Recursive SVM Based on TEDA

Dmitry Kangin[1(✉)] and Plamen Angelov[1,2]

[1] Data Science Group, School of Computing and Communications,
Lancaster University, Lancaster, UK
[2] Chair of Excellence, University Carlos III, Madrid, Spain
{d.kangin,p.angelov}@lancaster.ac.uk

Abstract. The new method for incremental learning of SVM model incorporating recently proposed TEDA approach is proposed. The method updates the widely renowned incremental SVM approach, as well as introduces new TEDA and RDE kernels which are learnable and capable of adaptation to data. The slack variables are also adaptive and depend on each point's 'importance' combining the outliers detection with SVM slack variables to deal with misclassifications. Some suggestions on the evolving systems based on SVM are also provided. The examples of image recognition are provided to give a 'proof of concept' for the method.

Keywords: SVM · TEDA · Incremental learning · Evolving system

1 Introduction

Nowadays, there are plenty of models for object classification. Some of them are aimed on off-line classification, which takes all the samples at once, other are aimed for the incremental classification, which work sample-by-sample. Those which do not require to hold all the sample set, are referred as 'online'. Finally, 'evolving' models are designed to change the structure of the system taking into account recent changes and forget those patterns which occurred long ago.

The problem stated in the article is raised by different approaches: SVM models, novel TEDA data analysis concept, and evolving systems.

SVM was first proposed as Generalised Portrait Algorithm by Vapnik and Lerner in 1963 [1], with some works together with Chervonenkis [2], but was not widely used until the beginning of the 1990-s, when new contributions by Vapnik and other authors were proposed [3], [4]. Now, the SVM family contains a huge number of different algorithms, capable of most widely known machine learning problems (clustering[5], outlier detection[6], structured learning[7]).

TEDA approach has recently emerged and gives promising results on data outlier detection[8], classification [9] and other machine learning problems [10]. It is based on data and provides attractive concept of learning the parameters by data, not by some pre-defined constraints.

Evolving systems describe the ability of structural adaptation "from scratch" to consider the changes in the data stream[11], [12]. The distinctive feature of such a system is that it takes into consideration more recent patterns in data, forgetting those

A. Gammerman et al. (Eds.): SLDS 2015, LNAI 9047, pp. 156–168, 2015.
DOI: 10.1007/978-3-319-17091-6_11

that happened many time ago and hence may be not relevant, by changing the structure of the classifier and forgetting the patterns which are not relevant. One of the popular applications of such systems is classification [13], [14], [15]. Such methods are based on fuzzy systems, as well as neural networks [16].

The outline of the article is as follows:

- first, the SVM systems are reviewed;
- second, the TEDA background and TEDA SVM statement is given;
- third, the novel TEDA kernel is proposed;
- fourth, incremental update procedure is formulated;
- fifth, the new samples incremental update procedure is proposed;
- sixth, the experimental results are discussed.

The article is finalised by conclusion, describing the research results.

2 SVM Model Formulation

Here the accurate statement of the supervised learning problem is performed. Let us have some object set, which we designate as Ω, and a finite space of object classes, referred as Υ. Then, let us have a subset, $\Omega_L \subset \Omega$, named training set, for each the following function is defined: $F_L: \Omega_L \to \Upsilon$. The problem is to build a function $F: \Omega \to \Upsilon$, approximating the function F_L on the set Ω_L. The assumption is that F will be a good mapping for further objects, Ω_V where the index V denotes validation. $\Omega_V \bigcap \Omega_L = \emptyset$; $\Omega_V, \Omega_L \subset \Omega$. The method we propose has its roots in the SVM problem statement [4, 17]. Here we introduce the basic formulation of SVM problem with slack variables and notation we further use. Consider a two class classification problem, where

$$\Omega_L = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_k\}, \Upsilon = \{-1, 1\}, F(\boldsymbol{x}_n) = t_n, n \in [1 \dots k]. \tag{1}$$

Here k is the number of the data samples.

Here we state the problem for overlapping classes (C-SVM) [17]

$$\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{k} \Xi_n \to \min_{\boldsymbol{w}, b, \Xi_n}, \tag{2}$$

w.r.t. following constraints:

$$t_n y(\boldsymbol{x}_n) \geq 1 - \Xi_n, \Xi_n \geq 0, n = 1 \dots k. \tag{3}$$

Here Ξ_n are so-called 'slack variables', C is the so-called 'box constraint' parameter [17]. This problem statement allows some misclassification of the training set for hardly-divisible data, and $y(x) = \boldsymbol{w}^T \boldsymbol{x} + b$, where $\boldsymbol{w}$, b are the parameters of the hyperplane. This formulation is widely used [4, 17,18]. If we write down Lagrangian, differentiate it and find an extremum [17], we get the dual problem

$$\check{L}(\alpha) = -\frac{1}{2}\sum_{n=1}^{k}\sum_{m=1}^{k}\alpha_n\alpha_m t_n t_m[\phi(\boldsymbol{x}_n)]^T\phi(\boldsymbol{x}_m) + \sum_{n=1}^{k}\alpha_n \to \min_{\alpha,b}, \quad 0 \le \alpha_n \le C, \sum_{n=1}^{k}\alpha_n t_n = 0. \tag{4}$$

Then, we can consider another problem (ν-SVM) [18]:

$$\frac{1}{2}\|\boldsymbol{w}\|^2 - \nu\gamma + \frac{1}{k}\sum_{n=1}^{k}\Xi_n \to \min_{\boldsymbol{w},b,\gamma,\Xi_n} \tag{5}$$

w.r.t. the following constraints:

$$t_n y(\boldsymbol{x}_n) \ge \gamma - \Xi_n, \Xi_n \ge 0, \qquad n = 1, \dots, k, \tag{6}$$

The parameter ν gives a lower bound for the fraction of support vectors, as well as an upper bound for the fraction of margin errors in case of $\gamma > 0$. This formulation can be proven to be equivalent to C-SVM with $\hat{C} = \frac{1}{k\gamma}$, if $\gamma > 0$ [18].

Then we can denote a kernel function $K(\boldsymbol{x}_n, \boldsymbol{x}_m) = [\phi(\boldsymbol{x}_n)]^T\phi(\boldsymbol{x}_m)$, where $\phi(\cdot)$ is some mapping function, taken from the (possibly infinite) mapping functions space. It can be recognised as a replacement to the ordinary scalar product, introducing non-linearity into the model. The widely known kernels are: linear, Mahalanobis [19], cosine [20], Gaussian [21], histogram intersection [22] or survival [23] kernels). One can also restore it based on the data using metric learning techniques. Hence, the feature transformation (for finite or infinite space) can be replaced by changing the distance metric, given by the kernel.

3 TEDA Approach Summary

TEDA framework provides a novel systematic method of a "per point" online data analysis [8], [9].

Consider that we have object space $\Omega \subseteq \mathbb{R}^p$ containing data samples, where p is the data dimensionality. This space is equipped with some distance (i.e. Euclidean, L_1, cosine or any others). Further we refer this distance as $d(\boldsymbol{x}, \boldsymbol{y})$. We can pick data samples sequence from this object space:

$$\{\boldsymbol{x}_1, \boldsymbol{x}_2 \dots \boldsymbol{x}_k \dots\}, \boldsymbol{x}_k \in \Omega, k \in \mathbb{N}, \tag{7}$$

where k is a sequence number of the object which can be represented as the instant of time, when the data sample has arrived. For this reason, index k will be referred to as time instant further for simplicity. We can construct sum distance to some particular point $\boldsymbol{x} \in \Omega$, for each element up to the k-th one [8], [10]:

$$\pi^k(\boldsymbol{x}) = \sum_{i=1}^{k} d(\boldsymbol{x}, \boldsymbol{x}_i), k \ge 1. \tag{8}$$

Based on this definition, we define the eccentricity at the time instant k:

$$\xi^k(\boldsymbol{x}) = \frac{2\pi^k(\boldsymbol{x})}{\sum_{i=1}^{k}\pi^k(\boldsymbol{x}_i)} = 2\frac{\sum_{i=1}^{k} d(\boldsymbol{x},\boldsymbol{x}_j)}{\sum_{i=1}^{k}\sum_{j=1}^{k} d(\boldsymbol{x}_i,\boldsymbol{x}_j)}, k \ge 2, \sum_{i=1}^{k}\pi^k(\boldsymbol{x}) > 0. \tag{9}$$

The complement of eccentricity, typicality is defined as:

$$\tau^k(\boldsymbol{x}) = 1 - \xi^k(\boldsymbol{x}). \tag{10}$$

The eccentricity and typicality are both bounded [8], [10]:

$$0 \leq \xi^k(\boldsymbol{x}) \leq 1, \sum_{i=1}^k \xi^k(\boldsymbol{x}_i) = 2, k \geq 2, \sum_{i=1}^k \pi^k(\boldsymbol{x}_i) > 0. \tag{11}$$

$$0 \leq \tau^k(\boldsymbol{x}) \leq 1, \sum_{i=1}^k \tau^k(\boldsymbol{x}_i) = k - 2, k \geq 2, \sum_{i=1}^k \pi^k(\boldsymbol{x}_i) > 0. \tag{12}$$

Normalised eccentricity and typicality can also be defined as [8], [10]:

$$\zeta^k(\boldsymbol{x}) = \frac{\xi^k(x)}{2}, \sum_{i=1}^k \zeta^k(\boldsymbol{x}_i) = 1, k \geq 2, \sum_{i=1}^k \pi^k(\boldsymbol{x}_i) > 0. \tag{13}$$

$$t^k(\boldsymbol{x}) = \frac{\tau^k(x)}{k-2}, \sum_{i=1}^k \tau^k(\boldsymbol{x}_i) = 1, k > 2, \sum_{i=1}^k \pi^k(\boldsymbol{x}_i) > 0. \tag{14}$$

The method's capability of online problems resolution follows from existence of the formulae of the incremental update [8], [9]. There exist convenient formulae for incremental calculation for Euclidean and Mahalanobis distance [19], but here we do not discuss it. Generally, for any distance, we can use the formula $\pi^{k+1}(\boldsymbol{x}) = \pi^k(\boldsymbol{x}) + \mathrm{d}(\boldsymbol{x}, \boldsymbol{x}_{k+1})$ and calculate all other quantities based on it.

4 The TEDA SVM Statement

In the classical formulation of SVM given above we aim to give an upper bound of the fraction of margin errors. But here we are targeting aiming to make it dependent on the 'importance' of each of the support vectors, i.e. make more '*typical*' support vectors penalised more for being out of boundaries, rather than '*anomalous*'. In other words, we care less for anomalous data samples and more for typical ones to be covered well by the classifier; we do not try to cover all data sample equally, but proportionally to their typicality.

Here, the "local" typicality in regards to each of the classes is proposed as a weight for each of the support vectors. For each class c_m we define typicality [8]

$$\tau_c^k(\boldsymbol{x}) = 1 - 2\frac{\sum_{x_j \in X_c} d(\boldsymbol{x}, \boldsymbol{x}_j)}{\sum_{x_i \in X_c} \sum_{x_j \in X_c} d(\boldsymbol{x}_i, \boldsymbol{x}_j)}, c \in \{c_m, c_{\bar{m}}\}, c_m \in \mathrm{C}, \mathrm{c}_{\bar{m}} = \mathrm{C} \backslash \mathrm{c}_m. \tag{15}$$

Here X_c denotes the objects with labels from the set c, which we build here by the scheme 'one versus the rest'. For each of the classes we build the following model:

$$\frac{1}{2}\|\boldsymbol{w}\|^2 + \sum_{n=1}^k C\tau_{c_m}^k(\boldsymbol{x_n})[y_n > 0]\Xi_n + \sum_{n=1}^k C\tau_{c_{\bar{m}}}^k(\boldsymbol{x_n})[y_n < 0]\Xi_n \to \min_{\boldsymbol{w}, \Xi, \mathrm{b}} \tag{16}$$

w.r.t. following constraints:

$$t_n y(\boldsymbol{x}_n) \geq 1 - \Xi_n, \Xi_n \geq 0, n = 1 \ldots k. \tag{17}$$

We can see that this model preserves the upper boundary property γ for the margin errors [1817], where $\gamma = \frac{1}{k\hat{C}}$, $C > 0$, as the upper boundary is known: $C\tau^k_{C_{\{m,\overline{m}\}}}(\boldsymbol{x}) \leq C$. But what we gain additionally is possibility to control the 'importance' of each data samples. The algorithm takes into account our desire to 'sacrifice' the correct recognition only for the most 'anomalous' data. The more typical is data in the model, the higher will be the box constraint for it because of higher typicality. Opposite case, when the object is more anomalous, is penalised less due to lower typicality in order to avoid fitting the model to anomalous cases.

5 TEDA Kernel

Further to TEDA formulation, we propose also novel formulation of a kernel within the TEDA framework. We define the following TEDA-like kernel:

$$\ddot{\zeta}^k(\boldsymbol{x},\boldsymbol{y}) = \langle \boldsymbol{x},\boldsymbol{y}\rangle\left(\zeta^k(\boldsymbol{x})\zeta^k(\boldsymbol{y})\right)^\gamma,$$
$$\ddot{\zeta}^k(\boldsymbol{x},\boldsymbol{y}) = \langle \boldsymbol{x},\boldsymbol{y}\rangle\left(\frac{\sum_{i=1}^k d(\boldsymbol{x}_i,\boldsymbol{x})\sum_{i=1}^k d(\boldsymbol{x}_i,\boldsymbol{y})}{\left(\sum_{i=1}^k\sum_{j=1}^k d(\boldsymbol{x}_i,\boldsymbol{x}_j)\right)^2}\right)^\gamma, \sum_{i=1}^k\sum_{j=1}^k d(\boldsymbol{x}_i,\boldsymbol{x}_j) > 0. \quad (18)$$

Here $\zeta^k(\boldsymbol{x})$ is a normalised data eccentricity, $\gamma > 0$ is some parbameter, showing the data eccentricity involvement. It helps us to take into account not only scalar product itself, but also the importance of each of the points. The interpretation of the kernel is to increase the kernel values for the '*anomalous'* points, and at the same time decrease the kernel values between *'typical'* points, bringing it closer in the data space.

To be the kernel, $\ddot{\zeta}^k$ should meet the following requirements:

1. $\ddot{\zeta}^k(\boldsymbol{x},\boldsymbol{y}) = \ddot{\zeta}^k(\boldsymbol{y},\boldsymbol{x})$.
2. $\ddot{\zeta}^k$ is positive semi-definite, i.e. $\forall \boldsymbol{y}_1,\boldsymbol{y}_2,\ldots\boldsymbol{y}_m \in \Omega, \forall \boldsymbol{x}_1,\boldsymbol{x}_2,\ldots\boldsymbol{x}_k \in \Omega,$ where Ω is a Hilbert space, the matrix $M \in \mathbb{R}^{m\times m}$, $M_{ij} = \ddot{\zeta}^k(\boldsymbol{y}_i,\boldsymbol{y}_j)$ is non-negative definite, that is for any $\alpha \in \mathbb{R}^m$ $\alpha^T M\alpha \geq 0$.

Proof sketch:

1. $\ddot{\zeta}^k(\boldsymbol{x},\boldsymbol{y}) = \langle \boldsymbol{x},\boldsymbol{y}\rangle\left(\frac{\sum_{i=1}^k d(\boldsymbol{x}_i,\boldsymbol{x})\sum_{i=1}^k d(\boldsymbol{x}_i,\boldsymbol{y})}{\left(\sum_{i=1}^k\sum_{j=1}^k d(\boldsymbol{x}_i,\boldsymbol{x}_j)\right)^2}\right)^\gamma = \langle \boldsymbol{y},\boldsymbol{x}\rangle\left(\frac{\sum_{i=1}^k d(\boldsymbol{x}_i,\boldsymbol{y})\sum_{i=1}^k d(\boldsymbol{x}_i,\boldsymbol{x})}{\left(\sum_{i=1}^k\sum_{j=1}^k d(\boldsymbol{x}_i,\boldsymbol{x}_j)\right)^2}\right)^\gamma =$
$= \ddot{\zeta}(\boldsymbol{y},\boldsymbol{x}).$
2. For Euclidean distance, it can be proven to be equivalent to:

$$\ddot{\zeta}^k(\boldsymbol{x},\boldsymbol{y}) \propto \langle \boldsymbol{x},\boldsymbol{y}\rangle\left(\left(\|\boldsymbol{x}-\boldsymbol{\mu}^k\|^2 + {\sigma^k}^2\right)\left(\|\boldsymbol{y}-\boldsymbol{\mu}^k\|^2 + {\sigma^k}^2\right)\right)^\gamma, \quad (19)$$

where $\boldsymbol{\mu}^k$ is the mean over $\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots\boldsymbol{x}_k$, σ^k is a variance over $\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots\boldsymbol{x}_k$ which depend on the data set and are the values derived from data. It can be proven to be a particular case of the polynomial kernel multiplied on the linear kernel.

The scalar product can be replaced here by any kernel $K(\boldsymbol{x},\boldsymbol{y})$, to get this expression:

$$\ddot{\zeta}^k(\boldsymbol{x},\boldsymbol{y}) \propto K(\boldsymbol{x},\boldsymbol{y})\left(\left(\|\boldsymbol{x}-\boldsymbol{\mu}^k\|^2+\sigma^{k^2}\right)\left(\|\boldsymbol{y}-\boldsymbol{\mu}^k\|^2+\sigma^{k^2}\right)\right)^T, \tag{20}$$

We can also make the similar statement via RDE [12]:

$$D(\boldsymbol{x},\boldsymbol{y}) = 1/(1+\|\boldsymbol{x}-\boldsymbol{y}\|^2 + \Sigma_k - \|\boldsymbol{\mu}_k^2\|)\,. \tag{21}$$

It is equivalent to Cauchy kernel [12] (up to the multiplier), and Cauchy kernel itself can be proven to be a proper kernel, hence it is also a kernel:

$$D(\boldsymbol{x},\boldsymbol{y}) \propto \frac{1}{1+\|x-y\|^2/\alpha}, \alpha > 0. \tag{22}$$

6 TEDA SVM Incremental Update

Here the standard problem of the incremental update is stated.

Let us have a training data sequence $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_k \dots\}$, $\boldsymbol{x}_k \in \Omega$ which arrives one-by one, where k is an order number of the data element. For each $\boldsymbol{x}_k$ there is a label $y_k \in \Upsilon$. Up to the k-th element we have the problem (16).

Then, the $(k+1)$-th element arrives, and the problem is re-formulated as

$$\frac{1}{2}\|\boldsymbol{w}\|^2 + \sum_{i=1}^{k+1} C\tau^i_{c_m}(\boldsymbol{x_i})[y_i > 0]\Xi_n + \sum_{i=1}^{k+1} C\tau^i_{c_{\overline{m}}}(\boldsymbol{x_i})[y_i < 0]\Xi_i \to \min_{\mathbf{w},\Xi,b} \tag{23}$$

w.r.t. following constraints:

$$t_i y(\boldsymbol{x}_i) \geq 1 - \Xi_i, \Xi_i \geq 0, i = 1 \dots k+1. \tag{24}$$

Also, we should take into account that

$$y(\boldsymbol{x}) = \boldsymbol{w}^T\phi(\boldsymbol{x}) + b, \tag{25}$$

as it was mentioned before, and $\phi(\boldsymbol{x})$ is a feature mapping. Hence, we should also update the feature mapping $\phi(\boldsymbol{x})$. Generally, it is not feasible as the mapping can be set on functional spaces including infinite-dimensional ones. It can be shown, that dual problem transition allows us to express $y(\boldsymbol{x})$ in terms of the kernel $K(\boldsymbol{x}_i,\boldsymbol{x}_j) = \phi^T(\boldsymbol{x}_i)\phi(\boldsymbol{x}_j)$ (this notation denotes scalar product, although the mapping can be given in infinite-dimensional space). Hence we need to update the kernel matrix $K \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}}$ only, where $K_{ij} = K(\boldsymbol{x}_i,\boldsymbol{x}_j)$.

Then we can see, that the incremental update consists of several stages:

- kernel update (section 6.2);
- solution update for the updated kernel (section 6.2);
- solution update for the updated box constraints (section 6.3);
- solution update incorporating the new data (section 6.1).

Then we can write down the dual problem as

$$\check{L}(\alpha) = \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k}\alpha_i\alpha_j t_i t_j [\phi(\boldsymbol{x}_i)]^T \phi(\boldsymbol{x}_j) - \sum_{i=1}^{k}\alpha_i + b\sum_{n=1}^{k} t_i\alpha_i =$$
$$= \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k}\alpha_i\alpha_j t_i t_j K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) - \sum_{i=1}^{k}\alpha_i + b\sum_{i=1}^{k} t_i\alpha_i \to \min_{\alpha,b}\ , \tag{26}$$
$$0 \le \alpha_i \le C_i, \forall i \in [1,k], \sum_{i=1}^{k}\alpha_i t_i = 0.$$

Differentiating the dual problem, we obtain Karush-Kuhn-Tucker conditions (KKT):

$$g_j(x_j) = \frac{\partial \check{L}(\alpha)}{\partial \alpha_j} = \sum_{i=1}^{k}\alpha_i t_i t_j K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) - 1 + t_j b = t_j y(\boldsymbol{x}_j) - 1,$$
$$\frac{\partial \check{L}(\alpha)}{\partial b} = \sum_{i=1}^{k} t_i\alpha_i = 0, g_j(x_j)\begin{cases} > 0, & \alpha_j = 0, \\ = 0, & 0 < \alpha_j < C_j. \\ < 0, & \alpha_j = C_j. \end{cases} \tag{27}$$

Let us denote all the training set as Ω_L. This set is divided onto three disjoined sets: margin vectors S, for which $g_j(\boldsymbol{x}_j) = 0$, error support vectors E, for which $g_j(\boldsymbol{x}_j) < 0$, and the rest of the vectors vectors R, for which $g_j(\boldsymbol{x}_j) > 0$, which are not included into the existing solution.

6.1 Adding new Samples

The incremental learning method for SVM was first described in [24].

Apart of this previously proposed method, here we do not choose one global box constraint C for every element of the SVM problem, but use its own box constraint for every object. More, we do not even make it constant, but update it from data.

When transferring from the problem (26) for k elements to the problem for $k+1$ elements, we should ensure, that the data is in equilibrium, i.e. the conditions of the problem are satisfied.

Let us denote for notation simplicity $M = k + 1$.

We denote as $Q_{ij} = t_i t_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

Then, we begin to change the new vector's coefficient α_M until the configuration within the system changes:

$$\Delta g_j(\boldsymbol{x}_j) = Q_{jM}\Delta\alpha_M + \sum_{n\in S} Q_{jn}\Delta\alpha_M + t_j\Delta b, \forall\, j \in \Omega_L \cup \{M\},$$
$$0 = t_M\Delta\alpha_M + \sum_{n\in S} t_n\Delta\alpha_n. \tag{28}$$

Then, we can define

$$\Theta = \begin{bmatrix} 0 & t_{s_1} & \cdots & t_{s_{l(S)}} \\ t_{s_1} & Q_{s_1 s_1} & \cdots & Q_{s_1 s_{l(S)}} \\ \vdots & \vdots & \ddots & \vdots \\ t_{s_{l(S)}} & Q_{s_{l(S)} s_1} & \cdots & Q_{s_{l(S)} s_{l(S)}} \end{bmatrix} \tag{29}$$

and write the KT conditions changing equations in the vector form as

$$\Theta\begin{bmatrix}\Delta b & \Delta\alpha_{s_1} & \dots & \Delta\alpha_{s_{l(S)}}\end{bmatrix}^T = -\begin{bmatrix}y_M & Q_{s_1M} & \dots & Q_{s_{l(S)}M}\end{bmatrix}^{\mathrm{T}}\Delta\alpha_M \tag{30}$$

for the support vector set.
Then we can continue with

$$\Delta b = \beta\Delta\alpha_M, \tag{31}$$

$$\Delta\alpha_j = \beta_j\Delta\alpha_M, \forall\, j \in D. \tag{32}$$

$$\begin{bmatrix}\beta & \beta_{s_1} & \dots & \beta_{s_{l(s)}}\end{bmatrix}^T = -\Theta^{-1}\begin{bmatrix}y_M & Q_{s_1M} & \dots & Q_{s_{l(s)}M}\end{bmatrix}^T \tag{33}$$

for all support vectors, and $\beta_n = 0\ \forall\, n \in T\backslash S$.
Then

$$\Delta g_j(x_j) = \Gamma_j\Delta\alpha_M, \forall\, j \in T\cup\{M\}; \Gamma_j = Q_{jM} + \textstyle\sum_{n\in S} Q_{jn}\beta_n + t_j\beta, \forall j \notin S. \tag{34}$$

After that, we find the maximal increment until the following event occur:

- $g_M \le 0$, with M joining S when $g_M = 0$;
- $\alpha_M \le 0$,with M joining E when $\alpha_M = 0$;
- $0 \le \alpha_j \le C_j, j \in S$ with $\alpha_j = 0$ when j-th vector transfers from S to R, and $\alpha_j = C_j$ when transferring from S to E;
- $g_j \le 0, \forall\, j \in E$, with $g_j = 0$ when j-th vector transfers from E to S;
- $g_j \ge 0,\ \forall j \in R,$ with $g_j = 0$ when j-th vector transfers from R to S.

After this procedure, the new support vectors should be added to the matrix Θ^{-1}.
It can be proven that

$$\Theta^{-1} \leftarrow \begin{bmatrix}\Theta^{-1} & 0\\ \vdots & \\ 0 \ \dots & 0\end{bmatrix} + \\ +\frac{1}{\Gamma_M}\begin{bmatrix}\beta & \beta_{s_1} & \dots & \beta_{s_{l(s)}} & 1\end{bmatrix}^T\begin{bmatrix}\beta & \beta_{s_1} & \dots & \beta_{s_{l(s)}} & 1\end{bmatrix}. \tag{35}$$

This formula allows us to add a vector into S.

The procedure is repeated until no transition between subsets R, E, and S occurs. Also, it should be noticed, that the procedure is proven to be reversible [24]. The process of deletion data during the learning process is referred as decremental learning [24].

6.2 Updating the Kernel

Here we address the learnable kernel update problem in SVM:

$$\check{L}_k(\alpha) = \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k}\alpha_i\alpha_j t_i t_j K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) - \sum_{i=1}^{k}\alpha_i + b\sum_{i=1}^{k} t_i\alpha_i \to \min_{\alpha,b}. \tag{36}$$

The problem is replaced by one with the new kernel $\widehat{K}$

$$\check{L}_{k+1}(\alpha) = \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k}\alpha_i\alpha_j t_i t_j \widehat{K}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) - \sum_{i=1}^{k}\alpha_i + b\sum_{i=1}^{k} t_i\alpha_i \to \min_{\alpha,b} \tag{37}$$

Again, the problems are constrained as in (26).

Therefore, the problem differs in kernel we use. As before, we denote

$$Q_{ij} = t_i t_j K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right), \hat{Q}_{ij} = t_i t_j \widehat{K}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right). \tag{38}$$

Let us denote also

$$\hat{Q}_{ij} - Q_{ij} = \Delta Q_{ij}. \tag{39}$$

Then we consider

$$\Delta g_j\left(x_j\right) = \sum_{n\in S}(Q_{jn} + \beta\Delta Q_{jn})\Delta\alpha_n + \beta\sum_{n\in S}\Delta Q_{jn}\alpha_n + t_j\Delta b, \sum_{n\in S} t_n\Delta\alpha_n = 0. \tag{40}$$

Here $0 \le \beta \le 1$. The problem is to express the corrections of $\Delta\alpha_n$ as a function of some coefficient $\beta \in [0,1]$.

Here we denote Θ as in (29) and

$$\breve{\Theta} = \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & \Delta Q_{s_1 s_1} & \dots & \Delta Q_{s_1 s_{l_S}} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \Delta Q_{s_{l_S} s_1} & \dots & \Delta Q_{s_{l_S} s_{l_S}} \end{bmatrix}. \tag{41}$$

Then

$$\left(\Theta + \beta\breve{\Theta}\right)\begin{bmatrix}\Delta b & \Delta\alpha_{s_1} & \dots & \Delta\alpha_{s_{l_S}}\end{bmatrix}^T = \beta\breve{\Theta}\begin{bmatrix} b & \alpha_{s_1} & \dots & \alpha_{s_{l_S}}\end{bmatrix}^T. \tag{42}$$

In this case

$$\begin{aligned}\begin{bmatrix}\Delta b & \Delta\alpha_{s_1} & \dots & \Delta\alpha_{s_{l_S}}\end{bmatrix}^T &= \beta\left(\Theta + \beta\breve{\Theta}\right)^{-1}\breve{\Theta}\begin{bmatrix} b & \alpha_{s_1} & \dots & \alpha_{s_{l_S}}\end{bmatrix}^T = \\ &= \begin{bmatrix} b & \alpha_{s_1} & \dots & \alpha_{s_{l_S}}\end{bmatrix}^T - \left(\Theta + \beta\breve{\Theta}\right)^{-1}\Theta\begin{bmatrix} b & \alpha_{s_1} & \dots & \alpha_{s_{l_S}}\end{bmatrix}^T.\end{aligned} \tag{43}$$

The maximal increment condition is the same as for the new data update, but additionally we should mind $\beta \in [0,1]$. The update of the matrix is performed the same way like Θ.

6.3 Updating Box Constraints

In this case, we use the same update equations, as the standard update for the new data, but here we check if the KKT constraints are violated on the first stage due to the change of the box constraints.

We remember, that for each of the objects we have (27) with the old constraints. For the stage $k+1$ we should have

$$g_j(\boldsymbol{x}_j)\begin{cases} > 0, & \alpha_j = 0, \\ = 0, & 0 < \alpha_j < C_j^{k+1}, \\ < 0, & \alpha_j = C_j^{k+1}. \end{cases} \tag{44}$$

but for some of the vectors $D \subset \Omega_L$ the conditions can be broken: $\alpha_j > C_j^{k+1}$. Also, because generally $C_j^{k+1} \neq C_j^k$, some transitions between sets E and R may occur. Hence, we need to correct the solution w.r.t. all the vectors violating the conditions.

The procedure is proposed in a following way. For each vector $\boldsymbol{x}_m$, violating the KKT conditions, we perform the procedure exactly the same way as for the new vector, considering that it is a newly-added vector to k-vector SVM problem with training set $D \backslash \boldsymbol{x}_m$ and new C_i. For the sets E and S, the vector should be preliminarily deleted by decremental learning procedure should be as [24].

7 Suggestions on the Method Implementation

Additionally, even despite the given method is not '*evolving*' but *'online'*, as it does not discard the previous data samples, the decremental learning procedure can be proposed, as the incremental procedure for SVM is reversible. The following simplest method can be considered:

- for each sample $\boldsymbol{x}$, update its within-class typicality $\tau_c^k(\boldsymbol{x})$;
- inspect all the samples which appeared in the train sequence up to the newest element $\boldsymbol{x}_k$: $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \ldots \boldsymbol{x}_k\}$, $\boldsymbol{x}_n \in \Omega$. If their typicality is lower than some given threshold T, we can state, that the sample should be rejected.
- For the sample set elements to be rejected, perform decremental learning, analogous to that described in [24].

Using this modification, we can employ all the benefits of evolving systems, such as possibility to adopt to the changes on the data neglecting those patterns that appeared long ago, but base it on the SVM framework, rather than fuzzy systems or neural networks. Usage of the SVM framework, instead of fuzzy rule-based evolving systems, gives us an opportunity of using strict optimisation problem statements, which gives benefits of better generalisation of the method.

8 Demonstrations for the Experimental Data

For the proof of concept, an example of human activities images classification was provided (**Fig. 1**, data set [25]). The feature transformation method was exactly borrowed from the previous research described in [13] and is composed from Haar and gist [26] features.

Fig. 1. Human activities data samples

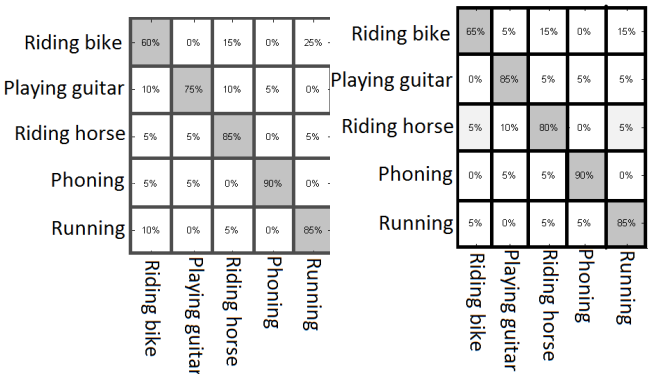

Fig. 2. Results of the recognition for different methods (left picture is SVM with histogram intersection kernel, accuracy rate is 79%, right picture is SVM with TEDA kernel, combined with histogram intersection kernel, and TEDA box constraints, accuracy rate is 81%).

For the experiments, the following methods were compared:

— SVM with Gaussian kernel with $\sigma = 34.4725, \quad C = 30$.
— SVM with TEDA kernel, combined with Gaussian kernel, with $\sigma = 34.4725, \gamma = 2, C = 30$, and TEDA weights.

Here the Gaussian kernel is

$$K_G(x, y) = \exp(-(\boldsymbol{x} - \boldsymbol{y})^T(\boldsymbol{x} - \boldsymbol{y})/(2\sigma^2)). \tag{45}$$

In contrast to Gaussian distribution probability density, the multiplier is neglected, just because the solution will be the same for any constant.

By TEDA kernel combined with Gaussian kernel we denote

$$K_{\text{TEDA}}(x, y) = K_G(x, y) \times \left(\left(\|\boldsymbol{x} - \boldsymbol{\mu}^k\|^2 + \sigma^{k^2}\right)\left(\|\boldsymbol{y} - \boldsymbol{\mu}^k\|^2 + \sigma^{k^2}\right)\right)^{\gamma}. \tag{46}$$

No other modifications are introduced into the model except the TEDA kernel and TEDA weights. Therefore, we suppose that the improvement in the kernel and box variables leads to the increase of the quality of the recognition.

The training data set contains 200 items, 40 images for each of the classes. The testing data set consists of 100 items, 20 images per class. Although the method was initially designed for handwritten symbol images, it can be also applied to describe any other images as well.

The results are given as it is depicted in **Fig. 2**. On the left part of the image, the results for Gaussian kernel are proposed. At the right part of the image, SVM was augmented with TEDA 'box constraints' and uses TEDA kernel, combined with Gaussian kernel. One can see, that the results were (slightly) improved. However, the method has more perspectives, as it makes the model more flexible giving different values to the 'box constraints' and enabling kernel learning 'from scratch'.

9 Conclusion

In this paper, the new modification of the SVM with slack variables for classification was proposed which changes slack variables independently for each of the data samples taking into account the 'typicality' of each data sample and forcing more 'anomalous' data to be misclassified sooner than typical ones. For this purpose, the novel 'box constraints' update model was developed based on the recently proposed TEDA framework, and the novel learnable kernels based on TEDA and RDE were proposed. The incremental SVM was modified to take into account changes of the box variables during the training process, as well as learnable kernels. Though the model, proposed here, is not evolving yet, the ideas were described how to make the model 'evolving'. In the experimental section, a few examples were given to approve the ideas of SVM with new learnable box constraints.

Acknowledgements. Plamen Angelov would like to acknowledge the support of Chair of Excellence award which he was given by Santander Bank, Spain for the period March-September 2015 which is being hosted by Carlos III University, Madrid, Spain.

References

1. Vapnik, V., Lerner, A.: Pattern recognition using generalised portrait method. Automation and Remote Control **24**, 774–780 (1963)
2. Vapnik, V.N., Ya, A., Chervonenkis.: Teorija raspoznavanija obrazov: Statisticheskie problemy obuchenija. (Russian) = Theory of pattern recognition: Statistical problems of learning. Nauka, Moscow (1974)
3. Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: COLT 1992: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, pp. 144–152. ACM Press, New York (1992)
4. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer-Verlag New York, Inc. (1995)
5. Yang, J., Estivill-Castro, V., Chalup, S.K.: Support vector clustering through proximity graph modelling. In: Proceedings of the 9th International Conference on Neural Information Processing, ICONIP 2002, vol. 2. IEEE (2002)
6. Chen, Y., Zhou, X., Huang, T.S.: One-class SVM for learning in image retrieval. In: Proceedings of the International Conference on Image Processing, vol. 1. IEEE (2001)
7. Tsochantaridis, I., et al.: Support vector machine learning for interdependent and structured output spaces. In: Proceedings of the Twenty-First International Conference on Machine Learning. ACM (2004)

8. Angelov, P.: Outside the Box: An Alternative Data Analytics Framework. Journal of Automation, Mobile Robotics and Intelligent Systems **8**(2), 53–59 (2014)
9. Kangin, D., Angelov, P.: New Autonomously Evolving Classifier TEDAClass. IEEE Transactions on Cybernetics (2014, submitted)
10. Angelov, P.: Anomaly detection based on eccentricity analysis. In: 2014 IEEE Symposium Series on Computational Intelligence, 9-12 December, Orlando, Florida (2014, to appear, accepted to publication)
11. Angelov, P., Filev, D., Kasabov, N. (eds.) Evolving Intelligent Systems: Methodology and Applications. IEEE Press Series on Computational Intelligence, p. 444. John Willey and Sons (April 2010) ISBN: 978-0-470-28719-4
12. Angelov, P.: Autonomous Learning Systems: From Data Streams to Knowledge in Real time. John Willey and Sons (December 2012) ISBN: 978-1-1199-5152-0
13. Angelov, P., Kangin, D., Zhou, X., Kolev, D.: Symbol recognition with a new autonomously evolving classifier autoclass. In 2014 IEEE Conference on Evolving and Adaptive Intelligent Systems, pp. 1–6. IEEE (2014); DOI BibTeX
14. Angelov, P., Yager, R.: A new type of simplified fuzzy rule-based systems. International Journal of General Systems **41**(2), 163–185 (2012)
15. Angelov, P., Zhou, X., Klawonn, F.: Evolving fuzzy rule-based classifiers. In: First 2007 IEEE International Conference on Computational Intelligence Applications for Signal and Image Processing, April 1-5, Honolulu, Hawaii, USA, pp. 220–225 (2007)
16. Kasabov, N., Liang, L., Krishnamurthi, R., Feigin, V., Othman, M., Hou, Z., Parmar, P.: Evolving Spiking Neural Networks for Personalised Modelling of Spatio-Temporal Data and Early Prediction of Events: A Case Study on Stroke. Neurocomputing **134**, 269–279 (2014)
17. Bishop, C.M.: Pattern recognition and machine learning, pp. 325–358. Springer, New York (2006)
18. Schölkopf, B., Smola, A., Williamson, R.C., Bartlett, P.L.: New support vector algorithms. Neural Computation **12**, 1207–1245 (2000)
19. Mahalanobis, P.C.: On the generalised distance in statistics. Proceedings of the National Institute of Sciences of India **2**(1), 49–55 (1936)
20. Li, B., Han, L.: Distance weighted cosine similarity measure for text classification. In: Yin, H., Tang, K., Gao, Y., Klawonn, F., Lee, M., Weise, T., Li, B., Yao, X. (eds.) IDEAL 2013. LNCS, vol. 8206, pp. 611–618. Springer, Heidelberg (2013)
21. Keerthi, S.S., Lin, C.-J.: Asymptotic behaviors of support vector machines with Gaussian kernel. Neural Computation **15**(7), 1667–1689 (2003)
22. Maji, S., Berg, A.C., Malik, J.: Classification using intersection kernel support vector machines is efficient. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2008. IEEE (2008)
23. Chen, B., Zheng, N., Principe, J.C.: Survival kernel with application to kernel adaptive filtering. In: The 2013 International Joint Conference on IEEE Neural Networks (IJCNN) (2013)
24. Cauwenberghs, G., Poggio, T.: Incremental and Decremental Support Vector Machine Learning. NIPS, pp. 409–415 (2000)
25. Li, L.-J., Fei-Fei, L.: What, where and who? Classifying events by scene and object recognition. In: IEEE 11th International Conference on Computer Vision, ICCV 2007, pp. 1–8 (2007)
26. Oliva, A., Torralba, A.: Modeling the shape of the scene: a holistic representation of the spatial envelope. International Journal of Computer Vision **42**(3), 145–175 (2001). Gist Descriptor (Matlab code). http://people.csail.mit.edu/torralba/code/spatialenvelope/b