Deep Support Vector Machines for Regression Problems

M.A. Wiering, M. Schutten, A. Millea, A. Meijster, and L.R.B. Schomaker
Institute of Artificial Intelligence and Cognitive Engineering
University of Groningen, the Netherlands
Contact e-mail: m.a.wiering@rug.nl

Abstract: In this paper we describe a novel extension of the support vector machine, called the deep support vector machine (DSVM). The original SVM has a single layer with kernel functions and is therefore a shallow model. The DSVM can use an arbitrary number of layers, in which lower-level layers contain support vector machines that learn to extract relevant features from the input patterns or from the extracted features of one layer below. The highest level SVM performs the actual prediction using the highest-level extracted features as inputs. The system is trained by a simple gradient ascent learning rule on a min-max formulation of the optimization problem. A two-layer DSVM is compared to the regular SVM on ten regression datasets and the results show that the DSVM outperforms the SVM.

Keywords: Support Vector Machines, Kernel Learning, Deep Architectures

1 Introduction

Machine learning algorithms are very useful for regression and classification problems. These algorithms learn to extract a predictive model from a dataset of examples containing input vectors and target outputs. Among all machine learning algorithms, one of the most popular methods is the SVM. SVMs have been used for many engineering applications such as object recognition, document classification, and different applications in bio-informatics, medicine and chemistry.

Limitations of the SVM. There are two important limitations of the standard SVM. The first one is that the standard SVM only has a single adjustable layer of model parameters. Instead of using such "shallow models", deep architectures are a promising alternative [4]. Furthermore, SVMs use a-priori chosen kernel functions to compute similarities between input vectors. A problem is that using the best kernel function is important, but kernel functions are not very flexible.

Related Work. Currently there is a lot of research in multi-kernel learning (MKL) [1, 5]. In MKL, different kernels are combined in a linear or non-linear way to create more powerful similarity functions for comparing input vectors. However, often only few parameters are adapted in the (non-linear) combination functions. In [2], another framework for two-layer kernel machines is described, but no experiments were performed in which both layers used non-linear kernels.

Contributions. We propose the deep SVM (DSVM), a novel algorithm that uses SVMs to learn to extract higher-level features from the input vectors, after which these features are given to the main SVM to do the actual prediction. The whole system is trained with simple gradient ascent and descent learning algorithms on the dual objective of the main SVM. The main SVM learns to maximize this objective, while the feature-layer SVMs learn to minimize it. Instead of adapting few kernel weights, we use large DSVM architectures, sometimes consisting of a hundred SVMs in the first layer. Still, the complexity of our DSVM scales only linearly with the number of SVMs compared to the standard SVM. Furthermore, the strong regularization power of the main SVM prevents overfitting.

2 The Deep Support Vector Machine

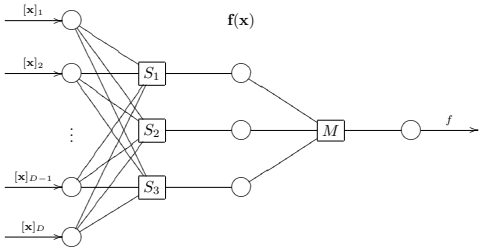

Fig. 1: Architecture of a two-layer DSVM. In this example, the feature layer consists of three SVMs S_a.

We use regression datasets: $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$, where $\mathbf{x}_i$ are input vectors and y_i are the target outputs. The architecture of a two-layer DSVM is shown in Figure 1. First, it contains an input layer of D inputs. Then, there are a total of d pseudo-randomly initialized SVMs S_a, each one learning to extract one feature $\mathbf{f}(\mathbf{x})_a$ from an input pattern $\mathbf{x}$. Finally, there is the main support vector machine M that approximates the target function using the extracted feature vector as

input. For computing the feature-layer representation $\mathbf{f}(\mathbf{x})$ of input vector $\mathbf{x}$, we use:

$$\mathbf{f}(\mathbf{x})_a = \sum_{i=1}^{\ell} (\alpha_i^*(a) - \alpha_i(a)) K(\mathbf{x}_i, \mathbf{x}) + b_a,$$

which iteratively computes each element $\mathbf{f}(\mathbf{x})_a$. In this equation, $\alpha_i^*(a)$ and $\alpha_i(a)$ are SVM coefficients for SVM S_a, b_a is its bias, and $K(\cdot,\cdot)$ is a kernel function. For computing the output of the whole system, we use:

$$g(\mathbf{f}(\mathbf{x})) = \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i) K(\mathbf{f}(\mathbf{x}_i), \mathbf{f}(\mathbf{x})) + b.$$

Learning Algorithm. The learning algorithm adjusts the SVM coefficients of all SVMs through a min-max formulation of the dual objective W of the main SVM:

$$\min_{\mathbf{f}(\mathbf{x})} \max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} W(\mathbf{f}(\mathbf{x}), \boldsymbol{\alpha}^{(*)}) = -\varepsilon \sum_{i=1}^{\ell} (\alpha_i^* + \alpha_i) + \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i) y_i$$
$$-\frac{1}{2} \sum_{i,j=1}^{\ell} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) K(\mathbf{f}(\mathbf{x}_i), \mathbf{f}(\mathbf{x}_j))$$

We have developed a simple gradient ascent algorithm to train the SVMs. The method adapts the SVM coefficients $\boldsymbol{\alpha}^{(*)}$ (standing for all α_i^* and α_i) toward a (local) maximum of W, where λ is the learning rate: $\alpha_i^{(*)} \leftarrow \alpha_i^{(*)} + \lambda \cdot \partial W / \partial \alpha_i^{(*)}$. The resulting gradient ascent learning rule for α_i is:

$$\alpha_i = \alpha_i + \lambda(-\epsilon - y_i + \sum_j (\alpha_j^* - \alpha_j) K(\mathbf{f}(\mathbf{x}_i), \mathbf{f}(\mathbf{x}_j)))$$

We use radial basis function (RBF) kernels in both layers of a two-layered DSVM. Results with other kernels were worse. For the main SVM:

$$K(\mathbf{f}(\mathbf{x}_i), \mathbf{f}(\mathbf{x})) = \exp(-\sum_a \frac{(\mathbf{f}(\mathbf{x}_i)_a - \mathbf{f}(\mathbf{x})_a)^2}{\sigma_m})$$

The system constructs a new dataset for each feature-layer SVM S_a with a backpropagation-like technique for making examples: $(\mathbf{x}_i, \mathbf{f}(\mathbf{x}_i)_a - \mu \cdot \delta W / \delta \mathbf{f}(\mathbf{x}_i)_a)$, where μ is some learning rate, and $\delta W / \delta \mathbf{f}(\mathbf{x}_i)_a$ is given by:

$$\frac{\delta W}{\delta \mathbf{f}(\mathbf{x}_i)_a} = (\alpha_i^* - \alpha_i) \sum_{j=1}^{\ell} (\alpha_j^* - \alpha_j) \frac{\mathbf{f}(\mathbf{x}_i)_a - \mathbf{f}(\mathbf{x}_j)_a}{\sigma_m} \cdot K(\mathbf{f}(\mathbf{x}_i), \mathbf{f}(\mathbf{x}_j))$$

The feature extracting SVMs are pseudo-randomly initialized and then alternated training of the main SVM and feature layer SVMs is executed a number of epochs. The bias values are computed from the average errors.

3 Experimental Results

We experimented with 10 regression datasets to compare the DSVM to an SVM, both using RBF kernels. Both methods are trained with our simple gradient ascent learning rule, adapted to also consider penalties, e.g. for obeying the bias constraint. The first 8 datasets are described in [3] and the other 2 datasets are taken from the UCI repository. The number of examples per dataset ranges from 43 to 1049, and the number of features is between 2 and 13. The datasets are split into 90% trainingdata and 10% testingdata. For optimizing the learning parameters we have used particle swarm optimization. Finally, we used 1000 or 4000 times cross-validation with the best found parameters to compute the mean squared error and its standard error.

Dataset	SVM results	DSVM results
Baseball	0.02413 ± 0.00011	**0.02294** ± 0.00010
Boston H.	0.006838 ± 0.000095	**0.006381** ± 0.000090
Concrete .	0.00706 ± 0.00007	**0.00621** ± 0.00005
Electrical	0.00638 ± 0.00007	0.00641 ± 0.00007
Diabetes	0.02719 ± 0.00026	**0.02327** ± 0.00022
Machine-CPU	0.00805 ± 0.00018	**0.00638 ± 0.00012**
Mortgage	0.000080 ± 0.000001	0.000080 ± 0.000001
Stock	0.000862 ± 0.000006	**0.000757 ± 0.000005**
Auto-MPG	6.852 ± 0.091	6.715 ± 0.092
Housing	**8.71 ± 0.14**	9.30 ± 0.15

Tab. 1: The mean squared errors and standard errors of the SVM and the two-layer DSVM on 10 regression datasets.

Table 1 shows the results. The results of the DSVM are significantly better for 6 datasets ($p < 0.001$) and worse on one. From the results we can conclude that the DSVM is a powerful novel machine learning algorithm. More research, such as adding more layers and implementing more powerful techniques to scale up to big datasets, can be done to discover its full potential.

References

[1] F.R. Bach, G.R.G. Lanckriet, and M.I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, pages 6–15, 2004.

[2] Francesco Dinuzzo. Kernel machines with two layers and multiple kernel learning. *CoRR*, 2010.

[3] M. Graczyk, T. Lasota, Z. Telec, and B. Trawinski. Nonparametric statistical analysis of machine learning algorithms for regression problems. In *Knowledge-Based and Intelligent Information and Engineering Systems*, pages 111–120. 2010.

[4] G.E. Hinton and R.R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313:504–507, 2006.

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