Incorporating a Priori Knowledge in the Form of Detractors into Support Vector Classification

Marcin Orchel
AGH University of Science and Technology, Mickiewicza Av. 30, 30-059 Kraków, Poland, marcin@orchel.pl

Abstract. In this article, we extend the idea of a priori knowledge in the form of detractors presented recently for Support Vector Classification. We show that detractor points can belong to the new type of support vectors – training examples which lie above the margin. We present the new application for a priori knowledge in the form of detractors – improving generalization performance of Support Vector Classification while reducing complexity of a model by removing a bunch of support vectors. Indeed, the experiments show that the new type of a priori knowledge improves generalization performance of reduced models. The tests were performed on various classification data sets, and on stock price data from public domain repositories.

Keywords: Support Vector Machines, a priori knowledge

1 Introduction

This article is a major extension of the [12], where the idea of a priori knowledge in the form of detractors for Support Vector Classification (SVC) has been introduced. The SVC method belongs to the group of methods called Support Vector Machines invented by Vapnik [14]. A priori knowledge in machine learning (ML) is defined as an additional knowledge to the existing training set. When it is formulated in terms of a particular domain, it is called domain dependent a priori knowledge, otherwise it is domain independent a priori knowledge. The example of the latter for a classification problem is the information about proper classification in knowledge sets (defined sets of points), particularly in continuous areas of the input space. Various types of areas were investigated recently for SVC: polyhedral sets [3] [2], ellipsoidal sets including spheroidal sets [4], nonlinear sets [11]. A polyhedral set is defined in the form of a set of linear equations, a spheroidal set with a center and a radius, an ellipsoidal set with a center and a matrix. Additionally, every set must have a classification value. Generally, the simplest formulation among these sets has a spheroidal set defined with a point, a number and a classification value.

The important aspect of a priori knowledge is efficient incorporation into the ML method. Generally, there are three methods of incorporation: modify input data like a set of features or some input parameters, modify the ML algorithm, or modify the ML method output. For SVC, the second method leads
to a modification of the optimization problem, particularly a modification of a
kernel function. For example, a priori knowledge in the form of classification of
a finite set of points could be directly incorporated by enlarging a training set,
the method is called a sample method \cite{7}. Polyhedral sets were incorporated to
SVC by modifying the optimization problem – by adding linear constraints \cite{3},
although an alternative incorporation scheme was proposed \cite{8}.

A priori knowledge in the form of detractors was first proposed in \cite{12}. A
detractor is a point in the input space with a classification value and a number,
called a detractor parameter which is a lower bound on the distance from this
point to the decision surface. A detractor can be interpreted as a knowledge
hypersphere with variable radius dependent on a decision function and hence
the one of differences between detractors and other mentioned earlier knowledge
sets is that the detractor knowledge hypersphere is defined dynamically, while
the others are defined statically with all parameters known before running the
ML method. Additionally, for the case of a soft margin classifier type of the SVC,
detractors could be treated as recommendations, which means that influential
power of detractors on a decision boundary depends on other factors, here on
slack variables. In the original SVC, only training examples which lie on or
below the margin could be support vectors. But in the SVC with detractors
knowledge, a detractor point can belong to the new type of support vectors –
training examples which lie above a margin. Complexity of a specification of
a detractor is similar to a spheroidal knowledge set, since there are only three
parameters a vector, a classification value and a number.

In this article, we use a priori knowledge in the form of detractors in reduced
models, which are created by removing a bunch of support vectors. The reduced
models were presented for a regression case in \cite{6}. The goal of creating such mod-
els is to reduce the complexity of the models, while preserving good performance
of the classifier. Reduced models are more suitable for further processing, such as
testing new examples. Comparing incorporation to the SVC method, detractors
are incorporated by adding detractor points to an input space and modifying the
SVC optimization problem by adding special weights to inequality constraints.
There are multiple attempts to incorporate spheroidal sets \cite{4}. Incorporation
of polyhedral sets proposed in \cite{3} is based on defining additional constraints for
the SVC optimization problem, and the method needs an optimization library to
solve new subproblems. For detractors, a modification of Sequential Minimiza-
tion Optimization (SMO) method \cite{13} which analytically solves two parameter
subproblems was proposed.

2 Detractors

A detractor for a classification case is defined as a point, called a detractor point
with a classification value, and with the additional parameter \(d\), called a detrac-
tor parameter, which is a lower bound on a distance from the detractor point
to the decision surface, measured in functional margin units. The incorporation
of detractors into SVC contains two steps: adding a detractor point with a clas-
sification value to a training set, and modifying the SVC primal optimization problem. If a training set already contains a detractor point, the first step is skipped. Now, we investigate closely a modification of the optimization problem. We use a formulation of the SVC optimization problem with training example weights, investigated for C-SVC in [17][15][5][10] and for ν-SVC in [16]. In this article, we consider incorporating detractors to C-SVC. A 1-norm soft margin SVC optimization problem for training examples \( a_i \) with sample weights \( C_i \) is

\[
\min f(w, b, \xi) = \frac{1}{2} \|w\|^2 + C \cdot \xi
\]

with constraints:

\[ y_i h(a_i) \geq 1 - \xi_i, \quad \xi \geq 0 \quad \text{for} \quad i \in \{1..L\}, \]

where \( C \gg 0 \), \( h(a_i) = w \cdot a_i + b \).

The \( i \)-th training example for which \( y_i h^*(a_i) = 1 \) is called a margin example. Margin boundaries are defined as the two hyperplanes \( h(x) = -1 \) and \( h(x) = 1 \). Optimal margin boundaries are defined as the two hyperplanes \( h^*(x) = -1 \) and \( h^*(x) = 1 \).

We introduce the SVC optimization problem with additional weights \( \phi \) for which \( d = 1 + \phi \)

\[
\min f(w, b, \xi) = \frac{1}{2} \|w\|^2 + C \cdot \xi
\]

with constraints: \( y_i h(a_i) \geq 1 - \xi_i + \phi_i, \quad \xi \geq 0 \quad \text{for} \quad i \in \{1..L\}, \)

where \( C \gg 0 \), \( \phi \geq 0 \), \( h(a_i) = w \cdot a_i + b \).

The new weights \( \phi \) are only present in constraints. When \( \phi = 0 \), the OP 2 is equivalent to the OP 1. A functional margin for a point \( p \) is defined as a value \( y_p h(p) \). A value \( v \) in functional margin units is equal to \( v / \|w\| \). We can see that a detractor parameter is a lower bound on a distance from a detractor example to a decision boundary measured in functional margin units: when we omit \( \xi_i \) in constraints for simplicity, we can see that \( y_i h^*(a_i) \geq d_i \), when we divide both sides by \( \|w\| \), we get \( y_i h^*(a_i) / \|w\| \geq d_i / \|w\| \). We can also note that when we take into account \( \xi_i \), detractors can be treated as recommendations, and their influential power depends on slack variables.

Note that modifying a detractor parameter does not always lead to a new decision boundary. Let’s assume that we modify only a one example \( p \) and \( \phi_p \) is equal to zero before the modification. When \( y_p h^*(p) > 1 \), then setting \( 0 < \phi_p \leq y_p h^*(p) - 1 \) does not affect a solution. When \( \phi_p > y_p h^*(p) - 1 \), the solution will be different, but not necessarily a decision boundary. Particularly, setting \( \phi_p > 0 \) could increase a slack variable and the solution would remain the same, when a value of \( C_p \) is small.
2.1 Interpretation of Detractors as Dynamic Hyperspheres

A detractor example $\mathbf{p}$ can be interpreted as a hypersphere with a radius equals to $\varphi_{\mathbf{p}}$ in functional margin units and therefore this is a dynamic hypersphere with a variable radius which depends on a decision function. The hypersphere must not intersect the margin boundary (in more than one point) $y_{\mathbf{p}}h(\mathbf{x}) = 1$. A value of the radius is represented in functional margin units and hence its absolute value varies among solution candidates. For the two solution candidates $h_1(\mathbf{x}) = 0$ and $h_2(\mathbf{x}) = 0$, where $h_2(\mathbf{x}) = a h_1(\mathbf{x})$ and $a \neq 0$ (both hyperplanes have the same geometric locations), the hyperspheres are respectively $S_1(\mathbf{p}, r)$, and $S_2(\mathbf{p}, r/a)$ (Fig. 1).

![Fig. 1. Interpretation of detractors as dynamic hyperspheres. We can see the two solution candidates for particular data ($h_1(\mathbf{x})$ on the left and $h_2(\mathbf{x})$ on the right) with detractors visualized by circles. In the right figure, radii of detractor’s circles differ from the first one proportionally to the changes of the functional margins for the detractors](image)

2.2 An Efficient Solution of the SVC Optimization Problem with Detractors

In order to construct an efficient algorithm for the OP 2 its dual form was derived. The final form of the dual problem is

**OP 3.** Maximize

$$d(\alpha) = \alpha \cdot (1 + \varphi) - \frac{1}{2} \alpha^T Q \alpha$$

with constraints $\alpha \cdot \mathbf{y} = 0$, $0 \leq \alpha \leq C$,

where $Q_{ij} = y_i y_j (\mathbf{a}_i \cdot \mathbf{a}_j)$, for all $i, j \in \{1..l\}$. 

It differs from the original SVC dual form by only $\alpha \cdot \varphi$ term. In the above formulation, similarly as for the original SVC, it is possible to introduce nonlinear decision functions by using a kernel function instead of a scalar product. The final decision boundary has a form:

$$h^* (x) = \sum_{i=1}^{l} y_i \alpha_i^* K (a_i, x) + b^* = 0 ,$$

where $K (\cdot, \cdot)$ is a kernel function. The $i$-th example is a support vector, when $\alpha_i^* \neq 0$. Based on the Karush-Kuhn-Tucker complementary condition

$$\begin{cases} 
\alpha_i \left( y_i h (a_i) - 1 - \varphi_i + \xi_i \right) = 0 \\
(C - \alpha_i) \xi_i = 0 
\end{cases}$$

we can conclude which examples could be support vectors. In the original SVC, only the example which lie on the optimal margin boundaries ($y_i h^* (a_i) = 1$) or below optimal margin boundaries ($y_i h^* (a_i) < 1$) could be a support vector. In the SVC with detractors, also the example fulfilling $\varphi_i > 0$ and lying above margin boundaries ($y_i h^* (a_i) > 1$) could be a support vector. Such example is called a detractor support vector. An output model is defined based on support vectors. Introducing the new type of support vectors leads to richer models, where additional examples lying above optimal margin boundaries could participate in defining a decision function.

In order to solve OP 3, a decomposition method similar to SMO [13] which solves the original SVC dual optimization problem was derived. For two chosen parameters $i_1$ and $i_2$ the solution without clipping is

$$\alpha_{i_2}^{\text{new}} = \alpha_{i_2} + \frac{y_{i_2} (E_{i_1} - E_{i_2})}{\kappa} ,$$

where $\kappa = K_{i_1 i_1} + K_{i_2 i_2} - 2K_{i_1 i_2}$ and

$$E_i = \sum_{j=1}^{l} y_j \alpha_j K_{ij} - y_i \varphi_i . \quad (1)$$

After that, $\alpha_{i_2}$ is clipped in the same way as for SMO, but with variable weights $C_i$

$$U \leq \alpha_{i_2}^{\text{clipped}} \leq V ,$$

where for $y_1 \neq y_2$: $U = \max \left( 0, \alpha_{i_1}^{\text{old}} - \alpha_{i_1}^{\text{old}} \right), V = \min \left( C_{i_2}, C_{i_1} - \alpha_{i_1}^{\text{old}} + \alpha_{i_2}^{\text{old}} \right)$, for $y_1 = y_2$: $U = \max \left( 0, \alpha_{i_1}^{\text{old}} + \alpha_{i_2}^{\text{old}} - C_{i_1} \right), V = \min \left( C_{i_2}, \alpha_{i_1}^{\text{old}} + \alpha_{i_2}^{\text{old}} \right)$. The parameter $\alpha_i$ is $\alpha_i^{\text{new}} = \gamma - y_i y_j \alpha_{i_2}^{\text{clipped}}$, where $\gamma = \alpha_{i_1}^{\text{old}} + y_i \varphi_i \alpha_{i_2}^{\text{old}}$. Based on the KKT complementary condition, it is possible to derive equations for the SVC heuristic and the SVC stopping criteria. After incorporating weights $\varphi$, a heuristic and stopping criteria are almost the same, with the one difference, that values of $E_i$ are computed as stated in (1).
2.3 Reduce a Model by Removing Support Vectors

We use the method of removing support vectors to decrease the SVC model complexity. Reduced models are more suitable for further processing, e.g. for testing new examples. However, reduced models have the disadvantage that generalization performance could be worse than for the original full models. The reduced models were recently proposed for Support Vector Regression [6], which solves a regression problem. We propose a new method which generates reduced models for classification problems. The proposed method generates reduced models from the original full model with incorporated a priori knowledge in the form of detractors. Reduced models with the additional a priori knowledge have better generalization performance compared to the reduced models without the additional knowledge. The procedure of generating knowledge in the form of detractors is as following. First, detractors are automatically generated from an existing solution by setting

$$\varphi_i = y_i h^*(a_i) - 1$$

for training examples for which $\varphi_i > 0$. Note that a number of detractors depends on data. It is possible, that no detractors would be generated for solutions when all training examples are support vectors. In this situation detractors could be generated automatically by adding the new examples with functional margins greater than one. Although this special case was not tested in this article. After that, a reduced model is generated by removing a bunch of support vectors – randomly selected support vectors, with maximal removal ratio of $p\%$ of all training vectors, where $p$ is a configurable parameter. Finally, we run the SVC method with reduced data.

3 Experiments

In experiments, we show that the reduced models with knowledge in the form of detractors have better performance than without the additional knowledge. The first method does not use knowledge in the form of detractors in reduced models, the second one use the additional knowledge. In the first experiment, we set arbitrarily $p = 70$. Note that for comparison purposes a reduced model is the same for both methods. We use the author implementation of SVC for both methods. In the second experiment, we show that the proposed method has better performance for variable $p$.

For all data sets, every feature is scaled linearly to $[0, 1]$ including an output. For variable parameters like the $C$, $\sigma$ for the RBF kernel, $\varphi$ for SVCR, and $\varepsilon$ for $\varepsilon$-SVR we use a grid search method for finding best values. The number of values searched by the grid method is a trade-off between an accuracy and a speed of simulations. Note that for particular data set it is possible to use more accurate grid searches than for massive tests with multiple number of simulations.
3.1 Synthetic Data Tests

We compare both methods on data generated from particular functions with added Gaussian noise for output values. We perform tests with a linear kernel on linear functions, with a polynomial kernel on the polynomial function, with the RBF kernel on the sine function. The tests with results are presented in Table 1. The method with knowledge in the form of detractors has better performance for every kernel, a number of support vectors is comparable. A testing performance gain varies from 0% to 51%.

Table 1. Description of test cases with results for synthetic data for generating reduced models by removing support vectors. Column descriptions: a function – a function used for generating data $y_1 = \sum_{i=1}^{\dim-1} x_i$, $y_4, y_5 = \left(\sum_{i=1}^{\dim-1} x_i\right)^{kerP}$, $y_6 = 0.5\sum_{i=1}^{\dim-1} \sin 10x_i + 0.5$, $p$ - the parameter $p$, $simC$ – a number of simulations, results are averaged, $\sigma$ – a standard deviation used for generating noise in output, $ker$ – a kernel (pol – a polynomial kernel), $kerP$ – a kernel parameter (for a polynomial kernel it is a dimension, for the RBF kernel it is $\sigma$), $trs$ – a training set size, $tes$ – a testing set size, $dm$ – a dimension of the problem, $tr12M$ – a percent average difference in correctly classified examples for training data, if greater than 0 than a method with detractors is better, $te12M$ – the same as $tr12M$, but for testing data, $tr12MC$ – a percent average difference in number of tests for training data in which a method with detractors is better (the method with detractors is better when a value is greater than 50%), $te12MC$ – the same as $tr12MC$, but for testing data, $s1$ – an average number of support vectors for a method without detractors, $s2$ – an average number of support vectors for a method with detractors, $d$ – an average number of detractors. The value 'var' means that we search for the best value comparing a number of correctly classified examples for the training data.

<table>
<thead>
<tr>
<th>function</th>
<th>p</th>
<th>simC</th>
<th>$\sigma$</th>
<th>ker</th>
<th>kerP</th>
<th>trs</th>
<th>tes</th>
<th>dm</th>
<th>tr12M</th>
<th>te12M</th>
<th>tr12MC</th>
<th>te12MC</th>
<th>s1</th>
<th>s2</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y_1$</td>
<td>70</td>
<td>100</td>
<td>0.04 lin</td>
<td>-</td>
<td>-</td>
<td>90</td>
<td>300</td>
<td>5</td>
<td>64%</td>
<td>51%</td>
<td>94%</td>
<td>90%</td>
<td>14</td>
<td>18</td>
<td>17</td>
</tr>
<tr>
<td>$y_2 = 3y_1$</td>
<td>70</td>
<td>100</td>
<td>0.04 lin</td>
<td>-</td>
<td>-</td>
<td>90</td>
<td>300</td>
<td>5</td>
<td>5%</td>
<td>4%</td>
<td>50%</td>
<td>50%</td>
<td>20</td>
<td>21</td>
<td>10</td>
</tr>
<tr>
<td>$y_3 = 1/3y_1$</td>
<td>70</td>
<td>100</td>
<td>0.04 lin</td>
<td>-</td>
<td>-</td>
<td>90</td>
<td>300</td>
<td>5</td>
<td>70%</td>
<td>50%</td>
<td>95%</td>
<td>90%</td>
<td>12</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>$y_4$</td>
<td>70</td>
<td>100</td>
<td>0.04 pol</td>
<td>3</td>
<td>-</td>
<td>90</td>
<td>300</td>
<td>5</td>
<td>1.5%</td>
<td>0%</td>
<td>50%</td>
<td>45%</td>
<td>20</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>$y_5$</td>
<td>70</td>
<td>50</td>
<td>0.04 rbf</td>
<td>var</td>
<td>-</td>
<td>90</td>
<td>300</td>
<td>5</td>
<td>10%</td>
<td>6%</td>
<td>80%</td>
<td>60%</td>
<td>21</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>$y_6$</td>
<td>70</td>
<td>10</td>
<td>0.04 rbf</td>
<td>var</td>
<td>-</td>
<td>90</td>
<td>300</td>
<td>5</td>
<td>3%</td>
<td>2%</td>
<td>30%</td>
<td>30%</td>
<td>25</td>
<td>25</td>
<td>5</td>
</tr>
</tbody>
</table>

3.2 Real World Data Sets

The real world data sets were taken from the LibSVM site [1] [9] except stock price data. The stock price data consist of monthly prices of the DJIA index from 1898 up to 2010. We generated the training set as follows: for every month the output value is a growth/fall comparing to the next month. Every feature $i$ is a percent price change between the month and the $i$-th previous month. In
every simulation, training data are randomly chosen, the remaining examples become test data. The tests with results are presented in Table 2. The method with knowledge in the form of detractors has better performance for all data sets, for all kernels with similar number of support vectors. The testing performance varies from 0% to 27%. For the djia data set, results are comparable.

Table 2. Description of test cases with results for real world data for generating reduced models by removing support vectors. Column descriptions: a name – a name of the test, breastCr – breastCancer test, p – the parameter p, simT – a number of random simulations, where training data are randomly selected, results are averaged, ker – a kernel (pol – a polynomial kernel), kerP – a kernel parameter (for a polynomial kernel it is a dimension, for the RBF kernel it is σ), trs – a training set size, all – a number of all data, it is a sum of training and testing data, dm – a dimension of the problem, tr12M – a percent average difference in correctly classified examples for training data, if greater than 0 than a method with detractors is better, te12M – the same as tr12M, but for testing data, tr12MC – a percent average difference in number of tests for training data in which a method with detractors is better (a method with detractors is better when a value is greater than 50%), te12MC – the same as tr12MC, but for testing data, s1 – an average number of support vectors for a method without detractors, s2 – an average number of support vectors for a method with detractors, d – an average number of detractors. The value ‘var’ means that we search for the best value comparing a number of correctly classified examples for the training data

<table>
<thead>
<tr>
<th>name</th>
<th>p</th>
<th>simT</th>
<th>ker</th>
<th>kerP</th>
<th>trs</th>
<th>all</th>
<th>dm</th>
<th>tr12M</th>
<th>te12M</th>
<th>tr12MC</th>
<th>te12MC</th>
<th>s1</th>
<th>s2</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1a1</td>
<td>70</td>
<td>100</td>
<td>lin</td>
<td>–</td>
<td>90</td>
<td>5835</td>
<td>123</td>
<td>20%</td>
<td>6%</td>
<td>85%</td>
<td>80%</td>
<td>17</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>a1a2</td>
<td>70</td>
<td>100</td>
<td>pol</td>
<td>3</td>
<td>90</td>
<td>5835</td>
<td>123</td>
<td>3%</td>
<td>1%</td>
<td>40%</td>
<td>75%</td>
<td>25</td>
<td>25</td>
<td>8</td>
</tr>
<tr>
<td>a1a3</td>
<td>70</td>
<td>20</td>
<td>rbf</td>
<td>var</td>
<td>90</td>
<td>5835</td>
<td>123</td>
<td>35%</td>
<td>5%</td>
<td>100%</td>
<td>70%</td>
<td>23</td>
<td>33</td>
<td>33</td>
</tr>
<tr>
<td>breastCr1</td>
<td>70</td>
<td>100</td>
<td>lin</td>
<td>–</td>
<td>90</td>
<td>639</td>
<td>10</td>
<td>40%</td>
<td>11%</td>
<td>65%</td>
<td>70%</td>
<td>7</td>
<td>7</td>
<td>25</td>
</tr>
<tr>
<td>breastCr2</td>
<td>70</td>
<td>100</td>
<td>pol</td>
<td>3</td>
<td>90</td>
<td>639</td>
<td>10</td>
<td>45%</td>
<td>27%</td>
<td>73%</td>
<td>82%</td>
<td>6</td>
<td>7</td>
<td>24</td>
</tr>
<tr>
<td>breastCr3</td>
<td>70</td>
<td>20</td>
<td>rbf</td>
<td>var</td>
<td>90</td>
<td>639</td>
<td>10</td>
<td>23%</td>
<td>10%</td>
<td>45%</td>
<td>60%</td>
<td>13</td>
<td>13</td>
<td>20</td>
</tr>
<tr>
<td>diabetes1</td>
<td>70</td>
<td>100</td>
<td>lin</td>
<td>–</td>
<td>90</td>
<td>768</td>
<td>8</td>
<td>8%</td>
<td>4%</td>
<td>65%</td>
<td>70%</td>
<td>15</td>
<td>15</td>
<td>13</td>
</tr>
<tr>
<td>diabetes2</td>
<td>70</td>
<td>100</td>
<td>pol</td>
<td>3</td>
<td>90</td>
<td>768</td>
<td>8</td>
<td>10%</td>
<td>2%</td>
<td>75%</td>
<td>60%</td>
<td>15</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>diabetes3</td>
<td>70</td>
<td>20</td>
<td>rbf</td>
<td>var</td>
<td>90</td>
<td>768</td>
<td>8</td>
<td>10%</td>
<td>5%</td>
<td>80%</td>
<td>60%</td>
<td>19</td>
<td>18</td>
<td>12</td>
</tr>
<tr>
<td>djia1</td>
<td>70</td>
<td>100</td>
<td>lin</td>
<td>–</td>
<td>90</td>
<td>1351</td>
<td>10</td>
<td>2%</td>
<td>0%</td>
<td>40%</td>
<td>40%</td>
<td>20</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>djia2</td>
<td>70</td>
<td>100</td>
<td>pol</td>
<td>3</td>
<td>90</td>
<td>1351</td>
<td>10</td>
<td>3%</td>
<td>0%</td>
<td>55%</td>
<td>50%</td>
<td>19</td>
<td>19</td>
<td>7</td>
</tr>
<tr>
<td>djia3</td>
<td>70</td>
<td>20</td>
<td>rbf</td>
<td>var</td>
<td>90</td>
<td>1351</td>
<td>10</td>
<td>4%</td>
<td>0%</td>
<td>50%</td>
<td>40%</td>
<td>25</td>
<td>23</td>
<td>4</td>
</tr>
</tbody>
</table>

3.3 Variable p

In the second experiment, we show that for variable p the second method has better performance. The example results for the first test case from synthetic tests are depicted in Fig. 2.
4 Conclusions

We show experimentally that knowledge in the form of detractors allows to construct reduced SVC models from existing ones with better performance than models without that knowledge. While removing support vectors, new models are generated with much fewer support vectors. When additionally detractors are used, new models preserve good generalization performance.

Acknowledgments. The research is financed by the Polish Ministry of Science and Higher Education project No NN519579338. I would like to express my sincere gratitude to Professor Witold Dzwinel (AGH University of Science and Technology, Department of Computer Science) for contributing ideas, discussion and useful suggestions.

References