Two Ellipsoids Support Vector Machines

Wojciech Marian Czarnecki\textsuperscript{a,*}, Jacek Tabor\textsuperscript{a}

\textsuperscript{a}Faculty of Mathematics and Computer Science, Jagiellonian University, prof. Stanisława Lojasiewicza 6, 30-348 Krakow, Poland

Abstract

In classification problems classes usually have different geometrical structure and therefore it seems natural for each class to have its own margin type. Existing methods using this principle lead to the construction of the different (from SVM) optimization problems. Although they outperform the standard model, they also prevent the utilization of existing SVM libraries. We propose an approach, named 2eSVM, which allows use of such method within the classical SVM framework.

This enables to perform a detailed comparison with the standard SVM. It occurs that classes in the resulting feature space are geometricaly easier to separate and the trained model have better generalization properties. Moreover, based on evaluation on standard datasets, 2eSVM brings considerable profit for the linear classification process in terms of training time and quality.

We also construct the 2eSVM kernelization and perform the evaluation on the 5-HT\textsubscript{2A} ligand activity prediction problem (real, fingerprint based data from the cheminformatic domain) which show increased classification quality, reduced training time as well as resulting model’s complexity.

Keywords: Support Vector Machines, Mahalanobis distance, classification, data preprocessing

1. Introduction

Binary classification is a core problem in machine learning, relevant from both theoretical and practical perspective. Over the past decade, Support Vector Machine (SVM) model [1] gained great interest due to its good mathematical formulation accompanied with a great number of empirical results. Several modifications were proposed, ranging from the modifications of norms used in the main SVM’s equation [2], through considering Bayesian treatment of the problem [3] to generalization from original separating hyperplanes to

\*Corresponding author, tel: +48 12 664 7556
Email addresses: wojciech.czarnecki@uj.edu.pl (Wojciech Marian Czarnecki), jacek.tabor@uj.edu.pl (Jacek Tabor)
hyperspheres and beyond [4]. From the perspective of our paper of crucial importance are the generalizations which implement the idea that every class should have its own margin type: Twin Mahalanobis SVM [5] and Maxi-Min Margin Machine (M^4) [6].

Most of existing SVM modifications have shown their superiority over the classical method in some contexts and applications, however in practice, Vapnik’s model (with later kernelization) is still the most commonly used. This is caused in particular by the fact that most SVM modifications require considerable amount of time and specialist knowledge to use, while Vapnik’s model is implemented in most machine learning packages.

This is why in this paper we introduce a Two Ellipsoids SVM (2eSVM) model which uses two distinct margins’ types and allows easy implementation within the classical SVM framework. In fact we treat SVM as a Black Box, and perform only the pre- and post-processing of the data, see Figure 1. Our approach allows not only the use of existing SVM libraries, but also gives the ability of careful comparison with the classical SVM modifications like Mahalanobis SVM.

The main idea behind Maxi-Min Margin Machine (M^4) [6] on which we based our ideas, is to seek for the hyperplane which simultaneously maximizes the size of different margins for classes $X_-$ and $X_+$. The process of finding the maximal separating margin in the standard SVM algorithm can be seen as searching for the biggest radius $r$ such that the sets

$$X_- + B(0,r) \text{ and } X_+ + B(0,r)$$

are linearly separable, where $B(0,r)$ denotes the standard ball with radius $r$ centered at zero. From the
geometrical and practical point of view it is better to use two different hyperellipsoids $B_-(0, r)$ and $B_+(0, r)$ (balls in different metrics) fitted for each class separately. Consequently, one seeks for the maximal $r$ such that the sets

$$X_- + B_-(0, r) \text{ and } X_+ + B_+(0, r)$$

are linearly separable, see Fig. 2.

![Figure 2: Visualization of the idea on two elliptical-shaped classes (horizontal – positive samples and vertical – negative ones).](image)

As a result, the separating hyperplane is located nearer the “vertical” class. This is a better solution, as a big horizontal variance of the other class suggests that more points drawn randomly from the underlying distributions, which lay on the $x$ axis “between” these two ellipsoids, are actually members of “horizontal” class.

This leads to the Second Order Cone Optimization Problem which cannot be solved by the standard SVM procedure [6]. We prove however that we can implement a similar principle with the Black Box use of SVM by applying the following steps:

1. transform the data (or the kernel) using a matrix computed from the sum of classes covariances,
2. train a classical SVM,
3. shift a decision boundary.

Evaluation on the typical datasets shows that the first operation allows the SVM to separate data easier and faster, while the third helps to obtain better classification results. In particular we obtain (see Section 6):
- a speed up of the learning process for the (both linear and kernelized) SVM of two to four times,
- reduction of mean number of support vectors of resulting model for the kernelized SVM by up to 10%,
- statistically significant better generalization results than the classical approach.

The most important differences between 2eSVM and M⁴ are:

- 2eSVM is much simpler to implement, as requires just few lines of algebraic operations,
- 2eSVM is much more robust, as it uses a SVM as an underlying optimization problem, which is a quadratic optimization with linear constraints, while M⁴ requires second order cone optimization which is much more complex optimization problem,
- M⁴ requires custom optimization, while 2eSVM can be easily integrated with almost any existing SVM library,
- However, even though 2eSVM implements similar idea to M⁴, due to its approximated nature, achieves smaller accuracy gain.

The idea behind our method is related to the problem of finding the best metric as well as complexity reduction techniques. Metric learning concerns the problem of finding the best metric for given model and data as the independent optimization problem. Methods of this type have been used to build a hybrid model using both k-nearest neighbours and SVM concept – LM-KNN [7]. There have also been presented multiple modifications of support vector machines [8], including incorporating the metric optimization in the core SVM optimization itself [9]. The Ellipsoid SVM model [10] is a particular example of such approach, where one looks for the best fitted hyper-ellipsoid around the data to construct the correct metric. Those methods help model to better fit the underlying geometry of the data at the cost of additional computational requirements and in general – increased complexity of the problem.

On the other hand proximal SVM [11] changes the basic formulation of the SVM to obtain a much simpler optimization problem. In this setting one searches for two parallel hyperplanes, around which points of particular classes are clustered, which are as far from each other as possible. Twin SVM [12] generalized this idea, so two hyperplanes can be non-parallel giving model better data geometrical fitting capabilities. Main strength of these methods lies in reduction of the complexity of the optimization problem by either weakening the parameters constraints [11] or by solving multiple smaller problems [12].
Proposed method differs from the above approaches, as it is based on the closed form pre- and postprocessing method of the traditional SVM. It does not require solving any additional optimization problems or fitting any extra parameters. It tries to accommodate both these concepts – to better fit the data geometry on one hand, and to reduce the problem’s complexity on the other. One of the most important aspects of such approach is that it can be easily incorporated into existing research methods and tools, without the need of changing the model.

Concluding, the natural application of proposed model are problems where classes are very diverse in terms of internal geometries. These in particular include:

- cheminformatical data – described in detail in evaluation section,
- Natural Language Processing [13, 14, 15] – the internal variety of a vocabulary used in different context tends to be very diversified,
- Financial data [16] – in economical applications it is common to have very small, geometrically condensed class (positive samples) surrounded by big (negative) class.

Our paper is structured as follows: in the next section we show the geometrical intuition of our method, next we focus on the theoretical justification of the proposed approach and show that under reasonable assumptions our algorithm finds a better solution to the classification problem than classical methods. In Section 4 we provide a pseudo-code of 2eSVM, with practical remarks regarding its usage. Next, the proposed approach is compared with common method of covariance matrix data transformation on both standard and real data (5-HT$_2$A ligand activity prediction with respect to various fingerprints types [17]). We conclude with a short discussion.

2. Geometrical intuition

Let us present the geometrical motivation behind our idea. In classical preprocessing approach in data analysis one transforms the input space through the $\Sigma^{-1/2}$ where $\Sigma$ is a covariance matrix of the whole dataset. This results in the feature space where points are more radially distributed. However, such approach completely ignores the fact that data in separate classes often have different covariances. Moreover, even if inner class covariances are identical, the covariance of the whole dataset is usually different.

As an example, let us consider two highly skewed 2D Gaussian distributions with variances on axes $x, y$ equal to 0.05 and 1 respectively, centered in points $(-0.15, -6)$ and $(0.15, 6)$ (Fig. 3a). After transformation
Figure 3: Toy example of two highly skewed Gaussian distributions. Grey separating line is the optimal solution in terms on underlying distributions.

using $\Sigma^{-1/2}$ our data gets vertically compressed so it forms two, almost parallel (in the sense of ellipsoidal shape) distributions (Fig. 3b). In such feature space, SVM would find a better separating hyperplane than applied to the original dataset (compare Fig. 3d and Fig. 3e), but the optimization process would be harder as there are more potential support vectors in this representation (more points are located near the separating hyperplane).

Instead, we show that the optimal results are obtained by applying two steps: transformation through $(\Sigma_+ + \Sigma_-)^{-1/2}$ where $\Sigma_+$ and $\Sigma_-$ are covariance matrices of positive and negative classes respectively and shifting the decision boundary (see Section 3). In the proposed toy example, our method correctly identifies inner class covariances. Transformation results in data which is easily separable (there is a low amount of potential support vectors as shown on Fig. 3c) and yields the best possible separation when using SVM (compare Fig. 3f with previous ones). One can verify that our approach yields the asymptotically optimal separation for Gaussians of equal covariances.

For a more realistic example, let us consider the Iris dataset restricted to the first two classes and first
two dimensions. Here, classes have different covariances. If we draw a confidence region of the covariance matrices one can exactly see the difference in calculating the whole set covariance (Fig. 4b) and sum of covariances (Fig. 4c). This is quite natural and intuitive to treat covariances as inner products or ellipsoids. This geometrical intuition (as well as representing them as self-adjoint operators in the theoretical part) will be used in further parts of our paper.

![Figure 4: Iris dataset restricted to first 2 classes and first 2 dimensions. Centered ellipses represent the inner product used in SVM. In the Fig. 4a we use standard Euclidean product; in the Fig. 4b we use inner product based on the covariance of the whole dataset and in the Fig. 4c we use inner product based on the sum of classes' covariances.](image)

After classical transformation many points are located near the separation hyperplane (Fig. 4e), while in our method this number is greatly reduced\(^1\) (Fig. 4f). As one can see, resulting ellipsoids become almost perpendicular, which results in complexity reduction. This outlines the main difference between our method and previous approach – we preprocess the data so it can be separated more easily using SVM by reducing the number of potential support vectors and exploiting the underlying geometrical divergence of the data.

As it was mentioned, the second part of our method is based on the shift of a decision boundary. The

\(^1\)Which is supported by our empirical evaluation on other datasets
motivation comes from theoretical calculations and the fact that using a sum of covariances loses information regarding proportions of individual covariances. Detailed description and justification of the shift is presented in Section 3.

Let us consider an example where we have classes in form of two 2D Gaussians of different variances, Fig. 5. The optimal separating hyperplane in the SVM sense is located exactly in the middle of the margin. However, better results are obtained if the separating hyperplane lies much closer to the class with smaller variance, Fig. 5a. Consequently in our method we shift the boundary location so its position in the margin captures the proportions of the particular variances along the perpendicular line to the separating hyperplane, see Fig. 5b. Due to the SVM formulation, both these actions are easy to compute and implement – projection to this line is performed by the machine itself and shifting a boundary is just a change of one parameter of the model. What is even more important, as is shown in Section 3, this step is motivated by theoretical considerations regarding separating data with two different margins using our preprocessing.

Let us now investigate what effect can have selection of scalar product on the number of support vectors of resulting kernel machine. Support Vector Machines are always a linear classifiers. Even if we use some kernel function, on the mathematical level they still look for a hyperplane, located in the feature space induced by selected kernel. In Mahalanobis kernel, defined by $K_\Sigma(x,y) = x^T \Sigma^{-1} y$ such feature space is given by $\phi_\Sigma(x) = \Sigma^{-1/2} x$, as $\langle \phi_\Sigma(x), \phi_\Sigma(y) \rangle = (\Sigma^{-1/2} x)^T (\Sigma^{-1/2} y) = x^T \Sigma^{-1} y = K_\Sigma(x,y)$. We now recall two Gaussians dataset (Fig. 3a) and corresponding feature space embeddings induced by the standard Mahalanobis
kernel and proposed method (Fig. 3b and Fig. 3c). If we look closely to the distribution of points distances to the separating hyperplane (in the feature space) we notice that proposed method shatters points in such a way that statistically their distances are much bigger. In the hard margin SVM case (which is strictly theoretical object, rarely used in practise, with \( C = \infty \)) the number of support vectors in both cases is the same. However, once we change optimization problem to soft margin SVM (with any \( C \in (0, \infty) \)) the support vectors set grows, and more points located further away from the separating hyperplane are added. For the suboptimally selected scalar product (in this case – Mahalanobis one) this process almost instantly adds most of the training samples to the support vectors class, while in case of the proposed method – this process is much more fine-grained and only those truly required for the construction of optimal linear separator are selected. This geometrical property is responsible for the reduction of the number of support vectors, which is reported in the evaluation section.

As mentioned in the previous section, 2eSVM is aimed at selection of such scalar product (metric) which better fits data geometry. In particular, proposed preprocessing typically transform the dataset in such a way, that in terms of resulting covariance ellipses, classes become much more perpendicular see Fig. 4c. Consequently, much less points are potential support vectors (compare with amount of points near separating hyperplane in Fig. 4b). Such reduction of the candidates set speeds up the SVM training algorithm. Similar argument can be drawn from the previous reasoning, if we choose wrong (not similar to classes’ covariances) Mahalanobis metric then finding the optimal decision boundary in the SVM sense requires checking of lots of potentially redundant point pairs.

Ability to increase the overall system’s accuracy comes directly from the fact of removing the SVM assumption of the heterogenous classes geometries. As further described in the theoirical section, classical SVM assumes that both classes are similar (in terms of shape) and so the selection of the hyperplane which is exactly in the middle of the distance (in terms of the euclidean metric) seems reasonable. However, once we remove this assumption and allow classes to have arbitrary shapes, the resulting classifier is not optimal even for the case of two Gaussian distributions which differ only in terms of variance (see Fig. 5). On the other hand taking into account this diversity is a more general framework which in the recalled example yields optimal (in the Bayesian sense) linear classifier. In actual real applications, internal heterogenousity of classes’ geometries is a common phenomenon (in particular we report this characteristics for datasets used during experiments in Evaluation section) and often results in increased accuracy. It is also a known fact that reduced number of support vector increases the generalization capabilities of SVM as it decreases the Vapnik-Chervonenkis dimension.
3. Theory

SVM aims at construction of an (optimal) linear classifier between sets $X_+$ (positive examples) and $X_-$ (negative examples) in an Euclidean space $E$. Recall that sets $Y_-$ and $Y_+$ are linearly separable if there exist nonzero $w \in E$ and $b \in \mathbb{R}$ such that

$$Y_- \subset w < b := \{x : \langle w, x \rangle < b \} \quad \text{and} \quad Y_+ \subset w > b := \{x : \langle w, x \rangle > b \}.$$ 

Thus the largest margin problem which gives the basics of SVM can be stated in the following manner:

**Geometrical formulation: classical SVM** Given sets $X_+, X_- \subset E$, find maximum of $r > 0$ such that the sets $X_- + B(0, r)$ and $X_+ + B(0, r)$ are linearly separable\(^2\),

\begin{equation}
X_- + B(0, r) \quad \text{and} \quad X_+ + B(0, r)
\end{equation}

where we interpret $r \to B(0, r)$ as the margin function.

In this section by a ball we understand open ball. By the Hahn-Banach Theorem condition (3.1) holds iff

$$[\text{conv}X_+ + B(0, r)] \cap [\text{conv}X_- + B(0, r)] = \emptyset.$$ 

However, this causes a natural problem: what ball (or equivalently distance) we should use for the separation. A common idea is to use the classical Mahalanobis distance [18], which usually best underlines the internal geometrical structure of the data. Consequently in many cases we use the covariance $\Sigma$ of the whole dataset $X_+ \cup X_-$ as the generator of the Mahalanobis norm. In practice this is done in two equivalent ways: either we change the scalar product and the norm to

$$\langle x, y \rangle_\Sigma := x^T \Sigma^{-1} y \quad \text{and} \quad \|x\|_\Sigma^2 := x^T \Sigma^{-1} x,$$

or we use the standard product on the data transformed\(^3\) by the operation $x \to \Sigma^{-1/2} x$.

Although this approach and its modifications are commonly used [10], it needs an underlying assumption that sets $X_+$ and $X_-$ have the same covariance and geometry, which is rarely the case, see for example a simple but typical picture of Iris data (Fig. 4a). Consequently, since we typically classify between classes which

---

\(^2\)Obviously to use the above result for classification, we need the $w$ and $b$ which define the linear separation

\(^3\)Sometimes we first transform also the mean of the data $m$ to origin to minimize the numerical errors, and consequently apply the transformation $x \to \Sigma^{-1/2}(x - m)$. 

---

10
have different covariance matrices and different internal geometry, one can argue [6] that each class should have its own margin function:

**Geometrical formulation: Maxi-Min Margin Machine** Assume that we are given two balls $B_+(0, 1)$ and $B_-(0, 1)$ in $E$. Given sets $X_-, X_+ \subset E$, find maximum of $r \geq 0$ such that the sets

$$X_- + B_-(0, r) \text{ and } X_+ + B_+(0, r)$$

are linearly separable.

Observe that to use the above result for binary classification, we need the nonzero $w \in E$ and $\bar{b} \in \mathbb{R}$ such that

$$X_- + B_-(0, r) \subset w < \bar{b} \text{ and } X_+ + B_+(0, r) \subset w > \bar{b}. \quad (3.2)$$

Then we classify point $x \in E$ according to the sign of $\langle w, x \rangle - \bar{b}$.

**Remark 1.** In practice we consider only balls which generate inner products (we call them ellipsoids). To define an ellipsoid we need a positive self-adjoint operator $S$, then we consider the Mahalanobis distance associated with it given by $\|x\|_S^2 := x^T S^{-1} x$, and the unit ellipsoid

$$B_S(0, 1) := S^{1/2} B(0, 1) = \{x : \|x\|_S < 1\}.$$  

Observe that formula (3.2) can be in fact reduced to the problem of searching for the supremum of $r > 0$ such that

$$[\text{conv}(X_-) + B_-(0, r)] \cap [\text{conv}(X_+) + B_+(0, r)] = \emptyset.$$  

The difference between this formulation and the classic one can be seen in Fig. 2, where two ellipsoidal sets are separated using the unit balls (Fig. 2a) and with two different ellipsoids (Fig. 2b).

We propose an alternative solution, which leads us to the formulation of 2eSVM. Let us first show that we can reduce separating by two margins $B_-$ and $B_+$ to separating by $B_- + B_+$.

**Proposition 1.** Let $B_-$ and $B_+$ be balls in $E$ and let $B = B_- + B_+$.

Let $r > 0$, $w$ be given. Then

$$\sup \{ \langle w, X_- \rangle \} + r/2 \sup \{ \langle w, B \rangle \} \leq \inf \{ \langle w, X_+ \rangle \} - r/2 \sup \{ \langle w, B \rangle \}. \quad (3.3)$$

iff

$$\sup \{ \langle w, X_- \rangle \} + r \sup \{ \langle w, B_+ \rangle \} \leq \inf \{ \langle w, X_+ \rangle \} - r \sup \{ \langle w, B_+ \rangle \}. \quad (3.4)$$

11
Proof. Observe that (3.3) holds iff
\[ \frac{r}{2} \sup \{ \langle w, B \rangle \} + \frac{r}{2} \sup \{ \langle w, B \rangle \} \leq \inf \{ \langle w, X \rangle \} - \sup \{ \langle w, X \rangle \}. \]

Analogically, (3.4) holds iff
\[ r \sup \{ \langle w, B \rangle \} + r \sup \{ \langle w, B \rangle \} \leq \inf \{ \langle w, X \rangle \} - \sup \{ \langle w, X \rangle \}. \]

Since \( B_- + B_+ = B = B_+/2 + B_-/2 \), we obtain the assertion of the proposition.

Following theorem and remark show how to calculate the shift of the decision boundary from the resulting separation with \( B \).

**Theorem 1.** Let \( B_- \) and \( B_+ \) be balls in \( E \) and let \( B = B_- + B_+ \).

Let \( r > 0, w \) be given. Assume that
\[ b := \sup \{ \langle w, X \rangle \} + r/2 \sup \{ \langle w, B \rangle \} = \inf \{ \langle w, X \rangle \} - r/2 \sup \{ \langle w, B \rangle \}. \]  

(3.5)

Then
\[ \sup \{ \langle w, X \rangle \} + r \sup \{ \langle w, B \rangle \} = \tilde{b} = \inf \{ \langle w, X \rangle \} - r \sup \{ \langle w, B \rangle \}, \]

(3.6)

where
\[ \tilde{b} := b - \frac{r}{\sup \{ \langle w, B \rangle \} + \sup \{ \langle w, B \rangle \}} \cdot \frac{1}{2} (\inf \{ \langle w, X \rangle \} - \sup \{ \langle w, X \rangle \}). \]

(3.7)

Proof. The equality of LHS and RHS in (3.6) follows directly from the previous proposition. Now we show the formula for \( \tilde{b} \). Observe that to prove (3.7), it is sufficient to show that
\[ \tilde{b} - b = \frac{r}{\sup \{ \langle w, B \rangle \} + \sup \{ \langle w, B \rangle \}} \cdot \frac{1}{2} (\inf \{ \langle w, X \rangle \} - \sup \{ \langle w, X \rangle \}). \]

By subtracting RHS of (3.6) from RHS of (3.5) we get
\[ \tilde{b} - b = r/2 \sup \{ \langle w, B \rangle \} - r \sup \{ \langle w, B \rangle \} = \]
\[ = r/2(\sup \{ \langle w, B \rangle \} + \sup \{ \langle w, B \rangle \}) - r \sup \{ \langle w, B \rangle \} = -r/2(\sup \{ \langle w, B \rangle \} - \sup \{ \langle w, B \rangle \}). \]

Since by (3.6)
\[ r = \frac{\inf \{ \langle w, X \rangle \} - \sup \{ \langle w, X \rangle \}}{\sup \{ \langle w, B \rangle \} + \sup \{ \langle w, B \rangle \}}, \]

we obtain the formula (3.7).
Since the previous theorem forms one of the basic steps in our algorithm, let us reformulate it in the SVM language.

**Remark 2.** The fact that we know how to separate with margin $B$, implies that we can find $w$ and scalar $b$ such that

$$\inf\{w \cdot X_+\} = b + 1, \sup\{w \cdot X_-\} = b - 1$$

and the size of the margin

$$\frac{2}{\|w\|_B}$$

is maximal. By the Theorem 1 the hyperplane (which optimally separates $X_+$ with margin function $r \rightarrow B_+(0, r)$ from $X_-$ with margin function $r \rightarrow B_-(0, r)$) is given by the formula $w_{\bar{b}} = \{x \in E : \langle w, x \rangle = \bar{b}\}$, where

$$\bar{b} = b - \frac{p_+ - p_-}{p_+ + p_-} \text{ and } p_\pm = \sup\{\langle w, B_\pm \rangle\}.$$

Formally, the obtained result considers only the linearly separable case, but it gives justification for using analogously defined shift for the non separable case. Remark 2 means that for optimal separation in terms of two different metrics we need to shift the location of the hyperplane from the middle point of a margin to the weighted sum of its limits. Fig. 2b makes it quite natural – once we know the $w$ (as a result of separating with $B$) we can find the correct $\bar{b}$ value according to the proportions of the balls $B_+$ and $B_-$. If $B$, $B_+$ and $B_-$ are ellipsoids, then the separation can be easily performed using SVM, and the shift would be proportional to the standard deviations of projections of points to the $w$:

$$\bar{b} = b - \frac{p_+ - p_-}{p_+ + p_-} \text{ and } p_\pm = \text{std}\{w^T x : x \in X_\pm\}.$$

The problem is that in general, even if $B_-$ and $B_+$ are ellipsoids, their algebraic sum $B_+ + B_-$ is rarely an ellipsoid. Consequently, as we have mentioned before, we cannot effectively separate by the margin $B$ using the standard SVM tools.

Thus in this part of the paper we construct an 2eSVM method by building an approximation of $B$ by an ellipsoid. We will need the following simple lemma.

**Lemma 1.** Let $S$ be a positive defined self-adjoint operator, $T$ self-adjoint and $m, c \in E$. Then the minimum of the function

$$z \rightarrow z^T Sz + m^T T z + z^T T m + c$$

equals

$$w = -m^T TS^{-1} T m + c$$

and is attained for $z_0 = -S^{-1} T m$. 

13
Proof. Observe that the above is an obvious consequence of the formula
\[ z^T S z + m^T T z + z^T T m + c = (z - z_0)^T S (z - z_0) + w \]
and the fact that \( S \) is positive defined. \( \square \)

Now we are going to proceed to the main theoretical result of our paper, which says that instead of using separation by the margin \( B = B_+ + B_- \), we can use its reasonable approximation by a hyper-ellipsoid, which enables us the use the whole existing SVM machinery.

**Theorem 2.** Let \( S_+ \) and \( S_- \) denote positively defined self-adjoint operators. Consider ellipsoids \( B_+ \) and \( B_- \). Let \( B = B_+ + B_- \) and \( S_0 = S_+ + S_- \). Then
\[ B_{S_0}(0, 1) \subset B(0, 1) \subset B_{S_0}(0, \sqrt{2}). \] (3.8)
Moreover, in general the bounds given in equation (3.8) cannot be improved.

**Proof.** Let us first observe that the norm for which \( B \) is the unit ball is given by
\[ \|x\|_B^2 := \inf \{ \max\{ \|x_+\|_{S_+}^2, \|x_-\|_{S_-}^2 \} \mid x_+, x_- \in E : x_+ + x_- = x \}. \]
Consider the norm defined by
\[ \|x\|_{B_0}^2 := \inf \{ \|x_+\|_{S_+}^2 + \|x_-\|_{S_-}^2 \mid x_+, x_- \in E : x_+ + x_- = x \}. \]
Let us first notice that, due to the inequality \( \frac{1}{2}(a^2 + b^2) \leq \max\{a^2, b^2\} \leq a^2 + b^2 \),
\[ \frac{1}{2} \|x\|_{B_0}^2 \leq \|x\|_B^2 \leq \|x\|_{B_0}^2. \]
Thus to prove the assertion of the theorem it is sufficient to prove that \( \|\cdot\|_{B_0} = \|\cdot\|_{S_0} \). Put \( R_+ = S_+^{-1}, \ R_- = S_-^{-1}. \)
Then we have
\[ \|x\|_{B_0}^2 = \inf \{ \|x_+\|_{S_+}^2 + \|x_-\|_{S_-}^2 \mid x_+, x_- \in E : x_+ + x_- = x \} = \inf \{ x^T R_+ x_+ + x^T R_- x_- \mid x_+, x_- \in E : x_+ + x_- = x \} = \inf \{ x^T R_+ x_+ + (x - x_+)^T R_- (x - x_+) \mid x_+ \in E \} = \inf \{ x^T R_+ (R_+ + R_-) x_+ - x^T R_- x_+ - x^T R_- x + x^T R_- x \mid x_+ \in E \}. \]
By Lemma 1 we conclude that the minimum of the above function equals
\[ -x^T R_- (R_+ + R_-)^{-1} R_- x + x^T R_+ x = x^T [-R_- (R_+ + R_-)^{-1} R_- + R_+] x. \]
Since
\[-R_\varepsilon (R_+ + R_-)^{-1}R_\varepsilon + R_\varepsilon = R_\varepsilon (R_+ + R_-)^{-1}(-R_\varepsilon) + R_\varepsilon (R_+ + R_-)^{-1}(R_+ + R_-)\]
\[= R_\varepsilon (R_+ + R_-)^{-1}R_\varepsilon = [R_\varepsilon^{-1}(R_+ + R_-)^{-1}]^{-1} = (R_\varepsilon^{-1} + R_-^{-1})^{-1},\]
we obtain that
\[
\|x\|_0^2 = x^T \left(R_\varepsilon^{-1} + R_-^{-1}\right)^{-1}x = x^T \left(S_+ + S_-\right)^{-1}x = x^T S_0^{-1}x = \|x\|_{S_0},
\]
which completes the proof of the estimation (3.8). The proof that one cannot improve those bounds (even if the operators $S_+$ and $S_-$ commute) is given in the following example.

**Example 1.** Consider the following positively defined operators on $\mathbb{R}^2$

\[
S_+ = \begin{bmatrix} 1 & 0 \\ 0 & \varepsilon \end{bmatrix}, \quad S_- = \begin{bmatrix} \varepsilon & 0 \\ 0 & 1 \end{bmatrix},
\]

with small $\varepsilon > 0$. Observe that the unit balls with $\varepsilon \to 0$ converge to the vertical and horizontal intervals $[-1, 1] \times \{0\}$ and $\{0\} \times [-1, 1]$. This implies that $B_{S_+}(0, 1) + B_{S_-}(0, 1)$ converges to the square $[-1, 1] \times [-1, 1]$. By trivial calculations we see that

\[
S_0 = S_+ + S_- = \begin{bmatrix} 1 + \varepsilon & 0 \\ 0 & 1 + \varepsilon \end{bmatrix},
\]

which means that $B_{S_0}(0, 1)$ converges to the unit circle $B(0, 1)$. Since

\[
B(0, 1) \subset [-1, 1]^2 \subset B(0, \sqrt{2}),
\]

and none of the constants of the above inclusion can be improved, we obtain that in general the estimations from the previous proposition cannot be improved, Fig. 6c visualizes such unit balls.

We showed a generalization of the SVM problem and its theoretical, computationally expensive solution, as well as an easily computable, sufficient approximation. Fig. 6 shows how proposed method approximates the optimal solution – from the optimistic situation, where resulting $B$ has nearly elliptic shape (Fig. 6a), through the average case when it is more parallelogram like (Fig. 6b) to the hardest case, when it degenerates to square like unit ball (Fig. 6c). Now we focus on practical aspects of such an approach by first providing general implementation scheme in Section 4 and then detailed evaluation in Section 6.
Figure 6: Sample ellipses $B_+ (0, 1)$ and $B_- (0, 1)$ (crossing at the origin of each image), the $B(0, 1) = B_+ (0, 1) + B_- (0, 1)$ (black stroke shape) and approximating ellipses $B_{S0} (0, 1)$ (darker ellipse) and $B_{S0} (0, \sqrt{2})$ (lighter one)

4. Two Ellipsoids SVM formulation

In the previous section we have shown, that the sum of classes’ covariances defines a scalar product which is a good approximation of the corresponding ellipsoids algebraic sum. As a result, we can define linear form of our problem in the language of the SVM optimization by simply substituting the Euclidean distance with the Mahalanobis one using summarized covariance.

Optimization formulation: linear 2eSVM

$$\begin{align*}
\text{minimize} & \quad \frac{1}{2} \| w \|^2 + \sum C \xi_i \\
\text{subject to} & \quad y_i (\langle w, x_i \rangle \Sigma_+ + \Sigma_- - b) \geq 1 - \xi_i, \; i = 1, \ldots, N
\end{align*}$$

For an efficient implementation, one can exploit previously mentioned fact that solving the problem with the Mahalanobis distance with covariance $\Sigma_+ + \Sigma_-$ is equivalent to solving the one with Euclidean distance after transformation through $(\Sigma_+ + \Sigma_-)^{-1/2}$. Algorithm 1 shows the pseudocode of such an approach.

Let us now assume that we are given some kernel function $K(\cdot, \cdot)$ such that $K(x, y) = \phi(x)^T \phi(y)$ for some feature space projection function $\phi(\cdot)$. We need to compute the sum of corresponding covariance matrices for samples in the feature space ($\Sigma_{\phi, -}$ and $\Sigma_{\phi, +}$). These matrices may be defined as follows:

$$\Sigma_{\phi, k} = \phi(X_k) J_k J_k^T \phi(X_k)^T, \; k \in \{ - , + \}$$

$$J_k = \frac{1}{\sqrt{n_k}} \left( I_{n_k} - \frac{1}{n_k} e_{n_k} \right), \; k \in \{ - , + \}$$
Algorithm 1 Linear 2eSVM implementation

1:  \textbf{procedure} \textsc{Train}(X_{+}, X_{-})
2: \hspace{1em} \Sigma_{k} \leftarrow \text{cov}(X_{k}), k \in \{+,-\} \quad \triangleright \text{computing the covariance matrices}
3: \hspace{1em} \Sigma_{0} \leftarrow \Sigma_{+} + \Sigma_{-} \quad \triangleright \text{creating transformation matrix}
4: \hspace{1em} \bar{X}_{k} \leftarrow \Sigma_{0}^{-1/2}X_{k}, k \in \{+,-\} \quad \triangleright \text{mapping data to the feature space}
5: \hspace{1em} w, b \leftarrow \text{SVM}(\bar{X}_{+}, \bar{X}_{-}) \quad \triangleright \text{running classical SVM, any implementation}
6: \hspace{1em} P_{k} \leftarrow w^{T}\bar{X}_{k}, k \in \{+,-\} \quad \triangleright \text{projecting data to the normal of the optimal hyperplane}
7: \hspace{1em} p_{k} \leftarrow \text{std}(P_{k}), k \in \{+,-\} \quad \triangleright \text{calculating standard deviations of points projections}
8: \hspace{1em} \bar{b} \leftarrow b - \frac{p_{+}-p_{-}}{p_{+}+p_{-}} \quad \triangleright \text{shifting the boundary proportionally to the stds}
9: \hspace{1em} \textbf{return} w, \bar{b}, \Sigma_{0} \quad \triangleright \text{we have to store the } \Sigma_{0} \text{ matrix}
10: \textbf{end procedure}
11:
12: \textbf{procedure} \textsc{Predict}(w, \bar{b}, \Sigma_{0}, x)
13: \hspace{1em} \textbf{return} \text{sgn}(w^{T}\Sigma_{0}^{-1/2}x - \bar{b}) \quad \triangleright \text{SVM prediction with projected data and new } b
14: \textbf{end procedure}

where \( n_{k} \) is the number of samples in \( k \) class, \( I_{l} \) is the \( l \times l \) identity matrix and \( e_{l} \) is the \( l \times l \) matrix of ones. Since \( \Sigma_{\phi,k} \) are often ill-conditioned, to obtain the inverse of their sum we add a small positive constant \( \varepsilon \) to the diagonal of \( \Sigma_{\phi,-} + \Sigma_{\phi,+} \). Thus, we actually compute the inverse of \( \varepsilon I_{l} + \Sigma_{\phi,-} + \Sigma_{\phi,+} \).

From Woodbury matrix identity [19]

\[
\left( \left[ U + V V_{+}^{T} \right] + V_{-}V_{+}^{T} \right)^{-1} = \\
= \left[ U + V V_{-}^{T} \right]^{-1} - \left[ U + V V_{-}^{T} \right]^{-1}V_{-}V_{+}^{T} \left( I + V_{+}^{T} \left[ U + V V_{-}^{T} \right]^{-1}V_{-} \right)^{-1}V_{+}^{T} \left[ U + V V_{-}^{T} \right]^{-1},
\]

using substitution \( V_{+} = \phi(X_{+})^I, \Sigma_{\phi,+} = \varepsilon I_{l} + \Sigma_{\phi,+} = U + V_{-}^{T}V_{-} \), we have

\[
K_{2e}(x,y) = \phi(x)^{T}(\varepsilon I_{l} + \Sigma_{\phi,-} + \Sigma_{\phi,+})^{-1}\phi(y) = \\
= \phi(x)^{T}\Sigma_{\phi,-}^{-1}\phi(y) - \phi(x)^{T}\Sigma_{\phi,+}^{-1}V_{-}(I + V_{+}^{T}\Sigma_{\phi,-}^{-1}V_{-})^{-1}V_{+}^{T}\Sigma_{\phi,-}^{-1}\phi(y). 
\]

But \( \phi(x)^{T}\Sigma_{\phi,-}^{-1}\phi(y) \) is a simple Mahalanobis kernel with the negative class covariance \( K_{-}(x,y) \) [5] (with added small constant on the diagonal). Analogically, we reduce remaining terms in the form of \( A^{T}\Sigma_{\phi,-}^{-1}B \) to the corresponding kernel values \( K_{-}(A,B) \):
\[ K_{2e}(x,y) = K_-(x,y) - K_-(X_+ J_+ (I + J_+^T K_-(X_+, X_+) J_+)^{-1} J_+^T K_-(X_+, y)) \]

\[ K_-(x,y) = \varepsilon^{-1} K(x,y) - \varepsilon^{-1} K(x,X) J_+ \left( \varepsilon I + J_+^T K(X_-, X_+) J_+ \right)^{-1} J_+^T K(X_-, y) \].

Similarly to the construction of Mahalanobis kernel \( K_-( \) (which is based on the covariance of the negative class, for full set covariance one would have to use \( X \) in place of \( X_+ \) in the above formula) from any kernel \( K \), we can obtain the two ellipsoids kernel \( K_{2e} \) from \( K_- \). Therefore we have the following quadratic optimization problem:

**Optimization formulation: kernelized 2eSVM**

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} K_{2e}(w, w) + C \sum_{i=1}^{N} \xi_i \\
\text{subject to} & \quad y_i (K_{2e}(w, x_i) - b) \geq 1 - \xi_i, \ i = 1, \ldots, N
\end{align*}
\]

This leads to the dual form kernelized 2eSVM formulation

\[
\begin{align*}
\text{maximize} & \quad \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K_{2e}(x_i, x_j) \\
\text{subject to} & \quad 0 \leq \alpha_i \leq C, \ i = 1, \ldots, N, \\
& \quad \sum_{i=1}^{N} \alpha_i y_i = 0
\end{align*}
\]

which can be directly solved with any SVM implementation capable of using custom kernel (preferably in the form of precomputed Gram matrix). For an efficient implementation kernel computation can be fully vectorized and all the required matrix inversions can be cached beforehand. We put

\[ G := K(X,X), \]

\[ G_- := K_-(X,X) = \varepsilon^{-1} G - \varepsilon^{-1} G |X_+ \times X_+ J_+ (I + J_+^T G |X_+ \times X_+ J_+)^{-1} J_+^T G |X_+ \times X_+, \]

\[ G_{2e} := K_{2e}(X,X) = G_- - G_- |X_+ \times X_+ J_+ (I + J_+^T G_- |X_+ \times X_+ J_+)^{-1} J_+^T G_- |X_+ \times X_+, \]

where by \( G_k |A \times B \) we denote the Gram matrix \( G_k \) restricted to the scalar products between elements from the sets \( A \) and \( B \). This leads to the preprocessing step complexity of \( O(N^{2.381}) \) (fast matrix inversion/multiplication).

This is the same complexity as the one required by analogous Mahalanobis kernel, which makes it reasonable to compare with this model.
5. Scalability

First, we show that proposed method scales well in the linear case. One has to perform following operations:

1. computation of the data covariance matrix,
2. computation of the inversion and square root of the correlation matrix,
3. dot product between transformed covariance matrix and data.

Let us assume that we are given data $X \in \mathbb{R}^{N \times d}$. Using naive approach, first operation can be performed in $O(Nd^2)$ operations. As a result, it scales linearly with the number of points. The quadratic complexity in terms of the data dimension limits the application to at most thousands of dimensions, however in practise higher dimensional data (like the one in the cheminformatics or Natural Language Processing) is often sparse, which makes even a naive covariance computation much faster ($O(d^2s)$ if for each feature, at most $s$ vectors have nonzero value). Covariance matrix $\Sigma$ is a square matrix of dimension $d$, so its inversion and square rooting (operation 2.) require at most $O(d^3)$ operations with naive methods, but can be reduced by more advanced algorithms. In particular, modern algorithms are able to compute square root of a $2000 \times 2000$ matrix in under 20 seconds in a single thread mode [20]. Dot product operation, similarly to the operation 1. can be computed in $O(Nd^2)$ and can greatly benefit from data sparseness (reducing to at least $O(Nds)$ if each training vector has at most $s$ nonzero values).

Remark 3. The computational complexity of a preprocessing in a linear case $2e\mathcal{SVM}$ of $N$ training vectors in $\mathbb{R}^d$ is $O((N+d)d^2)$. If data is sparse in the sense that each vector has at most $s$ nonzero feature values and each feature has at most $s$ nonzero vectors it is reduced to $O((N+d)ds + d^3)$.

Proof. The complexity comes directly from the sequential characteristics of the algorithm and the compounds complexities $O(Nd^2 + d^3 + Nd^2) = O((N+d)d^2)$. Similarly for the sparse case $O(sd^2 + d^3 + Nds) = O((N+d)d^2 + d^3)$.

It is worth noting, that each of the required operations (matrix dot product, inverse and square rooting) can be easily parallelized, making use of multi-core processors available in every modern computer. Speedup for the most expensive parts is nearly linear in number of cores ($k$ cores reduce computational time nearly $k$ times [20]).

Kernelized case has a similar cost to other kernel machines (including Support Vector Machines and Extreme Learning Machines). First of all, it requires computation of the Gram matrix, which by its definition
is $O(N^2)$. Analogically to the Mahalanobis Support Vector Machine, one has to invert a Gram matrix, which is $O(N^3)$ in a naive implementation and $O(N^{2.807})$ with Strassen algorithm [21] to even $O(N^{2.373})$ using William’s algorithm [22].

**Remark 4.** The computational complexity of a computation of the 2eSVM Gram matrix of $N$ training vectors using kernel computable in time $K$ is $O(N^{2.373} + N^2 K)$.

**Proof.** The complexity comes directly from the equation for $G_{2e}$, and the fact that the most numerically expensive parts are multiplications and inversions of $N \times N$ matrices. The starting point of the algorithm is the Gram matrix of the used kernel, which can be computed in $\Theta(N^2 K)$.

This remark leads to the following important scaling observation.

**Remark 5.** Kernelized 2eSVM adds asymptotically no overhead as compared to the traditional SVM training on $N$ data points if computation of the kernel between two arbitrary vectors takes at least $\omega(N^{0.373})$ and never adds overhead as compared to the traditional Mahalanobis SVM training.

**Proof.** The classical SVM requires computation of the Gram matrix, which takes $\Theta(N^2 K)$, so if $K \in \omega(N^{0.373})$ then $O(N^2 K) = O(N^2(K + N^{0.373})) = O(N^{2.373} + N^2 K)$. No overhead as compared to the Mahalanobis SVM comes from the fact that this model also requires inversion of the Gram matrix.

### 6. Evaluation

Proposed approach consists of the pre- and postprocessing methods, which can work in the Black Box scenario using any existing SVM implementation. For this reason we do not compare our method to models which alternate the optimization problem and require custom implementation (like Twin Mahalanobis SVM [5], Ellipsoidal Kernel Machine [10] or Maxi-Min Margin Machine [6]). In fact, 2eSVM (analogously like SVM or its basic adaptations) would be inferior in terms of resulting models quality, but this is not the aim of our research. We are proposing the method, implementable by non-specialists, usable with existing, very efficient SVM implementations and show how it outperforms the commonly used Mahalanobis pre-processing (Mahalanobis kernel). Such narrowing gives us also the objective comparision in terms of training time as we can express it in terms of the exact same optimization algorithm’s iterations (comparision between two different implementations/models not neceserly shows the superiority of one method, but rahter – better code). One more important aspect is the fact, that 2eSVM and Mahalanobis based SVM are working in the similar way, on the data in the analogous scale. As a result, using the same parameters in grid search does
not favor any of them – again, without such a guarantee, better results could be the effect of more detailed parameters selection (better suited for a particular method) instead of actual superiority.

Tests were performed using code written in Python with use of the scikit-learn (sklearn)\footnote{http://scikit-learn.org} [23] and scipy libraries [24] on the Intel i7, 8 core, 2.3Ghz processor machine. We used the model based on the one of the most popular libraries libsvm (SVC class in sklearn) to ensure repeatable and practical results. We compare our method with SVM that uses the whole set covariance as $\Sigma_0$ (referred to as cSVM) as it is a very popular and similar concept, with exactly the same computational complexity, additional memory requirements and similar behavior (so we can use the same range and resolution of the parameters grid search without favoring any of them).

6.1. Linear case

Experiments were performed on ten datasets from UCI library [25] (see Table 1 for reference), each using following scheme. For each considered score function we randomly split dataset into training part (66%) $T_i$ and testing part (33%) $\bar{T}_i$ for each $i \in 1, \ldots, 2500$. Next, for each of such splits $(T_i, \bar{T}_i)$ each of models independently searches for best parameters using the 5-fold cross-validation technique on the $T_i$ dataset (for currently considered score function value averaged across those 5 folds). Each model is then retrained using its best parameters on the whole $T_i$ dataset and evaluated on the never seen before $\bar{T}_i$ dataset. Parameters space was searched using simple adaptive mesh refinement technique (in form of static grid search followed by two grid searches around the best parameters with higher resolution) for more efficient search. For unbalanced datasets we used the class weighting technique, where $C$ parameter is replaced with $C_+$ and $C_-$ parameters inversely proportional to class frequencies.

As one can see in Table 1 all of the considered datasets consist of classes with different internal geometry. We also report the difference between the sum of covariances (used in our method) and the whole covariance (used in Mahalanobis approach), which shows, that we use significantly different scalar product.

Proposed method achieved slightly better classification results in terms of accuracy and Mathew’s Correlation Coefficient (MCC) in most of the datasets. These differences are statistically significant in 90% of conducted experiments. Much more noticeable difference between compared methods is the number of iterations during training. Depending on the dataset one can observe between two and four times faster training in terms of optimization’s routine iterations (which is a more reliable measure than the empirical
### Table 1: Characteristics of used datasets. We report Frobenious norms. We use the following notation $\Sigma_{2e} = \Sigma_+ + \Sigma_-$

<table>
<thead>
<tr>
<th>dataset</th>
<th>d</th>
<th>n_+</th>
<th>n_-</th>
<th>$|\Sigma_+ - I|_F$</th>
<th>$|\Sigma_- - I|_F$</th>
<th>$|\Sigma_+ - \Sigma_-|_F$</th>
<th>$|\Sigma_{2e} - \Sigma|_F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>australian</td>
<td>14</td>
<td>383</td>
<td>307</td>
<td>1.000</td>
<td>1.000</td>
<td>0.985</td>
<td>0.370</td>
</tr>
<tr>
<td>bank</td>
<td>4</td>
<td>762</td>
<td>610</td>
<td>0.915</td>
<td>0.945</td>
<td>0.282</td>
<td>0.291</td>
</tr>
<tr>
<td>breast cancer</td>
<td>9</td>
<td>444</td>
<td>239</td>
<td>0.422</td>
<td>0.829</td>
<td>0.765</td>
<td>0.391</td>
</tr>
<tr>
<td>crashes</td>
<td>20</td>
<td>46</td>
<td>494</td>
<td>0.998</td>
<td>0.998</td>
<td>0.028</td>
<td>0.344</td>
</tr>
<tr>
<td>diabetes</td>
<td>8</td>
<td>268</td>
<td>500</td>
<td>1.000</td>
<td>1.000</td>
<td>0.323</td>
<td>0.369</td>
</tr>
<tr>
<td>german number</td>
<td>24</td>
<td>700</td>
<td>300</td>
<td>0.991</td>
<td>0.995</td>
<td>0.348</td>
<td>0.385</td>
</tr>
<tr>
<td>heart</td>
<td>13</td>
<td>150</td>
<td>120</td>
<td>0.998</td>
<td>0.998</td>
<td>0.127</td>
<td>0.319</td>
</tr>
<tr>
<td>liver-disorders</td>
<td>6</td>
<td>145</td>
<td>200</td>
<td>0.997</td>
<td>0.998</td>
<td>0.238</td>
<td>0.310</td>
</tr>
<tr>
<td>sonar</td>
<td>60</td>
<td>111</td>
<td>97</td>
<td>0.634</td>
<td>0.651</td>
<td>0.326</td>
<td>0.322</td>
</tr>
<tr>
<td>splice</td>
<td>60</td>
<td>483</td>
<td>517</td>
<td>0.304</td>
<td>0.243</td>
<td>0.303</td>
<td>0.328</td>
</tr>
</tbody>
</table>

Computation time which despite being dependent on many non-algorithm related elements, also includes non optimization related elements), see Table 2.

In the Figure 7 we plotted the kernel density estimation of the distribution of the iterations count during all the experiments for the breast-cancer dataset. This shows not only that our method had less instances with the highest computational cost (the most significant peak of the plot) but also resulted in many instances which are much simpler than the simplest ones in the $cSVM$ training.

We further investigated this phenomenon as our experiments may be seen as triple-phased. We first search for the most promising parameters using low resolution grid search and then for two best parameters (in terms of cross-validation) we look for the parameters with greater resolution around them. We report mean iterations amount for each of these phases in Table 3. The most important observation is the fact that around the best parameter, training time speedup is significantly greater than during the low resolution grid search. This shows that the best solution is obtained during the faster training, so the experiments confirm our claim that for correctly selected parameters $2eSVM$ simplifies the geometry of the problem and leads to the easier optimization. As it was previously described in the Section 2 it is a result of better choice of the scalar product. Data in the feature space is more shattered and so, easier to separate by the SVM optimization algorithm.
Table 2: Mean scores after 2500 experiments and training times with corresponding mean and median iterations count. Bolded values indicate statistically significant difference (in terms of t-Student test).

<table>
<thead>
<tr>
<th>dataset</th>
<th>accuracy</th>
<th>MCC median</th>
<th>median time [s]</th>
<th>mean iterations</th>
<th>mean time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2eSVM</td>
<td>cSVM</td>
<td>2eSVM</td>
<td>cSVM</td>
<td>2eSVM</td>
</tr>
<tr>
<td>australian</td>
<td>0.848</td>
<td>0.848</td>
<td><strong>0.720</strong></td>
<td>0.718</td>
<td><strong>332</strong></td>
</tr>
<tr>
<td></td>
<td>0.011</td>
<td>0.032</td>
<td></td>
<td></td>
<td><strong>0.012</strong></td>
</tr>
<tr>
<td>bank</td>
<td>0.990</td>
<td><strong>0.991</strong></td>
<td>0.979</td>
<td><strong>0.981</strong></td>
<td><strong>105</strong></td>
</tr>
<tr>
<td></td>
<td>0.008</td>
<td>0.011</td>
<td></td>
<td></td>
<td><strong>0.009</strong></td>
</tr>
<tr>
<td>breast cancer</td>
<td><strong>0.960</strong></td>
<td>0.954</td>
<td><strong>0.939</strong></td>
<td>0.931</td>
<td><strong>186</strong></td>
</tr>
<tr>
<td></td>
<td>0.004</td>
<td>0.004</td>
<td></td>
<td></td>
<td><strong>0.004</strong></td>
</tr>
<tr>
<td>crashes</td>
<td><strong>0.864</strong></td>
<td>0.858</td>
<td><strong>0.591</strong></td>
<td>0.559</td>
<td><strong>297</strong></td>
</tr>
<tr>
<td></td>
<td>0.003</td>
<td>0.004</td>
<td></td>
<td></td>
<td><strong>0.003</strong></td>
</tr>
<tr>
<td>diabetes</td>
<td>0.804</td>
<td><strong>0.808</strong></td>
<td><strong>0.473</strong></td>
<td>0.470</td>
<td><strong>427</strong></td>
</tr>
<tr>
<td></td>
<td>0.012</td>
<td>0.023</td>
<td></td>
<td></td>
<td><strong>0.013</strong></td>
</tr>
<tr>
<td>german number</td>
<td><strong>0.720</strong></td>
<td>0.711</td>
<td>0.402</td>
<td>0.402</td>
<td><strong>604</strong></td>
</tr>
<tr>
<td></td>
<td>0.021</td>
<td>0.038</td>
<td></td>
<td></td>
<td><strong>0.031</strong></td>
</tr>
<tr>
<td>heart</td>
<td>0.830</td>
<td><strong>0.832</strong></td>
<td>0.657</td>
<td>0.662</td>
<td><strong>161</strong></td>
</tr>
<tr>
<td></td>
<td>0.002</td>
<td>0.004</td>
<td></td>
<td></td>
<td><strong>0.003</strong></td>
</tr>
<tr>
<td>liver disorders</td>
<td><strong>0.695</strong></td>
<td>0.679</td>
<td><strong>0.318</strong></td>
<td>0.311</td>
<td><strong>586</strong></td>
</tr>
<tr>
<td></td>
<td>0.002</td>
<td>0.002</td>
<td></td>
<td></td>
<td><strong>0.003</strong></td>
</tr>
<tr>
<td>sonar</td>
<td><strong>0.724</strong></td>
<td>0.717</td>
<td><strong>0.452</strong></td>
<td>0.438</td>
<td><strong>141</strong></td>
</tr>
<tr>
<td></td>
<td>0.002</td>
<td>0.003</td>
<td></td>
<td></td>
<td><strong>0.002</strong></td>
</tr>
<tr>
<td>splice</td>
<td><strong>0.802</strong></td>
<td>0.799</td>
<td><strong>0.604</strong></td>
<td>0.601</td>
<td><strong>599</strong></td>
</tr>
<tr>
<td></td>
<td>0.067</td>
<td>0.113</td>
<td></td>
<td></td>
<td><strong>0.108</strong></td>
</tr>
</tbody>
</table>
Table 3: Mean iterations amount (and time in seconds below each one) in different phases of each experiments. With mean/median ratio we denote the mean/median of ratio of the mean iterations’ count compared to cSVM’s for each dataset in a particular phase.

<table>
<thead>
<tr>
<th>dataset</th>
<th>low res gridsearch</th>
<th>around best</th>
<th>around second best</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2eSVM</td>
<td>cSVM</td>
<td>2eSVM</td>
</tr>
<tr>
<td>australian</td>
<td>9570</td>
<td>21390</td>
<td>747</td>
</tr>
<tr>
<td></td>
<td>0.015</td>
<td>0.038</td>
<td>0.011</td>
</tr>
<tr>
<td>bank</td>
<td>414</td>
<td>729</td>
<td>150</td>
</tr>
<tr>
<td></td>
<td>0.009</td>
<td>0.012</td>
<td>0.009</td>
</tr>
<tr>
<td>breast cancer</td>
<td>603</td>
<td>1500</td>
<td>474</td>
</tr>
<tr>
<td></td>
<td>0.004</td>
<td>0.005</td>
<td>0.004</td>
</tr>
<tr>
<td>crashes</td>
<td>891</td>
<td>1677</td>
<td>612</td>
</tr>
<tr>
<td></td>
<td>0.003</td>
<td>0.005</td>
<td>0.003</td>
</tr>
<tr>
<td>diabetes</td>
<td>2364</td>
<td>4851</td>
<td>1392</td>
</tr>
<tr>
<td></td>
<td>0.013</td>
<td>0.025</td>
<td>0.013</td>
</tr>
<tr>
<td>german number</td>
<td>11511</td>
<td>24729</td>
<td>8010</td>
</tr>
<tr>
<td></td>
<td>0.032</td>
<td>0.062</td>
<td>0.029</td>
</tr>
<tr>
<td>heart</td>
<td>1569</td>
<td>3129</td>
<td>864</td>
</tr>
<tr>
<td></td>
<td>0.003</td>
<td>0.005</td>
<td>0.002</td>
</tr>
<tr>
<td>liver disorders</td>
<td>933</td>
<td>1764</td>
<td>2259</td>
</tr>
<tr>
<td></td>
<td>0.002</td>
<td>0.003</td>
<td>0.003</td>
</tr>
<tr>
<td>sonar</td>
<td>360</td>
<td>1098</td>
<td>186</td>
</tr>
<tr>
<td></td>
<td>0.002</td>
<td>0.003</td>
<td>0.002</td>
</tr>
<tr>
<td>splice</td>
<td>55518</td>
<td>116616</td>
<td>9966</td>
</tr>
<tr>
<td></td>
<td>0.150</td>
<td>0.225</td>
<td>0.081</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>mean ratio</th>
<th>mean time ratio</th>
<th>median ratio</th>
<th>median time ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.473</td>
<td>1.0</td>
<td>0.355</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>0.618</td>
<td>1.0</td>
<td>0.592</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>0.481</td>
<td>1.0</td>
<td>0.249</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>0.628</td>
<td>1.0</td>
<td>0.415</td>
<td>1.0</td>
</tr>
</tbody>
</table>

24
6.2. Kernelized case

For evaluation of the 2eSVM kernelization we used real-life data. Experiments were performed on the cheminformatics data regarding ligands activity prediction [17] for the 5-HT$_{2A}$ serotonin receptor protein. In this problem, each data point is a chemical compound, represented as fixed length vector (called a fingerprint, see Figure 8). The task is to predict whether given compound does (positive samples) or does not (negative samples) react with the target protein. Both positive and negative samples are compounds of confirmed activity (inactivity) forming a datasets of 2686 molecules (68% of which are elements of positive class).

![Figure 8: Sample fingerprint of the chemical molecule. Each key is a predicate, which checks for example: existance of a particular substructure, required number of some atoms etc.](image)

The problem itself is very hard and scientific community have not yet found the best way of compounds representation [26]. For this reason many kinds of fingerprints exist, often applicable for different purposes. In our experiments we investigated five diverse data representations summarized in the Table 4.

![Table 4](image)

Since analyzed fingerprints are general use representations some of the features in the considered dataset equal 0 for all datapoints. Consequently the reduced problem’s dimension is reported as $d'$ in the Table 4. Even after this process, data dimension varies between 84 and 1009 dimensions. One more important char-
Table 4: Characteristics of the datasets used in the task of 5-HT$_{2A}$ protein chemical compounds activity prediction. Last column represents the ratio of data points, which are inconsistent (has exactly the same representation and different labels) calculated as the Jaccard coefficient.

| dataset   | d   | d’   | short fingerprint description                                                                 | $\frac{|X_\cap X'_+|}{|X_+ \cup X'_+|}$ |
|-----------|-----|------|-----------------------------------------------------------------------------------------------|----------------------------------------|
| ExtFP     | 1024| 1009 | Molecular graph is searched through using BFS and paths of maximum length of 6 atoms are hashed and stored in the 1024 bit vector. | 0.005                                  |
| KlekFP    | 4860| 816  | Very rich substructural representation, representing presence (or absence) of thousands of known substructures in the molecule. | 0.012                                  |
| PubchemFP | 881 | 541  | A substructure fingerprint based on structural keys – similar to the MACCSCFP, but using different set of keys. | 0.010                                  |
| MACCSFP   | 166 | 115  | A fingerprint based on the MACCS keys. Each bit is a representation of the presence (or the absence) of particular atoms, bonds, groups, properties, etc. in the chemical structure. | 0.029                                  |
| SubFP     | 307 | 84   | A fingerprint based on the presence of the SMART patterns [27], developed by Christian Laggner. | 0.134                                  |
acteristics of the data is its inconsistency – there exist compounds which have exactly the same fingerprint representation and appear in both positive and negative sets – see the last column of the mentioned table.

We analyzed two, most popular, kernels – the RBF and polynomial (of degree 2). Table 5 summarizes the results of the performed experiments, all values are taken from 10-fold cross-validation from the grid search of best parameters \( C \) and \( \gamma \) in case of RBF and \( C \) in case of polynomial. For both tested methods we have precomputed the Gram matrices using the formulas from the previous section.

**Table 5:** Results of evaluation of kernelized version of 2eSVM compared to the kernelized cSVM in the task of 5-HT\(_{2A}\) protein chemical compounds activity prediction.

<table>
<thead>
<tr>
<th>fingerprint (kernel)</th>
<th>2eSVM accuracy</th>
<th>cSVM accuracy</th>
<th>MCC median</th>
<th>iteration mean</th>
<th>median time [s]</th>
<th>mean time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExtFP (rbf)</td>
<td>0.893</td>
<td>0.893</td>
<td>0.751</td>
<td>0.748</td>
<td>2413</td>
<td>5402</td>
</tr>
<tr>
<td>KlekFP (rbf)</td>
<td>0.886</td>
<td>0.884</td>
<td>0.735</td>
<td>0.728</td>
<td>3106</td>
<td>5227</td>
</tr>
<tr>
<td>PubchemFP (rbf)</td>
<td>0.890</td>
<td>0.889</td>
<td>0.745</td>
<td>0.740</td>
<td>3004</td>
<td>5075</td>
</tr>
<tr>
<td>MACCSFP (rbf)</td>
<td>0.878</td>
<td>0.875</td>
<td>0.714</td>
<td>0.707</td>
<td>3577</td>
<td>4421</td>
</tr>
<tr>
<td>SubFP (rbf)</td>
<td>0.824</td>
<td>0.824</td>
<td>0.600</td>
<td>0.579</td>
<td>2556</td>
<td>2954</td>
</tr>
<tr>
<td>ExtFP (poly)</td>
<td>0.882</td>
<td>0.881</td>
<td>0.724</td>
<td>0.723</td>
<td>3126</td>
<td>5708</td>
</tr>
<tr>
<td>KlekFP (poly)</td>
<td>0.882</td>
<td>0.869</td>
<td>0.723</td>
<td>0.694</td>
<td>1490</td>
<td>5490</td>
</tr>
<tr>
<td>PubchemFP (poly)</td>
<td>0.870</td>
<td>0.862</td>
<td>0.700</td>
<td>0.672</td>
<td>1542</td>
<td>5874</td>
</tr>
<tr>
<td>MACCSFP (poly)</td>
<td>0.866</td>
<td>0.861</td>
<td>0.689</td>
<td>0.673</td>
<td>2575</td>
<td>7223</td>
</tr>
<tr>
<td>SubFP (poly)</td>
<td>0.825</td>
<td>0.814</td>
<td>0.591</td>
<td>0.550</td>
<td>21087</td>
<td>31496</td>
</tr>
</tbody>
</table>

Obtained results are similar to that of the linear case. First, one can notice slight increase in the quality of resulting classification (mostly in terms of the Mathew’s Correlation Coefficient), although scores are
similar, this suggest that indeed the use of two different margin functions can be beneficial in terms of classifier quality. The most noticeable difference is in the number of iterations required for the SVM to converge. One can observe from 1.5 to 3 times faster training, coming only from the custom data shattering induced by the $K_{2e}$ kernel. We underline that compared methods have the same computational complexity, so the number of iterations is an objective measure of training speed.

Table 6: Number of support vectors in different experiments. With best model we denote the model achieving best 10-fold accuracy score.

<table>
<thead>
<tr>
<th>fingerprint (kernel)</th>
<th>best model</th>
<th></th>
<th></th>
<th>whole experiment</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>median no. SVs</td>
<td>mean no. SVs</td>
<td>median no. SVs</td>
<td>mean no. SVs</td>
<td>median no. SVs</td>
<td>mean no. SVs</td>
</tr>
<tr>
<td>2eSVM</td>
<td>cSVM</td>
<td>2eSVM</td>
<td>cSVM</td>
<td>2eSVM</td>
<td>cSVM</td>
<td>2eSVM</td>
</tr>
<tr>
<td>ExtFP (rbf)</td>
<td>2366</td>
<td>2371</td>
<td>2361</td>
<td>2372</td>
<td>2228</td>
<td>2366</td>
</tr>
<tr>
<td>KlekFP (rbf)</td>
<td>2248</td>
<td>2339</td>
<td>2244</td>
<td>2339</td>
<td>1688</td>
<td>2048</td>
</tr>
<tr>
<td>PubchemFP (rbf)</td>
<td>2223</td>
<td>2333</td>
<td>2223</td>
<td>2331</td>
<td>1837</td>
<td>2016</td>
</tr>
<tr>
<td>MACCSFP (rbf)</td>
<td>2008</td>
<td>2116</td>
<td>2009</td>
<td>2112</td>
<td>1256</td>
<td>1627</td>
</tr>
<tr>
<td>SubFP (rbf)</td>
<td>1878</td>
<td>1887</td>
<td>1875</td>
<td>1881</td>
<td>1330</td>
<td>1344</td>
</tr>
<tr>
<td>ExtFP (poly)</td>
<td>2163</td>
<td>2340</td>
<td>2162</td>
<td>2340</td>
<td>2220</td>
<td>2271</td>
</tr>
<tr>
<td>KlekFP (poly)</td>
<td>1928</td>
<td>2292</td>
<td>1933</td>
<td>2292</td>
<td>930</td>
<td>2002</td>
</tr>
<tr>
<td>PubchemFP (poly)</td>
<td>1635</td>
<td>2299</td>
<td>1627</td>
<td>2297</td>
<td>1079</td>
<td>1997</td>
</tr>
<tr>
<td>MACCSFP (poly)</td>
<td>1704</td>
<td>1896</td>
<td>1709</td>
<td>1899</td>
<td>734</td>
<td>1147</td>
</tr>
<tr>
<td>SubFP (poly)</td>
<td>1637</td>
<td>1664</td>
<td>1631</td>
<td>1659</td>
<td>997</td>
<td>1043</td>
</tr>
</tbody>
</table>

Apart from classification scores and training time, one important aspect of SVM model is the resulting model’s complexity. In the kernelized formulation, the good measure of such quantity is the number of support vectors (in the linear case you can always reduce the model to the $d$ numbers representing the hyperplane in the input space). Table 6 summarizes this analysis in performed experiments. We report mean and median amount of support vectors in the whole experiment (for given fingerprint and kernel) as well as for the best model. It is worth noting, that not only our method produced better classification results, but also resulting machines are slightly simpler (up to 10% less support vectors – for example when using
polynomial kernel on the MACCSFP fingerprint). This is a consequence of described in the Section 2 change of the data geometry in the feature space. Proposed method shatters data in such a way, that points are more distant from the separating hyperplane, resulting in the smaller number of required support vectors.

7. Conclusions

In this paper we proposed the data pre- and postprocessing method for binary classification, which is similar to the classical Mahalanobis based transformation but allows to exploit differences between distinct classes. Fitting our method into the standard SVM problem optimization formulation allows us to use it in the Black Box SVM scenario, which makes it widely applicable with vast amount of existing libraries. We give strong theoretical foundations for such method based on the optimal solution given by Maxi-Min Margin Machine. Together with provided examples and empirical tests we show that this approach well corresponds to the underlying geometrical data structure and as a result leads to easier and better adapted representation for the SVM. From the practical point of view this is a simple modification which leads to no additional overhead in terms of memory, time or code complexity. It can be directly incorporated into most of the existing libraries, as long as they provide user with ability to preprocess data before model training and testing or to define kernel function.

The most important consequence of our experimental evaluation shows that our method speeds up the learning process. This advantage comes with no cost in terms of generalization abilities, in fact $2eSVM$ has a bigger chance to score better on typical data. Consequently our method is a valuable alternative for currently widely used covariance based preprocessing techniques in case of binary classification. Presented approach is mainly suited for the largest margin linear models (like SVM) but could also be used in more complex applications (like presented kernelized SVMs).

Due to its generality, presented work has many possible research directions.

1. In complex models, like those using ensembles [28] or hierarchical models [29], training time is crucial. It would be valuable to test, whether $2eSVM$ like preprocessing scheme can improve the maximum number of models used in such complex classifier (without sacrificing training time) and as a result increase the overall system’s accuracy.

2. Many actual applications deal with multi-class problem rather than binary classification [30], it would be valuable to investigate the best way to generalize proposed method to multi-class scenario.
3. Even though proposed method is based on a classical Support Vector Machine it would be valuable to test whether other classification models, especially similar in the geometrical sense [31, 32], could benefit from such data transformation.

4. As our method is based on the ellipsoids fitted to the data, one possible development area is a search for their better construction. Instead of covariance, one could use ellipsoids built in the optimization process like those used in Ellipsoid Kernel Machines [33].

5. 2eSVM is based on the remark that in many applications classes are very diverse in terms of internal geometries. Besides analysed in this paper – cheminformatical data – in our opinion, a natural candidates for such phenomena are problems in Natural Language Processing [13, 14, 15], as the internal variety of a vocabulary used in different context tends to be very diversified (similar argument holds for credit scoring [16]).

8. Acknowledgments

We would like to thank Sabina Smusz from the Institute of Pharmacology, Polish Academy of Sciences for providing us with cheminformatical data and the insights into the role of fingerprints in the ligands classification.


[22] V. V. Williams, Breaking the coppersmith-winograd barrier, unpublished manuscript (2011).


URL http://archive.ics.uci.edu/ml


