Fast model selection by limiting SVM training times

Aydın Demircioğlu[1], Daniel Horn[2], Tobias Glasmachers[1], Bernd Bischl[3], Claus Weihs[2] *

1- Ruhr-Universität Bochum - Institut für Neuroinformatik
44780 Bochum, Germany

2- Technische Universität Dortmund - Fakultät Statistik
44221 Dortmund, Germany

3- Ludwig-Maximilians-Universität München - Institut für Statistik
80539 München, Germany

Abstract. Kernelized Support Vector Machines (SVMs) are among the best performing supervised learning methods. But for optimal predictive performance, time-consuming parameter tuning is crucial, which impedes application. To tackle this problem, the classic model selection procedure based on grid-search and cross-validation was refined, e.g. by data subsampling and direct search heuristics. Here we focus on a different aspect, the stopping criterion for SVM training. We show that by limiting the training time given to the SVM solver during parameter tuning we can reduce model selection times by an order of magnitude.

1 Introduction

One of the standard classifiers for solving machine learning problems are kernelized Support Vector Machines (SVMs). While they yield excellent performance, they suffer from two problems: On the one hand, solving the underlying optimization problem to a fixed accuracy has at least quadratic complexity in the number of training points. On the other hand, both, the regularization parameter as well as the kernel parameters need heavy, problem-specific tuning.

The first problem is addressed by a multitude of approximate solvers. However, model selection is still in its infancy, with grid search being the most widely applied method. Though it works in practice, it is computationally demanding and rather wasteful. Furthermore, it is not a priori clear how to choose the grid.

Simplistic grid search is prone to explore large low-quality regions of the parameter space since it does not make use of already evaluated points to guide the search. Nested grid search improves in this aspect, but even pure random search scales better to high-dimensional search spaces [1]. Direct search methods like the simplex downhill algorithm and evolutionary optimization techniques are applicable [2], but they are known to require a potentially large number of parameter evaluations.

A simple and widespread technique to speed up parameter tuning is to subsample the data. While this technique is effective in terms of computation time,

*We acknowledge support by the Mercator Research Center Ruhr, under grant Pr-2013-0015 *Support-Vektor-Maschinen für extrem große Datenmengen*

it introduces a systematic bias since parameters are tuned for a much less well sampled variant of the learning problem.

In this paper we focus on the key idea that relatively inaccurate models are sufficient for identifying the well-performing parameter regime. This introduces yet another lever that is orthogonal to both subsampling and the search strategy in the sense that it can be combined freely with existing techniques. This lever is the stopping criterion of the SVM solver. Our approach is based on the observation that practically all SVM training procedures in common use are based on iterative optimization techniques with relatively low iteration cost. Hence these solvers can easily be turned into anytime algorithms, using (processor or wall clock) time as a stopping criterion.

Our main goal is to establish an efficient alternative to the standard proceeding. To this end we propose to combine time-limited training with the Efficient Global Optimization (EGO) search method [3]. We explore this approach for six different types of SVM solvers on a variety of large scale benchmark data sets. Our results show huge and in some cases surprising differences between solvers. They indicate that LASVM is particularly suitable for fast model selection. We show that the time for model selection with our method is faster than grid search by an order of a magnitude, while loosing only little accuracy.

Independent from our research, the effect of stopping stochastic gradient methods early was analyzed in [4] on a theoretical level. While our approach here can be deemed as an empirical verification of their theory, our experiments show that this insight also holds true for other training methods.

2 Kernelized Support Vector Machines

Kernelized Support Vector Machines (SVM) [5] are binary classifiers that use a kernel k to allow for non-linear classification decisions $x \mapsto \operatorname{sign}(\langle w, \phi(x)\rangle + b)$, where ϕ is the (non-linear) feature map corresponding to the kernel $k(x, x') = \langle \phi(x), \phi(x')\rangle$ in the reproducing kernel Hilbert space $\mathcal{H}$. We use the RBF kernel $k(x, x') = e^{-\gamma ||x-x'||^2}$, since it yields excellent performance and enjoys a universal approximation property. The SVM training problem is given by

$$\min_{w\in\mathcal{H}, b\in\mathbb{R}} \quad \frac{1}{2}||w||^2 + C \cdot \sum_{i=1}^{n} \max\Big(0, 1 - y_i\big(\langle w, \varphi(x_i)\rangle_{\mathcal{H}} + b\big)\Big), \tag{1}$$

where $\big\{(x_1, y_1), \dots, (x_n, y_n)\big\}$ are the labeled training points and $C > 0$ is a regularization parameter that controls the complexity of the predictive model.

There are several methods to find a solution to (1). The gold standard is Sequential Minimal Optimization (SMO), a decomposition method solving (1) in its dual form [6, 7]. In [8] a reordering of the SMO steps is proposed that allows for online learning. Alternatively, the primal problem can be optimized directly, e.g., BSGD applies stochastic gradient descent and introduces a budget to control the (computational) complexity of the solution. BVM and CVM solve a modified version of (1) with squared hinge loss. Both methods map the training problem to the geometric problem of finding a minimal enclosing ball.

SVM Solver	Method	URL
LIBSVM	SMO	`http://www.csie.ntu.edu.tw/~cjlin/libsvm/`
BGSD	Stochastic Gradient	`http://www.dabi.temple.edu/budgetedsvm/`
LASVM	Online SMO	`http://leon.bottou.org/projects/lasvm`
BVM/CVM	Enclosing Ball	`http://www.c2i.ntu.edu.sg/ivor/cvm.html`
SVMperf	Cutting Planes	`http://svmlight.joachims.org/svm_perf.html`

Table 1: Overview of the applied SVM solvers. All solvers are implemented in C/C++, all of them can take advantage of sparsely represented data.

3 SVM Model Selection Methods

Grid Search: One of the basic and most often used methods to find a good model is to solve the SVM problem on a discretized grid in the (C, γ) parameter space. The combination (C, γ) with the best cross-validation performance is eventually selected as the final model. Though straight-forward to use, grid search depends on the discretization (extent and resolution of the grid), and it is not clear how to choose it. A too small or too coarse grid might miss a good model, while a too wide or too fine grid is wasteful.
Efficient Global Optimization (EGO): EGO [3] is a sequential, model-based (Bayesian) optimization method. It models the error landscape with a surrogate regression model, trained with all previous parameter evaluations, and optimizes this model. A Kriging (Gaussian process) model is a standard choice for the surrogate. EGO is a global optimizer. It avoids getting stuck in local minima by optimizing the expected improvement (EI) instead of the surrogate model's mean response. It was applied to SVM model selection in [9].

4 Model Selection with Time-limited SVM Training

The ultimate goal of any model selection method is to find a good model, here, parameters (C, γ) resulting in low generalization error. To reach this goal we must be able to compare candidate parameters, e.g., with a (cross-) validation error measure. This requires training the learning machines for each parameter setting. When stopping an SVM solver early then we generally expect the error to be higher. However, if stopping early results mainly in a constant shift of the error then the error values of different parameter vectors are still in the correct order, and the best parameters can be identified. However, it can be expected that non-constant systematic biases exist, and that the noise increases. The question arises whether good and bad models can be identified early on during training or not. Our hypothesis is that this is indeed the case.

Therefore we propose to augment each SVM solver with an additional time-based stopping criterion. This introduces an additional parameter into the model selection process, the stopping time T controlling the approximation quality. It needs to be set with a heuristic.

5 Experimental Setup

We aim to answer two questions with our experiments:

1. *Is our model selection procedure (combining EGO with time-limited training) superior to the standard proceeding based on grid search?*

2. *Which SVM solver is best suited for time-limited model selection?*

Our method is targeted at large data sets, where model selection becomes a computational bottleneck. Ideally, we would compare our approach to an exhaustive grid search. However, this is computationally infeasible. Therefore we use the results of a previous experiment [10] based on ParEGO, a multi-criteria generalization of EGO, as a baseline method. ParEGO optimizes contradicting goals, here accuracy and training time, and is known to yield a very good approximation to the true Pareto front. As such it is a much stronger baseline than grid search in terms of training time, while yielding competitive accuracy.

All experiments were conducted on a 16-core CPU with 64 GB of RAM.[1] We consider the following data sets: arthrosis, aXa, cod-rna, covtype, ijcnn1, mnist, poker, protein, shuttle, spektren, vehicle and wXa. All data sets have been split randomly into training, validation, and test sets with a ratio of 2:1:1.

A user-adjustable time constraint[2] was added to the original software packages, see Table 1. We left all tunable parameters of the solvers at their defaults,[3] except for the budget of BSGD, which we fixed to 2048. For EGO we used the implementation in mlrMBO.[4] To speed up EGO, we ran a parallel version, in which we used the lower confidence bound criterion (LCB) instead of the usual expected improvement. In order to propose multiple points from the model for parallel evaluation, we sampled multiple values of the λ parameter of the LCB criterion and optimized each λ-LCB-function independently. Further details concerning EGO parallelization can be found in [11]. Similar to [10], C and γ were constrained to the range $[2^{-15}, 2^{15}]$, which is frequently used for a thorough grid search. The initial design for each run was chosen to consist of 20 points, and 10 sequential iterations with 20 points proposed in parallel were performed.

We fixed the training time-limit to $T = 2^{\log_{10}(n)+1}$ seconds, which we have chosen heuristically. For each point proposed by EGO a solver was run for T seconds on the training set and afterwards the resulting model was evaluated on the corresponding validation set. The final model was trained with the best parameters found with a time-limit of 8 hours and evaluated on the test set.

Statistical significance was tested with a Friedman test [12], where the null hypothesis is that there is no difference between the classifiers.

[1]This is a different hardware setup than the high performance cluster used in the ParEGO experiments. This is compliance with our goal to show that an extensive model search can be mimicked on a commodity workstation. Furthermore, the difference in speed per core of both setups can be expected to be a small factor and hence does not affect our main result.

[2]We make all our modified software packages, the experiment with all details and all results publicly available at `https://www.github.com/aydindemircioglu/TLEGO`.

[3]Refer to our repository and also to [10] for more details.

[4]`https://github.com/berndbischl/mlrMBO`

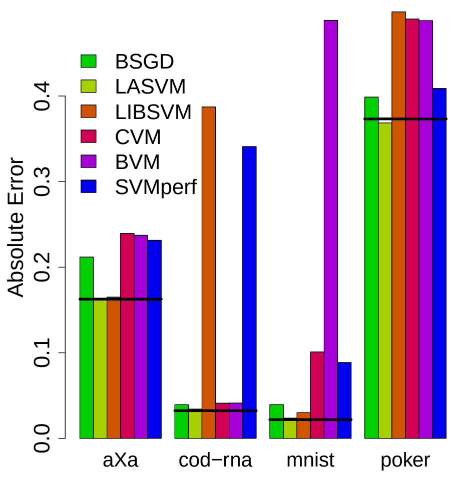

Fig. 1: Absolute Error rates (see text)

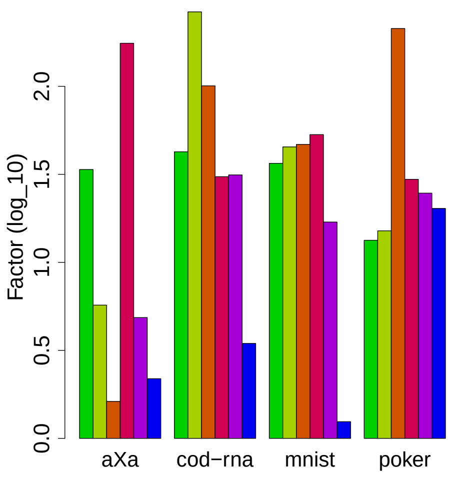

Fig. 2: Relative Timings (see text)

6 Results and Discussion

Figure 1 shows the error rates (lower is better) achieved by the final model when performing time limited model selection, for all six solvers. The black bars indicate the performance of the baseline experiment.[5] Figure 2 displays the corresponding runtime: the sum of training and validation times during model selection and the training time of the final model. The values are the order of magnitude (base-10 logarithm) of the factor by which our method is faster than the baseline (higher is better).

We start with our second question. From Figure 1 it is apparent that apart from LASVM the comparability of models suffers from time-limited training on at least one data set. None of the other solvers performs well consistently. Indeed, the Friedman test indicates a highly significant difference between the solvers ($p < 10^{-7}$). A post-hoc test using Holm's method [12] comparing ParEGO to all other solvers, shows that there is a significant difference between ParEGO and LASVM in one group, and all other solvers. Therefore, our method only applies to LASVM. Naively we would have expected LIBSVM to perform very similar to LASVM as both solve the dual problem with SMO, but this seems not to be true, as the results are not decisive enough. We suspect that LIBSVM's models are only comparable in a late convergence phase. LASVM's focus on online optimization helps to produce better (comparable) models from the very beginning.

Focusing only on LASVM, Figure 2 shows that our method yields a major speed up over ParEGO, and therefore over grid search, of roughly one to two orders of magnitudes. This gives a strong affirmative answer to our first question.

[5] Due to space constraints the figure shows only representative results for four data sets. All results are available at `http://largescalesvm.de/tlego/`.

7 Conclusions

We show that a simple modification of the stopping criterion by limiting the time spent on training together with the EGO search strategy can be used to speed up model selection without loosing accuracy significantly by more than an order of magnitude with respect to ParEGO and thus to grid search. This result depends heavily on the SVM solver, and holds true only for LASVM. On a theoretical level much needs to be done. There is no explanation yet for the different behavior of the SVM solvers. Also it is unclear how to choose the time limit. We have not explored a more dynamical method, e.g., making the time spent on training dependent on the relative performance and using a more refined search once the relevant region is determined. Finally, it is interesting to analyze how our stopping criterion interacts with subsampling, which is an alternative to speeding up model selection, and to apply this principle to other, more sophisticated model search methods.

References

[1] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, 2012.

[2] T. Glasmachers and C. Igel. Uncertainty handling in model selection for support vector machines. In G. Rudolph, T. Jansen, S. Lucas, C. Poloni, and N. Beume, editors, *Parallel Problem Solving from Nature (PPSN)*, pages 185–194. Springer, 2008.

[3] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[4] M. Hardt, B. Recht, and Y. Singer. Train faster, generalize better: Stability of stochastic gradient descent. *ArXiv e-prints*, September 2015.

[5] Corinna Cortes and Vladimir Vapnik. Support vector machine. *Machine learning*, 20(3):273–297, 1995.

[6] John Platt. Fast Training of Support Vector Machines using Sequential Minimal Optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 12, pages 185–208. MIT Press, 1998.

[7] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[8] Antoine Bordes, Seyda Ertekin, Jason Weston, and Léon Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6:1579–1619, 2005.

[9] Patrick Koch, Bernd Bischl, Oliver Flasch, Thomas Bartz-Beielstein, Claus Weihs, and Wolfgang Konen. Tuning and evolution of support vector kernels. *Evolutionary Intelligence*, 5(3):153–170, 2012.

[10] Daniel Horn, Aydın Demircioglu, Bernd Bischl, Tobias Glasmachers, and Claus Weihs. B: A comparative study on kernelized support vector machines. 2014.

[11] B. Bischl, S. Wessing, N. Bauer, K. Friedrichs, and C. Weihs. MOI-MBO: Multiobjective infill for parallel model-based optimization. In *Learning and Intelligent Optimization*, Lecture Notes in Computer Science, pages 173–186. Springer, 2014.

[12] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.