Classification using Intersection Kernel Support Vector Machines is Efficient *

Subhransu Maji
EECS Department
U.C. Berkeley
smaji@eecs.berkeley.edu

Alexander C. Berg
Yahoo! Research
aberg@yahoo-inc.com

Jitendra Malik
EECS Department
U.C. Berkeley
malik@eecs.berkeley.edu

Abstract

Straightforward classification using kernelized SVMs requires evaluating the kernel for a test vector and each of the support vectors. For a class of kernels we show that one can do this much more efficiently. In particular we show that one can build histogram intersection kernel SVMs (IKSVMs) with runtime complexity of the classifier ***logarithmic*** *in the number of support vectors as opposed to* ***linear*** *for the standard approach. We further show that by precomputing auxiliary tables we can construct an approximate classifier with* ***constant*** *runtime and space requirements, independent of the number of support vectors, with negligible loss in classification accuracy on various tasks. This approximation also applies to* $1 - \chi^2$ *and other kernels of similar form.*

We also introduce novel features based on a multi-level histograms of oriented edge energy and present experiments on various detection datasets. On the INRIA pedestrian dataset an approximate IKSVM classifier based on these features has the current best performance, with a miss rate 13% *lower at* 10^{-6} *False Positive Per Window than the linear SVM detector of Dalal & Triggs. On the Daimler Chrysler pedestrian dataset IKSVM gives comparable accuracy to the best results (based on quadratic SVM), while being* $15\times$ *faster. In these experiments our approximate IKSVM is up to* ***2000****$\times$ faster than a standard implementation and requires* ***200****$\times$ less memory. Finally we show that a* $50\times$ *speedup is possible using approximate IKSVM based on spatial pyramid features on the Caltech 101 dataset with negligible loss of accuracy.*

1. Introduction

Discriminative classification using Support Vector Machines (SVMs) and variants of boosted decision trees are two of the leading techniques used in vision tasks ranging from detection of objects in images like faces [18, 27], pedestrians [7, 16] and cars [20], multicategory object recognition in Caltech-101 [1, 15], to texture discrimination [30]. Part of the appeal for SVMs is that non-linear decision boundaries can be learnt using the so called 'kernel trick'. Though SVMs have faster training speed, the runtime complexity of a non linear SVM classifier is high. Boosted decision trees on the other hand have faster classification speed but are significantly slower to train and the complexity of training can grow exponentially with the number of classes [25]. Thus, linear kernel SVMs have become popular for real-time applications as they enjoy both faster training and classification speeds, with significantly less memory requirements than non-linear kernels due to the compact representation of the decision function.

Discriminative approaches to recognition problems often depend on comparing distributions of features, *e.g*. a kernelized SVM, where the kernel measures the similarity between histograms describing the features. In order to evaluate the classification function, a test histogram is compared to histograms representing each of the support vectors. This paper presents a technique to greatly speed up that process for histogram comparison functions of a certain form – basically where the comparison is a linear combination of functions of each coordinate of the histogram.

This more efficient implementation makes a class of kernelized SVMs used in many of the current most successful object detection/recognition algorithms efficient enough to apply much more broadly, even possibly to realtime applications. The class of kernels includes the pyramid matching or intersection kernels used in Grauman & Darell [11] ; and Lazebnik *et al*. [15], and the chi squared kernel used by Bosch *et al*. [1]; Varma & Ray [26]; and Chum & Zisserman [5], which together represent some of the best results on the Caltech and Pascal VOC datasets. In addition the decision functions learned are generalizations of linear decision functions and offer better classification performance at small additional computation cost. This allows improvements in performance of object detection systems based on linear classifiers such as those used by Dalal & Triggs [7]

*This work is funded by ARO MURI W911NF-06-1-0076 and ONR MURI N00014-06-1-0734

978-1-4244-2243-2/08/$25.00 ©2008 IEEE

and Felzenszwalb *et al*. [9]. These ideas will be illustrated with the histogram intersection kernel, and generalizations will be introduced in Section 3.3.

The histogram intersection kernel [24], $k_{HI}(h_a, h_b) = \sum_{i=1}^{n} \min(h_a(i), h_b(i))$ is often used as a measurement of similarity between histograms h_a and h_b, and because it is positive definite [17] it can be used as a kernel for discriminative classification using SVMs. Recently, intersection kernel SVMs (henceforth referred to as IKSVMs), have been shown to be successful for detection and recognition, *e.g*. pyramid match kernel [11] and spatial pyramid matching [15]. Unfortunately this success typically comes at great computational expense compared to simpler linear SVMs, because non-linear kernels require memory and computation linearly proportional to the number of support vectors for classification. [1].

Given feature vectors of dimension n and learned support vector classifier consisting of m support vectors, the time complexity for classification and space complexity for storing the support vectors of a standard IKSVM classifier is $\mathcal{O}(mn)$. We present an algorithm for IKSVM classification with time complexity $\mathcal{O}(n \log m)$ and space complexity $\mathcal{O}(mn)$. We then present an approximation scheme whose time and space complexity is $\mathcal{O}(n)$, independent of the number of support vectors. The key idea is that for a class of kernels including the intersection kernel, the classifier can be decomposed as a sum of functions, one for each histogram bin, each of which can be efficiently computed. In various experiments with thousands of support vectors we observe speedups and space savings up to **2000**$\times$ and **200**$\times$ respectively, compared to a standard implementation. On Caltech-101, one-vs-all classifiers on an average have 175 support vectors and the approximate IKSVM classifier is about 50x faster.

The rest of the paper is structured as follows: In Section 2 we describe the background on state of the art in improving support vector classification speeds. In Section 3 we describe the key insight of the algorithm and present various approximations. In Section 4 we present our multi-level histogram of oriented gradients based feature and experimental results on on various datasets. Conclusions are in Section 5.

2. Background

Due to the immense practical importance of SVM based classifiers, there has been a fair amount of research on reducing their run time complexity. The complexity of classification for a SVM using a non linear kernel is the number of support vectors $\times$ the complexity of evaluating the kernel function (see equation 4). The later is generally an increasing function of the dimension of the feature vectors. There are two general approaches to speeding up classification: reducing the number of support vectors by constructing sparse representations of the classifier, or reducing the dimension of the features using a coarse to fine approach. We discuss each of these briefly.

A class of methods start with the SVM classifier and construct sparse representations using a small subset of the support vectors [3, 19]. Instead of having a single approximation, one can have a series of approximations with more and more support vectors to obtain a cascade of classifiers, an idea which has been used in [22] to build fast face detectors. Another class of methods build classifiers by having a regularizer in the optimization function which encourages sparseness, (*e.g*. L_1*-norm* on the alphas) or pick support vectors in a greedy manner till a stopping criteria is met [14]. These methods achieve accuracies comparable to the full classifier using only a small fraction of the support vectors on various UCI-Datasets. However, our technique makes these methods irrelevant for intersection kernels as we have an exact method which has a logarithmic dependence on the number of support vectors.

The coarse to fine approach for speeding up the classification is popular in many realtime applications. The idea is to use simpler features and classifiers which are used to reject easy examples quickly in a cascade. This idea has been applied, to face detection [12] and recently to pedestrian detection [28] achieve a speedup of 6-8 over a non cascaded version of the detector. These kinds of speedups are orthogonal to the speedups we achieve using our methods. Thus our technique together with a coarse to fine approach may lead to even greater speedups.

There is not much literature on quick evaluation of the decision function [2] and the approach closest in spirit to our work is the work by Yang *et al*. [29] who use the Fast Gauss Transform to build efficient classifiers based on the Gaussian kernel. However their method is approximate and is suitable only when the dimension of the features is small.

3. Algorithm

We first begin with a review of support vector machines for classification. Given labeled training data of the form $\{(y_i, \mathbf{x}_i)\}_{i=1}^{N}$, with $y_i \in \{-1, +1\}$, $\mathbf{x}_i \in \mathcal{R}^n$, we use a C-SVM formulation [6]. For a kernel on data points, $k(\mathbf{x}, \mathbf{z}) : \mathcal{R}^n \times \mathcal{R}^n \rightarrow \mathcal{R}$, that is the inner product, $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{z})$, in an unrealized, possibly high dimensional, feature space, the algorithm finds a hyperplane which best separates the data by minimizing:

$$\tau(\mathbf{w}, \xi) = \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=i}^{N} \xi_i \qquad (1)$$

[1] It has also been shown recently that the number of support vectors grow linearly with the number of training examples [23] so the complexity can only get worse with more training data

[2] See[13] for work with related kernels and on-line learning.

subject to $y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$, where $C > 0$ is the tradeoff between regularization and constraint violation. In the dual formulation we maximize:

$$W(\alpha) = \sum_{i=i}^{N} \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \quad (2)$$

$$\textit{subject to:} \quad 0 \leq \alpha_i \leq C \text{ and } \sum \alpha_i y_i = 0 \quad (3)$$

The decision function is $sign\,(h(\mathbf{x}))$, where:

$$h(\mathbf{x}) = \sum_{l=1}^{m} \alpha_l y_l k(\mathbf{x}, \mathbf{x}_l) + b \quad (4)$$

For clarity, in a slight abuse of notation the features $\mathbf{x}_l : l \in \{1, 2, \ldots, m\}$ will be referred to as support vectors. Thus, in general m kernel computations are needed to classify a point with a kernelized SVM and all m support vector must be stored. For linear kernels we can do better because, $k(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle$, so $h()$ can be written as $h(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$, where $\mathbf{w} = \sum_{l=1}^{m} \alpha_l y_l \mathbf{x}_l$. As a result classifying with a linear SVM only requires $\mathcal{O}(n)$ operations, and $\mathcal{O}(n)$ memory.

3.1. Fast Exact Intersection in O(n log m)

Now we show that it is possible to speed up classification for IKSVMs. For feature vectors $\mathbf{x}, \mathbf{z} \in \mathcal{R}_+^n$, the intersection kernel is $k(\mathbf{x}, \mathbf{z})$:

$$k(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{n} \min\,(x(i), z(i)) \quad (5)$$

and classification is based on evaluating:

$$h(\mathbf{x}) = \sum_{l=1}^{m} \alpha_l y_l k(\mathbf{x}, \mathbf{x}_l) + b \quad (6)$$

$$= \sum_{l=1}^{m} \alpha_l y_l \left(\sum_{i=1}^{n} \min\,(x(i), x_l(i)) \right) + b \quad (7)$$

The non linearity of $\min$ prevents us from 'collapsing' in a similar manner to linear kernels. This is in general true for any non linear kernel including radial basis functions, polynomial kernels, etc. Thus the complexity of evaluating $h(\mathbf{x})$ in the naive way is $\mathcal{O}(mn)$. The trick for intersection kernels is that we can exchange the summations in equation 7 to obtain:

$$h(\mathbf{x}) = \sum_{l=1}^{m} \alpha_l y_l \left(\sum_{i=1}^{n} \min\,(x(i), x_l(i)) \right) + b \quad (8)$$

$$= \sum_{i=1}^{n} \left(\sum_{l=1}^{m} \alpha_l y_l \min\,(x(i), x_l(i)) \right) + b \quad (9)$$

$$= \sum_{i=1}^{n} h_i(x(i)) + b \quad (10)$$

Rewriting the function $h(\mathbf{x})$ as the sum of the individual functions, h_i, one for each dimension, where

$$h_i(s) = \sum_{l=1}^{m} \alpha_l y_l \min\,(s, x_l(i))) \quad (11)$$

So far we have gained nothing as the complexity of computing each $h_i(s)$ is $\mathcal{O}(m)$ with an overall complexity of computing $h(\mathbf{x})$ still $\mathcal{O}(mn)$. We now show how to compute each h_i in $\mathcal{O}(\log m)$ time.

Consider the functions $h_i(s)$ for a fixed value of i. Let $\bar{x}_l(i)$ denote the sorted values of $x_l(i)$ in increasing order with corresponding α's and labels as $\bar{\alpha}_l$ and $\bar{y}_l$. If $s < \bar{x}_1(i)$ then $h_i(s) = 0$, otherwise let r be the largest integer such that $\bar{x}_r(i) \leq s$. Then we have,

$$h_i(s) = \sum_{l=1}^{m} \bar{\alpha}_l \bar{y}_l \min\,(s, \bar{x}_l(i)) \quad (12)$$

$$= \sum_{1 \leq l \leq r} \bar{\alpha}_l \bar{y}_l \bar{x}_l(i) + s \sum_{r < l \leq m} \bar{\alpha}_l \bar{y}_l \quad (13)$$

$$= A_i(r) + s B_i(r) \quad (14)$$

Where we have defined,

$$A_i(r) = \sum_{1 \leq l \leq r} \bar{\alpha}_l \bar{y}_l \bar{x}_l(i), \quad (15)$$

$$B_i(r) = \sum_{r < l \leq m} \bar{\alpha}_l \bar{y}_l \quad (16)$$

Equation 14 shows that h_i is piecewise linear. Furthermore h_i is continuous because:

$$h_i(\bar{x}_{r+1}) = A_i(r) + \bar{x}_{r+1} B_i(r)$$

$$= A_i(r+1) + \bar{x}_{r+1} B_i(r+1).$$

Notice that the functions A_i and B_i are independent of the input data and depend only on the support vectors and α. Thus, if we precompute $h_i(\bar{x}_r)$ then $h_i(s)$ can be computed by first finding r, the position of $s = x(i)$ in the sorted list $\bar{x}(i)$ using binary search and linearly interpolating between $h_i(\bar{x}_r)$ and $h_i(\bar{x}_{r+1})$. This requires storing the $\bar{x}_l$ as well as the $h_i(\bar{x}_l)$ or twice the storage of the standard implementation. Thus the runtime complexity of computing $h(\mathbf{x})$ is $\mathcal{O}(n \log m)$ as opposed to $\mathcal{O}(nm)$, a speed up of $\mathcal{O}(m/\log m)$. In our experiments we typically have SVMs with a few thousand support vectors and the resulting speedup is quite significant.

3.2. Fast Approximate Intersection in O(n)

We now show how to efficiently approximate the functions h_i in constant time and space independent of the number of support vectors, which leads to significant savings in memory at classification time. We showed in Section 3.1

that $h_i(s)$ can be represented exactly using $m+1$ piecewise linear segments. In our experiments we observe the distributions of the support vectors along each dimension tend to be smooth and concentrated[3] *and* that the resulting h_i are also relatively smooth as seen in Figure 1. This allows approximating h_i with simpler functions that can be evaluated quickly and require much less storage than the exact method presented above. We consider both piecewise constant and piecewise linear approximations with uniform spacing between segments in both cases. Uniform spacing allows computing $h_i(s)$ by a table lookup in the piecewise constant approximation or two table lookups and linear interpolation in the piecewise linear approximation. In practice we find that 30-50 linear segments is enough to avoid any loss in classification accuracy yielding a huge decrease in memory requirements.

As an example in our experiments using the Daimler Chrysler pedestrian dataset, we routinely learn classifiers with more than 5000 support vectors. The piecewise linear approximations require only 30 values to obtain the same classification accuracy in our experiments, using less than 1% of the memory.

3.3. Generalizations of Histogram Intersection

The algorithms we propose can be applied to more than just the histogram intersection kernel. In fact whenever a function can be decomposed as in equation 8 we can apply the approximation speedup in Section 3.2. One example is the generalized histogram intersection kernel [2] defined by:

$$K_{GHI}(\mathbf{x},\mathbf{z}) = \sum_{i=1}^{n} \min\left(|x(i)|^{\beta}, |z(i)|^{\beta}\right)$$

that is positive definite for all $\beta > 0$. Chappelle *et al*. [4] observe that this remapping of the histogram bin values by $x \rightarrow x^{\beta}$, improves the performance of linear kernel SVMs to become comparable to RBF kernels on an image classification task over the Corel Stock Photo Collection. In fact, we can use

$$K^{f}_{GHI}(\mathbf{x},\mathbf{z}) = \sum_{i=1}^{n} \min\left(f\left(x(i)\right), f\left(z(i)\right)\right) \qquad (17)$$

for arbitrary f taking non-negative values as a kernel, evaluate it exactly, and approximate it with our technique. Another example is the $1-\chi^2$ kernel, which has been used successfully in [1] and could be approximated.

4. Experiments

We perform experiments to answer several questions; **(1)** How much does the fast exact method speed up classification? **(2)** What space savings does the approximate method provide and at what cost in terms of the classification performance? **(3)** What improvement in classification performance does IKSVM offer over other kernel SVMs? **(4)** Does the intersection kernel allow simpler features for pedestrian detection?

In order to answer the first three questions we use the INRIA and Daimler-Chrysler pedestrian dataset for detection oriented experiments and Caltech-101 for classification. In order to answer the fourth question, we introduce features based on histograms of oriented edge energy responses and test the framework on the two pedestrian detection datasets. We compare our features to the features used by the state of the art algorithms in each dataset in terms of both simplicity and classification accuracy. The next section describes how our features are computed.

4.1. Multi-Level Oriented Edge Energy Features

Our features are based on a multi-level version of the HOG descriptor [7] and use histogram intersection kernel SVM based on spatial pyramid match kernel introduced in [15]. Our features are computed as follows: **(1)** We compute the oriented edge energy responses in 8 directions using the magnitude of the odd elongated oriented filters from [21] at a fine scale ($\sigma = 1$), with non max suppression performed independently in each orientation. **(2)** The response is then L_1 normalized over all the orientations in non overlapping cells of fixed size $h_n \times w_n$ **(3)** At each level $l \in \{1, 2 \ldots L\}$, the image is divided into non overlapping cells of size $h_l \times w_l$, and a histogram feature is constructed by summing up normalized response within the cell. **(4)** The features at a level l are weighted by a factor c_l and concatenated to form our feature vector, which is used to train an IKSVM classifier. The oriented edge energy histogram based features have been used successfully before for gesture recognition [10]. Figure 2 shows the first three stages of our feature pipeline.

Compared to the HOG features used in the linear SVM based detector of Dalal and Triggs [7], our features are much simpler as we do not have overlapping cells, Gaussian weighting of the responses within a cell, or image normalization, leading to a much smaller dimensional feature vector. Our features also do not require the training step used to learn the Local Receptive Field features [16], which in conjunction with quadratic kernel SVMs give the best results on Daimler-Chrysler pedestrian dataset. Our experiments show the multi-level histogram of oriented edge energy features lead to better classification accuracies on INRIA dataset while being as good as the state of the art on Daimler-Chrysler dataset, when used with IKSVM and is significantly better than a linear classifier trained on the same features. The next three sections describe the experiments on these datasets in detail.

[3] This is interesting because the features themselves have a heavy tail distribution.

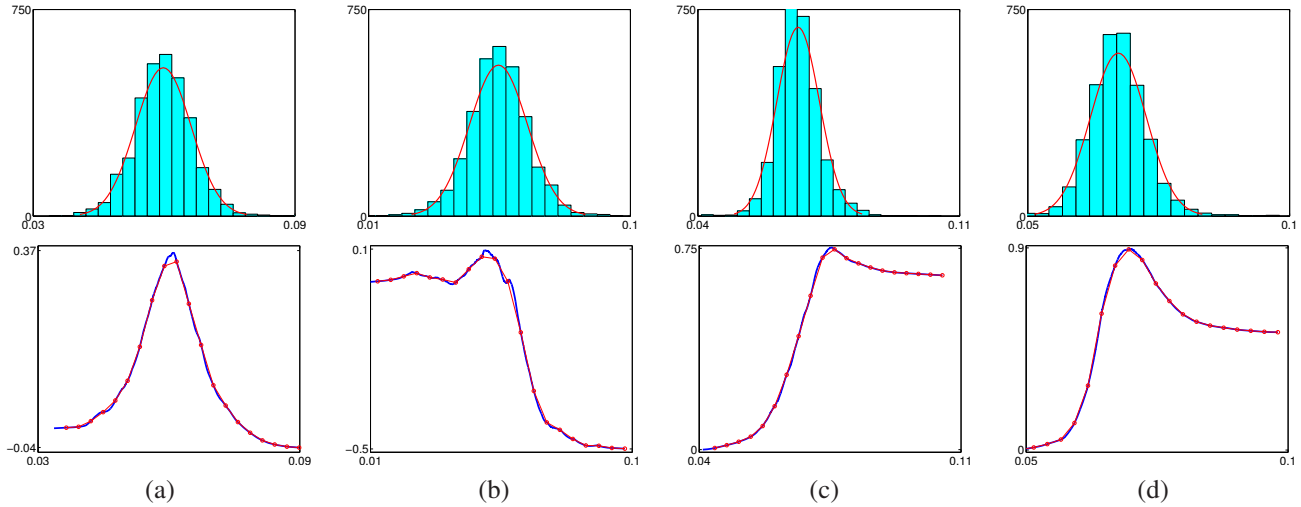

Figure 1. Each column (a-d) shows the distribution of the support vectors values along a dimension with a Gaussian fit (top) and the function $h_i(s)$ vs. s with a piecewise linear fit using 20 uniformly spaced points (bottom) of an IKSVM model trained on the INRIA dataset. Unlike the distribution of the training data which are heavy tailed, the values of the support vectors tend to be clustered.

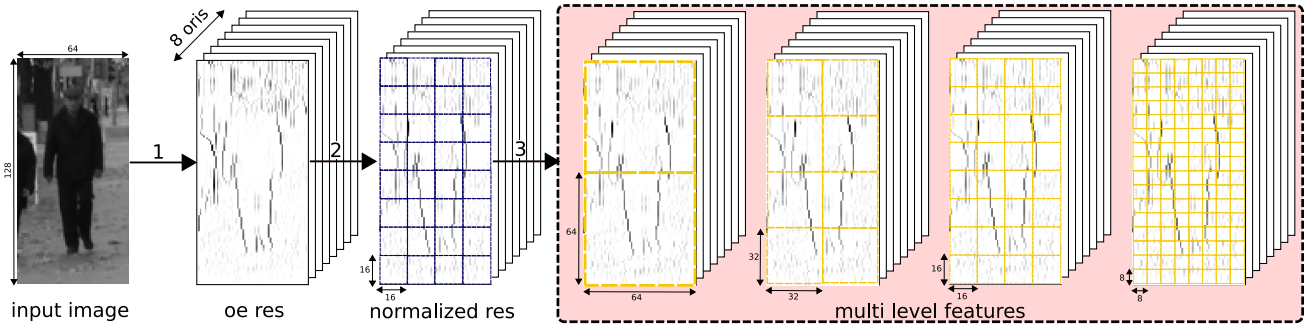

Figure 2. The three stage pipeline of the feature computation process. **(1)** The input grayscale image of size 64×128 is convolved with oriented filters ($\sigma = 1$) in 8 directions, to obtain oriented energy responses. **(2)** The responses are then L_1 normalized over all directions in each non overlapping 16×16 blocks independently to obtain normalized responses. **(3)** Multilevel features are then extracted by constructing histograms of oriented gradients by summing up the normalized response in each cell. The diagram depicts progressively smaller cell sizes from 64×64 to 8×8.

4.2. INRIA Pedestrian Dataset

The INRIA pedestrian dataset was introduced in [7]) as an alternate to the existing pedestrian datasets (*e.g.* MIT Pedestrian Dataset) and is significantly hard because of wide variety of articulated poses, variable appearance/clothing, illumination changes and complex backgrounds. Linear kernel SVMs with histograms of oriented gradients (HOG) features achieve high accuracy and speed on this dataset [7]. In our experiments we use the multilevel oriented edge energy features introduced in Section 4.1 with $L = 4$ levels, cell sizes of 64×64, 32×32, 16×16 and 8×8 at levels $1, 2, 3$ and 4 respectively and block normalization window size of $w_n \times h_n = 16 \times 16$. The features at a level l are weighted by a factor $c_l = 1/4^{(L-l)}$ to obtain a 1360 dimensional vector which is used to train an IKSVM classifier.

The performance of the exact IKSVM classifier and various approximations are shown in Figure 3. Firstly, the performace of the IKSVM classifier using our features is significantly better. We have a miss rate of 0.167 at 10^{-6} FPPW and 0.025 at 10^{-4} False Positive Per Window (FPPW), an improvement of about 13% and 11% respectively, over the Dalal and Triggs pedestrian detector. This is without any tuning of hyperparameters like the number of levels, level weights, choice of cell sizes, which is remarkable. We observe that the piecewise linear approximation with 30 bins is almost as accurate as the exact classifier. The piecewise constant approximation however requires, thrice as many bins to obtain similar accuracies. The detection results as a function of number of bins for piecewise linear and piecewise constant approximations are shown in Fig-

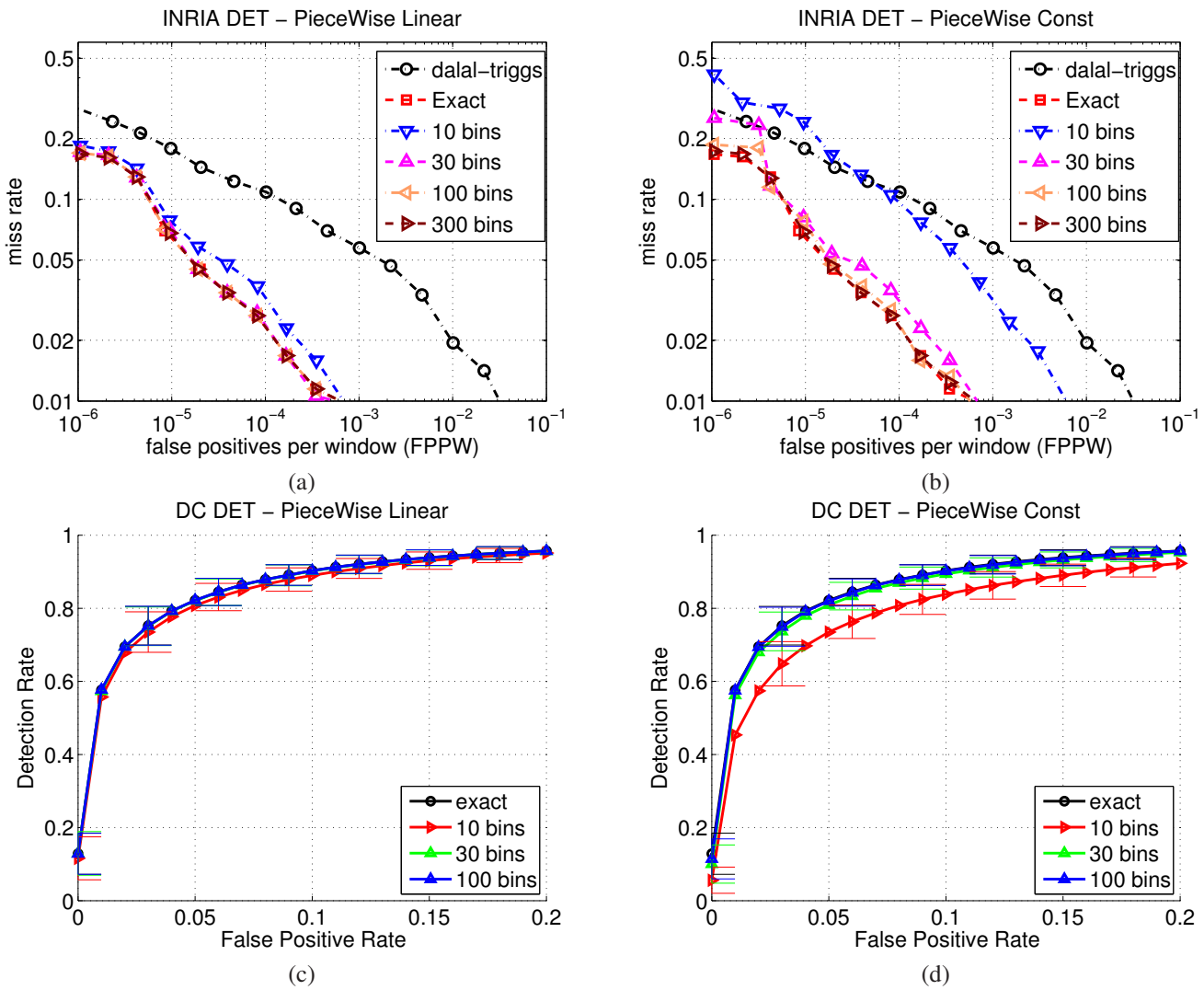

Figure 3. Performance of our methods on INRIA pedestrian dataset (top row) and Daimler Chrysler pedestrian dataset (bottom row). Detection plots for an exact IKSVM and its **(a)** piecewise linear approximation **(b)** piecewise constant approximation. The performance of the exact IKSVM is significantly better than the Dalal and Triggs detector. About 30 bins are required for the piecewise linear approximation to have an accuracy close to the exact, while the piecewise constant do the same with 100 bins. Mean detection rate with error bars of the exact IKSVM and its **(c)** piecewise linear approximation and **(d)** piecewise constant approximation. The error bars are plotted by training on 2 out of the 3 training sets and testing on 2 test sets for a total of 6 results. The results are as good as the best published results on this task. Once again 30 bins for a piecewise linear and 100 bins for a piecewise constant approximation are close to the exact.

ure 3. Table 1 compares the classification speeds of various methods. The classification speed of our classifier is $6\times$ slower than a linear SVM using 1360 features. As a comparison a cell size of 8×8, stride of 8×8, block size of 16×16 and 9 orientation bins used to generate the curve using the HOG features has a dimension of 3780, about three times as many features than ours. Thus, compared to a linear classifier with 3780 features the approximate IKSVM classifier is only about 2x slower. Overall the approximate IKSVM classifier is **1000**$\times$ faster and requires **100**$\times$ less memory than the standard implementation of intersection kernel SVM, with practically the same accuracy. Interestingly our features are not the optimal features for a linear SVM classifier, which has a miss rate of 0.5 at 10^{-4} FPPW. Figure 4 shows a sample of the errors made by our detector on this dataset.

4.3. Daimler Chrysler Pedestrian Dataset

We use the Daimler Chrysler Pedestrian Benchmark dataset, created by Munder and Gavrila [16]. The dataset is split into five disjoint sets, three for training and two for testing. Each training set has 5000 positive and negative examples each, while each test set has 4900 positive and neg-

ative examples each. We report the ROC curves by training on two out of three training sets at a time and testing on each of the test sets to obtain six curves from which the confidence intervals are plotted. The third training set is meant to be used for cross validation for tuning of the hyperparameters but we do not use it. Due to small size of the images (18×36), we only compute the multi-level features with only three levels ($L = 3$) of pyramid with cell sizes 18×18, 6×6 and 3×3 at levels 1, 2 and 3 respectively. The block normalization is done with a cell size of $w_n \times h_n = 18 \times 18$. The features at level l are weighted by a factor $c_l = 1/4^{(L-l)}$ to obtain a 656 dimensional vector, which is used to train an IKSVM classifier.

The classification results using the exact methods and approximations are shown in Figure 3. Our results are comparable to the best results for this task [16]. The experiments suggests that relatively simpler features compared to the LRF features when used in conjunction with IKSVMs perform as well as other kernel SVMs. Table 1 compares the classification speeds of various methods. The speedups obtained for this task are significantly large due to large number of support vectors for each classifier. The piecewise linear with 30 bins is about **2000**$\times$ faster and requires **200**$\times$ less memory, with no loss in classification accuracy. The piecewise constant approximation on the other hand requires about 100 bins for similar accuracies and is even faster. We do not however optimize the process of computing the features which on an implementation in Matlab takes about $17ms$ per image. The time for classification ($0.02ms$) is negligible compared to this. Compared to the $250ms$ required by the cascaded SVM based classifiers in the paper, our combined time for computing the features and classification is $15\times$ lesser. Figure 4 shows a sample of the errors made by our detector.

4.4. Caltech 101

Our third set of experiments are on Caltech-101 [8]. The aim here is to show that existing methods can be made significantly faster, even when the number of support vectors in each classifier is small. We use the framework of [15] and use our own implementation of their 'weak features' and achieve an accuracy of 52% (compared to their 54%), with 15 training and test examples per class and one-vs-all classifiers based on IKSVM. The performance of a linear SVM using the same features is about 40%. The IKSVM classifiers on average have 175 support vectors and a piecewise linear approximation with 60 bins is $50\times$ faster than a standard implementation, with 3 additional misclassifications out of 1530 test examples on average (see Table 1).

5. Conclusions

The theoretical contribution of this paper is a technique for exactly evaluating intersection kernel SVMs with runtime logarithmic in the number of support vectors as opposed to linear for the standard implementation. Furthermore we have shown that an approximation of the IKSVM classifier can be built with the same classification performance but runtime constant in the number of support vectors. This puts IKSVM classifiers in the same order of computational cost for evaluation as linear SVMs.

Our experimental results show that approximate IKSVM consistently outperforms linear SVMs at a modest increase in runtime for evaluation. This is especially relevant for detection tasks where a major component of training models is evaluating the detector on large training sets.

Finally we introduced a multi-scale histogram of oriented edge energy feature quite similar to HOG, but with a simpler design and lower dimensionality. This feature and IKSVM produce classification rates significantly better than the linear SVM based detector of Dalal and Triggs, leading to the current state of the art on pedestrian detection.

References

[1] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *ICIVR*, 2007.

[2] S. Boughorbel, J.-P. Tarel, and N. Boujemaa. Generalized histogram intersection kernel for image recognition. In *ICIP*, Genova, Italy, 2005.

[3] C. J. C. Burges. Simplified support vector decision rules. In *ICML*, pages 71–77, 1996.

[4] O. Chapelle, S. P. Haffner, and V. Vapnik. Support vector machines for histogram-based image classification. *IEEE Trans. on Neural Networks*, 10(5):1055–1064, May 1999.

[5] O. Chum and A. Zisserman. Presented at visual recognition challenge workshop. 2007.

[6] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[8] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE T. PAMI*, 28(4):594–611, 2006.

[9] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008.

[10] W. T. Freeman and M. Roth. Orientation histograms for hand gesture recognition. In *Intl. Workshop on Automatic Face and Gesture Recognition*, pages 296–301, 1995.

[11] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. *ICCV*, 2, 2005.

[12] B. Heisele, T. Serre, S. Prentice, and T. Poggio. Hierarchical classification and feature reduction for fast face detection with support vector machines. *Pattern Recognition*, 36:2007–2017(11), September 2003.

[13] M. Herbster. Learning additive models online with fast evaluating kernels. In *Fourteenth Annual Conference on Computational Learning Theory*, volume 2111, pages 444–460, 2001.

[14] S. S. Keerthi, O. Chapelle, and D. DeCoste. Building support vector machines with reduced classifier complexity. *JMLR*, 7:1493–1515, 2006.

[15] L. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[16] S. Munder and D. M. Gavrila. An experimental study on pedestrian classification. *IEEE T. PAMI*, 28(11):1863–1868, 2006.

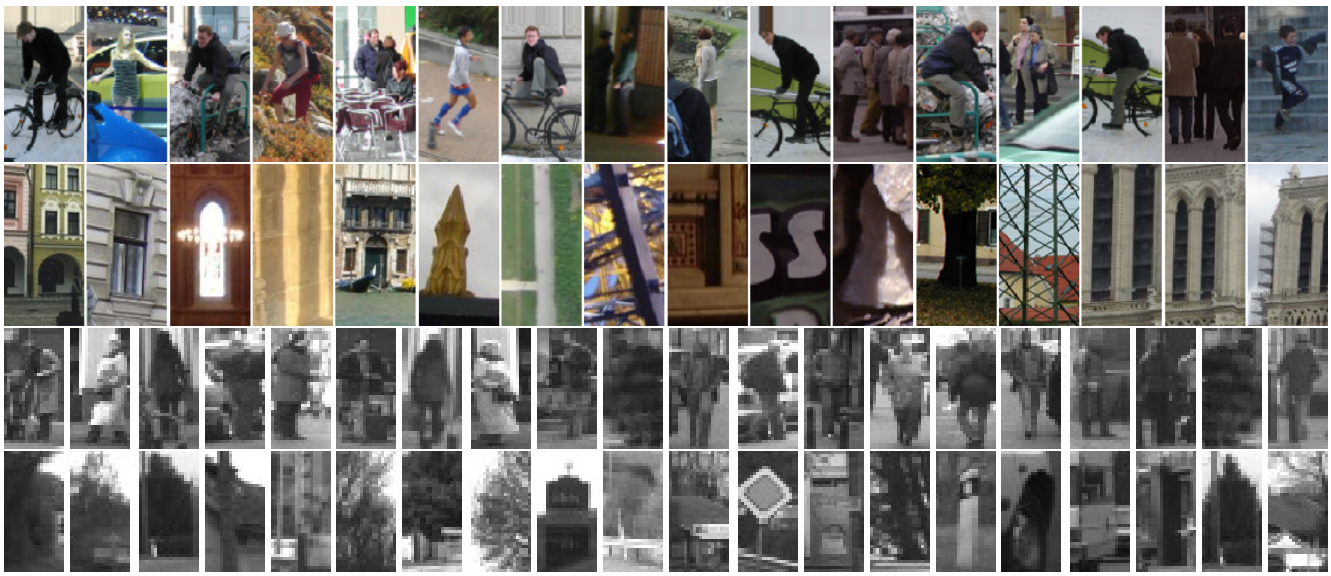

Figure 4. Top two rows are a sample of errors on the INRIA pedestrian dataset and the bottom two are on the Daimler-Chrysler dataset. For each set the first row is a sample of the false negatives, while the second row is a sample of the false positives.

	Model parameters		SVM kernel type		fast IKSVMs		
Dataset	#SVs	#features	linear	intersection	binary search	piecewise-const	piecewise-lin
INRIA Ped	3363	1360	0.07±0.00	659.1±1.92	2.57±0.03	0.34±0.01	0.43±0.01
DC Ped	5474±395	656	0.03±0.00	459.1±31.3	1.43±0.02	0.18±0.01	0.22±0.00
Caltech 101	175±46	1360	0.07±0.01	24.77±1.22	1.63±0.12	0.33±0.03	0.46±0.03

Table 1. **Time taken** in seconds to classify 10,000 features using various techniques. #SVs is the number of support vectors in the classifier and #features is the feature dimension. The next two columns show the speed of a linear kernel SVM with the same features and a standard implementation of a intersection kernel SVM (IKSVM). The next three columns show the speeds using the speedup techniques we propose. The numbers are averaged over various runs in the INRIA dataset, categories in the Caltech 101 dataset and six training and testing splits in the Daimler-Chrysler dataset. The binary search based IKSVM is exact and has logarithmic dependence on the number of support vectors, while the piecewise constant and linear approximations theoretically have a runtime independent of the number of support vectors. One can see this from the fact that even with about $20\times$ increase in the number of support vectors from Caltech 101 to INRIA pedestrian dataset, the speeds of the fast IKSVM using binary search increases $1.5\times$ while the speeds for the piecewise linear and constant approximations are unchanged. The speed of a naive implementation is about $25\times$ worse however. The approximate IKSVM classifiers are on an average just 5-6 times slower than linear kernel SVM. The exact version using binary search is also orders of magnitude faster on the pedestrian detection tasks than a naive implementation of the classifier. All the experiments were done on an Intel QuadCore 2.4 GHz machine with 4GB RAM.

[17] A. Odone F. Barla, A. Verri. Building kernels from binary strings for image matching. *IEEE T. Image Processing*, 14(2):169–180, Feb. 2005.

[18] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. *CVPR*, 1997.

[19] E. E. Osuna and F. Girosi. Reducing the run-time complexity in support vector machines. *Advances in kernel methods: support vector learning*, pages 271–283, 1999.

[20] C. Papageorgiou and T. Poggio. A trainable system for object detection. *IJCV*, 38(1):15–33, 2000.

[21] J. Perona, P.; Malik. Detecting and localizing edges composed of steps, peaks and roofs. *ICCV*, pages 52–57, 4-7 Dec 1990.

[22] S. Romdhani, P. Torr, B. Scholkopf, and A. Blake. Computationally efficient face detection. *ICCV*, 02:695, 2001.

[23] I. Steinwart. Sparseness of support vector machines-some asymptotically sharp bounds. In *NIPS*, 2003.

[24] M. J. Swain and D. H. Ballard. Color indexing. *IJCV*, 7(1):11–32, 1991.

[25] A. Torralba, K. Murphy, and W. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *CVPR*, 2004.

[26] M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. In *ICCV*, October 2007.

[27] P. Viola and M. J. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, 2004.

[28] G. J. Z. Wei Zhang and D. Samaras. Real-time accurate object detection using multiple resolutions. In *ICCV*, 2007.

[29] C. Yang, R. Duraiswami, N. A. Gumerov, and L. Davis. Improved fast gauss transform and efficient kernel density estimation. In *ICCV*, 2003.

[30] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. In *CVPRW*, 2006.