Pattern Recognition 43 (2010) 584–591

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.de/locate/pr

Optimal feature selection for support vector machines

Minh Hoai Nguyen *, Fernando de la Torre

Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA

ARTICLE INFO

Article history:
Received 9 February 2009
Received in revised form
17 June 2009
Accepted 1 September 2009

Keywords:
Support vector machine
Feature selection
Feature extraction

ABSTRACT

Selecting relevant features for support vector machine (SVM) classifiers is important for a variety of reasons such as generalization performance, computational efficiency, and feature interpretability. Traditional SVM approaches to feature selection typically extract features and learn SVM parameters independently. *Independently* performing these two steps might result in a loss of information related to the classification process. This paper proposes a *convex* energy-based framework to *jointly* perform feature selection and SVM parameter learning for linear and non-linear kernels. Experiments on various databases show significant reduction of features used while maintaining classification performance.

1. Introduction

Over the last decade, SVMs have become the reference for many classification problems because of their flexibility, computational efficiency and capacity to handle high dimensional data. Like other classifiers, in many supervised learning problems, failure to discard irrelevant features (e.g. noise, outliers, redundant features) will affect the system performance which includes classification accuracy, computational efficiency, and learning convergence. First, the implicit regularization achieved by feature pruning typically increases the generalization ability of classifiers; this generally leads to higher classification accuracy. Second, using irrelevant features also considerably increases the computation time. Third, too many features may render the convergence impossible, leading to random classification decisions. In addition to the system performance, identification of important variables that have intuitive physical interpretation is another critical requirement of many applications. Irrelevant features typically do not have intuitive justification. Due to the aforementioned reasons, feature selection has been a central topic in a variety of fields including signal processing, computer vision, statistics, neural networks, pattern recognition, and machine learning.

Traditionally, feature selection is performed independently of learning the classifier parameters [1–7]. However, separately performing these two steps might result in a loss of information relevant to classification tasks. Recently, several approaches [8–12] for joint feature selection and SVM construction have been proposed. However, the optimization problems of those methods are not convex; the classifier training procedure often converges to a local minimum. Extending previous work on feature selection and classification, this paper proposes a *convex* framework for *jointly* learning optimal feature weights and SVM parameters. We show, theoretically and experimentally, that the set of feature weights obtained by our method is naturally sparse and can be used for feature selection.

Fig. 1 illustrates the main point of the paper. Fig. 1a displays a 17×29 rectangular patch around an eye. Fig. 1b plots the ROC curve to detect eyes in unseen test images, using a linear SVM with all the pixels inside the rectangle (493 pixels) as features. Fig. 1c displays a sparse set of 64 pixels chosen by our algorithm (cyan dots). These pixels and their weights are learned jointly with the SVM parameters. Using only 64 pixels (13% of the features), our SVM classifier produces an ROC curve (Fig. 1d) that is almost identical to the one shown in Fig. 1b (using all pixels). Although the classification performance is not significantly better in this particular case, using only 13% of the features lead to a dramatic increase in speed. Notably, most selected pixels are located around the edges of the eye, which is consistent with our intuition for eye detection.

The rest of the paper is organized as follows. Section 2 reviews previous work on SVMs and feature extraction. Section 3 derives a normalized error function to jointly learn a parameterized kernel and the SVM parameters. Methods for learning feature weights in the input space and kernel space are provided in Sections 4 and 5, respectively. Section 6 relates our method to L_1-SVMs and provides a theoretical justification for the sparsity of the selected features. Section 7 describes extensive experiments on various standard datasets.

* Corresponding author.
E-mail addresses: minhhoai@cmu.edu (M.H. Nguyen), ftorre@cs.cmu.edu (F. de la Torre).
URL: http://www.andrew.cmu.edu/user/minhhoan/ (M.H. Nguyen).

0031-3203/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2009.09.003

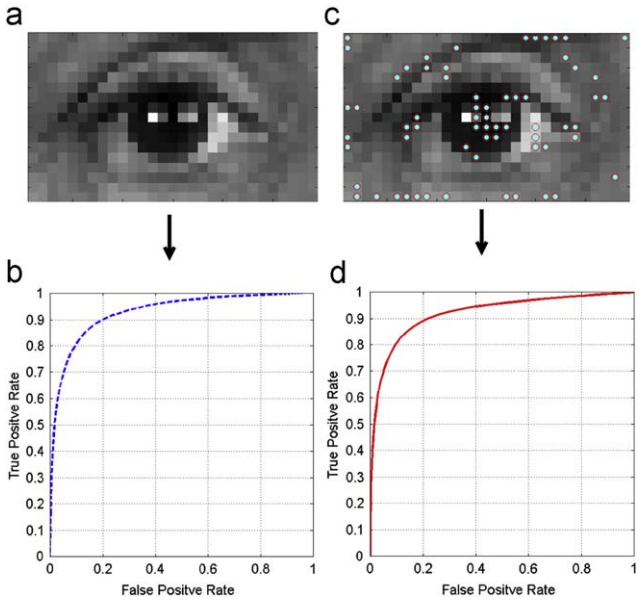

Fig. 1. (a) 17×29 rectangular patch used for eye detection. (b) ROC curve of a linear SVM classifier using all pixels as features. (c) 64 most discriminative pixels used by our SVM classifier that jointly optimizes pixel weighting and SVM parameters. (d) ROC curve of the learned SVM classifier, using only 64 pixels.

2. Previous work

This section reviews previous work on SVMs and feature selection for SVMs.

2.1. Support vector machines

Given a set of training data $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \Re^{d \times 1}$ (see notation)[1] with corresponding labels $y_1, \ldots, y_n \in \{-1, 1\}$, SVMs seek a separating hyperplane with the maximum margin [13]:

$$\begin{aligned} &\underset{\overline{\mathbf{w}},\overline{b},M}{\text{maximize}} \quad M \\ &\text{subject to} \quad y_i(\overline{\mathbf{w}}^T \varphi(\mathbf{x}_i) + \overline{b}) \geq M \quad \forall i, \quad \|\overline{\mathbf{w}}\|_2 = 1. \end{aligned} \tag{1}$$

Here, M is the margin, $\overline{\mathbf{w}}$ is the normal vector of the hyperplane, and $\varphi(\cdot)$ represents the mapping from the input space to the feature space.

Let $\mathbf{w} = \overline{\mathbf{w}}/M$, $b = \overline{b}/M$, then Eq. (1) is equivalent to

$$\underset{\mathbf{w},b}{\text{maximize}} \quad \frac{1}{\|\mathbf{w}\|_2} \quad \text{subject to} \quad y_i(\mathbf{w}^T \varphi(\mathbf{x}_i) + b) \geq 1 \quad \forall i. \tag{2}$$

The above is equivalent to

$$\underset{\mathbf{w},b}{\text{minimize}} \quad \frac{1}{2}\|\mathbf{w}\|_2^2 \quad \text{subject to} \quad y_i(\mathbf{w}^T \varphi(\mathbf{x}_i) + b) \geq 1 \quad \forall i. \tag{3}$$

Using a soft-margin instead of a hard-margin, we obtain the primal problem for SVMs:

$$\begin{aligned} &\underset{\mathbf{w},b,\xi}{\text{minimize}} \quad \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n} \xi_i \\ &\text{subject to} \quad y_i(\mathbf{w}^T \varphi(\mathbf{x}_i) + b) \geq 1 - \xi_i \quad \forall i, \quad \xi_i \geq 0 \quad \forall i. \end{aligned} \tag{4}$$

[1] Bold uppercase letters denote matrices (e.g. $\mathbf{X}$), bold lowercase letters denote column vectors (e.g. $\mathbf{x}$). $\mathbf{x}_i$ represents the ith column of the matrix $\mathbf{X}$. x_{ij} denotes the scalar in the row jth and column ith of the matrix $\mathbf{X}$. x_{ij} also denotes the scalar in the row jth of column vector $\mathbf{x}_i$. Non-bold letters represent scalar variables. $\|\mathbf{x}\|_2 = \sqrt{\mathbf{x}^T\mathbf{x}}$ designates Euclidean norm of $\mathbf{x}$. diag$(\cdot)$ is the operator that extracts the diagonal of a square matrix or constructs a diagonal matrix from a vector.

Here, $\{\xi_i\}_1^n$ are slack variables which allow for penalized constraint violation. C is the parameter controlling the trade-off between a large margin and less constrained violation.

2.2. Feature construction in SVM

This section discusses previous work on selecting features for SVMs. One popular technique for selecting features is RELIEF [14,15]. RELIEF assigns a weight to a particular feature based on the differences between the feature values of nearest neighbor pairs. Cao et al. [4] further developed this method by learning feature weights in kernel spaces. This method is often done as a data processing step, independent of classifier construction. de la Torre and Vinyals [2] learned a subspace-parameterized Taylor series kernel expansion that effectively downweighed irrelevant pixels for classification with SVMs. Recently, there have also been several papers that learn kernel matrices for classification [16–18]. A popular approach is to define a parameterized family of kernel matrices and optimize the kernel parameters to align with an ideal kernel. Another popular approach is to determine a desired property and learn a kernel which exhibits that property. In these approaches, the kernel is learned independently of the SVM parameters. This is a key difference between our proposed method and previous work.

To address the problem of jointly learning SVM parameters and kernels, Chapelle et al. [19] and Weston et al. [9] proposed a method for choosing SVM parameters including the parameters of kernels by minimizing the leave-one-out cross validation (LOOCV) error. However, since the LOOCV error could not be expressed analytically, they instead proposed to minimize some differentiable functions that were upper bounds of the LOOCV error. Mangasarian and Wild [12] introduced a modification to the objective function of SVMs, and performed feature selection by repeatedly sweeping through all features to decide weather to select or deselect a feature depending on which would decrease the value of the objective function.

One way to select a subset of good features is to prune away unnecessary ones. Hermes and Buhmann [8] started by constructing an SVM classifier using all available features and recursively removed the feature that had the least impact on the decision function. Similarly, Avidan [11] used a greedy sequential forward selection method to find a subset of features and support vectors that approximated the SVM solution obtained using all available features.

To further constraint the SVMs' parameters, some authors proposed modifying the objective function of SVMs by including regularization terms or constraints on the parameter $\mathbf{w}$ of SVMs. For example, Chan et al. [3] included two additional constraints on the L_1 and L_2 norms of $\mathbf{w}$ in the formulation of SVMs to achieve a sparse weight vector $\mathbf{w}$. Stoeckel and Fung [20] added a constraint on $\mathbf{w}$ to have the weight for each pixel depend not only on the pixel itself but also on its neighbors. Dundar et al. [21] added a regularization term on $\mathbf{w}$ in the objective function to encourage the decision function to produce similar results for neighboring pixels.

3. SVMs and parameterized kernels

Suppose that the mapping from the input space to the feature space can be parameterized by a parameter vector $\mathbf{p}$, i.e. $\varphi(\mathbf{x}_i) = \varphi(\mathbf{x}_i, \mathbf{p})$. We would like to find a parameter vector $\mathbf{p}$ and a separating hyperplane that have the largest margin. However, different values of $\mathbf{p}$ correspond to different feature spaces, and since the margins in two different feature spaces cannot be directly compared, it is necessary to consider *normalized margins*.

Let us consider the normalized margin as the ratio of the margin over the square root of sum of squared distances (in the feature space) between same-class data instances. In other words, the normalized margin is defined as

$$\frac{M}{\sqrt{\sum_{i,j}\frac{1+y_iy_j}{2}\|\varphi(\mathbf{x}_i,\mathbf{p})-\varphi(\mathbf{x}_j,\mathbf{p})\|_2^2}}. \tag{5}$$

Observe that the normalized margin defined above is invariant to scale and translation in the feature space.

The problem of finding the parameter $\mathbf{p}$ for the mapping and the parameters of the separating hyperplane that provides the largest normalized margin can be stated as

$$\begin{aligned}\underset{\overline{\mathbf{w}},\overline{b},M,\mathbf{p}}{\text{maximize}} \quad & \frac{M}{\sqrt{\sum_{i,j}\frac{1+y_iy_j}{2}\|\varphi(\mathbf{x}_i,\mathbf{p})-\varphi(\mathbf{x}_j,\mathbf{p})\|_2^2}} \\ \text{subject to} \quad & y_i(\overline{\mathbf{w}}^T\varphi(\mathbf{x}_i,\mathbf{p})+\overline{b})\geq M \quad \forall i, \quad \|\overline{\mathbf{w}}\|_2=1.\end{aligned} \tag{6}$$

Recall that if $\mathbf{p}$ is fixed, finding the hyperplane with the maximum normalized margin is equivalent to finding the hyperplane that maximizes the normal margin M.

Let $\mathbf{w}=\overline{\mathbf{w}}/M$, $b=\overline{b}/M$, and let $\phi(\mathbf{p})$ denote $\sum_{i,j}((1+y_iy_j)/2)\|\varphi(\mathbf{x}_i,\mathbf{p})-\varphi(\mathbf{x}_j,\mathbf{p})\|_2^2$, Eq. (6) is equivalent to

$$\begin{aligned}\underset{\mathbf{w},b,M,\mathbf{p}}{\text{maximize}} \quad & \frac{1}{\sqrt{\phi(\mathbf{p})}}\|\mathbf{w}\|_2 \\ \text{subject to} \quad & y_i(\mathbf{w}^T\varphi(\mathbf{x}_i,\mathbf{p})+b)\geq 1 \quad \forall i.\end{aligned} \tag{7}$$

The above is equivalent to

$$\begin{aligned}\underset{\mathbf{w},b,\mathbf{p}}{\text{minimize}} \quad & \frac{1}{2}\phi(\mathbf{p})\|\mathbf{w}\|_2^2 \\ \text{subject to} \quad & y_i(\mathbf{w}^T\varphi(\mathbf{x}_i,\mathbf{p})+b)\geq 1 \quad \forall i.\end{aligned} \tag{8}$$

Using soft-margin instead of hard-margin, we get

$$\begin{aligned}\underset{\mathbf{w},b,\mathbf{p},\xi}{\text{minimize}} \quad & \frac{1}{2}\phi(\mathbf{p})\|\mathbf{w}\|_2^2+C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\varphi(\mathbf{x}_i,\mathbf{p})+b)\geq 1-\xi_i \quad \forall i, \quad \xi_i\geq 0 \quad \forall i.\end{aligned} \tag{9}$$

Here, $\{\xi_i\}_1^n$ are slack variables which allow for penalized constraint violation. C is the parameter controlling the trade-off between having large normalized margin and having less constraint violation.

4. Learning feature weights

Consider a mapping that assigns different weights to different features $\varphi(\mathbf{x}_i,\mathbf{p})=\text{diag}(\mathbf{p})^{1/2}\mathbf{x}_i$, where $\mathbf{p}=[p_1\dots p_d]^T$ are the feature weights, and $p_i\geq 0\ \forall i$. We have

$$\phi(\mathbf{p})=\sum_{k=1}^{d}p_k\sum_{i,j=1}^{n}\frac{1+y_iy_j}{2}(x_{ik}-x_{jk})^2. \tag{10}$$

Since $\phi(\mathbf{p})$ is homogeneous in $\mathbf{p}$, we can always scale $\mathbf{w}$ and $\mathbf{p}$ appropriately to get $\phi(\mathbf{p})=1$. Therefore Eq. (9) is equivalent to

$$\begin{aligned}\underset{\mathbf{w},b,\mathbf{p},\xi}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2+C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\,\text{diag}(\mathbf{p})^{1/2}\mathbf{x}_i+b)\geq 1-\xi_i \quad \forall i, \\ & \sum_{k=1}^{d}p_k\sum_{i,j=1}^{n}\frac{1+y_iy_j}{2}(x_{ik}-x_{jk})^2=1, \\ & \xi_i\geq 0\ \ \forall i, \quad p_k\geq 0\ \ \forall k.\end{aligned} \tag{11}$$

Let $\mathbf{v}=\text{diag}(\mathbf{p})^{1/2}\mathbf{w}$ and consider the function $g:\Re\times\Re^+\to\Re$ defined by

$$g(x,y)=\begin{cases}\dfrac{x^2}{y} & \text{if } y>0,\\ 0 & \text{if } y=0,\ x=0,\\ \infty & \text{if } y=0,\ x\neq 0.\end{cases} \tag{12}$$

Eq. (11) is equivalent to

$$\begin{aligned}\underset{\mathbf{v},b,\mathbf{p},\xi}{\text{minimize}} \quad & \frac{1}{2}\sum_{k=1}^{d}g(v_k,p_k)+C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{v}^T\mathbf{x}_i+b)\geq 1-\xi_i \quad \forall i, \\ & \sum_{k=1}^{d}p_k\sum_{i,j=1}^{n}\frac{1+y_iy_j}{2}(x_{ik}-x_{jk})^2=1, \\ & \xi_i\geq 0\ \ \forall i, \quad p_k\geq 0\ \ \forall k.\end{aligned} \tag{13}$$

Since $g(\cdot,\cdot)$ is convex, the above optimization problem is also convex. It can be optimized using a standard convex optimization package such as CVX [22,23].

5. Feature weighting in feature space

Let $\mathbf{X}\in\Re^{d\times n}$ be the training dataset and $\mathbf{X}'\in\Re^{d\times m}$ be the testing dataset. Let $\varphi(\mathbf{X})$ denote $[\varphi(\mathbf{x}_1)\dots\varphi(\mathbf{x}_n)]$. The training kernel is $\mathbf{K}_{train}=\varphi(\mathbf{X})^T\varphi(\mathbf{X})$, and the testing kernel is $\mathbf{K}_{test}=\varphi(\mathbf{X}')^T\varphi(\mathbf{X})$. Suppose $\mathbf{K}_{train}=\mathbf{USU}^T$ is non-singular. Let $\mathbf{B}=\mathbf{S}^{-\frac{1}{2}}\mathbf{U}^T$, then $\mathbf{B}^T\mathbf{B}=\mathbf{K}_{train}$. Consider the mapping $\tilde{\varphi}:\Re^d\to\Re^n$, $\tilde{\varphi}(\mathbf{x})=\mathbf{B}\varphi(\mathbf{X})^T\varphi(\mathbf{x})$. Based on these conditions, the corresponding train and test kernels are

$$\tilde{\mathbf{K}}_{train}=\tilde{\varphi}(\mathbf{X})^T\tilde{\varphi}(\mathbf{X})=\varphi(\mathbf{X})^T\varphi(\mathbf{X})\mathbf{B}^T\mathbf{B}\varphi(\mathbf{X})^T\varphi(\mathbf{X})=\mathbf{K}_{train}, \tag{14}$$

$$\tilde{\mathbf{K}}_{test}=\tilde{\varphi}(\mathbf{X}')^T\tilde{\varphi}(\mathbf{X})=\varphi(\mathbf{X}')^T\varphi(\mathbf{X})\mathbf{B}^T\mathbf{B}\varphi(\mathbf{X})^T\varphi(\mathbf{X})=\mathbf{K}_{test}. \tag{15}$$

Thus we have defined a feature mapping $\tilde{\varphi}$ that induces the same training and testing kernels. Now, we can learn the feature weights as if the training data was $\mathbf{BK}_{train}$ and the testing data was $\mathbf{BK}_{test}^T$.

If $\mathbf{K}_{train}$ is singular or if we want to reduce the number of dimensions of the feature space, we can take $\mathbf{B}$ as $\mathbf{B}=\mathbf{S}_k^{-1/2}\mathbf{U}_k^T$. Here $\mathbf{U}_k$ contains the first k columns of $\mathbf{U}$ (corresponding to the largest eigenvalues of $\mathbf{K}_{train}$) and $\mathbf{S}_k$ is the sub-matrix of $\mathbf{S}$ containing the first k columns and k rows. In this case, $\tilde{\mathbf{K}}_{train}$ might not exactly match $\mathbf{K}_{train}$, but it is the best *rank-k* approximation.

6. Connection to L_1-SVMs and sparsity

This section discusses the connection between weighted SVMs and L_1-SVMs. First, recall L_1-SVMs minimize the L_1 norm of the weight vector $\mathbf{w}$:

$$\begin{aligned}\underset{\mathbf{w},b,\xi}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|_1+C\sum_i\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i+b)\geq 1-\xi_i \quad \forall i, \quad \xi_i\geq 0 \quad \forall i.\end{aligned} \tag{16}$$

We now show the tight connection between Eq. (13) and (16). Let $a_k=\sum_{i,j}((1+y_iy_j)/2)(x_{ik}-x_{jk})^2$, Eq. (13) becomes

$$\underset{\mathbf{v},b,\mathbf{p},\xi}{\text{minimize}} \quad \frac{1}{2}\sum_{k=1}^{d}g(v_k,p_k)+C\sum_{i=1}^{n}\xi_i \tag{17}$$

$$\text{subject to} \quad y_i(\mathbf{v}^T\mathbf{x}_i + b) \geq 1 - \xi_i \quad \forall i,$$
$$\sum_{k=1}^{d} p_k a_k = 1, \quad \xi_i \geq 0 \forall i, \; p_k \geq 0 \quad \forall k. \tag{18}$$

If there exists k such that $a_k = 0$, then $x_{ik} = x_{jk}$ $\forall i,j$. Thus the kth components of the feature vectors do not affect the classification result; they can be safely removed from the feature vectors. From now on, we assume $a_k > 0$ $\forall k$. We now state a theorem that leads to the connection between weighted SVMs and L_1-SVMs.

Theorem 1. *If* $\mathbf{v}, b, \mathbf{p}, \boldsymbol{\xi}$ *are the parameters that globally minimize Eq.* (17), *then* $\exists \alpha |v_k| = \alpha p_k \sqrt{a_k}$ $\forall k$.

The proof of this theorem is given in Appendix A. We now use this theorem to derive the relation between weighted SVMs and L_1-SVMs. Let α be the scalar such that $|v_k| = \alpha p_k \sqrt{a_k}$ $\forall k$.

$$\Rightarrow \sqrt{a_k}|v_k| = \alpha p_k a_k \quad \forall k,$$
$$\Rightarrow \sum_{k=1}^{d} \sqrt{a_k}|v_k| = \alpha \sum_{k=1}^{d} p_k a_k = \alpha,$$
$$\Rightarrow \alpha = \sum_{k=1}^{d} \sqrt{a_k}|v_k|. \tag{19}$$

On the other hand,

$$\sum_{k=1}^{d} g(v_k, p_k) = \sum_{k=1}^{d} \frac{v_k^2}{p_k} = \sum_{k=1}^{d} \frac{\alpha^2 p_k^2 a_k}{p_k} = \alpha^2 \sum_{k=1}^{d} p_k a_k. \tag{20}$$

From Eqs. (18)–(20), we have

$$\sum_{k=1}^{d} g(v_k, p_k) = \left(\sum_{k=1}^{d} \sqrt{a_k}|v_k| \right)^2. \tag{21}$$

Furthermore, if $p_k = |v_k|/\alpha\sqrt{a_k}$, then

$$\sum_{k=1}^{d} p_k a_k = \sum_{k=1}^{d} \frac{|v_k|}{\alpha\sqrt{a_k}} a_k = \sum_{k=1}^{d} \frac{\sqrt{a_k}|v_k|}{\alpha} = \frac{1}{\alpha} \sum_{k=1}^{d} \sqrt{a_k}|v_k|, = \frac{1}{\alpha}\alpha = 1. \tag{22}$$

Therefore, the constraint (18) can be dropped from the optimization problem (17). This observation, in addition with Eq. (21), show that Eq. (17) is equivalent to

$$\underset{\mathbf{v},b,\xi}{\text{minimize}} \quad \frac{1}{2}\left(\sum_{k=1}^{d} \sqrt{a_k}|v_k| \right)^2 + C \sum_{i=1}^{n} \xi_i$$
$$\text{subject to} \quad y_i(\mathbf{v}^T\mathbf{x}_i + b) \geq 1 - \xi_i \quad \forall i, \quad \xi_i \geq 0 \quad \forall i. \tag{23}$$

Eq. (23) is not exactly the same with the formulation of L_1-SVMs (Eq. (16)). However, there is a tight connection between the two. Eq. (23) minimizes a squared weighted L_1-norm of the weight vector $\mathbf{v}$ while L_1-SVMs minimizes L_1-norm of the weight vector. In fact, for every positive C, there exists a positive number $\tilde{C}$ such that Eq. (23) and the following optimization problem have the same optimal solution (reach their optimum values at the same set of variables $(\mathbf{v}, b, \boldsymbol{\xi})$):

$$\underset{\mathbf{v},b,\xi}{\text{minimize}} \quad \frac{1}{2} \sum_{k=1}^{d} \sqrt{a_k}|v_k| + \tilde{C} \sum_{i=1}^{n} \xi_i$$
$$\text{subject to} \quad y_i(\mathbf{v}^T\mathbf{x}_i + b) \geq 1 - \xi_i \quad \forall i, \quad \xi_i \geq 0 \quad \forall i. \tag{24}$$

This equivalence follows directly from Theorem 2 which is stated and proved in Appendix B.

Because weighted SVM (Eq. (13)) is equivalent to Eq. (23) which shares the same set of optimal variables as Eq. (24), one can expect the weight vector learned by weighted SVM will be sparse. Furthermore, though both Eqs. (13) and (24) are convex, the later can be optimized far more efficiently using an appropriate linear programming.

7. Experiments

This section compares weighted SVMs with state-of-the-art feature selection methods on MNIST, a large scale dataset. Weighted SVMs are also compared with normal SVMs on three image databases and three others from the UCI machine learning repository [24].

7.1. Handwritten digit recognition

In this section, we describe experiments on MNIST [25], a large-scale, publicly available dataset.[2] This data collection contains 28×28 images of handwritten digits. To focus on binary classification, we consider the task of distinguishing between the two most frequently confused digits: four and nine. Each digit comes with disjoint training and testing subsets that contain roughly 6000 and 1000 data samples, respectively. The provided training set was randomly split into two halves; one was used for training the classifier and the other one was used for tuning the parameters.

Very little was done for data preprocessing. Images of handwritten digits are first vectorized into 784×1 column vectors. Attribute values were rescaled to be in the range between 0 and 1 by dividing by 255. To further test the ability of our method and competing algorithms for selecting relevant features, each data sample was padded with 200 additional random features drawn from the standard normal distribution.

We compared our method with Hermes and Buhmann's [8], RELIEF [14,15], and Iterative RELIEF (I-RELIEF) [5,26] which are three state-of-the-art feature selection methods. While our method returns a sparse set of features and their associated weights, the other methods return a ranking of features. For a fair comparison, the same number of features is used for all methods.

Table 1 shows the accuracy of the evaluating methods performed on the testing set. It also displays the number of support vectors for the methods which use an SVM classifier. Unlike our method, RELIEF and I-RELIEF solely perform feature selection. The outputs of these methods need to be fetched into a subsequent classifier. In Table 1, RELIEF+SVM and I-RELIEF+SVM denote the method that uses RELIEF and I-RELIEF for feature selection and SVM for classification. Similarly, I-RELIEF+kNN indicates the method that combines I-RELIEF and k nearest neighbors (kNNs).

As can be seen from Table 1, the performance of weighted SVM is no worse than the performance of linear and Gaussian SVMs while using only 10% of the features. Weighted SVM also matches the performance of Hermes and Buhmann's method. Compared with RELIEF and I-RELIEF, our method produce a significantly better set of features for SVMs. It is worth to note that our method has the least number of support vectors compared with the other competing methods.

As expected, a two-stage approach for feature selection and classification is not always ideal; I-RELIEF works well when combined with a kNN but not with an SVM. Here, I-RELIEF+kNN is included as a reference rather than a competing method since we are more interested in feature selection for SVMs. In many situations, an SVM is the classifier of choice due to various

[2] http://www.cs.toronto.edu/~roweis/data.html

Table 1
Performance of different methods on MNIST dataset.

Method	Accuracy	# Features	# SVs
Linear SVM	96.23	984	711
Gaussian SVM	96.53	984	1646
Hermes and Buhmann	96.63	91	913
RELIEF+SVM	94.73	91	914
I-RELIEF+SVM	93.32	91	929
Weighted SVM (ours)	96.53	91	557
I-RELIEF+kNN	96.79	91	N/A

Linear SVM and Gaussian SVM are constructed using the full set of features. Hermes and Buhmann, RELIEF+SVM, and I-RELIEF+SVM are competing feature selection methods for SVMs. I-RELIEF+kNN does not use SVM; it is included as a reference rather than a competing method. As can be seen, our method works as well as Hermes and Buhmann and it outperforms RELIEF+SVM and I-RELIEF+SVM.

Fig. 2. Examples of faces from the CMU face database.

reasons. For example, compared with a kNN, an SVM is much faster during testing.

7.2. Pose classification

We performed experiments on the CMU face images dataset from the UCI machine learning repository [24]. The database contains 30×32 pixel facial images of 20 people under different expressions and poses. Some examples of faces from the database are given in Fig. 2. The classification task was to distinguish between two different poses: looking up and looking to the camera. Because the number of data instances in this database is small (only 312 faces), the experimental results were taken as the accuracy of 10-fold cross validation. We constructed four different SVM classifiers, namely linear SVM, linear weighted SVM, Gaussian SVM, and Gaussian weighted SVM. For all classifiers, we repeated the experiments for different values of the C parameter (and γ for Gaussian SVMs) and reported the best results. Table 2 shows the best results from all methods. Notably, weighted SVMs achieve similar classification accuracy while using a much smaller number of pixels and support vectors.

7.3. Eye detection

Following the approach of Everingham and Zisserman [27], we performed eye detection experiments on the gray-scale FERET database [28]. This database contains facial images of various subjects under different expressions and poses. All images have a 256×384 pixel resolution and limited lighting variation. Some images are associated with a set of four hand labeled landmarks (Fig. 3a). Among the images with labeled landmarks, we extracted all the 2963 available frontal faces for experiments. These images

Table 2
Comparison of weighted SVMs and normal SVMs on the UCI CMU face images dataset.

	Linear		Gaussian kernel	
	Normal	Weighted	Normal	Weighted
10-fold CV acc.	95.5	95.5	97.48	98.06
# Features used	960	67	312	74
# SVs	102	85	186	73

The weighted SVMs (both linear and Gaussian) achieve similar accuracy rates while using much fewer features and support vectors.

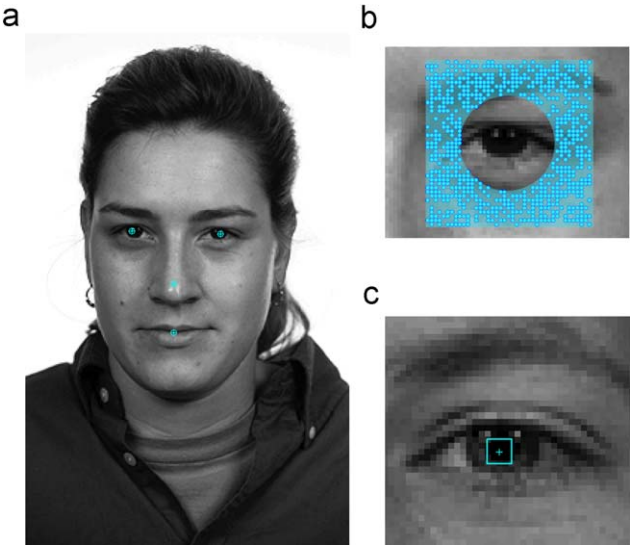

Fig. 3. (a) Example of four landmarks used in the FERET database. (b) Centers of negative training patches were sampled randomly inside the cyan region. (c) Region of correct classification, positively classified pixels were considered correct if they are located inside the square.

were further divided into disjointed training and testing sets (60% and 40%, respectively).

For training, we first performed Procrustes analysis [29] to align the landmarks w.r.t. the mean shape, removing rotation, translation, and scale variation. Positive training examples were obtained by sub-sampling 17×29 patches inside 27×47 rectangular regions around the left iris landmark of every training image. Similarly, negative examples were created by extracting rectangular patches around random points in an iris neighborhood. The neighborhood was defined as in Fig. 3b. Each patch was normalized by subtracting the mean intensity and dividing by the standard deviation.

For each training image, the OpenCV Viola-Jones face detector [30] was used to produce a square centered on the face. A linear regression predictor was trained to approximate the iris landmark from the position and scale of the face detector's output [27].

We performed experiments with two different SVM classifiers, namely normal SVM and weighted SVM. For weighted SVM, we first applied the method described in Section 4 to learn the optimal pixel weights. Pixels with insignificant weights ($< 10^{-5}$) were discarded, and an SVM classifier was constructed based on the remaining pixels, taking their weights into account. Fig. 1c shows the locations of 64 pixels (out of 493) chosen by our weighted SVM (cyan dots).

For each testing image, we used the previously learned linear regression to produce the first approximation for the iris' position.

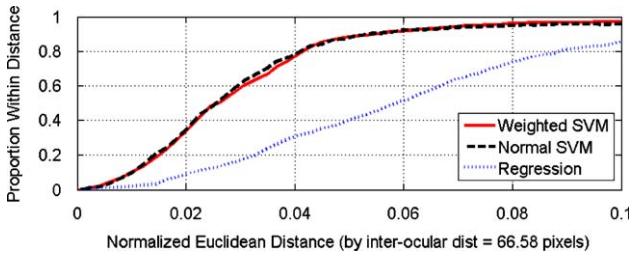

Fig. 4. Distance threshold versus the proportion of iris localization within the threshold. The distance is taken as the Euclidean distance from the ground truth landmark to the predicted iris location normalized by the inter-ocular distance. Weighted SVM performs as well as the other method while using a much smaller number of pixels. The *regression* curve is the result of using initial guess produced by the linear regression predictor.

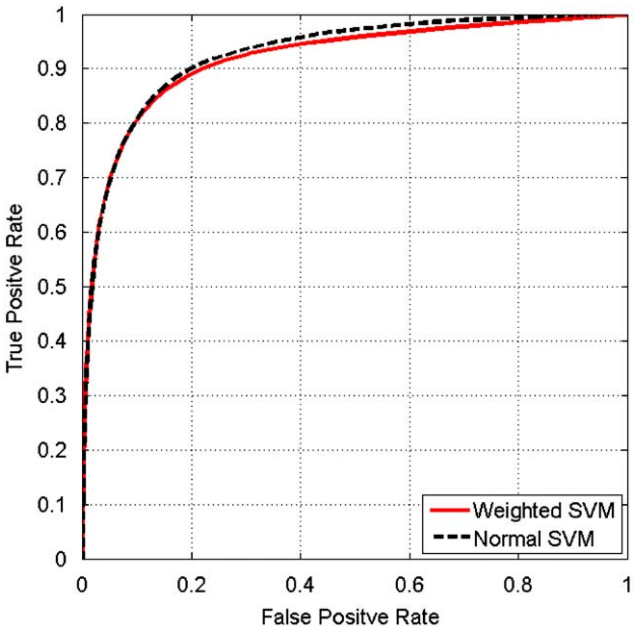

Fig. 5. ROC curves of two different methods. Weighted SVM performs as well as normal SVM while using a much smaller number of pixels.

A searching window was placed around this initial guess. With a sliding window approach, the pixel with the highest SVM decision value was chosen as the final result for the localization of the iris.

The performance of different algorithms was evaluated in two different ways. Fig. 4 plots the localization error threshold (x-axis) and the proportion of successful localizations within the threshold (y-axis). The Euclidean distance from the ground truth landmark to the predicted iris location was normalized by the inter-ocular distance (distance between the two iris landmarks) to account for different scales. Compared with normal SVM, weighted SVM achieves similar performance results while using a much smaller number of pixels.

To analyze the trade-off between true detections and false alarms, we classified all pixels inside the searching window and produced ROC curves (Fig. 5) by varying the threshold of the SVM classifier. The positively classified pixels were considered correct if they fell inside a square neighborhood around the true landmark. The size of this neighborhood was proportional to the inter-ocular distance of the subject (illustrated in Fig. 3c). As can be observed, the ROC curve produced by our weighted SVM is similar to the one produced by standard SVM. However, weighted SVM used only 13% of available pixels. In our experiments, C and other parameters of SVMs were tuned using cross validation.

Table 3
Comparison of weighted SVM and normal SVM on several standard datasets.

Dataset	Accuracy rate		# Features used		# SVs	
	Normal	Weighted	Normal	Weighted	Normal	Weighted
Ionosphere	86.89	88.50	34	22	79	65
Breast cancer	95.43	96.31	30	30	40	28
USPS	97.75	97.5	256	36	50	40
Pedestrian	72.83	74.9	648	147	1291	1159

The weighted SVM achieves similar or better accuracy rates while using smaller numbers of features and support vectors.

7.4. Experiments on other datasets

We performed experiments on several other standard datasets. The ionosphere and breast cancer are downloaded from the UCI machine learning repository [24]. Because the numbers of data instances in these datasets are small, the experimental results are taken as the accuracy of 10-fold cross validation. For both normal SVM and weighted SVM, we repeat the experiments for different values of the C parameter and report the best results. The USPS and pedestrian [31] are two image datasets for digit classification and pedestrian detection, respectively. These datasets contain enough instances so we divide them into disjoint subsets for training, validation, and testing. The validation sets are used to tune the C parameters of SVMs. We use the same C for learning the weights and for testing the performance of weighted SVMs.

The results are summarized in Table 3. The table shows that our method achieves similar accuracy rates compared with the normal SVM. However, the number of active features used by our proposed method is fewer than the number of the active features used by normal SVMs. Furthermore, the number of support vectors used by our proposed model is always smaller than the number of support vectors used in standard SVMs. In many problem domains, this could greatly speed up the application run time.

7.5. Software packages and training time

In our experiments, we used several publicly available software packages. For I-RELIEF, we used the implementation of *mlPy*,[3] a python library for machine learning. For RELIEF, we used the implementation bundled with *Weka* [32], a java-based machine learning toolkit.[4] Normal SVM classifiers were built using *LibSVM* [33]. To jointly optimize feature weights and construct weighted SVMs, we used *CVX*, a package for specifying and solving convex programs [22,23].

As proved in Section 6, our method can employ either (13) and (24) because optimizing them lead to equivalent solutions. However, for a generic convex program solver like CVX, it was actually more efficient to optimize (13) than to optimize (24). Because of this empirical reason, all of our experiments produce solutions by optimizing (13). Nevertheless, it is important to note that our method is potentially very efficient if an appropriate linear programming algorithm is used to optimize (24).

All experiments were performed on a MacOS machine with a 2.4 GHz Intel Core 2 Duo chipset and 4 GB of RAM. The main programming language is MATLAB. The training time of each method depends on the dataset, the number of training samples, and the parameter setting. For the experiment on the MNIST database (6000 training samples), the training phase of our

[3] https://mlpy.fbk.eu/
[4] http://www.cs.waikato.ac.nz/~ml/weka/index.html

method took 1 h 20 min. To produce ranking for features, RELIEF and I-RELIEF took 1 h 40 min and 14 h, respectively. Hermes and Buhmann's method required 6 min training time. These amounts of training time are not directly comparable since we used different software packages written in various programming languages. Hermes and Buhmann's method is much faster than the other methods because it is based on LibSVM [33], a highly customized optimization package.

8. Conclusion

In this paper, we have presented a convex method for jointly learning feature weights and parameters of SVM classifiers. Moreover, we related the proposed framework with L_1-SVMs and provided and theoretical justification for its use as a feature selection method. Experiments on seven standard datasets and different classification problems showed that our method produced SVM classifiers that used sparse sets of features and support vectors while retaining classification performance.

Acknowledgments

This work was supported by the US Naval Research Laboratory under Contract no. N00173-07-C-2040. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the US Naval Research Laboratory. We would like to thank Joan Perez for helping us with the experiment in Section 7.3.

Appendix A. Proof of Theorem 1

In this section, we provide a proof for Theorem 1 given in Section 6. We first prove that $v_j^2/a_j p_j^2 = v_l^2/a_l p_l^2$ for all pairs of j,l such that $p_j>0, p_l>0$.

Assume the contrary, $\exists j \neq l : v_j^2/a_j p_j^2 \neq v_l^2/a_l p_l^2$. Consider the function $f(\cdot)$ defined as follows:

$$f(\varepsilon) = g\left(v_j, p_j + \frac{\varepsilon}{a_j}\right) + g\left(v_l, p_l - \frac{\varepsilon}{a_l}\right). \tag{25}$$

The derivative of $f(\varepsilon)$ w.r.t. ε is

$$\frac{\partial f(\varepsilon)}{\partial \varepsilon} = \frac{\partial g\left(v_j, p_j + \frac{\varepsilon}{a_j}\right)}{\partial \varepsilon} + \frac{\partial g\left(v_l, p_l - \frac{\varepsilon}{a_l}\right)}{\partial \varepsilon}. \tag{26}$$

We have

$$\frac{\partial g\left(v_j, p_j + \frac{\varepsilon}{a_j}\right)}{\partial \varepsilon} = \frac{\partial \frac{v_j^2}{p_j + \frac{\varepsilon}{a_j}}}{\partial \varepsilon} = -\frac{v_j^2}{a_j\left(p_j + \frac{\varepsilon}{a_j}\right)^2}. \tag{27}$$

Similarly,

$$\frac{\partial g\left(v_l, p_l + \frac{\varepsilon}{a_l}\right)}{\partial \varepsilon} = \frac{\partial \frac{v_l^2}{p_l - \frac{\varepsilon}{a_l}}}{\partial \varepsilon} = \frac{v_l^2}{a_l\left(p_l - \frac{\varepsilon}{a_l}\right)^2}. \tag{28}$$

From Eqs. (26)–(28), we have

$$\left.\frac{\partial f(\varepsilon)}{\partial \varepsilon}\right|_{\varepsilon=0} = -\frac{v_j^2}{a_j p_j^2} + \frac{v_l^2}{a_l p_l^2}. \tag{29}$$

From the assumption that $v_j^2/a_j p_j^2 \neq v_l^2/a_l p_l^2$, we have $\partial f(\varepsilon)/\partial \varepsilon|_{\varepsilon=0} \neq 0$. Thus $\varepsilon = 0$ is not a local minimum of $f(\varepsilon)$. Therefore, together with the fact that $p_j, p_l > 0$, we can always find $\hat{\varepsilon}$ such that $f(\hat{\varepsilon}) < f(0)$, $p_j + \hat{\varepsilon}/a_j \geq 0$, and $p_l - \hat{\varepsilon}/a_l \geq 0$. Consider a new set of feature weights $\tilde{\mathbf{p}}$: $\tilde{p}_j = p_j + \hat{\varepsilon}/a_j$, $\tilde{p}_l = p_l - \hat{\varepsilon}/a_l$, and $\tilde{p}_i = p_i \ \forall i \neq j,l$. One can easily verify that $\mathbf{v}, b, \tilde{\mathbf{p}}, \boldsymbol{\xi}$ is the set of parameters that yield lower objective value for Eq. (17) while satisfying all the constraints. This contradicts with the assumption that $\mathbf{v}, b, \mathbf{p}, \boldsymbol{\xi}$ are the parameters that globally minimize Eq. (17). Hence, we must have $v_j^2/a_j p_j^2 = v_l^2/a_l p_l^2$ for all pairs of j,l such that $p_j > 0, p_l > 0$. Thus $v_k^2/a_k p_k^2 = const \ \forall k : p_k > 0$. Equivalently, $\exists \alpha : |v_k| = \alpha p_k \sqrt{a_k} \ \forall k : p_k > 0$. Furthermore, for $k : p_k = 0$, we must have $v_k = 0$. This means that $|v_k| = \alpha p_k \sqrt{a_k}$ for $k : p_k = 0$. Thus, $\exists \alpha : |v_k| = \alpha p_k \sqrt{a_k} \ \forall k$. The proof of the above theorem is complete.

Appendix B. Theorem 2

In this section, we state and prove a theorem that is used in Section 6.

Theorem 2. *Consider the following optimization problem*:

$$\text{minimize} \quad f(\mathbf{x})^2 + Cg(\mathbf{x}) \tag{30}$$

$$\text{subject to} \quad h_i(\mathbf{x}) \leq 0 \quad \forall i = \overline{1,k}, \tag{31}$$

in which $f(\mathbf{x}), g(\mathbf{x}), h_1(\mathbf{x}), \ldots, h_k(\mathbf{x})$ are multivariate convex functions and $f(\mathbf{x})$, $g(\mathbf{x})$ are one-sided directional differentiable at all points satisfying the constraints (31). *Let $\mathbf{x}^*$ be an optimal solution of this optimization problem and suppose $f(\mathbf{x}^*) > 0$. If $\tilde{C} = C/2f(\mathbf{x}^*)$ then $\mathbf{x}^*$ is also an optimal solution of the following optimization problem*:

$$\text{minimize} \quad f(\mathbf{x}) + \tilde{C}g(\mathbf{x}) \text{ subject to} \quad h_i(\mathbf{x}) \leq 0 \quad \forall i = \overline{1,k}. \tag{32}$$

Proof. Let $\tilde{\mathbf{x}}$ be an optimal solution of (32) and suppose $\mathbf{x}^*$ is not an optimal solution of (32). Thus,

$$f(\mathbf{x}^*) + \tilde{C}g(\mathbf{x}^*) > f(\tilde{\mathbf{x}}) + \tilde{C}g(\tilde{\mathbf{x}}). \tag{33}$$

Consider the directional vector $\mathbf{v} = \tilde{\mathbf{x}} - \mathbf{x}^*$. Because $h_1(\mathbf{x}), \ldots, h_k(\mathbf{x})$ are convex and both $\mathbf{x}^*$ and $\tilde{\mathbf{x}}$ satisfy all the constraints, all points between these two ends must also satisfied the constraints. Consider two one-dimensional functions: $F(\varepsilon) = f(\mathbf{x}^* + \varepsilon\mathbf{v})^2 + Cg(\mathbf{x}^* + \varepsilon\mathbf{v})$ and $\tilde{F}(\varepsilon) = f(\mathbf{x}^* + \varepsilon\mathbf{v}) + \tilde{C}g(\mathbf{x}^* + \varepsilon\mathbf{v})$ for $0 \leq \varepsilon \leq 1$. Because $f(\mathbf{x})$ and $g(\mathbf{x})$ are both convex, $\tilde{F}(\varepsilon)$ must also be convex. Consider the right derivative at $\varepsilon = 0$:

$$\left.\frac{\partial_+ F(\varepsilon)}{\partial \varepsilon}\right|_{\varepsilon=0} = 2f(\mathbf{x}^*)\left.\frac{\partial_+ f(\mathbf{x}^* + \varepsilon\mathbf{v})}{\partial \varepsilon}\right|_{\varepsilon=0} + C\left.\frac{\partial_+ g(\mathbf{x}^* + \varepsilon\mathbf{v})}{\partial \varepsilon}\right|_{\varepsilon=0} \tag{34}$$

$$= 2f(\mathbf{x}^*)\left(\left.\frac{\partial_+ f(\mathbf{x}^* + \varepsilon\mathbf{v})}{\partial \varepsilon}\right|_{\varepsilon=0} + \tilde{C}\left.\frac{\partial_+ g(\mathbf{x}^* + \varepsilon\mathbf{v})}{\partial \varepsilon}\right|_{\varepsilon=0}\right) \tag{35}$$

$$= 2f(\mathbf{x}^*)\left.\frac{\partial_+ \tilde{F}(\varepsilon)}{\partial \varepsilon}\right|_{\varepsilon=0}. \tag{36}$$

On the other hand, from (33) we have $\tilde{F}(0) > \tilde{F}(1)$. This, together with the convexity of $\tilde{F}(\varepsilon)$, leads to $\partial_+ \tilde{F}(\varepsilon)/\partial\varepsilon|_{\varepsilon=0} < 0$. Therefore, $\partial_+ F(\varepsilon)/\partial\varepsilon|_{\varepsilon=0} < 0$ (from Eq. (36)). Hence, $\varepsilon = 0$ is the not an optimal minimizer of $F(\varepsilon)$. However, this leads to a contradiction because $\mathbf{x} = \mathbf{x}^*$ is an optimal minimizer of $f(\mathbf{x})^2 + Cg(\mathbf{x})$. Thus the assumption that $\mathbf{x}^*$ is not an optimal solution of $f(\mathbf{x}) + \tilde{C}g(\mathbf{x})$ cannot hold. This completes the proof of Theorem 2. □

References

[1] B. Moghaddam, Y. Weiss, S. Avidan, Fast pixel/part selection with sparse eigenvectors, in: Proceedings of the IEEE International Conference on Computer Vision, 2007.

[2] F. de la Torre, O. Vinyals, Learning kernel expansions for image classification, in: IEEE Conference on Computer Vision and Pattern Recognition, 2007.

[3] A.B. Chan, N. Vasconcelos, G.R.G. Lanckriet, Direct convex relaxations of sparse SVM, in: Proceedings of International Conference on Machine Learning, 2007.

[4] B. Cao, D. Shen, J.-T. Sun, Q. Yang, Z. Chen, Feature selection in a kernel space, in: Proceedings of International Conference on Machine Learning, 2007.

[5] Y. Sun, J. Li, Iterative RELIEF for feature weighting, in: Proceedings of International Conference on Machine Learning, 2006.

[6] I. Kononenko, Estimating attributes: analysis and extensions of RELIEF, in: Proceedings of European Conference on Computer Vision, 1994.

[7] P.S. Bradley, O.L. Mangasarian, Feature selection via concave minimization and support vector machines, in: Proceedings of International Conference on Machine Learning, 1998.

[8] L. Hermes, J. Buhmann, Feature selection for support vector machines, in: Proceedings of International Conference on Pattern Recognition, 2000.

[9] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, V. Vapnik, Feature selection for SVMs, in: Advances in Neural Information Processing Systems, 2001.

[10] S. Avidan, Subset selection for efficient SVM tracking, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2003.

[11] S. Avidan, Joint feature-basis subset selection, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2004.

[12] O.L. Mangasarian, E.W. Wild, Feature selection for nonlinear kernel support vector machines, in: Workshop on Optimization-based Data Mining Techniques with Applications, IEEE International Conference on Data Mining, 2007.

[13] B. Schölkopf, A. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, MIT Press, Cambridge, MA, 2002.

[14] K. Kira, L. Rendell, The feature selection problem: traditional methods and new algorithm, in: Proceedings of AAAI Conference on Artificial Intelligence, 1992.

[15] K. Kira, L. Rendell, A practical approach to feature selection, in: Proceedings of International Workshop on Machine Learning, 1992.

[16] S.C.H. Hoi, M.R. Lyu, E.Y. Chang, Learning the unified kernel machines for classification, in: Proceedings of International Conference on Knowledge Discovery and Data Mining, 2006.

[17] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, J. Kandola, On kernel-target alignment, in: Advances in Neural Information Processing Systems, 2001.

[18] G.R. Lanckriet, N. Cristianini, P. Bartlett, L.E. Ghaoui, M.I. Jordan, Learning the kernel matrix with semidefinite programming, Journal of Machine Learning Research 5 (2004) 27–72.

[19] O. Chapelle, V. Vapnik, O. Bousquet, S. Mukherjee, Choosing multiple parameters for support vector machines, Machine Learning 46 (1–3) (2002) 131–159.

[20] J. Stoeckel, G. Fung, SVM feature selection for classification of SPECT images of Alzheimer's disease using spatial information, in: IEEE International Conference on Data Mining, 2005.

[21] M. Dundar, J. Theiler, S. Perkins, Incorporating spatial contiguity into the design of a support vector machine classifier, in: IEEE I nternational Conference on Geoscience and Remote Sensing Symposium, 2006.

[22] M. Grant, S. Boyd, CVX: Matlab software for disciplined convex programming (web page & software), October 2008 〈http://stanford.edu/~boyd/cvx〉.

[23] M. Grant, S. Boyd, Graph implementations for nonsmooth convex programs, in: V. Blondel, S. Boyd, H. Kimura (Eds.), Recent Advances in Learning and Control (a tribute to M. Vidyasagar) Lecture Notes in Control and Information Sciences, Springer, Berlin, 2008, pp. 95–110.

[24] A. Asuncion, D. Newman, UCI machine learning repository, 2007 〈http://www.ics.uci.edu/~mlearn/MLRepository.html〉.

[25] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[26] Y. Sun, Iterative relief for feature weighting: algorithms, theories, and applications, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (6) (2007) 1035–1051.

[27] M. Everingham, A. Zisserman, Regression and classification approaches to eye localization in face images, in: Proceedings of the Seventh International Conference on Automatic Face and Gesture Recognition, 2006.

[28] P. Philips, H. Moon, S. Rizvi, P. Rauss, The FERET evaluation methodology for face-recognition, Pattern Analysis and Machine Intelligence 22 (10) (2000) 1090–1104.

[29] T. Cootes, C. Taylor, Statistical models of appearance for computer vision, Technical Report, University of Manchester, 2001.

[30] P. Viola, M. Jones, Robust real-time face detection, International Journal of Computer Vision 57 (2) (2004) 137–154.

[31] S. Munder, D. Gavrila, An experimental study on pedestrian classification, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (11) (2006) 1863–1868.

[32] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, second ed., Morgan Kaufmann, San Francisco, 2005.

[33] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, 2001, software available at 〈http://www.csie.ntu.edu.tw/~cjlin/libsvm〉.

About the Author—MINH HOAI NGUYEN received his M.S. in Robotics from Carnegie Mellon University and B.E. in Software Engineering from the University of New South Wales — Australia in 2009 and 2005, respectively. Currently, he is a 4th year Ph.D. student at the Robotics Institute, Carnegie Mellon University. His research work focuses on learning optimal data representations for image classification, clustering, visualization, and modeling.

About the Author—FERNANDO DE LA TORRE received his Ph.D. in E.E. in 2002 from La Salle School of Engineering, Barcelona. Since 2005 he is an Associate Research Professor at the Robotics Institute, Carnegie Mellon University. Currently, he is directing the Component Analysis Lab (http://ca.cs.cmu.edu), and the Human Sensing Lab (http://humansensing.cs.cmu.edu) at CMU.