Hash-Based Support Vector Machines Approximation for Large Scale Prediction

Saloua Litayem
http://www-rocin.inria.fr/~litayem

Alexis Joly
www-sop.inria.fr/members/Alexis.Joly

Nozha Boujemaa
http://pages.saclay.inria.fr/nozha.boujemaa

INRIA Paris-Rocquencourt, France
INRIA Sophia-Antipolis, France
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 \( \hat{h}(x) \) be a trained linear SVM classifier defined as

\[
h(x) = \text{sgn}(\omega \cdot x + b)
\]

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

\[
f(x) = \text{sgn}(w \cdot x)
\]

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

\[
Pr[f(q) = f(v)] = 1 - \frac{1}{\pi} \cos^{-1}\left(\frac{q \cdot v}{\|q\| \|v\|}\right)
\]

The basic idea of our Hash based SVM classifier is that the inner product between \( x \) and \( \omega \) can actually be estimated by a Hamming distance between their respective hash codes \( F_D(x) \) and \( F_D(\omega) \), each being composed of \( D \) binary hash functions in \( \mathcal{F} \).

**Definition - Hash based SVM classifier**

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

\[
\hat{h}(x) = \begin{cases} 
\text{sgn}(r_{a,b} - d_D(F_D(x), F_D(\omega))) \\
\text{sgn}(\omega_{a,b} - \frac{D}{2} \cos^{-1}\left(\frac{a \cdot b}{\|a\| \|b\|}\right))
\end{cases}
\]

with \( d_D(F_D(x), F_D(\omega)) \) being the Hamming distance between \( x \) and \( \omega \).

Applying a Hash-based SVM classifier \( \hat{h}(x) \) with a brute-force scan instead of the exact classifier \( h(x) \) 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 \( X \) of \( N \) feature vectors \( x \) in \( \mathbb{R}^d \) that needs to be classified efficiently across a set \( C \) of \( K \) classes \( c_k \).

We then consider a one-against-one linear multi-class SVM \( H(x) \) that is assumed to have been trained to solve this classification problem. \( H(x) \) is defined as

\[
H(x) = \arg\max_{c_k \in C} \left\{ h_{k,j}(x) \right\}
\]

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

\[
h_{k,j}(x) = \text{sgn}(\omega_{k,j} \cdot x + b_k)
\]

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

\[
\hat{h}_{k,j}(x) \approx \text{sgn}(\hat{\omega}_{k,j} \cdot x + \hat{b}_j)
\]

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

\[
\hat{H}(x) = \arg\max_{c_k \in C} \left\{ \hat{h}_{k,j}(x) \right\}
\]

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 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.