well as the higher level loss function values. The MSE is plotted in dashed blue, the approximated Hinge loss in solid red and the smooth zero-one loss in solid green. The errors are plotted over the regularization parameter c and have been determined via grid search. We point out that the error values are not directly comparable hence we rescale them for better comparison.

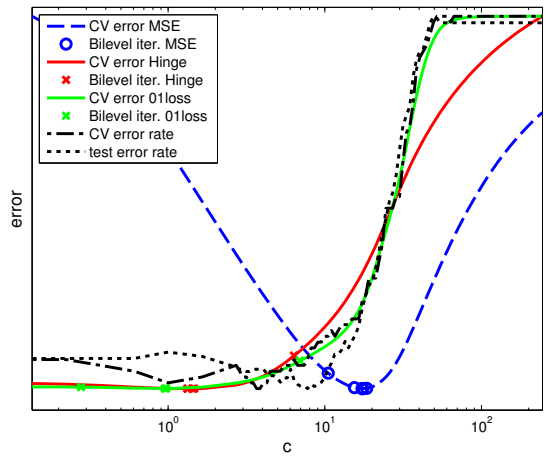

Figure 3. Comparing hard classification rates and the corresponding higher level loss function values over c using a linear SVM on the diabetes data set.

We observe that the minima of the CV and test classification error rates do not coincide exactly but the magnitudes are consistent. The smoothed Hinge loss seems to model the actual CV classification rates quite well, and the zero-one loss approximation fits even better (as expected). Their minima lie in the area of lowest classification error rates. The MSE does not correspond to the error rates, but still has the minimum in a reasonable area.

The second example shown in Fig. 4 illustrates the dependency of the kernel parameter γ of a simple RBF kernel SVM for a fixed c on the same (diabetes) data set. Here, the CV and test error rates have

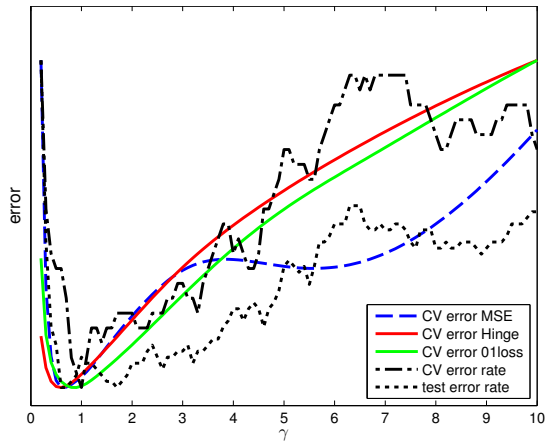

Figure 4. Comparing hard classification rates and the corresponding higher level loss function values with fixed c over γ using a kernel SVM on the diabetes data set.

again similar minima, and in this case, also all three flavours of higher level loss functions share approximately the same minima. For the MSE, we observe and additional local minimum at $\gamma \approx 5.5$ which is an unwanted property for optimization. At this point, no clear answer can be given which of the higher level loss functions is the best.

4.2. Classification rates for different settings

In Tab. 1 we summarize the CV and test error rates obtained by the linear and kernel bi-level SVM as well as the respective rates obtained using grid search on a comparable computational budget measured in trials. Each trial consists of the evaluation of the SVM for T folds for one set of hyper-parameters. The number of folds in our experiments was chosen with $T = 5$. Surprisingly, often the MSE yields good test classification rates, sometimes even best values, even though the CV error does not usually yield lowest rates. Here, often Hinge loss and sometimes zero-one loss lead to better results. In terms of number of trials used for optimization, the zero-one and Hinge loss are the best. Given the low computational budget assigned for grid search, only for one data set better rates were achieved (seeds), even though it is quite possible that grid search can outperform the bi-level approach using more trials in the linear/simple kernel case because the exact classification rates are taken to decide which set of hyper-parameters is best. However, as we will see in the following experiments, the classification rates can be significantly improved by using a more complex kernel for which it will be difficult to achieve a good result using exhaustive grid search.

Data set	Type	CV Err.	Test Err.	Trials
Diabetes	Lin01loss	**24.13**	20.33	10
	LinHinge	**24.13**	20.66	**7**
	LinMSE	25.87	**19.67**	**8**
	LinGrid	*24.34*	*20.66*	*15*
	Ker01loss	22.17	18.03	**8**
	KerHinge	21.30	**17.70**	26
	KerMSE	**18.70**	20.98	11
	KerGrid	*25.00*	*19.02*	*50*
Ionosph.	Lin01loss	12.86	8.57	7
	LinHinge	**11.43**	**7.86**	**5**
	LinMSE	16.19	**7.86**	8
	LinGrid	*14.29*	*8.57*	*15*
	Ker01loss	**0.95**	**2.85**	**17**
	KerHinge	2.86	3.57	32
	KerMSE	3.81	**2.86**	33
	KerGrid	*13.81*	*4.29*	*50*
Heart	Lin01loss	15.79	15	8
	LinHinge	**14.21**	**13.75**	**6**
	LinMSE	17.37	15	8
	LinGrid	*18.95*	***13.75***	*15*
	Ker01loss	15.26	15	**15**
	KerHinge	**14.74**	**13.75**	16
	KerMSE	17.89	**13.75**	34
	KerGrid	*18.95*	*15*	*50*
Seeds	Lin01loss	**6**	10	**5**
	LinHinge	**6**	10	6
	LinMSE	7.33	11.67	11
	LinGrid	*9.33*	***9.33***	*15*
	Ker01loss	2	8.33	**14**
	KerHinge	**1.33**	8.33	15
	KerMSE	7.33	**6.67**	23
	KerGrid	*10.67*	*10*	*50*

Table 1. Summary of classification rates on several datasets comparing the CV and test errors and the number of trials used. Results are reported for the linear ('lin') and kernel ('ker') SVM using the MSE, Hinge or zero-one ('01loss') higher level loss functions.

4.3. Learning multiple parameters

Learning one parameter γ_d per feature dimension. For this experiment, the seeds data set was used ($D = 8$). The results are summarized in Tab. 2.

Learning parameters γ_p and β_p for a multiple kernel SVM. For this experiment the parkinson data set was used [16]. The data contains 21 measurements of different orders of magnitude. Using the multiple kernel SVM we are able to combine the features into P groups of similar magnitude, and set the parameters γ_p and β_p via the bi-level optimization procedure. The results are summarized in Tab. 3. We achieve good results using no pre-processing and no filtering of correlated features compared to the original paper [16] where they report a test classification rate of $8.2\% \pm 2$. We observe an exceptionally low

number of necessary trials using the zero-one loss for both experiments, and very good test classification rates for Hinge and zero-one loss.

Data set	Type	CV Err.	Test Err.	Trials
Seeds	MSE	0.67	**3.33**	62
	Hinge	**0**	**3.33**	119
	01loss	2.67	**3.33**	**59**

Table 2. Results using a bi-level kernel SVM with γ_D parameters.

Data set	Type	CV Err.	Test Err.	Trials
Parkinson	MSE	**0**	9.09	53
	Hinge	8.75	**7.27**	80
	01loss	2.04	**7.27**	**30**

Table 3. Results using a bi-level multiple kernel SVM.

4.4. Image classification

The following image classification experiment was conducted on the Graz02 data set [18]. For feature extraction the VLFeat Library [2] was used. The data was pre-processed according to a bag of visual words model using PHOW features, a variant of SIFT features extracted at several scales [17]. Moreover, for this task we use exponential χ^2 kernels because they show naturally better performance on histogram data compared to RBF kernels [24].

Confusion Matrix, mAcc = 70.00%

	bike	cars	none	person
bike	96.67	0.00	0.00	3.33
cars	0.00	93.33	0.00	6.67
none	36.67	36.67	3.33	23.33
person	3.33	3.33	6.67	86.67

Figure 5. Resulting confusion matrix using a bi-level kernel SVM and the Hinge loss as a higher level loss function.

Confusion Matrix, mAcc = 73.33%

	bike	cars	none	person
bike	93.33	0.00	0.00	6.67
cars	3.33	86.67	3.33	6.67
none	36.67	20.00	26.67	16.67
person	3.33	3.33	6.67	86.67

Figure 6. Resulting confusion matrix using a bi-level kernel SVM and the MSE as a higher level loss function.

In Fig. 5 and Fig. 6 we compare the classification results for each of the four classes in the Graz02 data set, namely bike, cars, person and none (the background class). For training we used 60 images per class, and 30 for testing. Overall the accuracy using the MSE is better, but if we do not regard the background class the results using the Hinge loss are superior. By construction, the data for learning 1 vs. rest classifiers is imbalanced due to the low number of positive examples. That might explain why MSE performs worse than Hinge loss in the image classification example. The results via the kernel bi-level SVM were obtained using a mean of 9 trials per each 1 vs. rest classifier that was trained using Hinge loss and 8 trials using the MSE. The results of grid search and evaluating the CV error rate to determine the best hyper-parameters using 50 trials are shown in Fig. 7. We obtain a baseline of classification results on this data set for the relevant classes bike, cars and person.

Confusion Matrix, mAcc = 73.33%

	bike	cars	none	person
bike	93.33	0.00	0.00	6.67
cars	0.00	86.67	6.67	6.67
none	36.67	20.00	30.00	13.33
person	6.67	3.33	6.67	83.33

Figure 7. Resulting confusion matrix using a kernel SVM and grid search using 50 trials.

5. Conclusion

In this paper, we presented a novel bi-level optimization scheme that is able to perform continuous hyper-parameter optimization for linear and kernel SVMs based on different smoothed estimates of the CV error rate. Very good test classification rates are obtained using only a tiny fraction of trials that would be necessary to perform exhaustive grid search which makes the method very practical. High potential lies in the optimization of several kernel parameters: The classification rates are better than using only a simple kernel and optimizing the parameters is easy using the bi-level optimization approach. In the case of optimizing one or two parameters only, a very fine grid search might lead to better results than the bi-level approach because the exact classification errors are minimized, but at a much higher computational cost.

Acknowledgements

The authors acknowledge support from the Austrian Science Fund (FWF) under the START project BIVISION, No. Y729.

[2] http://www.vlfeat.org/

References

[1] F. R. Bach, G. R. G. Lankriet, and M. I. Jordan. Multiple Kernel Learning, Conic Duality, and the SMO Algorithm. *International Conference on Machine Learning*, 2004. 1, 5

[2] A. Beck and M. Teboulle. A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, Jan. 2009. 6

[3] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13:281–305, 2012. 1, 2

[4] D. P. Bertsekas. *Nonlinear programming*. Athena Scientific, 1999. 4

[5] R. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995. 6

[6] O. Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19(5):1155–78, May 2007. 1, 3, 4

[7] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, pages 131–159, 2002. 1

[8] B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. *Annals of Operations Research*, 153(1):235–256, Apr. 2007. 3

[9] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sept. 1995. 1

[10] K. Duan, S. Keerthi, and A. N. Poo. Evaluation of simple performance measures for tuning SVM hyperparameters. *Neurocomputing*, 51:41–59, Apr. 2003. 2

[11] M. Gönen and E. Alpaydn. Multiple kernel learning algorithms. *The Journal of Machine Learning Research*, 12:2211–2268, 2011. 1, 5

[12] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006. 1

[13] F. Hutter, H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. *Learning and Intelligent Optimization*, 2011. 2

[14] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, 25:1097–1105, 2012. 1

[15] G. Kunapuli and K. Bennett. Classification model selection via bilevel programming. *Optimization Methods & Software*, 23(4):475–489, 2008. 1

[16] M. a. Little, P. E. McSharry, E. J. Hunter, J. Spielman, and L. O. Ramig. Suitability of dysphonia measurements for telemonitoring of Parkinson's disease. *IEEE Transactions on Bio-medical Engineering*, 56(4):1015–1022, Apr. 2009. 7

[17] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov. 2004. 8

[18] A. Opelt and A. Pinz. Generic object recognition with boosting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3):416–431, 2006. 8

[19] B. Schölkopf and A. J. Smola. *Learning with kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2002. 1

[20] J. Snoek, H. Larochelle, and R. Adams. Practical Bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, pages 1–9, 2012. 2

[21] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006. 1, 5

[22] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):480–92, Mar. 2012. 1

[23] J. Zhang, R. Jin, Y. Yang, and A. Hauptmann. Modified logistic regression: An approximation to SVM and its applications in large-scale text categorization. *International Conference on Machine Learning*, pages 888–895, 2003. 3

[24] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study. *International Journal of Computer Vision*, 73(2):213–238, Sept. 2006. 8