Deep Support Vector Machines

Marco A. Wiering

Institute of Artificial Intelligence and Cognitive Engineering
University of Groningen, the Netherlands

Presentation at ROKS'13, Leuven, 09 July 2013

Contents

Machine Learning

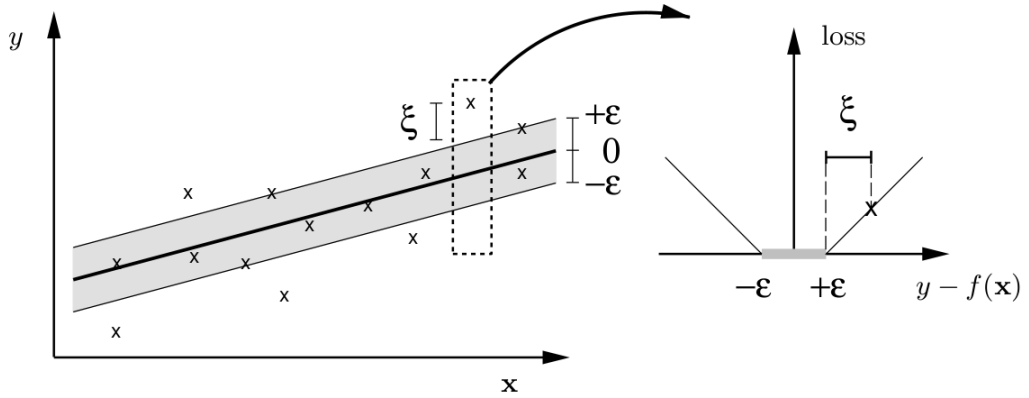
- Machine learning algorithms are very useful for many applications (Classification, Regression, Adaptive Control)
- They use datasets or experiences to fit a model
- Example applications are:
  - Object Recognition
  - Face Recognition
  - Handwritten Digit Recognition
  - fMRI-Scan Classification
  - Medical Diagnosis
  - Document Classification
  - Many Applications in Bio-informatics and Chemistry

Limitations of Support Vector Machines

- Support Vector Machines (SVM) often outperform other machine learning methods
- However, the standard SVM has a single adjustable layer of weights
- Instead of using such “shallow models”, deep architectures can be better alternatives
- SVMs use a-priori chosen kernel functions to compute similarities between input vectors
- A problem is that the choice of kernel function is important, but kernel functions are not very flexible
- Therefore we propose the deep SVM (DSVM)
- The DSVM contains multiple layers of SVMs

Support Vector Regression

- The objective function of the SVM is based on structural risk minimization theory developed by Vapnik in the 1960s
- Goal: find $g(\mathbf{x})$ most suitable to the data, e.g. for regression, ϵ-insensitive (Hinge) loss function:
  - $|y_i - g(\mathbf{x}_i)| \leq \varepsilon$
- But also generalize well!
  - $g(\mathbf{x})$ as *flat* as possible $\Rightarrow ||\mathbf{w}||$ as small as possible
- Yields a *convex optimization problem*



$$\min_{\mathbf{w},\boldsymbol{\xi}^{(*)},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*)$$

subject to constraints

$$y_i - \mathbf{w}\cdot\mathbf{x}_i - b \leq \varepsilon + \xi_i,$$

$$\mathbf{w}\cdot\mathbf{x}_i + b - y_i \leq \varepsilon + \xi_i^*,$$

$$\xi_i, \xi_i^* \geq 0.$$

SVM Regression Objective Function

- The resulting dual objective problem is:

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

subject to constraints

$$0 \leq \alpha_i, \alpha_i^* \leq C, \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*) = 0.$$

- Then: $o = g(\mathbf{x}) = \sum_{i=1}^{\ell}(\alpha_i^* - \alpha_i)(\mathbf{x}_i \cdot \mathbf{x}) + b$

Optimization Algorithms

- For SVMs specialized toolkits have been developed such as SVMLight and LibSVM.
- There are multiple optimization algorithms that aim to maximize the dual objective:
  - Sequential Minimal Optimization (SMO) is often used
  - Quadratic Programming can be used as well
  - A simple solution is to use gradient ascent:

$$\alpha_i \leftarrow \alpha_i + \lambda \cdot \frac{\partial W(\cdot)}{\partial \alpha_i} = \alpha_i + \lambda(-\epsilon - y_i + \sum_{j=1}^{\ell}(\alpha_j^* - \alpha_j) K_2(\mathbf{f}(\mathbf{x}_i|\theta), \mathbf{f}(\mathbf{x}_j|\theta)))$$

and the gradient ascent learning rule for α_i^* is:

$$\alpha_i^* \leftarrow \alpha_i^* + \lambda \cdot \frac{\partial W(\cdot)}{\partial \alpha_i^*} = \alpha_i^* + \lambda(-\epsilon + y_i - \sum_{j=1}^{\ell}(\alpha_j^* - \alpha_j) K_2(\mathbf{f}(\mathbf{x}_i|\theta), \mathbf{f}(\mathbf{x}_j|\theta)))$$

(Multiple) Kernel Learning

- A recent development in kernel methods and SVMs is (multiple) kernel learning
- Here kernels are adjusted to the data by some training process
- An example is to train weighting coefficients ϕ^a for the RBF kernel:

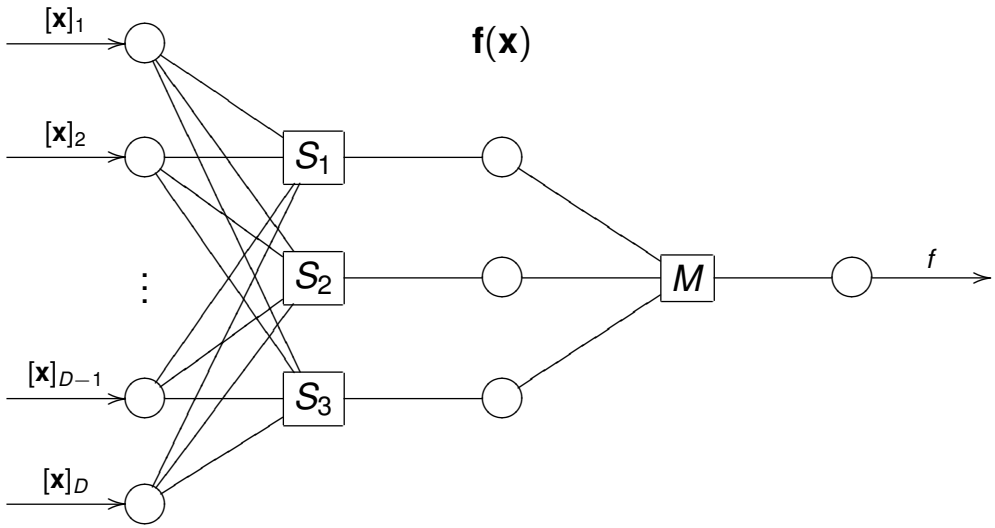
$$K(\mathbf{x}_i, \mathbf{x}) = \exp(-\sum_a \frac{\phi^a(\mathbf{x}_i^a - \mathbf{x}^a)^2}{\sigma})$$

- An interesting result is that kernel learning with the dual objective leads to a min-max optimization problem.

Main Ideas of DSVMs

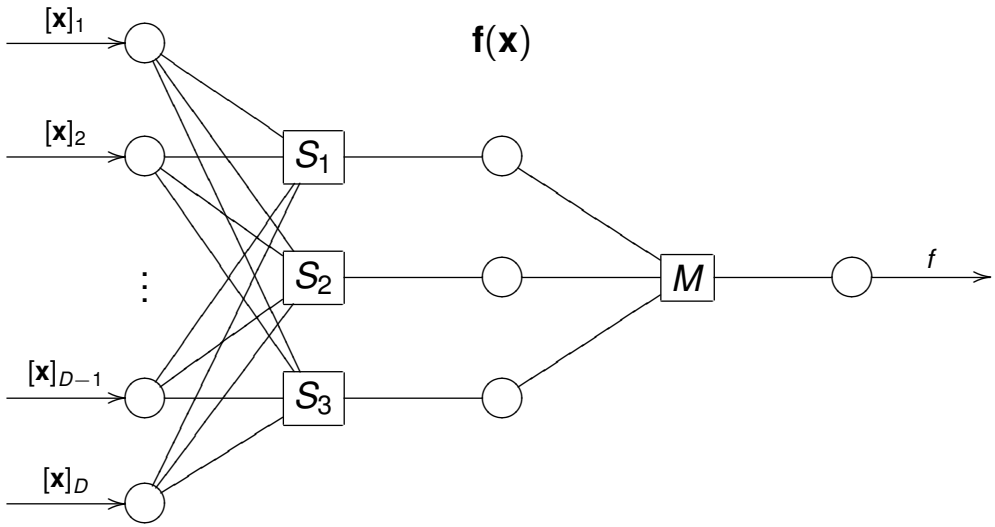
- Choosing the right (parameterized) kernel may be difficult
- Instead, we will use a set of SVMs to map the input vector $\mathbf{x}$ to a feature vector $\mathbf{f}(\mathbf{x})$
- More SVMs can be used to create larger feature representations
- All support vector coefficients (α-values) are trained using gradient ascent or descent on an adapted dual objective function
- Just like Multi-layer perceptrons consist of simple perceptrons, the DSVM consists of SVMs

Architecture



- Input layer of size D
- Total of d SVMs S_a, each one extracting one feature
- Central feature layer of size d
- Main support vector machine M

Workings



- $\mathbf{f}(\mathbf{x})$ the representation of $\mathbf{x}$ in the feature layer
- $\mathbf{f}(\mathbf{x})_a = \sum_{i=1}^{\ell}(\alpha_i^*(a) - \alpha_i(a))(K(\mathbf{x}_i, \mathbf{x})) + b_a.$
- $g(\mathbf{f}(\mathbf{x})) = \sum_{i=1}^{\ell}(\alpha_i^* - \alpha_i)(K(\mathbf{f}(\mathbf{x}_i), \mathbf{f}(\mathbf{x})) + b.$

Adapted Objective

- Output function: $g(\mathbf{x}) \Rightarrow g(\mathbf{f}(\mathbf{x}))$
- Objective function: $W(\boldsymbol{\alpha}^{(*)}) \Rightarrow W(\mathbf{f}(\mathbf{x}), \boldsymbol{\alpha}^{(*)})$
- New optimization problem:

$$\min_{\mathbf{f}(\mathbf{x})} \max_{\boldsymbol{\alpha}, \boldsymbol{\alpha}^*} W(\mathbf{f}(\mathbf{x}), \boldsymbol{\alpha}^{(*)})$$

- This is a min-max optimization problem
  - Adapt $\mathbf{f}(\mathbf{x})$ through gradient descent
  - Adapt $\boldsymbol{\alpha}^{(*)}$ through gradient ascent

Training Procedure (1)

- Adapt $\boldsymbol{\alpha}^{(*)}$ towards a (local) maximum of $W(\mathbf{f}(\mathbf{x}), \boldsymbol{\alpha}^{(*)})$
- $\alpha_i^{(*)} \leftarrow \alpha_i^{(*)} + \lambda \frac{\partial W}{\partial \alpha_i^{(*)}}$
- Remember:

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

- The resulting gradient ascent SVM training rule for α_i:

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

Training Procedure (2)

- Similarly, the output of S_a should be *decreased* by $\frac{\partial W}{\partial [\mathbf{f}(\mathbf{x}_i)]_a}$
- We will use radial basis function (RBF) kernels in both layers of a two-layered DSVM:

$$K(\mathbf{x}_i, \mathbf{x}) = e^{-\sum_k \frac{(x_i^k - x^k)^2}{\sigma s}}$$

- Consider $\frac{\partial W}{\partial [\mathbf{f}(\mathbf{x}_i)]_a}$ to be the error of S_a

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

Training Procedure (3)

- For the RBF kernel of the main SVM we have:

$$\frac{\delta K(\mathbf{f}(\mathbf{x}_i), \mathbf{f}(\mathbf{x}))}{\delta \mathbf{f}(\mathbf{x}_i)_a} = -2\frac{\mathbf{f}(\mathbf{x}_i)_a - \mathbf{f}(\mathbf{x}_j)_a}{\sigma_m} K(\mathbf{f}(\mathbf{x}_i), \mathbf{f}(\mathbf{x}_j))$$

- This leads to:

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

- We create a new dataset for each feature extracting SVM and then train it with the gradient ascent SVM algorithm
- We repeat the alternating training of the main SVM and feature layer SVMs a number of times

Related Work

The DSVM is related to the following methods:

- Kernel learning. Often relies on a fixed set of basis kernels, where
  - Parameters are learned for a kernel (e.g. RBF kernel), or:
  - Different kernels are linearly or non-linearly combined
  - There are recent developments in multi-layer kernel learning, e.g. Dinuzzo (2010)
- Suykens (1999) used logistic functions to learn features. The learning algorithm was quite different
- Vincent and Y. Bengio (2000) proposed a neural support vector network, but it used a random subset of support vectors and a heuristic to adapt the neural networks

Experiments on Regression Problems

- We experimented with 10 regression datasets
- The data is split in 90% trainingdata and 10% testingdata
- We perform 1000 or 4000 times crossvalidation
- For optimizing the parameters we have used Particle Swarm Optimization (PSO)
- First we used a global search for good parameters
- Then we used PSO for a finetuning phase
- The longest experiments to find parameters took around two days for the DSVM on a 32-core machine
- We will show average squared errors and standard errors

Results on Regression Problems

Dataset	#inst.	#feat.	N	SVM results	DSVM results	Graczyk results
Baseball	337	6	4000	0.02413 $\pm$ 0.00011	**0.02294** $\pm$ 0.00010	0.02588
Boston Housing	461	4	1000	0.006838 $\pm$ 0.000095	**0.006381** $\pm$ 0.000091	0.007861
Concrete Strength	72	5	4000	0.00706 $\pm$ 0.000070	**0.00621** $\pm$ 0.000054	0.008509
Diabetes	43	2	4000	0.02719 $\pm$ 0.000263	**0.02327** $\pm$ 0.000219	0.025154
Machine-CPU	188	6	1000	0.00805 $\pm$ 0.000181	**0.00638** $\pm$ **0.000123**	0.007766
Mortgage	1049	6	1000	0.000080 $\pm$ 0.000001	0.000080 $\pm$ 0.000001	0.000081
Stock	950	5	1000	0.00086 $\pm$ 0.000006	**0.00076** $\pm$ **0.000005**	0.002385
Breast Cancer	152	6	4000	0.06947 $\pm$ 0.000297	0.06910 $\pm$ 0.000295	**0.068615**
Auto-MPG	392	7	1000	6.852 $\pm$ 0.091	6.715 $\pm$ 0.092	N/A
Housing	506	13	1000	**8.71** $\pm$ **0.14**	9.30 $\pm$ 0.15	N/A

Experiments on Classification Problems

- We experimented with 8 classification datasets from the UCI repository
- The data is split in 90% trainingdata and 10% testingdata
- We perform 1000 times crossvalidation
- For optimizing the parameters we have again used Particle Swarm Optimization (PSO)
- We will show average accuracies and standard errors

Results on Classification Problems

Dataset	#Instances	#Features	#Classes	MLP	SVM	DSVM
Hepatitis	155	19	2	84.3 $\pm$ 0.3	81.9 $\pm$ 0.3	**85.1** $\pm$ 0.2
Breast Cancer W.	699	9	2	97.0 $\pm$ 0.1	96.9 $\pm$ 0.1	96.9 $\pm$ 0.1
Ionosphere	351	34	2	91.1 $\pm$ 0.1	94.0 $\pm$ 0.1	**95.5** $\pm$ 0.1
Ecoli	336	7	8	87.6 $\pm$ 0.2	87.0 $\pm$ 0.2	87.4 $\pm$ 0.2
Glass	214	9	7	64.5 $\pm$ 0.4	70.1 $\pm$ 0.3	**73.9** $\pm$ 0.3
Pima Indians	768	8	2	77.4 $\pm$ 0.1	77.1 $\pm$ 0.1	77.3 $\pm$ 0.1
Votes	435	16	2	96.6 $\pm$ 0.1	96.5 $\pm$ 0.1	96.5 $\pm$ 0.1
Iris	150	4	3	97.8 $\pm$ 0.1	96.5 $\pm$ 0.2	**98.2** $\pm$ 0.1
Average				87.0	87.5	88.9

Summary

- The DSVM learns feature representations relevant to task
- It does this through a complex interaction between different adaptive layers
- Training involves a challenging min-max optimization problem
- Resulting algorithms are non-convex
- Results better than state-of-the-art machine learning methods

Future Work

- We want to use the DSVM for autoencoding
- We want to run experiments on challenging supervised learning problems
- We want to see how well the DSVM performs for reinforcement learning
- We want to construct GPU implementation for the DSVM
- We want to study the theory of the min-max problem in the DSVM

Acknowledgments

I want to thank the following people for their help in this project:

- Marten Schutten
- Lambert Schomaker
- Arnold Meijster
- Adrian Millea
- Michiel van der Ree
- Aleke Nolte
- Egbert van der Wal
- Marijn Stollenga
- Mark Embrechts
- Magdalena Graczyk

Questions