Invariant Pattern Recognition by Semidefinite Programming Machines

Thore Graepel
Microsoft Research Ltd.
Cambridge, UK
thoreg@microsoft.com

Ralf Herbrich
Microsoft Research Ltd.
Cambridge, UK
rherb@microsoft.com

Abstract

Taking into account local invariances with respect to known pattern transformations can greatly improve the accuracy of classification. Previous approaches are either based on regularisation or on the generation of virtual (transformed) examples. We develop a new framework for learning linear classifiers under known transformations based on semidefinite programming. We present a new learning algorithm—the Semidefinite Programming Machine (SDPM)—which is able to find a maximum margin hyperplane when the training examples are polynomial trajectories instead of single points. The solution is found to be sparse in dual variables and allows to identify those points on the trajectory with minimal real-valued output as virtual support vectors. Extensions to segments of trajectories, to more than one transformation parameter, and to learning with kernels are presented. In experiments we use a Taylor expansion to locally approximate rotational invariance in pixel images from USPS and find improvements over known methods, which can be seen as approximations to the SDPM method.

1 Introduction

One of the central problems of pattern recognition is the exploitation of known invariances in the pattern domain. In images these invariances may include rotation, translation, shear, scaling, brightness, and lighting direction. In addition, specific domains such as handwritten digit recognition may exhibit invariances such as line thinning/thickening and other non-uniform deformations [8]. In the field of speech recognition, say phoneme classification, invariances include variations in base frequency or duration. When using machine learning approaches the challenge is to combine the training sample with the knowledge of invariances to obtain a good classifier.

Various approaches to the problem of incorporating invariances have been suggested. Possibly the most straightforward way of incorporating invariances is by including virtual examples into the training sample which have been generated from actual examples by the application of the invariance $\mathbf{T} : \mathbb{R}_\theta \times \mathbb{R}^n_\mathbf{x} \to \mathbb{R}^n_\mathbf{x}$ at some fixed $\theta \in \mathbb{R}_\theta$. This approach is well exemplified by the method of virtual support vectors [7].

Images $\mathbf{x}$ subjected to the transformation $\mathbf{T}(\theta, \cdot)$ describe a (complex) trajectory or manifold in pixel space. The tangent distance [8] approximates the distance between the trajectories (manifolds) by the distance between their tangent vectors (planes) at a given value $\theta = \theta_0$ and can be used with any kind of distance-based classifier. Another approach, tangent prop [8], incorporates the invariance $\mathbf{T}$ directly into the objective function for learning by penalising large values of the derivative of the classification function w.r.t. the given transformation parameter. A similar regulariser can be applied in the framework of support vector machines [1].

We take up the idea of considering the trajectory given by the combination of training vector and transformation. While data in machine learning are commonly represented as vectors $\mathbf{x} \in \mathbb{R}^n$ we instead consider more complex training examples each of which is represented as a (usually infinite) set

$$\{\mathbf{T}(\theta, \mathbf{x}_i) \; : \; \theta \in \mathbb{R}\} \subset \mathbb{R}^n \,, \tag{1}$$

which constitutes a trajectory in $\mathbb{R}^n$. Our goal is to learn a linear classifier that separates well the training trajectories belonging to different classes. In practice, we may be given a "standard" training example $\mathbf{x}$ together with a differentiable transformation $\mathbf{T}$ representing an invariance of the learning problem. The problem can be solved if the transformation $\mathbf{T}$ is approximated by a transformation $\tilde{\mathbf{T}}$ polynomial in θ, e.g., a Taylor expansion of the form

$$\tilde{\mathbf{T}}(\theta, \mathbf{x}_i) \approx \mathbf{T}(0, \mathbf{x}_i) + \sum_{j=1}^{r} \theta^j \cdot \left(\frac{1}{j!} \left. \frac{d^j \mathbf{T}(\theta, \mathbf{x}_i)}{d\theta^j} \right|_{\theta=0} \right) = \mathbf{T}(0, \mathbf{x}_i) + \sum_{j=1}^{r} \theta^j \cdot (\mathbf{X}_i)_{j,\cdot} \,. \tag{2}$$

Our approach is based on a powerful theorem by Nesterov [5] which states that the set $\mathcal{P}_{2l}^{+}$ of polynomials of degree $2l$ non-negative on the entire real line is a convex set representable by positive semidefinite (psd) constraints. Hence, optimisation over $\mathcal{P}_{2l}^{+}$ can be formulated as a semidefinite program (SDP). Recall that an SDP [9] is given by a linear objective function which is minimised subject to a linear matrix inequality (LMI),

$$\underset{\mathbf{w} \in \mathbb{R}^n}{\text{minimise}} \;\; \mathbf{c}^\top \mathbf{w} \quad \text{subject to} \quad \mathbf{A}(\mathbf{w}) := \sum_{j=1}^{n} w_j \mathbf{A}_j - \mathbf{B} \succeq \mathbf{0} \,, \tag{3}$$

with $\mathbf{A}_j \in \mathbb{R}^{m \times m}$ for all $j \in \{0, \ldots, n\}$. The LMI $\mathbf{A}(\mathbf{w}) \succeq \mathbf{0}$ means that $\mathbf{A}(\mathbf{w})$ is required to be positive semidefinite, i.e., that for all $\mathbf{v} \in \mathbb{R}^n$ we have $\mathbf{v}^\top \mathbf{A}(\mathbf{w}) \mathbf{v} = \sum_{j=1}^{n} w_j \left(\mathbf{v}^\top \mathbf{A}_j \mathbf{v} \right) - \mathbf{v}^\top \mathbf{B} \mathbf{v} \geq 0$ which reveals that LMI constraints correspond to infinitely many linear constraints. This expressive power can be used to enforce constraints for training examples as given by (1), i.e., constraints required to hold for all values $\theta \in \mathbb{R}$. Based on this representability theorem for non-negative polynomials we develop a learning algorithm—the Semidefinite Programming Machine (SDPM)—that maximises the margin on polynomial training samples, much like the support vector machine [2] for ordinary single vector data.

The structure of the paper is as follows: In Section 2 we develop the theory of SDPMs with a focus on conceptual clarity rather than technical details[1]. In Section 3 we discuss extensions to the basic SDPM as presented in Section 2, specifically, the restriction to segments of trajectories, an extension to more than one transformation parameter, and learning with kernels. Section 4 presents experimental results for a comparison with known methods.

[1] For more technical details, the interested reader is referred to a work-in-progress report at *http://www.research.microsoft.com/~rherb/papers/sdpm-tr.ps.gz*.

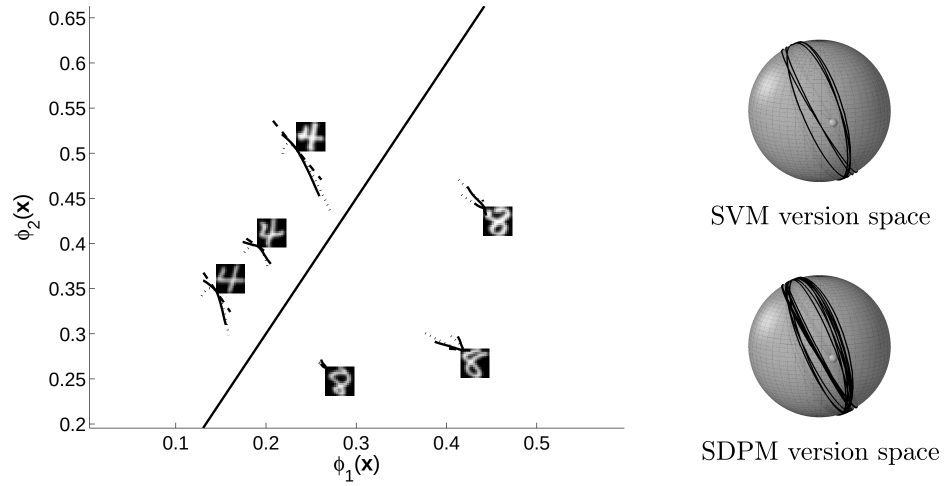


Figure 1: (**Left**) Approximated trajectories for rotated USPS images (2) for $r = 1$ (dashed line) and $r = 2$ (dotted line). The features are the mean pixel intensities in the top and bottom half of the image. (**Right**) Set of weight vectors $\mathbf{w}$ which are consistent with the six images (top) and the six trajectories (bottom). The dot corresponds to the separating plane in the left plot.

2 Semidefinite Programming Machines

2.1 Linear Classifiers and Polynomial Examples

We consider binary classification problems and linear classifiers. Given a training sample $((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)) \in (\mathbb{R}^n \times \{-1, +1\})^m$ we aim at learning a weight vector[2] $\mathbf{w} \in \mathbb{R}^n$ to classify examples $\mathbf{x}$ by $y(\mathbf{x}) = \operatorname{sign}(\mathbf{w}^\top \mathbf{x})$. Assuming linear separability of the training sample the principle of empirical risk minimisation recommends finding a weight vector $\mathbf{w}$ such that for all $i \in \{1, \ldots, m\}$ we have $y_i \mathbf{w}^\top \mathbf{x}_i \geq 0$. As such this constitutes a linear feasibility problem and is easily solved by the perceptron algorithm [6]. Additionally requiring the solution to maximise the margin leads to the well-known quadratic program of support vector learning [2].

In order to be able to cope with known invariances $\mathbf{T}(\theta, \cdot)$ we would like to generalise the above setting to the following feasibility problem:

$$\text{find } \mathbf{w} \in \mathbb{R}^n \quad \text{such that} \quad \forall i \in \{1, \ldots, m\} : \forall \theta \in \mathbb{R} : \quad y_i \mathbf{w}^\top \mathbf{x}_i(\theta) \geq 0\,, \tag{4}$$

that is we would require the weight vector to classify correctly every transformed training example $\mathbf{x}_i(\theta) := \mathbf{T}(\theta, \mathbf{x}_i)$ for every value of the transformation parameter θ. The situation is illustrated in Figure 1. In general, such a set of constraints leads to a very complex and difficult-to-solve feasibility problem. As a consequence, we consider only transformations $\tilde{\mathbf{T}}(\theta, \mathbf{x})$ of polynomial form, i.e., $\tilde{\mathbf{x}}_i(\theta) := \tilde{\mathbf{T}}(\theta, \mathbf{x}_i) = \mathbf{X}_i^\top \boldsymbol{\theta}$, each polynomial example $\tilde{\mathbf{x}}_i(\theta)$ being represented by a polynomial in the row vectors of $\mathbf{X}_i \in \mathbb{R}^{(r+1)\times n}$, with $\boldsymbol{\theta} := (1, \theta, \ldots, \theta^r)^\top$. Then the problem (4) can be written as

$$\text{find } \mathbf{w} \in \mathbb{R}^n \quad \text{such that} \quad \forall i \in \{1, \ldots, m\} : \forall \theta \in \mathbb{R} : \quad y_i \mathbf{w}^\top \mathbf{X}_i^\top \boldsymbol{\theta} \geq 0\,, \tag{5}$$

[2]We omit an explicit threshold to unclutter the presentation.

which is equivalent to finding a weight vector $\mathbf{w}$ such that the polynomials $p_i(\theta) = y_i \mathbf{w}^\top \mathbf{X}_i^\top \boldsymbol{\theta}$ are non-negative everywhere, i.e., $p_i \in \mathcal{P}_r^+$. The following proposition by Nesterov [5] paves the way for an SDP formulation of the above problem whenever $r = 2l$.

Proposition 1 (SD Representation of Non-Negative Polynomials [5])**.** *The set $\mathcal{P}_{2l}^+$ of polynomials non-negative everywhere on the real line is SD-representable in the sense that*

1. *for every positive semidefinite matrix $\mathbf{P} \succeq \mathbf{0}$ the polynomial $p(\theta) = \boldsymbol{\theta}^\top \mathbf{P} \boldsymbol{\theta}$ is non-negative everywhere, $p \in \mathcal{P}_{2l}^+$.*
2. *for every polynomial $p \in \mathcal{P}_{2l}^+$ there exists a positive semidefinite matrix $\mathbf{P} \succeq \mathbf{0}$ such that $p(\theta) = \boldsymbol{\theta}^\top \mathbf{P} \boldsymbol{\theta}$.*

Proof. Any polynomial $p \in \mathcal{P}_{2l}$ can be written as $p(\theta) = \boldsymbol{\theta}^\top \mathbf{P} \boldsymbol{\theta}$, where $\mathbf{P} = \mathbf{P}^\top \in \mathbb{R}^{(l+1)\times(l+1)}$. (1) $\mathbf{P} \succeq \mathbf{0}$ implies $\forall \theta \in \mathbb{R}: \; p(\theta) = \boldsymbol{\theta}^\top \mathbf{P} \boldsymbol{\theta} = \|\mathbf{P}^{\frac{1}{2}} \boldsymbol{\theta}\|^2 \geq 0$, hence $p \in \mathcal{P}_{2l}^+$. (2) Every non-negative polynomial $p \in \mathcal{P}_{2l}^+$ can be written as a sum of squared polynomials (see, e.g. [4]), hence $\exists q_i \in \mathcal{P}_l^+ : \; p(\theta) = \sum_i q_i^2(\theta) = \boldsymbol{\theta}^\top \left(\sum_i \mathbf{q}_i \mathbf{q}_i^\top\right) \boldsymbol{\theta}$ where $\mathbf{P} := \sum_i \mathbf{q}_i \mathbf{q}_i^\top \succeq \mathbf{0}$ and $\mathbf{q}_i$ is the coefficient vector of polynomial q_i. □

2.2 Maximising Margins on Polynomial Samples

In this subsection we develop an SDP formulation for the problem of learning a maximum margin classifier[3] given the polynomial constraints (5). It is well-known (see, e.g., [9]) that SDPs include quadratic programs as a special case. The squared objective $\|\mathbf{w}\|^2$ is minimised by replacing it with an auxiliary variable t subject to a quadratic constraint $t \geq \|\mathbf{w}\|^2$ that is written as an LMI using Schur's complement lemma,

$$\underset{(\mathbf{w},t)}{\text{minimise}} \quad \frac{1}{2} t \quad \text{subject to} \quad \mathbf{F}(\mathbf{w}, t) := \begin{pmatrix} \mathbf{I}_n & \mathbf{w} \\ \mathbf{w}^\top & t \end{pmatrix} \succeq \mathbf{0},$$

$$\text{and } \forall i: \; \mathbf{G}(\mathbf{w}, \mathbf{X}_i, y_i) := \mathbf{G}_0 + \sum_{j=1}^{n} w_j \mathbf{G}_j \left((\mathbf{X}_i)_{\cdot,j}, y_i \right) \succeq \mathbf{0}. \tag{6}$$

This constitutes an SDP (3) by the fact that a block-diagonal matrix is psd if and only if all its diagonal blocks are psd.

For the sake of illustration consider the case of $l = 0$ (the simplest non-trivial case). The matrix $\mathbf{G}(\mathbf{w}, \mathbf{X}_i, y_i)$ reduces to a scalar $y_i \mathbf{w}^\top \mathbf{x}_i - 1$, which translates into the standard SVM constraint $y_i \mathbf{w}^\top \mathbf{x}_i \geq 1$ linear in $\mathbf{w}$.

For the case $l = 1$ we have $\mathbf{G}(\mathbf{w}, \mathbf{X}_i, y_i) \in \mathbb{R}^{2\times 2}$ and

$$\mathbf{G}(\mathbf{w}, \mathbf{X}_i, y_i) := \begin{pmatrix} y_i \mathbf{w}^\top (\mathbf{X}_i)_{0,\cdot} - 1 & \frac{1}{2} y_i \mathbf{w}^\top (\mathbf{X}_i)_{1,\cdot} \\ \frac{1}{2} y_i \mathbf{w}^\top (\mathbf{X}_i)_{1,\cdot} & y_i \mathbf{w}^\top (\mathbf{X}_i)_{2,\cdot} \end{pmatrix}. \tag{7}$$

Although we require $\mathbf{G}(\mathbf{w}, \mathbf{X}_i, y_i)$ to be psd the resulting optimisation problem can be formulated in terms of a second-order cone program (SOCP) because the matrices involved are only 2×2.[4]

---

[3] In order to emphasise conceptual issues and to simplify the presentation we do not consider the soft-margin case and we omit the bias b.

[4] The characteristic polynomial of a 2×2 matrix is quadratic and has at most two

For the case $l \geq 2$ the resulting program constitutes a genuine SDP. Again for the sake of illustration we consider the case $l = 2$ first. Since a polynomial p of degree four is fully determined by its five coefficients $p_0, \ldots, p_4$, but the symmetric matrix $\mathbf{P} \in \mathbb{R}^{3\times3}$ in $p(\theta) = \boldsymbol{\theta}^\top \mathbf{P}\,\boldsymbol{\theta}$ has six degrees of freedom we require one auxiliary variable u_i per training example,

$$\mathbf{G}(\mathbf{w}, u_i, \mathbf{X}_i, y_i) = \begin{pmatrix} y_i\mathbf{w}^\top (\mathbf{X}_i)_{0,\cdot} - 1 & \frac{1}{2}y_i\mathbf{w}^\top (\mathbf{X}_i)_{1,\cdot} & \frac{y_i\mathbf{w}^\top (\mathbf{X}_i)_{2,\cdot} - u_i}{2} \\ \frac{1}{2}y_i\mathbf{w}^\top (\mathbf{X}_i)_{1,\cdot} & u_i & \frac{1}{2}y_i\mathbf{w}^\top (\mathbf{X}_i)_{3,\cdot} \\ \frac{y_i\mathbf{w}^\top (\mathbf{X}_i)_{2,\cdot} - u_i}{2} & \frac{1}{2}y_i\mathbf{w}^\top (\mathbf{X}_i)_{3,\cdot} & \frac{1}{2}y_i\mathbf{w}^\top (\mathbf{X}_i)_{4,\cdot} \end{pmatrix}.$$

In general, since a polynomial of degree $2l$ has $2l + 1$ coefficients and a symmetric $(l+1)\times(l+1)$ matrix has $(l+1)(l+2)/2$ degrees of freedom we require $(l-1)\,l/2$ auxiliary variables.

2.3 Dual Program and Complementarity

Let us consider the dual SDP corresponding to the optimisation problems above. For the sake of clarity, we restrict the presentation to the case $l = 1$. The dual of the general SDP (3) is given by

$$\underset{\boldsymbol{\Lambda}\in\mathbb{R}^{m\times m}}{\text{maximise}}\ \operatorname{tr}(\mathbf{B}\boldsymbol{\Lambda}) \quad \text{subject to} \quad \forall j \in \{1, \ldots, n\} : \operatorname{tr}(\mathbf{A}_j\boldsymbol{\Lambda}) = c_j;\ \boldsymbol{\Lambda} \succeq \mathbf{0},$$

where we introduced a matrix $\boldsymbol{\Lambda}$ of dual variables. The complementarity conditions for the optimal solution $(\mathbf{w}^*, t^*)$ read $\mathbf{A}((\mathbf{w}^*, t^*))\,\boldsymbol{\Lambda}^* = \mathbf{0}$. The dual formulation of (6) with (7) combined with the $\mathbf{F}(\mathbf{w}, t)$ part of the complementarity conditions reads

$$\underset{(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma})\in\mathbb{R}^{3m}}{\text{maximise}} \quad -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} y_i y_j \left[\tilde{\mathbf{x}}(\alpha_i, \beta_i, \gamma_i, \mathbf{X}_i)\right]^\top \left[\tilde{\mathbf{x}}(\alpha_j, \beta_j, \gamma_j, \mathbf{X}_j)\right] + \sum_{i=1}^{m}\alpha_i$$

$$\text{subject to} \quad \forall i \in \{1, \ldots, m\} : \mathbf{M}_i := \begin{pmatrix} \alpha_i & \beta_i \\ \beta_i & \gamma_i \end{pmatrix} \succeq \mathbf{0}\,, \tag{8}$$

where we define extrapolated training examples $\tilde{\mathbf{x}}(\alpha_i, \beta_i, \gamma_i, \mathbf{X}_i) := \alpha_i(\mathbf{X}_i)_{0,\cdot} + \beta_i(\mathbf{X}_i)_{1,\cdot} + \gamma_i(\mathbf{X}_i)_{2,\cdot}$. As before this program with quadratic objective and psd constraints can be formulated as a standard SDP in the form (3) and is easily solved by a standard SDP *solver*[5]. In addition, the complementarity conditions reveal that the optimal weight vector $\mathbf{w}^*$ can be expanded as

$$\mathbf{w}^* = \sum_{i=1}^{m} y_i \tilde{\mathbf{x}}(\alpha_i, \beta_i, \gamma_i, \mathbf{X}_i)\ , \tag{9}$$

in analogy to the corresponding result for support vector machines [2].

It remains to analyse the complementarity conditions related to the example-related $\mathbf{G}(\mathbf{w}, \mathbf{X}_i, y_i)$ constraints in (6). Using (7) and assuming primal and dual feasibility we obtain for all $i \in \{1, \ldots, m\}$ at the solution $(\mathbf{w}^*, t^*, \mathbf{M}_i^*)$,

$$\mathbf{G}(\mathbf{w}^*, \mathbf{X}_i, y_i) \cdot \mathbf{M}_i^* = \mathbf{0}\,, \tag{10}$$

the trace of which translates into

$$y_i\mathbf{w}^\top \left(\alpha_i^*(\mathbf{X}_i)_{0,\cdot} + \beta_i^*(\mathbf{X}_i)_{1,\cdot} + \gamma_i^*(\mathbf{X}_i)_{2,\cdot}\right) = \alpha_i^*\,. \tag{11}$$

These relations enable us to characterise the solution by the following proposition:

---

solutions. The condition that the lower eigenvalue be non-negative can be expressed as a second-order cone constraint. The SOCP formulation—if applicable—can be solved more efficiently than the SDP formulation.

[5] We used the SDP solver SeDuMi together with the LMI parser Yalmip under Matlab (see also *http://www-user.tu-chemnitz.de/~helmberg/semidef.html*).

Proposition 2 (Sparse Expansion)**.** *The expansion (9) of the optimal weight vector* $\mathbf{w}^*$ *in terms of training examples* $\mathbf{X}_i$ *is sparse in the following sense: Only those examples* $\mathbf{X}_i$ *("support vectors") may have non-zero expansion coefficients* α_i^* *which lie on the margin, i.e., for which* $\mathbf{G}_i(\mathbf{w}^*, \mathbf{X}_i, y_i) \succeq \mathbf{0}$ *rather than* $\mathbf{G}_i(\mathbf{w}^*, \mathbf{X}_i, y_i) \succ \mathbf{0}$*. Furthermore, in this case* $\alpha_i^* = 0$ *implies* $\beta_i^* = \gamma_i^* = 0$ *as well.*

Proof. We assume $\alpha_i^* \neq 0$ and derive a contradiction. From $\mathbf{G}(\mathbf{w}^*, \mathbf{X}_i, y_i) \succ \mathbf{0}$ we conclude using Proposition 1 that for all $\theta \in \mathbb{R}$ we have $y_i \mathbf{w}^{*,\top}((\mathbf{X}_i)_{0,\cdot} + \theta(\mathbf{X}_i)_{1,\cdot} + \theta^2(\mathbf{X}_i)_{2,\cdot}) > 1$. Furthermore, we conclude from (10) that $\det(\mathbf{M}_i^*) = \alpha_i^*\gamma_i^* - \beta_i^{*2} = 0$, which together with the assumption $\alpha_i^* \neq 0$ implies that there exists $\tilde{\theta} \in \mathbb{R}$ such that $\beta_i^* = \tilde{\theta}\alpha_i^*$ and $\gamma_i^* = \beta_i^{*2}/\alpha_i^* = \tilde{\theta}^2\alpha_i^*$. Inserting this into (11) leads to a contradiction, hence $\alpha_i^* = 0$. Then, $\det(\mathbf{M}_i^*) = 0$ implies $\beta_i^* = 0$ and the fact that $\mathbf{G}(\mathbf{w}^*, \mathbf{X}_i, y_i) \succ \mathbf{0} \implies y_i \mathbf{w}^{*,\top}(\mathbf{X}_i)_{2,\cdot} \neq 0$ ensures that $\gamma_i^* = 0$ holds as well. □

The expansion (9) of the weight vector is further characterised by the following proposition:

Proposition 3 (Truly Virtual Support Vectors)**.** *For all examples* $\mathbf{X}_i$ *lying on the margin, i.e., satisfying* $\mathbf{G}(\mathbf{w}^*, \mathbf{X}_i, y_i) \succeq \mathbf{0}$ *and* $\mathbf{M}_i^* \succeq \mathbf{0}$ *there exist* $\theta_i \in \mathbb{R} \cup \{\infty\}$ *such that the optimal weight vector* $\mathbf{w}^*$ *can be written as*

$$\mathbf{w}^* = \sum_{i=1}^{m} \alpha_i^* y_i \tilde{\mathbf{x}}_i(\theta_i) = \sum_{i=1}^{m} y_i \alpha_i^* \left((\mathbf{X}_i)_{0,\cdot} + \theta_i^*(\mathbf{X}_i)_{1,\cdot} + \theta_i^{*2}(\mathbf{X}_i)_{2,\cdot}\right)$$

Proof. (sketch) $\mathbf{M}_i \succeq \mathbf{0}$ implies $\det(\mathbf{M}_i) = \alpha\gamma - \beta^2 = 0$. We only need to consider $\alpha_i^* \neq 0$, in which case there exists θ_i^* such that $\beta_i^* = \theta_i^*\alpha_i^*$ and $\gamma_i^* = \theta_i^{*2}\alpha_i^*$. The other cases are ruled out by the complementarity conditions (10). □

Based on this proposition it is possible not only to identify which examples $\mathbf{X}_i$ are used in the expansion of the optimal weight vector $\mathbf{w}^*$, but also the corresponding value θ_i^* of the transformation parameter θ. This extends the idea of virtual support vectors [7] in that Semidefinite Programming Machines are capable of finding virtual support vectors that were not explicitly provided in the training sample.

3 Extensions to SDPMs

3.1 Optimisation on a Segment

In many applications it may not be desirable to enforce correct classification on the entire trajectory given by the polynomial example $\tilde{\mathbf{x}}(\theta)$. In particular, when the polynomial is used as a local approximation to a global invariance we would like to restrict the example to a segment of the trajectory. To this end consider the following corollary to Proposition 1.

Corollary 1 (SD-Representability on a segment [5])**.** *The set* $\mathcal{P}_l^+(-\tau, \tau)$ *of polynomials non-negative on a segment* $[-\tau, \tau]$ *is SD-representable.*

Proof. (sketch) Consider a polynomial $p \in \mathcal{P}_l^+(-\tau, \tau)$ where $p := x \mapsto \sum_{i=0}^{l} p_i x^i$ and

$$q := x \mapsto \left(1 + x^2\right)^l \cdot \left[p(\tau(2x^2(1+x^2)^{-1} - 1))\right].$$

If $q \in \mathcal{P}_{2l}^+$ is non-negative everywhere then p is non-negative in $[-\tau, \tau]$. □

The proposition shows how we can restrict the examples $\tilde{\mathbf{x}}(\theta)$ to a segment $\theta \in [-\tau, \tau]$ by effectively doubling the degree of the polynomial used. As a matter of fact, this is the SDPM version used in the experiments in Section 4. Note that the matrix $\mathbf{G}(\mathbf{w}, \mathbf{X}_i, y_i)$ is sparse because the resulting polynomial contains only even powers of θ.

3.2 Multiple Transformation Parameters

In practice it would be desirable to treat more than one transformation at once. For example, in handwritten digit recognition transformations like rotation, scaling, translation, shear, thinning/thickening etc. may all be relevant [8]. Unfortunately, Proposition 1 only holds for polynomials in one variable. However, its first statement may be generalised to polynomials of more than one variable: for every psd matrix $\mathbf{P} \succeq \mathbf{0}$ the polynomial $p(\theta) = \mathbf{v}_{\boldsymbol{\theta}}^{\top} \mathbf{P}\, \mathbf{v}_{\boldsymbol{\theta}}$ is non-negative everywhere, even if v_i is any power of θ_j. This means, that optimisation is only over a subset of these polynomials[6]. Considering polynomials of degree two and $\boldsymbol{\theta} := (1, \theta_1, \ldots, \theta_D)$ we have,

$$\tilde{x}_i(\boldsymbol{\theta}) \approx \boldsymbol{\theta}^{\top} \begin{bmatrix} x_i(\mathbf{0}) & \nabla_{\boldsymbol{\theta}}^{\top} x_i(\mathbf{0}) \\ \nabla_{\boldsymbol{\theta}} x_i(\mathbf{0}) & \nabla_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}}^{\top} x_i(\mathbf{0}) \end{bmatrix} \boldsymbol{\theta},$$

where $\nabla_{\boldsymbol{\theta}}^{\top}$ denotes the gradient and $\nabla_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}}^{\top}$ denotes the Hessian operator.

Note that the scaling behaviour with regard to the number D of parameters is more benign than that of the naive method of adding virtual examples to the training sample on a grid. Such a procedure would incur an exponential growth in the number of examples, whereas the approximation above only exhibits a linear growth in the size of the matrices involved.

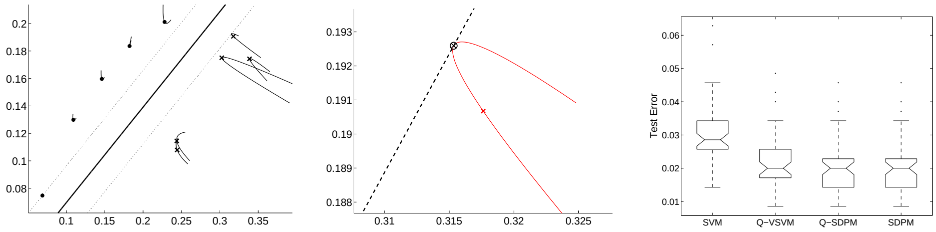
3.3 Learning with Kernels

Support vector machines derive much of their popularity from the flexibility added by the use of kernels [2, 7]. Due to space restrictions we cannot discuss kernels in detail. However, taking the dual SDPM (8) as a starting point and assuming the Taylor expansion (2) the crucial point is that in order to represent the polynomial trajectory in feature space we need to differentiate through the kernel function.

Let us assume a feature map $\boldsymbol{\phi} : \mathbb{R}^n \to \mathcal{F} \subseteq \mathbb{R}^N$ and $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be the kernel function corresponding to $\boldsymbol{\phi}$ in the sense that $\forall \mathbf{x}, \tilde{\mathbf{x}} \in \mathcal{X} : [\boldsymbol{\phi}(\mathbf{x})]^{\top} [\boldsymbol{\phi}(\tilde{\mathbf{x}})] = k(\mathbf{x}, \tilde{\mathbf{x}})$. The Taylor expansion (2) is now carried out in $\mathcal{F}$. Then an inner product expression between data points $\mathbf{x}_i$ and $\mathbf{x}_j$ differentiated, respectively, u and v times reads

$$\underbrace{\left[\boldsymbol{\phi}^{(u)}(\mathbf{x}_i)\right]^{\top} \left[\boldsymbol{\phi}^{(v)}(\mathbf{x}_j)\right]}_{k^{(u,v)}(\mathbf{x}_i, \mathbf{x}_j)} = \sum_{s=1}^{N} \left(\left. \frac{d^u \phi_s(\mathbf{x}(\theta))}{d\theta^u} \right|_{\mathbf{x}=\mathbf{x}_i, \theta=0} \right) \cdot \left(\left. \frac{d^v \phi_s(\tilde{\mathbf{x}}(\tilde{\theta}))}{d\tilde{\theta}^v} \right|_{\tilde{\mathbf{x}}=\mathbf{x}_j, \tilde{\theta}=0} \right).$$

The kernel trick may help avoid the sum over N feature space dimensions, however, it does so at the cost of additional terms by the product rule of differentiation. It turns out that for polynomials of degree $r = 2$ the exact calculation of elements of the kernel matrix is already $\mathcal{O}\left(n^4\right)$ and needs to be approximated efficiently in practice.



(a) (b)

Figure 2: (**a**) A linear classifier learned with the SDPM on 10 2D-representations of the USPS digits "1" and "9" (see Figure 1 for details). Note that the "support" vector is truly virtual since it was never directly supplied to the algorithm (right hand zoom-in). (**b**) Test error distribution of classifiers learned with the SVM, virtual SVM, quadratically approximated SDPM (see text) and SDPM algorithm on 50 independent training sets of size $m = 20$.

4 Experimental Results

In order to test and illustrate the SDPM we used the well-known USPS data set of 16×16 pixel images in $[0, 1]$ of handwritten digits. We considered the transformation *rotation by* θ and calculated the first and second derivatives $\mathbf{x}'_i\,(\theta = 0)$ and $\mathbf{x}''_i\,(\theta = 0)$ based on an image representation smoothed by a Gaussian of variance 0.09.

For the purpose of illustration we calculated two simple features, averaging the first and the second 128 pixel intensities, respectively. Figure 2 (a) shows a plot of 10 training examples of digits "1" and "9" together with the quadratically approximated trajectories for $\theta \in [-20^\circ, 20^\circ]$. The examples are separated by the solution found with an SDPM restricted to the same segment of the trajectory. Following Propositions 2 and 3 the weight vector found is expressed as a linear combination of truly virtual support vectors that had not been supplied in the training sample directly.

In a second experiment, we probed the performance of the SDPM algorithm on the full feature set of 256 pixel intensities using 50 training sets of size $m = 20$ of the digits "1" rotated by -10° and the digits "9" rotated by $+10^\circ$. We compared the performance of the algorithm (measured on an independent test set) to the performance of the original support vector machine (SVM) [2], the virtual support vector machine [7] provided with quadratic approximations to the rotation transformation (Q-VSVM), and a quadratic approximation to the SDPM (Q-SDPM) where we ran the SVM algorithm on $\mathbf{X}_i = (\tilde{\mathbf{x}}_i\,(-10^\circ)\,; \tilde{\mathbf{x}}_i\,(0)\,; \tilde{\mathbf{x}}_i\,(+10^\circ))$. The results are shown in Figure 2 (b). Clearly, taking into account the invariance is useful and the preselection of support vectors by the virtual support vector machine has its price in generalisation performance. The Q-SDPM almost reaches the generalisation performance of the SDPM which is due to the fact that the difference between the convex hull enclosed by the parabola and that of $\mathbf{X}_i$ is negligible for one parameter θ, $D = 1$. It can be expected that for increasing D the performance improvement becomes more pronounced by the effect that in high dimensions most volume is concentrated on the boundary of the convex hull of the polynomial manifold.

[6] There exist polynomials in more than one variable that are non-negative everywhere yet cannot be written as a sum of squares and are hence not SD-representable.

5 Conclusion

We introduced Semidefinite Programming Machines as a means for learning on infinite families of examples given in terms of polynomial trajectories in data space. The crucial insight lies in the SD-representability of non-negative polynomials which allows us to replace the simple non-negativity constraint in algorithms such as support vector machines by positive semidefinite constraints.

While the efficiency of SDP solvers has recently leapt forward due to the application of interior-point methods, it can still hardly be claimed that SDPMs are computationally competitive on real-world tasks. Rather, we would like SDPMs to be seen as a conceptual contribution in the sense that they are able to learn based on the information provided by polynomial examples. In fact, the method of virtual support vectors [7] can be seen as an approximation to the SDPM by linear constraints. On the practical side we note the following: (i) The resulting SDP is well structured (and hence can be solved relatively efficiently) in the sense that $\mathbf{A}(\mathbf{w}, t)$ is block-diagonal with many small blocks, which may often be sparse as well. (ii) It may often be sufficient to satisfy the constraints—e.g., by a version of the perceptron algorithm for semidefinite feasibility problems [3]—without necessarily maximising the margin.

Open questions remain about training SDPMs with multiple parameters (see Subsection 3.2) and about the efficient application of SDPMs with kernels (see Subsection 3.3). Finally, it would be interesting to obtain learning theoretical results regarding the fact that SDPMs effectively make use of an infinite number of (non IID) training examples.

References

[1] O. Chapelle and B. Schölkopf. Incorporating invariances in non-linear support vector machines. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 609–616, Cambridge, MA, 2002. MIT Press.

[2] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[3] T. Graepel, R. Herbrich, A. Kharechko, and J. Shawe-Taylor. Semidefinite programming by perceptron learning. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

[4] A. Nemirovski. Five lectures on modern convex optimization, 2002. Lecture notes of the C.O.R.E. Summer School on Modern Convex Optimization.

[5] Y. Nesterov. Squared functional systems and optimization problems. In H. Frenk, K. Roos, T. Terlaky, and S. Zhang, editors, *High Performance Optimization*, pages 405–440. Kluwer Academic Press, 2000.

[6] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[7] B. Schölkopf. *Support Vector Learning*. R. Oldenbourg Verlag, München, 1997. Doktorarbeit, TU Berlin. Download: http://www.kernel-machines.org.

[8] P. Simard, Y. LeCun, J. Denker, and B. Victorri. Transformation invariance in pattern recognition, tangent distance and tangent propagation. In G. Orr and M. K., editors, *Neural Networks: Tricks of the trade*. Springer, 1998.

[9] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.