Optimizing Circulant Support Vector Machines: the Exact Solution

Ramin Raziperchikolaei rraziperchikolaei@ucmerced.edu
Electrical Engineering and Computer Science
University of California, Merced
5200 Lake Rd, Merced, CA
Miguel Á. Carreira-Perpiñán mcarreira-perpinan@ucmerced.edu
Electrical Engineering and Computer Science
University of California, Merced
5200 Lake Rd, Merced, CA

Abstract

Binary hashing is an established approach for fast, approximate image search. The idea is to learn a hash function that maps a query image to a binary vector so that Hamming distances approximate image similarities. An important subproblem in binary hashing is to solve a set of independent classification problems, usually using support vector machines (SVMs). In this paper, we show that the hash function performs faster if we learn a set of circulant SVMs instead of the independent ones. Unlike the previously proposed algorithm that finds a suboptimal solution of the circulant SVMs, we show that the problem can be solved exactly and efficiently by casting it as a convex maximum margin classification problem on a modified dataset. We confirm experimentally that our approach solves the classification problem and the image search task better than the previous method.

1 Introduction

As the dataset of images continues to grow, searching for similar images becomes a more challenging problem. Binary hashing is a fast way to solve the similarity search problems approximately [4]. The main idea is to learn a hash function that maps high-dimensional images into binary codes and search for the similar images in the binary space. Representing images as binary vectors makes it possible to store large datasets with millions of images in the main memory of a single machine. To search for a query, one can create a hash table, indexed by the binary codes of the training images, and find the images inside a small Hamming distance of the query in $\mathcal{O}(1)$ [4].

The main goal in binary hashing is to learn a good hash function that can preserve the similarity between the points after mapping them to the binary space. To achieve this, different objective functions and optimization methods have been proposed [14, 6, 7, 2, 1, 9], usually based on hash functions of the form $\mathbf{h}(\mathbf{x}) = \operatorname{sgn}(\mathbf{W}\mathbf{x}) \in \{-1,+1\}^L$, where $\mathbf{x} \in \mathbb{R}^D$ is a D-dimensional feature vector representing the image. In most hashing papers, learning the hash function (the matrix $\mathbf{W}$) corresponds to solving L independent classification problems. Support Vector Machine (SVM) is the most popular classifier in the hashing literature [14, 6, 7, 2, 1, 9].

Note that the time needed to generate a binary vector for a query image is $\mathcal{O}(LD)$. As we increase the number of bits L, we can preserve the similarity better and hence improve the retrieval quality, but we need more time to generate the binary codes. A recent method, circulant binary embedding (CBE) [13], proposed to learn a hash function with a single circulant weight matrix, which gives better time and space complexity. As we show in section 2, using circulant weight matrices we need $\mathcal{O}(D \log D)$ to compute the binary codes and need $\mathcal{O}(D)$ to store the weight matrix.

To learn the circulant hash function, the classification problem turns into learning a set of circulant support vector machines, where the weight vector of each of the SVMs can be achieved by circularly shifting a single base vector of dimension D. In [13], the circulant SVMs are learned by doing optimization in the frequency domain. This has two important disadvantages. (1) It relaxes the classification problem into a regression problem (ignoring the sign function). (2) When $L < D$ (which is the case of interest), CBE adds $D - L$ zeros to the labels of each point and finds a suboptimal solution. This idea does not perform well in practice, as we will show in experiments.

OPTML 2017: 10th NIPS Workshop on *Optimization for Machine Learning* (NIPS 2017).

In this paper, we provide an algorithm that does learn the hash function's circulant matrix optimally even for $L < D$ bits. In section 3, we formulate the problem as a convex maximum margin classification problem, which can be solved exactly by training a single support vector machine on a suitably modified training set. In section 4 we show experimentally that this results in better classification accuracy and generalization, and better retrieval results in binary hashing.

2 Hashing with a circulant weight Matrix

We quickly review circulant matrices and explain their usefulness in binary hashing. A D-dimensional vector $\mathbf{w} = (w_0, w_1, \cdots, w_{D-1})$ is the basis for the $D \times D$ circulant matrix $\mathbf{W}$, which can also be regarded as a filter operating on the input image:

$$\mathbf{W} = \operatorname{circ}(\mathbf{w}) \equiv \begin{bmatrix} w_0 & w_{D-1} & \cdots & w_2 & w_1 \\ w_1 & w_0 & w_{D-1} & \cdots & w_2 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ w_{D-1} & w_{D-2} & \cdots & w_1 & w_0 \end{bmatrix}. \tag{1}$$

So the matrix $\mathbf{W}$ has only D (instead of D^2) free parameters and we only need $\mathcal{O}(D)$ to store it. Consider the hash function $\mathbf{h}(\mathbf{x}) = \operatorname{sgn}(\mathbf{Wx})$ where $\mathbf{W}$ is circulant. We show here that $\mathbf{h}(\mathbf{x})$ can be computed in $\mathcal{O}(D \log D)$ instead of D^2. Consider $\mathcal{F}(\cdot)$ as the discrete Fourier transform, and $\mathcal{F}^{-1}(\cdot)$ as the inverse discrete Fourier transform. Since $\mathbf{W}$ is circulant, the output of the hash function can be computed as [8]:

$$\mathbf{h}(\mathbf{x}) = \operatorname{sgn}(\mathbf{Wx}) = \operatorname{sgn}\left(\mathcal{F}^{-1}(\mathcal{F}(\mathbf{x}) \circ \mathcal{F}(\mathbf{w}))\right) \tag{2}$$

where $\mathcal{F}(\mathbf{x}) \circ \mathcal{F}(\mathbf{w})$ is the elementwise product of two vectors. Computing the sign and elementwise product takes $\mathcal{O}(D)$ and computing the discrete Fourier transform and the inverse Fourier transform (using the fast Fourier transform) takes $\mathcal{O}(D \log D)$. So the total time needed to generate the code for one input is $\mathcal{O}(D \log D)$.

If the hash function needs to generate $L < D$ bits, we only need the first L rows of $\operatorname{circ}(\mathbf{w})$, which we denote as $\operatorname{circ}(\mathbf{w})_L$. If we use the discrete Fourier transform, we first need to generate the D-bits codes and then use L of them, so the complexity remains $\mathcal{O}(D)$ space and $\mathcal{O}(D \log D)$ time. This is faster than directly computing $\mathbf{Wx}$ unless L is very small.

3 Circulant support vector machines

Assume we have N training points $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N) \in \mathbb{R}^{D \times N}$ in D-dimensional space. The goal is to learn a hash function $\mathbf{h}$: $\mathbb{R}^D \rightarrow \{-1, +1\}^L$ that maps D-dimensional points into the L-dimensional binary codes. We define $\mathbf{h}(\mathbf{x}) = \operatorname{sgn}(\mathbf{Wx} + \mathbf{b})$, where $\mathbf{W} \in \mathbb{R}^{L \times D}$ is the weight matrix and $\mathbf{b} \in \mathbb{R}^L$ is the bias. So learning the hash function corresponds to learning $\mathbf{W}$ and $\mathbf{b}$.

We first show how learning the hash functions appears with the form of a classification problem in hashing papers. Then, we describe our method to learn the optimal circulant weight matrix.

3.1 Learning the hash function by learning classifiers

In many hashing papers, learning the L-bits hash function involves solving L independent classification problems [14, 6, 7, 2, 1, 9], possibly iteratively. The main idea is to define N L-dimensional binary variables $\mathbf{Z} = (\mathbf{z}_1, \dots, \mathbf{z}_N) \in \{-1, +1\}^{L \times N}$, and define the objective over the binary codes $\mathbf{Z}$ instead of the hash function $\mathbf{h}$. After optimizing the objective over $\mathbf{Z}$, we need to learn the hash function given the codes. In [14, 6, 7], the hash function is learned a posteriori, and in [2, 1, 9], the algorithm iterates over learning codes and hash functions, which achieves better optima.

To learn the hash function given the codes, we need to solve the following problem:

$$\min_{\mathbf{W},\mathbf{b}} \sum_{l=1}^{L} \sum_{n=1}^{N} \left(\operatorname{sgn}\left(\mathbf{w}_l^T \mathbf{x}_n + b_l\right) - z_{ln}\right)^2 \tag{3}$$

where $\mathbf{w}_l^T$ is the lth row of $\mathbf{W}$, $\operatorname{sgn}\left(\mathbf{w}_l^T \mathbf{x}_n + b_l\right)$ gives the lth bit of the hash function, and $z_{ln} \in \{-1, +1\}$ is the lth bit of the nth training point. Since the rows of $\mathbf{W}$ are independent from each other, this problem can be solved by training L independent classifiers. For the lth problem, the (input,label) pairs is determined by ($\mathbf{X}$,$\mathbf{Z}_{l\cdot}$).

3.2 Using a circulant SVM classifier as the hash function

We explain our proposed method in this section. We assume that we have the binary codes $\mathbf{Z} \in \{-1,+1\}^{L\times N}$ and we try to learn the circulant matrix $\mathbf{W} = \operatorname{circ}(\mathbf{w})_L$ and the bias $\mathbf{b}$ that minimize the classification error of eq. (3). Since the weight matrix is circulant, the L classification problems are not independent: they all share the same weight values, but in different orders as shown in eq. (1).

To minimize the classification error, we consider the maximum margin formulation of SVMs. Consider $\mathbf{w}_l^T$ as the lth row of the matrix $\mathbf{W}$. The lth classification problem has the following form:

$$\min_{\mathbf{w}_l} \frac{1}{2}\|\mathbf{w}_l\|^2 + C\sum_{n=1}^{N}\xi_{ln} \quad \text{s.t.} \quad z_{ln}(\mathbf{w}_l^T\mathbf{x}_n + b_l) \geq 1-\xi_{ln},\ \xi_{ln}\geq 0,\ n=1,\dots,N$$

where z_{ln} and ξ_{ln} are the label and the slack variable of the nth point in the lth classification problem, $\mathbf{w}_l$ is the weight vector of the lth classifier and b_l is its bias. From eq. (1), each row of $\mathbf{W}$ is a permutation of the vector $\mathbf{w}$ (first column of $\mathbf{W}$). For this reason, we can write row l of $\mathbf{W}$ as $\mathbf{w}_l^T = \mathbf{w}^T\mathbf{P}_l$, where $\mathbf{P}_l \in \mathbb{R}^{D\times D}$ is a permutation matrix. Based on this formulation, we can rewrite the SVM formulation of the lth classification problem as:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}^T\mathbf{P}_l\|^2 + C\sum_{n=1}^{N}\xi_{ln} \quad \text{s.t.} \quad z_{ln}(\mathbf{w}^T\mathbf{P}_l\mathbf{x}_n + b_l) \geq 1-\xi_{ln}, \xi_{ln}\geq 0,\ n=1,\dots,N.$$

Since $\mathbf{P}_l^T\mathbf{P}_l = \mathbf{I}$, $\|\mathbf{w}^T\mathbf{P}_l\|^2 = \|\mathbf{w}\|^2$, so all L classification problems have the same margin term. Let us define $\mathbf{t}_{ln} = \mathbf{P}_l\mathbf{x}_n \in \mathbb{R}^D$. Since $\mathbf{P}_l$ is a permutation matrix, $\mathbf{P}_l\mathbf{x}_n$ does not change the values of $\mathbf{x}_n$, it only changes the order of the features. So $\mathbf{t}_{ln}$ has the same dimension and values as $\mathbf{x}_n$, but permuted based on the $\mathbf{P}_l$ matrix. Using the newly introduced vectors $\mathbf{t}_{ln}$, we can write all L classification problems in one formula as follows:

$$\min_{\mathbf{w}} \frac{L}{2}\|\mathbf{w}\|^2 + C\sum_{l=1}^{L}\sum_{n=1}^{N}\xi_{ln} \quad \text{s.t.} \quad z_{ln}(\mathbf{w}^T\mathbf{t}_{ln} + b_l) \geq 1-\xi_{ln},\ \xi_{ln}\geq 0,\ \begin{cases} n=1,\dots,N \\ l=1,\cdots,L. \end{cases} \tag{4}$$

This looks very similar to the SVM problem, where $\mathbf{w}\in\mathbb{R}^D$ is the weight vector that we try to learn and we have NL input points $\mathbf{t}_{ln}$ with labels z_{ln}. The only difference is the bias of SVMs: in this formulation we have to learn L different biases while in the traditional SVM only one bias exists. We augment the weight vector $\mathbf{w}$ with the bias vector $\mathbf{b}$ and write it as $\overline{\mathbf{w}} = [\mathbf{w};\mathbf{b}] \in \mathbb{R}^{D+L}$. We also augment each of the inputs $\mathbf{t}_{ln}$ by $\mathbf{e}_l$ and write it as $\mathbf{y}_{ln} = [\mathbf{t}_{ln};\mathbf{e}_l] \in \mathbb{R}^{D+L}$, where $\mathbf{e}_l \in \mathbb{R}^L$ has 1 in the lth element and zeros everywhere else. Now we can rewrite eq. (4) as :

$$\min_{\mathbf{w},\mathbf{b}} \|\mathbf{w}\|^2 + \frac{2C}{L}\sum_{l=1}^{L}\sum_{n=1}^{N}\xi_{ln} \quad \text{s.t.} \quad z_{ln}([\mathbf{w};\mathbf{b}]^T\mathbf{y}_{ln}) \geq 1-\xi_{ln},\ \xi_{ln}\geq 0,\ \begin{cases} n=1,\dots,N \\ l=1,\cdots,L. \end{cases} \tag{5}$$

This is now an SVM problem, with NL inputs $\mathbf{y}_{ln}$ and labels z_{ln}. To see the equivalence between eq. (4) and eq. (5), note that $[\mathbf{w};\mathbf{b}]^T\mathbf{y}_{ln} = \mathbf{w}^T\mathbf{t}_{ln} + b_l$. The only difference between eq. (5) and the more standard SVM formulation is that the first term (inverse of the margin) is defined over the first D elements of the weight vector $\overline{\mathbf{w}} = [\mathbf{w};\mathbf{b}]$, not over all the elements.

To summarize, we start with L classification problems as given in eq. 3, each of them defined over N training points, but which are coupled through the circulant weight matrix $\mathbf{W} = \operatorname{circ}(\mathbf{w})$. We convert this classification problem into one maximum margin classification problem over the vector $\mathbf{w}$, with an enlarged dataset of NL points and labels (eq. (5)). The new points are L different permutations of the original points $\mathbf{X}$ and the labels are the columns of the code matrix $\mathbf{Z}$.

Advantages of our circulant SVM over the optimization in the frequency domain. CBE [13] minimizes the L classification problems of eq. (3) in the frequency domain. The main issue of the CBE is that it always needs a binary matrix of size $N\times D$: each point has to have $L = D$ labels. For $L < D$, CBE adds $D-L$ labels of zero to all the points to make the code vector D-dimensional. This means that CBE returns suboptimal solutions for $L < D$. If $L \ll D$, the number of zeros in the labels becomes much more than the original labels, and the results become much worse.

Our proposed method always returns the optimal solution, even for the case of $L < D$. It formulates the classification problem as a maximum margin classification, which is a convex quadratic program. There are libraries available that solve SVM problems for a large number of points in a few seconds. Our experiments confirm that our circulant SVM always performs better than CBE.

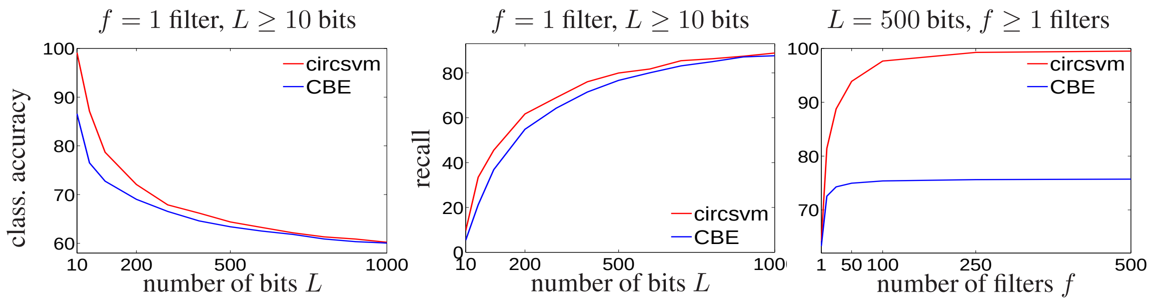

Figure 1: Comparison of our method circsvm with CBE [13]. Both methods are trained using $5\,000$ points of the CIFAR10 dataset. f is the number of circulant matrices (filters) used in solving the classification problems. *Left panel:* average classification accuracy of eq. (3). *Right and middle panels:* performance of the methods in the image retrieval task. circsvm finds the optimal solution and achieves better classification and retrieval results than the CBE.

4 Experiments

We use the CIFAR10 dataset [5] in our experiments, which contains $60\,000$ images. We consider $58\,000/2\,000$ images as the training/test set. We extract $D = 4\,096$ VGG network [11] features, which are the output of the last fully connected layer of the VGG network. Following the experiments of [13], the ground-truth set for each image consists of the first 10 nearest neighbors of the image in the original high-dimensional space. The retrieved set for each image consists of its k nearest neighbors in the Hamming space.

We make data points zero mean before learning the classifiers or hash functions. We train the SVM classifier using the VLFeat [12] library, which uses stochastic gradient descent in the optimization. We set the parameter λ of this library to 0.01 and the number of epochs to 10. It takes around 5 minutes to solve $L = 500$ classification problems, each defined over $N = 5\,000$ points.

We select a random subset of $5\,000$ points from the CIFAR10 as the input $\mathbf{X}$. We consider the output of ITQ [3] as the binary matrix $\mathbf{Z}$. Then, given $\mathbf{X}$ and $\mathbf{Z}$ as the input and the labels, we train a classifier with the circulant weights using our proposed method and CBE [13].

Our circulant SVM improves the classification accuracy. In the left panel of fig. 1, we report the average classification accuracy of L classification problems in eq. (3). We use $f = 1$ circulant matrix (filter) and change the number of bits from $L = 10$ to $1\,000$. We can see that circsvm performs better than CBE, specially for smaller number of bits. The reason is that circsvm finds the optimal solution (for any value of L), but CBE finds suboptimal solutions for $L < D$.

Better classification leads to better image retrieval (hashing) results. Now, we use the classifiers in the hashing setting. In the middle (right) column of fig. 1, we increase the number of bits L (filters f) and report the recall. The recall of circsvm is always above the recall of CBE, for different values of bits and filters. This is consistent with what we have seen in the classification results. While the recall of circsvm improves as we increase the number of filters, for CBE the recall goes down massively. The reason is that increasing the number of filters means learning circulant functions on smaller number of bits. CBE adds a massive number of zeros to the labels, which makes the inputs and labels of different hash functions very similar to each other. In this case, the L hash functions can end up being very similar to each other, which leads to losing diversity among them. As investigated in [10], lack of diversity in the set of hash functions leads to poor retrieval results. We can see that this happens for CBE in fig. 1.

5 Conclusion

Using a circulant matrix as the weight matrix of a hash function makes the computation of the binary codes very fast, $\mathcal{O}(D \log D)$ for D-dimensional inputs. This is very helpful in binary hashing, where the goal is fast image search. We showed that a previous method that learns the circulant matrix by optimizing in the frequency domain is suboptimal and its results become the more inaccurate the smaller the number of desired bits L is in the hash function. We also proposed to learn the circulant matrix in the original domain, by formulating the L classification problems using a circulant matrix as one maximum margin classification problem. This leads to a convex quadratic program whose optimal solution can be found efficiently for any desired number of bits L. This in turn gives better results in the retrieval task.

References

[1] M. Á. Carreira-Perpiñán and R. Raziperchikolaei. Hashing with binary autoencoders. In *Proc. of the 2015 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'15)*, pages 557–566, Boston, MA, June 7–12 2015.

[2] T. Ge, K. He, and J. Sun. Graph cuts for supervised binary coding. In *Proc. 13th European Conf. Computer Vision (ECCV'14)*, pages 250–264, Zürich, Switzerland, Sept. 6–12 2014.

[3] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A Procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 35(12):2916–2929, Dec. 2013.

[4] K. Grauman and R. Fergus. Learning binary hash codes for large-scale image search. In R. Cipolla, S. Battiato, and G. Farinella, editors, *Machine Learning for Computer Vision*, pages 49–87. Springer-Verlag, 2013.

[5] A. Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, Dept. of Computer Science, University of Toronto, Apr. 8 2009.

[6] G. Lin, C. Shen, D. Suter, and A. van den Hengel. A general two-step approach to learning-based hashing. In *Proc. 14th Int. Conf. Computer Vision (ICCV'13)*, pages 2552–2559, Sydney, Australia, Dec. 1–8 2013.

[7] G. Lin, C. Shen, Q. Shi, A. van den Hengel, and D. Suter. Fast supervised hashing with decision trees for high-dimensional data. In *Proc. of the 2014 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'14)*, pages 1971–1978, Columbus, OH, June 23–28 2014.

[8] A. V. Oppenheim and A. S. Willsky. *Signals and Systems*. Signal Processing Series. Prentice-Hall, second edition, 1996.

[9] R. Raziperchikolaei and M. Á. Carreira-Perpiñán. Optimizing affinity-based binary hashing using auxiliary coordinates. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 29, pages 640–648. MIT Press, Cambridge, MA, 2016.

[10] R. Raziperchikolaei and M. Á. Carreira-Perpiñán. Learning supervised binary hashing: Optimization vs diversity. In *IEEE Int. Conf. Image Processing (ICIP 2017)*, Beijing, China, Sept. 17–20 2017.

[11] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. of the 3rd Int. Conf. Learning Representations (ICLR 2015)*, San Diego, CA, May 7–9 2015.

[12] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms, 2008.

[13] F. Yu, S. Kumar, Y. Gong, and S.-F. Chang. Circulant binary embedding. In E. P. Xing and T. Jebara, editors, *Proc. of the 31st Int. Conf. Machine Learning (ICML 2014)*, pages 946–954, Beijing, China, June 21–26 2014.

[14] D. Zhang, J. Wang, D. Cai, and J. Lu. Self-taught hashing for fast similarity search. In *Proc. of the 33rd ACM Conf. Research and Development in Information Retrieval (SIGIR 2010)*, pages 18–25, Geneva, Switzerland, July 19–23 2010.