CENTER FOR
MACHINE PERCEPTION

CZECH TECHNICAL
UNIVERSITY

RESEARCH REPORT

A Novel Algorithm for Learning Support Vector Machines with Structured Output Spaces

Vojtěch Franc and Václav Hlaváč

xfrancv@cmp.felk.cvut.cz

K333–22/06, CTU–CMP–2006–04

May 2, 2006

Available at
ftp://cmp.felk.cvut.cz/pub/cmp/articles/franc/Franc-TR-2006-04.ps

Authors were supported by the EU projects COSPAL IST-004176 and PRINCESS INTAS 04-77-7347. The second author was also supported by The Czech Ministry of Education under Project CAK 1M0567.

Center for Machine Perception, Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University
Technická 2, 166 27 Prague 6, Czech Republic
fax +420 2 2435 7385, phone +420 2 2435 7637, www: http://cmp.felk.cvut.cz

Abstract

This report proposes a novel optimization algorithm for learning support vector machines (SVM) classifiers with structured output spaces introduced recently by Tsochantaridis et. al. Learning structural SVM classifier leads to a special instance of quadratic programming (QP) optimization with a huge number of constraints. The number of constraints is proportional to the cardinality of the output space which makes the QP task intractable by classical optimization methods. We propose a novel QP solver based on sequential minimal optimization (SMO). Unlike the original SMO, we propose a novel strategy for selecting variables to be optimized. The strategy aims at selecting such variables which yield the maximal improvement of optimization. We prove that the algorithm converges in a finite number of iterations to the solution which differs from the optimal one at most by a prescribed constant. Experiments performed show that the proposed algorithm is very competitive to a cutting plane algorithm of Tsochantaridis et. al. The proposed algorithm can be easily implemented and it does not require any external QP solver in contrary to the cutting plain algorithm. We demonstrated a capability of the algorithm on a problem of learning the Hidden Markov Network for color image segmentation and learning a structural classifier for car license plate recognition.

1 Introduction

The Support Vector Machines (SVM) [11] constitute the state-of-the-art approach to discriminative learning of classifiers from a labeled set of examples. The SVM have proven as powerful tool in numerous applications including optical character recognition, face detection, text categorization and many others. The original SVM learning is centered around the binary classification with the 0/1-loss function penalizing misclassifications. Recently, the SVM learning has been extended by Tsochantaridis et. al. [8] to more advanced setting which allows learning classifiers with structured output spaces. Structural classifiers are suitable for problems in which the estimated output is composed of an interdependent set of variables. For instance, the structural SVMs can be applied for supervised learning of classifiers based on the Hidden Markov Models (not only chains or trees) or probabilistic context-free grammars.

Learning structural SVM classifiers boils down to a specific instance of the quadratic programming (QP) task with a huge number of linear inequality constrains. The number of the constraints grows linearly with the cardinality of the output space. This makes the problem intractable by classical optimization methods. Tsochantaridis et. al. [8] proposed a cutting plain algorithm which finds an approximated solution in a polynomial time. The approximated solution guarantees that the constraints of the QP task are violated at most by a prescribed constant. The algorithm transforms solving the original problem into solving a series of QP tasks of a reduced size. Though the size of reduced QP tasks is much smaller compared to the original problem it can be still too large for conventional QP solvers. Moreover, a lot of reduced QP tasks must be solved which demands for fast QP solvers. We found standard general purpose packages for QP optimization improper for this problem (e.g., we tested the QUADPROG from the MATLAB optimization toolbox and the LOQO [9] optimization package).

In this report, we propose a novel QP solver based on the sequential minimal optimization (SMO) [4]. Unlike the original SMO which uses some heuristics, we propose a novel strategy for selecting variables to be optimized. The strategy aims at selecting such variables which yield the maximal improvement of the optimization. This selection strategy proved successful for optimization of a similar class of QP tasks [2]. In contrast to the cutting plane algorithm, our approach does not require an external QP solver. We proved that the proposed algorithm converges in a finite number of iterations to a solution which differs from the optimal one by a prescribed constant at most. We demonstrate that the method is very competitive to the cutting plane algorithm in terms of convergence speed. Last but not least the proposed algorithm can be easily implemented. We demonstrated a capability of the algorithm on a problem on two practical problems. First, we show an application to learning of the Hidden Markov Network for color image segmentation and, second, learning of a structural classifier for car license plate recognition.

Used notation

We use upper-case bold letters to denote matrices, e.g., H. Vectors are implicitly column vectors and are denoted by lower-case bold italic letters, e.g., $\boldsymbol{\alpha}$. The entries of matrices and vectors are denoted by unbold letters, e.g., H_{ij} denotes the entry in i-th row and j-th column of the matrix H; α_i is the i-the entry of the vector $\boldsymbol{\alpha}$. We use $[\cdot]_i$ to denote i-th entry of a vector given by an expression, e.g., $[\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}]_i$ is the i-th entry of the vector given by $\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}$. A dot product between vectors is denoted by $\langle \cdot, \cdot \rangle$, e.g., $\langle \boldsymbol{w}, \boldsymbol{z} \rangle$ is the dot product between vectors $\boldsymbol{w}$ and $\boldsymbol{z}$. We use operator $[\![\cdot]\!]$ which equals 1 if the logical expression in the brackets holds true and it is 0 otherwise.

2 Support Vector Machines with structured output spaces

This section describes the problem of learning a structural SVM classifier and the instance of the QP task to which the learning leads.

Let the analyzed object be described by an observable state $x \in \mathcal{X}$ and a hidden state $y \in \mathcal{Y}$. The set of hidden states $\mathcal{Y}$ is assumed to be finite. It is assumed that a pair (x, y) is a realization of random variables with an unknown joint distribution $P(x, y)$. We are given a set of examples $\mathcal{T}_{\mathcal{X}\mathcal{Y}} = \{(x_1, y_1), \ldots, (x_m, y_m)\} \in (\mathcal{X} \times \mathcal{Y})^m$ i.i.d. sampled from $P(x, y)$. Let $f \colon \mathcal{X} \to \mathcal{Y}$ denote a classifier for estimating a hidden state y from an observation x. Let $L \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ be some loss function which penalizes the estimation $f(x)$ given the true hidden state y. The goal is to find the rule f^* which minimizes the expected risk

$$R[f] = \int_{\mathcal{X} \times \mathcal{Y}} L(y, f(\boldsymbol{x})) dP(x, y) \,. \tag{1}$$

The generative approach to minimize $R[f]$ consists of estimating a distribution $\tilde{P}(x, y)$ from examples $\mathcal{T}_{XY}$ and minimizing (1) with $\tilde{P}(x, y)$ substituted for $P(x, y)$. This approach can be problematic for two reasons at least. First, the estimation of $\tilde{P}(x, y)$ may be a difficult or even intractable problem (e.g., estimation of a general Gibbs distribution). Second, we have no information how $R[f]$ changes after the estimated $\tilde{P}(x, y)$ is substituted and, eventually, the resulting classifier is evaluated on a finite sample.

The discriminative approach is an alternative way to minimize $R[f]$. It requires to determine a class of rules $f \in \mathcal{F}$ and to use a substitutional risk functional $\tilde{R}[f]$ which well approximates $R[f]$ and can be evaluated without unknown $P(x, y)$. Solving $f^* = \operatorname{argmin}_{f \in \mathcal{F}} \tilde{R}[f]$ is referred to as learning. For instance, a simple substitution of the expected risk $R[f]$ is the empirical risk

$$R_{emp}[f] = \frac{1}{m} \sum_{k=1}^{m} L(y_k, f(x_k)) \,.$$

The Support Vector Machines (SVM) [10] implement the discriminative approach for a class of linear classifiers

$$f(x; \boldsymbol{w}) = \operatorname*{argmax}_{y \in \mathcal{Y}} \langle \boldsymbol{w}, \Psi(x, y) \rangle \,. \tag{2}$$

The classifier is determined by a parameter vector $\boldsymbol{w} \in \mathbb{R}^d$ and a mapping $\Psi: \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$. The parameter vector $\boldsymbol{w}$ is learned from the examples while the mapping Ψ must be known beforehand. Originally, SVMs were designed for a binary case $|\mathcal{Y}| = 2$ and 0/1-loss function $L(y, y') = 0$ if $y = y'$ and $L(y, y') = 1$ if $y \neq y'$. The multi-class $|Y| > 0$ variants of SVM learning exist [1, 11] but they still assume the 0/1-loss function only.

Tsochantaridis et. al. [8] extended the SVM learning such that more complex loss functions can be used. In the concrete, the loss function can be arbitrary function $L: \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ which satisfies $L(y, y') = 0$ if $y = y'$ and $\infty > L(y, y') > 0$ if $y \neq y'$. This extension is advantageous for learning of structural classifiers, e.g., Hidden Markov Networks, for which the 0/1-loss function can be rather artificial.

Let $\mathcal{K} = \{1, \ldots, m\}$ be a set of indices. Learning of structural SVM classifiers leads to solving the following QP task

$$(\boldsymbol{w}^*, \boldsymbol{\xi}^*) = \operatorname*{argmin}_{\boldsymbol{w}, \boldsymbol{\xi}} \left(\frac{1}{2} \|\boldsymbol{w}\|^2 + \frac{C}{m} \sum_{k \in \mathcal{K}} \xi_k \right) \,, \tag{3}$$

subject to one of the following two feasible sets. In the first case, the feasible set reads

$$\begin{array}{rcll} \langle \boldsymbol{w}, \Psi(x_k, y_k) - \Psi(x_k, y) \rangle & \geq & L(y_k, y) - \xi_k \,, & \forall k \in \mathcal{K}, \forall y \in \mathcal{Y} \setminus \{y_k\} \,, \\ \xi_k & \geq & 0 \,, & \forall k \in \mathcal{K} \,, \end{array} \tag{4}$$

and it is denoted as the *margin re-scaling* approach. In the second case, the feasible set reads

$$\begin{array}{rcll} \langle \boldsymbol{w}, \Psi(x_k, y_k) - \Psi(x_k, y) \rangle & \geq & 1 - \dfrac{\xi_k}{L(y_k, y)} \,, & \forall k \in \mathcal{K}, \forall y \in \mathcal{Y} \setminus \{y_k\} \,, \\ \xi_k & \geq & 0 \,, & \forall k \in \mathcal{K} \,, \end{array} \tag{5}$$

and it is denoted as the *slack re-scaling* approach. The objective function of (3) comprises the regularization term $\frac{1}{2}\|\boldsymbol{w}\|^2$ and the sum $\frac{1}{m} \sum_{k \in \mathcal{K}} \xi_k$ weighted by the regularization constant $C > 0$. It can be shown [8] that the sum $\frac{1}{m} \sum_{k=1}^{m} \xi_k$ is an upper bound on the empirical risk $R_{emp}[f]$ of the linear classifier (2). Thus the objective is to minimize an upper bound on the empirical error and to penalize solutions with high $\|\boldsymbol{w}\|^2$. The trade-of between these two terms is controlled by the regularization constant C usually selected by a cross-validation. For detailed discussion on a difference between the margin-rescaling and the slack rescaling we refer to [8].

3 Proposed algorithm

Our goal is to propose an efficient QP solver for optimizing of the task (3) with constrains (4) and (5). Though two constraint sets are considered they give rise to a single QP task. The QP task (3) can be equivalently written as

$$(\boldsymbol{w}^*, \boldsymbol{\xi}^*) = \operatorname*{argmin}_{\boldsymbol{w},\boldsymbol{\xi}} Q_P(\boldsymbol{w},\boldsymbol{\xi}) = \operatorname*{argmin}_{\boldsymbol{w},\boldsymbol{\xi}} \left(\frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{m}\sum_{k\in\mathcal{K}} \xi_k \right) , \tag{6}$$

subject to

$$\langle \boldsymbol{w}, \boldsymbol{z}_i \rangle \geq b_i - \xi_k \,, \quad \forall k \in \mathcal{K},\, \forall i \in \mathcal{I}_k \,. \tag{7}$$

The symbol $\mathcal{I} = \{1,\dots,n\} = \mathcal{I}_1 \cup \ldots \cup \mathcal{I}_m$, $\mathcal{I}_1 \cap \ldots \cap \mathcal{I}_m = \emptyset$ stands for a set of indices its number equals to $n = m|\mathcal{Y}|$. The correspondence between the variables in (6) and primal task (3) is obvious. In the case of the margin re-scaling constraints (4), it holds: $\boldsymbol{z}_i = \Psi(x_k, y_k) - \Psi(x_k, y)$, $b_i = L(y_k, y)$, i is a linear index corresponding to k and y, i.e., $i = y + (k-1)|\mathcal{Y}|$ provided $\mathcal{Y} = \{1,\dots,|\mathcal{Y}|\}$. In the case of the slack re-scaling constraints (5), the variables are the same except for $\boldsymbol{z}_i = L(y_k, y)(\Psi(x_k, y_k) - \Psi(x_k, y))$. Notice, that the constraints $\xi_k \geq 0$ were absorbed to constraints $\langle \boldsymbol{w}, \boldsymbol{z}_i \rangle \geq b_i - \xi_k$ because $\boldsymbol{z}_i = 0$ and $b_i = 0$ for $y = y_k$.

The dual formulation (Wolf dual) of (6) reads

$$\boldsymbol{\alpha}^* = \operatorname*{argmax}_{\boldsymbol{\alpha}} Q_D(\boldsymbol{\alpha}) = \operatorname*{argmax}_{\boldsymbol{\alpha}} \left(\langle \boldsymbol{b}, \boldsymbol{\alpha} \rangle - \frac{1}{2}\langle \boldsymbol{\alpha}, \mathrm{H}\boldsymbol{\alpha} \rangle \right) , \tag{8}$$

subject to

$$\begin{aligned} \sum_{i\in\mathcal{I}_k} \alpha_i &= \frac{C}{m} \,, \quad \forall k \in \mathcal{K} \,, \\ \alpha_i &\geq 0 \,, \quad \forall i \in \mathcal{I} \,. \end{aligned} \tag{9}$$

The matrix $\mathrm{H} \in \mathbb{R}^{n\times n}$ contains dot products $H_{i,j} = \langle \boldsymbol{z}_i, \boldsymbol{z}_j \rangle$, $\boldsymbol{b} = [b_1,\dots,b_n]^T$ is a given vector and $\boldsymbol{\alpha} = [\alpha_1,\dots,\alpha_n]^T$ is the vector of optimized dual variables. Let $\mathcal{A}_D \subset \mathbb{R}^n$ denote a feasible set of the dual task (8), i.e., all the vectors $\boldsymbol{\alpha}$ which satisfy the constraints (9). Having the solution of the dual task $\boldsymbol{\alpha}^*$ the primal variables are computed as

$$\boldsymbol{w}^* = \sum_{i\in\mathcal{I}} \boldsymbol{z}_i \alpha_i^* \,, \quad \text{and} \quad \xi_k^* = \max_{i\in\mathcal{I}_k} [\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^*]_i \,,\ \forall k \in \mathcal{K} \,. \tag{10}$$

Substituting kernel functions for dot products allows for learning non-linear classifiers. In this paper, we concentrate solely to the linear case but deriving the kernel version is straightforward.

We propose an iterative algorithm which optimizes the dual task (8). The optimization problem is transformed to a sequence of auxiliary problems which have the same objective function $Q_D(\boldsymbol{\alpha})$ but they have much simpler auxiliary feasible sets $\mathcal{A}_L^{(0)}, \mathcal{A}_L^{(1)}, \dots, \mathcal{A}_L^{(t)}$. The superscript $^{(t)}$ denotes the iteration number. Maximizing $Q_D(\boldsymbol{\alpha})$ with respect to the auxiliary feasible sets yields a sequence of solution vectors $\boldsymbol{\alpha}^{(0)}, \boldsymbol{\alpha}^{(1)}, \dots, \boldsymbol{\alpha}^{(t)}$. The sequence $\mathcal{A}_L^{(0)}, \mathcal{A}_L^{(1)}, \dots, \mathcal{A}_L^{(t)}$ is constructed such that the sequence $Q_D(\boldsymbol{\alpha}^{(0)}) < Q_D(\boldsymbol{\alpha}^{(1)}) < \ldots < Q_D(\boldsymbol{\alpha}^{(t)})$ converges to the optimal solution $Q_D(\boldsymbol{\alpha}^*)$. The auxiliary feasible sets are line segments

$$\mathcal{A}_L^{(t)} = \left\{ \boldsymbol{\alpha} \middle| \boldsymbol{\alpha} = (1-\tau)\boldsymbol{\alpha}^{(t)} + \tau\boldsymbol{\beta}^{(t)}, 0 \leq \tau \leq 1 \right\} , \tag{11}$$

where $\boldsymbol{\alpha}^{(t)}$ is the current solution and $\boldsymbol{\beta}^{(t)}$ is a properly selected vector. The optimization of $Q_D(\boldsymbol{\alpha}^{(t)})$ over a line segment $\mathcal{A}_L^{(t)}$ can be computed efficiently because it has an analytical solution which will be shown in Section 3.1. The vectors $\boldsymbol{\beta}^{(t)}$ are selected to be from the feasible set $\mathcal{A}_D$ and thus the vectors $\boldsymbol{\alpha}^{(t)}$ are also feasible.

3.1 Optimization over the line segment

Let us consider the problem of maximizing $Q_D(\boldsymbol{\alpha})$ over a line segment $\mathcal{A}_L^{(t)}$ given by (11). The problem is equivalent to searching for the maximum of an univariate quadratic function

$$\begin{aligned} Q_L^{(t)}(\tau) &= Q_D((1-\tau)\boldsymbol{\alpha}^{(t)} + \tau\boldsymbol{\beta}^{(t)}) \\ &= (1-\tau)\langle \boldsymbol{b}, \boldsymbol{\alpha}^{(t)}\rangle + \tau\langle \boldsymbol{b}, \boldsymbol{\beta}^{(t)}\rangle - \frac{1}{2}(1-\tau)^2\langle \boldsymbol{\alpha}^{(t)}, \mathrm{H}\boldsymbol{\alpha}^{(t)}\rangle \\ &\quad -\tau(1-\tau)\langle \boldsymbol{\beta}^{(t)}, \mathrm{H}\boldsymbol{\alpha}^{(t)}\rangle - \frac{1}{2}\tau^2\langle \boldsymbol{\beta}^{(t)}, \mathrm{H}\boldsymbol{\beta}^{(t)}\rangle \,. \end{aligned}$$

over the closed interval $0 \leq \tau \leq 1$. The maximum is attained at the vector

$$\boldsymbol{\alpha}^{(t+1)} = (1-\tau^{(t)})\boldsymbol{\alpha}^{(t)} + \tau^{(t)}\boldsymbol{\beta}^{(t)} , \tag{12}$$

where

$$\tau^{(t)} = \underset{0 \leq \tau \leq 1}{\operatorname{argmax}}\, Q_L^{(t)}(\tau) \,. \tag{13}$$

It is obvious that $Q_L^{(t)}(0) = Q_D(\boldsymbol{\alpha}^{(t)})$ and $Q_L^{(t)}(1) = Q_D(\boldsymbol{\beta}^{(t)})$. The derivative of $Q_L^{(t)}(\tau)$ reads

$$\begin{aligned} \mathrm{d}Q_L^{(t)}(\tau) &= \langle \boldsymbol{b}, \boldsymbol{\beta}^{(t)} - \boldsymbol{\alpha}^{(t)}\rangle + (1-\tau)\langle \boldsymbol{\alpha}^{(t)}, \mathrm{H}\boldsymbol{\alpha}^{(t)}\rangle \\ &\quad -(1-2\tau)\langle \boldsymbol{\beta}^{(t)}, \mathrm{H}\boldsymbol{\alpha}^{(t)}\rangle - \tau\langle \boldsymbol{\beta}^{(t)}, \mathrm{H}\boldsymbol{\beta}^{(t)}\rangle \,. \end{aligned} \tag{14}$$

To guarantee the improvement $Q_D(\boldsymbol{\alpha}^{(t+1)}) - Q_D(\boldsymbol{\alpha}^{(t)}) > 0$, the vector $\boldsymbol{\beta}^{(t)}$ must be selected such that the derivative $\mathrm{d}Q_L^{(t)}(\tau)$ evaluated at $\tau = 0$ is positive, i.e.,

$$\mathrm{d}Q_L^{(t)}(0) = \langle \boldsymbol{\beta}^{(t)} - \boldsymbol{\alpha}^{(t)}, \boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}\rangle > 0 \,. \tag{15}$$

Assuming the condition (15) holds the maximum of $Q_L^{(t)}(\tau)$ w.r.t. $0 \leq \tau \leq 1$ is attained either within the open interval $0 < \tau < 1$ or on its boundary value $\tau = 1$. Therefore solution of (13) can be found by solving the equality $\mathrm{d}Q_L^{(t)}(\tau) = 0$ for τ. If the resulting τ exceeds 1 then the optimum is attained at $\tau = 1$. This allows to express the solution of (13) in a closed form

$$\tau^{(t)} = \min\left(1, \frac{\langle \boldsymbol{\beta}^{(t)} - \boldsymbol{\alpha}^{(t)}, \boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}\rangle}{\langle \boldsymbol{\alpha}^{(t)} - \boldsymbol{\beta}^{(t)}, \mathrm{H}(\boldsymbol{\alpha}^{(t)} - \boldsymbol{\beta}^{(t)})\rangle}\right) \,. \tag{16}$$

To sum up, the vector $\boldsymbol{\alpha}^{(t+1)}$ maximizing the objective function $Q_D(\boldsymbol{\alpha})$ over the line segment $\mathcal{A}_L^{(t)}$ between vectors $\boldsymbol{\alpha}^{(t)}$ and $\boldsymbol{\beta}^{(t)}$ is computed using formulas (16) and (12).

If the condition (15) holds then the optimization will lead to an improvement $\Delta^{(t)} = Q_D(\boldsymbol{\alpha}^{(t+1)}) - Q_D(\boldsymbol{\alpha}^{(t)}) > 0$. An exact value of the improvement can be derived substituting (12) and (16) to $\Delta^{(t)} = Q_D(\boldsymbol{\alpha}^{(t+1)}) - Q_D(\boldsymbol{\alpha}^{(t)})$. For $\tau^{(t)} < 1$ the improvement equals to

$$\Delta^{(t)} = \frac{\langle \boldsymbol{\beta}^{(t)} - \boldsymbol{\alpha}^{(t)}, \boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)} \rangle^2}{2\langle \boldsymbol{\alpha}^{(t)} - \boldsymbol{\beta}^{(t)}, \mathrm{H}(\boldsymbol{\alpha}^{(t)} - \boldsymbol{\beta}^{(t)}) \rangle} \tag{17}$$

and for $\tau^{(t)} = 1$ we get

$$\Delta^{(t)} = \left\langle \boldsymbol{\beta}^{(t)} - \boldsymbol{\alpha}^{(t)}, \boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)} \right\rangle - \frac{1}{2} \left\langle \boldsymbol{\alpha}^{(t)} - \boldsymbol{\beta}^{(t)}, \mathrm{H}\boldsymbol{\alpha}^{(t)} - \boldsymbol{\beta}^{(t)} \right\rangle . \tag{18}$$

3.2 Stopping conditions

The algorithm iterates until it gets sufficiently close to the optimal solution which is indicated by a stopping condition. We assume the stopping condition based on the duality gap

$$Q_P(\boldsymbol{w}^{(t)}, \boldsymbol{\xi}^{(t)}) - Q_D(\boldsymbol{\alpha}^{(t)}) \leq \varepsilon . \tag{19}$$

The vector $\boldsymbol{\alpha}^{(t)}$ is a solution produced by the algorithm after t iterations and vectors $(\boldsymbol{w}^{(t)}, \boldsymbol{\xi}^{(t)})$ are primal variables determined from $\boldsymbol{\alpha}^{(t)}$ by (10). The prescribed number $\varepsilon > 0$ is a desired precision. The proposed algorithm produces only feasible solutions therefore duality gap is always non-negative. Because the optimized QP task is convex the strong duality theorem guarantees that the duality gap vanishes at the optimum. Any solution satisfying (19) differs from the optimal one at most by the prescribed ε, i.e., $Q_P(\boldsymbol{w}^{(t)}, \boldsymbol{\xi}^{(t)}) - Q_P(\boldsymbol{w}^*, \boldsymbol{\xi}^*) \leq \varepsilon$ and $Q_D(\boldsymbol{\alpha}^*) - Q_D(\boldsymbol{\alpha}^{(t)}) \leq \varepsilon$ holds. Substituting (10) to (19) allows to rewrite the stopping condition in a concise form

$$\frac{C}{m} \sum_{k \in \mathcal{K}} \max_{i \in \mathcal{I}_k} [\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}]_i - \langle \boldsymbol{\alpha}^{(t)}, \boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)} \rangle \leq \varepsilon . \tag{20}$$

For practical purposes it is also advantageous to use a scale-invariant stopping condition

$$Q_P(\boldsymbol{w}^{(t)}, \boldsymbol{\xi}^{(t)}) - Q_D(\boldsymbol{\alpha}^{(t)}) \leq \varepsilon |Q_D(\boldsymbol{\alpha}^{(t)})| . \tag{21}$$

Henceforth we will consider only the stopping condition (19) for simplicity. The applications of the scale-invariant version (21) will be straightforward.

Finally, let us discuss the stopping condition used by the cutting plane algorithm of Tsochantaridis et. al. [8]. This method aims at selecting a subset of constraints with indices $\overline{\mathcal{I}}_k \subset \mathcal{I}_k$, $\forall k \in \mathcal{K}$, which induces a reduced QP task. The constraints are being gradually added such that the reduced QP task approximates well the original QP task. The algorithm finishes after the constraints of the original QP task are violated at most by prescribed $\overline{\varepsilon}$, i.e., the conditions

$$\max_{i \in \mathcal{I}_k} [\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}]_i - \max_{i \in \overline{\mathcal{I}}_k} [\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}]_i \leq \overline{\varepsilon}, \qquad \forall k \in \mathcal{K} , \tag{22}$$

hold. A disadvantage of using (22) as the stopping condition might be an unclear relation between the parameter $\overline{\varepsilon}$ and the objective function. That is the reason why we prefer stopping condition expressed explicitly in terms of the optimized criterion.

3.3 QP solver

In this section, we derive an iterative QP solver which uses the line segment approximation of the feasible set. The line segment is constructed such that the optimization changes just two variables per iteration. Two is the minimal number of variables which can be changed without violating the constraints. This resembles the Sequential Minimal Optimizer (SMO) [4] for the QP task with a single equality constraint. In contrast to the SMO which uses some heuristics, we aim at strategy which selects such two variables yielding the maximal improvement of the optimization. This selection strategy was proposed in [2] for solving a similar class of QP tasks.

The optimized variables $\boldsymbol{\alpha}^{(t)} = [\alpha_1^{(t)}, \ldots, \alpha_n^{(t)}]^T$ form m subsets $\mathcal{I}_1 \cup \ldots \cup \mathcal{I}_m = \{1, \ldots, n\}$. The variables with indices $\mathcal{I}_k$ are binded by a single linear constraint as follows from (9). It is reasonable to construct the optimization algorithm such that it passes through the individual subsets and, if necessary, it updates just the corresponding variables leaving the other fixed. To check whether the subset of variables with indices $\mathcal{I}_k$ needs to be updated we evaluate the condition

$$\frac{C}{m} \max_{i \in \mathcal{I}_k} [\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}]_i - \sum_{i \in \mathcal{I}_k} \alpha_i^{(t)} [\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}]_i > \frac{\varepsilon}{m} \,. \tag{23}$$

If the condition (23) holds then the variables with indices $\mathcal{I}_k$ are good candidates for the updated. In this case, as will be shown bellow, the improvement of the optimization is guaranteed to be greater than some positive constant. Notice, that if the condition (23) is violated for all $k \in \mathcal{K}$, i.e., no variables are to be updated, then the condition (20) holds. This can be seen by summing up both sides of (23) over all $k \in \mathcal{K}$ and comparing the result with (20).

Let (23) be satisfied for some $k \in \mathcal{K}$ and let u, v be two indices selected out of $\mathcal{I}_k$. We construct the line segment $\mathcal{A}_L^{(t)}$ between the current solution $\boldsymbol{\alpha}^{(t)}$ and the vector $\boldsymbol{\beta}^{(t)}$ given as follows

$$\beta_i^{(t)} = \begin{cases} \alpha_u^{(t)} + \alpha_v^{(t)} & \text{if} \quad i = u \,, \\ 0 & \text{if} \quad i = v \,, \\ \alpha_i^{(t)} & \text{if} \quad i \neq u \wedge i \neq v \,. \end{cases} \tag{24}$$

The optimization over the line segment $\mathcal{A}_L^{(t)}$ leads to an improvement $\Delta_{(u,v)}^{(t)}$ which is a function of the indices u and v used to construct the vector $\boldsymbol{\beta}^{(t)}$. Formulas for optimization over the line segment and computation of the improvement are given in Section 3.1 for general vector $\boldsymbol{\beta}^{(t)}$. Thanks to the particular form of $\boldsymbol{\beta}^{(t)}$ given by (24) the formulas for improvement further simplify. In particular, if the inequality

$$[\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}]_u - [\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}]_v < \alpha_v^{(t)} (H_{uu} - 2H_{uv} + H_{vv})$$

holds then the improvement reads

$$\Delta_{(u,v)}^{(t)} = \frac{\Big([\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}]_u - [\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}]_v \Big)^2}{2(H_{uu} - 2H_{uv} + H_{vv})} \tag{25}$$

otherwise

$$\Delta_{(u,v)}^{(t)} = \alpha_v^{(t)} \Big([\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}]_u - [\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}]_v \Big) - \frac{1}{2} \Big(\alpha_v^{(t)} \Big)^2 \Big(H_{uu} - 2H_{uv} + H_{vv} \Big) \,.$$

There are several options how to determine the indices u, v out of $\mathcal{I}_k$. Checking all possible combinations to maximize the improvement would require squared number of operations. We found a simpler selection strategy requiring only linear number of operations which yields nearly the same improvement as optimizing over both the indices. The strategy is based on optimizing the predominant term $[\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}]_u$ of the improvement formulas w.r.t. u while the other terms are ignored. Having u selected it is fixed and the improvement is maximized w.r.t. index v, i.e., the selection strategy reads

$$u \in \operatorname*{argmax}_{i \in \mathcal{I}_k} [\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}]_i \quad \text{and} \quad v \in \operatorname*{argmax}_{i \in \mathcal{J}_{(t,u)}} \Delta^{(t)}_{(u,i)} \,, \tag{26}$$

where

$$\mathcal{J}_{(t,u)} = \left\{ i \;\middle|\; i \in \mathcal{I}_k \wedge \left([\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}]_u > [\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}]_i \right) \wedge \alpha_i^{(t)} > 0 \right\} \,.$$

It follows from (15) that the improvement $\Delta^{(t)}_{(u,v)}$ is positive for any $v \in \mathcal{J}_{(t,u)}$. Having the procedure to construct the vector $\boldsymbol{\beta}^{(t)}$ we can use the line segment optimization described in Section 3.1. The resulting QP solver is described by Algorithm 1.

Algorithm 1: QP solver

1. Select arbitrary $\boldsymbol{\alpha}^{(0)} \in \mathcal{A}_D$.
2. For $k = 1, \ldots, m$, do:

   If the condition (23) holds then do:

   (a) Find the indices u and v using (26).

   (b) Compute

   $$\tau^{(t)} = \min\left(1, \frac{[\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}]_u - [\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}]_v}{\alpha_v^{(t)}(H_{uu} - 2H_{uv} + H_{vv})} \right) \,. \tag{27}$$

   (c) Compute a new solution $\boldsymbol{\alpha}^{(t+1)} = [\alpha_1^{(t+1)}, \ldots, \alpha_n^{(t+1)}]^T$ as

   $$\alpha_i^{(t+1)} = \begin{cases} \alpha_u^{(t)} + \tau^{(t)}\alpha_v^{(t)} & \text{if} \quad i = u \,, \\ \alpha_v^{(t)} - \tau^{(t)}\alpha_v^{(t)} & \text{if} \quad i = v \,, \\ \alpha_i^{(t)} & \text{if} \quad i \neq u \wedge i \neq v \,. \end{cases} \tag{28}$$

3. If the stopping optimality condition (20) holds then exit otherwise go to Step (2).

A bottleneck of Algorithm 1 is computing the vector $\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}$ which, computed from a scratch, requires $\mathcal{O}(n^2)$ operations. However, this can be circumvented by maintaining a vector $\boldsymbol{g}^{(t)} = \boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}$. It follows from (28) that the vector $\boldsymbol{g}^{(t+1)} = [g^{(t+1)}, \ldots, g_n^{(t+1)}]^T$ can be updated in each iteration as

$$g_i^{(t+1)} = g_i^{(t)} + \alpha_v^{(t)}\tau^{(t)}(H_{iv} - H_{iu}) \,,$$

which requires $\mathcal{O}(n)$ operations. Hence the total number of operations required by Algorithm 1 per iteration reduces to $\mathcal{O}(n)$.

Theorem 3.1 *Let $R^2 = \max_{i\in\mathcal{I},j\in\mathcal{I}}(H_{ii} - 2H_{ij} + H_{jj})$. Algorithm 1 returns $\boldsymbol{\alpha}^{(t)}$ which satisfies for any $\varepsilon > 0$ the stopping condition (19) after at most $t_{max} < \infty$ iterations where*

$$t_{max} = \Big(Q_D(\boldsymbol{\alpha}^*) - Q_D(\boldsymbol{\alpha}^{(0)})\Big)\,\frac{2(n-m)C^2R^2}{\varepsilon^2}\,.$$

PROOF: Two cases can occur when computing $\tau^{(t)}$ in Step 2b. In the first case, $\tau^{(t)} < 1$ and the improvement is given by (25). The numerator of (25) is given by indices u and v. Let us check the improvement gained when using indices

$$u \in \operatorname*{argmax}_{i\in\mathcal{I}_k}[\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}]_i \quad \text{and} \quad v \in \operatorname*{argmin}_{i\in\mathcal{J}_{(t,u)}}[\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}]_i\,,$$

which are definitely examined by the used selection strategy (26). The solution is updated only if the condition (23) holds. Multiplying the condition (23) by $\frac{m}{C}$ we get

$$\max_{i\in\mathcal{I}_k}[\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}]_i - \frac{m}{C}\sum_{i\in\mathcal{I}_k}\alpha_i^{(t)}[\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}]_i > \frac{\varepsilon}{C}\,. \tag{29}$$

The first term on the left hand side of (29) equals to $[\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}]_u$. The second term is a convex combination of numbers $[\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}]_i$, $i \in \mathcal{I}_k$, i.e., it must be greater or equal to $[\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}]_v$ and thus

$$[\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}]_u - [\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}^{(t)}]_v > \frac{\varepsilon}{C}\,. \tag{30}$$

The inequality (30) can be used to lower bound the numerator of the improvement (25). It is seen that the denominator of (25) can be bounded from above by $2R^2$. Using these bounds on numerator and denominator of (25) yields

$$\Delta_{min} \ge \frac{\varepsilon^2}{2C^2R^2} \qquad \text{for} \qquad \tau^{(t)} < 1\,. \tag{31}$$

In the second case, $\tau^{(t)} = 1$ the improvement cannot be bounded but the number of non-zero variables in $\boldsymbol{\alpha}^{(t+1)}$ decreases by one as seen from (28). It follows from the constraints (9) that the minimal number of non-zero variables equals to m. Therefore the case $\tau^{(t)} = 1$ cannot occur more than $(n-m)$ times in a in line. Using this reasoning and (31) yields the inequality

$$Q_D^* \ge \left(Q(\boldsymbol{\alpha}^{(0)}) + \frac{t}{n-m}\Delta_{min}\right) = \left(Q(\boldsymbol{\alpha}^{(0)}) + \frac{t\varepsilon^2}{(n-m)2C^2R^2}\right)\,,$$

which can be further rearranged to

$$t \le \Big(Q_D(\boldsymbol{\alpha}^*) - Q_D(\boldsymbol{\alpha}^{(0)})\Big)\,\frac{2(n-m)C^2R^2}{\varepsilon^2}\,,$$

which ends the proof.

3.4 Application to learning structural SVM classifiers

This section shows how to apply the proposed Algorithm 1 to learning structural SVM classifiers. We first concentrate on the margin re-scaling variant, i.e., solving the QP task (3) with constraints (4).

Algorithm 1 uses explicit representation of the matrix H, the vector $\boldsymbol{b}$ and the solution vector $\boldsymbol{\alpha}^{(t)}$. A large-scale variant of the algorithm can be derived by maintaining only parts of H, $\boldsymbol{b}$ corresponding to the non-zero entries of $\boldsymbol{\alpha}^{(t)}$. The selection strategy (26) searches for the index $u \in \mathcal{I}_k$ which is equivalent to solving

$$\hat{y} = \operatorname*{argmax}_{y \in \mathcal{Y}} \left(L(y_k, y) + \Big\langle \boldsymbol{w}, \Psi(x_k, y) \Big\rangle \right) . \tag{32}$$

We require (similarly as the algorithm of Tsochantaridis et. el. [8]) that an algorithm for solving the task (32) is available. The task (32) is a modification of the recognition problem (2) hence its likely that its solution is at ones disposal.

We found practical to split the algorithm into two parts. The first part optimizes over selected variables corresponding to H, $\boldsymbol{b}$ maintained in memory. The second part adds new entries of H, $\boldsymbol{b}$ if necessary. The method relies on the fact that the algorithm converges before H, $\boldsymbol{b}$ are too high. Fortunately, this is the case in most applications.

The method is described by Algorithm 2. In step 2, a reduced QP task is solved calling Algorithm 1 as a sub-procedure. The reduced QP task need not be solved precisely. To guarantee convergence, Algorithm 1 must perform at least one update. In our experiments, we set a limit on the number of updates to 500. In some applications, in which solving the task (32) is time consuming, it is better to optimize the reduced QP task precisely. In step 3, new constraints which correspond to variables being a good candidates for optimization are added. To ensure convergence, it is sufficient to add just one constraint in each step. However, we found that adding more constraints increases speed of convergence. In our experiments, we add 100 constraints in each step 3.

The slack re-scaling variant of Algorithm 2 is obtained by a slight change of step 3(a). It requires solving

$$\hat{y} := \operatorname*{argmax}_{y \in \mathcal{Y}} L(y_k, y) \left(1 - \Big\langle \boldsymbol{w}, \Psi(x_k, y_k) - \Psi(x_k, y) \Big\rangle \right) ,$$

and setting $\hat{\boldsymbol{z}} := L(y_k, \hat{y})(\Psi(x_k, y_k) - \Psi(x_k, \hat{y}))$. The rest of the algorithm is kept unchanged.

It is seen, that kernel variant of Algorithm (2) can be easily derived substituting $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i \boldsymbol{z}_i$ for all occurrences of $\boldsymbol{w}$.

4 Experiments

We implemented the proposed algorithm in MATLAB code with the core QP solver written in C language[1]. The algorithm can be applied to a variety of problems. It is enough to supply a problem specific procedure solving the modified recognition task in step 3(a) of Algorithm 2.

We experimentally compared a speed of convergence of the proposed algorithm to the state-of-the-art SVM^{struct} software. Results are presented in Section 4.1. We also applied the algorithm on two real applications. In Section 4.2, results of learning the Hidden Markov Network for a color image segmentation are presented. In Section 4.3, we show application to learning of a structural classifier for a car license plate recognition.

[1]The code will soon appear at `http://cmp.felk.cvut.cz`

Algorithm 2: Learning of structural SVM classifier (margin re-scaling variant)

1. Initialization: $n := 0$, $\mathcal{Z} := \emptyset$, $\mathcal{B} := \emptyset$, $\mathcal{I}_k := \emptyset$, $\forall k \in \mathcal{K}$.
2. If $n = 0$ then set $\boldsymbol{w} := 0$ else do:
   (a) Construct dot product matric $\mathrm{H} \in \mathbb{R}^{n\times n}$ from vectors $\mathcal{Z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n\}$ and vector $\boldsymbol{b} \in \mathbb{R}^n$ from numbers $\mathcal{B} = \{b_1, \ldots, b_n\}$.
   (b) Call Algorithm 1 to compute vector $\boldsymbol{\alpha}$ solving the reduced QP task given by H, $\boldsymbol{b}$ and $\mathcal{I}_k$, $k \in \mathcal{K}$. Set vector $\boldsymbol{w} := \sum_{i=1}^n \alpha_i \boldsymbol{z}_i$.
3. For $k = 1, \ldots, m$, do:
   (a) Solve the modified recognition task

   $$\hat{y} := \operatorname*{argmax}_{y\in\mathcal{Y}} \left(L(y_k, y) + \Big\langle \boldsymbol{w}, \Psi(x_k, y) \Big\rangle \right)$$

   and set $\hat{\boldsymbol{z}} := \Psi(x_k, y_k) - \Psi(x_k, \hat{y})$, $\hat{b} := L(y_k, \hat{y})$ and $\xi_k := \hat{b} - \langle \boldsymbol{w}, \hat{\boldsymbol{z}} \rangle$.
   (b) If the following conditions

   $$\frac{C}{m}(\hat{b} - \langle \boldsymbol{w}, \hat{\boldsymbol{z}} \rangle) - \sum_{i\in\mathcal{I}_k} \alpha_i [\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}]_i > \frac{\varepsilon}{m} \quad \text{and} \quad \hat{b} - \langle \boldsymbol{w}, \hat{\boldsymbol{z}} \rangle > \max_{i\in\mathcal{I}_k} [\boldsymbol{b} - \mathrm{H}\boldsymbol{\alpha}]_i$$

   hold then set $n := n + 1$, $\boldsymbol{z}_n := \hat{\boldsymbol{z}}$, $b_n := \hat{b}$ and update

   $$\mathcal{B} := \mathcal{B} \cup b_n \,, \qquad \mathcal{Z} := \mathcal{Z} \cup \boldsymbol{z}_n \,, \qquad \mathcal{I}_k := \mathcal{I}_k \cup n \,.$$

4. If the stopping condition

   $$\frac{C}{m} \sum_{k\in\mathcal{K}} \xi_k - \big(\langle \boldsymbol{b}, \boldsymbol{\alpha} \rangle - \|\boldsymbol{w}\|^2 \big) \leq \varepsilon$$

   holds then exit otherwise go to Step 2.

	USPS	ISOLET	LANDSAT	VOWEL
Num. of examples m	7,291	6,238	4,435	990
Num. of classes N_Y	10	26	6	11
Num. of optimized variables n	72,910	162,188	26,610	10,890

Table 1: Multi-class classification problems used for benchmarking.

4.1 Comparison to SVM^{struct} software

We compared the proposed approach to SVM^{struct} software[2] which is a state-of-the-art implementation of the cutting plane algorithm of Tsochantaridis et. al. [8]. The SVM^{struct} uses a large-scale QP optimizer proposed by Joachims [3]. SVM^{struct} package supports learning of (i) multi-class SVM classifiers with single output variable, (ii) probabilistic context-free grammars and (iii) aligning sequences. We used the multi-class SVM classifier learning for comparison because the recognition task is simple and most time is spent on the QP optimization.

The multi-class classifier assigns input feature vector $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ into class $y \in \mathcal{Y} = \{1, \ldots, N_Y\}$ using the classification rule

$$f(\boldsymbol{x}; \boldsymbol{w}) = \operatorname*{argmax}_{y \in \mathcal{Y}} \langle \boldsymbol{w}_y, \boldsymbol{x} \rangle \,. \tag{33}$$

The rule (33) is a particular case of the general linear classifier (2). The 0/1-loss function is assumed which leads to the ordinary formulation of learning multi-class SVM classifiers introduced by Singer and Crammer [1]. Notice, that in the case of 0/1-loss the margin re-scaling and slack re-scaling approaches coincide. The modified recognition task (32) for this particular problem reads

$$\hat{y} = \operatorname*{argmax}_{y \in \mathcal{Y}} \Big([\![y_k \neq y]\!] + \langle \boldsymbol{w}_y, \boldsymbol{x}_k \rangle \Big) \,,$$

and it can be solved simply by exhaustive search.

We used USPS database of handwritten characters and three multi-class classification problems selected from the UCI machine learning repository[3] for comparison. Dataset are summarized in Table 1.

The SVM^{struct} uses the stopping condition (22) different from that we propose. To make comparison fair we proceeded as follows. The classifier was first learned by the SVM^{struct} and its quality was assessed by the attained value of the primal criterion $Q_P(\boldsymbol{w}, \boldsymbol{\xi})$. Then we run our algorithm to find the solution with the same quality, i.e., with the value of $Q_P(\boldsymbol{w}, \boldsymbol{\xi})$ equal or less than the solution of SVM^{struct}. We run the experiment for different value of the regularization parameter C which influences complexity of the QP task. The speed of convergence was measured in the CPU time used for QP optimization on Linux machine with AMD 1600 MHz processor. The results obtained are summarized in Table 2. The convergence speed is also visualized in Figure 1. It is seen that both methods have a similar rate of convergence. The proposed algorithm converged faster for higher values C in the case of USPS, ISOLET and VOWEL. It was slower for LANDSAT data set. It can be also observed that the proposed algorithm

[2] `http://svmlight.joachims.org`

[3] `http://www.ics.uci.edu/~mlearn/MLRepository.html`

produced sparser solution (i.e., less support vectors) in all the cases. The proposed algorithm required slightly higher number of constraints of the reduced QP task. We also recorded the relative difference $\frac{Q_P - Q_D}{|Q_D|}$ to show relationship between parameter $\bar{\varepsilon}$ of the stopping condition (22) which was set to 0.01 in all the experiments (default setting of SVM^{struct}).

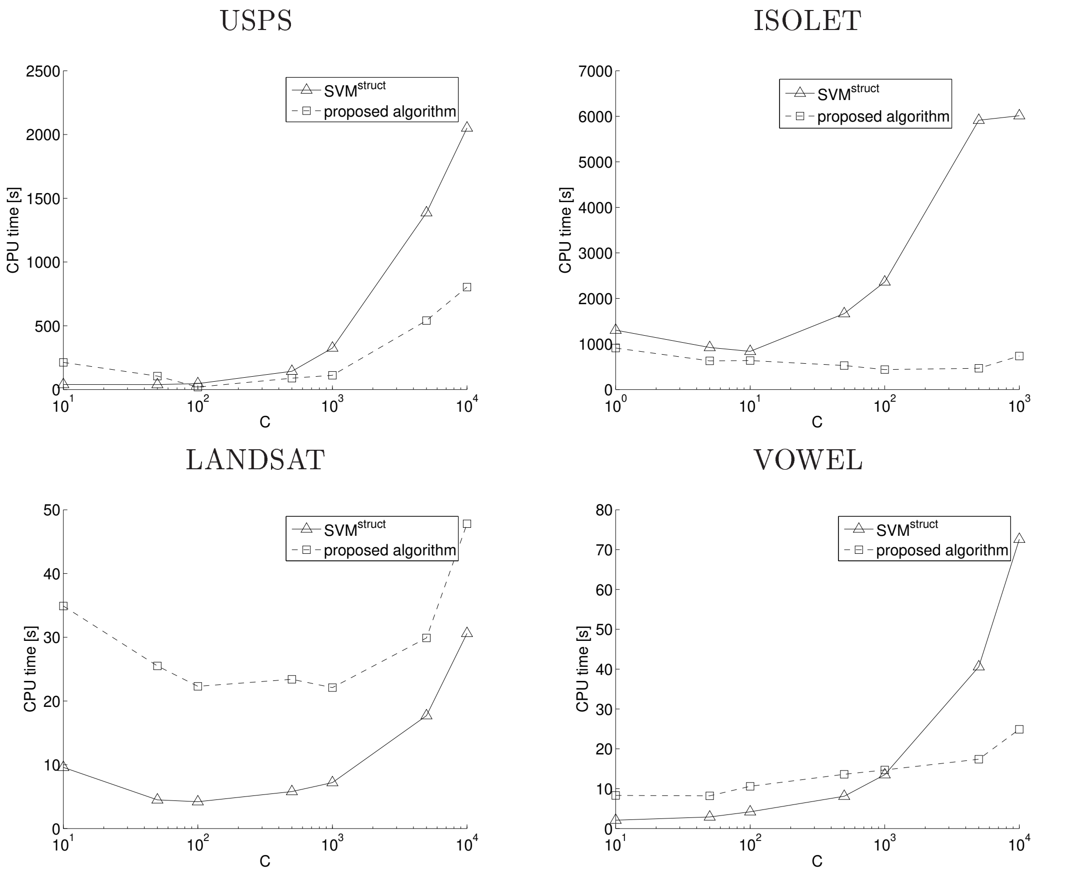

Figure 1: Convergence speed of the proposed algorithm compared to the SVM^{struct}.

4.2 Learning of Hidden Markov Networks

We consider a problem of learning the Hidden Markov Network (HMN) for image segmentation. The image is composed of a set of pixels $\mathcal{T} = \{(i,j) \mid i \in \{1,\ldots,W\}, j \in \{1,\ldots,H\}\}$. The observable state is a discrete function $x\colon \mathcal{T} \to X$ which assigns to each pixel $t \in \mathcal{T}$ corresponding value of RGB color channel $x(t) \in X = \{1,\ldots,N_X\}$. We used 3 bits for each channel which means that $N_X = 2^{3\cdot 3} = 512$. The hidden state is a discrete function $y\colon \mathcal{T} \to Y$ which assigns to each pixel $t \in \mathcal{T}$ a label $y(t) \in \mathcal{Y} = \{1,\ldots,N_Y\}$. In our particular application we assume $N_Y = 6$ labels each corresponding to a different color segment. A relation between the observable and hidden state of a single pixel is given by a discrete function $q\colon X \times Y \to \mathbb{R}$. A relation between hidden states of two pixels is given by discrete function $g\colon Y \times Y \to \mathbb{R}$. Let the functions q and g be repre-

USPS

	C	10	50	100	500	1000	5000	10000
SVM^{struct}	*time* [s]	39.6	38.8	46.3	143.5	325.2	1386.5	2051.6
	nsv	4353	2920	2474	2007	1897	1847	1824
	constr	4354	4265	3847	3416	3479	3696	3664
proposed	*time* [s]	212.5	105.4	85.9	88.7	111.4	541.3	803.6
	nsv	4202	2644	2225	1708	1576	1537	1434
	constr	6517	4700	3900	3300	3400	5600	6500
	$\frac{(Q_P - Q_D)}{\lvert Q_D \rvert}$	$2 \cdot 10^{-10}$	$1 \cdot 10^{-5}$	$8 \cdot 10^{-5}$	$5 \cdot 10^{-4}$	$1 \cdot 10^{-3}$	$2 \cdot 10^{-3}$	$4 \cdot 10^{-3}$

ISOLET

	C	1	5	10	50	100	500	1000
SVM^{struct}	*time* [s]	1304	923.5	837.7	1668.1	2362.8	5913.0	5705.3
	nsv	6722	5743	5028	3696	3412	3201	3211
	constr	10735	8373	7203	5630	5374	5328	5341
proposed	*time* [s]	911.1	630.4	637.2	526.9	438.6	447.9	735.0
	nsv	6968	5504	4724	3146	2781	2414	2473
	constr	12500	9400	8400	5900	4900	4700	5300
	$\frac{(Q_P - Q_D)}{\lvert Q_D \rvert}$	$2 \cdot 10^{-7}$	$2 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$6 \cdot 10^{-4}$	$1 \cdot 10^{-3}$	$5 \cdot 10^{-3}$	$1 \cdot 10^{-2}$

LANDSAT

	C	10	50	100	500	1000	5000	10000
SVM^{struct}	*time* [s]	9.6	4.5	4.4	5.8	7.2	17.7	30.6
	nsv	2939	2272	2034	1774	1708	1611	1575
	constr	4608	3435	3158	2867	2774	2734	2671
proposed	*time* [s]	34.9	25.5	22.3	23.4	22.1	29.9	47.8
	nsv	2875	2222	1997	1725	1631	1581	1502
	constr	4295	3420	3126	2954	2800	2977	3163
	$\frac{(Q_P - Q_D)}{\lvert Q_D \rvert}$	$3 \cdot 10^{-5}$	$8 \cdot 10^{-9}$	$2 \cdot 10^{-8}$	$3 \cdot 10^{-6}$	$9 \cdot 10^{-5}$	$8 \cdot 10^{-5}$	$1 \cdot 10^{-4}$

VOWEL

	C	10	50	100	500	1000	5000	10000
SVM^{struct}	*time* [s]	2.1	3.0	4.2	8.1	13.5	40.6	72.6
	nsv	1086	1101	1102	2110	1072	1021	999
	constr	1774	2002	2094	2110	2101	2102	2018
proposed	*time* [s]	8.4	8.2	10.6	13.6	14.7	17.4	24.9
	nsv	1101	1092	1085	1078	1049	1007	992
	constr	2117	2083	2078	2215	2207	2182	2433
	$\frac{(Q_P - Q_D)}{\lvert Q_D \rvert}$	$1 \cdot 10^{-7}$	$1 \cdot 10^{-5}$	$6 \cdot 10^{-7}$	$1 \cdot 10^{-6}$	$2 \cdot 10^{-5}$	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$

Table 2: Comparison between SVM^{struct} and the proposed algorithm. C is regularization constant; *nsv* is the number of non-zero variables; *constr* is the number of constraints of the reduced QP task.

sented as a vector $\boldsymbol{w} \in \mathbb{R}^{N_X \cdot N_Y + N_Y \cdot N_Y}$. A hidden state y_t of a pixel $t \in \mathcal{T}$ is assumed to be related to its four nearest neighbors (except for the boundary pixels). A graph of dependences of the hidden states forms 2-dimensional grid with edges $\mathcal{E} \subset \mathcal{T} \times \mathcal{T}$. The estimation of the hidden states y for given observations x leads to solving

$$f(x;\boldsymbol{w}) = \operatorname*{argmax}_{y \in \mathcal{Y}} \langle \boldsymbol{w}, \Psi(x,y) \rangle = \operatorname*{argmax}_{y \in \mathcal{Y}} \left(\sum_{t \in \mathcal{T}} q\big(x(t), y(t)\big) + \sum_{(t,t') \in \mathcal{E}} g\big(y(t), y(t')\big) \right) . \tag{34}$$

The linear classification rule (34) corresponds to the maximum aposteriori (MAP) estimation of random variables with the Gibbs distribution. The task (34), referred to as the max-sum problem, is known to be NP-complete problem in general. Fortunately, there exist algorithms which find an approximated solution in a polynomial time. We used an approach of Schlesinger's [6] to solve the max-sum problem. Namely, we used implementation described in [12] its code is public domain[4].

We learned the unknown parameters $\boldsymbol{w}$ by the proposed algorithm solving the task (3) with margin-rescaling constraints (4). The training set contained $m = 10$ examples, i.e., pairs (x, y) of RGB image x and a ground truth segmentation y produced by a human. The training images are of size 100×100 pixels. Figure 4.2 shows a sample of the first 4 training pairs. We assume the loss function

$$L(y, y') = \sum_{t \in \mathcal{T}} [\![y(t) \neq y'(t)]\!] ,$$

meaning that $L(y, y')$ equals to the number of pixels in which the segmentation y differs from y'. The learning algorithm requires solving the modified recognition task (32) which, in this particular case, reads

$$\hat{y} = \operatorname*{argmax}_{y \in \mathcal{Y}} \left(\sum_{t \in \mathcal{T}} \Big(q\big(x_k(t), y(t)\big) + [\![y_k(t) \neq y(t)]\!] \Big) + \sum_{(t,t') \in \mathcal{E}} g\big(y(t), y(t')\big) \right) . \tag{35}$$

The task (35) is also instance of the max-sum problem. The stopping condition (19) is given by the value of the primal $Q_P(\boldsymbol{w}, \boldsymbol{\xi})$ and the dual $Q_D(\boldsymbol{\alpha})$ criterion. In this particular case, however, the value $Q_P(\boldsymbol{w}, \boldsymbol{\xi})$ is only estimated because we cannot guaranteed that the modified recognition task is always solved optimally. This is caused by the NP-completeness of the problem max-sum problem. As a result it is not possible to say how far we are from the optimal solution after the algorithm exits. However, it is not an obstacle for the practical use. The regularization constant $C = 10$ was found by cross-validation. Figure 3 shows segmentation of testing images hold out of the learning phase.

Let us note, that only special cases of HMN with chain or tree structure $\mathcal{E}$ have been considered in literature so far. Solely Taskar et. al. [7] proposed an algorithm for learning HMN with an arbitrary structure but they consider only a special class of associative Markov networks (AMN). AMN require that the functions $g_{tt'}$ are of a special form

$$g_{tt'}(y(t), y(t')) = \begin{cases} 1 & \text{if} \quad y(t) = y(t') , \\ \lambda_{tt'} & \text{if} \quad y(t) \neq y(t') . \end{cases}$$

[4] `http://cmp.felk.cvut.cz/cmp/cmp_software.html`

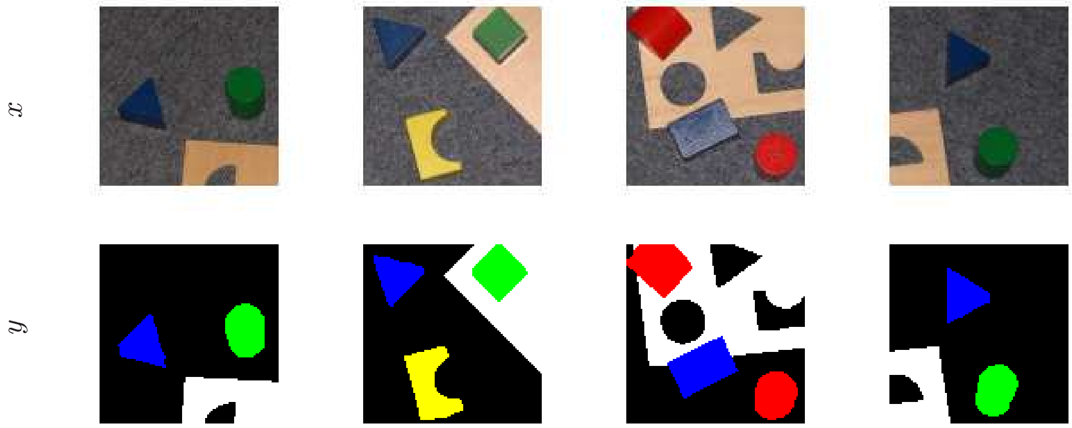

Figure 2: Examples of the first four training pairs of the observable and hidden states. In this applications the observable state is an RGB image and the hidden state is its segmentation.

We do not require this restriction and the functions $g_{tt'}$ can be of an arbitrary form as well as the structure $\mathcal{E}$. According to our best knowledge the SVM learning of a general HMN described in this report has not been published so far.

4.3 Learning car license plate recognition system

We consider a problem of learning car license plate recognition (CLPR) system from images. We use an approach similar to Savchynskyy and Kamotskyy [5] suitable for learning a general optical character recognition systems. Savchynskyy and Kamotskyy used perceptron algorithm for learning, i.e., they require a noiseless training data which can be classified without any error.

The input image is composed of a set of pixels $\mathcal{T} = \{(i,j) \mid i \in \{1,\ldots,W\}, j \in \{1,\ldots,H\}\}$. The observable state is a discrete function $x \colon \mathcal{T} \to \mathbb{R}$ which assigns each pixel $t \in \mathcal{T}$ a intensity value $x(t) \in \mathbb{R}$. The hidden state $y = (s_1,\ldots,s_L) \in \mathcal{Y}$ is a sequences of segments $s \in S$. The $y \in \mathcal{Y}$ is referred to as a segmentation and $L(y)$ denotes its length. A segment $s = (a,j)$ is a pair of a character identifier $a \in A$ and its beginning $j \in \mathbb{N}$. We use $a(s)$ to denote the character and $j(s)$ to denote the beginning of the given segment s. We further assume that a width $\omega \colon A \to \mathbb{N}$ of each character is known. In the concrete, the set of characters A is composed of numerals, letters, gap symbols and a symbol for dash line. The beginning $j(s)$ determines a position of the first column of the character $a(s)$ in the image. An example of an input image an its segmentation is given in Figure 4.

An appearance of each character $a \in A$ is modeled by a template image $e(a)$. The width of the template $e(a)$ equals $\omega(a)$ and its height is the same as the height H of the input image. A template $e(a)$ determines an ideal appearance of the character a without a noise. The set of all templates $\{e(a) \mid a \in A\}$ is unknown and its is represented as the parameter vector $\boldsymbol{w} \in \mathbb{R}^d$.

Let $col(x,j) = [x(j,1),\ldots,x(j,H)]^T$ denote a vector of intensity values of the j-the column of the input image. Similarly, $col(e,j)$ is the j-th column of the template image

x $\qquad\qquad$ $f(x;\boldsymbol{w})$

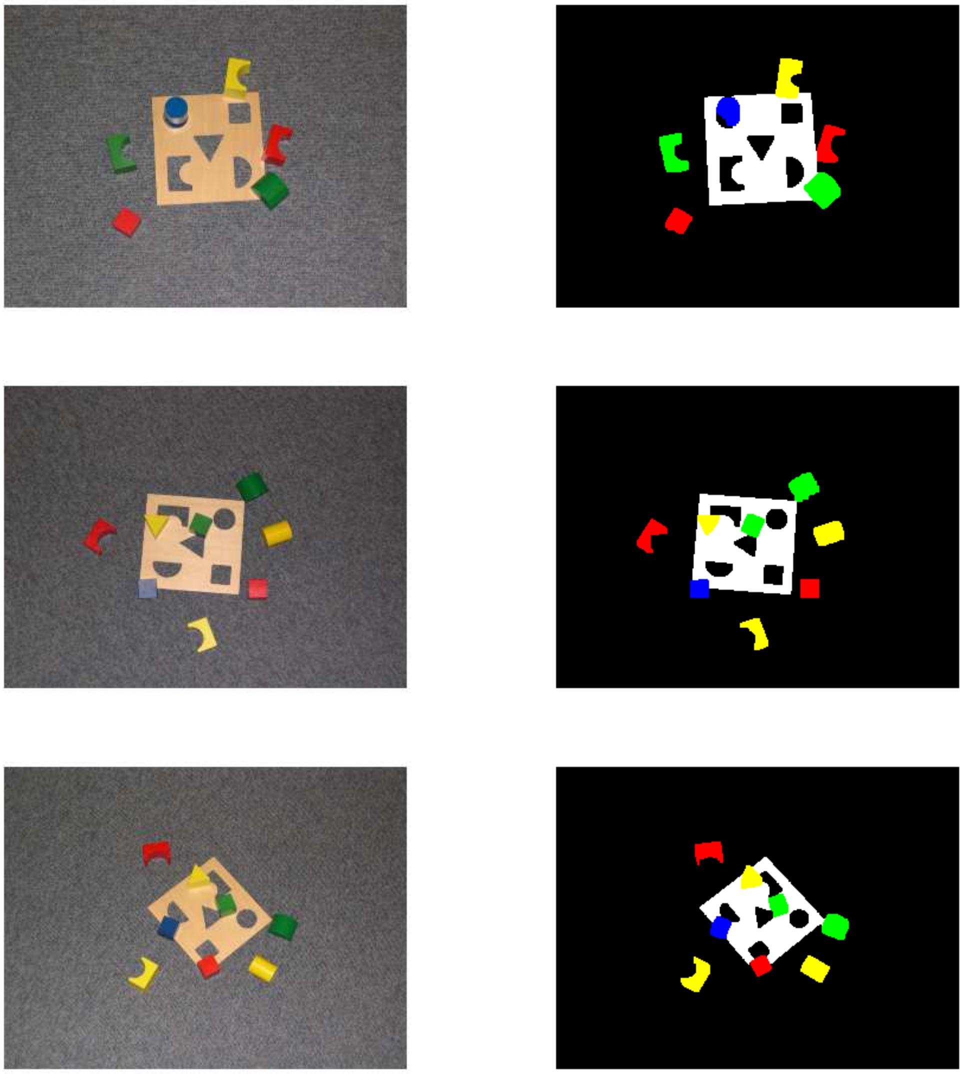

Figure 3: Examples of segmentation of testing images estimated by the learned Hidden Markov Network.

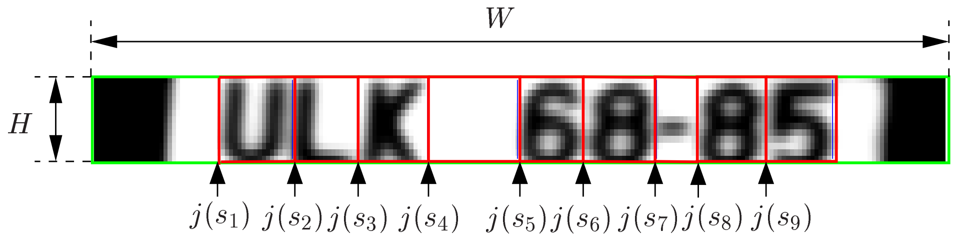

Figure 4: An example of input image and its segmentation $y = (s_1, \ldots, s_9)$. The positions $(j(s_1), \ldots, j(s_9))$ of character segments are denoted by arrows. The corresponding characters are $a(s_1) =' U'$, $a(s_2) =' L'$, ..., $a(s_9) =' 5'$.

e. The recognition task can be seen as a linear classifier

$$y^* = \operatorname*{argmax}_{y \in \mathcal{Y}} \langle \boldsymbol{w}, \Psi(x,y) \rangle = \operatorname*{argmax}_{y \in \mathcal{Y}} \sum_{i=1}^{L(y)} \sum_{l=1}^{\omega(s_i)} \Big\langle col\big(x, l+j(s_i)-1\big), col\big(e(a(s_i)), l\big) \Big\rangle . \quad (36)$$

Notice, that the recognition task, in contrast to common approaches, performs simultaneously segmentation and classification of the characters.

The set of all segmentations $\mathcal{Y}$ is determined by all admissible configurations of characters on a license plate and its placements in the input image. In our particular case, we assume two types of license plates most frequently used in Czech Republic their examples are shown in Figure 5. Thanks to a strict structure of the license plate the recognition task (36) can be solved simply by an exhaustive search. As a result the modified recognition task required by learning algorithm can be solved simply as well.

We used the loss function $L(y, y')$ which adds penalty 10 for each misclassified character and it adds penalty $|j(s_i) - j(s'_i)|$ for each incorrectly positioned character.

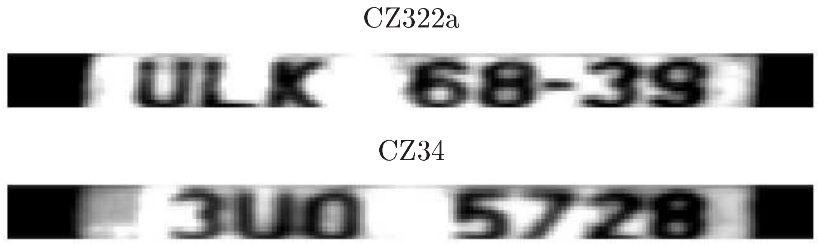

Figure 5: Examples two types of car license plates considered in conducted experiments.

We applied the proposed algorithm to learn the unknown templates from a training set of 1660 images of license plates and their corresponding segmentations provided by a human. We used the margin-rescaling approach. The results obtained were evaluated on a testing set of 300 license plates held out of the learning stage. As a baseline approach, we used the same type of classifier but with templates computed simply by averaging

	Misclassified license plates	Misclassified characters
Averaging	26.08%	4.92%
SVM	0.67%	0.01%

Table 3: Evaluation of car license plate recognition system on 300 testing data. The license plate is considered misclassified whenever one character is misclassified at least.

the training examples of individual characters. The results are presented in Table 3. Figure 6 shows some examples of recognized license plates.

5 Conclusions

In this report we proposed novel algorithm for optimizing the QP task required in learning SVM classifiers with structured output spaces. The proposed algorithm is based on the sequential minimal optimization with a novel strategy for selecting variables to be optimized. We proved that the algorithm convergences in a finite number of iterations. We experimentally showed that the algorithm is competitive to SVM^{struct} software which is the state-of-the-art implementation of the cutting plane algorithm of Tsochantaridis et. al. In contrast to the cutting plane algorithm, the proposed method does not require an external QP solver and it can be easily implemented. We demonstrated the capability of the algorithm on the problem of learning the Hidden Markov Network for color image segmentation and learning a structural classifier for a car license plate recognition.

References

[1] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

[2] V. Franc. *Optimization Algorithms for Kernel Methods*. Phd thesis, Center for Machine Perception, K13133 FEE Czech Technical University, Prague, Czech Republic, July 2005.

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

[4] J.C. Platt. Fast training of support vectors machines using sequential minimal optimization. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods*, pages 185–208. The MIT Press, 1998.

[5] B. Savchynskyy and O. Kamotskyy. Character templates learning for textual image recognition as an example of learning in structural recognition. In *2nd IEEE International Conference on Document Image Analysis for Libraries*, 2006. To be published.

[6] M.I. Schlesinger. Sintaksicheskiy analiz dvumernykh zritelnikh signalov v usloviyakh pomekh (syntactic analysis of two-dimensional visual signals in noisy conditions). *Kibernetika*, (4):113–130, 1976. In Russian.

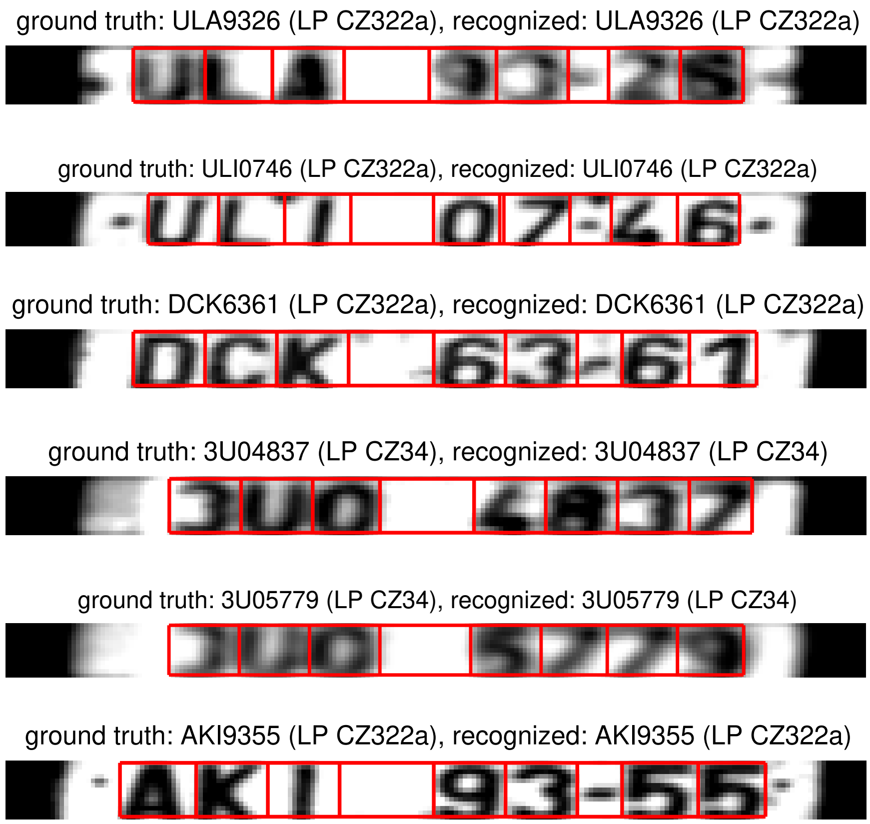

Figure 6: Examples of testing data with estimated segmentation of characters depicted on the license plates. The ground truth and estimated classification of the characters is printed on top of the images.

[7] B. Taskar, V. Chatalbashev, and D. Koller. Learning associative markov networks. In *Proceedings of the Twenty-First International Conference on Machine Learning (ICML)*, 2004.

[8] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

[9] R.J. Vanderbei. LOQO: an interior point code for quadratic programming. Technical report, Program in Statistics & Operations Research, Princeton University, 1994.

[10] V. Vapnik. *The nature of statistical learning theory*. Springer Verlag, 1995.

[11] V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, Inc., 1998.

[12] T. Werner. A linear programming approach to max-sum problem. Research Report CTU–CMP–2005–25, Center for Machine Perception, K13133 FEE Czech Technical University, Prague, Czech Republic, December 2005.