Software Documentation
# of the
Potential Support Vector Machine

Tilman Knebel and Sepp Hochreiter
Department of Electrical Engineering and Computer Science
Technische Universität Berlin
10587 Berlin, Germany
{tk,hochreit}@cs.tu-berlin.de

Contents

1 Introduction 3
1.1 Machine Learning 3
1.2 Support Vector Machines 3
1.3 The Potential Support Vector Machine 4

2 Software Features 6
2.1 Command Line Application 6
2.2 MATLAB interface 7

3 The components of the P-SVM 11
3.1 P-SVM for Measured Dyadic (Matrix) Data 11
3.2 P-SVM for Vectorial Data 12
3.3 P-SVM for Vectorial Data using Complex Feature Vectors 13
3.4 Hyperparameters 14
3.5 Comparison P-SVM with libSVM 15

References 17

Abstract

This documentation introduces the PSVM (see [ncpsvm]) software library which is designed for MICROSOFT Windows as well as for UNIX systems. Compiling the software results in a program which can be used with command line options (e.g. kernel type, learning/testing, etc.) which does not depend on other software or on a particular software-environment. The PSVM software package also includes a MATLAB interface for convenient working with the PSVM package. In addition lots of sample data and scripts are included. The PSVM software contains a classification, a regression, and a feature selection mode and is based on an efficient SMO optimization technique. The software can directly be applied to dyadic (matrix) data sets or it can be used with kernels like standard SVM software. In contrast to standard SVMs the kernel function does not have to be positive definite, e.g. the software already implements the indefinite sin-kernel. An important feature of the software is that is allows for n-fold cross validation and for hyperparameter selection. For classification tasks it offers the determination of the significance level and ROC data. In summary the basic features of the software are

- WINDOWS and UNIX compatible
- no dependencies to other software
- command line interface
- MATLAB interface
- n-fold cross validation
- hyperparameter selection
- relational data
- non-Mercer kernels
- significance testing
- computation of Receiver-Oprator-Characteristic (ROC) curves

If you use the software please cite [ncpsvm] in your publications.

1 Introduction

1.1 Machine Learning

The main tasks in machine learning are learning and prediction. Learning in this case means to fit a model with a so called training dataset and its known target values in order to apply the model to future, unknown test data. The prediction task then maps a set of inputs to the corresponding target values.
The difficulty is to choose a model, its degree of freedom, and the free parameters, while maintaining a good prediction performance on the test data, i.e. the outputs should match the unknown target values. This objective is known as to generalize to unknown data (generalization). If the dataset with known targets is small, the generalization capabilities can be measured by dividing the dataset into n parts and doing n training tasks while using one different part for testing each. The generalization performance is determined by averaging over all testing tasks. This is called n-fold crossvalidation.
Furthermore it may be interesting if a good generalization performance as measured for example via n-fold crossvalidation is caused by chance while drawing the training examples. Therefore, the performance is compared with the performance after shuffling the target data. Then the probability of being better on shuffled data is approximated with the relative frequency of training cycles, where the performance is the same or better than the performance for the original target values. This is called a significance test.

1.2 Support Vector Machines

A standard support vector machine (SVM) (Schölkopf and Smola, 2002; Vapnik, 1998) is a model design, which is applicable to real valued vectorial data, where each component is called an attribute of the examples. Ordinal attributes (or targets) can be converted into numerical attributes by creating a new attribute for each value and use $+1$ or -1 depending on whether this value is assumed or not. If targets are real valued the learning problem is called regression, if targets are binary it is called classification. Non-binary ordinal targets require a combination of multiple support vector machines, which is called a multi class support vector machine.
In SVM learning the data is internally transformed into a high dimensional feature space and a set of so called support vectors are selected from the training set. The support vectors define a hyperplane in the feature space which is used to calculate the target values. If a simple linear SVM is used, the feature space is equal to the data space. In most SVM algorithms the transformation into the feature space is done implicitly by a kernel function, which qualifies a relation between two ex-

amples in feature space.
The generalization performance of SVMs may depend on some user defined hyperparameters. A simple way to optimize these parameters is a grid search using the crossvalidation error as a measure of quality.

1.3 The Potential Support Vector Machine

Understanding the capabilities of the Potential Support Vector Machine needs a discussion of different representations of data:

- Vectorial Data:
  Most of the techniques for solving classification and regression problems were specifically designed for vectorial data, where data objects are described by vectors of numbers which are treated as elements of a vector space (figure 1.3A). They are very convenient, because of the structure imposed by the Euclidean metric. However, there are datasets for which a vector-based description is inconvenient or simply wrong, and representations which consider relationships between objects, are more appropriate.

- Dyadic Data:
  In dyadic descriptions, the whole dataset is represented using a rectangular matrix whose entries denote the relationships between “row” and “column” objects (figure 1.3C). The column objects are the objects to be described and the row objects serve for their description (Hofmann and Puzicha, 1998; Li and Loken, 2002; Hoff, 2005).

- Dyadic Pairwise Data:
  If “row” and “column” objects are from the same set, the representation is usually called pairwise data, and the entries of the matrix can often be interpreted as the degree of similarity (or dissimilarity) between objects (figure 1.3B).

Dyadic descriptions are more powerful than vector-based descriptions, as vectorial data can always be brought into dyadic form when required. This is often done for kernel-based classifiers or regression functions like the standard SVM, where a Gram matrix of mutual similarities is calculated before the predictor is learned. A similar procedure can also be used in cases where the “row” and “column” objects are from different sets. If both of them are described by feature vectors, a matrix can be calculated by applying a kernel function to pairs of feature vectors, one vector describing a “row” and the other vector describing a “column” object. In many cases, however, dyadic descriptions emerge, because the matrix entries are measured directly. Pairwise data representations as a special case of dyadic

A)

	a	b	c
α	2	4	7
β	-1	-3	6
χ	0	-5	8
δ	1	5	8
ϵ	-1	4	7

B)

	α	β	χ	δ	ϵ
α	1	-0.1	0.2	0.9	-0.5
β	-0.1	1	0.2	0.1	0.3
χ	0.2	0.2	1	-0.2	-0.2
δ	0.9	0.1	-0.2	1	0.5
ϵ	-0.5	0.3	-0.2	0.5	1

C)

	α	β	χ	δ	ϵ	ϕ
A	0	2	0	-1	-7	-8
B	1	7	2	0	0	-2
X	8	-9	-1	0	1	2
Δ	-7	0	8	-2	1	0
E	1	0	3	-2	0	0

Figure 1: A) vectorial data: The vectorial objects $\{\alpha, \beta, \ldots\}$ are described by its components $\{a, b, \ldots\}$. B) vectorial objects and descriptors: The application of a kernel function to vectorial data results in a matrix where the objects $\{\alpha, \beta, \ldots\}$ are described by mutual relations. C) true dyadic data: The column objects $\{\alpha, \beta, \ldots\}$ are described by measured relations to row objects $\{A, B, \ldots\}$.

data can be found for datasets where similarities or distances between objects are measured. Genuine dyadic data occur whenever two sets of objects are related. Traditionally, "row" objects have often been called "features" and "column" vectors of the dyadic data matrix have mostly been treated as "feature vectors".

We suggest to interpret the matrix entries of dyadic data as the result of a kernel function or measurement kernel, which takes a "row" object, applies it to a "column" object, and outputs a number. Using an improved measure for model complexity and a new set of constraints which ensure a good performance on the training data, we arrive at a generally applicable method to learn predictors for dyadic data. The new method is called the "potential support vector machine" (P-SVM). It can handle rectangular matrices as well as pairwise data whose matrices are not necessarily positive semidefinite, but even when the P-SVM is applied to regular Gram matrices, it shows very good results when compared with standard methods. Due to the choice of constraints, the final predictor is expanded into a usually sparse set of descriptive "row" objects, which is different from the standard SVM expansion in terms of the "column" objects. This opens up another important application domain: a sparse expansion is equivalent to feature selection (Guyon and Elisseeff, 2003; Hochreiter and Obermayer, 2004b; Kohavi and John, 1997; Blum and Langley, 1997).

2 Software Features

The software is written in C++ and implements the potential support vector machine from [ncpsvm, psvmsmo]. The source files are divided into the software library which offers the main P-SVM functions, a command line user interface and a MATLAB interface. Binaries for Microsoft Windows Systems are included, for other systems a C++ compiler is required to produce the binaries. The software includes sample datasets for testing the command line application, sample scripts for testing the MATLAB implementation, and a user manual. Some of the MATLAB sample scripts refer to functions from the libSVM library [libsvm], which is also included in the package.
The current version of the software requires that:

- inputs and targets must be real valued and ”don’t care” free
- binary inputs and targets must be encoded as +1. or -1.

Ordinal values are not supported directly, they must be encoded via binary attributes or labels as mentioned in (1.2).

2.1 Command Line Application

The command line application uses files in plain text format for data input and output. All progress and result messages are sent to the console output and additionally into a log file.
The offered functionality is

- P-SVM regression and classification
- P-SVM feature extraction
- the C- and ϵ-regularization schemes ([ncpsvm])
- training with n-fold crossvalidation, hyperparameter selection through grid search, significance testing
- choice between the five predefined kernel functions: linear, polynomial, radial basis, sigmoid, and plummer
- implementation of user defined kernels
- application of the P-SVM to true dyadic (matrix) data
- prediction of labels after training

- calculation of ROC curves for validation purposes
- SMO optimization options (epsilon annealing, block optimization, and reduction) for tradeoff between speed and precision
- GNU-Plot interface for the result of the hyperparameter selection through grid search (figure 2), for the ROC curves for validation (figure 3), and for the significance testing (figure 4) by exporting data and script files.
- calculation of an upper limit of hyperparameters ϵ and C for a given dataset (helps the user to select hyperparameters which affect the predictor)
- calculation of mean and variance for a given dataset (helps to select useful kernel parameters)

2.2 MATLAB interface

The package includes sourcecodes of six MEX-functions which bind the P-SVM library and serve as an interface for the main P-SVM routines to MATLAB. For Microsoft Windows Systems the binaries are included as dynamic link libraries which can directly be used as MATLAB functions. For other systems a C++ compiler is required to produce the binaries. The result is a system dependent library which can be accessed directly whithin MATLAB. The main MATLAB functions are "psvmtrain" and "psvmpredict" and offer:

- P-SVM regression and classification
- P-SVM feature extraction
- the C- and ϵ-regularization schemes ([ncpsvm])
- choice between the five predefined kernel functions: linear, polynomial, radial basis, sigmoid, and plummer
- implementation of user defined kernels
- application of the P-SVM to true dyadic (matrix) data
- prediction of labels after training
- SMO optimization options (epsilon annealing, block optimization, and reduction) for tradeoff between speed and precision

As explained in the next section the training algorithm within P-SVM divides into three tasks, which might be interesting to access separately within MATLAB. These functions are

- “psvmgetnp” and “psvmputnp” for accessing the normalization task
- “psvmkernel” for calculation of one of the five predefined kernel functions or a user defined kernel function
- “psvmsmo” for solving the P-SVM optimization problem

In contrast to the command line version, the compiled MATLAB functions offer no crossvalidation, modelselection and significance testing. Instead of that the package includes MATLAB scripts for doing

- crossvalidation
- exporting data matrices to files which can be processed directly through the P-SVM command line application
- exporting data matrices to files which can be processed directly through the application “libsvm” (see [libsvm])
- random shuffling of data matrices as preprocessing step for crossvalidation

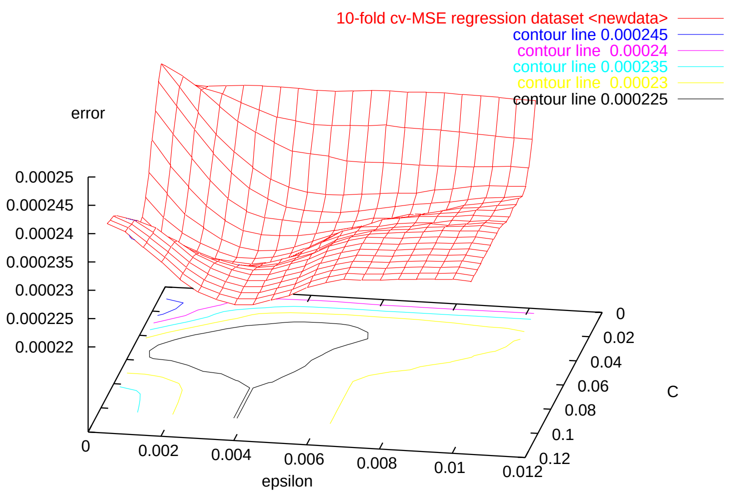

Figure 2: Generalization error estimated via 10-fold crossvalidation for different combinations of the hyperparameters. This plot is generated by default if the hyperparameter selection mode of the P-SVM software is used.

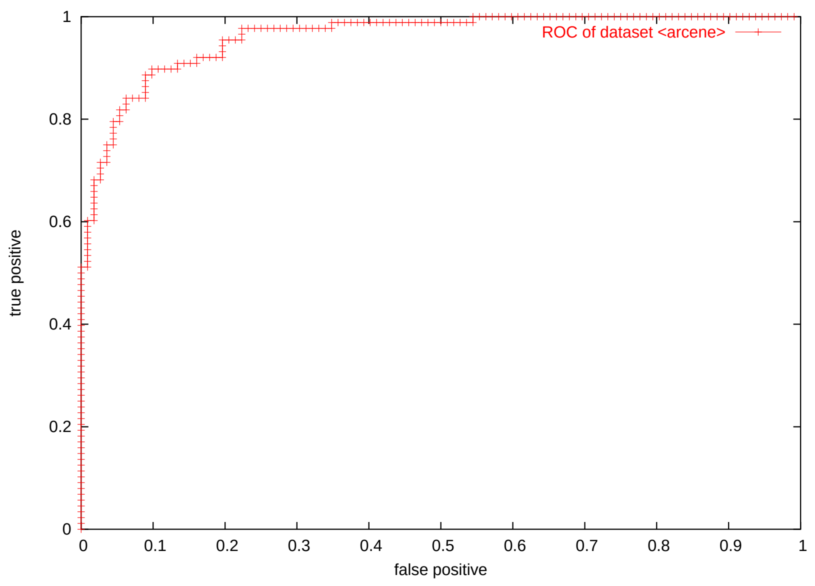

Figure 3: Receiver Operating Characteristic (ROC) of the P-SVM for a P-SVM classifier trained with the sample dataset 'arcene'. This plot is generated by default with option '-test' of the P-SVM software.

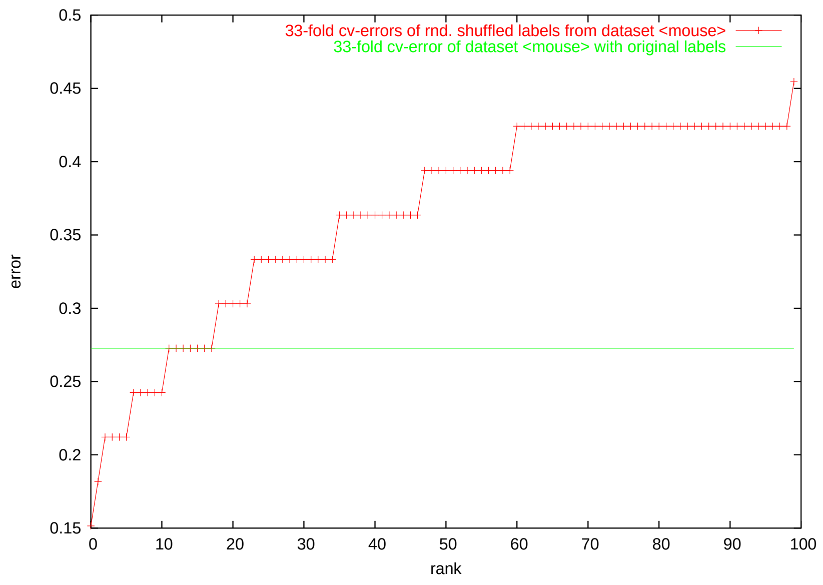

Figure 4: Visualization of the significance testing results: This plot shows the generalization errors of the sample dataset ’arcene’ estimated by 10-fold crossvalidation ordered by magnitude after random shuffling of the training label. Every data point corresponds to a specific label permutation and provides results after hyperparameter selection has been performed. The horizontal line marks the error for original labels. This plot is generated by default after enabling the significance test with the option ’-sig’. The corresponding confidence interval is calculated and printed to the console.

3 The components of the P-SVM

The software can handle dyadic (matrix) data as well as vectorial data via predefined kernels.

3.1 P-SVM for Measured Dyadic (Matrix) Data

The P-SVM predictor for dyadic data has the form:

$$o_c(\boldsymbol{k}) \;=\; \text{sign}(\text{norm}(\boldsymbol{k}, \boldsymbol{np}) \cdot \boldsymbol{\alpha} \;+\; b)$$

for classification and

$$o_r(\boldsymbol{k}) \;=\; \text{norm}(\boldsymbol{k}, \boldsymbol{np}) \cdot \boldsymbol{\alpha} \;+\; b$$

for regression tasks, where

$\boldsymbol{k}$:	dyadic description of the "column" object which is to be classified: P-component vector which quantifies the relations to P "row" objects
o_c :	P-SVM class prediction of $\boldsymbol{k}$ ($+1$ or -1)
o_r :	P-SVM real valued prediction of $\boldsymbol{k}$
$\boldsymbol{\alpha}$:	support features (part of the P-SVM model): P-component vector which quantifies the importances of "row" objects for serving as features.
b :	bias (part of the P-SVM model)
$\boldsymbol{np}$:	normalization statistics (part of the P-SVM model) - Breaks into maximum, minimum, mean, and variance for all P "row" objects and is used by the "norm"-function.

Learning proceeds by

1. Data Matrix: loading the $L \times P$ dyadic data matrix $\boldsymbol{K}$.
2. Normalization: Gets statistics for all P columns of $\boldsymbol{K}$ and stores its maxima, minima, mean, and a scaling factor in P-dimensional vectors $\boldsymbol{np}$. $\boldsymbol{np}$ is then used to normalize each column of $\boldsymbol{K}$.
3. Calculation of b: The bias b is set to the mean of the L labels represented by the label vector $\boldsymbol{y}$.
4. Sequential Minimal Optimization (SMO): The SMO reads the normalized kernel matrix $\boldsymbol{K}$ and the labels $\boldsymbol{y}$ and calculates a p-dimensional vector $\boldsymbol{\alpha}$ for a normalized kernel matrix $\boldsymbol{K}$ and labels $\boldsymbol{y}$. α_i corresponds to the i-th column of $\boldsymbol{K}$ and is normally sparse.

3.2 P-SVM for Vectorial Data

The P-SVM predictor for vectorial data has the form:

$$o_c(\boldsymbol{u}) \;=\; \text{sign}(\text{norm}(\boldsymbol{k}(\boldsymbol{u}, \boldsymbol{X}), \boldsymbol{np}) \cdot \boldsymbol{\alpha} \;+\; b)$$

for classification and

$$o_r(\boldsymbol{u}) \;=\; \text{norm}(\boldsymbol{k}(\boldsymbol{u}, \boldsymbol{X}), \boldsymbol{np}) \cdot \boldsymbol{\alpha} \;+\; b$$

for regression tasks, where

$\boldsymbol{u}$: "column" object which is to be classified (N-component vector)

o_c : P-SVM class prediction of $\boldsymbol{u}$ ($+1$ or -1)

o_r : P-SVM real valued prediction of $\boldsymbol{u}$

$\boldsymbol{X}$: $N \times P$ matrix composed by P "row" objects (N-component vectors). The "row" objects and "column" objects match and $\boldsymbol{X}$ equals the set initially used for training.

$\boldsymbol{k}()$: calculates a P component vector by applying a kernel function to a "column" object and a set of P "row" objects

$\boldsymbol{\alpha}$: support features (part of the P-SVM model): P-component vector which quantifies the importances of "row" objects for serving as features.

b : bias (part of the P-SVM model)

$\boldsymbol{np}$: normalization statistics (part of the P-SVM model) - Breaks into maximum, minimum, mean, and variance for all P "row" objects and is used by the "norm"-function.

Learning proceeds by

1. Vectorial Data: loading the dataset as $N \times P$ matrix $\boldsymbol{X}$ composed by P N-component data vectors. (Note that for compatibility reasons the MATLAB functions expect $\boldsymbol{X}^T$ as vectorial input.)

2. Kernel: calculating a kernel matrix which quantifies relations on training data vectors using a kernel function $\boldsymbol{K}_{i,j} = \boldsymbol{k}(\boldsymbol{x}_i, \boldsymbol{x}_j)$

3. Normalization: Gets statistics for all P columns of $\boldsymbol{K}$ and stores its maxima, minima, mean, and a scaling factor in P-dimensional vectors $\boldsymbol{np}$. Normalizes each column of $\boldsymbol{K}$.

4. Calculation of b: The offset b is set to the mean of the L labels represented by the label vector $\boldsymbol{y}$.

5. Sequential Minimal Optimization (SMO): The SMO reads the normalized kernel matrix $\boldsymbol{K}$ and the labels $\boldsymbol{y}$ and calculates a p-dimensional vector $\boldsymbol{\alpha}$, which is sparse in most cases. α_i corresponds to the i-th column of $\boldsymbol{K}$.

3.3 P-SVM for Vectorial Data using Complex Feature Vectors

The P-SVM predictor for vectorial data and complex feature vectors has the form:

$$o_c(\boldsymbol{u}) \;=\; \text{sign}(\text{norm}(\boldsymbol{k}(\boldsymbol{u},\boldsymbol{Z}),\boldsymbol{np}) \cdot \boldsymbol{\alpha} \;+\; b)$$

for classification and

$$o_r(\boldsymbol{u}) \;=\; \text{norm}(\boldsymbol{k}(\boldsymbol{u},\boldsymbol{Z}),\boldsymbol{np}) \cdot \boldsymbol{\alpha} \;+\; b$$

for regression tasks, where

$\boldsymbol{u}$: “column” object which is to be classified (N-component vector)

o_c : P-SVM class prediction of $\boldsymbol{u}$ ($+1$ or -1)

o_r : P-SVM real valued prediction of $\boldsymbol{u}$

$\boldsymbol{Z}$: $N \times P$ matrix composed by P “row” objects (N-component complex feature vectors).

$\boldsymbol{k}()$: calculates a P component vector by applying a kernel function to a “column” object and a set of P “row” objects

$\boldsymbol{\alpha}$: support features (part of the P-SVM model): P-component vector which quantifies the importances of “row” objects for serving as features.

b : bias (part of the P-SVM model)

$\boldsymbol{np}$: normalization statistics (part of the P-SVM model) - Breaks into maximum, minimum, mean, and variance for all P “row” objects and is used by the “norm”-function.

Learning proceeds by

1. Vectorial Data: loading the dataset as $N \times L$ matrix $\boldsymbol{X}$ composed by L N-component data vectors. (Note that for compatibility reasons the MATLAB functions expect $\boldsymbol{X}^T$ as vectorial input.)

2. Complex Feature Vector Matrix: loading the dataset as $N \times P$ matrix $\boldsymbol{Z}$ composed by P N-component data vectors. (Note that for compatibility reasons the MATLAB functions expect $\boldsymbol{Z}^T$ as vectorial input.)

3. Kernel: calculating a kernel matrix which quantifies relations on training data vectors and complex feature vectors using a kernel function $\boldsymbol{K}_{i,j} = \boldsymbol{k}(\boldsymbol{x}_i, \boldsymbol{z}_j)$

4. Normalization: Gets statistics for all P columns of $\boldsymbol{K}$ and stores its maxima, minima, mean, and a scaling factor in P-dimensional vectors $\boldsymbol{np}$. Normalizes each column in $\boldsymbol{K}$.

5. Calculation of b: The offset b is set to the mean of the L labels represented by the label vector $\boldsymbol{y}$.

6. Sequential Minimal Optimization (SMO): The SMO reads the normalized kernel matrix $\boldsymbol{K}$ and the labels $\boldsymbol{y}$ and calculates a p-dimensional vector $\boldsymbol{\alpha}$, which is sparse in most cases. α_i corresponds to the i-th column of $\boldsymbol{K}$.

3.4 Hyperparameters

Let $\boldsymbol{X}$ be the matrix of data vectors in some high-dimensional feature space ϕ, $\boldsymbol{w}$ be the normal vector of a separating hyperplane, y the attributes (binary in case of classification, or real valued in case of regression), and $\boldsymbol{K}$ the kernel matrix. Then the P-SVM "primal" optimization problem has the form

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{\xi}^+,\boldsymbol{\xi}^-} \quad & \frac{1}{2}\,\|\boldsymbol{X}^\top \boldsymbol{w}\|^2 \;+\; C\boldsymbol{1}^\top\left(\boldsymbol{\xi}^+ \;+\; \boldsymbol{\xi}^-\right) \\ \text{s.t.} \quad & \boldsymbol{K}^\top\,\left(\boldsymbol{X}^\top \boldsymbol{w} \;-\; \boldsymbol{y}\right) \;+\; \boldsymbol{\xi}^+ \;+\; \epsilon\,\boldsymbol{1} \;\geq\; \boldsymbol{0} \\ & \boldsymbol{K}^\top\,\left(\boldsymbol{X}^\top \boldsymbol{w} \;-\; \boldsymbol{y}\right) \;-\; \boldsymbol{\xi}^- \;-\; \epsilon\,\boldsymbol{1} \;\leq\; \boldsymbol{0} \\ & \boldsymbol{0} \;\leq\; \boldsymbol{\xi}^+, \boldsymbol{\xi}^- \end{aligned} \tag{1}$$

The parameters C and ϵ correspond to the two different regularization schemes, which have been suggested for the P-SVM method, where ϵ-regularization has been proven more useful for feature selection and the C-regularization for classification or regression problems ([ncpsvm]). $\boldsymbol{\xi}^+$ and $\boldsymbol{\xi}^-$ are the vectors of the slack variables describing violations of the constraints.
The parameter C controls the robustness against outliners by limiting the support vector weights α. If it is infinite, no regularization occurs. If it tends to zero, the largest weights of support vectors decrease and the possibly increase of the training error will be compensated through finding similar data vectors and increasing

their weights (they become support vectors). The number of support vectors increases in order to average over important support vectors (see figure 5).
The second parameter ϵ controls the tolerance level of small training errors. If it tends to infinity, the primal P-SVM problem (eq. 1) is solved without support vectors ($\boldsymbol{\alpha} = 0$). If it tends to zero, the tolerance level decreases and the training error decreases too as far as the number of support vectors increases. That means ϵ controls the tradeoff between a poor representation of the training data and overfitting.

3.5 Comparison P-SVM with libSVM

One of the effects of the P-SVMs primal optimization problem in contrast to the “libSVM” can be easily seen by comparing the support vector placement, when P-SVM and “libSVM” are used for vectorial data with a linear kernel function. The P-SVM finds support vectors in the area around the normal vector of the hyperplane, while the standard SVM finds support vectors around the hyperplane. The effect to the prediction error depends strongly from the data, but comparing figure 5 and 6 even shows a better matching of the P-SVM hyperplane with the optimal hyperplane according to the relying gaussian distribution (crossover of the circles).

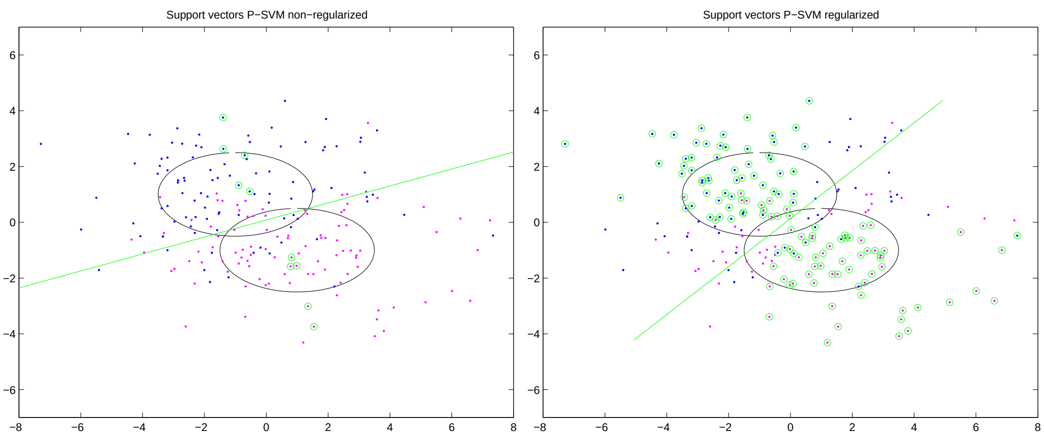

Figure 5: These plots show the data points of a two dimensional gaussian distributed classification problem with two classes and the hyperplane (line) with support vectors (small circles) found by the linear kernel P-SVM. The large circle marks the standard deviation of the gaussian distribution for the two classes. The left plot is for a non-regularized solution ($C = \infty$) and the right shows the effect of regularization ($C = 0.0001$).

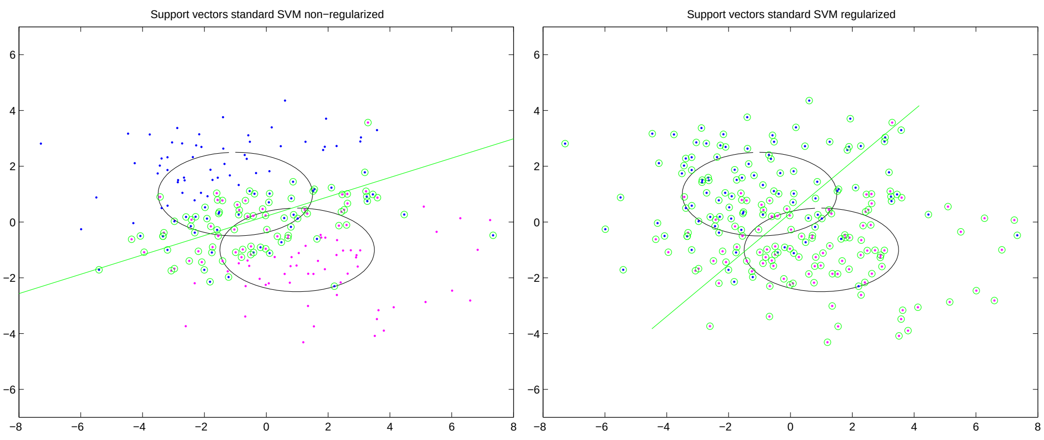

Figure 6: These plots show the same classification problem as used in figure 5 and the hyperplane (line) with support vectors (small circles) found by the linear kernel standard ϵ-SVM. The large circle marks the standard deviation of the gaussian distribution for the two classes. The left plot is for a non-regularized solution ($C = \infty$) and the right shows the effect of regularization ($C = 0.0001$). This experiment is done with the implementation from [libsvm].

References

[ncpsvm] Sepp Hochreiter, Klaus Obermayer; Support Vector Machines for Dyadic Data, *Neural Computation*, 2006.

[psvmsmo] Sepp Hochreiter, Tilman Knebel, Klaus Obermayer; An SMO algorithm for the Potential Support Vector Machine, *Neural Computation*, 2007.

[torch] Ronan Collobert, Samy Bengio, Johnny Mariéthoz; a machine learning library; http://www.torch.ch

[libsvm] C.-C. Chang, C.-J. Lin; LIBSVM; http://www.csie.ntu.edu.tw/˜cjlin/libsvm/