A Fast Dual Method for HIK SVM Learning

Jianxin Wu*

School of Computer Engineering, Nanyang Technological University
jxwu@ntu.edu.sg

Abstract. Histograms are used in almost every aspect of computer vision, from visual descriptors to image representations. Histogram Intersection Kernel (HIK) and SVM classifiers are shown to be very effective in dealing with histograms. This paper presents three contributions concerning HIK SVM classification. First, instead of limited to integer histograms, we present a proof that HIK is a positive definite kernel for non-negative real-valued feature vectors. This proof reveals some interesting properties of the kernel. Second, we propose ICD, a deterministic and highly scalable dual space HIK SVM solver. ICD is faster than and has similar accuracies with general purpose SVM solvers and two recently proposed stochastic fast HIK SVM training methods. Third, we empirically show that ICD is not sensitive to the C parameter in SVM. ICD achieves high accuracies using its default parameters in many datasets. This is a very attractive property because many vision problems are too large to choose SVM parameters using cross-validation.

1 Introduction

Recently, the Histogram Intersection Kernel (HIK) has attracted a lot of attention in the computer vision community. The success of HIK can be attributed to at least two important factors:

- First, histograms are frequently used in solving vision problems. At the feature level, many visual descriptors are histograms of various image measurements, e.g., SIFT [11], HOG [5], CENTRIST [20], or histogram of LBP [15], just to name a few. At the image level, histogram is also a popular representation, e.g., color histogram [17] or bag of visual words.
- Second, it is shown that HIK, as a measure for comparing the similarity (or dissimilarity) of *two histograms*, achieves better performances in various machine learning tasks than other commonly used measures, e.g., l_2 distance or RBF kernel. HIK is shown to have higher accuracies in SVM classification [13,19], and in the clustering of *histograms* [19].

Recently HIK becomes even more attractive for its fast evaluation speed [13,19]. It is shown that

$$\sum_{i=1}^{n} c_i \kappa_{\mathrm{HI}}(\boldsymbol{q}, \boldsymbol{x}_i) \tag{1}$$

* The author is supported by the NTU startup grant.

K. Daniilidis, P. Maragos, N. Paragios (Eds.): ECCV 2010, Part II, LNCS 6312, pp. 552–565, 2010.

can be computed in $O(d)$ steps, where $\{\boldsymbol{x}_i\}_{i=1}^n$ are a set of n d-dimensional histograms, $\boldsymbol{q}$ is a query histogram, and $\kappa_{\text{HI}}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{j=1}^d \min(x_j, y_j)$ is the histogram intersection kernel. Although the effectiveness of HIK and other measures for comparing histogram (e.g., the χ^2 distance) have not been extensively compared, the fast computing of Eqn. 1 makes HIK particularly attractive.

In this paper we make three contributions related to HIK SVM learning:

1. We show that HIK is a positive definite (PD) kernel for non-negative real-valued histograms. HIK is known to be a valid kernel for non-negative integer histograms [14]. We give a proof for non-negative real valued histograms. Our proof also completes the missing part of [12], which proved that HIK is conditionally positive definite (CPD).
2. We propose ICD, a fast dual HIK SVM training algorithm. ICD solves the SVM problem without re-encoding the input (which is a necessary step in [12]). It explicitly finds the feature space decision boundary, while the computations are carried out in the input space efficiently. ICD is a deterministic algorithm and do not need to choose a step size for optimization. We empirically show that ICD not only converges faster than the methods of [12,18], but also yields higher accuracies.
3. We show that ICD is robust to the SVM parameter C *in practice*. Choosing SVM parameters by cross validation is very time consuming, but crucial for linear and RBF kernels, or the methods in [12,18]. We empirically show that SVM parameters have only slight effects in ICD thus parameter selection is not necessary.

2 Related Work

The histogram intersection kernel (HIK) is originally proposed by Swain and Ballard for color-based object recognition [17]. It is further shown to be a positive definite kernel when the histograms only contain non-negative integers [14], which makes HIK suitable for SVM classification. HIK is shown to be a conditionally positive definite (CPD) kernel in real-valued cases [12]. We will give a proof that HIK on non-negative real-valued vectors is a positive definite kernel in Sec. 3.1.

HIK has shown to be a suitable similarity measure for comparing two histograms in different machine learning tasks. For example, it achieved higher accuracies in SVM classification than linear or RBF kernel [13,19] in different domains, including object recognition, object detection, place recognition, and scene recognition (whose feature vectors are histograms). In unsupervised learning tasks, kernel k-means clustering using HIK was also shown to produce better visual codebooks (and consequently achieved consistently higher accuracies in the resulting bag of visual words model) than the normal k-means algorithm [19].

A naive method to compute Eqn. 1 will take $O(nd)$ steps, which is very expensive when either n or d is large. However, Eqn. 1 (with different assignment of the weights c_i) is crucial in the training and testing of HIK SVM classifiers, and in HIK based clustering. Recently, [13] showed that Eqn. 1 can be computed

in $O(d \log n)$ steps (and $O(d)$ steps if an approximation is allowed with only slight loss of accuracy). Furthermore, [19] showed that by first quantizing the vectors to integers, exact answer can be obtained in $O(d)$ steps with less overhead than the method in [13]. Fast computation of Eqn. 1 enables the testing of HIK SVM classifiers to have the same complexity as that of linear SVM [13,19], and make HIK clustering almost as fast as the usual k-means clustering [19]. In Sec. 3.2 we will show that the method in [19] is not only a way to accelerate computation, but has a physical interpretation.

These computational methods are also applied in fast training of HIK SVMs, in which Eqn. 1 is again the speed bottleneck. Stochastic gradient descent (SGD) methods are used in [18,12] to train an HIK SVM. More than 10 fold acceleration can be achieved by using the fast method to compute Eqn. 1. PWLSGD [12] is based on the stochastic method Pegasos (Primal Estimated sub-GrAdient SOlver for SVM) [16], and SIKMA [18] is another SGD method. One drawback of these methods is that it is subtle to choose a step size in the gradient descent update, which is important to the success of SGD methods. Also, SGD methods give different results in different runs on the same dataset.

3 The Histogram Intersection Kernel

3.1 HIK in $\mathbb{R}_+$ Is a Positive Definite Kernel

Let $\mathbb{R}_+$ be the set of non-negative real numbers $\{x \geq 0 | x \in \mathbb{R}\}$. We will prove that the histogram intersection kernel $\kappa_{\mathrm{HI}}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sum_{j=1}^{d} \min(x_{1,j}, x_{2,j})$ is a valid positive definite kernel for $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}_+^d$.

We use n to denote the number of data points and d for the dimension, and $x_{i,j}$ as the j-th component of a vector $\boldsymbol{x}_i$. We will always use i to index a training example, and use j to index a feature dimension.

We first prove this fact for $d = 1$. Given n real numbers $x_1, \ldots, x_n \in \mathbb{R}_+$, we assume that $x_i \leq x_{i'}$ whenever $1 \leq i < i' \leq n$ without the loss of generality. Thus the kernel matrix K of this set has the property that

$$K_{ii'} = \min(x_i, x_{i'}) = x_{\min(i,i')}. \tag{2}$$

It is easy to verify that $\Lambda = R^T K R$, where R and Λ are defined as:

$$R_{ij} = \begin{cases} 1 & \text{if } i = j \\ -1 & \text{if } i = j - 1 \\ 0 & \text{otherwise} \end{cases}, \quad \text{and} \quad \Lambda_{ij} = \begin{cases} x_1 & \text{if } i = j = 1 \\ x_i - x_{i-1} & \text{if } i = j > 1 \\ 0 & \text{otherwise} \end{cases}. \tag{3}$$

The diagonal matrix Λ is positive semidefinite, so is K. Thus κ_{HI} is a positive definite kernel when $d = 1$. The generalization to $d > 1$ is straight forward, because the sum of Mercer kernels is again a Mercer kernel [4].

HIK is proved to be conditionally positive definite (CPD) in $\mathbb{R}$ [12], i.e., $\mathbf{x}^T K \mathbf{x} \geq 0$ when $\sum_j x_j = 0$. However, the proof in [12] is incomplete. The final step of the proof of [12] used the fact that HIK is a positive definite kernel

in $\mathbb{R}_+$, which we have just proved. CPD kernels can be safely used in an SVM if the bias term is included, but may have problem if we do not use the bias term (e.g., the SVM in Eqn 11 does not include the bias term.)

One important note is that "HIK is p.d. in $\mathbb{R}_+$" can be proved by setting $\beta = 1$ in Proposition 3 of [1]. Our method, though, provides a new intuitive proof that reveals interesting structures of HIK. It is also worth mentioning that Proposition 3 in [1] can also be easily proved using our technique.

3.2 Equation 1 and Its Feature Space Interpretation

There is a more intuitive way to illustrate that HIK is a Mercer kernel when the histograms only contain non-negative integers [14]. Given a d dimensional histogram $\boldsymbol{x}$, whose elements are all smaller than or equal to an upper bound $\bar{v}$. We define a mapping $B : \mathbb{N} \to \mathbb{R}^{\bar{v}}$ as (i.e., the unary representation)

$$B(x) = [\underbrace{1, 1, \ldots, 1}_{x \text{ times}}, \underbrace{0, 0, \ldots, 0}_{\bar{v}-x \text{ times}}], \tag{4}$$

Then the feature space spanned by κ_{HI} is $d\bar{v}$ dimensional, and a vector $\boldsymbol{x} \in \mathbb{N}^d$ is mapped to $B(\boldsymbol{x}) = [\, B(x_1), \cdots, B(x_d)\,] \in \mathbb{R}^{d\bar{v}}$.

This fact is easy to prove because $xy = \min(x, y)$ when $x, y \in \{0, 1\}$. Thus $\kappa_{\text{HI}}(\boldsymbol{x}, \boldsymbol{y}) = B(\boldsymbol{x})^T B(\boldsymbol{y})$. Using the feature space for integer histograms, we can give a clear interpretation of the method presented in [19] for computing Eqn. 1.

Given a dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ in which we assume that the elements of $\boldsymbol{x}_i$ are non-negative integers not larger than $\bar{v}$, and $y_i \in \{-1, +1\}$. An HIK SVM classifier will be

$$f(\boldsymbol{q}) = \sum\nolimits_{i=1}^{n} \alpha_i y_i \kappa_{\text{HI}}(\boldsymbol{q}, \boldsymbol{x}_i) - \theta \tag{5}$$

for a test example $\boldsymbol{q}$, in which α_i are the Lagrange multipliers. Note that if we set $c_i = \alpha_i y_i$, this equation is a special case of Eqn. 1.

We define a matrix $T \in \mathbb{R}^{d\bar{v}}$ as (in which $c_i = \alpha_i y_i$) $T_{j,k} = \sum_{i:k \geq x_{i,j}} c_i x_{i,j} + k \sum_{i:k<x_{i,j}} c_i$. Then it is shown in [19] that

$$f(\boldsymbol{q}) = \sum\nolimits_{j=1}^{d} T_{j,q_j} - \theta. \tag{6}$$

Now consider the dataset $\{(B(\boldsymbol{x}_i), y_i)\}_{i=1}^n$, i.e., we study the same problem in the feature space instead. Let us assume that a linear SVM in the feature space results in the solution vector $\boldsymbol{w} = [\, \boldsymbol{w}_1, \cdots, \boldsymbol{w}_d] \in \mathbb{R}^{d\bar{v}}$, where $\boldsymbol{w}_i \in \mathbb{R}^{\bar{v}}$ is the weights corresponding to $B(x_i)$. Then we must have

$$f(\boldsymbol{q}) = \boldsymbol{w}^T B(\boldsymbol{q}) - \theta = \sum\nolimits_{j=1}^{d} \boldsymbol{w}_j^T B(q_j) - \theta. \tag{7}$$

Comparing Eqn. 7 and 6, we get that

$$\boldsymbol{w}_j^T B(q_j) = T_{j,q_j} \qquad \forall j \in \{1, \ldots, d\}, q_j \in \{0, 1, \ldots, \bar{v}\}. \tag{8}$$

In other words, we have: for all $j \in \{1, \ldots, d\}$ and $k \in \{0, 1, \ldots, \bar{v}\}$

$$T_{j,k} = \sum\nolimits_{t=1}^{k} w_{j,t}. \tag{9}$$

In short, we just revealed that there is a bijection between the table T and the decision boundary $\boldsymbol{w}$ in the feature space. The key benefit of using the table T is that we do not need to explicitly store $\{B(\boldsymbol{x})\}_{i=1}^{n}$. Also, Eqn. 6 is very efficient ($O(d)$).

3.3 ICD: Intersection Coordinate Descent

This intuition can be used in fast training of HIK SVM classifiers. After quantizing the dataset to integers (with maximum feature value $\bar{v}$), we will solve a linear SVM problem in the feature space $\mathbb{R}^{d\bar{v}}$. However, instead of creating and storing $B(\boldsymbol{x}_i) \in \mathbb{R}^{d\bar{v}}, i = 1, \ldots, n$, we will make use of data structures like the table T to carry out computations in the input space $\mathbb{N}^d$. We do not need to re-encode the input data as the PWLSGD method in [12].

Our method is based on the SVM solver in LIBLINEAR [8], which uses a dual coordinate descent method. Given a dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ where $y_i \in \{-1, +1\}$, its corresponding dataset in the feature space is $\{(B(\boldsymbol{x}_i), y_i)\}_{i=1}^{n}$. A linear SVM in the feature space solves the following problem:

$$\min_{\boldsymbol{w}} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum\nolimits_{i=1}^{n} \xi\left(\boldsymbol{w}; B(\boldsymbol{x}_i), y_i\right), \tag{10}$$

where $\xi\left(\boldsymbol{w}; B(\boldsymbol{x}_i), y_i\right) = \max(1 - y_i\boldsymbol{w}^T B(\boldsymbol{x}_i), 0)^2$ is the L2-loss function. The parameter C controls a trade-off between maximum margin and empirical errors on the training set.

The primal problem (Eqn. 10) is equivalent to the following dual form

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \;\; & g(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T\bar{Q}\boldsymbol{\alpha} - \boldsymbol{e}^T\boldsymbol{\alpha} \\ \text{subject to} \;\; & 0 \le \alpha_i \le U, \forall i, \end{aligned} \tag{11}$$

where $U = \infty$, $\bar{Q} = Q + D$, $Q_{ii'} = y_i y_{i'} B(\boldsymbol{x}_i)^T B(\boldsymbol{x}_{i'})$, D is a diagonal matrix and $D_{ii} = 1/(2C)$ in an L2-loss SVM. Note that $\boldsymbol{\alpha} \in \mathbb{R}^n$ and $\boldsymbol{w} = \sum_i \alpha_i y_i B(\boldsymbol{x}_i)$.

In [8] the dual problem is solved using coordinate descent. The values of α_i are updated sequentially for $i = 1, 2, \ldots, n$. When updating α_i, a new α_i' is chosen such that it will reduce $g(\boldsymbol{\alpha})$ by the largest amount, while still in the range $[0\ U]$. The discriminant function $\boldsymbol{w}$ is then incremented by $(\alpha_i' - \alpha_i) y_i B(\boldsymbol{x}_i)$, i.e., updated using the i-th data point $B(\boldsymbol{x}_i)$. A hypothetical algorithm to solve SVM in the feature space is shown in Algorithm 1.

However, we will not use this algorithm in practice. The main difficulty of applying Algorithm 1 is to compute line 1, 4, and 9 without explicitly constructing the high dimensional vectors $\boldsymbol{w}$ and $B(\boldsymbol{x}_i)$. Instead we will use the table T.

We do not need to compute line 1 because there is a bijection between $\boldsymbol{w}$ and T. A proper initialization of T will replace line 1. We simply initialize all elements of T to 0, which is equivalent to initialize $\boldsymbol{w}$ to 0.

Algorithm 1. A hypothetical algorithm for HIK SVM in the feature space

1: Given $\boldsymbol{\alpha}$ and correspondingly $\boldsymbol{w} = \sum_i \alpha_i y_i B(\boldsymbol{x}_i)$
2: **while** $\boldsymbol{\alpha}$ is not optimal **do**
3: **for** $i = 1, \ldots, n$ **do**
4: $G = y_i \boldsymbol{w}^T B(\boldsymbol{x}_i) - 1 + D_{ii}\alpha_i$
5: $PG = \begin{cases} \min(G, 0) & \text{if } \alpha_i = 0 \\ \max(G, 0) & \text{if } \alpha_i = U \\ G & \text{if } 0 < \alpha_i < U \end{cases}$
6: **if** $|PG| \neq 0$ **then**
7: $\bar{\alpha}_i \leftarrow \alpha_i$
8: $\alpha_i \leftarrow \min(\max(\alpha_i - G/\bar{Q}_{ii}, 0), U)$
9: $\boldsymbol{w} \leftarrow \boldsymbol{w} + (\alpha_i - \bar{\alpha}_i) y_i B(\boldsymbol{x}_i)$
10: **end if**
11: **end for**
12: **end while**

Similarly, Eqn. 6 can be used to efficiently compute $\boldsymbol{w}^T B(\boldsymbol{x}_i)$ in $O(d)$ steps, which makes line 4 easy to compute. The remaining difficulty is then how to update $\boldsymbol{w}$, or equivalently, how to update T because we do not store $\boldsymbol{w}$.

Let us denote $(\alpha_i - \bar{\alpha}_i)y_i$ as δ_{α_i}. Then the change of $\boldsymbol{w}$ (line 9) is now $\Delta\boldsymbol{w} = \delta_{\alpha_i} B(\boldsymbol{x}_i)$, or equivalently ($B(x_{i,j})_t$ is the t-th element of $B(x_{i,j})$),

$$\Delta w_{j,t} = \delta_{\alpha_i} B(x_{i,j})_t, \text{ for all } 1 \leq j \leq d, 1 \leq t \leq \bar{v}.$$

Using Eqn. 9, it is easy to find that

$$\begin{aligned} \Delta T_{j,k} &= \sum_{t=1}^{k} \Delta w_{j,t} \\ &= \sum_{t=1}^{k} \delta_{\alpha_i} B(x_{i,j})_t = \delta_{\alpha_i} \sum_{t=1}^{k} B(x_{i,j})_t \\ &= \delta_{\alpha_i} \min(x_{i,j}, k). \end{aligned} \tag{12}$$

The last equality in Eqn. 12 follows from the identity $\sum_{t=1}^{k} B(x)_t = \min(x, k)$.

In summary, solving an HIK SVM optimization can be done using the dual coordinate descent approach. The computations are carried out on the original histograms instead of in the high dimensional feature space, which maintains the fast training speed. The HIK SVM training algorithm is shown in Algorithm 2, in which we assume that $T_{j,0} = 0$ for any $j \in \{1, \ldots, d\}$. We will refer to Algorithm 2 as the ICD (Intersection Coordinate Descent) method.

After an SVM is trained using ICD, a new example $\boldsymbol{q}$ can be classified in $O(d)$ steps using Eqn. 6, which is the same complexity as that of a linear SVM.

3.4 L1-Loss, Multi-class, Convergence, and All That

The ICD algorithm is provided at `http://www.ntu.edu.sg/home/jxwu` inside the libHIK package. Beyond Algorithm 2, fast HIK SVM training method using

Algorithm 2. ICD: A method for training HIK SVM

{Replace the following lines in Algorithm 1, the remaining lines of Algorithm 1 will be omitted here. }
line $1'$: $T_{j,k} \leftarrow 0,$ for all $j \in \{1, \ldots, d\}, k \in \{1, \ldots, \bar{v}\}$
{Note that the below commands update the table T using $\boldsymbol{x}_i$ }
line $4'$: $G = y_i \sum_{j=1}^{d} T_{j,x_{i,j}} - 1 + D_{ii}\alpha_i$
line $9'$: $T_{j,k} \leftarrow T_{j,k} + (\alpha_i - \bar{\alpha}_i) y_i \min(x_{i,j}, k), \forall j \in \{1, \ldots, d\}, k \in \{1, \ldots, \bar{v}\}$

L1-loss, in primal space, and for multi-class datasets are also provided. Due to the space limit, we will only briefly discuss some important issues.

ICD can use L1-loss function $\xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i) = \max(1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i, 0)$ [8], by setting $U = C$ and $D_{ii} = 0$ in Eqn. 11.

We can accelerate the solving of the primal problem (Eqn. 10) using the same idea of ICD. We maintain both $\boldsymbol{w}$ and the table T during training. There is no need to explicitly create $B(\boldsymbol{x}_i)$. Primal method is preferred when $d \gg n$.

We can solve multi-class problems using the one versus rest method. The Crammer-Singer formulation [3] can also be greatly accelerated by implicit feature space computations using Eqn. 6.

The global convergence Theorem 1 of [8] readily applies to ICD. Thus the ICD method obtains an ϵ-accurate solution in $O(\log(1/\epsilon))$ iterations.

In ICD there are $d\bar{v}$ numbers to change when updating each α_i,[1] while in linear SVM we only need to update d numbers. However, in practice ICD requires a much smaller number of iterations to converge than that of linear SVM. On many real world datasets, ICD converges within 30 iterations, while linear SVM is not converged after 1000 iterations. Thus the training time of ICD is much faster than $\bar{v}$ times of the linear SVM training time. On some difficult datasets ICD is even faster than LIBLINEAR (c.f. Sec. 4).

3.5 Quantization, Default Bin Number, and Default C Parameter

When the feature vectors are not natural integer histograms, we use a simple method to quantize it so that we can apply ICD. Given a dataset, we find $v_{\min}$, the minimum feature value in the training set. We also find $v_{\max}$, which is the 97.5-th percentile of all training feature values.[2] A feature value v is mapped (quantized) to an integer in $[0\ \bar{v}]$ as follows

$$v \rightarrow (\text{int})\,(\bar{v} \times (v - v_{\min})/(v_{\max} - v_{\min}))\,. \tag{13}$$

[1] In practice we do not need to update all these $d\bar{v}$ numbers. If $\bar{v}_j$ is the maximum feature value in dimension j, then we only need to update $T_{j,k}$, $k = 1, \ldots, \bar{v}_j$ for the j-th dimension. We usually observe that $\bar{v}_j \ll \bar{v}$.

[2] We do not use the maximum feature value as $v_{\max}$ because in computer vision it may have an artificial mode at the largest feature value. For example, in the densely sampled bag of visual words model, possibly more than half of the image patches will be mapped to the visual word that corresponds to a uniform image region.

With this simple quantization strategy, we can apply ICD to a much broader range of problems (e.g., whose feature vectors are not natural histograms and have negative feature values).

Choosing $\bar{v}$ is a very important decision. The number of quantization bins, $\bar{v}$, not only affects the training time and storage requirements. It is also directly related to the accuracy of trained classifiers. Obviously a small $\bar{v}$ value will result in low accuracy. However, it is adverse to have a large $\bar{v}$. A large $\bar{v}$ will eventually cause over-fitting and uses more memory and CPU cycles.

We experimented with $\bar{v} = 50$, 100, and 200. In our experiments different problems acquired best accuracies at different $\bar{v}$ values. However, $\bar{v} = 100$ achieved a fair balance between the memory/computation cost and classification accuracy across almost all the datasets. We use $\bar{v} = 100$ if quantization is needed, and if we do not explicitly specify $\bar{v}$ otherwise.

The default value for the C parameter in LIBLINEAR is 1, and feature vectors are usually normalized to the range $[-1\ 1]$. In ICD, κ_{HI} usually generates much large kernel values. Consequently, we choose $C = 10^{-3}$ as the default value for ICD.

4 Experimental Results

We conducted 4 sets of experiments to test various aspects of the ICD algorithm. First we compare ICD with two recently proposed fast HIK SVM training algorithm (Sec. 4.1). We then test ICD on a large scale pedestrian detection dataset (Sec. 4.2). The third set of experiments deal with three different object and scene recognition problems in computer vision (Sec. 4.3). Finally, we show that when cross-validation based SVM parameter selection is infeasible for huge datasets, ICD achieves both faster speed and higher accuracies comparing with linear and RBF kernels, using their default parameter settings (Sec. 4.4). Our empirical results show that ICD is very robust to the C parameter in practice.

Before applying ICD, we use Eqn. 13 to quantize a problem if necessary. We set $\bar{v} = 100$ if not otherwise specified. The one versus rest strategy is used for multi-class problems. We use the default value $C = 10^{-3}$ whenever ICD is used.

4.1 Comparing with PWLSGD and SIKMA

PWLSGD [12] and SIKMA [18] are two recent stochastic gradient descent (SGD) methods for fast training of HIK SVM. In this section we compare ICD with these two, using the software and datasets provided with [12] and [18].

Note that in PWLSGD and SIKMA, we use the SVM parameters that are carefully chosen using cross validation by their corresponding authors. While in ICD we simply use the default value $C = 10^{-3}$. PWLSGD provides sample data on Caltech 101 [7]. We report in Table 1(a) the results when 15 training and testing examples are used in each category. The SIKMA software provides sample data on the PASCAL VOC 2007 images [6]. The comparison results are reported in Table 1(b). SGD methods yield different results on the same dataset

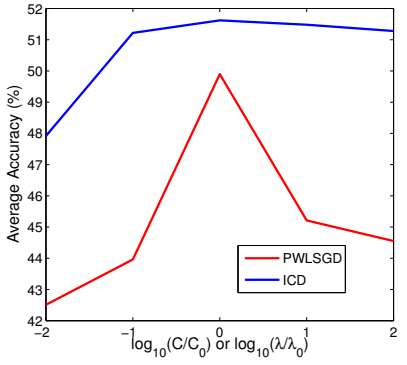
Table 1. Comparing training time and accuracy of HIK SVM methods

ICD		PWLSGD		LIBLINEAR	
56.5	**51.62%**	188.2	49.90%	43.3	48.05%

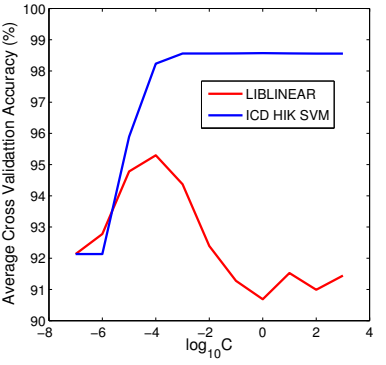
(a) Comparing ICD with PWLSGD

ICD		SIKMA		LIBLINEAR	
9.2	**97.21%**	13.2	96.97±.19%	1.9	96.00%

(b) Comparing ICD with SIKMA



(a) Result on Caltech 101



(b) Result on INRIA

Fig. 1. Effect of different SVM parameters

in multiple runs. Since SIKMA does not fix the seed of its random number generator, we report its average result in 5 runs. Following the setup of SIKMA, we set $\bar{v} = 50$ for this problem.

For every method, we show the training time (in seconds) followed by the classification accuracy. As shown in Table 1(a) and 1(b), ICD not only reduces training time by a large percentage, it also has higher classification accuracies, despite the fact that we do not tune the C parameter in ICD. An additional comparison with LIBLINEAR [8] is provided (with the default settings of LIBLINEAR). The proposed method enjoys higher accuracy with a reasonable amount of increase in training time. For example, on the Caltech 101 dataset, ICD only uses 30% more training time than LIBLINEAR (Table 1(a)).

One attractive property of ICD is that empirically it is not sensitive to SVM parameters. Let $\lambda_0 (= 0.0015)$ and $C_0 (= 0.001)$ denote the SVM parameters used in Table 1 for PWLSGD and ICD respectively. In Fig. 1a we show Caltech 101 results of different C and λ values where $\log_{10}(C/C_0)$ or $\log_{10}(\lambda/\lambda_0)$ ranges from -2 to 2. ICD has high accuracy at the default value C_0, and its accuracy is stable with larger C values. However, large variations are observed for PWLSGD. Time consuming cross-validation based parameter selection is needed for PWLSGD to choose an appropriate λ, but in ICD it is not necessary to choose C. ICD has stable accuracies when $C \geq 10^{-4}$. We observe in Sec. 4.2 again that ICD is robust to C.

Table 2. Results on INRIA pedestrian

	Time	Accuracy	Iterations
ICD	160 s	**98.56**%	21.4
LIBLINEAR	681 s	90.69%	1000

4.2 Pedestrian Detection

Next we use the INRIA pedestrian dataset [5] to evaluate ICD in a large scale vision problem. We use the 256 dimensional CENTRIST [20] visual descriptor as our base feature descriptor. A 108×36 image patch is divided into 9×4 blocks. Any neighboring 2×2 blocks are formed into a super-block. The concatenation of CENTRIST in all super-blocks generates a feature vector that has 6,144 dimensions. This feature vector is a natural histogram with $\bar{v} = 352$. We evaluate the SVM training algorithms in a "hard" dataset that is the result of bootstrapping the INRIA dataset (using the procedures in [5]). There are 30,711 examples. This dataset is mostly dense, resulting in a large scale problem with approximately 82 million non-zero feature values.

We compare ICD with LIBLINEAR. Five-fold cross validation is applied. The total training time, average accuracy, and average number of iterations needed to finish the optimization are reported in Table 2. Default C values are used in both methods in Table 2.

One interesting observation from Table 2 is that ICD only takes about 24% of the training time of LIBLINEAR. This is related to the number of iterations which is required to terminate the SVM optimization. HIK SVM has higher discrimination capability than a linear SVM, and ICD usually requires a small number of iterations to converge in practice.[3] In summary, ICD scales well to large problems, and is particularly suitable when the feature vectors are natural histograms.

The effect of SVM parameters are studies in Fig 1b, where the C value ranges from 10^{-7} to 10^{3}, with step size 10. Linear SVM is sensitive to C, and an overly large C will lead to a lower accuracy. In this dataset HIK SVM is not sensitive to C: the accuracy increases with C and a large C value will not lead to a lower accuracy. The same phenomenon is observed in almost all datasets we tested in this paper.

It is also interesting to compare with general purpose SVM learners with linear, RBF, or the histogram intersection kernel. However, on this large scale dataset, general purpose SVM solvers (e.g., LIBSVM [2]) requires more than 10 hours to converge. This fact makes these solvers impractical for large problems.

4.3 Object and Scene Recognition

In this section we evaluate ICD in 3 more benchmark vision problems: Caltech 101 [7], 15 class scene recognition [9], and 8 class sport events [10]. Images

[3] LIBLINEAR terminates when the iteration number is 1000. So the actual iterations needed for convergence is higher than 1000 in this problem.

Table 3. Results on various vision problems

	caltech			scene			sports		
	Time	Acc	Acc(cv)	Time	Acc	Acc(cv)	Time	Acc	Acc(cv)
ICD	71.5	60.0%	59.9%	20.1	81.9%	82.0%	10.4	81.3%	81.5%
LIBSVM+HIK	66.8	54.6%	54.9%	40.1	81.6%	81.7%	10.7	81.0%	81.0%
LIBLINEAR	6.3	54.1%	54.2%	1.4	75.3%	75.5%	0.5	76.7%	78.5%
LIBSVM+LIN	65.2	51.3%	51.4%	33.5	76.8%	76.8%	8.5	78.8%	78.8%
LIBSVM+RBF	67.3	18.2%	53.4%	58.2	35.2%	79.6%	14.1	12.5%	76.5%

are represented using the bag of visual words model, and feature vectors are generated by libHIK [19] with k-means visual codebooks. The results from the first train/test split of libHIK are reported.[4]

Three kernel types (linear, RBF, and HIK) are compared. The features are quantized for ICD and LIBSVM+HIK, because we want these two methods to use exactly the same data. In Table 3, the first two columns for each dataset report the training time and accuracy of a method when we use the default SVM parameters. We also use training set cross validation to choose SVM parameters in the range $\log_{10} C \in [-5 \;\; 3], \log_{10} \gamma \in [-5 \;\; -1]$, whose accuracies are reported in the 'Acc(cv)' column.

In terms of classification accuracy, HIK has clear advantages over linear and RBF kernel types. ICD achieves slightly higher accuracies than the general purpose LIBSVM solver with HIK. The Caltech 101 dataset shows an exception where ICD has a large 5.4% advantage over LIBSVM. This might be due to the fact that there are 101 classes in this problem, while the one versus one strategy of LIBSVM is not suitable for handling large number of classes.

In terms of training time, LIBLINEAR trains much faster than other methods, while all other methods have comparable training time. It is worth noting that the feature vectors are $d = 6,200$ dimensional in these problems, while the training set size n ranges from 560 to 1515. Dual space algorithms (including the proposed method) is not effective while $d \gg n$. However, ICD still has approximately the same training speed as LIBSVM.

Again we observed the phenomenon that ICD is not sensitive to SVM parameters, since 'Acc(cv)' only has slight advantage over 'Acc' (the accuracy using default ICD SVM parameters). The robustness to SVM parameters is very attractive because cross validation parameter selection is infeasible for huge datasets, e.g., accuracy of the RBF kernel is heavily affected by SVM and kernel parameters.

4.4 Working with Non-histograms and Default SVM Parameters

Nowadays many problems are too large to perform cross validation for SVM parameter selection. Thus it is important to emphasize *high accuracies using the default SVM parameters*. In the final set of experiments, we will evaluate ICD

[4] Note that the Caltech 101 features are different than those used in Sec. 4.1.

Table 4. Properties of the non-histogram datasets

	train size	test size	dimension	#class
ijcnn1	4,990	91,701	22	2
shuttle	43,500	14,500	9	7
acoustic	78,823	19,705	100	3
rcv1	677,399	20,242	47,236	2

Table 5. Comparing performances using default SVM parameters

	ijcnn1		shuttle		acoustic		rcv1	
	Time	Accuracy	Time	Accuracy	Time	Accuracy	Time	Accuracy
ICD	0.6	94.82%	0.7	99.50%	137.2	82.98%	34.3	97.95%
LIBSVM+HIK	42.9	95.27%	3.7	99.57%	1362.0	83.12%	$> 50,000$	
LIBLINEAR	0.4	91.79%	0.8	92.35%	14.6	80.18%	6.4	97.96%
LIBSVM+LIN	48.4	92.12%	7.4	97.10%	1736.0	80.46%	$> 50,000$	
LIBSVM+RBF	60.1	92.79%	14.3	97.23%	1824.0	79.69%	$> 50,000$	

on problems that are not natural histograms and on huge datasets. The four problems we experimented with are chosen from the LIBSVM dataset collection: ijcnn1, shuttle, acoustic (combined), and rcv1 (binary). We choose these datasets whose features are not histograms, and whose sizes range from medium to huge. The particulars of these problems are collected in Table 4. We switched the training and testing set of the rcv1 problem so that we have a huge training set to test the scalability of ICD.

We compare with LIBLINEAR and LIBSVM using linear, RBF, and HIK. We use the default value $C = 1$ for LIBLINEAR and LIBSVM on the original feature vectors, and use $C = 10^{-3}$ on quantized versions. Experimental results are reported in Table 5.

One important observation is that on these datasets both HIK SVM classifiers (ICD and LIBSVM+HIK) achieve higher accuracies even when their feature vectors are not natural histograms. We also compare the two pairs of methods that use the same kernel. Although the LIBSVM+LIN solver sometimes have noticeable advantage over LIBLINEAR at the cost of much longer training time (e.g., in the shuttle problem), the proposed ICD method has almost the same accuracy as LIBSVM+HIK.

ICD, however, trains a lot faster than LIBSVM+HIK, and its speedup is related to size of the datasets. ICD is about 5 times faster than LIBSVM+HIK on the shuttle dataset. However, in the rcv1 dataset, the speedup is more than 3 orders of magnitude. For all three kernel types, the LIBSVM solver did not converge after 50,000 seconds when running the rcv1 problem. Thus LIBSVM's accuracies on this dataset is not available in Table 5.

LIBLINEAR is faster than ICD. However, the speedup is usually smaller than 10. Given the fact that the training time is smaller than 1 minute even in the rcv1 dataset, we believe that the proposed algorithm is preferable for its higher classification accuracies.

It is generally accepted that for problems with a large number of feature dimensions, linear SVM usually works as well as other more complex kernel types. Thus it is not surprising to observe that on the rcv1 dataset, ICD requires more training time than LIBLINEAR, while both methods have approximately the same classification accuracy. Our experiments on rcv1, though, further illustrate the scalability of ICD. In computer vision, we usually work with a medium dimensional feature vector (e.g., around 5000, smaller than that of rcv1). The experiments on rcv1 illustrate that ICD is able to handle even millions of training examples in computer vision problems.

5 Conclusions and Future Work

Our contributions are threefold. First, we prove that the histogram intersection kernel (HIK) is a positive definite kernel for non-negative real numbers. Second, we give the physical meaning of the computational method that accelerates the kernel evaluation of HIK. Based on this interpretation, we propose ICD, a fast, accurate, and scalable HIK SVM solver. Third, we empirically show that ICD is not sensitive to the C parameter in SVM, and achieve high accuracies using its default settings on huge datasets.

As a summary of the theoretical analyses and experimental results, we list the advantages and limitations of the proposed method (+ for advantages and - for limitations).

Speed (+). ICD trains much faster than general purpose SVM solvers. It also trains faster than two recent SGD based methods (PWLSGD and SIKMA). The testing speed has the same complexity as linear classifiers.

Insensitivity to C (+). Accuracy of ICD generally increases with the SVM parameter C. However, a large C does not lead to low accuracy. This empirical property is particularly attractive on huge datasets where cross validation parameter selection is infeasible.

Scalability (+). It scales easily to large problems, and is most efficient for problems with a medium number of feature dimensions and a huge number of training examples. Many vision problems fit into this category.

Accuracy (+). It has comparable accuracies to general purpose SVM solvers, and the PWLSGD or SIKMA HIK SVM solver.

Simplicity (+). There is only 1 parameter in ICD (C), and the default value $C = 10^{-3}$ works well in most problems. It is also a deterministic algorithm, producing the same result on multiple runs. There is no need to re-encode input data.

Storage (-). The table T increases the storage cost, especially when d is large, and when the problem contains a large number of classes.

Quantization (-). The need for quantization introduces additional costs (although this cost is small), and the quantized version does not guarantee higher accuracy than linear SVM in a few cases (e.g., rcv1).

There are possible directions to address certain limitations of the proposed method. For example, we can make the table T sparse, which will answer the

storage problem at the cost of a reasonable increase in training and testing time. A rule of thumb can be developed to automatically choose between a linear SVM or HIK SVM. Finally, we can explore adaptive quantization methods to achieve better quantized feature vectors (and higher accuracies).

References

1. Boughorbel, S., Tarel, J.P., Boujemaa, N.: Generalized histogram intersection kernel for image recognition. In: ICIP (2005)
2. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001), `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
3. Crammer, K., Singer, Y.: On the algorithmic implementation of multiclass kernel-based vector machines. JMLR 2, 265–292 (2001)
4. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, Cambridge (2000)
5. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR, vol. 1, pp. 886–893 (2005)
6. Everingham, M., Gool, L.V., Williams, C., Winn, J., Zisserman, A.: The PASCAL visual object classes challenge 2007 (VOC 2007) results. Tech. rep (2007)
7. Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training example: an incremental bayesian approach tested on 101 object categories. In: CVPR 2004, Workshop on Generative-Model Based Vision (2004)
8. Hsieh, C.J., Chang, K.W., Lin, C.J., Keerthi, S.S., Sundararajan, S.: A dual coordinate descent method for large-scale linear SVM. In: ICML, pp. 408–415 (2008)
9. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR, vol. II, pp. 2169–2178 (2006)
10. Li, L.J., Fei-Fei, L.: What, where and who? Classifying events by scene and object recognition. In: ICCV (2007)
11. Lowe, D.: Distinctive image features from scale-invariant keypoints. IJCV 60(2), 91–110 (2004)
12. Maji, S., Berg, A.C.: Max-margin additive classifiers for detection. In: ICCV (2009)
13. Maji, S., Berg, A.C., Malik, J.: Classification using intersection kernel support vector machines is efficient. In: CVPR (2008)
14. Odone, F., Barla, A., Verri, A.: Building kernels from binary strings for image matching. IEEE Trans. Image Processing 14(2), 169–180 (2005)
15. Ojala, T., Pietikäinen, M., Mäenpää, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE TPAMI 24(7), 971–987 (2002)
16. Shalev-Shwartz, S., Singer, Y., Srebro, N.: Pegasos: Primal estimated sub-gradient solver for svm. In: ICML, pp. 807–817 (2007)
17. Swain, M.J., Ballard, D.H.: Color indexing. IJCV 7(1), 11–32 (1991)
18. Wang, G., Hoiem, D., Forsyth, D.: Learning image similarity from flickr groups using stochastic intersection kernel machines. In: ICCV (2009)
19. Wu, J., Rehg, J.M.: Beyond the euclidean distance: Creating effective visual codebooks using the histogram intersection kernel. In: ICCV (2009)
20. Wu, J., Rehg, J.M.: CENTRIST: A visual descriptor for scene categorization. Tech. Rep. GIT-GVU-09-05, GVU Center, Georgia Institute of Technology (2009)