ALBERT-LUDWIGS-UNIVERSITÄT FREIBURG

INSTITUT FÜR INFORMATIK

Lehrstuhl für Mustererkennung und Bildverarbeitung

Fast Support Vector Machine Classification of very large Datasets

Technical Report 2/07

Karina Zapién Arreola[1], Janis Fehr and Hans Burkhardt

March, 2007

[1]now at INSA de Rouen, LITIS
76801 St Etienne du Rouvray, France

Fast Support Vector Machine Classification of very largeDatasets

Karina Zapién Arreola Janis Fehr Hans Burkhardt

March, 2007

Abstract

In many classification applications, Support Vector Machines (SVMs) have proven to be high performing and easy to handle classifiers with very good generalization abilities. However, one drawback of the SVM is its rather high classification complexity which scales linearly with the number of Support Vectors (SV). This is due to the fact that for the classification of one sample one has to evaluate the Kernel-Function with all SVs. To speed up classification, several different approaches have been published, most of them trying to reduce the number of SVs. In our work, which is especially suitable for very large datasets, we follow a different approach: as we will show, it is effectively possible to approximate large SVM problems by decomposing the original problem into linear subproblems where each subproblem can be evaluated in $\Omega(1)$. This approach is especially successful when the assumption holds that the large classification problem can be split into mostly easy subproblems with only a few remaining hard subproblems. For this linear decomposition we introduce a modified numerical optimization process which preserves the maximum margin property. On standard benchmark datasets this approach achieved great speedups while suffering only sightly in terms of classification accuracy and generalization ability. We further extent the method using not only linear, but also non-linear subproblems for the decomposition of the original problem which further increases the classification performance. Additionally we introduce a set of heuristics which allow us to directly control the tradeoff between speedup and accuracy. An implementation of our method is available in [Rea04].

This document is largely based on the master thesis of Karina Zapién Arreola which has been supervised by Janis Fehr and Hans Burkhardt, and the resulting publications at ICPR 2006 [AFB06] and GfKl 2007 [FAB].

Contents

Chapter 1

Introduction

In terms of classification-speed, SVMs [1] are still outperformed by many standard classifiers when it comes to the classification of large problems. For a non-linear kernel function k, the classification function can be written as in Eq. (1.1). Thus, the classification complexity lies in $\Omega(n)$ for a problem with n SVs. However, for linear problems, the classification funct ion has the form of Eq. (1.2), allowing classification in $\Omega(1)$ by calculating the dot product with th e normal vector $\mathbf{w}$ of the hyperplane. In addition, the SVM has the problem that the complexity of a SVM model always scales with the most difficult samples, forcing an increase in Support Vectors. However, we observed that many large scale problems can easily be divided in a large set of rather simple subproblems and only a few difficult ones. Following this assumption, we propose a classification method based on a tree whose nodes consist most ly of linear SVMs.

$$f(\mathbf{x}) = \quad \operatorname{sign} \quad \left(\sum_{i=1}^{m} y_i \alpha_i k \left(\mathbf{x_i}, \mathbf{x} \right) + b \right) \tag{1.1}$$

$$f(\mathbf{x}) = \quad \operatorname{sign} \quad \left(\langle \mathbf{w}, \mathbf{x} \rangle + b \right) \tag{1.2}$$

1.1 Related Work

Speedup in SVM classification time has been approached in several different ways:

Direct reduction of number of SVs. Burges and Schölkopf [BS97] proposed a method to approximate $\mathbf{w}$ by a $\mathbf{w}'$ which can also be expressed by a list of vectors associated with corresponding coefficients α_i. However, the method for determining the reduced set is computationally very expensive. Later, Downs, Gates and Masteres [DGM01] developed a method to identify and discard unnecessary SVs - especially those SVs which linearly depend on other SVs - while leaving the SVM decision unchanged. A reduction in SVs as high as 40.96% has been reported.

Indirect reduction of number of SVs by reducing the size of the QP problem. This method called *RSVM* (Reduced Support Vector Machines) was proposed by Lee [LM04]. It preselects a subset of training samples as support vectors and solves a smaller QP. The authors reported that RSVM needs much less computational time and memory usage than standard SVM. A comparative study on RSVM and SVM by Lin et al. [LL03] showed that standard SVM possesses higher generalization ability, while RSVM may be suitable in very large training problems or those that have a large portion of training samples becoming SVs.

Reduction of the number of vector components. Lei and Govindaraju [LHL05] proposed a reduction of the feature space using principal component analysis (PCA) and Recursive Feature Elimination (RFE). The authors reported a speedup in classification time up to 10 times against a conventional SVM.

Enlarging margins in Perceptron Decision Trees. Bennett et al. [BCSTW00] experimentally proved that inducing large margins into decision trees with linear decision functions improved the generalization ability. Their methods relies on several parameters that have to be tuned in order to achieve satisfactory results.

Wavelet approximation of a SVM. Rätsch et al. [KW05] developed an approximation of a SVM decision function for face classification. This can be achieved by an over-complete Haar wavelet transformation of the raw data using a set of rectangles with constant gray-level values, allowing a very significant speedup. However, this method is only suitable for direct image classification like face classification and does not work for arbitrary feature vectors.

Chapter 2

Definitions

2.0.1 Support Vector Machines

The following discussion will focus on a two-class problem. It will be assumed that the set of features of each sample $\mathbf{x}$ belongs to a Hilbert space denoted by $\mathcal{H}$, which is a vector space with a dot product $\langle x, y \rangle$, with $x, y \in \mathcal{H}$ such that a $norm$ can be induced by $\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$.

2.0.2 Two-Class SVM

Definition 2.1 (Positive and Negative Class) *Let m_1 and m_2 be two natural numbers that fulfill $m = m_1 + m_2$, $m_1 > 0$, $m_2 > 0$ and $\mathscr{C} = \{1, ..., m\}$, without loss of generality we can define:*

Class 1 (Positive Class) *of size m_1, with index $\mathscr{C}_1 = \{1, ..., m_1\}$, conformed by the set $\{\mathbf{x}_i\}, i \in \mathscr{C}_1$, gravity center $\mathbf{s}_1 = \frac{1}{m_1} \sum_{i \in \mathscr{C}_1} \mathbf{x}_i$, $y_i = 1$ for all $i \in \mathscr{C}_1$, and for some later applications, a global penalization value D_1 is defined such that $C_i = D_1 \; \forall i \in \mathscr{C}_1$; C_i represents individual penalization values.*

Class 2 (Negative Class) *of size m_2, with index $\mathscr{C}_2 = \{m_1 + 1, ..., m_1 + m_2\}$, conformed by the set $\{\mathbf{x}_i\}, i \in \mathscr{C}_2$, gravity center $\mathbf{s}_2 = \frac{1}{m_2} \sum_{i \in \mathscr{C}_2} \mathbf{x}_i$, $y_i = -1$ for all $i \in \mathscr{C}_2$, and for some later application, a global penalization value D_2 is assigned to this class such that $C_i = D_2 \; \forall i \in \mathscr{C}_2$; C_i represent individual penalization values.*

Having two classes, we say that they are *linearly separable* if there is a hyperplane of the form $\mathcal{P} : \{\mathbf{x} \in \mathcal{H} | \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\}$, $\mathbf{w} \in \mathcal{H}, b \in \mathbb{R}$ that can perfectly divide the two classes. The vector $\mathbf{w}$ is a vector orthogonal to the hyperplane P and $\langle \mathbf{w}, \mathbf{x} \rangle$ is the length of $\mathbf{x}$ along the direction of $\mathbf{w}$.

We will be interested in finding the $canonical\ hyperplane$ with respect to $\mathbf{x}_i$, $i \in \mathscr{C}$ defined as the hyperplane with the pair $(\mathbf{w}, b) \in \mathcal{H} \times \mathbb{R}$ that is scaled such that

$$\min_{i=1,\dots,m} \|\langle \mathbf{w}, \mathbf{x} \rangle + b\| = 1. \tag{2.1}$$

In other words, the *canonical hyperplane* is the one with minimal distance to the samples equals $\frac{1}{\|\mathbf{w}\|}$.

To illustrate this, let's consider the two-class example depicted in Figure 2.1:

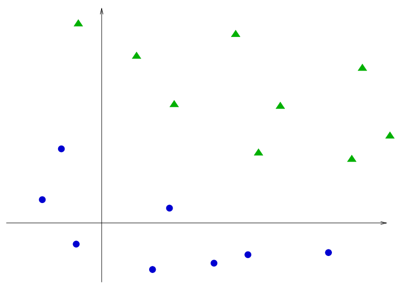


Figure 2.1: Example of a two-class problem

Without loss of generality, let the green triangles represent class 1 ($\mathscr{C}_1$) and the blue circles represent class 2 ($\mathscr{C}_2$). The hyperplanes in Figure 2.2 are all valid functions to divide them.

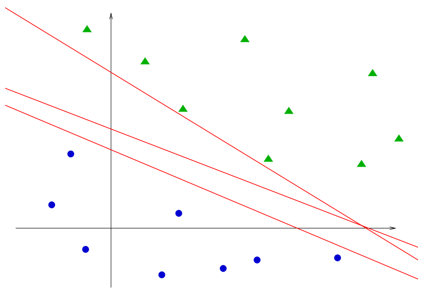


Figure 2.2: Possible dividing hyperplanes for a two-class problem

Figure 2.3 shows a hyperplane with maximal margin for the classification problem in Figure 2.1.

It has to be noticed that for $\mathbf{x}_i, i \in \mathscr{C}_1$ and $\mathbf{x}_j, j \in \mathscr{C}_2$, such that $\langle \mathbf{w}, \mathbf{x}_i \rangle = +1$ and $\langle \mathbf{w}, \mathbf{x}_j \rangle = -1$, we have $\langle \mathbf{w}, (\mathbf{x}_i - \mathbf{x}_j) \rangle = 2$ and therefore

$$\langle \frac{\mathbf{w}}{\|\mathbf{w}\|}, (\mathbf{x}_i - \mathbf{x}_j) \rangle = \frac{2}{\|\mathbf{w}\|} \tag{2.2}$$

With the previous equation, we can conclude that the distance of the closest vector to the hyperplane is $\frac{1}{\|\mathbf{w}\|}$, then, finding the hyperplane with the maximum distance is equivalent to maximize the norm of the orthogonal vector $\mathbf{w}$ that corresponds to the hyperplane that can divide the two classes.

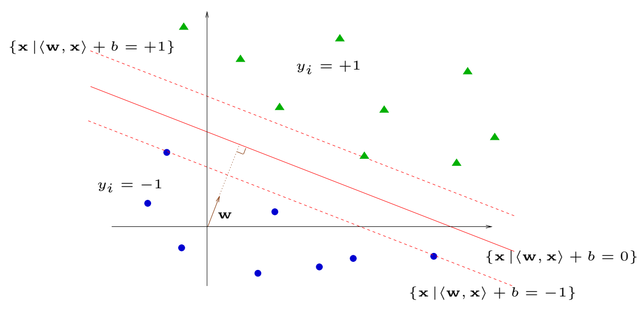

$\{\mathbf{x} \mid \langle \mathbf{w}, \mathbf{x} \rangle + b = +1\}$
$y_i = +1$
$y_i = -1$
$\mathbf{w}$
$\{\mathbf{x} \mid \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\}$
$\{\mathbf{x} \mid \langle \mathbf{w}, \mathbf{x} \rangle + b = -1\}$

Figure 2.3: Hyperplane with maximal margin for a two-class problem

2.0.3 Multi-Class SVM

Originally the SVM is only capable of solving two-class problems, but there are different strategies to extent SVMs to muti-class problemsi [HL01].

One vs. One If there are n classes, $\binom{n}{2}$ binary classifiers are trained pairwise. For classification, vectors are tested in all models giving a probability of belonging to a class. The following is an example of a one-vs.-one classifier.

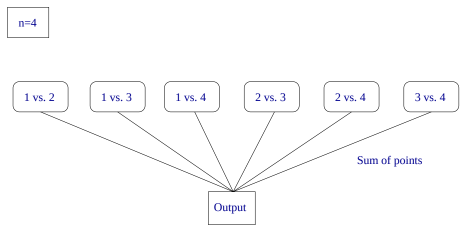


Figure 2.4: One versus one classifier for 4 classes

One vs. Rest If there are n classes, n two-class classifiers a re trained, where one class is differentiated from all the others. New samples are tested in all models and the results are compared. The following is an example of a one-vs.-rest classifier.

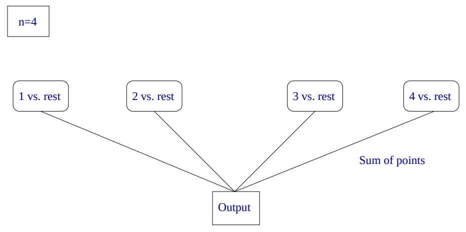


Figure 2.5: One versus rest classifier for 4 classes

2.1 SVM Constrained Optimization Problem

The following problem is a formal definition of the maximal margin hyperplane problem that needs to be solved in the SVM approach.

Problem 2.2 (SVM-Primal Optimization Problem) *Let a class 1 and a class 2 be defined as in Definition 2.1, the optimal margin hyperplane primal problem is defined as follows.*

$$\underset{\mathbf{w}\in\mathcal{H},b\in\mathbb{R}}{\text{minimize}} \quad \tau(\mathbf{w}) = \tfrac{1}{2}\|\mathbf{w}\|^2, \tag{2.3}$$

$$\text{subject to} \quad y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) \geq 1,\ i = 1, .., m, \tag{2.4}$$

And the corresponding decision function would look like

$$f(x) = sign(\langle \mathbf{w}, \mathbf{x}\rangle + b) \tag{2.5}$$

In problem 2.2, $f(x) = \tau(\mathbf{w})$, $\mathcal{E} = \emptyset$ and $\mathcal{I} = \mathscr{C}$. Following [NW99] and as in [SS02], the $Lagrangian$ function can be defined together with the objective function τ and the constraints in 2.4 as follows

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i(y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) - 1). \tag{2.6}$$

One of the KKT conditions states that the gradient of the Lagrangian function must equal zero:

$$\frac{\partial}{\partial b}\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = 0 \qquad \text{and} \tag{2.7}$$

$$\frac{\partial}{\partial \mathbf{w}}\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = 0. \tag{2.8}$$

this leads to

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \qquad \text{and} \tag{2.9}$$

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x_i}. \tag{2.10}$$

The solution vector has thus an expansion 2.10 in terms of a subset of the training patterns, namely those patterns with non-zero α_i, called ***Support Vectors*** (*SVs*). By the KKT conditions,

$$\alpha_i[y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) - 1] = 0 \text{ for all } i = 1, ..., m, \tag{2.11}$$

the SVs lie on the margin. All remaining training examples $(\mathbf{x}, y_i)$ are irrelevant: their constraint $y_j(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) \geq 1$ could just as well be left out, and they do not appear in the expansion. Thus, the hyperplane is completely determined by the patterns closest to it, the solution should not depend on the other examples.

2.1.1 SVM Optimization Problem

By substituting 2.9 and 2.10 into the Lagrangian 2.6, one eliminates the primal variables $\mathbf{w}$ and b, getting the following problem which is solved in practice since it depends only on the sample vectors.

Problem 2.3 (SVM-Dual Optimization Problem) *Let class 1 and class 2 defined as in Definition 2.1, the optimal margin hyperplane dual problem is defined as follows*

$$\underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\text{maximize}} \quad W(\boldsymbol{\alpha}) = \sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i,j=1}^m \alpha_i\alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j\rangle, \tag{2.12}$$

$$\text{subject to} \quad \alpha_i \geq 0,\ i = 1, ..., m, \tag{2.13}$$

$$\sum_{i=1}^m \alpha_i y_i = 0. \tag{2.14}$$

Using 2.10, the hyperplane decision function 2.5 can thus be written as

$$f(\mathbf{x}) = sign\left(\sum_{i=1}^m y_i\alpha_i\langle \mathbf{x_i}, \mathbf{x}\rangle + b\right) \tag{2.15}$$

Figure 2.6 shows a solution for the example in Figure 2.1, the yellow area shows the points in the space that will be labeled as class 1 and the cyan area shows the points that will be labeled as class 2. Vectors with $\alpha_i \neq 0$ are marked.

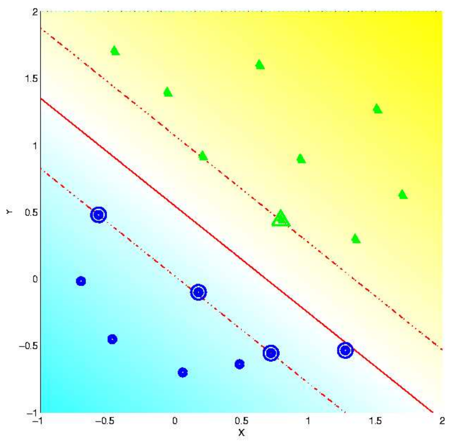


Figure 2.6: Two-class classification problem with linear solution

2.1.2 Soft Margin SVM

Often, the problem can be unfeasible because no linear solution is able to separate the classes properly. For such problems, the C-SV classifier was introduced allowing some mistakes through slack variables with a penalization in the objective function, leading to the following problem:

Problem 2.4 (C-SV Classifier Primal Problem) *For a two-class problem, the primal optimization problem with slack variables is defined as:*

$$\underset{\mathbf{w}\in\mathcal{H},b\in\mathbb{R},\boldsymbol{\xi}\in\mathbb{R}^m}{\text{minimize}} \quad \tau(\mathbf{w},\boldsymbol{\xi}) = \tfrac{1}{2}\|\mathbf{w}\|^2 + \textstyle\sum_{i=1}^m C_i\xi_i, \tag{2.16}$$

$$\text{subject to} \quad y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) \geq 1 - \xi_i,\ i = 1,..,m, \tag{2.17}$$

$$\xi_i \geq 0,\ i = 1,..,m. \tag{2.18}$$

Again using 2.9 and 2.10, the last problem can be converted into the following dual problem.

Problem 2.5 (C-SV Classifier Dual Problem) *In a two-class problem, the optimal margin hyperplane dual problem with slack variables is defined as follows*

$$\underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\text{maximize}} \quad W(\boldsymbol{\alpha}) = \textstyle\sum_{i=1}^m \alpha_i - \tfrac{1}{2}\sum_{i,j=1}^m \alpha_i\alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j\rangle, \tag{2.19}$$

$$\text{subject to} \quad 0 \leq \alpha_i \leq C_i,\ i = 1,...,m, \tag{2.20}$$

$$\textstyle\sum_{i=1}^m \alpha_i y_i = 0. \tag{2.21}$$

For this problem, the decision function remains as in 2.15.
Computation of threshold b: if there exist a solution for Problem 2.3 the hyperplane is placed in the middle of the margin. Nevertheless, for Problem 2.5 the value of the $\boldsymbol{\alpha}$ must be taken into account; the calculation of b can be done as proposed in [KSBM99] as following

$$b = \frac{1}{2}\left(\underset{i\in I_0\cup I_1\cup I_2}{min} \left\{\langle \mathbf{x_i}, \mathbf{w}\rangle\right\} + \underset{i\in I_0\cup I_3\cup I_3}{max} \left\{\langle \mathbf{x_i}, \mathbf{w}\rangle\right\} \right), \tag{2.22}$$

where,

$$I_0 = \{i|0 < \alpha_i < C_i\}, \tag{2.23}$$

$$I_1 = \{i|y_i = 1, \alpha_i = 0\}; \quad I_2 = \{i|y_i = -1, \alpha_i = C_i\}, \tag{2.24}$$

$$I_3 = \{i|y_i = 1, \alpha_i = C_i\}; \quad I_4 = \{i|y_i = -1, \alpha_i = 0\}. \tag{2.25}$$

2.1.3 Non-Linear SVM

Often, no satisfying linear solution can be found. To overcome this problem, feature-vectors are mapped into higher dimensional space $\mathcal{H}$ by the use of some non-linear function

$$\Phi(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathcal{H} \tag{2.26}$$

Φ is chosen in a way such that the classes can be separated in $\mathcal{H}$ by the trivial SVM decision function.

The linear case was developed in a Hilbert space $\mathcal{H}$. In order to make generalizations of this method, the dot product $\langle \mathbf{x}, \mathbf{x'}\rangle$ can be expressed in terms of the kernel k evaluated on input patterns x, x' in a transformed space induced by $\Phi(\mathbf{x}) = x$,

$$k(\mathbf{x}, \mathbf{x'}) = \langle x, x'\rangle = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x'})\rangle. \tag{2.27}$$

This substitution, which is referred to as the ***kernel trick***, is used to extend the method to transformed spaces with nonlinear Support Vector Machines in a new space $\mathcal{H}$ called the *linearization space* because in the new space, the samples are divided with an hyperplane (i.e. a linear function).

The kernel trick can be applied since all feature vectors in 2.15 and 2.19 only occurred in dot products. The vector $\mathbf{w}$ then becomes an expansion in feature space, and therefore will typically no longer correspond to the $\Phi - image$ of a single input space vector. We obtain a decision function of the form

$$\begin{aligned} f(x) &= sign\left(\sum_{i=1}^{m} y_i\alpha_i\langle\Phi(\mathbf{x}),\Phi(\mathbf{x}_i)\rangle + b\right) && (2.28)\\ &= sign\left(\sum_{i=1}^{m} y_i\alpha_i k(\mathbf{x},\mathbf{x}_i) + b\right) && (2.29)\end{aligned}$$

with the threshold b calculated similarly as in 2.22, but considering that now we have applied Φ to the original samples,

$$b = \frac{1}{2}\left(\underset{i\in I_0\cup I_1\cup I_2}{min}\left\{k(\mathbf{x_i},\mathbf{w})\right\} + \underset{i\in I_0\cup I_3\cup I_3}{max}\left\{k(\mathbf{x_i},\mathbf{w})\right\}\right), \qquad (2.30)$$

where the index I_k are defined as in 2.23, 2.24 and 2.25.

The following quadratic problem is the one formulated with the kernel trick

Problem 2.6 (C-SV Kernel Trick in Classifier) *In a two-class problem, the optimal margin hyperplane dual problem in the transformed Hilbert space induced by $k(\mathbf{x},\mathbf{x}')$ (with slack variables) is defined as follows*

$$\begin{aligned} \underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\text{maximize}} \quad & W(\boldsymbol{\alpha}) = \textstyle\sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i,\mathbf{x}_j), && (2.31)\\ \text{subject to} \quad & 0 \le \alpha_i \le C_i,\ i = 1,...,m, && (2.32)\\ & \textstyle\sum_{i=1}^{m}\alpha_i y_i = 0. && (2.33)\end{aligned}$$

The only restriction on kernels is that the eigenvalues have to satisfy $\lambda \geq 0$. Thus, K has to be *semi positive definite*. In general, a K is a kernel if and only if K holds $\sum_{i,j=1}^{m}\alpha_i\alpha_j K(\mathbf{x}_i,\mathbf{x}_j) \geq 0$.

This is stated by Mercer theorem:

Theorem 2.7 (Mercer Theorem) $K(\mathbf{x}_i,\mathbf{x}_j) = \langle\Phi(\mathbf{x}_i),\Phi(\mathbf{x}_j)\rangle$ *iff for arbitrary* $g(\mathbf{x})$ *with* $\int g(\mathbf{x})^2 d\mathbf{x} < \infty$ *holds:*

$$\int K(\mathbf{x}_i,\mathbf{x}_j)g(\mathbf{x}_i)g(\mathbf{x}_j)d\mathbf{x}_i d\mathbf{x}_j \geq 0. \qquad (2.34)$$

For $\mathbf{x} \in \mathbb{R}^n$ there are several proposed kernels [SS02]:

Homogeneous

$$k(\mathbf{x},\mathbf{x}') = \langle\mathbf{x},\mathbf{x}'\rangle \qquad (2.35)$$

Polynomial

$$k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle^d \tag{2.36}$$

Gaussian or Radial Basis Function (RBF)

$$k(\mathbf{x}, \mathbf{x}') = exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right) \tag{2.37}$$

Sigmoid

$$k(\mathbf{x}, \mathbf{x}') = tanh(\kappa\langle \mathbf{x}, \mathbf{x}' \rangle + \vartheta) \tag{2.38}$$

The Gaussian function, also known as Radial Basis Function (RBF), proposed by Boser, Guyon and Vapnik [BGV92], [GBV93] and [Vap95] is normally the first choice because it combines good performance with strong theoretical foundation. In [SS02] it is proven that the RBF-kernel is equivalent to the dot product of elements belonging to an infinite dimensional space. To show the capacity of this trick, let us illustrate it with the example in Figure 2.7. As it can be seen, there does not exist any hyperplane that can perfectly classify all training samples.

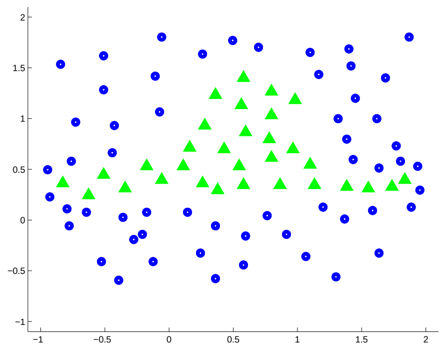


Figure 2.7: Example of a two-class problem with no linear solution

If we try to adjust a straight line to separate this problem, the result seen in Figure 2.8 would be obtained.

The solution to the problem in Figure 2.7 with a Gaussian kernel would look like in Figure 2.9.

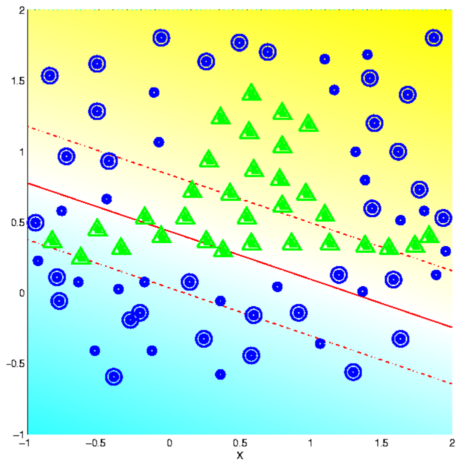


Figure 2.8: Two-class problem with the best adjusted hyperplane

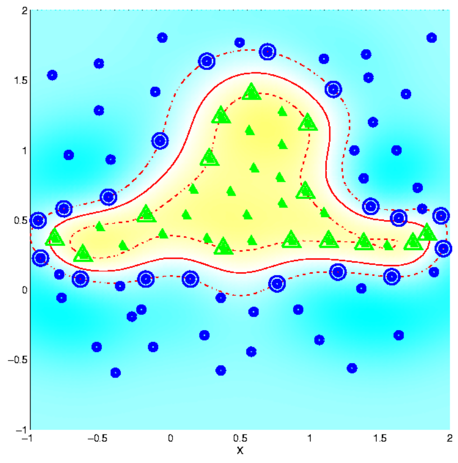


Figure 2.9: Solution for a two-class problem with Gaussian Kernel (Elephant in a snake-solution)

Chapter 3

Support Vector Machines in a Decision Tree

The classification time of a Support Vector Machine (SVM) with a non-linear kernel [BGV92] depends on the number of resulting Support Vectors (SVs) in the model, and in complex models, the number of SVs can be considerably large. If a test dataset is to be classified with such a model, the classification can be very slow.

However, we observed that many large scale problems can easily divided in a majority of rather simple subproblems and only a few difficult ones. Following this assumption, we propose a classification method based on a tree whose nodes consist mostly of linear SVM. This way each node in the decision tree will contain a decision hyperplane, and the classification will depend only on the number of nodes. The classification is then computed with the dot product of a test sample with the orthogonal vector to the corresponding hyperplane of each node.

In Section 3.1 the theoretical basis for this classifier is given, while Section 3.2 describes the proposed algorithm in detail.

3.1 Theoretical Approach

In our work we propose a linear approximation of a continuous function for classification. We assume that the training samples are represented in a Hilbert feature space. This space is divided in regions defined by linear inequalities (hyperplanes). This brings several advantages: one of these is that the transformed space is known since it is the original one; the tuning of parameters can be avoided and the number of hyperplanes needed to linearly approximate the classification function is far less than the number of needed support vectors.

With this aim, a decision tree is built. Each node corresponds to a hyperplane that can classify a specific region. Each hyperplane in the tree is trained in function of the previous hyperplane. For each node, a linear SVM is trained so that the resulting hyperplane is able to identify a region of the feature space where only samples of one class lie.

3.1.1 Zero Solution in SVM

If we have two classes, defined as in Definition 2.1, a SVM usually is trained so that it will make the least possible mistakes in both classes. With the classical approach, the importance of errors in each class can be tuned by adjusting the values of D_1 and D_2 which are the weights for class $\mathscr{C}_1$ and class $\mathscr{C}_2$ respectively.

To achieve this, a first constrained problem is solved in order to find the SVM that classifies perfectly one chosen class, say class $\mathscr{C}_1$, and make the least errors in the other.

In the case that class $\mathscr{C}_1$ is to be perfectly classified, intuitively a big value for D_1 and a very small value for D_2 would be proposed. But it can be faced that the resulting solution of the SVM with these parameters is the zero solution.

The zero solution can occur if the center of gravity if class $\mathscr{C}2$ lies in the convex hull of class $\mathscr{C}_1$ and D_1 is big enough. This follows as consequence of the following general theorem.

Theorem 1 (Zero Solution) *Let class $\mathscr{C}_k$ and class $\mathscr{C}_{\bar{k}}$ be defined similarly as in Definition 2.2. If the convex hull of class $\mathscr{C}_k$ intersects the convex hull of the other class $\mathscr{C}_{\bar{k}}$, then $\mathbf{w} = \mathbf{0}$ is a feasible solution for the primal Problem 2.4 if $D_{\bar{k}} \geq \max_{i \in \mathscr{C}_k}\{\lambda_i\} \cdot D_k$, where λ_i are such that*

$$\mathbf{p} = \sum_{i \in \mathscr{C}_k} \lambda_i \mathbf{x}_i,$$

for a point $\mathbf{p}$ that belongs to both convex hulls.

Proof

It has to be noticed that if $i \in \mathscr{C}_k$ and $j \in \mathscr{C}_{\bar{k}}$ then $y_i \cdot y_j = -1$; similarly, if $i \in \mathscr{C}_k$ and $j \in \mathscr{C}_k$ then $y_i \cdot y_j = 1$. Without loss of generality, let class $\mathscr{C}_k = \mathscr{C}_1$ and class $\mathscr{C}_{\bar{k}} = \mathscr{C}_2$, then the dual problem can be written as follows

$$\begin{array}{ll} \text{maximize} & \sum_{i \in \mathscr{C}_1} \alpha_i + \sum_{i \in \mathscr{C}_2} \alpha_i + \sum_{i \in \mathscr{C}_1, j \in \mathscr{C}_2} \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ & -\frac{1}{2} \sum_{i,j \in \mathscr{C}_1} \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle - \frac{1}{2} \sum_{i,j \in \mathscr{C}_2} \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ & \\ \text{subject to} & \sum_{i \in \mathscr{C}_1} \alpha_i y_i + \sum_{i \in \mathscr{C}_2} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq D_1 \text{ for all } i \in \mathscr{C}_1 \\ & 0 \leq \alpha_j \leq D_2 \text{ for all } j \in \mathscr{C}_2. \end{array}$$

If $\mathbf{p}$ belongs to the convex hull of both classes, then, it can be written as follows

$$\mathbf{p} = \sum_{i \in \mathscr{C}_1} \lambda_i \mathbf{x}_i \quad \text{and} \quad \mathbf{p} = \sum_{j \in \mathscr{C}_2} \lambda_j \mathbf{x}_j,$$

with $\lambda_i \geq 0$ for all $i \in \mathscr{C}_1$, $\sum_{i \in \mathscr{C}_1} \lambda_i = 1$ and $\lambda_j \geq 0$ for all $j \in \mathscr{C}_2$, $\sum_{j \in \mathscr{C}_2} \lambda_j = 1$.

Let $\alpha_i = \lambda_i D_1 \leq D_1$ for all $i \in \mathscr{C}_1$ and $\alpha_j = \lambda_j D_1 \leq \max_{j \in \mathscr{C}_1}\{\lambda_j\} D_1 \leq D_2$ for all $j \in \mathscr{C}_2$, then

$$\begin{aligned}\textstyle\sum_{i\in\mathscr{C}_1}\alpha_i y_i+\sum_{j\in\mathscr{C}_2}\alpha_j y_j &= \textstyle\sum_{i\in\mathscr{C}_1}\lambda_i D_1-\sum_{j\in\mathscr{C}_2}\lambda_j D_1\\ &= \textstyle D_1\sum_{i\in\mathscr{C}_1}\lambda_i - D_1\sum_{j\in\mathscr{C}_2}\lambda_j\\ &= D_1-D_1\\ &= 0\end{aligned}$$

Therefore $\alpha_i = \lambda_i D_1$ for all $i \in \mathscr{C}_1$ and $\alpha_j = \lambda_j D_1$ for all $j \in \mathscr{C}_2$ is a feasible solution for the dual problem. If we calculate the vector $\mathbf{w}$ with these values, we obtain:

$$\begin{aligned}\mathbf{w} &= \textstyle\sum_{i\in\mathscr{C}_1}\alpha_i\mathbf{x}_i y_i+\sum_{j\in\mathscr{C}_2}\alpha_j\mathbf{x}_j y_j\\ &= \textstyle\sum_{i\in\mathscr{C}_1}\lambda_i D_1\mathbf{x}_i-\sum_{j\in\mathscr{C}_2}\lambda_j D_1\mathbf{x}_j\\ &= \textstyle D_1\sum_{i\in\mathscr{C}_1}\lambda_i\mathbf{x}_i - D_1\sum_{j\in\mathscr{C}_2}\lambda_j\mathbf{x}_j\\ &= D_1\mathbf{p}-D_1\mathbf{p}\\ &= \mathbf{0}.\end{aligned}$$

Finally, we conclude that $\mathbf{w} = \mathbf{0}$ is a feasible solution for the primal Problem 2.2.

□

Bennett and Bredensteiner [BCSTW00] proved that Problem 2.2, SVM, has also another dual problem that can be seen in terms of the convex hull.

When a solution exists, a geometric interpretation of Problem 2.2 can be reduced to the problem of finding the two closest points of the two convex hulls and then, finding the line segment between the two points. Finally, the orthogonal plane to the line segment that bisects it, is chosen to be the separating plane (see Figure 3.1).

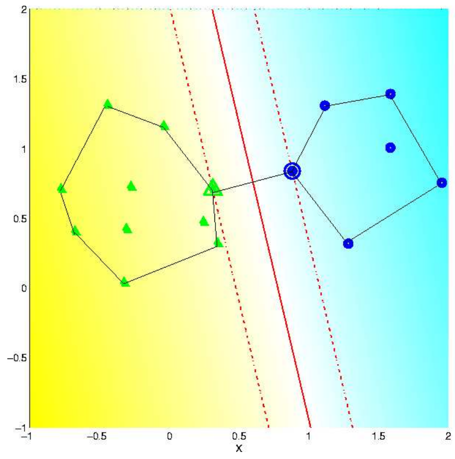


Figure 3.1: Convex Hull interpretation for the SVM solution

But a solution not always exists for Problem 2.2, therefore, Problem 2.4 was introduced allowing some mistakes.

Under this approach, Problem 2.4, C-SV Classifier, leads to the geometric interpretation of finding the closest points of the two reduced convex hulls according to D_1 and D_2 (see Figure 3.2).

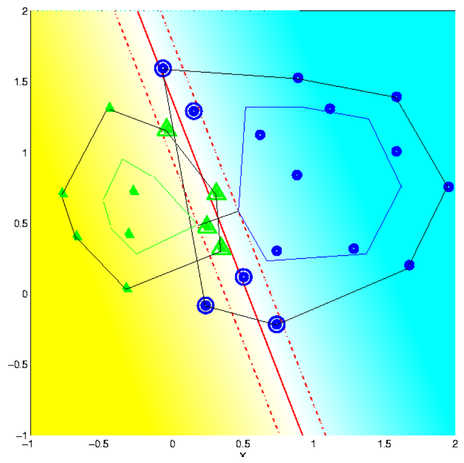


Figure 3.2: Reduced Convex Hull interpretation for the C-SVM solution

This reduced convex hull will be still around the center of gravity of the original convex hull. And in the case that the center of gravity of class $\mathscr{C}_2$ is inside the convex hull of class $\mathscr{C}_1$, if D_1 is big enough, the zero solution will be still a feasible solution. This is resumed in the following corollary.

Corollary 1 (Zero Solution with Gravity Center) *If the center of gravity of class $\mathscr{C}_2$, $\mathbf{s}_2$, is inside of the convex hull of class $\mathscr{C}_1$, then, it can be represented as*

$$\mathbf{s}_2 = \sum_{i \in \mathscr{C}_1} \lambda_i \mathbf{x}_i \quad \textit{and} \quad \mathbf{s}_2 = \sum_{j \in \mathscr{C}_2} \frac{1}{m_2} \mathbf{x}_j$$

with $\lambda_i \geq 0$ for all $i \in \mathscr{C}_1$ and $\sum_{i \in \mathscr{C}_1} \lambda_i = 1$.

If additionally $D_1 \geq \lambda_{max} D_2 m_2$, where $\lambda_{max} = \max_{i \in \mathscr{C}_1}\{\lambda_i\}$, then $\mathbf{w} = \mathbf{0}$ is a feasible solution for the primal Problem 2.2.

Proof

Let class $\mathscr{C}_1$ and class $\mathscr{C}_2$ be as in Definition 2.1, then the dual problem can be written as follows

$$\begin{array}{ll}
\text{maximize} & \sum_{i\in\mathscr{C}_1}\alpha_i+\sum_{i\in\mathscr{C}_2}\alpha_i+\sum_{i\in\mathscr{C}_1,j\in\mathscr{C}_2}\alpha_i\alpha_j\langle\mathbf{x}_i,\mathbf{x}_j\rangle,\\
& -\frac{1}{2}\sum_{i,j\in\mathscr{C}_1}\alpha_i\alpha_j\langle\mathbf{x}_i,\mathbf{x}_j\rangle-\frac{1}{2}\sum_{i,j\in\mathscr{C}_2}\alpha_i\alpha_j\langle\mathbf{x}_i,\mathbf{x}_j\rangle,\\
\\
\text{subject to} & \sum_{i\in\mathscr{C}_1}\alpha_i y_i+\sum_{i\in\mathscr{C}_2}\alpha_i y_i=0,\\
& 0\le\alpha_i\le D_1 \text{ for all } i\in\mathscr{C}_1,\\
& 0\le\alpha_j\le D_2 \text{ for all } j\in\mathscr{C}_2,
\end{array}$$

Let $\alpha_i=\lambda_i D_2 m_2\le\lambda_{max}D_2m_2\le D_1$ for all $i\in\mathscr{C}_1$ and $\alpha_j=D_2$ for all $j\in\mathscr{C}_2$, then,

$$\begin{array}{rcl}
\sum_{i\in\mathscr{C}_1}\alpha_i y_i+\sum_{j\in\mathscr{C}_2}\alpha_j y_j & = & \sum_{i\in\mathscr{C}_1}\lambda_i D_2 m_2-\sum_{j\in\mathscr{C}_2}D_2\\
& = & D_2m_2\sum_{i\in\mathscr{C}_1}\lambda_i-D_2m_2\\
& = & D_2m_2-D_2m_2\\
& = & 0
\end{array}$$

Therefore $\alpha_i=\lambda_iD_2m_2$ for all $i\in\mathscr{C}_1$ and $\alpha_j=D_2$ for all $j\in\mathscr{C}_2$ is a feasible solution for the dual problem.
If we calculate the vector $\mathbf{w}$ with these values, we obtain:

$$\begin{array}{rcl}
\mathbf{w} & = & \sum_{i\in\mathscr{C}_1}\alpha_i\mathbf{x}_i y_i+\sum_{j\in\mathscr{C}_2}\alpha_j\mathbf{x}_j y_j\\
& = & \sum_{i\in\mathscr{C}_1}\lambda_iD_2m_2\mathbf{x}_i-\sum_{j\in\mathscr{C}_2}D_2\mathbf{x}_j\\
& = & D_2m_2\sum_{i\in\mathscr{C}_1}\lambda_i\mathbf{x}_i-D_2m_2\mathbf{s}_2\\
& = & D_2m_2\mathbf{s}_2-D_2m_2\mathbf{s}_2\\
& = & \mathbf{0}.
\end{array}$$

Finally, we conclude that $\mathbf{w}=\mathbf{0}$ is a feasible solution for the primal Problem 2.2.

$\square$

Any reduced convex hull produced by a C-SV problem will still contain the gravity center of the class. For the case where the gravity center of class $\mathscr{C}_{\bar{k}}$ is in the convex hull of class $\mathscr{C}_k$ (see Figure 3.3) and if a non-degenerate solution wants to be found, the convex hull of class $\mathscr{C}_k$ must also be reduced enough (i.e. D_k must decrease). The resulting hyperplane can then be adjusted with the threshold parameter to have no errors in the hard class.

3.1.2 Reduction of Possibility of the Zero Solution

In order to reduce the classification time, the SVM with non-linear kernel will be substitute with a decision tree of linear support vector machines. The tree will first target an area in the feature space that can be clearly assigned to class $\{\bar{k}\}\in\{2,1\}$ by a linear classifier. This will be achieved by finding the hyperplane with the widest margin

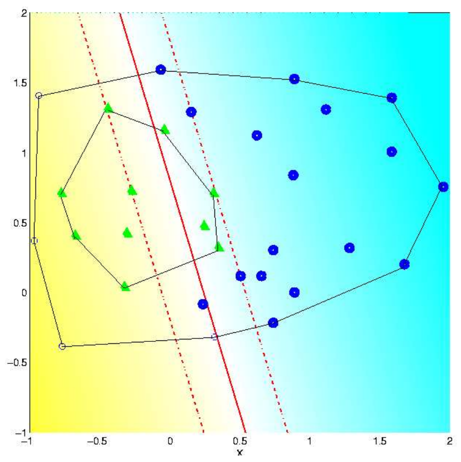


Figure 3.3: Reduced Convex Hull interpretation for the C-SVM solution

that made no errors in class $k = \{1, 2\}$ and that makes the least possible mistakes in class $\bar{k}$.

In order to decrease the number of trivial solutions (zero vector) that are in the approach, the following new problem will be introduced:

Problem 1 (H1-SVM: Hard Margin for 1 class (Primal Prob.)) *Let 2 classes be defined as in Definition 2.1, we will be interested on solving the following problem:*

$$\underset{\mathbf{w}\in\mathcal{H}, b\in\mathbb{R}}{\text{minimize}} \quad \tau(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i\in\mathscr{C}_{\bar{k}}} y_i(\langle\mathbf{x}_i, \mathbf{w}\rangle + b), \tag{3.1}$$

$$\text{subject to} \quad y_i(\langle\mathbf{x}_i, \mathbf{w}\rangle + b) \geq 1 \textit{ for all } i \in \mathscr{C}_k, \tag{3.2}$$

where $k = 1$ and $\bar{k} = 2$, or $k = 2$ and $\bar{k} = 1$.

Analyzing this problem more precisely, it can be seen that the feasible solution of this optimization problem is the one that classifies correctly all the samples in class k (because $y_i(\langle\mathbf{w}, \mathbf{x}_i\rangle + b) \geq 1$ for all $i \in \mathscr{C}_k$ is a constrain) with no slack variables. On the other hand, from all the vectors that satisfy this condition, the search vector is the one that has a balance between the size of the margin and the number of misclassified samples of class $\mathscr{C}_\mathbf{k}$. As before, this problem can be transformed into the following dual problem which is a special case of the original problem where all the α_k for one class are equal to one.

Problem 1 (H1-SVM: Hard Margin for 1 class (Dual Prob.)) *Let the two classes be* *defined as in Definition 2.1. The H1-SVM for 1 class is based on the following problem*

$$\begin{aligned} \underset{\boldsymbol{\alpha} \in \mathbb{R}^m}{\text{maximize}} \quad & W(\boldsymbol{\alpha}) = \textstyle\sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle, && (3.3)\\ \text{subject to} \quad & 0 \le \alpha_i \le C_i,\ i \in \mathscr{C}_k, && (3.4)\\ & \alpha_j = 1,\ j \in \mathscr{C}_{\bar{k}}, && (3.5)\\ & \textstyle\sum_{i=1}^m \alpha_i y_i = 0, && (3.6) \end{aligned}$$

where $k = 1$ *and* $\bar{k} = 2$*, or* $k = 2$ *and* $\bar{k} = 1$.

With this new definition of the SVM problem, the zero solution can only occur with a linear combination of the vector samples of the hard class. Without loss of generality, if the hard class is class $\mathscr{C}_1$

$$\begin{aligned} \mathbf{w} &= \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i && (3.7)\\ &= \sum_{i \in \mathscr{C}_1} \alpha_i \mathbf{x}_i - \sum_{i \in \mathscr{C}_2} \alpha_i \mathbf{x}_i && (3.8)\\ &= \sum_{i \in \mathscr{C}_1} \alpha_i \mathbf{x}_i - \sum_{i \in \mathscr{C}_2} \mathbf{x}_i && (3.9)\\ & && (3.10) \end{aligned}$$

if we define $\mathbf{z}_i = \sum_{i \in \mathscr{C}_2} \mathbf{x}_i$ and $|\mathscr{C}_1| \ge (n-1) = dim(\mathbf{z}_i) - 1$, then, there exist $\{\alpha_i\}, i \in \mathscr{C}_1, \alpha_i \ne 0$ such that

$$\mathbf{w} = \sum_{i \in \mathscr{C}_1} \alpha_i \mathbf{x}_i - \mathbf{z}_i = \mathbf{0}. \qquad (3.11)$$

So, the number of zero solutions that are feasible in the H1-SVM Problem 1 is a subset (strictly smaller) than the number of zero solutions in the original C-SVM Problem 2.5.

3.2 Description of the Algorithm

The aim is to build a tree which nodes are SVMs. At each step, a region defined by a hyperplane is labeled with a class until the whole space is labeled.

To illustrate this and the further description of the algorithm, let us consider the example in Figure 3.4.

This example can be found at the dataset web-page of the LIBSVM c++ library [CL05a] under the name of *Fourclass*. The problem has 2 features and therefore it can be represented in 2D. Class $\mathscr{C}_1$ is represented with green triangles and class $\mathscr{C}_2$ is represented with blue circles.

This example has clearly a non-linear solution, so a SVM with Gaussian Kernel was used. The graphical representation of the solution found is depicted in Figure 3.5.

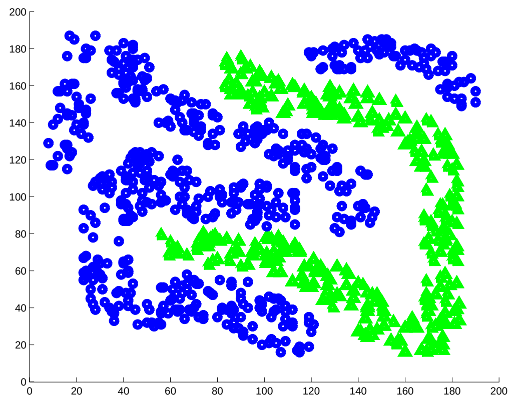


Figure 3.4: Fourclass example [CL05a]

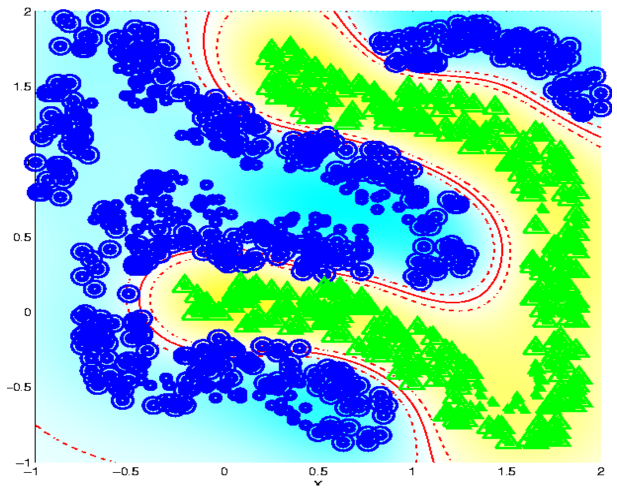


Figure 3.5: Solution for the fourclass with a SVM (Gaussian kernel)

The classification function corresponding to the found hyperplane in the transformed space is marked with a solid red line, the existing margin between the two classes can be seen with the spotted red lines. The thicker points are the needed support vectors for the classification. As can be seen, these are a big percent of the training data, therefore a large evaluation time for classification new points is needed.

3.2.1 Decision Tree with Linear SVM Nodes

Fast classification of a dataset is achived by the construction of a decision tree whose nodes are hyperplanes obtained with the training of a support vector machine with linear kernel.

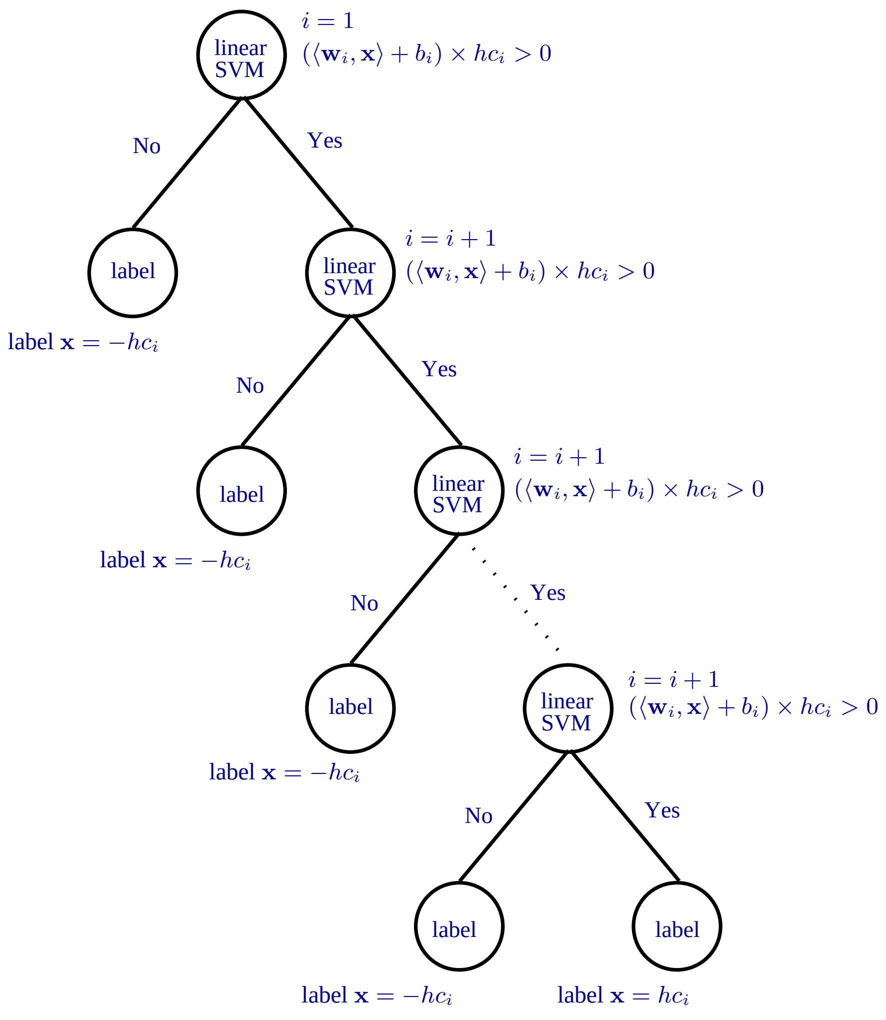

linear SVM
$i = 1$
$(\langle \mathbf{w}_i, \mathbf{x} \rangle + b_i) \times hc_i > 0$
No
Yes
label
label $\mathbf{x} = -hc_i$
linear SVM
$i = i + 1$
$(\langle \mathbf{w}_i, \mathbf{x} \rangle + b_i) \times hc_i > 0$
No
Yes
label
label $\mathbf{x} = -hc_i$
linear SVM
$i = i + 1$
$(\langle \mathbf{w}_i, \mathbf{x} \rangle + b_i) \times hc_i > 0$
No
Yes
label
label $\mathbf{x} = -hc_i$
linear SVM
$i = i + 1$
$(\langle \mathbf{w}_i, \mathbf{x} \rangle + b_i) \times hc_i > 0$
No
Yes
label
label
label $\mathbf{x} = -hc_i$
label $\mathbf{x} = hc_i$

Figure 3.6: Decision tree with linear SVM

To obtain the decision tree, at each step a ***hard class*** $\mathscr{C}_k$ is chosen (in a greedy way, see Chapter 4). Then a SVM is trained so that the resulting hyperplane will correctly distinguish all points belonging to class $\mathscr{C}_k$, thus all the samples $\mathbf{x}_i, i \in \mathscr{C}_k$ will lie on one side of the hyperplane and all points on the other side of the plane will belong to the non-hard class $\mathscr{C}_{\bar{k}}$. The number of samples is then reduced by leaving out the training samples of class $\mathscr{C}_{\bar{k}}$ that were correctly classified with this SVM. This process is repeated with the reduced problem until the samples left belong all to the same class.

The classification then takes places by identifying at each node if the sample belongs to the non-hard class 2 being labeled with it, or keeping with the evaluation to the next node. This is depicted in the diagram 3.6.

In the fourclass example, the class 1 (green triangles) is the hard class at the first step, the line (hyperplane) obtained by solving Problem 1 with hard class 1 will look like in Figure 3.7.

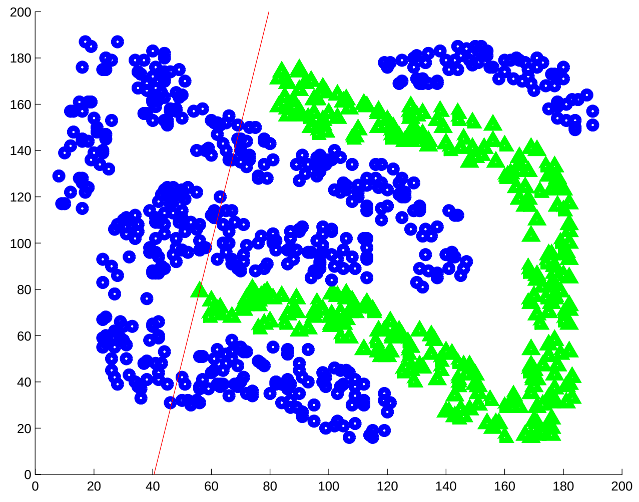


Figure 3.7: First hyperplane for Problem 1 for *fourclass* (hard class = triangles)

On Figure 3.8 the region which has exclusively samples belonging to the non-hard class is depicted in cyan and it all will be labeled as class 2.

Next, the problem is reduced by leaving out the samples that lie in the previously marked region. For the *fourclass* example, the new problem to solve is the one in Figure 3.9.

This procedure is repeated stepwise with the new sample-space marking the "safe" areas (i.e. areas where samples of only 1 class were found) as non-hard class.

Figures 3.10, 3.11 and 3.12 show which hyperplane is found by the algorithm at each step of the tree. A region in cyan represents the solution of a QP with the positive class (green triangles) as the hard class, thus all the elements in the cyan region are labeled as negative samples. Similarly, A region in yellow represents the solution of a QP with the negative class (blue squares) as the hard class, thus all the elements in the yellow region are labeled as positive samples.

Each time a hyperplane is chosen, the samples belonging the the non-hard class are removed and the QP for the remaining samples is solved. The space can be stepwise labeled by considering the region that is on the side of the hyperplane where only samples belonging to the non-hard class were found.

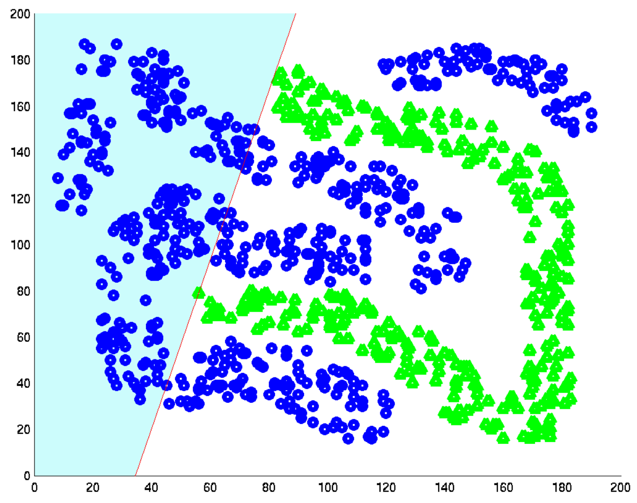


Figure 3.8: First labeling after resolution of Problem 1 for *fourclass*

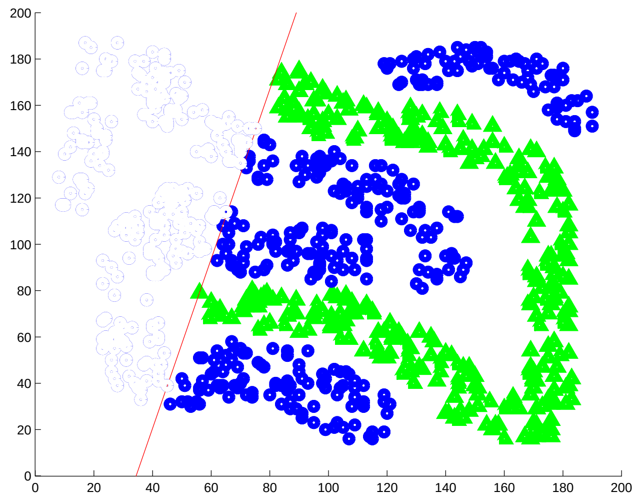
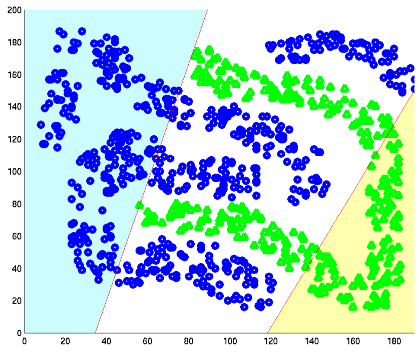


Figure 3.9: Reduced problem for next classification step

At each step, the algorithm chooses the hyperplane that can reduce the problem most, therefore it can happen that the same class is chosen as the hard class for consecutive nodes in the decision tree.


(a) Second plane

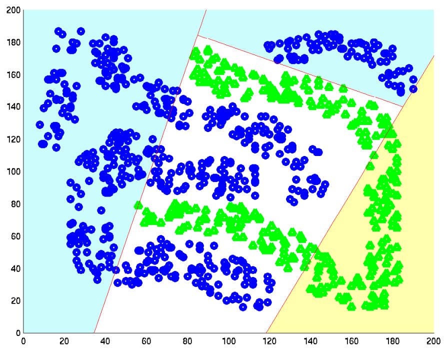
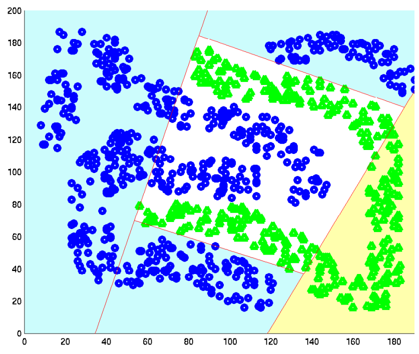

(b) Third Plane

Figure 3.10: Second and third plane for the *fourclass* problem


(a) Fourth plane

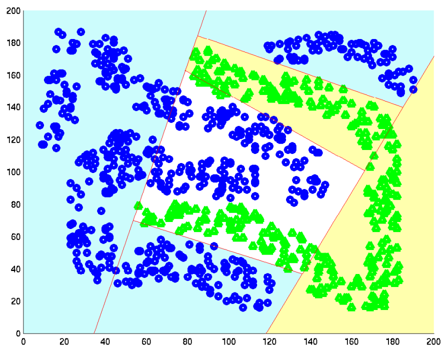
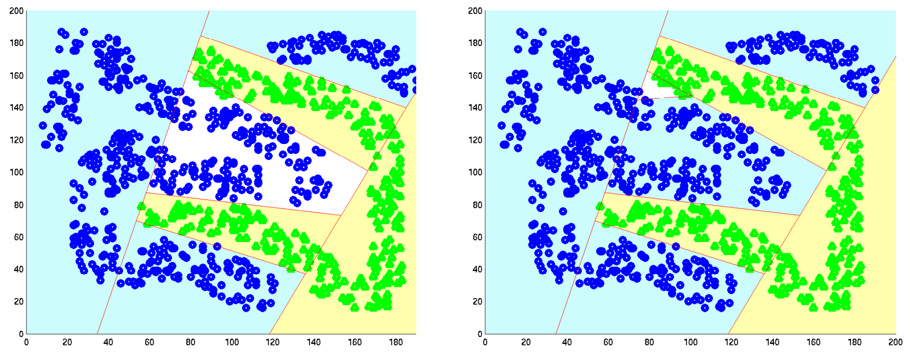

(b) Fifth Plane

Figure 3.11: Fourth and fifth plane for the *fourclass* problem



(a) Sixth plane (b) Seventh Plane

Figure 3.12: Sixth and seventh plane for the *fourclass* problem

By repeating this procedure, new regions are labeled until the remaining samples belong all to the same class. The algorithm will label the whole remaining region containing these samples with the class they belong.

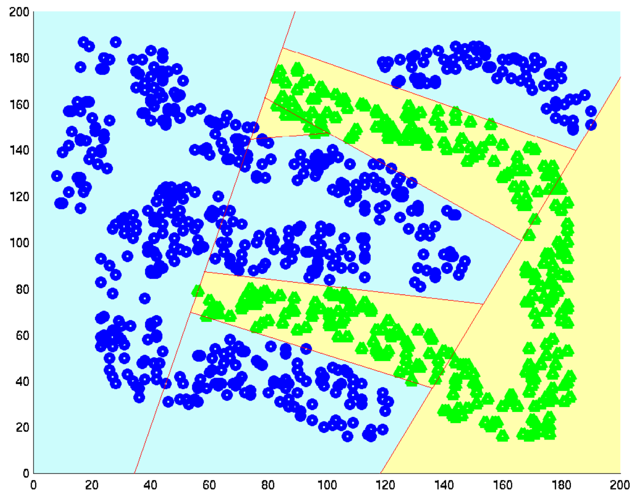


Figure 3.13: Final solution for the fourclass problem

Picture 3.13 depicts the final solution of the algorithm for the fourclass problem and how the space was divided according to the decision tree.

3.2.2 Search for the Best Hyperplane

As seen in the theoretical approach, two QP problems are considered. One is the approach given by Boser, Guyon and Vapnik [BGV92] with a large penalization value for the hard class. The second one is the new approach where a hard class is defined and the objective is to find a hyperplane with the maximum margin and the least possible mistakes on the non-hard class.

Once the QP Problem 2.5 or Problem 1 have been solved, the direction of the orthogonal vector can be calculated as.

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i. \tag{3.12}$$

The rest is to find the threshold b of this hyperplane, equivalent to the intersection with the axes of it.

It has to be taken into consideration that the searched hyperplane is one that makes no mistakes in the hard class and makes the less possible mistakes in the other class. The usual way of calculating the threshold in 2.22, cannot be longer used to define the hyperplane. Instead, the threshold is calculated by assigning to b_1 the minimum (maximum) value of $\langle \mathbf{w}, \mathbf{x}_i \rangle$ for all $i \in \mathscr{C}_1$ ($i \in \mathscr{C}_2$) and then, for those samples belonging to the non-hard class that are correctly classified, the maximum (minimum) value of $\langle \mathbf{w}, \mathbf{x}_i \rangle$ for all $i \in \mathscr{C}_2$ ($i \in \mathscr{C}_1$) is assigned to b_2 and the threshold is set to $b = \frac{1}{2}(b_1 + b_2)$. That is,

for hard class = 1

$$b = \frac{min_{i \in \mathscr{C}_1} \langle \mathbf{w}, \mathbf{x}_i \rangle + max_{\{j \in \mathscr{C}_2 \wedge \langle \mathbf{w}, \mathbf{x}_j \rangle < 0\}} \langle \mathbf{w}, \mathbf{x}_j \rangle}{2} \tag{3.13}$$

for hard class = -1

$$b = \frac{max_{j \in \mathscr{C}_2} \langle \mathbf{w}, \mathbf{x}_j \rangle + min_{\{i \in \mathscr{C}_1 \wedge \langle \mathbf{w}, \mathbf{x}_i \rangle > 0\}} \langle \mathbf{w}, \mathbf{x}_i \rangle}{2} \tag{3.14}$$

In Figure 3.14 the calculation of the threshold is depicted. For this example, the hard class is the positive class (the green triangles). An orthogonal vector $\mathbf{w}$ is given, the green hyperplane is the one with b_1 as threshold; this has the characteristic that all samples in class 1 are correctly classified, except for the ones that $\langle \mathbf{w}, \mathbf{x_i} \rangle = b_1$, $i \in \mathscr{C}_1$. From this threshold, the nearest hyperplane is searched such that the least possible errors in class $\mathscr{C}_2$ is obtained. This is represented with the blue hyperplane with threshold b_2. Finally, b is calculated as the average of these two values.

The proposed solution for Problem 2.7 can be seen in Figure 3.15, as usual, the yellow area represents the positive class and the cyan area represents the negative class with the assigned probability accordingly to the hyperplane that is classifying that area.

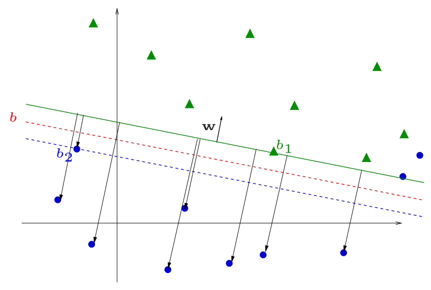


Figure 3.14: Search of threshold b for non-linear problem

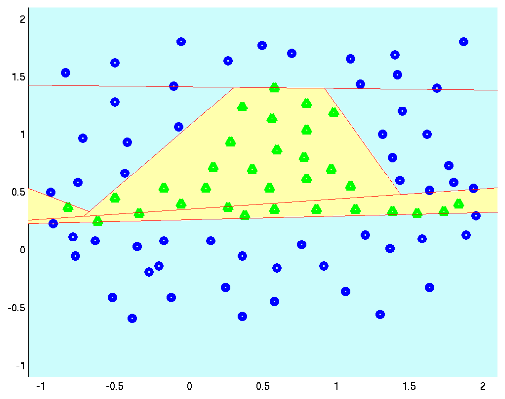


Figure 3.15: Final solution for problem in Figure 2.7 (Mexican Hat solution)

3.3 Non-Linear Extension

In order to classify a sample, one simply runs it down the SVM-tree. When using only linear nodes, we already obtained good results but we also observed that first of all, most errors occur in the last node, and second, that over all only a few samples will reach the last node during the classification procedure. This motivated us to add a non-linear node (e.g. using RBF kernels) to the end of the tree.

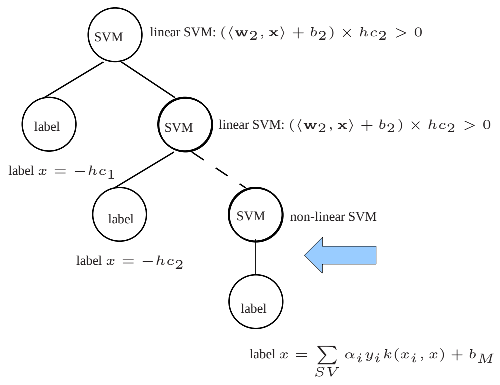

SVM
linear SVM: $((\langle \mathbf{w}_2, \mathbf{x}\rangle + b_2) \times hc_2 > 0$
label
SVM
linear SVM: $((\langle \mathbf{w}_2, \mathbf{x}\rangle + b_2) \times hc_2 > 0$
label $x = -hc_1$
label
SVM
non-linear SVM
label $x = -hc_2$
label
label $x = \sum_{SV} \alpha_i y_i k(x_i, x) + b_M$

Figure 3.16: SVM tree with non-linear extesion

Training of the extended SVM-tree is analog to the original case. First a pure linear tree is build. Then we use a heuristic (tradeoff between average classification depth and accuracy) to move the final, non-linear node from the last node up the tree. It is very important to notice that the final non-linear SVM has to be trained on the entire initial training set and not only on the samples remaining after the last linear node. Other wise the final node is very likely to suffer from strong overfitting. This way the final model will have many SVs but since only a few samples will reach the final node the average classification depth will hardly be effected as our experiments showed.

Chapter 4

Implementation Details

For the implementation, the library LibSVMTL [Rea04] was modified. This is a highly customizable C++ Support Vector Machine library based in the one designed by Chang and Lin [CL05a].

This chapter summarizes specific details about the implemented algorithms. The construction of the decision tree is straightforward; therefore, we focus on solving the resolution of the optimization problem and in obtaining the hyperplane in each node of the tree.

Section 4.1 deals with the QP problem and its resolution. Section 4.2 is focused on obtaining the parameters $\mathbf{w}$ and b to define the hyperplane. Finally, Section 4.3 shows the additional heuristics that were used.

4.1 Quadratic Problem

Each node in the decision tree consists of a hyperplane. A hyperplane can be defined with a vector $\mathbf{w}$ orthogonal to it and an offset b. The problem of finding the hyperplane with maximum margin and no errors in one class leads to an optimization problem (Problem 2.4 or 1) which dual problem (Problem 2.5 or 1, respectively) results to be a QP problem with the form:

Problem 1 (Constrained Optimization Problem)

$$\begin{array}{rc} \underset{\boldsymbol{\alpha}}{\text{minimize}} & f(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T Q \boldsymbol{\alpha} - \mathbf{e}^T \boldsymbol{\alpha} \quad , \\ \text{subject to} & L_i \leq \alpha_i \leq C_i, \\ & \mathbf{y}^T \boldsymbol{\alpha} = 0. \end{array}$$

If the problem has m samples, Q is a $m \times m$ matrix containing the kernel function applied to each pair of samples: $Q_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$; $\mathbf{e}$ is the vector of ones with length m; L_i and C_i are the lower and upper bound, respectively, for α_i. Finally, $\mathbf{y} = (y_1, ..., y_m)^T$ is the vector containing the labels for samples $\mathbf{x}_i$.

Finding an optimum in the dual space, is equivalent to finding an optimum in the primal space. The solution of these QP problems, contains the optimal values for the dual variables α_i. With the help of the KKT conditions 2.10, the vector $\mathbf{w}$ can be later calculated by:

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i, \tag{4.1}$$

while the bias b can be fixed according to the stated problem.

For the original approach of SVM (Problem 2.5) where $L_i = 0$, if we use the form or Problem 4.1, several simplifications to the resolution method (solver) could be applied to speed it up. Therefore, some small adjustments had to be implemented in the code of the LIBSVM in order to solve problems with $L_i \neq 0$. This algorithm can be found in [CL05b] under the name of ***Algorithm 1***.

For the resolution of the *Two-variable QP Subproblem*, the next algorithm was implemented. This consist only on a slight variation of the algorithm implemented by Fan [FCL05], Chang and Lin [CL05b], where the lower bound is allowed to be different from zero.

Algorithm 1 (SMO-iteration solution)

```
Used variables:
Q[i][j] = kernel evaluation of sample i, x[i] and sample j, x[j]
y[i] = label for sample x[i]
alpha = array of size m
G[i] = i-th element of the gradient of the objective function
L_i = lower bound for alpha[i]
C_i = upper bound for alpha[i]

if(y[i]!=y[j]) {
    delta = (-G[i]-G[j])/max{Q[i][i]+Q[j][j]+2*Q[i][j],0}
    diff = alpha[i] - alpha[j]
    alpha[i] += delta
    alpha[j] += delta

    if(diff > 0) {
        if(alpha[j] < L_j) {
            alpha[j] = L_j
            alpha[i] = diff + L_j } }
  else {
        if(alpha[i] < L_i) {
            alpha[i] = L_i
            alpha[j] = -diff + L_i } }
    if(diff > C_i - C_j) {
        if(alpha[i] > C_i) {
            alpha[i] = C_i
            alpha[j] = C_i - diff } }
    else {
        if(alpha[j] > C_j) {
            alpha[j] = C_j
            alpha[i] = C_j + diff } } }
else {
    delta = (G[i]-G[j])/max{Q[i][i]+Q[j][j]-2*Q[i][j],0}
    sum = alpha[i] + alpha[j]
    alpha[i] -= delta
    alpha[j] += delta
    if(sum > C_i) {
        if(alpha[i] > C_i) {
        alpha[i] = C_i
        alpha[j] = sum - C_i } }
else {
        if(alpha[j] < L_j) {
            alpha[j] = L_j
            alpha[i] = sum - L_j } }
if(sum > C_j) {
        if(alpha[j] > C_j) {
            alpha[j] = C_j
            alpha[i] = sum - C_j } }
    else {
        if(alpha[i] < L_i) {
            alpha[i] = L_i
            alpha[j] = sum - L_i } } }
```

4.1.1 QP Speed-up Techniques

Two techniques were used to improve the resolution time of the QP problem.

The first one, ***shrinking***, was proposed in [Joa98]. This technique is used since for many problems the number of free vectors (i.e. where $L_i < \alpha_i < C_i$) is small. The shrinking technique reduces the size of the working problem without considering some bounded variables. Near the end of the iterative process, the possible set A, where all final free α_i may reside in, is identified.

The other method used to reduce the computational time is the ***caching***. The elements of Q_{ij} are calculated as needed since Q is fully dense and may fit in the computer memory. Only the recently used Q_{ij} are stored. Hence, the computational cost of later iterations can be reduced.

This two methods were not modified from the original version in the LIBSVM library and are explained in detail in [Joa98] and [CL05b].

4.2 Obtaining the Decision Hyperplane

The aim of solving the QP problem is to go back to the primal Problem 2.4 and obtain the orthogonal vector $\mathbf{w}$ and the bias b for the corresponding node in the decision tree.

With this, a decision function with the form $sign(\langle \mathbf{w}, \mathbf{x} \rangle + b)$ can be used to classify new samples.

4.2.1 Orthogonal Vector

In the *two-class* Definition 2.1, the hard class will be denoted as $\mathscr{C}_k$ and the non-hard class as $\mathscr{C}_{\bar{k}}$. Two different problems are solved with $(k = 1, \bar{k} = 2)$ and later with $(k = 2, \bar{k} = 1)$.

The first problem solved is based directly on Problem 2.5, where a very large cost is given to the errors on the hard class and a standard low cost is given to the non-hard class, in this way, the hyperplane will avoid misclassification in the hard class:

$$\begin{aligned} \underset{\boldsymbol{\alpha} \in \mathbb{R}^m}{\text{maximize}} \quad & W(\boldsymbol{\alpha}) = \textstyle\sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle, \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C_k, \; i \in \mathscr{C}_k \\ & 0 \leq \alpha_j \leq 1, \; j \in \mathscr{C}_{\bar{k}} \\ & \textstyle\sum_{i=1}^m \alpha_i y_i = 0. \end{aligned} \tag{4.2}$$

where C_K is a value large enough so that the resulting hyperplane will classify the samples in class $\mathscr{C}_k$ correctly.

The initial solution for this problem is set to $\alpha_i = 0$ for all $i \in \mathscr{C}$.

The next problem is based on the new approach proposed in Problem 1. This problem is explicitly formulated so that the feasible solutions are the ones that does not allow any misclassification in the hard class (in numerical terms, this is equivalent to assigning a very large cost on the errors in the hard class) and the hyperplane is then adjusted to do the least possible errors in the other class. The quadratic problem takes the next form:

$$\begin{aligned} \underset{\boldsymbol{\alpha} \in \mathbb{R}^m}{\text{maximize}} \quad & W(\boldsymbol{\alpha}) = \textstyle\sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle, \\ \text{subject to} \quad & 0 \leq \alpha_i, \; i \in \mathscr{C}_k \\ & \alpha_j = 1, \; j \in \mathscr{C}_{\bar{k}} \\ & \textstyle\sum_{i=1}^m \alpha_i y_i = 0. \end{aligned} \tag{4.3}$$

The initial solution for this problem is $\alpha_i = 0$ for all $i \in \mathscr{C}_k$ and $\alpha_j = 1$ for all $j \in \mathscr{C}_{\bar{k}}$.

The solution of these problems will give the values for $\boldsymbol{\alpha}$ on the optimal point. To obtain the orthogonal vector $\mathbf{w}$, the KKT Condition 2.10 is used:

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x_i}.$$

After solving these problems, four possible solutions for $\mathbf{w}$ are obtained, two are the solutions of Problem 4.2 after solving the problem with $k = 1$ and then for $k = 2$.

The other two are the solutions obtained from Problem 4.3, again, after solving the problem for $k = 1$ and then for $k = 2$.

Each hyperplane reduces the problem by removing the samples of class $\bar{k}$ that were correctly classified. The hyperplane that reduces the problem the most is finally assigned to the next node in the decision tree with the appropriate threshold b. For the classification step, all samples $\mathbf{x}$ that have not been classified in a previous node, and that satisfy $y_{\bar{k}}(\langle \mathbf{w}, \mathbf{x}\rangle + b) > 0$ will be assigned to class $\bar{k}$ at this node of the tree. A reduced problem is stated with the unlabeled samples.

The algorithm iteratively finds hyperplanes for the reduced problem and builds with these the tree.

4.2.2 Threshold

The resolution of the QP problem is a vector $\mathbf{w}$ orthogonal to a hyperplane. The threshold b is then calculated as in Equation 2.22:

$$b = \frac{1}{2}\left(\underset{i \in I_0 \cup I_1 \cup I_2}{min} \left\{ \langle \mathbf{x_i}, \mathbf{w} \rangle \right\} + \underset{i \in I_0 \cup I_3 \cup I_3}{max} \left\{ \langle \mathbf{x_i}, \mathbf{w} \rangle \right\} \right), \tag{4.4}$$

with $I_0 = \{i | 0 < \alpha_i < C_i\}$; $I_1 = \{i | y_i = 1, \alpha_i = 0\}$; $I_2 = \{i | y_i = -1, \alpha_i = C_i\}$; $I_3 = \{i | y_i = 1, \alpha_i = C_i\}$; $I_4 = \{i | y_i = -1, \alpha_i = 0\}$.

This cannot be used for the new implementation since this is calculated considering a margin of error in both classes [KSBM99] and [KG02]. Our aim is to find a threshold that correctly classifies all the samples in the hard class and minimizes the number of misclassified samples in the non-hard class. The following algorithm was implemented to calculate b:

Algorithm 1 (Pseudo-code for Calculation of threshold) *Calculation of the threshold b for the problem in Equations 4.2 or 4.3.*

```
m = number of samples
n = feature space size
x[i] = feature vector of sample i
y[i] = label of sample i
hc = hard class (1 or -1)
w = orthogonal vector to the found hyperplane

ub = INF
lb = -INF
for  i=0 to m {
    if ((y[i])*hc > 0) {
        yG = x[i]' * w

        if(y[i] > 0)
            ub = min{ub,yG}
        else
            lb = max{lb,yG} } }
```

```
if (ub != INF)
    r1 = ub
else
    r1 = lb

ub = INF
lb = -INF
for  i=0 to m  {
    if ((y[i])*hc < 0) {
        yG = (x[i]’ * w) - r1

        if ( (y[i]*yG) > 0) {
            yG = yG + r1
            if(y[i] > 0)
                ub = min{ub,yG}
            else
                lb = max{lb,yG} } } }

if (ub != INF)
    r2 = ub
else if (lb !=-INF)
    r2 = lb
else
    r2 = r1

r=(r1+r2)/2

return r
```

4.3 Heuristics Used

Even though the optimization function is a quadratic function, numeric problems, speed-up techniques and semi-positive definite matrices can mislead the algorithm towards finding the global optimum. Several heuristics were implemented to assure the convergence of algorithm.

4.3.1 Greedy Heuristic

At each step, Problem 2.5 and Problem 1 are solved for both cases: first, using class $\mathscr{C}_1$ as the hard class and then class $\mathscr{C}_2$. The number of vectors that can be left out by using each of these 4 solutions is counted. The selected hyperplane is the one that reduces the problem the most. This heuristic is illustrated in Figure 4.1.

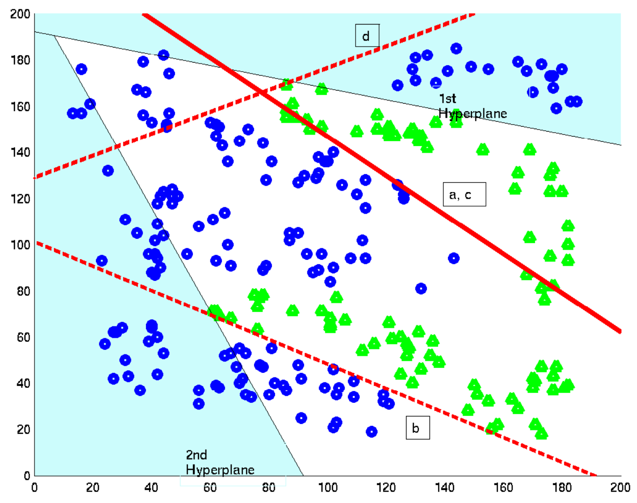


Figure 4.1: Greedy heuristic. The problem to be solved is defined with the samples in the white area (triangles = class 1, circles = class -1). The possible solutions for the next step are the red lines a, b, c and d. These were obtained with the method {C-SVM, hard class=-1}, {C-SVM, hard class=1}, {H1-SVM, hard class=-1} and {H1-SVM, hard class=1}, respectively (a and c resulted the same hyperplane). Line a will be added to the next node since it can correctly classify the most number of samples in the non-hard class (class -1).

4.3.2 Avoiding the Zero Solution

As seen in Corollary 1, the zero solution can result if the cost of the hard class is significantly bigger than the cost of the non-hard class. One method to avoid obtaining trivial solutions, is to reduce the upper bound, C_k, for the alphas in the hard class $\mathscr{C}_k$, is reduced until a solution different to the trivial one is found. If $\|\mathbf{w}\| < tol$, then C_k is adjusted as follows:

$$C_k = C_k/f, \tag{4.5}$$

where tol is a number close to zero that states the tolerance of the norm of vector $\mathbf{w}$ and the factor $f > 1$.

Another method to overcome the problem of the degenerated solution, is to set all $\alpha_i = 1$ for $i \in \mathscr{C}_{\bar{k}}$ as described in Problem 1.

4.3.3 Change of Sign of $\mathbf{w}$

In several cases, the given hyperplane in not able to reduce the problem. In this case, if no sample could be left out, then, $-\mathbf{w}$ is used instead. This is equivalent to changing the inequality direction for the classification. This is illustrated in Figure 4.2

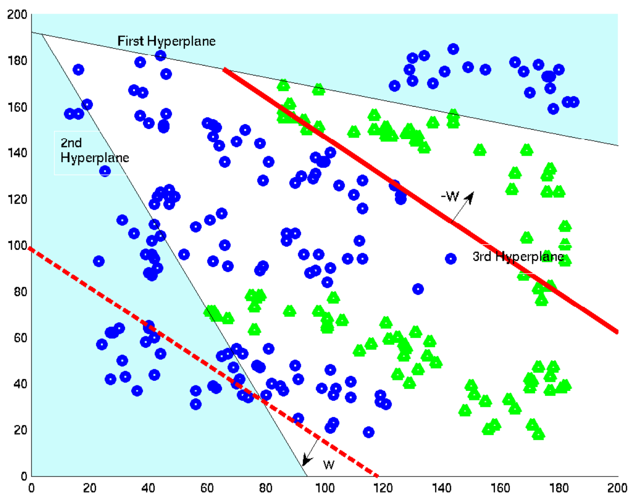


Figure 4.2: Change of sign of $\mathbf{w}$. The direction of vector $\mathbf{w}$ points towards the positive class. The line using this parameter (dashed line) can not classify correctly any sample in the non-hard class (positive class, represented with triangles). If the inequality is changed, this line can reduce the problem for the next iteration.

4.3.4 Perpendicular Hyperplanes

In the case were the given hyperplane cannot reduce the problem, it was observed that the hyperplane was oriented in the direction of the distribution of the samples. In such cases, some of the orthogonal hyperplanes could reduce the problem.

The use of these perpendicular hyperplanes in the decision tree –in a greedy way– increased the classification rate and the generalization ability. Experimentally, it was observed that this heuristic was not frequently used. It was used when the morphology of the problem had to be changed to be able to go further (i.e. when the algorithm got stuck).

An additional degree of greediness was implemented with this heuristic. This consists on having the option of considering also the orthogonal hyperplanes together with the original hyperplane that result from the QP problem and not only when it get stuck. Again the chosen hyperplane is the one that reduced the problem the most.

This heuristic is illustrated in Figure 4.3.

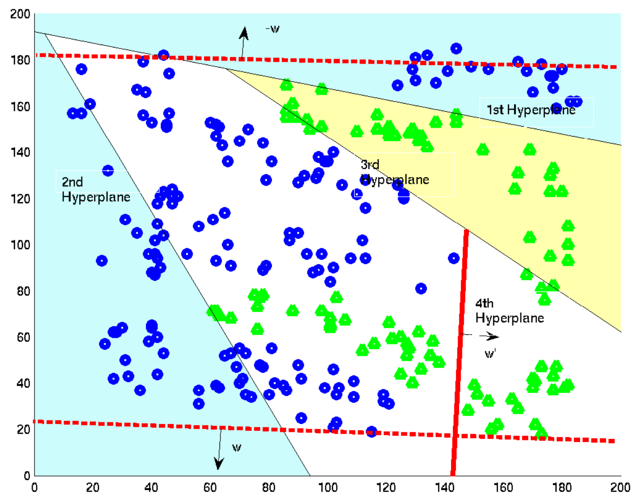


Figure 4.3: Searching perpendicular hyperplanes to $\mathbf{w}$. The direction of vector $\mathbf{w}$ points towards the positive class. The possible solutions (dashed lines) can not classify correctly any sample in the non-hard class (positive class, represented with triangles) for both inequalities. Instead, a perpendicular line (with $\mathbf{w}'$) is used.

4.3.5 Reduction of Useless Hyperplanes (Pruning)

The algorithm stops building the tree after all the samples have been left out (or, when all the remaining samples belong to the same class). At the end of the algorithm, several hyperplanes are useless in the classification since later hyperplanes were more general than these. If some hyperplanes deeper in the decision tree are used before, previous ones could be left aside. The size of the tree would be reduced (and therefore the classification time). Figure 4.4 shows an example of this case.

An algorithm was implemented to “clean” the set $\{(\mathbf{w}_i, b_i)\}$, where each $(\mathbf{w}_i, b_i)$ corresponds to the hyperplane at node i in the decision tree. The accuracy in the training set is measured by classifying it without a specific node (starting from the last one). If the accuracy does not decrease, the node is removed from the tree. An extension of this technique could be done by allowing a degree of error.

Algorithm 1 (Decision Tree Pruning) *Implemented algorithm to prune the obtained decision tree.*

```
Used variables:
w = two-dimensional array, w[i] contains the hyperplane in node i
rho[i] = threshold for node i
hard_class_vector[i] = hard class for node i
reached_errors = number of errors with the original decision
```

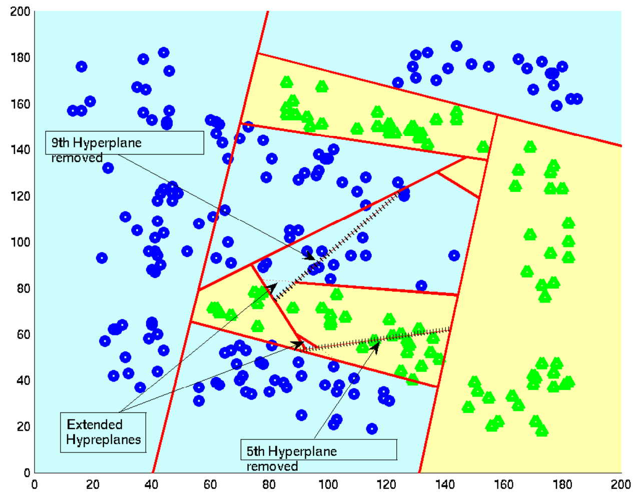


Figure 4.4: Pruning, useless lines are removed from the tree. The classification in the space will be slightly changed, normally generalized

```
                 tree, without reduction
x[j] = sample j
y[j] = label for sample j
Classify(w, rho, hard_class_vector, x) = function that classifies
    sample x with the decision tree that can be formed with:
    w, rho and hard_class_vector
function erase(i) = erase element i of the vector
function insert(i,obj) = insert obj at position i

w_temp= w[0]
rho_temp = 0
hcv_temp = 0

for (int i=w.size - 1; i>=0 ; i--) {
    errors = 0

    w_temp = w[i]
    rho_temp = rho[i]
    hcv_temp = hard_class_vector[i]

    w.erase(i)
    rho.erase(i)
    hard_class_vector.erase(i)
    for (int j=0; j<prob->l ; j++) {
      x_class = Classify(w,rho,hard_class_vector,x[j])
      if ( y[j]*x_class <=0 )
          errors++ }
    if ( errors > (reached_errors) ) {
        w.insert(i,w_temp)
        rho.insert(i,rho_temp)
        hard_class_vector.insert(i,hcv_temp) }
    if (w.size()==1)
        break }
```

To get an idea of how these heuristic were applied, the *fourclass* Problem [TKH96] was solved. Table 4.1 shows for each hyperplane in the decision tree if the hyperplane was obtained by solving the C-SVM (c) of the H1-SVM (h) Problem; if the sign of the inequality was changed which class was the hard class and if a perpendicular plane had to be used at any point.

Line Number	Solver used	Hard Class	Sign of $\mathbf{w}$	Use of a perpendicular hyperplane
1	h	-1	1	no
2	c	1	1	no
3	h	1	1	no
4	h	1	1	yes
5	h	1	1	no
6	h	-1	1	yes
7	c	-1	1	no
8	h	-1	1	no
9	h	-1	1	no
10	h	1	1	yes
11	c	1	1	no
12	c	1	1	no

Table 4.1: Used heuristics for the *fourclass* example

It can be observed that the H1-SVM could find in several occasions better hyperplanes than the C-SVM. The sign of $\mathbf{w}$ was never changed and in just a few cases where the algorithm got stuck, the used of perpendicular hyperplanes was needed.

After the pruning process, line number 9 and 5 were removed, thus, the final tree had depth 10.

Chapter 5

Experiments and Comparisons

5.1 Verification of the Approach

In order to show the validity and classification accuracy of our algorithm we performed a series of experiments on standard benchmark data-sets. In this series of experiments the data was split into training and test sets and normalized to minimum and maximum feature values (Min-Max) or standard deviation (Std-Dev). We used One-Vs-One multi-class algorithm.

Tables for each example are presented with the number of features of each dataset, the number of training and testing samples used, the number of require SVs or hyperplanes, depending on the method; training and classification time[1] (hh:mm:ss.00); finally the classification accuracy is shown.

Speedup comparison with similar works is difficult to state since most publications (see related work) used datasets with less than 1000 samples, where the training and testing time are negligible compared to the size of out datasets.

5.1.1 DNA Dataset

This dataset contains features of a DNA sequence [BJ]. Splice junctions are points on a DNA sequence at which 'superfluous' DNA is removed during the process of protein creation in higher organisms. The problem posed in this dataset is to recognize, given a sequence of DNA, the boundaries between exons (the parts of the DNA sequence retained after splicing) and introns (the parts of the DNA sequence that are spliced out). This problem consists of two subtasks: recognizing exon/intron boundaries (referred to as EI sites), and recognizing intron/exon boundaries (IE sites). (In the biological community, IE borders are referred to a "acceptors" while EI borders are referred to as "donors".)

[1]These experiments were run in a computer with a P4, 2.8 GHz and 1G in Ram.

From the original dataset, two sets were randomly created with the same proportion of elements of each class. One third of the the observations were used for the training set and the rest as testing set.

DNA (Min-Max)	RBF Kernel	H1-SVM g1=0	H1-SVM g1=1	RBF/H1 g1=0	RBF/H1 g1=1
Nr. Features	180	180	180		
Nr. Train Samples	1330	1330	1330		
Nr. SVs or Hyperplanes	798	3	3	266	266
Training Time	00:02.35	00:01.84	00:03.74	1.28	0.63
Nr. Test Samples	1446	1446	1446		
Classif. Accuracy	1354	1305	1305	1.04	1.04
Classification Time	00:06.70	00:01.86	00:01.81	***3.6***	***3.7***
Classif. Accurancy %	93.64 %	90.25 %	90.25 %	1.04	1.04

Table 5.1: Results for the DNA dataset with the Min-Max normalization method

DNA (Std-Dev)	RBF Kernel	H1-SVM g1=0	H1-SVM g1=1	RBF/H1 g1=0	RBF/H1 g1=1
Nr. Features	180	180	180		
Nr. Train Samples	1330	1330	1330		
Nr. SVs or Hyperplanes	881	3	3	293.67	293.67
Training Time	00:02.62	00:02.11	00:03.82	1.24	0.69
Nr. Test Samples	1446	1446	1446		
Classif. Accuracy	1351	1315	1315	1.03	1.03
Classification Time	00:06.78	00:01.85	00:01.71	***3.66***	***3.96***
Classif. Accurancy %	93.43 %	90.94 %	90.94 %	1.03	1.03

Table 5.2: Results for the DNA dataset with the Std-Dev normalization method

It can be observed that with the H1-SVM the classification could be done almost 4 times faster than with the C-SVM.

5.1.2 Faces Dataset

This dataset containing faces and non-faces images can be found in Peter Carbonetto's homepage [Car]. The objective is to determinate if the image is a face or not. From the original dataset, two sets were randomly created with the same proportion of elements of each class. Two thirds of the the observations were used for the training set and the rest as testing set.

Faces (Std-Dev)	RBF Kernel	H1-SVM g1=0	H1-SVM g1=1	RBF/H1 g1=0	RBF/H1 g1=1
Nr. Features	576	576	576		
Nr. Train Samples	9172	9172	9172		
Nr. SVs or Hyperplanes	1902	9	9	211.33	211.33
Training Time	31:53.67	43:59.77	47:46.43	0.72	0.67
Nr. Test Samples	4262	4262	4262		
Classif. Accuracy	4148	3926	3926	1.06	1.06
Classification Time	03:05.80	00:13.55	00:14.51	***13.71***	***12.8***
Classif. Accurancy %	97.33 %	92.12 %	92.12 %	1.06	1.06

Table 5.3: Results for the Faces dataset with the Std-Dev normalization method

Faces (Min-Max)	RBF Kernel	H1-SVM g1=0	H1-SVM g1=1	RBF/H1 g1=0	RBF/H1 g1=1
Nr. Features	576	576	576		
Nr. Train Samples	9172	9172	9172		
Nr. SVs or Hyperplanes	2206	4	4	551.5	551.5
Training Time	14:55.23	10:55.70	14:21.99	1.37	1.04
Nr. Test Samples	4262	4262	4262		
Classif. Accuracy	4082	3879	3879	1.05	1.05
Classification Time	03:13.60	00:14.73	00:14.63	***13.14***	***13.23***
Classif. Accurancy %	95.78 %	91.01 %	91.01 %	1.05	1.05

Table 5.4: Results for the Faces dataset with the Min-Max normalization method

With the new algorithm, the classification time was improved more than 10 times, although, the algorithm to solve the QP problem for the training still has to be optimized to solve linear problems.

5.1.3 Fourclass Dataset

This is a 2D Dataset from the Proceedings of the 13th International Conference on Pattern Recognition, Vienna, Austria [TKH96]. The test and the training samples were randomly generated; one third of the population became training samples

Fourclass (Std-Dev)	RBF Kernel	H1-SVM g1=0	H1-SVM g1=1	RBF/H1 g1=0	RBF/H1 g1=1
Nr. Features	2	2	2		
Nr. Train Samples	287	287	287		
Nr. SVs or Hyperplanes	135	7	5	19.29	27
Training Time	00:00.30	00:00.09	00:00.11	3.33	2.73
Nr. Test Samples	618	618	618		
Classif. Accuracy	538	600	593	0.9	0.91
Classification Time	00:00.18	00:00.05	00:00.07	***3.6***	***2.57***
Classif. Accurancy %	87.06 %	97.09 %	95.95 %	0.9	0.91

Table 5.5: Results for the Fourclass dataset with the Std-Dev normalization method

Fourclass (Min-Max)	RBF Kernel	H1-SVM g1=0	H1-SVM g1=1	RBF/H1 g1=0	RBF/H1 g1=1
Nr. Features	2	2	2		
Nr. Train Samples	287	287	287		
Nr. SVs or Hyperplanes	150	16	8	9.38	18.75
Training Time	00:00.10	00:00.18	00:00.11	0.56	0.91
Nr. Test Samples	618	618	618		
Classif. Accuracy	498	573	596	0.87	0.84
Classification Time	00:00.08	00:00.05	00:00.05	***1.6***	***1.6***
Classif. Accurancy %	80.58 %	92.72 %	96.44 %	0.87	0.84

Table 5.6: Results for the Fourclass dataset with the Min-Max normalization method

The classification time was considerably decreased (12 times) while the classification correctness was improved.

5.1.4 Isolet Dataset

Dataset for Isolated Letter Speech Recognition [Rep]. The test and the training samples were randomly generated two third of the population was testing samples

Isolet (Std-Dev)	RBF Kernel	H1-SVM g1=0	H1-SVM g1=1	RBF/H1 g1=0	RBF/H1 g1=1
Nr. Features	617	617	617		
Nr. Train Samples	155950	155950	155950		
Nr. SVs or Hyperplanes	35340	344	344	102.73	102.73
Training Time	07:13.75	18:51.98	1:04:11.3	0.38	0.11
Nr. Test Samples	1559	1559	1559		
Classif. Accuracy	1499	1472	1472	1.02	1.02
Classification Time	03:01.99	00:32.85	00:36.43	***5.54***	***5***
Classif. Accurancy %	96.15 %	94.42 %	94.42 %	1.02	1.02

Table 5.7: Results for the Isolet dataset with the Std-Dev normalization method

Isolet (Min-Max)	RBF Kernel	H1-SVM g1=0	H1-SVM g1=1	RBF/H1 g1=0	RBF/H1 g1=1
Nr. Features	617	617	617		
Nr. Train Samples	155950	155950	155950		
Nr. SVs or Hyperplanes	22932	325	325	70.56	70.56
Training Time	12:46.70	06:14.40	52:47.43	2.05	0.24
Nr. Test Samples	1559	1559	1559		
Classif. Accuracy	1493	1496	1496	1	1
Classification Time	03:16.56	00:39.92	00:24.37	***4.92***	***8.07***
Classif. Accurancy %	95.77 %	95.96 %	95.96 %	1	1

Table 5.8: Results for the Isolet dataset with the Min-Max normalization method

The classification time could be improved up to factor 8. Important is that the classification accuracy of a SVM with RBF-kernel could be reached.

5.1.5 USPS Dataset

The USPS data is a database for handwritten text recognition research [Hul94], the training set contains 2007 examples and the test set contains 7291 examples as provided.

USPS (Std-Dev)	RBF Kernel	H1-SVM g1=0	H1-SVM g1=1	RBF/H1 g1=0	RBF/H1 g1=1
Nr. Features	256	256	256		
Nr. Train Samples	18063	18063	18063		
Nr. SVs or Hyperplanes	4522	102	99	44.33	45.68
Training Time	00:27.52	00:35.26	02:37.48	0.78	0.17
Nr. Test Samples	7291	7291	7291		
Classif. Accuracy	7030	6798	6816	1.03	1.03
Classification Time	02:07.23	00:29.75	00:17.22	***4.28***	***7.39***
Classif. Accurancy %	96.42 %	93.24 %	93.49 %	1.03	1.03

Table 5.9: Results for the Usps dataset with the Std-Dev normalization method

USPS (Min-Max)	RBF Kernel	H1-SVM g1=0	H1-SVM g1=1	RBF/H1 g1=0	RBF/H1 g1=1
Nr. Features	256	256	256		
Nr. Train Samples	18063	18063	18063		
Nr. SVs or Hyperplanes	3597	49	49	73.41	73.41
Training Time	00:44.74	00:22.70	02:09.58	1.97	0.35
Nr. Test Samples	7291	7291	7291		
Classif. Accuracy	6986	6836	6836	1.02	1.02
Classification Time	01:58.59	00:19.99	00:20.07	***5.93***	***5.91***
Classif. Accurancy %	95.82 %	93.76 %	93.76 %	1.02	1.02

Table 5.10: Results for the Usps dataset with the Min-Max normalization method

As shown in the table, the classification time could be reduced between factor 4 to 7.

5.2 Classification of very large Datasets

Due to the lack of available very large datasets (we consider datasets with several million samples to be very large), we performed our experiments mainly on our own database of cell nuclei features. This data holds the initial assumption, that a large classification problem can be split into mainly easy and only a few hard subproblems.

5.2.1 Data

The experiments were performed on 3D volumetric data samples of chicken embryo chorioallantoic membrane (CAM) probes recorded by a confocal laser scanning microscope (LSM). The CAM is a widely used model for angiogenesis research at cellular level. An automatic localization and identification of the different cell types is crucial. Understanding angiogenesis has been found to be th e key to treatment of many frequent diseases, including cancer and heart ischemia. The samples were prepared as described in [K$^+$02] and treated with YoPro-1 and SMACy3 fluorescent markers and recorded in two channels. Fig. 5.1 shows a typical xy-slice of the YoPro channel with most frequent cell types.

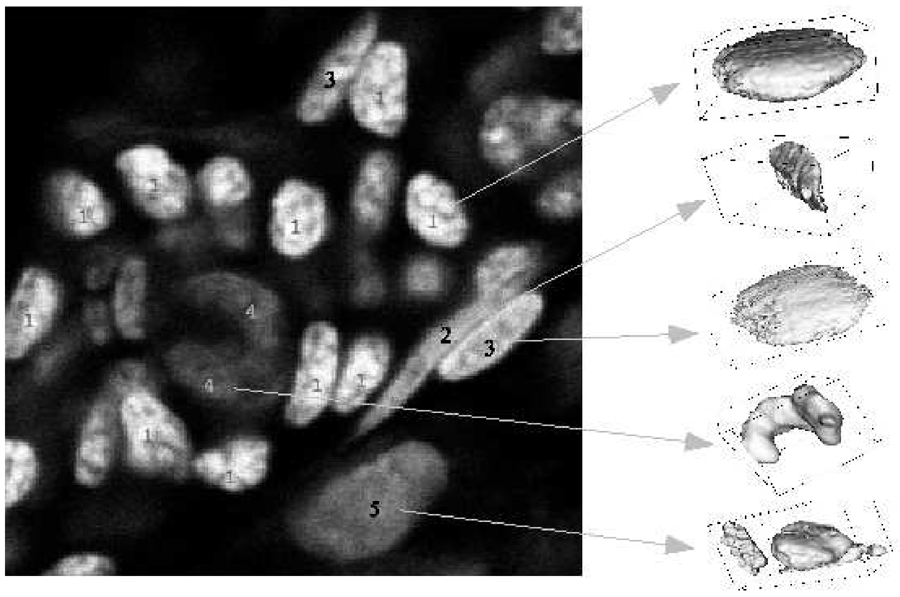


Figure 5.1: Sample data, cross section of a capillar y. Cell types with 3D reconstruction: 1. erythrocyte (**Ery**),2. endoth elial cell (**EC**), 3. pericyte (**PC**), 4. fibroblast (**F B**), 5. macrophage (**MΦ**).

5.2.2 Method

In total 32 gray scale invariants [ORB05, Feh04, SM95, BS01], that have already been successfully applied to the recognition of pollen grains in volumetric data sets [RBS02, RSB02] were extracted and used as features.

Nuclei (Std-Dev)	RBF Kernel	H1-SVM g1=0	H1-SVM g1=1	RBF/H1 g1=0	RBF/H1 g1=1
Nr. Features	32	32	32		
Nr. Train Samples	3372	3372	3372		
Nr. SVs or Hyperplanes	980	122	86	8.03	11.4
Training Time	00:00.98	00:03.03	00:02.43	0.32	0.4
Nr. Test Samples	65536	65536	65536		
Classif. Accuracy	64021	61480	62541	1.04	1.02
Classification Time	01:01.70	00:23.44	00:17.41	***2.63***	***3.54***
Classif. Accurancy %	97.69 %	93.81 %	95.43 %	1.04	1.02

Table 5.11: Results for the Nuclei dataset with the Std-Dev normalization method

Class	Correctness Accuracy %	Error %
0	6073/6887 (88.18%)	1009/58650 (1.72%)
2	498/539 (92.39%)	1010/65000 (1.554%)
4	437/565 (77.35%)	786/64970 (1.21%)
6	657/896 (73.33%)	195/64640 (0.3017%)
10	54876/56649 (96.87%)	142/8887 (1.598%)
total:	62541/65536 (95.43%)	2995/65536 (4.57%)

Table 5.12: Summary of classification results for the Nuclei data, Std-Dev, greedy l evel = 0

5.2.3 Non-linear Extension

In a final experiment, we compared the performance and accuracy of a strictly linear SVM-tree and one using the non-linear extension.

	RBF-Kernel	linear tree H1-SVM	non-linear tree H1-SVM
training time	≈1s	≈3s	≈5s
Nr. SVs or Hyperplanes	980	86	86
average classification depth	-	7.3	8.6
classifiaction time accuracy	≈1.5h 97.69%	≈2 min 95.43%	≈2 min 97.5%

Table 5.13: Comparison of the performance and accuracy of a strictly linear SVM-tree and one using the non-linear extension.

5.3 Conclusion

We have presented a new method for fast SVM classification. Compared to non-linear SVM and speedup methods our experiments showed a very competitive speedup while achieving reasonable classification results (loosing only marginal when we apply the non-linear extension compared to non-linear methods). Especially if our initial assumption holds , that large problems can be split in mainly easy and only a few hard problems, our algorithm achieves very good results. The advantage of this approach clearly lies in its simplicity since no parameter has to be tuned.

Bibliography

[AFB06] K. Zapién Arreola, J. Fehr, and H. Burkhardt, *Support vector machine classification using linear svms*, ICPR Hong Kong, 2006, pp. 366–369.

[BB00] Kristin P. Bennett and Erin J. Bredensteiner, *Duality and geometry in SVM classifiers*, Proc. 17th International Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA, 2000, pp. 57–64.

[BCSTW00] Kristin P. Bennett, Nello Cristianini, John Shawe-Taylor, and Donghui Wu, *Enlarging the margins in perceptron decision trees*, Machine Learning **41** (2000), no. 3, 295–313.

[BGV92] Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik, *A training algorithm for optimal margin classifiers*, Computational Learing Theory (In D. Haussler, ed.), 1992, pp. 144–152.

[BJ] P. Brazdil and J.Gama, *Statlog datasets*, http://www.liacc.up.pt/ML/statlog/datasets.html.

[BS97] Chris J. C. Burges and Bernhard Schölkopf, *Improving speed and accuracy of support vector learning machines*, Advances in Neural Information Processing Systems 9 (M. Jordan M. Mozer and T. Petsche, eds.), MIT Press, Cambridge, MA, 1997, pp. 375–381.

[BS01] Hans Burkhardt and S. Siggelkow, *Invariant features in pattern recognition – fundamentals and applications*, Nonlinear Model-Based Image/Video Processing and Analysis (C. Kotropoulos and I. Pitas, eds.), John Wiley & Sons, 2001, pp. 269–307.

[Car] Peter Carbonetto, *Faces datasets*, http://www.cs.ubc.ca/ pcarbo/.

[CL05a] Chih Chung Chang and Chih Jen Lin, *A library for support vector machines*, http://www.csie.ntu.edu.tw/ cjlin/libsvm/, 2005.

[CL05b] Chih-Chung Chang and Chih-Jen Lin, *Libsvm: a library for support vector machines*, www.csie.ntu.edu.tw/ cjlin/papers/libsvm.pdf, 2005.

[COL04] S. Cheong, S. H. Oh, and S. Y. Lee, *Support vector machines with binary tree architecture for multi-class classification*, Neural Information Processing - Letters and Reviews, vol. 2, 2004.

[Cri] David J. Crisp, *A geometric interpretation of nu-svm classifiers*, citeseer.ist.psu.edu/287847.html.

[DGM01] T. Downs, K. E. Gates, and A. Masters, *Exact simplification of support vector solutions*, Machine Learning **2** (2001), 293–297.

[FAB] J. Fehr, K. Zapién Arreola, and H. Burkhardt, *Fast support vector machine classification of very largedatasets*, submitted to GfKl 2007 post proceedings.

[FCL05] R.-E. Fan, P.-H. Chen, and C.-J. Lin, *Working set selection using the second order information for training svm*, Technical report, National Taiwan University, 2005.

[Feh04] Janis Fehr, *Segmentation and classification of cell nuclei in tissue slices using voxel-wise gray-scale invariants*, 2004.

[GBV93] I. Guyon, B. Boser, and V. Vapnik, *Automatic capacity tuning of very large VC-dimension classifiers*, Advances in Neural Information Processing Systems (Stephen José Hanson, Jack D. Cowan, and C. Lee Giles, eds.), vol. 5, Morgan Kaufmann, San Mateo, CA, 1993, pp. 147–155.

[HL01] C. Hsu and C. Lin, *A comparison of methods for multi-class support vector machines*, 2001.

[Hul94] J. J. Hull, *A database for handwritten text recognition research*, 1994, pp. 550–554.

[Joa98] T. Joachims, *Making large-scale support vector machine learning practical*, Advances in Kernel Methods: Support Vector Machines (A. Smola B. Schölkopf, C. Burges, ed.), MIT Press, Cambridge, MA, 1998.

[K$^+$02] H. Kurz et al., *Pericytes in experimental mda-mb231 tumor angiogenesis.*, Histochem Cell Biol (2002), 117:527–534.

[KG02] S. Keerthi and E. G. Gilbert, *Convergence of a generalized SMO algorithm for SVM classifier design*, Machine Learning **46** (2002), no. 1-3, 351–360.

[KSBM99] S. Keerthi, S. Shevade, C. Bhattacharyya, and K. Murthy, *Improvements to platt's smo algorithm for svm classifier design*, 1999.

[KW05] Hanbury A. Kropatsch W., Sablatnig R. (ed.), *Over-complete wavelet approximation of a support vector machine for efficient classification*, Proceedings of the 27th DAGM Symposium, Vienna, 2005.

[LHL05] Stine R. Lin H. and Auslender L. (eds.), *Speeding up multi-class svm evaluation by pca and feature selection*, 2005 SIAM Workshop, Newport Beach, CA, 2005.

[LL03] K. Lin and C. Lin, *A study on reduced support vector machines*, 2003.

[LM04] Y. Lee and O. Mangasarian, *Segmentation and classification of cell nuclei in tissue slices using voxel-wise gray-scale invariants*, 2004.

[MKS94] Sreerama K. Murthy, Simon Kasif, and Steven Salzberg, *A system for induction of oblique decision trees*, Journal of Artificial Intelligence Research **2** (1994), 1–32.

[NW99] J. Nocedal and S.J. Wright, *Numerical optimization*, Springer Series in Operations Research, Springer-Verlag New York, Inc., 1999.

[OFG97] E. Osuna, R. Freund, and F. Girosi, *Improved training algorithm for support vector machines*, 1997.

[ORB05] Janis Fehr Olaf Ronneberger and Hans Burkhardt, *Voxel-wise gray scale invariants for simultaneous segmentation and classification*, Inernal report, University of Freiburg, 2005.

[PCST00] J. Platt, N. Cristianini, and J. Shawe-Taylor, *Large margin dags for multiclass classification*, Advances in Neural Information Processing Systems 12 (S.A. Solla, T.K. Leen, and K.-R. Mueller, eds.), 2000, pp. 547–553.

[Pla99] J.C. Platt, *Fast training of support vector machines using sequential minimal optimization*, Advances in Kernel Methods: Support Vector Learning, MIT Press, Cambridge, MA, USA, 1999, pp. 185–208.

[RBS02] O. Ronneberger, H. Burkhardt, and E. Schultz, *General-purpose Object Recognition in 3D Volume Data Sets using Gray-Scale Invariants – Classification of Airborne Pollen-Grains Recorded with a Confocal Laser Scanning Microscope*, Proceedings of the International Conference on Pattern Recognition (Quebec, Canada), 2002.

[Rea04] Olaf Ronneberger and et al, *Svm template library*, http://lmb.informatik.uni-freiburg.de/lmbsoft/libsvmtl/index.en.html, 2004.

[Rep] UCI Repository, *Uci machine learning repository*, http://www.ics.uci.edu/ mlearn/MLRepository.html.

[RSB02] O. Ronneberger, E. Schultz, and H. Burkhardt, *Automated Pollen Recognition using 3D Volume Images from Fluorescence Microscopy*, Aerobiologia **18** (2002), 107–115.

[SM95] H. Schulz-Mirbach, *Invariant features for gray scale images*, 17. DAGM - Symposium “Mustererkennung” (Bielefeld) (G. Sagerer, S. Posch, and F. Kummert, eds.), Reihe Informatik aktuell, Springer, 1995, pp. 1–14.

[SS02] B. Schölkopf and A. Smola, *Learning with kernels*, The MIT Press, Cambridge, MA, USA, 2002.

[TKH96] *Building projectable classifiers of arbitrary complexity*, Vienna, Austria, 1996.

[Vap95] V. Vapnik, *The nature of statistical learning theory*, New York: Springer Verlag, 1995.

[VC79] V. Vapnik and A. Cervonenkis, *Theorie der zeichenerkennung*, Akademie Verlag, Berlin, 1979.

[WBCST99] Donghui Wu, Kristin P. Bennett, Nello Cristianini, and John Shawe-Taylor, *Large margin trees for induction and transduction*, Proc. 16th International Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA, 1999, pp. 474–483.