Hash-Based Support Vector Machines Approximation for Large Scale Prediction

Saloua LITAYEM
saloua.litayem@inria.fr
INRIA Paris-Rocquencourt, France

Alexis JOLY
alexis.joly@inria.fr
INRIA Sophia-Antipolis, France

Nozha BOUJEMAA
nozha.boujemaa@inria.fr
INRIA Saclay, France

Abstract

How-to train effective classifiers on huge amount of multimedia data is clearly a major challenge that is attracting more and more research works across several communities. Less efforts however are spent on the counterpart scalability issue: how to apply big trained models efficiently on huge non annotated media collections ? In this paper, we address the problem of speeding-up the prediction phase of linear Support Vector Machines via Locality Sensitive Hashing. We propose building efficient hash-based classifiers that are applied in a first stage in order to approximate the exact results and filter the hypothesis space. Experiments performed with millions of one-against-one classifiers show that the proposed hash-based classifier can be more than two orders of magnitude faster than the exact classifier with minor losses in quality.

1 Introduction

With the rapid development of information acquisition technology, we have witnessed an explosive growth in the scale of shared multimedia data collections. The need for automatic tools to annotate and structure this huge data is still a big challenge although consistent progress has been made in the last decade [5, 16].

The good news with the big-data shift paradigm is that it has also favored the emergence of very large annotated image and video datasets that can be used to train real-world classifiers with a realistic number of concepts [5, 17]. How-to efficiently train classifiers on such huge amounts of data is clearly an important challenge attracting more and more research studies across several communities [1, 6, 11]. But less efforts are spent on the counterpart scalability issue: how can big trained models, involving large numbers of individual classifiers, be efficiently applied on huge non annotated media collections ? In the context of multi-class problems with a very high number of categories and the increasing use of multiple features, kernels and bagging strategies, the number of individual classifiers to be applied to each media item might actually reach millions very quickly. Cloud computing will certainly have a major role to play in applying millions of classifiers on billions of web images. Notably because such prediction phase have low computational complexities (typically linear in the number of documents). Nevertheless, any algorithmic efficiency improvements might help

http://dx.doi.org/10.5244/C.26.86

to save considerable resources.
In this work, we are specifically interested in speeding-up the prediction phase of Linear Support Vector Machines. Previous research works on scalable SVMs has mainly been dedicated to the training phase with the objective of reducing both the processing time and the memory requirement of the solver [7, 20]. Several studies have also proposed methods for improving multi-class SVM efficiency for a large number of categories [3, 8, 9, 14, 19]. In [19], for instance, the authors proposed a method to reduce the model size in the case of text categorization and a cache-efficient implementation of multi-class linear SVMs based on a shrunk model. In [8], Gao et al. proposed a fast multi-class evaluation method exploiting the hierarchical structure in the label space by organizing a set of binary classifiers in a hierarchy and pruning a subset of confusing classes in each node. In [15], authors proposed a locality-sensitive classifier preserving the locality characteristic with improved generalization ability on the global feature space. It models the local regions with smaller intra-class variation by exploring the local correlation structure. Closer to our work, an efficient search method using LSH was proposed in [10] with the objective of efficiently solving hyperplane queries (i.e finding the closest feature points to the hyperplane) in the context of active learning. The work presented in this paper rather uses LSH to build efficient hash-based classifiers approximating any linear SVM and converging to the exact classification performances. We also introduce a filter-and-refine strategy for solving large-scale multi-class problems and show that our method can be more than two orders of magnitude times faster than the exact linear classifier. The remainder of this paper is organized as follows. Section 2 describes formally our hash-based SVM approximation method and discusses the performance analysis. In Section 3, we specify our proposed filter-and-refine strategy for approximating large scale one-against-one multi-class SVMs. Experiments are reported in section 4. Finally we conclude in Section 5.

2 Hash-based SVM approximation

Let $h(\mathbf{x})$ be a trained linear SVM classifier defined as

$$h(\mathbf{x}) = sgn\left(\omega.\mathbf{x} + b\right) \tag{1}$$

We suppose that all features $\mathbf{x} \in \mathbb{R}^d$ are L_2-normalized, so that $\|\mathbf{x}\| = 1$.
In addition, let us denote as $\mathcal{F}$, a family of binary hash functions $f : \mathbb{R}^d \to \{-1, 1\}$ such that:

$$f(\mathbf{x}) = sgn\left(\mathbf{w}.\mathbf{x}\right)$$

if $\mathbf{w} \in \mathbb{R}^d$ is a random variable distributed according to $p_w = \mathcal{N}(0, \mathbf{I})$, we get the popular LSH function family sensitive to the inner product [4, 13]. In this case, for any two points $\mathbf{q}, \mathbf{v} \in \mathbb{R}^d$ we have:

$$Pr[f(\mathbf{q}) = f(\mathbf{v})] = 1 - \frac{1}{\pi} cos^{-1}\left(\frac{\mathbf{q}.\mathbf{v}}{\|\mathbf{q}\|\,\|\mathbf{v}\|}\right) \tag{2}$$

Note that any feature $\mathbf{x}$ can be hashed into a D-length binary hash code $\mathbf{F}_D(\mathbf{x})$ by concatenating D hash functions from $\mathcal{F}$.
Now our goal is to build an approximation of the classifier $h(\mathbf{x})$ by hashing both ω (the normal to the hyperplane of the classifier) and the features $\mathbf{x}$ to be classified. The basic idea is that the inner product between $\mathbf{x}$ and ω can actually be estimated by a Hamming distance between their respective hash codes $\mathbf{F}_D(\mathbf{x})$ and $\mathbf{F}_D(\omega)$, each being composed of D binary

hash functions in $\mathcal{F}$.
More formally, let us denote this Hamming distance by $d_H(\mathbf{F}_D(\mathbf{x}), \mathbf{F}_D(\omega))$. According to equation 2, we know that the collision probability $p(\mathbf{x}, \omega)$ between $f(\mathbf{x})$ and $f(\omega)$ is equal to:

$$p(\mathbf{x}, \omega) = Pr[f(\mathbf{x}) = f(\omega)] = 1 - \frac{1}{\pi} cos^{-1}\left(\frac{\mathbf{x}.\omega}{\|\mathbf{x}\| \|\omega\|}\right) \tag{3}$$

so that $d_H(\mathbf{F}_D(\mathbf{x}), \mathbf{F}_D(\omega))$ is a random variable distributed according to a binomial distribution $B(\bar{p}, D)$ where the success probability $\bar{p}$ of each Bernouilli trial is equal to $1 - p(\mathbf{x}, \omega)$. The expectation of d_H is therefore equal to $\mu_H = D.(1 - p(\mathbf{x}, \omega))$ and an estimator of $p(\mathbf{x}, \omega)$ can be computed as:

$$\hat{p}(\mathbf{x}, \omega) = 1 - \frac{d_H(\mathbf{F}_D(\mathbf{x}), \mathbf{F}_D(\omega))}{D} \tag{4}$$

Now that we have an estimation of the collision probability between $\mathbf{x}$ and ω and still according to equation 3, we can estimate their normalized inner product $\kappa(\mathbf{x}, \omega) = \frac{\mathbf{x}.\omega}{\|\mathbf{x}\|\|\omega\|}$ by:

$$\hat{\kappa}(\mathbf{x}, \omega) = cos(\pi(1 - \hat{p}(\mathbf{x}, \omega))) \tag{5}$$

Since the main hypothesis of our method was that $\|\mathbf{x}\| = 1$, we can now derive an estimation of our SVM classifier $h(\mathbf{x})$ as:

$$\hat{h}(\mathbf{x}) = sgn(\hat{\kappa}(\mathbf{x}, \omega) \|\omega\| + b) \tag{6}$$

$$\hat{h}(\mathbf{x}) = sgn\left(cos(\pi.(1 - \hat{p}(\mathbf{x}, \omega))) + \frac{b}{\|\omega\|}\right) \tag{7}$$

$$\hat{h}(\mathbf{x}) = sgn\left(cos^{-1}\left(\frac{-b}{\|\omega\|}\right) - \pi.(1 - \hat{p}(\mathbf{x}, \omega))\right) \tag{8}$$

and finally

$$\hat{h}(\mathbf{x}) = sgn\left(\frac{D}{\pi} cos^{-1}\left(\frac{-b}{\|\omega\|}\right) - d_H(\mathbf{F}_D(\mathbf{x}), \mathbf{F}_D(\omega))\right) \tag{9}$$

So that the initial linear SVM $h(\mathbf{x})$ can be approximated by a simple range query in the Hamming cube, centred around the hash code of ω.

Definition - Hash based SVM classifier
For any linear SVM classifier $h(\mathbf{x}) = sgn(\omega.\boldsymbol{x} + b)$, and a hash function family $\mathcal{F}$, we define a Hash-based SVM classifier as :

$$\begin{cases} \hat{h}(\mathbf{x}) = sgn\left(r_{\omega,b} - d_H(\mathbf{F}_D(\mathbf{x}), \mathbf{F}_D(\omega))\right) \\ r_{\omega,b} = \frac{D}{\pi} cos^{-1}\left(\frac{-b}{\|\omega\|}\right) \end{cases} \tag{10}$$

Theorem - Convergence of a Hash based SVM classifier
The classification results of a hash based SVM classifier $\hat{h}(\mathbf{x})$ converge to the results of the exact SVM classifier $h(\mathbf{x})$ as the number D of binary hash functions tends to infinity.

Proof - Let us first suppose that $h(\mathbf{x}) = -1$ so that $\omega.\mathbf{x} < -b$ and consequently, according to equations 3 and 10,

$$p(\mathbf{x},\omega) < 1 - \frac{1}{\pi} cos^{-1}\left(\frac{-b}{\|\omega\|}\right) \Leftrightarrow r_{\omega,b} < D(1 - p(\mathbf{x},\omega)) \tag{11}$$

On the other hand, the probability $p_{mis} = Pr\left[\hat{h}(\mathbf{x}) \neq h(\mathbf{x})\right]$ of a misclassification is equal to

$$p_{mis} = Pr\left[\hat{h}(\mathbf{x}) = 1\right] = Pr\left[d_H\left(\mathbf{F}_D(\mathbf{x}),\mathbf{F}_D(\omega)\right) < r_{\omega,b}\right] = F_H(\lfloor r_{\omega,b} \rfloor)$$

where F_H is the cumulative distribution function of the discrete random variable d_H. Since d_H is distributed according to a binomial distribution $B(1 - p(\mathbf{x},\omega), D)$, F_H can be represented in terms of the regularized incomplete beta function

$$F_H(\lfloor r_{\omega,b} \rfloor) = I_{p(\mathbf{x},\omega)}\left(D - \lfloor r_{\omega,b} \rfloor, \lfloor r_{\omega,b} \rfloor + 1\right)$$

And since $\lfloor r_{\omega,b} \rfloor < D(1 - p(\mathbf{x},\omega))$ (cf. equation 11), an upper bound for the lower tail of the cumulative distribution function F_H can be derived according to Hoeffding's inequality

$$F_H(\lfloor r_{\omega,b} \rfloor) \leq \frac{1}{2} exp\left(-2\frac{\left(D(1-p) - \lfloor r_{\omega,b} \rfloor\right)^2}{D}\right)$$

so that

$$p_{mis}(D) \leq \frac{1}{2} exp\left(-\alpha_1 D - \alpha_2 \frac{1}{D} - \alpha_3\right)$$

with $\alpha_i > 0, \forall i$. Finally, the misclassification probability tends to zero when D tends to infinity:

$$\lim_{D\to\infty} p_{mis}(D) = 0$$

which proves the convergence theorem for the case $h(\mathbf{x}) = -1$.

A similar demonstration leads to the same conclusion if we now suppose that $h(\mathbf{x}) = 1$. In this case, it is easier to reformulate the hash-based SVM classifier as:

$$\hat{h}(\mathbf{x}) = sgn\left(\bar{d_H}\left(\mathbf{F}_D(\mathbf{x}),\mathbf{F}_D(\omega)\right) - (D - r_{\omega,b})\right)$$

where $\bar{d_H}\left(\mathbf{F}_D(\mathbf{x}),\mathbf{F}_D(\omega)\right) = D - d_H\left(\mathbf{F}_D(\mathbf{x}),\mathbf{F}_D(\omega)\right)$ is the Hamming distance to the two's complement of $\mathbf{F}_D(\omega)$. So that $\bar{d_H}$ is distributed according to a binomial distribution $B(p(\mathbf{x},\omega), D)$ rather than $B(1 - p(\mathbf{x},\omega), D)$. The probability of a misclassification is then equal to

$$p_{mis} = Pr\left[\hat{h}(\mathbf{x}) = -1\right] = Pr\left[\bar{d_H}\left(\mathbf{F}_D(\mathbf{x}),\mathbf{F}_D(\omega)\right) < D - r_{\omega,b}\right] = \bar{F_H}(D - \lceil r_{\omega,b} \rceil)$$

where $\bar{F_H}$ is the cumulative distribution function of the binomial discrete random variable $\bar{d_H}$. On the other hand, $h(\mathbf{x}) = 1$, now implies that $D - \lceil r_{\omega,b} \rceil \leq Dp(\mathbf{x},\omega)$ and we can therefore still apply Hoeffding's inequality leading in this case to:

$$p_{mis}(D) \leq \frac{1}{2} exp\left(-2\frac{\left(Dp - (D - \lceil r_{\omega,b} \rceil)\right)^2}{D}\right)$$

and finally

$$\lim_{D\to\infty} p_{mis}(D) = 0$$

2.1 Performance analysis

Applying a Hash-based SVM classifier $\hat{h}(\mathbf{x})$ with a brute-force scan instead of the exact classifier $h(\mathbf{x})$ does not change the prediction complexity, which is still $O(N)$ in the number of images to classify. Performance gains are more related to memory usage and the overall speed when a very large number of classifiers have to be applied simultaneously (which is often the case when dealing with a large number of classes). Within our implementation, the space requirement of a single exact classifier is actually $S_{SVM} = 12 + 8d$ bytes when using a double precision for ω and b. On the other hand, the space requirement of a single Hash based SVM classifier is $S_{HBMS} = 6 + \frac{D}{8}$ bytes. With the typical values $d = 1000$ and $D = 256$ bits used in our experiments, the memory usage required by Hash based SVM classifiers is about 200 times lower than the exact classifiers.
The second main advantage is to speed up the computation of the classification function. A Hamming distance on typically $D = 256$ bits can be much faster than an inner product on high-dimensional data with a double precision (particularly when benefiting from *pop-count* assembler instructions).

3 Filter-and-refine strategy

In this section, we propose a filter-and-refine strategy for approximating a one-against-one linear multi-class SVM with a large number of categories and a large dataset to be classified. Let us first introduce some notations. We consider a dataset $\mathcal{X}$ of N feature vectors $\mathbf{x}$ in $\mathbb{R}^d$ that needs to be classified efficiently across a set $\mathbf{C}$ of K classes c_k. We then consider a one-against-one linear multi-class SVM $H(\mathbf{x})$ that is assumed to have been trained to solve this classification problem. $H(\mathbf{x})$ is defined as

$$H(\mathbf{x}) = \underset{c_k \in \mathbf{C}}{\arg\max} \# \left\{ h_{k,j} \mid h_{k,j}(\mathbf{x}) = 1 \right\} \tag{12}$$

where $h_{k,j}$ represents the $K(K-1)/2$ one-against-one classifiers (c_k vs c_j) defined by:

$$h_{k,j}(\mathbf{x}) = sgn\left(\omega_{k,j}.\mathbf{x} + b_{k,j}\right) \tag{13}$$

Thanks to our hash-based SVM approximation method, each $h_{k,j}$ can be approximated by an efficient hash-based binary classifier $\hat{h}_{k,j}(\mathbf{x})$ such that:

$$\hat{h}_{k,j}(\mathbf{x}) = sgn\left(\hat{\kappa}(\mathbf{x}, \omega_{\mathbf{k,j}}) \left\| \omega_{\mathbf{k,j}} \right\| + b\right) \tag{14}$$

And finally, a hash-based multi-class SVM (HBMS) can be computed as

$$\hat{H}(\mathbf{x}) = \underset{c_k \in \mathbf{C}}{\arg\max} \# \left\{ \hat{h}_{k,j} \mid \hat{h}_{k,j}(\mathbf{x}) = 1 \right\} \tag{15}$$

3.1 Filtering step

Now we propose to use the approximate classifier $\hat{H}(\mathbf{x})$ as a filtering tool rather than as a strict classifier. Instead of keeping only the best class as in equation 15, we keep the top-k classes as potential solutions for the refinement step. We denote by $\mathbf{C}_k(\mathbf{x})$ the set of k classes returned by this filtering step for all $\mathbf{x} \in \mathbf{X}$. Note that k is supposed to be relatively small compared to K so that

$$k(k-1) << K(K-1) \tag{16}$$

3.2 Refinement step

Now that we have a reduced set of hypotheses for each $\mathbf{x} \in \mathbf{X}$, we can refine the classification by applying an exact multi-class SVM on the remaining classes. This is simply done by using the $k(k-1)/2$ one-against-one classifiers computed in the initial classifier $H(\mathbf{x})$. Our final approximate classifier can therefore be expressed as:

$$\hat{H}_k(\mathbf{x}) = \underset{c_m \in \mathbf{C}_k}{\arg\max} \# \left\{ h_{m,j} \mid \hat{h}_{m,j}(\mathbf{x}) = 1 \right\} \tag{17}$$

Note that the complexity of this step is $O(k(k-1))$ whereas the complexity of both the initial classifier $H(\mathbf{x})$ and the hash-based classifier $\hat{H}(\mathbf{x})$ is $O(K(K-1))$. Therefore, if we have $k(k-1) << K(K-1)$ then the cost of the refinement step is negligible.

3.3 Extension to bagging

We propose to use ensemble classifiers with bagging in the binary-class-level. Indeed, benefiting from bootstrapping and aggregation, bagging [2] lowers both the variance and the bias component of the error. Tailored to SVM [12] it has been shown that notably in the case of multi-class classification SVM ensembles outperform a single SVM in terms of accuracy [18]. For each binary classifier involving c_k vs c_j classes, we consider B bootstrap independently trained SVMs aggregated via a majority voting technique. Let $\hat{H}_i, i \in \{1,2,...,B\}$ be the decision function of the i^{th} hash-based multi-class SVM in the ensemble and $p^*_{k,j} \in \{c_k, c_j\}$

$$p^*_{k,j} = \underset{c_p \in \{c_k, c_j\}}{\arg\max} \# \left\{ i | \hat{H}_i(\mathbf{x}) = c_p \right\} \tag{18}$$

the winning class among the bootstrap samples. Then, our ensemble approximates classifier can be expressed as:

$$\hat{H}_{bagging}(\mathbf{x}) = \underset{p^*_{k,j} | \{c_k, c_j\} \in C}{\arg\max} \# \{ p^*_{k,j} \} \tag{19}$$

4 Experiments

4.1 Experimental setup

Experiments were performed on an Intel Xeon 5560 CPU (2.8 GHz) with 46G of main memory. We used the ImageNet-BOF dataset [5] provided within the ImageNet PASCAL VOC Large Scale Visual Recognition 2010 Challenge. It consists of a set of 1.2 M 1000-dimensional Bags of SIFT Features annotated with 1000 class labels. Linear SVM classifiers were trained using the efficient solver of the popular LIBLINEAR[1] library with the default parameter value C = 1. We performed the training of all one-against-one classifiers with 50 random examples per category over 20 bootstrap samples (resulting in a total of about 10 million classifiers). The set $\mathcal{X}$ of $N = 50K$ test pictures to be classified is the one provided in the testbed. The effectiveness of the classifiers are measured according to the *mean accuracy*, i.e. the number of correctly classified images divided by N.

[1] http://www.csie.ntu.edu.tw/ cjlin/liblinear/

4.2 Convergence of Hash-based multi-class SVM

Figure 1 illustrates the convergence of Hash-based multi-class SVM classifiers (independently from the filter-and-refine strategy) for $K = 300$ random categories and 20 bootstrap samples. It shows that the accuracy of Hash-based multi-class classifiers converges to that of the exact one-against-one classifiers as the hash code's length increases. About $D = 4096$ bits are required to approximate well individual classifiers well. With this setting, the average prediction time of the hash-based SVM is about 3.5 times faster than the exact linear SVM. As we will see below, smaller hash codes can be used with the filter-and-refine strategy.

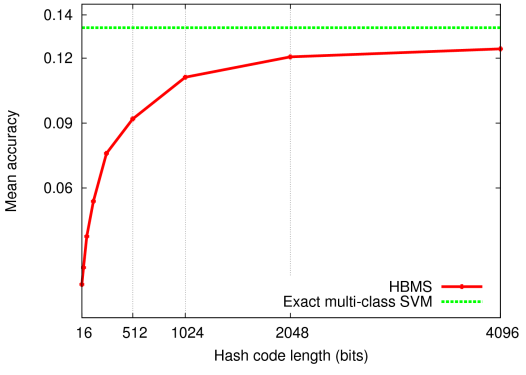

Figure 1: Exact vs Hash-based SVM

4.3 Filter-and-refine strategy evaluation

Figure 2 shows the convergence of the Hash-based multi-class SVM when using the filter-and-refine strategy (still with $K = 300$ random classes). It shows that the accuracy converges to that of the exact classifier when increasing either the number of bits or the number of filtered classes k. When $k = 30$ classes (i.e $k = K/10$) are passed to the refinement step, only 128 bits are sufficient to approximate the exact classifier very well. When $k = 3$ classes (i.e $k \approx K/100$), the classifier can be well approximated with 2048 bits. Figure 3 illustrates the mean accuracy of HBMS vs that of the exact multi-class classifier for a varying number of k classes and an increasing number of bits. We notice that HBMS outperforms the exact multi-class classifier and from $D = 512$ bits and typically $k = 9$ the accuracy does not improve anymore. Indeed, the accuracy improvement might be a result of a better generalization ability owing to refinement step with the best top-k classes obtained with our SVM approximation classifier in the filtering step. According to results, small hash codes and a small number of classes can be used to approximate the exact classifier well.

4.4 Scalability evaluation

Table 1 shows the impact of the number of classes on the accuracy of the exact multi-class SVM compared to that of the hash-based multi-class SVM with different hash code sizes and numbers of classes used in the refinement step. Let $r = k/K$ denotes the kept classes rate. The classes used for each value of K were selected at random from the 1000 initial classes. We ideally would need to average these results over several samples of K classes but the

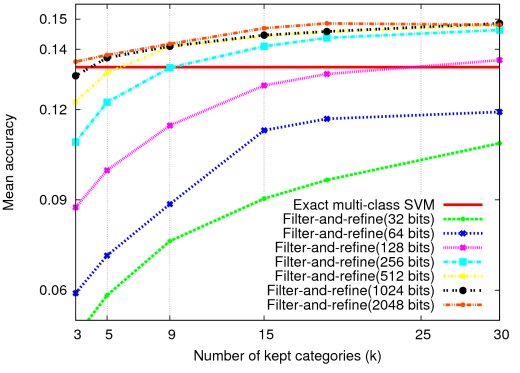

Figure 2: Exact Multi-class SVM vs Filter-and-refine method

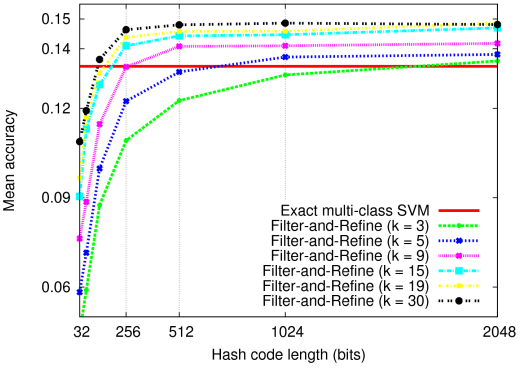

Figure 3: Exact Multi-class SVM vs Filter-and-refine method

expensive computation time of the exact SVM makes it a tricky task. We were not able to achieve one complete run with the exact classifier on more than 600 classes as exact SVM classifiers models could not fit in memory. Anyway, we believe the table is still convincing that the increasing number of categories does not affect more the accuracy of the Hash-based multi-class SVM than that of the exact classifier. We performed the prediction on the total testing set for a selected number of classes. In the available time, for more than 500 categories we used 5000 randomly selected images from the testing set and 10 images per category for the 1000 categories.

Figure 4 shows the average processing time per image of the hash-based filtering step for an increasing number of classes and varying hash code sizes. The figure clearly shows that the hash-based classifiers are much more efficient than the exact classifiers. For 600 classes, the filtering time is 21 times faster with 4096 bits and 221 times faster with 256 bits. This big gap in the processing time is not only due to the faster computation of the Hamming distances. It is also related to the larger memory requirement of the exact Multi-class SVM, which leads to many cache outputs and finally to swapping for large number of classes. The cost of the refinement step mainly depends on the number k of filtered classes. Experiments confirmed that its processing time is roughly $\left(\frac{K}{k}\right)^2$ times faster than the exact multi-class

Nb. categories	Exact SVM	Filter-and-Refine			
		$D = 256$ bits $r = 0.1$	$D = 512$ bits $r = 0.01$	$D = 1024$ bits $r = 0.03$	$D = 2048$ bits $r = 0.01$
50	0.3208	0.3312	0.2492	0.3344	0.2964
100	0.2314	0.2426	0.1722	0.2376	0.2128
200	0.1597	0.1722	0.1403	0.1713	0.1635
300	0.1341	0.1464	0.1226	0.1403	0.1359
500	0.1069	0.1149	0.1054	0.1169	0.1072
600	0.0922	0.1078	0.0926	0.1018	0.0990
1000	■	■	0.0693	0.0724	0.0745

Table 1: Filter-and-refine strategy accuracy vs exact multi-class SVM accuracy.

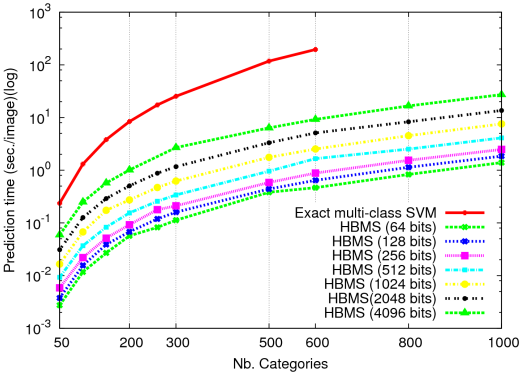

Figure 4: HBMS prediction time vs exact multi-class SVM prediction time

SVM. Figure 5 shows the average processing time per image for the filter-and-refine method for varying rates of passed classes to the refinement step and hash code lengths. To achieve very high accuracy with typically $k = K/10$ and $D = 256$ bits (still with 600 classes), the cost of the refinement step is preponderant and the whole filter-and-refine strategy is roughly 10 times faster. If moderate losses in quality are tolerated with typically $k = K/100$ and $D = 512$ bits, then the cost of the whole filter-and-refine strategy is roughly equal to the cost of the filtering step and the whole filter-and-refine strategy is 110 times faster.

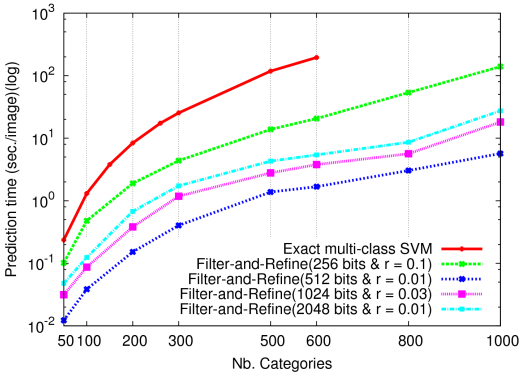

Figure 5: Filter-and-refine prediction time vs exact multi-class SVM prediction time

5 Conclusion and perspectives

To the best of our knowledge, this is the first time LSH has been used to build approximate versions of linear SVMs trained beforehand. As shown in our experiments, the resulting hash-based classifiers can be more than two orders of magnitude faster than exact linear multi-class SVMs. Future work will address the problem of indexing the hash-based classifiers rather than using a brute-force scan, notably in the context of hierarchical class labels. Another perspective is to extend this work to the kernel trick by using recent hash functions dedicated to kernelized feature spaces.

References

[1] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *NIPS*, 2007.

[2] Leo Breiman. Bagging predictors. *Mach. Learn.*, 1996.

[3] Jair Cervantes, Xiaoou Li, Wen Yu, and Javier Bejarano. Multi-class support vector machines for large data sets via minimum enclosing ball clustering. *Electrical and Electronics Engineering ICEEE 2007 4th International Conference on*, 2007.

[4] Moses S. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, 2002.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[6] Jia Deng, Alexander C. Berg, Kai Li, and Li Fei-Fei. What does classifying more than 10,000 image categories tell us? In *Proceedings of the 11th European conference on Computer vision: Part V*, ECCV'10, 2010.

[7] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 2008.

[8] Tianshi Gao and Daphne Koller. Discriminative learning of relaxed hierarchy for large-scale visual recognition. In *ICCV*, 2011.

[9] Shantanu Godbole, Sunita Sarawagi, and Soumen Chakrabarti. Scaling multi-class support vector machines using inter-class confusion. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002.

[10] Prateek Jain, Sudheendra Vijayanarasimhan, and Kristen Grauman. Hashing hyperplane queries to near points with applications to large-scale active learning. In *NIPS*, 2010.

[11] Thorsten Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006.

[12] Hyun Chul Kim, Shaoning Pang, Hong-Mo Je, Daijin Kim, and Sung Yang Bang. Support vector machine ensemble with bagging. In *Proceedings of the First International Workshop on Pattern Recognition with Support Vector Machines*, 2002.

[13] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, 2009.

[14] Ping Li, Anshumali Shrivastava, Joshua L. Moore, and Arnd C. König. Hashing algorithms for large-scale learning. In J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*. 2011.

[15] Guo-Jun Qi, Qi Tian, and T. Huang. Locality-sensitive support vector machine by exploring local correlation and global regularization. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, 2011.

[16] Alan F. Smeaton, Paul Over, and Wessel Kraaij. High level feature detection from video in trecvid: a 5-year retrospective of achievements. In *In Ajay Divakaran, editor, Multimedia Content Analysis, Theory and Applications*, 2008.

[17] Raphaël Troncy, Bartosz Malocha, and André T. S. Fialho. Linking events with media. In *Proceedings of the 6th International Conference on Semantic Systems*, 2010.

[18] Shi-Jin Wang, Avin Mathew, Yan Chen, Li-Feng Xi, Lin Ma, and Jay Lee. Empirical analysis of support vector machine ensemble classifiers. *Expert Syst. Appl.*, 2009.

[19] Jian xiong Dong, Ching Y. Suen, and Adam Krzyzak. Effective shrinkage of large multi-class linear svm models for text categorization. In *ICPR*, 2008.

[20] Hsiang-Fu Yu, Cho-Jui Hsieh, Kai-Wei Chang, and Chih-Jen Lin. Large linear classification when data cannot fit in memory. *ACM Trans. Knowl. Discov. Data*, 2012.