Published in: Proceedings of the
12th European Symposium on Artificial Neural Networks (ESANN 2004),
Evere, Belgium: d-side publications, 2004

Evolutionary Tuning of Multiple SVM Parameters

Frauke Friedrichs, Christian Igel*

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

Abstract. We consider the problem of choosing multiple hyperparameters for support vector machines. We present a novel, general approach using an evolution strategy (ES) to determine the kernel from a parameterized kernel space and to control the regularization. We demonstrate on benchmark datasets that the ES improves the results achieved by grid search and can handle much more kernel parameters. In particular, we optimize generalized Gaussian kernels with arbitrary scaling and rotation.

1 Introduction

Support vector machines (SVMs, e.g., [3]) are learning machines based on two key elements: a general purpose linear learning algorithm and a problem specific kernel that computes the inner product of input data points in a feature space. The choice of the kernel function is the crucial step in handling a learning task with an SVM. For example, it is important to achieve a distribution of the data in the feature space that reflects the affiliation to the class labels. Often a parameterized family of kernel functions is considered and the problem is to find an appropriate parameter vector for the given problem. In case of non-separable data one also has to choose a regularization parameter, which controls the trade-off between minimizing the training error and the complexity of the decision function. The kernel parameters together with the regularization parameter are called the hyperparameters of the SVM.

In practice the hyperparameters are usually determined by grid search. That is, the hyperparameters are varied with a fixed step-size through a wide range of values and the performance of every combination is assessed using some performance measure. Because of the computational complexity, grid search is only suitable for the adjustment of very few parameters. Perhaps the most elaborate systematic technique for choosing multiple hyperparameters are gradient descent methods [2, 4]. These algorithms iterate the following procedure: The SVM is trained using the current hyperparameter vector, the gradient of

*Christian.Igel@neuroinformatik.rub.de

some generalization error bound w.r.t. the hyperparameters is calculated, and a step is performed in the parameter space based on this gradient. However, this approach has some significant drawbacks. The kernel function has to be differentiable, which excludes for example string kernels. The score function for assessing the performance of the hyperparameters (or at least an accurate approximation of this function) also has to be differentiable with respect to kernel and regularization parameters, which excludes reasonable measures such as the number of support vectors. In [2, Sec. 6.2] separability of the dataset is assumed when computing the derivative, which is a very restrictive assumption. Finally, iterative gradient-based algorithms, which usually rely on smoothed approximations of a score function, do not ensure that the search direction is exactly the gradient of the original, often discontinuous generalization performance measure.

We propose a method for hyperparameter selection that does not suffer from the limitations described above, namely using the covariance matrix adaptation evolution strategy (CMA-ES, [5]) to search for an appropriate hyperparameter vector. The fitness function that is optimized directly corresponds to some generalization performance measure. We apply our method to tuning Gaussian kernels, where not only the scaling but also the orientation is adapted.

We give a short description of SVMs in Section 2 and of the CMA-ES in Section 3. The parameterization of general Gaussian kernels is introduced in Section 4. In Section 5 we present some experimental results.

2 Support Vector Machines

We consider L_1-norm soft margin SVMs for the discrimination of two classes. Let $(\boldsymbol{x}_i, y_i), 1 \leq i \leq \ell$, be the training examples, where $y_i \in \{-1, 1\}$ is the label associated with input pattern $\boldsymbol{x}_i \in X$. The main idea of SVMs is to map the input vectors to a feature space F and to classify the transformed data by a linear function. The transformation $\phi : X \to F$ is implicitly done by a kernel $K : X \times X \to \mathbb{R}$, which computes an inner product in the feature space, i.e., $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$. The linear function for classification in the feature space is chosen according to a generalization error bound considering a target margin and the margin slack vector, i.e., the amounts by which individual training patterns fail to meet that margin (cf. [3]). This leads to the SVM decision function $f(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{i=1}^{\ell} y_i \alpha_i^* K(\boldsymbol{x}_i, \boldsymbol{x}) + b\right)$, where the coefficients α_i^* are the solution of the following quadratic optimization problem: Maximize $W(\boldsymbol{\alpha}) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ subject to $\sum_{i=1}^{\ell} \alpha_i y_i = 0$ and $0 \leq \alpha_i \leq C$ for $i = 1, \ldots, \ell$. The optimal value for b can then be computed based on the solution $\boldsymbol{\alpha}^*$. The vectors $\boldsymbol{x}_i$ with $\alpha_i > 0$ are called support vectors. The regularization parameter C controls the trade-off between maximizing the target margin and minimizing the L_1-norm of the margin slack vector of the training data.

3 Evolution Strategies

Evolution strategies (ES, [1]) are one of the main branches of evolutionary algorithms, i.e., a class of iterative, direct, randomized optimization methods mimicking principles of neo-Darwinian evolution theory. We use the highly efficient CMA-ES [5]. A set of μ individuals that form the parent population is maintained. Each individual has a genotype that encodes a candidate solution for the optimization problem at hand, here an m-dimensional real-valued object variable vector. The fitness of an individual is equal to the objective function value at the point in the search space it represents. In each iteration of the algorithm, $\lambda > \mu$ new individuals, the offspring, are generated by partially stochastic variations of parent individuals. The fitness of the offspring is computed and the μ best of the offspring form the next parent population. This loop of variation and selection is repeated until a termination criterion is met.

The object variables are altered by global intermediate recombination and Gaussian mutation. That is, the genotypes $\boldsymbol{g}_k^{(t)}$ of the offspring $k = 1, \ldots, \lambda$ created in iteration t are given by $\boldsymbol{g}_k^{(t)} = \langle \tilde{\boldsymbol{g}} \rangle^{(t)} + \boldsymbol{\xi}_k^{(t)}$, where $\langle \tilde{\boldsymbol{g}} \rangle^{(t)}$ is the center of mass of the parent population in iteration t and the $\boldsymbol{\xi}_k^{(t)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{C}^{(t)})$ are independent realizations of an m-dimensional normally distributed random vector with zero mean and covariance matrix $\boldsymbol{C}^{(t)}$. The matrix $\boldsymbol{C}^{(t)}$ is updated online using the covariance matrix adaptation method (CMA). The key idea of the CMA is to alter the mutation distribution in a deterministic way such that the probability to reproduce steps in the search space that led to the actual population—i.e., produced offspring that were selected—is increased. The search path of the population over the past generations is taken into account, where the influence of previous steps decays exponentially. The CMA does not only adjust the mutation strengths in m directions, but also detects correlations between object variables. Thereby, it becomes invariant under orthogonal transformations of the search space (apart from the initialization).

4 Encoding General Gaussian Kernels

We consider general Gaussian kernels $K_{\boldsymbol{A}}(\boldsymbol{x}, \boldsymbol{z}) = e^{-(\boldsymbol{x}-\boldsymbol{z})^T \boldsymbol{A}(\boldsymbol{x}-\boldsymbol{z})}$, where $\boldsymbol{A}$ is a symmetric positive definite matrix. We use three different parameterizations of these kernels. Firstly, there are the ordinary Gaussian kernels $K_{\tau \boldsymbol{I}}$, where $\boldsymbol{I}$ is the unit matrix and $\tau > 0$ the only adjustable parameter. Secondly, we allow independent scalings of the components of the input vectors and consider the kernel family $K_{\boldsymbol{D}}$, where $\boldsymbol{D}$ is a diagonal matrix with positive entries. Finally, we allow arbitrary symmetric positive definite matrices $\boldsymbol{A}$, i.e., the input space can be scaled and rotated. This kernel family is denoted simply by $K_{\boldsymbol{A}}$.

The individuals in the ES encode C and the kernel parameters, and the variation operators obey restrictions of parameters to $\mathbb{R}^+$ when necessary. However, when encoding $K_{\boldsymbol{A}}$ kernels, we have to ensure that after variation the genotype still corresponds to a feasible (i.e., symmetric, positive definite) matrix. We use the fact that for any symmetric and positive definite $n \times n$ matrix $\boldsymbol{A}$ there exists an orthogonal $n \times n$ matrix $\boldsymbol{T}$ and a diagonal $n \times n$ matrix $\boldsymbol{D}$ with positive entries such that $\boldsymbol{A} = \boldsymbol{T}^T \boldsymbol{D} \boldsymbol{T}$ and

$$\boldsymbol{T} = \prod_{i=1}^{n-1} \prod_{j=i+1}^{n} \boldsymbol{R}(\alpha_{i,j}) \ ,$$

see [9]. The $n \times n$ matrices $\boldsymbol{R}(\alpha_{i,j})$ are elementary rotation matrices which are equal to the unit matrix except for $[\boldsymbol{R}(\alpha_{i,j})]_{ii} = [\boldsymbol{R}(\alpha_{i,j})]_{jj} = \cos \alpha_{ij}$ and $[\boldsymbol{R}(\alpha_{i,j})]_{ji} = -[\boldsymbol{R}(\alpha_{i,j})]_{ij} = \sin \alpha_{ij}$. Each genotype encodes the $n + (n^2 - n)/2$ parameters $(d_1, \dots, d_n, \alpha_{1,2}, \alpha_{1,3}, \dots, \alpha_{1,n}, \alpha_{2,3}, \alpha_{2,4}, \dots, \alpha_{2,n}, \dots, \alpha_{n-1,n})$, and we set $\boldsymbol{D} = \mathrm{diag}(|d_1|, \dots, |d_n|)$.

5 Experimental Evaluation

For the evaluation of our hyperparameter optimization method we used the common medical benchmark datasets *Breast-Cancer*, *Diabetes*, *Heart*, and *Thyroid* with input dimension n equal to 9, 8, 13, and 5, respectively, preprocessed and partitioned as in [8]. Each component of the input data is normalized to zero mean and unit standard deviation. There are 100 partitions of each dataset into disjoint training and test sets. In [7], appropriate SVM hyperparameters for the Gaussian kernel $K_{\tau \boldsymbol{I}}$ were determined using a two-stage grid search and the following score function: For each hyperparameter combination, five SVMs are built using the training sets of the first five data partitions and the average of the classification rates on the corresponding five test sets determines the score value (Test-5) of this parameter vector. The hyperparameter vector with the best score is selected and its performance is measured by calculating the score function using all 100 partitions (Test-100). We used this scenario as a basis for our evolutionary hyperparameter adaptation approach: We took the results of the grid search performed in [7] as initial values for the CMA-ES and used the score function described above to determine the fitness. The SVMs were trained using SVM^{light} [6]. The population sizes of the CMA-ES were chosen according to the heuristics $\lambda = \max(5, \min(m, 4 + \lfloor 3 \ln m \rfloor))$ and $\mu = \lfloor \lambda/4 \rfloor$ given in [5], where m is the number of object variables.

For each benchmark and each of the three kernel families, 20 evolutionary optimizations over 250 generations were performed. In each trial, the hyperparameter vector with the best fitness value (Test-5) was regarded as the solution. The performance of each solution was assessed by calculating the score using all 100 data partitions (Test-100), see Table 1.

We compared the start values of Test-5 and Test-100 with the results obtained by the ES. Except for a few cases, we achieved significantly (t-test, $p < 0.05$) better results by evolutionary optimization. The ES could even improve the performance when adapting $K_{\tau \boldsymbol{I}}$ in some cases, although the initial values were already tuned for $K_{\tau \boldsymbol{I}}$ by intensive grid search. The scaled kernels $K_{\boldsymbol{D}}$ as well as the scaled and rotated kernels $K_{\boldsymbol{A}}$ led to significantly ($p < 0.05$)

Data	Kernel	$\hat{t}$	λ	Test-5	Test-100	#SV-100
C	init			73.77	74.51	113.52
	$K_{\tau\boldsymbol{I}}$	10 ± 64	5	$74.00 \pm 0.35^\star$	$74.56 \pm 0.19^\star$	114.58 ± 4.34
	$K_{\boldsymbol{D}}$	83 ± 185	10	$76.91 \pm 4.19^\star$	$75.17 \pm 2.84^\star$	$113.10 \pm 3.89^\star$
	$K_{\boldsymbol{A}}$	179 ± 258	15	$77.47 \pm 2.32^\star$	$75.38 \pm 1.85^\star$	$112.70 \pm 2.97^\star$
D	init			76.34	76.67	247.83
	$K_{\tau\boldsymbol{I}}$	102 ± 340	5	$76.84 \pm 0.47^\star$	76.67 ± 0.16	251.63 ± 35.49
	$K_{\boldsymbol{D}}$	149 ± 244	9	$78.10 \pm 0.95^\star$	$76.88 \pm 1.1^\star$	$239.31 \pm 12.38^\star$
	$K_{\boldsymbol{A}}$	181 ± 243	14	$78.07 \pm 0.82^\star$	76.73 ± 1.39	$235.73 \pm 14.95^\star$
H	init			83.80	84.79	106.33
	$K_{\tau\boldsymbol{I}}$	10 ± 75	5	83.81 ± 0.19	84.74 ± 0.19	$103.37 \pm 4.23^\star$
	$K_{\boldsymbol{D}}$	97 ± 180	11	$85.71 \pm 1.86^\star$	$84.98 \pm 1.59^\star$	$76.52 \pm 9.15^\star$
	$K_{\boldsymbol{A}}$	153 ± 206	17	$85.86 \pm 0.75^\star$	$85.14 \pm 1.46^\star$	$75.51 \pm 6.53^\star$
T	init			96.27	95.83	16.36
	$K_{\tau\boldsymbol{I}}$	3 ± 22	5	$96.56 \pm 0.36^\star$	95.74 ± 0.86	15.99 ± 4.39
	$K_{\boldsymbol{D}}$	16 ± 151	6	$97.29 \pm 0.43^\star$	$96.01 \pm 0.17^\star$	$15.46 \pm 0.54^\star$
	$K_{\boldsymbol{A}}$	5 ± 14	12	$97.33 \pm 0^\star$	$96.01 \pm 0.23^\star$	$15.42 \pm 0.79^\star$

Table 1: Results averaged over 20 trials $\pm$ standard deviations. The first column specifies the benchmark: *Breast-Cancer* (C), *Diabetes* (D), *Heart* (H), and *Thyroid* (T). The second column indicates whether the results refer to the initial grid search values (init, [7]) or to the evolutionary optimized kernels $K_{\tau\boldsymbol{I}}$, $K_{\boldsymbol{D}}$, or $K_{\boldsymbol{A}}$. The $\hat{t}$ values are the generations needed to evolve the final solutions and λ is the number of offspring per generation. The percentages of correctly classified patterns on the 5 and 100 test sets are given by Test-5 and Test-100. The average numbers of support vectors over the 100 training sets is #SV-100. Results statistically significantly better compared to grid-search (init) are marked with $\star$ (two-sided t-test, $p < 0.05$).

better results compared to the ordinary Gaussian kernel $K_{\tau\boldsymbol{I}}$ (except for *Diabetes*: Test-100 with kernel $K_{\boldsymbol{A}}$). The major improvement was already achieved by allowing scaling. However, in one case (*Breast-Cancer*: Test-5) the kernel $K_{\boldsymbol{A}}$ with the additional rotation parameters gave significantly ($p < 0.05$) better results than just scaling. There is another remarkable advantage of the scaled kernel $K_{\boldsymbol{D}}$ and the scaled and rotated kernel $K_{\boldsymbol{A}}$: The number of support vectors decreases. Evolutionary adaptation of kernels $K_{\boldsymbol{D}}$ and $K_{\boldsymbol{A}}$ led to SVMs having significantly less support vectors compared to the ordinary kernel $K_{\tau\boldsymbol{I}}$ optimized by the ES or grid-search, see Table 1. Again, in one case (*Diabetes*) the kernel $K_{\boldsymbol{A}}$ with additional rotation parameters yielded a significantly ($p < 0.05$) better result than just scaling.

6 Conclusions

The CMA evolution strategy is a powerful, general method for SVM hyperparameter selection. It can handle a large number of kernel parameters and does neither require differentiable kernels and model selection criteria nor separability of the data. As typical selection criteria exhibit multiple local optima [4], we claim that evolutionary optimization is generally better suited for SVM model selection than gradient-based methods. We demonstrated that extended Gaussian kernels with scaling and rotating parameters can lead to a significantly better performance than standard Gaussian kernels. To our knowledge, for the first time a method that adapts the orientation of Gaussian kernels—i.e., that can detect correlations in the input data space *relevant for the kernel machine*—was presented. Our experiments show that by increasing the flexibility of the kernels the number of support vectors can be reduced. Future work will address other, especially non-differentiable kernels and optimization criteria.

References

[1] H.-G. Beyer and H.-P. Schwefel. Evolution strategies: A comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

[2] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1):131–159, 2002.

[3] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[4] C. Gold and P. Sollich. Model selection for support vector machine classification. *Neurocomputing*, 55(1-2):221–249, 2003.

[5] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[6] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, chapter 11, pages 169–184. MIT Press, 1999.

[7] P. Meinicke, T. Twellmann, and H. Ritter. Discriminative densities from maximum contrast estimation. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, Cambridge, MA, 2002. MIT Press.

[8] G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for adaboost. *Machine Learning*, 42(3):287–32, 2001.

[9] G. Rudolph. On correlated mutations in evolution strategies. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature 2 (PPSN II)*, pages 105–114. Elsevier, 1992.