Hash-Based Support Vector Machines Approximation for Large Scale Prediction

Saloua Litayem
http://www-rocq.inria.fr/~litayem
INRIA Paris-Rocquencourt, France
Alexis Joly
www-sop.inria.fr/members/Alexis.Joly
INRIA Sophia-Antipolis, France
Nozha Boujemaa
http://pages.saclay.inria.fr/nozha.boujemaa
INRIA Saclay, France

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.

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

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

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 $f : \mathbb{R}^d \rightarrow \{-1, 1\}$ such that:

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

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. In this case, for any two points $\mathbf{q}, \mathbf{v} \in \mathbb{R}^d$ we have:

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

The basic idea of our Hash based SVM classifier 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 $\mathcal{F}$.

Definition - Hash based SVM classifier

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

$$\begin{cases} \hat{h}(\mathbf{x}) = sgn\left(r_{\omega,b} - d_H\left(\mathbf{F}_D(\mathbf{x}), \mathbf{F}_D(\omega)\right)\right) \\ r_{\omega,b} = \frac{D}{\pi} cos^{-1}\left(\frac{-b}{\|\omega\|}\right) \end{cases} \tag{3}$$

with $d_H\left(\mathbf{F}_D(\mathbf{x}), \mathbf{F}_D(\omega)\right)$ being the Hamming distance between $\mathbf{x}$ and ω.

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). 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).

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. 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

$$H(\mathbf{x}) = \underset{c_k \in \mathbf{C}}{\arg\max} \#\left\{h_{k,j} \mid h_{k,j}(\mathbf{x}) = 1\right\} \tag{4}$$

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

$$h_{k,j}(\mathbf{x}) = sgn\left(\omega_{k,j}.\mathbf{x} + b_{k,j}\right) \tag{5}$$

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:

$$\hat{h}_{k,j}(\mathbf{x}) = sgn\left(\hat{\kappa}(\mathbf{x}, \omega_{\mathbf{k,j}})\,\|\omega_{\mathbf{k,j}}\| + b\right) \tag{6}$$

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

$$\hat{H}(\mathbf{x}) = \underset{c_k \in \mathbf{C}}{\arg\max} \#\left\{\hat{h}_{k,j} \mid \hat{h}_{k,j}(\mathbf{x}) = 1\right\} \tag{7}$$

Figure 1 illustrates the mean accuracy of HBMS vs that of the exact multi-class classifier for a varying number of k kept classes in the filtering step and an increasing number of bits. 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.

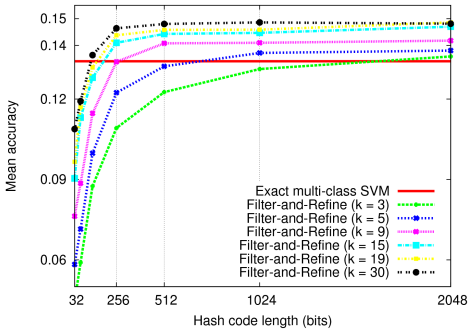

Figure 1: Exact Multi-class SVM vs Filter-And-Refine method

Figure 2 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. 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.

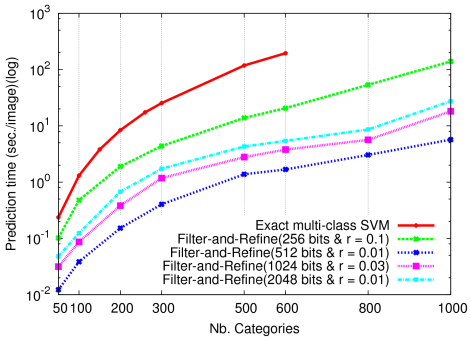

Figure 2: Filter-and-refine prediction time vs exact multi-class SVM prediction time