Maximum-Margin Feature Combination for Detection and Categorization

Gökhan H. Bakır, Mingrui Wu and Jan Eichhorn
Max Planck Institute for Biological Cybernetics
Spemannstraße 38, 72076 Tübingen, Germany
{first.last}@tuebingen.mpg.de

Abstract

In this paper we are concerned with the optimal *combination* of features of possibly different types for detection and estimation tasks in machine vision. We propose to combine features such that the resulting classifier maximizes the margin between classes. In contrast to existing approaches which are non-convex and/or generative we propose to use a discriminative model leading to convex problem formulation and complexity control. Furthermore we assert that decision functions should not compare apples and oranges by comparing features of different types directly. Instead we propose to combine different similarity measures for each different feature type. Furthermore we argue that the question: "Which feature type is more discriminative for task X?" is ill-posed and show empirically that the answer to this question might depend on the complexity of the decision function.

1 Introduction

The typical workflow in most detection and estimation tasks in machine vision is to first agree upon a set of features and then in a second *independent* step to use some decision technique to construct a model using these features. Often, however one faces the problem that there is no unique set of features for a given task, but that there are possibly multiple cues providing information about the task at hand. For example, in [5] various methods for categorization based on Color, Texture and Shape were investigated and it was concluded that for multi-class object categorization multiple features and different combinations of features are needed. In this paper we achieve this by using the *multi-kernel learning* (MKL) framework[1, 8], see Figure 1.

We propose that the separate steps of choosing features and constructing the prediction model should be combined into a single step. We would like to use a single algorithm to construct a decision function and at the same time choose the best combination of features. In addition, we investigate if the choice of feature and the complexity of the decision function are dependent. By controlling the tradeoff between training error and complexity of the final decision function, we will show that the choice of features does depend on the complexity of the decision function. This leads to our conclusion that the question of whether feature A is better than feature B for a specific task can not be answered without specifying the complexity of the involved decision function.

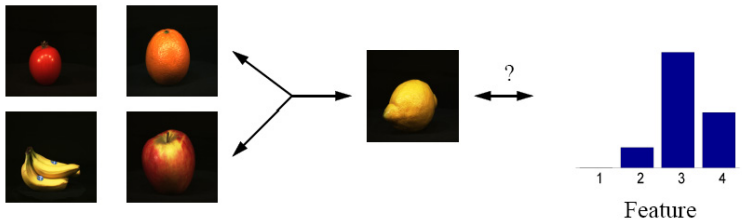


Figure 1: What is an optimal combination of Shape Statistics, Local image descriptors, Color histograms and Texture filters to discriminate lemon from other fruits? MKL constructs a possibly sparse combination of arbitrary feature types such that the margin of the resulting classifier is maximized. The resulting weights can be interpreted as relevance. Images taken from the ALLOI database[4].

The paper is organized as follows. In section 2 we review the relevant work for feature combination. Thereafter, in section 3 we introduce our used MKL algorithms. All our introduced algorithms are convex and thus all the determined feature weightings are obtained as the global solution to a convex optimization problem. In section 4 we demonstrate the MKL algorithms in face detection and object categorization tasks on some standard machine vision benchmark data. In section 5 we summarize our results and conclude.

2 Background and Relevant Work

For further discussions, let us denote by $f \in F$ a feature f belonging to the type F. By a feature f we mean the instance of an observed entity in the image and by the feature type we mean the nature of this entity. For example color is a feature type and red is a feature. Typically one observes multiple features $\{f_1^i, \ldots, f_{r_i}^i\}$ per feature type F_i, where r_i denotes the number of observed feature instances per type F_i. For example if we consider for F_1 local image descriptors and for F_2 color histograms, then typically r_1 is in the order of hundreds to thousands while r_2 is one per image.

Usually, using all possibly available types of features is not necessary and thus one prefers a sparse, task relevant subset of feature types due to computational performance reasons. In general a favored subset of all feature types is chosen and then they have to be combined and presented to a learning algorithm. Let us review in the following how features of different type are combined.

2.1 Feature Combination for Detection and Categorization

In the following we describe three types of existing feature combination strategies: Feature Stacking, Feature Dictionaries and Probabilistic Approaches.

Feature Stacking The most naive way of combining different types of features is to simply stack all features $\{f_j^i\}_{j=1}^{r_i}$ of each type F_i into a single big feature vector $x = [f_1^1, \ldots, f_{r_1}^1, \ldots, f_{r_m}^m] \in \bigoplus_i^m F_i^{r_i}$. The stacked vector can contain features from different cues like color histograms, responses from Gabor filters up to $2\frac{1}{2}D$ range data. Due to the nature of such stacked feature vectors, all components will carry a different amount of information that is relevant for the task at hand. Using then the stacked feature vectors x_i with corresponding class label y_i one can use any standard supervised learning technique for detection and categorization.

Unfortunately feature stacking is not always straight-forward. Consider the case that the observed number of instances of one feature type varies while another type generates always constant amount of features. For example the number of most kind of local image descriptors or shape-context descriptors vary whereas

the number of global features in an image like intensity histogram stays constant. Thus often stacking requires extra processing. Another issue with feature stacking is that learning algorithms which perform non-linear transformations, such as support vector machines with a non-linear kernel function, will lead to a *mixing* of the entries and thus of the different types. This implies that the information on which feature is essentially used might get lost and the result is not interpretable anymore. To this end, if feature stacking is used one has to use all possible available feature types since one does not know which feature type is essential in advance.

Feature Dictionaries An alternative approach is not to construct a single big feature vector x but to use a fixed dictionary D with $|D| = d$ overall number of a-priori fixed features $\{f_j^i\}_{j=1}^{r_i} \in F_i$ with $\sum_i^m r_i = d$. Current approaches in vision using feature dictionaries are mostly using a boosting training strategy [9, 11]. A quite convenient property of boosting-based techniques is that the model remains interpretable, since not necessarily all features in the dictionary will be selected for the task at hand. On the other hand, a disadvantage of such approaches is that they rely on early-stopping to control generalization behavior and thus might need plenty of manual tuning. See [2, 7] for discussion.

Probabilistic Approach A quite different approach to feature combination is to consider feature types as probabilistic information sources with a given prior belief function $p(f|\theta_i)$. In this case prediction can be done by Bayesian inference yielding the prediction equation

$$c(z) = \int_y p(y|z, \theta_0) \prod_{i=1}^{m} p(z|f_i) p(f_i|\theta_i) dy.$$

In this probabilistic setting the learning algorithm is concerned in finding good model parameters $\boldsymbol{\theta}$ given some training data. Note that the parameters θ_i which control the relevance of feature types are now parameters of probability distributions. These so called hyper-parameters, can be found e.g. by maximizing the conditional posterior distribution function

$$\theta_i^* = \arg\max_{\theta_i} p(\theta_i|y, z, f_1, \ldots, f_l, \theta_0) \tag{1}$$

From a mathematical perspective, we believe that this is the most rigorous approach of all. Unfortunately, from a practical viewpoint this seems to be an inconvenient approach since (1) often turns out to be a non-convex optimization problem. Standard approaches are based on Markov Chain Monte Carlo sampling or sequential approximations.

Thus all existing approaches so far are hard to optimize, not interpretable or not generally applicable to various kind of feature types. In the next section we describe how to overcome these limitations using kernels.

3 Combining features: The kernel approach

In kernel methods usually the *raw* input features are first mapped into a reproducing kernel hilbert space(RKHS) $\mathcal{F}$ via some nonlinear mapping $\phi : F \to \mathcal{F}$. Let us denote by $\mathcal{F}_i = \Phi_i(F_i)$ the RKHS associated with feature type F_i. The linear dot-product in this new feature space $\mathcal{F}_i$ is then typically evaluated using a kernel function $k_i : F_i \times F_i \to \mathbb{R}$ which can be interpreted as a similarity measure between two features $f_u^i, f_v^i \in F_i$. In this paper we follow the assert that features of different type are combinable but not comparable. To this end, we use a different similarity measure for each type of feature F_i and thus have a kernel function for each type of feature. A natural way to combine features of different type is then to use stacking

in the space $\bigoplus_{i=1}^{m} \mathcal{F}_i$ which results in the similarity measure

$$k(x,z) = \sum_{i=1}^{m} \beta_i k_i(x,z). \tag{2}$$

Here x, z are sets of features of arbitrary type. The only assumption is that both sets have at least one instance for each possible type of features. The task of the learning algorithm is now to choose in addition the weights β. This is the concern of the MKL algorithm which considers decision functions of the form

$$c(x) = \sum_{i=1}^{l} \alpha_i k(x_i, x) + b,$$

where the kernel function k is given as positive weighted sum as in (2). Thus the goal of MKL is to find the $l+m$ variables $\{\alpha, \beta\}$ according to the learning task at hand. The i-th entry of the α vector controls the participation of data point x_i in the decision function whereas the j-th entry of the β vector can be interpreted as the *relevance* of feature type F_j. A zero weight β_j would correspond to *blend out* the feature space associated with the corresponding kernels similar as in feature selection. Let us now review the detailed formulations of our used MKL algorithm.

3.1 MKL for detection and categorization

MKL was first explored in [12] for classification, regression and density estimation hence it was shown not to be limited to classification. However, in the following we will constrain ourselves on binary classification. We consider a linear model in the feature space $\mathcal{F} = \bigoplus_{i=1}^{m} \mathcal{F}_i$ and our aim is to build a maximum margin classifier given a set of training data $D_n = \{x_i, y_i\}_{i=1}^{n}$ where x_i is a set of features and y_i is a class label. Thus our decision function $c(z)$ takes now the form $c(z) = w^\top \phi(z) + b$. This can be formulated as follows

$$\begin{aligned} \underset{w_i,\xi,b}{\arg\min} \quad & \tfrac{1}{2}\Omega[w_1,\ldots,w_m] + C\textstyle\sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & y_i(\textstyle\sum_{j=1}^{m} w_j^\top \phi_j(x_i) + b) \geq 1 - \xi_i, \quad 1 \leq i \leq n. \end{aligned}$$

where the weight vector of the classifier w is stacked as $w = [w_1, \ldots, w_m]$. Note that ϕ_j acts only on the elements of x that are of type F_j. The regularizing functional $\Omega[w_1, \ldots, w_m]$ measures the norm of the resulting classifier and can be chosen in different ways, e.g. $\Omega_2[w_1, \ldots, w_m] = \sum_{i=1}^{m} ||w_i||^2$ would lead to standard SVM formulation. In [1] it was proposed to use $\Omega_1[w_1, \ldots, w_m] = (\sum_{i=1}^{m} ||w_i||)^2$, penalizing the one norm on the block level of w and thus favoring a sparse choice of subspaces $\Phi_i(F_i)$. The dual derived in [1] was shown to be a convex second order cone problem and therefore has quadratic constraints. Unfortunately, such problems are not easy to solve. This was the motivation for [8] to reformulate the dual in [1] by replacing the quadratic constraint by infinitely many linear constraints yielding a semi-infinite linear constrained problem:

$$\begin{aligned} \underset{\theta\in\mathbb{R},\quad \beta\in\mathbb{R}^m}{\arg\max} \quad & \theta \\ \text{subject to} \quad & \textstyle\sum_{j=1}^{m} \beta_j \left(\frac{1}{2}\alpha^\top H_j \alpha - \alpha^\top \mathbf{1}_n\right) \geq \theta \quad \text{for any } \alpha \in [0,C]^n \\ \text{and} \quad & \beta^\top \mathbf{1}_n = 1, \quad \alpha^\top y = 0, \quad \beta \geq 0 \end{aligned} \tag{3}$$

where $(H_j)_{rs} = y_r y_s k_j(x_r, x_s)$. Note that this optimization problem has a global solution. This reformulation has the advantage that it can be efficiently solved using the exchange algorithm for semi-infinite problems. This results in an alternating sequence of standard linear programs and SVM optimizations. Due to space limits we will not go into the details but give a description of the algorithm in figure 3.1. For categorization we use the one-vs-all classification scheme which reduced the multi-class problem to a pairwise binary problem.

Figure 3.1: Maximum Margin Feature Combination — The MKL algorithm.

Initialize $t = 1, \quad \beta_j = \frac{1}{m}, \quad 1 \leq j \leq m.$
Repeat until convergence
Train SVM to obtain $\alpha^t \in [0, C]^n$ with kernel function $k(x, z) = \sum \beta_i^t k_i(x, z)$.
Calculate SVM objective for each kernel corresponding to each one of the m feature types: $D_j^t = \alpha^\top H_j \alpha - \alpha^\top \mathbf{1}_n$, $1 \leq j \leq m$
Solve linear program (3) with the t linear constraints $\{D^1, \ldots, D^t\}$ and retrieve new β^{t+1} and margin θ.
$t = t + 1$

4 Experiments

In this section we demonstrate the application of MKL to several standard benchmark datasets used for categorization and detection in machine vision. We start with a face detection experiment, where our aim is to explore the combination of *spatial* features. Afterwards we investigate the combination of features of very different nature: color histograms and local image descriptors. In this case feature stacking in input space is not applicable and MKL provides an easy way to combine these cues.

4.1 Face Detection

The first experiment is conducted over the MIT CBCL Face Data Set which comes with an a-priori split into training and test set. The data set consists of 6977 cropped training images (2429 faces and 4548 non-faces) and 24045 cropped test images (472 faces and 23573 non-faces). As a training set we used a randomly selected subset of 1000 face images and 1000 non-face images. Testing is performed on the whole test set. All images are rescaled to 15 by 15 pixels and processed with histogram equalization.

According to the fragment idea from [10], it is reasonable to assume that different local regions in an image might be of different relevance for face detection. The MKL algorithm can provide a principled way to combine these fragments.

In this experiment we divide each image into 9 non-overlapping patches of size 5×5. Each patch is considered as a different feature type. The kernel function k_i $(1 \leq i \leq 9)$ in (2) between a pair of images is then simply defined as the Gaussian kernel restricted to the i-th patch. Note that for simplicity we only use non-overlapping patches, but it is straightforward to apply MKL to use arbitrary, possibly overlapping patches.

To further investigate the feature combination ability of the MKL approach, we provide additional types of features. To this end, we calculate *edge maps* obtained by convolving each image with horizontal and vertical Sobel filters. Then as for the raw images, each of the two obtained edge maps is divided into 9 patches. So now there are overall 27 patches for each image: 9 patches from the raw image and 9 patches from the horizontal and vertical edge maps each. As before the function k_i $(1 \leq i \leq 27)$ in (2) between a pair of images is defined as the Gaussian kernel function restricted to the i-th feature type.

In contrast to MKL current SVM approaches adopt the simple feature stacking scheme. In this case the kernel function between a pair of images is computed on the full images.

The comparison of the experimental results of the MKL algorithm and SVM approach are presented in Figure 2. As can be seen in the ROC curves in Figure 2 adding additional feature types can improve the classification accuracy of both MKL and SVM. But clearly MKL outperforms the SVM approach: the result of SVM approach *with* edge maps is even worse than the result of MKL algorithm *without* edge maps. A reason for this is that the SVM approach simply performs feature stacking and thus suffer from typical problems as described in section 2.1. On the contrary, MKL tries to handle different feature types separately and puts different weights to different feature types.

The column (c) in the left part of Figure 2 shows weight maps illustrating the weights for different patches. It can be seen that when only the raw image is used, the central patch

corresponding to the nose gets the highest weight. When edge maps are used, the central patch in the vertical edge map gets the highest weight.

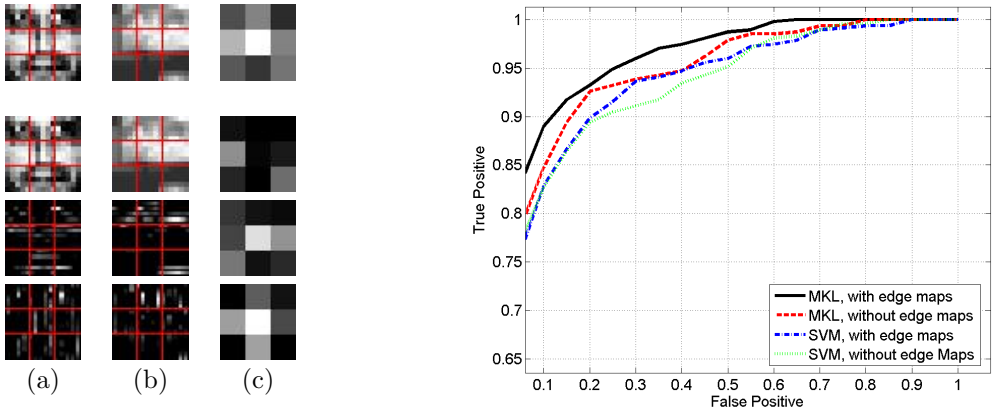


Figure 2: Left: (a) An example face image and its edge maps (b) An example non-face image and its edge maps (c) Weight maps, patches with lighter intensities have larger weight values. The first weight map is obtained without edge maps, the last three are the weight maps obtained with both raw images and edge maps. Right: the ROC curves of face detection on CBCL database.

4.2 Functional

The second experiment is a categorization task, which is performed on the ETH80-database [5]. The feature types used for this task are SIFT-descriptors [6] and color histograms. Due to the nature of local image descriptors (LID) (e.g. SIFT) and their dependency on salient points in the image, this description normally results in a set of a varying number of descriptor vectors. Therefore combining these two feature types in a stacking approach is hardly applicable and might require preprocessing. In contrast, MKL provides a natural approach to combine LIDs with color histograms via combining the kernel functions.

The database contains images of 80 individual objects grouped into 8 equally sized classes. We use a subset of 5 views of each object which leaves us with a total number of $5 \times 80 = 400$ images. Images come with a segmentation mask which allows masking out the background for color histogram computation. All experiments are performed in a one-vs-rest multi-class setting.

Color histograms are built via segmenting the three-dimensional color-space into 16 bins in each dimension and then computing a histogram over the image pixels while masking out the background. As kernel function for this representation we applied the χ^2-kernel which is based on the χ^2-distance: $k_{\chi^2}(f, f') = \exp(-d_{\chi^2}(f, f'))$ where $d_{\chi^2}(f, f') = \sum_i \left((f_i - f'_i)^2\right) / (f_i + f'_i)$.

SIFT-features are computed from local regions around interest points which are identified by an interest point detector (IPD) in a first step. Basically, the descriptor vectors themselves contain smoothed angular histograms of gradients from the region of interest. As IPD we apply a Harris corner detector. Note that all found SIFT-features in an images describes a set and therefore we use the Bhattacharyya-kernel for sets. For details on how to use the Bhattacharyya kernel see [3]. Therefore, in our experiment one feature type is a histogram and the other one is a set.

As already pointed out in [5] the performance of various feature-types may depend on the particular structure of the objects to be categorized. For the images of ETH80 it is very likely, that e.g. the tomato and the cup class will be very well discriminated by color features. In contrast, using LIDs for tomatoes is probably not such a good idea since there is not much structure in the object and only a few salient points are detected. Consequently, the combination weights of the MKL solution should reflect this property.

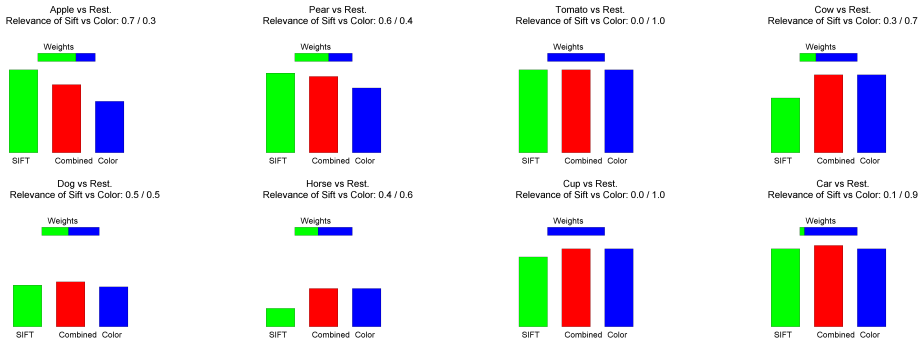
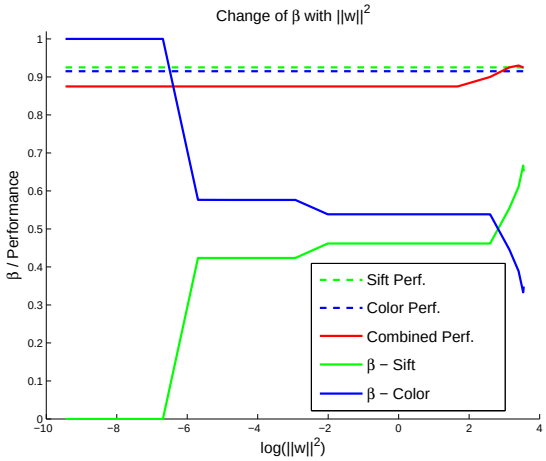


Figure 3: Weightings of the two feature types (horizontal bar) and leave-one-out performance of the combined kernel compared to the individual ones (vertical bars).



Overall Performance:

Combined Kernel	85.5%
Sift	76.5%
Color Histogram	77.0%

Figure 4: Left: Change of the weightings when varying the complexity of the classifier. Right: Mean performance over all classes.

Figure (3) shows the weights of all classes for the two feature types in a horizontal bar. The vertical bars indicate the performance of the combined kernel (middle) vs. the performance of the individual kernels. The performance was computed with a leave-one-out scheme where all views of a particular object were held out at the same time. In Figure (3) you can see e.g. in the third column (tomatoes and cups) that, in agreement with our expectations results in [5], the color histogram feature type gets all the weight due to the above mentioned reasons.

The aim of the second experiment is to investigate any dependency between the relevance of a feature and the complexity of the decision function. Therefore, we varied the regularization parameter C of the SVM that is inverse proportional to the margin of the resulting classifier and thus, using a nonlinear kernel, is related to the smoothness of the decision boundary. We recorded the resulting combination weights β_i and the test performance. In Figure (4) these weights are plotted against the complexity of the resulting classifier measured by the squared norm: $\|w\|^2$. Note that the combination weights vary dramatically with complexity. For small $\|w\|^2$ (i.e. low complexity) the color feature type has all the mass. This means that with low complexity classifiers (i.e. smooth boundaries) color is the best feature type for discrimination. On the other hand, when more complex boundaries are allowed (larger $\|w\|^2$) SIFT features become more relevant and finally lead to an increase in performance.

5 Discussion and Conclusion

In this paper we have shown that multi kernel learning is an appropriate framework to combine different feature types for detection and categorization. The three benefits from this approach are: a) No a-priori selection of feature types is necessary. b) The combination might lead to an improved performance. c) The final obtained relevance factors may give insights on the task. The used approach is based on a convex optimization problem, thus ensuring a globally optimal solution.

Furthermore we have discussed that the complexity of the decision function is a crucial factor in choosing the right feature type for a particular task. Therefore the question: "Which feature type is more discriminative for task X?" can not be answered without any knowledge about the finally used classifier.

References

[1] F.R. Bach, Lanckriet G.R.G, and M.I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004.

[2] Harris Drucker. Effect of pruning and early stopping on performance of a boosting ensemble. *Comput. Stat. Data Anal.*, 38(4):393–406, 2002.

[3] Jan Eichhorn and Olivier Chapelle. Object categorization with svm: kernels for local features. Technical Report 137, Max-Planck Institute for biological Cybernetics, 2004.

[4] Jan-Mark Geusebroek, Gertjan J. Burghouts, and Arnold W.M. Smeulders. The amsterdam library of object images. *International Journal of Computer Vision*, 61(1), 2005.

[5] B. Leibe and B. Schiele. Analyzing contour and appearance based methods for object categorization. In *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR'03)*, 2003.

[6] David G. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, pages 1150–1157, September 1999.

[7] G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, March 2001.

[8] S. Sonnenburg, G. Rtsch, and C. Schfer. Learning interpretable svms for biological sequence classification. In *RECOMB 2005*, pages 389–407. Springer-Verlag Berlin Heidelberg, 2005.

[9] Antonio Torralba, Kevin P. Murphy, and William T. Freeman. Sharing visual features for multiclass and multiview object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Washington, DC, 2004.

[10] S. Ullman, M. Vidal-Naquet, and E Sali. Visual features of intermediate complexity and their use in classification. *Nature Neuroscience*, 5:682–, 2002.

[11] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 511–518, 2001.

[12] J. Weston, A. Gammerman, M. O. Stitson, V. Vapnik, V. Vovk, and C. Watkins. Support vector density estimation. Technical Report CSD-TR-97-23, Royal Holloway, 1997.